Diseño de computadores empotrados

Departamento de Ingeniería informática

Universidad de Cádiz

Curso 2017/2018

Antonio Jesús Morales Rodríguez

Javier Gallardo Balber

Miguel Ángel Chaves Pérez

4 de Febrero del 2018

Índice general

1. INTRODUCCIÓN

1.1. 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:** Movimientos hacía adelante y atrás, además de movimientos rotatorios combinando de el giro de las ruedas independientemente
- **Detección:** Detección de marcas negras en el suelo mediante 3 sensores CNY, detección de distancia en un ángulo de 190º gracias la unión de un servo y un sensor de ultrasonido.
- **Recogida de información:** Se comprueba la existencia de paredes a menos de 20 cm en los angulos -90º (izq) 0º (frente) y 90º (der), se comprueban los 2 sensores de suelo traseros para alinear el robot en la celda y para saber que hemos llegado a ella, se comprueba el sensor delantero de suelo una vez el robot se ha alineado con la celda para determinar si hemos llegado a la celda objetivo. También se utiliza el sensor de ultrasonidos para comprobar si hay una pared a menos de 8cm cada vez que se alinea el robot en la celda, con el objetivo de girar el robot y evitar el coche con la pared.
- **Algoritmo de resolución del laberinto:** Se basa en la resolución de laberinto mediante la regla de la mano izq, esto significa ir avanzando por las celdas girando a la izquierda siempre que sea posible, si no hacia delante y como última opción derecha. Cuando se llega a un camino sin salida se da un giro de 180º y se sigue con la rutina.
- **Monitorización de información:** El robot envía información cada vez que llega a una casilla, manda las paredes de esa casilla , la distancia recorrida, el tiempo transcurrido y el numero de casillas recorridas.

1.2. Hardware empleado

Los distintos dispositivos hardware empleados para su total funcionamiento son:

- Arduino leonardo.
- Batería compuesta por 6 pilas.
- 3 Sensores CNY 70.
- 1 Sensor ultrasonicos
- 2 Motor DC
- 1 Servo motor 180º
- 1 Módulo Bluetooth

1.2.1. Arduino Leonardo

Arduino Leonardo. Esta placa, al igual que los demás modelos, es una placa electrónica basada en un microcontrolador. Una de las características de la placa es que cuenta con 20 pines de entrada/salida digitales, de los cuales, podemos usar 7 como salidas PWM (Pulse-Width Modulation) y 12 como entradas analógicas. Además cuenta con un oscilador de cristal que funciona a 16 MHz, una conexión micro USB, un conector de alimentación, una cabecera de ICSP (In-Circuit Serial Programming) y, un botón de reinicio. Cumple los requisitos necesarios para apoyar a la micro. Simplemente hay que conectarlo a nuestro PC mediante un cable USB o con una batería para comenzar a usarlo.

1.2.2. Sensores

El Sensor ultrasónico HC-SR04 es un circuito que detecta o mide la distancia y es compatible con Arduino. Básicamente, el sensor puede detectar objetos, distancia o nivel en un rango mínimo de 2 cm a un máximo de 400 cm. El sensor ultrasónico HC-SR04 se alimenta con 5 volts a 1.5 mA DC lo cual, de hecho, lo hace ideal para trabajar con Arduino, en realidad, con cualquier procesador lógico que funcione a 5V.
El dispositivo CNY70 es un sensor óptico infrarrojo, de un rango de corto alcance (menos de 5 cm) que se utiliza para detectar colores de objetos y superficies. Su uso más común es para construir pequeños robots siguelíneas. Contiene un emisor de -fotodiodo- y un receptor -fototransistor-. El fotodiodo emite un haz de radiación infrarroja, el fototransistor recibe ese haz de luz cuando se sobre alguna superficie u objeto.

1.2.3. Actuadores

Un actuador es un dispositivo capaz de transformar energía hidráulica, neumática o eléctrica en la activación de un proceso con la finalidad de generar un efecto sobre elemento externo. Este recibe la orden de un regulador, controlador o en nuestro caso un Arduino y en función a ella genera la orden para activar un elemento final de control como, por ejemplo, una válvula.
Mover un motor DC de 9V usando un integrado L293D (Quadruple Half-H driver). Para controlar la velocidad del motor se usará un potenciómetro conectado al pin A0. Además se usarán dos botones, uno conectado al pin digital 4 para controlar el sentido de giro del motor y otro conectado al pin digital 5 que controlará el encendido y apagado del motor. Con cada pulsación encendemos y apagamos el motor o usamos una dirección de giro u otra con el otro botón.

1.2.4. Elementos de comunicación

Se denomina Bluetooth al protocolo de comunicaciones diseñado especialmente para dispositivos de bajo consumo, que requieren corto alcance de emisión y basados en transceptores de bajo costo.

Los dispositivos que incorporan este protocolo pueden comunicarse entre sí cuando se encuentran dentro de su alcance. Las comunicaciones se realizan por radiofrecuencia de forma que los dispositivos no tienen que estar alineados y pueden incluso estar en habitaciones separadas si la potencia de transmisión es suficiente.

1.2.5. Alimentación

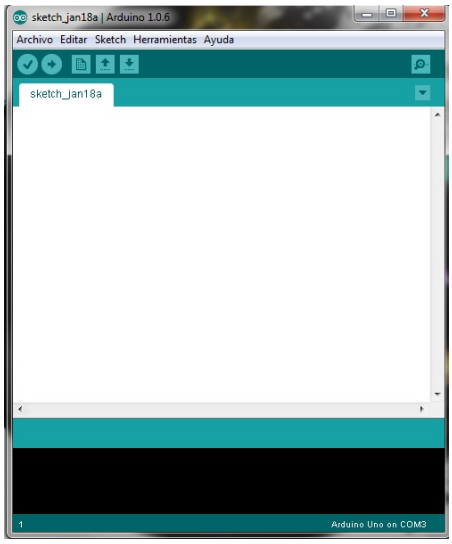
Empleamos una batería de 6 pilas AA en serie, proporcionando un total de 9V, es una de las opciones mas sencilla y ampliamente usado en pequeños proyectos y proyectos de iniciación. El precio de las pilas es barato, pero al no ser recargables a largo plazo no resulta económico.

1.3. Software empleado

1.3.1. Arduino IDE

Entorno de desarrollo integrado, llamado IDE (sigla en inglés de integrated development environment), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien puede utilizarse para varios.

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



1.3.2. Python 2.7

Python es un lenguaje de scripting independiente de plataforma y orientado a objetos, preparado para realizar cualquier tipo de programa, desde aplicaciones Windows a servidores de red o incluso, páginas web. Es un lenguaje interpretado, lo que significa que no se necesita compilar el código fuente para poder ejecutarlo, lo que ofrece ventajas como la rapidez de desarrollo e inconvenientes como una menor velocidad.

2. ESPECIFICACIÓN DE REQUISITOS

2.1. Requisitos funcionales

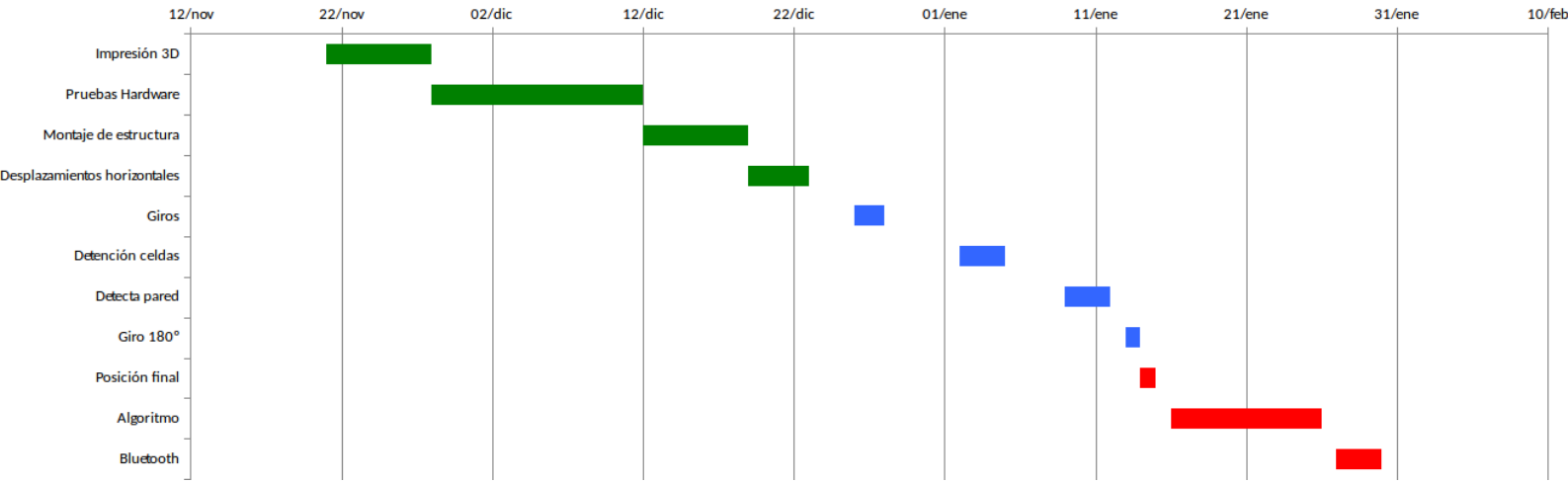
ID	CATEGORIA	DESCRIPCIÓN
RF01	Movimiento	El robot debe ser capaz de moverse
RF02	Movimiento	El robot debe ser capaz de pivotar 90º a derecha
RF03	Movimiento	El robot debe ser capaz de pivotar 90º a izquierda
RF04	Movimiento	El robot debe ser capaz de avanzar en línea recta
RF05	Movimiento	El robot avanza 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
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

2.2. Requisitos no funcionales

ID	CATEGORÍA	DESCRIPCIÓN	ACCIÓN
RNF01	Tamaño	Condicionado por las dimensiones del laberinto. Laberinto de 5 x 5 celdas, celda de 20 cm	
RNF02	Consumo	Condicionado por la batería disponible. Batería de 9v, compuesta por 6 pilas AAA	
RNF03	Correcciones	Robot choca contra una pared frontal	Se evita chocar mediante una detección temprana observando si hay un obstáculo frontal a menos de 8cm, en ese caso se retrocede durante 500 ms y sigue con la rutina como si hubiera llegado. Se manda un mensaje indicándolo
RNF04	Correcciones	Robot está muy cerca lateralmente de alguna pared	Se gira en el sentido opuesto al de la pared.
RNF5	Errores	Robot dirección lineal	El robot no presenta una dirección lineal exacta ya que tiende ir un poco hacia un lado
RNF6	Errores	Batería	El robot está diseñado para un nivel de batería alto, para un nivel de batería muy bajo, el robot puede presentar síntomas adversos al objetivo

3. PLANIFICACIÓN

Diagrama de Gantt



Impresión 3D: Antonio Jesús y Miguel Ángel

Prueba Hardware: Antonio Jesús, Javier, Miguel Ángel

Montaje de Estructura: Miguel Ángel

Desplazamiento: Antonio Jesús

Giros: Antonio Jesús

Detención Celda: Javier

Detectar Pared: Antonio Jesús

Posición final: Antonio Jesús

Algoritmo: Javier

Bluetooth: Javier

Memoria: Miguel Ángel

4. PRESUPUESTO

PRESUPUESTO

Proyecto Robótico

Nombre	DCE
Dirección	ESI
Población	Puerto Real
Provincia	Cádiz
CIF / NIF	UCA

FECHA PRESUPUESTO : 2 de Febrero del 2018

VALIDEZ : 7 DIAS

DESCRIPCION	UNIDADES	PRECIO	% DTO.	PRECIO DTO.	TOTAL
Arduino leonardo con cabezales A000057	1,00	21,70	2%	21,37	21,37
Batería pila AA Energizer Pack x6	1,00	6,66	2%	6,53	6,53
sensor CNY 70	3,00	0,62	2%	0,61	1,82
sensor ultrasonico HC-SR4	1,00	0,70	2%	0,69	0,69
Motor Dc	2,00	0,92	2%	0,90	1,80
Rueda giratoria	2,00	0,50	2%	0,49	0,98
servomotor HS-422	1,00	3,85	2%	3,77	3,77
Pack x25 cables de puentes arduino	1,00	0,86	2%	0,84	0,84
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
				0,00	0,00
			TOTAL BRUTO		37,81

I.V.A. %	0,00
Rec. Equiv. %	0,00

TOTAL PRESUPUESTO 37,81 €

Forma de pago :

Nombre, apellidos y firma de la persona que confecciona el presupuesto.

ACEPTO EL PRESUPUESTO. Nombre, apellidos y firma del cliente.

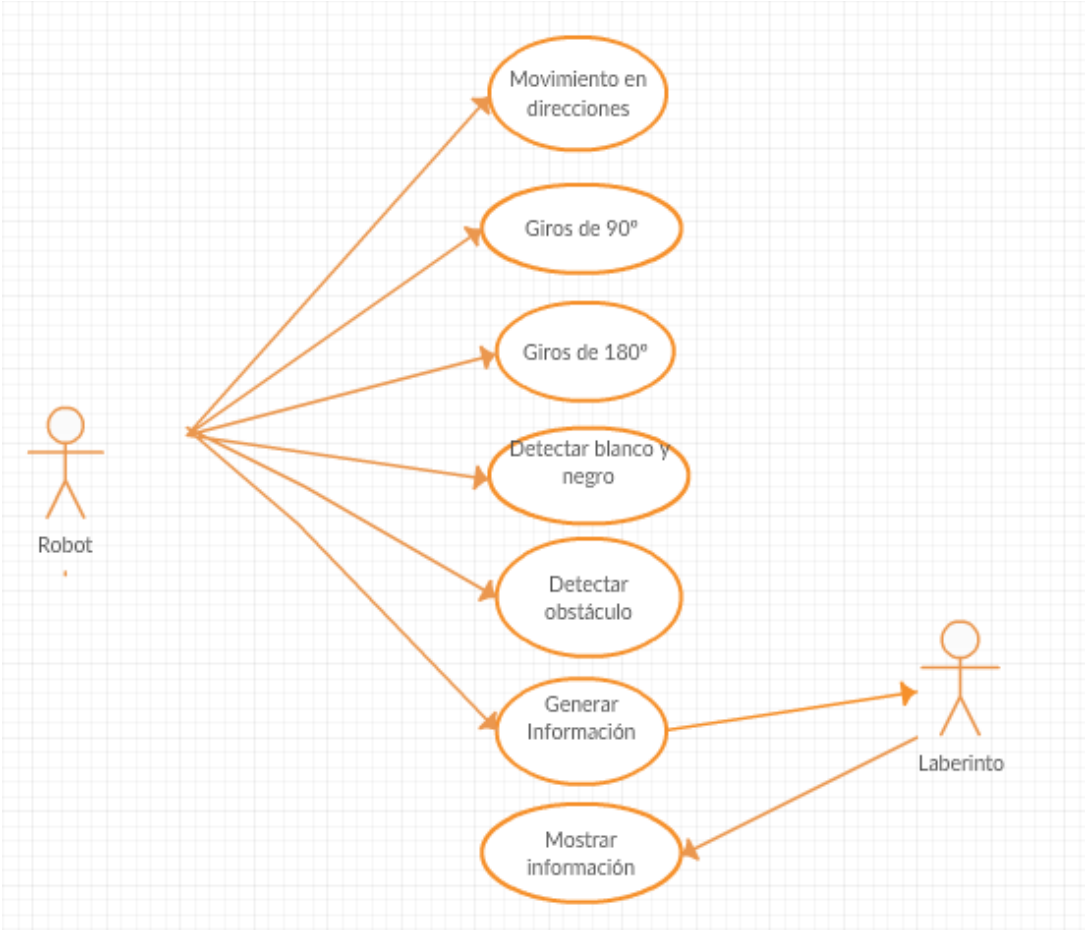
5. ANÁLISIS

5.1. Casos de uso

Actor	Robot	Identificador: ID001
Descripción	Robots móvil	
Características	Robots inteligente para salir de un laberinto	
Relación	Robot tiene relación con el Actor 2 Laberinto	
Referencias	Movimientos en direcciones, Giros de 90º, Giros de 180º, Detectar obstáculo, Detectar blanco y negro,	

Actor	Sistema	Identificador: ID002
Descripción	Interfaz Gráfica	
Características	Recopilación gráfico de los movimiento del Actor1 sobre Actor2	
Relación	Tiene relación con el Actor 2 Laberinto y Actor 1 Robot	
Referencias	Generar información, Mostrar información,	

Actor Secundario	Laberinto	Identificador: ID003
Descripción	Laberinto estático	
Características	Laberinto con una entrada y salida compuesta por paredes	
Relación	Laberinto tiene relación con el Actor 1 Robots	
Referencias		


Movimiento en direcciones
Giros de 90º
Giros de 180º
Detectar blanco y negro
Detectar obstáculo
Generar Información
Mostrar información
Robot
Laberinto

5.2. Diagrama de flujo

Diagrama de flujo de la aplicación Arduino:

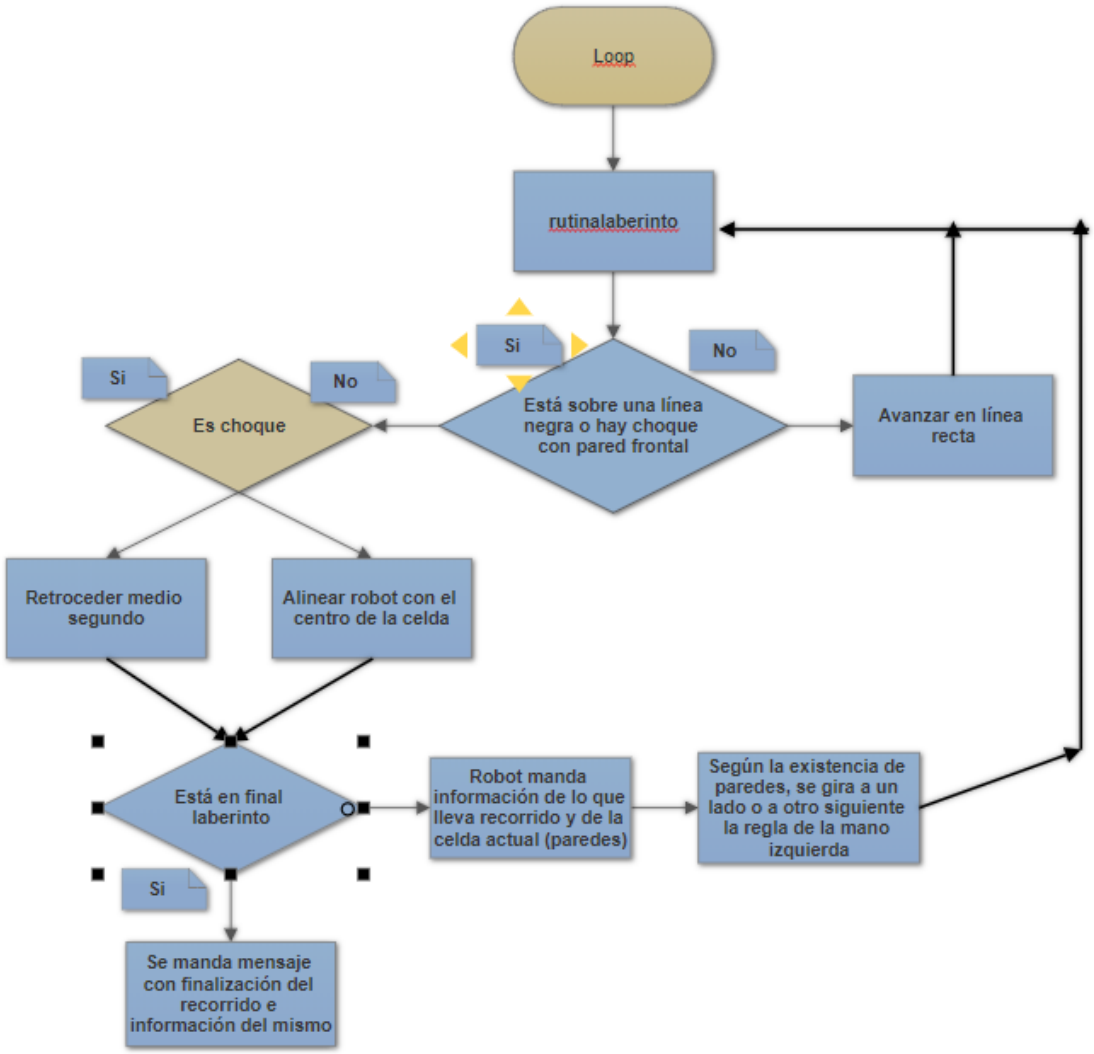
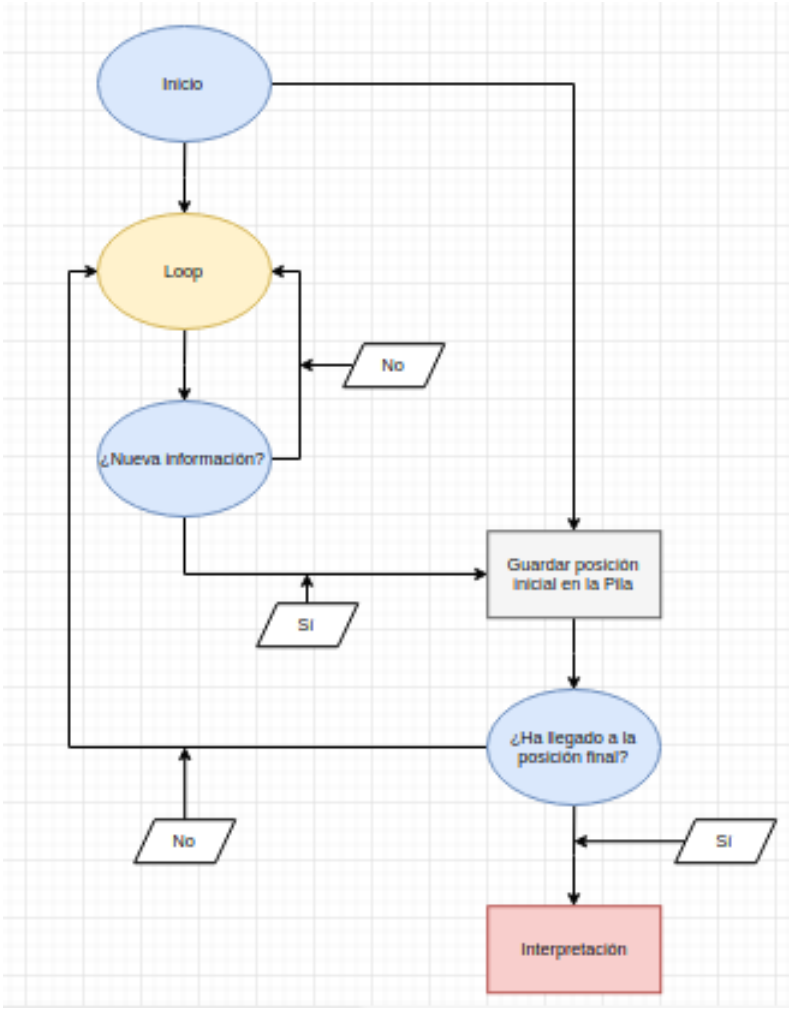


Diagrama de flujo de la aplicación Python



6. DISEÑO

6.1. Estructura

El programa principal hace uso de 8 librerías para llevar a cabo toda la lógica necesaria:

- Motor
- Giro
- ServoMotor
- SensorDistancia
- SensorDistanciaMovil
- Cny
- BotLaberinto
- Batería

Y para resolver el laberinto, llama en su bucle principal al método rutinaLaberinto de la clase BotLaberinto. En esta función se lleva a cabo el algoritmo de resolución de laberintos, haciendo uso de las demás librerías.

Adicionalmente cuenta con una función llamada testDeComponentes, donde hay varias pruebas de los componentes del robot.

6.2. Plan de pruebas

Se han realizado pruebas unitarias por cada librería:

BotLaberinto:

```
botLaberinto->avanzarFrente(50);
botLaberinto->retroceder(50);
Serial1.println("Hay pared izquierda?:");
Serial1.println(botLaberinto->hayParedIzquierda(20));
delay(250);
Serial1.println("Hay pared Derecha?:");
Serial1.println(botLaberinto->hayParedDerecha(20));
delay(250);
Serial1.println("Hay pared Frente?:");
Serial1.println(botLaberinto->hayParedFrente(20));
```

```
botLaberinto->avanzarFrente(50);
```

Motor:

```
motorIzq->acelerar(175);
motorDer->acelerar(175);
```

Giro:

```
giro->giroBiMotor(90);
  delay(5000);
  giro->giroBiMotor(-90);
  delay(5000);
  giro->giroBiMotor(-180);
```

ServoMotor:

```
servo->giro(-90);
  delay(1000);
  servo->giro(90);
  delay(1000);
  servo->giro(90);
  delay(1000);
  servo->giro(0);
  delay(1000);
  servo->giro(-90);
  delay(1000);
  servo->giro(90);
  delay(1000);
  servo->giro(0);
  delay(1000);
  servo->giroSuave(-90, 10);
 servo->giroSuave(0, 10);
 servo->giroSuave(90, 10);
```

SensorDistancia:

```
Serial1.println(sensorDistancia->getDistancia());
```

SensorDistanciaMovil:

```
Serial1.println("Distancia al frente:");
 Serial1.println(sensorDistanciaMovil->getDistancia(0));
 delay(250);
 Serial1.println("Distancia a la izquierda:");
 Serial1.println(sensorDistanciaMovil->getDistancia(-90));
 delay(250);
 Serial1.println("Distancia al frente:");
 Serial1.println(sensorDistanciaMovil->getDistancia(0));
 delay(250);
 Serial1.println("Distancia a la derecha:");
 Serial1.println(sensorDistanciaMovil->getDistancia(90));
 delay(250);
```

Cny:

```
Serial1.println("----------------------");
 Serial1.println("Cny delantero derecho");
 Serial1.println(cnyDelDer->esNegro(4));
 Serial1.println(cnyDelDer->getVoltaje());
Serial1.println("Cny trasero derecho");
 Serial1.println(cnyTraDer->esNegro(4.99));
 Serial1.println(cnyTraDer->getVoltaje());
 Serial1.println("Cny trasero izquierdo");
 Serial1.println(cnyTraIzq->esNegro(4));
 Serial1.println(cnyTraIzq->getVoltaje());
```

Batería:

```
Serial1.println((String)bateria->getVoltaje());
```

7. IMPLEMENTACIÓN

7.1. Librerías

- Motor: Proporciona funciones para el manejo del motor
- Giro: Proporciona funcionalidades para efectuar giros con dos ruedas
- ServoMotor: Proporciona funciones para el manejo del servo motor
- SensorDistancia: Proporciona funciones para la utilización del sensor ultrasonidos
- SensorDistanciaMovil: proporciona funcionalidades para el manejo conjunto del servo motor y el sensor de ultrasonidos
- Cny: Proporciona utilidades con respecto a el uso de los CNY para identificación binaria
- BotLaberinto: Clase que modela el robot laberinto, haciendo uso de todos los componentes, ofrece funciones útiles para resolver el laberinto
- Batería: Proporciona información sobre la batería

Todas estas librerías se pueden encontrar dentro de la carpeta “codigos”

7.2. Apuntes sobre el código

Se han encontrado muchos problemas a la hora de realizar las tareas que se precisan, a continuación se resumen los principales problemas:

Problemas con los Giros:

El mayor problema de los giros es que si no se hace nada para evitarlo, cuando calcular el tiempo que dejas encendido los motores para girar un determinado ángulo, a medida que se agota la batería y baja la tensión, las ruedas se mueven más lentamente y se necesita más tiempo para realizar el giro. Para resolver esta problemática se intento analizar la correlación entre voltaje y retardo necesario para girar unos determinados grados. Pero esta relación no es lineal, por lo que generar una función de la recta para resolver esta relación no resulto muy eficaz. Para solucionarlo, finalmente se acudió a los encoders de las ruedas, con los cuales se puede determinar la cantidad que gira cada rueda y así hacer el giro más preciso

Problemas con los CNY:

Los módulos CNY, han dado muchos problemas, pero casi todos relacionados a su conexión con la placa y fallos en las soldaduras del socket.

Problemas con el servo motor:

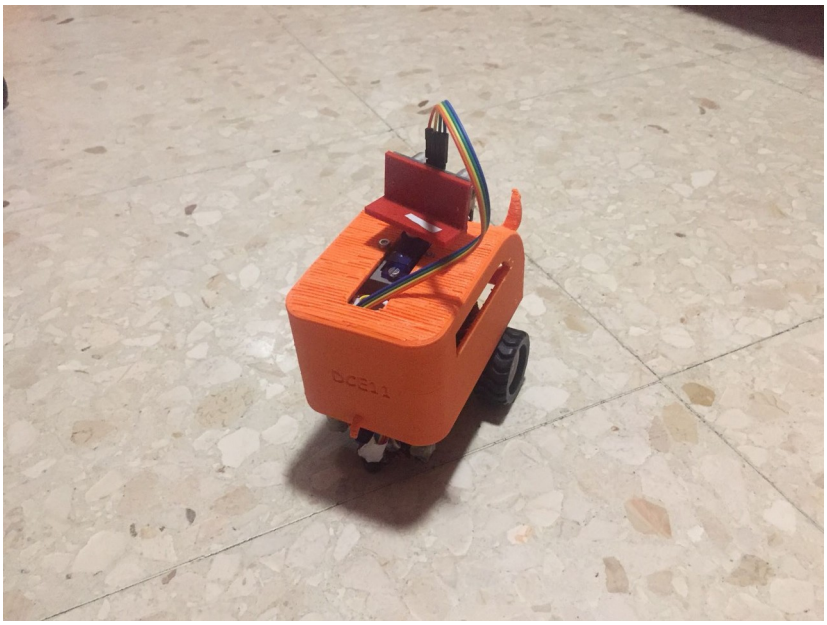
En un principio utilizamos la librería servo que trae por defecto el ide de Arduino, pero generaba problema con la rueda izq, concretamente con el pin D9, hasta donde sabemos, la librería servo utilizaba uno de los timmers del procesador. Eso hacía que no funcionara correctamente el pin D9. Tuvimos que hacer nosotros mismos una librería para el manejo del servo motor

Problemas con los encoders:

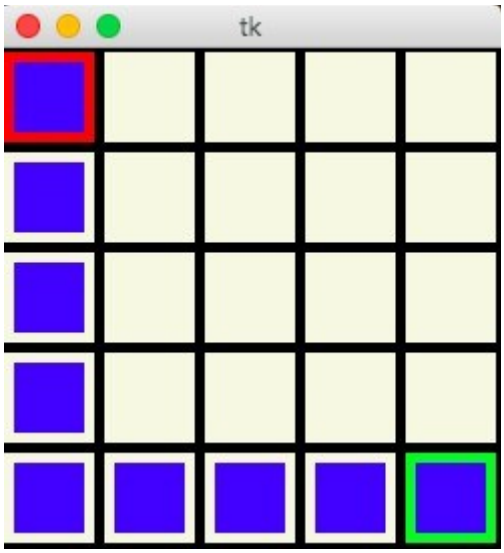
El encoder de la rueda izq no funciona, porque el pin D2 no funciona correctamente, hemos solventado este problema haciendo un puente entre el encoder y el pin D7.

8. MONTAJE

Robot Final



Interfaz Gráfica



9. PRUEBAS

Se ha construido un laberinto, de 2 x 4, emulando en que hay en clase, y hemos probado el funcionamiento del robot, en ese escenario, con respecto a los requisitos funcionales relacionados con la comunicación, hemos conectado con el smartphone con el robot y visto los mensajes mandados en el transcurso de la resolución del laberinto.

10. MEJORAS

Dado a lo poco preciso que son los giros en cuando a ángulo, se han llevado a cabo unas mejoras para resolver algunos casos que suceden en el transcurso de la resolución del laberinto.

Alineamiento con la línea de la celda

Dado que cada celda están marcadas por unas líneas negras, y teniendo en cuenta que tenemos 2 sensores cny traseros, hemos realizado una pequeña corrección del rumbo del robot, basándonos el alineamiento de estos sensores con la línea

Centrado lateral en la celda

Se usa el ultrasonido para determinar si en los laterales se está cerca de una pared, si es así se realiza un pequeño giro a la dirección opuesta, esto se realiza una vez se haya realizado el alineamiento con la línea que delimita la celda.

Bibliografía

[1] https://geekytheory.com/arduino-leonardo
[2] https://www.pccomponentes.com/arduino-leonardo-placa-de-desarrollo
[3] https://store.arduino.cc/arduino-leonardo-with-headers