
Universidad Autónoma de Baja California
Facultad de Ciencias
Carrera: Física

Alumnos/Matrículas: Diego Osvaldo Ochoa
de la Cruz 346427

Federico Soto Badilla 347428

Materia: Instrumentación

Docente: Eloisa del Carmen García Canseco

Proyecto Final

Osciloscopio con Arduino

30 de Mayo del 2018



Objetivos

- Aprender a utilizar la tarjeta Arduino
- Aprender a graficar en tiempo real en Python
- Utilizar lo aprendido en clase para poder elaborar un osciloscopio

1. Introducción

En electrónica, a la hora de querer implementar circuitos electrónicos, es de suma importancia el conocer el potencial de la onda que se le suministra al circuito, ya que tener bien conocido ese dato, entre otros, garantiza un buen funcionamiento del circuito utilizado así como la integridad del mismo. Para conocerlo es necesario un instrumento que tenga la capacidad de medir el potencial y así poder tener idea de él.

Este instrumento es conocido como osciloscopio y su función es medir y desplegar en pantalla la forma de la onda así como la amplitud (Voltaje) que tiene la onda que se le suministra, entre otros datos sobre ella.

Un inconveniente de este instrumento es el precio que tiene, el cual es demasiado elevado para el público en general por lo que tener una alternativa más económica es realmente útil. Es aquí donde Arduino puede ser útil para poder obtener uno por un precio mucho menor; sin embargo la exactitud es menor a la que se obtiene con uno comprado ya hecho pero no deja de ser una opción viable cuando se necesita uno de esos instrumentos.

Este proyecto consiste en implementar, con Arduino, un osciloscopio de bajo costo para tener una opción adicional a un osciloscopio ya hecho.

Cabe decir que este proyecto es la adaptación de un proyecto ya existente encontrado en internet.

Lista de Materiales Electrónicos

- Arduino UNO
- Computadora
- 8 Resistencias de 1kOhm

Mecánico

- Protoboard
- Jumpers

Software

- Editor de Arduino
- Python

2. Funcionamiento

Tarjeta Arduino: Es una placa para programar basada en un micro controlador ATMEL, el cual tiene como función recibir instrucciones, guardarlas y ejecutarlas. Físicamente este dispositivo está compuesto por millones de transistores y otros componentes que realizan operaciones lógicas y permiten que el micro controlador funcione. Estas instrucciones que se le dan se escriben utilizando el editor que proporciona Arduino.

Este micro controlador posee entradas y salidas tanto analógicas como digitales. Arduino se comunica con la computadora a través del puerto serial.

El funcionamiento específico del micro controlador dependerá del proyecto que se esté realizando.

Resistencia: Es un componente pasivo de dos terminales que se utiliza para agregar resistencia eléctrica a un circuito. Son usados para reducir el flujo de corriente, ajustar niveles de señales, dividir voltajes, entre otras aplicaciones.

Python: Python es un lenguaje de programación interpretado con muchas librerías orientadas al computo científico entre ellas existen tres que fueron de gran ayuda en este proyecto:

- **matplotlib.pyplot** Que nos permite utilizar funciones para graficar un conjunto de datos.
- **serial** Que nos proporciona una forma de interactuar con el puerto serial de la computadora para comunicarnos con la tarjeta Arduino.
- **drawnow** Proporciona la función del mismo nombre que permite tomar una gráfica ya hecha y actualizar los datos en ella, esta basada en una librería con el mismo nombre encontrada en matlab.

Cabe mencionar que matlab, entre otros lenguajes, tiene librerías y funciones similares pero escogimos python porque además de la utilidad que tiene ya estábamos familiarizados con el lenguaje.

Qucs(Quite Universal Circuit Simulator): Es un programa hecho para simulación de circuitos que sigue siendo desarrollado, lo escogimos a inicio del semestre para utilizarlo en las prácticas ya que era la mejor opción que encontramos en Linux (nuestro sistema operativo al momento), pero existe en otras plataformas y tiene lo necesario para este proyecto.

2.1. Código de Arduino

```
/*
Ejemplo para leer el voltaje análogo encontrado en el
archivo de ejemplos de Arduino,
solo se cambio los bits por segundo de 9600 a 115200
debido a que se nota una mejor
```

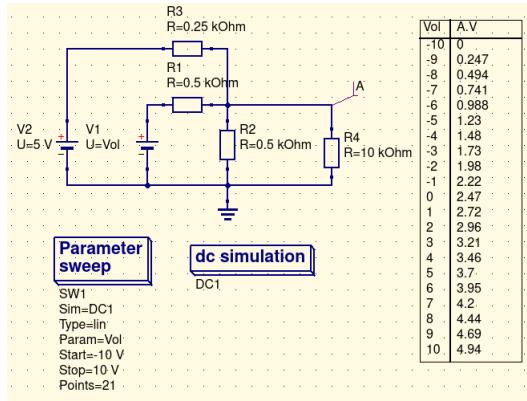


Figura 1: Simulación del circuito en Qucs

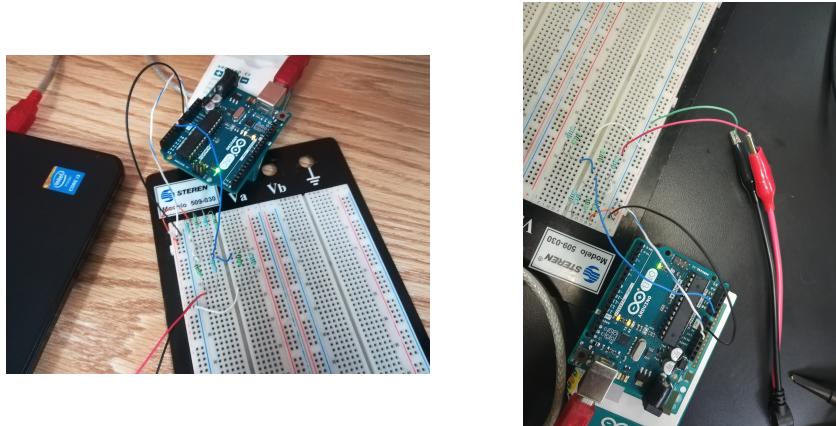


Figura 2: Circuito de resistencias conectado al Arduino

respuesta del osciloscopio con esto durante las pruebas
y se hizo la traducción de los
comentarios para la documentación
El código del ejemplo original esta en:
<http://www.arduino.cc/en/Tutorial/ReadAnalogVoltage>
*/

```
// El siguiente código corre cuando se presiona el botón
// de reinicio
void setup() {
    // Iniciar la comunicación con el puerto serial a
    // 115200 bits por segundo
    Serial.begin(115200);
```

```

}

// El siguiente código corre infinitamente
void loop() {
    // Leer el valor de entrada del puerto A0
    int sensorValue = analogRead(A0);
    // Convertir el valor de entrada (Que va de 0 - 1023) a
    // un voltaje (De 0 - 5v)
    float voltage = sensorValue * (5.0 / 1023.0);
    // Imprimir el voltaje leído en el puerto serial
    Serial.println(voltage);
    delay(10);
}

```

2.2. Código de Python

```

# Programa para leer el valor que imprime Arduino en el
# puerto serial en un arreglo y
# graficarlo en tiempo real, esto es una recompilación de
# algunos programas encontrados en
# múltiples fuentes de Internet sobre como utilizar las
# librerías drawnow, serial y
# matplotlib.pyplot así como en los ejemplos encontrados
# en la documentación de las
# mismas.

# Librería para leer de un puerto serial
import serial
# Librería para hacer el plot
import matplotlib.pyplot as plt
# Librería para graficar en tiempo real
from drawnow import *

# Función para actualizar la gráfica en tiempo real
def MakeFig():
    # Definir límites para Y
    plt.ylim(-10,10)
    # Definir límites para X
    plt.xlim(0,50)
    # Titulo de la gráfica
    plt.title('Osciloscopio')
    # Colocar la rejilla
    plt.grid()
    # Titulo para Y
    plt.ylabel('Volts')
    # Gráfica del arreglo

```

```

plt.plot(VOLT, 'ro-')

# Crear un objeto serial para los datos
ArduinoData = serial.Serial("/dev/ttyACM0", 115200)
VOLT = []
X = []
# Le dice a matplotlib que inicie en modo interactivo
plt.ion()

# Ciclo que continua para siempre
while True:
    # Leer una linea de texto del puerto serial
    ArduinoString = ArduinoData.readline()
    # Arduino arroja algunos datos que no pueden ser
    convertidos a float de forma
    # aparentemente aleatoria, por esto decidimos
    utilizar un try: except: que intente
    # correr el siguiente código y de ocurrir un error
    por los datos no hacer nada en ese
    # ciclo.
    try:
        # Convertir el texto a float y cambiarlo según el
        análisis de los datos
        Volt = 8*(float(ArduinoString) - 2.5)
        # Agregar datos al arreglo para el plot
        VOLT.append(Volt)
        # Llama a drawnow para actualizar el plot con la
        función MakeFig descrita
        # anteriormente
        drawnow(MakeFig)
        # Condicional para leer el largo del arreglo y
        preguntar si es mayor a 50
        # elementos
        if len(VOLT) > 50:
            # Eliminar el primer elemento
            VOLT.pop(0)
    except:
        # Esto solo salta el ciclo en caso de un error
        pass

```

3. Desarrollo del proyecto

1. Ahora procedemos a elaborar el circuito que permitirá que el Arduino lea voltajes negativos y amplíe el valor máximo que Arduino puede recibir; ya que estar limitados a un rango de 0 a 5 volts es un gran inconveniente, la simulación del circuito se muestra en la figura 1, asumiendo que Arduino

presenta una resistencia de $10k\Omega$ y con un voltaje de -10 a 10 volts.

2. Lo siguiente por hacer es el programa de Arduino para poder leer la señal entrante y mandar los datos a la computadora para poder procesarlos. Lo primero que se programa es el código del editor de Arduino, el cual se muestra en la sección 2.1.
3. Una vez programado eso lo siguiente es hacer el programa que graficará los datos que proporciona Arduino, para eso usamos el código de Python que se puede ver en la sección 2.2.
4. Una vez hecho lo anterior lo siguiente es conectar un generador de funciones al Arduino para empezar a medir. Se necesita tener cuidado de que señal se le suministrará ya que exceder el límite de voltaje podría causar daños al circuito (Recordar que el nuevo rango para el voltaje es de -10 a 10 volts).
5. Por último se debe compilar el programa de Arduino en el editor del mismo para mandarlo a la tarjeta y correr el programa en python para visualizar los datos que Arduino manda dando como resultado las imágenes de la figura 3

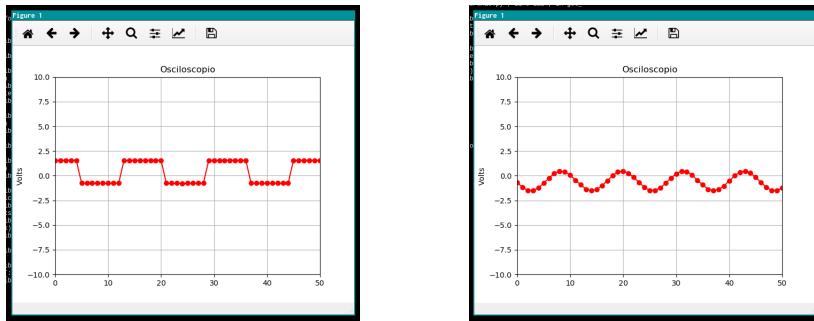


Figura 3: Resultados

4. Conclusión

Este proyecto si bien es económico en comparación a conseguir un osciloscopio “Profesional” cuenta con la gran desventaja de no ser tan exacto como el otro, aparte carece de todas las funciones adicionales que posee el antes mencionado. Esto lleva a que este proyecto sea un último recurso en caso de necesitarse un osciloscopio o bien hace que quede como un proyecto “recreativo”.

La más grande dificultad encontrada durante la realización de este proyecto fue el poder realizar gráficas en tiempo real de los datos que proporciona Arduino ya que si no se podía graficar así realmente el proyecto no funcionaba. Se buscó

en muchas páginas web opciones, sin embargo, muy pocas realmente proporcionaban una forma de realizar dicha acción.

El segundo mayor problema fue encontrar la manera de hacer que Arduino, de alguna manera, pudiera “leer” voltaje negativo. Se intentó usar off set del generador de funciones pero esa alternativa era un tanto imprecisa y complicaba un poco la programación, por lo que se buscó otra alternativa.

Como tercera dificultad se encontraba en valor máximo de voltaje que se debe proporcionar a Arduino.

Estos últimos dos problemas se pudieron solucionar con un solo circuito, el cual, “mapea” el voltaje que se le proporciona y lo manda a un rango de 0 - 5 volts. Una manera de mejorar este proyecto es incorporarle funciones adicionales en la programación para que, al desplegar en pantalla el voltaje, también pueda proporcionar otros valores, ya sea RMS, frecuencia, etc., además de que pierde precisión conforme la frecuencia de la onda de entrada aumenta debido a que Arduino no muestrea a intervalos regulares, de nuevo esto puede ser solucionado con unos cambios en el programa de Arduino, además la precisión podría mejorarse con un circuito que divida el voltaje de entrada en intervalos de 0 a 5 volts dirigidos a distintos puertos analógicos para tener mas puntos de muestreo.