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

Lectura de sensores y su visualización a través de una pantalla con la utilización de un raspberry pi pico

Prof. Rene Cabrera

Elaborado por:

Brazon Eulises

Barcelona, 6 de febrero del 2023

ABSTRACTO

En la presente practica indagaremos sobre el uso de distintos sensores con el uso del rasberripy pico, como realizar las respectivas conexiones y lecturas, con cada uno de los sensores utilizados, dependiendo del tipo de sensor a emplear, ya sea un sensor analógico o un sensor digital, posterior a ello se tomaran estos valores para ser procesados y mostrados a través de una pantalla, con ayuda de librerías que contiene una serie de instrucciones básicas que simplifican el proceso de desarrollo en el código empleado. Para el desarrollo del código se estará empleando microphyton.

OBJETIVOS

Objetivo General:

- Efectuar lectura de distintos sensores y poder visualizarlos a través de una pantalla utilizando el rasberripy pico

Objetivos Específicos:

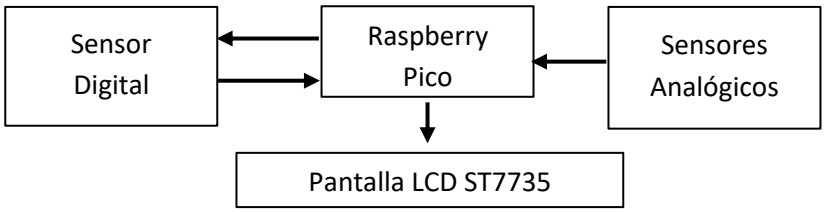
- Conectar un potenciómetro al rasberripy pico.
- Medir la temperatura a través de un sensor conectado al rasberripy pico.
- Configurar una fotorresistencia al rasberripy pico.
- Evaluar sensor de distancia de ultrasonido con el rasberripy pico.
- Implementar una pantalla en el rasberripy pico.

MATERIALES REQUERIDOS

- Potenciómetro (en este caso se utilizó uno de 10KΩ / analógico).
- LM35 (sensor de temperatura / analógico).
- Fotorresistencia (sensor de luz / analógico).
- HC-SR04 (sensor de distancia por ultrasonido / digital).
- Pantalla LCD TFT 1,8 pulgadas chip de control ST7735, de 128X160px.
- Raspberry pico.
- Resistencia 1KΩ (1unidad) y 220Ω (2unidades).
- Transistor 2N2222.
- Protoboard.
- Jumpers para la conexión.
- Alimentación 5V.
- Herramientas para la manipulación y testeo (multímetro, pinzas, entre otros).

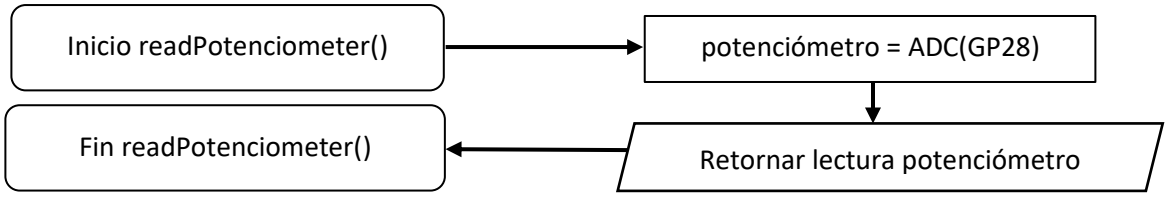
DESCRIPCION DEL PROYECTO

Diagrama de Bloques: Primero observemos una visión general de los distintos componentes a utilizar y cómo van a estar interactuando en conjunto antes de pasar a los circuitos utilizados en cada uno de ellos.

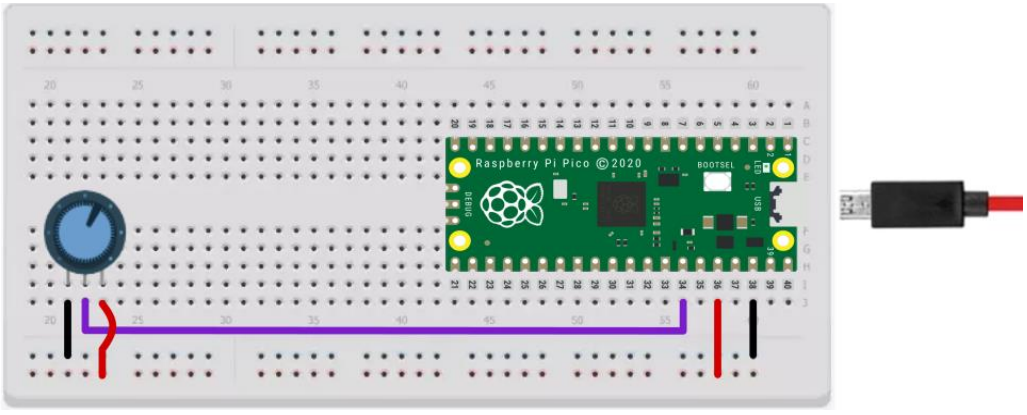


Se conecto un **potenciómetro** de 10KΩ, este potenciómetro generara una señal de tipo analógica cuya señal no debe variar su valor, a no ser, que se rote, ya sea en sentido horario o anti horario, para la lecturas de señales analógica el rasberripy pico cuenta con tres entrada ADC (Analog to Digital Converter), el GP28 es el utilizado en este caso, la señal recibida es procesada con ayuda de la librería machine.ADC, la cual nos ayuda en la conversión del voltaje que se está recibiendo en la entrada analógica del rasberripy pico (varía entre 0 a 3.3v), a un valor digital (en este caso de 16 Bits, que varía entre 0 y 65535), siendo este valor digital, el que entiende el rasberripy pico para realizar las operaciones necesarias.

Diagrama de flujo del Potenciómetro



Circuito Potenciómetro:

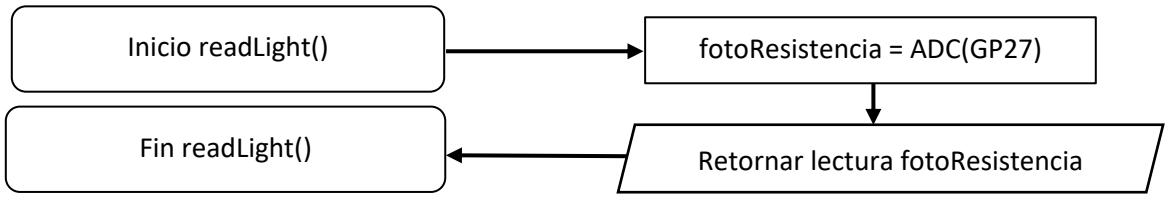


Código Potenciómetro:

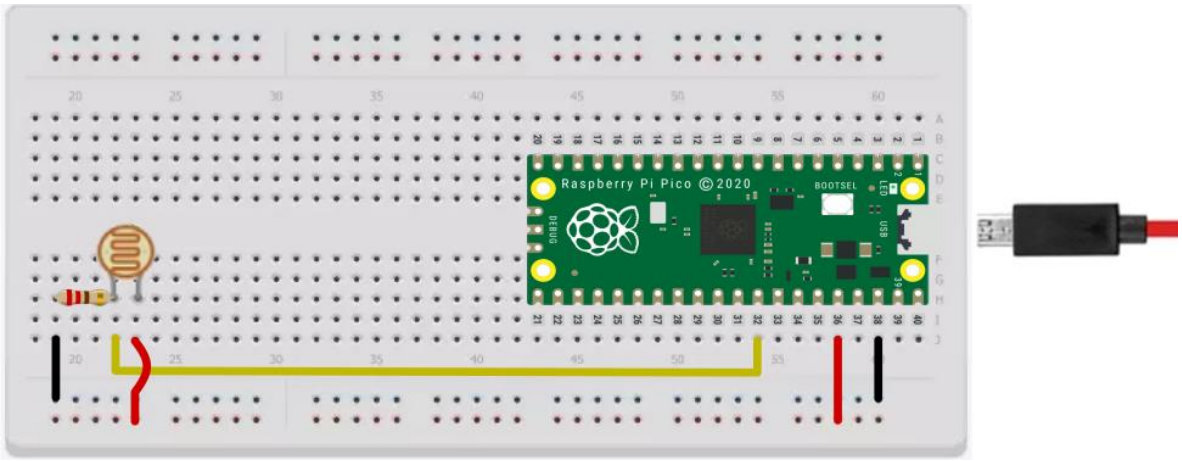
```
from machine import ADC

#define pin for potenciometro
potenciometer_analog_value = ADC(28)

def readPotenciometer():
    return potenciometer_analog_value.read_u16()
```

Ahora, para configurar una **fotorresistencia** (resistencia que varía su resistencia en función de la luz percibida), es necesario aplicar un divisor de voltaje para poder generar la señal analógica que necesitamos, para conectarlo a uno de los pines ADC que posee el rasberripy pico, en este caso se utilizó el GP27**,** esto se logra aplicando una resistencia en serie a la fotorresistencia (se usó una resistencia de 220Ω), esto es necesario solo si solo poseemos la fotorresistencia por si sola, pero existen módulos que integran esta resistencia en un circuito impreso, esta conexiones queda mejor referenciado en el Circuito de la Fotorresistencia.

Diagrama de flujo de la Fotorresistencia:



Circuito Fotorresistencia:

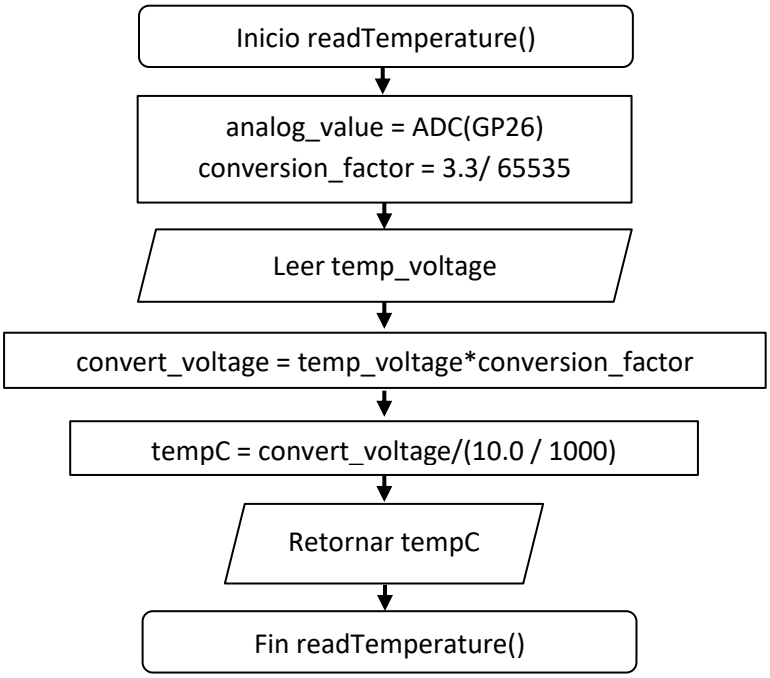


Código Fotorresistencia:

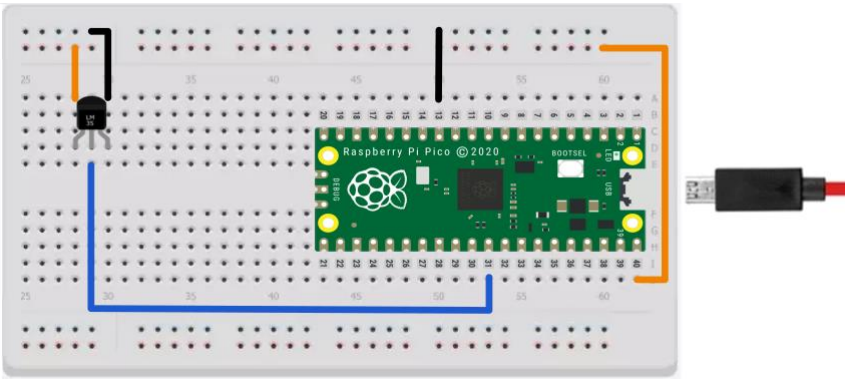
```
from machine import ADC

#define pin for photoresist
photoResist = ADC(27)

def readLight():
    return photoResist.read_u16()
```

En la conexión del sensor de **temperatura LM35** se sigue un procedimiento similar, conectamos los pines de GND, alimentación y señal, con la acotación de que este sensor para poder funcionar necesita una entrada de voltaje de al menos 4 Voltios para funcionar, por lo que se procede a alimentar desde el pin 40 que nos suministra 5 Voltios proveniente del cable micro usb, para la lectura del sensor de temperatura se designó el GP26, la lectura obtenida de este sensor no representa directamente la medición de la temperatura, para ello, se hace necesario procesar la señal para hacer la conversión necesaria, dado que lo que estamos obteniendo a la entrada es un voltaje (el LM35 varia su conductividad en función de la temperatura), con el voltaje obtenido se puede calcular la temperatura aplicando un factor de conversión, este factor de conversión es proporcionado por el fabricante.

Diagrama de flujo para el LM35:



Circuito LM35:



Código LM35:

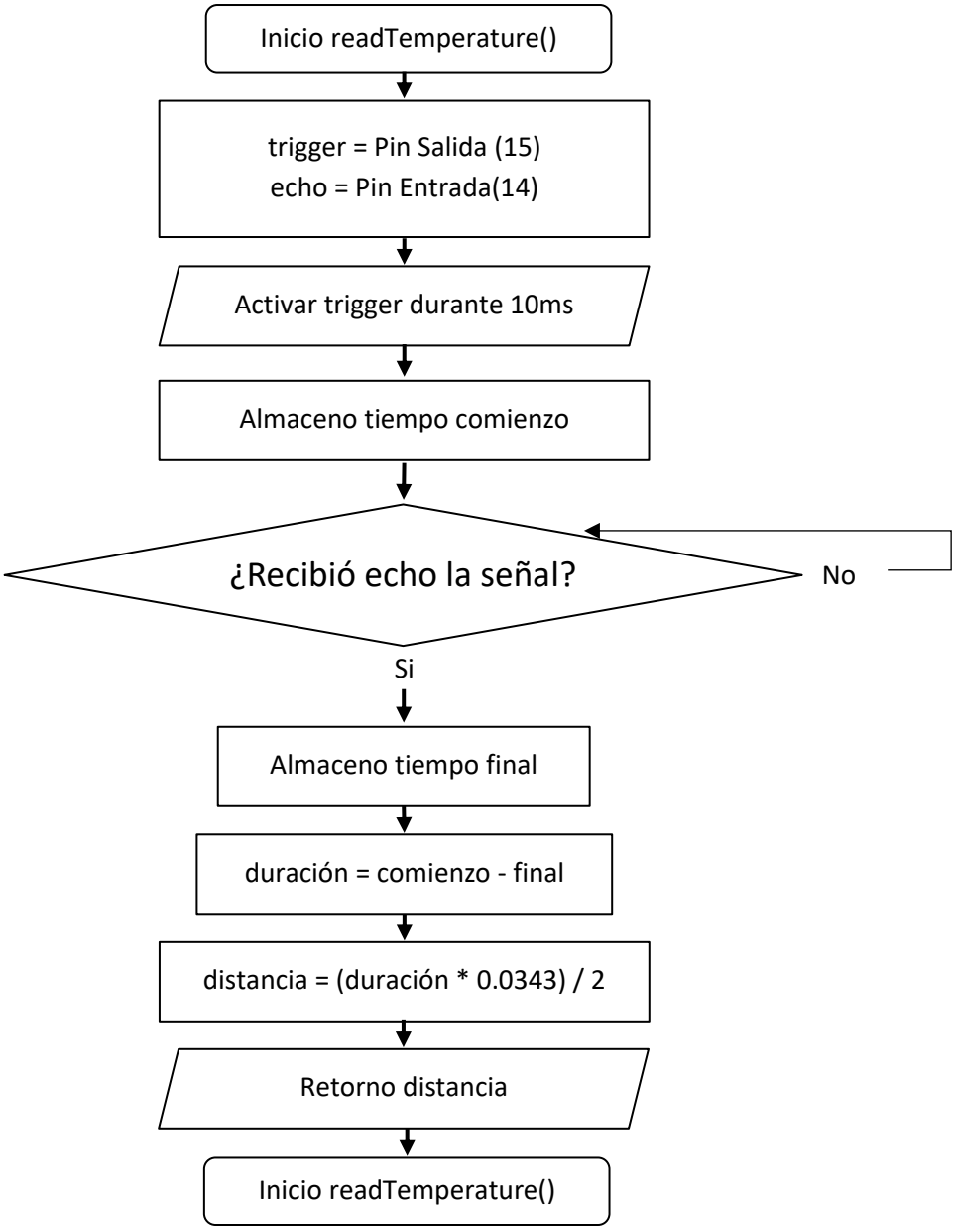
```
from machine import ADC

#define value for read LM35
analog_value = ADC(26)
conversion_factor = 3.3/ 65535

def readTemperature():
    temp_voltage_raw = analog_value.read_u16()
    convert_voltage = temp_voltage_raw*conversion_factor
    tempC = convert_voltage/(10.0 / 1000)
    return tempC
```

Todos los sensores utilizados hasta ahora, trabajan con señales analógicas, a continuación se evaluara un sensor digital, el **sensor de distancia de ultrasonido HC-SR04,** este sensor nos ayuda a evaluar la distancia de un objeto, cuenta un transmisor que emite un sonido a una alta frecuencia, que no puede ser percibido por el oído humano, esta onda transmitida rebota con el objeto que se encuentre frente y con el uso un receptor que capta este sonido de alta frecuencia, obtenemos el tiempo que tarda entre que se envía la señal y el tiempo en que se recibió, lo que nos ayuda para posteriormente calcular la distancia que recorrió la onda.

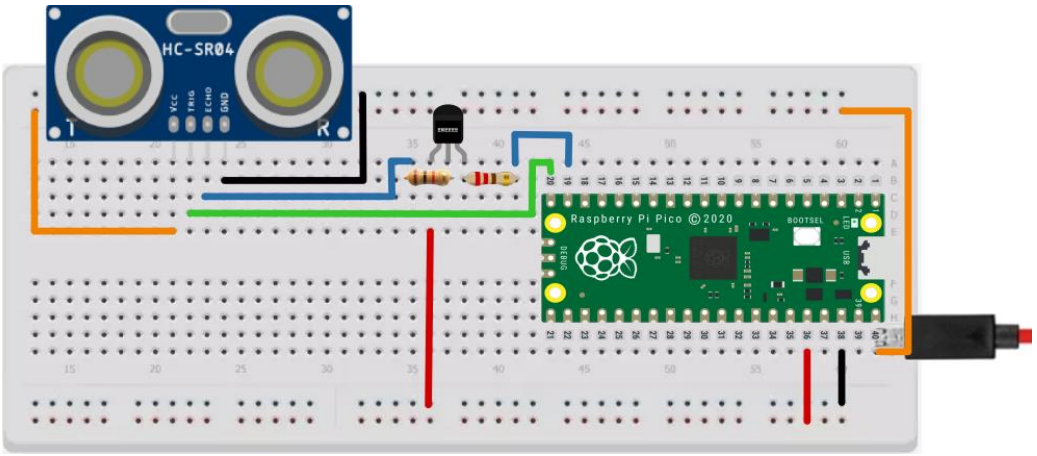
En la programación de los pines digitales se emplea la librería machine.Pin que nos ayuda a indicar si el pin será empleado como entrada o como salida de dato, para el caso del transmisor (trigger) se crea un pin de salida, y para el caso del receptor (echo) se crea un pin de entrada, una acotación **importante** es que este sensor trabaja con 5v, y su señal generada por el receptor también es de 5V, es necesario reducir este voltaje a los 3.3 V que utiliza el rasberripy pico, para ello tenemos varias opciones, como el uso de una resistencia en seria de al menos 1kΩ,o un divisor de tensión, o como en este caso, un transistor junto a dos resistencias.

Diagrama de flujo del sensor de distancia de ultrasonido HC-SR04:

- El trigger es la señal para activar el transmisor, se usó el GP15
- El echo es la señal que se activa cuando el sonido es recibido, se usó el GP14.
- La fórmula empleada está basada en la velocidad del sonido.



Circuito sensor de distancia ultrasónico HC-SR04:



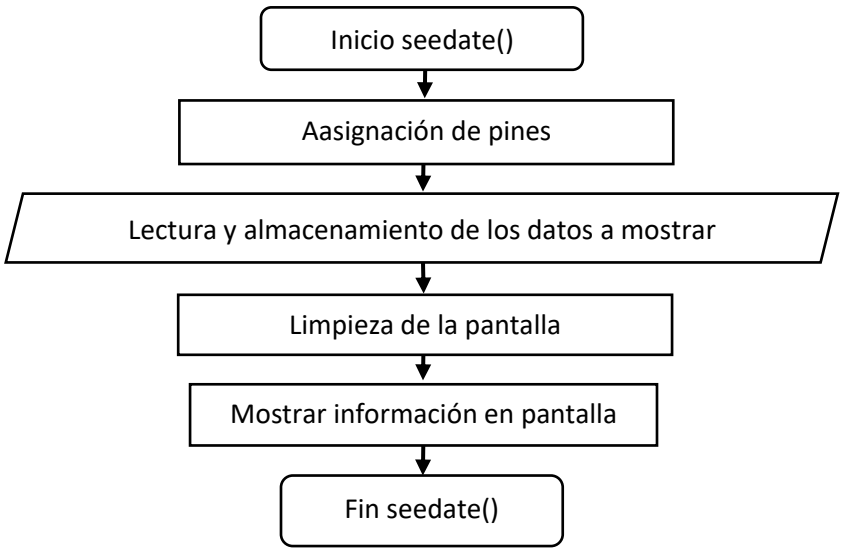
Código sensor de distancia ultrasónico HC-SR04:

```
from machine import Pin
from machine import ADC
import utime

#define pin distance
trigger = Pin(15,Pin.OUT)
echo = Pin(14, Pin.IN)

def readDistance():
    trigger.high()
    utime.sleep_ms(10)
    trigger.low()

    while echo.value() == 0:
        star = utime.ticks_us()
    while echo.value() ==1:
        end = utime.ticks_us()

    duration = end - star
    distance = (duration * 0.0343) / 2
    return distance
```

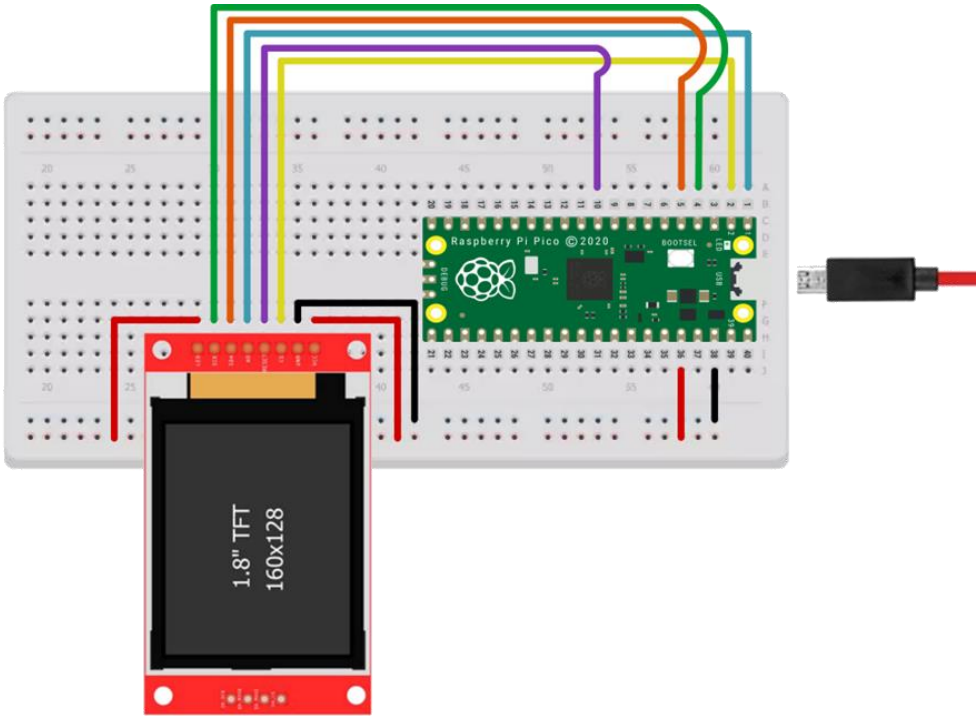
Luego de tener una serie de datos que podemos obtener de diferentes sensores, cada uno con sus respectivas conexiones y procesamiento de la señal para poder interpretar los datos, se necesita contar con una forma de visualizarlos, para ello se implementa una **pantalla LCD** TFT 1.8 pulgadas, chip de control ST7735, con una resolución de 128x160px. En su manipulación es necesario contar la librería para este modelo en particular, que nos brinda una serie de instrucciones básicas para poder imprimir estos datos en pantalla enfocándonos en el desarrollo de la lógica del programa en lugar de preocuparnos en la manipulación de pixel por pixel de la pantalla.

Adicional al uso de la librería para la pantalla ST7735 se hace uso de la librería machine.SPI, que es un protocolo de comunicación serial que emplea la pantalla para recibir la información.

Diagrama de flujo general en el uso de la pantalla ST7735:



Circuito pantalla ST7735:



	Conexiones
ST7735	Rasberripy
LED	3.3V
SCK	GP02
SDA	GP03
A0	GP00
RESET	GP07
CS	GP01
GND	GND
VCC	3.3V

Todos estos Elementos fueron utilizados en conjunto en un mismo circuito, el **código** empleado parasu funcionamiento, además de una seria de imágenes y video del circuito en funcionamiento, se encuentra disponible en el **repositorio personal**:

https://github.com/EulisesBrazon/sensors_and_display

Proyectos Referenciales:

Clase N° 6.1, Raspberry Pi Pico "MicroPython" (Sensor de temperatura LM35). Recuperado de: https://youtu.be/d11pck6qrzc

Medir Distancia con Sensor Ultrasónico Raspberry Pi Pico y Lectura en Pantalla Oled con Micropython. Recuperado de: https://youtu.be/u30R9aXTTx8

Raspberry PI Pico + St7735 + Micropython. Recuperado de: https://youtu.be/Xs78cCnkDVw