Interfacing TFmini Plus with Raspberry Pi 4B

Written by: Ibrahim (FAE)

This article explains how to interface TFmini Plus with Raspberry Pi 4B but it should work with all other versions of Raspberry Pi as long as the right pins are used. The following assumptions are made while writing this article:

1. The end-user has Raspberry Pi and TFmini Plus in hands
2. The operating system (tested on Raspbian version 10 buster) is installed on Raspberry Pi
3. The serial port is already enabled
4. Python2 or 3 is installed
5. Python editor is required for writing the script. I have Thonny installed on my RasPi.

NOTE: If you have installed full Raspbian, all the required packages are included in full version.

Pin configuration of Raspberry Pi:

The following image shows the pinout of Raspberry Pi board:

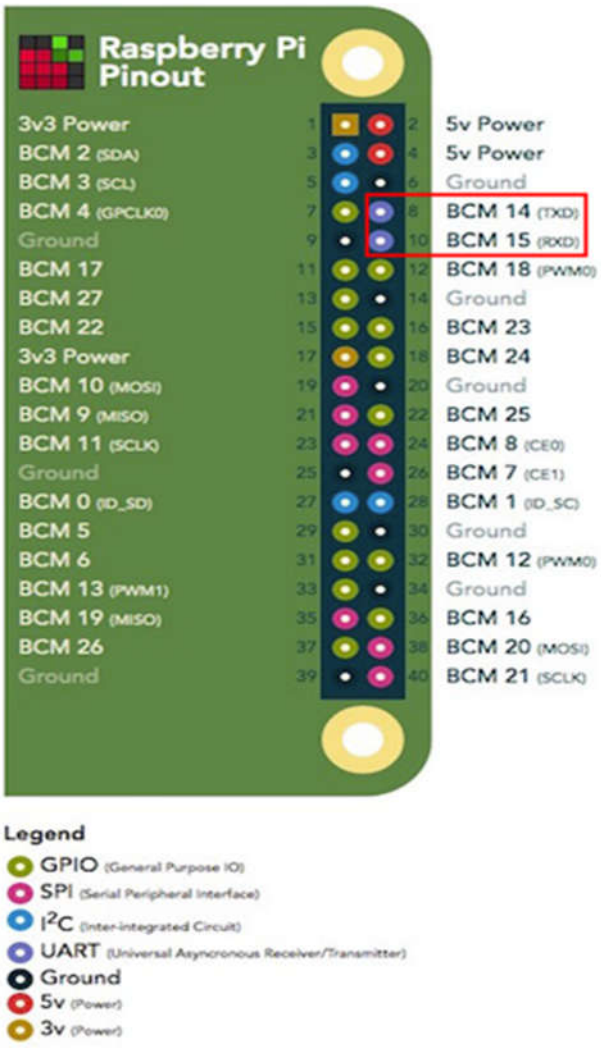

Figure 1: Pinouts

We are using the serial pins for connecting TFmini Plus with Raspberry Pi. **Pin-8 (TXD)** and **Pin-10 (RXD)** as enclosed in red rectangle. **Pin 2 and 6** or **4 and 6** can be used as power source because the operating voltage of TFmini Plus is 5V.

Wire sequence of TFmini Plus:

Figure 2 and table-I show all the details about pin description and the wiring sequence of TFmini Plus.

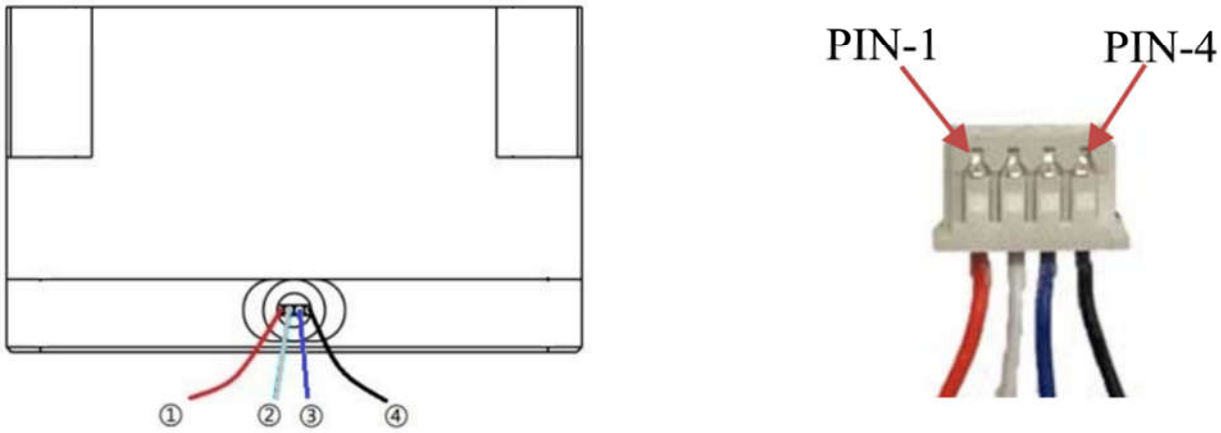

Figure 2: Color code of TFmini Plus

Table-I: Pin description

No.	Color	Corresponding PIN	Funciton	Comment
①	Red	PIN-1	+5V	Power supply
②	White	PIN-2	RXD/SDA	Receiving/Data
③	Blue/Green	PIN-3	TXD/SCL/IO	Transmitting/Clock/IO
④	Black	PIN-4	GND	Ground

Customized cable needed for proper connection:

If we look at the connector of TFmini Plus, it is GH1.25-4P connector. On the other hand Raspberry Pi has pin headers. So we need a cable which has **7P 1.55MM bar connector** (also called horizontal patch socket) on one side and **female DuPont connector** on other side for inserting it into Raspberry Pi. The following figure shows customized cable for this purpose.

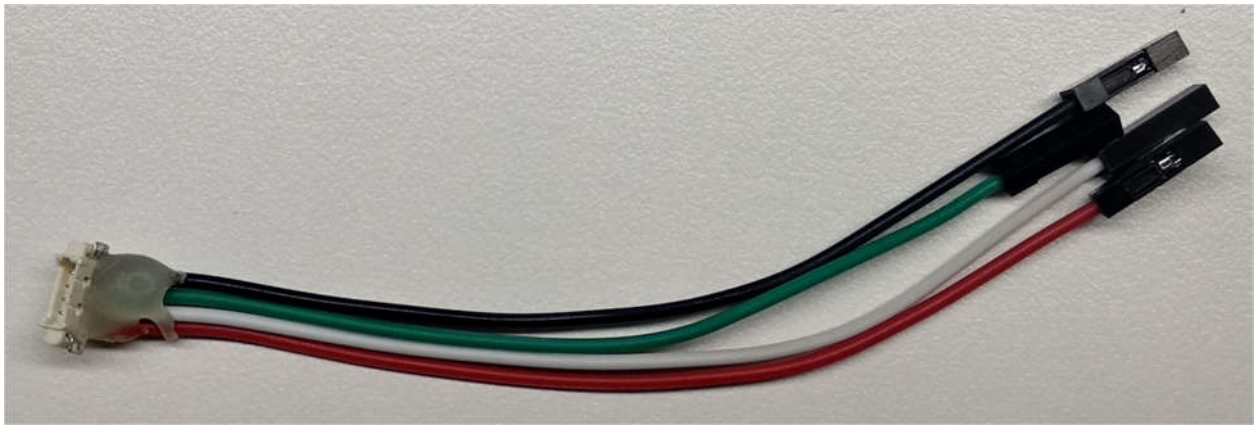

Figure 3: GH1.25 4P to dupont cable

Schematic diagram:

The next image show connection diagram of TFmini Plus with Raspberry Pi.

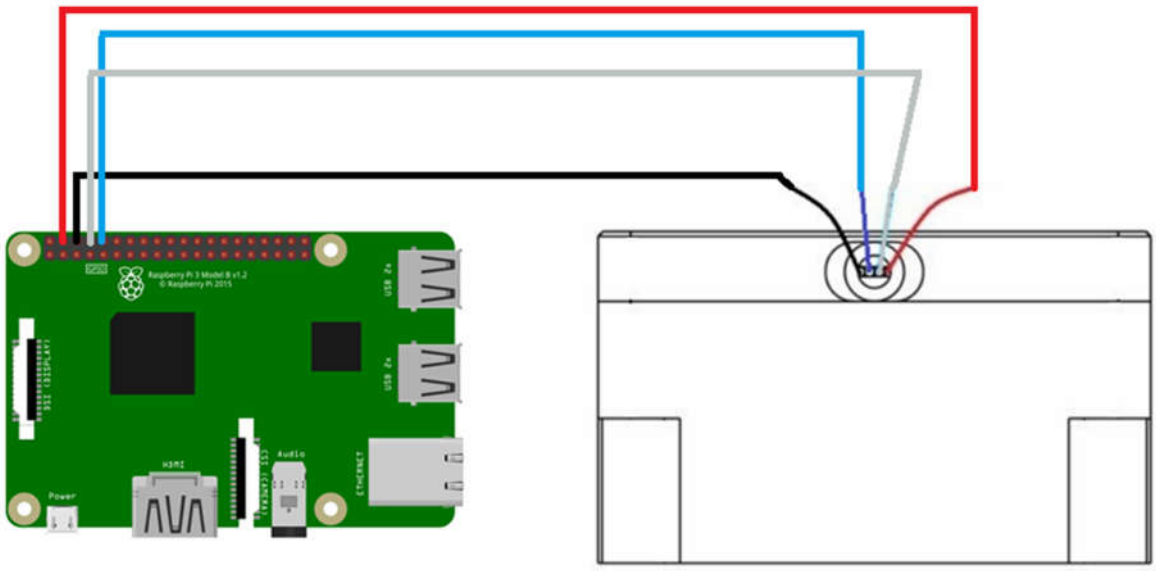

Figure 4: Schematic

It should be noted that if you are connecting more devices to your raspberry Pi, then it is highly recommended to use a high current power supply because the default adapter has only 2A current capacity. So choose your power supply depending upon the current requirement of the whole system. The actual connection settings can be seen in figure 5.

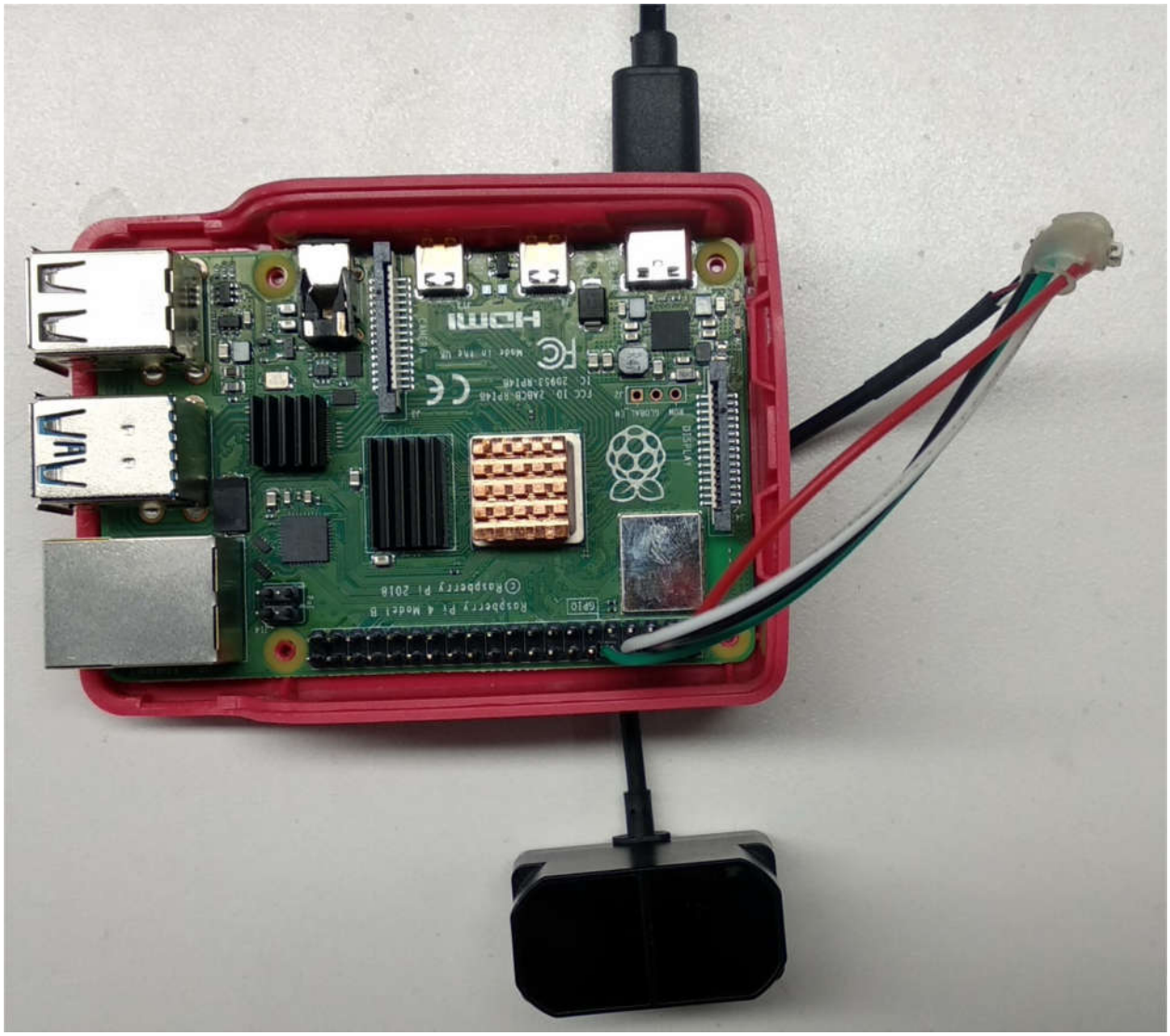

Figure 5: Actual connection

Python code:

In this section we will discuss the main parts of python code that are used for reading the data from TFmini Plus. The following tables show the serial protocol used by TFmini Plus, and data format:

Table II: Communication protocol

Communication interface	**UART**
Default baud rate	115200
Data bit	8
Stop bit	1
Parity check	None

Table III: Data format

Byte0 -1	Byte2	Byte3	Byte4	Byte5	Byte6	Byte7	Byte8
0x59 59	Dist_L	Dist_H	Strength_L	Strength_H	Temp_L	Temp_H	Checksum
Data code explanation							
Byte0	0x59, frame header, same for each frame						
Byte1	0x59, frame header, same for each frame						
Byte2	Dist_L distance value lower by 8 bits						
Byte3	Dist_L distance value higher by 8 bits						
Byte4	Strength_L low 8 bits						
Byte5	Strength_L high 8 bits						
Byte6	Temp_L low 8 bits (suit for version later than V1.3.0)						
Byte7	Temp_H high 8 bits (suit for version later than V1.3.0)						
Byte8	Checksum is the lower 8 bits of the cumulative sum of the numbers of the first 8 bytes.						

Following the above sequence of data, we will get the bytes from TFmini Plus using Python. First of all we need to install the serial port library which can be done using pip installer (works with both Python2 and 3):

```
python -m pip install pyserial
```

After successfully installing the library, it's time to write the script and import the serial port library.

```
import serial
```

Create the object of class Serial and pass the necessary parameters (serial port and baud rate):

```
ser = serial.Serial("/dev/ttyAMA0", 115200)
```

The main function starts from here and it searches if the port is open and call the function *read_data():*

```
if __name__ == "__main__":
    try:
        if ser.isOpen() == False:
            ser.open()
        read_data()
    except KeyboardInterrupt(): # ctrl + c in terminal.
        if ser != None:
            ser.close()
            print("program interrupted by the user")
```

The function *read_data()* checks the bytes, make some necessary shifting of bytes and then prints the data in terminal:

```
def read_data():
    while True:
        counter = ser.in_waiting # count the number of bytes of the serial port
        if counter > 8:
            bytes_serial = ser.read(9)
            ser.reset_input_buffer()

            if bytes_serial[0] == 0x59 and bytes_serial[1] == 0x59: # this portion is for python3
                print("Printing python3 portion")
                distance = bytes_serial[2] + bytes_serial[3]*256 # multiplied by 256, because the binary data is shifted by 8
                strength = bytes_serial[4] + bytes_serial[5]*256
                temperature = bytes_serial[6] + bytes_serial[7]*256
                temperature = (temperature/8) - 256
                print("Distance:"+ str(distance))
                print("Strength:" + str(strength))
                if temperature != 0:

                    print("Temperature:" + str(temperature))
                ser.reset_input_buffer()

            if bytes_serial[0] == "Y" and bytes_serial[1] == "Y":
                distL = int(bytes_serial[2].encode("hex"), 16)
                distH = int(bytes_serial[3].encode("hex"), 16)
                stL = int(bytes_serial[4].encode("hex"), 16)
                stH = int(bytes_serial[5].encode("hex"), 16)
                distance = distL + distH*256
                strength = stL + stH*256
                tempL = int(bytes_serial[6].encode("hex"), 16)


                tempH = int(bytes_serial[7].encode("hex"), 16)
                temperature = tempL + tempH*256
                temperature = (temperature/8) - 256
                print("Printing python2 portion")
                print("Distance:"+ str(distance) + "\n")
                print("Strength:" + str(strength) + "\n")
                print("Temperature:" + str(temperature) + "\n")
                ser.reset_input_buffer()
```

This code has two parts; first part is compatible with Python3 (upper part) and second is compatible with Python2 (lower section). For different byte positions, please refer to Table-III. The distance measured by LiDAR can be observed in the following figure.

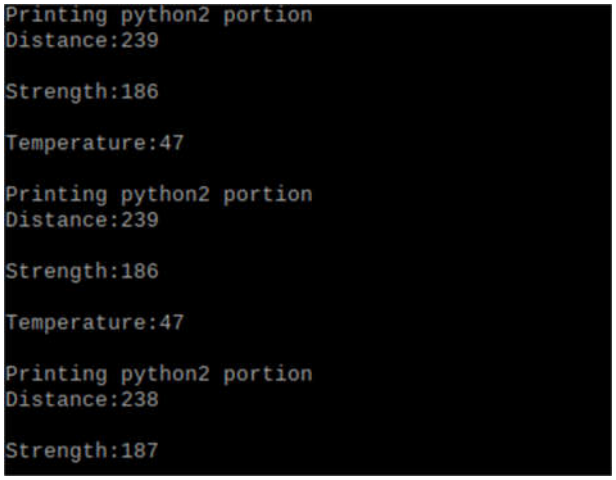

Figure 6: Data measured by LiDAR