Soil Moisture Sensor with Raspberry Pi Pico

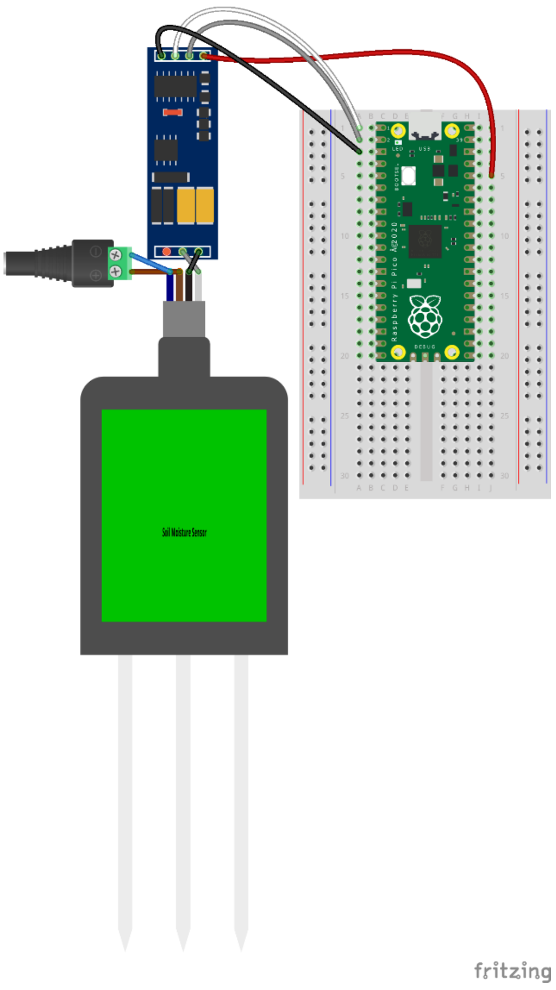
Requirement

- TR-TS Soil Moisture Sensor
- Raspberry Pi Pico
- RS485 Auto Direction Module
- 12 V DC Adapter
- แจ็กแยกขั้วไฟ ตัวเมีย

Language: Micropython

IDE: Thonny

Wiring Diagram

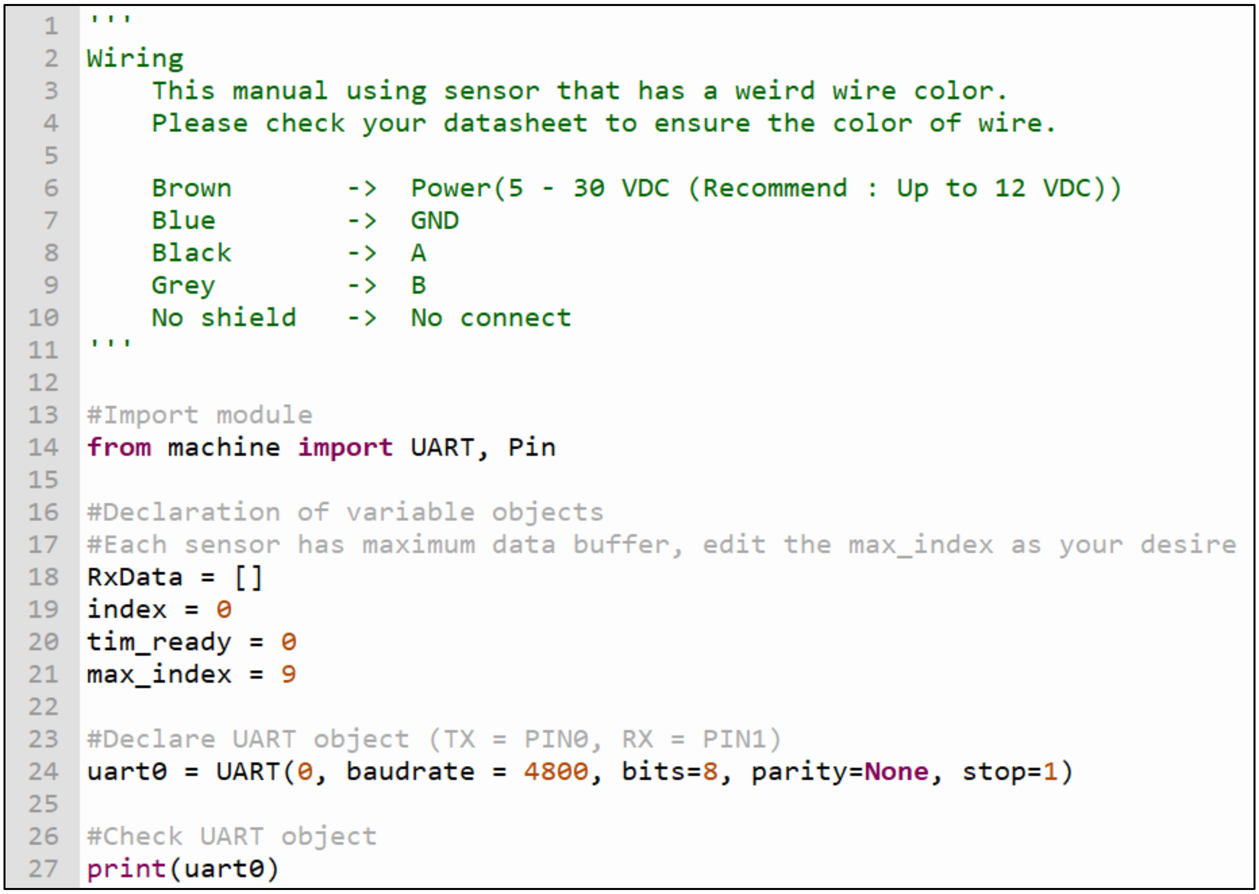


Pico		**RS485**		**PH Sensor**		**12 VDC Adapter**
3V3	↔	VCC		VCC	↔	+
UART0 TX	↔	TXD		GND	↔	-
UART0 RX	↔	RXD				
GND	↔	GND				
		A	↔	A		
		B	↔	B		

Coding

The example program will show the value of soil moisture and temperature every 1 second.

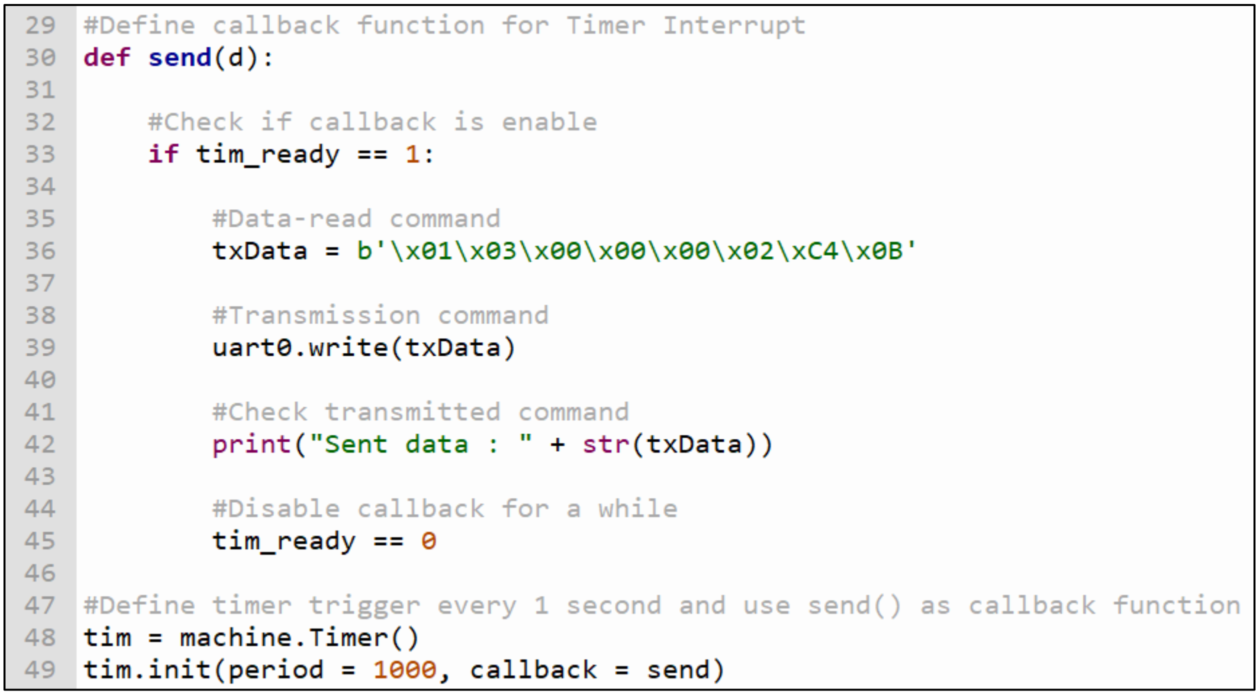
1. Import the module that we use in this program. In this case we only use “machine” as module. Import “UART” and “Pin” to use UART and GPIO of our Pico. Declare objects that we use. In this program, we will name in as “RxData”, “index”, “max_index” and “tim_ready”. RxData” is the empty list that we use to store the buffer that sensor has sent. “index” is the counting variable that use to check and limit data that Pico receive. “max_index” is variable that use to define the maximum value that “index” can reach. For this Soil Moisture sensor, maximum index is equal to 9 according to datasheet. “tim_ready” is variable that use to check if program is ready or not. 1 means the program is ready to request data. 0 means the program is not ready to request data. All declaration is already show in the picture below.



```
'''
Wiring
    This manual using sensor that has a weird wire color.
    Please check your datasheet to ensure the color of wire.

    Brown       ->  Power(5 - 30 VDC (Recommend : Up to 12 VDC))
    Blue        ->  GND
    Black       ->  A
    Grey        ->  B
    No shield   ->  No connect
'''

#Import module
from machine import UART, Pin

#Declaration of variable objects
#Each sensor has maximum data buffer, edit the max_index as your desire
RxData = []
index = 0
tim_ready = 0
max_index = 9

#Declare UART object (TX = PIN0, RX = PIN1)
uart0 = UART(0, baudrate = 4800, bits=8, parity=None, stop=1)

#Check UART object
print(uart0)
```

(Pic.1: Module import and object declaration)

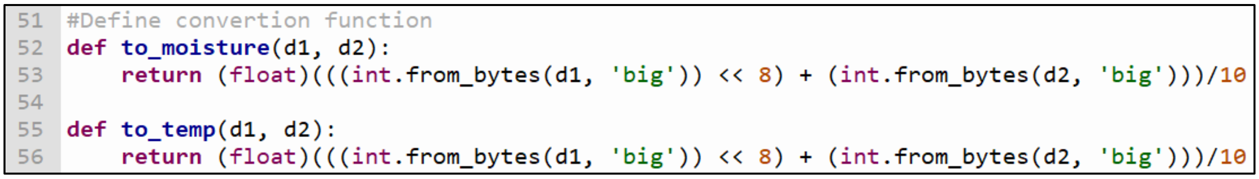
2. To get data from the sensor via Modbus RTU Protocol, we need to send the command to request data every time. This program will use timer interrupt to send request command every 1 second. Remember that every interrupt needs a callback function. So, we define the callback function of timer interrupt and then declare timer object as the picture below (Note that every callback function needs 1 parameter).



```
#Define callback function for Timer Interrupt
def send(d):

    #Check if callback is enable
    if tim_ready == 1:

        #Data-read command
        txData = b'\x01\x03\x00\x00\x00\x02\xC4\x0B'

        #Transmission command
        uart0.write(txData)

        #Check transmitted command
        print("Sent data : " + str(txData))

        #Disable callback for a while
        tim_ready == 0

#Define timer trigger every 1 second and use send() as callback function
tim = machine.Timer()
tim.init(period = 1000, callback = send)
```

(Pic.2: Callback function definition and timer-interrupt object declaration)

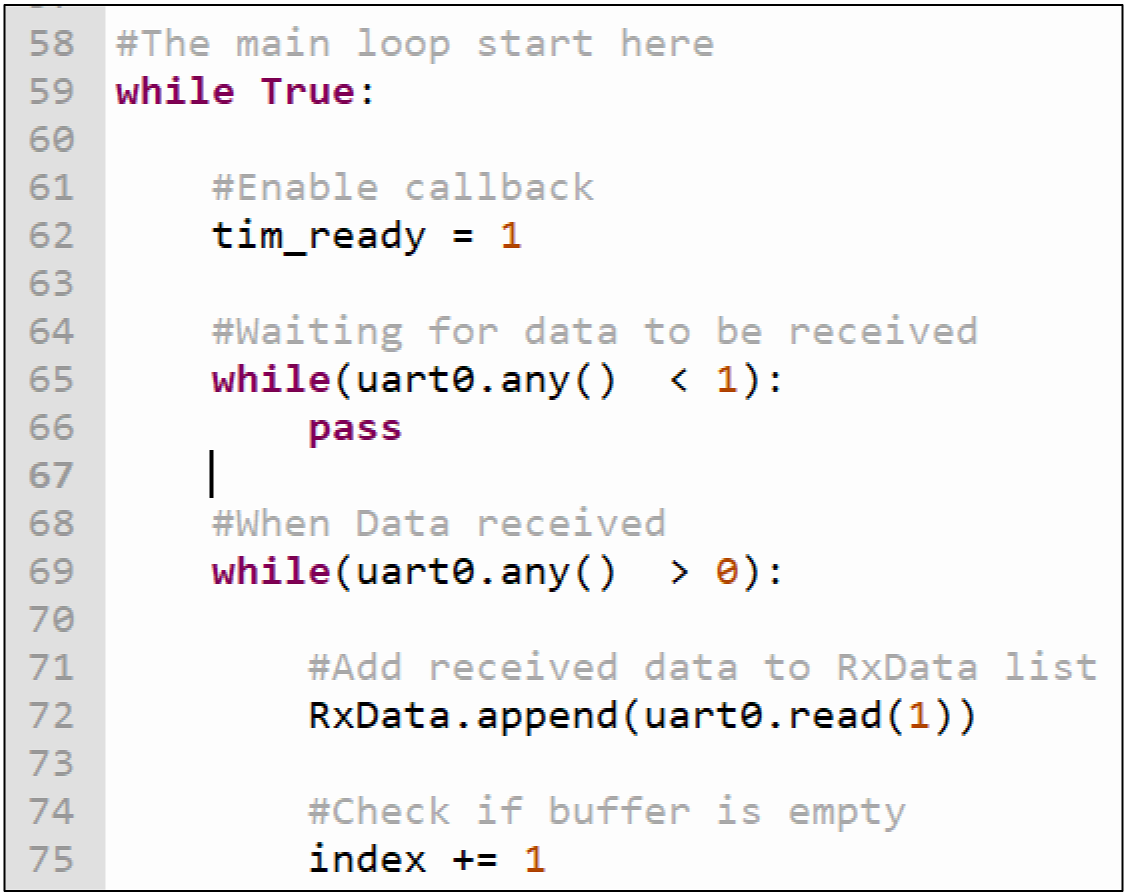
3. (Optional) Define the convert function name “to_moisture” and “to_temp” that return value of moisture and temperature when we pass parameters to function. According to datasheet, we can calculate the actual value by sum all data after convert byte data into int and use bitwise operation then divide it by 10.



```
#Define convertion function
def to_moisture(d1, d2):
    return (float)(((int.from_bytes(d1, 'big')) << 8) + (int.from_bytes(d2, 'big')))/10

def to_temp(d1, d2):
    return (float)(((int.from_bytes(d1, 'big')) << 8) + (int.from_bytes(d2, 'big')))/10
```

(Pic.3: Conversion function definition and data calculation)

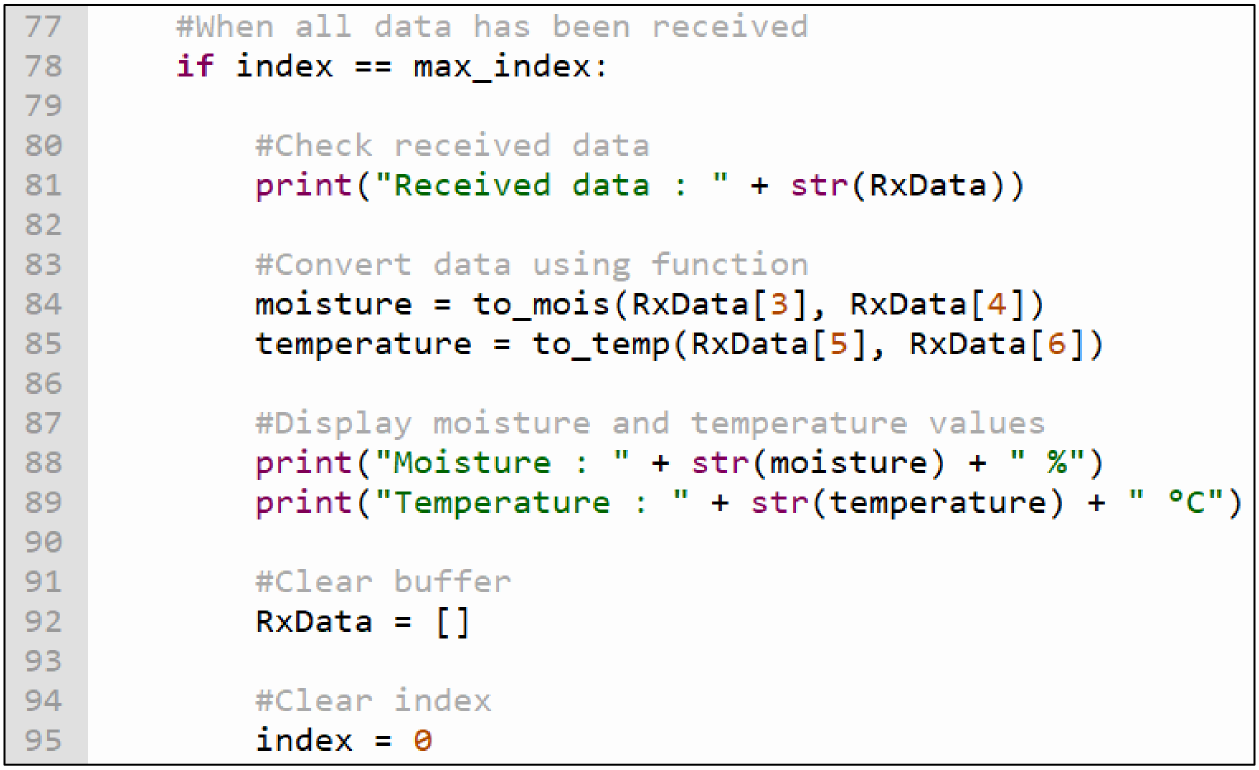
4. Create an infinity loop as a main program (all operation will run in the loop). In the picture below, we will set “tim_ready” to 1 to announce that program is ready to request data, then use loop (in line 65) to wait for data. If the data has been received, Program will be running to another loop (in line 69). This loop will add the data buffer into “RxData” and “index” will increase.



```
#The main loop start here
while True:

    #Enable callback
    tim_ready = 1

    #Waiting for data to be received
    while(uart0.any()  < 1):
        pass

    #When Data received
    while(uart0.any()  > 0):

        #Add received data to RxData list
        RxData.append(uart0.read(1))

        #Check if buffer is empty
        index += 1
```

(Pic.4: Inside the main loop)

5. In the picture below, moisture and temperature value will be calculated after we receive all bytes of message. According to datasheet, the actual value of moisture is the 3rd and 4th byte, actual value of temperature is the 5th and 6th byte we received. Pass the data into "to_moisture" and "to_temp" to get the actual value of moisture and temperature.



```
    #When all data has been received
    if index == max_index:

        #Check received data
        print("Received data : " + str(RxData))

        #Convert data using function
        moisture = to_mois(RxData[3], RxData[4])
        temperature = to_temp(RxData[5], RxData[6])

        #Display moisture and temperature values
        print("Moisture : " + str(moisture) + " %")
        print("Temperature : " + str(temperature) + " °C")

        #Clear buffer
        RxData = []

        #Clear index
        index = 0
```

(Pic.5: Getting actual data from defined functions)