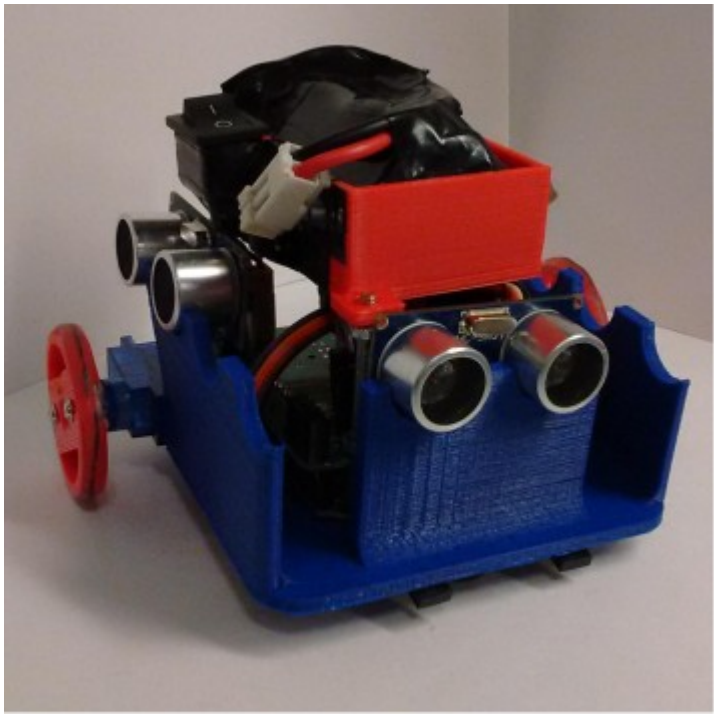
Competencia de robótica

JOHNNY 5

~2015~

Categoría: **Laberinto**

Nombre robot: **Zim**



Institución: **Club de Robótica FIUBA**

Participantes:

- **Gisela Farace** gisela.farace@gmail.com
- **Javier Choclin** jchoclin@gmail.com

Introducción

Este robot pertenece al proyecto Dreamster, desarrollado por el Club de Robótica FIUBA y esta destinado a ser una plataforma abierta de robótica. Además de una plataforma, Dreamster busca ser una comunidad. Al ser una plataforma abierta, todo lo que se desarrolle en el proyecto será públicado y compartido dentro de la comunidad para que la gente pueda replicarlo y hasta incluso mejorarlo.

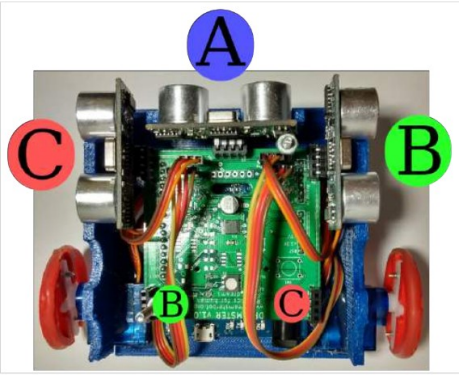
Dreamsterbot es un robot pensado para ser extremadamente flexible en su diseño, gracias a que sus partes estructurales están diseñadas e impresas en 3D, lo que permite modificarlas y recrearlas de forma muy sencilla, por cualquiera. Actualmente su hardware se compone de un *shield* personalizado, compatible para Arduino Uno y Arduino Leonardo, que permite controlar sensores de ultrasonido e IR, así como dos motores de DC.

Otro de los objetivos de Dreamster es difundir la robótica en la sociedad enfatizando especialmente en la incorporación de la robótica en las escuelas. Este proyecto fue invitado por la Universidad de Santiago de Chile en el marco del evento Robotics Day donde brindamos un workshop de tres días. Asistieron personas de todas las edades y la mayoria sin conocimientos de programación, y a partir del workshop lograron aprender a programar los robots y competir entre ellos. También brindamos una pasantía a alumnos del secundario ILSE, donde aprendieron a programar los Dreamsterbots y lograron hacerlos seguidores de líneas, laberintos y robots que empujan cajas fuera de un recinto.

Mecánica

Este robot cuenta con dos motores servos (modelo: Servo Tower Pro Sg90 9g) que fueron modificados en su interior para poder transformarse en motores comunes de continua.

Su alimentación esta provista por una bateria recargable de Li-ión de 7.2 V y 2.2 Ah.



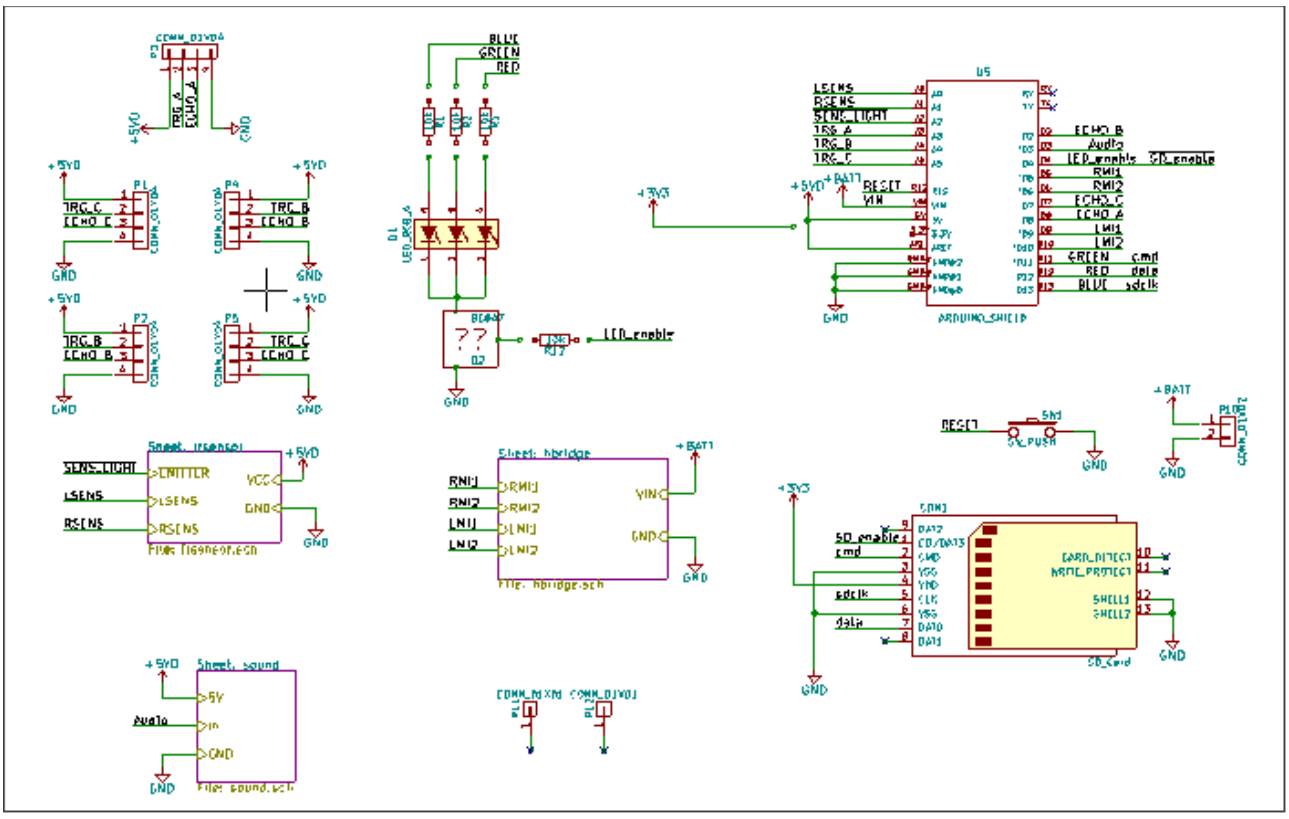
Los sensores de ultrasonido (HC-SR04) se encuentran ubicados como se muestra en la imagen. En el caso de los sensores B y C, hay dos posibles ubicaciones en la placa y se puede cambiar la posición del sensor. Estas cuatro ubicaciones en particular están conectadas en forma de cruz. Por esta razón no es posible ubicar dos sensores en forma cruzada, por ejemplo ambos en las ubicaciones B. La máxima cantidad de sensores que se pueden usar al mismo tiempo es de tres.

El robot principalmente se basa en piezas impresas en 3D. Su estructura posee pequeñas canaletas para poder colocar los sensores de ultrasonido sin que se muevan. Las ruedas fueron diseñadas para poder adaptarlas a los motores y además se le agrego silicona en los bordes para que la rueda no resvale. También se diseño un porta baterias para colocar sobre el robot.

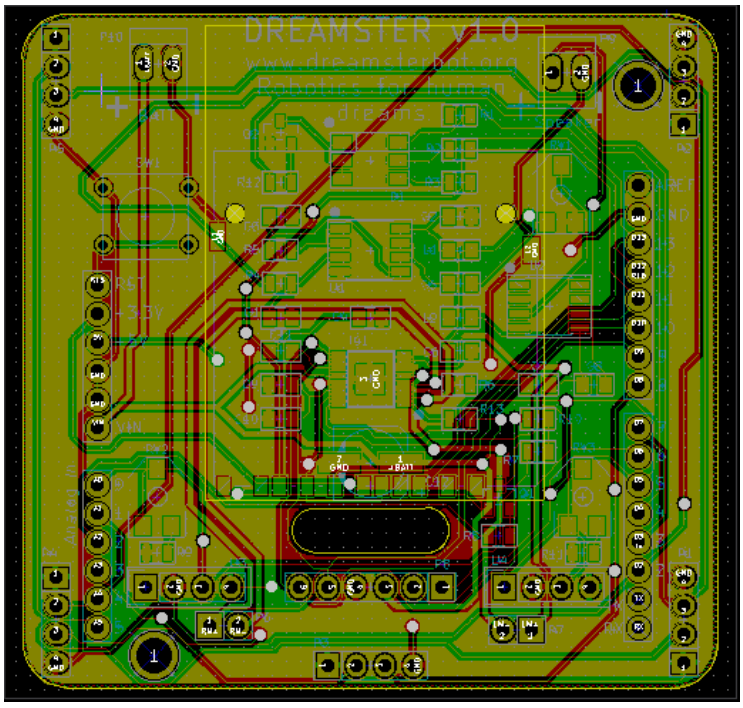
El robot mide aproximadamente 13 cm de ancho, 9.5 cm de largo y 8.5 cm de alto.

Electrónica

El robot posee un Arduino Leonardo y por encima una Shield conectada. Esta Shield conecta los motores, los sensores de ultrasonido, los sensores infrarojos y la bateria al Arduino. Además, tiene un led RGB, resistencias, capacitores, presets para regular los sensores infrarojos, y un puente H (DVR8833) para los motores. Todos los componentes son smd.



Esquemático del Shield en Kicad



Diseño del PCB del Shield en Kicad

Programación

La programación del robot se encuentra en el Arduino Leonardo. El lenguaje de programación es C++. Se utilizo la IDE de Arduino para escribir el código y programar el robot a través de un cable USB a mini USB.

La lógica del robot consiste en medir las distancias con los sensores de ultrasonido, dependiendo de las mediciones responde de diferentes maneras. Primero, el robot sigue la pared derecha del laberinto para poder salir. Si en las mediciones encuentra una pared adelante y una al costado derecho, el robot dobla hacia la izquierda. En caso de no encontrar pared del lado derecho entonces dobla a la derecha. Si encuentra pared derecha y no pared delantera, el robot debe seguir avanzando y mantieniendo su distancia con respecto a la pared para poder avanzar sin chocarse.

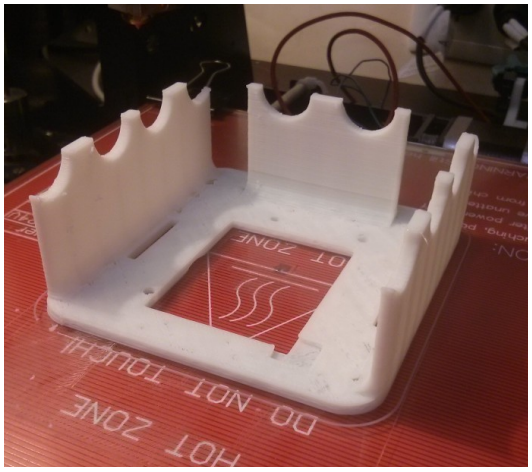
Conclusiones

El costo del robot es de aproximadamente 30 dólares.

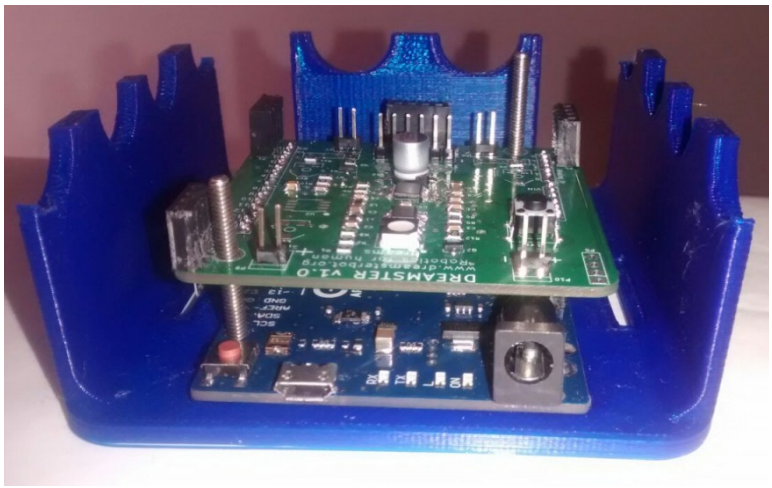
Todavía el robot se encuentra en su versión 1.0v. La próxima etapa incluirá una plataforma propia, Dreamduino Board, compatible con Arduino así como también con Raspberry Pi. Así mismo también estará disponible un stack de ROS (*Robot Operating System*) para correr sobre el hardware de la Raspberry Pi, con el objetivo de que Dreamsterbot pueda ser utilizado para hacer robótica de avanzada.

Anexo

Mecánica:



Diseño 3d de la estructura y el porta bateria



Ensamblado de la estructura con el Arduino y el Shield

Programación:

```
#include <NewPing.h> // Biblioteca que sirve para medir sensores de ultrasonido

//------------------------ DEFINICION DE PINES ---------------------------------//
// Sensores ultrasonido
#define Trigger_A A3
#define Echo_A 8
#define Trigger_I A4
#define Echo_I 2
#define Trigger_D A5
#define Echo_D 7
#define Max_Distancia 300

long Distancia_A = 0;
long Distancia_I = 0;
long Distancia_D = 0;

NewPing sensor_A(Trigger_A,Echo_A,Max_Distancia);
NewPing sensor_I(Trigger_I,Echo_I,Max_Distancia);
NewPing sensor_D(Trigger_D,Echo_D,Max_Distancia);
// Leds
#define led_verde 11
#define led_rojo 12
#define led_azul 13

// Sensores infrarrojos
#define sensor_izq A0
#define sensor_der A1

int sensor_ir_izq;
int sensor_ir_der;

// Motores
#define motor_izq_p 9
#define motor_izq_n 10
#define motor_der_p 5
#define motor_der_n 6

// Medidas de la celda y otros
# define ancho_celda 26
# define distancia_pared_min 9
# define distancia_pared 13
# define margen 1

enum Estados_Robot{DECIDIR, AVANZAR, GIRAR_DERECHA, GIRAR_IZQUIERDA, DETENER};
Estados_Robot estado_robot;

# define vel_avanzar 40
# define vel_dif 4
# define tiempo_avanzando 2300
# define tiempo_girando_der 500
# define tiempo_girando_izq 500

int t_0;
int t_1;
```

```
//------------------------------ FUNCIONES ------------------------------------//
// Funcion que mide los tres sensores de ultra sonido
void medir_sensores_us(){
  int echoTime_A = sensor_A.ping_median();
  Distancia_A = sensor_A.convert_cm(echoTime_A);
  int echoTime_I = sensor_I.ping_median(3);
  Distancia_I = sensor_I.convert_cm(echoTime_I);
  int echoTime_D = sensor_D.ping_median(3);
  Distancia_D = sensor_D.convert_cm(echoTime_D);
}

// Funcion que mide los dos sensores infrarojos
void medir_sensores_ir(){
  sensor_ir_der = analogRead(sensor_der);
  sensor_ir_izq = analogRead(sensor_izq);
}

// Funcion que detiene los motores
void detener_motores(){
  digitalWrite(motor_der_n, LOW);
  digitalWrite(motor_izq_n, LOW);
  digitalWrite(motor_der_p, LOW);
  digitalWrite(motor_izq_p, LOW);
}

void velocidad(int velocidad_der, int velocidad_izq){
  if (velocidad_der > 0 && velocidad_izq > 0){
    digitalWrite(motor_der_n, LOW);
    digitalWrite(motor_izq_n, LOW);
    analogWrite(motor_der_p, abs(velocidad_der));
    analogWrite(motor_izq_p, abs(velocidad_der));
  }
  else if (velocidad_der < 0 && velocidad_izq < 0){
    digitalWrite(motor_der_p, LOW);
    digitalWrite(motor_izq_p, LOW);
    analogWrite(motor_der_n, abs(velocidad_der));
    analogWrite(motor_izq_n, abs(velocidad_izq));
  }
  else if (velocidad_der > 0 && velocidad_izq < 0){
    digitalWrite(motor_der_n, LOW);
    digitalWrite(motor_izq_p, LOW);
    analogWrite(motor_der_p, abs(velocidad_der));
    analogWrite(motor_izq_n, abs(velocidad_izq));
  }
  else {
    digitalWrite(motor_der_p, LOW);
    digitalWrite(motor_izq_n, LOW);
    analogWrite(motor_der_n, abs(velocidad_der));
    analogWrite(motor_izq_p, abs(velocidad_izq));
  }
}


void tomar_decision(){
  detener_motores();
  delay(500);
  medir_sensores_us();
```

```
  t_0 = millis();
  if (Distancia_D > distancia_pared)
    estado_robot = GIRAR_DERECHA;
  else{
    if (Distancia_A > distancia_pared)
      estado_robot = AVANZAR;
    else
      estado_robot = GIRAR_IZQUIERDA;
  }
}

void girar_derecha(){
  t_1 = millis();
  if (t_1 < t_0 + tiempo_girando_der)
    velocidad(-vel_avanzar, vel_avanzar);
  else{
    estado_robot = AVANZAR;
    t_0 = millis();
  }
}

void girar_izquierda(){
  t_1 = millis();
  if (t_1 < t_0 + tiempo_girando_izq)
    velocidad(vel_avanzar, -vel_avanzar);
  else{
    estado_robot = DECIDIR;
  }
}


void avanzar_celda(){
  medir_sensores_us();
  t_1 = millis();
  if (t_1 < t_0 + tiempo_avanzando){
    // Si tenemos pared a ambos lados
    if (Distancia_D < distancia_pared && Distancia_I < distancia_pared){
      if (Distancia_D < Distancia_I){
        // Izquierda
        velocidad(vel_avanzar+vel_dif, vel_avanzar-vel_dif);
      }
      else{
        // Derecha
        velocidad(vel_avanzar-vel_dif, vel_avanzar+vel_dif);
      }
    }
    // Si solo hay pared a la izquierda
    else if (Distancia_D > distancia_pared && Distancia_I < distancia_pared){
      if (Distancia_I < distancia_pared_min)
        velocidad(vel_avanzar-vel_dif, vel_avanzar+vel_dif);
      else
        velocidad(vel_avanzar+vel_dif, vel_avanzar-vel_dif);
    }
    // Si hay pared a la derecha
    else if (Distancia_D < distancia_pared && Distancia_I > distancia_pared){
      if (Distancia_D < distancia_pared_min)
        velocidad(vel_avanzar+vel_dif, vel_avanzar-vel_dif);
```

```
      else
        velocidad(vel_avanzar-vel_dif, vel_avanzar+vel_dif);
    }
    // Si no hay pared a ambos lados
    else if (Distancia_D > distancia_pared && Distancia_I > distancia_pared){
      velocidad(vel_avanzar, vel_avanzar);
    }
  }
  else {
      estado_robot = DECIDIR;
      detener_motores();
  }
}


void serial(){
  Serial.print("A: ");
  Serial.print(Distancia_A  );
  Serial.print("// I: ");
  Serial.print(Distancia_I);
  Serial.print(" // D: ");
  Serial.println(Distancia_D);
}

//----------------------------- INICIALIZACION ----------------------------------//
void setup(){
  pinMode(led_azul, OUTPUT);
  pinMode(sensor_izq, INPUT);
  pinMode(sensor_der, INPUT);
  digitalWrite(led_azul, LOW);
  digitalWrite(led_verde, LOW);
  digitalWrite(led_rojo, LOW);
  detener_motores();
  Serial.begin(9600);
  delay(1000);
  estado_robot = DECIDIR;
  medir_sensores_us();
  t_0 = millis();
}
//---------------------------- PROGRAMA PRINCIPAL ---------------------------------//
/* Referencias:
- Sensor A: Adelante
- Sensor B: Derecho adelante
- Sensor C: Derecho atras
*/

void loop(){
  // Para evitar interferencias, mido 5 veces con el sensor A y busco la mediana
  //Distancia_B = sensor_B.ping_cm();
  //Distancia_C = sensor_C.ping_cm();
  // La funcion me devuelve 0 si no recibe echo en 300 cm
  medir_sensores_us();
  serial();
  switch(estado_robot)
  { case DECIDIR:{
      digitalWrite(led_azul, HIGH);
      tomar_decision();
```

```
      digitalWrite(led_azul, LOW);
      break;
    }
    case AVANZAR:{
      digitalWrite(led_verde, HIGH);
      avanzar_celda();
      digitalWrite(led_verde, LOW);
      break;
    }
    case GIRAR_DERECHA:{
      digitalWrite(led_rojo, HIGH);
      girar_derecha();
      digitalWrite(led_rojo, LOW);
      break;
    }
    case GIRAR_IZQUIERDA:{
      digitalWrite(led_rojo, HIGH);
      girar_izquierda();
      digitalWrite(led_rojo, LOW);
      break;
    }
    case DETENER:{
      detener_motores();
      break;

    }
  }
}
```