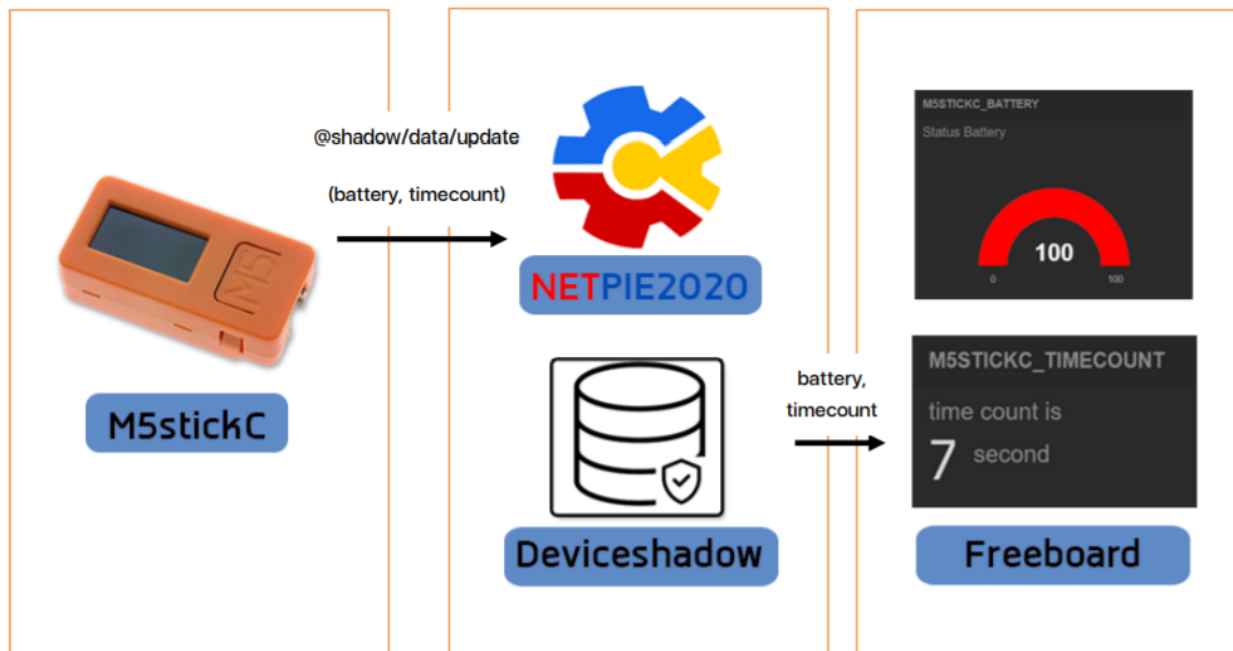


## DIY Wearable

### Project Description

This example demonstrates the DIY wearable using M5StickC, powered by ESP32 with 240MHz dual-core, 4MB flash memory, built-in WiFi, and Bluetooth capabilities. It is a portable, easy-to-use, open-source, IoT development board with features of a built-in 6-axis IMU, built-in Lipo battery, microphone, LCD (0.96 inches), and Real-Time Clock (RTC). A DIY wearable sends the battery and timer values to the NETPIE2020 and presents them on the Freeboard.

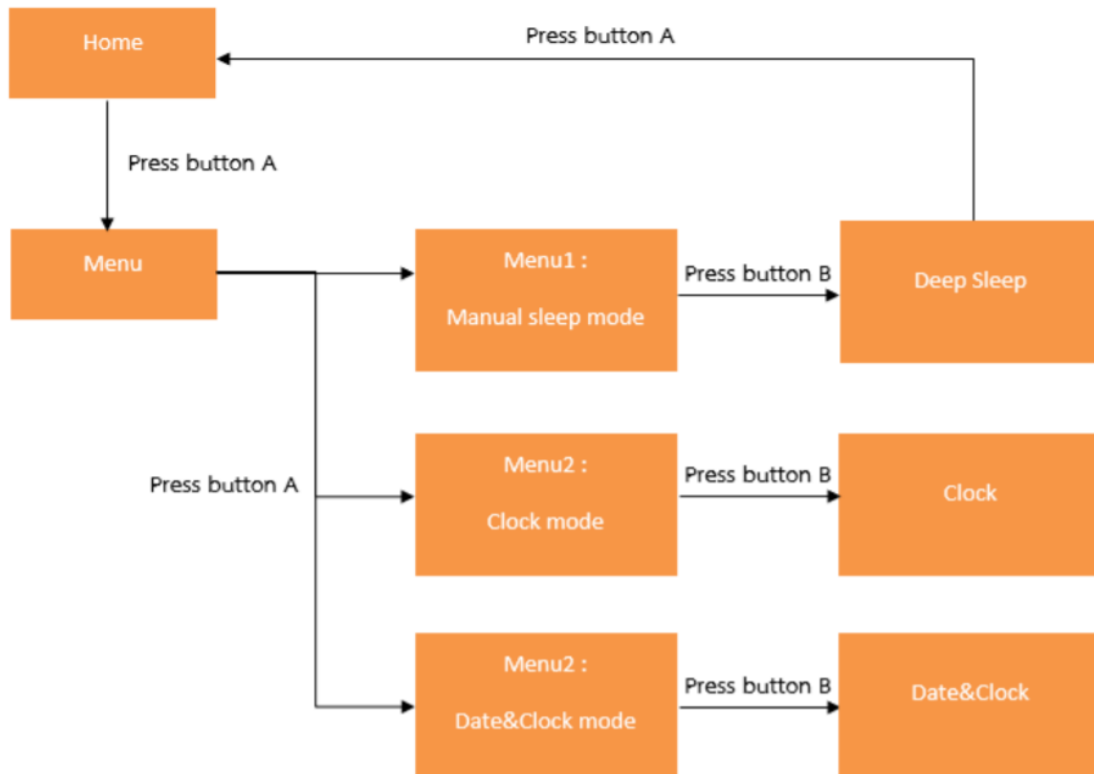


### Working model

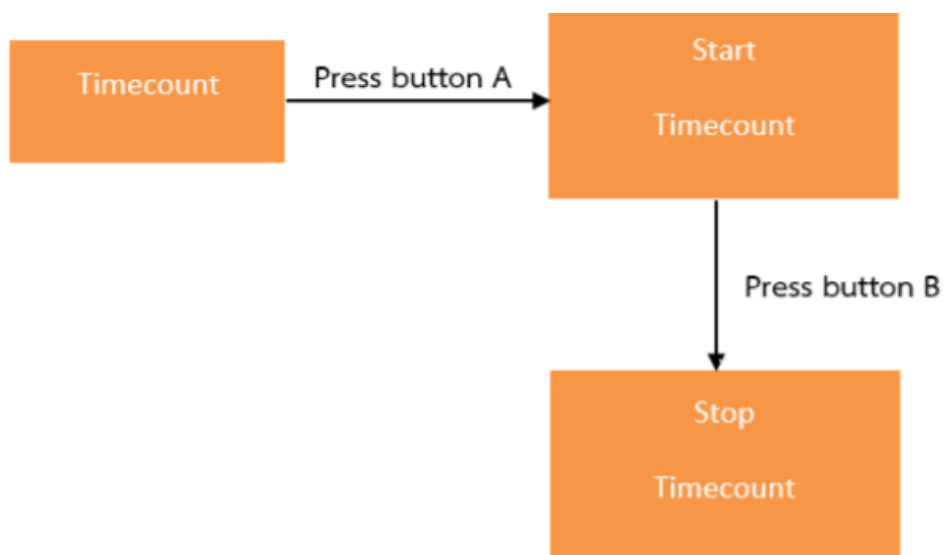
### Require Basic Knowledge of

- 1.Using NETPIE2020
- 2.Using NETPIE Freeboard
- 3.Using M5StickC

### A simple diagram of state transitions



### State transitions of various menus



### State transitions of timecount

## Equipment Used

### 1.M5StickC

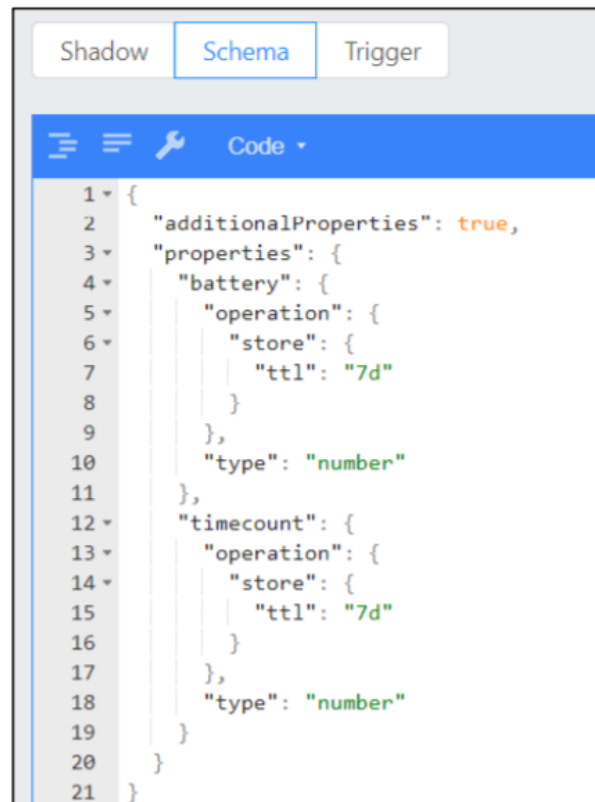
M5StickC is a compact M5Stack powered by the ESP32 chip, it is a portable, user-friendly, and easy-to-use IoT development board with 0.96 inch TFT LCD display of resolution 80 X 160 pixels, assembled in a well-designed plastic case (orange). It also comes with a 6-Axis IMU, IR transmitter, 3D antenna, USB-C port, LED, microphone, and buttons.



**Features of M5StickC**

## Defining Device Schema

The first part is to define the device schema, which is the data structure for the devices generating data. The server checks the data structure defined in the device schema and performs the required actions before storing the data in the Timeseries database. These actions include converting the data units and data validation.



```
1 {  
2   "additionalProperties": true,  
3   "properties": {  
4     "battery": {  
5       "operation": {  
6         "store": {  
7           "ttl": "7d"  
8         }  
9       },  
10      "type": "number"  
11    },  
12    "timecount": {  
13      "operation": {  
14        "store": {  
15          "ttl": "7d"  
16        }  
17      },  
18      "type": "number"  
19    }  
20  }  
21 }
```

## Description of Schema

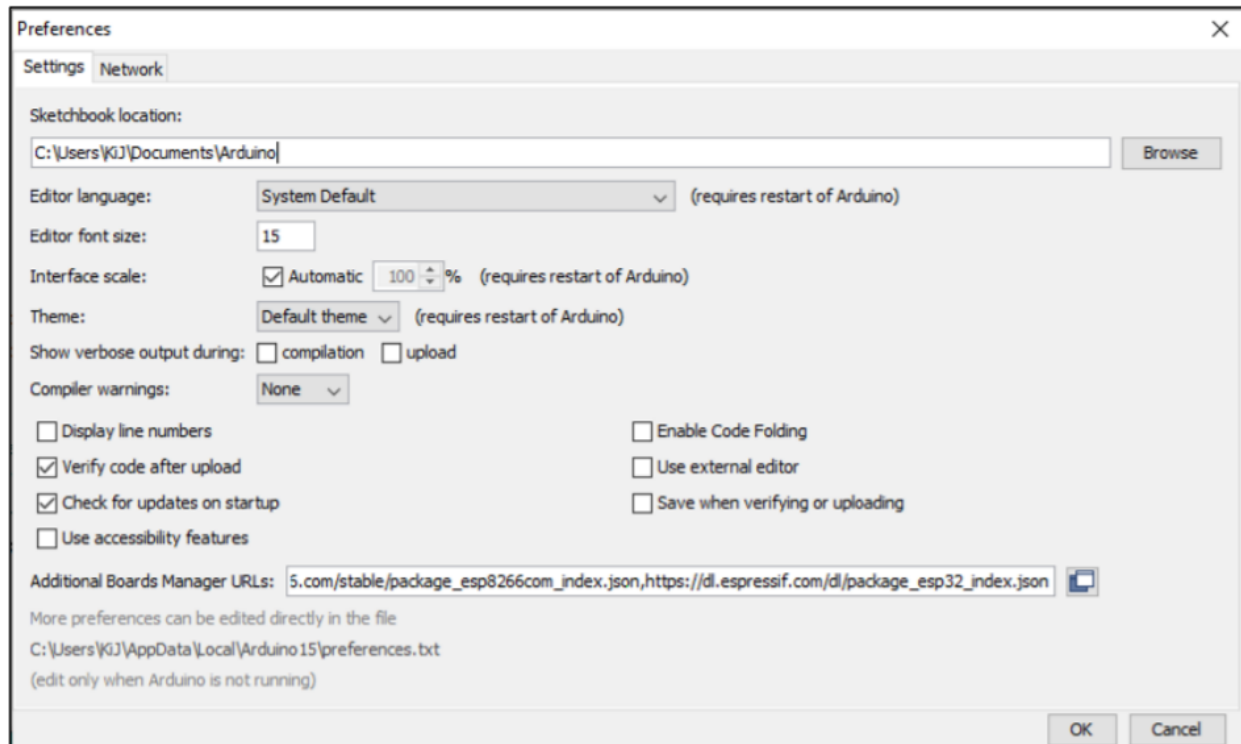
Stores the first variable 'battery' of type number in the Timeseries database with a retention period of 7 days.

Store the second variable 'timecount' of type number in the Timeseries database with a retention period of 7 days.

## Installing M5Stack in Arduino IDE

1. Install ESP32 Boards Manager.

Open Arduino IDE and go to File -> Preferences.

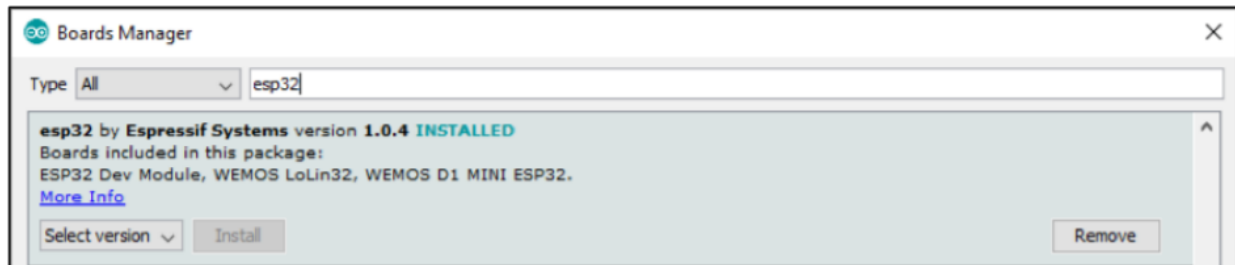


Copy the ESP32 Boards Manager URL into the Additional Boards Manager URLs, and press 'OK'.

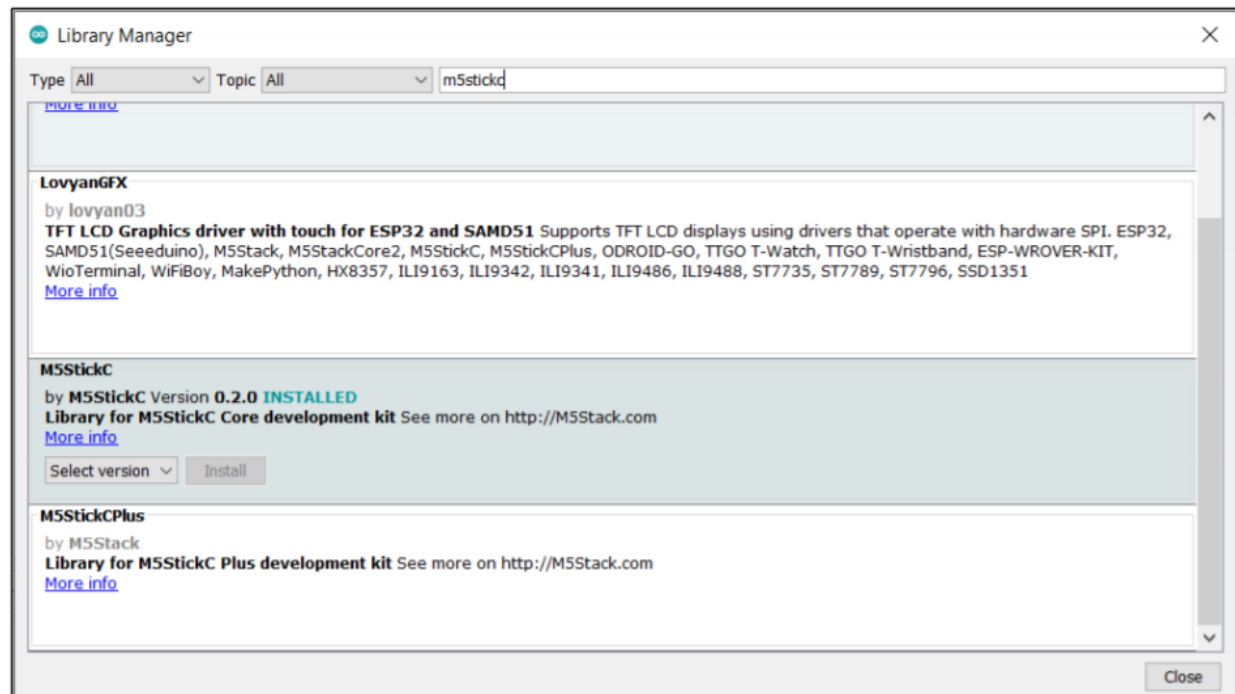
Manager URL: [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)



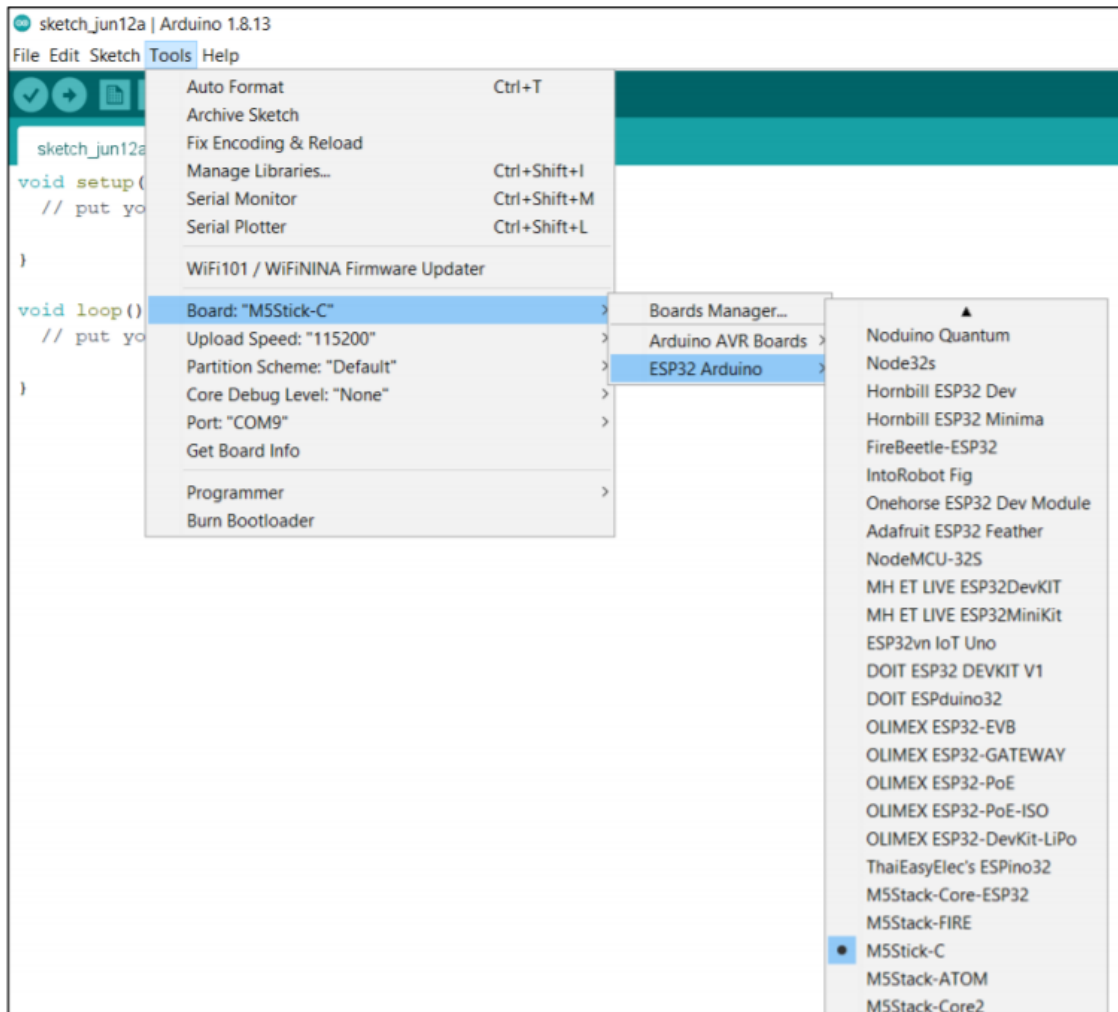
Next, go to Tools -> Boards -> Boards Manager.  
Search for esp32 and press 'Install'



2. Install the M5StickC library.  
Open Arduino IDE and go to Sketch -> Include Library -> Manage Library.  
Search for M5StickC and press 'Install'



3. Select Boards, for uploading code to the M5StickC.  
Tools -> Boards -> ESP32 -> M5StickC



## Description of Arduino Program

### **Transmitting battery values to the NETPIE2020**

The first part is to import the required libraries. Various functions of the M5StickC are added in the program that includes a time zone display. This example uses TIMEZONE 7, which means UTC+7 Bangkok, Thailand. Next, the network parameters like WiFi, Client ID, Device, Token, and Secret are configured for connecting to the NETPIE2020.

```

#include <M5StickC.h>
#include <Wire.h>
#include "time.h"
#include <WiFi.h>
#include <PubSubClient.h>

// Set RTC time
RTC_DateTypeDef DateStruct;
RTC_TimeTypeDef TimeStruct;
RTC_TimeTypeDef RTC_TimeStruct;
RTC_DateTypeDef RTC_DateStruct;

/* Custom setting area */
/* TIMEZONE setting */
#define TIMEZONE 7
/* TIMEZONE setting */

/* wifi or Smartphone tethering ssid and pass */
const char* ssid = "pp";
const char* password = "12345687";
const char* ntpServer = "ntp.jst.mfeed.ad.jp";
/* Custom setting area */
const char* mqtt_server = "broker.netpie.io";
const int mqtt_port = 1883;
const char* mqtt_Client = "13a6229d-c5da-4d17-b57c-9d0d6859165a";
const char* mqtt_username = "qqgUkEpHF3egu89VyLGJU2DuUQV7thDC";
const char* mqtt_password = "65M(8MpL0aUj1LDB(U1~jUccIl6lTPC9";

```

The next section defines the 'tm' as a structured datatype defined in the time.h library. It allows holding the different elements of the data separately, thereby allowing us to access certain parts of time such as day, month, and year.

```

struct tm timeInfo;

void time_m5(void){
    configTime(TIMEZONE * 3600, 0, ntpServer);
    getLocalTime(&timeInfo);
    TimeStruct.Hours = timeInfo.tm_hour;
    TimeStruct.Minutes = timeInfo.tm_min;
    TimeStruct.Seconds = timeInfo.tm_sec;
    M5.Rtc.SetTime(&TimeStruct);

    DateStruct.WeekDay = timeInfo.tm_wday;
    DateStruct.Month = timeInfo.tm_mon + 1;
    DateStruct.Date = timeInfo.tm_mday;
    DateStruct.Year = timeInfo.tm_year + 1900;
    M5.Rtc.SetData(&DateStruct);
}

```

The next part is to run the MQTT commands.



```
char msg[100];  
WiFiClient espClient;  
PubSubClient client(espClient);
```

The MQTT connect function is used to connect to the MQTT server. If the connection is successful, it will display the message saying 'connected'. But, if the connection is unsuccessful, a 'failed' message is displayed and will try to reconnect automatically.

```
void reconnect() {  
  while (!client.connected()) {  
    if (client.connect(mqtt_Client, mqtt_username, mqtt_password)) {  
      Serial.println("Connected NETPIE");  
    }  
  
    else {  
      Serial.print("failed, rc=");  
      Serial.print(client.state());  
      Serial.println("try again in 5 seconds");  
      delay(5000);  
    }  
  }  
}
```

Function for WiFi connection

```
void wifi_get(void){  
  WiFi.begin(ssid, password);  
  while (WiFi.status() != WL_CONNECTED) {  
    delay(500);  
    Serial.print(".");  
  }  
}
```

In the setup function, two pinModes are defined, which are M5\_BUTTON\_HOME and M5\_BUTTON\_RST to get input, M5.begin() is defined to initialize the function calls, the wifi\_get() is defined to connect to the WiFi, setRotation() is defined to select the screen rotation, and timem5() is defined to configure time zone, seconds, minutes, hours, day, month, and year.

```

void setup() {
    sleep_wake();    //bright up
    pinMode(M5_BUTTON_HOME, INPUT);    // button A = 37
    pinMode(M5_BUTTON_RST, INPUT);    // button B = 39
    M5.begin();
    wifi_get();
    client.setServer(mqtt_server, mqtt_port);
    M5.Lcd.setRotation(3);
    M5.Lcd.fillScreen(BLACK);
    M5.Axp.ScreenBreath(8);
    setCpuFrequencyMhz(21);
    time_m5();
    //WiFi.disconnect(true);
    M5.Lcd.setCursor(0, 35, 4);
    M5.Lcd.printf("PRESS M5!!");
}

```

In the loop function, a condition is defined, when M5\_BUTTON\_HOME is clicked all the three menus will be displayed as follows:

```

char menu1[]="1:Manual sleep mode";
const char menu2[]="2:Clock Mode";
const char menu3[]="3:Date&Clock Mode";

```

```

void loop(){
    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    if(digitalRead(M5_BUTTON_HOME) == LOW){
        clear_txt();
        time_disp_flg    = FALSE;
        timedate_disp_flg = FALSE;
        deep_sleep_flg    = FALSE;
        menu_cnt++;
        if(menu_cnt > NOMAL_DATE_CLOCK){
            menu_cnt = DEEP_SLEEP;
        }
        else{
            ;
        }
    }
}

```

```

switch (menu_cnt) {
case DEEP_SLEEP:
    for(int i=30;i>0;i--){
        M5.Lcd.setCursor(i, 30, 1);
        M5.Lcd.printf(&menu1[0]);
        delay(20);
    }
    delay(DISP_DELAY);
    break;
case NOMAL_CLOCK:
    for(int i=30;i>0;i--){
        M5.Lcd.setCursor(i, 40, 1);
        M5.Lcd.printf(&menu2[0]);
        delay(20);
    }
    delay(DISP_DELAY);
    break;
case NOMAL_DATE_CLOCK:
    for(int i=30;i>0;i--){
        M5.Lcd.setCursor(i, 50, 1);
        M5.Lcd.printf(&menu3[0]);
        delay(20);
    }
    delay(DISP_DELAY);
    break;
default:
    break;
}

```

This section defines the conditions on clicking M5\_BUTTON\_RST.

```

//----- select mode
if(digitalRead(M5_BUTTON_RST) == LOW){
    move_flg = TRUE;    //bright up
    clear_txt();

    if(menu_cnt == NOMAL_CLOCK){
        time_disp_flg = TRUE;

    }else if(menu_cnt == NOMAL_DATE_CLOCK){
        timedate_disp_flg = TRUE;

    }else if(menu_cnt == DEEP_SLEEP){
        deep_sleep_flg = TRUE;
    }
    else{
        def_disp(); // PRESS M5 !!
    }
    delay(DISP_DELAY);
}

```

```

//----- call mode

if(time_disp_flg == TRUE){
    time_disp();

}else if(timedate_disp_flg == TRUE){
    time_date_disp();

}else if(deep_sleep_flg == TRUE){
    deep_sleep();
}
//sleep_bar();
bat_disp();
tft_state();
}

```

The function `time_disp` is used to display the hours and minutes at a time.

```
void time_disp(void){
    int linedisp = 0;
    M5.Rtc.GetTime(&RTC_TimeStruct);
    M5.Lcd.setCursor(0, 12, 2);
    M5.Lcd.printf("Current time");
    M5.Lcd.setCursor(90, 40, 2);
    M5.Lcd.printf("%02d",RTC_TimeStruct.Seconds);
    M5.Lcd.setCursor(0, 33, 4);
    M5.Lcd.printf("%02d : %02d :",RTC_TimeStruct.Hours, RTC_TimeStruct.Minutes);

    linedisp = map(RTC_TimeStruct.Seconds, 0, 60, 0, 160);

    M5.Lcd.drawFastHLine(0, 64, linedisp-4, RED);
    M5.Lcd.drawFastHLine(0, 65, linedisp, RED);
    M5.Lcd.drawFastHLine(0, 66, linedisp-4, RED);

    if(RTC_TimeStruct.Seconds >= LINE_CL){
        line_flg = TRUE;
    }else{
        if( line_flg == TRUE){
            clear_txt();
            line_flg = FALSE;
        }
    }
}
```

The `time_date_disp` is used to display the current date and time in Thailand.

```
void time_date_disp(void){
    int linedisp = 0;
    M5.Lcd.setCursor(0, 13, 2);
    M5.Lcd.printf("Current time");
    M5.Rtc.GetTime(&RTC_TimeStruct);
    M5.Rtc.GetData(&RTC_DateStruct);
    M5.Lcd.setCursor(0, 29, 4);
    M5.Lcd.printf("%04d / %02d / %02d\n",RTC_DateStruct.Year, RTC_DateStruct.Month,RTC_DateStruct.Date);
    M5.Lcd.setCursor(0, 52, 2);
    M5.Lcd.printf("%02d : %02d : %02d\n",RTC_TimeStruct.Hours, RTC_TimeStruct.Minutes, RTC_TimeStruct.Seconds);
    linedisp = map(RTC_TimeStruct.Seconds, 0, 70, 0, 160);
    M5.Lcd.drawFastHLine(0, 70, linedisp-4, RED);
    M5.Lcd.drawFastHLine(0, 71, linedisp, RED);
    M5.Lcd.drawFastHLine(0, 72, linedisp-4, RED);

    if(RTC_TimeStruct.Seconds >= LINE_CL){
        line_flg = TRUE;
    }else{
        if( line_flg == TRUE){
            clear_txt();
            line_flg = FALSE;
        }
    }
}
```

The `bat_disp` function shows the battery status. The value of battery status in Shadow on NETPIE2020 is updated every 4 seconds.

```

void bat_disp(void){
  int bat_disp_print ;
  vbat = (M5.Axp.GetVapsData() * 1.4);
  bat_disp_print = map(vbat, 3300, 4110, 0, 100);
  if(bat_disp_print > BAT_MAX ){
    bat_disp_print = BAT_MAX;
  }

  if(bat_disp_print < LOW_BAT_DISP){          // <10
    M5.Lcd.setCursor(113, 0, 1);
    M5.Lcd.printf("Bat:LoW");
    client.publish("@shadow/data/update", "{\"data\": {\"battery\" : 9}}");
    delay(4000);

  }else if(bat_disp_print >= BAT_MAX){        // >= 100
    M5.Lcd.setCursor(113, 0, 1);
    M5.Lcd.printf("Bat:Ful");
    client.publish("@shadow/data/update", "{\"data\": {\"battery\" : 100}}");
    delay(4000);

  }else{
    M5.Lcd.setCursor(113, 0, 1);
    M5.Lcd.printf("Bat:%d%%\n",bat_disp_print);
    String data = "{\"data\": {\"battery\" : " + (String)(bat_disp_print) + "}}";
    Serial.println(data);
    data.toCharArray(msg, (data.length() + 1));
    client.publish("@shadow/data/update", msg);
    delay(4000);
  }

  M5.Lcd.drawFastHLine(0, 8, 160, RED);
  M5.Lcd.drawFastHLine(0, 9, 160, RED);
  M5.Lcd.drawFastHLine(0, 10, 160, RED);
}

```

The `depp_sleep()` function is used to enter the sleep mode, `disp_tft_tw()` to set the screen brightness, `M5.Axp.ScreenBreath(0)` to set the screen light off, and `esp_deep_sleep_start()` to make ESP32 enter into the deep sleep.

```

void deep_sleep(void){
  clear_txt();
  M5.Lcd.setCursor(0, 30, 2);
  M5.Lcd.printf("Go to sleep mode !!\n");
  delay(2000);
  clear_txt();
  M5.Lcd.setCursor(0, 30, 2);
  M5.Lcd.printf("Get up by press the M5 \n");
  disp_tft_dw();          // bright down
  delay(3900);
  clear_txt();
  esp_deep_sleep_start();
}

```

## Transmitting timecount values to the NETPIE2020

Returns the number of milliseconds passed since the board began running the current program. In the beginning, two working conditions are set based on `millis()`. First, If the `M5_BUTTON_HOME` is clicked, it keeps counting the time and the results are shown based on the equation  $(\text{currentMillis} - \text{previousMillis}) / 1000$  in seconds. Second, if the `M5_BUTTON_RST` is clicked, counting the time stops and the results are published to the NETPIE2020, to be stored in the Shadow.

```
unsigned long currentMillis;
unsigned long previousMillis = 0;
char msg[100];
int start = 0;
```

```
void loop(void) {

    if (!client.connected()) {
        reconnect();
    }
    client.loop();

    M5.Lcd.setTextSize(2);
    M5.Lcd.setCursor(20,20);
    M5.Lcd.printf("Time Count");

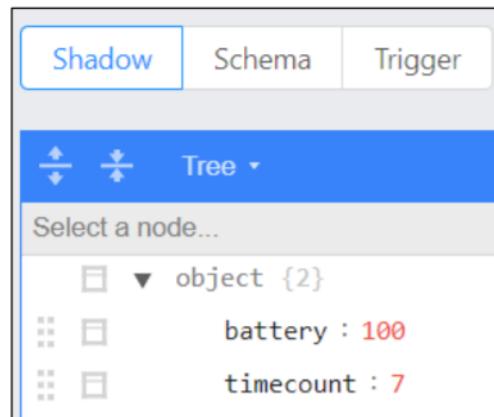
    currentMillis = millis();
    if (digitalRead(M5_BUTTON_HOME) == LOW){
        start = 1;
        previousMillis = currentMillis;
    }

    if (digitalRead(M5_BUTTON_RST) == LOW){
        start = 0;
        //previousMillis = 0;
        Serial.println("Stop !!");
        M5.Lcd.setTextSize(2);
        M5.Lcd.setCursor(20,40);
        M5.Lcd.fillScreen(BLACK);

        String showtime = (String)((currentMillis - previousMillis) / 1000) ;
        String data = "{\"data\": {\"timecount\" : " + (showtime) + "}}";
        Serial.println(data);
        data.toCharArray(msg, (data.length() + 1));
        client.publish("@shadow/data/update", msg);
        //delay(2000);
    }
}
```

```
if(start == 1){                                     //เมื่อ start = 1 จะเป็นคาราน loop จัเวลาไปเรื่อยๆ จนกว่าจะครบกำหนด
    Serial.print("Time: ");
    Serial.println((currentMillis - previousMillis) / 1000); //แสดงผลการคำนวณว่าจัเวลาไปกี่แล้ว
    String showtime = (String)((currentMillis - previousMillis) / 1000) ;
    Serial.println("Start ...");
    M5.Lcd.setTextSize(2);
    M5.Lcd.setCursor(20,40);
    M5.Lcd.println(showtime);
}
}
```

Messages sent to the NETPIE2020 and the values are saved in the shadow.



## Creating a device on NETPIE2020

1.Start by selecting the menu Device List > Create and name the device as shown in the figure below.

The 'Create' dialog box has a title bar with a close button. It contains three input fields: 'Name' (with a red asterisk and the value 'M5StickC'), 'Description' (empty), and 'Tag' (with a '+ New Tag' button). At the bottom right are 'Cancel' and 'Create' buttons.



The screenshot shows the 'NETPIE\_Training / device / Kidbright' page. It has a breadcrumb trail and an 'Edit' button. The page is divided into two sections: 'Description' on the left and 'Key' on the right. The 'Key' section contains a table of device information.

Key	
Client ID	: 8b878607-fe8d-4097-aa3d-3417a82abbc
Token	: oorQw5B7yMWEn2XWjdweF6iATJ9yy3Sf
Secret	: AWv8XJ(J)p\$3O8-x#(MSXqCMzBlrH-L-
Status	: <span style="color: green;">●</span> Online
Enable	: <span style="color: blue;">●</span>

## **Creating a device on NETPIE2020**

2.Fill in the details, which include Name, Device ID, and Device Token that can be found in the DeviceList named M5StickC.

**DATASOURCE**

NAME: M5stickC

DEVICE ID: 13a6229d-c5da-4d17-b57c-9d0d6859165a  
Client ID for Device ที่เลือกมาใช้งาน

DEVICE TOKEN: qqgUkEpHF3egu89VyLGJU2DuUQV7thDC  
Token for Device ที่เลือกมาใช้งาน

SUBSCRIBED TOPICS:   
Topic ที่เลือกมา Subscribe

FEED: YES ☒

SINCE: 6  
Hour

Display data points since ... ago

DOWN SAMPLING: 1  
Minute

Resolution of the data points

SAVE CANCEL

(NOTE: If the user enables the Feed, Freeboard keeps on calling the API. So, it is better to turn off the Feed if the user is not intended to use it.)

3.Create a widget of type Gauge for displaying the battery values by clicking on 'ADD PANE' and fill in the details as shown in the figure below.



**WIDGET**

TYPE: Gauge

TITLE: Status Battery

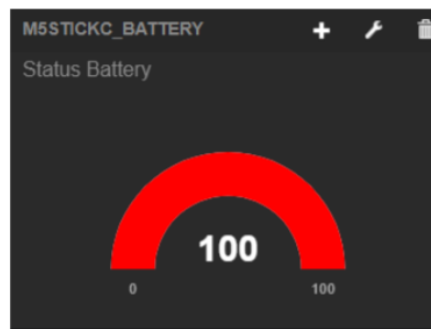
VALUE: datasources["M5stickC"]["shadow"]["battery"] + DATASOURCE JS EDITOR

UNITS:

MINIMUM: 0

MAXIMUM: 100

SAVE CANCEL



**Displaying battery data stored in the Shadow**

4. Create a widget of type Gauge for displaying the timecount values by clicking on 'ADD PANE' and fill in the details as shown in the figure below.

**WIDGET**

TYPE: Text

TITLE: time count is

SIZE: Regular

VALUE: datasources["M5stickC"]["shadow"]["timecount"] + DATASOURCE JS EDITOR

INCLUDE SPARKLINE: ☐ NO

ANIMATE VALUE CHANGES: YES ☒

UNITS: second

SAVE CANCEL

(NOTE: Enable ANIMATE VALUE CHANGES, to change the value in the VALUE field)



M5STICKC\_TIMECOUNT



time count is

7 second