Assignment 9

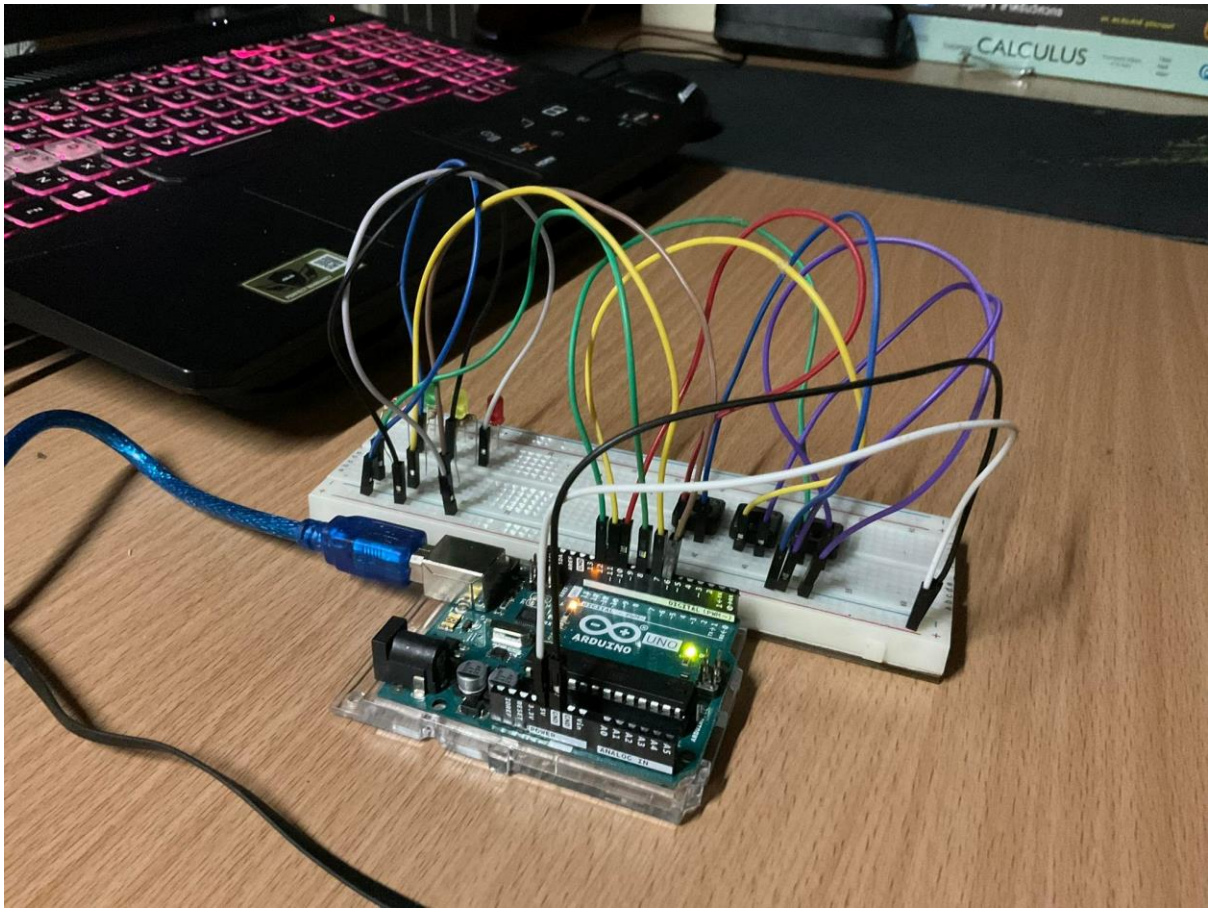
ชื่อกลุ่ม : 9 A.M.

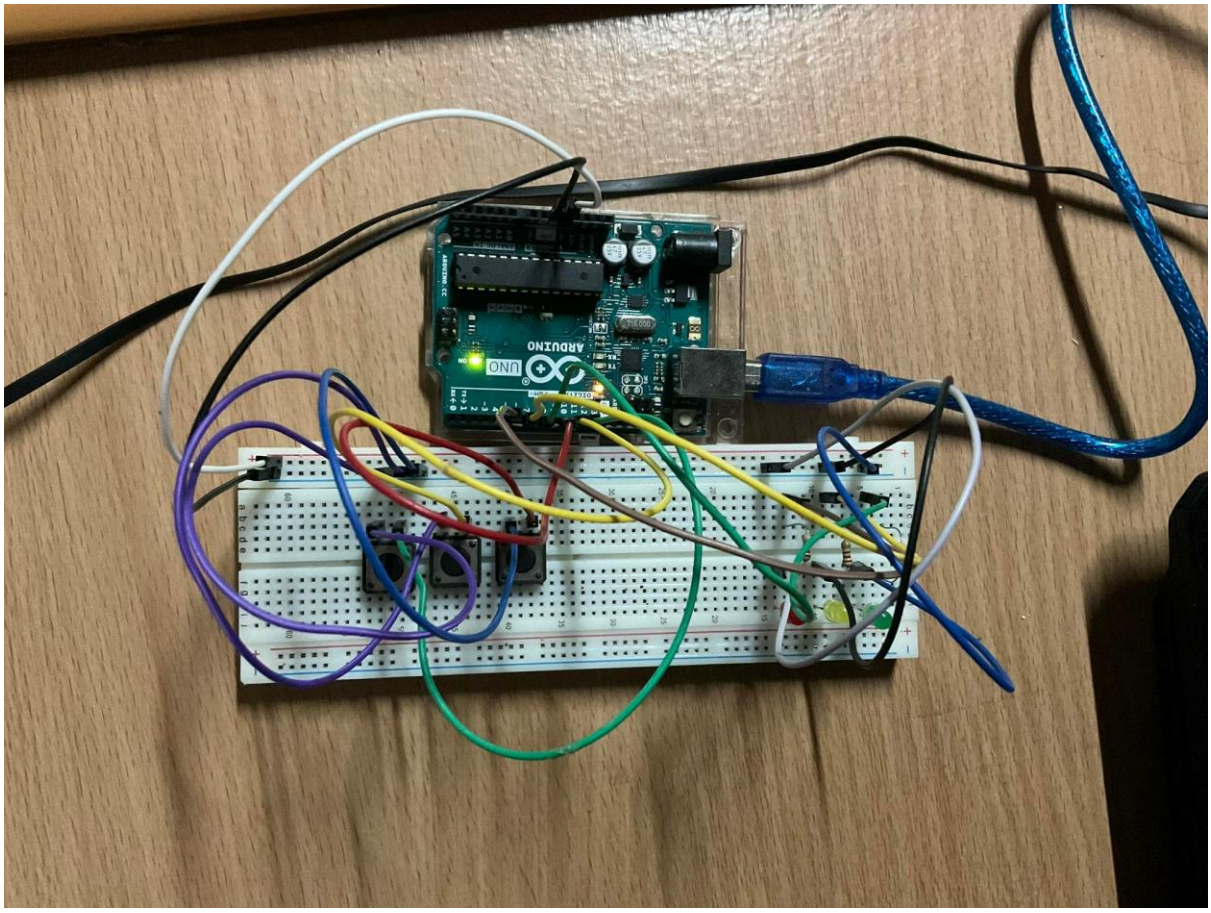
สมาชิกกลุ่ม : 64010761 นายวรพล รังษี

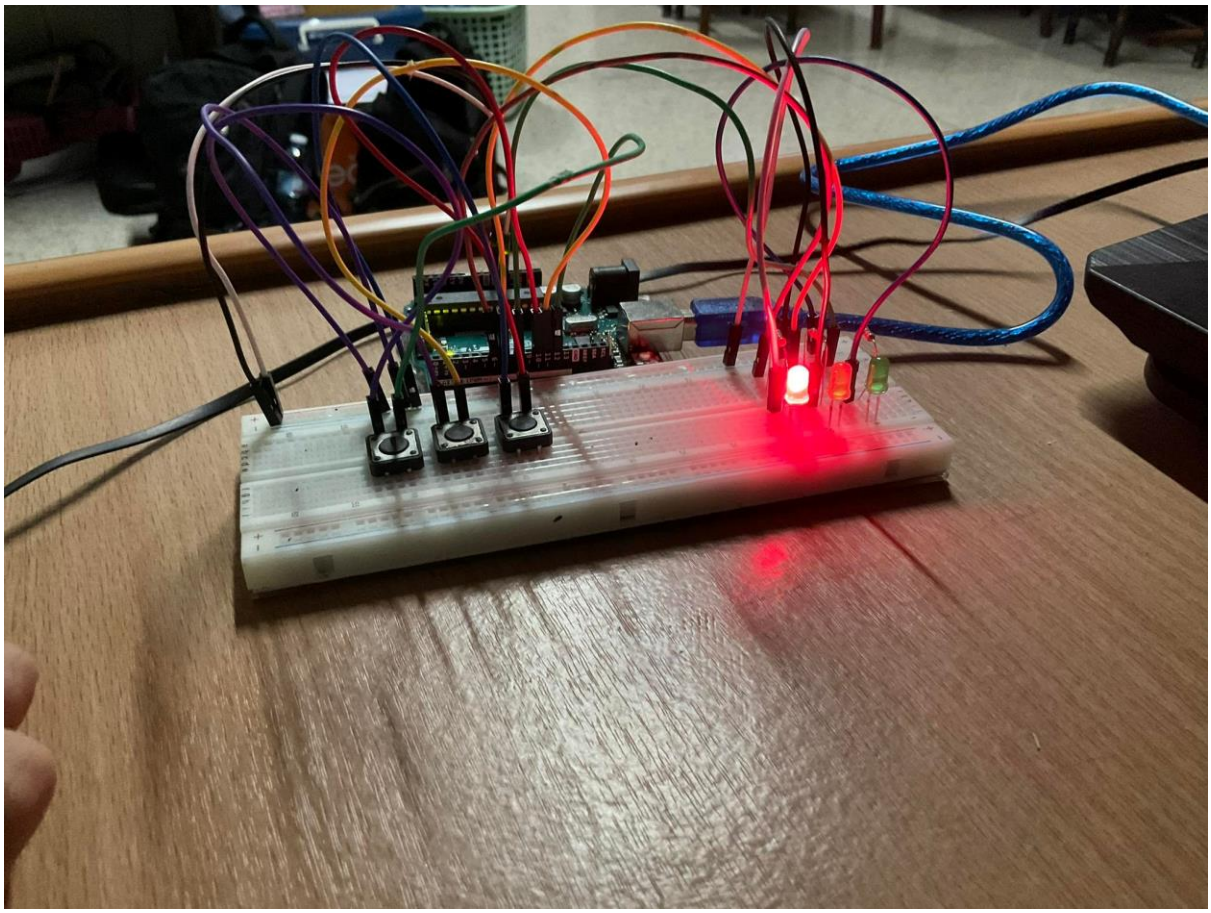
64010757 นายวรโชติ ใจเร็ว

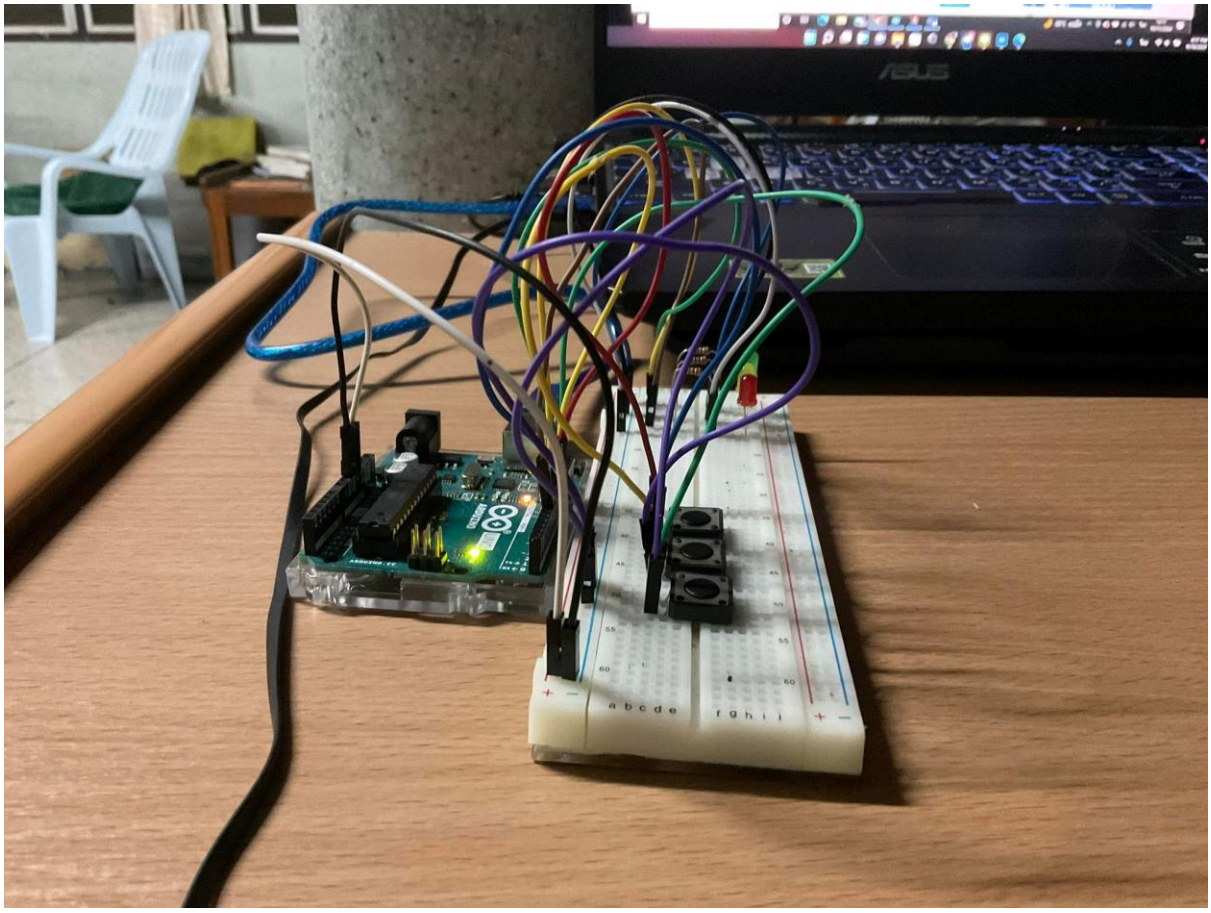
รายละเอียดโปรแกรมโดยย่อ : มีการกำหนด Task, Queue, Semophore ในส่วนของ SetUp โดยในครั้งนี้ ได้มีการนำ Semophore มาใช้เก็บ Token เพื่อกำหนดการให้ความสำคัญของ LED ตามโจทย์

รูปถ่ายชิ้นงาน :




ARDUINO
UNO





Code : https://github.com/worachote1/itc/blob/main/ITC_Assignment_09.ino

```
#include <Arduino_FreeRTOS.h>
#include "queue.h"
#include "semphr.h"//create handle for the mutex. It will be used to
referencemutex

#define RED     6
#define YELLOW  7
#define GREEN   8

#define SW1     12   //control red
#define SW2     11   //control yellow
#define SW3     10   //control green

#define debounce 500

SemaphoreHandle_t redMutex, greenMutex;
QueueHandle_t redQueue, yellowQueue, greenQueue;

//Semaphore เป็นโครงสร้างข้อมูลที่สามารถเก็บสิ่งที่เรียกว่า Tokens
/*
```

```
task สามารถดึง Token ออกจาก Semaphore ได้ ถ้ายังมีเหลืออยู่ แต่ถ้าในขณะนั้น ไม่มีเหลืออยู่ ทาส์กที่ต้องการจะนำ Token ออกไป (Take)
จะถูกเปลี่ยนสถานะจาก RUNNING เป็น BLOCKED เพื่อหยุดรอ และรอจนกว่า จะมี Token ถูกนำมาใส่กลับคืนมา (อย่างน้อยต้องมี 1) ทาส์กดังกล่าวจึงจะสามารถขอ Token ใหม่อีกครั้งได้
*/

//xSemaphoreTake() นำ Token ออกจากเซมาฟอร์
//xSemaphoreGive() นำ Token มาใส่คืนลงในเซมาฟอร์
void setup()
{
  Serial.begin(9600);
  //Serial.print("pdMS_TO_TICKS(3000) = ");
  //Serial.println(pdMS_TO_TICKS(3000));

  redQueue = xQueueCreate(1, sizeof(bool));//xQueueCreate(10,
sizeof(int32_t));
  yellowQueue = xQueueCreate(1, sizeof(bool)); //xQueueCreate(10,
sizeof(int32_t));
  greenQueue = xQueueCreate(1, sizeof(bool)); //xQueueCreate(10,
sizeof(int32_t));
  redMutex = xSemaphoreCreateMutex();
  greenMutex = xSemaphoreCreateMutex();
  xSemaphoreGive(redMutex);
  xSemaphoreGive(greenMutex);

  xTaskCreate(red_button,     "control red btn"   ,    100, NULL, 1, NULL);
  xTaskCreate(yellow_button,  "control yellow btn", 100, NULL, 1, NULL);
  xTaskCreate(green_button,   "control green btn" ,  100, NULL, 1, NULL);

  xTaskCreate(red,            "red",                100, NULL, 1, NULL);
  xTaskCreate(yellow,         "yellow",             100, NULL, 1, NULL);
  xTaskCreate(green,          "green",              100, NULL, 1, NULL);
}

unsigned long  pastTime = 0;
void red_button(void  *pvParameters)
{
  pinMode(SW1, INPUT_PULLUP);
  while (1)
  {
    if (digitalRead(SW1) == LOW && millis() - pastTime >= debounce  )
    {
      pastTime = millis();
      xQueueSend(redQueue, NULL, 0);
    }
    vTaskDelay(10);
  }
}
```

```
void yellow_button(void  *pvParameters)
{
  pinMode(SW2, INPUT_PULLUP);
  while (1)
  {
    if (digitalRead(SW2) == LOW  && xSemaphoreTake(redMutex, 0)== pdTRUE &&
xSemaphoreTake(greenMutex, 0) == pdTRUE && millis() - pastTime >= debounce )
    {
      pastTime = millis();
      xSemaphoreGive(redMutex);
      xSemaphoreGive(greenMutex);
      xQueueSend(yellowQueue, NULL, 0);
    }
    vTaskDelay(10);
  }
}
void green_button(void  *pvParameters)
{
  pinMode(SW3, INPUT_PULLUP);
  while (1)
  {
    if (digitalRead(SW3) == LOW && xSemaphoreTake(redMutex, 0) == pdTRUE &&
millis() - pastTime >= debounce)
    {
      pastTime = millis();
      xSemaphoreGive(redMutex);
      xQueueSend(greenQueue, NULL, 0);
    }
    vTaskDelay(10);
  }
}
void red(void  *pvParameters)
{
  pinMode(RED, OUTPUT);
  while (1)
  {
    if (xQueueReceive(redQueue, NULL, pdMS_TO_TICKS(3000)) == pdPASS)
    {
      //Serial.println("red pressed");
      digitalWrite(RED,!digitalRead(RED) ); // LED is HIGH and LED will be OFF
, if press button again
    }
    else
    {
      digitalWrite(RED, LOW); //LED will be OFF , if time runs out of 3 second
    }
    if (digitalRead(RED)) //if RED LED is HIGH then take token
    {
```

```
      xSemaphoreTake(redMutex, 0);
    }
    else
    {
      xSemaphoreGive(redMutex); //if RED LED is LOW then give token
    }
    vTaskDelay(10);
  }
}
void yellow(void  *pvParameters)
{
  pinMode(YELLOW, OUTPUT);
  while (1)
  {
    if (xQueueReceive(yellowQueue, NULL, 0) == pdPASS)
    {
      for (int i = 1 ; i<=2 ; i++)
      {
        digitalWrite(YELLOW, HIGH);
        vTaskDelay(pdMS_TO_TICKS(500)); //50
        digitalWrite(YELLOW, LOW);
        vTaskDelay(pdMS_TO_TICKS(500));
      }
    }
    vTaskDelay(10);
  }
}
void green(void  *pvParameters)
{
  pinMode(GREEN, OUTPUT);
  while (1)
  {
    if (xQueueReceive(greenQueue, NULL, pdMS_TO_TICKS(3000)) == pdPASS)
    {
      digitalWrite(GREEN, !digitalRead(GREEN));
    }
    else
    {
      digitalWrite(GREEN, LOW);
    }
    if (digitalRead(GREEN))
    {
      xSemaphoreTake(greenMutex, 0);
    }
    else
    {
      xSemaphoreGive(greenMutex);
    }
```

```
    vTaskDelay(10);
  }
}
void loop() {}
```