CS301: STM32 Communication with BlueTooth Module

Chutong Meng, Long Chen, Zhiyue Wang, Yunhao Luo
Group 8

December 2020

1 Introduction

In CS301, we learn about *Single Chip Microcomputer* (MCU) both practically and theoretically. In the experimental part, our team have successfully implement BlueTooth Communication Functionality. Our Arm Board model is *STM32F103RCTx* and we mainly use *STM32Cube IDE* in the development processing.

2 User Interface

2.1 UI Design

Since *STM32F103RCTx* has a useful LCD screen support, we mainly use it to display messages, including the concurrent configuration of our Arm Board and communication messages between two devices. According to the given requirement, we split the screen into two parts.

The upper $\frac{1}{4}$ of the screen contains information about the name of our BlueTooth Module and the BlueTooth connection status. To be clean and tidy, we design an icon to represent the BlueTooth status, shown in...

The lower $\frac{3}{4}$, as a result, is the communication messages block. We strictly follow the given requirement: the messages sent are right aligned and those received messages are left-justified following the design of *QQ* and *WeChat*. In addition, to clearly classify the message sequence, we add a horizontal

line between every messages and sent texts are colored in Blue, while texts received are colored in Red. More details are shown in fig

2.2 UI Implementation

We held several discussion before we started and come up an succinct design. We preserve two 1D-arrays: one to store the basic information of BlueTooth and the other is used to keep the icon. Moreover, a 2D-array is also Introduced in order to hold messages from both sides. A tricky problem is how to deal with the correct windows shift when the messages overflow, i.e., not enough places to show all data. In this case, we need to ensure that messages as well as our divided lines are up-shifted properly.

UART Callback Interrupt

Our interrupt handler *HAL_UART_RxCpltCallback* is defined in alg:1. In this function, whenever the board received a message, we first set a conditional branch depends on its size. If the messages is too long for one line, namely 13 chars, we will divide it into several lines and store them into corresponding arrays respectively. If there are to many lines that the screen cannot hold (the small screen can only display 12 rows in whole), we will first calculate the number required for this message and transfer each array forward with the calculated line numbers. In our design, those earlier messages are discarded, as we don't need to access them thereafter. At the same time, we update array *snum*, which stores information about the location of divided lines, thus those divided lines can be correctly shifted as well. Finally, the interrupt ends and go back to the LCD display part.

algorithm 1 HAL_UART_RxCpltCallback

Input: UART_HandleTypeDef, $*huart$;

Output: *void*

```
1: if sender is computer then
2:     divide the data into several packets with length 13 chars
3:     if the message is not out of the display then
4:         add 15 '' into each array of the LCD display for these data
5:         load data in each packet into the array for the LCD display
6:     else
7:         for each LCD array do
8:             load and forward this array to store the message
9:         end for
10:        for last few LCD arrays do
11:            load the corresponding packet for the data.
12:        end for
13:        change the position of the message dividing lines
14:    end if
15: end if
16: if sender is BlueTooth then
17:    divide the data into several packets with length 13 chars
18:    if the message is not out of the display then
19:        load data in each packet into the array for the LCD display
20:    else
21:        for each LCD array do
22:            load and forward this array to store the message
23:        end for
24:        for the last few LCD arrays do
25:            load the corresponding packet for the data.
26:        end for
27:        change the position of the message dividing lines
28:    end if
29: end if
```

LCD Display

Our LCD display modules mainly communicates with BlueTooth modules and UART interrupt function. For messages sent from localhost, we append 15 spaces in front of the actual message to make the text right-aligned. If the message is from BlueTooth module, nothing else is added. In this scenario, we just need to check the first 15 chars to decide which color to use to print out this message. At last, we redraw divided lines by the provided data in array *snum*. Our pseudo code is shown in alg: 2.

algorithm 2 LCD Display

Input: *input_text*;

Output: *void*

```
1: for each LCD Array and corresponding position do
2:     if the first 15 chars of input_text are ‘’: then
3:         point_color = BLUE
4:     else
5:         point_color = RED
6:     end if
7:     LCD_showString(array, position)
8: end for
9: point_color=GRAY
10: for each message divide line position do
11:     LCD_DrawLine(position)
12: end for
```

2.3 Difficulties and Solution in UI

- Data from BlueTooth cannot be display on LCD

  Solution: By repetitive testing and debugging, We learned that the message from BlueTooth might contains some unrecognized encoding, thereby causing the LCD display libraries malfunctioning. Then, we find out that such specially characters has length zero in arrays. By

virtue of that, we explicitly remove those zero-length arrays after the board received messages from BlueTooth and this issue finally resolved.

- Different Colors appear in the same line

  Solution: Before writing new data, it is better to clear one line first. Also, we exchange the execution order of reading and writing, this problem are correctly settled.

3 BlueTooth Module

3.1 BlueTooth Design

The goal of this project is that the user can configure the Bluetooth and communicate with another user using another Bluetooth through Serial Port Utility. For STM32F103RCTx, it is universal synchronous asynchronous receiver transmitter (USART) that offer the ability to transmit and receive data using two pins, TX and RX, on the board. Since we cannot send data directly to the Bluetooth module, it is necessary to use two USARTs to build up the communication between the user and Bluetooth. Thus, we design to have one USART1 to receive data from user, and then transmit it to Bluetooth. And another USART2 to receive data from Bluetooth and send it to user.

The Bluetooth module we use is HC05. It has five pins: STATE, EN, +5V, GND, RX and TX. STATE is to indicate the state of the module. The state can be close, AT Mode, wait for pair and paired. EN is to power on or power off the module. +5V and GND are for power and ground. RX and TX are for transmit and receive data. So, we utilize the EN, STATE and RX, TX to implement the functions. We control the voltage to EN to let the module on and off. We read the pin STATE to know what state the module is in. We use RX and TX to communicate with the user.

Another requirement is that we should configure the Bluetooth. We achieve this by pressing the button on the module. Then the module enters AT mode, and we can enter any AT code to configure it.

To use the Keys on the board to control the Bluetooth, we combine the GPIO EXTI interrupt with AT code.

3.2 LED Design

There are two LEDs on *STM32F103RCTx*, one is green and the other is red.

We make the red LED to indicate the state. When is does not shine, it means the Bluetooth module is not working. When the red LED shines quickly, at a frequency of two times per second, it shows that the Bluetooth module is trying to pair. When the LED keeps shining, it means the Bluetooth has paired successfully.

The green LED is to indicate the receiving of data. If data is received, the green LED will be lightened for one second, and then put out.

3.3 Implementation

The communication between user and Bluetooth module is through USART. More specifically, it can be dealt in the interrupt.

We assign USART1 for user and USART2 for Bluetooth module. In the interrupt handler *HAL_UART_RxCpltCallback*, we should judge which USART invokes the handler. If USART1 invokes it, then it is user that sends data, we should transmit the data received to Bluetooth module. If USART2 invokes it, it is Bluetooth that sends data, we should transmit it to user.

algorithm 3 HAL_UART_RxCpltCallback

Input: UART_HandleTypeDef $*huart$

Output: *void*

```
1: if sender is computer then
2:     send the data to Bluetooth
3: end if
4: if sender is Bluetooth  then
5:     send the data to Computer
6: end if
```

As for LED, we keep reading the pin STATE on Bluetooth and also detect whether the interrupt handler is invoked. To keep reading the value, we use the TIMER to constantly check the pin. Also, in the while loop of main, we keep detecting the USART handler.

algorithm 4 TIME3_IRQHandler

Input: *void*

Output: *void*

```
1: if EN is low then
2:     set LED0 pin to low
3: else
4:     if STATE is low then
5:         toggle LED0
6:     end if
7:     if STATE is high then
8:         set LED0 pin to high
9:     end if
10: end if
```

algorithm 5 main

Input: *void*

Output: *int*

```
1: while true do
2:     if USART was invoked then
3:         light up LED1 for one second
4:     end if
5: end while
```

In order to build up and cancel default connections using key, we send AT code in the GPIO EXTI interrupt. Also, we can open and close the module by setting EN pin in the interrupt.

3.4 Difficulties and Solution

- We got a manual which did not match the Bluetooth module we had. In the manual, it is KEY and LED pin, instead of EN and STATE pin. So, we could not follow the instruction.

algorithm 6 HAL_GPIO_EXTI_Callback

Input: $GPIO_Pin, int$

Output: *void*

```
1: if GPIO_Pin is Enable_Pin then
2:     toggle EN_Pin
3: end if
4: if  GPIO_Pin is KEY0_Pin then
5:     send AT + CMODE = 1 and AT+BIND to Bluetooth
6: end if
7: if GPIO_Pin is KEY1_Pin then
8:     send AT + COMDE = 0 to Bluetooth
9: end if
```

Solution: We searched for other instructions and follow them. It turned out that the usage was really different. We could not control EN pin to make the module enter AT mode. We could only press the button to achieve it.

- We tried to use *HAL_Delay()* in the interrupt to make the LED light up for some time. However, the code often crashed.

  Solution: By searching for solutions, we understood that we cannot use *HAL_Delay()* in interrupt. Then, we use TIMER to do the lighting.

4 Using Introduction

4.1 Line Connection

(1) Connect STATE pin on Bluetooth module with PC4 pin on the development board.

(2) Connect EN pin on Bluetooth module with PC12 pin on the development board.

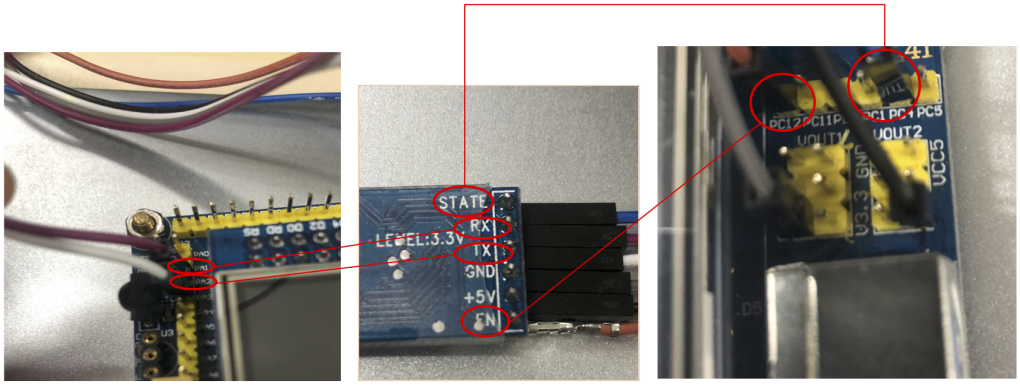

Figure 1: Line Connection Instruction

(3) Connect RX pin on Bluetooth module with PA1 pin on the development board.

(4) Connect TX pin on Bluetooth module with PA2 pin on the development board.

(5) Connect power supply pin on Bluetooth module with any corresponding pin on the development board.

(6) Connect the serial port on the development board with computer.

4.2 Operating Procedure

Connect development board with computer, turn on the power switch, it will start working. Bluetooth module are in non-working, the block icon on the LCD screen predict that Bluetooth module's status.

Press WK UP button, it will turn on the Bluetooth module. LED on Bluetooth module and development board will start blinking, icon on screen turn to turn-on.

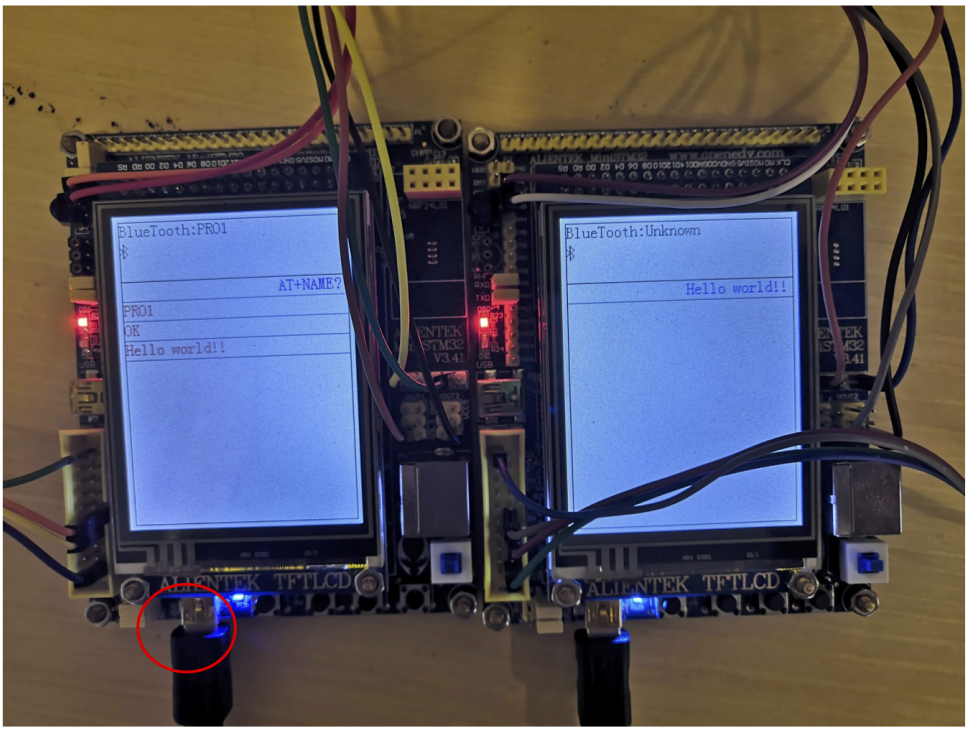

Figure 2: Initial Status

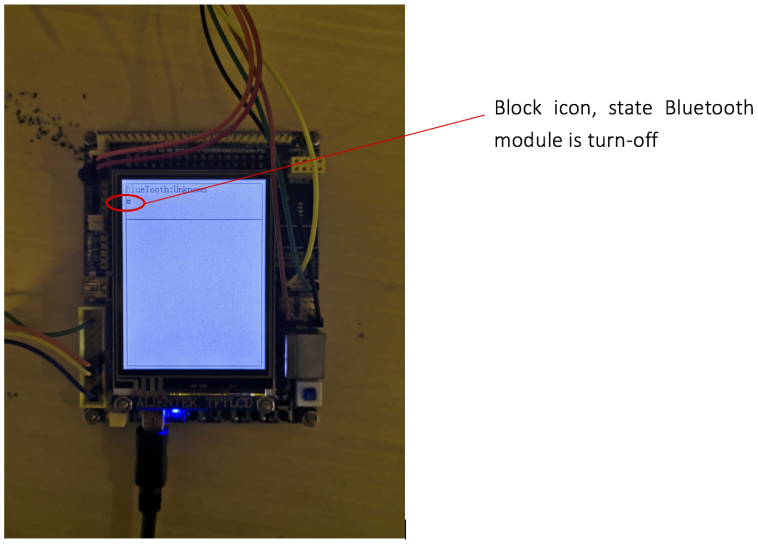

Figure 3: Our Custom Block Icon

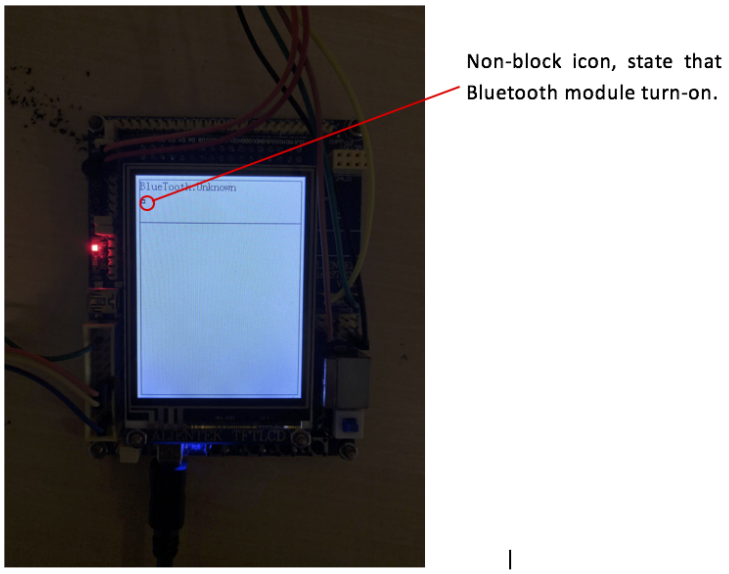

Figure 4: Icon Non-block — Connection Open

Press KEY0 button, it will connect the default Bluetooth address; Press

KEY1 button, it will connect any Bluetooth address. Remember, while press KEY0 or KEY1 you need to long press the button on the Bluetooth module.

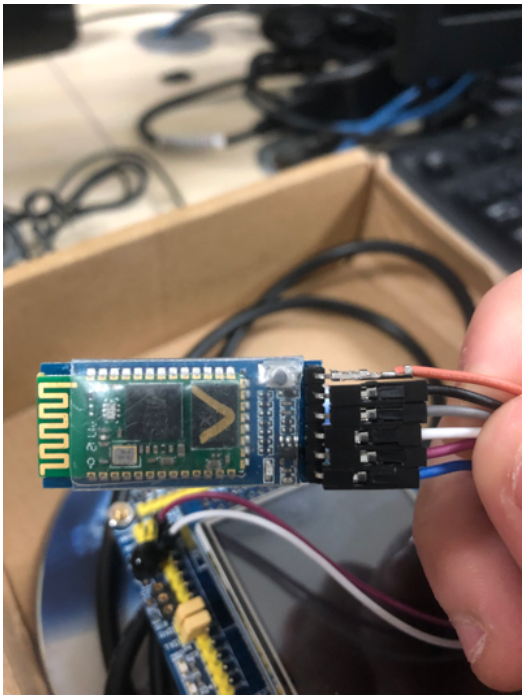

Figure 5: BlueTooth Controller Button

Press the button on the Bluetooth button, it will turn to AT pattern. In this status, type specific instruction can configure Bluetooth module. It is both work at connecting status. For example, type 'AT+NAME?' with carriage return on serial debugging software, it will return the name of the Bluetooth module. And the name will refresh on the top of the screen.

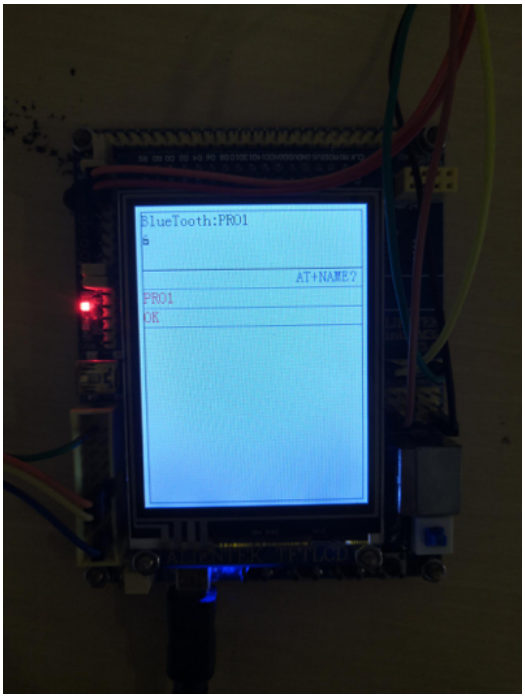

Figure 6: Connection Establishing Phrase

If Bluetooth connect the other one, icon will turn into the Connected status.

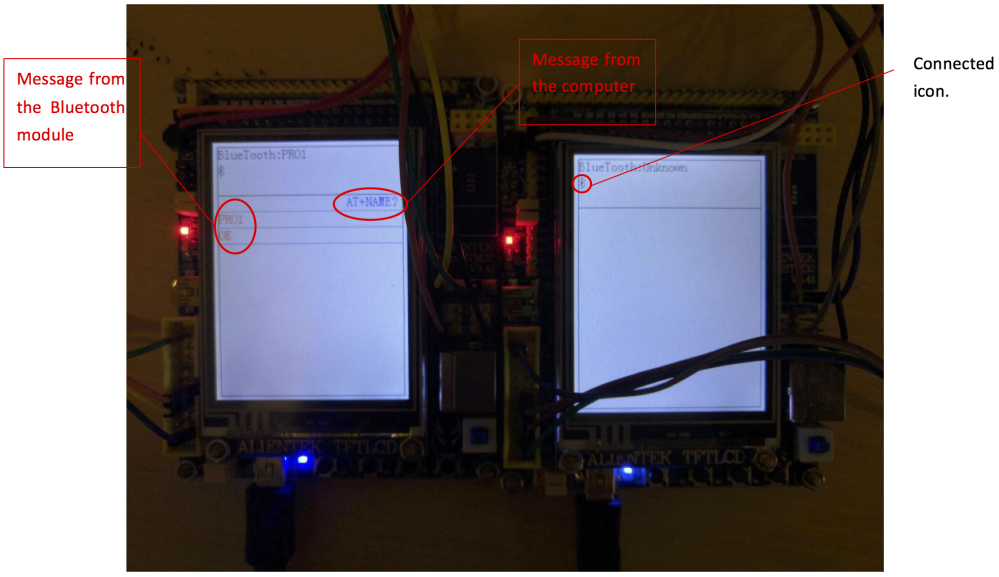

Figure 7: Connection Established Phrase

In connect status, red LED on the development board will normally on.And LED on the Bluetooth module will flicker once per two seconds. In connect status, type any chat content with carriage return in the serial debugging software and send ,it will transfer to the connected Bluetooth device.

The message trans into the development board from the computer will plot on the right of the screen;message from the Bluetooth module,include chat content from the other device and respond of the Bluetooth module,will plot on the left side of the screen. Any message send out and send in from the development board,the green LED on it will light for 1 second.

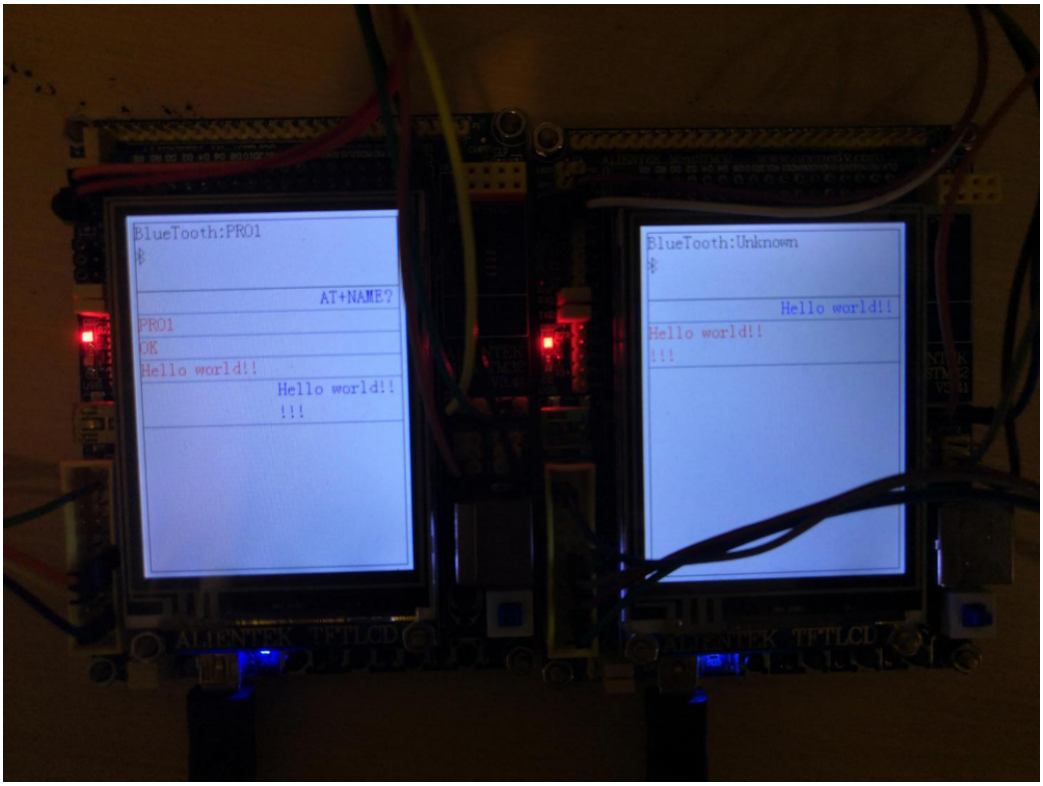

Figure 8: Exchange Messages via BlueTooth

Remember, any message and AT instruction send with serial debugging software must add carriage return at last.

5 Work Division

After discussion, We divide our team into two groups: Meng ChuTong and Long Chen first deal with the BlueTooth communication part and Wang Zhiyue and Luo Yunhao's task is to design a beautiful UI. However, this division is certainly not absolutely, since as we need to know the techniques of the other side when we are trying to connect our BlueTooth and UI. All of us participate in the writing of our documentations.

Table 1: WorkLoad Division

	Meng ChuTong	Long Chen	Wang ZhiYue	Luo Yunhao
BlueTooth	⋆⋆	⋆⋆	⋆	⋆
UI	⋆	⋆	⋆⋆	⋆⋆
Documentation	⋆	⋆	⋆	⋆

6 Gain and Conclusion

This project is somehow challenging but still meaningful and interesting. Technically, we learn plenty of lessons. Firstly, the decode packet char received from bluetooth may cause the LCD cannot correctly display the string. Secondly, if there is something wrong in LCD display, it is possibly that there are some errors in the code when implemented but there is no exception to be called. Thirdly, for each drawing, if it is not covered, then it will be reserved on the screen although for the other loop, it didn't draw it. So, the best way to erase the drawing displayed before is to cover the drawing.

In a nutshell, we have a basic idea of how to utilize Bluetooth module to communicate between two PCs. We learned how to change the state of Bluetooth. We also learned how to configure the module using AT instructions. Moreover, we have a better understanding of USART and how to use it as the relay between user and Bluetooth. We learned about the use of interrupt and utilize it to implement some functions.