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

Implementación
Protocolo de comunicación UART
y Almacenamiento de Información

Prof. Rene Cabrera

Elaborado por:

Brazon Eulises

Barcelona, 13 de febrero del 2023

ABSTRACCION

A través de dos raspberry pi pico (emisor y receptor), se estará implementando el protocolo de comunicación serial para la comunicación entre estos dos dispositivos, la presente practica está basada en la práctica anterior de "*Lectura de sensores y su visualización a través de una pantalla con la utilización de un raspberry pi pico*", el cual se puede conseguir en a través del siguiente enlace **https://github.com/EulisesBrazon/sensors_and_display**, en él se detalla de manera simplificada cada uno de los circuitos implementados, en esta ocasión se estará trabajando con los diversos circuitos funcionando en conjunto. Y el análisis se centrará en análisis del protocolo UART, utilizado para realizar la comunicación serial. Además del UART también ser utilizara archivos para que el receptor pueda ir almacenando la información que va recibiendo.

OBJETIVOS

Objetivo General:

- Entender el funcionamiento del protocolo UART

Objetivos Específicos:

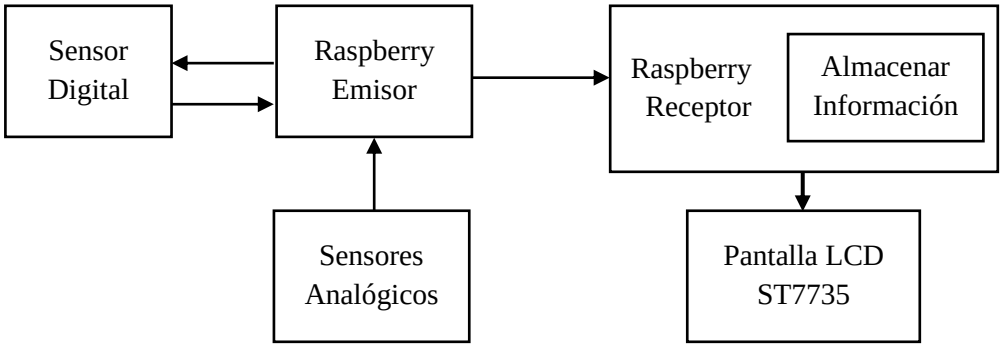
- Conocer como enviar información a través de UART
- Comprender como recibir información a través de UART
- Realizar el almacenamiento de información

MATERIALES

- Potenciómetro (en este caso se usó 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.
- Dos raspberry pi 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 bloque: A continuación, se muestra de manera simplificada como los distintos dispositivos se van a estar comunicando entre ellos



Se conectaron distintos sensores en el raspberry pi pico emisor el cual ayuda con la captación de distintos datos (Potenciómetro, Nivel de luz, sensor de Distancia, Sensor de Temperatura), todos estas datos deben organizarse correctamente antes de ser enviados por medio del protocolo UART, para que el receptor pueda identificar cuando está recibiendo la información referente al potenciómetro, o referente al nivel de luz, el receptor debe identificar a que sensor pertenece esa información que está recibiendo, para ello se decide enviar la información de madera ordenada, siendo así que el orden de los bytes, determina a quien pertenece la información que se está leyendo, adicional a ello se envía un byte "header", usado para identificar a partir de donde el receptor comenzara a realizar la lectura de los valores, para el grupo de bytes que está recibiendo.

Otro aspectos a considerar es lo siguiente, la información que deseamos enviar esta compuesta por 16 Bits, pero el protocolo UART envía 8bits en cada pulso de información, es por ello que se hace necesario descomponer la información de 16 bits en dos grupos de 8 bits lo cual se hace con la ayuda del método .to_bytes que forma parte la clase Integer, y al momento de reconstruir la información en el receptor se hace uso del método .from_bytes, que permite reconstruir esta información.

También hay que considerar un tiempo prudente en el envío de los datos, ya que el receptor debe procesar la información que está recibiendo y mostrarla, este proceso demora un cierto tiempo, y si el emisor envía información más rápido de lo que el receptor puede procesarla, se saturará el buffer donde se almacena la información temporalmente, ya que se estará llenando más rápido de lo que se vacía. Además de que la información que empezara a mostrar en pantalla serán de lecturas antiguas que aún no han sido procesadas, en lugar de lenturas con los valores actuales.

Para el manejo de archivo se implementa la función "open()" el cual posee diferentes modos de apertura (r,a,w,x) para diferentes tipos de archivo (t,b), con ayuda de esta función el receptor almacenara la información recibida en su propia memoria interna.

La Información a transmitir se representaría de la siguiente forma:

Head	Temperatura		Nivel Luz		Potenciómetro		Distancia	
8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits

Diagrama de Flujo emisor sendDates():

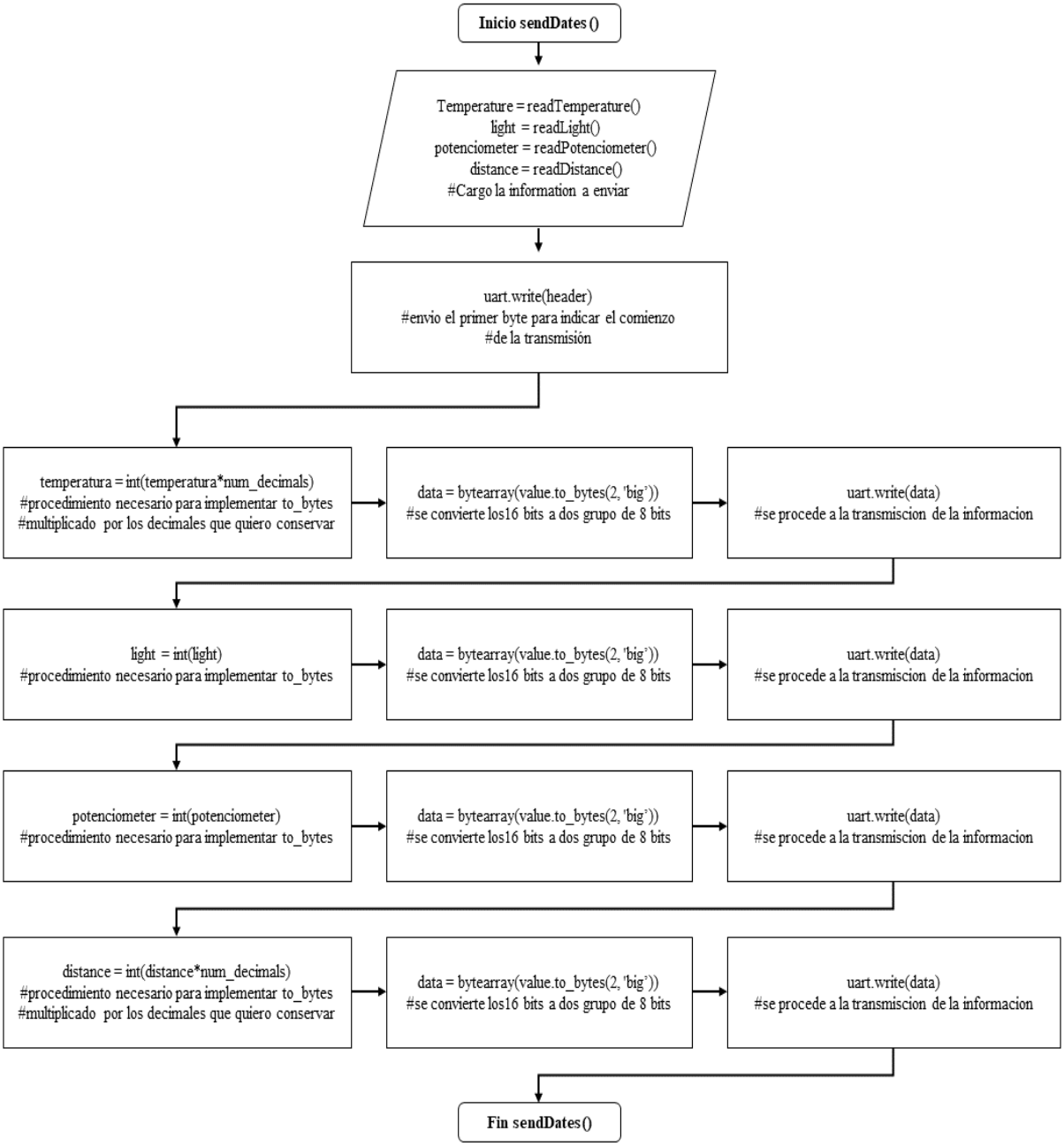


Diagrama de Flujo receptor readDate():

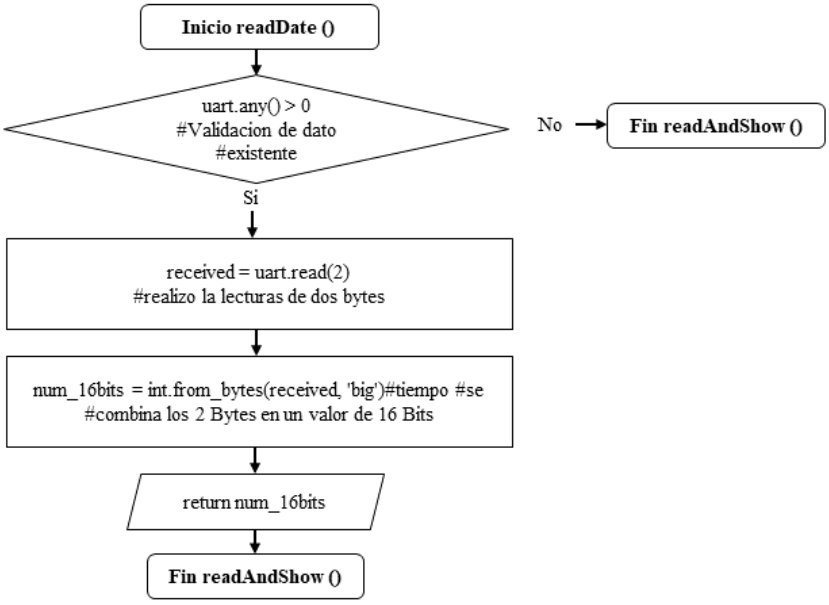
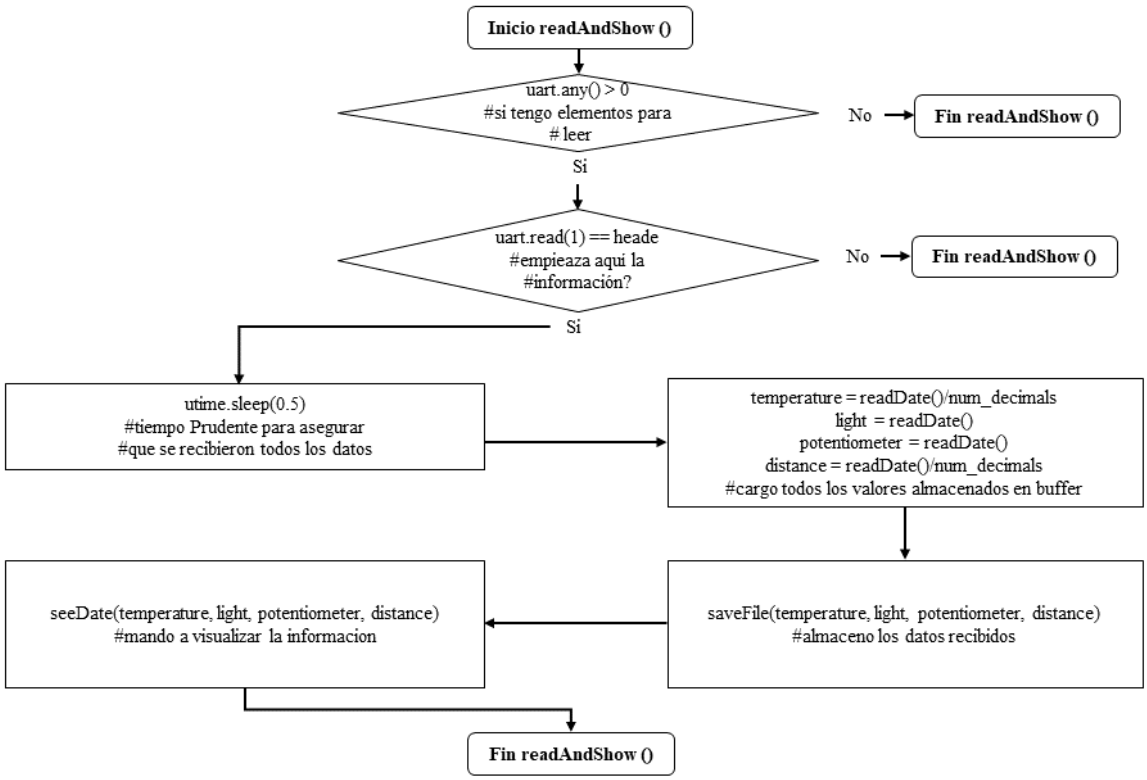


Diagrama de Flujo receptor readAndShow():



Circuito Electrónico

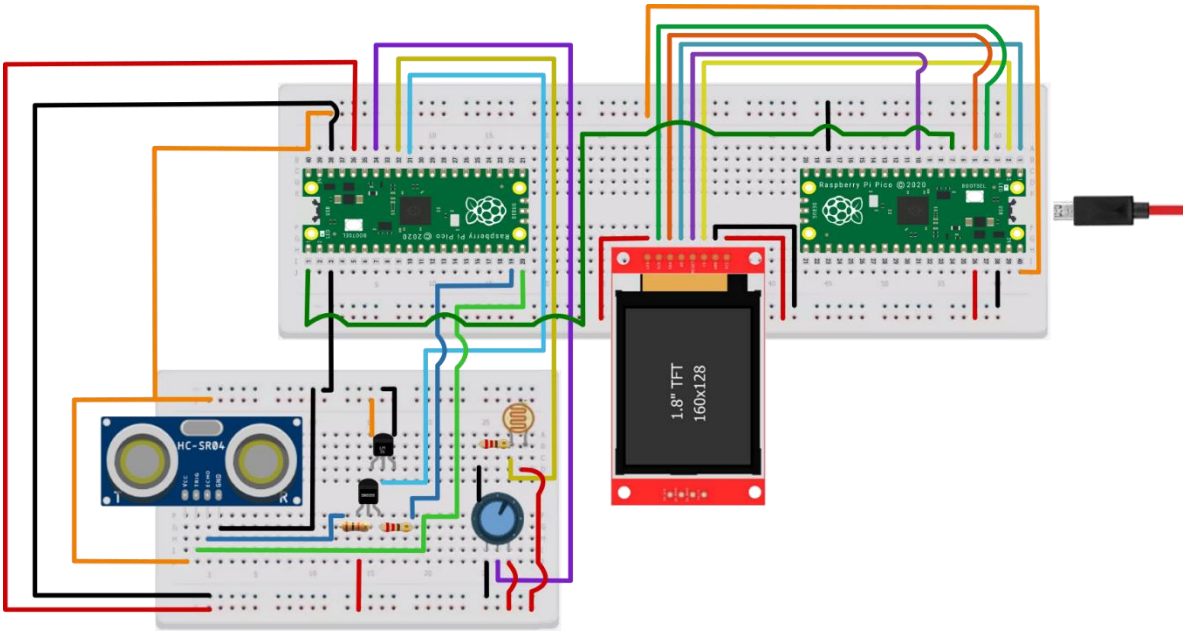
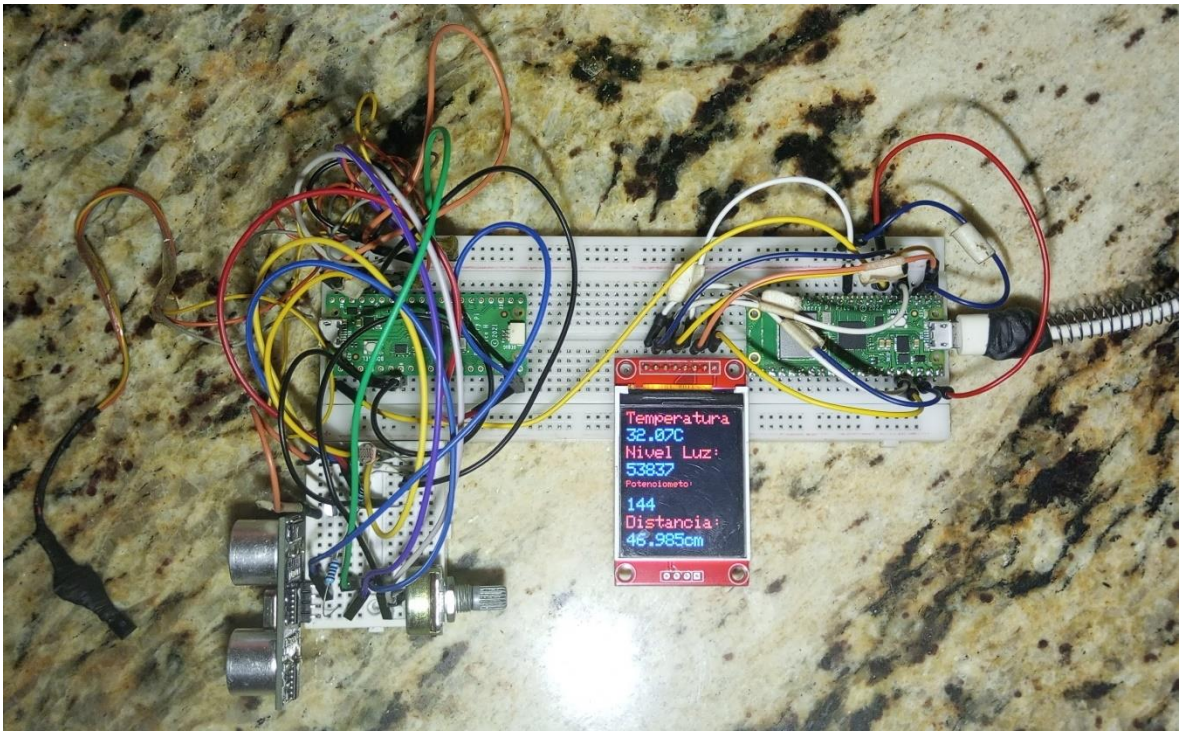


Imagen del circuito



Código alojado en:

https://github.com/EulisesBrazon/sensors_and_display_UART

Código Utilizado (Pico-emisor):

```python
from machine import UART
from machine import Pin,ADC
import machine
import utime

#define values for uart communication
uart= UART(0,9600)
uart.init(9600, bits=8, parity=0, stop=1)
num_decimals = 1000 #every 0 is a decimal
delay = 1 #one second
slaveA = b'0xA'
freeBus = Pin(16, Pin.IN)

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

#define pin for photoresist
photoResist = ADC(27)

#define pin for potenciometro
potenciometer_analog_value = ADC(28)

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

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

def readLight():
    return photoResist.read_u16()

def readPotenciometer():
    return potenciometer_analog_value.read_u16()

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

def sendValueInt(value):
    value = int(value)
    data = bytearray(value.to_bytes(2, 'big'))#convert 16-bit integer value to two 8-bit bytes
    uart.write(data) #send two bytes

def sendValueFloat(value):
    value = int(value*num_decimals)#save some decimals
    data = bytearray(value.to_bytes(2, 'big'))#convert 16-bit integer value to two 8-bit bytes
    uart.write(data) #send two bytes

def sendDates():
    #loading values
    temperature=readTemperature()
    light = readLight()
    potenciometer = readPotenciometer()
    distance = readDistance()

    #sending values
    uart.write(header)
    sendValueFloat(temperature)
    sendValueInt(light)
    sendValueInt(potenciometer)
    sendValueFloat(distance)
```

```
77
78 def main():
79     while True :
80         try:
81             sendDates()
82         except Exception as e:
83             print("Error:", e)
84         utime.sleep(2)#time required for the information to be displayed on the screen
85
86 if __name__ == '__main__': #entry point
87     main()
```

Código Utilizado (Pico-Receptor)

```
1  from machine import UART
2  from ST7735 import TFT
3  from machine import SPI,Pin,ADC
4  from sysfont import sysfont
5  import utime
6  import math
7
8
9
10 #separator for the text file
11 separator = " "
12
13 #define values for uart communication
14 uart= UART(1,9600)
15 uart.init(9600, bits=8, parity=0, stop=1)
16 num_decimals = 1000 #every 0 is a decimal
17 delay = 1 #one second
18 header = b'A'
19
20 #difien values for displey use
21 spi = SPI(0, baudrate=20000000, polarity=0, phase=0, sck=Pin(2), mosi=Pin(3), miso=Pin(4))
22 tft=TFT(spi,0,7,1)
23 tft.initr()
24 tft.rgb(True)
25
26 #received decimals
27 num_decimales = 1000 #cada 0 es un decimal
28
29 def seeDate(temperature, light, potentiometer, distance):
30
31     temperature = str(temperature)
32     light = str(light)
33     potentiometer = str(potentiometer)
34     distance = str(distance)
35
36     size=2
37     separation = 4
38     h=0
39
40     tft.fill(TFT.BLACK)#clean screen
41
42     #print values in the sreen
43     tft.text((0, h), "Temperatura:", TFT.RED, sysfont, size, nowrap=True)
44
45     #recalculating height
46     h += sysfont["Height"]*size+separation
47     tft.text((0, h), temperature+"C", TFT.BLUE, sysfont, size, nowrap=True)
48
49     h += sysfont["Height"]*size+separation
50     tft.text((0, h), "Nivel Luz:", TFT.RED, sysfont, size, nowrap=True)
51
52     h += sysfont["Height"]*size+separation
53     tft.text((0, h), light, TFT.BLUE, sysfont, size, nowrap=True)
54
55     h += sysfont["Height"]*size+separation
56     tft.text((0, h), "Potenciometo:", TFT.RED, sysfont, size-1, nowrap=True)
```

```
57
58      h += sysfont["Height"]*size+separation
59      tft.text((0, h), potentiometer, TFT.BLUE, sysfont, size, nowrap=True)
60
61      h += sysfont["Height"]*size+separation
62      tft.text((0, h), "Distancia:", TFT.RED, sysfont, size, nowrap=True)
63
64      h += sysfont["Height"]*size+separation
65      tft.text((0, h), distance +"cm", TFT.BLUE, sysfont, size, nowrap=True)
66
67  def readDate():
68      if uart.any() > 0 :#if there are elements in the buffer
69          received = uart.read(2)#read two bytes
70
71          num_16bits = int.from_bytes(received, 'big') #conversion to 16-bit integer
72          return num_16bits
73
74  def saveFile(temperature, light, potentiometer, distance):
75      try:
76          file = open("Info.txt", "a")#Try, if it doesn't exist
77          file.write(str(temperature)+separator+str(light)+separator+str(potentiometer)+separator+str(distance)+"\n")
78          file.close()
79      except:
80          file = open("Info.txt", "w")#changed the opening mode, which creates the file automatically
81          file.write(str(temperature)+separator+str(light)+separator+str(potentiometer)+separator+str(distance)+"\n")
82          file.close()
83
84  def readAndShow():
85      if uart.any() > 0:
86          if uart.read(1) == header:
87              utime.sleep(0.5)
88              temperature = readDate()/num decimals #some readings have decimals
89              light = readDate()
90              potentiometer = readDate()
91              distance = readDate()/num_decimals
92              saveFile(temperature, light, potentiometer, distance)
93              seeDate(temperature, light, potentiometer, distance)
94
95  def main():
96      #clean sreen
97      tft.fill(TFT.BLACK)
98      #show notification on screen
99      tft.text((0, 50), "Sincronizando...", TFT.GREEN, sysfont, 1, nowrap=True)
100
101     while True:
102         try:
103             readAndShow()
104         except Exception as e:
105             print("Error:", e)
106
107 if __name__ == '__main__':
108     main()
```

Referencias utilizadas:

Python Documentation. *"class UART – duplex serial communication bus".*

Recuperado de: https://docs.micropython.org/en/latest/library/machine.UART.html

The Data School. "Python FILE OPEN". Recuperado de:

https://thedataschools.com/python/file-open/