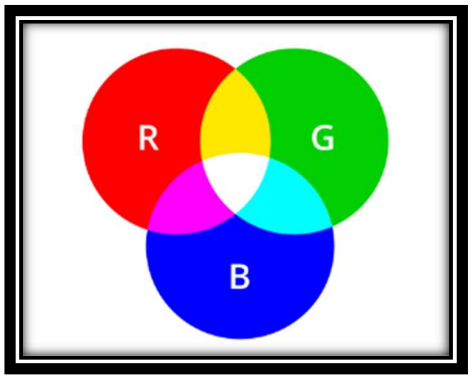
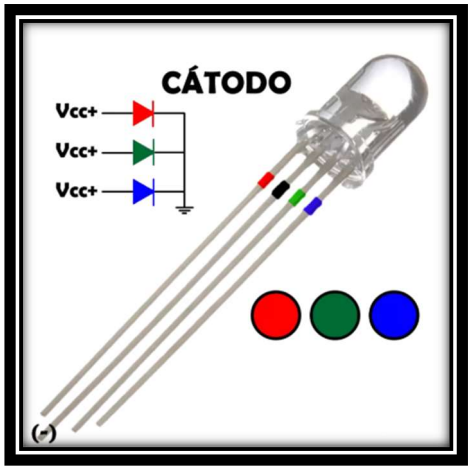
TECNOLOGÍA Y EDUCACIÓN PLÁSTICA

TEORÍA DEL COLOR Y ARDUINO

TEORÍA DEL COLOR



LED RGB



GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN, FORMACIÓN PROFESIONAL Y DEPORTES
Código Escuela 4.0
Junta de Castilla y León
Consejería de Educación
Programa financiado por el Ministerio de Educación, Formación Profesional y Deportes

ÌNDICE

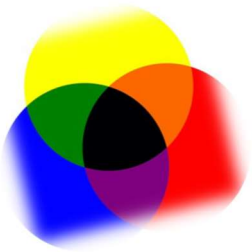
1 INTRODUCCIÓN

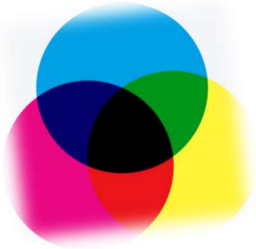
¿Te gusta el color?



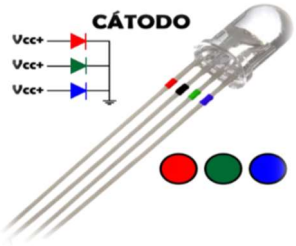

¿Sabes cuantos colores podemos crear con un solo led ? 16,777,216

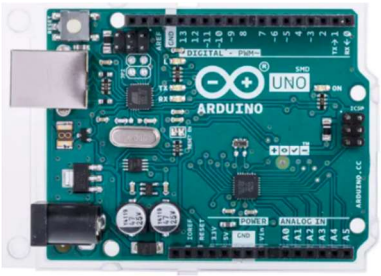
En este documento vamos a descubrir a traves de las posibilidades que nos ofrece el pensamiento computacional como se forman los colores y el modelo de color que utilizan tanto monitores como pantallas para mostrar la información.

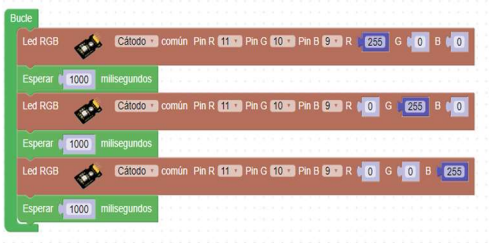




Mediante un LED RBG y otros componentes electrónicos recorreremos un camino que nos ayudará a entender como funciona el color y pensamiento computacional.









2 TEORÍA DEL COLOR

2.1 HISTORIA DE LA TEORÍA DEL COLOR

Isaac Newton describió en *Óptica, o tratado de las reflexiones, refracciones, inflexiones y colores de la luz,* la descomposición del espectro visible en los colores que lo integran. En esta obra, propuso que los colores primarios o puros eran el rojo, el amarillo y el azul, y que todos los demás colores se creaban a partir de la mezcla entre ellos. La propuesta de Newton dio lugar al modelo RYB, que se emplea en las artes plásticas.

En 1810, el escritor, filósofo y naturalista Johann Wolfgang von Goethe publicó un tratado llamado *Teoría de los colores*. La obra se fundamentaba en los principios que había desarrollado Newton y reflexiona sobre la naturaleza de los colores y la forma en que los percibimos. Goethe propuso un círculo cromático con el rojo, el amarillo y el azul como colores primarios, y el verde, el naranja y el violeta como colores secundarios.

Más adelante, a principios del siglo XX, el químico y filósofo alemán Wilhelm Ostwald formuló una teoría que describía el color a partir de su percepción psicológica. A diferencia de otras teorías del color, esta teoría se basa en la experiencia subjetiva y clasifica los colores en colores cálidos y colores fríos.

Ostwald propuso un círculo compuesto por 24 tonos y ubicó los colores cálidos (del rojo al amarillo) en la parte izquierda y los colores fríos (del azul al violeta) en la parte derecha. Esta clasificación es aún empleada en muchas disciplinas relacionadas con la psicología o el marketing.

El modelo de color RGB se basa en la teoría de Young-Helmholtz de la visión tricromática del color, desarrollada por Thomas Young y Hermann von Helmholtz a principios y mediados del siglo XIX, y en el triángulo cromático de James Clerk Maxwell.

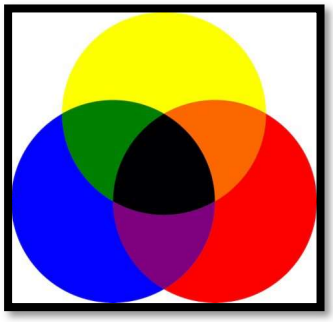
Los primeros experimentos con RGB en las primeras fotografías en color los realizó el Maxwell en 1861 . Combinó tres tomas separadas con filtros de color con el fin de reproducir una fotografía en color, fueron necesarias tres proyecciones coincidentes sobre una pantalla en un cuarto oscuro.El modelo RGB se utilizó en las placas de color Autochrome Lumière y otras tecnologías, como la pantalla de color Joly y el proceso de Paget a principios del siglo XX.

Cuando se empleó la reproducción de las impresiones de fotografías de tres placas se realizó mediante tintes o pigmentos utilizando el modelo CMY complementario, se utilizó las placas negativas de las tomas filtradas: el rojo inverso da la placa cian, y así sucesivamente.

El principal elemento de la teoría del color es el círculo cromático. Se trata de una representación gráfica circular que despliega los colores del espectro en una secuencia organizada, basada en las relaciones entre ellos.El círculo cromático se ordena de manera que los colores complementarios se enfrentan y los colores análogos se ubican uno al lado del otro.

A continuación mostramos los círculos cromáticos de los principales modelos.

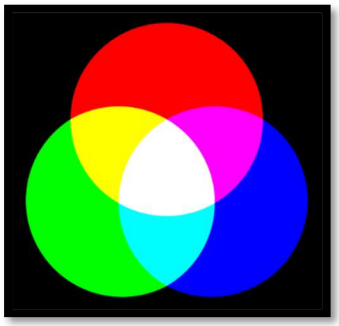
2.1.1 *Modelo de color RYB*



El modelo de color RYB (*red, yellow, blue* o rojo, amarillo, azul) se ha utilizado desde el siglo XVIII para mezclar y crear colores. Fue propuesto por Isaac Newton a partir de la idea de que **tres colores primarios (el rojo, el amarillo y el azul)** podían mezclarse para producir todos los colores.

Es un **modelo sustractivo**: a medida que se aplican pigmentos o tintas, la luz es menor y se llega al negro.

2.1.2 *Modelo de color RGB*

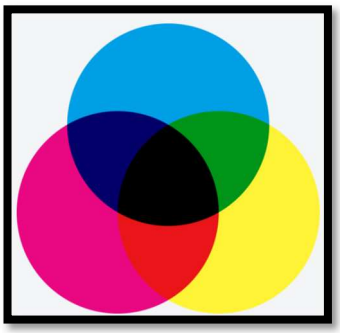


Colores primarios del modelo RGB: **rojo, verde y azul** *(red, green, blue)*, a partir de los cuales se compone el resto.

Es un **sistema de color aditivo**: cada combinación de color añade luz hasta llegar al blanco, que combina todos los colores.

Este sistema se usa en dispositivos como monitores o smartphone.

2.1.3 *Modelo de color CMYK*



Colores primarios del modelo CMYK**: cian, magenta, amarillo y negro** (el negro se designa con una K (key plate).

Este es un **modelo sustractivo**, que compone el color a partir de la absorción de luz, de modo que la mezcla de colores tiende al negro, que es la ausencia total de luz.

Se emplea para técnicas de impresión, ya que el papel no tiene las propiedades lumínicas de los monitores.

2.2 PROPIEDADES DEL COLOR

Las principales propiedades del color son:

- **Matiz o tono.** es la característica esencial,es decir, permite distinguir un color de otro. El tono o matiz está presente en el círculo cromático y lleva el nombre del color mismo. Por ejemplo: rojo, azul, violeta.
- **Luminosidad.** se trata de la cantidad de luz presente en el color, su nivel de claridad (cercanía al blanco) u oscuridad (cercanía al negro).
- **Saturación.** Es la intensidad o pureza del color. Se refiere a la cantidad de gris con que se mezcla el color. Cuanto más gris menos puro será el color y menor será su saturación.

3 LED RGB

¿Qué es un LED RGB?

El LED RGB es un término popular en el mundo de la tecnología y el diseño. Como hemos visto antes RGB significa Rojo, Verde y Azul, los colores primarios de la luz en el modelo RBG.

Un LED RGB es un tipo de diodo emisor de luz que combina estos tres colores. Esta tecnología se utiliza ampliamente en diversas aplicaciones, incluyendo pantallas, monitores y smartphone.

Comprendiendo como trabaja un LED RGB conoceremos un mundo de colores vibrantes y personalizables para proyectos audiovisuales, tecnológicos y de diseño.

Un LED RGB, por lo tanto, es un diodo emisor de luz que combina estos tres colores para producir más de 16 millones de tonos de luz. Esto es posible porque cada uno de los colores puede tener un nivel de brillo que va de 0 a 255, ofreciendo un amplio espectro de posibilidades de color.

¿Qué modelo de color utiliza el LED RGB?

El concepto de LED RGB está basado en el modelo de color aditivo, donde diferentes longitudes de onda de luz se suman para crear diferentes colores. Esto lo diferencia a los modelos RYB y CYMK de color sustractivo en los colores se crean absorbiendo o sustrayendo ciertas longitudes de onda de luz mientras se reflejan otras.

La tecnología LED RGB ha revolucionando la manera en que percibimos el color en las pantallas digitales. La ciencia detrás de esta tecnología es fascinante e involucra la combinación de luz roja, verde y azul en intensidades variables para producir un amplio espectro de colores.

Este proceso se conoce como mezcla de colores aditiva, donde los tres colores primarios de la luz se combinan de diferentes maneras para crear colores secundarios. Cuando los tres colores se combinan en su máxima intensidad, se produce luz blanca.

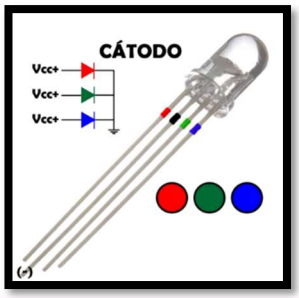
Una de las cosas más interesantes del LED RGB reside en su capacidad para producir negro que se logra al apagar completamente píxeles individuales.

4 CONTROL PROGRAMADO DE LED RGB

4.1 PRÁCTICA1.CONTROL PROGRAMADO DE UN LED RGB CON ARDUINOBLOCKS

La programación nos permite controlar el LED RBG, variando valores en la programación podemos conseguir que el LED RBG muestre cualquier de la gama de colores del modelo RBG. En esta práctica nos comenzaremos programando para mostrar los colores primarios del modelo para luego crear otros colores combinándolos.

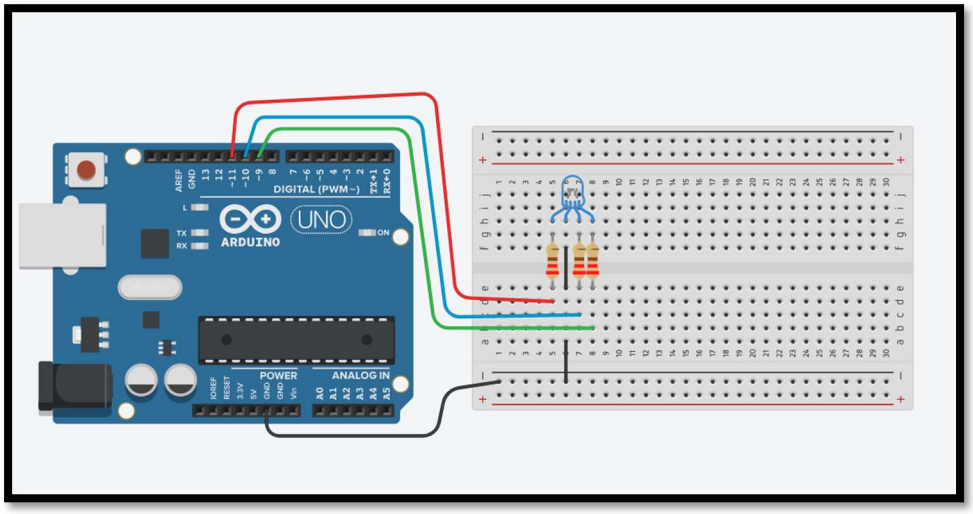
4.1.1 Materiales.



- 1 Placa de Arduino UNO.
- 1 Protoboard y latiguillos.
- 1 Diodo Led RGB (ánodo común).
- Resistencias de 220Ω.

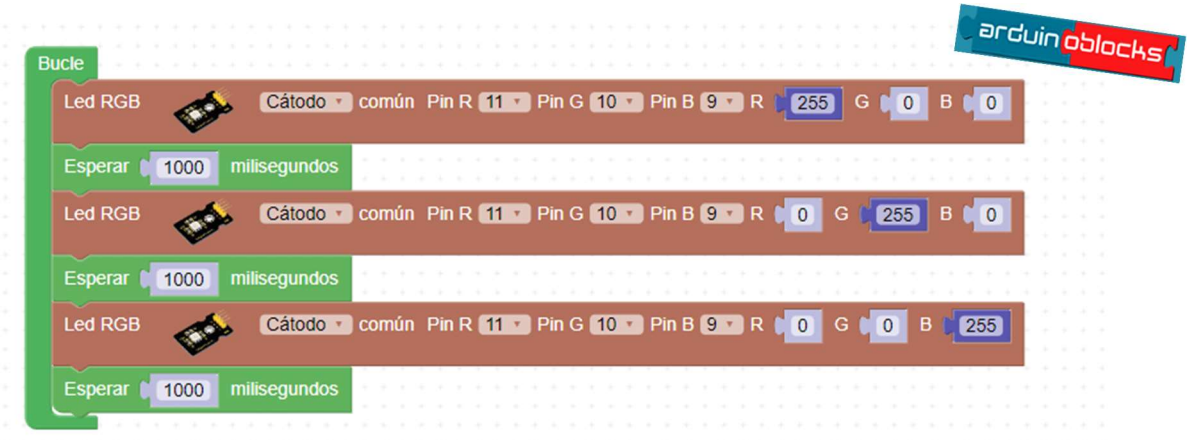
4.1.2 Montaje del circuito.

Esquema de montaje y de circuito

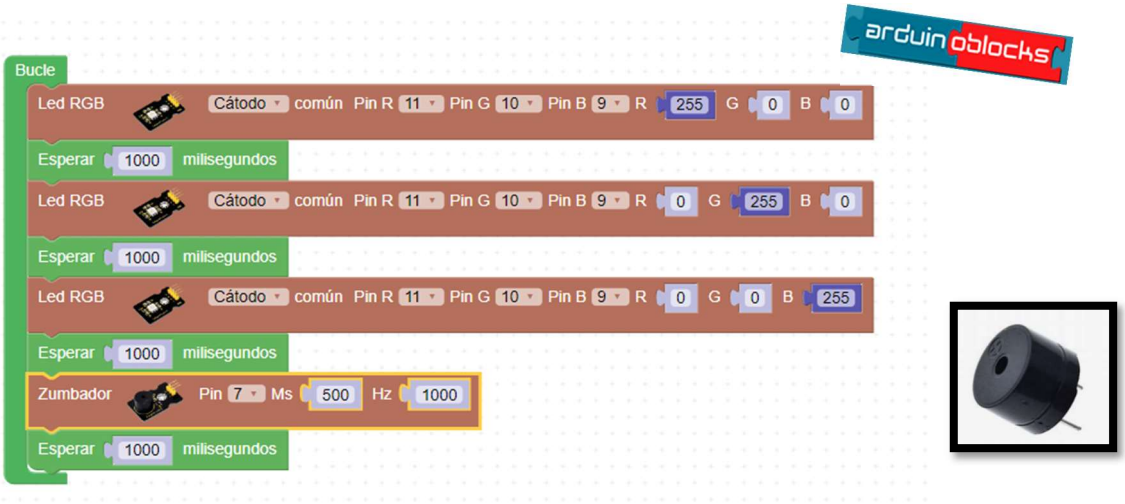


4.1.3 Programación con arduinoblocks

PROGRAMA CON ARDUINOBLOCK DE LED PARA MOSTRAR COLORES ROJO VERDE Y AZUL:

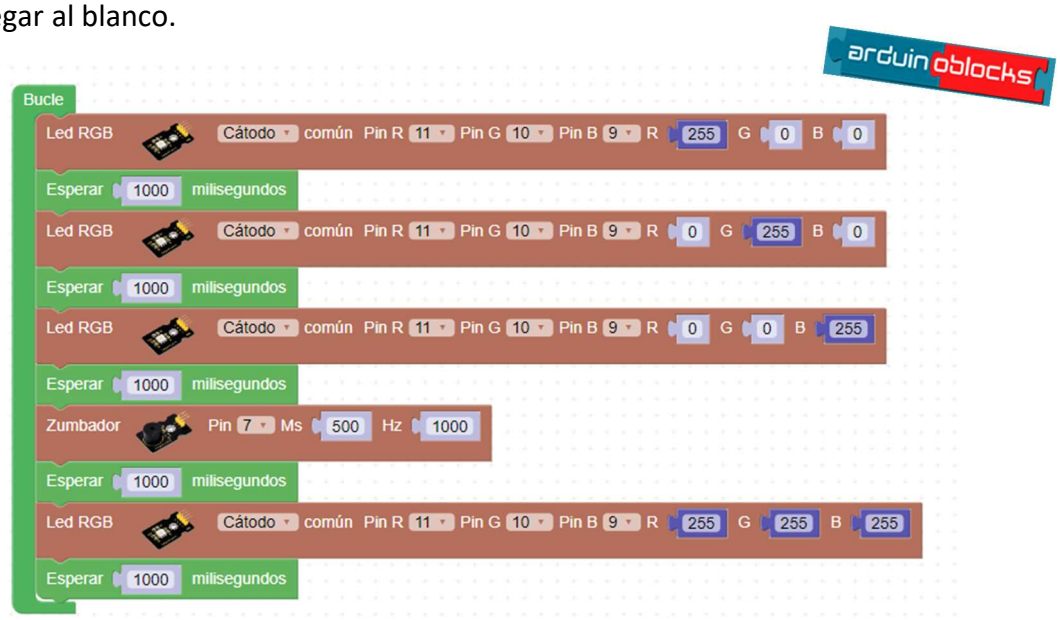


AMPLIACIÓN: ZUMBADOR AL FINAL DE LA SECUENCIA:

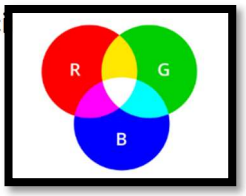


RETO: CREAR EL COLOR BLANCO DESPUÉS DEL PULSADOR:

Recuerda que es un **sistema de color aditivo**: cada combinación de color añade luz hasta llegar al blanco.



4.1.4 Ejercicios.



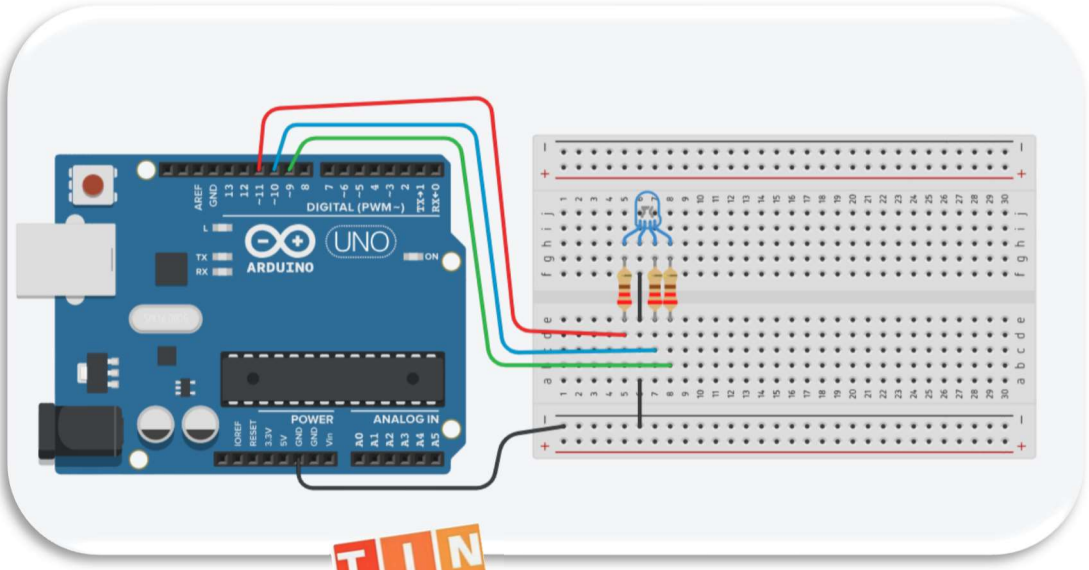
Crea la secuencia de colores cambiando los parámetros de la programaci

MAGENTA (1SG) – ROJO (1SG) – AZUL (1SG) – ZUMBADOR.

AMARILLO (1SG9 – MAGENTA(1SG) – CIAN(1SG) --ZUMBADOR.

MAGENTA(1SG) –BLANCO (1SG) – VERDE (1SG) ---ZUMBADOR.

4.2 PRÁCTICA 1 SIMULADA CON TINKERCAD

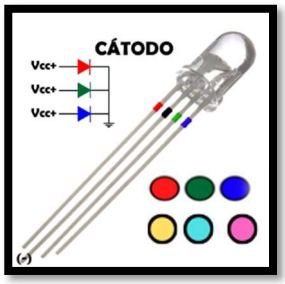



```
// C++ code
//
int counter;

void setup()
{
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop()
{
  for (counter = 0; counter < 10;
++counter) {
    digitalWrite(11, HIGH);
    digitalWrite(10, LOW);
    digitalWrite(9, LOW);
    delay(1000); // Wait for 1000
millisecond(s)
    digitalWrite(11, LOW);
    digitalWrite(10, HIGH);
    digitalWrite(9, LOW);
    delay(1000); // Wait for 1000
millisecond(s)
    digitalWrite(11, LOW);
    digitalWrite(10, LOW);
    digitalWrite(9, HIGH);
    delay(1000); // Wait for 1000
millisecond(s)
  }
}
```

4.3 PRÁCTICA 2.CONTROL PROGRAMADO DE LED RGB CON TRES PULSADORES. COLORES PRIMARIOS Y SECUNDARIOS

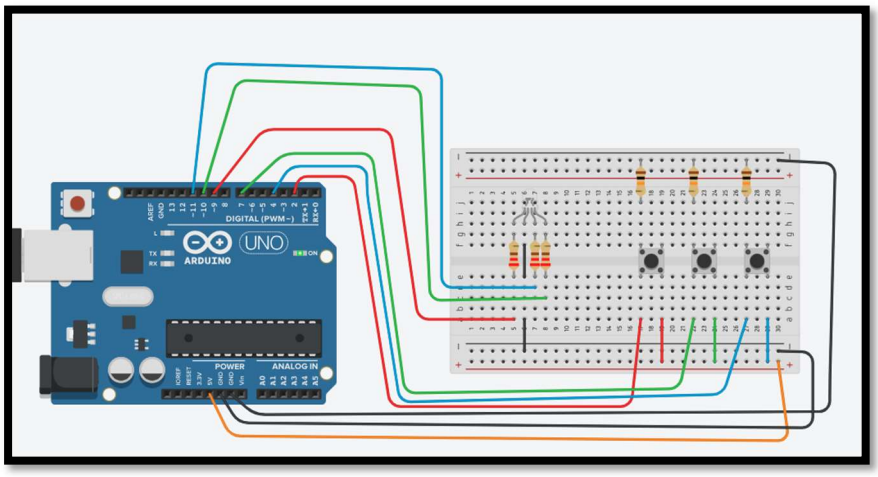
Esta programación nos permite controlar el LED RBG mediante tres pulsadores. Con su combinación podemos hacer que aparezcan en el LED RGB los colores primarios o secundarios.

4.3.1 Materiales.

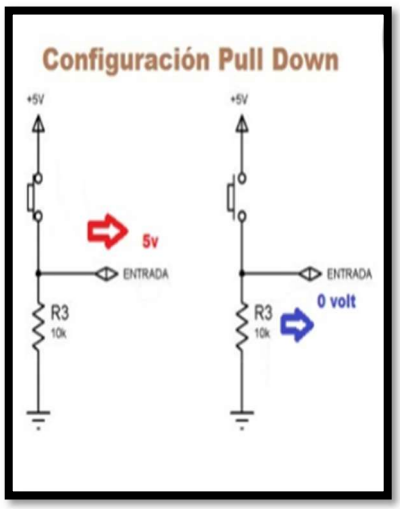


- 1 Placa de Arduino UNO.
- 1 Protoboard y Latiguillos.
- 1 Diodo Led RGB (ánodo común).
- 3 Resistencias de 220 Ω.
- 3 Resistencias de 10 kΩ.
- 3 Pulsadores.

4.3.2 Montaje del circuito.



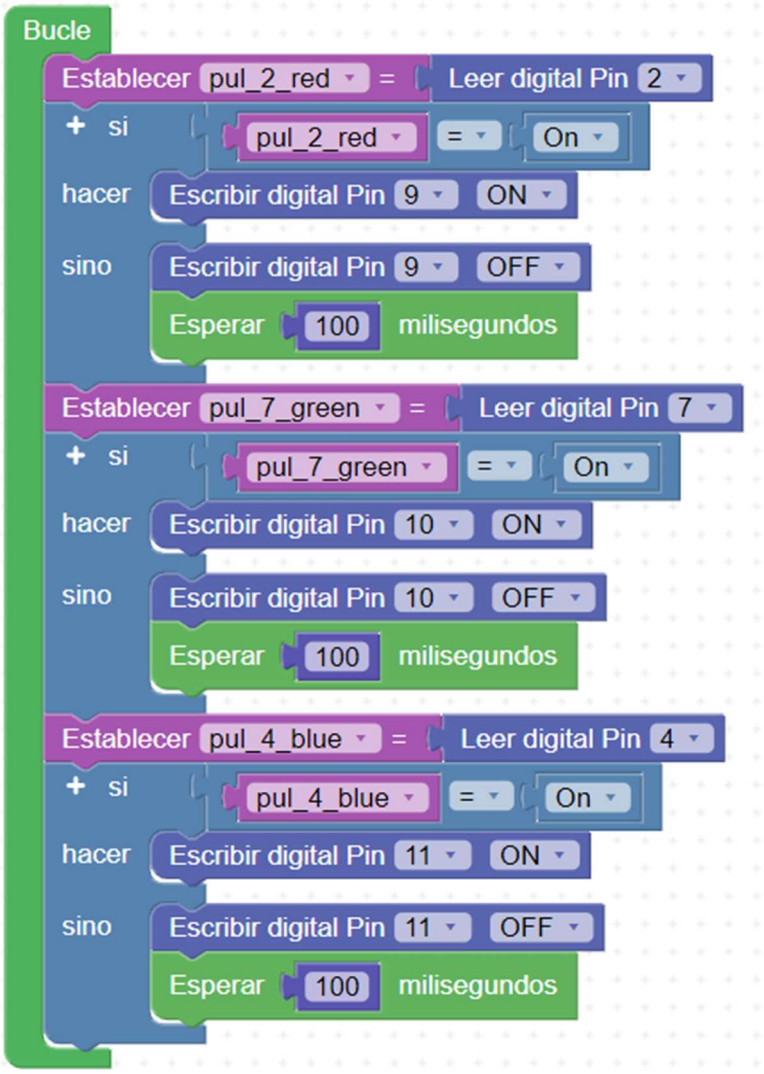
Para realizar el montaje correctamente, necesitaremos conectar una resistencia a cada pulsador del circuito (Pull-Down). Esta resistencia es un mecanismo habitual dentro del mundo de la electrónica.

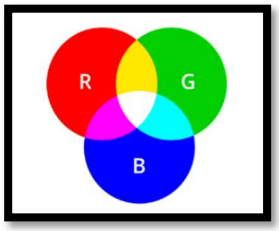


La resistencia de Pull-Down fuerza LOW (0, OFF) en el PIN cuando el pulsador está abierto y cuando está cerrado el PIN se pone a HIGH (1, ON).

Los valores ON son los que utilizamos como condición para que cada patilla del RGB

4.3.3 *Programación en ArduinoBlocks*





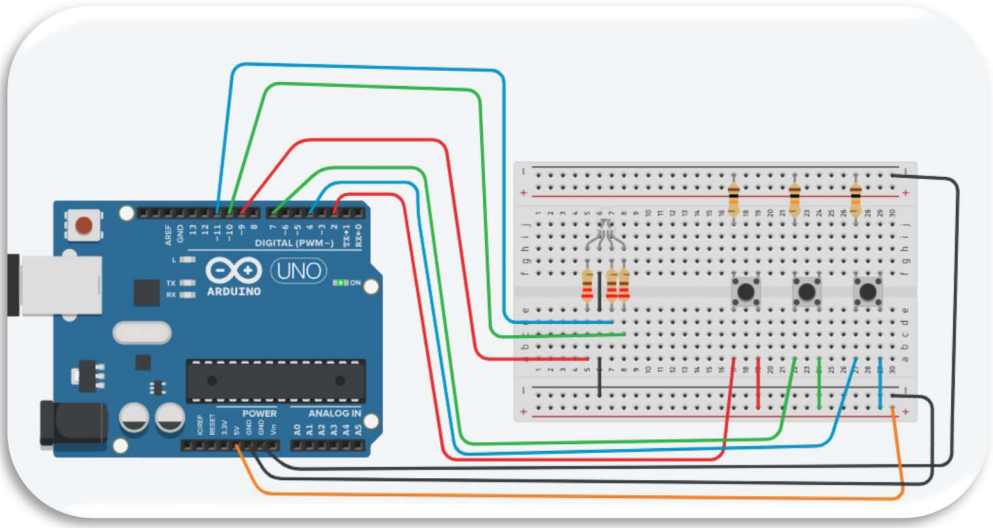
4.3.4 *Ejercicios.*

Crea los siguientes colores manejando los pulsadores:

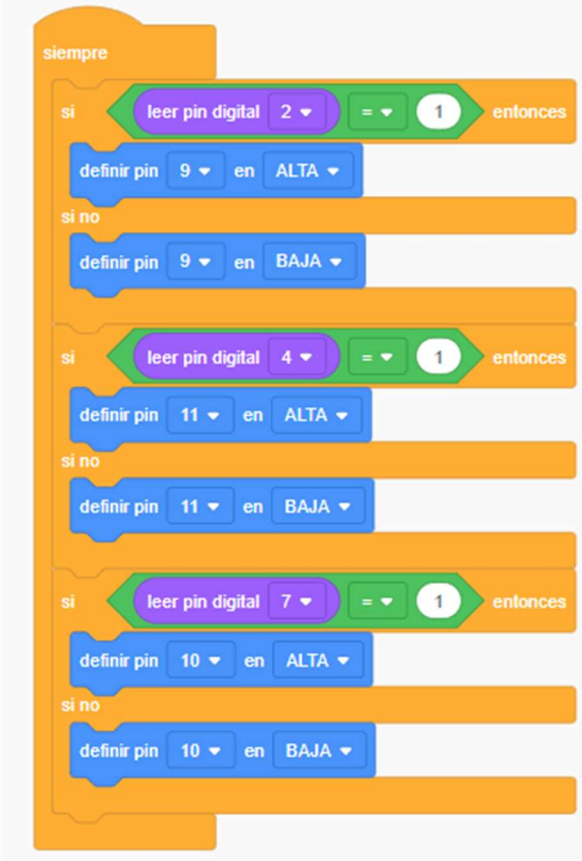
ROJO (PULSADOR2) – VERDE (PULSARDOR7) – AZUL (PULSARDOR4).

A. MAGENTA. (CON DOS PULSADORES).
B. AMARILLO (CON DOS PULSADORES).
C. CIAN. (CON DOS PULSADORES).
D. BLANCO. (CON TRES PULSADORES).

4.4 PRÁCTICA 2 SIMULADA CON TINKERCAD



Código para simulación en Tinkercad.

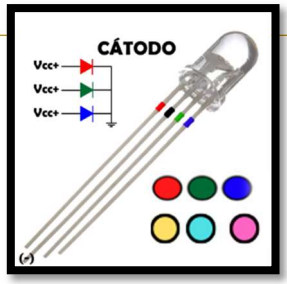


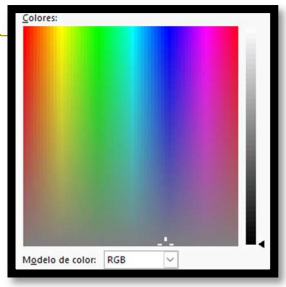

```
// C++ code
//
void setup()
{
  pinMode(2, INPUT); //rojo pulsador
  pinMode(11, OUTPUT);//azul led
  pinMode(10, OUTPUT);//verde led
  pinMode(9, OUTPUT);//rojo led
  pinMode(4, INPUT);//azul pulsador
  pinMode(7, INPUT);//verde pulsador
}

void loop()
{
  if (digitalRead(2) == HIGH) {
    digitalWrite(9, 255);

  } else {
    digitalWrite(9, 0);

}
 if (digitalRead(4) == HIGH) {
    digitalWrite(11, 255);

  } else {
    digitalWrite(11, 0);

     if (digitalRead(7) == HIGH) {
    digitalWrite(10, 255);

  } else {
    digitalWrite(10, 0);

}

}
}
```

4.5 PRÁCTICA 3.CONTROL PROGRAMADO DE UN LED RGB CON TRES POTENCIÓMETROS.

Esta programación nos permite controlar el LED RBG mediante potenciómetros, uno para cada color (rojo, verde y azul) de esta forma podemos obtener cualquier color de la gama que ofrece el modelo RBG.

4.5.1 Materiales.

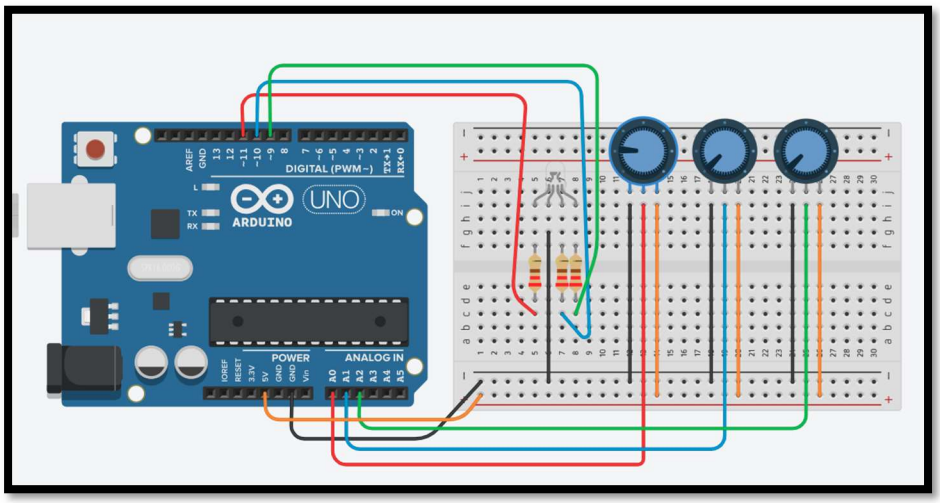




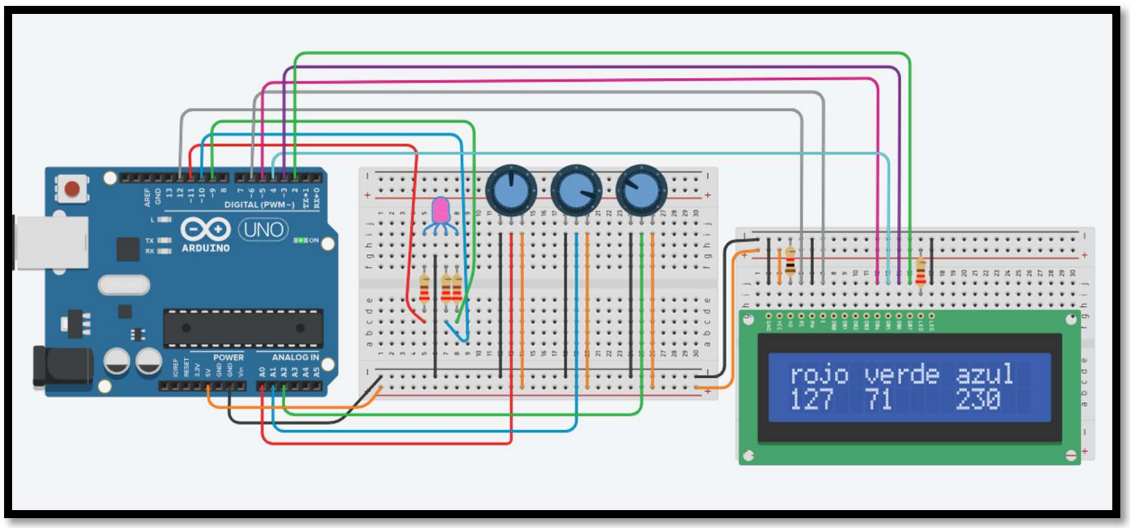
- 1 Placa de Arduino UNO.
- 1 Protoboard y Latiguillos.
- 1 Diodo Led RGB (ánodo común).
- 3 Resistencias de 220Ω.
- 3 Potenciómetros.

4.5.2 Montaje del circuito.

Circuito con RBG tres potenciómetros, sin LCD.

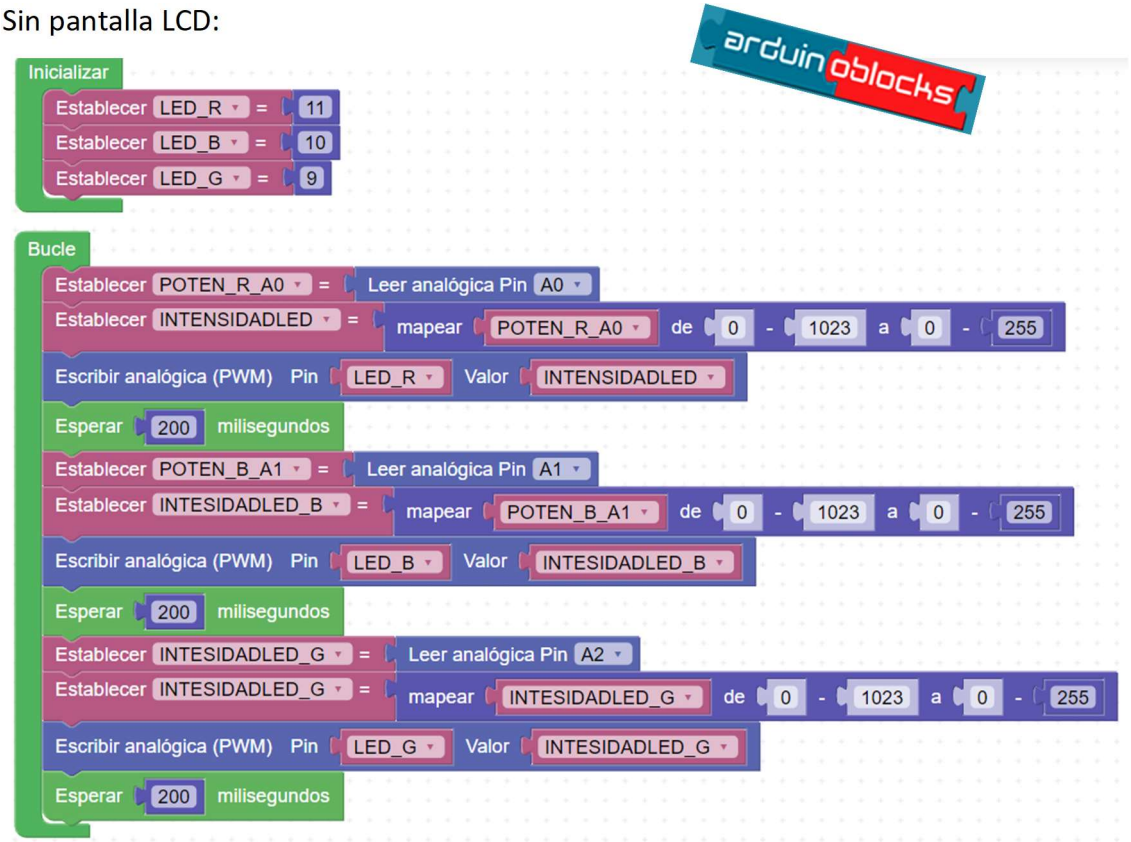


Circuito con RBG tres potenciómetros, conde.

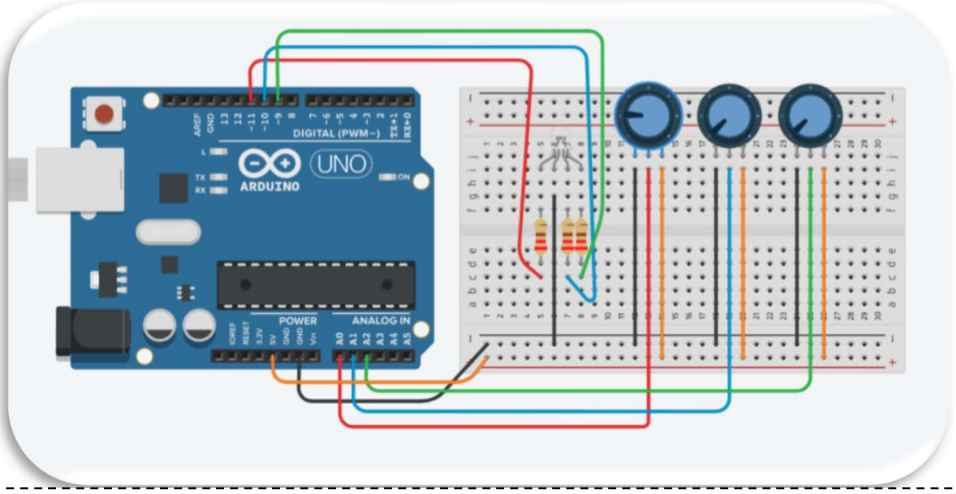


Programación con ArduinoBlocks

Sin pantalla LCD:



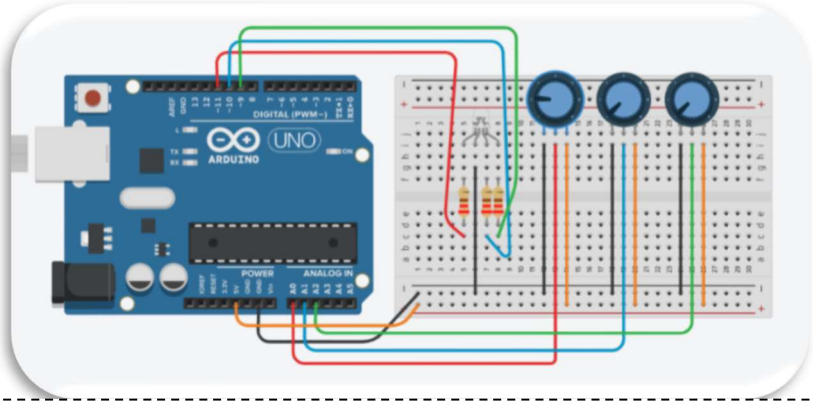
4.6 PRÁCTICA 3 SIN LCD SIMULADA CON TINKERCAD



Código para simulación en Tinkercad.


```
// C++ code
//
int potRojo = 0;

int potAzul = 0;

int potverde = 0;

int ledRojo = 0;

int ledVerde = 0;

int ledAzul = 0;

void setup()
{
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop()
{
  potRojo = analogRead(A0);
  potAzul = analogRead(A1);
  potverde = analogRead(A2);
  analogWrite(11, potRojo);
  analogWrite(10, potAzul);
  analogWrite(9, potverde);
  ledRojo = (potRojo / 4);
  ledVerde = (potverde / 4);
  ledAzul = (potAzul / 4);
  delay(10); // Delay a little bit to
improve simulation performance
}
```

4.7 PRÁCTICA 3 SIN LCD PROGRAMADA CON ARDUINO IDE

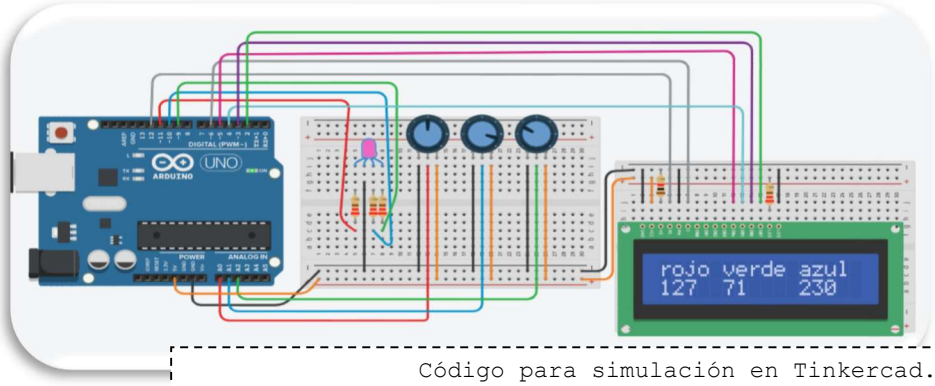


Código depurado con Arduino IDE para cargar en la placa (montaje físico):

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 6, 5, 4, 3, 2);

float potrojo = 0;
float ledrojo = 0;
float potverde = 0;
float ledverde = 0;
float potazul = 0;
float ledazul = 0;

void setup() {
  pinMode(A0, INPUT);
  pinMode(9, OUTPUT);
  pinMode(A3, INPUT);
  pinMode(10, OUTPUT);
  pinMode(A5, INPUT);
  pinMode(11, OUTPUT);
  lcd.begin(16, 2);
  Serial.begin(9600);
}

void loop() {
  // Leer y mapear valores del
potenciómetro rojo (A0) a 0-255 para el
LED
  potrojo = analogRead(A0);
  ledrojo = map(potrojo, 50, 1016, 0,
255);  // Ajusta el valor mínimo
detectado a 2
  ledrojo = constrain(ledrojo, 0, 255);
// Limitar el valor a 0-255
  analogWrite(11, ledrojo);
// Enviar señal PWM al LED rojo

  // Leer y mapear valores del
potenciómetro azul (A1) a 0-255 para el
LED
  potazul = analogRead(A3);
  ledazul = map(potazul, 50, 1016, 0,
255);  // Ajusta el valor mínimo
detectado a 2
  ledazul = constrain(ledazul, 0, 255);
// Limitar el valor a 0-255
  analogWrite(10, ledazul);
// Enviar señal PWM al LED azul

  // Leer y mapear valores del
potenciómetro verde (A2) a 0-255 para el
LED
  potverde = analogRead(A5);
  ledverde = map(potverde, 50, 1016, 0,
255);  // Ajusta el valor mínimo detectado
a 2
  ledverde = constrain(ledverde, 0, 255);
// Limitar el valor a 0-255
  analogWrite(9, ledverde);
// Enviar señal PWM al LED verde

  // Actualizar el LCD con los valores de
los LEDs
  lcd.setCursor(0, 0);  // Mostrar en la
primera fila
  lcd.print("R");
  lcd.setCursor(0, 1);
  lcd.print(ledrojo, 1);  // Mostrar el
valor de brillo del LED rojo

  lcd.setCursor(6, 0);  // Mostrar en la
primera fila
  lcd.print("G");
  lcd.setCursor(6, 1);
  lcd.print(ledverde, 1);  // Mostrar el
valor de brillo del LED verde

  lcd.setCursor(12, 0);  // Mostrar en la
primera fila
  lcd.print("B");
  lcd.setCursor(12, 1);
  lcd.print(ledazul, 1);  // Mostrar el
valor de brillo del LED azul

  // Mostrar los valores en el monitor
serie para depuración
  Serial.print("Rojo: ");
  Serial.println(potrojo);
  Serial.print("Verde: ");
  Serial.println(potverde);
  Serial.print("Azul: ");
  Serial.println(potazul);

  delay(3000);  // Esperar 1 segundo entre
actualizaciones
}
```


Programa financiado por el Ministerio de Educación, Formación Profesional y Deportes

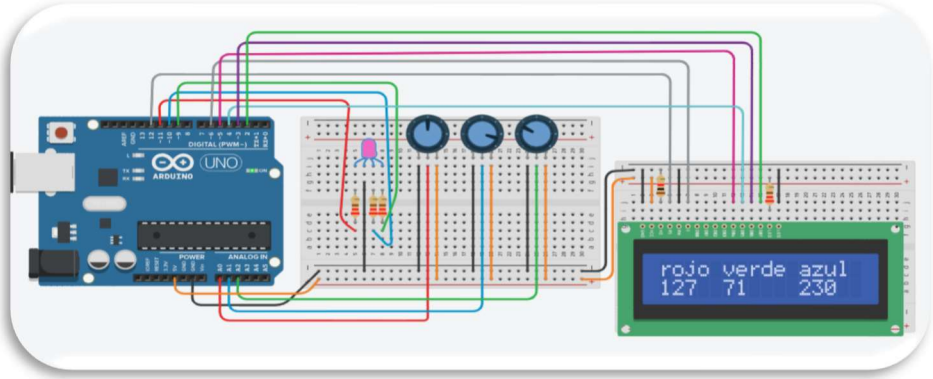
4.8 PRÁCTICA 3 CON LCD SIMULADA CON TINKERCAD



Código para simulación en Tinkercad.


```
#include <LiquidCrystal.h>
int potRojo = 0;
int potAzul = 0;
int potverde = 0;
int ledRojo = 0;
int ledVerde = 0;
int ledAzul = 0;
LiquidCrystal lcd_1(12, 6, 5, 4, 3, 2);

void setup()
{
  lcd_1.begin(16, 2); // Set up the number
of columns and rows on the LCD.
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(11, OUTPUT);
  pinMode(10, OUTPUT);
  pinMode(9, OUTPUT);
}

void loop()
{
  potRojo = analogRead(A0);
  potAzul = analogRead(A1);
  potverde = analogRead(A2);
  analogWrite(11, potRojo);
  analogWrite(10, potAzul);
  analogWrite(9, potverde);
  ledRojo = (potRojo / 4);
  ledVerde = (potverde / 4);
  ledAzul = (potAzul / 4);
  lcd_1.setCursor(0, 0);
  lcd_1.print("rojo");
  lcd_1.setCursor(0, 1);
  lcd_1.print(ledRojo);
  lcd_1.setCursor(5, 0);
  lcd_1.print("verde");
  lcd_1.setCursor(5, 1);
  lcd_1.print(ledVerde);
  lcd_1.setCursor(11, 0);
  lcd_1.print("azul");
  lcd_1.setCursor(11, 1);
  lcd_1.print(ledAzul);
  delay(1000); // Wait for 1000
millisecond(s)
  lcd_1.clear();
}
```

4.9 PRÁCTICA 3 CON LCD PROGRAMADA CON ARDUINOIDE



Código depurado con Arduino IDE para cargar en la placa (montaje físico:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 6, 5, 4, 3, 2);

float potrojo = 0;
float ledrojo = 0;
float potverde = 0;
float ledverde = 0;
float potazul = 0;
float ledazul = 0;

void setup() {
  pinMode(A0, INPUT);
  pinMode(9, OUTPUT);
  pinMode(A3, INPUT);
  pinMode(10, OUTPUT);
  pinMode(A5, INPUT);
  pinMode(11, OUTPUT);
  lcd.begin(16, 2);
  Serial.begin(9600);
}

void loop() {
  // Leer y mapear valores del
potenciómetro rojo (A0) a 0-255 para el
LED
  potrojo = analogRead(A0);
  ledrojo = map(potrojo, 50, 1016, 0,
255);  // Ajusta el valor mínimo
detectado a 2
  ledrojo = constrain(ledrojo, 0, 255);
// Limitar el valor a 0-255
  analogWrite(11, ledrojo);
// Enviar señal PWM al LED rojo

  // Leer y mapear valores del
potenciómetro azul (A1) a 0-255 para el
LED
  potazul = analogRead(A3);
  ledazul = map(potazul, 50, 1016, 0,
255);  // Ajusta el valor mínimo
detectado a 2
  ledazul = constrain(ledazul, 0, 255);
// Limitar el valor a 0-255
  analogWrite(10, ledazul);
// Enviar señal PWM al LED azul
```

```
  // Leer y mapear valores del
potenciómetro verde (A2) a 0-255 para el
LED
  potverde = analogRead(A5);
  ledverde = map(potverde, 50, 1016, 0,
255);  // Ajusta el valor mínimo
detectado a 2
  ledverde = constrain(ledverde, 0,
255);     // Limitar el valor a 0-255
  analogWrite(9, ledverde);
// Enviar señal PWM al LED verde

  // Actualizar el LCD con los valores
de los LEDs
  lcd.setCursor(0, 0);  // Mostrar en la
primera fila
  lcd.print("R");
  lcd.setCursor(0, 1);
  lcd.print(ledrojo, 1);  // Mostrar el
valor de brillo del LED rojo

  lcd.setCursor(6, 0);  // Mostrar en la
primera fila
  lcd.print("G");
  lcd.setCursor(6, 1);
  lcd.print(ledverde, 1);  // Mostrar el
valor de brillo del LED verde

  lcd.setCursor(12, 0);  // Mostrar en
la primera fila
  lcd.print("B");
  lcd.setCursor(12, 1);
  lcd.print(ledazul, 1);  // Mostrar el
valor de brillo del LED azul

  // Mostrar los valores en el monitor
serie para depuración
  Serial.print("Rojo: ");
  Serial.println(potrojo);
  Serial.print("Verde: ");
  Serial.println(potverde);
  Serial.print("Azul: ");
  Serial.println(potazul);

  delay(3000);  // Esperar 1 segundo
entre actualizaciones
}
```

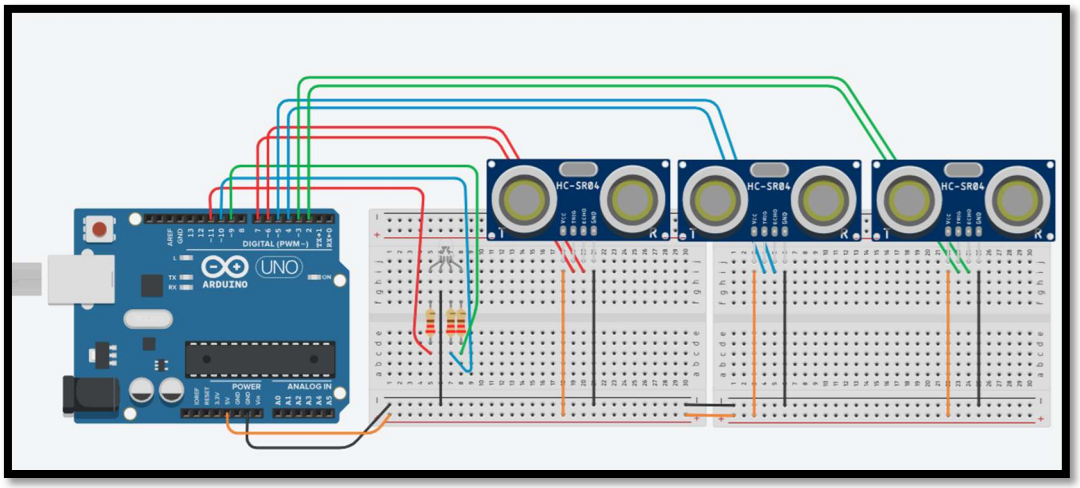
4.10 PRÁCTICA 4. CONTROL PROGRAMADO DE UN LED RGB CON TRES SENSORES DE ULTRASONIDO.

Esta programación nos permite controlar el LED RBG mediante sensores de ultrasonidos (uno para cada color) de forma que aparezca cualquier color de la gama de colores que ofrece el modelo RBG.

4.10.1 Materiales

- 1 Placa de Arduino UNO.
- 1 Protoboard y Latiguillos.
- 1 Diodo Led RGB (ánodo común).
- 3 Resistencias de 220Ω.
- 3 Sensores de ultrasonido.

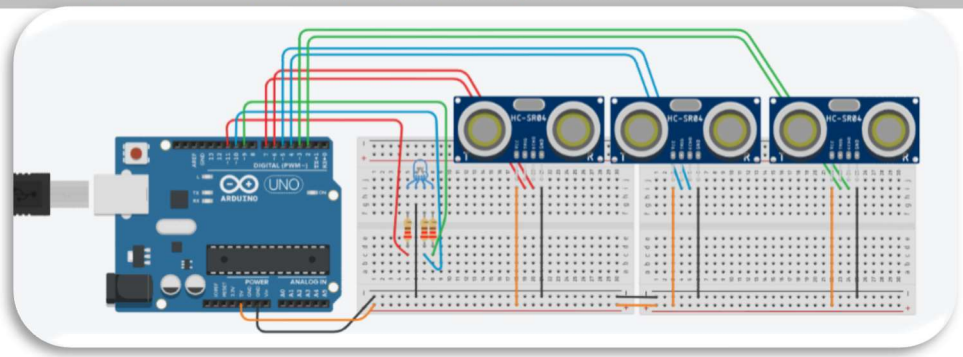
4.10.2 Montaje del circuito



4.10.3 Programación

Esta práctica se programa mediante texto con el IDE de Arduino, podemos ver el código en el siguiente apartado.

4.11 PRÁCTICA 4 PROGRAMADA CON ARDUINOIDE



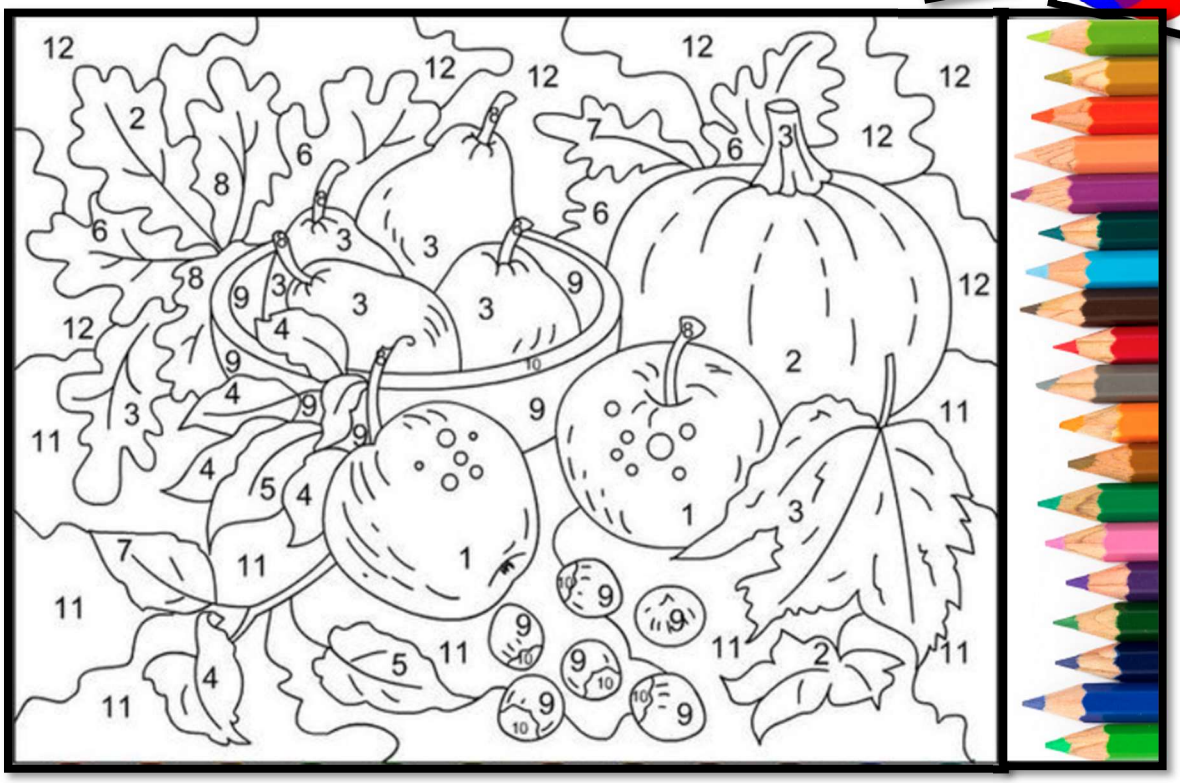
```
// Pines del LED RGB
const int pinRojo = 11;
const int pinVerde = 9;
const int pinAzul = 10;

// Pines de los sensores ultrasónicos
const int trigRojo = 7;
const int echoRojo = 6;
const int trigVerde = 2;
const int echoVerde = 3;
const int trigAzul = 4;
const int echoAzul = 5;

// Función para medir la distancia con el sensor HC-SR04
long medirDistancia(int trigPin, int echoPin)
{
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duracion = pulseIn(echoPin, HIGH);
  long distancia = duracion * 0.034 / 2;  // Convertimos a distancia en milímetros
  return distancia;
}
 void setup() {
  // Configurar los pines del LED RGB como salidas
  pinMode(pinRojo, OUTPUT);
  pinMode(pinVerde, OUTPUT);
  pinMode(pinAzul, OUTPUT);
   // Configurar los pines de los sensores ultrasónicos
  pinMode(trigRojo, OUTPUT);
  pinMode(echoRojo, INPUT);
  pinMode(trigVerde, OUTPUT);
  pinMode(echoVerde, INPUT);
  pinMode(trigAzul, OUTPUT);
  pinMode(echoAzul, INPUT);

  Serial.begin(9600);  // Para monitorear datos en el puerto serie
}

void loop() {
  // Medir la distancia de cada sensor
  long distanciaRojo = medirDistancia(trigRojo, echoRojo);
  long distanciaVerde = medirDistancia(trigVerde, echoVerde);
  long distanciaAzul = medirDistancia(trigAzul, echoAzul);
```

```
  // Mapear las distancias (0 a 2550 mm) a valores PWM (0 a 255)
  int valorRojo = map(distanciaRojo, 0, 255, 0, 255);
  int valorVerde = map(distanciaVerde, 0, 255, 0, 255);
  int valorAzul = map(distanciaAzul, 0, 255, 0, 255);
// Limitar las distancias a un rango aceptable


// Limitar las distancias a un rango aceptable
  if (distanciaRojo < 20 || distanciaRojo > 255) {
    valorRojo = 0;  // Fuera de rango, se apaga el color rojo
  }
  if (distanciaVerde < 20 || distanciaVerde > 255) {
    valorVerde = 0;  // Fuera de rango, se apaga el color verde
  }
  if (distanciaAzul < 20 || distanciaAzul > 255) {
    valorAzul = 0;  // Fuera de rango, se apaga el color azul
  }
  // Limitar los valores mapeados para que estén entre 0 y 255
  valorRojo = constrain(valorRojo, 0, 255);
  valorVerde = constrain(valorVerde, 0, 255);
  valorAzul = constrain(valorAzul, 0, 255);

  // Enviar los valores PWM al LED RGB
  analogWrite(pinRojo, valorRojo);
  analogWrite(pinVerde, valorVerde);
  analogWrite(pinAzul, valorAzul);

  // Mostrar los valores en el monitor serie para depuración
  Serial.print("Distancia Rojo: ");
  Serial.print(distanciaRojo);
  Serial.print(" mm, Valor PWM Rojo: ");
  Serial.println(valorRojo);

  Serial.print("Distancia Verde: ");
  Serial.print(distanciaVerde);
  Serial.print(" mm, Valor PWM Verde: ");
  Serial.println(valorVerde);

  Serial.print("Distancia Azul: ");
  Serial.print(distanciaAzul);
  Serial.print(" mm, Valor PWM Azul: ");
  Serial.println(valorAzul);

  delay(100);  // Pausa pequeña para estabilidad
}
```


Programa financiado por el Ministerio de Educación, Formación Profesional y Deportes

5 ANEXOS

5.1 ANEXO 1. RYB: FICHA PARA ALUMNOS: COLOREAR CON PINTURAS



Pinta la imagen con estos colores.

Obtén los colores en el led RBG que has programado.

1	ROJO	RGB (255.0.0)	*colorea*

Práctica 1: Programación de LED RBG

1	ROJO	RGB (255.0.0)	*colorea*
2	VERDE	RGB (0.255.0)	*colorea*
3	AZUL	RGB (0.0.255)	*colorea*

Práctica 2: Programación de pulsadores y LED RBG

4	AMARILLO	RGB (255.255.0)	*colorea*
5	CIÁN	RGB (0.255.255)	*colorea*
6	MAGENTA	RGB (255.0.255)	*colorea*

Práctica 3: Programación de potenciómetro y LED RBG:

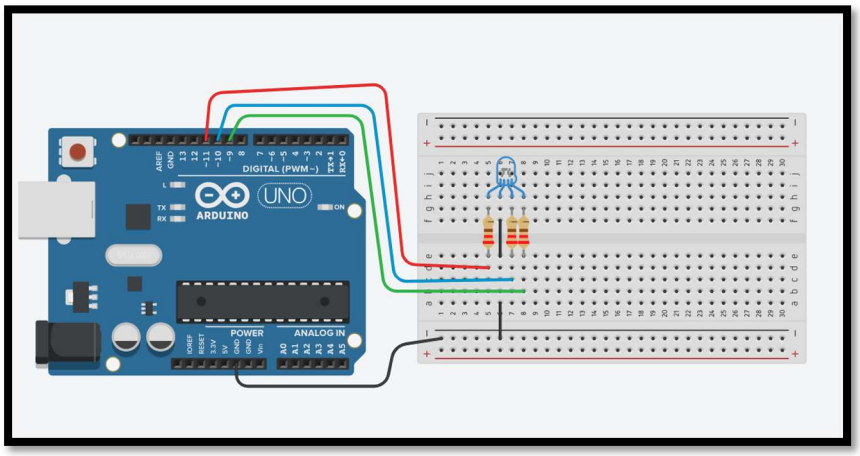
7		RGB (255.130.110)	*colorea*
8		RGB (142.64.58)	*colorea*
9		RGB (55.126.71)	*colorea*

Práctica 4: Programación de potenciómetro y LED RBG. Crea tus colores

10		RGB (___.___.___)	*colorea*
11		RGB (___.___.___)	*colorea*
12		RGB (___.___.___)	*colorea*

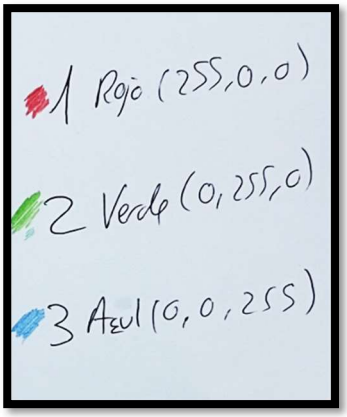
5.1.1 RYB: Proceso de desarrollo de la actividad

Montaje 1. Colores Primarios.

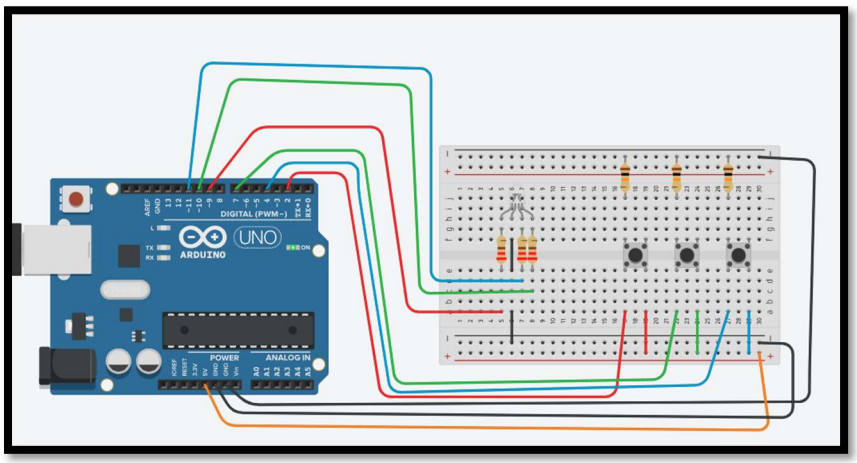


Con este montaje, variando los datos de programación desde el ordenador, conseguimos los colores primarios. Una vez observados los colores podemos rellenar la primera parte del dibujo (rojo, verde, azul)

Ejemplo de dibujo después de práctica 1:

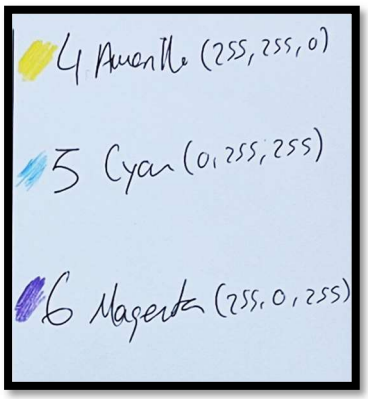


Montaje 2. Colores Secundarios.

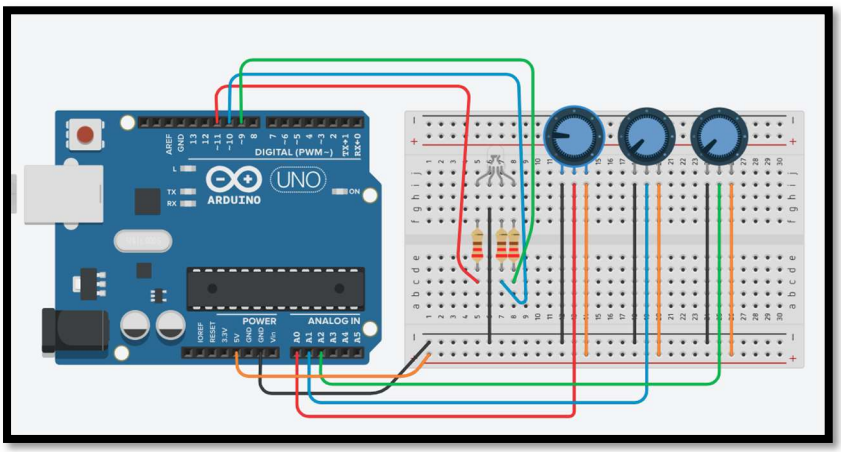


En este montaje programamos para controlar el led desde los pulsadores, pulsando un solo pulsador conseguimos un color primario, pero si pulsamos dos a la vez obtenemos un color secundario. Una vez observados los colores secundarios podemos rellenar la segunda parte del dibujo (amarillo, cian, magenta)

Ejemplo de dibujo después de práctica 1:

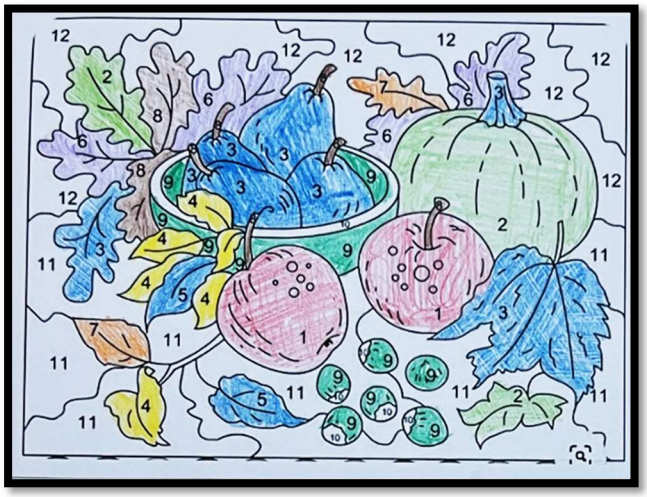


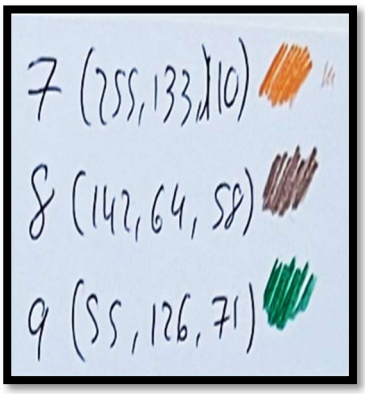
Montaje 3. Toda la gama de Colores.



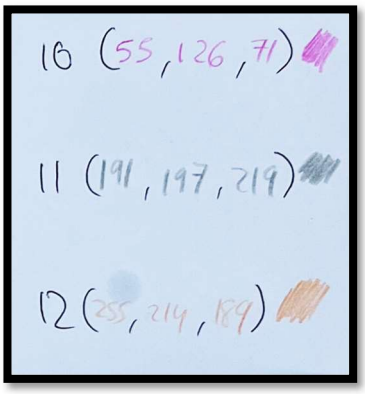
Con este montaje programamos para controlar el led desde los potenciómetros, girando cada potenciómetro conseguimos cualquier color de la gama. Una vez observados los colores rellenamos el resto del dibujo.

Ejemplo de dibujo después de práctica 3 y colores asignados:

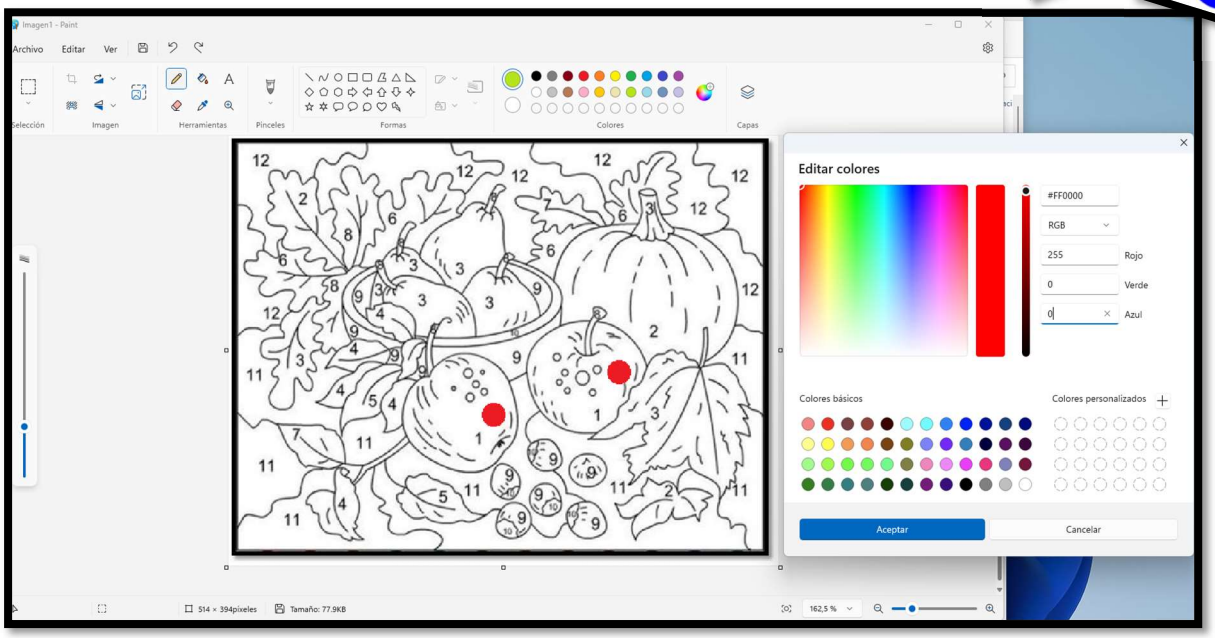




Ejemplo de dibujo después de práctica 3 y colores creados por alumnos:



5.2 ANEXO 2. RBG: FICHA PARA ALUMNOS: COLOREAR EN PANTALLA TÁCTIL



Obtén los colores en el led RBG que has programado y utiliza una herramienta digital, por ejemplo, Paint, para pintar la imagen con estos colores.

Práctica 1: Programación de LED RBG

1	ROJO	RGB (255.0.0)	*colorea*
2	VERDE	RGB (0.255.0)	*colorea*
3	AZUL	RGB (0.0.255)	*colorea*

Práctica 2: Programación de pulsadores y LED RBG

4	AMARILLO	RGB (255.255.0)	*colorea*
5	CIÁN	RGB (0.255.255)	*colorea*
6	MAGENTA	RGB (255.0.255)	*colorea*

Práctica 3: Programación de potenciómetro y LED RBG

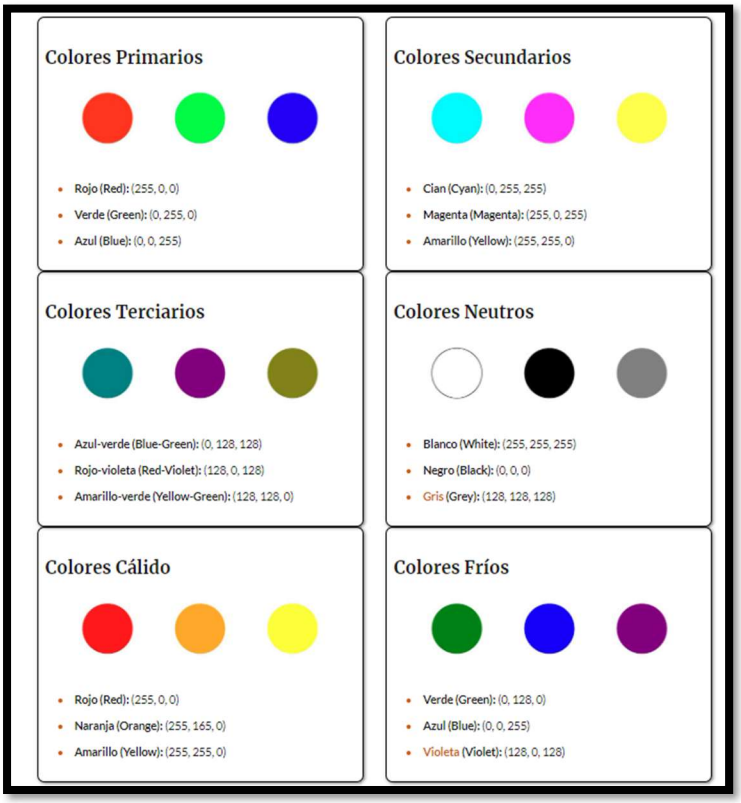
7		RGB(255.130.110)	*colorea*
8		RGB(142.64.58)	*colorea*
9		RGB(55.126.71)	*colorea*

Práctica 4: Programación de potenciómetro y LED RBG. Crea tus colores

10		RGB(___.___.___)	*colorea*
11		RGB(___.___.___)	*colorea*
12		RGB(___.___.___)	*colorea*

5.3 ANEXO 3. AGRUPACIÓN DE COLORES EN RBG

Categóricas según apariencia y uso:

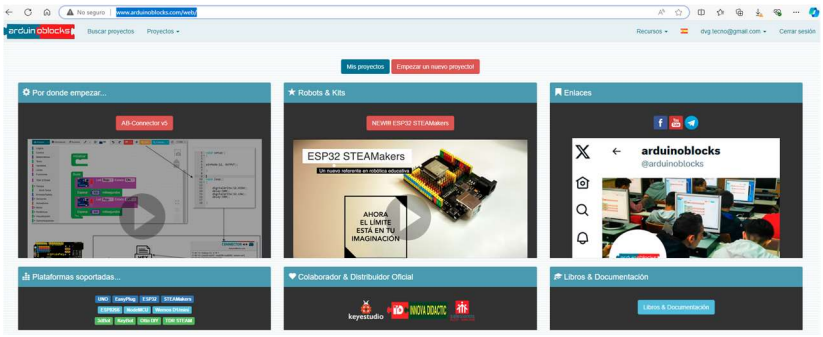


Códigos numéricos en para designar colores en RGB

Color	Nombre HTML / CSS	Código hexadecimal #RRGGBB	Código decimal (R, G, B)
	Negro	# 000000	(0,0,0)
	Blanco	#FFFFFF	(255,255,255)
	Rojo	# FF0000	(255,0,0)
	Lima	# 00FF00	(0,255,0)
	Azul	# 0000FF	(0,0,255)
	Amarillo	# FFFF00	(255,255,0)
	Cian / Aqua	# 00FFFF	(0,255,255)
	Magenta / Fucsia	# FF00FF	(255,0,255)
	Plata	# C0C0C0	(192,192,192)
	gris	# 808080	(128,128,128)
	Granate	# 800000	(128,0,0)
	Aceituna	# 808000	(128,128,0)
	Verde	# 008000	(0,128,0)
	Púrpura	# 800080	(128,0,128)
	Verde azulado	# 008080	(0,128,128)
	Armada	# 000080	(0,0,128)

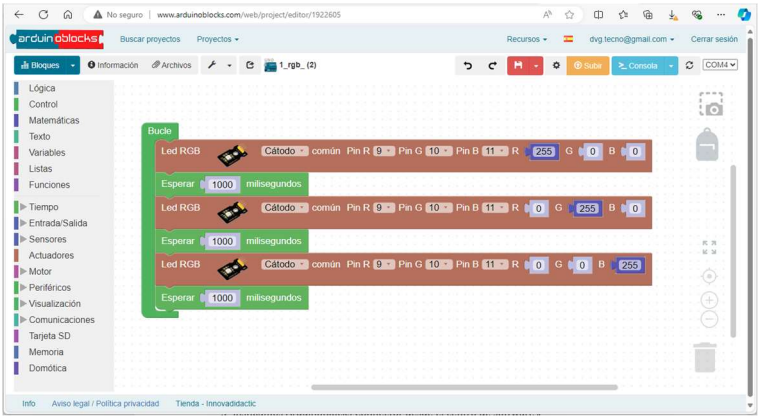
5.4 ANEXO 4. CONECTAR ARDUINOBLOCK CON PLACA ARDUINO UNO.

1. Abrir página web Arduinoblock e iniciar sesión (si no estamos registrados nos registramos , si me han creado una clase entramos en ella).

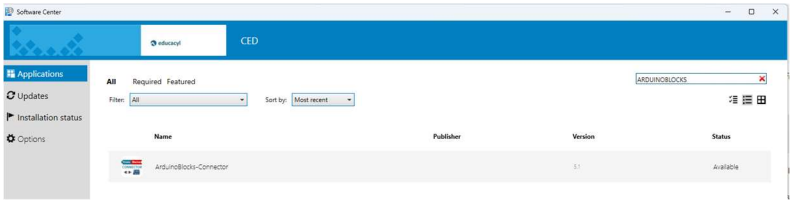


http://www.arduinoblocks.com/web/

2. Programo en la interfaz de Arduinoblock.



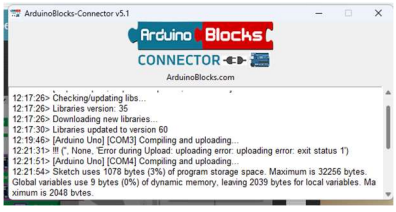
3. Instalamos ArduinoBlocks Connector desde el centro de software.



Si no estamos en red lo podemos instalar desde la página de ArduinoBlocks. *http://www.arduinoblocks.com/web/site/abconnector5*

4. Ejecutamos ArduinoBlocks Connector desde el acceso dierecto que se ha creado en el escritorio.

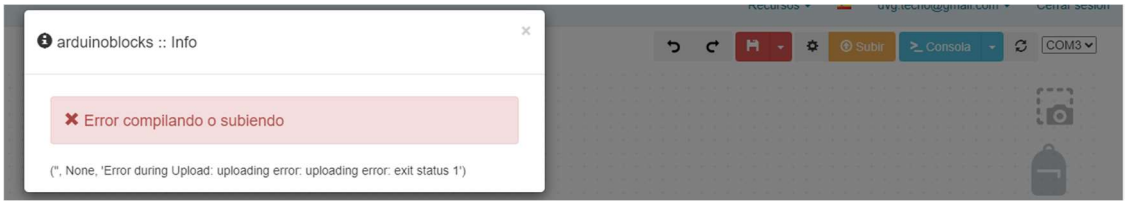
Icono de escritorio



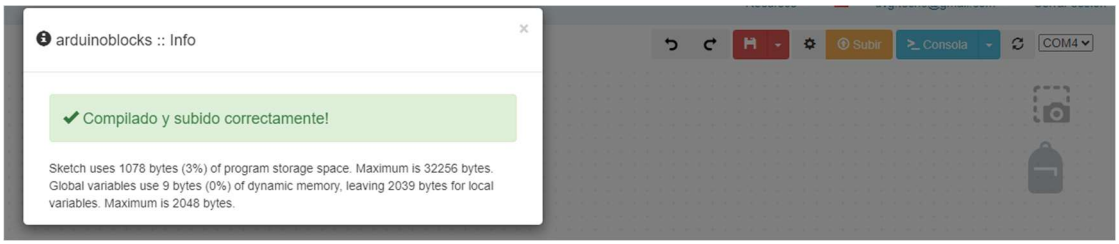
Pantalla que sale cuando está ejecutado.

5. Conecto la placa de Arduinon selecciono un puerto (COM3,COM4,etc.) y lo cargo en la placa con un clik en botón Subir

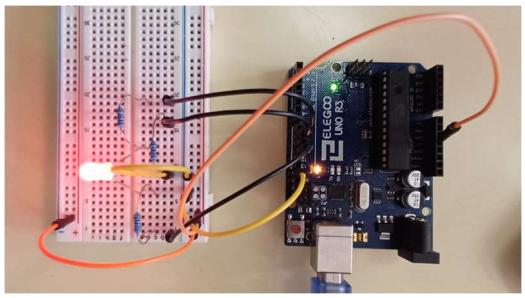
Si me sale un Error compilando hay que comprobar el puerto al que tengo conectada la placa.



Selecciono otro puerto y lo intento subir de nuevo a la placa.



6. Compruebo que funciona en la placa y depuro errores.

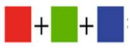
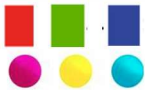
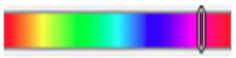
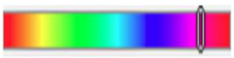


7. Para guardar el programa hacemos clik en icono de guardar

8. Si quiero ver el código de Arduino hago click en bloques – ver cógido.

5.5 ANEXO 5. RESUMEN DE PRÁCTICAS

Cuadro resumen de las prácticas plantedas:

	PRÁCTICA 1	PRÁCTICA 2	PRÁCTICA 3	PRÁCTICA 4
ETAPA	ESO (segundo ciclo)	ESO (segundo ciclo)	ESO (segundo ciclo)	ESO (segundo ciclo)
CURSO	3º ESO	3º ESO	3º ESO	3º ESO (Altas capacidades)
CONTENIDO ED. PLÁSTICA	Tª DEL COLOR	Tª DEL COLOR	Tª DEL COLOR	Tª DEL COLOR
MODELO DE COLOR	Intoducción a Modelos de color. RYB y RGB. Colores primarios	Modelo de color RGB. Colores secundarios	Modelo de color RGB. Toda a la gama	Modelo de color RGB. Toda la gama
Nº DE COLORES	3	6	16,777,216	16,777,216
COLORES				
CONTENIDO TECNOLOGÍA	CONTROL PROGRAMADO (Secuencial y salidas digitales)	CONTROL PROGRAMADO (Condicionales, entradas y salidas digitales)	CONTROL PROGRAMADO (Condicionales y entradas y salidas analógicas)	CONTROL PROGRAMADO (Condicionales y entradas y salidas analógicas)
ACTUADOR	LED RGB	LED RGB	LED RGB	LED RGB
PLACA	ARDUINO UNO	ARDUINO UNO	ARDUINO UNO	ARDUINO UNO
SENSOR	SIN SENSOR	TACTO (Pulsador)	TACTO (Potenciómetro)	ULTRASONIDO
PROGRAMACIÓN	BLOQUES	BLOQUES	BLOQUES Y TEXTO	BLOQUES Y TEXTO
HERRAMIENTA DE SIMULACIÓN	TINKERCAD	TINKERCAD	TINKERCAD	TINKERCAD
HERRAMIENTA PARA PLACA	ARDUINOBLOCKS	ARDUINOBLOCKS	ARDUINOBLOCKS IDE ARDUINIO	ARDUINOBLOCKS IDE ARDUINO