

INFORME PROYECTO FINAL

YOJHAN LEONARDO RODRIGUEZ ASCENCIO

20131078023



MSC. NORBERTO NOVOA TORRES

ARQUITECTURA DE COMPUTADORES

UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD TECNOLÓGICA

BOGOTÁ D.C

2016

Introducción

La electrónica digital es una rama de electrónica en la cual se estudia o se aplica solo dos estados de valores, magnitudes o tensiones: alto-bajo, cero-uno. En la representación digital los valores no se denotan por valores proporcionales, sino por símbolos llamados dígitos. Cuando se manejan diversos valores es importante que podamos representar sus cantidades o magnitud con eficiencia y exactitud. Existen básicamente dos maneras de representar el valor numérico de las cantidades: la analógica y la digital. La diferencia principal entre los sistemas analógicos y digitales es que el sistema analógico es continuo y el digital es discreto es decir paso a paso.

Los sistemas digitales son una combinación de dispositivos diseñada para manejar cantidades físicas o información que están representadas de forma digital; esto es, que solo pueden tomar valores discretos. Estos dispositivos pueden ser magnéticos, neumáticos, mecánicos o electrónicos. Algunos de los sistemas digitales más conocidos son los relojes digitales, las computadoras, las calculadoras digitales y los controladores de señales del tráfico. Los sistemas analógicos contienen dispositivos que manejan cantidades físicas representadas de forma analógica, es decir que las cantidades varían entre un rango continuo de valores.

Con respecto a los beneficios de la electrónica digital se pretende desarrollar un sistema de luces LED (Diodo emisor de luz) sincronizado con una alarma programada cada hora para indicar el cambio de tiempo, un alarma a través de un buzzer para indicar este cambio y la implementación de un sensor de movimiento para que igualmente cambie su estado los diodos emisores de luz. Todo esto se le adiciona la importante implementación de una aplicación de escritorio desarrollada en Netbeans en el lenguaje de programación Java el cual se conecta con el sistema controlador libre de hardware Arduino y pretende darle al usuario la opción de

cambia la luz a cualquier color posible en RGB (Red-Green-Blue). Por último este proyecto se contendrá en una base de acrílico emisora de luz con el logo llamativo del lenguaje de programación JavaScript.

Objetivos

General

- ✓ Diseñar, construir y desarrollar un sistema de control de diodos de luces (LED) que sea controlado y administrado por medio de un software y que sea igualmente automático.

Específicos:

- ✓ Conocer e interactuar con los diferentes dispositivos de hardware de interacción del PC.
- ✓ Verificar e implementar 4 estados posibles al sistema controlador de LED.
- ✓ Controlar y comunicar el hardware encargado del sistema utilizando los diferentes puertos y conexiones del PC.
- ✓ Controlar el hardware electrónico del mediante un lenguaje de programación.

Materiales:

- Arduino UNO
- LED RGB 1mt
- 170mm M/M Jumper Wires x 10
- 170mm F/M Jumper Wires x 10
- Buzzer 5v
- Transistores TIP31 x 3
- Cable de UTP
- Fuente de poder 12 v

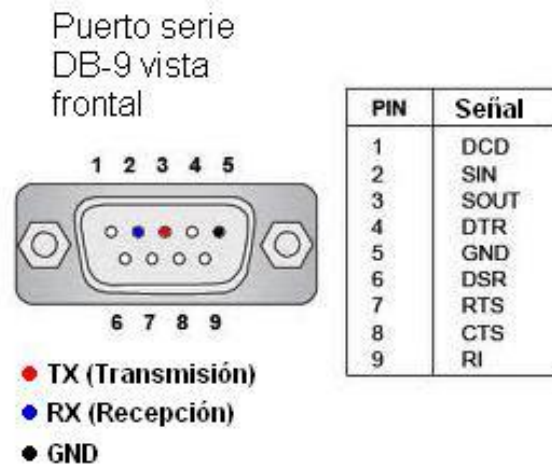
- Protoboard
- Resistencias de 1K ohm x 3
- Resistencia de 100 ohm x 1
- Sensor de movimiento PIR HC 05
- Un octavo de material acrílico
- Un pliego de cartón
- Bisturí

Marco Teórico

Puerto COM

Permite el envío de datos, uno detrás de otro, mientras que un paralelo se dedica a enviar los datos de manera simultánea. La sigla COM es debido al término ("COMmunications"), que traducido significa comunicaciones. Es un conector semitrapezoidal de 9 terminales, que permite la transmisión de datos desde un dispositivo externo (periférico), hacia la computadora; por ello es denominado puerto.





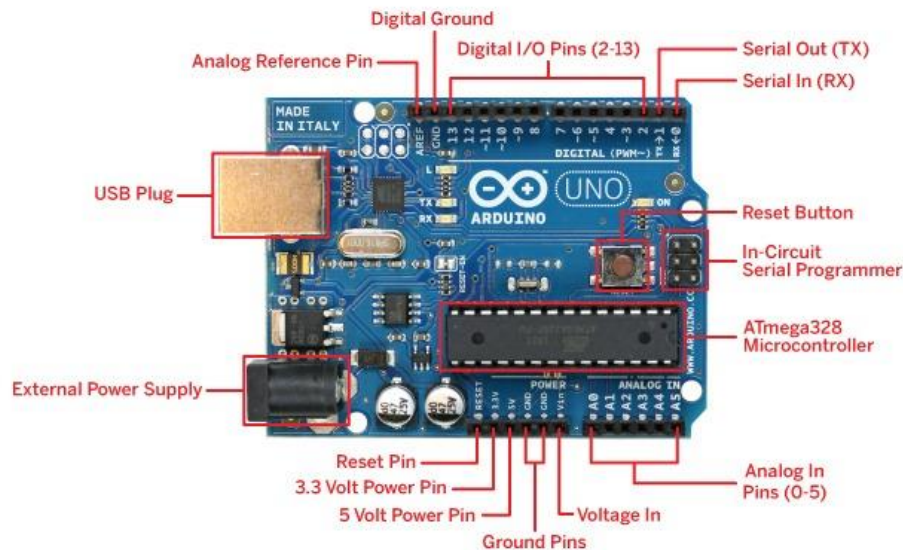
Puerto USB

Universal Serial Bus más conocido por la sigla USB, es un bus estándar industrial que define los cables, conectores y protocolos usados en un bus para conectar, comunicar y proveer de alimentación eléctrica entre computadoras, periféricos y dispositivos electrónicos. Un puerto USB6 7 8 permite conectar hasta 127 dispositivos y ya es un estándar en los ordenadores de última generación, que incluyen al menos cuatro puertos USB 3.0 en los más modernos, y algún USB 1.1 en los más anticuados. Es totalmente Plug and play, es decir, con sólo conectar el dispositivo (con el ordenador ya encendido), el dispositivo es reconocido e instalado de manera inmediata.

Arduino

Arduino 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 multidisciplinarios, Arduino se inició en el año 2005 como un proyecto para estudiantes en el Instituto IVREA, en Ivrea (Italia). En ese tiempo,

los estudiantes usaban el microcontrolador BASIC Stamp, cuyo coste era de 100 dólares estadounidenses, lo que se consideraba demasiado costoso para ellos. Por aquella época, uno de los fundadores de Arduino, Massimo Banzi, daba clases en Ivrea.



RGB Led Strip

Los diodos son componentes electrónicos que permiten el paso de la corriente en un solo sentido, en sentido contrario no deja pasar la corriente (como si fuera un interruptor abierto). Un diodo Led es un diodo que además de permitir el paso de la corriente solo un sentido, en el sentido en el que la corriente pasa por el diodo, este emite luz.

Cuando se conecta un diodo en el sentido que permite el paso de la corriente se dice que está polarizado directamente. Ahora si la definición correcta será: Un diodo Led es un diodo que cuando está polarizado directamente emite luz. Además la palabra LED viene del inglés Light Emitting Diode que traducido al español es Diodo Emisor de Luz. Este es el símbolo que se usa para los diodos led en los esquemas eléctricos, donde el ánodo será la patilla larga. Debido a su capacidad de operación

a altas frecuencias, son también útiles en tecnologías avanzadas de comunicaciones y control. Los ledes infrarrojos también se usan en unidades de control remoto de muchos productos comerciales incluyendo equipos de audio y video.

Resistencia

Resistencia eléctrica es toda oposición que encuentra la corriente a su paso por un circuito eléctrico cerrado, atenuando o frenando el libre flujo de circulación de las cargas eléctricas o electrones. Cualquier dispositivo o consumidor conectado a un circuito eléctrico representa en sí una carga, resistencia u obstáculo para la circulación de la corriente eléctrica.

Normalmente los electrones tratan de circular por el circuito eléctrico de una forma más o menos organizada, de acuerdo con la resistencia que encuentren a su paso. Mientras menor sea esa resistencia, mayor será el orden existente en el micromundo de los electrones; pero cuando la resistencia es elevada, comienzan a chocar unos con otros y a liberar energía en forma de calor. Esa situación hace que siempre se eleve algo la temperatura del conductor y que, además, adquiera valores más altos en el punto donde los electrones encuentren una mayor resistencia a su paso. Se representa con la letra griega omega (Ω), en honor al físico alemán Georg Ohm, quien descubrió el principio que ahora lleva su nombre.

Parlantes o buzzer

Es un transductor electroacústico utilizado para la reproducción de sonido. La transducción sigue un doble procedimiento: eléctrico-mecánico-acústico. En la primera etapa convierte las ondas eléctricas en energía mecánica, y en la segunda convierte la energía mecánica en ondas de frecuencia acústica. Es por tanto la puerta por donde sale el sonido al exterior desde los aparatos que posibilitaron su amplificación, su transmisión por medios telefónicos o radioeléctricos, o su tratamiento. El sonido se transmite mediante ondas sonoras, en este caso, a través

del aire. El oído capta estas ondas y las transforma en impulsos nerviosos que llegan al cerebro y se transforman en señales que se identifican con cosas como música, sonidos y onomatopeyas. Si se dispone de una grabación de voz, de música en soporte magnético o digital, o si se recibe estas señales por radio, se dispondrá a la salida del aparato de señales eléctricas que deben ser convertidas en sonidos; para ello se utiliza el altavoz.

Vatios

El vatio es la unidad de potencia del Sistema Internacional de Unidades. Su símbolo es W. Es el equivalente a 1 Joule por segundo (1 J/s) y es una de las unidades derivadas. Expresado en unidades utilizadas en electricidad, un vatio es la potencia eléctrica producida por una diferencia de potencial de 1 voltio y una corriente eléctrica de 1 amperio (1 voltamperio). La potencia eléctrica de los aparatos eléctricos se expresa en vatios, si son de poca potencia, pero si son de mediana o gran potencia se expresa en kilovatios (kW) que equivale a 1000 vatios. Un kW equivale a 1,35984 caballos de vapor.

Voltaje

Es una magnitud física que cuantifica la diferencia de potencial eléctrico entre dos puntos. También se puede definir como el trabajo por unidad de carga ejercido por el campo eléctrico sobre una partícula cargada para moverla entre dos posiciones determinadas. Se puede medir con un voltímetro. Su unidad de medida es el voltio. Si dos puntos que tienen una diferencia de potencial se unen mediante un conductor, se producirá un flujo de electrones. Parte de la carga que crea el punto de mayor potencial se trasladará a través del conductor al punto de menor potencial y, en ausencia de una fuente externa (generador), esta corriente cesará cuando ambos puntos igualen su potencial eléctrico. Este traslado de cargas es lo que se conoce como corriente eléctrica.

Protoboard

Es un tablero con orificios conectados eléctricamente entre sí, habitualmente siguiendo patrones de líneas, en el cual se pueden insertar componentes electrónicos y cables para el armado y prototipado de circuitos electrónicos y sistemas similares. Está hecho de dos materiales, un aislante, generalmente un plástico, y un conductor que conecta los diversos orificios entre sí. Uno de sus usos principales es la creación y comprobación de prototipos de circuitos electrónicos antes de llegar a la impresión mecánica del circuito en sistemas de producción comercial.

Modulación por anchos de pulso

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, por ejemplo), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga.

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$D = \frac{\tau}{T}$$

D es el ciclo de trabajo

τ es el tiempo en que la función es positiva (ancho del pulso)

T es el período de la función

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. Una de las entradas se conecta a un oscilador de onda dientes de sierra, mientras que la otra queda disponible para la señal

moduladora. En la salida la frecuencia es generalmente igual a la de la señal dientes de sierra y el ciclo de trabajo está en función de la portadora.

La principal desventaja que presentan los circuitos PWM es la posibilidad de que haya interferencias generadas por radiofrecuencia. Éstas pueden minimizarse ubicando el controlador cerca de la carga y realizando un filtrado de la fuente de alimentación.

Transistor

Un transistor es un dispositivo que regula el flujo de corriente o de tensión actuando como un interruptor o amplificador para señales electrónicas. El transistor, inventado en 1951, es el componente electrónico estrella, pues inició una auténtica revolución en la electrónica que ha superado cualquier previsión inicial. También se llama Transistor Bipolar o Transistor Electrónico.

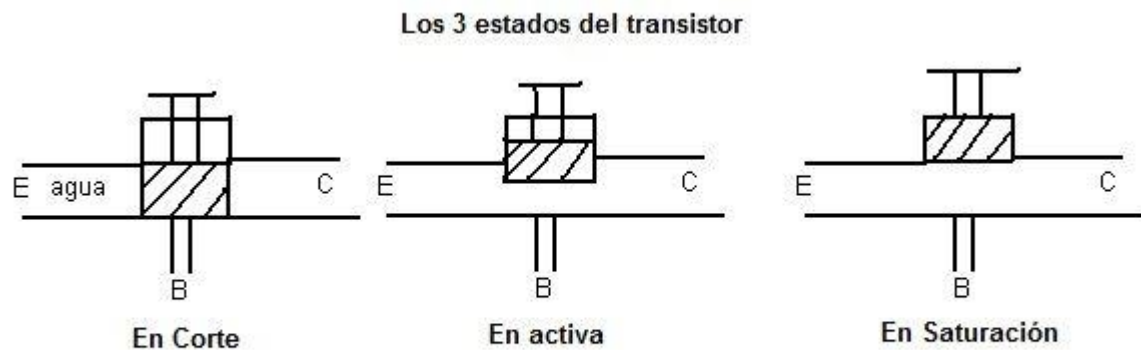
El Transistor es un componente electrónico formado por materiales semiconductores, de uso muy habitual, pues lo encontramos presente en cualquiera de los aparatos de uso cotidiano como las radios, alarmas, automóviles, ordenadores, etc. Vienen a sustituir a las antiguas válvulas termoiónicas de hace unas décadas. Gracias a ellos fue posible la construcción de receptores de radio portátiles llamados comúnmente "transistores", televisores que se encendían en un par de segundos, televisores en color, etc. Antes de aparecer los transistores, los aparatos a válvulas tenían que trabajar con tensiones bastante altas, tardaban más de 30 segundos en empezar a funcionar, y en ningún caso podían funcionar a pilas debido al gran consumo que tenían. Los transistores son unos elementos que han facilitado, en gran medida, el diseño de circuitos electrónicos de reducido tamaño, gran versatilidad y facilidad de control. En la siguiente imagen podemos ver varios transistores diferentes.

Funcionamiento del Transistor

Un transistor puede tener 3 estados posibles en su trabajo dentro de un circuito:

- ✓ En activa: deja pasar más o menos corriente.
- ✓ En corte: no deja pasar la corriente.
- ✓ En saturación: deja pasar toda la corriente.

Para comprender estos 3 estados lo vamos hacer mediante un símil hidráulico que es más fácil de entender. Lo primero imaginemos que el transistor es una llave de agua como la de la figura. Se puede aplicar con el concepto de agua para entender el funcionamiento, pero solo hay que cambiar el agua por corriente eléctrica, y la llave de agua por el transistor y ya estaría entendido.

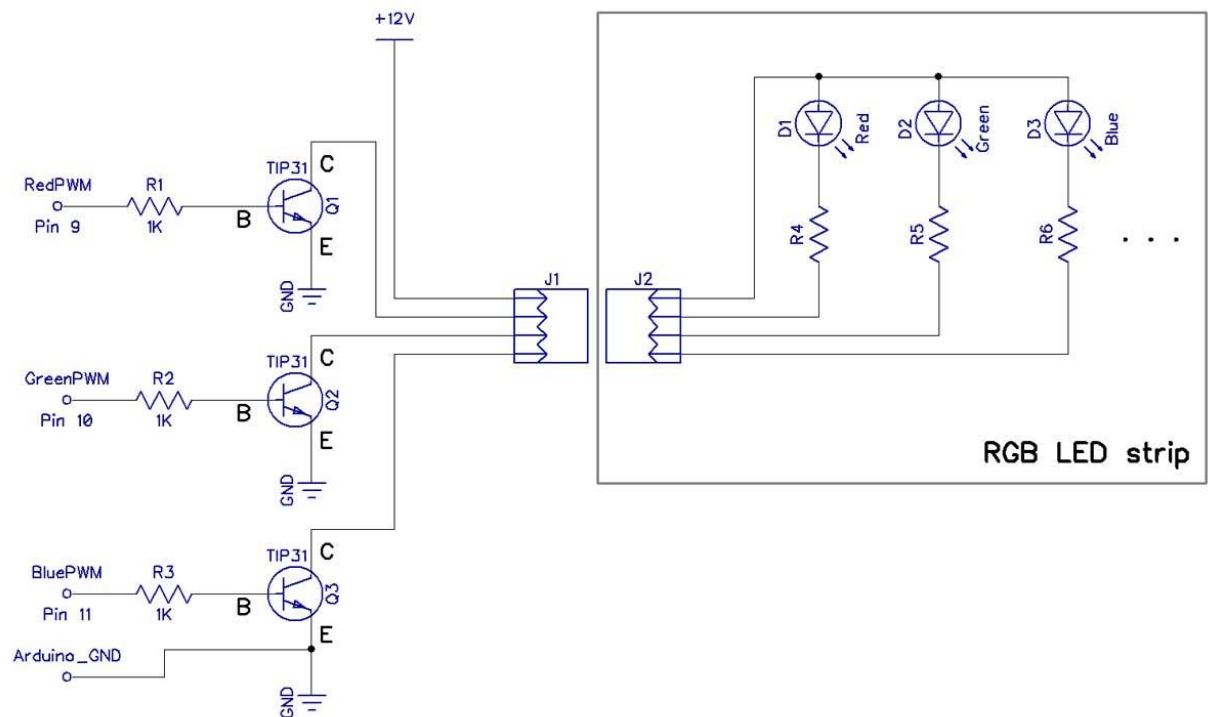


www.areatecnologia.com

En la figura vemos la llave de agua en 3 estados diferentes. Para que la llave suba y pueda pasar agua desde la tubería E hacia la tubería C, es necesario que entre algo de agua por la pequeña tubería B y empuje la llave hacia arriba (que el cuadrado de líneas suba y permita el paso de agua). En el símil tenemos: B = base, E = Emisor y C = Colector.

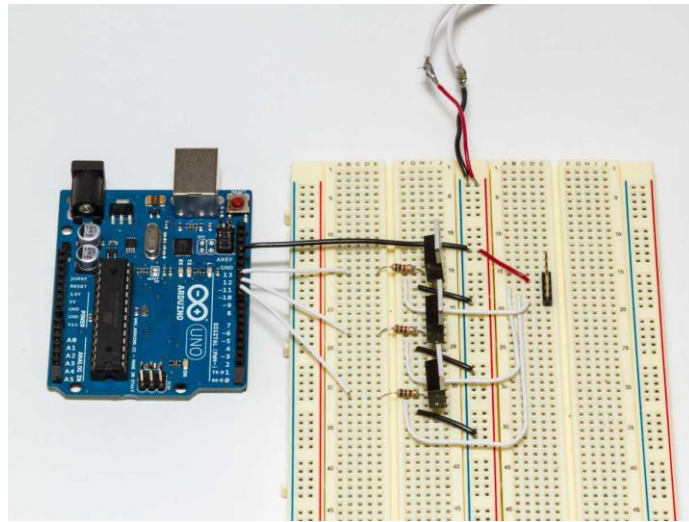
Procedimiento

Al momento de tener los materiales requeridos precedemos cablear y ubicar cada parte del circuito como en el siguiente diagrama, el cual es el bosquejo de lo que será la parte principal del sistema. Es recomendable siempre diseñar un bosquejo primero entendiendo la lógica de lo que se está construyendo y así evitar errores comunes.

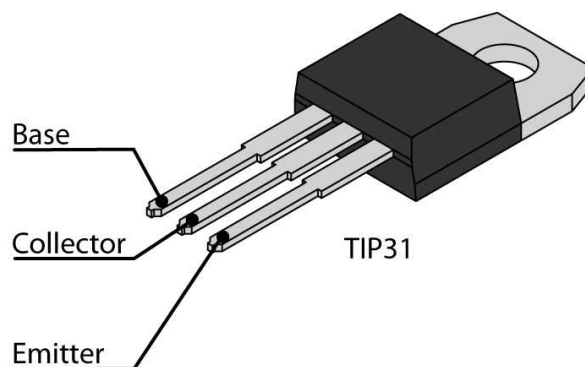


Una recomendación es tomar el circuito primero, después verificarlo que cada componente funcione correctamente antes de construir la maqueta para montar el circuito, este circuito utiliza el puerto USB de alimentación de 5V al igual que usa la conexión COM en el PC. Otro punto a tener en cuenta es que se debe anexar la

fuentes de poder de 12V a una canaleta de positivo. Al construir la parte principal del circuito con el bosquejo anterior las conexiones deben quedar como la siguiente.



A la hora de ubicar los transistores tenemos que tener siempre en cuenta como funciona y que este en la posición correcta para evitar posibles daños en el Arduino. Se debe tener muy en cuenta cual transistor controlara que color y conectar los transistores de la forma correcta de acuerdo a los pines del TIP31. Y no olvidar conectar el pin de tierra de Arduino a la tierra de la fuente de poder de 12V.



En adición tener en cuenta que los RGB LED tienen usualmente cuatro cables: Uno para poder que son 12 V y uno de los tres colores: rojo, verde y azul. Cuando la cinta se le proporciona poder, haciendo contacto cualquiera de las líneas de control de colores a tierra causará que el color LED brille al máximo. Si se usa una modulación por anchos de pulso en estas líneas de control de color permite modular el brillo de las luces. Una cinta de un metro de largo puede arrastrar aproximadamente 1A cuando está en máximo brillo el rojo, verde y azul. El output o salida de un Arduino solo puede proveer cerca de 40mA cada uno, por eso es necesario que se implemente un soporte con un controlador de circuito para mejorar la energía. Este circuito diseñado toma tres señales PWM del arduino y las usa para manejar tres transistores que proveen energía roja, verde y azul, dando el control total sobre el brillo de cada color, dando la posibilidad de crear cualquier color del espectro. En este caso al desear conectar un metro de LED se debe usar los transistores de lo contrario si fuera uno o dos LED se puede conectar directamente.

El primer controlador de circuito es un transistor amplificador básico repetido tres veces. Una baja corriente de 5v de la señal PWM del arduino es dirigida a la base (B) del transistor a través de una resistencia de 1K ohm. Esta señal cambia el transistor permitiéndole conducir una corriente mayor a 12V por el colector (C) y el emisor (E) a través de los LEDs. El transistor puede cambiar suficientemente rápido permitiendo que el poder del LED es la señal PWM que se recibe de entrada dando resultados deseados.

Usamos programación en Arduino, permitiéndonos hacer uso del puerto COM, el programa lo descargamos de la página oficial de Arduino.

Arduino IDE

Arduino 1.0.6

Download

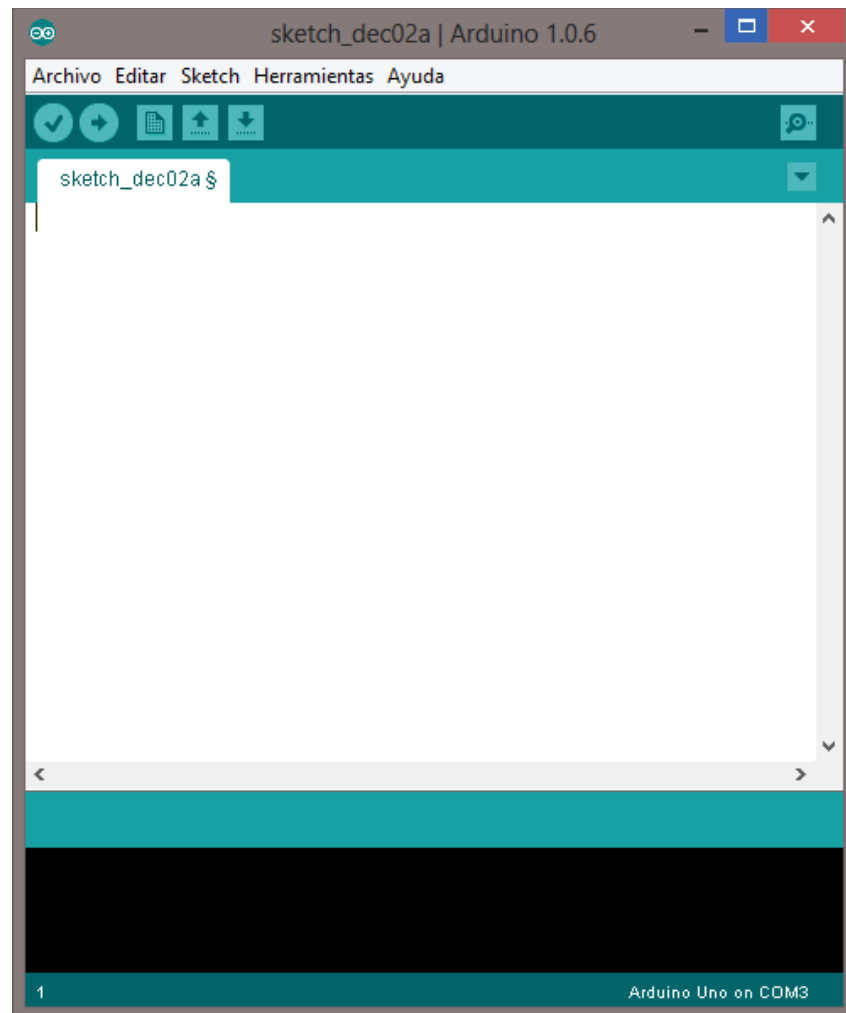
Arduino 1.0.6 (release notes):

- [Windows Installer](#), Windows ZIP file (for non-administrator install)
- Mac OS X
- Linux: 32 bit, 64 bit
- [source](#)

Next steps

[Getting Started](#)
[Reference](#)
[Environment](#)
[Examples](#)
[Foundations](#)
[FAQ](#)

Después procedemos a programar nuestro circuito con el IDE principal de Arduino que se puede descargar fácilmente desde su página oficial.



Código en Arduino Sistema de control de diodos de luces:

```

/*
  Super Mario Brothers Overworld Canción

  Note Durations:
    1 = whole note, 2 = half note,
    4 = quarter note, 8 = eighth note,
    and so on.

  */
#include "pitches.h"

int arrayLength;

// intro notes and durations
int smbIntroNotes[] = {
    NOTE_E4, NOTE_E4, 0, NOTE_E4, 0, NOTE_C4, NOTE_E4, 0,
    NOTE_G4, 0, 0, NOTE_G3, 0, 0
};
int smbIntroDurations[] = {
    8, 8, 8, 8, 8, 8, 8, 8,
    8, 8, 4, 8, 8, 4
};

// part A notes and durations
int smbANotes[] = {
    NOTE_C4, 0, 0, NOTE_G3, 0, NOTE_E3, 0,
    0, NOTE_A3, 0, NOTE_B3, 0, NOTE_AS3, NOTE_A3, 0,
    NOTE_G3, NOTE_E4, NOTE_G4, NOTE_A4, 0, NOTE_F4, NOTE_G4,
    0, NOTE_E4, 0, NOTE_C4, NOTE_D4, NOTE_B3, 0,
    NOTE_C4, 0, 0, NOTE_G3, 0, NOTE_E3, 0,
    0, NOTE_A3, 0, NOTE_B3, 0, NOTE_AS3, NOTE_A3, 0,
    NOTE_G3, NOTE_E4, NOTE_G4, NOTE_A4, 0, NOTE_F4, NOTE_G4,
    0, NOTE_E4, 0, NOTE_C4, NOTE_D4, NOTE_B3, 0
};
int smbADurations[] = {
    8, 8, 8, 8, 4, 8, 8,
    8, 8, 8, 8, 8, 8, 8,
    6, 6, 6, 8, 8, 8, 8,
    8, 8, 8, 8, 8, 8, 4,
    8, 8, 8, 8, 4, 8, 8,
    8, 8, 8, 8, 8, 8, 8,
    6, 6, 6, 8, 8, 8, 8,
    8, 8, 8, 8, 8, 8, 4
};

// part B notes and durations
int smbBNotes[] = {
    0, NOTE_G4, NOTE_FS4, NOTE_FF4, NOTE_DS4, 0, NOTE_E4,
    0, NOTE_GS3, NOTE_A3, NOTE_C4, 0, NOTE_A3, NOTE_C4, NOTE_D4,
    0, NOTE_G4, NOTE_FS4, NOTE_FF4, NOTE_DS4, 0, NOTE_E4,
    0, NOTE_C5, 0, NOTE_C5, NOTE_C5, 0, 0,
    0, NOTE_G4, NOTE_FS4, NOTE_FF4, NOTE_DS4, 0, NOTE_E4,

```



```

0, NOTE_GS3, NOTE_A3, NOTE_C4, 0, NOTE_A3, NOTE_C4, NOTE_D4,
0, NOTE_DS4, 0, 0, NOTE_D4, 0,
NOTE_C4, 0, 0, 0,
0, NOTE_G4, NOTE_FS4, NOTE_FF4, NOTE_DS4, 0, NOTE_E4,
0, NOTE_GS3, NOTE_A3, NOTE_C4, 0, NOTE_A3, NOTE_C4, NOTE_D4,
0, NOTE_G4, NOTE_FS4, NOTE_FF4, NOTE_DS4, 0, NOTE_E4,
0, NOTE_C5, 0, NOTE_C5, NOTE_C5, 0, 0,
0, NOTE_G4, NOTE_FS4, NOTE_FF4, NOTE_DS4, 0, NOTE_E4,
0, NOTE_GS3, NOTE_A3, NOTE_C4, 0, NOTE_A3, NOTE_C4, NOTE_D4,
0, NOTE_DS4, 0, 0, NOTE_D4, 0,
NOTE_C4, 0, 0, 0
};
int smbBDurations[] = {
4, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8,
4, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 4,
4, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8,
4, 8, 8, 8, 8, 4,
8, 8, 4, 2,
4, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8,
4, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 4,
4, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 8,
4, 8, 8, 8, 8, 4,
8, 8, 4, 2
};

// part C notes and durations
int smbCNotes[] = {
NOTE_C4, NOTE_C4, 0, NOTE_C4, 0, NOTE_C4, NOTE_D4, 0,
NOTE_E4, NOTE_C4, 0, NOTE_A3, NOTE_G3, 0, 0,
NOTE_C4, NOTE_C4, 0, NOTE_C4, 0, NOTE_C4, NOTE_D4, NOTE_E4,
0,
NOTE_C4, NOTE_C4, 0, NOTE_C4, 0, NOTE_C4, NOTE_D4, 0,
NOTE_E4, NOTE_C4, 0, NOTE_A3, NOTE_G3, 0, 0,
NOTE_E4, NOTE_E4, 0, NOTE_E4, 0, NOTE_C4, NOTE_E4, 0,
NOTE_G4, 0, 0, NOTE_G3, 0, 0
};
int smbCDurations[] = {
8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 4,
8, 8, 8, 8, 8, 8, 8, 8,
1,
8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 8, 8, 8, 8, 4,
8, 8, 8, 8, 8, 8, 8, 8,
8, 8, 4, 8, 8, 4
};

// part D notes and durations
int smbDNotes[] = {
NOTE_E4, NOTE_C4, 0, NOTE_G3, 0, NOTE_GS3, 0,
NOTE_A3, NOTE_F4, 0, NOTE_F4, NOTE_A3, 0, 0,

```

```

NOTE_B3, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_G4, NOTE_F4,
NOTE_E4, NOTE_C4, 0, NOTE_A3, NOTE_G3, 0, 0,
NOTE_E4, NOTE_C4, 0, NOTE_G3, 0, NOTE_GS3, 0,
NOTE_A3, NOTE_F4, 0, NOTE_F4, NOTE_A3, 0, 0,
NOTE_B3, NOTE_F4, 0, NOTE_F4, NOTE_F4, NOTE_E4, NOTE_D4,
NOTE_C4, NOTE_E3, 0, NOTE_E3, NOTE_C2, 0, 0
};
int smbDDurations[] = {
    8, 8, 8, 8, 4, 8, 8,
    8, 8, 8, 8, 8, 8, 4,
    6, 6, 6, 6, 6, 6,
    8, 8, 8, 8, 8, 8, 4,
    8, 8, 8, 8, 4, 8, 8,
    8, 8, 8, 8, 8, 8, 4,
    8, 8, 8, 8, 6, 6, 6,
    8, 8, 8, 8, 8, 8, 4
};

```

```
int tempo = 1000;
```

```

/*-----
-----CODIGO---ARDUINO-----
-----*/

```

```
//Definiciones de pines. Deben ser pines PWM en el Arduino para el RGB Led!
```

```

const int bluePin = 9;
const int redPin = 10;
const int greenPin = 11;
const int pirPin = 2;
const int pinBuzzer = 12;

```

```
//Recibe las unidades, decenas y centenas y el resultado p
```

```

int u, d, c, p;
//Donde se cambiara el color
int input;
//Amarillo por defecto
int lastColor[3] = {246, 112, 0};

```

```

unsigned long previousMillis = 0;    // will store last time LED was updated
const long interval = 90000;        // interval at which to blink (milliseconds)

```

```

void setup() {
    //Se asignan los pines a usar
    pinMode(redPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
    pinMode(bluePin, OUTPUT);
    pinMode(pirPin, INPUT);
}

```

```

//Se empieza la serializacion
Serial.begin(9600);

```

```
for (int i = 0; i < 4; i++) {
```

```

    inicio(); //Parpadeo de luces para indicar inicio
  }
}
/*-----*/
void loop() {

  /*Revisa si se estan enviando datos desde la App*/
  if (Serial.available()) {
    input = Serial.read() - 48;
    delay(10);
    c = Serial.read() - 48;
    delay(10);
    d = Serial.read() - 48;
    delay(10);
    u = Serial.read() - 48;
    delay(10);
    p = (100 * c) + (10 * d) + u;
    if (input == 1) {
      analogWrite(redPin, p);
      lastColor[0] = p;
    }
    if (input == 2) {
      analogWrite(greenPin, p);
      lastColor[1] = p;
    }
    if (input == 3) {
      analogWrite(bluePin, p);
      lastColor[2] = p;
    }
  }
  /*Revisa si hay moviminto en el PIR*/
  int value = digitalRead(pirPin);
  if (value == HIGH)
  {
    movimiento();
  }
  else {
    analogWrite(redPin, lastColor[0]);
    analogWrite(greenPin, lastColor[1]);
    analogWrite(bluePin, lastColor[2]);
  }

  /*Revisa si ya ha pasado el tiempo estimado*/
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {

    cambioHora();

    previousMillis = millis();
    analogWrite(redPin, lastColor[0]);
    analogWrite(greenPin, lastColor[1]);
    analogWrite(bluePin, lastColor[2]);
  }
}

```

```

/*-----*/

void inicio() {
  amarillo();
  delay(300);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
  delay(300);
  amarillo();
}

void movimiento() {
  azul();
  for(int i=255; i>1; i--){
    analogWrite(greenPin,i);
    delay(7);
  }
  for(int i=0; i<255; i++){
    analogWrite(greenPin,i);
    delay(7);
  }
}

void cambioHora() {
  int FADESPEED = 1;

  // fade from blue to violet
  for (int red = 0; red < 256; red++) {
    analogWrite(redPin, red);
    delay(FADESPEED);
  }
  // fade from violet to red
  for (int blue = 255; blue > 0; blue--) {
    analogWrite(bluePin, blue);
    delay(FADESPEED);
  }
  // fade from red to yellow
  for (int green = 0; green < 256; green++) {
    analogWrite(greenPin, green);
    delay(FADESPEED);
  }
  // fade from yellow to green
  for (int red = 255; red > 0; red--) {
    analogWrite(redPin, red);
    delay(FADESPEED);
  }
  // fade from green to teal
  for (int blue = 0; blue < 256; blue++) {
    analogWrite(bluePin, blue);
    delay(FADESPEED);
  }
  // fade from teal to blue
  for (int green = 255; green > 0; green--) {
    analogWrite(greenPin, green);
    delay(FADESPEED);
  }
}

```

```

digitalWrite(redPin, LOW);
digitalWrite(greenPin, LOW);
digitalWrite(bluePin, LOW);

comenzarCancion();

for (int i = 0; i < 30; i++) {
  amarillo();
  delay(20);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
  delay(20);
}
for (int i = 0; i < 30; i++) {
  azul();
  delay(20);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
  delay(20);
}
for (int i = 0; i < 30; i++) {
  rojo();
  delay(20);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
  delay(20);
}
for (int i = 0; i < 30; i++) {
  verde();
  delay(20);
  digitalWrite(redPin, LOW);
  digitalWrite(greenPin, LOW);
  digitalWrite(bluePin, LOW);
  delay(20);
}
}

void playNotes(int myNotes[], int myDurations[], int myLength) {

  Serial.println(arrayLength);
  for (int thisNote = 0; thisNote < arrayLength; thisNote++) {
    // to calculate the note duration, take one second
    // divided by the note type.
    int noteDuration = tempo / myDurations[thisNote];
    analogWrite(redPin, myNotes[thisNote]);
    tone(pinBuzzer, myNotes[thisNote], noteDuration);

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
  }
}

```

```

    // stop the tone playing:
    noTone(pinBuzzer);

}
}

void comenzarCancion() {
    arrayLength = sizeof(smbIntroDurations) / sizeof(int);
    playNotes(smbIntroNotes, smbIntroDurations, arrayLength);

    arrayLength = sizeof(smbADurations) / sizeof(int);
    playNotes(smbANotes, smbADurations, arrayLength);

    arrayLength = sizeof(smbBDurations) / sizeof(int);
    playNotes(smbBNotes, smbBDurations, arrayLength);

    arrayLength = sizeof(smbCDurations) / sizeof(int);
    playNotes(smbCNotes, smbCDurations, arrayLength);

    arrayLength = sizeof(smbADurations) / sizeof(int);
    playNotes(smbANotes, smbADurations, arrayLength);

    arrayLength = sizeof(smbDDurations) / sizeof(int);
    playNotes(smbDNotes, smbDDurations, arrayLength);

}

void amarillo() {
    analogWrite(redPin, 246);
    analogWrite(greenPin, 112);
    analogWrite(bluePin, 0);
}

void azul() {
    analogWrite(redPin, 0);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 255);
}

void rojo() {
    analogWrite(redPin, 255);
    analogWrite(greenPin, 0);
    analogWrite(bluePin, 0);
}

void verde() {
    analogWrite(redPin, 0);
    analogWrite(greenPin, 255);
    analogWrite(bluePin, 0);
}

```

Este código se encarga básicamente de configurar en sensor PIR para que cuando un usuario sea detectado, la lámpara cambie de color, que cuando pase una hora suene una canción y avise al usuario el cambio de hora, para configurar el control de leds a través de una aplicación, etc.

Al momento de terminar el código y sin errores procedemos a abrir Java, ya que lo usaremos para encender el circuito del Parquero y cerrar la comunicación de Java a Arduino.

Ejecutamos Java preferiblemente la Versión 7.3.1 en adelante

Descargamos la librería de Arduino Versión 2.2 en adelante

Para poder utilizar Java y Arduino necesitamos utilizar una librería especial en Java llamada RXTX. Esta librería es requerida por Java para enviar y recibir información a través del puerto serie.

Descargamos el archivo que nos corresponda según la arquitectura que estemos utilizando.

Ahora tenemos que instalar las librerías.

Nos dirigimos al disco C, o donde sea que tengamos nuestro sistema operativo instalado. Buscamos archivos de programas, Java. Cuando entremos a la carpeta Java encontramos otra carpeta llamada jre. En mi caso me aparece jre7.

Ahora hacemos lo siguiente:




El archivo RXTXcomm.jar lo copiamos en la carpeta lib/ext que está dentro de la carpeta jre.

También es necesario copiarlo en la siguiente ruta:

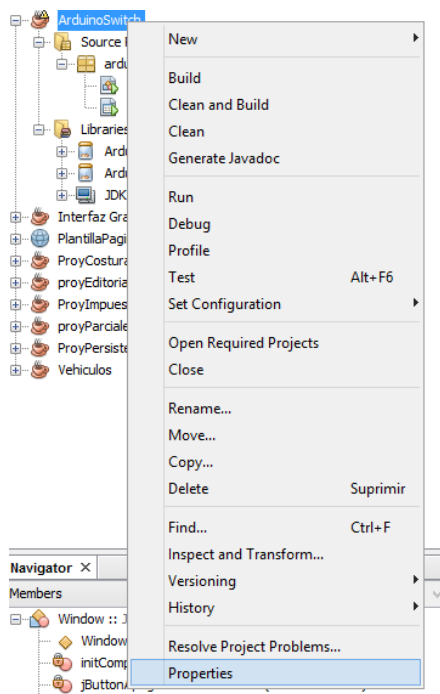
C: Program Files/Java/jdk1.7.0_15/bin

Haciendo esto hemos preparado nuestra computadora para que sea capaz de comunicar a Arduino con una aplicación Java.

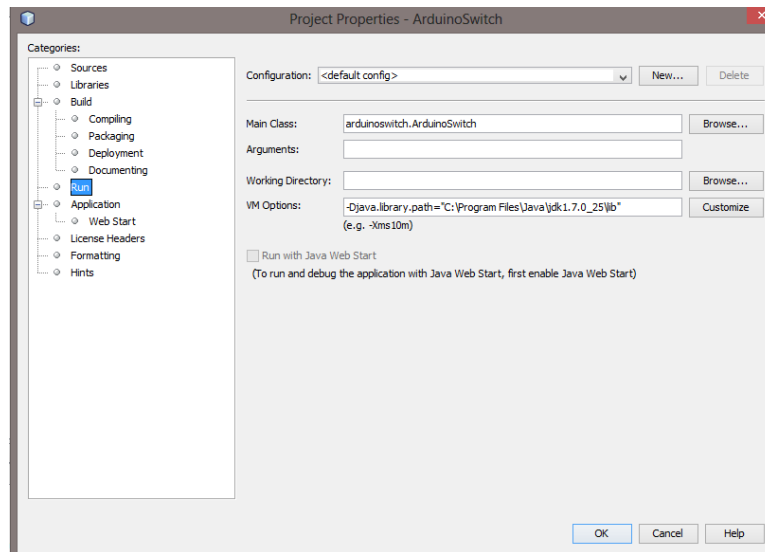
Una vez terminados los pasos anteriores si no funcionan los comandos de comunicación RX, TX o RXTX realizamos lo siguiente nos dirigimos a C:\Program Files\Java\jdk1.7.0_25, copiamos estos 3 archivos tanto en la carpeta lib como en la carpeta bin.

	RXTXcomm	07/12/2008 10:45 ...	Executable Jar File	60 KB
	rxtxParallel.dll	07/12/2008 10:45 ...	Extensión de la apl...	83 KB
	rxtxSerial.dll	07/12/2008 10:45 ...	Extensión de la apl...	127 KB

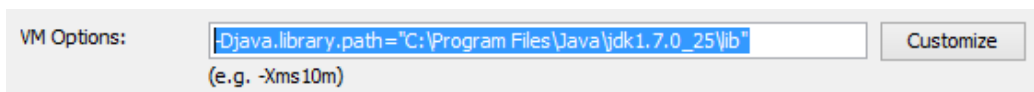
Después seleccionamos nuestro proyecto y damos click derecho



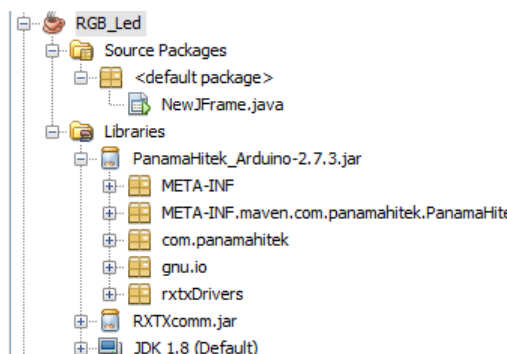
Después en la ventana que aparece, en la lista de la parte izquierda buscamos RUN



En la opción VM Options digitamos la ruta de la siguiente imagen, la dirección que está en comillas puede variar ya que es la ruta de instalación de java específicamente en la carpeta lib, lo demás es igual solo cambia la ruta que está en comillas.



Al hacer esto damos click en OK y procedemos a usar RXTX. Procedemos a crear nuestro proyecto de Java Application creamos un JFrame el cual vamos a diseñar para controlar el color que queremos desde la aplicación. Con la configuración de las librerías debe mostrarse algo así.



Código del JFrame principal.

```
import com.panamahitek.PanamaHitek_Arduino;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.colorchooser.ColorSelectionModel;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author YojhanLR
 */
public class NewJFrame extends javax.swing.JFrame {

    int R = 0, G = 0, B = 0; //Variables para cada //color (Rojo, Verde y Azul)
    PanamaHitek_Arduino Arduino = new PanamaHitek_Arduino(); //Variable para //instanci
    String OutputR, OutputG, OutputB; //Variables que //contendrán los valores a enviar

    public NewJFrame() {
        initComponents();
        try {
            Arduino.ArduinoTX("COM3", 2000, 9600);
        } catch (Exception ex) {
            System.out.println("Error: "+ex);
        }

        this.setLocationRelativeTo(null); //Ubica ventana en la mitad de la pantalla

        ColorSelectionModel model = jColorChooser1.getSelectionModel();
        ChangeListener changeListener = new ChangeListener() {

            @Override
            public void stateChanged(ChangeEvent e) {
                try {
                    System.out.println("Cambio de color: "+jColorChooser1.getColor());
                } catch (Exception ex) {
                    System.out.println("Error: "+ex);
                }
            }
        };
    }
}
```

```

43         R = jColorChooser1.getColor().getRed();
44         G = jColorChooser1.getColor().getGreen();
45         B = jColorChooser1.getColor().getBlue();
46         Thread.sleep(100);
47         SetData();
48         try {
49             Arduino.sendData(OutputR);
50             Arduino.sendData(OutputG);
51             Arduino.sendData(OutputB);
52         } catch (Exception ex) {
53             System.out.println("Error enviando datos: "+ex);
54         }
55     } catch (InterruptedException ex) {
56         Logger.getLogger(NewJFrame.class.getName()).log(Level.SEVERE, null, ex);
57     }
58 }
59 };
60 model.addChangeListener(changeListener);
61 }
62
63 /**
64  * This method is called from within the constructor to initialize the form.
65  * WARNING: Do NOT modify this code. The content of this method is always
66  * regenerated by the Form Editor.
67  */
68 @SuppressWarnings("unchecked")
69 Generated Code
70
71 /**
72  * @param args the command line arguments
73  */
74 public static void main(String args[]) {
75     /* Set the Nimbus look and feel */
76     Look and feel setting code (optional)
77
78     /* Create and display the form */
79     java.awt.EventQueue.invokeLater(new Runnable() {
80         public void run() {
81             new NewJFrame().setVisible(true);
82         }
83     });
84 }

```

```

    public void SetData() {
        OutputR = "1";
        OutputG = "2";
        OutputB = "3";
        if (R < 10) {
            OutputR = OutputR + "00" + R;
        } else if (R < 100) {
            OutputR = OutputR + "0" + R;
        } else {
            OutputR = OutputR + R;
        }

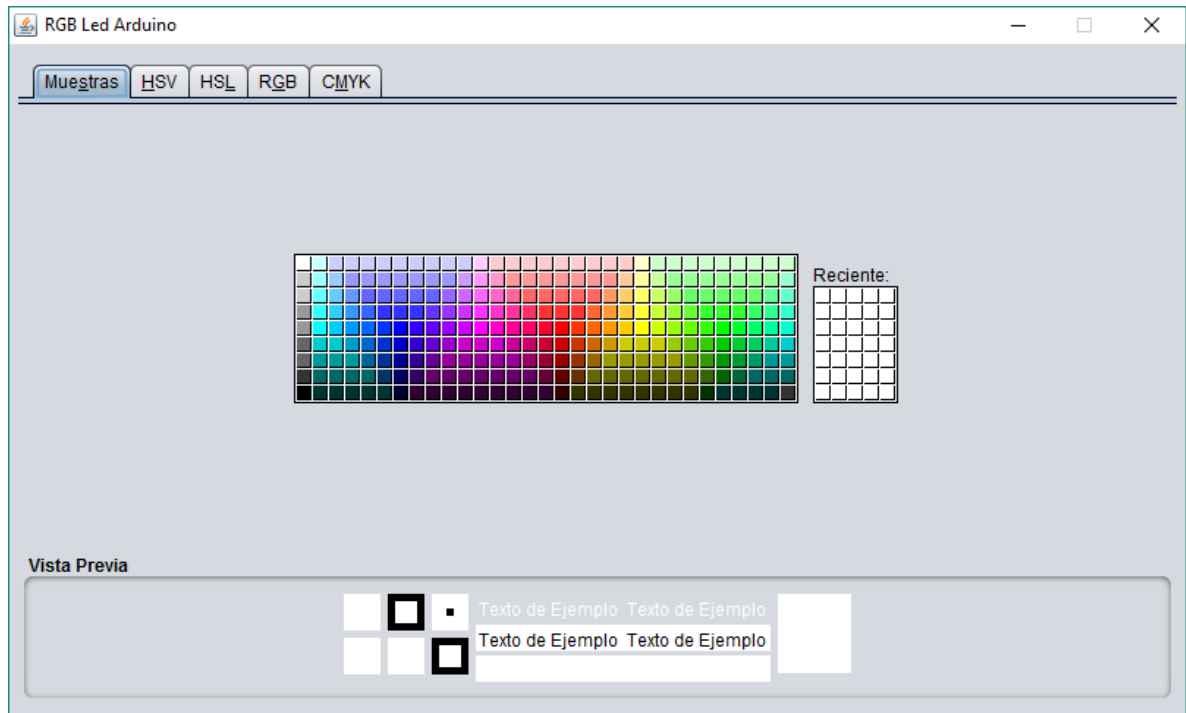
        if (G < 10) {
            OutputG = OutputG + "00" + G;
        } else if (G < 100) {
            OutputG = OutputG + "0" + G;
        } else {
            OutputG = OutputG + G;
        }

        if (B < 10) {
            OutputB = OutputB + "00" + B;
        } else if (B < 100) {
            OutputB = OutputB + "0" + B;
        } else {
            OutputB = OutputB + B;
        }
    }

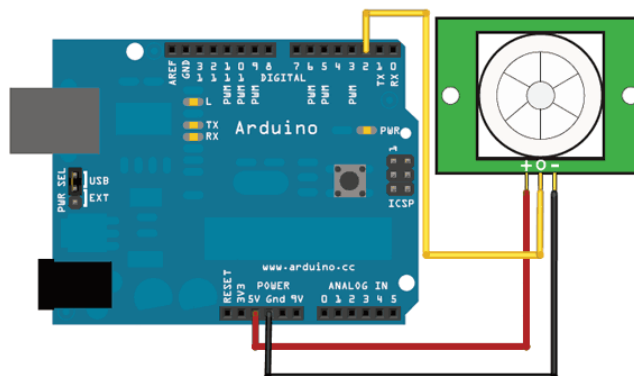
    // Variables declaration - do not modify
    private javax.swing.JColorChooser jColorChooser1;
    // End of variables declaration

```

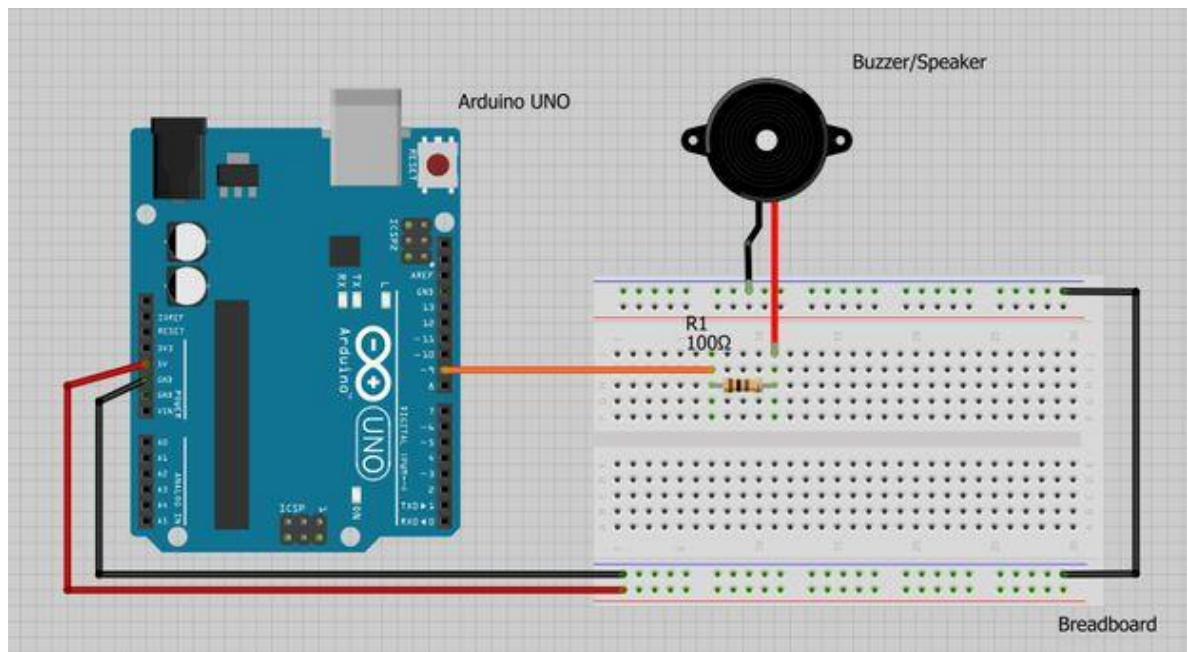
Una vez compilada la aplicación, podemos conectarnos con el Arduino desde nuestra aplicación y así poder escoger el color que deseamos.



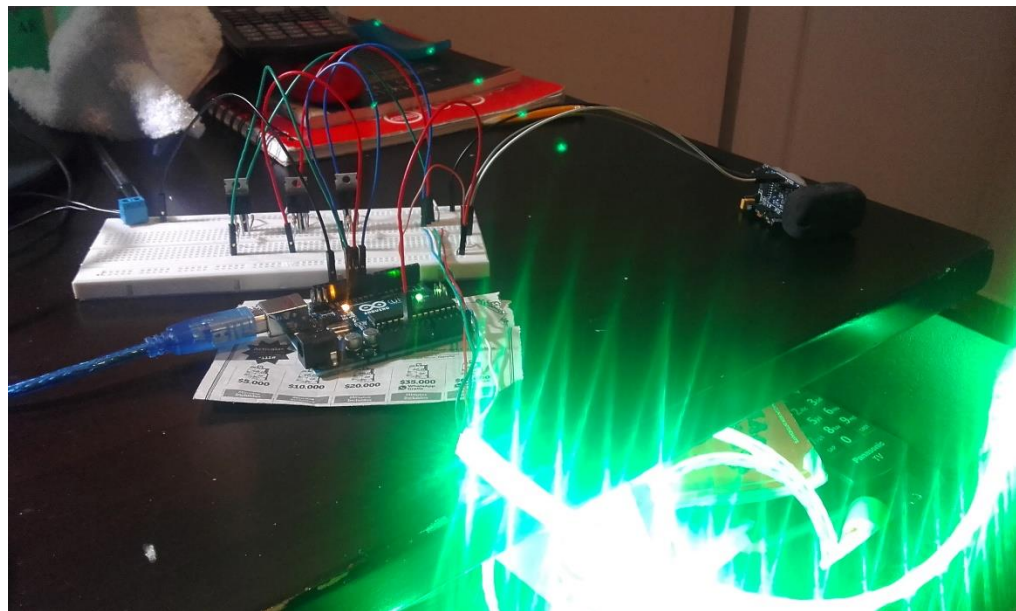
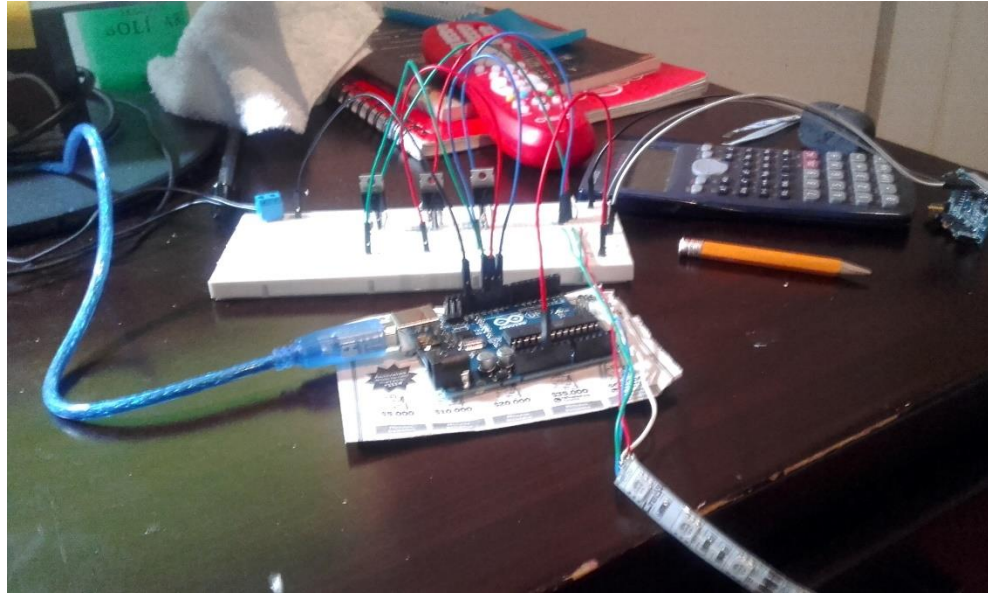
Un elemento que falta por explicar es el el PIR que para instalarlo de manera correcta debemos conectarlo al pin numero dos del arduino y tener en cuenta ademas de los tres pines que éste incluye: positivo, señal y tierra.

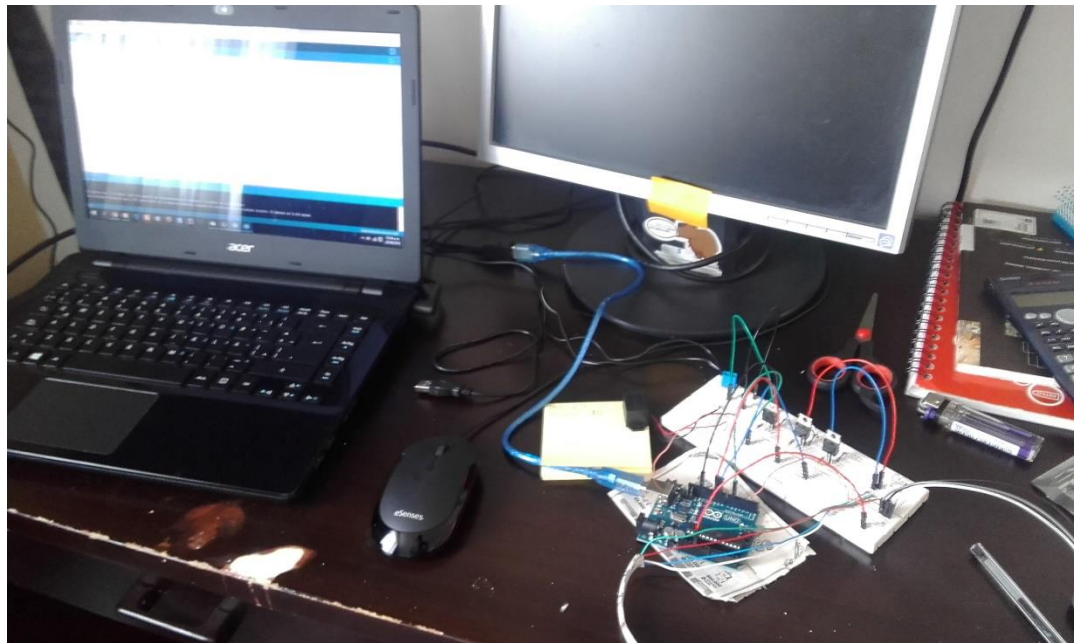


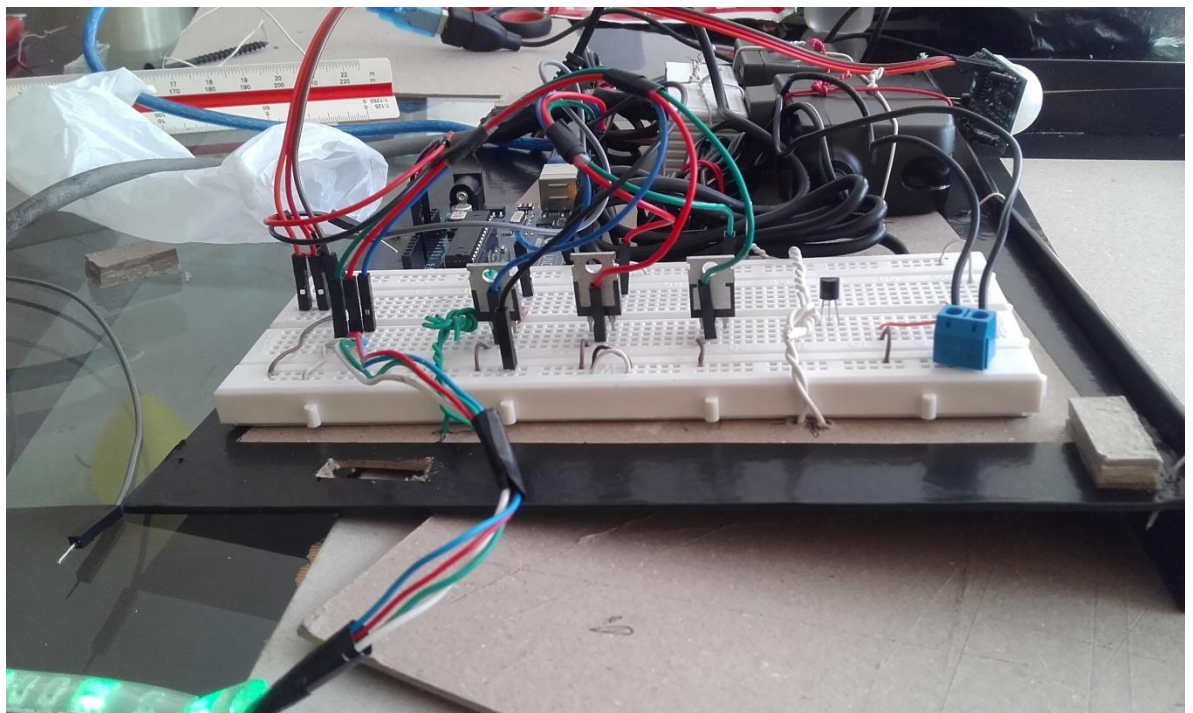
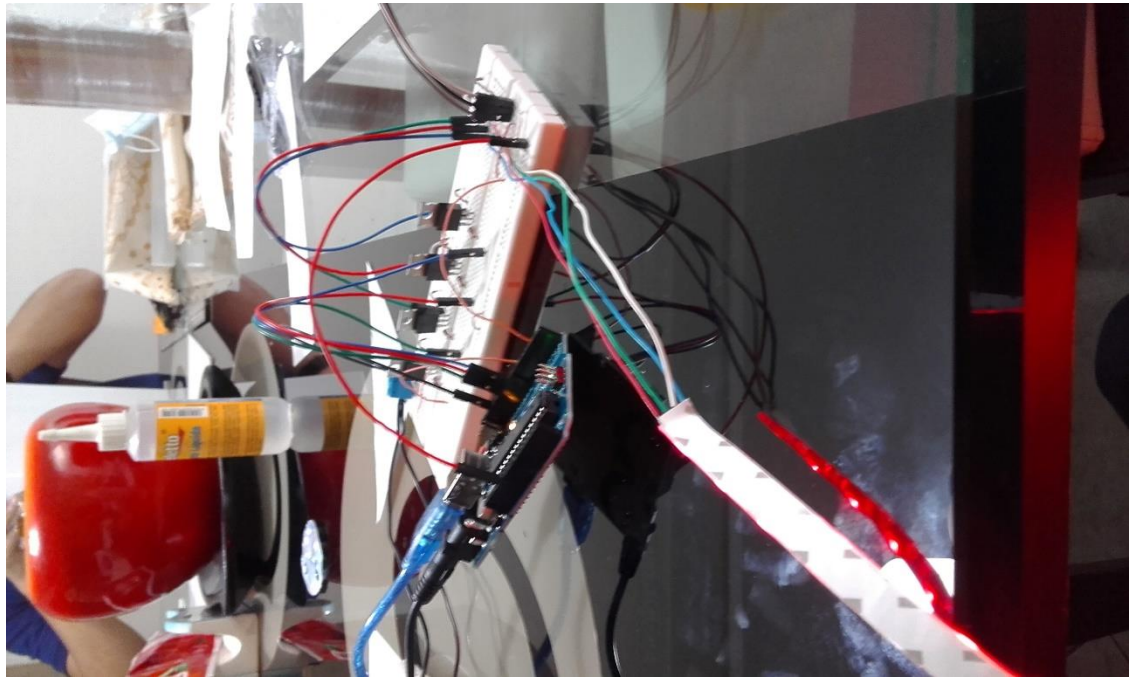
Por ultimo el buzzer de 5v debe ir conectado con una resistencia de 100 ohm por seguridad y puede ir en cualquier pin.

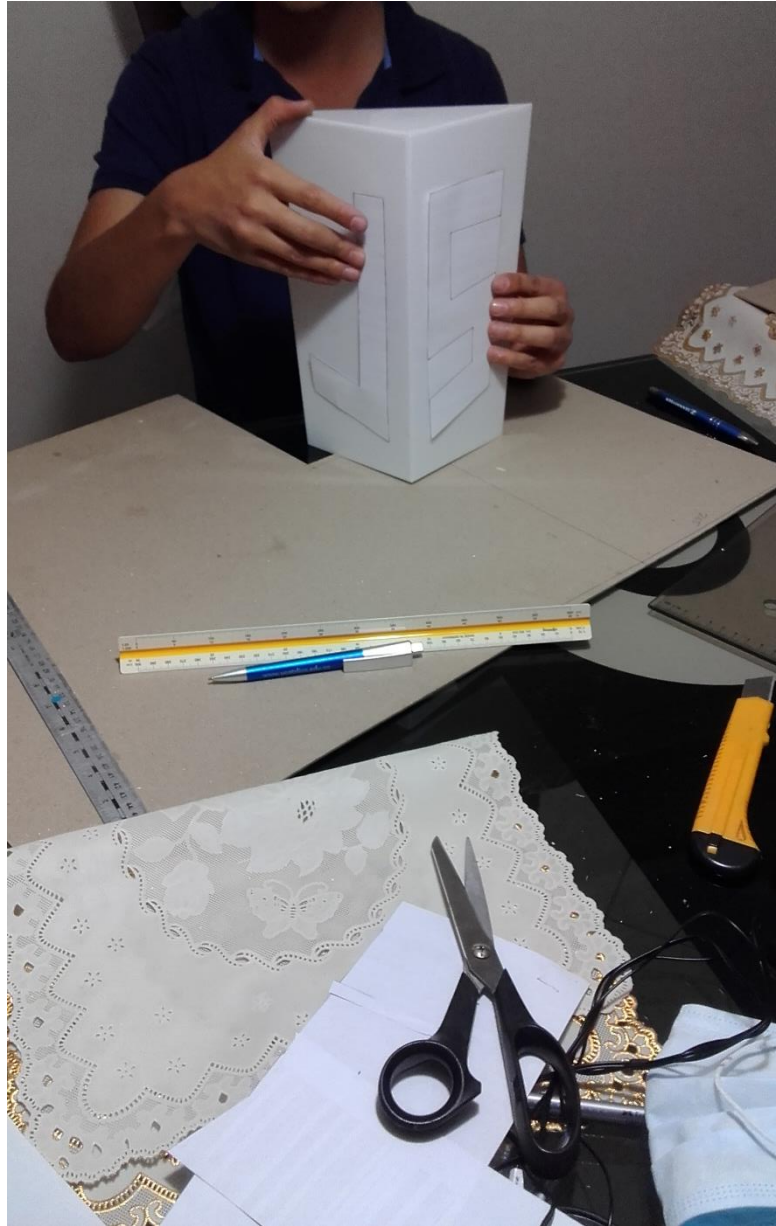


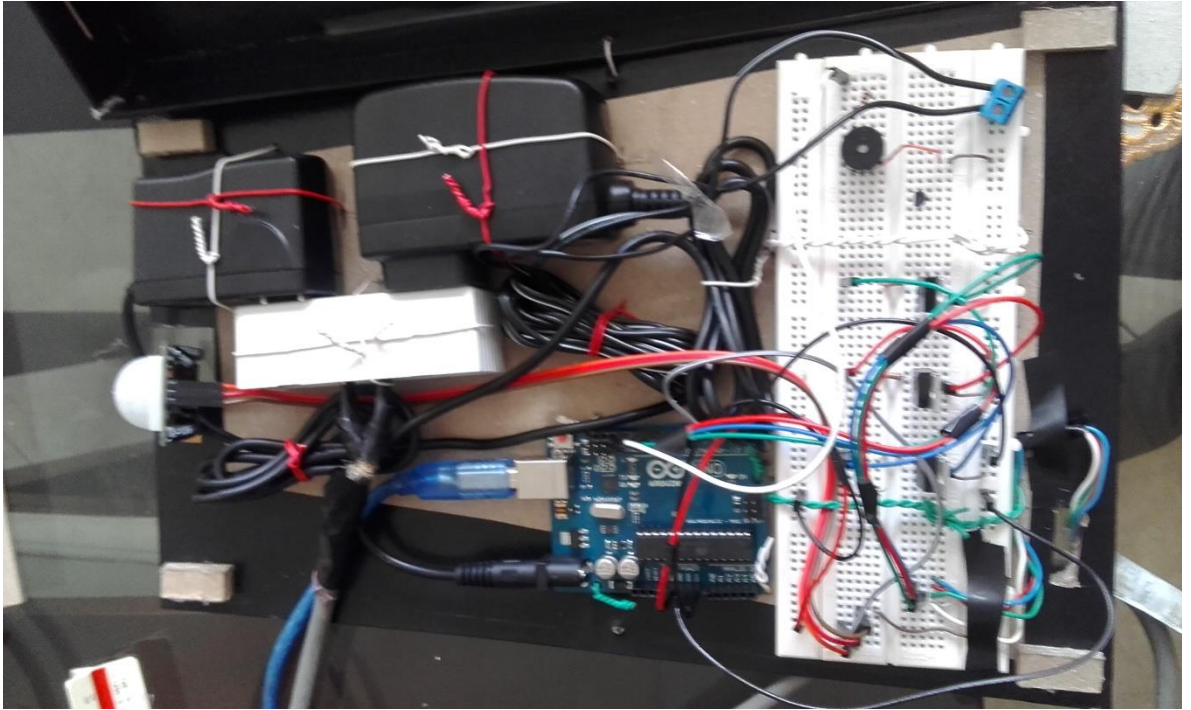
Montaje del sistema control de diodos de luz.









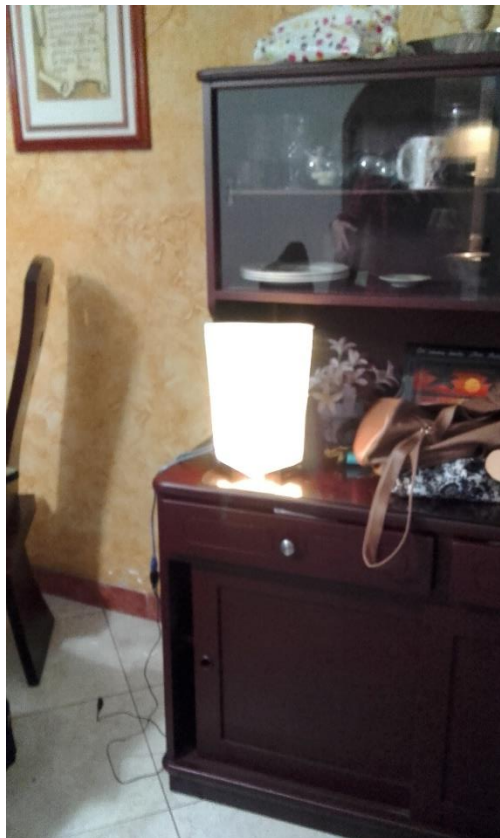


Resultados

En el momento de programar cada componente en Arduino se dificulto el tiempo y el retraso de la iluminación y asignarle el control automático y manual al sistema. De la misma manera fue un reto controlar que por el tiempo sonara una melodía.

Al momento de compilarlo aparece la siguiente ventana en Arduino

```
Cargando...  
Tamaño binario del Sketch: 6.294 bytes (de un máximo de 32.256 bytes)  
1
```







Conclusiones

Este proyecto es bastante llamativo ya que permite al usuario controlar el sistema desde su computador y asignarle cualquier color del espectro para conseguir el efecto deseado. Además su sistema de detección de intruso puede alertar al usuario que alguien ha entrado a una determinada área. En adición a esto, su sistema de alarma junto a una melodía para indicarle al usuario el cambio de hora es bastante útil para la persona dueña del sistema.

Que este sistema de control LED sea capaz de ser manejado tanto de manera manual como automática le da gran ventaja al usuario en no depender de ninguna máquina para disfrutar de este fabuloso accesorio. Este sistema desarrollado es código libre y se encuentra alojado en github para su uso.

Referencias

- <http://panamahitek.com/>
- <http://proyectos-sobre-arduino.blogspot.com/>
- <http://playground.arduino.cc/Es/Projects>
- <http://www.taringa.net/post/hazlo-tu-mismo/14826557/Proyectos-de-electronica-programable-para-Arduino.html>
- <http://www.cyberhades.com/2010/01/25/top-40-proyectos-para-arduino-en-la-web/>
- <http://makezine.com/projects/android-arduino-led-strip-lights/>
- <http://www.instructables.com/id/How-to-use-a-Buzzer-Arduino-Tutorial/step2/The-Circuit/>
- <http://forum.arduino.cc/index.php?topic=103188.0>
- https://es.wikipedia.org/wiki/Modulaci%C3%B3n_por_ancho_de_pulsos

