Building and programming an autonomous Robot

García Sebastián

MatesLab

Mar del Plata, Buenos Aires, Argentina.

eldraco@gmail.com

Abstract

This paper is intended to show how it was designed, built and programmed an autonomous robot cheap controlled using the Arduino open platform. Having no prior experience in electronics, it was necessary to learn almost every electronic principle and to overcome a lot of mechanical problems. New distance sensors had to be created from scratch and several test were carried on before ending with a working prototype.

Keywords: robotics, arduino, microcontroler

1. Introduction

Making robots and learn how to program them has been and it is the dream of a lot of young boys, hackers, researchers and a lot people in general. Being able to design and build your own robot seems to boost the imagination and creativity. Some robot build-your-self kits had been even sold in street newspapers kiosks with success and a lot of academic and non-academic work has been done in the field. New kits and peripherals, now cheap enough to be bough by most of people has also helped to bring this desire closer.

Nevertheless, building a robot still needs of microcontrolers to be programmed and electronic components to be understood, but even this complex subject is more and more accessible now a days. Programming languages are more intuitive than before, and new microcontrolers building-frameworks are available, making the building of a robot an achievable issue.

This work was born as a side-project of an information science engineer that wants to learn and build its own robot and also go deeper into the insights of microcontrolers programming. Having a strong background in computer programming and mechanical developments make it easier to understand and implement the ideas, resulting that after several months of study, testing and errors, the Toti robot was finished.

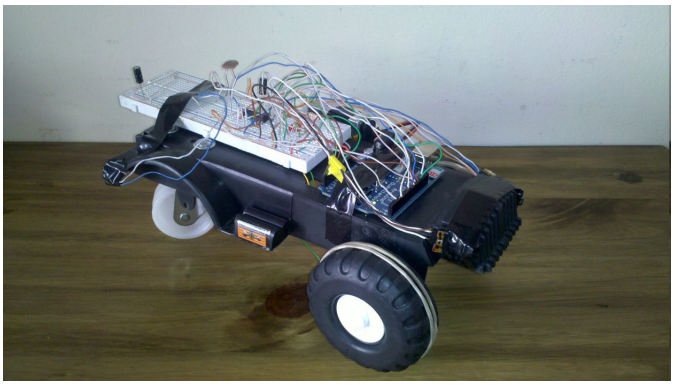

Figure 1: Toti finished

Toti, as it can be seen finished in Figure 1, was built upon the Arduino platform with the important premise of being as cheap as possible. We want to probe that using almost only recycled electronic components (like an old CD player) a cheap robot can be made at home. The complete list of Toti's features is:

- Double independent front-motor powered traction.
- Object detection and avoidance in dark or light conditions.
- Ambient light sensing.
- Dynamic light threshold configuration.
- Start and stop button.

This paper goal is to show the building and development process of Toti, its mayor advantages and disadvantages, the designs decisions taken and the results achieved, hoping they will be of some help to other researchers.

Our main contributions are:

1. The design of a cheap and functional robot.
2. The design of an object detection pulse-based algorithm.

In Section 2 we show the basic designs choices behind Toti, in Section 3 the most important steps of its construction are shown, in Section 4 we share the complete Toti program, in Section 5 we share some common problems and fixes when testing Toti, in Section 6 we show the future PCB board that is under construction and in Section 7 we present our conclusions.

2. Design

Toti's design can be described from both the mechanical/electronical point of view or the programming point of view. From the mechanical point of view Toti has a simple design, implementing only three wheels with two motors in the front. This may not be the most implemented and optimal configuration, but was useful enough to work fine. At the front it also has two infrared custom-made distance sensors which helps Toti to avoid objects and a light sensor to determine when to stop and hide in the dark.

From the programming point of view, Toti was developed using the Arduino User Interface, that makes it easy to control the different components and sensors. This interface gives also the opportunity to program the AtMega microcontroler with every low-level original primitives. At first we choose to use the Arduino Duemilanove open platform because of its simplicity, wide adoption, user base support and powerful capabilities. The Arduino has made it easy to create the robot. In the next subsections we are going to better describe every component of Toti.

2.1. Arduino

The Arduino platform[1] was selected because of its easy of adoption, development and versatility. We are not going to talk long about Arduino in this paper, but some of the features that make us use it were:

- Uses FTDI chip for USB communication.
- Uses ATmega328 Microcontroller.
- Digital I/O Pins 14 (of which 6 provide PWM output).
- Analog Input Pins 6.
- DC Current per I/O Pin 40 mA.
- DC Current for 3.3V Pin 50 mA.
- Clock Speed 16 MHz.
- It is immensely popular and used as the core for many educational classes.
- It has both 3.3V and 5V regulated power supplies broken out.
- It has both USB and barrel jack ports for easy power, and communication.
- New to the Duemilanove is an auto power detection circuit. This will automatically select power from the barrel jack or USB.
- All pins are broken out to female headers for easy connections.

[1] `http://www.arduino.cc`

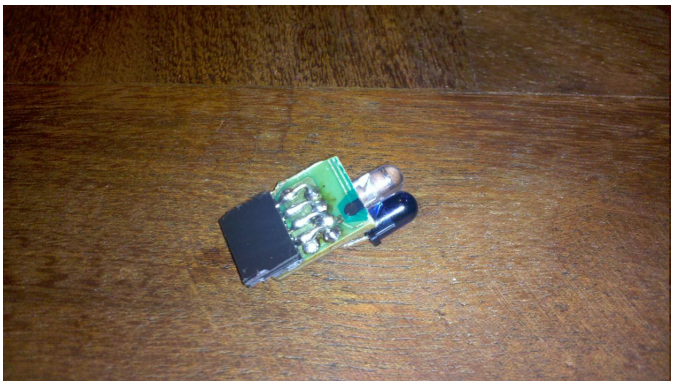

Figure 2: Self made IR sensor with one IR emitter and one IR receiver.

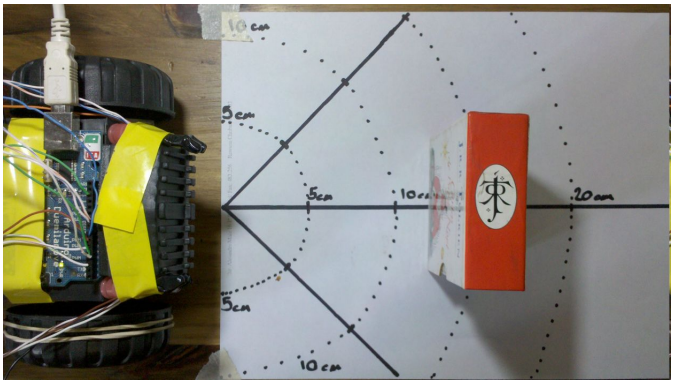

Figure 3: IR sensor calibration

2.2. Sensors

Usually, distance sensors are an expensive piece of equipment and not very easy to find sometimes. This limitation has lead us to experiment and create our own distance infrared sensors. Playing with sensors has been very entertaining and educational and shed light over many of the complex situations that we can find when building a robot. The first tests were done using recycled mouse IR sensors, which worked fine but had a very short distance range. After these experiments, we buy two infrared light led receivers and two infrared led transmitters to build some home made distance sensors. These sensors, as seen in Figure 2, were adjusted and tested until we found the correct way to position them and detect objects successfully. This leds were weld into a broken PCB board to avoid movement and a four-row connector was attached to their end, granting an easy assembly and disassembly.

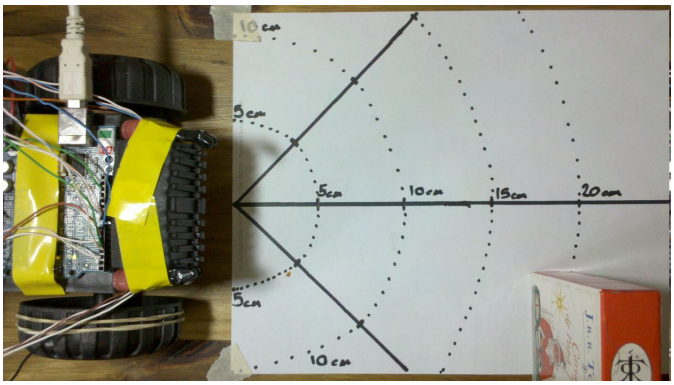

Figure 4: IR sensors calibration to detect sided objects

2.2.1. Object detection

Our sensors does not have any internal logic or pre-programming capabilities, so we need to develop the detection logic in the main Arduino program. Object detection was designed to avoid objects in Toti's path, but multiple configurations were possible. Sensors were positioned 45 degree from Totis largest axis in order to detect objects near the wheels. When, in the first tests, sensors were positioned alongside (in parallel) with the largest Toti's axis, we were only able to detect objects going straight to the sensors. In Figure 3, we show how these sensors were calibrated to accomplish its task without problems. For example, in Figure 4, we can see that Toti is able to detect objects perfectly almost 17cm ahead and not fully in front of it. This was achieved mainly because of the use of 45 degree IR sensors and a proper configuration.

Thanks to José Marone advisory, we developed a detection algorithm which was configured to send and detect the reflection of an infrared (from now on IR) beam. The main idea is to generate IR pulses with a certain shape and to detect them with the IR receiver at the same moment. When no pulse is being generated, no pulse should to be detected with the sensor. This is the main reason of why we can detect objects with different light conditions and in the dark. An schema of the pulse detection principle can be seen in Figure 5.

2.3. Light detection

Toti was equipped with an LDR light sensor to detect when it is under some light threshold value in order to stop working. The sensor was originally configured to work with a fixed value but this caused troubles when the environmental light characteristics change. To solve this issue, we reprogrammed the button functionality to enable a quick light threshold value configuration.

The new idea, was that before using Toti you could configure it on-the-fly to work under the light characteristics of the place where you are. To configure Toti

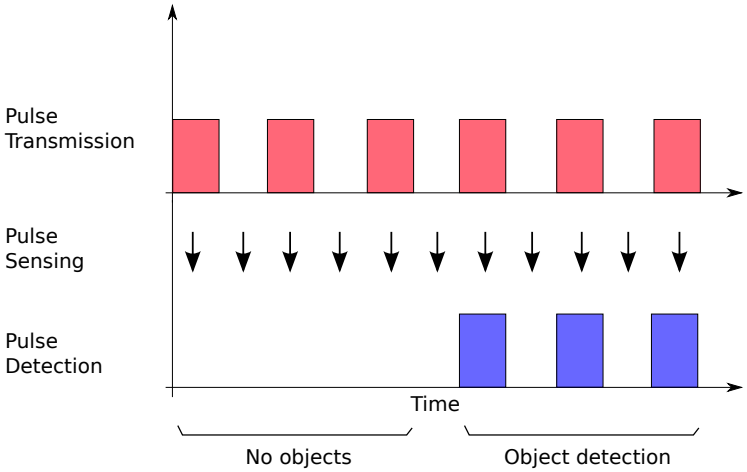

Figure 5: Object detection schema

dynamically, you must put it under the amount of light that you want to trigger the *stop* condition and quickly double press the start button. If everything goes right, the status led will blink twice and Toti will be ready to be used properly. This dynamic configuration of Toti empower the test of Toti under different circumstances.

2.4. Start and Stop functions

To stop and start the robot, it was equipped with a recycled CD reader button. This button was placed at the rear of the robot and allows you to stop and start it properly.

2.5. Motors

When trying to design the traction base, we have the same problem of very expensive components as with sensors. It is very difficult to find small and cheap motors in the market, so we decided to reuse old CD players DC motors. These motors are very powerful and three of them can be found on every old CD player. We manage to control them using the PWM (pulse width modulation) techniques embedded in the Arduino. In Figure 6 we can see our first motor tests, trying to find out how was the optimal performance of it.

Motor's gear used in Toti were also the ones coming along with the motors in the CD players, because they were already designed to work properly. After a correct assembly, we can see in Figure 7 how the two main motors were put in place. This last part of the motor building was the most time consuming and difficult, because we have to literally completely build the whole schema.

Robot motors were controlled by an L293D H-bridge.

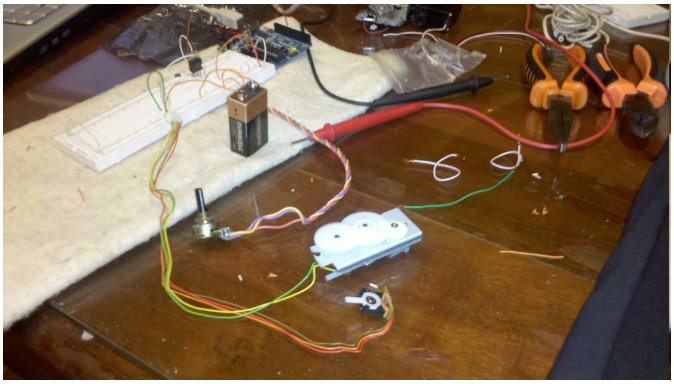

Figure 6: DC motor test with a potentiometer.

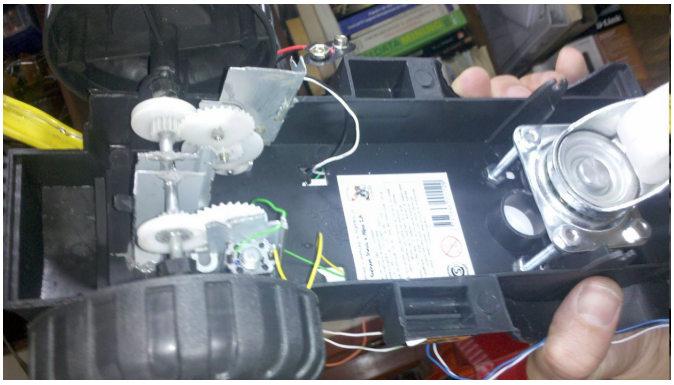

Figure 7: Motors detail and back wheel detail.

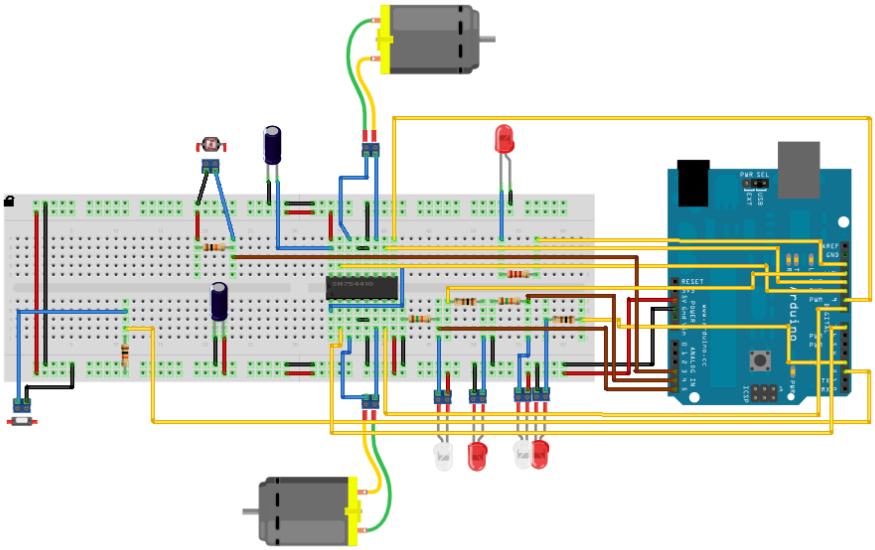

Figure 8: Toti's final connection schema

3. Building procedure

Toti connection schema was done with the Fritzing software[2], as can be seen in Figure 8, and it allowed us to clearly design, measure and think every aspect of it.

Toti was first build in a shoes box, until every test was finished. Once we knew which motors and sensors we were going to use, and that the program was executing correctly, we start to build the final Toti shape from a tractor toy's structure, as shown in Figure 9. From this base we create and assembly the motors (Section 2.5), the sensors and the rest of the components.

3.1. Total cost

Toti's build cost was approximately of U$S 55, including the Arduino Duemilanove, but our latest research about Arduino replacements has given us a new opportunity to lower the total cost of Toti to U$S41 welding our own *Severino* board[3].

[2] `http://fritzing.org/`
[3] `http://www.arduino.cc/en/Main/ArduinoBoardSerialSingleSided3`

Figure 9: Trunk chassis to build Toti

4. Program

Arduino comes with a great user interface to correctly develop and program the AtMega microcontroler. The USB connection makes it really easy to develop, upload and test the improvements within seconds, speeding up the process.

The final Toti's program can be found at AppendixA. Development was done trying to create functions that could be reused later and easily modified using no external libraries except for the AVR include files. Most of the functionality was achieved using common Arduino's functions, but when we need to manage time interrupts correctly, we have to use AtMega's functions directly (i.e. ISR function)

The AtMega programming was quickly done mostly because of Arduino's capability of wrapping up the microcontroler functions. For example, when handling the digital outputs, we only need to assign the *HIGH* value using the digitalWrite function.

5. Testing

Toti was tested at every stage for errors and corrections. And several videos of these test can be seen online. The complete list of videos is:

- Toti was born: `http://www.youtube.com/user/el2000draco#p/u/11/F_7-pWR3jZU`
- Second test, one motor: `http://www.youtube.com/user/el2000draco#p/u/10/-9bnMqFHg70`
- Third test, one faster DC motor: `http://www.youtube.com/user/el2000draco#p/u/9/73bsBmH0bQA`

- Fourth test, DC motor speed control: http://www.youtube.com/user/el2000draco#p/u/8/B5hEAzHt7x4
- Toti has an eye using a mouse optical infrared sensor: http://www.youtube.com/user/el2000draco#p/u/7/1zriACR8kYY
- Toti with a distance sensor dodging objects: http://www.youtube.com/user/el2000draco#p/u/6/qMGdY0s7Rvo
- Toti is smarter, dodging objects with and without light: http://www.youtube.com/user/el2000draco#p/u/3/qu_qIJtRKS8
- Toti with two motors dodging objects: http://www.youtube.com/user/el2000draco#p/u/2/Dyvbyy-sdTY
- Toti with two motors, dodging objects with more control: http://www.youtube.com/watch?v=g_S219gpiJQ
- Toti using two DC motors, dodging objects with two sensors and sensing daylight to sleep: http://www.youtube.com/user/el2000draco#p/u/0/lX3ZSk9cZ40
- Added capacitors to add strength to the hardware design: http://www.youtube.com/user/el2000draco#p/a/u/0/4v82Kq66QBw

6. PCB construction

After several months of usage, Toti shows great stability and precision, allowing us to continue to the next phase of development. We started to design and build the final PCB board that will hold Toti's components from now on. In Figure 10 we can see the Fritzing design of this future PCB board, which was made to match exactly with the Arduino shield shape.

7. Conclusion

This project has been very educational to us, resulting in an entertained and useful experience. We were able to create and program a complete and functional robot with U$S 55, which was tested on several different environments. The Arduino platform was a key component that worked fine along the tests, even supporting some electrical misconfigurations. We want to thanks the MatesLab members and the teachers of the UNICEN University who have made possible the finalization of this project.

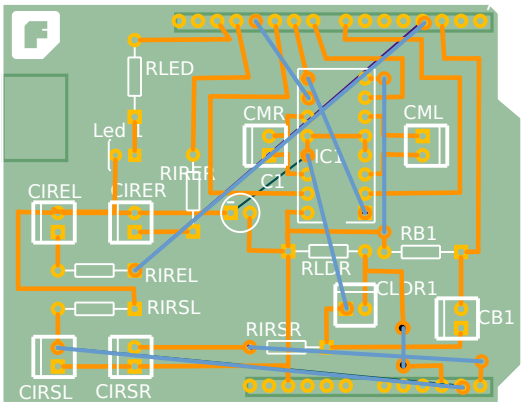

Figure 10: Toti's future PCB board shield

AppendixA. Arduino PDE program

```
#include <avr/io.h>
#include <avr/interrupt.h>

// Definitions
#define INIT_TIMER_COUNT 0
#define RESET_TIMER2 TCNT2 = INIT_TIMER_COUNT


  //
  // Constants
  //
  const int switchPin = 2;     // switch input. Digital
  // Motor 1 is right motor
  const int motor1Pin1 = 10; // H-bridge leg 1 (Motor 1 pin 1)
  const int motor1Pin2 = 11; // H-bridge leg 2 (Motor 1 pin 2)
  // Motor 2 is left motor
  const int motor2Pin1 = 8;     // H-bridge leg 3 (Motor 2 pin 1)
  const int motor2Pin2 = 7;     // H-bridge leg 4 (Motor 2 pin 2)

  const int enablePin = 9;     // H-bridge enable pin
  const int ledPin = 13;       // LED
  const int irPin = 5; // Lector de luz 1. Analogic
  const int irPin2 = 4; // Lector de luz 2. Analogic
  const int irdPinOutput = 3; // Digital. IR Emitter 1 (Left)
  const int irdPinOutput2 = 12; // Digital. IR Emitter 2 (Right)
  const int ldrInput1 = 3; // Analogic.Light sensor
```

```
// Variables
  int ldrHideValue = 900 ; // Ldr hide value. Under this light value
                           // , toti stop traveling because it wants to hide
                           // in shadow!
  int velocidad = 0;
  int irValue = 0;
  int previousVel = 0;
  int distanceValue = 10;  // Valor ad-hoc que parece funcionar
                           // para el sensor ir 1 de entrada
  int distanceValue2 = 10;  // Valor ad-hoc que parece funcionar
                           // para el sensor ir 2 de entrada.
                           // IR sensor 2 is the right one.
  long time = 0;           // la ultima vez que el pin de salida fue
                           // basculado. For powerin on and off.
  long time2 = 0;           // la ultima vez que el pin de salida fue
                           // basculado. For ldr configuration.
  long debounce = 400;     // Button's DEbounce time. It's to avoid doing
                           // something WHILE you are pressing the button
                           // from top to bottom. Buttons' state reading is
                           // very fast, and this helps to wait until you
                           // finish pushing the button to do some action.
  int reading;             // lectura actual del pin de entrada
  int previous = LOW;      // lectura anterior del pinX de entrada
  int state = LOW;         // estado actual del pin de salida
  int int_counter = 0;
  int obstacle[2];
  int ldrValue1 = 0;  //
  int ldrMaxValue = 0;  // Ldr max value


/*
 Timer handler interruption
*/
ISR(TIMER2_OVF_vect) {
  int_counter += 1;

  // This make us check obstacles and light almost every 500ms..
  if (int_counter == 100) {
      senseObstacle(&obstacle[0]);
      if (obstacle[0] == HIGH){
        Serial.println("There is an obstacule in the right");
      }
      if (obstacle[1] == HIGH){
        Serial.println("There is an obstacule in the left");
      }
```

```
      int_counter = 0;


      // light management
      ldrValue1 = analogRead(ldrInput1);
      //Serial.println(ldrValue1);
      if (ldrValue1 >= ldrMaxValue) {
         ldrMaxValue = ldrValue1;
      }

  }
}




// Sense obstacles
void senseObstacle(int *obstacle)
{
  int valor1_P2 = 0; // IR Sensor 2
  int valor1_A2 = 0; // IR Sensor 2
  int valor2_P2 = 0; // IR Sensor 2
  int valor2_A2 = 0; // IR Sensor 2

  int valor1_P1 = 0; // IR Sensor 1
  int valor1_A1 = 0; // IR Sensor 1
  int valor2_P1 = 0; // IR Sensor 1
  int valor2_A1 = 0; // IR Sensor 1

  int i;

  //// Second IR
  // First pulse
  digitalWrite(irdPinOutput,HIGH);
  valor1_P1 = analogRead(irPin2);

  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }
  digitalWrite(irdPinOutput,LOW);
  valor1_A1 = analogRead(irPin2);
  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }
```

```
  // Second pulse
  digitalWrite(irdPinOutput,HIGH);
  valor2_P1 = analogRead(irPin2);
  //Serial.println(millis());
  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }
  digitalWrite(irdPinOutput,LOW);
  valor2_A1 = analogRead(irPin2);
  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }

// End second IR


  // First IR
  // First pulse
  digitalWrite(irdPinOutput2,HIGH);
  valor1_P2 = analogRead(irPin);

  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }
  digitalWrite(irdPinOutput2,LOW);
  valor1_A2 = analogRead(irPin);
  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }

  // Second pulse
  digitalWrite(irdPinOutput2,HIGH);
  valor2_P2 = analogRead(irPin);
  //Serial.println(millis());
  for(i=0;i<500;i++)
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }
  digitalWrite(irdPinOutput2,LOW);
  valor2_A2 = analogRead(irPin);
  for(i=0;i<500;i++)
```

```
  {
    __asm__("nop\n\t""nop\n\t""nop\n\t""nop\n\t""nop\n\t");
  }
  // End first IR

//
// IR Sensor 2
//  Serial.print("S1:P2=");
//  Serial.print(valor2_P2);
//  Serial.print(":A2=");
//  Serial.print(valor2_A2);
//  Serial.print(" (");
//  Serial.print(valor2_A2 - valor2_P2);
//  Serial.print(") - ");

// IR Sensor 1
//  Serial.print("S2:P2=");
//  Serial.print(valor2_P1);
//  Serial.print(":A2=");
//  Serial.print(valor2_A1);
//  Serial.print(" (");
//  Serial.print(valor2_A1 - valor2_P1);
//  Serial.println(")");

  // We detected an object at right
  if ((valor2_A1 - valor2_P1) > distanceValue){
    obstacle[0] = HIGH;
  }
  else{
    obstacle[0] = LOW;
  }

  // We detected an object at left
  if ((valor2_A2 - valor2_P2) > distanceValue2){
    obstacle[1] = HIGH;
  }
  else{
    obstacle[1] = LOW;
  }

}
```

```
void setup() {
  // Setting pin modes
  pinMode(switchPin, INPUT);
  pinMode(irPin, INPUT);
  pinMode(ldrInput1,INPUT);
  pinMode(motor1Pin1, OUTPUT); //motor1
  pinMode(motor1Pin2, OUTPUT); //motor1
  pinMode(motor2Pin1, OUTPUT); //motor2
  pinMode(motor2Pin2, OUTPUT); //motor2
  pinMode(enablePin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(irdPinOutput, OUTPUT);
  pinMode(irdPinOutput2, OUTPUT);


  // set enablePin high so that motors can work
  digitalWrite(enablePin, HIGH);

  Serial.begin(9600);
  // blink the LED 3 times. This should happen only once.
  // if you see the LED blink three times, it means that the module
  // reset itself,. probably because the motor caused a brownout
  // or a short.
  blink(ledPin, 3, 100);


  // Button management
  // This specifies a function to call when an external interrupt occurs
  // Most Arduino boards have two external interrupts: numbers 0
      // (on digital pin 2) and 1 (on digital pin 3)
  //
  // Third parameter defines when the interrupt should be triggered.
      // Four contstants are predefined as valid values:
  // * LOW to trigger the interrupt whenever the pin is low,
  // * CHANGE to trigger the interrupt whenever the pin changes value
  // * RISING to trigger when the pin goes from low to high,
  // * FALLING for when the pin goes from high to low.
  attachInterrupt(0, prender, FALLING);

  //
  // Timer setings
  //
  // No Timer Prescaler,
  TCCR2A |= ((0<<CS22) | (0<<CS21) | (1<<CS20));
  // Use normal mode
```

```
  TCCR2A |= (0<<WGM21) | (0<<WGM20);
  // Use internal clock - external clock not used in Arduino
  ASSR |= (0<<AS2);
  //Timer2 Overflow Interrupt Enable
  TIMSK2 |= (1<<TOIE2) | (0<<OCIE2A);
  RESET_TIMER2;
  //Enable Global Interrupts
  sei();
}
```

```
//////////////////////////
/////// STARTS ///////////
//////////////////////////
```

```
void loop() {

// State is controled by the button
if (state == HIGH and ldrValue1 <= ldrHideValue){
  // obstacle is managed by the sensors

  // If no obstacle at front, go ahead!
  if (obstacle[0] == LOW and obstacle[1] == LOW) {
   adelante('D',0,previousVel,150,30);

  // Is there an obstacle at right?
  }else if (obstacle[0] == HIGH){
   // Go backwards
   atras('D',1000,previousVel,255,40);
   // Turn left 45 degrees
   turn1('l',180);
   // Continue
  }
  else if (obstacle[1] == HIGH){
   // Go backwards
   atras('D',1000,previousVel,255,40);
   // Turn right 45 degrees
   turn1('r',180);
```

```
    // Continue

  }
  }else if (state == LOW or ldrValue1 > ldrHideValue){
     // Disable motor
     digitalWrite(motor1Pin1, LOW);    // set leg 1 of the H-bridge low
     digitalWrite(motor1Pin2, LOW);    // set leg 2 of the H-bridge low
     digitalWrite(motor2Pin1, LOW);    // set leg 3 of the H-bridge low
     digitalWrite(motor2Pin2, LOW);    // set leg 4 of the H-bridge low
     previousVel=0;
  }
  } // void




//////////////////////////
//////////ENDS//////////
//////////////////////////




////////// FUNCTIONS ////////

/*
motor1 is the right motor
motor2 is the left motor
To move any motor forwards we set pin2 low and pin1 high
*/


/*
turn1: Turn using both motors. One forwards and the other backwards
side is 'r' for right and 'l' for left
angle can be 45 or 90 for now.
*/
 void turn1(char side, int angle)
 {
   if (side=='r'){
       // Move second motor forwards and first motor backwards
      digitalWrite(motor1Pin1, LOW);
      digitalWrite(motor2Pin2, LOW);

      digitalWrite(motor1Pin2,HIGH);
```

```
      digitalWrite(motor2Pin1,HIGH);
      if (angle == 45){
        delay(250);
      }else if (angle == 90){
        delay(500);
      }else if (angle == 180){
        delay(1000);
      }
      digitalWrite(motor1Pin2,LOW);
      digitalWrite(motor2Pin1,LOW);

   }
   else if (side=='l'){
      // Move first motor forwards and second motor backwards
      digitalWrite(motor1Pin2, LOW);
      digitalWrite(motor2Pin1, LOW);

      digitalWrite(motor1Pin1,HIGH);
      digitalWrite(motor2Pin2,HIGH);
      if (angle == 45){
        delay(250);
      }else if (angle == 90){
        delay(500);
      }else if (angle == 180){
        delay(1000);
      }
      digitalWrite(motor1Pin1,LOW);
      digitalWrite(motor2Pin2,LOW);
   }
   previousVel = 0;

 }


/*
Both motors backwards
For function prototype see the 'adelante' function
*/
 void atras(char modo, int tiempo, int velMin, int velMax, int faseDeAcel)
 {
    digitalWrite(motor1Pin1, LOW);   // set leg 1 of the H-bridge low
    digitalWrite(motor2Pin1, LOW);   // set leg 3 of the H-bridge low
    for (velocidad=velMin;velocidad<=velMax;velocidad=velocidad+faseDeAcel){
      analogWrite(motor1Pin2,velocidad);
      analogWrite(motor2Pin2,velocidad);
      delay(0);
```

```
    }
    // We store the velocity were we ended in the previosVel
    previousVel = velMax;
    if (tiempo != 0){
      delay(tiempo);
    }
    if (modo == 'F'){
      // This Frenar function is called going backwards
      frenar('B',velMax,faseDeAcel);
    }
 }

/*
Both motors forward
if modo = 'D' it goes directly ahead without stopping.
If modo = 'F', it will stop after the time delay.
adelante(modo: 'D' para Directo (sin dejar de ir para adelante),
'F' para Frenar despues de cierto tiempo, tiempo que anda
hasta que frena,vel de comienzo, vel de final, aceleracion)
*/
 void adelante(char modo, int tiempo, int velMin, int velMax, int faseDeAcel)
 {
    digitalWrite(motor1Pin2, LOW);   // set leg 1 of the H-bridge low
    digitalWrite(motor2Pin2, LOW);   // set leg 1 of the H-bridge low
    for (velocidad=velMin;velocidad<=velMax;velocidad=velocidad+faseDeAcel){
      analogWrite(motor1Pin1,velocidad);
      analogWrite(motor2Pin1,velocidad);
      // This delay allows the robot to show the slow change.
      delay(10);
    }
    // We store the velocity were we ended in the previosVel
    previousVel = velMax;
    if (modo == 'F'){
      // This Frenar function is called going forwards
          if (tiempo != 0){
            delay(tiempo);
          }
          frenar('F',velMax,faseDeAcel);
    }
 }


/*
Stop slowly both motors
*/
 void frenar(char hacia, int velOriginal, int faseDeDesAcel)
```

```
{
    if (hacia == 'F'){
     // We were going forward before this
     for (velocidad=velOriginal;velocidad>0;velocidad=velocidad-faseDeDesAcel){
        analogWrite(motor2Pin2,velocidad);
        analogWrite(motor1Pin1,velocidad);
        delay(30);
     }
    }else if (hacia == 'B'){
     // We were going backwards before this
      for (velocidad=velOriginal;velocidad>0;velocidad=velocidad-faseDeDesAcel){
        analogWrite(motor1Pin2,velocidad);
        analogWrite(motor2Pin1,velocidad);
        delay(30);
     }
    }
    previousVel = 0;
}


  /*
    blinks an LED
   */
  void blink(int whatPin, int howManyTimes, int milliSecs) {
    int i = 0;
    for ( i = 0; i < howManyTimes; i++) {
      digitalWrite(whatPin, HIGH);
      delay(milliSecs/2);
      digitalWrite(whatPin, LOW);
      delay(milliSecs/2);
    }
  }


/*
Prender el led
*/
  void prender()
  {
    // Read the button, HIGH is pressed
    reading = digitalRead(switchPin);

//    Serial.print("Reading: ");
//    Serial.print(reading);
//    Serial.print(" / ");
//    Serial.print("Debounce if: ");
```

```
//    Serial.print(millis() - time);
//    Serial.print(' OO ');


   if (reading == HIGH) {
     Serial.println(millis() - time);

     // Time == 0 sometimes when the poweron and poweroff is used.
     if (millis() - time > 0 && millis() - time < 65) {
        // If you press the button twice quickly, set the ldr hide value,
        // to the value read just now in the LDR. It is to configure
        // under which amount of light the robot will function.
        // We also substract 50 so the robot does not start and
        // stop without knowing what to do if the light
        // is a little bit variable.
       ldrHideValue = analogRead(ldrInput1)-50;
       Serial.print("LdrHide set to : ");
       Serial.println(ldrHideValue);
       state = LOW;
       digitalWrite(ledPin, state);
       time = millis();
     }


        // reading == 1 means that we are going to do something
        // ONLY when the button comes from being pressed to
        // NOT being press. That is, when you release it!
     if (reading == HIGH && millis() - time > debounce) {
       if (state == HIGH) {
         state = LOW;
       }
       else {
         state = HIGH;
       }
       digitalWrite(ledPin, state);
     }

   time = millis();
   }
  }
```