



TECNOLÓGICO
NACIONAL DE MÉXICO®



Reporte de Practicas Arduino

Docente: Miriam Puente Jimenez

Carrera: Ing. Mecatrónica

Semestre: 2°do semestre

Integrantes:

- Gaona Hernandez Camila Lisset
- Guerrero Sandoval Pedro
- López Cruz Azul Alexandra
- Moreno Borjas Brayan
- Sanchez Ramos Angel Eduardo

Lunes 12 de mayo de 2025.

Contenido

Resumen.....	2
Introducción	3
Materiales y métodos	4
Materiales	4
Métodos	9
Resultados	19
Discusión	20
Conclusión	21
Referencias.....	22
Anexos	23

Resumen

La presente práctica tuvo como objetivo aplicar los conocimientos básicos de electrónica y programación mediante el desarrollo de proyectos con la plataforma Arduino. Se implementaron cinco montajes: un semáforo, un juego de luces, un sensor de luz, un LED intermitente y un zumbador musical. Para ello, se utilizaron componentes electrónicos básicos y programación en lenguaje Arduino.

Los principales resultados mostraron el correcto funcionamiento de cada montaje, permitiendo observar la relación entre teoría y práctica. Asimismo, se presentaron dificultades comunes como errores de conexión o de código, los cuales fueron solucionados mediante trabajo en equipo y consulta de fuentes de apoyo.

Se concluye que esta práctica resultó muy efectiva para consolidar habilidades técnicas y fomentar el aprendizaje activo y colaborativo.

Introducción

En el aprendizaje de la electrónica y la programación, la práctica desempeña un papel fundamental para consolidar los conocimientos teóricos adquiridos. En este contexto, el desarrollo de proyectos con Arduino permite a los estudiantes aplicar conceptos clave mediante la construcción de circuitos interactivos y funcionales.

Esta práctica tuvo como finalidad diseñar y programar diversos proyectos electrónicos, como un semáforo, un juego de luces, un sensor de luz, un LED intermitente y un zumbador musical, utilizando Arduino como plataforma principal.

Arduino es una plataforma de prototipado electrónico de código abierto basada en hardware y software fáciles de usar. Su versatilidad y accesibilidad la han convertido en una herramienta educativa clave para la enseñanza de la electrónica, la robótica y la programación en múltiples niveles académicos.







Gracias a su enfoque práctico, permite a los estudiantes experimentar directamente con sensores, actuadores y programación, desarrollando habilidades en lógica, resolución de problemas y diseño de sistemas electrónicos (Banzi & Shiloh, 2014).

El objetivo general de esta práctica fue fomentar el aprendizaje significativo de la electrónica básica mediante la implementación de proyectos con Arduino, desarrollando habilidades en programación, diseño de circuitos y trabajo colaborativo.



Materiales y métodos

Materiales






❖ Zumbador musical

Materiales	Descripción	Imagen
Placa Arduino UNO	Microcontrolador principal que ejecuta el programa.	
Cable USB tipo B	Utilizado para conectar la placa Arduino a la laptop, permitiendo tanto la carga del programa como la alimentación eléctrica del circuito.	
Protoboard	Utilizada para realizar conexiones sin necesidad de soldaduras.	
Zumbador (buzzer) pasivo	Dispositivo que emite sonidos de diferentes frecuencias programadas.	
Resistencia de 220 ohms	Conectada en serie para limitar el flujo de corriente hacia el buzzer y protegerlo.	
Cables macho-macho (2)	Empleados para realizar las conexiones entre la placa Arduino y la protoboard.	

❖ **Sensor de luz**

Material	Descripción	Imagen
Placa Arduino UNO	Interpreta la señal de entrada del sensor de luz y controla la salida al LED.	
Cable USB tipo B	Conecta la placa con la laptop para programación y energía.	
Protoboard	Permite organizar el circuito sin soldaduras.	
LED	Se enciende o apaga dependiendo del nivel de luz captado.	
Foto resistencia (LDR)	Sensor que varía su resistencia según la cantidad de luz recibida.	
Resistencias de 220 ohmios (2)	Una limita corriente al LED, y la otra forma un divisor de voltaje con la foto resistencia.	
Latiguillos macho-macho (6)	Permiten enlazar todos los componentes con la placa Arduino.	

❖ **Leds intermitentes**

Material	Descripción	Imagen
Placa Arduino UNO	Ejecuta el código encargado de hacer parpadear el LED.	
Cable USB tipo B	Sirve para cargar el código desde la computadora y alimentar la placa.	
Protoboard	Base de conexiones rápida y reutilizable.	
LED	Elemento que parpadea al recibir instrucciones de la placa.	
Resistencia de 220 ohms	Protege al LED del exceso de corriente.	
Latiguillos board macho (2)	Cables para conectar el LED y la resistencia con la placa Arduino.	

❖ **Semáforo**

Material	Descripción	Imagen
Placa Arduino UNO	Utilizada como unidad de control para la secuencia de los LEDs.	
Cable USB tipo B	Permite la conexión entre la placa Arduino y la computadora.	
Protoboard	Facilita el montaje temporal del circuito sin necesidad de soldar.	
LEDs (3)	Representan las luces del semáforo.	
Resistencia de 220 ohms (3)	Protege al LED del exceso de corriente.	
Latiguillos board macho (4)	Empleados para realizar las conexiones entre la placa Arduino y los componentes montados en la protoboard.	

❖ **Juego de luces**

Material	Descripción	Imagen
Placa Arduino UNO	controla la secuencia de iluminación de los LEDs.	
Cable USB tipo B	Usado para la conexión y alimentación desde la laptop.	
Protoboard	Proporciona una plataforma para ensamblar el circuito.	
LEDs (11)	Se utilizan para crear diferentes patrones de encendido en secuencia.	
Resistencia de 220 ohms (1)	Protege los LEDs del exceso de corriente.	
Latiguillos board macho (13)	Utilizados para conectar todos los LEDs y el botón con la placa Arduino.	
Push button de 2 pines (1)	Permite la interacción con el usuario (ej. iniciar la secuencia).	

Métodos

❖ Zumbador musical

¿Cómo se programó a la placa de Arduino?

El código se programó en la placa UNO de Arduino utilizando el IDE de Arduino.

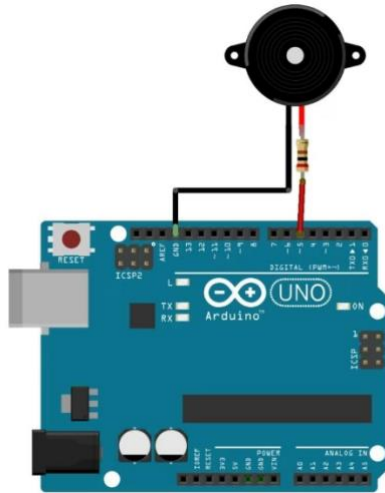
Se conectó un zumbador pasivo al pin digital 5, y a través del código se le enviaron frecuencias correspondientes a notas musicales para reproducir la melodía principal del videojuego de Mario Bros.

El funcionamiento se basa en usar la función `tone()` para generar una señal PWM que produce sonidos en el zumbador, y `noTone()` para detenerlos. Además, se definen dos arreglos: uno con las notas musicales (frecuencias en Hz) y otro con la duración de cada nota (en milisegundos).

Explicación del diagrama (la conexión física)

- La pata larga del zumbador (ánodo) se conecta al pin digital 5 del Arduino.
- La pata corta (cátodo) se conecta a GND (tierra).
- Se utiliza una resistencia de 220 ohmios para limitar la corriente, colocada en serie con el buzzer.
- Se utilizan dos cables para hacer las conexiones entre el zumbador, la resistencia, y la placa Arduino.

Diagrama del circuito:



Fragmento de código:

```
1. int buzzer = 5; // Pin donde está conectado el zumbador

2. void setup() {

3.   tone(buzzer, 262); // Reproduce la nota DO (262 Hz)

4.   delay(500); // La mantiene por 500 milisegundos

5.   noTone(buzzer); // Detiene el sonido

6. }

7. void loop() {

8.   // No se repite, el código suena una vez al iniciar

9. }
```

❖ **Sensor de luz**

El código se programó en la placa UNO de Arduino con un código usando el IDE del Arduino que al acercar una fuente de luz al sensor o al modificar la iluminación ambiental, se observó que el led respondía, se encendía y apagaba en secuencia cada vez que el sensor detectaba un cambio de luz según lo programado que simula un sistema de respuesta visual ante cambios de iluminación. El código asigna lo siguiente:

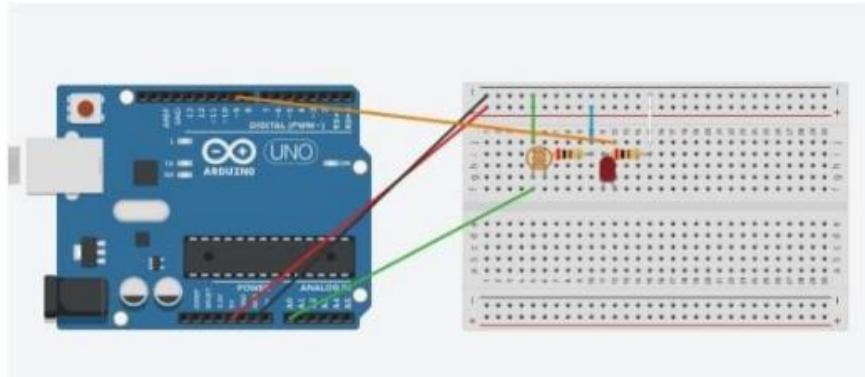
- Un led al pin digital 9.
- Un sensor de luz LDR al pin analógico A0.
- Un valor límite de luminosidad (valor limite = 490) que decide cuándo encender o apagar el led.

El código asigna un led al pin 9 y el sensor de luz al pin A0 y utiliza delay() para que pause el programa(en este caso es de 300 milisegundos) y el digitalWrite() que Enciende(HIGH) y apaga(LOW) el Led según el nivel de luz.

Explicación de diagrama (La conexión física)

- Un extremo del LDR está conectado a la línea positiva de la protoboard, que a su vez está conectada al pin de 5V del Arduino Uno.
- El otro extremo del LDR está conectado a un punto intermedio de la protoboard y a ese mismo punto se conecta una resistencia que va hacia la línea de tierra (GND).
- El nodo de unión entre el LDR y la resistencia se conecta con un cable amarillo al pin A0 del Arduino. Esto permite leer el voltaje generado por el divisor de voltaje y así detectar el nivel de luz.
- El ánodo (pata larga) del LED está conectado a una resistencia y de ahí al pin digital 9 del Arduino.
- El cátodo (pata corta) del LED va conectado directamente a la línea de tierra (GND) de la protoboard.

Diagrama de circuito:



Fragmento del código:

1. `#define LED 9` // El LED está conectado al pin digital 9
2. `#define LDR 0` // El LDR (sensor de luz) está conectado al pin analógico A0

❖ Leds intermitentes

¿Cómo se programó la placa Arduino?

La placa Arduino UNO se programó con un código en lenguaje C++ (usando el IDE de Arduino) que controla un LED intermitente, encendiéndolo y apagándolo continuamente con intervalos de tiempo definidos. Este proyecto simula el parpadeo de una luz de advertencia o señal.

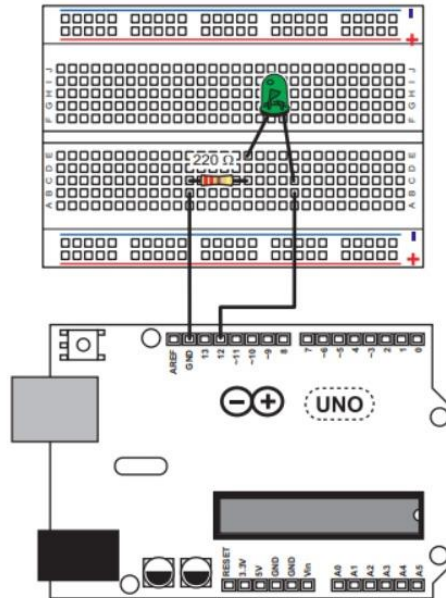
El código asigna un pin digital (12) al LED y utiliza funciones como `digitalWrite()` y `delay()` para encender y apagar el LED con una duración específica.

Explicación del diagrama (la conexión física)

El diagrama conecta un LED (verde, por ejemplo) a una protoboard (breadboard) y a la placa Arduino UNO:

- El cátodo del LED (pierna corta) se conecta a GND (tierra).
- El ánodo del LED (pierna larga) se conecta a través de una resistencia de 220 ohmios al pin digital 12 de la placa Arduino.
- La resistencia limita la corriente para evitar dañar el LED.

Diagrama del circuito:



Fragmento del código:

1. `digitalWrite(12, HIGH); // Enciende el LED`
2. `delay(500); // Espera medio segundo`

❖ **Semáforo**

¿Cómo se programó la placa Arduino?

La placa Arduino UNO se programó con un código en lenguaje C++ (usando el IDE de Arduino) que controla tres LEDs simulando el funcionamiento de un semáforo. Cada LED representa una luz:

- Rojo: Detenerse
- Amarillo: Precaución
- Verde: Avanzar

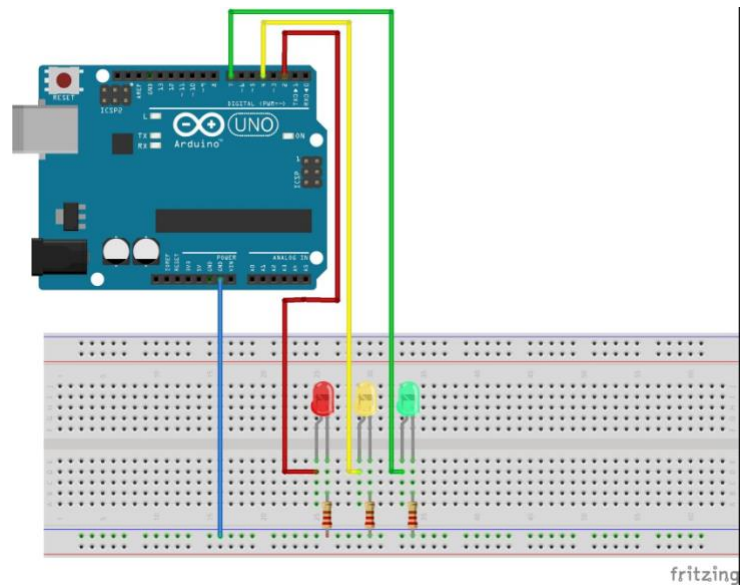
El código asigna un pin digital a cada color y utiliza funciones como `digitalWrite()` y `delay()` para encender/apagar los LEDs con tiempos específicos.

Explicación del diagrama (la conexión física)

El diagrama conecta los tres LEDs (rojo, amarillo y verde) en una protoboard (breadboard) y a la placa Arduino:

- Los pines digitales 2, 4 y 7 se conectan a las patas positivas (ánodos) de cada LED.
- Las patas negativas (cátodos) se conectan a tierra (GND) a través de resistencias para limitar la corriente.
- El orden en la protoboard es importante solo para facilitar el armado, pero puede variar mientras las conexiones sean correctas.

Diagrama de circuito:



Fragmento del código:

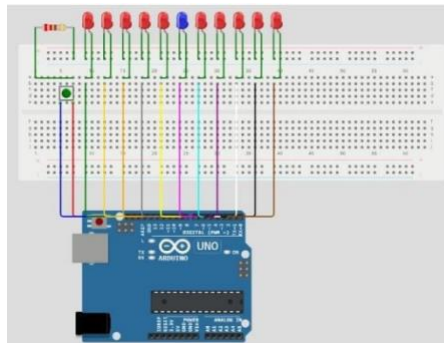
1. `int rojo = 2; // LED rojo conectado al pin digital 2`
2. `int amarillo = 4; // LED amarillo conectado al pin digital 4`
3. `int verde = 7; // LED verde conectado al pin digital 7`

❖ Juego de luces

Explicación:

El circuito consiste en 11 LEDs colocados en fila, 5 LEDs rojos van colocados del lado izquierdo y derecho, y en medio de esos de ellos va colocado un LED azul. Todos ellos a la orilla del Protoboard con su terminal negativa conectada al lado negativo del Protoboard y las positivas en los carriles neutros, después con cables macho a macho se van poniendo enfrente de las terminales positivas del LED y colocando al Arduino UNO desde el GND hasta la terminal 2 para su funcionamiento. También a lado de estos se colocó una resistencia de 220 Ohms y un push button en el cual cada uno de sus pines van conectados uno a positivo y el otro a negativo en el Arduino.

Diagrama de circuito:



Fragmento del código:

1. `const int leds[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};` // Arreglo con los pines de los LEDs
2. `const int pinBoton = 13;` // Pin asignado al botón

Resultados

- **Zumbador musical:** Se logró reproducir correctamente la melodía del videojuego Mario Bros. El buzzer respondió a las frecuencias programadas y el sonido fue claro y distinguible. Se observó que una programación precisa de los retardos influye directamente en la calidad del sonido.
- **Sensor de luz (LDR):** El LED respondió adecuadamente a los cambios de iluminación. Al cubrir el sensor o modificar la luz ambiental, se activaba o desactivaba la salida del LED. Esto evidenció la utilidad del divisor de voltaje y la lectura de pines analógicos para controlar salidas digitales.
- **LED intermitente:** El LED parpadeó con intervalos regulares, validando el uso del comando `delay()` y la lógica de control digital. Se verificó su utilidad como sistema de señalización visual.
- **Semáforo:** La secuencia programada (verde – amarillo – rojo) se ejecutó de forma cíclica. Este montaje permitió simular un sistema real de control de tráfico. Se comprobó la importancia del temporizado correcto y el orden en las conexiones.
- **Juego de luces:** La secuencia de encendido progresivo de los 11 LEDs funcionó adecuadamente, incluso tras repetidas pulsaciones del botón. Esto demostró el uso eficiente de arreglos, ciclos `for` y lectura de entradas digitales.

Discusión

Durante el desarrollo de los proyectos con Arduino, fue posible observar cómo conceptos teóricos como el control de flujo, la manipulación de entradas y salidas digitales, y el uso de componentes electrónicos básicos (resistencias, LEDs, pulsadores, sensores y zumbadores) se aplican de manera directa en el montaje de circuitos funcionales. Por ejemplo, al programar el semáforo, se aplicaron estructuras de control condicional y temporizadores que previamente se habían estudiado en clase, lo cual facilitó la comprensión del uso práctico del código.

Entre las dificultades más relevantes que enfrentamos se encontraron la conexión incorrecta de componentes, errores en la programación (como el uso de variables mal declaradas o ciclos mal definidos), y la necesidad de calcular resistencias adecuadas para proteger los componentes. Estas barreras fueron superadas mediante la revisión en equipo, el apoyo de recursos digitales y el ensayo-error, lo cual fortaleció nuestro aprendizaje colaborativo.

Además, cada proyecto nos permitió reafirmar conocimientos clave: en el sensor de luz se comprendió el funcionamiento de los sensores analógicos y la variación de valores en función de la iluminación; el zumbador musical nos permitió aplicar la generación de frecuencias; y el juego de luces nos sirvió para practicar secuencias lógicas en bucles de programación.

Conclusión

La práctica con Arduino resultó ser una herramienta altamente efectiva para consolidar los conocimientos teóricos en electrónica básica. A través de la elaboración de proyectos sencillos pero significativos, pudimos observar el comportamiento real de los componentes y la importancia de una programación correcta.

Entre los hallazgos clave destacan:

- La comprensión práctica de los pines digitales y analógicos en Arduino.
- La importancia de planificar los esquemas eléctricos antes del montaje.
- El valor del trabajo en equipo para superar obstáculos técnicos.

Estas experiencias no solo reforzaron nuestras habilidades técnicas, sino que también impulsaron nuestra capacidad para resolver problemas, adaptarnos y aprender de los errores.

Referencias

Banzy, M., & Shiloh, M. (2014). Getting Started with Arduino (3rd ed.). Maker Media, Inc.

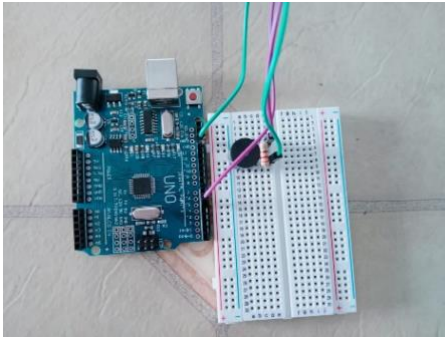
Monk, S. (2016). Programming Arduino: Getting Started with Sketches (2nd ed.). McGraw-Hill

Education.

Arduino. (n.d.). Arduino Official Documentation. Retrieved from
<https://www.arduino.cc/en/Guide/HomePage>

Anexos

Zumbador musical



```
1 // Pin del buzzer pines
2 const int buzzer = 5;
3
4 // Notas musicales (Frecuencias en Hz)
5 #define NOTE_B0 31
6 #define NOTE_C1 33
7 #define NOTE_C#1 35
8 #define NOTE_D1 37
9 #define NOTE_D#1 39
10 #define NOTE_E1 41
11 #define NOTE_F1 44
12 #define NOTE_F#1 46
13 #define NOTE_G1 49
14 #define NOTE_G#1 52
15 #define NOTE_A1 55
16 #define NOTE_A#1 58
17 #define NOTE_B1 62
18 #define NOTE_C2 65
19 #define NOTE_C#2 69
20 #define NOTE_D2 73
21 #define NOTE_D#2 78
22 #define NOTE_E2 82
23 #define NOTE_F2 87
24 #define NOTE_F#2 93
25 #define NOTE_G2 98
26 #define NOTE_G#2 104
```

```
27 #define NOTE_A2 110
28 #define NOTE_A#2 117
29 #define NOTE_B2 123
30 #define NOTE_C3 131
31 #define NOTE_C#3 139
32 #define NOTE_D3 147
33 #define NOTE_D#3 156
34 #define NOTE_E3 165
35 #define NOTE_F3 175
36 #define NOTE_F#3 185
37 #define NOTE_G3 196
38 #define NOTE_G#3 208
39 #define NOTE_A3 220
40 #define NOTE_A#3 233
41 #define NOTE_B3 247
42 #define NOTE_C4 262
43 #define NOTE_C#4 277
44 #define NOTE_D4 294
45 #define NOTE_D#4 311
46 #define NOTE_E4 330
47 #define NOTE_F4 349
48 #define NOTE_F#4 370
49 #define NOTE_G4 392
50 #define NOTE_G#4 415
51 #define NOTE_A4 440
```

```
52 #define NOTE_A4 440
53 #define NOTE_B4 494
54 #define NOTE_C5 523
55 #define NOTE_C#5 554
56 #define NOTE_D5 587
57 #define NOTE_D#5 622
58 #define NOTE_E5 659
59 #define NOTE_F5 698
60 #define NOTE_F#5 740
61 #define NOTE_G5 784
62 #define NOTE_G#5 831
63 #define NOTE_A5 880
64 #define NOTE_A#5 932
65 #define NOTE_B5 988
66 #define NOTE_C6 1047
67 #define NOTE_C#6 1109
68 #define NOTE_D6 1175
69 #define NOTE_D#6 1245
70 #define NOTE_E6 1319
71 #define NOTE_F6 1397
72 #define NOTE_F#6 1480
73 #define NOTE_G6 1568
74 #define NOTE_G#6 1661
75 #define NOTE_A6 1760
76 #define NOTE_A#6 1865
77 #define NOTE_B6 1976
```

```
78 #define NOTE_C7 2093
79 #define NOTE_C#7 2217
80 #define NOTE_D7 2349
81 #define NOTE_D#7 2489
82 #define NOTE_E7 2637
83 #define NOTE_F7 2794
84 #define NOTE_F#7 2969
85 #define NOTE_G7 3156
86 #define NOTE_G#7 3352
87 #define NOTE_A7 3529
88 #define NOTE_A#7 3729
89 #define NOTE_B7 3951
90 #define NOTE_C8 4196
91 #define NOTE_C#8 4455
92 #define NOTE_D8 4639
93 #define NOTE_D#8 4978
94
95 // Canción principal de Mario Bros (melodía + duración)
96 int melody[] = {
97   NOTE_E7, NOTE_F7, 0, NOTE_E7,
98   0, NOTE_C7, NOTE_E7, 0,
99   NOTE_C7, 0, 0, 0,
100  NOTE_G6, 0, 0, 0,
101  NOTE_C7, 0, 0, NOTE_G6,
102  0, 0, NOTE_E6, 0,
103
104  0, NOTE_A6, 0, NOTE_B6,
105  0, NOTE_A6, NOTE_A6, 0,
106
107  NOTE_G6, NOTE_F7, NOTE_E7,
108  NOTE_A7, 0, NOTE_F7, NOTE_E7,
109  0, NOTE_E7, 0, NOTE_C7,
110  NOTE_D7, NOTE_B6, 0, 0
111 };
112
113 int noteDurations[] = {
114   125, 125, 125, 125,
115   175, 125, 125, 125,
116   125, 125, 125, 125,
117   125, 125, 125, 125,
118
119   125, 125, 125, 125,
120   125, 125, 125, 125,
121   125, 125, 125, 125,
122   125, 125, 125, 125,
123
124   83, 83, 125,
125   125, 125, 125, 125,
126   125, 125, 125, 125,
127   125, 125, 125, 125
128 };
129
130 void setup() {
131   for (int thisNote = 0; thisNote < sizeof(melody)/sizeof(int); thisNote++) {
132     int noteDuration = noteDurations[thisNote];
133     if (melody[thisNote] == 0) {
134       delay(noteDuration);
135     } else {
136       tone(buzzer, melody[thisNote], noteDuration);
137       delay(noteDuration * 1.3);
138       noTone(buzzer);
139     }
140   }
141 }
142
143 void loop() {
144   // Espera un segundo y repite la canción
145   delay(1000);
146   setup();
147 }
```

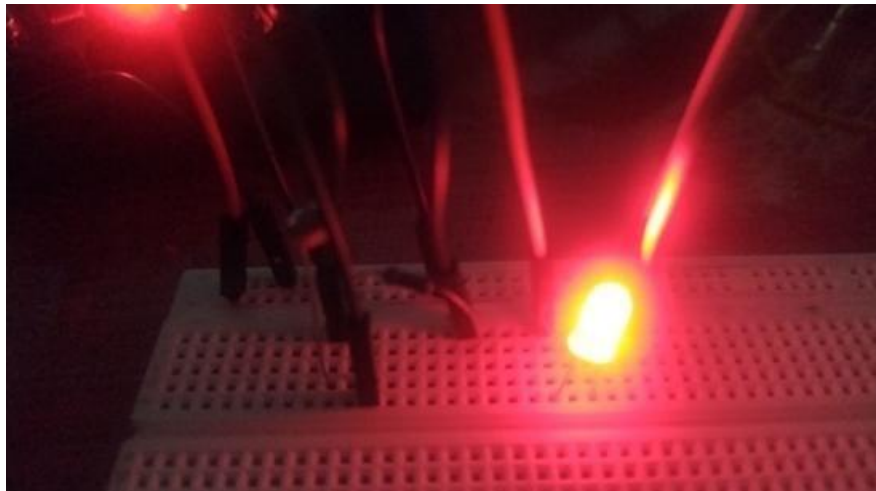
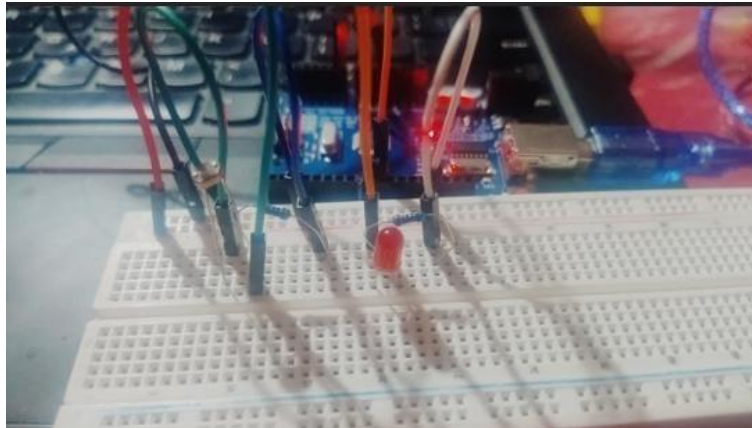

Led intermitente



Leds_Intermitentes

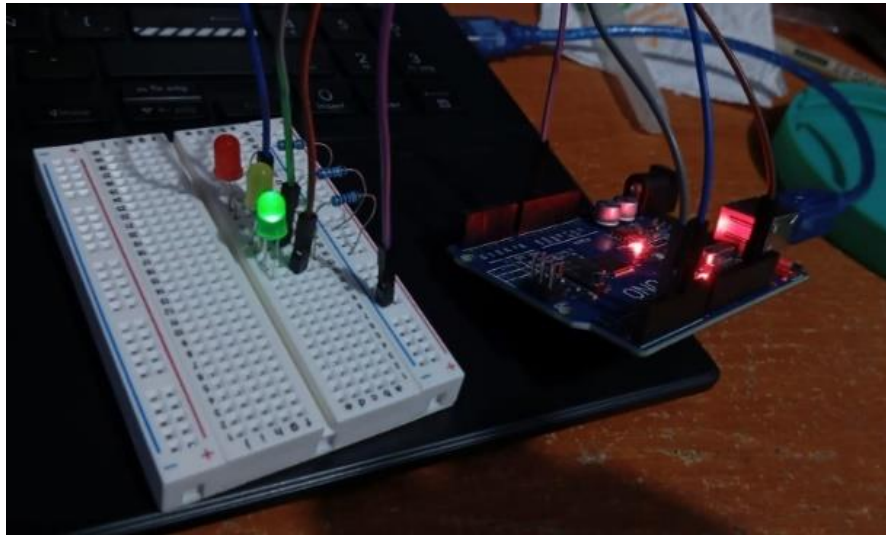
```
void setup() {  
    pinMode(12, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(12, HIGH);  
    delay(500);  
    digitalWrite(12, LOW);  
    delay(500);  
}
```

Sensor de luz:



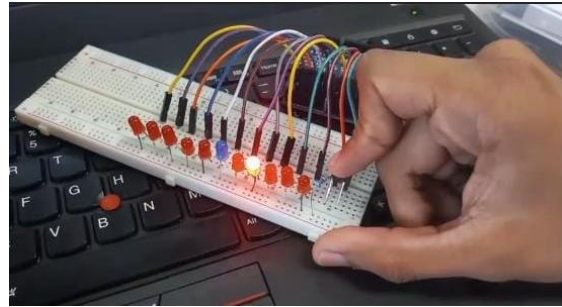
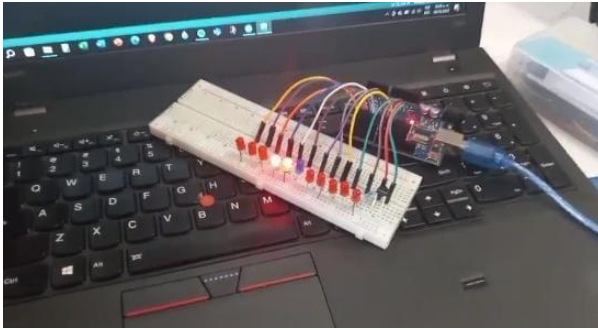
```
sketch_may12a | Arduino IDE 2.3.6
File Edit Sketch Tools Help
[Icons] Arduino Uno
sketch_may12a.ino
1 #define LED 9 //El LED esta conectado en el pin 9
2 #define LDR 0 //El LDR esta conectado en el pin A0
3
4 int luz = 0;
5 int valor_sensor = 0;
6 int valor_limite = 490; //Este valor hara que el LED cambie de estado a una determinada luminosidad (podeis con distintos valores para ajustar l
7
8
9 void setup() { //Configuracion de los pin como entrada o salida
10   pinMode(LED, OUTPUT);
11   pinMode(LDR, INPUT);
12 }
13
14 void loop() { //Configuracion de los valores del LDR
15 {
16   valor_sensor = analogRead(LDR);
17   luz = (5.0 * valor_sensor * 100.0)/1024.0; //Hacemos que el valor luz este entre 0 y 1023, es decir 1024 valores.
18   delay(300);
19
20   if (luz <= valor_limite) //Si el valor de luz es menor o igual que el valor limite
21   {
22     digitalWrite(LED, LOW); //El led se apaga
23   }
24   if (luz > valor_limite) //Si es mayor que el valor limite
25   {
26     digitalWrite(LED, HIGH); //El led se enciende
27   }
28 }
```

Semáforo:



```
semáforo | Arduino IDE 2.3.6
File Edit Sketch Tools Help
└─ Arduino Uno
semáforo.ino
1  /* SEMAFORO */
2  int rojo=2; //definimos el valor del pin para el led rojo
3  int amarillo=4; //definimos el valor del pin para el led amarillo
4  int verde=7; //definimos el valor del pin para el led verde
5
6  /** Programa */
7
8  void setup() {
9    pinMode(verde,OUTPUT); //declaramos el pin verde como salida
10   pinMode(amarillo,OUTPUT); //declaramos el pin amarillo como salida
11   pinMode(rojo,OUTPUT); //declaramos el pin rojo como salida
12 }
13
14 void loop() {
15   digitalWrite(verde,HIGH); //encendemos el led rojo
16   delay(2000); //esperamos 2 segundos
17   digitalWrite(verde,LOW); //apagamos el led rojo
18   delay(500); //esperamos medio segundo
19
20   digitalWrite(amarillo,HIGH); //encendemos el led amarillo
21   delay(2000); //esperamos 2 segundos
22   digitalWrite(amarillo,LOW); //apagamos el led amarillo
23   delay(500); //esperamos medio segundo
24
25   digitalWrite(rojo,HIGH); //encendemos el led verde
26   delay(2000); //esperamos 2 segundos
27   digitalWrite(rojo,LOW); //apagamos el led verde
28   delay(500); //esperamos medio segundo
29 }
Output
Ln 6, Col 1 Arduino Uno on COM3 (not connected)
```

Juego de luces:



```
Proyecto_Juego_de_Luces | Arduino IDE 2.3.5
File Edit Sketch Tools Help

Proyecto_Juego_de_Luces.ino
1 const int leds[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
2 const int numerosleds = 11;
3 const int ledjezo = 4;
4
5 const int pinboton = 13;
6
7 int ledactual = 0;
8 int ledanterior = -1;
9 bool subiendo = true;
10 unsigned long tiempoanterior = 0;
11 unsigned long tiempoboton = 0;
12 const int intervalo = 300; // tiempo entre cambios de LED (en ms)
13
14 const int debounceTiempo = 50;
15 bool botonPresionado = false;
16 bool juegoEncurso = true;
17
18 void setup() {
19   // put your setup code here, to run once:
20   for (int i = 0; i < numerosleds; i++) {
21     pinMode(leds[i], OUTPUT);
22     digitalWrite(leds[i], LOW);
23   }
24   pinMode(pinboton, INPUT_PULLUP);
25 }
26
Output
```

```
Proyecto_Juego_de_Luces | Arduino IDE 2.3.5
File Edit Sketch Tools Help

Proyecto_Juego_de_Luces.ino
27 void loop() {
28   // put your main code here, to run repeatedly:
29   if (juegoEncurso) {
30     unsigned long tiempoactual = millis();
31     if (tiempoactual - tiempoanterior > intervalo) {
32       tiempoanterior = tiempoactual;
33
34       if (ledanterior >= 0) {
35         digitalWrite(leds[ledanterior], LOW);
36       }
37
38       digitalWrite(leds[ledactual], HIGH);
39       ledanterior = ledactual;
40
41       if (subiendo) {
42         ledactual++;
43         if (ledactual == numerosleds) {
44           ledactual = numerosleds - 2;
45           subiendo = false;
46         }
47       } else {
48         ledactual--;
49         if (ledactual < 0) {
50           ledactual = 1;
51           subiendo = true;
52         }
53       }
54
55       if (digitalRead(pinboton) == LOW && (tiempoactual - tiempoboton > debounceTiempo)) {
56         tiempoboton = tiempoactual;
57         for (int i = 0; i < 10; i++) {
58           digitalWrite(leds[ledanterior], LOW);
59           digitalWrite(leds[ledactual], LOW);
60           delay(200);
61           digitalWrite(leds[ledactual], HIGH);
62           delay(200);
63         }
64         delay(2000);
65       }
66     }
67   }
68 }
69
Output
```

```
Proyecto_Juego_de_Luces | Arduino IDE 2.3.5
File Edit Sketch Tools Help

Proyecto_Juego_de_Luces.ino
41 if (ledactual == numerosleds) {
42   ledactual = numerosleds - 2;
43   subiendo = false;
44 }
45
46 } else {
47   ledactual--;
48   if (ledactual < 0) {
49     ledactual = 1;
50     subiendo = true;
51   }
52 }
53
54
55 if (digitalRead(pinboton) == LOW && (tiempoactual - tiempoboton > debounceTiempo)) {
56   tiempoboton = tiempoactual;
57   for (int i = 0; i < 10; i++) {
58     digitalWrite(leds[ledanterior], LOW);
59     digitalWrite(leds[ledactual], LOW);
60     delay(200);
61     digitalWrite(leds[ledactual], HIGH);
62     delay(200);
63   }
64   delay(2000);
65 }
66
67 }
68 }
69
Output
```

https://youtu.be/exE_RudeBs