Contenido

[1. INTRODUCCIÓN 3](#_Toc536028343)

[1.1. Objetivo 3](#_Toc536028344)

[1.2. Hardware empleado 3](#_Toc536028345)

[1.2.1. Arduino Leonardo 3](#_Toc536028346)

[1.2.2. Sensores 4](#_Toc536028347)

[1.2.3. Actuadores 4](#_Toc536028348)

[1.2.4. Elementos de comunicación 4](#_Toc536028349)

[1.2.5. Alimentación 4](#_Toc536028350)

[1.3. Software empleado 4](#_Toc536028351)

[1.3.1. Arduino IDE 4](#_Toc536028352)

[1.3.2. Unity 4](#_Toc536028353)

[1.3.3. C# 4](#_Toc536028354)

[2. ESPECIFICACIÓN DE REQUISITOS 5](#_Toc536028355)

[2.1. Requisitos funcionales 5](#_Toc536028356)

[2.2. Requisitos no funcionales 6](#_Toc536028357)

[3. PLANIFICACIÓN 6](#_Toc536028358)

[4. PRESUPUESTO 6](#_Toc536028359)

[5. ANÁLISIS 7](#_Toc536028360)

[5.1. Casos de uso 7](#_Toc536028361)

[7](#_Toc536028362)

[5.2. Diagrama de flujo 8](#_Toc536028363)

[6. DISEÑO 9](#_Toc536028364)

[6.1. Estructura 9](#_Toc536028365)

[6.2. Plan de pruebas 14](#_Toc536028366)

[6.2.1. Test Batería 14](#_Toc536028367)

[6.2.2. Test Bluetooth 15](#_Toc536028368)

[6.2.3. Test CNY70 16](#_Toc536028369)

[6.2.4. Tests Motores 18](#_Toc536028370)

[6.2.5. Test Ultrasonido 20](#_Toc536028371)

[6.2.6. Test Servo 21](#_Toc536028372)

[6.3. Librerías 21](#_Toc536028373)

[6.4. Apuntes sobre el código 21](#_Toc536028374)

[7. MONTAJE 22](#_Toc536028375)

[22](#_Toc536028376)

[8. PRUEBAS 28](#_Toc536028377)

[9. MEJORAS 29](#_Toc536028378)

# INTRODUCCIÓN

## Objetivo

Robot móvil capaz de recorrer un laberinto de 5x5 celdas tratando de encontrar la salida. Las acciones llevadas a cabo por el robot para conseguir su objetivo son:

* Movimientos:
  + Avanzar: Avanza una casilla.
  + Giro Izquierda: Gira 90º a la izquierda.
  + Giro Derecha: Gira 90º a la derecha.
  + Giro 180º: Gira 180º para dar la vuelta.
* Detección:
  + Ultrasonido: Detectará si hay caminos libres.
  + CNY70: Detectará cuando ha pasado por una línea negra y corregirá su rumbo, si en los cuatro sensores detecta negro, ha llegado al final.
* Recogida de información: No se recoge ningún tipo de información.
* Algoritmo de resolución del laberinto: Siempre que haya camino a la derecha, lo cogerá, si no, el del centro y por último el de la izquierda. SI no encuentra camino gira 180º y vuelve a verificar si tiene camino.
* Monitorización de información: Tiene comunicación bluetooth indicando todos sus movimientos y si se encuentra muros con una aplicación Unity la cuál se encargará de dibujar en 3 dimensiones el recorrido que realiza el robot y de colocar los muros que se encuentre.

## Hardware empleado

* **Switch**: Para encender y apagar el robot. Es el que viene incluido con la placa.
* **CNY70**: Para detectar cuando pasa el robot por una franja negra y poder redirigirse y para poder reconocer si está en la casilla final.
* **Ultrasonido**: Para detectar si hay pared o si hay un camino disponible.
* **DC Controlador**: Controladores para los motores. Se utiliza para realizar los movimientos de los motores. Vienen incluidos con la placa.
* **DC Motor**: Motores de corriente continua para mover el robot.
* **Servo**: Sirve para mover el ultrasonido y así poder comprobar si hay camino disponible, en la derecha, en frente o a la izquierda del robot.
* **Bluetooth**: Sirve para poder realizar la comunicación de información con el PC.

## Arduino Leonardo

Para ello, se utilizará una placa basado en microcontrolador, Arduino Leonardo, y un ordenador personal.

**Arduino Leonardo** es una placa microcontrolador basada en ATmega32u4. Tiene 20 pines digitales de entrada y salida (Los cuales 7 de ellos pueden ser utilizados como salidas PWM y 12 de ellos como entradas analógicas), un oscilador de cristal de 16MHz, una conexión micro USB, un conector de alimentación, un ICSP header, y un botón de reset.

## Sensores

|  |  |  |  |
| --- | --- | --- | --- |
| Sensores | Input/Output | Tipo de pin | Pin |
| 1 Ultrasonido | Input | Analógico | A3(Trigger),A4(Echo) |
| 4 CNY70 | Input | Analógico | A0,A1,A2,A5 |

## Actuadores

|  |  |  |  |
| --- | --- | --- | --- |
| Actuadores | Input/Output | Tipo de pin | Pin |
| DC Controlador Der. | Output | PWM | 5 |
| DC Controlador Izq. | Output | PWM | 6 |
| DC Motor Der. | Output | --- | --- |
| DC Motor Izq. | Output | --- | --- |
| 1 Servo | Output | Digital | 11 |
| 1 Bluetooth | Output | Serial1 (TX/RX) | 0, 1 |

## Elementos de comunicación

* Bluetooth.

## Alimentación

* Batería de 6 pilas tipo AA.

## Software empleado

## Arduino IDE

Es un software que hace que sea fácil de escribir código y subirlo a la placa, se puede ejecutar en Windows, Mac OS X y Linux, el entorno está escrito en Java y basado en el procesamiento y el otro software de código abierto.

Este software se puede utilizar con cualquier placa Arduino.

## Unity

Unity es un motor de videojuego multiplataforma creado por Unity Technologies. Unity está disponible como plataforma de desarrollo para Microsoft Windows, OS X, Linux. La plataforma de desarrollo tiene soporte de compilación con diferentes tipos de plataformas, una de ellas que utilizamos es C#. Utilizaremos Unity para mostrar el entorno visual en 3 dimensiones del laberinto.

## C#

C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET, que después fue aprobado como un estándar por la ECMA (ECMA-334) e ISO (ISO/IEC 23270). C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común.

Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Utilizaremos C# para realizar la conexión puerto serie con Arduino y poder establecer una comunicación entre el robot y la aplicación Unity.

# ESPECIFICACIÓN DE REQUISITOS

## Requisitos funcionales

| **ID** | **CATEGORIA** | **DESCRIPCIÓN** |
| --- | --- | --- |
| RF01 | Movimiento | El robot debe ser capaz de moverse |
| RF02 | Movimiento | El robot debe ser capaz de pivotar 90º a derecha |
| RF03 | Movimiento | El robot debe ser capaz de pivotar 90º a izquierda |
| RF04 | Movimiento | El robot debe ser capaz de avanzar en línea recta |
| RF05 | Movimiento | El robot avanza adecuadamente hasta la siguiente celda |
| RF06 | Detección | El robot debe ser capaz de detectar una pared frontal |
| RF07 | Detección | El robot debe ser capaz de detectar una pared lateral derecha |
| RF08 | Detección | El robot debe ser capaz de detectar una pared lateral izquierda |
| RF09 | Detección | El robot debe ser capaz de detectar la transición entre celdas |
| RF10 | Detección | El robot debe ser capaz de detectar la celda de salida |
| RF11 | Resolución | El robot almacena información sobre las celdas del laberinto |
| RF12 | Resolución | El robot es capaz de decidir el siguiente movimiento en base a la información sobre la celda |
| RF13 | Resolución | El robot es capaz de recorrer varias celdas del laberinto siguiendo el algoritmo empleado |
| RF14 | Resolución | El robot es capaz de salir del laberinto |
| RF15 | Información | El robot envía al PC información sobre el número de celdas recorridas |
| RF16 | Información | El robot envía al PC información sobre obstáculos en cada celda |
| RF17 | Información | El robot envía al PC información sobre la velocidad de movimiento |
| RF18 | Información | El robot envía al PC información sobre la distancia que lleva recorrida |
| RF19 | Información | El robot envía al PC información sobre el tiempo transcurrido desde la entrada al laberinto |
| RF20 | Información | El robot envía al PC información sobre la trayectoria ejecutada |
| RF21 | Información | El robot envía al PC información sobre el número de celdas recorridas |
| RF22 | Información | El PC muestra una representación gráfica del laberinto |
| RF23 | Usuario | Interfaz gráfica en PC que recopile información |
| RF24 | Pruebas | Incluir modo test al arranque del robot |

## Requisitos no funcionales

| **ID** | **CATEGORIA** | **DESCRIPCIÓN** | **ACCIÓN** |
| --- | --- | --- | --- |
| RNF01 | Tamaño | Condicionado por las dimensiones del laberinto. El laberinto será de 5x5, altura de paredes de 20cm y ancho de celdas también de 20cm. El robot mide 12cm de largo y 10cm de ancho. |  |
| RNF02 | Consumo | Condicionado por la batería disponible. Batería de 6 pilas tipo AA. |  |
| RNF03 | Errores | El robot choca contra una pared | Se vuelve a colocar en el centro de la casilla. |
| RNF04 | Errores | Batería a punto de agotarse | Cambio de pilas. |
| RNF05 | Errores | El robot gira continuamente | Reinicio del algoritmo y de su posición. |
| RNF06 | Errores | El robot sobrepasa 20 minutos sin conseguir salir | Volver a colocar el robot al inicio del laberinto reiniciando el algoritmo. |

# PLANIFICACIÓN

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Tiempo / Secciones | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Estado Batería |  |  |  |  |  |  |  |  |  |
| Cargar caminos |  |  |  |  |  |  |  |  |  |
| Movimiento servo |  |  |  |  |  |  |  |  |  |
| Ultrasonido |  |  |  |  |  |  |  |  |  |
| Giros |  |  |  |  |  |  |  |  |  |
| Avanzar/Retroceder |  |  |  |  |  |  |  |  |  |
| CNY70 |  |  |  |  |  |  |  |  |  |
| Parar |  |  |  |  |  |  |  |  |  |

# PRESUPUESTO

|  |  |  |  |
| --- | --- | --- | --- |
| PRODUCTO | CANTIDAD | PRECIO UNIDAD | PRECIO TOTAL |
| Arduino leonardo | 1 | 19,90€ | 19,90€ |
| Switch | 1 | 0,95€ | 0,95€ |
| CNY70 | 4 | 1,20€ | 4,80€ |
| Ultrasonido | 1 | 3,95€ | 3,95€ |
| Servo | 1 | 3,80€ | 3,80€ |
| Motor DC | 2 | 13,20€ | 26,40€ |
| Controlador Motor | 1 | 1,20€ | 1,20€ |
| Bluetooth | 1 | 11,75€ | 11,75€ |
| Rueda | 2 | 5,90€ | 11,80€ |
| Cables (10u) | 2 | 2,50€ | 5,00€ |
| Shield | 1 | 1,90€ | 1,90€ |
| **TOTAL** |  |  | **91,45€** |

# ANÁLISIS

## Casos de uso

## 

Una vez iniciada la conexión con el robot:

## Diagrama de flujo

Mira

Paredes

¿Camino der.

Libre?

¿Camino centro

Libre?

¿Camino izq.

Libre?

Gira Izq +

Avanzar

Si

Gira Der +

Avanzar

Si

Si no

Avanzar

Si no

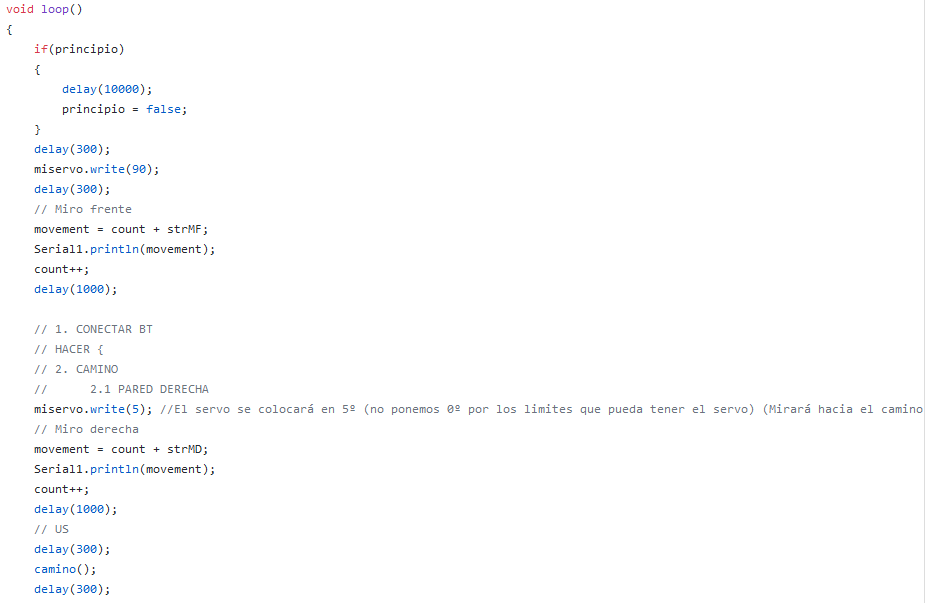
Si

Si no

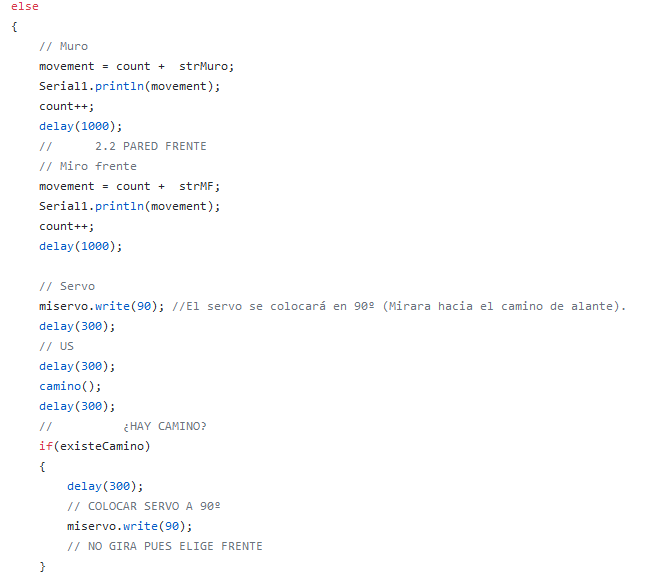
Giro 180º

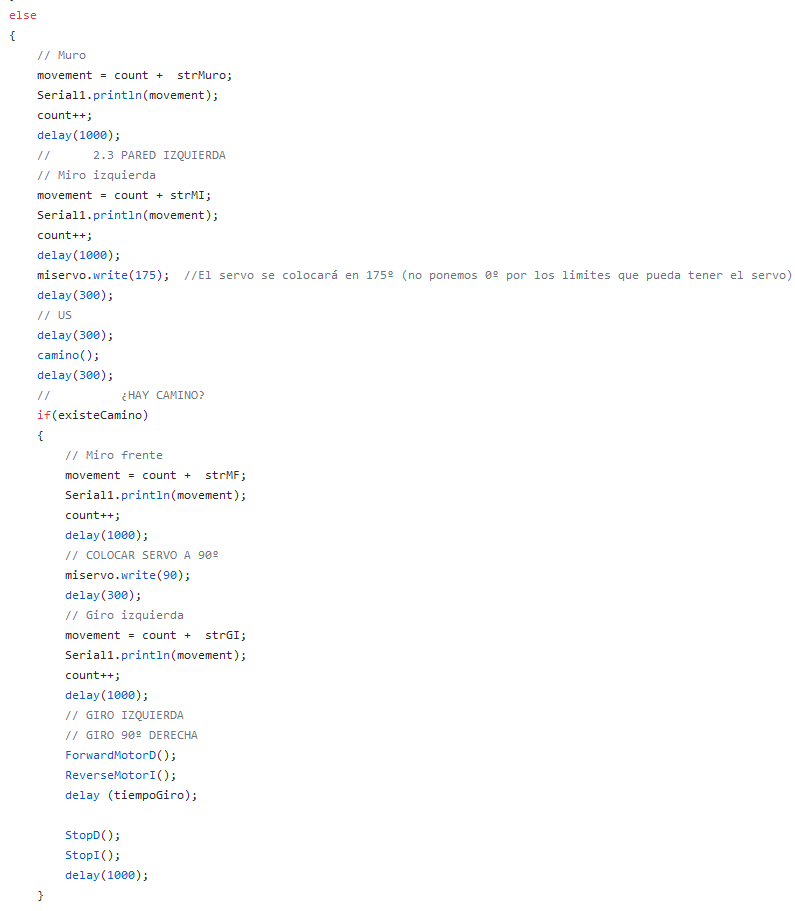
# DISEÑO

## Estructura

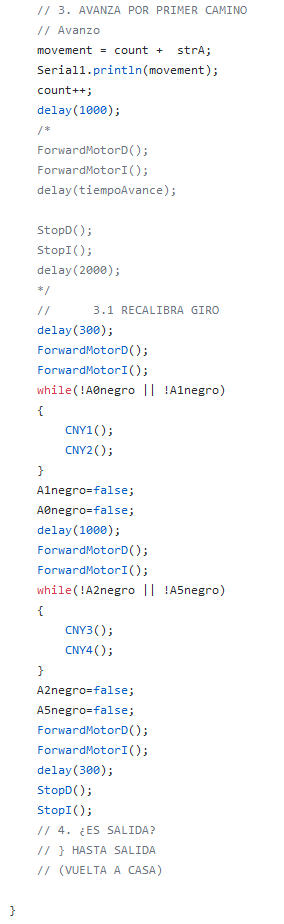










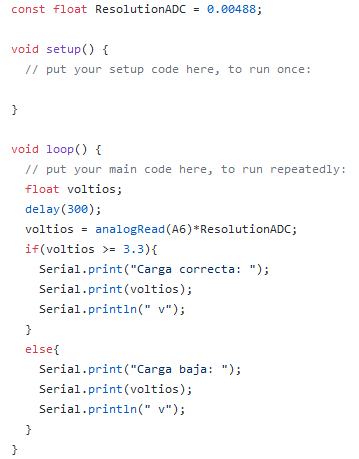


## Plan de pruebas

Estos son los tests realizados para comprobar que cada componente del robot funciona de forma correcta.

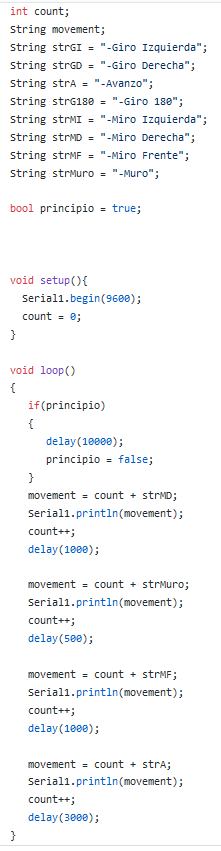
## Test Batería

Comprobamos si la batería tiene suficiente potencia para todos los componentes.



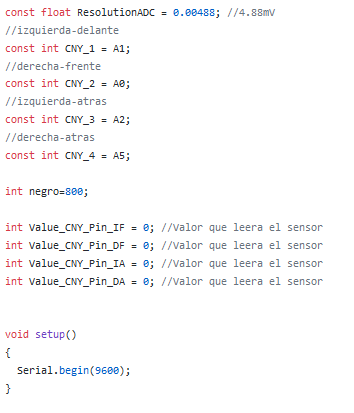
## Test Bluetooth

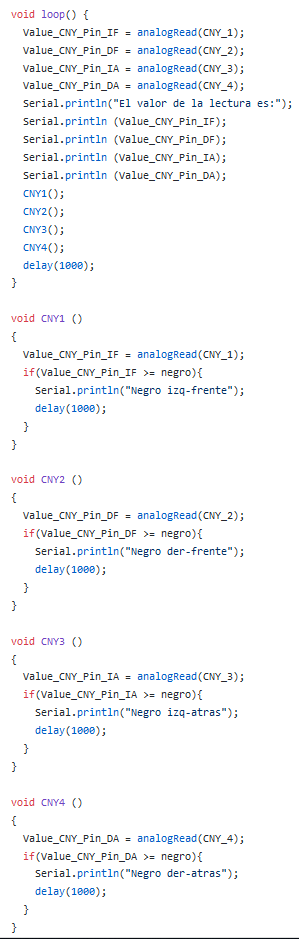
Todos los comandos que utilizamos para darle información a la aplicación Unity y así poder observar en la aplicación los movimientos que realiza el robot.



## Test CNY70

El programa imprime los valores que recoge el sensor y en el caso de que detecte el color negro imprime por consola “línea”.



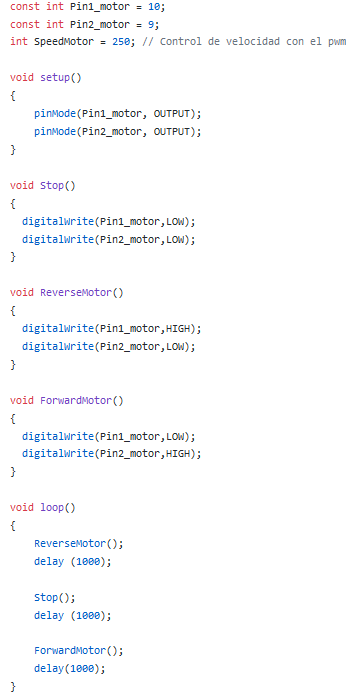


## Tests Motores

Programa de prueba del motor derecho que se encarga de hacer girar al eje en ambos sentidos con esperas de 1s.

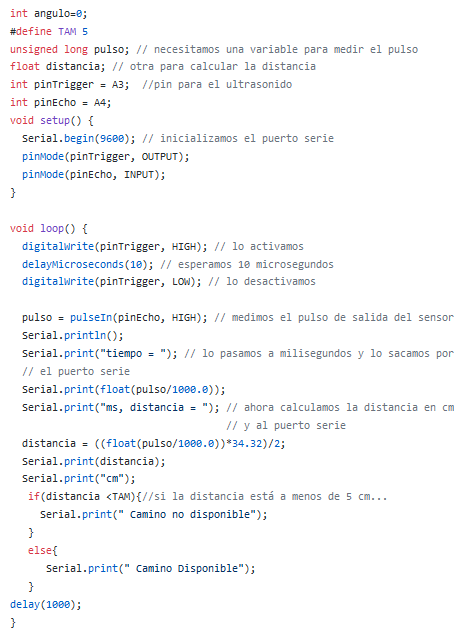


Programa de prueba del motor izquierdo que se encarga de hacer girar al eje en ambos sentidos con esperas de 1s.



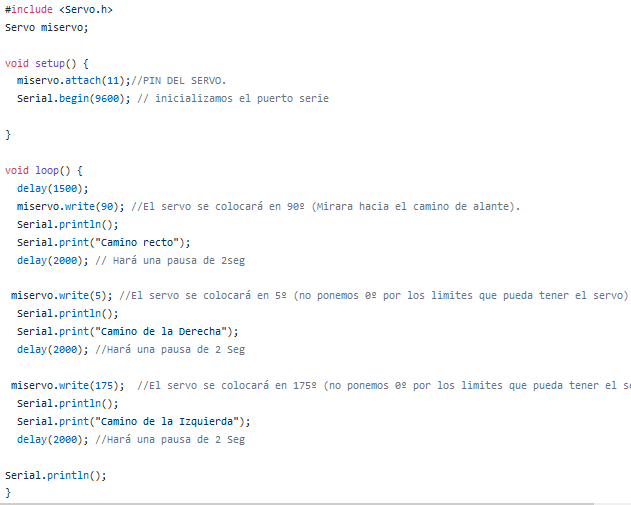
## Test Ultrasonido

Si la pared se encuentra a una posición mayor a 5cm, se muestra por pantalla “Camino disponible”, si se encuentra a menor distancia, mostrará “Camino no disponible”.



## Test Servo

Programa de prueba en el que observamos en el monitor la posición en la que está el servo, éste cambiará su posición cada 2 segundos mostrando por pantalla su nueva posición.



## Librerías

[Utilizamos la librería Servo para el movimiento del servo.](https://www.arduino.cc/en/Reference/Servo)

## Apuntes sobre el código

A la hora de utilizar los motores, teníamos problemas con los pines dado que el servo y los motores (utilizando analog) utilizaban los mismos pines. Pues lo cambiamos y utilizamos los pines digitales para los motores.

En la aplicación Unity había problemas a la hora de recoger la información del robot y era debido a que la lectura del puerto serie estaba sincronizada con la función de actualización de los fotogramas, por lo que para solucionarlo, utilizamos una hebra que constantemente vaya recogiendo la información que nos envía el robot por el puerto serie.

# MONTAJE

Ruedas

CNY70

Placa

Arduino UNO

Shield

Motores

Servo

Ultrasonido

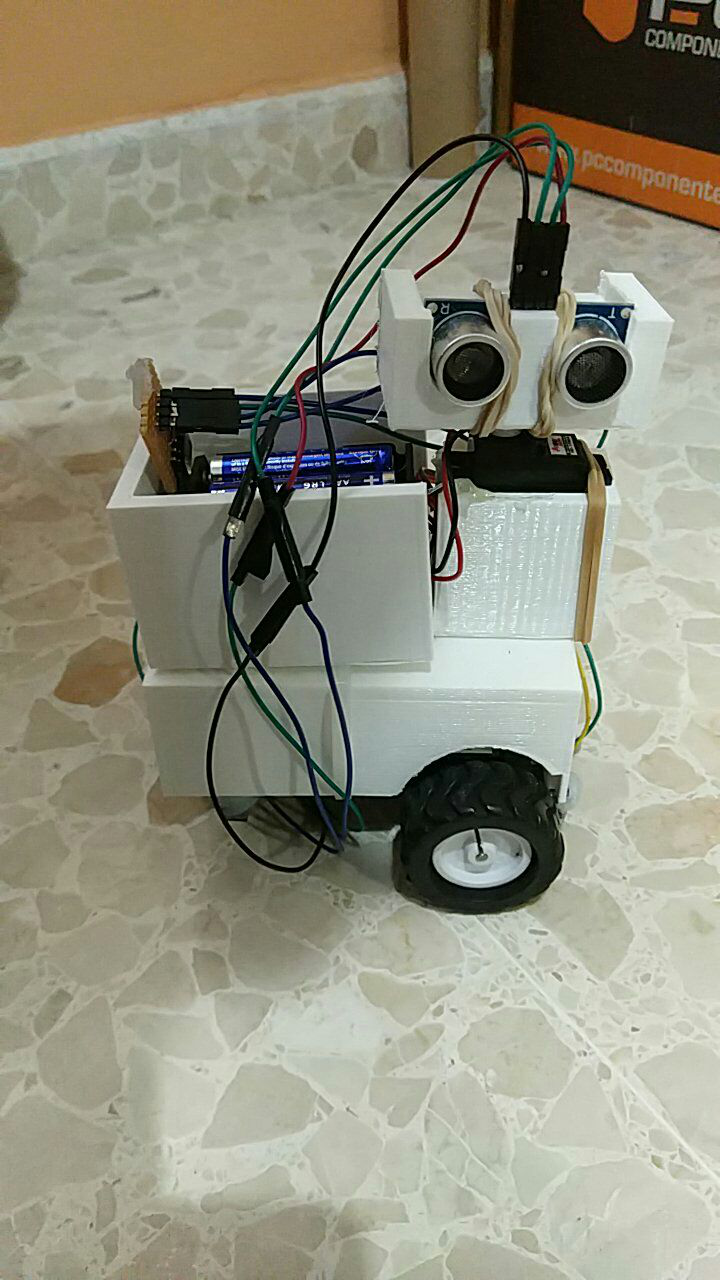
Batería

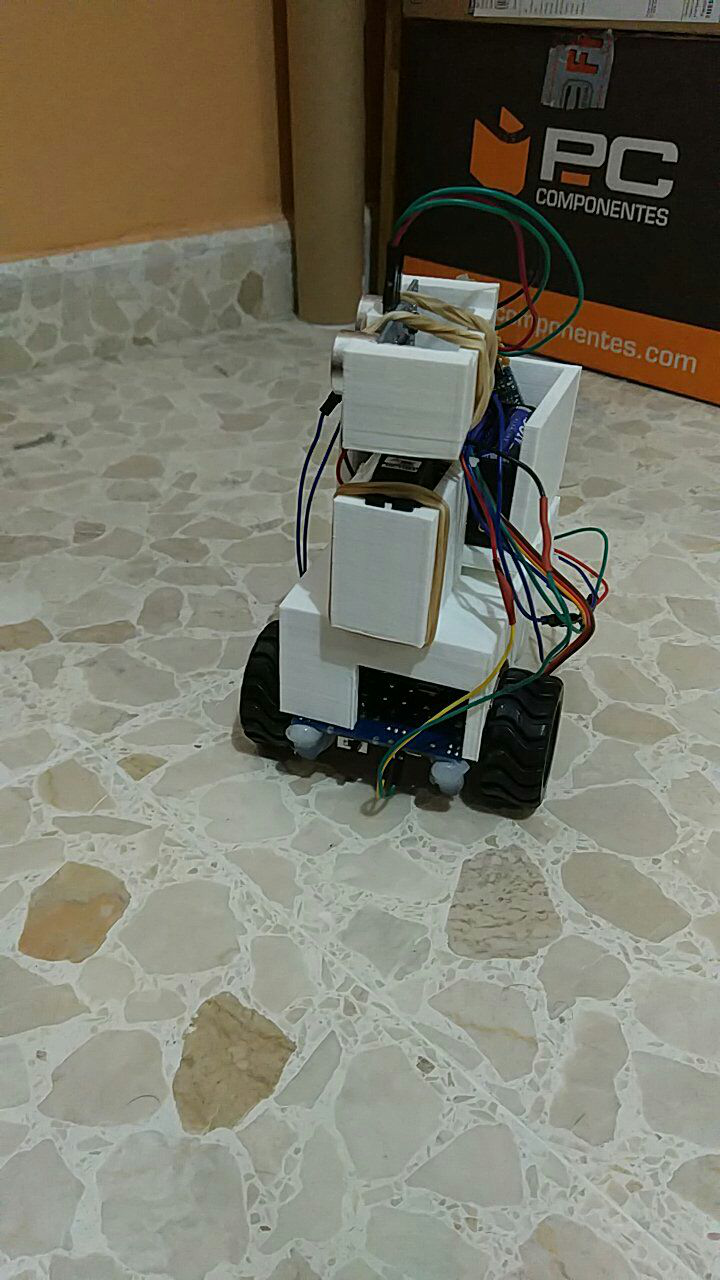
Bluetooth

Rueda

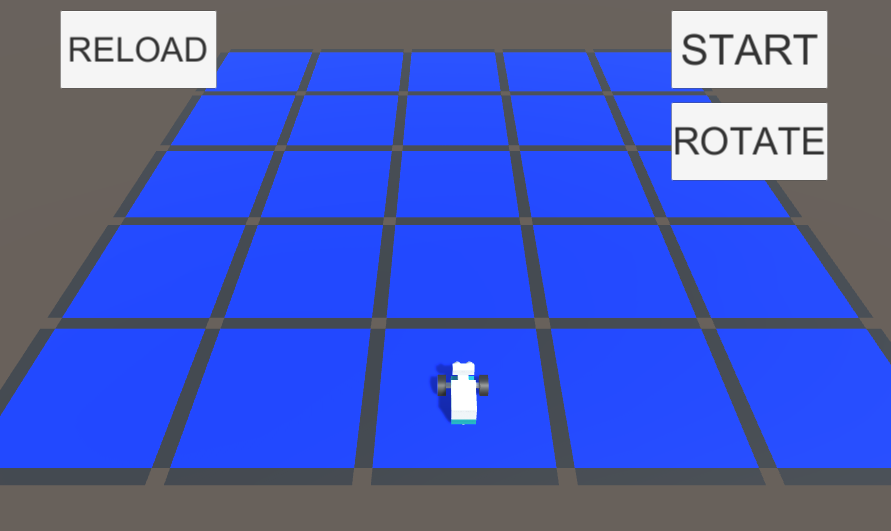
de apoyo trasera

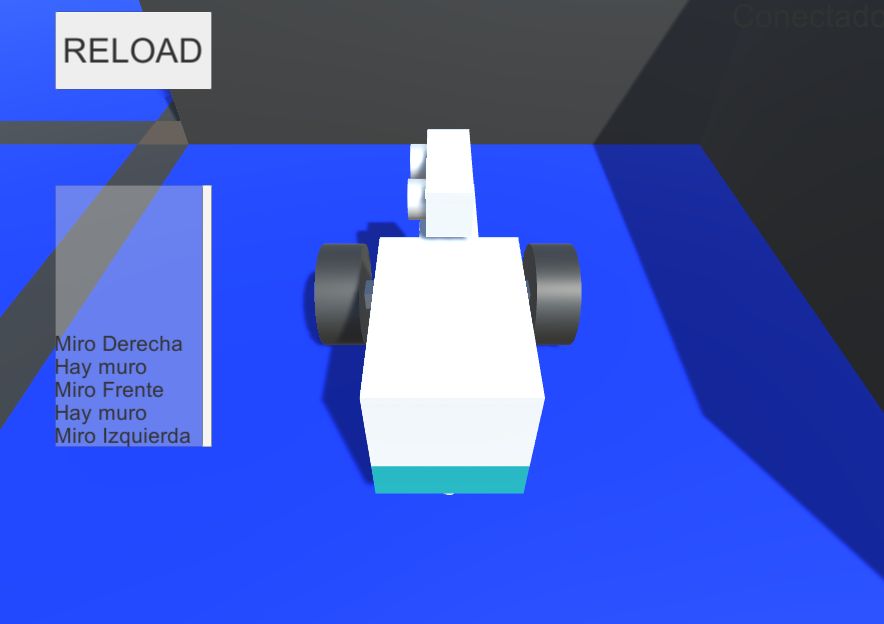


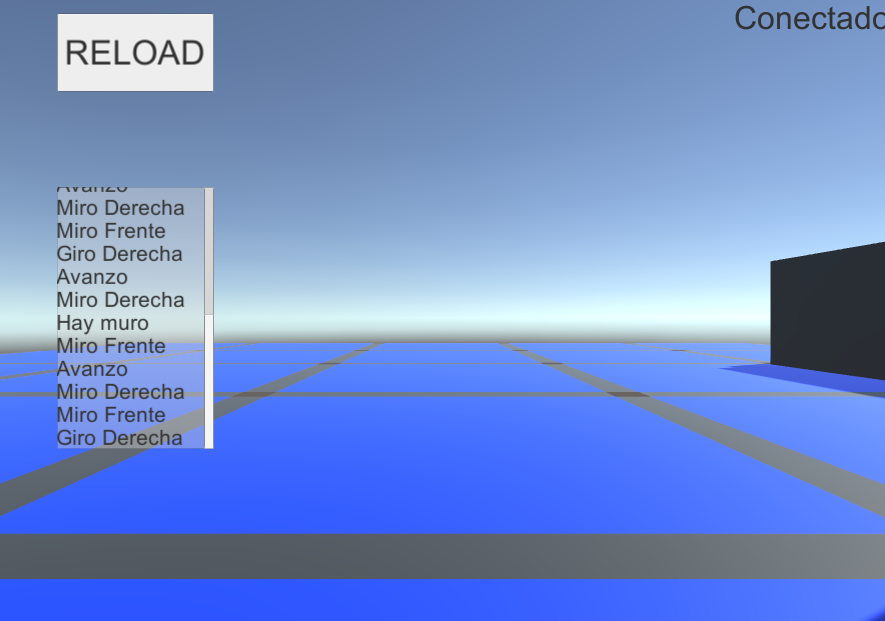


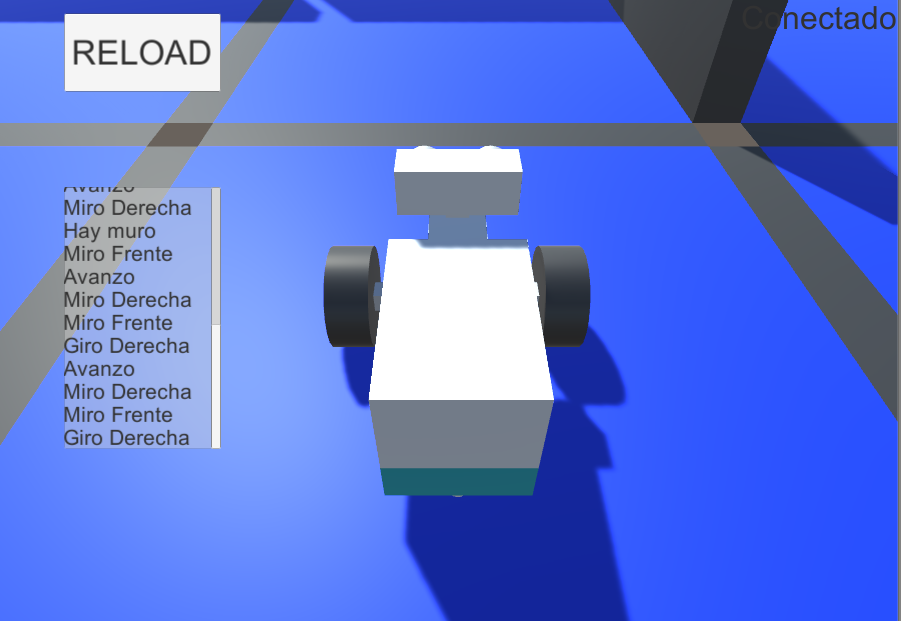












# PRUEBAS

| **ID** | **CATEGORIA** | **DESCRIPCIÓN** | **COMPROBACIÓN** |
| --- | --- | --- | --- |
| RF01 | Movimiento | El robot debe ser capaz de moverse | Función Avance. |
| RF02 | Movimiento | El robot debe ser capaz de pivotar 90º a derecha | Función GiroD. |
| RF03 | Movimiento | El robot debe ser capaz de pivotar 90º a izquierda | Función GiroI. |
| RF04 | Movimiento | El robot debe ser capaz de avanzar en línea recta | Función Avance. |
| RF05 | Movimiento | El robot avanza adecuadamente hasta la siguiente celda | Función Avance. |
| RF06 | Detección | El robot debe ser capaz de detectar una pared frontal | Función Camino. |
| RF07 | Detección | El robot debe ser capaz de detectar una pared lateral derecha | Función Camino. |
| RF08 | Detección | El robot debe ser capaz de detectar una pared lateral izquierda | Función Camino. |
| RF09 | Detección | El robot debe ser capaz de detectar la transición entre celdas | Función Avance. |
| RF10 | Detección | El robot debe ser capaz de detectar la celda de salida | Variable booleana. |
| RF11 | Resolución | El robot almacena información sobre las celdas del laberinto | Solo de la celda actual. |
| RF12 | Resolución | El robot es capaz de decidir el siguiente movimiento en base a la información sobre la celda | Sabe identificar si tiene muro o no a sus alrededores y toma un camino posible. |
| RF13 | Resolución | El robot es capaz de recorrer varias celdas del laberinto siguiendo el algoritmo empleado | Sí. |
| RF14 | Resolución | El robot es capaz de salir del laberinto | No comprobado. |
| RF15 | Información | El robot envía al PC información sobre el número de celdas recorridas | Mediante la función Serial1. |
| RF16 | Información | El robot envía al PC información sobre obstáculos en cada celda | Mediante la función Serial1. |
| RF17 | Información | El robot envía al PC información sobre la velocidad de movimiento | No procede. |
| RF18 | Información | El robot envía al PC información sobre la distancia que lleva recorrida | No procede. |
| RF19 | Información | El robot envía al PC información sobre el tiempo transcurrido desde la entrada al laberinto | No procede. |
| RF20 | Información | El robot envía al PC información sobre la trayectoria ejecutada | Mediante la función Serial1. |
| RF21 | Información | El robot envía al PC información sobre el número de celdas recorridas | Mediante la función Serial1. |
| RF22 | Información | El PC muestra una representación gráfica del laberinto | Mediante el entorno 3D de Unity. |
| RF23 | Usuario | Interfaz gráfica en PC que recopile información | Mediante el uso de Unity. |
| RF24 | Pruebas | Incluir modo test al arranque del robot | No procede. |

# MEJORAS

**Mejoras (Visuales):** Hemos realizado la impresión en 3D de varias piezas, una para proteger la placa Arduino y para colocar las demás piezas. Otra para colocar el servo, otra para la batería y una última para tener ajustado el ultrasonido.

**Mejoras (Software**): En la aplicación Unity se puede observar a vista del robot (en primera persona), cómo realiza el trayecto y cómo va reconociendo los obstáculos que se va encontrando. Una vez iniciada la aplicación y colocado el robot en la posición inicial del laberinto, al darle al botón START y cuando aparezca conectado, se puede alternar la cámara de tercera persona a primera persona pulsando la tecla “Espacio” del teclado.