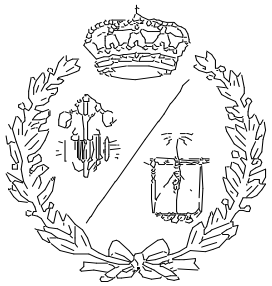
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

SISTEMA DE POSICIONAMIENTO Y LOCALIZACIÓN DE UN ROBOT AUTÓNOMO EN SUPERFICIES EXTERIORES

(POSITIONING AND LOCATION SYSTEM OF AN AUTONOMOUS ROBOT AT OUTDOOR SURFACES)

Para acceder al Título de

GRADUADA EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Cecilia Abascal Falagán

Febrero - 2021

AGRADECIMIENTOS

Antes de cerrar esta etapa tan importante en mi vida, quiero agradecer a todas las personas que me han acompañado en ella. No ha sido una etapa fácil para mí, he trabajado mucho en todos los sentidos para llegar hasta aquí.

A todos los profesores que me han enseñado y ayudado en todo este tiempo.

A mis compañeros, por los días de clase y biblioteca que parecían infinitos. Por apoyarme en mis días malos, han sido muchas horas juntos y me alegro mucho de haberlas compartido con vosotros.

A toda mi familia por el apoyo, pero en especial a mis abuelos, que, sin ellos, no podría estar hoy aquí. Gracias por confiar siempre en mí.

A mi tutora y amiga Elena por acompañarme en el desarrollo de este proyecto final.

Y a Pablo por darme apoyo y cariño incondicional en este último empujón.

Gracias.

RESUMEN

El objetivo de este Trabajo de Fin de Grado es realizar el estudio e implementación de un sistema que mejore el posicionamiento y localización de un robot autónomo comercial para el segado de superficies exteriores.

A modo de introducción se plantean los objetivos y antecedentes del proyecto. Se revisa el mundo de la robótica en la jardinería y los sistemas de posicionamiento utilizados actualmente.

Se combina la visión artificial y la automática para aportar una solución óptima para el correcto posicionamiento y localización del robot. Para ello se hace un estudio previo de los componentes a implementar y del software que los controla.

Finalmente, en función de los resultados obtenidos y de los conocimientos adquiridos se presentan las conclusiones sobre las dificultades generadas durante la ejecución del proyecto y sus líneas de trabajo futuras.

ABSTRACT

The objective of this Final Degree Project is to study and implement a system that improves the positioning and location of a commercial autonomous robot for mowing exterior surfaces.

As an introduction, the objectives and background of the project are presented. It is reviewed the robotics branch in the gardening scope and the positioning systems used nowadays.

Artificial vision and automatic are combined to provide an optimal solution for the correct positioning and location of the robot. For this, a preliminary study of the components to be implemented and the software that controls them is made.

Finally, based on the results obtained and the knowledge acquired, the conclusions on the difficulties generated during the execution of the project and its future lines of work are presented.

Índice de Documentos

Documento Nº1:

MEMORIA

Índice General

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1 ANTECEDENTES

La empresa Vega Cantabria S.L. dedicada a la jardinería en la cual trabajan con un robot segador autónomo propone mejorar cuestiones técnicas en el uso de este.

El principal problema del robot es su posicionamiento y localización en el espacio para poder optimizar su trabajo. Una buena localización es necesaria para distintas situaciones:

- Volver a su estación de carga de manera óptima sin recorrer zonas ya segadas o por trayectorias más largas, ya que, de lo contrario, la batería podría llegar a agotarse al tomar esos caminos más largos y no cumplir con el objetivo.
- La interpretación de su entorno es vital para su correcto funcionamiento cuando se encuentran imprevistos por el tipo de terreno o por la presencia de objetos no identificados.

Con el fin de mejorar estas cuestiones, se plantean varias ideas iniciales. Para hallar su posición en cada momento se propone implementar un encoder rotatorio encajado a las ruedas del robot.

También se propone incluir una cámara adicional para analizar el entorno de forma inteligente con el objetivo de capturar imágenes del entorno que le rodea y que el propio robot sea capaz de procesarlas y generar información útil para hallar su posición.

Para ello se puede colocar unas marcas por el jardín también conocidas como LandMarks con el objetivo de ser reconocidas por la cámara y que el robot pueda sacar conclusiones mediante unos algoritmos programados previamente.

La última idea fue introducir varios sensores que sean capaces de proporcionar la orientación del robot y a partir de ahí poder hallar su posición.

Para poder analizar correctamente todas las soluciones propuestas se necesita hacer un estudio previo.

1.2 OBJETIVO

El objetivo final de este proyecto es estudiar e implementar un sistema de localización y posicionamiento del robot en el espacio que se encuentra. Para llevar a cabo esto se emplean conocimientos de visión artificial y automática.

Se desglosa en tres principales partes:

- Identificación de balizas colocadas en el jardín. Mediante la visión artificial se pretende analizar el entorno y clasificar las balizas.
- Hallar la orientación del robot en el instante que encuentra una baliza. Al obtener la orientación, podemos saber el posicionamiento.
- Calcular la distancia entre el robot y el objeto identificado. Al saber la distancia, se puede dar una estimación de la localización del robot en el entorno.

Se demuestra a pequeña escala con la ayuda de un controlador.

2. PLANTEAMIENTO DEL PROBLEMA

2.1 ROBOTS EN LA JARDINERÍA

El mundo de la jardinería ha ido avanzando poco a poco desde 1830 con la creación del primer corta césped. Pero a partir de 1995 es cuando el mundo de la jardinería se vinculó al mundo de la electrónica y las nuevas tecnologías dando paso a su primer robot jardinero automático. Ese año se lanzó el primer robot jardinero totalmente automático y robotizado, dejando en un segundo plano al jardinero tradicional y garantizando el corte del jardín sin la intervención de ningún humano[1].

Desde entonces, no ha parado de crecer y mejorar, con el objetivo de dar comodidad, facilidad y un mejor acabado al usuario que desee este mantenimiento para su jardín, ya sea por falta de tiempo o capacidad para realizarlo por el mismo.

Para ello, también se utiliza como herramienta la inteligencia artificial, la cual proporciona una amplia variedad de soluciones para que el robot sea más eficaz como pueden ser: reconocer las características del suelo y césped y tomar decisiones basadas en algoritmos programados previamente; trabajar en terrenos con pendientes y evitar ciertos obstáculos como perros, piscinas; ahorrar tiempo para el usuario y tener la capacidad de controlarlo incluso cuando no estén en su domicilio[2].

2.1.1 Componentes principales

El funcionamiento de estos robots está compuesto por los siguientes componentes externos e internos [3]:

- Estación de carga: Es la base donde el robot recarga su batería.
- Cable perimetral: Este cable se coloca alrededor del jardín para indicar el circuito en el que tiene que trabajar.
- Cable guía: Este cable sirve para poder encontrar mejor la estación de carga. No es muy común utilizarlo.
- Dispositivo GPS: Se utiliza para un funcionamiento avanzado (no está disponible en todos los robots). Realiza un mapeo del área de trabajo el cual le permite optimizar el trabajo y evita pasar por las áreas ya cortadas del césped.

En algunos casos más avanzados se puede realizar un mapeo multizona, esto quiere decir que cuando se tenga un jardín de amplias o pequeñas dimensiones con diferentes características en su área, se puede modificar el tipo de corte.

- Sensor de obstáculos: Dispositivo para evitar objetos como macetas, animales, objetos domésticos, etc, los cuales le dificultan su trabajo.
- Sensor anticaída: Dispositivo que detecta un escalón o algún hueco irregular del jardín evitando su caída.
- Sensor de lluvia: Detecta las condiciones meteorológicas específicamente la lluvia. Por lo tanto, si comienza a llover, el robot regresará a su base para protegerse. A pesar de que está preparado para resistir ciertas condiciones meteorológicas para evitar un mayor deterioro.

Este tipo de robots se manejan desde el móvil a través de una aplicación móvil para facilitar aún más la programación al usuario y la información sobre él en cuanto a recordatorios o algún imprevisto que pueda existir.

Una de sus ventajas es que posee una batería recargable y evita utilizar combustibles eliminando la emisión de humos y contribuyendo así al medioambiente. Otra de sus ventajas es el tiempo que se ahorra el usuario en las tareas del jardín. El usuario solo debe gastar tiempo en programar el trabajo del robot a su gusto mediante la aplicación móvil.

Por otro lado, presenta varias desventajas, se puede decir que la instalación es compleja y es necesario un profesional. Hay que programar los modos de trabajo a utilizar dependiendo del jardín y realizar un circuito con un cable perimetral para acotar el espacio de trabajo del robot. Además, actualmente el coste de estos robots es muy elevado.

2.1.2 Características a la hora de escoger un robot

Hay que tener en cuenta una serie de características a la hora de escoger un robot comercial para obtener el mayor rendimiento posible:

- Tamaño del jardín: Cada robot está fabricado para unos metros cuadrados determinados, así que la elección dependerá de las dimensiones del jardín.
- Autonomía de la batería: Cuanta más autonomía tengas, mejor rendimiento tendrá el robot.

- Peso del robot: Se tiene en cuenta para las personas con movilidad reducida o personas mayores que no sean capaces de transportarlo en caso de que haya alguna incidencia.
- Ruido: En el caso de las personas que cuenten con vecinos o simplemente porque no quieran tener un sonido incómodo durante un periodo de tiempo determinado. A pesar de que la mayoría de los robots son bastante silenciosos.
- Instalación: Existe una gama que cuentan con un asistente de programación para facilitar la instalación. Esto es muy útil para la mayoría de los usuarios que no estén familiarizados con el mundo de las tecnologías.

Actualmente las marcas comerciales con mayor éxito en la fabricación de este tipo robot son las que se muestran a continuación[4]:

Tabla 1. Competencia en el mercado. (Fuente: [4])

También se muestra una tabla comparativa de los robots más demandados por los usuarios[4]:

Modelo	Dimensiones (centímetros)	Precio (Precio)	Peso (kilogramos)	Tamaño Jardín (metros cuadrados)	Figura del modelo
WORX WR105SI Robot	62x53x29 cm	899,99€	16.5 kg	Hasta 500 m^2	
Robomow RC304U	63x46x21 cm	1199,00 €	20.6 kg	Hasta 1000 m^2	
Bosch Robot cortacésped Indego	44x36x22.2 cm	942.99€	7.6 kg	Hasta 350 m^2	

Tabla 2. Modelos comerciales más famosos. (Fuente: [4])

Robomow, aparte de ser de las mejores marcas, trabaja con la empresa Vega Cantabria S.L. que me ha encargado el estudio. Todos los robots de jardinería que distribuyen son de esta marca.

2.2 ROBOMOW

Se parte de la base de estos dos modelos de Robomow, los cuales se han visto en funcionamiento para poder observar cómo trabajan y los problemas que dan en el día a día. Gracias a ello, se ha tenido la posibilidad de analizar estos problemas de forma real.

Hay dos modelos principales adquiridos por los clientes:

2.2.1 Modelo RC

Este modelo es el más simple, pero únicamente porque está compuesto solo por una cuchilla de acero.

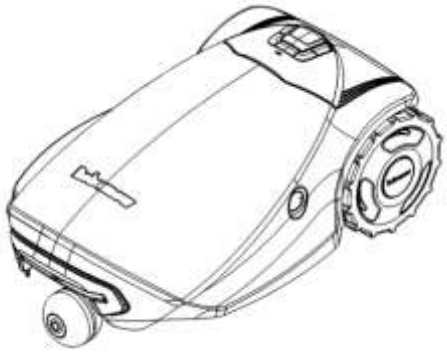


Figura 1. Diseño exterior del modelo RC (Fuente: [5])

A pesar de esto, ofrece una gran cantidad de posibilidades como es el área del jardín que puede abarcar como máximo 2000 metros cuadrados[5].

Una de sus ventajas son sus pequeñas dimensiones (63x43x21 cm) y esto hace que sea más fácil de manejar. Cuenta con una serie de sensores de choque, elevación e inclinación y con un modo para cambiar la altura y ser capaz de poder cortar el césped en dos áreas con diferente altura. Para ello también tiene unas ruedas anchas y de gran agarre.

El precio de este robot puede llegar hasta los 2000€ dependiendo de todas las características adicionales que lleven integrados en cada modelo.

2.2.2 Modelo RS

Este modelo es el que más se conoce ya que se ha trabajado con él la mayor parte del tiempo, puesto que en la zona de Cantabria tiene una mayor demanda:

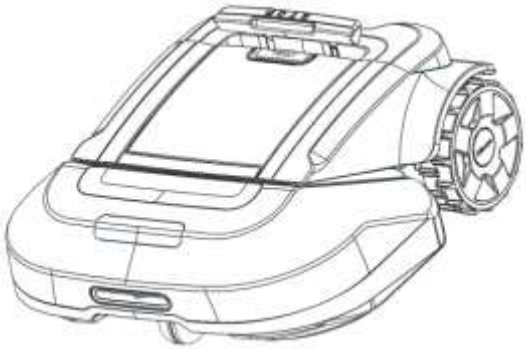


Figura 2. Diseño exterior modelo RS (Fuente: [6])

A diferencia del modelo RC, este cuenta con dos cuchillas lo cual hace que sus dimensiones sean mayores (73,5 x 66 x 31 cm). Puede trabajar en jardines de hasta 4000 metros cuadrados.

Una de sus ventajas es que el motor no tiene escobillas por lo que gana más velocidad, menos ruido y requiere un bajo mantenimiento. Esto hace que sea más potente y eficiente en el corte y que tenga una mayor durabilidad a la larga. [6]

2.2.3 Control para los usuarios

Cualquiera de los modelos posee una aplicación móvil que facilitan al usuario el control del robot que se muestra en la siguiente figura (3). Esta aplicación llamada Robomow App ofrece una serie de ventajas:

- Poner en marcha el robot cuando quieras.
- Regresar a su estación de carga.
- Establecer el horario de corte.
- Estado de carga.
- Contacto con servicio técnico.

Figura 3. Aplicación para el control de Robomow. (Fuente: [7])

2.3 SISTEMAS DE POSICIONAMIENTO EN LA ROBÓTICA

Los robots autónomos dedicados a trabajar en exteriores deben de tener en cuenta que el dato básico para poder realizar un trabajo óptimo es su velocidad y posición. Ya que debe realizar sus tareas de manera autónoma y tener en cuenta estos datos. Hoy en día existen gran multitud de aplicaciones y procesos para hallar la localización de un robot móvil. A continuación, se explican los más comunes[7]:

2.3.1 Sistemas odométricos

Estos sistemas se basan en la Odometría. La Odometría se incorpora en la gran mayoría de procesos de localización. Es capaz de proporcionar la trayectoria, posición y orientación del robot gracias al movimiento de las ruedas sabiendo la velocidad de sus ejes y su velocidad angular. Estos sistemas normalmente estan vinculados a sensores u otros dispositivos que les puedan ayudar a la hora de hallar su posición inicial u otros datos útiles[7].

Una de sus ventajas es la velocidad de respuesta a la hora de obtener datos. Y uno de sus inconvenientes es que a la hora de tomar las medidas puede ocurrir una diferencia entre el valor de la posición real y la estimada, la cual se puede ir acumulando medida tras medida y ocasionar un resultado totalmente distinto.

2.3.2 Sistemas de navegación inercial

Estos sistemas están compuestos por un sensor de movimiento(acelerómetro) y por un sensor de rotación(giróscopo). Proporcionan la velocidad, orientación y posición de un objeto sin necesidad de una referencia externa. Existen múltiples dispositivos que integran estos dos componentes. En este proyecto, se elige un sensor el cual proporciona estas funciones que se explican más adelante en detalle[7].

Su mayor inconveniente es la sensibilidad a la hora de tomar medidas, ya que a la mínima variación que el sensor detecte, puede ser un error y el elevado coste de mantenimiento.

Y su mayor ventaja es que no le hace falta ninguna información externa (velocidad de las ruedas, etc.) para poder hallar todos estos datos.

2.3.3 Relocalización por medio de marcas

Existen dos tipos de marcas para la localización:

Activas

Se basa en la medición de las direcciones de incidencia de varias marcas emisoras. Estas marcas pueden ser paneles luminosos, transmisores de radio frecuencia, ultrasonidos, etc.

Se basa en un patrón estudiado previamente y programado para que el robot sea capaz de detectar y pueda saber la orientación y posición en ese mismo instante. Esto hace que el principal problema sea la detección de dicho patrón correctamente[7].

Pasivas

En el caso de las marcas pasivas, estas marcas emisoras se basan en la forma geométrica de las figuras u objetos acompañadas de códigos QR, colores u otras opciones.

Es importante que estas marcas las puedan identificar a pesar de las marcas similares que presente el entorno.

Dentro de estas marcas pueden clasificarse entre naturales y artificiales. Las marcas naturales están asociadas a entornos naturales como puede ser un entorno industrial o

urbano. En cambio, las marcas artificiales están diseñadas específicamente para facilitar la identificación de ellas en el entorno.

En este proyecto, se utilizan marcas pasivas artificiales para que sea identificado con mayor facilidad en el entorno exterior[7].

2.3.4 Sistema de posicionamiento global

Este sistema es el standard GPS, que se basa en la posición de tres satélites. Previamente sus relojes y receptores deben estar sincronizados para que tengan el mismo código.

Una vez que esta condición se cumpla, hay que multiplicar el tiempo de vuelo de la señal que envía el satélite y le llega al robot por su velocidad de propagación.

Gracias a esta sincronización de los tres satélites se puede hallar la posición del objeto que no siempre es exacta debido a la imprecisión de los receptores[7].

2.3.5 Relocalización mediante mapas a priori

Este sistema parte de un mapa previamente introducido en su memoria y otro mapa que reproduce a través de las señales que capta del entorno o bien, generado desde un sistema CAD.

Esto hace posible la comparación de los dos mapas pudiendo generar la posición y orientación debido a las similitudes [7].

3. ESTUDIO PREVIO

3.1 INTELIGENCIA ARTIFICIAL

La inteligencia artificial es una rama que está en desarrollo continuo, ya que abarca numerosas aplicaciones o mejoras de dispositivos más clásicos. Desde el ámbito de las finanzas y la economía hasta la medicina. A lo largo de los años ha ido evolucionando de manera muy rápido ya que al final depende de la evolución y avances de la tecnología a lo largo del tiempo[8].

Uno de los avances más significativos fue la creación del Cloud Computing (“computación en la nube”), que trabaja con datos que se encuentran en la nube y por lo tanto no hace falta una memoria física para almacenarles. Esto abre un mundo de posibilidades que impulsa al desarrollo de la inteligencia artificial, ya que se trabaja con una amplitud de datos normalmente elevada. Cuantos más datos y de mayor calidad sean, podrás tener una mejor aplicación en cualquier ámbito. Este avance revoluciono la industria, ya que abrió la posibilidad a pequeñas y medias empresas de poder trabajar sin contar con grandes infraestructuras para almacenar los datos. Y la posibilidad de trabajar con servidores de datos abiertos que podrían ser útiles para sus proyectos. Hay varios lenguajes que acompañan esta tecnología como C++, CUDA o Python. Este último es uno de los más destacados ya que se trata de un software de uso libre y sin costes. Se trata de un lenguaje de programación de alto nivel con una sintaxis muy simple, lo cual estas características son suficientes para optar por su uso[8].

Uno de sus mayores beneficios es que puede automatizar el proceso de forma “inteligente”, es decir, simula el comportamiento de un humano. Para ello se necesita crear las llamadas “redes neuronales” para que nuestra aplicación o dispositivo funcione correctamente. Estas redes funcionarán dependiendo del entorno y sus señales. Una de las técnicas más famosas de la inteligencia artificial es el Deep learning o también conocido como machine learning [9].

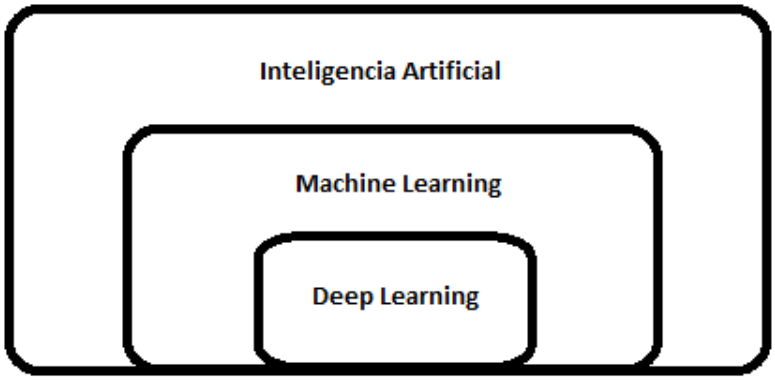


Figura 4. Esquema visual de la inteligencia artificial y lo que abarca.

El machine learning o aprendizaje automático es una de las ramas basadas en la inteligencia artificial. Se basa en no seguir unas reglas escritas y programadas por un usuario, el propio ordenador puede hacer sus propias reglas y aprender por sí mismo para realizar el proceso que desee automáticamente. Crea un algoritmo con el que se ayuda para buscar patrones similares y a partir de esto crea un modelo para poder clasificar los objetos deseados en la imagen digital. Hay muchos enfoques para implementar esta técnica de aprendizaje automático para optimizar los resultados. Se destaca tres tipos de implementación: Aprendizaje supervisado, no supervisado y de refuerzo según la naturaleza de los datos que recibe[10].

Se puede decir que hay dos fases fundamentales para el aprendizaje automático: Fase de entrenamiento en la que se aprende de los datos que se obtienen o los que se van recogiendo. Y por otro lado la fase de predicción, en la que en función de la primera fase podrá tomar una decisión u otra del modelo más adecuado para el sistema.

EL Deep learning (‘’aprendizaje profundo’’) lleva a cabo el machine learning mediante una red neuronal artificial. Esta red neuronal tiene niveles en los que cada nivel tiene un objetivo. En el primer nivel la red aprende algo y lo envía al siguiente nivel. En el siguiente nivel combina esa información con otra algo más compleja así va pasando nivel a nivel. Si por ejemplo se desea analizar la foto de un coche e identificar esa forma como “coche” en el primer nivel identifica las formas más simples como por ejemplo la circularidad de la rueda, en el siguiente nivel identifica los rectángulos que componen el coche y así sucesivamente hasta poder decir que todas esas formas y datos recogidos componen lo que sería un coche. Esto tiene una gran ventaja que es el procesamiento de un número muy elevado de datos y puede llegar a identificar formas sin ninguna etiqueta[11].

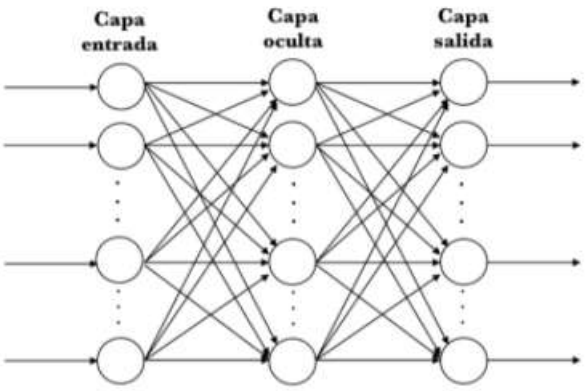


Figura 5.Estructura de una red neuronal compuesta por tres capas. (Fuente: [11])

Hoy en día, es una modalidad muy avanzada y tiene un foco de interés muy amplio para todas las empresas.

Sin embargo, resulta una parte de la ciencia más parecida al arte ya que abarca tal magnitud de datos y de procesos intermedios en los que pueden generarse de cualquier forma que se trata de un concepto más abstracto que la programación tradicional.

3.1.1 Visión artificial

Uno de los sentidos más desarrollados del ser humano es la visión, el cual se basa en interpretar constantemente el entorno analizando formas, texturas, tonos… Recibimos la luz y esta es transformada en una señal y procesada en el cerebro, el que toma la decisión correspondiente.

Ya lo dijo Arthur Brisbane: *"Usa una imagen. Vale más que mil palabras"* reflejando la importancia de las imágenes, ya que, en el ámbito científico se utiliza una amplia cantidad resultados gráficos para hallar conclusiones y observar de manera más fácil y cómoda.

La visión artificial lo que quiere es trasladar esta idea, con el fin de que un dispositivo funcione de manera autónoma gracias a procesar y analizar una serie de imágenes.

La captación de colores es una característica de percepción humana. Se posee dos tipos de fotorreceptores: los bastones y los conos. Los conos se encargan de aportar la información del color y está relacionada con el espectro visible. Dependiendo de las longitudes de onda del espectro magnético visible, se ve un color u otro. Hay tres tipos de conos con respuestas frecuenciales diferentes y que forman el código de colores RGB (red-green-blue), basado en ondas cortas (azules, 450nm), medias (verde. 540nm) y largas (rojo, 650nm).

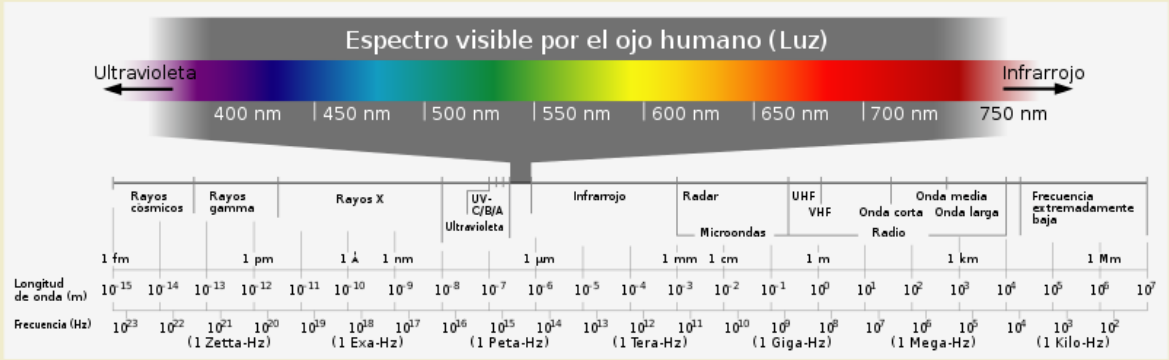


Figura 6. Longitud de onda y frecuencia del espectro visible. (Fuente: [12])

Una vez que llega la señal luminosa, se transforma en pulsos eléctricos por los conos y bastones y se transporta al cerebro por medio del nervio óptico. Al hemisferio derecho le llega la información del ojo izquierdo y viceversa. Existen zonas especiales para extraer las características de la imagen.

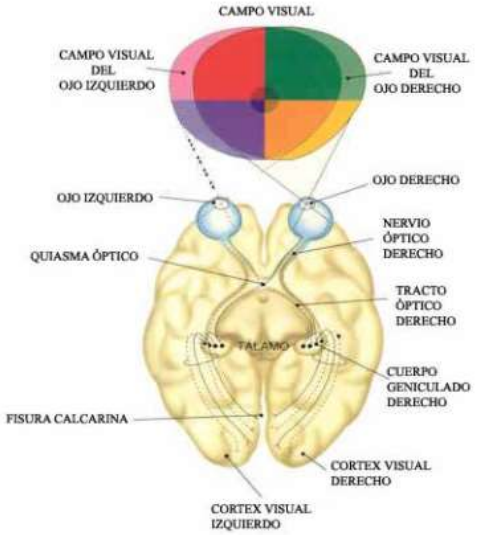


Figura 7. Cortex visual.(Fuente: [13])

En este caso, la cámara actua como un globo ocular y el ordenador procesa las señales que le lleguen. Este concepto ha ido creciendo a medida que se han ido desarrollando las cámaras, que hacen posible captar y poder visualizar los datos más relevantes que nos aportan. Por lo tanto, cuanta más calidad de imagen tenga la cámara, se obtiene un mayor número de datos sobre el entorno.

A pesar de todo lo que queda por desarrollar y los fallos que puede llegar a dar esta tecnología, en las aplicaciones que ya está implementada de manera repetitiva se obtiene unos resultados muy satisfactorios. (Ej. Determinar la trayectoria de un vehículo en una autopista, verificación de montajes, lector de pantallas…)

Se muestra una comparación de las ventajas del sistema artificial frente al sistema humano:

Sistema humano	Sistema artificial
Mejor reconocimiento de objetos	Control de calidad con medidas más exactas
Mejor adaptación a situaciones imprevistas	Rapidez y constancia en cada proceso
Utilización de conocimiento previo	Mejores resultados en tareas rutinarias
Procesamiento de tareas complejas	Menos capacidad de error

Tabla 3. Tabla comparativa de un sistema humano y uno artificial.

3.1.1.1. Etapas básicas del proceso

Adquisición de imágenes

La adquisición de la imagen se basa en un proceso mediante el cual una información luminosa 3D es proyectada en un plano 2D. La cámara actúa de la misma forma que el ojo humano como hemos explicado anteriormente[14].

En esta primera etapa se ajusta y calibra nuestra resolución de la cámara e indicar en el algoritmo cuantas capturas deseamos obtener.

Para calibrar la cámara de manera más fácil se utiliza una plantilla como puede ser el tablero de ajedrez el cual sus medidas son conocidas y se puede hallar los vectores unitarios y el vector de posición de la cámara.

Una vez configurado toda esta información, ya está listo para adquirir la imagen.

En la siguiente figura se puede ver un esquema simplificado del proceso de adquisición de imágenes:

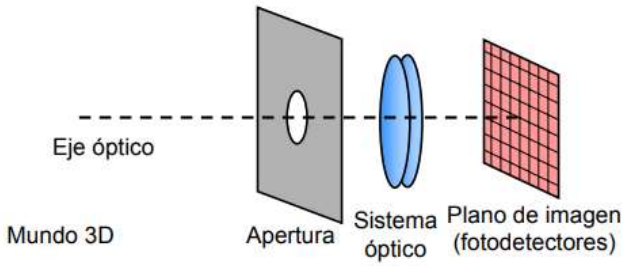


Figura 8.Adquisición de imágenes con cámara. (Fuente: [14])

En esta etapa se aprecian los tonos, intensidad y matices de la imagen adquirida en forma de una matriz numérica para el ordenador.

Etapa de preprocesado

Una vez se adquiere la imagen, pasaremos a la etapa del preprocesado. Se crea una serie de algoritmos para mejorar la imagen adquirida. Estos algoritmos son capaces de atenuar las imperfecciones y resaltar las características.

En esta etapa se indica el código de colores en el que va a trabajar, el más utilizado es el RGB (Red Green Blue) el cual utiliza tres colores primarios (rojo, verde y azul) para la captación de los colores en la imagen. Con este código se tiene varias opciones para analizar nuestra foto como por ejemplo simplemente resaltar el color rojo.

Otra opción es el formato en escala de grises. Esta opción es la más recomendable a la hora de analizar una foto.

Cada píxel en la foto tiene un valor, ya sea en formato RGB o en la escala de grises, que va desde 0 a 255. Si la imagen fuese binaria, es decir, solo aparece blanco y negro, el blanco sería 1 y el negro sería 0. Pero en el caso de la escala de grises con diferentes tonos, el blanco es 255 y el negro es 0. Y entre esos valores estan los diferentes tonos. En cambio, en el formato de colores RGB cada canal de color tendrá 255 valores.

En este proyecto esta etapa resulta fundamental ya que al ser una imagen en el exterior tiene brillos y reflejos que van a causar problemas a la hora de analizarla.

Segmentación

Esta etapa se basa en la extracción de información contenida en una imagen aplicando una serie de algoritmos.

Estos algoritmos se basan en tres propiedades básicas: Discontinuidad, la cual detecta los cambios bruscos de intensidad como los bordes, puntos más significativos, etc; Similitud que hace que se agrupen regiones con características similares y conectividad que surge cuando los puntos de cada región están conectados entre sí.

Y teniendo en cuenta estas propiedades se utilizan varios métodos para la segmentación de la imagen:

- *Umbralización*

Se basa en sacar el valor del umbral de la imagen. En la siguiente figura se muestra el histograma de una imagen y sus umbrales con respecto al objeto y el fondo:

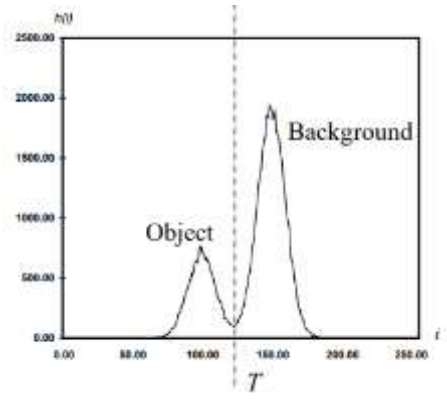


Figura 9.Ejemplo de histograma. (Fuente: [15])

Como se puede apreciar en el histograma los valores van de 0 a 255. Se ve el ruido de la imagen en el objeto y el paisaje. El valle que se forma entre ellos es el umbral óptimo(T).

La Umbralización está definida por la siguiente función[15]:

$$T = T[x, y, p(x, y), f(x, y)]$$

Donde,

$f(x, y)$ es el nivel de gris del punto (x, y).

$p(x, y)$ representa alguna propiedad local de ese punto, por ejemplo la media del nivel de gris de un punto vecino centrada en (x,y)[16].

Una imagen umbralizada está definida como:

$$g(x, y) = \begin{cases} 1 \; si \; f(x, y) > T \\ 0 \; si \; f(x, y) \leq T \end{cases}$$

Esto quiere decir que los pixeles que están asignados con 1 son objetos y los que son 0 corresponden con el fondo.

Cuando el umbral depende de $f(x, y)$,es decir, de sus valores en escala de grises, se dice que es un umbral global.

Y cuando depende de $f(x,y)\ y\ p(x,y)$, es decir, de sus valores en la escala de grises y sus píxeles vecinos, se dice que es un umbral local.

Existen varios tipos de Umbralización. La Umbralización óptima consiste en partir de la idea de que solo existen dos regiones de brillo. El histograma que se forma a partir de sus valores, se considerar como una función de densidad (p(z)). Siendo esta función la suma de dos densidades, una para regiones claras y otra para oscuras. Conociendo la forma de las funciones de densidad, se puede sacar el umbral óptimo:

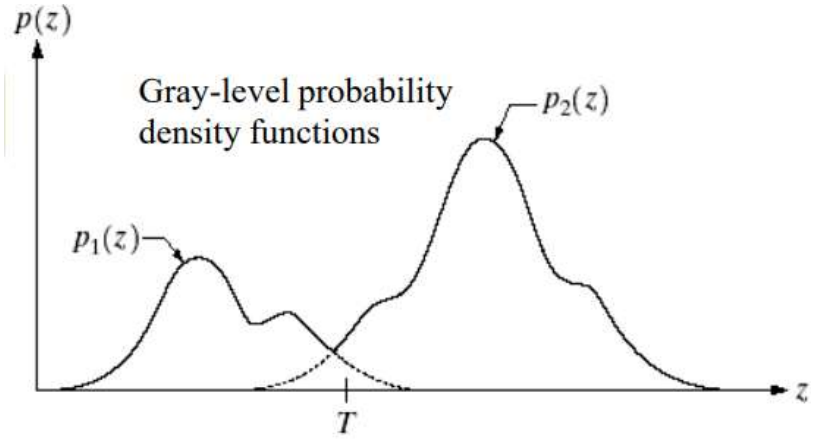


Figura 10. Función de densidad frente al nivel de grises de una imagen(Fuente: [15])

La función de densidad de probabilidad es la siguiente:

$$p(z) = P_1 p_1(z) + P_2 p_2(z)$$

Con la condicion:

$$P_1 + P_2 = 1$$

Siendo z los valores de la escala de grises, $p_1\ y\ p_2$ son las funciones de densidad de probabilidad, $P_1\ y\ P_2$ la función de probabilidad.

Para hallar el umbral óptimo:

$$P_1 p_1(z) = P_2 p_2(z); \quad z = Umbral$$

Si se considera una distribución Gaussiana:

$$p(z) = P_1 p_1(z) = P_2 p_2(z) = \frac{P_1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(z-\mu_1)^2}{2{\sigma_1}^2}} + \frac{P_2}{\sqrt{2\pi}\sigma_2} e^{-\frac{(z-\mu_2)^2}{2{\sigma_2}^2}}$$

Donde,

$\mu_1\ y\ \sigma_1^2$ son la media y la varianza de la densidad Gaussiana del objeto.

μ_2 y σ_2^2 son la media y la varianza de la densidad Gaussiana del otro objeto.

Y si $\sigma_1 = \sigma_2 = \sigma$ podremos encontrar el umbral óptimo:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma}{\mu_1 + \mu_2} \ln \frac{P_2}{P_1}$$

La Umbralización simple consiste en aplicar un umbral para eliminar las sombras y brillos y dejando solo los objetos de la foto. Como se puede ver en la siguiente demostración tenemos una imagen (a) del cual se saca su histograma (b) y se le aplica un umbral de T = 90 para segmentar la imagen. Se puede ver el resultado(c)[17].

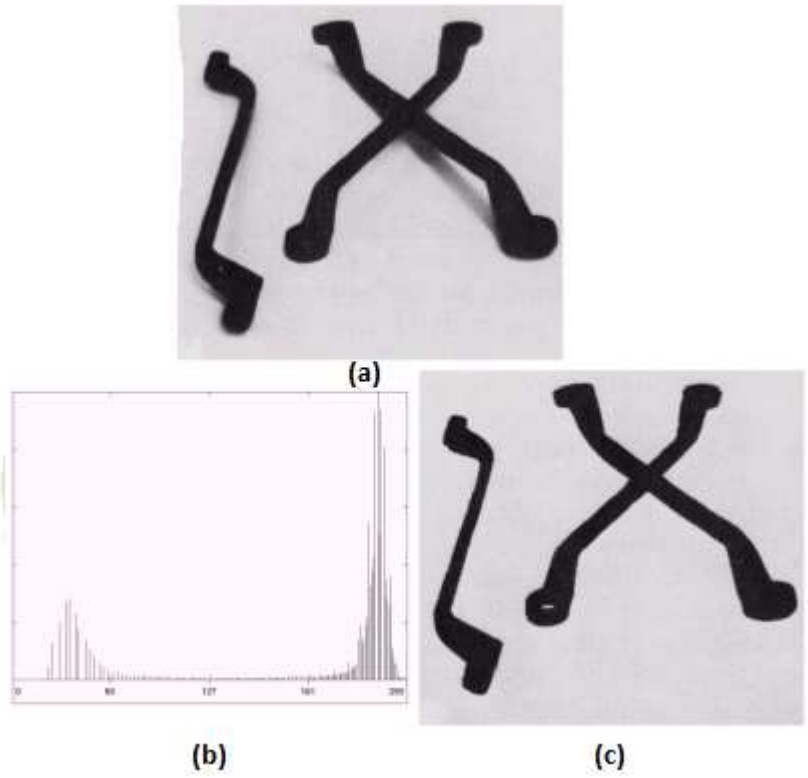


Figura 11. Imagen procesada mediante Umbralización simple(Fuente: [15])

Otro aspecto para tener en cuenta es la iluminación que tiene cada imagen.

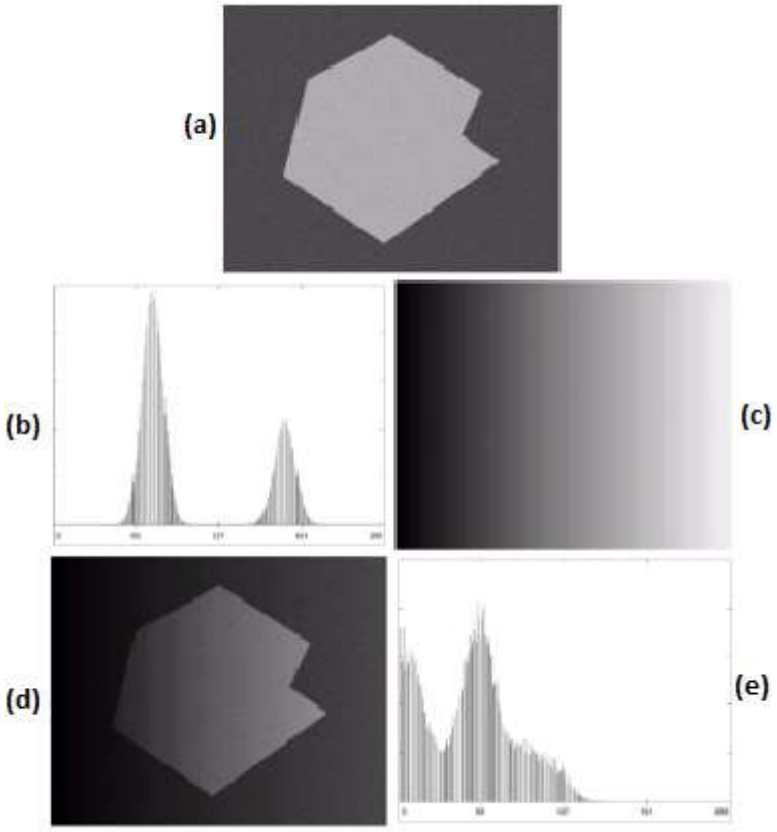


Figura 12. Impacto de la iluminación (Fuente: [15])

Se tiene la imagen original (a) y una imagen generada por el ordenador con brillos (c), se saca su histograma(b). Se juntan las dos imágenes mediante una operación matemática (multiplicando sus píxeles) y se obtiene la imagen que se puede ver(d).

Y como resultado en su histograma(e) se ve como varia con respecto al inicial. Se comprueba que la iluminación afecta mucho a la hora de analizar una imagen.

- *Agrupamiento (Clustering):*

Este método intenta separar datos que proporciona la imagen captada. Es decir, intenta dividir en subgrupos que contengan características similares. Existen tres diferentes tipos: Agrupamiento por particiones, basado en la intensidad y jerárquico.

- *Basado en modelos*

Esta técnica parte del conocimiento de las características de los objetos de la imagen como las rectas, circularidad, curvas, etc. El método más conocido se basa en la transformada de Hough, este método sirve para obtener las curvas del objeto basándose en la ecuación de la recta en forma polar. [18]

- *Detección de bordes*

Se basa en la detección de bordes en los objetos de nuestra imagen. Los bordes son conjuntos de píxeles conectados que se encuentran en el límite entre dos regiones. Esta técnica se usa normalmente en escala de grises para poder captar mejor el contraste de los píxeles vecinos para poder detectarlos. [15] Su principio de detección se puede basar en la primera derivada o en la segunda derivada.

La primera derivada (Gradiente) proporciona la información de la existencia de un borde. Analiza los píxeles vecinos y saca su gradiente teniendo en cuenta el umbral. Los métodos principales que se basan en el gradiente son: Prewitt, Sobel y Drog. A continuación, se explica el método Sobel puesto que es el que se utiliza para tratar nuestra foto:

Para calcular el gradiente en dicho método, se estudia sus píxeles vecinos, como se muestra a continuación de manera gráfica:

1	0	-1
K	0	-K
1	0	-1

*F*igura 14. Hr(i,j).

$\frac{1}{2+K}$

-1	-K	-1
0	0	0
1	K	1

$\frac{1}{2+K}$

Figura 13. Hc(i,j)

Sobel (k=2) es el doble que el operador de Prewitt (k=1). El operador Sobel calcula el gradiente de la intensidad de una imagen en cada píxel. A parte de la magnitud del gradiente, también proporciona su dirección y sentido desde el más oscuro al más claro.[19]

Es sensible a los bordes diagonales y produce un aislamiento en la imagen dejando un mayor efecto de suavidad en la imagen. Es uno de los más utilizados ya que se puede programar e implementar fácilmente en el hardware. [20]

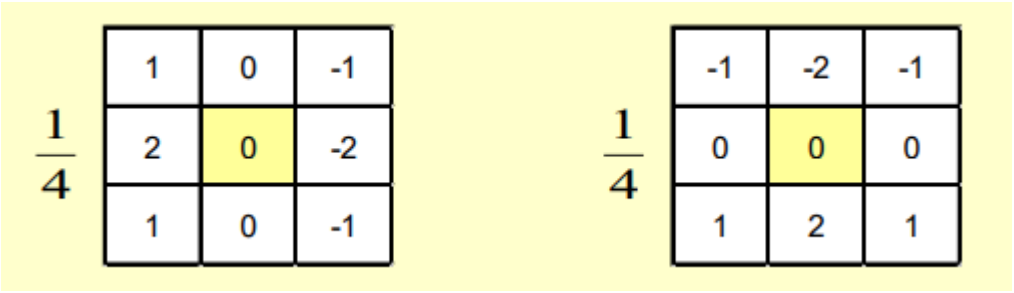


Figura 15. Máscaras del operador de Sobel con derivadas en X (izquierda) y en Y (derecha). Fuente: [20]

Previamente se calcula su umbral. Cuando se le aplica un umbral muy pequeño tiene respuestas múltiples ya que detecta las interferencias que genera el ruido. En cambio, si le aplicas un umbral muy alto pierdes información sobre los bordes en la foto.[20]

Por otro lado, el signo de la segunda derivada (Laplaciano) determina su posición.

Sabiendo su posición, se le aplica una serie de procesos para ensamblar los píxeles de los bordes en algo que tenga sentido.[15] Uno de los métodos más utilizados basados en el Laplaciano es el método Log.

Extracción de características

Es necesaria para el reconocimiento y la localización. Esto da una descripción matemática de los objetos aislados en la escena[21].

Las características que se extraen puede ser el contorno, la región o la textura del objeto. Consiste en obtener un conjunto de valores numéricos que representen al objeto. Existen varios métodos para la extracción de características:

- *Descriptores de contorno: Códigos en cadena*

Se utiliza para representar los límites a través de segmentos. Estos segmentos tienen una longitud y un número finito de direcciones posibles. Puede ser de 4 a 8 direcciones y se las asigna de la siguiente manera [21]:

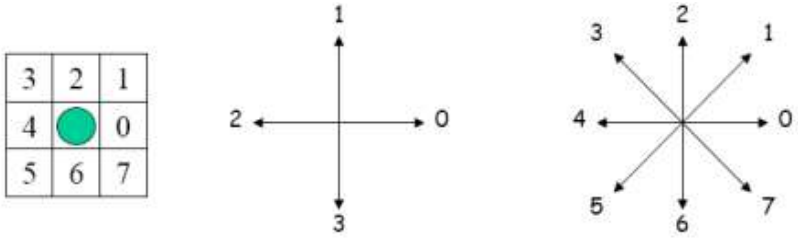


Figura 16. Códigos de cadena y sus direcciones (Fuente: [21]).

Se puede ver en la figura anterior que cada píxel tiene una dirección diferente. El problema de este método es que tiene demasiado ruido.

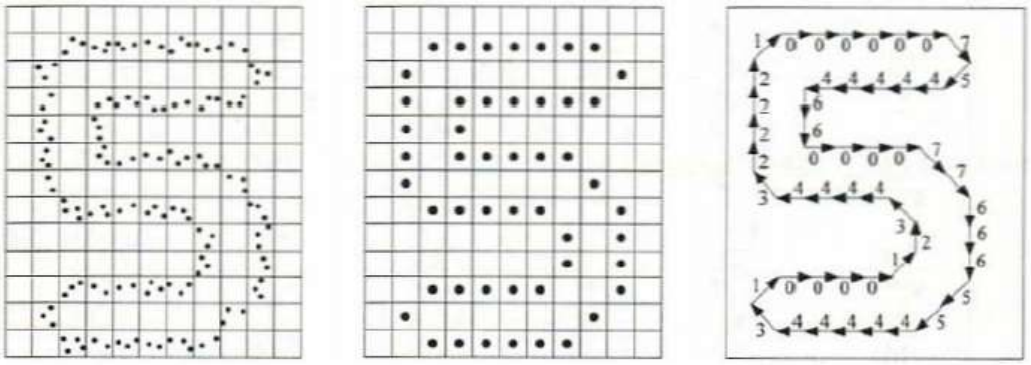


Figura 17.Ejemplo de código de cadena. (Fuente: [21])

Si conectas cada píxel con el límite se puede implementar mejor como se ve en la figura anterior.

- *Representaciones polares*

Se describe el contorno mediante el centroide del objeto. Se transforma una imagen en 2D en un vector 1D. Este vector se forma con las distancias del centroide a los puntos límite del objeto.[21]

Uno de los problemas de este método es el error que puede sufrir a la hora de calcular el centroide.

- *Aproximación poligonal*

Por la similitud de las curvas cerradas del objeto, se puede asociar a un polígono. La precisión de este método depende del número de segmentos.

Se trata de un proceso iterativo hasta que alcanza una condición para dividir la curva en varios segmentos. El algoritmo comienza con el segmento más largo que encuentre en la imagen[21].

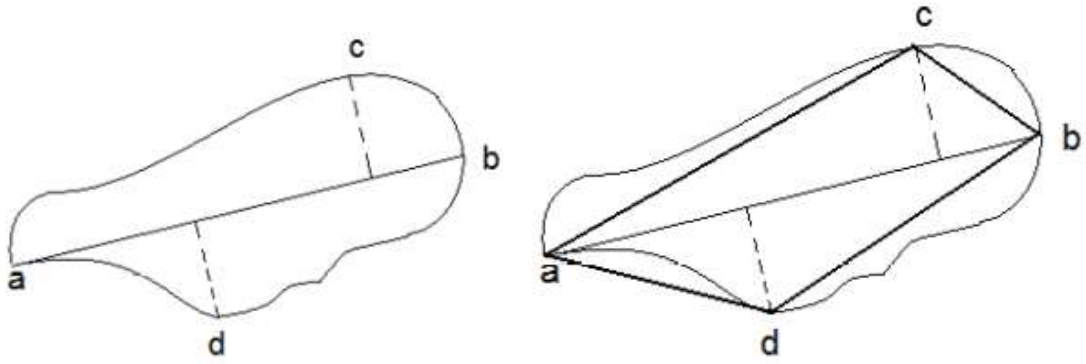


Figura 18. Ejemplo de objeto sometido a un algoritmo. (Fuente: [21])

- *Descriptores de región*

Se encuentran al analizar el contorno del objeto o por sus características propias como el área, centroide, los momentos centrales, la orientación, centro de gravedad, etc [10].

Para medir el área se utiliza el vector que define su contorno. Y se debe etiquetar previamente la imagen.

Por eso se etiqueta la foto con el comando 'label' antes de hallar su área. Este comando etiqueta la foto dando valor a cada píxel. Una vez esta etiquetada, se calcula su área.

			X	1	1	1	X			=	5
		X	1	1	1	1	X			=	6
	X	1	1	1	1	1	1	X		=	8
X	1	1	1	1	1	1	1	1	X	=	10
X	1	1	1	1	1	1	1	1	X	=	10
X	1	1	1	1	1	1	1	1	X	=	10
X	1	1	1	1	1	1	1	1	X	=	10
	X	1	1	1	1	1	1	1	X	=	9
	X	1	1	1	1	1	1	X		=	8
		X	1	1	1	1	X			=	6
			X	1	1	1	X			=	5

Figura 19. Contorno de un agujero. (Fuente: [10])

El área es el número real de píxeles en la región. Se calcula de la siguiente manera:

$$Área = \sum_{h=min(filas)}^{max(filas)} max(columna(h)) - min(columna(h))$$

Sabiendo el área, ya se puede hallar su circularidad:

$$Circularidad = \frac{4 \cdot Área \cdot pi}{Perimetro^2}$$

La circularidad es la redondez de los objetos. Para un círculo perfecto el valor de la circularidad es 1[22].

- *Transformaciones morfológicas*

Son capaces de transformar la forma y estructura de los objetos en la imagen. Todas estas transformaciones están basadas en la teoría de conjuntos. Esto permite modificar y separar los objetos del fondo y obtener contornos y formas más simples. También permite extraerlo de imágenes muy ruidosas y distorsionadas.[16]

La dilatación se puede observar en el siguiente ejemplo. Teniendo dos conjuntos A y B.

0	1	0
1	1	1
0	1	0

(a)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	1	1	1	0	0
0	0	1	0	1	0	0
0	0	1	1	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

(b)

Figura 20. Conjunto A (a) y Conjunto B (b) (Fuente: [10])

El proceso de dilatación consiste en que para cada punto de B, la máscara creada en el conjunto A se solape. Es decir, si coincide un 1 del conjunto A con el de B, coloca un punto a 1. Irá desplazando el conjunto A por toda la estructura B y si alguno de sus elementos en los píxeles vecinos coincide con un pixel del entorno es cuando se pone a 1.

A efectos visuales, lo que hace es destacar más los bordes de la foto, da una sensación de expansión de los objetos.

Para explicar la erosión se repite el mismo ejemplo anterior de los dos conjuntos A y B. Cuando se tiene una imagen B se erosiona por A cuando para todos los puntos x tales que A, trasladado por x, esta contenido en B. Esto quiere decir que la erosión pone a cero todos los píxeles de la imagen que contengan completamente al elemento en su entorno.[10]

Esta operación es la dual a la erosión, es decir, a efectos visuales se ve cómo se reduce los objetos. Se utiliza para separar objetos que están unidos de manera muy reducida por el contorno.

La Apertura se basa en realizar una erosión y después una dilatación. Esta operación se utiliza para tratar la imagen que posee mucho ruido debido al entorno. Esto hará que se separe objetos que se encuentran muy pegados entre sí.[10]

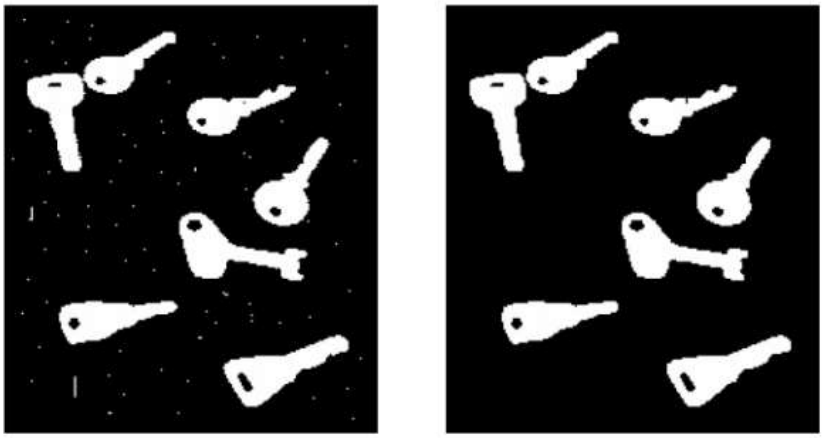


Figura 21. Ejemplo de apertura. (Fuente: [16])

El cierre se trata justo de lo contrario a la apertura. Primero se somete a la dilatación y después a la erosión. Esta técnica se usa combinándola con la de apertura para poder obtener una mejor segmentación de la foto.

Figura 22. Ejemplo de cierre (Fuente: [16])

- *Operaciones lógicas*

Estas operaciones son posibles en imágenes binarias. En niveles de intensidad se podría decir que una intensidad nula es 0, y una intensidad con valor 255 es 1.

Por esto, es posible utilizar las operaciones lógicas para tratar las imágenes, existen varias: and, or, xor, not...

En este caso se destacan las operaciones And y Or.

La operación And se realiza píxel por píxel. Se muestra su tabla de verdad para saber cómo funciona este operador:

Pixel 1	Pixel 2	Resultado
0	0	0
0	1	0
1	0	0
1	1	1

Tabla 4. Tabla de verdad del operador And.

Se muestran dos imágenes (a) y (b) binarizadas con sus valores en forma de matriz. Y la imagen (c) es el resultado de usar la operación lógica AND entre ellas.

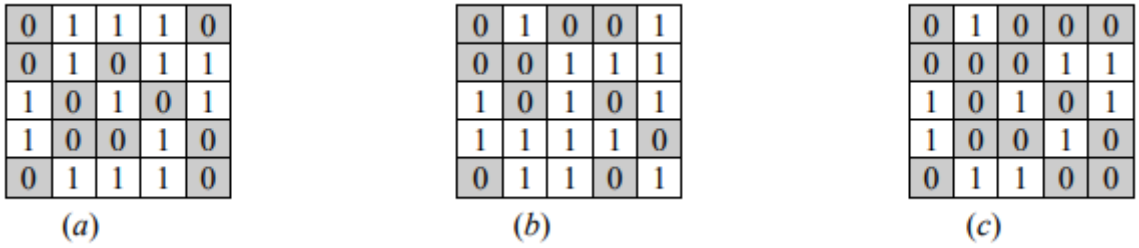


Figura 23. Imágenes binarizadas (a) y (b). Imagen resultado (c) (Fuente: [23])

De manera visual el resultado sería el siguiente:

Figura 24. Resultado visual del operador AND. (Fuente: [51])

Se muestra la tabla de verdad del operador Or:

Pixel 1	Pixel 2	Resultado
0	0	0
0	1	1
1	0	1
1	1	1

Tabla 5. Tabla de verdad del operador Or.

El resultado de aplicar el operador Or de manera visual entre dos imágenes es el siguiente:

Figura 25. Resultado del operador Or. (Fuente: [21])

Reconocimiento de objetos y clasificación

Una vez la imagen ha pasado por el proceso de segmentación y se extraen sus características, los objetos ya pueden llegar a su etapa final.

Se muestra un esquema de todas las etapas que afectan a la parte más importante: conocimiento. Aquí se almacenan todos los datos que se van extrayendo de cada etapa para poder llegar a la etapa final e interpretarlos.

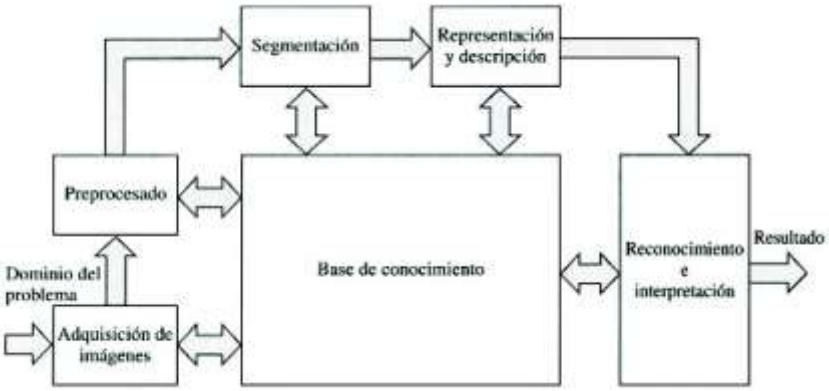


Figura 26. Etapas de la visión artificial. (Fuente: [24])

Se puede enfocar de muchas maneras la clasificación. Se va a destacar la que se ha aplicado en este proyecto.

Teniendo en cuenta que la baliza está compuesta por una esfera y una base trapezoidal.

Por un lado, se saca el área desde distintos ángulos y orientaciones ya que el robot se puede encontrar o muy cerca o muy lejos. El área es la cantidad de píxeles que hay en el objeto.

Y, por otro lado, se estudia su circularidad. Esta es adimensional. Se combinan estos dos descriptores para clasificar el objeto.

Esta fase es muy importante, ya que hay que analizar bien todas las características que se han extraído en las anteriores etapas. En este caso, la mejor opción es combinar varios datos previamente extraídos debido a la forma del objeto.

Como se muestra en la siguiente imagen es muy común realizar varios patrones para el reconocimiento de objetos.

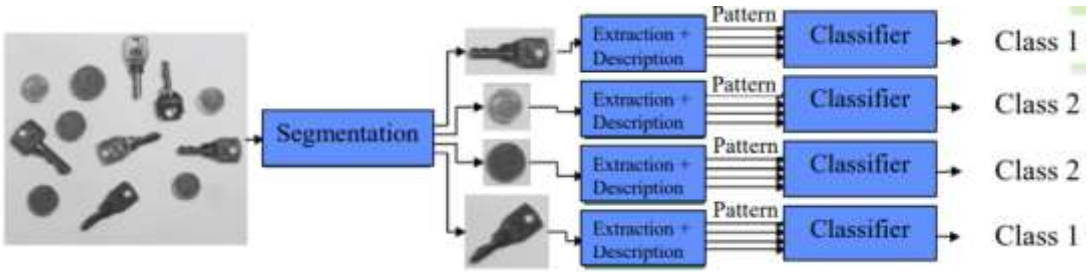


Figura 27. Proceso de reconocimiento de objetos basado en patrones(Fuente: [25]).

3.1.1.2. Elección de la baliza

Esta elección depende de varios factores:

- Estética (tamaño, color, adaptabilidad en el entorno…): Es uno de los requisitos al ser un robot comercial. Se tiene en cuenta el impacto visual en el entorno.
- Funcionalidad: Facilidad para captar y sacar sus rasgos más característicos con respecto al jardín. Para que sea un procesamiento más simple y no nos condicione a la hora de elegir el controlador.

Figura 28. Baliza.

El diámetro de la esfera es de 15 cm con un área de 706,85 cm^2. Se muestra las medidas de la base en centímetros. Es un trapecio con un área de 67,5 cm^2.

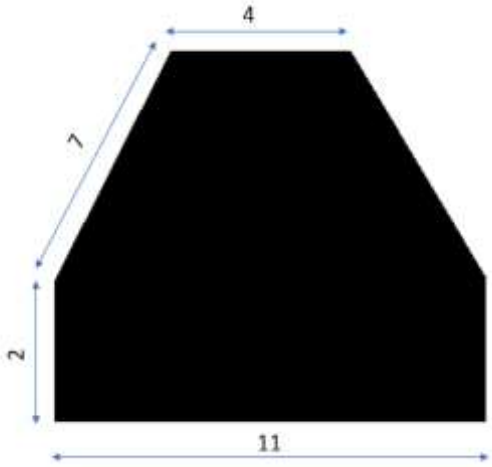


Figura 29. Medidas de la base en centímetros.

Es una baliza de color negro con una esfera y una base trapezoidal. Se trata de una marca pasiva artificial como se explica de forma más extensa en los sistemas de posicionamiento.

El color negro no causa tanto impacto visual en un jardín y se ajusta a las condiciones. La forma puede llegar a distinguirse sin problema mediante algoritmos con respecto cualquier imprevisto en el medio en el que nos encontramos.

3.2 SENSORES

Un sensor es un dispositivo capaz de recibir una señal del medio físico en el que se encuentra y generar una respuesta dependiendo de dicha señal. Esto ayuda a captar toda la información del medio y poder gestionarla y transformarla en una señal eléctrica.

Atendiendo a varias características, se puede clasificar:

- Por la naturaleza de la señal: analógicos o digitales
- Por la capacidad de influencia en el proceso en el que se encuentra: Pasivos, en el caso de que no influyan, o activos cuando consumen parte de energía en él.
- Por la forma de expresar sus parámetros: mecánicos, eléctricos, electromagnéticos… En general, los más conocidos son esta última clasificación, dependiendo de la magnitud que miden.

En este caso, se destacan los sensores que proporcionan la orientación de un robot y la distancia entre el robot y un objeto.

3.2.1 Sensores de distancia

Los sensores de distancia son capaces de calcular el desplazamiento lineal. Estos sensores se pueden clasificar dependiendo del rango de estas distancias, es decir, hay tipos de sensores que solo alcanzan la magnitud de micras y otros pueden llegar a medir hasta cientos de metros.

No obstante, también hay que fijarse en las condiciones que va a trabajar dicho sensor, si se necesita que este en el exterior sometido a humedad y condiciones no predecibles se demandará un tipo de sensor con mayor robustez.

Por cable

Este tipo de sensor es muy versátil ya que su salida puede ser analógica o digital, puede estar hecho de plástico o aluminio lo que le hace bastante ligero

Si le queremos más robusto puede llegar a ser de zinc y el montaje resulta muy fácil. [26]

Simplemente anclas el sensor en la superficie que desees medir y pones el otro extremo del cable a la superficie móvil. Una vez que la superficie se empieza a mover, crea una señal de salida que es proporcional a la distancia que recorre. Estos sensores tienen un rango de medida de entre 50 y 60000 mm

Figura 30. Sensor de distancia por cable. (Fuente: [26])

Por cinta

Son similares a los sensores de distancia por cable, con la única diferencia que en este caso el cable es una cinta de acero inoxidable. Esto da grandes ventajas, ya que aporta una mayor resistencia a las condiciones adversas y una redirección de la cinta por poleas y ejes de engranajes.[27]

Su rango de medida se encuentra entre 50 mm y 20000 mm. Mide la distancia de la misma forma que por cable, es decir, una vez se mueve la superficie móvil, este ya puede calcular la distancia proporcional que recorre

Figura 31.Sensor de distancia de cinta. (Fuente:[27])

Magnetostrictivos

EL funcionamiento de estos sensores se basa en el efecto Villary.

Dicho dispositivo se compone por un imán que genera un campo magnético y dicho campo magnético interfiere con una onda que cuando choca se refleja en la parte electrónica en la que se mide el tiempo transcurrido de ida y vuelta de la señal. [28]

Los rangos que abarcan van desde 100 mm hasta 6000mm.

También dispone de salidas analógicas y digitales. Una de sus ventajas es que pueden ser sumergidos.

Figura 32. Sensor de distancia magnetrostictivos. (Fuente:[28])

LVDT

Su funcionamiento se basa en el principio inductivo de transformador de núcleo variable. El transformador por el que este compuesto posee tres bobinas alrededor del objeto, en el que tiene un núcleo ferromagnético el cual se desplaza a lo largo de una barra en la que mediremos la distancia[29].

Cuando la corriente circula por el devanado primario provoca un voltaje que es empujado al devanado secundario. A medida que el núcleo ferromagnético se mueve por el cilindro, la inductancia que tienen en común va cambiando y hace que el voltaje en el devanado secundario varié. Las bobinas se encuentran invertidas, por esta razón el voltaje de salida es la diferencia entre los dos voltajes secundarios.

A medida que el núcleo se va desplazando, el voltaje en una bobina aumenta y en la otra disminuye, esto hace que el voltaje de salida aumente de cero a su máximo.

Lo que crea dicho sensor es una relación lineal entre el desplazamiento y el voltaje.

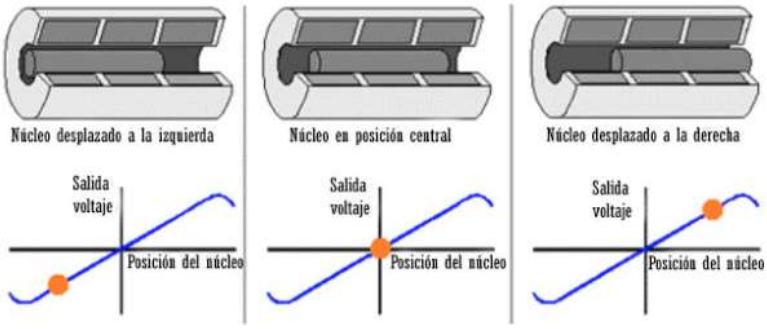


Figura 33. Relación de linealidad entre el desplazamiento y el voltaje dependiendo de la posición del núcleo (Fuente:[29])

Sus principales ventajas son: su precisión, su alta resolución, la duración de su vida útil, la resistencia a las interferencias y a los ambientes con condiciones pésimas y la facilidad para instalarlo.

Inductivos

Estos sensores se basan en las leyes de Faraday las cuales explican que un conductor de corriente podría “inducir” el flujo de corriente en otro conductor. Es decir, una corriente alterna viaja por la bobina y cuando un objeto conductivo (por ejemplo, una bola de acero) se acerque a dicha bobina, cambia el valor de su resistencia[30].

Una de sus ventajas es que puede adaptarse bien a la suciedad y la humedad, que nos dará un rango de posibilidades para poder trabajar en diferentes entornos.

Otra de las ventajas que podemos encontrar es que el circuito de procesamiento de las señales no tiene que colocarse cerca de las bobinas de detección. Esto ayuda a la hora de la instalación del dispositivo, facilitando la colocación de las bobinas en el entorno que dispongamos, que de otra manera no sería capaz de detectar en otros métodos. Estos dispositivos son muy utilizados en la industria del sector militar, aeroespacial y la industria pesada.

A pesar de sus grandes ventajas, su utilización no esta tan extendida a ámbitos más cotidianos debido a su gran volumen de instalación y a su coste.

En la siguiente figura representamos su funcionamiento (8), como podemos observar el objeto se va acercando hacia la bobina. En cuanto esté dentro del rango que abarca la corriente de la bobina, cambiará el valor de la resistencia y a partir de ahí podemos hallar la distancia entre ambos.

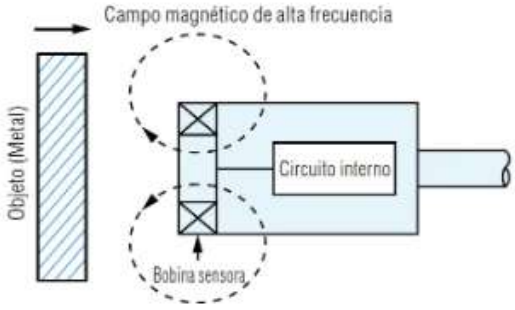


Figura 34.Esquema de funcionamiento de un sensor inductivo. (Fuente:[30])

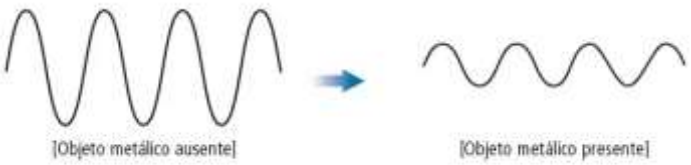


Figura 35. Representación de la forma de onda de la corriente en las dos situaciones. (Fuente: [31])

Láser

Este sensor cuenta con salidas analógicas y digitales contando con interfaces como ethercar, ethernet o profibus. Es una buena opción para diversas aplicaciones ya que abarca un rango muy amplio desde 5 mm hasta cientos de metros con el único inconveniente de que la superficie a tratar deba estar limpia y con pocas interferencias en el terreno que puedan complicar las mediciones[32].

Su sistema de funcionamiento se basa en emitir una señal láser al objeto, la cual se refleja y vuelve a nuestro dispositivo. Debido al tiempo transcurrido que ha pasado en dicho proceso, el controlador con una simple fórmula calcula la distancia que hay entre el dispositivo y el objetivo marcado[32]:

$$d = \frac{1}{2} \cdot c \cdot t$$

Donde c es la velocidad de la luz y t el tiempo transcurrido que tarda el láser en ir y volver al dispositivo.

En la siguiente figura (10) tenemos una representación de los pulsos que emite el sensor y el pulso que recibe. Ese periodo de tiempo que transcurre entre las dos ondas es T.

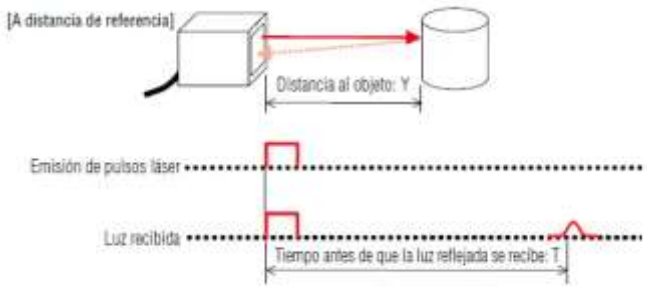


Figura 36.Esquema de funcionamiento del sensor de distancia láser. (Fuente:[32])

Infrarrojo

Su funcionamiento es muy similar al del sensor ultrasonidos que explicaremos más adelante.

Emite una luz infrarroja a altas frecuencias, el objeto la recibe y rebota en él y se lo devuelve al receptor del dispositivo.

Uno de sus inconvenientes es que puede llegar a falsear el dispositivo con la luz solar, ya que emite este espectro, por eso su poco uso en las aplicaciones.[33]

Su rango de medida abarca entre 0.5 cm a 200cm.

El rango aumenta o disminuye según el tamaño de la lente que emite dicha luz. En la siguiente figura (11) tenemos una representación del funcionamiento del sensor:

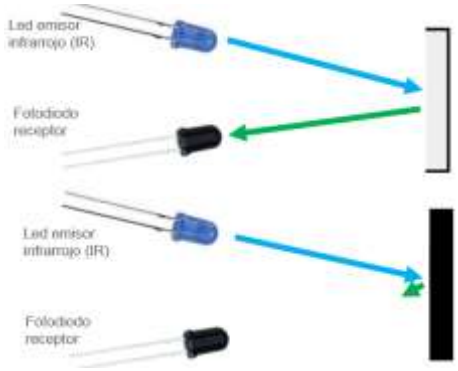


Figura 37.Funcionamiento sensor Infrarrojos. (Fuente:[33])

Como vemos en la figura (11), si la luz no rebota y va hacia el receptor es porque no ha encontrado dicho objeto o porque el objeto es de un color el cual ha absorbido el espectro de luz.

Sensor ultrasonidos

En el proyecto se usará este tipo de sensor.

Su funcionamiento se basa en mandar una señal ultrasónica y que este rebote contra el objeto y vuelva a llegar al sensor en forma de eco. Una vez la onda ha llegado, se calcula el tiempo que ha tardado en realizar este proceso. Cuando tenemos el tiempo transcurrido, podemos calcular la distancia con la siguiente fórmula:

$$d = \frac{1}{2} \cdot v \cdot t$$

Donde, v es la velocidad del sonido en el aire (340 m/s), t es el tiempo transcurrido y, por lo tanto, d es la distancia total. Lo multiplicamos por $\frac{1}{2}$ ya que la onda hace un camino de ida hasta el objeto y otro de vuelta. Por lo tanto, su distancia real sería la mitad de ese valor.

Las ondas ultrasónicas son aquellas que existen en una frecuencia de 40KHz, las cuales no podremos llegar a escuchar debido a nuestro límite auditivo que va de una frecuencia de 16Hz a 20Khz.

En la siguiente figura (12) se muestra como la onda que transmite rebota con el objeto y vuelve al sistema receptor.

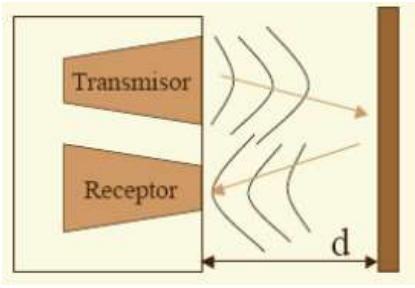


Figura 38.Funcinamiento del sensor ultrasónico. (Fuente: [34])

3.2.2 Sensor inercial

Un sistema de medición inercial permite conocer la orientación del dispositivo gracias a las fuerzas inerciales que experimenta este. Este tipo de sensores son fundamentales en los sistemas de navegación de barcos, aviones, helicópteros, satélites o incluso en los móviles.

Su principio de funcionamiento se basa en la medida de las fuerzas de aceleración y velocidad angular sobre las masas que se encuentran en el interior de este dispositivo. Se caracteriza por aplicar las dos primeras leyes de Newton[35]:

- La Ley de inercia se basa en la primera Ley de Newton, si sobre un cuerpo no actúa ninguna fuerza externa, este permanecerá moviéndose en línea recta con velocidad constante.
- La segunda ley de Newton o Principio fundamental establece que la aceleración que experimenta un cuerpo es proporcional a la fuerza que recibe, es decir:

$$F = m \cdot a$$

Con la combinación de estas dos leyes se puede hallar su aceleración, velocidad y posición a partir de su magnitud y dirección de la fuerza que aplicas a dicho dispositivo.

Este sensor está compuesto por un conjunto de acelerómetros y giróscopos con el fin de obtener los datos de uno o más ejes ortogonales.

El acelerómetro permite transformar las vibraciones en una señal eléctrica que será proporcional a la aceleración del objeto. Este hecho hace que la carga sea también proporcional a la aceleración.

Por un lado, será capaz de medir la aceleración gravitacional estática en la que se obtiene el ángulo de desviación que tiene nuestro dispositivo en la vertical y por otro lado la aceleración gravitacional dinámica gracias a que detecta las vibraciones de baja amplitud y frecuencia (golpes, movimientos, impactos…)[36].

Hay varios tipos de acelerómetros:

- *Capacitivos*: Están compuestos por un sistema de suspensión y una masa de prueba cuya posición y distancia entre los elementos nos dará los datos de aceleración[36].En la siguiente figura (14) se muestra un ejemplo:

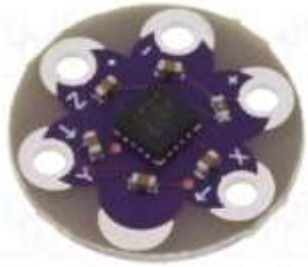


Figura 39.Acelerómetro capacitivo MEMS. (Fuente: [37])

Una de sus mayores ventajas es la posibilidad de implementación directamente en la placa con el circuito impreso.

- *Piezorresistivos*: Están compuestos de material piezorresistivo que, al encontrarse sometido a una fuerza externa, se deforma y produce un cambio en la resistencia. Y esta resistencia se transforma en una señal eléctrica[38]. Su mayor ventaja es que tiene un gran rango de mediciones. En la siguiente figura se muestra una representación de este mismo:

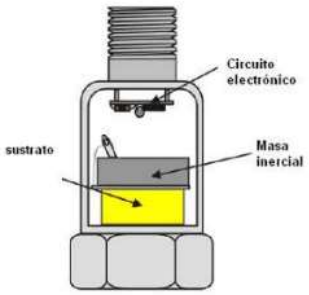


Figura 40.Acelerómetro piezorresistivo. (Fuente: [38])

- *Piezoeléctricos:* Está compuesto por una masa inercial, un piezoeléctrico de cristal y su carcasa. Cuando se le ejerce una fuerza sobre la carcasa como se puede ver en la siguiente figura(16), varia su estructura cristalina y produce una señal eléctrica[38]. Su mayor ventaja es la generación de un valor de tensión proporcional aplicada en la salida del amplificador y es independiente del conexionado que tenga en el exterior, por lo tanto, no necesita nada más que el sensor, obteniendo así una salida más cómoda. En la siguiente figura se muestra un esquema de este cuando se le aplica una fuerza vertical:

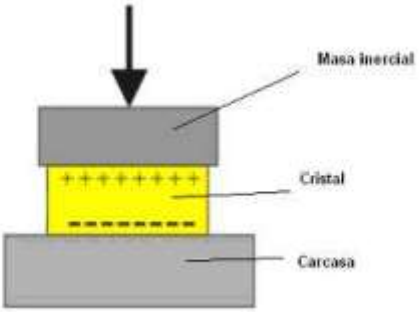


Figura 41.Acelerómetro piezo-eléctrico. (Fuente:[38])

- *Mecánico:* Está compuesto por una masa, resortes elásticos y una galga extensométrica. Mediante un puente Wheatstone se detecta la variación en la corriente eléctrica y se halla la deformación en la galga, la cual es directamente proporcional a la aceleración aplicada[38]. En la siguiente figura se ve una representación de este:

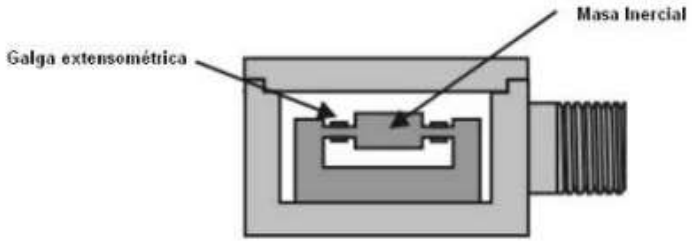


Figura 42.Acelerómetro mecánico.(Fuente: [38])

A parte del acelerómetro, también está compuesto por un giróscopo.

El giróscopo es una herramienta que permite medir el movimiento del objeto y además generar una respuesta para poder modificar dicha dirección. Tiene una gran serie de aplicaciones, desde detectar si el dispositivo se está cayendo al suelo hasta aplicaciones militares para orientar a los proyectiles.

Su principio de funcionamiento se basa en el efecto giroscópico[39].

Consiste en que una masa gire sobre su propio eje de simetría siendo este eje móvil o no. Este efecto se explica fácilmente si se piensa en el funcionamiento de una peonza o rueda de bicicleta. Al girar una rueda de bicicleta a gran velocidad sin estar anclada al vehículo, se observa que esta gira perpendicularmente al suelo hasta que su velocidad va disminuyendo y por lo tanto es cuando empieza a tambalearse y finalmente caerse.

Por lo tanto, este dispositivo lo que hace es calcular la velocidad angular y poder hallar su orientación. Una vez hecho esto va calculando las variaciones que se van produciendo y los pequeños cambios para poder estabilizar el objeto[40].

Existen varios tipos:

- *Electrónico*: Su principio de funcionamiento está basado en la fuerza de Coriolis[41]. No es una fuerza como tal, pero desde el punto de vista de un observador físico se ve que el objeto que está rotando sobre el radio de un disco en rotación es tambaleado por una fuerza ficticia, que en este caso sería la aceleración relativa. Esta aceleración es perpendicular al eje de rotación del sistema y a la velocidad del cuerpo.
- *Analógico:* Tiene el mismo funcionamiento que el anterior con la gran diferencia que devuelve los datos de manera analógica a la aplicación y por lo tanto se adapta mucho mejor a cantidad de aplicaciones industriales[41].

Esta fuerza tiene dos componentes: un componente tangencial y otro componente radial. Y se puede hallar de la siguiente forma[41]:

$$\overrightarrow{F_{Coriolis}} = -2 \cdot m \cdot (\vec{\omega} \times \vec{v})$$

Donde:

m: masa del cuerpo.

v: velocidad del cuerpo en el sistema en rotación.

ω : velocidad angular del sistema en rotación vista desde un sistema inercial.

Una vez hallas la fuerza, se puede determinar la rotación en el giroscopio.

3.2.3 Sensor electrónico: Magnetómetro

El magnetómetro es un dispositivo capaz de medir o cuantificar la fuerza de una señal magnética.[42]

Este dispositivo mide la densidad de flujo magnético que genera el campo magnético en la atmósfera. A la hora de estar rodeado de materiales magnéticos, es capaz de detectar la cantidad de distorsión que causan estos materiales en el campo de la Tierra.

También puede medir la fuerza de los campos magnéticos. Esta información se puede utilizar para hallar la dirección, rotación y ángulo de los campos magnéticos.

Hay dos tipos de magnetómetros[42]:

- Magnetómetro escalar: Mide la intensidad del campo magnético al que está sometido.
- Magnetómetro vectorial: Mide las propiedades del campo magnético que circulan en una determinada dirección.

3.3 RASPBERRY PI

La Rasberry Pi tiene sus orígenes en Reino Unido en el año 2009, inicialmente el objetivo de esta placa era el mismo que la de Arduino. Se define como un ordenador, ya que sus componentes son prácticamente iguales que los de una torre de control con diferencia de que no tiene botón de apagado y encendido. Para poder controlarle de manera más cómoda se conectan a sus periféricos la pantalla, el teclado y el ratón[43].

Puede ser utilizada para navegar por internet, procesar textos, ser implementada para la automatización o incluso jugar a videojuegos[44].

3.3.1 Componentes

Para que todo esto sea posible se compone por un System on Chip (SoC) el cual es un ordenador integrado en un solo chip. En el SoC se suele incluir el procesador, la tarjeta gráfica, la de sonido, etc. En el resto de la placa suele ir lo que no se puede incluir aquí como por ejemplo los pines (GPIO), el módulo wifi, bluetooth… Además de esto, cuenta con ranura para SD y conexiones para periféricos de bajo nivel

3.3.2 Pines GPIO

General Purpose Input Output (GPIO) es un sistema de entrada y salida de propósito general. Como su propio nombre indica, son pines que se pueden configurar para realizar distintas funciones, de ahí que sean de propósito general y no para un uso específico. Será el usuario en tiempo de ejecución el que pueda configurar estos pines GPIO para que hagan lo que él quiere. Se puede hacer de diferentes formas, como con ciertos códigos o scripts desde la consola o con el programa Python como bien citan en [45].

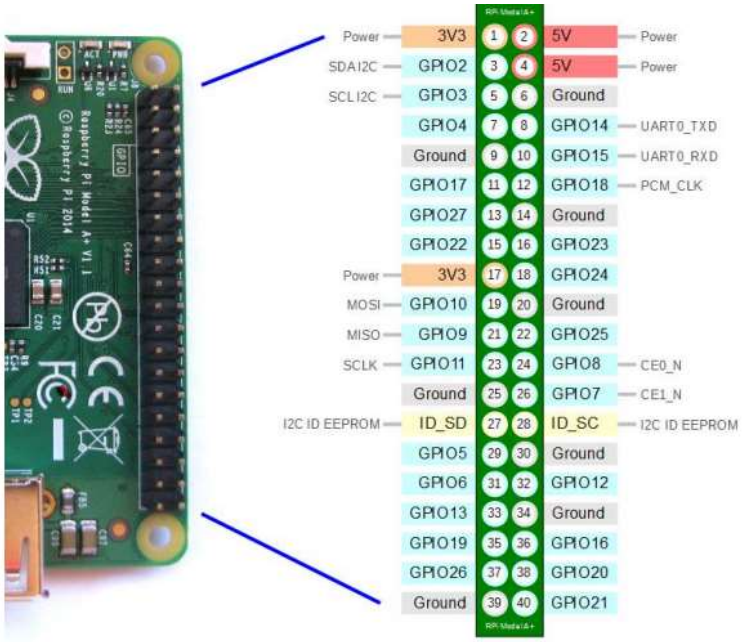


Figura 43.''Raspberry Pi: GPIO'' [[45]].

Como se puede observar, está compuesta por:

- Alimentación: Dos pines de 5V y otros dos de 3.3V (limitados a 50mA).
- Tierra (GND): cinco pines.
- Pines digitales: En azul tenemos 17.
- SDA I2C: GPIO2 para recoger datos mediante comunicación I2C.
- SCL I2C: GPIO3 la señal de reloj.
- No cuenta con ningún tipo de protección a la hora de un sobretensión o cortocircuito.

3.3.3 Comparativa de modelos

Se reflejan los modelos más destacados hasta noviembre de 2018 con sus características más significativas[46].

MODELO / CUALIDADES	1B	2B	B+	Zero	3B	3B+
SOC	Broadcom BCM2835	Broadcom BCM2836	Broadcom Bcm2835	Broadcom BCM2835	Broadcom BCM2387	Broadcom BCM2837B0
RAM	512 MB	1GB	512 MB	512Mb	1GB	1GB
Memoria	Micro SD	Micro SD	Micro SD	Micro SD	Micro SD	MicroSD
USB	2	4	4	1	4	4
Ethernet	Si	Si	Si	No	Si	Si
Wi-fi	No	No	No	No	Si	Si
Bluetooth	No	No	No	No	Si	Si
HDMI	Si	Si	Si	Mini	Si	Si
CÁMARA	Si	Si	Si	Si	Si	Si
ALTURA	85.6 mm	85.6mm	85 mm	65mm	85mm	85mm
ANCHO	56mm	56.5mm	56 mm	30 mm	56mm	56mm
PROFUNDIDAD	17mm	17 mm	17 mm	5 mm	17mm	17mm
PESO	45 gr	45 g		9 g	45g	45 gr
CONSUMO	700mA	820 mA		350 Ma		
PRECIO	33€	39€	26€	6€		

Tabla 6. Comparativa de modelos de Raspberry PI hasta noviembre de 2018. (Fuente: [47])

3.4 ARDUINO

Es una plataforma de creación de electrónica que tuvo su origen en el año 2005, con el objetivo de poder aprender electrónica de una forma más barata y con una amplia gama de proyectos a realizar. Cuenta con un entorno de programación llamado Arduino IDE con su propio lenguaje de programación[48].

Sus principales ventajas son las siguientes:

- Software y Hardware libre, por lo que para la comunidad es una gran ventaja para poder aprender de los proyectos ya realizados. El Software libre se refiere a los programas cuyo código es accesible para cualquiera. Y el Hardware libre son los dispositivos cuyas especificaciones y diagramas son de acceso público.
- Respuesta instantánea al ejecutar el código e implementarle en la placa. No hace falta la instalación previa de un sistema operativo.

- Puedes conectar cualquier periférico de entrada o salida externo a la placa, esto le da una gran versatilidad.
- Bajo coste.
- Opción de ensamblar a mano o encargarlas preensambladas.
- Compatibilidad con otros modelos de placas que no son Arduino original.

Y sus desventajas:

- Problemas de espacio en los pines a la hora de conectar los periféricos de entrada y salida.
- No sirve para proyectos complejos en los que se requiera un control autónomo con unos cálculos complejos previos a la ejecución del código.

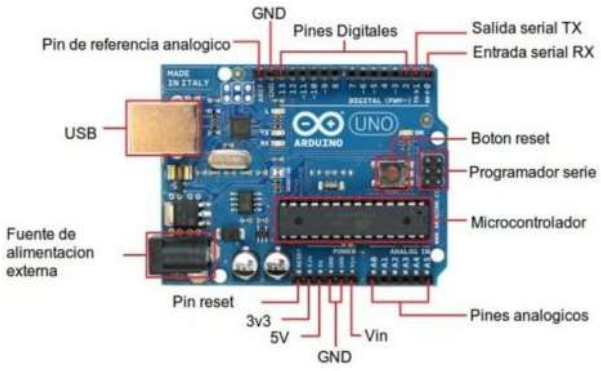


Figura 44.Ejemplo de la placa Arduino UNO (Fuente: [49])

Como se puede ver es una placa bastante completa para el bajo coste y las ventajas que ofrece. Se compone de[48]:

- Pines digitales.
- Pines analógicos de entrada y de salida.
- Fuentes de alimentación: 3.3V, 5V, externa.
- Puertos de comunicación: I2C, SPI, USB.
- Tres tipos de memoria: SRAM donde se encuentran las variables que creas y ejecutas, EEPROM donde se trata de una memoria no volátil en caso de que se apaga o se reinicie la placa y memoria FLASH donde se almacena el programa
- Microcontrolador de ATMega328 (todos los microcontroladores que lleva Arduino o placas compatibles son fabricado por ATMEL), en él se encuentran las memorias nombradas anteriormente que sirven para poder ejecutar el programa que se introduce desde el ordenador. Y gracias a él es capaz de procesarlo.

Existen multitud de placas compatibles con Arduino que utilizan su entorno y sus componentes.

3.5 PROTOCOLOS DE COMUNICACIÓN

Es una herramienta que permite que dos o más entidades se puedan llegar a comunicar entre sí y puedan transferirse datos. Se puede implementar por Hardware o Software depende de las necesidades y de las especificaciones de los componentes.

A continuación, se muestra un esquema de los distintos protocolos que existen:

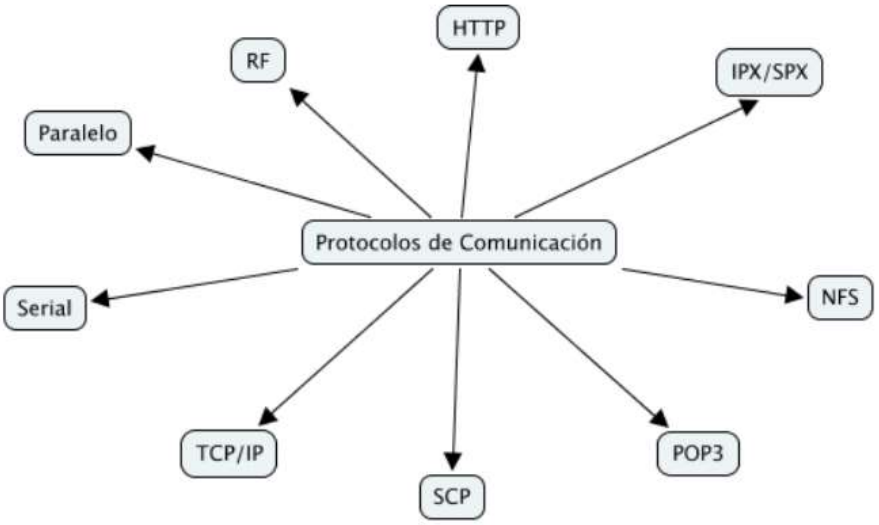


Figura 45.Esquema de los tipos de protocolos de comunicación. (Fuente:[50])

En este caso, se destaca el protocolo serie y el protocolo I2C, ya que, es el que se utiliza para comunicar el sistema embebido y el sensor[51].

3.5.1 Puerto serie

Se denomina puerto serie debido a que los datos se transfieren en forma de bits uno a uno en serie. Para ello se necesita dos dispositivos y un cable para poder conectarse a los puertos serie.

Todos los ordenadores tienen incorporado un puerto serie para realizar dicha comunicación siendo regulado por la norma RS-232-C.

La transferencia de datos puede ser de dos tipos:

- Asincrónica: Se requiere una señal del emisor antes de transferir cualquier dato. Su mayor ventaja es la fiabilidad.

- Sincrónica: El emisor y el receptor utilizan la misma señal de reloj cuando transfieren los datos. Su mayor ventaja es la velocidad de transmisión[51].

3.5.2 Protocolo I2C

Este protocolo puede llegar a conectar 127 dispositivos esclavos.

Es de los más utilizados para conectarse con sensores digitales ya que dentro de su trama permite tener la opción de confirmación de datos recibidos. Se trata de un puerto usado para la transmisión de datos de forma bidireccional.

Este puerto está compuesto por dos líneas: una de datos seriales (SDA) y otra de reloj serial (SCL). Para que funcione correctamente deberían de conectarse dos resistencias pull-up en cada línea.

Este tipo de conexión se diferencia un maestro y un esclavo. El maestro se encarga de controlar la línea que va al reloj serial para iniciar y parar la comunicación. Por la otra línea SDA se envía la información binaria serial funcionando como maestro-transmisor o maestro-receptor[52].

3.6 SOFTWARE

3.6.1 Matlab

Matlab se trata de un software matemático para analizar y diseñar sistemas. Cuenta con un entorno Desarrollo integrado (IDE) con un lenguaje propio (Lenguaje M) y multiplataforma. Se caracteriza por ser un lenguaje sencillo, pero a la vez potente y rápido. Es el más utilizado en la Universidad de Cantabria para llevar a cabo el control de plantas o aplicaciones analizando sus datos[53].

A parte de escoger este software por la facilidad y el conocimiento sobre él, cuenta con multitud de funciones como puede ser la presentación gráfica de datos, implementación de algoritmos, creación de aplicaciones para el usuario (GUI), comunicación con programas en diferente lenguaje y con otros dispositivos, entre otras muchas[50].

Una de sus ventajas es que permite actuar como usuario o como programador, proporciona una variedad muy amplia de acciones para cualquiera de los dos. En este caso, desde el programador, puedes utilizar muchas de sus librerías para ayudarte a conseguir los datos

deseados con tal solo una función sin necesidad de implementar un programa demasiado extenso[53].

Se utiliza bastante las funciones que tiene para trabajar con la visión artificial y por otro lado las cajas de herramientas que proporcionan ellos mismos para permitir la conexión entre dispositivos.

3.6.2 Arduino IDE

Se trata de un software creado por la compañía Arduino para programar sus placas y sistemas embebidos. Es un entorno IDE basado en el entorno de Processing y su lenguaje de programación fundamentado en Wiring[48].

La gran ventaja de este software es que es gratuito y tiene licencia de Código abierto a diferencia de Matlab, lo cual, hace posible que se puedan compartir los programas y todo el mundo tenga acceso a ello de una manera fácil[48].

4. IMPLEMENTACIÓN DEL SISTEMA

4.1 DISEÑO DEL CIRCUITO

El diseño final se compone por un lado por el sensor ultrasonidos HC – R04 conectado a la Raspberry Pi. Y, por otro lado, un sensor inercial MPU6050 conectado a una placa Arduino UNO. La placa que mejor se adapta a nuestras condiciones es la placa Arduino UNO con todas las características que se explican en el estudio previo. A continuación, os muestro el montaje del circuito:

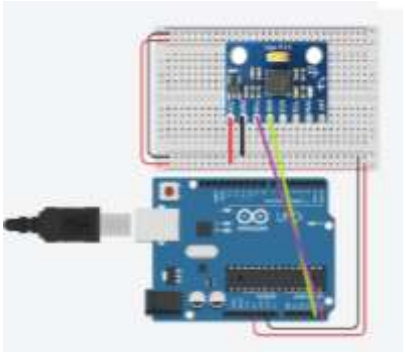


Figura 46. Conexiones Arduino - Sensor IMU.

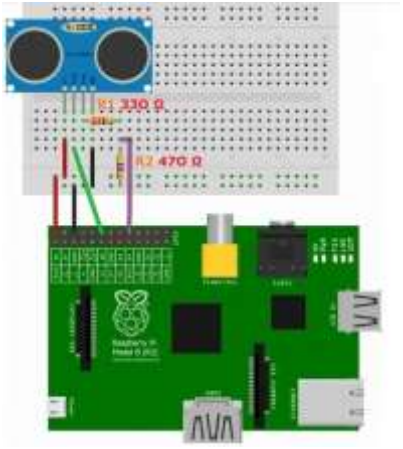


Figura 47. Raspberry Pi - Sensor ultrasonidos. (Fuente: [56])

Estos dos circuitos se relacionan desde el software de Matlab que permite combinar dos circuitos con placas diferentes en un mismo Script.

4.1.1 Sensor ultrasonidos: HC – SR04

El modelo de sensor ultrasonidos que más se ajusta a las necesidades del proyecto es el siguiente:

Figura 48. Sensor ultrasonidos Modelo HC-SR04. (Fuente: [54])

Está compuesto por cuatro salidas como se puede ver en la figura (13):

- Vcc: Pin para alimentar el sensor, se alimenta a 5V.
- Trigger: Se encarga de disparar la señal ultrasónica.
- Echo: Se encarga de recibir la señal que devuelven los objetos.
- Gnd: Tierra.

Este sensor puede alcanzar una distancia de hasta 4 metros.

Para conectarlo a la Raspberry Pi se debe introducir dos resistencias. Esto se debe a que la Raspberry Pi no aguanta más de 3,3V en los pines GPIO.

Una de las ventajas de este sensor es su bajo coste y su bajo consume (Entre 2 y 15 mA)

La finalidad será calcular la distancia al objeto. Para calcular la distancia se sabe que la velocidad del sonido es 340 m/s a una temperatura de 20 ºC, al 50% de humedad y presión atmosférica a nivel del mar. Se pasa la velocidad a las unidades de lectura del sensor:

$$340\frac{m}{s} \times 100\frac{cm}{m} \times \frac{1\,s}{1000000\,\mu s} = \frac{1}{29}\frac{cm}{\mu s}$$

Es decir, el sonido tarda 29 microsegundos en recorrer un centímetro. Por lo tanto, la distancia en centímetros se calcula:

$$Distancia\ (cm) = \frac{Tiempo\ (\mu s)}{29 \cdot 2}$$

También hay que tener en cuenta las limitaciones recomendadas por el fabricante[55]:

4.0 PRODUCT SPECIFICATION AND LIMITATIONS

Parameter	Min	Typ.	Max	Unit
Operating Voltage	4.50	5.0	5.5	V
Quiescent Current	1.5	2	2.5	mA
Working Current	10	15	20	mA
Ultrasonic Frequency	-	40	-	kHz

Figura 49. Especificaciones y limitaciones dadas por el fabricante. (Fuente: [55])

4.1.2 Sensor inercial: IMU MPU-6050 – Módulo GY-521

El modelo que más se ajusta a las necesidades del proyecto es el siguiente:

Figura 50. Sensor IMU MPU-6050 -Módulo GY 521. (Fuente: [56])

Sus dimensiones son 4x4x0.9 mm.

Tiene seis grados de libertad, es decir combina un acelerómetro de tres ejes y un giróscopo de tres ejes

Está compuesto por un acelerómetro, giróscopo y un sensor de temperatura. Este tipo de dispositivos tiene integrado un módulo GY-521 para facilitar la conexión con Arduino[57].

A continuación, en la figura 18 se muestran los ejes y sus rotaciones correspondientes respecto a ellos:

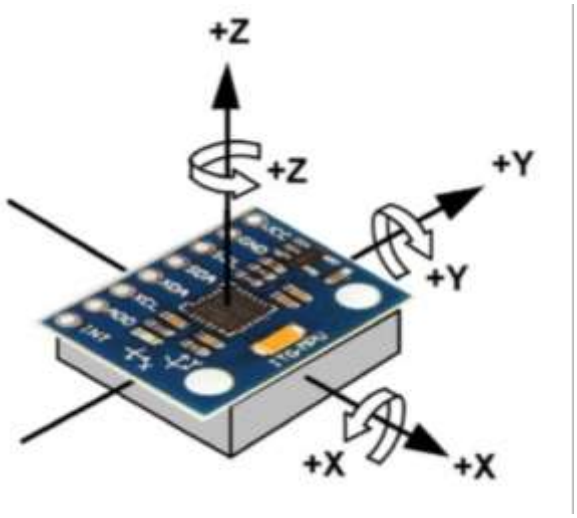


Figura 51. Ejes y sus respectivas orientaciones. (Fuente: [57])

El acelerómetro proporciona su aceleración. Suponiendo que el sensor está colocado en paralelo al plano horizontal X-Y con el eje Z apuntando hacia arriba. Con esta aceleración se miden los ángulos de orientación. Estos ángulos se pueden hallar con el método de la tangente[58]:

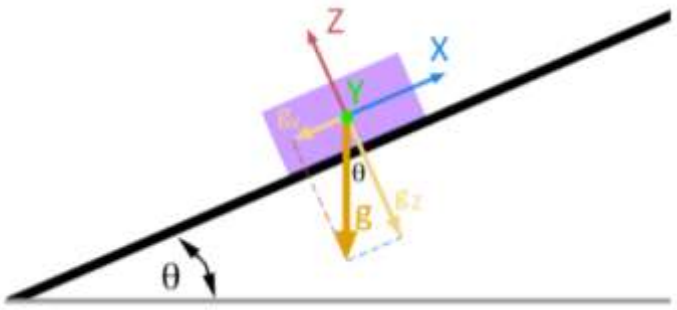


Figura 52. Ángulo en el plano inclinado (Fuente: [58])

$$\theta = atan\frac{A_x}{A_y}$$

Se aplica en cada eje:

$$\theta_x = atan\frac{A_x}{\sqrt{A_y^2 + A_z^2}}$$

$$\theta_y = atan\frac{A_y}{\sqrt{A_x^2 + A_z^2}}$$

$$\theta_z = atan\frac{\sqrt{A_x^2 + A_y^2}}{A_z}$$

El inconveniente es que necesita otro método o filtro para que pueda eliminar los errores de deriva. Para solventar este error se aplica un filtro, pero en el eje Z no es posible aplicarlo. Por lo tanto, el método de la tangente no es fiable a corto plazo.

El giróscopo permite conocer un ángulo de rotación relativo con respecto a una referencia arbitraria para conocer el ángulo absoluto. Permite saber la variación de la rotación en forma relativa.

Al ser mediciones relativas, los errores de deriva se van acumulando y la precisión es bastante mala. Para ello se calcula el ángulo de la siguiente manera:

$$\theta_{gyro} = \omega_{gyro} \cdot \Delta t$$

Hay diferentes formas de medir las inclinaciones, pero la más utilizada es el filtro complementario o filtro Kalman.

Para solventar este problema se usa el filtro complementario. Se calcula el ángulo de la siguiente manera cuando se le aplica el filtro complementario:

$$\theta = 0{,}98 \cdot \left(\theta + \theta_{gyro} \cdot \Delta t\right) + 0{,}02 \cdot \theta_{Accel}$$

Donde, θ_{gyro} es el ángulo del giróscopo y θ_{Accel} es el ángulo del acelerómetro calculado con el método de la tangente. Δt es el tiempo transcurrido desde la última vez que se ha aplicado el filtro. Esta fórmula se aplica para el eje X e Y.

El sensor realiza la lectura de los datos en los siguientes rangos por defecto:

Variable	Valor mínimo	Valor central	Valor máximo
Lectura MPU6050	-32768	0	+32767
Aceleración	-2g	0g	+2g
Velocidad angular	-250 °/s	0 °/s	+250°/s

Tabla 7.Rangos de lectura del sensor.[58]

Los valores de la lectura son valores sin pasar ningún tipo de conversión. Para pasarlo al sistema internacional tendremos que saber los rangos de aceleración y velocidad angular como se muestran en la tabla 7.

El mayor inconveniente es que la orientación del eje Z no es válida. Es necesaria implementar otro dispositivo como un magnetómetro o una brújula que proporcione su orientación de manera más fiable para asignarle su ángulo.

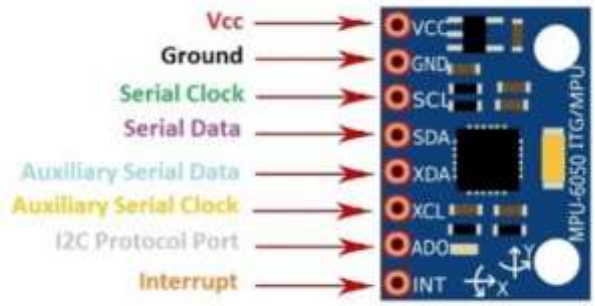


Figura 53. Pines del sensor IMU. (Fuente: [57])

Se muestran los pines del dispositivo:

- Vcc: Alimentación del chip a 5 V. Este dispositivo integra un regulador de tensión que reduce la tensión a 3.3V.
- Ground: Tierra.
- Serial Clock: Señal de reloj.
- Serial Data: Señal en la que van todos los datos del sensor.
- XDA: pin auxiliar para los datos.
- XCL: pin auxiliar de la señal de reloj.
- I2C Protocolo Port: Por defecto está a 0 y la dirección en la que se realiza la comunicación es 0x68. Si lo pones a 1, la dirección será 0x69.
- Int: Pin de interrupciones.

Características principales[59]:

- Su tensión de alimentación es de 2.4-3.6V, por eso cuenta con un regulador de tensión como bien se nombra antes permitiendo que se conecte a 5 V, ya que si no se tiene que realizar un divisor de tensión.

- Cuenta con un procesador digital de movimiento (Digital Motion Processor) el cual es capaz de fusionar los datos del acelerómetro y el giróscopo evitando tener que usar filtros complementarios.
- Consta de conversores analógicos digitales de 16 bits.
- El rango del acelerómetro puede ser ajustado a ±2g, ±4g, ±8g y ±16g y el del giroscopio a ±250, ±500, ±1000 y ±2000 ˚/sec.

4.1.2.1 Calibración del IMU 6050

El dispositivo presenta varios problemas a la hora de trabajar con él como puede ser ruido, error de offset o error acumulativo de medida[60].

El ruido se trata de una señal eléctrica que interfiere o se suma a nuestra señal de medida útil y, por lo tanto, altera su resultado. En este caso, afecta más ya que nuestra señal de medida es una señal analógica y esta posee mayor sensibilidad a dicho fenómeno. En cambio, en las señales digitales afecta en menor medida, aunque su impacto puede ser mucho mayor. La amplitud de la señal dará pie a una mayor cantidad de ruido. Para minimizar dicho problema se aplican distintas soluciones, no hay ninguna que sea standard puesto que todo depende de tu proyecto y las condiciones a las que esté sometido, teniendo en cuenta que la posibilidad de eliminar el ruido completamente es nula[61]:

- Reducir el ruido en su fuente: Consiste en separar la señal útil de su señal de distorsión, utilizada para su ruido externo. Esta técnica no es muy usada, ya que es muy compleja a la hora de implementarla.
- Aplicar filtros: Esta técnica puede ser utilizada tanto en el software como en el hardware. Es decir, se aplica por ejemplo un filtro paso bajo introduciéndole en el código o conectándolo con la placa Arduino. Como el filtro paso bajo hay una gran variedad de filtros con el mismo objetivo.

Estas son dos de las soluciones más comunes frente a otras muchos, pero también se destaca que la propia placa cuenta internamente con varios de estos métodos como son los filtros.

El Offset es un error de desplazamiento muy común en los dispositivos electrónicos analógicos. En este caso el sensor presenta un error de Offset en cada eje y por lo tanto puede darnos unos resultados de medida erróneos. Se reduce el valor de offset utilizando un sketch dado por JeffRowberg[62] en el que halla el offset de cada eje. Una vez tienes estos

valores, mediante la función: setAccelOffset y setGyroOffset fijamos el offset para cada uno de los ejes.

El error acumulado de medida (Drift) es muy común en los sensores IMU. Se trata de un error acumulativo a la hora de tomar las medidas. Tenemos el problema de que el acelerómetro no es del todo preciso y cualquier movimiento lo detecta y esto cuando se suman muchas medidas pueden llegar a distorsionar el valor final del que realmente es. Por otro lado, el giróscopo es más preciso, pero a la hora de hacer los cálculos para dar el ángulo este error se va acumulando y se produce este fenómeno. Para solventar esto se recoge un número elevado de muestras y se hace la media para coger una medida óptima. Aunque la mejor solución sería aplicar un filtro como se comenta anteriormente en el caso del ruido.

4.1.3 Sensor inercial: IMU MPU-9250

El sensor IMU 9250 tiene nueve grados de libertad y cuenta con tres sensores: acelerómetro, giróscopo y magnetómetro. Todo esto en un mismo integrado y gracias a esto, tiene la capacidad de realizar las mediciones de manera simultánea.[63]

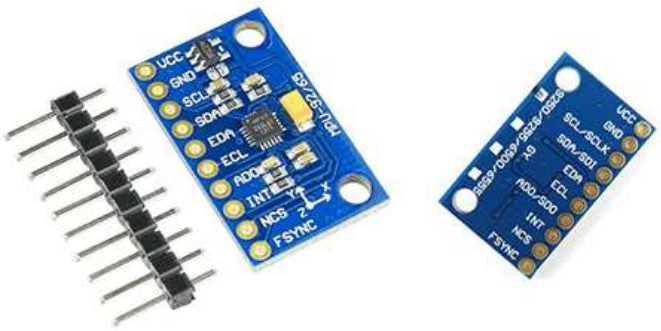


Figura 54. Sensor inercial IMU MPU-9250. (Fuente: [63])

Este modelo incorpora un integrado IMU6050, que está formado por el acelerómetro y el giróscopo como se explica en el apartado anterior. Y un magnetómetro AK8975. La comunicación se puede realizar tanto por bus SPI como por el bus I2C.

Su tensión de alimentación va de 2,4 a 3,6V. Pero de la misma forma que el anterior sensor, cuenta con un regulador de tensión para conectarlo a 5V.

Dispone de conversores analógicos digitales (ADC) de 16 bits. El rango del acelerómetro puede ajustarse a ±2g, ±4g, ±8g y ±16g, el del giroscopio a ±250, ±500, ±1000 y ±2000 ˚/s y el magnetómetro hasta ±4800µT.

También está compuesto por un procesador interno Digital Motion Processor (DMP), el cual ejecuta los algoritmos complejos evitando tener que usar filtros de forma exterior.

La ventaja de este sensor es que al tener un magnetómetro elimina el error de deriva (Drift). Es la mejor opción para el proyecto ya que el robot utiliza este sensor durante un periodo largo de tiempo.

4.1.4 Rapberry Pi Model 3B+

Es necesario un procesador con la suficiente potencia para poder compilar nuestros programas, pines digitales para poder comunicarse con nuestro sensor ultrasonidos y un módulo wifi.

El modelo 3 B+ recoge todas las características necesarias.

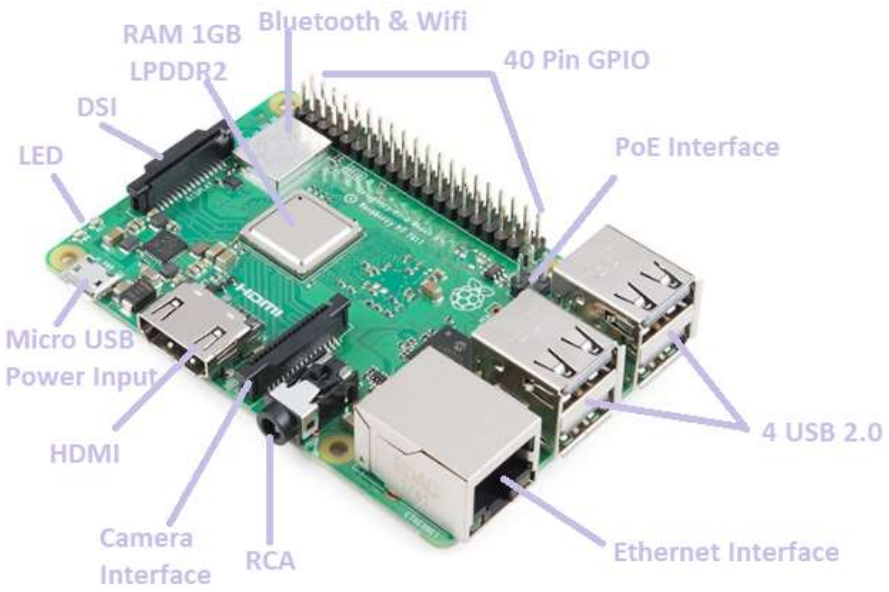


Figura 55.Partes de la Raspberry Pi 3 Model B+.

Como se puede ver es una placa bastante completa ya que cuenta con todas las conexiones posibles para realizar un proyecto de este alcance, entre ellas se destaca:

- Interfaz para poder conectarlo a una cámara.

- Módulo wifi o bluetooth para realizar las correspondientes conexiones entre cliente y servidor (Raspberry-Matlab).
- Pines GPIO (General porpuse input output), los cuales se utilizan como entradas o salidas dependiendo de nuestras necesidades.
- Puerto RCA que envía señales de video analógicos para posibles mejoras futuras.
- Puerto HDMI, se utiliza para conectarlo a una pantalla y poder así configurar nuestra Raspberry de una manera mucho más cómoda y sencilla.
- Procesador de 1GB de RAM, que en este caso es una capacidad bastante asequible.

4.1.4.1 Sistema operativo

Para poder utilizar el dispositivo necesitas instalar un sistema operativo. Previamente tienes que disponer de una tarjeta SD para almacenar este sistema operativo. En este caso se utilizó dos tarjetas SD, una de 8GB y otra de 128GB. La primera tarjeta SD resulto dañada por el sistema operativo tras un largo periodo de tiempo usándola. Y para evitar que se dañase la Raspberry Pi se comenzó a utilizar la otra.

El Sistema operativo que se instaló es Raspbian. Este Sistema operativo es el más recomendado por los fabricantes, ya que es el que menos problemas genera para nuestra Raspberry Pi y básicamente está diseñado para ella. Se trata de una distribución del Sistema operativo GNU/Linux basado en Debian[64].

Se trata de un software libre y cuenta con un soporte optimizado para cálculos en coma flotante que permite dar mayor rendimiento dependiendo del caso y con la ventaja de su instalación que no hace falta configurarlo de manera manual.

Su principal herramienta de Desarrollo es IDLE para lenguajes de programación de Python o Scratch.

En este caso la instalación se hace desde la página oficial de Raspberry Pi (www.raspberrypi.org).

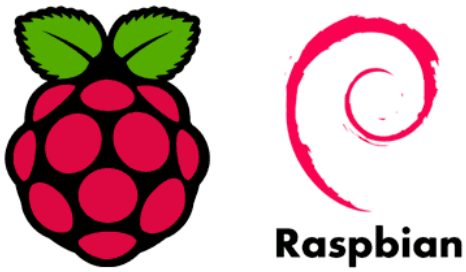


Figura 56.Logo del sistema operativo Raspbian. (Fuente: [65])

4.2 CONEXIÓN CON MATLAB

4.2.1 Conexión Arduino con Matlab

Para realizar la conexión entre Arduino y Matlab debes descargar el siguiente paquete desde Matlab:

Figura 57. Paquete para la conexión de Arduino y Matlab.

Gracias a las funciones, algoritmos y herramientas de representación gráfica integradas en Matlab nos va a permitir sacar todos esos datos que previamente son procesados sin necesidad de realizar un protocolo de comunicación I2C o SPI.

4.2.2 Conexión Matlab con Rasberry pi

Para realizar la conexión entre nuestra Rapberry Pi y Matlab debes descargar el siguiente paquete:

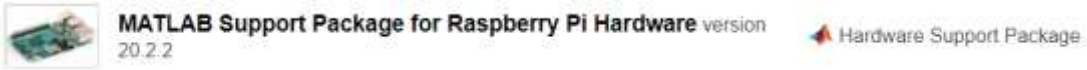


Figura 58. Paquete para la conexión de Raspberry Pi y Matlab.

Una vez lo descargas, también debes descargar un fichero generado durante la configuración del paquete que debemos introducir en la tarjeta SD de la Raspberry PI donde se encuentra el sistema operativo para que esta conexión sea posible.

Existen varias opciones para realizar la conexión entre Matlab y Raspberry Pi, pero la que interesa es la conexión remota mediante wifi, así facilitará la conexión con el simple hecho de que estén conectados a la misma red inalámbrica.

La conexión se realiza conectando Matlab a un servidor de la Raspberry Pi el cual está integrado en su firmware. Este proceso se hace a través de TCP/IP. En el código se crea un objeto como se muestra en la siguiente figura:

```
r = raspi;
```

Y en este objeto se encuentran todos los datos del dispositivo: Periféricos disponibles, dirección IP, modelo de Raspberry Pi…

Para comprobar que la comunicación es correcta se realiza un código simple recomendado en Matlab Help Center en el que se enciende y se apaga el Led de nuestra Raspberry Pi de manera intermitente.

4.3 ESTRUCTURA DEL PROGRAMA

El funcionamiento del programa está basado en la visión artificial.

El sistema simula recoger fotos cada cierto tiempo del entorno (proporcionadas por nosotros en este caso). Una vez recoge las fotos, las analiza y destaca la baliza que busca. Este entorno le reconoce en base a dos características geométricas, el área del objeto y su circularidad, las cuales hacen más fácil su reconocimiento en la imagen. Previamente a esto, se procesa la imagen y se segmenta aplicando máscaras y funciones que mejoren y eliminen interferencias en esta.

Después el sensor inercial le dará la orientación del robot mediante un programa creado en Matlab.

Una vez sabe su orientación, sabe a qué sensor ultrasonidos recurrir. Este sensor se pondrá en marcha y calculará la distancia a la baliza. Emitirá una señal ultrasónica en busca del objeto. Cuando esta choca con el objeto, el eco llega a el receptor y desde el programa calcula el tiempo transcurrido entre estas dos señales y saca la distancia a esta referencia.

4.3.1 Reconocimiento de la baliza

El programa está estructurado en las etapas explicadas en el estudio previo.

Preprocesado

Se pasa la imagen a escala de grises para que pueda ser procesada y analizada de manera más fácil.

```
I1 = imread('Foto2.jpeg');
Igray = rgb2gray(I1);
figure,imshow(Igray); title('Escala de grises');
```

Con la función multithresh genera el umbral en cuatro niveles de la imagen en la escala de grises y se escoge el tercero que es el óptimo.

```
thresh = multithresh(Igray,4);      %Imagen binarizada
  thresh1 = thresh(3)
  Igrayc = Igray<thresh1;
```

Segmentación

Mediante rgb2hsv se pasa a formato HSV la imagen para poder analizar su tono, saturación y valor de la imagen. El primer objetivo fue analizar la tonalidad para obtener más datos del objeto. Debido a varios problemas para poder segmentar bien la foto, se descartó esta característica y, por lo tanto, no hace falta analizar la foto en el espacio HSV.

```
Ihsv=rgb2hsv(I1);                                %espacio HSV
  hImage = Ihsv(:,:,1); figure,imshow(hImage); title('Tono');
  sImage = Ihsv(:,:,2); figure,imshow(sImage); title('Saturación');
  vImage = Ihsv(:,:,3); figure,imshow(vImage); title('Valores');
```

Se crean varias máscaras con el fin de eliminar los reflejos y brillos de la imagen y poder llegar a captar mejor nuestro objetivo. Utilizamos la herramienta colorThresholder. Se basa en unos valores en formato RGB. Extrae los umbrales de la imagen para trabajar en valores determinados y segmenta la imagen basándose en su color. Estas máscaras se adjuntarán en los anexos.

```
    PhotoF = createMaskTry(I1) %Primera máscara
    Photo1   =   imfill(PhotoF,'Relleno');   figure(),imshow(PhotoF);
title('Primera Máscara');

    Foto = createMaskTry2(I1)
    Photo3    =    imfill(Foto,'Relleno');    figure(),imshow(Foto);
title('Segunda Máscara');

    PhotoB = BlackMask(I1);
    Photo2   =   imfill(PhotoB,'Relleno');   figure(),imshow(PhotoB);
title('Tercera Máscara');
```

```
    Photo4 = createMaskP4(I1)
    Photoo4   =   imfill(Photo4,'Relleno');   figure(),imshow(Photo4);
title('Cuarta Máscara');
```

Con la operación lógica ‘or’ se unen los objetos que se encuentren en esas dos fotos. Y mediante operaciones morfológicas como es imfill rellena los huecos que vea vacíos.

```
    Ima1 = or(Photoo4,(Photo2));
    New1 = imfill(Ima1,'huecos');
```

Se aplica operaciones morfológicas basadas en la dilatación, erosión, apertura y cierre. Y aplicamos la operación lógica ‘and’.

```
    BW1=bwmorph(Photo1,'dilate');
    BW12=bwmorph(BW1,'erode',1);
    BW2=bwmorph(Photo3,'erode',1);
    BW4=and(BW12,BW2);
    R = and(Photo2,New1);
    HOLES=bwareaopen(R,2000);
    R1 = and(BW12,HOLES);
    BW=bwmorph(R1,'close');
```

Se aplica el método Sobel para el reconocimiento de bordes teniendo en cuenta nuestro umbral calculado al principio del programa.

```
  [Iedge,thresh]=edge(R,'Sobel');
    Iedge = edge(R,'Sobel',thresh);
    figure(), imshow(Iedge); title('Bordes hallados');
```

Extracción de características

Se etiquetan todos los píxeles de la imagen con el comando label.

```
        labelBW = bwlabel(BW);
```

Teniendo la información de todos los objetos de la imagen, mediante regionprops se sacan todas las características que nos interesen.

Como la primera opción es analizar su tonalidad, se extrae la media de su intensidad (MeanIntensity).

```
        ObjProperties = regionprops(labelBW, BW, 'all');
        ObjPropertiesGray     =     regionprops(labelBW,     Igray,
'MeanIntensity');
```

Se saca el número de objetos que identifica en la imagen.

```
        numberOfObj = size(ObjProperties, 1);
        figure(),imshow(I1); title('Resultados');

            hold on
```

Se calcula el centroide, el área y la circularidad.

```
for i = 1 : numberOfObj

        OptCentroid = ObjProperties(i).Centroid;            %Centroids
        OptArea = ObjProperties(i).Area;                       %Area
        OptCircularity        =        ObjProperties(i).Circularity;
%Circularity
```

Clasificación de objetos

Se clasifica mediante los valores que se estiman previamente al hacer el estudio de muchas imágenes en distintos ángulos y distancias para asegurar su correcta clasificación.
El centroide se saca para etiquetar el objeto de manera centrada y visualmente más cómoda para el usuario en la imagen.

```
    if  OptArea>=44000 & OptArea<=69000)
            if(OptCircularity>=0.155 & OptCircularity<=0.335)
                text(OptCentroid(1)                                  +20,
OptCentroid(2),'BALIZA','Color','#A2142F',      'FontSize',      16,
'FontWeight', 'Bold');
            end
    end
```

```
    if (OptArea>=9900 & OptArea<=18000)
            if(OptCircularity>=0.39 & OptCircularity<=0.48)
                text(OptCentroid(1)                                +20,
OptCentroid(2),'BALIZA','Color','#A2142F',      'FontSize',      16,
'FontWeight', 'Bold');
            end
    end
```

4.3.2 Cálculo de distancia

Para hallar la distancia, previamente se realiza la conexión con la Raspberry Pi para poder extraer los datos que nos hacen falta. Se crea un objeto llamado ‘raspi’ para comenzar la conexión:

```
r = raspi;
```

Se configura los puertos de entrada y salida en la Raspberry Pi y se dispara una señal en el trigger:

```
Trig = configurePin(r,18,'DigitalOutput');
configurePin(r,24,'DigitalInput');
writeDigitalPin(r,18,1);
```

Se saca la hora, minutos y segundos actuales.

```
c = clock;
[c tf] = clock;
Horas=c(1,4) ;
Minuto=c(1,5);
Segundo=c(1,6);
```

Se lee el pin donde devuelve la señal ultrasónica para comprobar si ha devuelto la señal. Si el pin está a 0, el programa no hará ningún cálculo de ningún tipo:

```
Echo = readDigitalPin(r,24)
while Echo == 0
        StartTime = clock;
end
```

Y si el pin está a 1, para el tiempo. Se saca el tiempo transcurrido y se calcula la distancia mediante la fórmula.
La velocidad hay que pasarla a cm por segundo ya que la distancia nos la va a dar en centímetros.

```
while Echo == 1

    StopTime = clock;
    [StopTime tf] = clock;
    Hora2 = StopTime(1,4);
    Minuto2 = StopTime(1,5);
    Segundos2 = StopTime(1,6);

    FinalTime = [ Hora2-Horas, Minuto2-Minuto, Segundos2-Segundo];
    tiempoF = num2str(FinalTime,'%3.2f');

    v = 1/29; %cm/s Velocidad del sonido
    dist = (1/2)* v * tiempoF;
%Cálculo de la distancia
    fprintf('El dispositivo se encuentra a %4.2f cm de la
baliza.\n',dist);  %Mostramos por pantalla
    Echo = 0;
end
```

4.3.3 Cálculo de la aceleración y orientación

Este programa calcula la aceleración y orientación. Esta basado en las referencias dadas[66] [67].

Arduino ide

Se utiliza la librería Wire.h y la del sensor MPU-6050 para poder utilizar el código.

```
#include <Wire.h>
```

```
#include <MPU6050.h>
```

Se define la dirección 0x68 para realizar la comunicación I2C.

```
#define MPU 0x68
```

Como se define en el diseño del circuito, se sabe el valor mínimo y máximo de la lectura del MPU6050. Para pasarlo al Sistema internacional se divide el valor máximo de la lectura entre el valor máximo de la aceleración y el resultado es m/s^2. Y, por otro lado, se divide el valor máximo de lectura entre el valor máximo de la velocidad angular y el resultado es °/s.

```
#define A_R 16384.0 // 32768/2

#define G_R 131.0 // 32768/250
```

Para convertirlo a grados se multiplica por el siguiente valor:

```
#define RAD_A_DEG = 57.295779
```

Los valores que recogemos del IMU son enteros de 16 bits.[68]

```
//Valores RAW

int16_t AcX, AcY, AcZ, GyX, GyY, GyZ;
```

Se definen los ángulos como un valor tipo float.

```
//Angulos

float Acc[2];

float Gy[3];

float Angle[2];

String valores;


long tiempo_prev;

float dt;
```

Se realiza la configuración en el setup.

```
void setup()

{

Wire.begin(); // D2(GPIO4)=SDA / D1(GPIO5)=SCL

Wire.beginTransmission(MPU);

Wire.write(0x6B);

Wire.write(0);
```

```
Wire.endTransmission(true);
Serial.begin(9600);
```

Se aplican los Offset calculados previamente con el programa de calibración que se adjunta en los anexos.

```
mpu.initialize();   //Calibration offset
mpu.setXAccelOffset(-1540);
mpu.setYAccelOffset(-4211);
mpu.setZAccelOffset(1234);
mpu.setXGyroOffset(13);
mpu.setYGyroOffset(-28);
mpu.setZGyroOffset(26);
}
```

En el bucle se leen los valores del acelerómetro y giróscopo y los resultados se dan en Unidades del Sistema internacional.

```
void loop()
{
   //Leer los valores del Acelerometro de la IMU
   Wire.beginTransmission(MPU);
   Wire.write(0x3B); //Pedir el registro 0x3B - corresponde al AcX
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,6,true);    //A partir del 0x3B, se piden 6 registros
   AcX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   AcY=Wire.read()<<8|Wire.read();
   AcZ=Wire.read()<<8|Wire.read();
```

A partir de los valores del acelerómetro se calcula los ángulos con el método de la tangente:

```
   Acc[1]      =      atan(-1*(AcX/A_R)/sqrt(pow((AcY/A_R),2)      +
pow((AcZ/A_R),2)))*RAD_TO_DEG;
```

```
   Acc[0]        =        atan((AcY/A_R)/sqrt(pow((AcX/A_R),2)        +
pow((AcZ/A_R),2)))*RAD_TO_DEG;
```

Lee los valores del Giroscopio:

```
   Wire.beginTransmission(MPU);
   Wire.write(0x43);
   Wire.endTransmission(false);
   Wire.requestFrom(MPU,6,true);    //A partir del 0x43, se piden 6
registros
   GyX=Wire.read()<<8|Wire.read(); //Cada valor ocupa 2 registros
   GyY=Wire.read()<<8|Wire.read();
   GyZ=Wire.read()<<8|Wire.read();
```

Calcula el ángulo de orientación en cada eje:

```
   Gy[0] = GyX/G_R;
   Gy[1] = GyY/G_R;
   Gy[2] = GyZ/G_R;

   dt = (millis() - tiempo_prev) / 1000.0;
   tiempo_prev = millis();
```

Se le aplica el filtro complementario al eje X y el eje Y.

```
   Angle[0] = 0.98 *(Angle[0]+Gy[0]*dt) + 0.02*Acc[0];
   Angle[1] = 0.98 *(Angle[1]+Gy[1]*dt) + 0.02*Acc[1];
```

Para hallar la orientación aproximada del Angulo Z se integra respecto del tiempo.

```
   Angle[2] = Angle[2]+Gy[2]*dt;
```

Muestra los valores por pantalla:

```
   valores = "90, " +String(Angle[0]) + "," + String(Angle[1]) + ","
+ String(Angle[2]) + ", -90";

   valores = (Angle[2]) ;

   Serial.println(valores);


   delay(1000);

}
```

Matlab

En Matlab se leen todos los valores que están en el puerto serial y les ponemos en una tabla sacando la aceleración y su velocidad angular en unidades del sistema internacional.

```
clear all
a = arduino('COM3', 'Uno', 'Libraries', 'I2C');
imu =
mpu6050(a,'SampleRate',50,'SamplesPerRead',5,'ReadMode','
Latest');
[sensorReadings,overrun] = read(imu)
```

5. RESULTADOS

5.1 VISIÓN ARTIFICIAL

Se muestra las imágenes iniciales que analiza el programa.

Figura 59. Secuencia de imágenes.

Como se puede ver las fotos son de distintos ángulos, luminosidad y proximidades a la baliza.

5.1.1. Proceso

Muestra los resultados de cada etapa en el procesamiento de la imagen.

Preprocesamiento de la imagen

La imagen se pasa a escala de grises ya que es necesaria para poder sacar todas las características:

Figura 60.Imagen en escala de grises.

Segmentación y extracción de características

Se representa en el espacio HSV.

Figura 63. Hue (Tono).

Figura 61.Saturation (Saturación)

Figura 62.Value (Valor)

Se somete la imagen a varias máscaras. El objetivo de estas máscaras es la eliminación de reflejos, ruido y demás interferencias con nuestro objeto.

Figura 65. Primera Máscara.

Figura 64.Segunda Máscara.

Figura 67. Tercera mascara.

Figura 66. Cuarta mascara.

Todas las máscaras estan sometidas posteriormente a una operación morfológica llamada imfill la cual rellena los agujeros y puede eliminar un determinado rango de ruido de cada imagen.

- **Operaciones lógicas y transformaciones morfológicas.**

Se muestran el resultado de las operaciones lógicas y trasformaciones morfológicas aplicadas a la imagen.

Figura 68. Imagen resultada de una OR entre la figura 66 y 67.

Figura 69. Dilatación de la primera máscara (fig. 65) después de la dilatación.

Figura 70. Erosión

Se puede ver que el entorno del objeto mejora. Hay menos marcas en el jardín.

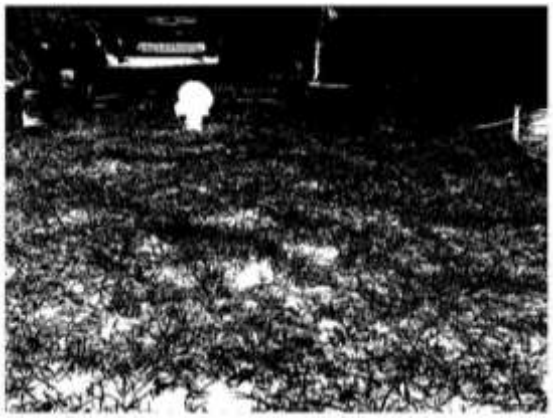


Figura 71. Operación lógica AND de la tercera máscara (67) y la figura (70).

Se realiza una apertura y se puede ver el notable cambio.

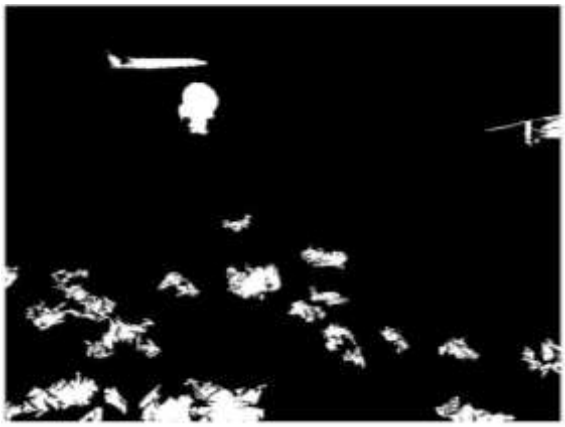


Figura 72. Apertura de la imagen 71.

Se somete a otra operación lógica para juntar la imagen anterior con la figura 70. Y este es el resultado:

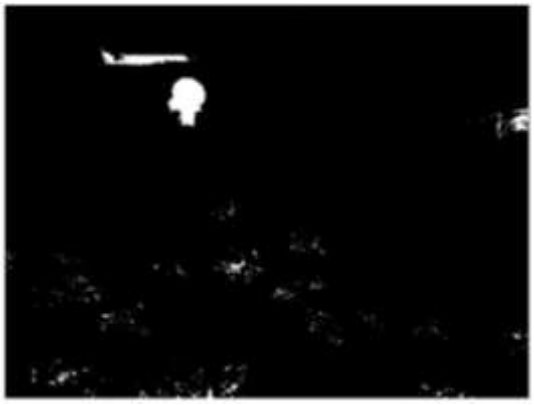


Figura 73. Resultado de la operación AND.

Como se puede ver, se muestra la región que tienen en común sus píxeles.

Para mejorar la operación lógica, se aplica una operación morfológica de cierre:

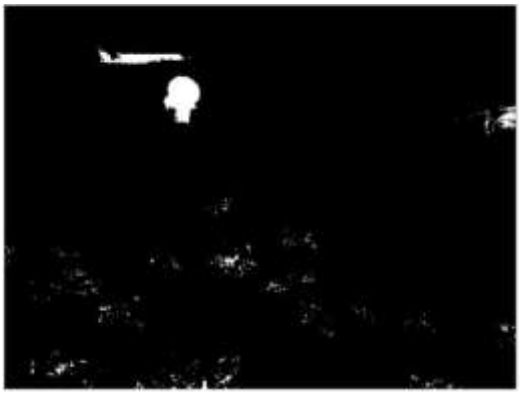


Figura 74. Operación morfológica de cierre.

- **Método sobel**

Se hallan los bordes mediante la función ‘edge’ y se le aplica el método sobel teniendo en cuenta su umbral hallado anteriormente.

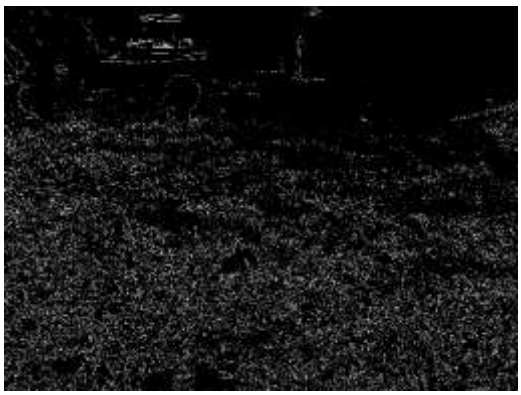


Figura 75. Método Sobel.

5.1.2. Resultado final

Por razón de recursos se decidió analizar el entorno en base a su área y su circularidad.

Se muestra que reconoce la baliza en su entorno, nombrándola como “BALIZA”:

Figura 76. Secuencia de resultados.

5.2 SENSÓRICA

5.2.1 Sensor ultrasonidos HC – SR04

Al usuario se le muestra por pantalla el siguiente mensaje:

```
El dispositivo se encuentra a 8.28 cm de la baliza.
```

5.2.2 Sensor MPU-6050

Se muestra los datos que recoge el sensor:

Acceleration			AngularVelocity		
x	y	z	x	y	z
-0.41673	0.30656	9.3478	-0.041302	0.011724	0.47564
-0.32213	0.21675	9.3058	-0.046498	0.0074609	0.56343
-0.29818	0.089813	9.4292	-0.054358	0.011458	0.66602
0.66462	0.59157	9.4939	-0.063685	0.0029311	0.80352
0.25148	0.50774	9.4124	-0.079273	-0.0063951	1.0551

Figura 77. Resultados sensor IMU MPU-6050.

Las unidades son las del sistema internacional.

Aceleración: $\frac{m}{s^2}$ Velocidad angular: $\frac{rad}{s}$

Como se puede observar la aceración en el eje Z es muy cercana a 9,8. Se sabe que el sensor está midiendo de forma correcta gracias a esto, puesto que la gravedad tiene un valor de 9,8 en ese eje.

Se halla la orientación en el eje Z. Para ello, se sabe su velocidad angular final y la inicial, se resta y se pasa a grados multiplicándolo por 180/pi. Se obtiene un valor, pero no es un valor real y fiable ya que no se le puede aplicar un filtro complementario capaz de eliminar ese error de deriva que debido a las vibraciones y el ruido hace que las medidas no sirvan para dar una orientación adecuada del robot. Estas medidas son valores relativos así que no se pueden tomar para hallar su orientación en valores absolutos.

Para conocer la orientación en el eje Z en valor absoluto se requiere multitud de cálculos y softwares complejos para hallarlo.

En el caso de que la orientación en el eje Z salga con valores absolutos sin ningún tipo de error. Lo que se haría posteriormente, con la ayuda de un programa en Matlab se clasifica

esa orientación en Norte, noreste, este, sureste, sur, suroeste, oeste y noroeste. Al saber esta orientación se escogería entre uno de los seis sensores ultrasonidos para poder calcular la distancia al objeto con mayor precisión.

Figura 78. Orientación de una brújula. (Fuente: [69])

6 CONCLUSIONES

La idea principal era procesar todo con la Raspberry Pi puesto que tiene un procesador más potente (1GB RAM) y abarca más autonomía a la hora de trabajar. Se encuentran distintos problemas para implementar Matlab ya que al usar la parte de visión artificial requiere un gran peso en el sistema. Matlab es capaz de funcionar correctamente con mínimo 1GB de RAM, pero es recomendable usar 4GB.

Existe la versión de Matlab online el cual se puede trabajar sin problema en la Raspberry Pi, pero se decide finalmente por comodidad y rapidez procesar los programas desde un ordenador conectando la Raspberry Pi con Matlab de manera online.

Otro de los problemas surgió con el sensor inercial MPU 6050. Los datos que se obtienen de este sensor son relativos, por lo tanto, se llega a la conclusión de que hay que combinarlo con un magnetómetro para que se pueda dar el eje de orientación Z de una manera fiable. Debido a esto, se decide cambiar de modelo de IMU, por el IMU - MPU9250, un modelo que además de integrar un magnetómetro, es compatible con Matlab.

Con el sensor IMU MPU-9250 no fue posible recoger los datos que se buscaban. No fue posible la conexión con Matlab, no es capaz de detectar nuestro sensor. Se comprueba con el software de Arduino y este si que lo detecta. Por lo tanto, el problema que se produjo no pudo ser solventado y no se pudo obtener otro dispositivo de este modelo. En el caso de obtener otro dispositivo del mismo modelo, se obtienen los datos del magnetómetro y se halla la orientación del eje Z y se puede hallar de manera fiable la orientación del robot sobre la superficie.

En cuanto a la visión artificial, se destaca el entorno y sus condiciones como limitaciones. Al estar en un entorno exterior conlleva reflejos, brillos y multitud de ruido en la foto. Esto hace que el proceso de segmentación sea mucho más costoso y por lo tanto, el algoritmo es más complejo. La visión artificial demuestra que es una herramienta que puede dar muchas ventajas en la robótica. Constata la importancia de esta rama en la ingeniería a la hora de realizar proyectos de mejora.

Por otro lado, el sensor ultrasonidos cumple su función calculando la distancia a la baliza. Aunque sería mucho más fiable esta distancia si se complementa con un sensor infrarrojos.

En conclusión, el posicionamiento y localización en exteriores de un robot conlleva bastante trabajo. El entorno exterior alberga multitud de interferencias y malas condiciones ambientales para implementar estos sistemas.

Hay que tener en cuenta muchos factores previos como las condiciones de humedad y temperatura en las que trabaja. La precisión de los dispositivos es importante al ser un entorno amplio y cerrado, un pequeño error que se acumula puede desajustar todos los resultados. Al hacer un proyecto de este tipo, se debe hacer un amplio estudio previo para poder diseñar el algoritmo que mejor se ajuste al robot.

7. FUTURAS LÍNEAS DE TRABAJO

Esta aplicación la podemos llevar a cabo totalmente implementando una cámara capacitada para el desarrollo de aplicaciones de visión artificial y seis sensores ultrasonidos HC-SR04 para poder calcular con mayor precisión a partir de su orientación. De esta manera, al calcular la orientación, se elige el sensor ultrasonidos que se instala en esa orientación y calcula con mayor precisión la distancia a la baliza.

También se puede trabajar con otro tipo de sensor como puede ser el LVDT que alcance mayor rango de amplitud y trabaja en condiciones del entorno pésimas, incluyendo un encoder el cual pueda registrar el recorrido del dispositivo.

La mejor opción sobre el lenguaje de programación será Python, el cual es muy similar a Matlab, con la ventaja de que es un lenguaje libre y de licencia gratuita. Python es un lenguaje muy usado hoy en día debido a sus grandes ventajas y cuenta con una herramienta para trabajar con visión artificial con multitud de funciones.

Con respecto al sensor inercial, el modelo MPU9250 se compone de un sensor de temperatura. Con estos datos se realiza un algoritmo que detecte cuando hace demasiado frio y lluvia junto a otro sensor de lluvia o que haga demasiado calor para que el aparato trabaje y no esté sometido a una alta o muy baja temperatura perjudicial para los dispositivos.

Otra de las opciones que puede ser implementada es que se distribuyan varias balizas de distintos colores con el objetivo de hacer un mapa después de analizar el entorno. Para realizar esta opción se utilizaría un algoritmo que analice en el espacio HSV para poder clasificar los distintos colores por el tono y la saturación. Simplemente se cogen los valores del tono y saturación que tienen esos colores, además de incluir el área u otra característica geométrica para que lo distinga de cualquier otro objeto.

Se observa que la visión artificial alberga infinidad de posibilidades y por lo tanto, es una buena opción para combinar con la electrónica y automática para el desarrollo de futuros sistemas.

8.BIBLIOGRAFÍA

[1] “Evolución jardinería.” https://guiadejardin.com/20-anos-de-historia-del-cortacesped/ (accessed Sep. 12, 2020).

[2] “Robotica en la jardineria.” https://guiadejardin.com/mi-experiencia-con-el-robot-cortacesped/ (accessed Sep. 12, 2020).

[3] “Robots autónomos segadores.” https://www.comercturro.com/blog/jardin/los-robots-cortacesped-ventajas-desventajas-y-opinion.html#:~:text=El principal beneficio de un,más tiempo%2C especialmente en verano. (accessed Oct. 12, 2020).

[4] “Comparativa de robots.” .

[5] S. Manual, “C model Service Guide Table of Content.” 2016.

[6] S. Manual, “S model Service Guide.” 2016.

[7] L. J. Marín Paniagua, “Localización de robots móviles de recursos limitados basada en fusión sensorial por eventos,” 2014.

[8] L. Rouhiainen, “Inteligencia artificial 101,” p. 352, 2018, [Online]. Available: https://planetadelibrosar0.cdnstatics.com/libros_contenido_extra/40/39307_Inteligencia_artificial.pdf.

[9] *Torres, J. (2018). Deep learning. Introducción práctica con Keras, Barcelona: Lulu. com.* .

[10] A. Çelik *et al.*, *Técnicas basadas en visión artificial*, vol. 1, no. 1. 2018.

[11] “Machine Learning.” https://empresas.blogthinkbig.com/que-algoritmo-elegir-en-ml-aprendizaje/ (accessed Jul. 12, 2020).

[12] “Espectro visible.” .

[13] “Campo visual - Cortex.” https://m.facebook.com/story.php?story_fbid=2647983275426541&id=137728638582957 6 (accessed Jul. 10, 2020).

[14] P. Audiovisual, “Representación De Imágenes .”

[15] C. Vision, “2D VISION INTRODUCTION BASIC IMAGE PROCESSING EDGE DETECTION MORPHOLOGICAL TRANSFORMS IMAGE SEGMENTATION FEATURE EXTRACTION OBJECT RECOGNITION.”

[16] C. Vision, “2D VISION INTRODUCTION - BASIC IMAGE PROCESSING -

MORPHOLOGICAL TRANSFORMATIONS.” pp. 1–18.

[17] W. Jara, “Visión Artificial : Análisis Teórico del Tratamiento Digital de Imágenes Para su aplicación en la identificación de objetos.,” 2006.

[18] +Universidad de Jaén, “Segmentación. Transformada de Hough.,” p. 5, 2006.

[19] P. D. E. Imágenes, “3. procesamiento de imágenes.”

[20] C. Vision, “2D VISION-INTRODUCTION- BASIC IMAGE PROCESSING - EDGE DETECTION - MORPHOLOGICAL TRANSFORMS - IMAGE SEGMENTATION FEATURE EXTRACTION OBJECT RECOGNITION.” .

[21] C. Vision, “Computer vision - Feature extraction.” pp. 1–24.

[22] “Parámetros en Matlab.” https://es.mathworks.com/help/images/ref/regionprops.html (accessed Aug. 02, 2020).

[23] J. A. Cancelas, R. C. González, I. Álvarez, and J. M. Enguita, *Procesamiento Morfológico, Visión 3D: Estereoscopía, Álgebra lineal básica para visión por computador, Geometría Proyectiva para Visión 3D*, vol. 1. 2016.

[24] M. Ii, “Etapas de la visión artificial,” 2008.

[25] C. Vision, “2D VISION - OBJECT RECOGNITION Knowledge base Problem.” pp. 1–20.

[26] “Sensor por cable.” https://sensores-de-medida.es/catalogo/sensor-de-distancia-por-cable-asm-ws10/]%22] (accessed Jul. 12, 2020).

[27] “Sensor por cinta.” https://sensores-de-medida.es/catalogo/sensor-de-distancia-por-cinta-asm-wb12/ (accessed Jul. 15, 2020).

[28] “Sensor magnetrostictivo.” https://sensores-de-medida.es/catalogo/transductor-de-posicion-lineal-perfil-plano-pcfp/ (accessed Jul. 14, 2020).

[29] “LVDT.” https://www.ingmecafenix.com/automatizacion/lvdt/ (accessed Jul. 14, 2020).

[30] “Sensor inductivo.” https://www.keyence.com.mx/ss/products/sensor/sensorbasics/proximity/info/ (accessed Jul. 14, 2020).

[31] “Sensor inductivo 2.” https://www.keyence.com.mx/ss/products/sensor/sensorbasics/proximity/info/ (accessed Jul. 14, 2020).

[32] “Sensor Láser : funcionamiento.” https://www.keyence.com.mx/ss/products/sensor/sensorbasics/laser_location/info/ (accessed Sep. 14, 2020).

[33] “Infrarrojo.” https://didactronica.com/detector-de-obstaculos-y-movimiento/ (accessed Jul. 16, 2020).

[34] “Sensor ultrasónico.” .

[35] R. A. Serway and L. D. Kirkpatrick, *Physics for Scientists and Engineers with Modern Physics*, vol. 26, no. 4. 1988.

[36] R. Ambrosio and J. Mireles, “Análisis y caracterización de un acelerómetro capacitivo fabricado con tecnología polymump’s,” *Superf. y vacío*, vol. 23, no. 3, pp. 26–31, 2010.

[37] “Acelerómetro ADX.” https://www.robotshop.com/us/es/acelerometro-adxl335-para-lilypad.html (accessed Jul. 20, 2020).

[38] M. Arenas, “Sistema Para la Adquisicion y monitorizacion de aceleracion mediante microprocesador,” *Univ. Sevilla*, pp. 39–54, 2008.

[39] “Giróscopo.” https://www.5hertz.com/index.php?route=tutoriales/tutorial&tutorial_id=13 (accessed Jul. 25, 2020).

[40] “Giróscopo.” .

[41] J. Manuel and Q. Reboul, “Evaluación de un Giróscopo MEMS en un pendulo,” p. 12, 2011, [Online]. Available: http://www.gte.us.es/ASIGN/SEA/pracs/Eval_Gir_con_datasheet.pdf.

[42] “Magnetómetro.” https://www.5hertz.com/index.php?route=tutoriales/tutorial&category_id=1&tutorial_id=1#2 (accessed Dec. 10, 2020).

[43] “Introducción Raspberry PI.” https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi (accessed Sep. 26, 2020).

[44] “Introducción Raspberry PI.” .

[45] 上田哲史 and 板東孝文, “Raspberry Pi,” p. 241, 2017, [Online]. Available: https://www.raspberrypi.org/.

[46] “Comparativa modelos Raspberry PI.” https://raspberryparatorpes.net/raspberry-pi-tabla-tecnica-completa/ (accessed Sep. 23, 2020).

[47] “Comparativa modelos Raspberry PI.” .

[48] R. Enríquez Herrador, “Guía de Usuario de Arduino,” p. 49, 2009.

[49] “Arduino image.” https://www.ingmecafenix.com/electronica/arduino/] (accessed Sep. 15, 2020).

[50] M. Cleary, “Fundamentos de Matlab,” *J. Chem. Inf. Model.*, vol. 53, no. 9, pp. 1689–1699, 2019.

[51] “Protocolo de comunicacion : Serial.” https://hetpro-store.com/TUTORIALES/puerto-serial/#:~:text=Un puerto Serial es un,-duplex%2C Duplex y Simplex. (accessed Oct. 29, 2020).

[52] E. Carletti, “Comunicación - Bus I2C Descripción y funcionamiento,” *Retrieved from http//robotsargentina. com.ar/Comunicacion_busI2C.htm*, pp. 1–5, 2008.

[53] “Matlab helps.” https://es.mathworks.com/help/matlab/getting-started-with-matlab.html?searchHighlight=MATLAB&s_tid=srchtitle.

[54] “HC RS4.” https://medium.com/disruptiva/aws-iot-raspberry-pi-4-obteniendo-datos-de-un-sensor-de-ultrasonido-en-tiempo-real-4a5a0b25d943V.

[55] “Sensor ultrasonidos HC-SR04 - Datasheet.”

[56] “sensor IMU 6050 -GY521.” https://www.naylampmechatronics.com/blog/45_Tutorial-MPU6050-Acelerómetro-y-Giroscopio.html (accessed Jul. 25, 2020).

[57] “sensor IMU 6050 -GY521.” .

[58] “Plano inclinado acelorómetro.” https://www.luisllamas.es/como-usar-un-acelerometro-arduino/ (accessed Dec. 12, 2020).

[59] I. Del VillarFernández, “Diseño e implementación en PCB de un robot auto-balanceado mediante Arduino con módulo inalámbrico Grado en Ingeniería Eléctrica y Electrónica,” 2015.

[60] “Calibración IMU.” https://www.hispavila.com/el-sensor-mpu-6050/ (accessed Sep. 12, 2020).

[61] “Eliminación de ruido : IMU.” https://programarfacil.com/podcast/eliminar-el-ruido-en-arduino/ (accessed Sep. 16, 2020).

[62] “Calibration IMU.” Arduino script for MPU-6050 auto-calibration – 42 Bots) (accessed Sep. 23, 2020).

[63] “IMU 9250.” https://www.luisllamas.es/usar-arduino-con-los-imu-de-9dof-mpu-

9150-y-mpu-9250/ (accessed Feb. 10, 2021).

[64] “Sistema operativo Raspbian.” https://www.ecured.cu/Raspbian (accessed Sep. 25, 2020).

[65] “Raspbian image.” https://www.luisllamas.es/instalar-raspbian-en-raspberry-pi-con-etcher/ (accessed Sep. 23, 2020).

[66] “Calibration IMU MPU 6050.” https://codebender.cc/sketch:97892#MPU6050 calibration.ino (accessed Dec. 08, 2020).

[67] “Aceleración y orientación.” https://robologs.net/2014/10/15/tutorial-de-arduino-y-mpu-6050/ (accessed Dec. 10, 2020).

[68] B. Ave, D. Number, and R. Date, “MPU 6050 - Datasheet,” 2013. [Online]. Available: www.inversense.com.

[69] “Brújula.” https://de-senderismo.net/blog/como-funciona-la-brujula/ (accessed Jan. 12, 2021).

[70] S. El, “Lección 7 – Redacción del proyecto Contenidos ▪ Introducción ▪ Normas UNE-ISO de la serie 157000 ▪ Documentos según Norma UNE 157001 : 2014 ▪ Documentos con entidad propia.” pp. 1–49, 2014.

Documento Nº2:

ANEXOS

ÍNDICE DE ANEXOS

ANEXO 1: VISIÓN ARTIFICIAL

Se adjuntan las máscaras utilizadas para el procesamiento de imágenes

Primera máscara

```
function [BW,maskedRGBImage] = createMaskTry(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%   [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB
using
%   auto-generated code from the colorThresholder app. The
colorspace and
%  range for each channel of the colorspace were set within the
app. The
%  segmentation mask is returned in BW, and a composite of the
mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 29-Jun-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.286;
channel1Max = 0.978;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 0.452;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 0.634;
```

```
% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max)
& ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Segunda máscara

```
function [BW,maskedRGBImage] = createMaskTry2(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%   [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB
using
%   auto-generated code from the colorThresholder app. The
colorspace and
%  range for each channel of the colorspace were set within the
app. The
%  segmentation mask is returned in BW, and a composite of the
mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 29-Jun-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);
```

```
% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.633;
channel1Max = 0.845;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.280;
channel2Max = 0.602;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.000;
channel3Max = 0.758;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max)
& ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

Tercera máscara

```
function [BW,maskedRGBImage] = BlackMask(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%   [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB
using
%   auto-generated code from the colorThresholder app. The
colorspace and
```

```
%  range for each channel of the colorspace were set within the
app. The
%  segmentation mask is returned in BW, and a composite of the
mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 15-Apr-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.000;
channel1Max = 1.000;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.000;
channel2Max = 0.844;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.032;
channel3Max = 0.263;

% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max)
& ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;
```

```
end
```

Cuarta máscara

```
function [BW,maskedRGBImage] = createMaskP4(RGB)
%createMask  Threshold RGB image using auto-generated code from
colorThresholder app.
%   [BW,MASKEDRGBIMAGE] = createMask(RGB) thresholds image RGB
using
%   auto-generated code from the colorThresholder app. The
colorspace and
%  range for each channel of the colorspace were set within the
app. The
%  segmentation mask is returned in BW, and a composite of the
mask and
%  original RGB images is returned in maskedRGBImage.

% Auto-generated by colorThresholder app on 02-Jul-2020
%------------------------------------------------------


% Convert RGB image to chosen color space
I = rgb2hsv(RGB);

% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.588;
channel1Max = 0.956;

% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.097;
channel2Max = 0.484;

% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.032;
channel3Max = 0.312;
```

```
% Create mask based on chosen histogram thresholds
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max)
& ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Initialize output masked image based on input image.
maskedRGBImage = RGB;

% Set background pixels where BW is false to zero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;

end
```

ANEXO 2: CALIBRACIÓN DEL IMU MPU-6050

Se adjunta el programa para la calibración del sensor IMU MPU-6050.

```
#include <Wire.h>
#include <MPU6050.h>
#include <I2Cdev.h>
////////////////////////////////   CONFIGURATION   //////////////////////////////
//Change this 3 variables if you want to fine tune the skecth to your needs.
int buffersize=1000;     //Amount of readings used to average, make it higher to get more precision but sketch will be slower  (default:1000)
int acel_deadzone=8;        //Acelerometer error allowed, make it lower to get more precision, but sketch may not converge  (default:8)
int giro_deadzone=1;      //Giro error allowed, make it lower to get more precision, but sketch may not converge  (default:1)

// default I2C address is 0x68
```

```
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
//MPU6050 accelgyro;
MPU6050 accelgyro(0x68); // <-- use for AD0 high

int16_t ax, ay, az,gx, gy, gz;

int mean_ax,mean_ay,mean_az,mean_gx,mean_gy,mean_gz,state=0;
int ax_offset,ay_offset,az_offset,gx_offset,gy_offset,gz_offset;

//////////////////////////////////   SETUP   //////////////////////////////////
void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  Wire.begin();
  // COMMENT NEXT LINE IF YOU ARE USING ARDUINO DUE
  TWBR = 24; // 400kHz I2C clock (200kHz if CPU is 8MHz). Leonardo measured 250kHz.
  // initialize serial communication
  Serial.begin(9600);

  // initialize device
  accelgyro.initialize();

  // wait for ready
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available()){
    Serial.println(F("Send any character to start sketch.\n"));
```

```
    delay(1500);
  }
  while (Serial.available() && Serial.read()); // empty buffer again

  // start message
  Serial.println("\nMPU6050 Calibration Sketch");
  delay(2000);
  Serial.println("\nYour MPU6050 should be placed in horizontal position, with package letters facing up. \nDon't touch it until you see a finish message.\n");
  delay(3000);
  // verify connection
  Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050 connection failed");
  delay(1000);
  // reset offsets
  accelgyro.setXAccelOffset(0);
  accelgyro.setYAccelOffset(0);
  accelgyro.setZAccelOffset(0);
  accelgyro.setXGyroOffset(0);
  accelgyro.setYGyroOffset(0);
  accelgyro.setZGyroOffset(0);
}

/////////////////////////////////   LOOP   /////////////////////////////////
void loop() {
  if (state==0){
    Serial.println("\nReading sensors for first time...");
    meansensors();
    state++;
```

```
  delay(1000);
}
if (state==1) {
  Serial.println("\nCalculating offsets...");
  calibration();
  state++;
  delay(1000);
}
if (state==2) {
  meansensors();
  Serial.println("\nFINISHED!");
  Serial.print("\nSensor readings with offsets:\t");
  Serial.print(mean_ax);
  Serial.print("\t");
  Serial.print(mean_ay);
  Serial.print("\t");
  Serial.print(mean_az);
  Serial.print("\t");
  Serial.print(mean_gx);
  Serial.print("\t");
  Serial.print(mean_gy);
  Serial.print("\t");
  Serial.println(mean_gz);
  Serial.print("Your offsets:\t");
  Serial.print(ax_offset);
  Serial.print("\t");
  Serial.print(ay_offset);
  Serial.print("\t");
```

```
    Serial.print(az_offset);
    Serial.print("\t");
    Serial.print(gx_offset);
    Serial.print("\t");
    Serial.print(gy_offset);
    Serial.print("\t");
    Serial.println(gz_offset);
    Serial.println("\nData is printed as: acelX acelY acelZ giroX giroY giroZ");
    Serial.println("Check that your sensor readings are close to 0 0 16384 0 0 0");
    Serial.println("If calibration was succesful write down your offsets so you can set them
in your projects using something similar to mpu.setXAccelOffset(youroffset)");
    while (1);
  }
}
////////////////////////////////////   FUNCTIONS   ////////////////////////////////////
void meansensors(){
  long i=0,buff_ax=0,buff_ay=0,buff_az=0,buff_gx=0,buff_gy=0,buff_gz=0;

  while (i<(buffersize+101)){
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    if (i>100 && i<=(buffersize+100)){ //First 100 measures are discarded
      buff_ax=buff_ax+ax;
      buff_ay=buff_ay+ay;
      buff_az=buff_az+az;
      buff_gx=buff_gx+gx;
      buff_gy=buff_gy+gy;
```

```
      buff_gz=buff_gz+gz;
    }
    if (i==(buffersize+100)){
      mean_ax=buff_ax/buffersize;
      mean_ay=buff_ay/buffersize;
      mean_az=buff_az/buffersize;
      mean_gx=buff_gx/buffersize;
      mean_gy=buff_gy/buffersize;
      mean_gz=buff_gz/buffersize;
    }
    i++;
    delay(2); //Needed so we don't get repeated measures
  }
}
void calibration(){
  ax_offset=-mean_ax/8;
  ay_offset=-mean_ay/8;
  az_offset=(16384-mean_az)/8;

  gx_offset=-mean_gx/4;
  gy_offset=-mean_gy/4;
  gz_offset=-mean_gz/4;
  while (1){
    int ready=0;
    accelgyro.setXAccelOffset(ax_offset);
    accelgyro.setYAccelOffset(ay_offset);
    accelgyro.setZAccelOffset(az_offset);
```

```
    accelgyro.setXGyroOffset(gx_offset);
    accelgyro.setYGyroOffset(gy_offset);
    accelgyro.setZGyroOffset(gz_offset);

    meansensors();
    Serial.println("...");

    if (abs(mean_ax)<=acel_deadzone) ready++;
    else ax_offset=ax_offset-mean_ax/acel_deadzone;

    if (abs(mean_ay)<=acel_deadzone) ready++;
    else ay_offset=ay_offset-mean_ay/acel_deadzone;

    if (abs(16384-mean_az)<=acel_deadzone) ready++;
    else az_offset=az_offset+(16384-mean_az)/acel_deadzone;

    if (abs(mean_gx)<=giro_deadzone) ready++;
    else gx_offset=gx_offset-mean_gx/(giro_deadzone+1);

    if (abs(mean_gy)<=giro_deadzone) ready++;
    else gy_offset=gy_offset-mean_gy/(giro_deadzone+1);

    if (abs(mean_gz)<=giro_deadzone) ready++;
    else gz_offset=gz_offset-mean_gz/(giro_deadzone+1);

    if (ready==6) break;
  }
}
```

Documento Nº3:

PLIEGO DE CONDICIONES

ÍNDICE DEL PLIEGO DE CONDICIONES

1. DISPOSICIONES GENERALES

1.1 RESUMEN DEL PROYECTO

El proyecto se basa en implementar un sistema de posicionamiento y localización para un tipo de robot segador autónomo en entornos exteriores. Para ello se buscan varias soluciones.

En primer lugar, se programa un sistema basado en inteligencia artificial capaz de analizar el entorno donde se encuentra y ubicar ciertas marcas en el jardín.

En segundo lugar, se halla la orientación de nuestro robot gracias a un sensor inercial que nos dará estos datos.

Por último, se calcula la distancia a dicha marca en el jardín con la ayuda de un sensor de distancia.

1.2 ALCANCE Y APLICABILIDAD DEL PLIEGO DE CONDICIONES

El pliego de condiciones establece las condiciones técnicas, económicas, administrativas, facultativas y legales para que el objeto del Proyecto pueda materializarse en las condiciones especificadas, evitando posibles interpretaciones diferentes de las deseadas tal y como se cita en [70].

En este pliego encontrarás todas las condiciones resolutorias para cualquier problema que se plantee en la ejecución del proyecto.

2. CONDICIONES TÉCNICAS

2.1 ESPECIFICACIONES DEL SISTEMA

El sistema está diseñado para ser alimentado por a 5V, tanto la Raspberry como el Arduino, por lo tanto, se necesita un transformador que cumpla estas condiciones.

Por otro lado, se necesita un ordenador para transferir los datos de la placa Arduino mediante el puerto serial, a no ser que se implemente un módulo wifi para conectarlo de forma inalámbrica. Hay que tener en cuenta que hace falta wifi para poder conectar estos dispositivos al ordenador y obtener todos sus datos.

También se dispone de dos resistencias para conectarlas al sensor ultrasonidos para intentar regular la tensión ya que el sensor no puede conectarse directamente a 5V. En el caso de la placa Arduino el sensor que implementamos cuenta con un regulador de tensión por lo que no es necesario añadir ninguna resistencia.

2.2 CONDICIONES DE EJECUCIÓN

El sistema deberá basarse en el diseño tal y como viene descrito en el documento nº1 (Memoria). Cualquier cambio tiene que estar debidamente testeado anteriormente para asegurar que no pones en peligro la integridad de los dispositivos.

2.3 CONDICIONES DE USO

Cualquiera de los dispositivos que componen en el proyecto no pueden conectarse a una tensión superior de la especificada que establece el fabricante en sus documentos técnicos[55][68]. Si no se cumple esto puede ocurrir un deterioro o falla de cualquiera de los dispositivos en el circuito.

A la hora del montaje debes seguir las especificaciones que se plantean en el documento 1, en la implementación donde especifico el diseño del circuito y su instalación.

3.CONDICIONES LEGALES

3.1 USO DEL SISTEMA

El sistema de posicionamiento y localización que se ha diseñado es simplemente un prototipo para poder testear y realizar pruebas experimentales, por lo que no se posee ningún permiso legal para poder ser implementado para su comercialización.

El sistema puede ser utilizado para pruebas experimentales apoyando a estudios, pero siempre bajo la responsabilidad de uno mismo. En el caso del uso incorrecto o indebido de dicho sistema, el proyectante no tendrá ninguna responsabilidad civil ni penal.

3.2 PROPIEDAD INTELECTUAL

La propiedad intelectual corresponde al proyectante como a su institución. Asimismo, las partes de este proyecto en el que están involucradas trabajos externos pertenecen a su autor original.

4. CONDICIONES ECONÓMICAS

El sistema resultante del presente proyecto posee carácter de prototipo, por lo que no está disponible para su comercialización.

Por otro lado, se valora una futura explotación del diseño, siempre y cuando sea con fines de investigación o de mejora. Las partidas presupuestarias que puedan surgir a partir de esto se reflejarán en el marco de la futura investigación.

Documento Nº4:

PRESUPUESTO

ÍNDICE DEL PRESUPUESTO

1. COSTES DIRECTOS

1.1 MANO DE OBRA DIRECTA

Este proyecto requiere contratar a un ingeniero técnico que sea capaz de diseñar, desarrollar, realizar pruebas con el prototipo y elaborar la documentación adecuada sobre ello.

A continuación, vamos a establecer los tiempos aproximados invertidos en el proyecto teniendo en cuenta que su sueldo bruto es 15€/hora.

Trabajo desempeñado	Nº de horas	Total
Estudio y documentación	50	750€
Diseño y programación	50	750€
Implementación del circuito	30	450€
Pruebas del dispositivo	18	270€
Elaboración de la memoria	40	600€
TOTAL	188	2820€

Tabla 8. Presupuesto mano de obra directa.

1.2 MATERIAS PRIMAS

A continuación, vamos a mostrar un listado de los componentes utilizados en el proyecto y su precio unitario.

Producto	Unidades	€/ud.	Precio total
Placa protoboard	1	4,95	4,95€
Resistencias	2	0,153	0,306€
Cables	9	0,10	0,90€
Sensor MPU6050-Módulo GY 521	1	9,90	9,90€
Sensor HC-SR04	1	1,95	1,95€
Tarjeta microSD 128Gb	1	20	20€
Placa ELEGOO	1	22	22€
Placa Raspberry Pi	2	38	76€
TOTAL			**136,006€**

Tabla 9. Presupuesto de materias primas.

1.3 PUESTO DE TRABAJO Y COSTE DE EQUIPO

Para calcular los gastos en el puesto de trabajo debemos tener en cuenta tanto las herramientas usadas como las horas que hemos invertido en el proyecto.

En las herramientas que hemos usado están por un lado el software que debemos tener en cuenta ya que para la realización y programación hemos utilizado dos tipos:

- Arduino IDE: se trata de un software libre y por lo tanto no nos genera ningún gasto.
- Matlab: Es un software que requiere licencia, su coste de licencia individual son 800€, pero debido a que mi universidad tiene un convenio con dicho programa no nos ha generado ningún coste.

Por otro lado, está el hardware y el consumo energético que conlleva utilizar todos los aparatos informáticos. En mi caso he utilizado mi ordenador personal en todo el proyecto y debemos tener en cuenta su consumo y el desgaste durante el proceso.

A continuación, vamos a calcular el coste energético sabiendo que en la actualidad el consumo medio de energía en España es de 0,127 €/KWh:

$$Coste\ energético\ =\ 1kWh\ \times\ 0.127\frac{€}{kWh}\ \times\ 188\,h\ =\ 23{,}88\,€$$

Sabiendo que la inversión en nuestro equipo informático es de 799€, vamos a calcular el coste de amortización teniendo en cuenta su desgaste. El tiempo de uso ha sido de 10 meses y suponemos que su vida útil será de 6 años.

$$Coste\ de\ Amortización\ =\ \frac{Tiempo\ de\ uso}{Ciclo\ vida\ útil}\ \times\ Inversión\ del\ Equipo$$

$$Coste\ de\ Amortización\ =\ \frac{10\ meses}{6\ años\ \times\ 12\frac{meses}{año}}\ \times\ 799€\ =\ 110{,}97\,€$$

Se muestra una tabla con todos los gastos del puesto de trabajo:

Útil	Coste
Coste energético	23,88 €
Coste de amortización	110,97€
Software	0€
TOTAL	**134,85 €**

Tabla 10. Presupuesto del puesto de trabajo.

2. COSTES INDIRECTOS

Los costes indirectos incluyen los gastos de mano de obra indirecta, gastos generales y gastos sociales.

- Mano de obra indirecta: deberían adjuntarse los gastos de aquellas personas involucradas en la fabricación del producto.
- Gastos generales: Reflejan los costes por mantenimiento de la instalación donde realizas el proyecto.
- Gastos sociales: Incluyen los recursos invertidos en investigación y planificación. También tendrá incluido la seguridad social de los trabajadores.

En nuestro caso estos costes no nos afectan, ya que no hemos trabajado en el entorno de un laboratorio o una empresa como para generar dichos gastos.

3. PRESUPUESTO TOTAL

Se muestra el presupuesto total del proyecto:

Concepto	Coste
Mano de Obra directa	2820€
Materia prima	136,01€
Puesto de trabajo	134,85 €
Costes indirectos	0€
Total	**3090,85 €**

Tabla 11. Presupuesto total.

El presupuesto total del proyecto es de 3090,85 €.