# 임베디드시스템설계

## Embedded System Capstone Design

ICE3015-001


과제 1

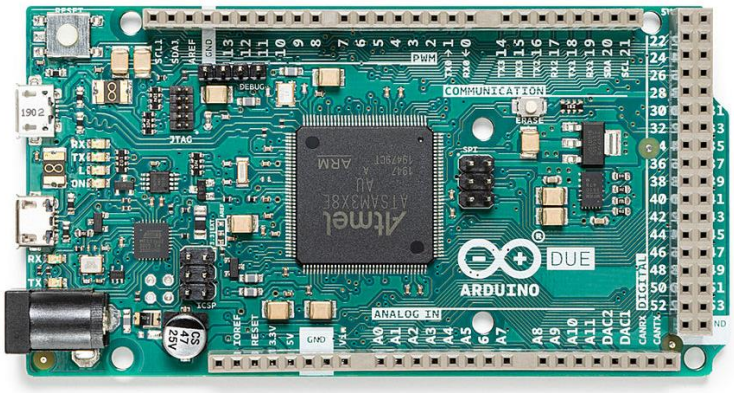
**실습팀 4**

**정보통신공학과 12181855 황용하**

**정보통신공학과 12191720 곽재현**

**정보통신공학과 12191765 박승재**

# Introduction




Arduino Due 는 Atmel SAM3X8E ARM Cortex-M3 CPU 를 사용하는 마이크로 컨트롤러 보드이다. 동작 전원은 3.3V 이며 12 개의 Analog Input 을 가지고 있다. Arduino 는 사용가능한 문법이 제한된 C++언어를 이용해 프로그래밍할 수 있으며, 코드의 구성부로 크게 `setup` 함수와 `loop` 함수로 나뉘어 있다.

```
// https://github.com/arduino/ArduinoCore-sam/blob/master/cores/arduino/main.cpp
int main( void )
{
    // Initialize watchdog
    watchdogSetup();
    init();
    initVariant();
    delay(1);
#if defined(USBCON)
    USBDevice.attach();
#endif
    setup();
    for (;;)
    {
        loop();
        if (serialEventRun) serialEventRun();
    }
    return 0;
}
```

Arduino Core 의 main 함수를 들여다보면 가장 먼저 `watchdogSetup` 함수가 호출된다.

```
// https://github.com/arduino/ArduinoCore-sam/blob/master/cores/arduino/watchdog.cpp
extern "C"
void _watchdogDefaultSetup (void)
{
    WDT_Disable (WDT);
}
```

```
void watchdogSetup (void) __attribute__ ((weak,
alias("_watchdogDefaultSetup")));
```

```
// https://github.com/arduino/ArduinoCore-sam/blob/master/system/libsam/source/wdt.c
extern void WDT_Disable( Wdt* pWDT )
{
    pWDT->WDT_MR = WDT_MR_WDDIS;
}
```

watchdogSetup 는 watchdog 을 비활성화한다.

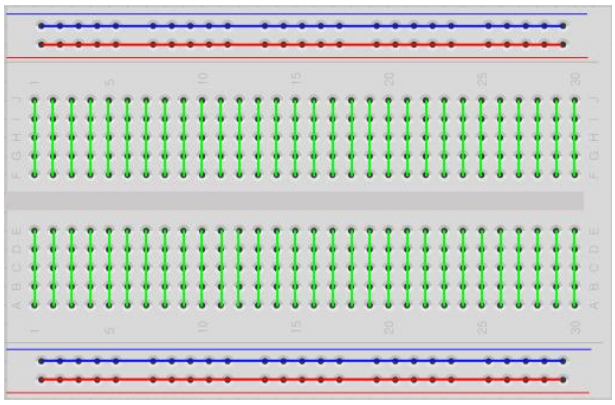
```
// https://github.com/arduino/ArduinoCore-sam/blob/master/variants/arduino_due_x/variant.cpp
void init( void )
{
  SystemInit();
  // Set Systick to 1ms interval, common to all SAM3 variants
  if (SysTick_Config(SystemCoreClock / 1000))
  {
    // Capture error
    while (true);
  }
  // Initialize C library
  __libc_init_array();
  // Disable pull-up on every pin
  for (unsigned i = 0; i < PINS_COUNT; i++)
      digitalWrite(i, LOW);
  // Enable parallel access on PIO output data registers
  PIOA->PIO_OWER = 0xFFFFFFFF;
  PIOB->PIO_OWER = 0xFFFFFFFF;
  PIOC->PIO_OWER = 0xFFFFFFFF;
  PIOD->PIO_OWER = 0xFFFFFFFF;
  // Initialize Serial port U(S)ART pins
  PIO_Configure(
    g_APinDescription[PINS_UART].pPort,
    g_APinDescription[PINS_UART].ulPinType,
    g_APinDescription[PINS_UART].ulPin,
    g_APinDescription[PINS_UART].ulPinConfiguration);
    // ...
}
```

그 뒤의 init 에서는 Systick 을 1ms 로 설정하고, C 라이브러리와 보드의 핀 데이터를 기본값으로 초기화한다. init 이 끝나면 setup 이 호출된다. setup 은 아두이노 프로그램 코드의 시작점이 되는 함수이다. 주로 Serial 값이나 pin 데이터를 초기화하는 코드가 들어간다. init 이후, for 을 이용한 무한 반복문으로 loop 가 호출된다.

```
// https://github.com/arduino/ArduinoCore-sam/blob/master/variants/arduino_due_x/variant.cpp
UARTClass Serial(UART, UART_IRQn, ID_UART, &rx_buffer1, &tx_buffer1);
void serialEvent() __attribute__((weak));
void serialEvent() { }
USARTClass Serial1(USART0, USART0_IRQn, ID_USART0, &rx_buffer2,
```

```
&tx_buffer2);
void serialEvent1() __attribute__((weak));
void serialEvent1() { }
USARTClass Serial2(USART1, USART1_IRQn, ID_USART1, &rx_buffer3,
&tx_buffer3);
void serialEvent2() __attribute__((weak));
void serialEvent2() { }
USARTClass Serial3(USART3, USART3_IRQn, ID_USART3, &rx_buffer4,
&tx_buffer4);
void serialEvent3() __attribute__((weak));
void serialEvent3() { }

void serialEventRun(void)
{
  if (Serial.available()) serialEvent();
  if (Serial1.available()) serialEvent1();
  if (Serial2.available()) serialEvent2();
  if (Serial3.available()) serialEvent3();
}
```

loop 가 끝날 때마다 serialEventRun 이 호출되는데, serialEventRun 은 Serial 버퍼에 데이터가 있다면 serialEvent 를 호출한다. serialEvent 는 프로그래머가 시리얼 ISR 로 구현 가능한 함수이다. loop 는 일정 시간마다 호출되는 함수가 아니라, 이전 루프가 끝나는 대로 for 에 의해 연속적으로 호출되는 함수이다.



브레드보드(빵판) 전자회로의 시제품을 만들 때 사용하는 재사용 가능한 무납땜 장치이다. 브레드보드 위아래의 파랑/빨강선은 버스라고 불리며 모두 이어져 있다. 내부의 초록선은 가로로 이어져 있으며, 주변의 초록선과는 격리되어 있다. 실습 때 사용하는 브레드보드는 사진의 브레드보드를 가로로 2 개 이은 형태이다. 따라서 파랑/빨강선은 총 4 쌍이며, 파랑/빨강선끼리 서로 이어져 있지 않으니, GND 를 연결할 때 주의해서 연결해야 한다.

## pinMode

지정한 핀을 INPUT 또는 OUTPUT 을 작동하게 설정한다. INPUT, INPUT_PULLUP, OUTPUT 3 가지 값을 사용할 수 있다.

### digitalWrite

디지털 핀의 값을 HIGH 또는 LOW 로 수정한다. 해당 핀이 OUTPUT 으로 설정되었다면, HIGH 에는 전압이 3.3V, LOW 에는 전압이 0V 가 되게 한다. 해당 핀이 INPUT 이라면 내부 Pullup 값을 조절한다. HIGH 와 LOW 는 코드상에서 각각 unsigned int 타입의 1 과 0 으로 표현된다.

### delay

지정된 시간(밀리초) 동안 프로그램을 일시정지시킨다.

```
// https://github.com/arduino/ArduinoCore-sam/blob/master/cores/arduino/wiring.c
void delay( unsigned long ms )
{
    if (ms == 0)
        return;
    uint32_t start = GetTickCount();
    do {
        yield();
    } while (GetTickCount() - start < ms);
}
```

반복문을 이용한 Busy-waiting 방식으로 구현되었다.

### digitalRead

디지털 핀의 값이 HIGH 인지 LOW 인지 읽어온다.

### Serial.begin

Baud Rate 를 설정한다. 단위는 bps 이다.

### Serial.available

Serial 에서 읽어올 수 있는 바이트 수를 반환한다.

### Serial.read

Serial 버퍼의 값을 반환한다.

### Serial.println

Serial 포트로 사람이 읽을 수 있는 ASCII 텍스트를 출력한다. print 와 달리, 맨 마지막에 '₩n'을 자동으로 붙여준다.

# Problem

1. Analyze the code below and print the results of the execution.

```
void setup() {
    Serial.begin(9600);
}

void loop() {
    if (Serial.available()) {
        char c = Serial.read();
        if (('A' <= c) && (c <= 'Z')) {
            c += 0x20;
        } else if (('a' <= c) && (c <= 'z')) {
            c -= 0x20;
        }
        Serial.write(c);
    }
}
```

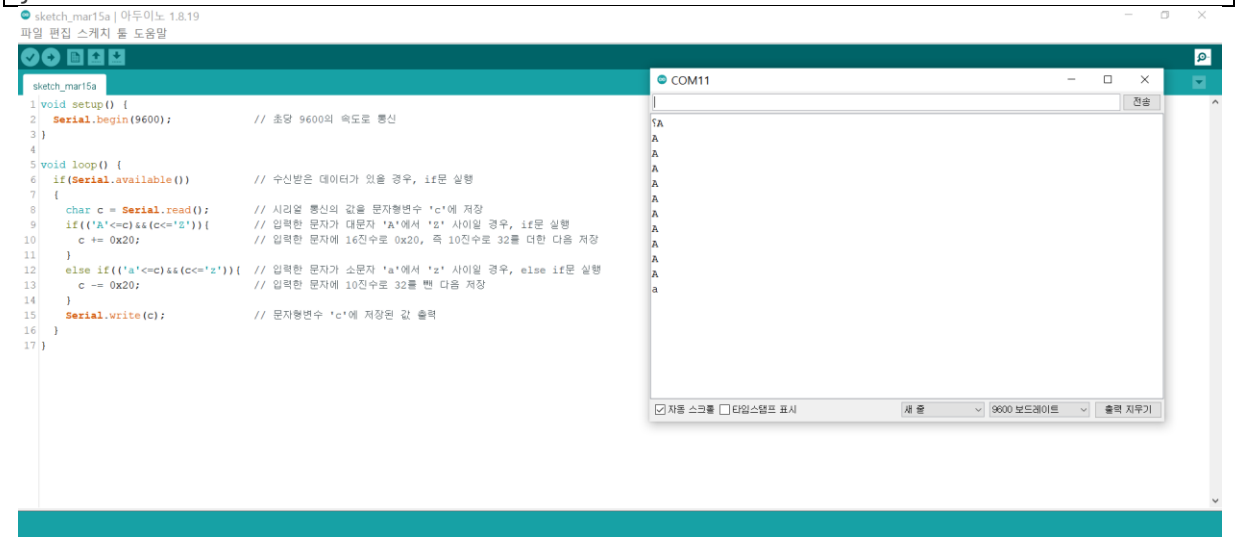
2. Write a program that controls 3 LED lights according to the switch input.
   - SW1: green led toggle
   - SW2: red led toggle
   - SW3: blue led toggle
3. Create a program so that when the user enters 'g' 'r' 'y', that color LED is blinking. If user enters 'g' when green is blinking, green led turn off.

# Result

## Problem 1

```
void setup() {
    Serial.begin(9600); // Serial 의 Baud Rate 를 9600bps 로 설정
}

void loop() {
    if (Serial.available()) { // Serial 을 통해 읽을 수 있는 비트 수를 반환
        char c = Serial.read(); // Serial 에서 1 비트 읽기
        if (('A' <= c) && (c <= 'Z')) { // 읽어온 문자가 알파벳 대문자
            c += 0x20; // 알파벳 소문자로 변경
        } else if (('a' <= c) && (c <= 'z')) { // 읽어온 문자가 알파벳 소문자
```

```
            c -= 0x20; // 알파벳 대문자로 변경
        }
        Serial.write(c); // Serial 에 작성 (Serial 모니터에 출력)
    }
}
```



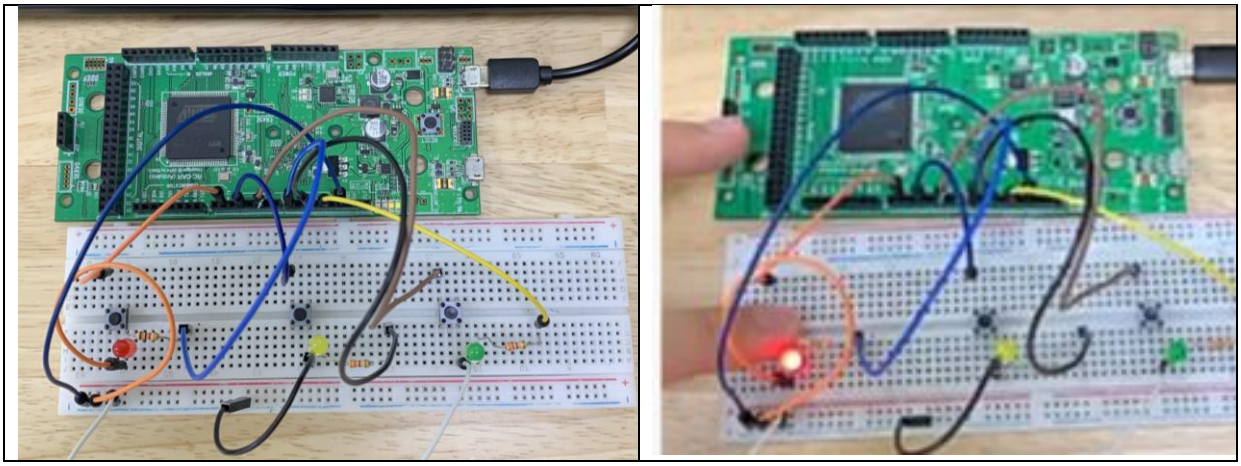

| 10진수 | 부호 | 10진수 | 부호 | 10진수 | 부호 | 10진수 | 부호 |
|---|---|---|---|---|---|---|---|
| 032 | | 056 | 8 | 080 | P | 104 | h |
| 033 | ! | 057 | 9 | 081 | Q | 105 | i |
| 034 | " | 058 | : | 082 | R | 106 | j |
| 035 | # | 059 | ; | 083 | S | 107 | k |
| 036 | $ | 060 | < | 084 | T | 108 | l |
| 037 | % | 061 | = | 085 | U | 109 | m |
| 038 | & | 062 | > | 086 | V | 110 | n |
| 039 | ' | 063 | ? | 087 | W | 111 | o |
| 040 | ( | 064 | @ | 088 | X | 112 | p |
| 041 | ) | 065 | A | 089 | Y | 113 | q |
| 042 | * | 066 | B | 090 | Z | 114 | r |
| 043 | + | 067 | C | 091 | [ | 115 | s |
| 044 | , | 068 | D | 092 | \ | 116 | t |
| 045 | - | 069 | E | 093 | ] | 117 | u |
| 046 | . | 070 | F | 094 | ^ | 118 | v |
| 047 | / | 071 | G | 095 | _ | 119 | w |
| 048 | 0 | 072 | H | 096 | ` | 120 | x |
| 049 | 1 | 073 | I | 097 | a | 121 | y |
| 050 | 2 | 074 | J | 098 | b | 122 | z |
| 051 | 3 | 075 | K | 099 | c | 123 | { |
| 052 | 4 | 076 | L | 100 | d | 124 | \| |
| 053 | 5 | 077 | M | 101 | e | 125 | } |
| 054 | 6 | 078 | N | 102 | f | 126 | ~ |
| 055 | 7 | 079 | O | 103 | g | | |

void setup()에서 Serial 통신의 Baud Rate 를 9600bps 로 설정을 한다. loop 함수 안에서는 if 문을 사용하여 'Serial.available'을 통해서 읽어올 수 있는 byte 의 수를 반환을 받고 반환된 값이 True 가 되면 Serial 을 통해서 읽어온 1 비트를 'Serial.read()'를 통해서 입력을 받는다. 이때 입력을 받은 비트는 문자고 해당 문자가 대문자 범위일 경우는 +0x20 연산을 통해서 소문자로, 소문자 범위일 경우는 -0x20 연산을 통해서 대문자로 변환한다. 이후 if 문이 종료를 하게 되면 c 에 저장된 문자열이 출력이 된다.

## Problem 2

```
// LED와 연결할 버튼 핀번호 설정
#define REDBUTTON 1
#define YELLOWBUTTON 3
#define GREENBUTTON 5

// 3가지 색의 LED 핀번호 설정
#define REDLED 8
#define YELLOWLED 10
#define GREENLED 12

// 버튼이 눌러진 상태를 저장하는 변수 선언
bool clickedRed = false;
bool clickedYellow = false;
bool clickedGreen = false;

// LED ON/OFF 상태를 저장하는 변수 선언
bool red = false;
bool yellow = false;
bool green = false;

void setup() {
    // 각각의 버튼을 입력 핀으로 설정하고, 내부 풀업 저항을 사용
    pinMode(REDBUTTON, INPUT_PULLUP);
    pinMode(YELLOWBUTTON, INPUT_PULLUP);
    pinMode(GREENBUTTON, INPUT_PULLUP);

    // 각각의 LED를 출력 핀으로 설정
    pinMode(REDLED, OUTPUT);
    pinMode(YELLOWLED, OUTPUT);
    pinMode(GREENLED, OUTPUT);

    // Serial 통신 시작
    Serial.begin(9600);
}

void loop() {
    // 빨간색 LED와 연결된 버튼 누를 시 if문 실행
    if (!clickedRed && digitalRead(REDBUTTON) == LOW) {
        clickedRed = true;          // 버튼이 눌린 상태 저장
        Serial.println("press1");    // 시리얼 모니터에 "press1" 출력
        red = !red;                 // LED ON/OFF 상태 변경
    }
    // 빨간색 LED와 연결된 버튼 떼어낼 시 else if문 실행
    else if (clickedRed && digitalRead(REDBUTTON) == HIGH) {
        clickedRed = false;         // 버튼이 떼어진 상태 저장
        Serial.println("release1");  // 시리얼 모니터에 "release1" 출력
```

```
    }
    digitalWrite(REDLED, (unsigned int)red); // 빨간색 LED 제어

    // 노란색 LED와 연결된 버튼 누를 시 if문 실행
    if (!clickedYellow && digitalRead(YELLOWBUTTON) == LOW) {
        clickedYellow = true;       // 버튼이 눌린 상태 저장
        Serial.println("press2");    // 시리얼 모니터에 "press2" 저장
        yellow = !yellow;           // LED ON/OFF 상태 변경
    }
    // 노란색 LED와 연결된 버튼 떼어낼 시 else if문 실행
    else if (clickedYellow && digitalRead(YELLOWBUTTON) == HIGH) {
        clickedYellow = false;      // 버튼이 떼어진 상태 저장
        Serial.println("release2");  // 시리얼 모니터에 "release2" 출력
    }
    digitalWrite(YELLOWLED, (unsigned int)yellow); // 노란색 LED 제어

    // 초록색 LED와 연결된 버튼 누를 시 if문 실행
    if (!clickedGreen && digitalRead(GREENBUTTON) == LOW) {
        clickedGreen = true;        // 버튼이 눌린 상태 저장
        Serial.println("press3");    // 시리얼 모니터에 "press3" 출력
        green = !green;             // LED ON/OFF 상태 변경
    }
    // 초록색 LED와 연결된 버튼 떼어낼 시 else if문 실행
    else if (clickedGreen && digitalRead(GREENBUTTON) == HIGH) {
        clickedGreen = false;       // 버튼이 떼어진 상태 저장
        Serial.println("release3");   // 시리얼 모니터에 "release3" 출력
    }
    digitalWrite(GREENLED, (unsigned int)green); // 초록색 LED 제어
}
```



'pinMode'를 통해서 REDBUTTON, YELLOWBUTTON, GREENBUTTON 에 각각 'INPUT_PULLUP'을 넣어서 내부 'PULL_UP' 저항을 사용하게 만들었고 REDLED, YELLOWLED, GREENLED 를 통해서 OUTPUT 을 넣어서 버튼을 눌렀을 때 전압의 차이가 발생하면 불이 켜지게 만들었다.

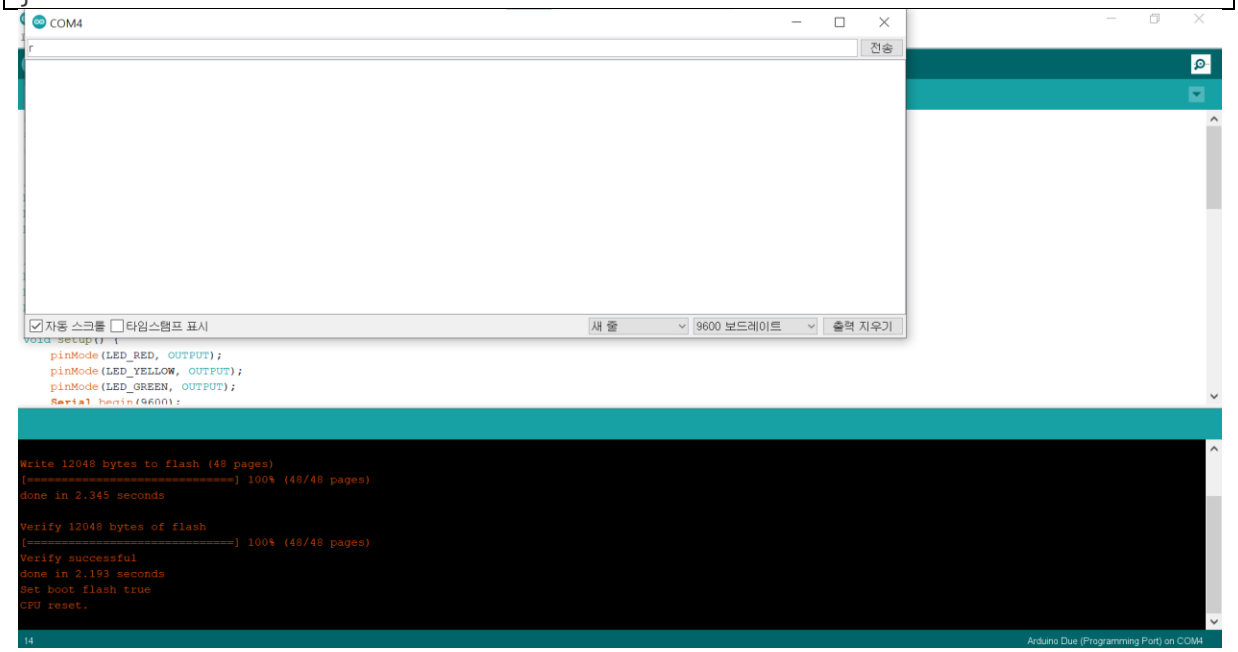
전역변수로 bool 형의 'clickedRED, clickedYellow, clickedGreen'을 모두 false 로 선언을 한 이후에 버튼을 눌러 led 가 켜진다. 버튼을 누르면 'clickedRED, clickedYellow, clickedGreen'의 상태가 변하게 되고 이를 통해서 LED 를 상태를 변환할 수 있게 만들었다.

테스트 중 계속 LED 가 비정상적으로 깜빡이는 문제가 발생하여, Serial write 코드를 넣어 디버깅을 하였다. 그 결과, 코드나 회로의 문제가 아니라 스위치에서 발생한 Chattering 현상이라고 추정했다.

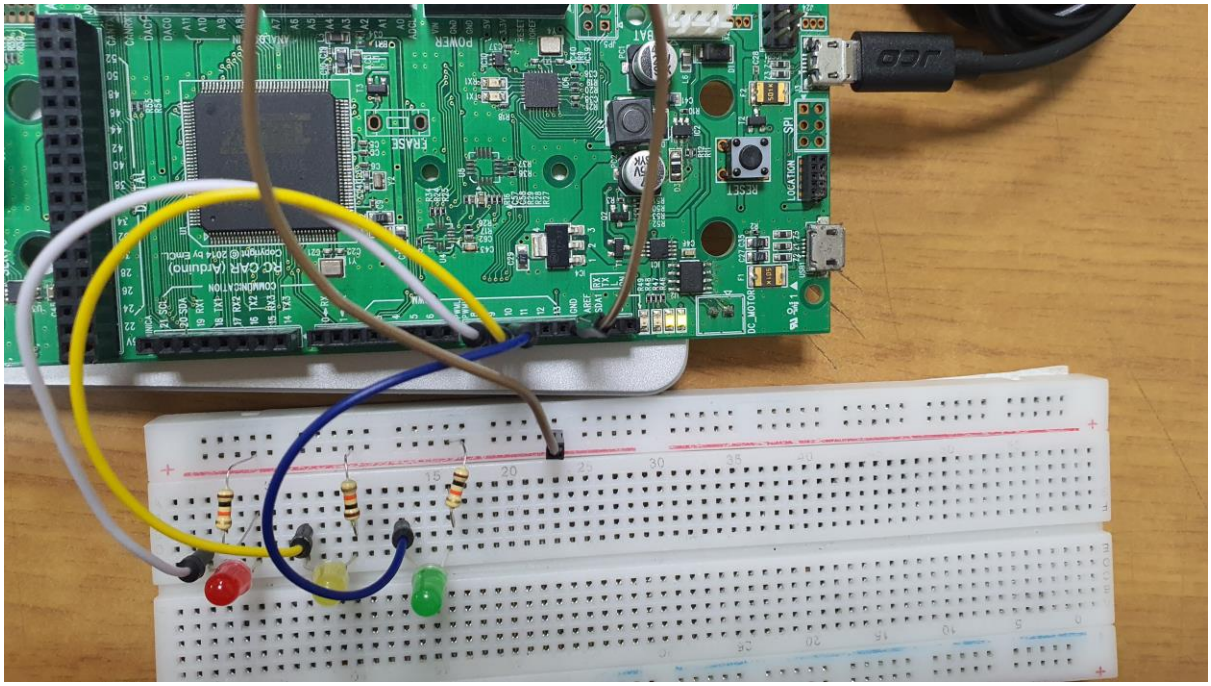
## Problem 3

```
#define LED_RED 8
#define LED_YELLOW 9
#define LED_GREEN 10

// LED enable
bool red = false;
bool yellow = false;
bool green = false;

// LED blink status
bool blinkRed = false;
bool blinkYellow = false;
bool blinkGreen = false;

void setup() {
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_YELLOW, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    Serial.begin(9600);
}

void loop() {
    // Read Serial
    if (Serial.available() > 0) { // The number of bytes available to read
        char c = Serial.read();
        c |= ' '; // to lowercase
        switch (c) {
        case 'r':
            red = !red;
            digitalWrite(LED_RED, (unsigned int) red); // LED off
```

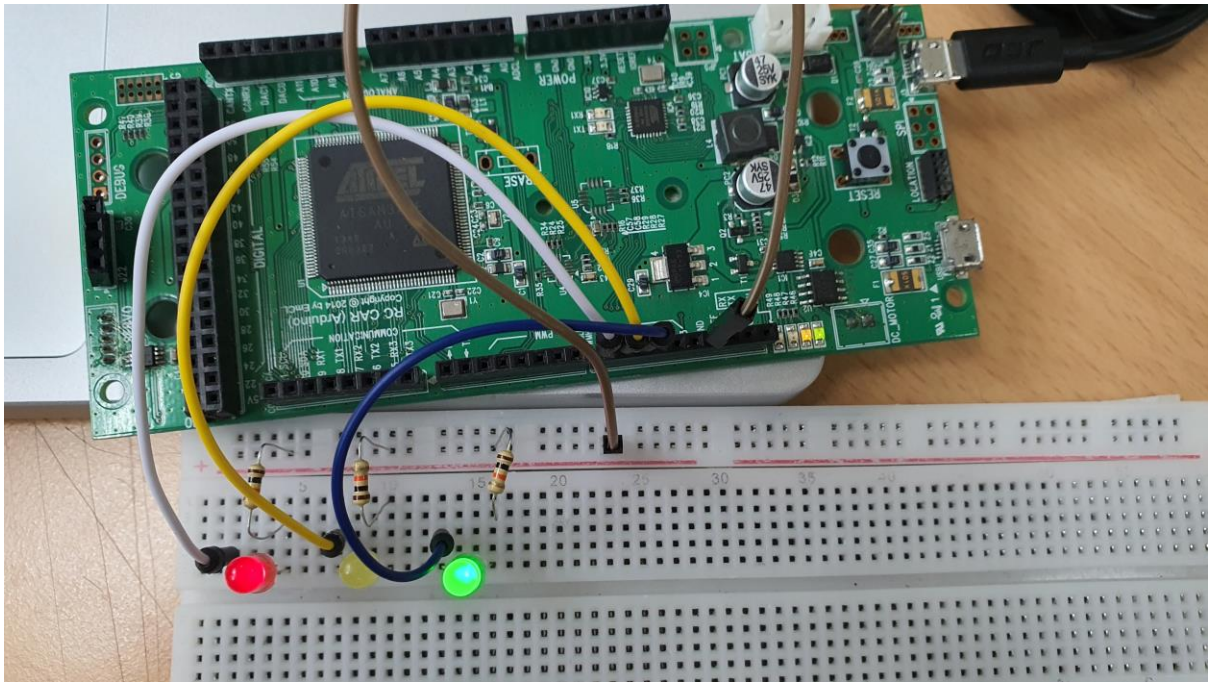
```
            break;
        case 'y':
            yellow = !yellow;
            digitalWrite(LED_YELLOW, (unsigned int) yellow); // LED off
            break;
        case 'g':
            green = !green;
            digitalWrite(LED_GREEN, (unsigned int) green); // LED off
            break;
        default:
            break;
        }
    }

    // Blink LED
    if (red) {
        blinkRed = !blinkRed;
        digitalWrite(LED_RED, (unsigned int) blinkRed);
    }
    if (yellow) {
        blinkYellow = !blinkYellow;
        digitalWrite(LED_YELLOW, (unsigned int) blinkYellow);
    }
    if (green) {
        blinkGreen = !blinkGreen;
        digitalWrite(LED_GREEN, (unsigned int) blinkGreen);
    }
    delay(100);
}
```



`setup` 에서 Output 핀과 Serial 의 Baud Rate 를 설정했다. `loop` 에서 Serial 로부터 값을 읽어 `switch` 를 통해 값을 비교하도록 했다. `c |= ' ';`는 임의의 ASCII 문자를 소문자로 바꿔주는 코드이다. LED 의 점멸 간격은 0.1 초로 설정했다.



사진과 같이 LED 와 저항을 이용해 회로를 구성했다. 각각의 LED 마다 저항을 달았으며, 보드의 8, 9, 10 핀에 연결되어 있다.



Serial 로 r 을 보내면 사진과 같이 빨간색 LED 반복적으로 점등된다. 다시 r 을 보내면 빨간색 LED 가 완전히 꺼진다. 노란색과 초록색 LED 도 각각 y 와 g 문자를 전송하면 반복적으로

점등하는 것을 확인했다. 동시에 여러 개의 LED 가 점등할 수 있게 구현했다. `c |= ' ';`로 입력받은 문자를 대응하는 소문자로 치환했기 때문에 입력으로 대소문자를 가리지 않는다.

# Conclusion

## problem1

문제 1 번의 대소문자 변환코드는 아래와 같은 Bitwise Trick 을 이용해 간결하게 구현할 수 있다.

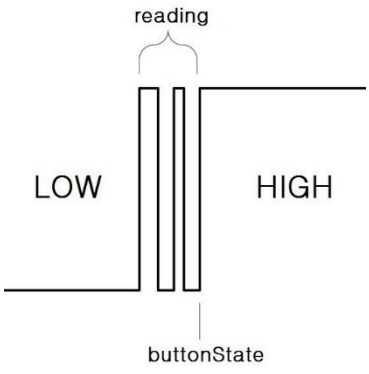
`c |= ' ';`를 통해 임의의 문자 중 알파벳만 소문자로 치환 가능

`c &= '_';`를 통해 임의의 문자 중 알파벳만 대문자로 치환 가능

위 방식을 사용하면 if 문을 이용해 c 의 범위를 체크하는 과정이 사라지기 때문에 더 빠르게 코드를 동작시킬 수 있다.

## problem2

문제 2 번의 수행과정에서는 버튼을 클릭할 때 소프트웨어 상에서 더블클릭으로 인식하여 LED 가 빠르게 켜졌다 꺼지는 문제가 발생했다. 이를 Chattering 이라 하며, Chattering 현상은 다음과 같이 스위치의 접점이 기계적인 진동에 의해서 붙거나 떨어지는 것이 반복되는 현상을 의미한다.

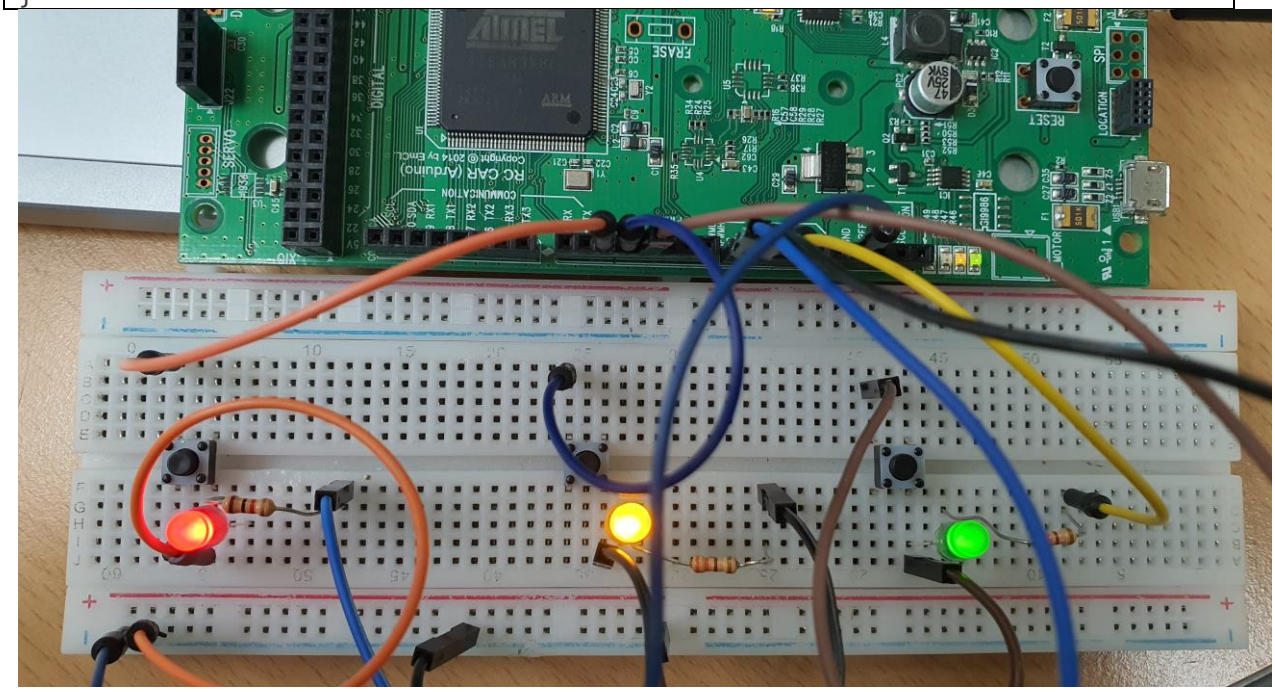



사진과 같이 매우 짧은 시간 동안 1 과 0 의 신호가 반복한다는 것을 알 수 있다. Chattering 현상을 억제하는 방법은 하드웨어적 소프트웨어적 방법이 있으며 하드웨어적 방법으로는 RS flip-flop 회로를 추가하거나 capacitor 를 추가하는 방법이 있으며 소프트웨어적으로는 디바운스 코드를 사용함으로써 Chattering 을 방지할 수 있다. 아래는 문제 2 번의 Chattering 현상을 보완해 다시 작성한 코드이다.

```
#define BUTTON_RED 2
#define BUTTON_YELLOW 3
#define BUTTON_GREEN 4
#define LED_RED 8
#define LED_YELLOW 9
#define LED_GREEN 10

// LED enable
bool red = false;
```

```
bool yellow = false;
bool green = false;

// Button pressed status
bool pressedRed = false;
bool pressedYellow = false;
bool pressedGreen = false;

// Button released timestamp
unsigned long releasedRed = 0; // overflow after 50 days
unsigned long releasedYellow = 0;
unsigned long releasedGreen = 0;

void setup() {
    pinMode(BUTTON_RED, INPUT_PULLUP);
    pinMode(BUTTON_YELLOW, INPUT_PULLUP);
    pinMode(BUTTON_GREEN, INPUT_PULLUP);
    pinMode(LED_RED, OUTPUT);
    pinMode(LED_YELLOW, OUTPUT);
    pinMode(LED_GREEN, OUTPUT);
    Serial.begin(9600); // debug
}

void loop() {
    if (!pressedRed && (releasedRed + 100) < millis() &&
digitalRead(BUTTON_RED) == LOW) {
        Serial.println("press red");
        pressedRed = true;
        red = !red;
    } else if (pressedRed && digitalRead(BUTTON_RED) == HIGH) {
        Serial.println("release red");
        pressedRed = false;
        releasedRed = millis();
    }
    digitalWrite(LED_RED, (unsigned int) red);

    if (!pressedYellow && (releasedYellow + 100) < millis() &&
digitalRead(BUTTON_YELLOW) == LOW) {
        Serial.println("press yellow");
        pressedYellow = true;
        yellow = !yellow;
    } else if (pressedYellow && digitalRead(BUTTON_YELLOW) == HIGH) {
        Serial.println("release yellow");
        pressedYellow = false;
        releasedYellow = millis();
    }
    digitalWrite(LED_YELLOW, (unsigned int) yellow);
```

```
    if (!pressedGreen && (releasedGreen + 100) < millis() &&
digitalRead(BUTTON_GREEN) == LOW) {
        Serial.println("press green");
        pressedGreen = true;
        green = !green;
    } else if (pressedGreen && digitalRead(BUTTON_GREEN) == HIGH) {
        Serial.println("release green");
        pressedGreen = false;
        releasedGreen = millis();
    }
    digitalWrite(LED_GREEN, (unsigned int) green);
}
```



버튼에서 손을 뗀 시간을 기록하여 100ms 동안에는 버튼이 눌려도 반응하지 않도록 구현했다. 코드를 보드에 올려 테스트해보니, 더블클릭 문제가 수정되었다.

## problem3

LED enable 변수와 LED blink status 변수 2 종류를 사용해서 구현했다. LED enable 변수는 Serial 을 통해 깜빡임 시작 명령이 들어왔을 때 활성화되며, blink status 변수는 LED 의 on/off 여부를 나타내며 실제 깜빡임을 구현하는 값이 들어간다. Serial 로 깜빡임 종료 명령이 들어오면 `digitalWrite(LED_RED, LOW);`을 통해 강제적으로 LED 포트의 출력을 LOW 로 만들어, 켜진 상태에서 깜빡임이 종료되지 않도록 한다.