Run µC/OS-II on Arduino

2020.12. 2

Outline

- Create Hello µC/OS-II
- Run Hello µC/OS-II
- Hardware Event

Create Hello µC/OS-II

- Create a new file and then add the following code.

```
/*****************************************************
 *                    Preprocessor
 *****************************************************/
/*  using uCOS-II to manage tasks */
#include <ucos_ii.h>

#define   TASK_STACKSIZE  2048
#define   TASK1_PRIORITY  1
#define   TASK2_PRIORITY  2
#define   TASK3_PRIORITY  3
```

Create Hello µC/OS-II

```
/*********************************************************
 *                        Globals
 *********************************************************/

OS_STK    task1_stk[TASK_STACKSIZE];
OS_STK    task2_stk[TASK_STACKSIZE];
OS_STK    task3_stk[TASK_STACKSIZE];

/*********************************************************
 *                        Prototypes
 *********************************************************/

void task1(void* pdata);
void task2(void* pdata);
void task3(void* pdata);
```

Create Hello µC/OS-II

```
/*****************************************************
 *                      Functions
 *****************************************************/
void task1(void* pdata){
  while (1){
    Serial.println("Hello from task1");
    OSTimeDly( 300 );
  }
}

void task2(void* pdata){
  while (1){
    Serial.println("Hello from task2");
    OSTimeDly( 500 );
  }
}

void task3(void* pdata){
  while (1){
    Serial.println("Hello from task3");
    OSTimeDly( 700 );
  }
}
```

Create Hello µC/OS-II

```
/*****************************************************
 *                      Main
 *****************************************************/
void setup() {
    /* Use serial port for control (board rate : 115200) */
    Serial.begin(115200);
    OSInit();                    /* initialize uCOS */
    OSTaskCreateExt(task1,
          NULL,
          (unsigned int *)&task1_stk[TASK_STACKSIZE - 1],
          TASK1_PRIORITY,
          1,
          task1_stk,
          TASK_STACKSIZE,
          NULL,
          0);
    OSTaskCreateExt(task2,
          NULL,
          (unsigned int *)&task2_stk[TASK_STACKSIZE - 1],
          TASK2_PRIORITY,
          2,
          task2_stk,
          TASK_STACKSIZE,
          NULL,
          0);
```

Create Hello µC/OS-II

```
    OSTaskCreateExt(task3,
          NULL,
          (unsigned int *)&task3_stk[TASK_STACKSIZE - 1],
          TASK3_PRIORITY,
          3,
          task3_stk,
          TASK_STACKSIZE,
          NULL,
          0);

    OSStart();          /* start uCOS */

}
```

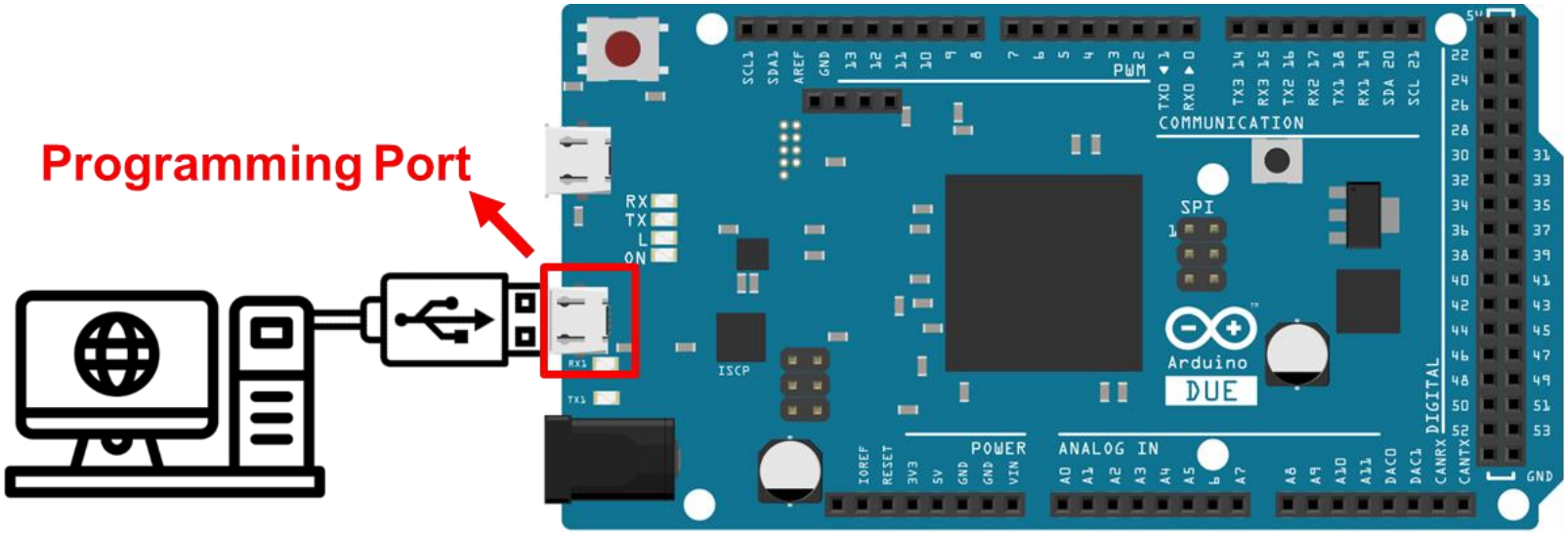
Create Hello µC/OS-II

- Do nothing in loop().

```
/*****************************************************
 *                      Fin
*****************************************************/
void loop() {/* do nothing */}
```

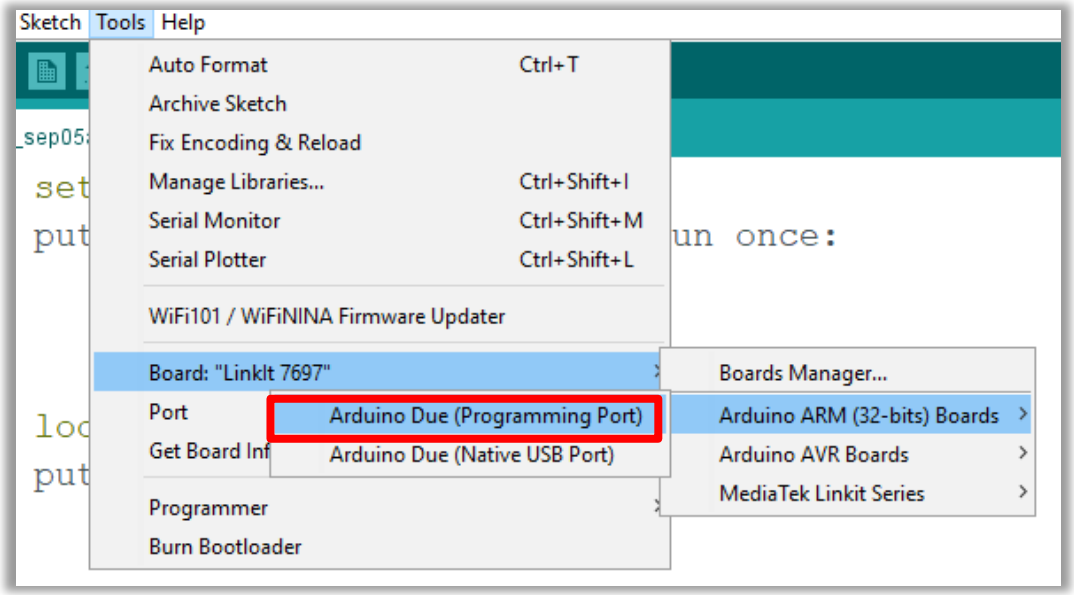
Run Hello µC/OS-II

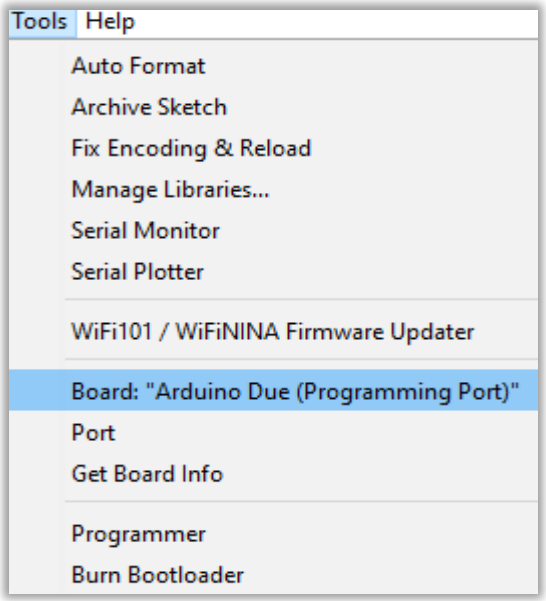
- Connect Arduino DUE to PC using the programming port.



Run Hello µC/OS-II

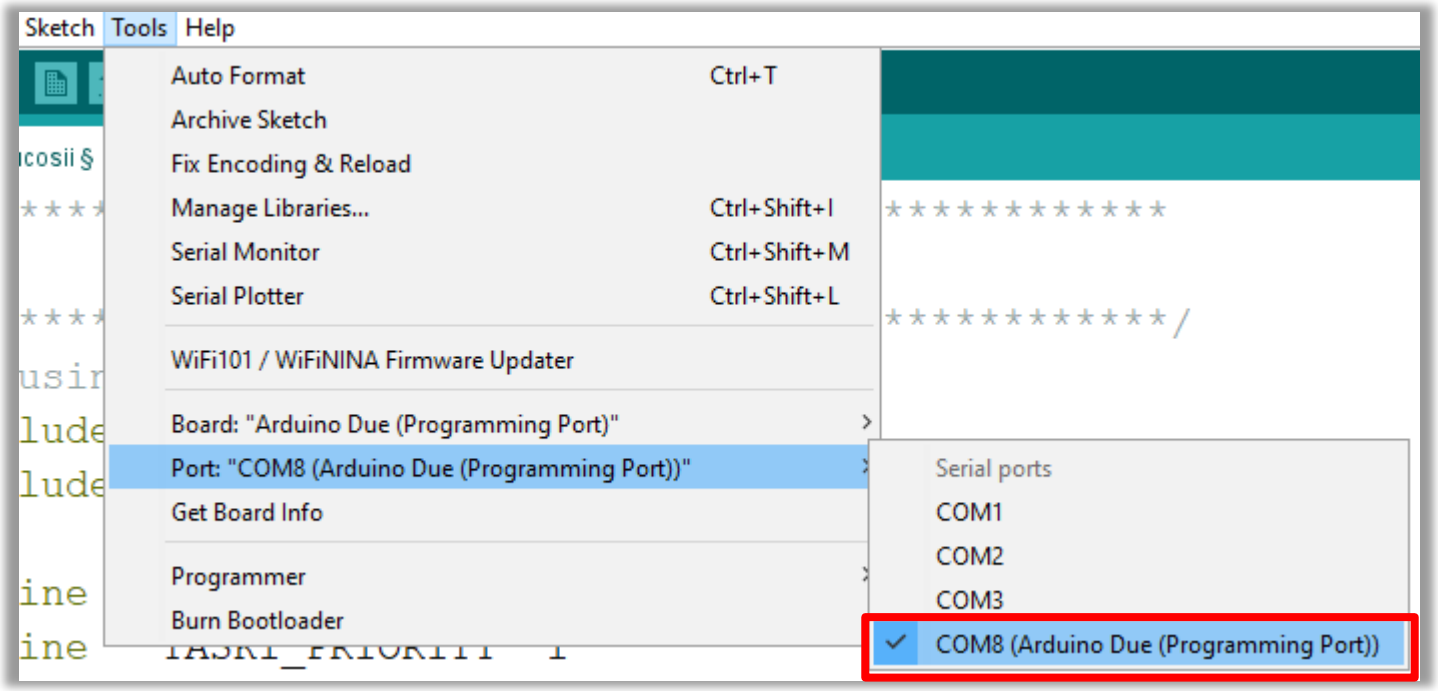
- Select “Programming Port”.





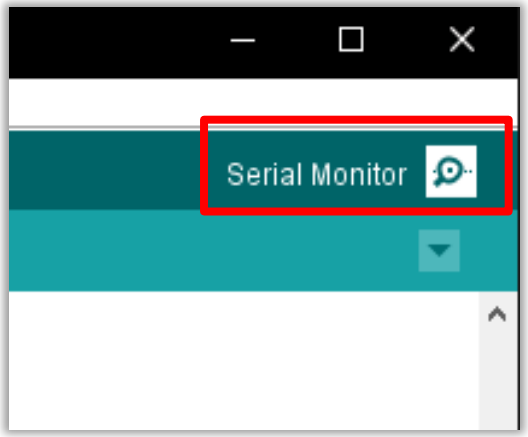
Run Hello µC/OS-II

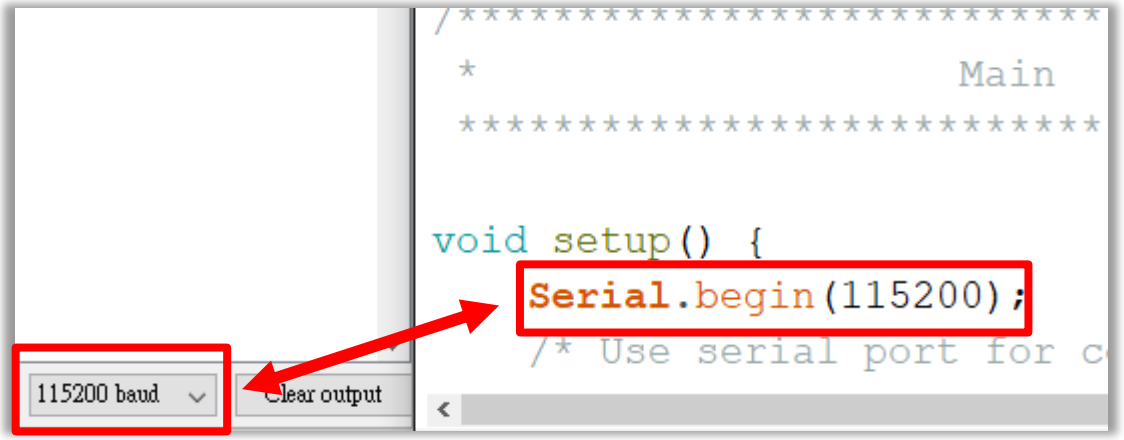
- Select “COM# (Arduino Due)”.



Run Hello µC/OS-II

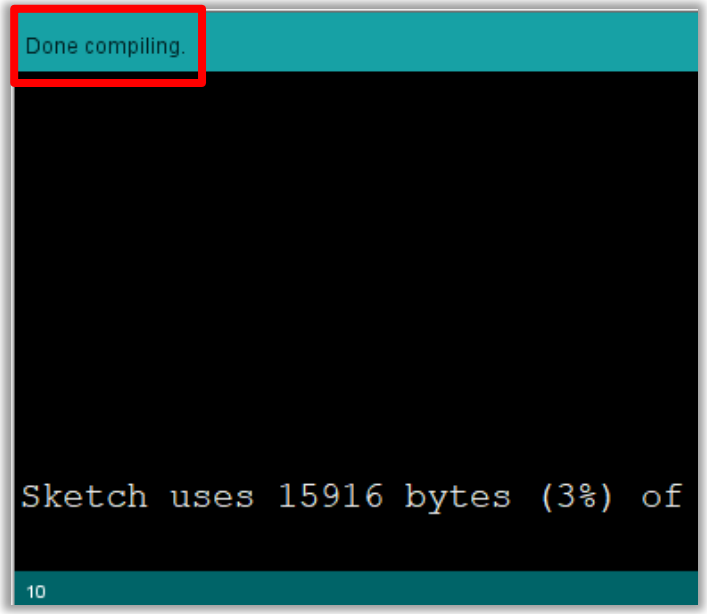
- Open “Serial Monitor” and then change baud rate to 115200.





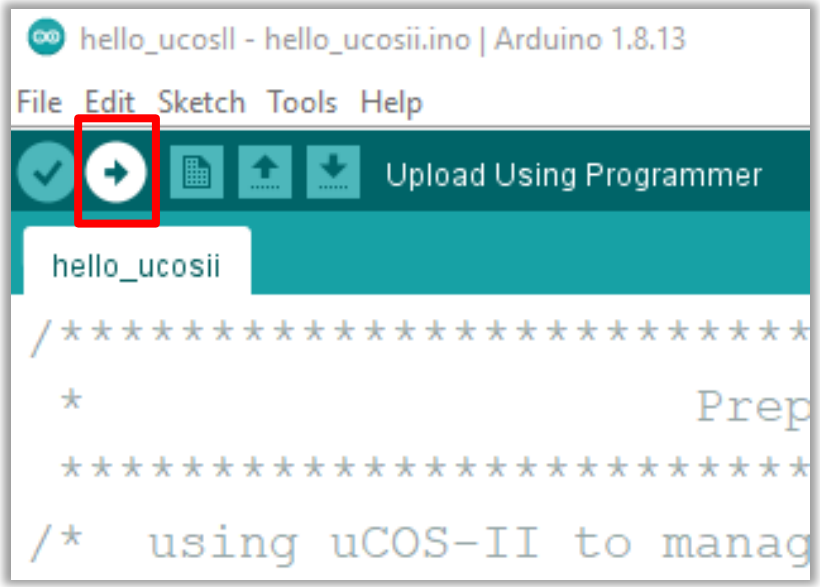
Run Hello µC/OS-II

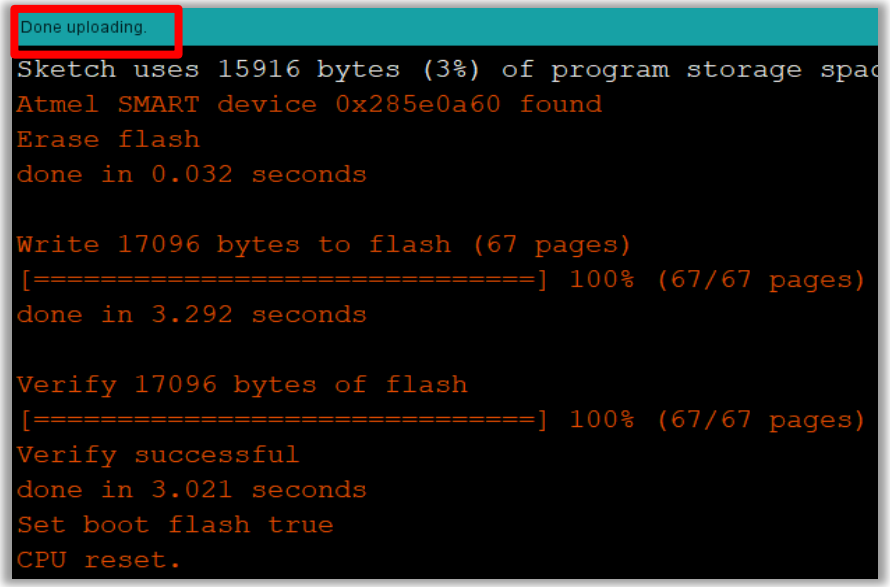
- Verify code.



Run Hello µC/OS-II

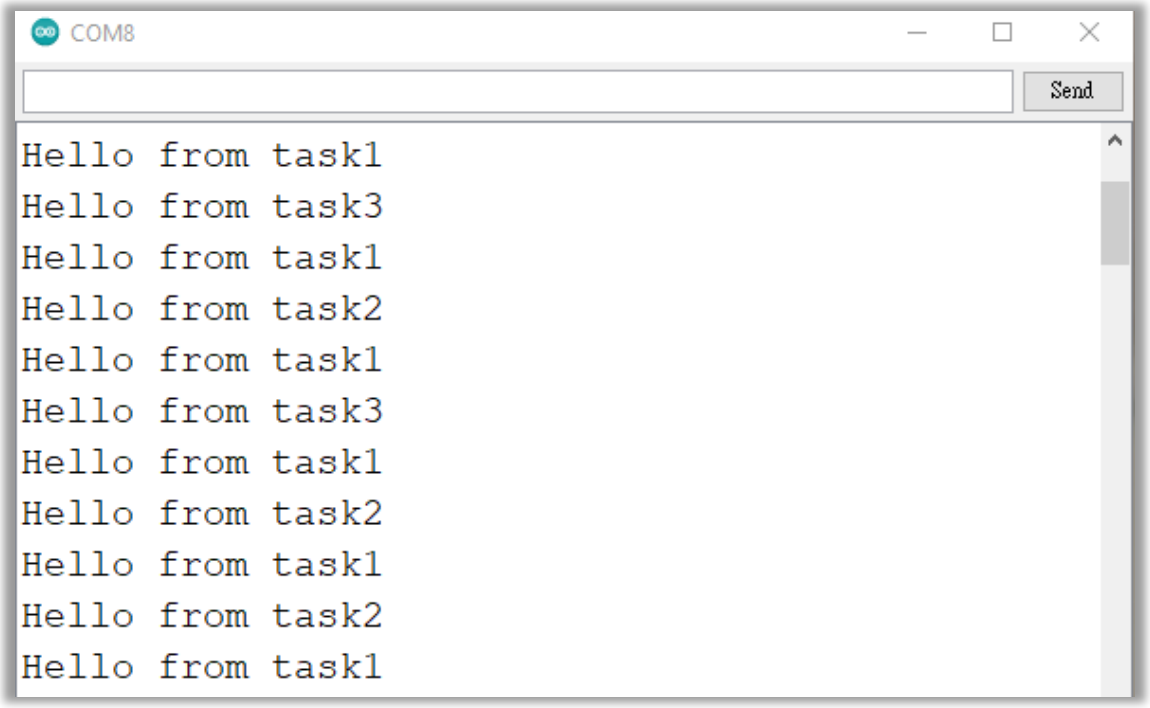
- Upload your project to board.



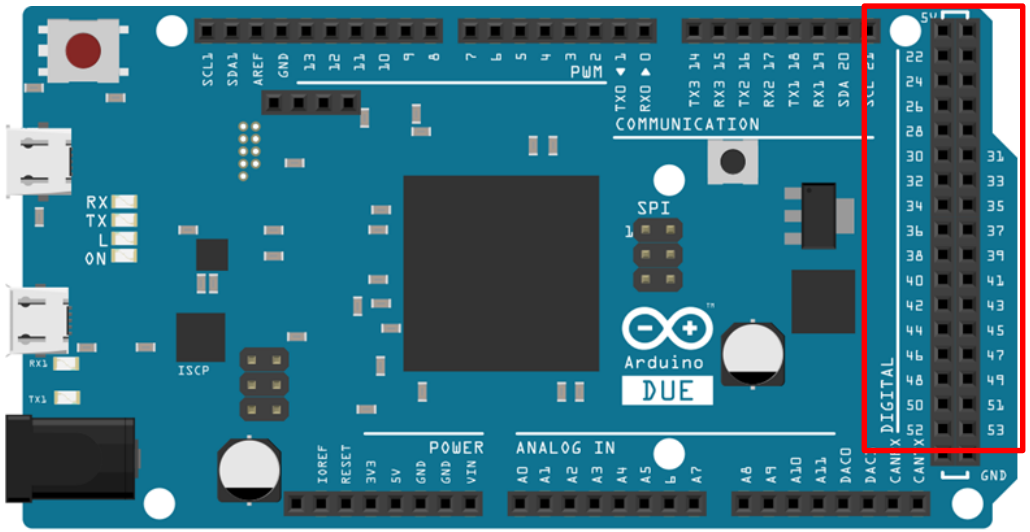


Run Hello µC/OS-II

- Then you can see the results of µC/OS-II running on Arduino board.



Hardware Event



Digital pin: High/Low

Fig. Arduino DUE

These consist of two rows, each column has five holes are connected.

All holes on a row are connected.

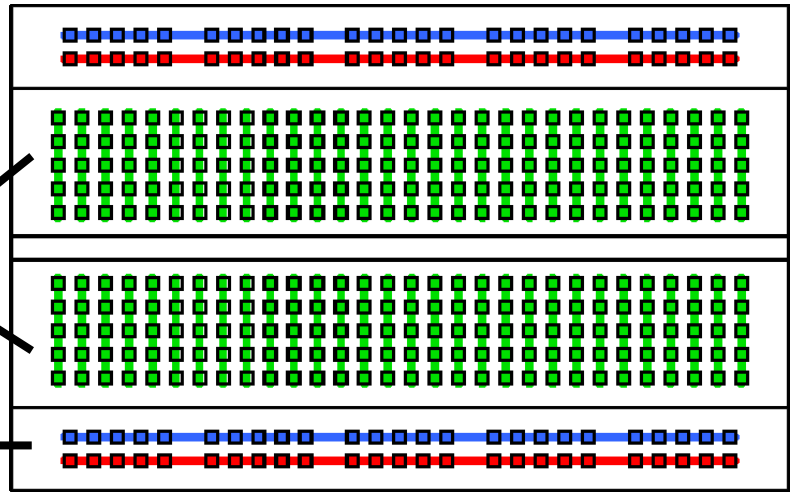
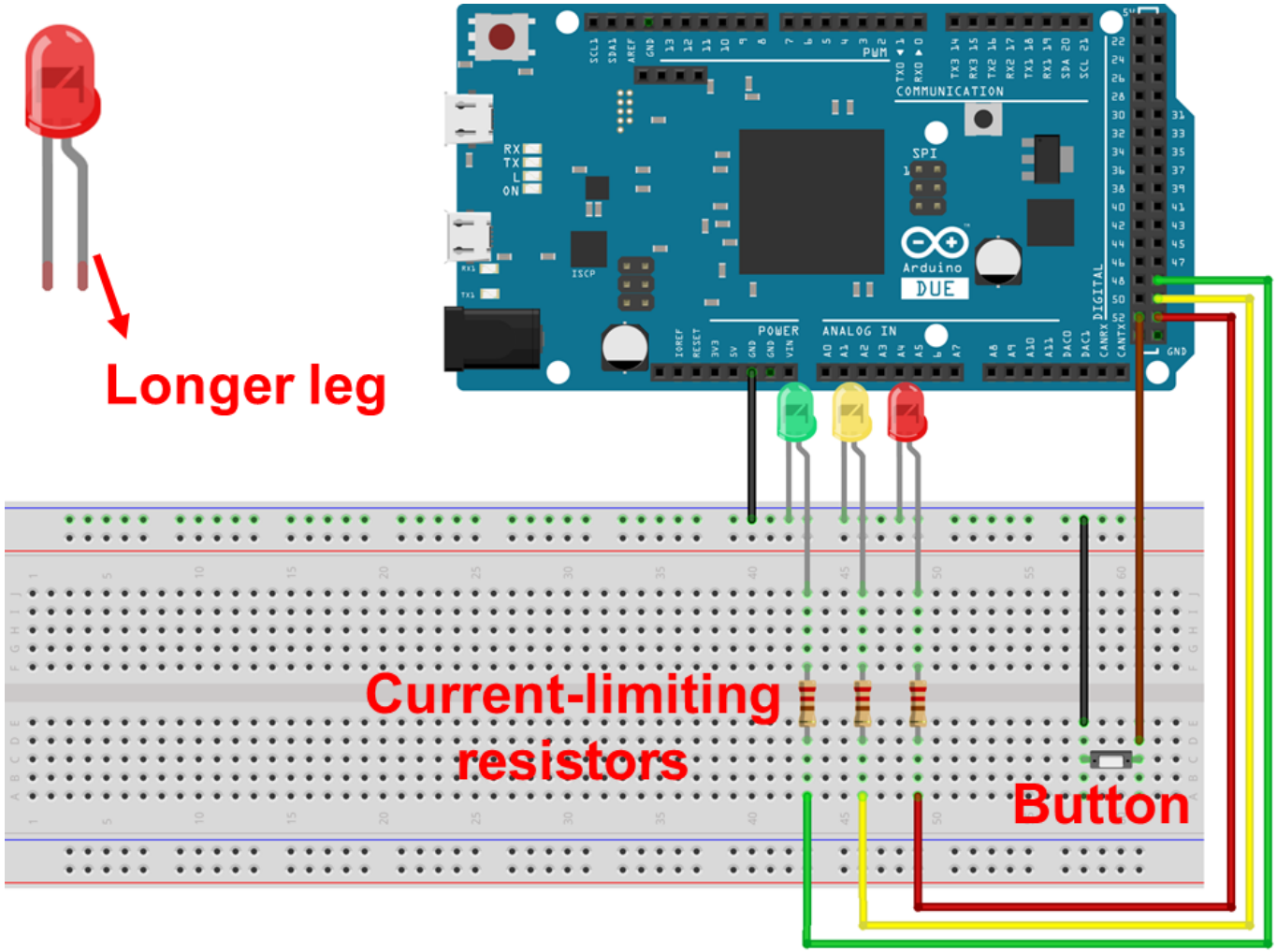


Fig. Breadboard

Hardware Event



Hardware Event – LED Blink

- Declare variable as a digital pin that **YOU** connected the LED.

```
/*****************************************************
 *                      Globals
 *****************************************************/

OS_STK    task1_stk[TASK_STACKSIZE];
OS_STK    task2_stk[TASK_STACKSIZE];
OS_STK    task3_stk[TASK_STACKSIZE];

const byte T1_LEDPin = 49;
const byte T2_LEDPin = 51;
const byte T3_LEDPin = 53;
```

Set these value according the pin you connected.

Hardware Event – LED Blink

- You have to use function **pinMode(pin,mode)** to set the digital pin as output.

```
/*****************************************************
 *                      Main
 *****************************************************/
void setup() {
    Serial.begin(115200);
    /* set Digital Pin to output */
    pinMode(T1_LEDPin, OUTPUT);
    pinMode(T2_LEDPin, OUTPUT);
    pinMode(T3_LEDPin, OUTPUT);
```

Hardware Event – LED Blink

- And write a **HIGH** or a **LOW** value to a digital pin by using function **digitalWrite(pin,value)**.

```
/*****************************************************
 *                    Functions
 *****************************************************/
void task1(void* pdata){
  boolean state = HIGH;
  while (1){
    Serial.println("Hello from task1");
    digitalWrite(T1_LEDPin, state);
    state = !state;
    OSTimeDly( 300 );
  }
}
```

※Modify the other tasks in the same way.

Hardware Event – LED Blink

- Verify your code and upload to board, you can see three LEDs blinking.

Hardware Event – Button Trigger

- Using **Interrupt** to deal with button trigger event.

- Syntax: **attachInterrupt(digitalPinToInterrupt(pin), ISR,mode)[1]**
  - Assign interrupt service routine (ISR).

[1]https://www.arduino.cc/reference/en/

Hardware Event – Button Trigger

- Syntax:
  attachInterrupt(digitalPinToInterrupt(pin), ISR,mode)

- digitalPinToInterrupt(pin) can translate the actual digital pin to the specific interrupt number.

- All digital pins of Arduino DUE are usable for interrupts.

Hardware Event – Button Trigger

- Syntax:
  attachInterrupt(digitalPinToInterrupt(pin), ISR,mode)

- The ISR to call when the interrupt occurs.

- This function **MUST** take **no parameters**, **no other interrupts** and **return nothing**.

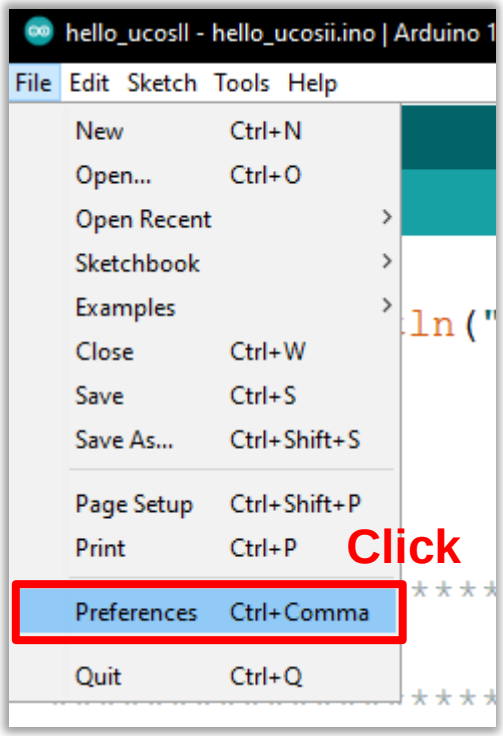
Hardware Event – Button Trigger

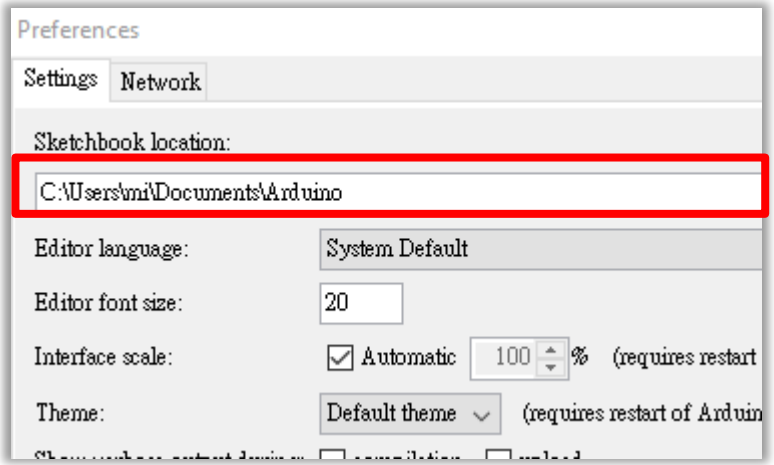
- Syntax:
  attachInterrupt(digitalPinToInterrupt(pin), ISR,mode)

- Mode can define when the interrupt should be triggered.
  - **LOW** : whenever the pin is low
  - **HIGH** : whenever the pin is high
  - **CHANGE** : whenever the pin changes value
  - **RISING** : whenever the pin goes from low to high
  - **FALLING** : whenever the pin goes from high to low

Hardware Event – Button Trigger

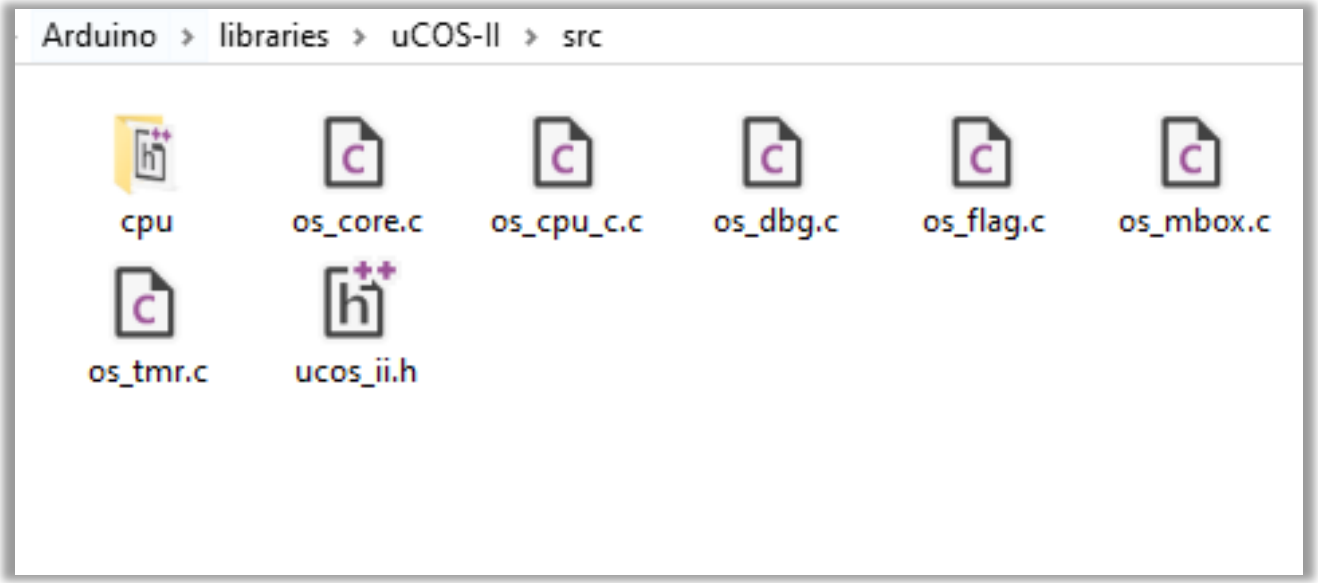
- To find the source file, check "*Sketchbook location*" first.





Hardware Event – Button Trigger

- Source file path:
  [Sketchbook location]\libraries\uCOS-II\src
- Open “os_core.c” by code editor.



Hardware Event – Button Trigger

- Add the following code in “os_core.c”.

```
#define  MICRIUM_SOURCE

#ifndef  OS_MASTER_FILE
#define  OS_GLOBALS
#include "ucos_ii.h"
/********************LAB2 Modified*******************************/
#include "stdio.h"
#include <Arduino.h>
/****************************************************************/
#endif
```



```
/************************LAB2 Modified************************/
const byte interruptPin = 52;
volatile bool push;

void button_trigger() {
    push = true;
}
/************************LAB2 Modified************************/
```

Set this value according the pin you connected.

Define ISR function.

Hardware Event – Button Trigger

- In OSStart function, add pinMode(…) and attachInterrupt(…).

```
void  OSStart (void)
{
/***********************LAB2 Modified***********************/
    push = false;
    pinMode(interruptPin, INPUT_PULLUP);
    attachInterrupt(digitalPinToInterrupt(interruptPin)
                    , button_trigger
                    , FALLING);
/***********************LAB2 Modified***********************/
```

Set button pin to INPUT_PULLUP

Hardware Event – Button Trigger

- In OSTimetick function, add button trigger event.

```
933  void  OSTimeTick (void)
934  {
935  /*********************LAB2 Modified*******************************/
936      if(push) {
937          printf("You pushed button!!\n");
938          push = false;
939      }
940  /*****************************************************************/
```

Hardware Event – Button Trigger

- Verify your code and upload to board, you can see the button trigger event.