

UART - MBED-OS -

Handong university

Jong-won Lee

2

UART

USART in STM32-F411

3

□ USART Features

- ▣ Full duplex, asynchronous communications
 - x8 or x16 oversampling
- ▣ Supports USART synchronous communications
 - Transmitter clock output for synchronous transmission
 - Has the same format of asynchronous transmission
 - No clock at the start and stop bit
- ▣ Supports LIN (local interconnection network)
- ▣ Single-wire half-duplex communication
- ▣ Supports the asynchronous protocol of Smartcards as defined in the ISO 7816-3 standard
- ▣ IrDA SIR encoder decoder
- ▣ Can use DMA

USART in STM32-F411RE

4

□ 사용 가능한 USART 장치 및 USART 장치의 TX/RX 핀

USART 장치	RTS/CTS 신호선	최대 전송속도 (Mbps) (x16/x8 클럭 사용시)	비고 (Mbed-os에서 사용 가능한 핀) {TX/RX}
USART1	있음	6.25/12.5	{PA_9(D8)/PA_10(D2) {PA_15/PB_3} {PB_6(D10)/PB_7(CN7_21)}
USART2	있음	3.12/6.25	USB 디버거 포트를 PC와 연결하였을 때, Virtual comm 포트 {PA_2/PA_3}
USART6	없음	6.25/12.5	{PA_11/PA_12} {PC_6/PC_7}

□ USART 장치의 RX 입력 신호

■ 5V tolerant

USART in Nucleo-F411RE

5

▣ Cf.: https://github.com/ARMmbed/mbed-os/blob/master/targets/TARGET_STM/TARGET_STM32F4/TARGET_STM32F411xE/TARGET_NUCLEO_F411RE/PeripheralPinMaps.h

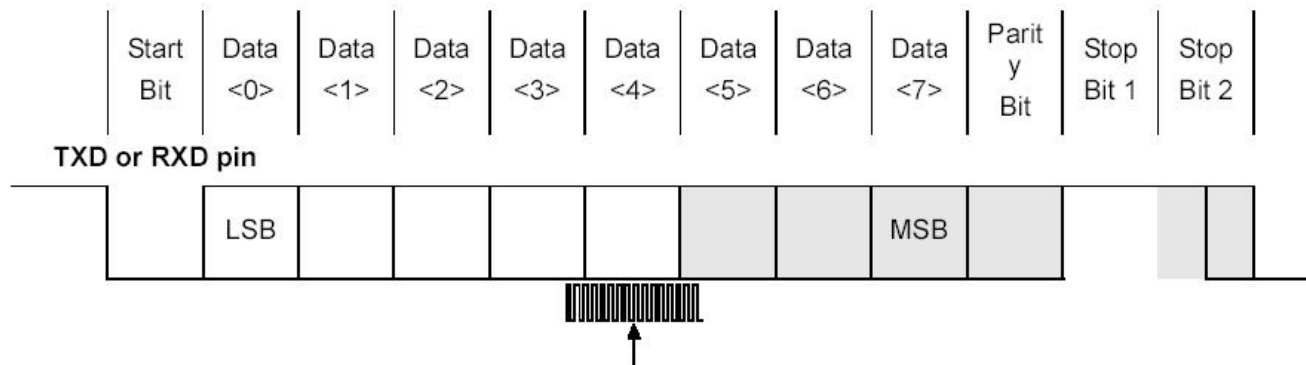
```
MSTD_CONSTEXPR_OBJ_11 PinMap PinMap_UART_TX[] = {
    {PA_2,      UART_2,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART2)}, // Connected to STDIO_UART_TX
    {PA_9,      UART_1,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART1)},
    {PA_11,     UART_6,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF8_USART6)},
    {PA_15,     UART_1,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART1)},
    {PB_6,      UART_1,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART1)},
    {PC_6,      UART_6,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF8_USART6)},
    {NC, NC, 0}
};
```

```
MSTD_CONSTEXPR_OBJ_11 PinMap PinMap_UART_RX[] = {
    {PA_3,      UART_2,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART2)}, // Connected to STDIO_UART_RX
    {PA_10,     UART_1,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART1)},
    {PA_12,     UART_6,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF8_USART6)},
    {PB_3,      UART_1,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART1)}, // Connected to SW0
    {PB_7,      UART_1,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF7_USART1)},
    {PC_7,      UART_6,  STM_PIN_DATA(STM_MODE_AF_PP, GPIO_PULLUP, GPIO_AF8_USART6)},
    {NC, NC, 0}
};
```

UART

6

□ Data frame

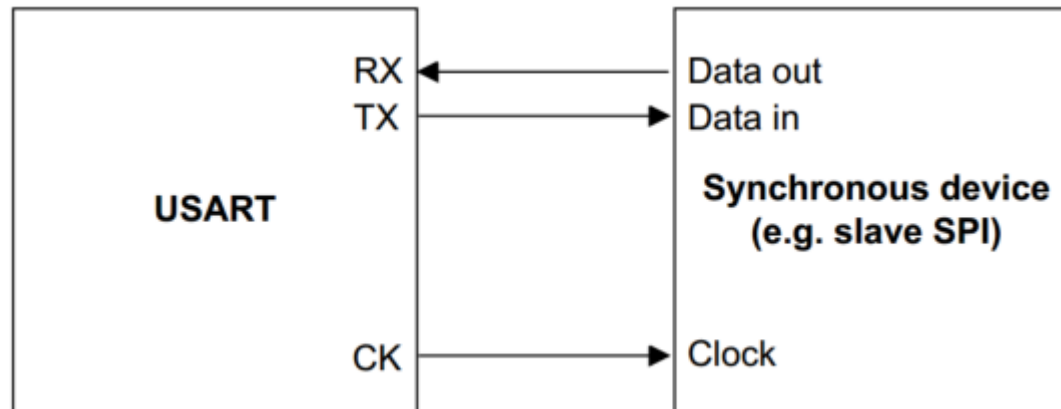


- TX: parallel-to-serial conversion with a start bit, and followed by the data bits (LSB first), parity bit, and the stop bits according to the programmed.
- RX: serial-to-parallel conversion after a valid start pulse has been detected.
 - Overrun, parity, frame error checking, and line-break detection are also performed.

Synchronous Mode in F411RE

7

- In synchronous mode the USART transmitter works exactly like in asynchronous mode.
- The CK pin is the output of the USART transmitter clock.
 - ▣ No clock pulses are sent to the CK pin during start bit and stop bit.

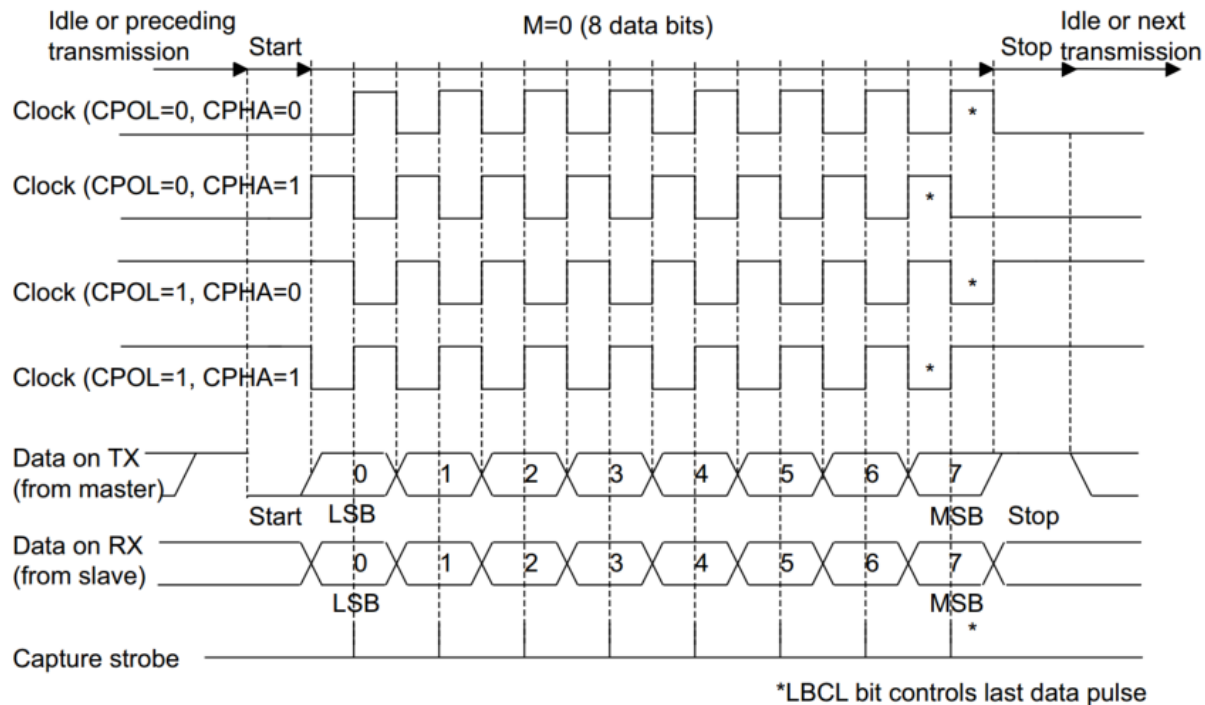


Synchronous Mode in F411RE

8

□ Timing diagram

- The same as that of SPI
- M=0 (8 data bits)

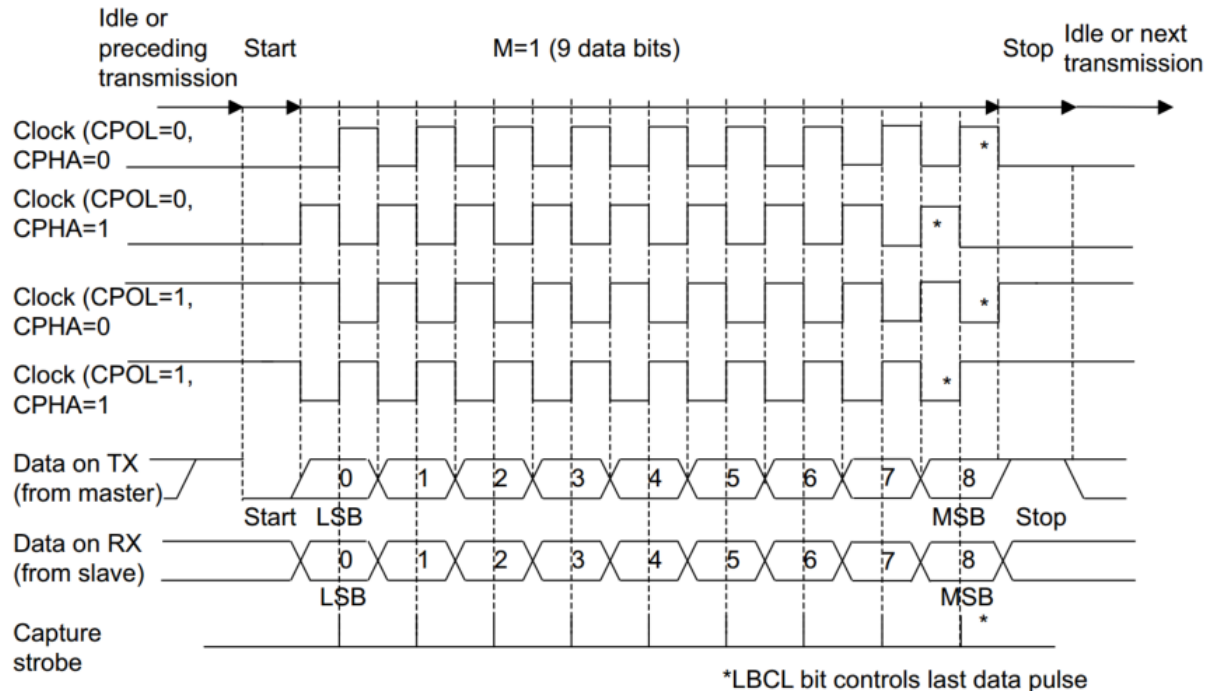


Synchronous Mode in F411RE

9

□ Timing diagram

- The same as that of SPI
- M=1 (9 data bits)



UART - MBED-OS -

Handong university

Jong-won Lee

Mbed-OS UART API

2

- Two classes
 - ▣ BufferedSerial and UnbufferedSerial
- Compared to the previous Mbed-os version, there are many changes.

Deprecated API	Replaced by
Serial	printf and puts which are used in the console . BufferedSerial for efficient serial communication. UnbufferedSerial for bypassing locks in IRQ or short of RAM.
RawSerial	UnbufferedSerial
UARTSerial	BufferedSerial

Mbed-OS BufferedSerial API

3

□ BufferedSerial

- When the receive interrupt is triggered when receiving data from a device, the BufferedSerial class stores the byte(s) available to read **from the hardware buffer to an internal intermediary buffer**.
When a read request is made, the BufferedSerial class uses a mutex lock and enters a critical section to read out the number of bytes requested if as many are available in the intermediary buffer.
- To transmit multiple bytes, the class **uses an intermediary buffer to store the bytes to send** and monitors the serial interface to transfer them to the hardware buffer as soon as it is available.
- The RX and TX buffers are circular buffers with preallocated sizes configurable using the configuration parameters **uart-serial-rxbuf-size** and **uart-serial-txbuf-size**. You can find both configuration parameters in `drivers/mbed_lib.json`.

BufferedSerial API

4

□ BufferedSerial class

생성자	BufferedSerial (PinName tx, PinName rx, int baud = MBED_CONF_PLATFORM_DEFAULT_SERIAL_BAUD_RATE)
설명	주어진 tx, rx 핀을 사용하는 BufferedSerial 객체를 생성한다. Default baud rate = 9600
사용 예	static BufferedSerial serial_port(CONSOLE_TX, CONSOLE_RX, 115200);
함수	void set_baud (int baud);
설명	UART의 baud rate를 설정함.
사용 예	serial_port.set_baud(115200);
함수	void set_format (int bits = 8, Parity parity = BufferedSerial::None, int stop_bits = 1)
설명	UART의 전송 포맷을 설정함. Parity: BufferedSerial::None, BufferedSerial::Odd, BufferedSerial::Even,
사용 예	serial_port.set_format(8, BufferedSerial::Odd, 2);

BufferedSerial API

5

□ BufferedSerial class

함수	<code>bool readable () const</code>
설명	읽을 데이터가 존재하면 true를 반환한다.
사용 예	<pre>If (serial_port.readable()) num = serial_port.read(buf, sizeof(buf));</pre>
함수	<code>bool writable () const</code>
설명	write()를 위한 버퍼가 존재하면 true를 반환한다.
사용 예	<pre>If (serial_port.writable()) serial_port.write(buf, num);</pre>
함수	<code>int set_blocking (bool blocking)</code>
설명	Blocking 혹은 non-blocking 모드로 설정. (디폴트 모드는 <u>blocking</u> 모드이다.)
사용 예	<code>serial_port.set_blocking(true);</code>

blocking
true : 기다림
false : EAGSAIN

BufferedSerial API

6

□ BufferedSerial class

함수	<code>ssize_t read (void * buffer, size_t size)</code>
설명	<p>버퍼로 수신한 데이터를 읽는다.</p> <p>읽을 데이터가 없을 경우:</p> <ul style="list-style-type: none">- Blocking function이면, 읽을 데이터가 있을 때까지 기다린다.- Non-blocking function이면, <code>-EAGAIN</code> 값을 반환한다. <p>반환값:</p> <ul style="list-style-type: none">- <u>읽은 데이터의 개수.</u>- error인 경우는 음수 값을 반환한다.
사용 예	<code>num = serial_port.read(buf, sizeof(buf));</code>



BufferedSerial API

7

□ BufferedSerial class

함수	<code>ssize_t write (void * buffer, size_t size)</code>
설명	<p>버퍼의 내용을 전송한다.</p> <p>충분한 tx 버퍼가 없을 경우:</p> <ul style="list-style-type: none">- Blocking function이면, 모든 데이터를 쓸 때까지 기다린다.- Non-blocking function 일 경우:<ul style="list-style-type: none">.tx 버퍼가 full이면, EAGAIN을 반환한다..일부 데이터를 쓸 수 있으면, 일부 데이터를 쓴다. <p>반환값:</p> <ul style="list-style-type: none">- 쓴 데이터의 개수. <i>return</i>- error인 경우는 음수 값을 반환한다.
사용 예	<code>serial_port.write(buf, num);</code>

Blocking : 모든 데이터를 쓸 때까지 기다린다

*non-blocking : tx 버퍼가 full이면 EAGAIN return
아니면 일부 데이터를 쓴다.*

BufferedSerial API

8

□ BufferedSerial class

함수	void sigio (Callback< void()> func)
설명	상태 변화에 대한 callback 함수를 등록한다. (쓸 수 있거나, 읽을 수 있는 것과 같이 상태 변화가 있을 때에 callback 함수가 실행된다.) Callback 함수 내에서 heavy job을 수행하지 않아야 한다. (어떤 interrupt context 안에서 callback 함수가 불리움.)
사용 예	serial_port.sigio (rx_handler);

상태 변화 (쓸 수 있거나, 읽을 수 있거나)

Floating point in mbed-os v.6.0 printf()

9

- Refer to: <https://github.com/ARMmbed/mbed-os/blob/master/platform/source/minimal-printf/README.md>
- To reduce the code size Mbed introduced Minimal printf and snprintf.
 - ▣ As of Mbed OS 6.0 it is enabled by default. Floating point parameters are only present when minimal-printf-enable-floating-point config is set to true (disabled by default).
- To enable floating-point parameters, configure it in **mbed_app.json**

```
{
  "target_overrides": {
    "*": {
      "target.printf_lib": "std"
    }
  }
}
```

```
{
  "target_overrides": {
    "*": {
      "target.printf_lib": "minimal-printf"
    }
  }
}
```

Lab4-1 ~ Lab4-3: UART

10

□ 실습 목적

- ▣ 1. Mbed-OS 상에서 UART 장치를 동작시키는 방법에 대해 이해한다.
 - BufferedSerial and UnbufferedSerial
- ▣ 2. 터미널 에뮬레이션 프로그램을 사용할 수 있다.

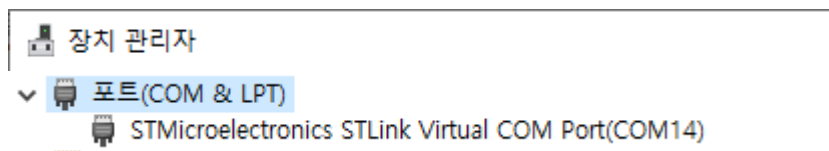
□ 실습 과정

- ▣ PC와 Nucleo board를 USB 케이블을 이용하여 연결한다.
- ▣ TeraTerm을 실행시키고 동작 환경을 설정한다. (포트 연결 및 Baudrate 설정)
 - Baud rate: 115,200 bps
 - Format: 8 bit data, 1 stop bit, no parity
- ▣ 사용자의 입력이 터미널 에뮬레이터에 나타나도록 코드를 구현한다.
 - 사용자의 입력을 polling/interrupt 방식에 의해서 감지하여라.
- ▣ 사용자가 Enter(엔터) 키를 입력할 경우에 다음 줄로 넘어가도록 한다.

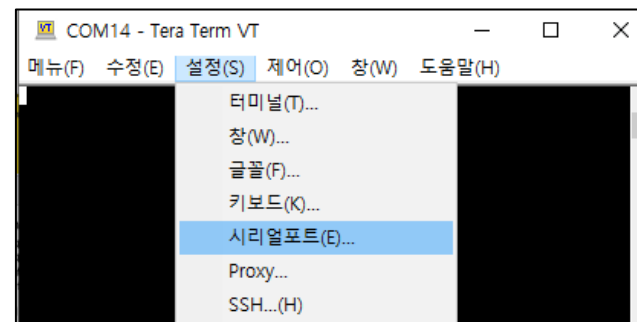
Lab4-1: TeraTerm 설정

11

① [제어판] – [장치 관리자]: 연결 확인



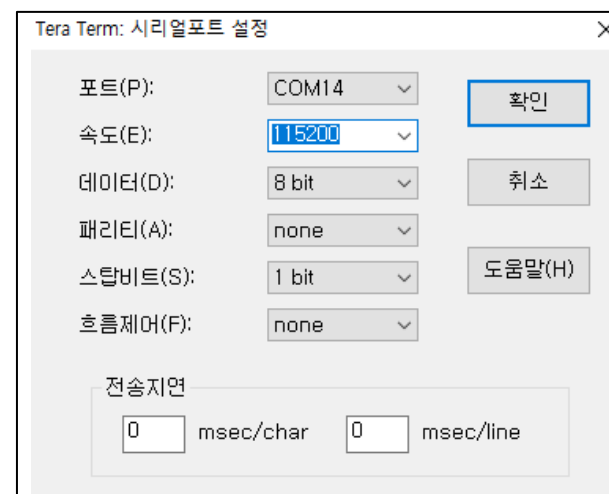
③ TeraTerm [설정] – [시리얼포트]



② TeraTerm 실행 시 시리얼 포트 선택



④ 시리얼 포트 속도 포맷 설정



Lab4-1: UART

12

- USART2 TX and RX pin
 - ▣ Pin naming in Mbed: In PinNames.h

```
// STDIO for console print
#ifdef MBED_CONF_TARGET_STDIO_UART_TX
    CONSOLE_TX = MBED_CONF_TARGET_STDIO_UART_TX,
#else
    CONSOLE_TX = PA_2,
#endif
#ifdef MBED_CONF_TARGET_STDIO_UART_RX
    CONSOLE_RX = MBED_CONF_TARGET_STDIO_UART_RX,
#else
    CONSOLE_RX = PA_3,
#endif
```

Lab4-1: UART (BufferedSerial)

13

□ A sample code

```
#include "mbed.h"

// Maximum number of element the application buffer can contain
#define MAXIMUM_BUFFER_SIZE 80
char buf[MAXIMUM_BUFFER_SIZE] ;

// Create a DigitalOutput object to toggle an LED whenever data is received.
static DigitalOut led(LED1);

// Create a BufferedSerial object with a default baud rate.
static BufferedSerial pc(CONSOLE_TX, CONSOLE_RX, 115200);

int main(void)
{
    printf("mbed-os version: %d.%d.%d\r\n",
           MBED_MAJOR_VERSION, MBED_MINOR_VERSION, MBED_PATCH_VERSION);
    sprintf(buf, "Hello New Serial function in mbed-os v.%.1f\r\n", 6.0);
    pc.write(buf, strlen(buf));

    sprintf(buf, "Enter characters...\r\n");
    pc.write(buf, strlen(buf));

    while (1) {

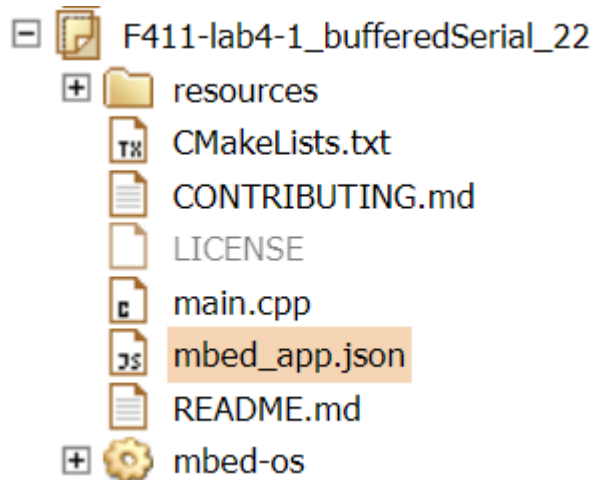
        led = !led;

        // Echo the input back to the terminal.
        int num = pc.read(buf, sizeof(buf));
        pc.write(buf, num);
        if (buf[0] == '\r') {
            pc.write("\n", 1);
        }
    }
}
```

Lab4-1: UART (BufferedSerial)

14

□ mbed_app.json



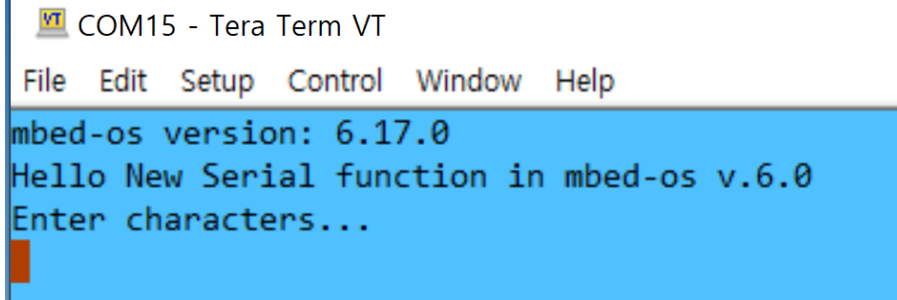
A code editor view showing the contents of the 'mbed_app.json' file. The file is open in a tab alongside 'main.cpp'. The JSON content is as follows:

```
1 {  
2     "target_overrides": {  
3         "*": {  
4             "target.printf_lib": "std"  
5         }  
6     }  
7 }
```

Lab4-1: UART

15

- 결과 (mbed-os version in 2023: 6.17.0)



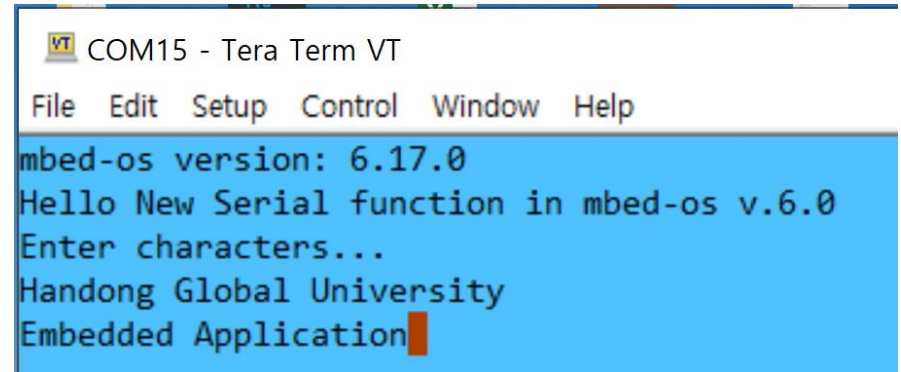
VT COM15 - Tera Term VT

File Edit Setup Control Window Help

```
mbed-os version: 6.17.0
Hello New Serial function in mbed-os v.6.0
Enter characters...

```

〈초기 화면〉



VT COM15 - Tera Term VT

File Edit Setup Control Window Help

```
mbed-os version: 6.17.0
Hello New Serial function in mbed-os v.6.0
Enter characters...
Handong Global University
Embedded Application

```

〈동작 화면〉

UART - MBED-OS -

Handong university

Jong-won Lee

UnbufferedSerial API

2

- The UnbufferedSerial class provides UART functionality with an **API similar to the BufferedSerial class**.
- Unlike the BufferedSerial class, the UnbufferedSerial class does **not use intermediary buffers** to store bytes to transmit to or read from the hardware.
- The method to read data returns only one byte for every call. However, you can write multiple bytes at once.
- You can use this class for use in interrupt handlers with the RTOS.
- Because it does not acquire a mutex lock, you must ensure only one instance uses the serial port.
- For normal blocking applications, BufferedSerial performs better than UnbufferedSerial and causes less CPU load and fewer latency issues.
- Only applications that are short of RAM and cannot afford buffering or that need more control of the serial port and use it from IRQ should use UnbufferedSerial.

UnbufferedSerial API

3

□ UnbufferedSerial class

생성자	UnbufferedSerial (PinName tx, PinName rx, int baud = MBED_CONF_PLATFORM_DEFAULT_SERIAL_BAUD_RATE)
설명	주어진 tx, rx 핀을 사용하는 UnbufferedSerial 객체를 생성한다.
사용 예	static UnbufferedSerial serial_port(CONSOLE_TX, CONSOLE_RX);
함수	void baud (int baud);
설명	UART의 baud rate를 설정함.
사용 예	serial_port.baud(115200);
함수	void format (int bits = 8, Parity parity = SerialBase::None, int stop_bits = 1)
설명	UART의 전송 포맷을 설정함.
사용 예	serial_port.format(8, SerialBase::Odd, 2);

UnbufferedSerial API

4

□ UnbufferedSerial class

함수	<code>int readable ()</code>
설명	읽을 데이터가 존재하면 '1'을 반환한다.
사용 예	<pre>If (serial_port.readable()) num = serial_port.read(buf, sizeof(buf));</pre>
함수	<code>int writable ()</code>
설명	write()를 위한 버퍼가 존재하면 '1'을 반환한다.
사용 예	<pre>If (serial_port.writable()) serial_port.write(buf, num);</pre>

UnbufferedSerial API

5

□ UnbufferedSerial class

함수	<code>ssize_t read (void * buffer, size_t size)</code>
설명	버퍼로 수신한 데이터를 읽는다. 정확하게 <u>한 바이트만</u> 읽는다. 그때까지 <u>block</u> 된다. 반환값: - 읽은 데이터의 수.
사용 예	<code>num = serial_port.read(buf, sizeof(buf));</code>

L block

UnbufferedSerial API

6

□ UnbufferedSerial class

함수	<code>ssize_t write (void * buffer, size_t size)</code>
설명	버퍼의 내용을 전송한다. 모든 데이터를 전송할 때까지 block된다. 반환값: - 쓴 데이터의 개수.
사용 예	<code>serial_port.write(buf, num);</code>

block

UnbufferedSerial API

7

□ UnbufferedSerial class

함수	void attach (Callback< void()> func, IrqType type= RxIrq)
설명	주어진 인터럽트가 발생할 때마다 인터럽트 핸들러를 부른다.
사용 예	serial_port.attach (rx_handler, SerialBase::RxIrq);

input

Lab4-3: UART (UnbufferedSerial)

8

□ 실습 목적

- ▣ UnbufferedSerial 동작을 이해한다.

□ 실습 과정

- ▣ Lab4-1과 동일한 실험 환경을 구성한다.
- ▣ 사용자의 입력이 터미널 에뮬레이터에 나타나도록 코드를 구현한다.
 - 사용자의 입력을 interrupt 방식에 의해서 감지하여라.
- ▣ 사용자가 Enter(엔터) 키를 입력할 경우에 다음 줄로 넘어가도록 한다.

Lab4-3: UART (UnbufferedSerial)

9

□ A sample code (interrupt)

```
#include "mbed.h"

// Maximum number of element the application buffer can contain
#define MAXIMUM_BUFFER_SIZE 80
char buf[MAXIMUM_BUFFER_SIZE] ;

// Create a DigitalOutput object to toggle an LED whenever data is received.
static DigitalOut led(LED1);

// Create a BufferedSerial object with a default baud rate.
static UnbufferedSerial pc(CONSOLE_TX, CONSOLE_RX);

void rx_handler()
{
    char c;

    // Toggle the LED.
    led = !led;

    // Read the data to clear the receive interrupt.
    if (pc.read(&c, 1)) {
        // Echo the input back to the terminal.
        pc.write(&c, 1);
        if (c == '\r') {
            pc.write("\n", 1);
        }
    }
}
```

Lab4-3: UART (UnbufferedSerial)

10

□ A sample code (con't)

```
int main(void)
{
    pc.baud(115200);
    pc.format(8, SerialBase::None, 1);

    // Register a callback to process a Rx (receive) interrupt.
    pc.attach(rx_handler, SerialBase::RxIrq);

    sprintf(buf, "Hello New Serial function in mbed-os v.%.1f\r\n", 6.0);
    pc.write(buf, strlen(buf));

    sprintf(buf, "Enter characters...\r\n");
    pc.write(buf, strlen(buf));

    while (1) {
    }
}
```

UART LABS - MBED-OS - LAB

Handong university

Jong-won Lee

Lab4-2: UART

2

□ 실습 목적

- ▣ BufferedSerial 동작을 이해한다.
- ▣ Sigio 의 동작을 이해한다.

□ 실습 과정

- ▣ Lab4-1과 동일한 실험 환경을 구성한다.
- ▣ 사용자의 입력이 터미널 에뮬레이터에 나타나도록 코드를 구현한다.
 - 사용자의 입력을 sigio 에 의해서 감지하여라.
- ▣ 사용자가 Enter(엔터) 키를 입력할 경우에 다음 줄로 넘어가도록 한다.

Lab4-2: UART (BufferedSerial)

3

- A skeleton code (with signal handler): complete the code
 - ▣ Must have the same function of that of Lab4-1

```
#include "mbed.h"

// Maximum number of element the application buffer can contain
#define MAXIMUM_BUFFER_SIZE 80
char buf[MAXIMUM_BUFFER_SIZE] ;

static DigitalOut led(LED1);

static BufferedSerial pc(CONSOLE_TX, CONSOLE_RX, 115200);

// RX interrupt handler
void rx_handler()
{
    led = !led; // Toggle the LED.

    // complete your code
    // ....
}
```

Lab4-2: UART (BufferedSerial)

4

- A skeleton code (with signal handler): complete the code

```
int main(void)
{
    pc.set_blocking(false); // non-critical in this example
    pc.sigio(rx_handler); //event handler

    sprintf(buf, "Hello New Serial function in mbed-os v.%.1f\r\n", 6.0);
    pc.write(buf, strlen(buf));
    sprintf(buf, "Enter characters...\r\n");
    pc.write(buf, strlen(buf));

    while (1) {
        // complete your code
        // ...

    }
}
```

Lab4-4: UART

5

- 실습 목적
 - ▣ 블루투스 모듈 사용법을 익힌다.
 - ▣ 스마트폰 앱을 이용하여 Nucleo 보드 상의 GPIO를 제어한다.
- 실습 시나리오
 - ▣ 스마트폰 앱에 입력된 글자들은 PC의 Terminal(TeraTerm) 창에 출력되어야 한다.
 - ▣ 스마트폰 앱을 통하여 주어진 명령어 따라 LED가 제어 되어야 한다.
 - RedLED on, RedLED off, RedLED status
 - GreenLED on, GreenLED off, GreenLED status
 - 첫번째 argument와 두번째 argument 사이에 space가 하나가 아니더라도 동일한 명령어로 인식되어야 한다.
 - Ex.: “RedLED on”이란 명령어도 “RedLED on”과 동일하게 동작하여야 함.

Lab4-4: UART

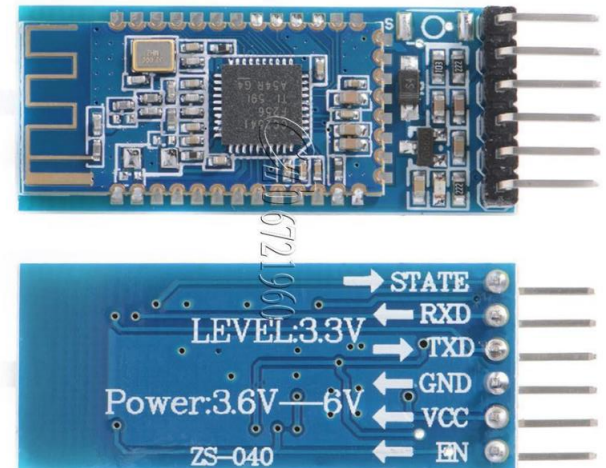
6

- 실습 시나리오 (con't)
 - ▣ 주어진 명령이 실행된 다음에 스마트폰 앱에 결과가 다음과 같이 표시되어야 한다.
 - 정상적인 명령이 주어졌을 경우에는 명령이 실행된 다음 혹은 현재의 LED의 상태가 다음과 같이 표시되어야 한다.
 - RedLED status: on 혹은 RedLED status: off
 - GreenLED status: on 혹은 GreenLED status: off
 - 잘못된 명령어가 주어졌을 경우에는 다음과 같이 표시되어야 한다.
 - Undefined command

Lab4-4: UART

7

- Bluetooth module: HM-10 (BLE 기반의 Bluetooth-serial)
 - ▣ Master 혹은 slave 기능 모두 가능함.
 - ▣ 디폴트 시리얼 통신 포맷
 - 9600 bps, No parity, Data: 8 bits, 1 stop bit, No flow control
 - ▣ Module name: nes-xx (xx: 01 ~ 99)
 - ▣ 연결할 핀
 - GND,
 - VCC (3.6V ~ 6V)
 - RXD, TXD (5V logic level)
 - 내부 BLE 모듈은 3.3V 전원으로 동작.
 - Voltage regulator와 level shifter 회로 존재.



Lab4-4: UART

8

□ 회로 구성

▣ Nucleo-F411RE 보드와 HM-10 모듈 연결 방법

- VCC (HM-10) ⇔ CN6-5 (+5V, Nucelo 보드)
- GND (HM-10) ⇔ CN6-6 (GND, Nucelo 보드)
- TXD (HM-10) ⇔ D2 (USART1 RXD, Nucelo 보드)
- RXD (HM-10) ⇔ D8 (USART1 TXD, Nucelo 보드)

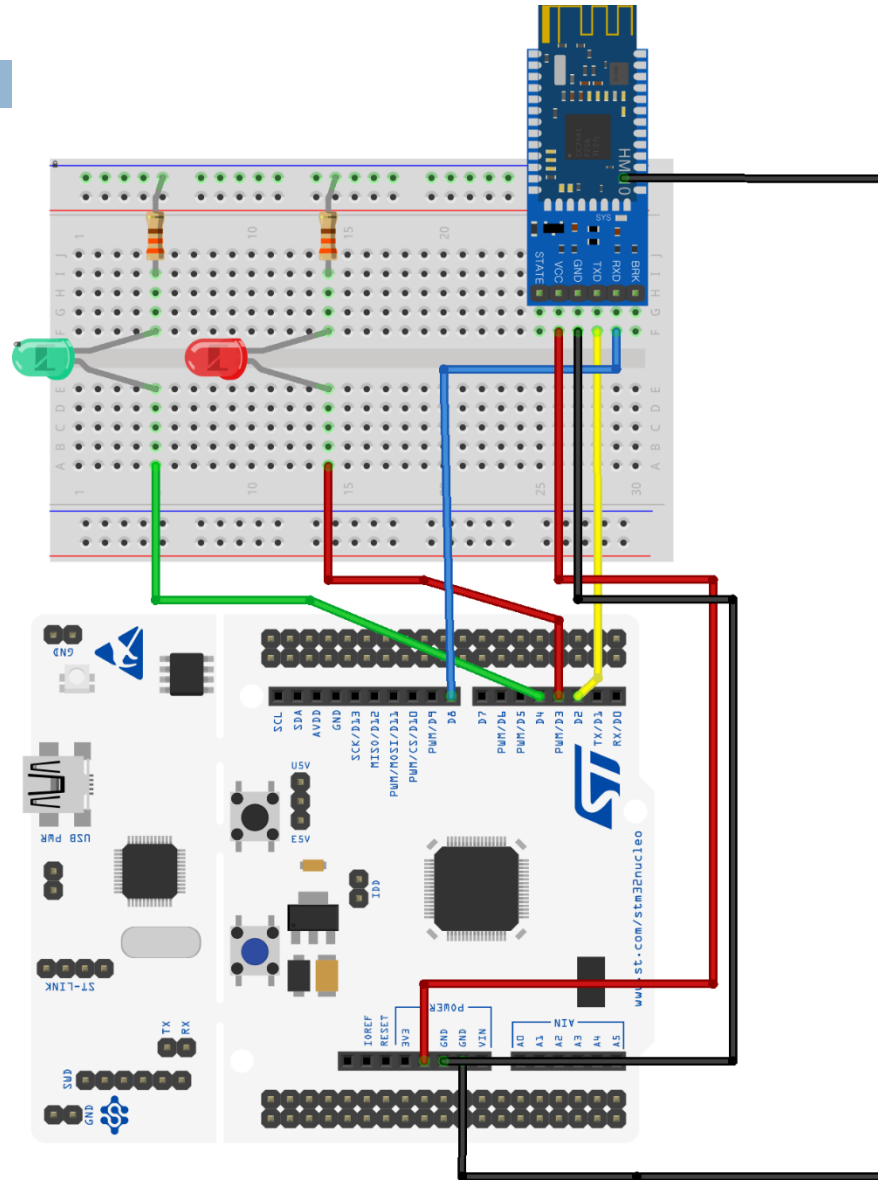
▣ LED 제어를 위한 회로 구성

- D3, D4에 전선을 연결 후 저항($220\sim330\Omega$)과 LED 연결
- D3 (Red LED), D4(Green LED)

Lab4-4: UART

9

회로 구성



fritzing

Lab4-4: UART

10

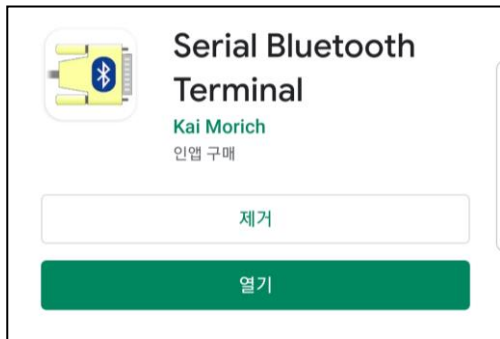
- 휴대폰에서 사용 가능한 앱
 - ▣ 안드로이드폰: **Serial Bluetooth Terminal**
 - Bluetooth Classic & BLE
 - ▣ iPhone: **HM10 Bluetooth Serial Lite**

Lab4-4: UART

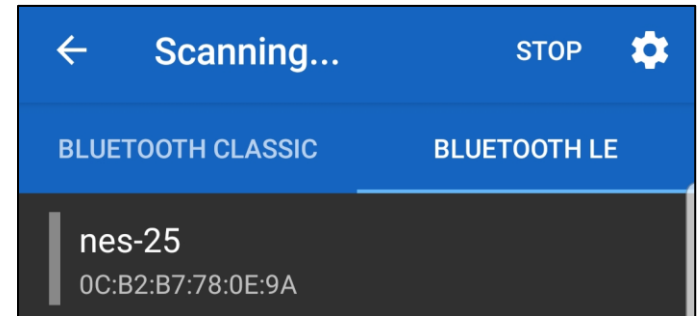
11

□ 앱 동작 과정: 안드로이드폰: **Serial Bluetooth Terminal**

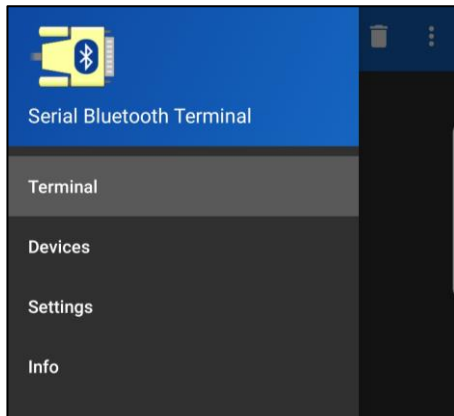
① App(Serial Bluetooth Terminal) 설치



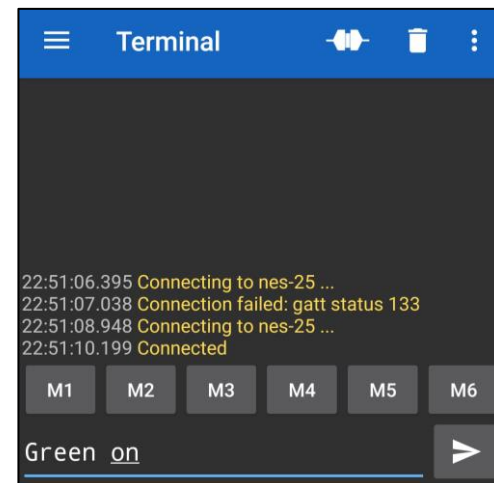
③ 우측 상단 [SCAN] 클릭



② 좌측 상단 - [Devices] 선택



④ 해당 Bluetooth module 연결



Lab4-4: UART

12

□ 동작 예

```
Terminal
21:46:00.120 ♥ Hello, HM-10!
21:46:00.301 Hello, HM-10!
21:46:46.485 RedLED on
21:46:46.591 RedLED status: on
21:46:53.329 RedLED off
21:46:53.401 RedLED status: off
21:47:03.540 RedLED status
21:47:03.631 RedLED status: off
21:47:21.769 GreenLED on
21:47:21.841 GreenLED status: on
21:47:30.054 GreenLED off
21:47:30.151 GreenLED status: off
21:47:36.487 GreenLED on
21:47:36.571 GreenLED status: on
21:47:45.207 GreenLED status
21:47:45.301 GreenLED status: on
21:48:03.919 GreenLED Toggle
21:48:03.991 Undefined command
21:48:26.636 YellowLED on
21:48:26.731 Undefined command

M1 M2 M3 M4 M5 M6 M7
YellowLED on
```

```
COM5 - Tera Term VT
File Edit Setup Control Window Help
LED Control Example by a CellPhone
LED Control Example by a CellPhone
LED Control Example by a CellPhone
RedLED on
RedLED status: on
RedLED off
RedLED status: off
RedLED status
RedLED status: off
GreenLED on
GreenLED status: on
GreenLED off
GreenLED status: off
GreenLED on
GreenLED status: on
GreenLED status
GreenLED status: on
GreenLED Toggle
Undefined command
YellowLED on
Undefined command
```