

# KidBright Project in Action

## สร้างโปรเจคใช้งานจริง

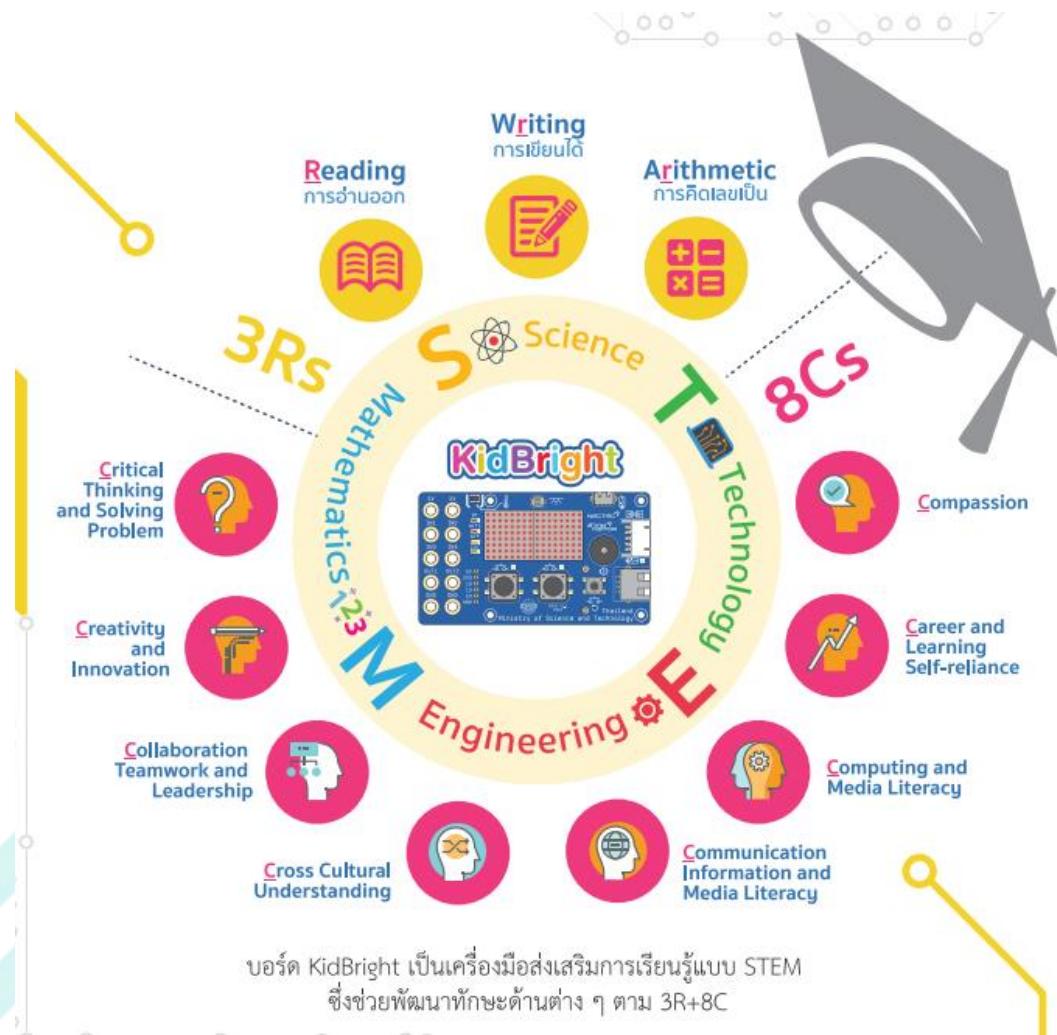
คอมเดช เพ็วดพด @ Khon Kaen Maker Club



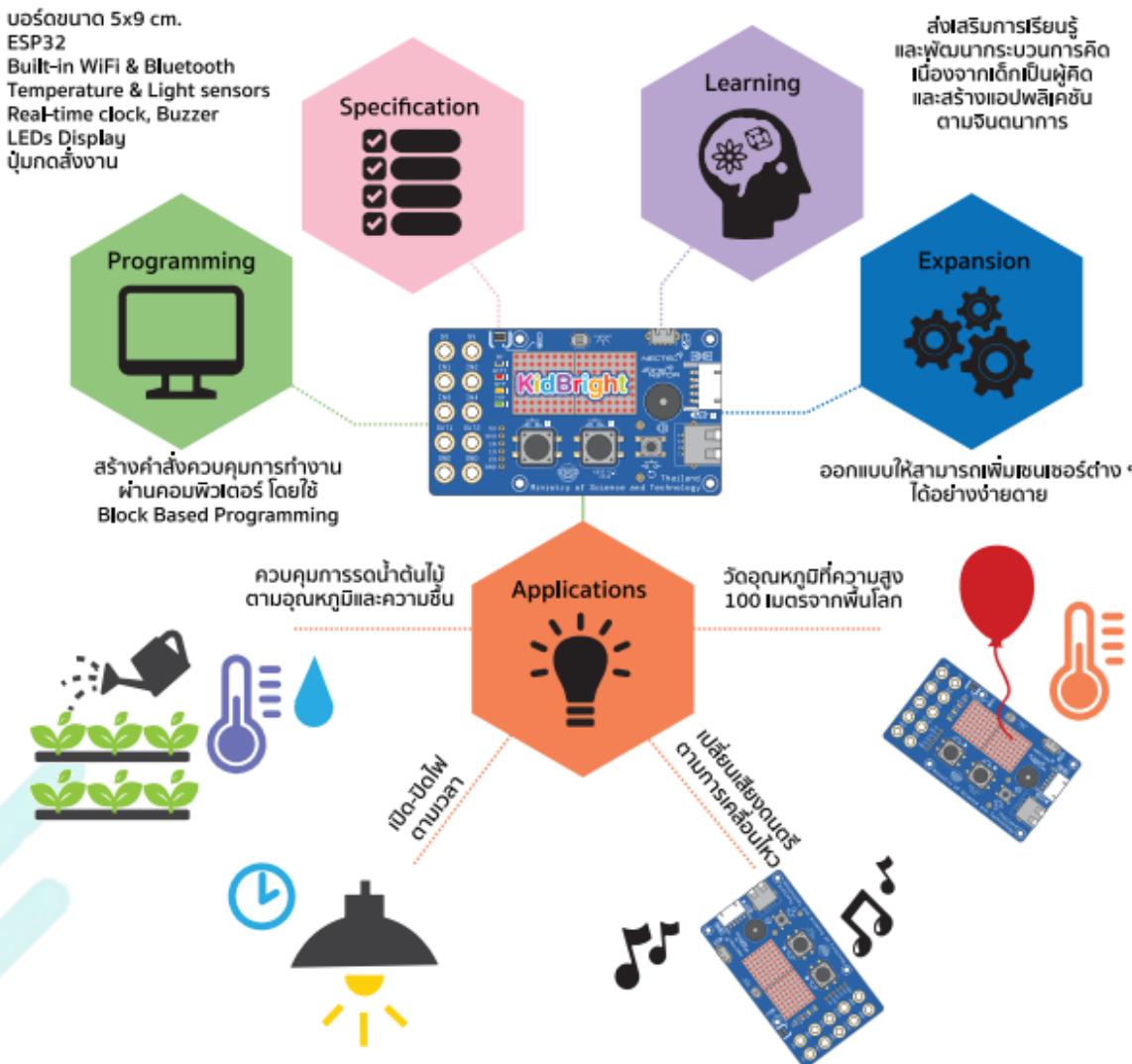
# หัวข้ออบรม

- แนะนำบอร์ด KidBright และการใช้งาน
- การเชื่อมต่อกับโมดูลภายนอกอย่างง่าย
- การทำ Block บน IDE แบบง่าย
- Project brain storm
- การเขียน ESP-IDF เพื่อใช้งานกับ Block
- การเชื่อมต่อกับเซ็นเซอร์แบบมีโปรโตคอลสื่อสาร
- การเขียน ESP-IDF ใช้งาน I2C/SPI ผ่าน KB-Chain
- ปรึกษาโครงการและตัวอย่าง plugin เพิ่มเติม

# รู้จักบอร์ด KidBright

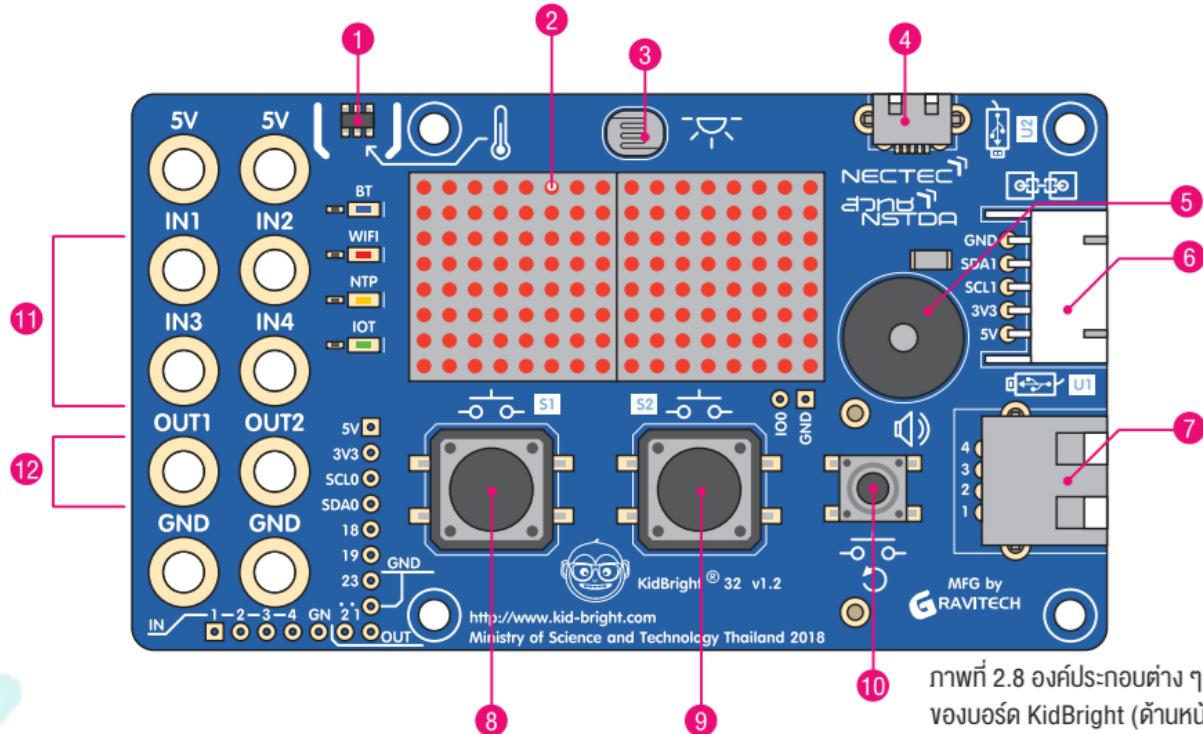


# KidBright เอาไปทำอะไรได้บ้าง ?

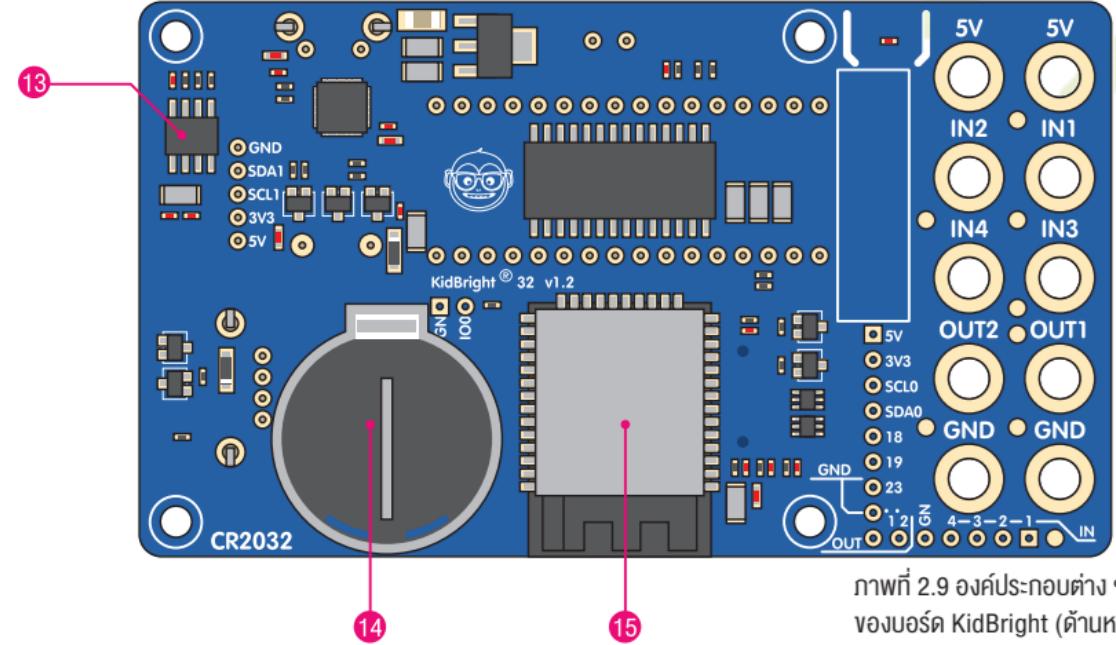


- <https://www.kid-bright.org/showcase/article/250>
- เครื่องเตือนภัยบ้านก่อวัม 24 ชั่วโมง รร.สวงพิกายาคม จ.แพร่
- กรงเลี้ยงชากดอัตโนมัติ รร.อนุบาลแม่ฟ้าหลวง จ.เชียงราย
- เครื่องแยกและนับเหรียญ รร.บ้านทุ่งข้าวพวง จ.เชียงใหม่
- ระบบเฝ้าระวังความปลอดภัยในรถ รร.เชียงคำวิทยาคม จ.พะเยา
- เครื่องควบคุมการจ่ายสารเคมีอัจฉริยะ บ้านทุ่งข้าวพวง จ.เชียงใหม่
- ระบบแจ้งเตือนเหตุอุทกภัย รร.วิทยาศาสตร์จุฬาภรณราชวิทยาลัย ตรัง จ.ตรัง
- บันทึกремเครื่องตากปลาอัจฉริยะ โดยใช้บอร์ด KidBright ss. สกิงพระวิทยา จ.สงขลา
- ระบบควบคุมการดูดบ้านภายในโรงเพาะเพ็ດอัตโนมัติ รร.บ้านปลายคลอง จ.สุราษฎร์ธานี
- “ดูดตัวอิเล็กทรอนิกส์” : การประดิษฐ์เครื่องดูดตัวอิเล็กทรอนิกส์ด้วย KidBright รร. ก้าสาลาประสีกธีศึกษา จ.บุรีรัมย์
- ขวดบ้าแลกเพลง [Music Bin] รร.สาริต มหาวิทยาลัยเกษตรศาสตร์วิทยาเขตกำแพงแสน จ.นครปฐม
- และอีกมากมาย

# ส่วนประกอบของบอร์ด KidBright

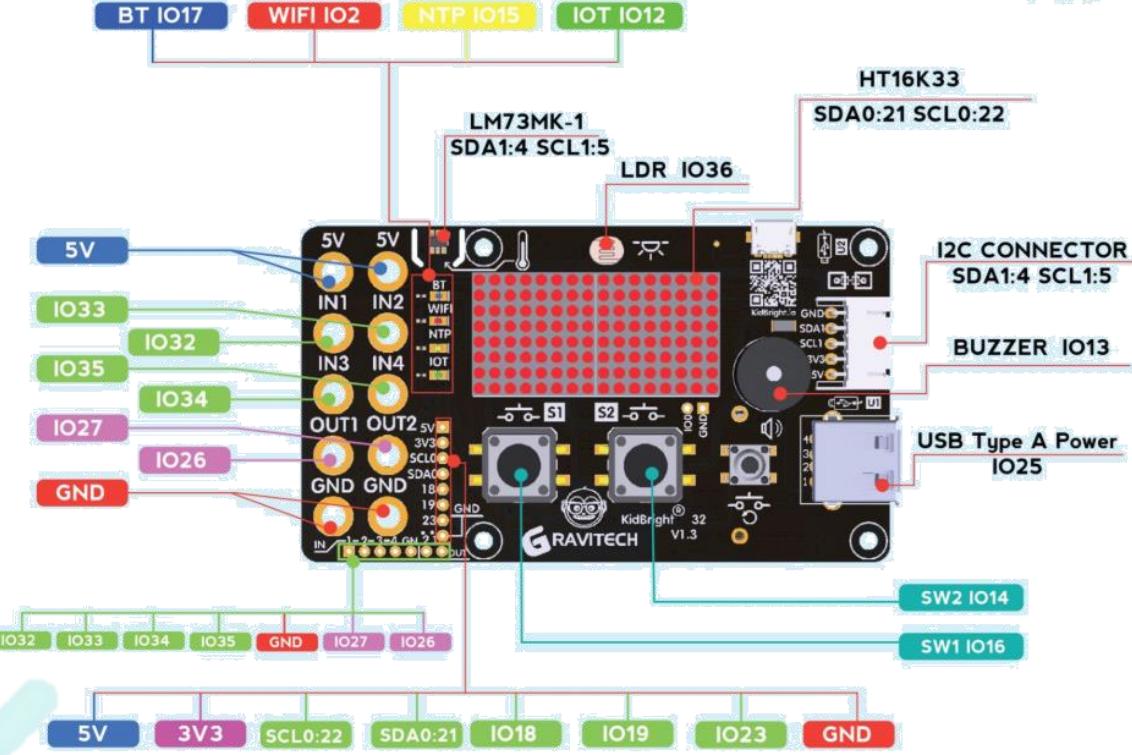


- 1 เชบเซอร์วิดอุณหภูมิ
- 2 LED แสดงผล
- 3 เชบเซอร์วิดแสง
- 4 ช่องเสียบสายไมโครยูเอสบี
- 5 ลำโพง
- 6 คูลเบกเกตอร์
- 7 พอร์ตยูเอสบี
- 8 สวิตซ์ 1
- 9 สวิตซ์ 2
- 10 สวิตซ์รีเซ็ต
- 11 ช่องสัญญาณอินพุต 1-4
- 12 ช่องสัญญาณเอาต์พุต 1-2



- 13 นาฬิกาเรียลไทม์
- 14 รางไส้แบตเตอรี่
- 15 ส่วนควบคุมการทำงาน

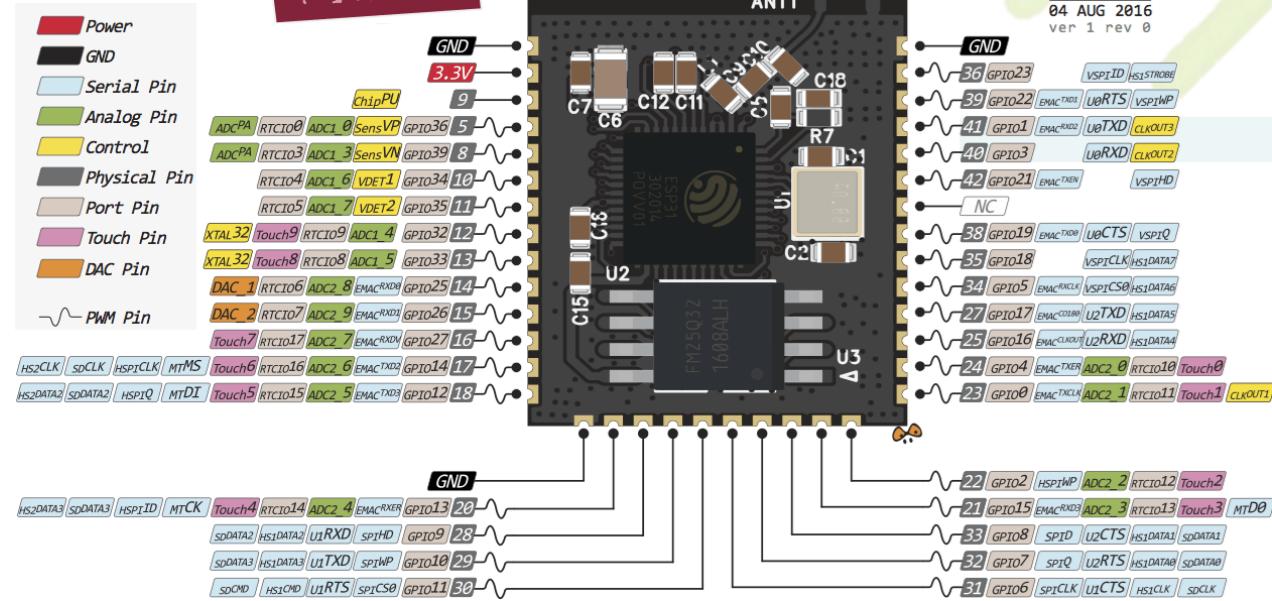
# การเชื่อมต่อบอร์ด KidBright



V1.3



PINOUT



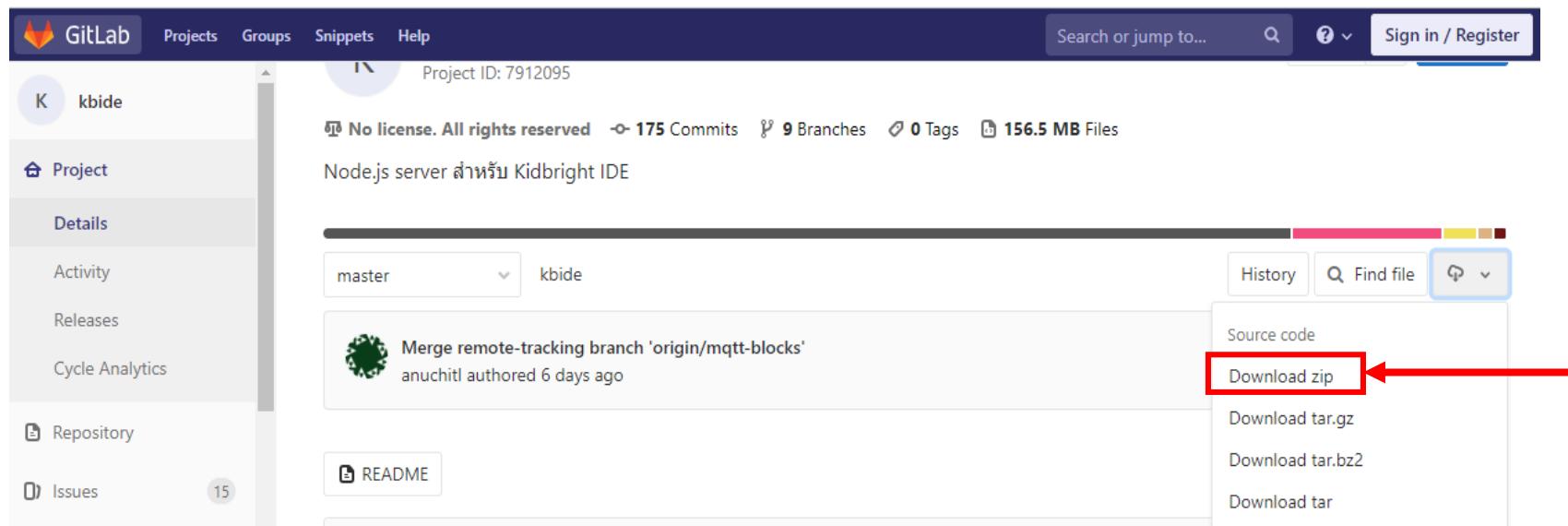
- เขียนโปรแกรมโดยใช้สถาปัตยกรรม ESP32
  - ใช้ ESP-IDF ในการเข้ากับ I/O
    - I/O อิสระ 7 ช่อง (22,21,18,19,23,4,5)
    - Analog ของ ESP32 ควรใช้แค่ ADC1 เหลือ pin (5) pin เดียว

# ทำไมต้องใช้ IDE NodeJS version ?

- ทำการพัฒนาได้รวดเร็วกว่า (เสียเวลาแค่การติดตั้งครั้งแรกเท่านั้น)
- ไม่ต้องรออัพเดตตอนเปิดโปรแกรม
- ติดตั้งได้ทุกที่ copy folder ได้
- เมื่อโปรแกรม error สามารถรับให้ได้ทันที (ตัว IDE หลักไม่มี error handling)
- ค้นหาไฟล์และแก้ไขได้ง่าย
- สามารถเขียนโค้ดได้หลายหน้าต่าง (เปิด Incognito โหมดบนเว็บ)
- สามารถเปิดเขียนโค้ดแบบหลายเครื่องได้ ถึงแม้จะมีบอร์ดแค่อันเดียว !

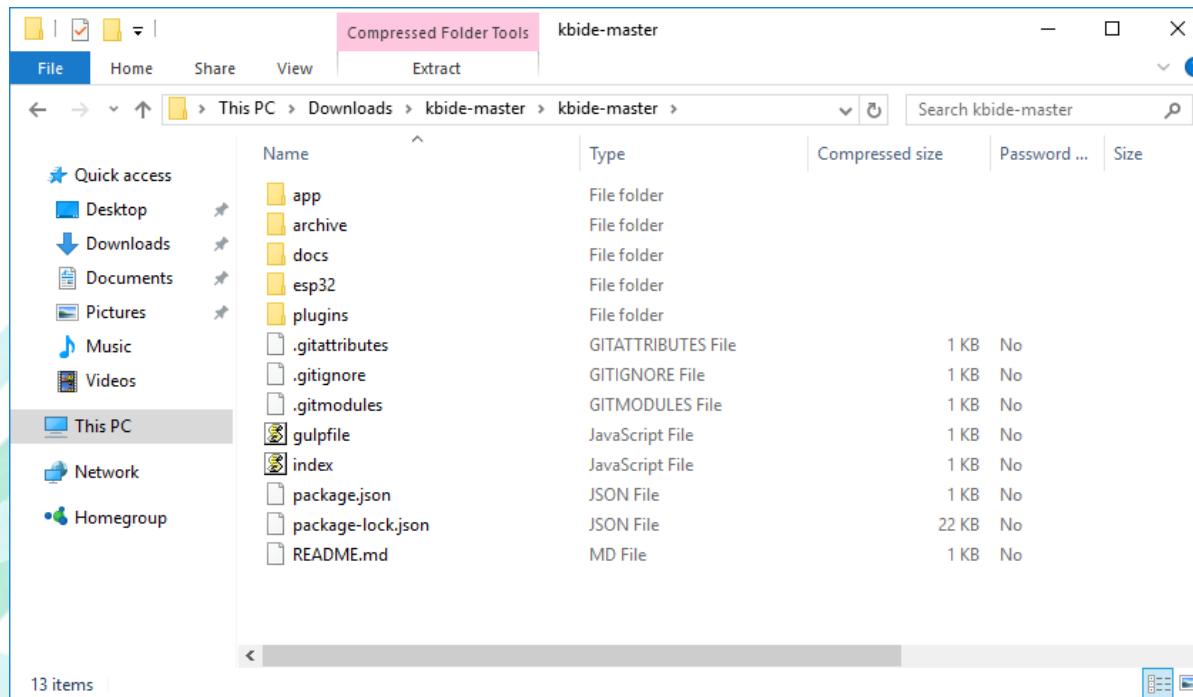
# การติดตั้ง IDE NodeJS version (ต่อ)

- ทำการติดตั้ง NodeJS จาก <https://nodejs.org/en/download>
- ทำการดาวน์โหลด IDE มาจาก <https://gitlab.com/kidbright/kbide/>



# การติดตั้ง IDE NodeJS version (ต่อ)

- เปิด Command Prompt ทำการ cd เข้าไปยัง folder “kbide-master” ที่แตกไฟล์ไว้
- รันคำสั่ง npm run build
- ตรวจสอบว่าจะติดตั้งสำเร็จ



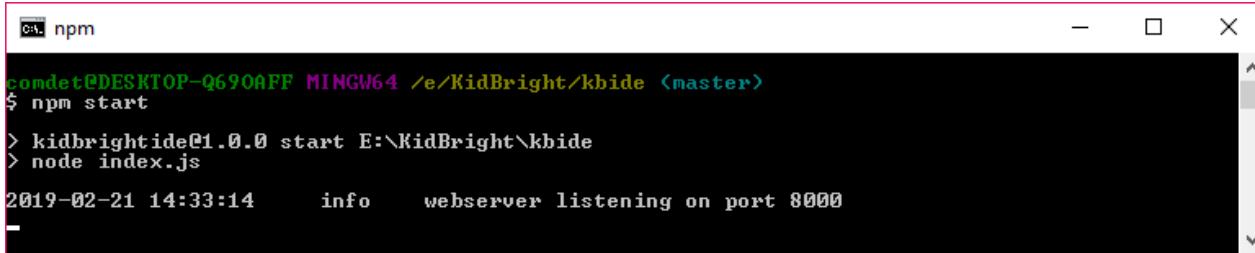
```
C:\Users\comdet\Desktop\kbide-master>npm run build

[22:31:11] Finished 'download_xtensa' after 1.23 min
[22:31:11] Starting 'download_esptool'...
[22:31:13] Finished 'download_esptool' after 2.24 s
[22:31:13] Starting 'decompress'...
[22:33:19] Finished 'decompress' after 2.08 min
[22:33:19] Starting 'chmod_linux'...
[22:33:19] Finished 'chmod_linux' after 135 µs
[22:33:19] Starting 'del'...
[22:33:19] Finished 'del' after 183 ms
[22:33:19] Starting 'build'...
[22:33:19] Finished 'build' after 56 µs

C:\Users\comdet\Desktop\kbide-master>
```

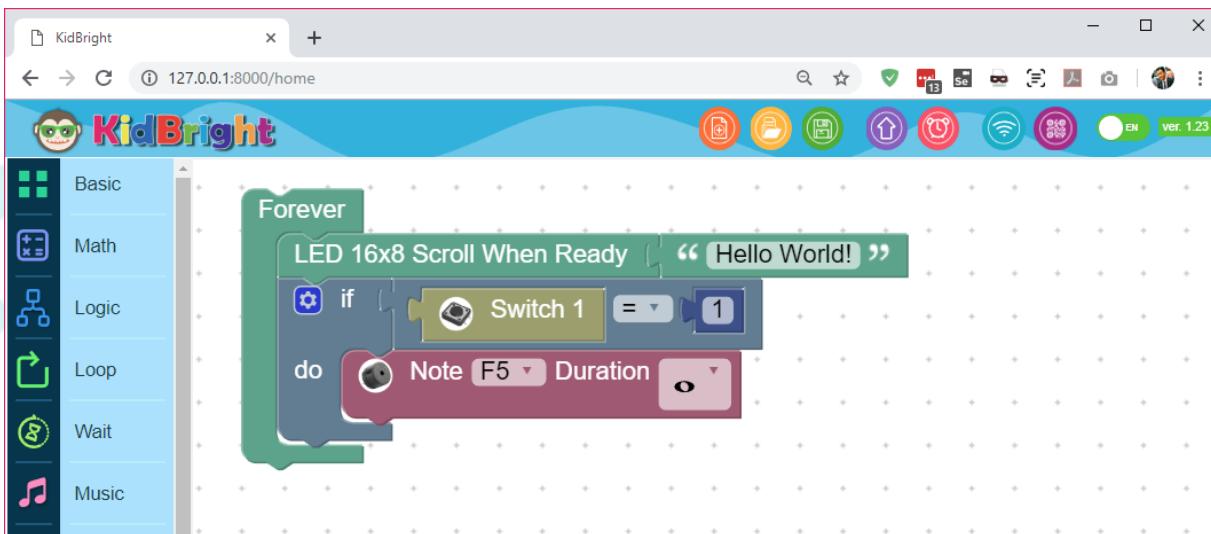
# ทดสอบเบื้องต้นโปรแกรมแสดงข้อความ

- รันคำสั่ง npm start ที่ command prompt



```
npm
comdet@DESKTOP-Q690AFF MINGW64 /e/KidBright/kbide <master>
$ npm start
> kidbrightide@1.0.0 start E:\KidBright\kbide
> node index.js
2019-02-21 14:33:14    info    webserver listening on port 8000
```

- ทำการเข้า Web Browser <http://localhost:8000> หรือ <http://127.0.0.1:8000>
- ทำการทดสอบเบื้องต้นอย่างง่าย



# พักผ่อน 10 บาท

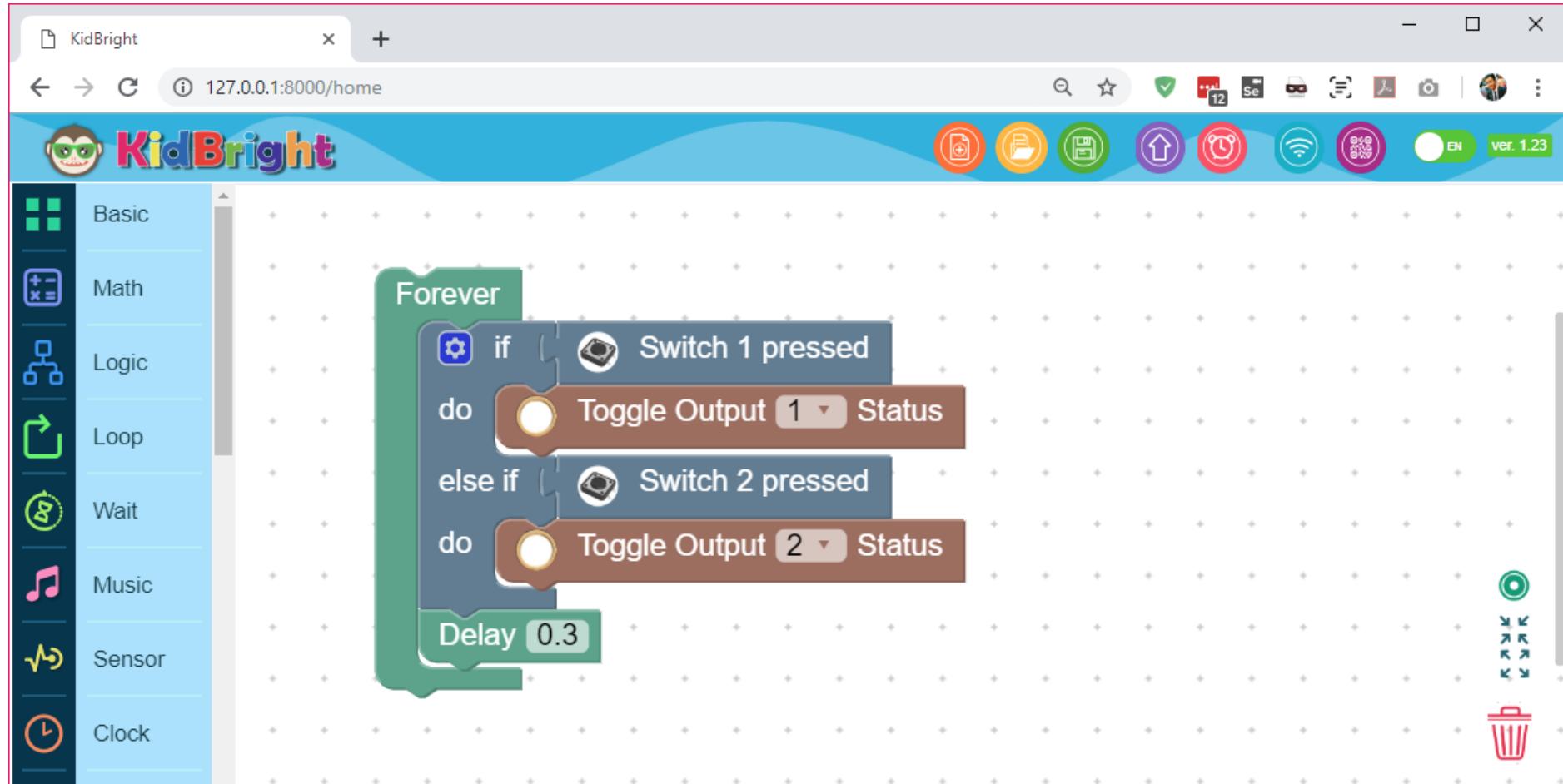


# Lab 1 : เครื่องตั้งเวลา crud นำตัวบันทึก

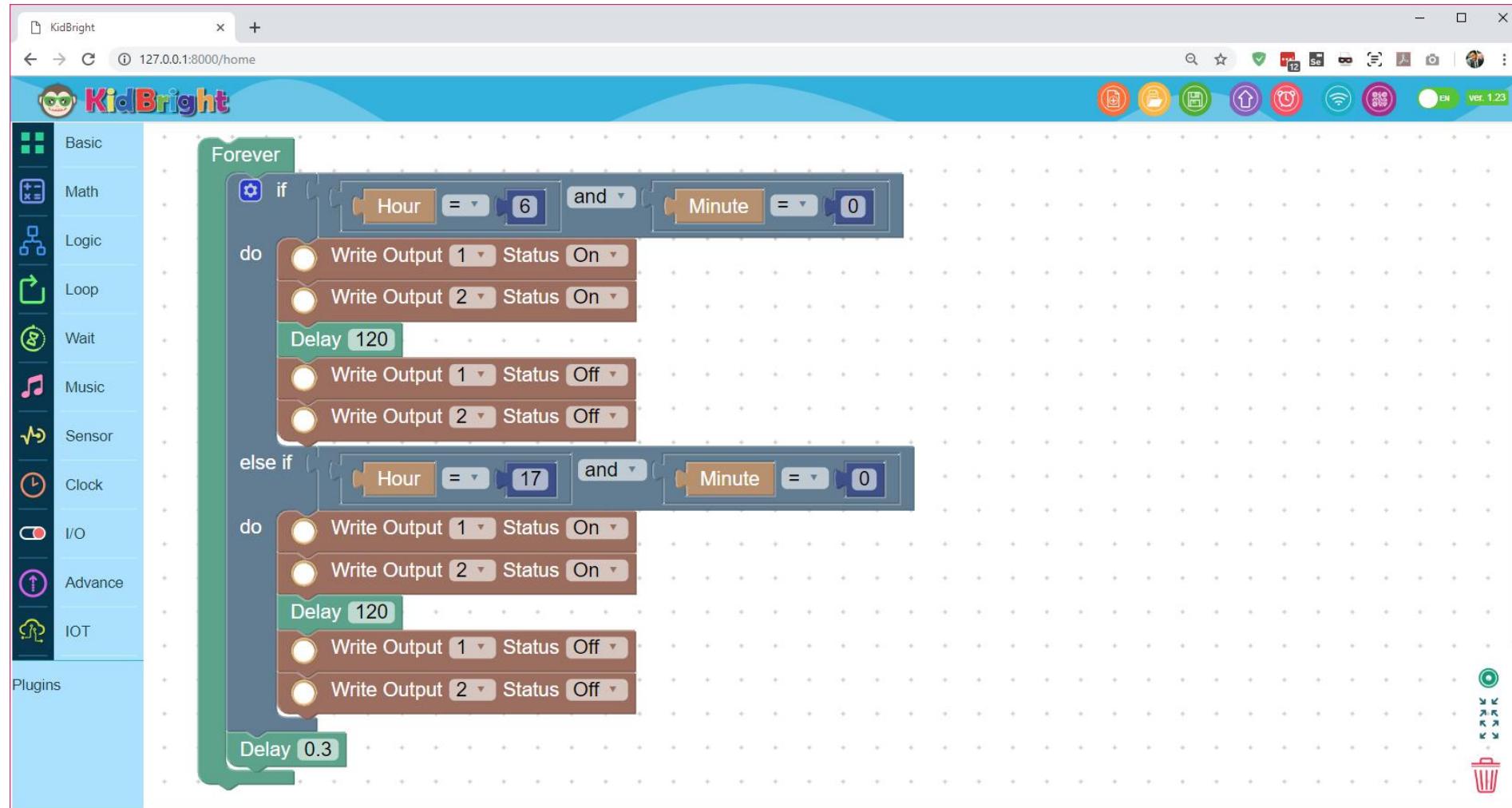
- เป้าหมาย : ใช้งาน Digital Input แบบง่าย + ใช้งาน Timer ในบอร์ด
- โมดูล : Relay
- วิธีการทำ : ทำการเชื่อมต่อ relay เข้ากับบอร์ดโดยไฟเลี้ยง 5V และ GND ใช้ช่อง OUT1 และ OUT2 เพื่อควบคุมการปิด/เปิด เข้า IN1 IN2 ของ Relay
- โจทย์ :
  - ข้อ 1 เมื่อต่อสำเร็จ ให้เขียนโปรแกรมเพื่อสั่งการ โดยหากกด S1 ให้ Relay CH1 ติด กดอีกครั้งเพื่อดับ และ S2 ให้กำลังจะเดียวกันกับ Relay CH2
  - ข้อ 2 ทำการเขียนโปรแกรมเพิ่ม โดยให้ relay ทำงานตอน 6 โบนเข้าและ 5 โบนเย็บ เป็นเวลา 2 นาที แล้วปิดการทำงาน
  - ข้อ 3 เขียนโปรแกรมให้พูดตั้งเวลาปิดเปิดเองได้ โดยกด S1 เพื่อตั้งค่า ชั่วโมง S1 เพื่อตั้ง นาที
  - ข้อ 4 ทำการแก้ไขปัญหาของโค้ดข้อ 3



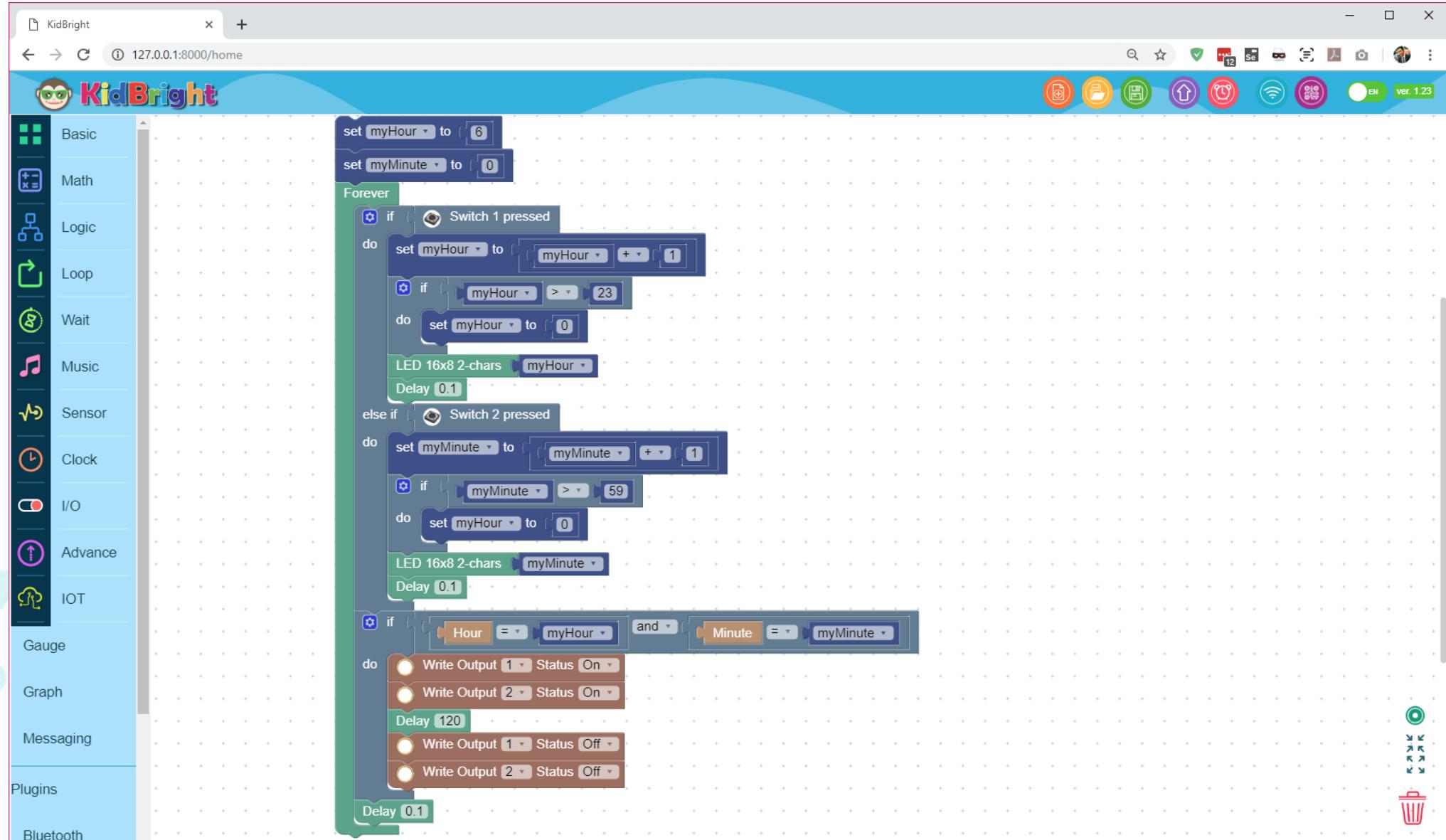
# Lab 1 : เครื่องตั้งเวลา Ardunio ใหม่ (วิธีทำ ข้อ 1) {kb\_ans1}



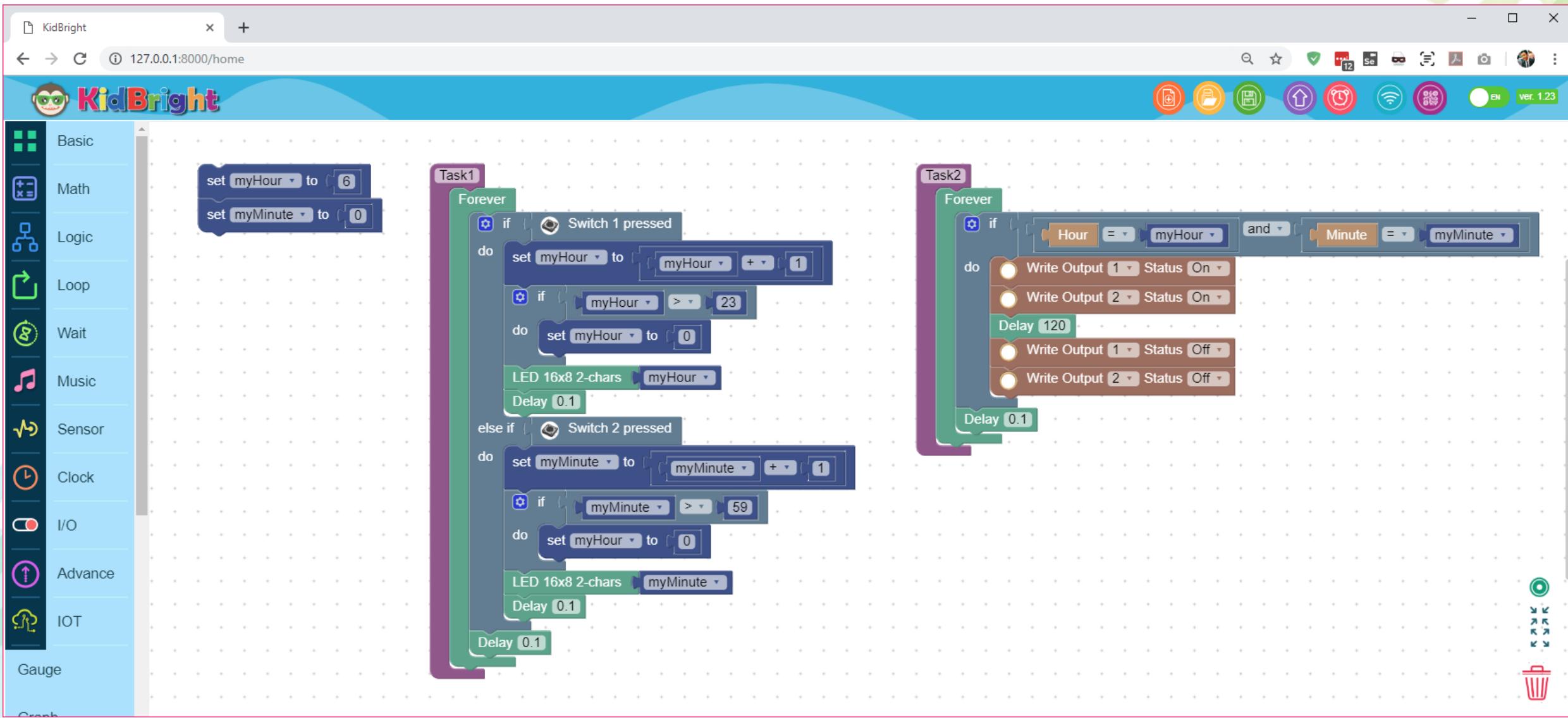
# Lab 1 : เครื่องตั้งเวลา Ardunio ใหม่ (วิธีทำ ข้อ 2) {kbbnk48}



# Lab 1 : เครื่องตั้งเวลา crud นำตัวบันทึก (วิธีทำ ข้อ 3) {kidbite}



# Lab 1 : เครื่องตั้งเวลา/run นำตัวบิน (วิธีทำ ข้อ 4){kidbike}

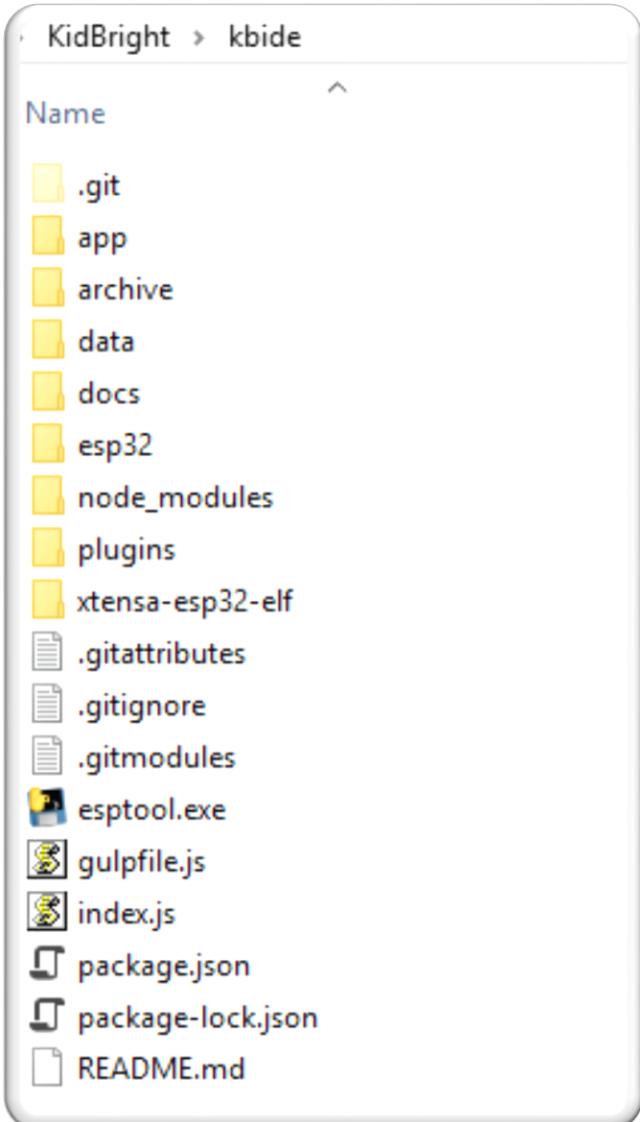


# รู้จักกับ ESP IoT Development Framework (ESP-IDF)

The screenshot shows the official documentation for the ESP-IDF (Espressif IoT Development Framework) on the ESP32 platform. The top navigation bar includes links for 'ESP-IDF Programming Guide', 'latest' version, and a search bar. Below this, there are sections for 'Get Started' and 'API Reference'. The 'API Reference' section is expanded, showing categories like 'Bluetooth', 'Networking', 'Peripherals', and 'H/W Reference'. Under 'Peripherals', a detailed list of components is provided, including ADC, CAN, DAC, GPIO (including RTC low power I/O), I2C, I2S, LED Control, MCPWM, Pulse Counter, Remote Control, SDMMC Host, SD SPI Host, SDIO Slave, Sigma-delta Modulation, SPI Master, SPI Slave, Timer, and Touch Sensor.

- เป็นชุดคอมไพล์เลอร์ ESP32 ที่พัฒนาโดยบริษัท Espressif
- ทำงานครอบคลุมบน Free Real-time OS (FreeRTOS)
- ทำการแก้ไขดัดแปลงของ ESP8266 SDK ได้เกือบทั้งหมด
- Arduino ESP32 SDK ทำขึ้นมาครอบ ESP-IDF อีกรอบ เพื่อให้ใช้คำสั่ง Arduino ได้
- ทำการ Build ได้รวดเร็วกว่า Arduino SDK มาก
- ESP32 รุ่นใหม่ต้องเขียนด้วย ESP-IDF เก่านั้นไม่สามารถใช้ RTOS-SDK version เก่า ๆ ได้
- IDF เป็น component base เป็นหลัก
- ข้อมูลการพัฒนาและเอกสารการใช้งานทั้งหมดสามารถหาเพิ่มเติมได้ผ่าน <https://docs.espressif.com/projects/esp-idf/en/latest/>

# โครงสร้างของ KBIDE



- KBIDE เขียนด้วย JavaScript แยกเป็นฟัง Server กับ UI โดยคัดกันหมดเก็บอยู่ใน folder “app”
- ใน folder “esp32/build/xx-xx-xx..” จะมีไฟล์ user\_app.cpp ซึ่งคัดโปรแกรมภาษา C ที่ได้บล็อก และ xx-xx-xx.bin คือ firmware (xx-xx-xx.. คือหมายเลข Serial แต่ละบอร์ด)

The screenshot shows the code for user\_app.cpp in Sublime Text:

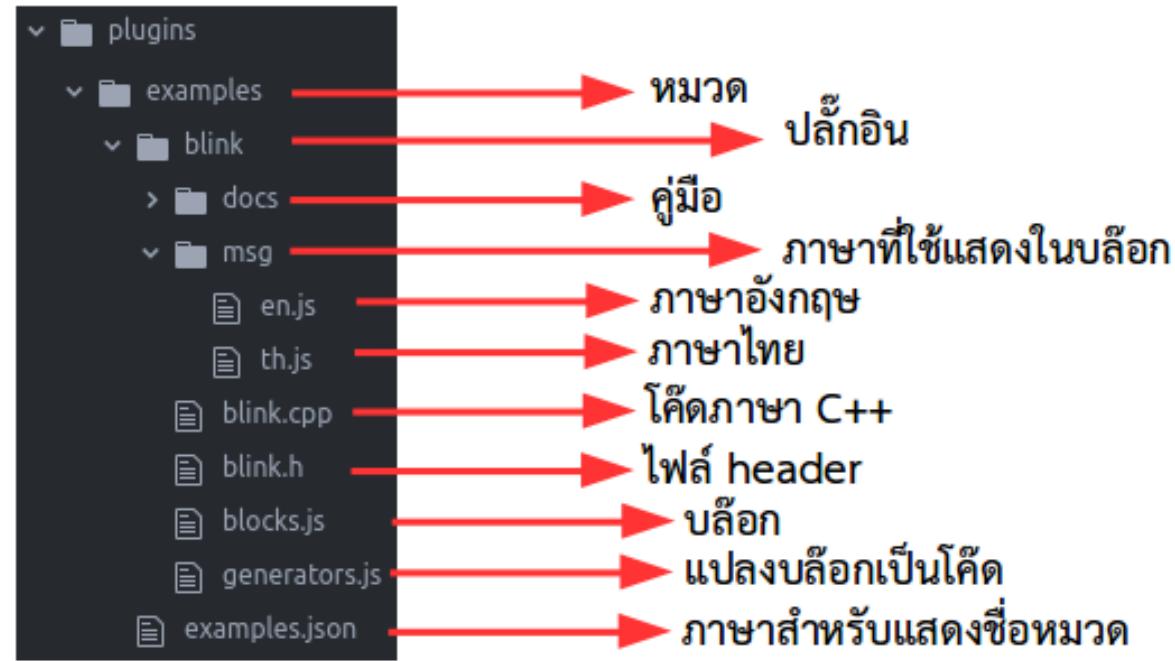
```
52 void user_app(void) {
53     xTaskCreatePinnedToCore(iotTask, "iotTask", 4096, NULL, USERAPP_TASK_PRIORITY, NULL, KIDBRIGHT_RUNNING_CORE);
54     srand(mcp7940n.get(5));
55     // setup
56     myHour = (double)6;
57     myMinute = (double)0;
58     while(1) {
59         if (get_B1stateClicked() || button12.is_sw1_pressed()) {
60             myHour = myHour + (double)1;
61             if (myHour > (double)23) {
62                 myHour = (double)0;
63             }
64             ht16k33.scroll(myHour, false);
65             vTaskDelay(100 / portTICK_RATE_MS);
66         } else if (get_B2stateClicked() || button12.is_sw2_pressed()) {
67             myMinute = myMinute + (double)1;
68             if ((myMinute > (double)59) {
```

Line 59, Column 61

Tab Size: 4 C++

- ใน folder “plugins” เป็นที่เก็บปลั๊กอินของบล็อกที่เสริมเข้าไป เราสามารถสร้าง บล็อก เองได้โดยสร้างไฟล์ไว้ที่ folder นี้
- “xtensa-esp32-elf” เป็นคอมไพล์เลอร์ของบอร์ด ESP32

# โครงสร้างของปลั๊กอิน KBIDE



- ปลั๊กอินคือ “กลุ่มบล็อค” มีหน้าที่ในการ “สร้างกลุ่มโค้ด” เพื่อนำไปสร้างเป็นโปรแกรมที่สมบูรณ์
- หนึ่งหมวดหมู่อาจมีได้หลายปลั๊กอิน เช่น หมวดหมู่ Temperature อาจมีปลั๊กอินของ Sensor DHT11, DS18B20, DHT22 เป็นต้น
- การเขียนปลั๊กอินต้องเขียนภาษา JavaScript ร่วมกันภาษา C++ ตามโครงสร้างของ ESP-IDF

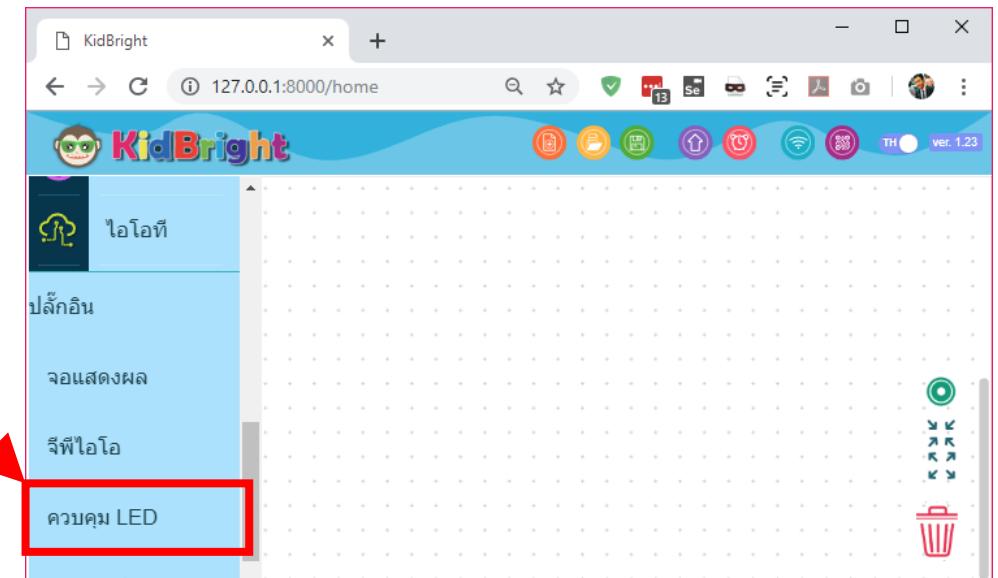
# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย

- เป้าหมาย : สร้างปลั๊กอินเสริมอย่างง่ายขึ้นมาเพื่อควบคุม LED ในบอร์ด โดยมี Block สำหรับสั่งเปิดไฟกระพริบ และอีก Block สำหรับสั่งปิดไฟกระพริบ
- วิธีการทำ :
  - ใน folder “plugins” ทำการสร้าง folder “led”
  - เข้าไปที่ folder “led” ที่สร้างขึ้น ทำการสร้างไฟล์ led.json โดยข้อมูลในไฟล์นี้จะเป็นตัวที่แสดงข้อความบน IDE โดยกำหนดค่าดังนี้

```
File Edit Selection View Go Debug Terminal Help
EXPLORER
OPEN EDITORS 1 UNSAVED
UNTITLED (WORKSPACE)
led
led.json
1 {
  "name": {
    "en": "LED control",
    "th": "ควบคุม LED"
  },
  "color": 160
}
```

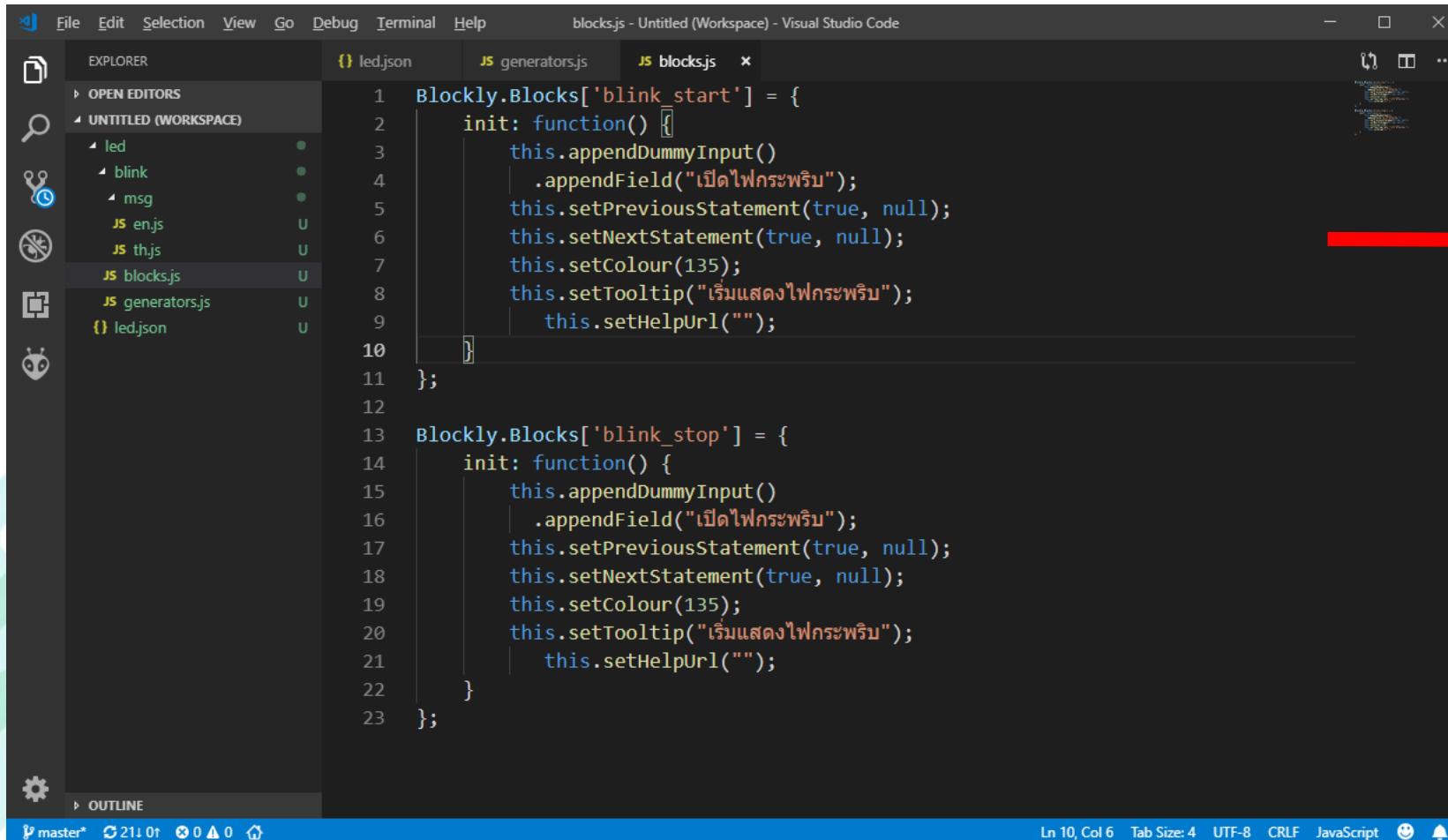
The screenshot shows a code editor window with the title "led.json - Untitled (Workspace)". The code editor displays the following JSON content:

```
{ "name": { "en": "LED control", "th": "ควบคุม LED" }, "color": 160 }
```



# Lab 2 : ทำปลั๊กอินไฟกระพริบอย่างง่าย (ต่อ 1.)

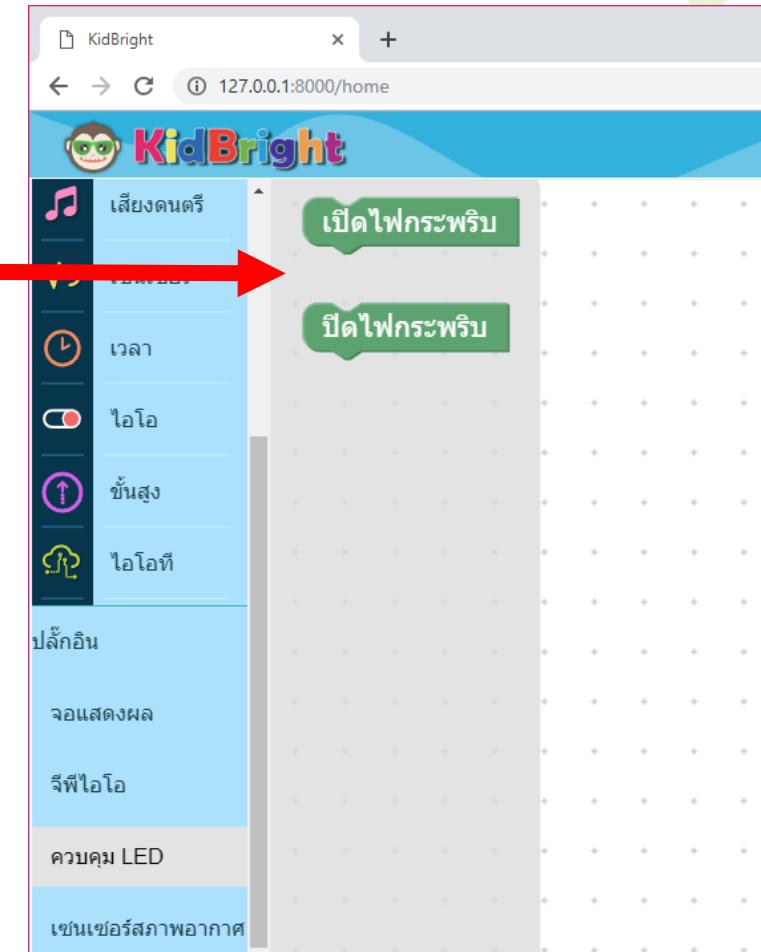
- ทำการสร้าง folder “blink” ขึ้นมา และเข้าไปในนั้น ทำการสร้าง folder “msg” เข้าไปสร้างไฟล์ en.js และ th.js ในนั้น (สร้างไฟล์เปล่า ๆ ยังไม่ต้องใส่เนื้อหาในไฟล์)
- ทำการสร้างไฟล์ blocks.js ใน folder “blink” และใส่โค้ดในไฟล์ดังนี้



```
File Edit Selection View Go Debug Terminal Help
blocks.js - Untitled (Workspace) - Visual Studio Code

EXPLORER
OPEN EDITORS
UNTITLED (WORKSPACE)
led.json generators.js blocks.js
1 Blockly.Blocks['blink_start'] = {
2     init: function() {
3         this.appendDummyInput()
4             .appendField("เปิดไฟกระพริบ");
5         this.setPreviousStatement(true, null);
6         this.setNextStatement(true, null);
7         this.setColour(135);
8         this.setTooltip("เริ่มแสดงไฟกระพริบ");
9         this.setHelpUrl("");
10    }
11 };
12
13 Blockly.Blocks['blink_stop'] = {
14     init: function() {
15         this.appendDummyInput()
16             .appendField("ปิดไฟกระพริบ");
17         this.setPreviousStatement(true, null);
18         this.setNextStatement(true, null);
19         this.setColour(135);
20         this.setTooltip("เริ่มแสดงไฟกระพริบ");
21     }
22 };

Ln 10, Col 6 Tab Size: 4 UTF-8 CRLF JavaScript OUTLINE master* 21:0t 0 0 0 0 0
```



# Lab 2 : กำลังก็อบ ไฟกระพริบอย่างง่าย (ต่อ 2.)

- ทำการสร้างไฟล์ generators.js ขึ้นมาใน folder “blink” และใส่โค้ดในไฟล์ดังนี้



A screenshot of Visual Studio Code showing the "generators.js" file open. The file contains the following JavaScript code:

```
1 Blockly.JavaScript['blink_start'] = function(block) {
2     var code = 'DEV_IO.BLINK().start();\n';
3     return code;
4 };
5
6 Blockly.JavaScript['blink_stop'] = function(block) {
7     var code = 'DEV_IO.BLINK().stop();\n';
8     return code;
9 };
10
```

The left sidebar shows a file tree with the following structure:

- OPEN EDITORS
- UNTITLED (WORKSPACE)
  - led
  - blink
    - msg
    - JS en.js
    - JS th.js
    - blocks.js
  - JS generators.js
  - { led.json }

The status bar at the bottom shows: master\* 21:01 0 0 ▲ 0 ⌂ In 10, Col 1 Spaces: 4 UTF-8 CRLF JavaScript

# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (ต่อ 3.)

- ทำการสร้างไฟล์ blink.h ขึ้นมาใน folder “blink” และใส่โค้ดในไฟล์ดังนี้



A screenshot of Visual Studio Code showing the file "blink.h" in the workspace. The code defines a class BLINK that inherits from Device. It includes private members for state machine logic (enum s\_detect, s\_blink; state), blink\_status, and blink\_flag. The public interface includes tickcnt (for timekeeping), constructor, and methods for initialization, property handling (prop\_count, prop\_name, prop\_unit, prop\_attr, prop\_read, prop\_write), and control (start, stop).

```
#ifndef __BLINK_H__
#define __BLINK_H__
#include "driver.h"
#include "device.h"
#include "driver/gpio.h"
class BLINK : public Device {
private:
    enum {
        s_detect, s_blink
    } state;
    int blink_status;
    bool blink_flag;
public:
    TickType_t tickcnt;
    // constructor
    BLINK(void);
    // override
    void init(void);
    void process(Driver *drv);
    int prop_count(void);
    bool prop_name(int index, char *name);
    bool prop_unit(int index, char *unit);
    bool prop_attr(int index, char *attr);
    bool prop_read(int index, char *value);
    bool prop_write(int index, char *value);
    // method
    void start(void);
    void stop(void);
};
#endif
```

- สร้างคลาส BLINK สืบทอดจากคลาส Device
- ตัวแปรแบบ private
  - state ใช้ในเม็ด-ร็อด process ซึ่งทำงานแบบ state machine
  - blink\_status เก็บสถานะ LED
  - blink\_flag กำหนดสถานะให้กระพริบ หรือหยุดกระพริบ
- ตัวแปรแบบ public
  - tickcnt ใช้เก็บค่าเวลาครุ่งก่อน เพื่อนำมาคำนวณเวลาที่ผ่านไป ว่าได้ระยะเวลาที่ต้องการหรือยัง ด้วยฟังก์ชัน is\_tickcnt\_expired()
- เม็ด-ร็อดแบบ public
  - init ใช้ใส่โค้ดสำหรับกำหนดค่าเริ่มต้นต่างๆ เม็ด-ร็อดนี้จะถูกเรียกโดย Device Manager ก่อนเข้าสู่การทำงาน state machine ของเม็ด-ร็อด process
  - process ใช้สำหรับใส่โค้ดซึ่งทำงานแบบ state machine และจะถูกเรียกให้ทำงานเป็นระยะๆ ด้วย Device Manager
  - เม็ด-ร็อดที่ขึ้นต้นด้วย prop\_ ใช้สำหรับใส่โค้ดเพื่อรับ Command Line Interface (CLI)
  - start ใช้เขียนโค้ดให้ LED เริ่มกระพริบ
  - stop ใช้เขียนโค้ดให้ LED หยุดกระพริบ

# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (ต่อ 4.)

- ทำการสร้างไฟล์ blink.cpp ขึ้นมาใน folder “blink” และใส่โค้ดในไฟล์ดังนี้



```
bug Terminal Help blink.cpp - Untitled (Workspace) - Visual Studio Code
led.json      JS generators.js  blink.cpp  blink.h  blocks.js

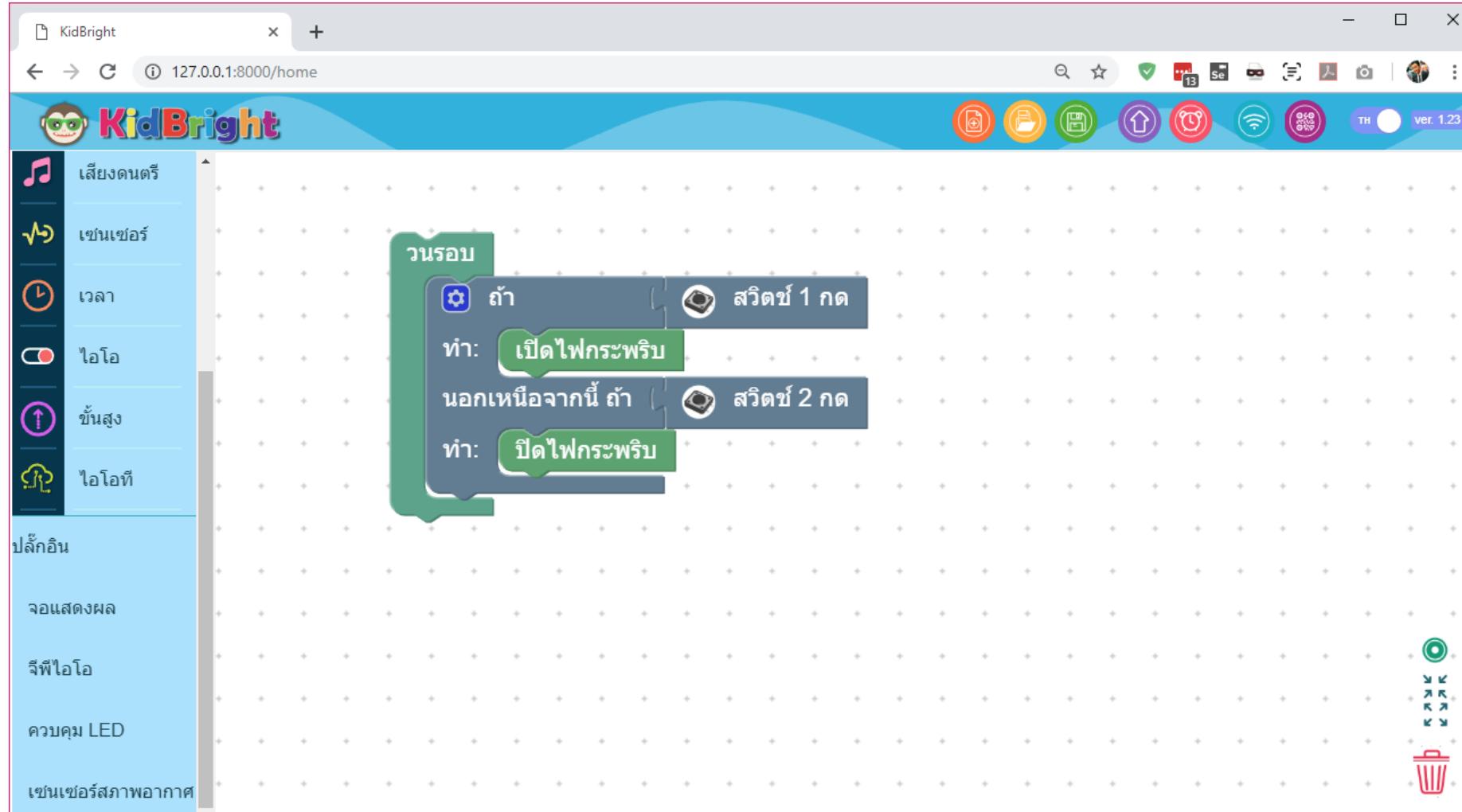
1 #include "esp_system.h"
2 #include "kidbright32.h"
3 #include "blink.h"
4
5 BLINK::BLINK(void) {
6     //
7 }
8
9 void BLINK::init(void) {
10    gpio_config_t io_conf;
11
12    // outputs init
13    io_conf.intr_type = GPIO_INTR_DISABLE;           // disable interrupt
14    io_conf.mode = GPIO_MODE_OUTPUT;                 // set as output mode
15    io_conf.pin_bit_mask = (1ULL << BT_LED_GPIO); // pin bit mask
16    io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;   // disable pull-down mode
17    io_conf.pull_up_en = GPIO_PULLUP_DISABLE;        // disable pull-up mode
18    blink_status = 1;
19    gpio_set_level(BT_LED_GPIO, blink_status);       // active low
20    gpio_config(&io_conf);
21
22    blink_flag = false;
23    state = s_detect;
24}
25
26 int BLINK::prop_count(void) {
27    // not supported
28    return 0;
29}
30
31 bool BLINK::prop_name(int index, char *name) {
32    // not supported
33    return false;
}

BLINK::stop(void)  Ln 91, Col 22  Spaces: 4  UTF-8  CRLF  C++  V
```

- เม็ด-ร็อด process เป็น state machine มีการทำงานดังนี้
  - s\_detect เป็น state สำหรับตรวจสอบ Device ในกรณีไม่ใช้งานให้กำหนด error เป็น false เพื่อแสดงสถานะของ Device เมื่อ error และกำหนดให้ initialized เป็น true เพื่อแสดงสถานะของ Device ว่าพร้อมใช้งาน หลังจากนั้นกำหนด state ไปที่ r\_blink
  - r\_blink ใน state นี้จะตรวจสอบ blink\_flag ว่าต้องการให้ LED กระพริบหรือไม่ ถ้าต้องการให้กระพริบ จะคำนวณค่าเวลาที่ผ่านไปด้วยพิ้งก์ชัน is\_tickcnt\_elapsed โดยเปรียบเทียบกับค่า tickcnt ล่าสุด กับค่า 500 ms ถ้าเวลาผ่านไปได้ตามกำหนดนี้แล้ว ให้ทำการดูกระพริบ LED แล้วบันทึกค่า tickcnt ปัจจุบันด้วยพิ้งก์ชัน get\_tickcnt() เพื่อใช้ในการคำนวณในรอบต่อไป
  - เม็ด-ร็อด start จะสั่งให้ LED ติด บันทึกค่า tickcnt ปัจจุบัน แล้วตั้งค่า blink\_flag เป็น true เพื่อให้เม็ด-ร็อด process ทำการกระพริบ LED
  - เม็ด-ร็อด stop จะตั้งค่า blink\_flag เป็น false เพื่อหยุดการกระพริบ LED ในเม็ด-ร็อด process แล้วสั่งดับ LED

# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (ต่อ 5.)

- ทำการทดสอบปลั๊กอินที่สร้างขึ้นใหม่ หากทำถูกต้องไฟ LED BT จะกระพริบทุก 500ms



# Lab 2 : ทำให้ Block รองรับภาษาไทยและอังกฤษ

- ให้ทำการแก้ไขไฟล์ blocks.js โดยย้ายข้อความภาษาไทยที่ฟังไว้ ใส่ใน th.js และ แปลเป็นภาษาอังกฤษ ในไฟล์ en.js โดยตั้งชื่อตัวแปรให้สับพันธุ์กัน

```
blocks.js - Untitled (Workspace) - Visual Studio Code
Debug Terminal Help
led.json generators.js blink.cpp blink.h blocks.js
1 Blockly.Blocks['blink_start'] = {
2     init() {
3         this.appendDummyInput()
4             .appendField(Blockly.Msg.BLINK_START_TITLE);
5         this.setPreviousStatement(true, null);
6         this.setNextStatement(true, null);
7         this.setColour(155);
8         this.setTooltip(Blockly.Msg.BLINK_START_TOOLTIP);
9         this.setHelpUrl("");
10    }
11 };
12
13 Blockly.Blocks['blink_stop'] = {
14     init() {
15         this.appendDummyInput()
16             .appendField(Blockly.Msg.BLINK_STOP_TITLE);
17         this.setPreviousStatement(true, null);
18         this.setNextStatement(true, null);
19         this.setColour(125);
20         this.setTooltip(Blockly.Msg.BLINK_STOP_TOOLTIP());
21         this.setHelpUrl("");
22    }
23};
```

Ln 20, Col 55 Tab Size: 4 UTF-8 CRLF JavaScript 😊

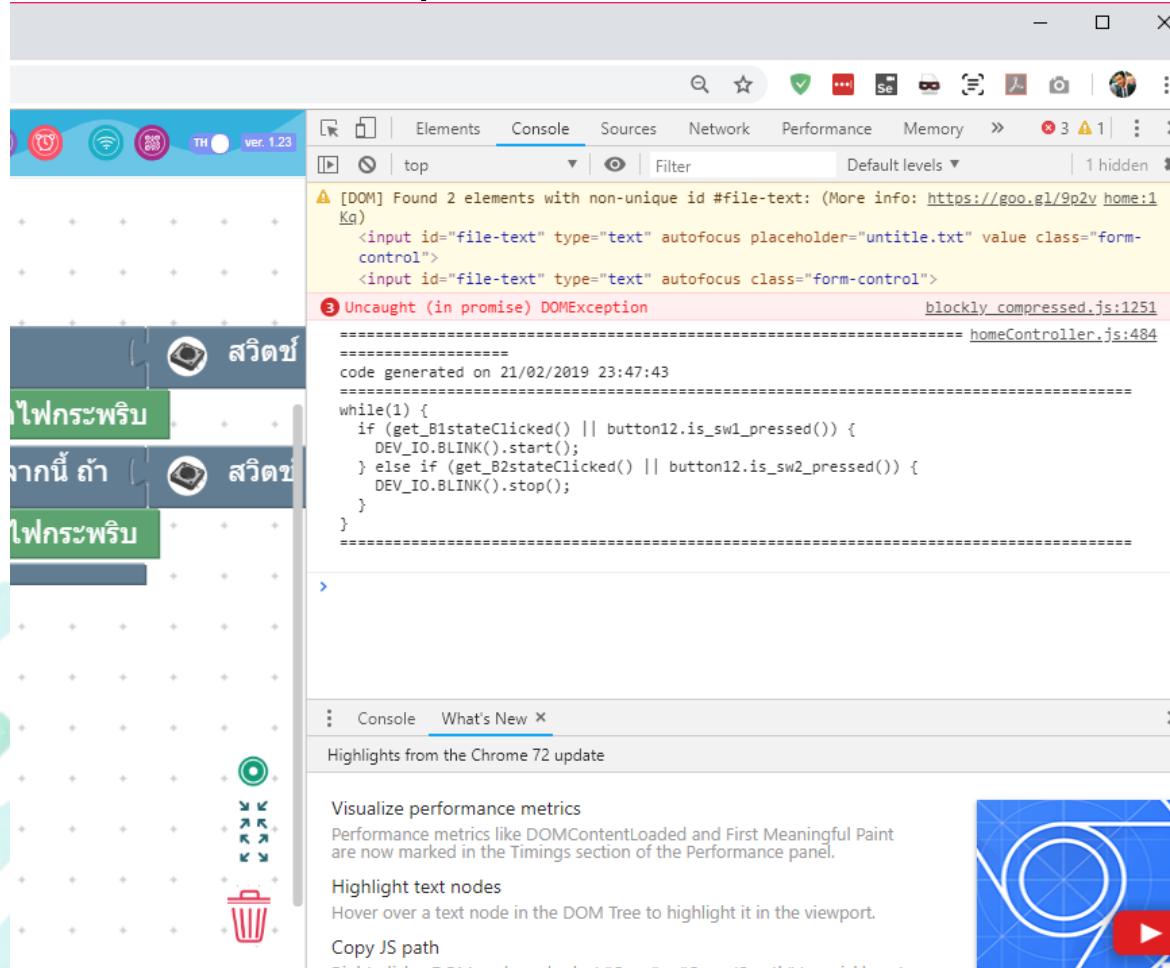
```
E:\KidBright\kbide\plugins\led\blink\msg\en.js - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
en.js
1 Blockly.Msg.BLINK_START_TITLE = "Start Blinking";
2 Blockly.Msg.BLINK_START_TOOLTIP = "Start Blinking LED";
3
4 Blockly.Msg.BLINK_STOP_TITLE = "Stop Blinking";
5 Blockly.Msg.BLINK_STOP_TOOLTIP = "Stop Blinking LED";
```

```
E:\KidBright\kbide\plugins\led\blink\msg\th.js - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
th.js
1 Blockly.Msg.BLINK_START_TITLE = "เปิดไฟกระพริบ";
2 Blockly.Msg.BLINK_START_TOOLTIP = "เริ่มแสดงไฟกระพริบ";
3
4 Blockly.Msg.BLINK_STOP_TITLE = "ปิดไฟกระพริบ";
5 Blockly.Msg.BLINK_STOP_TOOLTIP = "หยุดแสดงไฟกระพริบ";
6 |
```

# Lab 2 : ວຽກ Compile ຂອງໂຄດ

- ໃຫ້ກຳທາງກດ F12 ຢັ້ງ Ctrl + Shift + J ຈະເຮັດໂຄດທີ່ Generate ອອກມາຈາກບລືວຄ ແລະດຸດແປລູນເປັນໂຄດທີ່ພຽວມ compile



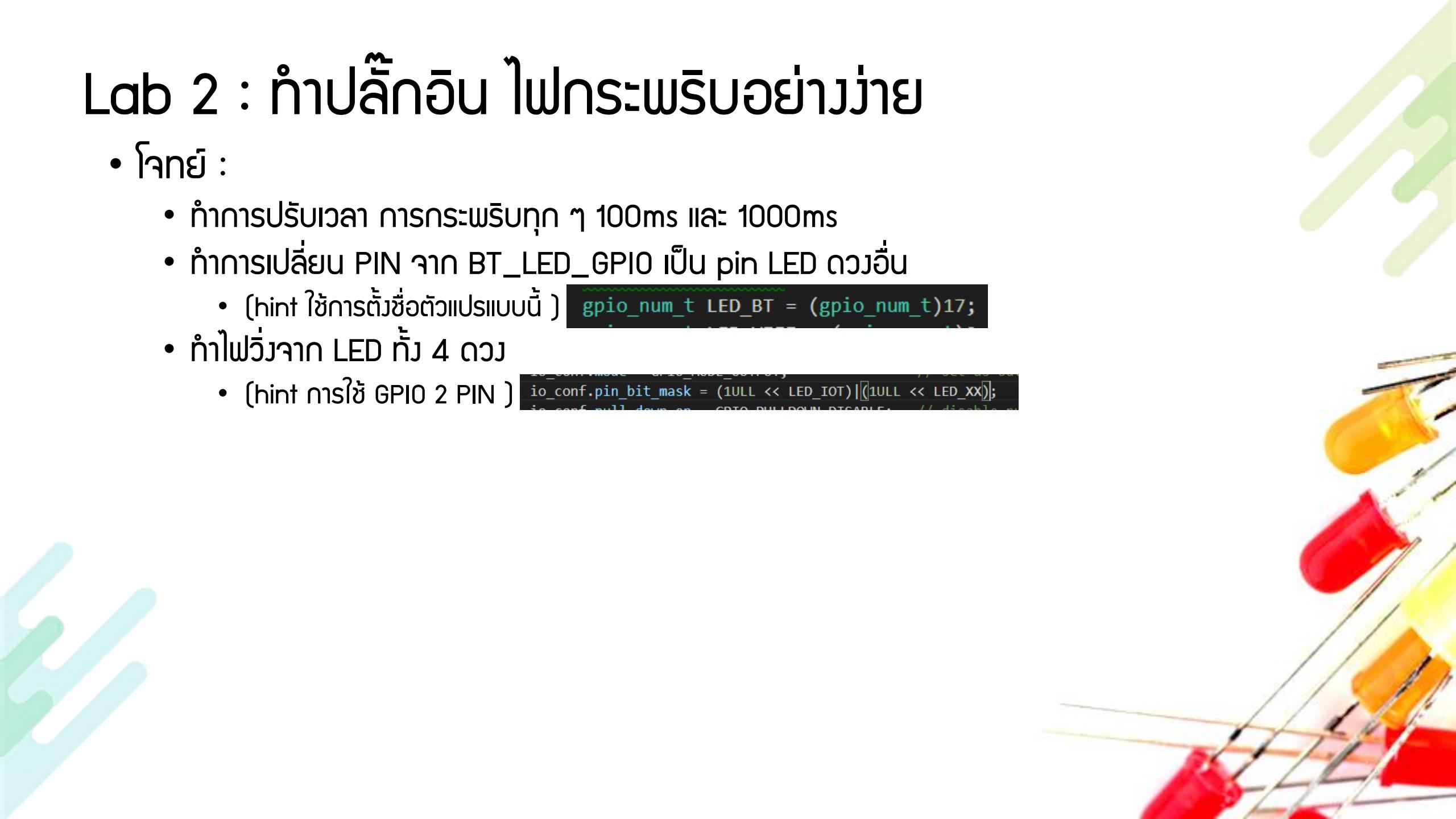
The screenshot shows a Sublime Text editor window with an open file named 'user\_app.cpp'. The code is written in C++ and includes functions like 'iotTask', 'user\_app', and 'vUserAppTask'. The code handles button presses and creates tasks. The status bar at the bottom indicates 'Line 1, Column 1'.

```
43 // =====
44 // plug-ins devices
45 // =====
46 BLINK blink_0 = BLINK();
47
48 void iotTask(void *pvParameters) {
49 vTaskDelay(500 / portTICK_RATE_MS);
50     vTaskDelete(NULL);
51 }
52
53 void user_app(void) {
54 xTaskCreatePinnedToCore(iotTask, "iotTask", 4096, NULL, USERAPP_TASK_PRIORITY);
55 srand(mcp7940n.get(5));
56 // setup
57 while(1) {
58     if (get_B1stateClicked() || button12.is_sw1_pressed()) {
59         blink_0.start();
60     } else if (get_B2stateClicked() || button12.is_sw2_pressed()) {
61         blink_0.stop();
62     }
63
64     // create tasks
65 }
66
68 void vUserAppTask(void *pvParameters) {
```

# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย

- โจทย์ :

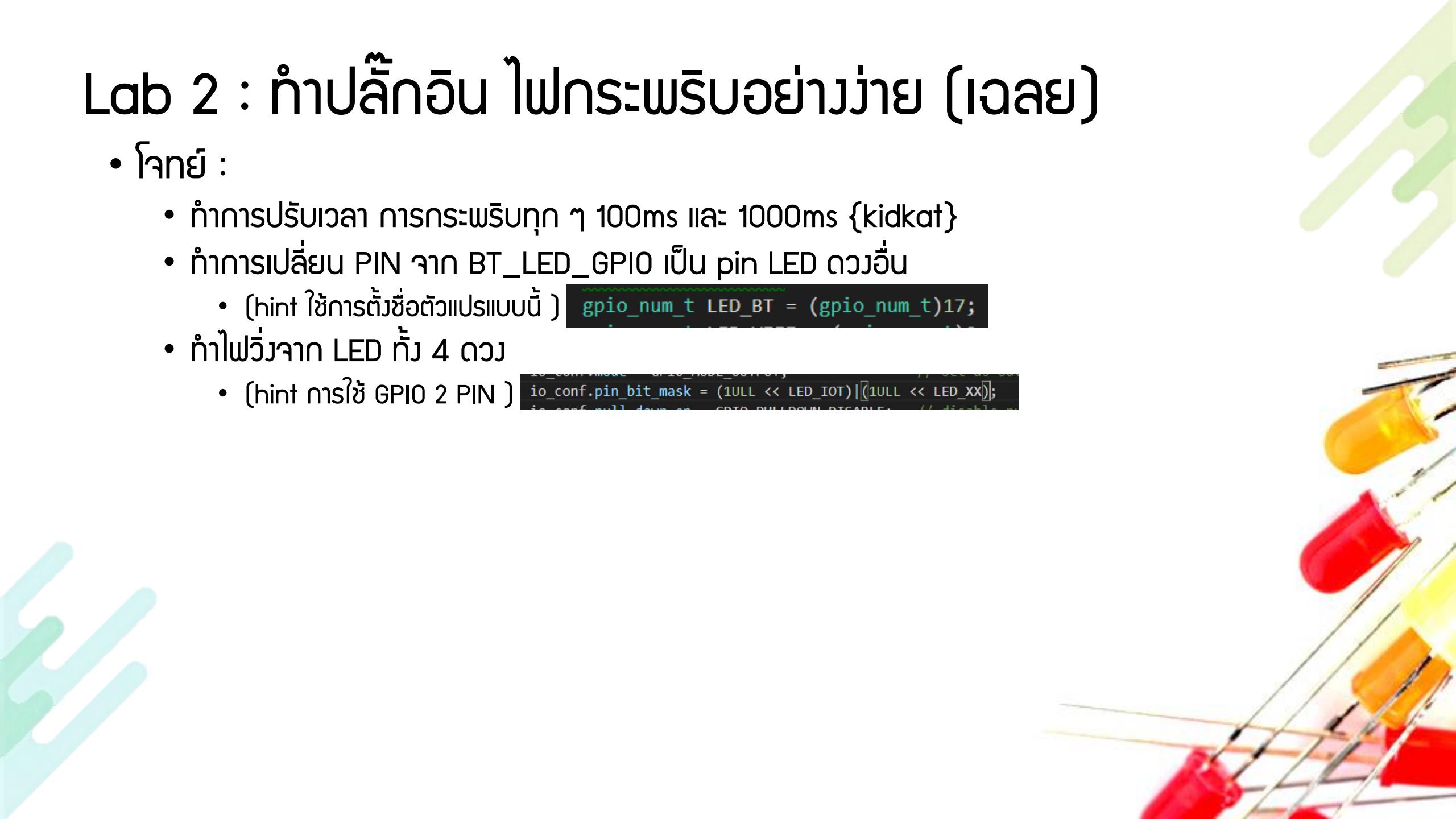
- ทำการปรับเวลา การกระพริบทุก ๆ 100ms และ 1000ms
- ทำการเปลี่ยน PIN จาก BT\_LED\_GPIO เป็น pin LED ด้วงอึบ
  - (hint ใช้การตั้งชื่อตัวแปรแบบนี้ ) `gpio_num_t LED_BT = (gpio_num_t)17;`
- ทำไฟวิ่งจาก LED กั้ง 4 ดวง
  - (hint การใช้ GPIO 2 PIN ) `io_conf.pin_bit_mask = (1ULL << LED_IOT) | (1ULL << LED_xx);`



# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (เฉลย)

- โจทย์ :

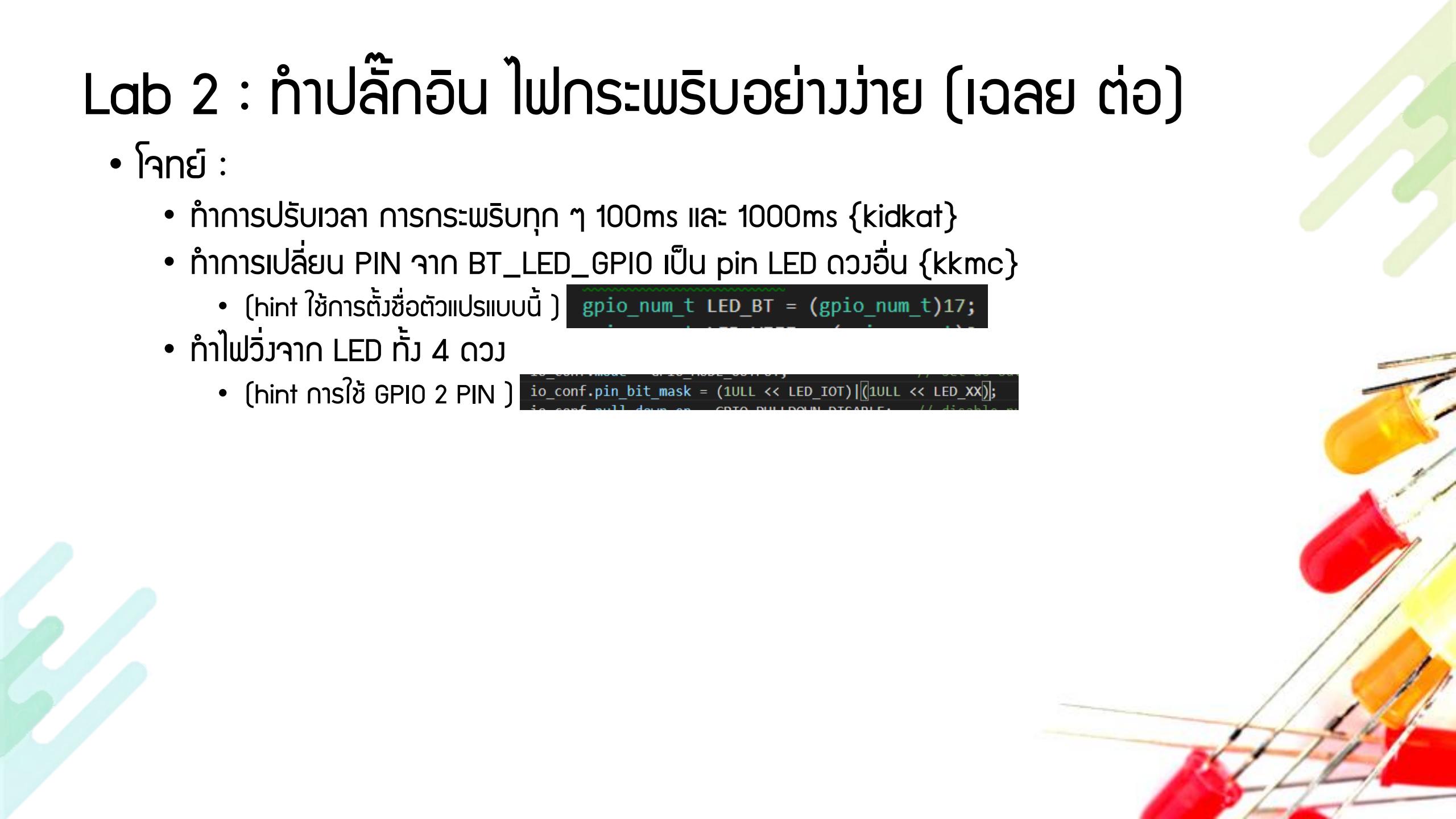
- ทำการปรับเวลา การกระพริบทุก ๆ 100ms และ 1000ms {kidkat}
- ทำการเปลี่ยน PIN จาก BT\_LED\_GPIO เป็น pin LED ด้วงอึบ
  - (hint ใช้การตั้งชื่อตัวแปรแบบนี้ ) `gpio_num_t LED_BT = (gpio_num_t)17;`
- ทำไฟวิ่งจาก LED ก้าว 4 ด้วง
  - (hint การใช้ GPIO 2 PIN ) `io_conf.pin_bit_mask = (1ULL << LED_IOT) | (1ULL << LED_xx);`



# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (เฉลย ต่อ)

- โจทย์ :

- ทำการปรับเวลา การกระพริบทุก ๆ 100ms และ 1000ms {kidkat}
- ทำการเปลี่ยน PIN จาก BT\_LED\_GPIO เป็น pin LED ด้วงอื่น {kkmc}
  - (hint ใช้การตั้งชื่อตัวแปรแบบนี้ ) `gpio_num_t LED_BT = (gpio_num_t)17;`
- ทำไฟวิ่งจาก LED ก้าว 4 ด้วง
  - (hint การใช้ GPIO 2 PIN ) `io_conf.pin_bit_mask = (1ULL << LED_IOT) | (1ULL << LED_xx);`



# Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (เฉลย ต่อ)

- โจทย์ :

- ทำการปรับเวลา การกระพริบทุก ๆ 100ms และ 1000ms {kidkat}
- ทำการเปลี่ยน PIN จาก BT\_LED\_GPIO เป็น pin LED ด้วงอื่น {kkmc}
  - (hint ใช้การตั้งชื่อตัวแปรแบบนี้ ) `gpio_num_t LED_BT = (gpio_num_t)17;`
- ทำไฟวิ่งจาก LED ก้าว 4 ด้วง {kku}
  - (hint การใช้ GPIO 2 PIN ) `io_conf.pin_bit_mask = (1ULL << LED_IOT) | (1ULL << LED_xx);`
- Full plugin : {kkkk}

Lab 2 : กำปั๊กอัน ไฟกระพริบอย่างง่าย

ไฟกระพริบอย่างง่าย !?

ง่ายกว่านี้มีอีกใหม่ ... ง่ายแบบ `ctrl+c ctrl+v`

ພັກເຖິ່ຍງ  
12.00 - 13.00 ພ.



# Lab 3 : Copy & Paste Block

- เป้าหมาย : สร้าง Block เปิดการใช้งานและควบคุม GPIO โดยออกแบบใหม่
- วิธีการทำ : ทำการคัดลอก “kbgpio” จาก “lab\_direction” ใส่ใน folder “led” ใน plugins
- จุดยัง :

  - ข้อ 1 ทำการอุปกรณ์ลือค ของ function setOutput(int pin) และ digitalWrite(int pin,int value) ใส่ในไฟล์ blocks.js และเรียกใช้ function กันสองฝ่ายในไฟล์ generators.js
  - ข้อ 2 เขียนโปรแกรมไฟวิ่งบน Block ทดสอบเรียกใช้ block ที่สร้างขึ้น
  - ข้อ 3 ทดสอบควบคุม Relay ด้วย PIN อื่นๆ

# LAB 3 : ວອກແບບ GPIO ໃຫມ່

```
#ifndef __KBGPIO_H__
#define __KBGPIO_H__
#include "driver.h"
#include "device.h"
#include "driver/gpio.h"
class KBGPIO : public Device {
    private:
    public:
        // constructor
        KBGPIO(void);
        // override
        void inline init(void){ return; };
        void inline process(Driver *drv){ error = false; initialized = true; }
        int inline prop_count(void){ return 0; }
        bool inline prop_name(int index, char *name){ return false; }
        bool inline prop_unit(int index, char *unit){ return false; }
        bool inline prop_attr(int index, char *attr){ return false; }
        bool inline prop_read(int index, char *value){ return false; }
        bool inline prop_write(int index, char *value){ return false; }
        // method
        void setOutput(int pin);
        void digitalWrite(int pin,int value);
};
#endif
```

- ໂຍນກາຮະກາຮັດກຳນົດກຳນົດຂອງໂປຣແກຣມໄປໃຫ້  
ພູ້ໃຊ້ ເຮົາວອກແບບແລ້ວ function  
ພື້ນສູານ
- ຕັດກາຣ override state ວະໄຣຕ່າງໆ  
ວອກ ກຳໃຫ້ໂຄດດຸເຂົ້າໃຈງ່າຍ
- ໃຊ້ concept ທີ່ເຄຍຮູ້ຈັກ (arduino)  
ເປີດ pinMode ກ່ອນໃຊ້ງານ ແລະ ໃຊ້ຄໍາສົ່ງ  
DigitalWrite ໃນກາຣគົບຄຸມ

# LAB 3 : ວອກແບບ GPIO ໃຫມ່

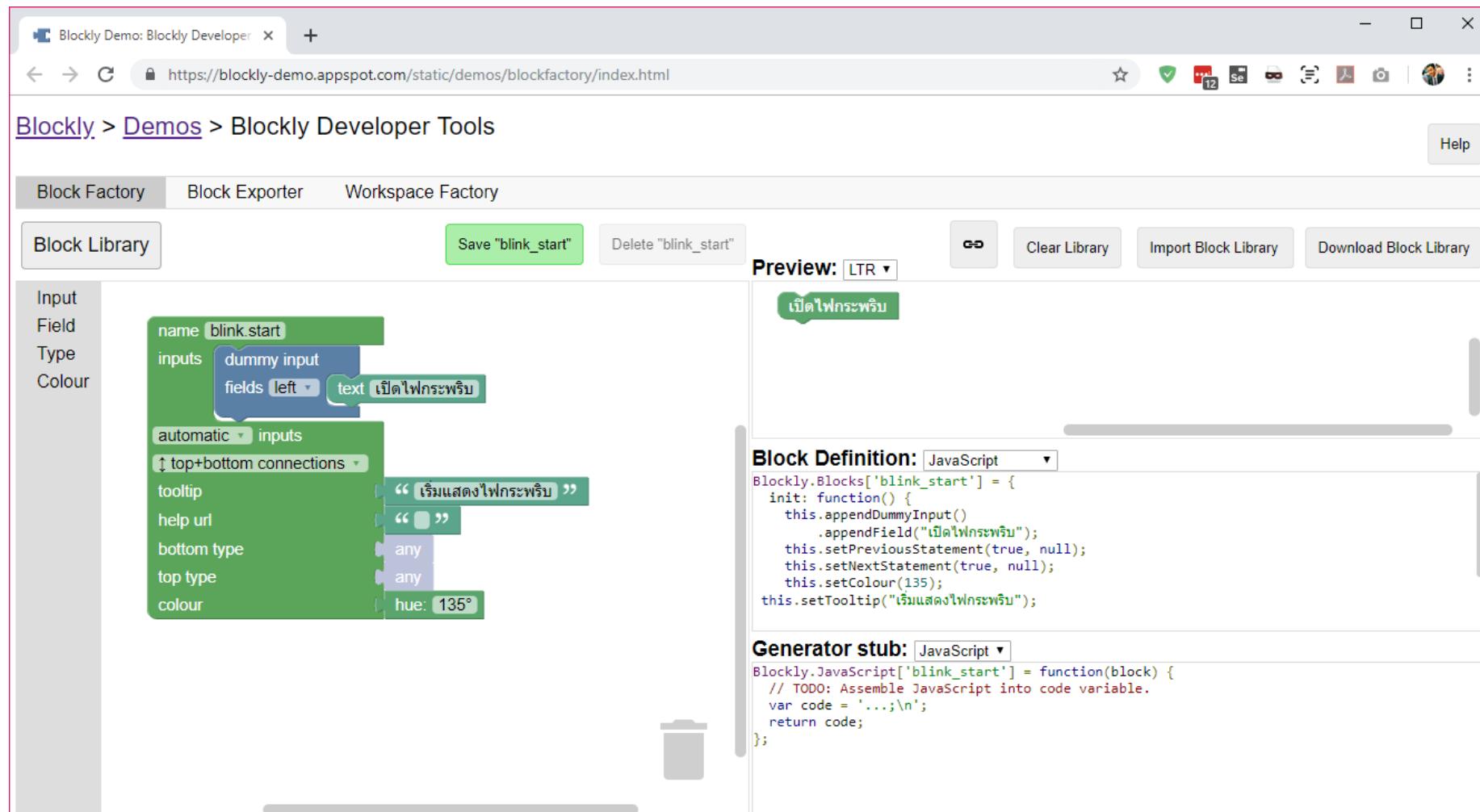
```
#include "esp_system.h"
#include "kidbright32.h"
#include "kbgpio.h"
KBGPIO::KBGPIO(void) {
    //
}
void KBGPIO::setOutput(int pin) {
    gpio_config_t io_conf;
    // outputs init
    io_conf.intr_type = GPIO_INTR_DISABLE;          // disable interrupt
    io_conf.mode = GPIO_MODE_OUTPUT;                // set as output mode
    io_conf.pin_bit_mask = (1ULL << pin);         // pin bit mask
    io_conf.pull_down_en = GPIO_PULLDOWN_DISABLE;   // disable pull-down mode
    io_conf.pull_up_en = GPIO_PULLUP_DISABLE;        // disable pull-up mode
    gpio_config(&io_conf);
    gpio_set_level((gpio_num_t)pin, 1);      // active low
}

void KBGPIO::digitalWrite(int pin,int value)
{
    gpio_set_level((gpio_num_t)pin, value);
}
```

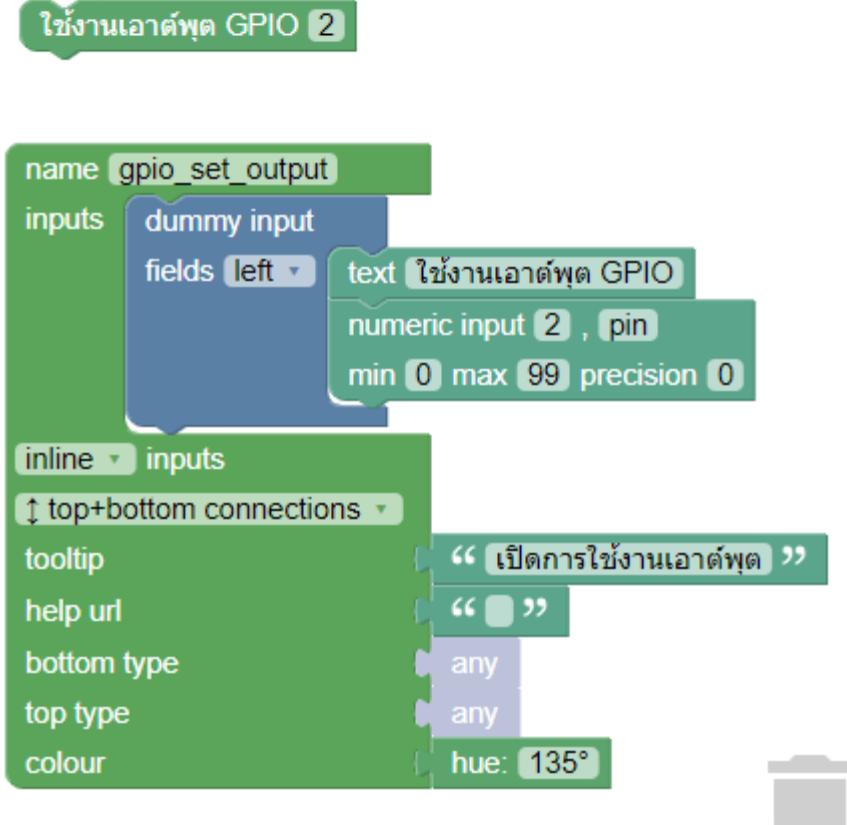
- `setOutput` เป็นการกำหนดค่า `pin` เขียนแบบ ESP-IDF โดยคัดลอกมา จากปลั๊กอินก่อนหน้านี้
- ไม่มี `state machine` แล้ว
- ไม่มี `override` แล้ว (เขียนไปในไฟล์ `header` หมดแล้ว)
- คำสั่ง `digitalWrite` ตรงตัว

# LAB 3 : ตัวช่วยในการสร้างและอ่านแบบบล็อก

- <https://blockly-demo.appspot.com/static/demos/blockfactory/index.html>



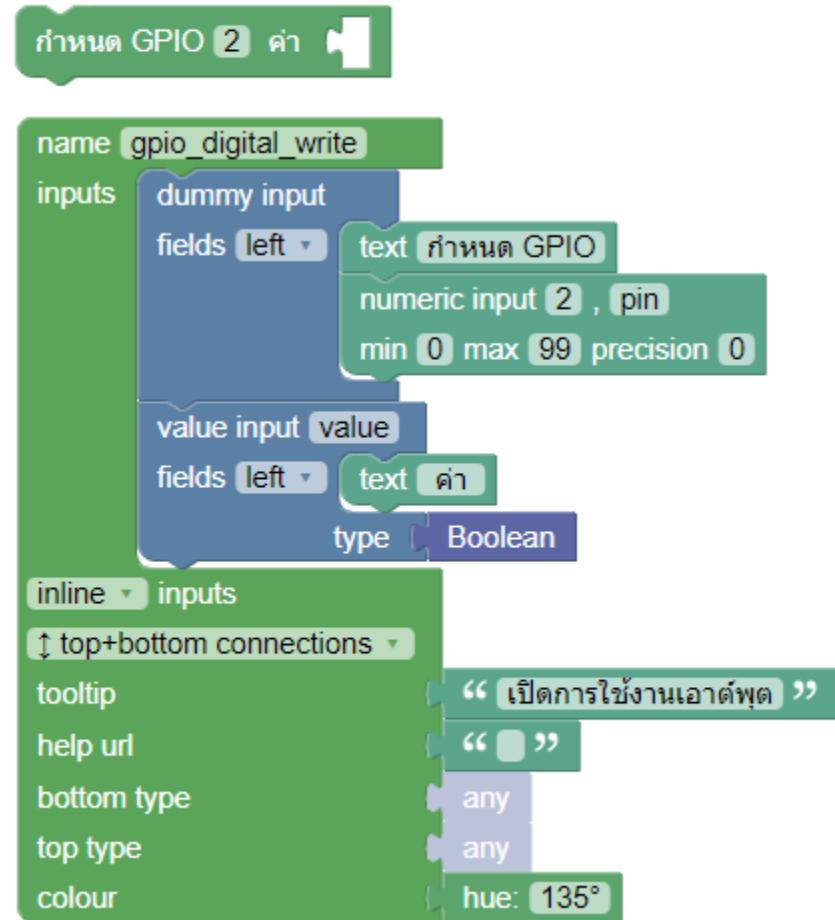
# LAB 3 : វិវេកាសមុខ 1



```
Blockly.Blocks['gpio_set_output'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("ใช้งานเอาតិពុទ GPIO")
      .appendField(new Blockly.FieldNumber(2, 0, 99), "pin");
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(135);
    this.setTooltip("เปิดการใช้งานเอาតិពុទ");
    this.setHelpUrl("");
  }
};
```

```
Blockly.JavaScript['gpio_set_output'] = function(block) {
  var number_pin = block.getFieldValue('pin');
  var code = 'DEV_IO.BLINK().setOutput('+number_pin+');\n';
  return code;
};
```

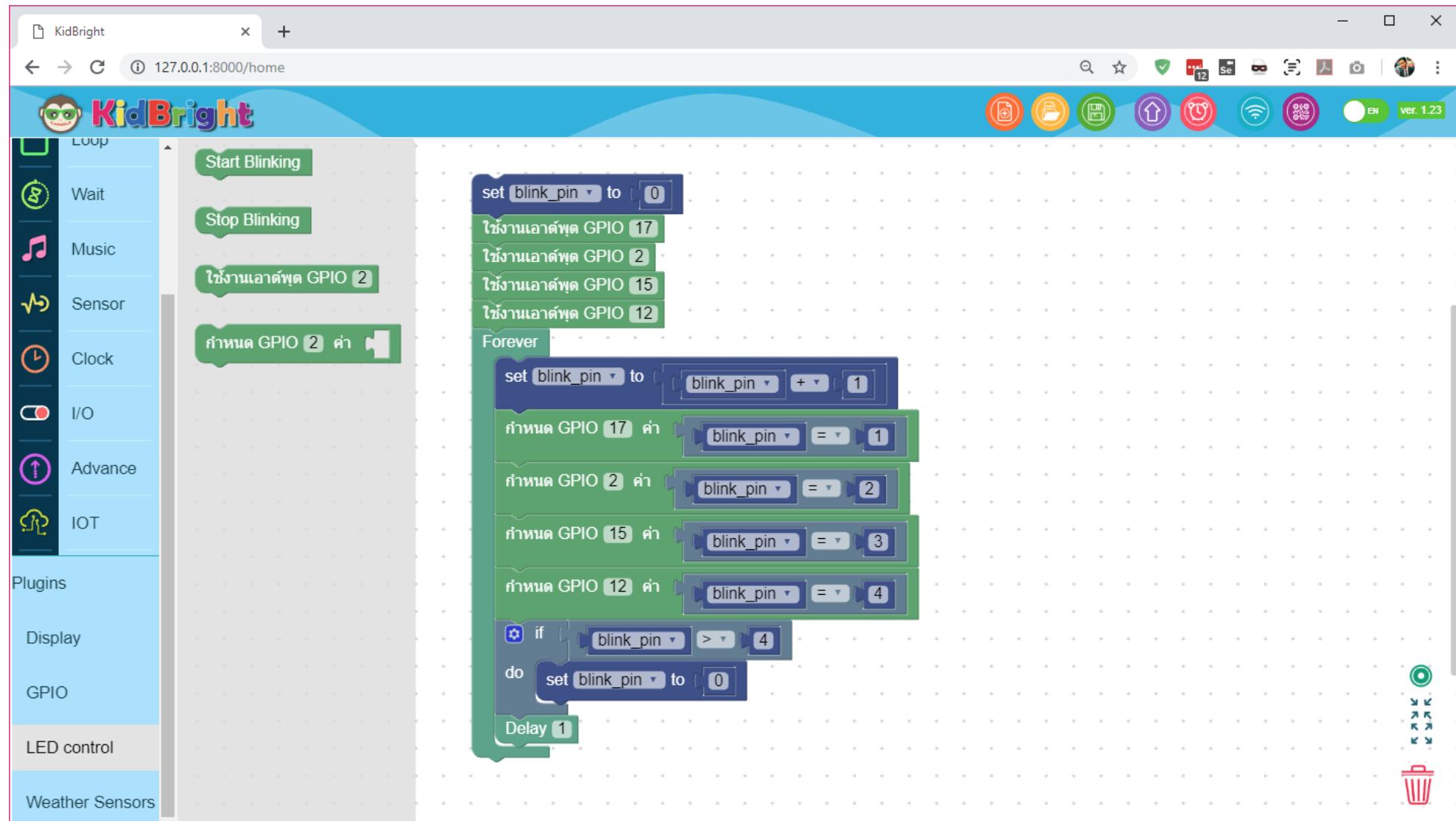
# LAB 3 : ວິທີກຳຂົວກີ່າ (ຕ່ອ) {nopass}



```
Blockly.Blocks['gpio_digital_write'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("ກໍານົດ GPIO")
      .appendField(new Blockly.FieldNumber(2, 0, 99), "pin");
    this.appendValueInput("value")
      .setCheck("Boolean")
      .appendField(" ດ້ວຍ ");
    this.setInputsInline(true);
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(135);
    this.setTooltip("ເປີດການໃຊ້ງານເຄົາຕໍ່ພຸດ");
    this.setHelpUrl("");
  }
};
```

```
Blockly.JavaScript['gpio_digital_write'] = function(block) {
  var number_pin = block.getFieldValue('pin');
  var value_value = Blockly.JavaScript.valueToCode(block, 'value', Blockly.JavaScript.ORDER_ATOMIC);
  var code = 'DEV_IO.BLINK().digitalWrite('+number_pin+','+value_value+');\n';
  return code;
};
```

# LAB 3 : ວິທີກຳຂົວກໍ່ 2 (ຕົວອຍ່າງ){nopass}



# ESP-IDF : การใช้งาน GPIO และ Time

Arduino	ESP-IDF
pinMode(5,[INPUT OUTPUT])	gpio_set_direction((gpio_num_t)5, [GPIO_MODE_OUTPUT GPIO_MODE_INPUT]);
pinMode(5,INPUT_PULLUP);	gpio_set_direction((gpio_num_t)5,GPIO_MODE_INPUT); gpio_set_pull_mode((gpio_num_t)5, GPIO_PULLUP_ONLY);
delay(100);	vTaskDelay(100/portTICK_PERIOD_MS);
delayMicroseconds(10);	ets_delay_us(10); // #include <freertos/task.h>
digitalWrite(5,HIGH);	gpio_set_level((gpio_num_t)5, 0);
digitalRead(5);	gpio_get_level((gpio_num_t)5);
micros();	esp_timer_get_time(); // #include< esp32_timer.h>
millis();	esp_timer_get_time()/1000ULL;
analogWrite(A1,125); //0-255 PWM	ค่อยว่ากัน
analogRead(A1);	ค่อยว่ากัน

A silhouette of a person standing on top of a dark mountain peak against a bright sunset sky. The sky is filled with large, white, fluffy clouds and a warm gradient of orange, yellow, and blue. The sun is visible on the right side, partially obscured by the clouds.

END of DAY 1

# Lab 4 : วงศ์รักษ์น้อย

- เป้าหมาย : ใช้งาน Digital Input และ Timer ผ่าน ESP-IDF
- โนดูล : Ultrasonic Distance measuring sensor HR-SR04
- วิธีการทำ :
  - ทำการเชื่อมต่อ Ultrasonic SR04 เข้ากับ KidBright โดยต่อไฟเลี้ยง 5v และ GND บนบอร์ด, Echo เข้ากับ IN1, Trig เข้ากับ OUT1
  - ทำการสร้างหมวดหมู่ปลั๊กอินใหม่ folder Ultrasonic โดยแสดงคำว่า “Distance Sensor” “โนดูลวัดระยะทาง” ใน IDE
  - ทำการคัดลอก ปลั๊กอิน kbgpio ใน Lab direction มาสร้างปลั๊กอินใหม่ โดยแก้ไข kbgpio กั้งหมวด ในทุกไฟล์เป็น “sr04”
  - ทำการออกแบบปลั๊กอินกั้งหมวด 1 block ให้มีลักษณะดังนี้

อ่านค่า sr04 ขา trig 26 ขา Echo 32



# Lab 4 : ວິຈາຮັບໂນຍ (ຕ່ອ)

- ກໍາດາກແກ້ໄຂໄຟລ໌ sr04.h ໂດຍສ້າງຕົວແປສ private ກໍາເນດ constructor ແລະໃສ່ include ໄຟລ໌

```
1 #ifndef __SR04_H__
2 #define __SR04_H__
3 #include "driver.h"
4 #include "device.h"
5 #include "driver/gpio.h"
6 #include <freertos/FreeRTOS.h>
7 #include "esp_timer.h"
8 #include "freertos/task.h"
9 class SR04 : public Device {
10     private:
11         gpio_num_t trig;
12         gpio_num_t echo;
13     public:
14         // constructor
15         SR04(int trig_Pin,int echo_pin);
16         // override
17         void inline init(void){ return; };
18         void inline process(Driver *drv){ error = false; initialized = true; }
19         int inline prop_count(void){ return 0; }
20         bool inline prop_name(int index, char *name){ return false; }
21         bool inline prop_unit(int index, char *unit){ return false; }
22         bool inline prop_attr(int index, char *attr){ return false; }
23         bool inline prop_read(int index, char *value){ return false; }
24         bool inline prop_write(int index, char *value){ return false; }
25         // method
26         float read();
27     };
28 #endif
```

# Lab 4 : ວິຈາຮັບໂມຍ (ຕ່ອ)

- ກໍາການແກ້ໄຂໄຟລ໌ sr04.cpp ໂດຍໃຫ້ໂຄດຈາກ Lab direction

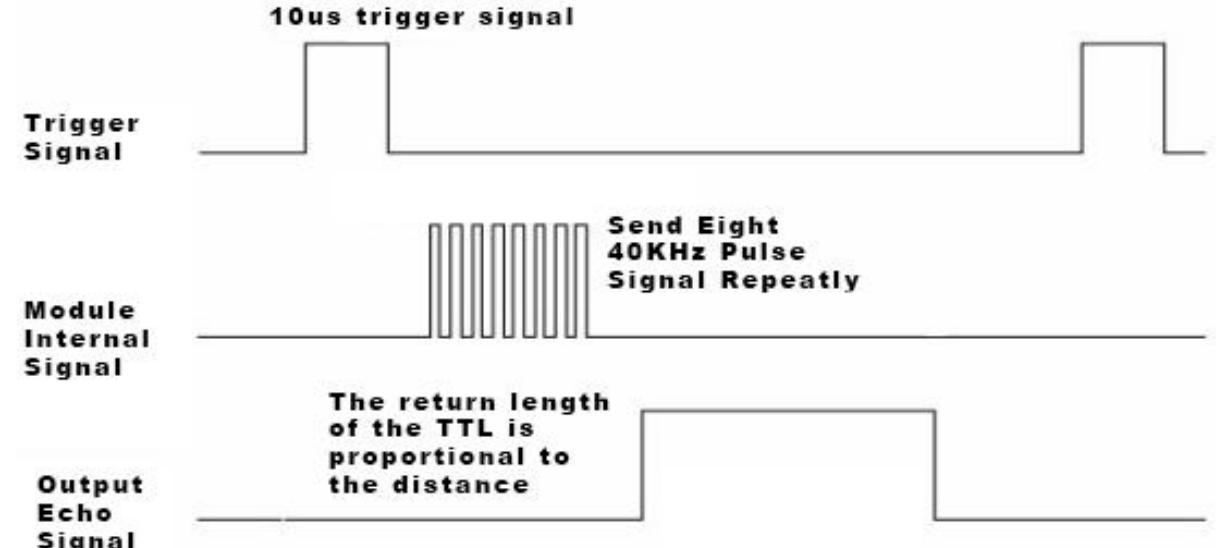


```
Debug Terminal Help sr04.cpp - Untitled (Workspace) - Visual Studio Code
JS generators.js sr04.cpp JS blocks.js C sr04.h
1 #include "esp_system.h"
2 #include "kidbright32.h"
3 #include "SR04.h"
4 SR04::SR04(int trig_pin, int echo_pin) {
5     this->trig = (gpio_num_t)trig_pin;
6     this->echo = (gpio_num_t)echo_pin;
7     gpio_set_direction(this->trig, GPIO_MODE_OUTPUT);
8     gpio_set_direction(this->echo, GPIO_MODE_INPUT);
9     gpio_set_level(this->trig, 0);
10 }
11 float SR04::read()
12 {
13     gpio_set_level(this->trig, 0);
14     ets_delay_us(4);
15     gpio_set_level(this->trig, 1);
16     ets_delay_us(10);
17     gpio_set_level(this->trig, 0);

18     if (gpio_get_level(this->echo)){ return -1; } // Previous ping isn't done yet

19     uint32_t start = esp_timer_get_time();
20     while (!gpio_get_level(this->echo)) // Wait for echo
21     {
22         if(esp_timer_get_time() - start > 100000ULL){ return -1; }
23     }
24     uint32_t echo_start = esp_timer_get_time();
25     uint32_t time = echo_start;
26     while (gpio_get_level(this->echo)) // got echo, measuring
27     {
28         time = esp_timer_get_time();
29         if (time - echo_start > 1000000ULL){ return -1; }
30     }
31     return (time - echo_start) / 58.0; //58 = ROUND TRIP
32 }
33 }
34 }
```

SR04::read() Ln 31, Col 37 Spaces: 4 UTF-8 CRLF C++ Win32 1



- Timing ດາວໂຫຼດ ຕ່າງໆ

# Lab 4 : วงศ์รักษ์มอย (โจกย์)

- โจกย์ :

- ทำการเขียนโปรแกรมแบบบล็อกเพื่อกดสوبบล็อกที่ได้
- ทำการเขียนโค้ดเพื่อให้ Sensor ทำงานตอนกลางคืนหลัง 8.00PM - 6.00AM ถ้าพบการเคลื่อนไหวให้ส่งเสียงขึ้นมาด้วย Buzzer



# Lab 4 : วงศ์รักนขโมย (เฉลย 1)

- โจทย์ :

- ทำการเขียนโปรแกรมแบบบล็อกเพื่อกดสوبบล็อกที่ได้ {k}
- ทำการเขียนโค้ดเพื่อให้ Sensor ทำงานตอนกลางคืนหลัง 8.00PM - 6.00AM ด้วยการเคลื่อนไหวให้ส่งเสียงขึ้นมาด้วย Buzzer



# Lab 4 : วงศ์รักษ์มอย (เฉลย 2)

- โจทย์ :

- ทำการเขียนโปรแกรมแบบบล็อกเพื่อกดสوبบล็อกที่ได้ {k}

- ทำการเขียนโค้ดเพื่อให้ Sensor ทำงานตอนกลางคืนหลัง 8.00PM - 6.00AM ด้วยการเคลื่อนไหวให้ส่งเสียงขึ้นมาด้วย Buzzer{knb84}



# พักเลսค 10 บาท



# การใช้งาน AnalogWrite (PWM) ใน KidBright

- การใช้งาน PWM นั้นจะมี Driver เวลาบน ESP32 เรียกว่า “LEDC”
- สามารถกำหนดคุณลักษณะของ PWM ได้หลายอย่างมาก แล้วแต่ Config
- วิธีการใช้งานมีดังนี้
  - ทำการ `#include <driver/ledc.h>`
  - ตั้งค่า timer

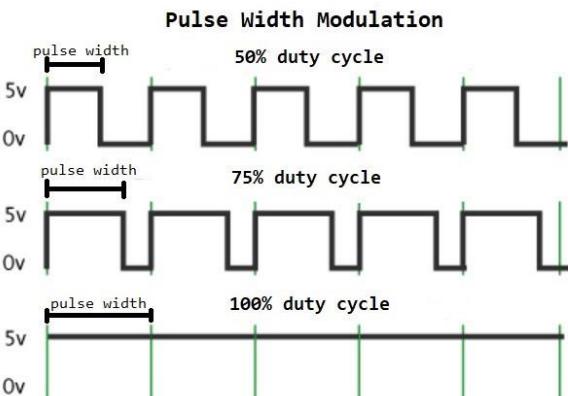
```
ledc_timer_config_t timer_conf;
timer_conf.duty_resolution = LEDC_TIMER_16_BIT;
timer_conf.freq_hz = 50;
timer_conf.speed_mode = LEDC_HIGH_SPEED_MODE;
timer_conf.timer_num = LEDC_TIMER_3;
ledc_timer_config(&timer_conf);
```

และตั้งค่า pin

```
ledc_channel_config_t ledc_conf;
ledc_conf.channel = _ledcChannel;
ledc_conf.duty = 0xFFFF;
ledc_conf.gpio_num = pin;
ledc_conf.intr_type = LEDC_INTR_DISABLE;
ledc_conf.speed_mode = LEDC_HIGH_SPEED_MODE;
ledc_conf.timer_sel = LEDC_TIMER_3;
ledc_channel_config(&ledc_conf);
```

- เรียกใช้งานผ่านการปรับ duty width

```
int us = angle * (_max - _min) / 180.0 + _min;
ledc_set_duty(LEDC_HIGH_SPEED_MODE, _ledcChannel, us);
ledc_update_duty(LEDC_HIGH_SPEED_MODE, _ledcChannel);
```

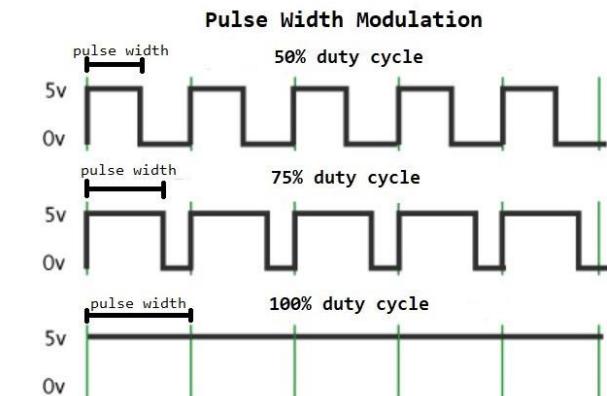


# การใช้งาน AnalogWrite (PWM) ใน KidBright

- การใช้งาน PWM นั้นจะมี Driver เวลาบน ESP32 เรียกว่า “LEDC”
- สามารถกำหนดคุณลักษณะของ PWM ได้หลายอย่างมาก แล้วแต่ Config
- วิธีการใช้งานมีดังนี้
  - ทำการ `#include <driver/ledc.h>`
  - ตั้งค่า timer และตั้งค่า pin

```
ledc_timer_config_t timer_conf;
timer_conf.duty_resolution = LEDC_TIMER_16_BIT;
timer_conf.freq_hz = 50;
timer_conf.speed_mode = LEDC_HIGH_SPEED_MODE;
timer_conf.timer_num = LEDC_TIMER_3;
ledc_timer_config(&timer_conf);
```

```
ledc_channel_config_t ledc_conf;
ledc_conf.channel = _ledcChannel;
ledc_conf.duty = 0xFFFF;
ledc_conf.gpio_num = pin;
ledc_conf.intr_type = LEDC_INTR_DISABLE;
ledc_conf.speed_mode = LEDC_HIGH_SPEED_MODE;
ledc_conf.timer_sel = LEDC_TIMER_3;
ledc_channel_config(&ledc_conf);
```

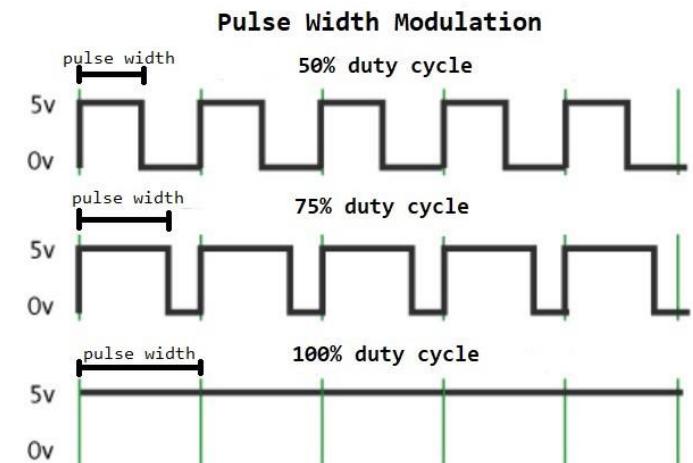
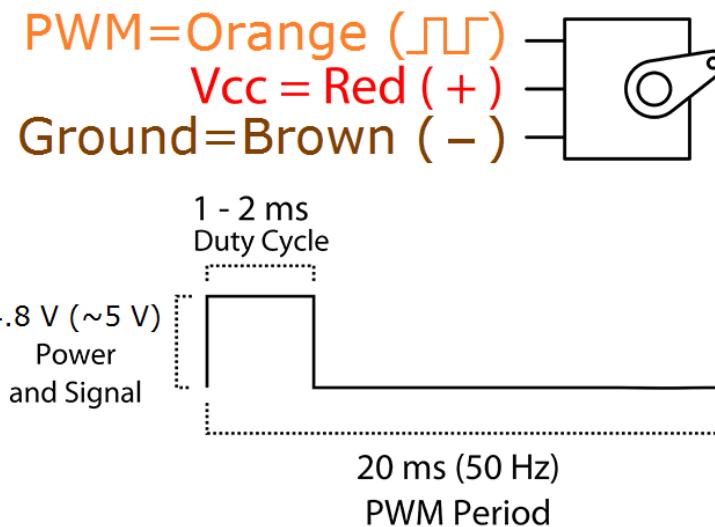


- เรียกใช้งานผ่านการปรับ duty width

```
int uS = angle * (_max - _min) / 180.0 + _min;
ledc_set_duty(LEDC_HIGH_SPEED_MODE, _ledcChannel, uS);
ledc_update_duty(LEDC_HIGH_SPEED_MODE, _ledcChannel);
```

# การใช้งาน PWM กับ Servo Motor

- Servo ส่วนใหญ่ใช้ PWM ในการควบคุม
- การควบคุมจะใช้ duty cycle แค่ประมาณ 10-20% เก่ามัน (ตามรูป)
- ความถี่ของ PWM อยู่ที่ 50Hz
- จากตัวอย่างเมื่อสักครู่ เราสามารถเขียนโค้ดควบคุม Servo ได้ง่าย ๆ



# ตัวอย่างโค้ดภาษา C++ ควบคุม Servo Motor

```
#ifndef __SERVO_H__
#define __SERVO_H__
#include "driver.h"
#include "device.h"
#include "driver/gpio.h"
#include "driver/ledc.h"
#include <freertos/FreeRTOS.h>
#include "esp_timer.h"
#include "freertos/task.h"
class Servo : public Device {
private:
    ledc_channel_t _ledcChannel;
    double _min;
    double _max;
public:
    // constructor
    Servo();
    // override
    void inline init(void){ return; }
    void inline process(Driver *drv){ error = false; initialized = true; }
    int inline prop_count(void){ return 0; }
    bool inline prop_name(int index, char *name){ return false; }
    bool inline prop_unit(int index, char *unit){ return false; }
    bool inline prop_attr(int index, char *attr){ return false; }
    bool inline prop_read(int index, char *value){ return false; }
    bool inline prop_write(int index, char *value){ return false; }
    // method
    void attach(int pin, double minus = 0.5, double maxus = 2.5, int ledcChannel = 0);
    void detach();
    void write(unsigned int value);
};
#endif
```

```
#include "Servo.h"

Servo::Servo(){

}

void Servo::attach(int pin, double minus, double maxus, int ledcChannel){
    _min = 0xFFFF * minus / 20.0;
    _max = 0xFFFF * maxus / 20.0;
    _ledcChannel = static_cast<ledc_channel_t>(ledcChannel);

    ledc_timer_config_t timer_conf;
    timer_conf.duty_resolution = LEDC_TIMER_16_BIT;
    timer_conf.freq_hz = 50;
    timer_conf.speed_mode = LEDC_HIGH_SPEED_MODE;
    timer_conf.timer_num = LEDC_TIMER_3;
    ledc_timer_config(&timer_conf);

    ledc_channel_config_t ledc_conf;
    ledc_conf.channel = _ledcChannel;
    ledc_conf.duty = 0xFFFF;
    ledc_conf gpio_num = pin;
    ledc_conf.intr_type = LEDC_INTR_DISABLE;
    ledc_conf.speed_mode = LEDC_HIGH_SPEED_MODE;
    ledc_conf.timer_sel = LEDC_TIMER_3;
    ledc_channel_config(&ledc_conf);
}

void Servo::detach(){
    ledc_stop(LEDC_HIGH_SPEED_MODE, _ledcChannel, 0);
}

void Servo::write(unsigned int angle) {
    // 0 = MinServoAngle ; 180 = Max ServoAngle;
    int us = angle * (_max - _min) / 180.0 + _min;
    ledc_set_duty(LEDC_HIGH_SPEED_MODE, _ledcChannel, us);
    ledc_update_duty(LEDC_HIGH_SPEED_MODE, _ledcChannel);
}
```

# Lab 5 : កំណើនគម្រោង

- เป้าหมาย : ใช้งาน PWM เพื่อควบคุม Servo Motor และใช้ร่วมกับ Ultrasonic Module
  - บอร์ด : Servo Motor, HR-SR04
  - วิธีการทำ :
    - ทำการเชื่อมต่อ Ultrasonic SR04 เข้ากับ KidBright โดยต่อไฟเลี้ยง 5v และ GND บนบอร์ด, Echo เข้ากับ IN1, Trig เข้ากับ OUT1
    - ทำการต่อ Servo มอเตอร์เข้ากับบอร์ดโดยต่อ GND เข้าสายไฟลสีน้ำตาล ไฟ 5v สายสีแดง และต่อ PIN 18 กับสายไฟสีส้ม
    - ทำการสร้างหมวดหมู่ปลั๊กอินใหม่ folder “motor” โดยแสดงคำว่า “Motor” “มอเตอร์” ใน IDE
  - จุดยึด :
    - ทำการสร้างปลั๊กจาก Class ที่กำหนดให้ใน Lab Direction
    - ทำการเขียนโปรแกรมด้วย Block เพื่อทดสอบการทำงานของ Servo Motor
    - ทำการเขียนโปรแกรมด้วย Block สั่งการ Servo Motor จาก Sonar Module

# Lab 5 : កំណើនរឿងរបៀប

- ផោមាយ : ិច្ចាបេរិយាល័យ PWM ដើម្បីគ្រប់គ្រង Servo Motor និង ិច្ចាបេរិយាល័យ Ultrasonic Module
- សំណុល : Servo Motor, HR-SR04
- វិធានការកាំ
  - ការបង្កើតនៃការទូទាត់ Ultrasonic SR04 ដោយពាក្យិក 5v និង GND បន្ទប់បន្ទប់ Echo ដោយពាក្យិក IN1, Trig និង OUT1
  - ការបង្កើតនៃការការពាក្យិក 5v និង GND ដោយពាក្យិក 5v និង GND បន្ទប់បន្ទប់ PIN 18 ដោយពាក្យិក 5v និង GND
  - ការបង្កើតនៃការការពាក្យិក 5v និង GND ដោយពាក្យិក 5v និង GND បន្ទប់បន្ទប់ PIN 18 ដោយពាក្យិក 5v និង GND
- ទូទាត់
  - ការបង្កើតនៃការការពាក្យិក 5v និង GND ដោយពាក្យិក 5v និង GND បន្ទប់បន្ទប់ PIN 18 ដោយពាក្យិក 5v និង GND
  - ការបង្កើតនៃការការពាក្យិក 5v និង GND ដោយពាក្យិក 5v និង GND បន្ទប់បន្ទប់ PIN 18 ដោយពាក្យិក 5v និង GND
  - ការបង្កើតនៃការការពាក្យិក 5v និង GND ដោយពាក្យិក 5v និង GND បន្ទប់បន្ទប់ PIN 18 ដោយពាក្យិក 5v និង GND

# Lab 5 : កំណើនគម្រោង

- เป้าหมาย : ใช้งาน PWM เพื่อควบคุม Servo Motor และใช้ร่วมกับ Ultrasonic Module
  - โมดูล : Servo Motor, HR-SR04
  - วิธีการทำ :
    - ทำการเชื่อมต่อ Ultrasonic SR04 เข้ากับ KidBright โดยต่อไฟเลี้ยง 5v และ GND บนบอร์ด, Echo เข้ากับ IN1, Trig เข้ากับ OUT1
    - ทำการต่อ Servo มอเตอร์เข้ากับบอร์ดโดยต่อ GND เข้าสายไฟลสีน้ำตาล ไฟ 5v สายสีแดง และต่อ PIN 18 กับสายไฟสีส้ม
    - ทำการสร้างหมวดหมู่ปลั๊กอินใหม่ folder “motor” โดยแสดงคำว่า “Motor” “มอเตอร์” ใน IDE
  - จกย :
    - ทำการสร้างปลั๊กจาก Class ที่กำหนดให้ใน Lab Direction {kkms}
    - ทำการเขียนโปรแกรมด้วย Block เพื่อทดสอบการทำงานของ Servo Motor {kidbyte}
    - ทำการเขียนโปรแกรมด้วย Block สั่งการ Servo Motor จาก Sonar Module

# Lab 5 : កំណើនគម្រោង

- เป้าหมาย : ใช้งาน PWM เพื่อควบคุม Servo Motor และใช้ร่วมกับ Ultrasonic Module
  - โมดูล : Servo Motor, HR-SR04
  - วิธีการทำ :
    - ทำการเชื่อมต่อ Ultrasonic SR04 เข้ากับ KidBright โดยต่อไฟเลี้ยง 5v และ GND บนบอร์ด, Echo เข้ากับ IN1, Trig เข้ากับ OUT1
    - ทำการต่อ Servo บอเตอร์เข้ากับบอร์ดโดยต่อ GND เข้าสายไฟลสีบ้ำตาล ไฟ 5v สายสีแดง และต่อ PIN 18 กับสายไฟลสีส้ม
    - ทำการสร้างหน่วยบัญญัติอินไหม folder “motor” โดยแสดงคำว่า “Motor” “บอเตอร์” ใน IDE
  - จกย :
    - ทำการสร้างบล็อกจาก Class ที่กำหนดให้ใน Lab Direction {kkms}
    - ทำการเขียนโปรแกรมด้วย Block เพื่อกดสอบการทำงานของ Servo Motor {kidbyte}
    - ทำการเขียนโปรแกรมด้วย Block สั่งการ Servo Motor จาก Sonar Module {yeah}

# การแปลงโค้ดเป็นบล็อก

- 1. คัดลอกปลั๊กอิน kbgpio จาก lab3 เปลี่ยนชื่อเป็น kbdht



- 2. ใน folder kbdht เปลี่ยนจาก kbgpio เป็น kbdht ก็จะหมด



- 3. ในไฟล์ kbdht.h ทำการแก้ไขคำว่า KBGPI0 เป็น KBDHT ก็จะหมด และลบ function setOutput และ digitalWrite ออก

```
#ifndef __KBDHT_H__
#define __KBDHT_H__
#include "driver.h"
#include "device.h"
#include "driver/gpio.h"
class KBDHT : public Device {
private:
public:
    // constructor
    KBDHT(void);
    // override
    void inline init(void){ return; }
    void inline process(Driver *drv){ error = false; initialized = true; }
    int inline prop_count(void){ return 0; }
    bool inline prop_name(int index, char *name){ return false; }
    bool inline prop_unit(int index, char *unit){ return false; }
    bool inline prop_attr(int index, char *attr){ return false; }
    bool inline prop_read(int index, char *value){ return false; }
    bool inline prop_write(int index, char *value){ return false; }
    // method
};
#endif
```

# การแปลงโค้ดเป็นบล็อก (ต่อ)

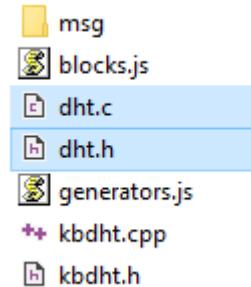
- 4. ในไฟล์ kbdht.cpp เปลี่ยน KBGPI0 เป็น KBDHT ก็งหมด

```
#include "esp_system.h"
#include "kidbright32.h"
#include "kbdht.h"
KBDHT::KBDHT(void) {
    ...
}
```

- 5. เราจะใช้ folder นี้เป็นโครงสร้างสำหรับกำกัลกอิน (ให้ทำการสำรวจโฟล์เดอร์นี้ไว้)

# การใช้โค้ดจาก library มาทำเป็นบล็อก (dht11)

- 1. นำไฟล์ dht.c และ dht.h จาก direction มาวางไว้ใน folder kbdht



- 2. ตรวจสอบบញ្ជី function จาก dht.h អ្នែក ឬ example เพื่อដូច្នេរការใช้งาน

```
#include <stdio.h>
#include <freertos/FreeRTOS.h>
#include <freertos/task.h>
#include <dht.h>

static const dht_sensor_type_t sensor_type = DHT_TYPE_DHT11;
static const gpio_num_t dht_gpio = 17;

void dht_test(void *pvParameters)
{
    int16_t temperature = 0;
    int16_t humidity = 0;

    // DHT sensors that come mounted on a PCB generally have
    // pull-up resistors on the data pin. It is recommended
    // to provide an external pull-up resistor otherwise...

    //gpio_set_pull_mode(dht_gpio, GPIO_PULLUP_ONLY);

    while (1)
    {
        if (dht_read_data(sensor_type, dht_gpio, &humidity, &temperature) == ESP_OK)
            printf("Humidity: %d%% Temp: %dC\n", humidity / 10, temperature / 10);
        else
            printf("Could not read data from sensor\n");

        vTaskDelay(300 / portTICK_PERIOD_MS);
    }
}
```

# การใช้โค้ดจาก library มาทำเป็นบล็อก (dht11) (ต่อ)

- 3. แก้ไขไฟล์ kbdht.h โดยสร้าง function เพื่อเรียกใช้ dht.h วันที่

```
bool inline prop_read(int index, char *value){ return fa  
bool inline prop_write(int index, char *value){ return f  
// method  
float readTemperature(int pin);  
float readHumidity(int pin);  
};
```

```
#include "driver/gpio.h"  
#include "dht.h"  
class KBDHT : public Devi;
```

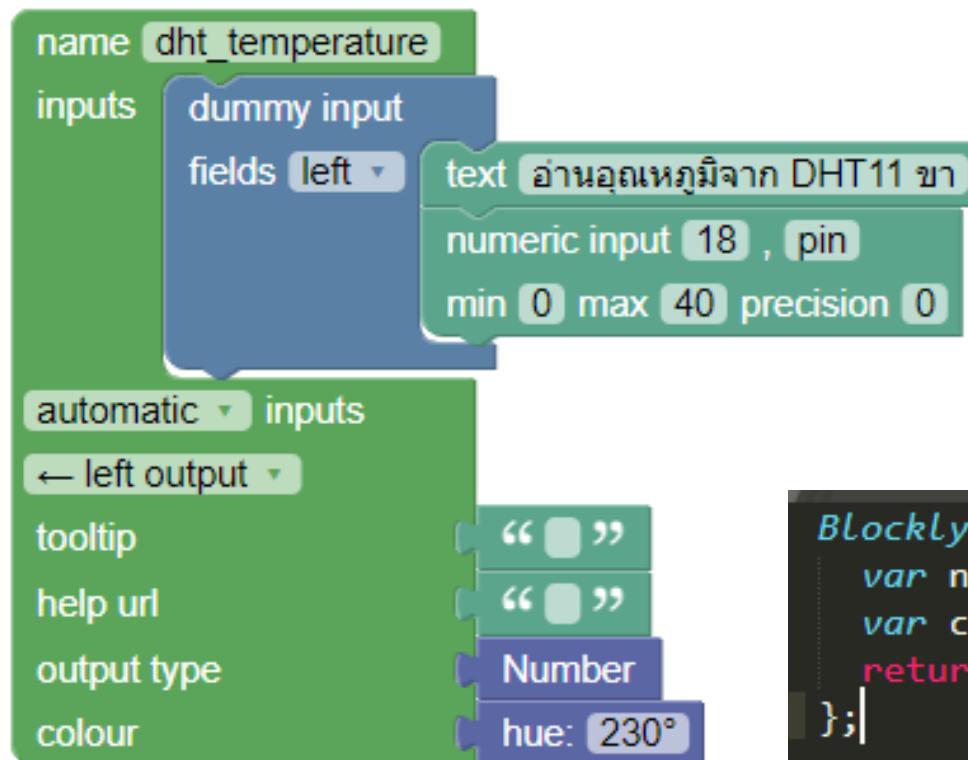
- 4. ทำการแก้ไขไฟล์ kbdht.cpp เรียกใช้งาน function ใน dht.h

```
#include "esp_system.h"  
#include "kidbright32.h"  
#include "kbdht.h"  
static const dht_sensor_type_t sensor_type = DHT_TYPE_DHT11;  
KBDHT::KBDHT(void) {  
  
}  
float KBDHT::readTemperature(int pin){  
    int16_t temperature = 0;  
    int16_t humidity = 0;  
    if (dht_read_data(sensor_type,(gpio_num_t)pin, &humidity, &temperature) == ESP_OK)  
        return temperature/10;  
    return -1;  
}  
float KBDHT::readHumidity(int pin){  
    int16_t temperature = 0;  
    int16_t humidity = 0;  
    if (dht_read_data(sensor_type,(gpio_num_t)pin, &humidity, &temperature) == ESP_OK)  
        return humidity/10;  
    return -1;  
}
```

# การใช้โค้ดจาก library มาทำเป็นบล็อก (dht11) (ต่อ)

- 5. ทำการอ่านแบบ block และแก้ไขไฟล์ blocks.js และ generators.js

อ่านอุณหภูมิจาก DHT11 ขา 18

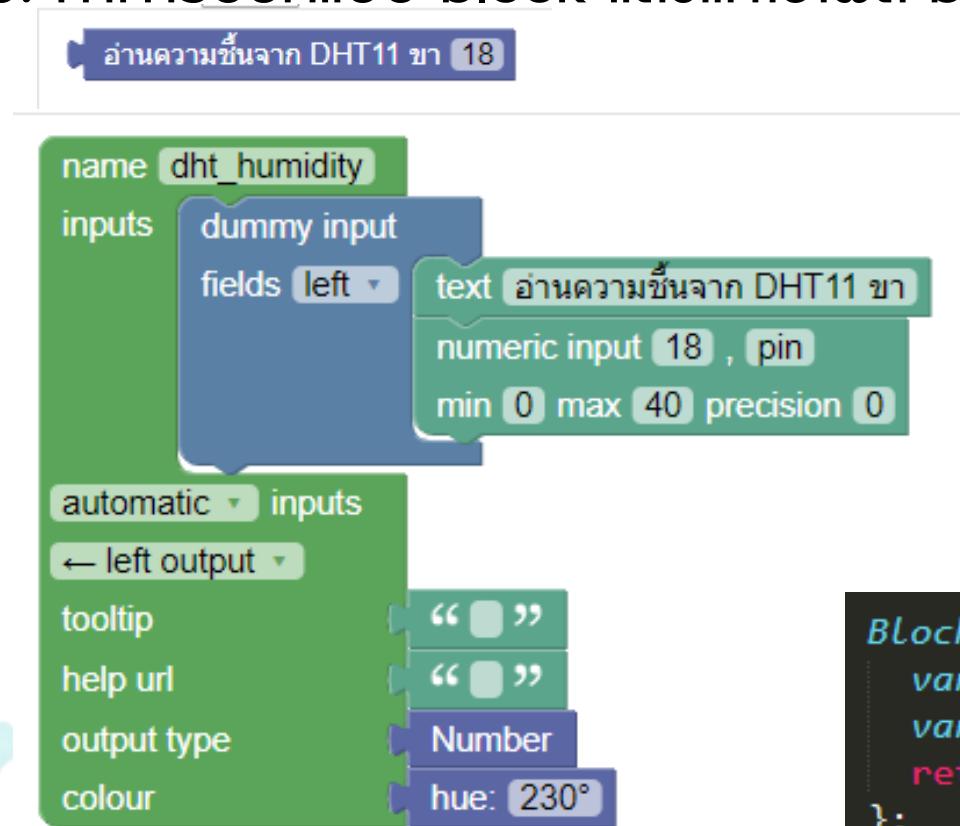


```
Blockly.Blocks['dht_temperature'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("อ่านอุณหภูมิจาก DHT11 ขา")
      .appendField(new Blockly.FieldNumber(18, 0, 40), "pin");
    this.setOutput(true, "Number");
    this.setColour(230);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.JavaScript['dht_temperature'] = function(block) {
  var number_pin = block.getFieldValue('pin');
  var code = 'DEV_IO.KBDHT().readTemperature('+number_pin+')';
  return [code, Blockly.JavaScript.ORDER_NONE];
};
```

# การใช้โค้ดจาก library มาทำเป็นบล็อก (dht11) (ต่อ)

- 5. ทำการอุดแบบ block แล้วแก้ไขไฟล์ blocks.js และ generators.js



```
▼ Blockly.Blocks['dht_humidity'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("อ่านความชื้นจาก DHT11 ขา")
      .appendField(new Blockly.FieldNumber(18, 0, 40), "pin");
    this.setOutput(true, "Number");
    this.setColour(230);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

```
Blockly.JavaScript['dht_humidity'] = function(block) {
  var number_pin = block.getFieldValue('pin');
  var code = 'DEV_IO.KBDHT().readHumidity('+number_pin+')';
  return [code, Blockly.JavaScript.ORDER_NONE];
};
```

# Lab 6 : วัดความชื้นในเรือนเพาะปลูก

- เป้าหมาย : ใช้งาน DHT11 เพื่อควบคุม Servo Motor และใช้ร่วมกับ Relay
- โนดูล : Servo Motor, Relay, DHT11
- วิธีการทำ :
  - ทำการเชื่อมต่อ Relay โดยควบคุมจาก OUT1 และ OUT2 และ Servo เข้ากับ KidBright ขา 19
  - ทำการเชื่อมต่อ DHT 11 เข้ากับ KidBright ต่อ “+” เข้ากับ 3V3 “-” เข้ากับ GND ต่อ OUT ไปขา 18
- โจทย์ :
  - ทำการทดสอบบล็อกอ่านค่าอุณหภูมิและความชื้นที่สร้างเสร็จแล้ว
  - เขียนโปรแกรมแบบบล็อกหาก
    - อุณหภูมิ สูงกว่า 30°C ให้ relay เปิดทำงาน และปิดเมื่อต่ำกว่า 30°C
    - ความชื้นสูงกว่า 80% ให้หมุน Servo ไป 180° หากต่ำกว่าให้ servo หมุนกลับ

# การขยายพอร์ตด้วย PCF8574

- ออกแบบบล็อกสำหรับการตั้งค่า I2C

The screenshot shows the Blockly interface with two main sections: 'Block Definition' and 'Generator stub'.

**Block Definition:** JavaScript

```
Blockly.Blocks['pcf8574_init'] = {
  init: function() {
    this.appendDummyInput()
        .appendField("ตั้งค่า pcf8574 address")
        .appendField(new Blockly.FieldNumber(39, 0, 255), "address");
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(230);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

**Generator stub:** JavaScript

```
Blockly.JavaScript['pcf8574_init'] = function(block) {
  var number_address = block.getFieldValue('address');
  // TODO: Assemble JavaScript into code variable.
  var code = '...;\n';
  return code;
};
```

**pcf8574\_init Block Details:**

- name:** pcf8574\_init
- inputs:** dummy input
- fields:** left
  - text: ตั้งค่า pcf8574 address
  - numeric input: 39, address
  - min: 0, max: 255, precision: 0
- automatic inputs:** top+bottom connections
- tooltip:** “ ”
- help url:** “ ”
- top type:** any
- bottom type:** any
- colour:** Number hue: 230°

# การขยายพอร์ตด้วย PCF8574 (ต่อ)

- ออกแบบบล็อกคำสั่งรับ ใช้งานขา GPIO (เขียนค่า)

The screenshot shows the Blockly interface with the following details:

- Block Definition:** JavaScript
- Code:**

```
Blockly.Blocks['pcf8574_write'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("เขียนค่า pcf8574 address")
      .appendField(new Blockly.FieldNumber(39, 0, 255), "address");
    this.appendValueInput("pin")
      .setCheck("Number")
      .appendField("ขาที่");
    this.appendValueInput("value")
      .setCheck(null)
      .appendField("ค่า");
    this.setInputsInline(true);
  }
};
```
- Generator stub:** JavaScript
- Code:**

```
Blockly.JavaScript['pcf8574_write'] = function(block) {
  var number_address = block.getFieldValue('address');
  var value_pin = Blockly.JavaScript.valueToCode(block, 'pin', Blockly.JavaScript.ORDER_ATOMIC);
  var value_value = Blockly.JavaScript.valueToCode(block, 'value', Blockly.JavaScript.ORDER_ATOMIC);
  // TODO: Assemble JavaScript into code variable.
  var code = '...;\n';
  return code;
};
```
- Block Details:**
  - Name:** pcf8574\_write
  - Inputs:**
    - dummy input
    - fields left:
      - text เขียนค่า pcf8574 address
      - numeric input 39 , address
      - min 0 max 255 precision 0
    - value input pin
      - fields left:
        - text ขาที่
        - type Number
    - value input value
      - fields left:
        - text ค่า
        - type any
  - inline inputs**
  - top+bottom connections**
  - tooltip**: “ ”
  - help url**: “ ”
  - top type**: any
  - bottom type**: any
  - colour**: hue: 230°

# การขยายพอร์ตด้วย PCF8574 (ต่อ)

- ออกแบบล็อกคสำหรับ ใช้งานขา GPIO (อ่านค่า)

The screenshot shows the Blockly workspace with the following details:

- Block Definition:** A dropdown menu labeled "Block Definition" is open, set to "JavaScript". It contains the following code:

```
.appendField("อ่านค่า pcf8574 address")
.appendField(new Blockly.FieldNumber(39, 0, 255), "address");
this.appendValueInput("pin")
.setCheck("Number")
.appendField("ขาที่");
this.setInputsInline(true);
this.setOutput(true, "Number");
this.setColour(230);
this.setTooltip("");
this.setHelpUrl("");
};
```
- Generator stub:** A dropdown menu labeled "Generator stub" is open, set to "JavaScript". It contains the following code:

```
Blockly.JavaScript['pcf8574_read'] = function(block) {
  var number_address = block.getFieldValue('address');
  var value_pin = Blockly.JavaScript.valueToCode(block, 'pin', Blockly.JavaScript.ORDER_ATOMIC);
  // TODO: Assemble JavaScript into code variable.
  var code = '...';
  // TODO: Change ORDER_NONE to the correct strength.
  return [code, Blockly.JavaScript.ORDER_NONE];
};
```

# Lab 7 : ขยายขา IO ให้มากขึ้น

- เป้าหมาย : ใช้งาน pcf8574 เพื่อขยายขา IO ให้ KidBright
- โมดูล : Relay
- วิธีการทำ :
  - ทำการเชื่อมต่อ PCF8574 เข้ากับ KidBright ผ่าน KBCHAIN
  - ต่อ Relay เข้ากับ PCF8574 และทดสอบใช้งาน
- โจทย์ :
  - เขียนโปรแกรมแบบบล็อกคเพื่อส่งปิดเปิด Relay ด้วย pcf8574

# การขยายพอร์ตด้วย Analog ด้วย ADS1115

- ออกแบบบล็อกสำหรับ สำหรับตั้งค่า I2C

The screenshot shows the Blockly interface with two main sections: 'Block Definition' and 'Generator stub'.

**Block Definition:** JavaScript

```
Blockly.Blocks['ads1115_init'] = {
  init: function() {
    this.appendDummyInput()
        .appendField("ตั้งค่า ads1115 address")
        .appendField(new Blockly.FieldNumber(72, 0, 255), "address");
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(230);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

**Generator stub:** JavaScript

```
Blockly.JavaScript['ads1115_init'] = function(block) {
  var number_address = block.getFieldValue('address');
  // TODO: Assemble JavaScript into code variable.
  var code = '...;\n';
  return code;
};
```

**Left Panel (Block Definition):**

- name:** ads1115\_init
- inputs:** dummy input
- fields:** left
  - text: ตั้งค่า ads1115 address
  - numeric input (72, address)
  - min (0) max (255) precision (0)
- automatic inputs:** top+bottom connections
- tooltip:** “ ”
- help url:** “ ”
- top type:** any
- bottom type:** any
- colour:** 230°

# การขยายพอร์ตด้วย Analog ด้วย ADS1115 (ต่อ)

- ออกแบบล็อกสำหรับ สำหรับการอ่านค่า Analog

The image shows the Blockly interface with the definition of a custom block named 'ads1115\_read'. The block has two input fields: 'dummy input' and 'value input [NAME]'. The 'value input' field is set to type 'Number'. The block also has an 'inline' input section with a tooltip and help URL, and an output type set to 'any'. The color is set to hue 230.

**Block Definition:** JavaScript

```
Blockly.Blocks['ads1115_read'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("อ่าน ads1115 address")
      .appendField(new Blockly.FieldNumber(72, 0, 255), "address");
    this.appendValueInput("NAME")
      .setCheck("Number")
      .appendField("ชานี");
    this.setInputsInline(true);
    this.setOutput(true, null);
    this.setColour(230);
    this.setTooltip("");
    this.setHelpUrl("");
  }
};
```

**Generator stub:** JavaScript

```
Blockly.JavaScript['ads1115_read'] = function(block) {
  var number_address = block.getFieldValue('address');
  var value_name = Blockly.JavaScript.valueToCode(block, 'NAME', Blockly.JavaScript.ORDER_ATOMIC);
  // TODO: Assemble JavaScript into code variable.
  var code = "...";
  // TODO: Change ORDER_NONE to the correct strength.
  return [code, Blockly.JavaScript.ORDER_NONE];
};
```

# Lab 7 : ใช้งาน External Analog ผ่าน ADS1115

- เป้าหมาย : ใช้งาน ADS1115 เพื่อขยายขา Analog ให้ KidBright
- โมดูล : ADS1115
- วิธีการทำ :
  - ทำการเชื่อมต่อ ADS1115 เข้ากับ KidBright ผ่าน KBCHAIN
  - ต่อ LDR เข้ากับ ADS1115
- โจทย์ :
  - เขียนโปรแกรมแบบบล็อกคเพื่อแสดงค่าความสว่างของ LDR



Thank you