



Universidad Nacional de La Matanza

Sistemas Operativos Avanzados

Año 2019

Trabajo Practico

“LaberintoSmart”

Integrantes Equipo de Proyecto

D.N.I.	Nombre	E-Mail
39.336.665	Focaraccio, Ezequiel	eafocaraccio@gmail.com
39.923.347	Galluzzo, Luciano	luchogalluzzo421@gmail.com
40.378.661	Agustín, Riva	aagusriva@gmail.com

Profesores:

- Graciela De Luca (Jefa de Catedra)
- Waldo A. Valiente
- Sebastián Barillaro
- Mariano Volker
- Carnuccio, Esteban Andrés
- Gerardo García



Índice

Introducción	3
Descripción del proyecto	4
Auto-Robot creado con Arduino	4
Aplicación Android	8
Inicio	8
Resolviendo.....	10
Registros	12
Información.....	13
Componente HW utilizado.....	14
Microcontroladora	14
Arduino UNO	14
Sensores	14
Sensor Óptico Reflectivo Infrarrojo Cny70	14
Sensor Ultrasonido Hc-sr04 Distancia	14
Modulo Bluetooth Hc06 Uart Ttl Esclavo	15
Actuadores	15
Motor Dc 3v A 6v Caja Reductora	15
Doble Puente H Driver L298n	15
Modulo Bluetooth Hc06 Uart Ttl Esclavo	15
Diagramas	16
Diagrama de Estados	16
Diagrama de Conexiones	17
Diagrama de Software	18
Diagrama Físico.....	18
Diagrama Funcional.....	19
Diagrama Lógico.....	19
Anexo – Código Arduino	20



Introducción

LaberintoSmart es un proyecto que nace de la idea de los alumnos con la curiosidad de ver cómo funcionan los seguidores de líneas. A partir de eso, y luego de varias discusiones, se llegó al común acuerdo de agregarle una distinción para que pueda resolver laberintos.

El proyecto está dividido en dos partes: una aplicación Android y un robot formado con un microcontrolador Arduino.

Por un lado, el robot es un auto a escala utilizando ciertos sensores específicos para lograr el objetivo. Dicho dispositivo será capaz de, como bien dijimos anteriormente, resolver un laberinto construido con líneas negras sobre una base blanca.

Además, el coche será capaz de evitar choques frente a objetos que aparezcan en el lado posterior del mismo gracias a un sensor ultrasónico capaz de medir distancias específicas.

Por otro lado, tendremos la posibilidad de realizar diferentes funcionalidades desde una aplicación en sistema Android. Dicha aplicación nos permitirá desde encender el robot hasta tener un seguimiento detallado del historial de laberintos realizados.



Descripción del proyecto

Como bien hemos mencionado en la introducción, LaberintoSmart es un proyecto que consiste en 2 partes bien definidas: un robot formado con un microcontrolador Arduino y una aplicación Android.

A continuación, entraremos en más detalle sobre cada uno de estos.

Auto-Robot creado con Arduino

La primera parte del proyecto consiste en un auto a escala creado en base a un microcontrolador Arduino UNO capaz de moverse de forma completamente autónoma guiándose a través de líneas negras sobre un fondo negro, a lo que comúnmente se llama “Seguidor de líneas”.

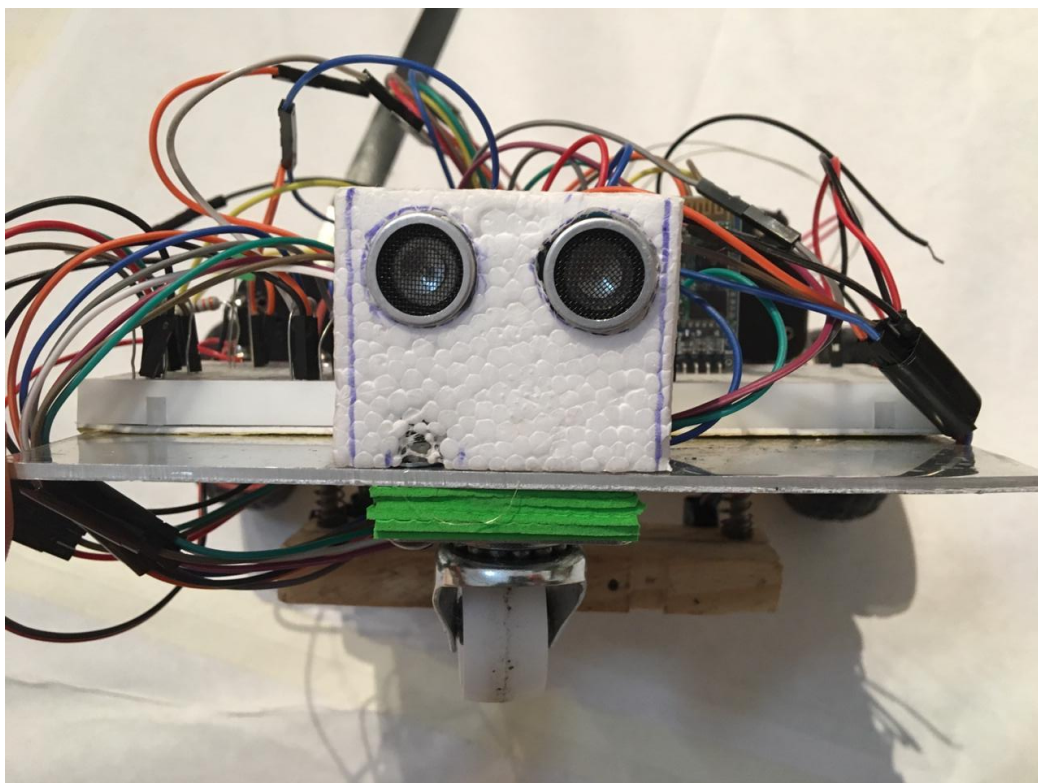
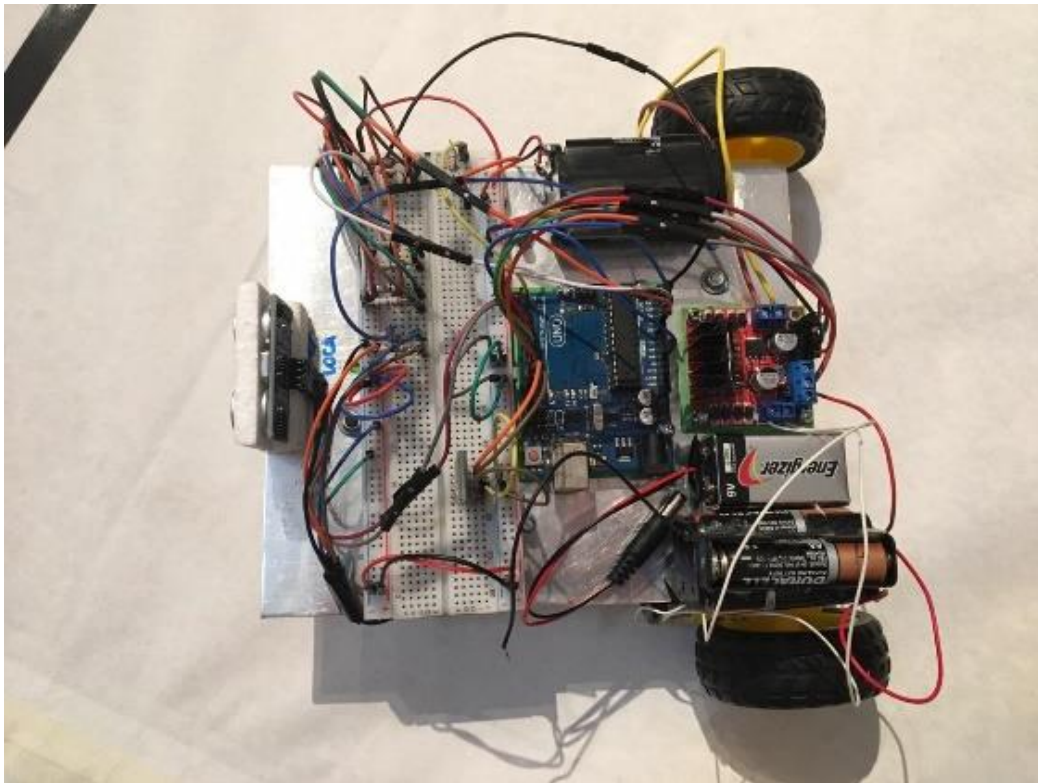
Este seguidor de líneas será capaz de también resolver cualquier laberinto que se le presente a través de la regla de la mano izquierda (dicha regla especifica que, para resolver un laberinto, siempre se debe dejar el brazo izquierdo pegado a la pared, dándole más prioridad doblar a la izquierda que cualquier otra cosa).

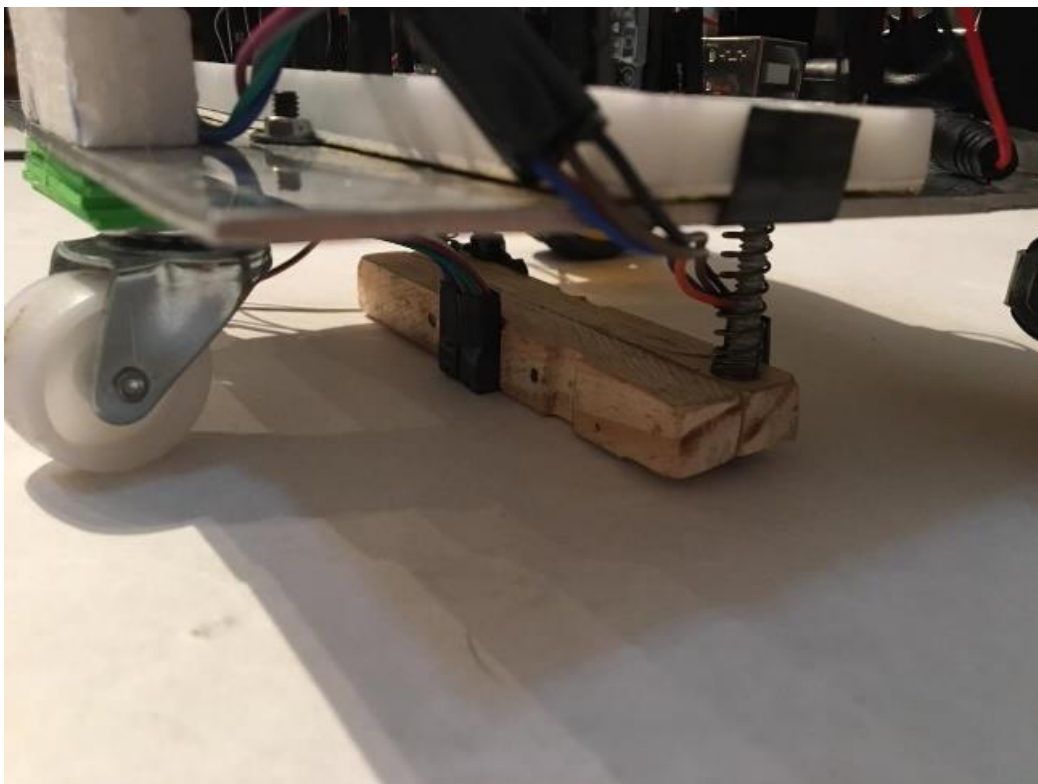
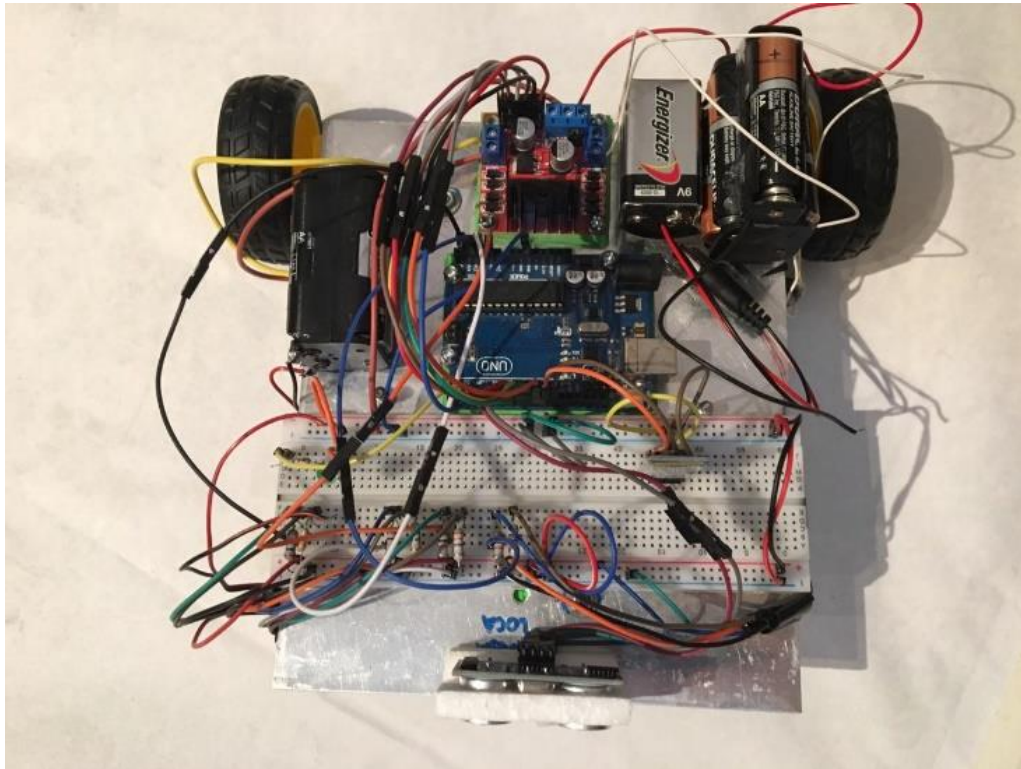
Además, el auto ofrece la posibilidad de, una vez que haya resuelto el laberinto, optimizarlo para encontrar el camino más corto hasta la salida, sin la necesidad de tener que realizar “caminos muertos” o “giros en U”.

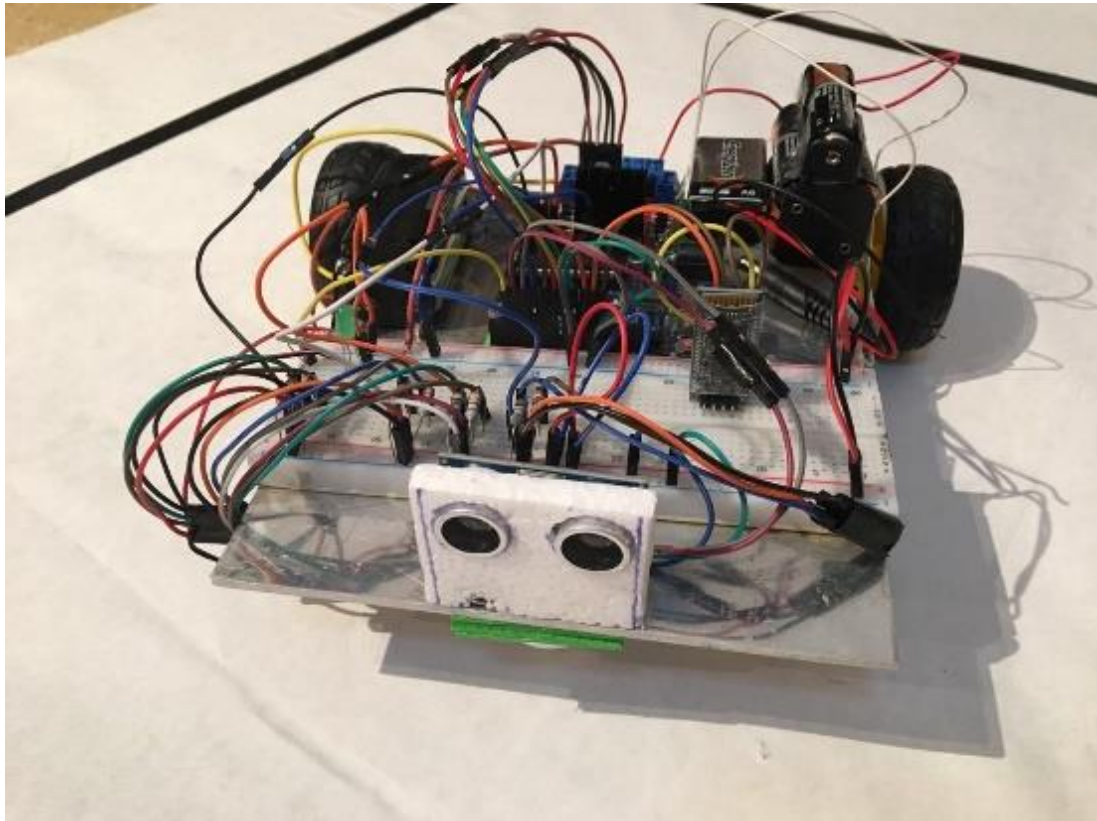
Como si esto fuera poco, el robot permite de un modo “Manual” en el que en cada intersección que encuentre en el camino, este se quede esperando por la indicación del ser humano para dónde dirigirse a través de la aplicación móvil.

Al finalizar el modo manual, otorgamos la posibilidad de que el robot demuestre que aprendió el camino exactamente como le fue indicando por el ser humano. Para esto, al igual que en la optimización, se deberá colocar al robot en el mismo lugar de partida y este realizará el mismo camino que le fue indicado anteriormente.

Por último, el auto posee la capacidad de detenerse frente a objetos que surjan en el camino del laberinto para no chocar con estos, dando aviso mediante la aplicación.









Aplicación Android

Esta segunda parte del proyecto consiste en una aplicación móvil que nos sirve de ayuda para ejecutar y poner en práctica todas las funcionalidades del robot. Además, ofrece la posibilidad de llevar un registro de todas esas ejecuciones dentro de un historial para luego observar los tiempos y el estado de cada una.

Inicio





Comprobar Conexión

Dentro de esta sección se podrá crear una conexión con el Bluetooth del robot realizando una búsqueda en caso de que todavía no se encuentre emparejado nuestro celular con el dispositivo.

Para ello, debemos presionar en el botón “Comprobar Conexión” y esperar a que se nos abra una nueva pantalla en donde se observaran otros 3 botones:

1. *Activar/Desactivar*: Aquí podremos encender y apagar la conexión bluetooth de nuestro celular.
2. *Ver dispositivos emparejados*: Aquí podremos observar aquellos dispositivos que ya se encuentran emparejados a nuestro celular.
3. *Buscar dispositivos*: Aquí podremos buscar nuestro HC-06 con el nombre de “LSmart_BT” e ingresar la clave “4321”.



Iniciar

También, en la pantalla de inicio, se podrá dar inicio al robot para la resolución del laberinto. En el momento en que se presione el botón “Iniciar”, aparecerá un pop-up preguntando si deseamos ejecutar el modo Automático o el modo Manual.

Luego de elegir una de las opciones, automáticamente se conectará a nuestro celular con el bluetooth del robot y nos llevará a la sección de “Resolviendo”.



Resolviendo

En esta pantalla nos mostrará el estado del robot en el que se encuentra en la resolución del laberinto y nos otorgará la posibilidad de finalizarlo en cualquier momento. Además, mostrará un cronometro con el tiempo que va tardando el robot en resolver el laberinto.

En esta actividad, también tenemos la posibilidad de encender un led si es que el sensor de Luz del celular detecta oscuridad y de frenar momentáneamente al robot si es que el sensor de Distancia del celular detecta que esta siendo tapado.

Modo Manual

Si el modo elegido fue el modo Manual, cuando el robot detecte una intersección se frenará esperando a que nosotros le indiquemos que decisión tomar. Esto lo haremos con el sensor Acelerómetro del celular, girando este para el lugar que queremos que doble. Es decir, si queremos que el robot doble a la derecha, debemos inclinar nuestro celular 90° a la derecha y viceversa. En cambio, si queremos que siga adelante, debemos inclinar el celular para adelante de forma que la pantalla quede boca arriba.

Para que esto funcione correctamente, el celular debe encontrarse en la posición por defecto (perpendicular al piso) antes de ejecutar el modo manual y mantenerlo así durante toda la ejecución.

Una vez que el robot haya resuelto el laberinto, nos mostrará una pantalla con el tiempo que tardo y el estado con el que finalizo. Además, nos ofrecerá la posibilidad de pedirle al robot que vuelva a recorrer el laberinto ejecutando el mismo camino que nosotros le dijimos que haga.

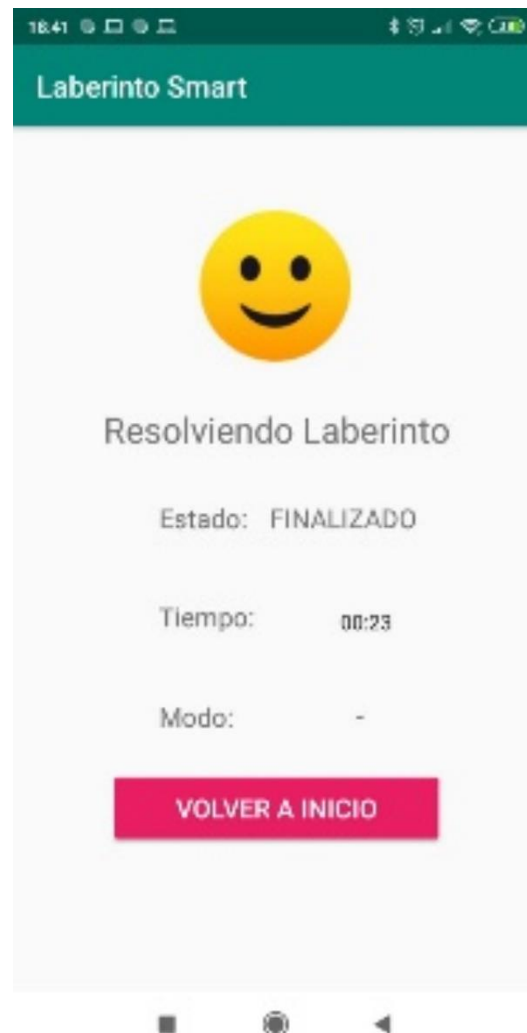




Modo Automático

Si el modo que elegimos fue el modo Automático, el robot resolverá el laberinto de forma autónoma hasta encontrar el fin.

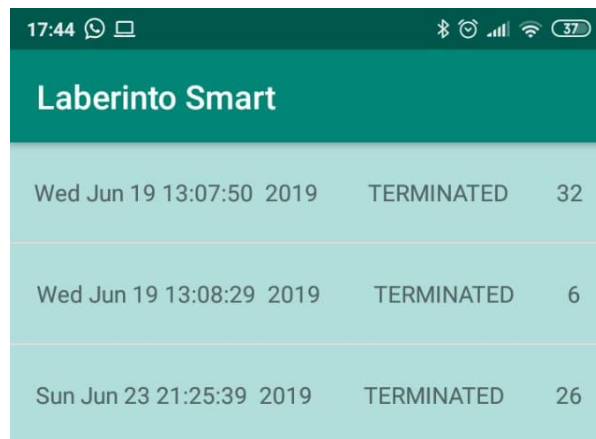
Una vez que esto ocurra, nos mostrará una pantalla con el tiempo que tardo y el estado con el que finalizo. Además, nos ofrecerá la posibilidad de pedirle al robot que vuelva a recorrer el laberinto de manera optimizada, es decir, con el camino más corto aprendido durante la ejecución anterior.





Registros

En esta sección se podrá visualizar todos los registros que hemos realizado en nuestro celular con el robot de LaberintoSmart. En él se podrá ver la fecha y la hora en la que fue ejecutado, el estado con el que finalizó y el tiempo en segundos que tardó.



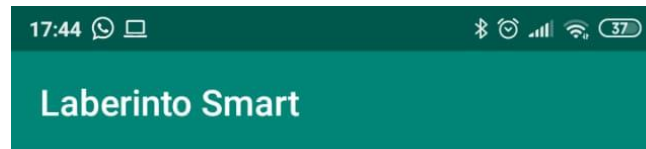
The screenshot shows the 'Laberinto Smart' app interface. At the top, there's a status bar with the time 17:44 and various icons. Below it, the app title 'Laberinto Smart' is displayed in a teal header. The main content area is a list of three records, each with a date and time, a status, and a duration in seconds.

Fecha y Hora	Estado	Tiempo (segundos)
Wed Jun 19 13:07:50 2019	TERMINATED	32
Wed Jun 19 13:08:29 2019	TERMINATED	6
Sun Jun 23 21:25:39 2019	TERMINATED	26



Información

En esta sección se podrá encontrar los integrantes del desarrollo de este proyecto.



LABERINTO SMART

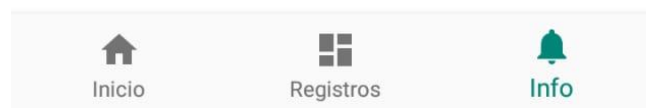
SISTEMAS OPERATIVOS AVANZADOS

Integrantes:

Focaraccio Ezequiel

Galluzo Luciano

Riva Agustín



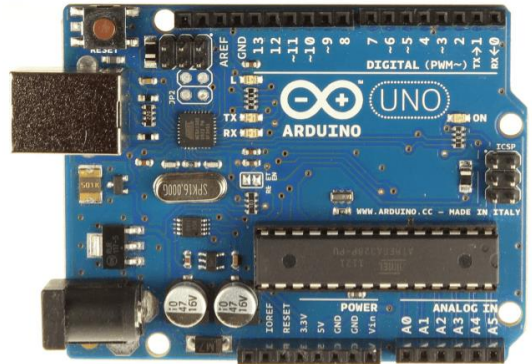


Componente HW utilizado

Microcontroladora

Arduino UNO

Es la placa mediante la cual se procesará todo el código junto con la lógica para la correcta resolución del laberinto, tanto en el modo Manual como en el modo Automático. Además, en ella se procesará el algoritmo de optimización para obtener el camino mas corto hasta la salida.



Sensores

Sensor Óptico Reflectivo Infrarrojo Cny70

Mediante este sensor obtendremos la reflexión de la superficie pudiendo detectar entre un color u otro (blanco/negro) y así lograr que nuestro dispositivo pueda seguir una línea negra en un fondo blanco o viceversa



Sensor Ultrasonido Hc-sr04 Distancia

Mediante este sensor obtendremos la distancia de los objetos que se encuentre delante de él, ya que permite ejecutar un ultrasonido y al recibir dicha señal de vuelta luego de haber rebotado en el objeto, calcular la distancia al mismo.





Modulo Bluetooth Hc06 Uart Ttl Esclavo

Este sensor nos permitirá capturar las señales emitidas con el celular para realizar los cálculos y el funcionamiento necesario.



Actuadores

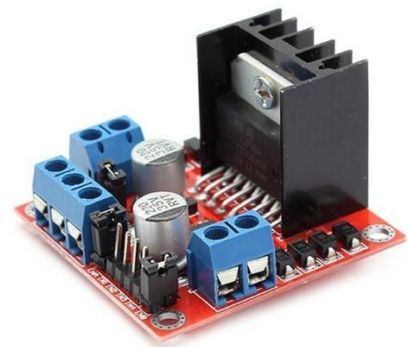
Motor Dc 3v A 6v Caja Reductora

Dicho motor, junto con la rueda, va a ser el que nuestro dispositivo pueda movilizarse por la superficie.



Doble Puente H Driver L298n

Este controlador nos permitirá ir variando en la potencia emitida a los motores junto con la dirección de esta.



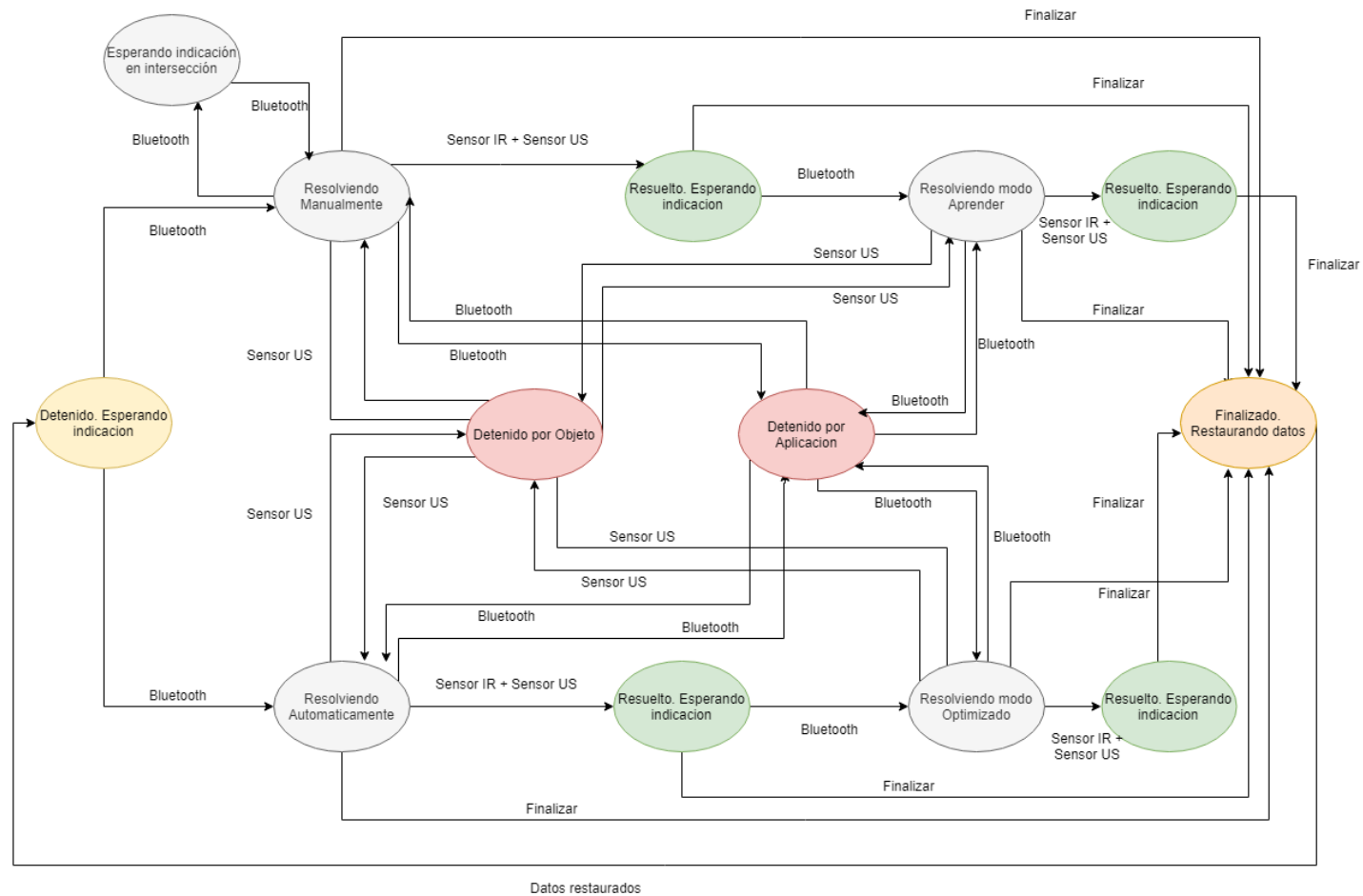
Modulo Bluetooth Hc06 Uart Ttl Esclavo

Viéndolo como un actuador, nos permitirá emitir las funciones necesarias al celular según lo que queramos que aparezca en la pantalla.



Diagramas

Diagrama de Estados



fritzing



Diagrama de Software

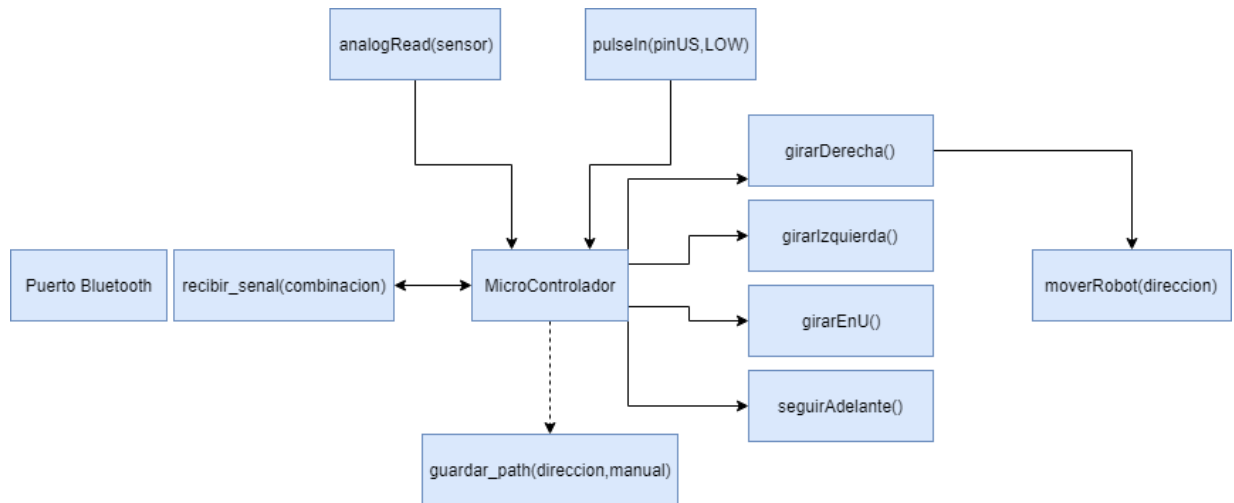


Diagrama Físico

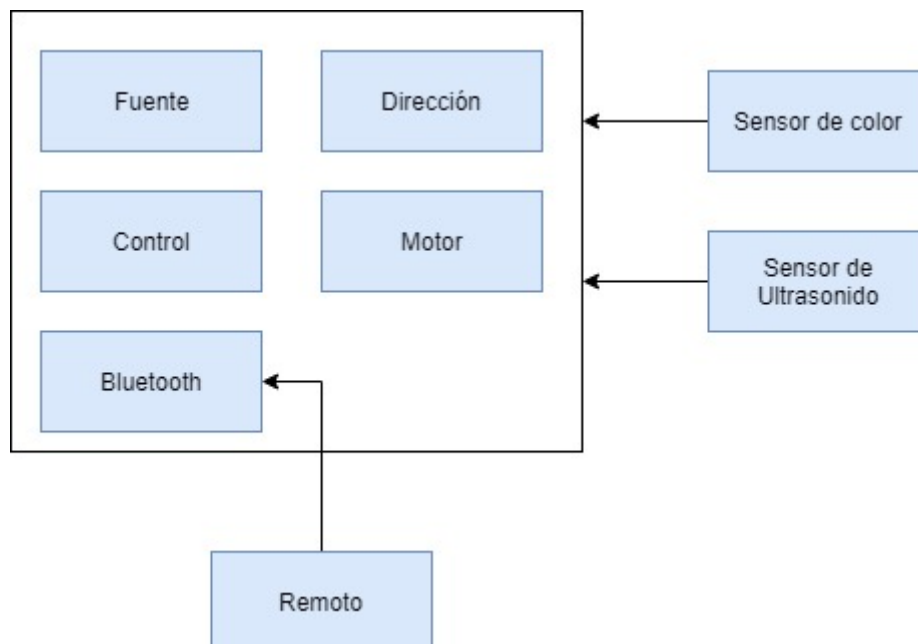




Diagrama Funcional

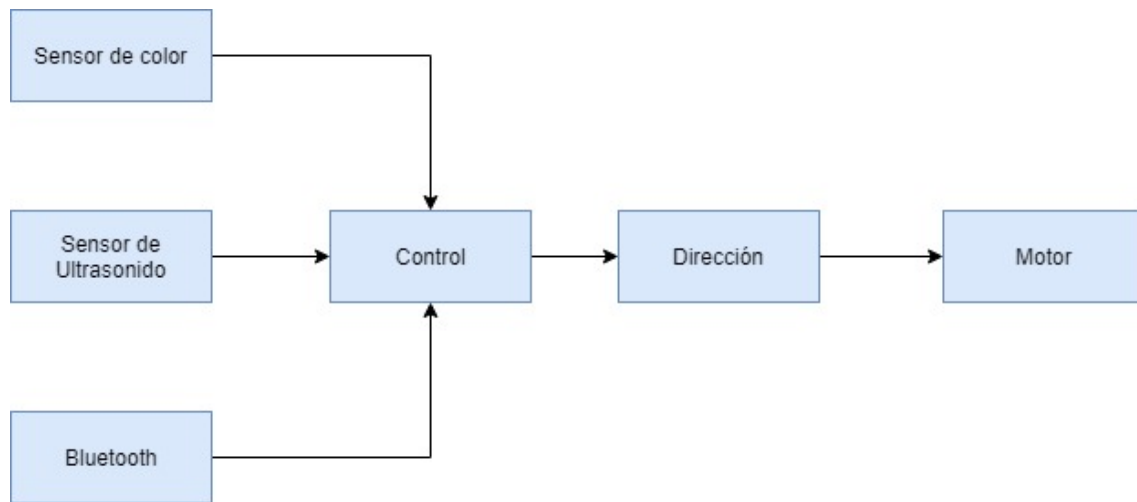
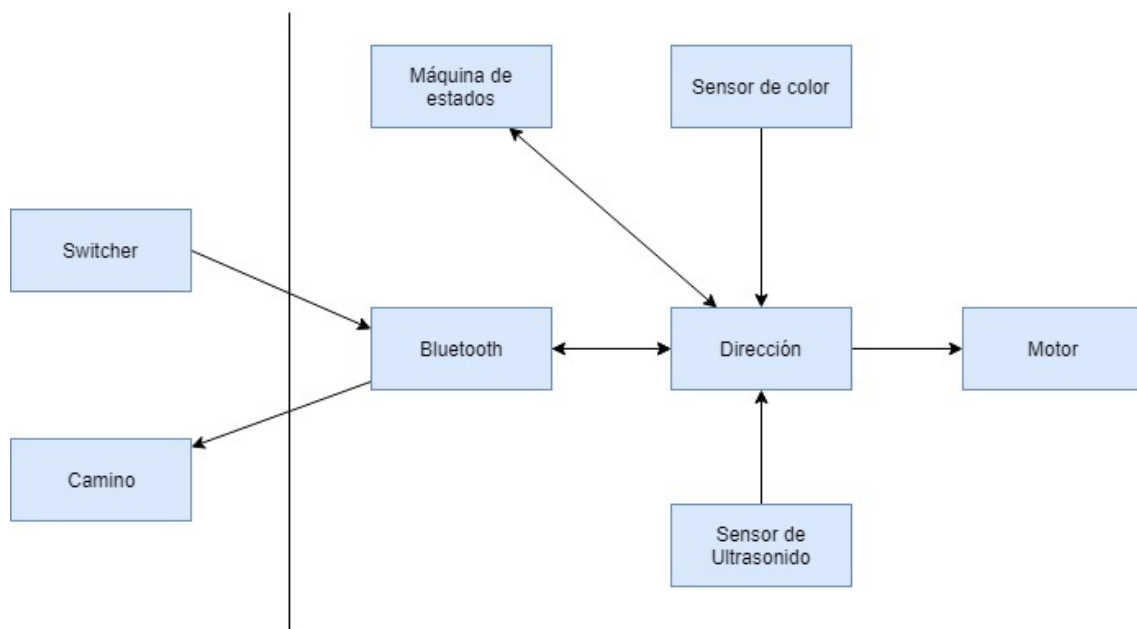


Diagrama Lógico





Anexo – Código Arduino

```
char laberinto_path[100] = "";
unsigned char laberinto_longitud = 0;
int laberinto_indice = 0;
char band_inicio = 'N';
char band_manual = 'N';
int z;
int estaParado = 0;
```

```
//Led
```

```
#define ledPin 2
```

```
//Movimientos
```

```
#define GO_AHEAD 1
```

```
#define TURN_RIGHT 2
```

```
#define TURN_LEFT 3
```

```
#define TURN_RIGHT_SOFT 4
```

```
#define TURN_LEFT_SOFT 5
```

```
#define ADAPT_RIGHT 6
```

```
#define ADAPT_LEFT 7
```

```
#define ADAPT_RIGHT_SOFT 8
```

```
#define ADAPT_LEFT_SOFT 9
```

```
#define TURN_U 10
```

```
#define TURN_U_SOFT 11
```

```
#define STOP 12
```

```
#define TURN_RIGHT_MANUAL 13
```

```
#define TURN_LEFT_MANUAL 14
```

```
//Infrarrojo
```

```
int valorCI = 0;
```

```
int valorCD = 0;
```

```
int valorEI = 0;
```

```
int valorED = 0;
```

```
#define sensorCI 1
```

```
#define sensorCD 2
```

```
#define sensorEI 3
```

```
#define sensorED 4
```

```
//Motores
```

```
//Motor izquierda
```

```
#define pinENA 10
```

```
#define pinIN1 9
```

```
#define pinIN2 8
```

```
//Motor derecha
```



```
#define pinIN3 7
#define pinIN4 6
#define pinENB 5

//Ultrasonido
long t; //tiempo que demora en llegar el eco
long d; //distancia en centimetros
int finalizoLaberinto = 0;
#define pinUltraSonidoTrig 3
#define pinUltraSonidoEcho 4

//Bluetooth
#include <SoftwareSerial.h>
#define txBT 12
#define rxBT 11
char mensajeBT;
SoftwareSerial BT1(txBT,rxBT);

void setup() {
  Serial.begin(9600); // Comenzamos comunicacion serial.

  //LED
  pinMode(ledPin , OUTPUT);

  //Motor
  pinMode (pinENA, OUTPUT);
  pinMode (pinENB, OUTPUT);
  pinMode (pinIN1, OUTPUT);
  pinMode (pinIN2, OUTPUT);
  pinMode (pinIN3, OUTPUT);
  pinMode (pinIN4, OUTPUT);

  //Ultrasonido
  pinMode(pinUltraSonidoTrig, OUTPUT); //pin como salida
  pinMode(pinUltraSonidoEcho, INPUT); //pin como entrada
  digitalWrite(pinUltraSonidoTrig, LOW); //Inicializamos el pin con 0

  //Bluetooth
  BT1.begin(9600);
}

void loop() {
  /*valorCI = analogRead(sensorCI);
  valorCD = analogRead(sensorCD);
  valorEI = analogRead(sensorEI);
```



```
valorED = analogRead(sensorED);
```

```
BT1.print(valorCI);
```

```
BT1.print('\t');
```

```
BT1.print(valorCD);
```

```
BT1.print('\t');
```

```
BT1.print(valorEI);
```

```
BT1.print('\t');
```

```
BT1.print(valorED);
```

```
BT1.print('\n');
```

```
if(valorEI <= 150){
```

```
    BT1.print('N');
```

```
}else{
```

```
    BT1.print('B');
```

```
}
```

```
BT1.print('\t');
```

```
if(valorCI <= 150){
```

```
    BT1.print('N');
```

```
}else{
```

```
    BT1.print('B');
```

```
}
```

```
BT1.print('\t');
```

```
if(valorCD <= 150){
```

```
    BT1.print('N');
```

```
}else{
```

```
    BT1.print('B');
```

```
}
```

```
BT1.print('\t');
```

```
if(valorED <= 150){
```

```
    BT1.print('N');
```

```
}else{
```

```
    BT1.print('B');
```

```
}
```

```
BT1.print('\n');
```

```
delay(150);*/
```

```
while(band_inicio != 'B' && band_inicio != 'T' && band_inicio != 'F' && band_inicio !=  
'M' && band_inicio != 'L' && band_inicio != 'O' && band_inicio != 'S'){
```

```
    if (BT1.available()){
```

```
        band_inicio = BT1.read();
```

```
    }
```

```
}
```

```
if(band_inicio != 'T' && band_inicio != 'F' && band_inicio != 'M' && band_inicio != 'L'  
&& band_inicio != 'O' && band_inicio != 'S'){
```

```
    encender_led();
```

```
    band_inicio = 'N';
```



```
}else if(band_inicio != 'B' && band_inicio != 'F' && band_inicio != 'M' && band_inicio !=  
'L' && band_inicio != 'O' && band_inicio != 'S'){  
    apagar_led();  
    band_inicio = 'N';  
}  
}else if(band_inicio != 'B' && band_inicio != 'T' && band_inicio != 'M' && band_inicio !=  
'L' && band_inicio != 'O' && band_inicio != 'S'){  
    band_inicio = 'N';  
}  
}else if(band_inicio != 'B' && band_inicio != 'T' && band_inicio != 'F' && band_inicio !=  
'L' && band_inicio != 'O' && band_inicio != 'S'){  
    //Entramos en el modo Manual  
    for(z=0;z<100;z++){  
        laberinto_path[z] = 'Z';  
    }  
    laberinto_longitud = 0;  
    laberinto_indice = 0;  
    while(band_inicio != 'F'){  
        if(band_inicio == 'B'){  
            encender_led();  
        }  
        if(band_inicio == 'T'){  
            apagar_led();  
        }  
        if(band_inicio == 'P'){  
            //Debo frenar por lectura de sensor  
            moverRobot(STOP);  
        }else{  
            digitalWrite(pinUltraSonidoTrig, HIGH);  
            delayMicroseconds(10); //Enviamos un pulso de 10us  
            digitalWrite(pinUltraSonidoTrig, LOW);  
  
            t = pulseIn(pinUltraSonidoEcho, HIGH); //obtenemos el ancho del pulso  
            d = t/59; //escalamos el tiempo a una distancia en cm  
  
            if(d <= 10){  
                if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) <= 150 &&  
analogRead(sensorEI) <= 150 && analogRead(sensorED) <= 150){  
                    //Llego al final del camino  
                    moverRobot(STOP);  
                    BT1.print('F');  
                    //BT1.print('\n');  
                    band_inicio = 'F';  
                }else{  
                    //Esta parado por un objeto  
                    moverRobot(STOP);  
                    //Aca debemos mandar un aviso por bluetooth de que esta parado por un  
objeto y no puede continuar  
                    if(estaParado == 0){
```




```
BT1.print('P');
//BT1.print('\n');
estaParado = 1;
}
}
}else{
if(estaParado == 1){
    estaParado = 0;
    BT1.print('J');
    //BT1.print('\n');
}
if(analogRead(sensorEI) <= 150 && analogRead(sensorED) <= 150){
    while(analogRead(sensorEI) <= 150);
    if(analogRead(sensorCI) <= 150 || analogRead(sensorCD) <= 150){
        //Encontramos la combinacion I-A-D
        moverRobot(STOP);
        band_manual = recibir_senal(4);
        if(band_manual == 'I'){
            girarIzquierda(1);
        }else if(band_manual == 'D'){
            girarDerecha(1);
        }else{
            seguirAdelante(1);
        }
        band_manual = 'N';
    }else{
        //Encontramos la combinacion I-D
        moverRobot(STOP);
        band_manual = recibir_senal(3);
        if(band_manual == 'I'){
            girarIzquierda(1);
        }else{
            girarDerecha(1);
        }
        band_manual = 'N';
    }
}else if(analogRead(sensorEI) <= 150 && analogRead(sensorED) > 150){
    while(analogRead(sensorEI) <= 150);
    if(analogRead(sensorCI) <= 150 || analogRead(sensorCD) <= 150){
        //Encontramos la combinacion I-A
        moverRobot(STOP);
        band_manual = recibir_senal(1);
        if(band_manual == 'I'){
            girarIzquierda(1);
        }else{
            seguirAdelante(1);
        }
    }
}
```



```
    band_manual = 'N';
  }else{
    //Solo puede ir a la izquierda y lo hace
    girarIzquierda(1);
  }
}else if(analogRead(sensorEI) > 150 && analogRead(sensorED) <= 150){
  while(analogRead(sensorED) <= 150);
  if(analogRead(sensorCI) <= 150 || analogRead(sensorCD) <= 150){
    //Encontramos la combinacion A-D
    moverRobot(STOP);
    band_manual = recibir_senal(2);
    if(band_manual == 'D'){
      girarDerecha(1);
    }else{
      seguirAdelante(1);
    }
    band_manual = 'N';
  }else{
    //Solo puede ir a la derecha y lo hace
    girarDerecha(1);
  }
}
}else if(analogRead(sensorEI) > 150 && analogRead(sensorED) > 150 &&
analogRead(sensorCI) > 150 && analogRead(sensorCD) > 150){
  //Todos en blanco, girar en U
  girarEnU(1);
}else{
  //Solo linea para adelante, seguir derecho
  if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) > 150){
    moverRobot(ADAPT_LEFT_SOFT);
    delay(2);
  }else if(analogRead(sensorCI) > 150 && analogRead(sensorCD) <= 150){
    moverRobot(ADAPT_RIGHT_SOFT);
    delay(2);
  }else{
    moverRobot(GO_AHEAD);
  }
}
}
}
}
}
//Leo por si recibo alguna señal de que debemos parar por lectura del sensor de
distancia
if (BT1.available()){
  band_inicio = BT1.read();
  if(band_inicio == 'F'){
    moverRobot(STOP);
  }
}
}
```



```
}  
//imprimirCaminoSimplificado();  
}else if(band_inicio != 'B' && band_inicio != 'T' && band_inicio != 'F' && band_inicio !=  
'M' && band_inicio != 'S' && band_inicio != 'L'){  
    //Entramos en el modo Optimizado  
    laberinto_indice = 0;  
    while(band_inicio != 'F'){  
        if(band_inicio == 'B'){  
            encender_led();  
        }  
        if(band_inicio == 'T'){  
            apagar_led();  
        }  
        if(band_inicio == 'P'){  
            //Debo frenar por lectura de sensor  
            moverRobot(STOP);  
        }else{  
            digitalWrite(pinUltraSonidoTrig, HIGH);  
            delayMicroseconds(10); //Enviamos un pulso de 10us  
            digitalWrite(pinUltraSonidoTrig, LOW);  
  
            t = pulseIn(pinUltraSonidoEcho, HIGH); //obtenemos el ancho del pulso  
            d = t/59; //escalamos el tiempo a una distancia en cm  
  
            if(d <= 10){  
                if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) <= 150 &&  
analogRead(sensorEI) <= 150 && analogRead(sensorED) <= 150){  
                    //Llego al final del camino  
                    moverRobot(STOP);  
                    BT1.print('F');  
                    //BT1.print("\n");  
                    band_inicio = 'F';  
                }else{  
                    //Esta parado por un objeto  
                    moverRobot(STOP);  
                    //Aca debemos mandar un aviso por bluetooth de que esta parado por un  
objeto y no puede continuar  
                    if(estaParado == 0){  
                        BT1.print('P');  
                        //BT1.print("\n");  
                        estaParado = 1;  
                    }  
                }  
            }else{  
                if(estaParado == 1){  
                    estaParado = 0;  
                    BT1.print('J');
```



```
//BT1.print('\n');
}
if(analogRead(sensorEI) <= 150 || analogRead(sensorED) <= 150){
  band_manual = sacar_path();
  if(band_manual == 'I'){
    girarIzquierda(2);
  }else if(band_manual == 'D'){
    girarDerecha(2);
  }else{
    seguirAdelante(2);
  }
  band_manual = 'N';
}else{
  //Solo linea para adelante, seguir derecho
  if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) > 150){
    moverRobot(ADAPT_LEFT_SOFT);
    delay(2);
  }else if(analogRead(sensorCI) > 150 && analogRead(sensorCD) <= 150){
    moverRobot(ADAPT_RIGHT_SOFT);
    delay(2);
  }else{
    moverRobot(GO_AHEAD);
  }
}
}
}
}
//Leo por si recibo alguna señal de que debemos parar por lectura del sensor de
distancia
if (BT1.available()){
  band_inicio = BT1.read();
  if(band_inicio == 'F'){
    moverRobot(STOP);
  }
}
}
}
}else if(band_inicio != 'B' && band_inicio != 'T' && band_inicio != 'F' && band_inicio !=
'M' && band_inicio != 'S' && band_inicio != 'O'){
  //Entramos en el modo Aprender (Learning)
  laberinto_indice = 0;
  while(band_inicio != 'F'){
    if(band_inicio == 'B'){
      encender_led();
    }
    if(band_inicio == 'T'){
      apagar_led();
    }
    if(band_inicio == 'P'){
```



```
//Debo frenar por lectura de sensor
moverRobot(STOP);
}else{
  digitalWrite(pinUltraSonidoTrig, HIGH);
  delayMicroseconds(10);//Enviamos un pulso de 10us
  digitalWrite(pinUltraSonidoTrig, LOW);

  t = pulseIn(pinUltraSonidoEcho, HIGH);//obtenemos el ancho del pulso
  d = t/59;//escalamos el tiempo a una distancia en cm

  if(d <= 10){
    if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) <= 150 &&
    analogRead(sensorEI) <= 150 && analogRead(sensorED) <= 150){
      //Llego al final del camino
      moverRobot(STOP);
      BT1.print('F');
      //BT1.print("\n");
      band_inicio = 'F';
    }else{
      //Esta parado por un objeto
      moverRobot(STOP);
      //Aca debemos mandar un aviso por bluetooth de que esta parado por un
      objeto y no puede continuar
      if(estaParado == 0){
        BT1.print('P');
        //BT1.print("\n");
        estaParado = 1;
      }
    }
  }else{
    if(estaParado == 1){
      estaParado = 0;
      BT1.print('J');
      //BT1.print("\n");
    }
    if(analogRead(sensorEI) <= 150 || analogRead(sensorED) <= 150){
      band_manual = sacar_path();
      if(band_manual == 'I'){
        girarIzquierda(2);
      }else if(band_manual == 'D'){
        girarDerecha(2);
      }else{
        seguirAdelante(2);
      }
      band_manual = 'N';
    }else if(analogRead(sensorCI) <= 150 || analogRead(sensorCD) <= 150){
      if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) > 150){
```




```
moverRobot(ADAPT_LEFT_SOFT);
delay(2);
}else if(analogRead(sensorCI) > 150 && analogRead(sensorCD) <= 150){
  moverRobot(ADAPT_RIGHT_SOFT);
  delay(2);
}else{
  moverRobot(GO_AHEAD);
}
}else{
  girarEnU(2);
}
}
}
//Leo por si recibo alguna señal de que debemos parar por lectura del sensor de
distancia
if (BT1.available()){
  band_inicio = BT1.read();
  if(band_inicio == 'F'){
    moverRobot(STOP);
  }
}
}
}else if(band_inicio != 'B' && band_inicio != 'T' && band_inicio != 'F' && band_inicio !=
'M' && band_inicio != 'L' && band_inicio != 'O'){
  for(z=0;z<100;z++){
    laberinto_path[z] = 'Z';
  }
  laberinto_longitud = 0;
  laberinto_indice = 0;
  //Entramos en el modo Automatico
  while(band_inicio != 'F'){
    if(band_inicio == 'B'){
      encender_led();
    }
    if(band_inicio == 'T'){
      apagar_led();
    }
    if(band_inicio == 'P'){
      //Debo frenar por lectura de sensor
      moverRobot(STOP);
    }else{
      digitalWrite(pinUltraSonidoTrig, HIGH);
      delayMicroseconds(10);//Enviamos un pulso de 10us
      digitalWrite(pinUltraSonidoTrig, LOW);

      t = pulseIn(pinUltraSonidoEcho, HIGH);//obtenemos el ancho del pulso
      d = t/59;//escalamos el tiempo a una distancia en cm
```



```
if(d <= 10){
    if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) <= 150 &&
analogRead(sensorEI) <= 150 && analogRead(sensorED) <= 150){
        //Llego al final del camino
        moverRobot(STOP);
        BT1.print('F');
        //BT1.print('\n');
        band_inicio = 'F';
    }else{
        //Esta parado por un objeto
        moverRobot(STOP);
        if(estaParado == 0){
            BT1.print('P');
            //BT1.print('\n');
            estaParado = 1;
        }
    }
}else{
    if(estaParado == 1){
        estaParado = 0;
        BT1.print('J');
        //BT1.print('\n');
    }
    if(analogRead(sensorEI) <= 150){
        girarIzquierda(0);
    }else if(analogRead(sensorED) <= 150){
        //Lo hago avanzar hasta que deje de detectar la linea de la derecha
        while(analogRead(sensorED) <= 150);
        //Entonces pregunto si los de adelante son negros (alguno de los dos) y debe
seguir adelante, sino girar a la derecha
        if(analogRead(sensorCI) <= 150 || analogRead(sensorCD) <= 150){
            //BT1.print('A');
            //BT1.print('\n');
            guardar_path('A',0);
        }else{
            girarDerecha(0);
        }
    }else if(analogRead(sensorCI) <= 150 || analogRead(sensorCD) <= 150){
        if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) > 150){
            moverRobot(ADAPT_LEFT_SOFT);
            delay(2);
        }else if(analogRead(sensorCI) > 150 && analogRead(sensorCD) <= 150){
            moverRobot(ADAPT_RIGHT_SOFT);
            delay(2);
        }else{
            moverRobot(GO_AHEAD);
        }
    }
}
```



```
}
}else{
    //Todos los sensores detectan blanco, debemos girar en U
    girarEnU(0);
}
}
}
//Leo por si recibo alguna señal de que debemos parar por lectura del sensor de
distancia
if (BT1.available()){
    band_inicio = BT1.read();
    if(band_inicio == 'F'){
        moverRobot(STOP);
    }
}
}
//imprimirCaminoSimplificado();
}
}

void encender_led(){
    digitalWrite(ledPin , HIGH);
}

void apagar_led(){
    digitalWrite(ledPin , LOW);
}

char recibir_senal(int combinacion){
    char band_rx = 'N';
    //BT1.print('R');
    //BT1.print('\n');
    while(band_rx != 'I' && band_rx != 'A' && band_rx != 'D'){
        if (BT1.available()){
            band_rx = BT1.read();
            if(combinacion == 1){
                if(band_rx == 'D'){
                    //Solo puede recibir I o A
                    band_rx = 'N';
                }
            }else if(combinacion == 2){
                if(band_rx == 'I'){
                    //Solo puede recibir A o D
                    band_rx = 'N';
                }
            }else if(combinacion == 3){
                if(band_rx == 'A'){
```



```
//Solo puede recibir I o D
band_rx = 'N';
}
}
//Si la combinacion es la 4, no preguntamos nada porque cualquier cosa que
recibamos (I, A o D) sirve
if(band_rx == 'B'){
    encender_led();
    band_rx = 'N';
}
if(band_rx == 'T'){
    apagar_led();
    band_rx = 'N';
}
}
}
return band_rx;
}

void girarEnU(int manual){
    moverRobot(STOP);
    if(manual != 2){
        guardar_path('U', manual);
    }
    //BT1.print('U');
    //BT1.print('\n');
    delay(100);
    moverRobot(TURN_RIGHT);
    delay(500);
    while(analogRead(sensorCI) > 150 && analogRead(sensorCD) > 150){
        moverRobot(TURN_RIGHT_SOFT);
    }
    moverRobot(STOP);
    delay(300);
}

void girarIzquierda(int manual){
    int velocidad;
    if(manual != 1){
        velocidad = TURN_LEFT;
    }else{
        velocidad = TURN_LEFT_MANUAL;
    }
    delay(100); //100
    moverRobot(STOP);
    if(manual != 2){
        guardar_path('I', manual);
    }
}
```



```
//BT1.print('I');  
//BT1.print('\n');  
delay(100);  
moverRobot(velocidad);  
delay(500);  
while(analogRead(sensorCI) > 150 && analogRead(sensorCD) > 150){  
    moverRobot(TURN_LEFT_SOFT);  
}  
moverRobot(STOP);  
delay(300);  
}
```

```
void girarDerecha(int manual){  
    int velocidad;  
    if(manual != 1){  
        velocidad = TURN_RIGHT;  
    }else{  
        velocidad = TURN_RIGHT_MANUAL;  
    }  
    delay(100);//100  
    moverRobot(STOP);  
    if(manual != 2){  
        guardar_path('D', manual);  
    }  
    //BT1.print('D');  
    //BT1.print('\n');  
    delay(100);  
    moverRobot(velocidad);  
    delay(400);  
    while(analogRead(sensorCI) > 150 && analogRead(sensorCD) > 150){  
        moverRobot(TURN_RIGHT_SOFT);  
    }  
    moverRobot(STOP);  
    delay(300);  
}
```

```
void seguirAdelante(int manual){  
    moverRobot(STOP);  
    if(manual != 2){  
        guardar_path('A', manual);  
    }  
    //BT1.print('A');  
    //BT1.print('\n');  
    int tiempo_inicial = millis();  
    int tiempo_transcurrido = millis() - tiempo_inicial;  
    while(tiempo_transcurrido < 500){  
        if(analogRead(sensorCI) <= 150 && analogRead(sensorCD) > 150){
```



```
moverRobot(ADAPT_LEFT_SOFT);
delay(2);
}else if(analogRead(sensorCI) > 150 && analogRead(sensorCD) <= 150){
    moverRobot(ADAPT_RIGHT_SOFT);
    delay(2);
}else{
    moverRobot(GO_AHEAD);
}
tiempo_transcurrido = millis() - tiempo_inicial;
}
}

char sacar_path(){
    char direccion = laberinto_path[laberinto_indice];
    laberinto_indice ++;
    return direccion;
}

void guardar_path(char direccion, int manual){
    laberinto_path[laberinto_longitud] = direccion;
    laberinto_longitud ++;
    if(manual == 0){
        resolver_labirinto();
    }
}

void resolver_labirinto(){
    if(laberinto_longitud < 3 || laberinto_path[laberinto_longitud-2] != 'U'){
        return; //sólo simplificar el camino si la segunda a la última vuelta fue una 'U'
    }
    int totalAngle = 0;
    int i;
    for(i=1;i<=3;i++)
    {
        switch(laberinto_path[laberinto_longitud-i]){
            case 'D':
                totalAngle += 90;
                break;
            case 'I':
                totalAngle += 270;
                break;
            case 'U':
                totalAngle += 180;
                break;
        }
    }
}
```



```
totalAngle = totalAngle % 360; //Obtener el ángulo como un número entre 0 y 360
grados.
```

```
switch(totalAngle){ //reemplazar todos los giros con una sola.
```

```
case 0:
```

```
    laberinto_path[laberinto_longitud - 3] = 'A';
```

```
    break;
```

```
case 90:
```

```
    laberinto_path[laberinto_longitud - 3] = 'D';
```

```
    break;
```

```
case 180:
```

```
    laberinto_path[laberinto_longitud - 3] = 'U';
```

```
    break;
```

```
case 270:
```

```
    laberinto_path[laberinto_longitud - 3] = 'I';
```

```
    break;
```

```
}
```

```
laberinto_longitud -= 2; //La ruta es ahora dos pasos más cortos.
```

```
}
```

```
void imprimirCaminoSimplificado(){
```

```
    int i;
```

```
    //BT1.print("\n");
```

```
    for(i=0;i<laberinto_longitud;i++){
```

```
        BT1.print(laberinto_path[i]);
```

```
        BT1.print('\t');
```

```
    }
```

```
    //BT1.print("\n");
```

```
}
```

```
String GetLine()
```

```
{ String S = "" ;
```

```
    if (Serial.available())
```

```
    { char c = Serial.read(); ;
```

```
        while ( c != '\n') //Hasta que el caracter sea intro
```

```
        { S = S + c ;
```

```
            delay(25) ;
```

```
            c = Serial.read();
```

```
        }
```

```
        return( S + '\n' ) ;
```

```
    }
```

```
}
```

```
void moverRobot(int direccion){
```

```
    switch(direccion){
```

```
        case GO_AHEAD :
```

```
            analogWrite (pinENA, 170);
```

```
            analogWrite (pinENB, 170);
```




```
//Direccion motor A
digitalWrite (pinIN1, HIGH);
digitalWrite (pinIN2, LOW);
//Direccion motor B
digitalWrite (pinIN3, HIGH);
digitalWrite (pinIN4, LOW);
break;
case ADAPT_RIGHT :
    analogWrite (pinENA, 170);
    analogWrite (pinENB, 0);
    //Direccion motor A
    digitalWrite (pinIN1, HIGH);
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
    digitalWrite (pinIN3, HIGH); //HIGH
    digitalWrite (pinIN4, LOW);
    break;
case ADAPT_LEFT :
    analogWrite (pinENA, 0);
    analogWrite (pinENB, 170);
    //Direccion motor A
    digitalWrite (pinIN1, HIGH); //HIGH
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
    digitalWrite (pinIN3, HIGH);
    digitalWrite (pinIN4, LOW);
    break;
case ADAPT_RIGHT_SOFT :
    analogWrite (pinENA, 170);
    analogWrite (pinENB, 0);
    //Direccion motor A
    digitalWrite (pinIN1, HIGH);
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
    digitalWrite (pinIN3, HIGH); //HIGH
    digitalWrite (pinIN4, LOW);
    break;
case ADAPT_LEFT_SOFT :
    analogWrite (pinENA, 0);
    analogWrite (pinENB, 170);
    //Direccion motor A
    digitalWrite (pinIN1, HIGH); //HIGH
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
    digitalWrite (pinIN3, HIGH);
    digitalWrite (pinIN4, LOW);
    break;
```



```
case TURN_RIGHT :
    analogWrite (pinENA, 105);//85
    analogWrite (pinENB, 200);
    //Direccion motor A
    digitalWrite (pinIN1, HIGH);
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
    digitalWrite (pinIN3, LOW);
    digitalWrite (pinIN4, HIGH);
    break;
case TURN_LEFT :
    analogWrite (pinENA, 170);
    analogWrite (pinENB, 100);//85
    //Direccion motor A
    digitalWrite (pinIN1, LOW);
    digitalWrite (pinIN2, HIGH);
    //Direccion motor B
    digitalWrite (pinIN3, HIGH);
    digitalWrite (pinIN4, LOW);
    break;
case TURN_RIGHT_SOFT :
    analogWrite (pinENA, 95);//70
    analogWrite (pinENB, 95);//70
    //Direccion motor A
    digitalWrite (pinIN1, HIGH);
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
    digitalWrite (pinIN3, LOW);
    digitalWrite (pinIN4, HIGH);
    break;
case TURN_LEFT_SOFT :
    analogWrite (pinENA, 95);//70
    analogWrite (pinENB, 95);//70
    //Direccion motor A
    digitalWrite (pinIN1, LOW);
    digitalWrite (pinIN2, HIGH);
    //Direccion motor B
    digitalWrite (pinIN3, HIGH);
    digitalWrite (pinIN4, LOW);
    break;
case TURN_RIGHT_MANUAL :
    analogWrite (pinENA, 150);//70
    analogWrite (pinENB, 150);//70
    //Direccion motor A
    digitalWrite (pinIN1, HIGH);
    digitalWrite (pinIN2, LOW);
    //Direccion motor B
```



```
digitalWrite (pinIN3, LOW);  
digitalWrite (pinIN4, HIGH);  
break;  
case TURN_LEFT_MANUAL :  
    analogWrite (pinENA, 150);//70  
    analogWrite (pinENB, 150);//70  
    //Direccion motor A  
    digitalWrite (pinIN1, LOW);  
    digitalWrite (pinIN2, HIGH);  
    //Direccion motor B  
    digitalWrite (pinIN3, HIGH);  
    digitalWrite (pinIN4, LOW);  
    break;  
case TURN_U :  
    analogWrite (pinENA, 170);  
    analogWrite (pinENB, 100);//85  
    //Direccion motor A  
    digitalWrite (pinIN1, LOW);  
    digitalWrite (pinIN2, HIGH);  
    //Direccion motor B  
    digitalWrite (pinIN3, HIGH);  
    digitalWrite (pinIN4, LOW);  
    break;  
case TURN_U_SOFT :  
    analogWrite (pinENA, 110);//70  
    analogWrite (pinENB, 110);//70  
    //Direccion motor A  
    digitalWrite (pinIN1, LOW);  
    digitalWrite (pinIN2, HIGH);  
    //Direccion motor B  
    digitalWrite (pinIN3, HIGH);  
    digitalWrite (pinIN4, LOW);  
    break;  
case STOP :  
    analogWrite (pinENA, 0);  
    analogWrite (pinENB, 0);  
    //Direccion motor A  
    digitalWrite (pinIN1, LOW);  
    digitalWrite (pinIN2, LOW);  
    //Direccion motor B  
    digitalWrite (pinIN3, LOW);  
    digitalWrite (pinIN4, LOW);  
    break;  
default :  
    analogWrite (pinENA, 0);  
    analogWrite (pinENB, 0);  
    //Direccion motor A
```



```
digitalWrite (pinIN1, LOW);  
digitalWrite (pinIN2, LOW);  
//Direccion motor B  
digitalWrite (pinIN3, LOW);  
digitalWrite (pinIN4, LOW);  
break;  
}  
}
```