Interfacing TF Series Single Point LiDARs with Raspberry Pi Pico

Written by: Ibrahim (FAE)

www.benewake.com
Benewake (Beijing) Co., Ltd.

This article explains how to interface Benewake single point LiDARs with Raspberry Pi Pico microcontroller. The following assumptions are made while writing this article:

1. The end-user has Pico microcontroller and any of the single point LiDARs (Luna, TFmini-Plus, mini-S, TF02-pro, TF03-UART) in hands
2. The development environment (including Pico SDK) for compiling the C code has already been set
3. All the necessary cables and 5V power supply
4. The LiDAR interface mode is TTL
5. Any editor for writing and modifying the script (VS Code is recommended)

Pin configuration of Pico Controller:

Full documentation about Pico is available on Raspberry Pi website, but here we will discuss briefly the pinouts for easiness and we will see which pins can be used for UART interfacing. The following image shows the pinout of Pico (to see the details zoom in the picture):

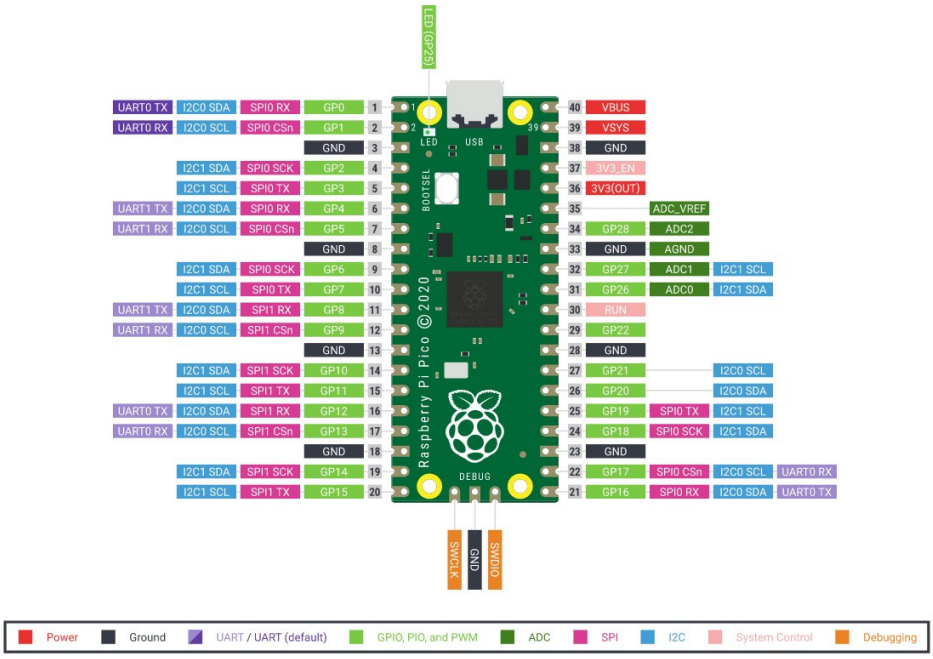

Figure 1: Pinouts

According to the datasheet of Pico, it has two UART ports and these ports can be directed to any of the available pins. Looking at the above image (GP0, GP1), (GP4, GP5), (GP8, GP9), (GP12, GP13), and (GP16, GP17) are suitable for UART communication. In our code we used GPIO-4 and 5. It should be noted that there is only single pin for voltage supply and that is also 3.3V, while the rated voltage for Benewake LiDAR is 5V. If the voltage is not 5V, the accuracy will be affected and error will be more than rated error, in some cases the error is non-linear and it increases with increase in distance. It may reach up to 40cm. So it is advised to use separate 5V supply for LiDAR.

Schematic diagram:

The connection diagram for interfacing LiDAR to Pico and using a separate power supply is given below:

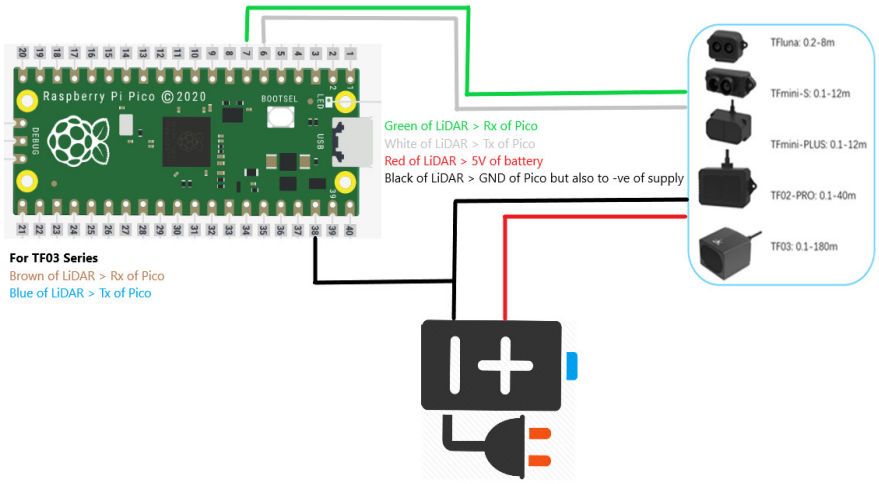

Figure 2: Connection Diagram

For exact details about LiDAR current rating and your supply capacity, please refer to their respective data-sheets.

Default UART Communication Parameters of TF Series:

According to the data-sheet of TF series LiDARs the parameters of UART:

Item	Content
Communication protocol	UART
Baud rate	115200
Data bit	8
Stop bit	1
Checksum bit	None

Figure 3: UART Communication

So the UART communication in code script is established based on this information.

Cable Selection and Connector:

All Benewake Single Points LiDARs have either 4-pin or 7pin (for TF03) Molex connector, so the required mating connector is JST 1.25mm. Please refer to the link[1]. You can choose either 4-pin or 7-pin depending upon the LiDAR that you have. On the controller side if you are using breadboard then you can choose male-DuPont connector or any other suitable connector based on user-end side. One of the sample cable is shown below:

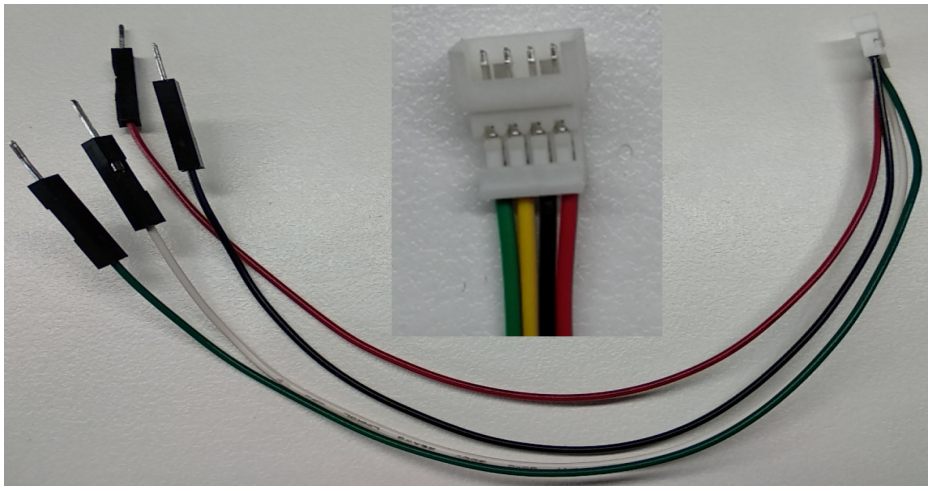

Figure 4: JST 1.25 4P to DuPont cable

Table III: Data format (for mid and small range LiDARs)

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 low 8 bits						
Byte3	Dist_H distance value high 8 bits						
Byte4	Strength_L low 8 bits						
Byte5	Strength_H high 8 bits						
Byte6	Temp_L low 8 bits						
Byte7	Temp_H high 8 bits						
Byte8	Checksum is the lower 8 bits of the cumulative sum of the numbers of the first 8 bytes.						

[1] https://www.aliexpress.com/i/4000580232835.html

Table IV: Data format (TF03 Series)

Data bit	Definition	Description
Byte0	Frame header	0x59
Byte1	Frame header	0x59
Byte2	DIST_L	DIST low 8-bits
Byte3	DIST_H	DIST high 8-bits
Byte4	Strength_L	Signal strength low 8-bits
Byte5	Strength_H	Signal strength high 8-bits
Byte6	Reserved bit	/
Byte7	Reserved bit	/

C code for Pico Controller:

In this section we will discuss the main parts of C code that are used for reading the data from Benewake LiDAR. Adding necessary libraries:

```
#include<stdio.h>
#include "pico/stdlib.h"
#include "hardware/gpio.h"
#include "pico/binary_info.h"
#include "hardware/uart.h"
#include "tusb.h" // this header file will handle the problem of losing initial o
utput
```

Pin initialization for UART port:

Like mentioned above there are two serial ports (port-0 and port-1). I used port-1.

```
#define BAUD_RATE 115200
#define UART_ID1 uart1
#define UART1_TX_PIN 4 // pin-6
#define UART1_RX_PIN 5 // pin-7
```

I used structures of C language to handle LiDAR data.

```
//**************************Structure and Union for handling LiDAR Data********
***

//Dist_L Dist_H Strength_L Strength_H Temp_L Temp_H Checksum
typedef struct{
unsigned short Header;
```

```
unsigned short Dist;
unsigned short Strength;
}structLidar;

union unionLidar{
unsigned char Byte[9];
structLidar lidar;
};

unsigned char lidarCounter=0;
union unionLidar Lidar;
//*************************Structure and Union for handling LiDAR Data********
***
```

Function for processing UART communication:

```
//*************************Function to read serial data***********
int isLidar(uart_inst_t * uart, union unionLidar * lidar)
{
  int loop;
  int checksum;
  unsigned char serialChar;

  while(uart_is_readable(uart))
  {
   if(lidarCounter > 8)
    {
      lidarCounter=0;
      return 0; // something wrong
    }

   serialChar = uart_getc(uart); // Read a single character to UART.
   lidar->Byte[lidarCounter]= serialChar;

   switch(lidarCounter++)
   {
    case 0:
    case 1:
         if(serialChar !=0x59)
              lidarCounter=0;
           break;
    case 8: // checksum
            checksum=0;
```

```
            lidarCounter=0;
            for(loop=0;loop<8;loop++)
                checksum+= lidar->Byte[loop];
            if((checksum &0xff) == serialChar)
                {
                  //printf("checksum ok\n");
                  lidar->lidar.Dist = lidar->Byte[2] | lidar->Byte[3] << 8;
                  lidar->lidar.Strength = lidar->Byte[4] | lidar->Byte[5] << 8;
                  return 1;
                }
      //printf("bad checksum %02x != %02x\n",checksum & 0xff, serialChar);
    }
   }
    return 0;
}
//**************************Function to read serial data***********
```

In the main function I run while loop which will continuously monitor the data flow on serial port and print it to the screen. The LED of Pico which is connected to Pin-25, will continuously blinking as data is processed. There are some binary info related code which I didn't explain, that is only used for debugging purpose and is not very necessary.

```
//*********************************************************
cdcd_init();
printf("waiting for usb host");
while (!tud_cdc_connected()) {
  printf(".");
  sleep_ms(500);
}
printf("\nusb host detected!\n");
//*********************************************************
```

The above part of code will wait for USB host to get connected, once it is connected, the code flow will move to the next step. While the following code checks if UART port is enabled:

```
//*********************************************************
// In a default system, printf will also output via the default UART
sleep_ms(5000);
ret = uart_is_enabled (uart1);
    if(ret == true){
        printf("UART-1 is enabled\n");
    }
  printf("Ready to read data from Benewake LiDAR\n");
```

While Loop:

```
while(true){
  gpio_put(LED_PIN, 0);
  sleep_ms(100);
  gpio_put(LED_PIN, 1);
 if(isLidar(UART_ID1,&Lidar))
     {
      // ok we got valid data
      // Here we utilized the Union
      printf("Dist:%u Strength:%u \n",\
             Lidar.lidar.Dist,\
             Lidar.lidar.Strength);
     }
 }
```

CMakeLists.txt:

add_executable(tf_series tf_series.c)

Pull in our pico_stdlib which pulls in commonly used features, also add hardware uart because we are going to use uart port

target_link_libraries(tf_series pico_stdlib hardware_uart)

enable/diable usb output, and uart output

pico_enable_stdio_usb(tf_series 1) # 1 means enable and 0 means disable

pico_enable_stdio_uart(tf_series 1)

create map/bin/hex file etc.

pico_add_extra_outputs(tf_series)

add url via pico_set_program_url

example_auto_set_url(tf_series)

You can either get the code from Technical Support of Benewake or download from GitHub Link[2].

[2] https://github.com/ibrahimqazi/Benewake-TF-Series-LiDAR-Interfacing-with-Pico