

KidBright Project in Action

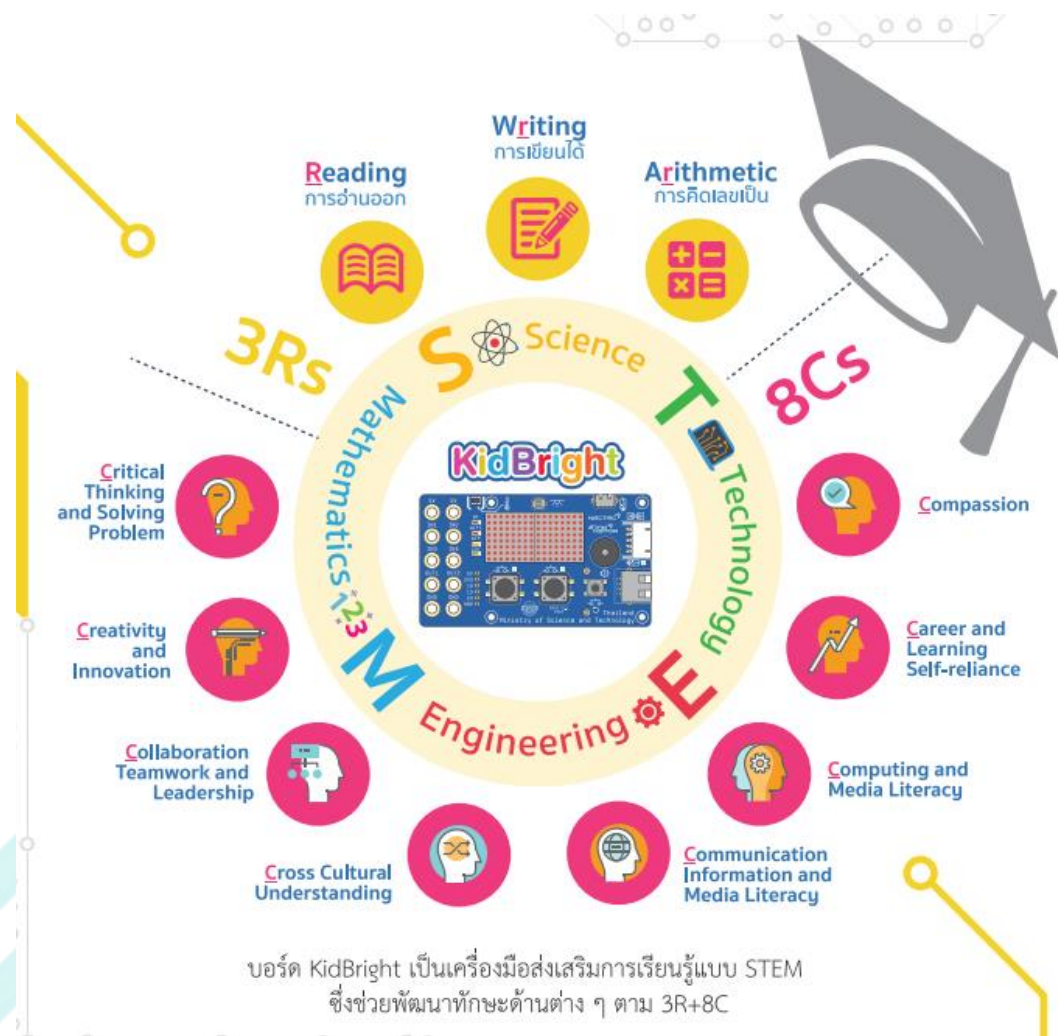
สร้างโปรเจกต์ใช้งานจริง



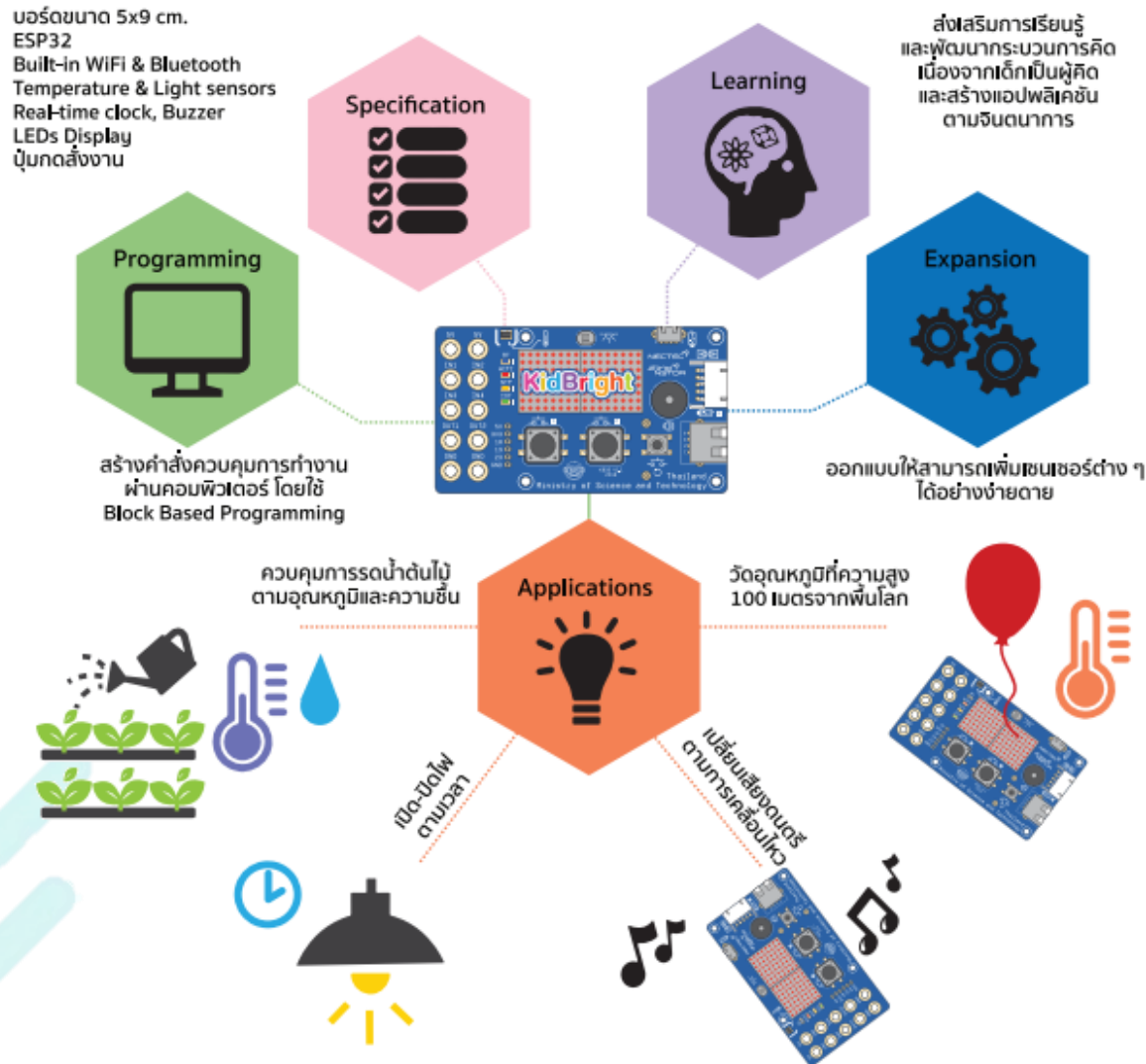
หัวข้ออบรม

- 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
- ปรึกษาโครงงานและตัวอย่าง plugin เพิ่มเติม

รู้จักบอร์ด KidBright

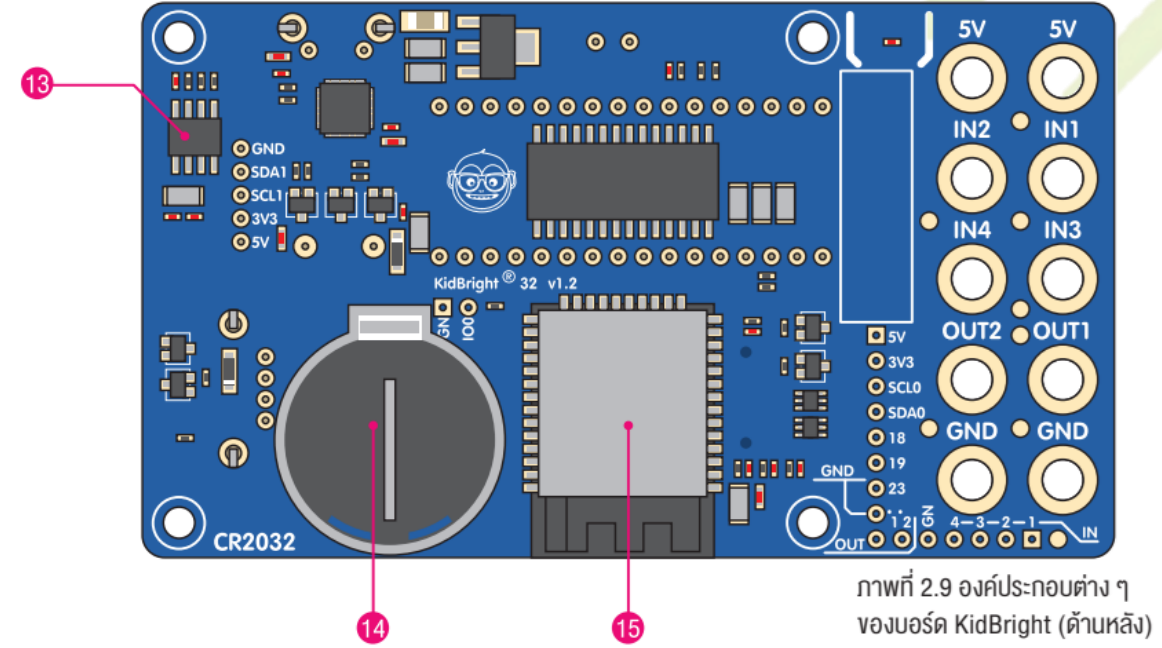
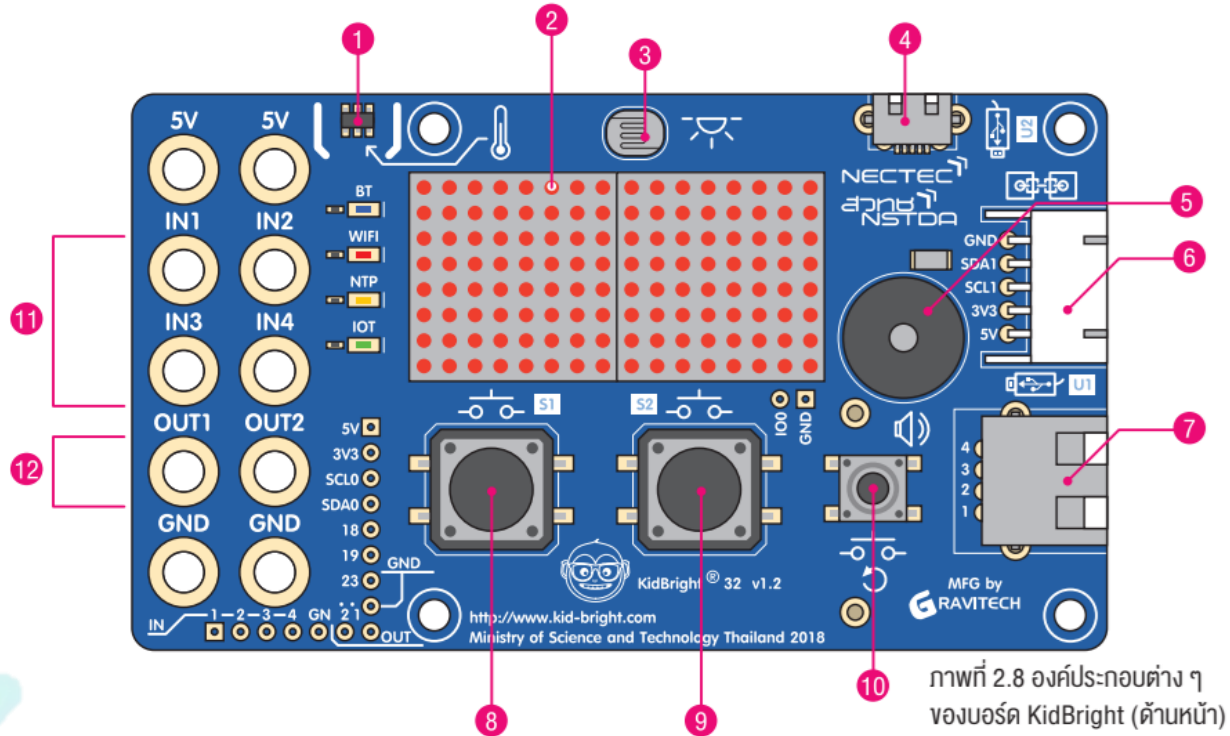


KidBright เอาไปทำอะไรได้บ้าง ?



- <https://www.kid-bright.org/showcase/article/250>
- [เครื่องเตือนภัยน้ำท่วม 24 ชั่วโมง](#) รร.สอวพิทยาคม จ.แพร่
- [กรรไลยฆะมดอัตโนมัติ](#) รร.อนุบาลแม่ฟ้าหลวง จ.เชียงราย
- [เครื่องแยกและนับเหรียญ](#) รร.บ้านทุ่งข้าวพวง จ.เชียงใหม่
- [ระบบฟ้าะวังความปลอดภัยในรถ](#) รร.เขียงคำวิทยาคม จ.พะเยา
- [เครื่องควบคุมการจ่ายสารเคมีอัจฉริยะ](#) บ้านทุ่งข้าวพวง จ.เชียงใหม่
- [ระบบแจ้งเตือนเหตุอุทกภัย](#) รร.วิทยาศาสตร์จุฬาภรณราชวิทยาลัย ตรัง จ.ตรัง
- [นวัตกรรมเครื่องตากปลาอัจฉริยะ](#) โดยใช้บอร์ด KidBright รร. สทิวพระวิทยา จ.สงขลา
- [ระบบควบคุมการรดน้ำภายในโรงเพาะเห็ดอัตโนมัติ](#) รร.บ้านปลายคลอง จ.สุราษฎร์ธานี
- [“ดนตรีอิเล็กทรอนิกส์” : การประดิษฐ์เครื่องดนตรีอย่างง่ายด้วย KidBright](#) รร. ท่าศาลาประสิทธิ์ศึกษา จ. นครศรีธรรมราช
- [ขบวนาแลกเพลจ \(Music Bin\)](#) รร.สาริตมหาวิทยาลัยเกษตรศาสตร์วิทยาเขตกำแพงแสน จ.นครปฐม
- และอีกมากมาย

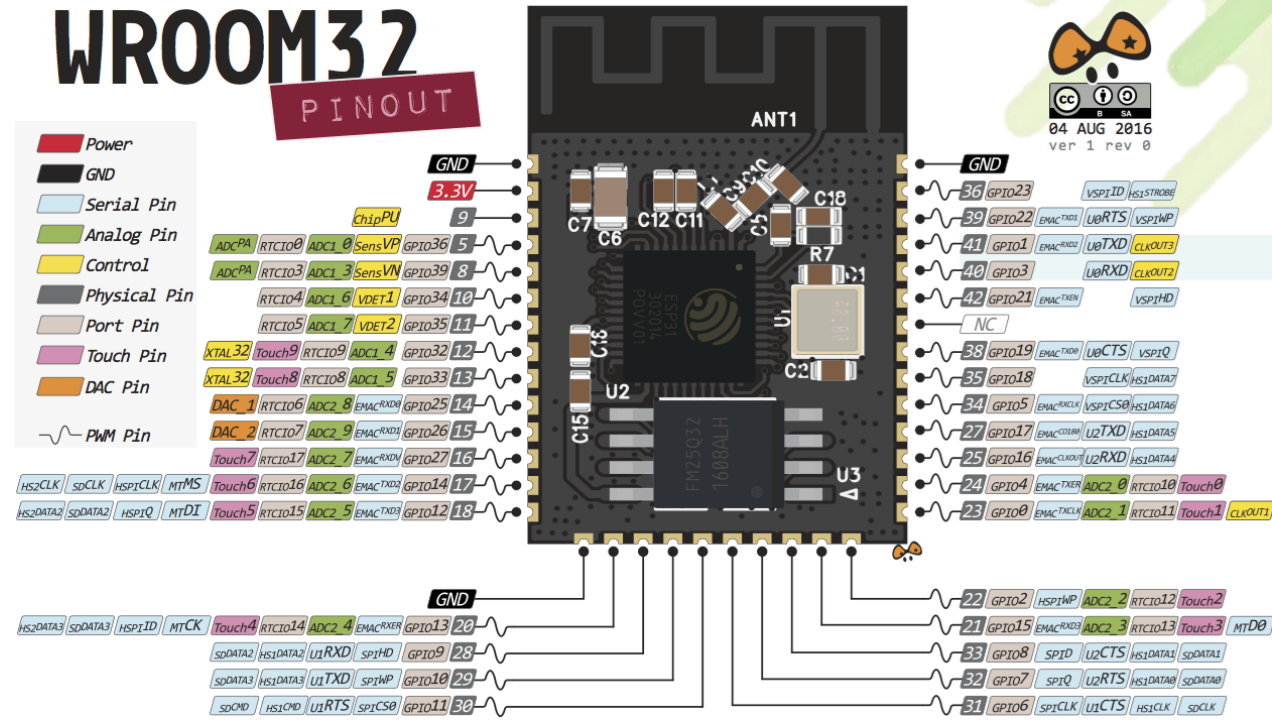
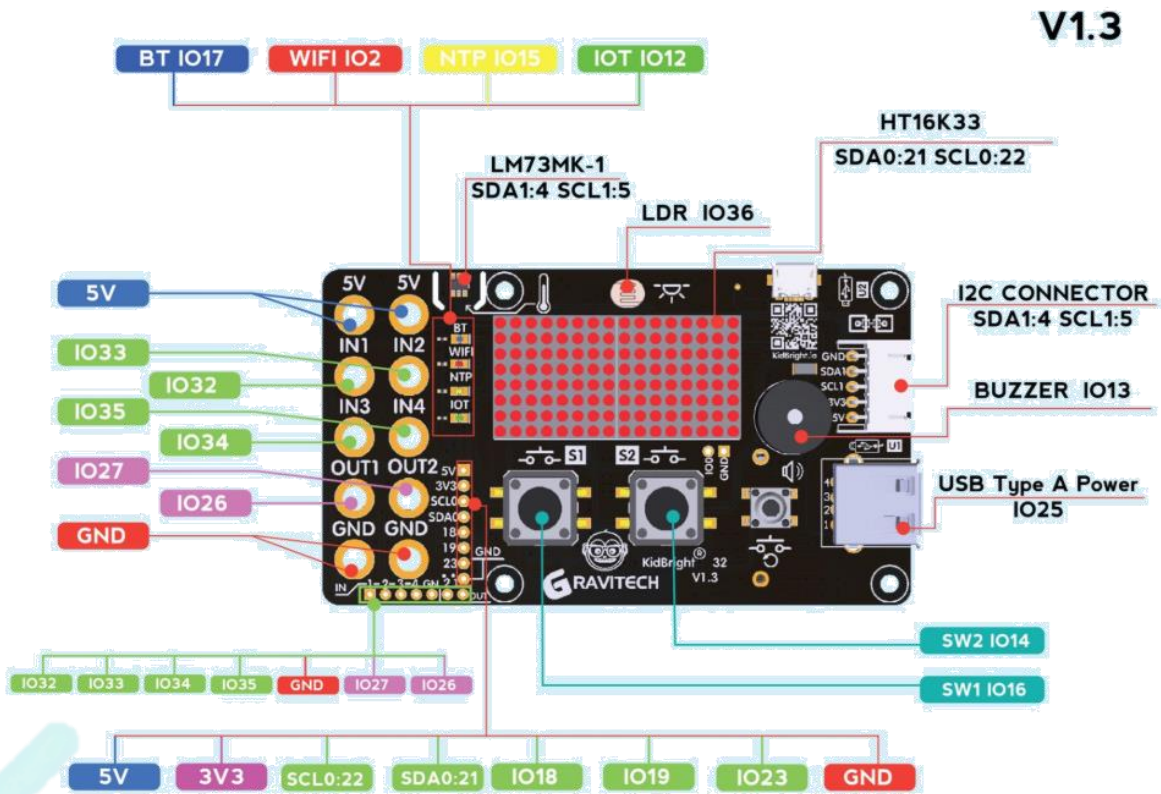
ส่วนประกอบของบอร์ด KidBright



- | | | |
|----------------------------|-------------------------|---------------------------|
| 1 เซนเซอร์วัดอุณหภูมิ | 2 LED แสดงผล | 3 เซนเซอร์วัดแสง |
| 4 ช่องเสียบสายไมโครยูเอสบี | 5 ลำโพง | 6 คอนเนกเตอร์ |
| 7 พอร์ตยูเอสบี | 8 สวิตช์ 1 | 9 สวิตช์ 2 |
| 10 สวิตช์รีเซ็ต | 11 ช่องสัญญาณอินพุต 1-4 | 12 ช่องสัญญาณเอาต์พุต 1-2 |

- | | | |
|--------------------|--------------------|-----------------------|
| 13 นาฬิกาเรียลไทม์ | 14 รางใส่แบตเตอรี่ | 15 ส่วนควบคุมการทำงาน |
|--------------------|--------------------|-----------------------|

การเชื่อมต่อบอร์ด KidBright



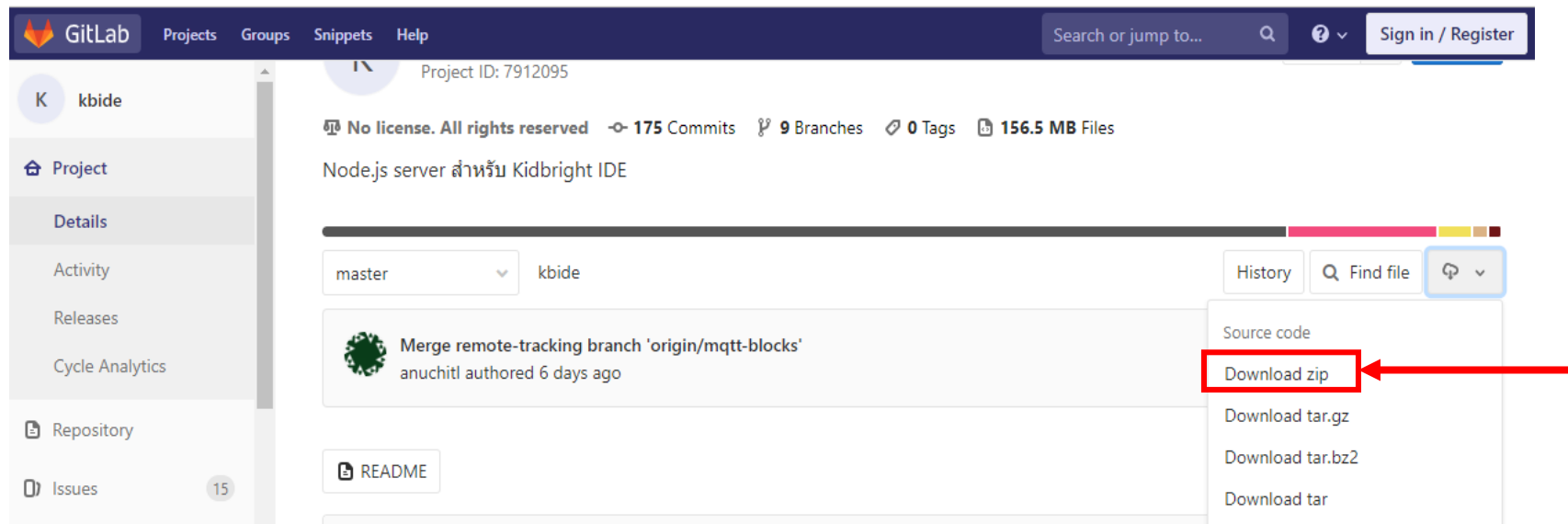
- เขียนโปรแกรมโดยใช้สถาปัตยกรรม ESP32
- ใช้ ESP-IDF ในการเข้าถึง I/O
- I/O อิสระ 7 ช่อง (22,21,18,19,23,4,5)
- Analog ของ ESP32 ควรใช้แค่ ADC1 เหลือ pin (5) pin เดียว

ทำไมต้องใช้ IDE NodeJS version ?

- ทำการพัฒนางานได้รวดเร็วกว่า (เสียเวลาแค่การติดตั้งครั้งแรกเท่านั้น)
- ไม่ต้องรออัปเดตก่อนเปิดโปรแกรม
- ติดตั้งได้ทุกที่ copy folder ได้
- เมื่อโปรแกรม error สามารถรับใหม่ได้ทันที (ตัว IDE หลักไม่มี error handling)
- ค้นหาไฟล์และแก้ไขได้ง่าย
- สามารถเขียนโค้ดได้หลายหน้าต่าง (เปิด Incognito โหมดบนเว็บ)
- สามารถเปิดเขียนโค้ดแบบหลายเครื่องได้ ถึงแม้จะมีบอร์ดแค่อันเดียว !

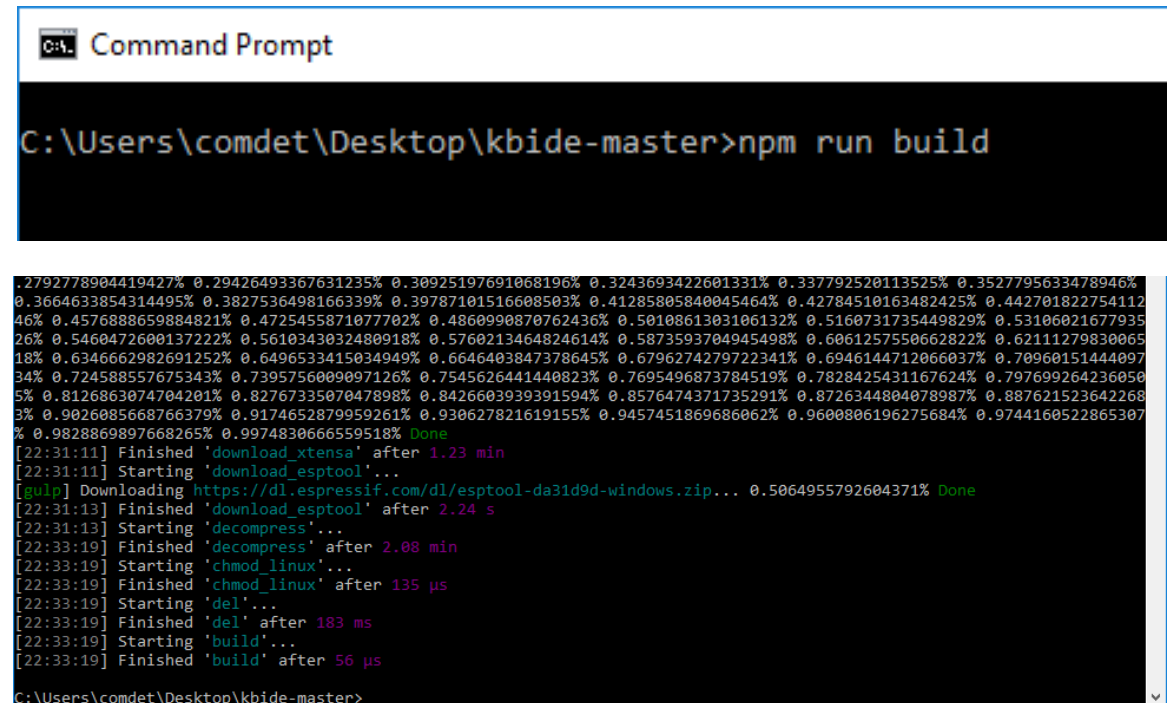
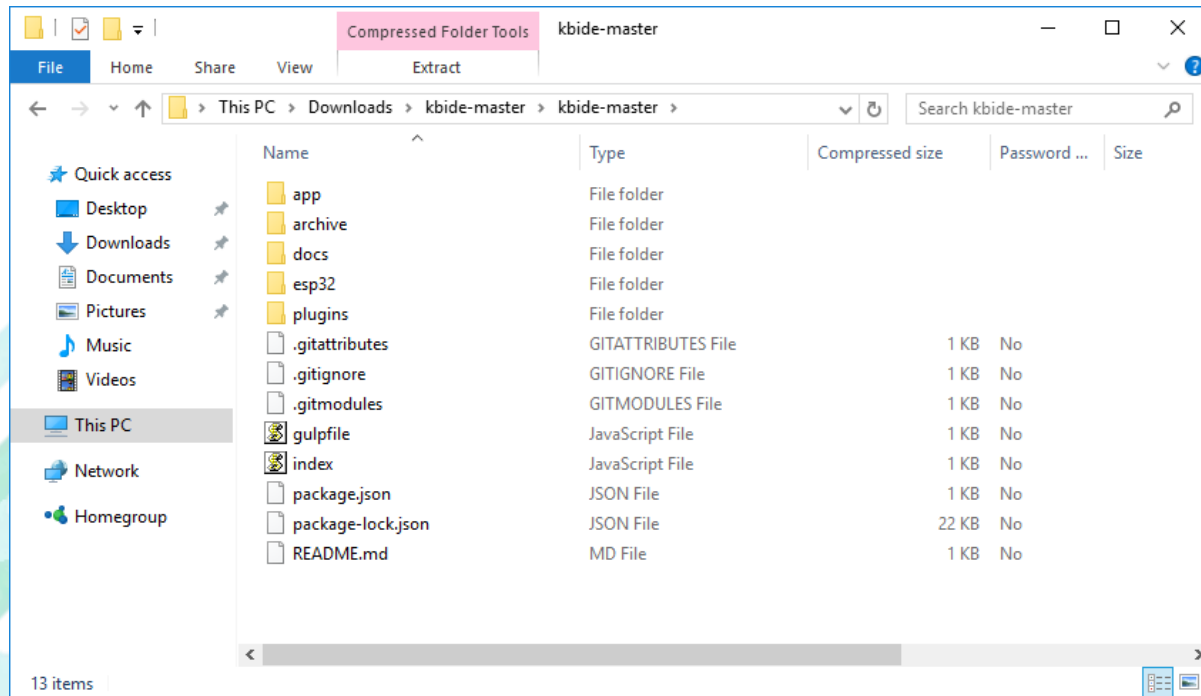
การติดตั้ง IDE NodeJS version (ต่อ)

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



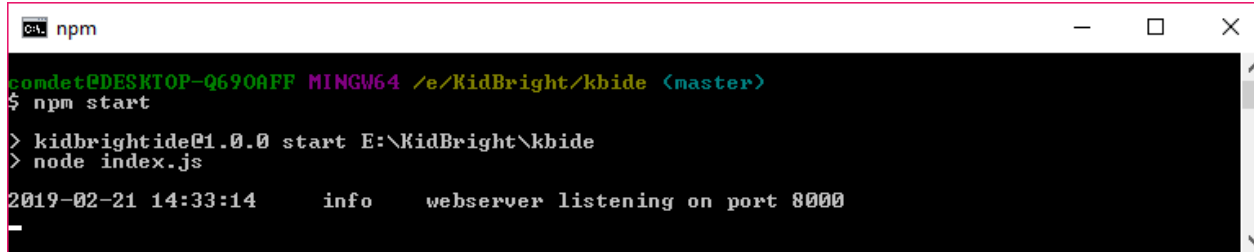
การติดตั้ง IDE NodeJS version (ต่อ)

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



ทดสอบเขียนโปรแกรมแสดงข้อความ

- รับคำสั่ง npm start ที่ command prompt

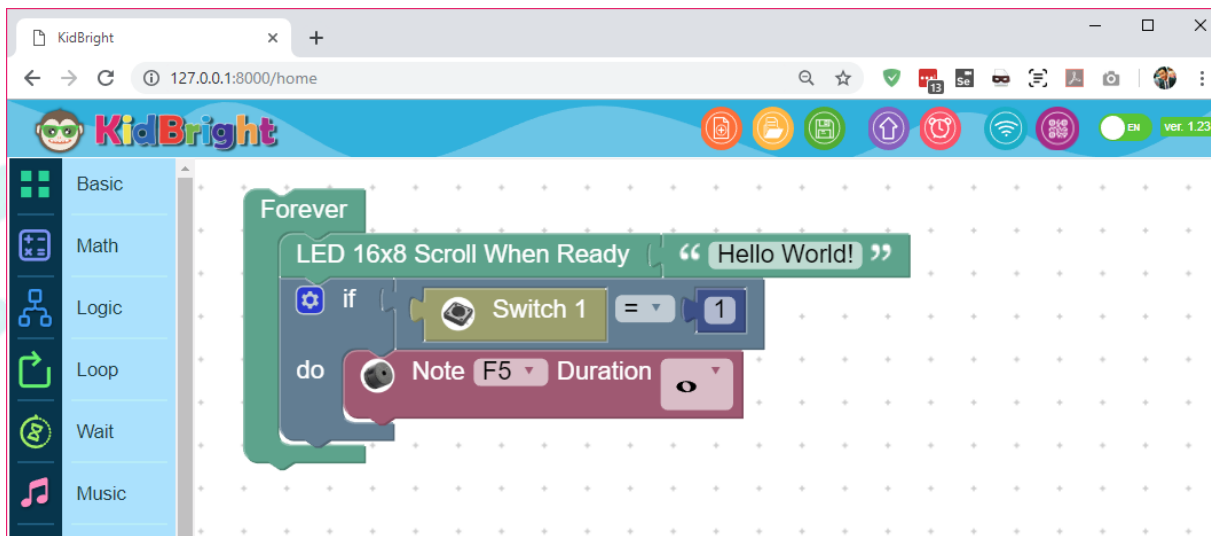


```
cmd: npm
comdet@DESKTOP-Q690AFF MINGW64 /e/KidBright/kbide <master>
$ npm start

> 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>
- ทำการทดสอบเขียนโปรแกรมอย่างง่าย

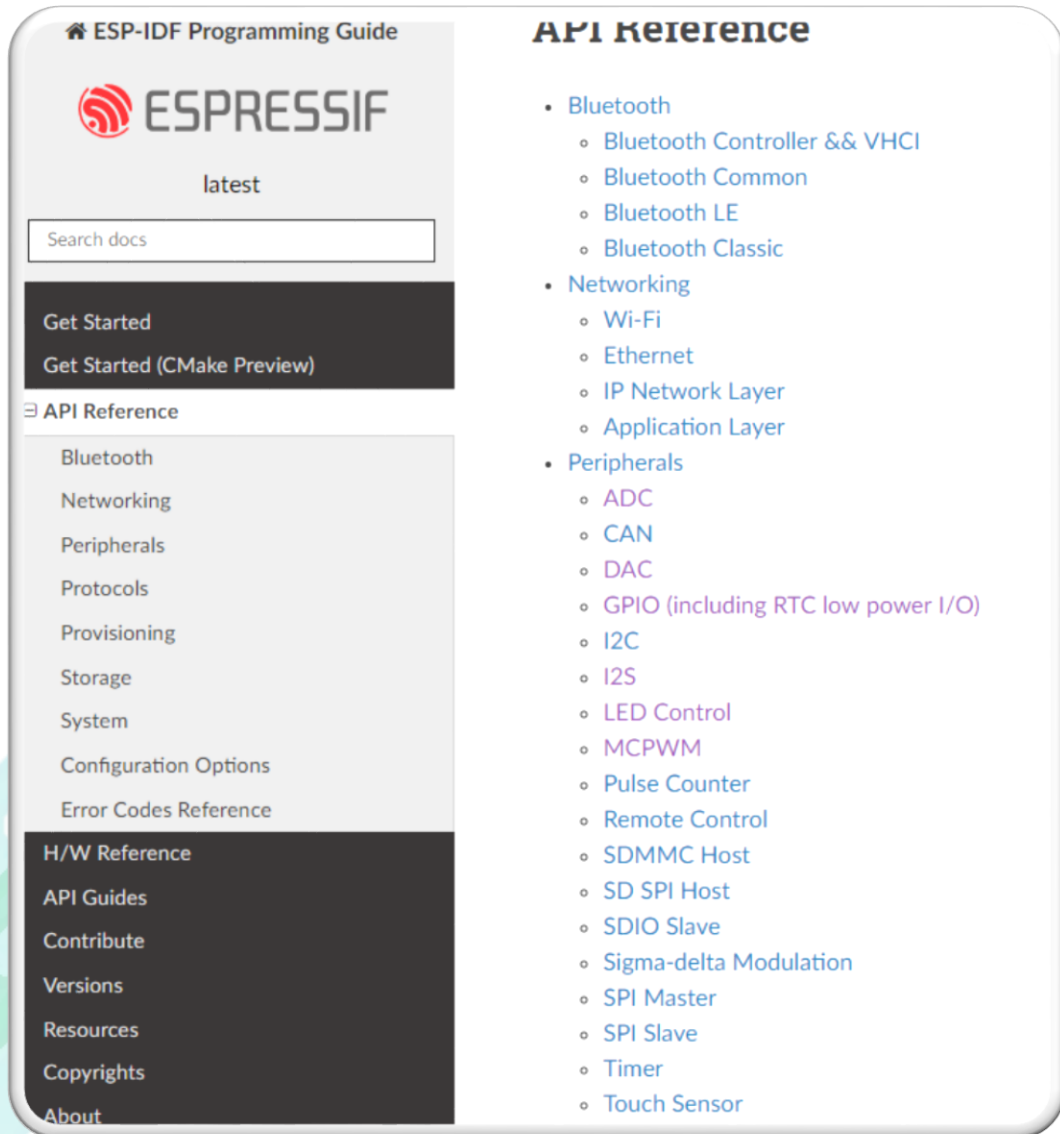


Lab 1 : เครื่องตั้งเวลารดน้ำต้นไม้

- 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 ให้เขียนโค้ดเพื่อสั่งการ โดยหากกด S1 ให้ Relay CH1 ตัด กดอีกครั้งเพื่อดับ และ S2 ให้ทำลักษณะเดียวกันกับ Relay CH2
 - ข้อ 2 ทำการเขียนโปรแกรมเพิ่ม โดยให้ relay ทำงานตอน 6 โมงเช้าและ 5 โมงเย็น เป็นเวลา 2 นาที แล้วปิดการทำงาน
 - ข้อ 3 เขียนโปรแกรมให้ผู้ใช้ตั้งเวลาปิดเปิดเองได้ โดยกด S1 เพื่อตั้งค่า ชั่วโมง S1 เพื่อตั้ง นาที
 - ข้อ 4 ทำการแก้ไขปัญหของโค้ดข้อ 3



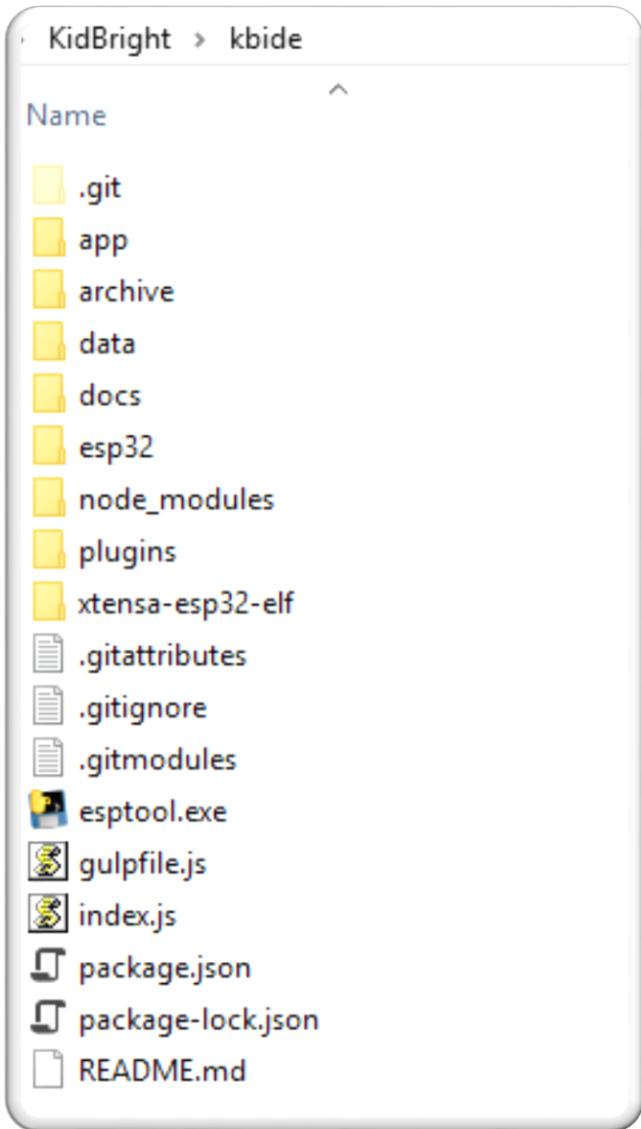
รู้จักกับ ESP IoT Development Framework (ESP-IDF)



- เป็นชุดคอมไพล์เลอร์ ESP32 ที่พัฒนาโดยบริษัท Espressif
- ทำงานครอบงำบน 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 เท่านั้นไม่สามารถไปใช้ RTOS-SDK version เก่า ๆ ได้
- IDF เน้นเป็น component base เป็นหลัก
- ข้อมูลการพัฒนาและเอกสารการใช้งานทั้งหมดสามารถหาเพิ่มเติมได้ผ่าน

<https://docs.espressif.com/projects/esp-idf/en/latest/>

โครงสร้างของ KBIDE



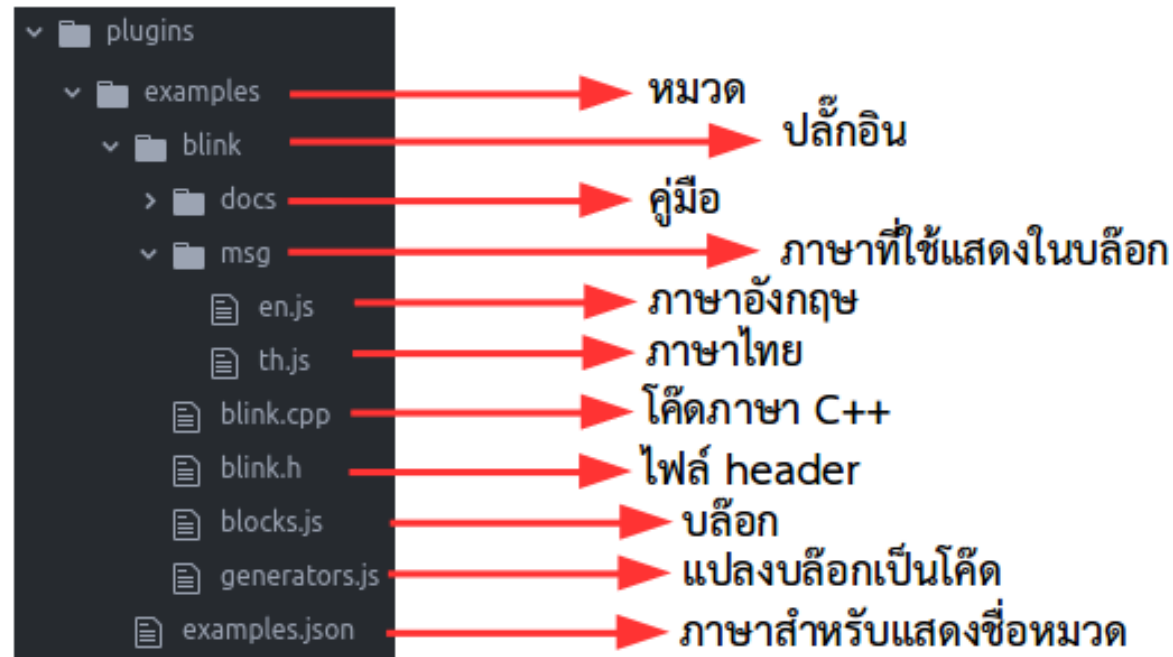
- KBIDE เขียนด้วย JavaScript แยกเป็นฝั่ง Server กับ UI โดยโค้ดทั้งหมดเก็บอยู่ใน folder “app”
- ใน folder “esp32/build/xx-xx-xx../” จะมีไฟล์ user_app.cpp ซึ่งโค้ดโปรแกรมภาษา C ที่ดัดบล็อก และ xx-xx-xx.bin คือ firmware (xx-xx-xx.. คือหมายเลข Serial แต่ละบอร์ด)

A screenshot of a Sublime Text editor window showing the code for user_app.cpp. The code is in C++ and includes comments in Thai. It defines a task for an ESP32, sets up a while loop to handle button presses, and updates a display (ht16k33) with the current hour and minute. The code is as follows:

```
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) {
```

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

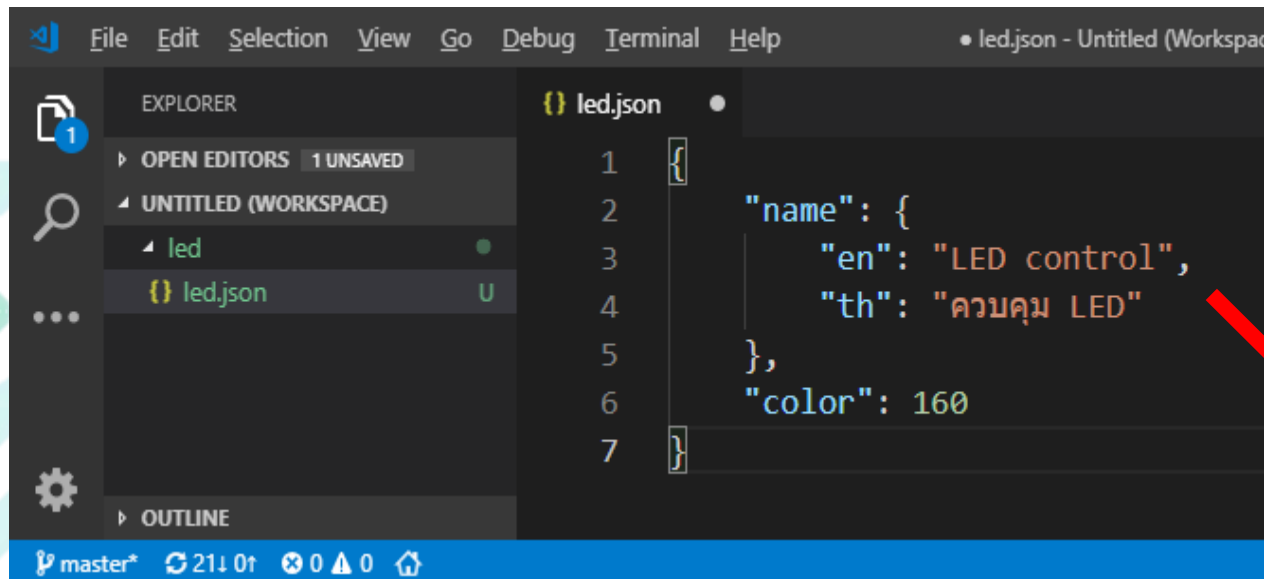
โครงสร้างของปลั๊กอิน KBIDE



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

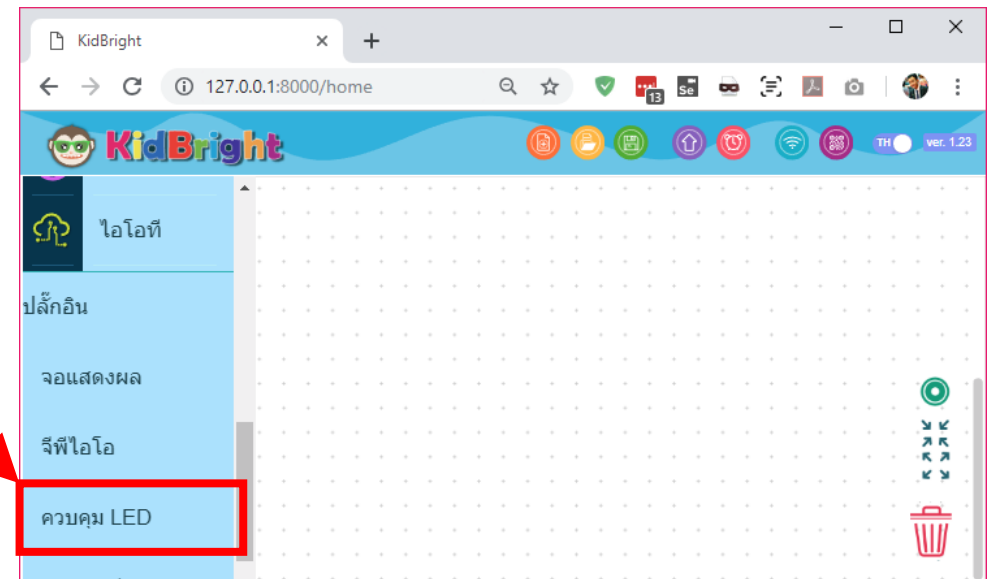
Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย

- 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



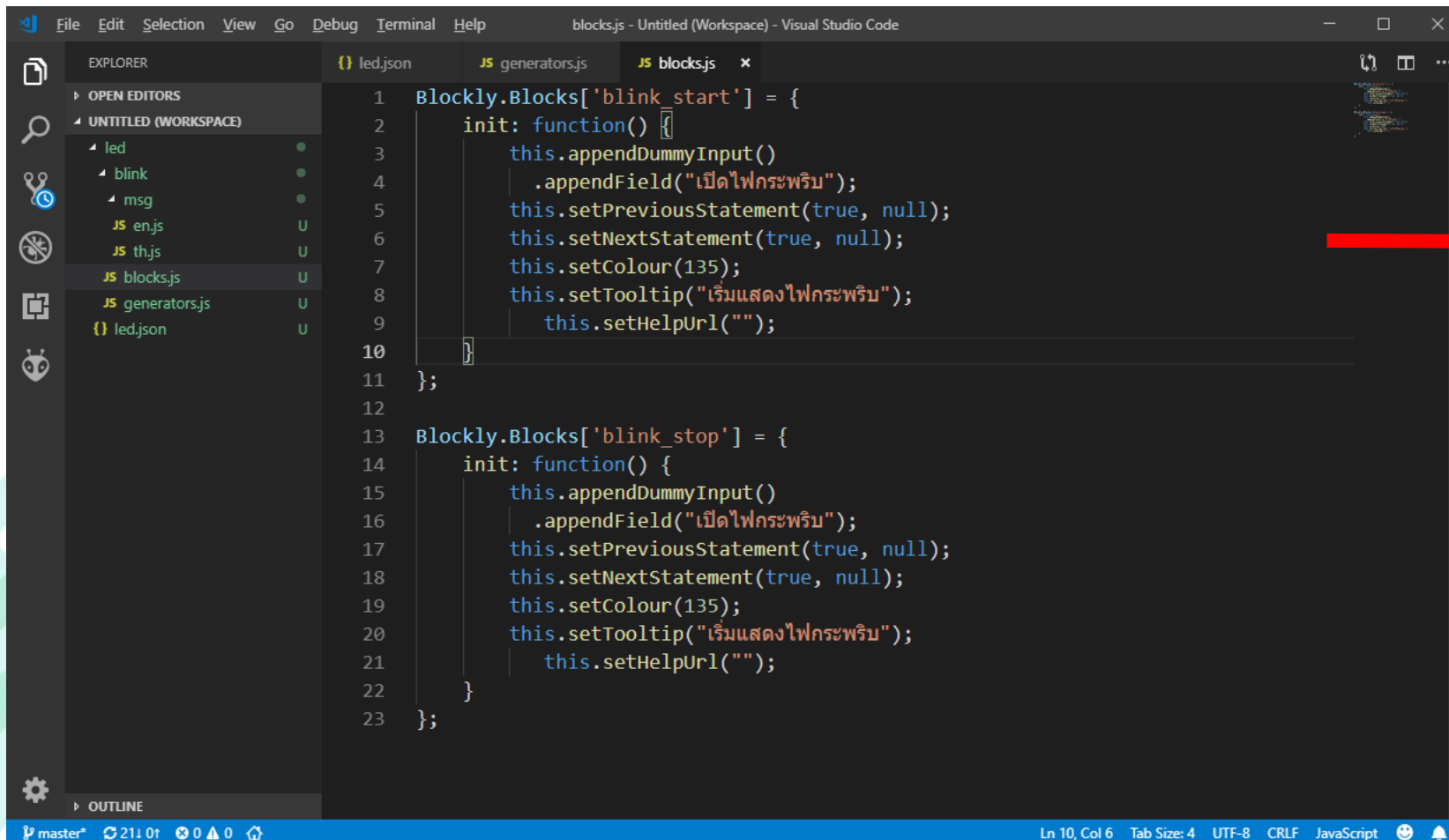
The screenshot shows the VS Code editor with the file explorer on the left displaying a folder named 'led' containing a file 'led.json'. The main editor window shows the content of 'led.json' as follows:

```
1 {  
2   "name": {  
3     "en": "LED control",  
4     "th": "ควบคุม LED"  
5   },  
6   "color": 160  
7 }
```

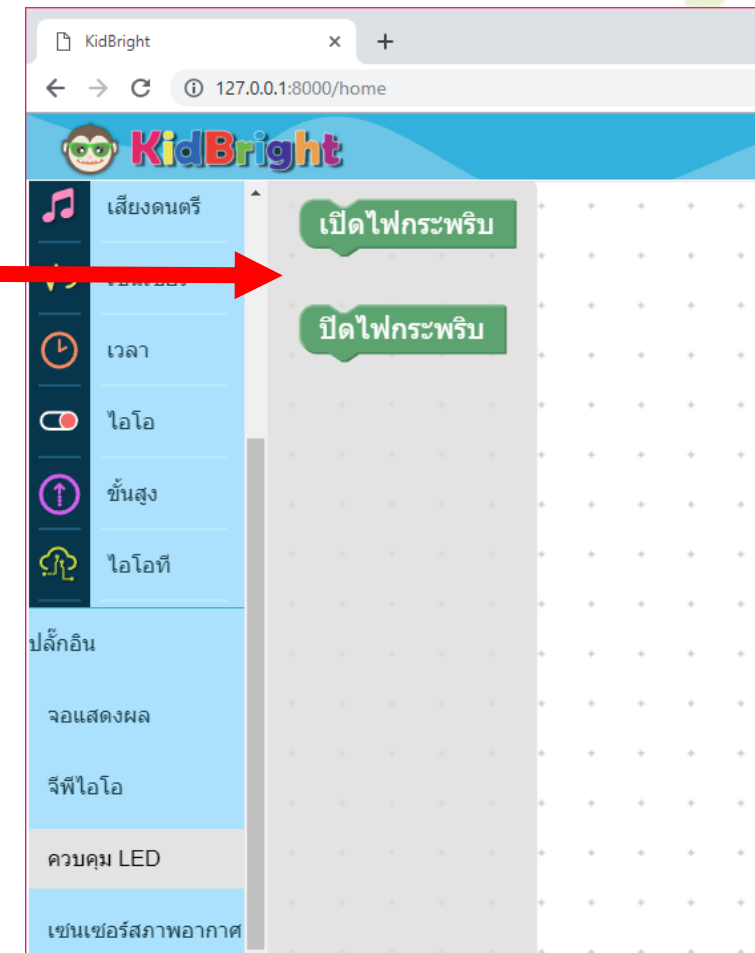


Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (ต่อ 1.)

- 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

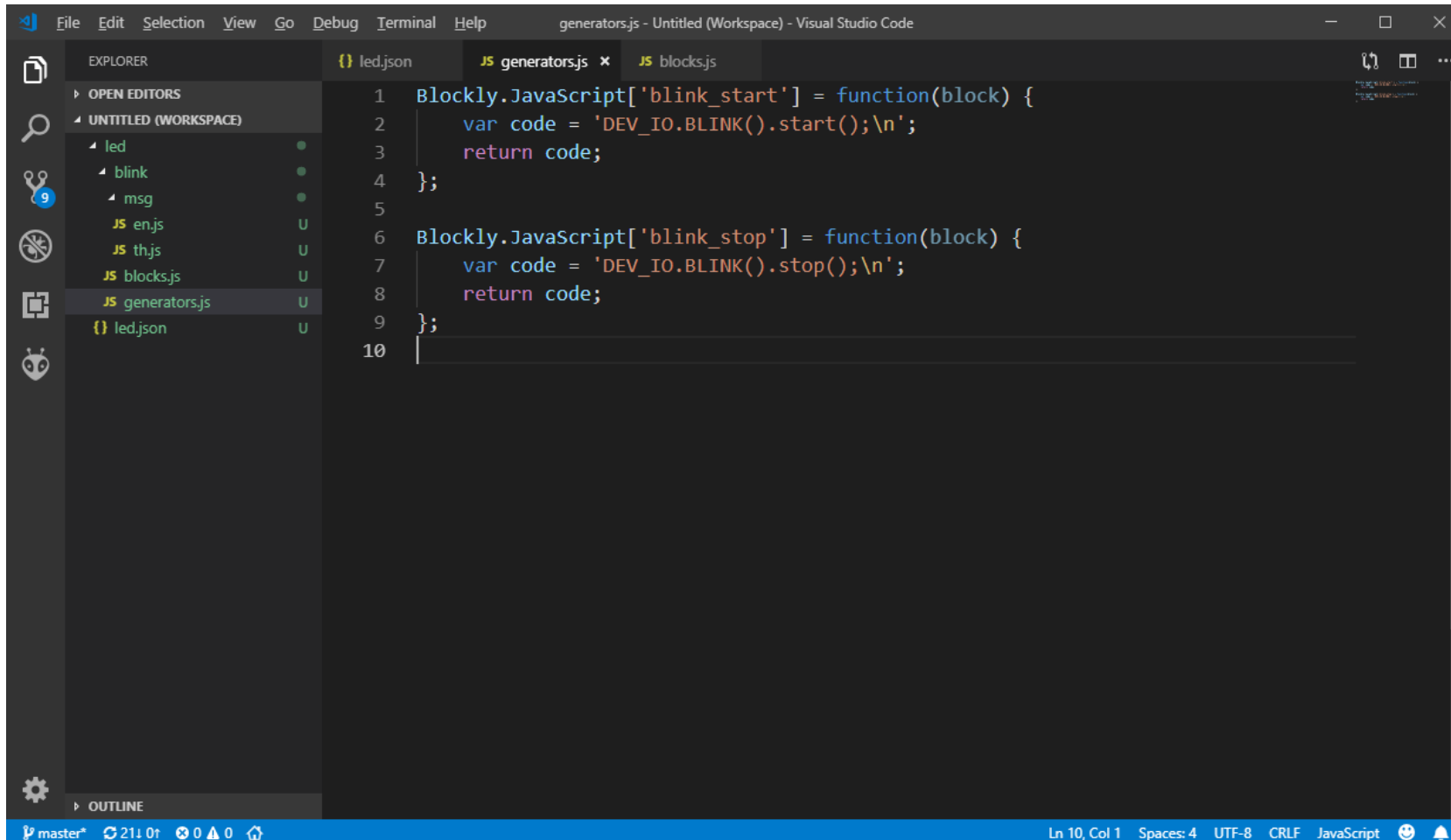


```
1 Blockly.Blocks['blink_start'] = {
2   init: function() {
3     this.appendDummyInput()
4       .appendField("เปิดไฟกระพริบ");
5     this.setPreviousStatement(true, null);
6     this.setNextStatement(true, null);
7     this.setColour(135);
8     this.setTooltip("เริ่มแสดงไฟกระพริบ");
9     this.setHelpUrl("");
10  };
11
12
13 Blockly.Blocks['blink_stop'] = {
14   init: function() {
15     this.appendDummyInput()
16       .appendField("เปิดไฟกระพริบ");
17     this.setPreviousStatement(true, null);
18     this.setNextStatement(true, null);
19     this.setColour(135);
20     this.setTooltip("เริ่มแสดงไฟกระพริบ");
21     this.setHelpUrl("");
22  };
23 }
```



Lab 2 : ทำปพลิเคชัน ไฟกระพริบอย่างง่าย (ต่อ 2.)

- ทำการสร้างไฟล์ generators.js ขึ้นมาใน folder “blink” และใส่โค้ดในไฟล์ดังนี้



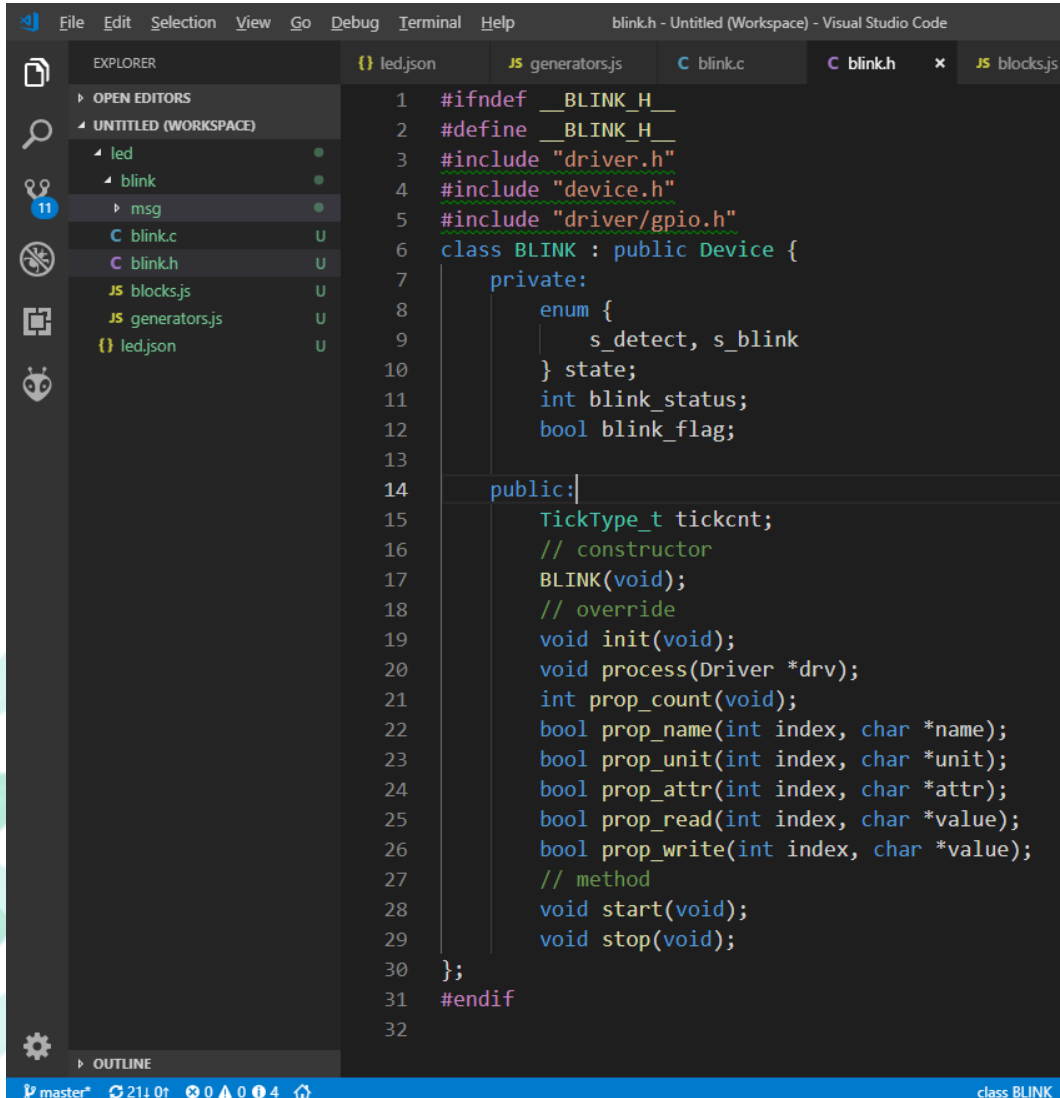
The screenshot shows the Visual Studio Code interface with the following details:

- Explorer Panel:** Shows a project structure with a folder named "led" containing a sub-folder "blink". Inside "blink", there are files "en.js", "th.js", "blocks.js", and "generators.js". The "generators.js" file is selected and highlighted.
- Editor Panel:** Displays the content of "generators.js" with the following 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
```
- Status Bar:** At the bottom, it shows "Ln 10, Col 1", "Spaces: 4", "UTF-8", "CRLF", and "JavaScript".

Lab 2 : ทำปพลิเคชัน ไฟกระพริบอย่างง่าย (ต่อ 3.)

- ทำการสร้างไฟล์ blink.h ขึ้นมาใน folder “blink” และใส่โค้ดในไฟล์ดังนี้



The screenshot shows the Visual Studio Code interface with the blink.h file open. The Explorer panel on the left shows the project structure with a folder named 'blink' containing blink.c and blink.h. The main editor area displays the following C code:

```
1 #ifndef __BLINK_H__
2 #define __BLINK_H__
3 #include "driver.h"
4 #include "device.h"
5 #include "driver/gpio.h"
6 class BLINK : public Device {
7     private:
8         enum {
9             s_detect, s_blink
10        } state;
11        int blink_status;
12        bool blink_flag;
13
14     public:
15        TickType_t tickcnt;
16        // constructor
17        BLINK(void);
18        // override
19        void init(void);
20        void process(Driver *drv);
21        int prop_count(void);
22        bool prop_name(int index, char *name);
23        bool prop_unit(int index, char *unit);
24        bool prop_attr(int index, char *attr);
25        bool prop_read(int index, char *value);
26        bool prop_write(int index, char *value);
27        // method
28        void start(void);
29        void stop(void);
30    };
31 #endif
32
```

- สร้างคลาส BLINK สืบทอดจากคลาส Device
- ตัวแปรแบบ private
 - state ใช้ในเปิด-รีด process ซึ่งทำงานแบบ state machine
 - blink_status เก็บสถานะ LED
 - blink_flag กำหนดสถานะให้กระพริบ หรือหยุดกระพริบ
- ตัวแปรแบบ public
 - tickcnt ใช้เก็บค่าเวลาคูรั้งก่อน เพื่อนำมาคำนวณเวลาที่ผ่านไป ว่าได้ระยะเวลาที่ต้องการหรือยัง ด้วยฟังก์ชัน is_tickcnt_elapsed()
- เปิด-รีดแบบ public
 - init ใช้ใส่โค้ดสำหรับกำหนดค่าเริ่มต้นต่างๆ เปิด-รีดนี้จะถูกเรียกโดย Device Manager ก่อนเข้าสู่การทำงาน state machine ของเปิด-รีด process
 - process ใช้สำหรับใส่โค้ดซึ่งทำงานแบบ state machine และจะถูกเรียกให้ทำงานเป็นระยะๆ ด้วย Device Manager
 - เปิด-รีดที่ขึ้นต้นด้วย prop_ ใช้สำหรับใส่โค้ดเพื่อรองรับ Command Line Interface (CLI)
 - start ใช้เขียนโค้ดให้ LED เริ่มกระพริบ
 - stop ใช้เขียนโค้ดให้ LED หยุดกระพริบ

Lab 2 : ทำปพลิเคชัน ไฟกระพริบอย่างง่าย (ต่อ 4.)

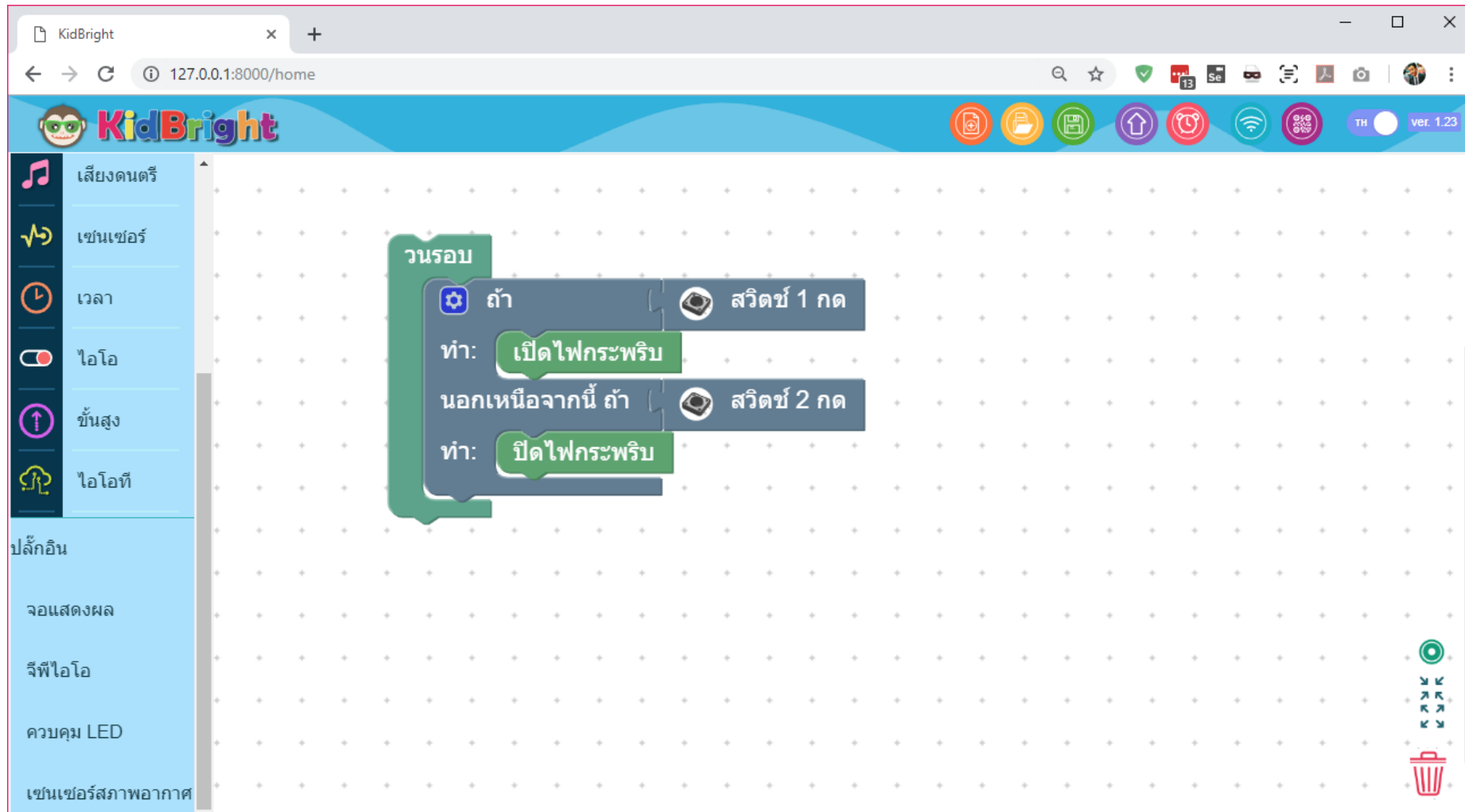
- ทำการสร้างไฟล์ blink.cpp ขึ้นมาใน folder “blink” และใส่โค้ดในไฟล์ดังนี้

```
bug Terminal Help blink.cpp - Untitled (Workspace) - Visual Studio Code
{} led.json JS generators.js blink.cpp x C blink.h JS blocks.js
1 #include "esp_system.h"
2 #include "kiddbright32.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;
```

- เมื่อดู-รีด process เป็น state machine มีการทำงานดังนี้
 - s_detect เป็น state สำหรับตรวจสอบ Device ในกรณีไม่ใช้งานให้กำหนด error เป็น false เพื่อแสดงสถานะของ Device ไม่มี error และกำหนดให้ initialized เป็น true เพื่อแสดงสถานะของ Device ว่าพร้อมใช้งาน หลังจากนั้นกำหนด state ไปที่ s_blink
 - s_blink ใน state นี้จะตรวจสอบ blink_flag ว่าต้องการให้ LED กระพริบหรือไม่ ถ้าต้องการให้กระพริบ จะคำนวณค่าเวลาที่ผ่านไปด้วยฟังก์ชัน is_tickcnt_elapsed โดยเปรียบเทียบกับค่า tickcnt ล่าสุด กับค่า 500 ms ถ้าเวลาผ่านไปได้ตามกำหนดนี้แล้ว ให้ทำการกระพริบ LED แล้วบันทึกค่า tickcnt ปัจจุบันด้วยฟังก์ชัน get_tickcnt() เพื่อใช้ในการคำนวณในรอบต่อไป
- เมื่อดู-รีด start จะสั่งให้ LED ตัด บันทึกค่า tickcnt ปัจจุบัน แล้วตั้งค่า blink_flag เป็น true เพื่อให้เมื่อดู-รีด process ทำการกระพริบ LED
- เมื่อดู-รีด stop จะตั้งค่า blink_flag เป็น false เพื่อหยุดการกระพริบ LED ในเมื่อดู-รีด process แล้วสั่งดับ LED

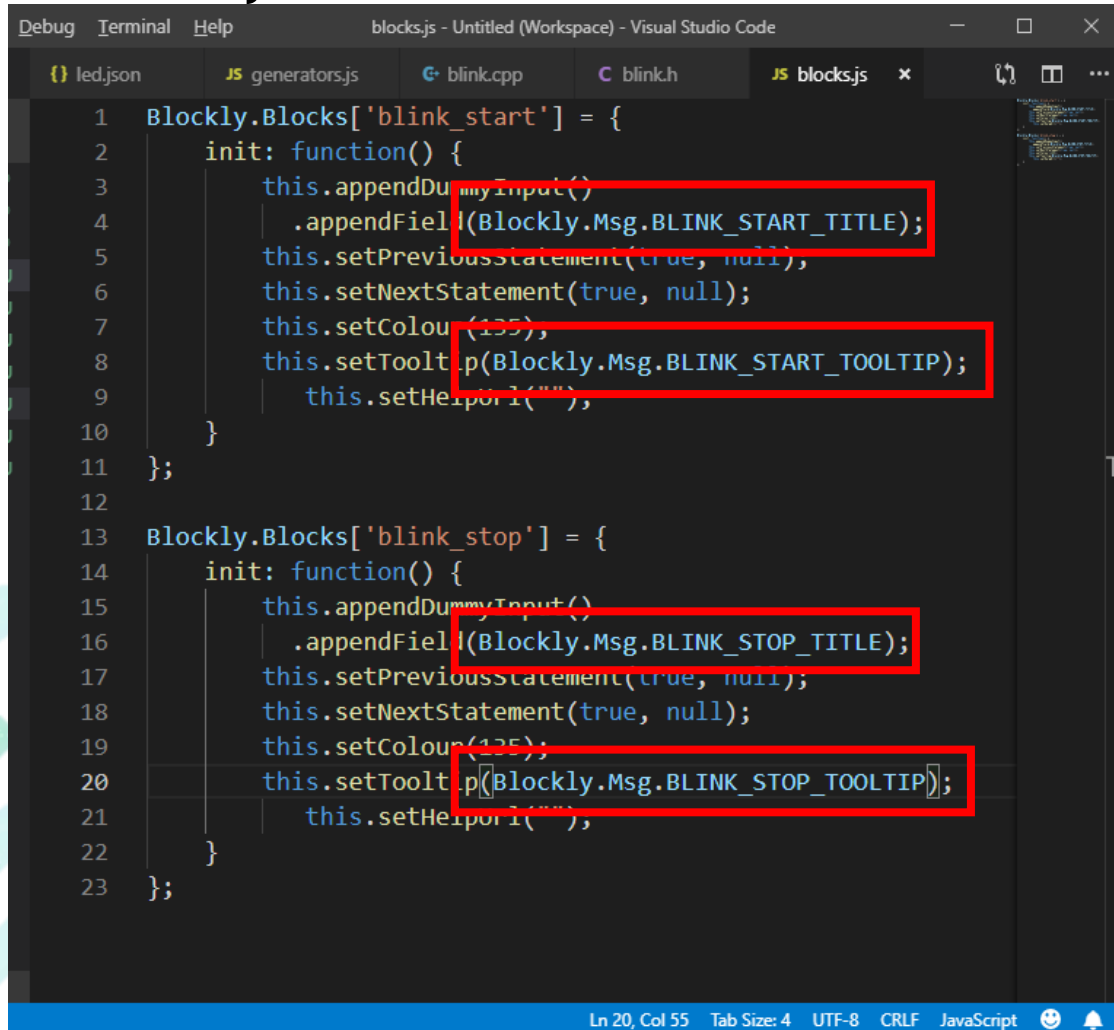
Lab 2 : ทำปลั๊กอิน ไฟกระพริบอย่างง่าย (ต่อ 5.)

- ทำการทดสอบปลั๊กอินที่สร้างขึ้นใหม่ หากทำถูกต้องไฟ LED BT จะกระพริบทุก 500ms

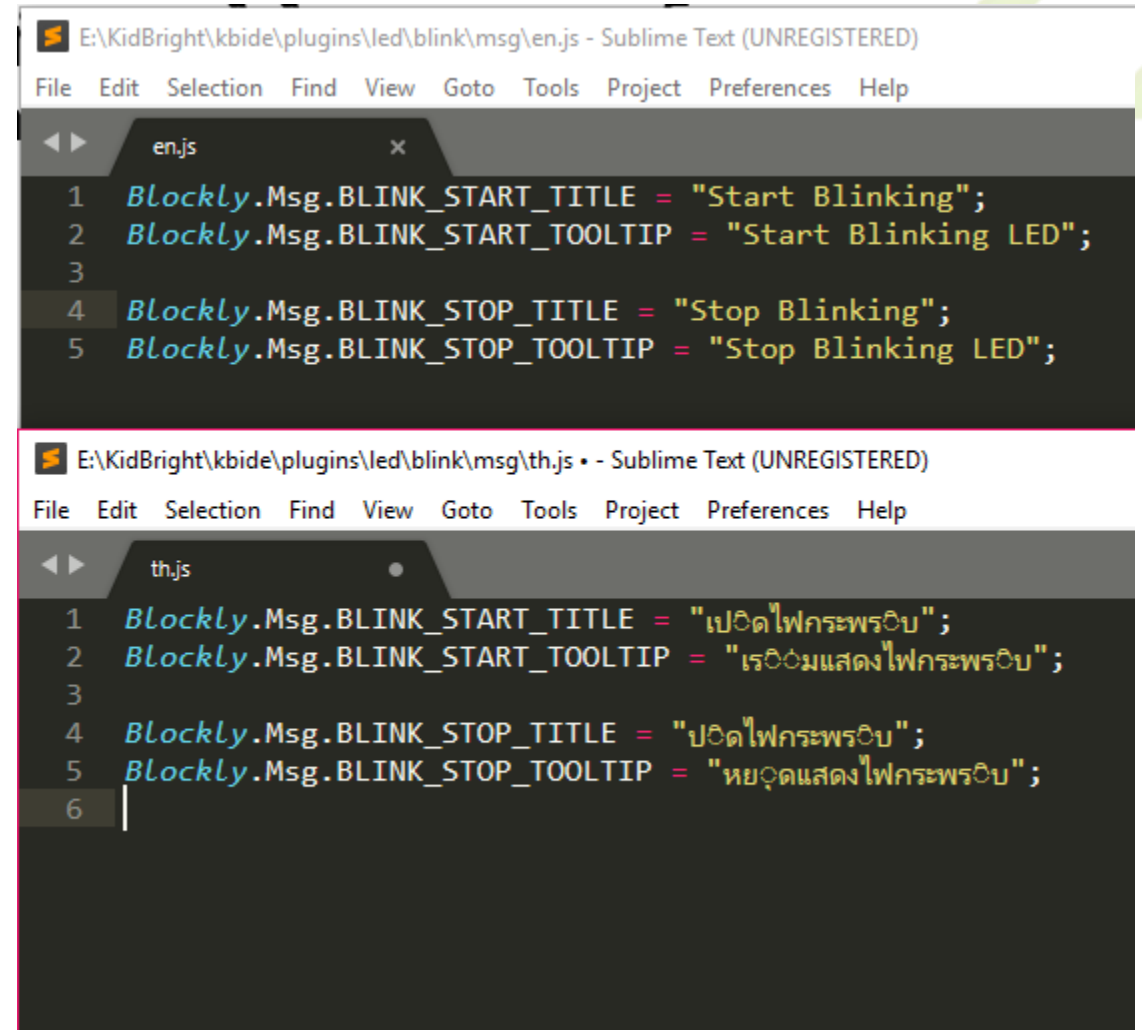


Lab 2 : ทำให้ Block รองรับภาษาไทยและอังกฤษ

- ให้ทำการแก้ไขไฟล์ blocks.js โดยย้ายข้อความภาษาไทยที่ฝังไว้ ใสไว้ใน th.js และ แปลเป็นภาษาอังกฤษ ในไฟล์ en.js โดยตั้งชื่อตัวแปรให้สัมพันธ์กัน



```
1 Blockly.Blocks['blink_start'] = {
2   init: function() {
3     this.appendDummyInput()
4     .appendField(Blockly.Msg.BLINK_START_TITLE);
5     this.setPreviousStatement(true, null);
6     this.setNextStatement(true, null);
7     this.setColour(135);
8     this.setTooltip(Blockly.Msg.BLINK_START_TOOLTIP);
9     this.setHelpUrl("");
10  }
11 };
12
13 Blockly.Blocks['blink_stop'] = {
14   init: function() {
15     this.appendDummyInput()
16     .appendField(Blockly.Msg.BLINK_STOP_TITLE);
17     this.setPreviousStatement(true, null);
18     this.setNextStatement(true, null);
19     this.setColour(135);
20     this.setTooltip(Blockly.Msg.BLINK_STOP_TOOLTIP);
21     this.setHelpUrl("");
22  }
23 };
```



```
E:\KidBright\kbide\plugins\led\blink\msg\en.js - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

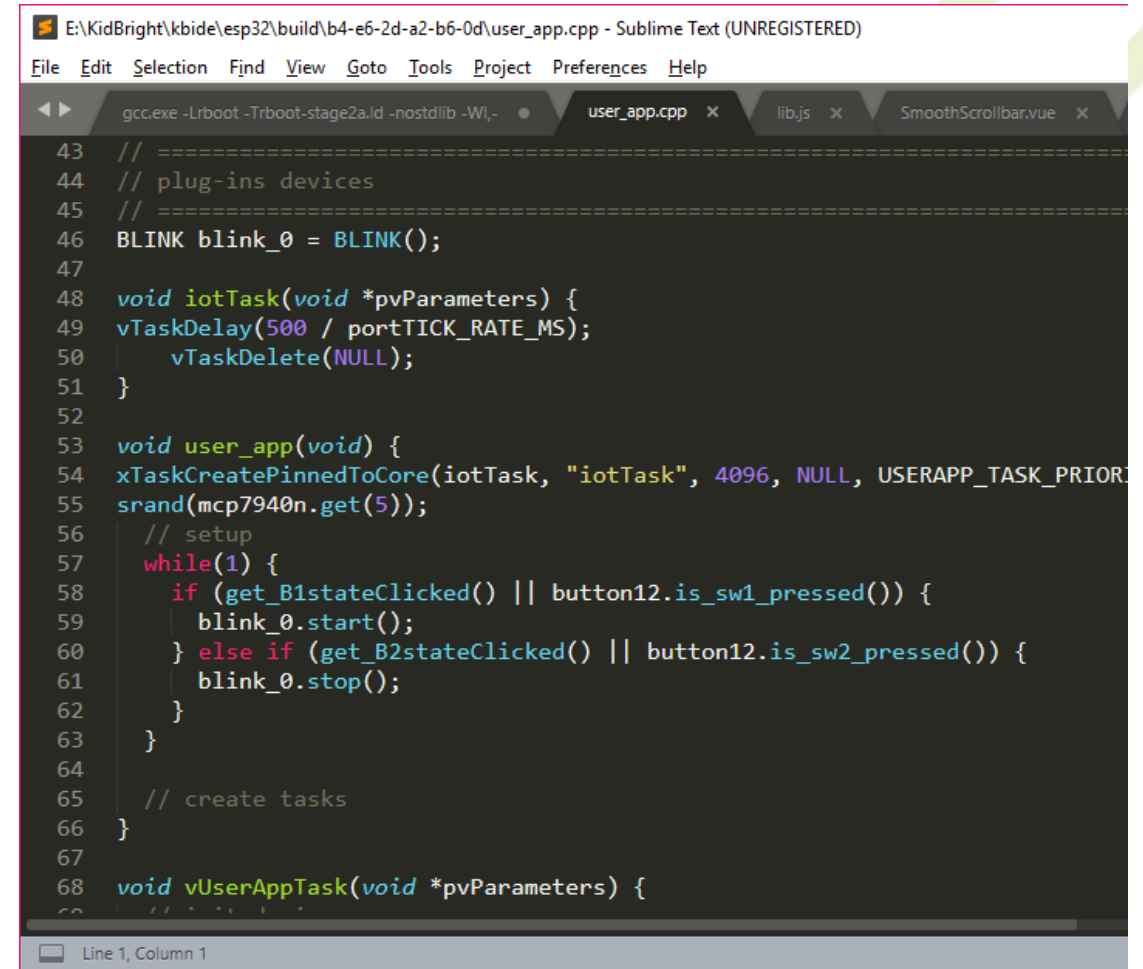
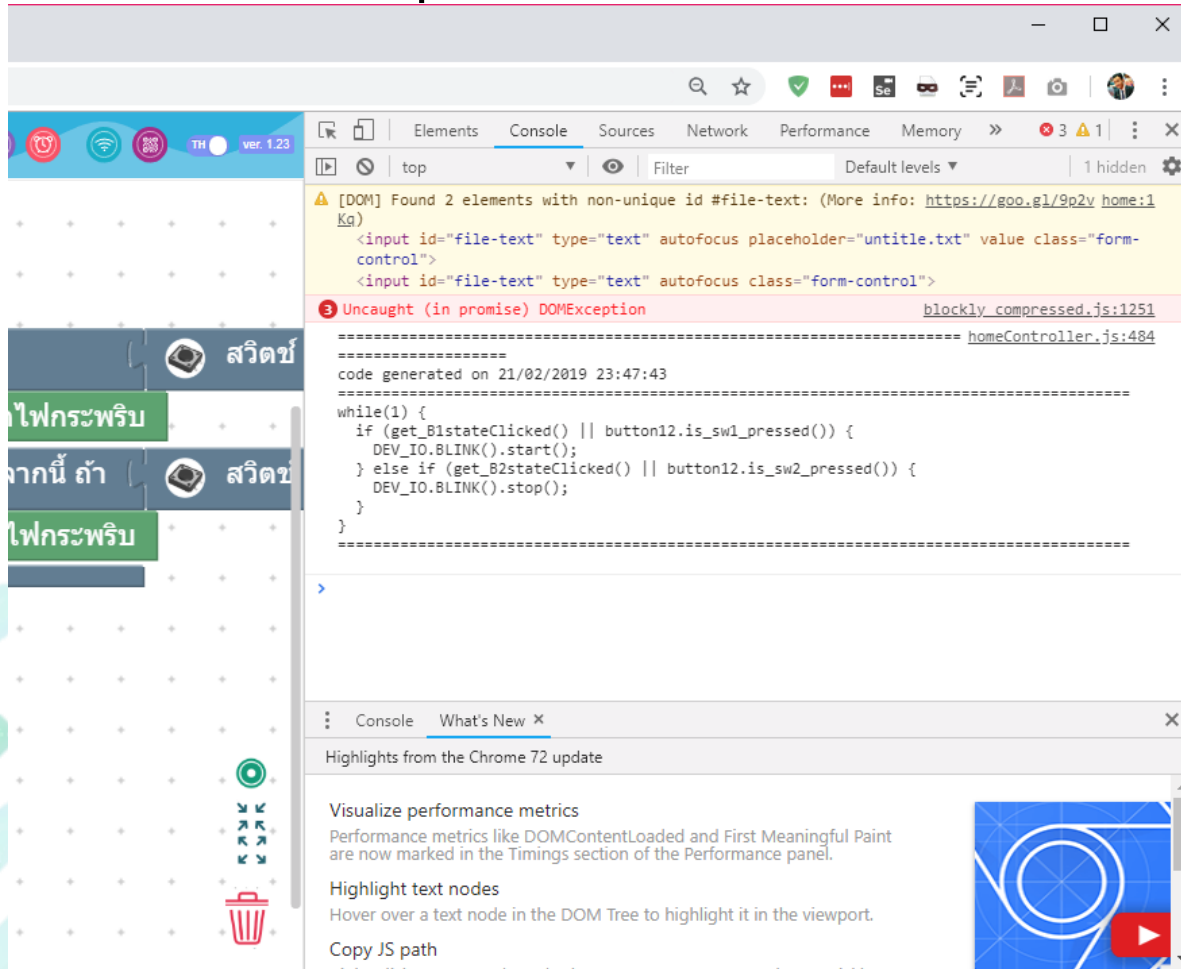
en.js
1 Blockly.Msg.BLINK_START_TITLE = "Start Blinking";
2 Blockly.Msg.BLINK_START_TOOLTIP = "Start Blinking LED";
3
4 Blockly.Msg.BLINK_STOP_TITLE = "Stop Blinking";
5 Blockly.Msg.BLINK_STOP_TOOLTIP = "Stop Blinking LED";

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 : อธิบายการ Compile ของโค้ด

- ให้ทำการกด F12 หรือ Ctrl + Shift + J จะเห็นโค้ดที่ Generate ออกมาจากบล็อก และถูกแปลงเป็นโค้ดที่พร้อม compile



Lab 2 : ทำปพลิเคชัน ไฟกระพริบอย่างง่าย

- โจทย์ :

- ทำการปรับเวลา การกระพริบทุก ๆ 100ms และ 1000ms
- ทำการเปลี่ยน PIN จาก BT_LED_GPIO เป็น pin LED ดวงอื่น

- (hint ใช้การตั้งชื่อตัวแปรแบบนี้) `gpio_num_t LED_BT = (gpio_num_t)17;`

- ทำไฟวิ่งจาก LED ทั้ง 4 ดวง

- (hint การใช้ GPIO 2 PIN) `io_conf.pin_bit_mask = (1ULL << LED_IOT) | (1ULL << LED_XX);`



Lab 2 :ทำปพลิเคชัน ไฟกระพริบอย่างง่าย

ไฟกระพริบอย่างง่าย !?
ง่ายกว่านี้มีอีกไหม ... ง่ายแบบ ctrl+c ctrl+v

Lab 3 : Copy & Paste Block

- เป้าหมาย : สร้าง Block เปิดการใช้งานและควบคุม GPIO โดยออกแบบใหม่
- วิธีการทำ : ทำการคัดลอก “kbgpio” จาก “lab_direction” ใส่ใน folder “led” ใน plugins
- โจทย์ :
 - ข้อ 1 ทำการออกแบบบล็อก ของ function `setOutput(int pin)` และ `digitalWrite(int pin,int value)` ใส่ในไฟล์ `blocks.js` และเรียกใช้ function ทั้งสองผ่านไฟล์ `generators.js`
 - ข้อ 2 เขียนโปรแกรมไฟวิ่งบน Block ทดสอบเรียกใช้ block ที่สร้างขึ้น
 - ข้อ 3 ทดสอบควบคุม Relay ด้วย PIN อื่นๆ

LAB 3 : ออกแบบ GPIO ใหม่

```
#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
```

- โยบการะการทำงานขอโปรแกรมไปให้ผู้ใช้ เราออกแบบแค่ function พื้นฐาน
- ตัดการ override state อะไรต่าง ๆ ออก ทำให้โค้ดดูเข้าใจง่าย
- ใช้ concept ที่เคยรู้จัก (arduino) เปิด pinMode ก่อนใช้งาน และ ใช้คำสั่ง digitalWrite ในการควบคุม

LAB 3 : ออกแบบ GPIO ใหม่

```
#include "esp_system.h"
#include "kiddbright32.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 โดยคัดลอกมาจากปลั๊กอินก่อนหน้านี้
- ไม่มี state machine แล้ว
- ไม่มี override แล้ว (เขียนไปไฟล์ header หมดแล้ว)
- คำสั่ง digitalWrite ตรงตัว

LAB 3 : ตัวช่วยในการสร้างและออกแบบบล็อก

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

The screenshot displays the Blockly Developer Tools interface in a web browser. The browser's address bar shows the URL <https://blockly-demo.appspot.com/static/demos/blockfactory/index.html>. The page title is "Blockly Demo: Blockly Developer". The interface includes a navigation bar with "Block Factory", "Block Exporter", and "Workspace Factory". Below this is a "Block Library" section with buttons for "Save 'blink_start'", "Delete 'blink_start'", "Preview", "Clear Library", "Import Block Library", and "Download Block Library". The main workspace shows a block definition for "blink_start". The block has a "name" of "blink_start", an "inputs" section with a "dummy input" and a "text" field containing "เปิดไฟกระพริบ", and an "automatic" section with "inputs" and "top+bottom connections". The block's "toolTip" is "เริ่มแสดงไฟกระพริบ", and its "colour" is "hue: 135°". The "Block Definition" section shows the JavaScript code for the block, and the "Generator stub" section shows the JavaScript code for the block's generator.

Blockly Demo: Blockly Developer

[Blockly](#) > [Demos](#) > Blockly Developer Tools

Block Factory Block Exporter Workspace Factory

Block Library Save "blink_start" Delete "blink_start" Preview: LTR Clear Library Import Block Library Download Block Library

Input Field Type Colour

name blink_start

inputs dummy input fields left text เปิดไฟกระพริบ

automatic inputs top+bottom connections

toolTip “เริ่มแสดงไฟกระพริบ”

help url “ ”

bottom type any

top type any

colour hue: 135°

Block Definition: JavaScript

```
Blockly.Blocks['blink_start'] = {
  init: function() {
    this.appendDummyInput()
      .appendField("เปิดไฟกระพริบ");
    this.setPreviousStatement(true, null);
    this.setNextStatement(true, null);
    this.setColour(135);
    this.setTooltip("เริ่มแสดงไฟกระพริบ");
  }
};
```

Generator stub: JavaScript

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

ESP-IDF : การใช้งาน GPIO และ Time

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

Lab 4 : วารกัณขโมย

- เป้าหมาย : ใช้งาน Digital Input และ Timer ผ่าน ESP-IDF
- โมดูล : 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” “โมดูลวัดระยะทาง” ใน IDE
 - ทำการคัดลอก ปลั๊กอิน kbgpio ใน Lab direction มาสร้างปลั๊กอินใหม่ โดยแก้ไข kbgpio ทั้งหมดในทุกไฟล์เป็น “sr04”
 - ทำการออกแบบปลั๊กอินทั้งหมด 1 block ให้มีลักษณะดังนี้

อ่านค่า sr04 ขา trig 26 ขา Echo 32



Lab 4 : วารกัณขมอย (ต่อ)

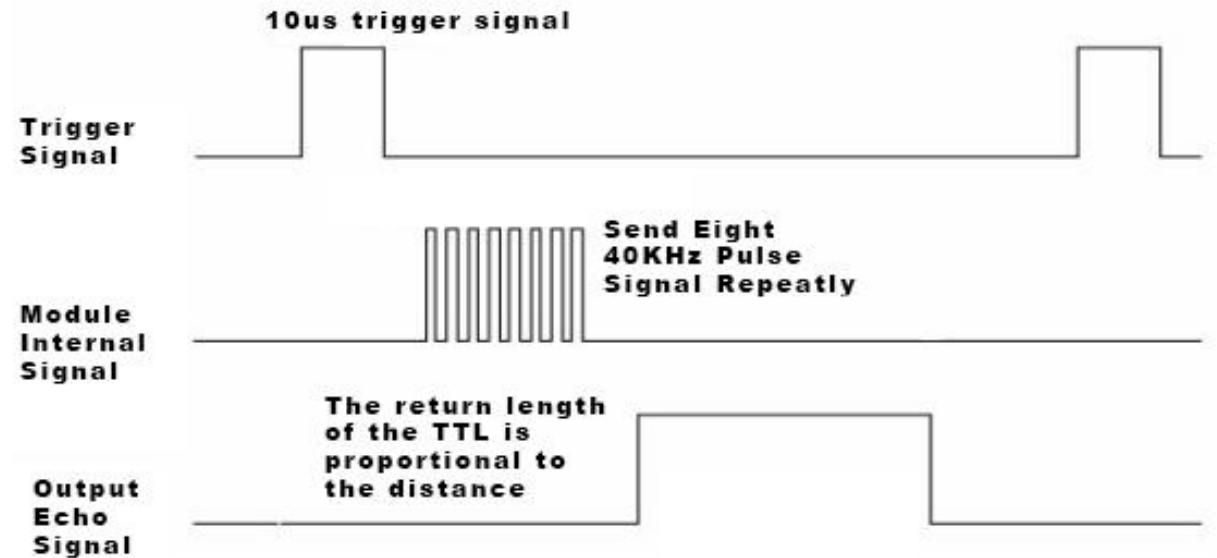
- ทำการแก้ไขไฟล์ sr04.h โดยสร้างตัวแปร private กำหนด constructor และใส่ include ไฟล์

```
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 : วารกัณขมอย (ต่อ)

- ทำการแก้ไขไฟล์ sr04.cpp โดยใช้โค้ดจาก 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
12 float SR04::read()
13 {
14     gpio_set_level(this->trig, 0);
15     ets_delay_us(4);
16     gpio_set_level(this->trig, 1);
17     ets_delay_us(10);
18     gpio_set_level(this->trig, 0);
19
20     if (gpio_get_level(this->echo)){ return -1; } //// Previous ping isn
21
22     uint32_t start = esp_timer_get_time();
23     while (!gpio_get_level(this->echo)) // Wait for echo
24     {
25         if(esp_timer_get_time() - start > 100000ULL){ return -1; }
26     }
27     uint32_t echo_start = esp_timer_get_time();
28     uint32_t time = echo_start;
29     while (gpio_get_level(this->echo)) // got echo, measuring
30     {
31         time = esp_timer_get_time();
32         if (time - echo_start > 100000ULL){ return -1; }
33     }
34     return (time - echo_start) / 58.0; //58 = ROUND TRIP
35 }
36
```



- Timing การส่งการ Ultrasonic SR04

Lab 4 : ตรวจจับขโมย (โจทย์)

- โจทย์ :

- ทำการเขียนโปรแกรมแบบบล็อกเพื่อทดสอบบล็อกที่ได้
- ทำการเขียนโค้ดเพื่อให้ Sensor ทำงานตอนกลางคืนหลัง 8.00PM – 6.00AM ถ้าพบการเคลื่อนไหวให้ส่งเสียงขึ้นมาด้วย Buzzer



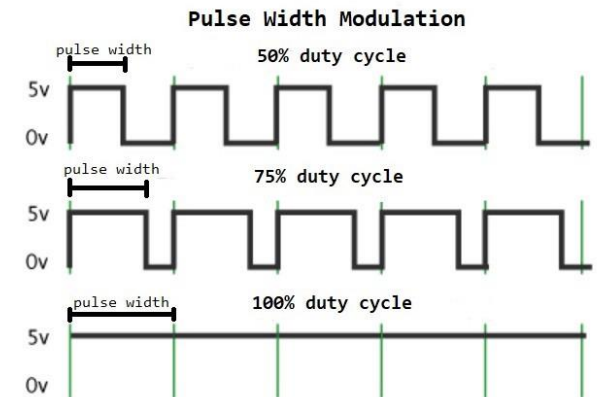
การใช้งาน AnalogWrite (PWM) ใน KidBright

- การใช้งาน PWM นั้นจะมี Driver เฉพาะบน 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);
```

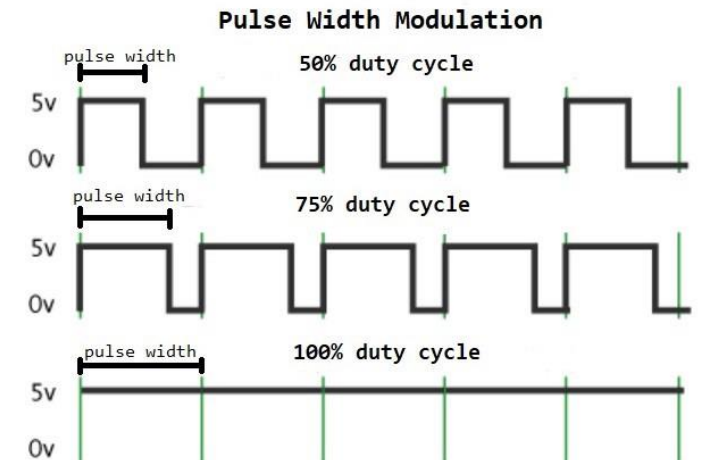
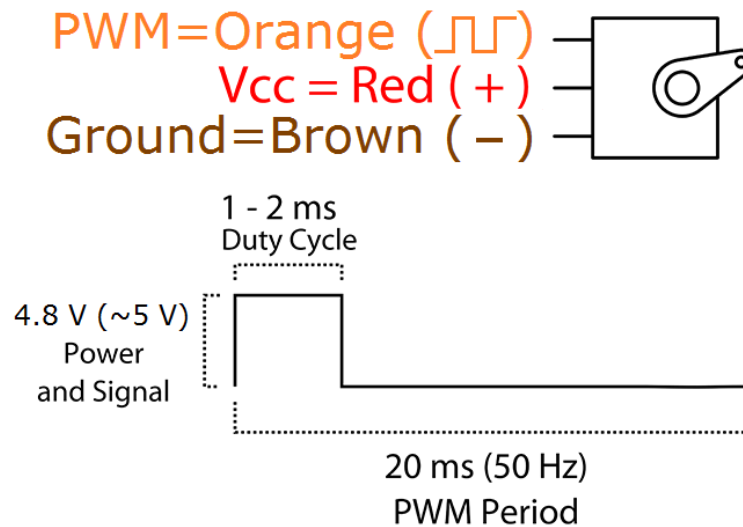


- เรียกใช้งานผ่านการปรับ duty width

```
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 แคลประมาณ 10-20% เท่านั้น (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 : ทักษะอัจฉริยะ

- 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 เข้าสายไฟสีน้ำตาล ไฟ 5v สายสีแดง และต่อ PIN 18 กับสายไฟสีส้ม
 - ทำการสร้างหมวดหมู่ปลั๊กอินใหม่ folder “motor” โดยแสดงคำว่า “Motor” “มอเตอร์” ใน IDE
- ใจกย :
 - ทำการสร้างบล็อกจาก Class ที่กำหนดไว้ใน Lab Direction
 - ทำการเขียนโปรแกรมด้วย Block เพื่อทดสอบการทำงานของ Servo Motor
 - ทำการเขียนโปรแกรมด้วย Block สั่งการ Servo Motor จาก Sonar Module



END of DAY 1