강의명: 임베디드 시스템
숙제 번호: 5
숙제 제목: Serial communication(직렬통신)
학생 이름: 정우성
학번: 201810890

1. 프로그램 uart-tx/rx

1.1 uart-tx 프로그램 코드 쓰기

```
#include "mbed.h"
Serial uart_tx(D1, D0); // uart_tx = (D1=TX, D0=RX)
DigitalIn sw(D2, PullDown); // sw = D2(PullDown)
int main()
{
while(true) {
if(sw == 1) { // if sw is on
uart_tx.putc('1'); // send '1'
thread_sleep_for(200); // wait 200 ms
}
}
}
```

1.2 uart-rx 프로그램 코드 쓰기

```
#include "mbed.h"
Serial uart_rx(D1, D0); // uart_rx = (D1=TX, D0=RX)
DigitalOut led(D2); // led = D2
int main()
{
while(true) {
if(uart_rx.getc() == '1')  // if getc() is '1'
{
led = 1; // on LED
thread_sleep_for(200); // wait 200 ms
led = 0; // off LED
}
}
}
```

1.3 프로그램 작성 아이디어 혹은 이유 설명 쓰기

먼저 Tx보드인 uart-tx는 serial이라는 class와 DigitalIn이라는 class를 사용한다. 이 DigitalIn에 객체 sw를 통해 D2의 핀으로 입력이 들어왔는지 판별하고, 들어오면 serial에 uart_tx라는 객체를 통하여 문자 '1'을 전송한다.
Rx보드는 Tx보드와 같이 serial이라는 class에 객체를 선언해 통신을 한다. DigitalOut이라는 class에 객체를 활용한다. 이 보드는 Tx보드로부터 '1'을 수신 받으면, LED를 200ms동안 켜는 작업을 수행한다.

1.4 하드웨어 구성 사진 첨부하기

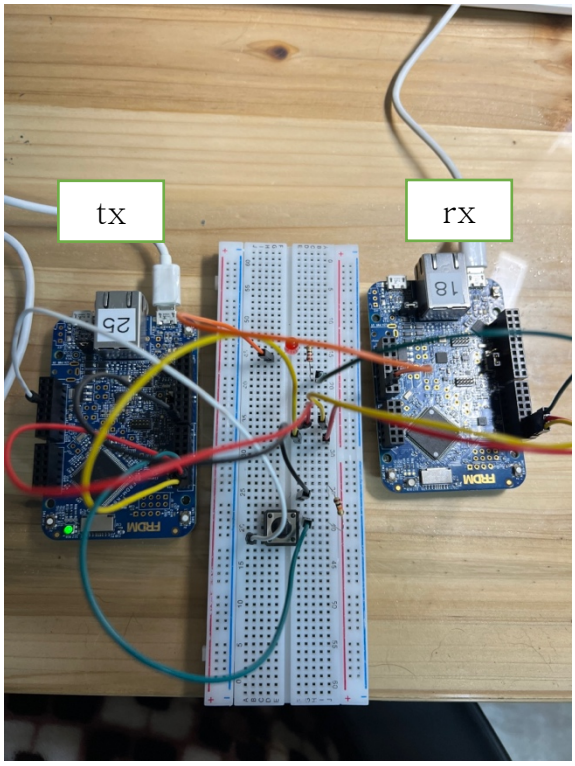

Figure 1 uart-tx/rx 하드웨어 구성사진

1.5 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtube.com/shorts/8hZQ86KUMdg

2 프로그램 spi-master/slave

2.1 spi-master 프로그램 코드 쓰기

```
#include "mbed.h"
SPI spi(PTD2, PTD3, PTD1); // spi = (MOSI, MISO, SCLK)
DigitalOut ss(PTD0, 0x01); // ss = PTD0
DigitalIn sw(D0, PullDown); // sw = D0(PullDown)
int main()
{
spi.format(8, 3); // 8-bit data, mode 3
spi.frequency(1000000); // frequency 1,000,000 Hz
while(true) {
if(sw == 1) { // if sw is on
ss = 0; // enable slave
spi.write('1'); // send '1' and receive
ss = 1; // disable slave
thread_sleep_for(200); // wait 200 ms
}
}
}
```

2.2 spi-slave 프로그램 코드 쓰기

```
#include "mbed.h"
```

```
SPISlave spi_slave(PTD2, PTD3, PTD1, PTD0); // spi_slave = (MOSI, MISO, SCLK, SS)
DigitalOut led(D0); // led = D0
int main()
{
unsigned char r;
spi_slave.format(8, 3); // 8-bit data, mode 3
spi_slave.frequency(1000000); // frequency 1,000,000 Hz
led = 0;
while(true) {
if(spi_slave.receive()) { // if spi data is ready
r = spi_slave.read(); // r = receive data
spi_slave.reply(r); // and send r
if(r == '1') { // if r == '1'
led = 1; // on LED
thread_sleep_for(200); // wait for 200 ms
led = 0; // off LED
}
}
}
}
```

2.3 프로그램 작성 아이디어 혹은 이유 설명 쓰기

Master측 코드는 SPI라는 class를 활용하여 통신한다. 통신을 하기 전 SPI에 멤버함수를 통해 mode를 3번 데이터 크기를 8bit로 설정하고, 클럭 주파수를 1000000으로 잡는다. 그 후 while문에서 switch에서 입력이 들어오면 write하는 멤버함수를 통해 '1'라는 문자를 전송하는데, 이 함수는 slave에서 '1'을 보낸 것을 수신 받는 작업까지 수행한다. 따라서 이 함수가 종료되면 ss에 다시 전압이 인가되고 통신이 종료된다.
Slave는 SPISlave라는 class에 멤버함수를 통해 데이터 수신 및 전송을 한다. 이쪽도 데이터 크기 8bit, mode는 3번, 클럭 주파수는 1000000으로 설정한다. 그 후 while문에서 receive()라는 함수를 통해 통신 상황을 판별한다. 그리고 통신을 해야 하는 상황이면 read()라는 함수를 통해 수신 받은 데이터를 얻는다. 위에서 언급했든 수신 후 전송이 필요해 reply이라는 함수로 수신 받은 데이터를 전송한다. 이 작업을 통해 master에서는 통신을 종료하고, slave는 받은 데이터가 '1'인지 판단하고 '1'이면 LED를 200ms동안 켠다.

2.4 하드웨어 구성 사진 첨부하기

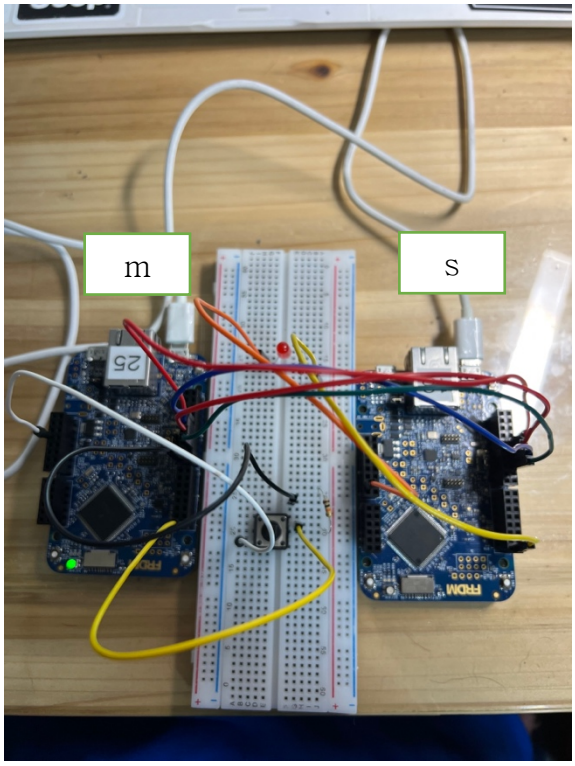

Figure 2 spi-master.slave 하드웨어 사진

2.5 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtube.com/shorts/tc19Z8kcdG8

3. 프로그램 i2c-master/slave

3.1 i2c-master 프로그램 코드 쓰기

```
#include "mbed.h"
I2C i2c(I2C_SDA, I2C_SCL); // i2c = (I2C_SDA, I2C_SCL)
DigitalIn sw(D0, PullDown); // sw = D0(PullDown)
const int addr = 0xa0; // slave addr (even no)
int main()
{
while(true) {
if(sw == 1) { // if sw is on
i2c.start(); // start condition
i2c.write(addr | 0x00); // write (addr | WRITE)
i2c.write('1'); // write '1'
i2c.stop(); // stop condition
thread_sleep_for(200); // wait 200 ms
}
}
}
```

3.2 i2c-slave 프로그램 코드 쓰기

```
#include "mbed.h"
I2CSlave i2c_slave(I2C_SDA, I2C_SCL); // i2c_slave = (I2C_SDA, I2C_SCL)
```

```
DigitalOut led(D0); // led = D0
const int addr = 0xa0; // slave addr (even no)
int main() {
int i, r;
led = 0;
i2c_slave.address(addr); // set slave address
while(true) {
i = i2c_slave.receive(); // check if my address is selected
switch(i) {
case I2CSlave::WriteAddressed: // WRITE means slave read
i2c_slave.read(); // read address skips
r = i2c_slave.read(); // r = read data
if(r == '1') { // if r == '1'
led = 1; // on led
thread_sleep_for(200); // wait 200 ms
led = 0; // off led
}
break;
default:
break;
}
}
}
```

3.3 프로그램 작성 아이디어 혹은 이유 설명 쓰기

I2C에 프로토콜은 4가지로 구성되어 1단계는 start상태 전달 2단계 주소 및 read/write판별 비트 전송, 3단계 데이터 전송 및 수신, 4단계 stop상태 전달 이 4가지 단계로 통신합니다.
Master는 I2C라는 class를 활용하여 통신한다. 통신 방법은 1단계인 start상태를 전달한다. 이것은 start()하는 I2C class에 멤버함수로 전달했다. 그 다음 2단계인 주소 전달은 미리 const int 형 변수에 담아둔 주소를 write라는 멤버함수에 인자로 넘겨 구현했다. 이때 변수에 담아둔 주소는 read/write판별 비트도 포함된 형태이며, 코드에서 0(write)로 전달한다. 그 후 3단계 데이터전달로 이는 write멤버 함수를 활용해 '1'를 전송했다. 마지막은 4단계 stop 상태전달은 stop이라는 멤버함수로 SCL를 HIGH로 올렸다. 따라서 이러한 알고리즘으로 slave 보드에 '1'이라는 데이터를 전송했으며, 이 전송 알고리즘은 switch가 눌리면 진행되게 if문으로 구성했다.
Slave는 I2CSlace라는 class를 활용하여 구현했다. Slave는 자신에 주소를 설정해야 자신에 맞는 주소일 때 통신을 한다 따라서 address라는 멤버 함수를 통해 자신의 주소를 설정한다. 그러고 while문에서 receive()라는 함수를 통해 수신 받은 주소 값을 확인한다. 그리고 이 주소가 맞으면 switch문에 case I2CSlave::WriteAddressed:가 작동해 이 아래줄이 수행된다. 따라서 다음 줄인 read()함수가 수행되어 수신 받은 정보를 얻고, '1'이면 if문이 작동해 LED가 200ms동안 켜진다.

3.4 하드웨어 구성 사진 첨부하기

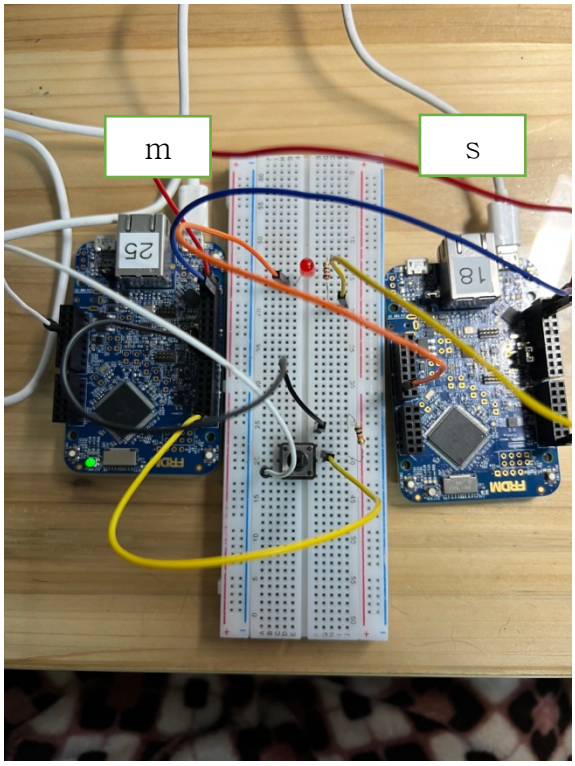

Figure 3 i2c-master/slave **하드웨어 사진**

3.5 프로그램 수행 사진/동영상(Youtube 링크) 첨부하기

https://youtube.com/shorts/mrNgu_Xd7aM

끝.