2018

LENI – Robot recorre laberintos

COORDINADORA DEL PROYECTO: MIRIAN CIFREDO CHACÓN

HOLGADO DURÁN, JAVIER; HURTADO GARCÍA, ADRIÁN JOSÉ; PARDO SÁNCHEZ, JOSÉ MIGUEL

GRUPO 3 | Escuela Superior de Ingeniería, Universidad de Cádiz

Contenido

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: El Robot realiza movimientos hacia adelante, parar, a la izquierda, a la derecha y giros sobre sí mismo. Su movimiento por defecto es girar a la derecha. La corrección de los movimientos para estabilizarse la realizará por medio de los dos sensores CNY70 que se encuentran en la parte delantera.
- Detección: Para detectar obstáculos, y celdas para su posterior análisis hemos empleado dos sensores de distancia SHARP, un sensor de distancia por ultrasonidos y un sensor CNY70. El robot cuando detecta una línea negra se detiene, y pasa a utilizar los sensores de distancia para analizar la celda.
- Recogida de información: La información se mostrará en un móvil que esté conectado al robot por medio de Bluetooth. Para mostrar la información en el teléfono es necesaria una aplicación, en nuestro caso hemos utilizado "Bluetooth Terminal" disponible en Google Play Store.
- Algoritmo de resolución del laberinto: Para que el robot recorra el laberinto hemos empleado la técnica de ir pegado siempre a la pared derecha. Por tanto, su movimiento por defecto es girar a la derecha siempre que sea posible, si no, comprobará la pared de delante. En el caso de que también existiera una pared en frente, pasará a comprobar si hay pared a su izquierda. Si también existiera una pared en dicho lado, el robot procedería a dar un giro de 180º para volver sobre sus pasos, hasta la siguiente celda.
- Monitorización de información: En el teléfono se mostrará en todo momento los pasos que hace el robot. Cuando este llegue al final se mostrará un resumen donde se podrá ver el número de giros empleados a cada dirección y el número de celdas total recorridas.

1.2. Hardware empleado

Nombre	Descripción
Placa Arduino Leonardo	Placa de desarrollo de hardware compuesta principalmente por Microcontroladores.
Motor DC	Máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio. Se le aplica una tensión continua y empieza a girar.
Sensores Sharp	Sensor de distancia que detecta objetos o señales que se encuentran cerca del elemento sensor. Nuestros sensores son de distancia por infrarrojos que proporcionan un valor de tensión según la distancia a la que esté el objeto detectado.
Sensor Ultrasonido	El sensor ultrasónico determina la distancia a un objeto midiendo el tiempo que tarda en ir y volver una onda ultrasónica.

Sensor CNY70	Sensor que contiene un diodo LED que emite rayos infrarrojos y contiene un fototransistor. Lo aplicamos en el trabajo como detector de color blanco/negro. El blanco refleja todo el rayo, mientras que el negro lo absorbe.
Batería	Acumulador que proporciona electricidad a los motores.
Ruedas	Elementos aplicados a los motores DC para permitir el movimiento del robot.

1.2.1. Arduino Leonardo

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

El Arduino Leonardo contiene una tabla de microcontroladores basados en el ATmega32u4. Tiene 20 pines digitales de entrada / salida (los cuales 7 pueden usarse como salidas PWM y 12 como entradas analógicas), un oscilador de cristal, una conexión micro USB para facilitar la comunicación con el PC, un conector de alimentación, un encabezado ICSP y un botón de reinicio.

1.2.2. Sensores

El robot en cuanto a sensores está compuesto de:

- Sensor de Ultrasonido: Sensor colocado en la parte delantera del robot que va a ir determinando la distancia con obstáculos que vayan apareciendo delante suya.
- Sensor CNY70: Dos sensores CNY70 que van a ir comprobando mediante rayos infrarrojos cuando el robot ha llegado a una celda, que será cuando detecte una línea negra.
- Sensor Sharp: Dos sensores Sharp colocados en una pieza céntrica del robot que van a ir midiendo las distancias con los obstáculos que tengan tanto a su derecha como a su izquierda.

1.2.3. Actuadores

El robot en cuanto a actuadores está compuesto de dos motores DC acoplados a dos ruedas, que va a permitir el movimiento de éste. La velocidad se ajustará según la batería que tengamos conectada.

1.2.4. Elementos de comunicación

Los elementos de comunicación para el robot son un PC que va a ir recogiendo información del robot como la velocidad que mantiene, las celdas que lleva recorridas, la distancia desde la salida hasta su posición actual, etc. Además, también de un gráfico que mostrará el laberinto completo y al final el PC mostrará también vía Bluetooth las decisiones que va tomando el robot. Ya sea cuando avanza, cuando vaya a detenerse, cuando analiza una celda, etc.

1.2.5. Alimentación

El robot va a ir alimentado por una batería compuesta por una base que contiene seis pilas AA.

1.3. Software empleado

1.3.1. Arduino IDE

Lenguaje en el que se realiza el algoritmo de resolución del robot. Permite que podamos interactuar con nuestra placa Arduino y poder programarla. El algoritmo de resolución contendrá todo lo necesario para que el robot cumpla el objetivo de llegar al final.

1.3.2. Bluetooth Terminal

Aplicación Android utilizada para mostrar los datos recogidos por el robot en el recorrido del laberinto.

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
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

2.2. Requisitos no funcionales

ID	CATEGORIA	DESCRIPCIÓN	ACCIÓN
RNF01	Tamaño	Condicionado por las dimensiones del laberinto. Laberinto de tamaño 5x5	El robot es capaz de superar un laberinto de cualquier tamaño.
RNF02	Consumo	Condicionado por las pilas.	
RNF03	Errores	El robot choca contra una pared	
RNF04	Errores	Pilas a punto de agotarse	El robot corregirá los giros cortos gracias a los sensores CNY70 delanteros.
RNF05	Errores	El robot gira continuamente	
RNF06	Errores	El robot sobrepasa 20 minutos sin conseguir salir	

3. PLANIFICACIÓN

3.1. Diagrama de Gantt

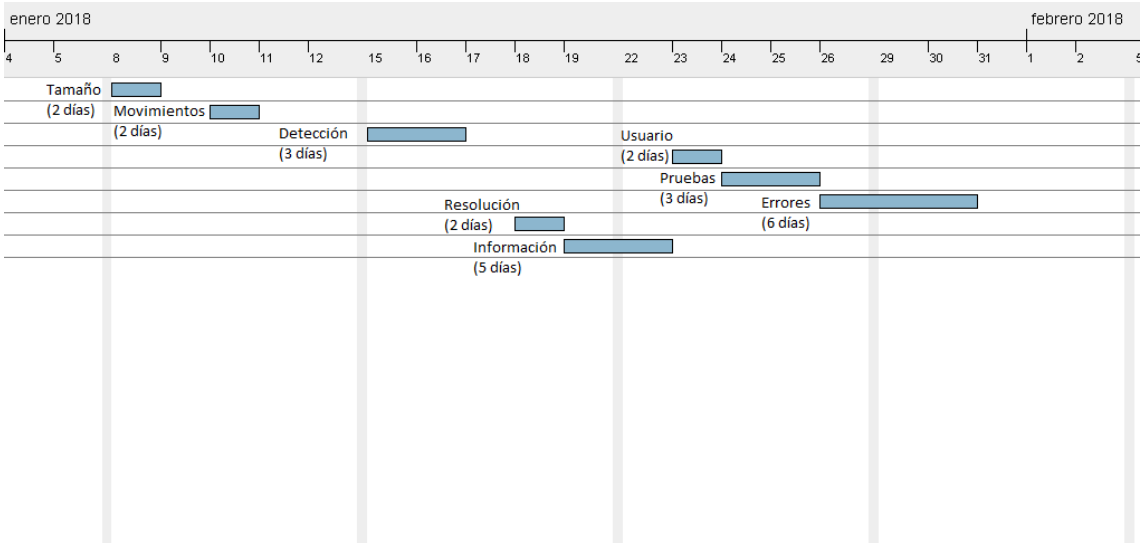


Diagrama de Gantt

4. PRESUPUESTO

Nombre	Unidades	Precio Sin IVA	IVA	Precio con IVA por unidad	Precio total con IVA
PCB Shield Arduino	1	23,70 €	21%	28,68 €	28,68 €
Arduino Leonardo	1	4,64 €	21%	5,61 €	5,61 €
Motor DC L293D + Ruedas	2	6,20 €	21%	7,50 €	15,00 €
Tornillos y Tuercas	9	2,37 €	21%	2,87 €	25,81 €
Sensor CNY70	2	2,02 €	21%	2,44 €	4,89 €
Sensor Sharp	2	9,89 €	21%	11,97 €	23,93 €
Sensor Ultrasonidos	1	0,75 €	21%	0,91 €	0,91 €
Base para 6 pilas	1	0,91 €	21%	1,10 €	1,10 €
Paquete de 8 pilas AAA	1	1,40 €	21%	1,69 €	1,69 €
Total	**20**	**51,88 €**	**21%**	**62,77 €**	**107,63 €**

5. ANÁLISIS

5.1. Diagrama de flujo

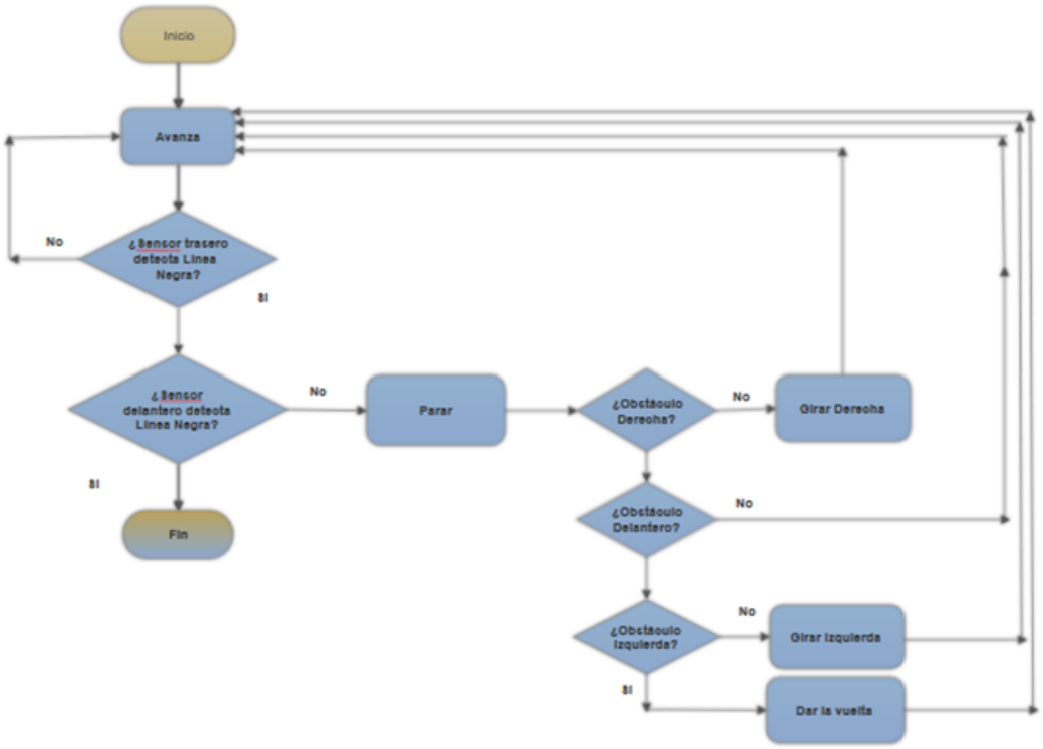


Diagrama de Flujo

6. DISEÑO

6.1. Estructura

```
1. class Robot{
2.   public:
3.       bool  sensorDelanteroIZQ();
4.       bool  sensorDelanteroDER();
5.       void  correccionDelantera();
6.       bool  sensorTrasero();
7.       int   analizarCelda();  //Adelante 1, Derecha 2, Izquierda 3,
   Vuelta 4
8.       void  movAdelante();
9.       void  movDerecha();
10.      void  movIzquierda();
11.      void  movVuelta();
12.      void  parar();
13.      int   numCeldasRecorridas() {return numCeldasRecorridas_;}
14.      int   numGirosDerecha()    {return numGirosDerecha_;}
15.      int   numGirosIzquierda() {return numGirosIzquierda_;}
16.      int   numVueltas()  {return numVueltas_;}
17.      void  meta();
18.   private:
19.      //Datos de interes sobre el recorrido
20.      int   numCeldasRecorridas_ = 0;
21.      int   numGirosDerecha_ = 0;
22.      int   numGirosIzquierda_ = 0;
23.      int   numVueltas_ = 0;
24.      //Constante de velocidad
25.      int   velocidad = 255;
26.      //Funciones internas
27.      bool  paredFre();
28.      bool  paredIzq();
29.      bool  paredDer();
30.      double  GetDistance(float v);
31.      //Pines
32.      //Motores
33.      const int mDerA = 5;
34.      const int mDerB = 6;
35.      const int mIzqA = 10;
36.      const int mIzqB = 9;
37.      //Sensores Luminicos
38.      const int CNY_Pin_DelanteroDER = A0;
39.      const int CNY_Pin_DelanteroIZQ = A1;
40.      const int CNY_Pin_Trasero = A5;
41.      const int NEGRO =750;
42.      //Sensores distancia
43.      const int pinSeDelantero = 8;
```

```
      const int pinSeIzquierda = A3;
      const int pinSeDerecha = A4;
      //Bluetooth
      const int RXBT = 0;
      const int TXBT = 1;
      //LEDS
      const int led1 = 12;
      const int led2 = 13;
      const int led3 = 11;
};

bool corregir = true;

//Calcular distancia sensores laterales
double Robot::GetDistance(float v){
  if(v<0.5 || v>2.7){
    return 0;
  }
  return (12/(v-0.1))-0.42;
}

//Comprueba si hay pared, true = hay pared, false = espacio libre.
bool Robot::paredFre(){

  float distancia;
  unsigned long time_bounce;

  pinMode(pinSeDelantero, OUTPUT);
  digitalWrite(pinSeDelantero, LOW);
  delayMicroseconds(5);

  digitalWrite(pinSeDelantero, HIGH);
  delayMicroseconds(10);
  pinMode(pinSeDelantero,INPUT);
  digitalWrite(pinSeDelantero, LOW);

  time_bounce = pulseIn(pinSeDelantero, HIGH);

  distancia = 0.017 * time_bounce; //Formula distancia

  return distancia < 20;
}

//Detector pared derecha

bool Robot::paredDer(){
  float ResolutionADC = 0.00488;
```

```
91.   double Value_Pin = analogRead(pinSeDerecha);
92.   double Voltage = Value_Pin*ResolutionADC;
93.   double distancia = GetDistance(Voltage);
94.
95.   return distancia < 15 && distancia > 5;
96.}
97.
98.//Detector pared izquierda
99.bool Robot::paredIzq(){
100.       float ResolutionADC = 0.00488;
101.       double Value_Pin = analogRead(pinSeIzquierda);
102.       double Voltage = Value_Pin*ResolutionADC;
103.       double distancia = GetDistance(Voltage);
104.
105.       return distancia < 15 && distancia > 5;
106.     }
107.
108.     //Detector de ambos sensores delanteros
109.     bool Robot::sensorDelanteroIZQ(){
110.       int ValorSensorDelanteroIZQ =
   analogRead(CNY_Pin_DelanteroIZQ);
111.       return ValorSensorDelanteroIZQ > NEGRO + 100;
112.     }
113.
114.     bool Robot::sensorDelanteroDER(){
115.       int ValorSensorDelanteroDER =
   analogRead(CNY_Pin_DelanteroDER);
116.       return ValorSensorDelanteroDER > NEGRO;
117.     }
118.
119.     //Detector de sensores trasero
120.     bool Robot::sensorTrasero(){
121.       int  ValorSensorTrasero = analogRead(CNY_Pin_Trasero);
122.
123.       if (ValorSensorTrasero > NEGRO)
124.         numCeldasRecorridas_++;
125.
126.       return ValorSensorTrasero > NEGRO;
127.     }
128.
129.     //Analiza celda en función de las paredes.
130.     int   Robot::analizarCelda(){
131.         if(paredDer()){
132.           if(paredFre()){
133.             if(paredIzq()){
134.               return 4;
135.             }else{
```

```
136.               return 3;
137.             }
138.           }else{
139.             return 1;
140.           }
141.         }else{
142.           return 2;
143.         }
144.     }
145.
146.     //----------------------
147.     //------MOVIMIENTOS-----
148.     //----------------------
149.     void  Robot::movAdelante(){
150.         analogWrite(mDerA, 150);
151.         analogWrite(mDerB, 0);
152.         analogWrite(mIzqA, 0);
153.         analogWrite(mIzqB, 150);
154.     }
155.
156.     void Robot::movDerecha(){
157.
158.       analogWrite(mDerA, 0);
159.       analogWrite(mDerB, 0);
160.       analogWrite(mIzqA, 0);
161.       analogWrite(mIzqB, 255);
162.       delay(1250);
163.       corregir = true;
164.
165.       numGirosDerecha_++;
166.
167.       //correccionDelantera();
168.     }
169.
170.     void Robot::movIzquierda(){
171.       analogWrite(mDerA, 255);
172.       analogWrite(mDerB, 0);
173.       analogWrite(mIzqA, 0);
174.       analogWrite(mIzqB, 0);
175.       delay(1250);
176.       corregir = true;
177.
178.       numGirosIzquierda_++;
179.
180.       //correccionDelantera();
181.     }
182.
```

```
183.    void Robot::movVuelta(){
184.
185.      analogWrite(mDerA, 150);
186.      analogWrite(mDerB, 0);
187.      analogWrite(mIzqA, 0);
188.      analogWrite(mIzqB, 150);
189.
190.      delay(300);
191.
192.      analogWrite(mDerA, velocidad/4);
193.      analogWrite(mDerB, 0);
194.      analogWrite(mIzqA, velocidad/4);
195.      analogWrite(mIzqB, 0);
196.      delay(3200);
197.
198.      analogWrite(mDerA, 150);
199.      analogWrite(mDerB, 0);
200.      analogWrite(mIzqA, 0);
201.      analogWrite(mIzqB, 150);
202.
203.
204.      numVueltas_++;
205.    }
206.
207.    void Robot::parar(){
208.      analogWrite(mDerA, 0);
209.      analogWrite(mDerB, 0);
210.      analogWrite(mIzqA, 0);
211.      analogWrite(mIzqB, 0);
212.    }
213.
214.    void Robot::meta(){
215.      if(sensorDelanteroDER() && sensorTrasero()){
216.        pinMode(led1, OUTPUT);
217.        pinMode(led2, OUTPUT);
218.        pinMode(led3, OUTPUT);
219.      }
220.      while(true){
221.        digitalWrite(led1, HIGH);
222.        delay(100);
223.        digitalWrite(led1, LOW);
224.        delay(100);
225.        digitalWrite(led2, HIGH);
226.        delay(100);
227.        digitalWrite(led2, LOW);
228.        delay(100);
229.        digitalWrite(led3, HIGH);
```

```
230.          delay(100);
231.          digitalWrite(led3, LOW);
232.          delay(100);
233.        }
234.      }
235.
236.      void Robot::correccionDelantera(){
237.
238.        if(sensorDelanteroDER()){
239.          while(!sensorDelanteroIZQ()){
240.            analogWrite(mDerA, 0);
241.            analogWrite(mDerB, 0);
242.            analogWrite(mIzqA, 0);
243.            analogWrite(mIzqB, 100);
244.          }
245.          corregir = false;
246.        }
247.        if(sensorDelanteroIZQ()){
248.          while(!sensorDelanteroDER()){
249.            analogWrite(mDerA, 100);
250.            analogWrite(mDerB, 0);
251.            analogWrite(mIzqA, 0);
252.            analogWrite(mIzqB, 0);
253.          }
254.          corregir = false;
255.        }else{
256.          analogWrite(mDerA, 150);
257.          analogWrite(mDerB, 0);
258.          analogWrite(mIzqA, 0);
259.          analogWrite(mIzqB, 150);
260.        }
261.
262.      }
263.
264.      void setup() {
265.        Serial1.begin(9600);
266.      }
267.
268.      bool final = false;
269.
270.      void loop() {
271.        Robot robot;
272.        int movimiento;
273.        int empezar = 0;
274.
275.        delay(5000);
276.
```

```
277.        Serial1.println("Hola, soy Leni. El robot que va a recorrer
    este laberinto.");
278.
279.        /*while(empezar != 288)
280.          empezar = Serial1.read();*/
281.
282.
283.
284.        do{
285.
286.            robot.movAdelante();
287.            delay(100);
288.
289.            corregir = true;
290.
291.            while(!robot.sensorTrasero()){
292.              if(corregir)
293.                robot.correccionDelantera();
294.              else
295.                robot.movAdelante();
296.            }
297.
298.            corregir = true;
299.
300.            robot.parar();
301.
302.            if(robot.sensorDelanteroIZQ() and
    robot.sensorDelanteroDER())
303.              final = true;
304.            else{
305.              movimiento = robot.analizarCelda();
306.
307.              switch(movimiento){
308.                case 1:
309.                    Serial1.println("Leni ha detectado una pared a
    su derecha, por lo tanto sigue adelante");
310.                    robot.movAdelante();
311.                    delay(200);
312.                    break;
313.                case 2:
314.                    Serial1.println("Leni no ha detectado una pared
    a la derecha, por tanto sigue su movimiento por defecto, girar a la
    derecha");
315.                    robot.movDerecha();
316.                    break;
317.                case 3:
```

```
318.                    Serial1.println("Leni ha detectado paredes
   tanto a su derecha, como en frente por tanto gira a la izquierda");
319.                    robot.movIzquierda();
320.                    break;
321.                case 4:
322.                    Serial1.println("Leni se ha encontrado
   acorralado por todos lados, por tanto decide dar la vuelta sobre si
   mismo, para volver sobre sus pasos.");
323.                    robot.movVuelta();
324.                    break;
325.                }
326.            }
327.
328.            Serial1.println(robot.numCeldasRecorridas());
329.
330.        } while(!final);
331.
332.        Serial1.println("Leni ha terminado el recorrido del
   laberinto y estos son los resultados: ");
333.
334.        Serial1.print("Número de celdas recorridas: ");
335.        Serial1.println(robot.numCeldasRecorridas());
336.
337.        Serial1.print("Número de giros a la derecha: ");
338.        Serial1.println(robot.numGirosDerecha());
339.
340.        Serial1.print("Número de giros a la izquierda: ");
341.        Serial1.println(robot.numGirosIzquierda());
342.
343.        Serial1.print("Número de vueltas dadas: ");
344.        Serial1.println(robot.numVueltas());
345.
346.        robot.meta();
347.        delay(50000);
348.
349.    }
```

6.2. Plan de pruebas

Las pruebas se realizaron directamente con el código arriba descrito. Al declarar cada sensor como una función de ámbito público podíamos llamarla de manera individual para comprobar su correcto funcionamiento.

7. IMPLEMENTACIÓN

7.1. Librerías

No se han hecho uso de librerías externas.

7.2. Apuntes sobre el código

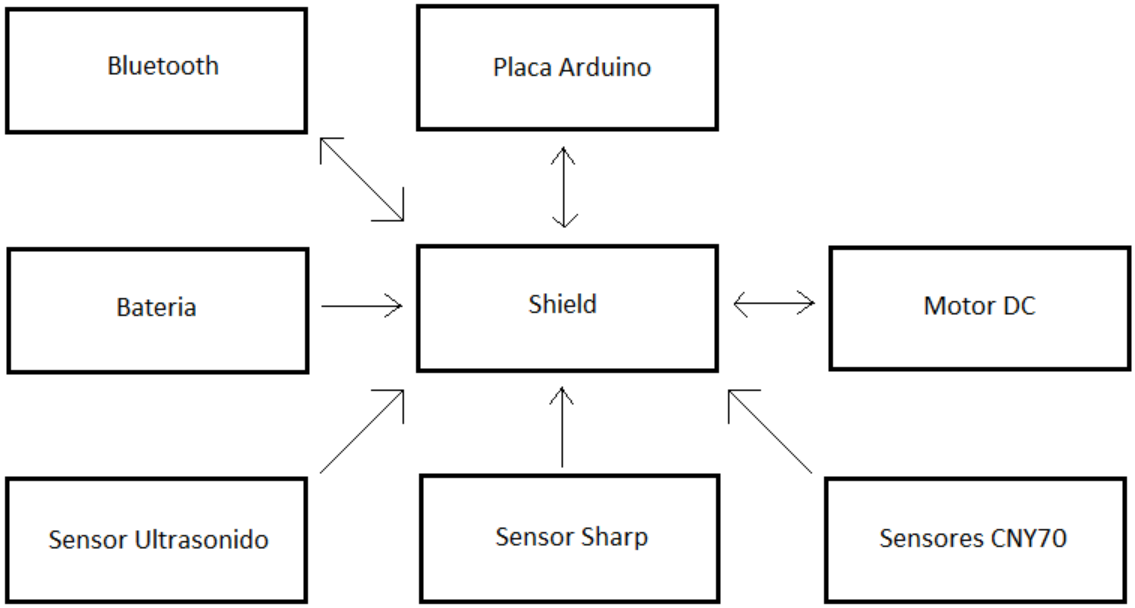
El código se ha realizado por medio de una clase “Robot” que es la que se encarga de todo lo relacionado con el robot de forma simplificada.

- bool sensorDelanteroIZQ()
  - Precondición:
  - Postcondicion: Devuelve true si el sensor CNY70 delantero izquierdo ha detectado una línea negra.
- bool sensorDelanteroDER()
  - Precondición:
  - Postcondicion: Devuelve true si el sensor CNY70 delantero derecho ha detectado una línea negra.
- void correccionDelantera()
  - Precondición: esté realizando un movimiento hacia delante.
  - Postcondicion: se encarga de corregir la trayectoria del robot cuando está haciendo un movimiento adelante.
- bool sensorTrasero()
  - Precondición:
  - Postcondición: devuelve true si el sensor CNY70 trasero ha detectado una línea negra.
- int analizarCelda()
  - Precondición:
  - Postcondición: Devuelve un entero en función de las paredes que haya detectado el robot. 1 = Adelante, 2 = giro a la derecha, 3 = giro a la izquierda, 4 = giro 180º.
- void movAdelante()
  - Precondición:
  - Postcondición: el robot se mueve hacia delante.
- void movDerecha()
  - Precondición:
  - Postcondición: el robot gira a la derecha.
- void movIzquierda()
  - Precondición:
  - Postcondición: el robot gira a la izquierda.
- void movVuelta()
  - Precondición:
  - Postcondición: el robot realiza un giro de 180º.
- void parar()
  - Precondición: el robot esté realizando algún movimiento.
  - Postcondición: el robot se para.
- int numCeldasRecorridas()
  - Precondición:

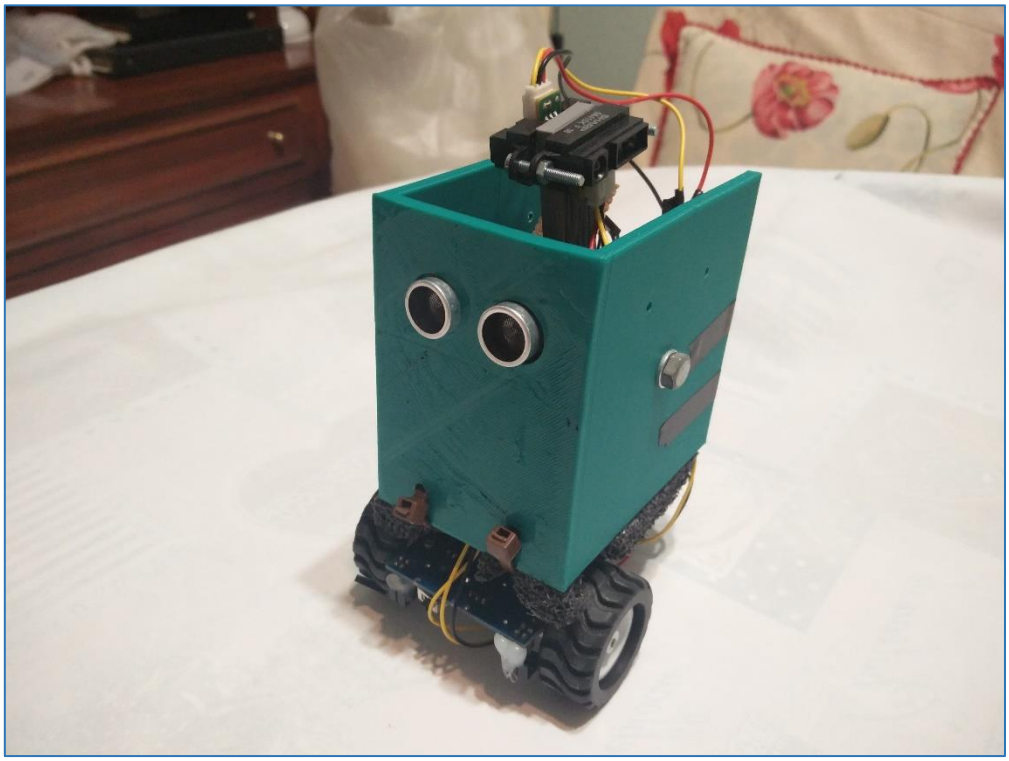
    - Postcondición: devuelve el número de celdas recorridas.
- int numGirosDerecha()
    - Precondición:
    - Postcondición: devuelve el número de giros realizados a la derecha.
- int numGirosIzquierda()
    - Precondición:
    - Postcondición: devuelve el número de giros realizados a la izquierda.
- int numVueltas()
    - Precondición:
    - Postcondición: devuelve el número de vueltas realizadas.
- void meta()
    - Precondición: los 3 sensores de CNY70 han de estar sobre negro.
    - Postcondición: el robot enciende sus leds.

8. MONTAJE

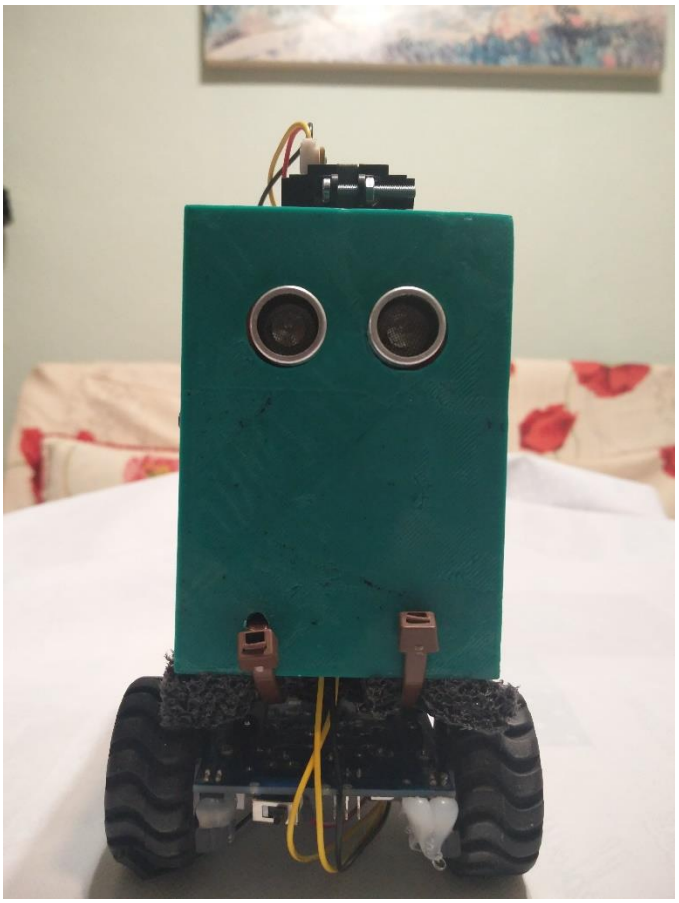
8.1. Diagrama de bloques



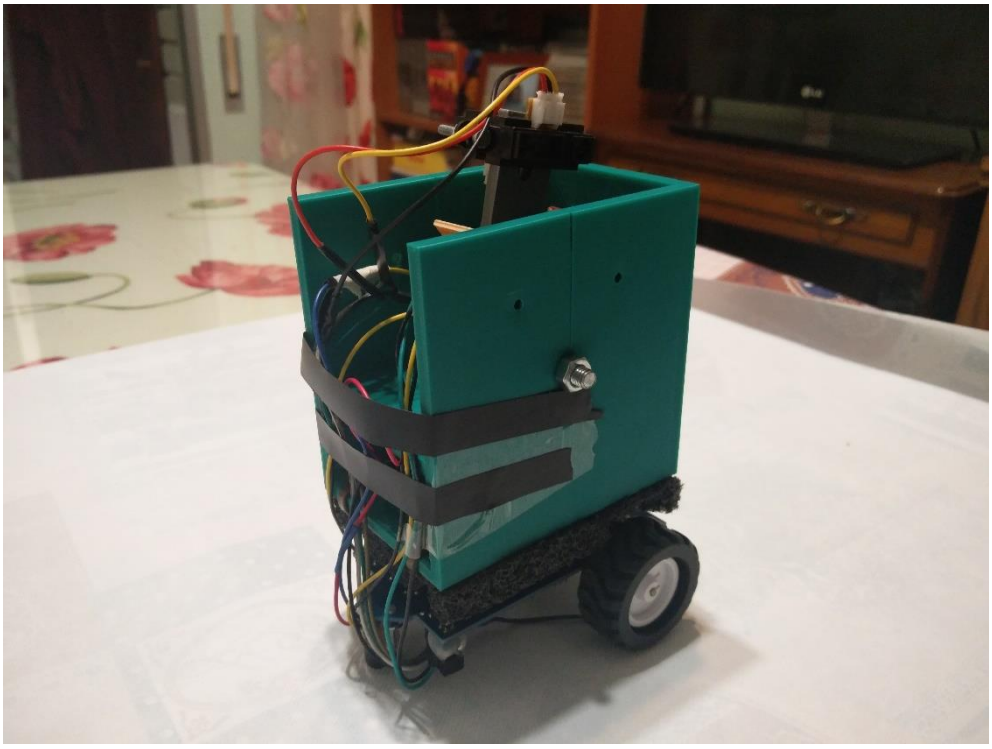
8.2. Fotos



LENI – Vista general



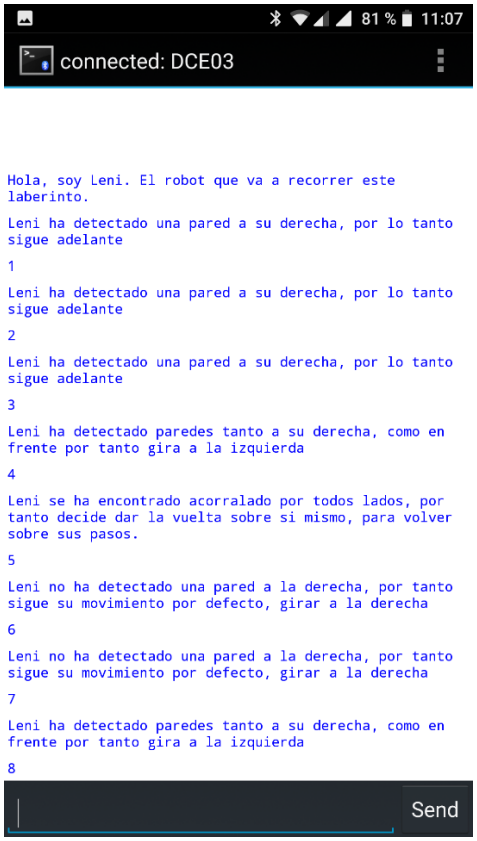
LENI - Vista frontal



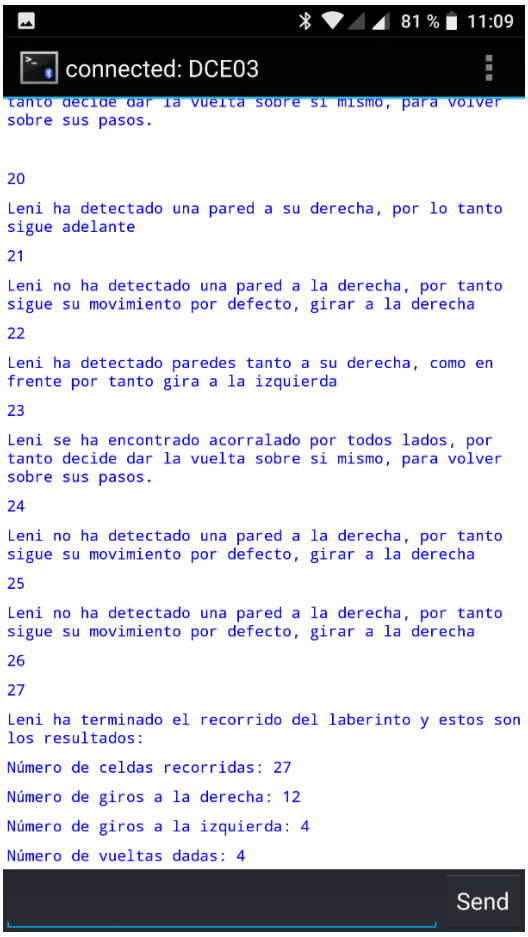
LENI - Vista trasera 1

8.3. Vídeo y comprobación de funcionamiento

LENI recorriendo un laberinto de 5x5 (Click en la imagen superior para ver el vídeo)



Móvil - Al comienzo del recorrido



Móvil - Final del recorrido

9. PRUEBAS

ID	CATEGORIA	DESCRIPCIÓN	Superado
RF01	Movimiento	El robot debe ser capaz de moverse	Sí
RF02	Movimiento	El robot debe ser capaz de pivotar 90º a derecha	Sí
RF03	Movimiento	El robot debe ser capaz de pivotar 90º a izquierda	Sí
RF04	Movimiento	El robot debe ser capaz de avanzar en línea recta	Sí
RF05	Movimiento	El robot avanza adecuadamente hasta la siguiente celda	Sí
RF06	Detección	El robot debe ser capaz de detectar una pared frontal	Sí
RF07	Detección	El robot debe ser capaz de detectar una pared lateral derecha	Sí
RF08	Detección	El robot debe ser capaz de detectar una pared lateral izquierda	Sí
RF09	Detección	El robot debe ser capaz de detectar la transición entre celdas	Sí
RF10	Detección	El robot debe ser capaz de detectar la celda de salida	Sí
RF11	Resolución	El robot almacena información sobre las celdas del laberinto	Sí
RF12	Resolución	El robot es capaz de decidir el siguiente movimiento en base a la información sobre la celda	Sí
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	Sí
RF15	Información	El robot envía al PC información sobre el número de celdas recorridas	Sí
RF16	Información	El robot envía al PC información sobre obstáculos en cada celda	Sí
RF17	Información	El robot envía al PC información sobre la velocidad de movimiento	Sí
RF18	Información	El robot envía al PC información sobre la distancia que lleva recorrida	Sí
RF19	Información	El robot envía al PC información sobre el tiempo transcurrido desde la entrada al laberinto	No
RF20	Información	El robot envía al PC información sobre la trayectoria ejecutada	Sí
RF21	Información	El robot envía al PC información sobre el número de celdas recorridas	Sí
RF22	Información	El PC muestra una representación gráfica del laberinto	No
RF23	Usuario	Interfaz gráfica en PC que recopile información	No
RF24	Pruebas	Incluir modo test al arranque del robot	No

10. MEJORAS

Se han optado por diversas mejoras, respecto a los sensores que se detallan a continuación:

- Reubicación de los sensores SHARP: Estos sensores iban colocados en un principio en las paredes laterales del robot por cuestiones de estética y facilidad. Esta opción causaba problemas a la hora de comprobar las paredes ya que los sensores SHARP son incapaces de medir distancias menores a 5 cm, por lo que se decidió su instalación en el eje central del robot con la ayuda de una pieza auxiliar. También se podría haber optado por instalar los sensores en la parte delantera, pero esto conllevaba a reimprimir la estructura del robot, por lo que se descartó dicha opción.
- Sustitución del sensor CNY70 delantero derecho, al causar problemas de medición.
- Superficie de poliespán para evitar el contacto de la estructura con la placa Arduino Leonardo y las ruedas delanteras.