Protocolo de comunicación.

Cáceres Pinzón Brian Sebastian, Rodriguez Gelves Sebastian Camilo.
{est.brian.caceres,est.sebastianc.rod2}@unimilitar.edu.co
Profesor David Martinez.

Resumen*—*En este documento se realiza la comunicación entre un microcontrolador y un programa que grafica la señal recibida, aplicando una comunicación seria y desarrollando un protocolo de comunicación básico.

Palabras Claves-Comunicación serial, Sensores, Gráficas, Señal.

I. INTRODUCCIÓN.

Una señal en el dominio del tiempo es una señal análoga, la cual puede tener valores infinitos para cada valor de tiempo, por lo cual es importante si se va a trabajar con señales analógicas, entender que para poder registrar esta señal dentro de un sistema embebido es necesario realizar su digitalización, y esto conlleva tomar muestras de la señal cada determinado tiempo, por lo cual ahora se estaría trabajando con muestras y valores discretos almacenados dentro de un microcontrolador.

Por otro lado se debe entender el concepto de periodo de muestreo, simplemente es el tiempo que hay entre muestra y muestra, y debe ser siempre el mismo para que toda la teoría de señales funcione adecuadamente, la frecuencia de muestreo de igual forma que en señales análogas representa el inverso del periodo, es decir cuántas muestras se estarían tomando por unidad de tiempo, sin embargo se debe tener en cuenta que si una señal tiene determinada frecuencia, para muestrear se debe usar por lógica una frecuencia mayor a la de la misma, debido a que no tendría lógica intentar muestrear una señal cada cierto tiempo si está ya cumplió su ciclo, al hacer esto se obtiene un fenómeno que se conoce como aliasing, simplemente la señal muestreada sería indistinguible de la original.

Para este caso en específico, el microcontrolador que recibe los datos sería el procesador del computador y desde python se define el periodo de muestreo, el cual claramente es el mismo periodo de muestreo con el que se envían los datos desde el microcontrolador, una vez recibidos los datos se imprimen en tiempo real por medio de una interfaz gráfica de usuario realizada en python.

II. Trabajo previo

Arduino

Arduino es una plataforma de creación de electrónica de código abierto, la cual está basada en hardware y software libre, flexible y fácil de utilizar para los creadores y desarrolladores. Esta plataforma permite crear diferentes tipos de microordenadores de una sola placa a los que la comunidad de creadores puede darles diferentes tipos de uso.

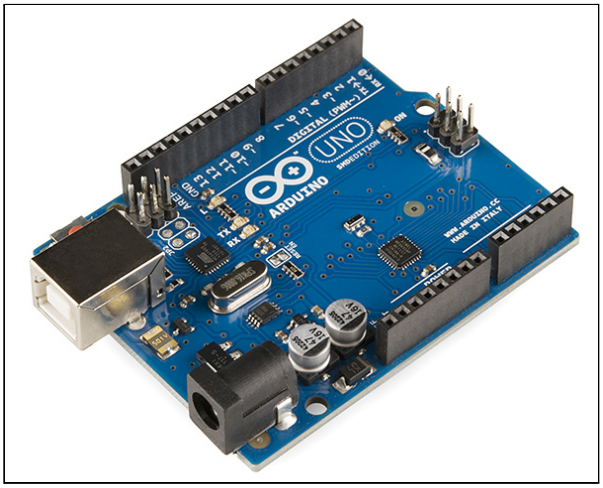

Imagen 1.Arduino uno.

Python

Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma [4].

Imagen 1.Logo Python

Comunicación serial

La comunicación seria es un protocolo de comunicación entre dispositivos que se incluye de manera estándar en prácticamente cualquier computadora. La mayoría de las computadoras incluyen puertos seriales. Actualmente hay puertos USB, aunque aún se encuentran algunas con puerto serial RS-232. La comunicación serial RS232 es un protocolo común utilizado por dispositivos y equipos usados en instrumentación. La comunicación serial puede ser utilizada para adquisición de datos, control, depuración de código, etc.

El concepto de comunicación serial, permite la transmisión-recepción bit a bit de un byte completo, este método de comunicación puede alcanzar mayores distancias. Por el contrario, la especificación IEEE 488 (comunicación en paralelo) determina que el largo del cable para el equipo no puede ser mayor a 20 metros, con no más de 2 metros entre cualesquier dos dispositivos; por el contrario, utilizando comunicación serial el largo del cable puede llegar a los 1200 metros.

Típicamente, la comunicación serial se utiliza para transmitir datos en formato ASCII. Para realizar la comunicación se utilizan 3 líneas de transmisión:

- Tierra (o referencia),
- Transmitir. (RX).
- Recibir.(TX).

Debido a que la transmisión es asíncrona, es posible enviar datos por una línea mientras se reciben datos por otra [3].

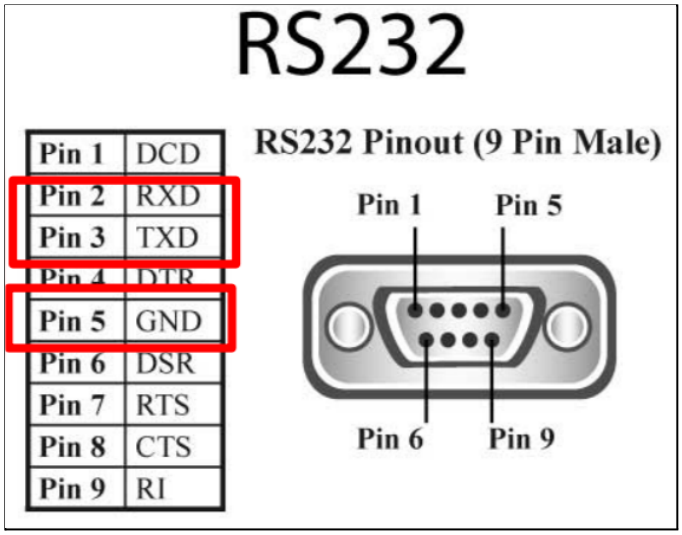

Imagen 1. Puerto Serial físico.

III. Desarrollo de la práctica.

El grupo de trabajo debe desarrollar un protocolo que permita la comunicación entre un sistema embebido y un computador por RS232. De acuerdo a lo ilustrado en la figura 1.

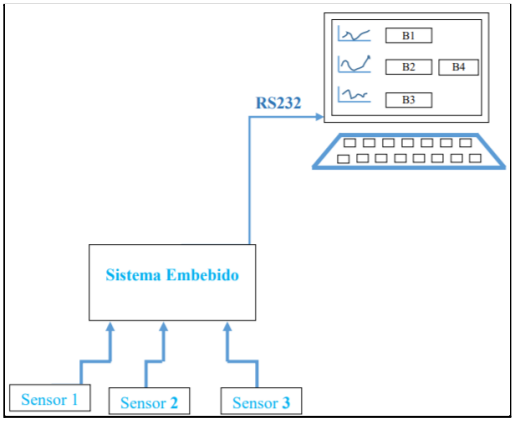

Figura 1.Sistema Propuesto.

Para desarrollar la parte del sistema embebido se realizó la activación de 3 módulos ADC en arduino, lo cuales leen el valor de voltaje entre 0 y 1023 para 0 y 5 voltios respectivamente, por lo cual se asigna un valor en esta escala para cada sensor recibido, por ello luego al realizar el envío del dato por medio del puerto serial, se realiza una escalación, para obtener nuevamente el valor correspondiente de voltaje, estos datos son enviados mediante un arreglo por el puerto serial con un periodo de envío definido aproximadamente en 40 milisegundos, es decir, cada 40 milisegundos se envía un arreglo con los voltajes de los 3 sensores.
Ver código implementado arduino anexo 1.

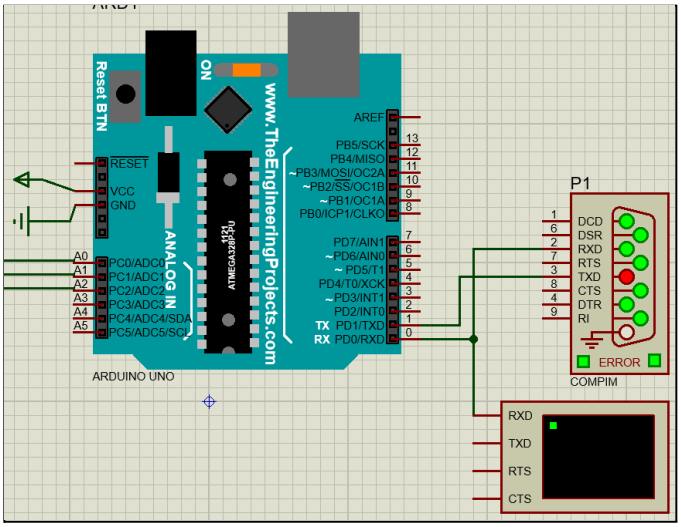

Figura 2. Arduino simulado con puerto serial virtual y entrada de 3 sensores en los puertos analogicos

Una vez desarrollado el código se compila y se extrae un archivo ".hex", el cual se pone dentro de la simulación en proteus, donde se monta el sistema que consta de las 3 entradas de los sensores al arduino como se puede ver en la figura 2, también se conecta un puerto serial en el cual debe ir tx con tx y rx con rx, debido a que es un módulo que se acopla al arduino, y la conexión común rx-tx se realiza por medio de un puerto virtual que se explicará más adelante.

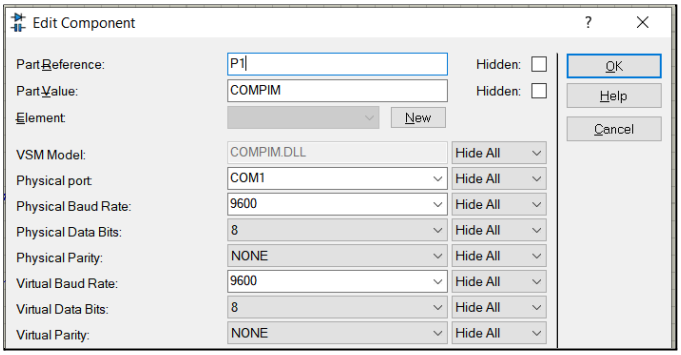

Figura 3. Arduino compim propiedades.

Como se puede observar en la figura 3 se configuró el puerto serial de arduino por medio de proteus el el COM1, la idea es conectar python por medio del COM2 y unirlos mediante un puerto serial.

Para configurar la conexión entre los puertos seriales se usó el programa "Virtual Serial Port Driver 9.0", aquí es importante recalcar que el puerto que se configura dentro de proteus es el com1, con una velocidad de 9600 bps, 8 bits de datos virtuales y ningún bit de paridad, la velocidad de transmisión utilizada en proteus debe ser la misma velocidad que en python, para no producir problemas en la conexión serial y evitar errores.

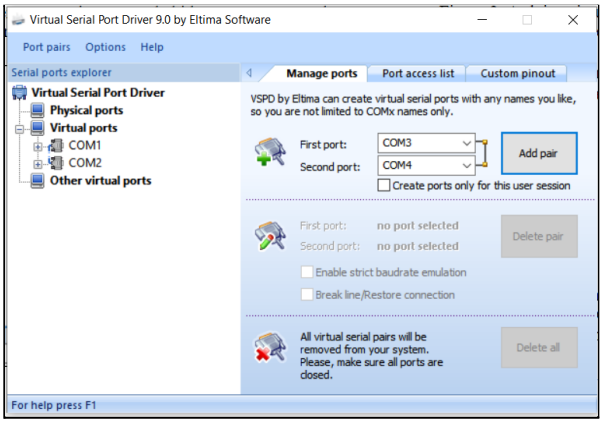

Figura 4. Conexión COM1 y 2 de forma virtual.

Desarrollando el código en python, lo primero es llamar las librerías a utilizar, maplotlib y tkinter son la principales, la primera es para realizar gráficas dinámicas con animaciones en tiempo real y la segunda es para poder trabajar con elementos de interfaz de usuario GUI como paneles, botones etc...

Laboratorio1.py - C:\Users\Cx001la\Desktop\aaa\Laboratorio1.py (...

File Edit Format Run Options Window Help

```
import serial
import time
import collections
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from tkinter import *
from matplotlib.lines import Line2D
import numpy as np

def getSerialData(self,Samples,numData,serialConnection,lines):
    for i in range(numData):
        value = float(serialConnection.readline().strip()) #Lee

        data[i].append(value)                               #Gua
        lines[i].set_data(range(Samples),data[i])           #Dib

        data2[i].append(value)
        lines2[i].set_data(range(Samples),data2[i])

serialPort = 'COM2'                                         #Pue
baudRate = 9600                                             #Bau
try:
    serialConnection = serial.Serial(serialPort,baudRate)   #Inst
except:
    print('No se conecto al puerto')
```

Figura 5. Librerías utilizadas e instanciación del puerto serial.

También se inicializa el puerto serial en el COM2, el cual está conectado como se puede ver con el COM1 en la figura 4, también se observa en la imagen el método de adquisición de datos el cual consiste en una variable llamada value donde se almacena el valor obtenido convertido en string y eliminando los espacios y saltos de línea con el comando strip(), posteriormente este dato de guarda en un vector llamado data que se agrega con el método append().

```
Samples = 100                                               #Numero
sampleTime = 100                                            #Tiempo
numData=3                                                   #Cantidad
lines = []
data = []
lines2=[]
data2=[]
for i in range(numData):
    data.append(collections.deque([0] * Samples, maxlen=Samples))
    lines.append(Line2D([],[],color='red'))

    data2.append(collections.deque([0] * Samples, maxlen=Samples))
    lines2.append(Line2D([],[],color='red'))
def grafica1():
    #Limites de los Ejes
    #serialConnection.write(str.encode('a'))
    time.sleep(1)
    xmin=0
    xmax=Samples
    ymin =0
    ymax =6
    fig = plt.figure(figsize=(13,16))                       #Defino t
    ax1 = plt.axes(xlim=(xmin,xmax),ylim=(ymin,ymax))       #Creo los
    plt.title('Sensor # 1')
    ax1.set_xlabel("Muestras")
    ax1.set_ylabel("voltaje")
    ax1.add_line(lines[0])                                  #Se elige
    anim = animation.FuncAnimation(fig,getSerialData, fargs =(Sample
    plt.show()
    ventana.mainloop()
    serialConnection.close()
```

Figura 6. Librerías utilizadas e instanciación del puerto serial.

Lo siguiente que se realizó fue inicializar las variables samples que equivale al número de muestras, sampleTime es el tiempo de muestreo y numData es la cantidad de sensores que estamos usando.

Como se puede apreciar se creó una variable llamada grafica1(), la cual como su nombre lo indica y se puede intuir es la gráfica del primer sensor, esta función es llamada por un botón, entonces lo primero que se hace es dar tiempo al programa de recibir datos con la función time.sleep(), luego se declaran los valores máximos y mínimos que van a tener los ejes de la gráfica, se instancia el objeto de la clase figura y se instancia también un objeto llamado ax1 que es el encargado de generar la gráfica con los ejes, se añaden labels con los nombres de las variables, muestras y voltaje y se usa el comando ax.add_line(Lines[0]) con el cual se genera la línea utilizando los valores obtenidos en el vector.

Para las demás gráficas se realiza el mismo procedimiento pero obviamente generando una gráfica distinta y utilizando una posición del vector Lines[i] correspondiente, es decir, sensor 2 se emplea el vector Lines[2], es decir el vector lines en la posición 2, de esta forma se generan todas las gráficas, lo único que cambia en la última gráfica es que se muestran todas las posiciones del vector Lines[i]. al mismo tiempo.

```
def grafica4():
    #Limites de los Ejes
    #serialConnection.write(str.encode('d'))
    #time.sleep(1)
    xmin=0
    xmax=Samples
    ymin = [0,0,0]
    ymax = [6,6,6]

    fig = plt.figure(figsize=(13,16))                                   #Crea una nueeva
    ax4 = fig.add_subplot(3,1,1,xlim=(xmin,xmax),ylim=(ymin[0],ymax[0]))
    ax4.title.set_text('Sensor # 1')
    ax4.set_ylabel("voltaje")
    ax4.add_line(lines2[0])

    ax5 = fig.add_subplot(3,1,2,xlim=(xmin,xmax),ylim=(ymin[1],ymax[1]))
    ax5.title.set_text('Sensor # 2')
    ax5.set_ylabel("voltaje")
    ax5.add_line(lines2[1])

    ax6 = fig.add_subplot(3,1,3,xlim=(xmin,xmax),ylim=(ymin[2],ymax[2]))
    ax6.title.set_text('Sensor # 3')
    ax6.set_xlabel("Muestras")
    ax6.set_ylabel("voltaje")
    ax6.add_line(lines2[2])

    anim = animation.FuncAnimation(fig,getSerialData, fargs =(Samples,numData,serialConnection
    plt.show()
    ventana.mainloop()
    serialConnection.close()
```

Figura 7. Código para la última gráfica con los 3 sensores a la vez.

También se muestra a continuación el código implementado para llamar las funciones por medio de botones.

```
ventana =Frame(height=500,width=435)
ventana.pack(padx=10, pady=10)
ventana.configure(background='gray')
Titulo= Label(text="Graficas en tiempo real de sensores",font=("Verdana",15)).place(x=35,y=50)
boton1= Button(ventana,font=("Verdana",8),command=grafica1, text=("Grafica sensor 1"), width=3
boton2= Button(ventana,font=("Verdana",8),command=grafica2, text=("Grafica sensor 2"), width=3
boton3= Button(ventana,font=("Verdana",8),command=grafica3, text=("Grafica sensor 3"), width=3
boton4= Button(ventana,font=("Verdana",8),command=grafica4, text=("Todas las Graficas"), width
ventana.mainloop()
serialConnection.close()
```

Figura 8. Código para generar botones.

RESULTADOS:

Una vez implementado todo lo anterior, se ejecuta el programa de proteus, el cual comienza la transmisión de datos por el puerto serial.

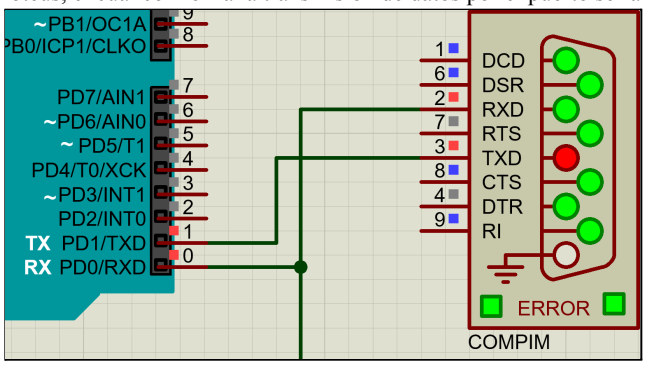

Figura 9. Simulación proteus enviando datos.

Como se puede observar se activa la simulación e inicia la transmisión de datos por el puerto serial, estos datos se reciben el python y por meros propósitos ilustrativos se imprimen los valores del los sensores en la consola de python, así se puede verificar que está funcionando correctamente la recepción de datos y la decodificación de los datos.

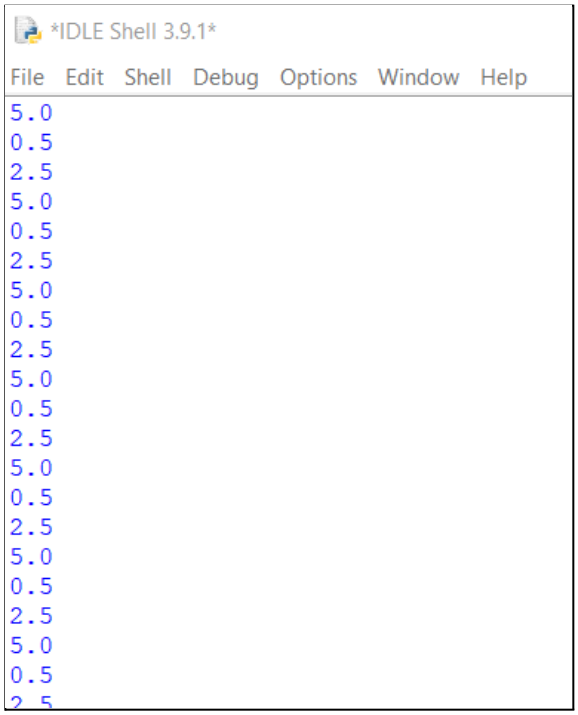

Figura 10. Recepción de datos en python.

Una vez ejecutado el python vemos la interfaz diseñada.

Figura 11. Interfaz gráfica diseñada.

Al apretar el botón “Gráfica sensor 1”

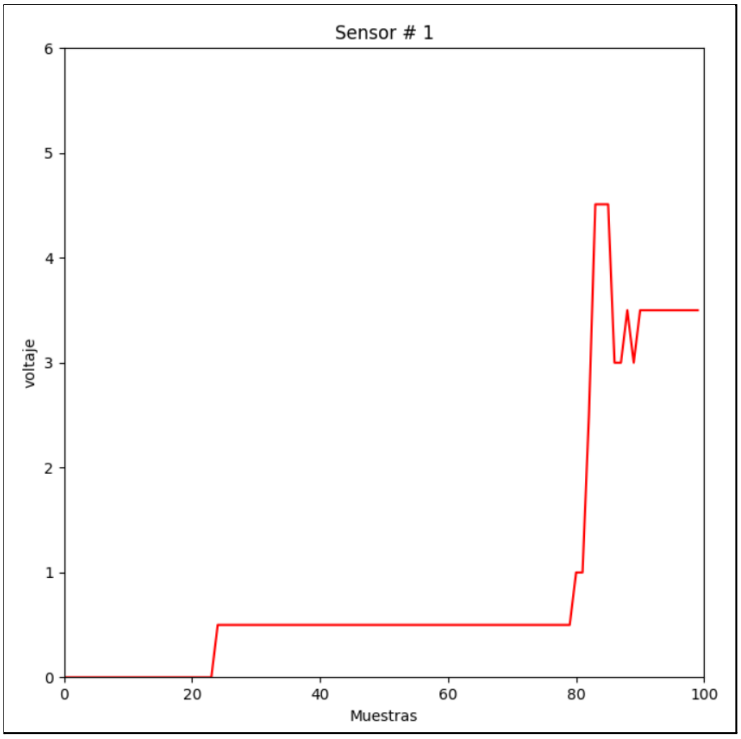

Figura 12. Gráfica del primer sensor.

Lo mismo sucede con las gráficas correspondientes al segundo y tercer sensor, a continuación se presenta la gráfica con los tres sensores de manera simultánea en tiempo real.

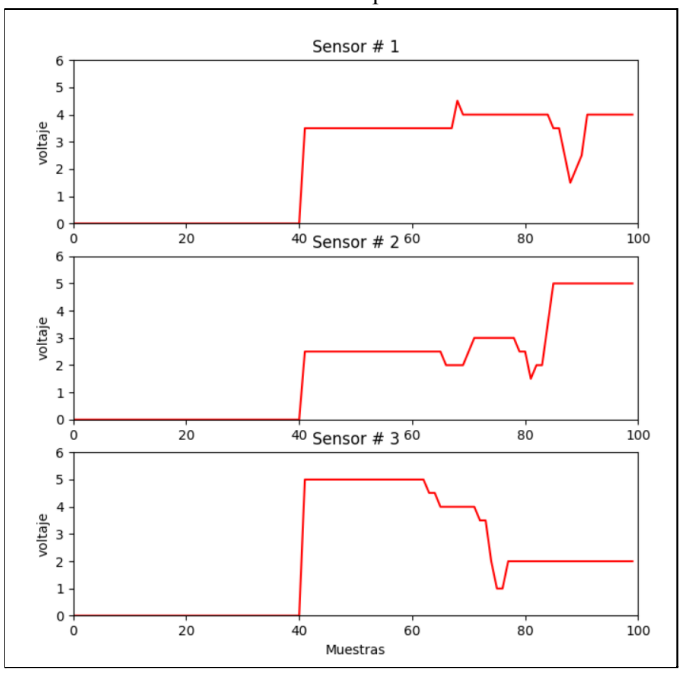

Figura 13. Gráfica de los tres sensores a la vez.

Ethernet

El módulo que se implementó para el manejo de ethernet por medio de arduino fue el W5100, el cual se encarga de implementar los protocolos TCP/IP, para poder usarlo se implementó la librería ethernet la cual lee y escribe los flujos de datos que pasan por el puerto ethernet.

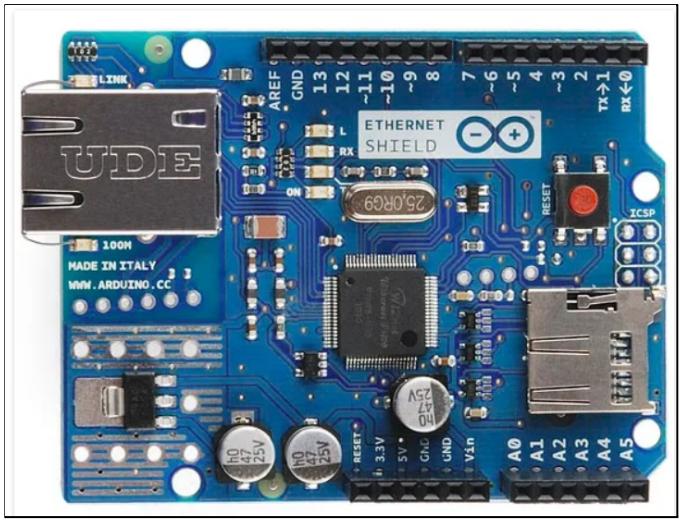

Figura 14. Shield arduino ethernet.

Código implementado en python.

Para python usa la librería socket, entonces las configuraciones a realizar deben ser la dirección ip, el puerto y se instancia la clase.

```
from socket import *
import time

direccion = ("169.254.33.58", 8000)
cliente_socket = socket(AF_INET, SOCK_DGRAM)
cliente_socket.settimeout(1)
```

Figura 15. Configuración implementada en python..

```
#*********************************
direccion = ("169.254.33.58", 8000)
cliente_socket = socket(AF_INET, SOCK_DGRAM)
cliente_socket.settimeout(1)
def getSerialData(self, Samples, numData, lines):
    data = "1" #Preguntamos por el sensor #1
    cliente_socket.sendto(data.encode(), direccion)
    try:
        rec_data, addr = cliente_socket.recvfrom(2048)
        sen1 = float(rec_data)
        #print("sensor #1 : {0}".format(sen1))
    except:
        pass
    #time.sleep(2)
    data = "2" #Preguntamos por el sensor #1
    cliente_socket.sendto(data.encode(), direccion)
    try:
        rec_data, addr = cliente_socket.recvfrom(2048)
        sen2 = float(rec_data)
        #print("sensor #2 : {0}".format(sen2))
    except:
        pass
    #time.sleep(2)
```

Figura 16. Código implementado para recibir los datos.

Código implementado en arduino.

Se implementó la librería "Ethernet.h" para utilizar las funciones de comunicación del módulo ethernet W5100, como se muestra a continuación para inicializar el puerto ethernet, la ip del arduino y la dirección mac y por ultimo el envio de datos por este modo de comunicación :

```
byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Direc
IPAddress ip(169, 254, 33, 58);//dirección IP del pc
unsigned int port = 8000;//puerto asignado
```

Figura 17. Configuración de comunicación ethernet en arduino.

```
if (datReq == "1")
{
    udp.print(voltageValue[0]);
}
```

Figura 18. Envío de datos desde arduino usando la librería.

ANÁLISIS DE RESULTADOS.

Se puede observar que con el método utilizado la señal que se gráfica está representada de manera continua como si se tratase de una señal analogica, para cualquiera de los tres sensores en las gráficas generadas, sin embargo es importante destacar que estas gráficas deberían ser representadas por medio de puntos debido a que al generarlas por medio de arduino y muestrearse en python, se está realizando la digitalización de una señal, no obstante python al el método de graficación hace una interpolación, conectando las muestras por medio de líneas, obteniendo la gráfica que observamos.

También se observó que cuando se enviaban datos por el puerto serial, la velocidad de respuesta en python se tardaba aproximadamente el 10 segundos, ello se debe a que el número de datos a graficar aumentaba, debido a esto se disminuye la frecuencia de envío de datos a 40 milisegundos por cada muestra del sensor, de esta forma se consiguió disminuir este retardo en las gráficas, y se obtuvo un resultado mucho mejor para la apreciación de los datos gráficamente en tiempo real.

CONCLUSIONES.

- Se desarrolló con éxito un protocolo de comunicación entre microcontrolador y python, por medio del puerto serial y a su vez se experimentaron fenómenos como el aliasing cuando la frecuencia de muestreo con la que se reciben datos era menor a la frecuencia con la que se enviaban datos y por esto mismo se deberían usar las mismas velocidades de transmisión de bits en arduino y

python.

- -Se determinó que cuando se realiza el envío de datos desde arduino, el periodo de envío de datos debe ser un valor moderado, no demasiado bajo como para saturar el programa en python y producir un efecto de retardo en la visualización de datos, y tampoco demasiado alto para producir una baja resolución en la gráfica obtenida.

REFERENCIAS.

[1] xataka, Yúbal Fernández, 2021. [online] https://www.xataka.com Available at:<https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno> [Recuperado 12 Febrero 2021].

[2] wikipedia, 2021. [online] at:<https://es.wikipedia.org/wiki/Python> [Recuperado 12 Febrero 2021].

[3] Desarrollo de sistemas embebidos, Comunicaciones Digitales: Protocolos seriales (uC) 2021. [online] at:<http://www.itq.edu.mx/carreras/IngElectronica/archivos_contenido/Apuntes%20de%20materias/ETD1022_Microcontroladores/4_SerialCom.pdf> [Recuperado 12 Febrero 2021].

[4] Buses De Campo y Protocolos en Redes Industriales, Salazar Cesar, Universidad de Manizales, Facultad de Ciencias e Ingeniería,2011.

[5] H. Bhojwani, G. K. Sain and G. P. Sharma, "Computerized Fuse Auto Changeover System with RS485 Bus Reporting & Multiple IOT Cloud Connectivity Avenues," 2018 3rd Technology Innovation Management and Engineering Science International Conference (TIMES-iCON), Bangkok, Thailand, 2018, pp. 1-5, doi: 10.1109/TIMES-iCON.2018.8621681.

[6] L. Huihui, "Design and Realization of SPI Driver Based on VxWorks," 2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chongqing, China, 2020, pp. 200-205, doi: 10.1109/ITNEC48623.2020.9084665.

ANEXOS

```
const int analogInPin1 = A0;  // Analog input pin 1
const int analogInPin2 = A1;  // Analog input pin 2
const int analogInPin3 = A2;  // Analog input pin 3
int sensorValue[2];         // value read from the pot
float voltageValue[2];         // voltage output
unsigned long lastTime = 0 , sampleTime = 10; // time value

void setup() {
  // initialize serial communications at 9600 bps:
  Serial.begin(9600);
}

void loop() {
  if(millis()-lastTime>sampleTime){
  lastTime = millis();
  // read the analog in value:
  sensorValue[0] = analogRead(analogInPin1);
  sensorValue[1] = analogRead(analogInPin2);
  sensorValue[2] = analogRead(analogInPin3);
  // map it to the range of the analog out:
  voltageValue[0] = scaling(sensorValue[0], 0, 1023, 0, 5);
  voltageValue[1] = scaling(sensorValue[1], 0, 1023, 0, 5);
  voltageValue[2] = scaling(sensorValue[2], 0, 1023, 0, 5);
  // Send data to Python:
  Serial.println(voltageValue[0]); delay(100);
  Serial.println(voltageValue[1]); delay(100);
  Serial.println(voltageValue[2]); delay(100);
  }
}

float scaling(float x, float in_min, float in_max, float out_min, float out_max)
{
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Anexo 1. Código implementado en arduino.

```
import serial
import time
import collections
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from tkinter import *
from matplotlib.lines import Line2D
import numpy as np

def getSerialData(self,Samples,numData,serialConnection,lines):
    for i in range(numData):
        value = float(serialConnection.readline().strip()) #Leer sensor
        print(value)
        data[i].append(value)                              #Guarda lectura en la ultima posicion
        lines[i].set_data(range(Samples),data[i])          #Dibujar nueva linea

        data2[i].append(value)
        lines2[i].set_data(range(Samples),data2[i])

serialPort = 'COM2'                                        #Puerto Serial Arduino
baudRate = 9600                                            #Baudios
try:
    serialConnection = serial.Serial(serialPort,baudRate)  #Instanciar objeto serial
except:
    print('No se conecto al puerto')

Samples = 100                                               #Numero de muestras
sampleTime = 100                                           #Tiempo de muestreo
numData=3                                                  #Cantidad de sensores
lines = []
data = []
lines2=[]
data2=[]
for i in range(numData):
    data.append(collections.deque([0] * Samples, maxlen=Samples))    #Creacion de lista
    lines.append(Line2D([],[],color='red'))

    data2.append(collections.deque([0] * Samples, maxlen=Samples))    #Creacion de lista
    lines2.append(Line2D([],[],color='red'))
def grafica1():
    #Limites de los Ejes
    #serialConnection.write(str.encode('a'))
    time.sleep(1)
    xmin=0
    xmax=Samples
    ymin =0
    ymax =6
    fig = plt.figure(figsize=(13,16))                      #Defino tamañano de la ventana y Crea una nueva figura
    ax1 = plt.axes(xlim=(xmin,xmax),ylim=(ymin,ymax))      #Creo los ejes de la grafica
    plt.title('Sensor # 1')
    ax1.set_xlabel("Muestras")
    ax1.set_ylabel("voltaje")
    ax1.add_line(lines[0])                                 #Se elige la variable a graficar
    anim = animation.FuncAnimation(fig,getSerialData, fargs =(Samples,numData,serialConnection,lines),interval=sampleTime)
    plt.show()
    ventana.mainloop()
    serialConnection.close()
```

Anexo 2. Código implementado en python para realizar gráfica 1 y para recibir los datos del puerto serial.(Se omiten las funciones gráfica 1 y 3, puesto que es el mismo código pero la instanciación de la clase es es decir los objetos son ax2 y ax3)

```
def grafica4():
    #Limites de los Ejes
    #serialConnection.write(str.encode('d'))
    #time.sleep(1)
    xmin=0
    xmax=Samples
    ymin = [0,0,0]
    ymax = [6,6,6]

    fig = plt.figure(figsize=(13,16))                                    #Crea una nueeva figura
    ax4 = fig.add_subplot(3,1,1,xlim=(xmin,xmax),ylim=(ymin[0],ymax[0]))
    ax4.title.set_text('Sensor # 1')
    ax4.set_ylabel("voltaje")
    ax4.add_line(lines2[0])

    ax5 = fig.add_subplot(3,1,2,xlim=(xmin,xmax),ylim=(ymin[1],ymax[1]))
    ax5.title.set_text('Sensor # 2')
    ax5.set_ylabel("voltaje")
    ax5.add_line(lines2[1])

    ax6 = fig.add_subplot(3,1,3,xlim=(xmin,xmax),ylim=(ymin[2],ymax[2]))
    ax6.title.set_text('Sensor # 3')
    ax6.set_xlabel("Muestras")
    ax6.set_ylabel("voltaje")
    ax6.add_line(lines2[2])

    anim = animation.FuncAnimation(fig,getSerialData, fargs =(Samples,numData,serialConnection,lines),interval=sampleTime)
    plt.show()
    ventana.mainloop()
    serialConnection.close()

ventana =Frame(height=500,width=435)
ventana.pack(padx=10, pady=10)
ventana.configure(background='gray')
Titulo= Label(text="Graficas en tiempo real de sensores",font=("Verdana",15)).place(x=35,y=50)
boton1= Button(ventana,font=("Verdana",8),command=grafica1, text=("Grafica sensor 1"), width=30,heigh=3).place(x=100,y=100)
boton2= Button(ventana,font=("Verdana",8),command=grafica2, text=("Grafica sensor 2"), width=30,heigh=3).place(x=100,y=200)
boton3= Button(ventana,font=("Verdana",8),command=grafica3, text=("Grafica sensor 3"), width=30,heigh=3).place(x=100,y=300)
boton4= Button(ventana,font=("Verdana",8),command=grafica4, text=("Todas las Graficas"), width=30,heigh=3).place(x=100,y=400)
ventana.mainloop()
```

Anexo 3. Código implementado en python para las gráficas de los tres sensores de manera simultánea, junto con el código de los botones.

```
#include <Ethernet.h>
#include <EthernetUdp.h>
#include <SPI.h>

const int analogInPin1 = A5;
const int analogInPin2 = A1;

int sensorValue[2];
float voltageValue[2];

int a;

byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED }; //Dirección MAC (constante)
IPAddress ip(169, 254, 33, 58);//dirección IP del pc
unsigned int port = 8000;//puerto asignado
char paqueteBuffer[UDP_TX_PACKET_MAX_SIZE]; //dimensión del char
String dataReq; // cadena para eure String
int paquete_size;// paketgröße
EthernetUDP udp;//crea objeto UDP
char packetBuffer[UDP_TX_PACKET_MAX_SIZE];//


void setup() {
    Ethernet.begin(mac, ip);
    udp.begin(port); //inicializar el udp
    delay(1500);
}
```

Anexo 4. Código implementado en Arduino por Visual Studio - Visual Macro, Inicialización de las variables, los sensores y el puerto Ethernet.

```
void loop() {

    sensorValue[0] = analogRead(analogInPin1);
    sensorValue[1] = analogRead(analogInPin2);
    sensorValue[2] = analogRead(analogInPin2);

    voltageValue[0] = scaling(sensorValue[0], 0, 1023, 0, 5);
    voltageValue[1] = scaling(sensorValue[1], 0, 1023, 0, 5);
    voltageValue[2] = scaling(sensorValue[2], 0, 1023, 0, 5);
```

Anexo 5. Código implementado en python por Visual Studio - Visual Macro, Conversión de la entrada de 0-1024 a 0-5.

```
if (udp.parsePacket() > 0)
{
    udp.read(packetBuffer, UDP_TX_PACKET_MAX_SIZE);
    String datReq(packetBuffer);
    Serial.println(packetBuffer);
    udp.beginPacket(udp.remoteIP(), udp.remotePort());
    if (datReq == "1")
    {
        udp.print(voltageValue[0]);
    }
    if (datReq == "2")
    {
        udp.print(voltageValue[1]);
    }
    if (datReq == "3")
    {
        udp.print(voltageValue[1]);
    }
    udp.endPacket();
}
// put your main code here, to run repeatedly:
```

Anexo 6. Código implementado en python por Visual Studio - Visual Macro, Envío del dato del sensor, según el dato enviado desde python.

```
float scaling(float x, float in_min, float in_max, float out_min, float out_max) {
    return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```

Anexo 6. Código implementado en python por Visual Studio - Visual Macro, Función para la conversión del anexo 5.