Universidad del País Vasco
Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FIN DE GRADO

CONTROL DE SISTEMA BALL AND BEAM MEDIANTE SIMULINK-ARDUINO

DOCUMENTO I- *MEMORIA*

Alumna: Brenlla, Garcia, Henar
Director (1): Casquero, Oyarzabal, Oskar
Directora (2): Armentia, Diaz de Tuesta, Aintzane

Curso: 2018-2019

Fecha: Bilbao, 17 de julio de 2019

RESUMEN

El control PID es uno de los controladores más empleados por su capacidad de conseguir una buena respuesta en sistemas realimentados, pero para ello, cada parámetro debe ser ajustado correctamente en un microcontrolador. Su programación puede realizarse en código C, ensamblador, entorno gráfico, etc. Afinar un control PID programando un microcontrolador Arduino requiere de varios intentos hasta conseguir una respuesta válida y por ello se ha pensado optimizar el tiempo que se emplea programando el ajuste vía Simulink. Mediante esta aplicación se ha desarrollado un control PID en Simulink utilizando un sistema no lineal "ball and beam" como punto de partida. Este proyecto emplea Arduino y Simulink para controlar la respuesta de un servomotor y mantener una pelota a una distancia específica. Un sensor infrarrojo de distancia medirá la distancia a la se encuentra la bola y un control PID ajustará una señal parecida a una PWM pero que en realidad es una señal de servo de pulso variable que recibirá el servomotor para mantenerla a la distancia deseada. Para implementar el código Arduino en Simulink, se ha creado un bloque personalizado partiendo del bloque "s-function builder". El modelo completo crea un algoritmo de control PID que se descarga en la placa de Arduino y se ejecuta de manera independiente (standalone) a Simulink. Ejecutando el sistema en Simulink es posible afinar los parámetros del control PID en Simulink de manera externa en tiempo real y no cargar un programa por cada cambio en la placa de Arduino, ahorrando así tiempo.

Palabras clave: Sistema no lineal, PID, Simulink, s-function builder, ball & beam.

LABURPENA

PID kontrolatzailea erabilienetariko bat da berrelikaduradun sistemetan lor dezakeen erantzun onagatik. Horretarako parametro bakoitza era egokian doitu behar da mikrokontrolatzaile bat erabiliz. Bere programazioa C kodez, mihiztatzailez, ingurune bisualez, etab. bitartez egin daiteke. PID kontrolatzaile baten doiketa fina lortzeko beharrezkoa da Arduino mikrokontrolatzailearentzako programan haztamuka parametro egokiak lortzea eta horregatik denbora optimizatzeko doiketa Simulink bidez programatzea pentsatu da. Aplikazio honen bitartez, PID kontrolatzaile bat garatu da Simulink-en “ball and beam” sistema ez lineal batetan oinarrituz. Proiektu honek Arduino eta Simulink erabiltzen ditu serbomotore baten erantzuna kontrolatzeko eta horrela distantzia espezifiko batera mantentzeko pilota. Infragorrizko distantzia sentsoreak pilota zein distantziara dagoen neurtuko du eta PID kontrolatzaileak serbomotoreak eskuratuko duen PWM seinale baten antzeko pultsu aldakorreko serbo seinale bat sortuko du, pilota distantzia egokira mantenduz. Arduino kodea Simulink-en inplementatzeko bloke pertsonalizatu bat sortu da, “s-function builder” blokea. Modelo osoa exekutatzean PID algoritmo bat sortzen da non Arduino plakara deskargatu ahal den eta era honetan Simulink-ekiko era independentean exekutatu (standalone). Simulink-en exekutatuz gero, parametroen doiketa denbora errealean eta kanpotik egitea lor daiteke Arduinoko programan egindako aldaketa bakoitzeko programa bat kargatu behar izatea ekidinez, denbora aurreztuz.

Gako-hitzak: Sistema ez lineala, PID, Simulink, s-function builder, ball & beam.

ABSTRACT

A PID controller is one of the most used controllers for its ability to achieve an accurate response in feedback mechanisms, but due to this, each parameter must be regulated correctly with a microcontroller. Its programming can be done in C code, assembler, graphical notation, etc. Tuning a PID controller by programming an Arduino microcontroller requires several attempts until a valid behaviour is obtained. In order to optimize the time used reprogramming the parameters, they can be acquired via Simulink. In this project, a PID controller has been developed with Simulink using a ball and beam nonlinear system as a starting point. By using Arduino and Simulink together, the response of a servomotor can be controlled so the ball keeps at a specific distance. An infrared distance sensor will measure the distance of the ball and a PID controller will create a servo signal of variable pulse similar to a PWM signal that the servomotor needs to keep the ball at the desired distance. To implement the Arduino code in Simulink, a custom block was created using the s-function builder block. By executing the model of the system, a PID control algorithm is created and it can be downloaded to the Arduino board so it becomes a standalone device running independently of Simulink. When the model is executed in Simulink, it is possible to tune the parameters of the PID controller externally and in real time using Simulink, to avoid loading each new program to the Arduino board and, therefore, saving time.

Keywords: Nonlinear system, PID, Simulink, s-function builder, ball & beam.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

PID	*Proportional-Integral-Derivative controller*
MEX	*Matlab executable*
TLC	*Target Language Compiler*
PWM	*Pulse Width Modulation*
ISR	*Interrupt Service Routine*
GND	*Ground Node*
PWR	*Power*
GPIO	*General Purpose Input/Output*
Vcc	*Common Collector Voltage*
Vout	*Output Voltage*
CPP	*C pre-processor*
C	*Programming Language*
ADC	*Analog Digital Converter*
TF	*Transfer Function*
Ts	*Sample time*
IR	*Infrared Radiation*
HW	*Hardware*
SW	*Software*
DC	*Direct Current*
FFF/FDM	*Fused Filament Fabrication/Fused Deposition Modelling*
PLA	*Polylactic Acid (Thermoplastic Aliphatic Polyester)*
ABS	*Acrylonitrile Butadiene Styrene (Thermoplastic Polymer)*
STL	*Standard Triangle Language*
CAM	*Computer Aided Manufacturing*
CAD	*Computer Aided Design*
MUX	*Multiplexer*
EEPROM	*Electrically Erasable Programmable Read-Only Memory*
SRAM	*Static Random-Access Memory*
ISP	*In-System Programming*
IDE	*Integrated Development Environment*

1. INTRODUCCIÓN

A la hora de diseñar una planta segura y productiva se necesitan controladores industriales. Gran variedad de controladores se utiliza para el control de procesos industriales. Entre ellos se encuentra uno de los más simples y eficaces: el controlador PID. Es mucho más práctico que el típico controlador on/off porque permite un mejor ajuste del sistema. A causa de esto, muchos controladores industriales usan esquemas de control PID. Sistemas hidráulicos, neumáticos, electrónicos, etc. se transforman en formas digitales mediante el uso de microcontroladores y se les aplica este tipo de control, el cual resulta útil cuando el modelo matemático de la planta se desconoce y no es posible aplicarle métodos de diseño analítico.

Hoy en día el mercado ofrece productos que implementan las técnicas de control a analizar en este proyecto, aunque cabe señalar la diferencia sobre la finalidad de la aplicación. Podemos encontrar sistemas comerciales y sistemas no comerciales. En este caso el prototipo diseñado y desarrollado en la UPV/EHU no tiene un fin económico.

En este proyecto se utiliza un microcontrolador Arduino pues es la opción más económica y de las más utilizadas en este tipo de aplicaciones actualmente. Además, Arduino trabaja en tiempo real respondiendo a su entorno en un tiempo establecido, cosa que lo convierte en la plataforma más adecuada en este caso. A su vez, para la implementación del controlador PID se hace uso de un modelo de bloques diseñado en Simulink por la flexibilidad que ofrece gracias a la opción de implementar código C. De este modo se ve la posibilidad de interconectar distintas plataformas para facilitar el desarrollo de un control sobre un sistema no lineal.

2. CONTEXTUALIZACIÓN

El diseño y control de un sistema ball and beam (barra y esfera) es de los más comunes y simples para estudiar técnicas básicas de control. El hecho de que se trate de un sistema mecatrónico que está compuesto por una base, una barra y una esfera, la cual rueda libremente a lo largo de la barra, lo convierte en un sistema no lineal e inestable. Para su control se implementa un sistema de realimentación en bucle cerrado que muestra cómo la entrada cambia el comportamiento de la salida.

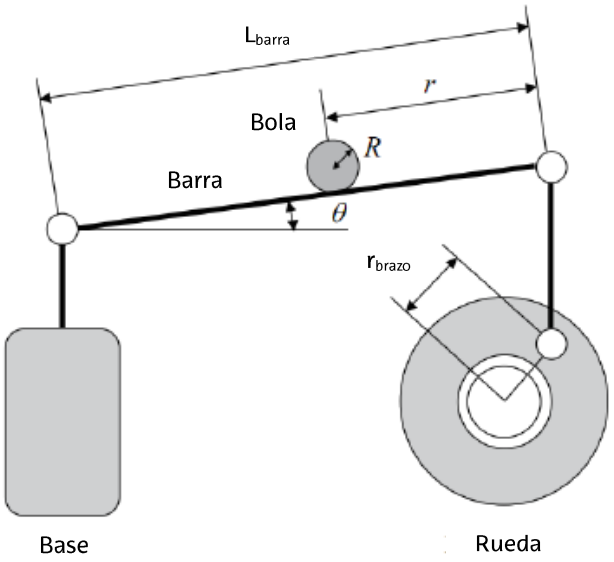


Figura 2.1. Esquema de sistema ball and beam

Fuente. http://cort.as/-LEKw

En este proyecto se desarrolla la implementación antes comentada en la introducción, es decir, el control de un sistema ball and beam mediante simulink-arduino. Se ejecuta un modelo de bloques capaz de procesar código C para detectar la distancia de la pelota por sensor de distancia colocado en uno de los extremos de la barra. Para obtener el punto deseado en el que detener la bola, el movimiento de la barra será controlado por un servomotor que hace de actuador. En base a la distancia leída por el sensor en cada momento, un bloque de control PID discreto realizará los cambios necesarios para alcanzar la consigna en el tiempo establecido y actualizará la orientación de la barra a través del actuador.

3. OBJETIVOS Y ALCANCE DEL TRABAJO

El objetivo principal de este proyecto es equilibrar una bola a una distancia determinada sobre una barra desde cualquier posición inicial. Para diseñar un controlador para un sistema ball and beam se han desarrollado los siguientes objetivos secundarios:

- SW:
  - Captura de los datos analógicos leídos por el sensor de distancia.
  - Conversión de los datos obtenidos mediante regresión polinomial.
  - Control del ángulo del servomotor a través de un mapeo con la distancia medida.
  - Elección correcta del tiempo de muestreo.
  - Control del sistema de dos modos para comparar su flexibilidad:
    - MATLAB: utilizar Simulink para la creación de un modelo de bloques en el que se implementa código C con un bloque s-function builder y otro sin implementar código de alto nivel.
- HW:
  - Diseño e impresión de piezas 3D.
  - Construcción de una maqueta Ball and Beam.
  - Sintonización adecuada del PID para la planta.

La distancia leída por el sensor se logrará mediante programación en Arduino y después se realizará el proceso de lograr ajustar el PID en el entorno de desarrollo Simulink de Matlab utilizando el Arduino como puente entre la planta y Simulink y así mantener el programa con el PID ejecutándose continuamente en Matlab y hacer cambios en la planta en tiempo real sin afectar en el comportamiento del Arduino que será únicamente un organizador de entradas y salidas.

Al tratarse de un proyecto "Open Source" ofrece la posibilidad de utilizar el mismo código adaptándolo a HW/SW distinto para cualquier usuario. Servirá de guía para futuros proyectos en los que se busque implementar C en Simulink.

4. DESCRIPCIÓN DE REQUERIMIENTOS

Para cumplir los objetivos marcados este sistema constará de un sensor que detecte la posición de la pelota, un servomotor que con su movimiento oriente las dos barras y un controlador PID que estabilice la posición de la pelota según la consigna elegida.

Por tanto, en el desarrollo del proyecto se cumplirán las siguientes condiciones técnicas:

- HW:
  - Barra(s).
  - Pelota.
  - Estructura de la planta.
  - Sensor de distancia.
  - Actuador.
  - Microcontrolador:
    - Pin digital para el actuador.
    - Pin analógico para el sensor.
    - Pin de alimentación para el sensor y el actuador.
    - Pin de toma tierra para el sensor y el actuador.
    - Un timer para la Interrupción.
    - Un timer para generar la selal de control del actuador.
  - Cables de conexión.
  - PC.

- SW:
  - Arduino IDE para desarrollar el código y realizar la lectura del sensor de manera eficiente.
  - Implementar código C en MATLAB/Simulink.
  - Pantalla interactiva de Simulink para controlar la planta.

También se cumplirán las siguientes condiciones funcionales:

- Planta:
    - Cuando el sistema se encuentre en reposo, es decir, cuando la barra sea paralela al suelo, el valor de la señal del servomotor se debe tomar como 0º.
        - El rango de operación del servo se tomará de -90º a 90º, esas serán las inclinaciones máximas de la barra.
    - La distancia mínima de recorrido de la planta debe ser de 450mm.

5. ANÁLISIS DE ALTERNATIVAS

En este punto se analizan las alternativas disponibles en el mercado para la elección de los componentes y dispositivos con el fin de desarrollar el diseño propuesto de la mejor manera posible.

La aplicación consiste en que un sensor de proximidad detecte una bola que rueda libremente sobre una barra. La barra debe poder orientarse con un actuador controlado por una placa basada en un microcontrolador. El control se hará mediante un controlador PID que estabilice la posición de la bola en la barra según la consigna. Las alternativas se analizan según su propósito en esta aplicación y así se decidirá cuál es la mejor opción para este proyecto.

5.1 HARDWARE

5.1.1 SENSOR DE DISTANCIA

Se ha valorado el uso de sensores de distancia infrarrojos y de ultrasonidos. Entre los de ultrasonido encontramos el HC-SR04 de salida digital y el US-016 de salida analógica. Aunque los sensores ultrasónicos tengan mayor alcance y precisión, no es necesario en la aplicación elegida y por ello y por los 4 pines de conexión que necesitan frente a los 3 pines que necesita el sensor infrarrojo se ha decidido utilizar el Sharp GP2Y0A21YK. (Naylamp mechatronics, 2019)

El sensor IR seleccionado permite medir la distancia a la que se encuentra el objeto mediante la reflexión de un rayo IR sobre este. Consta de una salida analógica que puede convertirse con un ADC en un microcontrolador. Su rango de operación está entre los 10 y los 80 centímetros. Los 3 pines que necesita son: Vcc (5V), GND y Vout.

Tabla 5.1. Alternativas de sensores de distancia

	Rango 10%	Conexión 20%	Disponibilidad 20%	Precisión 20%	Precio 30%	Resultado
Sharp GP2Y0A21	9	9	8	7	8	8,1
HC-SR04	8	7	8	7	9	7,9
US-016	10	7	8	9	6	7,6

Fuente. Elaboración propia.

5.1.2 MOTORES

A la hora de elegir motor para el sistema se ha tenido en cuenta la facilidad de uso del servomotor frente al motor de pasos. Además, tras el análisis destaca frente a su competidor. (BricoGeek, 2019)

El servomotor Futaba S3003 resulta una gran opción para el desarrollo de este tipo de aplicaciones al ser capaces de dar giro o torque a elementos de la planta. Para accionarlo sólo se necesita una conexión Vcc (5V), una conexión GND y una conexión para la señal de control. (Naylamp mechatronics, 2019)

Tabla 5.2. Alternativas de motores

	Dimensiones 10%	Facilidad de uso 20%	Peso soportado 20%	Ancho de pulso 20%	Precio 30%	Resultado
Servomotor Futaba S3003	8	8	9	8	9	8,5
Motor de pasos Nema 17	8	7	5	8	8	7,2

Fuente. Elaboración propia.

5.1.3 MICROCONTROLADOR

Se ha descartado usar Raspberry por ser una minicomputadora con SO propio que no trabaja como Arduino en tiempo real.

El criterio seguido a la hora de elegir placa Arduino ha sido el siguiente:

- Tamaño (no es de gran importancia ya que haremos uso de un ordenador).
- Programación o suministro eléctrico en la misma placa.
- Conexiones necesarias.
- Timers de 16 bits disponibles.

Se ha descartado usar Raspberry por ser una minicomputadora con SO propio que no trabaja como Arduino en tiempo real.

Como en este proyecto no hay limitación de tamaño se ha optado por una placa Arduino Mega por ser la más potente y tener la mayor memoria.

(La tecnología moderna, 2019). Las conexiones se hacen en la placa sin necesidad de una protoboard. (Ingeniería Electrónica. org., 2015).

- Número de pines:
  - 3 para el servomotor (Vcc, señal de control del servo y GND).
  - 3 para el sensor de distancia (Vcc, GND y Vout).
  - Total: 2 pines GPIO + 2 pines de alimentación (Vcc y GND).
- Puertos Serie:
  - Mega: 4
  - Uno: 1
- Número de Timers:
  - Mega: 6 (3 de ellos de 16 bits)
  - Uno: 3 (1 de ellos de 16 bits)

Tabla 5.3. Alternativas de microcontroladores

	Tamaño 10%	Disponibilidad 20%	Programación 20%	Cantidad I/O 20%	Precio 30%	Resultado
Arduino Mega	7	9	8	9	6	7,7
Raspberry Pi 3B	6	9	6	7	5	6,5
Arduino Uno	8	9	8	6	7	7,5

Fuente. Elaboración propia.

Como puede verse, con una placa Arduino Uno no sería suficiente porque se necesita un timer para generar una ISR y otro para generar la señal para el servomotor. Como se disponía de placas Arduino Mega 2560 y Mega ADK y con vistas a ampliar el proyecto incorporando un LCD se ha elegido finalmente hacer uso de una placa Arduino Mega ADK. (Aprendiendo Arduino, s.f.)

5.2 SOFTWARE

5.2.1 ENTORNOS DE PROGRAMACIÓN

MATLAB es el entorno de computación numérica más popular y comercializado y Octave es su alternativa Open Source, pero con limitaciones. Ambos son capaces de ofrecer una gran eficacia a la hora de

resolver problemas. Cabe destacar que MATLAB posee unos toolbox especializados que no pueden utilizarse con otro SW por licencia, además, es más rápido que Octave.

Octave es casi por completo compatible con MATLAB, pero como en la universidad disponemos de un aula con ordenadores que tienen MATLAB, por comodidad y posibilidades que este ofrece a la hora de conectarse con un microcontrolador con varios toolbox, se ha escogido dicho entorno.

Tabla 5.4. Alternativas de entorno de programación

	Uso industrial 20%	Programación 20%	Compatibilidad 10%	Calidad del SW 30%	Precio 20%	Resultado
MATLAB	8	8	8	10	5	8
GNU Octave	7	7	7	7	10	7,6

Fuente. Elaboración propia.

6. SELECCIÓN DE LA SOLUCIÓN PROPUESTA

Tras analizar las alternativas éstas son las elecciones para el diseño final:

Hardware: Procesamiento

- Modelo: Arduino Mega 2560 o ADK.
  - Pin digital para el servomotor.
  - Pin analógico para el sensor.
  - Pin de alimentación para los componentes.
  - Timer 5 para generar señal de servomotor.
  - Timer 1 para generar una ISR.
- PC: Requerimientos mínimos.
  - RAM: 4 GB.
  - Windows: 7, 10, Server.
  - Procesador: Intel o AMD x86-64.
  - Espacio en Disco: 2.9 GB sólo MATLAB, 5-8 GB para instalación por defecto. 29 GB para una instalación completa.

Hardware: Componentes y materiales

- Sharp GP2Y0A21YK: Sensor de distancia IR.
- Futaba S3003: Servomotor con un ángulo de giro de 180 grados.
- Piezas PLA del prototipo de impresión 3D: Seis piezas en total detalladas más adelante.
- Cables: Jumpers de conexión y USB tipo A/B para Arduino.
- Barras de aluminio:
  - Longitud: 480mm.
  - Diámetro: 5mm.
  - Separación entre barras (de centro a centro): 20mm.
- Pelota
  - Radio: 50mm.
  - Peso: 41g.

Software:

- Programación: Arduino IDE (v.1.8.5. utilizada).
- Programación de modelo de bloques: MATLAB/Simulink v.2018a.
    - Toolbox Arduino para Simulink.

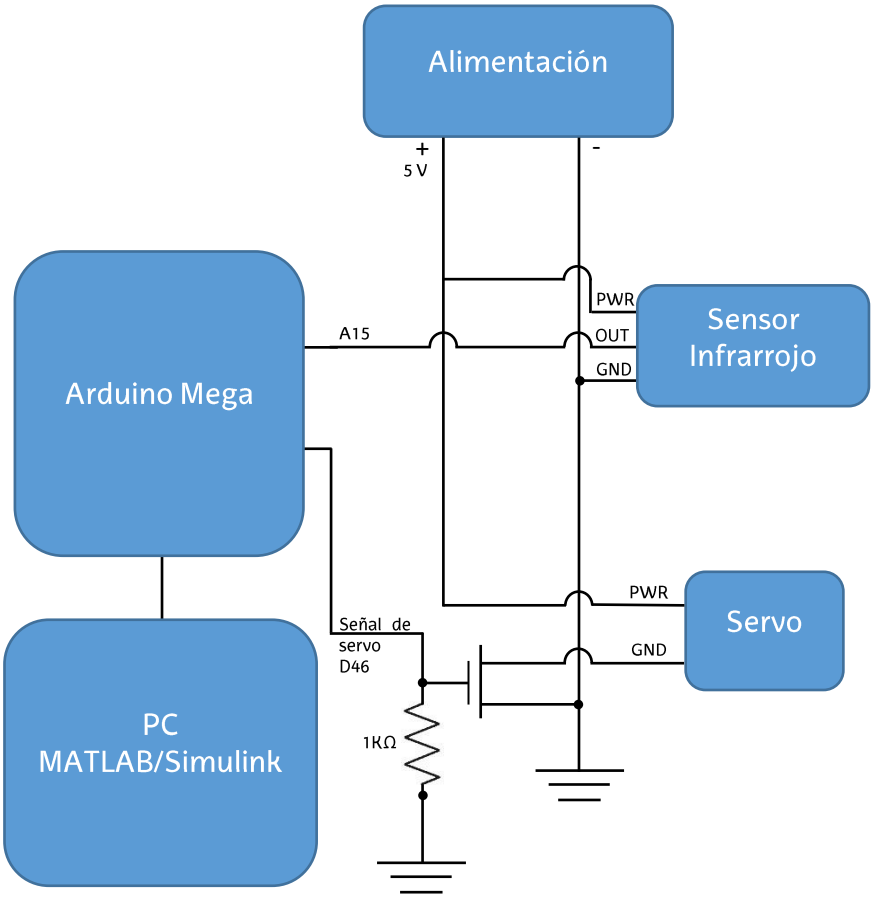


Figura 6.1. Esquema eléctrico del sistema.

Fuente. Elaboración propia.

Como se puede ver en la figura anterior, el sensor se conecta a un pin analógico, el pin A15 en este caso, porque envía una señal analógica al microcontrolador para procesarla en el conversor ADC integrado. Una vez obtenida la lectura en valor digital y convertida a centímetros será comparada con el valor de consigna elegido y la diferencia pasará a ser la entrada de un controlador PID. La salida del PID contiene entonces el valor del ángulo necesario para devolver la bola al lugar deseado. Ese valor se envía codificado al servomotor a modo de señal PWM por un pin digital de la placa, en este caso el pin D46.

Lo primero que debe hacer el sistema es detectar la pelota que rueda libremente sobre las barras con el sensor de distancia IR Sharp. Para ello, la señal analógica que envía el sensor será tratada en el ADC integrado en el microcontrolador Arduino Mega para recibir la lectura digitalizada. Una vez digitalizada se convertirá el valor a centímetros mediante una regresión polinómica. El valor será comparado con la consigna establecida y el error será enviado al controlador PID discreto con aproximación trapezoidal para regular la señal de control que necesita el servomotor para ajustar la distancia a la que debe quedar la pelota.

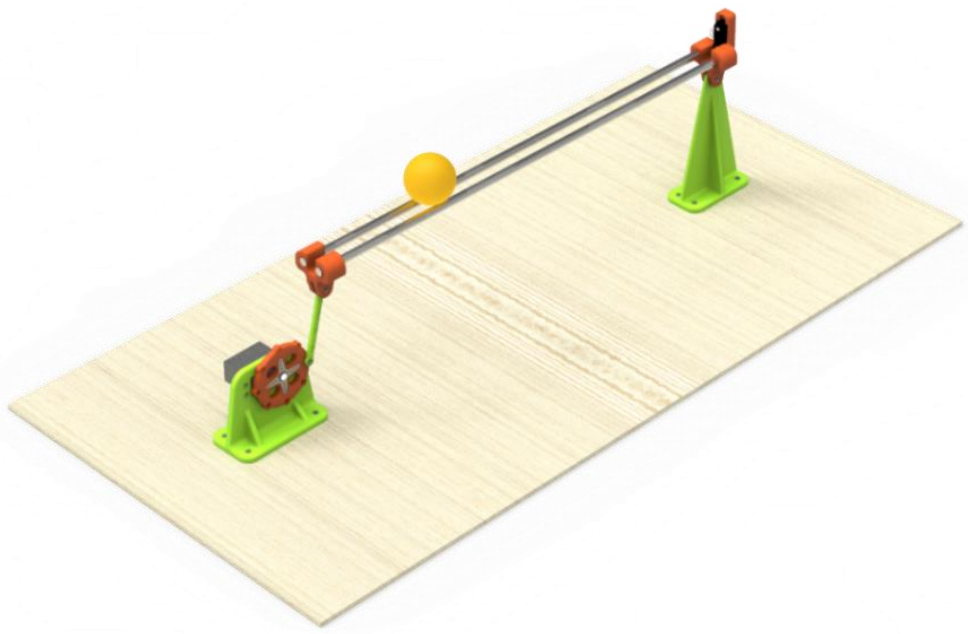


Figura 6.2. Vista isométrica del prototipo.

Fuente. Elaboración propia.

Teniendo en cuenta los requisitos del sistema, en la Figura 6.1. podemos ver el diseño en 3D del diseño propuesto.

En este apartado se resume el diseño propuesto para profundizar en él en el apartado de diseño. El sistema desarrollado físicamente está formado por tres partes fundamentales: la del sensor, la del controlador PID y la del servomotor. A continuación, se puede observar un diagrama de bloques del sistema diseñado:

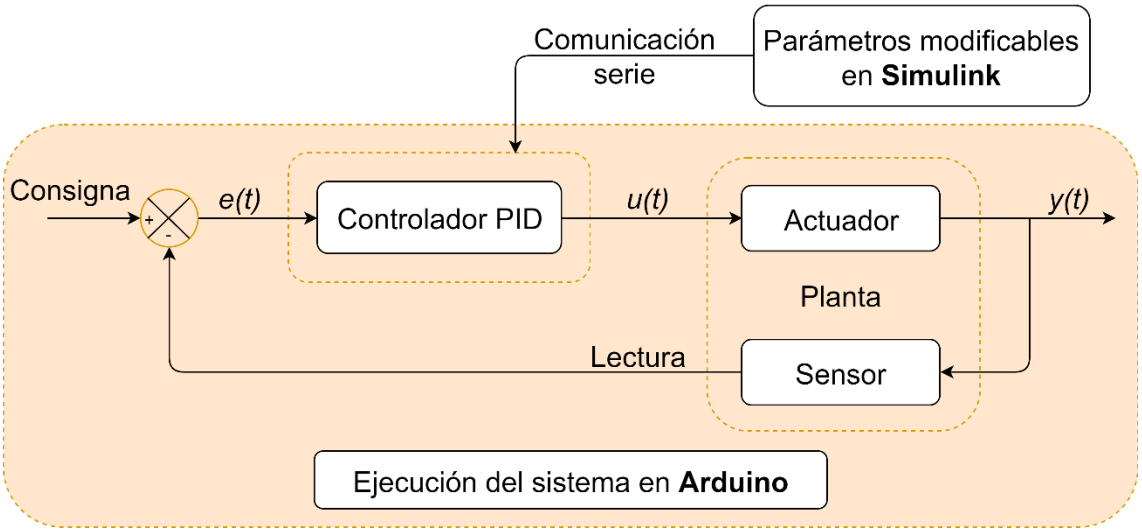


Figura 6.3. Diagrama de bloques del sistema.

Fuente. Elaboración propia.

7. CONCEPTOS PREVIOS

7.1 ARDUINO

Arduino es una plataforma creada en 2003 que diseña y produce placas de desarrollo de HW y SW libre con las que crear distintas aplicaciones. Cada placa se compone de circuitos impresos con su microcontrolador y disponen de los elementos necesarios para conectar periféricos a los pines GPIO del microcontrolador. (Aprendiendo Arduino, s.f.)

Al ser una plataforma de código abierto, sirve para el desarrollo de proyectos electrónicos de HW flexible y de uso fácil. Para la programación de su HW se encuentra disponible para el usuario un entorno de desarrollo integrado llamado Arduino IDE.

7.1.1 ARDUINO IDE

Este es el entorno de desarrollo de código abierto de Arduino. Está diseñado para facilitar la programación de la electrónica y darles así distintos usos. Aquí se realiza la programación de todas las placas de Arduino. El código queda dividido en dos partes: inicialización y bucle. Una vez realizada la inicialización la placa se mantendrá ejecutando continuamente el trozo de código correspondiente al bucle. Su aspecto es el siguiente:

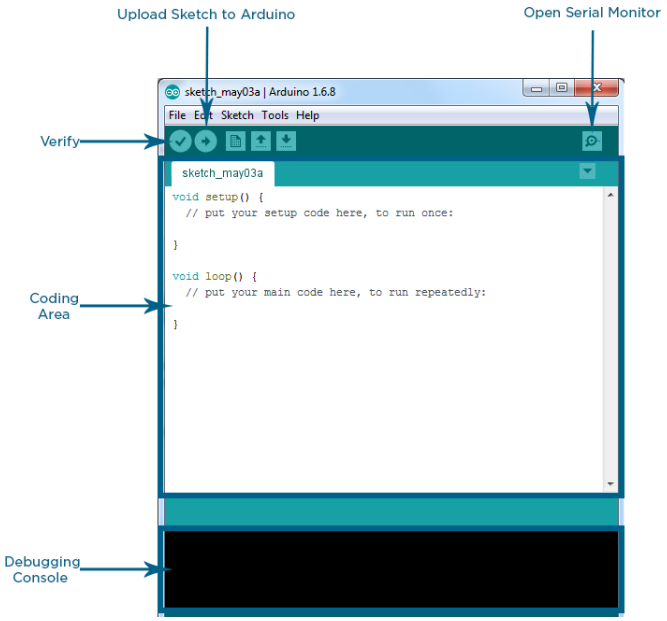


Figura 7.1. Entorno de programación Arduino IDE.

Fuente. http://cort.as/-LbuJ.

7.2 IMPRESIÓN 3D

Existen diferentes tecnologías de impresión 3D en el mercado actual. Pero para elaborar una pieza todas hacen uso de un proceso aditivo. Este proceso consiste en crear un objeto añadiendo el material por capas.

El método más extendido es el llamado FFF/FDM que imprime piezas mediante un flujo de deposición de plástico fundido y es el usado en este proyecto. Como ejemplo de los materiales que se pueden utilizar en este método encontramos el PLA o el ABS. Este material viene enrollado en una bobina que lo suministra a una boquilla que se calienta para fundir el material y comenzar la extrusión. La boquilla se orienta en los ejes XYZ por control numérico y es controlada por un SW de fabricación CAM. Una vez extruido, el material termoplástico se posa en una base con la que cuenta la impresora en la cual se endurece rápidamente. (Wikipedia, 2019)

Esta es la tecnología usada en el proyecto para construir el prototipo con piezas de diseño flexible. El material escogido es el PLA por las siguientes razones:

- Es biodegradable y no es tóxico.
- Funde a unos 185°C.
- No encoge al solidificarse.
- Alta velocidad de enfriamiento y solidificación.

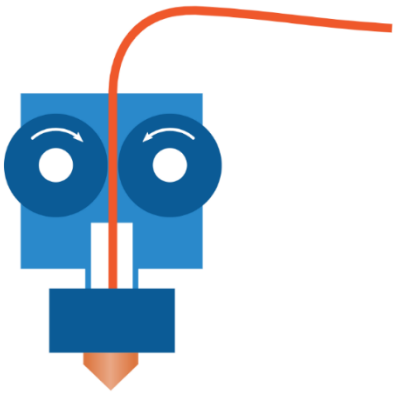


Figura 7.2. Extrusión de filamento en impresora 3D.

Fuente. Elaboración propia.

A la hora de diseñar la pieza se debe tener en cuenta que la pieza debe ser un sólido cerrado y a ser posible de geometría no muy complicada. Su tamaño no debe exceder la zona de impresión. En caso de necesitar imprimir ángulos mayores a 45º se hará uso de soportes en la estructura de la pieza para que tenga donde apoyarse el material fundido.

Cuando se hayan diseñado las piezas en 3D con un programa de CAD los archivos deben guardarse como archivo ".STL" para que se defina la pieza como una malla de triángulos cerrada. Este formato define la geometría de las piezas 3D sin tener en cuenta información como la textura o el color.

Tras obtener las piezas en formato ".STL" se utiliza un SW de laminado para tratar los archivos ".STL". Algunos de los programas de laminado más utilizados son los siguientes: 3DPrinterOS, Cura, Craftware y Astroprint. En este proyecto se utiliza el Cura Ultimaker. Lo que se consigue al laminar la pieza es descomponerla en varias capas perfectamente alineadas para conseguir el recorrido que debe seguir la boquilla al depositar el material fundido necesario. Cuanto menor sea la distancia entre capas mayor será la resolución de la pieza. En este paso también se puede observar cual es la mejor manera de colocar la pieza.

El lenguaje más común en control numérico es el G code para controlar por ordenador la posición y la velocidad de una máquina herramienta. Por ello, después del laminado de la pieza se obtiene un archivo ".gcode". Este archivo es el que necesita la impresora 3D para comenzar el proceso de impresión. En este caso el archivo es guardado en una memoria SD que se introduce en la impresora 3D. (Todo-3D, 2017)

7.3 CONTROLADOR PID

Uno de los controladores más extendidos para lazos de control es el controlador PID. Se compone de tres elementos que permiten acciones de control proporcional, integral y derivativo.

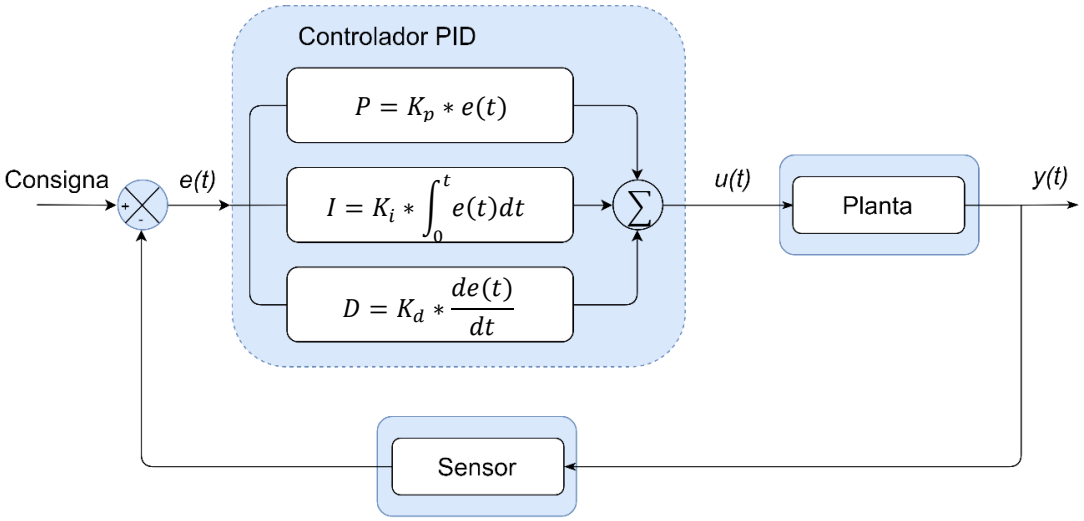


Figura 7.3. Diagrama de bloques de un sistema con PID.

Fuente. Elaboración propia.

A continuación, se explica brevemente en qué afecta cada una de ellas:

- **Control proporcional:** compara la consigna y el valor actual y actúa en consecuencia para intentar minimizar el error. Es el producto entre la constante proporcional y el error para conseguir que el error en estado estacionario se aproxime a cero. Kp es la constante proporcional sintonizable.

$$P = u(t) = K_p * e(t) \tag{7.1}$$

- **Control integral:** cuanto menor sea el error el ajuste correctivo será más rápido, pero hay que tener cuidado pues el sistema puede desestabilizarse, generar oscilaciones y/o vibraciones en el actuador. Puede que existan perturbaciones que provoquen un error en el régimen permanente que el control proporcional no pueda eliminar. El objetivo del integrador es reducir hasta eliminar ese error. Su modo de actuar consiste en integrar la diferencia entre la consigna y la variable en el tiempo para después sumarla a la acción proporcional. Se interpreta así que al ser un controlador que da una salida proporcional al error acumulado su actuación es lenta. Debido a que añade cierta inercia al sistema hay que tener cuidado al sintonizarlo

pues el error se acumula con un mal ajuste y hace el sistema más inestable.

$$I = u(t) = K_i * \int_0^t e(t)dt \quad (7.2)$$

- **Control derivativo:** su modo de actuar consiste en prever el error y corregir la tendencia. Es decir, da una respuesta con antelación a la velocidad del cambio del error y actúa antes de que el error se haga demasiado grande intentando frenar las inercias que causan sobre impulso u oscilaciones. Sólo actúa cuando el error absoluto varía, si el error es constante actuarán sobre él las acciones proporcional e integral. Su salida es proporcional a la diferencia entre el error anterior y el error actual. Su acción amplifica en el proceso las señales de ruido y puede saturar el actuador si está mal sintonizado.

  $$D = u(t) = K_d * \frac{de(t)}{dt} \quad (7.3)$$

  La derivada que se aprecia en la formula anterior es la velocidad que lleva la pelota en un instante determinado:

$$\frac{de(t)}{dt} = \frac{e_{actual} - e_{anterior}}{Ts} \quad (7.4)$$

Dependiendo de la aplicación, es posible implementar controladores P, PI, PD, I o PID ya que puede que no sean necesarios todos los parámetros. El más utilizado es el controlador PI ya que el D es muy susceptible al ruido.

7.4 MATLAB

Uno de los programas más populares para analizar y diseñar sistemas es MATLAB, abreviatura de laboratorio de matrices. Es un sistema de computación numérica que se expresa con un lenguaje M, basado en matrices, y que ofrece un IDE de desarrollo integrado. Su entorno gráfico logra que visualizar y obtener datos sea tarea fácil. (Mathworks., s.f.)

Es un programa capaz de analizar conjuntos masivos de datos, todo ello con código MATLAB que se puede implementar en otros lenguajes de

programación convirtiéndolo en uno de los programas más usados en la industria.

7.4.1 SIMULINK

MATLAB dispone de una toolbox llamada Simulink para simular el comportamiento y la tendencia de sistemas dinámicos. Es capaz de simular sistemas como los siguientes: lineales, no lineales, modelos de tiempo continuo, modelos de tiempo discreto y sistemas fruto de la mezcla de los anteriores. Este entorno gráfico permite que los modelos que se simulan sean construidos simplemente seleccionando y arrastrando una serie de bloques disponibles en la biblioteca de bloques de Simulink o "library browser". Los ficheros de Simulink se guardan como archivos MDL (modelos de simulación). (ETSETB, 2012)

Hay ciertos toolbox de MATLAB que incluyen bloques de Simulink como por ejemplo Neural Network Toolbox, System Identification Toolbox y Control Systems Toolbox. Además, algunos toolbox sirven para que las tarjetas de adquisición de datos o las placas interactúen con los bloques del modelo de Simulink. Gracias a esta última opción se puede utilizar en este proyecto la toolbox de Arduino para Simulink para comunicar la planta con el Arduino y Simulink.

7.4.2 FUNCIONES-S

Dentro del entorno de Simulink encontramos unos bloques llamados "s-function" que son programables. Gracias a estos es posible describir modelos dinámicos más complicados como por ejemplo sistemas no lineales o parámetros variables respecto al tiempo. Además de esto, también sirve para comunicar MATLAB con HW exterior para realizar una simulación "hardware-in-the-loop", es decir, la técnica usada para el desarrollo y comprobación de sistemas en tiempo real. Para realizar una "s-function" con conexión al HW es necesario programar en código C los comandos para la comunicación con placas, procesadores de señales digitales y tarjetas de adquisición de datos. El código utilizado debe ser compilado y unirse a MATLAB mediante archivos MEX. (Mathworks., s.f.)

El archivo MEX es un tipo de archivo ejecutable de MATLAB que crea una interfaz entre éste y las funciones en código C, C++ o Fortran programadas. Al compilar el programa los archivos MEX van cargándose dinámicamente para que MATLAB pueda llamar a las funciones externas como si estuvieran integradas en él. En este proyecto se utiliza un bloque “s-function” para programar el apartado de adquisición de datos del sensor.

8. DISEÑO

El modelo del sistema está separado en dos partes; la parte del hardware que está formada por toda la estructura, componentes y microcontrolador y la parte de software que contempla la programación y modos de ejecución.

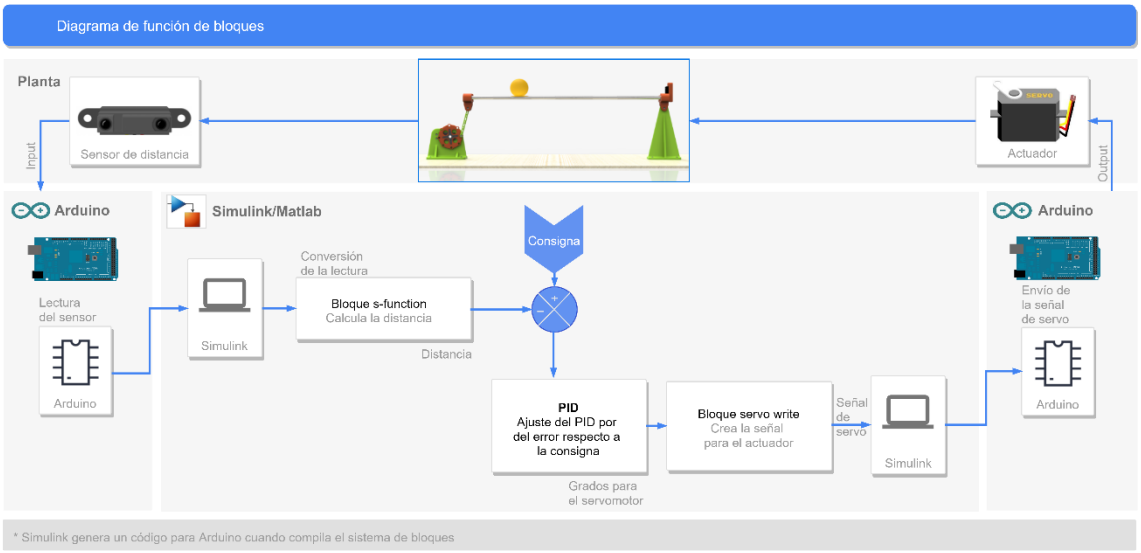


Figura 8.1. Diagrama general del sistema.

Fuente. Elaboración propia.

8.1 HARDWARE

8.1.1 CAPTURA DE DATOS

Para saber a qué distancia se encuentra la pelota y poder realizar un control sobre su posición es necesario que un sensor mida esa distancia. El Sharp escogido hace uso de la tecnología optoelectrónica que es capaz de medir la radiación infrarroja de los cuerpos que se encuentren en su campo de visión. Funciona con una Vcc entre 0.3V y 7V. Cabe destacar que su comportamiento es mejor al detectar objetos claros y tiene un rango mínimo (de unos 7 centímetros) que debe sobrepasar el objeto a detectar para que las medidas puedan considerarse válidas.

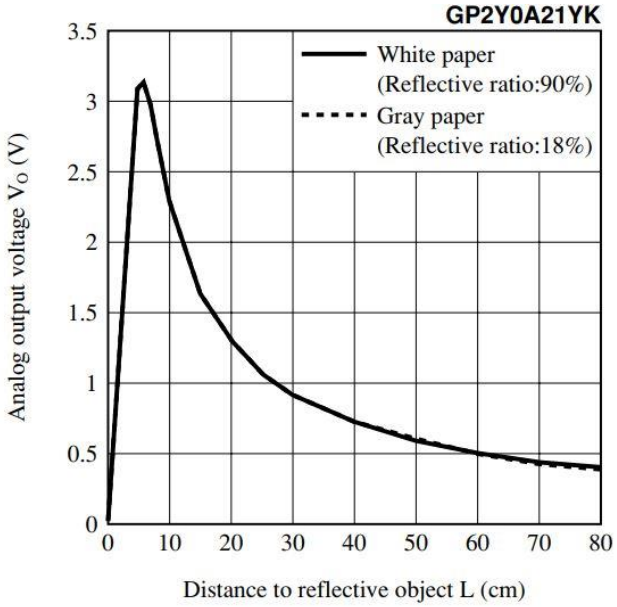


Figura 8.2. Curva característica aproximada del sensor Sharp GP2Y0A21YK.

Fuente. http://cort.as/-LEYW.

Para calcular la distancia en centímetros se debe definir una curva característica, en esta ocasión polinómica, en Excel siguiendo los pasos que se detallan a continuación:

- Realizar la conexión pertinente entre la placa Arduino y el sensor.
- Generar código que lea y muestre en pantalla la distancia obtenida por el sensor en voltaje.
- Colocar un objeto a distintas distancias y recoger los datos medidos por el sensor.
- Introducir las coordenadas (x, y) a modo de tabla.
- Crear un gráfico de dispersión.
- Introducir la tendencia polinómica y el grado del polinomio.
- Calcular los coeficientes del polinomio y su gráfica.

Datos medidos:

Tabla 8.1. Datos recogidos de la prueba de medición.

Voltaje (V)	Distancia (cm)
0.395	45
0.512	40
0.623	35
0.779	30
0.941	25
1.182	20
1.622	15
2.202	10
3	5

Fuente. Elaboración propia.

Dispersión distancia/voltaje:

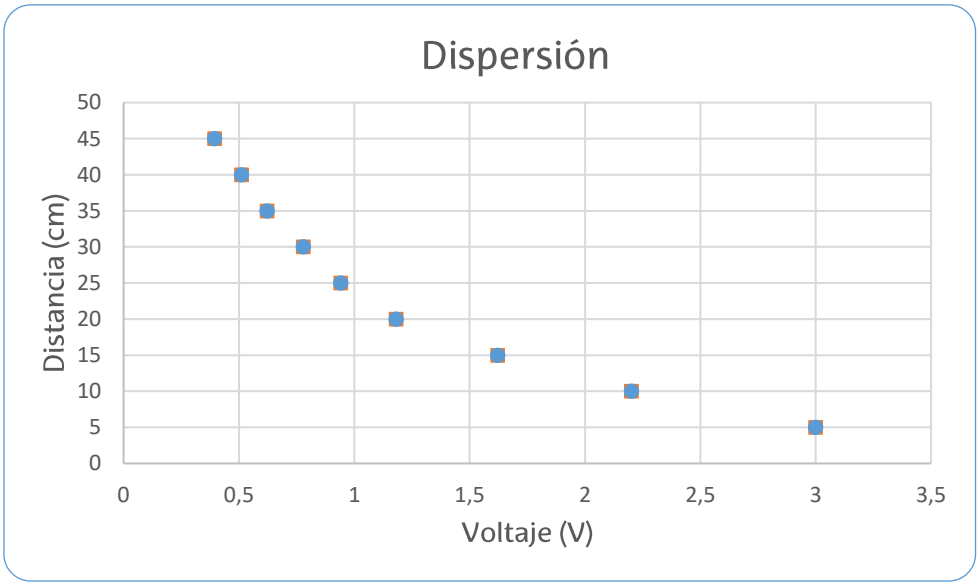


Figura 8.3. Gráfico de dispersión para distintas distancias.

Fuente. Elaboración propia.

Curva polinómica de orden 6 para definir los datos leídos por el sensor:

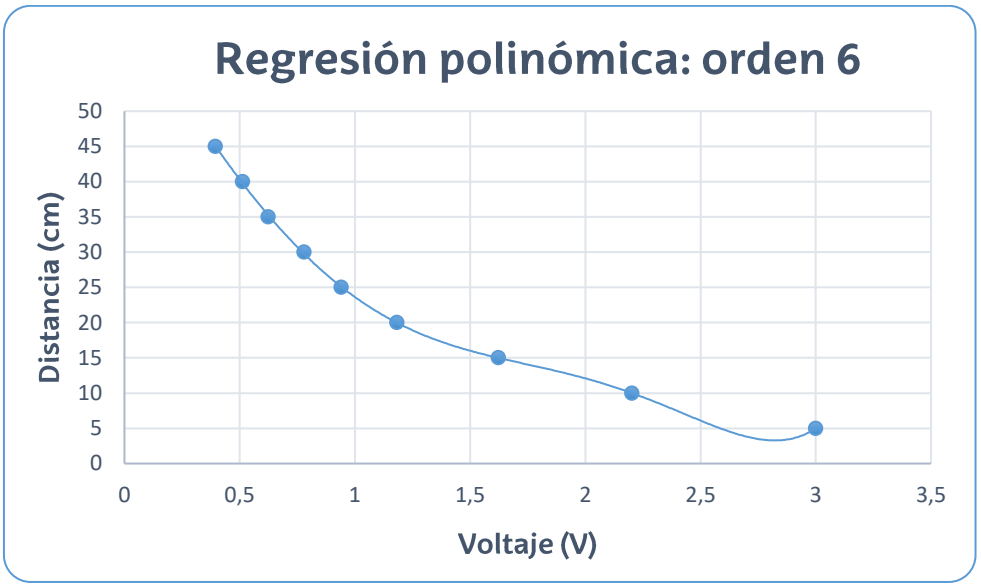


Figura 8.4. Regresión polinómica que relaciona la distancia y el voltaje medidos.

Fuente. Elaboración propia.

A la hora de definir la tendencia del gráfico se han tenido en cuenta varios tipos de aproximaciones tales como la potencial, logarítmica y polinómica. Para optimizar la elección se ha tenido en cuenta el coeficiente de determinación (R2) obtenido con cada aproximación. Este coeficiente da la proporción de variación de una variable respecto a una variable explicativa. Su valor comprende el rango entre 0 y 1 ambos incluidos y si llegara a ser igual a 1 significaría el modelo está perfectamente ajustado y no habría error. (López, J.F., 2019)

En este caso la regresión polinómica resulta ser la más adecuada cuando se trata de una ecuación polinómica de sexto orden ya que su coeficiente de determinación es de 0.9999 y cuanto mayor sea significa que mejor será la predicción y menor el error.

$$R^2 = \frac{\sum_{t=1}^{T}(\hat{Y}_t - \bar{Y})^2}{\sum_{t=1}^{T}(Y_t - \bar{Y})^2} = 0.9999 \quad (8.1)$$

$$y = 0.6945x6 - 3.7828x5 + 3.7846x4$$
$$+ 5.0294x3 + 8.859x2 - 56.587x + 65.642 \quad (8.2)$$

El sensor IR escogido tiene un rango de medición de 10 a 80 centímetros (0.4 a 3.1V) y en esta aplicación funcionará en un rango entre 5 y 45 centímetros. Al tratarse de un sensor analógico la señal no es muy fiable y oscila al ser susceptible al ruido así que para compensar esa falta de precisión se le ha añadido un buffer circular al código correspondiente a la captura de la señal del sensor para obtener la media de varias lecturas y compensar la medida.

Consta de un conector JST PH de 3 pines de conexión:

- Alimentación: Vcc (5V).
- Tierra: GND (0V).
- Vout: voltaje analógico de salida. Pin analógico 15.

Teniendo en cuenta que este sensor realiza una lectura en 52.9ms sin tener en cuenta el tiempo de conversión ADC, en un principio se estableció que el Ts fuera de 100ms, pero en la práctica la respuesta del PID era demasiado lenta con ese TS así que se ha establecido que el Ts del sistema sea mayor de 60ms, pero menor de 100ms.

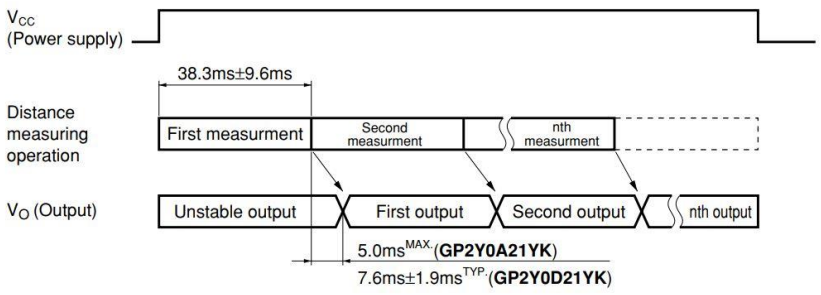


Figura 8.5. Tiempo de lectura del sensor Sharp mostrado en su datasheet.

Fuente. http://cort.as/-LEYW.

8.1.2 PROCESAMIENTO

Para poder procesar los datos leídos por el sensor analógico se hace uso de una placa Arduino Mega basada en un microcontrolador ATmega2560.

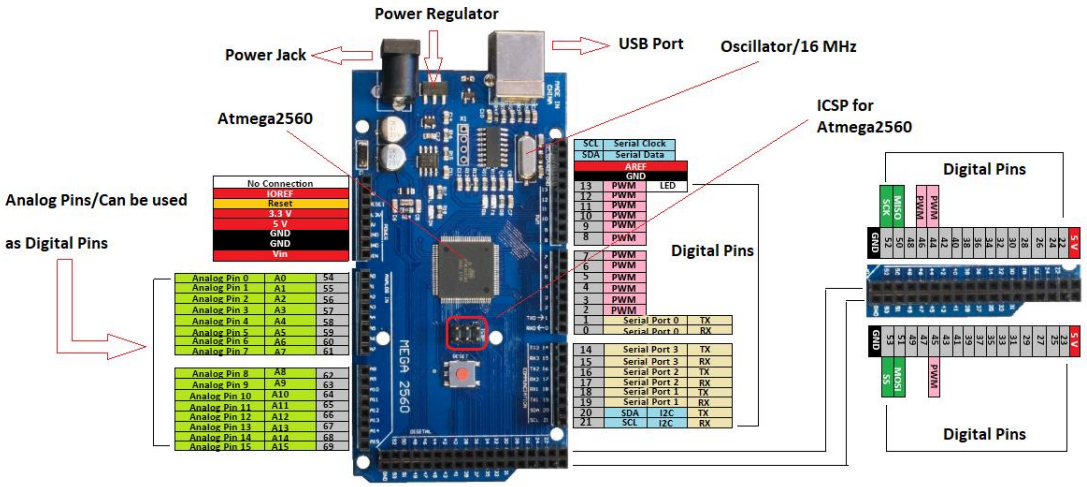


Figura 8.6. Pinout de la placa Arduino Mega.

Fuente. http://cort.as/-LGZn.

Esta placa cuenta con las siguientes especificaciones:

Tabla 8.2. Especificaciones de la placa Arduino Mega ADK

Especificaciones técnicas	
Microcontrolador	ATmega2560
Voltaje de operación	5V
Voltaje de entrada	9V
Límite de voltaje de entrada	7-18V
Pines I/O digitales	54 (14 para señal PWM)
Pines de entrada analógica	16
Corriente DC por pin I/O	40mA
Corriente DC por pin 3.3V	50mA
Memoria Flash	256KB
SRAM	8KB
EEPROM	4KB
Velocidad del reloj	16MHz

Fuente. Elaboración propia.

Los microcontroladores sólo entienden señales digitales (0 y 1) así que cuando reciben una señal por alguno de sus pines de entrada analógica necesitan convertir el valor analógico a digital mediante un convertidor ADC. La placa escogida tiene un ADC integrado que utiliza la tecnología TTL (0-5V) y tiene una resolución de 10 bits que logra unos valores entre 0 y 1023. Es posible cambiar la tensión de referencia que utiliza el ADC si el valor no supera los 5V. En esta ocasión se ha utilizado la referencia de 3.3V para que la resolución de los datos leídos por el sensor de distancia sea la mejor posible porque el sensor envía valores entre 0.4 y 3.1V. A continuación, se muestra la diferencia entre pasos, es decir, la precisión según la referencia escogida:

$$Paso\ 1 = \frac{5V}{1024} = 4.88mV \tag{8.3}$$

$$Paso\ 2 = \frac{3.3V}{1024} = 3.22mV \tag{8.4}$$

Este Arduino dispone de un solo ADC, pero no supone un problema ya que con un MUX selecciona de las 16 entradas analógicas la que va a convertir.

Para obtener los valores convertidos se aplica lo siguiente:

$$Vout = \frac{3.3V}{1024} * \text{lectura} \tag{8.5}$$

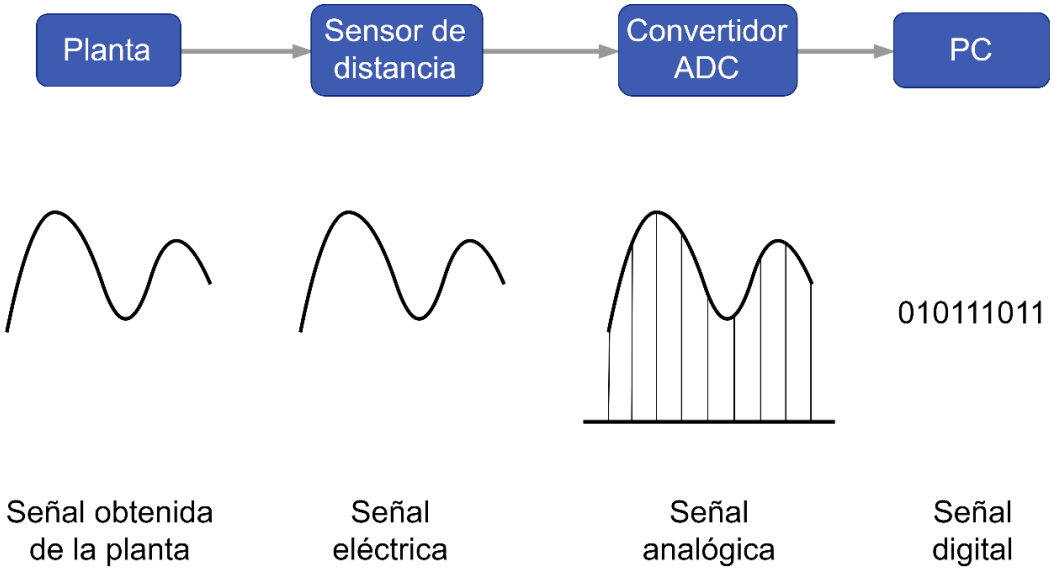


Figura 8.7. Conversión ADC.

Fuente. Elaboración propia.

Una señal analógica tiene una amplitud que en teoría alcanza cualquier valor dentro de cierto rango y por el contrario una señal digital representa valores discretos. Gracias a un conversor ADC la tensión leída puede ser convertida en una señal digital con un cuantificador y codificándolo a código binario. La digitalización comienza con el proceso de muestreo. En él se toman muestras de la señal analógica periódicamente y se retendrán. Una vez conseguida la señal muestreada se hará la cuantificación en la que se medirá el nivel de voltaje de cada muestra y a continuación se hará la codificación que sirve para convertir los valores cuantificados a código binario. Durante el sample & hold la señal todavía es analógica, pero a partir de la cuantificación ya es digital.

Como se puede observar, hay que tener en cuenta el tiempo de muestreo y es importante tratar de conseguir una buena resolución ajustada al rango de voltaje del sensor que se vaya a utilizar.

En la aplicación propuesta se realiza una conversión ADC de dos maneras, mediante programación utilizando el conversor integrado en la placa de Arduino y mediante una serie de bloques de Simulink.

Como se ha mencionado anteriormente, Arduino Mega es capaz de procesar una entrada analógica gracias al conversor ADC del que dispone, pero no es capaz de generar una señal analógica de salida. Para lograrlo hace uso de los pines de salida PWM. Así, con un único pin posibilita que se posicione el servo. Para el servo elegido se genera una señal PWM cuadrada periódica de 50HZ que varía el ancho de pulso.

A continuación se muestra el esquema de conexiones para el proyecto:

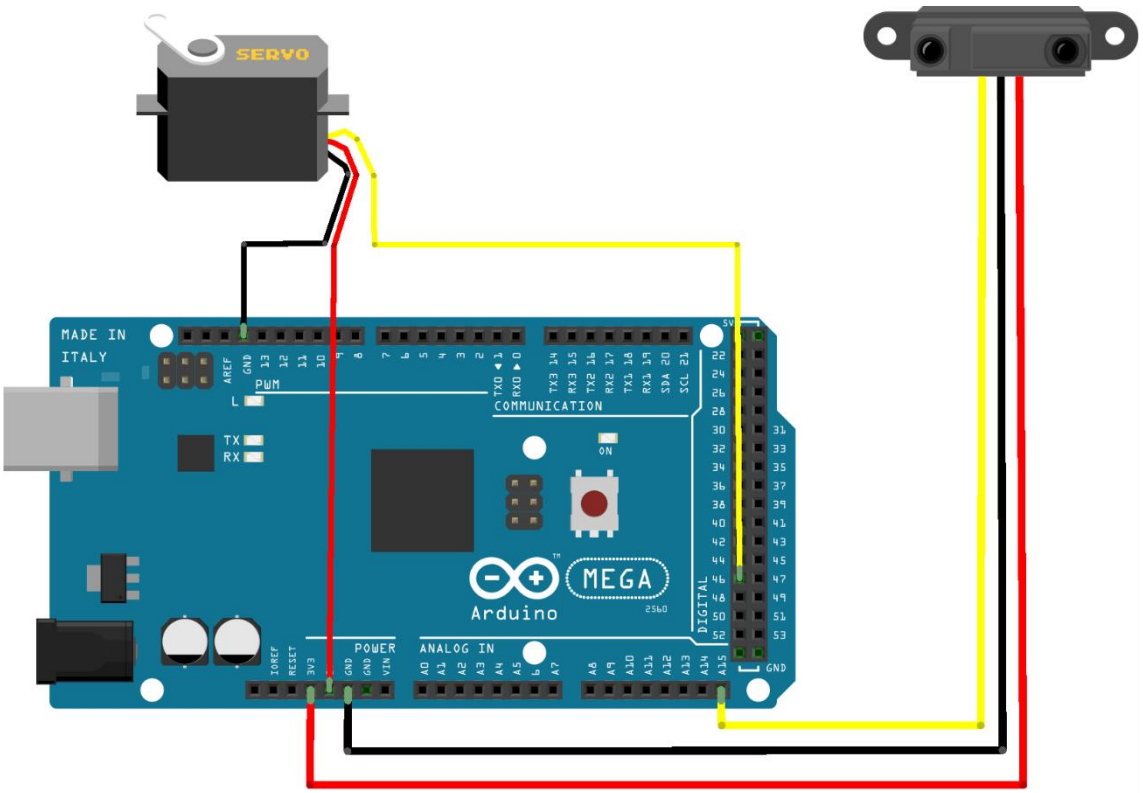


Figura 8.8. Esquema de conexiones.

Fuente. Elaboración propia.

8.1.3 CREACIÓN DE LA SEÑAL DE SALIDA

Un servomotor es un tipo especial de motor c.c. que, dentro de su intervalo de operación, es capaz de cambiar de posición de forma inmediata. Cuando recibe una señal de servo realiza un movimiento acorde a esa señal. Comúnmente es llamada señal PWM, pero esta afirmación no es completamente correcta ya que un servo puede resultar dañado si recibe una señal PWM real. (R. González, V., 2002-03)

Una señal de servo es una señal de pulso variable seguida por una pausa de alrededor de 10ms o 20ms. La anchura del pulso determina la posición del servo. (Sourceforge, 2019)

Para posicionar el servo elegido se genera una señal PWM a través del pin digital 46 de Arduino Mega tras recibir el ángulo de salida obtenido del controlador PID. La señal es cuadrada con periodo de 20ms que varía su

ancho de pulso para controlar su movimiento. En este caso el servo opera en un rango de 0 a 170 grados. Se envía por la línea de control del Arduino al servo para comunicarle en qué ángulo se debe posicionar. (Wikipedia, 2019)

La frecuencia y la amplitud del pulso son constantes y sólo varía el ancho de pulso de 0.545ms a 2.4ms como se puede observar en la siguiente imagen:

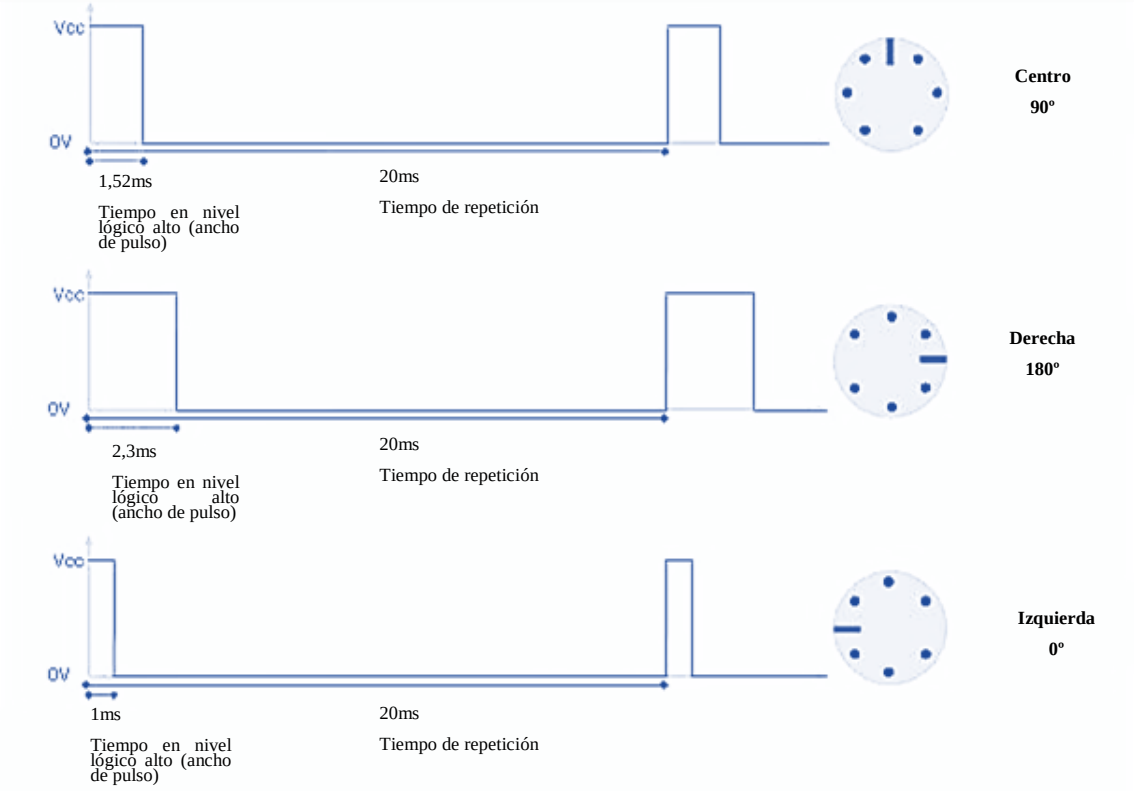


Figura 8.9. Modulación de pulso variable para el servomotor Futaba S3003.

Fuente. http://cort.as/-LKlg.

En este caso el servo se compone de un amplificador, un sistema reductor y un motor DC. El sistema reductor son unos engranajes de plástico que lo hacen capaz de soportar hasta 4kg por centímetro con el servo alimentado a 5V. Su rango de operación angular es de casi 180 grados. Para posicionar este servo la frecuencia de la señal debe ser de 50Hz (20ms).

Consta de un conector con 3 cables de conexión:

- Alimentación: Vcc (5V).
- Tierra: GND(0V).
- Señal de control: recibe la señal PWM para posicionar el servo.

8.1.4 IMPRESIÓN DE ESTRUCTURA EN 3D

En la universidad se dispone de acceso a una impresora 3d Prusa i3 y con ella se han realizado las piezas de la estructura necesarias para llevar a cabo el proyecto. El diseño de las piezas de la estructura del "Ball and Beam" se ha realizado con Autodesk Inventor, programa de diseño asistido por ordenador (CAD). La elección de Inventor por delante de otros más conocidos (Solidworks, SoliEdge, CATIA) se debe a que en la universidad se dispone de la licencia para su uso. Se ha utilizado para diseñar en 3D las piezas, realizar el montaje virtual de la maqueta y acotar los planos en DIN A4 horizontal. Una vez obtenidas las piezas en formato ".STL" se han abierto en el programa Cura Ultimaker para su laminado y así crear archivos "g code". Ese archivo se pasa a una SD y la SD a la impresora para comenzar la impresión.

A continuación, se aprecia un breve resumen del proceso de impresión:

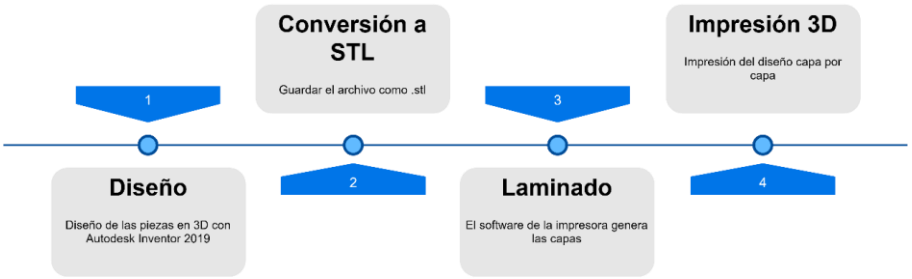


Figura 8.10. Proceso de impresión 3D.

Fuente. Elaboración propia.

Las seis piezas diseñadas son las siguientes:

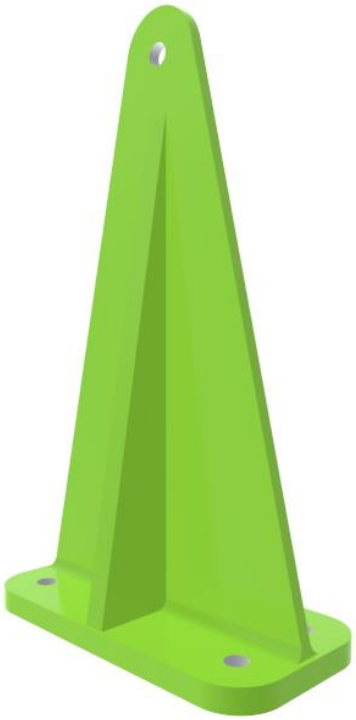


Figura 8.11. Base del extremo del sensor.

Fuente. Elaboración propia.

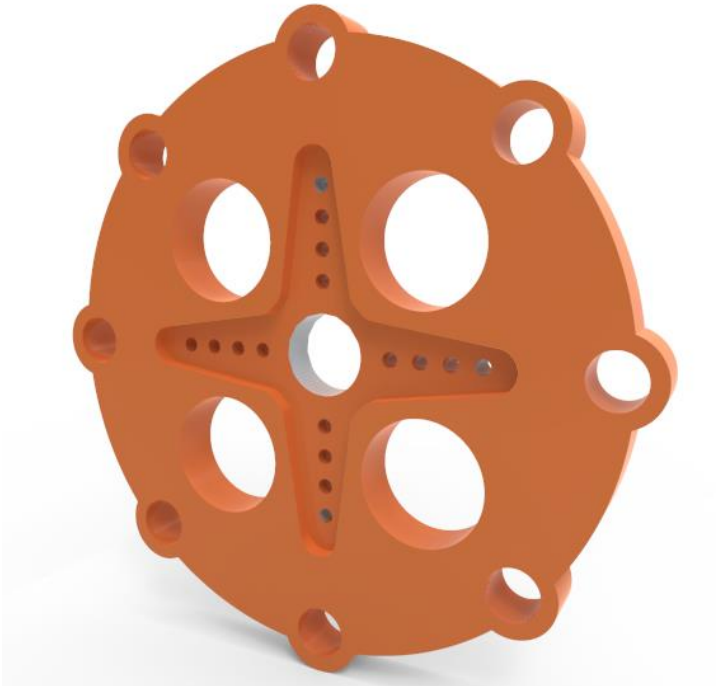


Figura 8.12. Disco ranurado.

Fuente. Elaboración propia.

Figura 8.13. Base del servomotor.

Fuente. Elaboración propia.

Figura 8.14. Soporte del sensor.

Fuente. Elaboración propia.

Figura 8.15. Manivela.

Fuente. Elaboración propia.

Figura 8.16. Soporte.

Fuente. Elaboración propia.

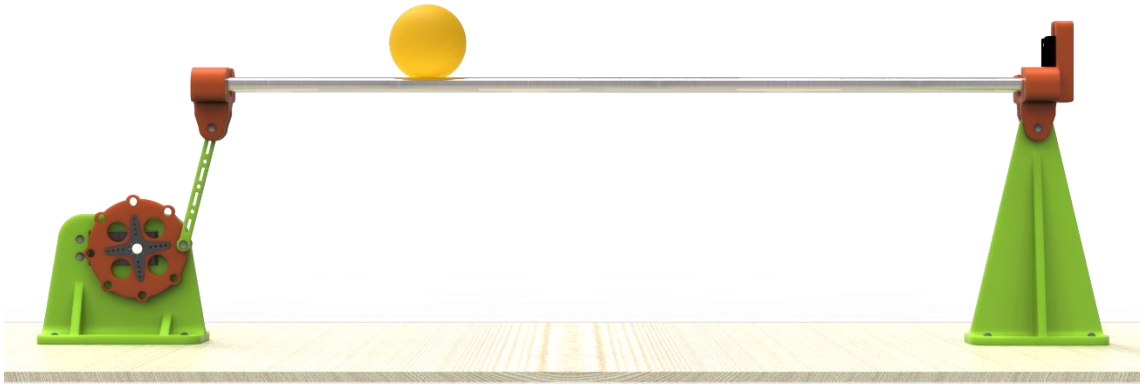


Figura 8.17. Vista frontal del diseño en 3D.

Fuente. Elaboración propia.

La maqueta montada es la siguiente:

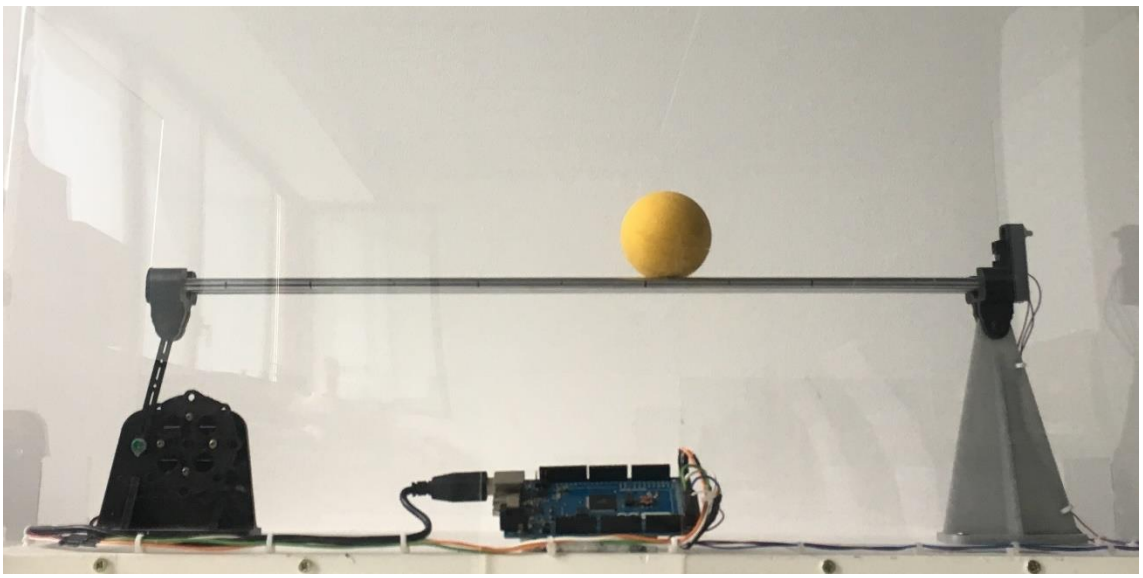


Figura 8.18. Prototipo montado.

Fuente. Elaboración propia.

8.2 SOFTWARE

En este apartado se analizan las diferencias entre hacer uso de un bloque s-function y bloques predefinidos de Arduino. El funcionamiento general del sistema se explica detalladamente en los siguientes apartados, pero en la siguiente figura se puede ver un diagrama general del sistema.

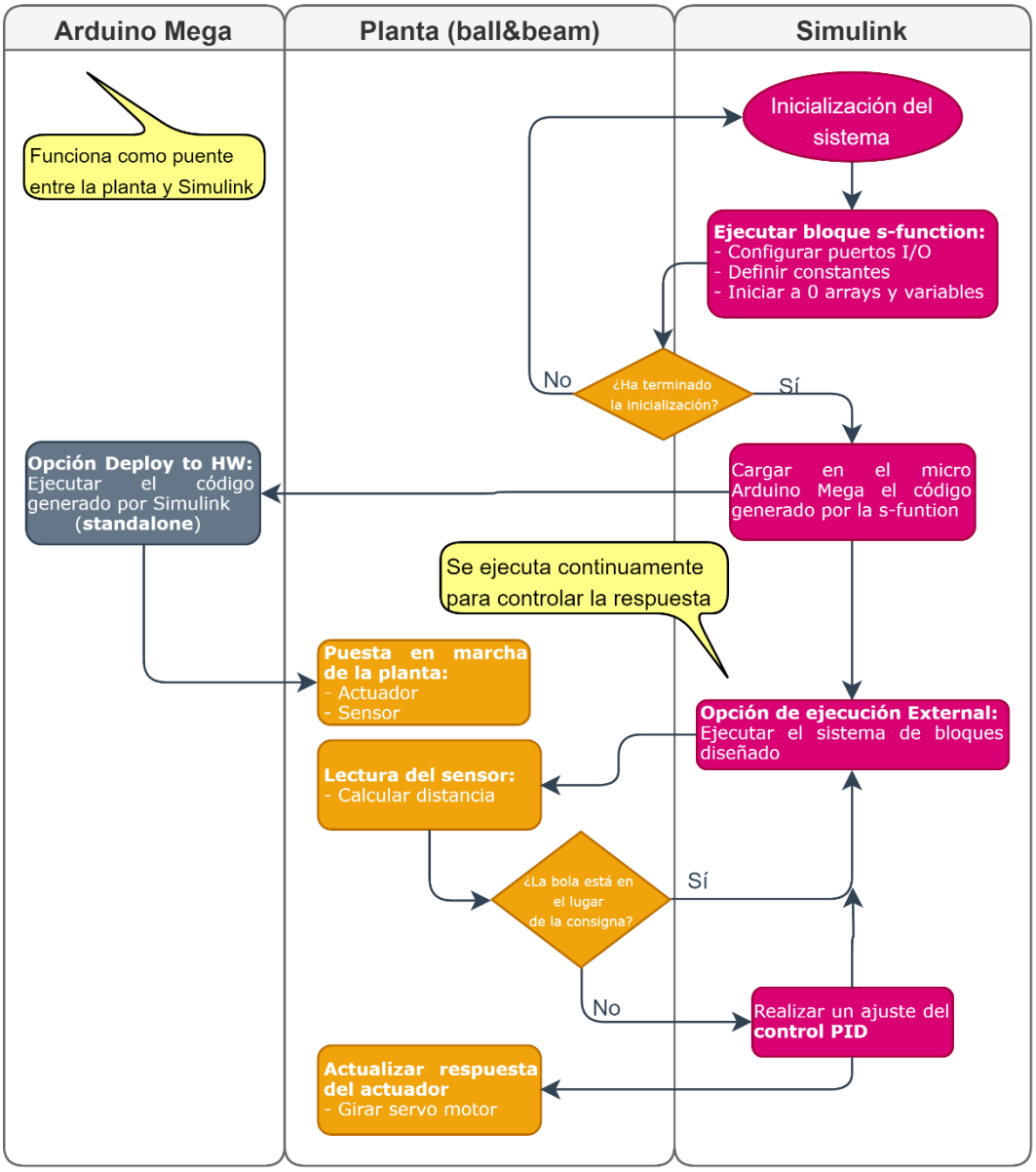


Figura 8.19. Diagrama general del sistema.

Fuente. Elaboración propia.

8.2.1 ARDUINO

En este apartado se explica cómo sería el desarrollo total del código programado en Arduino IDE para el control de la planta para comprender su complejidad. El puerto USB que se utiliza para la conexión de la placa al PC como se puede ver en la figura 8.20, se hace a través del puerto serie 0 de Arduino de tal manera que el PC pueda interactuarcon Arduino y se visualicen los datos en el Monitor serie del Arduino IDE. Por lo tanto, ese puerto no se puede usar a la vez que algún otro periférico. En caso de

Arduino Mega, el cual posee más de un puerto serie (4), puede configurarse para utilizar otros periféricos a la vez que se comunica con el PC.

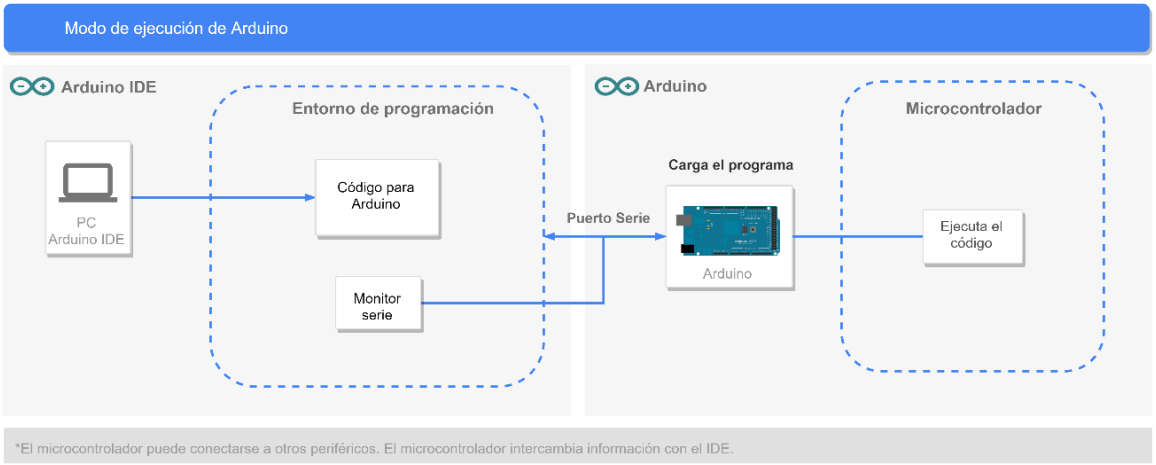


Figura 8.20. Modo de ejecución: Arduino-Arduino IDE.

Fuente. Elaboración propia.

A la hora de programar el microcontrolador junto con el sensor Sharp se ha visto que este último es bastante susceptible al ruido y que por tanto las mediciones obtenidas no eran del todo fiables. Para solucionar este problema y obtener una buena lectura se han implementado recursos como un buffer circular y un filtro paso bajo. De este modo se ha conseguido una lectura más estable y limpia.

El buffer circular utiliza una estructura FIFO que toma las diez últimas distancias leídas por el sensor y calcula su media. Una vez inicializado el array con las diez lecturas del sensor de distancia, se hace la suma y se divide entre el número de muestras. Una vez guardado ese valor se le resta la muestra más antigua desplazando el array y se le añade un nuevo valor muestreado. Así, en vez de esperar a rellenar un array de 10 muestras se va actualizando la media únicamente restando un valor y sumando otro para recalcular la media. Es decir, se ha reducido el proceso de sacar la media a 3 operaciones simples ahorrando tiempo. Como el sensor necesita unos 60ms para realizar una lectura, si se quiere hacer la media de diez lecturas se necesitarían 600ms, resultando en un método incompatible con nuestro Ts elegido ya que no le daría tiempo en 60ms a realizar dicha media.

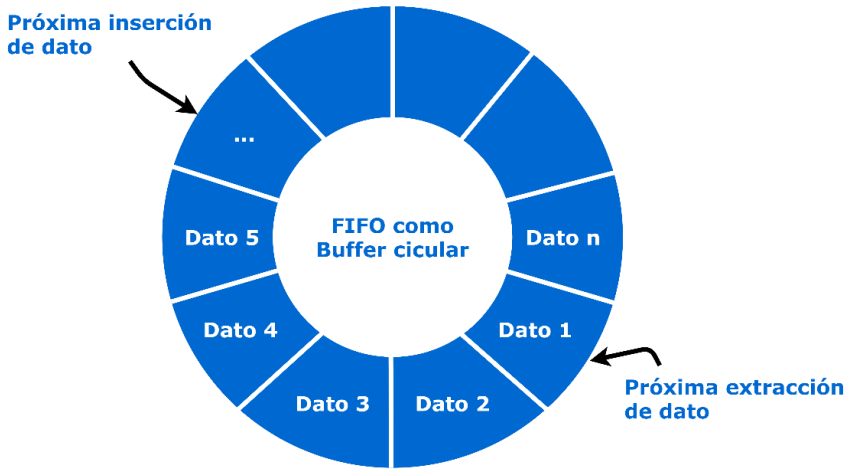


Figura 8.21. Funcionamiento de un buffer circular.

Fuente. Elaboración propia.

Este recurso es una implementación eficiente en código C que permite una lectura casi continua.

El otro recurso utilizado es el filtro paso bajo. Es un filtro electrónico que deja pasar señales de baja frecuencia y bloquea señales de alta frecuencia. De esta manera se obtiene una lectura con menos ruido.

La lectura obtenida es una señal analógica y para que el microcontrolador pueda procesarla primero realiza una conversión ADC. Para obtener la mayor resolución posible en esta conversión se ha programado mediante el comando analogReference() tomar como referencia el voltaje de 3.3V en lugar del de 5V, ya que el sensor trabaja en un rango de hasta 3.1V.

Una vez realizada la conversión de la lectura se debe convertir a la unidad deseada, en este caso a centímetros. Para ello se hace el cálculo de la curva de regresión polinómica que mejor se ajuste al modelo del sensor y se programa la ecuación obtenida para almacenar en una variable la distancia en centímetros.

El siguiente paso es comparar la distancia leída con la consigna preestablecida y enviar el error resultante al controlador PID diseñado para que se genere una señal de control para el servomotor que sea capaz de orientar la barra y corregir la posición de la pelota.

Como Arduino es un microcontrolador discreto se deben parametrizar las ecuaciones del controlador PID continuo para poder trabajar con uno discreto.

El PID diseñado se basa en una aproximación trapezoidal (Tustin). Existen otras aproximaciones tales como Euler hacia adelante y Euler hacia atrás. Como puede observarse en los gráficos siguientes esas aproximaciones cogen área de más y de menos y por ello se elige la opción que mejor se ajusta al gráfico y menor error tiene: Tustin o bilineal. (Control class, 2018)

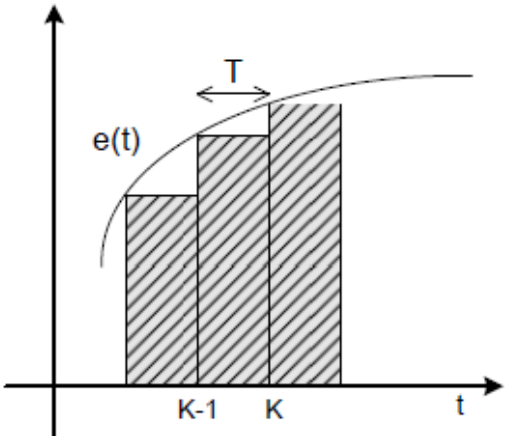


Figura 8.22. Aproximación de Euler hacia adelante.

Fuente. http://cort.as/-LUre.

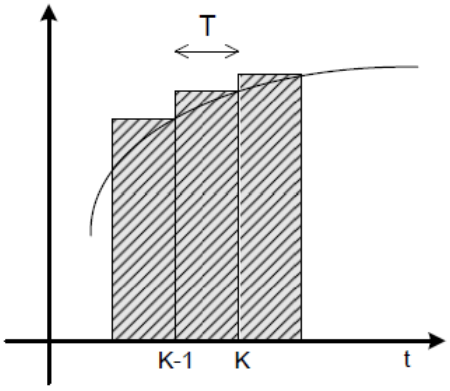


Figura 8.23. Aproximación de Euler hacia atrás.

Fuente. http://cort.as/-LUre.

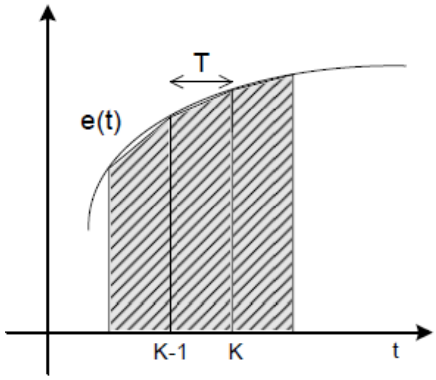


Figura 8.24. Aproximación trapezoidal.

Fuente. http://cort.as/-LUre.

La discretización sigue el siguiente algoritmo:

$$u(t) = k_p * e(t) + k_i * \int_t^{t+1} e(t)dt + k_d * \frac{de(t)}{dt} \begin{cases} e(t) \rightarrow e[m] \\ \int_t^{t+1} e(t)dt \rightarrow \sum_m (e[m] * Ts) \\ \frac{de(t)}{dt} \rightarrow \frac{e[m]-e[m-1]}{Ts} \end{cases} \quad (8.6)$$

$$u(m) = k_p * e(m) + k_i * \sum_m (e[m] * Ts) + k_d * \frac{e[m]-e[m-1]}{Ts} \quad (8.7)$$

La señal de control generada por el controlador PID es la codificación del ángulo que debe alcanzar el servomotor para la corrección de la posición de la bola.

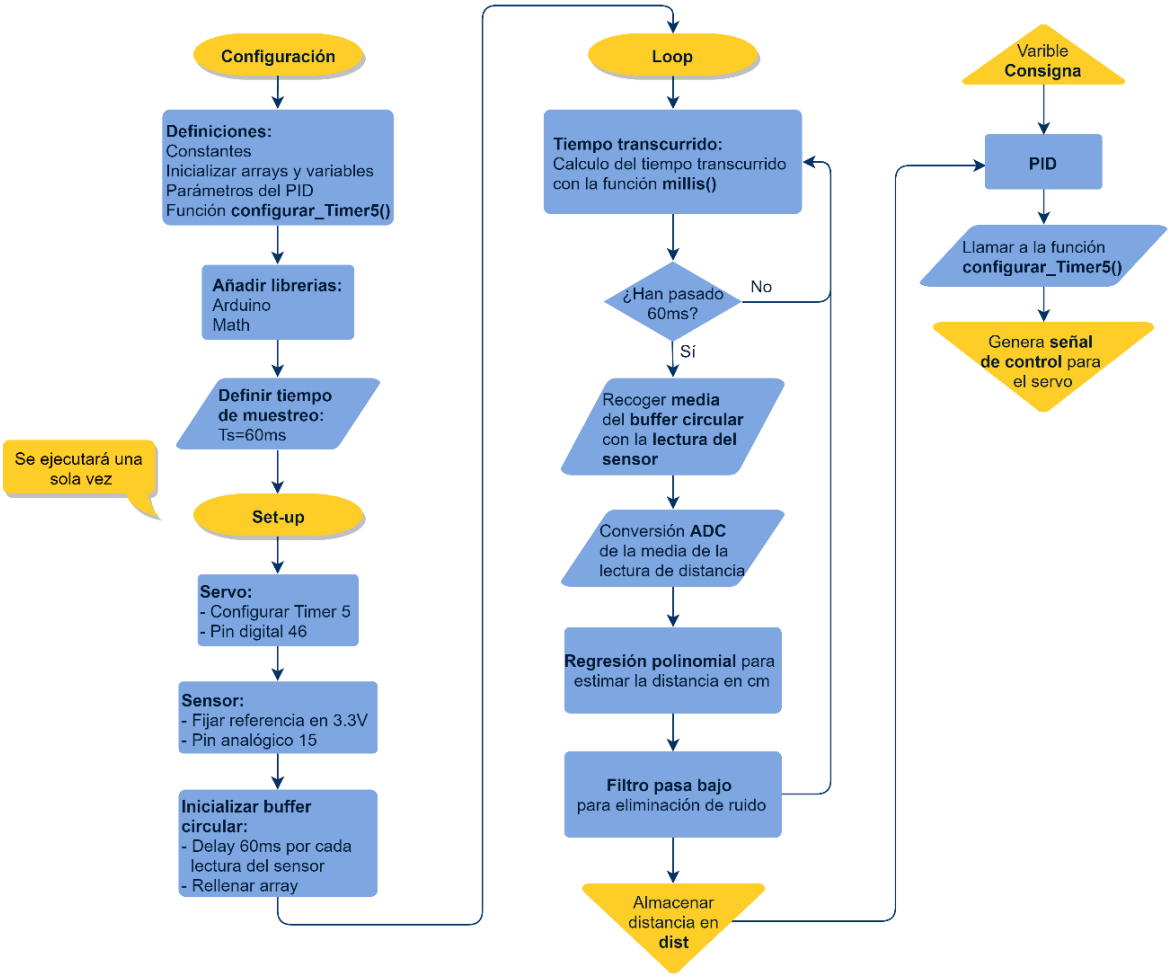


Figura 8.25. Descripción algorítmica del programa completo de Arduino. Fuente. Elaboración propia.

Como puede observarse la programación completa en C de un control PID para un sistema ball and beam se vuelve compleja y requiere de varios intentos para sintonizar los parámetros adecuados del PID. Por ello se fijó el objetivo de utilizar la parte esencial de este código e implementarla en Simulink para facilitar el proceso de sintonización del PID. Esa parte permite un mayor control sobre la lectura del sensor como puede verse en la siguiente figura:

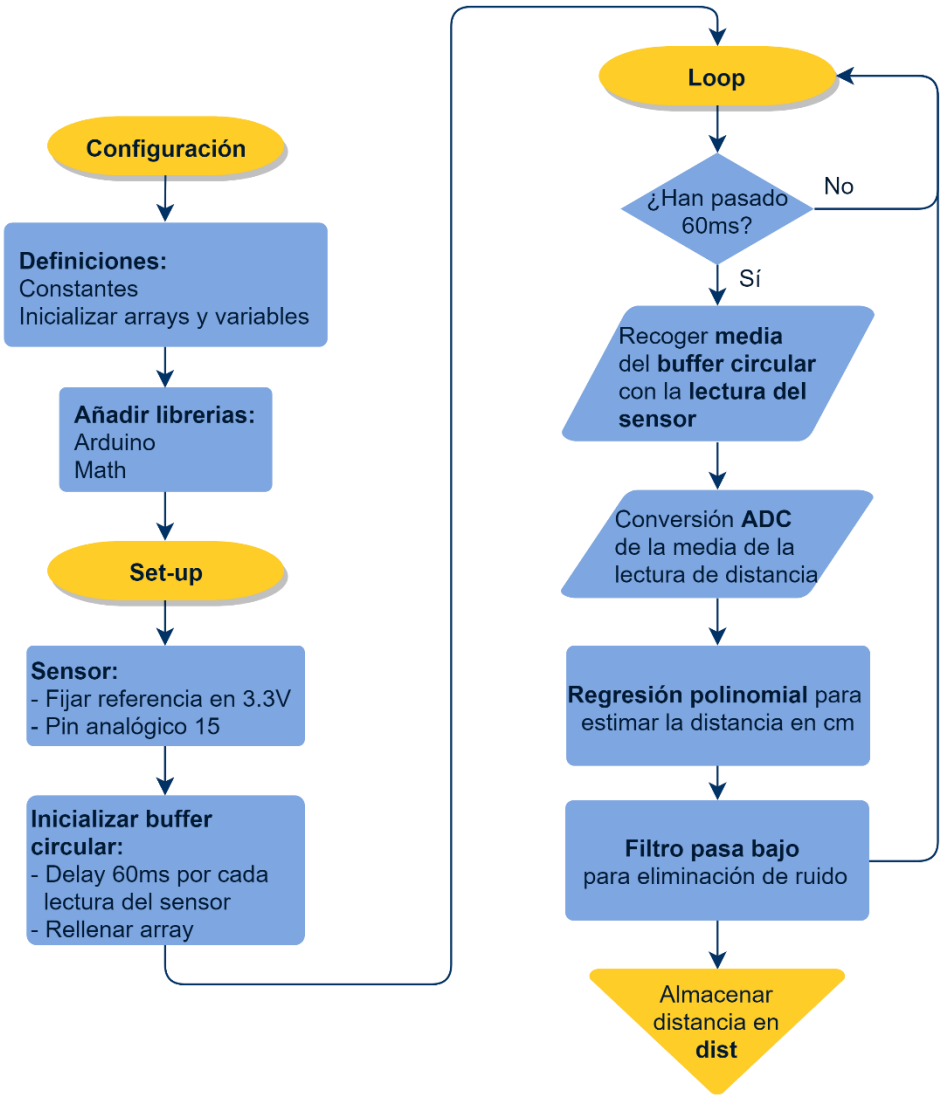


Figura 8.26. Descripción algorítmica de la parte programada de Arduino. Fuente. Elaboración propia.

8.2.2 MATLAB/SIMULINK

Este apartado contempla la conversión de la programación anterior para compatibilizarla con Simulink y el diseño del modelo de bloques. La descripción del desarrollo se ha dividido en cinco apartados, pero para una visión general se describen brevemente los pasos seguidos:

- Se ha instalado la toolbox de Arduino en MATLAB.
- Se han diseñado dos modelos; el primero consta de bloques predefinidos compatibles con Arduino y el segundo implementa el código C del sensor que le aporta flexibilidad al diseño, aunque hay que configurar Simulink para poder ejecutarlo.

- Pruebas con la placa y el sensor en conjunto con Simulink para comprobar el funcionamiento.
- Pruebas con la placa, el sensor y el actuador mapeado en conjunto con Simulink para comprobar el funcionamiento.
- Fijar el Ts del sistema adecuado.
- Sintonizar el bloque PID discreto de manera manual sin entrar en el campo de identificación del sistema.
- Pruebas con el modelo completo con Arduino en conjunto con Simulink para comprobar el funcionamiento de manera interactiva.
- Diferenciar los dos modos de ejecución disponibles: deploy to HW y external.

8.2.2.1 TOOLBOX DE ARDUINO

Para poder utilizar Arduino en conjunto con Matlab/Simulink el primer paso es instalar el add-on "toolbox de Arduino". Este add-on incluye bloques para la programación mediante Simulink de diferentes placas de Arduino. Al utilizar juntas estas plataformas se encuentran dos opciones a la hora de ejecutar los modelos: deploy to hardware y external mode.

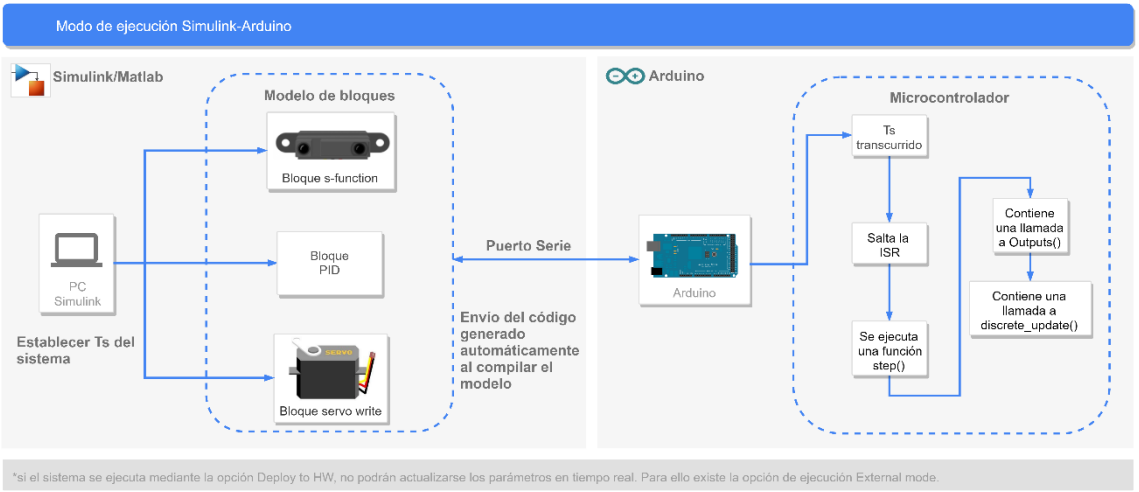


Figura 8.27. Modos de ejecución en conjunto: deploy to HW vs External. Fuente. Elaboración propia.

Trabajar con este soporte de MATLAB para Arduino posibilita dos modos de trabajo:

- La lectura, escritura y análisis de datos recogidos por los sensores de Arduino de forma interactiva en "external mode".
- El desarrollo de algoritmos que se pueden ejecutar de forma independiente en la placa mediante la opción "deploy to HW"

La primera opción permite diseñar un diagrama de bloques para leer y escribir de manera sencilla y sin necesidad de esperar a la compilación del código. Aunque el resultado se obtiene de forma más rápida no se puede cortar la conexión entre la placa y el equipo ya que el procesamiento se lleva a cabo en el Arduino mientras que Simulink permite simular y ajustar los parámetros hasta obtener el resultado esperado.

Figura 8.28. Ejecutar en external mode manteniendo la conexión.

Fuente. Elaboración propia.

En cambio, en la segunda opción se ejecuta el modelo de bloques diseñado y genera código automáticamente para descargarlo y ejecutarlo de forma autónoma en la placa Arduino. De esta manera el procesamiento se produce en Arduino y no es necesario mantener la conexión USB de la placa al PC. (Mathworks, s.f.)

En ambos casos el funcionamiento es el mostrado en la Figura 8.27; se ejecuta el modelo en Simulink y se genera código para Arduino. El código llama a una función step() en Arduino haciendo saltar una ISR cada Ts. Esa función contiene lo siguiente: una llamada a Outputs()y una llamada a Update() (códigos programados en las pestañas de la s-function).

8.2.2.2 S-FUNCTION BUILDER DEL MODELO

Como se ha explicado en el apartado 7, la posibilidad de adaptar código C a un bloque de Simulink permite sacar mayor partido y tener mucho más control sobre la lectura del sensor. Esto permite utilizar todo el poder computacional del código C en conjunto con Simulink haciendo una lectura mucho más eficiente.

Mediante los bloques de "s-function builder" se puede adaptar el código C programado en Arduino IDE para incorporarlo al modelo creado en Simulink. Para poder hacerlo, se deben emular los estados loop y set-up del código ya que no es posible para MATLAB procesar el código C sin modificarlo. A lo largo de este apartado se explica cómo es posible adaptar el código detallando todo el proceso.

El comportamiento de estos bloques es como el de una máquina de estados. En outputs se ejecuta el estado actual y en Update se evalúa el siguiente estado. En la programación desarrollada se consigue que cada Ts se ejecute Update pero sin producir ningún cambio de estado, es decir, se ejecuta siempre pero sólo realiza un cambio de estado tras la primera ejecución para que se mantenga en ejecución un único estado corresponidiente a lo programado en Outputs.

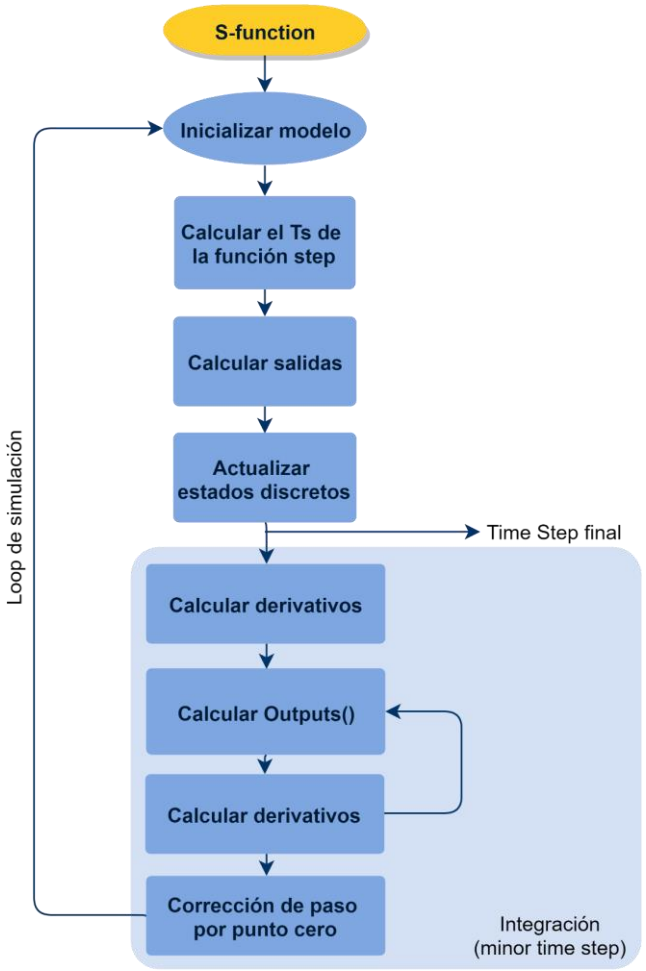


Figura 8.29. Modo de ejecución genérico de una s-function.

Fuente. Elaboración propia.

Por ello cuando se diseña el bloque "s-function builder" es necesario emular los estados de Arduino IDE set-up y loop. Para ello asignaremos dos estados discretos dentro del bloque:

- Estado discreto 0: emula la inicialización del sistema y se debe ejecutar una sola vez.
- Estado discreto 1: emula el bucle que se ejecutará cada Ts.

Para ello el bloque debe configurarse de la manera explicada a continuación. Lo primero es entrar en Simulink y elegir el bloque "s-function builder". En él, dentro de la pestaña "initialization" se señaliza el número de estados discretos; el estado uno será el loop y el cero será el setup de Arduino. Una vez elegido el Ts (en este caso 60ms) se inicializará la actualización del estado (pestaña discrete update). Jugando con estas opciones, se asigna el código C del set-up al estado 0 y una vez ejecutado éste se pasará a ejecutar el estado 1 consiguiendo así no volver a ejecutar la inicialización. Es decir, una vez pasado el Ts asignado, entrará a actualizar el estado, pero como estarás en el 1 no ejecutará la inicialización y pasará a actualizar las salidas.

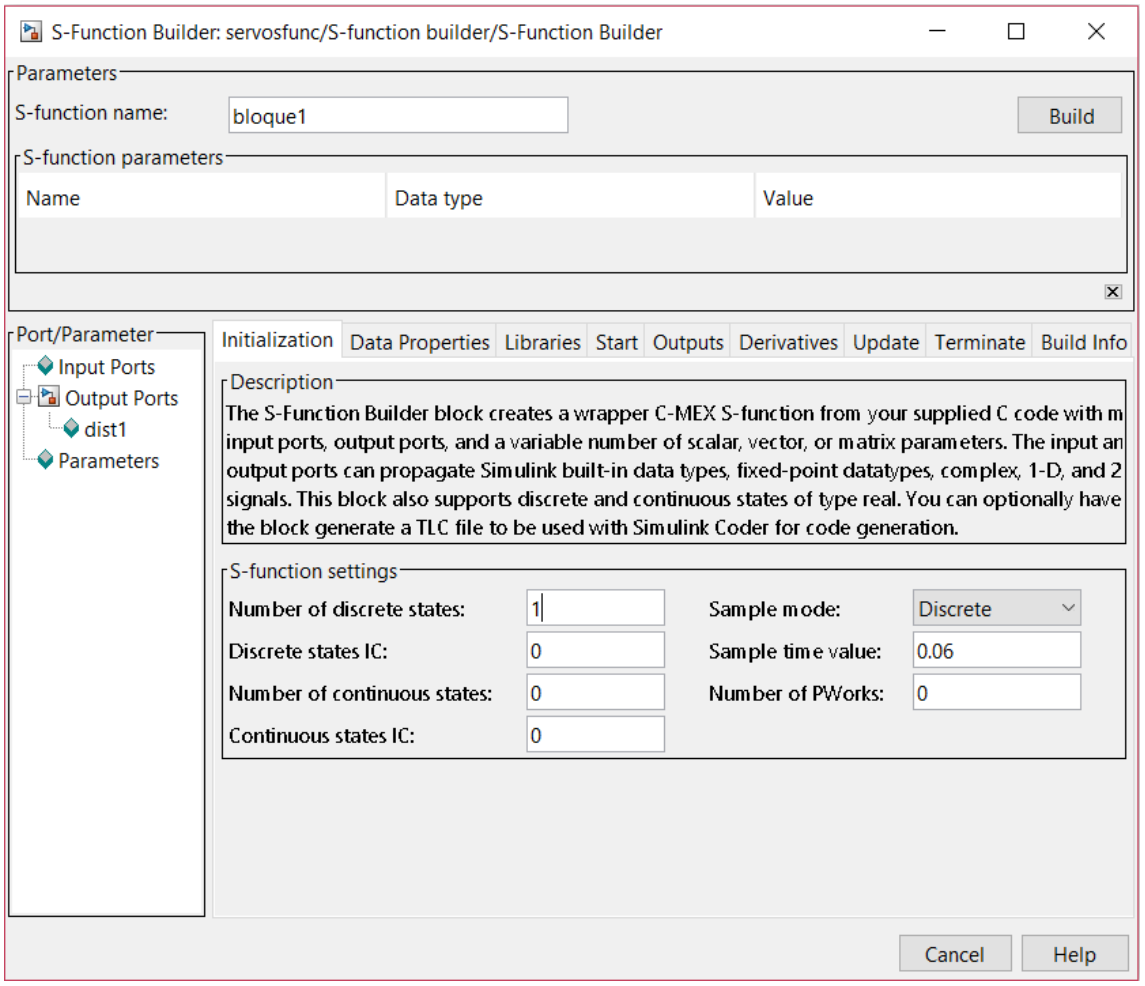


Figura 8.30. Pestaña Inicialization del bloque s-function.

Fuente. Elaboración propia.

Después se deben incluir los headers servo.h y math.h utilizados en Arduino en la pestaña "Libraries". Se copiará allí el encabezado del código, pero MATLAB no lo entenderá si se hace directamente. Al copiarlo se deben hacer ligeras modificaciones al código añadiendo #ifndef MATLAB_MEX_FILE para señalarle a MATLAB que va a ejecutar código C. Para ello el programa creará una interfaz entre MATLAB y las funciones en C. Al utilizar ese ejecutable se debe señalar cuándo termina dicho archivo mediante el comando #endif.

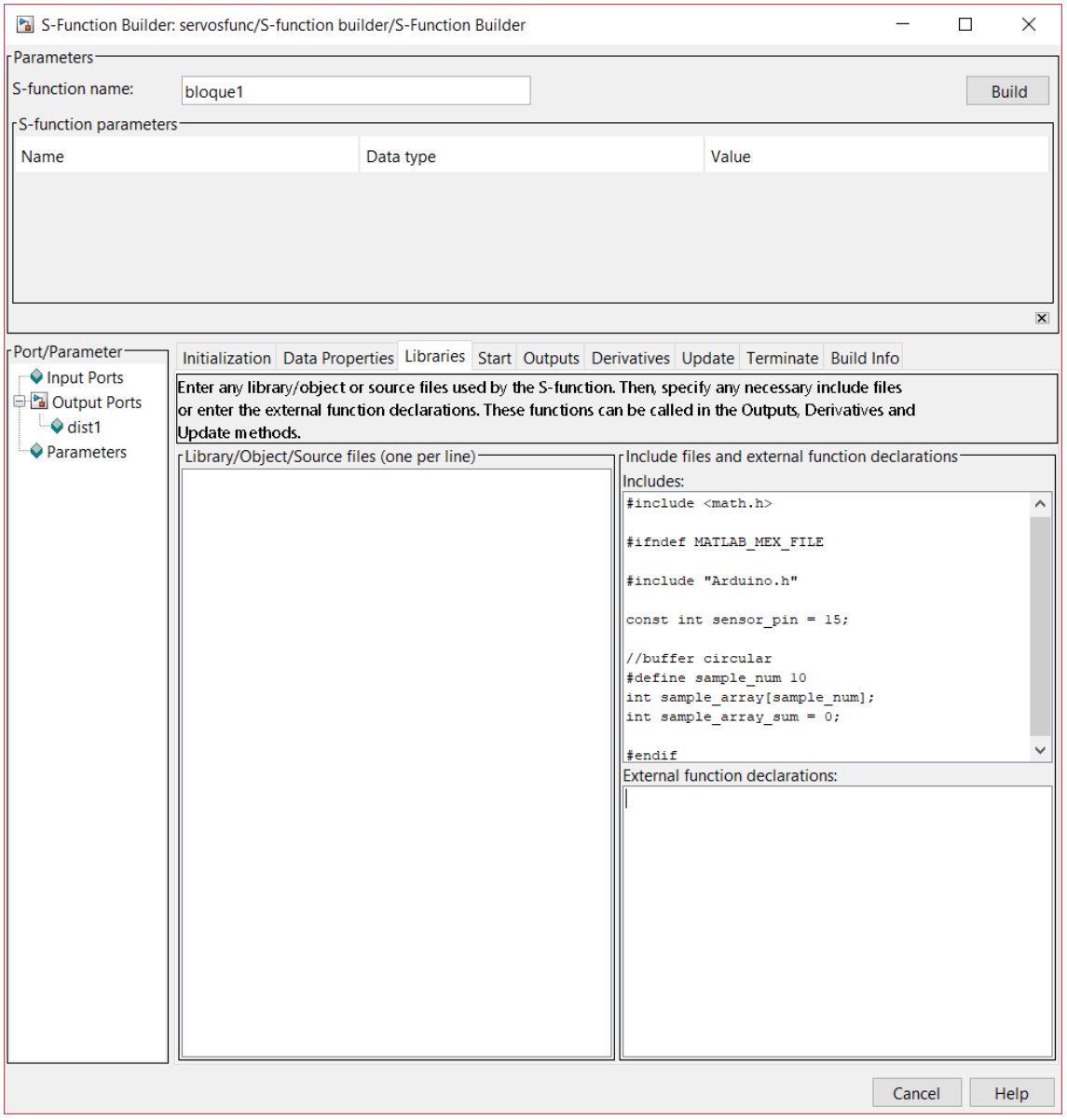


Figura 8.31. Pestaña Libraries del bloque s-function.

Fuente. Elaboración propia.

Tras esto, en la pestaña "Outputs" se debe adaptar ligeramente el código de Arduino para convertirlo en un código que MATLAB entienda. En este estado se emula el loop o estado uno de Arduino, ya que el estado cero es asociado al set-up de Arduino en la pestaña "Update".

En esta pestaña de actualización de las salidas se indica que al ser igual a uno el estado, se debe ejecutar el código C que corresponde al loop excepto la parte de inicializar un array, eso debe hacerse en el estado 0 junto con el código de inicialización. Se explicará más adelante el motivo.

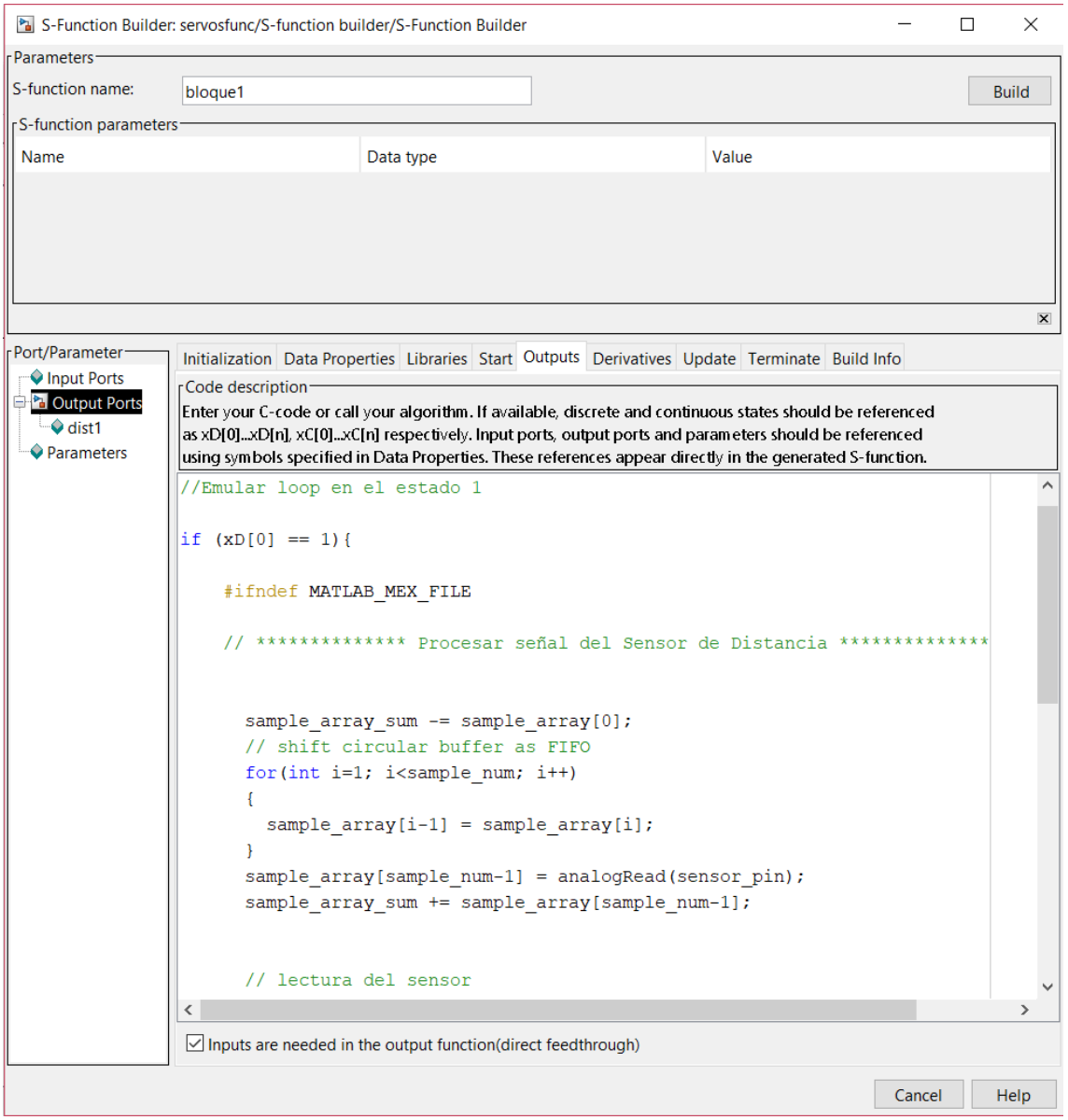


Figura 8.32. Pestaña Outputs del bloque s-function.

Fuente. Elaboración propia.

Al finalizar la actualización de las salidas se les asigna el valor deseado a las variables declaradas dentro de la pestaña "Data Properties" (ex: dist1 para la lectura del sensor). De nuevo, se debe crear una interfaz entre Matlab y el código C para poder ejecutarlo y se logra con la cabecera #ifndef MATLAB_MEX_FILE para iniciar y con la cabecera #endif para señalar el final.

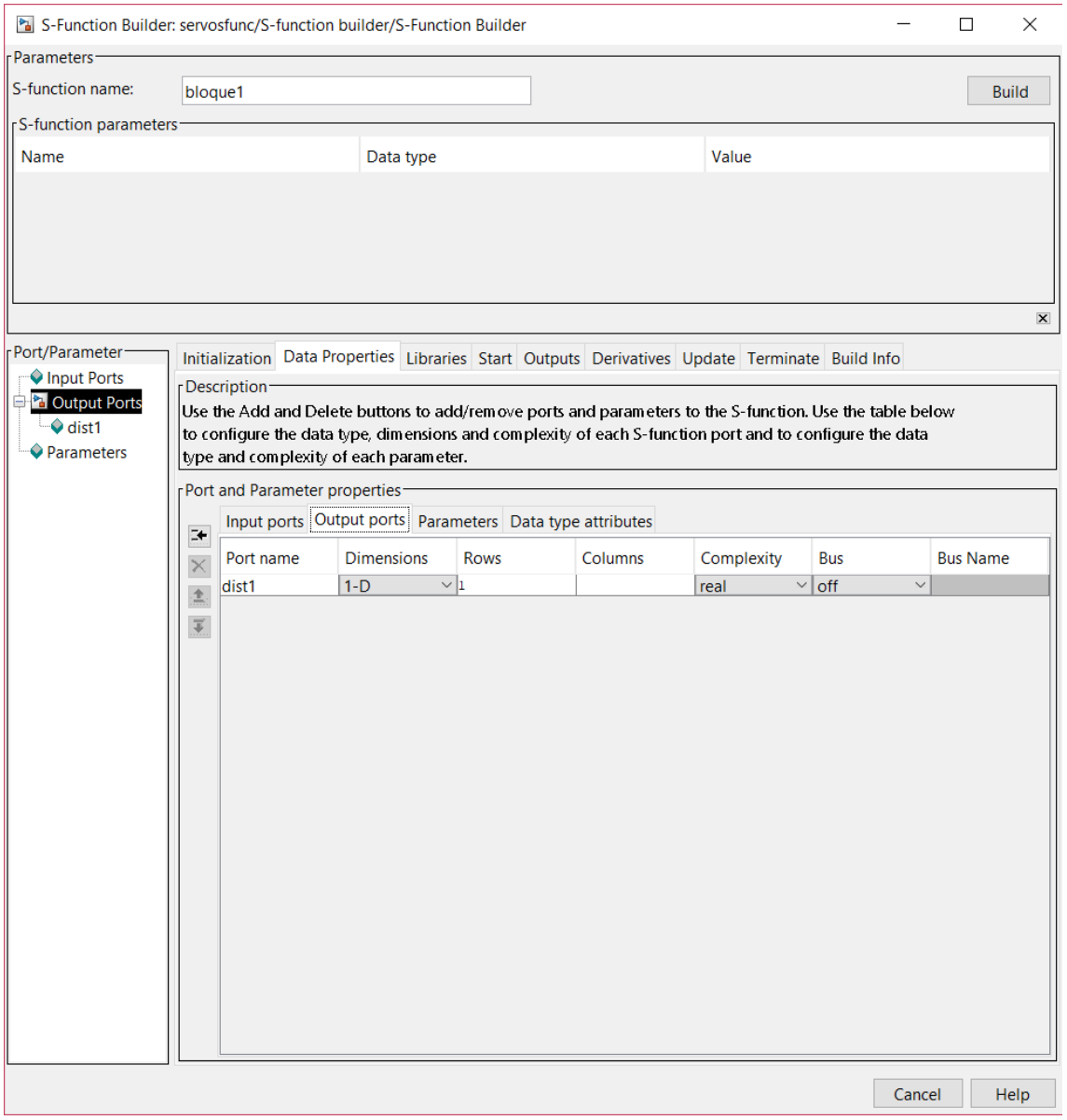


Figura 8.33. Pestaña Data Properties del bloque s-function.

Fuente. Elaboración propia.

En la pestaña "Update" se define el estado discreto inicial/cero mediante el cual se produce la inicialización del Arduino, es decir, se ejecuta el set-up. Como se pretende ejecutar la inicialización una sola vez, al finalizarla se le indica al programa que debe cambiar de estado y pasar al estado uno, en el cual se debe mantener durante el resto de la ejecución. El cambio de estado se indica mediante el comando "xD[0]=1". De este modo, cuando transcurra el Ts fijado y el sistema vaya a ejecutar de nuevo el código de la pestaña "Update" se encontrará con que el estado no es "xD[0]" si no "xD[1]" y como

la condición para ejecutar el set-up es "xD[0]=0" saltará a ejecutar el código de la pestaña "Outputs", es decir, el estado uno o loop que es el funcionamiento deseado.

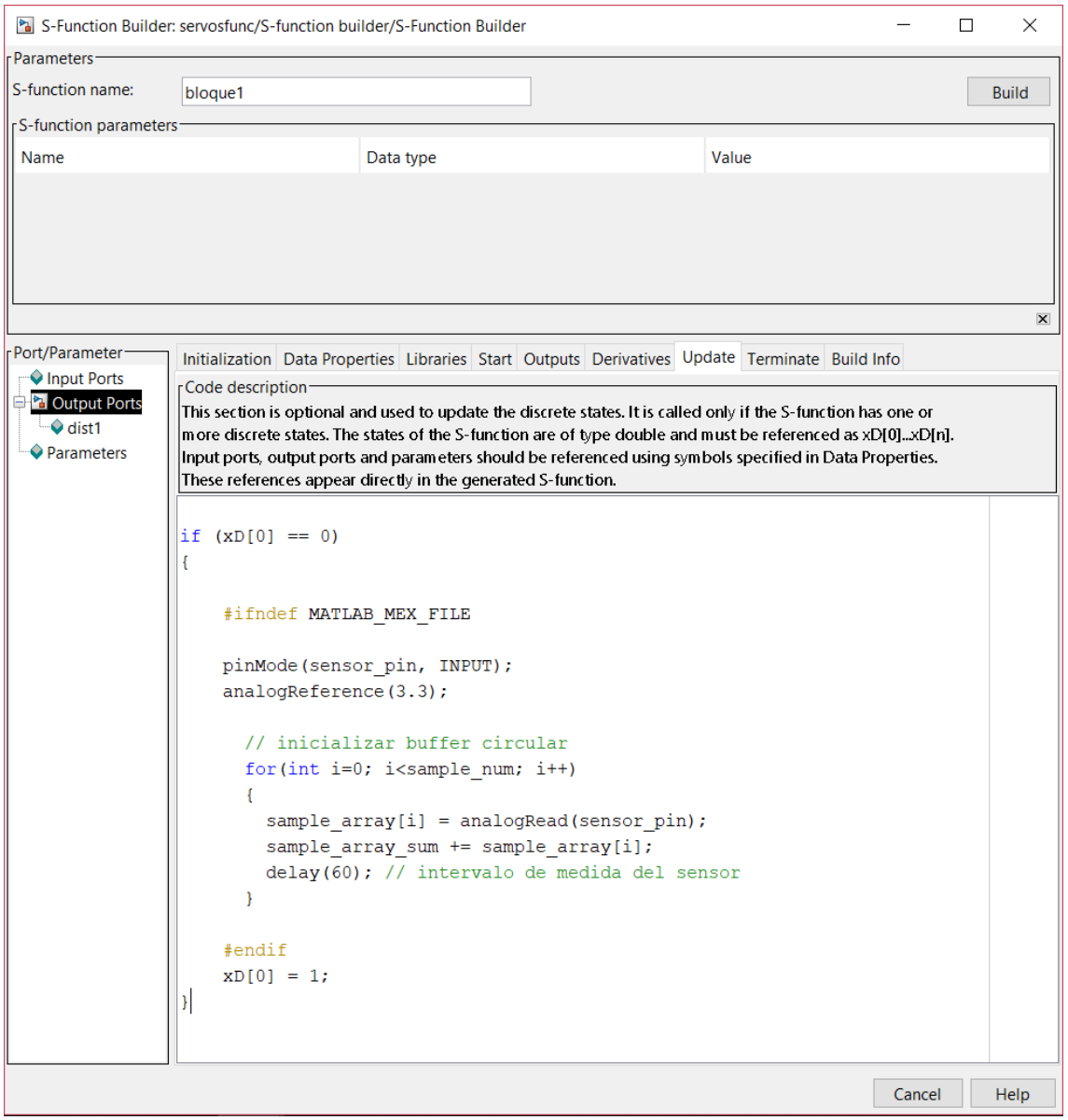


Figura 8.34. Pestaña Update del bloque s-function.

Fuente. Elaboración propia.

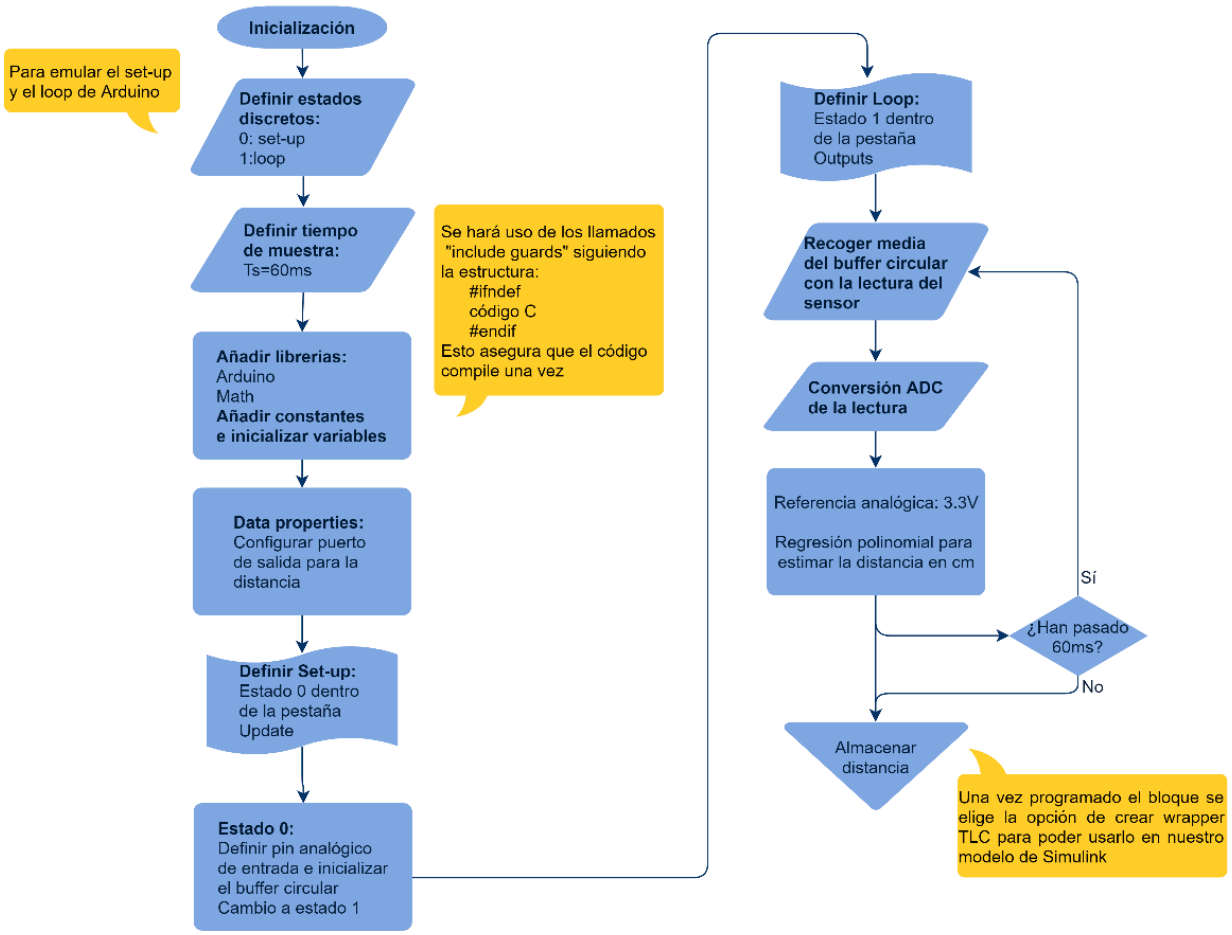


Figura 8.35. Descripción algorítmica del bloque s-function builder.

Fuente. Elaboración propia.

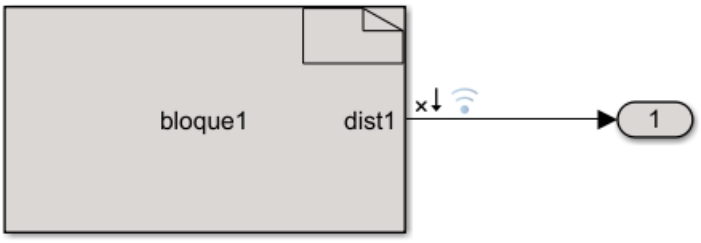


Figura 8.36. Bloque s-function builder.

Fuente. Elaboración propia.

Cuando se hace el "build" del bloque "s-function" se obtienen cuatro archivos: un .c (el que define el bloque), un .tlc, un .mex32 y un .mex64.

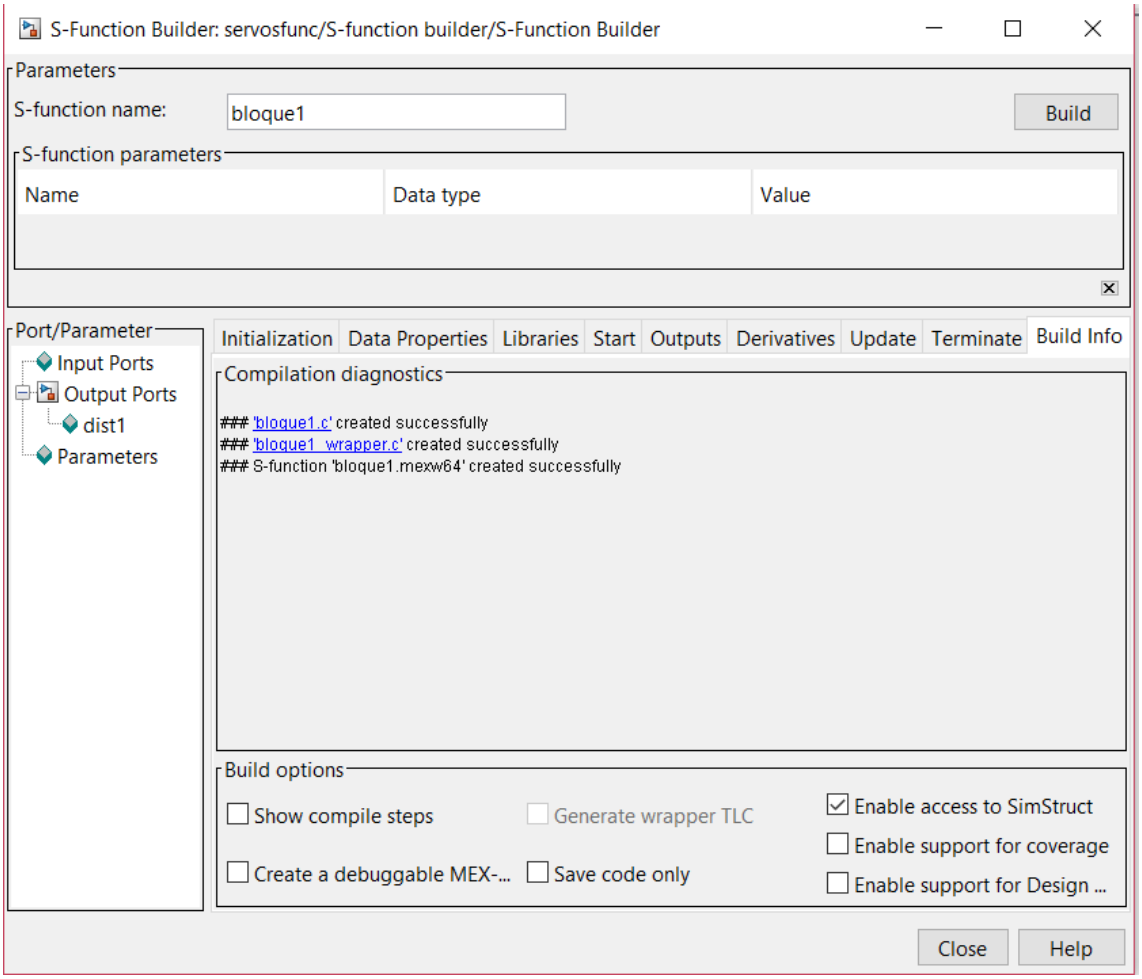


Figura 8.37. Construcción del bloque s-function builder.

Fuente. Elaboración propia.

MATLAB llama al compilador de C y se crean conflictos pues no es capaz de acceder a las librerias C++. El primer problema que se encuentra entonces es que la Arduino.h y math.h incluidas en Arduino IDE son unas librerías en C++ que no pueden ser incluidas en un archivo TLC directamente para poder utilizar sus funciones. Un archivo TLC es el que indica a Simulink cómo debe compilar el código para la placa seleccionada de Arduino en la configuración de parámetros de Simulink. Para que MATLAB pueda acceder a las funciones de C++ necesita que se le cambie la cabecera al archivo .c. La clave para solucionar este problema es la utilización de funciones "wrapper". De esta manera las funciones C++ son envueltas con un wrapper de función C para poder ser llamadas por el archivo TLC. Este cambio se consigue mediante el comando renc2cpp('nombre_bloque_sfunction'). Al ejecutar

este comando, el programa busca el wrapper .c creado por el "build" del bloque s-function para convertir el archivo a un archivo .cpp.

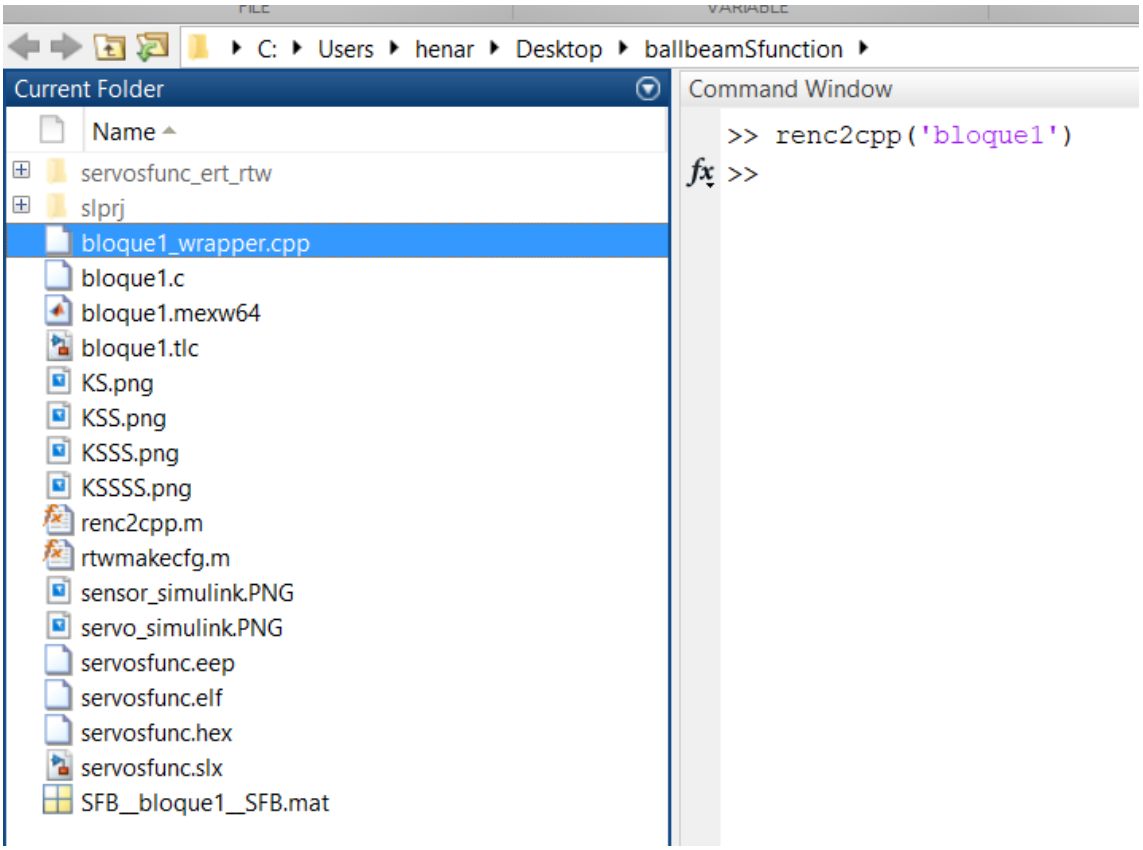


Figura 8.38. Comando renc2cpp genera wrapper .cpp.

Fuente. Elaboración propia.

El segundo problema que se presenta es comprender la generación automática de código para Arduino. Para ello, se han revisado las opciones de compilación y los ficheros en C/C++ que se crean (Simulink genera un informe en HTML con su con su estructura) y se ha visto que la ejecución del modelo de Simulink se implementa cada Ts mediante una ISR que en el caso del Arduino Mega se realiza mediante el Timer1. El generador de código de Simulink calcula automáticamente el valor del prescaler y del registro TCNT para que la cuenta hasta el valor límite se corresponda con el Ts elegido; en este caso para un Ts de 60ms:

$$f_{bit} = \frac{16MHz}{256} = 62.5kHz \rightarrow T_{bit} = \frac{1}{f_{bit}} = 16\mu s \quad (8.8)$$

$$T_{timer} = (2^{16} - 61786) * T_{bit} = 60ms = Ts \quad (8.9)$$

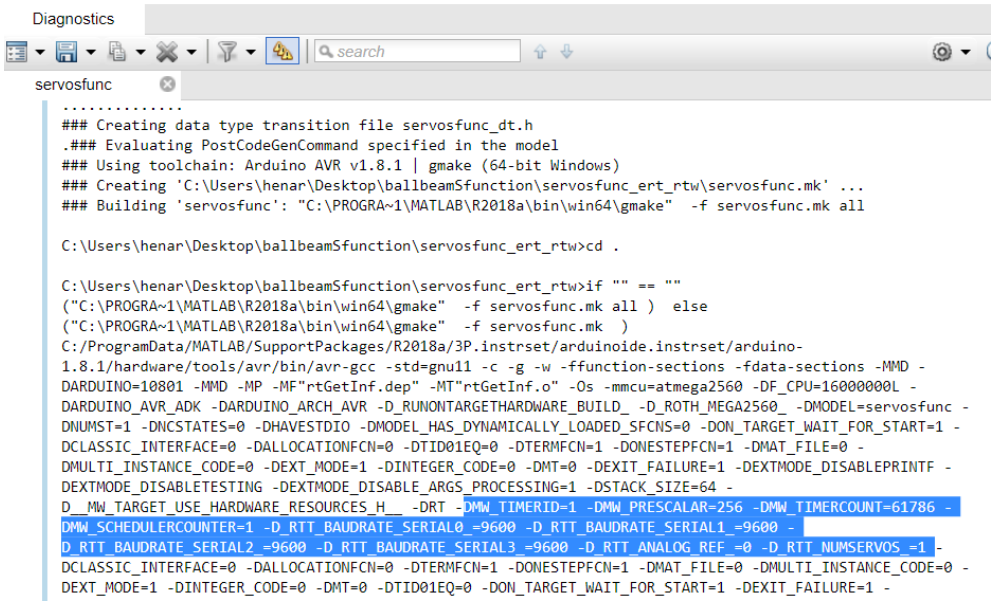


Figura 8.39. Código generado por MATLAB para que coincida el Ts.

Fuente. Elaboración propia.

Cuando salta la ISR del timer en Arduino se ejecuta cada Ts una función step() que contiene una llamada a outputs() (código de la pestaña de la S-Function) y a discrete_update() (código de la pestaña de la S-Function).

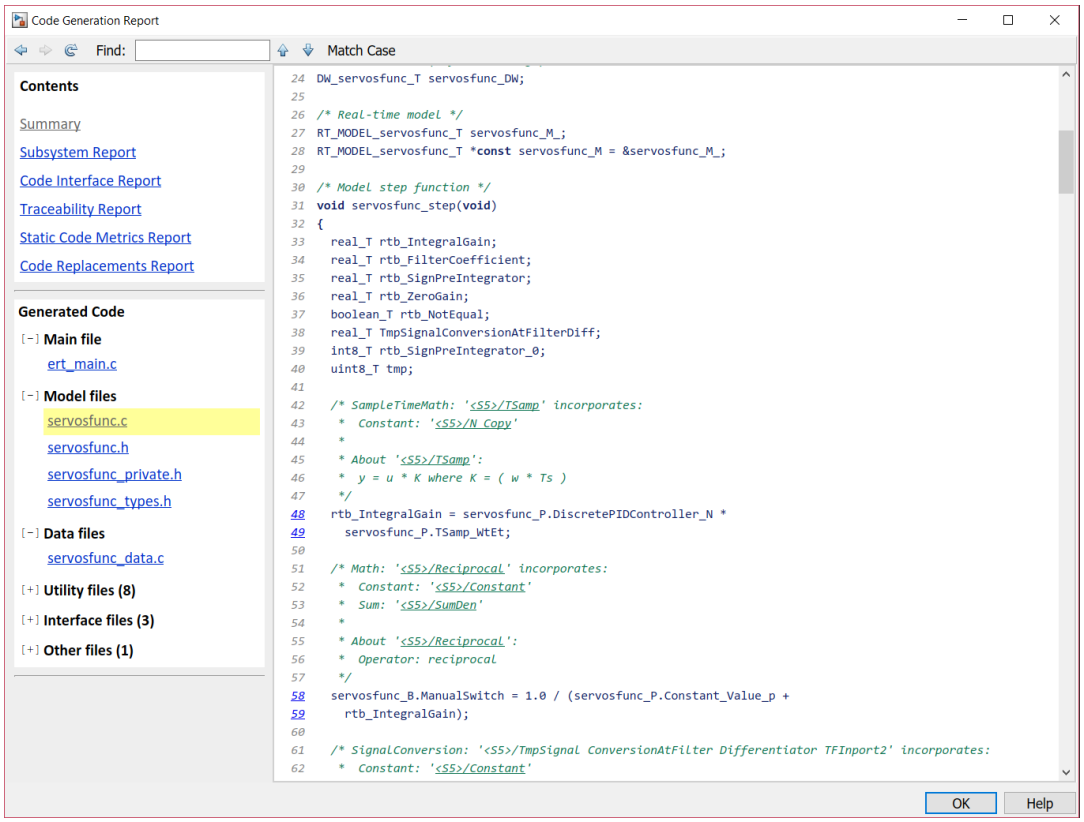


Figura 8.40. Código generado en el archivo .c.

Fuente. Elaboración propia.

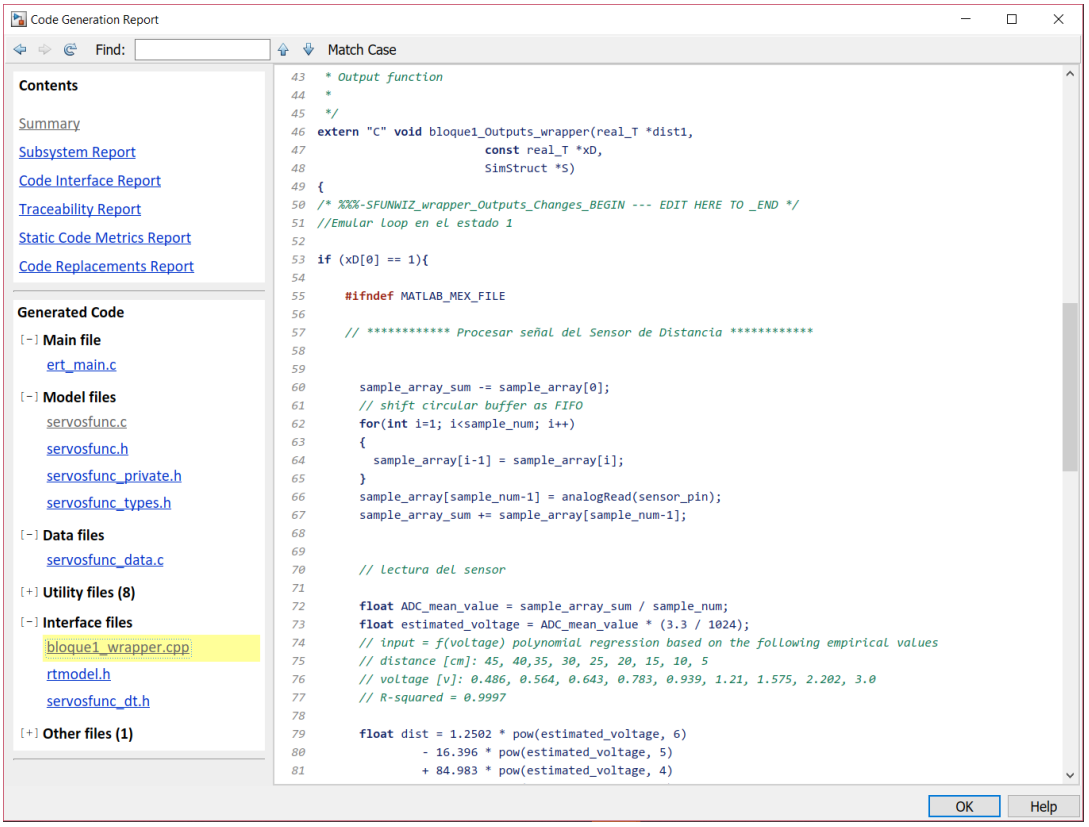


Figura 8.41. Código generado en el wrapper.cpp.

Fuente. Elaboración propia.

En caso de que el Ts del modelo no coincida con el Ts del bloque S-Function, se crean dos funciones step(): una para el Ts base del modelo (con el Timer) y otra para el Ts del bloque (pero el Ts del bloque s-function debe ser múltiplo del Ts base del modelo). Para evitar conflictos con el Ts se ha definido en "configuration parameters" de Simulink "auto" como "sample time" para que coja el Ts que se ha definido previamente en del bloque "s-function builder" por defecto. Además, en el bloque de PID se ha puesto "-1" en el "sample time" para que automáticamente coja el valor de Ts del bloque de la s-function.

La "servo library" usa el Timer 5 de 16bits para generar la señal de control del servo, por lo que, si se utiliza Arduino Mega como target, se puede crear un solapamiento entre la configuración del Ts y la de la PWM, por eso desaconsejan en la documentación del toolbox de Arduino hacer uso del bloque servo write con el External mode. Aunque en este caso el timer utilizado para generar la ISR ha sido el TMR1.

8.2.2.3 MODELO DE BLOQUES

La programación se ha realizado en dos modelos; el primero está compuesto por bloques predefinidos y el segundo consta de un bloque "s-function builder". Los dos modelos se dividen en tres partes principales: la captura de datos, el controlador PID y la generación de la señal para el servo.

- Modelo de bloques predefinidos:

La programación de un sistema en el entorno de Simulink resulta sencillo utilizando bloques predefinidos. En este caso la señal recibida del sensor se lee por el pin A15 de la placa. Esa señal debe convertirse como se ha explicado en apartados anteriores y para ello se ha creado el subsistema "Cálculo de la distancia". Este subsistema consta de una serie de bloques operacionales que logran transformar la señal a centímetros. Después se calcula el error respecto a la consigna y se envía a un bloque PID discreto fácil de sintonizar en modo "External" mientras el modelo se ejecuta. La salida del PID es la señal de control que recibe el servomotor.

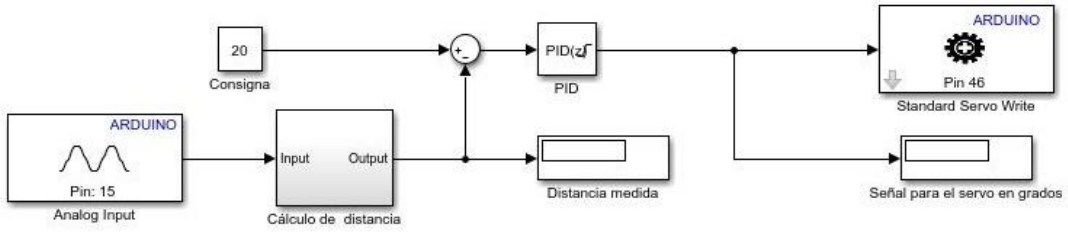


Figura 8.42. Modelo de bloques predefinidos.

Fuente. Elaboración propia.

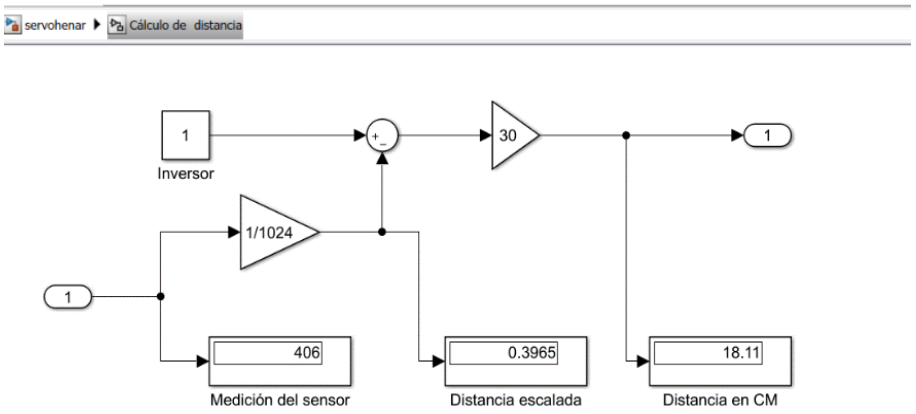


Figura 8.43. Subsistema de bloques para la conversión de la lectura a cm.

Fuente. Elaboración propia.

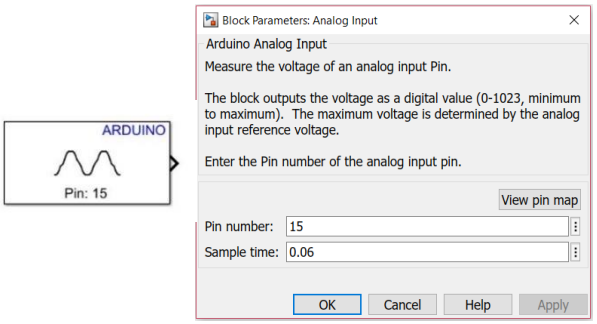


Figura 8.44. Configuración del bloque analog input.

Fuente. Elaboración propia.

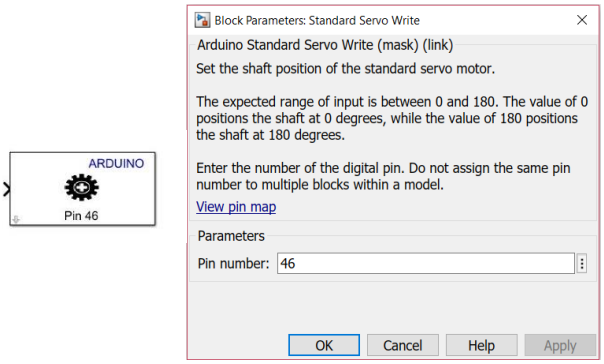


Figura 8.45. Configuración del bloque servo write.

Fuente. Elaboración propia.

Como vemos, los bloques predefinidos son una opción atractiva por la facilidad de programación, pero limitan el control sobre la lectura ya que no se puede hacer uso de recursos como un buffer circular, un filtro paso bajo, la posibilidad de cambiar la escala de referencia para el ADC o la ecuación de la regresión polinómica obtenida del sensor. Ahora se muestra el modelo de bloques que hace uso de una s-function. Para poder hacer uso de estos recursos tenemos el siguiente modelo.

- Modelo de bloques con s-function builder:

En este caso el cambio se produce en el modo de capturar la lectura del sensor, el resto del modelo es igual al anterior. Con la posibilidad de implementar los recursos mencionados en el modelo anterior gracias al uso de código C, se consigue aprovechar todo el poder computacional para controlar la lectura del sensor.

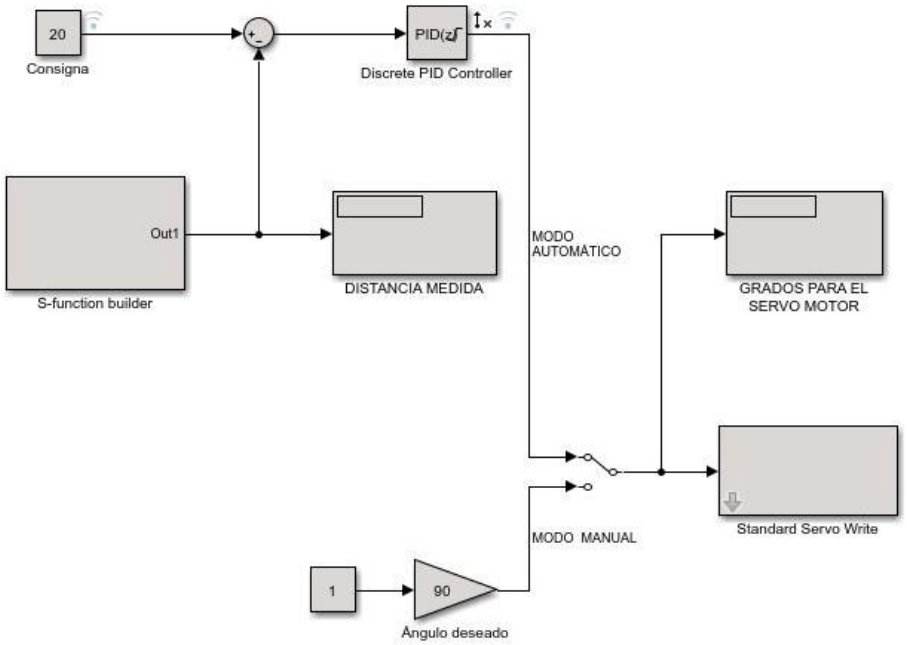


Figura 8.46. Diseño de bloques del modelo con s-function.

Fuente. Elaboración propia.

En el diagrama que se presenta a continuación, se observa el funcionamiento del sistema con s-function, el cual demuestra la ventaja de implementar C en Simulink:

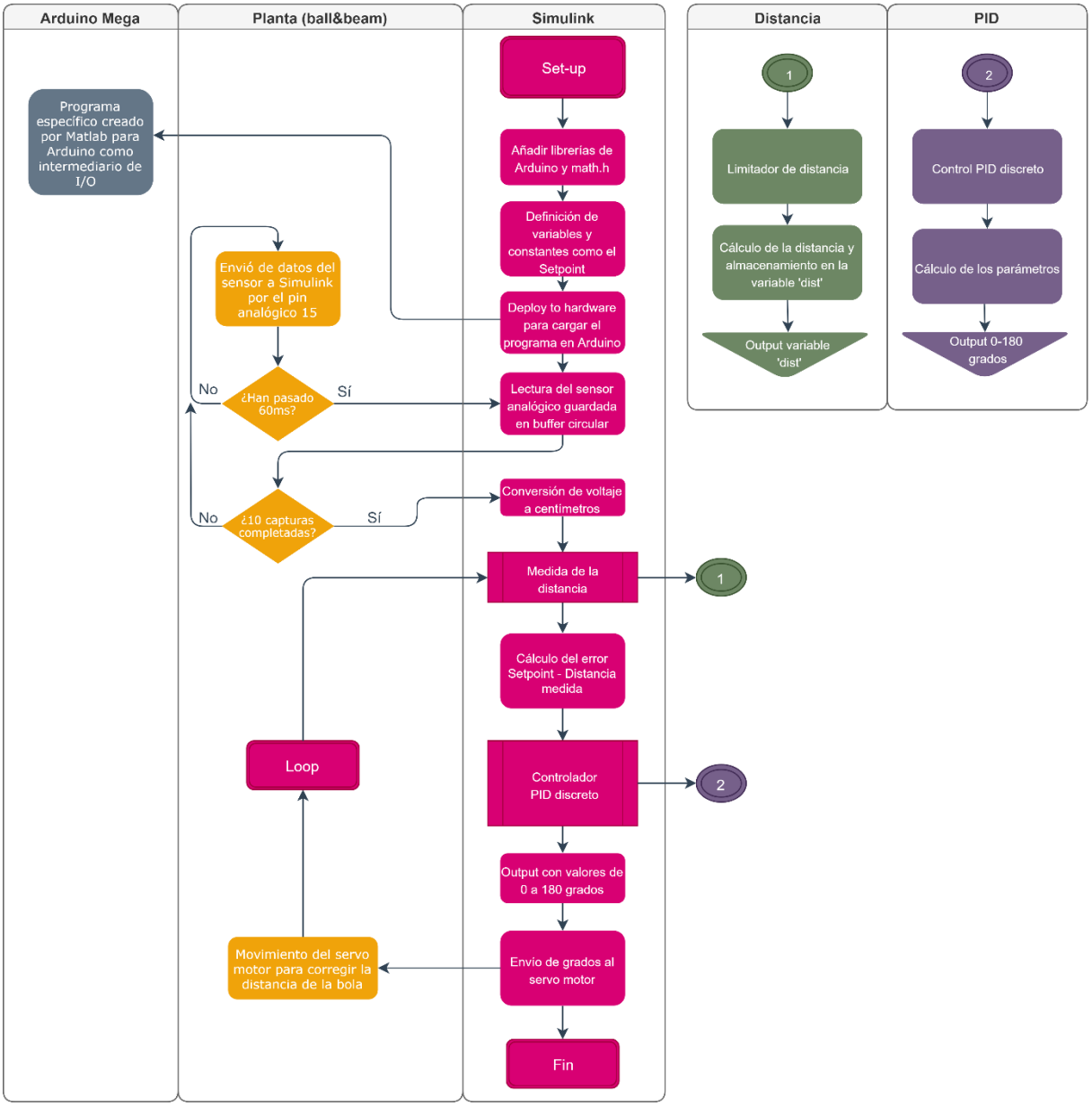


Figura 8.47. Diagrama de funcionamiento del sistema completo con s-function. Fuente. Elaboración propia.

8.2.2.4 SINTONIZACIÓN DEL BLOQUE PID

Un sistema de control por realimentación necesita ejecutarse cada un tiempo determinado. A continuación, se aprecia la configuración de un control PID discreto para usar en conjunto con un sistema Simulink-Arduino.

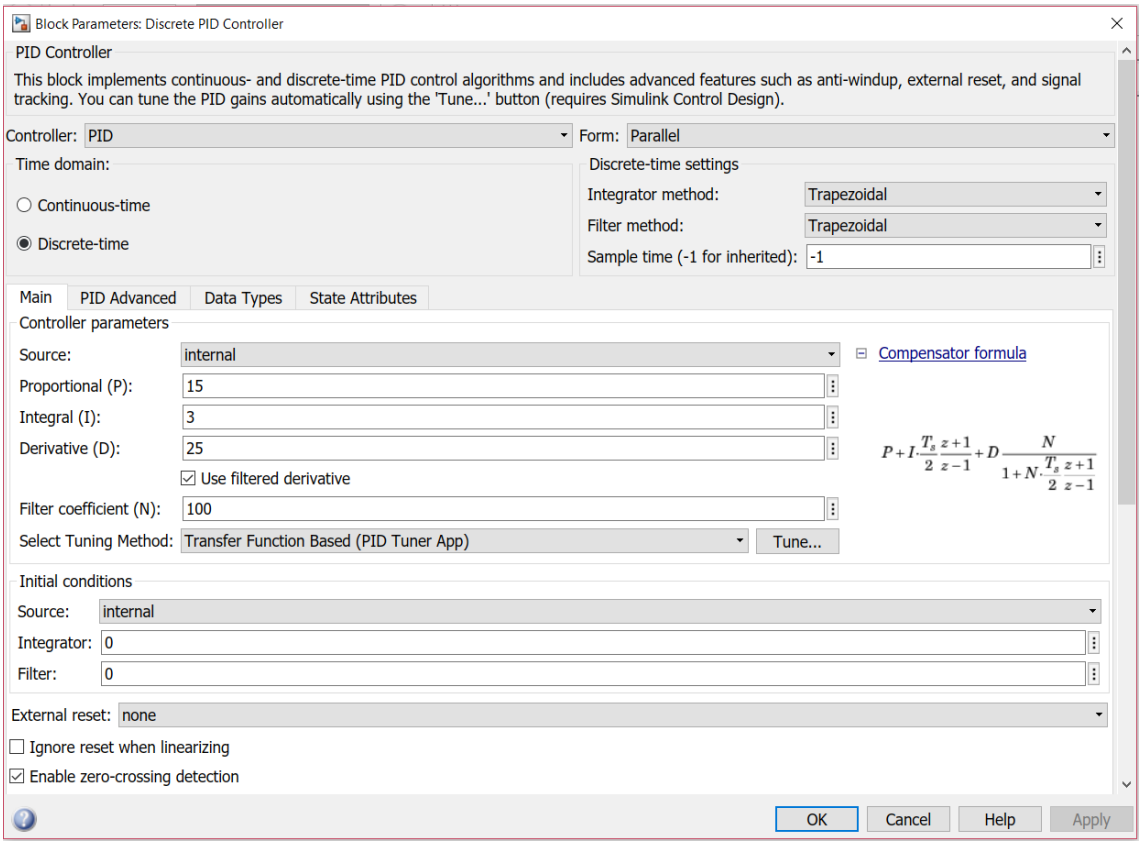


Figura 8.48. Configuración del bloque PID.

Fuente. Elaboración propia.

Se ha seleccionado el bloque PID discreto porque el microcontrolador trabaja con tiempos discretos.

Como hemos visto en los apartados anteriores, un controlador PID permite mantener controlada la variable principal del proceso para que se acerque lo máximo posible al valor determinado como consigna a través de un lazo de realimentación. En este caso el control se realiza sobre la posición de la bola orientando unas barras con el eje de un servomotor conectado a una manivela. En el proceso de estabilización del sistema entran en juego las tres acciones que definen este tipo de control: control proporcional, control

integral y control derivativo. Cada uno de los componentes actúa sobre la señal de error. (Picuino, 2019)

La correcta sintonización de estos parámetros sirve para controlar de manera efectiva el comportamiento de la planta. Una mala sintonización puede resultar en un sistema inestable. El PID implementado en Simulink es un controlador PID discreto con aproximación trapezoidal al ser la más adecuada en este caso. El valor de salida se corresponde al ángulo que debe tomar el servo y por ello la salida está en un rango de 0 a 180 grados.

Los parámetros PID han sido ajustados manualmente siguiendo este procedimiento:

- La distancia establecida como consigna es de 20cm.
- Se desestabiliza la bola y se espera a que se estabilice de nuevo.

Los parámetros han sido sintonizados mediante distintas pruebas en la maqueta y los resultados han sido recogidos en el repositorio de GitHub de este proyecto. El enlace se encuentra en los anexos.

8.2.2.5 PANTALLA INTERACTIVA

Dentro de Simulink encontramos la herramienta Dashboard Block Library. Con ella se puede diseñar una interfaz en la que el usuario vea la información e interactúe con la planta de manera sencilla cambiando los valores de los parámetros. Se ha diseñado la siguiente interfaz:

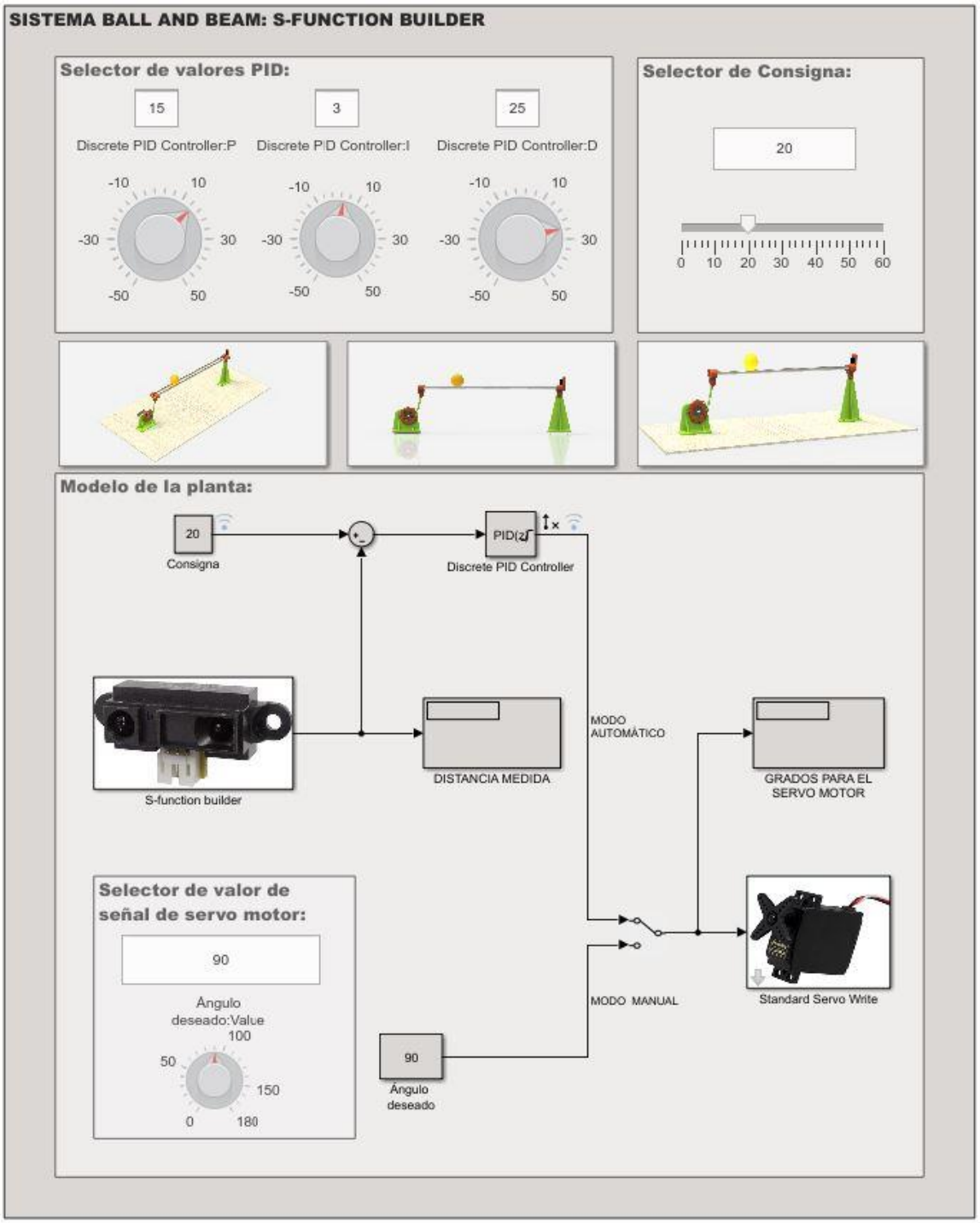


Figura 8.49. Pantalla interactiva creada en Simulink.

Fuente. Elaboración propia.

9. TRABAJO FUTURO

Con vistas a mejorar el proyecto desarrollado se han tenido en cuenta los siguientes factores:

- Visualización de los datos: cuando el sistema corre standalone en Arduino sin estar conectado a Simulink, se podría incluir una pantalla LCD Keypad Shield para Arduino para visualizar los parámetros del PID, la consigna y la distancia medida.
- Fiabilidad: algunas lecturas resultan erróneas y afectan a la fiabilidad del sistema. Se podría solucionar cambiando el sensor de distancia analógico por uno digital ya que disminuye el ruido. También se podría cambiar el servomotor por uno que emita menos ruido.

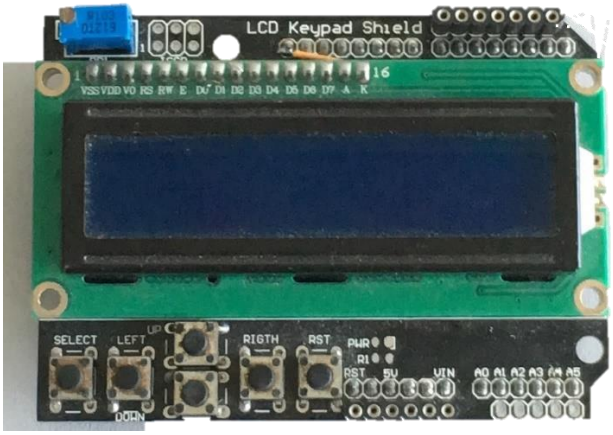


Figura 9.1. LCD Keypad Shield para Arduino.

Fuente. Elaboración propia.

- Optimización: cuando Simulink compila el modelo y genera código para ejecutar en Arduino, se ha visto analizando el código generado que hace uso del timer 5 para crear una ISR. A su vez, el timer 5 es utilizado por defecto en servo library para generar la señal de control del servo. Quizá el sistema no esté funcionando debidamente por el solapamiento que se produce. Habría que configurar otro timer (por ejemplo, el 1).

10. PLAN DE TRABAJO

Una vez descrito el diseño del proyecto es necesario definir un plan de trabajo compuesto por un equipo de trabajo, paquetes de trabajo y tareas necesarias para su desarrollo. Tras definir la planificación se presenta toda la información en un diagrama de Gantt.

En los siguientes apartados se va a detallar la duración, los responsables y los recursos utilizados durante todo el proyecto. A su vez, se detalla la fecha de inicio y la fecha de fin de proyecto en conjunto con cada tarea realizada.

10.1 EQUIPO DE TRABAJO

En este plan el equipo encargado de llevar a cabo el proyecto y el cargo de cada uno se presenta en la siguiente tabla:

Tabla 10.1 Equipo de trabajo.

Código	Nombre	Responsabilidad
H1	Oskar Casquero Oyarzabal	Tutor de Proyecto
H2	Henar Brenlla Garcia	Ingeniera Junior
H3	Ekain Uriguen Fernandez	Gestor
H4	Cesar Perez Barrio	Técnico de laboratorio

Fuente. Elaboración propia.

10.2 PAQUETES DE TRABAJO

Cada paquete de trabajo tiene diferentes tareas que lo completan.

Tabla 10.2 Paquetes de trabajo del proyecto.

Paquete de trabajo	Nombre	Comienzo	Final	Entregables
WP1	Gestión	Día 1	Día 123	
WP2	Preparación	Día 1	Día 17	D2.1, D2.2
WP3	Desarrollo	Día 17	Día 52	D3.1, D3.2, D3.3
WP4	Pruebas	Día 52	Día 89	D4.1
WP5	Documentación	Día 89	Día 123	D5.1

Fuente. Elaboración propia.

Cada paquete de trabajo consta de distintos entregables e hitos. A continuación, se indica en qué consisten y cómo afectan en la organización del plan.

Los entregables son los siguientes:

- **D2.1:** Redacción de análisis de alternativas y descripción de requerimientos.
  Entrega: Día 4
- **D2.2:** Selección de la solución propuesta.
  Entrega: Día 6
- **D3.1:** Impresión de las piezas y montaje.
  Entrega: Día 28
- **D3.2:** Programación en Arduino.
  Entrega: Día 34
- **D3.3:** Programación en Simulink.
  Entrega: Día 40
- **D4.1:** Realización de pruebas en la maqueta.
  Entrega: Día 85
- **D5.1:** Documentación completa del proyecto.
  Entrega: Día 123.

Tabla 10.3 Hitos del plan de proyecto.

Paquete de trabajo	Nombre	Fecha
H1	Análisis de alternativas finalizado	Día 4
H2	Estudio de los conceptos a utilizar en el proceso completado	Día 10
H3	Montaje de la estructura terminado	Día 29
H4	Modelo de bloques y programación del sensor creados	Día 44
H5	Desarrollo del programa terminado	Día 52
H6	Optimización del código tras realización de pruebas realizado	Día 89
H7	Entrega de la documentación	Día 123

Fuente. Elaboración propia.

10.2.1GESTIÓN

Para una correcta organización y ejecución del plan se ha definido el WP1. Ha sido necesario organizar el trabajo a realizar en entregables e hitos. Para las comprobaciones y resolución de problemas se han fijado reuniones durante todo el proyecto cada dos semanas. La duración de la gestión ha sido de 123 días.

Responsables: Oskar Casquero Oyarzabal, Henar Brenlla Garcia.

Participantes: Todos.

10.2.2PREPARACIÓN Y RECOPILACIÓN DE DATOS

El WP2 consiste en recabar la información y el conocimiento necesario para poder comenzar a desarrollar la aplicación. Para ello se requieren las siguientes tareas:

- **Tarea 2.1:** Análisis de alternativas y descripción de requerimientos.

Consiste en reunir la información necesaria para comparar las alternativas disponibles para el proyecto.

Responsable: Henar Brenlla Garcia.

Participantes: Todos.

Resultados: D2.1 Redacción de análisis de alternativas y descripción de requerimientos.

Duración: 5 días.

  - **Tarea 2.1.1:** Entender las posibilidades que ofrece la implementación de un bloque s-function builder en Simulink.

Responsable: Oskar Casquero Oyarzabal.

Participantes: Henar Brenlla Garcia, Ekain Uriguen Fernandez.

Resultados: Comprensión del uso de una s-function y sus ventajas.

Duración: 2 días.

- **Tarea 2.2:** Selección de la solución propuesta.

Consiste en redactar la solución propuesta tras analizar las alternativas y seleccionar los componentes y materiales.

Responsable: Henar Brenlla Garcia.

Participantes: Todos.

Resultados: D2.2 Selección de la solución propuesta.

Duración: 1 día.

- **Tarea 2.3:** Reunión con el tutor del proyecto.

Responsable: Oskar Casquero Oyarzabal.

Participantes: Henar Brenlla Garcia.

Resultados: Solución de problemas y aclaración de conceptos.

Duración: 1 día.

10.2.3DESARROLLO

En el WP3 se procede a montar la estructura del prototipo y programar el microcontrolador junto con simulink.

- **Tarea 3.1:** Diseño e impresión de las piezas y montaje de la estructura.

Consiste en la impresión de piezas y montaje del prototipo tras su diseño.

Responsable: Henar Brenlla Garcia.

Participantes: Todos.

Resultados: D3.1 Impresión de las piezas y montaje.

Recursos: Impresora 3D.

Duración: 12 días.

- **Tarea 3.2:** Programación del sensor en Arduino.

Consiste en programar la captura de datos en C mediante Arduino IDE.

Responsable: Henar Brenlla Garcia.

Participantes: Oskar Casquero Oyarzabal.

Resultados: D3.2 Programación en Arduino.

Recursos: Arduino IDE, Microsoft Excel.

Duración: 8 días.

- **Tarea 3.3:** Programación del modelo en Simulink.

Consiste en hacer el modelo de bloques en Simulink para controlar la planta.

Responsable: Henar Brenlla Garcia.

Participantes: Oskar Casquero Oyarzabal.

Resultados: D3.3 Programación en Simulink.

Recursos: MATLAB.

Duración: 7 días.

- **Tarea 3.4:** Programación para testear el servomotor en conjunto con el sensor.

Consiste en programar la captura de datos en C mediante Arduino IDE.

Responsable: Henar Brenlla Garcia.

Participantes: Ekain Uriguen Fernandez.

Resultados: Estudiar el rango de operación y modo de trabajo del servomotor.

Recursos: Arduino IDE, Microsoft Excel.

Duración: 8 días.

- **Tarea 3.5:** Reunión con el tutor del proyecto.

Responsable: Oskar Casquero Oyarzabal.

Participantes: Henar Brenlla Garcia.

Resultados: Solución de problemas y aclaración de conceptos.

Duración: 1 día.

10.2.4PRUEBAS

Una parte fundamental del proyecto consiste en la realización de pruebas para comprobar el correcto funcionamiento del desarrollo.

- **Tarea 4.1:** Realización de pruebas en la maqueta

Consiste en probar la respuesta de la planta con el control aplicado.

Responsable: Henar Brenlla Garcia.

Participante: Ekain Uriguen Fernandez.

Resultados: D4.1 Realización de pruebas en la maqueta.

Recursos: MATLAB.

Duración: 33 días.

- **Tarea 4.1.1:** Prueba de lectura del sensor.

Responsable: Oskar Casquero Oyarzabal.

Participante: Henar Brenlla Garcia.

Resultados: Optimizar el control sobre la lectura.

Recursos: MATLAB.

Duración: 9 días.

- **Tarea 4.1.2:** Prueba de movimiento del servo según la lectura del sensor.

Responsable: Oskar Casquero Oyarzabal.

Participantes: Henar Brenlla Garcia.

Resultados: Comprobar el funcionamiento de los bloques.

Recursos: MATLAB.

Duración: 7 días.

- **Tarea 4.1.3:** Prueba de modelo con PID discreto implementado.

Responsable: Oskar Casquero Oyarzabal.

Participante: Henar Brenlla Garcia.

Resultados: Comprobar el funcionamiento de los bloques.

Recursos: MATLAB.

Duración: 17 días.

- **Tarea 4.2:** Reunión con el tutor del proyecto.

Responsable: Oskar Casquero Oyarzabal.

Participantes: Henar Brenlla Garcia.

Resultados: Solución de problemas y aclaración de conceptos.

Duración: 1 día.

10.2.5DOCUMENTACIÓN

- **Tarea 5.1:** Documentación completa del proyecto.

Consiste en la redacción de la memoria que debe ser aprobada por el tutor del proyecto. Este documento se puede servir de base para futuros proyectos.

Responsable: Oskar Casquero Oyarzabal.

Participante: Henar Brenlla Garcia.

Resultados: D5.1 Documentación completa del proyecto.

Recursos: Microsoft Word.

Duración: 34 días.

- **Tarea 5.1.1:** Diagramas del sistema.

Diagramas explicativos del sistema.

Responsable: Oskar Casquero Oyarzabal.

Participante: Henar Brenlla Garcia.

Resultados: Diagramas de flujo usados en el proyecto.

Recursos: Microsoft Word, Draw.io.

Duración: 6 días.

- **Tarea 5.1.2:** Planos.

Realización de planos del programa.

Responsable: Oskar Casquero Oyarzabal.

Participantes: Henar Brenlla Garcia.

Resultados: Diagramas de flujo usados en el proyecto.

Recursos: Autodesk Inventor 2019.

Duración: 9 días.

- **Tarea 5.1.3:** Redacción.

  Redacción de la documentación.

  Responsable: Oskar Casquero Oyarzabal.

  Participantes: Henar Brenlla Garcia.

  Resultados: Diagramas de flujo usados en el proyecto.

  Recursos: Microsoft Word.

  Duración: 19 días.

- **Tarea 5.2:** Reunión con el tutor del proyecto.

Resultados: Solución de problemas y aclaración de conceptos.

Duración: 1 día.

10.3 DIAGRAMA DE GANTT

El tiempo total para desarrollar el proyecto ha sido de 310 horas. En este apartado se puede observar la línea temporal de todo el proyecto mediante un diagrama de Gantt.

EDT	Nombre	Fecha de inicio	Fecha de fir
Proyecto	Control de un sistema ball and beam mediante simulink-arduino	8/10/18	18/07/19
WP1	Gestión	8/10/18	17/07/19
	H7	18/07/19	18/07/19
WP2	Preparación y recopilación de datos	24/10/18	2/11/18
T 2.1	Análisis de alternativas y descripción de requerimientos	24/10/18	30/10/18
T 2.1.1	Entender las posibilidades que ofrece la implementación de un bloque s-function builder en Simulink	24/10/18	30/10/18
	H1	31/10/18	31/10/18
T 2.2	Selección de la solución propuesta	31/10/18	31/10/18
	H2	2/11/18	2/11/18
T 2.3	Reunión con el tutor	2/11/18	2/11/18
WP3	Desarrollo	21/01/19	22/03/19
T 3.1	Diseño e impresión de las piezas y montaje de la estructura	21/01/19	27/02/19
	H3	28/02/19	28/02/19
T 3.2	Programación del sensor en Arduino	28/02/19	11/03/19
T 3.3	Programación del modelo en Simulink	12/03/19	20/03/19
	H4	21/03/19	21/03/19
T 3.4	Programación para testear el servomotor en conjunto con el sensor	21/03/19	21/03/19
	H5	22/03/19	22/03/19
T 3.5	Reunión con el tutor	22/03/19	22/03/19
WP4	Pruebas	26/03/19	12/06/19
T 4.1	Realización de pruebas en la maqueta Consiste en probar la respuesta de la planta con el control aplicado	26/03/19	11/06/19
T 4.1.1	Prueba de lectura del sensor	26/03/19	8/04/19
T 4.1.2	Prueba de movimiento del servo según la lectura del sensor	9/04/19	26/04/19
T 4.1.3	Prueba de modelo con PID discreto implementado	29/04/19	11/06/19
	H6	12/06/19	12/06/19
T 4.2	Reunión con el tutor	12/06/19	12/06/19
WP5	Documentación	13/06/19	18/07/19
T 5.1	Documentación completa del proyecto	13/06/19	17/07/19
T 5.1.1	Diagramas del sistema	13/06/19	17/06/19
T 5.1.2	Planos	18/06/19	25/06/19
T 5.1.3	Redacción	26/06/19	17/07/19
T 5.2	Reunión con el tutor	18/07/19	18/07/19

Figura 10.1. Organización de tareas del diagrama Gantt.

Fuente. Elaboración propia.

En el diagrama de Gantt se diferencian los paquetes de trabajo según los siguientes colores:

- Verde: Gestión
- Azul: Preparación y recopilación de datos
- Naranja: Desarrollo
- Morado: Pruebas
- Rojo: Documentación

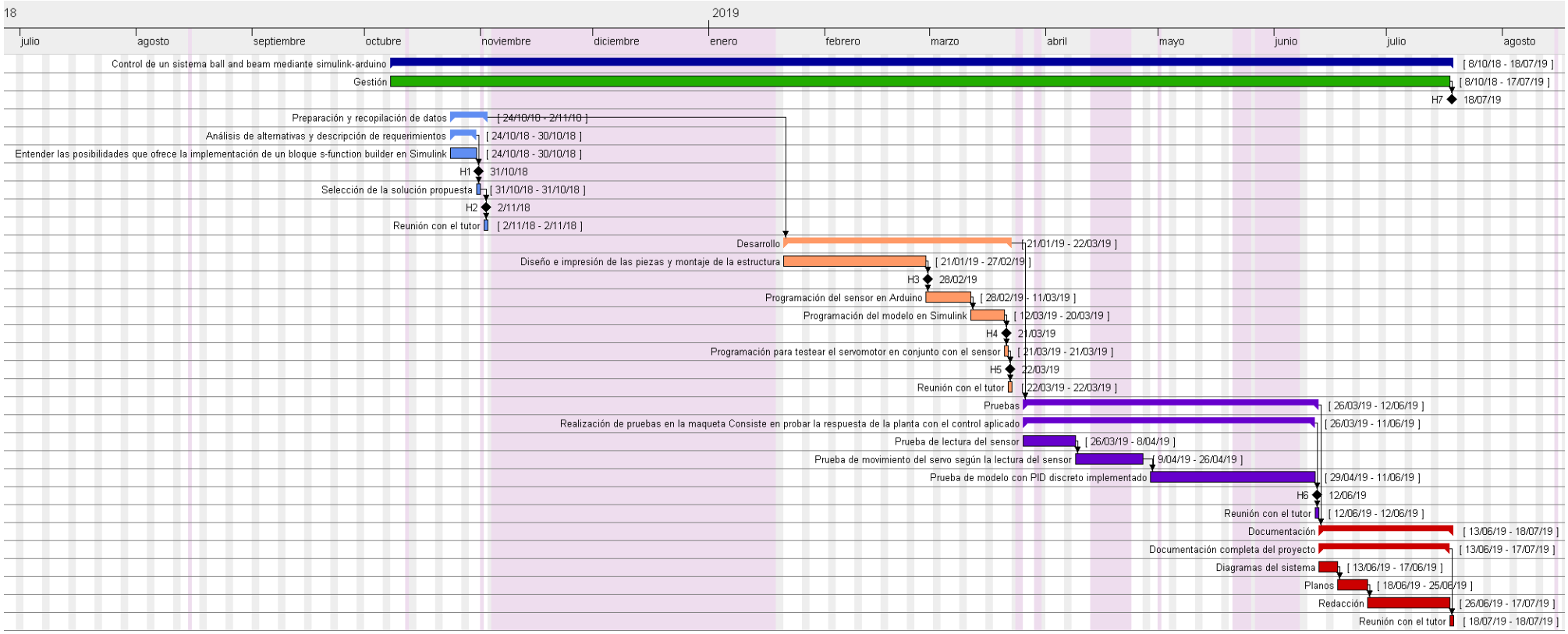


Figura 10.2. Diagrama de Gantt del proyecto.

Fuente. Elaboración propia.

11. PRESUPUESTO

Tras definir la duración y el propósito del proyecto, se detallarán dos presupuestos:

- Costes de desarrollo del proyecto: se calcula a partir de las horas invertidas por el equipo para desarrollar el sistema y los gastos conceptuales. Los costes se dividen en gastos de amortización y gastos de mano de obra.
- Costes de ejecución del proyecto: es el total de los costes para desarrollar la aplicación. Los costes se dividen en gastos materiales, gastos de amortización y gastos de mano de obra.

11.1 GASTOS DE DESARROLLO DEL PROYECTO

En la siguiente tabla aparecen los gastos totales para el desarrollo del proyecto sumando los gastos de mano de obra y de amortización.

Tabla 11.1 Presupuesto total del desarrollo del proyecto.

Concepto	Unidades	Coste Unitario	N.º de unidades	Coste total	Total
Horas internas					8.970 €
Tutor de proyecto	horas	44 €	50	2.200 €	
Técnico de laboratorio	horas	20 €	40	800 €	
Ingeniero junior	horas	12,00 €	310	3.720 €	
Gestor	horas	15,00 €	150	2.250 €	
Amortizaciones					75,90 €
Ordenador	meses	13,50 €	5	67,50 €	
Arduino IDE	SW gratuito				
Office 2016	meses	4,20 €	2	8,40 €	
Costes directos					9.045,90 €
Costes indirectos	10%				904,56 €
Subtotal					9.950,46 €
Imprevistos	10%				995,05 €
TOTAL					**10.945,51 €**

Fuente. Elaboración propia.

11.2 GASTOS DE EJECUCIÓN DEL PROYECTO

En esta tabla aparecen los gastos unitarios de los materiales necesarios para poner en funcionamiento el proyecto.

Tabla 11.2. Presupuesto total de ejecución del proyecto.

Concepto	Unidades	Coste Unitario	N.º unidades	Coste total	Total
Horas internas					2.100 €
Tutor de proyecto	horas	44 €	10	440 €	
Técnico de laboratorio	horas	20 €	20	400 €	
Ingeniero junior	horas	12,00 €	80	960 €	
Gestor	horas	15,00 €	20	300 €	
Amortizaciones					34,95 €
Ordenador	meses	13,50 €	2	27,00 €	
Office 2016	meses	4,20 €	1	4,20 €	
Impresora 3D Prusa i3	meses	3,75 €	1	3,75 €	
Gastos materiales					3.493,45 €
Arduino Mega ADK		36,00 €	1	36,00 €	
Pelota		1,00 €	1	1,00 €	
Barra de aluminio		2,00 €	2	4,00 €	
Sensor Sharp GP2Y0A21		9,90 €	1	9,90 €	
Servomotor Futaba S3003		16,99 €	1	16,99 €	
Cables de conexión		0,60 €	6	3,60 €	
Tabla de madera		3,00 €	1	3,00 €	
Autodesk Inventor 2019		2.613,60 €	1	2.613,60 €	
MATLAB 2018a		800,00 €	1	800,00 €	
PLA		5,00 €	1	5,00 €	
Tornillos		0,04 €	9	0,36 €	
Costes directos					5.628,40 €
Costes indirectos	10%				562,84 €
Subtotal					6.191,24 €
Imprevistos	10%				619,12 €
TOTAL					**6.810,36 €**

Fuente. Elaboración propia.

Analizando los dos presupuestos se observa que la mayor diferencia son la partida de horas internas y los gastos materiales. Si este prototipo fuera producido en masa la economía de costes se vería significativamente reducida.

12. CONCLUSIONES

Tras haber realizado todo el proceso de diseño del control de sistema ball and beam mediante simulink-arduino que se desarrolla en el presente trabajo, se pueden señalar las siguientes ideas principales:

- Utilizar MATLAB/Simulink para la programación con Arduino: MATLAB posibilita que la lectura y escritura de los datos recogidos por el sensor se haga de manera interactiva sin tener que esperar a que el código compile en el microcontrolador. Además, permite el procesado de señales y modelado mediante funciones prediseñadas para el análisis de los datos del sensor. Gracias a los distintos tipos de gráficos que ofrece MATLAB se pueden visualizar los datos de manera rápida. En Simulink se pueden desarrollar y simular algoritmos que generan automáticamente código para ejecutarlo en Arduino.

- Diseño del controlador: en un proyecto de hardware como este ha permitido el diseño de control y procesamiento de señales. A su vez, el ajuste y la optimización de parámetros se hace de forma interactiva mediante los bloques del dashboard de Simulink mientras el código se ejecuta en Arduino.

- Flexibilidad de uso y reproducción: la modificación de algoritmos se efectua de manera sencilla para poder ser adaptado a otras plataformas. Se ha conseguido una maqueta ball and beam completamente funcional con 6 piezas de impresión 3D. Cabe destacar el bajo coste de los componentes utilizados respecto a las plataformas ball and beam comercializadas.

- Uso de HW y SW: se han utilizado tipos de software y hardware libres para poder ser adaptados a otros proyectos con facilidad. Entre el SW utilizado encontramos Cura Ultimaker, Drawio o Fritzing. Entre el HW utilizado encontramos la placa Arduino Mega ADK, así como los sensores y el actuador utilizado.

- Optimización: se ha concluido que habría que identificar el sistema y linealizar el modelo para una sintonización del controlador PID óptima.

13. REFERENCIAS BIBLIOGRÁFICAS

Aprendiendo arduino (s.f.). *Entradas y Salidas Analógicas Arduino PWM*. Recuperado el 09/10/2018 de http://cort.as/-LC-n.

Aprendiendo arduino (s.f.). *Que es arduino*. Recuperado el 09/10/2018 de http://cort.as/-LC-j.

BricoGeek. (2019). *Motor paso a paso 3.2 Kg/cm, Nema 17*. Recuperado el 09/10/2018 de http://cort.as/-LC-q.

Control Class. (2018). *Tema 6. Diseño de controladores discretos*. [archivo PDF]. Recuperado el 17/10/2018 de http://cort.as/-LUre.

Core Electronics. (2017). *Our Arduino IDE Tutorial*. [Figura]. Recuperado el 20/10/2018 de http://cort.as/-LbuJ.

ETSETB. (2012). *MATLAB. Fundamentos y/o Aplicaciones*. [archivo PDF]. Recuperado el 15/10/2018 de http://cort.as/-LEKw.

Ingeniería Electrónica. org. (2015). *Tipos de Arduino, detalles y diferencias entre las placas*. Recuperado el 10/10/2018 de http://cort.as/-LC-f.

La tecnología moderna. (2019). *Tipos de arduino y sus características*. Recuperado el 10/10/2018 de http://cort.as/-LC-c.

López, J.F. (2019). *Coeficiente de determinación (R cuadrado)*. Recuperado el 28/02/2019 de http://cort.as/-HjlO.

Mathworks. (s.f.). *Arduino programming in MATLAB-Simulink*. Recuperado el 18/10/2018 de http://cort.as/-LSBk.

Mathworks. (s.f.). *Matlab. The Language of Technical Computing*. Recuperado el 12/06/2019 de http://cort.as/-LBzt.

Mathworks. (s.f.). *PID Controller*. Recuperado el 15/10/2018 de http://cort.as/-LC-y.

Mathworks. (s.f.). *S-function builder*. Recuperado el 15/10/2018 de http://cort.as/-LC-I.

Microelectronicos. (2016). *Especificaciones de la placa Arduino Mega*. [archivo PDF]. Recuperado el 15/10/2018 de http://cort.as/-LELy.

Naylamp mechatronics. (2019). *Sensor Infrarrojo de distancia SHARP GP2Y0A21*. Recuperado el 16/10/2018 de http://cort.as/-LC-V.

Naylamp mechatronics (2019). *Sensor Ultrasonido HC-SR04*. Recuperado el 16/10/2018 de http://cort.as/-LC-r.

Naylamp mechatronics (2019). *Sensor Ultrasonido Analógico US-016*. Recuperado el 16/10/2018 de http://cort.as/-LC-s.

Naylamp mechatronics. (2019). *Servo S3003. Modelo SERVO-04K*. Recuperado el 16/10/2018 de http://cort.as/-LC-O.

Naylamp mechatronics. (2019). *Tutorial Sensor de Distancia SHARP*. Recuperado el 28/02/2019 de http://cort.as/-LC-X.

Naylamp mechatronics. (2019). *Tutorial uso de servomotores con arduino*. Recuperado el 14/03/2019 de http://cort.as/-LC-S.

Oniram de Aquino, J. (2014). *Control of the Ball and Beam using Kalman Filter*. [Figura]. Recuperado el 20/10/2018 de http://cort.as/-LEIL.

Picuino. (2019). *Controlador PID*. Recuperado el 20/10/2018 de http://cort.as/-LC-w.

R. González, V. (2002-03). *Servomotores*. Recuperado el 14/03/2019 de http://cort.as/-LC-U.

Sourceforge. (2019). *Servo signal*. Recuperado el 14/03/2019 de http://cort.as/-LC-W.

SparkFun Electronics. (2019). *Components. GP2Y0A21YK*. [archivo PDF]. Recuperado el 17/10/2018 de http://cort.as/-LEYW.

The Engineering Projects (2019). *Introduction to Arduino Mega 2560*. [Figura]. Recuperado el 15/07/2019 de http://cort.as/-LGZn.

Todo-3D.com. (2017). *FDM-FFF o modelado pro deposición fundida*. Recuperado el 21/01/2019 de http://cort.as/-LC_0.

Wikipedia. (2019). *3D printing filament*. Recuperado el 22/01/2019 de http://cort.as/-LC_4.

Wikipedia (2019). Ácido poliláctico. Recuperado el 22/01/2019 de http://cort.as/-LC_7.

Wikipedia. (2019). Acrilonitrilo butadieno estireno. Recuperado el 22/01/2019 de http://cort.as/-LC_9.

Wikipedia. (2019). *Servo control*. Recuperado el 16/07/2019 de http://cort.as/-LKIg.

ÍNDICE

1. ANEXO I

1.1 PLANOS DEL CONJUNTO

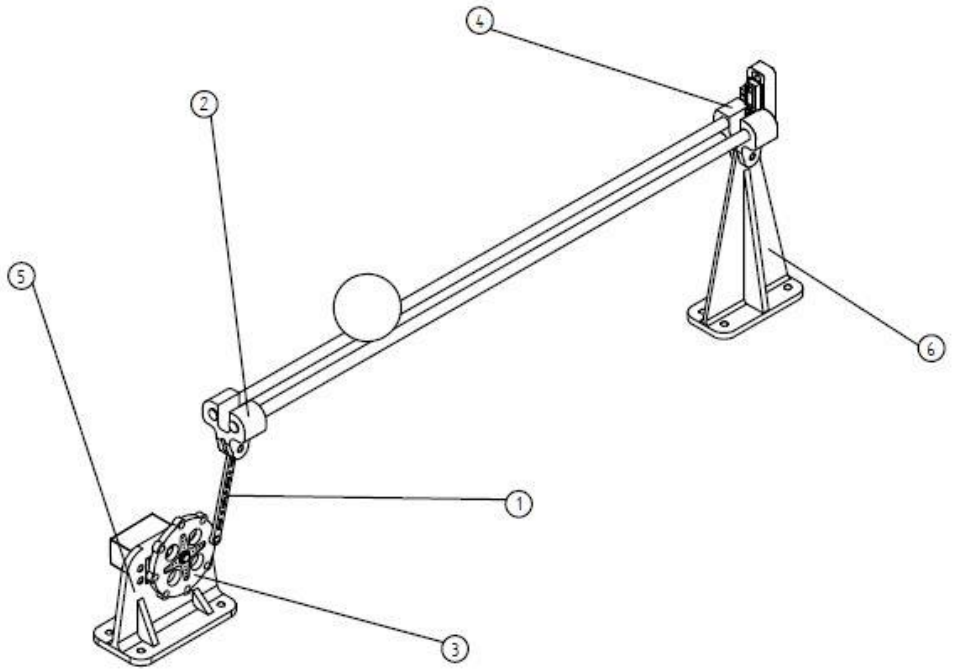
En este apartado se recogen los planos de las siguientes piezas diseñadas en 3D:

Tabla 14.1. Piezas del conjunto

Núm. plano	Descripción	Formato	Escala
C01	Conjunto	DIN A4	1:5
P01	Manivela	DIN A4	2:1
P02	Soporte	DIN A4	2:1
P03	Disco Ranurada	DIN A4	2:1
P04	Base Sensor	DIN A4	3:4
P05	Base Servomotor	DIN A4	1:1
P06	Soporte Sensor	DIN A4	1:1

Fuente. Elaboración propia.

Vista isométrica
Escala 1:5



Cantidad	Denominación	Norma/Plano	Marca	Material
1	Manivela	P01	1	PLA
1	Soporte	P02	2	PLA
1	Disco ranurado	P03	3	PLA
1	Base sensor	P04	4	PLA
1	Base servo motor	P05	5	PLA
1	Soporte sensor	P06	6	PLA

	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

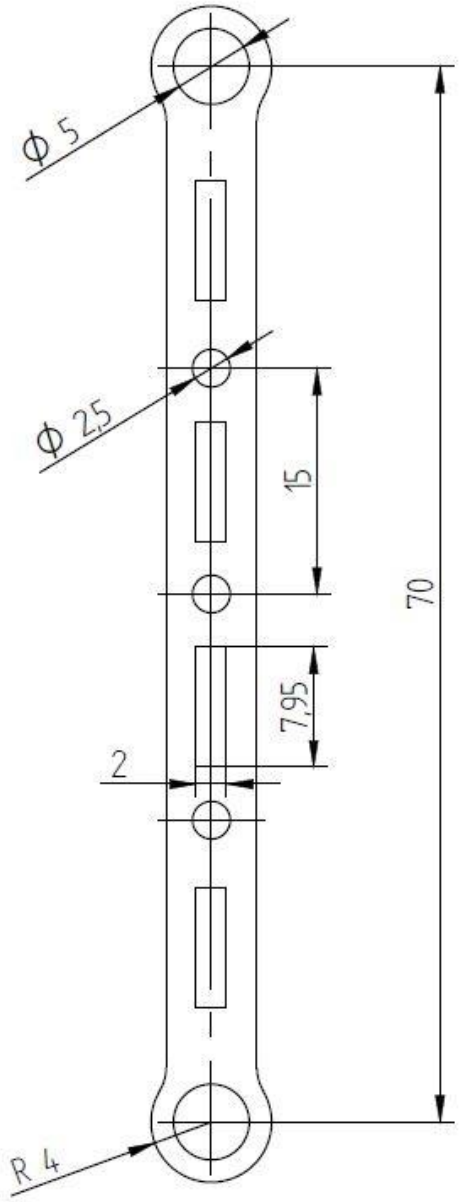
UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA
INDUSTRIAL Y AUTOMÁTICA

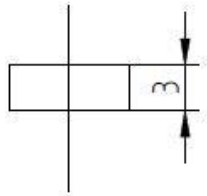
Escala:
1:5

CONJUNTO

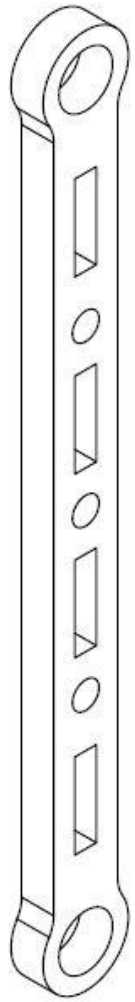
SISTEMA BALL & BEAM

Plano Nº: C01
Plano Cant.: 1/7
Calificación:





Vista isométrica
Escala 2:1



	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

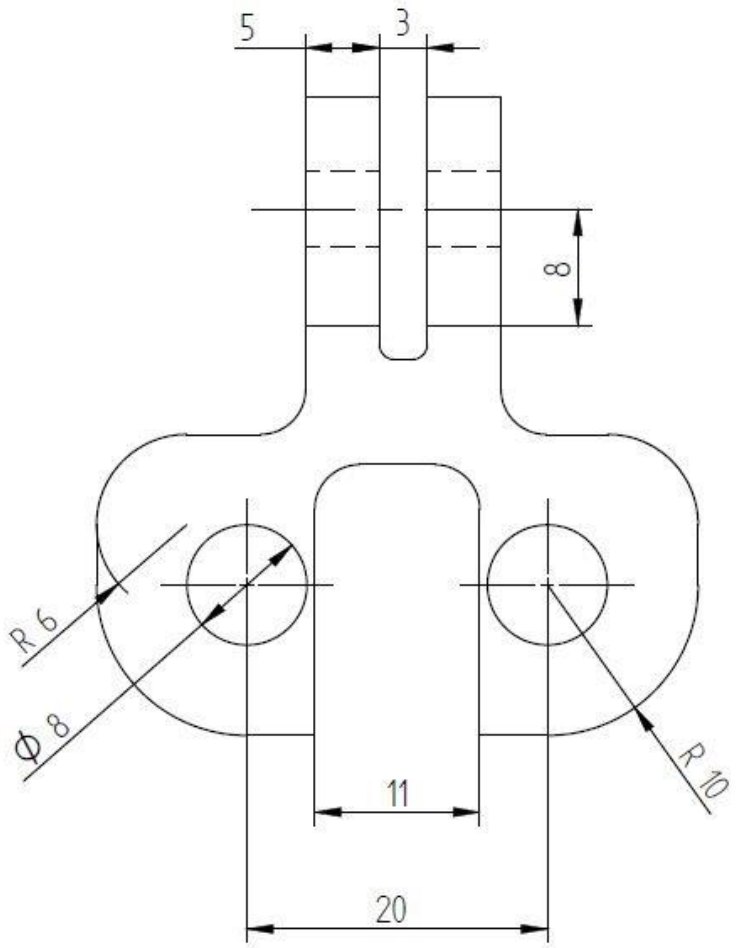
UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

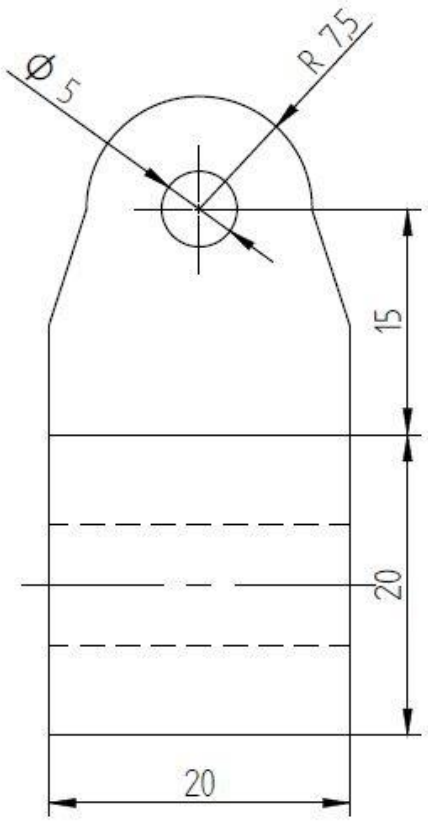
Escala: 2:1

MANIVELA

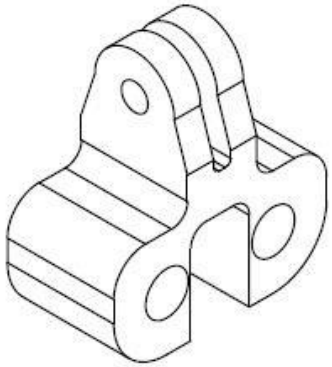
SISTEMA BALL & BEAM

Plano Nº: P01
Plano Cant.: 2/7
Calificación:





Vista isométrica
Escala 1:1



	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

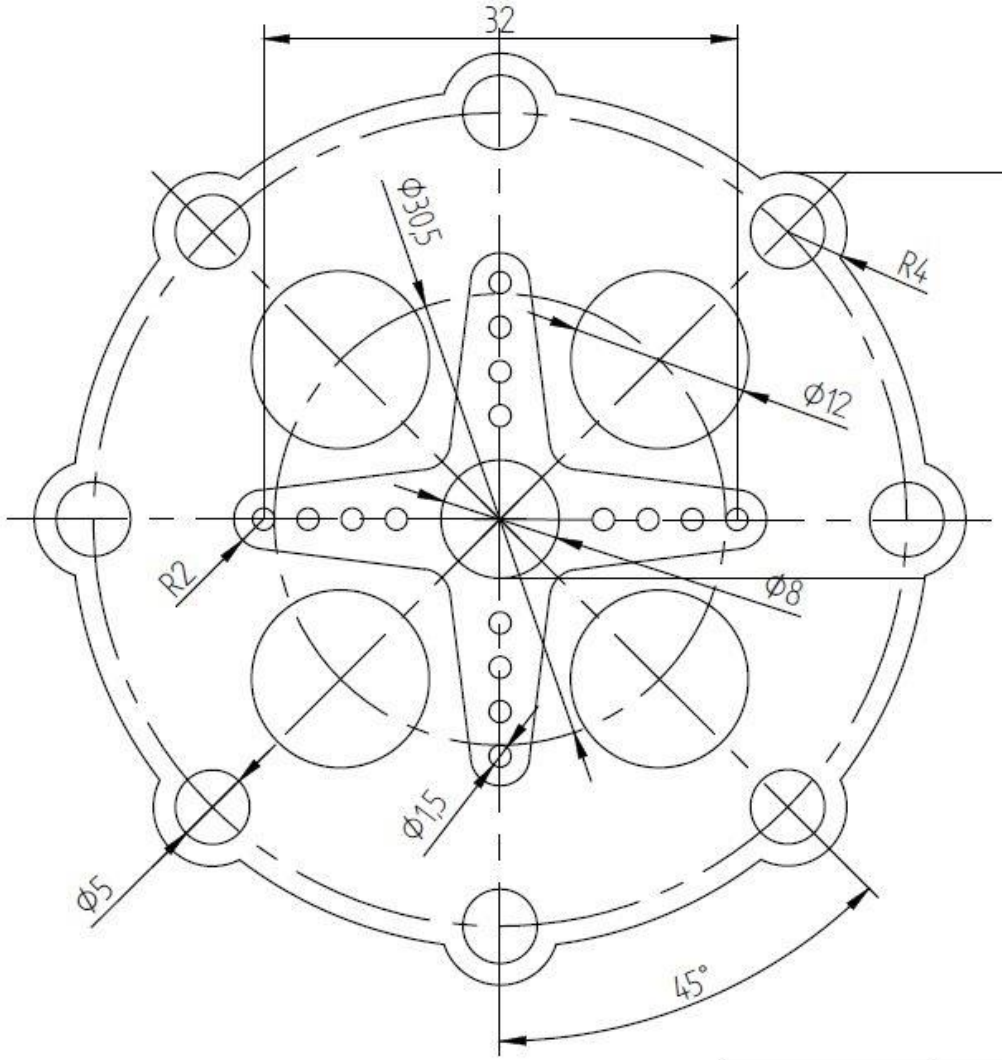
UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

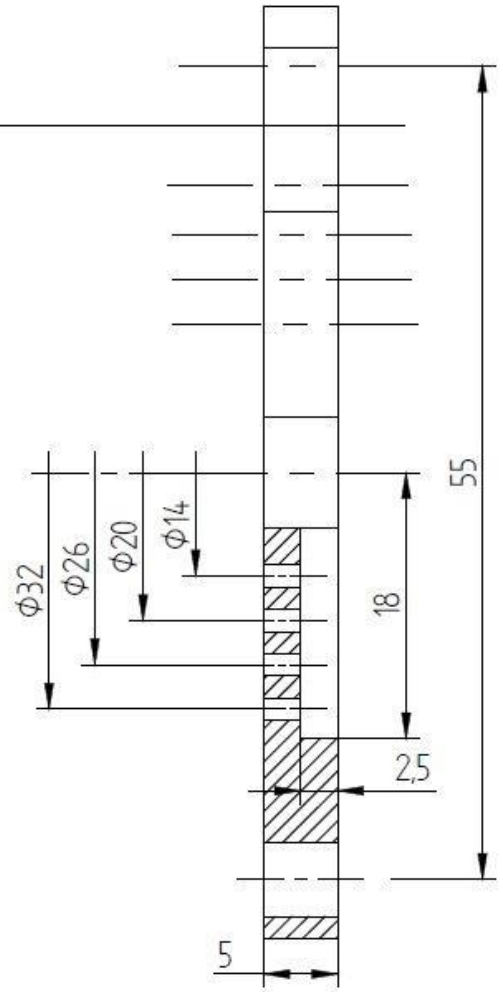
Escala: 2:1

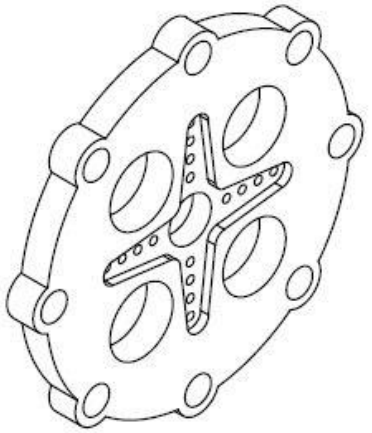
SOPORTE

SISTEMA BALL & BEAM

Plano Nº: P02
Plano Cant.: 3/7
Calificación:







Vista isométrica
Escala 1:1

	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

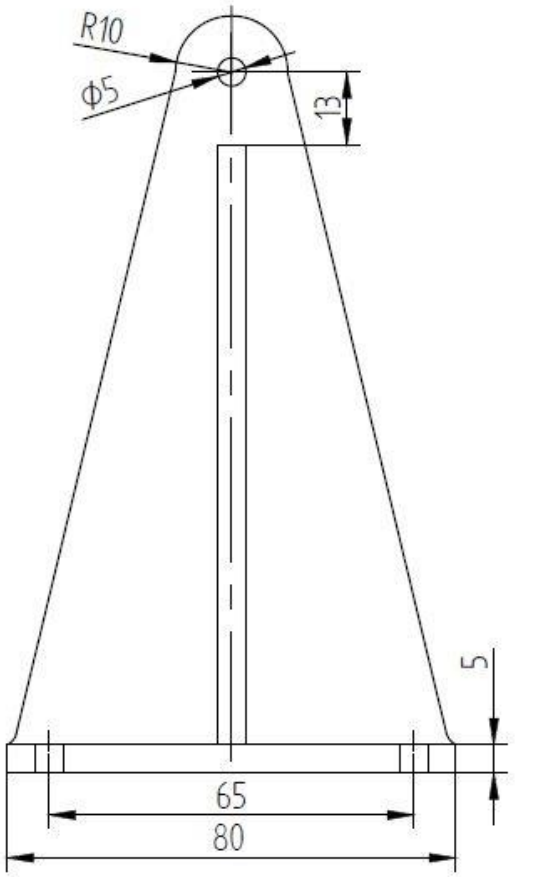
UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

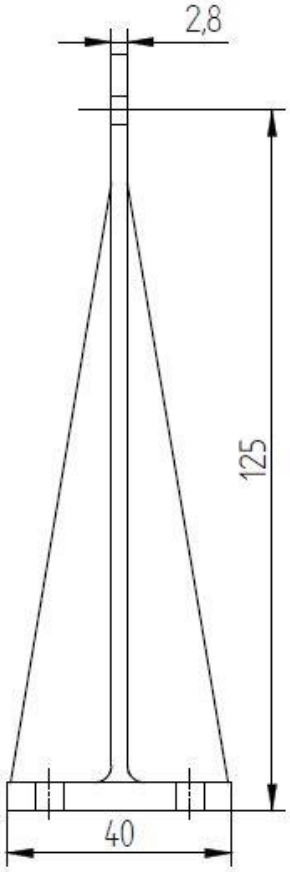
Escala: 2:1

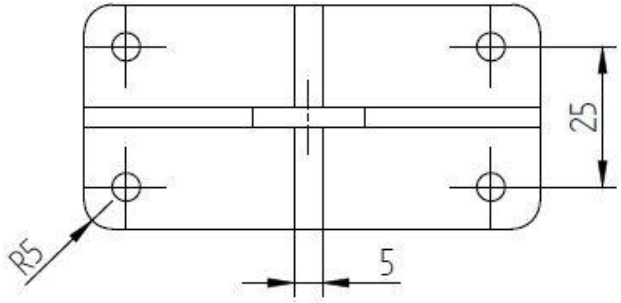
DISCO RANURADO

SISTEMA BALL & BEAM

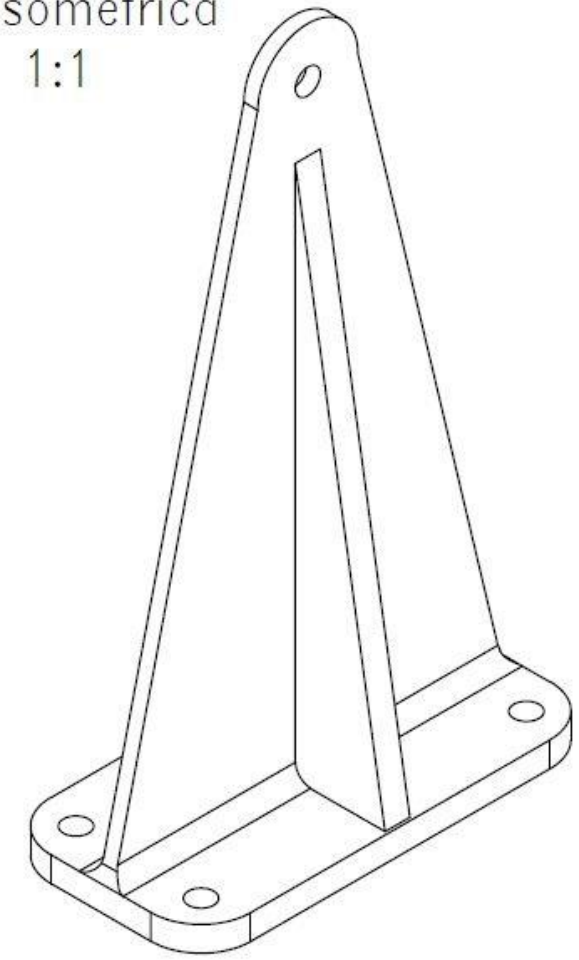
Plano Nº: P03
Plano Cant.: 4/7
Calificación:







Vista isométrica
Escala 1:1



	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

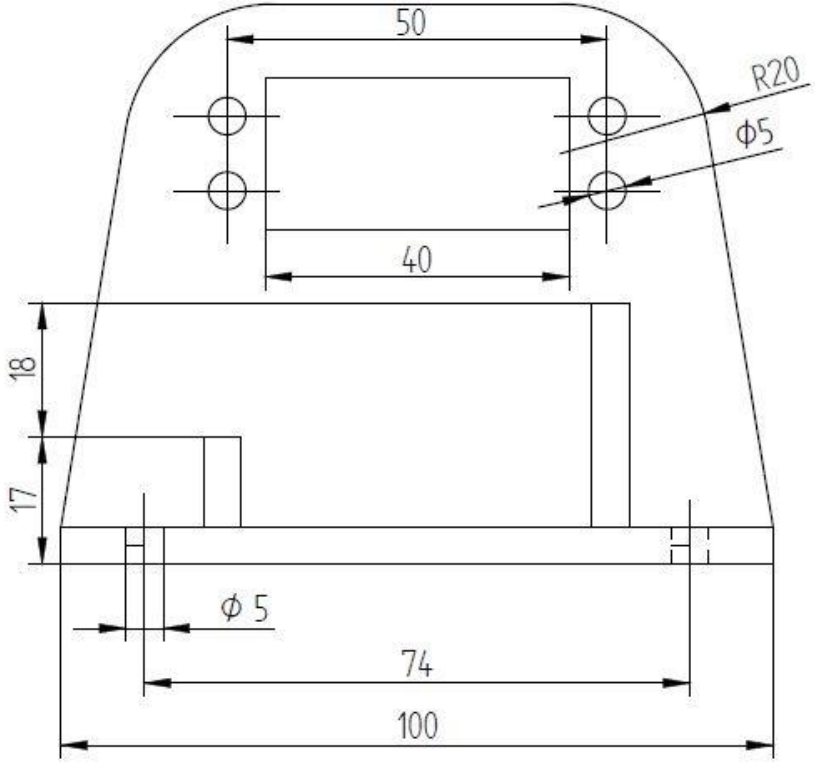
UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

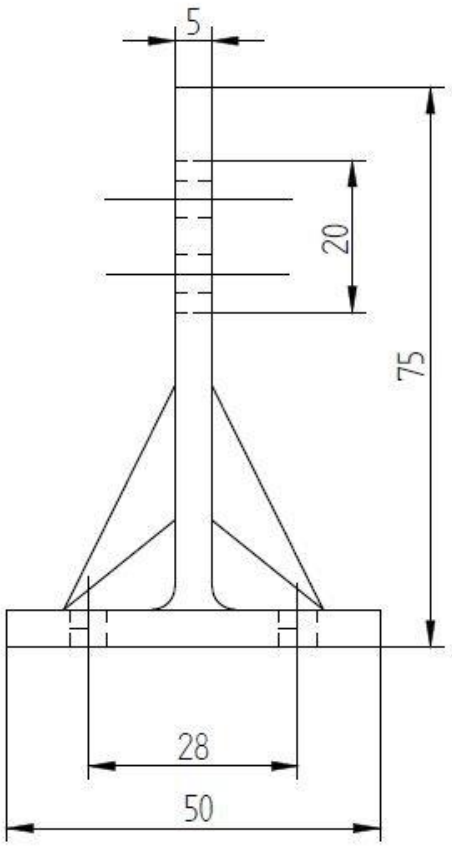
Escala: 3:4

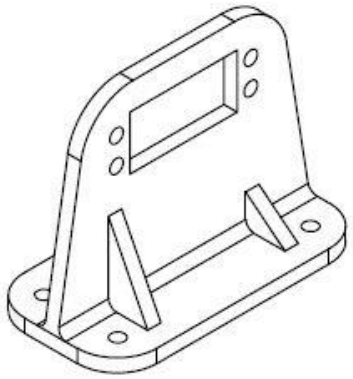
BASE SENSOR

SISTEMA BALL & BEAM

Plano Nº: P04
Plano Cant.: 5/7
Calificación:







Vista isométrica
Escala 1:2

	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

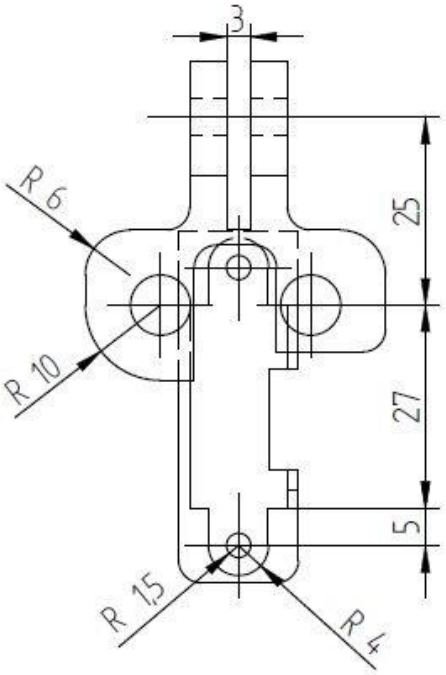
UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

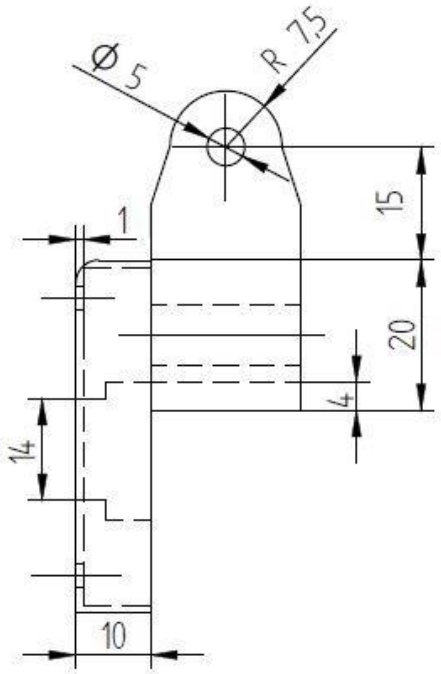
Escala: 1:1

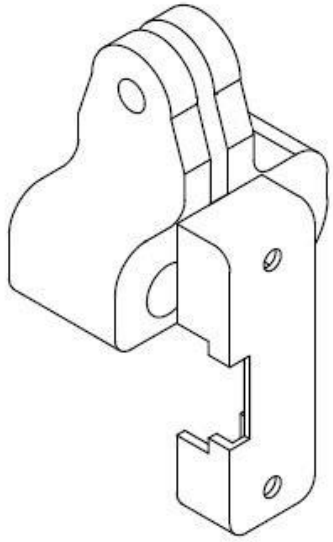
BASE SERVOMOTOR

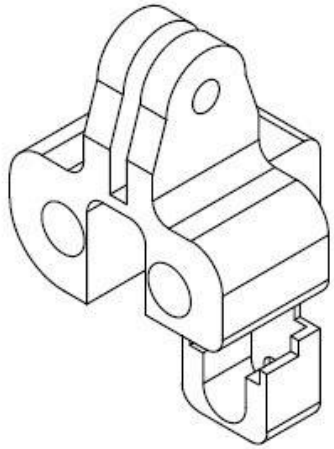
SISTEMA BALL & BEAM

Plano Nº: P05
Plano Cant.: 6/7
Calificación:

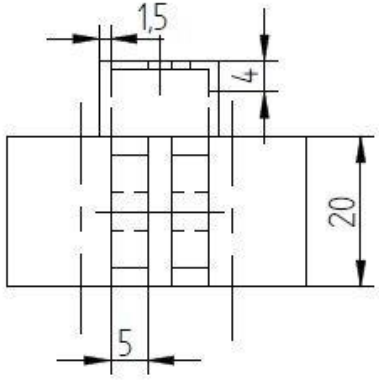








Vista isométrica
Escala 1:1



	Fecha:	Nombre:	Firma:
Dibujado:	12/05/2019	BRENLLA, HENAR	
Comprob.:	13/05/2019	BRENLLA, HENAR	
Dirigido:	12/05/2019	CASQUERO, OSKAR	

UNIVERSIDAD DE PAIS VASCO
EUSKAL HERRIKO UNIBETSITATEA
ESCUELA DE INGENIERÍA DE BILBAO
BILBOKO INGENIARITZA ESKOLA
GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Escala: 1:1

SOPORTE SENSOR

SISTEMA BALL & BEAM

Plano Nº: P06
Plano Cant.: 7/7
Calificación:

1.2 ESQUEMA DE CONEXIONES: ARDUINO MEGA

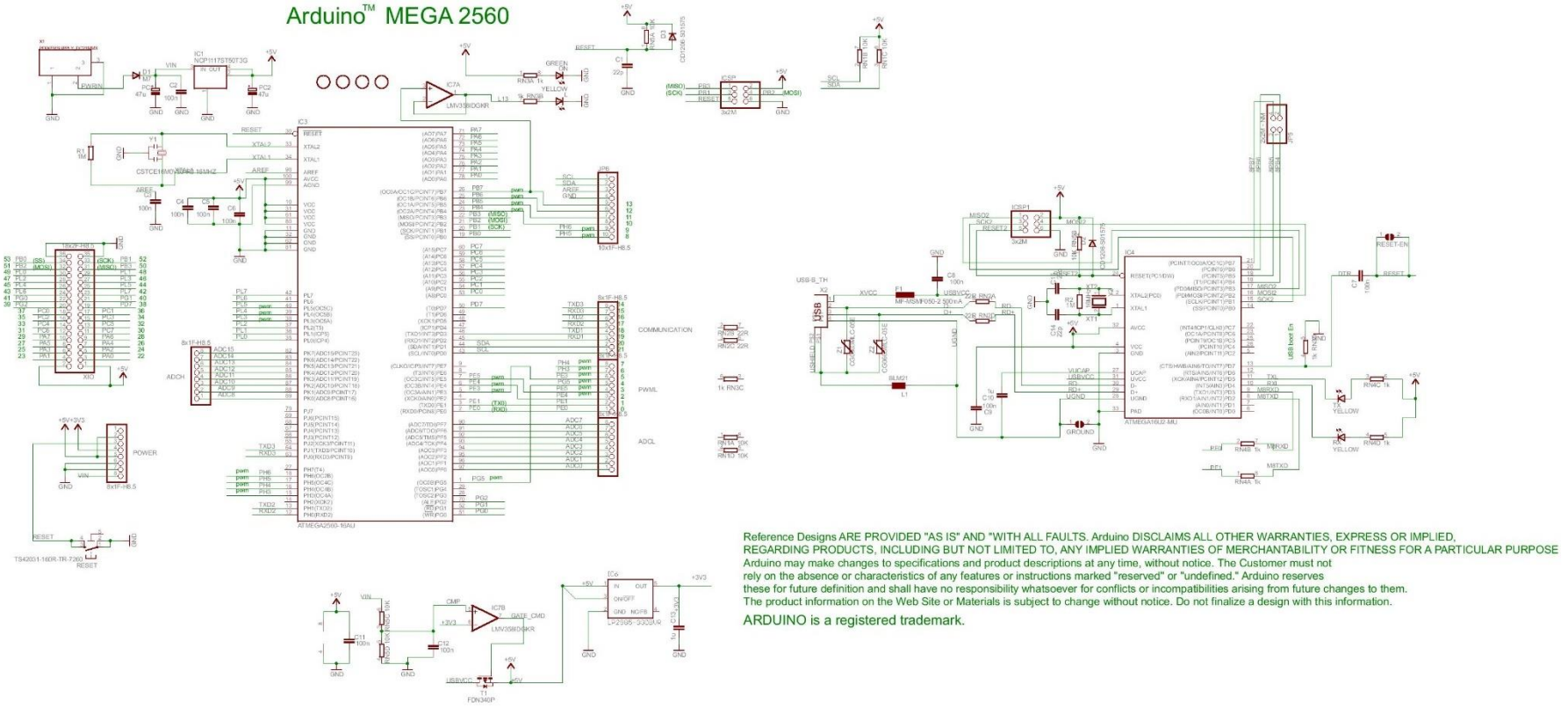


Figura 1.1. Esquema de conexiones de la placa Arduino Mega.

Fuente. http://cort.as/-LELy

1.3 CÓDIGO

En este apartado se encuentra el link al repositorio del proyecto en GitHub donde se encuentra lo siguiente: el código del programa en Arduino, los modelos de bloques de Simulink y los resultados de la sintonización del PID.

https://github.com/HenarB/controldedistanciapelota.git

2. ANEXO II

2.1 DATASHEET DEL SENSOR

GP2Y0A21YK/ GP2Y0D21YK

General Purpose Type Distance Measuring Sensors

■ Features

1. Less influence on the color of reflective objects, reflectivity
2. Line-up of distance output/distance judgement type
   Distance output type (analog voltage) : **GP2Y0A21YK**
   Detecting distance : 10 to 80cm
   Distance judgement type : **GP2Y0D21YK**
   Judgement distance : 24cm
   (Adjustable within the range of 10 to 80cm [Optionally available])
3. External control circuit is unnecessary
4. Low cost

■ Applications

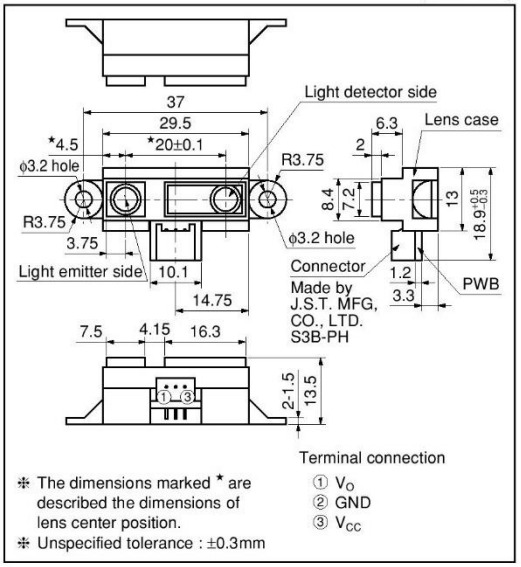
1. TVs
2. Personal computers
3. Cars
4. Copiers

■ Absolute Maximum Ratings (T_a=25°C, V_{CC}=5V)

Parameter	Symbol	Rating	Unit
Supply voltage	V_{CC}	−0.3 to +7	V
Output terminal voltage	V_O	−0.3 to V_{CC} +0.3	V
Operating temperature	T_{opr}	−10 to +60	°C
Storage temperature	T_{stg}	−40 to +70	°C

■ Outline Dimensions

(Unit : mm)



❊ The dimensions marked * are described the dimensions of lens center position.
❊ Unspecified tolerance : ±0.3mm

■ Recommended Operating Conditions

Parameter	Symbol	Rating	Unit
Operating supply voltage	V_{CC}	4.5 to +5.5	V

■ Electro-optical Characteristics

(T_a=25°C, V_{CC}=5V)

Parameter		Symbol	Conditions	MIN.	TYP.	MAX.	Unit
Distance measuring range		ΔL	*1 *3	10	–	80	cm
Output terminal voltage	GP2Y0A21YK	V_O	L=80cm *1	0.25	0.4	0.55	V
	GP2Y0D21YK	V_{OH}	Output voltage at High *1	V_{CC} –0.3	–	–	V
		V_{OL}	Output voltage at Low *1	–	–	0.6	V
Difference of output voltage	GP2Y0A21YK	ΔV_O	Output change at L=80cm to 10cm *1	1.65	1.9	2.15	V
Distance characteristics of output	GP2Y0D21YK	V_O	*1 *4 *2	21	24	27	cm
Average Dissipation current		I_{CC}	L=80cm *1	–	30	40	mA

Note) L : Distance to reflective object
*1 Using reflective object : White paper (Made by Kodak Co. Ltd. gray cards R-27 · white face, reflective ratio ; 90%)
*2 We ship the device after the following adjustment : Output switching distance L=24cm±3cm must be measured by the sensor
*3 Distance measuring range of the optical sensor system
*4 Output switching has a hysteresis width. The distance specified by Vo should be the one with which the output L switches to the output H

Fig.1 Internal Block Diagram

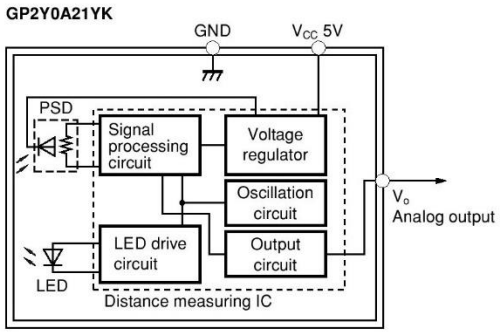


Fig.2 Internal Block Diagram

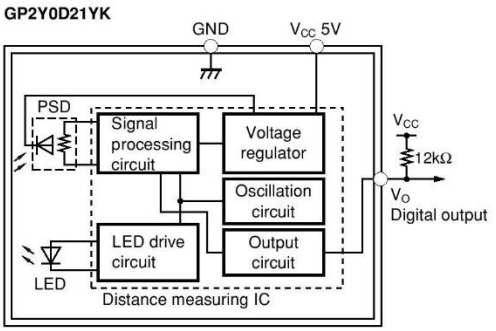


Fig.3 Timing Chart

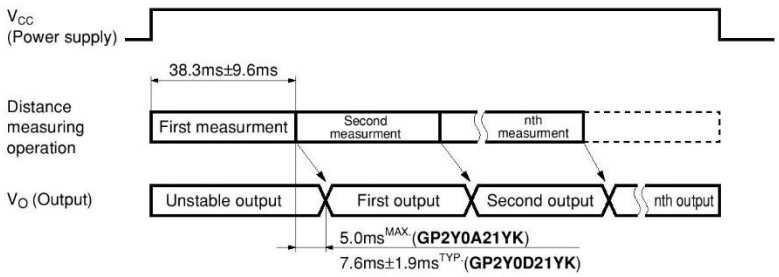


Fig.4 Distance Characteristics

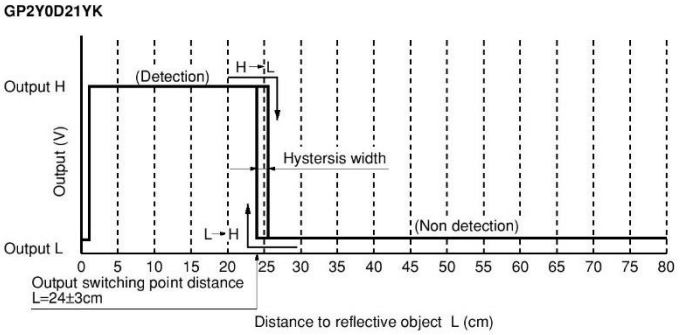
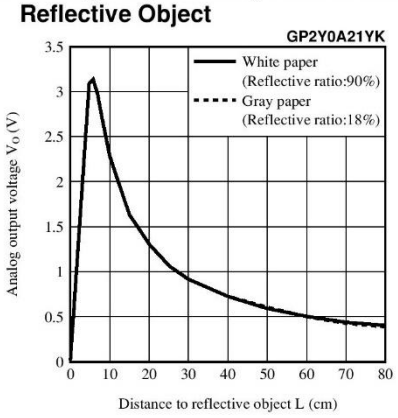


Fig.5 Analog Output Voltage vs. Distance to Reflective Object



NOTICE

- The circuit application examples in this publication are provided to explain representative applications of SHARP devices and are not intended to guarantee any circuit design or license any intellectual property rights. SHARP takes no responsibility for any problems related to any intellectual property right of a third party resulting from the use of SHARP's devices.
- Contact SHARP in order to obtain the latest device specification sheets before using any SHARP device. SHARP reserves the right to make changes in the specifications, characteristics, data, materials, structure, and other contents described herein at any time without notice in order to improve design or reliability. Manufacturing locations are also subject to change without notice.
- Observe the following points when using any devices in this publication. SHARP takes no responsibility for damage caused by improper use of the devices which does not meet the conditions and absolute maximum ratings to be used specified in the relevant specification sheet nor meet the following conditions:

  (i) The devices in this publication are designed for use in general electronic equipment designs such as:
  --- Personal computers
  --- Office automation equipment
  --- Telecommunication equipment [terminal]
  --- Test and measurement equipment
  --- Industrial control
  --- Audio visual equipment
  --- Consumer electronics

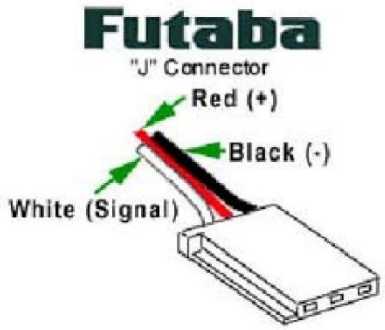
  (ii) Measures such as fail-safe function and redundant design should be taken to ensure reliability and safety when SHARP devices are used for or in connection with equipment that requires higher reliability such as:
  --- Transportation control and safety equipment (i.e., aircraft, trains, automobiles, etc.)
  --- Traffic signals
  --- Gas leakage sensor breakers
  --- Alarm equipment
  --- Various safety devices, etc.

  (iii) SHARP devices shall not be used for or in connection with equipment that requires an extremely high level of reliability and safety such as:
  --- Space applications
  --- Telecommunication equipment [trunk lines]
  --- Nuclear power control equipment
  --- Medical and other life support equipment (e.g., scuba).

- Contact a SHARP representative in advance when intending to use SHARP devices for any "specific" applications other than those recommended by SHARP or when it is unclear which category mentioned above controls the intended use.
- If the SHARP devices listed in this publication fall within the scope of strategic products described in the Foreign Exchange and Foreign Trade Control Law of Japan, it is necessary to obtain approval to export such SHARP devices.
- This publication is the proprietary product of SHARP and is copyrighted, with all rights reserved. Under the copyright laws, no part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, in whole or in part, without the express written permission of SHARP. Express written permission is also required before any use of this publication may be made by a third party.
- Contact and consult with a SHARP representative if there are any questions about the contents of this publication.

2.2 DATASHEET DEL SERVOMOTOR

S3003 FUTABA SERVO



...S3003 FUTABA SERVO...

Detailed Specifications

Control System:	+Pulse Width Control 1520usec Neutral	Current Drain (4.8V):	7.2mA/idle
Required Pulse:	3-5 Volt Peak to Peak Square Wave	Current Drain (6.0V):	8mA/idle
Operating Voltage:	4.8-6.0 Volts	Direction:	Counter Clockwise/Pulse Traveling 1520-1900usec
Operating Temperature Range:	-20 to +60 Degree C	Motor Type:	3 Pole Ferrite
Operating Speed (4.8V):	0.23sec/60 degrees at no load	Potentiometer Drive:	Indirect Drive
Operating Speed (6.0V):	0.19sec/60 degrees at no load	Bearing Type:	Plastic Bearing
Stall Torque (4.8V):	44 oz/in. (3.2kg.cm)	Gear Type:	All Nylon Gears
Stall Torque (6.0V):	56.8 oz/in. (4.1kg.cm)	Connector Wire Length:	12"
Operating Angle:	45 Deg. one side pulse traveling 400usec	Dimensions:	1.6" x 0.8"x 1.4" (41 x 20 x 36mm)
360 Modifiable:	Yes	Weight:	1.3oz. (37.2g)

3. ANEXO III

3.1 SW UTILIZADO

Fritzing: Programa de código abierto de diseño de circuitos electrónicos. Se ha utilizado para crear el esquema electrónico.

AutoCAD 2019: SW de diseño gráfico en 2 dimensiones. Se ha utilizado para crear los planos de las piezas de la maqueta.

Autodesk Inventor 2019: SW de diseño gráfico en 3 dimensiones. Se ha utilizado para crear los modelos en 3D de variar piezas de la maqueta.

KeyShot 7: SW para renderizar en 3 dimensiones. Se ha utilizado para crear los modelos en 3D de variar piezas de la maqueta.

Draw.io: Programa de código abierto para crear diagramas. Se ha utilizado para crear todos los diagramas del proyecto.

Matlab: Programa de cálculo numérico que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Se ha utilizado para crear el modelo de bloques en Simulink y ejecutar el sistema.

Arduino IDE: Programa para programar Arduino en lenguaje C. Se ha utilizado para programar la máquina de quemado.

GanttProject: Software de código abierto de creación de diagramas Gantt. Se ha usado para crear el diagrama Gantt del proyecto.

Excel: Programa de hojas de cálculo. Se ha utilizado para crear el gráfico de regresión polinómica y el presupuesto.

Cura Ultimaker: Programa de código abierto para el procesado de modelos 3D para su impresión.

Word: Programa de procesamiento de textos. Se ha utilizado para redactar todo el proyecto.