Memoria del proyecto final de la asignatura Diseño de Computadores Empotrados

Corbacho Sánchez, Manuel Jesús Arias Gomez-Calcerrada, Jose Joaquín

6 de febrero de 2018

1. Introducción

1.1. Objetivo

Robot móvil capaz de recorrer un laberinto de 5x5 celdas tratando de buscar una casilla objetivo para posteriormente volver a la casilla de salida.

Las acciones llevadas a cabo por el robot para conseguir su objetivo son:

- Movimientos
  - El robot es capaz de avanzar hacía delante en linea recta, teniendo la capacidad de corregir su trayectoría gracias a los sensores incorporados.
  - El robot es capaz de girar 90 grados a la izquierda o a la derecha, además combinando 2 veces uno de estos giros puede girar 180 grados.
- Detección
  - El robot puede detectar si la casilla sobre la que se encuentra es la casilla objetivo, cuya superficie es de color negro, detectando negro sobre los 3 CNY incorporados.
  - El robot tiene la capacidad de leer cada CNY por separado para saber si cada uno está sobre blanco o sobre negro.
  - El robot es capaz de detectar cuando ha cambiado de casilla.
  - El robot puede alinearse sobre las lineas negras para poder centrarse antes de cambiar de dirección.
  - El robot detecta la distancia entre este y las paredes gracias a los sensores Sharp incorporados en los laterales.
  - El robot puede conocer la distancia entre este y la pared que se encuentra delante suya, gracias al sensor de ultrasonidos incorporado en el frontal de este.
- Recogida de información
  - El robot almacena en un vector en qué dirección se ha avanzado en el último cambio de casilla($0 \rightarrow$ Arriba, $1 \rightarrow$ Abajo, $2 \rightarrow$ Izquierda, $3 \rightarrow$ Derecha).
  - Si el robot da la vuelta para coger otra dirección en un giro anterior, este descarta los movimientos del vector.
  - El robot almacena cuántas casillas se ha movido para poder actualizar el vector de movimientos.
- Algoritmo de resolución del laberinto
  - Para resolver el laberinto el robot aplica el algoritmo de la mano derecha, que consisten en mantenerse pegado a la pared por la derecha(como si tuviera una mano, la colocara en la pared y no la separara mientras avanza por el laberinto.
  - Para salir del laberinto, lo que hace es aplicar los movimientos contrarios a los almacenados en el vector que lo llevaron a la casilla objetivo .
- Monitorización de información
  - El robot lee continuamente los valores de los sensores y actua en consecuencia de estos.
  - Se ha emulado un diferencial para los motores usando los sensores SHARP colocados en los laterales del robot para poder corregir su trayectoria si este avanza de forma que vaya a chocarse con alguna pared.

1.2. Hardware empleado

Para el robot se ha empleado una placa arduino, sensores de infrarrojos, sensores de ultrasonidos y sensores

1.2.1. Arduino Leonardo

Para ello, se utilizará una placa basado en microcontrolador, Arduino Leonardo, y para programarla un ordenador personal. A continuación su página de referencias:

- https://store.arduino.cc/arduino-leonardo-with-headers

1.2.2. Sensores

1.2.2.1. Sensores Sharp

El robot dispone de dos sensores Sharp, que usan infrarrojos para medir la distancia hasta un obstáculo, colocados en los laterales, su rango de actuación correcto va desde 4 a 30cm

1.2.2.2. Sensor HC-SR04

El robot dispone de un sensor de ultrasonidos HC-SR04 en el frontal para detectar obstáculos en su trayectoria. Su rango teórico es de 2cm a 400 cm.

1.2.2.3. Sensores CNY70

El robot dispone en la plataforma proporcionada de 3 sensores de infrarrojos CNY70 que se usan para identificar las bandas negras que representan las casillas en el suelo, o la casilla objetivo que es negra, al contrario que las demás que son blancas.

1.2.3. Actuadores

El robot dispone de un botón de encendido para iniciar su recorrido del laberinto.

1.2.4. Elementos de Comunicación

Se nos ha proporcionado un módulo bluetooth HC-06 que nos permite aprovechar dispositivos con este tipo de comunicación para enviar y recibir información con el robot.

1.2.5. Alimentación

El robot se alimentará a traves de 6 pilas de 1.5V que generan una salida de 9.0V a máxima carga.

1.3. Software empleado

1.3.1. Arduino IDE

Para realizar la programación y la carga del programa de la placa Arduino Leonardo proporcionado se ha empleado el IDE que proporcionan los creadores de Arduino.

1.3.2. Python

Para la aplicación de escritorio se ha empleado el lenguaje de programación Python en su versión 2.7.

1.3.3. ATOM

Para la escritura del código que no se cargue en la placa Arduino se ha utilizado el Editor de Texto ATOM.

2. Especificación de requisitos

2.1. Requisitos Funcionales

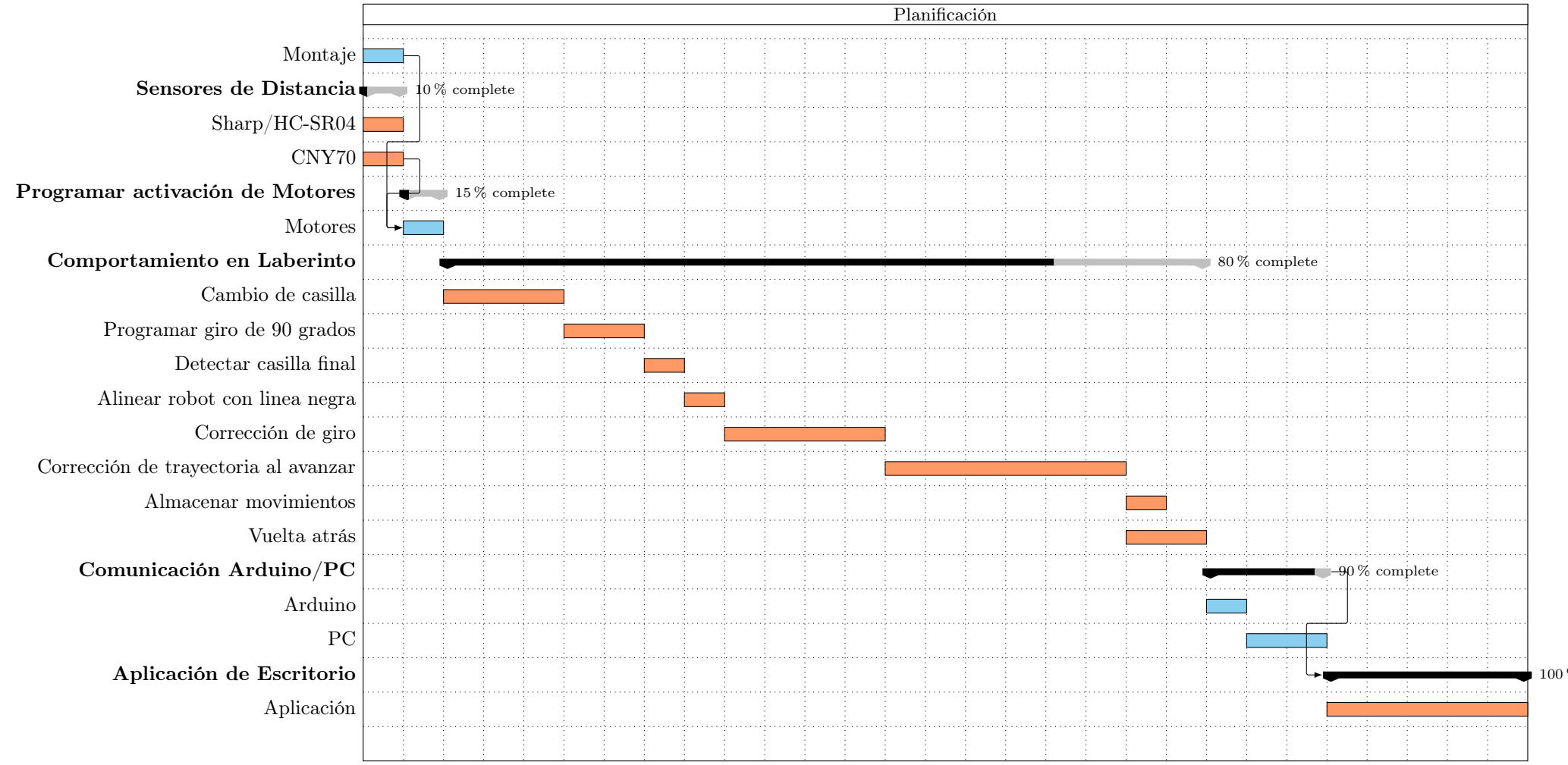
ID	Categoria	Descripción
RF01	Movimiento	El robot debe ser capaz de moverse
RF02	Movimiento	El robot debe ser capaz de pivotar 90º a derecha
RF03	Movimiento	El robot debe ser capaz de pivotar 90º a izquierda
RF04	Movimiento	El robot debe ser capaz de avanzar en línea recta
RF05	Movimiento	El robot avanza 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	Usuarios	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	Laberinto 5x5celdas de 20x20x15cm	Robot compacto
RNF02	Consumo	Condicionado por la batería disponible. 6 pilas de 1.5V	
RNF03	Errores	El robot choca contra una pared	
RNF04	Errores	Batería a punto de agotarse	Cambiar pilas
RNF05	Errores	El robot gira continuamente	
RNF01	Errores	El robot sobrepasa 20 minutos sin conseguir salir	

3. Planificación

3.1. Diagrama de gantt



3.2. Número de horas dedicadas a cada tarea

Las tareas son las especificadas en el diagrama de gantt

Tarea	Horas dedicadas
Sharp/HC-SR04	3
CNY70	2
Motores	2
Cambio de casilla	1
Programar giro de 90 grados	5
Detectar casilla final	1
Alinear robot con linea negra	2
Corrección de giro	6
Corrección de trayectoria al avanzar	30
Almacenar movimientos	1
Vuelta atrás	12
Arduino	3
PC	18
Aplicación	9
Total	95

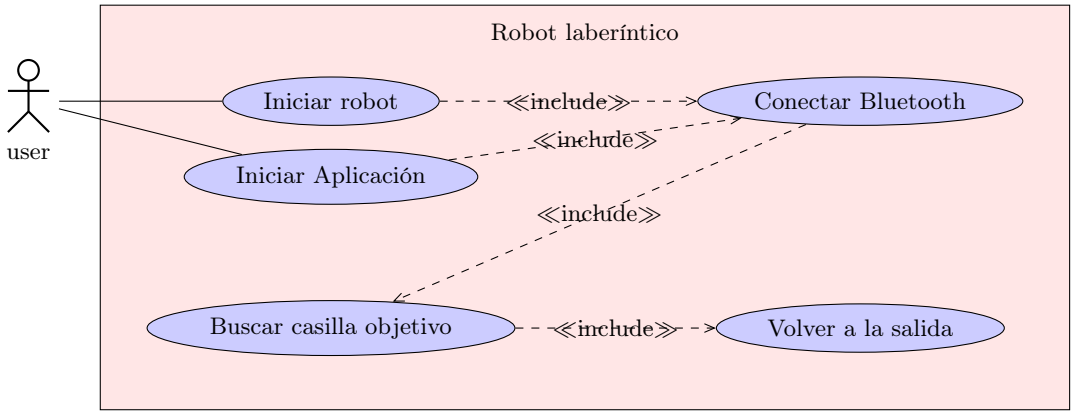
4. Presupuesto

En caso de usar un pack en el artículo aparecerá el número de unidades del pack.

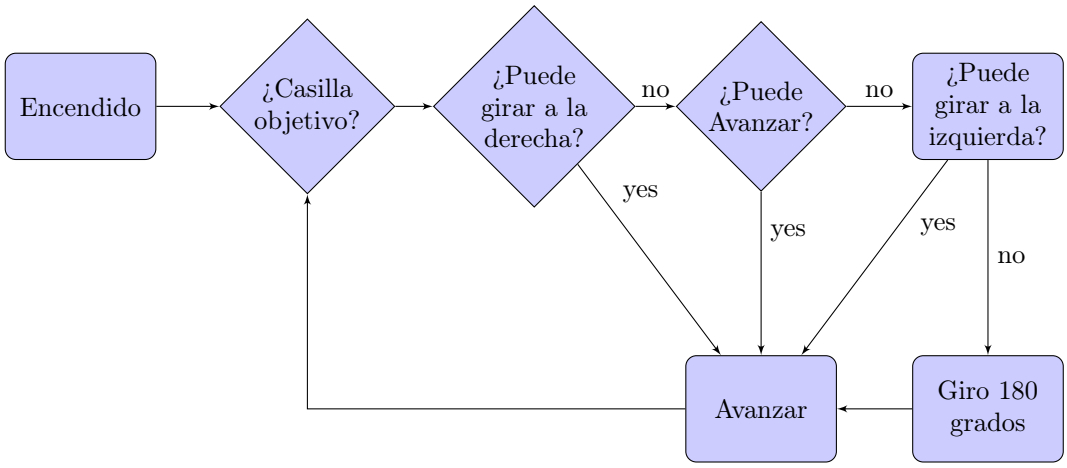
Unidades	**Artículo**	**P.U(€)**	**Total(€)**	**Enlace**
1	5x CNY70	3.10	3.10	`http://amzn.eu/04Bca91`
1	HC-SR04	1.36	1.36	`http://amzn.eu/7xnxkuq`
1	Arduino Leonardo	31.29	31.29	`http://amzn.eu/aI4n3g3`
2	Sensor Sharp	12.62	25.24	`http://amzn.eu/adqckr1`
1	Módulo Bluetooth HC-06	8.49	8.49	`http://amzn.eu/aAHWAI2`
1	Pack de cables de Conexión	7.09	7.09	`http://amzn.eu/7aVOmfy`
211	Horas de trabajo	20	4220	
1	Plataforma Proporcionada	30	30	
1	Impresión de piezas 3d(5gramos)	0.13	0.13	
			4326.70	

5. Análisis

5.1. Casos de uso



5.2. Diagrama de flujo



6. Diseño

6.1. Estructura

```
/**
 *  Authors:
 *      Arias Gomez-Calcerrada, Jose Joaquin
 *      Corbacho Sanchez, Manuel Jesus
 *  Arduino model: Leonardo
*/

/**
* Wheels on the back of the robot, ball in the front
* Control of the floor using 3 CNY70 sensors
* 1 in the front.
* 2 in the back
*/

#include <SoftwareSerial.h>
SoftwareSerial bluetooth(2,3);

/*Constantes y variables*/
const float ResolutionADC=0.00488; //4.88mV
float distance;
unsigned long time_bounce;
const int tiempo_giro=700;
int Value_SharpR=0;
int Value_SharpL=0;
float VoltageR, VoltageL;
float Voltage_CNY_Back_L=0.0f;
float Voltage_CNY_Back_R=0.0f;
float Voltage_CNY_Front=0.0f;
int tiempo=0;
int casillas=0;
//
int antPin2=1;
int antPin3=1;
int transicionPin2=0;
int transicionPin3=0;
//
int contadorCeldas=0;
bool espera=false;
bool derecha=false;
bool senal_derecha=false;
```

```
41 int ultimo_movimiento=0; //-1->No asignado 0->Arriba 1->Abajo 2->
       Izquierda 3->Derecha
42 int indice=0;
43 int* vector;
44 bool terminar=false;
45 int diferencia;
46 int estado=1; /* 1->Dentro de una casilla; 2->Tocando con sensor del
       frente; 3->Tocando con sensor de atras */
47 //Prueba
48 bool parar=false;
49 //
50 int Value_CNY_Back_L=0;
51 int Value_CNY_Back_R=0;
52 int Value_CNY_Front=0;
53
54 /* CNY pins */
55 const int CNY_Back_L=A0;
56 const int CNY_Back_R=A1;
57 const int CNY_Front=A5;
58 /* Motor pins */
59 const int pin1Left_Motor= 9;
60 const int pin2Left_Motor=10;
61 const int pin1RightMotor= 5;
62 const int pin2RightMotor= 6;
63 /* Sharp pins */
64 const int SharpR=A3;
65 const int SharpL=A8;
66 /* Ultrasonic pins */
67 const int Trigger=11;
68 const int Echo=7;
69
70 void stop() {
71     analogWrite(pin2Left_Motor, 0);
72     analogWrite(pin1RightMotor, 0);
73     analogWrite(pin1Left_Motor, 0);
74     analogWrite(pin2RightMotor, 0);
75 }
76
77 void stopRight() {
78     analogWrite(pin1RightMotor, 0);
79     analogWrite(pin2RightMotor, 0);
80 }
81
82 void stopLeft() {
83     analogWrite(pin2Left_Motor, 0);
84     analogWrite(pin1Left_Motor, 0);
85 }
86
87 void forward(int AnalogValue) {
88     stop();
89     //Suponiendo una diferencia rotacional entre ruedas del 30 %
90     diferencia=0.2*AnalogValue;
91     analogWrite(pin2Left_Motor, 0);
92     analogWrite(pin1RightMotor, 0);
93     analogWrite(pin1Left_Motor, AnalogValue-diferencia);
94     analogWrite(pin2RightMotor, AnalogValue);
95 }
96
97 void tackRight() {
98     analogWrite(pin2Left_Motor, 0);
99     analogWrite(pin1RightMotor, 0);
100    analogWrite(pin1Left_Motor, 128);
101    analogWrite(pin2RightMotor, 255);
```

```
}

void tackLeft() {
    analogWrite(pin2Left_Motor, 0);
    analogWrite(pin1RightMotor, 0);
    analogWrite(pin1Left_Motor, 255);
    analogWrite(pin2RightMotor, 128);
}

void back(int AnalogValue) {
    stop();
    analogWrite(pin1Left_Motor, 0);
    analogWrite(pin2RightMotor, 0);
    analogWrite(pin2Left_Motor, AnalogValue);
    analogWrite(pin1RightMotor, AnalogValue);
}

void backLeft() {
    stop();
    analogWrite(pin1Left_Motor, 0);
    analogWrite(pin2RightMotor, 0);
    analogWrite(pin2Left_Motor, 255);
    analogWrite(pin1RightMotor, 0);
}

void backRight() {
    stop();
    analogWrite(pin1Left_Motor, 0);
    analogWrite(pin2RightMotor, 0);
    analogWrite(pin2Left_Motor, 0);
    analogWrite(pin1RightMotor, 255);
}

void turnRight(int AnalogValue=255) {
    analogWrite(pin2Left_Motor, 0);
    analogWrite(pin2RightMotor, 0);
    analogWrite(pin1Left_Motor, 0);
    analogWrite(pin1RightMotor, AnalogValue);
}

void turnLeft(int AnalogValue=255) {
    analogWrite(pin2Left_Motor, 0);
    analogWrite(pin2RightMotor, 0);
    analogWrite(pin1Left_Motor, AnalogValue);
    analogWrite(pin1RightMotor, 0);
}

void spinL(int AnalogValue=255) {
    analogWrite(pin2Left_Motor, 0);
    analogWrite(pin2RightMotor, 0);
    analogWrite(pin1Left_Motor, AnalogValue);
    analogWrite(pin1RightMotor, AnalogValue);
}

void spinR(int AnalogValue=255) {
    analogWrite(pin1Left_Motor, 0);
    analogWrite(pin1RightMotor, 0);
    analogWrite(pin2Left_Motor, AnalogValue);
    analogWrite(pin2RightMotor, AnalogValue);
}

bool lineFoundFront() {
    //Read CNY values
```

```
165     Value_CNY_Back_L = analogRead(CNY_Back_L);
166     Value_CNY_Back_R = analogRead(CNY_Back_R);
167     Value_CNY_Front = analogRead(CNY_Front);
168 
169     //Voltage calculation
170     Voltage_CNY_Back_L = Value_CNY_Back_L*ResolutionADC;
171     Voltage_CNY_Back_R = Value_CNY_Back_R*ResolutionADC;
172     Voltage_CNY_Front = Value_CNY_Front*ResolutionADC;
173 
174     bool found=false;
175     if(Voltage_CNY_Front>3.4 && (Voltage_CNY_Front-min(
            Voltage_CNY_Back_L, Voltage_CNY_Back_R))>1) {
176       found=true;
177     }
178     return found;
179 }
180 
181 bool lineFoundBack() {
182     //Read CNY values
183     Value_CNY_Back_L = analogRead(CNY_Back_L);
184     Value_CNY_Back_R = analogRead(CNY_Back_R);
185     Value_CNY_Front = analogRead(CNY_Front);
186 
187     //Voltage calculation
188     Voltage_CNY_Back_L = Value_CNY_Back_L*ResolutionADC;
189     Voltage_CNY_Back_R = Value_CNY_Back_R*ResolutionADC;
190     Voltage_CNY_Front = Value_CNY_Front*ResolutionADC;
191 
192     bool found=false;
193     if((Voltage_CNY_Back_L>3.4 || Voltage_CNY_Back_R>3.4) && (max(
            Voltage_CNY_Back_L, Voltage_CNY_Back_R)-Voltage_CNY_Front)>1) {
194       found=true;
195     }
196     return found;
197 }
198 
199 bool lineFoundBack_L() {
200     //Read CNY values
201     Value_CNY_Back_L = analogRead(CNY_Back_L);
202     Value_CNY_Back_R = analogRead(CNY_Back_R);
203     Value_CNY_Front = analogRead(CNY_Front);
204 
205     //Voltage calculation
206     Voltage_CNY_Back_L = Value_CNY_Back_L*ResolutionADC;
207     Voltage_CNY_Back_R = Value_CNY_Back_R*ResolutionADC;
208     Voltage_CNY_Front = Value_CNY_Front*ResolutionADC;
209 
210     bool found=false;
211     if(Voltage_CNY_Back_L>3.4 && (max(Voltage_CNY_Back_L,
            Voltage_CNY_Back_R)-Voltage_CNY_Front)>1) {
212       found=true;
213     }
214     return found;
215 }
216 
217 bool lineFoundBack_R() {
218     //Read CNY values
219     Value_CNY_Back_L = analogRead(CNY_Back_L);
220     Value_CNY_Back_R = analogRead(CNY_Back_R);
221     Value_CNY_Front = analogRead(CNY_Front);
222 
223     //Voltage calculation
224     Voltage_CNY_Back_L = Value_CNY_Back_L*ResolutionADC;
```

```
225     Voltage_CNY_Back_R = Value_CNY_Back_R*ResolutionADC;
226     Voltage_CNY_Front = Value_CNY_Front*ResolutionADC;
227
228     bool found=false;
229     if(Voltage_CNY_Back_R>3.4 && (max(Voltage_CNY_Back_L,
            Voltage_CNY_Back_R)-Voltage_CNY_Front)>1) {
230       found=true;
231     }
232     return found;
233 }
234
235 bool todoNegro() {
236     //Read CNY values
237     Value_CNY_Back_L = analogRead(CNY_Back_L);
238     Value_CNY_Back_R = analogRead(CNY_Back_R);
239     Value_CNY_Front = analogRead(CNY_Front);
240
241     //Voltage calculation
242     Voltage_CNY_Back_L = Value_CNY_Back_L*ResolutionADC;
243     Voltage_CNY_Back_R = Value_CNY_Back_R*ResolutionADC;
244     Voltage_CNY_Front = Value_CNY_Front*ResolutionADC;
245
246     bool found=false;
247     if(Voltage_CNY_Front>3.4 && Voltage_CNY_Back_L>3.4 &&
            Voltage_CNY_Back_R>3.4) {
248       found=true;
249     }
250     return found;
251 }
252
253 bool noLineFound() {
254   return !lineFoundFront() && !lineFoundBack();
255 }
256
257 double getDistance(double valor) {
258   double centimetros=0;
259   if(valor<0.5 || valor>2.7) {
260     return centimetros;
261   }
262   else {
263     centimetros=valor/13.05;
264     centimetros=(1-centimetros*0.42)/centimetros;
265     return centimetros;
266   }
267 }
268
269 void showSharpR() {
270   Value_SharpR=analogRead(SharpR);
271   VoltageR=Value_SharpR*ResolutionADC;
272   Serial.print("Sharp␣R:␣");
273   Serial.print(getDistance(VoltageR));
274   Serial.println("␣cm");
275 }
276
277 void showSharpL() {
278   Value_SharpL=analogRead(SharpL);
279   VoltageL=Value_SharpL*ResolutionADC;
280   Serial.print("Sharp␣L:␣");
281   Serial.print(getDistance(VoltageL));
282   Serial.println("␣cm");
283 }
284
285 double getSharpR() {
```

```
286   Value_SharpR=analogRead(SharpR);
287   VoltageR=Value_SharpR*ResolutionADC;
288   return getDistance(VoltageR);
289 }
290
291 double getSharpL() {
292   Value_SharpL=analogRead(SharpL);
293   VoltageL=Value_SharpL*ResolutionADC;
294   return getDistance(VoltageL);
295 }
296
297 double getUltrasonic() {
298   digitalWrite(Trigger, LOW);
299   delayMicroseconds(5);
300   digitalWrite(Trigger, HIGH);
301   delayMicroseconds(10);
302   digitalWrite(Trigger, LOW);
303   time_bounce=pulseIn(Echo, HIGH);
304   distance = 0.017 * time_bounce; //Formula para calcular la distancia
305   return distance;
306 }
307
308 void setup(){
309     //bluetooth.begin(9600); //Using for Information
310     Serial1.begin(9600);
311     pinMode(pin1Left_Motor, OUTPUT);
312     pinMode(pin2Left_Motor, OUTPUT);
313     pinMode(pin1RightMotor, OUTPUT);
314     pinMode(pin2RightMotor, OUTPUT);
315     pinMode(Trigger, OUTPUT);
316     pinMode(Echo, INPUT);
317     pinMode(2,INPUT);
318     pinMode(3,INPUT);
319     vector=new int[25]; //25 casillas como maximo de recorrido.
320     for(int i=0; i<25; i++) {
321       vector[i]=-1;  //Empiezan sin asignar
322     }
323 }
324
325 bool hasPassedBox() {
326   int estado_anterior=estado;
327   //Actualizar estado
328   switch(estado) {
329     case 1: if(lineFoundFront()) {
330               estado=2;
331             } break;
332     case 2: if(lineFoundBack() && !lineFoundFront()) {
333               estado=3;
334             } break;
335     case 3: if(!lineFoundBack()) {
336               estado=1;
337             } break;
338   }
339   //Fin actualizar estado
340   if(estado_anterior==3 && estado==1) {return true;}
341   else {return false;}
342 }
343
344 void balanceLeft() {  /* Mantener el robot con la referencia de la
      pared de la izquierda */
345   double distL=getSharpL();
346   if(distL<10) {   //Valia 6
347     //Serial1.println("VIRANDO A LA DERECHA");
```

```
348     tackRight();
349     delay(50);
350     stop();
351   }
352   else {
353     //Serial1.println("VIRANDO A LA IZQUIERDA");
354     tackLeft();
355     delay(50);
356     stop();
357   }
358 }
359
360 void balanceRight() {  /* Mantener el robot con la referencia de la
      pared de la derecha */
361   double distR=getSharpR();
362   if(distR<10) {   //Valia 6
363     tackLeft();
364     delay(50);
365     stop();
366   }
367   else {
368     tackRight();
369     delay(50);
370     stop();
371   }
372 }
373
374 void intelligentForward(int potencia=150) {
375   double distR=getSharpR();
376   double distL=getSharpL();
377   double diferencia=distR-distL;
378   double umbralp=2;
379   double umbraln=-umbralp;
380   double topeR=12;
381   double topeL=12;
382
383   if(distR>topeR || distR==0 || distL>topeL || distL==0) {
384     if((distR>topeR || distR==0) && (distL>topeL || distL==0)) {
385       forward(potencia);
386     }
387     if((distR>topeR || distR==0) && !(distL>topeL || distL==0)) {
388       balanceLeft();
389       forward(potencia);
390     }
391     if(!(distR>topeR || distR==0) && (distL>topeL || distL==0)) {
392       balanceRight();
393       forward(potencia);
394     }
395   }
396   else {
397     if(diferencia>umbralp) {
398       tackRight();
399       delay(50);
400       stop();
401     }
402     else {
403       if(diferencia<umbraln) {
404         tackLeft();
405         delay(50);
406         stop();
407       }
408       else {
409         forward(potencia);
```

```
410       }
411     }
412   }
413 }
414
415 void alinear() {
416   stop();
417   bool alineado=false;
418   while(!alineado) {
419     if(!lineFoundBack()) {
420       //Ir atras
421       back(255);
422     }
423     else {
424       if(lineFoundBack_R() && lineFoundBack_L()) {
425         //Ya esta alineado
426         alineado=true;
427         stop();
428       }
429       else {
430         if(lineFoundBack_L()) {
431           //Mover culo atras a la izquierda
432           backLeft();
433         }
434         if(lineFoundBack_R()) {
435           //Mover culo atras a la derecha
436           backRight();
437         }
438       }
439     }
440   }
441 }
442
443 bool revolucionesPin2(int transiciones) {
444   transicionPin2=0;
445   while(true) {
446     int Pin2=digitalRead(2);
447     if((antPin2==1 && Pin2==0) || (antPin2==0 && Pin2==1)) {
448       if(antPin2==1 && Pin2==0) {
449         antPin2=0;
450       }
451       else {
452         antPin2=1;
453       }
454       //Actualizar transicionPin2
455       ++transicionPin2;
456       if(transicionPin2==transiciones) {
457         return true;
458       }
459     }
460   }
461 }
462
463 bool revolucionesPin3(int transiciones) {
464   transicionPin3=0;
465   while(true) {
466     int Pin3=digitalRead(3);
467     if((antPin3==1 && Pin3==0) || (antPin3==0 && Pin3==1)) {
468       if(antPin3==1 && Pin3==0) {
469         antPin3=0;
470       }
471       else {
472         antPin3=1;
```

```
473       }
474       //Actualizar transicionPin2
475       ++transicionPin3;
476       if(transicionPin3==transiciones) {
477         return true;
478       }
479     }
480   }
481 }
482
483 void rotarDerecha(int iteraciones=10) {
484   bluetooth.write('r');
485   for(int a=0; a<iteraciones; a++) {
486     analogWrite(pin2Left_Motor, 150);
487     while(!revolucionesPin3(1)) {}
488     analogWrite(pin2RightMotor, 150);
489     while(!revolucionesPin2(1)) {}
490   }
491   stop();
492 }
493
494 void rotarIzquierda(int iteraciones=10) {
495   bluetooth.write('l');
496   for(int a=0; a<iteraciones; a++) {
497     analogWrite(pin1Left_Motor, 150);
498     while(!revolucionesPin3(1)) {}
499     analogWrite(pin1RightMotor, 150);
500     while(!revolucionesPin2(1)) {}
501   }
502   stop();
503 }
504
505 void atras(int iteraciones=7) {
506   for(int a=0; a<iteraciones; a++) {
507     bool pararIzquierda=false;
508     bool pararDerecha=false;
509     if(pararIzquierda) {
510       stopLeft();
511     }
512     else {
513       //Mover rueda izquierda durante x revoluciones
514       analogWrite(pin2Left_Motor, 255);
515       if(revolucionesPin3(1)) {
516         pararIzquierda=true;
517       }
518     }
519     if(pararDerecha) {
520       stopRight();
521     }
522     else {
523       //Mover rueda derecha durante x revoluciones
524       analogWrite(pin1RightMotor, 255);
525       if(revolucionesPin2(1)) {
526         pararDerecha=true;
527       }
528     }
529   }
530   stop();
531 }
532
533 void rotar180() {
534   bluetooth.write('b');
535   rotarIzquierda(6);
```

```
  atras();
  rotarIzquierda(17);
  estado=2;
}

int actualizar_movimiento(int ultimo_movimiento, int movimiento) {
  switch(ultimo_movimiento) {
    case 0: if(movimiento==0) {
              return 0;
            }
            if(movimiento==1) {
              return 1;
            }
            if(movimiento==2) {
              return 2;
            }
            if(movimiento==3) {
              return 3;
            }
    case 1: if(movimiento==0) {
              return 1;
            }
            if(movimiento==1) {
              return 0;
            }
            if(movimiento==2) {
              return 3;
            }
            if(movimiento==3) {
              return 2;
            }
    case 2: if(movimiento==0) {
              return 2;
            }
            if(movimiento==1) {
              return 3;
            }
            if(movimiento==2) {
              return 1;
            }
            if(movimiento==3) {
              return 0;
            }
    case 3: if(movimiento==0) {
              return 3;
            }
            if(movimiento==1) {
              return 2;
            }
            if(movimiento==2) {
              return 0;
            }
            if(movimiento==3) {
              return 1;
            }
     default: return -1; break;
  }
}

void loop(){
    ++tiempo;

    //Read CNY values
```

```
599     Value_CNY_Back_L = analogRead(CNY_Back_L);
600     Value_CNY_Back_R = analogRead(CNY_Back_R);
601     Value_CNY_Front = analogRead(CNY_Front);
602
603     if(terminar) {
604       stop();
605     }
606     else {
607       if(hasPassedBox()) {
608         Serial1.println("HA␣PASADO␣UNA␣CELDA");
609         delay(100);
610         alinear();
611         bluetooth.write('m');
612         ++contadorCeldas;
613         //Meter el ultimo movimiento que ha efectuado en el vector.
                Ejemplo (vector[i]=3; --Izquierda-- ++i;)
614         if(indice<25) {
615           vector[indice]=ultimo_movimiento;
616           ++indice;
617         }
618       }
619       if(todoNegro()) {
620         Serial1.println("HA␣PASADO␣UNA␣CELDA");
621         ++contadorCeldas;
622         if(indice<25) {
623           vector[indice]=ultimo_movimiento;
624           ++indice;
625         }
626         terminar=true;
627       }
628       if(!parar) {
629         //Ir recto
630         ultimo_movimiento=actualizar_movimiento(ultimo_movimiento, 0);
631         intelligentForward();
632         //Si encuentra un hueco a la derecha cuando encuentra una
                liinea atras, girar a la derecha
633         double distR=getSharpR();
634         double distL=getSharpL();
635         double topeR=12;
636         double topeL=12;
637         if((distR>topeR || distR==0) && lineFoundBack()) {
638           senal_derecha=true;
639         }
640         if(senal_derecha && lineFoundFront()) {
641           derecha=true;
642           parar=true;
643         }
644         if(getUltrasonic()<=5 && tiempo>50) {
645           parar=true;
646           derecha=false;
647           senal_derecha=false;
648         }
649       }
650       else {
651         if(derecha) {
652           //Girar a la derecha
653           ultimo_movimiento=actualizar_movimiento(ultimo_movimiento, 3)
                ;
654           alinear();
655           intelligentForward(255);
656           delay(500);
657           stop();
658           rotarDerecha();
```

```
659           parar=false;
660           derecha=false;
661           senal_derecha=false;
662         }
663         else {
664           //alinear();
665           while(getUltrasonic()>6) {
666             intelligentForward();
667           }
668           stop();
669           //
670           alinear();
671           intelligentForward(255);
672           delay(500);
673           stop();
674           //
675           //Analizar hacia que lado girar
676           double distR=getSharpR();
677           double distL=getSharpL();
678           double topeR=12;
679           double topeL=12;
680           if(distR>topeR || distR==0 || distL>topeL || distL==0) {
681             if(distR>topeR || distR==0) {
682               //Girar a la derecha
683               ultimo_movimiento=actualizar_movimiento(ultimo_movimiento
                      , 3);
684               delay(100);
685               rotarDerecha();
686               parar=false;
687             }
688             if(!(distR>topeR || distR==0) && (distL>topeL || distL==0))
                    {
689               //Girar a la izquierda
690               ultimo_movimiento=actualizar_movimiento(ultimo_movimiento
                      , 2);
691               delay(100);
692               rotarIzquierda();
693               parar=false;
694             }
695           }
696           else {
697             //Dar media vuelta
698             ultimo_movimiento=actualizar_movimiento(ultimo_movimiento,
                    1);
699             delay(100);
700             rotar180();
701             parar=false;
702           }
703         }
704       }
705     }
706 }
```

6.2. Plan de pruebas

Para la obtención de las distancias de cada uno de los Sharp en centimetros, se usa el siguiente código.

```
1 double getDistance(double valor) {
2   double centimetros=0;
3   if(valor<0.5 || valor>2.7) {
4     return centimetros;
5   }
6   else {
7     centimetros=valor/13.05;
```

```
    centimetros=(1-centimetros*0.42)/centimetros;
    return centimetros;
  }
}
```

A la cual se le pasa por valor el voltaje obtenido por dicho Sharp. Para el valor de los CNYs, se usa una constante de la siguiente forma.

```
//Read CNY values
Value_CNY_Back_L = analogRead(CNY_Back_L);
//Voltage calculation
Voltage_CNY_Back_L = Value_CNY_Back_L*ResolutionADC;
```

Para obtener la distancia obtenida por el sensor Ultrasónico, usamos el siguiente código.

```
double getUltrasonic() {
  digitalWrite(Trigger, LOW);
  delayMicroseconds(5);
  digitalWrite(Trigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(Trigger, LOW);
  time_bounce=pulseIn(Echo, HIGH);
  distance = 0.017 * time_bounce; //Formula para calcular la distancia
  return distance;
}
```

7. Implementación

7.1. Librerias

- SoftwareSerial : Libreria que permite la comunicación serial con otros dispositivos.

  Su página de referencia es `https://www.arduino.cc/en/Reference/SoftwareSerial`

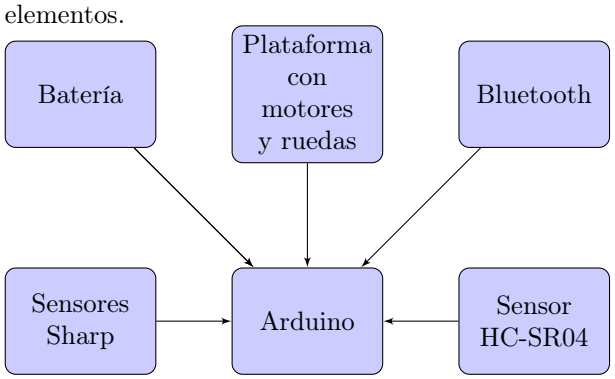
7.2. Apuntes sobre el código

- Se ha realizado un diferencial por software para permitir al robot avanzar en linea recta, intentando mantenerse en el centro de la casilla entre las paredes.
- Se han tenido en cuenta situaciones en las que sólo hay paredes en 1 lado del robot para decidir hacía donde debe avanzar este, además de mantener la distancia a las paredes.
- El robot al pasar una celda se alinea para poder avanzar hacía delante con la linea negra que indica el cambio de casilla.
- Se han usado los encoder para que independientemente de la potencia, el robot gire un número de grados.

8. Montaje

8.1. Diagrama de bloques

A continuación podemos observar el diagrama de bloques que representa la interconexión de los elementos.



8.2. Fotos del robot

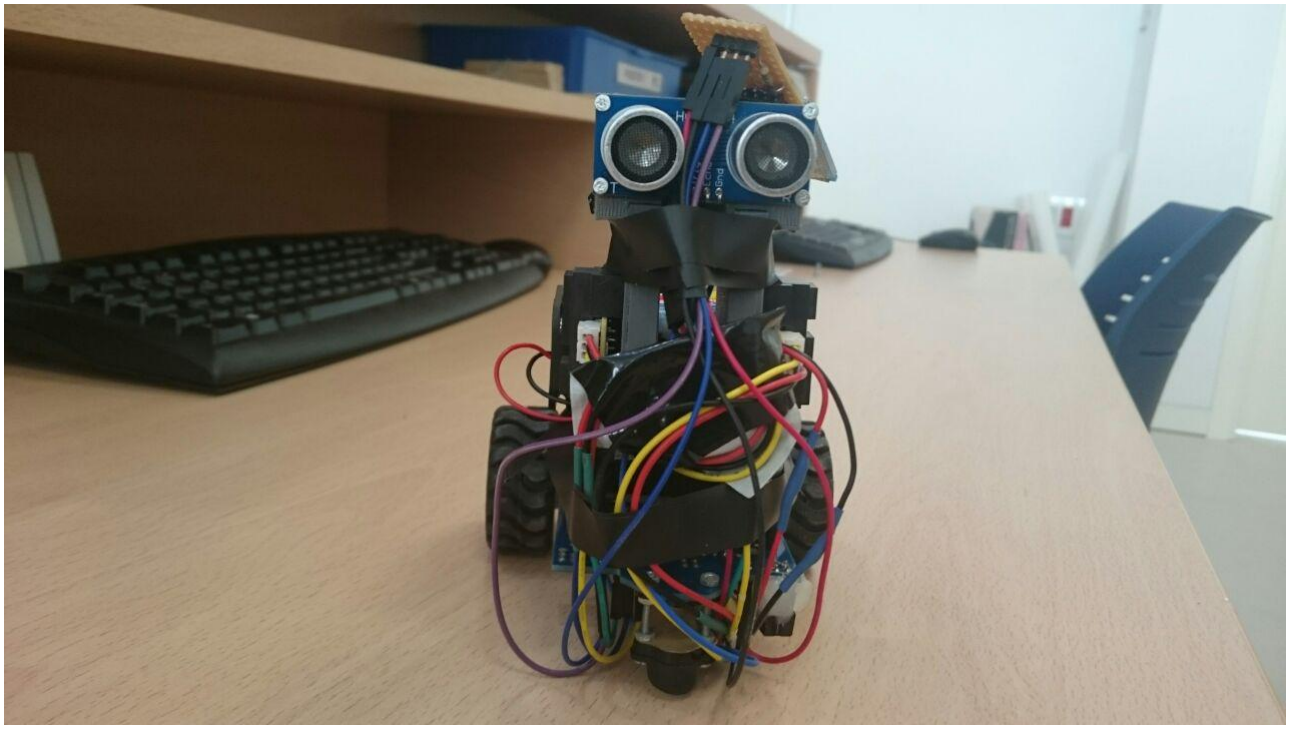


Figura 1: Vista Frontal

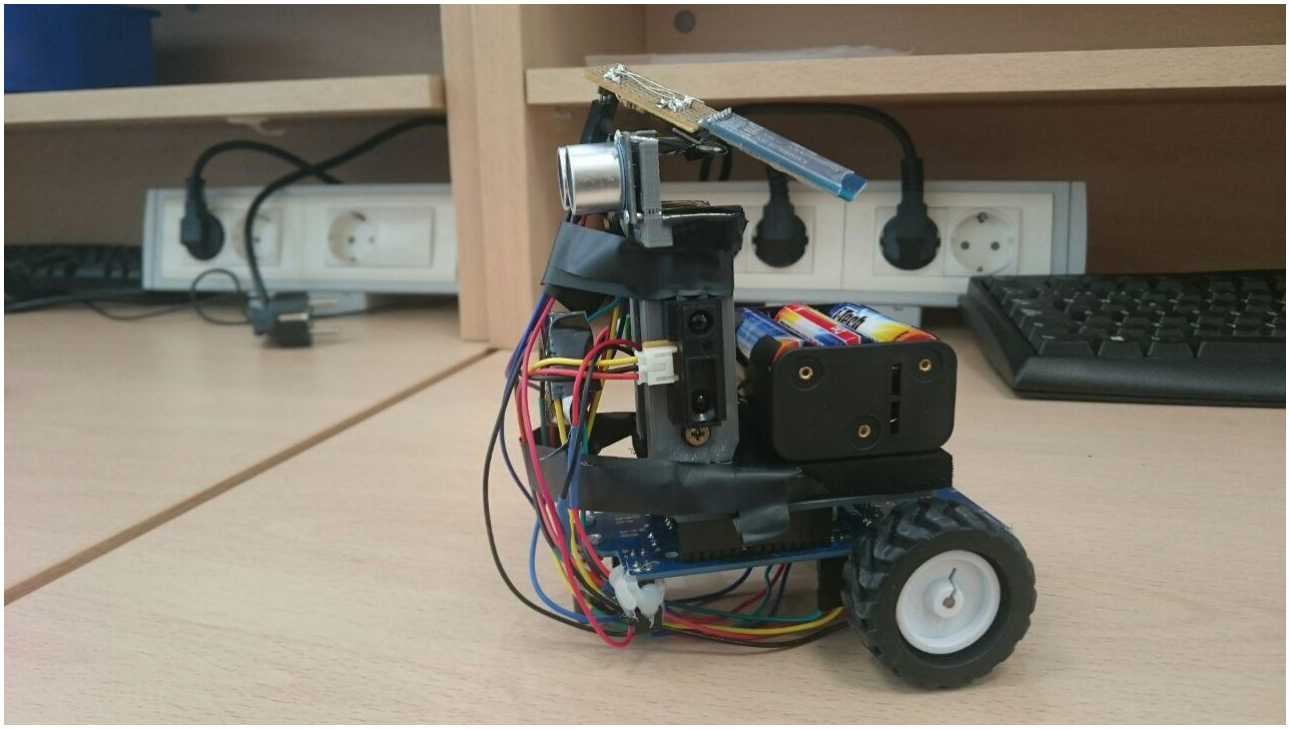


Figura 2: Vista de Perfil

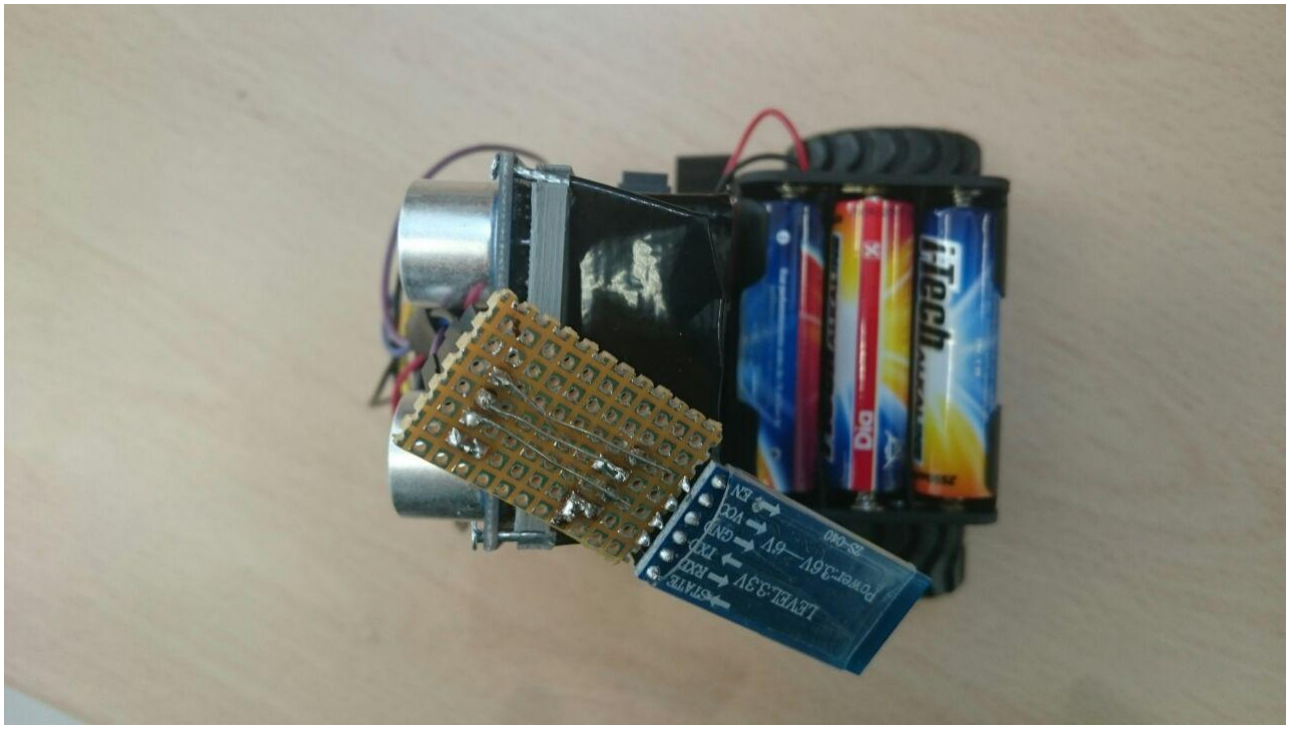


Figura 3: Planta

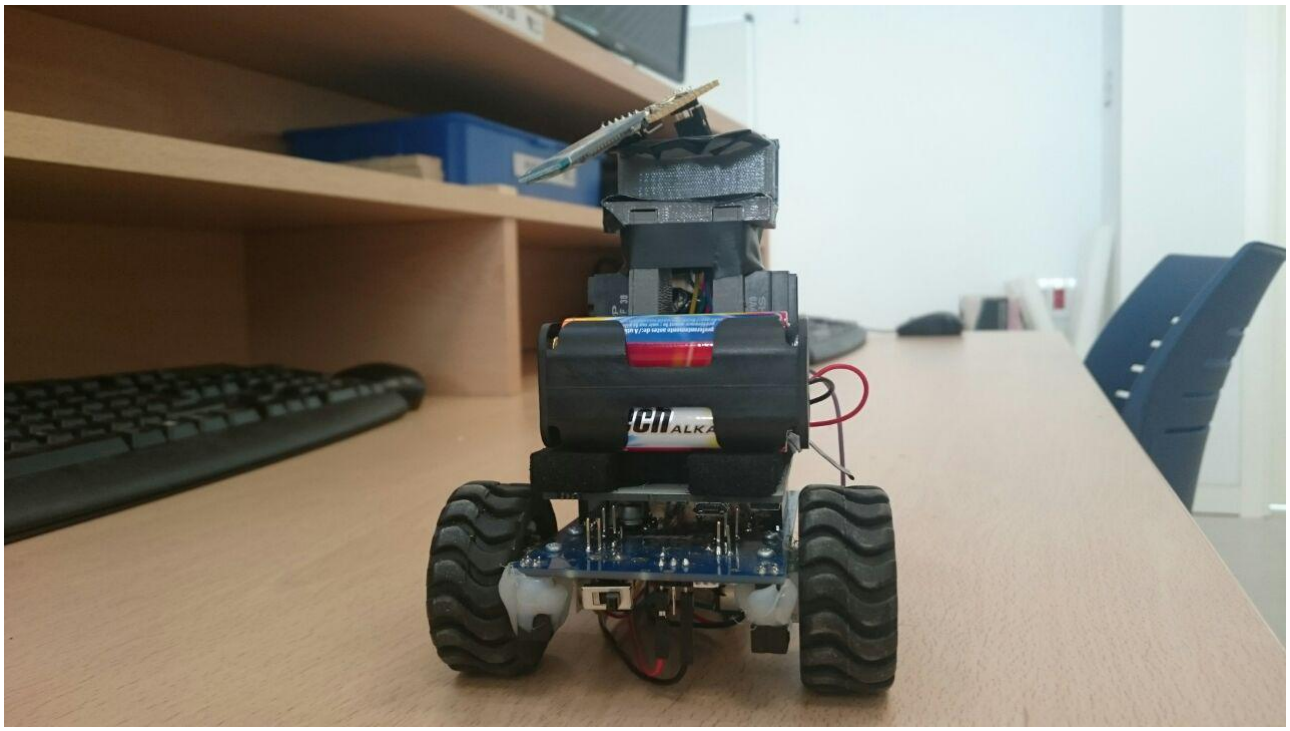


Figura 4: Vista Trasera

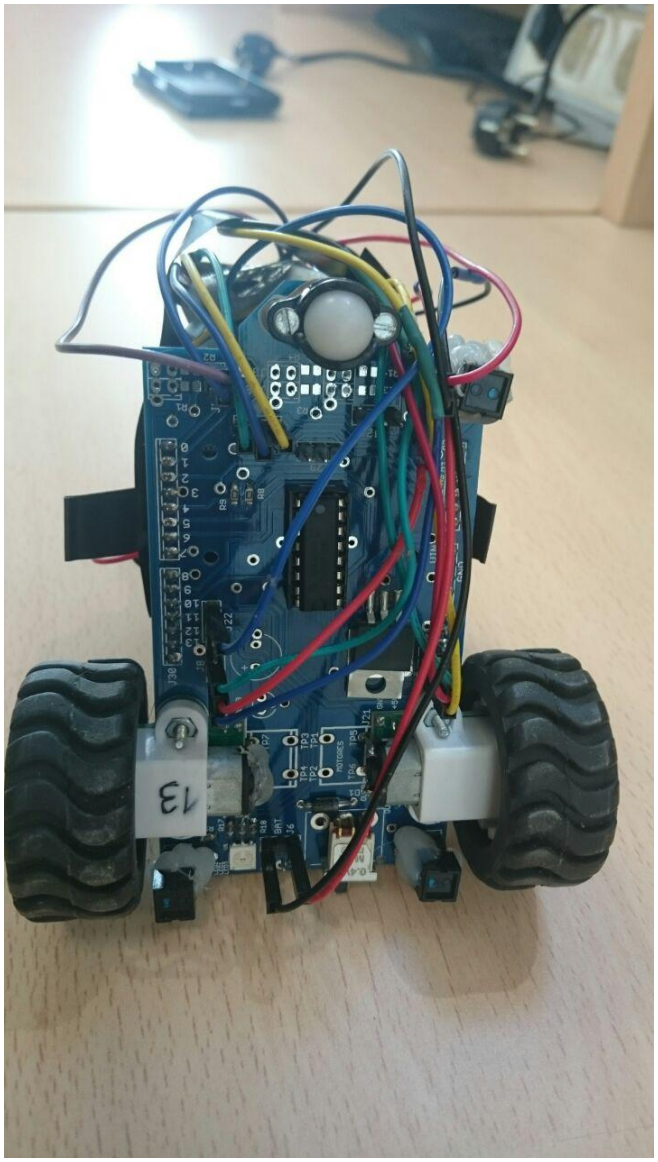


Figura 5: Vista Inferior

8.3. Capturas de la aplicación

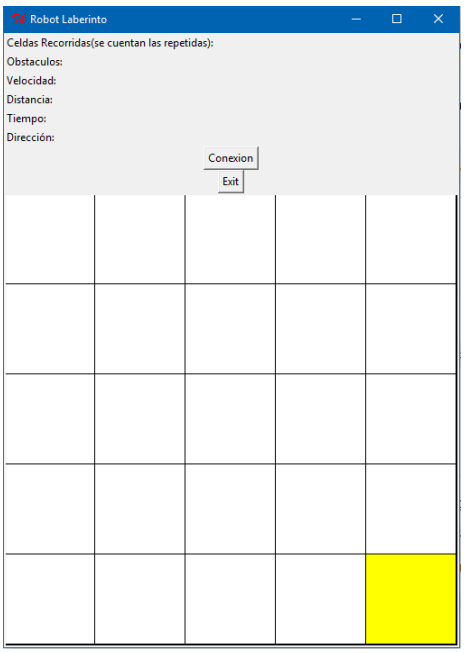


Figura 6: Aplicación de Python

9. Pruebas

ID del Requisito	Comprobación	Superado
RF01	Comprobado en laberinto	Sí
RF02	Comprobado en laberinto	Sí
RF03	Comprobado en laberinto	Sí
RF04	Comprobado en laberinto	Sí
RF05	Comprobado en laberinto	Sí
RF06	Comprobado en laberinto	Sí
RF07	Comprobado en laberinto	Sí
RF08	Comprobado en laberinto	Sí
RF09	Comprobado en laberinto	Sí
RF10	Comprobado en laberinto	Sí
RF11	Incremento de una variable en la función de cambio de celda	sí
RF12	Comprobado en laberinto	Sí
RF13	Comprobado en laberinto	Sí
RF14	Comprobado en laberinto	Sí
RF15	Aplicación	Sí
RF16	Aplicación	Sí
RF17	Aplicación	Sí
RF18	Aplicación	Sí
RF19	Aplicación	Sí
RF20	Aplicación	Sí
RF21	Aplicación	Sí(repetido en RF15)
RF22	Aplicación	Sí
RF23	Aplicación	Sí
RF24		

10. Mejoras

- Se ha añadido mediante programación un diferencial para los motores para permitir la corrección de la trayectoria del robot y que este avance en línea recta.