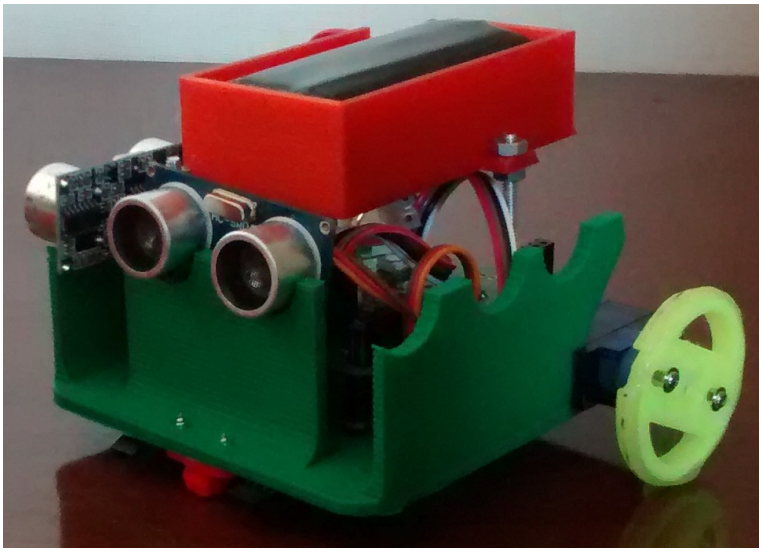
Competencia de robótica

JOHNNY 5

~2015~

Categoría: **LABERINTO**

Nombre robot: **LUJU**



Institución: **CDR FIUBA**

Participantes: (Nombre completo y mail)

- **Julieta Marasco (juli.marasco@gmail.com)**
- **Lucia Prado (prado.lucia1@gmail.com)**

Introducción:

La presente documentación describe el hardware y funcionamiento del robot LuJu, el cual se basa en una plataforma Dreamster. El objetivo de este robot es resolver un laberinto en la menor cantidad de tiempo. Aquí presentamos las partes que componen el robot, la electrónica y su código.

Descripción del robot:

Como se aclaró en la introducción LuJu está basado en la plataforma Dreamster[1]. Esta plataforma consta de un Arduino junto con un *shield* para conectar los servomotores, sensores de distancia por ultrasonido y sensores infrarrojos (en esta competencia éstos últimos no fueron utilizados).

El objetivo principal de este robot fue iniciarnos en la programación de Arduino para conocer las estructuras del lenguaje C y aprender el funcionamiento de cada componente del robot. Existe una librería exclusiva de Dreamster pero no se utilizó para esta competencia.

Las partes que componen a LuJu son:

Ruedas: están impresas con una impresora 3D, tienen tracción trasera. Lo que significa que hay motores independientes en las ruedas de un mismo eje. Se usó este modelo porque es de construcción sencilla y permite radios de giro del orden del tamaño del vehículo.

Motores: los utilizados fueron los Servo Tower Pro Sg90 9g[2]. Los mismos se encuentran modificados en el interior, ya que se les eliminó toda la electrónica dejando únicamente la mecánica para obtener así motores de continua con reducción. Estos servomotores presentan un torque de 1,80 Kg/cm.

1 *http://dreamsterbot.org/es/*
2 *http://www.servodatabase.com/servo/towerpro/sg90*

Arduino Leonardo: Arduino[3] es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares. Puede tomar información del entorno a través de sus entradas analógicas y digitales. Hay varios modelos de Arduino, en este caso se utilizó un Arduino Leonardo.

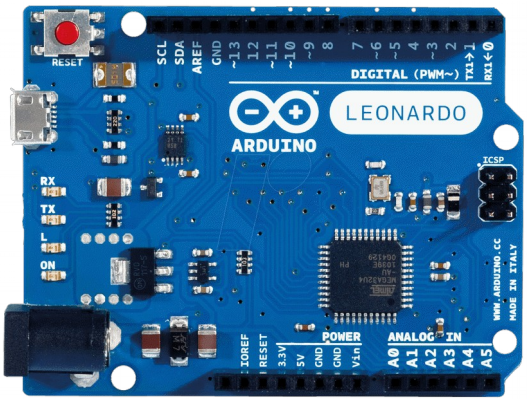


Ilustración 1: Aspecto del Arduino Leonardo

Shield: Para manejar los motores de corriente continua se usó un circuito integrado (DRV8833[4]) que tiene dos puentes H. Con el puente H se puede cambiar el sentido de los motores (avance, retroceso o freno). [Ver Apéndice]

Carcasa impresa: la misma fue impresa con una impresora 3D. El diseño del robot es libre y puede ser reproducido totalmente. [Ver Apéndice]

Batería: para este robot se usó una batería recargable de 7,4 V de Li-Ión. La misma entrega una intensidad de corriente de 2,2 Ah.

Sensores ultrasonido: Los sensores utilizados son los HC - SR04[5]. Estos sensores permiten medir la distancia a un objeto mediante el uso de pulsos ultrasónicos. A través del Trigger se envía un pulso que le avisa al sensor que debe empezar a medir. Luego, el sensor envía una señal ultrasónica que rebota

3 *http://www.arduino.cc*
4 *http://www.alldatasheet.es/datasheet-pdf/pdf/437529/TI1/DRV8833.html*
5 *http://www.electroschematics.com/8902/hc-sr04-datasheet/*

contra el objeto y vuelve al sensor. Esta señal que vuelve ingresa por el pin Echo e indica un proporcional a la distancia a la que se encuentra el objeto.

Los esquemáticos y partes impresas del robot se encuentran en su respectivo repositorio de Github[6]. Como se mencionó anteriormente el objetivo de hacer este robot era iniciarnos en el uso de Arduino y de la programación en lenguaje C, por eso simplificamos la mecánica y electrónica al máximo a modo de poder concentrarnos en lo que nos interesaba. Como guía utilizamos el tutorial[7] de Dreamster.

La siguiente tabla muestra las conexiones del Dreamster en la placa de Arduino Leonardo:

Función	Pin Name	Función	Pin Name
Motor Izquierda P	9	Sensor US A Trigger	A3
Motor Izquierda N	10	Sensor US A Echo	8
Motor derecha P	5	Sensor US B Trigger	A4
Motor derecha N	6	Sensor US B Echo	2
Led Azul	13	Sensor US C Trigger	A5
Led Rojo	12	Sensor US C Echo	7
Led Verde	11		

La distribución de sensores que utilizamos es, mirando desde el frente, dos sensores a la derecha (B y C) y uno al frente (A).

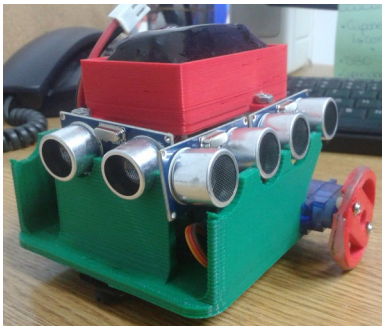


Ilustración 2:
Distribución de sensores.

6 *https://github.com/dreamster*
7 *http://dreamsterbot.org/es/comunidad/*

Con esta distribución de sensores laterales logramos que el robot avance a una distancia constante de la pared, ya que se comparan los valores que obtiene cada uno de los sensores laterales. El sensor frontal lo utilizamos para detectar si hay un obstáculo al frente. El laberinto se resuelve siguiendo siempre la pared izquierda.

Código:

```
/*Definicion de los pines necesrios para el funcionamiento de los sensores de ultrasonido.*/
/*Se determina su ubicacion de la siguiente manera: A=adelenate; B=lateral_atras; C=lateral_adelante*/
const int TriggerPinA = A3;
const int EchoPinA = 8;
const int TriggerPinB = A4;
const int EchoPinB = 2;
const int TriggerPinC = A5;
const int EchoPinC = 7;

/*Definicion de variables para medir la distancia de la pared con el sensor*/
int DistanceA_mm;
int DistanceB_mm;
int DistanceC_mm;

/*Definicion de los pines necesarios para el funcionamiento del led*/
const int ledA = 13;
const int ledR = 12;
const int ledV = 11;

/*Deficion de los pines necesarios para el funcionamiento del motor*/
int kMotorLP = 9;
```

```
int kMotorLN = 10;
int kMotorRP = 5;
int kMotorRN = 6;

const int Vel_MotorR_adelante = 45;
const int Vel_MotorL_adelante = 47;

/*Definicion de varibles varias, necesarias para el desarrollo del código*/
int Diferencia;
int outA;
int outB;
int outC;
int duration;

const int debugMode = 0;



void setup() {

/*Iniciallizacion de pines como entradas o salidas*/
  pinMode(ledA, OUTPUT);
  pinMode(ledR, OUTPUT);
  pinMode(ledV,OUTPUT);

  pinMode(TriggerPinA, OUTPUT);
  pinMode(EchoPinA, INPUT);
  pinMode(TriggerPinB, OUTPUT);
  pinMode(EchoPinB, INPUT);
  pinMode(TriggerPinC, OUTPUT);
  pinMode(EchoPinC, INPUT);

  Serial.begin(9600);
```

```
/*Inicializacion de variables, primera escritura*/
  digitalWrite(ledA,LOW);
  digitalWrite(ledR,LOW);
  digitalWrite(ledV,LOW);

  digitalWrite(kMotorLP, LOW);
  digitalWrite(kMotorLN, LOW);
  digitalWrite(kMotorRP, LOW);
  digitalWrite(kMotorRN, LOW);
}

int readSensor(int trigger, int echo) {
   digitalWrite(trigger, LOW);
   delayMicroseconds(2);
   digitalWrite(trigger, HIGH);
   delayMicroseconds(10);
   digitalWrite(trigger, LOW);

   duration = pulseIn(echo, HIGH);

   if (duration < 0) duration = 0;
   return ((duration / 2.9) / 2);
}

void loop() {

 /*Lectura de la distancia de los sensores ultrasónicos con la pared*/
  DistanceA_mm = readSensor(TriggerPinA, EchoPinA);
  delay (60);
  DistanceB_mm = readSensor(TriggerPinB, EchoPinB);
  delay (60);
```

```
  DistanceC_mm = readSensor(TriggerPinC, EchoPinC);

  outA = map(DistanceA_mm, 0, 1023, 120, 255);
  analogWrite(ledR, outA);
  outB = map(DistanceB_mm, 0, 1023, 0, 255);
  analogWrite(ledA, outB);
  outC = map(DistanceC_mm, 0, 1023, 0, 255);
  analogWrite(ledV, outC);

/*Cálculo de la diferencia de lectura entre los sensores laterales*/
  Diferencia = DistanceB_mm - DistanceC_mm;

/*Separacion por caso según resultado de la variable anterior*/

/*Caso 1: los sensores estan a una distancia de la pared izquierda entre 5 y 9 cm, la diferencia entre sensores es menor a 2 cm*/

  if ( (Diferencia < 20) && (DistanceB_mm > 50) && (DistanceB_mm < 90) && (DistanceC_mm > 50) && (DistanceB_mm < 90) && (DistanceA_mm > 70) ) {
      //Los motores deben avanzar derecho
        analogWrite(kMotorLP, Vel_MotorL_adelante);
        analogWrite(kMotorLN, 0);
        analogWrite(kMotorRP, Vel_MotorR_adelante);
        analogWrite(kMotorRN, 0);
        if (debugMode) {
          Serial.print("caso_1 = ");
          Serial.println(Diferencia, DEC);
          delay(500);
        }
 }

/*Caso 2: algún sensor esta más lejos que 9 cm de la pared izquierda*/
```

```
  else if ( ((DistanceB_mm > 90) || (DistanceC_mm > 90)) && (DistanceA_mm > 70) ) {

/*Caso 2_a: la diferencia es mayor que 0, por ende DistanceC_mm es menor que DistanceB_mm.*/
      if (Diferencia > 0) {
        //El motor izquierdo debe ir más rápido para "enderezar" el robot
        analogWrite(kMotorLP, Vel_MotorL_adelante - 5);
        analogWrite(kMotorLN, 0);
        analogWrite(kMotorRP, Vel_MotorR_adelante - 5);
        analogWrite(kMotorRN, 0);
        if (debugMode) {
          Serial.print("caso_2a = ");
          Serial.println(Diferencia, DEC);
          delay(500);
        }
      }

/*Caso 2_b: la diferencia es menor que 0, por ende DistanceC_mm es mayor que DistanceB_mm.*/
    if (Diferencia < 0) {
      //El motor derecho debe ir más rápido para "enderezar" el robot
      analogWrite(kMotorLP, 0);
      analogWrite(kMotorLN, 0);
      analogWrite(kMotorRP, Vel_MotorR_adelante - 10);
      analogWrite(kMotorRN, 0);
      if (debugMode) {
         Serial.print("caso_2b = ");
         Serial.println(Diferencia, DEC);
         delay(500);
        }
```

```
    }

  }

/*Caso 3: algún sensor esta a menos de 5 cm de la pared izquierda*/

  else if ( ((DistanceB_mm < 50) || (DistanceC_mm < 50)) && (DistanceA_mm > 70) ) {

/*Caso 3_a: la diferencia es mayor que 0, por ende DistanceC_mm es menor que DistanceB_mm.*/
      if (Diferencia > 0) {
        //El motor izquierdo debe ir más rápido para "enderezar" el robot
        analogWrite(kMotorLP, Vel_MotorL_adelante - 10);
        analogWrite(kMotorLN, 0);
        analogWrite(kMotorRP, 0);
        analogWrite(kMotorRN, 0);
        if (debugMode) {
          Serial.print("caso_3a = ");
          Serial.println(Diferencia, DEC);
          delay(500);
        }
      }

/*Caso 3_b: la diferencia es menor que 0, por ende DistanceC_mm es mayor que DistanceB_mm.*/
    if (Diferencia < 0) {
      //El motor derecho debe ir más rápido para "enderezar" el robot
      analogWrite(kMotorLP, Vel_MotorL_adelante);
      analogWrite(kMotorLN, 0);
      analogWrite(kMotorRP, Vel_MotorR_adelante);
      analogWrite(kMotorRN, 0);
```

```
    if (debugMode) {
        Serial.print("caso_3b = ");
        Serial.println(Diferencia, DEC);
        delay(500);
      }
   }

 }

/*Caso 4: se llega a una esquina, por ende debe girar*/

 else if ( (DistanceA_mm < 70) && ( (DistanceB_mm < 100) || (DistanceC_mm
< 100) ) ) {
    while(DistanceA_mm<70){
      // gira a la derecha
      analogWrite(kMotorLP, Vel_MotorL_adelante);
      analogWrite(kMotorLN, 0);
      analogWrite(kMotorRP, 0);
      analogWrite(kMotorRN, Vel_MotorR_adelante);
      DistanceA_mm = readSensor(TriggerPinA, EchoPinA);
      delay (15);
      if (debugMode) {
         Serial.print("caso_4a_A = ");
         Serial.println(DistanceA_mm, DEC);
         Serial.print("caso_4a_B = ");
         Serial.println(DistanceB_mm, DEC);
         delay(500);
      }
    }
   }
   else {
    while(DistanceA_mm<70){
```

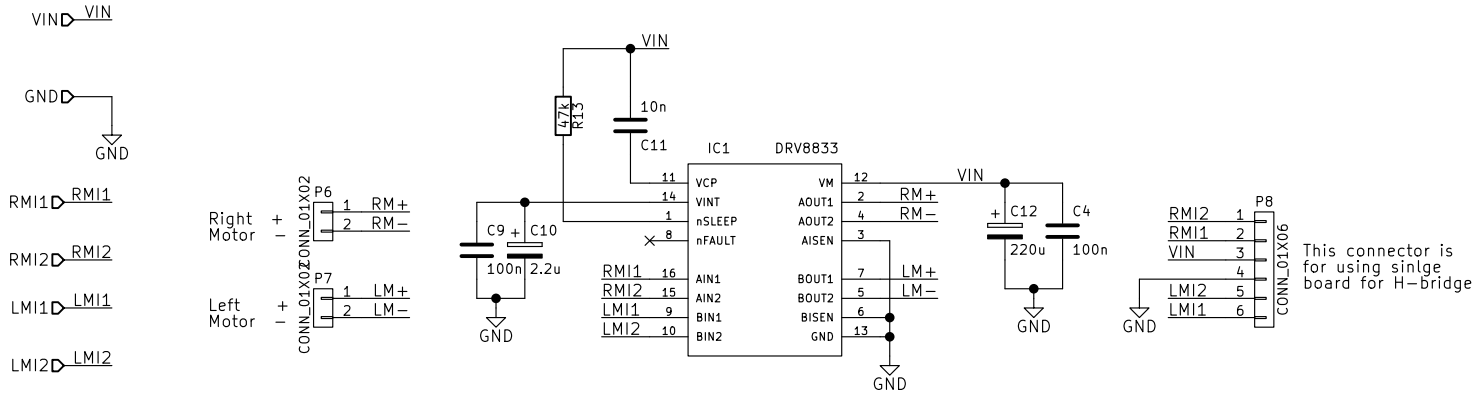
```
      // gira a la izquierda
      analogWrite(kMotorLP, 0);
      analogWrite(kMotorLN, Vel_MotorL_adelante);
      analogWrite(kMotorRP, Vel_MotorR_adelante);
      analogWrite(kMotorRN, 0);
      DistanceA_mm = readSensor(TriggerPinA, EchoPinA);
      delay (15);
      if (debugMode) {
        Serial.print("caso_4b_A = ");
        Serial.println(DistanceA_mm, DEC);
        Serial.print("caso_4b_B = ");
        Serial.println(DistanceB_mm, DEC);
        delay(1000);
       }
     }
     //Los motores deben avanzar derecho
     analogWrite(kMotorLP, Vel_MotorL_adelante);
     analogWrite(kMotorLN, 0);
     analogWrite(kMotorRP, Vel_MotorR_adelante);
     analogWrite(kMotorRN, 0);
     delay (2000);
   }


  //delay(20);
}
```

Apéndice:

Los diseños del chasis, el porta batería y ruedas se puede descargar del siguiente link: https://github.com/dreamster/dreamster-cad

El esquemático del s*hield* se encuentra a continuación:



Dreamster	
Sheet: /hbridge/	
File: hbridge.sch	
Title: H−bridge	
Size: A4	Date: mié 18 feb 2015 **Rev: 0.1**
KiCad E.D.A. kicad 0.201508100901+608028ubuntu14.04.1−product	Id: 4/4