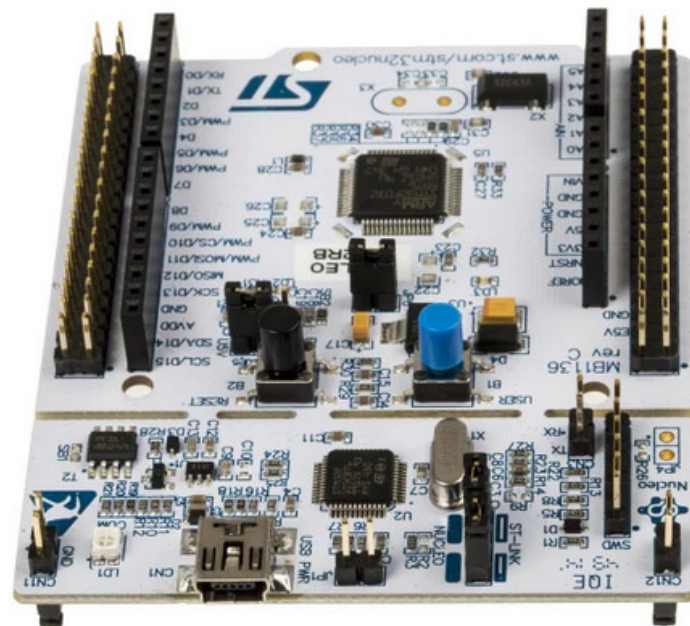


Interfacing TF Series Single Point LiDARs with Nucleo-64 STM32F411 Development Board



Written by: Ibrahim (FAE)

This tutorial explains how to interface Benewake single point LiDARs with Nucleo-64 development boards carrying STM32F411RE series microcontrollers. The following assumptions are made while writing this tutorial:

1. The customer has Nucleo-F411RE development board and any of the single point LiDARs (Luna, TFmini-Plus, mini-S, TF02-pro, TF03-UART) in hands
2. Have basic understanding of development environment (Keil-5, CubeMX) for compiling C code, and initial configurations of board and environment has already been set
3. All the necessary cables (USB cable, JST Socket (with 1.25mm pitch) to male dupont connector etc.) and 5V power supply is available
4. The LiDAR interface mode is TTL/UART
5. For code editing you could use CodeBlocks, VS Code or Keil but for compilation you need to use Keil, as it has everything set up

Pin configuration of Nucleo-64:

For full documentation about Nucleo-F411RE you can refer to ST website, but here we will discuss briefly the pinouts for convenience and we will see which pins can be used for UART interfacing. The following image shows the pin details of Nucleo-F411RE:

CN7 odd pins		CN7 even pins		CN10 odd pins		CN10 even pins	
Pin No.	Name	Name	Pin No.	Pin No.	Name	Name	Pin No.
1	PC10	PC11	2	1	PC9	PC8	2
3	PC12	PD2	4	3	PB8	PC6	4
5	VDD	E5V	6	5	PB9	PC5	6
7	BOOT0 ⁽¹⁾	GND	8	7	AVDD	U5V ⁽²⁾	8
9	-	-	10	9	GND	-	10
11	-	IOREF	12	11	PA5	PA12	12
13	PA13 ⁽³⁾	RESET	14	13	PA6	PA11	14
15	PA14 ⁽³⁾	+3V3	16	15	PA7	PB12	16
17	PA15	+5V	18	17	PB6	-	18
19	GND	GND	20	19	PC7	GND	20
21	PB7	GND	22	21	PA9	PB2	22
23	PC13	VIN	24	23	PA8	PB1	24
25	PC14	-	26	25	PB10	PB15	26
27	PC15	PA0	28	27	PB4	PB14	28
29	PH0	PA1	30	29	PB5	PB13	30
31	PH1	PA4	32	31	PB3	AGND	32
33	VBAT	PB0	34	33	PA10	PC4	34
35	PC2	PC1 or PB9 ⁽⁴⁾	36	35	PA2	-	36
37	PC3	PC0 or PB8 ⁽⁴⁾	38	37	PA3	-	38

Figure 1: Pin numbering

This development board has three USART interfaces. USART-2 is connected to ST-Link debugger for virtual com port which can be used to display the data on computer. Although it's possible use this UART port for connecting peripherals or TTL-USB adapter but that's beyond the scope of this tutorial. We will use only USART-1 and USART6 for connecting Benewake LiDAR. As you can see in above image that pins marked in red are connected to debugger, while pins marked in black are free and can be used to connect peripherals. The pin out-view taken from CubeMX is shown below, which are pin configuration:

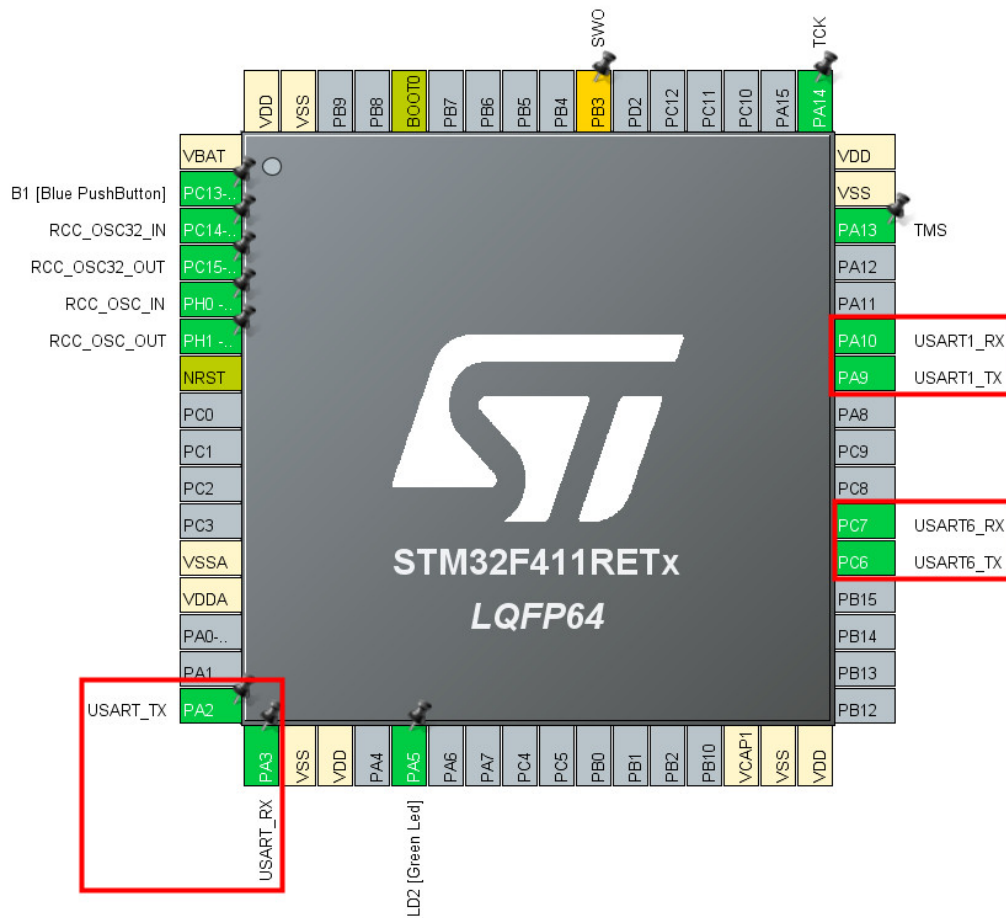


Figure 2: Pinout View

It should be noted that LiDAR needs 5V voltage for stable operation so connect LiDAR to any 5V output on Nucleo-64 board. If the voltage is not 5V, the accuracy will be affected and error will be more than rated accuracy, in some cases the error is non-linear and it increases with increase in distance. It may reach up to 40cm or even more.

Schematic diagram:

The connection diagram for interfacing two LiDARs to Nucleo board and using a separate power supply is shown below:

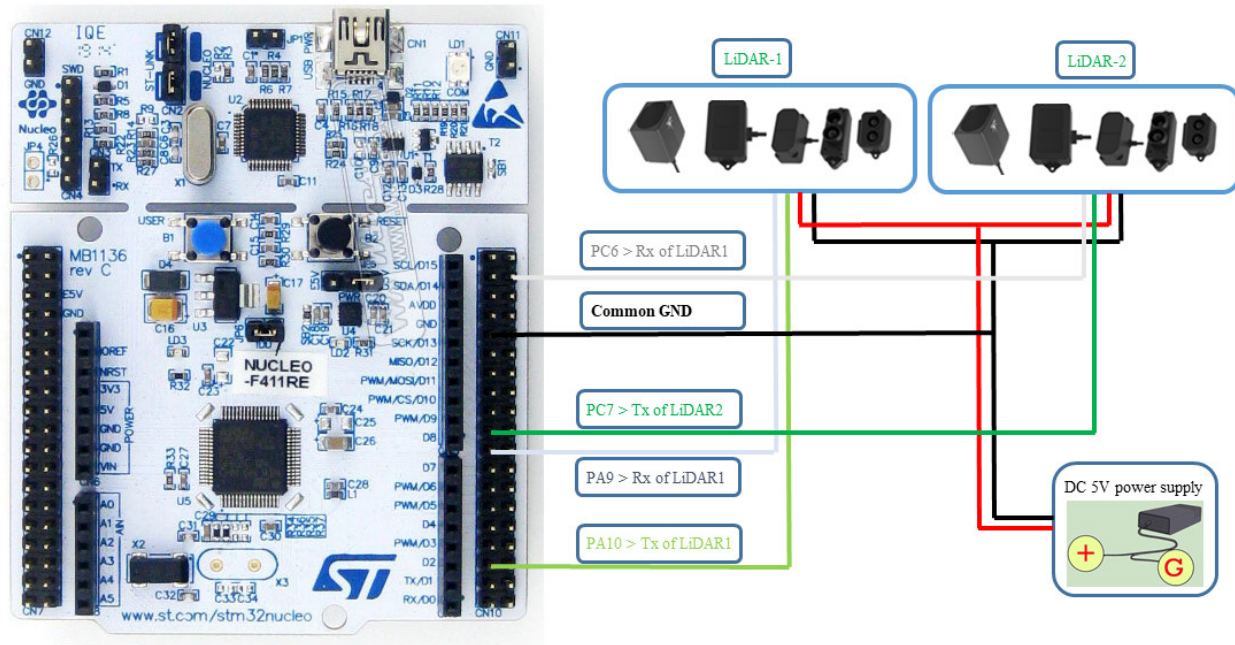


Figure 3: Connection Diagram

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 communication:

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

Figure 4: UART Communication

So the UART ports are configured in CubeMX and initial code is generated based on this information.

Cable Selection and Connector:

All Benewake Single Points LiDARs have either 4-pin or 7pin (for TF03) Molex connector or JST with 1.25mm pitch, so the required mating connector is JST socket with 1.25mm. Please refer to the link¹. In case if you are designing your own PCB so you can choose either 4-pin or 7-pin depending upon the LiDAR that you have. On the controller you may need Dupont male connector or female connector, I used female Dupont connector or you can choose any suitable connector based on your design. One of the sample cable is shown below:

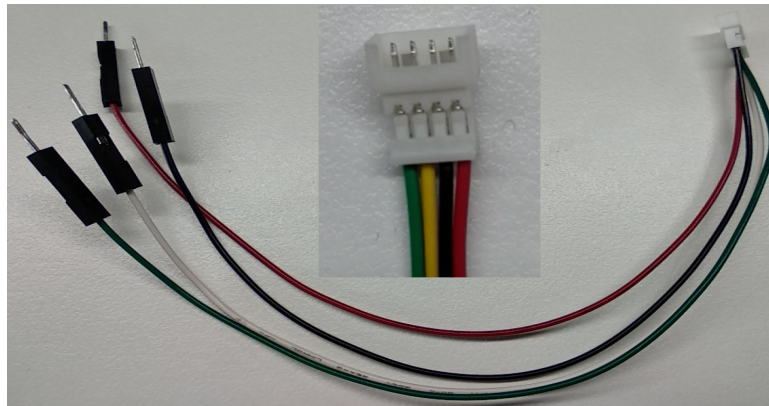


Figure 4: JST 1.25 4P to DuPont male cable

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	/

CubeMX related settings:

Following are the settings for all USART ports.

Mode

Mode

Asynchronous

Hardware Flow Control (RS232)

Disable

Configuration

Reset Configuration

Parameter Settings

User Constants

NVIC Settings

DMA Settings

GPIO Settings

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Baud Rate

115200 Bits/s

Word Length

8 Bits (including Parity)

Parity

None

Stop Bits

1

Advanced Parameters

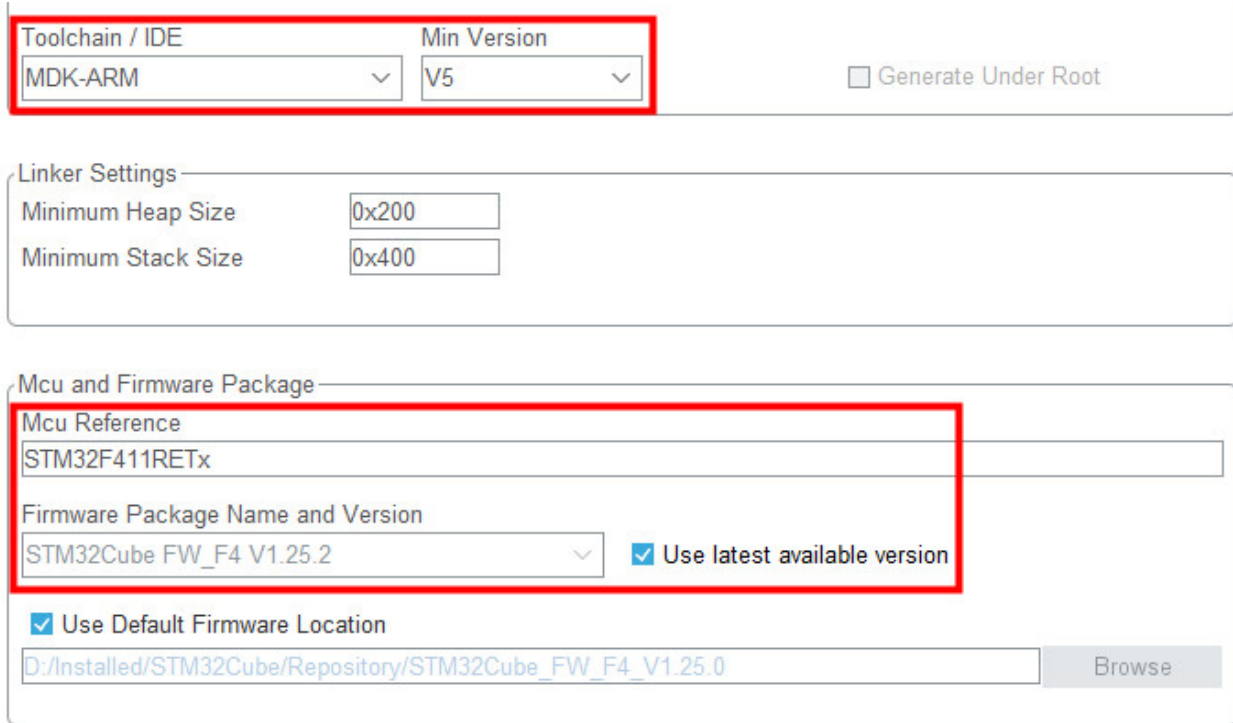
Data Direction

Receive and Transmit

Over Sampling

16 Samples

Toolchain and firmware settings:



The screenshot shows the STM32CubeMX configuration interface. The 'Toolchain / IDE' section is highlighted with a red box, showing 'MDK-ARM' selected for the IDE and 'V5' for the Min Version. Below this, the 'Linker Settings' section shows 'Minimum Heap Size' as '0x200' and 'Minimum Stack Size' as '0x400'. The 'Mcu and Firmware Package' section is also highlighted with a red box, showing 'STM32F411RETx' for the Mcu Reference, 'STM32Cube FW_F4 V1.25.2' for the Firmware Package Name and Version, and 'Use latest available version' checked. Below this, 'Use Default Firmware Location' is checked, and the path 'D:/Installed/STM32Cube/Repository/STM32Cube_FW_F4_V1.25.0' is shown with a 'Browse' button.

C code for Nucleo Board:

After you generate the initial code using CubeMX, we will need to modify two files **main.c** and **main.h** to add our code for LiDAR interfacing and reading the data from sensors and then print it to virtual com port. So in this section we will discuss the main parts of C code that are necessary for communicating with Benewake LiDARs. Adding necessary libraries:

```
/* USER CODE BEGIN Includes */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
/* USER CODE END Includes */
```

In the while loop of main function we will call two functions to read from sensor:

```
while (1)
{
    /* USER CODE END WHILE */
```

```
/* USER CODE BEGIN 3 */
    read_data_uart1();
    read_data_uart6();
}
```

These two functions are defined in main.c declared in main.h:

```
/* USER CODE BEGIN EFP */
void read_data_uart1(void);
void read_data_uart6(void);
/* USER CODE END EFP */
```

```
/**Function for reading data from Sensor-1***/
void read_data_uart1(){
    memset( usart_rx_buf, 0, sizeof( usart_rx_buf)); // Clear the data buffer.
    HAL_UART_Receive(&huart1, usart_rx_buf, 9, HAL_MAX_DELAY);
    uint16_t dist, strength=0;
    uint8_t checksum, chip_temp=0;
    if (usart_rx_buf[0]==HEADER && usart_rx_buf[1]==HEADER){
        for (int i=0;i<8;i++){
            checksum += usart_rx_buf[i];
        }
        if (usart_rx_buf[8] == (checksum&0xff)){
            dist = usart_rx_buf[2] | (usart_rx_buf[3]<<8);
            strength = usart_rx_buf[4] | (usart_rx_buf[5]<<8);
            chip_temp = usart_rx_buf[6] | (usart_rx_buf[7]<<8);
            chip_temp = chip_temp/8 - 256;
            //printf("Distance: %d cm \t", dist);
            //printf("Strength: %d \t", strength);
            //printf("Chip temp: %d degree celcius\t\n", chip_temp);
            printf("LiDAR-1: dist = %d cm; strength = %d; temp = %d.\r\n",
dist,strength,chip_temp); // Print everything in single line.
            //HAL_UART_Transmit(&huart2,usart_rx_buf,sizeof(usart_rx_buf),
HAL_MAX_DELAY); // Print full Datapacket in HEX format.

            for(int j=0; j<1000; j++){}; // This will create a short delay.
            memset( usart_rx_buf, 0, sizeof( usart_rx_buf)); // Clear the data
buffer.
        }
    }
}
```

```

/*****Function for reading data from Sensor-1*****/

/*****Function for reading data from Sensor-2*****/
void read_data_uart6(){
    memset( usart_rx_buf, 0, sizeof( usart_rx_buf)); // Clear the data buffer.
    HAL_UART_Receive(&huart6, usart_rx_buf, 9, HAL_MAX_DELAY);
    uint16_t dist, strength=0;
    uint8_t checksum, chip_temp=0;
    if (usart_rx_buf[0]==HEADER && usart_rx_buf[1]==HEADER){
        for (int i=0;i<8;i++){
            checksum += usart_rx_buf[i];
        }
        if (usart_rx_buf[8] == (checksum&0xff)){
            dist = usart_rx_buf[2] | (usart_rx_buf[3]<<8);
            strength = usart_rx_buf[4] | (usart_rx_buf[5]<<8);
            chip_temp = usart_rx_buf[6] | (usart_rx_buf[7]<<8);
            chip_temp = chip_temp/8 - 256;
            //printf("Distance: %d cm \t", dist);
            //printf("Strength: %d \t", strength);
            //printf("Chip temp: %d degree celcius\t\n", chip_temp);
            printf("LiDAR-2: dist = %d cm; strength = %d; temp = %d.\r\n",
dist,strength,chip_temp); // Print everything in single line.
            //HAL_UART_Transmit(&huart6,usart_rx_buf,sizeof(usart_rx_buf),
HAL_MAX_DELAY); // Print full DataPacket in HEX format.

            for(int j=0; j<1000; j++){}; // This will create a short delay.
            memset( usart_rx_buf, 0, sizeof( usart_rx_buf)); // Clear the data
buffer.
        }
    }
}

/*****Function for reading data from Sensor-2*****/

```

In order to print data to the virtual com port which is connected to debugger we will need to add the following part of code:

```

/*****Function for printing data to virtual Com port*****/
#ifdef __GNUC__
    /* With GCC/RAISONANCE, small printf (option LD Linker->Libraries->Small printf
    set to 'Yes') calls __io_putchar() */
    #define PUTCHAR_PROTOTYPE int __io_putchar(int ch)

```

```
#else
#define PUTCHAR_PROTOTYPE int fputc(int ch, FILE *f)
#endif /* __GNUC__ */
/**
 * @brief Retargets the C library printf function to the USART.
 * @param None
 * @retval None
 */
PUTCHAR_PROTOTYPE
{
    /* Place your implementation of fputc here */
    /* e.g. write a character to the EVAL_COM1 and Loop until the end of
    transmission */
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, HAL_MAX_DELAY); // If you want to
    direct printf to some other ports let say uart1 or 6 then change the first param.

    return ch;
}
//*****Function for printing data to virtual Com port*****
```

In the main function “while loop” runs continuously to monitor the data flow on serial ports and print it to the serial port tool through virtual com port. If your settings go successful and the code is compiled without any issue then you should be able to see the data being printed to serial port tool.

You can either get the code from Technical Support of Benewake or download from GitHub Link².

² <https://github.com/ibrahimgazi/TF-Series-LiDAR-Interfacing-with-Nucleo-F411RE-using-UART-Interface>