UNIVERSIDAD DE ORIENTE
NÚCLEO DE ANZOÁTEGUI
ESCUELA DE INGENIERÍA Y CIENCIAS APLICADAS
DEPARTAMENTO DE COMPUTACIÓN Y SISTEMAS
PROYECTOS DIGITALES AVANZADOS

PRÁCTICA N°5:
ADC Y SENSOR ANALÓGICO

Profesor:

Pedro Rene

Cabrera

Bachiller:

Claudia Rodríguez

C.I: 27.943.668

Barcelona, noviembre de 2024

1. OBJETIVOS DE LA PRÁCTICA

1.1. Comprender el uso de los pines ADC del Raspberry Pi Pico y establecer el factor de conversión necesario para obtener el valor real.

1.2. Recopilar la información utilizando el sensor analógico LM35, utilizado para la detección de temperatura de manera análoga y posteriormente convertirlo en un valor digital y poder establecer su valor correspondiente (se explicará más adelante el como se hizo el factor de conversión en el planteamiento del problema)

1.3. Almacenar la información en una memoria micro SD de 2GB usando el modulo micro SD y modo de comunicación SPI. Se utilizo un archivo TXT donde se detalle fecha, hora, número de muestra y muestra tomada.

1.4. Recuperar dicha muestra almacenada en el archivo TXT y poderla mostrar en la pantalla OLED usando el protocolo I2C, se mostrará número de muestra, temperatura, fecha y hora.

1.5. Procesar dicha información de manera tal de obtener un promedio de las muestras tomadas, en este caso, de las 10 muestras tomadas, que es una variable que se puede modificar ya sea para tomar más o menos muestras.

1.6. Graficar las muestras tomada en un histograma usando la pantalla OLED.

1.7. Avisar al usuario que ya las muestras están tomadas y preguntarle si desea volver a tomar muestras con el uso de botones y LEDS, ya sea para tomar nuevas muestras o finalizar el programa.

2. DESARROLLO

2.1. PLANTEAMIENTO DEL PROBLEMA

En la práctica anterior, se uso un sensor digital ultrasonido HCSR-04, para tomar las muestras de distancias, por lo tanto no se necesitaba realizar alguna conversión análoga digital, en la presente práctica se estará usando un sensor análogo denominado LM35, el cual, es utilizado para realizar muestras de temperatura. Debido a que este sensor es análogo, se tiene que conectar a uno de los pines ADC que posee el Raspberry Pi Pico, se conecto al GPIO26, que es el ADC0, ademas de ello, se requiere el uso de una formula de conversión, para ello, tenemos que verificar el voltaje de referencia del ADC en el Pico, el cual es de 3.3V y ademas que el valor máximo que puede leer el ADC es de 16 bits, por lo tanto 2^{16} = 65535 valores, posteriormente se tiene que leer el valor análogo del ADC para luego multiplicarlo por un factor de conversión, por último, se tiene que acudir al datasheet para verificar su escala de conversión, la cual es 10-mV/°C. Luego de esto, se procede a

poner en práctica lo ya estudiado en las prácticas anteriores, que es almacenar las muestras tomadas, con fecha y hora, en el modulo micro SD usando el protocolo de comunicación SPI y mostrar en la pantalla del computador o en la pantalla OLED igual usando el protocolo I2C.

	Método	**Sintaxis**	Operación
Comandos archivos	write()	**with open**("/sd/nombre_archivo.txt", "**w**") as file: file.write("")	Escribe una cadena
	writelines()	**with open**("/sd/nombre_archivo.txt", "**w**") as file: for line in ["Hello World \n", "You are welcome to Fcc\n"]: file.write(line)	Escribe varias cadenas
	write()	**with open**("/sd/**nombre_archivo**.txt", **"a**") as file: file.write(f'{contador}: {distance}\n')	Añade en el **txt**
	read()	**with open(**"/sd/**nombre_archivo**.txt", "**r**") as file: print(file.read())	Lee un archivo **txt**
	readline()	**with open(**"/sd/**nombre_archivo**.txt", "**r**") as file: print(file.readline())	Lee una línea del archivo **txt**
Pantalla OLED	fill(0) / fill(1)	oled.fill(0) // oled.fill(1)	Rellena la pantalla
	text(str,,0,0)	oled.text("texto", posición columna, posición fila)	Colocar texto en pantalla
	hline(10,30, 20,20)	oled.hline(posición columna, posición fila, tamaño, color de la linea)	Línea horizontal
	vline(10,30, 20,10)	oled.vline(posición columna, posición fila, tamaño, color de la linea)	Línea vertical
	show()	oled.show()	Muestra en pantalla
Cadenas	str(text)	str(claudia)	Convierte un tipo a cadena
	rstrip('caracter')	rstrip(':')	Elimina un caracter de derecha a izquierda
	split('caracter')	numero_muestra, temperatura = contador_temp.split(': ')	Utilizado para dividir una cadena en listas separadas por un caracter

Tabla de comandos utilizados en el código.

2.2. Solución

2.2.1. Se invocan aquellas librerías necesarias para el funcionamiento del programa, entre ellas, *machine, sdcard, ssd1306, utime, uos, time.*

2.2.2. Se define una función denominada **obtener_temperatura** donde se realizará la conversión anteriormente explicada en el planteamiento, esta conversión se realiza es en grados Celsius y se redondea con dos decimales.

2.2.3. Se establece una función denominada **obtener_fecha_hora_actual** con el objetivo que retorne dos variables, la fecha y la hora, para que las anote al momento de realizar la muestra y almacene en la tarjeta micro SD.

2.2.4. Se hace uso nuevamente de la función **pantalla** de prácticas anteriores, donde se establecen los parámetros de comunicación (como los pines SCL y SDA), dimensiones de la pantalla, etc y retorna un objeto de tipo oled.

2.2.5. Se hace uso nuevamente de la función **microsd,** igualmente, de prácticas anteriores, donde se inicializa la comunicación con el modulo de la tarjeta SD con el protocolo SPI, definiendo respectivamente sus pines.

2.2.6. De la práctica anterior, se toma igual la función **graficar**, la cual recibe dos parámetros, *oled y temperaturas***,** las cuales son utilizadas para poder realizar el gráfico tipo histograma. Cada barra tiene un ancho de 10 pixeles y un alto equivalente a la muestra que se desea reflejar.

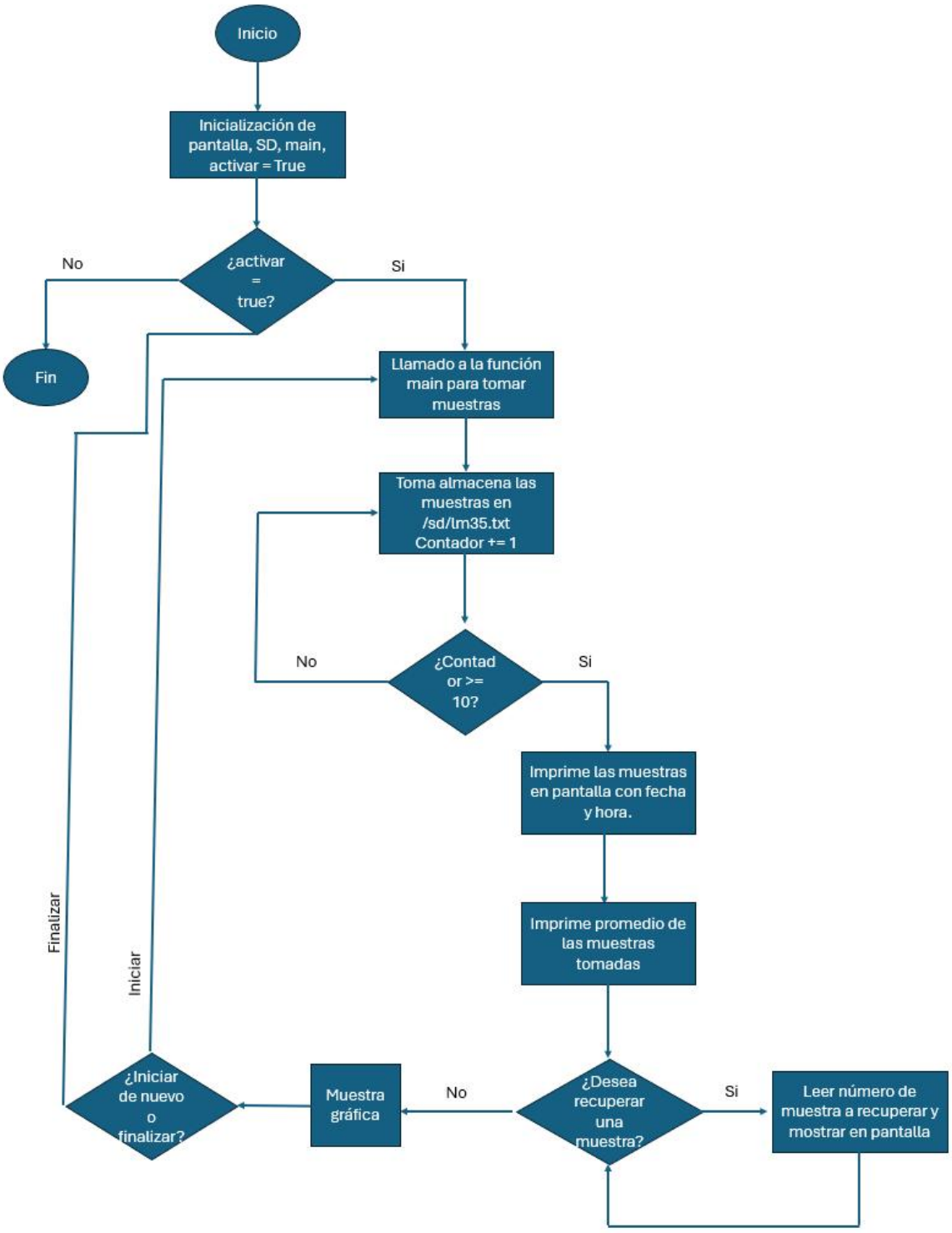
2.2.7. Aparece una nueva función denominada **avisar**, esta función, por iniciativa de la programadora, fue creada con el objetivo de avisarle al usuario que ya se tomaron las muestras y si lo desea puede tomar otras, teniendo en cuenta que se borrarán las anteriormente programadas o simplemente puede finalizar la ejecución del programa y quedarse con las últimas muestras tomadas y almacenadas en la memoria SD. Para avisar que ya las muestras fueron tomadas, se utilizo un LED de manera que con la función toggle parpadee de manera intermitente cada dos segundos.

2.2.8. Se define una función llamada **recuperar** con el objetivo de preguntar al usuario si desea recuperar una de las muestras tomadas, si el usuario desea recuperar otra muestra, puede hacerlo. Esta función esta disponible las n veces que el usuario desee visualizar nuevamente una de las muestras tomdas. Una vez que recupere las muestras que desee, pasaría a graficar las muestras tomadas. Esto se puede hacer ya sea por el teclado de la PC o del Keypad.

2.2.9. Se define la función **main**, que recibe como parámetro oled. Esta es la ejecución principal del programa donde se hará el llamado a las funciones anteriormente mencionadas, si así se requiere. De igual manera, las muestras se visualizan en la pantalla del computador y en la pantalla oled.

2.2.10. Se usarán igual ciertas funciones del teclado, como es su **inicialización, scan_teclado**, y **teclitas** para poder igual insertar algun dato que sea requerido por medio del Keypad.

2.3. Diagrama de flujo



2.4. Diagrama UML

Temperatura
- fila_pines: List[Pin] - columna_pines: List[Pin] - teclas: List[List[str]]
+ __init__(): + teclitas(): + init_teclado(): + scan_teclado(): + obtener_temperatura(): + obtener_fecha_hora_actual(): + pantalla(): + microsd(): + graficar(): + avisar(): + recuperar(): + main():

3. ANEXOS

3.1. Código

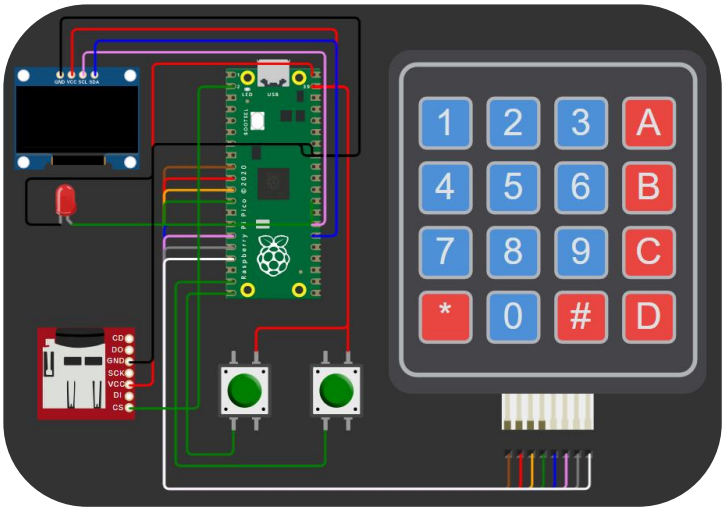
```
from time import sleep
import time
import machine
import sdcard
import uos
from machine import Pin, I2C
from ssd1306 import SSD1306_I2C
import utime

TECLA_ARRIBA = const(0)
TECLA_ABAJO = const(1)

class Temperatura:
    def __init__(self):
        self.fila_pines, self.columna_pines = self.teclitas()
        self.teclas =[['1', '2', '3', 'A'], ['4', '5', '6', 'B'], ['7', '8',
'9', 'C'], ['*', '0', '#', 'D']]

    def teclitas(self):
        filas = [6, 7, 8, 9]                    # gpios pines
        columnas = [10, 11, 12, 13]                    #
gpios pines

        fila_pines = [Pin(nombre_pin, mode=Pin.OUT)
for nombre_pin in filas]
        columna_pines = [Pin(nombre_pin, mode=Pin.IN,
pull=Pin.PULL_DOWN) for nombre_pin in columnas]
        return fila_pines, columna_pines

    def init_teclado(self):
        for fila in range(0, 4):
            self.fila_pines[fila].low()

    def scan_teclado(self, fila, columna):
        self.fila_pines[fila].high()
        tecla = None

        if      self.columna_pines[columna].value()      ==
TECLA_ABAJO:
            tecla = TECLA_ABAJO
        if      self.columna_pines[columna].value()      ==
TECLA_ARRIBA:
            tecla = TECLA_ARRIBA

        self.fila_pines[fila].low()  # precaucion, bajar dsp
de leerlos
        return tecla

    # datos tomados por el ADC0 = GPIO26
    # utilice LM35 retorna temp en celsius
    def obtener_temperatura(self):
        analog_value = machine.ADC(26)
        conversion_factor = 3.3/ 65535
        temp_voltage_raw = analog_value.read_u16()
        convert_voltage                                    =
temp_voltage_raw*conversion_factor
        tempC = convert_voltage/(10.0 / 1000)
        tempC = round(tempC, 2)
        return tempC

    # func para obtener la fecha y hora
    # de cuando se toma la muestra
    # retorna fecha y hora
    def obtener_fecha_hora_actual(self):
```

```
        current_time = time.time()
        local_time = time.localtime(current_time)

        year = local_time[0]
        month = local_time[1]
        day = local_time[2]
        hour = local_time[3]
        minute = local_time[4]
        second = local_time[5]

        fecha = "{:02}/{:02}/{}".format(day, month, year)
        hora = "{:02}:{:02}:{:02}".format(hour, minute, second)

        return fecha, hora

    # funcion para inicializar la comunicacion entre
    # la pantalla oled y el pico, pines gpio21 y gpio20
    def pantalla(self):

        WIDTH = 128                                     # oled dimensiones
        HEIGHT = 64
        i2c = I2C(0, scl=Pin(21), sda=Pin(20), freq=200000)     # pines gpio 21 y 20
        #print("I2C Address : "+hex(i2c.scan()[0]).upper()) # dir oled
        #print("I2C Configuration: "+str(i2c))
        oled = SSD1306_I2C(WIDTH, HEIGHT, i2c)
        return oled

    # funcion para inicializar la comunicacion spi
    # entre el modulo micro sd y pico
    def microsd(self):
            # <-inicializacion microsd
        cs = machine.Pin(1, machine.Pin.OUT)
        spi = machine.SPI(0, baudrate=1000000, polarity=0, phase=0, bits=8, firstbit=machine.SPI.MSB, sck=machine.Pin(2), mosi=machine.Pin(3), miso=machine.Pin(4))
        sd = sdcard.SDCard(spi, cs)
        vfs = uos.VfsFat(sd)
        uos.mount(vfs, "/sd")
        return sd                                       # retorna sd

    # funcion para grafico de barras histograma
    # parametros, oled y temperaturas
    # leidas desde la sd
    def graficar(self, oled, temperaturas):
        oled.fill(0)
        oled.text("Mostrando",0,0)
        oled.text("grafica...",0,16)
        oled.show()
        sleep(3)
        oled.fill(0)
        # eje 'X' y 'Y'
        # plano cartesiano, 1er cuadrante
        oled.vline(20,0,55,1) #vertical
        oled.hline(20,55,128,1) #horizontal
        for i in range(len(temperaturas)):
            temperaturas[i] = float(temperaturas[i]) # conversion de str a float
            oled.hline(abs(20 + 10 * i), 55 - (int(temperaturas[i])) , 10, 1) # columna, fila, distancia
        columna = 20
        for i in range(len(temperaturas)):
            oled.vline(columna, 55 - (int(temperaturas[i])), (int(temperaturas[i])) , 1)
            columna = columna + 10
            oled.vline(columna, 55 - (int(temperaturas[i])), (int(temperaturas[i])) , 1)
        oled.show()
        return

    # incorporada por la programaoora
    # para notificar al usuario que ya
    # estan listas las muestras
    def avisar(self, oled):
        boton_tomar_mas = machine.Pin(14, machine.Pin.IN, machine.Pin.PULL_DOWN)  # cable negro
        boton_finalizar = machine.Pin(15, machine.Pin.IN, machine.Pin.PULL_DOWN)  # cable verde
        led_onboard = machine.Pin(22, machine.Pin.OUT)
        sleep(5)
        oled.fill(0)
        oled.text("Desea tomar", 0, 0)
        oled.text("nuevas muestras?", 0, 16)
        sleep(3)
        oled.show()

        while True:
            led_onboard.toggle()
            utime.sleep(2)

            if boton_tomar_mas.value() == 1:    # cable negro
                oled.fill(0)
                oled.text("Tomando nuevas", 0, 0)
                oled.text("muestras...", 0, 16)
                oled.show()
                sleep(2)
                return True

            if boton_finalizar.value() == 1:  # cable verde
                oled.fill(0)
                oled.text("Finalizando...", 0, 0)
                led_onboard.value(0)
                oled.show()
                sleep(2)
                return False

    def recuperar(self, oled, numeros_muestra, temperaturas):
        self.init_teclado()
        backup = False
```

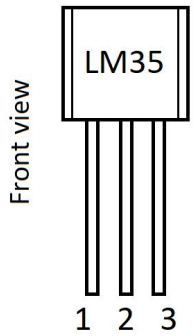
```
        muestra_recuperar = ""
        ultima_tecla_presionada = ""
        print("¿Desea recuperar una de las muestras tomadas? (si/no)")
        while not backup:
            oled.fill(0)
            oled.text("Desea recuperar",0,0)
            oled.text("alguna muestra?",0,16)
            oled.text("A: SI / B: NO",0,32)
            oled.show()
            for fila in range(4):
                for columna in range(4):
                    tecla = self.scan_teclado(fila, columna)
                    if tecla == TECLA_ABAJO:
                        tecla_presionada = self.teclas[fila][columna]
                        if tecla_presionada == 'A':
                            print("Digite número de muestra que desea recuperar:")
                            oled.fill(0)
                            oled.text("Ingrese numero",0,0)
                            oled.text("de muestra", 0,16)
                            oled.show()
                            sleep(2)
                            capturada = False
                            while not capturada:
                                for fila in range(4):
                                    for columna in range(4):
                                        tecla = self.scan_teclado(fila, columna)
                                        if tecla == TECLA_ABAJO:
                                            sleep(0.5)

                                            ultima_tecla_presionada = self.teclas[fila][columna]
                                            if ultima_tecla_presionada in '0123456789':

                                                muestra_recuperar += ultima_tecla_presionada
                                                print("Numero:", muestra_recuperar)
                                                oled.text("Muestra: " + str(muestra_recuperar),0,32)
                                                oled.show()

                                            if ultima_tecla_presionada == '#':
                                                capturada = True
                                                print("Muestra seleccionada: ", muestra_recuperar)
                                                oled.text("Muestra seleccionada: ",0,0)

                                                oled.text(str(muestra_recuperar),0,16)
                                                sleep(1)
                                                break

                            muestra_recuperar = muestra_recuperar.strip() + ':'
                            if muestra_recuperar in numeros_muestra:
                                oled.fill(0)
                                muestra_recuperar = int(muestra_recuperar.rstrip(':'))
                                oled.text(f'Muestra: {muestra_recuperar}', 0, 0)
                                oled.text(f'Temp: {temperaturas[muestra_recuperar-1]} C', 0, 16)
                                oled.show()
                                sleep(5)
                                # se implementa recursividad para que si quiere recuperar
                                # otra muestra pueda hacerlo
                                self.recuperar(oled, numeros_muestra, temperaturas)
                            else:
                                muestra_recuperar = ""
                                print("Nmro de muestra no valido. Intente de nuevo.")

                        elif tecla_presionada == "B":
                            print("No se recuperara ninguna muestra.")
                            backup = True  # Finalizar el bucle
                            break

                        else:
                            print("Respuesta no valida. Por favor responda 'A:SI' o 'B:NO'.")
                            oled.fill(0)
                            oled.text("Respuesta no",0,0)
                            oled.text("valida. Ingrese",0,16)
                            oled.text("A:SI / B:NO",0,32)
                            oled.show()
                        break
    def main(self, oled):
        led_onboard = machine.Pin(22, machine.Pin.OUT)
        led_onboard.value(0)
        oled = self.pantalla()

        muestreo = []
        prom_temp = 0
        contador = 1
        cantidad_muestras = 10

        with open("/sd/lm35.txt", "w") as file:
            file.write("")

        with open("/sd/lm35.txt", "r") as file:
            print("Leyendo archivo... Vacio")
            print(file.read())

        while True:
            if contador <= cantidad_muestras:
                tempC = self.obtener_temperatura()
```

```
            fecha, hora = self.obtener_fecha_hora_actual()

            print("Temperatura: ",tempC, "C"," ", fecha, hora)

            muestreo.append(float(round(tempC,2)))
            prom_temp = prom_temp + tempC

            oled.fill(0)
            oled.text(str(contador) + " Temperatura", 0, 0)
            oled.text(str(tempC), 0, 16)
            oled.text(str(fecha),0,32)
            oled.text(str(hora),0,48)
            oled.show()

            time.sleep(1)

            with open("/sd/lm35.txt", "a") as file:
                file.write(f'{contador}: {tempC} {fecha} {hora}\n')

            time.sleep(1)
            contador += 1

        else:
            with open("/sd/lm35.txt", "r") as file:
                print("\n Imprimiendo archivo")
                oled.fill(0)
                oled.text("Imprimiendo", 0, 0)
                oled.text("archivo", 0, 16)
                data = file.read()
                cadena = data.split("\n")
                numeros_muestra = []
                temperaturas = []

                for i in range(len(cadena) - 1):
                    if cadena[i] != '':
                        partes = cadena[i].split(' ')
                        if len(partes) >= 3:
                            contador_temp = partes[0] + ": " + partes[1]
                            fecha_hora = ' '.join(partes[2:])

                            fecha, hora = fecha_hora.split(' ')

                            numero_muestra, temperatura = contador_temp.split(': ')
                            numeros_muestra.append(numero_muestra)

                            temperaturas.append(temperatura)

                            oled.fill(0)
                            oled.text(f'Muestra: {numero_muestra}', 0, 0)
                            oled.text(f'Temp: {temperatura} C', 0, 16)
                            oled.text(fecha, 0, 32)
                            oled.text(hora, 0, 48)
                            oled.show()
                            sleep(1)

                oled.fill(0)
                prom_temp = round(((prom_temp) / (contador-1)), 2)
                oled.text("Impresion", 0, 0)
                oled.text("terminada", 0, 16)
                oled.show()
                sleep(2)

                oled.fill(0)
                oled.text("Promedio de", 0,0)
                oled.text("muestras", 0, 16)
                oled.text(str(prom_temp) + "C", 0, 32)
                oled.show()

                sleep(2)
                self.recuperar(oled, numeros_muestra, temperaturas)
                self.graficar(oled, temperaturas)
                break

if __name__ == "__main__":
    temp = Temperatura()
    activar = True
    sd = temp.microsd()
    oled = temp.pantalla()
    oled.fill(0)
    oled.text("BIENVENIDO", 0, 0)
    oled.show()
    sleep(2)

    while activar:
        temp.main(oled)
        activar = temp.avisar(oled) #<- por si quiere iniciar de nuevo
```

3.2. Circuito



3.3. Pinout del LM35 y Datasheet



LM35 Pinout:

1. +Vs
2. Vout
3. GND

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

3.4. Bibliografía

Python para impacientes. Enlace: https://python-para-impacientes.blogspot.com/2017/03/el-modulo-time.html

Clase N° 6.1, Raspberry Pi Pico "MicroPython" (Sensor de temperatura LM35). Enlace: https://www.youtube.com/watch?v=d11pck6qrzc&t=44s

LM35 Precision Centigrade Temperature Sensors. Enlace: https://www.ti.com/lit/ds/symlink/lm35.pdf

string — Common string operations. Enlace: https://docs.python.org/es/3/library/string.html