임베디드 시스템 설계 및 실험 보고서

6 주차 실험_인터럽트 방식을 활용한 GPIO 제어 및 UART 통신

분반 : 001 분반

교수님 : 정 상화 교수님

조교님 : 유 동화 조교님

실험일 : 2019-10-07

제출일 : 2019-10-14

00. 목차

7 조

장 수현

박 창조

임 다영

이 힘찬

01. 실험 목적

- Interrupt 방식을 활용한 GPIO 제어 및 UART 통신
- 라이브러리 함수 사용법 숙지

02. 실험 과제

2.1 주 과제

- Interrupt 방식을 활용해 조이스틱, 버튼입력 제어 및 오실로스코프 문자 출력

2.2 세부 과제

- 개발 환경 구축
- DS-5 에서 프로젝트 생성 및 설정
- DB 파일, 라이브러리, scatter 파일, flashclear 파일을 프로젝트 폴더 안으로 복사
- 버튼 입력을 통해 Putty 로 문자 출력 및 오실로스코프로 확인
- 조이스틱을 이용해 LED 물결 방향 조절 및 Putty 를 통해 입력 받은 숫자 LED 점등 제외

03. 실험 준비

3.1 실험에 필요한 기초지식

3.1.1 Poling

- 특정 주기를 가지고 주기마다 처리를 위한 시그널이 들어왔는지 체크하는 방식
- 커널과 같은 Interrupt handler 가 필요하지 않다.
- Interrupt 에 비해 구현이 쉽지만 시스템의 리소스를 많이 차지하여 성능 저하의 원인이 되기도 한다.

3.1.2 Interrupt

- 특정 주기를 가지고 계속 입력을 확인하는 것이 아니라, 인터럽트가 들어오면 해당 작업을 수행하고 다시 돌아오는 방식
- Interrupt handler 가 필요하다.

3.1.3 H/W Interrupt

- 비동기식 이벤트 처리로 주변장치의 요청에 의해 발생하는 Interrupt

3.1.4 S/W Interrupt

- 동기식 이벤트 처리로 사용자가 프로그램 내에서 Interrupt 가 발생하도록 설정하는 Interrupt

3.1.5 UART

- 범용 비동기화 송수신기
- 병렬 데이터를 직렬 형식으로 전환하여 데이터를 전송하는 컴퓨터 하드웨어의 일종
- 시리얼 기반 통신으로 RS_232 를 통해 통신을 지원

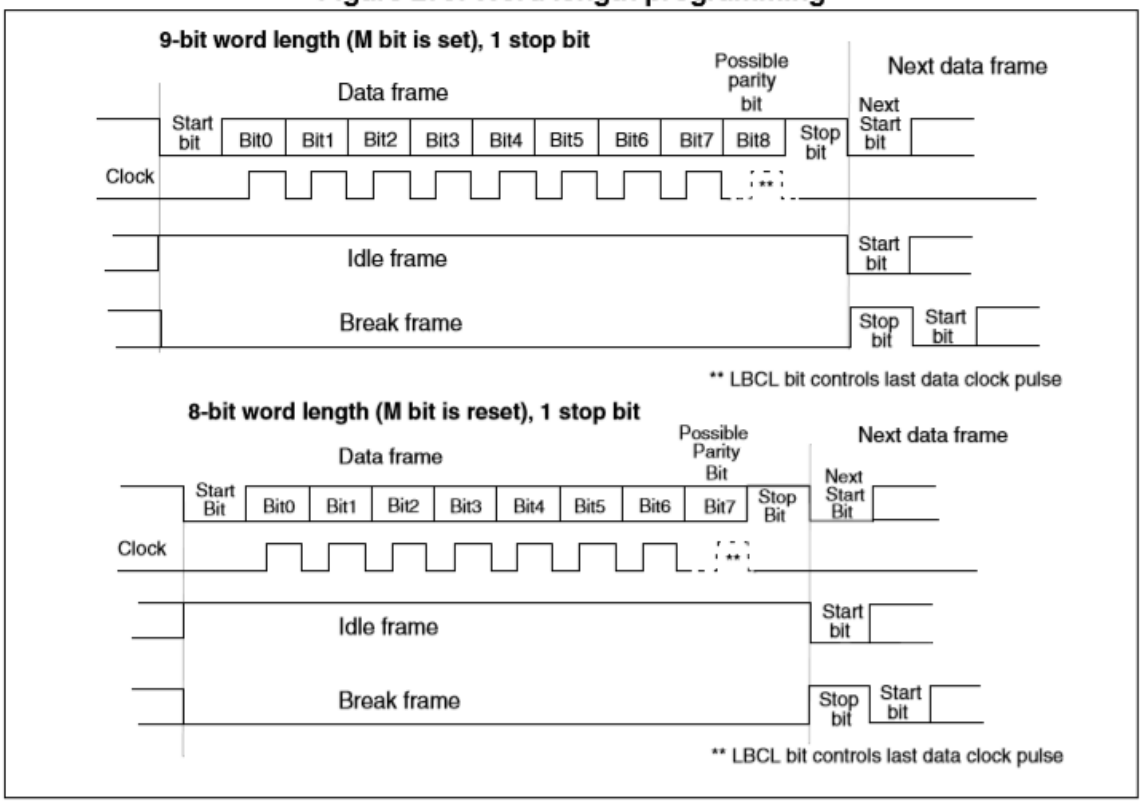
3.1.6 USART

- 범용 동기화 송수신기
- 동기화 통신까지 지원하는 UART

3.1.7 Data frame

- Start bits : 통신의 시작을 의미하는 것으로 0 으로 설정
- Data bits : 송/수신되는 데이터를 8-9bit 으로 나타낸다.
- Parity bits : 오류 검증을 위한 값으로 레지스터 설정에 따라 짝/홀/사용안함 으로 선택
- Stop bits : 통신 종료를 의미하는 것으로 레지스터 설정에 따라 비트 수가 나뉜다.
- Baud rate : 초당 얼마나 많은 심볼을 전송할 수 있는가를 표현, 초당 신호(signal)요소의 수

Figure 279. Word length programming



3.1.8 NVIC

- Cortex-M3 에서 중첩된 인터럽트를 제어
- Preemption priority – sub priority 순으로 우선순위를 결정하며, 숫자가 작을수록 우선수위가 큼
- Interrupt Handler 를 호출하여 Interrupt 처리가 가능

3.1.9 EXTI

- 외부에서 신호가 입력될 경우 device 에 event./Interrupt 가 발생되는 기능
- 입력 받을 수 있는 신호는 Rising/ Falling edge 를 각각 또는 같이 받을 수 있음
- event / interrupt mode 로 설정 가능하며 interrupt mode 로 설정해야 interrupt handler 를 통해 처리 가능

3.2 실험 진행 시 주의사항

- Project – Properties – Settings – Arm Linker 5 - General의 Image entry point 를 공백으로 설정할 것

- 헤더파일 선언할 때 꼭 #include “core_cm3.h” 포함할 것

- SystemInit – RCC_configure – GPIO_Configure – UART_Configure – EXTI_Configure – NVIC_Configure 순서로 함수를 호출해야 한다.

- RCC Configuration 을 구현할 때, AFIO 가 Interrupt 에 관여하기 때문에 함께 ENABLE 시켜주어야 한다.

- 직접 주소를 지정하지 않고 헤더파일에 구현되어 있는 구조체, 상수 사용할 것

- 납땜 시 인두가 뜨거울 때 손 데이지 않게 조심

- 납땜 중 소량의 납이 튈 수 있음 주의

3.3 환경 설정

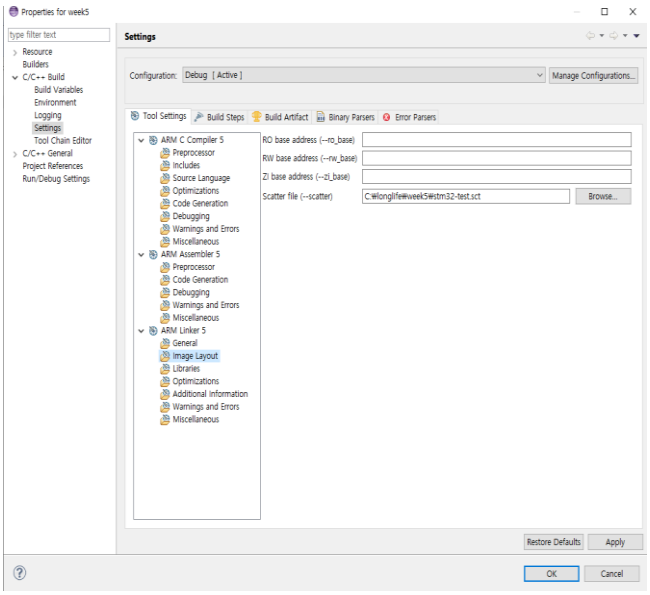
실험을 진행하기 전 아래의 프로젝트 생성 및 기본 환경 설정을 해주어야 한다.

3.3.1 프로젝트 생성

C 언어로 소스코드를 작성하기 때문에 C++프로젝트가 아닌 C 프로젝트를 생성해준다. Project type 은 Bare-metal Executable -〉 Empty Project 로, Toolchains 은 Arm Compiler 5 로 설정한다.

3.3.2 프로젝트 Properties-Settings

4 주차 실험과 동일하게 스캐터 파일을 이용한다.



3.3.3 보드 연결

보드 연결 시에는 반드시 연결 순서를 지키고, 규격에 맞는 전원선을 사용해야한다.

연결 순서 : 보드와 Dstream JTAG 연결 -〉 보드전원선 연결(보드 전원은 OFF) -〉 Dstream 전원 연결 및 ON -〉 Dstream Status LED 점등 확인 후 보드 전원 ON -〉 Dstream Target LED 점등 확인 후 DS-5 에서 ‘connect target’

3.3.4 데이터 베이스 설정

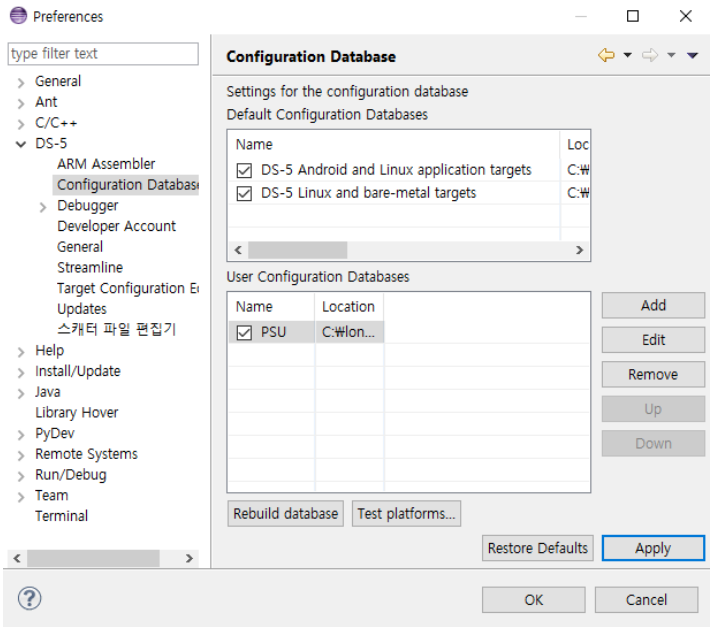
Dstream 를 USB 로 컴퓨터에 연결하고 Debug Hardware config 에서 보드를 connect 한다. 보드 연결 후에는 Auto Configure 를 클릭하여 설정을 진행하고, rvc 파일로 저장한다. rvc 파일 저장 경로에 한글이 들어가지 않도록 주의한다.

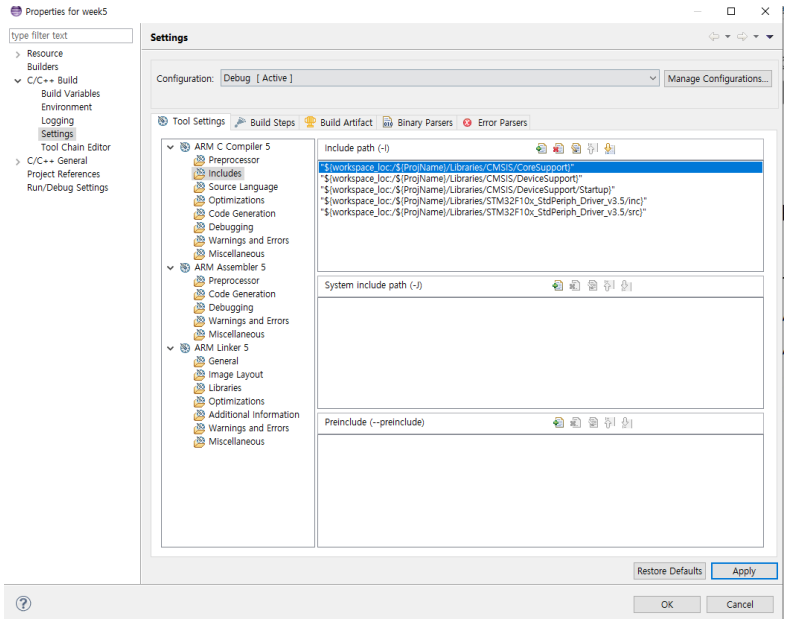
3.3.5 cdbimporter

DS-5 Command Prompt 에서 RVC 파일이 있는 폴더로 이동 후 아래와 같이 명령문을 작성한다.

3.3.6 데이터 베이스 등록 및 라이브러리 추가

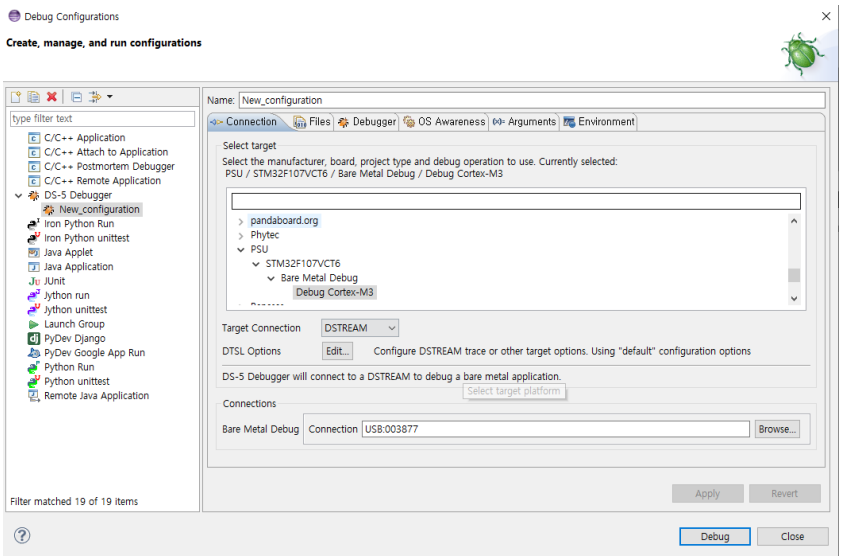
아래와 같이 데이터 베이스 등록 후 프로젝트 폴더에 라이브러리를 추가한다.

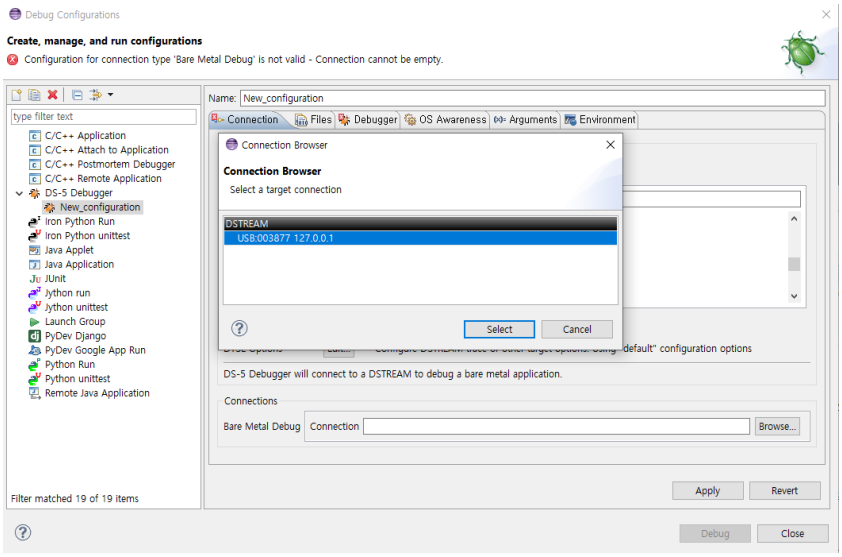


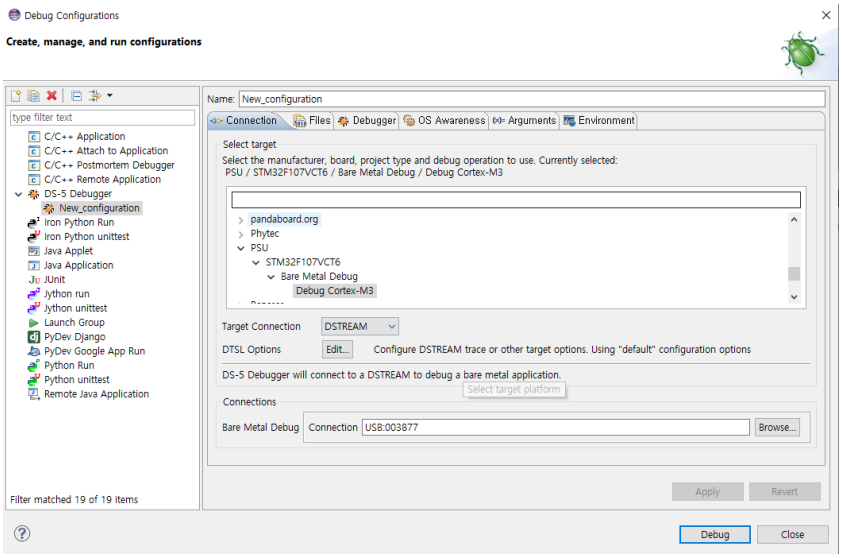


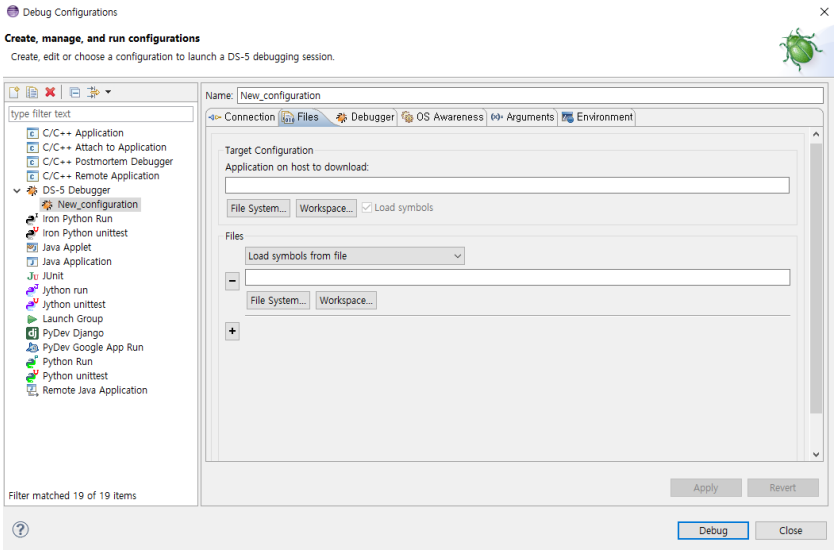
3.3.8 디버그 설정

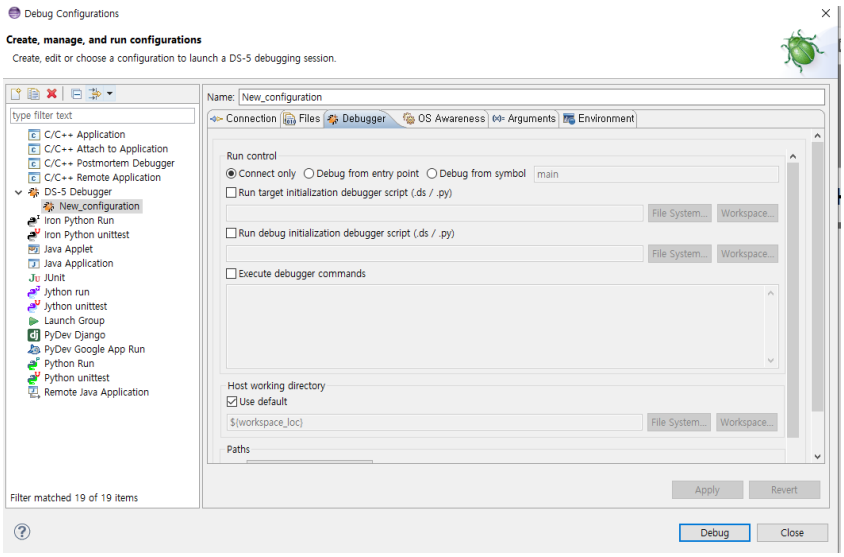
C 언어 소스파일을 만들어 빌드하고 디버그 설정을 한다. 4 주차와 동일한 환경으로 진행한다.






Debug Configurations
Create, manage, and run configurations
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Select target
Select the manufacturer, board, project type and debug operation to use. Currently selected:
PSU / STM32F107VCT6 / Bare Metal Debug / Debug Cortex-M3
pandaboard.org
Phytec
PSU
STM32F107VCT6
Bare Metal Debug
Debug Cortex-M3
Target Connection
DSTREAM
DTSL Options
Edit...
Configure DSTREAM trace or other target options. Using "default" configuration options
DS-5 Debugger will connect to a DSTREAM to debug a bare metal application.
Select target platform
Connections
Bare Metal Debug
Connection
USB:003877
Browse...
Apply
Revert
Debug
Close


Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch a DS-5 debugging session.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Target Configuration
Application on host to download:
File System...
Workspace...
Load symbols
Files
Load symbols from file
File System...
Workspace...
Apply
Revert
Debug
Close


Debug Configurations
Create, manage, and run configurations
Create, edit or choose a configuration to launch a DS-5 debugging session.
type filter text
C/C++ Application
C/C++ Attach to Application
C/C++ Postmortem Debugger
C/C++ Remote Application
DS-5 Debugger
New_configuration
Iron Python Run
Iron Python unittest
Java Applet
Java Application
JUnit
Jython run
Jython unittest
Launch Group
PyDev Django
PyDev Google App Run
Python Run
Python unittest
Remote Java Application
Filter matched 19 of 19 items
Name: New_configuration
Connection
Files
Debugger
OS Awareness
Arguments
Environment
Run control
Connect only
Debug from entry point
Debug from symbol
main
Run target initialization debugger script (.ds / .py)
File System...
Workspace...
Run debug initialization debugger script (.ds / .py)
Execute debugger commands
Host working directory
Use default
${workspace_loc}
Paths
Apply
Revert
Debug
Close

04. 실험 및 과제 해결

4.1 RCC Configuration

```
void RCC_Configure() {
      RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
      /*TODO : APB2PeriphClockEnable */
     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOC|RCC_APB2Periph_GP
IOD|RCC_APB2Periph_USART1, ENABLE);
}
```

- Usart ,조이스틱(포트 C), LED(포트 D), Usart(포트 A), 버튼(포트 D)를 모두 ENABLE 해준다.

4.2 GPIO Configuration

```
void GPIO_Configure() {

      GPIO_InitTypeDef GPIO_InitStructure;

      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
      GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 | GPIO_Pin_7);
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(GPIOD, &GPIO_InitStructure);

      /*TODO: USART1, JoyStick Config */
      GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_5 );
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
      GPIO_Init(GPIOC, &GPIO_InitStructure);

      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
      GPIO_Init(GPIOA, &GPIO_InitStructure);

      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
      GPIO_Init(GPIOA, &GPIO_InitStructure);

      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
      GPIO_Init(GPIOD, &GPIO_InitStructure);

      /*TODO: GPIO EXTILineConfig*/
      GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource2);
      GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource5);
      GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);
}
```

```
typedef struct
{
  uint16_t GPIO_Pin;             /*!< Specifies the GPIO pins to be configured.
                                      This parameter can be any value of @ref GPIO_pins_define */

  GPIOSpeed_TypeDef GPIO_Speed;  /*!< Specifies the speed for the selected pins.
                                      This parameter can be a value of @ref GPIOSpeed_TypeDef */

  GPIOMode_TypeDef GPIO_Mode;    /*!< Specifies the operating mode for the selected pins.
                                      This parameter can be a value of @ref GPIOMode_TypeDef */
}GPIO_InitTypeDef;
```

- LED 핀 2,3,4,7 을 Mode 및 Speed 설정
- 조이스틱은 INPUT 으로 설정한다. 위, 아래 만 사용하므로 pin 2, 5.
- USART 는 Input, Output 으로 각각 설정한다(Rx, Tx). 그리고 버튼도 Input 으로 설정한다.
- EXTILineConfig 로 Interrupt 걸 핀 소스를 설정한다. 조이스틱과 버튼(PD11)만 입력(Interrupt)를 줄 것이므로 Pinsource 2, 5, 11

4.3 USART Configuration

```
void USART_Configure() {
	USART_InitTypeDef USART_InitStructure;

	/*TODO: USART1 configuration*/
	USART_InitStructure.USART_BaudRate = 9600;
	USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
	USART_InitStructure.USART_Parity = USART_Parity_Even;
	USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
	USART_InitStructure.USART_WordLength = USART_WordLength_9b;
	USART_InitStructure.USART_StopBits = USART_StopBits_1;
	USART_Init(USART1, &USART_InitStructure);

	/*TODO: USART1 cmd ENABLE*/
	USART_Cmd(USART1,ENABLE);

	/*TODO: USART1 IT Config*/
	USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);
```

```
typedef struct
{
  uint32_t USART_BaudRate;            /*!< This member configures the USART communication baud rate.
                                           The baud rate is computed using the following formula:
                                            - IntegerDivider = ((PCLKx) / (16 * (USART_InitStruct->USART_BaudRate)))
                                            - FractionalDivider = ((IntegerDivider - ((u32) IntegerDivider)) * 16) + 0.5 */

  uint16_t USART_WordLength;          /*!< Specifies the number of data bits transmitted or received in a frame.
                                           This parameter can be a value of @ref USART_Word_Length */

  uint16_t USART_StopBits;            /*!< Specifies the number of stop bits transmitted.
                                           This parameter can be a value of @ref USART_Stop_Bits */

  uint16_t USART_Parity;              /*!< Specifies the parity mode.
                                           This parameter can be a value of @ref USART_Parity
                                           @note When parity is enabled, the computed parity is inserted
                                                 at the MSB position of the transmitted data (9th bit when
                                                 the word length is set to 9 data bits; 8th bit when the
                                                 word length is set to 8 data bits). */

  uint16_t USART_Mode;                /*!< Specifies wether the Receive or Transmit mode is enabled or disabled.
                                           This parameter can be a value of @ref USART_Mode */

  uint16_t USART_HardwareFlowControl; /*!< Specifies wether the hardware flow control mode is enabled
                                           or disabled.
                                           This parameter can be a value of @ref USART_Hardware_Flow_Control */
} USART_InitTypeDef;
```

- 과제에 제시한 설정대로 Parity 는 Even, Wordlength 는 9bit, HardwareFlowControl 은 None 으로 설정했다.
- Baudrate, Stopbits 는 default 로 9600, 1 비트를 주었다.
- 이번 실험은 Rx, Tx 모두 사용하므로 모드에 두 값을 모두 주었다.
- 그 후 Usart Init, Cmd Enable, IT configuration 을 순서대로 진행한다.

4.4 EXTI Configuration

```
void EXTI_Configure() {
	/*TODO: EXTI configuration [ mode interrupt ] [Trigger_falling] */
	EXTI_InitTypeDef EXTI_InitStructure;

	EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
	EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
	EXTI_InitStructure.EXTI_Line = EXTI_Line11;
	EXTI_InitStructure.EXTI_LineCmd = ENABLE;
	EXTI_Init(&EXTI_InitStructure);

	EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
	EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
	EXTI_InitStructure.EXTI_Line = EXTI_Line2;
	EXTI_InitStructure.EXTI_LineCmd = ENABLE;
	EXTI_Init(&EXTI_InitStructure);

	EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
	EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
	EXTI_InitStructure.EXTI_Line = EXTI_Line5;
	EXTI_InitStructure.EXTI_LineCmd = ENABLE;
	EXTI_Init(&EXTI_InitStructure);
}
```

```
typedef struct
{
  uint32_t EXTI_Line;               /*!< Specifies the EXTI lines to be enabled or disabled.
                                       This parameter can be any combination of @ref EXTI_Lines */

  EXTIMode_TypeDef EXTI_Mode;       /*!< Specifies the mode for the EXTI lines.
                                       This parameter can be a value of @ref EXTIMode_TypeDef */

  EXTITrigger_TypeDef EXTI_Trigger; /*!< Specifies the trigger signal active edge for the EXTI lines.
                                       This parameter can be a value of @ref EXTIMode_TypeDef */

  FunctionalState EXTI_LineCmd;     /*!< Specifies the new state of the selected EXTI lines.
                                       This parameter can be set either to ENABLE or DISABLE */
}EXTI_InitTypeDef;
```

- 이번 실험에서 조이스틱 위, 아래, 문자 'M'을 보내는 버튼 이 3 가지 인터럽트에 대해서 다룰 것이므로 3 가지 인터럽트만 설정해서 Init 한다.
- 각 인터럽트 라인을 설정하고, 모드는 이벤트 모드가 아닌 인터럽트 모드로 설정한다.
- 트리거는 Rising 과 Falling 방식을 설정하고, 라인 커맨드를 Enable 한다.

4.5 NVIC Configuration

```
void NVIC_Configure() {
	/*TODO: NVIC_configuration */

	NVIC_InitTypeDef NVIC_InitStructure;

	NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&NVIC_InitStructure);

	NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&NVIC_InitStructure);

	NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&NVIC_InitStructure);

	NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
	NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
	NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
	NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
	NVIC_Init(&NVIC_InitStructure);
}
```

```
/******  STM32 specific Interrupt Numbers *******************************************************/
  WWDG_IRQn                   = 0,      /*!< Window WatchDog Interrupt                                */
  PVD_IRQn                    = 1,      /*!< PVD through EXTI Line detection Interrupt                */
  TAMPER_IRQn                 = 2,      /*!< Tamper Interrupt                                         */
  RTC_IRQn                    = 3,      /*!< RTC global Interrupt                                     */
  FLASH_IRQn                  = 4,      /*!< FLASH global Interrupt                                   */
  RCC_IRQn                    = 5,      /*!< RCC global Interrupt                                     */
  EXTI0_IRQn                  = 6,      /*!< EXTI Line0 Interrupt                                     */
  EXTI1_IRQn                  = 7,      /*!< EXTI Line1 Interrupt                                     */
  EXTI2_IRQn                  = 8,      /*!< EXTI Line2 Interrupt                                     */
  EXTI3_IRQn                  = 9,      /*!< EXTI Line3 Interrupt                                     */
  EXTI4_IRQn                  = 10,     /*!< EXTI Line4 Interrupt                                     */
  DMA1_Channel1_IRQn          = 11,     /*!< DMA1 Channel 1 global Interrupt                          */
  DMA1_Channel2_IRQn          = 12,     /*!< DMA1 Channel 2 global Interrupt                          */
  DMA1_Channel3_IRQn          = 13,     /*!< DMA1 Channel 3 global Interrupt                          */
  DMA1_Channel4_IRQn          = 14,     /*!< DMA1 Channel 4 global Interrupt                          */
  DMA1_Channel5_IRQn          = 15,     /*!< DMA1 Channel 5 global Interrupt                          */
  DMA1_Channel6_IRQn          = 16,     /*!< DMA1 Channel 6 global Interrupt                          */
  DMA1_Channel7_IRQn          = 17,     /*!< DMA1 Channel 7 global Interrupt                          */
```

- EXTI 0~4 까지는 개별로 번호가 있지만 다른 번호는 아래 사진에서처럼 묶여져있다.

```
#ifdef STM32F10X_LD
  ADC1_2_IRQn                 = 18,     /*!< ADC1 and ADC2 global Interrupt                           */
  USB_HP_CAN1_TX_IRQn         = 19,     /*!< USB Device High Priority or CAN1 TX Interrupts           */
  USB_LP_CAN1_RX0_IRQn        = 20,     /*!< USB Device Low Priority or CAN1 RX0 Interrupts           */
  CAN1_RX1_IRQn               = 21,     /*!< CAN1 RX1 Interrupt                                       */
  CAN1_SCE_IRQn               = 22,     /*!< CAN1 SCE Interrupt                                       */
  EXTI9_5_IRQn                = 23,     /*!< External Line[9:5] Interrupts                            */
  TIM1_BRK_IRQn               = 24,     /*!< TIM1 Break Interrupt                                     */
  TIM1_UP_IRQn                = 25,     /*!< TIM1 Update Interrupt                                    */
  TIM1_TRG_COM_IRQn           = 26,     /*!< TIM1 Trigger and Commutation Interrupt                   */
  TIM1_CC_IRQn                = 27,     /*!< TIM1 Capture Compare Interrupt                           */
  TIM2_IRQn                   = 28,     /*!< TIM2 global Interrupt                                    */
  TIM3_IRQn                   = 29,     /*!< TIM3 global Interrupt                                    */
  I2C1_EV_IRQn                = 31,     /*!< I2C1 Event Interrupt                                     */
  I2C1_ER_IRQn                = 32,     /*!< I2C1 Error Interrupt                                     */
  SPI1_IRQn                   = 35,     /*!< SPI1 global Interrupt                                    */
  USART1_IRQn                 = 37,     /*!< USART1 global Interrupt                                  */
  USART2_IRQn                 = 38,     /*!< USART2 global Interrupt                                  */
  EXTI15_10_IRQn              = 40,     /*!< External Line[15:10] Interrupts                          */
  RTCAlarm_IRQn               = 41,     /*!< RTC Alarm through EXTI Line Interrupt                    */
  USBWakeUp_IRQn              = 42      /*!< USB Device WakeUp from suspend through EXTI Line Ir
#endif /* STM32F10X_LD */

#ifdef STM32F10X_LD_VL
  ADC1_IRQn                   = 18,     /*!< ADC1 global Interrupt                                    */
  EXTI9_5_IRQn                = 23,     /*!< External Line[9:5] Interrupts                            */
  TIM1_BRK_TIM15_IRQn         = 24,     /*!< TIM1 Break and TIM15 Interrupts                          */
  TIM1_UP_TIM16_IRQn          = 25,     /*!< TIM1 Update and TIM16 Interrupts                         */
  TIM1_TRG_COM_TIM17_IRQn     = 26,     /*!< TIM1 Trigger and Commutation and TIM17 Interrupt         */
  TIM1_CC_IRQn                = 27,     /*!< TIM1 Capture Compare Interrupt                           */
  TIM2_IRQn                   = 28,     /*!< TIM2 global Interrupt                                    */
  TIM3_IRQn                   = 29,     /*!< TIM3 global Interrupt                                    */
  I2C1_EV_IRQn                = 31,     /*!< I2C1 Event Interrupt                                     */
  I2C1_ER_IRQn                = 32,     /*!< I2C1 Error Interrupt                                     */
  SPI1_IRQn                   = 35,     /*!< SPI1 global Interrupt                                    */
  USART1_IRQn                 = 37,     /*!< USART1 global Interrupt                                  */
  USART2_IRQn                 = 38,     /*!< USART2 global Interrupt                                  */
  EXTI15_10_IRQn              = 40,     /*!< External Line[15:10] Interrupts                          */
  RTCAlarm_IRQn               = 41,     /*!< RTC Alarm through EXTI Line Interrupt                    */
  CEC_IRQn                    = 42,     /*!< HDMI-CEC Interrupt                                       */
  TIM6_DAC_IRQn               = 54,     /*!< TIM6 and DAC underrun Interrupt                          */
  TIM7_IRQn                   = 55      /*!< TIM7 Interrupt                                           */
#endif /* STM32F10X_LD_VL */
```

```
typedef struct
{
  uint8_t NVIC_IRQChannel;                    /*!< Specifies the IRQ channel to be enabled or disabled.
                                                   This parameter can be a value of @ref IRQn_Type
                                                   (For the complete STM32 Devices IRQ Channels list, please
                                                    refer to stm32f10x.h file) */

  uint8_t NVIC_IRQChannelPreemptionPriority;  /*!< Specifies the pre-emption priority for the IRQ channel
                                                   specified in NVIC_IRQChannel. This parameter can be a value
                                                   between 0 and 15 as described in the table @ref NVIC_Priority_Table */

  uint8_t NVIC_IRQChannelSubPriority;         /*!< Specifies the subpriority level for the IRQ channel specified
                                                   in NVIC_IRQChannel. This parameter can be a value
                                                   between 0 and 15 as described in the table @ref NVIC_Priority_Table */

  FunctionalState NVIC_IRQChannelCmd;         /*!< Specifies whether the IRQ channel defined in NVIC_IRQChannel
                                                   will be enabled or disabled.
                                                   This parameter can be set either to ENABLE or DISABLE */
} NVIC_InitTypeDef;
```

- 앞 서 정의했던 3 가지 인터럽트에 대해서 우선순위를 정해야한다. NVIC 가 이런 역할을 한다.
- 먼저 설정했던 핀소스 번호에 맞는 채널들을 설정해주고 채널 커맨드는 Enable 한다.
- PreemptionPriority 가 주 우선순위고, Subpriority 가 보조 우선순위라고 생각하면 된다.
- 위와 같이 모두 0 을 주면 세 인터럽트 사이에 우선순위는 없는 것이나 마찬가지이다.

4.6 Interrupt Handler

```
void EXTI_DeInit(void);
void EXTI_Init(EXTI_InitTypeDef* EXTI_InitStruct);
void EXTI_StructInit(EXTI_InitTypeDef* EXTI_InitStruct);
void EXTI_GenerateSWInterrupt(uint32_t EXTI_Line);
FlagStatus EXTI_GetFlagStatus(uint32_t EXTI_Line);
void EXTI_ClearFlag(uint32_t EXTI_Line);
ITStatus EXTI_GetITStatus(uint32_t EXTI_Line);
void EXTI_ClearITPendingBit(uint32_t EXTI_Line);
```

```
void EXTI15_10_IRQHandler(void) {
            if(EXTI_GetITStatus(EXTI_Line11) != RESET)
             USART_SendData(USART1,'M');
            EXTI_ClearITPendingBit(EXTI_Line11);
}
```

- 인터럽트는 모두 인터럽트 핸들러를 통해 처리가 가능한데, 먼저 문자 'M'을 출력하는 버튼의 인터럽트 핸들러다.
- 핀소스 11 번을 통해 인터럽트가 들어온경우, 상태를 확인한후 Usart 를 통해 문자 'M'을 전송한다.
- Putty 에 연결해서 문자가 제대로 출력되는지 확인할 수 있다.
- ClearITPendingBit 함수를 통해 핀소스 11 번 인터럽트 플래그를 클리어 해준다.

```
void EXTI9_5_IRQHandler(void) {
      if(EXTI_GetITStatus(EXTI_Line5) != RESET)
            flagStick=1;
      EXTI_ClearITPendingBit(EXTI_Line5);
}

void EXTI2_IRQHandler(void) {
```

```
	if(EXTI_GetITStatus(EXTI_Line2) != RESET)
		flagStick=0;
	EXTI_ClearITPendingBit(EXTI_Line2);
}
```

- 조이스틱 Up, Down 에 관련된 Interrupt Handler 이다.
- 조이스틱이 Up 이라면 flagstick=1, Down 이라면 flagstick=0 이다.

```
void USART1_IRQHandler(void) {
	char c;
	if(USART_GetITStatus(USART1, USART_IT_RXNE ) != RESET) {
		c = (char)USART_ReceiveData(USART1);
		flagNum[c-'1'] = 1;
	}
	USART_ClearITPendingBit(USART1,USART_IT_RXNE);
}
```

- Putty 에 숫자 1~4 중 하나를 입력하면 USART_ReceiveData 로 숫자를 입력받아 해당 LED flag 를 1 로 처리한다.

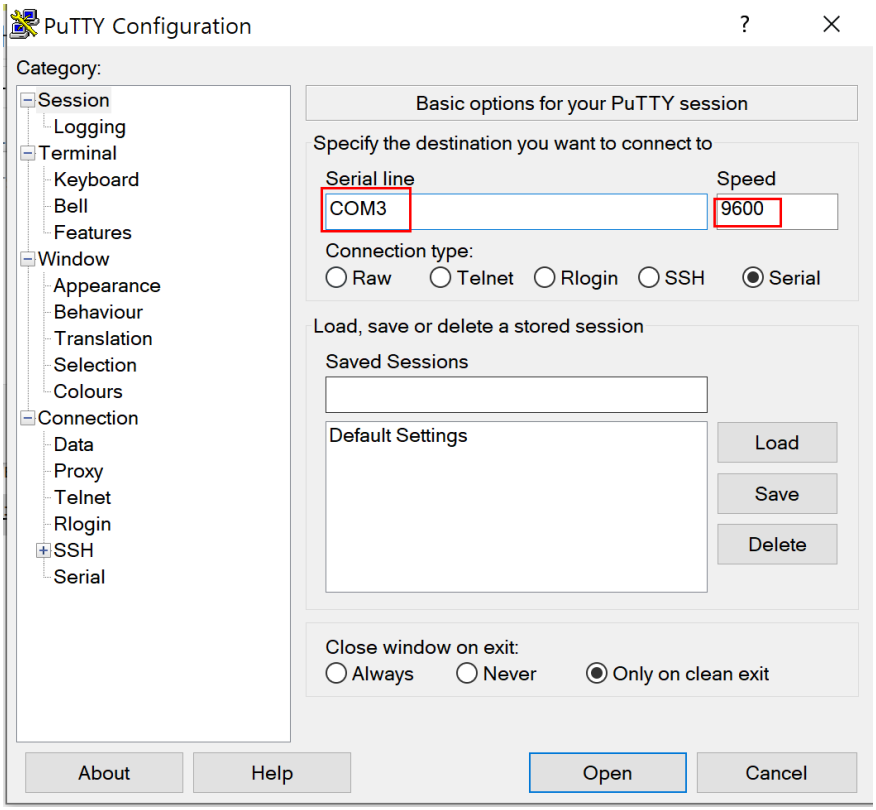
4.7 Main & delay function

```
void delay(void) {
	int i=0;
	for(i=0;i<1000000;++i);
}

int main()
{
	int i=0;

	flagStick = 2;
	for(i=0;i<4;++i) flagNum[i] = 0;
	pinLED[0] = GPIO_Pin_2;
	pinLED[1] = GPIO_Pin_3;
	pinLED[2] = GPIO_Pin_4;
	pinLED[3] = GPIO_Pin_7;

	SystemInit();
	RCC_Configure();
	GPIO_Configure();
	USART_Configure();
	EXTI_Configure();
	NVIC_Configure();

	while(1) {
		if(flagStick==1) {                    // up
```

```
                for(i=0;i<4 && flagStick==1 ;++i) {
                        if(pinLED[i]==1) continue;
                        GPIO_SetBits(GPIOD,pinLED[i]);
                        delay();
                        GPIO_ResetBits(GPIOD,pinLED[i]);
                }
        }
        if(flagStick==0) {                      // down
                for(i=3;i>=0 && flagStick==0 ;--i) {
                        if(pinLED[i]==1) continue;
                        GPIO_SetBits(GPIOD,pinLED[i]);
                        delay();
                        GPIO_ResetBits(GPIOD,pinLED[i]);
                }
        }
    }
}
```

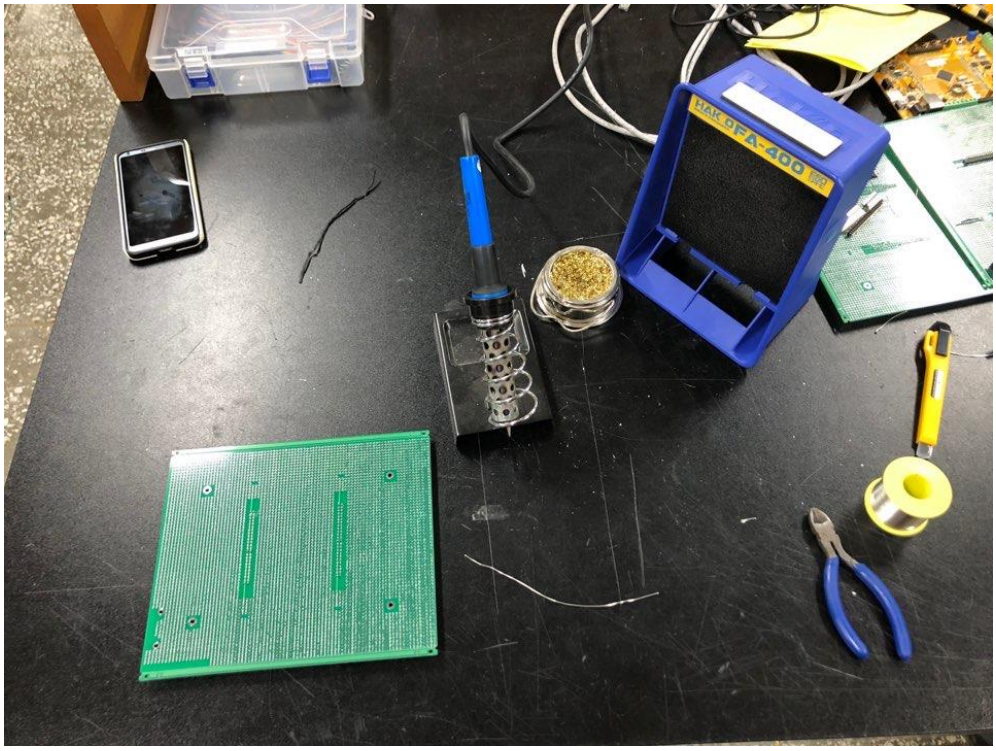
- flagstick 은 2 로 초기화 시키고 나머지 flag 값들은 모두 0 으로 초기화한다.
- up 일 때 LED 를 순서대로 켜고, Putty 를 통해 입력받은 숫자 LED 핀은 제외하고 킨다.

4.8 Putty setting



UART 통신을 사용하여 Putty 에 문자열을 출력하기 위해 Putty 를 다운로드하고 위와 같이 설정해준다. Serial Line 에는 보드와 연결된 COM3 port 를 설정해주고, Speed 에는 baud rate 인 9600 을 설정한다. Serial 통신을 할 것이기 때문에 Connection type 은 Serial 로 설정한다.

4.9 납땜



초록색 만능기판에 제공 받은 핀을 하얀 표시 부분에 따라 크기에 맞게 조절 후 납땜한다. 이 후 실험이나 텀 프로젝트에서 중요하게 쓰인다. 납땜을 하다보면 인두에 납 잔해가 뭉쳐서 수세미에 인두를 문질러서 제거한다. 이 때 납이 튀지 않도록 조심해야한다.

05. 실험 결과

5.1 소스코드

위의 내용을 바탕으로 작성한 전체 소스코드는 아래와 같다.

```
#include "misc.h"
#include "core_cm3.h"
#include "stm32f10x.h"
#include "stm32f10x_exti.h"
#include "stm32f10x_gpio.h"
#include "stm32f10x_rcc.h"
#include "stm32f10x_usart.h"

int flagStick;
int flagNum[4];
int pinLED[4];

void RCC_Configure() {
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO, ENABLE);
        /*TODO : APB2PeriphClockEnable */
        RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOC|RCC_APB2Per
iph_GPIOD|RCC_APB2Periph_USART1, ENABLE);
}

void GPIO_Configure() {
        GPIO_InitTypeDef GPIO_InitStructure;

        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
```

```
        GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_3 | GPIO_Pin_4 |
GPIO_Pin_7);
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        /*TODO: USART1, JoyStick Config */
        GPIO_InitStructure.GPIO_Pin = (GPIO_Pin_2 | GPIO_Pin_5 );
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
        GPIO_Init(GPIOC, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_10;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
        GPIO_Init(GPIOA, &GPIO_InitStructure);

        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_11;
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPD;
        GPIO_Init(GPIOD, &GPIO_InitStructure);

        /*TODO: GPIO EXTILineConfig*/
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource2);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource5);
        GPIO_EXTILineConfig(GPIO_PortSourceGPIOD, GPIO_PinSource11);

}

void USART_Configure() {
        USART_InitTypeDef USART_InitStructure;

        /*TODO: USART1 configuration*/
        USART_InitStructure.USART_BaudRate = 9600;
        USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
        USART_InitStructure.USART_Parity = USART_Parity_Even;
        USART_InitStructure.USART_HardwareFlowControl =
USART_HardwareFlowControl_None;
        USART_InitStructure.USART_WordLength = USART_WordLength_9b;
        USART_InitStructure.USART_StopBits = USART_StopBits_1;
        USART_Init(USART1, &USART_InitStructure);

        /*TODO: USART1 cmd ENABLE*/
        USART_Cmd(USART1,ENABLE);

        /*TODO: USART1 IT Config*/
        USART_ITConfig(USART1, USART_IT_RXNE, ENABLE);

4.4 EXTI Configuration
void EXTI_Configure() {
        /*TODO: EXTI configuration [ mode interrupt ] [Trigger_falling] */
        EXTI_InitTypeDef EXTI_InitStructure;

        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
        EXTI_InitStructure.EXTI_Line = EXTI_Line11;
        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        EXTI_Init(&EXTI_InitStructure);

        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
```

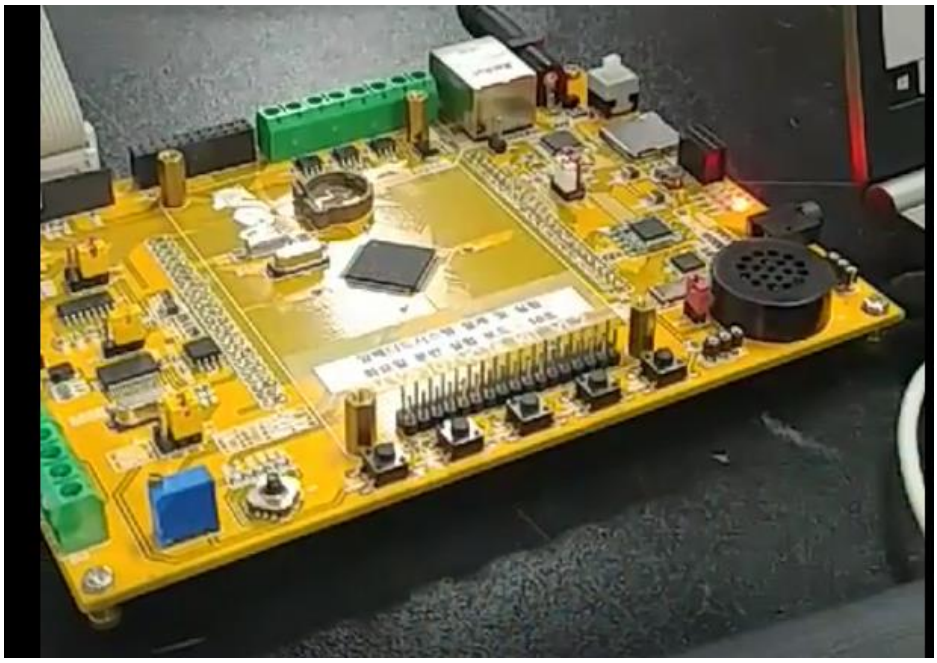
```
        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
        EXTI_InitStructure.EXTI_Line = EXTI_Line2;
        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        EXTI_Init(&EXTI_InitStructure);

        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
        EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
        EXTI_InitStructure.EXTI_Line = EXTI_Line5;
        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        EXTI_Init(&EXTI_InitStructure);
}

void NVIC_Configure() {
        /*TODO: NVIC_configuration */

        NVIC_InitTypeDef NVIC_InitStructure;

        NVIC_InitStructure.NVIC_IRQChannel = EXTI2_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&NVIC_InitStructure);

        NVIC_InitStructure.NVIC_IRQChannel = EXTI9_5_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&NVIC_InitStructure);

        NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&NVIC_InitStructure);

        NVIC_InitStructure.NVIC_IRQChannel = USART1_IRQn;
        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
        NVIC_Init(&NVIC_InitStructure);
}

void EXTI15_10_IRQHandler(void) {
        if(EXTI_GetITStatus(EXTI_Line11) != RESET)
                USART_SendData(USART1,'M');
        EXTI_ClearITPendingBit(EXTI_Line11);
}

void EXTI9_5_IRQHandler(void) {
        if(EXTI_GetITStatus(EXTI_Line5) != RESET)
                flagStick=1;
        EXTI_ClearITPendingBit(EXTI_Line5);
}

void EXTI2_IRQHandler(void) {
        if(EXTI_GetITStatus(EXTI_Line2) != RESET)
                flagStick=0;
        EXTI_ClearITPendingBit(EXTI_Line2);
}

void USART1_IRQHandler(void) {
```

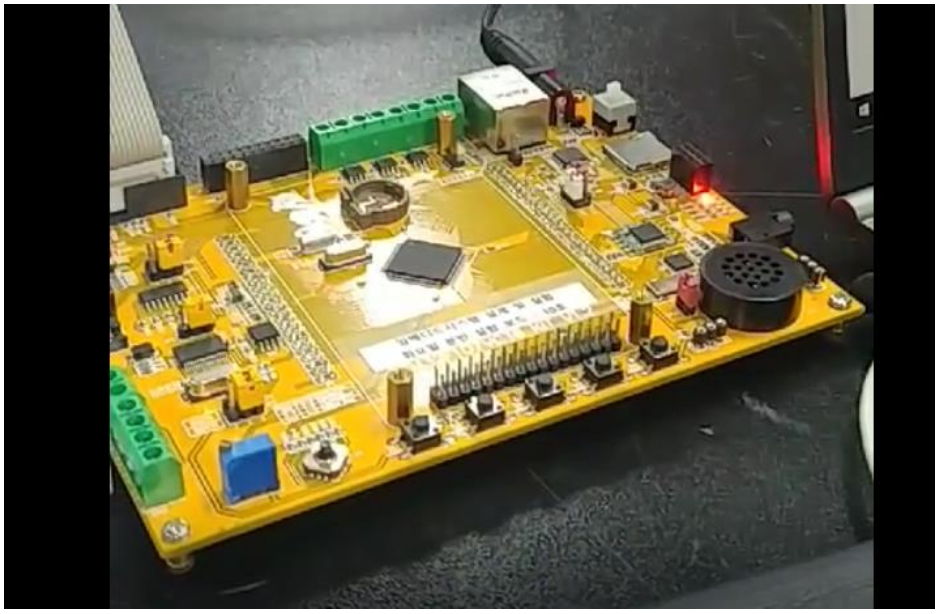
```
        char c;
        if(USART_GetITStatus(USART1, USART_IT_RXNE ) != RESET) {
                c = (char)USART_ReceiveData(USART1);
                flagNum[c-'1'] = 1;
        }
        USART_ClearITPendingBit(USART1,USART_IT_RXNE);
}

void delay(void) {
        int i=0;
        for(i=0;i<1000000;++i);
}

int main()
{
        int i=0;

        flagStick = 2;
        for(i=0;i<4;++i) flagNum[i] = 0;
        pinLED[0] = GPIO_Pin_2;
        pinLED[1] = GPIO_Pin_3;
        pinLED[2] = GPIO_Pin_4;
        pinLED[3] = GPIO_Pin_7;

        SystemInit();
        RCC_Configure();
        GPIO_Configure();
        USART_Configure();
        EXTI_Configure();
        NVIC_Configure();

        while(1) {
                if(flagStick==1) {                                  // up
                        for(i=0;i<4 && flagStick==1 ;++i) {
                                if(pinLED[i]==1) continue;
                                GPIO_SetBits(GPIOD,pinLED[i]);
                                delay();
                                GPIO_ResetBits(GPIOD,pinLED[i]);
                        }
                }
                if(flagStick==0) {                                  // down
                        for(i=3;i>=0 && flagStick==0 ;--i) {
                                if(pinLED[i]==1) continue;
                                GPIO_SetBits(GPIOD,pinLED[i]);
                                delay();
                                GPIO_ResetBits(GPIOD,pinLED[i]);
                        }
                }
        }
}
```

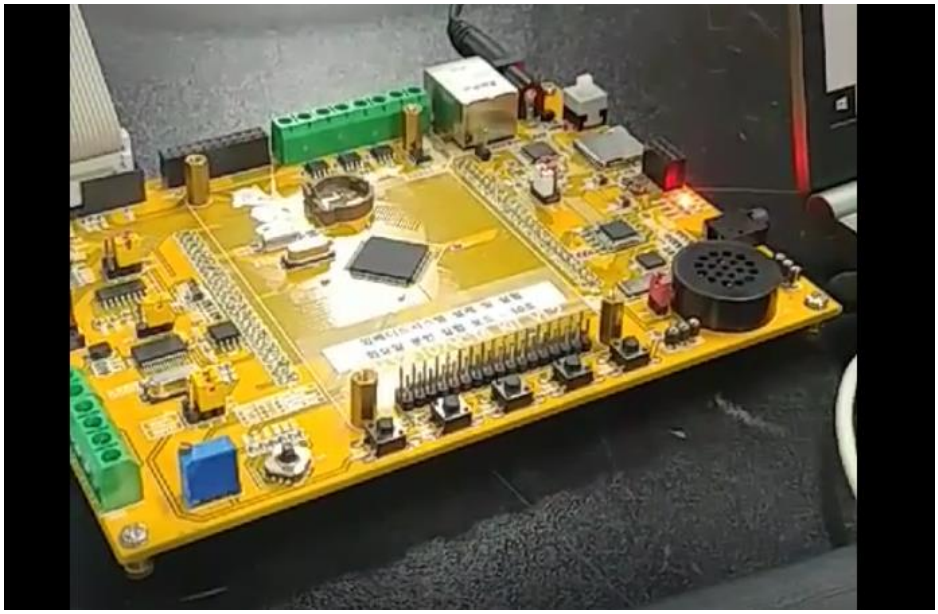
5.2 결과 확인 및 사진

해당 소스 코드대로 동작하는 프로그램은 다음과 같이 확인되었다.

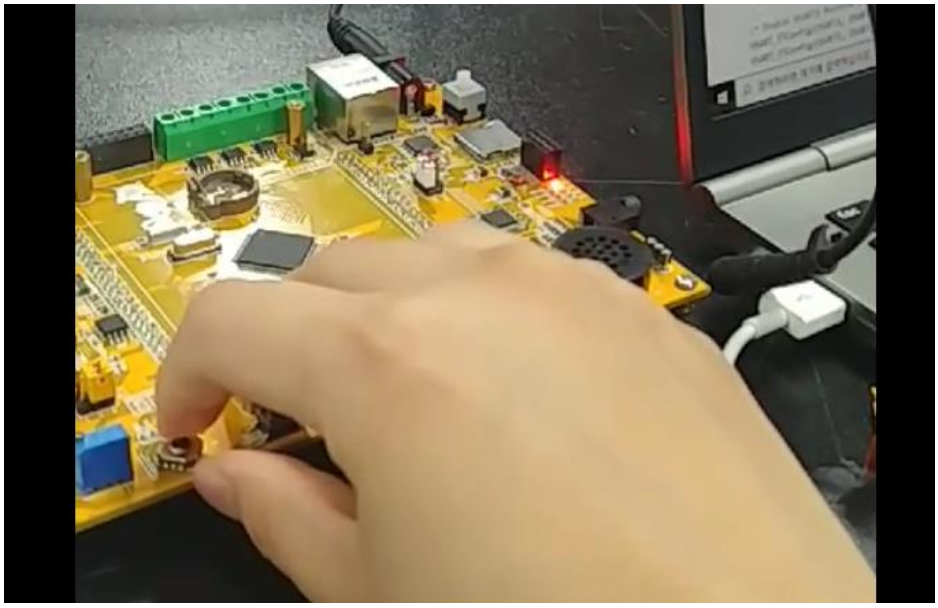
조이스틱 Up : 조이스틱 Up 버튼을 누르면 LED 가 순서대로 물결친다.

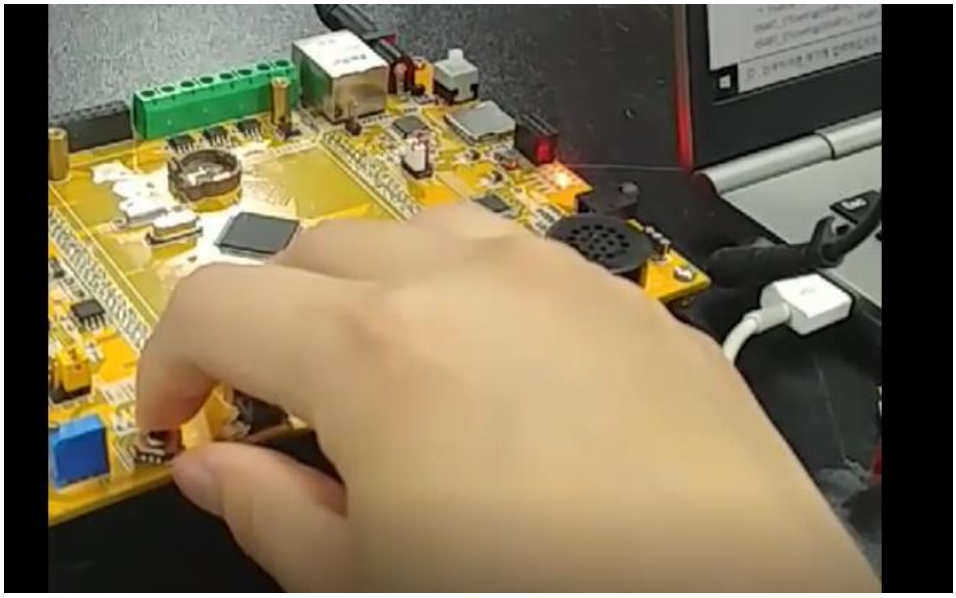




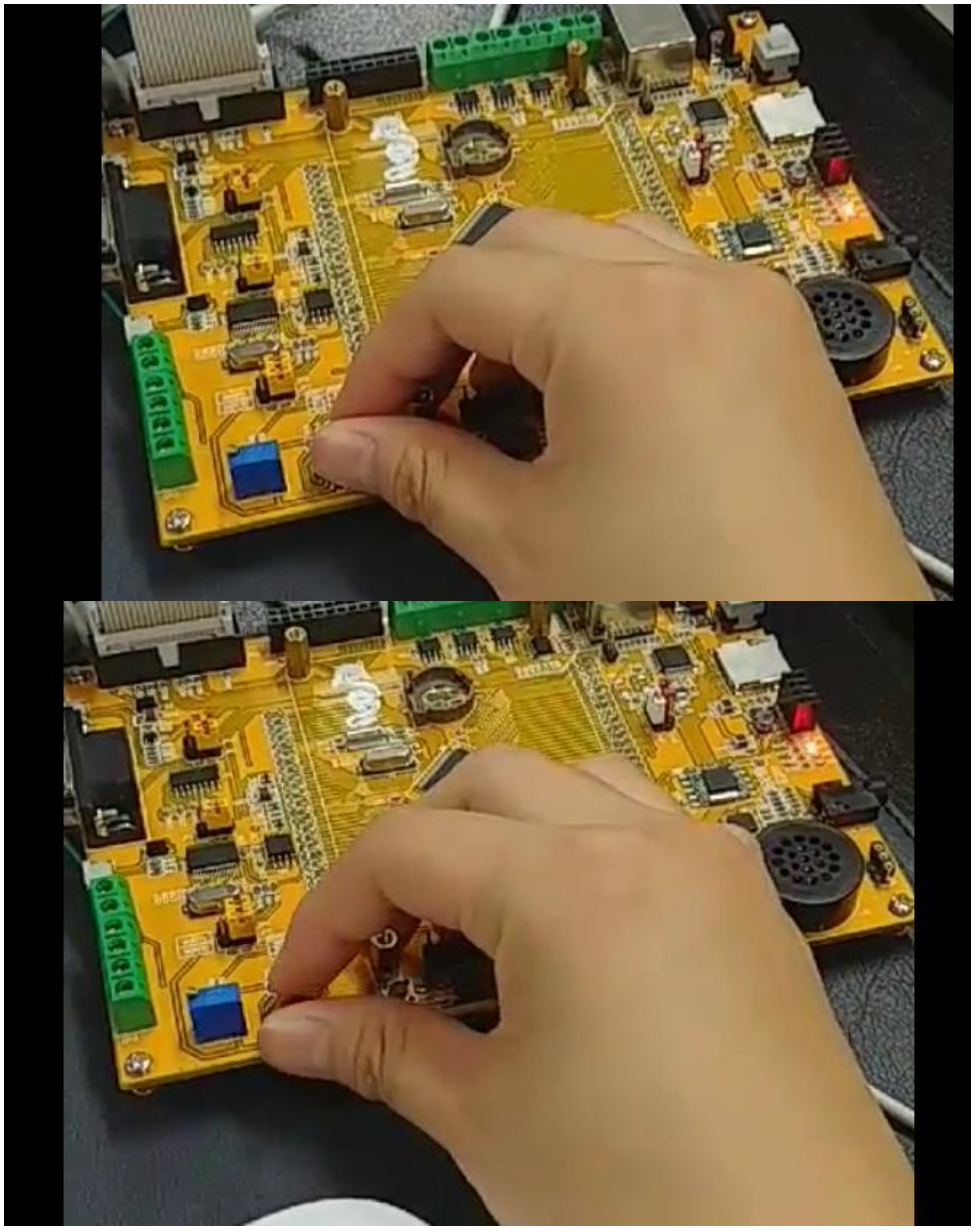


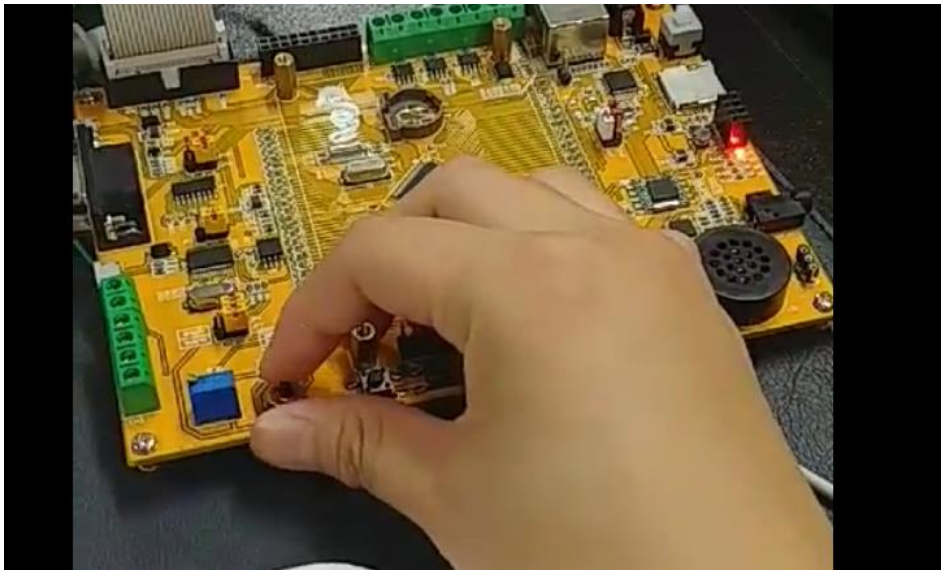
조이스틱 Down: 조이스틱 Down 버튼을 누르면 LED 가 반대 순서로 물결친다.



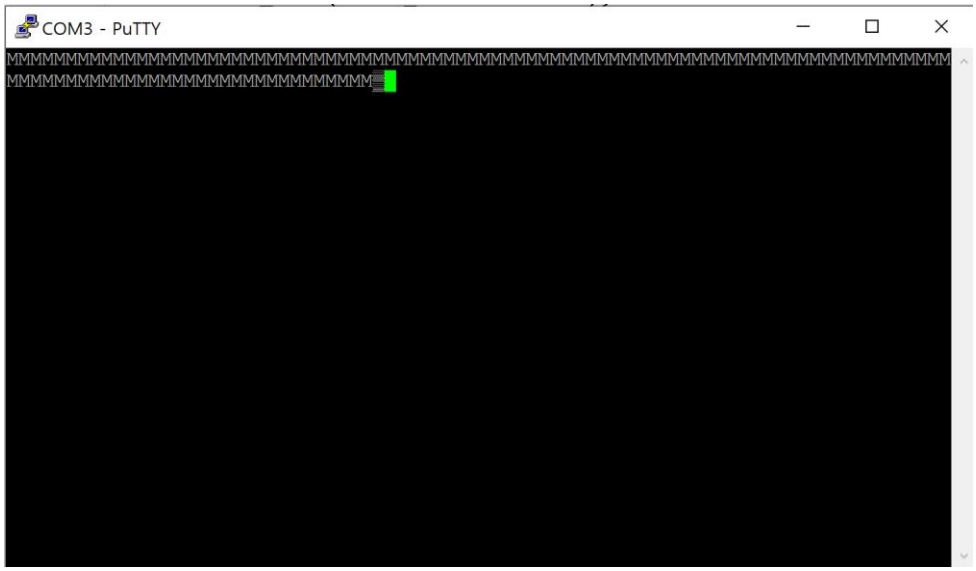


Putty 로 숫자 1~4 입력 : 해당 입력 숫자에 해당하는 LED 가 계속 꺼진 채로 나머지 LED 가 물결친다.



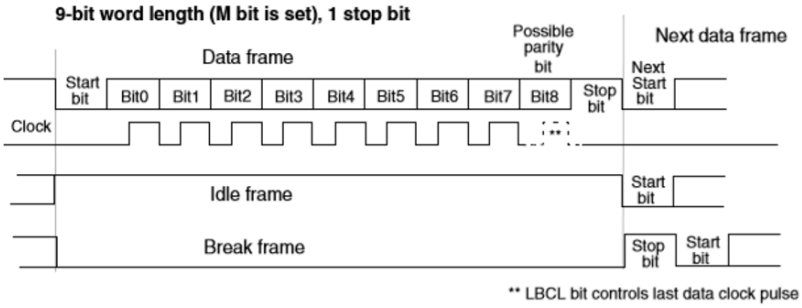


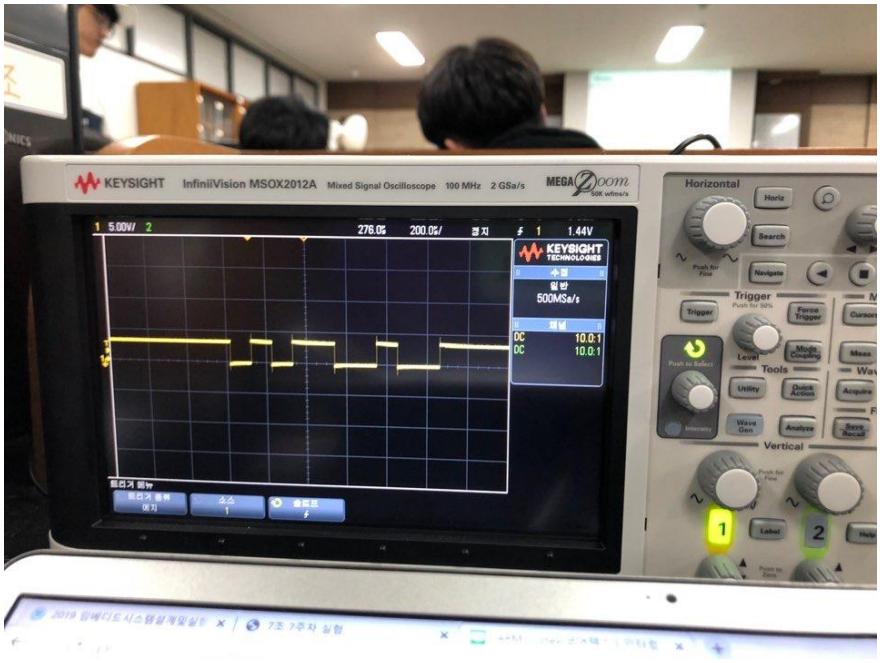
버튼(PD11) : Putty 를 통해 문자 'M'이 출력되고, 오실로스코프에도 입력이 들어간다.

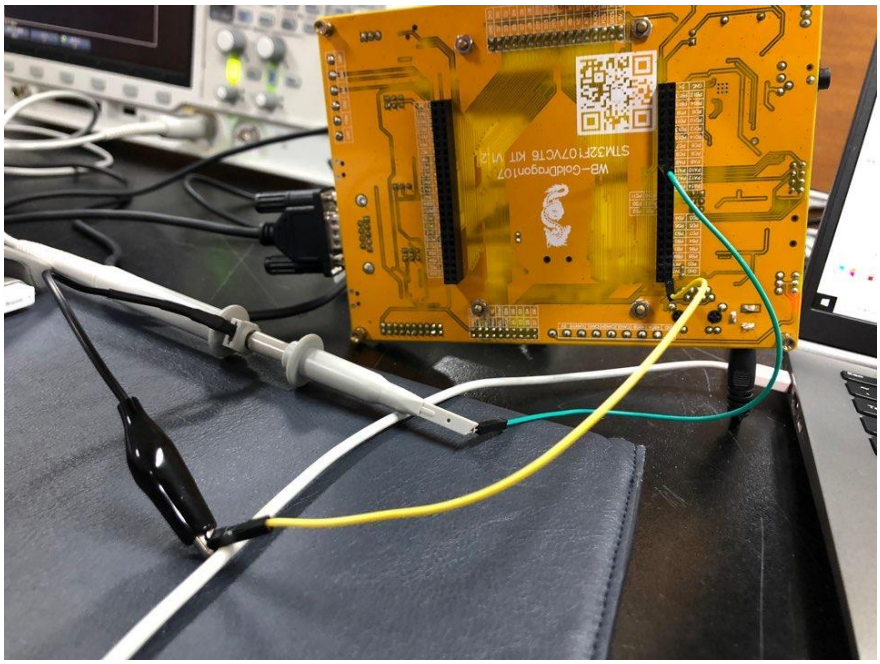


5.3 오실로스코프를 이용한 문자 'M' 출력 확인

출력해야 하는 M 의 아스키코드 값은 0x4D 이다. 이진수로 나타내면 0100 1101 이다. Even parity 를 사용할 경우 전체 프레임 비트는 0 0100 1101 0 1 이다. 그러나 데이터 비트는 순서가 뒤집어지므로 오실로스코프에 출력되는 파형은 0 1011 0010 0 1 로 출력이 된다.







오실로스코프-보드 연결 사진이다.

06. 결론

이번 실험에서는 ctrl + space 를 활용해서 주어진 구조체에 대해서 함수를 찾고 이해해서, 값을 입력하는 식으로 코딩을 진행했다. 평소보다 구조체와, 구성 요소 값 등을 이해하고, 익숙하지 않아서 시간이 더 오래 걸렸다. 오늘 처음으로 인터럽트 방식을 활용해 코드를 작성하고 실험을 진행했는데 라이브러리 함수와 구조체만 잘 이해한다면 폴링 방식보다 훨씬 깔끔한 코딩이 가능하고, 구현도 쉬운 것 같다. 또 동작도 훨씬 부드럽고, 원하는 대로 오류 없이 잘 동작했던 것 같다. 최근에 임베디드 시스템 과목에서도 인터럽트에 대해 공부했었는데, 이론으로만 배운 인터럽트로 동작하는 방식을 실제 임베디드 보드에 코딩을 통해 실습해보니 이해에 더 도움이 된 것 같다. 또 매주 오실로스코프를 사용하다 보니 사용법도 자연스럽게 익혀지는 것 같다.