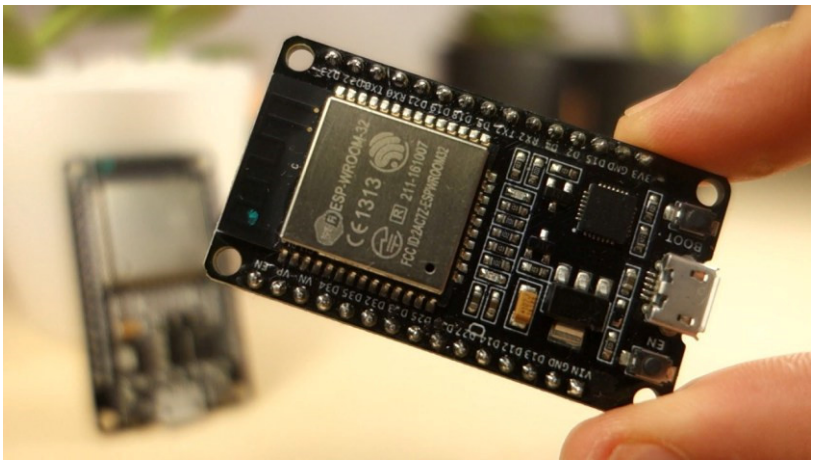
Interfacing of TF Series LiDAR with ESP32 Micro-controller

Written by: Ibrahim (FAE)

Introduction:

In this tutorial, the interfacing of Benewake Single Point LiDARs with ESP32 (ESP32-WROOM-32) is discussed. First of all, the pin-out details of ESP32 will be discussed in details. Next step will be how to install the required library of ESP32 in order to write and burn the code using Arduino IDE. Connection details and code is also the part of this tutorial.

The ESP32 chip comes with 48 pins with multiple functions. Not all pins are exposed in all ESP32 development boards, and there are some pins that cannot be used. This tutorial will identify all those pins which can be used for different types of interfacing. The figure below illustrates **ESP-WROOM-32.**



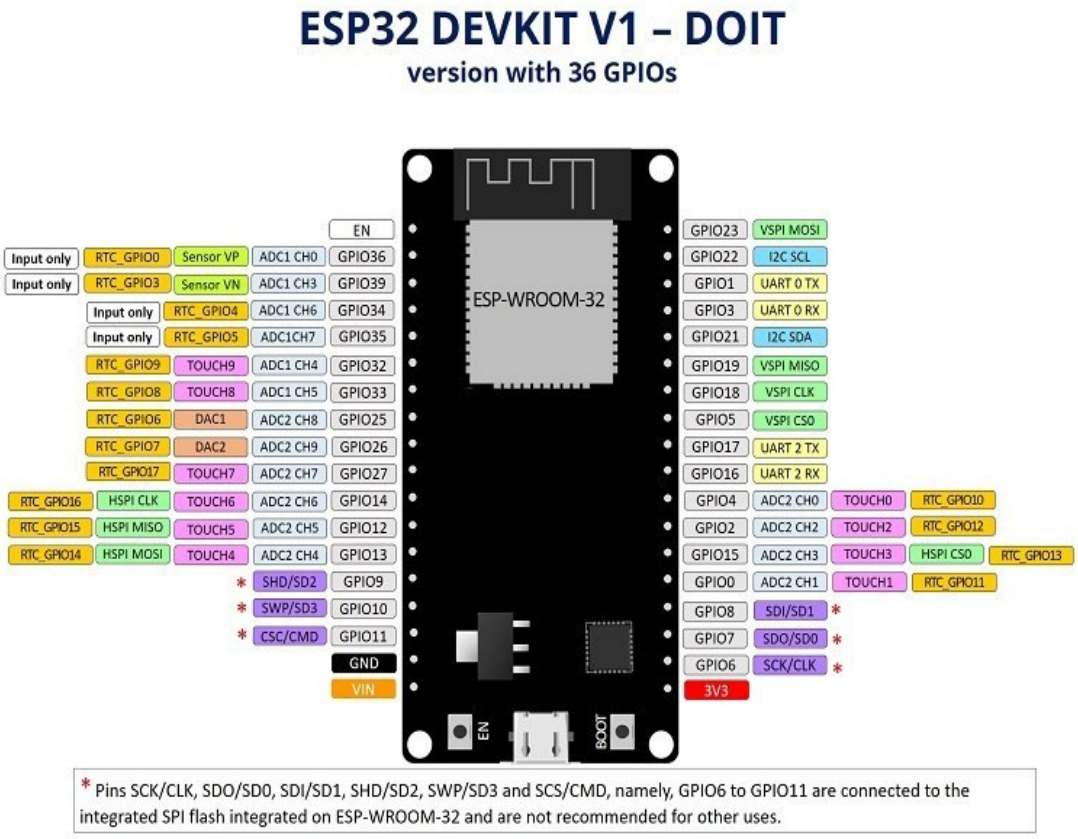
Note: not all GPIOs are accessible in all development boards, but each specific GPIO works in the same way regardless of the development board you're using.

ESP32 Peripherals

The ESP32 peripherals include:

- 18 Analog-to-Digital Converter (ADC) channels
- 3 SPI interfaces
- 3 UART interfaces
- 2 I2C interfaces
- 16 PWM output channels
- 2 Digital-to-Analog Converters (DAC)
- 2 I2S interfaces
- 10 Capacitive sensing GPIOs

The pins highlighted in green are OK to use. The ones highlighted in yellow are OK to use, but you need to pay attention because they may have unexpected behavior mainly at boot. **The pins highlighted in red are not recommended to use as inputs or outputs**.



GPIO Details of ESP32:

GPIO	Input	Output	Notes
0	pulled up	OK	outputs PWM signal at boot
1	TX pin	OK	debug output at boot
2	OK	OK	connected to on-board LED
3	OK	RX pin	HIGH at boot
4	OK	OK	
5	OK	OK	outputs PWM signal at boot
6	x	x	connected to the integrated SPI flash
7	x	x	connected to the integrated SPI flash
8	x	x	connected to the integrated SPI flash
9	x	x	connected to the integrated SPI flash
10	x	x	connected to the integrated SPI flash
11	x	x	connected to the integrated SPI flash
12	OK	OK	boot fail if pulled high
13	OK	OK	
14	OK	OK	outputs PWM signal at boot
15	OK	OK	outputs PWM signal at boot
16	OK	OK	
17	OK	OK	
18	OK	OK	
19	OK	OK	
21	OK	OK	
22	OK	OK	
23	OK	OK	
25	OK	OK	
26	OK	OK	
27	OK	OK	
32	OK	OK	
33	OK	OK	
34	OK		input only
35	OK		input only
36	OK		input only
39	OK		input only

Input only pins

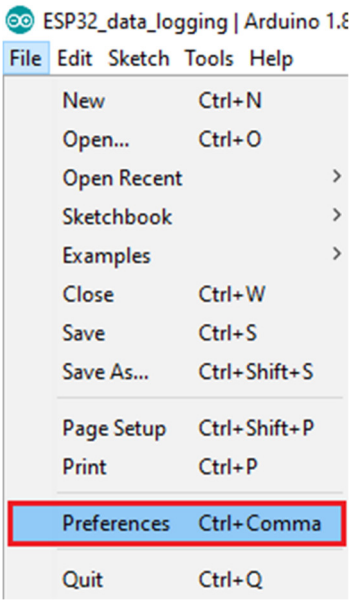
GPIOs 34 to 39 are GPIs – input only pins. **These pins don't have internal pull-ups or pull-down resistors**. They can't be used as outputs, so use these pins only as inputs:

- GPIO 34
- GPIO 35
- GPIO 36
- GPIO 39

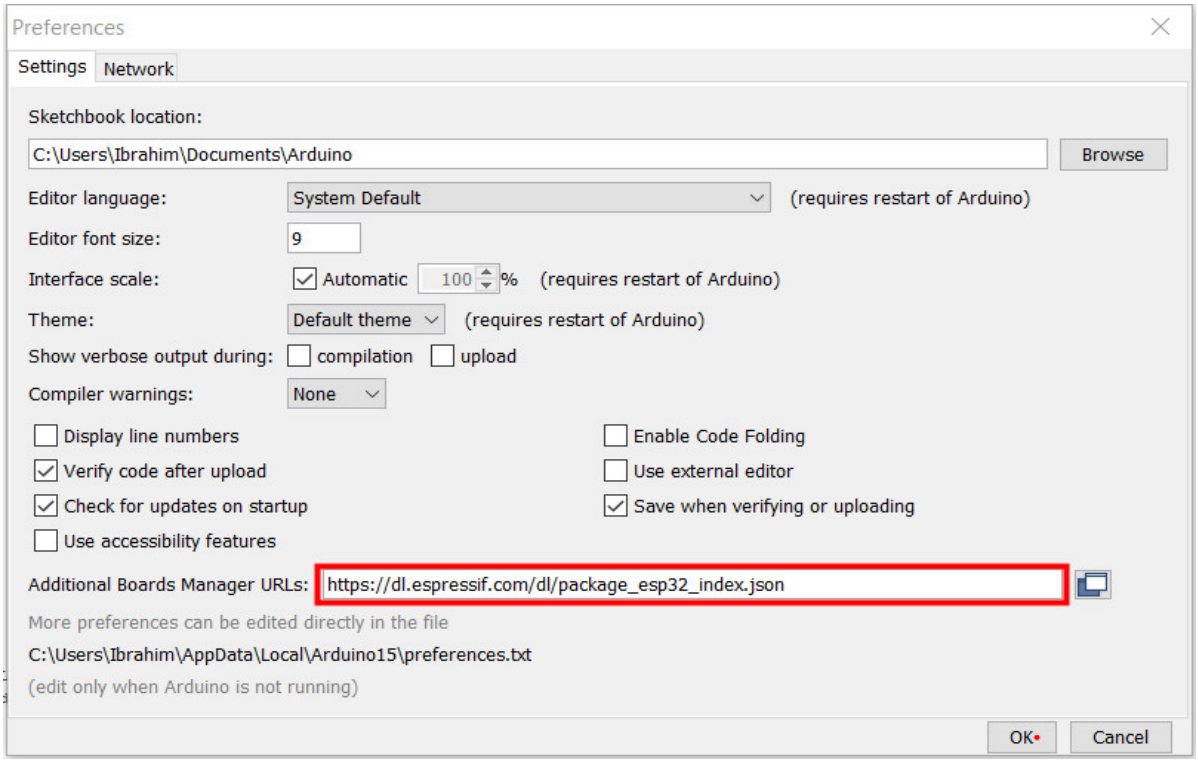
Installing ESP32 Library using Arduino IDE:

To install the ESP32 board in your Arduino IDE, follow these next instructions:

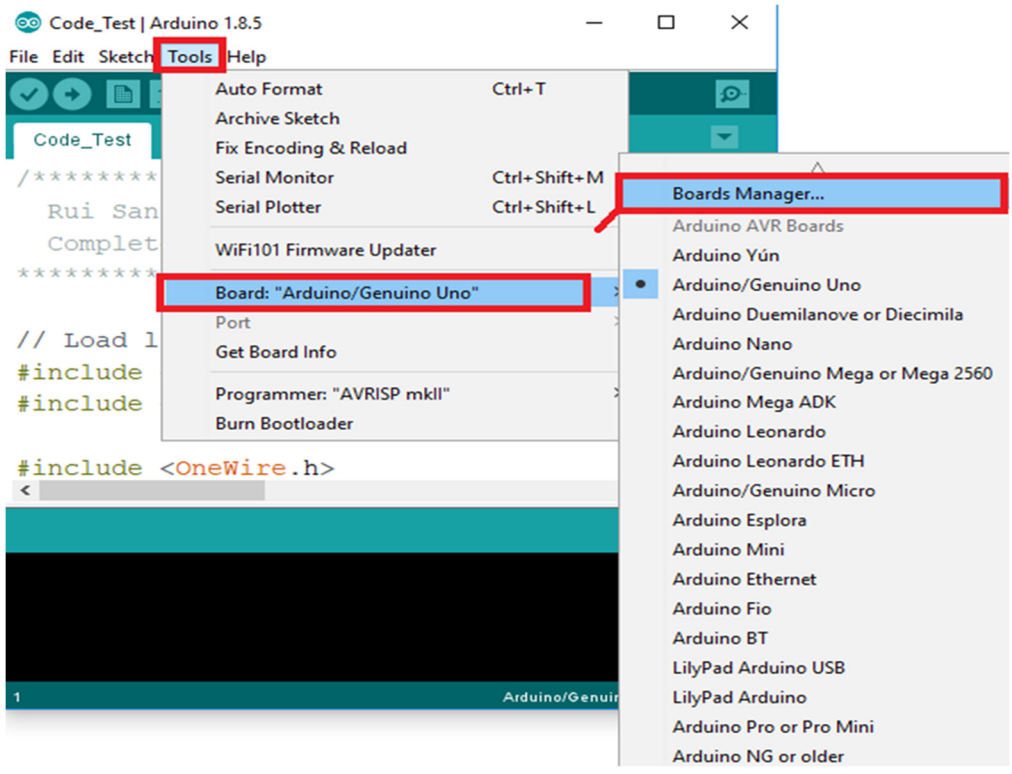
1. In your Arduino IDE, go to **File**> **Preferences**



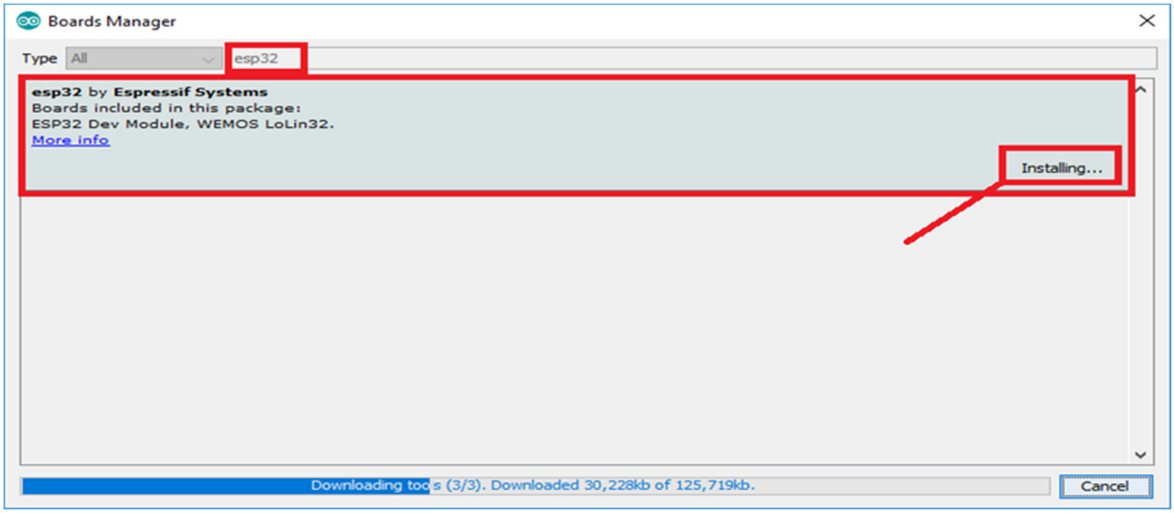
2. Enter **https://dl.espressif.com/dl/package_esp32_index.json** into the "Additional Board Manager URLs" field as shown in the figure below. Then, click the "OK" button:



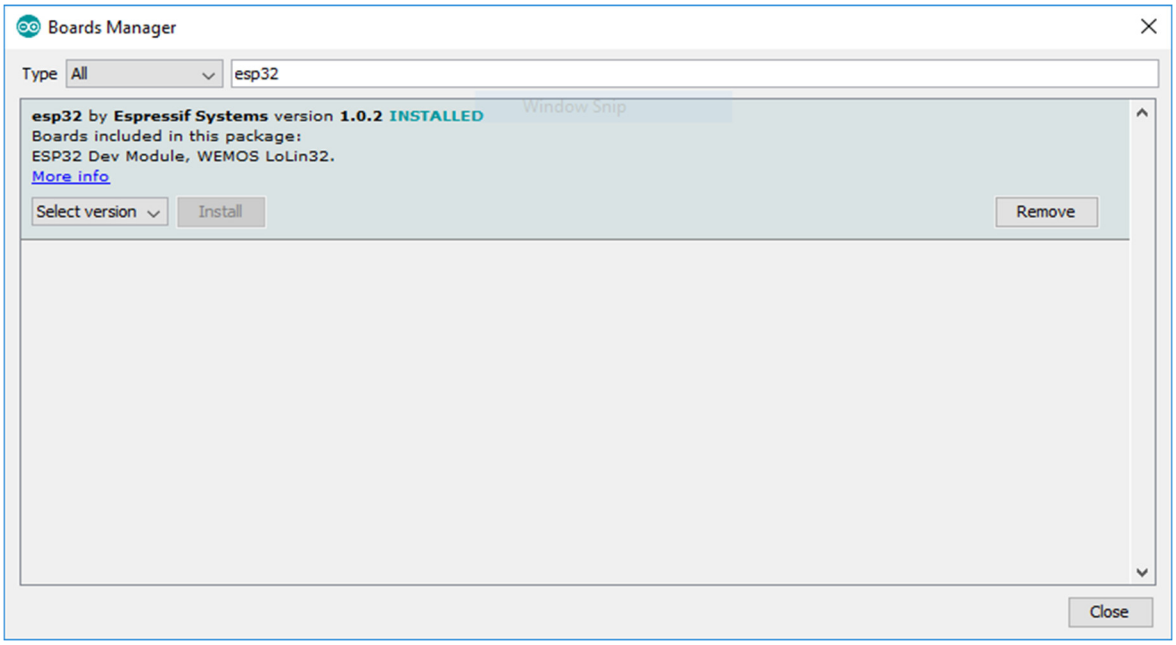
3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**



4. Search for **ESP32** and press install button for the "**ESP32 by Espressif Systems**":



5. That's it. It should be installed after a few seconds.

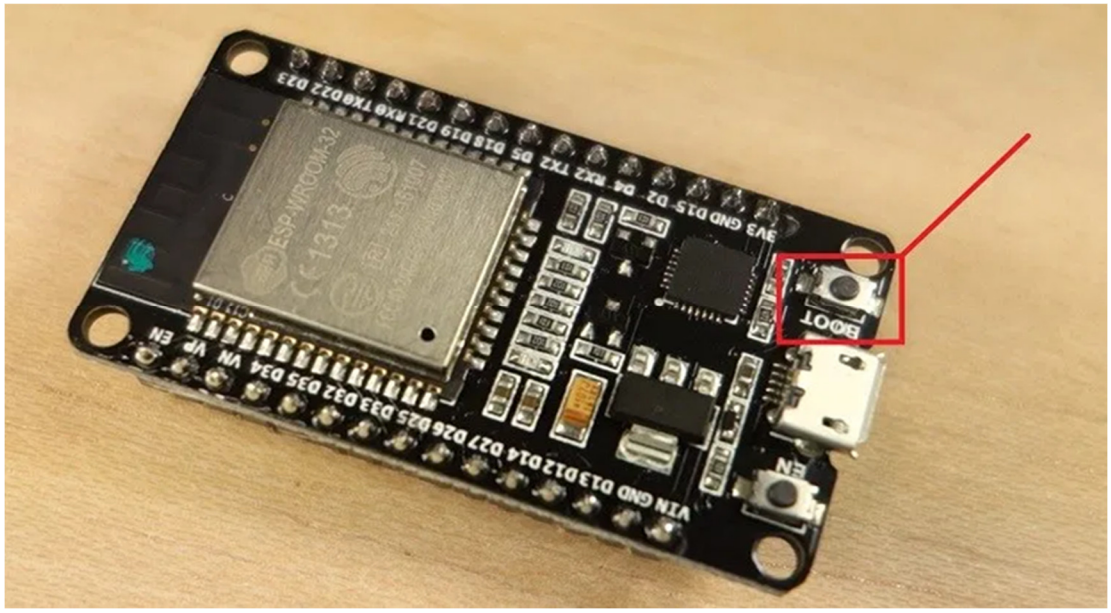


Troubleshooting

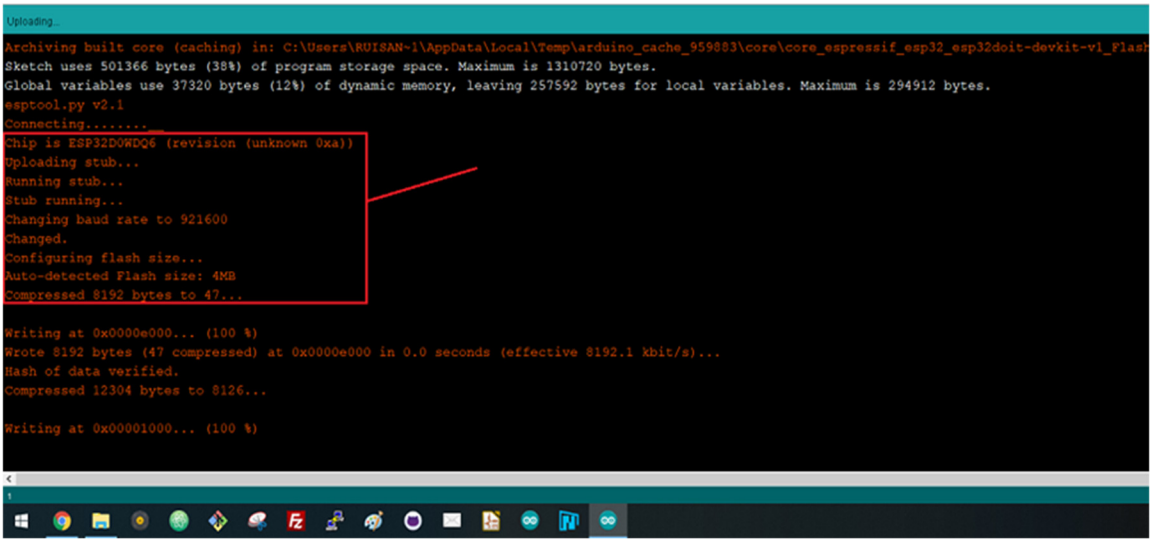
If you try to upload a new sketch to your ESP32 and you get this error message "*A fatal error occurred: Failed to connect to ESP32: Timed out… Connecting…*". It means that your ESP32 is not in flashing/uploading mode. In my case it worked several times without any issue.

Having the right board name and COM port selected, follow these steps:

- Hold-down the "**BOOT**" button in your ESP32 board



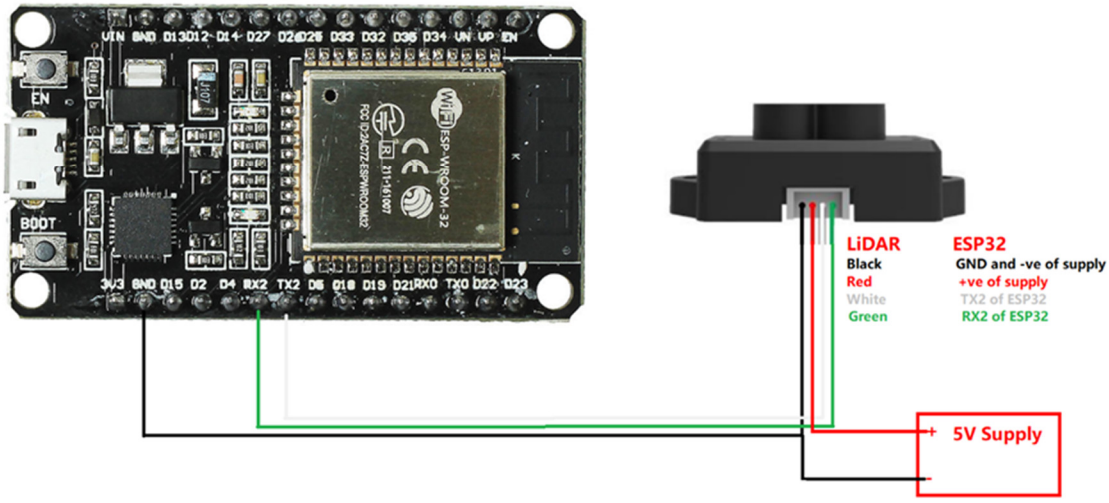
- Press the "**Upload**" button in the Arduino IDE to upload your sketch.
- After you see the "**Connecting….**" message in your Arduino IDE, release the finger from the "**BOOT**" button:



- After that, you should see the "**Done uploading**" message

That's it. Your ESP32 should have the new sketch running. Press the "**ENABLE**" button to restart the ESP32 and run the new uploaded sketch. You'll also have to repeat that button sequence every time you want to upload a new sketch.

Connecting LiDAR with ESP32:



In this tutorial I used TFmini-S as an example but all other TF Series LiDARs can be interfaced in the same way by connecting the right wires because the code part and protocol are same.

NOTE: Here you should notice that I used external power supply of 5V in the diagram, actually 3.3V of ESP32 can be used (I did testing with 3.3V) but using 3.3V will reduce the accuracy of measurement because the rated voltage of TF series LiDAR is 5V. So if you want to get documented accuracy of measurement you need to provide 5V.

ESP32 Code Script:

I used hardware serial port of ESP32. In order to use other pins for serial communication, addition of Software Serial Library will be required. This code is uploaded to GitHub repository[1].

This section defines pins, defining necessary variables, initialization of serial ports that are required in order to process the LiDAR data.

[1] https://github.com/ibrahimqazi/TFmini-S-interfacing-with-ESP32

```
**********************************************************************************
// Note the format for setting a serial port is as follows: Serial2.begin(baud-rate, protocol, RX pin, TX pin);
#define RXD2 16
#define TXD2 17
int dist; /*----actual distance measurements of LiDAR---*/
int strength; /*----signal strength of LiDAR----------------*/
float temprature;
unsigned char check;       /*----save check value-----------------------*/
int i;
unsigned char uart[9];  /*----save data measured by LiDAR-------------*/
const int HEADER=0x59; /*----frame header of data package------------*/
int rec_debug_state = 0x01;//receive state for frame

void setup() {
        delay(2000);
        Serial.begin(115200);
        Serial.println("\nBenewake TFmini-S UART LiDAR Program");
        Serial2.begin(115200);
}

**********************************************************************************
```

Next section includes processing LiDARs Data and printing it to the terminal.

```
void loop() {
        Get_Lidar_data();
}
void Get_Lidar_data(){
if (Serial2.available()) //check if serial port has data input
  {
  if(rec_debug_state == 0x01)
    {  //the first byte
      uart[0]=Serial2.read();
      if(uart[0] == 0x59)
        {
          check = uart[0];
          rec_debug_state = 0x02;
        }
    }
else if(rec_debug_state == 0x02)
   {//the second byte
    uart[1]=Serial2.read();
    if(uart[1] == 0x59)
      {
        check += uart[1];
        rec_debug_state = 0x03;
      }
    else{
        rec_debug_state = 0x01;
      }
    }
```

```
else if(rec_debug_state == 0x03)
      {
        uart[2]=Serial2.read();
        check += uart[2];
        rec_debug_state = 0x04;
      }
else if(rec_debug_state == 0x04)
      {
        uart[3]=Serial2.read();
        check += uart[3];
        rec_debug_state = 0x05;
      }
else if(rec_debug_state == 0x05)
      {
        uart[4]=Serial2.read();
        check += uart[4];
        rec_debug_state = 0x06;
      }
else if(rec_debug_state == 0x06)
      {
        uart[5]=Serial2.read();
        check += uart[5];
        rec_debug_state = 0x07;
      }
```

```
else if(rec_debug_state == 0x07)
      {
        uart[6]=Serial2.read();
        check += uart[6];
        rec_debug_state = 0x08;
      }
else if(rec_debug_state == 0x08)
      {
        uart[7]=Serial2.read();
        check += uart[7];
        rec_debug_state = 0x09;
      }
else if(rec_debug_state == 0x09)
      {
        uart[8]=Serial2.read();
if(uart[8] == check)
         {

           dist = uart[2] + uart[3]*256;//the distance
           strength = uart[4] + uart[5]*256;//the strength
           temprature = uart[6] + uart[7] *256;//calculate chip temprature
           temprature = temprature/8 - 256;
           Serial.print("dist = ");
           Serial.print(dist); //output measure distance value of LiDAR
           Serial.print('\n');
```

```
            Serial.print("strength = ");
            Serial.print(strength); //output signal strength value
            Serial.print('\n');
            Serial.print("\t Chip Temprature = ");
            Serial.print(temprature);
            Serial.println(" celcius degree"); //output chip temperature of Lidar
            while(Serial2.available()){Serial2.read();} // This part is added becuase some previous packets are there in the buffer so to clear serial buffer and get fresh data.
            delay(100);
           }
         rec_debug_state = 0x01;
       }
}
}
```

While burning the code to ESP32, the board needs to be chosen is ESP32 Dev Module. After successfully burning the code, you should be see the data in Serial Monitor of Arduino IDE or any other serial port tool can be used.

The data can be displayed using Serial Monitor of Arduino IDE or any other Serial Port Tool.

COM5

```
dist = 231
strength = 679
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 681
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 685
        Chip Temprature = 40.00 celcius degree
dist = 232
strength = 681
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 676
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 683
        Chip Temprature = 40.00 celcius degree
dist = 232
strength = 685
        Chip Temprature = 40.00 celcius degree
dist = 232
strength = 679
        Chip Temprature = 40.00 celcius degree
dist = 232
strength = 689
        Chip Temprature = 40.00 celcius degree
dist = 232
strength = 683
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 678
        Chip Temprature = 40.00 celcius degree
dist = 232
strength = 680
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 683
        Chip Temprature = 40.00 celcius degree
dist = 231
strength = 676
        Chip Temprature = 40.00 celcius degree
```