RIC

Diseño y construcción de un prototipo robótico de un automóvil para transporte personal y económico con la tecnología Arduino

Design and construction of a robotic prototype of a car for the personal and economic technology with Arduino transport

Daniela Campo Jorge[1]; Miguel Sastre Vásquez[1]; Israel Gamas Faulkner[1] & Euclides Samaniego González[2*]

[1]Ingeniería de Sistemas y Computación – Facultad de Ingeniería de Sistemas Computacionales – Universidad Tecnológica de Panamá

[2]Grupo de Investigación en Inteligencia Computacional – GIICOM – Facultad de Ingeniería de Sistemas Computacionales – Universidad Tecnológica de Panamá

Resumen Este proyecto está basado en la necesidad de un sistema de transporte más eficiente y seguro; siendo capaz de evitar los errores humanos, a través de sensores, placas Arduino, motores y una codificación estructurada y exitosa. Se trata de utilizar dos Arduino, el máster, que trabaja en conjunto con los sensores, recopila la información que obtienen los sensores para enviársela al Arduino *slave*, el cual está a cargo de los motores del automóvil para realizar la acción que requiera, dependiendo de la información que recolecte.

Palabras claves Circuito, comunicación serial, robot, robótica, sensor, sistema, tecnología.

Abstract This project is based on the need for a system more efficient and safe transport, being able to avoid human error, through sensors, Arduino boards, motors and a structured and successful coding. It is using two Arduino, the master, which works in conjunction with sensors, collects the information obtained sensors to send to Arduino slave, which is in charge of the engines to perform the action required, depending on the information collected.

Keywords Circuit, serial communication, robot, robotics, sensor, system, technology.

*** Corresponding author:** euclides.samaniego@utp.ac.pa

1. Introducción

La intención de este proyecto es implementar la seguridad de las personas en la calle, como podemos ver en el día a día, las personas cometemos errores y malas decisiones, en cambio un robot, programado con reglas estructuradas y maneras de tomar decisiones ya hechas, no comete estos errores ni malas decisiones, y también reacciona de manera rápida y precisa.

Con esto en mente, se propone diseñar un prototipo de automóvil robótico que pueda evitar el error humano y las malas decisiones en la calle mientras se desplazan.

El automóvil robótico, no podrá saltarse una luz en rojo porque tiene prisa, o distraerse mientras está en movimiento y chocarse con un automóvil al frente por ejemplo, gracias a los sensores que contiene el prototipo y la programación adecuada para el manejo de la información del ambiente, el automóvil robótico será capaz de estar constantemente alerta y procesando lo que está pasando a su alrededor.

2. Aspectos generales del proyecto

A lo largo de los años se ha estado pensando en la forma de un automóvil inteligente. Estas dos palabras tienen un significado amplio, un automóvil que hiciera más cómodo y seguro el transporte de las personas, fácil de usar y amigable, con gran variedad de dispositivos y opciones.

Ya se encuentran bastantes investigaciones y algunos prototipos de automóviles inteligentes que son capaces de transportar gente de una manera segura, que son capaces de llevar personas a cualquier lugar, presionando solo botones.

2.1 Características del problema

Uno de los problemas que se ve diariamente en nuestra sociedad en el ámbito automovilístico es el de los accidentes de tráfico. Algunos accidentes son inevitables, pero otros son causados por errores humanos que se pudieron haber evitado.

Entre esos errores están: hablar por teléfono, conducir en estado de ebriedad o drogado, conducir distraído y no atento a la calle, desobedecer las señales de tránsito y semáforo, etc. Si las normas de conducción se cumplieran, se podrían evitar muchos accidentes y muertes.

2.2 Justificación

La aplicación más importante de los autómatas en la ingeniería automotriz es la de los automóviles autónomos. Estos automóviles tienen como meta final tener la capacidad de conducir por cuenta propia, sin necesidad de que el ser humano tenga que intervenir.

En etapas iniciales se espera que el ser humano intervenga en caso de emergencia. El hecho de que el ser humano no tenga que conducir le permitiría leer, dormir, trabajar o alguna otra acción mientras que el carro lo lleva a su destino. Se espera que los automóviles autónomos reduzcan la cantidad de accidentes.

El primer modelo de automóvil autónomo fue en 1939 en la feria Futurama patrocinada por General Motors el cual era un automóvil eléctrico por un circuito embebido en el pavimento (Vehículo Autónomo, 2015).

Los automóviles autómatas son comunes, estos incluyen radares, luces láser, sensores, GPS y cámaras para ubicarse y rodar por la ciudad.

Entre los proyectos líderes están el de Google (Google X) y AutonomosLab desarrollado por la Universidad Libre de Berlín.

El automóvil sin conductor de Google (Google *driverless car*) es un proyecto de Google consistente en el desarrollo de la tecnologías necesarias para crear automóviles sin conductor.

Este automóvil es capaz de conducir autónomamente tanto por la ciudad, como por carretera, detectando otros vehículos, señales de tráfico, peatones, entre otros (Automóvil sin conductor de Google, 2015).

Ha pasado por una serie de etapas comenzando por un Toyota Prius, un Lexus RX450 y el último prototipo que es un automóvil realizado desde cero para el proyecto.

Estos automóviles ya funcionan; sin embargo, por cuestiones legales y de precios, todavía no han salido al mercado. Además,

deben superar ciertos obstáculos, tales como peatones, ciclistas o calles en reparación. Con propuestas como estas, los automóviles autónomos dejan de ser ciencia-ficción y pronto serán una realidad.

2.3 Restricciones y limitaciones

- El prototipo es capaz de moverse.
- Es capaz de evitar obstáculos.
- Analiza colores y actúa dependiendo del color para semáforos (rojo para parar, verde para avanzar y amarillo para desacelerar), en señales de tráfico se asigna un color para las distintas señales.
- Se mantiene en el carril que está excepto que se quiera cambiar de carril.

2.4 Objetivos

2.4.1 Objetivo general

Diseñar y construir un prototipo robótico de un automóvil autónomo para transporte personal y económico.

2.4.2 Objetivos específicos

- Aplicar los fundamentos del área de lenguajes formales, autómatas y compiladores en el área de ingeniería automotriz.
- Aplicar conocimientos de robótica para la creación del automóvil.
- Diseñar el prototipo robótico de un automóvil autónomo para transporte personal y económico.
- Codificar el segmento de código para el prototipo robótico del automóvil autónomo para transporte personal y económico.
- Crear el diagrama del prototipo robótico de un automóvil autónomo para transporte personal y económico.
- Identificar los elementos que se requieren para que el prototipo robótico de un automóvil autónomo analice diferentes tipos de señales de luces y colores, y que actúe dependiendo del tipo de señal, creando un automóvil seguro y auto manejable, que pueda reaccionar ante las distintas señales en la carretera y debidamente a sus especificaciones.
- Utilizar electricidad como medio de energía para el movimiento del automóvil.
- Lograr que el prototipo logre evitar posibles accidentes.

3. Materiales de trabajo

3.1 *Software*

Arduino Programming Languange/Arduino Development Environment: Arduino es una plataforma electrónica de código abierto basada en el lenguaje Arduino Programing Language. Este lenguaje, a pesar de estar basado en lenguaje *Wiring*, es muy similar a C, por lo tanto no es mucho lo que hay que aprender si ya se tiene noción de C.

El entorno de programación es el *Arduino Development Environment*, la cual está basada en *Processing*.

3.2 *Hardware*

3.2.1 Arduino UNO

El Arduino UNO es una tarjeta controladora desarrollada en el 2005 en Italia, por un grupo de estudiantes del Instituto IVREA. Es muy utilizado en el área de robótica para controlar motores servo, también para instrumentos de medición y similares ya que es fácil agregarle y controlar sensores.

El microcontrolador más utilizado en los Arduino es el ATmega328.

El Arduino UNO posee todo lo que se necesita para manejar el controlador, simplemente se conecta a un computador por medio del cable USB o se puede alimentar utilizando una batería o un adaptador AC-DC. Si se conecta por USB, la alimentación externa no es necesaria.

3.2.2 Sensor de detección de color TCS3200

Es un sensor de frecuencia de luz de color programable, detecta colores e incluye cuatro LEDs blancos. El sensor TCS3200 puede detectar una cantidad casi ilimitada de colores

visibles. Las aplicaciones incluyen lectura de prueba, la clasificación por color, sensor de luz ambiental y de calibración, y la coincidencia de color, por nombrar solo algunos.

3.2.3 Cables de conexión (*Hook-up Wire*)

Alambres de enganche para conectar los diferentes dispositivos a la placa Arduino y entre sí. Para lograr la construcción del circuito.

3.2.4 *MotorShield*

Es una placa que permite controlar las direcciones y la velocidad del motor al usar un Arduino.

Para usar correctamente esta placa, hay que hacer uso de los pines incorporados a esta. Esta placa permite también darle poder al motor hasta de 12v.

3.2.5 LEDs de colores

Fotoceldas de colores que emiten luz.

3.2.6 Resistencias

Dispositivo que hace oposición o resistencia al paso de una corriente eléctrica.

3.2.7 HC-SR04

Es un sensor de rango ultrasónico que brinda medidas de objetos cercanos sin contacto. El sensor consta de un transmisor, un receptor y un controlador.

3.2.8 Motor *Smart Robot Car Chassis Kit*

Plataforma con ruedas y motores para Arduino.

4. Diseño del sistema prototipo de robot

4.1 Diseño funcional de la aplicación

Mediante los dispositivos que se muestran en la figura 1, que son los motores, los dos Arduino, el MotorShield, tres sensores de proximidad, un sensor de color, una batería y los cables uniendo todo el circuito.

Se hizo el prototipo real de un automóvil robótico, el cual fue capaz de desplazarse analizando el ambiente que lo rodea.

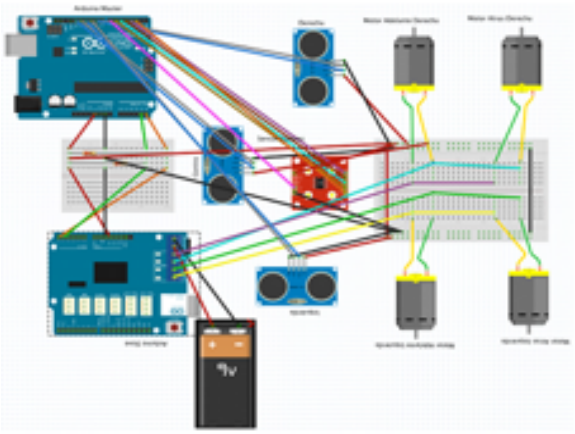


Figura 1. Diseño funcional de la aplicación.

Adicionalmente el prototipo analiza el color que se le coloque en la parte frontal.

Dependiendo del color que capte, reacciona de diferentes maneras.

Por ejemplo, con rojo, con verde avanza y con azul cambiará de velocidad (rápido a despacio o viceversa), y con los sensores de proximidad, el que se encontraba al frente activaba a los dos de los lados cuando encontraba un obstáculo para planear y realizar un giro hacia donde tenía espacio y de esta manera hacerlo.

4.2 Esquema conceptual del sistema

En la figura 2 se presenta el esquema conceptual del sistema desarrollado.

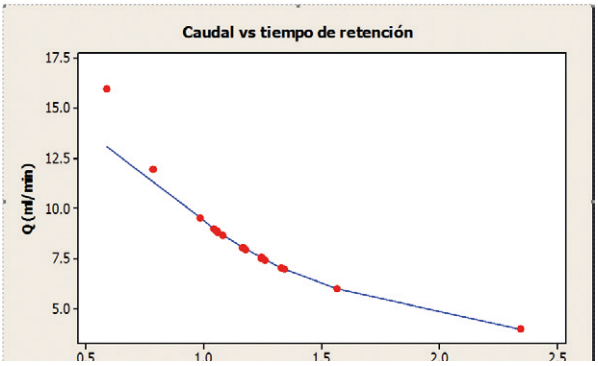


Figura 2. Esquema conceptual del sistema.

4.3 Diagrama esquemático

El diagrama que se muestra en la figura 3, es una representación del circuito del prototipo de automóvil robótico.

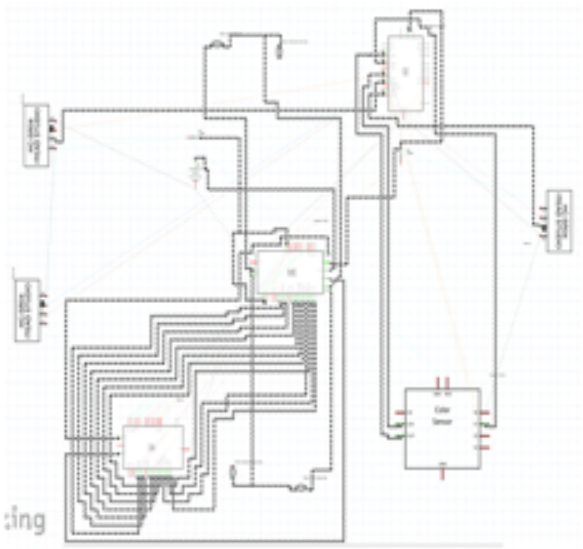


Figura 3. Diagrama esquemático del sistema.

En la figura se muestran todos los componentes conectados a los Arduino.

4.4 Ensamblado de los componentes del robot y diseño de actividades de interacción

En este punto se muestra los diseños de los componentes del automóvil robótico individualmente y se explicará el funcionamiento de cada uno de estos dispositivos.

Sensor de Proximidad. Se conecta el pin 4 del Arduino al pin *echo* del sensor de proximidad, y el pin 7 al disparador del sensor, y para cerrar el circuito, el *ground* o tierra del Arduino con la tierra del sensor, y el pin de voltaje del Arduino con el pin del voltaje del sensor. Ver figura 4.

Para el código, se utilizarán las funciones de *digitalWrite* (disparador, HIGH); para que emita un pulso de sonido de alta frecuencia, se pone un pequeño *delay* y se apaga el disparador, solo hace falta un pequeño pulso para que tome la distancia, así que después del *delay*, se apaga el disparador con digitalWrite (disparador, Low); una vez realizado esto, se le ordena al sensor de proximidad que tome el rebote del pulso que se le acaba de enviar con la siguiente función, duración = pulseIn (echo, High); dependiendo del tipo de medida que se desee, se transforma la duración, en nuestro caso, usamos centímetros, con lo cual se toma la siguiente función: distancia = duración/58.

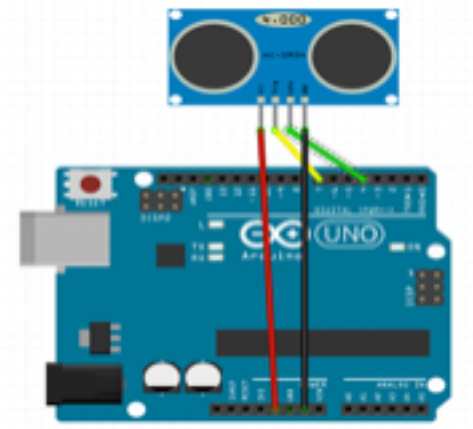


Figura 4. Sensor de proximidad.

En las primeras pruebas con un solo sensor de proximidad se quería que tomara la distancia del frente del automóvil robótico constantemente, y si la distancia es menor que 5cm, retrocedía hasta que la distancia entre el objeto y el automóvil fuera de 10cm, cuando cumplía con la condición de estar a 10cm de distancia de un objeto, el automóvil giraba y cambiaba de dirección.

Tres sensores de proximidad. Se utilizarán los pines 13, 11 y 9 para los disparadores del sensor del centro, izquierda y derecha respectivamente, 12, 10 y 8 para los *echos*, para cerrar el circuito, el *ground* o tierra del Arduino con el de tierra del sensor, y el pin de voltaje del Arduino con el pin del voltaje del sensor. (Ver figura 5).

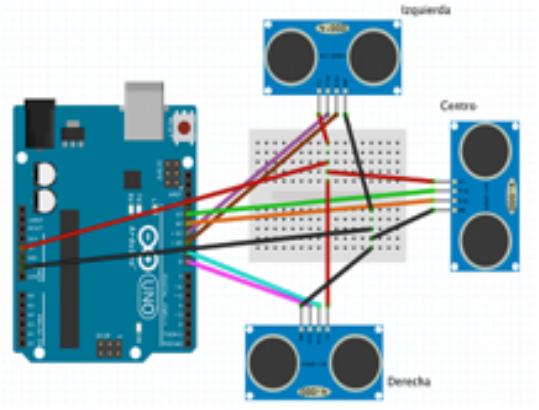


Figura 5. Tres sensores de proximidad.

Se utilizan las mismas funciones, *digitalWrite* (disparador, *High)*; para activar emisión del pulso de sonido, un *delay*, *digitalWrite* (disparador, *Low*); para parar la emisión, duración = *pulseIn* (*echo,High*); para cronometrar el tiempo que le tomó y distancia = duración/58; para convertirlo en distancia.

La diferencia, no es solo de adjuntar dos sensores de proximidad extra, si no que va a ver uno que active a los demás, el sensor del centro va a estar constantemente activo, emitiendo pulsos de sonido, y cuando encuentre un objeto a menos de 10cm, se manda la orden para encender los otros dos sensores que se encuentran a los lados del automóvil para que analicen si se encuentran objetos en alguno de los lados para poder realizar el giro del automóvil sin ningún problema y riesgo a choque.

Sensor de Color. Se utilizarán los pines 3 y 4 para la lectura de la salida de frecuencia del color que va a leer el sensor de color, o sea las salidas S0 y S1 respectivamente, los pines 5 y 6 para la distinción de colores del sensor de color, o sea las salidas S2 y S3 respectivamente, el pin 7 para los leds incorporados en el sensor de color, salida OE y el pin 2 para la salida de la lectura del sensor de color, o sea la salida OUT. Ver figura 6.

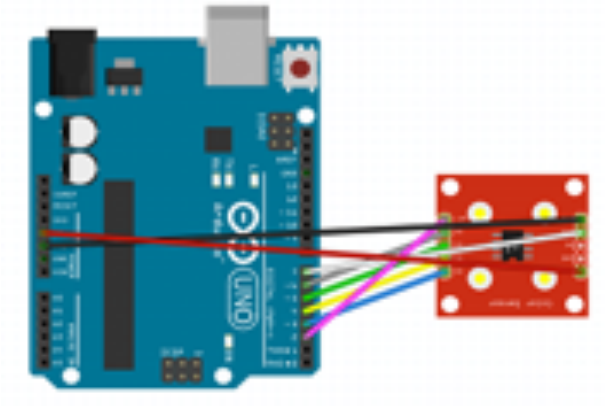


Figura 6. Sensor de color.

Se utiliza la función *digitalWrite* (7, *High*); para encender todos los *leds* incorporados del sensor, para determinar cuál es el color de una superficie se hacen combinaciones de los pines S2 y S3.

Blanco: Rojo:
digitalWrite(S2,HIGH); digitalWrite(S2,LOW);
digitalWrite(S3,LOW); digitalWrite(S3,LOW);
Verde: Azul:
digitalWrite(S2,HIGH); digitalWrite(S2,LOW);
digitalWrite(S3,HIGH); digitalWrite(S3,HIGH);

Motorshield. Esta tarjeta se conecta directamente encima del Arduino. Para controlar los motores se utilizan los pines 12 & 13 para ambos canales de motor. Se utilizan los pines 8 & 9 para ambos canales de frenos. Se utilizan los pines 3 & 11 para asignar las velocidades a cada motor. Ver figura 7.

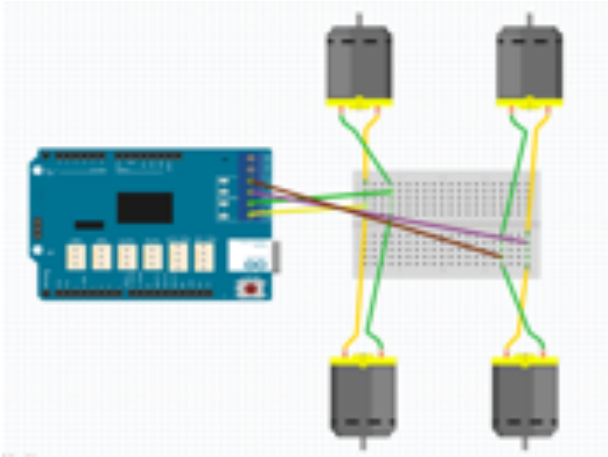


Figura 7. *Motorshield.*

Comunicación serial I2C: se utilizarán los pines analógicos A4 y A5 para la comunicación serial I2C en ambos Arduino.

El pin A4 es el pin de transferencia de datos, mientras que el pin A5 es la línea del reloj. Los pines A4 y A5 están conectados a los mismos pines A4 y A5 del otro Arduino.

También se conectan al voltaje 5V por medio de resistencias. El voltaje 5V y el *ground* son puntos en común para ambos Arduino y deben estar conectados. Ver figura 8.

5. Implementación del prototipo con la tecnología Arduino

Al comenzar el proyecto se tenía el Arduino con el *Motor Smart Robot Car Chassis Kit*, que es una plataforma con ruedas y motores para

poder movilizar el Arduino y sus componentes, todos los sensores que se van a utilizar, y un *motorshield* para adjuntar al Arduino para controlar los motores del *Motor Smart Robot Car Chassis Kit.*

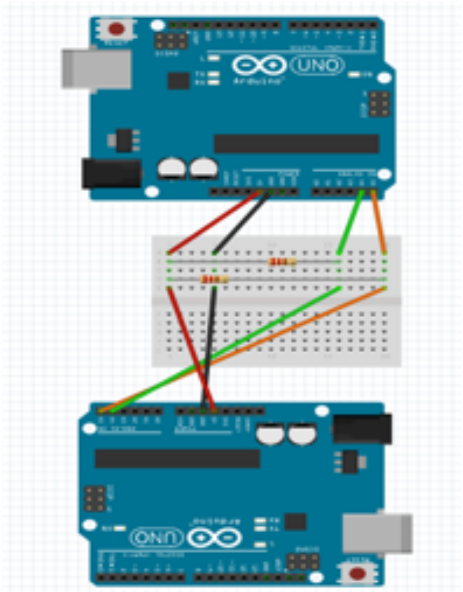


Figura 8. Comunicación serial 12C.

Se procedió a revisar los motores y a hacer pruebas para revisar que todo esté funcionando.

Control de Motores Primera Prueba

```
//Motor A Adelante
digitalWrite(12, HIGH); //Establece dirección del canal A
digitalWrite(9, LOW);   //Quita el freno del canal A
analogWrite(3, 255);    //Arranca el motor del canal A a
toda velocidad

//Motor B Adelante
digitalWrite(13, HIGH); // Establece dirección del canal B
digitalWrite(8, LOW);   // Quita el freno del canal B
analogWrite(11, 255);   // Arranca el motor del canal B a
toda velocidad

//Motor A Atrás
digitalWrite(12, LOW);  //Establece dirección del canal A
digitalWrite(9, LOW);   //Quita el freno del canal A
analogWrite(3, 255);    //Arranca el motor del canal A a
toda velocidad

//Motor B Atrás
digitalWrite(13, LOW);  // Establece dirección del canal B
digitalWrite(8, LOW);   // Quita el freno del canal B
analogWrite(11, 255);   // Arranca el motor del canal B a
toda velocidad
```

Más adelante se trabajó con un sensor de proximidad por separado, probando que todo esté correcto antes de integrarlo al Arduino con el motor. El sensor de proximidad emite un pulso de sonido de alta frecuencia y cronometra el tiempo en que éste llega al objeto, rebota y vuelve al sensor, y con esto logramos hacer que el automóvil robótico evite chocarse contra objetos.

Sensor de Proximidad Primera Prueba

```
digitalWrite(trigger_frente, LOW); //se
declaran las respectivas salidas del sensor
ultrasónico
delayMicroseconds(2); //y se almacenan en la
variable del sensor que convierte la
digitalWrite(trigger_frente, HIGH); //
velocidad del sonido que es de 340 m/s o 29
microsegundos
delayMicroseconds(5); //por centímetro, como
la señal va y viene, ese tiempo es el de la
digitalWrite(trigger_frente, LOW); // mitad,
siendo sensor= tempo/29/2
durac_frente = pulseIn (echo_frente, HIGH);
frente = duracion_frente/29/2;
```

Una vez que todo funcione correctamente, se integra al *motorshield* con los motores y se realiza un código para que trabajen en conjunto. Se le agregan unos cambios a las funciones para hacer que si se detecta un obstáculo a una distancia menor que cinco centímetros, el automóvil robótico retrocede hasta estar a una distancia segura y verifica la distancia nuevamente, esto sirve por si algún objeto se le atraviesa este reacciona, y si encuentra un obstáculo menor o igual a quince centímetros, este gira hacia la derecha o izquierda, dependiendo de un valor aleatorio generado por código.

Sensor de Proximidad Segunda Prueba

if(distancia<5) //si la distancia entre el automóvil robótico y el obstáculo es menor que

```
5
 {
  do
  {
       backwards(); //el automóvil robótico retrocede
    distancia=calcular(); //calcula la distancia nuevamente y verifica
   }while(distancia<10);
 }
 if (distancia <=15) { //si la distancia entre el automóvil robótico y el obstáculo es menor o igual a 15
   rand=random(0,1000); //genera un numero aleatorio
    if (rand%2==0)    //si el número es par gira a la derecha, si es impar gira a la izquierda
    {
     turnright();
    }
    else
    {
     turnleft();
    }
  }
 else forward();
```

Para la siguiente fase, se trabajó con foto receptores, con la intención de detectar el ambiente y los colores que se le presentan, para poder identificarlos y actuar ante ellos.

El fotorreceptor es capaz de convertir la energía óptica de la luz que está sobre una superficie en energía eléctrica mediante la transducción (transformación de un tipo de señal o energía en otra de distinta naturaleza).

En la naturaleza, los fotorreceptores son células fotosensibles del sistema visual de los seres vivos, pero también hay fotorreceptores electrónicos, que son componentes electrónicos que detectan la luz.

Para hacer el proceso más acertado y fluido, colocamos leds de colores verde, rojo, azul y amarillo, para que al fotorreceptor le sea más fácil distinguir la energía que emite cada luz de color.

Los *leds* se encenderán uno por uno mientras la fotoreceptora realiza lecturas de cada uno para ver cuál es el color que se hace más fuerte contra el objeto.

Fotorreceptor Primera Prueba

```
double ambient = analogRead(photoResistor);
Serial.print("Ambient: ");
Serial.println(ambient*ambientAdj);

digitalWrite(g, HIGH);
delay(saturationTime);
double green = analogRead(photoResistor);
Serial.print("Green: ");
Serial.println(green*gAdj);
digitalWrite(g, LOW);
delay(colldownTime);

digitalWrite(r, HIGH);
delay(saturationTime);
double red = analogRead(photoResistor);
Serial.print("Red: ");
Serial.println(red*rAdj);
digitalWrite(r, LOW);
delay(colldownTime);

digitalWrite(b, HIGH);
delay(saturationTime);
double blue = analogRead(photoResistor);
Serial.print("Blue: ");
Serial.println(blue*bAdj);
digitalWrite(b, LOW);
delay(colldownTime);

digitalWrite(y, HIGH);
delay(saturationTime);
double yellow = analogRead(photoResistor);
Serial.print("Yellow: ");
Serial.println(yellow*yAdj);
digitalWrite(y, LOW);
delay(colldownTime);

double rawData[] = {(green*gAdj), (red*rAdj), (blue*bAdj), (yellow*yAdj)};
  double maximum = ambient*ambientAdj;
  int decision;
```

Ya que el fotorreceptor utilizado no es muy preciso a la hora de detectar la energía de la luz, se procede a utilizar un distinto sensor de color, el *ColorPal.*

El *ColorPal*, detecta los colores de una superficie. El sensor posee tres luces de colores internas, el rojo, verde y azul, y las enciende alternadamente haciendo brillar la superficie que va a ser analizada, para detectar la cantidad de iluminación que va a ser reflejada para cada una de esas luces de colores. (McComb, 2013)

```
// Reset al ColorPA
void reset() {
  delay(200);
  pinMode(sio, OUTPUT);
  digitalWrite(sio, LOW);
  pinMode(sio, INPUT);
  while (digitalRead(sio) != HIGH);
  pinMode(sio, OUTPUT);
  digitalWrite(sio, LOW);
  delay(80);
  pinMode(sio, INPUT);
  delay(waitDelay);
}

void readData() {
  char buffer[32];
    if (serin.available() > 0) {
    buffer[0] = serin.read();
    if (buffer[0] == '$') {
      for(int i = 0; i < 9; i++) {
        while (serin.available() == 0);
        buffer[i] = serin.read();
        if (buffer[i] == '$')
          return;
      }
      parseAndPrint(buffer);
      delay(1000);
    }
  }
}

void parseAndPrint(char * data) {
  sscanf (data, "%3x%3x%3x", &red, &grn,
&blu);
  char buffer[32];
  sprintf(buffer, "R%4.4d G%4.4d B%4.4d",
red, grn, blu);
  Serial.println(buffer);
}
```

Ya que el Arduino posee un número limitado de pines, y con todos los sensores, el motor y los *leds* no hay espacio suficiente, se propone hacer un estudio para realizar una comunicación serial entre Arduino, se utilizarán dos Arduino, trabajando en conjunto como un sistema para poder trabajar con todos los dispositivos a la vez antes de utilizarlo el método.

Existen varias opciones para hacer que dos Arduinos se comuniquen, tales como la comunicación UART (requiere de los pines digitales 0 y 1, que se utilizan en la comunicación entre Arduino y computadora), el DIGITAL I/0 (que tiene comandos y dispositivos limitados) y la seleccionada, la comunicación I2C (Inter-Integrated Circuit). Esta permite conectar más de 100 Arduinos esclavos de manera sencilla a un Arduino maestro.

El I2C es un protocolo serial síncrono, con un rango de velocidades desde 10-100 Kb/s, 128 posibles direcciones, de las cuales 16 están reservadas (la dirección 0 está reservada para el Arduino maestro) y 112 son asignables a los dispositivos esclavos.

Para utilizar I2C, se requiere utilizar un SDA (*Serial Data Line*) y un SCL (*Serial CLock Line*), además de que se debe compartir el pin de 5V y GND (I2C Between Arduino, s.f.).

A pesar de que el protocolo I2C permite, en otros contextos, múltiples maestros y múltiples esclavos, en Arduino solo se utiliza un maestro y múltiples esclavos. (I2C, 2015).

A nivel de programación se utiliza la librería Wire (se debe incluir) para comunicar a los Arduinos. El maestro puede enviar y/o solicitar información de los Arduinos esclavos.

Para la aplicación de este proyecto, el Arduino donde están conectados los sensores será el Arduino maestro, el cual enviará la información al Arduino esclavo (donde están

conectados los motores). (Master Writer / Slave Receiver, s.f.)

En los Arduinos UNO el SDA está en el pin analógico 4, mientras que el SCL está en el pin analógico 5. Estos pines también deben estar conectados a la resistencia *pull-up* (los Arduinos más nuevos tienen esta función incluida de fábrica).

El Arduino maestro siempre iniciará la comunicación en él mismo y el Arduino esclavo con el cual desea comunicarse. Direcciones deben asignarse a los Arduino para identificarlos.

5.1 Comunicación Arduino *Master-Slave* Primera Prueba

Ejemplo utilizado para probar la comunicación entre Arduinos.

```
//Slave
#include <Wire.h>
void setup()
{
 Wire.begin(5);
 Wire.onReceive(receiveEvent);
 pinMode(13,OUTPUT);
 digitalWrite(13,LOW);
}
void receiveEvent(int howMany)
{
 while(Wire.available())
 {
  char c = Wire.read();
    if(c == 'H')
  {
   digitalWrite(13,HIGH);
  }
  else if(c == 'L')
  {
   digitalWrite(13,LOW);
  }
 }
}
```

```
//Master
#include <Wire.h>
void setup()
{
 Serial.begin(9600);
 Wire.begin();
}
void loop()
{
 while(Serial.available())
 {
  char c = Serial.read();
  if(c == 'H')
  {
Wire.beginTransmission(5);
   Wire.write('H');
  Wire.endTransmission();
  }
  else if(c == 'L')
  {  Wire.beginTransmission(5);
   Wire.write('L');
  Wire.endTransmission();
  }}}
```

Antes de armar el automóvil robótico con todos los dispositivos, se implementan todas las partes por separado, se agregarán dos sensores de proximidad más y se hará que los tres trabajen en conjunto, uno al frente y los otros dos a cada lado del automóvil, haciéndolo más independiente y seguro en curvas u otros obstáculos que puedan presentarse en los lados.

El sensor del frente utiliza los pines 4 y 5 para *trigger* y *echo* (emite el impulso, recibe el impulso) respectivamente, el sensor de la izquierda usa los pines 2 y 3, y el de la derecha el 6 y el 7.

Sensores de Proximidad Primera Prueba

```
void loop() {
  // declaración de variables
     long durac_frente, durac_izq, durac_der,
der, izq, frente;
     digitalWrite(trigger_frente, LOW);  // se
declaran las salidas y entradas del sensor
    delayMicroseconds(2);
    digitalWrite(trigger_frente, HIGH);
    delayMicroseconds(5);
    digitalWrite(trigger_frente, LOW);
        durac_frente = pulseIn(echo_frente,
HIGH);
    frente = durac_frente/29/2;
   Serial.print("frente:");
    Serial.println(frente);
    digitalWrite(trigger_izq, LOW);
    delayMicroseconds(2);
    digitalWrite(trigger_esq, HIGH);
    delayMicroseconds(5);
    digitalWrite(trigger_izq, LOW);
    durac_izq = pulseIn(echo_izq, HIGH);
    izq = durac_izq/29/2;
   Serial.print("izquierda:");
    Serial.println(izq);
    digitalWrite(trigger_der, LOW);
    delayMicroseconds(2);
    digitalWrite(trigger_der, HIGH);
    delayMicroseconds(5);
    digitalWrite(trigger_der, LOW);
    durac_dir = pulseIn(echo_der, HIGH);
    der = durac_der/29/2;
     Serial.print("Derecha:");
    Serial.println(der);
   delay(500);
   }
```

Por problemas de obtención de colores del *ColorPal,* no era muy preciso, no diferenciaba los colores mucho, cambiamos a un sensor de color TS3200, más confiable con leds integrados para su calibración.

El TSC3200 es un convertidor de luz a frecuencia programable, puede filtrar los datos RGB de la fuente de luz y convertirlo en una onda cuadrada on frecuencia directamente proporcional a la intensidad de luz (irradiación) (Sensor de Color TCS3200, s.f.).

GND es la tierra del circuito, VDD es para administrar el voltaje necesario, OUT es el pin para la frecuencia de salida del sensor, S0 y S1 son los pines para la frecuencia de escala de entrada, permite leer colores desde diferentes distancias, y S2 y S3 son los que, en conjunto, determinan el color (Programmable Color Light-to-Frequency Converter, 2011).

Se colocará el sensor mirando hacia el suelo desde el automóvil, con lo cual S0 y S1 no son necesarios, solamente se trabajará con S2 y S3 para detectar el color que se está presentando.

Sensor de Color TCS3200 Primera Prueba

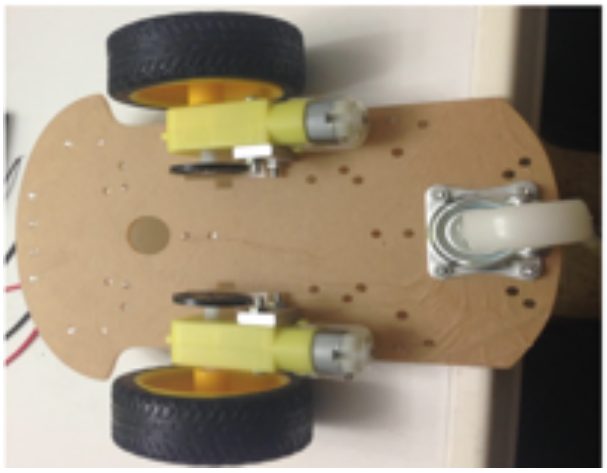
```
void asses(int colldownTime, int saturationTime)
{
if (s2==LOW && s3== LOW)
Serial.println("Red");
else if (s2==LOW && s3== HIGH)
Serial.println("Blue");
else if (s2==HIGH && s3== HIGH)
Serial.println("Green");
else if (s2==HIGH && s3== LOW)
Serial.println("Undefined");}
```

Una vez que todo funcione correctamente, se harán pruebas de los sensores de distancia y el sensor de color trabajando a la vez en un solo Arduino, que futuramente será el Arduino master, que será el que maneje toda la información y la organice para mandársela al Arduino esclavo que tendrá el motor, y solo controlará los movimientos del motor dependiendo de lo que el master le envíe.

Por ejemplo, si el maestro detectó el color rojo a través del sensor de color, mandará la lectura al Arduino esclavo y este hará que los motores se paren, y si lee verde, lo manda y los motores arrancarán otra vez.

Se creó un código prueba para comprobar el manejo de todos los sensores en un solo Arduino para comprobar que todo esté correcto y en funcionamiento antes de formar la comunicación con otro Arduino.

En las figuras 9 y 10 a continuación se presentan imágenes de la evolución del prototipo desde la primera versión y fase 2, con sus diferentes vistas.



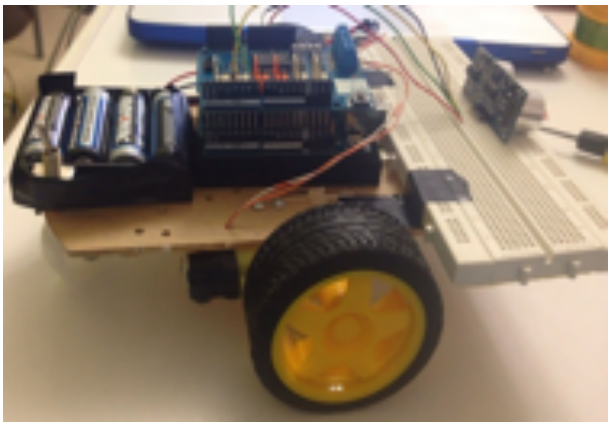


Figura 9. Evolución del prototipo (primera versión) y con sensor de proximidad (fase 1).

En las imágenes anteriores se muestra el nuevo chasis, el cual es más amplio para contener todos los dispositivos necesarios para producir el automóvil robótico. Ver figura 11.

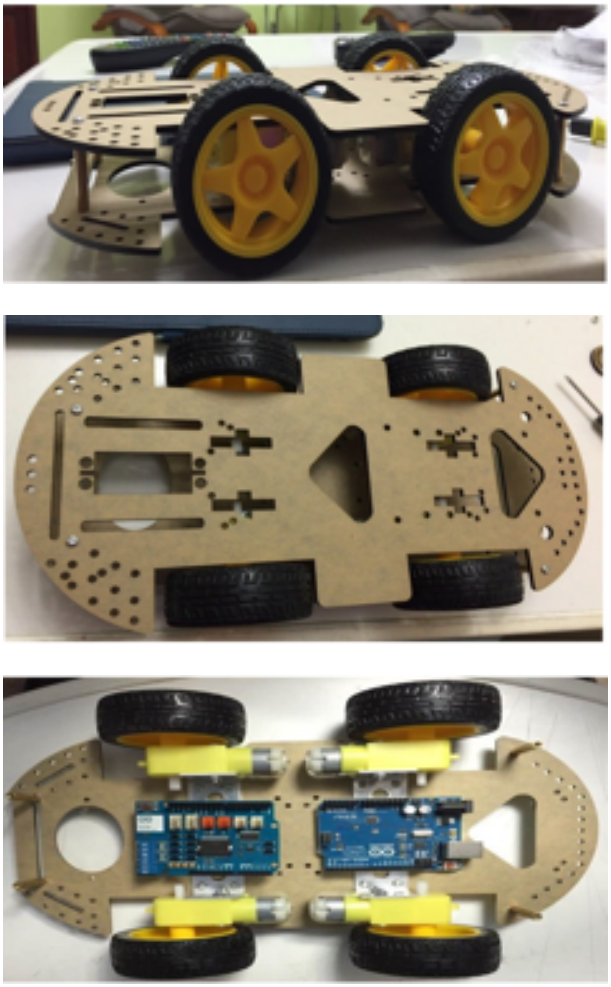


Figura 10. Evolución del prototipo (fase 2); vista superior y vista interna respectivamente.

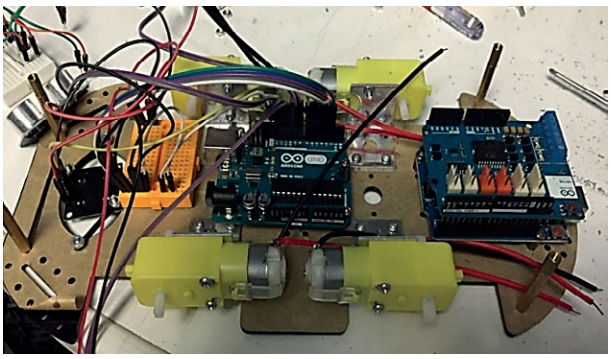


Figura 11. Evolución del prototipo (fase 2 – implementación 1).

En la imagen de la figura 12 se muestra el chasis con los dos Arduinos a utilizar, el *motorshield,* el sensor de color y el cableado necesario para su funcionamiento.

Figura 12. Chasis.

En la imagen de la figura 13 se muestra la preparación del posicionamiento del contenido del automóvil robótico en el chasis, todos los sensores, cableado, motores, Arduino y *motorshield.*

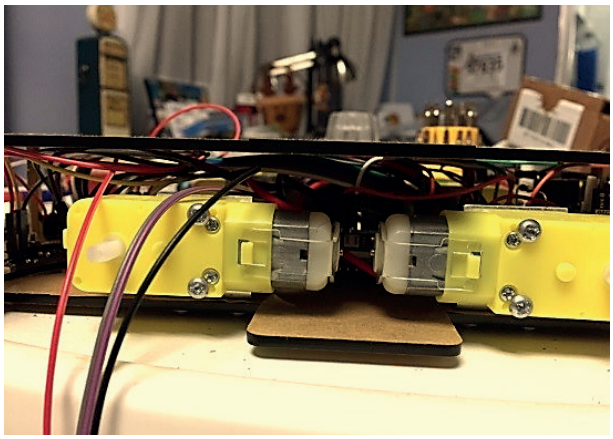


Figura 13. Evolución del prototipo (fase 2 – vista interna lateral).

En la imagen mostrada, se muestra el chasis completo, una vez ya colocado todo el interior, que ya cuenta con el sensor de color, los sensores de proximidad, los dos Arduino, el *motorshield,* todo el cableado y los motores.

Una vez que todo está ordenado y listo para programar, se realiza un código, que ya va a ser el de la fase final, para utilizar dos Arduino, uno con los sensores y otro con los motores, transfiriéndose información a través de la comunicación serial seleccionada anteriormente.

En la figura 14 se muestran los sensores ultrasónicos para la detección de distancia de frente y de lado y se muestran las pruebas realizadas con el sensor de color, ubicado en la parte inferior del automóvil inteligente.

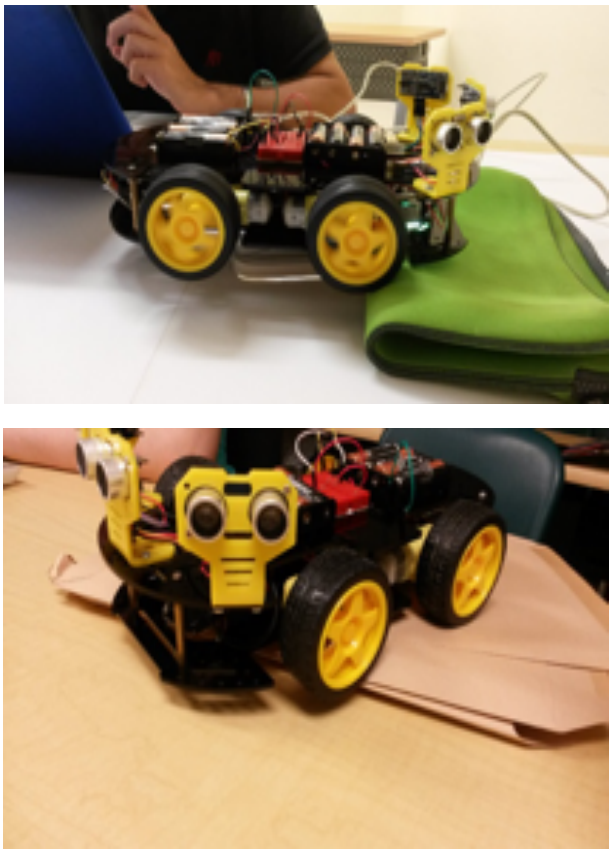


Figura 14. Pruebas de frente y de lado y de sensor de color.

6. Resultados y discusiones

Después de un largo período de pruebas de código y de uso de componentes, se logró completar con éxito el objetivo del proyecto, se logró el funcionamiento deseado.

Se requiere de muchas pruebas para llegar a la funcionalidad deseada, en estas pruebas se deben revisar los componentes que se van a utilizar para ver si son los más adecuados.

En este proyecto se empezó usando foto receptores antes que el dispositivo TCS3200, lo cual representó una gran mejora para el proyecto. Se cambió el chasis a otro más grande para que fuera más fácil la manipulación de todos los componentes del proyecto.

7. Conclusiones

- El protocolo I2C utiliza la librería Wire para programación, permite solamente un Arduino maestro, pero puede utilizar hasta 100 Arduinos esclavos. Además, solamente utiliza los pines analógicos 4 y 5 para comunicación, y los puertos de voltaje y *ground* deben estar en común.
- Los sensores ultrasónicos son sensibles al voltaje aplicado. Además, pueden tener puntos ciegos, ya sea que el objeto esté ligeramente fuera del rango de lectura, o si el objeto está pegado al sensor ultrasónico.
- El peso del prototipo y la corriente que utilizan los motores pueden causar fluctuaciones en el funcionamiento de los sensores ultrasónicos.

REFERENCIAS

[1] Samaniego González, E. (2011). La Robótica con Arduino. In E. Samaniego González, Ingeniería de Sistemas Robóticos: Aplicaciones sobre Arduinos (pág. 14). Panamá, Panamá, Panamá: L&J Publicaciones. Recuperado el 18 de Septiember de 2014.

[2] Sensor de detección de color TCS3200. Wikipedia http://www.dfrobot.com/wiki/index.php/TCS3200_Color_Sensor_%28SKU:SEN0101%29

[3] HC-SR04 Sensor ultrasónico http://www.micropik.com/PDF/HCSR04.pdf

[4] Grinberg, M. (2012). Building an Arduino Robot, Part I: Hardware Components. Recuperado el 18 de Septiembre de 2014, de Miguelgrinberg.com: http://blog.miguelgrinberg.com/post/building-an-arduino-robot-part-i-hardware-components

[5] Using an RGB LED to Detect Colours. (s.f.). Recuperado el 18 de Septiembre de 2014, de Instructables: http://www.instructables.com/id/Using-an-RGB-LED-to-Detect-Colours/?ALLSTEPS

[6] Accidentes de tránsito en la República y ciudades de Panamá y Colón, por clase, según mes: año 2011. (2012). Recuperado el 17 de Septiembre de 2014, de Instituto Nacional de Estadísticas y Censos: http://www.contraloria.gob.pa/inec/archivos/P4361451-02.pdf

[7] Accidentes de tránsito, heridos y muertos en la República, por provincia y comarca indígena, según mes: año 2011. (2012). Recuperado el 17 de Septiembre de 2014, de Instituto Nacional de Estadística y Censo: http://www.contraloria.gob.pa/inec/archivos/P4361451-13.pdf

[8] Arduino. (s.f.). Recuperado el 18 de Septiembre de 2014, de Arduino Uno: http://arduino.cc/en/Main/ArduinoBoardUno

[9] Arduino. (2015). Recuperado el 10 de Julio de 2015, de Wikipedia: https://es.wikipedia.org/wiki/Arduino#Lenguaje_de_programaci.C3.B3n_Arduino

[10] Arduino Motor Shield. (s.f.). Recuperado el 18 de Septiembre de 2014, de Arduino: http://arduino.cc/en/Main/ArduinoMotorShieldR3

[11] Automóvil sin conductor de Google. (2015). Recuperado el 10 de Julio de 2015, de Wikipedia: https://es.wikipedia.org/wiki/Autom%C3%B3vil_sin_conductor_de_Google

[12] Características Técnicas del Arduino UNO. (s.f.). Recuperado el 10 de Julio de 2015, de http://www3.gobiernodecanarias.org/medusa/ecoblog/ralvgon/files/2013/05/Caracter%C3%ADsticas-Arduino.pdf

[13] Definición de Robótica. (s.f.). Recuperado el 5 de Mayo de 2015, de Definición.De: http://definicion.de/robotica/)

[14] Fernández, A. (2 de Junio de 2014). A bordo del coche que conduce solo de Volvo. Obtenido de Autopista.es: http://www.autopista.es/reportajes/articulo/volvo-conduccion-autonoma-drive-me-100746

[15] Grinberg, M. (2012). Building an Arduino Robot, Part I: Hardware Components. Recuperado el 18 de Septiembre de 2014, de Miguelgrinberg.com: http://blog.miguelgrinberg.com/post/building-an-arduino-robot-part-i-hardware-components

[16] Historia de la Robótica. (s.f.). Recuperado el 5 de Mayo de 2015, de Historia de la Robótica: https://roboticstoday.wikispaces.com/Historia+de+la+Rob%C3%B3tica

[17] I2C. (2015). Recuperado el 25 de Marzo de 2015, de Wikipedia: http://en.wikipedia.org/wiki/I%C2%B2C

[18] I2C Between Arduinos. (s.f.). Recuperado el 25 de Marzo de 2015, de Instructables: http://www.instructables.com/id/I2C-between-Arduinos/

[19] Los robots avanzan sobre la economía mundial. (2013). Recuperado el 5 de Mayo de 2015, de BBC Mundo: http://www.bbc.co.uk/mundo/noticias/2013/02/130222_robots_avanzan_sobre_economia_mundial_mj

[20] Martínez Ramos, S. B. (2014). Evolución de Arduino. Recuperado el 10 de Julio de 2015, de http://es.slideshare.net/witwicky/tipos-de-arduino-y-sus-caracteristicas

[21] Martínez Ramos, S. B. (2014). Evolución de Arduino. Recuperado el 27 de Junio de 2015, de Slideshare: http://es.slideshare.net/witwicky/tipos-de-arduino-y-sus-caracteristicas

[22] Master Writer / Slave Receiver. (s.f.). Recuperado el 25 de Marzo de 2015, de Arduino: http://arduino.cc/en/Tutorial/MasterWriter

[23] McComb, G. (2013). ColorPal Arduino Demo. Recuperado el 10 de Julio de 2015, de Learn.Paralax.Com.

[24] Programmable Color Light-to-Frequency Converter. (2011). Recuperado el 18 de Febrero de 2015, de http://www.mouser.com/catalog/specsheets/TCS3200-E11.pdf

[25] Samaniego González, E. (2011). La Robótica con Arduino. En E. Samaniego González, Ingeniería de Sistemas Robóticos: Aplicaciones sobre Arduinos (pág. 14). Panamá, Panamá, Panamá: L&J Publicaciones. Recuperado el 18 de September de 2014.

[26] Sensor de Color TCS3200. (s.f.). Recuperado el 10 de Julio de 2015, de blog.patagoniatecnology.com: http://saber.patagoniatecnology.com/sensor-de-color-tcs230-arduino-argentina-ptec-elecfreaks/

[27] TCS3200, TCS3210 PROGRAMMABLE COLOR LIGHT-TO-FREQUENCY CONVERTER. (Agosto de 2011). Recuperado el Febrero de 2015, de TAOS.

[28] Using an RGB LED to Detect Colours. (s.f.). Recuperado el 18 de Septiembre de 2014, de Instructables: http://www.instructables.com/id/Using-an-RGB-LED-to-Detect-Colours/?ALLSTEPS

[29] Vehículo Autónomo. (2015). Recuperado el 10 de Julio de 2015, de Wikipedia: https://es.wikipedia.org/wiki/Veh%C3%ADculo_aut%C3%B3nomo