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: Izquierda, Derecha, Adelante y Atras
- Detección: Detección de obstáculos delante y hacia los lados y detección del cambio de color en la base del robot
- Recogida de información: Casillas por las que pasamos y obstáculos encontrados
- Algoritmo de resolución del laberinto: Vuelta atrás no recursiva
- Monitorización de información: A traves de un modulo Bluetooth para recoger información para registrarlo en la interfaz

1.2. Hardware empleado

1.2.1. Arduino Leonardo

Para ello, se utilizará una placa basado en microcontrolador, Arduino Leonardo, y un ordenador personal. El ordenador cumple con las especificaciones basicas para que funcione el IDE de Arduino y pueda trasmitir información entre él y el microcontrolador Arduino Leonardo.

1.2.2. Sensores

Cuatro CNY70, dos Sharp 2D120X, un HC-SR04 y un Microswitch

1.2.3. Actuadores

Dos Montores DC-3V y un Driver L293D

1.2.4. Elementos de comunicación

Modulo Bluetooth HC-06

1.2.5. Alimentación

Baterias

1.3. Software empleado

1.3.1. Arduino IDE

1.3.2. Python 3

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

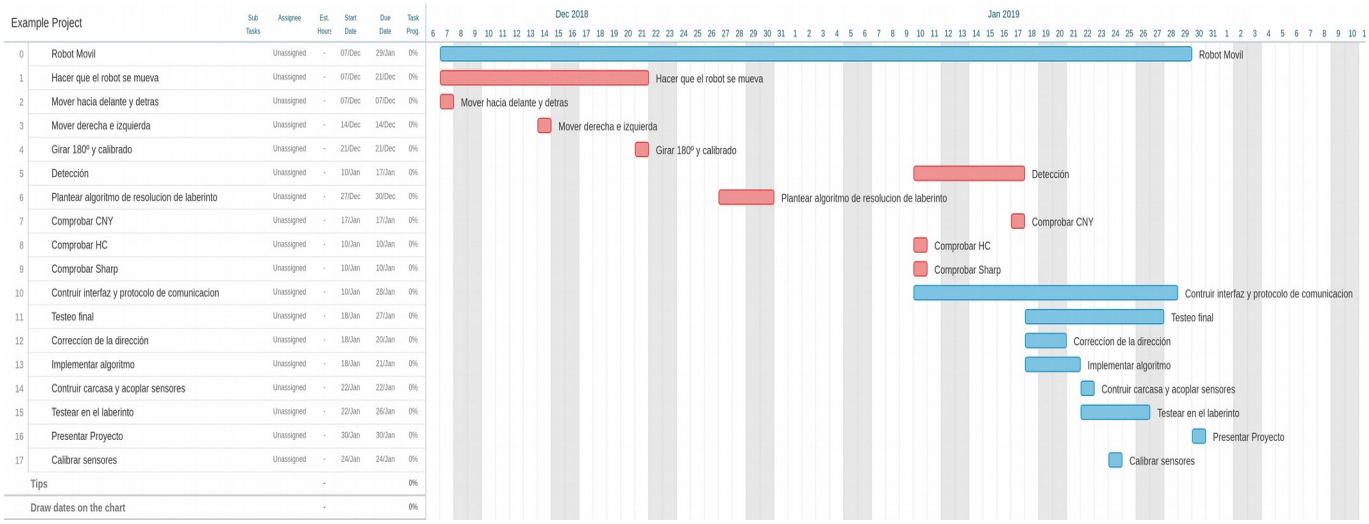
ID	CATEGORIA	DESCRIPCIÓN
RF03	Movimiento	El robot debe ser capaz de pivotar 90º a izquierda
RF04	Movimiento	El robot debe ser capaz de avanzar en línea recta
RF05	Movimiento	El robot avanza adecuadamente hasta la siguiente celda
RF06	Detección	El robot debe ser capaz de detectar una pared frontal
RF07	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
RNF0	Tamaño	Condicionado por las	

ID	CATEGORIA	DESCRIPCIÓN	ACCIÓN
1		dimensiones del laberinto. 20cm x 20cm	
RNF02	Consumo	Condicionado por la batería disponible. 6 pilas AA de 1.5 V	
RNF03	Errores	El robot choca contra una pared	< Indicar acción>
RNF04	Errores	Batería a punto de agotarse	< Indicar acción>
RNF05	Errores	El robot gira continuamente	< Indicar acción>
RNF06	Errores	El robot sobrepasa 20 minutos sin conseguir salir	< Indicar acción>

3. PLANIFICACIÓN



4. PRESUPUESTO

Aproximadamente 10€ por 2 Sharp, 4€ por 4 CNY70 , 3€ por 1 HC-SR04, 8€ por 1 HC-06, 5€ por 6 pilas AA, 40€ por 1 Arduino Leonardo y 20€ por 1 PCB.

En total serían unos 90€.

5. ANÁLISIS

5.1. Casos de uso

Calibrar Sensor Blanco:

Precondición:

-El robot está colocado en una casilla blanca.

Escenario Principal:

1-El usuario pulsa el botón de calibrar de la interfaz.

2-La interfaz envia un mensaje al robot para avisarlo de que calibre el color blanco con los sensores.

3-El robot realizar una lectura con los sensores de color.

4-El robot hace la media de esos valores y lo almacena.

5-El robot envia una señal a la interfaz avisando de que el calibrado se ha realizado.

6-La interfaz muestra un mensaje avisando al usuario de que el proceso se ha realizado.

Calibrar Sensor Negro:

Precondiciones:

-El robot está colocado sobre una casilla negra.

-Ya se ha calibrado el color blanco.

Escenario Principal:

1-El usuario pulsa el botón de calibrar de la interfaz.

2-La interfaz envia un mensaje al robot para avisarlo de que calibre el color negro con los sensores.

3-El robot realizar una lectura con los sensores de color.

4-El robot hace la media de esos valores y lo almacena.

5-El robot hace la media entre los valores blanco y negro para establecer un punto de separación entre ambos.

6-El robot envia una señal a la interfaz avisando de que el calibrado se ha realizado.

7-La interfaz muestra un mensaje avisando al usuario de que el proceso se ha realizado.

Iniciar Robot:

Precondiciones:

-El robot se encuentra en la casilla de salida.

-Se han calibrado los colores blanco y negro.

Escenario Principal:

1-El usuario pulsa el botón Start en la interfaz.

2-La interfaz envia un mensaje al robot para avisarlo de que debe comenzar el recorrido del laberinto.

3-El robot envia datos a la interfaz de los muros que va detectando y de sus movimientos.

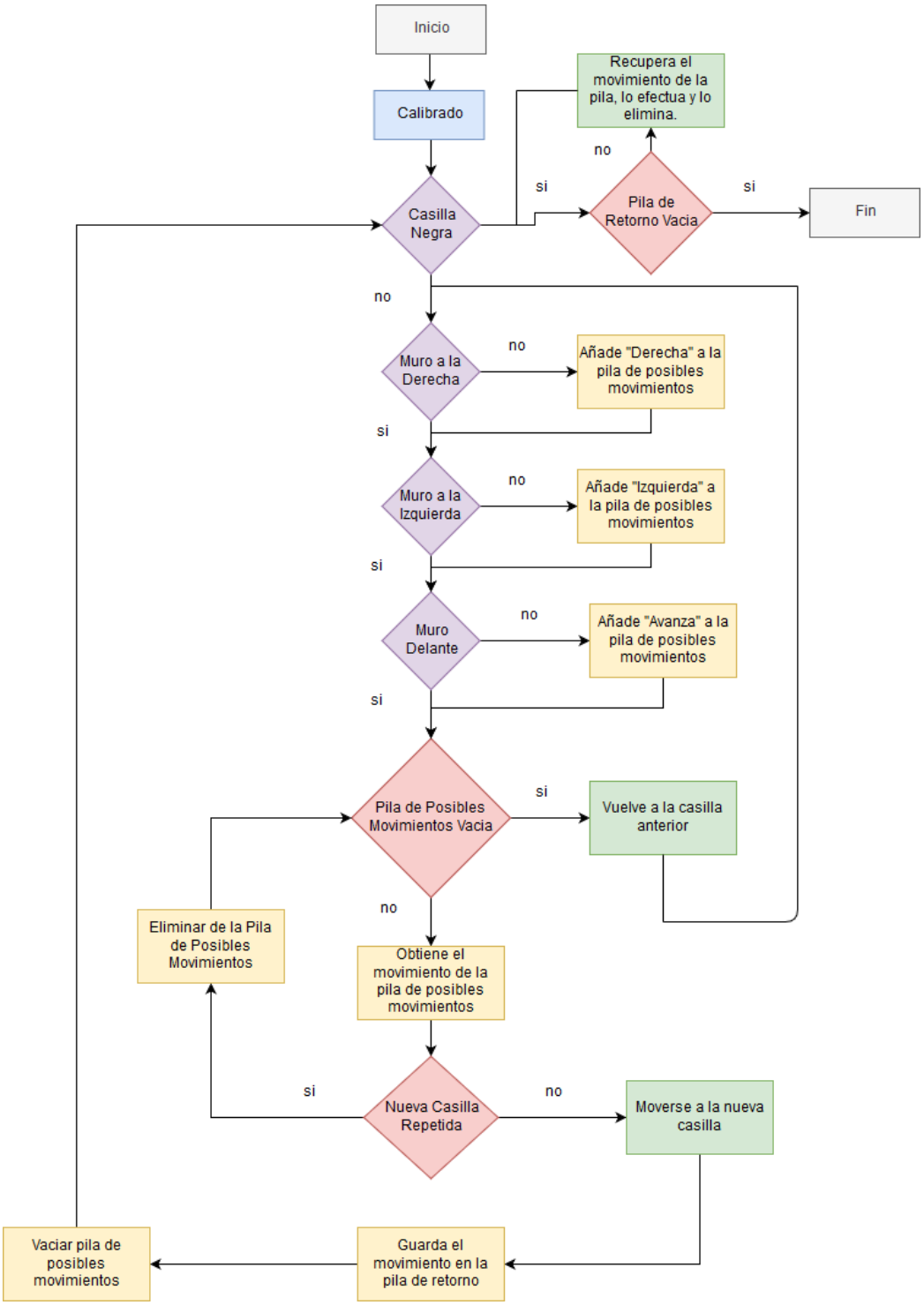
4-La interfaz actualiza la representación del laberinto con los datos que recibe del robot.

5- El robot ha terminado el recorrido.

6- El robot envia una señal a la interfaz avisando de que su ejecución ha concluido.

7-La interfaz muestra el mensaje de finalización al usuario.

5.2. Diagrama de flujo



6. DISEÑO

6.1. Estructura

```
//Código para obtener datos del CNY70

    float c;
    int CNY70(int CNY_Pin) //Blanco 0 y negro 1 provisional
    {
```

```
 const float Value_CNY_Pin=analogRead(CNY_Pin);
 const float ResolutionADC=0.0048;
 float Voltage=0.0048*Value_CNY_Pin;



 Serial1.println(Value_CNY_Pin);
 Serial1.print("Voltaje: ");
 Serial1.print(Voltage);
 Serial1.print(" V");

 if(Voltage>=c-0.5)
 {
  Serial1.println(" Negro");
  return 1;
 }
 else
 {
  Serial1.println(" Blanco");
  return 0;
 }



}

float CNY70_V(int CNY_Pin)
{
 const float Value_CNY_Pin=analogRead(CNY_Pin);
 const float ResolutionADC=0.0048;
 float Voltage=0.0048*Value_CNY_Pin;
 return Voltage;
}
//Codigo para obtener datos del los Sharp

float sharp(int Sharp_Pin)
{
 // Serial1.begin(9600);

 float Value_Sharp_Pin=analogRead(Sharp_Pin);
 float Voltage=Value_Sharp_Pin*ResolutionADC;
 Serial1.println(Value_Sharp_Pin);
 Serial1.print(" Voltage");
 Serial1.print(Voltage);
 Serial1.print(" V");
 Serial1.print(" Distancia: ");
 float distance=GetDistance(Voltage);
 Serial1.print(distance);
 //GetDistance(Voltage);
 Serial1.println(" cm");
 delay(1000);
 return distance;
}

float GetDistance(float Voltage)
{
 if(Voltage<0.5 || Voltage>2.7)return 0;
   else return (1/((Voltage+0.3568)/15.71)+0.42);
}
```

```
//Codigo para obtener datos del sensor de ultrasonido

float distance;
unsigned long time_bounce;

float ultrasonic_distance(int ultra) {
  //Serial1.begin(9600);

  pinMode(ultra,OUTPUT);

  digitalWrite(ultra,LOW);
  delayMicroseconds(5);

  digitalWrite(ultra,HIGH);
  delayMicroseconds(10);
  digitalWrite(ultra,LOW);

   pinMode(ultra,INPUT);


  time_bounce=pulseIn(ultra,HIGH);
  distance=0.017*time_bounce;

  Serial1.println("Distancia: ");
  Serial1.print(distance);
  Serial1.println(" cm");

  delay(500);
  return distance;
}

//Codigos para los movimientos basicos

        int input;

        void StopROBOT(int pin1,int pin2,int pin3,int pin4)
        {
          UnaRuedaAdelante(pin1,pin2,0);
          UnaRuedaAdelante(pin3,pin4,0);
          delay(1000);
        }

        void ForwardROBOT(int pin1,int pin2,int pin3,int pin4,int velocidad)
        {
           UnaRuedaAdelante(pin1,pin2,800);
           UnaRuedaAdelante(pin3,pin4,800);
        }

        void BackwardROBOT(int pin1,int pin2,int pin3,int pin4,int velocidad)
        {
            UnaRuedaAdetras(pin1,pin2,800);
           UnaRuedaAdetras(pin3,pin4,800);
        }

        void UnaRuedaAdelante(int pin1,int pin2,int velocidad)//Para ir hacia adelante tenemos que
llamar dos veces a esta funcion con pines distintos
        {
           input=map(velocidad,0,1023,0,254);
```

```
   analogWrite(pin1,input);
   analogWrite(pin2,0);
}

void UnaRuedaAdetras(int pin1,int pin2,int velocidad)//Para ir hacia atras tenemos que llamar
dos veces a esta funcion con pines distintos
{
   input=map(velocidad,0,1023,0,254);
   analogWrite(pin1,0);
   analogWrite(pin2,input);
}

void RightROBOT(int pin1,int pin2,int pin3,int pin4,int velocidad)//Dos primeros pines, rueda
derecha,dos ultimos, rueda izquierda
{
  input=map(velocidad,0,1023,0,254);
  UnaRuedaAdelante(pin3,pin4,velocidad);
  UnaRuedaAdetras(pin1,pin2,velocidad);
}

void LeftROBOT(int pin1,int pin2,int pin3,int pin4,int velocidad)
{
  input=map(velocidad,0,1023,0,254);
  UnaRuedaAdelante(pin1,pin2,velocidad);
  UnaRuedaAdetras(pin3,pin4,velocidad);
}


//Codigo principal

#include <StackList.h>

//Motores
const int pin1_Motor=10;
const int pin2_Motor=9;
const int pin3_Motor=5;
const int pin4_Motor=6;
//
const int pin1_CNY=A2;
const int pin2_CNY=A5;
const int pin3_CNY=A0;
const int pin4_CNY=A1;
//
const int pin1_SHARP=A3;
const int pin2_SHARP=A4;
//
const int pin_ULTRA=2;

const float ResolutionADC=0.0048;
float dist;

struct posicion
{
  int fila=0;
  int columna=0;
};
typedef struct posicion Posicion;
```

```
enum Direcc{F,L,R,B};//La N es nothing//Enviar esto como char a python
Direcc actual;
//Para los CNY, calibrado
float v_blanco=0;
float v_negro=0;

StackList <Direcc> posibles; //Almacenara maximo 3 direcc
StackList <Posicion> recorrido;

StackList <Direcc> Camino;//Almacenara el camino direccto a seguir desde la salida

Posicion pos_actual;
Posicion visitados[25];
int pos=0;
int v;//La velocidad, testear
int v_e;

bool calibrar=false;

int casilla_repetida(int ,int ,struct posicion [],int );
int es_fin(int ,int ,int ,int );


int casilla_repetida(int f,int c,struct posicion v[],int tam)
{
 int i;
 for(i=0;i<tam;++i)
 {
  if(v[i].fila==f && v[i].columna==c) return 1;
 }
 return 0;
}

void Derecha(int pin1_Motor,int pin2_Motor,int pin3_Motor,int pin4_Motor,int pin1_CNY,int
pin2_CNY)
{
 int v=700;
 RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
 delay(750);
 StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
 delay(1000);
 Adelante(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,pin1_CNY,pin2_CNY);
}

void Izquierda(int pin1_Motor,int pin2_Motor,int pin3_Motor,int pin4_Motor,int pin1_CNY,int
pin2_CNY)
{
 int v=700;
 LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
 delay(760);
 StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
 delay(1000);
 Adelante(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,pin1_CNY,pin2_CNY);
}

void Adelante(int pin1_Motor,int pin2_Motor,int pin3_Motor,int pin4_Motor,int pin1_CNY,int
pin2_CNY)
{
```

```
  int v_e=500;
  int v=700;
   while(CNY70(pin1_CNY)==0 && CNY70(pin2_CNY)==0)
   {
    ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
   }
    StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
    delay(1000);
    if(CNY70(pin1_CNY)==1 && CNY70(pin2_CNY)==0)
    {
     RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v_e);
     while(CNY70(pin1_CNY)!=0){}
        StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);

           Serial1.println("Esta en el primer caso");

     delay(1000);
    }
    else
    {
     if(CNY70(pin1_CNY)==0 && CNY70(pin2_CNY)==1)
     {
      LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v_e);
      while(CNY70(pin2_CNY)!=0){}
         StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);

       Serial1.println("Esta en el segundo caso");
       delay(1000);
     }
     else
     {
      Serial1.println("Ambos negros");
  delay(1000);

     }

    }
    ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
    delay(1300);
    StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
    delay(2000);

}
void Atras(int pin1_Motor,int pin2_Motor,int pin3_Motor,int pin4_Motor,int pin3_CNY,int
pin_CNY)
{

  int v_e=500;
   int v=500;
   while(CNY70(pin3_CNY)==0 && CNY70(pin4_CNY)==0)
   {
     BackwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
   }
    StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
    delay(1000);
    if(CNY70(pin3_CNY)==1 && CNY70(pin4_CNY)==0)
    {
```

```
                RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v_e);
                while(CNY70(pin3_CNY)!=0){}
                StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
                Serial1.println("Esta en el primer caso");
                delay(1000);
               }
               else
               {
                if(CNY70(pin3_CNY)==0 && CNY70(pin4_CNY)==1)
                {
                  LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v_e);
                  while(CNY70(pin4_CNY)!=0){}
                     StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);

                   Serial1.println("Esta en el segundo caso");
                   delay(1000);
                }
                else
                {
                  Serial1.println("Ambos negros");
                  delay(1000);
                }
               }
               BackwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
               delay(1800);
               StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
               delay(1000);
            }

            float calibrar_blanco(int pin1_CNY,int pin2_CNY,int pin3_CNY,int pin4_CNY)
            {
            return (CNY70_V(pin1_CNY)+CNY70_V(pin2_CNY)+CNY70_V(pin3_CNY)+CNY70_V(pin4_CNY))/4;
            }

            float calibrar_negro(int pin1_CNY,int pin2_CNY,int pin3_CNY,int pin4_CNY)
            {
            return (CNY70_V(pin1_CNY)+CNY70_V(pin2_CNY)+CNY70_V(pin3_CNY)+CNY70_V(pin4_CNY))/4;
            }

            int es_fin(int pin1_CNY,int pin2_CNY,int pin3_CNY,int pin4_CNY)
            {
             if(CNY70(pin1_CNY)==1 && CNY70(pin2_CNY)==1 && CNY70(pin3_CNY)==1 &&
CNY70(pin4_CNY)==4)
             {
               return 1;
             }
             else
             {
               return 0;
             }
            }

            void setup()
            {
             Serial1.begin(9600);
             Serial.begin(9600);

             pinMode(pin1_Motor,OUTPUT);
```

```
 pinMode(pin2_Motor,OUTPUT);
 pinMode(pin3_Motor,OUTPUT);
 pinMode(pin4_Motor,OUTPUT);

 pinMode(pin1_CNY,INPUT);
 pinMode(pin2_CNY,INPUT);
 pinMode(pin3_CNY,INPUT);
 pinMode(pin4_CNY,INPUT);

 pinMode(pin1_SHARP,INPUT);
 pinMode(pin2_SHARP,INPUT);

 pinMode(pin_ULTRA,INPUT);

 pos_actual.fila=0;
 pos_actual.columna=0;
 visitados[pos]=pos_actual;

calibrar=true;

}

void loop()
{
 if(calibrar)
 {
   float b=calibrar_blanco(pin1_CNY,pin2_CNY,pin3_CNY,pin4_CNY);
   Serial1.print("Blanco: ");
   Serial1.println(b);
   Serial1.println();
    delay(5000);

   /**
   Serial1.println("Poner en negro");
   aux=  while(Serial.read()!='r'){}
   */

   float n=calibrar_negro(pin1_CNY,pin2_CNY,pin3_CNY,pin4_CNY);
   Serial1.print("Negro: ");
   Serial1.println(n);
   Serial1.println();
   c=(b+n)/2;
   Serial1.println(c);
   delay(5000);
   calibrar=false;
 }
 if(!es_fin(pin1_CNY,pin2_CNY,pin3_CNY,pin4_CNY))
 {
    //Comprobamos las direcciones
  if(sharp(pin1_SHARP)>5)
  {
   posibles.push(R);
  }

   if(sharp(pin2_SHARP)>5)  //Por ahora este es el que apunta a la izquierda, el que esta
puesto ahora
   {
```

```
 posibles.push(L);
}

if(ultrasonic_distance(pin_ULTRA)>5)
{
 posibles.push(F);//ultrasonic es el de delante
}

//Daremos prioridad al de delante

delay(1000);//¿?¿?¿

//Aqui comprabamos si una casilla esta repetida
bool valido=0;
while(!posibles.isEmpty() && !valido)
{

  actual=posibles.pop();
  //Verificar el movimiento para el enderezado
  switch (actual)
 {
  case F:
       if(!casilla_repetida(pos_actual.fila+1,pos_actual.columna,visitados,pos))
       {
        recorrido.push(pos_actual);
        ++pos;
        valido=1;
        pos_actual.fila=pos_actual.fila+1;
        visitados[pos]=pos_actual;
        Camino.push(F);
        ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
       }
       break;

  case R:
       if(!casilla_repetida(pos_actual.fila,pos_actual.columna-1,visitados,pos))
       {
        recorrido.push(pos_actual);  ++pos;
        valido=1;
        pos_actual.columna=pos_actual.columna-1;
        visitados[pos]=pos_actual;
        Camino.push(L);//Metemos el opuesto
        RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
        ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
       }
       break;

  case L:
       if(!casilla_repetida(pos_actual.fila,pos_actual.columna+1,visitados,pos))
       {
        recorrido.push(pos_actual);  ++pos;
                  valido=1;
        pos_actual.columna=pos_actual.columna+1;
        visitados[pos]=pos_actual;
        Camino.push(R);
        LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
        ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
```

```
          }
          break;
   }
  }


  if(posibles.isEmpty() && !valido)//No hay direcciones posibles
  {
   BackwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
   Posicion pos_aux=recorrido.pop();
   pos_actual.fila=pos_aux.fila;
   pos_actual.columna=pos_aux.columna;
   //--pos;
   actual=Camino.pop();
   if(actual!=F)
   {
    if(actual==L)
    {
      LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
    }
    else
    {
      RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
    }
   }
  }

  while(!posibles.isEmpty())//Vaciamos la lista de posibles
  {
   actual=posibles.pop();
  }

}
else
{
  RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
  RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
  ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);

  while(!Camino.isEmpty())
  {
   actual=Camino.pop();

    switch (actual)//
   {
    case F:
          ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
          Camino.push(F);
          break;

    case R:
          RightROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
          ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
          Camino.push(R);
          break;

    case L:
```

```
                LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
                ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
                Camino.push(L);
                break;
            }
           }
          }
         }
```

6.2. Plan de pruebas

Hemos desarrollado pequeños codigos para probar los distintos sensores y actuadores y así poder calibrarlos

```
//para testear el giro y el movimiento hacia delante
v=700;
LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
delay(710);
LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
delay(710);
LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
delay(710);
LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
delay(710);
LeftROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
delay(800);
StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
delay(4000);
ForwardROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor,v);
delay(1000);
StopROBOT(pin1_Motor,pin2_Motor,pin3_Motor,pin4_Motor);
//Codigo para probar los sensores y el modulo Bluetooth
//Estos métodos envían información al modulo de Bluetooth los cuales podemos ver a través de un Serial
int aux;
aux=CNY70(pin1_CNY);
aux=CNY70(pin2_CNY);
aux=CNY70(pin3_CNY);aux=CNY70(pin4_CNY);
float d;
d=sharp(pin1_SHARP)
d=sharp(pin2_SHARP)
d=ultrasonic_distance(pin_ULTRA)
```

7. IMPLEMENTACIÓN

7.1. Librerías

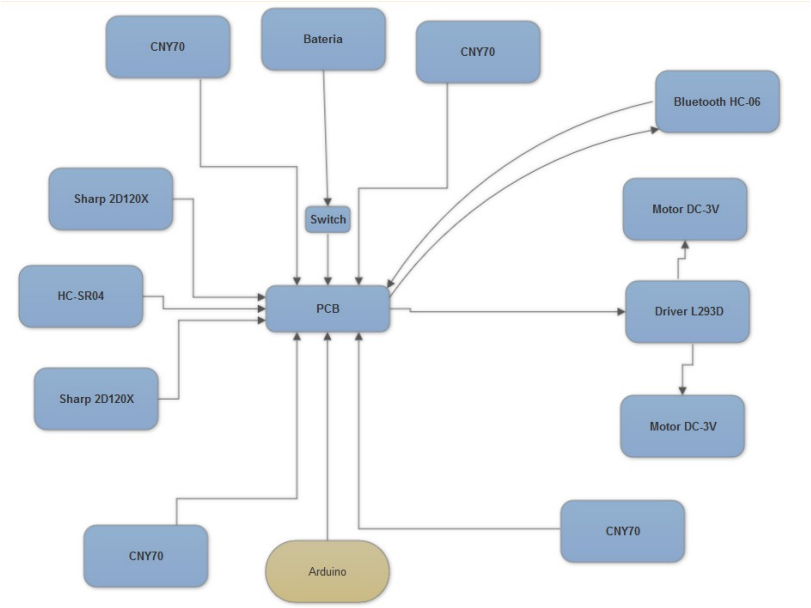
Librería StackList para la implementación de la estructura de datos Pila

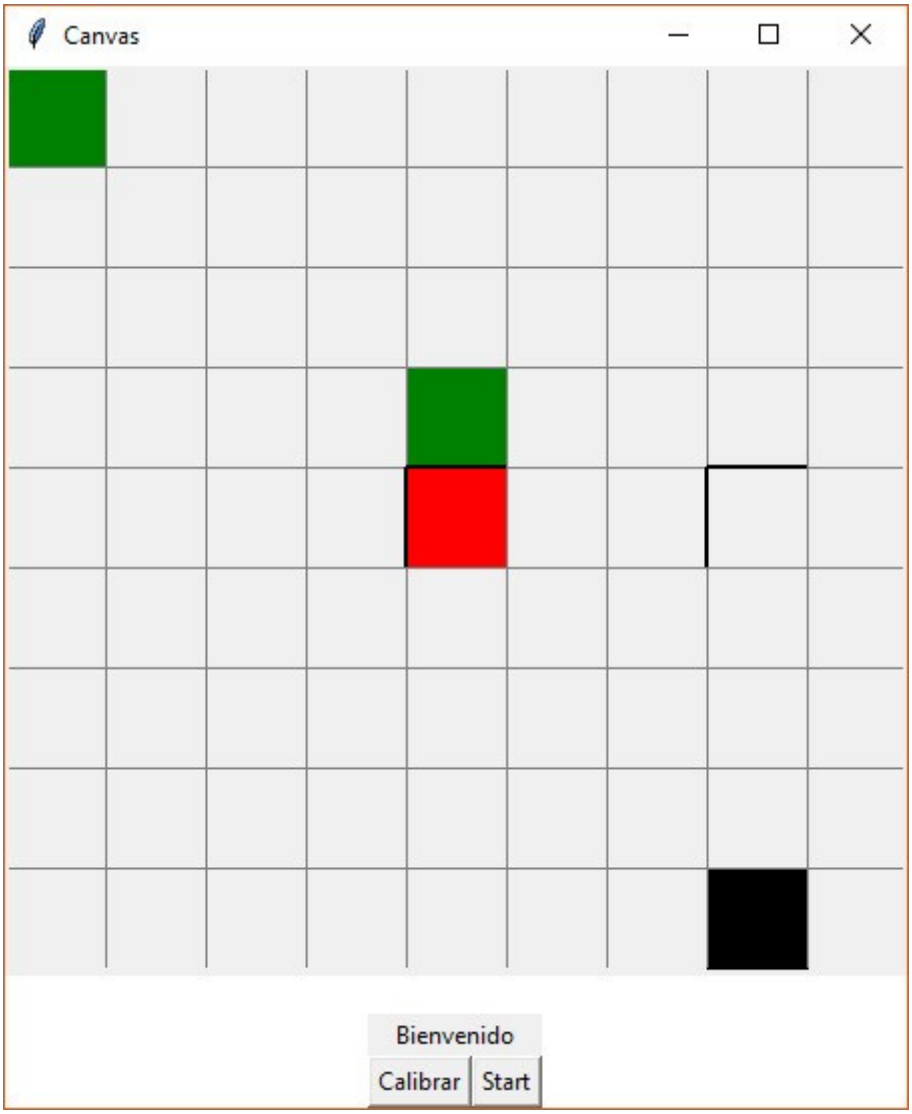
7.2. Apuntes sobre el código

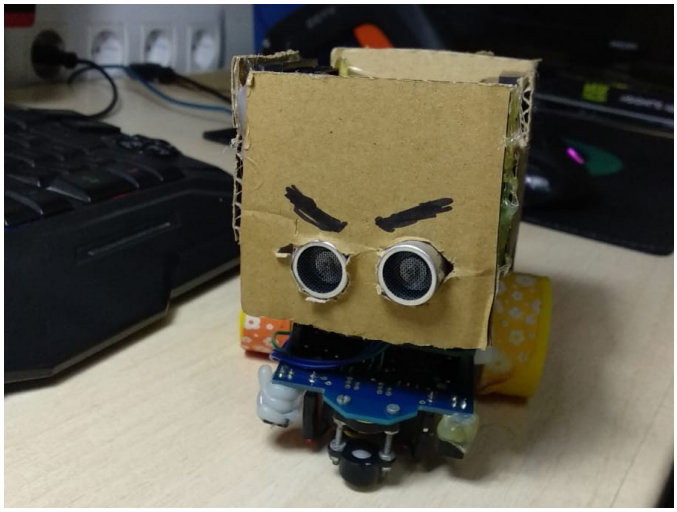
Para poder girar y movernos x distancia hacemos uso de los delay, indicando cuanto tiempo queremos que se mueva el robot a cualquier direccíon

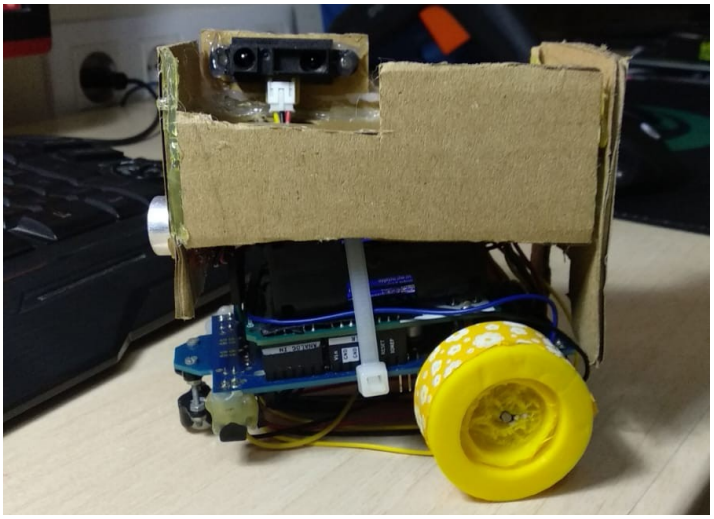
Para enderezar el robot usamos todos los sensores CNY, los de delante cuando se mueve hacia delante y los de atrás cuando se mueve hacia atras

8. MONTAJE




Canvas
Bienvenido
Calibrar
Start





9. PRUEBAS

ID	CATEGORIA	DESCRIPCIÓN	Comprobacíon
RF01	Movimiento	El robot debe ser capaz de moverse	Activamos los pines adecuados para que los motores se muevan hacia delantes
RF02	Movimiento	El robot debe ser capaz de pivotar 90º a derecha	Movemos un motor hacia delante y otro hacia atrás durante x tiempo, entonces detenemos los motores
RF03	Movimiento	El robot debe ser capaz de pivotar 90º a izquierda	Movemos un motor hacia delante y otro hacia atrás durante x tiempo, después detenemos los motores
RF04	Movimiento	El robot debe ser capaz de avanzar en línea recta	Nos movemos hacia delante, si en los CNY de delante se detecta uno en negro y otro en blanco, giramos el robot en la dirección adecuada hasta que ambos estén en negro
RF05	Movimiento	El robot avanza adecuadamente hasta la siguiente celda	Una vez enderezado el robot, nos movemos hacia delante x tiempo , después detenemos los motores
RF06	Detección	El robot debe ser capaz de detectar una pared frontal	Usando el sensor HC-SR04, una vez nos detengamos si la distancia obtenida es menor de 15 cm, hay pared
RF07	Detección	El robot debe ser capaz de detectar una pared lateral derecha	Usando el sensor HC-SR04, una vez nos detengamos si la distancia obtenida es menor de 15 cm, hay pared
RF08	Detección	El robot debe ser capaz de detectar una pared lateral izquierda	Usando el sensor Sharp 2D120X, una vez nos detengamos si la distancia obtenida es menor de 20 cm, hay pared
RF09	Detección	El robot debe ser capaz de detectar la transición entre celdas	Usando el sensor Sharp 2D120X, una vez nos detengamos si la distancia obtenida es menor de 20 cm, hay pared
RF10	Detección	El robot debe ser capaz de detectar la celda de salida	Una vez nos detengamos, si todos los sensores CNY70 detectan negro, estamos en la salida
RF11	Resolución	El robot almacena información sobre las celdas del laberinto	Almacenamos sus coordenadas para identificarla,se almacenan en un vector y partiendo de que la casilla de salida es la 0,0 y puede haber coordenadas negativas
RF12	Resolución	El robot es capaz de decidir el siguiente movimiento en base a la información sobre la celda	EL robot solo elegirá una dirección en la que no haya una pared y que lleve a una casilla que aun no haya visitado
RF13	Resolución	El robot es capaz de recorrer varias celdas del laberinto siguiendo el algoritmo empleado	Siguiendo las anteriores resoluciones, el robot es capaz de moverse de forma correcta varias casillas
RF14	Resolución	El robot es capaz de salir del laberinto	Aun necesita algo de calibrado en el movimiento, pero es capaz de llegar a la salida usando las

ID	CATEGORIA	DESCRIPCIÓN	Comprobacíon
			resoluciones anteriores
RF15	Información	El robot envía al PC información sobre el número de celdas recorridas	Solo lo hace a la hora de volver a la casilla inicial desde la salida
RF16	Información	El robot envía al PC información sobre obstáculos en cada celda	La información de las paredes se la indicamos al PC mediante Bluetooth
RF17	Información	El robot envía al PC información sobre la velocidad de movimiento	No lo hace, no es necesario
RF18	Información	El robot envía al PC información sobre la distancia que lleva recorrida	No lo hace, no es necesario
RF19	Información	El robot envía al PC información sobre el tiempo transcurrido desde la entrada al laberinto	No lo hace, no es necesario
RF20	Información	El robot envía al PC información sobre la trayectoria ejecutada	Si, todas las casilla por la que pasa el robot son marcadas en la interfaz
RF22	Información	El PC muestra una representación gráfica del laberinto	Si, pero solo de la información que logra recoger del robot
RF23	Usuario	Interfaz gráfica en PC que recopile información	Si, realizada en Python 3 con Tkinter
RF24	Pruebas	Incluir modo test al arranque del robot	Realizamos un calibrado de los CNY70 al menos, aunque quizas se añada uno para los sensores Sharp

10. MEJORAS

Ninguna