Traffic Lights with Arduino

1. INTRODUCCIÓN

En este Proyecto vamos a simular el funcionamiento de un semáforo para vehículos de forma automatizada por la acción de un usuario o peatón a través de la acción de un pulsador.

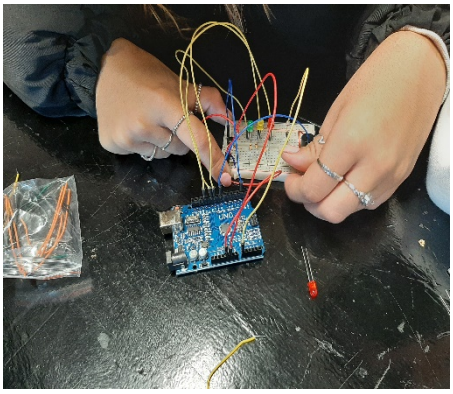
El principal motivo de la realización del presente Proyecto radica en el profundo interés en las capacidades de Arduino como herramienta de control electrónico.

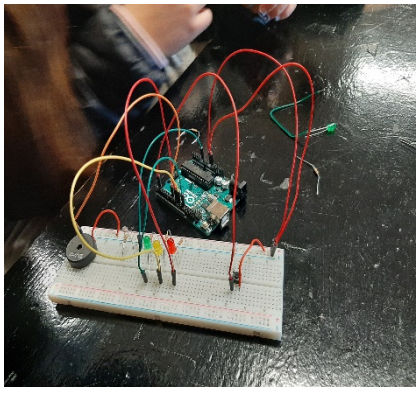
El manejo de Arduino implica una gran reducción de tiempo en diseño electrónico y en la programación de su microcontrolador.

Sus características hacen de él una oportunidad de negocio en cuanto a su implementación en los sectores de la automatización y las comunicaciones industriales, a fin de una futura dedicación laboral en el marco de estos sectores industriales.

2. OBJETIVOS:

- El principal objetivo de este proyecto es la familiarización con arduino
- Realizar conexiones de componentes físicos en placa.
- Programar autómatas digitales: Implementación de arduino en el control de grupos de semáforos que ejemplifique un sistema semafórico real.



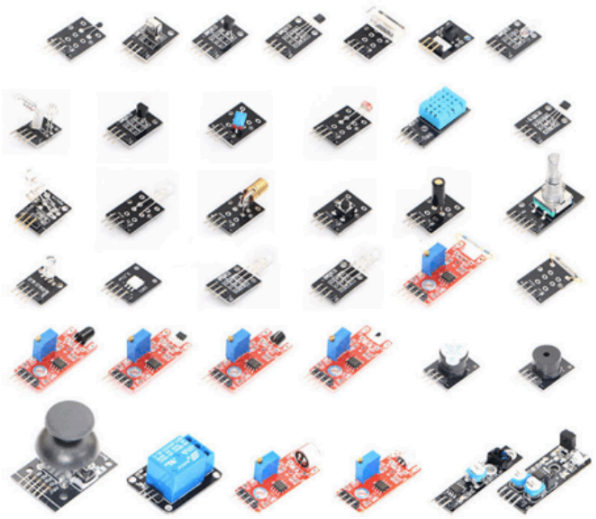


3. COMPONENTES

- Placa Arduino UNO
- Breadboard PCB
- 5 LEDs colores
- 1 Pulsador
- 1 Timbre o buzzer
- 1 Sensor Fotoluminoso
- 7 Resistencias de 220Ω
- 12 cables dupont

CALCULADORA RESISTENCIA:

https://www.digikey.es/es/resources/conversion-calculators/conversion-calculator-resistor-color-code



4. CONOCEMOS LA PLACA ARDUINO

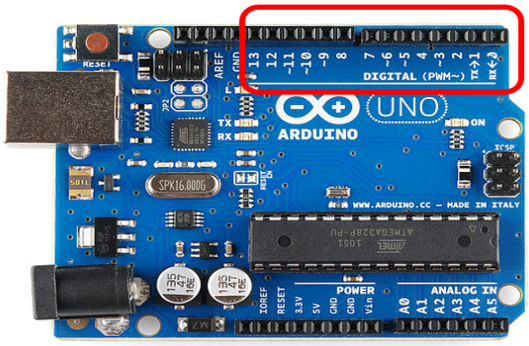
En este proyecto vamos a trabajar con la placa Arduino uno, concretamente la R3. Las características principales de esta placa es que tienen por un lado energía o power, por otro pines analógicos y finalmente 13 pines digitales. Destacar que la energía tendrá valores de 5v y de 3,3v, además de su correspondiente conexión GND o Masa.

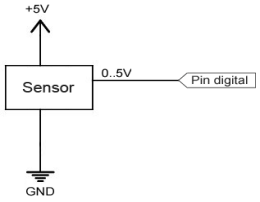
Hardware:

PIN DIGITALES EN ARDUINO

Una de las funciones más interesantes (si no la más) de Arduino es generar autómatas. Esta interacción se lleva a cabo en gran parte mediante el uso de las entradas y salidas.

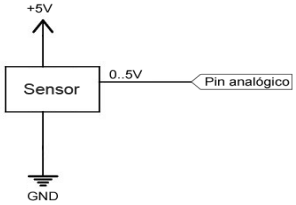
Una señal digital es una variación de voltaje entre -Vcc a +Vcc sin pasar por los valores intermedios. Por lo tanto, una señal digital dispone solo de dos estados, activado y desactivado





PIN ANALÓGICOS EN ARDUINO

Las entradas analógicas funcionan de una forma similar a las entradas digitales, por lo que en la práctica el montaje y código final son muy similares. Una señal analógica es una magnitud que puede tomar cualquier valor dentro de un intervalo –Vcc y + Vcc. Por ejemplo, una señal analógica de tensión entre 0V y 5V podría valer 2,72V, o cualquier otro valor con cualquier número de decimales.



GND: DAR CORRIENTE DE FORMA CONTÍNUA

Software

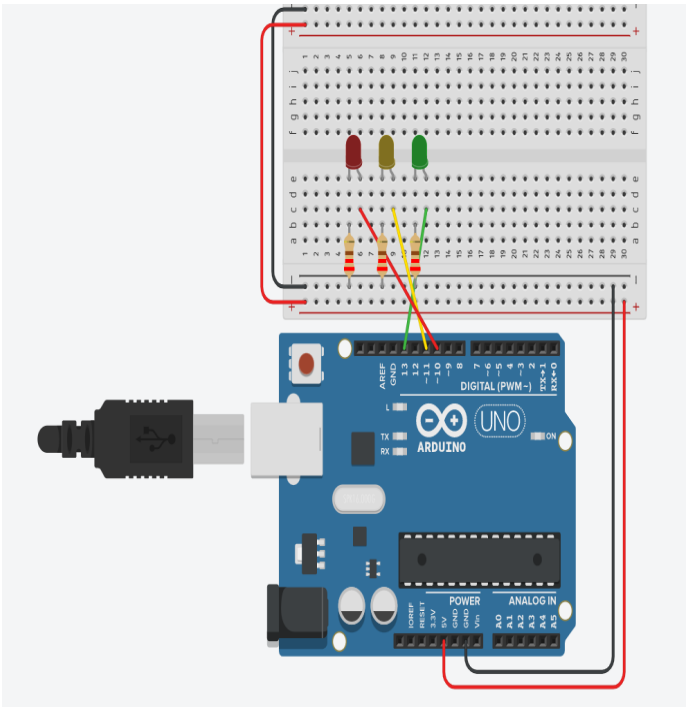
Ahora configuraremos las dos funciones principales de Arduino que son Setup y Loop.

La función Setup será la encargada de inicializar los parámetros y pines con la configuración predeterminada. La función Loop Esta función es la que se va a repetir indefinidamente. Aquí comprobaremos el estado de los pulsadores, encenderemos y apagaremos los múltiples LEDs según corresponda.

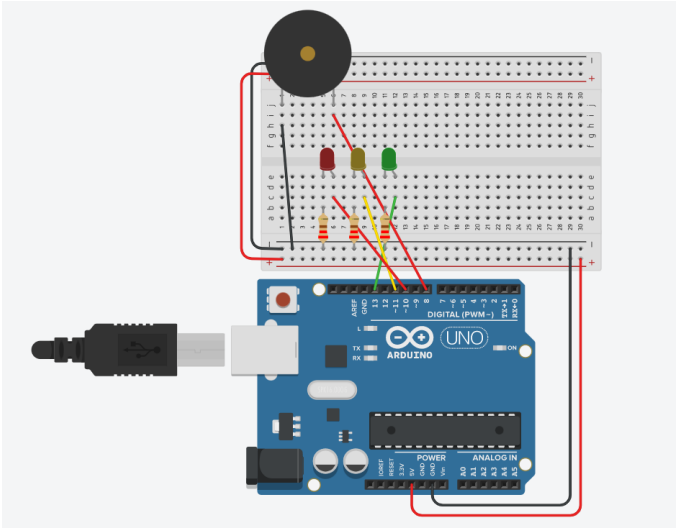
5. FASE MECÁNICA DEL PROYECTO

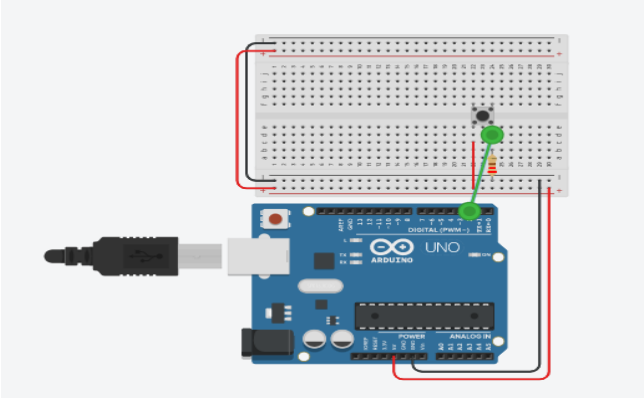
- Creación de Grafcet como autoayuda para clarificar los pasos de montaje del proyecto.
- Conexión y programación de 3 LEDs.
- Conexión de Buzzer e implementación.
- Montaje de un pulsador como pin de entrada.
- Sensor Futoluminoso capaz de detectar intensidad lumínica.

Comenzamos nuestro proyecto con el montaje de 5 **LEDs** en nuestra placa PCB. Se usarán 3 LEDs (Rojo, amarillo y verde) para simular el montaje de un semáforo para coches y por otro lado usaremos 2 LEDs (Rojo y azul) para simular Para ello conectaremos los extremos del LED (ánodo y cátodo) a cada uno de los extremos del PCB. El ánodo o parte positiva del diodo irá conectada a un pin digital. Se debe conectar a un pin digital ya que el valor de este LED en todo momento será activado o desactivado (es decir 0 o 1). El otro extremo del LED irá conectado a masa o GND, aunque previamente debe tener conectada una resistencia de 220Ω. De forma análoga se realizará la conexión de los otros dos LED, cada uno a un pin diferente. Concretamente en nuestro proyecto usaremos el pin 13, 11, 10 para los LED verde, amarillo y rojo. Y usaremos los pin 7 y 5 (led azul y led rojo).



A continuación montamos en nuestra placa el **Buzzer**, la conexión será de forma análoga a los diodos ya que el buzzer dispone de dos polos (positivo y negativo). El polo positivo del dispositivo irá conectado a un pin digital de arduino, concretamente usaremos el pin 8. Para el buzzer no usaremos resistencia para dejar que funcione en más frecuencias al no perder potencia eléctrica.

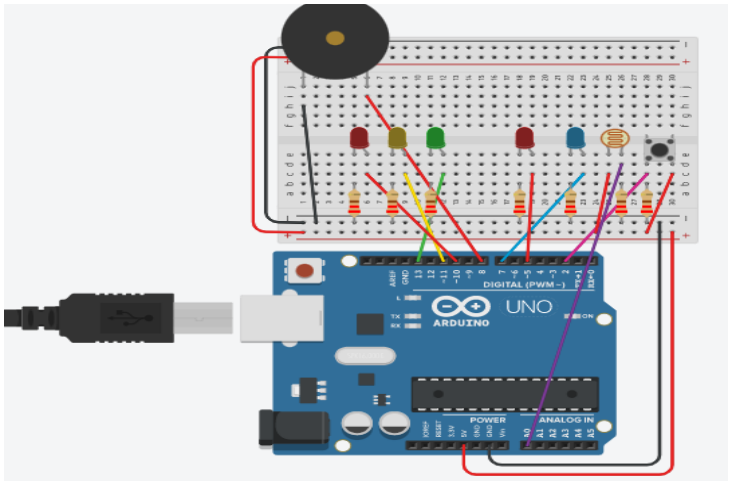




Seguidamente montaremos en la placa PCB el **pulsador.** Recordamos que el pulsador tiene dos estados, (abierto o cerrado). Nuestro objetivo será conectar el pulsador directamente a 5V y GND al otro extremos como podemos ver en el siguiente esquema, aunque recordamos la utilización de una resistencia de 220Ω para regular en todo momento el voltaje que pasa por el circuito. En el pin del PCB, como si de un polímetro se tratase, conectaremos un cable hasta el pin digital 2. Este pin será configurado como entrada ya que necesitamos captar en qué momento ha sido accionado el pulsador. Hemos usado un pin digital ya que solo necesitamos conocer si el pulsador ha sido accionado o no pero en ningún caso controlar con qué potencia o fuerza se ha accionado.

Con este esquema tendríamos funcionando un semáforo capaz de ser autómata con la acción de un usuario o peatón. Dada la situación actual que vivimos de COVID, nos ha parecido interesante una mejora a este proyecto. Concretamente, para evitar que el usuario tenga que accionar el pulsador de forma física, a nuestra maqueta le hemos añadido un sensor o resistencia fotoluminosa. Al tratarse de un diodo conectamos de forma análoga a los LED con una resistencia de 220 Ω. Este sensor irá conectado al pin Analógico 0, ya que en este caso necesitamos conocer los valores concretos que está recibiendo el sensor.

A continuación mostraremos la imagen final o grafcet del circuito.



6. FASE LÓGICA DEL PROYECTO: PROGRAMACIÓN

Como se ha comentado en secciones anteriores, vamos a construir dos semáforos (uno para coches y otro peatonal) los cuales serán autómatas. El semáforo de los coches comenzará en verde y los led amarrillo y rojo apagados. Por otro lado, el semáforo de los peatones comenzará con el led rojo encendido y el azul apagado.

Tras el accionamiento del pulsador o la recepción de la resistencia fotoluminosa de un valor elevador de luz (más adelante concretaremos el valor) nuestro proyecto realizará la siguiente secuencia:

- El semáforo de vehículos:
  - Apaga el led verde y enciende el amarillo durante 1,25s.
  - Apaga el led amarillo y enciende el rojo por
- El semáforo peatonal :
  - Apaga el LED rojo y activa el Azul durante
- Buzzer
  - Durante el tiempo que el semáforo de vehículos está en rojo, el buzzer emitirá un sonido intermitente indicando a los peatones que pueden cruzar. El sonido intermitente aumentará la frecuencia de repetición progresivamente, indicando que finaliza el tiempo de encendido.

Para la programación de LED, en el void setup () activaremos los pin digitales a través de la función PinMode(“número de pin”, OUTPUT). En nuestro caso todos los LED serán elementos de salida y pondremos todos apagados excepto el led verde del semáforo de vehículos y el rojo del semáforo peatonal. Para ello usaremos la función digitalWrite (“PIN”, HIGH). Finalizaremos esta función dejando los otros pines activados, es decir el pin del buzzer como dispositivo de entrada, mientras que el pulsador y el sensor de luz serán elementos de entrada. pinMode(“número de pin”, INPUT).

En la función void loop(), vamos a realizar la función principal del proyecto. En nuestro caso haremos la lectura del sensor y del pulsador. En el pulsador se hará lectura con la función digitalRead(“número de pin”) ya que se trata de un pin digital, por otro lado para el sensor de luz realizaremos una llamada la función analogRead(“número de pin analógico). La función estará compuesta por un condicional (IF / ELSE). En el IF usaremos un condicional compuesto, en este caso si el pulsador está desactivado y el sensor de luz tiene un valor menor a los parámetros determinados), el semáforo está en estado inicial. En otro caso (else) se activará secuencialmente lo pasos descritos en el proceso anterior.

El sensor de luz estará condicionado a la situación luminosa de la sala o medio en el que se encuentre el proyecto. Para ello, previamente añadiremos en el loop la función serial.print(), de esta manera en el puerto serie imprimiremos los valores que está recibiendo el sensor en el ambiente donde lo vamos a utilizar. Una vez observados dichos parámetros si comprobamos por ejemplo que no son superiores a 100, bastará añadir en el condicional anterior que el parámetro determinado no sea superior a 130 (hay que dejar un margen superior para asegurar que ninguna luz indeseada active el semáforo).

CODIGO DE PROGRAMACIÓN:

```
int GREENCAR = 13;
int YELLOWCAR = 11;
int REDCAR = 10;
int BLUEPERSON = 7;
int REDPERSON = 5;
int BUZZER = 8;
int BUTTON = 2;
int LIGHTSENSOR= A0;
int PUSH;
int PASSLIGHT;
void setup() {
  // put your setup code here, to run once:
 pinMode (GREENCAR, OUTPUT);
 digitalWrite (GREENCAR, HIGH);
 pinMode (YELLOWCAR, OUTPUT);
 digitalWrite (YELLOWCAR, OUTPUT);
 pinMode (REDCAR, OUTPUT);
 digitalWrite (REDCAR, LOW);
 pinMode (BLUEPERSON,OUTPUT);
 digitalWrite (BLUEPERSON, LOW);
 pinMode (REDPERSON, OUTPUT);
 digitalWrite (REDPERSON, LOW);
 pinMode (BUZZER,OUTPUT);
 pinMode(BUTTON, INPUT);
 Serial.begin(9600);

}

void loop() {
  // put your main code here, to run repeatedly:
  PUSH = digitalRead(BUTTON);
  PASSLIGHT = analogRead (A0);
```

```
  Serial.print("el valor de luz es "); // Para imprimir
  Serial.print(PASSLIGHT); //
  Serial.print("\n"); //
  if (PUSH == LOW && PASSLIGHT<100){
  digitalWrite(GREENCAR, HIGH);
  digitalWrite(REDCAR, LOW);
  digitalWrite(YELLOWCAR, LOW);
  digitalWrite(REDPERSON, HIGH);
  digitalWrite(BLUEPERSON, LOW);
}else{
  digitalWrite(YELLOWCAR, HIGH);
  digitalWrite(GREENCAR, LOW);
  digitalWrite(REDCAR, LOW);
  digitalWrite(BLUEPERSON, LOW);
  digitalWrite(REDPERSON, HIGH);
delay(1250);
digitalWrite(YELLOWCAR,LOW);
digitalWrite(GREENCAR, LOW);
digitalWrite(REDCAR, HIGH);
digitalWrite(BLUEPERSON, HIGH);
digitalWrite(REDPERSON, LOW);
tone(BUZZER,234);
delay(1000);
noTone(BUZZER);
delay(1000);
tone(BUZZER,234);
delay(700);
noTone(BUZZER);
delay(700);
tone(BUZZER,234);
delay(500);
noTone(BUZZER);
delay(500);
tone(BUZZER,234);
```

```
delay(300);
noTone(BUZZER);
delay(300);
tone(BUZZER,234);
delay(100);
noTone(BUZZER);
delay(100);
tone(BUZZER,234);
delay(100);
noTone(BUZZER);
delay(80);
tone(BUZZER,234);
delay(50);
noTone(BUZZER);
delay(30);
tone(BUZZER,234);
delay(4000);
noTone(BUZZER);
digitalWrite(REDCAR,HIGH);
digitalWrite(YELLOWCAR,LOW);
digitalWrite(GREENCAR,LOW);
digitalWrite(BLUEPERSON,LOW);
digitalWrite(REDPERSON,LOW);
delay(3000);



  }

}
```