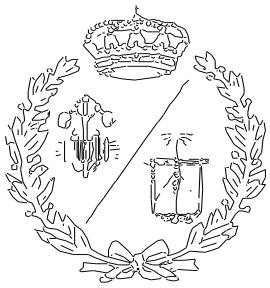
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA



Proyecto Fin de Grado

MODELO VIRTUAL Y CONTROL DE BRAZO ROBÓTICO MEDIANTE 3DEXPERIENCE Y ARDUINO

(Virtual Model and Control of Robotic Arm through 3DEXPERIENCE and ARDUINO)

Para acceder al Título de

GRADUADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: Andrés Medrano Montero

Septiembre-2019

ÍNDICE

RESUMEN **3**

ABSTRACT **4**

TABLA DE ILUSTRACIONES **5**

1 INTRODUCCIÓN **7**

1.1 OBJETIVO 7
1.2 UN PROYECTO MULTIDISCIPLINAR 7
1.3 INDUSTRIA 4.0, TECNOLOGÍAS 3D Y SISTEMAS PLM 7
1.4 3DEXPERIENCE 9
1.5 ARDUINO 12
1.6 PLATAFORMA 3DEXPERIENCE Y ARDUINO 18
1.7 INGENIERÍA DE SISTEMAS CON CATIA SYSTEMS 19
1.8 DEFINICIÓN DE RFLP 19
1.9 GESTIÓN DE PROYECTOS 20
1.10 MODELO VIRTUAL Y MODELO REAL 21

2 INTERNET OF THINGS **21**

2.1 DEFINICIÓN DE INTERNET OF THINGS (IOT) 21
2.2 INTEGRACIÓN ENTRE EL MUNDO FÍSICO Y LOS SISTEMAS BASADOS EN COMPUTADOR 22

3 CONTROL DE UN SERVOMOTOR **23**

3.1 QUÉ ES Y CÓMO FUNCIONA UN SERVOMOTOR 23
3.2 ARDUINO 24
3.2.1 Conexión del servomotor a Arduino *24*
3.2.2. Lanzamiento del software de Arduino *25*
3.2.3. Programas que utiliza Arduino *25*
3.3 3DEXPERIENCE 36
3.3.1 MODELICA y CATIA *37*
3.3.2 Modelo Virtual *38*
3.3.3 PID *40*
3.3.4 Simulación *45*

4 CONTROL DEL BRAZO ROBÓTICO **48**

4.1 PARTES DEL BRAZO ROBÓTICO 48
4.2 ESPECIFICACIONES DEL BRAZO ROBÓTICO 50
4.3 HERMANAMIENTO DIGITAL DE BRAZO ROBÓTICO INDUSTRIAL EN MINIATURA 50
4.4 RANGOS DE POSICIÓN DE CADA GRADO DE LIBERTAD 52

4.5 CONEXIÓN DEL BRAZO ROBÓTICO A ARDUINO 54
4.6 DIFERENCIAS ENTRE EL CONTROL CON ARDUINO Y EL CONTROL CON 3DEXPERIENCE 55
4.7 FUNCIONAMIENTO DEL BLOQUE LÓGICO DE ARDUINO Y DE LA FMI 56
4.8 SIMULACIÓN 57

5 LÍNEAS FUTURAS 59

6 BIBLIOGRAFÍA 60

RESUMEN

El presente proyecto está vinculado con las tecnologías asociadas a lo que se viene denominando la Industria 4.0 (I4.0) o Industria del Futuro (IoF) y, más concretamente, con la aplicación conjunta de Tenologías de Diseño 3D y las técnicas de Ingeniería de Sistemas y Automática.

En este caso, de forma más concreta, se utilizará la plataforma PLM 3DEXPERIENCE para desarrollar un modelo virtual (Gemelo Virtual) de un pequeño brazo robot (de cinco grados de libertad más una pinza servo-controlada), así como el sistema de control que deberá proporcionar las señales adecuadas para controlar la posición final del robot virtual.

A través de un estudio en detalle de la plataforma 3DEXPERIENCE implementada por la empresa Dassault Systèmes se pueden conocer las aplicaciones y capacidades que ofrece en el diseño digital y el desarrollo de productos posteriores.

Para el control del robot real se estudiará el uso y los algoritmos de Arduino como interfaz entre 3DEXPERIENCE y el brazo robótico.

ABSTRACT

The current project is vinculated to the technologies associated to the denominated as Industry 4.0 (I4.0) or Industry of Future (IoF), and specifically, with the united application of 3D Design Technologies and the Systems and Automatic Engineering Technologies.

In this case, more concretely, will be used PLM platform 3DEXPERIENCE to develop a virtual model (virtual twin) of a small robotic arm (five degrees of freedom plus a servo-controlled gripper), as well as the control system that should provide the appropriate signals to control the final position of the virtual robot.

Through a detailed study of 3DEXPERIENCE platform developed by Dassault Systèmes, can learn about the applications and capabilities it offers in digital design and the development of subsequent products.

For the control of the real robot, the use and algorithms of Arduino as an interface between 3DEXPERIENCE and the robotic arm will be studied.

TABLA DE ILUSTRACIONES

Ilustración 1. Brújula de la plataforma 3DEXPERIENCE 10
Ilustración 2. Placa Arduino Uno 12
Ilustración 3. Distribución de pines de Arduino Uno 13
Ilustración 4. Representación mediante interruptores de una I/O digital en un microcontrolador de Arduino 15
Ilustración 5.. Representación de diiferentes valores de duty cicle 17
Ilustración 6. Interfaz de programación de Arduino 18
Ilustración 7. Metodología RFLP 19
Ilustración 8. Ciclo RFLP 20
Ilustración 9. Proceso RFLP 20
Ilustración 10. Enfoque iterativo RFLP 21
Ilustración 11. Servomotor 23
Ilustración 12. Puertos GND de Arduino 24
Ilustración 13. Puerto Vin de Arduino 24
Ilustración 14. Pin 2 de Arduino 25
Ilustración 15. Icono de Arduno y su COM Number 25
Ilustración 16. Selección del compilador de C 37
Ilustración 17. Notificación del correcto funcionamiento del compilador 37
Ilustración 18. Modelo Virtual para el control de un servomotor 38
Ilustración 19. Bloque lógico de la señal de entrada 39
Ilustración 20. Bloque lógico del controlador PID 40
Ilustración 21. Diagrama de bloques del sistema 41
Ilustración 22. Gráfica de respuestas con diferentes valores de Kp 42
Ilustración 23. Gráfica de respuestas con diferentes valores de Ki 43
Ilustración 24. Gráfica de respuestas con diferentes valores de Kd 44
Ilustración 25. Bloque lógico del servomotor 45
Ilustración 26. Evolución de señales del PID (1) 46
Ilustración 27. Evolución de señales del PID (2) 46
Ilustración 28. Evolución de señales del PID (3) 47
Ilustración 29. Evolución de señales del PID (4) 47
Ilustración 30. Evolución de señales del PID (5) 48
Ilustración 31. Patyes del brazo articulado 49
Ilustración 32. Giros de la muñeca 49

Ilustración 33. Modelo virtual del brazo robótico 51
Ilustración 34. Rango de giro (1) 52
Ilustración 35. Rango de giro (2) 52
Ilustración 36. Rango de giro (3) 53
Ilustración 37. Rango de giro (4) 53
Ilustración 38. Rango de giro (5) 53
Ilustración 39. Rango de apertura de la pinza 54
Ilustración 40. Conexión de las masas de los servomotores 54
Ilustración 41. Pines de los cables de posición de los motores 55
Ilustración 42. Entorno virtual del brazo robótico 58
Ilustración 43. Simulador del robot virtual 58

1 INTRODUCCIÓN

1.1 OBJETIVO

Conocer el "Internet of Things" con 3DEXPERIENCE para implementar mecatrónica virtual o teleoperada.

Con la plataforma 3DEXPERIENCE, se puede controlar el brazo robótico y su modelo virtual, usando ARDUINO como interfaz. La conexión entre ambos procesos permite que 3DEXPERIENCE controle el robot y devuelva una realimentación para actualizar el modelo.

Por tanto, el objetivo es aprender, por una parte, a entender el proceso genérico de conectar y operar dispositivos de laboratorio a sus "Virtual Twins" (Gemelos Virtuales) en 3DEXPERIENCE a través de ARDUINO o microcontroladores similares, y, por otra parte, a conectar CATIA y ARDUINO para un control bidireccional y realimentación (Software/Hardware/Model-in-the-loop).

1.2 UN PROYECTO MULTIDISCIPLINAR

Los sistemas mecatrónicos requieren habilidades en varias disciplinas: Mecatrónica, Electrónica, Computación y otros muchos campos.

Durante el diseño, estas disciplinas interactúan. Es importante administrar el proceso de ingeniería multidisciplinaria para garantizar que el producto concebido satisfaga las necesidades del cliente.

Es necesario tener:

- Una visión común para compartir el mismo entendimiento.
- Un proceso estructurado de desarrollo para integrar todas las disciplinas y los especialistas en el mismo equipo.

1.3 INDUSTRIA 4.0, TECNOLOGÍAS 3D Y SISTEMAS PLM

Debido a los exigentes requisitos de un mercado globalizado por los que las empresas se ven afectadas, como la competitividad en el precio, la globalización del diseño o la necesidad de ingeniería concurrente, las industrias se ven obligadas a aplicar un conjunto de tecnologías novedosas. Esta innovación en el campo de las tecnologías digitales y la inteligencia artificial se denomina "Industria 4.0" (I4.0) o "Industria del Futuro" (IoF) en Europa.

Para establecer la Industria 4.0 es necesaria la utilización combinada de un conjunto de tecnologías entre las que se encuentran las Tecnologías-3D (T-3D) y las Técnicas y Tecnologías vinculadas con el

área de conocimiento de Ingeniería de Sistemas y Automática (TT-ISA). Estos dos métodos utilizaban hasta hace muy poco entornos de diseño y simulación muy diferentes y resultaba complicado conseguir un diseño integrado del producto y su correspondiente proceso de fabricación y servicio.

Aquí es donde entran las herramientas software de gestión de ciclo de vida del producto (PLM) que permiten integrar ambas tecnologías sobre una misma plataforma.

La utilización de plataformas PLM presenta la ventaja de que, además de poder ser utilizadas para diseñar experiencias prácticas formativas vinculadas con los contenidos propios de las asignaturas de automática, al utilizarlas se está formando un entorno colaborativo cada vez más empleado en ingeniería y que es pieza clave en la Industria 4.0.

El objetivo de la Industria del Futuro es crear un modelo de fabricación con gran capacidad de adaptación y flexibilidad, de elevado rendimiento y mejorando tanto la ergonomía en los puestos de trabajos como la seguridad de las personas.

La Industria 4.0 está basada es seis pilares básicos:

- Automatizada: búsqueda de la mayor automatización posible en base a PLCs y Robots Industriales que incorporan sensores para interactuar de forma más inteligente y coordinada con el entorno durante el proceso de fabricación.
- Flexible: sistemas de producción idóneos para la adaptación a nuevos productos, cantidades y plazos según las necesidades del cliente.
- Inteligente: dotación de cierto nivel de inteligencia artificial en los recursos de fabricación de forma que puedan afrontar decisiones en función de los cambios del entorno.
- Sostenible: minimización de la contaminación, ahorro de energía y materiales.
- Conectada: tanto los recursos como las personas están conectadas con los centros de producción, diseño y gestión, independientemente de la distancia a la que se encuentren.
- Colaborativa: utilización de plataformas colaborativas para potenciar la colaboración al máximo nivel entre las personas y todos los agentes implicados en el ciclo de vida del producto; y espacios de colaboración entre personas y máquinas para el desarrollo de tareas conjuntas, asegurando la total seguridad de ambas.

Las tecnologías 3D son aquellas que se ocupan de la ingeniería virtual, es decir, de la tecnología que utiliza computadores para la simulación física y geométrica de sistemas reales.

En la actualidad, es imprescindible el uso de estas tecnologías para el diseño, desarrollo y fabricación en la ingeniería. Forman parte de ellas el software de diseño asistido por ordenador (CAD), la Ingeniería Asistida por Ordenador (CAE) y la Fabricación Asistida por Ordenador (CAM).

Hoy en día los equipos multidisciplinarios deben trabajar juntos para diseñar y crear nuevos productos y usualmente utilizan el software de ingeniería asistida por computadora (CAE) para lograr sus metas en el diseño. Es un elemento clave de diseño porque permite a los ingenieros simular y probar diseños en diferentes campos (mecánicos, termodinámicos, eléctricos y electrónicos, sistemas de control, etc).

La gestión de datos de producto (PDM) surgió como la necesidad de proporcionar un acceso fácil y seguro a los datos creados durante el desarrollo del producto. PDM se centró en el dominio de la ingeniería, pero fracasó en otras actividades relacionadas con negocios tales como marketing, ventas, etc.

Los sistemas PLM (Product Lifecycle Management) surgieron a principios de los años 2000 y se propusieron como una herramienta para intercambiar información y trabajar juntos en el ciclo de vida completo del producto. Los objetivos del PLM tratan de cubrir todas las etapas del desarrollo del producto integrando los procesos y las personas que participan en el proyecto. Toma en cuenta la necesidad de una colaboración multidisciplinar, independientemente de la ubicación geográfica de las empresas y partes interesadas que intervienen en el proyecto.

Este procedimiento de ingeniería colaborativa permite a los ingenieros llevar a cabo tareas de superposición en paralelo, haciendo posible reducir tiempo y coste asociado a los diseños, y mejorar la calidad de los productos. PLM también incorpora ingeniería virtual, pero va más allá del mero aspecto de ingeniería del desarrollo de proyectos, sino que trata de gestionar toda la información del producto en todas las etapas del ciclo de vida.

Las principales ventajas que ofrece este enfoque son que ayuda a ofrecer productos y procesos más innovadores en un tiempo más corto, lo que acorta el tiempo al mercado. Además, establece una relación más extensa y colaborativa con todos los elementos de la cadena de valor, como clientes, proveedores y socios comerciales.

1.4 3DEXPERIENCE

Dassault Systèmes es una pionera en las plataformas B2B (*Business to Business*).

El término *Business to Business* hace referencia a las transacciones comerciales entre empresas, es decir, a aquellas que típicamente se establecen entre un fabricante y el distribuidor de un producto, o entre un distribuidor y un comercio minorista.

El término B2C (*Business to Client*) es aquella modalidad de transacción que vincula directamente a las empresas con los consumidores y puede corresponder a diversos modelos, como por ejemplo los basados en la publicidad, en la comunidad y en tarifas.

Durante los últimos años las Plataformas *Business to Client*, lideradas por potentes empresas como Amazon o Alibaba, han experimentado un elevado crecimiento en todo el mundo. Actualmente, esto está dando un giro hacia otra dirección donde los modelos de negocio de plataformas B2B, cada vez más en auge, les están quitando gran parte de protagonismo.

Según un estudio de la consultora tecnológica Frost&Sullivan, se calcula que para el año 2020 las plataformas *Business to Business* podrían llegar a alcanzar unas ventas de hasta 6,7 billones de dólares a nivel mundial, lo que supone el doble de lo proyectado para los modelos *Business to Client*.

Una de las empresas pioneras en este tipo de plataformas B2B es la multinacional francesa Dassault Systèmes, con su Plataforma 3DEXPERIENCE lanzada en el año 2012, y catalogada como una plataforma de experiencia comercial e innovación de productos.

La plataforma 3DEXPERIENCE es principalmente un sistema de gestión del ciclo de vida del producto (PLM) y tiene como objetivo apoyar el diseño digital y el desarrollo de productos que posteriormente se fabrican. Un usuario inicia sesión en una interfaz y puede acceder a todas las capacidades instaladas y con licencia de 3DEXPERIENCE desde ahí. La plataforma se basa en una base de datos, lo que permite el almacenamiento y la indexación de datos.

Dassault Systèmes explica su plataforma en términos de una brújula 3D ilustrada a continuación:

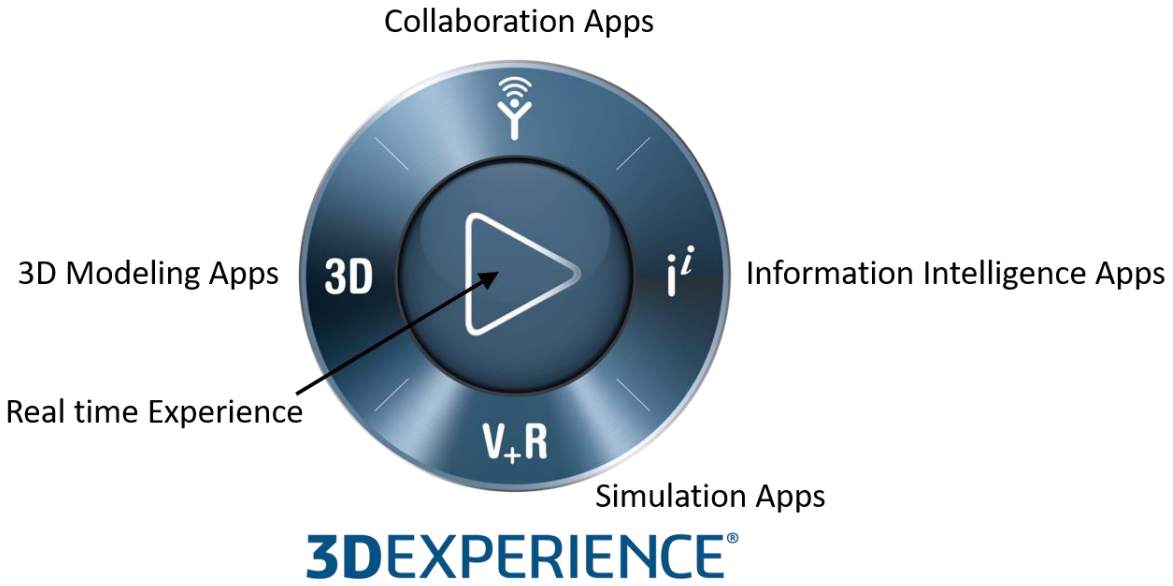


Ilustración 1. Brújula de la plataforma 3DEXPERIENCE

Los cuatro cuadrantes, comenzando en la parte superior y siguiendo en el sentido de las agujas del reloj:

- Las aplicaciones de colaboración incluyen funcionalidades que fomentan la colaboración informal entre equipos extendidos (SWYM) y la colaboración estructurada, como la gestión formal del cambio (ENOVIA).
- Las aplicaciones de inteligencia de información están diseñadas para manejar *Big Data*, que son los conjuntos de datos o combinaciones de conjuntos de datos cuyo tamaño (volumen), complejidad (variabilidad) y velocidad de crecimiento dificultan su captura, gestión, procesamiento o análisis mediante tecnologías y herramientas convencionales, tales como bases de datos y estadísticas convencionales. También a extraer de múltiples fuentes y presentar al usuario resúmenes concisos de la información que necesitan (NETVIBES).
- Las aplicaciones de simulación están destinadas a la validación digital virtual y la prueba de diseños. Tradicionalmente, esto se conoce como FEA (SIMULIA). También se pude extender a la simulación virtual de fábricas o diseños de tiendas minoristas.
- Las aplicaciones de modelado 3D son quizás por lo que Dassault Systèmes es más conocido e incluyen CATIA y Solidworks.

Debido a que toda esta funcionalidad está en una sola plataforma, esto le permite al usuario una experiencia realista e inmediata en un mundo virtual (experiencia en tiempo real).

Algunos puntos más pertinentes con respecto a la plataforma 3DEXPERIENCE:

- La plataforma es escalable para adaptarse a los requisitos de cada organización que utiliza la tecnología. Los adoptantes pueden elegir comenzar con la funcionalidad básica y luego pasar a capacidades más avanzadas.
- El concepto de una plataforma de diseño surgió de la necesidad de almacenar y mantener todos los datos digitales generados durante el desarrollo del producto después de la adopción generalizada de aplicaciones CAD. Ahora tiene amplias dimensiones más allá de eso.
- Dassault Systèmes agrega constantemente nuevas aplicaciones y funciones a la plataforma, por lo que el rango de capacidades se está expandiendo.
- Más y más de la plataforma se está moviendo a la nube. Dassault Systémes ahora ofrece un modelo SAAS completo para muchas aplicaciones dentro de la plataforma. Esto reduce drásticamente la complejidad de la implementación.

Integra los estándares de intercambio ISO STEP AP203/214/242, cubriendo la fuerte demanda de los clientes que necesitan desarrollar una estrategia de archivo a largo plazo. También incluye muchos estándares verticales como AUTOSAR (transporte y movilidad) o IFC (construcción), y estándares multisectoriales como MODELICA para generar contenidos sofisticados para sistemas físicos.

La revolucionaria plataforma está apoyada por un creciente número de aplicaciones y soluciones verticales para todas las industrias. Ateniendo a la demanda y a las contribuciones de los clientes, incluye una gama propietaria de 41 experiencias de solución verticales con sus correspondientes 183 procesos, además de una gama dedicada a la nube compuesta por 14 experiencias de solución, con más de 60 procesos, apropiadas para empresas de cualquier tamaño.

La plataforma 3DEXPERIENCE basada en la arquitectura V6 es un catalizador de la transformación empresarial que permite a las compañías conectar todos los puntos de la empresa extendida. Sus procesos de negocio probados y listos para usar permiten a las compañías beneficiarse rápidamente de las ventajas que ofrece una colaboración eficaz. Este es su aspecto más importante, ya que permite que todo lo que antes se hacía de manera aislada en los diferentes departamentos (diseño, fabricación, marketing o ingeniería) actúen simultáneamente y de manera colaborativa a través de esta plataforma permitiendo crear unas experiencias excepcionales en el consumidor.

Además, se podría decir que es la única plataforma realmente digital, ya que otro tipo de soluciones electrónicas únicamente almacenan archivos estáticos, pero no pueden funcionar como plataformas en tiempo real, como es el caso de la herramienta creada por Dassault Systèmes, que está disponible tanto en un entorno privado “on premise” como “on the cloud”.

1.5 ARDUINO

Es una placa con un microcontrolador (ATMEGA8 y ATMEGA168 de Atmel), un circuito integrado programable capaz de realizar operaciones matemáticas complejas a gran velocidad, conectado bajo la configuración de “sistema mínimo” sobre una placa de circuito impreso a la que se le pueden conectar placas de expansión (shields) a través de la disposición de los puertos de entrada y salida presentes en la placa seleccionada.

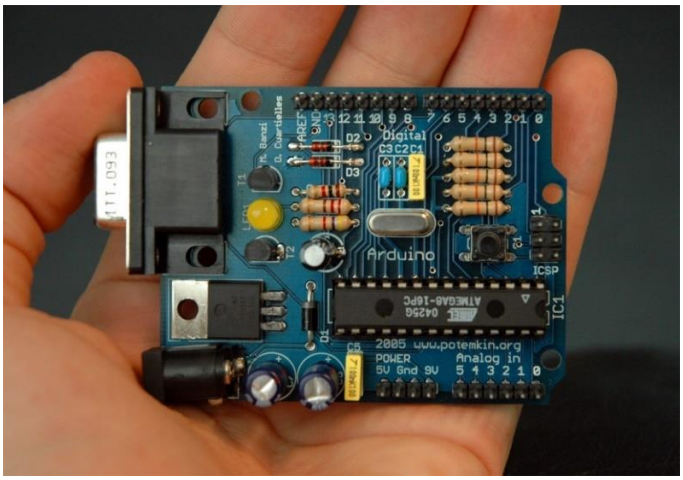


Ilustración 2. Placa Arduino Uno

Las shields complementan la funcionalidad del modelo de la placa empleada, agregando circuitería, sensores y módulos de comunicación externos a la placa original. Las shields más importantes de Arduino son:

- Ethernet Shield: la placa Arduino se comunica con el módulo W5100 y la micro-SD utilizando el bus SPI (mediante el conector ICSP). Esto se encuentra en los pines digitales 11, 12 y 13 en el modelo UNO. El pin 10 es utilizado para seleccionar el W5100 y el pin 4 para la micro-SD.
- Wifi Shield: permite conectar un Arduino a Internet mediante Wifi. También dispone de un slot para una tarjeta micro-SD. Esta shield se comunica con Arduino a través del bus SPI mediante los pines 4 y 10 al igual que la Ethernet Shield.
- GSM Shield: conecta Arduino a Internet mediante GPRS, usando una tarjeta SIM. También permite enviar y recibir mensajes y llamadas de voz, añadiendo un micrófono y un altavoz. Por el consumo de esta shield, se hace necesario alimentar a Arduino mediante una fuente externa y no desde un USB, ya que no es capaz de proporcionar suficiente energía.
- Motor Shield: permite manejar dos motores de corriente continua, controlando su dirección y velocidad. Está basado en un chip de doble puente. Este shield usa dos canales y cada canal usa cuatro pines, por lo que ocupa ocho pines del Arduino.

La principal diferencia para elegir la utilización de un microcontrolador (como es el caso de Arduino) por encima de un microprocesador, es la capacidad de entradas y salidas de las que se disponen, así como el mejor rendimiento de la CPU y el menor coste.

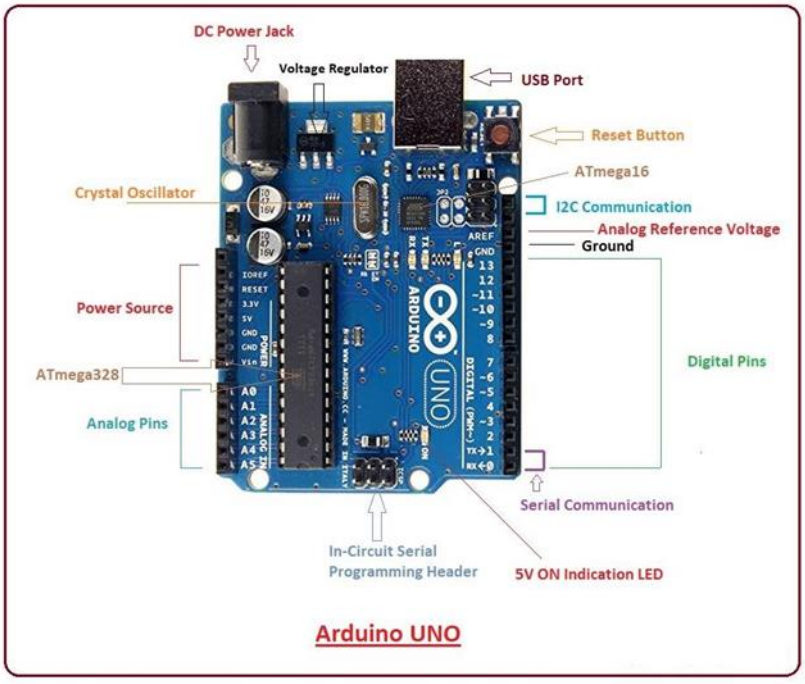


Ilustración 3. Distribución de pines de Arduino Uno

Esta placa es muy barata, utiliza código abierto (C++) y está basada en hardware y software flexibles y fáciles de usar. Cuenta con pines digitales y analógicos que pueden ser configurados con entradas o salidas. Así mismo, permite su conexión y energización en serie vía USB.

Contiene la siguiente distribución de pines:

- 14 pines digitales que pueden ser configurados como entradas o salidas, de los cuales 6 pueden ser utilizados como señales digitales PWM.
- 6 pines analógicos desde A0 hasta A5 para las entradas analógicas.
- 3 pines GND para conectar a tierra los circuitos.
- 2 pines de alimentación de 5 V y 3,3 V respectivamente.

La mayoría de las placas Arduino pueden ser programadas a través del puerto serie que incorporan haciendo uso del Bootloader que traen programado por defecto. El software de Arduino consiste en dos elementos: un entorno de desarrollo (IDE), basado en el entorno de processing y en la estructura de programación Wiring, y en el cargador de arranque (bootloader, por su traducción al inglés), que es ejecutado de forma automática dentro del microcontrolador en cuanto este se enciende.

Ya es conocido y adoptado en muchas instituciones tecnológicas y en la mayoría de las universidades e investigaciones.

Entradas y salidas digitales

Los sistemas digitales, como por ejemplo un microcontrolador, usan la lógica de dos estados representados por dos niveles de tensión eléctrica, uno alto, H, y otro bajo, L, (de High y Low, respectivamente, en inglés). Por abstracción, dichos estados se sustituyen por ceros y unos, lo que facilita la aplicación de la lógica y la aritmética binaria. Si el nivel alto se representa por 1 y el bajo por 0, se habla de lógica positiva y en caso contrario de lógica negativa.

En Arduino para tratar las entradas y salidas digitales se usan las siguientes funciones:

- pinMode() – configura en el pin especificado si se va a comportar como una entrada o como una salida.
- digitalWrite() – escribe un valor HIGH o LOW en el pin digital especificado. Si el pin está configurado como OUTPUT, pone la tensión correspondiente en el pin seleccionado. Si el pin está configurado como INPUT, habilita o deshabilita la resistencia interna de pull-up del correspondiente pin.
- digitalRead() – lee el valor del pin correspondiente como HIGH o LOW.

En la imagen siguiente se muestra el estado por defecto de una I/O digital en un microcontrolador de Arduino. Se ha simplificado con interruptores la compleja electrónica que hay dentro. Por defecto los pines digitales I/O están configurados como inputs en un estado de alta impedancia (equivalente a una resistencia de 100 MΩ en frente del pin), es decir, SW3 a ON y no hace falta llamar a la función pinMode() aunque es recomendable para aclarar el código.

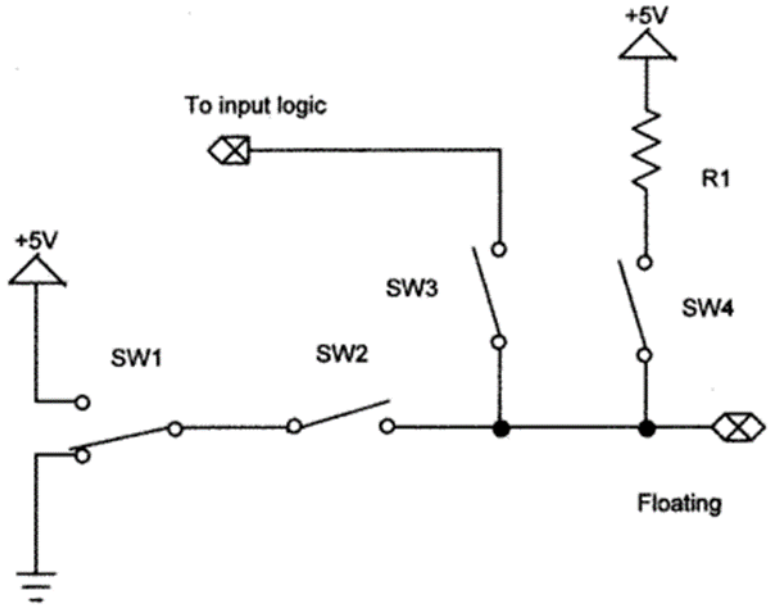


Ilustración 4. Representación mediante interruptores de una I/O digital en un microcontrolador de Arduino

- pinMode(x, INPUT) → SW3 = ON (resto a OFF). Los valores leídos serán aleatorios si el pin de Arduino está al aire. El pin está en un estado de alta impedancia.
- pinMode(x, INPUT_PULLUP) → SW3 = ON & SW4 = ON (resto a OFF). Los valores leídos sin nada conectado al pin es HIGH. La resistencia R1 tiene un valor dependiendo del microcontrolador, pero tiene un valor entre 20 kΩ y 150 kΩ.
- pinMode(x, OUTPUT) & digitalWrite(x, HIGH) → SW2 = ON & SW1 = 5 V (resto a OFF). Estado de baja impedancia, no hay resistencia interna y es necesario poner una resistencia adecuada a la salida del pin para no superar los 40mA (source) máximos admitidos.
- pinMode(x, OUTPUT) & digitalWrite(x, LOW) → SW2 = ON & SW1 = GND (resto a OFF). Estado de baja impedancia, no hay resistencia interna y es necesario poner una adecuada para no superar los 40 mA (sink) máximos admitidos.

Entradas y salidas analógicas

Una señal eléctrica analógica es aquella en la que los valores de la tensión varían constantemente y pueden tomar cualquier valor.

Un sistema de control (como un microcontrolador) no tiene capacidad alguna para trabajar con señales analógicas, de modo que necesita convertir las señales analógicas en señales digitales para poder trabajar con ellas.

La señal digital obtenida de una analógica tiene dos propiedades fundamentales:

- Valores: qué valor en voltios define 0 y 1. En este caso, es tecnología TTL (0 – 5 V).
- Resolución analógica: número de bits que se usan para representar como una notación digital una señal analógica.

En el caso de Arduino Uno, el valor de 0 V analógico es expresado en digital como B0000000000 (0) y el valor de 5 V analógico es expresado en digital como B1111111111 (1023). Por lo tanto, todo valor analógico intermedio es expresado con un valor entre 0 y 1023, es decir, sumo 1 en binario cada 4,883 mV.

Arduino Uno tiene una resolución de 10 bits.

Los microcontroladores de Arduino contienen en la placa un conversor analógico-digital de 6 canales. El conversor tiene una resolución de 10 bits, devolviendo enteros entre 0 y 1023. Los pines analógicos de Arduino también tienen todas las funcionalidades de los pines digitales. Por lo tanto, si se necesitan más pines digitales se pueden usar los pines analógicos. La nomenclatura para los pines analógicos es A0, A1, A2, A3, A4 y A5.

En Arduino para tratar las entradas y salidas analógicas usamos las siguientes funciones:

- analogReference() – configura la referencia de tensión usada para la entrada analógica.
- analogRead() – lee el valor del pin analógico especificado.
- analogWrite() – escribe un valor analógico (onda PWM) al pin especificado. No en todos los pines digitales se puede aplicar PWM.

Salida analógica PWM

Arduino Uno tiene entradas analógicas que gracias a los conversores analógico-digital hacen que el microcontrolador pueda entender ese valor, pero no tiene salidas analógicas puras y para solucionar esto, usa la técnica de PWM.

Las salidas PWM (Pulse Width Modulation) permiten generar salidas analógicas desde pines digitales.

La modulación por ancho de pulsos (PWM) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una sinusoidal o una cuadrada, por ejemplo), ya

sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el periodo (duty cicle = (tiempo que la salida está a uno o HIGH) / (periodo de la función).

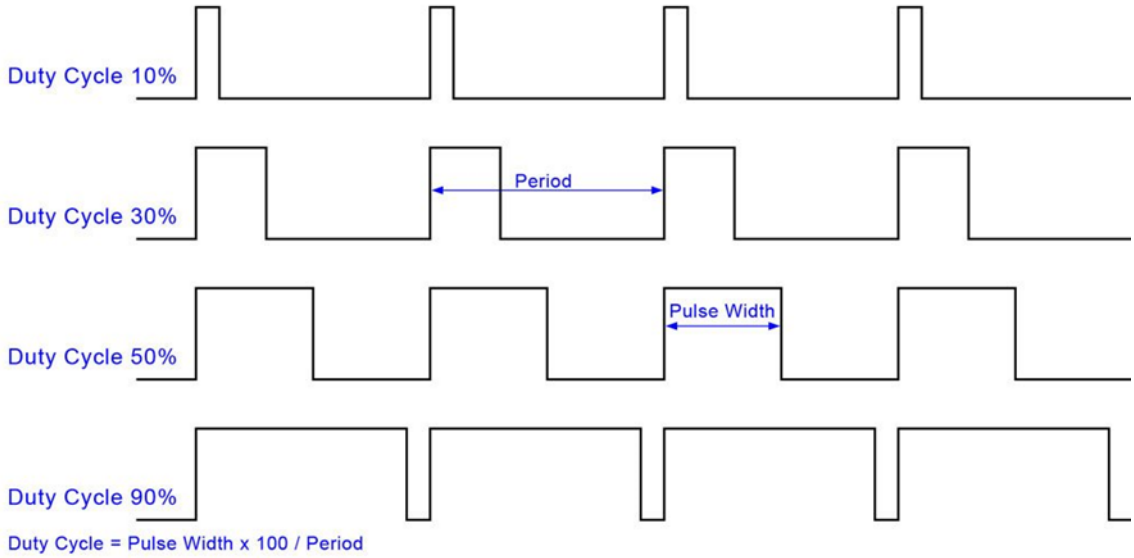


Ilustración 5.. Representación de diiferentes valores de duty cicle

En Arduino la frecuencia de PWM es de 500 Hz, pero es un valor que se puede modificar en caso de que sea necesario.

Software de Arduino

El software de Arduno es un IDE, entorno de desarrollo integrado (siglas en inglés de Integrated Development Environment). Es un programa informático compuesto por un conjunto de herramientas de programación.

El IDE de Arduino es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). Además, incorpora las herramientas para cargar el programa ya compilado en la memoria flash del hardware.

Los programas de Arduino están compuestos por un solo fichero con extensión ".ino", aunque es posible organizarlo en varios ficheros. El fichero principal siempre debe estar en una carpeta con el mismo nombre que el fichero.

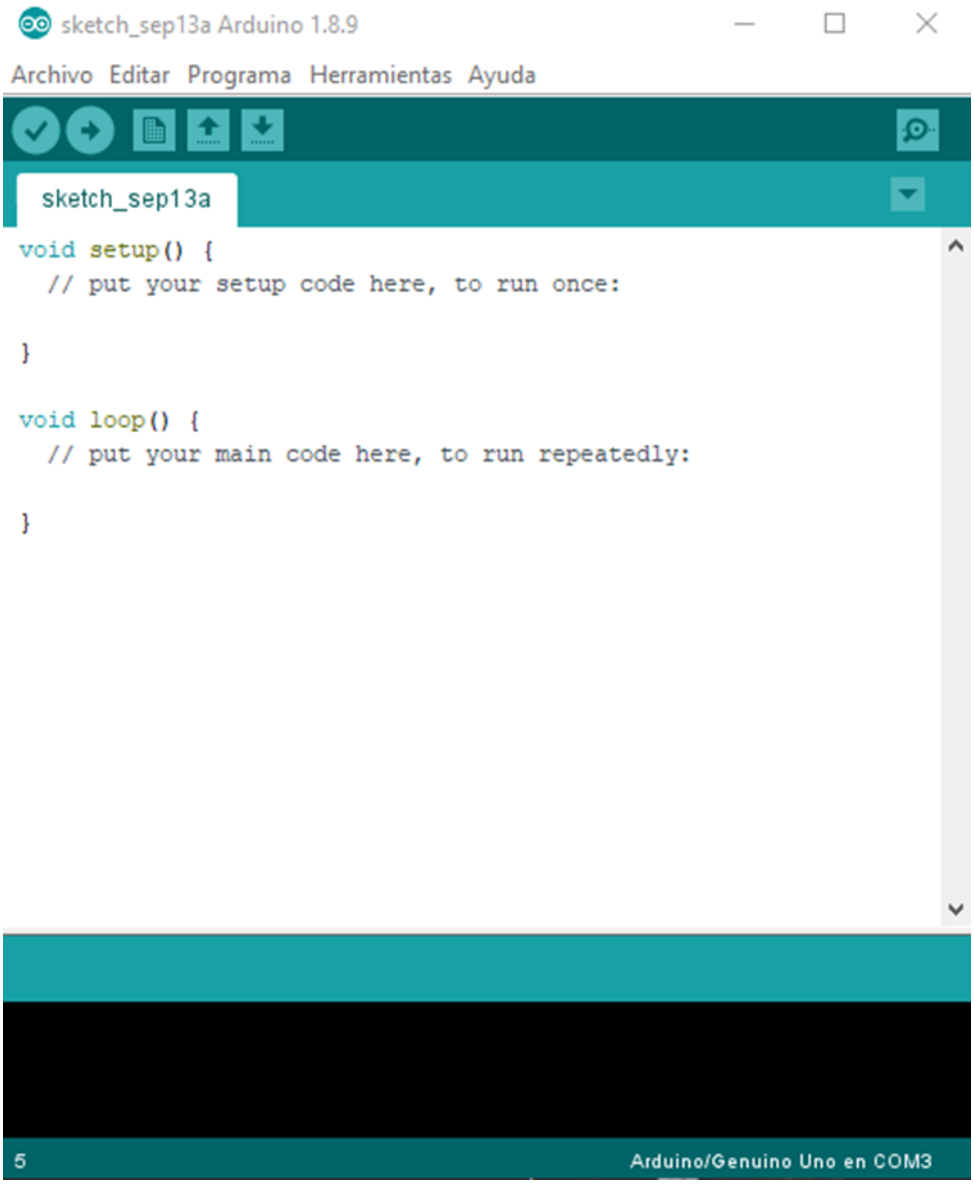


Ilustración 6. Interfaz de programación de Arduino

Es destacable desde la aparición de la versión 1.6.2. la incorporación de la gestión de librerías y la gestión de placas muy mejoradas respecto a la versión anterior y los avisos de actualización de versiones de librerías y cores.

1.6 PLATAFORMA 3DEXPERIENCE Y ARDUINO

Un proyecto de DASSAULT SYSTEMS Education Lab que cuenta con:

- Laboratorio virtual
- Control de un brazo robótico de 6 grados de libertad
- Uso de la plataforma de Arduino
- Mecatrónica y escenarios transdisciplinarios

- Potencial sin fin de desarrollo de aplicaciones

Conexión ciberfísica a través de CATIA SYSTEMS FMI:

- Conexión entre CATIA Systems y Arduino
- Control mutuo y realimentación
- Operaciones simultáneas de Hard & Software
- Software/Hardware/Model-in-the-loop
- FMI estándar y reutilizable en cualquier sistema impulsado por Arduino

1.7 INGENIERÍA DE SISTEMAS CON CATIA SYSTEMS

Esta figura representa la Ingeniería de Sistemas y la metodología RFLP. Explica la importancia de los modelos Requirement, Function, Logical Design y Phisical en la metodología RFLP.

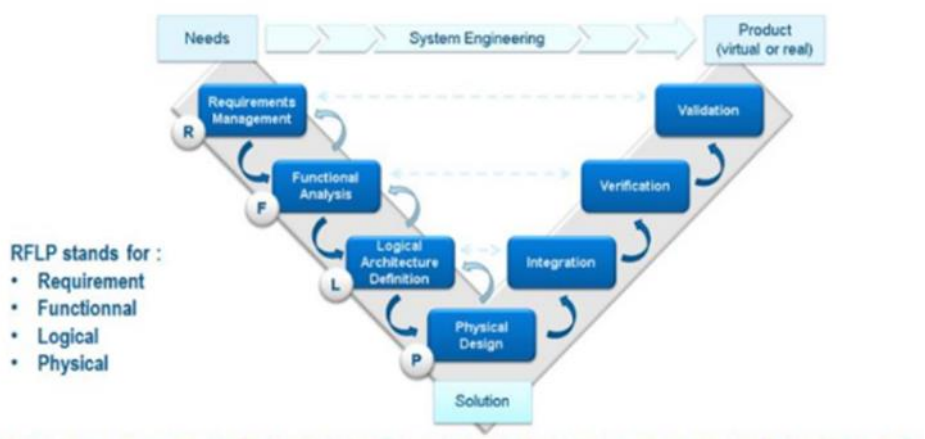


Ilustración 7. Metodología RFLP

Siguiendo esta representación se conseguirá:

- Utilizar el enfoque de la Ingeniería de Sistemas para gestionar los procesos de ingeniería multidisciplinarios concurrentes.
- Aplicar el requisito (Requirement), función (Function) y enfoque lógico (Logical approach) para optimizar el proceso de diseño.

Vamos a utilizar el enfoque de la Ingeniería de Sistemas gracias al proceso de “Gemelo Virtual”.

1.8 DEFINICIÓN DE RFLP

La RFLP es una fase de la Ingeniería de Procesos basada en el desarrollo de diseño del ciclo virtual. Los principios generales del enfoque de proceso son:

Ilustración 8. Ciclo RFLP

El proceso que sigue la RFLP es este:

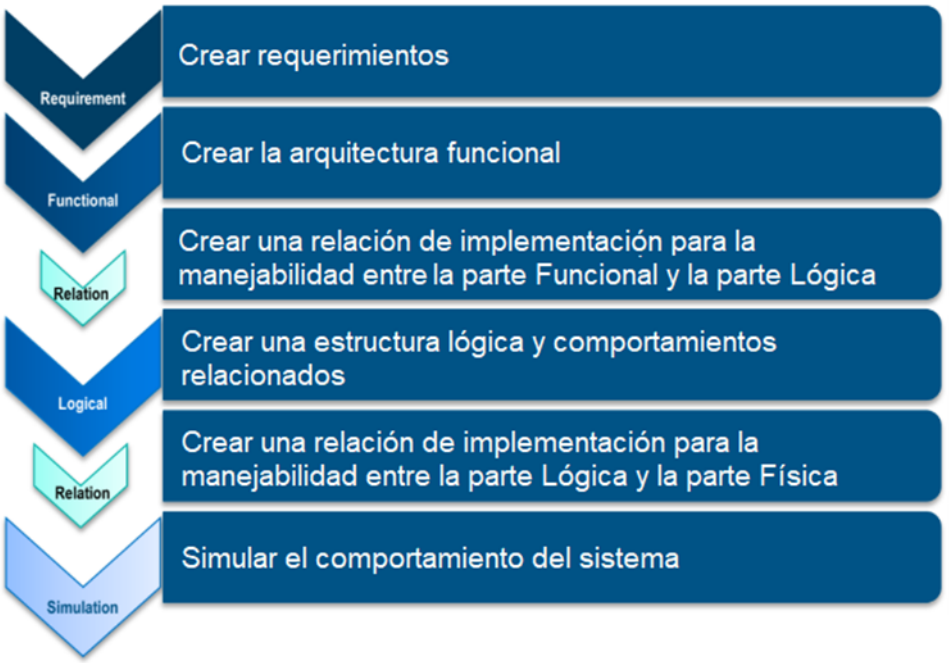


Ilustración 9. Proceso RFLP

1.9 GESTIÓN DE PROYECTOS

La Gestión de Proyectos consiste en varios pasos:

- Expresión de una necesidad: primero, el enfoque de un proyecto supone la identificación de una necesidad.
- Declaración del alcance: expresa la demanda del consumidor al proveedor. Este documento debe ser preciso para asegurar que los productos están hechos de acuerdo con las expectativas.

- Especificación: en este paso, la necesidad del consumidor es analizada por el proveedor. Ayuda a reducir la complejidad y definir los límites.
- Concepción (diseño): consiste en inventar, crear y desarrollar un producto usando el conocimiento, técnicas, herramientas y métodos tanto de Ciencias como de Técnicas de la Industria y desarrollo sostenible.
- Producción: manufacturación de los elementos que constituyen el producto.

1.10 MODELO VIRTUAL Y MODELO REAL

Cada modelo virtual se une en un enfoque iterativo. Algunas iteraciones son necesarias para completar los distintos pasos de un proyecto. El enfoque no es secuencial sino simultáneo.

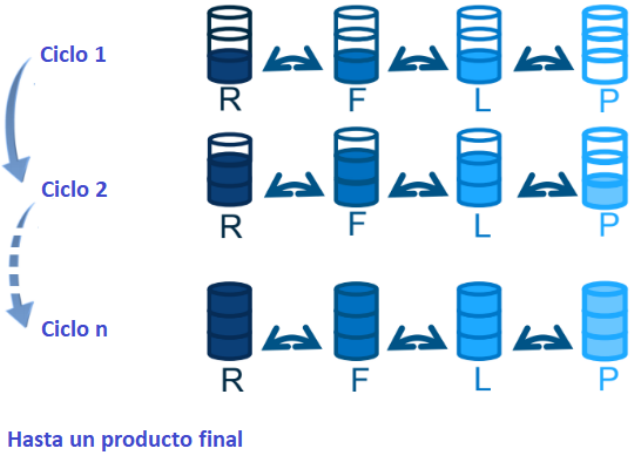


Ilustración 10. Enfoque iterativo RFLP

La versión real del robot es capaz de validar las soluciones elegidas para encontrar las mejores especificaciones.

2 INTERNET OF THINGS

2.1 DEFINICIÓN DE INTERNET OF THINGS (IOT)

Es la red de objetos físicos o "cosas" integrados con electrónica, software, sensores y conectividad para permitir a los objetos intercambiar datos con la producción, el operador y/o otros dispositivos conectados en la infraestructura de la Iniciativa de Estándares Globales de la Unión Internacional de Telecomunicaciones.

Permite a los objetos ser detectados y controlados a distancia a través de la infraestructura de red existente, creando oportunidad a una integración más directa entre el mundo físico y los sistemas

basados en computador, dando como resultado una mejor eficiencia, precisión y beneficio económico.

Una definición más sencilla describe el concepto “Internet of Things” como la interconexión digital de objetos cotidianos con Internet. Alternativamente, “Internet de las cosas” es la conexión de Internet con más “cosas u objetos” que personas.

El concepto de “Internet of Things” fue propuesto por Kevin Ashton en el Auto-ID Center del MIT en 1999, donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red y tecnologías de sensores.

El “Internet de las cosas” debería codificar de 50 a 100.000 millones de objetos y seguir el movimiento de estos. Se calcula que todo ser humano está rodeado, al menos, de entre 1000 a 5000 objetos. Según la empresa Gartner, en 2020 habrá en el mundo aproximadamente 26 mil millones de dispositivos con un sistema de conexión al “Internet de las cosas”. Abi Research, por otro lado, asegura que para el mismo año existirían 30 mil millones de dispositivos inalámbricos conectados a Internet. Con la próxima generación de aplicaciones de Internet (protocolo IPv6) se podrían identificar todos los objetos, algo que no se podía hacer con IPv4. Este sistema sería capaz de identificar instantáneamente por medio de un código a cualquier tipo de objeto.

2.2 INTEGRACIÓN ENTRE EL MUNDO FÍSICO Y LOS SISTEMAS BASADOS EN COMPUTADOR

- Técnicas de producción personalizadas, ingeniería distribuida y fabricación, instalaciones de producción inteligentes, grupos interesados dispersados a nivel mundial…
- IoT necesita un entorno digital que provea una representación con alta fidelidad de objetos y sus sistemas de producción. Permanentemente conectados con sus avatares físicos, cuyas representaciones anticipan y conducen el comportamiento de las “cosas”, pero que también asimile continuamente información proporcionada por los dispositivos reales para representar su estado real en operación: el Gemelo Virtual (Virtual Twin).
- Debido a que es virtual y reproduce la realidad lo más fielmente posible, el gemelo virtual proporciona un contexto asequible y realista para los educadores que buscan hacer de las prácticas de la nueva economía una parte integral de la experiencia de aprendizaje de sus estudiantes.

3 CONTROL DE UN SERVOMOTOR

3.1 QUÉ ES Y CÓMO FUNCIONA UN SERVOMOTOR

Un servomotor es un dispositivo similar a un motor de corriente continua que tiene la capacidad de ubicarse en cualquier posición dentro de su rango de operación, y mantenerse estable en dicha posición. Lleva incorporado un sistema de regulación que puede ser controlado tanto en velocidad como en posición.

Ilustración 11. Servomotor

Está conformado por un motor y un circuito de control. También potencia proporcional para cargas mecánicas. Un servo, por consiguiente, tiene un consumo de energía reducido.

Resumiendo, un servomotor es un motor especial al que se ha añadido un sistema de control (tarjeta electrónica), un potenciómetro y un conjunto de engranajes. Hoy en día existen servomotores en los que puede ser controlada su posición y velocidad en los 360 grados.

Los servomotores hacen uso de la modulación por ancho de pulsos (PWM) para controlar la dirección o posición de los motores de corriente continua. La mayoría trabaja en la frecuencia de los 50 Hz, así las señales PWM tendrán un periodo de 20 ms. La electrónica dentro del servomotor responderá al ancho de la señal modulada de la siguiente manera:

- Si los circuitos dentro del servomotor reciben una señal de entre 1 a 1,4 ms, éste se moverá en sentido horario.
- Si los circuitos dentro del servomotor reciben una señal de entre 1,6 a 2 ms moverá el servomotor en sentido antihorario.
- Si los circuitos dentro del servomotor reciben una señal de 1,5 ms representa un estado neutro para los servomotores estándares.

Las especificaciones de los servomotores elegidos son las siguientes:

- Mantiene cualquier posición entre 0 y 180 grados
- Acepta cuatro tornillos de montaje
- El engranaje de alta precisión hecho de resina POM (poliacetal) permite un funcionamiento suave sin holgura
- Pesa solo 44 gramos
- Par motor de 1 kg a 6 V

3.2 ARDUINO

3.2.1 Conexión del servomotor a Arduino

El servomotor cuenta con tres cables cuyos colores corresponden a funciones específicas:

- El cable negro está conectado a la masa del servomotor.
- El cable rojo está conectado a la fuente de alimentación del servomotor.
- El cable blanco es la entrada del servomotor y está conectado a las posiciones del motor.

Los cables del servomotor se deben conectar a la tarjeta Arduino de la siguiente manera:

- El cable negro (masa del servo) en uno de los puertos GND.

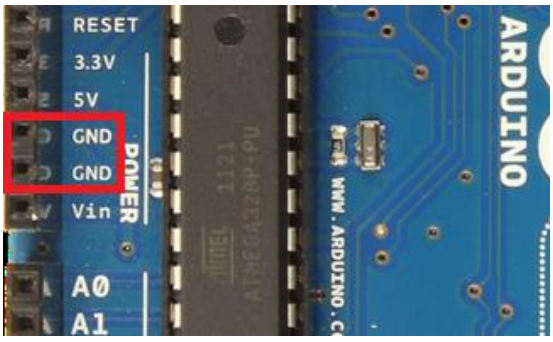


Ilustración 12. Puertos GND de Arduino

- El cable rojo (fuente de alimentación del servo) en el puerto Vin.

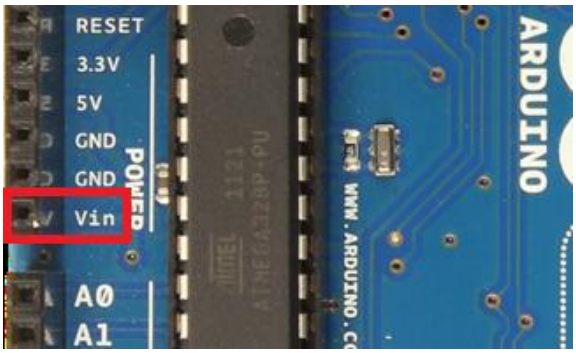


Ilustración 13. Puerto Vin de Arduino

- El cable blanco (posición del servo) en el puerto digital que se haya elegido, en este caso, el pin 2.

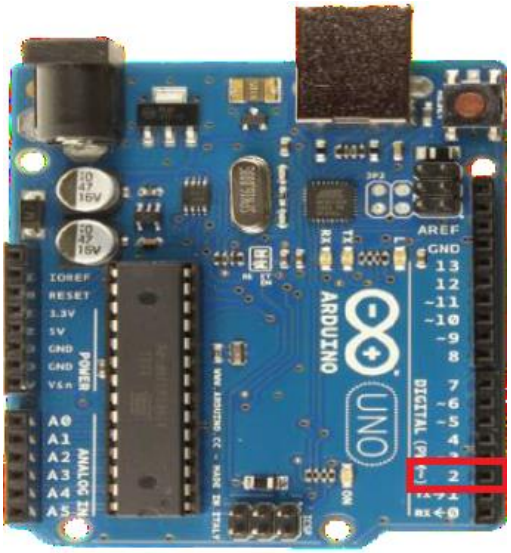


Ilustración 14. Pin 2 de Arduino

3.2.2. Lanzamiento del software de Arduino

Se debe conectar el cable USB de Arduino al PC y detectar el "COM Number" desde el Administrador de Dispositivos de Windows.

Ilustración 15. Icono de Arduno y su COM Number

Dentro del programa de Arduino, hay que asegurarse de que la correcta placa (Arduino Uno) y el correcto "COM Number" están seleccionados. Tras esto, ya se puede lanzar el programa a la placa de Arduino.

3.2.3. Programas que utiliza Arduino

En primer lugar, se necesitan dos librerías: *SimpleSoftwareServo* y *MsTimer2*, que serán añadidas a las que ya ofrece Arduino por defecto, ya que serán llamadas desde la función principal para ejecutar acciones recurrentes y no repetir el mismo código dentro del programa principal.

SimpleSoftwareServo

Su función es conducir servomotores en cualquier número de pines al mismo tiempo.

```
#ifndef SimpleSoftwareServo_h
#define SimpleSoftwareServo_h

#include "Arduino.h"

#include <stdint.h>

class SimpleSoftwareServo
{
  private:
    uint8_t pin;
    uint8_t angle;
    uint16_t pulse;
    uint16_t min_pulse;
    uint16_t max_pulse;
    class SimpleSoftwareServo *next;
    static SimpleSoftwareServo* first;
  public:
    SimpleSoftwareServo();
    uint8_t attach(uint8_t pin);
    uint8_t attach(uint8_t pin, uint16_t newMin, uint16_t newMax);
    void detach();
    void write(uint8_t);
    uint8_t read();
    uint8_t attached();
    void setMinimumPulse(uint16_t);
    void setMaximumPulse(uint16_t);
    static void refresh();
    static const uint16_t refresh_interval = 20000;
};

#endif
```

Mediante este código, se establecen un ángulo (de 0 a 180 grados), un pulso (en microsegundos) y un pulso mínimo y máximo.

Conectará con un pin, establecerá el modo de ese pin, junto con el mínimo y el máximo para el ángulo del servo, y no posicionará el servomotor hasta que aparezca el comando *write()*, el cual es el que especifica el ángulo (en grados y de 0 a 180, como se mencionó anteriormente) .

Se debe actualizar con el comando *refresh()* al menos cada 50 ms aproximadamente para mantener activo el servomotor; cada vez que lo llamas, se genera un pulso y pasarán hasta 20 ms.

Un ejemplo sencillo del uso de este programa es el siguiente, en el cual se hace que un servomotor realice barridos de ida y vuelta:

```
#include <SimpleSoftwareServo.h>

SimpleSoftwareServo servo1;

int servo1_pin = 2;

void setup() {
  servo1.attach(servo1_pin);
}

void loop() {
  int pos;
  for(pos = 0; pos < 180; pos += 1)
  {
    servo1.write(pos);
    SimpleSoftwareServo::refresh();
  for(pos = 180; pos>=1; pos -= 1)
  {
    servo1.write(pos);
    SimpleSoftwareServo::refresh();
  }
}

#end
```

Se puede observar cómo inicialmente se crea la variable *servo1* y se selecciona el pin 2. Tras esto, el servomotor irá de 0 a 180 grados en intervalos de 1 grado, seguido de un delay de 20 ms, y después irá de 180 a 0 grados, seguido de otra pausa de 20 ms.

MsTimer2

Permite llamar a una función periódicamente.

```
#ifndef MsTimer2_h
#define MsTimer2_h

#include <avr/interrupt.h>

namespace MsTimer2 {
    extern unsigned long msecs;
    extern void (*func)();
    extern volatile unsigned long count;
    extern volatile char overflowing;
    extern volatile unsigned int tcnt2;

    void set(unsigned long ms, void (*f)());
    void start();
    void stop();
    void _overflow();
}

#endif
```

Consta de tres partes:

- *MsTimer2::set* – especifica una función de devolución de llamada ejecutada en el controlador de interrupción de temporizador de intervalo especificado.
- *MsTimer2::start* – inicia el temporizador después del "set".
- *MsTimer2::stop* – detiene el temporizador.

Este ejemplo muestra su uso en el caso de que se necesitara encender un LED en el pin 13 cada segundo:

```
#include <MsTimer2.h>

void flash() {
  static boolean output = HIGH;

  digitalWrite(13, output);
  output = !output;
}

void setup() {
  pinMode(13, OUTPUT);

  MsTimer2::set(500, flash);
  MsTimer2::start();
}

void loop() {
}
```

Una vez definidas estas dos funciones, se puede dar paso al programa principal.

Programa principal

Primero, en la parte de declaración de variables, se establece un tamaño para el "buffer" de 95, que es un buen límite para la operación del programa. También se especifica que el número de pines es 20 y que 14 de ellos son digitales (los seis restantes son los pines analógicos de A0 a A5).

```
#include <SimpleSoftwareServo.h>
#include <MsTimer2.h>

int const limite = 95;
char buffer[limite];
int nbDonnee = 0;
boolean informationClosed = true;

int const nbPin = 20;
int const nbDigitalPin = 14;
```

A continuación, se fija que cada pin tendrá 3 atributos. El primero es el tipo y los valores podrán ser: -1 = no usar este pin, 0 = digital, 1 = PWM, 2 = analógico); el segundo atributo es el modo que puede ser: 0 = Salida, 1 = Entrada, 2 = Servo; y el tercer atributo es el último valor de la salida.

Los puertos PWM (modulación de ancho de pulso) se emplean en una amplia variedad de aplicaciones, que van desde la medición y las comunidades hasta el control de potencia y la conversión.

Después se define el tipo de los pines de Arduino de la siguiente manera (cada valor del tipo es asignado según lo establecido en el párrafo anterior):

1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
-1	-1	0	1	0	1	1	0	0	1	1	1	0	0	2	2	2	2	2	2

En la variable “pin” se crea una especie de tabla de estado de cada pin. Esta tabla de estado tiene 20 filas (número de pines) y 3 columnas (número de atributos de cada pin).

Con la función *SimpleSoftwareServo*, se crea un “objeto servo” en todos los pines para controlar un servomotor si es necesario y se inician el pin y el valor actual a cero. En este caso, sólo nos interesa el pin 2 para controlar el único servomotor disponible.

```
int const nbAttribute = 3;
int const typ = 0; int const mod = 1; int const val = 2;
int defPinType[nbPin] = {-1, -1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 2, 2, 2, 2, 2, 2};
int pin[nbPin][nbAttribute];
SimpleSoftwareServo servo[nbPin];

int currentPin = 0;
int value = 0;
int lect = 0;
```

En la parte del “setup”, se inicia el monitor serie y se establece la velocidad de datos en bits por segundo (baudios) para la transmisión de datos en serie. En este caso la velocidad es 9600.

Después, se configuran todos los pines como salidas y se ponen todas las posiciones de “pin” (tabla de estado) a cero. Se utiliza la función *MsTimer2* para fijar e iniciar un periodo de 20 ms.

```
void setup()
{

  Serial.begin(9600);

  for(int i = 0; i < nbPin; i++)
  {
    pin[i][typ] = defPinType[i];
    pinMode(i, OUTPUT);
  }

  for(int i = 0; i < nbPin; i++)
  {
      for(int j = 1; j < nbAttribute; j++)
        {
          pin[i][j] = 0;
        }
  }

  MsTimer2::set(20, refresh);
  MsTimer2::start();
}


void refresh()
{
  SimpleSoftwareServo::refresh();
}
```

En la parte del bucle (loop), se define que cuando en el “buffer” se detecte una orden, se ejecute la acción pertinente y, si no se detecta nada, se pasará a la siguiente línea de órdenes.

```
void loop()
{

  if(Serial.available() > 0)
  {
    lecture(buffer, limite, &nbDonnee);
  }

  if(completeInformation(buffer))
  {
    if(buffer[0] == 'p')
    {
        eraseFirstChar(buffer);
        if(isAValue(buffer))
        {
            value = atoi(buffer);
            modifyPinSelected(value);
        }
    }
```

En el corte de código de la imagen superior, se realiza una lectura del "buffer" y, en el primer condicional, si se recibe la orden 'p', significa que se quiere modificar el pin seleccionado, por lo que se elimina esa orden mediante *eraseFirstChar* y se guarda en la variable "value" (la función *atoi()* permite convertir una cadenas de caracteres (char) a una variable entera (int)) el nuevo pin seleccionado, para seguidamente introducirlo en la función *modifyPinSelected()*.

El resto del "loop" son otros casos de órdenes y acciones recibidas en el "buffer". En la siguiente imagen se observa un caso como el anterior, pero esta vez para modificar el modo del pin.

```
else if(buffer[0] == 'm')
{
    eraseFirstChar(buffer);
    if(isAValue(buffer))
    {
      value = atoi(buffer);
      modifyPinMode(currentPin, value);
    }
}
```

En la siguiente imagen se observa un caso como el anterior, pero esta vez para modificar el valor del pin.

```
else if(buffer[0] == 'v')
{
    eraseFirstChar(buffer);
    if(isAValue(buffer))
    {
      value = atoi(buffer);
      modifyPinValue(currentPin, value);
    }
}
```

Para terminar el "loop", los últimos condicionantes introducen los casos de que no se recibiera ninguna orden, por lo que se pasaría a la siguiente línea,

```
else if(buffer[0] == ' ')
{
    eraseFirstChar(buffer);
    Serial.println("");
}
```

O de que se recibiera una orden no convencional, por lo que se borraría el primer valor del "buffer",

```
else if(nbDonnee > 0)
{
    eraseFirstChar(buffer);
}

informationClosed = false;
```

O de que no hubiese ninguna orden más.

```
  else
      {
        if(!informationClosed)
        {
          Serial.print("c");
          informationClosed = true;
        }
      }
}
```

La parte final del programa corresponde a la creación de las funciones utilizadas dentro del 'loop', que tienes las tareas de cambiar el pin seleccionado (*modifyPinSelected*), cambiar el modo del pin (*modifyPinMode*), cambiar el valor del pin (*modifyPinValue*) y otras funciones necesarias incluidas dentro de las anteriores (*applyPinValue*, *readPinValue*, …).

modifyPinSelected

Cambia el pin seleccionado: sustituye en la variable "value" su valor por el nuevo.

```
void modifyPinSelected(int value)
{
  if(value >= 0 && value < nbPin)
  {
      currentPin = value;
      Serial.print("p"); Serial.print(value);
  }
}
```

modifyPinMode

Cambia el modo del pin seleccionado: en esta función existen 3 casos: cambio a salida, cambio a entrada y cambio a servo.

El primer caso es si se quiere cambiar el modo del pin a salida, donde separa la "variable servo" de ese pin. Después se cambia el atributo del modo a salida y se le aplica al pin el último valor enviado.

```
void modifyPinMode(int currentPin, int value)
{
  if(pin[currentPin][typ] == 0 || pin[currentPin][typ] == 1 || pin[currentPin][typ] == 2)
  {
    if(value == 0)
    {
        if(servo[currentPin].attached() == 1)
        {
          servo[currentPin].detach();
        }
        pinMode(currentPin, OUTPUT);
        pin[currentPin][mod] = value;
        Serial.print("m"); Serial.print(value);
        applyPinValue(currentPin, pin[currentPin][val]);
}
```

El segundo caso es si se quiere cambiar el modo del pin a entrada, donde separa la “variable servo” de ese pin. Después se cambia el atributo del modo a entrada y se lee el valor del pin actual.

```
    else if(value == 1)
    {
        if(servo[currentPin].attached() == 1)
        {
          servo[currentPin].detach();
        }
        pinMode(currentPin, INPUT);
        pin[currentPin][mod] = value;
        Serial.print("m"); Serial.print(value);
        Serial.print("v"); Serial.print(readPinValue(currentPin));
    }
```

El último caso es si se quiere cambiar el modo del pin a servo, donde se une el “objeto servo” a ese pin. Después se cambia el atributo del modo a servo y se le aplica el último valor enviado.

```
    else if(value == 2)
    {
        Serial.print("m"); Serial.print(value);
        if(servo[currentPin].attached() != 1)
        {
          servo[currentPin].attach(currentPin);
        }
        pin[currentPin][mod] = value;
        applyPinValue(currentPin, servo[currentPin].read());

    }
  }
}
```

<u>modifyPinValue</u>

Cambia el valor del pin seleccionado: siempre que el pin del que se trata sea una entrada, se cambia el modo del pin de entrada a salida y se le aplica el valor actual.

```
void modifyPinValue(int currentPin, int value)
{
  if(pin[currentPin][typ] == 0 || pin[currentPin][typ] == 1 || pin[currentPin][typ] == 2)
  {
      if(pin[currentPin][mod] == 1)
      {
        Serial.print("m0");
        pin[currentPin][mod] = 0;
        pinMode(currentPin, OUTPUT);
      }
    applyPinValue(currentPin, value);
  }
}
```

applyPinValue

Aplica el valor obtenido al pin seleccionado: puede que el pin sea un servo o sea una entrada y también puede haber tres casos: que el valor sea el mínimo, el máximo o un valor intermedio entre ellos.

· Si el pin es una entrada, sus valores varían entre 0 y 255.

```
void applyPinValue(int currentPin, int value)
{
  if(pin[currentPin][mod] == 0)
  {
      if(value == 0)
      {
        digitalWrite(currentPin, LOW);
        pin[currentPin][val] = value;
        Serial.print("v"); Serial.print(value);
      }
      else if(value == 255)
      {
        digitalWrite(currentPin, HIGH);
        pin[currentPin][val] = value;
        Serial.print("v"); Serial.print(value);
      }
      else if(pin[currentPin][typ] == 1 && value > 0 && value < 255)
      {
        analogWrite(currentPin, value);
        pin[currentPin][val] = value;
        Serial.print("v"); Serial.print(value);
      }
```

· Si el pin es un servo, sus valores varían entre 0 y 180 grados.

```
  else if(pin[currentPin][mod] == 2)
  {
      if(value >= 0 && value <= 180)
      {
        Serial.print("v"); Serial.print(value);
        if(servo[currentPin].attached() != 1)
        {
          servo[currentPin].attach(currentPin);
        }
      servo[currentPin].write(value);
    }
  }
}
```

readPinValue

Lee el valor del pin seleccionado. En el primer condicional se da el caso de que el pin sea una entrada y el que caso de que sea un pin digital o PWM, o sea un pin analógico:

```
int readPinValue(int currentPin)
{
  if(pin[currentPin][mod] == 1)
  {
      if(pin[currentPin][typ] == 0 || pin[currentPin][typ] == 1)
      {    // Pin digital o PWM
      return digitalRead(currentPin)*1023;
      }
      else if(pin[currentPin][typ] == 2)
      {    // Pin analógico
        return analogRead(currentPin);
      }
    }
```

En el segundo condicional se da el caso de que el pin sea un servo, donde se devuelve el último valor enviado:

```
  else if(pin[currentPin][mod] == 2)
  {
      if(servo[currentPin].attached() == 1)
      {
        return servo[currentPin].read();
      }
  }
  return -1;
}
```

eraseFirstChar

Borra la orden anterior que haya en el “buffer”.

```
void eraseFirstChar(char * buffer)
{
    for(int i = 0; i < nbDonnee-1; i++)
    {
      buffer[i] = buffer[i+1];
    }
    buffer[nbDonnee-1] = '\0';
    nbDonnee = nbDonnee-1;
}
```

isAValue

Detecta si hay algún valor en el “buffer”.

```
boolean isAValue(char * buffer)
{
    boolean isValue = false;
    if(atoi(buffer) != 0 || buffer[0] == '0')
    {
      isValue = true;
    }
    return isValue;
}
```

lecture

Lee el “buffer” y detecta si el tamaño de la orden es menor al tamaño máximo establecido.

```
void lecture(char * buffer, int maxSize, int *lenght)
{
    while(Serial.available() > 0 && *lenght < maxSize-1)
    {
      buffer[*lenght] = Serial.read();
      (*lenght)++;
    }
  buffer[*lenght] = '\0';
}
```

3.3 3DEXPERIENCE

El software de 3DEXPERIENCE necesita ciertos compiladores, en este caso, un compilador de lenguaje C, para el cual es válida la última versión de Microsoft Visual Studio.

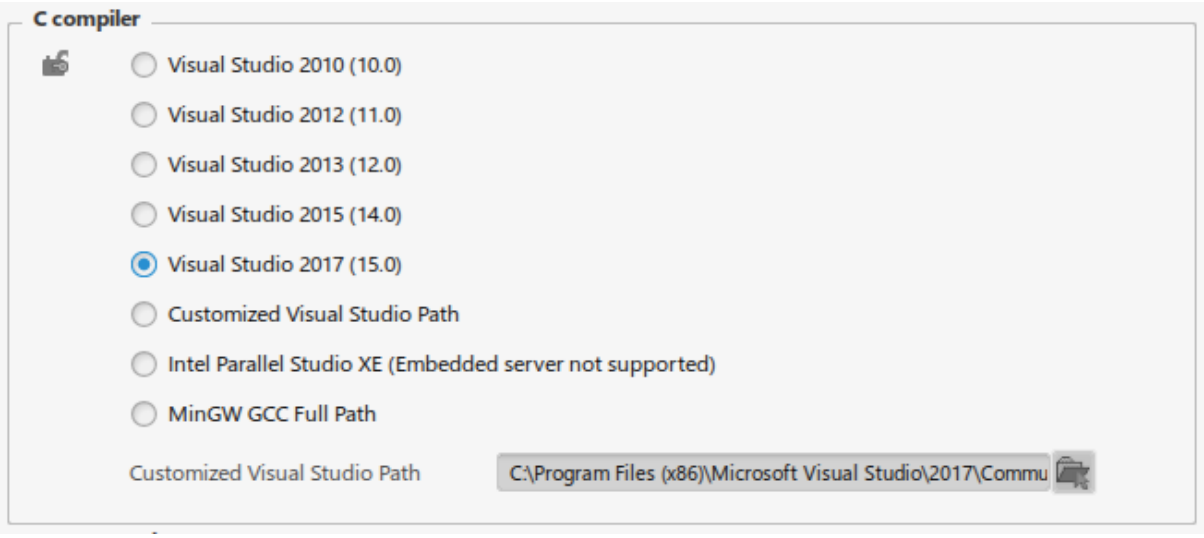


Ilustración 16. Selección del compilador de C

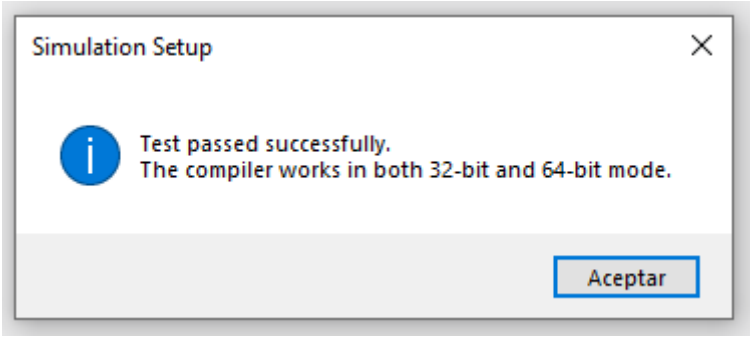


Ilustración 17. Notificación del correcto funcionamiento del compilador

3.3.1 MODELICA y CATIA

MODELICA (Modelica 3.2.2.) – Modelica es un lenguaje de modelización orientado a objetos para el modelado de componentes de sistemas complejos, por ejemplo, sistemas que contienen subcomponentes mecánicos, eléctricos, electrónicos, hidráulicos, de control, de energía eléctrica u orientados a procesos. Las características de Modelica (acausal, orientada a objetos, dominio neutral) lo hacen muy adecuado para la simulación a nivel de sistema, un dominio donde Modelica ahora está bien establecido.

La Asociación Modelica también desarrolla la biblioteca estándar de Modelica que contiene aproximadamente 1360 componentes genéricos del modelo y 1280 funciones en varios dominios, a partir de la versión 3.2.1.

CATIA (CATIA 1.3.1.) – CATIA es un paquete de software multiplataforma para diseño asistido por computadora (CAD), fabricación asistida por computadora (CAM), ingeniería asistida por computadora (CAE), PLM y 3D, desarrollado por la compañía francesa Dassault Systèmes.

CATIA usa el lenguaje abierto Modelica tanto en CATIA Dynamic Behavior Modeling como en Dymola, para modelar y simular rápida y fácilmente el comportamiento de sistemas complejos que abarcan múltiples disciplinas de ingeniería. CATIA y Dymola se extienden aún más a través de la disponibilidad de una serie de bibliotecas de Modelica específicas de la industria y el dominio que permiten al usuario modelar y simular una amplia gama de sistemas complejos, desde la dinámica de vehículos automotores hasta la dinámica de vuelo de aeronaves.

3.3.2 Modelo Virtual

El siguiente esquema muestra el modelo virtual para el control dinámico de un servomotor.

A la izquierda se encuentra una señal sinusoidal de entrada. Directamente conectada con la mencionada señal está el bloque que representa la tarjeta Arduino. A continuación, se representa un controlador PID (controlador Proporcional, Integral y Derivativo) y finalmente, se encuentra el servomotor para cerrar el sistema.

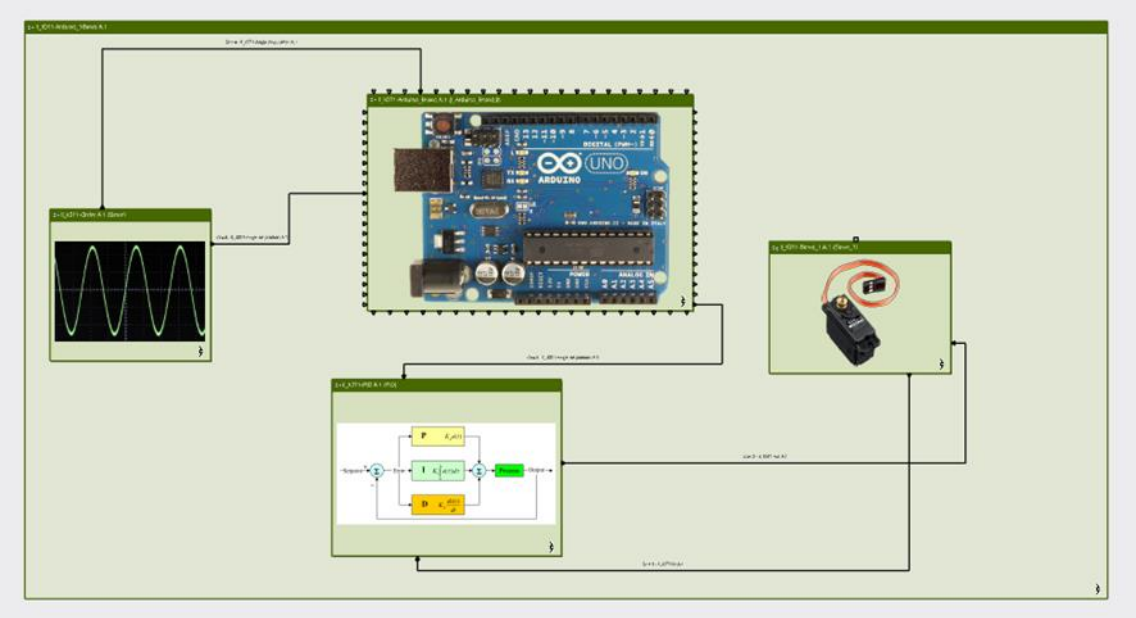


Ilustración 18. Modelo Virtual para el control de un servomotor

El primer bloque, que representa la señal sinusoidal de entrada, contiene este esquema:

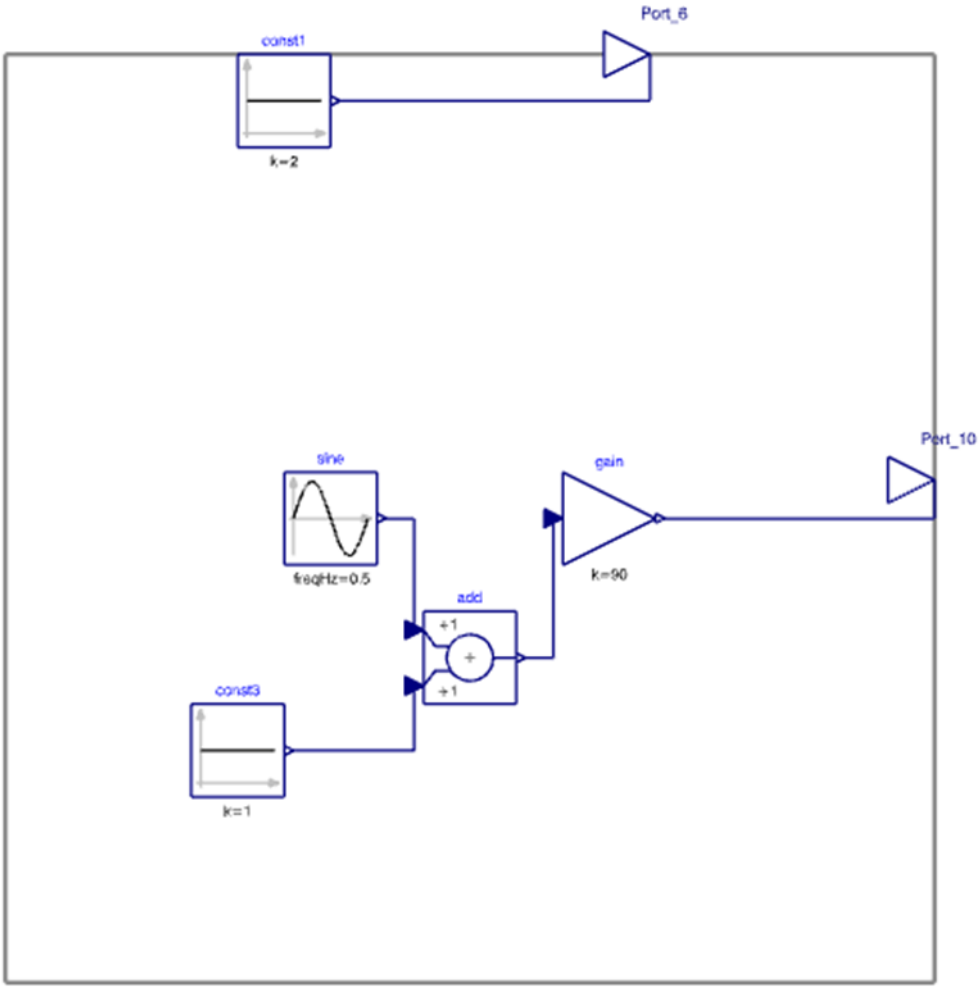


Ilustración 19. Bloque lógico de la señal de entrada

El bloque superior es una constante de valor 2, que indica a la tarjeta Arduino que el pin 2 es el pin de entrada que está conectado al servomotor.

Debajo, una señal sinusoidal seguida de una ganancia es la señal de entrada que alimentará el servomotor.

El bloque del controlador PID contiene este esquema:

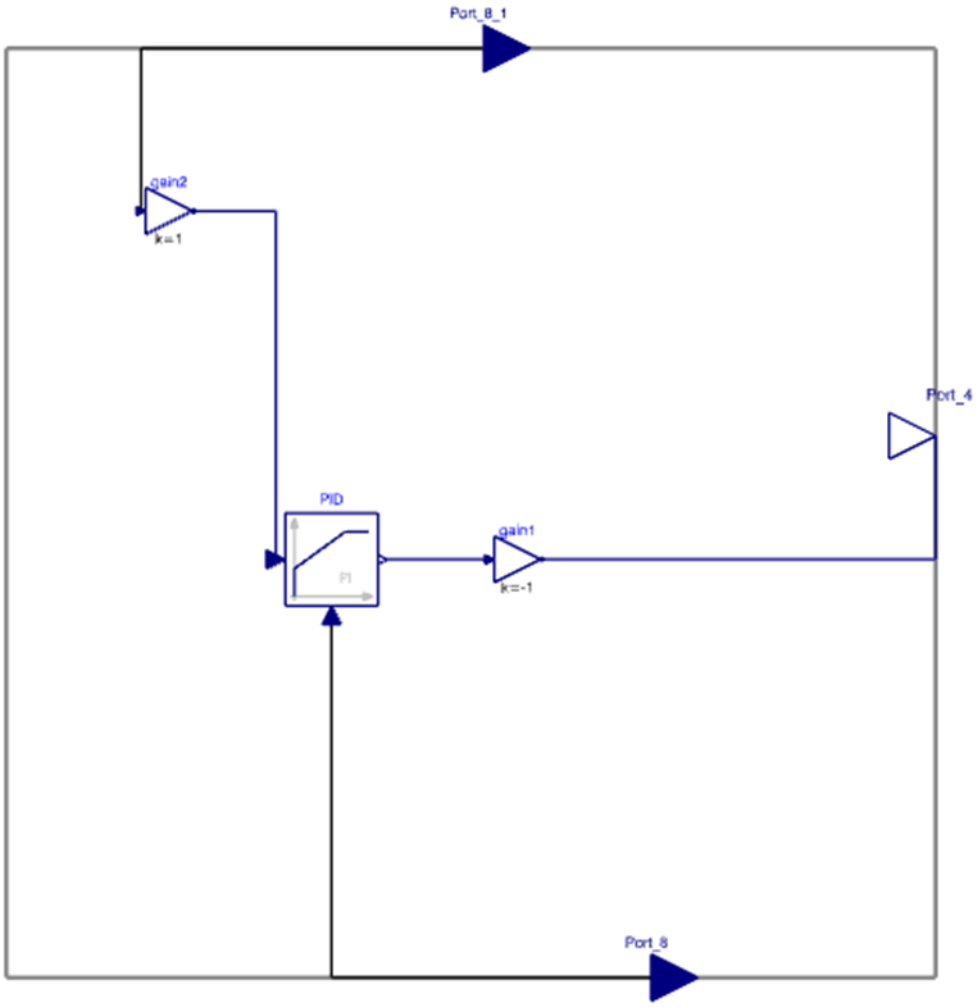


Ilustración 20. Bloque lógico del controlador PID

Antes de comentar cuáles son las entradas y la salida de este controlador, se debe explicar el funcionamiento de este operador.

3.3.3 PID

Un controlador PID (Proporcional, Integral y Derivativo) es un mecanismo de control simultáneo que calcula el error entre un valor medido y un valor deseado. Los tres parámetros que constituyen el algoritmo del control PID son:

- Proporcional: valor que depende del error actual.
- Integral: valor que depende de los errores pasados.
- Derivativo: predicción de los errores futuros.

La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control, como en este caso, la posición de un servomotor.

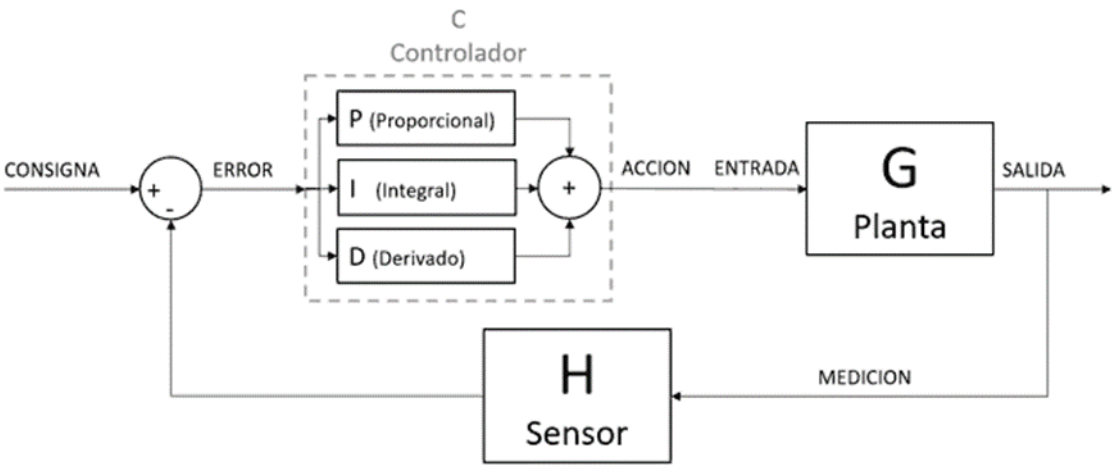


Ilustración 21. Diagrama de bloques del sistema

Para el correcto funcionamiento de un controlador PID que regule un sistema se necesitan, al menos:

- Un sensor, que determinará el estado del sistema. Este proporciona una señal analógica o digital al controlador, la cual representa el punto actual en el que se encuentra el sistema.
- Un controlador, que generará la señal que gobierna el actuador. El controlador recibe una señal externa que representa el valor que se desea alcanzar; esta señal recibe el nombre de punto de consigna, la cual es de la misma naturaleza y tiene el mismo rango de valores que la señal que proporciona el sensor.

  El controlador resta la señal de punto de consigna, obteniendo así la señal de error, que determina en cada instante la diferencia que hay entre el valor deseado y el valor medido.

  La señal de error es utilizada por cada uno de los tres componentes del controlador PID. Las tres señales sumadas, componen la señal de salida que el controlador va a utilizar para gobernar el actuador. La señal resultante de la suma de estas tres se llama variable manipulada y no se aplica directamente sobre el actuador, sino que debe ser transformada para ser compatible con el actuador utilizado.
- Un actuador, que modificará el sistema de manera controlada (el servomotor).

Proporcional

La parte proporcional consiste en el producto entre la señal de error y la constante proporcional para lograr que el error en estado estacionario se aproxime a cero, pero en la mayoría de los casos, estos valores solo serán óptimos en una determinada porción del rango total de control, siendo distintos valores óptimos para cada porción del rango.

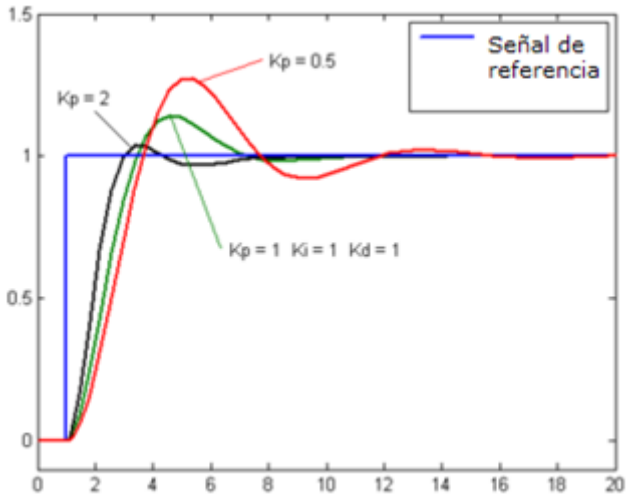


Ilustración 22. Gráfica de respuestas con diferentes valores de Kp

Sin embargo, existe también un valor límite en la constante proporcional (Kp) a partir del cual, en algunos casos, el sistema alcanza valores superiores a los deseados. Este fenómeno se llama sobreoscilación y, por razones de seguridad, no debe sobrepasar el 30%, aunque es conveniente que la parte proporcional ni siquiera produzca sobreoscilación. Hay una relación lineal continua entre el valor de la variable controlada y la posición del elemento final de control (el servomotor se mueve al mismo valor por unidad de desviación). La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto del tiempo, es incluyendo y configurando las acciones integral y derivativa.

Integral

El modo de control integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por perturbaciones exteriores y los cuales no pueden ser corregidos por el control proporcional.

El control integral actúa cuando hay una desviación entre la variable y el punto de consigna, integrando esta desviación en el tiempo y sumándola a la acción proporcional. El error es integrado, lo cual tiene la función de sumarlo por un periodo determinado.

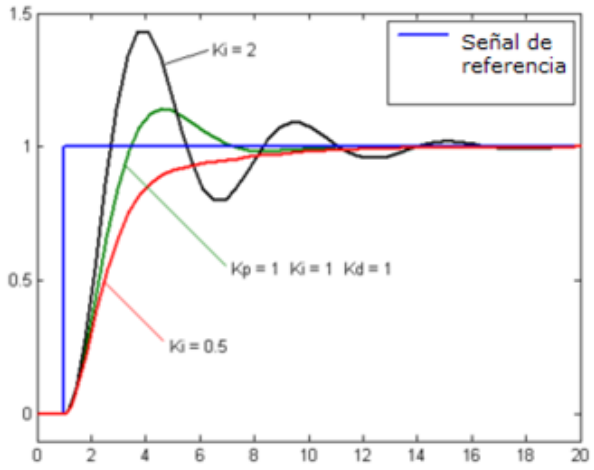


Ilustración 23. Gráfica de respuestas con diferentes valores de Ki

Luego es multiplicado por una constante (Ki). Posteriormente, la respuesta integral es adicionada al modo proporcional para formar el control P + I con el propósito de obtener una respuesta estable del sistema sin error estacionario. El modo integral presenta un desfase en la respuesta de 90º que sumados a los 180º de la realimentación, que es negativa, acercan al proceso a tener un retraso de 270º, luego entonces solo será necesario que el tiempo muerto contribuya con 90º de retardo para provocar la oscilación del proceso.

El control integral se utiliza para obviar el inconveniente del offset, que es la desviación permanente de la variable con respecto punto de consigna, de la banda proporcional.

Derivativo

La acción derivativa se manifiesta cuando hay un cambio en el valor absoluto del error. Si el error es constante, solamente actúan los modos proporcional e integral.

El error es la desviación existente entre el punto de medida y el valor consigna.

La función de la acción derivativa es mantener el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce, de esta manera evita que el error se incremente.

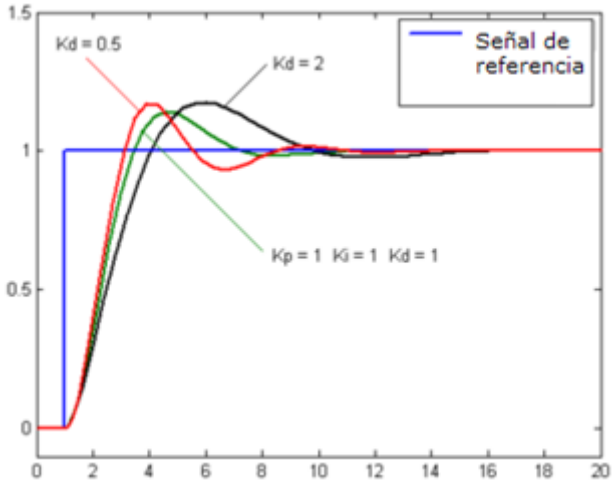


Ilustración 24. Gráfica de respuestas con diferentes valores de Kd

Se deriva con respecto al tiempo y se multiplica por una constante (Kd) y luego se suma a las señales anteriores (P + I). Es importante adaptar la respuesta de control a los cambios en el sistema ya que una mayor derivativa corresponde a un cambio más rápido y el controlador puede responder de la manera correcta.

El control derivativo se caracteriza por el tiempo de acción derivada en minutos de anticipo. La acción derivada es adecuada cuando hay retraso entre el movimiento de variable de control y su repercusión a la variable controlada.

El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones.

Conociendo el funcionamiento de este controlador, se puede observar que, en el esquema del bloque del controlador PID en 3DEXPERIENCE, la señal conocida como punto de consigna procede de la tarjeta Arduino y es la señal sinusoidal de entrada; la señal que se obtiene del PID (variable resultante) es la que se aplica al servomotor; y la señal de realimentación del PID es la que procede del sensor de posición del servomotor (la cual se comentará a continuación).

El bloque que representa el servomotor está compuesto por el siguiente sistema:

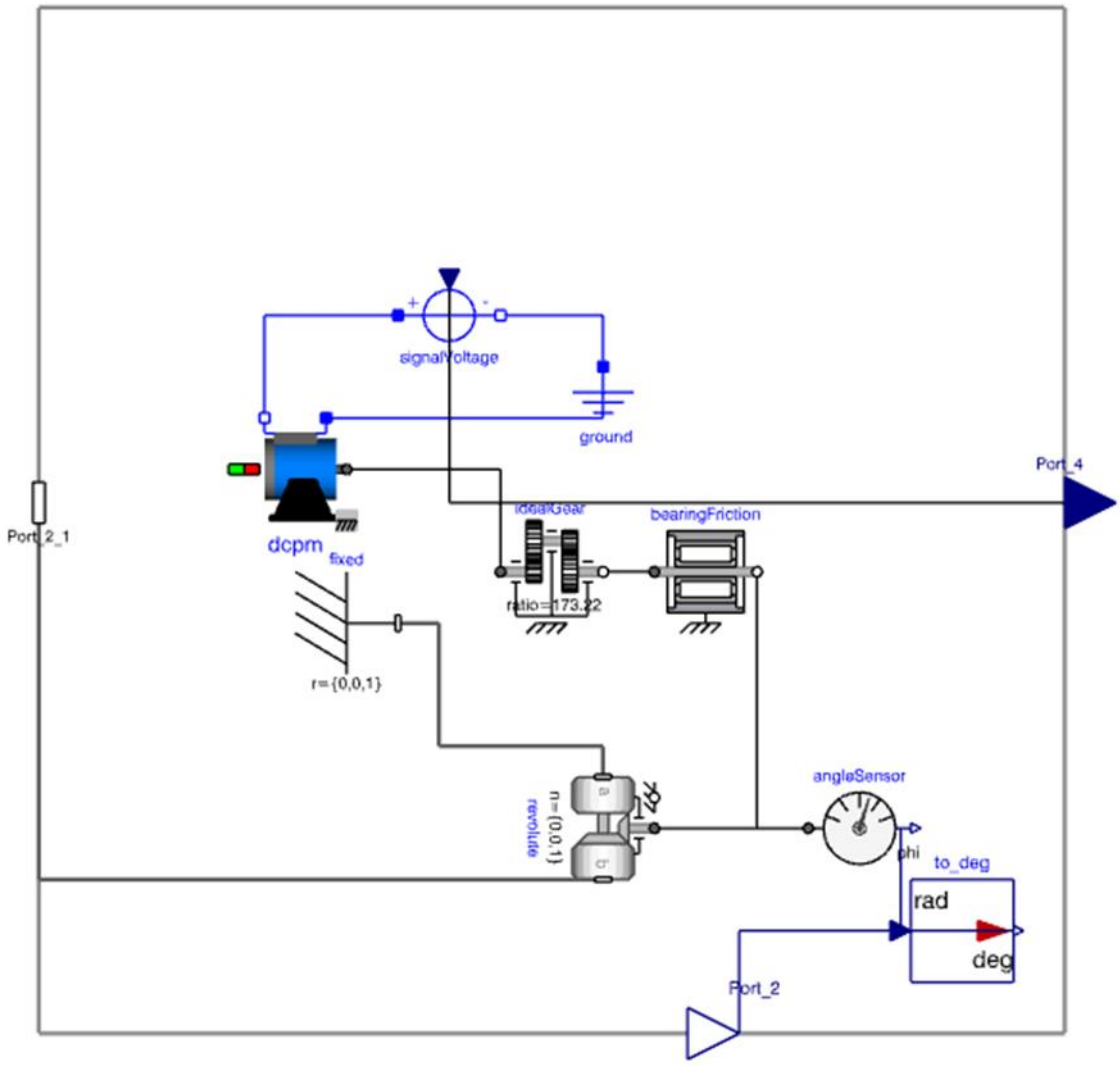


Ilustración 25. Bloque lógico del servomotor

El bloque recibe la señal conocida como variable aplicada, procedente del controlador PID. Esta señal es usada para alimentar al servomotor, que está representado por un motor de corriente continua, un engranaje ideal y un rodamiento de fricción.

Tras esto, hay un sensor el cual adquiere el ángulo de posición del servomotor. Este ángulo es transformado de radianes a grados. La señal resultante es la realimentación del controlador PID que, restada a la señal de entrada, forman la señal de error.

Este bucle se repetirá hasta que la señal de error sea nula, lo que implica que el servomotor se encuentra en la posición deseada.

3.3.4 Simulación

Al ejecutar la simulación, interesa prestar atención a las tres señales que intervienen en el controlador PID; son las siguientes:

- Señal azul claro: esta señal es la consigna, es decir, la señal sinusoidal de entrada.
- Señal azul oscuro: esta señal es la realimentación. Proviene del sensor del servomotor y representa su ángulo de posición en el instante actual.
- Señal roja: es la señal obtenida por el controlador PID, la cual es aplicada al servomotor para intentar obtener la posición requerida.

Esta serie de imágenes representa la evolución de las tres señales anteriores:

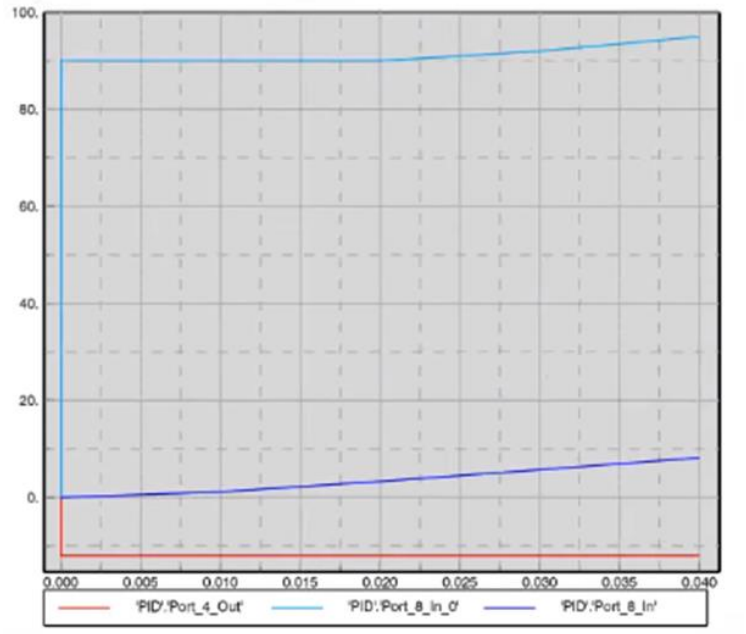


Ilustración 26. Evolución de señales del PID (1)

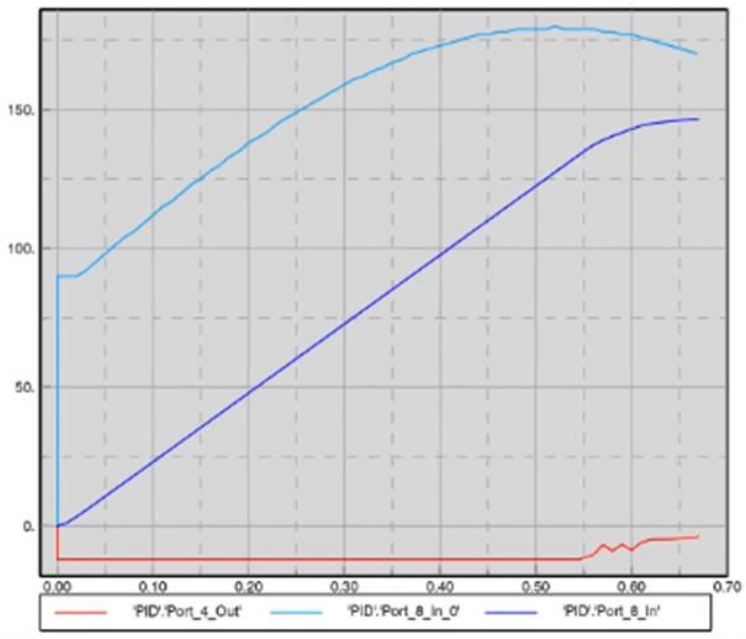


Ilustración 27. Evolución de señales del PID (2)

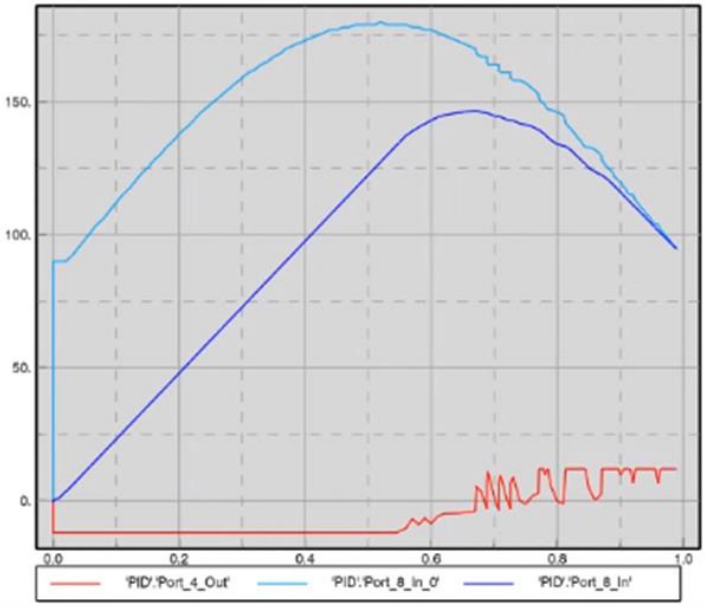


Ilustración 28. Evolución de señales del PID (3)

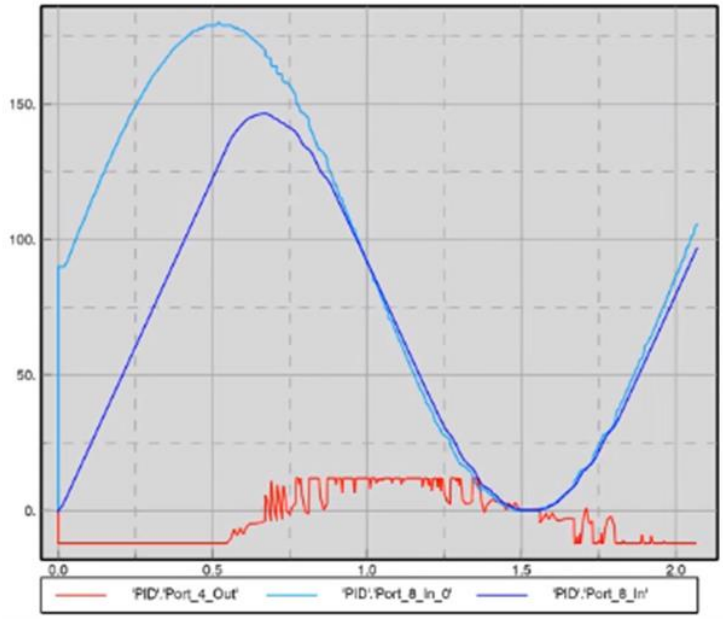


Ilustración 29. Evolución de señales del PID (4)

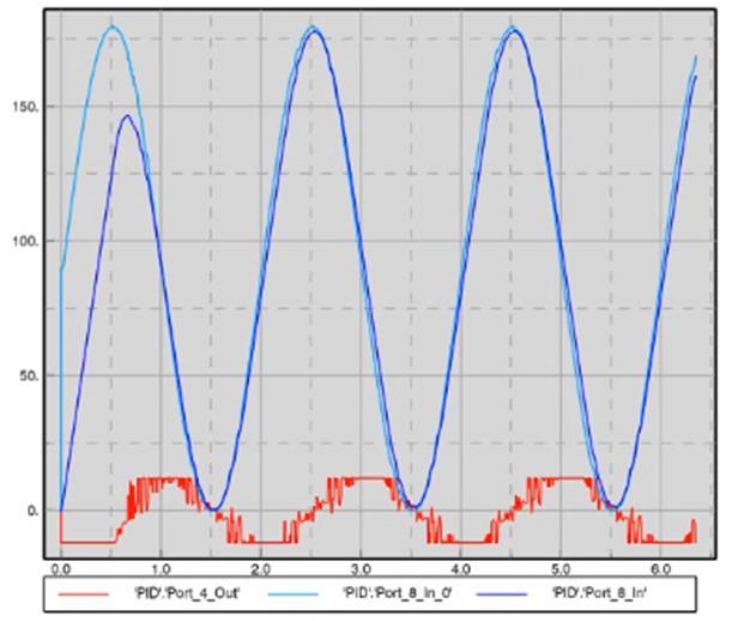


Ilustración 30. Evolución de señales del PID (5)

El controlador PID actúa de forma correcta, ya que rápidamente se consigue la posición deseada del ángulo del servomotor.

4 CONTROL DEL BRAZO ROBÓTICO

El brazo robótico ha cubierto una gran variedad de campos, incluida la industria manufacturera, el tratamiento médico, el control de seguridad y otras aplicaciones.

DFLG6DOF es un brazo robótico que consta de seis servomotores, que corresponden al brazo, el codo, el antebrazo, la muñeca (dos grados de libertad) y una base giratoria. Cada articulación puede moverse en cierto rango.

El brazo robótico consta de 6 partes con movimientos relativos de giro en un eje entre ellos. Su construcción y funcionamiento se asemeja a la de un brazo humano (robot antropomórfico), aunque habría que sustituir la mano por una pinza y se tendría que restar un giro a la muñeca. Para nombrar cada una de sus partes se aprovecha a las partes del brazo.

4.1 PARTES DEL BRAZO ROBÓTICO

A continuación, se nombran y señalan estas partes del brazo articulado:

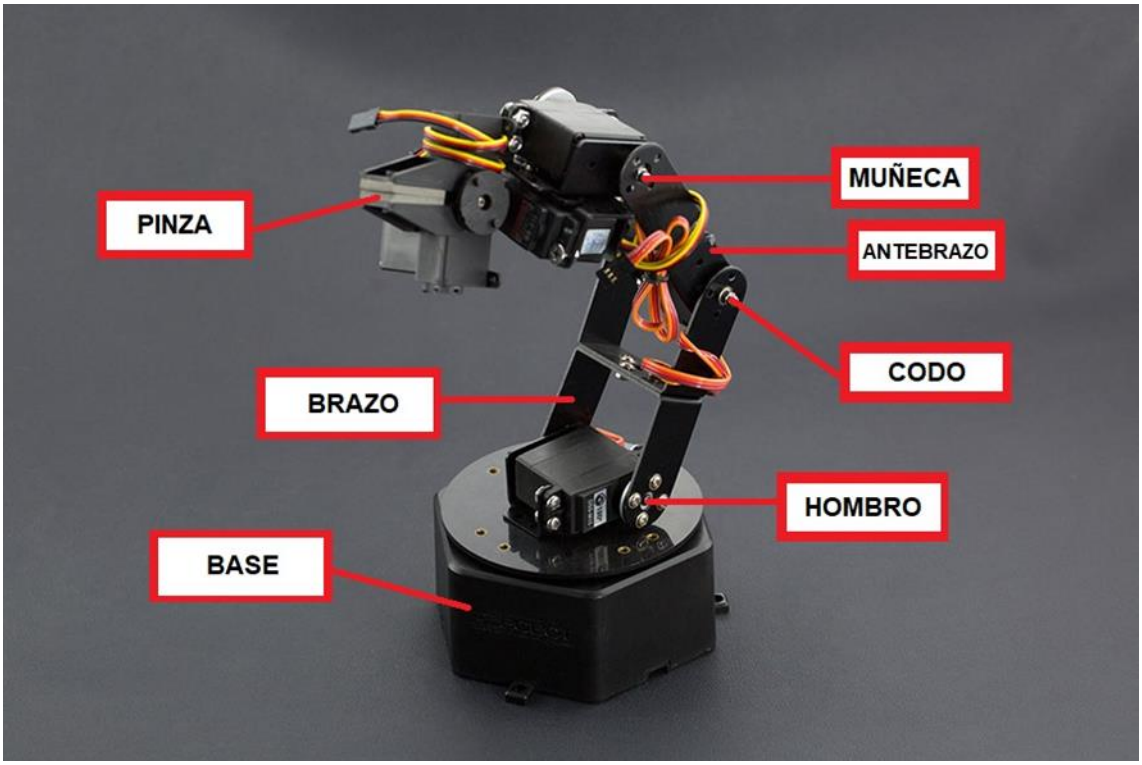


Ilustración 31. Patyes del brazo articulado

A diferencia de las demás articulaciones, en la muñeca se dan dos giros:

- Elevación de la pinza: es el movimiento de giro del conjunto pinza respecto al antebrazo con el que se consigue variar el ángulo que forman entre ellos.
- Giro de la pinza: no altera el ángulo de la pinza respecto del antebrazo.

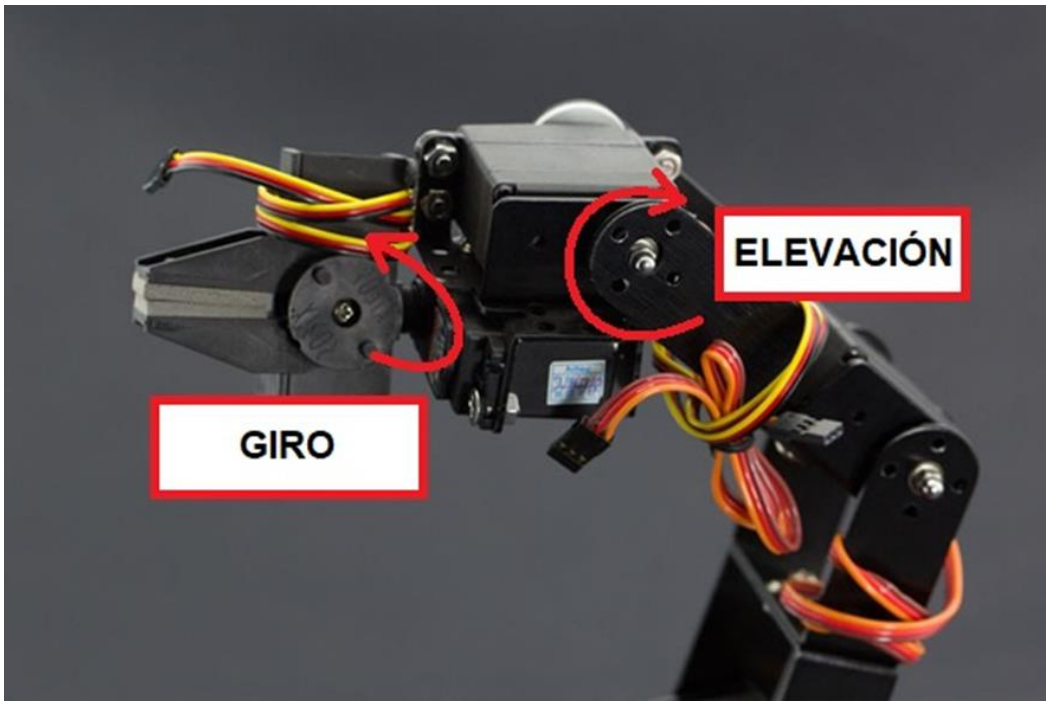


Ilustración 32. Giros de la muñeca

4.2 ESPECIFICACIONES DEL BRAZO ROBÓTICO

Este brazo robótico tiene las siguientes especificaciones:

- Tensión de funcionamiento: 4,8 – 7,2 V
- Carga máxima: 500 g
- Corriente de funcionamiento: 3 A @ 5 V (máx.)
- Tipo de interfaz: interfaz XH2.54-3P
- Longitud (ensamblado): 280 mm
- Altura (ensamblado): 340 mm
- Peso: 1096 g

4.3 HERMANAMIENTO DIGITAL DE BRAZO ROBÓTICO INDUSTRIAL EN MINIATURA

Un gemelo digital se define, fundamentalmente, como un perfil digital en evolución del comportamiento histórico y actual de un objeto o proceso físico que ayuda a optimizar el rendimiento del negocio.

El gemelo digital se basa en mediciones de datos masivas, acumulativas, en tiempo real y del mundo real en una variedad de dimensiones. Estas mediciones pueden crear un perfil evolutivo el objeto o proceso en el mundo digital que puede proporcionar información importante sobre el rendimiento del sistema, lo que lleva a acciones en el mundo físico, como un cambio en el diseño del producto o del proceso de fabricación.

Hoy en días, los métodos convencionales de diseño y operación a menudo se descuidan, ya que no es factible almacenar datos en tiempo real. Mediante el uso de modelos digitales, es posible interpretar las mediciones, los datos operativos y de la flota de una manera diferente en lugar de solo detectar desviaciones de la norma. Se pueden simular varios modos de falla para la situación actual que intenta reproducir las señales de medición reales. La comparación de las señales simuladas con las medidas puede ayudar a identificar el modo de falla. En general, el Digital Twin es una tecnología práctica, que produce mejores resultados.

Sobre el brazo robótico y su gemelo digital:

- La versión en miniatura del brazo robótico industrial está diseñada para demostrar el principio de funcionamiento del robot industrial.
- Es un modelo que tiene seis grados de libertad y trabaja en un sistema que realiza funciones de recoger y colocar un objeto.

A través de la plataforma 3DEXPERIENCE, se crea un mundo virtual del brazo robótico.

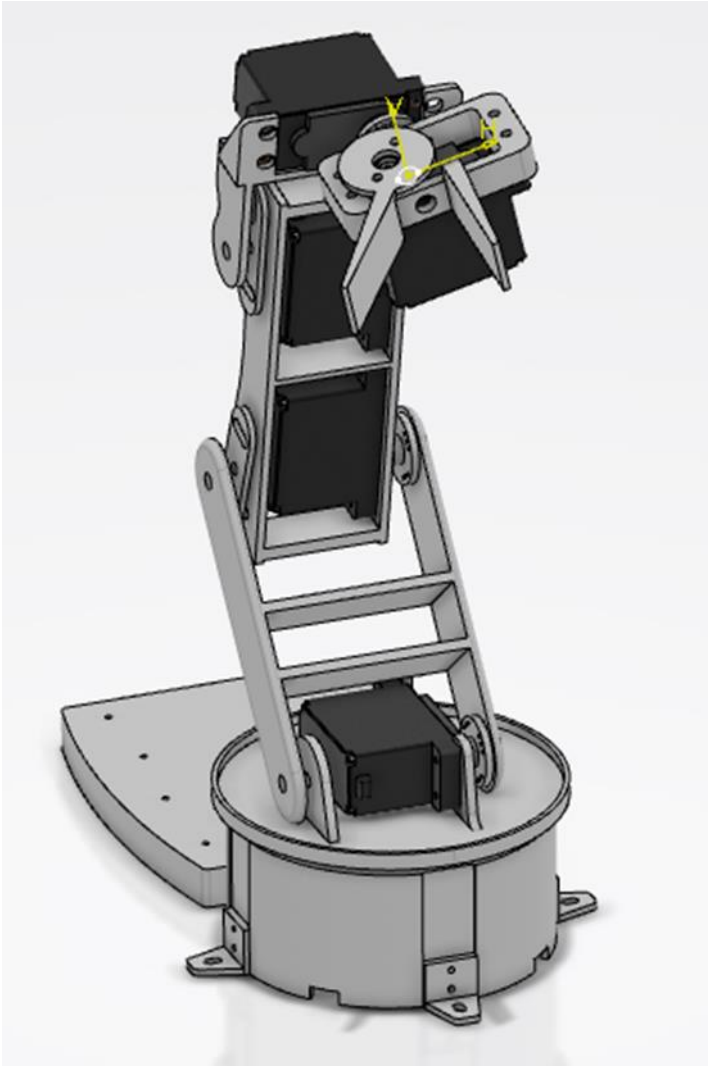


Ilustración 33. Modelo virtual del brazo robótico

Se describen las funciones y se construyen lógicas para obtener los resultados deseados del proceso. Esto se realiza en las aplicaciones RFLP y Dymola Behavior Modeling. Estas aplicaciones tienen la capacidad de construir procesos del mundo físico de manera lógica.

Para ejecutar este proceso, se agrega un conjunto predefinido de valores de ángulo en el formato de tabla.

Esta lógica se ejecutará y procesará el modelo FMI de Arduino, que enviará la información al Arduino físico conectado externamente al sistema a través del puerto USB.

Para validar la lógica, las conexiones del bloque FMI Arduino se realizan con el bloque del modelo virtual y luego se simulan. Después, se puede ver que ambos modelos, el modelo virtual presente en el sistema y el modelo físico real del brazo se comportarán de manera similar. Dado que las señales

van del sistema virtual al brazo real, se crea un bucle, que continúa continuamente enviando la señal al modelo físico desde el sistema. Por lo tanto, el proceso se llama “Virtual to Real Digital Twin”.

4.4 RANGOS DE POSICIÓN DE CADA GRADO DE LIBERTAD

La primera articulación alcanza un rango de 180 grados:

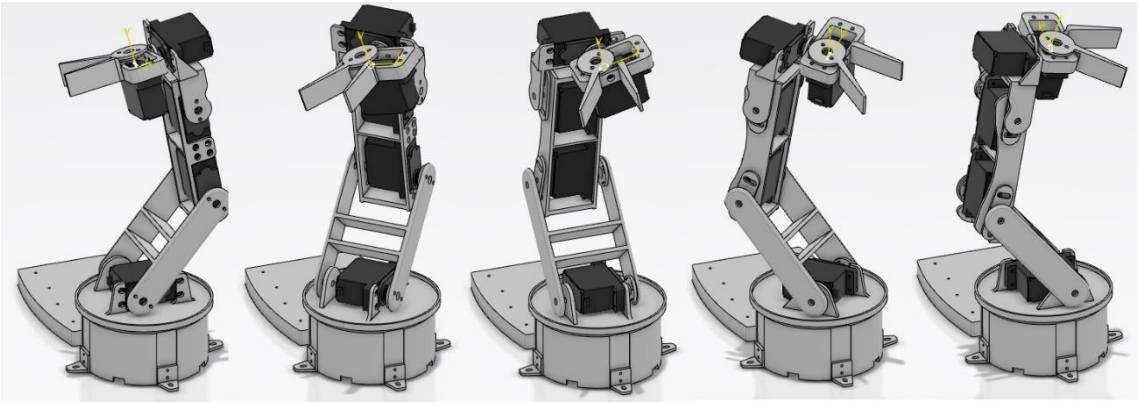


Ilustración 34. Rango de giro (1)

La segunda articulación alcanza un rango de 180 grados:

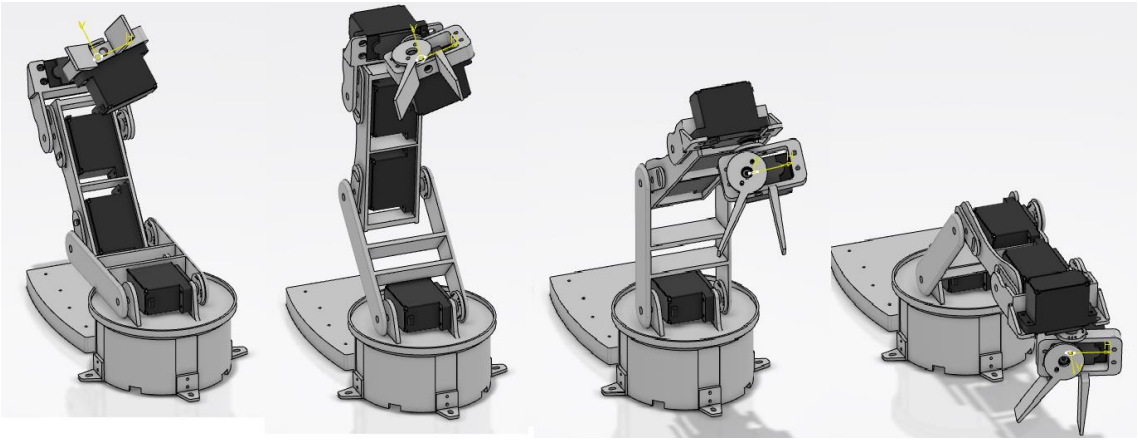


Ilustración 35. Rango de giro (2)

La tercera articulación alcanza un rango de 180 grados:

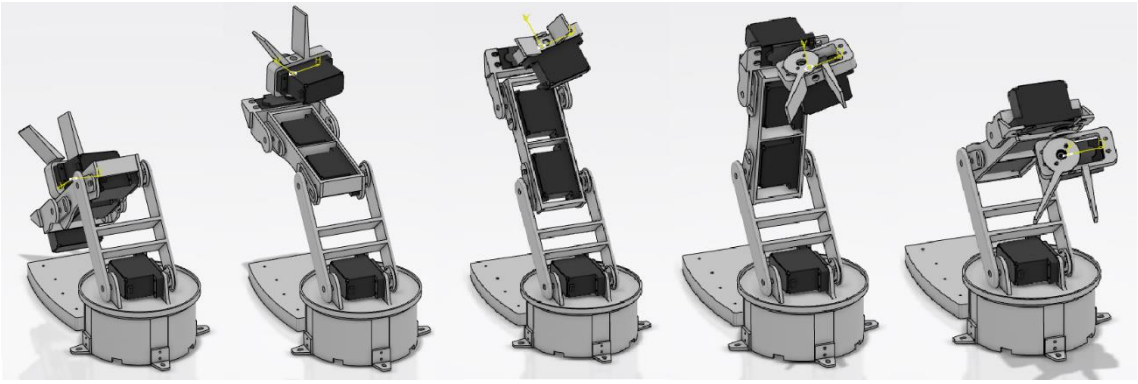


Ilustración 36. Rango de giro (3)

La cuarta articulación alcanza un rango de 180 grados:

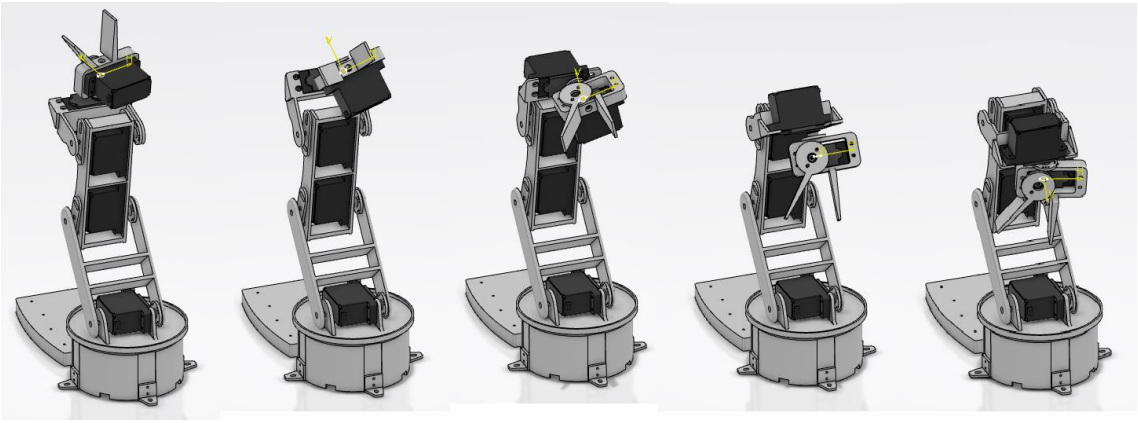


Ilustración 37. Rango de giro (4)

La quinta articulación alcanza un rango de 180 grados:

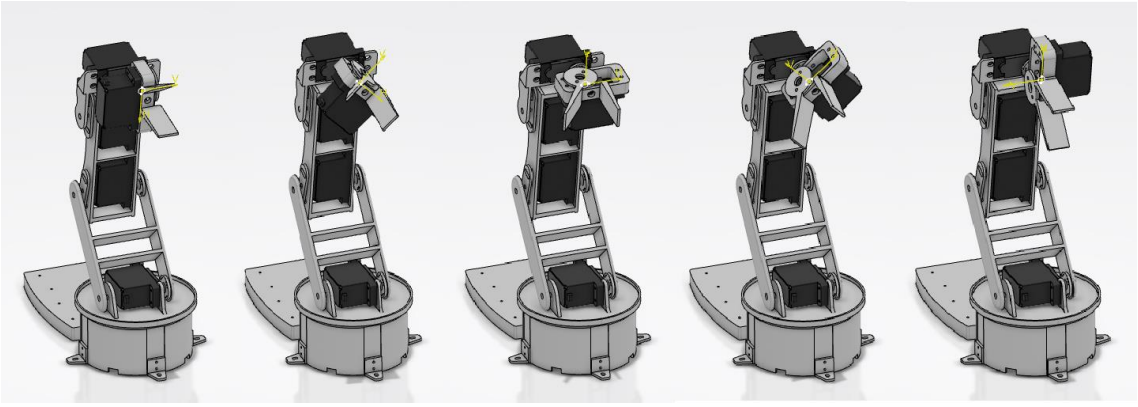


Ilustración 38. Rango de giro (5)

La apertura y cierre de la pinza alcanza un rango de 45 grados:

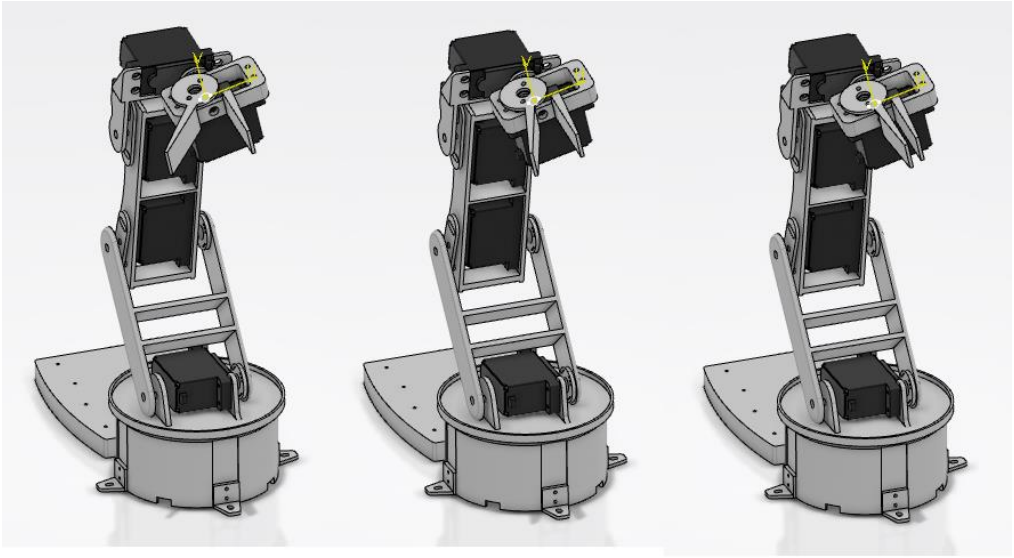


Ilustración 39. Rango de apertura de la pinza

4.5 CONEXIÓN DEL BRAZO ROBÓTICO A ARDUINO

Hay que recordar que los cables de los servomotores del brazo robótico tienen colores que corresponden a una función específica.

- Los cables negros están conectados a la masa del servomotor.
- Los cables rojos están conectados a la fuente de alimentación.
- Los cables blancos son las entradas de los servos conectados a la posición de los motores.

Los cinco agujeros en la fila de la placa están conectados entre sí internamente. Para vincular dos filas, es necesario conectar un cable en un puerto de cada fila.

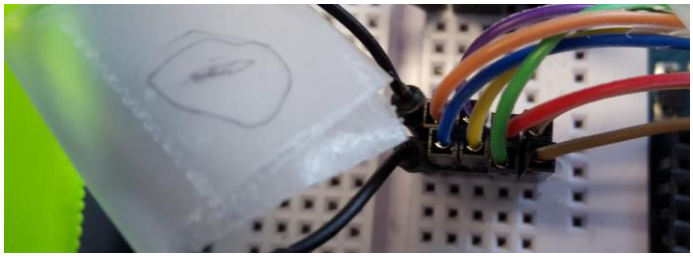


Ilustración 40. Conexión de las masas de los servomotores

Después, hay que conectar los seis cables unidos a las masas de los servomotores en dos filas de la placa de pruebas y luego conectar estas dos filas.

Los cables de posición de los motores de cada servomotor tienen su correspondiente pin de la tarjeta Arduino.

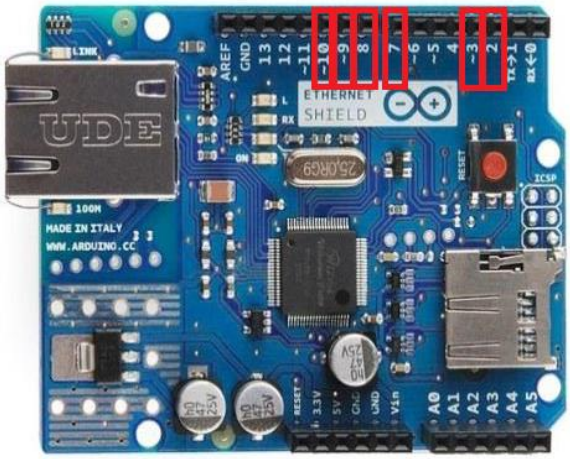


Ilustración 41. Pines de los cables de posición de los motores

Con el objetivo de ser coherente con el modelo 3DXML y el código de Arduino, se debe seguir esta configuración:

- Base: pin 2
- Hombro: pin 3
- Codo: pin 7
- Elevación de la muñeca: pin 8
- Giro de la muñeca: pin 9
- Apertura y cierre de la pinza: pin 10

4.6 DIFERENCIAS ENTRE EL CONTROL CON ARDUINO Y EL CONTROL CON 3DEXPERIENCE

Hay dos maneras posibles de controlar el brazo robótico, a saber:

- Control con ARDUINO: la tarjeta Arduino da las órdenes para comandar el modelo real y el modelo virtual. Las órdenes vienen directamente desde el código escrito en la tarjeta Arduino, así que el bloque lógico correspondiente con la orden desaparece del entorno virtual.

  Flujo de datos para la versión controlada por Arduino:

  1) La tarjeta Arduino manda las posiciones de los servomotores al brazo robótico.
  2) La tarjeta Arduino manda las posiciones de los servomotores a 3DEXPERIENCE.
  3) El bloque lógico de Arduino manda la posición al modelo virtual.

- Control con 3DEXPERIENCE: este es el caso que se estudia en este proyecto. 3DEXPERIENCE da las órdenes comandar el modelo real y el modelo virtual. Las órdenes están escritas como

un programa dentro del entorno virtual, el cual copia la posición de cada servomotor a su momento.

Flujo de datos para versión controlada por 3DEXPERIENCE:

1) La posición de los servomotores en ese momento es guardada en el primer bloque lógico.
2) Las posiciones son enviadas al bloque lógico de Arduino.
3) El bloque lógico de Arduino manda la información a la tarjeta Arduino.
4) La tarjeta Arduino traduce la información a señales eléctricas para comandar los servomotores reales.
5) La tarjeta Arduino reenvía las posiciones a 3DEXPERIENCE.
6) El bloque lógico manda la posición al modelo virtual.

4.7 FUNCIONAMIENTO DEL BLOQUE LÓGICO DE ARDUINO Y DE LA FMI

El corazón de la comunicación entre Arduino y 3DEXPERIENCE es el bloque lógico de Arduino. Este bloque lógico contiene la FMI (Functional Mock-up Interface), que sería la interfaz funcional de prototipos o modelos, la cual envía y recibe la información de la tarjeta Arduino.

Antes de cada simulación, los parámetros de la FMI necesitan ser inicializados. Estos parámetros principales son:

- *comNumber*: puerto USB conectado a la tarjeta Arduino.
- *Mastership* (no es necesario cada vez): para cambiar entre el modo controlado por Arduino o el modo controlado por 3DEXPERIENCE. En este caso, no es necesario.

Los puertos en el bloque lógico de Arduino representan los 20 pines disponibles en la tarjeta Arduino real. Para cada pin hay definidos 4 puertos lógicos en el bloque:

- Valores de entrada: estos puertos transfieren el valor desde 3DEXPERIENCE al bloque lógico de Arduino (ej.: posición de los servomotores).
- Valores de salida: estos puertos mandan los valores recibidos de la tarjeta Arduino al modelo virtual en 3DEXPERIENCE.
- Modos de entrada: estos puertos definen el modo del pin de la tarjeta Arduino al 3DEXPERIENCE (entrada, salida, etc).
- Modos de salida: estos puertos recuperan el modo del pin de la tarjeta Arduino (entrada, salida, etc).

La FMI es un código C++ introducido como un comportamiento en el entorno RFLP.

La visión de la FMI es apoyar este enfoque: si el producto real se ensambla a partir de una amplia gama de partes que interactúan de manera compleja, cada una controlada por un conjunto de leyes físicas, entonces debería ser posible crear un producto virtual que pueda ensamblarse a partir de un conjunto de modelos que representan cada uno una combinación de partes, cada uno un modelo de las leyes físicas, así como un modelo de los sistemas de control ensamblados digitalmente.

Para crear el estándar FMI, una gran cantidad de empresas de software y centros de investigación han trabajado en un proyecto de cooperación establecido a través de un consorcio europeo que ha sido dirigido por Dassault Systèmes bajo el nombre de MODELISAR. El proyecto MODELISAR comenzó en 2008 para definir las especificaciones de FMI, entregar estudios de tecnología, probar los conceptos de FMI a través de casos de uso elaborados por los socios del consorcio y permitir a los proveedores de herramientas construir prototipos avanzados.

En la práctica, la implementación de FMI mediante una herramienta de modelado de software permite la creación de un modelo de simulación que puede interconectarse o la creación de una biblioteca de software llamada FMU (Functional Mock-up Unit).

Cada unidad de maqueta funcional (FMU) se distribuye en un archivo zip con la extensión ".fmu" que contiene:

- Un archivo XML que contiene, entre otras cosas, la definición de las variables utilizadas por la FMU.
- Todas las ecuaciones utilizadas por el modelo (definido como un conjunto de funciones C).
- Otros datos opcionales, como tablas de parámetros, interfaz de usuario, documentación que pueda necesitar el modelo, etc).

Para centrar este tema en el proyecto que se trata, básicamente, la FMI toma la información que llega de 3DEXPERIENCE y la envía a la tarjeta Arduino en un lenguaje que la tarjeta pueda entender y viceversa.

Puede hacerse a través de cadenas de caracteres enviadas mediante la conexión USB o Ethernet.

El código lanzado a la tarjeta Arduino, es el mismo utilizado en el control de un único servomotor, ya que recibe la información proveniente de 3DEXPERIENCE y establece los modos y los valores de los pines de la misma manera.

4.8 SIMULACIÓN

El entorno virtual que se obtiene en 3DEXPERIENCE es el siguiente:

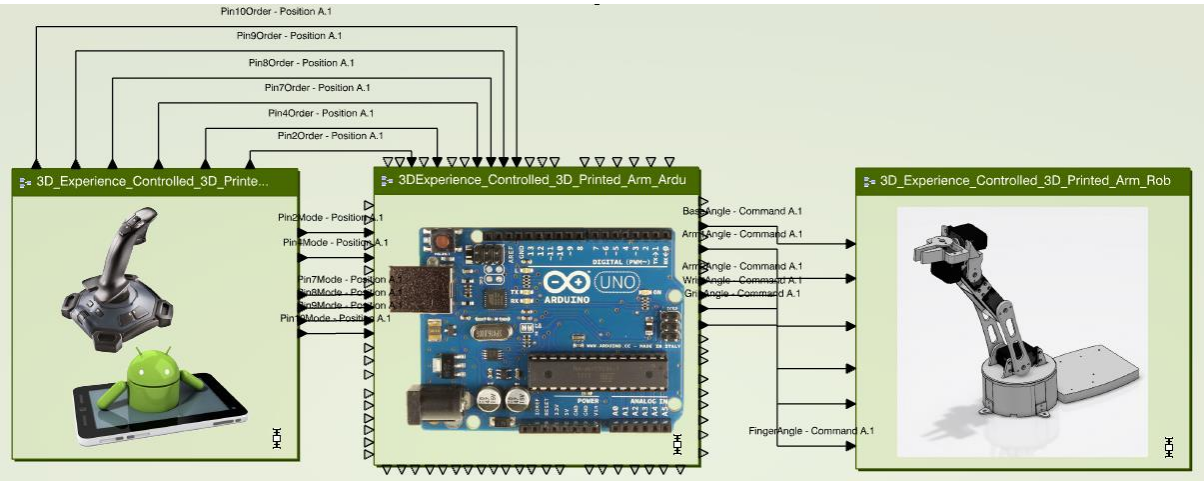


Ilustración 42. Entorno virtual del brazo robótico

Como se ve en esta distribución, el primer bloque lógico representa el elemento que envía las órdenes, seguido por el bloque lógico de Arduino y cerrando el sistema aparece el actuador, es decir, el brazo robótico.

Si en lugar de esta representación se quisiera una más tosca y primitiva, consistiría en seis sistemas similares al empleado en el control de un servomotor, aunque con un solo bloque lógico de Arduino, el cual permite controlar los seis servomotores.

3DEXPERIENCE posibilita la visualización de la posición y orientación del brazo robótico a través de la pantalla de un ordenador. Así, el usuario puede operar el robot visualizándolo en el simulador sin necesidad de estar viendo el robot real.

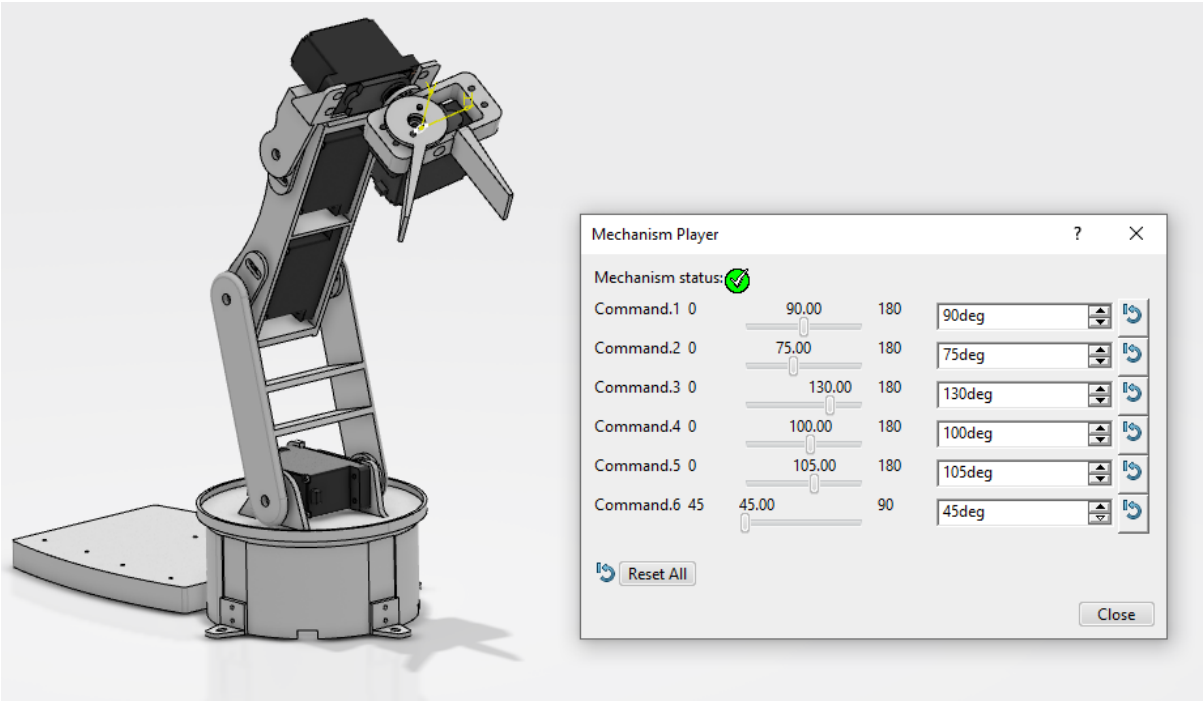


Ilustración 43. Simulador del robot virtual

Esta simulación permite controlar el robot y alcanzar el ángulo que se desee en cualquiera de las articulaciones dentro de su rango de movimiento.

5 LÍNEAS FUTURAS

Brazo robótico controlado remotamente

El Arduino Ethernet Shield ofrece la capacidad de conectar un Arduino a una red ethernet. Es la parte física que implementa la pila de protocolos TCP/IP. Está basada en el chip ethernet Wiznet W5100. El Wiznet W51000 provee de una pila de red IP capaz de soportar TCP y UDP. Soporta hasta cuatro conexiones de sockets simultáneas. Usa la librería Ethernet para leer y escribir los flujos de datos que pasan por el puerto ethernet. Permitirá escribir sketches que se conecten a Internet usando la shield.

Arduino usa los pines digitales 10, 11, 12 y 13 para comunicarse con el W1500 en la Ethernet Shield. Estos pines no pueden ser usados para entradas o salidas genéricas.

Para usar la Ethernet Shield solo hay que montarla sobre la placa Arduino. Para cargar los sketches a la placa con el shield, conectarla al ordenador mediante el cable USB como se hace normalmente. Luego conectar la Ethernet a un ordenador, a un switch o a un router utilizando un cable ethernet estándar.

Resumiendo, se puede emplear este controlador con un procesador como Arduino para implementar una comunicación por Internet y poder controlar el brazo robótico real a distancia desde un computador en el que se gobierna el modelo virtual.

6 BIBLIOGRAFÍA

DASSAULT SYSTÈMES. 3DS ACADEMY. 2017. The DS Learning Lab's Arduino Robot Project. Disponible en: https://eu1-academia-ifwe.3dexperience.3ds.com/#dashboard:1874e7d8-7126-4c4a-9315-abac15ebe0fe/tab:About

DASSAULT SYSTÈMES. 3DS ACADEMY. 2019. Digital Twinning of Miniature Industrial Robotic Arm (MIRA). Disponible en: https://academy.3ds.com/en/projects/digital-twinning-miniature-industrial-robotic-arm-mira

APRENDIENDO ARDUINO. 2015. Shields para Arduino. Disponible en: https://aprendiendoarduino.wordpress.com/2015/03/23/shields-para-arduino/

APRENDIENDO ARDUINO. 2016. Entradas y Salidas de Arduino. Disponible en: https://aprendiendoarduino.wordpress.com/2016/12/18/entradas-y-salidas-arduino-2/

INGENIERÍA, INFORMÁTICA Y DISEÑO. 2017. Conectar Arduino a Internet o LAN con Shield Ethernet W5100. Disponible en: https://www.luisllamas.es/arduino-ethernet-shield-w5100/

CADTECH. 2014. La plataforma 3DEXPERIENCE de Dassault Systémes, basada en la arquitectura V6, impulsa el éxito de las principales compañías del mundo. Disponible en: https://cadtech.es/la-plataforma-3dexperience-de-dassault-systemes-basada-en-la-arquitectura-v6-impulsa-el-exito-de-las-principales-companias-del-mundo/

PICUINO. 2015. Controlador PID. Disponible en: https://www.picuino.com/es/arduprog/control-pid.html

WIKIPEDIA. 2015. Functional Mock-up Interface. Disponible en: https://reference.wolfram.com/system-modeler/UserGuide/ModelCenterFunctionalMockupInterface.html

ÁREA TECNOLOGÍA. RECUSOS CONOCIMIENTOS Y TEMAS DE TECNOLOGÍA. 2017. Servomotores. Disponible en: https://www.areatecnologia.com/electricidad/servomotor.html

GUBBI, J.; BUYYA, R.; MARUSIC, S.; PALANISWAMI, M. 2012. Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. En: *Open Access Cornell University*. Disponible en: https://arxiv.org/ftp/arxiv/papers/1207/1207.0203.pdf

LLATA J.R.; SARABIA E.G.; TORRE-FERRERO, C.; SACIBRIAN, R. 2017. Educación en Automática e Industria 4.0 mediante la aplicación de Tecnologías 3D. En: *Actas de las XXXVIII Jornadas de Automática, Gijón, 6, 7 y 8 de septiembre de 2017.* Universidad de Cantabria.