TEC | Tecnológico de Costa Rica

Escuela de Ingeniería Electromecánica

Ingeniería en Mantenimiento Industrial

Instructivo de Laboratorio

Laboratorio de Control Eléctrico

22 de agosto 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.

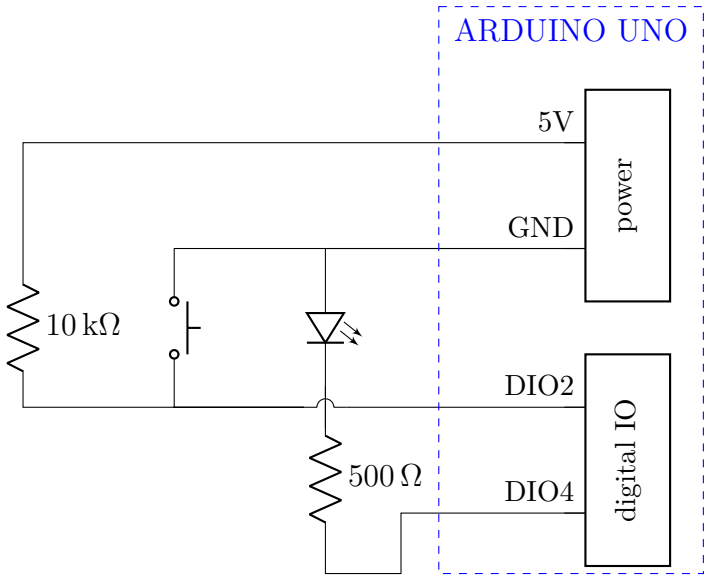

Figura 1.1: Conexión de circuito para Actividad 2

1.4.2. Actividad 3

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. Imprima los resultados en el puerto serial, ajuste la velocidad del puerto con valores de: { 9600, 14400, 19200, 28800, 38400, 57600}bps según la frecuencia seleccionada del generador de funciones.

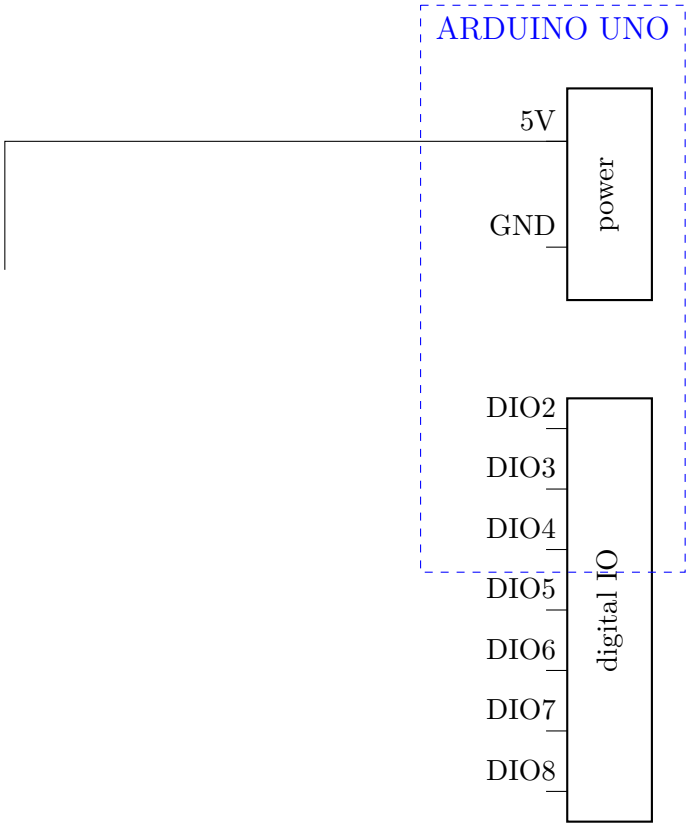

Figura 1.2: Conexión de circuito para Actividad 3

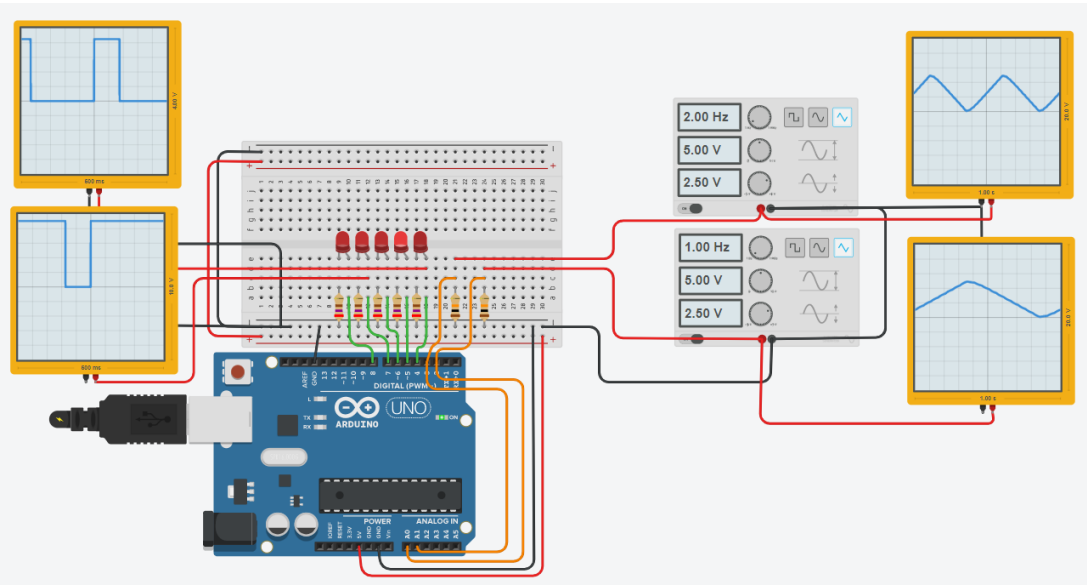

Figura 1.3: Esquema de conexión del Arduino, Generador de funciones y Osciloscopio

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?

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

```
/*
    ASCII table

    Prints out byte values in all possible formats:
    - as raw binary values
    - as ASCII-encoded decimal, hex, octal, and binary values

    For more on ASCII, see http://www.asciitable.com and http://en.wikipedia.org/wiki/ASCII

    The circuit: No external hardware needed.

    created 2006
    by Nicholas Zambetti <http://www.zambetti.com>
    modified 9 Apr 2012
    by Tom Igoe

    This example code is in the public domain.

    https://www.arduino.cc/en/Tutorial/BuiltInExamples/ASCIITable
*/

void setup() {
    //Initialize serial and wait for port to open:
    Serial.begin(9600);
    while (!Serial) {
    ;  // wait for serial port to connect. Needed for native USB port only
    }

    // prints title with ending line break
    Serial.println("ASCII Table ~ Character Map");
}

// first visible ASCIIcharacter '!' is number 33:
int thisByte = 33;
// you can also write ASCII characters in single quotes.
// for example, '!' is the same as 33, so you could also use this:
// int thisByte = '!';

void loop() {
    // prints value unaltered, i.e. the raw binary version of the byte.
    // The Serial Monitor interprets all bytes as ASCII, so 33, the first number,
    // will show up as '!'
    Serial.write(thisByte);

    Serial.print(", dec: ");
    // prints value as string as an ASCII-encoded decimal (base 10).
    // Decimal is the default format for Serial.print() and Serial.println(),
```

```
    // so no modifier is needed:
    Serial.print(thisByte);
    // But you can declare the modifier for decimal if you want to.
    // this also works if you uncomment it:

    // Serial.print(thisByte, DEC);


    Serial.print(", hex: ");
    // prints value as string in hexadecimal (base 16):
    Serial.print(thisByte, HEX);

    Serial.print(", oct: ");
    // prints value as string in octal (base 8);
    Serial.print(thisByte, OCT);

    Serial.print(", bin: ");
    // prints value as string in binary (base 2) also prints ending line break:
    Serial.println(thisByte, BIN);

    // if printed last visible character '~' or 126, stop:
    if (thisByte == 126) {  // you could also use if (thisByte == '~') {
    // This loop loops forever and does nothing
    while (true) {
        continue;
    }
    }
    // go on to the next character
    thisByte++;
}
```

B.2. Código actividad 2

```
    /*
    Instituto Tecnológico de Costa Rica
    Laboratorio de Control Eléctrico.

    Este programa prende el LED cuando se presiona el botón.
    */

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

    // CONFIGURACION DE ENTRADAS Y SALIDAS
    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
        if (buttonState == HIGH) {
            digitalWrite(LED, HIGH);  // apagamos el LED
        } else {
            digitalWrite(LED, LOW);   // encendemos el LED
        }
    }
```

B.3. Código actividad 3

```
/*
Instituto Tecnológico de Costa Rica
Laboratorio de Control Eléctrico.
Lab #1: Conectiva logicas y señal analógica
Fecha: 24/01/2023
```

```
Ing. Luis D. Murillo

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 < (sizeof(PinSalidas)/2); i++) {
  if (i<(sizeof(PinEntrada)/2)) {
    pinMode(PinEntrada[i], INPUT);
  }
  pinMode(PinSalidas[i], OUTPUT);
  }
delay(2);
}

void loop()
{
  //LECTURA DE LAS ENTRADAS Y SALIDAS
  for(int i=0; i<(sizeof(PinSalidas)/2);i++){
    if(i<(sizeof(PinEntrada)/2)){
      // Lectura de entradas digitales
      Mapa_entradas[i]=digitalRead(PinEntrada[i]);
      // Lectura de los valores analógicos
      ValorAnalojLeido[i] = analogRead(PinAnalogico[i]);
      // Funcion de mapeo :
    ValorVoltage[i]= (map(ValorAnalojLeido[i], 0, 1023, 0, 500)/100.0);
    Serial.println(ValorVoltage[i]);
    }
  // Escritura de salida digital
  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.