

**MATERIA:**

Sistemas Programables

**CARRERA:**

Ingeniería en Sistemas Computacionales



**PRESENTA:**

Pablo Cesar Velázquez Espinoza

**NOMBRE DEL MAESTRA:**

Ing. Levy Rojas Carlos Rafael

**PRACTICA:**

Semáforo con Arduino

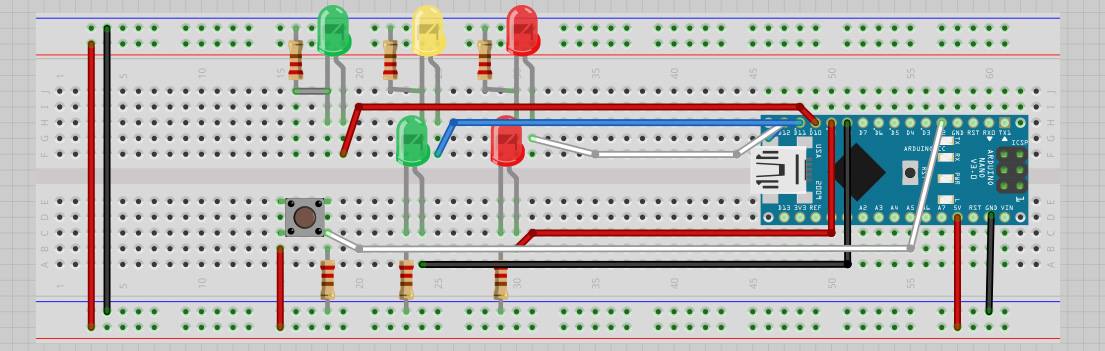
**LEÓN, GUANAJUATO Periodo: Enero – Junio 2018**

**INTRODUCCIÓN**

Con esta práctica ilustraremos cómo debe estructurar un programa en Arduino, también veremos cómo son las sentencias básicas usadas con esta IDE, cómo cargar nuestro programa a la placa y para finalizar, realizaremos un ejemplo con el que encenderemos leds con Arduino montando nuestro propio semáforo basado en el funcionamiento del Reino Unido.

Un código Arduino es una serie de comandos de programación que le dirán a nuestro microcontrolador como configurarse al iniciarse y qué acciones tiene que realizar mientras esté en funcionamiento. Estos comandos utilizados en Arduino son sentencias muy fáciles e intuitivas.

**DIAGRAMA DE CONFIGURACIÓN**

****

**CÓDIGO FUENTE**

/\*

\*Instituto Tecnologíco de León

\*Last Modification: February 14th, 2018

\*Objective of the program: Simulate the interaction of the functionality of a traffic light that regulates the traffic of vehicles where,

the Green light indicates when the vehicles must cross, the Yellow light indicates that the vehicles must decrease the speeD to stop and

the Red light indicates to the vehicles that must be in total height. The sequencing of the lights will be based on the system of the

United Kingdom, where first goes from Green to Yellow and then to Red, after a certain time the Red light goes to the Yellow and finally

to Green where the vehicles can circulate again.

On the other hand, also the program simulates a pedestrian traffic light, where the Green light indicates the pedestrian who can cross

the street and Red light that waits for the vehicles to stop. When the red light of the traffic lights of the vehicles is Green, the light

Red of the pedestrians traffic light must be in Red, and when the traffic light of the vehicles is Red, the Green light the pedestrian

traffic light must be on.

\*/

const int CAR\_RED = 12; //red light of vehicular traffic initialized in 12

const int CAR\_YELLOW = 11; //yellow light of vehicular traffic initialized in 11

const int CAR\_GREEN = 10; //green light of vehicular traffic initialized in 10

const int PED\_RED = 9; // red light of the pedestrian traffic initialized in 9

const int PED\_GREEN = 8; // green light of the pedestrian traffic initialized in 8

const int BUTTON = 2; // button pin initialized in 2

int crossTime = 10000; //time alloyoud to cross

unsigned long changeTime; //time since BUTTON pressed

int state = LOW; //if is push button

/\*

The method setup() configures the pin of the lights of the traffic lights to behave like an exit with the method pinMode()

Also the method digitalWrite() Write a HIGH or a LOW value to a digital pin.

\*/

void setup() {

pinMode(CAR\_RED, OUTPUT);

pinMode(CAR\_YELLOW, OUTPUT);

pinMode(CAR\_GREEN, OUTPUT);

pinMode(PED\_RED, OUTPUT);

pinMode(PED\_GREEN, OUTPUT);

pinMode(BUTTON, INPUT); // button on pin 2

digitalWrite(CAR\_GREEN, HIGH);

digitalWrite(PED\_RED, HIGH);

}

/\*

The method loop()has two conditions, one that verifies if the button is pressed, if it is true the variable "state" takes the value of "high",

the second validation is responsible for verifying if the variable "satate" has the value of "high" and if the time that has passed since

the button was pressed in greater than 10 seconds

\*/

void loop() {

if (digitalRead(BUTTON)) //Change of state is button is pressed

state = HIGH;

if (state == HIGH && (millis() - changeTime) > crossTime) {

changeLight();

state = LOW;

}

}

/\*

The mothod changeLight is responsible for changing the lights of the traffic lights with the digitalWrite() method

\*/

void changeLight() {

digitalWrite(CAR\_GREEN, LOW); //green off

digitalWrite(CAR\_YELLOW, HIGH); //yellow on

delay(2000); //wait 2 seconds

digitalWrite(CAR\_YELLOW, LOW); //green off

digitalWrite(CAR\_RED, HIGH); //yellow on

delay(2000); //wait 2 seconds

digitalWrite(PED\_RED, LOW); //green off

digitalWrite(PED\_GREEN, HIGH); //yellow on

delay(crossTime); //wait for

for (int i = 0; i < 10; i++) {

digitalWrite(PED\_RED, HIGH); //Turn on red pedestrian traffic light

delay(250);//wait 250 milliseconds

digitalWrite(PED\_GREEN, LOW);//Turn off green pedestrian traffic light

delay(250);//wait 250 milliseconds

}

digitalWrite(PED\_RED, HIGH);//Turn on red pedestrian traffic light

delay(500);//wait 500 milliseconds

digitalWrite(CAR\_YELLOW, HIGH);//turn on the yellow light of the traffic lights of the vehicles

digitalWrite(CAR\_RED, LOW);//turn on the red light of the traffic lights of the vehicles

delay(1000);//wait 500 milliseconds

digitalWrite(CAR\_GREEN, HIGH);//turn on the green light of the traffic lights of the vehicles

digitalWrite(CAR\_YELLOW, LOW);//turn on the yellow light of the traffic lights of the vehicles

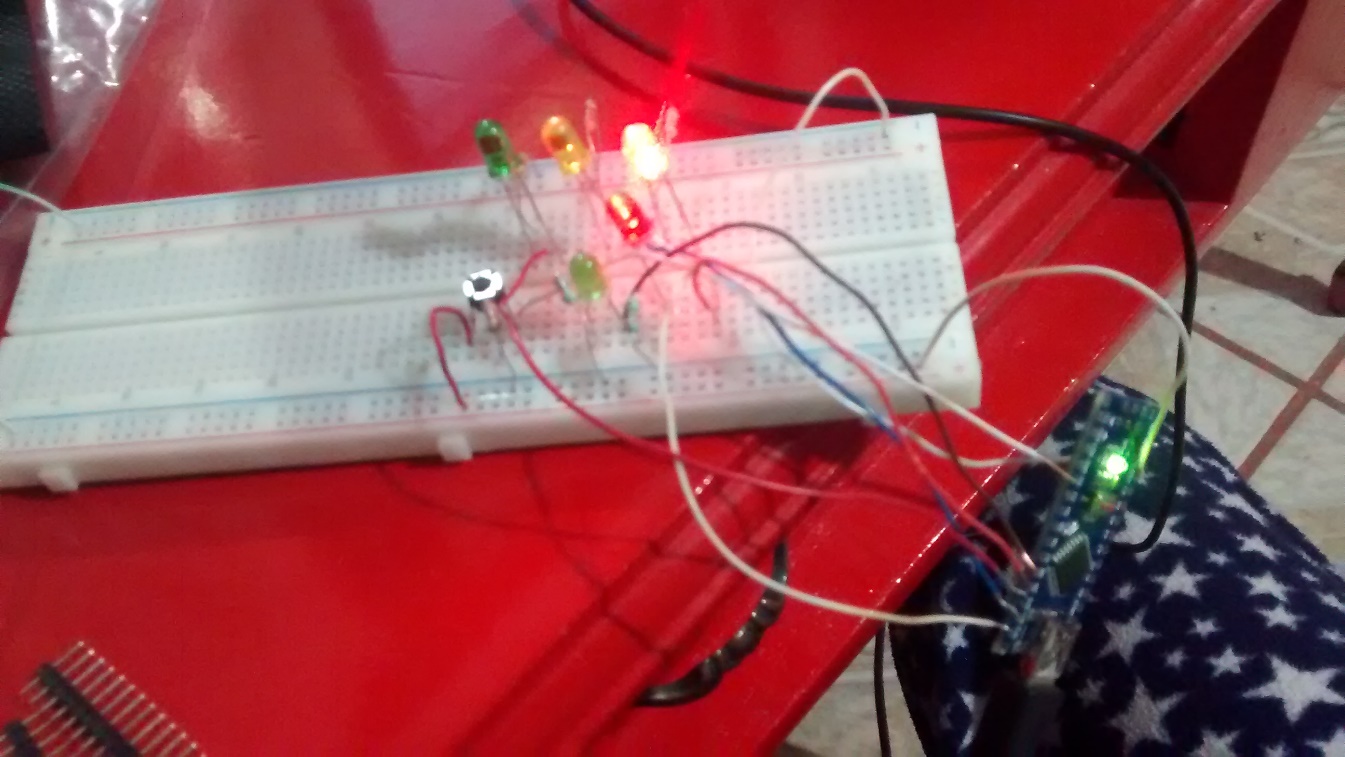
//record the time size last change of lights

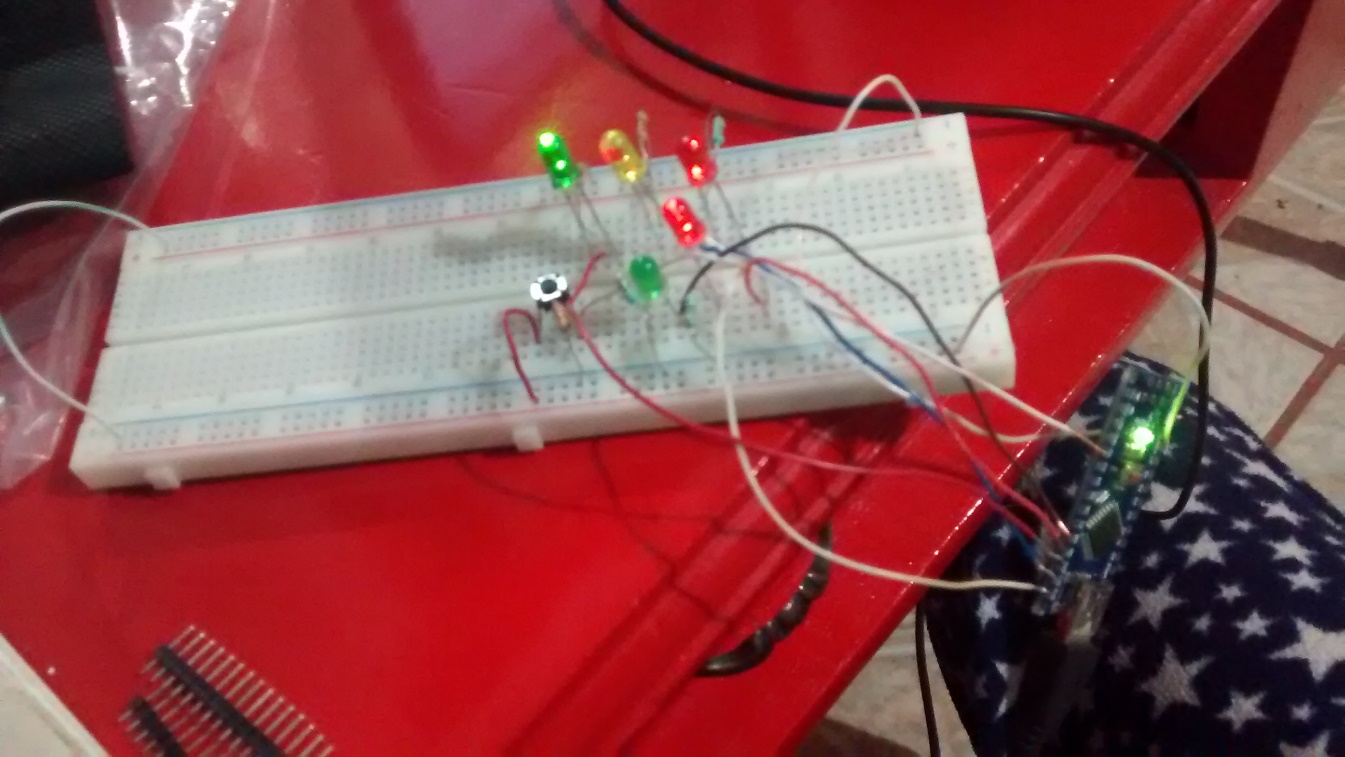
changeTime = millis();

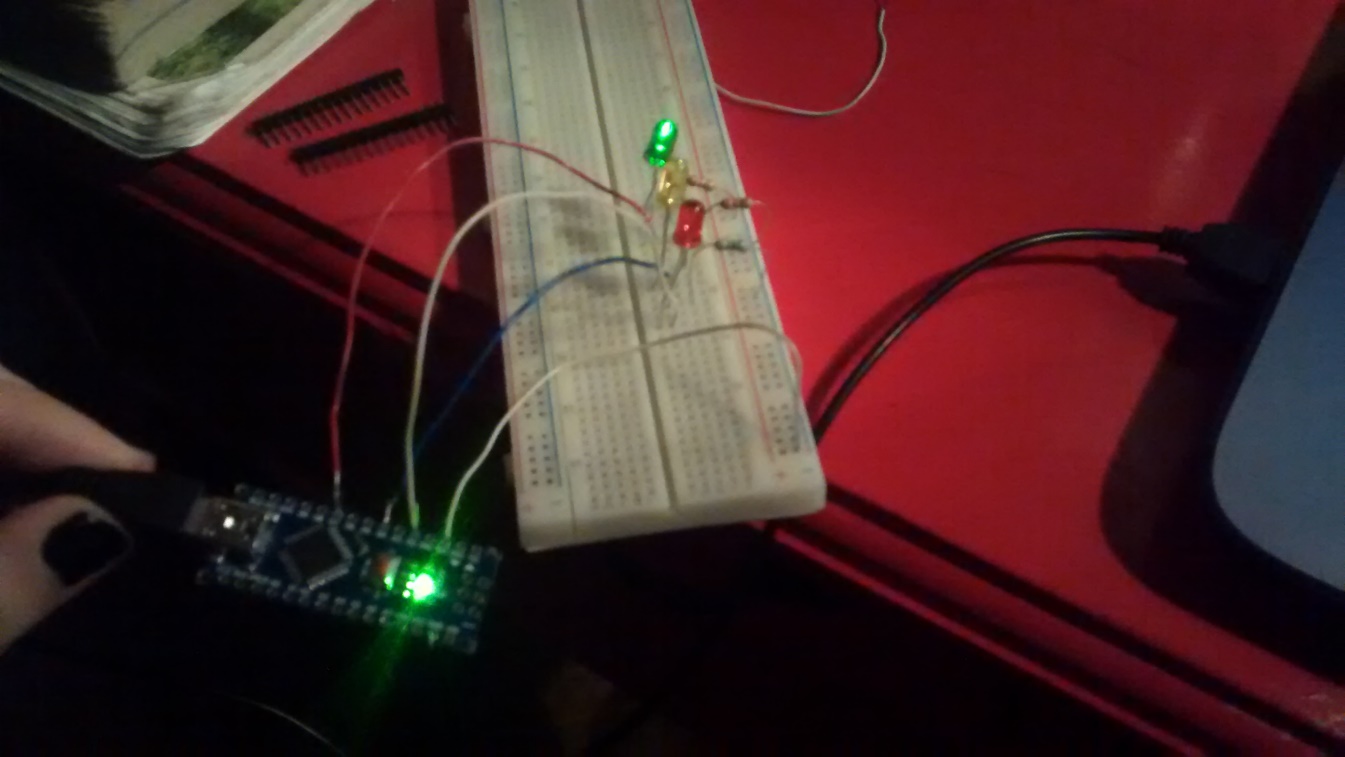
//then return to the my program loop

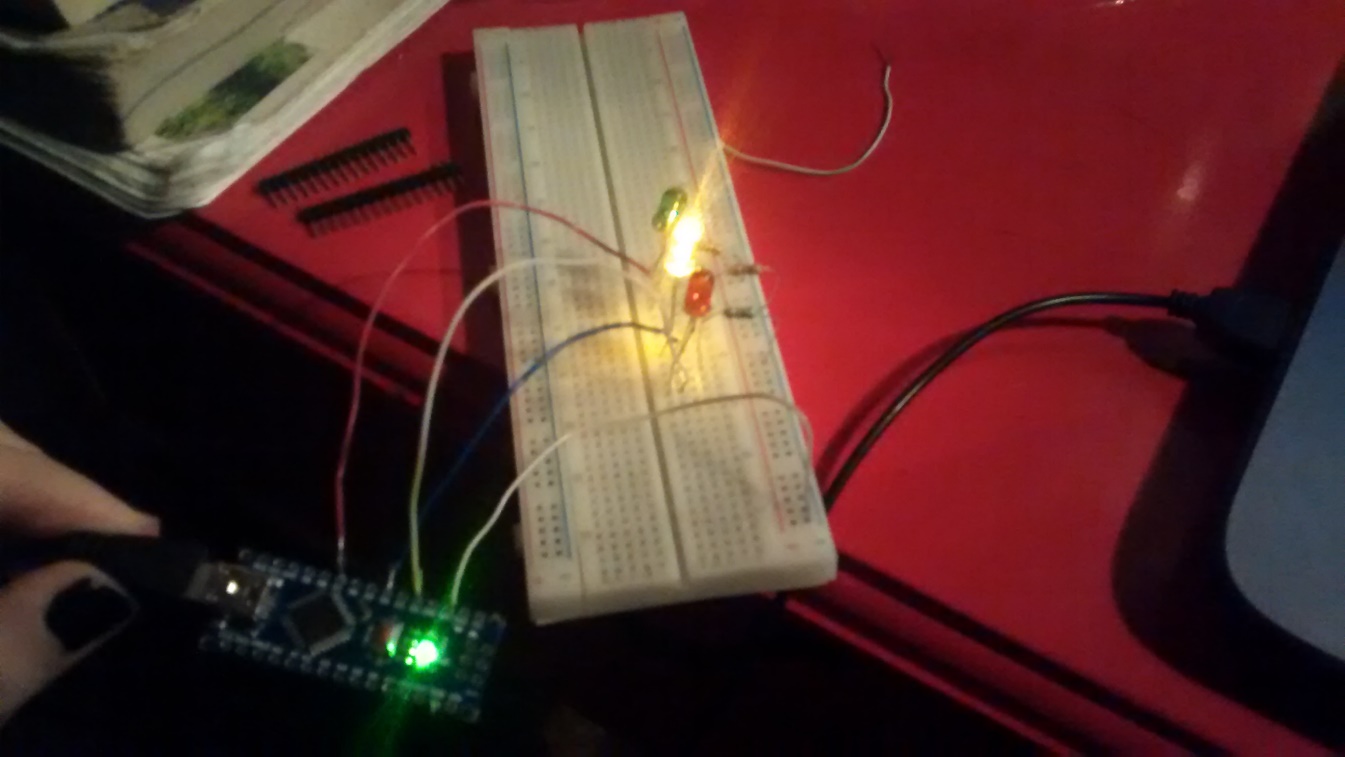
}

**RESULTADOS**

****

****

****

****

**CONCLUSIÓN**

En esta práctica pudimos implementar un semáforo vehicular y peatonal en base a la funcionalidad que se utiliza en el Reino Unido utilizando el ambiente de desarrollo de Arduino, el cual contiene instrucciones muy simples de utilizar y que nos permiten programar la funcionalidad en base a las necesidades que ocupemos. Al terminar de programar las instrucciones necesarias en el Arduino pudimos ensamblarlo en el Protoboard y probar que funciona el semáforo.