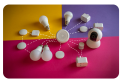


LoRa Long Distance Device Control and Communication System

Preface

- I. Project Description
- II. Project Goals

Background Knowledge

- I. Wireless Sensor Network Mind Map
- II. Introduction to Low Power Wide Area Network (LPWAN)
- III. Introduction to LoRa

Project Scenario

- I. System Architecture
- II. LoRa Data Reception Sequence
- III. LoRa Data Transmission Sequence

Development Tools

- I. Development Platform
- II. Project Package
- III. Technologies Used

Program

- I. Program Structure
- II. Program 1
  - Pin Define
  - Includes
  - Initialization
  - Get/Set RF Setting
  - LoRa Rx
  - LoRa Tx Rx
- III. Program 2
  - LoRa Tx Rx
- IV. Program 3

Reference

Preface

I. Project Description

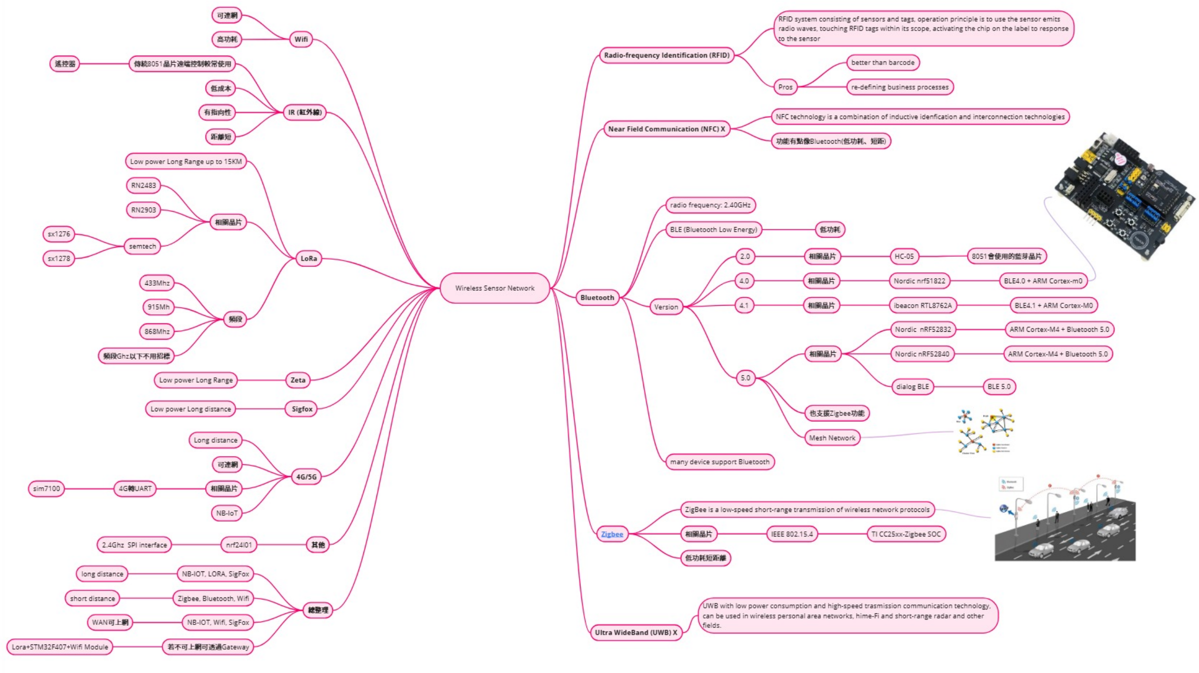
This project builds a long distance device control and communication system which is capable of sending and retrieving data from far-away devices, and broadcast the retreived data to relevant parties. Our system includes three main participants: End-device (Cortex M7), Mediator (LoRa+Cortex M3+USB) and MQTT broker (PC).

II. Project Goals

1. Build a long distance device control and communication system to send and retrieve data from far-away devices.
2. Buid a user interface for the main controller to easily access all the functionalities.
3. Provide the main controller the function to broadcast the recieved data to all relevant parties through MQTT mechanism.

Background Knowledge

I. Wireless Sensor Network Mind Map


Wireless Sensor Network
Wifi
可連網
高功耗
IR (紅外線)
傳統8051晶片遠端控制較常使用
遙控器
低成本
有指向性
距離短
LoRa
Low power Long Range up to 15KM
相關晶片
RN2483
RN2903
semtech
sx1276
sx1278
頻段
433Mhz
915Mh
868Mhz
頻段Ghz以下不用招標
Zeta
Low power Long Range
Sigfox
Low power Long distance
4G/5G
Long distance
可連網
相關晶片
4G轉UART
sim7100
NB-IoT
其他
nrf24l01
2.4Ghz SPI interface
總整理
NB-IOT, LORA, SigFox
long distance
Zigbee, Bluetooth, Wifi
short distance
NB-IOT, Wifi, SigFox
WAN可上網
若不可上網可透過Gateway
Lora+STM32F407+Wifi Module
Radio-frequency Identification (RFID)
RFID system consisting of sensors and tags, operation principle is to use the sensor emits radio waves, touching RFID tags within its scope, activating the chip on the label to response to the sensor
Pros
better than barcode
re-defining business processes
Near Field Communication (NFC) X
NFC technology is a combination of inductive idenfication and interconnection technologies
功能有點像Bluetooth(低功耗、短距)
Bluetooth
radio frequency: 2.40GHz
BLE (Bluetooth Low Energy)
低功耗
Version
2.0
相關晶片
HC-05
8051會使用的藍芽晶片
4.0
相關晶片
Nordic nrf51822
BLE4.0 + ARM Cortex-m0
4.1
相關晶片
ibeacon RTL8762A
BLE4.1 + ARM Cortex-M0
5.0
相關晶片
Nordic nRF52832
ARM Cortex-M4 + Bluetooth 5.0
Nordic nRF52840
ARM Cortex-M4 + Bluetooth 5.0
dialog BLE
BLE 5.0
也支援Zigbee功能
Mesh Network
many device support Bluetooth
Zigbee
ZigBee is a low-speed short-range transmission of wireless network protocols
相關晶片
IEEE 802.15.4
TI CC25xx-Zigbee SOC
低功耗短距離
Ultra WideBand (UWB) X
UWB with low power consumption and high-speed trasmission communication technology, can be used in wireless personal area networks, hime-Fi and short-range radar and other fields.

II. Introduction to Low Power Wide Area Network (LPWAN)

1. **Comparison with other wireless network**

Wireless Network	Local Area Network	Low Power Wide Area Network (LPWAN)	Cellular Network
Purpose	Internet of Things: Short Range Communication	Internet of Things: Long Range Communication	Traditional M2M
Pros	1. Well established standards. 2. In building	1. Low power consumption 2. Long range 3. Low cost	1. Existing coverage 2. High data rate
Cons	1. Rely on battery live	1. Low data rate	1. Cost of ownership
Technology	Bluetooth, Zigbee, WiFi	Lora, Sigfox, NB-IoT	4G, 5G

2. **Features of LPWAN**
    a. Long battery life (10 years)
    b. Low device cost (US $10)
    c. Long range (10-15 km in rural area, 2-5 km in urban area)
    d. Low data rate
    e. Support for massive number of devices

III. Introduction to LoRa

1. **What is LoRa?**
    a. LoRa (short for long range) is a spread spectrum modulation technique derived from chirp spread spectrum (CSS) technology, defined by Semtech
    b. Designed for the use in battery powered devices
    c. Uses "Spread Spectrum Technology"
    d. Range up to 15-20km
    e. LoRa ISM Band:

i. 868 MHz for Europe

ii. 915 MHz for North America

iii. 433 MHz for Asia

iv. 920-928 MHz for Taiwan

f. Don't need a license to use the LoRa band

2. **Features of LoRa**

Features	Description
Low data rate	a. 250bps-50Kbps
Long range	a. Greater than cellular b. Deep indoor coverage
Long battery life	a. Low-power b. More than 10 years on AA battery
Low cost	a. Minimal infrastucture b. Low-cost end-devices

3. **Important Terms in LoRa**

a. Chirp Spread Spectrum (CSS)

In digital communications, chirp spread spectrum (CSS) is a spread spectrum technique that uses wideband linear frequency modulated chirp pulses to encode information. [wiki]

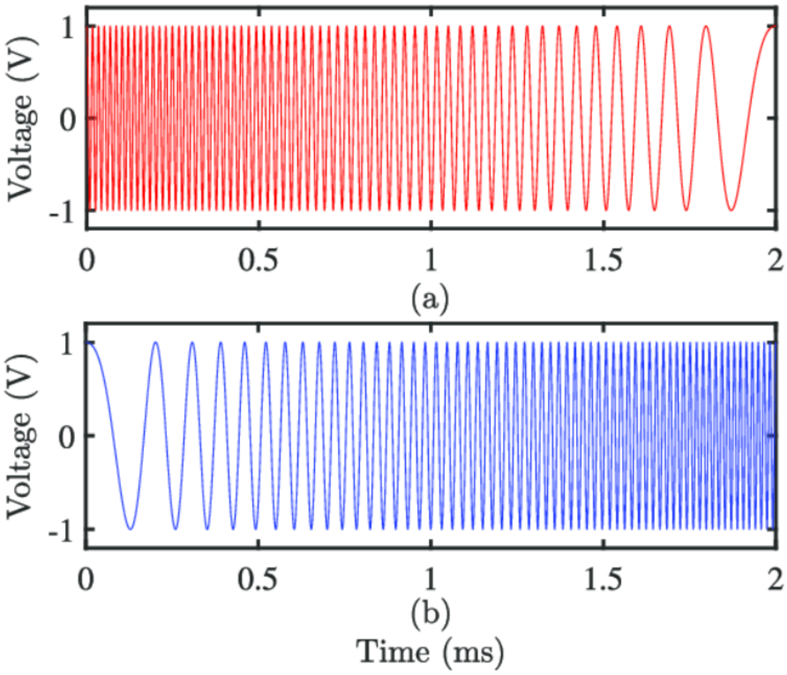
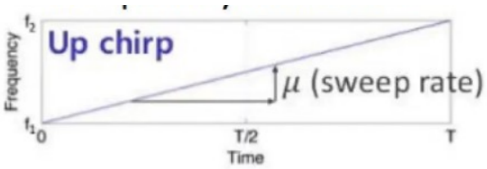
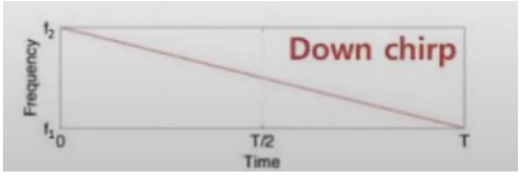


Illustration of chirp signals used to modulate data. (a) ''down-chirp'' signal. (b) ''up-chirp'' signal. [citation]



Up-chirp: Frequency increases



Down-chirp: Frequency decreases

b. Symbols

    i. LoRa Sumbols contain up-chirps and down-chirps

    ii. Each symbol encodes SF bits

    iii. A symbol has 2^SF values (eg. data: 1010111 ⇒ SF=8, symbol has 2^8 values range 0-255)

c. Spreading Factors (SF)

    i. Number of raw bits that can be encoded by the symbol

    ii. Often range 7-12

d. Forward Error Correction (FEC)

    i. Helps to restore data

    ii. Adding more error correction bitd decreses data rate and battery life

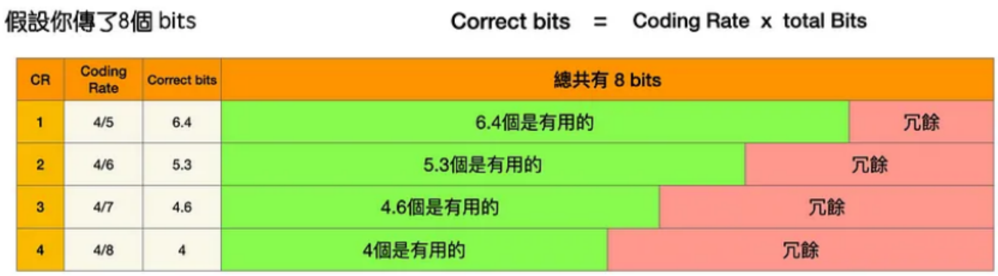
e. Coding Rate (CR)

    i. The coding rate refers to the proportion of transmitted bits that actually carry information

    ii. LoRa has 4 CR values: 1-4

    iii. Coding Rate = 4 / (4 + CR)

    iv. Correct bits calculation example [citation]



假設你傳了8個 bits   Correct bits = Coding Rate x total Bits

CR	Coding Rate	Correct bits	總共有 8 bits	
1	4/5	6.4	6.4個是有用的	冗餘
2	4/6	5.3	5.3個是有用的	冗餘
3	4/7	4.6	4.6個是有用的	冗餘
4	4/8	4	4個是有用的	冗餘

    f. Bandwidth (BW)

        i. LoRa uses three bandwidth: 125KHz, 250KHz, 500KHz

4. Spreading Factor Impact

    a. Far with obstacles:

        i. High sensitivity required

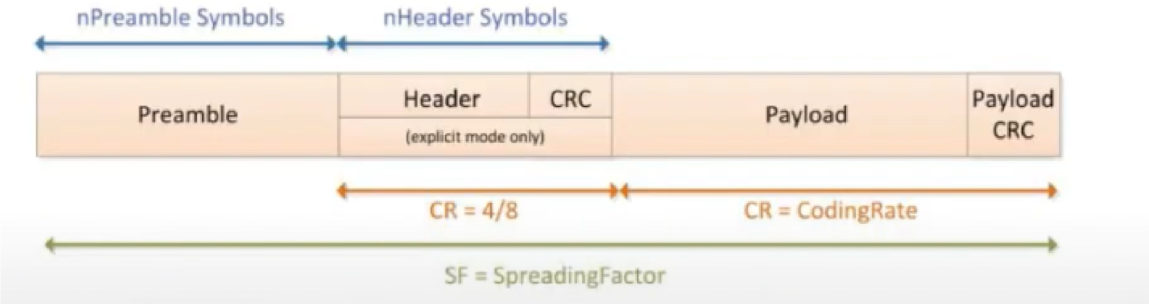
        ii. Increase the SF ⇒ Throughput decreases ⇒ Transmission time increase

    b. Close:

        i. Low sensitivity is sufficient

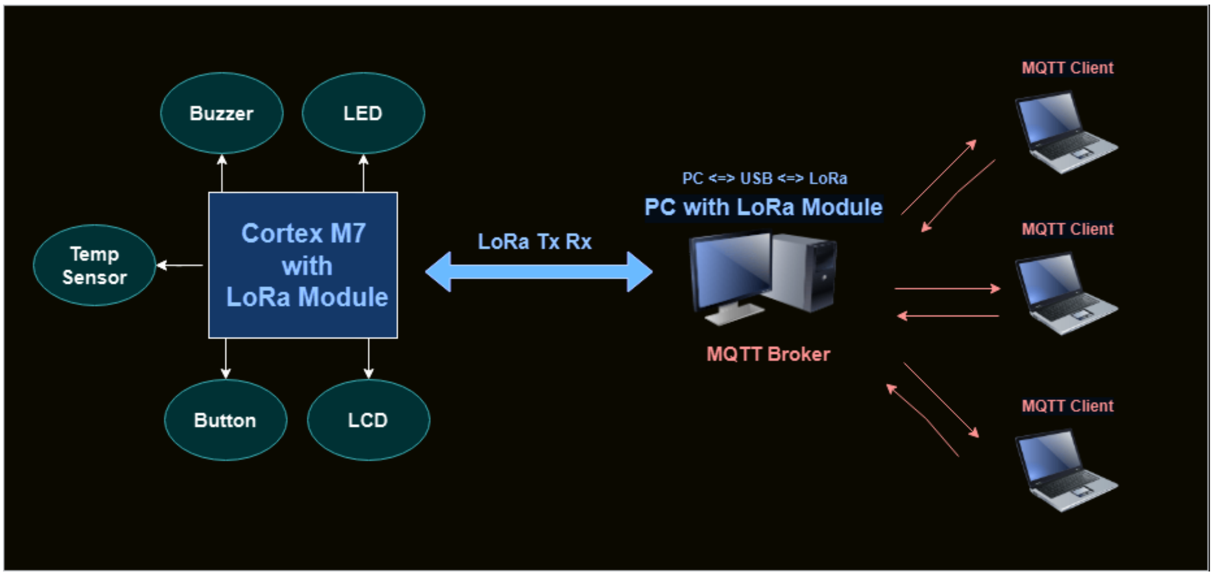
        ii. Decrease the SF ⇒ Throughput increases ⇒ Transmission time decrease

5. LoRa Packet Format



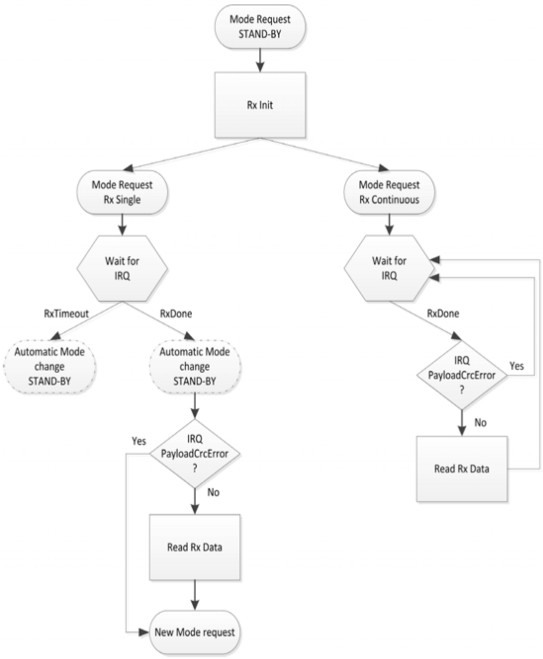
Project Scenario

I. System Architecture



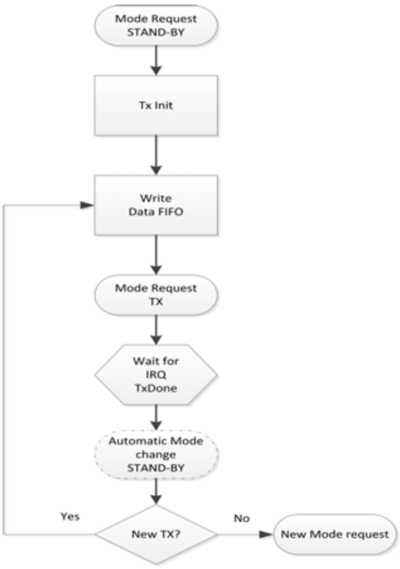
System Architecture

II. LoRa Data Reception Sequence



Typical LoRa receive sequences for both single and continuous receiver modes of operation [Semtech]

III. LoRa Data Transmission Sequence



LoRa Modulation Transmission Sequence [Semtech]

Development Tools

I. Development Platform

1. Keil **uVision** 5 & 4
2. Visual Studio Code

II. Project Package

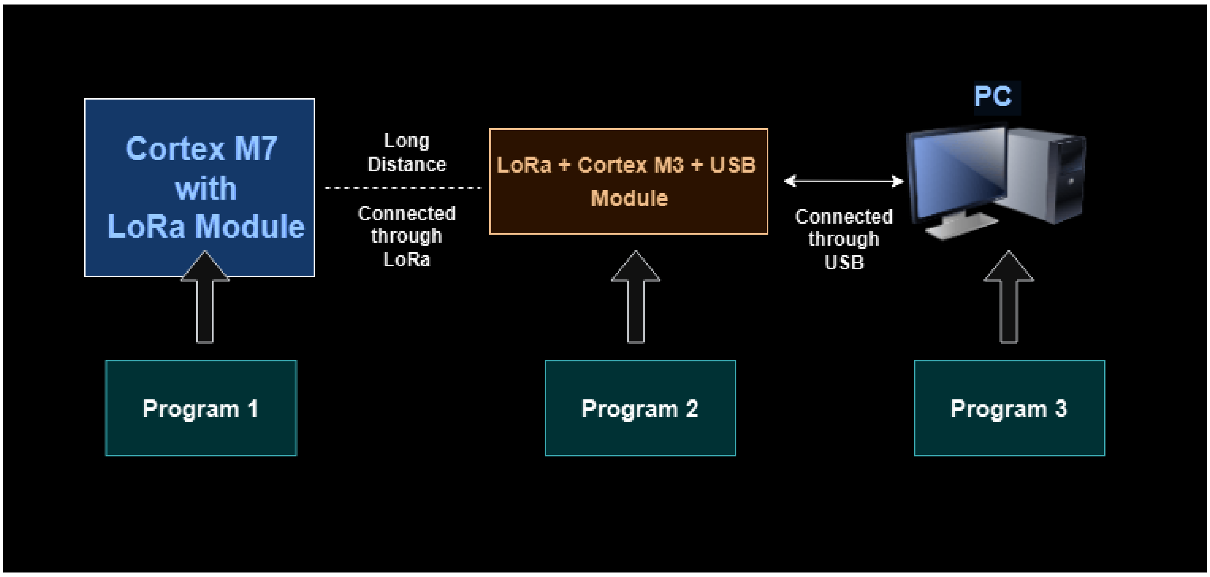
1. STM32H723ZG (Cortex M7)
2. 2.8 inch 320*240 Touch LCD
3. LM75A temperature sensor module
4. Buzzer (on board module)
5. LED (on board module)
6. Button (on board module)
7. LoRa module (connected to Cortex M7 through SPI)
8. LoRa to USB module (connected to PC through USB)

III. Technologies Used

1. MCU (Microcontroller Unit)
2. LoRa (L**ong Range)**
3. USB (Universal Serial Bus)
4. MQTT (Message Queuing Telemetry Transport)

Program

I. Program Structure

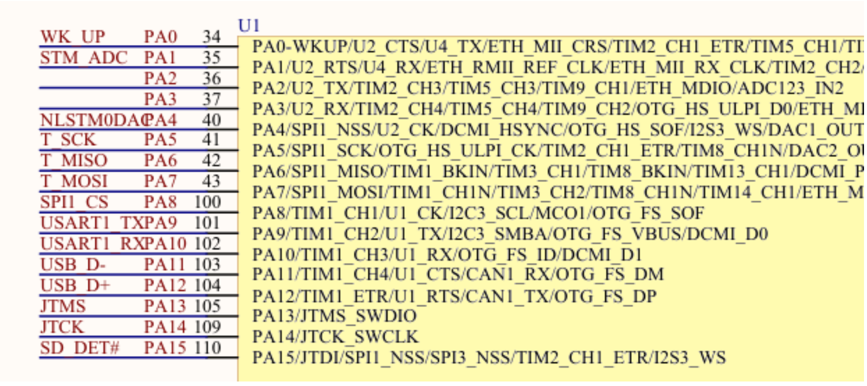


Program Structure

II. Program 1

Pin Define

Cortex M7 Pin	Function
PA5	SPI1 SCK
PA6	SPI1 MISO
PA7	SPI1 MOSI
PA8	CS (Chip Select)
PG9	GPIO (input mode) ⇒ IRQ(UINT2)
PG13	GPIO (output mode) ⇒ RST(UI03)


U1
WK_UP PA0 34
STM_ADC PA1 35
PA2 36
PA3 37
NLSTM0DACPA4 40
T_SCK PA5 41
T_MISO PA6 42
T_MOSI PA7 43
SPI1_CS PA8 100
USART1_TXPA9 101
USART1_RXPA10 102
USB_D- PA11 103
USB_D+ PA12 104
JTMS PA13 105
JTCK PA14 109
SD_DET# PA15 110
PA0-WKUP/U2_CTS/U4_TX/ETH_MII_CRS/TIM2_CH1_ETR/TIM5_CH1/TI
PA1/U2_RTS/U4_RX/ETH_RMII_REF_CLK/ETH_MII_RX_CLK/TIM2_CH2/
PA2/U2_TX/TIM2_CH3/TIM5_CH3/TIM9_CH1/ETH_MDIO/ADC123_IN2
PA3/U2_RX/TIM2_CH4/TIM5_CH4/TIM9_CH2/OTG_HS_ULPI_D0/ETH_MI
PA4/SPI1_NSS/U2_CK/DCMI_HSYNC/OTG_HS_SOF/I2S3_WS/DAC1_OUT
PA5/SPI1_SCK/OTG_HS_ULPI_CK/TIM2_CH1_ETR/TIM8_CH1N/DAC2_OU
PA6/SPI1_MISO/TIM1_BKIN/TIM3_CH1/TIM8_BKIN/TIM13_CH1/DCMI_P
PA7/SPI1_MOSI/TIM1_CH1N/TIM3_CH2/TIM8_CH1N/TIM14_CH1/ETH_M
PA8/TIM1_CH1/U1_CK/I2C3_SCL/MCO1/OTG_FS_SOF
PA9/TIM1_CH2/U1_TX/I2C3_SMBA/OTG_FS_VBUS/DCMI_D0
PA10/TIM1_CH3/U1_RX/OTG_FS_ID/DCMI_D1
PA11/TIM1_CH4/U1_CTS/CAN1_RX/OTG_FS_DM
PA12/TIM1_ETR/U1_RTS/CAN1_TX/OTG_FS_DP
PA13/JTMS_SWDIO
PA14/JTCK_SWCLK
PA15/JTDI/SPI1_NSS/SPI3_NSS/TIM2_CH1_ETR/I2S3_WS

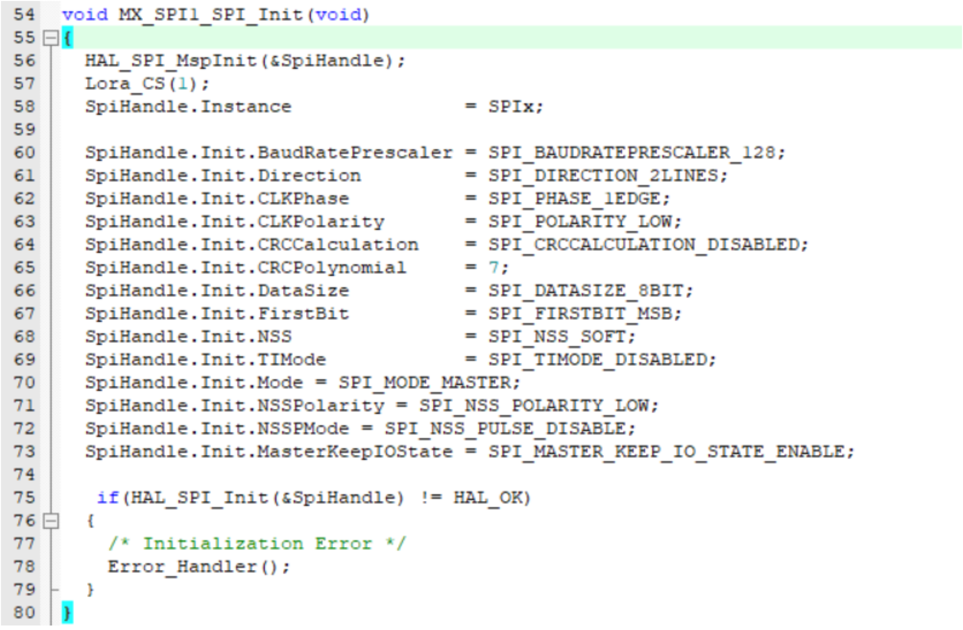
Cortex M7 Pin Define

Includes

Initialization

1. `MX_SPI1_SPI_Init()` (main.c) ⇒

   `SPI_HandleTypeDef` struct member setting(spi.c)



```
void MX_SPI1_SPI_Init(void)
{
  HAL_SPI_MspInit(&SpiHandle);
  Lora_CS(1);
  SpiHandle.Instance                = SPIx;

  SpiHandle.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_128;
  SpiHandle.Init.Direction         = SPI_DIRECTION_2LINES;
  SpiHandle.Init.CLKPhase          = SPI_PHASE_1EDGE;
  SpiHandle.Init.CLKPolarity       = SPI_POLARITY_LOW;
  SpiHandle.Init.CRCCalculation    = SPI_CRCCALCULATION_DISABLED;
  SpiHandle.Init.CRCPolynomial     = 7;
  SpiHandle.Init.DataSize          = SPI_DATASIZE_8BIT;
  SpiHandle.Init.FirstBit          = SPI_FIRSTBIT_MSB;
  SpiHandle.Init.NSS               = SPI_NSS_SOFT;
  SpiHandle.Init.TIMode            = SPI_TIMODE_DISABLED;
  SpiHandle.Init.Mode = SPI_MODE_MASTER;
  SpiHandle.Init.NSSPolarity = SPI_NSS_POLARITY_LOW;
  SpiHandle.Init.NSSPMode = SPI_NSS_PULSE_DISABLE;
  SpiHandle.Init.MasterKeepIOState = SPI_MASTER_KEEP_IO_STATE_ENABLE;

   if(HAL_SPI_Init(&SpiHandle) != HAL_OK)
  {
    /* Initialization Error */
    Error_Handler();
  }
}
```

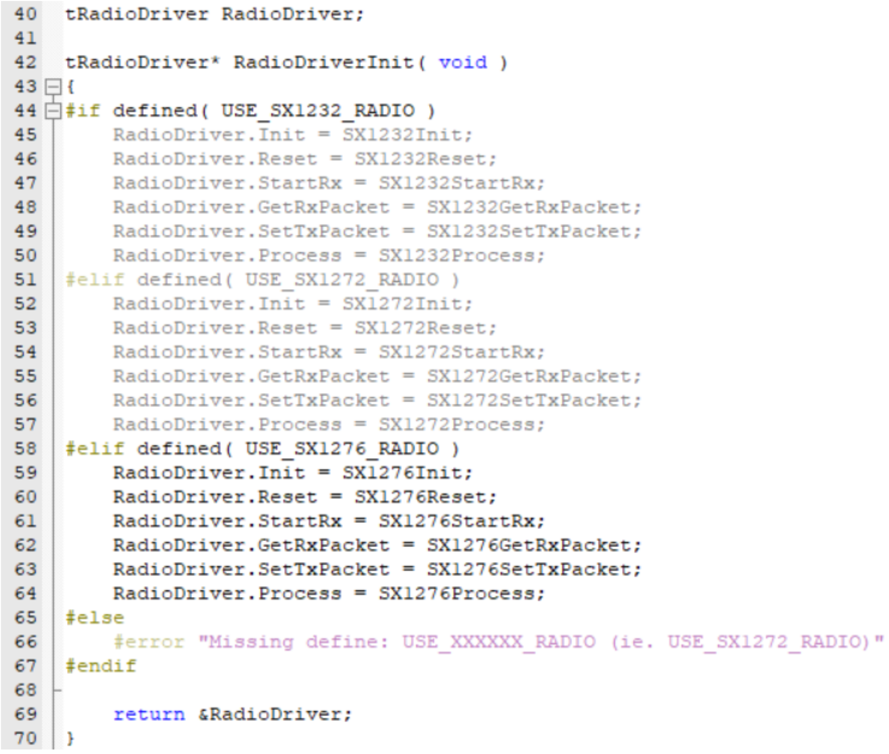
2. `HAL_SPI_Init()` (spi.c) ⇒

   `HAL_SPI_MspInit()` (stm32h7xx_hal_spi.c) ⇒

   clock enable & GPIO pin configuration (spi.c)

3. `Radio = RadioDriverInit()` (main.c) ⇒

   according to the chip selected, define the different function pointers within `sRadioDriver` struct and return the struct instance (radio.c, provided by SEMTECH)



```
tRadioDriver RadioDriver;

tRadioDriver* RadioDriverInit( void )
{
#if defined( USE_SX1232_RADIO )
    RadioDriver.Init = SX1232Init;
    RadioDriver.Reset = SX1232Reset;
    RadioDriver.StartRx = SX1232StartRx;
    RadioDriver.GetRxPacket = SX1232GetRxPacket;
    RadioDriver.SetTxPacket = SX1232SetTxPacket;
    RadioDriver.Process = SX1232Process;
#elif defined( USE_SX1272_RADIO )
    RadioDriver.Init = SX1272Init;
    RadioDriver.Reset = SX1272Reset;
    RadioDriver.StartRx = SX1272StartRx;
    RadioDriver.GetRxPacket = SX1272GetRxPacket;
    RadioDriver.SetTxPacket = SX1272SetTxPacket;
    RadioDriver.Process = SX1272Process;
#elif defined( USE_SX1276_RADIO )
    RadioDriver.Init = SX1276Init;
    RadioDriver.Reset = SX1276Reset;
    RadioDriver.StartRx = SX1276StartRx;
    RadioDriver.GetRxPacket = SX1276GetRxPacket;
    RadioDriver.SetTxPacket = SX1276SetTxPacket;
    RadioDriver.Process = SX1276Process;
#else
    #error "Missing define: USE_XXXXXX_RADIO (ie. USE_SX1272_RADIO)"
#endif

    return &RadioDriver;
}
```

4. `Radio->Init()` (main.c) ⇒

   `SX1276Init()` (sx1276.c, provided by SEMTECH) ⇒

   `SX1276InitIo()` clock enable & GPIO pin configuration for IRQ and Reset pin (sx1276-Hal.c, provided by SEMTECH)

5. `SX1276Init()` (sx1276.c, provided by SEMTECH) ⇒

   `SX1276Read(0x06, &TempReg)` read from register 0x06 to check whether the initialization is done (sx1276-Hal.c, provided by SEMTECH) ⇒

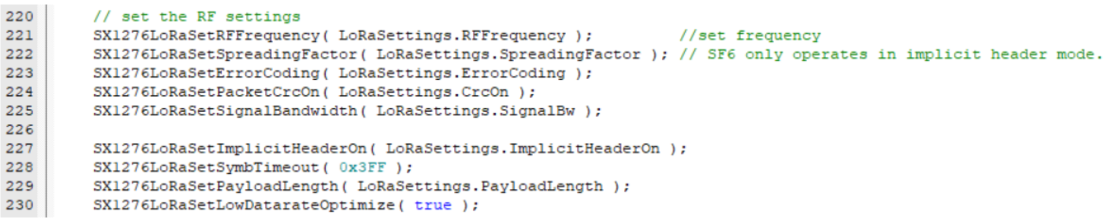
   `SX1276ReadBuffer()` (sx1276-Hal.c, provided by SEMTECH) ⇒

   `SpiInOut()` (spisx1278.c, provided by SEMTECH) ⇒

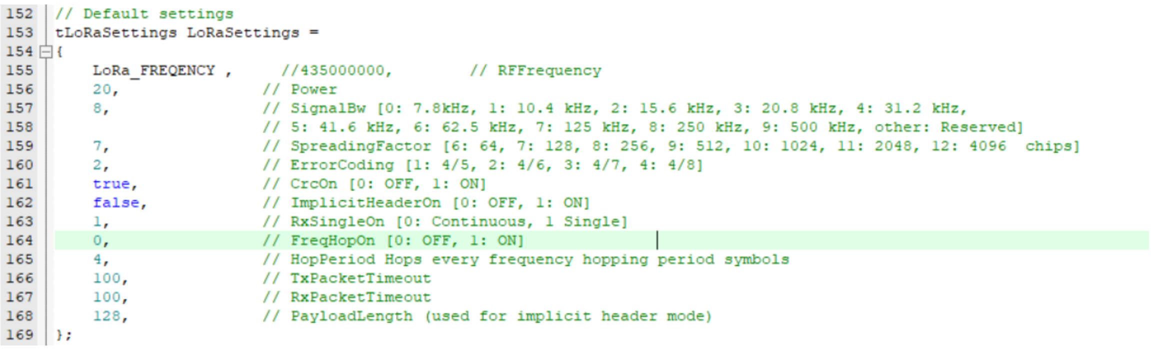
   `HAL_SPI_TransmitReceive()` (stm32h7xx_hal_spi.c)

6. `SX1276Init()` (sx1276.c, provided by SEMTECH) ⇒

   `SX1276LoRaInit()` set the RF setting (sx1276-LoRa.c, provided by SEMTECH)



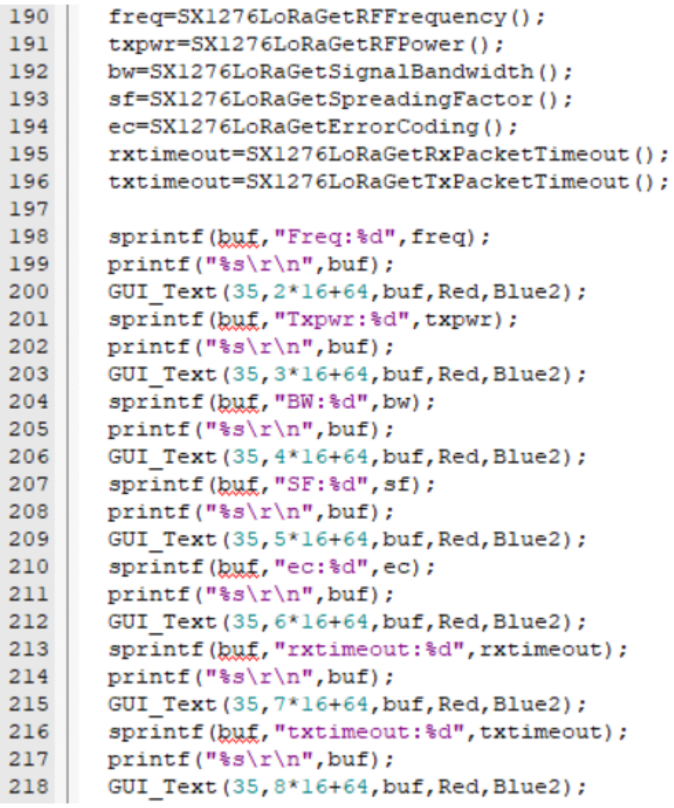
```
    // set the RF settings
    SX1276LoRaSetRFFrequency( LoRaSettings.RFFrequency );          //set frequency
    SX1276LoRaSetSpreadingFactor( LoRaSettings.SpreadingFactor ); // SF6 only operates in implicit header mode.
    SX1276LoRaSetErrorCoding( LoRaSettings.ErrorCoding );
    SX1276LoRaSetPacketCrcOn( LoRaSettings.CrcOn );
    SX1276LoRaSetSignalBandwidth( LoRaSettings.SignalBw );

    SX1276LoRaSetImplicitHeaderOn( LoRaSettings.ImplicitHeaderOn );
    SX1276LoRaSetSymbTimeout( 0x3FF );
    SX1276LoRaSetPayloadLength( LoRaSettings.PayloadLength );
    SX1276LoRaSetLowDatarateOptimize( true );
```



```
// Default settings
tLoRaSettings LoRaSettings =
{
    LoRa_FREQENCY ,      //435000000,        // RFFrequency
    20,                 // Power
    8,                  // SignalBw [0: 7.8kHz, 1: 10.4 kHz, 2: 15.6 kHz, 3: 20.8 kHz, 4: 31.2 kHz,
                        // 5: 41.6 kHz, 6: 62.5 kHz, 7: 125 kHz, 8: 250 kHz, 9: 500 kHz, other: Reserved]
    7,                  // SpreadingFactor [6: 64, 7: 128, 8: 256, 9: 512, 10: 1024, 11: 2048, 12: 4096  chips]
    2,                  // ErrorCoding [1: 4/5, 2: 4/6, 3: 4/7, 4: 4/8]
    true,               // CrcOn [0: OFF, 1: ON]
    false,              // ImplicitHeaderOn [0: OFF, 1: ON]
    1,                  // RxSingleOn [0: Continuous, 1 Single]
    0,                  // FreqHopOn [0: OFF, 1: ON]
    4,                  // HopPeriod Hops every frequency hopping period symbols
    100,                // TxPacketTimeout
    100,                // RxPacketTimeout
    128,                // PayloadLength (used for implicit header mode)
};
```

Get/Set RF Setting

1. get RF setting (main.c)



```
    freq=SX1276LoRaGetRFFrequency();
    txpwr=SX1276LoRaGetRFPower();
    bw=SX1276LoRaGetSignalBandwidth();
    sf=SX1276LoRaGetSpreadingFactor();
    ec=SX1276LoRaGetErrorCoding();
    rxtimeout=SX1276LoRaGetRxPacketTimeout();
    txtimeout=SX1276LoRaGetTxPacketTimeout();

    sprintf(buf,"Freq:%d",freq);
    printf("%s\r\n",buf);
    GUI_Text(35,2*16+64,buf,Red,Blue2);
    sprintf(buf,"Txpwr:%d",txpwr);
    printf("%s\r\n",buf);
    GUI_Text(35,3*16+64,buf,Red,Blue2);
    sprintf(buf,"BW:%d",bw);
    printf("%s\r\n",buf);
    GUI_Text(35,4*16+64,buf,Red,Blue2);
    sprintf(buf,"SF:%d",sf);
    printf("%s\r\n",buf);
    GUI_Text(35,5*16+64,buf,Red,Blue2);
    sprintf(buf,"ec:%d",ec);
    printf("%s\r\n",buf);
    GUI_Text(35,6*16+64,buf,Red,Blue2);
    sprintf(buf,"rxtimeout:%d",rxtimeout);
    printf("%s\r\n",buf);
    GUI_Text(35,7*16+64,buf,Red,Blue2);
    sprintf(buf,"txtimeout:%d",txtimeout);
    printf("%s\r\n",buf);
    GUI_Text(35,8*16+64,buf,Red,Blue2);
```

2. `SX1276LoRaGetRFFrequency()` , `SX1276LoRaSetRFFrequency()` and other RF set/get functions are both in (sx1276-LoRaMisc.c, provided by SEMTECH)

3. `SX1276LoRaSetRFFrequency()` (sx1276-LoRaMisc.c, provided by SEMTECH) ⇒ `SX1276WriteBuffer()` (sx1276-Hal.c, provided by SEMTECH) ⇒

   `SpiInOut()` (spisx1278.c, provided by SEMTECH) ⇒

   `HAL_SPI_TransmitReceive()` (stm32h7xx_hal_spi.c)

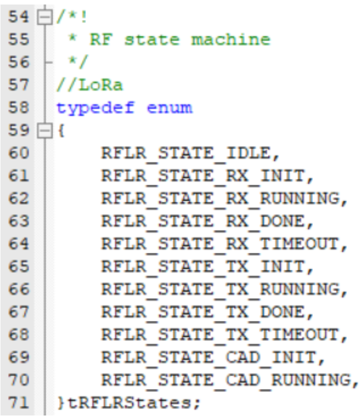
LoRa Rx

1. `Radio->StartRx()` (main.c) ⇒

   `SX1276StartRx()` (sx1276.c, provided by SEMTECH) ⇒

   `SX1276LoRaSetRFState()` (sx1276-LoRa.c, provided by SEMTECH) ⇒

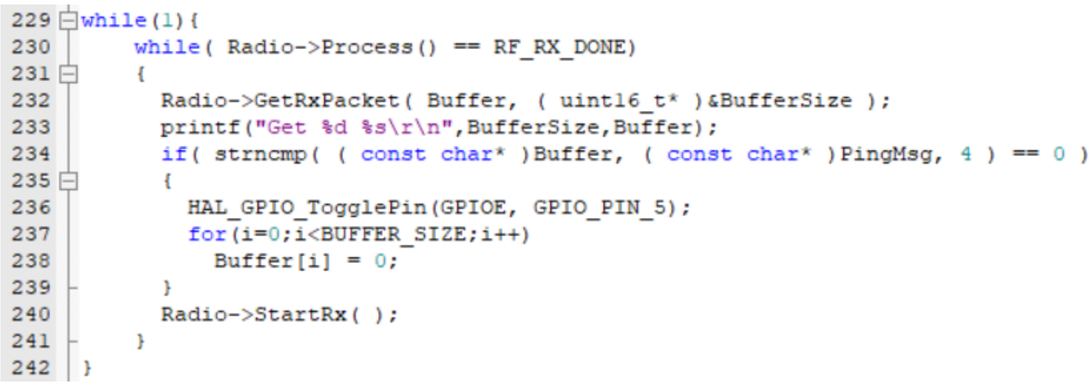
   `RFLRState = state;` set `RFLRState` to `RFLR_STATE_RX_INIT`



```
/*!
 * RF state machine
 */
//LoRa
typedef enum
{
    RFLR_STATE_IDLE,
    RFLR_STATE_RX_INIT,
    RFLR_STATE_RX_RUNNING,
    RFLR_STATE_RX_DONE,
    RFLR_STATE_RX_TIMEOUT,
    RFLR_STATE_TX_INIT,
    RFLR_STATE_TX_RUNNING,
    RFLR_STATE_TX_DONE,
    RFLR_STATE_TX_TIMEOUT,
    RFLR_STATE_CAD_INIT,
    RFLR_STATE_CAD_RUNNING,
}tRFLRStates;
```

RF State Machine

2. 此段程式會將收到的資料存進 `Buffer` ，並與 `PingMsg` 做比對，若前4位相同，則將LED燈做Toggle，並將 `Buffer` 清空，最後回到接收狀態 `RFLR_STATE_RX_INIT`



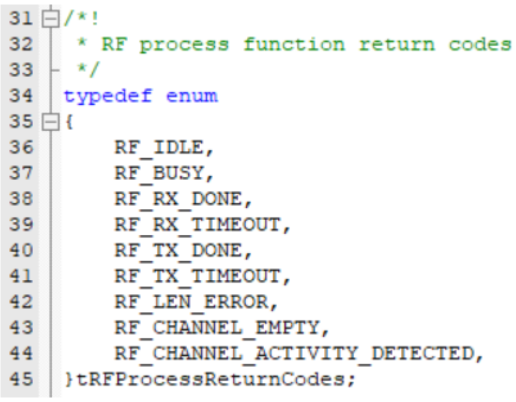
```
while(1){
    while( Radio->Process() == RF_RX_DONE)
    {
      Radio->GetRxPacket( Buffer, ( uint16_t* )&BufferSize );
      printf("Get %d %s\r\n",BufferSize,Buffer);
      if( strncmp( ( const char* )Buffer, ( const char* )PingMsg, 4 ) == 0 )
      {
        HAL_GPIO_TogglePin(GPIOE, GPIO_PIN_5);
        for(i=0;i<BUFFER_SIZE;i++)
          Buffer[i] = 0;
      }
      Radio->StartRx( );
    }
}
```

3. 我們可以去細看 `Radio->Process()` 的程式碼(main.c) ⇒

   `SX1276Process()` (sx1276.c, provided by SEMTECH) ⇒

   `SX1276LoRaProcess()` (sx1276-LoRa.c, provided by SEMTECH) ⇒

   此段程式會將 `RFLRState` 從 `RFLR_STATE_RX_INIT` 轉到 `RFLR_STATE_RX_RUNNING` ，若收到資料，則再將狀態轉到 `RFLR_STATE_RX_DONE` ，若CRC error check沒問題即會回傳 `RF_RX_DONE`

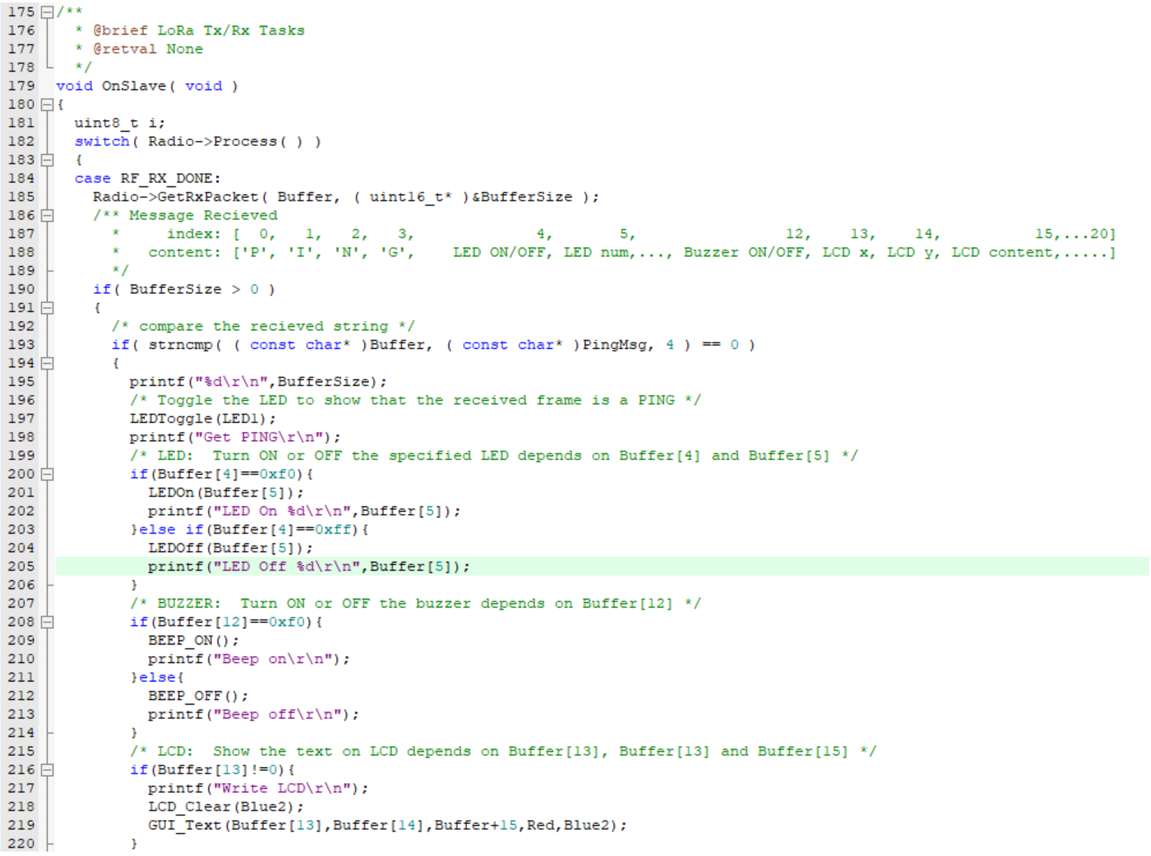


```
/*!
 * RF process function return codes
 */
typedef enum
{
    RF_IDLE,
    RF_BUSY,
    RF_RX_DONE,
    RF_RX_TIMEOUT,
    RF_TX_DONE,
    RF_TX_TIMEOUT,
    RF_LEN_ERROR,
    RF_CHANNEL_EMPTY,
    RF_CHANNEL_ACTIVITY_DETECTED,
}tRFProcessReturnCodes;
```

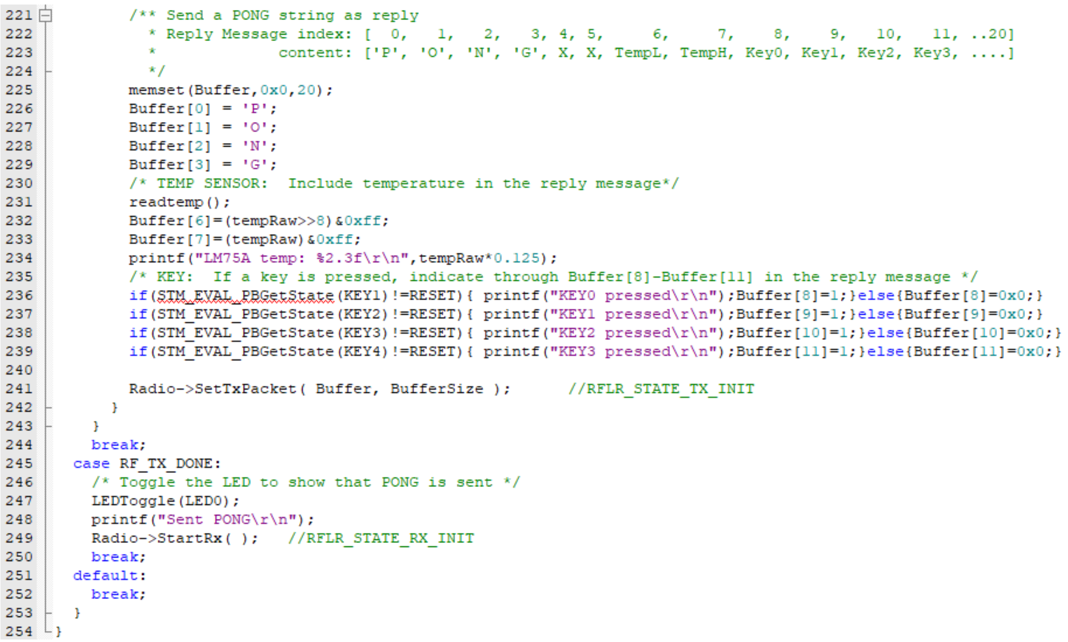
RF Process Function Return Codes

LoRa Tx Rx

1. `Radio->StartRx()` 先將Cortex M7 設為接收方，狀態為 `RFLR_STATE_RX_INIT` (main.c)
2. 與Program 2輪流互相收送資料，使用switch來根據RF process function return codes判斷目前是收到資料還是完成傳送資料，並根據收到的資料判斷如何控制Cortex M7 的 LED、Buzzer、LCD、Temperature Sensor 和 Key，最後切換收發狀態



```
/**
  * @brief LoRa Tx/Rx Tasks
  * @retval None
  */
void OnSlave( void )
{
  uint8_t i;
  switch( Radio->Process( ) )
  {
  case RF_RX_DONE:
    Radio->GetRxPacket( Buffer, ( uint16_t* )&BufferSize );
    /** Message Recieved
      *     index: [  0,    1,    2,    3,             4,        5,                12,    13,    14,          15,...20]
      *   content: ['P', 'I', 'N', 'G',    LED ON/OFF, LED num,..., Buzzer ON/OFF, LCD x, LCD y, LCD content,.....]
      */
    if( BufferSize > 0 )
    {
      /* compare the recieved string */
      if( strncmp( ( const char* )Buffer, ( const char* )PingMsg, 4 ) == 0 )
      {
        printf("%d\r\n",BufferSize);
        /* Toggle the LED to show that the received frame is a PING */
        LEDToggle(LED1);
        printf("Get PING\r\n");
        /* LED:  Turn ON or OFF the specified LED depends on Buffer[4] and Buffer[5] */
        if(Buffer[4]==0xf0){
          LEDOn(Buffer[5]);
          printf("LED On %d\r\n",Buffer[5]);
        }else if(Buffer[4]==0xff){
          LEDOff(Buffer[5]);
          printf("LED Off %d\r\n",Buffer[5]);
        }
        /* BUZZER:  Turn ON or OFF the buzzer depends on Buffer[12] */
        if(Buffer[12]==0xf0){
          BEEP_ON();
          printf("Beep on\r\n");
        }else{
          BEEP_OFF();
          printf("Beep off\r\n");
        }
        /* LCD:  Show the text on LCD depends on Buffer[13], Buffer[13] and Buffer[15] */
        if(Buffer[13]!=0){
          printf("Write LCD\r\n");
          LCD_Clear(Blue2);
          GUI_Text(Buffer[13],Buffer[14],Buffer+15,Red,Blue2);
        }
```



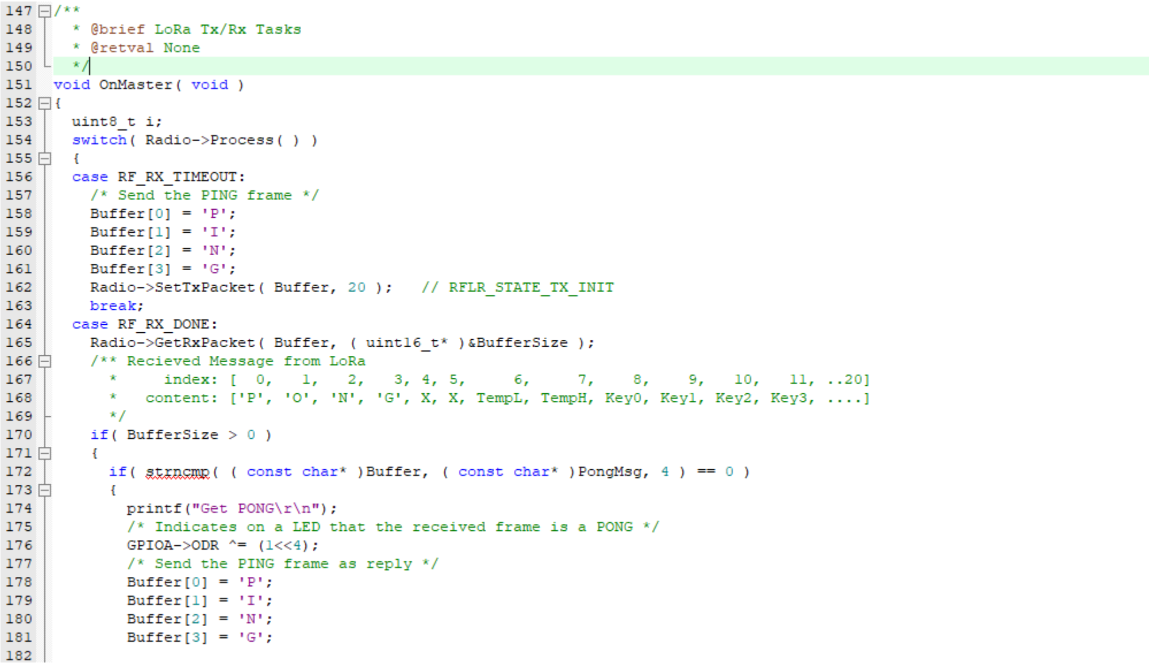
```
        /** Send a PONG string as reply
          * Reply Message index: [  0,    1,    2,    3, 4, 5,     6,     7,     8,     9,    10,    11, ..20]
          *             content: ['P', 'O', 'N', 'G', X, X, TempL, TempH, Key0, Key1, Key2, Key3, ....]
          */
        memset(Buffer,0x0,20);
        Buffer[0] = 'P';
        Buffer[1] = 'O';
        Buffer[2] = 'N';
        Buffer[3] = 'G';
        /* TEMP SENSOR:  Include temperature in the reply message*/
        readtemp();
        Buffer[6]=(tempRaw>>8)&0xff;
        Buffer[7]=(tempRaw)&0xff;
        printf("LM75A temp: %2.3f\r\n",tempRaw*0.125);
        /* KEY:  If a key is pressed, indicate through Buffer[8]-Buffer[11] in the reply message */
        if(STM_EVAL_PBGetState(KEY1)!=RESET){ printf("KEY0 pressed\r\n");Buffer[8]=1;}else{Buffer[8]=0x0;}
        if(STM_EVAL_PBGetState(KEY2)!=RESET){ printf("KEY1 pressed\r\n");Buffer[9]=1;}else{Buffer[9]=0x0;}
        if(STM_EVAL_PBGetState(KEY3)!=RESET){ printf("KEY2 pressed\r\n");Buffer[10]=1;}else{Buffer[10]=0x0;}
        if(STM_EVAL_PBGetState(KEY4)!=RESET){ printf("KEY3 pressed\r\n");Buffer[11]=1;}else{Buffer[11]=0x0;}

        Radio->SetTxPacket( Buffer, BufferSize );      //RFLR_STATE_TX_INIT
      }
    }
    break;
  case RF_TX_DONE:
    /* Toggle the LED to show that PONG is sent */
    LEDToggle(LED0);
    printf("Sent PONG\r\n");
    Radio->StartRx( );   //RFLR_STATE_RX_INIT
    break;
  default:
    break;
  }
}
```

III. Program 2

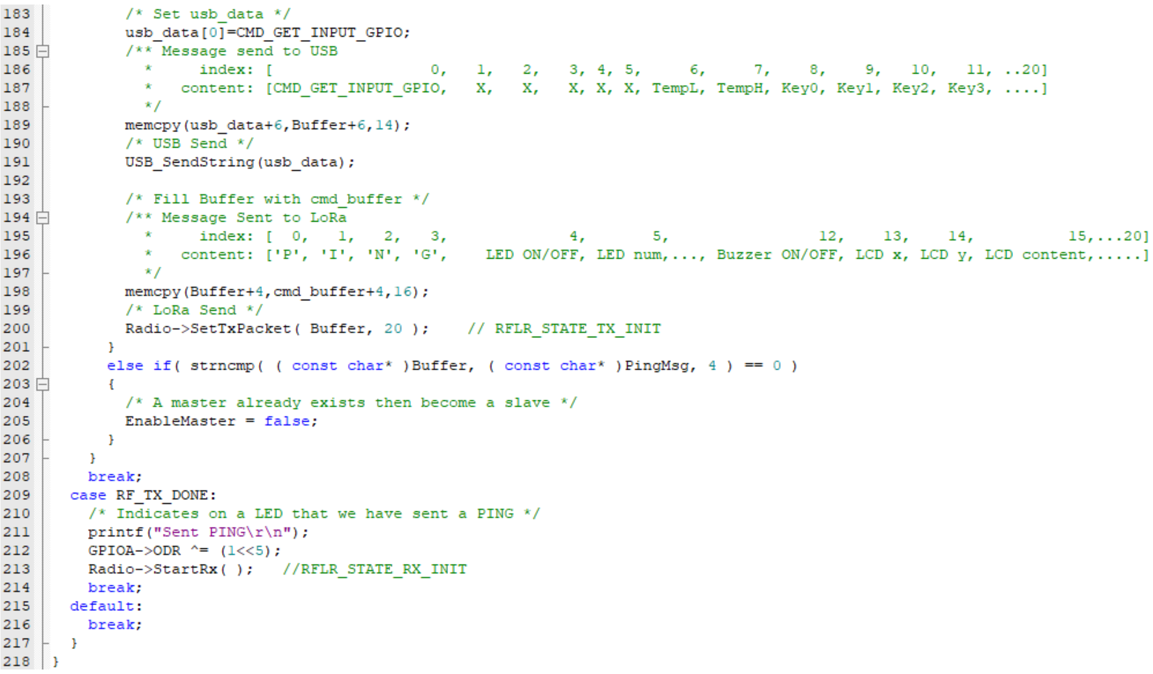
LoRa Tx Rx

1. `Radio->SetTxPacket(PingMsg, 4)` 先將Cortex M7 設為接送方，狀態為 `RFLR_STATE_RX_INIT` (main.c)

2. 與Program 1輪流互相收送資料，使用switch來根據RF process function return codes判斷目前是收到資料還是完成傳送資料，並切換收發狀態，若長時間未收到資料則為 `RF_RX_TIMEOUT` ，直接轉為發送方，若有收到資料，則將收到的資料透過USB轉送給PC

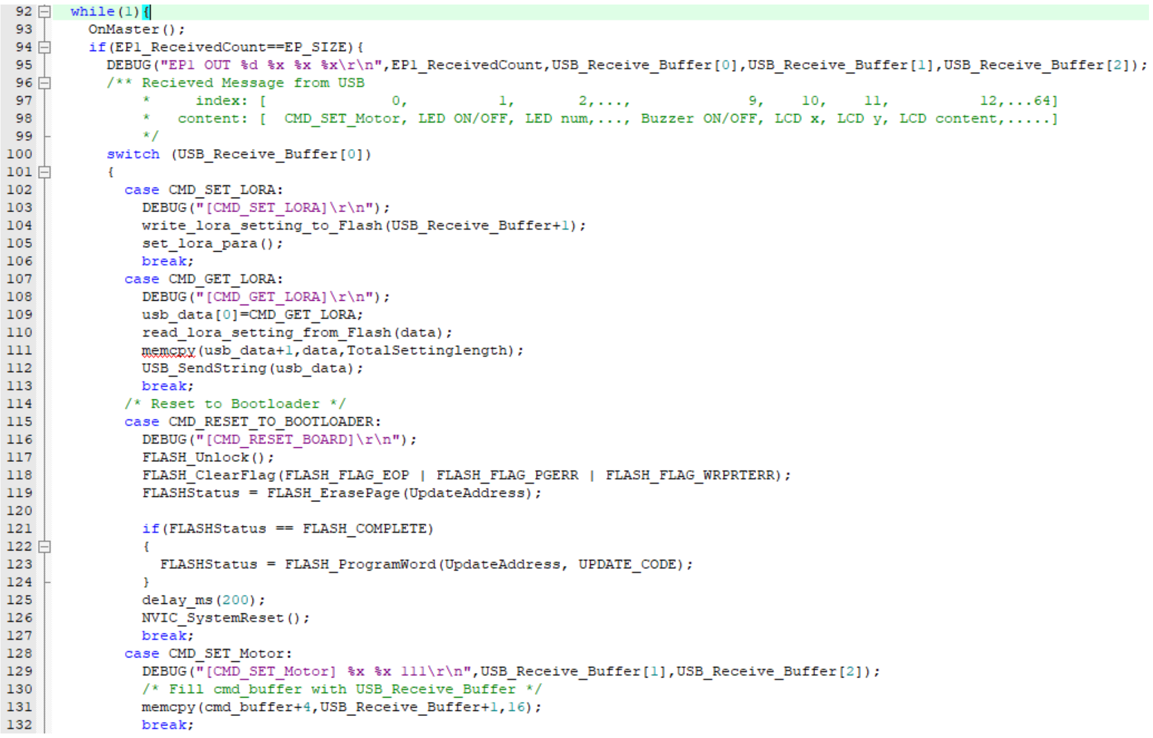


```
/**
  * @brief LoRa Tx/Rx Tasks
  * @retval None
  */
void OnMaster( void )
{
  uint8_t i;
  switch( Radio->Process( ) )
  {
  case RF_RX_TIMEOUT:
    /* Send the PING frame */
    Buffer[0] = 'P';
    Buffer[1] = 'I';
    Buffer[2] = 'N';
    Buffer[3] = 'G';
    Radio->SetTxPacket( Buffer, 20 );    // RFLR_STATE_TX_INIT
    break;
  case RF_RX_DONE:
    Radio->GetRxPacket( Buffer, ( uint16_t* )&BufferSize );
    /** Recieved Message from LoRa
      *     index: [  0,    1,    2,    3, 4, 5,     6,     7,     8,     9,    10,    11, ..20]
      *   content: ['P', 'O', 'N', 'G', X, X, TempL, TempH, Key0, Key1, Key2, Key3, ....]
      */
    if( BufferSize > 0 )
    {
      if( strncmp( ( const char* )Buffer, ( const char* )PongMsg, 4 ) == 0 )
      {
        printf("Get PONG\r\n");
        /* Indicates on a LED that the received frame is a PONG */
        GPIOA->ODR ^= (1<<4);
        /* Send the PING frame as reply */
        Buffer[0] = 'P';
        Buffer[1] = 'I';
        Buffer[2] = 'N';
        Buffer[3] = 'G';

```

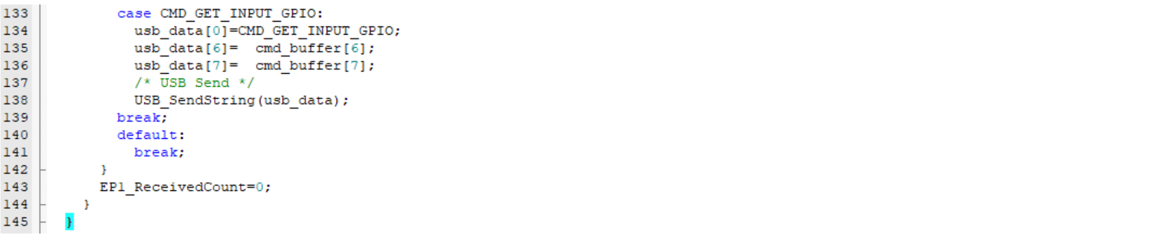


```
        /* Set usb_data */
        usb_data[0]=CMD_GET_INPUT_GPIO;
        /** Message send to USB
          *     index: [                  0,    1,    2,    3, 4, 5,      6,      7,     8,     9,    10,    11, ..20]
          *   content: [CMD_GET_INPUT_GPIO,    X,    X,    X, X, X, TempL, TempH, Key0, Key1, Key2, Key3, ....]
          */
        memcpy(usb_data+6,Buffer+6,14);
        /* USB Send */
        USB_SendString(usb_data);

        /* Fill Buffer with cmd_buffer */
        /** Message Sent to LoRa
          *     index: [  0,    1,    2,    3,             4,        5,                12,     13,     14,          15,...20]
          *   content: ['P', 'I', 'N', 'G',     LED ON/OFF, LED num,..., Buzzer ON/OFF, LCD x, LCD y, LCD content,.....]
          */
        memcpy(Buffer+4,cmd_buffer+4,16);
        /* LoRa Send */
        Radio->SetTxPacket( Buffer, 20 );    // RFLR_STATE_TX_INIT
      }
      else if( strncmp( ( const char* )Buffer, ( const char* )PingMsg, 4 ) == 0 )
      {
        /* A master already exists then become a slave */
        EnableMaster = false;
      }
    }
    break;
  case RF_TX_DONE:
    /* Indicates on a LED that we have sent a PING */
    printf("Sent PING\r\n");
    GPIOA->ODR ^= (1<<5);
    Radio->StartRx( );    //RFLR_STATE_RX_INIT
    break;
  default:
    break;
  }
}
```

3. 除了透過LoRa與Cortex M7做溝通，Program 2也負責透過USB接收與傳送訊息給PC，以下是根據PC端C#人機互動介面獲得指令所採取的對應工作



```
while(1){
  OnMaster();
  if(EP1_ReceivedCount==EP_SIZE){
    DEBUG("EP1 OUT %d %x %x %x\r\n",EP1_ReceivedCount,USB_Receive_Buffer[0],USB_Receive_Buffer[1],USB_Receive_Buffer[2]);
    /** Recieved Message from USB
        *     index: [             0,          1,        2,...,              9,     10,     11,          12,...64]
        *   content: [  CMD_SET_Motor, LED ON/OFF, LED num,..., Buzzer ON/OFF, LCD x, LCD y, LCD content,.....]
        */
    switch (USB_Receive_Buffer[0])
    {
      case CMD_SET_LORA:
        DEBUG("[CMD_SET_LORA]\r\n");
        write_lora_setting_to_Flash(USB_Receive_Buffer+1);
        set_lora_para();
        break;
      case CMD_GET_LORA:
        DEBUG("[CMD_GET_LORA]\r\n");
        usb_data[0]=CMD_GET_LORA;
        read_lora_setting_from_Flash(data);
        memcpy(usb_data+1,data,TotalSettinglength);
        USB_SendString(usb_data);
        break;
      /* Reset to Bootloader */
      case CMD_RESET_TO_BOOTLOADER:
        DEBUG("[CMD_RESET_BOARD]\r\n");
        FLASH_Unlock();
        FLASH_ClearFlag(FLASH_FLAG_EOP | FLASH_FLAG_PGERR | FLASH_FLAG_WRPRTERR);
        FLASHStatus = FLASH_ErasePage(UpdateAddress);

        if(FLASHStatus == FLASH_COMPLETE)
        {
          FLASHStatus = FLASH_ProgramWord(UpdateAddress, UPDATE_CODE);
        }
        delay_ms(200);
        NVIC_SystemReset();
        break;
      case CMD_SET_Motor:
        DEBUG("[CMD_SET_Motor] %x %x 111\r\n",USB_Receive_Buffer[1],USB_Receive_Buffer[2]);
        /* Fill cmd_buffer with USB_Receive_Buffer */
        memcpy(cmd_buffer+4,USB_Receive_Buffer+1,16);
        break;
```

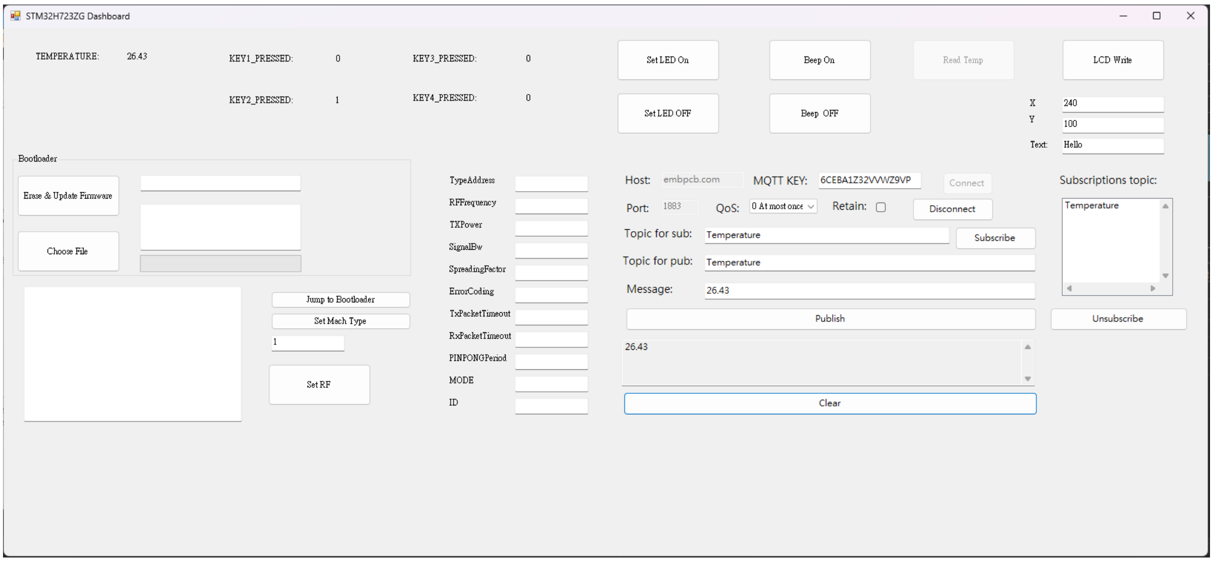


```
      case CMD_GET_INPUT_GPIO:
        usb_data[0]=CMD_GET_INPUT_GPIO;
        usb_data[6]=  cmd_buffer[6];
        usb_data[7]=  cmd_buffer[7];
        /* USB Send */
        USB_SendString(usb_data);
      break;
      default:
        break;
    }
    EP1_ReceivedCount=0;
  }
}
```

IV. Program 3

此程式由C#撰寫，功能有以下幾點:

1. 提供使用者介面來遠端控制Cortex M7 LED、Buzzer、LCD等元件
2. 顯示Cortex M7即時溫度與按鍵狀態
3. 使用MQTT對訂閱Temperature此主題之Client自動發送Cortex M7 LM75A溫度訊息
4. 提供LoRa+Cortex M3+USB模組燒錄功能



TEMPERATURE: 26.43

KEY1_PRESSED: 0

KEY2_PRESSED: 1

KEY3_PRESSED: 0

KEY4_PRESSED: 0

X 240

Y 100

Text: Hello

1

Host: embpcb.com

MQTT KEY: 6CEBA1Z32VVWZ9VP

Port: 1883

QoS: 0 At most once

Retain: □

Topic for sub: Temperature

Topic for pub: Temperature

Message: 26.43

26.43

Subscriptions topic:

Temperature

Reference

LORA與LORAWAN基本觀念

各國LORAWAN頻段

LoRa相關Driver

中華行動數位科技教材