TEC | Tecnológico de Costa Rica

Escuela de Ingeniería Electromecánica

Ingeniería en Mantenimiento Industrial

Instructivo de Laboratorio

Laboratorio de Control Eléctrico

5 de septiembre de 2025
Versión: 0.2

Dr.-Eng. Luis Diego Murillo
Dr.-Eng. Juan J. Rojas

Índice general

Laboratorio 1

Introducción a Arduino

1.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Desarrollar un programa sencillo en la plataforma de Arduino.
- Implementar en C las funciones lógicas AND, OR, XOR, NAND y NOR

1.2. Materiales y equipo

- 1 Arduino UNO MINIMA R4.
- 1 Multímetro.
- 5 Resistencias de 270 o 330 Ω.
- 2 Resistencias de 1 kΩ
- 2 interruptores pulsadores.
- 1 Protoboard.
- 5 Diodos emisor de luz (LEDs).
- 2 Generadores de funciones.
- 2 Osciloscopio.
- 1 Computadora portátil.

1.3. Procedimiento

Este laboratorio tiene una duración de 4 lecciones, repartidas en dos semanas. Los estudiantes deben mostrar durante las clases programadas las tres actividades propuestas. Deben recabar fotografías y resultados de los equipos de medición para elaborar las evidencias. Las evidencias se subirán al TecDigital la semana siguiente finalizadas las actividades.

1.4. Práctica en Clase

1.4.1. Actividad 1

El código del apéndice B.1 apaga y enciende un LED según el estado de un botón. La figura 1.1 muestra el esquema básico de conexión.
Elimine el botón y alimente el Arduino con un generador de funciones que brinde una señal cuadrada de 0 a 5 voltios. Conecte los dos canales del osciloscopio en la entra y salida. Recuerde que todas las tierras deben estar conectadas entre si.

Conteste las preguntas:

¿Cuanto es el retardo de la señal? ¿Cual es la frecuencia de la señal de entrada en que la señal de salida se distorsiona? Es decir, la señal de salida no es inversa a la señal de entrada. Recuerde capturar las pantallas del osciloscopio en ambos casos.

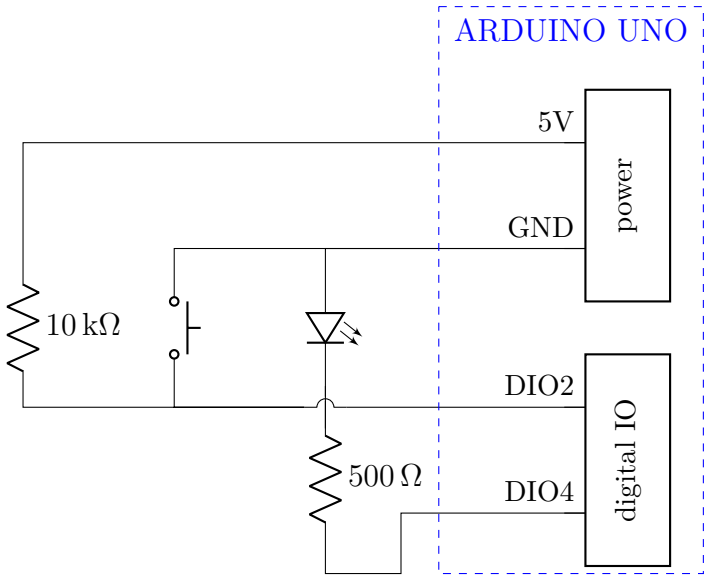

Figura 1.1: Conexión de circuito para Actividad 2

1.4.2. Actividad 2

Esta actividad permite analizar el comportamiento de las conectivas lógicas AND, OR, XOR, NAND, y NOT, y exportar los resultados al puerto serial, así como mostrar los resultados mediante los LEDs conectados a los pines $\{4, 5, 6, 7, 8\}$; el código de ejemplo se muestra en el apéndice B.2. Las entradas declaradas mediante los pines $\{2, 3\}$ reciben las señales de dos generadores de funciones. Los generadores brindan una señal triangular que oscila ente $[0 - 5]$ voltios, debe ajustar cuidadosamente con el osciloscopio cada señal. La señal del pin 2 tendrá una frecuencia del doble del pin 3. Adicionalmente las señales triangulares alimentaran el convertidor Analógico-Digital (ADC) mediante los pines A0 y A1. Conecte su Arduino según el esquema de la Figura 1.3.

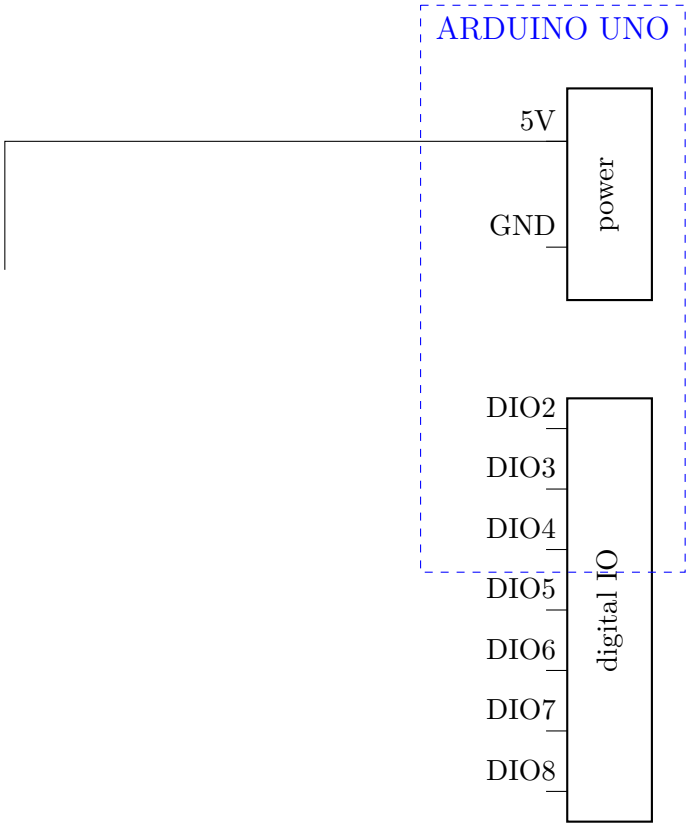

Figura 1.2: Conexión de circuito para Actividad 3

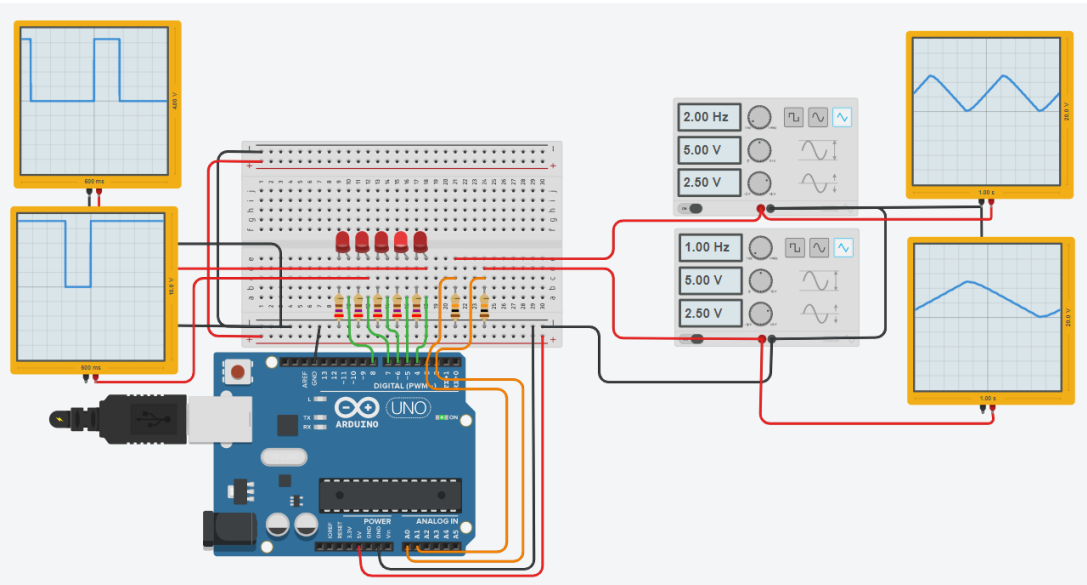

Figura 1.3: Esquema de conexión del Arduino, Generador de funciones y Osciloscopio

El código del Anexo B implementa las funciones lógicas AND, OR. Un repaso de como implementar funciones en C se muestra en este enlace.

Conteste las preguntas:

Guarde los datos del puerto serial en un archivo .TXT e importelos en MS EXCEL para graficar los datos. Se recomienda utilizar el monitor serial de MS CODE STUDIO dado que este permite salvar los datos. ¿Se logra apreciar las señales triangulares y digitales? ¿Las gráficas de las señales digitales de entrada y salida cumplen las tablas de verdad de las conectivas lógicas? ¿Cual es el voltaje de entrada en bajo máximo $V_{IL}max$?, ¿Cual es el voltaje de entrada en alto mínimo $V_{IH}min$?, ¿Cual es el error que presenta las mediciones de los voltajes ?, ¿Si desea un error de ± 1 mV, de cuantos bits debe ser el ADC?¿Explique?

Laboratorio 2

Programación de funciones combinacionales.

2.1. Objetivos

Al finalizar este laboratorio el estudiante estará en capacidad de:

- Programar funciones booleanas combinacionales en Arduino.
- Verificar la compuertas NAND y NOR son en si mismas un conjunto Universal, equivalentes a el conjunto de conectivas lógicas OR, AND y NOT.
- Verificar que a partir de los mintérminos de una expresión lógica se obtienen los circuitos con compuertas NAND y de los maxtérminos, los circuitos con compuertas NOR.
- Verificar que para obtener la solución mínima de un circuito lógico, es necesario sintetizar los circuitos tanto por mintérminos como por maxtérminos.
- Programar multiplexores 4×1 y 8×1.

2.2. Materiales y equipo

- 1 Arduino UNO o equivalente: MEGA o ESP32.
- 1 Multímetro.
- 10 Resistencias de 270 o 330 Ω.
- 5 Resistencias de 1kΩ.
- 10 interruptores pulsadores.
- 1 Protoboard.
- 10 Diodes emisor de luz (LEDs).
- 1 Computadora portátil.

2.3. Marco de referencia

Un circuito combinacional digital es un circuito que produce una o varias salidas en función de sus entradas actuales. Es decir, no mantiene ningún estado interno y su salida depende exclusivamente de sus entradas en el momento actual. Existen muchas formas de implementar circuitos combinacionales, a partir de contactos eléctricos, válvulas neumáticas, transistores, software, etc. Sin embargo cualquier circuito lógico combinacional se puede expresar mediante cuatro formas equivalentes que describen su funcionamiento: ecuación lógica, tabla de verdad, diagrama de tiempo, y circuito eléctrico. Puede encontrar información adicional en el siguiente enlace https://es.wikipedia.org/wiki/Sistema_combinacional.

2.3.1. Implementan de expresiones Booleanas

Un circuito combinacional puede ser representado de distintas formas, pero la más común es mediante una ecuación lógica, por ejemplo:

$$F(a,b,c) = a \cdot b + \bar{a} \cdot \bar{b} \cdot c \tag{2.1}$$

La expresión anterior posee una representación en suma de productos (SDP), puede ser implementada en C de la siguiente forma:

```
bool F(bool a,bool b,bool c){
  return  (a && b) || (!a && !b && c);
}
```

Existen ocho formas estándar de implementar la ecuación (2.1) digital-mente y por consecuencia en software . Otras maneras de implementar las funciones lógicas son con el producto de sumas (PDS), las implementaciones a dos niveles NAND/NAND y NOR/NOR, todas las implementaciones deben arrojar los mismos resultados.
Por otra parte, la ecuación (2.1) es equivalente a la expresión booleana NAND/NAND si se niega dos veces y se aplica el Teorema de Morgan.

$$F(a,b,c) = a \cdot b + \bar{a} \cdot \bar{b} \cdot c \tag{2.2}$$

$$F(a,b,c) = \overline{\overline{a \cdot b + \bar{a} \cdot \bar{b} \cdot c}} \tag{2.3}$$

$$F(a,b,c) = \overline{\overline{a \cdot b} \cdot \overline{\bar{a} \cdot \bar{b} \cdot c}} \tag{2.4}$$

Notece que la ecuacion (2.4) necesita conectivas lógicas (funciones) de 2 y 3 entradas. Esto no es un problema cuando se implementa en SOFTWARE, pero sí se requiere hacer una implementación fisica hay que aplicar el Algebra Booleana para dejar la expresión en términos de conectivas de solamente dos entradas. Lo que se hace es que el término de tres literales, le aplicamos una doble negación tal como sigue.

$$F(a,b,c) = \overline{\overline{a \cdot b} \cdot \overline{\overline{\bar{a} \cdot \bar{b} \cdot c}}} \tag{2.5}$$

La expresión (2.5) se encuentra en términos de conectivas NAND de dos entradas y se puede implementar en ARDUINO de la siguiente forma:

```
bool F(bool a, bool b, bool c){
  bool result = false;
  result = NAND(NAND(a,b),NAND(NAND(NAND(NAND(a,true),NAND(b,true)),true),c));
  return result;
}
```

donde la NAND de dos entradas es:

```
bool NAND(bool x, bool y){
  return !(x && y);
}
```

2.3.2. Multiplexores y Decodificadores

Un multiplexor (Mux) es un circuito combinacional con 2^n entrada, más n entradas selectoras o de control y una salida. Este circuito combinacional selecciona con las n entradas control, el valor booleano de una entrada entre $[0, 2^n - 1]$ y coloca dicho valor en la salida. La ecuación general de cualquier multiplexor es la siguiente,

$$MUX = \sum_{i=0}^{2^n-1} I_i \cdot \mathbb{S}_i \tag{2.6}$$

donde $\mathbb{S}_i$ es la i-ésima permutacion de las entradas de selección. A modo de ejemplo un Mux 2×1 posee dos entradas digitales, una de control y una salida. Un Mux 4×1 posee cuatro entradas digitales, dos de control y una salida, Un Mux 8×1 posee ocho entradas digitales, tres de control y una salida.
La ecuación de un Mux 4×1 es la siguiente,

$$F(I_0, I_1, I_2, I_3, S_1, S_0) = I_0\bar{S}_1\bar{S}_0 + I_1\bar{S}_1S_0 + I_2S_1\bar{S}_0 + I_3S_1S_0 \tag{2.7}$$

El código para implementar dicho multiplexor es similar al siguiente,

```
bool F(bool I0,bool I1,bool I2,bool I3,bool S1,bool S0){
  return (I0&&!S1&&!S0 || I1&&!S1&&S0 || I2&&S1&&!S0 || I3&&S1&&S0) ;
}
```

Los circuitos que realizan función inversa del multiplexor se llama Demultiplexores y usualmente tiene una entrada digital más n entradas de selección y 2^n salidas digitales.
Otro ejemplo de circuito combinacional son los Decodificadores y Codificadores. Un codificador es un circuito combinacional con 2^n entradas y n salidas, cuya misión es presentar en la salida el código binario correspondiente a la entrada activada. Mientras que un decodificador hace el trabajo inverso, recibe un código binario y lo transforma en cualquier otro código, ya sea binarios como el Gray o exceso 3; o otras reprentaciones numericas: octal, decimal, hexagecimal. La figura 2.1 muestra la implementan digital de un decodificador de 2 a 4 líneas, la tabla de verdad del circuito y las ecuaciones características de cada salida.
Una posible implementación para Arduino de este decodificador es la siguiente.

```
byte  DECO (bool A0, bool A1)
{
  int i=0;
  bool D[4]={false, false, false, false};
  byte deco=0;
```

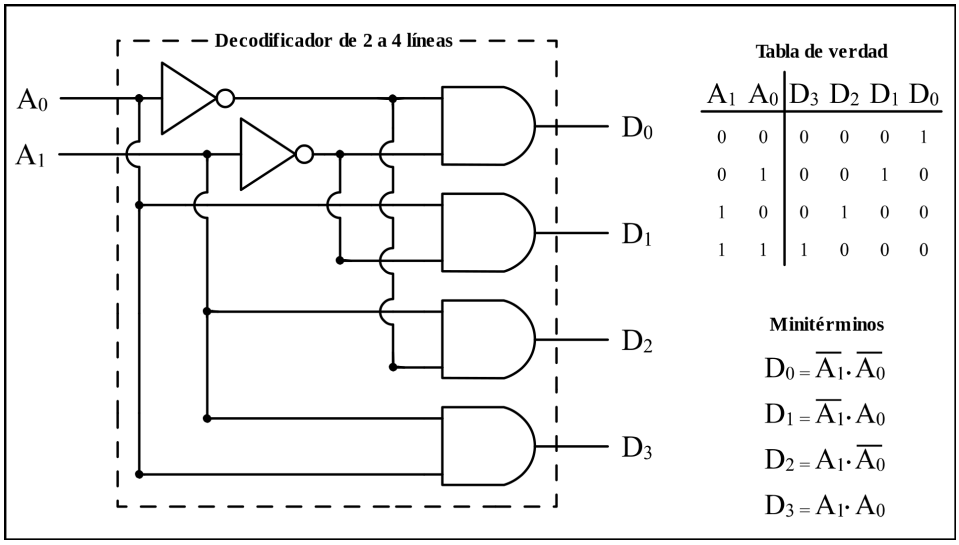

Figura 2.1: Decodificador de código binario de dos bits a cuatro líneas de salida.

```
  D[0]=!A1 && !A0;
  D[1]=!A1 && A0;
  D[2]=A1 && !A0;
  D[3]=A1 && A0;

  for (i=0; i<sizeof(D);i++)
  {
    bitWrite(deco, i, D[i]);
  }
return deco;
}
```

2.4. Metodología

Este laboratorio tiene una duración de 4 lecciones, repartidas en dos semanas. Los estudiantes deben mostrar durante las clases programadas las tres actividades propuestas. Deben recabar fotografías y resultados de los equipos de medición para elaborar las evidencias. Las evidencias se subirán al TecDigital la semana siguiente finalizadas las actividades.

2.5. Práctica en Clase

2.5.1. Actividad 1

Programe en Arruino una lógica combinacional que resuelva el siguiente problema:

- En una planta industrial se desea automatizar un motor y una alarma de acuerdo a 4 sensores llamados a, b, c y d

- La señal del motor se representa por la función lógica $f1$, y se activará cuando los sensores a y b estén encendidos, o cuando los sensores b y d estén encendidos, o cuando los sensores a, c y d están encendidos.

$$f1 = ab + bd + acd$$

- La señal de alarma se representa con la función lógica $f2$, y se activará siempre que estén todos los sensores apagados excepto a o todos apagados excepto c o estén encendidos c y d y el resto apagados. También la alarma se enciende cuando todos los sensores están encendidos o cuando están todos encendidos excepto d o todos encendidos excepto c.

$$f2 = a\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c\bar{d} + \bar{a}\bar{b}cd + abcd + abc\bar{d} + ab\bar{c}d$$

$$f2 = a\bar{b}\bar{c}\bar{d} + \bar{a}\bar{b}c + abc + abd$$

- Los pines $\{2, 3, 4, 5\}$ serán las entradas digitales $\{a, b, c, d\}$.
- Los pines $\{6, 7, 8\}$ se reservan para $f1$. Se deben implementar tres funciones: la suma de productos (SDP), el producto de sumas (PDS) y la función resuelta por conectivas $NAND/NAND$.
- Los pines $\{8, 9, 10\}$ se reservan para $f2$. Se deben implementar tres funciones: la suma de productos (SDP), el producto de sumas (PDS) y la función resuelta por conectivas NOR/NOR.

Conteste las preguntas:

¿Cómo sería el circuito digital de cada una de las funciones implementadas? ¿Como es la tabla de verdad experimental de cada salida versus la tabla teórica?

2.5.2. Actividad 2

Programe las funciones de un Multiplexor 2x1, 4x1 y 8x1. Por otra parte, existe un circuito combinacional que se comporta como la tabla de verdad mostrada en la Tabla 2.1.

Tabla 2.1: Comportamiento esperado de la estructura lógica.

a	b	$F(a,b)$
0	0	$c + d$
0	1	$\overline{c + d}$
1	0	$\overline{c \cdot d}$
1	1	$c \oplus d$

Conteste las preguntas:

¿El comportamiento de un multiplicador 8×1 coincide con la tabla teórica? En la funcion Loop() , llame la función del MUX 8×1 e ingrese parámetros $\{a, b, c, d\}$ de tal forma que se comporte igual que la tabla de verdad presentada en el Cuadro 2.1.¿Tiene el mismo comportamiento? ¿Es posible implementar la T.V. con dos MUX 4 y un MUX 2?¿Como se implementa?

Apéndice A

Marcos de referencia

A.1. Arduino

Un micro-controlador es un circuito integrado pequeño que contiene un microprocesador, memoria y periféricos como entradas/salidas (IO), comunicación, almacenamiento y sensores. Estos componentes están integrados en un solo chip y pueden ser programados para controlar diferentes dispositivos y sistemas.
Entre las partes que pueden contener los micro-controladores se encuentran:

- Microprocesador: es la unidad aritmética lógica, registros asociados y controladores que realizan las operaciones del sistema y es responsable de ejecutar las instrucciones del programa.
- Memoria: donde se almacena el programa y los datos.
- Entradas/Salidas (IO): permite la comunicación entre el micro-controlador y el mundo exterior.
- Comunicación: permite que el micro-controlador se comunique con otros dispositivos a través de diferentes protocolos como USB, Ethernet, etc.
- Almacenamiento: permite almacenar datos en el dispositivo, como una EEPROM o una memoria flash.
- Sensores: permiten medir diferentes variables ambientales como la temperatura, la humedad, la presión, etc.

Los micro-controladores tienen muchos usos, incluyendo:

- Control de motores,
- Automatización de procesos industriales,
- Dispositivos de medida y monitoreo,
- Control de sistemas de iluminación y calefacción,
- Control de sistemas de seguridad,
- Control de electrodomésticos y dispositivos de consumo,

- Control de sistemas de comunicación,
- Control de sistemas de vehículos,
- Control de robots y sistemas automatizados.

A.1.1. Arduino

El Arduino es una plataforma de desarrollo de hardware y software libre basada en un micro-controlador. La programación de Arduino se basa en el lenguaje de programación C++ y utiliza un entorno de desarrollo integrado (IDE) específico llamado Arduino IDE. El Arduino IDE es una aplicación de escritorio que se utiliza para escribir, depurar y cargar código en el micro-controlador. El procedimiento básico de programación del Arduino es el siguiente:

a. Conectar el Arduino a la computadora mediante un cable USB.

b. Abrir el Arduino IDE y seleccionar el tipo de Arduino y la tarjeta que se va a utilizar en el menú Herramientas.

c. Escribir el código en el editor de código del Arduino IDE, utilizando el lenguaje de programación C++.

d. Verificar el código compilando el código utilizando el botón verde de compilación del Arduino IDE.

e. Cargar el código en el Arduino utilizando el botón azul de carga del Arduino IDE.

f. Observar el comportamiento del código en los pines de entrada y salida del Arduino, si es necesario realizar cambios en el código para obtener el resultado deseado.

g. Repetir los pasos 3-6 hasta que el código funcione correctamente.

El esquema básico de programación de Arduino utiliza dos funciones esenciales [1]: setup() y loop(). La función setup(): Es la primera función que se ejecuta cuando el Arduino se enciende o se re-inicia. En esta función se configuran los pines de entrada y salida, se establecen las velocidades de comunicación, se inicializan las variables, entre otras tareas de configuración.
La función loop(): Es la función que se ejecuta continuamente después de que se ha ejecutado la función setup(). En esta función se escriben las instrucciones que se deben ejecutar continuamente, como la lectura de sensores, el control de actuadores, la comunicación con otros dispositivos, entre otras tareas.
Las instrucciones y funciones del lenguaje C++ para Arduino pueden ser consultadas en el siguiente vínculo.

Apéndice B

Repositorio de código

B.1. Código actividad 1

```
/*
Instituto Tecnológico de Costa Rica
Laboratorio de Control Eléctrico.
Lab #1: Introducción a Arduino
Este programa prende el LED cuando se presiona el botón.
*/

// Declaraciones de entradas y salidas
const int BUTTON = 2;  // Boton conectado al pin 2
const int LED = 4;     // LED conectado al pin 4

// Configuración
void setup() {
    pinMode(LED, OUTPUT);    // configuramos el pin del LED como salida
    digitalWrite(LED, HIGH);  // Encendemos el LED
    pinMode(BUTTON, INPUT);  // configuramos el pin del botón como entrada
}
//  Programa principal
void loop() {
    bool buttonState = !digitalRead(BUTTON);  // leemos el estado del botón
    digitalWrite(LED, buttonState); // Actualizamos el estado del LED
}
```

B.2. Código actividad 2

```
/*
Instituto Tecnológico de Costa Rica
Laboratorio de Control Eléctrico.
Lab #1: Introducción a Arduino
Implementación de funciones lógicas AND, OR, XOR, NAND, NOR de 2 entradas
*/


// Declaracione de constantes
  const int PinEntrada[2]={2,3};
  const int PinSalidas[5]={4,5,6,7,8};
  const int PinAnalogico[2]={A0,A1};
  int ValorAnalojLeido[2]={0,0};
  float ValorVoltage[2]={0.0,0.0};
  boolean Mapa_entradas[2];
  boolean ResultadoLogico[5]={false, false, false, false, false};

// Configuracion de Pines de entrada y salida
void setup(){
  Serial.begin(115200);
```

```
  Serial.println("-----");
// Initializa los pines:
for (int i = 0; i < 2; i++) {
  pinMode(PinEntrada[i], INPUT);
}
for (int i = 0; i < 5; i++) {
  pinMode(PinSalidas[i], OUTPUT);
}
delay(2);
}

void loop()
{
    //Lectura de entradas
    for(int i=0; i < 2; i++){
        Mapa_entradas[i] = digitalRead(PinEntrada[i]);
        ValorAnalojLeido[i] = analogRead(PinAnalogico[i]);
        ValorVoltage[i]= (map(ValorAnalojLeido[i], 0, 1023, 0, 500)/100.0);
    }
    //Escritura de salidas
    for(int i=0; i < 5; i++){
        digitalWrite(PinSalidas[i],ResultadoLogico[i]);
    }
//EJECUCION DEL PROGRAMA
  ResultadoLogico[0]=AND(Mapa_entradas[1],Mapa_entradas[0]);
  ResultadoLogico[1]=OR(Mapa_entradas[1],Mapa_entradas[0]);
  ResultadoLogico[2]=XOR(Mapa_entradas[1],Mapa_entradas[0]);
  ResultadoLogico[3]=NAND(Mapa_entradas[1],Mapa_entradas[0]);
  ResultadoLogico[4]=NOR(Mapa_entradas[1],Mapa_entradas[0]);

//IMPRESION DE RESULTADOS ENTRADA Y SALIDAS

  Serial.print(Mapa_entradas[0]);
  Serial.print(",");
  Serial.print(ValorVoltage[0]);
  Serial.print(",");
  Serial.print(Mapa_entradas[1]);
  Serial.print(",");
  Serial.print(ValorVoltage[1]);
  Serial.print(",");
  Serial.print(ResultadoLogico[0]);
  Serial.print(",");
  Serial.print(ResultadoLogico[1]);
  Serial.print(",");
  Serial.print(ResultadoLogico[2]);
  Serial.print(",");
  Serial.print(ResultadoLogico[3]);
  Serial.print(",");
  Serial.println(ResultadoLogico[4]);
  delay(10);
}

//DEFINICION DE LAS FUNCIONES LÓGICAS
// Forma de programacion Booleana
bool AND (bool X, bool Y ){
  return (X & Y);
}

// Forma de programacion con estructuras de control
bool OR (boolean X, bool Y ){
  return (X | Y);
}

// Completar código
bool XOR (bool X, bool Y ){
  return (X ^ Y);
}

bool NAND (bool X, bool Y ){
  return !(X & Y);
```

```
}

bool NOR (bool X, bool Y ){
  return !(X | Y);
}
```

Bibliografía

[1] M. Margolis, B. Jepson, and N. Weldin, *Arduino Cookbook: Recipes to Begin, Expand, and Enhance Your Projects*. O'Reilly Media, 2020.