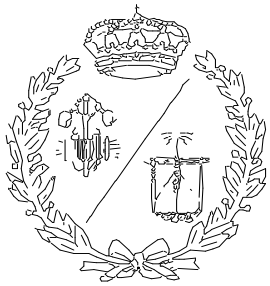
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

DISEÑO Y CONSTRUCCIÓN DE UN LEVITADOR NEUMÁTICO

DESIGN AND CONSTRUCTION OF A PNEUMATIC LEVITATOR

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Fernando Quevedo Revilla

Julio - 2022

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS:

RESUMEN

El proyecto que se presenta consiste, en el diseño y posterior construcción de un sistema de levitación neumática de bajo costo.

Dicho sistema será capaz de hacer levitar un cuerpo ligero a una altura de hasta 1m. Todo el sistema estará gobernado por un microcontrolador Arduino, a través del cual se permitirá variar la referencia y los parámetros característicos del regulador/es, tanto físicamente a través de potenciómetros junto con una pantalla, como a través de una aplicación conectándose a un ordenador vía puerto USB. Además, se incluye la posibilidad de exportar las señales para su posterior análisis. La parte mecánica del sistema será construida mediante fabricación aditiva previo modelo 3D.

El principal objetivo es el uso por parte de los futuros alumnos, en asignaturas de automática y control de la universidad de Cantabria, para afianzar conceptos mediante la puesta en práctica de lo aprendido en las sesiones teóricas y de esta manera facilitar el aprendizaje.

ABSTRACT

The project consists of the design and implementation of a low-cost pneumatic levitation system.

This system will be able to levitate a light body at a height of up to 1m. The whole system will be governed by an Arduino microcontroller, through which it will be possible to adjust the reference and the characteristic parameters of the regulator/s, both physically through potentiometers looking the selected value in a display, and through an application connecting to a computer via USB port. In addition, the possibility of exporting the signals for further analysis is included. The mechanical part of the system will be built by additive manufacturing prior 3D model.

The main objective is the use by future students, in subjects of automation and control of the University of Cantabria, to strengthen concepts by putting into practice what they have learned in the theoretical sessions and thus facilitate learning.

ACRÓNIMOS

PID: Regulador proporcional integral y derivativo.

PWM: Modulación por ancho de pulso.

USB: Bus universal de comunicación serial.

TOF: Tiempo de vuelo, generalmente aplicado a sensores.

UI: Abreviación interfaz de usuario.

SPI: Del inglés Serial Peripheral Interface, protocolo de comunicación.

TFT: Referido a la pantalla utilizadas, del inglés Thin Film Transistor.

PCB: Circuito impreso, del inglés printed circuit board.

HMI: Del inglés human-machine interface, referido a pantallas industriales.

CI: Circuito integrado.

1 INTRODUCCIÓN

El presente trabajo de fin de grado pretende construir un prototipo de levitación neumática, que haga levitar un objeto ligero. El flujo de aire que permitirá elevar el objeto será generado por un ventilador. Ya que, por su naturaleza es un sistema inestable, hay que escoger un sistema de control que permita variar el flujo de aire y con este, la fuerza de sustentación. Para realizar esta labor se ha escogido un microcontrolador ARDUINO MEGA, el cual detectará la posición del objeto a través de un sensor ultrasónico, que junto con la entrada de referencia ejecutará un algoritmo de control PID o se comunicará con un ordenador que esté ejecutando cualquier tipo de regulador.

Ya que la electrónica de control no es capaz de alimentar al actuador (ventilador), el microcontrolador enviará una señal PWM “Pulse Width Modulation” al circuito de potencia, quien será el que alimentará el ventilador en consecuencia.

1.1 MOTIVACIÓN Y ALCANCE

El control automático o automática, trata de regular, con la mínima intervención humana, el comportamiento dinámico de un sistema mediante órdenes de mando [1].

Esta necesidad de controlar de forma automática una labor o tarea, ha existido siempre, el ejemplo por excelencia que representa a los ingenieros surge en 1788 cuando James Watt desarrolló el gobernador de velocidad de las máquinas de vapor, este regulador es considerado como el primer sistema de control realimentado, pero no fue hasta 1920 cuando Nicholas Minorsky empezó a trabajar en los primeros controladores automáticos y formuló la expresión que hoy en día se conoce como PID, en su caso aplicado al ámbito de la navegación náutica. A partir de ese momento se empezó a desarrollar distintos métodos enfocados al estudio de los sistemas y métodos de control, destacando ilustres personajes como Harry Nyquist (técnicas frecuenciales), Hendrink Bode, (métodos de diseño en el dominio de la frecuencia), Nichols y Ziegler (sintonización PID), Evans (método del lugar de las raíces) [2]. Actualmente, la automática no ha sido relegada al ámbito industrial, sino que también es ampliamente utilizada en sistemas biológicos, sistemas económicos y socioeconómicos e incluso en los aparatos cotidianos. Hoy en día existe un gran abanico de controladores, de forma resumida, a continuación, se exponen las técnicas de control en los que se suelen englobar los reguladores:

1. Control todo/nada: Quizás es el más intuitivo de todos, este tipo de control se fundamenta en aplicar toda la acción cuando la salida del sistema es inferior a la

consigna, y en el caso de que la salida este por encima del valor deseado (referencia), no se aplicará ninguna acción.

2. Control analógico: Ofrecen mayor flexibilidad de control, ya que estos aplican una acción variable (no booleana).
3. Control digital: Sin duda, son los más versátiles, ya que, con el actual estado de la técnica, el llevar los reguladores a un ordenador o microcontrolador hace que las capacidades de configuración del regulador sean amplias.
4. Control de eventos discretos: Cuando sucede una acción, se actúa en consecuencia.

Como se ha comentado anteriormente, uno de los campos donde mayor uso tiene la automática es el industrial. En este campo se engloban desde grandes instalaciones de generación de energía, a aplicaciones de robótica y mecatrónica, pasando por factorías o transporte.

Las ventajas de la automática en este campo son evidentes, el sustituir un humano, por una máquina implica una importante reducción de costes, además de aumentar la seguridad de la instalación si el lugar de trabajo es un ambiente hostil. Otra de sus ventajas radica en la propia definición de la automática, el sistema al no requerir, en el caso ideal, la intervención humana, se puede dejar al sistema trabajando 24 horas al día, 7 días a la semana, lo que a efectos prácticos se traduce en una mayor productividad, pero esto no es todo, ya que al relegar la mano de obra humana a un segundo plano, la posibilidad de fallo humano es menor, por lo que, por norma general, los sistemas automatizados serán capaces de ofrecer al público un producto de mayor calidad, lo que se traduce en un mayor valor añadido, que a su vez implica una mayor rentabilidad económica.

Debido a que las aplicaciones son innumerables y que su uso está muy extendido, la automática es estudiada en muchas de las titulaciones de ingeniería.

La docencia de la automática comúnmente suele ser clasificada por parte de los alumnos como árida, ya que en esta rama hay muchos conceptos que, si no se comprenden convenientemente, resulta imposible seguir las explicaciones posteriores. Por este motivo, cuando un alumno no está familiarizado con conceptos básicos de la rama de ingeniería de control, generalmente una manera de afianzar esos conceptos se consigue mediante la puesta en práctica de lo aprendido en el aula. La manera de realizar esto, bien puede ser lograda, mediante simulaciones software o bien mediante el uso de equipos didácticos físicos enfocados a la docencia.

Respecto a las herramientas software que más se adaptan a este uso y que se utilizan a lo largo del grado son las siguientes:

- MATLAB:

Este programa, no está enfocado únicamente al uso de automática, y este es su punto fuerte, ya que, al estar preparado para tareas de análisis de datos, desarrollo hardware, visión por computador, robótica e incluso inteligencia artificial, ofrece la posibilidad de combinar todas sus utilidades en un mismo entorno. El MATLAB es un software diseñado específicamente para ingenieros y científicos, el lenguaje de programación es similar a cualquier otro, salvando las diferencias en lo que respecta a la sintaxis. Respecto al entorno de programación al igual que sucede con el leguaje, es similar a otros entornos de desarrollo software.

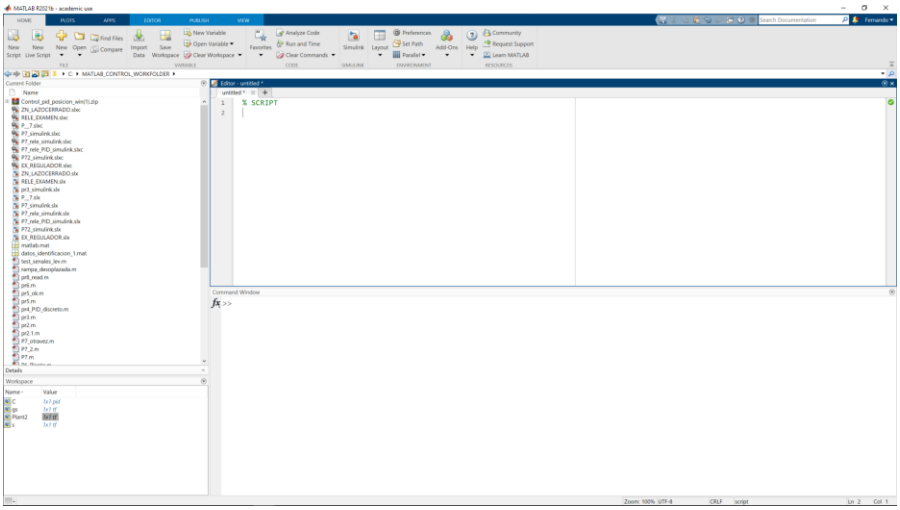


Figura 1.1: Entorno software MATLAB.

Otro de los puntos fuertes de este entorno, es su gran número de funciones y aplicaciones que tiene disponibles. Esto hace que el código se simplifique bastante. Respecto a las aplicaciones relacionadas con la automática, se destacan las siguientes:

-Systemidentification: Esta herramienta permite realizar distintos modelos matemáticos de sistemas, tanto lineales como no lineales.

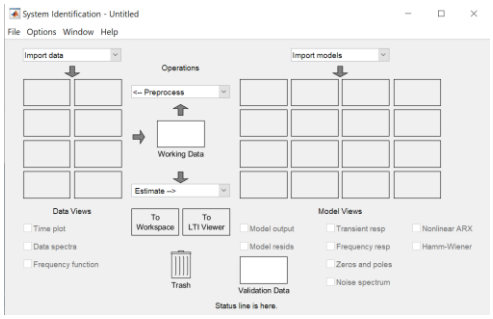


Figura 1.2: Herramienta "SystemIdentification" de MATLAB.

-PID Tunner: Permite sintonizar un regulador del tipo PID de forma sencilla, tan solo hay que especificarle las características deseadas del sistema.

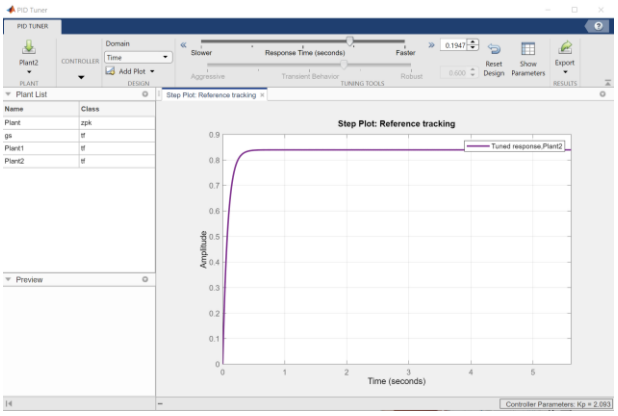


Figura 1.3: Herramienta "PID Tunner" MATLAB.

-Fuzzy Logic Designer: Cuando no se conoce el modelo matemático o es muy complejo calcularlo, otro tipo de control que se suele aplicar a estos casos es el control borroso o difuso. MATLAB tiene una aplicación que permite realizar este tipo de control.

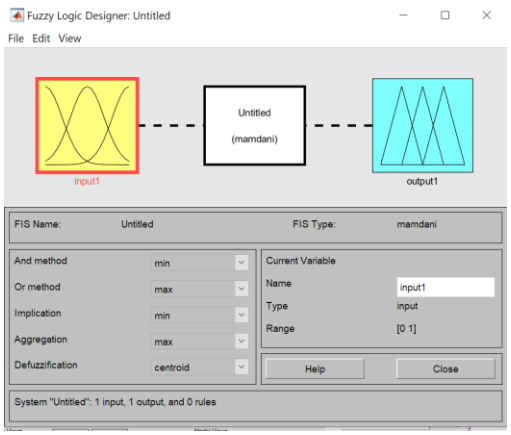


Figura 1.4: Herramienta de control fuzzy o difuso.

Pero sin duda una de las utilidades que hace que MATLAB sea muy aconsejable para estudiantes de ingeniería, es su funcionamiento conjunto con Simulink. Simulink es un entorno de programacion mediante diagramas de bloques (sin necesidad de escribir código), lo cual lo hace idóneo para iniciarse en la automática.

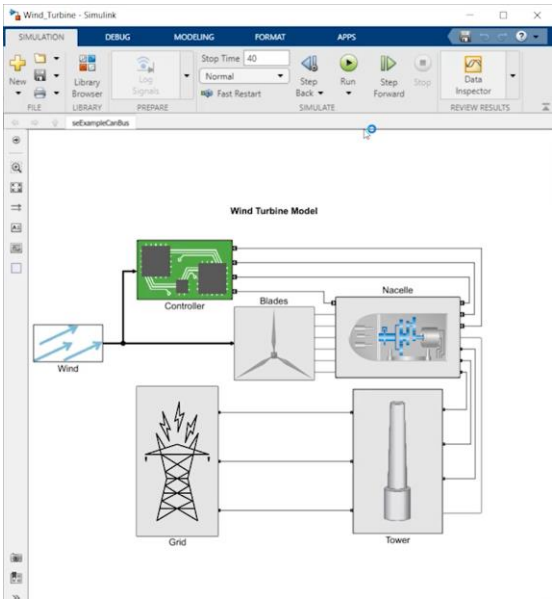


Figura 1.5: Modelado de un aerogenerador en el entorno Simulink [3].

- OpenModelica:

Es muy similar al entorno SIMULINK de MATLAB, permite modelar, diseñar y simular sistemas de cualquier naturaleza (eléctrica, mecánica, hidráulica …) al igual que MATLAB. La principal desventaja frente al primero es que no es tan completo, por otro lado, es un software libre (no sujeto al pago de cuota), otra de las ventajas es capaz de trabajar con otros programas comerciales de diseño y simulación.

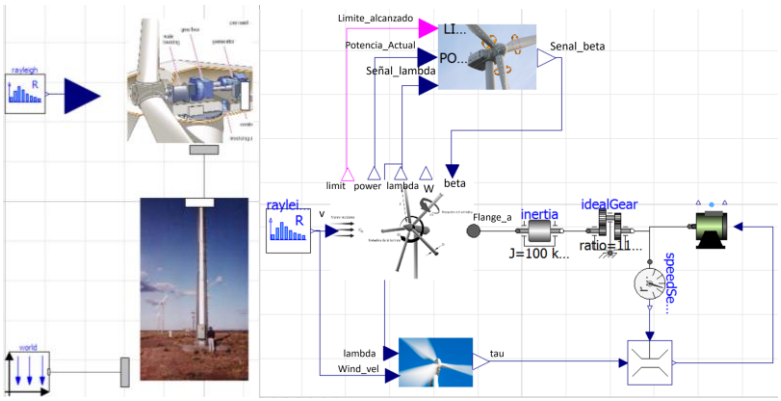


Figura 1.6: Modelado de un aerogenerador en MODELICA.

Estas herramientas software son especialmente útiles cuando el alumno ya tiene conocimientos y está acostumbrado a la interfaz de usuario de cada programa, ya que para comprender mediante simulaciones la automática es necesario conocer el funcionamiento del programa. Por esta razón, la metodología de enseñanza que se suele aplicar es la puesta en práctica físicamente en una planta y posteriormente simularlo, y realizar modificaciones con

una herramienta software, a pesar de que, en el mundo laboral, lo que se hace es diseñarlo, simularlo y finalmente construir un prototipo físico (orden inverso).

Con esta finalidad se introdujeron plantas de control enfocadas al entorno académico.

Actualmente la Universidad de Cantabria cuenta con varios tipos de plantas con el fin de poder mostrar a los alumnos distintos ejemplos prácticos.

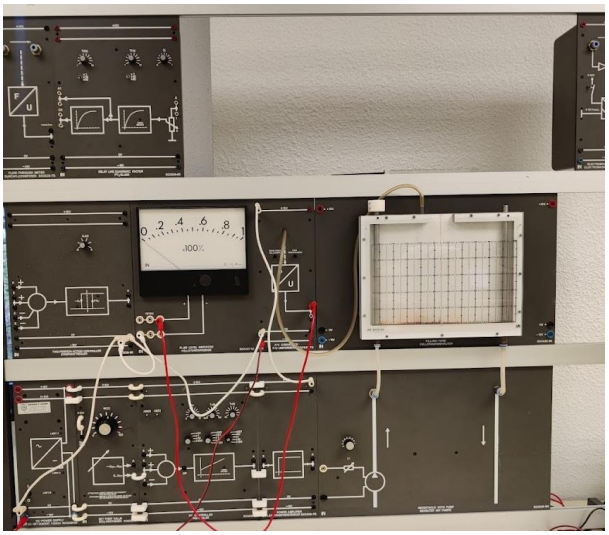


Figura 1.7: Planta de control de nivel de la Universidad de Cantabria, fabricante LUCAS NÜLLE.

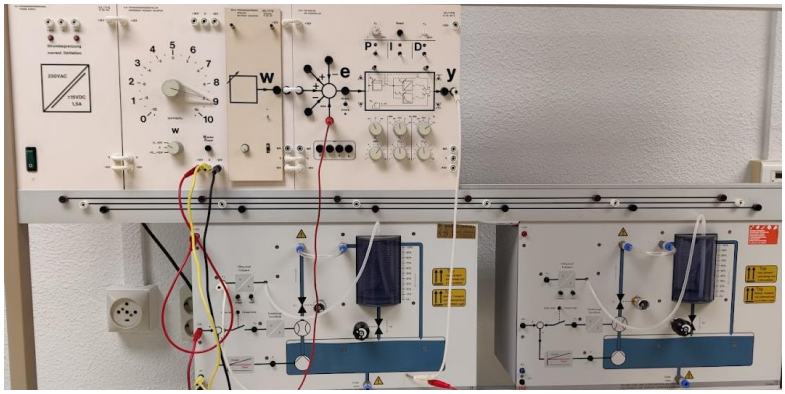


Figura 1.8: Planta de control de nivel de la Universidad de Cantabria más reciente, fabricante LUCAS NÜLLE.

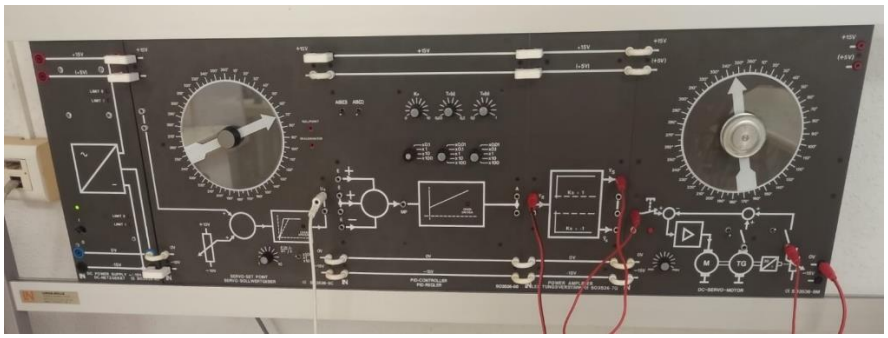


Figura 1.9: Planta de control de posición de la Universidad de Cantabria, fabricante LUCAS NÜLLE.

El funcionamiento de estas plantas es sencillo, disponen de varios módulos, usualmente el bloque de entrada, bloque de regulador, etapa de potencia, bloque de salida. Las figuras 1.7 y 1.8 son correspondientes a un control de nivel de un depósito, mientras que la figura 1.9 se corresponde con una planta de control de posición. Las plantas comerciales enfocadas al uso académico actuales son algo distintas a las actuales del laboratorio, ya que muchas incluyen aplicaciones de escritorio para poder realizar el control y visualizar las señales (tarea que actualmente se realiza a través de una tarjeta de adquisición de datos externa).

Figura 1.10: Planta comercial, de temperatura, caudal, presión y nivel, Fabricante Bytronic [4].

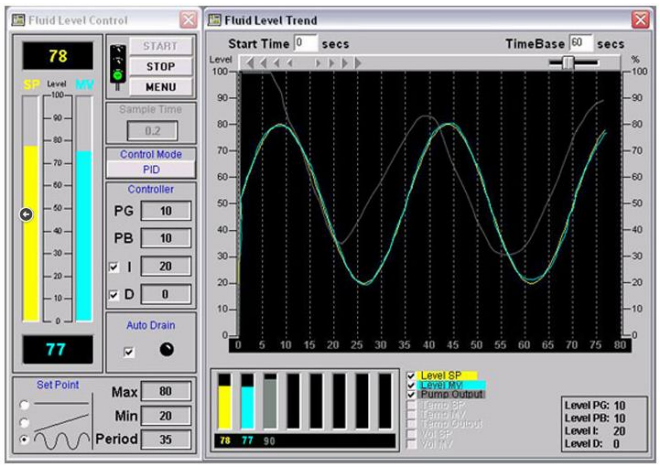


Figura 1.11: Interfaz software, asociada al manejo de la planta de la figura 10 [4].

A menudo este tipo de plantas de entrenamiento son caras, por lo que lo que se pretende, es diseñar e implementar otro sistema de otra naturaleza distinta, a un coste menor que un producto comercial equivalente.

1.2 OBJETIVOS Y METODOLOGÍA

El objetivo principal de este trabajo de fin de grado es dotar a los futuros estudiantes de la universidad de Cantabria con un sistema de control moderno a través del cual experimentar y poner en práctica conceptos como los relacionados con la teoría de control clásico.

Así mismo los objetivos que debe satisfacer la planta son los que se exponen a continuación:

1. Económico: No tendría sentido realizar un diseño de un sistema, cuyo presupuesto excediese el coste de una planta comercial equivalente.
2. Comunicación USB: El sistema deberá incluir un USB a partir del cual los alumnos puedan conectarse a la planta para realizar ajustes, visualizar y exportar las señales a otros entornos software.
3. Autónomo: Debe ser capar de trabajar de forma aislada de un ordenador, por lo que debe incluir un conjunto de pulsadores, interruptores y potenciómetros a partir de los cuales poder interactuar con la planta. Además, derivado de esta característica debe de ser independiente desde el punto de vista de la alimentación, por lo que se debe integrar un sistema de alimentación.
4. Intuitiva: Su uso debe ser lo más sencillo posible, ya que como se ha comentado anteriormente está enfocado hacia un público no experto en la rama de ingeniería de control.
5. Compacta y estética: Al tratarse de un producto final, que será utilizado por los estudiantes, se debe cuidar tanto las dimensiones, como aspectos relativos a la construcción, materiales utilizados, ajustes etc.
6. Visual: Se debe poder ver el estado del sistema visualmente de forma sencilla.
7. Actualizable y reparable: El sistema no podrá estar condicionado por lo componentes actuales, es decir debe de estar previsto la posibilidad de incluir más periféricos o sustituir cualquier componente cuando agote su vida útil por un recambio equivalente (reparable).
8. Posibilidad de incluir cualquier tipo de regulador: Al estar enfocado a un uso didáctico, la posibilidad de poder cargar varios tipos de controladores hace que el sistema no esté condicionado por el controlador precargado, y de esta manera facilita el poner en práctica el regulador que se esté explicando en el aula.

9. Reciclaje de viejos componentes: Sin dejar de lado el apartado medioambiental se intentará aprovechar componentes de algún aparato al que se le haya acabado su vida útil.

Previamente al diseño de la planta que finalmente se construiría, se valoraron plantas de diversa naturaleza. La planta que se pretendía construir inicialmente era un levitador magnético, que por medio de una bobina crease un campo magnético que hiciese levitar un pequeño objeto ferromagnético. Tras realizar la fase de estudios previos, se calculó el tamaño de la bobina para hacer levitar una esfera metálica del tamaño de una canica, 5 cm y a priori parecía una opción viable. Antes de realizar el diseño se decidió realizar un pequeño prototipo para comprobar si los cálculos se correspondían fielmente a la realidad. Para la construcción de ese prototipo, en base al objetivo 9, para la construcción de la bobina se utilizó un devanado de alta tensión de un aparato microondas. Cuando se estaba validando el prototipo, se nota una merma respecto a la altura que debería levitar, haciendo que de cara a los estudiantes no sea visual.

Figura 1.12: Primer prototipo de levitador magnético.

La otra planta que se valoró fue la que finalmente se desarrolló y la que se expone en el presente documento. La elección de la planta se desarrolló teniendo en cuenta los objetivos previos.

1.3 ESTRUCTURA DEL DOCUMENTO

El presente documento está dividido en varios capítulos. En cada uno de ellos se expondrán detalladamente los procedimientos seguidos para el diseño e implementación de cada parte del proyecto. A continuación, se expone una breve explicación de cada capítulo:

Capítulo 2: SISTEMAS DE CONTROL

Se explican brevemente los sistemas de control, realizando una comparación entre los sistemas realimentados y los de lazo abierto.

Capítulo 3: CONTROL PROPORCIONAL, INTEGRAL Y DERIVATIVO

Este capítulo está dedicado al control proporcional, integral y derivativo, tanto su expresión como su implementación.

Capítulo 4: SISTEMA DE CONTROL DESARROLLADO

Se explicará en detalle cada aspecto de diseño y constructivo de la planta desarrollada. A su vez está dividida en varios capítulos:

Capítulo 4.1: ESTRUCTURA MECÁNICA

En este capítulo se detallan los aspectos constructivos que se han seguido para el diseño de la estructura de la planta. También se valoran los distintos métodos de fabricación. Además, se podrá encontrar los modelos 3D de las piezas que conforman la estructura.

Capítulo 4.2: ELECTRÓNICA DE POTENCIA

Este capítulo está dedicado a las fases de diseño y construcción de un circuito que será capaz de alimentar todos los componentes de la planta. También se explica la electrónica de potencia asociada al control del ventilador.

Capítulo 4.3: MODELO MATEMÁTICO

Se mostrará el proceso de modelado del sistema con el fin de obtener una función de transferencia que aproxime el comportamiento del sistema.

Capítulo 4.4: CONTROL

Se diseñará el apartado hardware del sistema de control (microcontrolador, sensores…). Se tendrán en cuenta varios aspectos para la selección del equipo principal y del sensor.

Capítulo 4.5: INTERFAZ

Se muestran las dos interfaces de la planta. Por un lado, la física en la que se detallarán los elementos de los que se compone, cómo interactúan entre sí, así como su funcionamiento. Mientras que la otra forma de interactuar con la planta es a través de una aplicación basada en Python, también se mostrarán las herramientas software utilizadas.

Capítulo 4.6: SOFTWARE

Se explicará a grandes rasgos el funcionamiento de sendos programas, Python y Arduino.

Capítulo 5: RESULTADOS

En este capítulo se mostrarán los resultados de cada parte del trabajo, es decir, los resultados finales de la estructura, de la electrónica, control, así como datos que se han obtenido mediante el uso de la propia aplicación.

Dentro de este capítulo también se detallan los procesos llevados a cabo para obtener un modelo que se asemeje al comportamiento de la planta construida. Se intentará realizar un modelo lineal, un modelo no lineal con parámetros lineales y finalmente se intentará hacer lo propio con una red neuronal.

Capítulo 6: PRESUPUESTO

Se incluirá una lista de los materiales utilizados, así como su importe y el presupuesto general de la planta.

Capítulo 7: CONCLUSIONES Y TRABAJOS FUTUROS

Se expondrán las conclusiones tras finalizar el proyecto, así como las capacidades que se han adquirido a lo largo del desarrollo del proyecto. También se propondrán mejoras a realizar para futuros trabajos.

Al final del documento, se encuentran los anexos a los que se hace referencia a lo largo del documento.

2 SISTEMAS DE CONTROL

Previamente, ya se han comentado los tipos de reguladores, pero todavía no se ha explicado los elementos que conforman un sistema de control. En primer lugar, está obviamente el proceso o planta que se pretende controlar. Para influir sobre la planta, se utiliza un actuador quien será el encargado de traducir la señal de control a otra que haga modificar la salida de la planta. Esa señal de control generalmente proviene de un regulador (de cualquier tipo de los anteriores comentados). A estos tres elementos, se les conoce como cadena directa del sistema.

2.1 REALIMENTACIÓN

Ahora bien, dependiendo de si el valor de la salida influye al sistema se pueden distinguir entre dos tipos de controles:

2.1.1 Control en lazo abierto

El regulador no considera el valor de la salida para calcular la señal de control, el diagrama de bloques es el que se muestra a continuación.

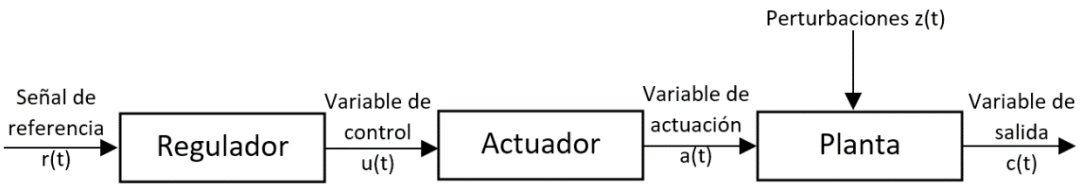


Figura 2.1: Diagrama de bloques control en lazo abierto.

La señal de referencia actúa sobre el regulador, quien excita el actuador, influyendo de esta manera sobre la planta con el fin de buscar la salida deseada. El principal inconveniente de esta arquitectura es que es muy sensible frente a perturbaciones.

2.1.2 Control en lazo cerrado

En este caso la salida del sistema se mide mediante un transductor y esta medida se compara con la consigna o referencia, siendo la diferencia entre ambas la señal de error, y es esta

quien actúa sobre el regulador, cuya salida de igual manera que anteriormente excita al actuador para finalmente influir a la planta.

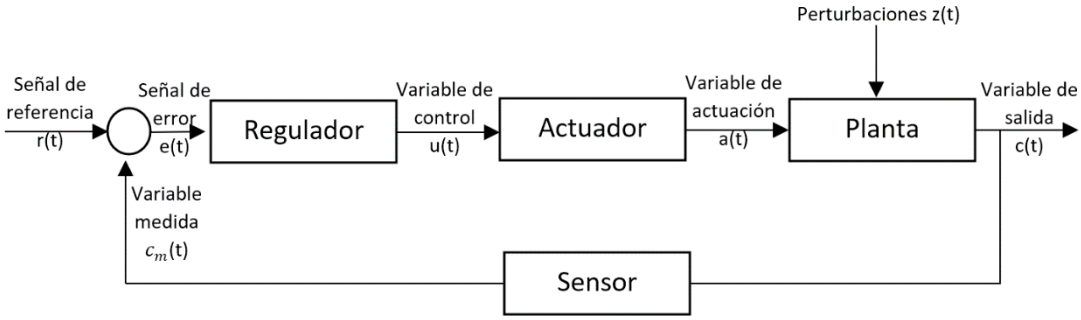


Figura 2.2: Diagrama de bloques de control en lazo cerrado.

De este modo se consigue un sistema más robusto frente a perturbaciones, ya que las perturbaciones que han influido sobre la planta son tenidas en cuenta al comparar la referencia con la salida.

Por esta razón usualmente se suele trabajar con sistemas de control en lazo cerrado.

3 CONTROL PROPORCIONAL, INTEGRAL Y DERIVATIVO

La finalidad de todo regulador es intentar modificar la respuesta tanto transitoria como en régimen permanente del sistema. De todos los tipos de reguladores que se han comentado previamente, hay uno que destaca por su porcentaje de uso, y no es otro que el regulador PID (utilizado alrededor del 90% de los procesos). El acrónimo PID, hace referencia a las tres acciones de control, proporcional, integral y derivativa. El comportamiento de este tipo de regulador se rige por el siguiente modelo matemático [5]:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) \cdot d\tau + K_d \cdot \frac{de(t)}{dt} \qquad (3.1)$$

El primer sumando se corresponde con el término correspondiente a la acción proporcional, ya que su contribución a la señal de control es proporcional al error. El segundo sumando se corresponde con la acción integral por razones evidentes, y finalmente el último sumando, se corresponde con la acción derivativa.

Llevando esa expresión al dominio de Laplace, resultaría la siguiente expresión [5]:

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) \cdot d\tau + K_d \cdot \frac{de(t)}{dt} \qquad (3.2)$$

3.1 ACCIÓN PROPORCIONAL

La salida es proporcional al error (diferencia consigna-salida), siendo la constante de proporcionalidad K_p.

$$u(t) = K_d \cdot \frac{de(t)}{dt} \qquad (3.3)$$

$$U(s) = K_d \cdot s \cdot E(s) \qquad (3.4)$$

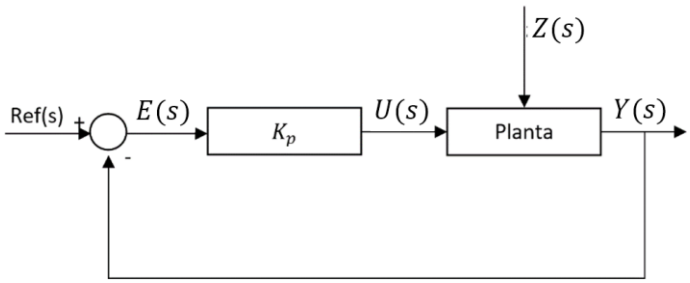


Figura 3.1: Diagrama de bloques, acción proporcional.

El efecto que tiene aumentar la constante de proporcionalidad es que el sistema va a ser más rápido y además reduce el error en estado estacionario, en cambio se sacrifica el transitorio, ya que, al aumentarla, aumenta la magnitud de las oscilaciones, llegando incluso a volver al sistema inestable.

3.2 ACCIÓN INTEGRAL

La constante K_i es conocida comúnmente como ganancia de la acción integral.

$$u(t) = K_i \cdot \int_0^t e(\tau) \cdot d\tau \qquad (3.5)$$

$$U(s) = \frac{K_i}{s} \cdot E(s) \qquad (3.6)$$

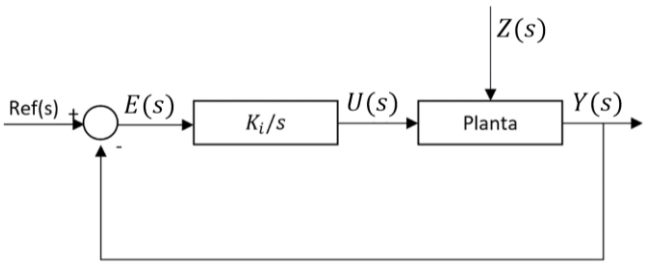


Figura 3.2: Diagrama de bloques, acción integral.

Los efectos de incluir una acción integral al regulador hacen que se elimine el error en régimen permanente. Como principal desventaja, al aumentar la ganancia integral, aumentan las oscilaciones en el transitorio.

3.3 ACCIÓN DERIVATIVA

Aportará una acción proporcional a la derivada del error, dicha ganancia proporcional se la denomina K_d o ganancia de la acción derivativa.

$$u(t) = K_d \cdot \frac{de(t)}{dt} \qquad (3.7)$$

$$U(s) = K_d \cdot s \cdot e(s) \qquad (3.8)$$

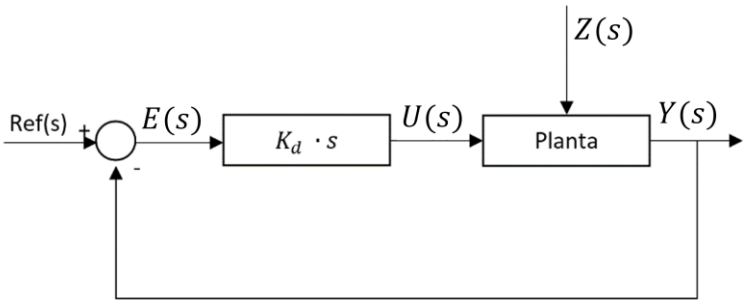


Figura 3.3: Diagrama de bloques, acción derivativa.

La acción derivativa hace que se disminuyan las oscilaciones en el transitorio, ya que esta parte dota al regulador de una anticipación del comportamiento futuro. La manera de anticiparse es únicamente calculando la derivada de la curva del error en cada instante. Su principal desventaja, está relacionada con la amplificación de las perturbaciones.

3.4 PID CONTINUO

Como se ha comentado, la variación de cualquier ganancia (proporcional, integral o derivativa), modifica la respuesta del sistema, la parte complicada de todo esto es al juntar todas las acciones, el buscar unos valores óptimos para cada ganancia. El encontrar esos valores a menudo es una tarea ardua, especialmente si se realiza manualmente, por este motivo existen distintos métodos para obtener esas ganancias (sintonización del PID).

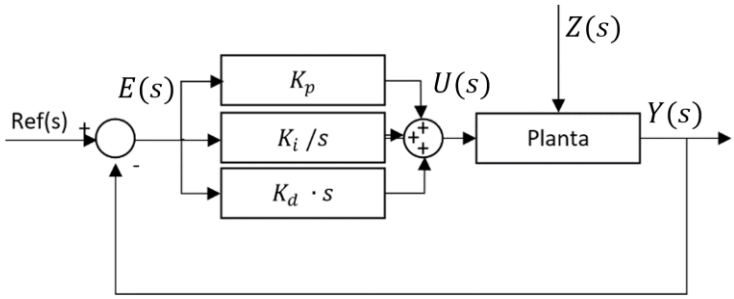


Figura 3.4: Diagrama de bloques, control en lazo cerrado con regulador PID.

3.5 PID DISCRETO

Una vez comprendido el funcionamiento del PID continuo resulta más fácil comprender el PID discreto. Existen multitud de métodos para discretizar reguladores, uno de los métodos más utilizados es el método de integración hacia atrás. A efectos prácticos únicamente hay que realizar la siguiente sustitución.

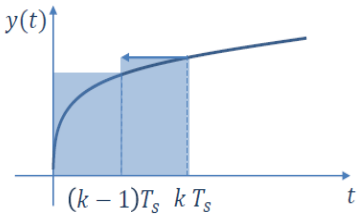


Figura 3.5: Representación gráfica, integración hacia atrás.

$$s = \frac{z-1}{T_s \cdot z} \qquad (3.9)$$

Por tanto, discretizando por separado cada acción del PID, se obtienen las siguientes expresiones:

$$U_p(s) = K_p \cdot E(s) \rightarrow U_p(z) = K_p \cdot E(z) \qquad (3.10)$$

$$U_i(s) = \frac{K_i}{s} \cdot E(s) \rightarrow \ U_i(z) = \frac{K_i}{\frac{z-1}{T_s \cdot z}} \cdot E(z) = \frac{K_i T_s \cdot z}{z-1} \cdot E(z) \qquad (3.11)$$

$$U_d(s) = K_d \cdot s \cdot E(s) \rightarrow \ U_d(z) = K_d \cdot \frac{z-1}{T_s \cdot z} \cdot E(z) \qquad (3.12)$$

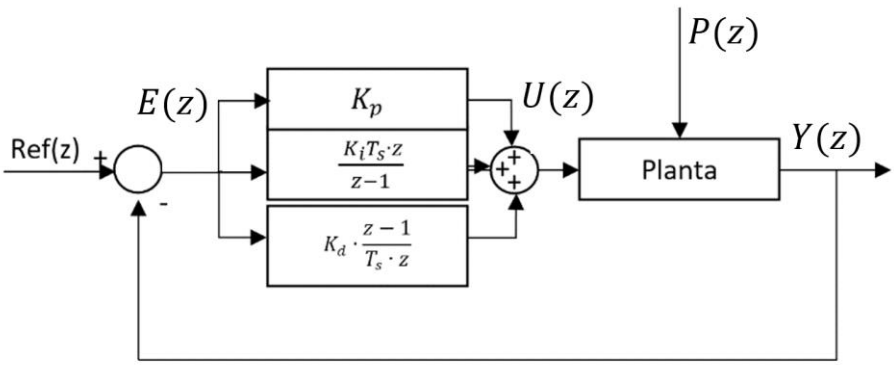


Figura 3.6 Diagrama de bloques, en el dominio Z con regulador PID

Obteniendo la ecuación en diferencias de cada sumando:

$$u_{p_k} = K_p \cdot e_k \qquad (3.13)$$

$$u_{i_k} = \sum_{0}^{t} K_i \cdot e_k \qquad (3.14)$$

$$u_{d_k} = K_d \cdot \frac{e_k - e_{k-1}}{T_S} \qquad (3.15)$$

Estas expresiones, ya son aptas para ser ejecutadas por un microcontrolador.

4 SISTEMA DESARROLLADO

A lo largo de este capítulo se detallarán los aspectos relativos a la planta escogida, se estudiará su modelo matemático, y se tratarán los aspectos tenidos en cuenta tanto para el diseño como para la construcción. El sistema está compuesto principalmente por un microcontrolador Arduino, que junto con un conjunto ventilador-sensor será capaz de hacer levitar una bola de poliestireno. Adicionalmente existen otros componentes como una fuente de tensión para alimentar el conjunto, periféricos para interactuar con el equipo (pantalla, interruptores, pulsadores, potenciómetros…) y finalmente, pero no por ello menos importante, una estructura que alberga todos los componentes, y fija el tubo de metacrilato por donde levitará el objeto. Al margen derecho, se muestra una vista general de la planta que se pretende construir.

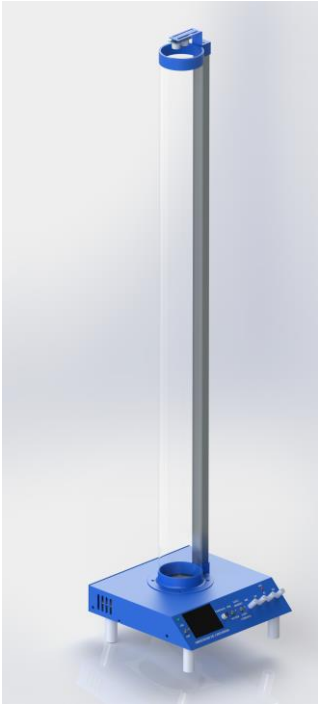


Figura 4.1: Modelo CAD de la planta que se desea implementar.

Con el fin de exponerlo de una manera más clara este, capítulo está dividido en varios subapartados que se detallan a continuación.

4.1 ESTRUCTURA

La estructura que alojará todo el sistema, además de perfectamente funcional, en base al objetivo secundario 5, deberá de ser compacta y estética, por lo que se deberá escoger algún método de fabricación cuyo resultado ofrezca un acabado de calidad, con buenos ajustes y duradero. Tras valorar distintos métodos de fabricación, finalmente se debe escoger entre fabricación manual con ocumen o fabricación aditiva por medio de una impresora 3D.

Gracias a la tecnología actual, las impresoras 3D tienen un precio asumible para el público en general, estas impresoras son capaces de conseguir fabricar un sólido mediante la colocación de varias capas finas de material una encima de otra. Solo es necesario realizar un modelo 3D en cualquier software de modelado 3D, exportarlo a un formato compatible con la impresora y escoger un material acorde a las necesidades de la pieza que se pretende fabricar (esfuerzos mecánicos, color, post-procesado…).

El precio es otro factor decisivo a la hora de tomar la decisión final, tras hacer un pequeño análisis de costes, se determina que al tener acceso a una impresora 3D, debido a las

posibilidades de personalización, acabado y el económico precio de los consumibles, la estructura se realizará mediante fabricación aditiva.

Como se ha comentado previamente, uno de los aspectos clave a la hora de realizar una impresión de una pieza que conforma un producto final, es el material del que tiene que estar hecho. Los materiales que usualmente se suelen utilizar con este tipo de equipos, son tres, ABS, PLA, PETG (también existen otras variantes mejoradas). Cada uno de ellos tiene sus características, por lo que habrá que seleccionar aquel que mejor satisfaga las condiciones del diseño. En este caso, el material escogido debe tener una rigidez alta, ya que debe soportar el peso del conjunto. Por otro lado, la resistencia térmica del material, ya que su emplazamiento será el laboratorio de control de la universidad de Cantabria (no está a la intemperie, espacio climatizado), no tendrá por qué ser alta. Adicionalmente se valora otros aspectos secundarios como la facilidad de impresión, calidad de la impresión u olor producido por el material. A continuación, se expone de manera breve las características principales de cada material.

Tabla 4.1: Comparativa materiales comunes para impresión 3D [6].

	ABS	**PETG**	**PLA**
Precio (750 g)	20-25 €	25-30 €	20-25€
Resistencia	Media	Baja	Alta
Resistencia a impactos	Alta	Media	Baja
Rigidez	Alta	Baja	Alta
Resistencia térmica	<100ºC	<80ºC	<30ºC
Temperatura de impresión	220-240	230-250	200-215
Dificultad de impresión	Alta	Alta	Baja
Calidad de impresión	Alta	Media	Alta
Olor	Mucho	Poco	Nada

Como se puede apreciar, el material que mejor se adapta a las necesidades del diseño es el PLA, pero quizás un aspecto que se debería mejorar es el de la fragilidad, por lo que, tras ver las distintas opciones en el mercado, se escoge un PLA mejorado el cual ofrece más dureza y menos fragilidad que un PLA convencional. No existe una nomenclatura normalizada para estos filamentos de PLA mejorados, cada casa lo vende con su propio nombre comercial, en el caso del PLA que se va a utilizar, el fabricante VELLEMAN, lo denomina PLA-SATÉN. Respecto al tono escogido, se ha decidido un color lo más similar posible a los tonos azulados representativos de la universidad de Cantabria.

Una vez escogido el material del filamento a utilizar, otro aspecto clave a la hora de realizar una impresión es el propio diseño de la pieza, ya que no todas las piezas son fácilmente realizables por la impresora 3D. Otro factor importante es la orientación de la pieza a la hora de imprimirla, hay que escoger una orientación y posición adecuada para reducir el uso de soportes y con ello optimizar el filamento. La estructura que se plantea principalmente está compuesta por varios elementos:

4.1.1 Alojamiento

Se trata de una caja, que albergará todos los componentes, electrónica de potencia, electrónica de control, así como el actuador. Además, servirá de soporte para otros elementos como la consola y el conducto por el que levitará la bola. Tras seleccionar todos los componentes de la planta se distribuyen de la forma más eficiente con el fin de diseñar una estructura lo más compacta posible.

Figura 4.2: Modelo CAD, Distribución en planta de los elementos principales.

Con el fin de simplificar la impresión, el alojamiento estará formado por una base y 2 tapas laterales, unidas entre sí mediante uniones atornilladas. En la figura 4.3 se muestra el renderizado 3D de dichas piezas.

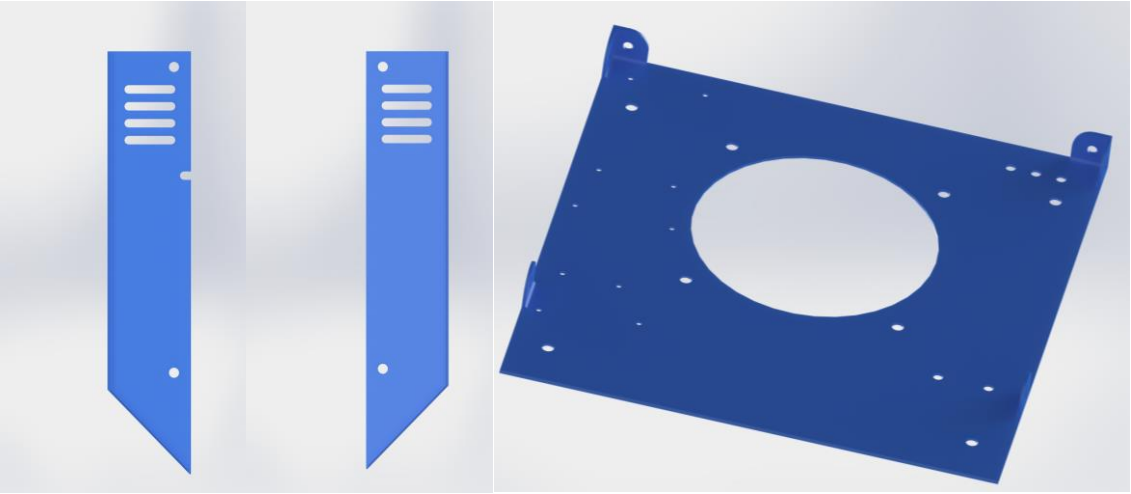


Figura 4.3: Modelos CAD, de izquierda a derecha: Tapa lateral izquierda; Tapa lateral derecha; Base principal.

Fijándose en las tapas laterales se han incluido unas aberturas para que la refrigeración pasiva de los componentes electrónicos funcione de manera óptima. También la tapa lateral izquierda tiene una pequeña ranura a través de la cual se pasará el cable USB destinado a la comunicación planta-ordenador.

Además, a este alojamiento, se le añadirá un bastidor de aluminio anodizado, con vistas que a futuro y debido al paso del tiempo y al esfuerzo de portar todos los elementos, el PLA no se deforme.

4.1.2 Soporte de lámina

Una tubería de metacrilato del diámetro y longitud deseadas, son bastante difíciles de conseguir, ya que su uso fuera de esta aplicación es bastante reducido, lo cual hace también que el precio sea elevado, perjudicando el objetivo de que fuera económico. Tras valorar distintas opciones, finalmente se ingenia un conducto formado por una lámina de metacrilato flexible fijada con unos soportes que además realizan la función de soporte para una barra hueca de aluminio de sección cuadrada, a través de la cual se pasarán los cables desde el alojamiento hasta la parte superior donde irá fijado el sensor que realizará la medida. Este sistema soporte-lámina resulta alrededor de un 90% más económico que una tubería de metacrilato de longitud y diámetro equivalente.

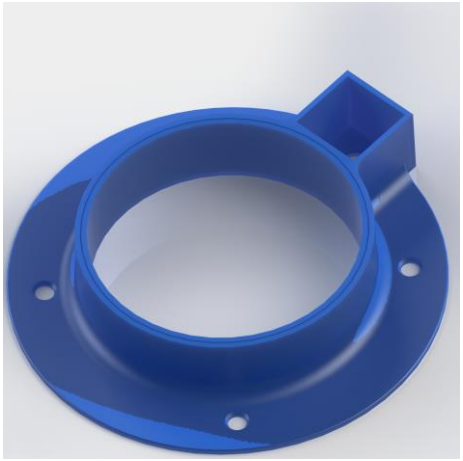


Figura 4.4: Modelo CAD, Soporte inferior de la lámina de metacrilato.

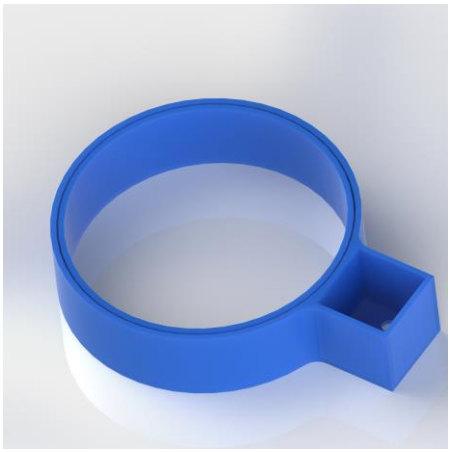


Figura 4.5: Modelo CAD, Soporte superior de la lámina de metacrilato.

4.1.3 Tapa trasera

Es la encargada de cerrar la parte trasera del alojamiento, esta tiene un orificio a través del cual pasar el cable de alimentación. Además, incluye otro orificio para colocar el interruptor general de la planta. Su unión con el alojamiento es del mismo modo, mediante tornillos. Por otro lado, esta pieza tiene escrito en relieve información que explica los peligros que conlleva el desmontaje del equipo, autoría, tensión de alimentación…etcétera.

Figura 4.6: Modelo CAD, Tapa trasera.

4.1.4 Consola

La consola es el elemento con el cual interactuará el usuario final. Esta contendrá los alojamientos destinados para la pantalla, interruptores o potenciómetros. Al tratarse del elemento que va a manejar el usuario final y que más se va a utilizar, se ha decidido incluir un refuerzo de aluminio por la parte trasera, lo que se consigue de esta forma es que al mover o tocar cualquier elemento de maniobra, el “feedback” percibido por el usuario sea de un producto sólido y robusto. La función de cada uno de los elementos se encuentra detallado en texto en relieve.

Figura 4.7: Modelo CAD, Consola.

4.1.5 Canalizador de flujo

El ventilador y el conducto no son del mismo diámetro. El ventilador tiene un diámetro mayor que el conducto, para que el actuador sea capaz de elevar solventemente la bola. Con la finalidad de aprovechar todo el flujo, se diseña la pieza que se muestra en la figura 4.8.

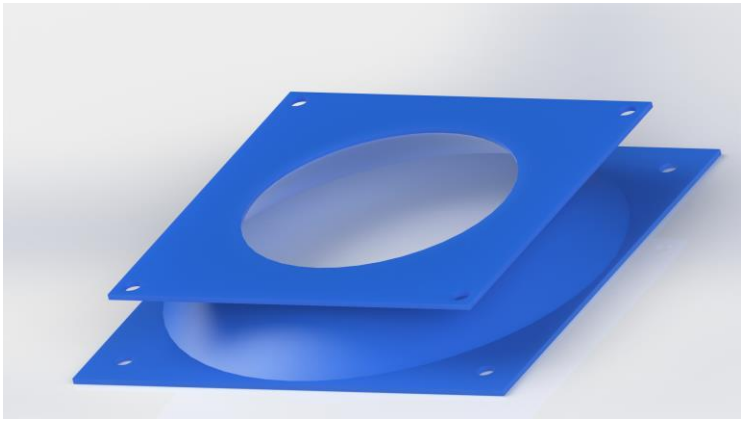


Figura 4.8: Modelo CAD, Canalizador de flujo o embudo, sin suplementos.

Los cálculos aerodinámicos a menudo son complejos, hasta el extremo de que los grandes fabricantes de automóviles testan sus prototipos de nuevos vehículos en túneles de viento, ya que es la mejor manera de obtener datos fieles a la realidad.

La idea inicial para evitar realizar cálculos complejos y simulaciones que después quizás no se correspondan con lo que sucede en la realidad, es realizar el modelo del concentrador de flujo, y posteriormente experimentar cómo se comporta el objeto al levitar y cómo le influyen las turbulencias. Tras realizar varios ensayos físicos, se llega a la conclusión de que en la zona inicial del conducto, las turbulencias eran excesivas para el funcionamiento óptimo del sistema. Para encontrar una solución a este problema, se llevaron a cabo modificaciones provisionales y reversibles sobre el concentrador de flujo inicial. Principalmente esas modificaciones consistían en añadir y fijar aletas de papel o cartón que traten de conseguir un flujo lo más lineal posible. Finalmente, tras varias pruebas se implementan los siguientes suplementos.

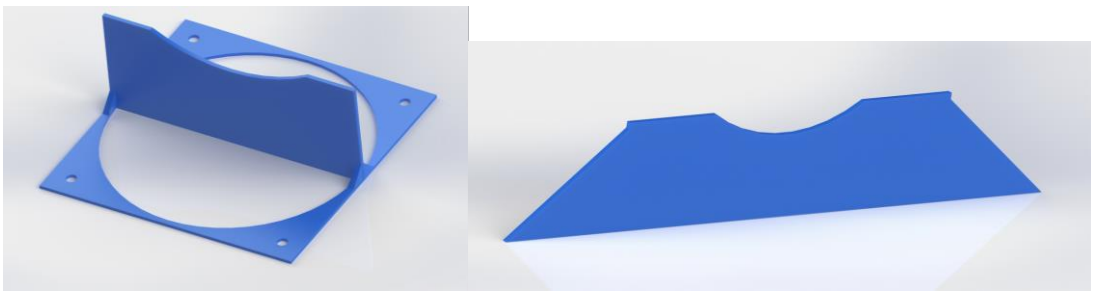


Figura 4.9: Modelo CAD, Suplementos para canalizador de flujo.

4.1.6 Tapa superior

Apoyado sobre el canalizador de flujo y las tapas laterales, se coloca la tapa superior del sistema. Esta contiene los orificios necesarios para acoplarse al canalizador de flujo mediante tornillos. Además, tiene un orificio por el cual pasa el flujo creado por el ventilador.

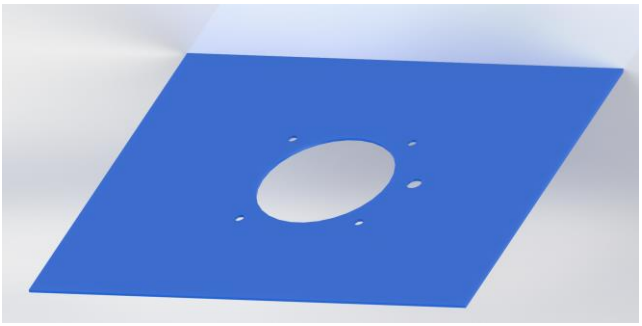


Figura 4.10: Modelo CAD, Tapa superior.

4.1.7 Soporte del sensor

Es un elemento que se acopla con el soporte superior, y sujeta el sensor. Esta pieza tiene dos ranuras para poder ajustar la posición del sensor convenientemente.

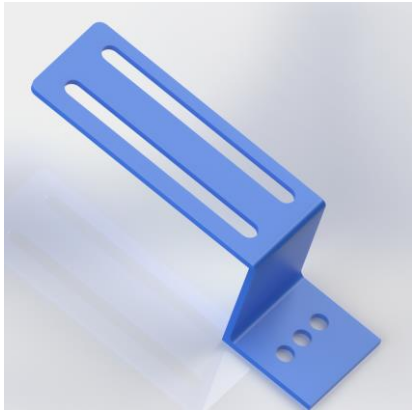


Figura 4.11: Modelo CAD, Soporte del sensor.

4.1.8 Patas

Son cuatro elementos de sección cilíndrica que elevan el alojamiento a una determinada altura para permitir que el ventilador pueda absorber el aire correctamente. Su montaje es atornillado a la base del alojamiento y al chasis metálico.

Figura 4.12: Modelo CAD, Pata.

En base al objetivo de que fuese reparable se ha pretendido evitar aquello que impida el desensamblaje del equipo como pueden ser las uniones pegadas con adhesivo o sellados con siliconas. Si se desea, está disponible en el anexo 2 los planos de cada una de las piezas, así como la vista explosionada del conjunto.

4.2 ELECTRÓNICA DE POTENCIA

A esta altura del documento ya se ha comentado que el ventilador será comandado por un microcontrolador Arduino. Como se explica en el apartado “selección del equipo principal”, Arduino carece de salidas analógicas (que ofrezcan una tensión continua) para conectar al ventilador, y aunque las tuviese el microcontrolador sería incapaz de suministrar la potencia necesaria para mover el actuador. Atendiendo a la hoja de características del ventilador (anexo 5), la potencia nominal es la siguiente:

$$P_{ventilador} = V \cdot I = 12 \cdot 0.25 = 3\ W$$

Con el fin de satisfacer el objetivo de poder funcionar de manera autónoma (únicamente ligado a una conexión a un enchufe Schuko convencional), se debe incluir una electrónica de potencia que sea capaz de alimentar tanto al ventilador, como al microcontrolador, como los periféricos (pantalla, leds…). A continuación, se desglosan las potencias de cada elemento que debe alimentar:

$$P_{Led} = V \cdot I = 2 \cdot 25 \times 10^{-3} = 50\ mW \qquad (4.1)$$

Al tener dos leds, la potencia total aproximada consumida por estos:

$$P_{Leds} = 2 \cdot 50\ mW = 100\ mW \qquad (4.2)$$

La corriente consumida por el microcontrolador Arduino MEGA, según su hoja de características ronda los 93mA.

$$P_{Arduino} = 5 \cdot 93\ \times 10^{-3} = 465\ mW \qquad (4.3)$$

La potencia necesaria para la pantalla de la consola según hoja de características es la siguiente.

$$P_{Pantalla} = 3.3 \cdot 300\ \times 10^{-3} = 990\ mW \qquad (4.4)$$

Además, se añade un margen de seguridad, para la potencia disipada por los potenciómetros, resistencias...etc. Otra razón por la que se añade un poco más de potencia es por si en el futuro se desea realizar alguna actualización, que sea posible añadirlo sin mayor inconveniente.

$$P_{seguridad} = 1\ W \qquad (4.5)$$

La potencia total resultante será pues:

$$P_{Total} = P_{Ventilador} + P_{Leds} + P_{Arduino} + P_{Pantalla} + P_{Seguridad} \qquad (4.6)$$

$$= 3 + 50 \times 10^{-3} + 100 \times 10^{-3} + 465 \times 10^{-3} + 990 \times 10^{-3} + 1 = 5.605\ W$$

El siguiente paso es buscar la tensión más adecuada para dar esa potencia. En este caso la tensión más alta es la de alimentación del ventilador, luego realizando los cálculos con una tensión de 12 V, se obtiene el siguiente resultado.

$$I_{Total} = \frac{P_{Total}}{V} = \frac{5.605}{12} = 0.467A \qquad (4.7)$$

El sistema de potencia que se pretende construir es una fuente de tensión regulada y tiene el diagrama de bloques de la figura 4.13.

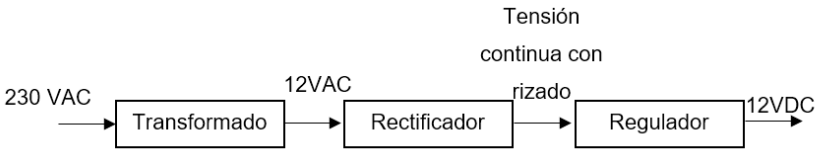


Figura 4.13: Diagrama de bloques, fuente de tensión.

Tras hacer este pequeño cálculo se empieza a seleccionar los componentes necesarios:

1.Transformador:

En virtud del objetivo medioambiental, se ha utilizado un viejo transformador proveniente de unas luminarias halógenas que debían estar alimentadas a 12 V de tensión alterna, lo que lo hace ideal para ser utilizado en este proyecto. Las características del transformador son las que se detallan en la tabla 4.2.

Tabla 4.2: Resumen características transformador.

Tensión entrada	230 VAC
Tensión de salida	12 VAC
Corriente salida	3.9 A

Como se puede apreciar satisface con creces los requisitos que se buscaban, ya que este transformador está diseñado en origen para alimentar bombillas halógenas de 12V- 50W.

Además, como se puede ver en la figura 4.2 está diseñado con un formato bastante compacto.

4.2.1 Rectificado

Para pasar de la tensión de 12 voltios de corriente alterna a continua, es necesario un rectificador. En este caso para llevar a cabo eta función se realizará un puente de diodos. En el mercado existen puentes de diodos comerciales encapsulados. Al ser un producto que se va a usar (un prototipo funcional), se va a realizar el montaje de todos los componentes sobre un circuito impreso (o comúnmente conocido como PCB), por lo que en principio no habría diferencia sustancial entre utilizar el circuito integrado o conectar cuatro diodos mediante las pistas de la PCB. Finalmente se opta por ensamblar cuatro diodos separados debido a problemas logísticos del distribuidor.

Antes de pasar a la fase de montaje, es necesario validar el diseño, en este caso se utilizará CAPTURE. Esta herramienta permite añadir el componente exacto (con la misma referencia comercial), por lo que ofrece unos resultados que se corresponden fielmente con la realidad.

A continuación, se muestra la simulación del puente de diodos diseñado:

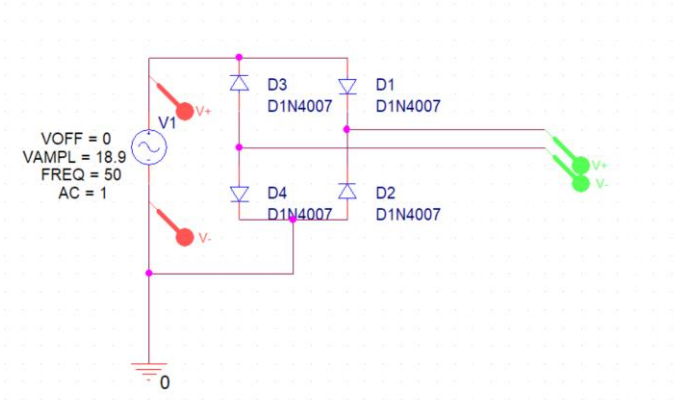


Figura 4.14: Simulación rectificador de onda completa, puente de diodos.

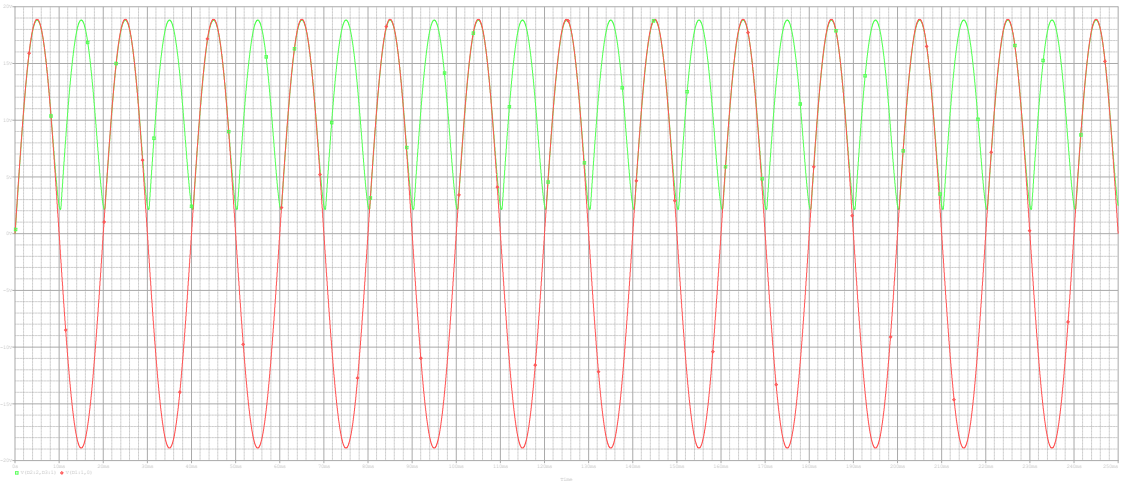


Figura 4.15: Resultado de la simulación del esquemático de la figura 4.14.

En rojo se puede ver la forma de onda de la tensión, que resultaría de la salida del transformador, mientras que en tono verde está representado la salida del puente de diodos.

Los resultados son malos en términos de rizado de la tensión, para solucionarlo se introduce un condensador electrolítico entre la salida del puente:

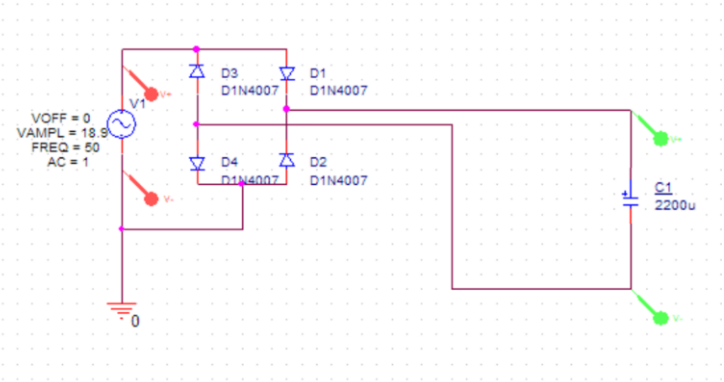


Figura 4.16: Simulación tras añadir un condensador para corregir el rizado.

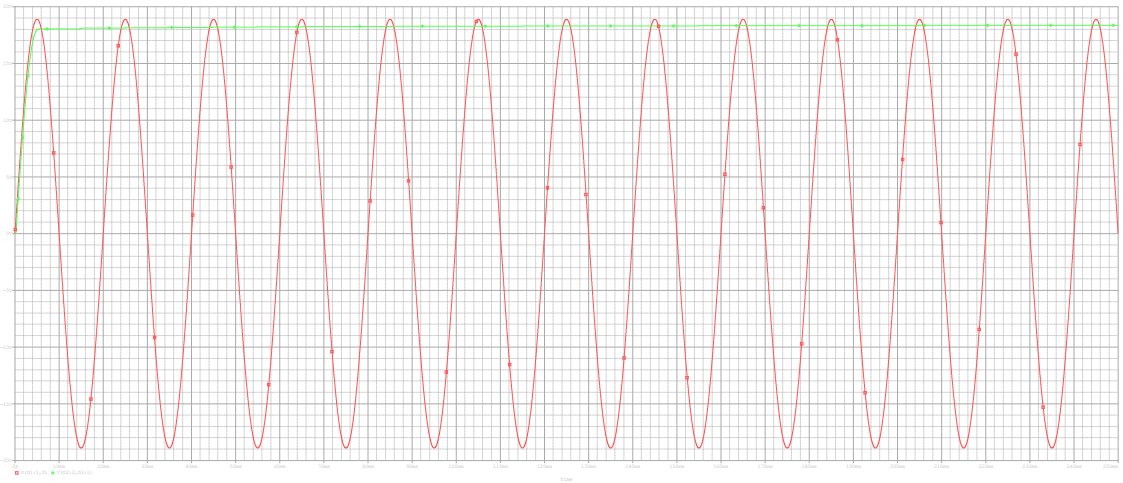


Figura 4.17: Resultados de la simulación del esquemático de la figura 4.16.

En verde se puede observar la salida del puente de diodos, al añadir un condensador de 2200 microfaradios. Lógicamente, este no va a ser el comportamiento final, ya que el circuito se encuentra en vacío, pero si se realiza el cálculo del condensador, para la corriente estimada, se llega al valor del rizado[7] :

$$C = \frac{I_o}{V_{r(pp)} \times 100} \rightarrow V_{r(pp)} = \frac{0.467}{2200 \times 10^{-6} \times 100} = 2.12 V_{pp}\ (a\ plena\ carga) \qquad (4.8)$$

En rojo igual que la figura anterior se representa la tensión de salida del transformador. La próxima etapa es la de regulación de tensión para ello se utilizarán los más que conocidos reguladores de la serie 78XX. Esta familia está compuesta por reguladores de tensión positiva que son capaces de dar una corriente máxima de 1A. Estos reguladores ofrecen una tensión continua y estabilizada, además de estar protegidos contra sobrecargas y cortocircuitos. Su funcionamiento interno está basado en el uso de transistores que proporcionan la amplificación necesaria, mientras que la estabilidad es debida al uso de diodos de tipo Zener. Visualmente, estos reguladores se comercializan en distintos encapsulados, para esta aplicación se utiliza el encapsulado TO-220. Según recomienda el fabricante del LM78XX, tanto a la entrada y a la salida, se deben colocar condensadores de valor 330 y 100 nano Faradios respectivamente. La principal función de estos condensadores, es la de filtrar los transitorios o picos de tensión (ver figura 4.18). Respecto a otras características del CI, destacar que para funcionar correctamente es necesario que al menos la tensión de entrada sea unos 2V superior a la que se debería regular.

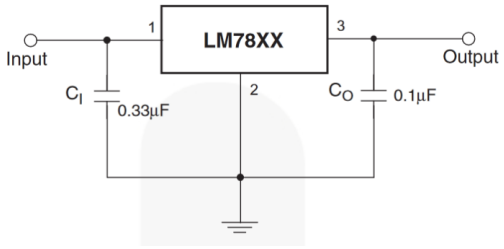


Figura 4.18 Recomendación del fabricante de los condensadores para la familia LM78XX

Conociendo que la potencia máxima que puede dispar el CI es de 15 W se puede calcular la diferencia de tensión máxima. Escogiendo un regulador de 12 V (LM7812):

$$P_{Disipada} = I_{out}(V_{in} - V_{out}) \quad (4.9)$$

$$P_{Disipada_{max}} = 15 = I_{out}(12 + \Delta V - 12) \quad (4.10)$$

Si se desea obtener 1 A en la salida:

$$15 = 1 \cdot (12 + \Delta V - 12) \rightarrow \Delta V = 15\ V \quad (4.11)$$

Es decir, como máximo se podría introducir una tensión de 27 V (12+15), en este caso, no existe este problema ya que la tensión que le llega al regulador ronda los 18 V (ver figura 4.17).

Se ha escogido un regulador de 12 V ya que es la tensión a la que se debe alimentar el ventilador. El inconveniente de esto último es que el microcontrolador Arduino MEGA, necesita de una tensión de 7-12V, lo que haría que estuviese trabajando en su zona de tensión máxima, quedando vulnerable ante un pico u oscilaciones de tensión provocados, por ejemplo, por las conmutaciones de la señal PWM. Entre otros, por este motivo en la salida de tensión de 12 V se decide conectar otro regulador de 9 V que alimentará a la placa de Arduino.

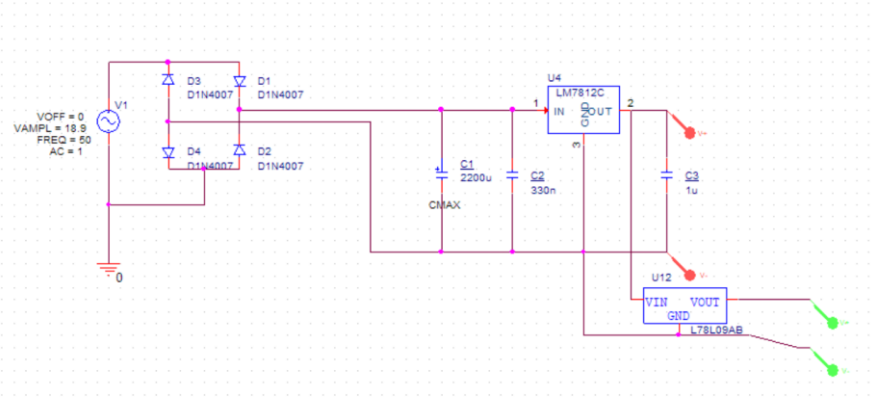


Figura 4.19: Simulación reguladores de tensión positiva de 12 y 9 V.

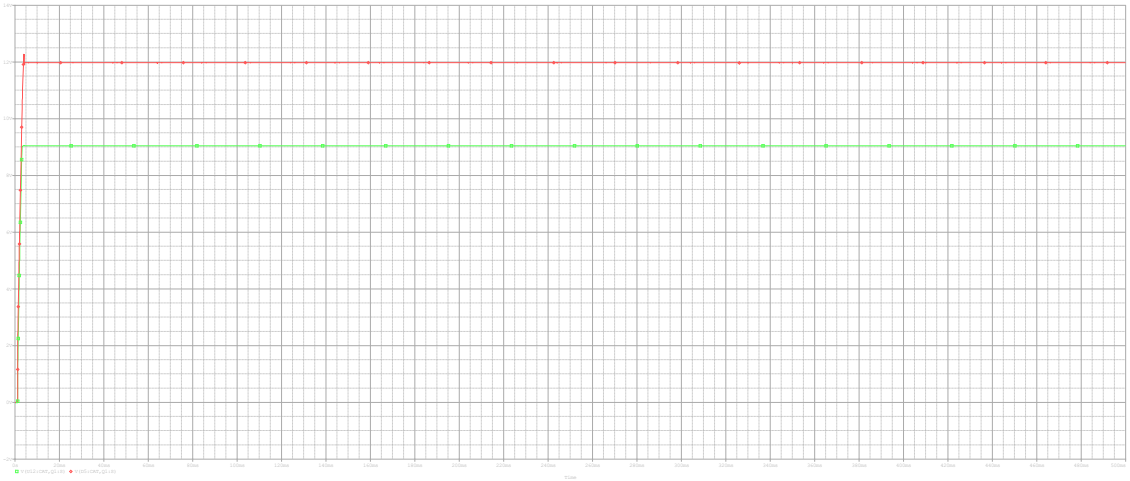


Figura 4.20: Tensiones a la salida de los reguladores de tensión.

4.2.2 Modulación por ancho de pulso

Para finalizar con el diseño de la electrónica de potencia, se necesita un elemento que sea capaz de controlar la tensión que llega al ventilador, este elemento es un transistor.

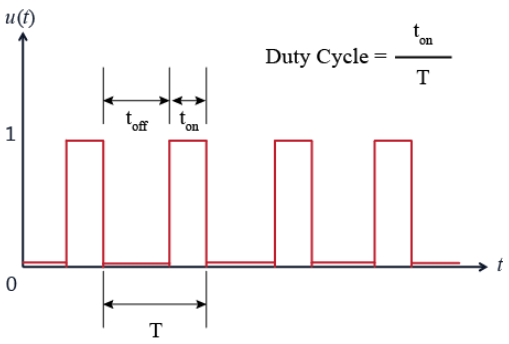


Figura 4.21: Diagrama de tiempos de una señal PWM [30].

Como ya se ha comentado, en concreto el microcontrolador escogido no ofrece una salida analógica, únicamente es capaz de sacar por algunos de sus pines señales PWM. Una señal PWM no es más que una señal digital que alterna estados en los que el valor de la señal es alto (5 V) y otros en los que la señal toma un valor bajo o 0 V. Se define como ciclo de trabajo el cociente

del tiempo cuando la señal está en su valor alto y la suma de ese tiempo más el tiempo que está en valor bajo (o conocido también como periodo de la señal) [8].

$$D = \frac{t_{on}}{t_{on} + t_{off}} = \frac{t_{on}}{T} \qquad (4.12)$$

La clave de las señales de modulación por ancho de pulso es que se puede variar el ciclo de trabajo de la señal. Lo que se consigue al variar el ciclo de trabajo es que varíe el valor medio de la tensión [9].

$$\bar{V} = \frac{1}{T} \cdot \int_0^T V(t)dx \qquad (4.13)$$

La manera clásica de explicar esto, es ver la integral como el área bajo la curva, luego el valor medio de la señal en función del ciclo de trabajo tiene la siguiente expresión.

$$\bar{V} = \frac{1}{T} \cdot \hat{V} \cdot t_{on} = D \cdot \hat{V} \qquad (4.14)$$

Así todo, esta señal no sería capaz de mover el ventilador, por dos motivos, el primero la tensión media equivalente estaría en un rango de 0-5 V, y por otro lado la corriente que es capaz de aportar Arduino a través de un pin PWM no se recomienda que sobrepase los 20mA. El ventilador necesita para funcionar correctamente una tensión de 0-12 V y su corriente es de 0.25 A (consultar anexo 5). La solución a esto es actuar con la señal que saca Arduino sobre un transistor. De esta manera se consigue pasar de una señal PWM de amplitud 5 V a otra señal con la misma forma (aproximadamente), pero de amplitud 12 V.

Por lo tanto, se conseguirían tensiones medias de 0-12 V ($\bar{V} = D \cdot \hat{V}$). La manera en la que resulta intuitiva colocar el transistor es en serie con la carga, de forma que interrumpa el paso de la corriente cuando este se encuentre cortado. Los motores, son cargas inductivas por su naturaleza, por lo que variar la corriente que circula por ellos de forma brusca, provoca un aumento de la tensión con el fin de contrarrestar esa caída de corriente.

$$V = L \cdot \frac{di}{dt} \qquad (4.15)$$

Si ese cambio es prácticamente instantáneo, el resultado son picos de tensión en bornas del ventilador, lo que puede resultar dañino para el ventilador. A pesar de que algunos ventiladores están preparados para esto, en este caso, por seguridad y debido al bajo coste que tiene un diodo y la protección que ofrece, se ha colocado un diodo de recirculación en paralelo con el ventilador. El montaje que se desea realizar es el mostrado en el esquema de la figura 4.22.

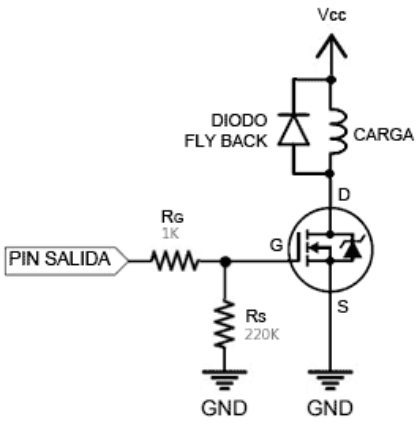


Figura 4.22: Esquema de conexión del ventilador (carga), para ser controlado con la señal PWM [10].

Las resistencias R_g y R_s, son necesarias para que el transistor funcione de forma correcta. El valor de la resistencia de puerta debe ser suficientemente grande para que la corriente que aporta el pin digital (en este caso el número 7) sea asumible por el microcontrolador. Por otro lado, hay que tener cuidado con aumentar en exceso el valor de esa resistencia, ya que, si se aumenta demasiado, las transiciones entre las regiones de corte y saturación serán más lentas (más tiempo en zona lineal), lo cual se traduce en un aumento de temperatura. La finalidad de la resistencia R_s, es poner a tierra el terminal de puerta del MOSFET cuando el pin digital del microcontrolador se encuentra en valor bajo. Al no disponer datos sobre la inductancia del ventilador, únicamente a modo ilustrativo se ha realizado una simulación del comportamiento de una carga resistiva en serie con el transistor que a su vez está controlado por una señal cuadrada.

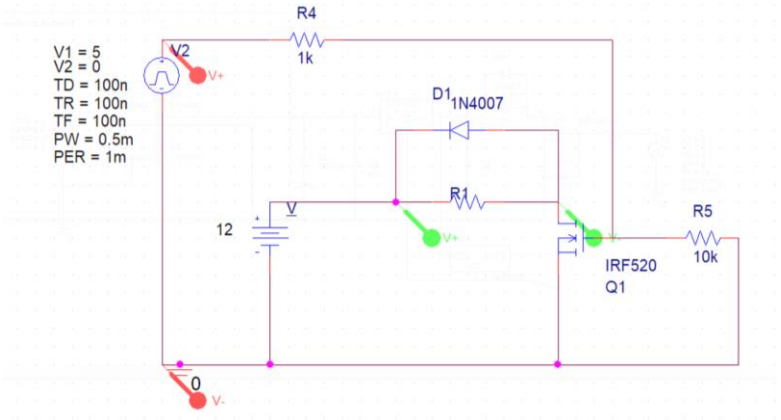


Figura 4.23: Simulación carga resistiva, manejada a través de una señal PWM usando un transistor.

Figura 4.24 Resultados de la simulación de la figura 4.23.

Como se puede apreciar en la figura 4.24, cuando la señal cuadrada que emula el PWM, está en valor alto, el transistor conduce, provocando en bornes de la carga una tensión de 12 V durante el tiempo que este la señal de puerta en valor alto.

En la siguiente figura (4.25) se muestra el circuito en conjunto:

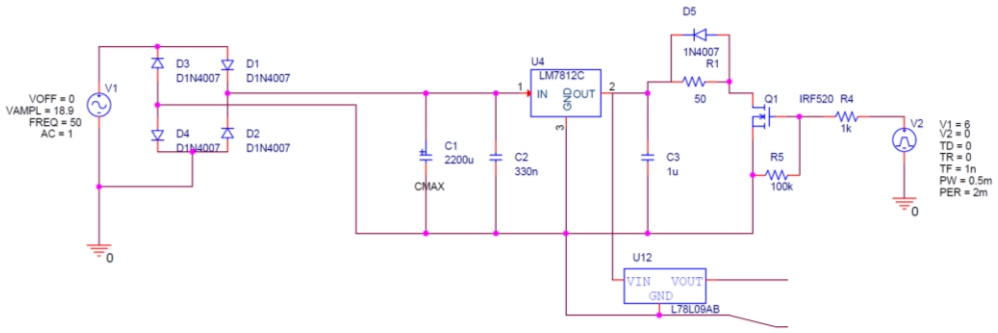


Figura 4.25 Esquemático que resume las conexiones a realizar

4.2.3 Circuito impreso, diseño y fabricación:

El siguiente paso que se desarrolló fue la implementación del circuito previo sobre una PCB.

Actualmente existen multitud de empresas que se dedican a la fabricación de PCB, a un precio unitario más que razonable, el inconveniente de estas, es que la mayor parte exigen una tirada mínima de unidades, como para esta aplicación únicamente es necesaria una unidad, la PCB, se realizará de forma casera, con el fin de reducir costes.

Para ello fue necesario diseñar una máscara, que posteriormente se transferirá al cobre de la PCB virgen. Para el diseño de esta, se ha realizado mediante el software gratuito

EASYEDA. Esta herramienta es similar a otros entornos de simulación como ORCAD-CAPTURE o SPICE. La ventaja de esta es que la creación del diseño de la PCB es muy intuitiva, únicamente hay que replicar el circuito que se muestra en la figura 4.25, elegir el encapsulado de cada componente y distribuir cada elemento como se desee. Al tratarse de un método de grabado casero, no es posible realizar un diseño con varias capas, lo cual resultaría en un formato más compacto. La distribución final de los componentes es la que se muestra en la siguiente figura:

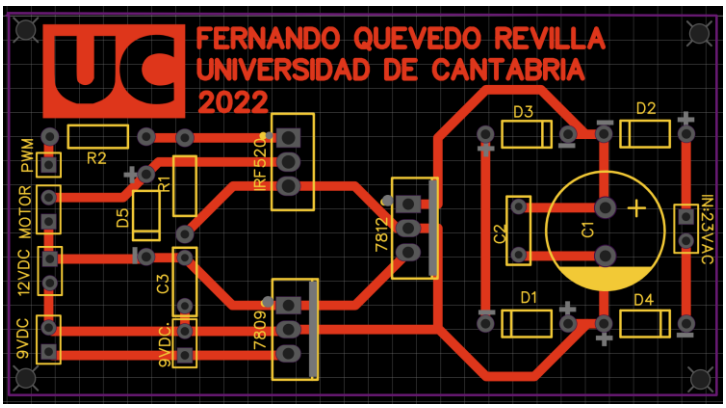


Figura 4.26: Distribución de los componentes electrónicos sobre la PCB, donde los tonos rojos corresponden con las pistas de cobre.

En tonos rojos está representado las pistas o elementos que se deben mantener con el material conductor. En amarillo se representa el serigrafiado que debería tener cada componente, esto puede servir de poka-yoke a la hora de soldar los componentes. Finalmente, en morado aparecen los límites físicos de la placa.

Otra de los puntos fuertes de esta herramienta es la posibilidad de crear y exportar el modelo 3D de la PCB con los componentes a otros programas de diseño CAD. El resultado de la simulación se muestra en la siguiente imagen:

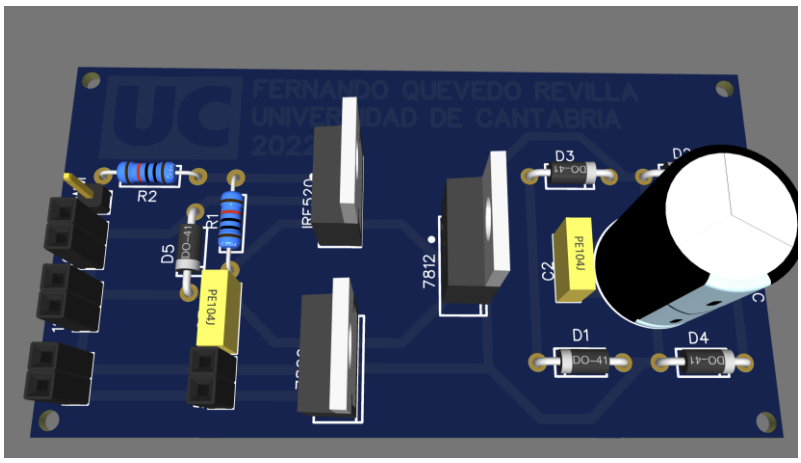


Figura 4.27: Modelo CAD de la PCB diseñada.

Como se ha comentado anteriormente, el método de grabado consiste en un inicio en proteger las zonas que no se desean atacar con una máscara. La forma de crear dicha máscara es exportando el diseño de la PCB a un formato de imagen, con el fin de imprimirlo en una impresora con tecnología láser (uso de tóner). Esto debe de ser así porque el siguiente paso radica en la transferencia del patrón a la placa de cobre mediante calor (por tanto, las impresoras basadas en inyección de tinta no son válidas para este propósito). Un detalle importante a la hora de realizar la impresión es utilizar la máxima resolución y contraste posibles, además de utilizar la opción de “espejo”, ya que, de lo contrario a la hora de transferirlo los textos, pistas y el logotipo, estos quedarían invertidos.

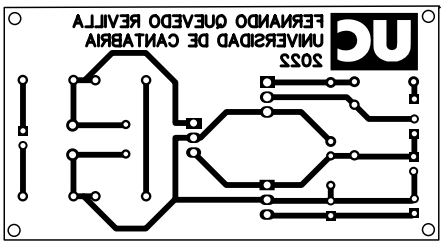


Figura 4.29: Máscara para grabado.

Figura 4.28: PCB de fibra de vidrio y cobre virgen.

Como se vaticinaba anteriormente la transferencia, se realiza colocando el papel con la máscara sobre la placa de cobre previamente limpiada y/o lijada con el fin de que la superficie de cobre sea uniforme y esté libre de impurezas. Posteriormente se aplica calor para que el tóner se adhiera al cobre. Tras ello el papel queda también pegado al cobre, por lo que es necesario retirarlo, sumergiendo el conjunto en agua. El resultado de esta fase se muestra en la ilustración siguiente:

Figura 4.30: Resultado de la transferencia de la máscara a la superficie de la PCB.

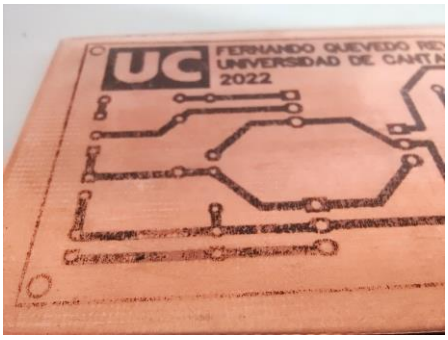


Figura 4.31: Vista en detalle de las zonas con carencia de tóner.

Como se aprecia en sendas figuras (4.30 y 4.31), algunas de las pistas de la zona inferior izquierda, tienen carencia de tóner. Para solucionar este problema y que esas zonas no queden vulnerables frente al atacante, se repasan con un rotulador permanente. Si no se realiza este paso, en la fase de grabado esas zonas podrían ser eliminadas, lo cual haría que la PCB no fuese funcional.

Figura 4.32: Resultado tras repasar el conjunto de la máscara.

Para el grabado, en el mercado se comercializan varias soluciones atacantes, pero en este caso, buscando la reducción de costes, se realizará el grabado con una disolución de ácido clorhídrico y agua oxigenada obtenida a partir de productos que pueden ser adquiridos en supermercados. Las proporciones son, 25% de agua oxigenada de 110 volúmenes (H_2O_2- Peróxido de hidrógeno), 25% de salfumán (HCl- ácido clorhídrico) y un 50% de agua destilada. Durante el grabado se está llevando a cabo la siguiente reacción redox:

$$Cu + H_2O_2 + 2HCl \rightarrow 2H_2O + CuCl_2 \qquad (4.16)$$

Para determinar si es el ácido clorhídrico es el que se reduce $H^+ \rightarrow H_2$, o es en cambio el agua oxigenada, es necesario consular la tabla de potenciales estándar.

Para el ácido, el potencial de reducción es de $E^0\ (\mathrm{V}) = 0$, mientras para el agua oxigenada, es de 1.76. Fuente [11] .

$2H^+(aq) + 2e^-$	→	$H_2(g)$	0.00

$H_2O_2(aq) + 2H^+ + 2e^-$	→	$2H_2O$	+1.76

Por lo que, dado que el potencial del agua oxigenada es mayor, está más favorecida que la creación de H_2 por los protones del ácido.

Por otro lado, el cobre es el agente reductor (se oxida) al perder 2 electrones:

$$Cu^0 - 2e \rightarrow Cu^2 \qquad (4.17)$$

Mientras el H_2O_2 es el agente oxidante (se reduce), ya que el oxígeno gana 2 electrones:

$$2O^{-1} + 2e \rightarrow 2O^{-2} \qquad (4.18)$$

Tras sumergir la PCB con la máscara en las zonas que se desean mantener de cobre, la disolución que inicialmente era incolora empieza a tomar un tono característico del cloruro de cobre (II).

Figura 4.33: Proceso de grabado (I).

Tras un par de minutos se incrementa la intensidad del color de la disolución y se empieza a observar cómo el atacante va eliminando las zonas más vulnerables, en este caso los bordes de la PCB.

Figura 4.34: Proceso de grabado (II).

Es muy importante agitar la disolución para que las características del atacante sean homogéneas por toda la PCB. Como se comentó en el primer paso, antes de colocar la mascará era necesario obtener una superficie lo más uniforme posible, esto es para que la disolución termine de eliminar el cobre de todas las zonas en un tiempo similar.

Figura 4.35: Proceso de grabado (III).

Figura 4.36: Fin del proceso de grabado.

Finalmente, toda la superficie de cobre que no ha sido protegida es eliminada de la PCB. Tras extraer la PCB de la cubeta con ácido, es necesario aclararla con abundante agua para detener la reacción. El resultado del grabado se muestra en la figura 4.37, como se puede observar la máscara ha cumplido su misión de proteger la superficie. El siguiente paso es retirar precisamente la máscara, para ello se utiliza acetona.

Figura 4.37: Resultado tras el grabado.

Tras retirar la máscara, el paso previo antes de proceder con la soldadura de los componentes electrónicos es realizar taladros en las zonas donde se precise. Esos taladros deberán ser de un diámetro tal que permita introducir las patillas de los componentes, pero sin ser excesivo ya que, para realizar la soldadura, es necesario que el estaño por capilaridad sea capaz de rellenar el taladro. Para esta aplicación se han realizado agujeros de diámetro 0.8mm.

Figura 4.38: Resultado tras la retirada de la máscara y el taladrado.

Tras soldar cada uno de los componentes, y añadir el cableado necesario, se obtiene una PCB como la mostrada en la figura 4.39.

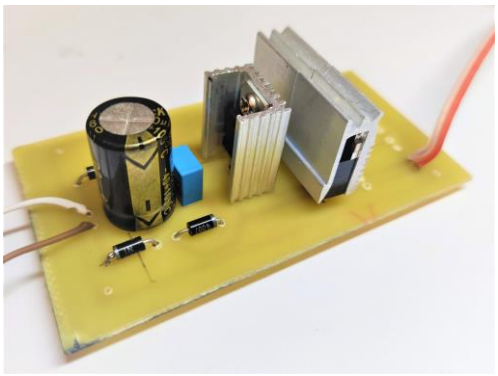


Figura 4.39: Resultado del circuito impreso final, con los elementos soldados.

Remarcar que a pesar de que no es completamente necesario, el uso de disipadores se ha decidido incluirlos para asegurarse una correcta refrigeración pasiva, ya que su coste comparado con el costo de sustitución y montaje de un circuito integrado es ínfimo.

4.3 MODELO MATEMÁTICO

En este capítulo se detalla el proceso de modelado de cada elemento principal de la planta. Como se ha descrito, a lo largo del documento, el sistema está basado en el control de un motor de corriente continua, que gira solidariamente las aspas de un ventilador que a su vez crea un flujo de aire que produce una fuerza sobre la esfera, de igual dirección y sentido contrario al peso de esta. De forma visual, el funcionamiento puede resumir a través del siguiente diagrama de bloques.

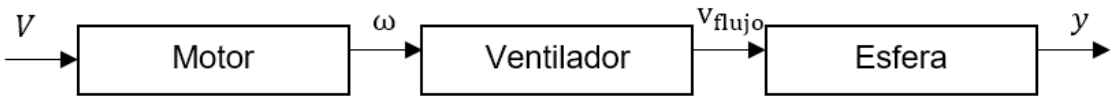


Figura 4.40: Diagrama de bloques simplificado del sistema.

A continuación, se resume el método de obtención de la función de transferencia de cada elemento principal.

4.3.1 Motor DC:

Partiendo de los principios generales de funcionamiento de un motor de corriente continua, se llega al siguiente diagrama.

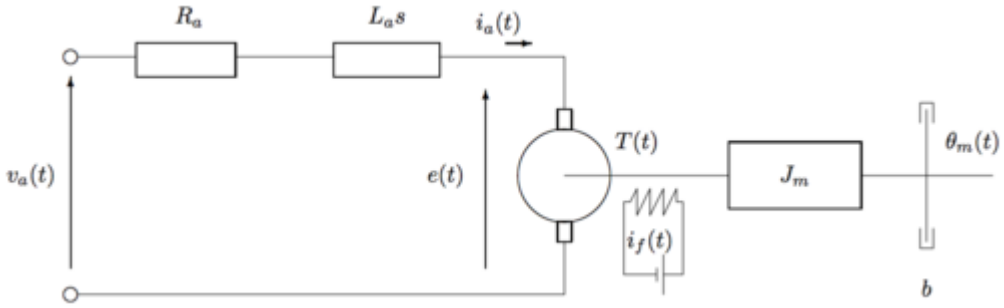


Figura 4.41: Representación del modelo de un motor de corriente continua [12].

Donde los parámetros eléctricos:

R_a: Resistencia de la armadura

L_a: Inductancia de la armadura

i_a: Corriente por la armadura

V_a: Tensión aplicada a la armadura

i_f: Corriente de campo

Donde los parámetros mecánicos:

θ: Desplazamiento angular del eje

T: Par desarrollado por el motor

J: Momento de inercia, del motor y la carga referido al eje.

b: Coeficiente de fricción viscosa del motor y la carga referida al eje

Relación de las ecuaciones eléctricas con mecánicas:

Si la corriente de campo se mantiene constante, la relación entre la corriente de armadura y el par es la siguiente [13].

$$T = K \cdot i_a \qquad (4.19)$$

Donde: K es la constante de par motriz.

Del mismo modo, si se mantiene la corriente de campo constante, se conoce la expresión que relaciona la fuerza contraelectromotriz y la velocidad angular del eje del motor [13].

$$e = K_b \cdot \left(\frac{d\theta}{dt}\right) \qquad (4.20)$$

Donde: K_b es la constante de fuerza contraelectromotriz.

Además, se puede obtener la ecuación de malla del circuito eléctrico mostrado previamente:

$$V_a = V_{R_a} + V_{L_a} + e \qquad (4.21)$$

Sustituyendo el valor de las tensiones V_{L_a} y V_{R_a} en función de la corriente de la armadura:

$$V_a = R_a \cdot i_a + L_a \cdot \frac{di_a}{dt} + e \qquad (4.22)$$

Finalmente, la expresión que caracteriza al subsistema mecánico tiene la siguiente forma [13]:

$$T = J \cdot \frac{d^2\theta}{dt^2} + b \cdot \frac{d\theta}{dt} \qquad (4.23)$$

Llevando estas expresiones al dominio de Laplace y considerando condiciones iniciales nulas, se tiene que:

$$e = K_b \cdot \left(\frac{d\theta}{dt}\right) \rightarrow E(s) = K_b \cdot s \cdot \theta(s) \qquad (4.24)$$

$$V_a = R_a \cdot i_a + L_a \cdot \frac{di_a}{dt} + e \rightarrow V_a(s) = R_a \cdot I_a(s) + L_a \cdot s \cdot I_a(s) + E(s) \qquad (4.25)$$

$$T = J \cdot \frac{d^2\theta}{dt^2} + b \cdot \frac{d\theta}{dt} \rightarrow T(s) = J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s) \qquad (4.26)$$

Agrupando términos se llega a la función de transferencia del motor:

Sustituyendo la primera (4.24) en la tercera (4.26), resulta la expresión (4.27):

$$V_a(s) = R_a \cdot I_a(s) + L_a \cdot s \cdot I_a(s) + K_b \cdot s \cdot \theta(s) \qquad (4.27)$$

Sustituyendo (4.19) en (4.26):

$$K \cdot I_a(s) = J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s) \qquad (4.28)$$

Despejando la corriente por la armadura en la última expresión (4.28):

$$I_a(s) = \frac{J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s)}{K} \qquad (4.29)$$

Sustituyendo la corriente por la armadura (4.29), en la expresión de la tensión de la armadura (4.27):

$$V_a(s) = R_a \cdot \frac{J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s)}{K} + L_a \cdot s \cdot \frac{J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s)}{K} + K_b \cdot s \cdot \theta(s) \qquad (4.30)$$

Operando (4.30):

$$V_a(s) = \frac{R_a \cdot \left(J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s)\right) + L_a \cdot s \cdot \left(J \cdot s^2 \cdot \theta(s) + b \cdot s \cdot \theta(s)\right) + K \cdot K_b \cdot s \cdot \theta(s)}{K} =$$

$$= \frac{R_a \cdot J \cdot s^2 \cdot \theta(s) + R_a \cdot b \cdot s \cdot \theta(s) + L_a \cdot s \cdot J \cdot s^2 \cdot \theta(s) + L_a \cdot s \cdot b \cdot s \cdot \theta(s) + K \cdot K_b \cdot s \cdot \theta(s)}{K}$$

(4.31)

Sacando el desplazamiento angular como factor común del numerador (4.31) y pasándolo, dividiendo al otro lado de la expresión:

$$\frac{V_a(s)}{\theta(s)} = \frac{R_a \cdot J \cdot s^2 + R_a \cdot b \cdot s + L_a \cdot s \cdot J \cdot s^2 \cdot + L_a \cdot s \cdot b \cdot s + K \cdot K_b \cdot s}{K} \qquad \textit{(4.32)}$$

Agrupando términos de la expresión (4.32):

$$\frac{V_a(s)}{\theta(s)} = \frac{L_a \cdot J \cdot s^3 + s^2(R_a \cdot J + L_a \cdot b) + s \cdot b(R_a + K \cdot K_b)}{K} \qquad \textit{(4.33)}$$

Poniendo (4.33) en forma de función de transferencia, donde la entrada es la tensión y la salida el desplazamiento angular:

$$\frac{\theta(s)}{V_a(s)} = \frac{K}{L_a \cdot J \cdot s^3 + s^2(R_a \cdot J + L_a \cdot b) + s \cdot b(R_a + K \cdot K_b)} \qquad \textit{(4.34)}$$

Conociendo que la velocidad angular es la derivada del desplazamiento angular:

$$\omega(t) = \frac{d\theta}{dt} \rightarrow \omega(s) = \theta(s) \cdot s \qquad \textit{(4.35)}$$

Se obtiene la función de transferencia que relaciona la tensión de la armadura con la velocidad de giro del eje del motor (4.36).

$$\frac{\omega(s)}{V_a(s)} = \frac{K}{L_a \cdot J \cdot s^2 + s\,(R_a \cdot J + L_a \cdot b) + b(R_a + K \cdot K_b)} \qquad \textit{(4.36)}$$

4.3.2 Ventilador:

El modelado de las propias aspas del ventilador puede ser complejo y posteriormente no corresponderse fielmente a la realidad, se aproxima que la velocidad del flujo de aire que emite es directamente proporcional a la velocidad de giro de las aspas.

$$v_{flujo} = K_{vent} \cdot \omega \rightarrow \mathrm{v_{flujo}}(s) = K_{vent} \cdot \omega(s) \rightarrow \frac{v_{flujo}(s)}{\omega(s)} = K_{vent} \qquad (4.37)$$

4.3.3 Esfera

Aplicando la segunda ley de Newton a la bola de poliestireno:

$$\sum F = m * a \rightarrow F_a - P = m * a \rightarrow F_a - m * g = m * a = m\,\ddot{y} \qquad (4.38)$$

Donde:

a : Aceleración del objeto

F_a: Fuerza de arrastre debido al flujo de aire

m: Masa del objeto

g: Aceleración de la gravedad

y: Altura del objeto

Así mismo se conoce que la fuerza de arrastre (o drag) tiene la siguiente expresión [14]:

$$F_a = \frac{1}{2}\, c_d\, \rho\, \mathrm{A}\, \left(\mathrm{v_{flujo}} - \mathrm{v_{objeto}}\right)^2 \qquad (4.39)$$

Donde:

c_d: Coeficiente de arrastre, para una esfera lisa toma el valor de 0.47 ($Re = 10^5$) [15]

ρ : Densidad del aire, en condiciones estándar 1.1839 kg/m3

A : Área de la sección transversal a la dirección del movimiento

$\mathrm{v_{flujo}}$: Velocidad del flujo de aire

$\mathrm{v_{objeto}}$: Velocidad del objeto

Si se consideran los tres primeros parámetros constantes de la expresión (4.39):

$$K_a = \frac{1}{2}\, c_d\, \rho\, \mathrm{A} \qquad (4.40)$$

Sustituyendo en la primera expresión (4.38):

$$K_a \left(\mathrm{v}_{\mathrm{flujo}} - \mathrm{v}_{\mathrm{objeto}}\right)^2 - m \cdot g \qquad (4.41)$$

$$K_a \left(\mathrm{v}_{\mathrm{flujo}} - \dot{y}\right)^2 = m\,\ddot{y} \qquad (4.42)$$

Despejando la aceleración de (4.42):

$$\ddot{y} = \frac{K_a \left(\mathrm{v}_{\mathrm{flujo}} - \dot{y}\right)^2}{m} - g \qquad (4.43)$$

Como se puede observar, es una ecuación no lineal, ya que la aceleración depende entre otros parámetros, del cuadrado de la velocidad. Para poder obtener una función de transferencia, se linealiza la ecuación previa (4.43).

Cuando la pelota se estabilice a una determinada altura (cuando la fuerza de arrastre iguale al peso), se puede afirmar que, tanto la aceleración, como la velocidad del objeto es nula:

$$0 = \frac{K\,\left(\mathrm{v}_{\mathrm{flujo}_{\mathrm{equilibrio}}} - 0\right)^2}{m} - g \rightarrow \;\; 0 = \frac{K\,\mathrm{v}^2_{\mathrm{flujo}_{\mathrm{equilibrio}}}}{m} - g \qquad (4.44)$$

Despejando la velocidad del flujo que hace que la esfera no se mueva ($\ddot{y} = \dot{y} = 0$):

$$\mathrm{v}_{\mathrm{flujo}_{\mathrm{equilibrio}}} = \sqrt{\frac{m \cdot g}{K_a}} \qquad (4.45)$$

Linealizando, mediante la aproximación al primer término de la serie de Taylor de la expresión no lineal anterior (4.43):

$$\ddot{y} = \ddot{y}_{eq} + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{v}_{\mathrm{flujo}}}\right|_{(v_{flujo_{eq}}, \dot{y}_{eq})} \cdot (\mathrm{vflujo} - \mathrm{vflujoeq}) + \left.\frac{\partial \mathrm{f}}{\partial \dot{y}}\right|_{(v_{flujo_{eq}}, \dot{y}_{eq})} \cdot \left(\dot{y} - \dot{y_{eq}}\right) \ldots (términos\ de\ orden\ superior) \qquad (4.46)$$

Llamando:

$$k_v = \left.\frac{\partial \mathrm{f}}{\partial \mathrm{v}_{\mathrm{flujo}}}\right|_{(v_{flujo_{eq}}, \dot{y}_{eq})} = \frac{2 \cdot K_a}{m}\left(v_{flujo_{eq}} - \dot{y}_{eq}\right) = \frac{2 \cdot K_a}{m} \cdot \sqrt{\frac{m \cdot g}{K_a}} = 2\sqrt{\frac{K_a \cdot g}{m}} \qquad (4.47)$$

$$k_y = \frac{\partial \mathrm{f}}{\partial \dot{y}}\Big|_{(v_{flujo_{eq}}, \dot{y}_{eq})} = \frac{-2 \cdot K_a}{m}\left(v_{flujo_{eq}} - \dot{y}_{eq}\right) = -k_v = -2\sqrt{\frac{K_a \cdot g}{m}} \qquad (4.48)$$

Si se llama $\bar{y}$ a la desviación de la altura de la pelota respecto al punto de equilibrio, se obtiene:

$$\bar{y} = y - y_{eq} \;\rightarrow \dot{\bar{y}} = \dot{y} - \dot{y_{eq}} \rightarrow \; \ddot{\bar{y}} = \ddot{y} - \ddot{y_{eq}} \qquad (4.49)$$

A su vez también se llama $\overline{v_{flujo}}$ a la diferencia de la velocidad del aire respecto a la velocidad del aire en el punto de equilibrio.

$$\overline{v_{flujo}} = v_{flujo} - v_{flujo_{eq}} \qquad (4.50)$$

Juntando las expresiones anteriores, resulta la expresión lineal que modeliza el comportamiento físico de la bola.

$$\ddot{\bar{y}} = k_v \cdot \overline{v_{flujo}} - k_v \cdot \dot{\bar{y}} \; = k_v \cdot \left(\overline{v_{flujo}} - \dot{\bar{y}}\right)(Lineal) \qquad (4.51)$$

Llevando la ecuación previa (4.51) al dominio de Laplace se obtiene:

$$Y(s) \cdot s^2 = k_v \cdot \overline{v_{flujo}}(s) - k_v \cdot Y(s) \cdot s \rightarrow \frac{Y(s)}{\overline{v_{flujo}}(s)} = \frac{k_v}{s^2 - k_v \cdot s} = \frac{k_v}{s \cdot (s - k_v)} \qquad (4.52)$$

4.3.4 Función de transferencia de lazo abierto:

Una vez que se han obtenido las funciones de transferencia de cada elemento, para obtener la función de transferencia de lazo abierto del sistema, basta con realizar la serie o producto del conjunto (4.36),(4.37) y (4.52).

$$G(s) = \frac{\omega(s)}{V_a(s)} \cdot \frac{v_{flujo}(s)}{\omega(s)} \cdot \frac{Y(s)}{\overline{v_{flujo}}(s)}$$

$$G(s) = \frac{K}{L_a \cdot J \cdot s^2 + s\,(R_a \cdot J + L_a \cdot b) + b(R_a + K \cdot K_b)} \cdot K_{vent} \cdot \frac{k_v}{s \cdot (s - k_v)}$$

$$G(s) = \frac{K \cdot K_{vent} \cdot k_v}{[L_a \cdot J \cdot s^2 + s\,(R_a \cdot J + L_a \cdot b) + b(R_a + K \cdot K_b)] \cdot s \cdot (s - k_v)}$$

(4.53)

Operando (4.53), se llega a la siguiente expresión (4.54):

$$G(s) = \frac{K \cdot K_{vent} \cdot k_v}{[L_a \cdot J \cdot s^3 + s^2 (R_a \cdot J + L_a \cdot b) + b \cdot s \cdot (R_a + K \cdot K_b)](s - k_v)}$$

$$G(s) = \frac{K \cdot K_{vent} \cdot k_v}{L_a \cdot J \cdot s^4 + s^3 (R_a \cdot J + L_a \cdot b) + b \cdot s^2 \cdot (R_a + K \cdot K_b) - k_v \cdot L_a \cdot J \cdot s^3 - k_v \cdot s^2 (R_a \cdot J + L_a \cdot b) - k_v \cdot b \cdot s \cdot (R_a + K \cdot K_b)}$$

(4.54)

Agrupando los términos de mismo orden, resulta la expresión (4.55):

$$G(s) = \frac{K \cdot K_{vent} \cdot k_v}{L_a \cdot J \cdot s^4 + s^3 (R_a \cdot J + L_a \cdot b - k_v \cdot L_a \cdot J) + s^2 \cdot [b \cdot (R_a + K \cdot K_b) - k_v (R_a \cdot J + L_a \cdot b)] - k_v \cdot b \cdot s \cdot (R_a + K \cdot K_b)}$$

(4.55)

Como se aprecia, aun despreciando efectos como por ejemplo, el debido a la viscosidad del aire, la función de transferencia es compleja, a lo que se le suma la dificultad de medir cada parámetro del conjunto motor-ventilador, ya que en este caso el fabricante del motor-ventilador, no ofrece el valor de los parámetros internos, sin los cuales resultaría extremadamente complejo calcular de forma analítica una función de transferencia que asemeje el funcionamiento del sistema. Por ese motivo, en el apartado 4.7 IDENTIFICACIÓN, se puede consultar el proceso de identificación de la planta.

4.4 CONTROL

Este capítulo se dividirá en dos grandes apartados, el análisis de soluciones, y la implementación.

Previamente a desarrollar el programa principal, será necesario escoger adecuadamente, el elemento que gobernará el sistema, así como la sensórica y periferia necesaria.

4.4.1 Análisis de soluciones

En este subcapítulo se detallarán las distintas opciones disponibles en el mercado para seleccionar el hardware de control de la planta.

4.4.1.1 Selección del equipo principal

El equipo principal, además de actualizable, suficientemente solvente para la aplicación, fiable …etc., se requiere que pueda comunicarse a través de USB (ya que es una de las características que debe incluir la planta). Por otro lado, debe ser relativamente robusto, y de dimensiones comedidas. A continuación, se exponen las opciones contempladas:

-Microcontrolador PIC:

De las opciones contempladas sin duda es la más económica a la hora de adquirirlo (incluso puede ser gratuito, acogiéndose al programa de muestras de grandes fabricantes como MICROCHIP). La principal desventaja de este es que es necesario un aparato externo para programarlo, lo cual haría aumentar el presupuesto. Por otro lado, si se desea programar el equipo utilizando lenguajes de bajo nivel, haría que los costes derivados de la programación aumenten, lo cual no tiene sentido si únicamente se fabricará una unidad, por lo que esta opción quedaría descartada al no poder prorratear los gastos de diseño, ni los gastos del equipo de programación.

-PLC:

Un autómata programable hace que la planta esté abierta a posibles modificaciones futuras, ya que es fácilmente actualizable tanto software como hardware. Los fabricantes de autómatas industriales suelen seguir dando soporte a equipos antiguos, en lo que respecta a su compatibilidad con las herramientas de programación. Por otro lado, el importe de adquisición del equipo es elevado comparado con las otras opciones. Las necesidades del sistema, hace que en cierto modo las características de un PLC (aunque sea el más básico de la gama) queden infrautilizadas, desperdiciando su potencial.

-Plataforma Arduino:

El proyecto Arduino nació en 2003, cuando varios estudiantes del Instituto de Diseño Interactivo de Ivrea (Italia) desarrollaron una alternativa económica para facilitar el acceso por parte de los alumnos a la programación. La idea del proyecto era juntar en una placa todos los elementos necesarios asociados a un microcontrolador con el fin de conectar periféricos. Por lo que Arduino, no deja de ser otro microcontrolador, distribuido en distintos formatos de placas y con distintas configuraciones hardware. El entorno de desarrollo software (Arduino IDE) es bastante amigable y permite de una manera sencilla escribir código y cargarlo a la placa. Además, el uso de Arduino está bastante extendido con gran cantidad de información, tutoriales y librerías que facilitan la programación. Por otro lado, el programa incluye ejemplos prácticos que se pueden modificar y cargar a la placa, además cabe destacar que, a diferencia

de otros entornos de programación, Arduino IDE, hace que el código sea fácil e intuitivo de depurar. En especial, la nueva versión 2.0 que todavía se encuentra en su fase de desarrollo final, ofrece una interfaz más moderna que la previa, así como funciones de autocompletado e incluso la posibilidad de depurar en tiempo real.

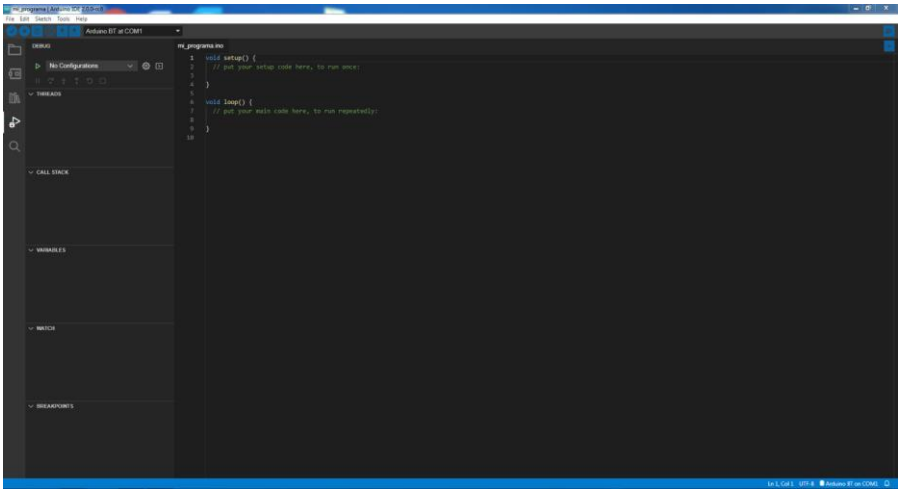


Figura 4.42: Entorno ARDUINO IDE versión 2.0 RC.

Figura 4.43: Entorno ARDUINO IDE versión 1.8.19.

Existen varias configuraciones de placas Arduino, cada una con sus propias características, en base a estas, cada modelo se ajustará más o menos a una aplicación determinada. Existen placas pequeñas enfocadas a aplicaciones donde se requieren dimensiones comedidas y hardware modesto (por ejemplo, para aplicaciones portátiles) y en contraposición existen modelos como Arduino MEGA, que dispone de 256 KB de memoria hasta 54 pines digitales (configurables como entrada o salida), 16 entradas analógicas, compatibilidad con distintos protocolos de comunicación… etc., todo ello acompañado de un microcontrolador

ATmega2560, de la casa MICROCHIP. A continuación, se muestra una tabla comparativa de las versiones de Arduino más utilizadas:

	Arduino One	Etherner	Leonardo	Arduino DUE	ADK
					
Microcontroller	ATmega328	ATmega328	ATmega32U4	Atmel SAM3U4E ARM Cortex M3	ATmega2560
Clock	16 MHz	16 MHz	16 MHz	96 MHz	16 MHz
Flash Memory	32 KB	32 KB	32 KB	256 KB	256 KB
SRAM	2 KB	2 KB	3.3 KB	50 KB	8 KB
Digital I/O Pins	14	14 (10)	14	54	54
Analog Pins	6	6	6	16 (12bit)	16

Figura 4.44: Comparativa versiones de Arduino principales [16].

Ahora bien, ¿Por qué decidirse por un Arduino frente a las otras opciones? La respuesta, no radica en una característica clave, sino que es un conjunto de ellas: económico, facilidad de programación, comunidad de usuarios, entorno de programación sencillo, así como el bagaje previo de Arduino. Finalmente, por la diferencia de precios entre un Arduino Mega y un Arduino Uno (en el distribuidor, la diferencia no superaba los 5€ a favor de Arduino Uno), se decide escoger Arduino Mega, y así además hacer que el sistema pueda gobernar más periféricos si en el futuro así se desea.

4.4.1.2 Sensor

A la hora de escoger el sensor cuya medida realimentará el sistema se deben tener en cuenta los siguientes factores:

1. Económico: El sensor debe ser relativamente económico
2. Preciso: De entre todos los sensores del mercado se debe escoger uno cuyo fondo de escala sea de al menos un metro y cuya resolución sea de al menos 0.5 cm.
3. Compatible con Arduino: Existen gran cantidad de sensores que se comunican mediante algunos protocolos de comunicación que no son compatibles con Arduino. De forma resumida debe trabajar de forma digital o a través de I2C o SPI.

Tras buscar las distintas alternativas actuales del mercado se seleccionan los siguientes sensores, ultrasónico WPSE306N, láser VL53L0X. A continuación, se detallan las principales características.

Tabla 4.3 Comparativa sensor ultrasónico y TOF [17]

	Ultrasónico WPSE306N	Láser VL53L0X
Tensión de alimentación	3.3-5 VDC	3.3-5 VDC
Corriente	50-100 mA	10mA
Resolución	0.3 cm	+/- 3 %
Rango de medida	3-550 cm	3-200 cm
Periodo de muestro mínimo	50 ms	30 ms
Comunicación	Digital	I2C
PVP	5.74€	15.26€

Para más información consultar anexo 6.

1.Sensores ultrasónicos:

Este tipo de sensores se basan en que un emisor envía una señal mecánica a través del aire, cuando esta choca con un objeto sólido, la onda rebota y es captada por el transductor del propio sensor. Pues la forma en la que se obtiene la medida es precisamente la diferencia de tiempos entre que se emite la onda ultrasónica y se recibe.

La expresión se deduce a continuación:

La velocidad del sonido en gases es la siguiente [18]:

$$v = \sqrt{\frac{\Upsilon\, R\, T}{M}} \qquad (4.56)$$

Siendo:

$$\Upsilon \rightarrow Coeficiente\ de\ dilatación\ adiabática$$

$$R \rightarrow Constante\ universal\ de\ los\ gases$$

$$M \rightarrow Masa\ molar\ del\ gas$$

$$T \rightarrow Temperatura$$

Concretando para el caso del aire:

$$\Upsilon = 1.4$$

$$R = 8.314 \frac{J}{mol\ K}$$

$$M = 0.029 \frac{kg}{mol}$$

$$T = 293{,}15\ K\ (20ºC)$$

Fuente [18]

Luego la velocidad a 20ºC:

$$v = \sqrt{\frac{1.4 * 8.314 * 293.15}{0.029}} = 343.016 \frac{m}{s}$$

$$343.016 \frac{m}{s} * 100 \frac{cm}{m} * \frac{1}{1000000} \frac{s}{\mu s} = \frac{1}{29.15} \frac{cm}{\mu s} \qquad (4.57)$$

La anterior velocidad es para el caso del aire a una temperatura ambiente de 20ºC, luego derivado de esta condición surge su primer inconveniente, su dependencia de la temperatura ambiente. A continuación, se estima el error debido a la temperatura.

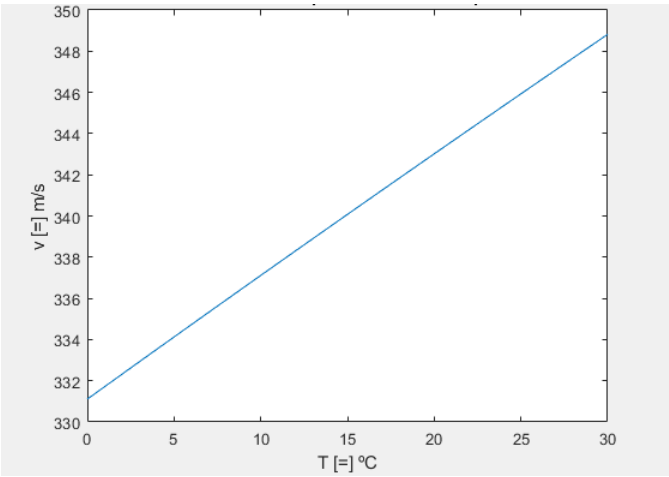


Figura 4.45: Gráfico de la variación de la velocidad del sonido en función de la temperatura.

Calculando la pendiente de la recta anterior, se obtiene que aproximadamente por cada grado centígrado que varía la temperatura ambiente, la velocidad del sonido varia 0.5903 m/s.

Una vez que se obtiene la velocidad, conociendo el tiempo entre que se emite y recibe la onda, se puede calcular la distancia recorrida:

$$Distancia_{recorrida} = v * Tiempo$$

Pero la distancia recorrida no es la distancia sensor-objeto, ya que la onda ha realizado el camino de ida y de vuelta, luego la distancia será:

$$Distancia = v * \frac{Tiempo}{2}$$

$$Distancia[cm] = \frac{1}{2 * 29.15} \frac{cm}{\mu s} * Tiempo[\mu s] \quad (4.58)$$

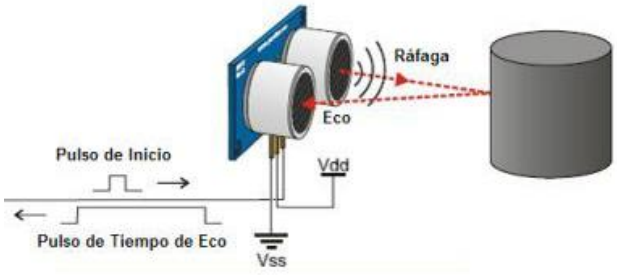


Figura 4.46: Esquema de funcionamiento de sensores ultrasónicos [19]

Concretamente el sensor WPSE306N dispone de 5 pines, uno de ellos es el conocido como disparador (Trigger), este es una entrada del sensor que cuando detecta nivel lógico alto, se emiten 8 pulsos a través de su transductor piezoeléctrico a una frecuencia de 40kHz.

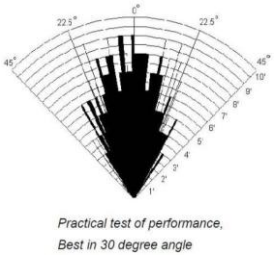


Figura 4.47: Dispersión de la radiación ultrasónica. [20].

Debido a la naturaleza del emisor y del medio, la radiación ultrasónica es de forma cónica con un ángulo de dispersión de entre 15 y 20º (ver figura 4.47).

Dicha onda ultrasónica viajará por el medio hasta chochar contra un objeto. El choque hace reflexionar la onda, haciendo que cambie su sentido, de vuelta al sensor, y es cuando llega a este, cuando el otro piezo-eléctrico detecta la onda y activa su salida digital a través del pin de eco (echo).

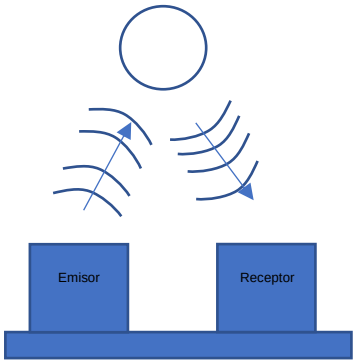


Figura 4.48: Esquema de funcionamiento del sensor WPSE306N.

A efectos prácticos, el tiempo que el pulso eco se mantiene en valor alto será el que se utilice para obtener la medida (ver figura 4.48 y 4.49)

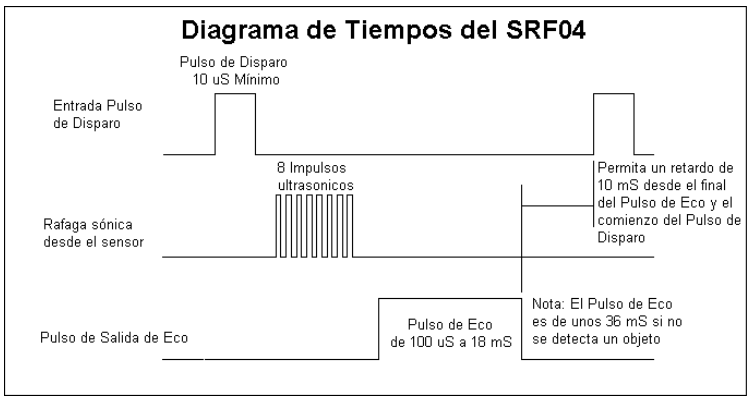


Figura 4.49: Diagrama de tiempos de un sensor similar al contemplado [20].

Sobre el rango de medida del sensor, satisface con creces los requisitos del sistema, ya que la distancia máxima del objeto a detectar es de un metro. Uno de los principales problemas que se pueden derivar debido al entorno donde se realizará la medida es la presencia de aire en movimiento (procedente del flujo que permitirá la sustentación), por lo que antes de escoger el sensor definitivo habrá que realizar algún ensayo para comprobar cómo se comporta este. También resulta intuitivo que el material y el tipo de superficie del objeto que se desea detectar influye positiva o negativamente en función de la capacidad del material de reflexionar el sonido (no se comporta igual ante una espuma, que ante un objeto macizo) y si la superficie presenta irregularidades o no.

2. Sensor láser:

Los sensores de distancia infrarrojo láser tienen un principio de funcionamiento similar a los sensores ultrasónicos, pero en este caso, lo que se emite y recibe es un haz de luz, e igual que sucede con el sensor ultrasónico la distancia se obtiene de la medición del tiempo entre que es emitido y recibido, por este motivo a los sensores láser de distancia se suelen comercializar bajo la denominación de TOF (time of flight).

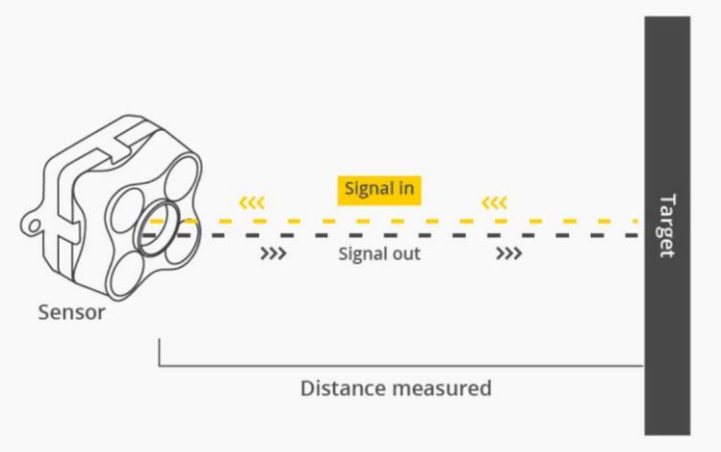


Figura 4.50: Principio de funcionamiento de un sensor TOF [21].

En concreto el sensor VL53L0X, utiliza un emisor láser de 940 nm de longitud de onda VCSEL (del inglés Vertical-cavity surface-emitting laser) y un receptor SPAD (del inglés Single Photon Avalanche Diodes). Además, este sensor cuenta con una electrónica interna que convierte esa diferencia de tiempo en distancia. Adicionalmente la placa donde se integra este sensor, incluye un chip encargado de las comunicaciones I2C. Fuente [21].

A pesar de que, sobre el papel, el sensor láser ofrece ligeramente mejores características, finalmente se escoge el sensor ultrasónico por diversos motivos. El principal quizás sea el económico, los precios de venta al público de ambos sensores son baratos, si bien es verdad que el láser triplica el precio de un ultrasónico. El otro motivo que justifica la elección es que el sensor debe tener algún inconveniente, es decir, si se escoge el sensor láser los alumnos cuando realicen la programación y/o ajuste de la planta, no necesitarán introducir ningún filtro de la señal (lo cual a efectos de aprendizaje es perjudicial). En cambio, si se escoge el sensor ultrasónico, la medida de la altura se verá perturbada por el flujo de aire incidiendo sobre la superficie de los transductores del sensor, lo cual se plantea como un ejemplo práctico real de señales con ruido que se deben filtrar. Pero previamente a la implementación física final en la planta, fue necesario realizar un ensayo a ese sensor con el fin de comprobar si un flujo

de aire similar puede hacer que la lectura del sensor sea demasiado ruidosa para incluso poderse utilizar. Los resultados de dicho ensayo se muestran en las figuras 4.51 y 4.52. El trazo rojo, representa gráficamente la lectura del sensor ultrasónico, viéndose afectado por un flujo de aire de diversa magnitud.

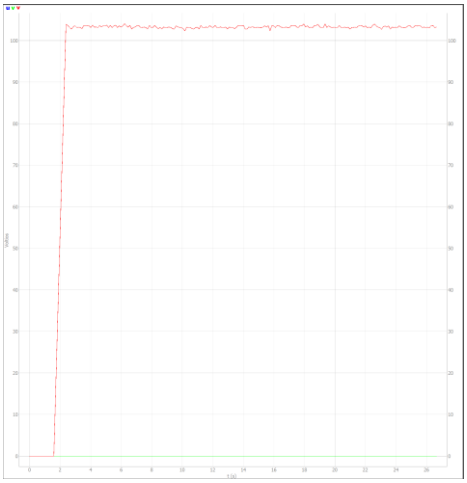


Figura 4.51: Ensayo del sensor ultrasónico con aire débil incidiendo sobre los piezo-eléctricos.

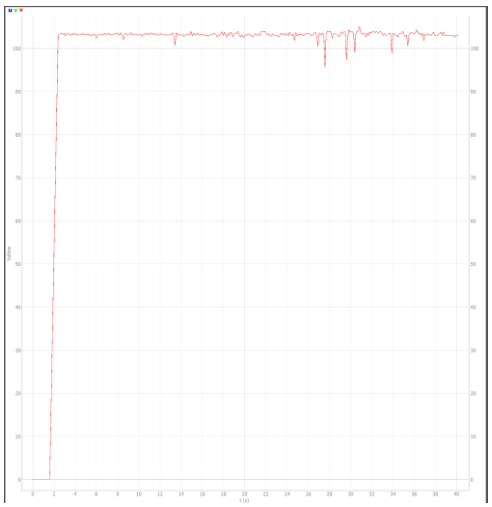


Figura 4.52: Ensayo sensor ultrasónico, con aire fuerte incidiendo sobre este.

Como se puede observar, la perturbación no es demasiado grande, por lo que el sistema es capaz de funcionar sin filtro (pero no se obtendrán resultados óptimos). La figura que está perturbada con un viento de mayor magnitud (4.52), tiene picos, lo cual es planteado para que los alumnos escojan una tipología de filtro e intenten reducir esos picos. Remarcar que para las fases de desarrollo del proyecto se ha filtrado la señal, y posteriormente se ha eliminado el código correspondiente a esa parte precisamente para que dicha labor sea realizada por los alumnos.

Tras haber escogido tanto el equipo principal como el sensor, y haber comprobado su compatibilidad se puede pasar a la implementación.

4.4.2 Implementación

La comunicación Arduino – Sensor se realiza a través de los pines digitales “trigger” y “echo”. Por lo que en la placa de Arduino se deben destinar una salida y una entrada digital. En este caso para este fin de destinan los pines 8 y 9 (además de los dos cables correspondientes a la alimentación).

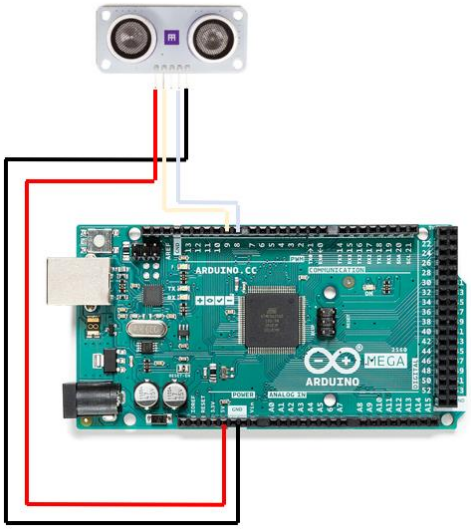


Figura 4.53: Esquema de conexionado del sensor ultrasónico a la placa Arduino.

La medida de la altura, como se ha explicado anteriormente, proviene de la diferencia de tiempo entre que la onda ultrasónica es emitida y recibida. La forma de programar esto en la placa Arduino es sencilla. En este caso se ha introducido la secuencia en una función.

```
float muestrea_distancia(){
    //Lectura sensor ultrasonico
    digitalWrite(trigger,1);  // Activar el trigger (disparo)
    delayMicroseconds(10);    // Mantenerlo activado 10us
    digitalWrite(trigger,0);  // Pasado ese tiempo,desactivarlo
    tiempo=pulseIn(echo,1);   // Obtener el tiempo hasta que se detecte echo
    float d=100-tiempo/58.3 -3.75; // Calcular el la velocidad a partir del tiempo
    // Si debido a ruido, la distancia es negativa, forzarla a cero
    if(d<=0){
      return 0;
    }
    return  d;                //Si la distancia es positiva, devolver ese valor
  }
```

Figura 4.54: Función muestrea_distancia Arduino.

Dentro de esa función, se activa el “trigger” durante 10 microsegundos, y se obtiene el tiempo que el pulso echo se mantiene activo. Las siguientes ilustraciones muestran las señales capturadas con un osciloscopio.

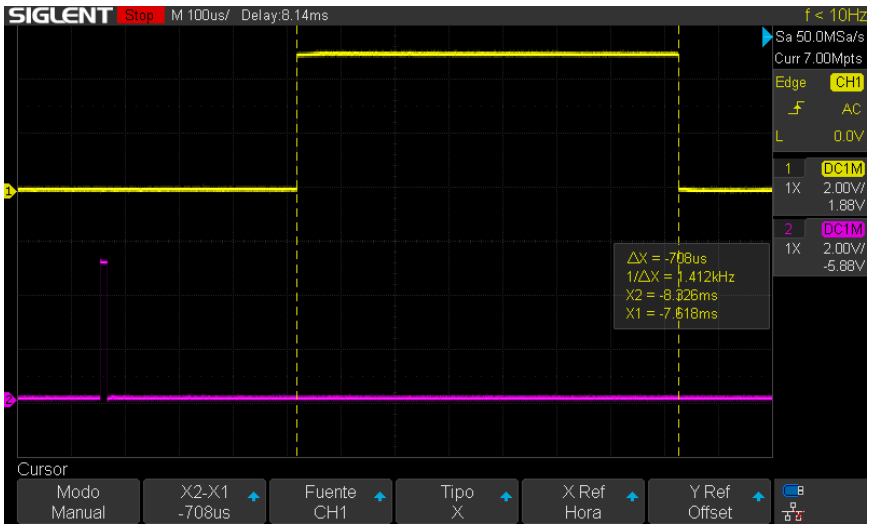


Figura 4.55: Captura pantalla osciloscopio; Señal de disparo y eco (magenta y amarillo respectivamente).

En morado se representa la señal de disparo del sensor, mientras que en amarillo se muestra la señal de eco. En la figura (4.55) se muestra ese tiempo en la que el pulso eco está en alto.

Para este caso real:

$$Distancia[cm] = \frac{1}{2 * 29.15}\frac{cm}{\mu s} * Tiempo[\mu s] \qquad (4.59)$$

$$Distancia[cm] = \frac{1}{2 * 29.15}\frac{cm}{\mu s} * 708\mu s = 12.144cm$$

Pero como se muestra en el apartado anterior, el sensor se encuentra en la parte superior se la planta, por lo que esa distancia no es en sí la altura, si no la distancia entre el sensor y la esfera. Como se desea calcular la altura de la esfera se deberá tener en cuenta la posición del sensor. A la vista del esquema de la figura del margen derecho (4.56), la relación entre la distancia medida y la distancia desde la base de la planta es la siguiente:

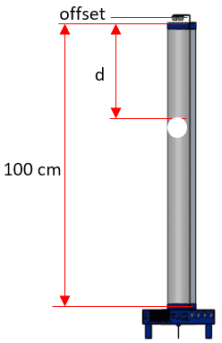


Figura 4.56: Esquema relación medida tomada por el sensor y altura de la bola.

$$d_{esfera-base} = 100 - d_{sensor-esfera} - offset \quad (4.60)$$

El offset es completamente necesario, ya que el sensor es capaz de medir según su hoja de características desde 2 centímetros (por debajo de esta distancia no funciona bien). Por ese motivo el sensor se encuentra sobre elevado respecto al tubo y la escala numerada.

Una vez obtenida la altura, la placa Arduino o el ordenador al que esté conectado la planta procesará dicha información y calculará la tensión media que debe ser aplicada al actuador (señal de control).

4.5 INTERFAZ

Se ha comentado en múltiples ocasiones, que el sistema se podrá manejar bien físicamente interactuando con la pantalla, potenciómetros …etcétera o bien mediante un ordenador conectado a la planta. Con el fin de diferenciar estas dos interfaces, este capítulo se dividirá en dos partes:

4.5.1 Interfaz física

Con objeto de poderse iniciar con el manejo de la planta de manera sencilla, se ha diseñado una interfaz básica compuesta por un lado por unos elementos de control, y por otro lado una pantalla donde poder consultar datos. El microcontrolador escogido es capaz de comunicarse a través de varios protocolos, entre los más relevantes I2C, SPI, Serial. A la hora de seleccionar la pantalla se han valorado varias opciones, desde pantallas basadas en la tecnología OLED, hasta pantallas TFT, pasando por las LCD, cada una de tamaños variados desde 0.96” hasta 3.5”. En concreto las pantallas que se valoraron fueron las siguientes:

-MIKROE-55:

Es una pantalla LCD, que permite mostrar hasta 32 caracteres distribuidos en dos filas. Incorpora un controlador HD44770 [22]. No dispone por sí misma, de un protocolo de comunicación que simplifique el conexionado, por lo que habría que cablear todos los pines de la pantalla al Arduino. Para solventar esto, existe un complemento que posibilita la comunicación pantalla-Arduino a través de I2C. El coste de

Figura 4.57: Pantalla MIKROE-55 [22].

este equipo es el más bajo de la comparativa (7-10€). Sus limitaciones radican en el escaso número de caracteres y ausencia de color debido a su naturaleza.

-VMA437:

Se trata de una pantalla basada en la tecnología OLED de 0.96". Existen variantes monocromáticas o bitono. Dispone de una resolución de 128 x 64 píxeles, lo cual ofrece la posibilidad de poder mostrar gráficas, iconos y hasta 8 líneas de caracteres. Su comunicación es mediante el protocolo I2C. A grandes rasgos este protocolo de comunicación se caracteriza por su escaso conexionado, ya que únicamente necesita dos conductores (figura 4.58). Respecto a su apartado económico, el precio de adquisición es ligeramente mayor que la LCD anterior (10-15€).

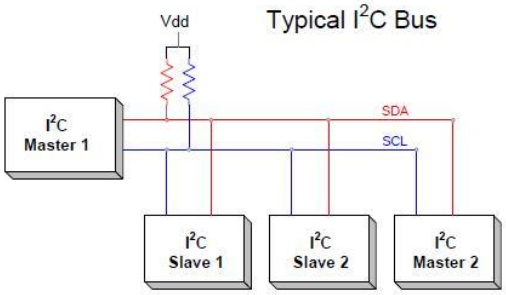


Figura 4.58: Conexionado I2C [23].

Sus limitaciones son, la ausencia de color, y el tamaño reducido. A pesar de esto, es una buena opción, por esta razón el desarrollo inicial estuvo basado en esta pantalla prestada (ver figura 4.59). A pesar de todas sus bondades y debido a la crisis de componentes electrónicos que atañe a este año (2022), se descartó esta pantalla debido a la rotura de stock de este producto.

Figura 4.59: Pantalla OLED 0.96", con la que se empezó el desarrollo de la interfaz.

-VMA412:

Se trata de un display TFT a color de 2.8", su resolución es de 240 x 320 píxeles. Hay versiones que se comunican mediante el bus SPI. La comunicación SPI (del inglés Serial

Peripheral Interface) está basado en una arquitectura maestro-esclavo, donde el maestro puede iniciar la comunicación con uno o varios esclavos, enviando o recibiendo datos de ellos. Esto no sucede a la inversa, es decir, los esclavos no pueden iniciar la comunicación con el maestro ni comunicarse entre esclavos. La comunicación entre maestro y esclavo es full dúplex (puede enviar y recibir datos simultáneamente), ya que la información se transmite por dos líneas distintas (MOSI y MISO).

La comunicación SPI es síncrona, el maestro genera la señal de reloj y se trasmite por un cable al esclavo para mantener ambos equipos sincronizados. Así pues, este bus de comunicación necesita al menos tres cables:

SCK: Señal de reloj

MOSI: (Del inglés Master-Out, Slave-In), envío de información del maestro al esclavo

MISO: (Del inglés Master-In, Slave-Out), envío de información del esclavo al maestro.

Adicionalmente, si existen varios esclavos conectados a un mismo maestro, se necesita escoger el esclavo del cual se quiere recibir o enviar la información, para ello, se utiliza la línea SS (Slave Select), el cual es activo bajo.

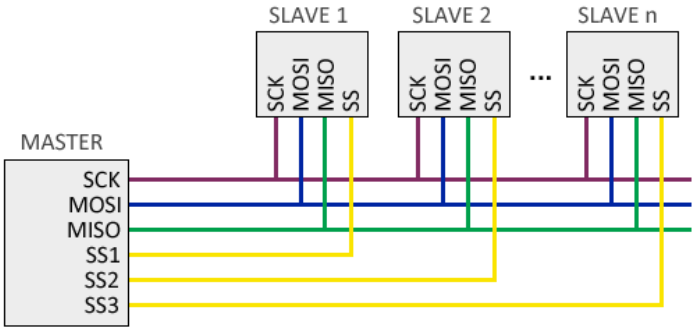


Figura 4.60: Diagrama de conexionado SPI [24].

El funcionamiento es sencillo. En un primer instante el maestro activa la línea SS (valor bajo) del esclavo con el que desea comenzar la comunicación. En cada flanco ascendente de la señal de reloj, el maestro envía un bit al esclavo, al mismo tiempo sucede lo mismo en la otra dirección, es decir el maestro recibe un bit del esclavo activo. La ventaja de este bus de comunicación es su alta velocidad de transmisión (hasta 8 MHz), además de ser full dúplex. Su pega es que a diferencia de I2C, se necesitan 3 cables más y uno adicional por cada esclavo. Este bus de comunicación suele ser utilizado para el slot microSD que suelen incorporar estos modelos para cargar archivos o mostrar imágenes en la pantalla, mientras que para la pantalla utiliza un interfaz paralelo de 8 bits.

Enfocándose en el apartado del precio, es prácticamente el mismo que la pequeña pantalla OLED. Las ventajas decisivas por la que finalmente se selecciona esta pantalla, son la capacidad de representación de colores, junto con la alta resolución (comparado con las otras alternativas) y un tamaño adecuado y vistoso para esta aplicación.

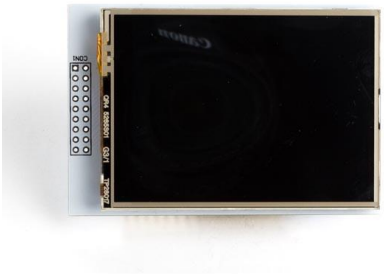


Figura 4.61: Pantalla 2.8" TFT [25].

Con la finalidad de aprovechar todo el potencial que una pantalla de este tipo brinda, se ha decidido implementar varios modos de visualización, fácilmente seleccionables a través del pulsador. Existe un modo de visualización general que aporta la información imprescindible para manejar la planta y sintonizar el PID. La siguiente vista se trata de una gráfica de la salida de la planta (la altura), a través de la cual se puede visualizar el histórico de las medidas, y así poder notar de forma visual e intuitiva aspectos como el sobre impulso u otros conceptos de automática. Finalmente existe una última vista donde se reflejan de forma separada la contribución de cada acción del PID a la señal de control, lo cual puede ser útil para comprender cómo funciona cada sumando del PID y ajustarlo convenientemente. Adicionalmente a estas vistas existen otras vistas secundarias como la pantalla de bienvenida y recomendaciones de uso u otra vista específica para cuando esté seleccionado el modo USB.

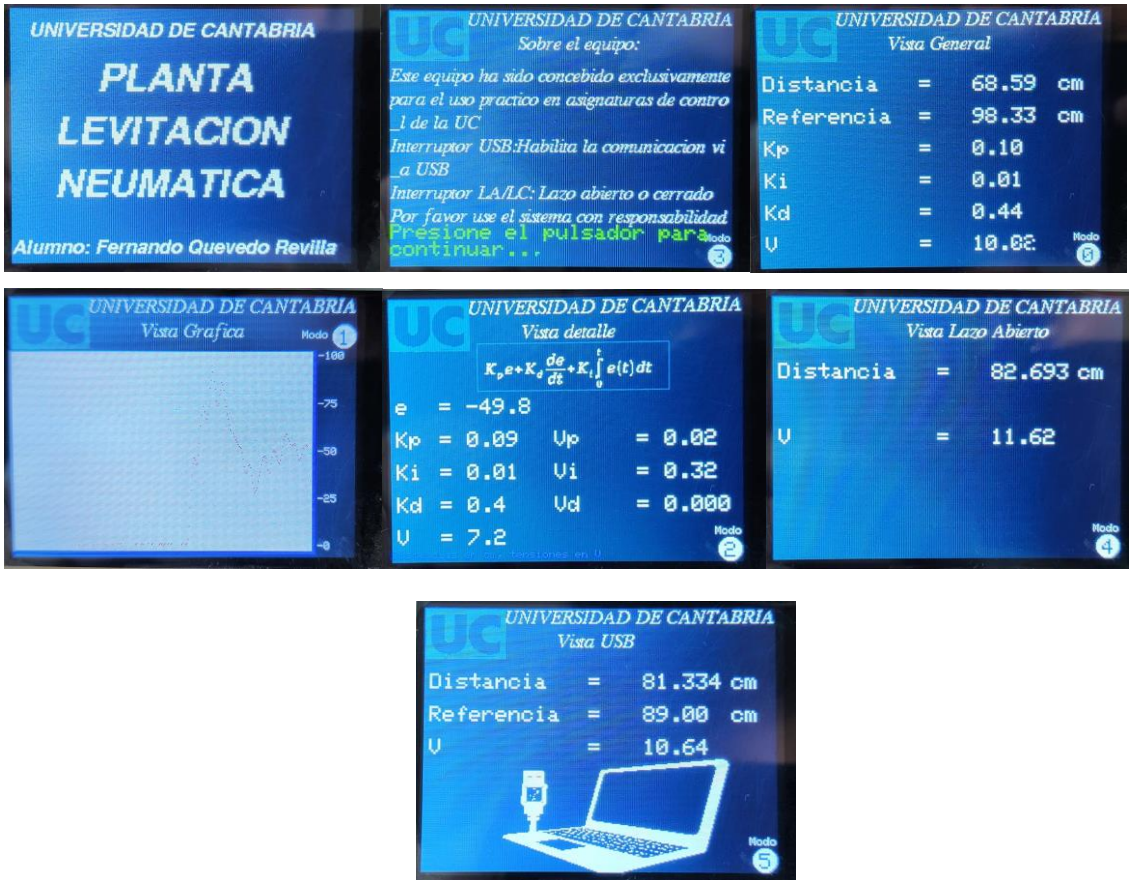


Figura 4.62: Modos de visualización disponibles en la pantalla de la planta.

Por otro lado, existirán 4 potenciómetros de los cuales se tomarán las lecturas fijadas por el usuario referencia, K_p, K_i y K_d (únicamente cuando no se encuentre en modo USB). Para seleccionar el modo de funcionamiento hay dos selectores de palanca, uno para determinar si se desea trabajar en modo USB o no, y el otro para seleccionar si se desea ejecutar el control en lazo abierto o cerrado (únicamente disponible en modo no USB). Para consultar las utilidades de cada modo consular anexo 1. Adicionalmente hay otros dos elementos, otro selector, en este caso para alimentar o no el microcontrolador y el otro elemento de control se corresponde con un pulsador. Dicho pulsador sirve para cambiar el modo de visualización de la pantalla.

Figura 4.63: Resultado final de la consola, con todos los elementos ensamblados.

Unos indicadores luminosos led, de alimentación de la placa Arduino y saturación del actuador, junto con una escala metálica situado a la derecha del tubo termina de redondear una interfaz física funcional, simple y estética. En lo que respecta a los interruptores de control (USB y LAZO ABIERTO) y el pulsador, están conectados a entradas digitales de la placa de Arduino a través de unas resistencias “pull down”. Esta configuración sigue el esquema eléctrico que se muestra en la figura 4.64.

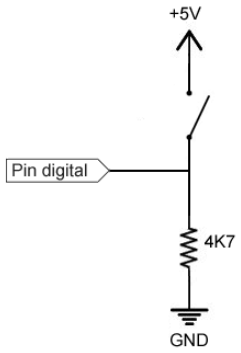


Figura 4.64: Esquema de conexión para entradas digitales en configuración pull-down [26].

Las distintas entradas digitales, están conectadas a tierra a través de la resistencia de $4.7\ k\Omega$, únicamente se detectará una señal alta cuando el interruptor o pulsador se encuentre cerrado (permitiendo el paso de la corriente). Lo que se consigue de esta manera es evitar que cuando el interruptor no esté acoplado, la señal no esté “al aire”, lo que produciría en algunos casos problemas. Del mismo modo, para evitar problemas, esas resistencias se encuentran centralizadas en una PCB pequeña.

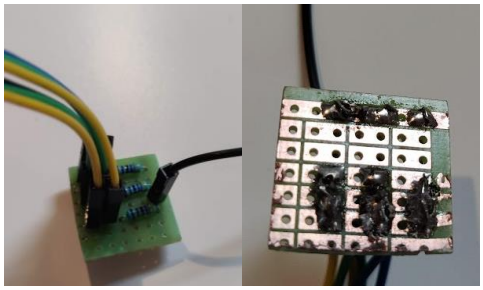


Figura 4.65: PCB Auxiliar para conectar las resistencias pull-down.

En este caso al tratarse de una PCB, sencilla se decide montarla usando una PCB madre pretaladrada. Este tipo de placas disponen por su parte trasera de “matrices” o puntos de cobre que pueden ser conectados como se desee mediante estaño. El cable negro es la tierra, mientras los tres pares de conductores amarillo, azul y verde están conectados por un extremo a cada interruptor / pulsador y por otro lado a cada pin digital de la tarjeta Arduino. Notar que el pulsador no tiene ningún condensador conectado, por lo que, en el capítulo referente al software se deberá tener en cuenta esto para evitar los rebotes procedentes de la conmutación.

Por último, la funcionalidad de los potenciómetros es permitir que el usuario seleccione las ganancias y la referencia del sistema. Para ello, los potenciómetros deberán ofrecer una salida de tensión variable. Un potenciómetro no deja de ser una resistencia de valor fijo sobre la que se desplaza un cursor que divide esa resistencia.

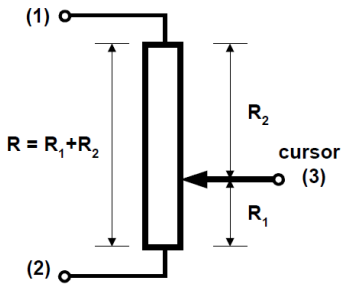


Figura 4.66; Principio de funcionamiento de un potenciómetro. [31].

La manera de obtener una tensión en función de la posición del potenciómetro es conectando sus terminales fijos (en la figura 4.66, los terminales 1 y 2) a una diferencia de tensión, en este caso como son alimentados por Arduino, 5 V. De esta manera dependiendo si el cursor está más cerca o más lejos de cada extremo dará una tensión u otra, siguiendo la expresión de un divisor de tensión.

$$V_{32} = \frac{R_1}{R_1 + R_2} \cdot V_{12} \qquad (4.61)$$

Lógicamente el cursor deberá estar conectado a un pin analógico de Arduino, es decir aquel que ofrezca la posibilidad de digitalizar la señal con un convertidor A/D. El siguiente esquema muestra todas las conexiones necesarias para cada elemento.

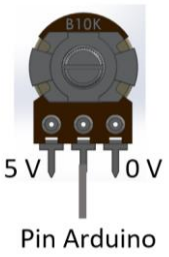


Figura 4.67 Modelo CAD del potenciómetro utilizado, conexionado

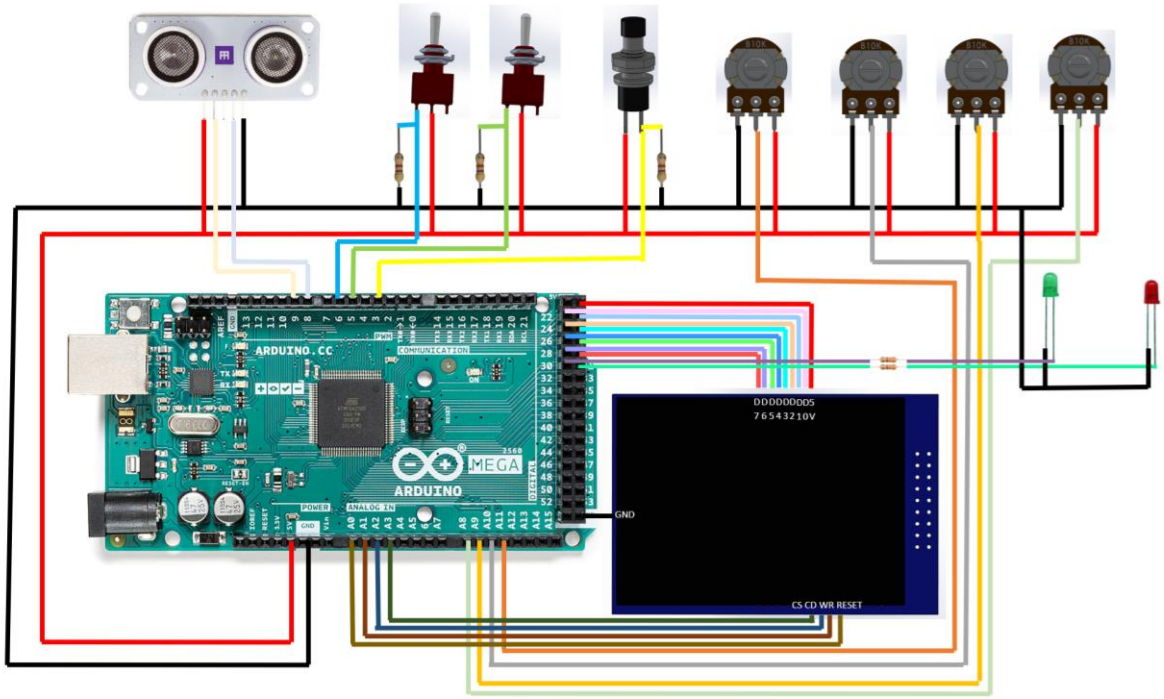


Figura 4.68: Esquema del conexionado en conjunto Arduino-Periféricos.

4.5.2 Interfaz software:

Otro de los objetivos que debe satisfacer la planta, es su posibilidad de poder ser manejada desde un ordenador. Para ello, se pretende realizar una aplicación diseñada específicamente para funcionar con la planta. Las características que debe tener la aplicación, son todas aquellas que sean necesarias para satisfacer los objetivos de la planta en general, los cuales pueden resumirse en: funcionalidad, fluidez, estética y con posibilidad de añadir reguladores con cualquier estructura (no solo limitado al PID). Otro aspecto importante es que esta aplicación está pensada para que los alumnos la descarguen y ejecuten en sus propios ordenadores, por lo que también debe ser compatible con la mayor cantidad de equipos posible, tanto por las limitaciones hardware (principalmente resolución de la pantalla), como software. Para el desarrollo de la aplicación existen en la actualidad multitud de entornos de desarrollo, pero las que más se ajustan a las necesidades del trabajo, son MATLAB y Python. A continuación, se valoran ambas opciones.

Como se ha comentado anteriormente, MATLAB es un software diseñado por y para ingenieros y científicos, además tiene diversas utilidades que lo hacen muy versátil, en el caso que atañe este apartado, se centrará la explicación y forma de programación de la utilidad “appdesigner” de MATLAB. El uso de este entorno de desarrollo de aplicaciones web y de escritorio, está basado en arrastrar cada elemento (menú izquierdo) que compondrán la interfaz a un lienzo o página. Posteriormente, utilizando el menú derecho modificar el comportamiento de ese elemento.

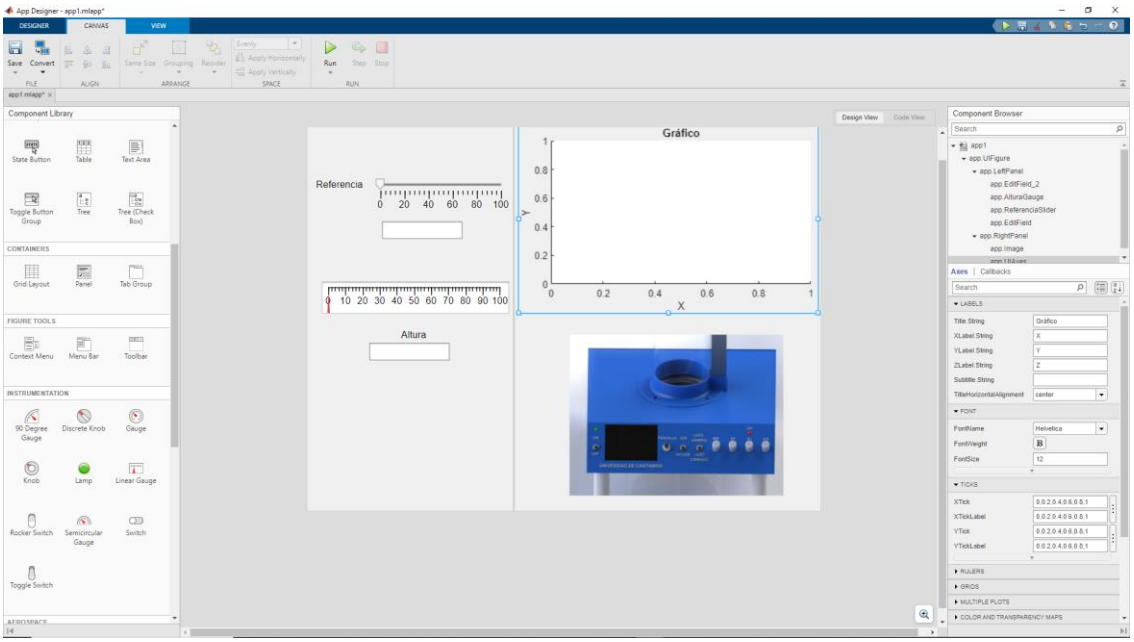


Figura 4.69: Interfaz básica creada en entorno MATLAB.

Por lo que el proceso de creación de la aplicación se podría asimilar a la forma en la que se realizaría un HMI industrial. Esto en parte podría constituir una desventaja, ya que a pesar de que los menús de configuración de cada componente son bastante completos, no nos permite personalizar completamente cada elemento. A cambio, al trabajar en el entorno MATLAB, resultaría más sencillo el análisis de las señales obtenidas por la planta, ya que en parte contiene funciones predefinidas para trabajar con funciones de transferencia, identificación de sistemas…. El lenguaje de programación MATLAB, es un lenguaje cerrado y sujeto al uso de una licencia, lo que implica que no se permite acceder al código fuente de él, para modificar alguna línea de código de alguna función o realizar otro tipo de ajustes.

Por otro lado, Python, a diferencia de MATLAB es un lenguaje enfocado a un propósito general. Es un software libre que permite visualizar y modificar el código fuente, lo cual ofrece un gran abanico de posibilidades. Del mismo modo que sucedía con MATLAB, las aplicaciones no suelen ser creadas tipeando líneas de código, si no que disponen de entornos gráficos que hacen que el desarrollo sea más amigable. Quizás las herramientas más utilizadas para este cometido son QTDESIGNER y THINKER. En este caso se valorará QTDESIGNER, debido al bagaje previo con esta aplicación. QTDESIGNER un software gratuito que, además suele venir incluido al descargar algunos entornos PYTHON como el conocido ANACONDA. El entono de desarrollo de aplicaciones es bastante similar al “appdesigner” de MATLAB, incluso son similares en la distribución de los menús en pantalla. A la izquierda se encuentran los elementos, o como se denominan en este y otros programas “widgets”. A la derecha se puede encontrar la estructura de la aplicación, junto con todos los elementos que la componen. Cuando se clica sobre cada uno de ellos se permite la edición de algunos parámetros del widget.

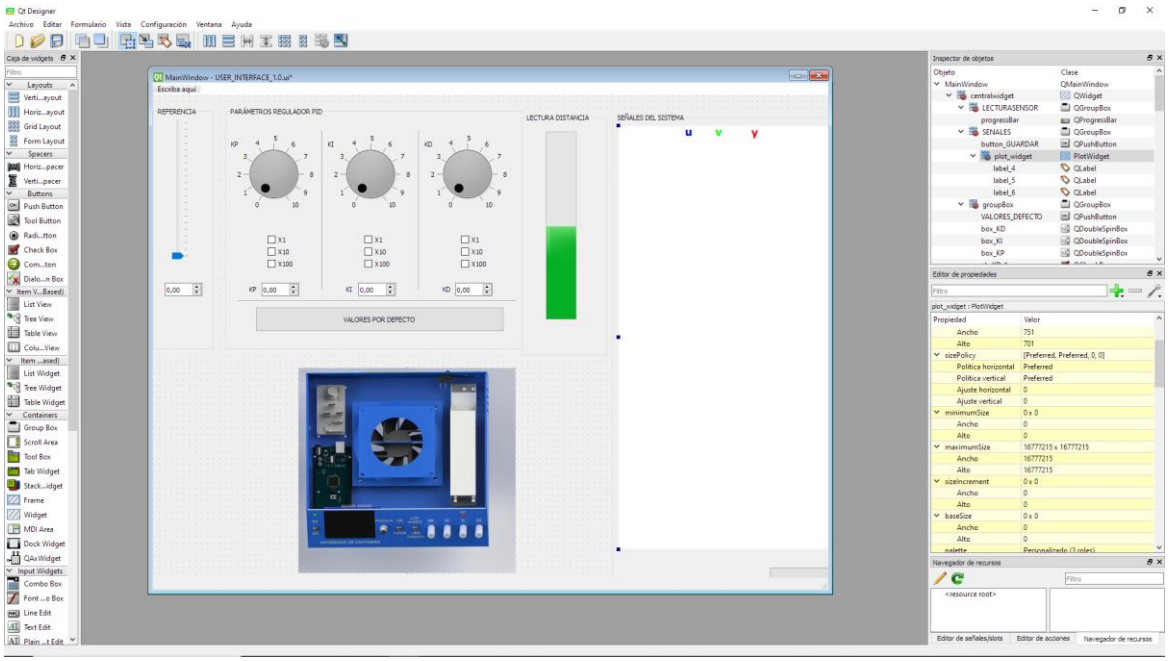


Figura 4.70: Interfaz básica creada en QtDesigner.

Ahora bien, las posibilidades de ajustes, no se terminan aquí, ya que cada elemento tiene otras muchas más propiedades que pueden ser cambiadas, a través de código, bien en el propio QTDESIGNER o en programa PYHTON principal asociado. En la figura 4.71 se muestran ajustes de distintos parámetros a través de QTDESIGNER, y a través del programa principal PYTHON (figura 4.72).

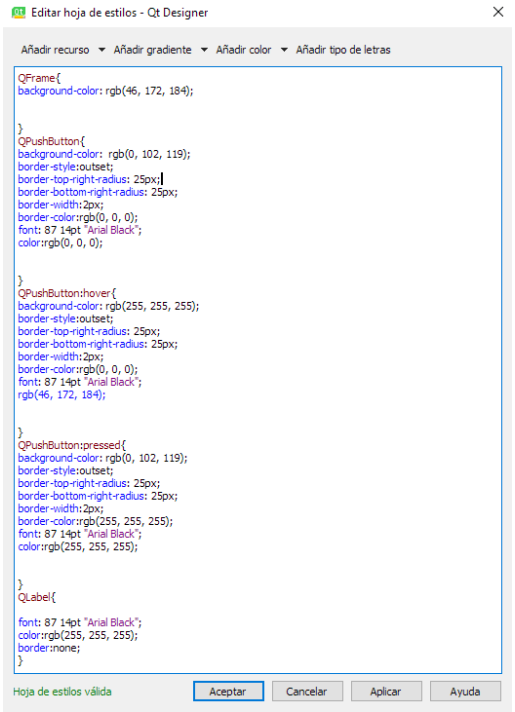


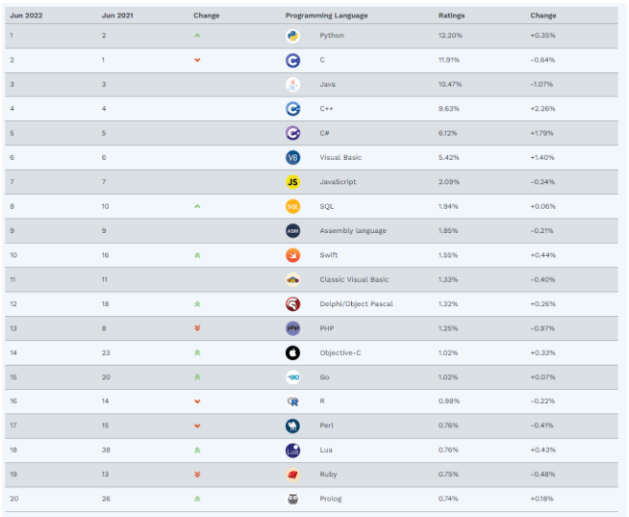
Figura 4.71: Ajuste de campos de un widget, usando la hoja de estilos en QtDesigner.

```
#Ajustes de inicio
self.plot_widget.setBackground('w')
self.plot_widget.setBackground('w')
self.plot_widget.setLabel('left', 'Centímetros')
self.plot_widget.setLabel('right', 'Voltios')
self.plot_widget.setLabel('bottom', 't (s)')
self.plot_widget.setTitle(" ")
self.plot_widget.showGrid(x = True, y = True)
self.plot_widget.setXRange(0.0, 10.0, padding = 0)
self.plot_widget.setYRange(0.0, 105.0, padding = 0)

#otros ajustes....
self.center_Screen()
#eliminar barra superior
self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
self.setWindowOpacity(1)
```

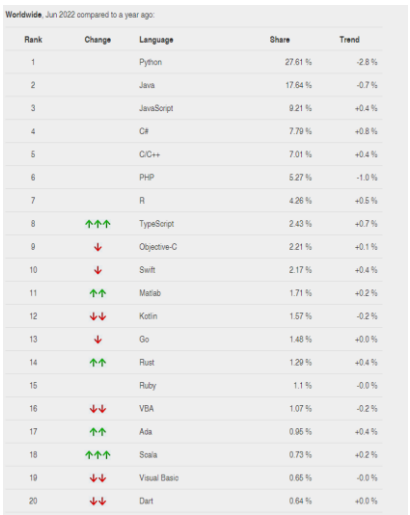
Figura 4.72: Ajuste de campos, desde el programa Python principal.

Esto último solventa la principal limitación de MATLAB, pero el realizar la aplicación basada en Python tiene a su vez otros inconvenientes. El principal, podría resumirse en que algunas librerías no contienen información detallada sobre su uso, a diferencia de MATLAB, quien ofrece en su sitio web oficial información que explica en detalle el uso de cada función (incluso a través de videotutoriales). La otra cara de la moneda de porqué sucede esto, es que realmente esas explicaciones no son del todo imprescindibles, ya que el lenguaje Python este año (2022) ha destronado al que hasta ahora era el número 1, el lenguaje C. Existen multitud de índices que establecen la popularidad de los lenguajes de programación, pero quizás los dos índices más conocidos que reflejan la popularidad de cada lenguaje de programación son el PYPL y el TIOBE. Ambos coinciden en que el lenguaje de programación más popular es Python.



Jun 2022	Jun 2021	Change	Programming Language	Ratings	Change
1	2	^	Python	12.20%	+0.35%
2	1	v	C	11.91%	-0.64%
3	3		Java	10.47%	-1.07%
4	4		C++	9.63%	+2.26%
5	5		C#	6.72%	+1.79%
6	6		Visual Basic	5.42%	+1.40%
7	7		JavaScript	2.09%	-0.24%
8	10	^	SQL	1.94%	+0.06%
9	9		Assembly language	1.85%	-0.21%
10	16	^	Swift	1.55%	+0.44%
11	11		Classic Visual Basic	1.33%	-0.40%
12	18	^	Delphi/Object Pascal	1.32%	+0.26%
13	8	v	PHP	1.25%	-0.97%
14	23	^	Objective-C	1.02%	+0.33%
15	20	^	Go	1.02%	+0.07%
16	14	v	R	0.98%	-0.22%
17	15	v	Perl	0.76%	-0.41%
18	38	^	Lua	0.76%	+0.43%
19	13	v	Ruby	0.75%	-0.48%
20	26	^	Prolog	0.74%	+0.18%

Figura 4.73: Popularidad de los lenguajes de programación según el índice TIOBE [33].



Worldwide, Jun 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	27.61 %	-2.8 %
2		Java	17.64 %	-0.7 %
3		JavaScript	9.21 %	+0.4 %
4		C#	7.79 %	+0.8 %
5		C/C++	7.01 %	+0.4 %
6		PHP	5.27 %	-1.0 %
7		R	4.26 %	+0.5 %
8	↑↑↑	TypeScript	2.43 %	+0.7 %
9	↓	Objective-C	2.21 %	+0.1 %
10	↓	Swift	2.17 %	+0.4 %
11	↑↑	Matlab	1.71 %	+0.2 %
12	↓↓	Kotlin	1.57 %	-0.2 %
13	↓	Go	1.48 %	+0.0 %
14	↑↑	Rust	1.29 %	+0.4 %
15		Ruby	1.1 %	-0.0 %
16	↓↓	VBA	1.07 %	-0.2 %
17	↑↑	Ada	0.95 %	+0.4 %
18	↑↑↑	Scala	0.73 %	+0.2 %
19	↓↓	Visual Basic	0.65 %	-0.0 %
20	↓↓	Dart	0.64 %	+0.0 %

Figura 4.74: Popularidad de los lenguajes de programación según el índice PYPL [34].

Estas estadísticas, en si no indican si un lenguaje es mejor que otro, eso sí, una mayor popularidad generalmente se traduce en una mayor comunidad de usuarios, lo cual puede llegar a suplir en cierto modo la carencia de información oficial sobre algunas funciones.

A estas alturas del documento, no existen características decisivas que hagan inclinar la balanza a favor de MATLAB o de Python. Pero en realidad sí que existe una diferencia clave a favor de Python, y esa es la fluidez. MATLAB, no es precisamente un software ligero, y así se ve reflejado en los requisitos hardware recomendados.

Operating System	Windows 11 Windows 10 (version 1909 or higher) Windows Server 2019 **Note**: Windows 7 is no longer supported Windows Server 2016 is no longer supported
Processor	**Minimum**: Any Intel or AMD x86-64 processor **Recommended**: Any Intel or AMD x86-64 processor with four logical cores and AVX2 instruction set support
RAM	**Minimum**: 4 GB **Recommended**: 8 GB For Polyspace, 4 GB per core is recommended
Storage	3.6 GB for just MATLAB 5-8 GB for a typical installation 31.5 GB for an all products installation An SSD is strongly recommended

Figura 4.75: Requisitos recomendados para MATLAB [27].

Mientras que Python 3, únicamente recomienda 4GB de memoria RAM, 5GB de almacenamiento y prácticamente cualquier procesador relativamente moderno (x86 64-bit CPU (Intel / AMD)) [28]. Por lo que, en base a esto, y a la experiencia previa con MATLAB, quizás la aplicación que se diseñe, así como la comunicación Arduino-PC no se ejecuten en tiempo real, lo cual resultaría inasumible de cara a realizar el control de la planta.

Una vez explicado el motivo por el cual se decide finalmente desarrollar la aplicación con Python, a continuación, se expone los widgets y elementos que componen la interfaz.

Siguiendo la misma línea de diseño, la aplicación tendrá los colores corporativos de la universidad, siempre que lo estético no condicione lo funcional.

La estructura de la interfaz de usuario de la aplicación está basada en un menú de navegación desplegable en el lateral izquierdo. A través de dicho menú se podrá seleccionar la página que se desea visualizar. Los accesos a las páginas son botones con un nombre representativo y acompañados de iconos, lo cual hace que la navegación sea más visual. A continuación, se explica en detalle el contenido y utilidades de cada página.

-Inicio:

Se trata de una página de bienvenida, donde se presenta una vista general de la planta, así como la estructura de la aplicación (ver figura 4.76).

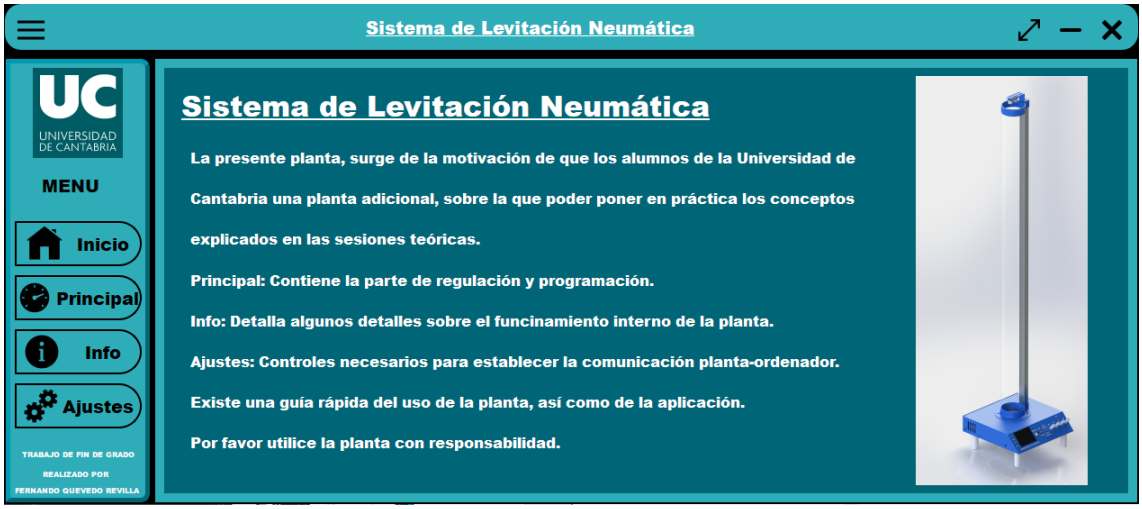


Figura 4.76: Página de inicio de la aplicación.

-Principal:

Es la página a través de la cual se realizará el control propiamente de la planta. Está distribuido en 5 bloques cada uno de ellos con un propósito específico.

Figura 4.77: Página principal de la aplicación.

Funcionamiento: Contiene los botones que inician o paran la comunicación PC-Planta.

Figura 4.78: Bloque de funcionamiento.

Referencia: Constituye la entrada de la planta, es decir, la referencia o consigna a la cual debe levitar la esfera. Existen dos maneras de establecer la consigna, bien de forma numérica haciendo uso de la caja de texto inferior, o bien a través del “slider” que se encuentra en el mismo bloque. Destacar que a esa barra le acompaña una escala numerada.

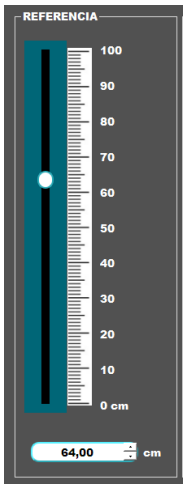


Figura 4.79: Bloque de referencia o consigna.

Lectura distancia: Refleja tanto visualmente como numéricamente la altura de la esfera. Al igual que sucede con la referencia, la barra de altura es acompañada por una escala en centímetros. Pero a diferencia del anterior bloque, este, no contiene ningún elemento editable por el usuario, ya que se trata del valor de la lectura del sensor de la planta.

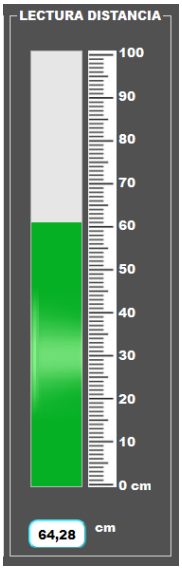


Figura4.80 : Bloque de lectura de la altura de la esfera.

Controlador: Es uno de los bloques más importantes de la aplicación, ya que a partir de él se calculará la señal de control que se envía a la planta. El elemento central de este bloque es una ventana donde se puede editar el código que define el regulador discreto. El código debe de ser escrito siguiendo la sintaxis de Python, así todo, como en algunos grados no reflejan en sus guías académicas este lenguaje, se proporcionan algunos ejemplos explicando cada parte de la que se compone el código. La estructura que se plantea es la siguiente:

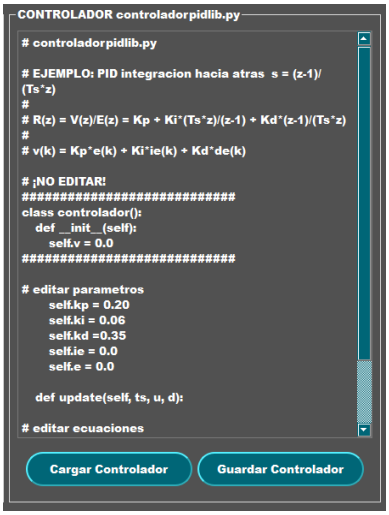


Figura 4.81: Bloque controlador.

Bloque de código 1: Se corresponde con el bloque de comentarios del código, en ella se pretende que se explique la obtención de la ecuación en diferencias del regulador, u otras anotaciones relevantes.

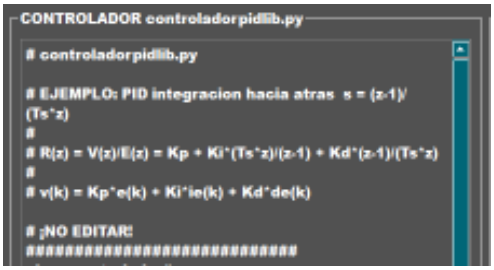


Figura 4.82: Controlador: Bloque de código 1.

Bloque de código 2: En esta parte se define la clase “controlador” y se define su inicio.

```
class controlador():
    def __init__(self):
        self.v = 0.0
##############################
```

Figura 4.83: Controlador: Bloque de código 2.

Bloque de código 3: A estas alturas del código se definen las constantes del regulador, en el ejemplo inferior se muestra las ganancias de un regulador PID.

```
# editar parametros
      self.kp = 0.20
      self.ki = 0.06
      self.kd =0.35
      self.ie = 0.0
      self.e = 0.0
```

Figura 4.84: Controlador: Bloque de código 3.

Bloque de código 4: La última parte, es en sí donde se programa el regulador (campo “update”).

Siguiendo con el mismo ejemplo del regulador PID, en la figura 4.85, se reflejan las ecuaciones. Como se puede observar, la sintaxis que definen ecuaciones es bastante simple.

```
def update(self, ts, u, d):

# editar ecuaciones
    e = u - d
    self.ie += e*ts
    de = (e - self.e)/ts
    self.v = self.kp*e + self.ki*self.ie + self.kd*de
    self.e = e
```

Figura 4.85: Controlador: Bloque de código 4.

Por lo que, se puede decir que está diseñado con la finalidad de que cualquier alumno, sea capaz de programar su propio regulador independientemente de sus conocimientos previos de Python. Aunque si el público, está en su fase inicial de aprendizaje se permite cargar controladores predefinidos, como por ejemplo un PID básico o el código necesario para el manejo de la planta en lazo abierto. Cargar el controlador es tan fácil como clicar sobre el botón “Cargar Controlador”, al instante aparecerá una ventana emergente a la que se le debe indicar el “path” y nombre del controlador que se desea abrir.

Figura 4.86: Botones de cargar y guardar controlador.

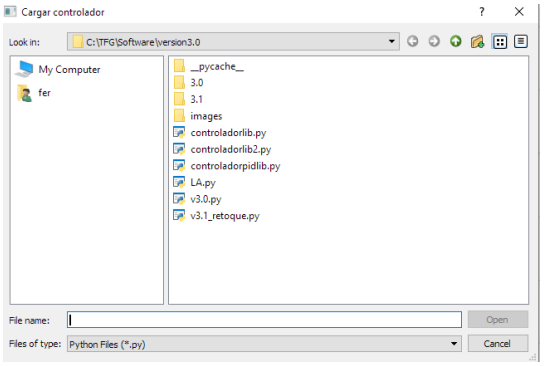


Figura 4.87: Emergente: Cargar controlador.

Tras realizar alguna modificación, sobre el código o crear un código nuevo, se puede guardar, presionando el botón “Guardar Controlador”. El procedimiento es similar al proceso de cargar,

pero en este caso se debe especificar el lugar donde se desea guardar, así como el nombre del archivo (incluyendo la extensión “.py”).

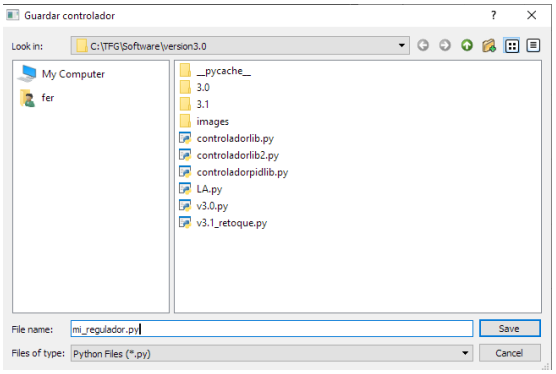


Figura 4.88: Emergente: Guardar controlador.

El último bloque que compone la página principal es un gráfico de dimensiones generosas. En este se representan las principales señales del sistema, como son la señal de entrada (u), señal de control (v) y finalmente la señal de altura de la bola (y). También, se permite exportar las señales a otros entornos como MATLAB una vez parada la planta. Remarcar que el eje de ordenadas refleja dos magnitudes, por un lado, las distancias están en centímetros y la tensión de la señal de control en voltios.

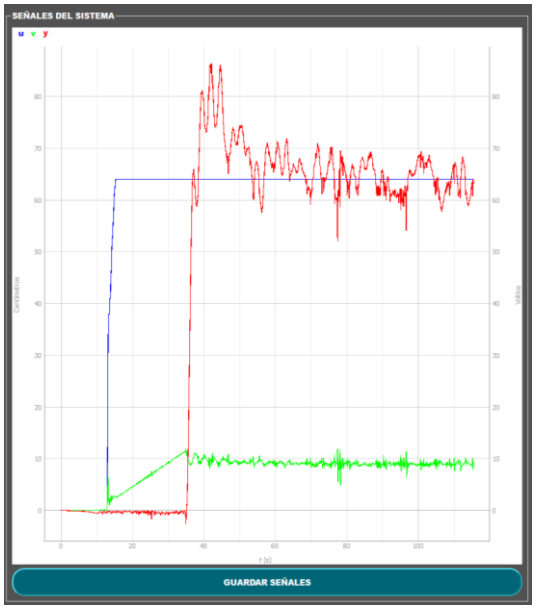


Figura 4.89: Vista del gráfico principal de la aplicación.

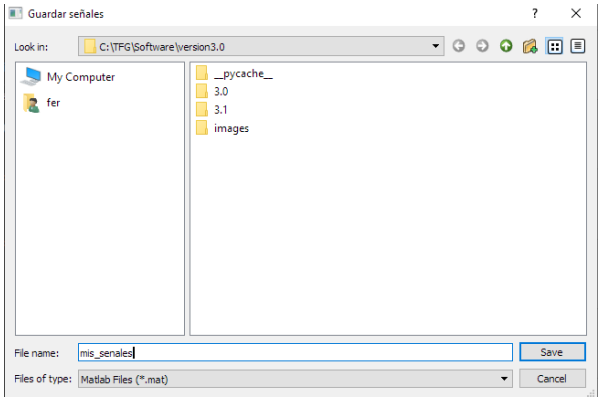


Figura 4.90: Emergente: Guardar señales.

-Información:

La siguiente pantalla está dedicada especialmente para el público más curioso que desea conocer más detalles sobre el funcionamiento interno de la planta. Se explica en líneas generales, algunos aspectos concretos que no son imprescindibles conocer para el manejo de la planta. Además, se incluye una imagen de su construcción y elementos internos.

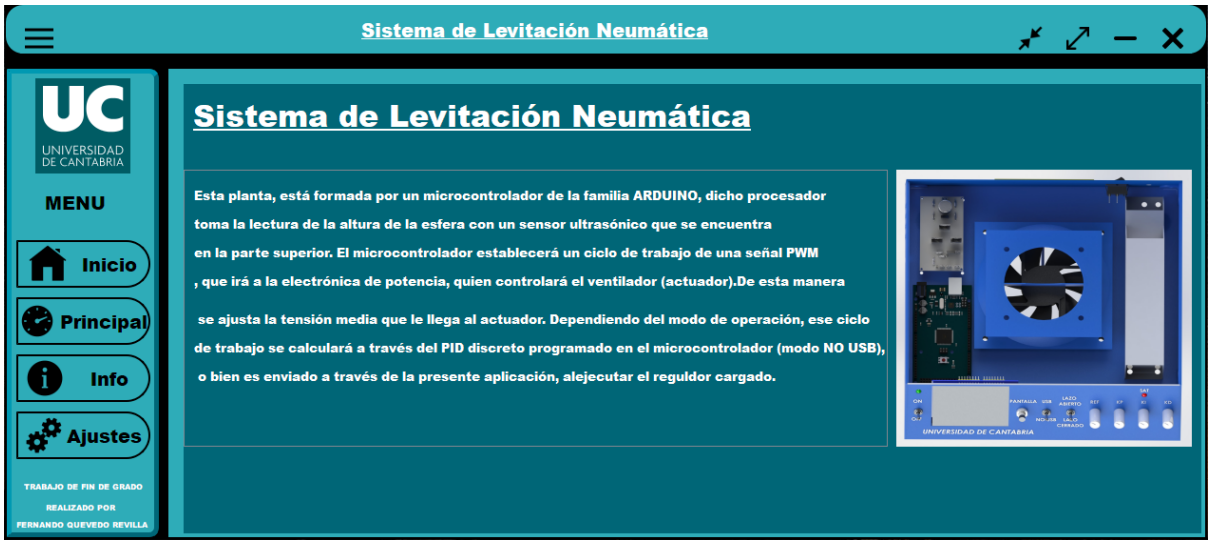


Figura 4.91: Página de información.

-Ajustes:

Finalmente, la última página, contiene los ajustes necesarios para establecer la comunicación Planta-PC. Los parámetros característicos que definen la comunicación serial son el puerto COM donde está conectada la planta, y la tasa de baudios (del inglés baud rate). Con el fin de simplificar al máximo este proceso, se ha incluido un botón “BUSCAR” que rastrea y encuentra los dispositivos conectados al ordenador, por lo que si solo se tiene conectado la planta, el proceso es completamente automático para el usuario.

Además, se incluye una imagen, donde se especifica la posición de los interruptores de palanca necesarias para activar el modo USB. También se especifica que el interruptor de encendido que alimenta el microcontrolador debe de estar en la posición “ON”. Esto último puede resultar obvio, pero se ha incluido a modo de poka-yoke, ya que, al conectar la planta al ordenador, esta enciende la pantalla independientemente del estado del interruptor de alimentación. Esto es así ya que la conexión USB aporta los 5V necesarios para poner en marcha el microcontrolador y los periféricos asociados (no la parte correspondiente a la electrónica de potencia).

Figura 4.92: Página de ajustes.

Un último aspecto que remarcar sobre la aplicación, es su compatibilidad con distintas resoluciones de pantalla, ya que cada elemento adapta (dentro de unos límites) su tamaño en función de la resolución de la pantalla donde se muestra la aplicación. Para una experiencia de usuario buena, se recomienda una resolución mínima de 1152 x 864 píxeles, a pesar de que no se ha decidido implementar ningún algoritmo que impida su uso en pantallas de menor resolución. En el otro límite, se ha testado la aplicación en monitores de hasta 4K (3840 x 2160 píxeles), pero la única ventaja que ofrece usar pantallas de tan alta resolución únicamente radica en el tamaño del gráfico, ya que el tamaño de los botones está limitado.

Figura 4.93: Aplicación en pantalla completa, en monitor con la resolución mínima recomendada (1152x864 píxeles).

4.6 SOFTWARE:

Este capítulo se dividirá a su vez en dos partes, la primera explicará el código cargado en la placa Arduino, mientras la segunda se centra en el código Python junto con la interfaz de usuario asociada. Para ilustrar las explicaciones se incluirán los flujogramas de cada código, así como una explicación detallada de cada proceso.

4.6.1 Código Arduino:

El flujograma de trabajo de este código es el siguiente.

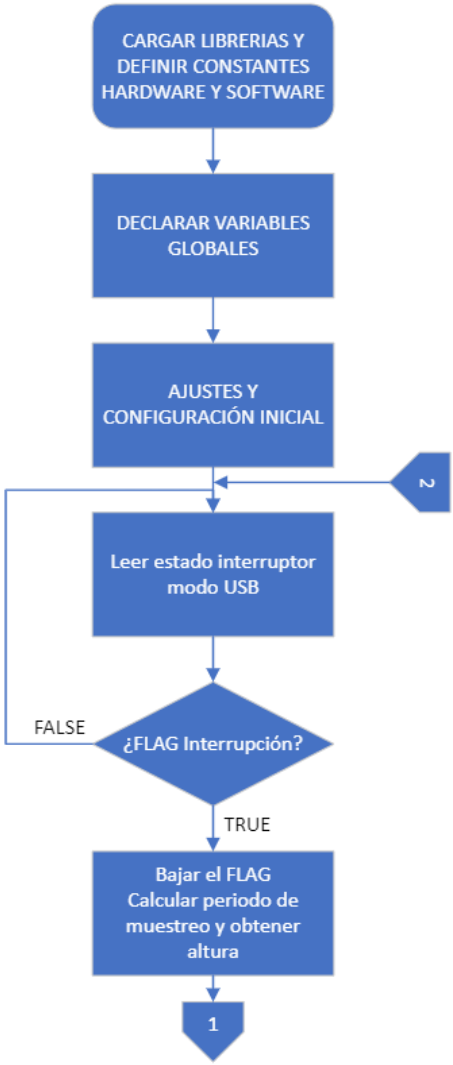


Figura 4.94 Flujograma principal del código Arduino (I).

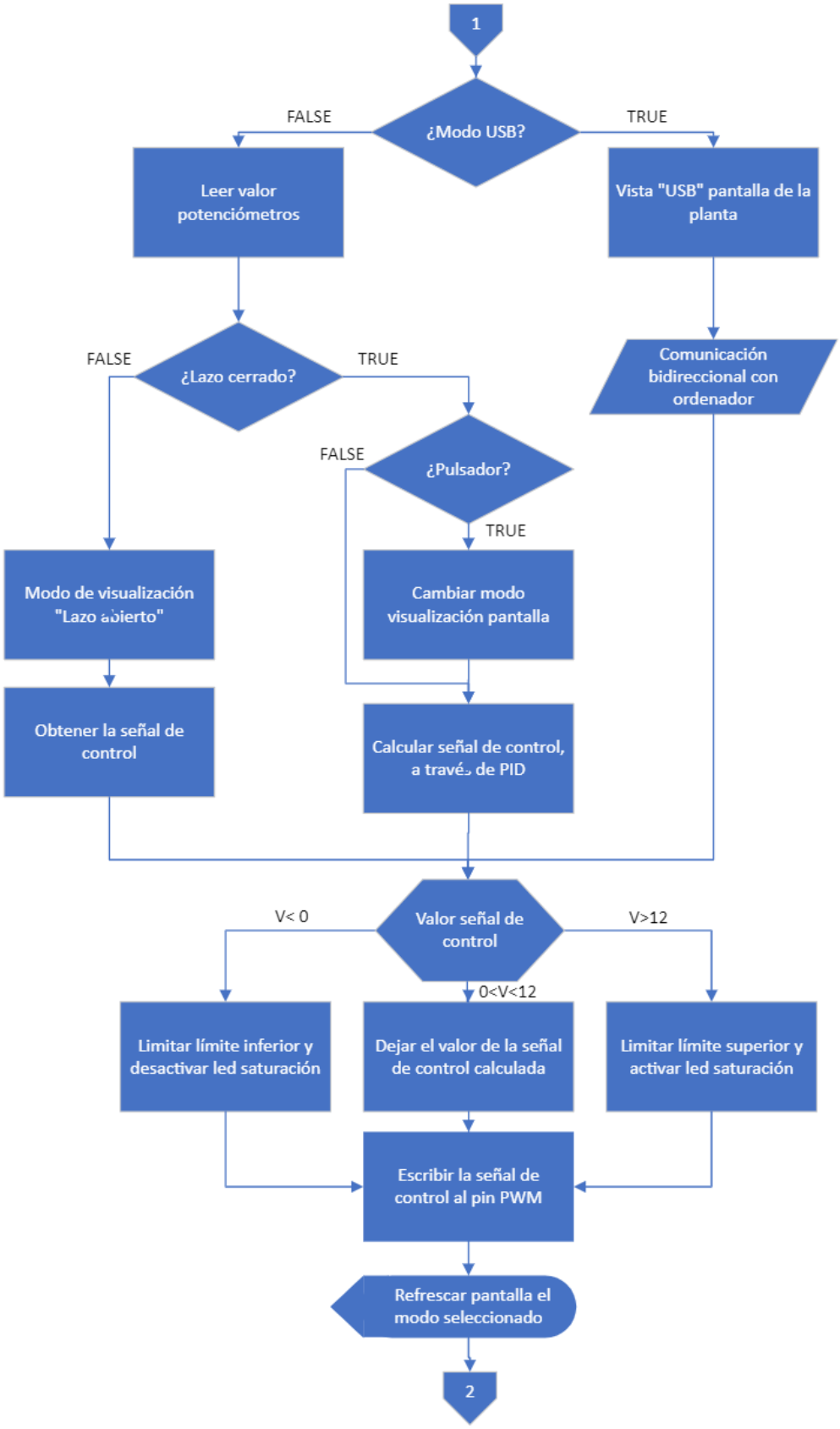


Figura 4.95: Flujograma principal del código Arduino (II).

Como se observa en el diagrama de flujo previo, la primera acción que se lleva a cabo es cargar las librerías necesarias. En este caso se utilizan las siguientes:

- Adafruit TFTLCD y GFX: Contiene las funciones necesarias para establecer la comunicación Arduino-Pantalla. Por otro lado, la librería GFX ofrece funciones que ayudan a representar figuras geométricas, imágenes …etc.
- Fuentes de texto: Por otro lado, se incluyen algunos tipos de letras, usadas para la escritura de algunos textos en pantalla.
- TimerOne: Esta librería, simplifica el uso de interrupciones.

La manera de cargarlas es simplemente, descargarlas a través del administrador de bibliotecas de Arduino, para posteriormente incluirlas en nuestro programa con el comando "#include".

```
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <Fonts/FreeSansBoldOblique9pt7b.h>
#include <Fonts/FreeSerifItalic9pt7b.h>
#include <TimerOne.h>
```

Figura 4.96: Código Arduino: Incluir librerías.

Por otro lado, se definen constantes relativas a dónde están conectados los periféricos. Por ejemplo, las dos primeras líneas de código que se muestran en la siguiente figura (4.97), se corresponden a la localización de los pines echo y trigger del sensor ultrasónico. Este paso no es totalmente necesario, ya que en el código se puede sustituir el nombre de la constante por su valor, pero facilita tanto las modificaciones futuras como el mantenimiento de la planta.

```
//Constantes hardware
#define trigger 8           // Trigger pin for ultrasonic sensor
#define echo 9              // echo pin for ultrasonic sensor
#define v_pin 7             // Control signal PWM
#define ref_pot A8          // Analog input for  pot
#define kp_pot A9           // Entrada analógica
#define ki_pot A10          // Entrada analógica
#define kd_pot A11          // Entrada analógica
#define power_on_led 30     // Led alimentacion Verde
#define sat_led      31     // Led Saturacion PID Rojo
#define pulsador      3     // Pulsador cambio de modo pantalla
#define interruptor_USB 5   // Interruptor modo USB
#define interruptor_REALIMENTACION  6 // Interruptor lazo cerrado o abierto
//Hardware pins para display TFT
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0
#define LCD_RESET A4 // TFT reset
```

Figura 4.97: Código Arduino: Constantes hardware.

Para concluir con la definición de las constantes, es necesario definir constantes software. En este caso, se han incluido 3 imágenes (logotipo de la universidad, expresión PID y el icono del modo USB) y una lista de colores básicos. La forma de definir las imágenes es únicamente transformar la imagen en formato de mapa de bits, y después declararla en el código Arduino.

Al ser una constante no tiene sentido que esa variable se almacene en la memoria SRAM, por lo que para almacenarlo en la memoria flash se introduce la palabra clave PROGMEM.

```
// Constantes relativas a software
// Universidad de Cantabria logo
#define LOGO_WIDTH 74    // Anchura del logo UC
#define LOGO_HEIGHT 48   // Longitud del logo UC
const unsigned char PROGMEM UC[]={
```

Figura 4.98 Código Arduino: Constantes software.

El siguiente proceso que se lleva a cabo es declarar las variables globales (aquellas deben tener acceso todas las funciones). Para ello, se precederá el tipo de la variable al nombre de esta, tal y como se muestra en la siguiente captura.

```
//Variables globales
float tiempo,distancia,e_act,e_prev=0,de,vp,vi,vd,error_suma=0,v;
float ref=0,kp=0,ki=0,kd=0;
float v_volts; // Tensión del actuador en voltios
//Variables globales de pantalla
int modo=3,modo_max=3,modo_init=1,modo_anterior=2; // Administrar modos
int M1[275],M2[275];                               // Gráfico
int blink_signal;                                  // Parapadeo

// Variables interruptores y pulsador
bool USB_POT=0,LA_LC=0;

//Variables interrupciones y cálculo del periodo de muestreo
volatile unsigned int interrupt = 0;
unsigned long tsamp = 10*1000;
float ts = 0;
float other=0;
unsigned long t = 0;
unsigned long t0 = 0;
```

Figura 4.99: Código Arduino: Variables globales.

Es importante que estas variables tengan un nombre fácilmente identificable ya que, de esta manera se facilita la modificación de código por parte de terceros, si se desea realizar alguna modificación futura. De esta manera, aunque las variables sean más largas, el esfuerzo en realidad es menor, al evitar tener que comentar lo que representa esa variable allá donde se use.

Por último, aunque no es una variable como tal, se define la instancia “tft” a la cual se hará referencia cada vez que se pretenda actuar sobre la pantalla.

```
// Instancia de la pantalla
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);
```

Figura 4.100: Código Arduino: Instancia de la pantalla.

Tras ejecutar esto, Arduino salta a la función de ajustes iniciales, que comúnmente se le conoce como “setup”. En esta función se configuran los pines (entrada o salida), se inicializarán las variables que así lo requieran, y en este caso, se mostrará una pantalla de bienvenida, así como una secuencia de parpadeo de los diodos led, indicando que la planta se encuentra lista para su uso. La configuración de los pines se realiza con el comando “pinMode”, cuyos argumentos de entrada son la localización del pin, y si es entrada o salida.

```
//Configuracion de los pines
pinMode(trigger,OUTPUT);
pinMode(power_on_led,OUTPUT);
pinMode(sat_led,OUTPUT);
pinMode(echo,INPUT);
pinMode(ref_pot,INPUT);
pinMode(kp_pot,INPUT);
pinMode(ki_pot,INPUT);
pinMode(kd_pot,INPUT);
pinMode(pulsador,INPUT);
```

Figura 4.101: Código Arduino: Configuración de los pines.

Posteriormente se inicializan las variables que necesiten tener un valor determinado. Para esta aplicación se ha desactivado el pin de disparo del sensor, y limpiado las matrices correspondientes a la representación de la gráfica.

```
// Inicializar variables
digitalWrite(trigger,0);
digitalWrite(power_on_led,1);
digitalWrite(sat_led,1);
for(int i=275; i>=0;i--){ // matrices de enteros para la representación gráfica
        M1[i]=200;
        M1[i]=200;
}
```

Figura 4.102: Código Arduino: Inicializar variables.

También es en esta altura del código, cuando se enciende el led verde, así como el led de saturación, con el fin de realizar un juego de luces al inicio. Tras esto, se inicia el puerto serie. Después se busca con el comando el controlador de la pantalla (en este caso el ILI9341), para finalmente iniciar la pantalla.

```
Serial.begin(2000000); //Iniciar comunicación serial
tft.reset();           // Reset display
uint16_t identifier = tft.readID(); // Leer el identificador de la pantalla
tft.begin(identifier);              // iniciar la pantalla
```

Figura 4.103: Código Arduino: Inicio comunicaciones.

Otra configuración que se realiza en esta función es la de iniciar y configurar la subrutina de atención a las interrupciones. En este caso cuando haya trascurrido el periodo de muestreo (tsamp), se ejecutará la función “interrupt_routine”, donde únicamente se activará el “flag” de interrupción.

```
//Iniciar y configurar el timer
Timer1.initialize(tsamp);
Timer1.attachInterrupt(interrupt_routine);
```

Figura 4.104: Código Arduino: Configuración Timer1.

Una vez terminada la configuración, comienza la secuencia de bienvenida. Se llama a la función “initDisplay”, que representa en la pantalla la página de bienvenida, y tras ello los diodos led parpadean.

```
//Secuencia de bienvenida
digitalWrite(power_on_led,0);
digitalWrite(sat_led,0);
delay(200);
digitalWrite(power_on_led,1);
digitalWrite(sat_led,1);
delay(200);
digitalWrite(sat_led,0);
```

Figura 4.105: Código Arduino: Secuencia de parpadeo inicial leds.

Tras realizar las animaciones de bienvenida, la placa Arduino estará ejecutando el bucle principal. Donde se estará leyendo permanentemente el estado del interruptor USB, a no ser que el “flag” de interrupción este activo, indicando que ha trascurrido el periodo de muestreo. En ese caso se realizarán los siguientes procesos.

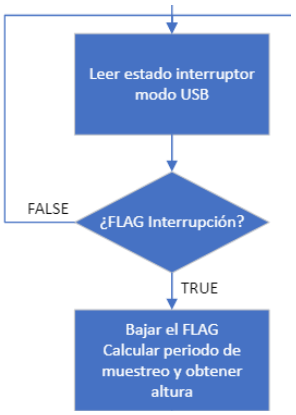


Figura 4.106: Bucle mientras no se active el flag.

Lo primero, es calcular el periodo de muestreo para comprobar que Arduino está funcionando de manera correcta y que no está bloqueado.

Después se desactiva el “flag” de interrupciones, dejándolo preparado para la siguiente vez que salte la interrupción. Tras ello, se llama a la función muestrea distancia, quien devuelve el valor de la altura de la esfera. Una vez obtenida esa medida, en función del estado del interruptor que controla el modo USB, se ejecutará un bloque de código u otro.

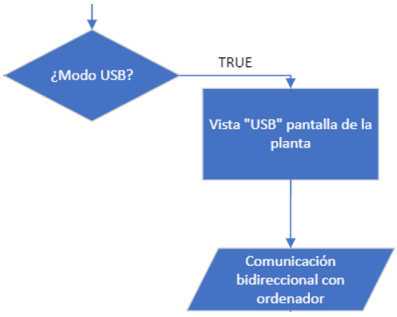


Figura 4.107: Procesos a realizar en modo. USB

Si en el momento que saltó la interrupción el interruptor estaba seleccionando el modo USB, se procede a la comunicación bidireccional Arduino-Ordenador, además de seleccionar el modo de visualización de pantalla exclusivo para este modo. Para la comunicación se han creado dos funciones, una para leer del puerto serial y otra para su escritura.

```
if(USB_POT==0){ //Si estoy en modo usb interfaz USB_POT
   // MODO USB
   modo=5; // Mostrar el modo de visualización específico para USB
   read_Serial(); // Leer puerto serie
   write_serial(ts,v*12.0/255,distancia,other);//Escribir puerto serie
}
```

Figura 4.108: Código Arduino: Llamada a las funciones en modo USB.

La función de lectura envía un carácter (‘s’) que indica que se comienza la comunicación, sirve para sincronizar las dos partes de código. Por otro lado, la lectura se realiza leyendo los bytes disponibles en el puerto serie de 4 en 4, ya que son variables tipo float de 32 bits, luego los primeros cuatro bytes son leídos y almacenados en la dirección de memoria de la primera variable, y además se borran del puerto serie, quedando los otros cuatro disponibles para la siguiente variable, y almacenándolos en la dirección de memoria a la que apunta cada variable. Un detalle es que nada más recibir la señal de control en voltios esta función la convierte a ciclo de trabajo. La estructura en el puerto serie es exactamente la operación inversa, en este caso se escriben los 4 bytes de donde apunta la dirección de la variable que se pretende enviar. La escritura del puerto serie termina con la instrucción “Serial.flush”, que espera hasta que la transmisión de los datos se complete.

```
void read_Serial(){
 Serial.write('s');
 Serial.readBytes((byte*)&ref, sizeof(float));
 Serial.readBytes((byte*)&v_volts, sizeof(float));
 v=(int)floor(v_volts/12*255);
 }
void write_serial(float ts, float s_control, float salida, float other){
 Serial.write((byte*)&ts, sizeof(float));
 Serial.write((byte*)&s_control, sizeof(float));
 Serial.write((byte*)&salida, sizeof(float));
 Serial.write((byte*)&other, sizeof(float));
 Serial.flush();

 }
```

Figura 4.109: Código Arduino: Funciones de comunicación serial.

En el caso contrario (que se encuentre seleccionado el modo NO USB / autónomo), se toma la lectura analógica de los potenciómetros, con la función “read_pot”, posteriormente se comprueba si se desea ejecutar un control en lazo cerrado o en lazo abierto.

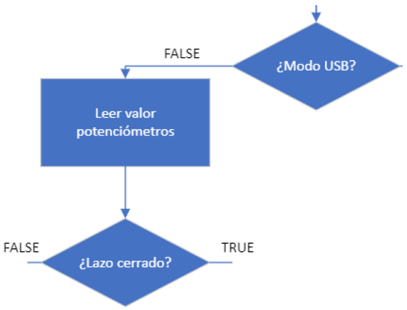


Figura 4.110: Paso actual del diagrama de flujo.

En el caso de que se desee ejecutar el lazo cerrado, se comprueba el estado del pulsador, si este se encuentra activo, se aumenta en uno el valor de la variable que escoge el modo, y posteriormente se para el programa durante solo 10ms para evitar el problema de los rebotes. Tras varios ensayos, el aumentar 10 ms el tiempo de muestreo únicamente cuando se detecta el pulsador, no implica ningún efecto adverso para el sistema.

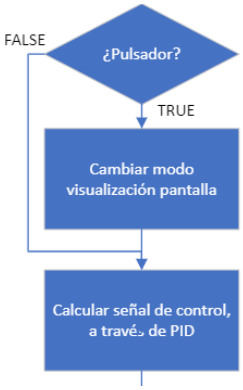


Figura 4.111: Acciones dependiendo del estado del pulsador.

```
if(LA_LC==1){
  // MODO LAZO CERRADO
  bool P=digitalRead(pulsador);// Leer estado pulsador
  if(P==1){ // Si esta pulsado ...
    modo=1+modo; // Cambio el modo de visualización modo++
    if (modo>=modo_max){// Si el modo es mayor que el modo max
      modo=0;//Volver al primer modo
    }
    delay(10); // Parar durante milisegundos
  }
```

Figura 4.112: Código Arduino: Modo Lazo cerrado, no USB, lectura del pulsador.

Tras haber o no modificado el modo de visualización, se ejecuta el código del PID discreto que se muestra en el capítulo 3 de este documento.

```
//Regulador PID
e_act=ref/100*12-distancia/100*12; // error en voltios
de=(e_act-e_prev)/ts;              // derivada del error
error_suma +=ki*e_act;             // Accion integral
e_prev=e_act;                      // Preparar para el siguiente paso
v_volts=kp*e_act+error_suma+kd*de; // Señal de control en voltios
v=int(v_volts/12*255);             // Señal de control en ciclo de trabajo
```

Figura 4.113: Código Arduino: PID discreto.

Si por otro lado se desea ejecutar el control en lazo abierto, lo único que se hace es escalar la referencia para pasar de centímetros de la consigna a ciclo de trabajo (o a efectos prácticos tensión media).

```
else{
  // LAZO ABIERTO
  v=round(ref/100*255);            // La señal de control es igual que la referencia (escalada)
  modo=4;                          // Fuerzo el modo de visualización al específico de lazo abierto
  }
```

Figura 4.114: Código Arduino: Señal de control, Lazo abierto.

Tras haber calculado la señal de control, bien a través de la expresión del PID, o bien a partir de la medida del potenciómetro de referencia, o proveniente del ordenador junto con la aplicación, hay que limitarla si está fuera de rango.

La forma en la que se limita, es forzando el ciclo de trabajo de la señal PWM a que este entre un valor máximo y mínimo. El ciclo de trabajo de la señal PWM con la que trabaja esta versión de Arduino es en un rango de 0 a 255 ya que tiene una resolución de 8 bits), si se supera este valor, se forzará a 255, y en el caso contrario si el ciclo de trabajo negativo, se limita a cero. También se encenderá o apagará el led de saturación respectivamente.

```
// Limitar y administrar el led de saturación
if (v>255){
 v=255;
 digitalWrite(sat_led,1);
 }
else if (v<0){
 v=0;
 digitalWrite(sat_led,0);

}
else {
 digitalWrite(sat_led,0);
}
```

Figura 4.115: Código Arduino: Limitación del ciclo de trabajo y administración de los leds.

Tras esto, se escribe el valor del ciclo de trabajo limitado en el pin correspondiente y se llama a la función que controla los modos de visualización de la pantalla. La programación asociada a la visualización datos en la pantalla, se puede clasificar como una programación de alto nivel, ya que en concreto se ha utilizado la librería de manejo que adjunta el fabricante (adafruit_GFX). De esta librería, las funciones más utilizadas son las que se detallan a continuación:

- setCursor(x0, y0): Su cometido es indicar dónde se debe empezar a representar una figura o un texto. Los argumentos de entrada se corresponden con la coordenada “x” e “y” de la pantalla física.
- setTextSize(int): Establece un factor de escala a un texto, por defecto 1.
- println(char): Escribe el texto indicado como argumento de entrada, en la posición definida por el cursor. Admite introducir variables.

- drawRect(x0, y0, ancho, alto, color): Representa un rectángulo con un vértice en la posición “x0” y “y0”, con el ancho y largo indicado.
- tft.drawBitmap(x0,y0,variabe_mapabits,ancho,alto,color): Representa un mapa de bits, comenzando desde la posición “x0” y “y0”.

Carece de sentido explicar en detalle la programación de cada modo de visualización, ya que son los mismos comandos generalmente (se utilizan algunos más de los comentados previamente), encadenados por secuencias “swich”, y condicionales. Dos aspectos que resultan interesantes de comentar, es la estructura general de esas instrucciones. Con el fin de optimizar los recursos consumidos por la pantalla, se ha ideado un sistema de fondos o plantillas para evitar tener que representar todos los píxeles de la pantalla cada vez que se necesite refrescar un dato. De este modo el formato, fondo, y etiquetas fijas sólo se cargarán la primera vez que se muestre el modo de visualización concreto. Únicamente se refrescarán los píxeles que estén asociados a campos (tanto textos, como gráficos) variables. A continuación, se resume el modo de funcionamiento de este sistema a través de un diagrama de flujo.

Figura 4.116: Flujograma general de funcionamiento de la función refresh_display.

A pesar de todas las virtudes de la librería “adafruit_GFX”, esta no incluye una función de gráfico. Por esa razón hubo que diseñarlo desde cero. De cara a la pantalla una gráfica no

deja de ser de nuevo una plantilla o fondo, al que encima se le colocan una serie de medidas, en este caso píxeles. Esto en si, no resulta complejo teniendo en cuenta los modos de visualización previos, la característica que lo complica es que la gráfica sea dinámica, es decir, que se vaya actualizando conforme trascurre el tiempo. Los elementos estáticos en este caso son la cabecera con el logotipo de la universidad, los textos correspondientes al modo de visualización, una escala numerada y finalmente un lienzo blanco con sus ejes.

Figura 4.117: Elementos estáticos modo visualización de la gráfica.

Sobre este fondo hay que representar las medidas, pero todas tienen que ser almacenadas, ya que en todo el gráfico se muestran 275 muestras. Para almacenarlas, se han creado dos arrays de enteros, donde cada índice se corresponde con una columna de píxeles del gráfico, y el valor, es la medida de la distancia en centímetros.

```
case 1:
  //Gráfica
  tft.fillRect(0, 49,280, 185, WHITE);
  M1[275]=(100-distancia)/100*188+40;
  for(int i=275; i>=1;i--){
    tft.drawPixel(i,M1[i],RED);
    tft.drawPixel(i,M2[i],WHITE);
    M2[i-1]=M1[i];

    }
  for(int i=275; i>=0;i--){
    M1[i]=M2[i];
    }
break;
```

Figura 4.118: Código Arduino: Gráfico.

La razón por la que se han utilizado dos arrays en vez de uno, es que es necesario borrar las medidas previas (en realidad representarlas de color blanco) y a la vez representar los nuevos valores para cada columna.

Es cierto que se podría haber realizado esta misma labor con sólo una variable, pero en cambio, se ha implementado este método y resulta que ralentiza el tiempo de ejecución de esta función.

Se adjunta en el anexo 3 el código Arduino completo con las funciones a las que se han llamado en esta explicación.

4.6.2 Código Python:

Al igual que sucede con el código Arduino, para el código Python se utilizarán varias librerías, entre ellas se encuentran:

- numpy: Permite realizar operaciones matemáticas y trabajar con arrays.
- pyqt5: hace que se pueda acceder a los elementos de la interfaz entre otras funcionalidades
- uic: Útil para cargar la interfaz de usuario.
- Serial: Permite la comunicación a través del puerto serie
- Time: proporciona las funciones necesarias para trabajar con fechas, horas…

```
#Importar las librerias necesarias
import sys
from PyQt5 import QtCore, uic
from PyQt5.QtWidgets import QMainWindow, QApplication, QFileDialog,QMessageBox
from pyqtgraph import QtGui, PlotWidget, plot
from PyQt5.QtCore import QPropertyAnimation,QEasingCurve
from time import time, sleep
import pyqtgraph as pg
import numpy as np
import scipy.io
import serial, time,struct,serial.tools.list_ports
from import_file import import_file
controladorlib = import_file('./controladorlib.py')
```

Figura 4.119: Código Python: Bibliotecas importadas.

Tras esto, se definen las funciones necesarias, en este caso únicamente se ha creado la función “tic” y “toc”. Dichas funciones son útiles para calcular diferencias de tiempos, como si de un cronómetro se tratara, se llama a tic cuando se pretende empezar a contar y cuando se desea pararlo, se llama a “toc”, quien devuelve la diferencia de tiempo.

Tras esto se crea la clase principal “control” (ventana principal). Dentro de esta, se definen varias funciones

Una de las más relevantes es “__init__” ya que es una función especial cuya finalidad es iniciar los atributos del objeto, es decir, es lo primero que se ejecuta cuando se crea el objeto control.

En esta función se carga la interfaz y se realizan algunos ajustes iniciales, en especial, algunos campos como legenda, rejilla …etc. relativos al gráfico de la interfaz. Esto último se

ha realizado de esta forma, ya que esos campos no están disponibles en el entorno QtDesigner.

```
#Clase CONTROL PRINCIPAL
class control(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi("GUI.ui",self)    #Cargar interfaz.ui
        #Ajustes de inicio
        self.plot_widget.setBackground('w')
        self.plot_widget.setBackground('w')
        self.plot_widget.setLabel('left', 'Centímetros')
        self.plot_widget.setLabel('right', 'Voltios')
        self.plot_widget.setLabel('bottom', 't (s)')
        self.plot_widget.setTitle(" ")
        self.plot_widget.showGrid(x = True, y = True)
        self.plot_widget.setXRange(0.0, 10.0, padding = 0)
        self.plot_widget.setYRange(0.0, 105.0, padding = 0)
```

Figura 4.120: Código Python: Clase control: Inicio.

Por otro lado, también, se ajustan algunos aspectos relativos a la pantalla, como el centrado de la ventana cuando se muestra por primera vez (llamando a la función center_Screen), o forzar la opacidad de la pantalla a su valor máximo.

```
        #otros ajustes....
        self.center_Screen()
        #eliminar barra superior
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setWindowOpacity(1)
```

Figura 4.121: Código Python: Clase control: Ajuste parámetros.

Para finalizar, una de las funcionalidades clave de "__init__" es la de aportar la funcionalidad a cada elemento de la interfaz, es decir, la acción que se va a ejecutar al pulsar, mover, o escribir sobre cada elemento. A modo de ejemplo se muestra cómo se haría con el botón cerrar la aplicación. Se accede al atributo que se desea valorar de cada elemento, en este caso al ser un botón, resulta interesante el atributo "clicked". Si ese atributo es verdadero, se conecta con la función "CLOSE_CLICKED", que se define a continuación. Observar que el nombre del botón tiene que ser exactamente el que tiene la interfaz de usuario. Ese nombre se puede consultar y cambiar en el menú derecho de Qtdesigner.

```
#BUTTON
self.button_CLOSE.clicked.connect(self.CLOSE_CLICKED)
```

Figura 4.122: Código Python: Clase control: button_CLOSE.

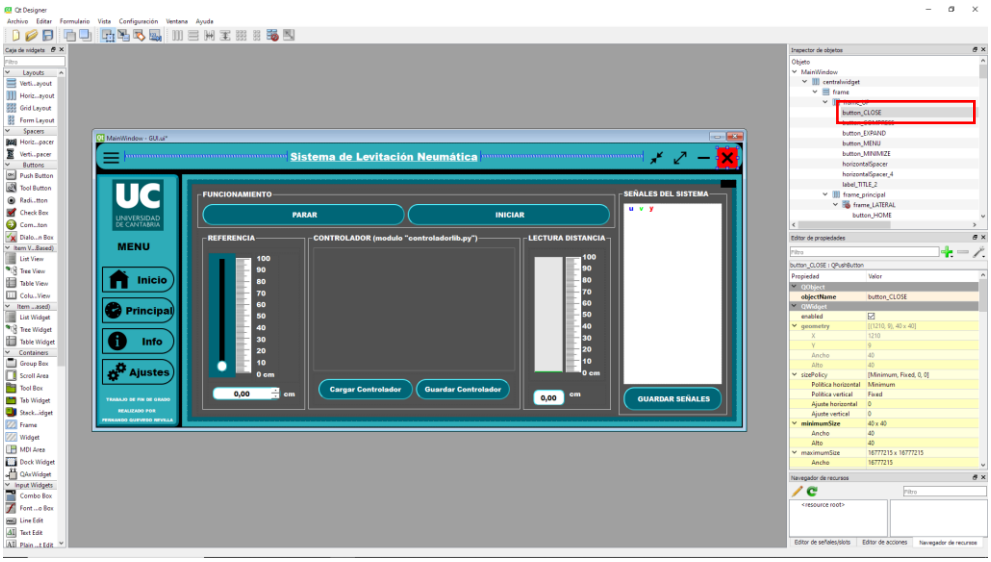


Figura 4.123: QtDesigner: button_CLOSE.

La función “CLOSE_CLICKED” únicamente pone a cero, la señal de funcionamiento y cierra la interfaz, como se muestra a continuación.

```
def CLOSE_CLICKED(self):
    self.run=0
    self.close()
```

Figura 4.124: Código Python: Función CLOSE_CLICKED.

Tras explicar cómo se procede con el botón de cerrar, el resto de los elementos se realizan de la misma manera, salvando que lógicamente cada uno está conectado con funciones distintas.

```
#FUNCIONES DE CADA ELEMENTO DE LA INTERFAZ
#SLIDER
self.slider_REFERENCIA.valueChanged.connect(self.slider_REFERENCIA_CHANGE)
#SPINBOX
self.box_REFERENCIA.valueChanged.connect(self.box_REFERENCIA_CHANGE)
# up frame movement
self.frame_UP.mouseMoveEvent=self.MOVE_SCREEN
#BUTTON
self.button_CLOSE.clicked.connect(self.CLOSE_CLICKED)
self.button_EXPAND.clicked.connect(self.EXPAND_CLICKED)
self.button_MINIMIZE.clicked.connect(self.MINIMIZE_CLICKED)
self.button_COMPRESS.clicked.connect(self.COMPRESS_CLICKED)
self.button_MENU.clicked.connect(self.MENU_CLICKED)
self.button_HOME.clicked.connect(self.HOME_CLICKED)
self.button_MAIN.clicked.connect(self.MAIN_CLICKED)
self.button_INFO.clicked.connect(self.INFO_CLICKED)
self.button_SETTINGS.clicked.connect(self.SETTINGS_CLICKED)
self.button_STOP.clicked.connect(self.STOP_CLICKED)
self.button_START.clicked.connect(self.START_CLICKED)
self.button_SAVE.clicked.connect(self.SAVE_CLICKED)
self.button_SEARCH.clicked.connect(self.SEARCH_CLICKED)
self.button_LOAD_CONTROLLER.clicked.connect(self.LOAD_CONTROLLER_CLICKED)
self.button_SAVE_CONTROLLER.clicked.connect(self.SAVE_CONTROLLER_CLICKED)
```

Figura 4.125: Código Python: Clase control: Conexión botones con las funciones.

A continuación, se explican los aspectos más relevantes de cada función:

-center_Screen: Como se vaticinaba anteriormente la finalidad de esta función es que la aplicación se muestre en el centro de la pantalla. Para lograrlo, lee la resolución de la pantalla en la que se lanza la aplicación, y se mueve la pantalla a la posición central (calculada como aparece en la figura inferior (4.126)). Si en el futuro se desea implementar un mensaje de error si la resolución de la pantalla no es la óptima, únicamente dentro de esta función se deberá introducir una condicional que compruebe que la resolución de la pantalla cumple con los requisitos mínimos. Pero, se ha decidido no añadir esta limitación para permitir a todos los alumnos independientemente de su equipo, puedan lanzar la aplicación, aunque la experiencia de usuario no sea la óptima.

```
def center_Screen (self):
    resolution = QtGui.QDesktopWidget().screenGeometry()
    self.move(int((resolution.width()-self.frameSize().width())/2),
              int((resolution.height()-self.frameSize().height())/2))
```

Figura 4.126: Código Python: Clase control: center_Screen.

-MENU_CLICKED:

Esta función maneja el menú lateral desplegable. De igual manera que se accede a los atributos de botones y sliders, también se puede acceder a las dimensiones del contenedor (o frame como se denomina en Qtdesigner) del menú lateral. De esa manera con una condicional, se varia la anchura del menú, si este estaba plegado, la anchura es nula, por lo que la nueva anchura tras presionar el botón deberá ser la que haga que el menú se despliegue. Con el fin de que esa transición sea lo más agradable visualmente, se añade al menú una animación que permite modificar el tiempo de transición, así como la curva característica del movimiento.

```
    def MENU_CLICKED(self):
        width=self.frame_LATERAL.width()
        if width==0:
            new_width=160

        else:
            new_width=0

        self.animation=QPropertyAnimation(self.frame_LATERAL,b'minimumWidth')
        self.animation.setDuration(500)
        self.animation.setStartValue(width)
        self.animation.setEndValue(new_width)
        self.animation.setEasingCurve(QtCore.QEasingCurve.InOutQuart)
        self.animation.start()
```

Figura 4.127: Clase control: MENU_CLICKED.

-Botones del menú: Estas funciones pueden ser explicadas de manera conjunta, ya que presentan una funcionalidad igual, pero cambiando el valor del atributo.

```
def HOME_CLICKED(self):
    self.stackedWidget.setCurrentWidget(self.page_HOME)
def MAIN_CLICKED(self):
    self.stackedWidget.setCurrentWidget(self.page_MAIN)
def INFO_CLICKED(self):
    self.stackedWidget.setCurrentWidget(self.page_INFO)
def SETTINGS_CLICKED(self):
    self.stackedWidget.setCurrentWidget(self.page_SETTINGS)
```

Figura 4.128: Clase control: Acceso a las páginas.

Cuando se presione cada botón se modifica la página actual del elemento “stackedWidget”, por la página deseada.

-Botones de manejo de la ventana:

Los controles de maximizar, comprimir y minimizar actúan sobre sus respectivas funciones, modificando así el tamaño de la ventana de la aplicación. Por otro lado, también se permite mover la ventana, clicando y arrastrando a la posición deseada.

```
def MINIMIZE_CLICKED(self):
     self.showMinimized()
def EXPAND_CLICKED(self):
     self.showMaximized()
     self.button_COMPRESS.show()
     self.button_EXPAND.hide()
def COMPRESS_CLICKED(self):
     self.showNormal()
     self.button_COMPRESS.hide()
     self.button_EXPAND.show()
def mousePressEvent(self, event):
    self.click_pos=event.globalPos()

def MOVE_SCREEN(self,event):

    if event.buttons()== QtCore.Qt.LeftButton:
        self.move(self.pos()+event.globalPos()-self.click_pos)
        self.click_pos=event.globalPos()
        event.accept()
```

Figura 4.129: Clase control: Botones de ajustes del tamaño de la ventana.

-SAVE_CLICKED: Esta función es la encargada de abrir una ventana emergente para guardar las señales que se han obtenido de la aplicación y de Arduino.

```
def SAVE_CLICKED(self):

    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    fileName, _ = QFileDialog.getSaveFileName(self,"Guardar señales","","Matlab Files (*.mat);;All Files (*)", options=options)
    filename = fileName.split('.')[0]
    if len(filename) > 0:
        scipy.io.savemat(filename+'.mat', dict(t = np.vstack(self.t), u = np.vstack(self.ref), v = np.vstack(self.v), y = np.vstack(self.d)))
```

Figura 4.130: Clase control: SAVE_CLICKED.

-SEARCH_CLICKED: Esta función se ejecuta cuando se presiona el botón de buscar en la pantalla de ajustes. Cada vez que se presiona limpia el valor de los desplegables o también llamados en varios programas “comboBox”. El desplegable de los puertos se rellena con los valores los puertos COM disponibles, mientras que el correspondiente a la tasa de baudios, se carga de una lista, donde por defecto aparece la opción correcta. Si se presiona el botón

de buscar y no se encuentra ningún puerto COM activo (la planta no está conectada o no es detectada) se muestra un desplegable indicando una advertencia e indicando las posibles causas.

```
def SEARCH_CLICKED(self):
    self.comboBox_BAUDRATE.clear()
    self.comboBox_COM.clear()
    ports = [port.device for port in serial.tools.list_ports.comports()]
    baudrates = ['2000000','1200', '2400', '4800', '9600', '19200', '38400', '115200']
    self.comboBox_COM.addItems(ports)
    self.comboBox_BAUDRATE.addItems(baudrates)
    if ports==[]:
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("No se encontró ningún dispositivo USB")
        msg.setInformativeText("Revise las conexiones PC-Planta")
        msg.setWindowTitle("Aviso")
        msg.setDetailedText("Posibles causas:\n"
                            "1.No la planta no está conectada al equipo\n"
                            "2.El cable no esta conectado al sistema\n"
                            "3.No se reconoce al sistema\n")
        msg.setStandardButtons(QMessageBox.Cancel)
        retval = msg.exec_()
```

Figura 4.131: Clase control: SEARCH_CLICKED.

-Cargar y guardar controlador:

Estas funciones son las encargadas de abrir o guardar el archivo Python del controlador. El funcionamiento es similar al de la función “SAVE_CLICKED”, a excepción de que en estas se accede al elemento editor de textos para escribir o leer respectivamente, el código escrito en él.

```
def LOAD_CONTROLLER(self, filename):
    self.filecontrolador = open(filename,'r')
    self.textEdit_CONTROLLER.setPlainText(self.filecontrolador.read())
    self.filecontrolador.close()
    self.groupBox_CONTROLLER.setTitle('CONTROLADOR ' + filename)

def SAVE_CONTROLLER(self, filename):
    self.filecontrolador = open(filename,'w')
    self.filecontrolador.write(self.textEdit_CONTROLLER.toPlainText())
    self.filecontrolador.close()
    self.groupBox_CONTROLLER.setTitle('CONTROLADOR ' + filename)

def LOAD_CONTROLLER_CLICKED(self):
    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    fileName, _ = QFileDialog.getOpenFileName(self,"Cargar controlador","","Python Files (*.py);;All Files (*)", options=options)
    filename = fileName.split('/')[-1]
    self.LOAD_CONTROLLER(filename)

def SAVE_CONTROLLER_CLICKED(self):
    options = QFileDialog.Options()
    options |= QFileDialog.DontUseNativeDialog
    fileName, _ = QFileDialog.getSaveFileName(self,"Guardar controlador","","Python Files (*.py);;All Files (*)", options=options)
    filename = fileName.split('/')[-1]
    self.SAVE_CONTROLLER(filename)
```

Figura 4.132: Clase control: LOAD_CONTROLLER Y SAVE_CONTROLLER.

- Slider y caja de texto de referencia: Estas dos funciones están encaminadas a que cuando se modifique la referencia bien a través del cuadro de texto, o bien a través del slider, uno tome el valor del otro. Explicado de otra manera si en el cuadro de texto se escribe un valor, se debe actualizar la nueva posición del slider acorde a esa nueva posición, o a la inversa. Pero para que esto no se convierta en un bucle infinito de actualización. Cada vez que se actualiza el valor de uno, se debe bloquear temporalmente la señal de ese.

```
def slider_REFERENCIA_CHANGE(self):
    self.box_REFERENCIA.blockSignals(True)
    self.box_REFERENCIA.setValue(self.slider_REFERENCIA.value())
    self.box_REFERENCIA.blockSignals(False)

def box_REFERENCIA_CHANGE(self):
    self.slider_REFERENCIA.blockSignals(True)
    self.slider_REFERENCIA.setValue(self.box_REFERENCIA.value())
    self.slider_REFERENCIA.blockSignals(False)
```

Figura 4.133: Clase control: Sincronización deslizadera y cuadro de texto slider_REFERENCIA_CHANGE y box_REFERENCIA_CHANGE.

-START_CLICKED: Es la función, que se encarga de realizar el control, y la comunicación, por lo que es una de las más importantes de todo el código, y ese es uno de los motivos por la que es la más larga. Con el fin de poder ilustrar convenientemente las explicaciones se muestra el flujograma simplificado correspondiente a esta función.

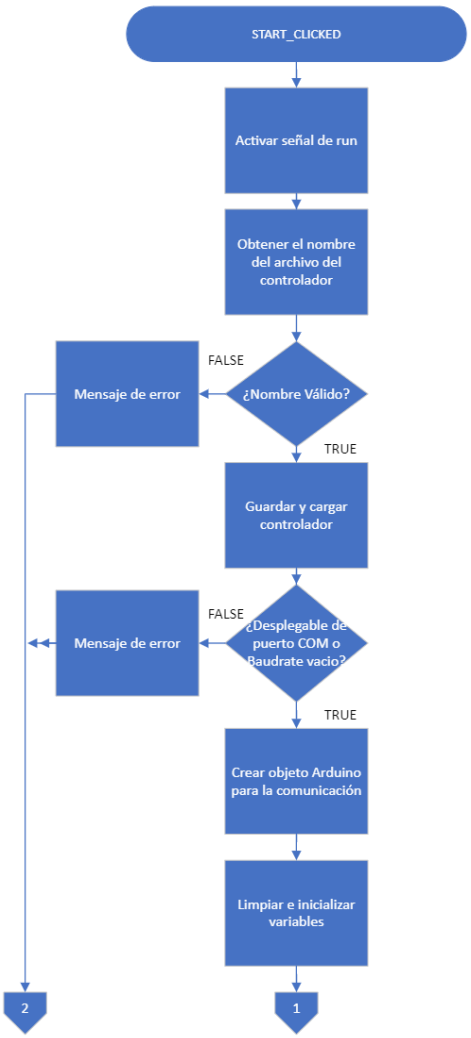


Figura 4.134: Diagrama de flujo de la función START_CLICKED (I).

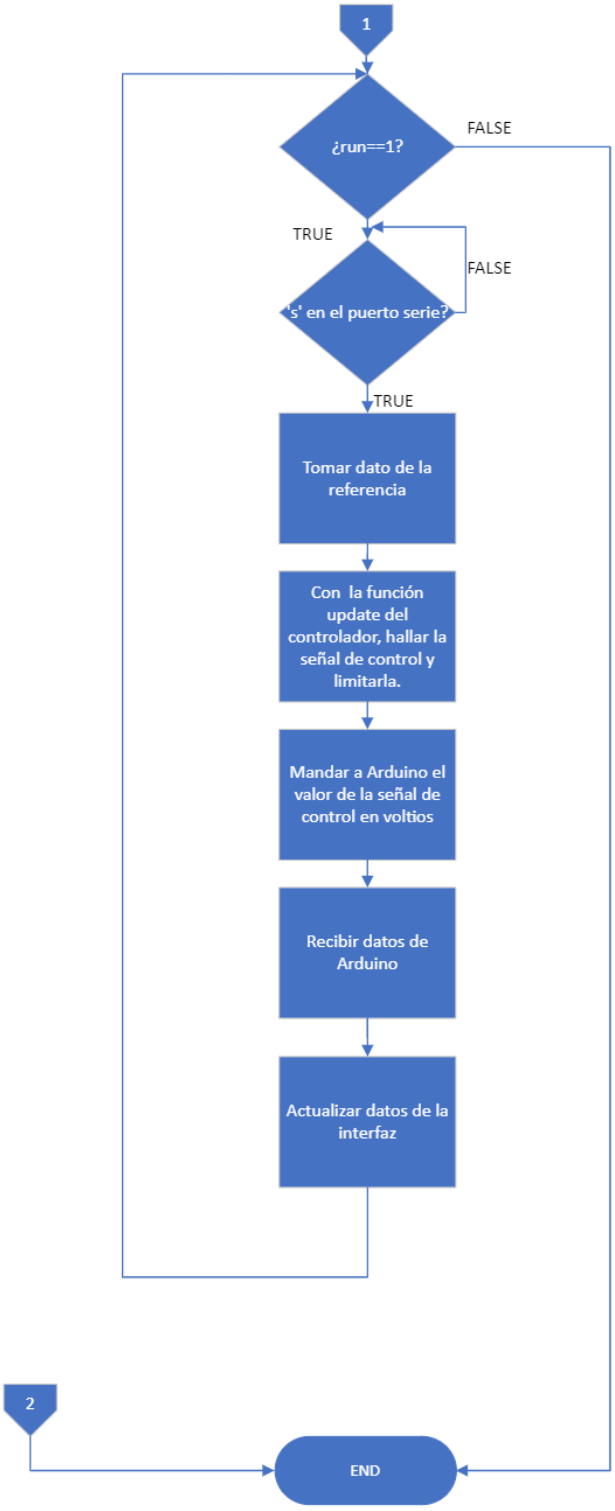


Figura 4.135 Diagrama de flujo de la función START_CLICKED(II)

Como se refleja en el flujograma anterior, el primer paso a llevar a cabo es poner a 1 el valor de la variable “run” y comprobar si se ha cargado previamente un controlador. La forma en la que se ha hecho eso es comprobando el nombre del groupBox_CONTROLLER, ya que cuando se carga un controlador, se actualiza en su atributo de título, el nombre del archivo del controlador, precedido por el string “CONTROLADOR”, luego si este título tiene el valor por defecto, implica que no se ha guardado ni cargado ningún archivo, por lo que no resultaría posible ejecutar la labor de control. Por ese, motivo se muestra un mensaje de error emergente indicando este fallo.

```
def START_CLICKED(self):
    #print('Start clicked') #For debug
    self.run=1
    filename = self.groupBox_CONTROLLER.title().split()[-1]
    #print(filename)#For debug
    if filename=='"SELECCIONE_O_CREE_UN_ARCHIVO!")':
        msg = QMessageBox()
        msg.setIcon(QMessageBox.Warning)
        msg.setText("No se encontró ningún controlador")
        msg.setInformativeText("Revise la ubicación del controlador")
        msg.setWindowTitle("Aviso")
        msg.setDetailedText("Posibles causas:\n"
                            "1.Problemas con la extensión .py\n"
                            "2.El usuario no tiene premisos\n"
                            "3.El nombre no se corresponde\n")
        msg.setStandardButtons(QMessageBox.Cancel)
        retval = msg.exec_()
```

Figura 4.136: Clase control: START_CLICKED (I).

Si el resultado de esa comprobación es favorable, se guarda, carga e importa el archivo del controlador.

```
    self.SAVE_CONTROLLER(filename)
    self.LOAD_CONTROLLER(filename)
    controladorlib = import_file(filename)
    control = controladorlib.controlador()
```

Figura 4.137: Clase control: START_CLICKED (II): Guardar y cargar controlador.

Lo que sigue es comprobar si se ha seleccionado algún puerto COM o tasa de baudios para establecer la comunicación, para ello se comprueba si los desplegables de sendos campos están vacíos. Si están vacíos se muestra un mensaje indicando que no se puede iniciar el control debido a que no se han especificado dichos campos, así como a donde debe acudir para configurar esos campos.

```
#Comenzar comunicacion Arduino
if self.comboBox_COM.currentText()=="" or self.comboBox_BAUDRATE.currentText()=="":
    QMessageBox.critical(self, "Error", "No se pudo conectar con la planta\nRevise los ajustes de comunicación\n"
                         "Acuda a la página de ajustes")
    return
```

Figura 4.138: Clase control: START_CLICKED (III): Error de comunicación.

Si, por el contrario, sí que existen esos campos, se crea el objeto “arduino” para la comunicación haciendo uso de la biblioteca “Serial”.

Posteriormente se limpia el gráfico y se define el formato (tono y grosor) de cada señal y se inicializan a cero todos los arrays de datos que se utilizarán posteriormente. Para esto último se utiliza la biblioteca “numpy”. Además, se inicia el valor de índice de muestras “n” a valor 1.

```
# Formato gráfica plot
self.plot_widget.clear()
ref_pen = pg.mkPen(color = (0, 0, 255), width = 1.25)
v_pen = pg.mkPen(color = (0, 255, 0), width = 1.25)
d_pen = pg.mkPen(color = (255, 0, 0), width = 1.25)
ref_trace = self.plot_widget.plot([], [], name = "u", pen = ref_pen)
v_trace = self.plot_widget.plot([], [], name = "v", pen = v_pen)
d_trace = self.plot_widget.plot([], [], name = "y", pen = d_pen)
t_init=tic()
packages_sent=0
packages_received=0
#Crear vectores
n=1
self.t = np.linspace(0.0, 100, 100000, dtype = 'float')
self.ref = np.zeros_like(self.t)
self.v= np.zeros_like(self.t)
self.d = np.zeros_like(self.t)
```

Figura 4.139: Clase control: START_CLICKED (IV): Inicializar variables y formato del gráfico.

Una vez ejecutado este código de preparación de las señales y objetos, comienza un bucle “while” mientras la señal de run tome el valor de la unidad (recordar que ese valor se fuerza a 1 al entrar a la función START_CLICKED y se fuerza a cero cuando se presiona el botón STOP). El código que se ejecuta dentro de este bucle está formado por dos comandos “try-except”. Este comando lo que hace es comprobar si se puede ejecutar correctamente (sin errores) el código que está dentro de “try”, si esto es así, se ejecuta, pero si contiene errores en vez de parar el programa, lo que hace es pasar a la sección de código que contiene “except”. En el código que se presenta, no hay código en “except” (en realidad “pass” es un comando “null” o vacío), por lo que, si se detecta algún error, únicamente no se realizará ninguna acción y se seguirá ejecutando el bucle.

El primer “try-except”, se corresponde con el envío de datos del código Python a Arduino. Tras haber calculado y limitado la señal de control en voltios, haciendo uso de “update” del archivo Python del controlador, se envía a través del puerto serie la referencia en centímetros y la señal de control en voltios (una vez recibida la señal de control por Arduino la transforma a ciclo de trabajo). Es importante enviar los datos como variables tipo float de 32 bits, ya que Arduino no trabaja con 64 bits. Además de enviar esos datos a Arduino, la aplicación también los guarda en un array para después realizar la representación y poder exportar las señales.

```
#ENVIO PYTHON -> ARDUINO
try:
    while arduino.read()!=b's':
        pass
    #Lectura de parámetros, interfaz
    ref_= np.float32(self.box_REFERENCIA.value())
    self.ref[n]=ref_
    control.update(self.ts, self.ref[n]/100*12,self.d[n]/100*12)
    self.v[n]=control.v
    if control.v>=12:
        self.v[n]=np.float32(12)
    elif control.v<=0:
        self.v[n]=np.float32(0)
    else :
        self.v[n]=np.float32(control.v)

    arduino.write(np.float32(ref_))
    #print('send ref=',ref_)
    arduino.write(np.float32(self.v[n]))
    #print('send v=',self.v[n])
    packages_sent +=1


except:
    pass
```

Figura 4.140: Clase control: START_CLICKED (V): Comunicación de Python a Arduino.

El otro “try-except”, de dentro del bucle es relativo a la comunicación desde Arduino dirección Python. En este caso se reciben desde Python 4 datos del tipo float de 32 bits, por lo que se deben leer 128 bits o 16 bytes como se indica en el código. Esos cuatro, datos son en periodo de muestreo real de Arduino, la señal de control que ha sido enviada previamente, para finalizar un campo auxiliar (los dos últimos a efectos de facilitar el depurado del código).

```
        #RECEPCION ARDUINO -> PYTHON
        try:
            n=n+1
            self.ts,self.v[n],self.d[n],other = struct.unpack('ffff', arduino.read(16))
            packages_received +=1
            #print('d = ', self.d[n])
            #Actualizar pantalla
            self.progressBar.setValue(self.d[n])
            self.box_DISTANCE.setValue(self.d[n])
            # Calculo el tiempo
            self.t[n]=self.t[n-1]+self.ts;
            #Actualiza grafica
            ref_trace.setData(self.t[:n], self.ref[:n])
            v_trace.setData(self.t[:n], self.v[:n])
            d_trace.setData(self.t[:n], self.d[:n])
            self.plot_widget.update(True)

        except:
            pass
        QApplication.processEvents()
    arduino.close()
```

Figura 4.141: Clase control: START_CLICKED (VI): Comunicación de Arduino a Python.

El bucle “while” finaliza procesando los eventos de la interfaz.

Cuando se sale del bucle se cierra o interrumpe la comunicación con Arduino.

5 RESULTADOS

En este capítulo se mostrarán los resultados finales obtenidos tras las fases de diseño y construcción. Comentar que no se mostrarán los apartados correspondientes a las interfaces de usuario de pantalla y de la aplicación, ya que ambas ya han sido mostradas en su capítulo dedicado. Lo primero que se mostrará es el resultado tras haber impreso cada pieza, además de su proceso de ensamblaje.

5.1 ESTRUCTURA Y ENSAMBLAJE

El elemento central sobre los que se acoplan todos, es la base que se muestra a continuación. Esta contiene los orificios en la posición exacta para que sea capaz de aguantar la placa Arduino, la PCB, el transformador, ventilador, así como las tapas laterales y trasera.

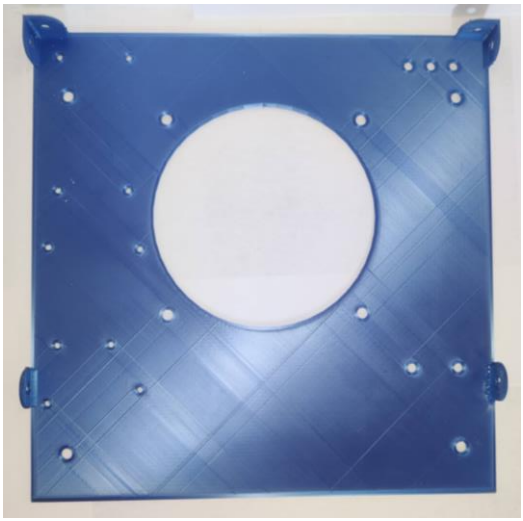


Figura 5.1: Resultado base principal.

Sobre la base se monta el bastidor de aluminio que refuerza la misma y con la misma tornillería se fijan las patas de la planta.

Figura 5.2: Resultado patas.

Figura 5.3: Bastidor aluminio.

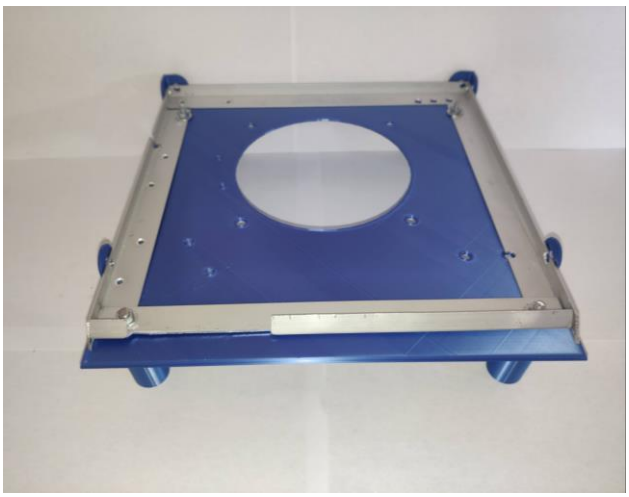


Figura 5.4: Ensamblaje base principal-patas-bastidor.

Destacar que el bastidor de aluminio anodizado está mecanizado tomando en cuenta la disposición y dimensiones de los elementos que soportará la base, esto implica la existencia de taladros allá donde se requieran o las dobleces necesarias para no entorpecer el paso del cableado, e incluso el avellanado de los taladros para embutir los tornillos donde sea necesario.

El siguiente paso del ensamblaje consiste en atornillar las tapas laterales, previo paso del cable USB. Además, también resulta un buen momento para colocar toda la tornillería (con sus correspondientes tuercas y arandelas), que servirán para fijar los elementos, el resultado de esta etapa es el que se muestra en la siguiente imagen.

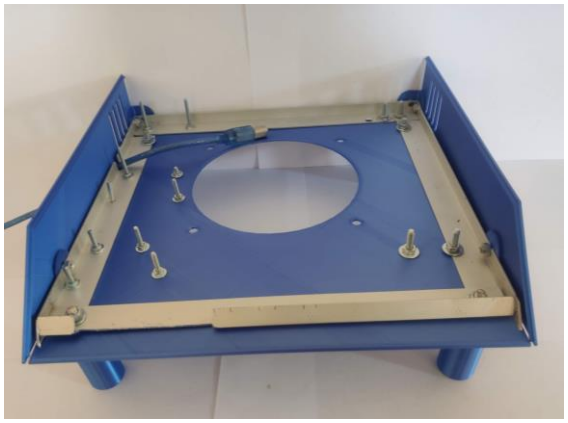


Figura 5.5: Ensamblaje base principal-patas-bastidor y tapas laterales.

Tras esto, se colocan la placa de Arduino, la PCB y el transformador, con su correspondiente cableado de alimentación. Observar que la posición de la PCB está sobre elevada respecto a la de Arduino, para permitir la conexión del cable USB del Arduino, y el conector Jack de alimentación. Esto estaba diseñado desde un principio así para realizar el equipo más compacto, la razón por la que no se escogió una distribución aún más compacta es que no se conocía exactamente cuál iba a ser el calentamiento del Arduino y de la PCB, lo cual, si estuvieran uno encima del otro, podría radicar en un problema de sobrecalentamiento de algún elemento.

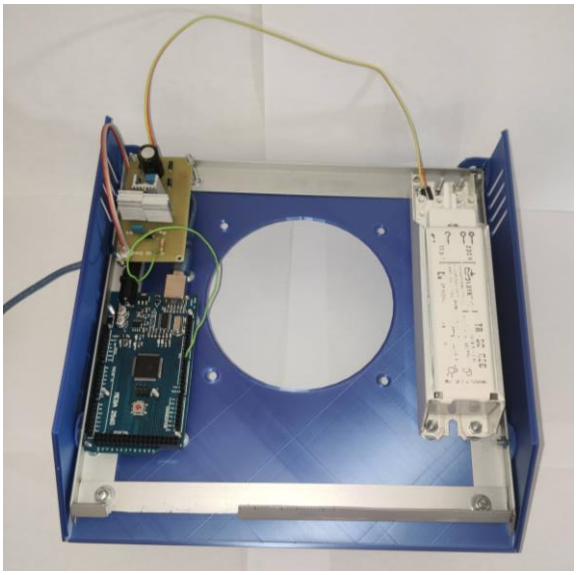


Figura 5.6: Montaje de los componentes eléctricos y electrónicos.

Tras ello se coloca el refuerzo de la consola sobre la misma, y se montan los elementos abrazando la consola de PLA con el refuerzo de aluminio.

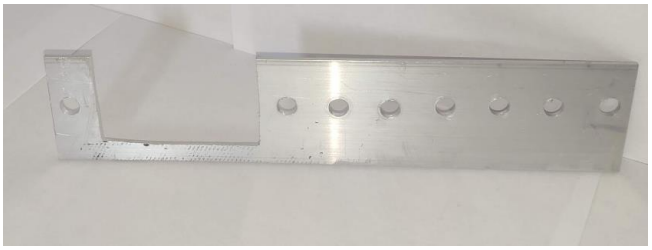


Figura 5.7: Refuerzo trasero consola.

Figura 5.8: Vista trasera de la consola con los periféricos instalados.

El siguiente paso es montar los elementos más voluminosos de dentro del alojamiento, como son el ventilador y el embudo. El resultado al juntar la consola con la base es el que se muestra en la siguiente imagen.

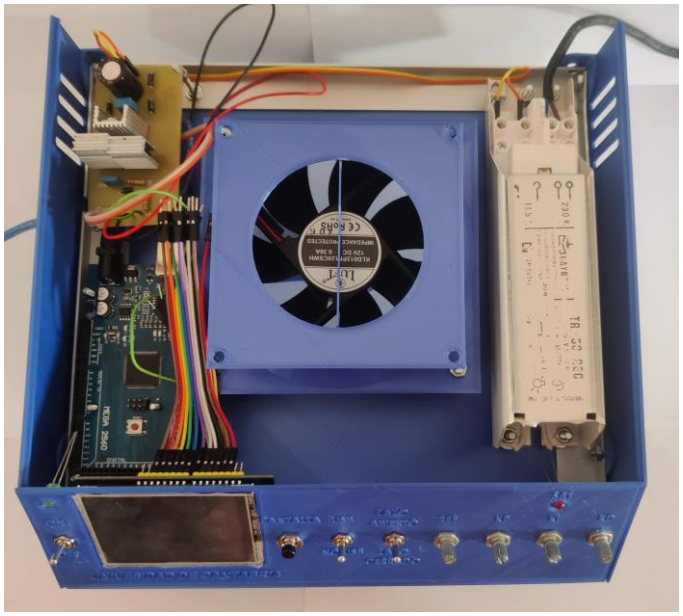


Figura 5.9: Resultado tras incluir el ventilador con sus adaptadores y la consola.

Lo que sigue es quizás uno de los trabajos más tediosos, y es el encajar la lámina de metacrilato en ambos soportes (inferior y superior), quienes además deben ir acoplados a una barra de aluminio de sección cuadrada. A continuación, se muestra una vista en detalle donde se aprecia el ajustado espacio donde encaja la lámina

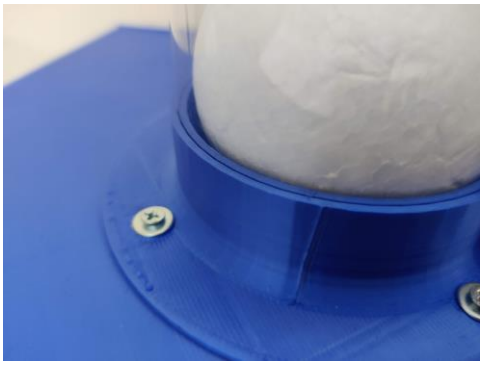


Figura 5.10: Vista en detalle del ajuste de la lámina en el soporte.

El montaje de la estructura termina al colocar la tapa superior, lateral, así como fijar el soporte del sensor, con el mismo. Además, se fija una escala numérica en la que puede leer la altura de la esfera en centímetros

Figura 5.11: Instalación del sensor.

Figura 5.12 Resultado de la impresión de la tapa trasera.

El resultado final de la estructura es la que se muestra a continuación.

Figura 5.13: Vista general de la planta ensamblada.

Aunque el montaje ha sido narrado centrándose en las partes mecánicas, la secuencia de montaje real ha sido incluyendo de por medio labores de cableado, soldadura, además de comprobaciones del correcto funcionamiento de cada elemento.

Los puntos de alimentación para cada elemento han sido obtenidos mediante la soldadura de cableado entre, sí. Destacar que cada soldadura que contiene la planta ha sido protegida mediante funda termo retráctil para evitar problemas futuros y de esta manera aumentar la fiabilidad y seguridad del equipo. E incluso en las soldaduras de varios hilos, como por ejemplo los que suben hasta la posición del sensor, además de estar recubierto cada hilo, se ha incluido una funda termo retráctil que aísla el conjunto

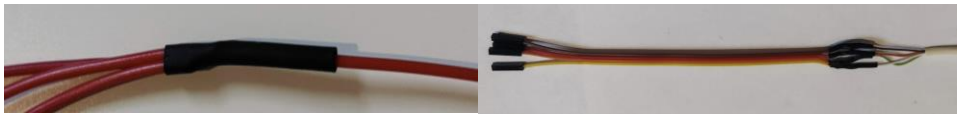


Figura 5.14: Protección de las soldaduras con funda termo retráctil.

En la siguiente imagen se muestra el cableado antes de instalar algunas tapas.

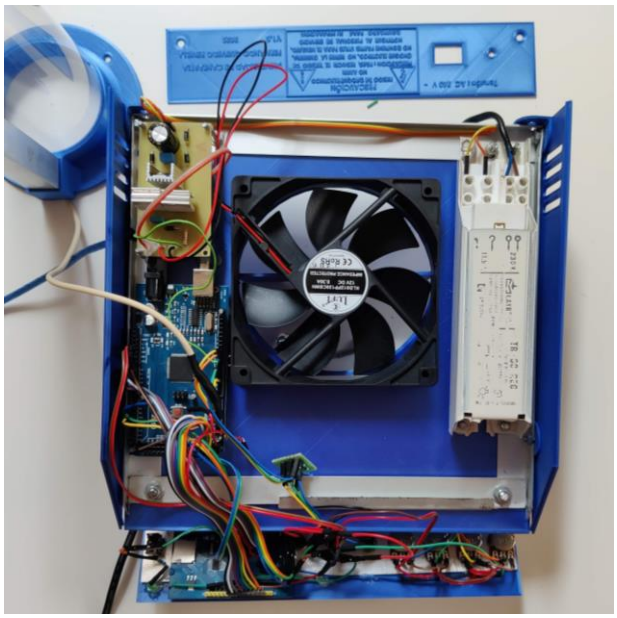


Figura 5.15: Cableado interno de todos los componentes.

Como se puede apreciar, a pesar de que se ha procurado seguir un código de colores y distribuir el cableado de forma eficiente y lógica, resultaría difícil para un tercero realizar el mantenimiento o reparación del equipo sin la documentación que se muestra en este documento, de ahí la importancia de que, a la hora de realizar un proyecto de tales características, al final todo quede convenientemente documentado.

5.2 ELECTRÓNICA

Una vez que se ha realizado el montaje de todos los elementos, no está de más volver a comprobar que toda la electrónica no ha sufrido ningún deterioro y que esta funciona bien. Para ello se capturarán las señales más importantes que maneja el equipo.

La primera comprobación que se realiza es la tensión a la salida del transformador, obteniéndose una forma de onda como la mostrada en la siguiente captura.

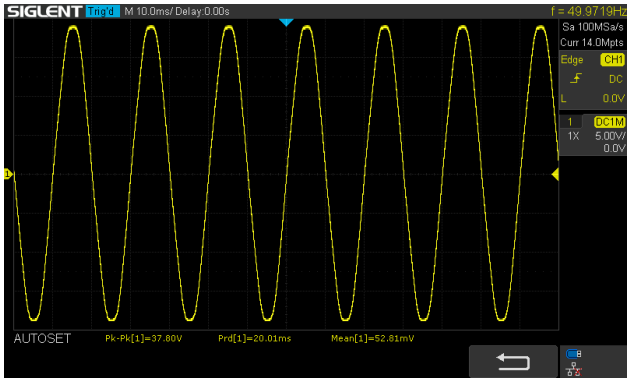


Figura 5.16: Captura osciloscopio: Tensión a la salida del transformador. En carga.

Tras esta comprobación, se analiza la siguiente etapa de la fuente de tensión, la correspondiente a la salida del puente de diodos y del filtro.

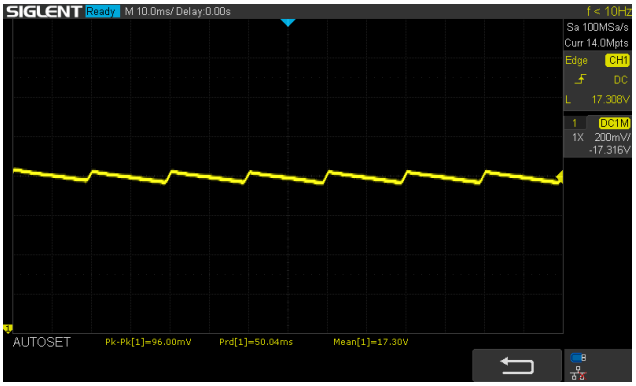


Figura 5.17: Captura osciloscopio: Tensión a la salida del puente de diodos y condensador. En carga.

Como se puede apreciar en las medias efectuadas el valor medio de la señal es de 17.30 V, mientras que solo tiene una medida pico a pico muy pequeña, de 96 mV. Después se mide la tensión a la salida de cada regulador de tensión, obteniéndose los siguientes resultados.

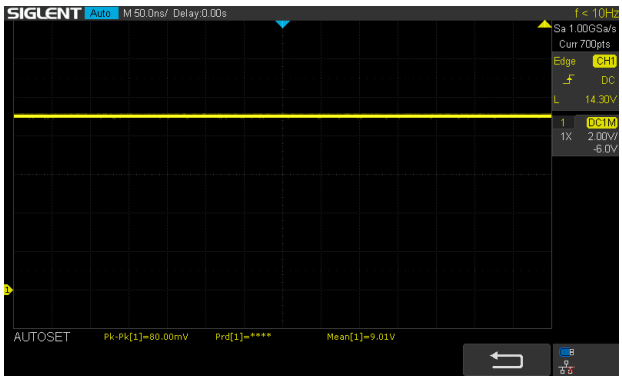


Figura 5.18: Tensión a la salida del regulador de tensión de 9 V.

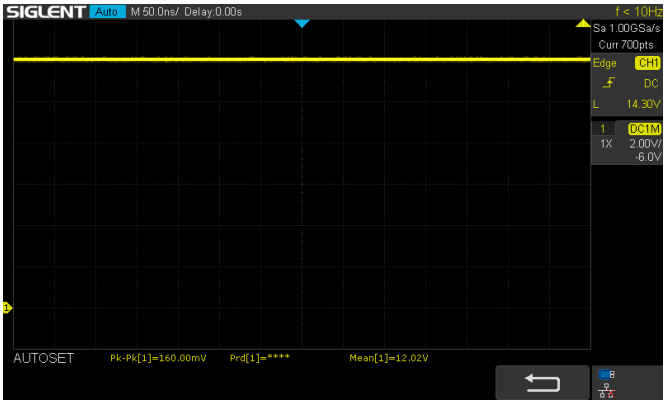
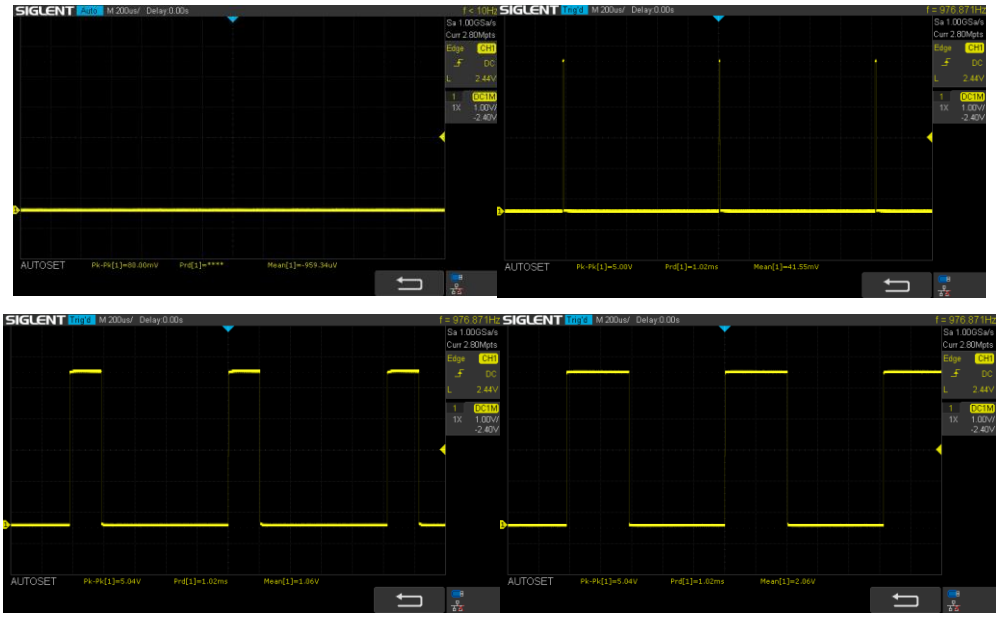


Figura 5.19: Tensión a la salida del regulador de tensión de 12 V.

Antes de comprobar si el transistor funciona correctamente, se verifica que la señal que le llega de Arduino es la que debería ser, para ello se fuerza el valor del ciclo de trabajo. Para esta prueba se han escogido varios ciclos de trabajo variados. En concreto las figuras que se muestra a continuación se corresponden con ciclos de trabajo de 0;1,50;100;150;200;250;254;255, numerado de arriba debajo de izquierda a derecha. Notar que una forma de comprobar es que el osciloscopio tiene la función de calcular el valor medio que como se explica en el capítulo “4.2 ELECTRÓNICA DE POTENCIA” se deduce el ciclo de trabajo es proporcional a la tensión media medida.



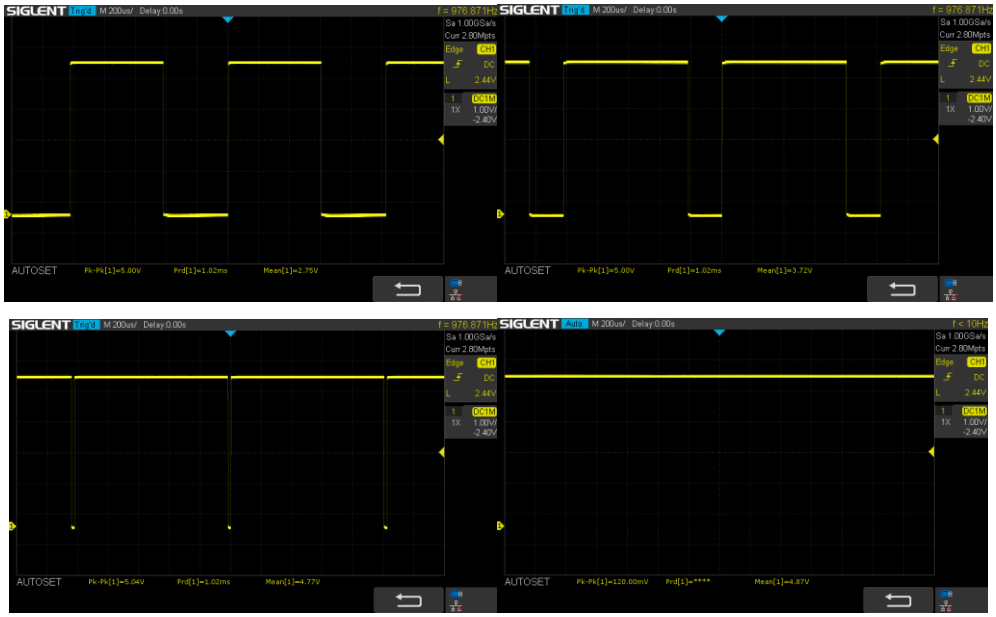


Figura 5.20: Captura osciloscopio: Variación del ciclo de trabajo de la señal PWM generada por Arduino, medida en la puerta del transistor.

Tras la comprobación de que Arduino genera la señal PWM correctamente, se fija un ciclo del 70% y se captura la señal de tensión en los extremos del ventilador, el resultado se muestra a continuación.

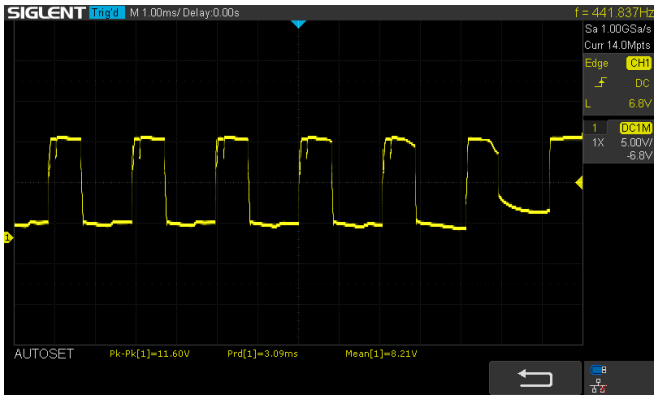


Figura 5.21: Captura osciloscopio: Tensión en bornes del ventilador (I).

El valor de la tensión media teórico debería ser el 70% de 12 V, es decir unos 8.4 V, en cambio se obtiene que la lectura pico a pico de la señal es de 11.8 V, en vez de los 12 V, teóricos, lo cual hace que la tensión media del ventilador se desvíe unos 0.2 V menos de la tensión media esperada.

Para comprobar si esta desviación se agrava al aumentar el ciclo de trabajo se repite el ensayo anterior, pero esta vez con un ciclo de trabajo del 80% (aleatorio, pero superior que el anterior).

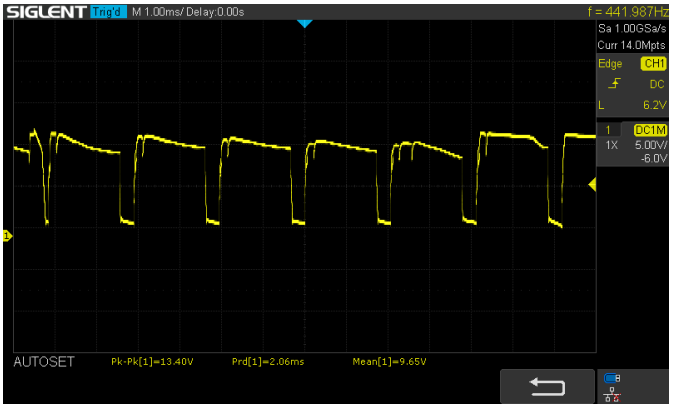


Figura 5.22: Captura osciloscopio: Tensión en bornes del ventilador (II).

En este caso, la lectura pico a pico es superior a la esperada, llegando en el momento de la captura a 13.40 V, lo cual hace que el valor medio de la señal sea superior al esperado. Cabe comentar que estos datos son de un cierto instante, lo que implica que estos valores tanto el flujo de aire en cada momento, como la altura y tendencia de la bola, como de las propias inercias del ventilador pueden hacer variar la señal (ya que no es lo mismo que el ventilador se encuentre acelerando que frenando). Una vez comprobado que esta parte de la electrónica funciona correctamente se comprueba con un polímetro que la tensión de los potenciómetros varia correctamente al girarlos.

5.3 SOFTWARE

Tras esto se realizan pruebas finales, tanto trabajando de forma aislada, como trabajando en sincronía con el ordenador. Estas pruebas tienen con finalidad ajustar el offset del sensor, o ajustar la zona muerta de los potenciómetros, todo ello realizado vía software. Para probar el correcto funcionamiento de la aplicación se carga, tal y como lo haría un alumno, el PID en la ventana de código de la aplicación, en el caso que se muestra con unas ganancias obtenidas mediante sintonización manual. Tras ello se inicia la comunicación y el control, y se observa la respuesta de la planta ante un escalón. También se testean el resto de las funcionalidades como la de exportar las señales.

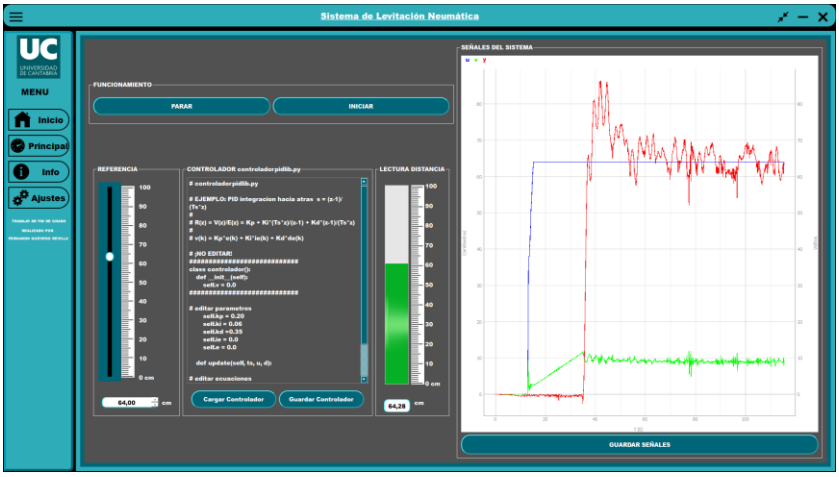


Figura 5.23: Funcionamiento de la aplicación en conjunto con la planta.

5.4 IDENTIFICACIÓN

5.4.1 Procesado de las señales

Para realizar la identificación de la planta se coloca el sistema en lazo abierto, y a través de la interfaz de usuario que se explica en el documento, se obtienen las señales de tiempo, entrada, señal de control y la salida del sistema. Al representar el tiempo se nota que debido a los tiempos de inicio y animación del sistema, hacen que la captura de las señales comience aproximadamente a los 7 segundos. También se observa un ligero desvío en el periodo de muestreo deseado (0.1 s) en las primeras muestras. A continuación, se muestra el código con su correspondiente resultado de lo que se comenta en este párrafo.

```
%Cargar datos exportados de la aplicación
load lazo_abierto.mat;
% Calcular el periodo de muestreo como tiempo entre muestras consecutivas
ts = diff(t);
ts = [ts; ts(end)];
% Representar el periodo de muestro y el tiempo gráficamente
figure('color', 'white');
subplot(2,1,1);
plot(t, 'linewidth', 2);
legend('t');
xlabel('k');
title('Tiempo t');
subplot(2,1,2);
plot(ts, 'linewidth', 2);
legend('ts')
xlabel('k');
title('Periodo de muestreo ts');
```

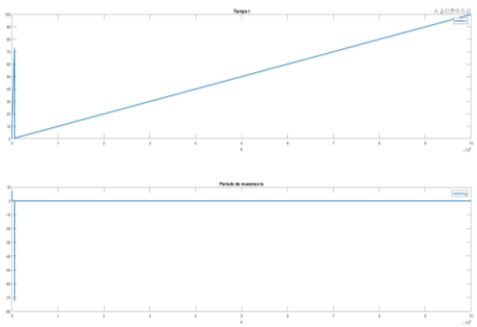


Figura 5.24: Código y resultado del vector de tiempos y del periodo de muestreo.

Estas anomalías del vector de tiempos son solucionadas desestimando las muestras cuyo periodo de muestreo exceda 0.2 segundos o sea inferior al deseado. La manera de realizar esto se muestra en la siguiente figura (5.25).

```
% Quedarse únicamente con las muestras cuyo periodo de
% muestreo sea consistente
n = find((ts > 0.1) & (ts < 0.2));
t = t(n);
u = u(n);
v = v(n);
y = y(n);
ts = ts(n);

% Quitar el offset de tiempo del vector de tiempos
t = t-t(1);
```

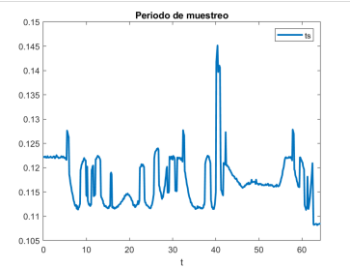


Figura 5.25: Código MATLAB: Eliminar muestras inconsistentes y representación del periodo de muestreo.

Una vez corregido el conjunto de las señales cuyo periodo de muestreo no es correcto, se representan el conjunto de las señales para su posterior análisis.

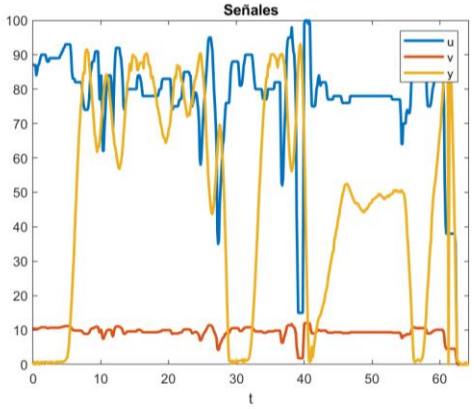


Figura 5.26: Señales originales, eliminando las muestras inconsistentes.

Las señales que se corresponden con distancias están representadas en centímetros (entrada y la salida), en cambio la señal de control está en voltios. Por comodidad, se decide normalizan las señales (dividir cada señal por su fondo de escala).

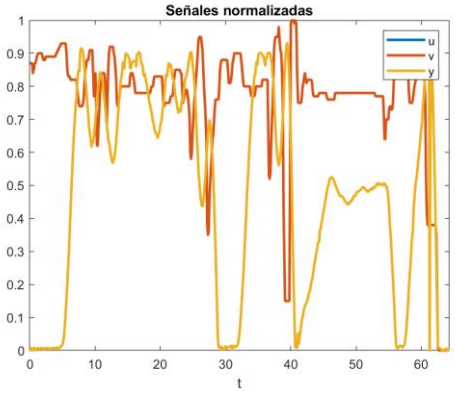


Figura 5.27: Señales normalizadas.

Una vez que se observan las señales normalizadas, se obtiene que la sesión de obtención de datos se realizó como previamente se conocía en lazo abierto y con una ganancia unidad, ya que la señal de control es igual que la entrada. Como se observa en la figura 5.25, el periodo de muestreo no es exactamente constante, si no que este fluctúa ligeramente. Para solventar esto, lo que se hace es calcular un periodo de muestreo medio y después crear un vector de tiempos a partir de ese periodo de muestreo medio. Pero esto a su vez requiere un ajuste sobre las señales, ya que estas no han sido tomadas exactamente cada periodo de muestreo medio calculado, por este motivo, se interpolan las mismas para obtener el valor las señales cada ese periodo de muestreo medio.

```
% ts no es constante,
% se utilizará la media del periodo de muestreo
ts = mean(ts)

% Nuevo vector de tiempos, con ts constante
tq = [0:length(t)-1]'*ts;

% Interpolar las señales para calcular su valor cada ts
u = interp1(t,u,tq,'linear');
v = interp1(t,v,tq,'linear');
y = interp1(t,y,tq,'linear');
t = tq;

% Representar las señales interpoladas
figure('color', 'white');
plot(t, [u, v, y], 'linewidth', 2);
legend('u', 'v', 'y');
xlim([t(1),t(end)]);
xlabel('t');
title('Señales intepoladas');
```

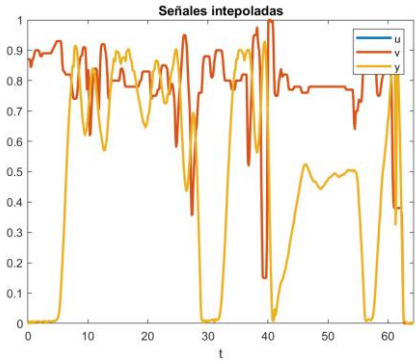


Figura 5.28: Vector de tiempos constante e interpolación de señales.

A simple vista no existe diferencia entre las señales interpoladas y las señales sin interpolar. El último paso antes de proceder a la identificación del sistema en sí misma, es quitar las medias, o dicho de otro modo referir cada señal respecto a su punto de trabajo medio.

```
% Punto de trabajo medio de cada señal
v1m = mean(v)
y1m = mean(y)

% Desviación respecto el punto de trabajo
v = v-v1m;
y = y-y1m;

% Representar las desviaciones
figure('color', 'white');
plot(t, [v, y], 'linewidth', 2);
legend('v', 'y');
xlim([t(1),t(end)]);
xlabel('t');
title('Desviaciones en torno al punto de trabajo');
```

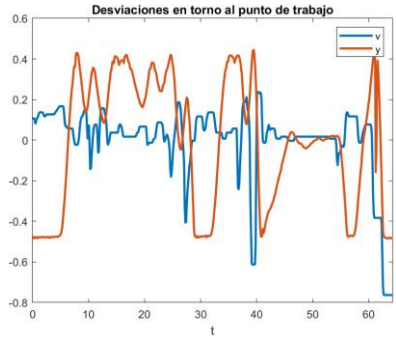


Figura 5.29: Desviación respecto al punto de trabajo de la señal de control y la salida.

Como se esperaba, la señal tiene la misma forma, pero eliminando el valor medio de cada señal (ver figura 5.29).

5.4.2 Identificación lineal

Tras varias pruebas, con distintos modelos, con el que se obtuvieron mejores resultados fue con un modelo de décimo orden, es decir, la ecuación en diferencias de la que se desean obtener sus parámetros tiene la siguiente forma:

$$y_k = b_9v_{k-1}+b_8v_{k-2} + b_7v_{k-3} + b_6v_{k-4} + b_5v_{k-5}+b_4v_{k-6} + b_3v_{k-7} + b_2v_{k-8} + b_1v_{k-9} + b_0v_{k-10} + b_{10}v_{k-10} - a_9y_{k-1} - a_8y_{k-2} - a_7y_{k-3} - a_6y_{k-4} - a_5y_{k-5} - a_4y_{k-6} - a_3y_{k-7} - a_2y_{k-8} - a_1y_{k-9} - a_0y_{k-10}$$

(5.1)

Donde se conoce todo para cada valor de la muestra k a excepción de los parámetros a y b. También se puede escribir de la siguiente forma:

$$y_k = m_k \cdot \theta \qquad \textit{(5.2)}$$

Donde:

y_k: Salida del sistema en cada muestra (conocido)

m_k: Vector regresor de la forma $(v_{k-1}, v_{k-2}, \dots, v_{k-10}, y_{k-1}, y_{k-2}, y_{k-10})$ (conocido)

θ: Vector de parámetros de la forma $(b_9, b_8, \dots, b_0, a_9, a_8, \dots, a_0)^T$ (desconocido)

Por lo que, si se juntan todas las ecuaciones para cada muestra, se obtiene un sistema de matrices del que se puede despejar el vector deseado de parámetros θ.

$$y = M\,\theta \qquad \textit{(5.3)}$$

$$\begin{pmatrix} y_{10} \\ y_9 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} v_{10} & v_9 & \dots & v_0 & -y_{10} & -y_9 & \dots & -y_0 \\ v_{11} & v_{10} & \dots & v_1 & -y_{11} & -y_{10} & \dots & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ v_{n-1} & v_{n-2} & \dots & v_{n-10} & -y_{n-1} & -y_{n-2} & \dots & -y_{n-10} \end{pmatrix} \begin{pmatrix} b_9 \\ b_8 \\ \vdots \\ b_0 \\ a_9 \\ a_8 \\ \vdots \\ a_0 \end{pmatrix} \qquad \textit{(5.4)}$$

Donde:

n: Número de muestras.

Despejando el vector de parámetros de la ecuación matricial, resulta:

$$\theta = M^+ y \qquad \textit{(5.5)}$$

Siendo M^+ la matriz pseudo inversa.

La forma de operar esto es utilizando MATLAB, ya que el orden y el número de muestras haría inviable el realizarlo manualmente.

```
%% IDENTIFICACION LINEAL
o = 10;   % Orden del modelo
k = o+1;  % Orden +1
b = y(k:end); % Salida desde muestra 11 al final
m = []; %iniciar la matriz
for n = 1:o
    m(:,end+1) = -y(k-n:end-n);
end
for n = 1:o
    m(:,end+1) = v(k-n:end-n);
end
x = m\b %% Obtención del vector de parámetros
```

Figura 5.30: Código de identificación lineal.

Para comprobar cómo se ajusta esos parámetros a la salida, lo que se hace a continuación es definir la función de transferencia equivalente a la ecuación en diferencias con los parámetros calculados. Ya que esa función de transferencia tiene la siguiente forma.

$$Gz(z) = \frac{b_9 z^9 + \cdots + b_0}{z^{10} + a_9 z^9 + \cdots + a_0} \qquad (5.6)$$

Una vez definida, se obtiene la respuesta de esa función de transferencia frente a la misma entrada real del sistema, y después se calcula numéricamente el error cuadrático medio comparando esa salida estimada con la salida real de la planta.

```
% Funcion de transferencia
gz = tf(1, 1, ts);
gz.den = [1 x(1:o)'];
gz.num = [0 x(o+1:end)']
gz = zpk(gz)
p = pole(gz)
abs(p)

% Salida del modelo frente a la misma entrada
ym = lsim(gz, v, t);

% Calcular el error cuadrático medio
ecm = sum((y-ym).^2)/length(t)
% Represntar en un mismo gráfico la salida del modelo y la real
figure('color', 'white');
plot(t, [v, y, ym], 'linewidth', 2);
legend('v', 'y', 'ym');
xlim([t(1),t(end)]);
xlabel('t');
title('Respuestas simulada y medida');
```

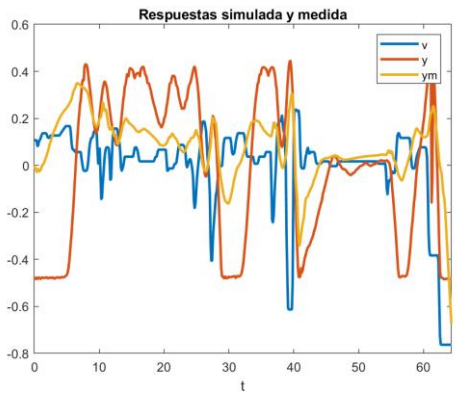


Figura 5.31: Definir la función de transferencia, comparar la diferencia de las señales de salida del modelo y la real.

```
gz =

   -0.10789 (z+0.6391) (z-1.158) (z-0.5466) (z^2 + 0.4216z + 0.4544) (z^2 - 1.43z + 0.9928) (z^2 + 0.8949z + 1.995)
  ------------------------------------------------------------------------------------------------------------------
  (z-0.9554) (z-0.8272) (z-0.4557) (z+0.3997) (z^2 + 0.9087z + 0.3918) (z^2 - 0.672z + 0.5007) (z^2 + 0.2284z + 0.4578)
```

Figura 5.32: Función de transferencia obtenida

Como se puede observar en la figura 5.31, el ajuste de la señal estimada con la medida es pésimo, llegando en algunos tramos a ni siquiera seguir la tendencia. Esto se refleja en un error cuadrático medio de 0.0728. Por lo que se llega a la conclusión de que no es posible obtener un modelo lineal decente para esta planta.

5.4.3 Identificación no lineal

Tras desestimar la opción del modelo lineal, en este apartado se intenta obtener un modelo no lineal mejor. En este caso la ecuación de diferencias se ha escogido que sea lineal en los parámetros “a” , “b”…. y la no linealidad será de cuarto orden quedando de la siguiente forma:

$$\begin{aligned} y_k = a_1 y_{k-1} + b_1 v_{k-1} + a_2 y_{k-2} + b_2 v_{k-2} + \cdots + \\ c_1 y_{k-1}^2 + d_1 v_{k-1}^2 + c_2 y_{k-2}^2 + d_2 v_{k-2}^2 + \cdots + \\ e_1 y_{k-1}^3 + f_1 v_{k-1}^3 + e_2 y_{k-2}^3 + f_2 v_{k-2}^3 + \cdots + \\ g_1 y_{k-1}^4 + h_1 v_{k-1}^4 + g_2 y_{k-2}^4 + h_2 v_{k-2}^4 + \cdots + \end{aligned} \qquad (5.7)$$

De igual manera que se procedió en el apartado “5.4.2 Identificación lineal”, se plantea y resuelve el sistema.

$$\begin{pmatrix} y_4 \\ y_5 \\ y_6 \\ \vdots \\ y_N \end{pmatrix} = \begin{pmatrix} y_3 & y_2 & y_1 & v_3 & v_2 & v_1 & y_3^2 & y_2^2 & y_1^2 & v_3^2 & v_2^2 & v_1^2 \\ y_4 & y_3 & y_2 & v_4 & v_3 & v_2 & y_4^2 & y_3^2 & y_2^2 & v_4^2 & v_3^2 & v_2^2 \\ y_5 & y_4 & y_3 & v_5 & v_4 & v_3 & y_5^2 & y_4^2 & y_3^2 & v_5^2 & v_4^2 & v_2^2 \\ & & & & & & \vdots & & & & & \\ y_{N-1} & y_{N-2} & y_{N-3} & v_{N-1} & v_{N-2} & v_{N-3} & y_{N-1}^2 & y_{N-2}^2 & y_{N-3}^2 & v_{N-1}^2 & v_{N-2}^2 & v_{N-3}^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ b_1 \\ b_2 \\ b_3 \\ c_1 \\ c_2 \\ c_3 \\ d_1 \\ d_2 \\ d_3 \end{pmatrix} \qquad (5.8)$$

La manera en la que se ha resuelto este sistema en MATLAB, es análogo a lo mostrado en el capítulo de identificación lineal, se expone a continuación.

```
o = 6;   % orden del modelo
k = o+1; % orden + 1
b = y(k:end); % Salida desde orden+1 hasta el final
m = []; % Iniciar la matriz
for n = 1:o
    m(:,end+1) = y(k-n:end-n);
end
for n = 1:o
    m(:,end+1) = v(k-n:end-n);
end
for n = 1:o
    m(:,end+1) = y(k-n:end-n).^2;
end
for n = 1:o
    m(:,end+1) = v(k-n:end-n).^2;
end
for n = 1:o
    m(:,end+1) = y(k-n:end-n).^3;
end
for n = 1:o
    m(:,end+1) = v(k-n:end-n).^3;
end
for n = 1:o
    m(:,end+1) = y(k-n:end-n).^4;
end
for n = 1:o
    m(:,end+1) = v(k-n:end-n).^4;
end
x = m\b
```

Figura 5.33: Llenado de la matriz M, y cálculo del vector de parámetros.

Al tratarse de una función no lineal, se simulará el comportamiento de este modelo, con la ecuación en diferencias cuyos parámetros acaban de ser obtenidos.

```
% Extraer del vector de parámetros los coeficientes a,b...h
a = x(1:o)
b = x(o+1:2*o)
c = x(2*o+1:3*o)
d = x(3*o+1:4*o)
e = x(4*o+1:5*o)
f = x(5*o+1:6*o)
g = x(6*o+1:7*o)
h = x(7*o+1:8*o)

% Respuesta del modelo
ym = zeros(size(y));
ym(1:o) = y(1:o);
for k = o+1:length(ym)
    ym(k) = 0;
    for n = 1:o
        ym(k) = ym(k) + a(n)*ym(k-n) + b(n)*v(k-n) + ...
                c(n)*y(k-n)^2 + d(n)*v(k-n)^2 + ...
                e(n)*y(k-n)^3 + f(n)*v(k-n)^3 + ...
                g(n)*y(k-n)^4 + h(n)*v(k-n)^4;
    end
end

% Error cuadrático medio
ecm = sum((y-ym).^2)/length(t)
% Representación salida del modlo, salida real
figure('color', 'white');
plot(t, [v, y, ym], 'linewidth', 2);
legend('v', 'y', 'ym');
xlim([t(1),t(end)]);
xlabel('t');
title('Respuestas simulada y medida');
```

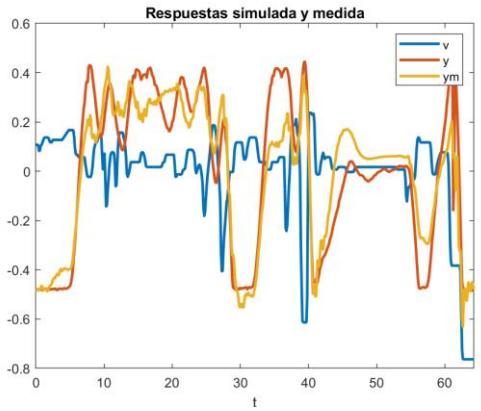


Figura 5.34: Comparación señal real y la del modelo.

Los resultados obtenidos con este método son mejores que los obtenidos previamente, a simple vista se nota que la señal del modelo es más próxima a la señal medida que en el caso anterior. Objetivamente se puede afirmar que arroja un error cuadrático medio de 0.0156, lo cual es 4.67 veces menor que en la identificación lineal.

5.4.4 Identificación neuronal

La inteligencia artificial es una rama de conocimiento que actualmente se encuentra en auge. Uno de los objetivos que persigue es conseguir que un ordenador sea capaz de imitar o aproximar su comportamiento a un proceso biológico. Tratando de emular el comportamiento real de una neurona, Rosenblatt creó una estructura que recibe el nombre de perceptrón, la cual constituye la aproximación más cercana a una neurona. Un perceptrón está formado por varios elementos, el primero de ellos, son las entradas, que bien pueden venir de las salidas de otra neurona o bien provenir del "exterior" estas entradas serían las dendritas de una neurona real. Adicionalmente a las entradas existe otra entrada que durante el funcionamiento se mantiene constante, y se la conoce como umbral o bias. Dichas entradas llegan al sumatorio multiplicadas cada una por un peso. El siguiente elemento es la función de activación, que emularía la labor del soma, es decir, determina cuando se debe disparar la señal eléctrica. Finalmente, el axón es representado como la salida.

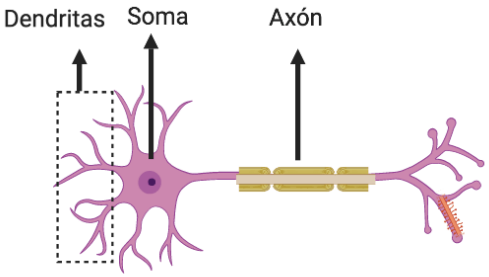


Figura 5.36: Neurona biológica [32]

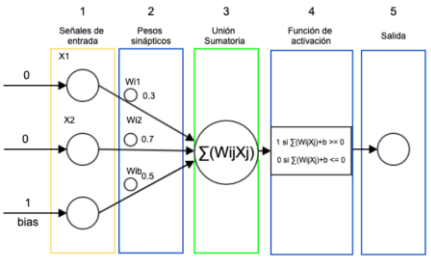


Figura 5.35: Estructura perceptrón [32]

Respecto a la función de activación, hay que comentar que existen varios tipos, pero en este caso únicamente se utilizarán la función sigmoidea para las neuronas intermedias y lineal para la neurona final.

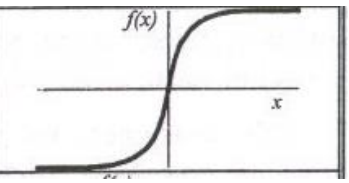
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	[0, +1] [-1, +1]	

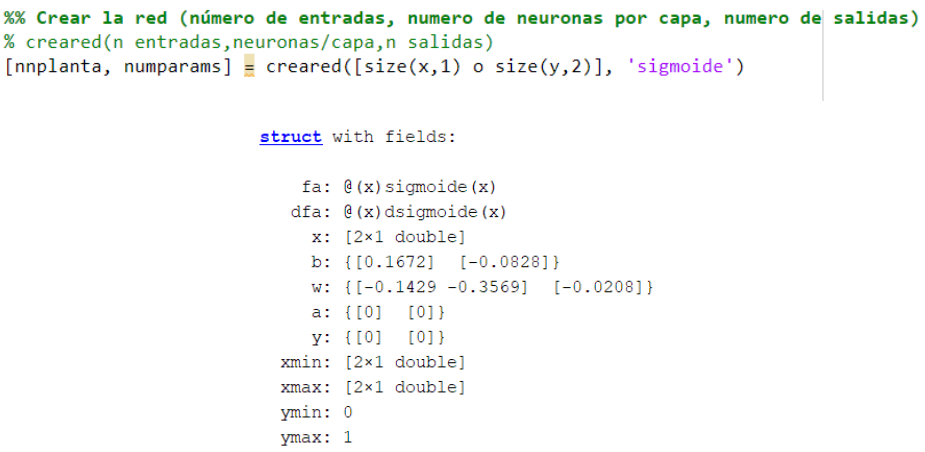
Figura 5.37: Función de activación sigmoidea [29].

Una red neuronal no deja de ser por tanto un conjunto de perceptrones conectados entre sí, generalmente suelen estar distribuidas en capas. Existen distintas tipologías de redes neuronales, pero en este caso se aplica la más simple posible, que se conocen con el nombre de redes prealimentadas o “feedforward”, en las que las salidas de las neuronas están conectadas a las entradas de las neuronas de las capas posteriores.

Debido a que existe, muchos conceptos y teoría sobre las redes neuronales, se obviará la explicación de esto, en virtud de la explicación de la aplicación práctica de identificación de la planta utilizando este método. Para ello se llevan a cabo tres pasos principales, crear la red, entrenarla y finalmente comprobar su funcionamiento. Para simplificar cada etapa, se utilizarán funciones.

Crear la red:

Se puede crear una red neuronal con el número de neuronas y capas que se desee, es decir, tan compleja como se quiera. En este caso se ha intentado buscar una solución de compromiso entre complejidad de la red y resultados obtenidos.



```
%% Crear la red (número de entradas, numero de neuronas por capa, numero de salidas)
% creared(n entradas,neuronas/capa,n salidas)
[nnplanta, numparams] = creared([size(x,1) o size(y,2)], 'sigmoide')
```

```
struct with fields:

    fa: @(x)sigmoide(x)
   dfa: @(x)dsigmoide(x)
     x: [2×1 double]
     b: {[0.1672]  [-0.0828]}
     w: {[-0.1429 -0.3569]  [-0.0208]}
     a: {[0]  [0]}
     y: {[0]  [0]}
  xmin: [2×1 double]
  xmax: [2×1 double]
  ymin: 0
  ymax: 1
```

Figura 5.38: Código para crear la red y resultados.

La red neuronal creada tiene dos neuronas, una oculta (con función de activación sigmoidea) y la otra correspondiente a la final (función de activación lineal). Los valores de bias y pesos son iniciados aleatoriamente por la función y deberán ser ajustados en la siguiente etapa, el entrenamiento.

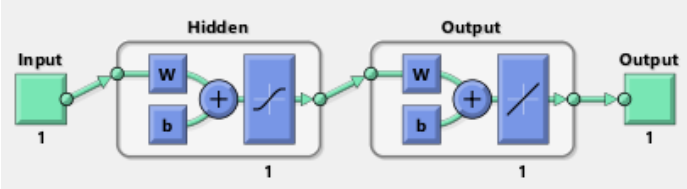


Figura 5.39: Representación de la red neuronal creada.

Entrenamiento de la red

Para entrenar la red se ha hecho uso de la función “entrenared”, cuyos parámetros de entrada y salida se muestran en la figura (5.40)

```
%% Entrenamiento de la red
% Parámetros entrenamiento
g = 1e0;        % Factor de aprendizaje
Ne = 1000;      % Número de epoch máximo (n de veces que se pasan los datos)
Emin = 1e-9;    % Error cuadrático deseado
ntest = 0.25;   % Porcentaje de muestras para test (el resto para entreno)
batchsize = 1;  % tamaño del batch
% Entrenamiento de la red
t0 = tic; % Temporizador para obtener el tiempo de entreno
%entrenared
[nnplanta, ebest, ecm, g] = entrenared(nnplanta, x, tg, g, Ne, Emin, ntest, batchsize);
toc(t0)
nnplanta
```

```
struct with fields:

    fa: @(x)sigmoide(x)
   dfa: @(x)dsigmoide(x)
     x: [2×1 double]
     b: {[-1.1700]  [-0.4181]}
     w: {[2.4632 0.0770]  [1.7462]}
     a: {[-1.1700]  [-0.0045]}
     y: {[0.2369]  [-0.0045]}
  xmin: [2×1 double]
  xmax: [2×1 double]
  ymin: 0
  ymax: 92.9036
```

Figura 5.40: Código de entrenamiento de la red y resultados.

Como se muestra en la siguiente imagen, los resultados son bastente buenos, en azul se encuentran los puntos utilizados para el entrenamiento, mientras en tono rojo se encuentran los correspondientes a la simulación de la red frente al 25% de nuevos puntos para la red. Finalmente en negro se representa la recta de la red neuronal. Los ajustes de los puntos de entrenamiento y validación se ajustan a esa recta en una 98.8% y 99.6% respectivamente, lo cual son factores bastante elevados teniendo en cuanta la simplicidad de la red neuronal.

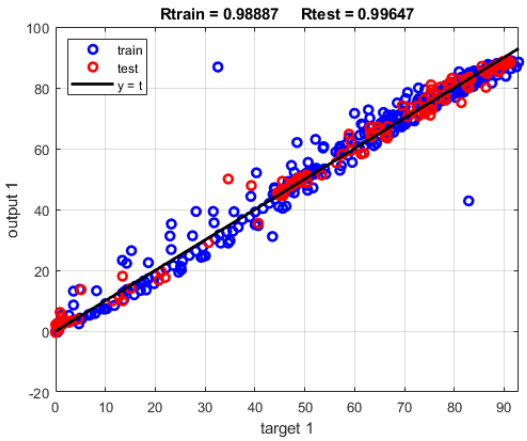


Figura 5.41: Puntos de entrenamiento y de prueba.

Simulación de la red

Finalmente, para mostrar los buenos resultados obtenidos se visualiza en un mismo gráfico la salida real de la planta (medida) y la salida de la red neuronal.

```
%% Comparativa de la respuesta simulada y medida
ym = zeros(size(t)); ym(1:2) = y(1:2);
[ym(ny+1:end), nnplanta] = calculared(nnplanta, x);
figure('color', 'white');
plot(t, [v, y, ym], 'linewidth', 2);
legend('v', 'y', 'ym');
xlim([t(1),t(end)]);
xlabel('t');
title('Respuestas simulada y medida');
ecm = sum((y/100-ym/100).^2)/length(t)
```

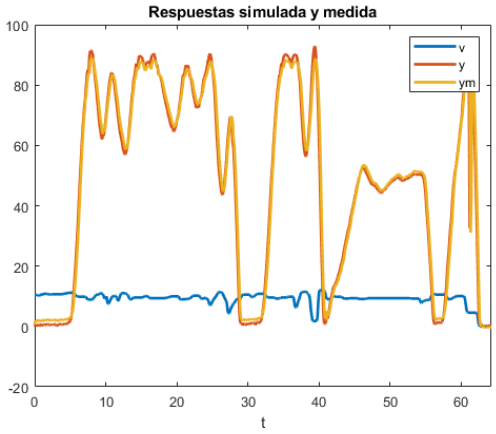


Figura 5.42: Comparativa entre la señal medida y la del modelo de la red neuronal.

Como se puede, apreciar en la figura 5.42, la salida del modelo se ajusta prácticamente a la perfección a la salida que debería dar (salida medida), y esto se traduce en un error cuadrático medio de tan solo 0.0018, unas diez veces menor que el modelo no lineal obtenido previamente.

A modo de conclusión de este apartado, hay que comentar que únicamente se ha sido capaz de obtener un buen modelo realizando la identificación con una red neuronal. Los modelos lineales y no lineales que se han mostrado no podrían ser utilizados por ejemplo para diseñar un regulador o incluir un observador de estados a la planta, ya que no se ajustan al comportamiento real del sistema. También se ha demostrado que el modelo lineal obtenido en el apartado 4.3 MODELO MATEMÁTICO, únicamente queda relegado a comprender los fenómenos físicos que tienen lugar en la planta.

6 PRESUPUESTO

6.1 PRECIOS UNITARIOS

Tabla 6.1: Precios unitarios.

Código	Unidad	Descripción	Referencia fabricante	Precio [€]	Precio escrito
C1	u	Cond. Electrolítico 2200uF	CE220025	0.69	SESENTA Y NUEVE CÉNTIMOS
C2	u	Cond. Electrolítico 330nF	CE22025	0.18	DIECIOCHO CÉNTIMOS
C3	u	COND.MKT 1uF	Desconocido	0.19	DIECINUEVE CÉNTIMOS
J1	u	Jack alimentación 3.6mm	J11202	0.88	OCHENTA Y OCHO CÉNTIMOS
DIS1	u	Disipador pequeño	18205	1.28	UN EURO CON VEINTIOCHO CÉNTIMOS
I1	u	Conmutador 11435	C11435	0.80	OCHENTA CÉNTIMOS
P1	u	Pulsador	R1829B	1.07	UN EURO CON SIETE CÉNTIMOS
uC1	m	Microcontrolador	FUNMEGA	18.95	DIECIOCHO EUROS CON NOVENTA Y CINCO CÉNTIMOS
T1	u	Transistor MOSFET	IRF520	2.39	DOS EUROS CON TREINTA Y NUEVE CÉNTIMOS
D1	u	Diodo	1N4007	0.15	QUINCE CÉNTIMOS

REG1	u	Reg. tensión Positivo 12v	7812P	0.79	SETENTA Y NUEVE CÉNTIMOS
REG2	u	Reg. tensión Positivo 5v	7805P	1.56	UN EURO CON CINCUENTA Y SEIS CÉNTIMOS
I2	u	Conmutador balancín	CM11185	0.99	NOVENTA Y NUEVE CÉNTIMOS
L1	u	LED 5mm rojo	Desconocido	0.20	VEINTE CÉNTIMOS
L2	u	LED 5mm rojo	Desconocido	0.20	VEINTE CÉNTIMOS
R1	u	Resistencia 180 Ω 1/4W	Desconocido	0.10	DIEZ CÉNTIMOS
R2	u	Resistencia 1$k\Omega$ 1/4W	Desconocido	0.10	DIEZ CÉNTIMOS
R3	u	Resistencia 100 $k\Omega$ 1/4W	Desconocido	0.10	DIEZ CÉNTIMOS
R4	u	Resistencia 4,7 $k\Omega$ 1/4W	Desconocido	0.10	DIEZ CÉNTIMOS
F1	u	Filamento PLA 750	PLA175BS07	19.00	DIECINUEVE EUROS
S1	u	Sensor ultrasónico	WPSE306	5.74	CINCO EUROS CON SETENTA Y CUATRO CÉNTIMOS
V1	u	Ventilador 120 12V	VEN015	8.73	OCHO EUROS CON SETENTA Y TRES CÉNTIMOS
PANT1	u	Pantalla 2.8”	WPSH412	16.45	DIECEISEIS EUROS CON CUARENTA Y CINCO EUROS
CPOT	u	Capucha potenciómetro.	Desconocido	1.00	UN EURO

ES1	u	Escala numerada	Desconocido	5.00	CINCO EUROS
AL1	u	Perfil de aluminio anodizado 20x20mm	Desconocido	3.56	TRES EUROS CON CINCUENTA Y SEIS CÉNTIMOS
AL2	u	Perfil de aluminio anodizado 50x2mm	Desconocido	4.12	CUATRO EUROS CON DOCE CÉNTIMOS
MTR1	u	Lámina de metacrilato 1m	Desconocido	2.50	DOS EUROS CON CINCUENTA CÉNTIMOS
PCBV	u	Placa Virgen PCB	Desconocido	2.51	DOS EUROS CON CINCUENTA Y UN CÉNTIMOS
POT1	u	Potenciómetro 10 $k\Omega$	Desconocido	1.40	UN EURO CON CUARENTA CÉNTIMOS

6.2 CUADRO DE PRECIOS DESCOMPUESTOS

6.2.1 Estructura

Tabla 6.2: Precios descompuestos: Estructura.

PRECIO DESCOMPUESTO: Estructura					
Código		**Unidad**	**Designación**		
ESTR1		u	Estructura impresa en PLA, con refuerzos de aluminio, conducto de metacrilato y complementos.		
Código	**Unidad**	**Cantidad**	**Concepto**	**Precio unitario**	**Total**
F1	u	1	Bobina PLA	19.00€	19.00€
AL1	u	1	Perfil de aluminio anodizado 20x20mm	3.56€	3.56€
AL2	u	1	Perfil de aluminio anodizado 50x2mm	4.12€	4.12€
MTR1	u	1	Lámina de metacrilato 1m	2.50€	2.50€
ES1	u	1	Escala numerada	5.00€	5.00€
				Total	**34.18€**
La unidad de la partida ESTR1, importa TREINTA Y CUATRO EUROS CON DIECIOCHO CÉNTIMOS					

6.2.2 Hardware

Tabla 6.3: Precios descompuestos: Estructura.

PRECIO DESCOMPUESTO: PCB electrónica de potencia					
Código		**Código**	**Código**		
PCB1		u	PCB correspondiente a la electrónica de potencia, grabada y soldada.		
Código	**Unidad**	**Cantidad**	**Concepto**	**Precio unitario**	**Total**
PCBV	u	1	PCB Virgen	2.51€	2.51€
C1	u	1	Cond. Electrolítico 2200uF	0.69€	0.69€
C2	u	1	Cond. Electrolítico 330nF	0.18€	0.18€
C3	u	1	Cond. MKT 1uF	0.19€	0.19€
R2	u	1	Resistencia 1K ohmios 1/4W	0.10€	0.10€
R3	u	1	Resistencia 100K ohmios 1/4W	0.10€	0.10€
D1	u	5	Diodo	0.15€	0.75€
REG1	u	1	Reg. tensión Positivo 12v	0.79€	0.79€
REG2	u	1	Reg. tensión Positivo 5v	1.56€	1.56€
T1	u	1	Transistor MOSFET	2.39€	2.39€
DIS1	u	3	Disipador pequeño	1.28€	3.84€
				Total	**13,10€**
La unidad de la partida PCB1, importa TRECE EUROS CON DIEZ CÉNTIMOS					

Tabla 6.4: Precios descompuestos: Arduino, periféricos y actuador.

PRECIO DESCOMPUESTO: Arduino, periféricos y actuador					
Código		**Código**	**Código**		
ARD_PERIF		u	Equipo principal Arduino Mega, con sensor ultrasónico, periféricos y actuador.		
Código	**Unidad**	**Cantidad**	**Concepto**	**Precio unitario**	**Total**
J1	u	1	Jack alimentation 3.6mm	0.88€	0.88€
I1	u	1	Conmutador 11435	0.80€	0.80€
P1	u	1	Pulsador R1829B	1.07€	1.07€
uC1	u	1	Microcontrolador FUNMEGA /Arduino MEGA	18.95€	18.95€
I2	u	2	Conmutador balancín	0.99€	1.98€
L1	u	1	Led 5mm Rojo	0.20€	0.20€
L2	u	1	Led 5mm Verde	0.20€	0.20€
R1	u	2	Resistencia 1/4W 180 ohmios	0.10€	0.20€
R4	u	3	Resistencia 1/4W 4.7 K ohmios	0.10€	0.30€
S1	u	1	Sensor ultrasónico	5.74€	5.74€
PANT1	u	1	Pantalla 2.8”	16.45€	16.45€
V1	u	1	Ventilador 120mm 12V	8.73€	8.73€
POT1	u	3	Potenciómetro 10 K ohmios	1.40€	4.20€
				Total	**59.70€**
La unidad de la partida ARD_PERIF, importa CINCUENTA Y NUEVE EUROS CON SETENTA CÉNTIMOS					

6.3 DESGLOSE DEL PRESUPUESTO

Tabla 6.5: Desglose del presupuesto.

CAPÍTULO 1: Estructura				
Código	Descripción	Cantidad	Precio	Importe
ESTR1	Estructura de la planta	1	34.18€	34.18€
TOTAL, CAPÍTULO ESTRUCTURA				**34.18€**
CAPÍTULO 2: Hardware				
Código	Descripción	Cantidad	Precio	Importe
PCB1	Placa electrónica de potencia	1	13.10€	13.10€
ARD_PERIF	Arduino, periféricos y sensor	1	59.70€	59.70€
TOTAL, CAPÍTULO HARDWARE				**72.80€**
TOTAL, PRESUPUESTO (SIN I.V.A)				**106.98€**
I.V.A (21%)				**22.47€**
TOTAL, CON I.V. A				**129.45€**
CON IMPUESTOS, LA CUANTÍA TOTAL DEL PRESUPUESTO ASCIENDE A CIENTO VEINTINUEVE EUROS CON CUARENTA Y CINCO CÉNTIMOS.				

Remarcar que este presupuesto no tiene en cuenta los gastos derivados de la impresión 3D (amortización de la maquinaria, ni consumibles, ni consumo eléctrico), así como tampoco se han tenido en cuenta los gastos de mano de obra, tanto de labores de diseño, soldadura o mecanizado. Tampoco se han tenido en cuenta los precios de elementos reciclados, los cuales podrían asumir como costo nulo. Se ha decidido realizarlo de esta manera para aportar una idea del coste general de este equipo en concreto.

7 CONCLUSIONES Y TRABAJOS FUTUROS

La realización de este trabajo ha supuesto un reto, ya que se ha diseñado cada elemento de la planta, es decir, todo aquello que permite ser diseñado y fabricado con unos medios de coste asumible, así se ha hecho. Ejemplo de esto se puede encontrar en prácticamente todos los elementos del sistema, PCB, estructura de la planta, conducto de levitación, soportes, refuerzos de aluminio …etcétera. Dicha labor hace que el diseño y la fabricación de la pieza sea el protagonista y el centro de atención en cada fase, ya que el diseño de cada elemento interacciona y condiciona el resto de los componentes. Por tanto, el tener una visión global del conjunto en las fases de diseño es fundamental. A pesar de esa visión general, la magnitud y la envergadura del proyecto hace que sea muy difícil tener en cuenta todas las restricciones de diseño condicionadas por cada elemento. Por ese motivo, se ha comprobado lo fundamental que resulta la validación del diseño, mediante el uso de herramientas software, previa construcción. Sin la ayuda de herramientas como SPICE o los ensamblajes de la herramienta de diseño CAD (en este caso SOLIDWORKS), resultaría muy complejo realizar un buen diseño sin partir de ningún modelo existente, como es el caso. Por otro lado, la realización del trabajo ha evidenciado la labor multidisciplinar a la que se debe afrontar cuando se pretende solventar un problema real. Durante la realización de este trabajo, se han utilizado conocimientos de neumática, física, teoría de control, electrónica de potencia, diseño asistido por computador, programación en Python, programación en C++, sensores, buses de comunicación, cálculo.… e incluso aunque en menor medida química. Dominar cada uno de esos apartados que se han enumerado, es bastante complicado ya que en la mayoría de los casos está asociado a la experiencia, por lo que, el conseguir que cada pieza de la planta trabaje al unísono, también es una muestra de la formación previa, investigación y capacidad de solución autónoma de problemas, que a menudo son labores que pasan desapercibidas. La idea original era diseñar desde cero, una planta para el uso didáctico por parte de los alumnos, pero previo a la decisión del tipo de sistema, se identificaron las necesidades que debía satisfacer la misma. Tras ello, otra labor que, quizás no se ha detallado en profundidad, es el proceso de la búsqueda de un tipo de sistema que se adapte a las necesidades. Este proceso, consiste en realizar pequeños estudios de viabilidad, tanto funcional, como económica con la sorprendente cantidad de horas previas antes incluso de decidir la naturaleza de la planta. También a medida que se ha ido avanzando en el desarrollo del trabajo, se aprecia que el proyecto ha sufrido varias modificaciones, encaminadas a la implementación de mejoras que no estaban contempladas en el diseño inicial. Otra capacidad que se ha adquirido durante la realización del trabajo es la de adaptarse a las piezas y

especialmente componentes electrónicos, ya que tras realizar varios pedidos y comenzar el diseño y programación offline con los componentes pedidos, debido a la que ya se denomina como la “crisis de los semiconductores” se tuvieron que modificar algunos componentes para cumplir con la fecha ejecución de la planta.

Habiendo finalizado la ejecución material de la planta, así como la redacción de este documento, se ha podido construir un sistema que cumple las expectativas, ya que se ha conseguido controlar un sistema que es no lineal y bastante complejo debido a las perturbaciones provocadas por las turbulencias del flujo de aire. Además de mostrar ese sistema de forma compacta, estética y sencilla.

Otro de los aspectos que se tenían claros al comenzar la planta, era permitir que los propios alumnos que utilicen esta planta el próximo año académico, se animen a programar ellos mismos el regulador que consideren óptimo y de esta manera experimentar la teoría de control mediante la puesta en práctica. Es más, muchas de las condiciones de diseño relacionadas con el tema de control se han preferido dejar en su modo más elemental. Un ejemplo de esto es el regulador PID que compila Arduino. Como se ha mostrado, es un PID simple, sin ninguna mejora, como antiwindup, filtro de la derivada, y esto se ha hecho así precisamente para evidenciar los problemas que se pueden encontrar si se implementa un PID simple. Lo que se pretende es que los alumnos detecten esas carencias del regulador precargado en el Arduino y lo intenten solventar diseñando un controlador mediante el uso de la aplicación.

Pero no solo eso, también se pretende que implementen algunas mejoras sobre la propia planta. Algunas de esas posibles mejoras se detallan a continuación:

- Modificar la interfaz de usuario gráfica y el código Python asociado, para que en la ventana de código exista alguna opción para depurar el propio código del controlador.
- Añadir a la interfaz gráfica y el código Python, algún tipo de control para poder variar algunos parámetros del regulador que se esté utilizando en cada caso, de esta forma, la planta seria aún más interactiva.
- Incluir otro ventilador en la parte superior, con la finalidad de controlar un sistema con doble actuador.
- Realizar un modelado de la planta más exhaustivo, teniendo en cuenta datos obtenidos de la propia planta, teóricos y además datos procedentes de simuladores de flujos.

- Añadir posibilidad de incluir controladores en el dominio de Laplace, y que para ser ejecutado se obtenga internamente la ecuación en diferencias.
- Realizar un control mediante redes neuronales en Python.

8 AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a todos los docentes que me han impartido clase en estos cuatro años en la universidad de Cantabria, en especial al director del trabajo, Luciano Alonso Rentería, a quien he acudido para realizar consultas en momentos puntuales. En concreto destacar su inestimable ayuda en el proceso de identificación de la planta ya que, pese a la dificultad de esta, finalmente se consiguieron resultados buenos.

También a mi familia por todo el apoyo que me han brindado a lo largo de este camino.

9 BIBLIOGRAFÍA

[1] C. T. Ferrero, «MÉTODOS Y TÉCNICAS DE CONTRO CONCEPTOS Y COMPONENTES BÁSICOS».

[2] F. O. Tellez, «CONTROL ANALÓGICO I».

[3] «MATHWORKS,» [En línea]. Available: https://es.mathworks.com/products/simulink.html. [Último acceso: 01 06 2022].

[4] «BYTRONIC,» [En línea]. Available: http://www.bytronic.net/. [Último acceso: 5 6 2022].

[5] K. Ogata, «Controles PID e introducción al control robusto,» de *INGENIERÍA DE CONTROL MODERNA, tercera edición*, Pearson.

[6] «of3lia,» [En línea]. Available: https://of3lia.com/pla-vs-abs-vs-petg-comparativa/.

[7] A. H. Donate, «Circuitos básicos de alimentación,» de *Electrónica aplicada*, marcombo.

[8] «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Ciclo_de_trabajo.

[9] «Universidad de Córdoba, valor medio y valor eficaz,» [En línea]. Available: http://www.uco.es/grupos/giie/cirweb/teoria/tema_01/tema_01_06.pdf .

[10] L. Llamas, «Controlar grandes cargas con Arduino y transistor MOSFET,» [En línea]. Available: https://www.luisllamas.es/arduino-transistor-mosfet/.

[11] «Wikipedia, Tabla de potenciales de reducción,» [En línea]. Available: https://es.wikipedia.org/wiki/Anexo:Tabla_de_potenciales_de_reducci%C3%B3n.

[12] «Universidad de Oviedo, Modelado de un motor CC,» [En línea]. Available: http://isa.uniovi.es/~idiaz/ADSTel/Practicas/ModeladoMotorCC.html.

[13] C. T. Ferrero, REPRESENTACIÓN DE LOS SISTEMAS DE CONTROL, MOTOR CC.

[14] «What is Drag Coefficient – Drag Characteristics – Definition,» [En línea]. Available: https://www.thermal-engineering.org/what-is-drag-coefficient-drag-characteristics-definition/.

[15] «Drag coefficient,» [En línea]. Available: https://automobile.fandom.com/wiki/Drag_coefficient .

[16] «eduteka,» [En línea]. Available: https://eduteka.icesi.edu.co/imgbd/27/27-09/TablaComparativaArduino.jpg.

[17] «velleman,» [En línea]. Available: https://www.velleman.eu/products/view/?id=463692.

[18] «Wikipedia, velocidad del sonido,» [En línea]. Available: https://es.wikipedia.org/wiki/Velocidad_del_sonido.

[19] «arduinoamuete, Sensor ultasónico HC -SR04,» [En línea]. Available: http://arduinoamuete.blogspot.com/2016/12/sensor-ultrasonico-hc-sr04.html.

[20] «SuperRobotica; Sensor distancias por ultrasonidos SRF04,» [En línea]. Available: http://www.superrobotica.com/s320110.htm.

[21] «Hardwarelibre; VL53L0X: sensor de distancia láser de alta precisión,» [En línea]. Available: https://www.hwlibre.com/vl53l0x/.

[22] «RS-online; MIKROE 55,» [En línea]. Available: https://es.rs-online.com/web/p/kits-de-desarrollo-de-pantallas/7916476?cm_mmc=ES-PLA-DS3A-_-google-_-CSS_ES_ES_Raspberry_Pi_%26_Arduino_y_Modulos_de_Desarrollo_Whoop-_-(ES:Whoop!)+Kits+de+Desarrollo+de+Pantallas-_-7916476&matchtype=&pla-477731620556&gclid.

[23] S. A. C. Giraldo, «Comunicación I2C,» [En línea]. Available: https://controlautomaticoeducacion.com/microcontroladores-pic/comunicacion-i2c/.

[24] L. Llamas, «EL bus SPI en Arduino,» [En línea]. Available: https://www.luisllamas.es/arduino-spi/.

[25] «velleman; PANTALLA TÁCTIL 2.8",» [En línea]. Available: https://www.velleman.eu/products/view?id=435582&country=us&lang=es.

[26] L. Llamas, «Leer un pulsador,» [En línea]. Available: https://www.luisllamas.es/leer-un-pulsador-con-arduino/.

[27] «System Requirements for MATLAB R2022a,» [En línea]. Available: https://es.mathworks.com/support/requirements/matlab-system-requirements.html.

[28] «Python Minimum System Requirements and Installation,» [En línea]. Available: https://app.iclasspro.com/portal/sanfrancisco/news/15.

[29] F. J. P. Burgos, «ibiblio,» [En línea]. Available: https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x38.html.

[30] «wikimedia,» [En línea]. Available: https://commons.wikimedia.org/wiki/Category:Pulse_width_modulation#/media/File:Pwm2.png.

[31] Sensores Potenciométicos, ITES PARANINFO.

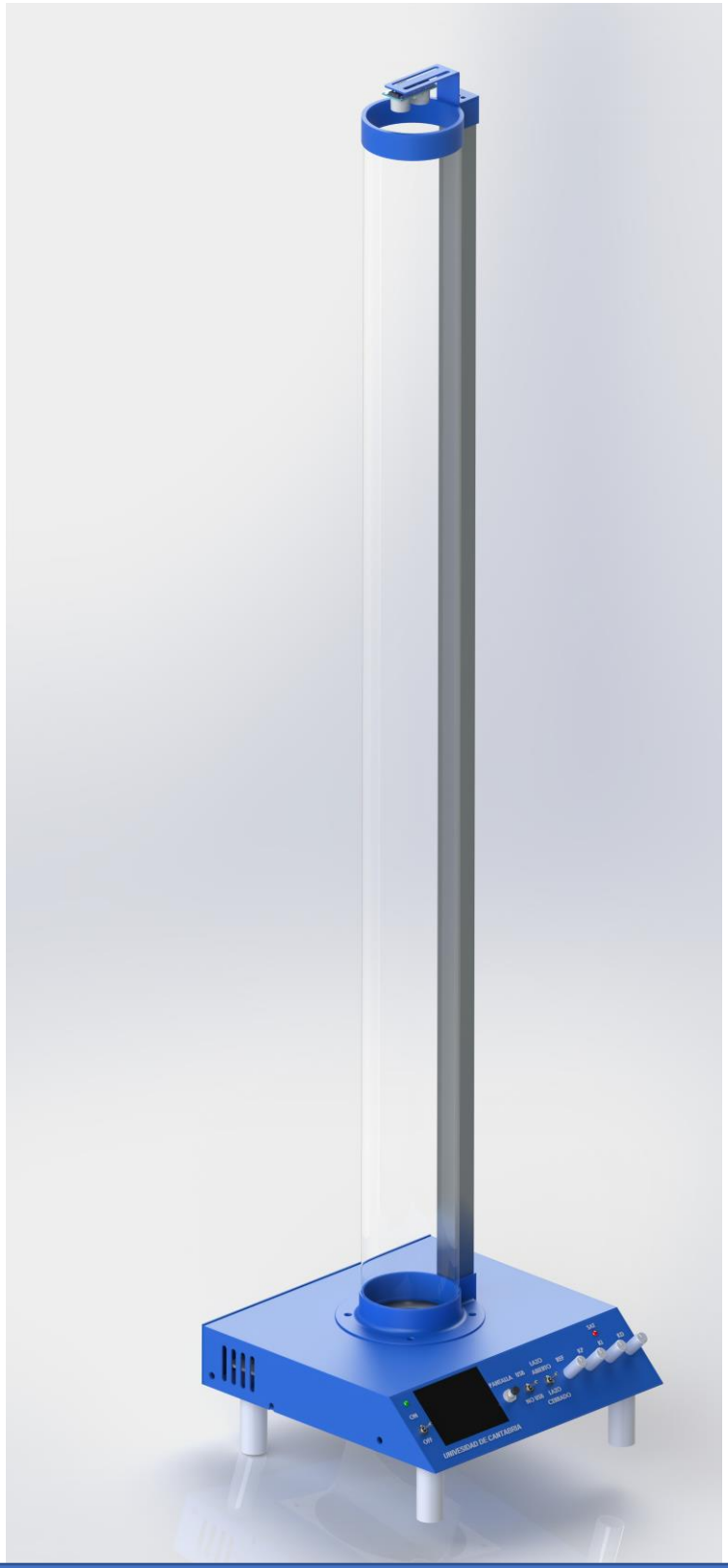
[32] «Introducción a las Redes Neuronales,» [En línea]. Available: https://futurelab.mx/redes%20neuronales/inteligencia%20artificial/2019/06/25/intro-a-redes-neuronales-pt-1/.

[33] «TIOBE,» [En línea]. [Último acceso: 06 2022].

[34] «PYPL,» [En línea]. Available: https://pypl.github.io/PYPL.html. [Último acceso: 06 2022].

ANEXOS

Guía de inicio rápido para el manejo de la planta de levitación neumática



Universidad de Cantabria

ANEXOS

Contenidos:

1. Elementos.
2. Funcionamiento general.
3. Modos de visualización.
4. Conexión y funcionamiento con la aplicación.
5. Errores y soluciones

1.Elementos:

Elemento	Nomenclatura	Funcionalidad	Elemento	Nomenclatura	Funcionalidad
1	LED Encendido	Indica que la planta está preparada para ser usada	**7**	Ruleta ajuste de referencia	Permite variar la consigna del sistema
2	Interruptor de alimentación del microcontrolador	Alimenta el microcontrolador	**8**	Ruleta de ajuste de la ganancia proporcional	Permite variar la ganancia proporcional de la planta
3	Pantalla LCD	Mostrar información	**9**	Led Saturación	Indica que el actuador no

					admite más tensión
4	Pulsador	Permite el cambio del modo de visualización	**10**	Ruleta de ajuste de ganancia integral	Permite variar la ganancia integral de la planta
5	Interruptor de modo de funcionamiento	Seleccionar si se desea o no utilizar el equipo conectado a un ordenador	**11**	Ruleta de ajuste de la ganancia derivativa	Permite variar la ganancia derivativa de la planta
6	Interruptor realimentación	Solo para el modo “NO USB”, permite seleccionar si se desea realizar un control en lazo abierto o cerrado	**12**	Interruptor general	Corta la alimentación de la planta

2.Funcionamiento:

Enchufe el cable de alimentación a un enchufe convencional, tras ello permita el paso de la corriente activando el interruptor general 12 (parte trasera del equipo). Si desea alimentar el equipo principal de la planta, accione en interruptor 1 a la posición ON. La pantalla de la planta se enciende y los leds parpadean indicando que el equipo se está preparando para ser usado. Al finalizar esta etapa se apaga el led rojo, manteniéndose el led verde encendido, la planta está preparada para su uso. La planta puede operar en distintos modos, pero independientemente del seleccionado los interruptores 1 y 12 deben estar en la posición ON.

⚠ Tras su uso desconecte la planta de la red eléctrica, únicamente utilizar la planta bajo supervisión humana.

Acceso y funcionamiento de cada modo.

-MODO NO USB:

Seleccionable colocando el interruptor 2 en su posición inferior “NO USB”, este modo hace que la planta trabaje de forma aislada, sin necesidad de un ordenador. Además, si este modo se encuentra activo, permite escoger entre realizar un control en lazo abierto o en lazo cerrado dependiendo la posición del interruptor 6.

- MODO NO USB; LAZO ABIERTO:

Mueva el interruptor 6 a su posición “LAZO ABIERTO”

La señal que se envía al actuador es igual que la referencia. La referencia puede ser ajustada con la ruleta de referencia 7. El resto de las ruletas no serán tenidas en cuenta.

- MODO NO USB; LAZO CERRADO:

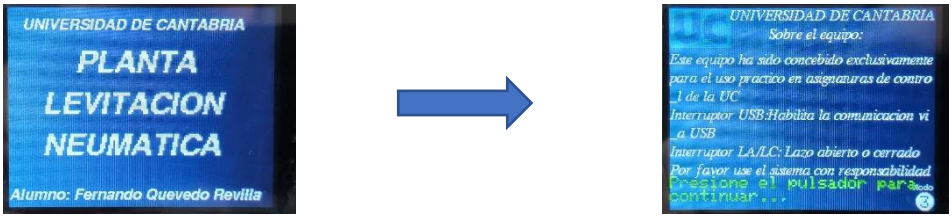
Mueva el selector 6 a la posición “LAZO CERRADO”, tras ello internamente se ejecutará el control mediante un regulador PID, con las ganancias ajustables a través de las ruletas 8, 10 y 11. Se permite variar la consigna girando la ruleta de referencia 7.

-MODO USB:

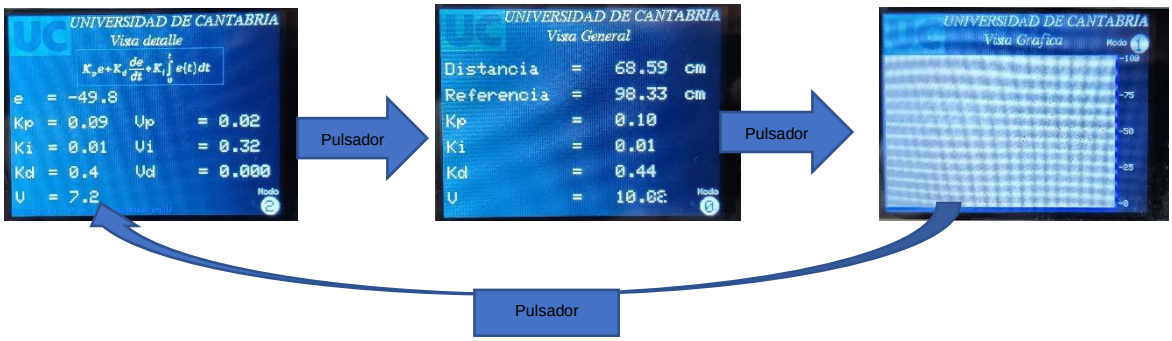
Mueva el interruptor 5 a la posición “USB”. Este modo de funcionamiento hace que la planta trabaje en sincronía, con el ordenador que esté conectado a través del puerto USB. Por tanto, este modo precisa que el ordenador, disponga de la aplicación de escritorio de la planta. El interruptor 6 y todas las ruletas del equipo carecen de funcionalidad en este modo.

3.Modos de visualización:

Al colocar el interruptor 1 en su posición “ON”, se enciende la pantalla, mostrando la pantalla de bienvenida. Tras ello aparece una descripción general del funcionamiento.

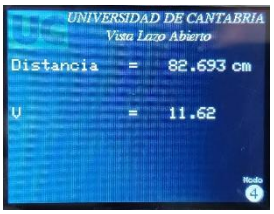


Para continuar se debe presionar el pulsador 4. Se muestra el modo de visualización cero o “Vista general”, para cambiar de modo de visualización presione el pulsador de nuevo.



Esta secuencia de modos de visualización se ve interrumpida si:

- El sistema está en modo “NO USB” (interruptor 5) y en “LAZO ABIERTO” (interruptor 6), simultáneamente. En ese caso se muestra la pantalla específica de este modo.



- El sistema se encuentra en modo USB (interruptor 5), se muestra la vista USB.

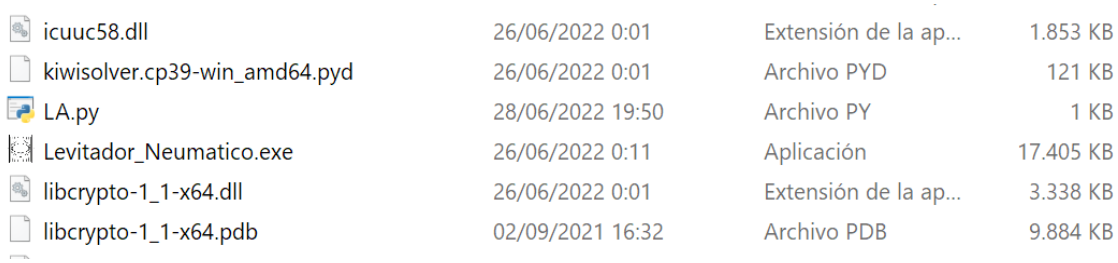
Si se ha cambiado la posición de los interruptores 5 y 6, y se desea volver al flujo normal de modos de visualización, presionar el pulsador.

4.Conexión y funcionamiento con la aplicación:

Para conectar el ordenador a la planta se recomienda seguir la siguiente secuencia:

- Conecte el cable de alimentación a la red y mueva el interruptor de alimentación general 12 a su posición “ON”.
- Mueva el interruptor 1 a su posición “ON”.
- Conecte el cable USB al ordenador.
- Lance la aplicación con permisos de administrador.

La aplicación se suministra comprimida en un archivo “.rar”, descomprímala y cree un acceso directo del ejecutable “Levitador_Neumatico.exe”. Tras ello ejecute la aplicación como administrador.



icuuc58.dll	26/06/2022 0:01	Extensión de la ap...	1.853 KB
kiwisolver.cp39-win_amd64.pyd	26/06/2022 0:01	Archivo PYD	121 KB
LA.py	28/06/2022 19:50	Archivo PY	1 KB
Levitador_Neumatico.exe	26/06/2022 0:11	Aplicación	17.405 KB
libcrypto-1_1-x64.dll	26/06/2022 0:01	Extensión de la ap...	3.338 KB
libcrypto-1_1-x64.pdb	02/09/2021 16:32	Archivo PDB	9.884 KB

Para establecer la comunicación con la planta acuda a la página de ajustes desde el menú izquierdo. Presione Buscar.

Vuelva a comprobar la posición de los selectores que se indican en dicha página.

Aparecerán en los desplegables los ajustes necesarios. Si no aparecen, acuda al apartado solución de problemas de esta guía rápida.

Para iniciar el control, acuda a la página “Principal” y cargue o cree un nuevo controlador.

⚠ El archivo Python del controlador se debe encontrar en el mismo directorio que el programa y además se recomienda poseer permisos de administrador.

Si no dispone de ningún controlador, en la carpeta de la aplicación se incluyen el de lazo abierto “LA.py” y un PID “cotroladorlibpid.py”.

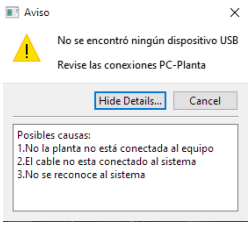
Una vez cargado, presione iniciar para comenzar el control, mueva el valor de la referencia como desee. Aparecen las señales representadas en el gráfico de la derecha.

⚠ Por defecto, el tamaño de la gráfica es de 10 segundos, puede hacer que se aumente según trascurre el tiempo presionando el botón que se encuentra en la esquina inferior izquierda.

Si desea parar la comunicación presione “PARAR”, una vez parado se permite exportar las señales que aparecen en la gráfica, si así lo desea presione “Guardar señales”.

4.Solución de problemas:

4.1 Aparece el siguiente aviso:

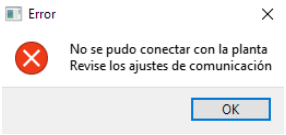


Problema: No se encuentra la planta.

Soluciones propuestas:

- Revise las conexiones USB.
- Revise la configuración de los puertos COM del ordenador.
- Encienda la planta y seleccione el modo USB.
- Cambie de puerto USB del ordenador.
- Si el problema persiste apague totalmente la planta y reinicie el ordenador.

4.2 Aparece el siguiente error:

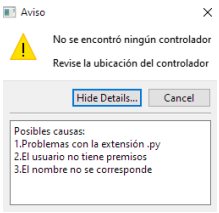


Problema: Los ajustes de comunicación están vacíos.

Solución:

- Acuda a la página de ajustes y presione buscar

4.3 Aparece el siguiente aviso.



Problema: No se pudo cargar el controlador.

Solución:

- Revise que el controlador se encuentra en el directorio de la aplicación.
- Revise que la extensión del controlador es “.py”
- Revise que la aplicación dispone de permisos de administrador para abrir ese archivo.

4.4 No aparece ningún error, pero no se guardan los archivos:

Solución:

- Revise que se tiene permiso para escribir en el directorio deseado.
- Pruebe en otro directorio.

4.5 La planta no es capaz de hacer levitar la esfera o está muy caliente:

Solución:

- Desconecte totalmente la planta de la red durante 10 minutos y vuelva a intentarlo.

4.6 La planta no se enciende:

Solución:

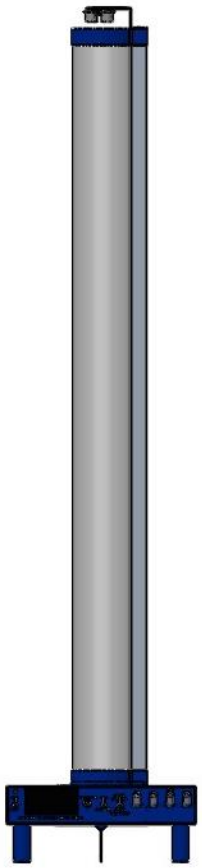
- Revise la posición de los interruptores 1 y 12.

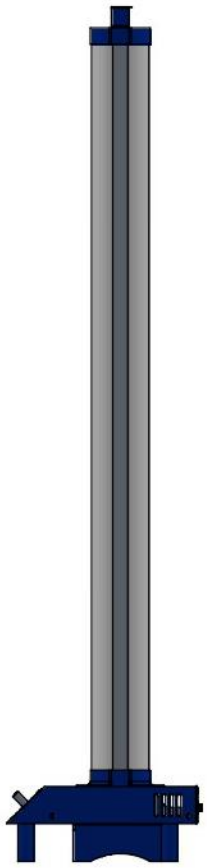
4.7 Otros problemas:

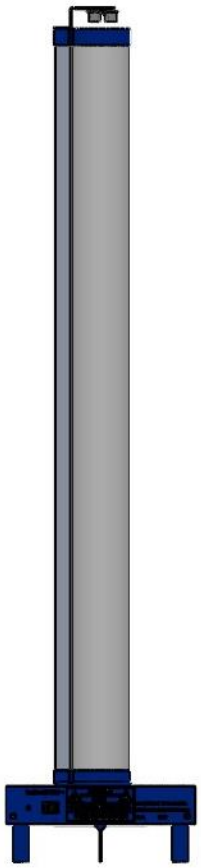
Solución:

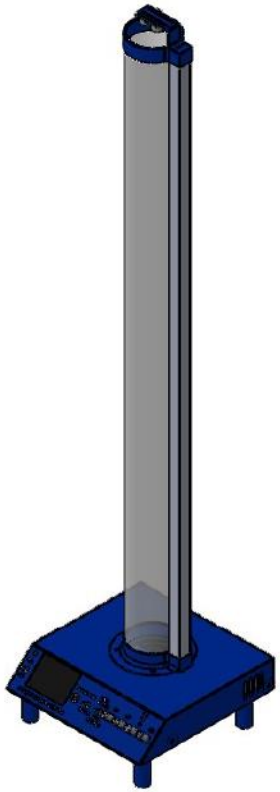
- Desenchufe la planta y consulte al docente.

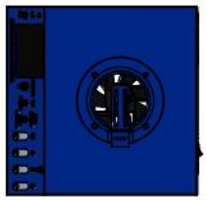
ANEXO 2: MODELOS CAD



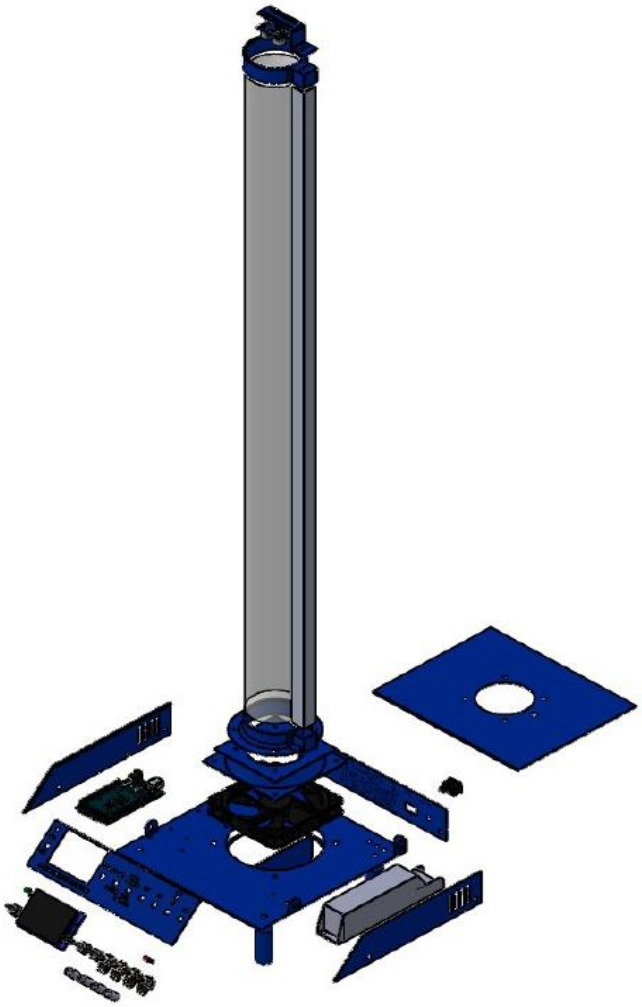








SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:				REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO: VISTA EN CONJUNTO GENERAL
DIBUJ.	FERNANDO QUEVEDO REVILLA		05/05/2022				
VERIF.	FERNANDO QUEVEDO REVILLA		07/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA			N.º DE DIBUJO A4
				PESO:			ESCALA:1:50 HOJA 1 DE 1



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.	Fernando Quevedo Revilla		05/05/2022
VERIF.	Fernando Quevedo Revilla		07/05/2022
APROB.			
FABR.			
CALID.			

TÍTULO:

Vista en conjunto explosionada

MATERIAL:

PLA

N.º DE DIBUJO

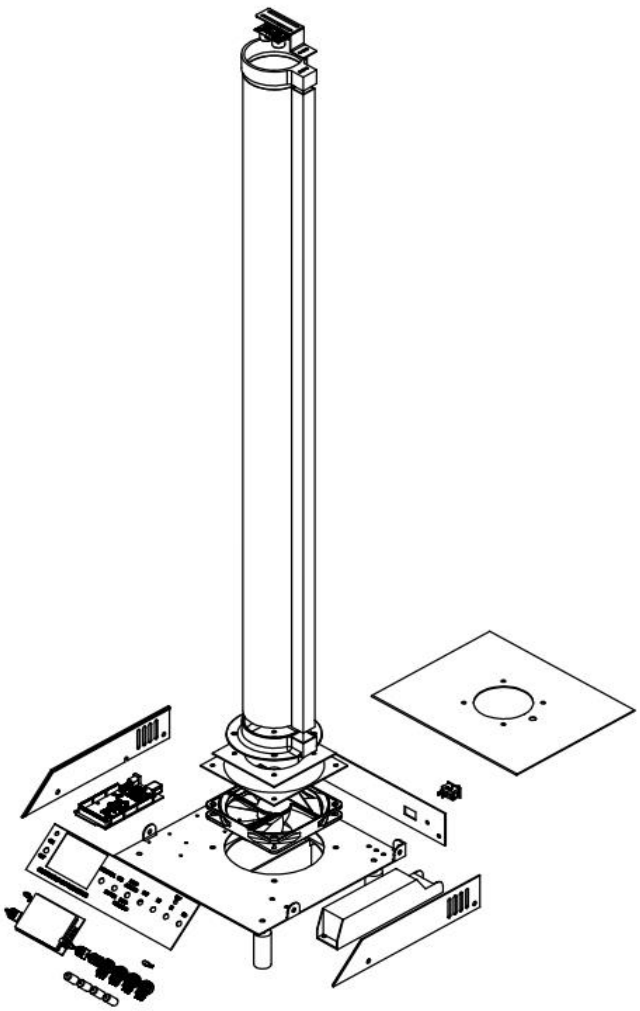
VISTA_EXPLOSIONADA

A4

PESO:

ESCALA:1:50

HOJA 1 DE 1



SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA / REVISIÓN

	NOMBRE	FIRMA	FECHA
DIBUJ.	Fernando Quevedo Revilla		05/05/2022
VERIF.	Fernando Quevedo Revilla		07/05/2022
APROB.			
FABR.			
CALID.			

TÍTULO:

Vista en conjunto explosionada

MATERIAL:

PLA

N.º DE DIBUJO

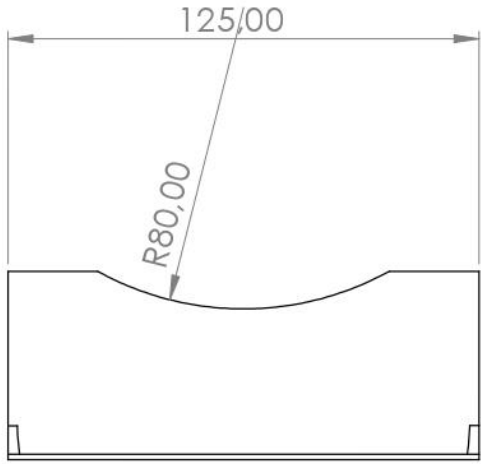
VISTA_EXPLOSIONADA

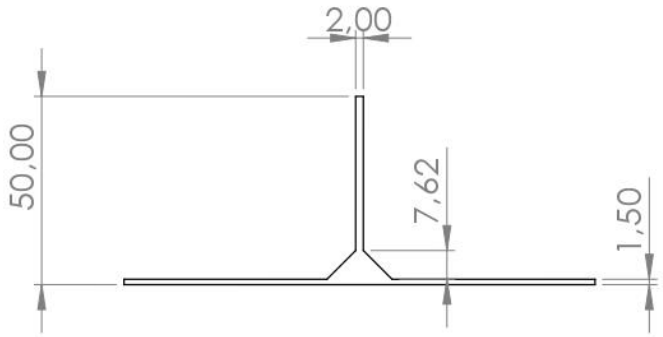
A4

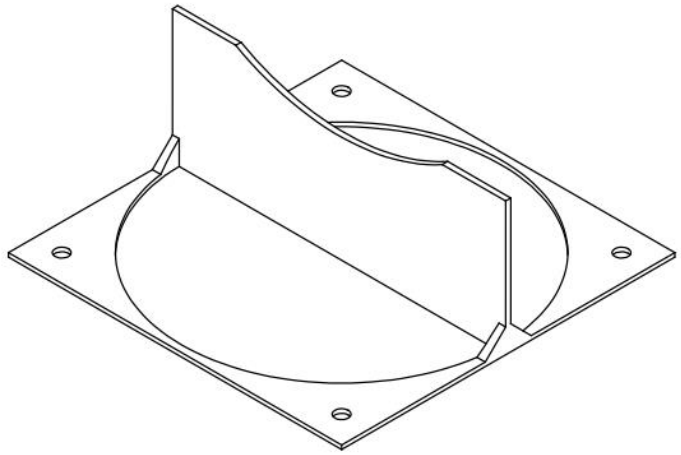
PESO:

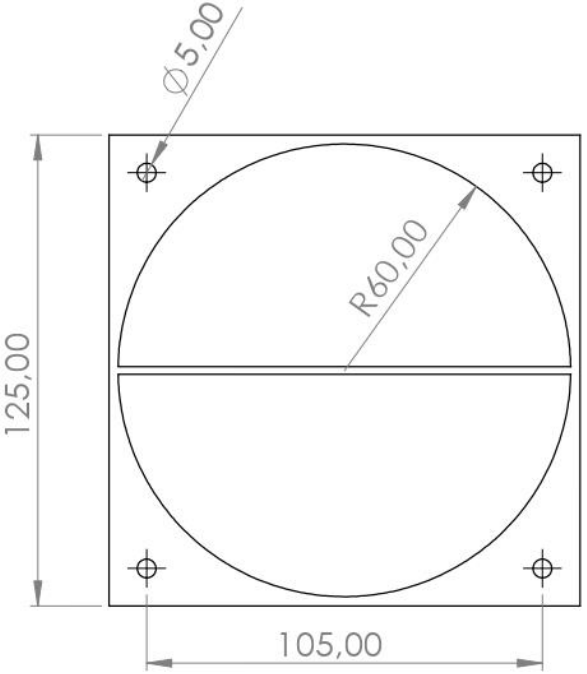
ESCALA:1:50

HOJA 1 DE 1

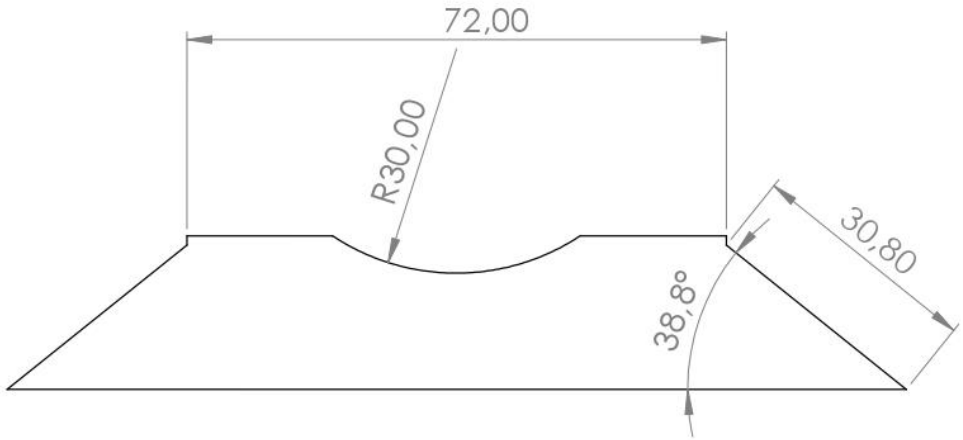


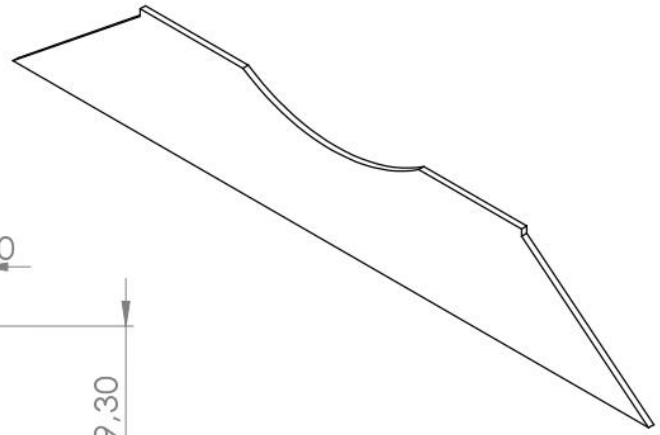


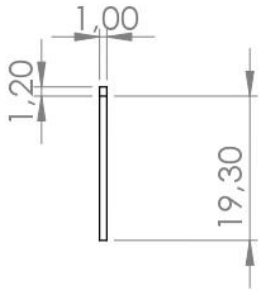


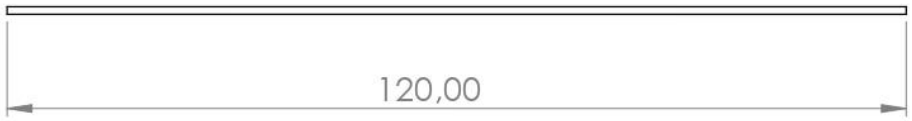


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA		17/05/2022			SUPLEMENTO 2	
VERIF.	FERNANDO QUEVEDO REVILLA		19/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL:		N.º DE DIBUJO	A4
				PLA		SUPLEMENTO 2	
				PESO:		ESCALA:1:2	HOJA 1 DE 1

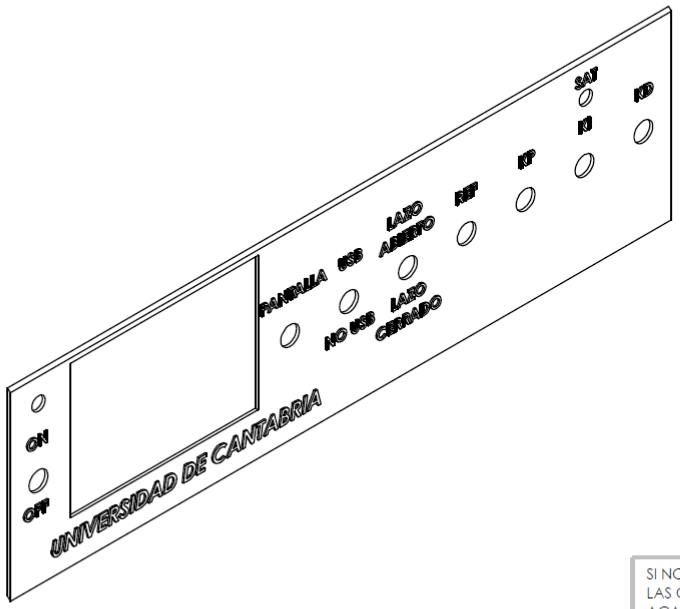


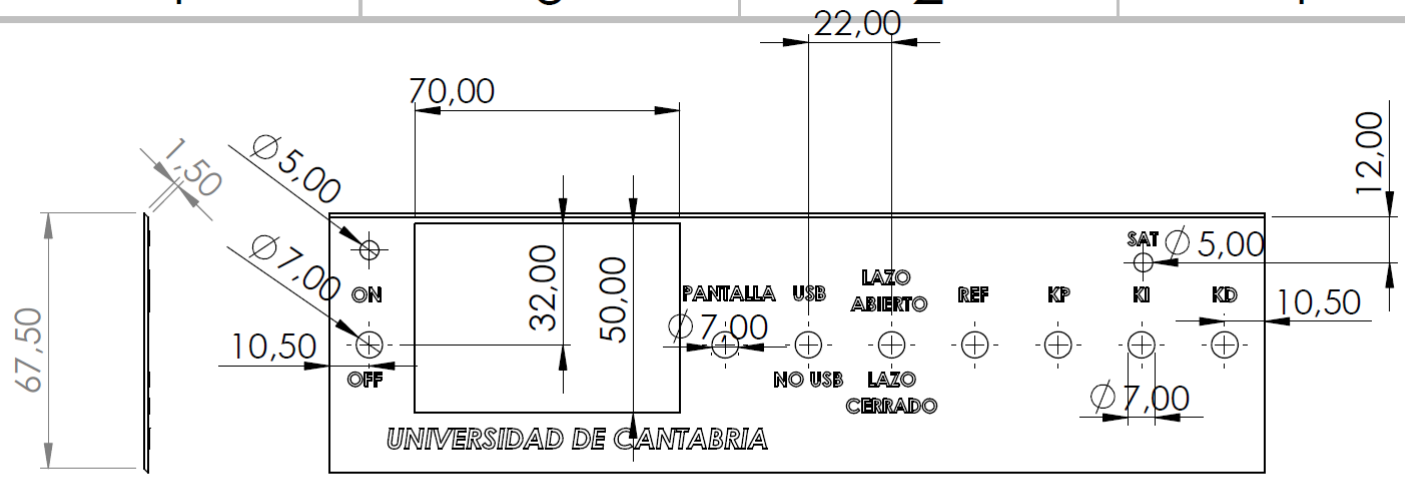


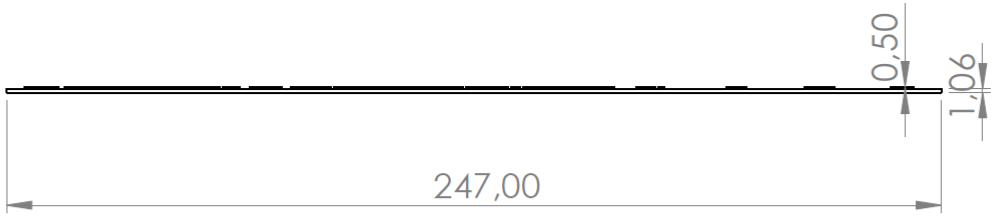




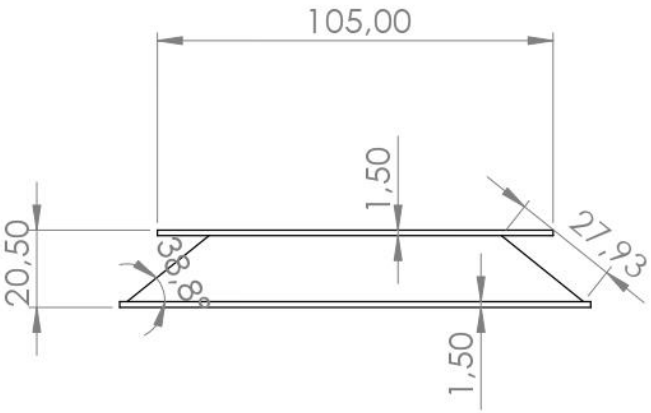
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:				REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA				TÍTULO: SUPLEMENTO 1	
DIBUJ.	FERNANDO QUEVEDO REVILLA		15/05/2022					
VERIF.	FERNANDO QUEVEDO REVILLA		20/05/2022					
APROB.								
FABR.								
CALID.				MATERIAL: PLA			N.º DE DIBUJO SUPLEMENTO_1	A4
				PESO:			ESCALA:1:1	HOJA 1 DE 1

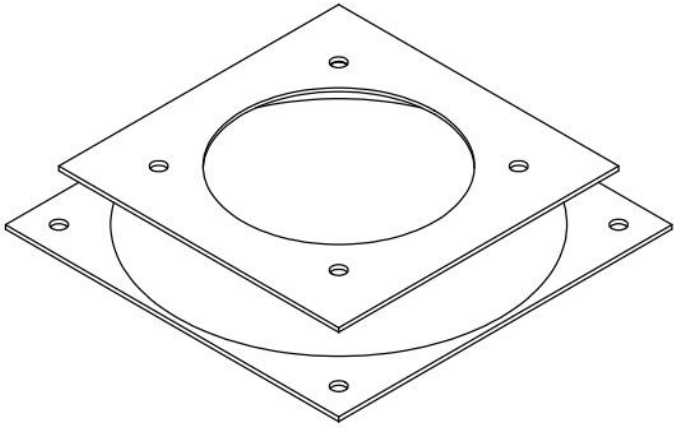


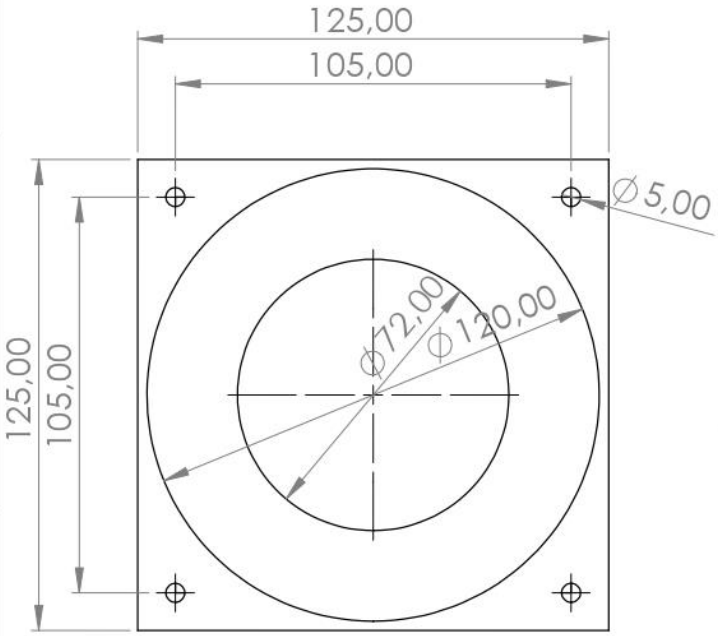




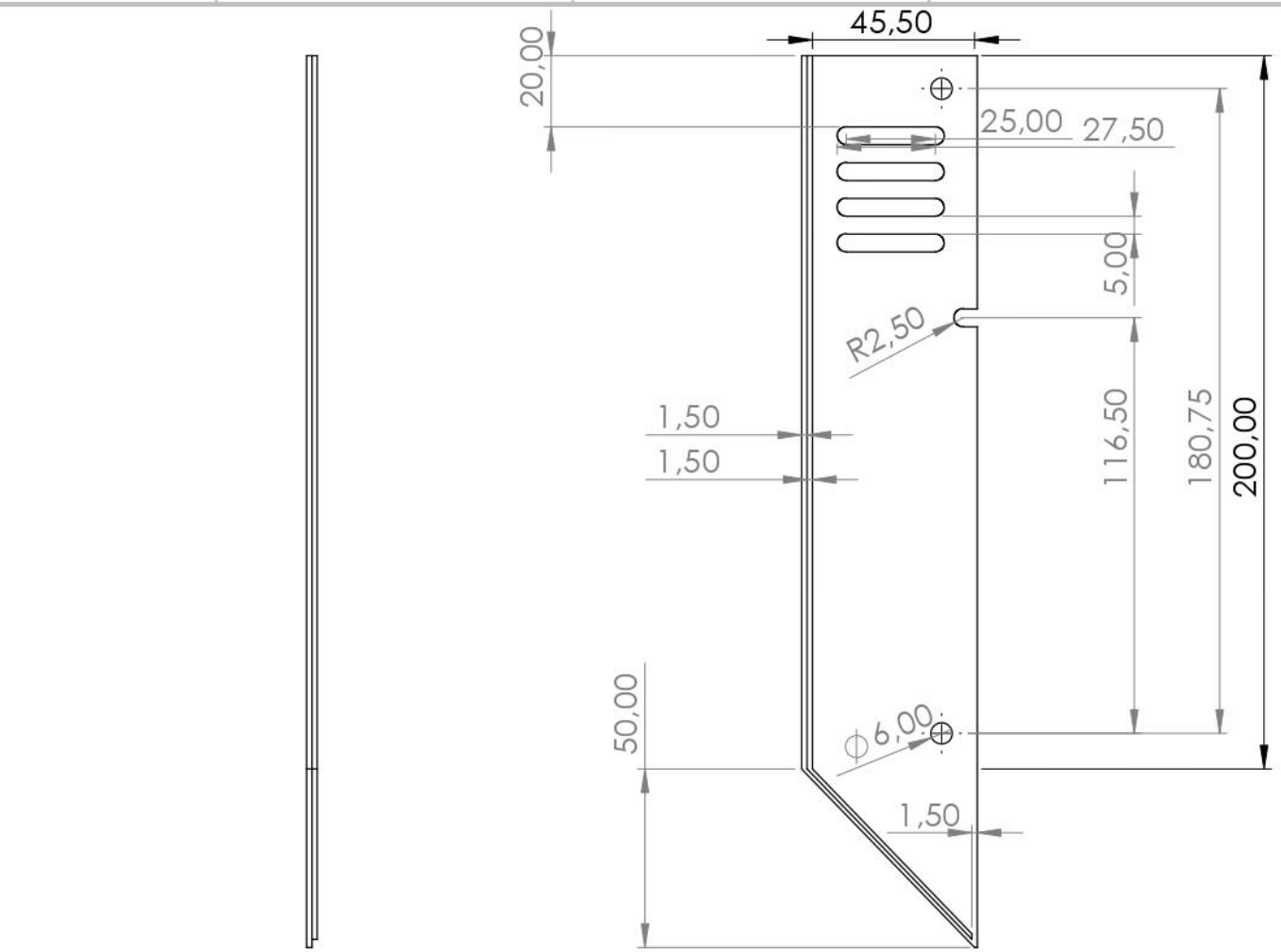
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:	ACABADO:	REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN		
	NOMBRE	FIRMA	FECHA			TÍTULO:
DIBUJ.	FERNANDO QUEVEDO REVILLA		03/05/2022			CONSOLA
VERIF.	FERNANDO QUEVEDO REVILLA		08/05/2022			
APROB.						
FABR.						
CALID.				MATERIAL: PLA	N.º DE DIBUJO CONSOLA	A4
				PESO:	ESCALA:1:2	HOJA 1 DE 1

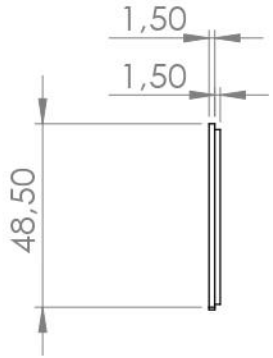




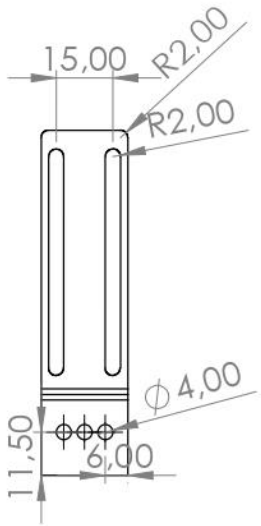


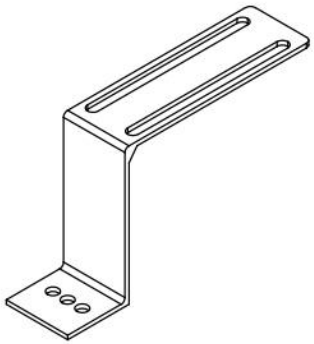
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN	
	NOMBRE	FIRMA	FECHA		TÍTULO:		
DIBUJ.	FERNANDO QUEVEDO REVILLA		09/05/2022		EMBUDO		
VERIF.	FERNANDO QUEVEDO REVILLA		15/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA	N.º DE DIBUJO EMBUDO		A4
				PESO:	ESCALA:1:2	HOJA 1 DE 1	

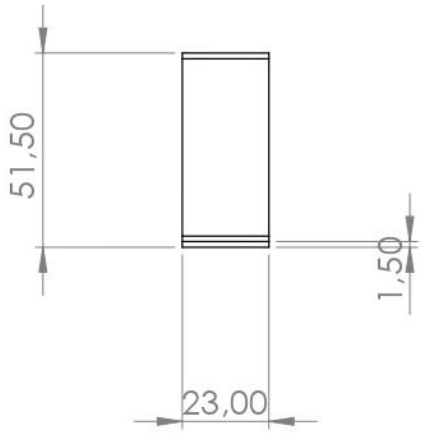


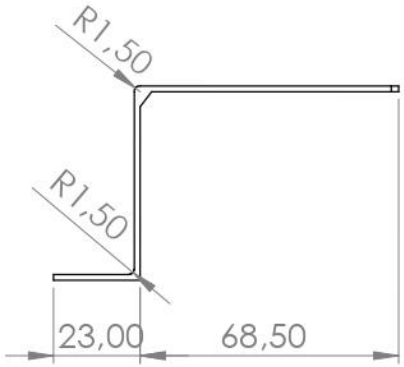


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA		07/05/2022			LATERAL IZQUIERDO	
VERIF.	FERNANDO QUEVEDO REVILLA		08/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO: LATERAL_IZDO	A4
				PESO:		ESCALA:1:5	HOJA 1 DE 1

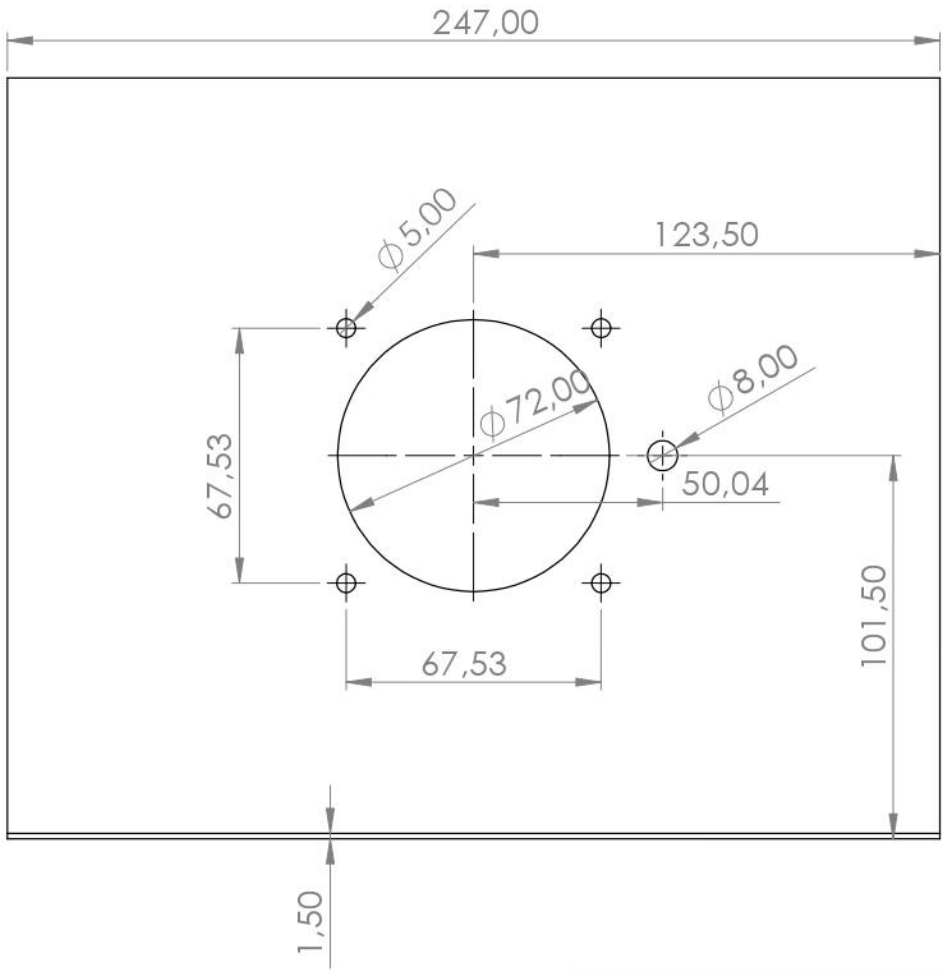


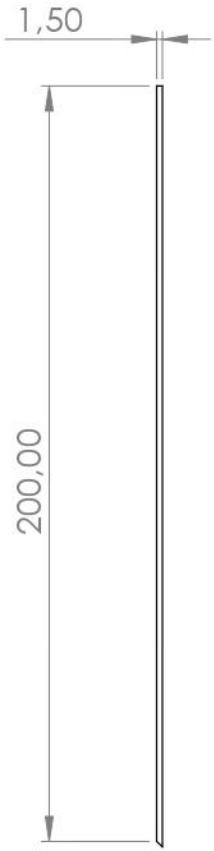




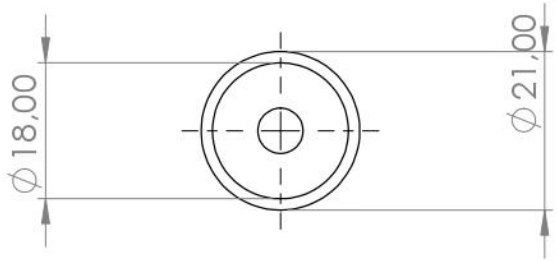


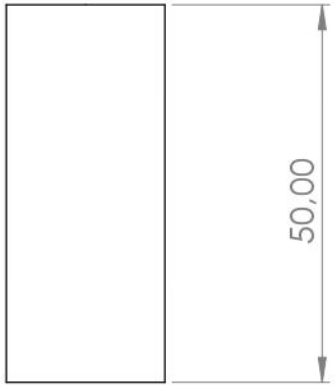
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO: SOPORTE SENSOR	
DIBUJ.	FERNANDO QUEVEDO REVILLA		08/15/2022				
VERIF.	FERNANDO QUEVEDO REVILLA		09/15/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO SOPORTE_SENSOR	A4
				PESO:		ESCALA:1:1	HOJA 1 DE 1

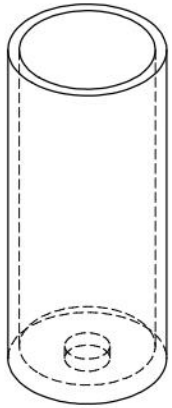


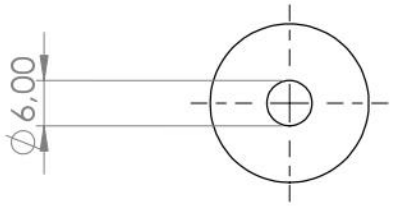


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO: TAPA SUPERIOR	
DIBUJ.	FERNANDO QUEVEDO REVILLA		04/05/2022				
VERIF.	FERNANDO QUEVEDO REVILLA		06/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO TAPA_SUPERIOR	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1

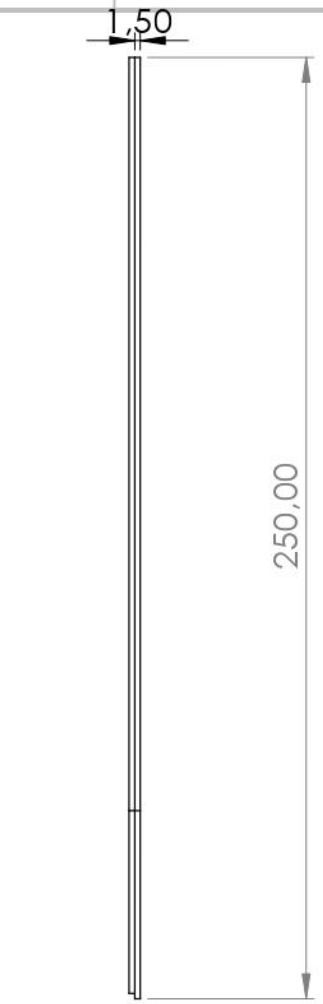


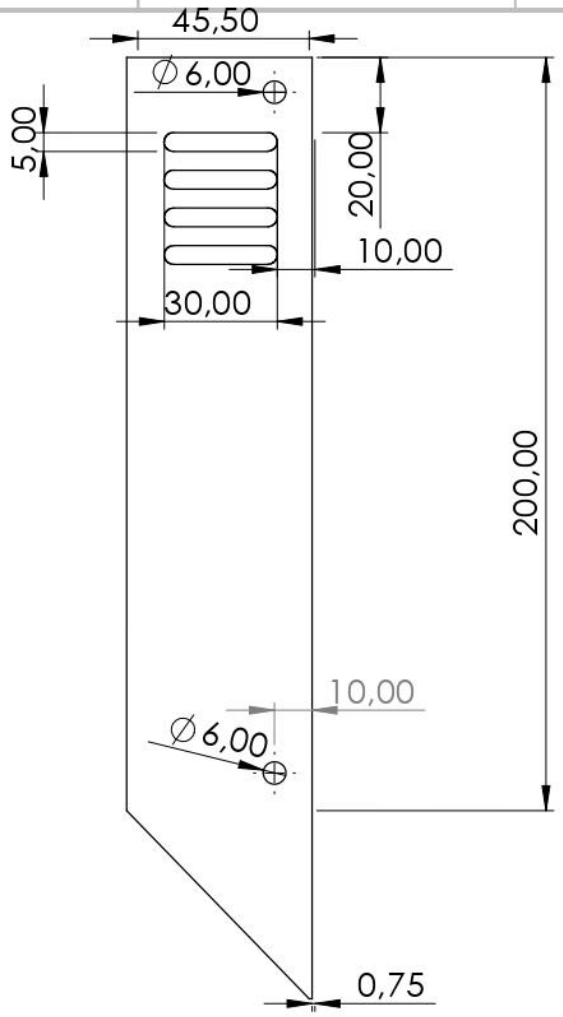


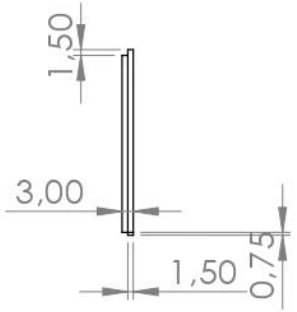




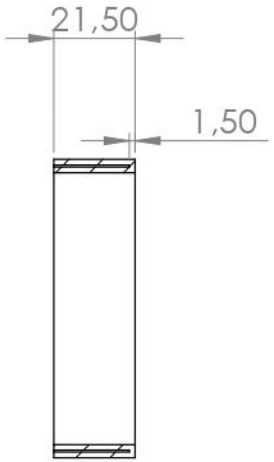
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA			06/05/2022		PATA	
VERIF.	FERNANDO QUEVEDO REVILLA			10/05/2022			
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO Pata	A4
				PESO:		ESCALA:1:1	HOJA 1 DE 1



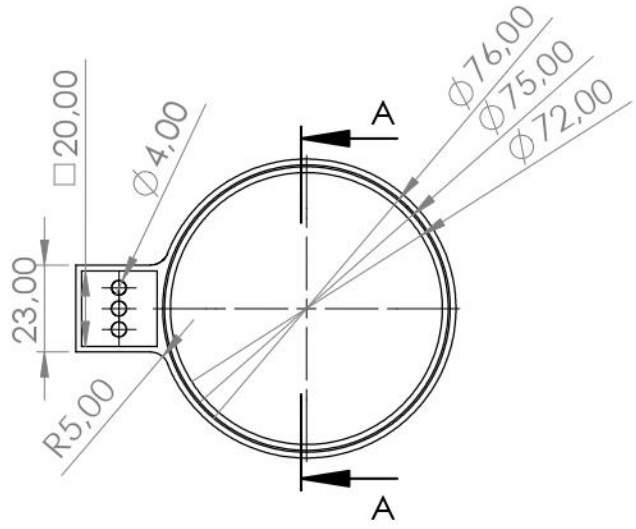


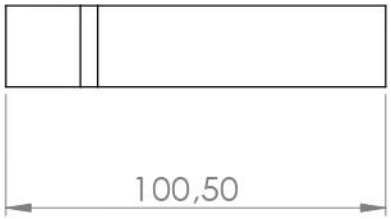


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA		05/05/2022			LATERAL DERECHO	
VERIF.	FERNANDO QUEVEDO REVILLA		06/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO LATERAL_DCHO	A4
				PESO:		ESCALA:1:5	HOJA 1 DE 1

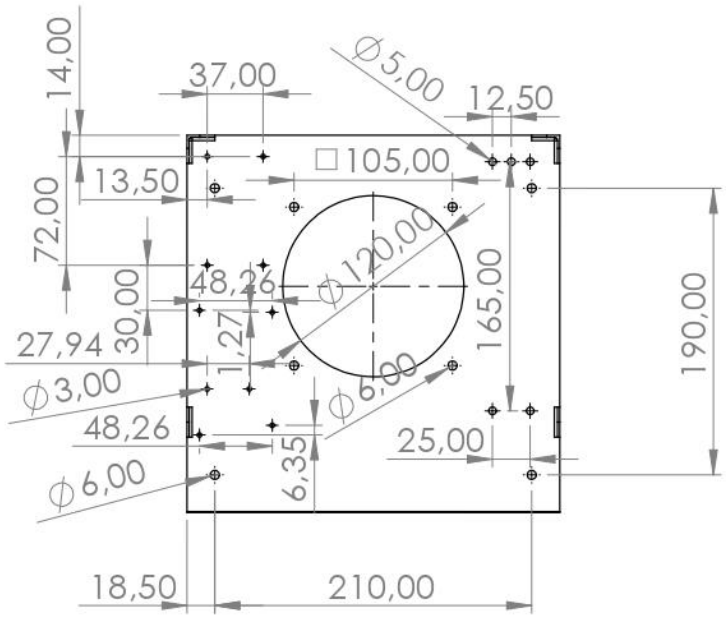


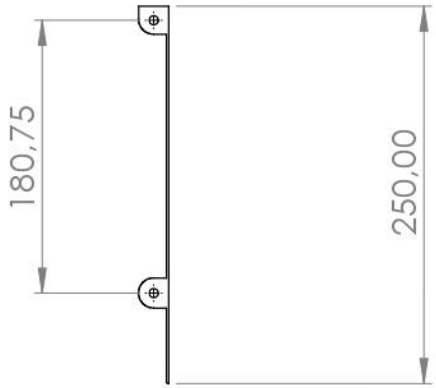
SECCIÓN A-A

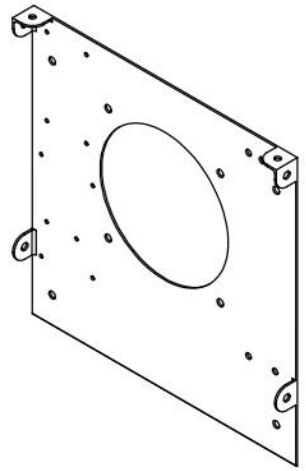


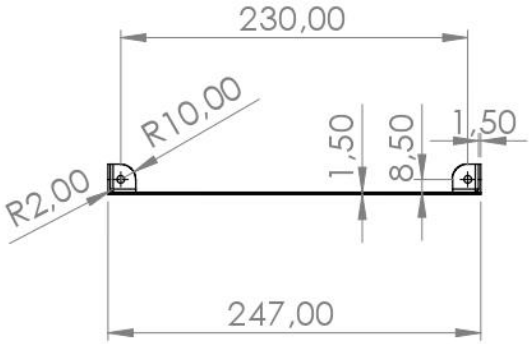


SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO: SOPORTE LÁMINA SUPERIOR	
DIBUJ.	FERNANDO QUEVEDO REVILLA		15/05/2022				
VERIF.	FERNANDO QUEVEDO REVILLA		16/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO SOPORTE_SUP	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1

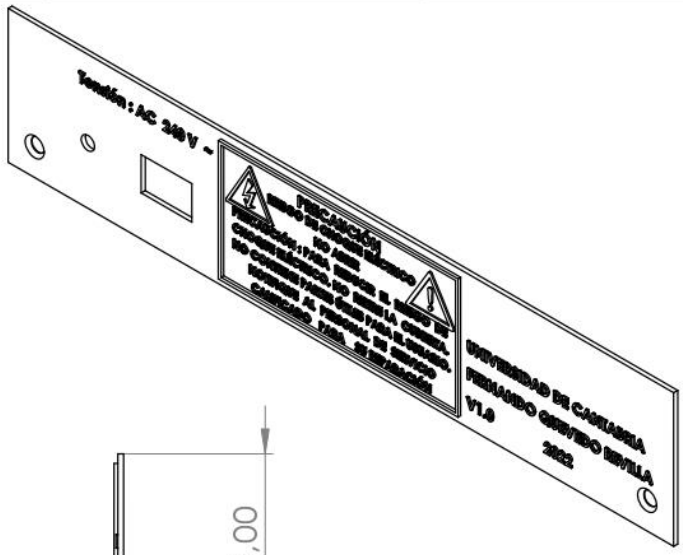


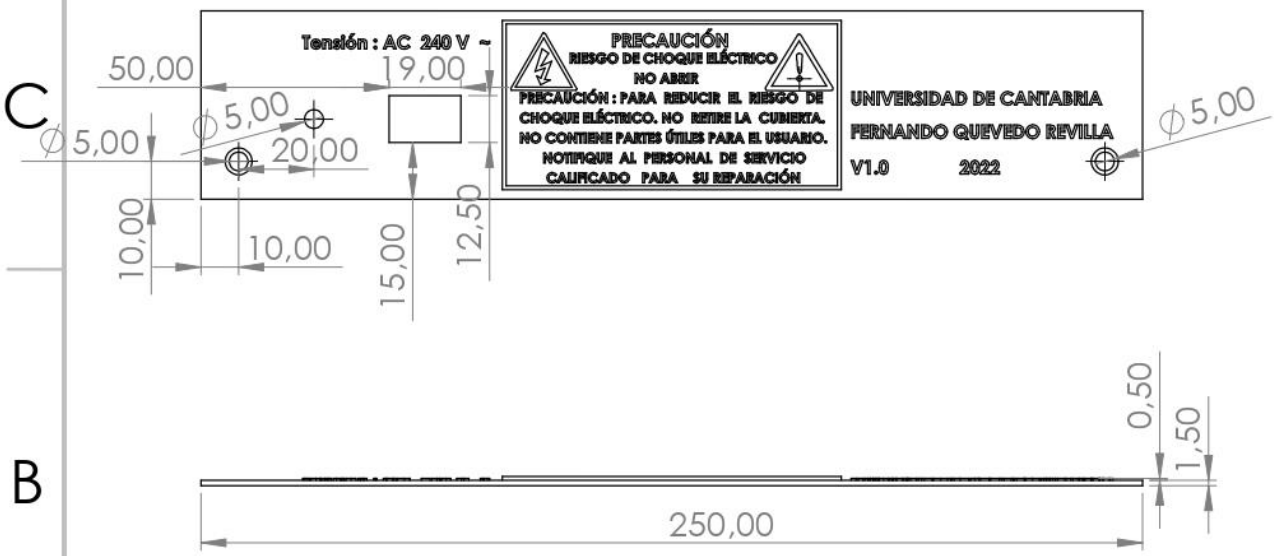




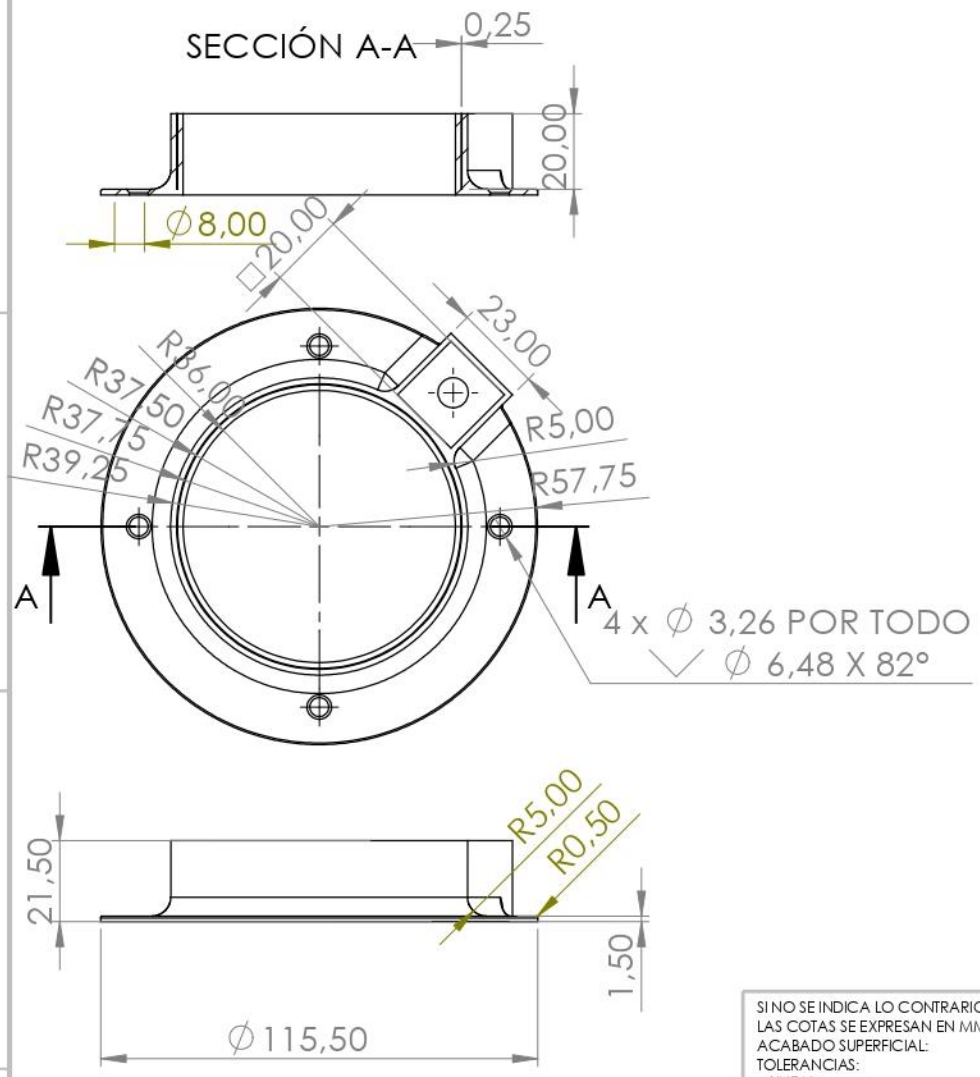


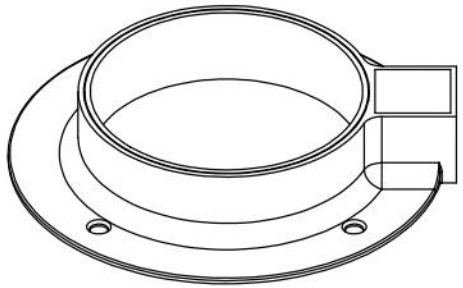
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:		REBARBAR Y ROMPER ARISTAS VIVAS		NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA		01/05/2022			BASE PRINCIPAL	
VERIF.	FERNANDO QUEVEDO REVILLA		03/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO BASE	A4
				PESO:		ESCALA:1:5	HOJA 1 DE 1

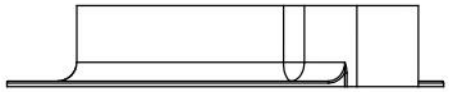




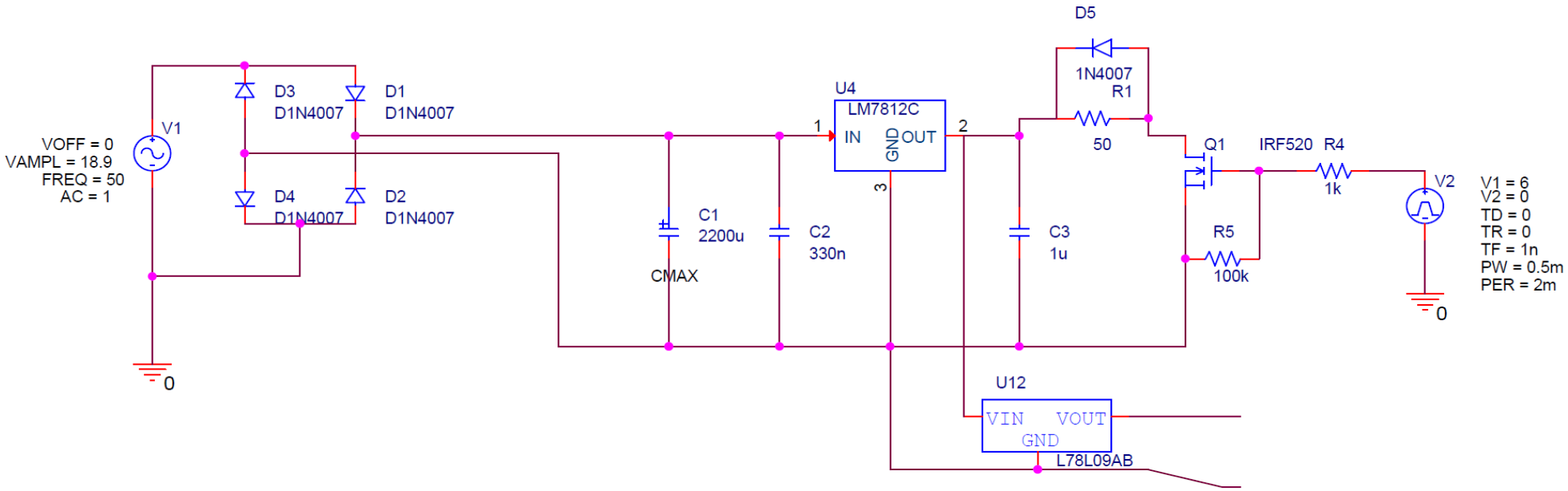
SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA		02/05/2022			TAPA TRASERA	
VERIF.	FERNANDO QUEVEDO REVILLA		05/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO: TRASERA	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1







SI NO SE INDICA LO CONTRARIO: LAS COTAS SE EXPRESAN EN MM ACABADO SUPERFICIAL: TOLERANCIAS: LINEAL: ANGULAR:		ACABADO:			REBARBAR Y ROMPER ARISTAS VIVAS	NO CAMBIE LA ESCALA	REVISIÓN
	NOMBRE	FIRMA	FECHA			TÍTULO:	
DIBUJ.	FERNANDO QUEVEDO REVILLA		15/05/2022			SOPORTE LÁMINA INFERIOR	
VERIF.	FERNANDO QUEVEDO REVILLA		17/05/2022				
APROB.							
FABR.							
CALID.				MATERIAL: PLA		N.º DE DIBUJO Soporte_lamina	A4
				PESO:		ESCALA:1:2	HOJA 1 DE 1



Title	ELECTRÓNICA DE POTENICA; LEVITADOR NEUMÁTICO	
Size A	Document Number DISEÑADO POR: FERNANDO QUEVEDO REVILLA	Rev 1.3
Date: Tuesday, June 28, 2022	Sheet 1 of 1	

ANEXO 3: CÓDIGO ARDUINO

```
//PROGRAMA ARDUINO DE CONTROL DE UN LEVITADOR NEUMÁTICO
//AUTOR:FERNANDO QUEVEDO REVILLA
//UNIVERSIDAD DE CANTABRIA
//FECHA: 30/05/2020
//VERSIÓN: 3.3


//Librerias utiizadas
#include <Adafruit_GFX.h>    // Core graphics library
#include <Adafruit_TFTLCD.h> // Hardware-specific library
#include <Fonts/FreeSansBoldOblique9pt7b.h>
#include <Fonts/FreeSerifItalic9pt7b.h>
#include <TimerOne.h>

//Constantes hardware
#define trigger 8           // Trigger pin for ultrasonic sensor
#define echo 9              // echo pin for ultrasonic sensor
#define v_pin 7             // Control signal PWM
#define ref_pot A8          // Analog input for  pot
#define kp_pot A9           // Entrada analógica
#define ki_pot A10          // Entrada analógica
#define kd_pot A11          // Entrada analógica
#define power_on_led 30     // Led alimentacion Verde
#define sat_led      31     // Led Saturacion PID Rojo
#define pulsador      3     // Pulsador cambio de modo pantalla
#define interruptor_USB 5   // Interruptor modo USB
#define interruptor_REALIMENTACION  6 // Interruptor lazo cerrado o
abierto
//Hardware pins para display TFT
#define LCD_CS A3 // Chip Select goes to Analog 3
#define LCD_CD A2 // Command/Data goes to Analog 2
#define LCD_WR A1 // LCD Write goes to Analog 1
#define LCD_RD A0 // LCD Read goes to Analog 0
#define LCD_RESET A4 // TFT reset
```

```
// Constantes relativas a software
// Universidad de Cantabria logo
#define LOGO_WIDTH 74    // Anchura del logo UC
#define LOGO_HEIGHT 48   // Longitud del logo UC
const unsigned char PROGMEM UC[]={
  0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff,
0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xc0, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xc0, 0x7f, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x7f, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,
  0xff, 0xc0, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
0x7e, 0x00, 0x3f, 0xe0,
  0x03, 0xff, 0xf8, 0x00, 0x3f, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0xff,
0xc0, 0x00, 0x0f, 0xc0,
  0x7c, 0x00, 0x3f, 0xe0, 0x03, 0xff, 0x80, 0x00, 0x07, 0xc0, 0x7c, 0x00,
0x3f, 0xe0, 0x03, 0xfe,
  0x00, 0x00, 0x07, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0xfc, 0x00, 0x00,
0x07, 0xc0, 0x7c, 0x00,
  0x3f, 0xe0, 0x03, 0xf8, 0x00, 0x00, 0x07, 0xc0, 0x7c, 0x00, 0x3f, 0xe0,
0x03, 0xf0, 0x00, 0x00,
  0x07, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0xe0, 0x00, 0x00, 0x07, 0xc0,
0x7c, 0x00, 0x3f, 0xe0,
  0x03, 0xc0, 0x00, 0x00, 0x07, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0xc0,
0x00, 0x00, 0x07, 0xc0,
  0x7c, 0x00, 0x3f, 0xe0, 0x03, 0x80, 0x00, 0x3f, 0x07, 0xc0, 0x7c, 0x00,
0x3f, 0xe0, 0x03, 0x80,
  0x00, 0xff, 0xe7, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0x80, 0x03, 0xff,
0xff, 0xc0, 0x7c, 0x00,
  0x3f, 0xe0, 0x03, 0x00, 0x07, 0xff, 0xff, 0xc0, 0x7c, 0x00, 0x3f, 0xe0,
0x03, 0x00, 0x07, 0xff,
  0xff, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0x00, 0x0f, 0xff, 0xff, 0xc0,
0x7c, 0x00, 0x3f, 0xe0,
  0x03, 0x00, 0x0f, 0xff, 0xff, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0x00,
0x0f, 0xff, 0xff, 0xc0,
  0x7c, 0x00, 0x3f, 0xe0, 0x03, 0x00, 0x0f, 0xff, 0xff, 0xc0, 0x7c, 0x00,
0x3f, 0xe0, 0x03, 0x00,
  0x0f, 0xff, 0xff, 0xc0, 0x7c, 0x00, 0x3f, 0xe0, 0x03, 0x00, 0x0f, 0xff,
0xff, 0xc0, 0x7c, 0x00,
  0x3f, 0xe0, 0x03, 0x00, 0x07, 0xff, 0xff, 0xc0, 0x7c, 0x00, 0x3f, 0xe0,
0x03, 0x00, 0x07, 0xff,
  0xff, 0xc0, 0x7c, 0x00, 0x3f, 0xc0, 0x03, 0x80, 0x03, 0xff, 0xff, 0xc0,
0x7c, 0x00, 0x1f, 0xc0,
  0x03, 0x80, 0x01, 0xff, 0xf7, 0xc0, 0x7c, 0x00, 0x0f, 0x00, 0x03, 0x80,
0x00, 0x7f, 0xc7, 0xc0,
```

```
  0x7e, 0x00, 0x00, 0x00, 0x03, 0xc0, 0x00, 0x00, 0x07, 0xc0, 0x7e, 0x00,
0x00, 0x00, 0x07, 0xc0,

  0x00, 0x00, 0x07, 0xc0, 0x7e, 0x00, 0x00, 0x00, 0x07, 0xe0, 0x00, 0x00,
0x07, 0xc0, 0x7f, 0x00,

  0x00, 0x00, 0x0f, 0xf0, 0x00, 0x00, 0x07, 0xc0, 0x7f, 0x80, 0x00, 0x00,
0x0f, 0xf0, 0x00, 0x00,

  0x07, 0xc0, 0x7f, 0x80, 0x00, 0x00, 0x1f, 0xf8, 0x00, 0x00, 0x07, 0xc0,
0x7f, 0xc0, 0x00, 0x00,

  0x3f, 0xfe, 0x00, 0x00, 0x07, 0xc0, 0x7f, 0xf0, 0x00, 0x00, 0x7f, 0xff,
0x00, 0x00, 0x07, 0xc0,

  0x7f, 0xf8, 0x00, 0x01, 0xff, 0xff, 0xc0, 0x00, 0x0f, 0xc0, 0x7f, 0xff,
0x00, 0x07, 0xff, 0xff,

  0xf8, 0x00, 0x3f, 0xc0, 0x7f, 0xff, 0xf8, 0xff, 0xff, 0xff, 0xff, 0x83,
0xff, 0xc0, 0x7f, 0xff,

  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x7f, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

  0xff, 0xc0, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0,
0x7f, 0xff, 0xff, 0xff,

  0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00

  };



// Expresion PID, 168x47px

const unsigned char PROGMEM PID[]=  {

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x01, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x02,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07,
0x00, 0x00, 0x00, 0x00,
```

```
0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x03, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3e, 0x1e,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x7e,

0x36, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0xc6, 0x63, 0x00, 0x00, 0x00, 0x01, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xc6, 0x6e, 0x00,
0x00, 0x00, 0x01, 0x60,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xc6, 0xfc,

0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x20, 0x18, 0x00, 0x00, 0x00, 0x00,
0x7b, 0xc0, 0x00, 0x00,

0x07, 0x9c, 0x00, 0xcc, 0xc4, 0x00, 0x07, 0x9c, 0x03, 0x00, 0x00, 0x60,
0x08, 0x03, 0x80, 0x00,

0x00, 0x7b, 0xc0, 0x00, 0x00, 0x07, 0x9c, 0x00, 0xfc, 0x6c, 0x00, 0x07,
0xbc, 0x03, 0x00, 0x00,

0x43, 0x0c, 0x01, 0x98, 0x00, 0x00, 0x36, 0x00, 0x00, 0x00, 0x03, 0x30,
0x00, 0x60, 0x38, 0x00,

0x03, 0x70, 0x03, 0x00, 0x00, 0x43, 0x04, 0x01, 0x98, 0x00, 0x00, 0x6c,
0x00, 0x0f, 0x06, 0x03,

0x60, 0x00, 0x00, 0x00, 0x06, 0x02, 0xc0, 0x03, 0x00, 0x78, 0x47, 0xc4,
0x1f, 0xbe, 0x00, 0x00,

0x78, 0x00, 0x19, 0x86, 0x03, 0x80, 0x00, 0x00, 0x00, 0x06, 0x07, 0x80,
0x03, 0x00, 0xcc, 0xc3,

0x04, 0x33, 0x18, 0x00, 0x00, 0x78, 0x00, 0x31, 0x9f, 0x83, 0x80, 0x00,
0x00, 0x00, 0x1f, 0x87,

0x80, 0x03, 0x01, 0x8c, 0xc6, 0x04, 0x33, 0x10, 0x00, 0x00, 0x7c, 0x00,
0x3f, 0x1f, 0xc7, 0xc0,

0x01, 0xff, 0xff, 0x9f, 0x87, 0xc0, 0x03, 0x01, 0xf8, 0xc6, 0x04, 0x63,
0x30, 0x00, 0x00, 0x6c,

0x00, 0x3c, 0x06, 0x06, 0xe0, 0x30, 0x00, 0x00, 0x06, 0x06, 0xc0, 0x83,
0x01, 0xe0, 0xc6, 0x04,

0x63, 0x30, 0x00, 0x00, 0x4e, 0x00, 0x31, 0x06, 0x06, 0x70, 0x10, 0x00,
0x00, 0x06, 0x06, 0x60,

0x83, 0x01, 0x98, 0x46, 0x84, 0x67, 0x34, 0x00, 0x01, 0xe7, 0x9e, 0x3f,
0x00, 0x0f, 0x38, 0xf0,

0x03, 0x80, 0x00, 0x1f, 0x79, 0x83, 0x01, 0xf0, 0x47, 0x84, 0x3e, 0x3c,
0x00, 0x00, 0x00, 0x1b,
```

```
0x00, 0x00, 0x00, 0x01, 0xb0, 0x01, 0x98, 0x00, 0x00, 0x01, 0x83, 0x00,
0x00, 0x40, 0x04, 0x00,

0x00, 0x00, 0x00, 0x00, 0x13, 0x00, 0x00, 0x00, 0x01, 0x20, 0x03, 0x10,
0x00, 0x00, 0x01, 0x03,

0x00, 0x00, 0x60, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x16, 0x00, 0x00,
0x00, 0x01, 0xe0, 0x1f,

0x7c, 0x00, 0x00, 0x01, 0x0f, 0x00, 0x00, 0x20, 0x18, 0x00, 0x00, 0x00,
0x00, 0x00, 0x1c, 0x00,

0x00, 0x00, 0x00, 0xe0, 0x3f, 0x7c, 0x00, 0x00, 0x01, 0x8f, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x63, 0x30, 0x00,
0x00, 0x00, 0x0c, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x62, 0x30,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x62, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x66, 0x6c, 0x00, 0x00,
0x00, 0x02, 0x80, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x7e, 0x6c, 0x00,

0x00, 0x00, 0x06, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x06, 0xc0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x06, 0xc0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x06, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff};




// Icono ordenador modo USB 250x150px

const unsigned char PROGMEM USB[]={

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xfe,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff,
0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff,
0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff,
0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xe0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x03, 0xff,
0xe0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x01, 0xff,
0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00, 0xff,
0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x7f, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff,
0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x03, 0xff,

0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff,
0xc0, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xff, 0xff,

0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff,
0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0xff, 0xff,

0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff,
0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0xfc, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff,
0x80, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0xe0, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0xc0, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0xc0, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x78, 0x70, 0x00, 0x00, 0x00, 0x00,
0x00, 0x03, 0x80, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x07, 0x80, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x07, 0x80, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x07, 0x80, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0f, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0f, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0f, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0e, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1e, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xfc,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xef, 0xff, 0xde, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1e, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x3e, 0x00, 0x00, 0x00, 0x00,
0x00, 0x1e, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x30, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3c, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3c, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x06, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3c, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x06, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x38, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x04, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x78, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf0, 0x10, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x78, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xf1, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x78, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x20, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0x70, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0xf0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xe0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x30, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0xf0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xc0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x40, 0x3c, 0x00, 0x00, 0x00, 0x00,
0x00, 0xf0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xc0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xf0, 0x00, 0x7c, 0x00, 0x00, 0x00, 0x00,
0x01, 0xe0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xc0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xef, 0xff, 0xdc, 0x00, 0x00, 0x00, 0x00,
0x01, 0xe0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xc0,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00,
0x01, 0xe0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0x80,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00,
0x01, 0xc0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0x80,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x03, 0xc0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0x80,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x03, 0xc0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x03, 0xc0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x07, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x07, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x07, 0x80, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x07, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x00,
0x0f, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x0f, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x0f, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x1e, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x1e, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xfc, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x1e, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00,
0x1f, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x3f, 0xe0, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x3f, 0xff, 0xc0, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf8, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x3f, 0xff, 0xff, 0x80,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x3f, 0xff, 0xff, 0xff,

0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xf0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00,
0x7f, 0xff, 0xff, 0xff,

0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xf0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x03,
0xff, 0xff, 0xff, 0xff,

0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xf0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x00, 0x0f, 0xfd,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xfe, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x00, 0x1f, 0xf8, 0x61,
0x87, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf8, 0x00, 0x7f, 0xe0, 0x18, 0x19,
0x30, 0x3f, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x00, 0xff, 0xe0, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xf9, 0xff, 0xc0, 0x04, 0x30, 0xc1,
0x8f, 0x00, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0xff, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xe0, 0x02, 0x01, 0xe3, 0x06,
0x0c, 0x78, 0x03, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x00, 0x01, 0xff, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xfe, 0x38, 0x20, 0x0e, 0x08,
0x30, 0x61, 0x40, 0x1f,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xc0, 0x03, 0xff, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0xff, 0xff, 0xe0, 0x0f, 0x00, 0x30,
0x61, 0x81, 0x8d, 0x80,

0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xdf, 0xff, 0xc0, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x30, 0x01,
0x80, 0x8c, 0x03, 0x63,

0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x01, 0xc0,
0x08, 0x01, 0x60, 0x07,

0x06, 0x0f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x19,
0xc0, 0x40, 0x03, 0x80,

0x0c, 0x03, 0x7f, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0x1f, 0xe0, 0x00,
0x80, 0x86, 0xc0, 0x19,

0x80, 0x60, 0x01, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x07, 0xff, 0xff, 0xff, 0xff, 0x00, 0x01, 0xfe, 0x00,
0x08, 0x00, 0xe0, 0x40,

0xc0, 0xc3, 0x60, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff, 0x80, 0x00, 0x00, 0x1f, 0xe0,
0x00, 0xf0, 0x01, 0x00,

0x36, 0x00, 0x3c, 0x18, 0x3f, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xf8, 0x00, 0x00, 0x00, 0x03, 0xfe,
0x00, 0x04, 0x18, 0x1c,

0x00, 0x34, 0x00, 0x60, 0x03, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x3f,
0xe0, 0x00, 0x40, 0x03,
```

```
0x81, 0x01, 0x80, 0x83, 0x60, 0x07, 0xff, 0xff, 0xff, 0xfe, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xe0, 0x00, 0x00, 0x00, 0x03,
0xfe, 0x00, 0x06, 0xc0,

0x04, 0x00, 0x3c, 0x00, 0x18, 0x04, 0x1f, 0xff, 0xff, 0xfe, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xfc, 0x00, 0x00, 0x00, 0x00,
0x3f, 0xe0, 0x00, 0x20,

0x18, 0x6c, 0x00, 0x63, 0x00, 0x60, 0x00, 0x7f, 0xff, 0xfe, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0x80, 0x00, 0x00, 0x00,
0x7f, 0xfe, 0x06, 0x03,

0x00, 0x02, 0x00, 0x63, 0x00, 0x13, 0x00, 0x03, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xf8, 0x00, 0x00, 0x1f,
0xff, 0xff, 0xe0, 0x00,

0x30, 0x38, 0x3c, 0x00, 0x1b, 0x00, 0x18, 0x03, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f, 0xff, 0xff, 0x00, 0x07, 0xff,
0xff, 0xff, 0xfe, 0x0c,

0x01, 0x00, 0x01, 0x80, 0x00, 0xc0, 0x00, 0xff, 0xff, 0xfc, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0xe1, 0xff, 0xff,
0xff, 0xff, 0xff, 0xe0,

0x00, 0x10, 0x00, 0x0c, 0x00, 0x06, 0x0f, 0xff, 0xff, 0xf8, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0x00, 0x71, 0x80, 0x00, 0xe0, 0x00, 0x7f, 0xff, 0xff, 0xf0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xe6, 0x00, 0x18, 0x00, 0x06, 0x07, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0x00, 0x0c, 0x80, 0x00, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7f, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xf1, 0xc0, 0x0c, 0x0f, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0f, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0x00, 0x00, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xf0, 0x1f, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3f, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1f,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03,
0xff, 0xff, 0xff, 0xff,

0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x7f, 0xff, 0xff, 0xff,

0xff, 0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x1f, 0xff, 0xff, 0xff,

0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x03, 0xff, 0xff, 0xff,
```

```
0xff, 0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x7f, 0xff, 0xff,

0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0f, 0xff, 0xff,

0xf0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x01, 0xff, 0xff,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x1f, 0xf0,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00};

// Definir colores básicos
#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

//Variables globales
float tiempo,distancia,e_act,e_prev=0,de,vp,vi,vd,error_suma=0,v;
float ref=0,kp=0,ki=0,kd=0;
float v_volts; // Tensión del actuador en voltios
//Variales globales de pantalla
int modo=3,modo_max=3,modo_init=1,modo_anterior=2; // Administrar modos
int M1[275],M2[275];                                // Gráfico
int blink_signal;                                   // Parapadeo

// Variables interruptores y pulsador
bool USB_POT=0,LA_LC=0;

//Variables interrupciones y cálculo del periodo de muestreo
```

```
volatile unsigned int interrupt = 0;
unsigned long tsamp = 10*1000;
float ts = 0;
float other=0;
unsigned long t = 0;
unsigned long t0 = 0;


// Instancia de la pantalla
Adafruit_TFTLCD tft(LCD_CS, LCD_CD, LCD_WR, LCD_RD, LCD_RESET);



void setup(void) {
  //Configuracion de los pines
  pinMode(trigger,OUTPUT);
  pinMode(power_on_led,OUTPUT);
  pinMode(sat_led,OUTPUT);
  pinMode(echo,INPUT);
  pinMode(ref_pot,INPUT);
  pinMode(kp_pot,INPUT);
  pinMode(ki_pot,INPUT);
  pinMode(kd_pot,INPUT);
  pinMode(pulsador,INPUT);
  // Inicializar variables
  digitalWrite(trigger,0);
  digitalWrite(power_on_led,1);
  digitalWrite(sat_led,1);
  for(int i=275; i>=0;i--){ // matrices de enteros para la representación
gráfica
          M1[i]=200;
          M1[i]=200;
  }
  Serial.begin(2000000); //Iniciar comunicación serial
  tft.reset();           // Reset display
  uint16_t identifier = tft.readID(); // Leer el identificador de la
pantalla
  tft.begin(identifier);              // iniciar la pantalla
  //Iniciar y configurar el timer
```

```
  Timer1.initialize(tsamp);
  Timer1.attachInterrupt(interrupt_routine);
  initDisplay();
  //Secuencia de bienvenida
  digitalWrite(power_on_led,0);
  digitalWrite(sat_led,0);
  delay(200);
  digitalWrite(power_on_led,1);
  digitalWrite(sat_led,1);
  delay(200);
  digitalWrite(sat_led,0);
}


void loop() {
  USB_POT=digitalRead(interruptor_USB); //Leer estado de interruptor USB
  if (interrupt==1){ // Si ha saltado la interrupción
     //Calcular el periodo de muestreo
     t0 = t;
     t = micros();
     ts = (t-t0)*1e-6;
     //Desactivar el flag de interrupciones
     interrupt=0;
     //Muestrear salida
     distancia=muestrea_distancia();
     if(USB_POT==0){ //Si estoy en modo usb interfaz USB_POT
        // MODO USB
        modo=5; // Mostrar el modo de visualización específico para USB
        read_Serial(); // Leer puerto serie
        write_serial(ts,v*12.0/255,distancia,other);//Escribir puerto serie
     }
     else{
      // MODO NO USB
      LA_LC=digitalRead(interruptor_REALIMENTACION); // Leer el estado de
interruptor de realimentación
      read_pot();                                    // Leer potenciómetros
      if(LA_LC==1){
        // MODO LAZO CERRADO
        bool P=digitalRead(pulsador);// Leer estado pulsador
```

```
        if(P==1){ // Si esta pulsado ...
          modo=1+modo; // Cambio el modo de visualización modo++
          if (modo>=modo_max){// Si el modo es mayor que el modo max
            modo=0;//Volver al primer modo
          }
          delay(10); // Parar durante milisegundos
        }
        //Regulador PID
        e_act=ref/100*12-distancia/100*12; // error en voltios
        de=(e_act-e_prev)/ts;              // derivada del error
        error_suma +=ki*e_act;             // Accion integral
        e_prev=e_act;                      // Preparar para el siguiente
paso
        v_volts=kp*e_act+error_suma+kd*de; // Señal de control en voltios
        v=int(v_volts/12*255);             // Señal de control en ciclo de
trabajo
        }
      else{
        // LAZO ABIERTO
        v=round(ref/100*255);            // La señal de control es igual
que la referencia (escalada)
        modo=4;                          // Fuerzo el modo de
visualización al específico de lazo abierto
        }
      }
     // Limitar y administrar el led de saturación
     if (v>255){
      v=255;
      digitalWrite(sat_led,1);
      }
     else if (v<0){
      v=0;
      digitalWrite(sat_led,0);

     }
     else {
      digitalWrite(sat_led,0);
     }
     // Establecer la señal de cotrol
     analogWrite(v_pin,floor(v));
```

```
      // Refrescar display
      refresh_display(modo,distancia,ref,kp,ki,kd,v*12.0/255,kp*e_act,error_
suma,kd*de);
    }

  }


int UC_LIGHT_BLUE = tft.color565(46, 172,184 );
int UC_DARK_BLUE = tft.color565(0, 103,113 );
void initDisplay(){
  // Función que muestra las animaciones en pantalla de bienvenida
  tft.setRotation(3);
  int UC_LIGHT_BLUE = tft.color565(46, 172,184 );
  int UC_DARK_BLUE = tft.color565(0, 103,113 );
  tft.fillScreen(UC_DARK_BLUE);
  tft.setFont(&FreeSansBoldOblique9pt7b);
  tft.setTextColor(WHITE);
  tft.setCursor(17,25);
  tft.setTextSize(1);
  tft.println("UNIVERSIDAD DE CANTABRIA");
  tft.setTextSize(2);
  tft.setCursor(80,75);
  tft.println("PLANTA");
  tft.setCursor(40,125);
  tft.println("LEVITACION");
  tft.setCursor(40,175);
  tft.println("NEUMATICA");
  tft.setCursor(0,230);
  tft.setTextSize(1);
  tft.println("Alumno: Fernando Quevedo Revilla");
  delay(5000);
  tft.setFont();

  }

void refresh_display(int modo_actual, float distancia,float ref, float
kp,float ki,float kd,float tension,float vp,float vi,float vd){
```

```
if(modo_init==1){
  modo_anterior=modo_actual+1;
  modo_init=0;
  }
if(modo_actual!=modo_anterior ){
  // Actualizar plantilla o fondo
  modo_anterior=modo_actual;
  tft.setTextColor(WHITE);
  tft.fillScreen(UC_DARK_BLUE);
  tft.drawBitmap(0,0,UC,LOGO_WIDTH,LOGO_HEIGHT,UC_LIGHT_BLUE);
  tft.setTextSize(1);
  tft.setCursor(72, 7);
  tft.setFont(&FreeSerifItalic9pt7b);
  tft.print("UNIVERSIDAD DE CANTABRIA");
  switch (modo_actual) {
  case 0:
    // Pantalla general
    tft.setCursor(120, 35);
    tft.print("Vista General");
    tft.setTextSize(2);tft.setFont();
    tft.setCursor(5, 60);
    tft.println("Distancia");
    tft.setCursor(150, 60);
    tft.println("=");
    tft.setCursor(280, 60);
    tft.println("cm");
    tft.setCursor(5, 90);
    tft.println("Referencia");
    tft.setCursor(150, 90);
    tft.println("=");
    tft.setCursor(280, 90);
    tft.println("cm");
    tft.setCursor(5, 120);
    tft.println("Kp");
    tft.setCursor(150, 120);
    tft.println("=");
    tft.setCursor(5, 150);
    tft.println("Ki");
    tft.setCursor(150, 150);
```

```
  tft.println("=");
  tft.setCursor(5, 180);
  tft.println("Kd");
  tft.setCursor(150, 180);
  tft.println("=");
  tft.setCursor(5, 210);
  tft.println("V");
  tft.setCursor(150, 210);
  tft.println("=");
  tft.setCursor(295, 208);
  tft.setTextSize(1);
  tft.println("Modo");
  tft.fillCircle(308, 228,10, WHITE);
  tft.setCursor(303, 222);
  tft.setTextSize(2);
  tft.setTextColor(UC_LIGHT_BLUE);
  tft.println(modo_actual);


break;
case 1:
  // Vista gráfico
  tft.setCursor(120, 35);
  tft.print("Vista Grafica");
  tft.setTextSize(2);tft.setFont();
  tft.fillRect(0, 235,280, 3, BLUE);
  tft.fillRect(280,49,3, 189, BLUE);
  tft.setTextSize(1);
  tft.setCursor(285,225);
  tft.print("-0");
  tft.setCursor(285,181);
  tft.print("-25");
  tft.setCursor(285,137);
  tft.print("-50");
  tft.setCursor(285,93);
  tft.print("-75");
  tft.setCursor(285,49);
  tft.print("-100");
  tft.setCursor(270, 30);
```

```
  tft.setTextSize(1);
  tft.println("Modo");
  tft.fillCircle(308, 35,10, WHITE);
  tft.setCursor(303, 29);
  tft.setTextSize(2);
  tft.setTextColor(UC_LIGHT_BLUE);
  tft.println(modo_actual);
break;
case 2:
  // Vista detalle
  tft.drawBitmap(80,43,PID,168,45,WHITE);
  tft.drawRect(80,40,175,44,UC_LIGHT_BLUE);
  tft.setCursor(120, 35);
  tft.print("Vista detalle");
  tft.setTextSize(2);tft.setFont();
  tft.setCursor(5, 90);
  tft.println("e");
  tft.setCursor(45, 90);
  tft.println("=");
  tft.setCursor(5, 120);
  tft.println("Kp");
  tft.setCursor(45, 120);
  tft.println("=");
  tft.setCursor(5, 150);
  tft.println("Ki");
  tft.setCursor(45, 150);
  tft.println("=");
  tft.setCursor(5, 180);
  tft.println("Kd");
  tft.setCursor(45, 180);
  tft.println("=");
  tft.setCursor(5, 210);
  tft.println("V");
  tft.setCursor(45, 210);
  tft.println("=");
  tft.setCursor(150, 120);
  tft.println("Vp");
  tft.setCursor(225, 120);
  tft.println("=");
```

```
        tft.setCursor(150, 150);
        tft.println("Vi");
        tft.setCursor(225, 150);
        tft.println("=");
        tft.setCursor(150, 180);
        tft.println("Vd");
        tft.setCursor(225, 180);
        tft.println("=");


        tft.setCursor(295, 208);
        tft.setTextSize(1);
        tft.println("Modo");
        tft.fillCircle(308, 228,10, WHITE);
        tft.setCursor(303, 222);
        tft.setTextSize(2);
        tft.setTextColor(UC_LIGHT_BLUE);
        tft.println(modo_actual);
        tft.setCursor(0,230);
        tft.setTextColor(BLUE);
        tft.setTextSize(1);
        tft.println("Distancias en cm, tensiones en V");

      break;
      case 3:
        // Vista sobre el equipo
        tft.setCursor(120, 35);
        tft.print("Sobre el equipo:");
        tft.setFont(&FreeSerifItalic9pt7b);

        tft.setCursor(0, 65);
        tft.println("Este equipo ha sido concebido exclusivamente para el
uso practico en asignaturas de contro_l de la UC\n"
        "Interruptor USB:Habilita la comunicacion vi_a USB\n"
        "Interruptor LA/LC: Lazo abierto o cerrado\n"
        "Por favor use el sistema con responsabilidad\n");


        tft.setFont();
```

```
  tft.setCursor(295, 208);
  tft.setTextSize(1);
  tft.println("Modo");
  tft.fillCircle(308, 228,10, WHITE);
  tft.setCursor(303, 222);
  tft.setTextSize(2);
  tft.setTextColor(UC_LIGHT_BLUE);
  tft.println(modo_actual);
break;
case 4:
  // Vista modo USB
  tft.setCursor(120, 35);
  tft.print("Vista Lazo Abierto");
  tft.setTextSize(2);tft.setFont();
  tft.setCursor(5, 60);
  tft.println("Distancia");
  tft.setCursor(150, 60);
  tft.println("=");
  tft.setCursor(280, 60);
  tft.println("cm");
  tft.setCursor(5, 120);
  tft.println("V");
  tft.setCursor(150, 120);
  tft.println("=");

  tft.setFont();
  tft.setCursor(295, 208);
  tft.setTextSize(1);
  tft.println("Modo");
  tft.fillCircle(308, 228,10, WHITE);
  tft.setCursor(303, 222);
  tft.setTextSize(2);
  tft.setTextColor(UC_LIGHT_BLUE);
  tft.println(modo_actual);


break;

case 5:
```

```
  // Vista modo USB
  tft.setCursor(120, 35);
  tft.print("Vista USB");
  tft.setTextSize(2);tft.setFont();
  tft.setCursor(5, 60);
  tft.println("Distancia");
  tft.setCursor(150, 60);
  tft.println("=");
  tft.setCursor(280, 60);
  tft.println("cm");
  tft.setCursor(5, 90);
  tft.println("Referencia");
  tft.setCursor(150, 90);
  tft.println("=");
  tft.setCursor(280, 90);
  tft.println("cm");
  tft.setCursor(5, 120);
  tft.println("V");
  tft.setCursor(150, 120);
  tft.println("=");


  tft.drawBitmap(50,115,USB,250,150,WHITE);

  tft.setFont();
  tft.setCursor(295, 208);
  tft.setTextSize(1);
  tft.println("Modo");
  tft.fillCircle(308, 228,10, WHITE);
  tft.setCursor(303, 222);
  tft.setTextSize(2);
  tft.setTextColor(UC_LIGHT_BLUE);
  tft.println(modo_actual);
default:

break;

}
```

```
}
else{
  // Actualizar parámetros dinámicos
  switch (modo_actual) {
  case 0:
    // Pantalla general
    tft.setTextColor(WHITE,UC_DARK_BLUE);
    tft.setTextSize(2);
    tft.setCursor(200, 60);
    tft.println(distancia,2);
    tft.setCursor(200, 90);
    tft.println(ref,2);
    tft.setCursor(200, 120);
    tft.println(kp,2);
    tft.setCursor(200, 150);
    tft.println(ki,2);
    tft.setCursor(200, 180);
    tft.println(kd,2);
    tft.setCursor(200, 210);
    tft.println(tension);
    if(tension>=11.8){
      tft.drawRect(2,208,290,20,RED);
      tft.setTextColor(RED,UC_DARK_BLUE);
      tft.setTextSize(2);
      tft.setCursor(242, 210);
      tft.println("SAT!");
      //digitalWrite(sat_led,1);
    }
    else{
      tft.drawRect(2,208,290,20,UC_DARK_BLUE);
      tft.setTextColor(UC_DARK_BLUE,UC_DARK_BLUE);
      tft.setCursor(242, 210);
      tft.println("SAT!");
    }
  break;
  case 1:
    //Gráfica
    tft.fillRect(0, 49,280, 185, WHITE);
    M1[275]=(100-distancia)/100*188+40;
```

```
  for(int i=275; i>=1;i--){
    tft.drawPixel(i,M1[i],RED);
    tft.drawPixel(i,M2[i],WHITE);
    M2[i-1]=M1[i];

    }
  for(int i=275; i>=0;i--){
    M1[i]=M2[i];
    }
break;
case 2:
// Vista modo 2
  tft.setTextColor(WHITE,UC_DARK_BLUE);
  tft.setTextSize(2);
  tft.setCursor(60, 60);
  tft.setCursor(70, 90);
  tft.println(distancia-ref,1);
  tft.setCursor(70, 120);
  tft.println(kp,2);
  tft.setCursor(70, 150);
  tft.println(ki,2);
  tft.setCursor(70, 180);
  tft.println(kd,1);
  tft.setCursor(70, 210);
  tft.println(tension,1);
  tft.setCursor(230, 90);

  tft.setCursor(250, 120);
  tft.println(vp,2);
  tft.setCursor(250, 150);
  tft.println(vi,2);
  tft.setCursor(250, 180);
  tft.println(vd,2);

break;

case 3:
// Vista modo 3
tft.setCursor(0,200);
```

```
if (blink_signal<=0){
  tft.setTextColor(GREEN);
  blink_signal++;
}
else{
  tft.setTextColor(BLUE);
  blink_signal++;
  if(blink_signal>=10){
    blink_signal=-10;
  }
}
tft.print("Presione el pulsador para continuar...");


break;
case 4:
// Vista modo 4
  tft.setTextColor(WHITE,UC_DARK_BLUE);
  tft.setTextSize(2);
  tft.setCursor(200, 60);
  tft.println(distancia);
  tft.setCursor(200, 120);
  tft.println(tension,2);
  tft.setCursor(200, 150);

  if(tension>=11.8){
    tft.drawRect(2,208,290,20,RED);
    tft.setTextColor(RED,UC_DARK_BLUE);
    tft.setTextSize(2);
    tft.setCursor(135, 210);
    tft.println("SAT!");
    //digitalWrite(sat_led,1);
  }
  else{
    //digitalWrite(sat_led,0);
    tft.drawRect(2,208,290,20,UC_DARK_BLUE);
    tft.setTextColor(UC_DARK_BLUE,UC_DARK_BLUE);
    tft.setCursor(135, 210);
    tft.println("SAT!");
```

```
        }

       break;
       case 5:
       // Vista modo 5
        tft.setTextColor(WHITE,UC_DARK_BLUE);
        tft.setTextSize(2);
        tft.setCursor(200, 60);
        tft.println(distancia);
        tft.setCursor(200, 90);
        tft.println(ref,2);
        tft.setCursor(200, 120);
        tft.println(tension,2);
        tft.setCursor(200, 150);
        if(tension>=11.8){
          tft.drawRect(2,118,310,20,RED);
          tft.setTextColor(RED,UC_DARK_BLUE);
          tft.setTextSize(2);
          tft.setCursor(265, 120);
          tft.println("SAT!");
          //digitalWrite(sat_led,1);
        }
        else{
          //digitalWrite(sat_led,0);
          tft.drawRect(2,118,310,20,UC_DARK_BLUE);
          tft.setTextColor(UC_DARK_BLUE,UC_DARK_BLUE);
          tft.setCursor(265, 120);
          tft.println("SAT!");
        }
      break;
      default:

      break;

      }
     }
 }
void read_Serial(){
 // Función leer del puerto serie
```

```
  Serial.write('s');
  Serial.readBytes((byte*)&ref, sizeof(float));
  Serial.readBytes((byte*)&v_volts, sizeof(float));
  v=(int)floor(v_volts/12*255);
  }
void write_serial(float ts, float s_control, float salida, float other){
  // Función escribir en el puerto serie
  Serial.write((byte*)&ts, sizeof(float));
  Serial.write((byte*)&s_control, sizeof(float));
  Serial.write((byte*)&salida, sizeof(float));
  Serial.write((byte*)&other, sizeof(float));
  Serial.flush();

  }
void read_pot(){
  // Función leer potenciómetros
  ref=(analogRead(ref_pot)*102.0/1023)-0.4;
  kp=(analogRead(kp_pot)*2.0/1023);
  ki=analogRead(ki_pot)*2.0/1023;
  kd=analogRead(kd_pot)*2.0/1023-0.04;
  }
float muestrea_distancia(){
    //Lectura sensor ultrasónico
    digitalWrite(trigger,1);  // Activar el trigger (disparo)
    delayMicroseconds(10);    // Mantenerlo activado 10us
    digitalWrite(trigger,0);  // Pasado ese tiempo,desactivarlo
    tiempo=pulseIn(echo,1);   // Obtener el tiempo hasta que se detecte
echo
    float d=100-tiempo/58.3 -3.75; // Calcular el la velocidad a partir del
tiempo
    // Si debido a ruido, la distancia es negativa, forzarla a cero
    if(d<=0){
      return 0;
    }
    return  d;                //Si la distancia es positiva, devolver ese
valor
  }

 void interrupt_routine(){
    interrupt=1;
```

```
    }
```

ANEXO 4: CÓDIGO PYTHON:

```
# -*- coding: utf-8 -*-
"""
Apliación levitador neumático UNIVERSIDAD DE CANTABRIA
CREADO Sat Feb 12 07:30:12 2022
ULTIMA VERSION 03/06/2022
AUTOR: FERNANDO QUEVEDO REVILLA
"""
#Importar las librerias necesarias
import sys
from PyQt5 import QtCore, uic
from PyQt5.QtWidgets import QMainWindow, QApplication,
QFileDialog,QMessageBox
from pyqtgraph import QtGui, PlotWidget, plot
from PyQt5.QtCore import QPropertyAnimation,QEasingCurve
from time import time, sleep
import pyqtgraph as pg
import numpy as np
import scipy.io
import serial, time,struct,serial.tools.list_ports
from import_file import import_file
controladorlib = import_file('./controladorlib.py')

#Funciones
def tic():
    return time.time()
def toc(t_ref):
    return time.time()-t_ref
```

```
#Clase CONTROL PRINCIPAL
class control(QMainWindow):
    def __init__(self):
        super().__init__()
        uic.loadUi("GUI.ui",self)     #Cargar interfaz.ui
        #Ajustes de inicio
        self.plot_widget.setBackground('w')
        self.plot_widget.setBackground('w')
        self.plot_widget.setLabel('left', 'Centímetros')
        self.plot_widget.setLabel('right', 'Voltios')
        self.plot_widget.setLabel('bottom', 't (s)')
        self.plot_widget.setTitle(" ")
        self.plot_widget.showGrid(x = True, y = True)
        self.plot_widget.setXRange(0.0, 10.0, padding = 0)
        self.plot_widget.setYRange(0.0, 105.0, padding = 0)

        #otros ajustes....
        self.center_Screen()
        #eliminar barra superior
        self.setWindowFlag(QtCore.Qt.FramelessWindowHint)
        self.setWindowOpacity(1)
        #FUNCIONES DE CADA ELEMENTO DE LA INTERFAZ
        #SLIDER

self.slider_REFERENCIA.valueChanged.connect(self.slider_REFERENCIA_CHANGE)
        #SPINBOX

self.box_REFERENCIA.valueChanged.connect(self.box_REFERENCIA_CHANGE)
        # up frame movement
        self.frame_UP.mouseMoveEvent=self.MOVE_SCREEN
        #BUTTON
        self.button_CLOSE.clicked.connect(self.CLOSE_CLICKED)
        self.button_EXPAND.clicked.connect(self.EXPAND_CLICKED)
```

```
        self.button_MINIMIZE.clicked.connect(self.MINIMIZE_CLICKED)
        self.button_COMPRESS.clicked.connect(self.COMPRESS_CLICKED)
        self.button_MENU.clicked.connect(self.MENU_CLICKED)
        self.button_HOME.clicked.connect(self.HOME_CLICKED)
        self.button_MAIN.clicked.connect(self.MAIN_CLICKED)
        self.button_INFO.clicked.connect(self.INFO_CLICKED)
        self.button_SETTINGS.clicked.connect(self.SETTINGS_CLICKED)
        self.button_STOP.clicked.connect(self.STOP_CLICKED)
        self.button_START.clicked.connect(self.START_CLICKED)
        self.button_SAVE.clicked.connect(self.SAVE_CLICKED)
        self.button_SEARCH.clicked.connect(self.SEARCH_CLICKED)

self.button_LOAD_CONTROLLER.clicked.connect(self.LOAD_CONTROLLER_CLICKED)

self.button_SAVE_CONTROLLER.clicked.connect(self.SAVE_CONTROLLER_CLICKED)

    def center_Screen (self):
        resolution = QtGui.QDesktopWidget().screenGeometry()
        self.move(int((resolution.width()-self.frameSize().width())/2),
                  int((resolution.height()-self.frameSize().height())/2))
    #BOTONES
    def MENU_CLICKED(self):
        width=self.frame_LATERAL.width()
        if width==0:
            new_width=160


        else:
            new_width=0


self.animation=QPropertyAnimation(self.frame_LATERAL,b'minimumWidth')
        self.animation.setDuration(500)
        self.animation.setStartValue(width)
        self.animation.setEndValue(new_width)
```

```
        self.animation.setEasingCurve(QtCore.QEasingCurve.InOutQuart)
        self.animation.start()
    def HOME_CLICKED(self):
        self.stackedWidget.setCurrentWidget(self.page_HOME)
    def MAIN_CLICKED(self):
        self.stackedWidget.setCurrentWidget(self.page_MAIN)
    def INFO_CLICKED(self):
        self.stackedWidget.setCurrentWidget(self.page_INFO)
    def SETTINGS_CLICKED(self):
        self.stackedWidget.setCurrentWidget(self.page_SETTINGS)

    def CLOSE_CLICKED(self):
         self.run=0
         self.close()
    def MINIMIZE_CLICKED(self):
         self.showMinimized()
    def EXPAND_CLICKED(self):
         self.showMaximized()
         self.button_COMPRESS.show()
         self.button_EXPAND.hide()
    def COMPRESS_CLICKED(self):
         self.showNormal()
         self.button_COMPRESS.hide()
         self.button_EXPAND.show()
    def mousePressEvent(self, event):
        self.click_pos=event.globalPos()

    def MOVE_SCREEN(self,event):

        if event.buttons()== QtCore.Qt.LeftButton:
            self.move(self.pos()+event.globalPos()-self.click_pos)
            self.click_pos=event.globalPos()
            event.accept()
```

```
    def SAVE_CLICKED(self):


        options = QFileDialog.Options()

        options |= QFileDialog.DontUseNativeDialog

        fileName, _ = QFileDialog.getSaveFileName(self,"Guardar
señales","","Matlab Files (*.mat);;All Files (*)", options=options)

        filename = fileName.split('.')[0]

        if len(filename) > 0:

            scipy.io.savemat(filename+'.mat', dict(t = np.vstack(self.t), u
= np.vstack(self.ref), v = np.vstack(self.v), y = np.vstack(self.d)))


    def SEARCH_CLICKED(self):

        self.comboBox_BAUDRATE.clear()

        self.comboBox_COM.clear()

        ports = [port.device for port in
serial.tools.list_ports.comports()]

        baudrates = ['2000000','1200', '2400', '4800', '9600', '19200',
'38400', '115200']

        self.comboBox_COM.addItems(ports)

        self.comboBox_BAUDRATE.addItems(baudrates)

        if ports==[]:

            msg = QMessageBox()

            msg.setIcon(QMessageBox.Warning)

            msg.setText("No se encontró ningún dispositivo USB")

            msg.setInformativeText("Revise las conexiones PC-Planta")

            msg.setWindowTitle("Aviso")

            msg.setDetailedText("Posibles causas:\n"

                                "1.No la planta no está conectada al
equipo\n"

                                "2.El cable no esta conectado al sistema\n"

                                "3.No se reconoce al sistema\n")

            msg.setStandardButtons(QMessageBox.Cancel)

            retval = msg.exec_()


    def LOAD_CONTROLLER(self, filename):
```

```
        self.filecontrolador = open(filename,'r')
        self.textEdit_CONTROLLER.setPlainText(self.filecontrolador.read())
        self.filecontrolador.close()
        self.groupBox_CONTROLLER.setTitle('CONTROLADOR ' + filename)

    def SAVE_CONTROLLER(self, filename):
        self.filecontrolador = open(filename,'w')
        self.filecontrolador.write(self.textEdit_CONTROLLER.toPlainText())
        self.filecontrolador.close()
        self.groupBox_CONTROLLER.setTitle('CONTROLADOR ' + filename)

    def LOAD_CONTROLLER_CLICKED(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getOpenFileName(self,"Cargar
controlador","","Python Files (*.py);;All Files (*)", options=options)
        filename = fileName.split('/')[-1]
        self.LOAD_CONTROLLER(filename)

    def SAVE_CONTROLLER_CLICKED(self):
        options = QFileDialog.Options()
        options |= QFileDialog.DontUseNativeDialog
        fileName, _ = QFileDialog.getSaveFileName(self,"Guardar
controlador","","Python Files (*.py);;All Files (*)", options=options)
        filename = fileName.split('/')[-1]
        self.SAVE_CONTROLLER(filename)

    def START_CLICKED(self):
        #print('Start clicked') #For debug
        self.run=1
        filename = self.groupBox_CONTROLLER.title().split()[-1]
        #print(filename)#For debug
        if filename=='"SELECCIONE_O_CREE_UN_ARCHIVO!")':
            msg = QMessageBox()
```

```
            msg.setIcon(QMessageBox.Warning)
            msg.setText("No se encontró ningún controlador")
            msg.setInformativeText("Revise la ubicación del controlador")
            msg.setWindowTitle("Aviso")
            msg.setDetailedText("Posibles causas:\n"
                                "1.Problemas con la extensión .py\n"
                                "2.El usuario no tiene premisos\n"
                                "3.El nombre no se corresponde\n")
            msg.setStandardButtons(QMessageBox.Cancel)
            retval = msg.exec_()

        self.SAVE_CONTROLLER(filename)
        self.LOAD_CONTROLLER(filename)
        controladorlib = import_file(filename)
        control = controladorlib.controlador()

        #Comenzar comunicacion Arduino
        if self.comboBox_COM.currentText()=="" or
self.comboBox_BAUDRATE.currentText()=="":
            QMessageBox.critical(self, "Error", "No se pudo conectar con la
planta\nRevise los ajustes de comunicación\n"
                                 "Acuda a la página de ajustes")
            return
        arduino =
serial.Serial(self.comboBox_COM.currentText(),self.comboBox_BAUDRATE.curren
tText())
        time.sleep(0.25)
        #print('Arduino conected')
        # Formato gráfica plot
        self.plot_widget.clear()
        ref_pen = pg.mkPen(color = (0, 0, 255), width = 1.25)
        v_pen = pg.mkPen(color = (0, 255, 0), width = 1.25)
        d_pen = pg.mkPen(color = (255, 0, 0), width = 1.25)
        ref_trace = self.plot_widget.plot([], [], name = "u", pen =
ref_pen)
```

```
        v_trace = self.plot_widget.plot([], [], name = "v", pen = v_pen)
        d_trace = self.plot_widget.plot([], [], name = "y", pen = d_pen)
        t_init=tic()
        packages_sent=0
        packages_received=0
        #Crear vectores
        n=1
        self.t = np.linspace(0.0, 100, 100000, dtype = 'float')
        self.ref = np.zeros_like(self.t)
        self.v= np.zeros_like(self.t)
        self.d = np.zeros_like(self.t)
        #print('Running')
        while self.run==1:
            #ENVIO PYTHON -> ARDUINO
            try:
                while arduino.read()!=b's':
                    pass
                #Lectura de parámetros, interfaz
                ref_= np.float32(self.box_REFERENCIA.value())
                self.ref[n]=ref_
                control.update(self.ts,
self.ref[n]/100*12,self.d[n]/100*12)
                self.v[n]=control.v
                if control.v>=12:
                    self.v[n]=np.float32(12)
                elif control.v<=0:
                    self.v[n]=np.float32(0)
                else :
                    self.v[n]=np.float32(control.v)

                arduino.write(np.float32(ref_))
                #print('send ref=',ref_)
                arduino.write(np.float32(self.v[n]))
                #print('send v=',self.v[n])
```

```
                packages_sent +=1



            except:
                pass


            #RECEPCION ARDUINO -> PYTHON
            try:
               n=n+1
               self.ts,self.v[n],self.d[n],other = struct.unpack('ffff',
arduino.read(16))
               packages_received +=1
               #print('d = ', self.d[n])
               #Actualizar pantalla
               self.progressBar.setValue(self.d[n])
               self.box_DISTANCE.setValue(self.d[n])
               # Calculo el tiempo
               self.t[n]=self.t[n-1]+self.ts;
               #Actualiza grafica
               ref_trace.setData(self.t[:n], self.ref[:n])
               v_trace.setData(self.t[:n], self.v[:n])
               d_trace.setData(self.t[:n], self.d[:n])
               self.plot_widget.update(True)


            except:
                pass
            QApplication.processEvents()
        arduino.close()



    def STOP_CLICKED(self):
        self.run=0
```

```
    def slider_REFERENCIA_CHANGE(self):
        self.box_REFERENCIA.blockSignals(True)
        self.box_REFERENCIA.setValue(self.slider_REFERENCIA.value())
        self.box_REFERENCIA.blockSignals(False)

    def box_REFERENCIA_CHANGE(self):
        self.slider_REFERENCIA.blockSignals(True)
        self.slider_REFERENCIA.setValue(self.box_REFERENCIA.value())
        self.slider_REFERENCIA.blockSignals(False)


if __name__ == "__main__":
    app = QApplication(sys.argv)
    control_lev = control()
    control_lev.show()
    sys.exit(app.exec_())
```

ANEXO 5: HOJA DE CARACTERÍSTICAS DEL VENTILADOR

LUFT

PRODUCT SPECIFICATION
KLD012PP120CSWH

A. General Specification

	Item	Specification	Condition
1	Model No.	KLD012PP120CSWH	
2	Outline Dimension	120X120X25mm	
3	Rated Voltage	12VDC	
4	Starting Voltage	8VDC	
5	Rated Current Max.	0,25 A	At Rated Voltage, 25°C, 65% RH
6	Power Consumption	3	At Rated Voltage, 25°C, 65% RH
7	Speed	2500 RPM	At Rated Voltage, 25°C, 65% RH Free Air
8	Max. Airflow	88CFM 150M^3/h	At Rated Voltage AMCA Standard At Rated Current
9	Max. Static Pressure	5,2 mmH_2O	
10	Noise Level	41dB	At Rated Voltage Measured in a non-echo Chamber CNS 8753 Standard ISO 3744 Test Condition
11	Life	30,000/hrs at 25°C	MTBF (Mean Time Between Failures) Conf. Level 90%
12	No. of Blade	7 Blades	
13	No. of Pole	2 Poles	
14	Rotating Direction	Counter-Clockwise View From Label Side	
15	Tolerance	± 10%	At Rated Voltage
16	Weight	240	
17	Motor Type	Brushless motor, external rotor.	

B. Main Materials / Parts Specification

	Materials / Parts	Specification
1	Housing & impeller	Plastiac PBT UL 94.
2	Bearing	Sleeve bearing.
3	Terminal	Leader wire 1007 AWG 26
4	Connector	Available

C. Safety Approvals :

Safety Approvals	Safety Approvals	Safety Approvals	Safety Approvals
CE	**UL**		

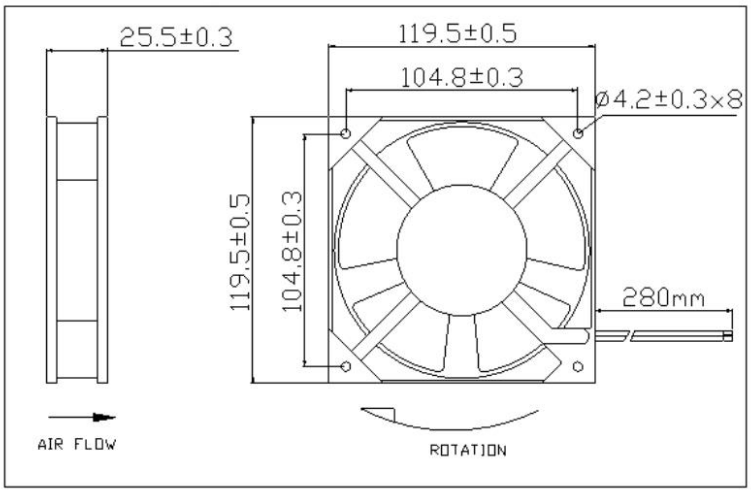
D. Environmental Specification

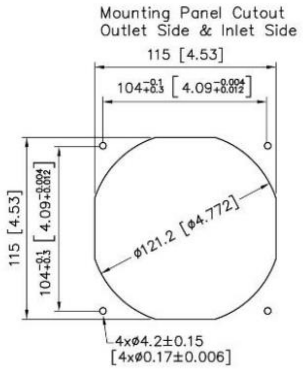
Item		Specification / Condition
1	Operating Temp. Range	Temperature : -10°C ~ + 70°C Humidity : 35% - 85% RH
2	Storage Temperature	Temperature : -30°C ~+ 180°C Hot test. Humidity : 35% - 85% RH
3	Humidity	Per MIL-STD 202F Method 103B; Life: 96 hours; Humidity : 95% RH; Temperature: 40 ± 2°C
4	Thermal Shock	Per MIL-STD 202F Method 107D, Condition D
5	Insulation Shock	UL: Class A
6	Packing Vibration Test	Packing Condition: X, Y, Z 3directions, 1.1 G load vibration test for 30 min.
7	Packing Shock Proof Test	1 corner, 3 edges, 6 faces natural drop from 60 cm high packing

E. Electrical Specification

Item		Specification/Condition
1	Insulation Resistance	100MΩ between frame and unshielded wire at 500VDC/min
2	Dielectric Strength	5mA Max at 500VDC 60Hz one minute, (between frame and terminal).
3	Locker Rotor Protection	Impedance protected.
4	Polarity Protection	None

F. Outline Dimension **UNIT: mm**





UNIT: mm[inch]

www.luft.es info@luft.es

ANEXO 6. HOJA DE CARACTERÍSTICAS SENSOR ULTRASÓNICO:

velleman®

VMA306

HC-SR05 ULTRASONIC SENSOR

USER MANUAL

VMA306

USER MANUAL

1. Introduction

To all residents of the European Union

Important environmental information about this product

This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.

If in doubt, contact your local waste disposal authorities.

Thank you for choosing Velleman®! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

2. Safety Instructions

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

- Indoor use only.
  Keep away from rain, moisture, splashing and dripping liquids.

3. General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- Familiarise yourself with the functions of the device before actually using it.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorised way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Due to constant product improvements, the actual product appearance might differ from the shown images.
- Product images are for illustrative purposes only.
- Do not switch the device on immediately after it has been exposed to changes in temperature. Protect the device against damage by leaving it switched off until it has reached room temperature.
- Keep this manual for future reference.

4. Overview

The VMA306 ultrasonic distance sensor provides an easy way for your projects to measure distances up to 4.5 m.

Arduino®		VMA306
+5 V	▶	VCC
D4	▶	ECHO
D2	▶	TNG
GND	▶	GND

voltage ..... 4.5 to 5.5 VDC
sound frequency ..... 40 KHz
measurement resolution ..... 0.3 cm
measurement angle ..... 15°
supply current ..... 10 to 40 mA
trigger pin format ..... 10 µS pulse
connector ..... 5-pin male
detection distance ..... 2 to 450 cm
dimensions ..... 45 x 20 x 13 mm

Use this device with original accessories only. Velleman nv cannot be held responsible in the event of damage or injury resulting from (incorrect) use of this device. For more info concerning this product and the latest version of this manual, please visit our website www.velleman.eu. The information in this manual is subject to change without prior notice.

Velleman® Service and Quality Warranty

Since its foundation in 1972, Velleman® acquired extensive experience in the electronics world and currently distributes its products in over 85 countries.

All our products fulfil strict quality requirements and legal stipulations in the EU. In order to ensure the quality, our products regularly go through an extra quality check, both by an internal quality department and by specialized external organisations. If, all precautionary measures notwithstanding, problems should occur, please make appeal to our warranty (see guarantee conditions).

General Warranty Conditions Concerning Consumer Products (for EU):

• All consumer products are subject to a 24-month warranty on production flaws and defective material as from the original date of purchase.

• Velleman® can decide to replace an article with an equivalent article, or to refund the retail value totally or partially when the complaint is valid and a free repair or replacement of the article is impossible, or if the expenses are out of proportion.

You will be delivered a replacing article or a refund at the value of 100% of the purchase price in case of a flaw occurred in the first year after the date of purchase and delivery, or a replacing article at 50% of the purchase price or a refund at the value of 50% of the retail value in case of a flaw occurred in the second year after the date of purchase and delivery.

• Not covered by warranty:

- all direct or indirect damage caused after delivery to the article (e.g. by oxidation, shocks, falls, dust, dirt, humidity...), and by the article, as well as its contents (e.g. data loss), compensation for loss of profits;

- consumable goods, parts or accessories that are subject to an aging process during normal use, such as batteries (rechargeable, non-rechargeable, built-in or replaceable), lamps, rubber parts, drive belts... (unlimited list);

- flaws resulting from fire, water damage, lightning, accident, natural disaster, etc....;

- flaws caused deliberately, negligently or resulting from improper handling, negligent maintenance, abusive use or use contrary to the manufacturer's instructions;

- damage caused by a commercial, professional or collective use of the article (the warranty validity will be reduced to six (6) months when the article is used professionally);

- damage resulting from an inappropriate packing and shipping of the article;

- all damage caused by modification, repair or alteration performed by a third party without written permission by Velleman®.

• Articles to be repaired must be delivered to your Velleman® dealer, solidly packed (preferably in the original packaging), and be completed with the original receipt of purchase and a clear flaw description.

• Hint: In order to save on cost and time, please reread the manual and check if the flaw is caused by obvious causes prior to presenting the article for repair. Note that returning a non-defective article can also involve handling costs.

• Repairs occurring after warranty expiration are subject to shipping costs.

• The above conditions are without prejudice to all commercial warranties.

The above enumeration is subject to modification according to the article (see article's manual).

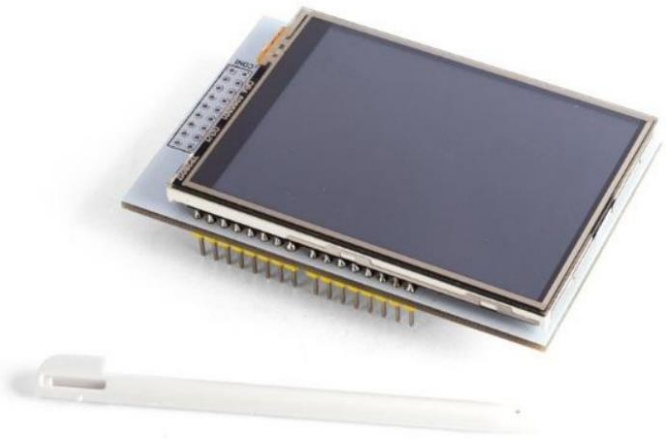
Made in PRC
Imported by Velleman nv
Legen Heirweg 33, 9890 Gavere, Belgium
www.velleman.eu

ANEXO 7. HOJA DE CARACTERÍSTICAS PANTALLA

velleman®

VMA412

2.8 INCH TOUCH SCREEN FOR ARDUINO® UNO/MEGA



USER MANUAL

VMA412

USER MANUAL

1. Introduction

To all residents of the European Union

Important environmental information about this product

This symbol on the device or the package indicates that disposal of the device after its lifecycle could harm the environment. Do not dispose of the unit (or batteries) as unsorted municipal waste; it should be taken to a specialized company for recycling. This device should be returned to your distributor or to a local recycling service. Respect the local environmental rules.

If in doubt, contact your local waste disposal authorities.

Thank you for choosing Velleman®! Please read the manual thoroughly before bringing this device into service. If the device was damaged in transit, do not install or use it and contact your dealer.

2. Safety Instructions

- This device can be used by children aged from 8 years and above, and persons with reduced physical, sensory or mental capabilities or lack of experience and knowledge if they have been given supervision or instruction concerning the use of the device in a safe way and understand the hazards involved. Children shall not play with the device. Cleaning and user maintenance shall not be made by children without supervision.

- Indoor use only.
  Keep away from rain, moisture, splashing and dripping liquids.

3. General Guidelines

- Refer to the Velleman® Service and Quality Warranty on the last pages of this manual.
- Familiarise yourself with the functions of the device before actually using it.
- All modifications of the device are forbidden for safety reasons. Damage caused by user modifications to the device is not covered by the warranty.
- Only use the device for its intended purpose. Using the device in an unauthorised way will void the warranty.
- Damage caused by disregard of certain guidelines in this manual is not covered by the warranty and the dealer will not accept responsibility for any ensuing defects or problems.
- Nor Velleman nv nor its dealers can be held responsible for any damage (extraordinary, incidental or indirect) – of any nature (financial, physical…) arising from the possession, use or failure of this product.
- Due to constant product improvements, the actual product appearance might differ from the shown images.
- Product images are for illustrative purposes only.
- Do not switch the device on immediately after it has been exposed to changes in temperature. Protect the device against damage by leaving it switched off until it has reached room temperature.
- Keep this manual for future reference.

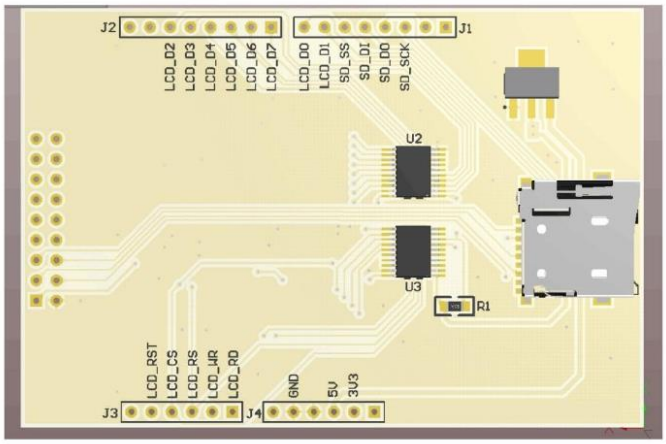
4. What is Arduino®

Arduino® is an open-source prototyping platform based in easy-to-use hardware and software. Arduino® boards are able to read inputs – light-on sensor, a finger on a button or a Twitter message – and turn it into an output – activating of a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so, you use the Arduino programming language (based on Wiring) and the Arduino® software IDE (based on Processing).

Surf to www.arduino.cc and www.arduino.org for more information.

5. Overview

Add a beautiful touch screen to your project with this 2.8" display and a microSD slot on a shield.
The shield comes completely assembled, tested and ready-to-use. No cables, no soldering required. Just plug in the shield, load the library and your display will be up and running in less than 10 minutes!



pin	description
LCD_RST	LCD reset pin
LCD_CS	LCD chip select
LCD_RS	LCD register select
LCD_WR	LCD write
LCD_RD	LCD read
GND	ground
5V	5 V
3V3	3.3 V
LCD_D0	LCD data bit 0
LCD_D1	LCD data bit 1
LCD_D2	LCD data bit 2
LCD_D3	LCD data bit 3
LCD_D4	LCD data bit 4
LCD_D5	LCD data bit 5
LCD_D6	LCD data bit 6
LCD_D7	LCD data bit 7
SD_SS	SD card slave select
SD_DI	SD card serial data in
SD_DO	SD card serial data out
SD_SCK	SD card serial clock

VMA412

resolution ..... 240 RGB (H) x 320 (V)
colour depth..... 262 000 colours
system interface
8-bits, 9-bits, 16-bits, 18-bits interface with 8080-I /8080-II series MCU
6-bits, 16-bits, 18-bits RGB interface with graphic controller
3-line / 4-line serial interface
display mode
full-colour mode (Idle mode OFF) . 262 000 colour (selectable colour depth mode by software)
reduced colour mode (Idle mode ON)..... 8-colour
operating temperature..... -40 °C to +85 °C

Use this device with original accessories only. Velleman nv cannot be held responsible in the event of damage or injury resulting from (incorrect) use of this device. For more info concerning this product and the latest version of this manual, please visit our website www.velleman.eu. The information in this manual is subject to change without prior notice.

Velleman® Service and Quality Warranty

Since its foundation in 1972, Velleman® acquired extensive experience in the electronics world and currently distributes its products in over 85 countries.

All our products fulfil strict quality requirements and legal stipulations in the EU. In order to ensure the quality, our products regularly go through an extra quality check, both by an internal quality department and by specialized external organisations. If, all precautionary measures notwithstanding, problems should occur, please make appeal to our warranty (see guarantee conditions).

General Warranty Conditions Concerning Consumer Products (for EU):

• All consumer products are subject to a 24-month warranty on production flaws and defective material as from the original date of purchase.

• Velleman® can decide to replace an article with an equivalent article, or to refund the retail value totally or partially when the complaint is valid and a free repair or replacement of the article is impossible, or if the expenses are out of proportion.

You will be delivered a replacing article or a refund at the value of 100% of the purchase price in case of a flaw occurred in the first year after the date of purchase and delivery, or a replacing article at 50% of the purchase price or a refund at the value of 50% of the retail value in case of a flaw occurred in the second year after the date of purchase and delivery.

• **Not covered by warranty:**

- all direct or indirect damage caused after delivery to the article (e.g. by oxidation, shocks, falls, dust, dirt, humidity...), and by the article, as well as its contents (e.g. data loss), compensation for loss of profits;

- consumable goods, parts or accessories that are subject to an aging process during normal use, such as batteries (rechargeable, non-rechargeable, built-in or replaceable), lamps, rubber parts, drive belts... (unlimited list);

- flaws resulting from fire, water damage, lightning, accident, natural disaster, etc....;

- flaws caused deliberately, negligently or resulting from improper handling, negligent maintenance, abusive use or use contrary to the manufacturer's instructions;

- damage caused by a commercial, professional or collective use of the article (the warranty validity will be reduced to six (6) months when the article is used professionally);

- damage resulting from an inappropriate packing and shipping of the article;

- all damage caused by modification, repair or alteration performed by a third party without written permission by Velleman®.

• Articles to be repaired must be delivered to your Velleman® dealer, solidly packed (preferably in the original packaging), and be completed with the original receipt of purchase and a clear flaw description.

• Hint: In order to save on cost and time, please reread the manual and check if the flaw is caused by obvious causes prior to presenting the article for repair. Note that returning a non-defective article can also involve handling costs.

• Repairs occurring after warranty expiration are subject to shipping costs.

• The above conditions are without prejudice to all commercial warranties.

The above enumeration is subject to modification according to the article (see article's manual).

Made in PRC
Imported by Velleman nv
Legen Heirweg 33, 9890 Gavere, Belgium
www.velleman.eu

ANEXO 8. EXTRACTO DE HOJAS DE CARACTERISTICAS DEL LM7812 Y LM7809

Electrical Characteristics (LM7812A)

Refer to the test circuit, 0°C < T_J < 125°C, I_O = 1 A, V_I = 19 V, C_I = 0.33 µF, C_O = 0.1 µF, unless otherwise specified.

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Unit
V_O	Output Voltage	T_J = +25°C		11.75	12.00	12.25	V
		I_O = 5 mA to 1 A, $P_O \leq$ 15 W, V_I = 14.8 V to 27 V		11.50	12.00	12.50	
Regline	Line Regulation(26)	V_I = 14.8 V to 30 V, I_O = 500 mA			10	120	mV
		V_I = 16 V to 22 V			4	120	
		T_J = +25°C	V_I = 14.5 V to 27 V		10	120	
			V_I = 16 V to 22 V		3	60	
Regload	Load Regulation(26)	T_J = +25°C, I_O = 5 mA to 1.5 A			12	100	mV
		I_O = 5 mA to 1 A			12	100	
		I_O = 250 mA to 750 mA			5	50	
I_Q	Quiescent Current	T_J = +25°C			5	6	mA
ΔI_Q	Quiescent Current Change	I_O = 5 mA to 1 A				0.5	mA
		V_I = 14 V to 27 V, I_O = 500 mA				0.8	
		V_I = 15 V to 30 V, T_J = +25°C				0.8	
$\Delta V_O/\Delta T$	Output Voltage Drift(27)	I_O = 5 mA			-1		mV/°C
V_N	Output Noise Voltage	f = 10 Hz to 100 kHz, T_A = +25°C			76		µV
RR	Ripple Rejection(27)	f = 120 Hz, V_O = 500 mA, V_I =14 V to 24 V			60		dB
V_{DROP}	Dropout Voltage	I_O = 1 A, T_J = +25°C			2		V
R_O	Output Resistance(27)	f = 1 kHz			18		mΩ
I_{SC}	Short-Circuit Current	V_I = 35 V, T_J = +25°C			250		mA
I_{PK}	Peak Current(27)	T_J = +25°C			2.2		A

Notes:

26. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.
27. These parameters, although guaranteed, are not 100% tested in production.

Electrical Characteristics (LM7809)

Refer to the test circuit, -40°C < T_J < 125°C, I_O = 500 mA, V_I = 15 V, C_I = 0.33 µF, C_O = 0.1 µF, unless otherwise specified.

Symbol	Parameter	Conditions		Min.	Typ.	Max.	Unit
V_O	Output Voltage	T_J = +25°C		8.65	9.00	9.35	V
		I_O = 5 mA to 1 A, $P_O \leq$ 15 W, V_I = 11.5 V to 24 V		8.60	9.00	9.40	
Regline	Line Regulation[8]	T_J = +25°C	V_I = 11.5 V to 25 V		6	180	mV
			V_I = 12 V to 17 V		2	90	
Regload	Load Regulation[8]	T_J = +25°C	I_O = 5 mA to 1.5 A		12	180	mV
			I_O = 250 mA to 750 mA		4	90	
I_Q	Quiescent Current	T_J = +25°C			5	8	mA
ΔI_Q	Quiescent Current Change	I_O = 5 mA to 1 A				0.5	mA
		V_I = 11.5 V to 26 V				1.3	
$\Delta V_O/\Delta T$	Output Voltage Drift[9]	I_O = 5 mA			-1		mV/°C
V_N	Output Noise Voltage	f = 10 Hz to 100 kHz, T_A = +25°C			58		µV
RR	Ripple Rejection[9]	f = 120 Hz, V_I = 13 V to 23 V		56	71		dB
V_{DROP}	Dropout Voltage	I_O = 1 A, T_J = +25°C			2		V
R_O	Output Resistance[9]	f = 1 kHz			17		mΩ
I_{SC}	Short-Circuit Current	V_I = 35 V, T_J = +25°C			250		mA
I_{PK}	Peak Current[9]	T_J = +25°C			2.2		A

Notes:

8. Load and line regulation are specified at constant junction temperature. Changes in V_O due to heating effects must be taken into account separately. Pulse testing with low duty is used.
9. These parameters, although guaranteed, are not 100% tested in production.

Typical Applications

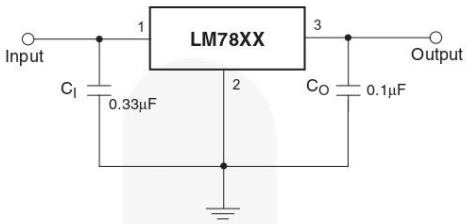


Figure 6. DC Parameters

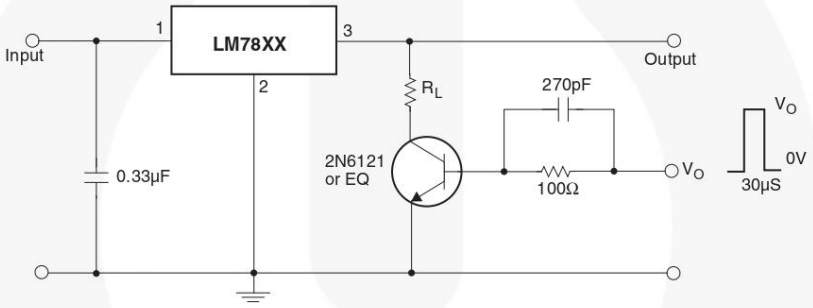


Figure 7. Load Regulation

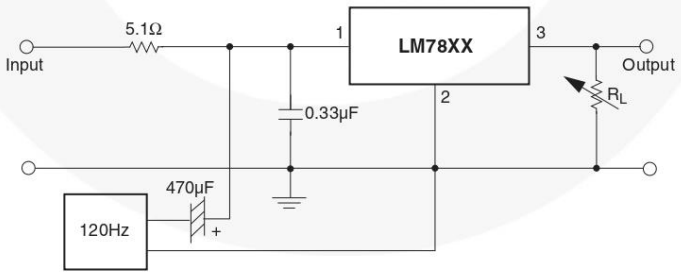


Figure 8. Ripple Rejection