TEDIZEN

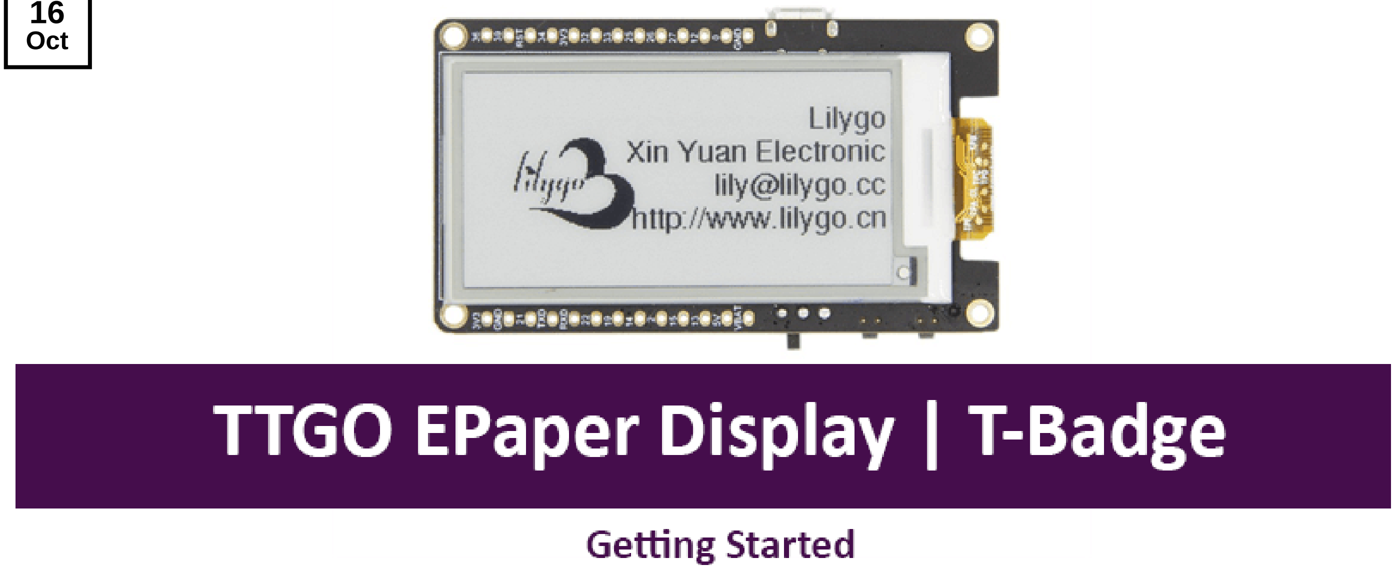
0

UNCATEGORIZED

TTGO E-Paper/T-Badge Getting Started Tutorial

POSTED ON OCTOBER 16, 2019 BY SARATH KUMAR

16
Oct

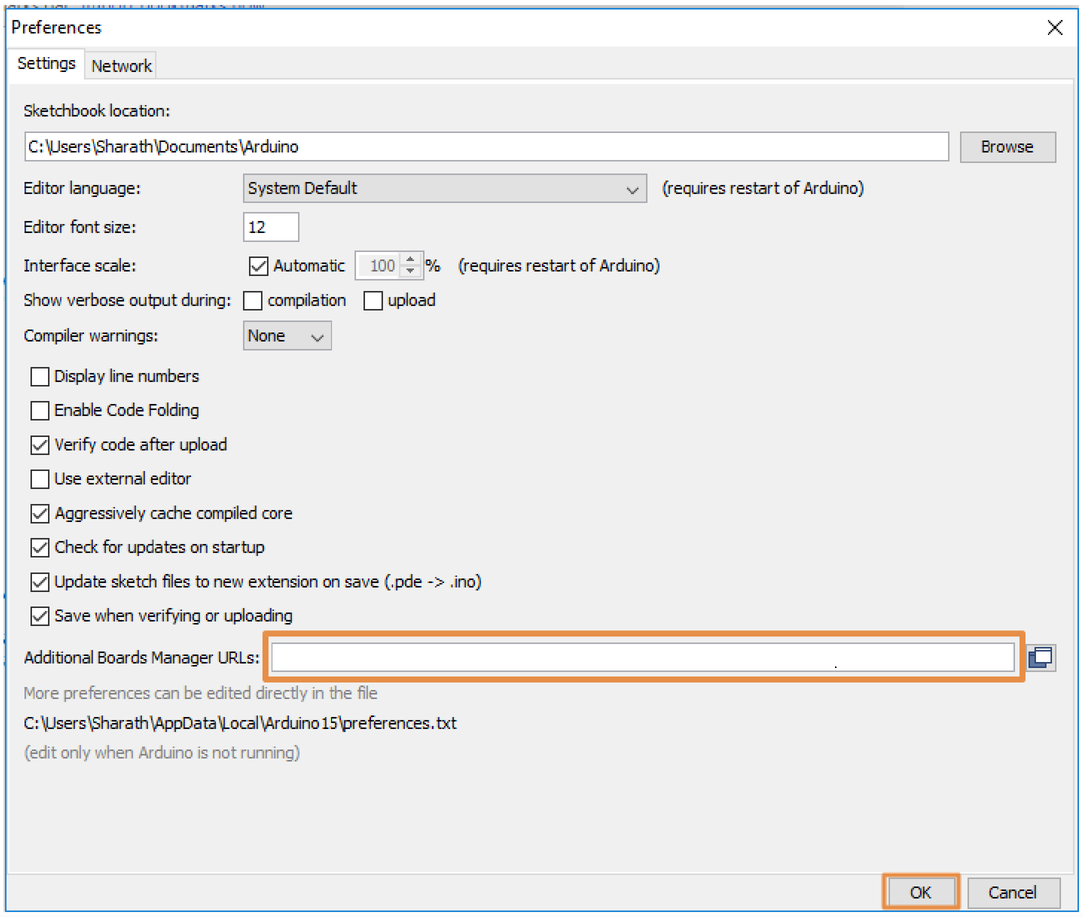


TTGO T-Badge E-Paper Display is one of the Lilygo Boards. It has wide range of boards that are very useful for development purposes as it integrates the necessary functionalities. The T-Badge board comes with E-Ink Displays. When I getting started I have only limited resources and find trouble in making it work. I'm going to share you the issues I have faced and how I resolved it to make it work.

TEDIZEN

GitHub Repository: https://github.com/lewisxhe/TTGO-EPaper-Series

Step 1: The first step is to install the ESP32 Boards in our Arduino IDE if it is not installed already. To do that go to Files>Preferences. In the additional Boards Manager URLs, we need to add the ESP32 Boards link to install all ESP32 Boards. It is a simple json file containing the board types and it will grab the necessary files that we need to install.


Preferences

Settings Network

Sketchbook location: C:\Users\Sharath\Documents\Arduino

Editor language: System Default (requires restart of Arduino)

Editor font size: 12

Interface scale: ☑ Automatic 100 % (requires restart of Arduino)

Show verbose output during: ☐ compilation ☐ upload

Compiler warnings: None

☐ Display line numbers

☐ Enable Code Folding

☑ Verify code after upload

☐ Use external editor

☑ Aggressively cache compiled core

☑ Check for updates on startup

☑ Update sketch files to new extension on save (.pde -> .ino)

☑ Save when verifying or uploading

Additional Boards Manager URLs:

More preferences can be edited directly in the file

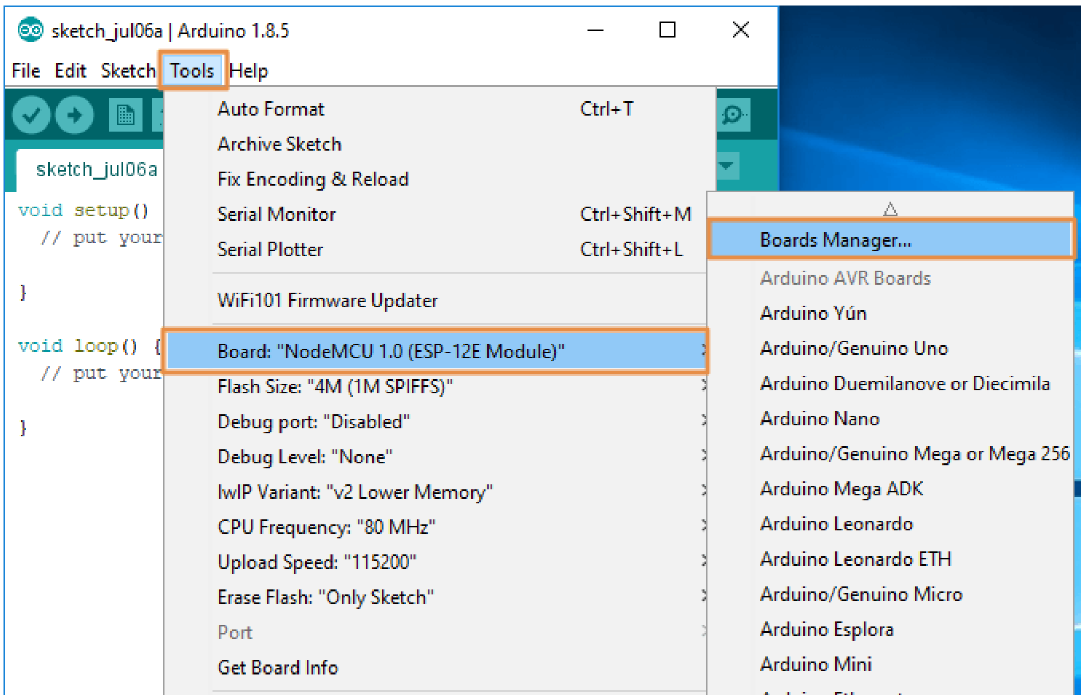
C:\Users\Sharath\AppData\Local\Arduino15\preferences.txt

(edit only when Arduino is not running)

If you have already ESP8266 URL or ATTiny, etc. You can simply put a comma(,) between the URLs and click OK.

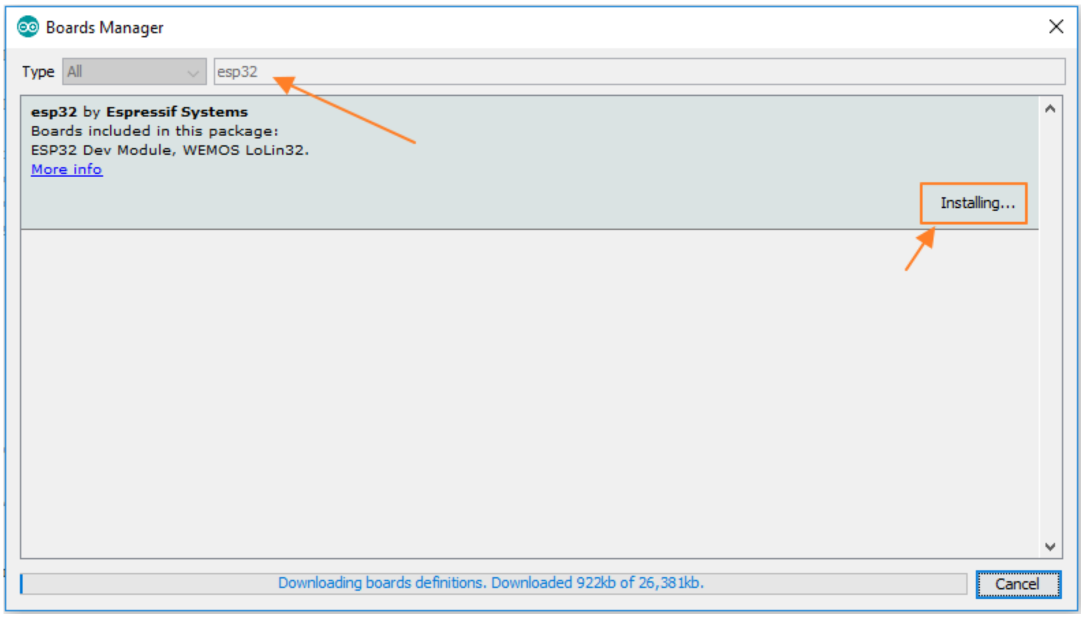
```
https://dl.espressif.com/dl/package_esp32_index.json,
http://arduino.esp8266.com/stable/package_esp8266com_index.json
```

Step 2: Now go to Tools>Boards>Boards Manager. Search ESP32 and install the boards. After installation, you can able to see ESP32 related boards on the boards menu.



```
void setup()
  // put your
}

void loop() {
  // put your
}
```

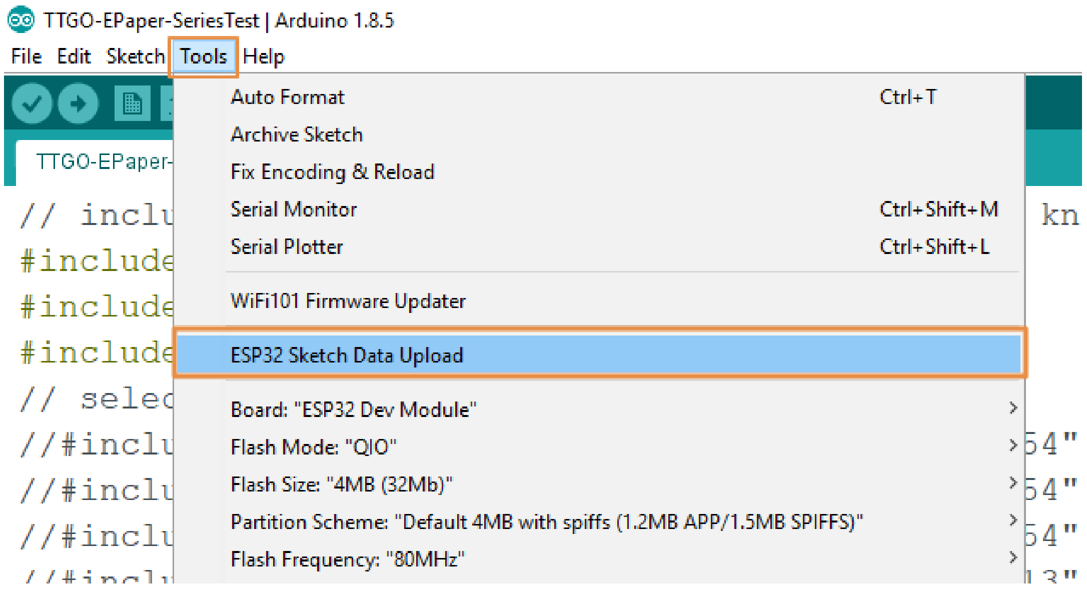


esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.

Downloading boards definitions. Downloaded 922kb of 26,381kb.

Step 3: Install a webserver resource tool to our IDE. It is a simple Java based tool to upload our image/data files to ESP32 Memory using SPIFFS.

Now you can able to see a new menu in Ardunio IDE, Tools -> ESP32 Sketch data Upload.


TTGO-EPaper-SeriesTest | Arduino 1.8.5
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 Firmware Updater
ESP32 Sketch Data Upload
Board: "ESP32 Dev Module"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Flash Frequency: "80MHz"

Note: *Inside the folder of Arduino File (.ino), there should be folder called 'data'. In that folder we need to keep the files to be uploaded to ESP32. Here we will be keeping, HTML Files, BMP, CSS,etc in this folder to make it as a mini server.*

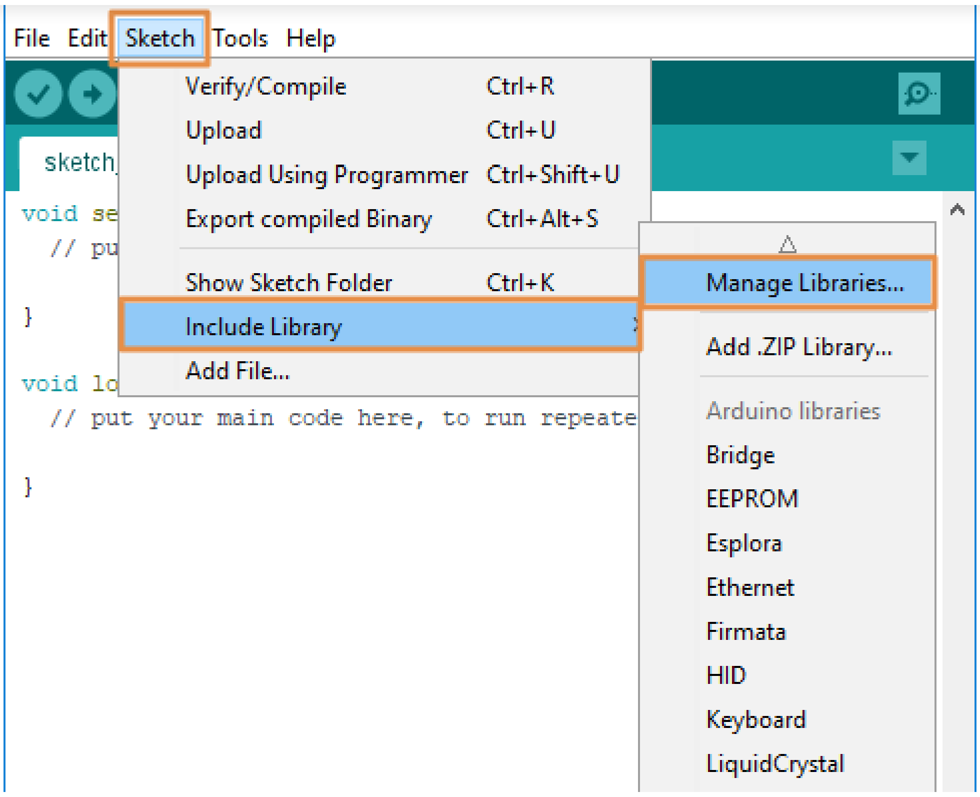
Now if we open the .ino file and choose Tools -> ESP32 Sketch data Upload, the files in the data folder will be uploaded to the ESP32's memory.

Now we just need to upload the .ino file to use ESP Programs. The data files will already in ESP Memory and will not erase when we upload code.

Before uploading the code, we need to install the dependency files, that is our library files. Here we need to install the following libraries,

- Button2
- GxEPD
- Adafruit-GFX-Library
- AsyncTCP
- ESPAsyncWebServer
- ArduinoJson
- ESP8266Audio

CAUTION: The Arduino Library Manager installs the ArduinoJSON version 6 by default. In here using version 5 is recommended because version 6 is still in beta stage. Please select 5.13.2



```
void se
  // pu

}

void lo
  // put your main code here, to run repeate

}
```

Now we are setup almost everything. Let’s gets started with the Arduino Coding part.

There are two ways we can use the example file,

- AP Mode (ESP will act as Hotspot)
- Station Mode (Accessing ESP32 via an existing network with network credentials).

We are going to use station mode example as it will be easier for beginners. The given code is for Station mode. You need to modify few things in the code.

Have a look at your Display model and Size. **The model which I use is TTGO Epaper 2.13 inch B/W. The version is T5_V2.3_2.13 (2.13 mentions the display size here).**

TEDIZEN

0

Change 1:

Now navigate to C:\Users\YourUsername\Documents\Arduino\libraries\TTGO-EPaper-Series-master folder. Open the 'board_def.h' file. At the start of the file, we will have our board versions. We need to change the version of our board from 0 to 1 and keep 0 for all other boards. I have

```
#define TTGO_T5_1_2 0
```

```
#define TTGO_T5_2_0 0
```

```
#define TTGO_T5_2_1 0
```

```
#define TTGO_T5_2_2 0
```

```
#define TTGO_T5_2_3 1 //Changed to 1 for my board version
```

```
#define TTGO_T5_2_4 0
```

```
#define TTGO_T5_2_8 0 //! Silk screen marking T5_V28_27
```

Now save and exit the file.

Open the example file given. By default, all the header file are commented. We need to uncomment the header file for our board. But as it is in development they included a new header file type for my display type that is not available in the old code. So I need to include the following header file,

Change 2: Set Your Wi-Fi Credentials here

```
#define WIFI_SSID "Your SSID"
#define WIFI_PASSWORD "Your Password"
```

Change 3: We are not using AP Mode. So the following line should be removed or 'uncomment it'.

```
//#define USE_AP_MODE
```

That's it. We can now upload the code. Select the COM Port and Choose the Board Type as 'ESP32 Dev Module' or TTGO T-Beam. Upload the code.

Code:

```
// include library, include base class, make path known
#include <GxEPD.h>
#include <GxIO/GxIO_SPI/GxIO_SPI.h>
#include <GxIO/GxIO.h>
// select the display class to use, only one
//#include <GxGDEP015OC1/GxGDEP015OC1.h>    // 1.54" b/w
//#include <GxGDEW0154Z04/GxGDEW0154Z04.h>  // 1.54" b/w/r 200x200
//#include <GxGDEW0154Z17/GxGDEW0154Z17.h>  // 1.54" b/w/r 152x152
//#include <GxGDE0213B1/GxGDE0213B1.h>      // 2.13" b/w
//#include <GxGDEW0213Z16/GxGDEW0213Z16.h>  // 2.13" b/w/r
// #include <GxGDEH029A1/GxGDEH029A1.h> // 2.9" b/w
//#include <GxGDEW029Z10/GxGDEW029Z10.h>    // 2.9" b/w/r
//#include <GxGDEW027C44/GxGDEW027C44.h>    // 2.7" b/w/r
// #include <GxGDEW027W3/GxGDEW027W3.h> // 2.7" b/w
//#include <GxGDEW042T2/GxGDEW042T2.h>      // 4.2" b/w
//#include <GxGDEW042Z15/GxGDEW042Z15.h>    // 4.2" b/w/r
//#include <GxGDEW0583T7/GxGDEW0583T7.h>    // 5.83" b/w
//#include <GxGDEW075T8/GxGDEW075T8.h>      // 7.5" b/w
// #include <GxGDEW075Z09/GxGDEW075Z09.h>    // 7.5" b/w/r
#include <GxGDE0213B72B/GxGDE0213B72B.h>      // 2.13" b/w

#include <Fonts/FreeMono9pt7b.h>
#include <Fonts/FreeMonoBoldOblique9pt7b.h>
#include <Fonts/FreeMonoBold9pt7b.h>
```

```
#include <Fonts/FreeSansOblique9pt7b.h>
#include <Fonts/FreeSerif9pt7b.h>
#include <Fonts/FreeSerifBold9pt7b.h>
#include <Fonts/FreeSerifBoldItalic9pt7b.h>
#include <Fonts/FreeSerifItalic9pt7b.h>

//#define DEFALUT_FONT  FreeMono9pt7b
// #define DEFALUT_FONT  FreeMonoBoldOblique9pt7b
// #define DEFALUT_FONT FreeMonoBold9pt7b
// #define DEFALUT_FONT FreeMonoOblique9pt7b
#define DEFALUT_FONT FreeSans9pt7b
// #define DEFALUT_FONT FreeSansBold9pt7b
// #define DEFALUT_FONT FreeSansBoldOblique9pt7b
// #define DEFALUT_FONT FreeSansOblique9pt7b
// #define DEFALUT_FONT FreeSerif9pt7b
// #define DEFALUT_FONT FreeSerifBold9pt7b
// #define DEFALUT_FONT FreeSerifBoldItalic9pt7b
// #define DEFALUT_FONT FreeSerifItalic9pt7b

const GFXfont *fonts[] = {
    &FreeMono9pt7b,
    &FreeMonoBoldOblique9pt7b,
    &FreeMonoBold9pt7b,
    &FreeMonoOblique9pt7b,
    &FreeSans9pt7b,
    &FreeSansBold9pt7b,
    &FreeSansBoldOblique9pt7b,
    &FreeSansOblique9pt7b,
    &FreeSerif9pt7b,
    &FreeSerifBold9pt7b,
    &FreeSerifBoldItalic9pt7b,
    &FreeSerifItalic9pt7b};

#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <ESPmDNS.h>
#include <Wire.h>

#include "SD.h"
#include "SPI.h"
#include <SPIFFS.h>
#include <FS.h>
#include <ArduinoJson.h>
#include "esp_wifi.h"
#include "Esp.h"
#include "board_def.h"
#include <Button2.h>

#define FILESYSTEM SPIFFS

//#define USE_AP_MODE

/*100 * 100 bmp fromat*/
```

```
#define WIFI_SSID "Your SSID"
#define WIFI_PASSWORD "Your Password"
#define CHANNEL_0 0
#define IP5306_ADDR 0X75
#define IP5306_REG_SYS_CTL0 0x00

typedef struct
{
    char name[32];
    char link[64];
    char tel[64];
    char company[64];
    char email[64];
    char address[128];
} Badge_Info_t;

typedef enum
{
    RIGHT_ALIGNMENT = 0,
    LEFT_ALIGNMENT,
    CENTER_ALIGNMENT,
} Text_alignment;

AsyncWebServer server(80);

GxIO_Class io(SPI, ELINK_SS, ELINK_DC, ELINK_RESET);
GxEPD_Class display(io, ELINK_RESET, ELINK_BUSY);

Badge_Info_t info;
static const uint16_t input_buffer_pixels = 20;       // may affect performance
static const uint16_t max_palette_pixels = 256;       // for depth <= 8
uint8_t mono_palette_buffer[max_palette_pixels / 8];  // palette buffer for depth <= 8 b/w
uint8_t color_palette_buffer[max_palette_pixels / 8]; // palette buffer for depth <= 8 c/w
uint8_t input_buffer[3 * input_buffer_pixels];        // up to depth 24
const char *path[2] = {DEFALUT_AVATAR_BMP, DEFALUT_QR_CODE_BMP};

Button2 *pBtns = nullptr;
uint8_t g_btns[] = BUTTONS_MAP;

void button_handle(uint8_t gpio)
{
    switch (gpio)
    {
#if BUTTON_1
    case BUTTON_1:
    {
        // esp_sleep_enable_ext0_wakeup((gpio_num_t)BUTTON_1, LOW);
        esp_sleep_enable_ext1_wakeup(((uint64_t)(((uint64_t)1) << BUTTON_1)), ESP_EXT1_WAKE
        Serial.println("Going to sleep now");
        delay(2000);
        esp_deep_sleep_start();
    }
    break;
#endif
```

```
            static int i = 0;
            Serial.printf("Show Num: %d font\n", i);
            i = ((i + 1) >= sizeof(fonts) / sizeof(fonts[0])) ? 0 : i + 1;
            display.setFont(fonts[i]);
            showMianPage();
        }
        break;
#endif

#if BUTTON_3
    case BUTTON_3:
    {
        static bool index = 1;
        if (!index)
        {
            showMianPage();
            index = true;
        }
        else
        {
            showQrPage();
            index = false;
        }
    }
    break;
#endif
    default:
        break;
    }
}

void button_callback(Button2 &b)
{
    for (int i = 0; i < sizeof(g_btns) / sizeof(g_btns[0]); ++i)
    {
        if (pBtns[i] == b)
        {
            Serial.printf("btn: %u press\n", pBtns[i].getAttachPin());
            button_handle(pBtns[i].getAttachPin());
        }
    }
}

void button_init()
{
    uint8_t args = sizeof(g_btns) / sizeof(g_btns[0]);
    pBtns = new Button2[args];
    for (int i = 0; i < args; ++i)
    {
        pBtns[i] = Button2(g_btns[i]);
        pBtns[i].setPressedHandler(button_callback);
    }
}
```

```
            pBtns[i].loop();
        }
    }

    void displayText(const String &str, int16_t y, uint8_t alignment)
    {
        int16_t x = 0;
        int16_t x1, y1;
        uint16_t w, h;
        display.setCursor(x, y);
        display.getTextBounds(str, x, y, &x1, &y1, &w, &h);

        switch (alignment)
        {
        case RIGHT_ALIGNMENT:
            display.setCursor(display.width() - w - x1, y);
            break;
        case LEFT_ALIGNMENT:
            display.setCursor(0, y);
            break;
        case CENTER_ALIGNMENT:
            display.setCursor(display.width() / 2 - ((w + x1) / 2), y);
            break;
        default:
            break;
        }
        display.println(str);
    }

    void saveBadgeInfo(Badge_Info_t *info)
    {
        // Open file for writing
        File file = FILESYSTEM.open(BADGE_CONFIG_FILE_NAME, FILE_WRITE);
        if (!file)
        {
            Serial.println(F("Failed to create file"));
            return;
        }
    #if ARDUINOJSON_VERSION_MAJOR == 5
        StaticJsonBuffer<256> jsonBuffer;
        JsonObject &root = jsonBuffer.createObject();
    #elif ARDUINOJSON_VERSION_MAJOR == 6
        StaticJsonDocument<256> root;
    #endif
        // Set the values
        root["company"] = info->company;
        root["name"] = info->name;
        root["address"] = info->address;
        root["email"] = info->email;
        root["link"] = info->link;
        root["tel"] = info->tel;

    #if ARDUINOJSON_VERSION_MAJOR == 5
        if (root.printTo(file) == 0)
```

```
            Serial.println(F("Failed to write to file"));
        }
        // Close the file (File's destructor doesn't close the file)
        file.close();
    }

    void loadDefaultInfo(void)
    {
        strlcpy(info.company, "Xin Yuan Electronic", sizeof(info.company));
        strlcpy(info.name, "Lilygo", sizeof(info.name));
        strlcpy(info.address, "ShenZhen", sizeof(info.address));
        strlcpy(info.email, "lily@lilygo.cc", sizeof(info.email));
        strlcpy(info.link, "http://www.lilygo.cn", sizeof(info.link));
        strlcpy(info.tel, "0755-83380665", sizeof(info.tel));
        saveBadgeInfo(&info);
    }

    bool loadBadgeInfo(Badge_Info_t *info)
    {
        if (!FILESYSTEM.exists(BADGE_CONFIG_FILE_NAME))
        {
            Serial.println("load configure fail");
            return false;
        }

        File file = FILESYSTEM.open(BADGE_CONFIG_FILE_NAME);
        if (!file)
        {
            Serial.println("Open Fial -->");
            return false;
        }

    #if ARDUINOJSON_VERSION_MAJOR == 5
        StaticJsonBuffer<256> jsonBuffer;
        JsonObject &root = jsonBuffer.parseObject(file);
        if (!root.success())
        {
            Serial.println(F("Failed to read file, using default configuration"));
            file.close();
            return false;
        }
        root.printTo(Serial);
    #elif ARDUINOJSON_VERSION_MAJOR == 6
        StaticJsonDocument<256> root;
        DeserializationError error = deserializeJson(root, file);
        if (error)
        {
            Serial.println(F("Failed to read file, using default configuration"));
        }
    #endif
        if (!root.get<const char *>("company") || !root.get<const char *>("name") || !root.get<
        {
            file.close();
            return false;
        strlcpy(info->email, root["email"], sizeof(info->email));
        strlcpy(info->link, root["link"], sizeof(info->link));
        strlcpy(info->tel, root["tel"], sizeof(info->tel));
        file.close();
        return true;
    }

    void WebServerStart(void)
    {

    #ifdef USE_AP_MODE
        uint8_t mac[6];
        char apName[18] = {0};
        IPAddress apIP = IPAddress(192, 168, 1, 1);

        WiFi.mode(WIFI_AP);

        WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));

        esp_wifi_get_mac(WIFI_IF_STA, mac);

        sprintf(apName, "TTGO-Badge-%02X:%02X", mac[4], mac[5]);

        if (!WiFi.softAP(apName, NULL, 1, 0, 1))
        {
            Serial.println("AP Config failed.");
            return;
        }
        else
        {
            Serial.println("AP Config Success.");
            Serial.print("AP MAC: ");
            Serial.println(WiFi.softAPmacAddress());
        }
    #else
        WiFi.mode(WIFI_STA);
        WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

        while (WiFi.waitForConnectResult() != WL_CONNECTED)
        {
            Serial.print(".");
            esp_restart();
        }
        Serial.println(F("WiFi connected"));
        Serial.println("");
        Serial.println(WiFi.localIP());
    #endif

        if (MDNS.begin("ttgo"))
        {
            Serial.println("MDNS responder started");
        }

        server.serveStatic("/", FILESYSTEM, "/").setDefaultFile("index.html");
        server.on("js/jquery.min.js", HTTP_GET, [](AsyncWebServerRequest *request) {
            request->send(FILESYSTEM, "js/jquery.min.js", "application/javascript");
        });
        server.on("js/tbdValidate.js", HTTP_GET, [](AsyncWebServerRequest *request) {
            request->send(FILESYSTEM, "js/tbdValidate.js", "application/javascript");
        });
        server.on("/data", HTTP_POST, [](AsyncWebServerRequest *request) {
            request->send(200, "text/plain", "");

            for (int i = 0; i < request->params(); i++)
            {
                String name = request->getParam(i)->name();
                String params = request->getParam(i)->value();
                Serial.println(name + " : " + params);
                if (name == "company")
                {
                    strlcpy(info.company, params.c_str(), sizeof(info.company));
                }
                else if (name == "name")
                {
                    strlcpy(info.name, params.c_str(), sizeof(info.name));
                }
                else if (name == "address")
                {
                    strlcpy(info.address, params.c_str(), sizeof(info.address));
                }
                else if (name == "email")
                {
                    strlcpy(info.email, params.c_str(), sizeof(info.email));
                }
                else if (name == "link")
                {
                    strlcpy(info.link, params.c_str(), sizeof(info.link));
                }
                else if (name == "tel")
                {
                    strlcpy(info.tel, params.c_str(), sizeof(info.tel));
                }
            }
            saveBadgeInfo(&info);
        });

        server.onFileUpload([](AsyncWebServerRequest *request, const String &filename, size_t i
            static File file;
            static int pathIndex = 0;
            if (!index)
            {
                Serial.printf("UploadStart: %s\n", filename.c_str());
                file = FILESYSTEM.open(path[pathIndex], FILE_WRITE);
                if (!file)
                {
                    Serial.println("Open FAIL");
                    request->send(500, "text/plain", "hander error");
                    return;
                Serial.println("Write fail");
                request->send(500, "text/plain", "hander error");
                file.close();
                return;
            }

            if (final)
            {
                Serial.printf("UploadEnd: %s (%u)\n", filename.c_str(), index + len);
                file.close();
                request->send(200, "text/plain", "");
                if (++pathIndex >= 2)
                {
                    pathIndex = 0;
                    showMianPage();
                }
            }
        });

        server.onNotFound([](AsyncWebServerRequest *request) {
            request->send(404, "text/plain", "Not found");
        });

        MDNS.addService("http", "tcp", 80);

        server.begin();
    }

    void showMianPage(void)
    {
        displayInit();
        display.fillScreen(GxEPD_WHITE);
        drawBitmap(DEFALUT_AVATAR_BMP, 10, 10, true);
        displayText(String(info.name), 30, RIGHT_ALIGNMENT);
        displayText(String(info.company), 50, RIGHT_ALIGNMENT);
        displayText(String(info.email), 70, RIGHT_ALIGNMENT);
        displayText(String(info.link), 90, RIGHT_ALIGNMENT);
        display.update();
    }

    void showQrPage(void)
    {
        displayInit();
        display.fillScreen(GxEPD_WHITE);
        drawBitmap(DEFALUT_QR_CODE_BMP, 10, 10, true);
        displayText(String(info.tel), 50, RIGHT_ALIGNMENT);
        displayText(String(info.email), 70, RIGHT_ALIGNMENT);
        displayText(String(info.address), 90, RIGHT_ALIGNMENT);
        display.update();
    }

    uint16_t read16(File &f)
    {
        // BMP data is stored little-endian, same as Arduino.
```

```
}

uint32_t read32(File &f)
{
    // BMP data is stored little-endian, same as Arduino.
    uint32_t result;
    ((uint8_t *)&result)[0] = f.read(); // LSB
    ((uint8_t *)&result)[1] = f.read();
    ((uint8_t *)&result)[2] = f.read();
    ((uint8_t *)&result)[3] = f.read(); // MSB
    return result;
}

void drawBitmap(const char *filename, int16_t x, int16_t y, bool with_color)
{
    File file;
    bool valid = false; // valid format to be handled
    bool flip = true;   // bitmap is stored bottom-to-top
    uint32_t startTime = millis();
    if ((x >= display.width()) || (y >= display.height()))
        return;
    Serial.println();
    Serial.print("Loading image '");
    Serial.print(filename);
    Serial.println('\'');

    file = FILESYSTEM.open(filename, FILE_READ);
    if (!file)
    {
        Serial.print("File not found");
        return;
    }

    // Parse BMP header
    if (read16(file) == 0x4D42)
    { // BMP signature
        uint32_t fileSize = read32(file);
        uint32_t creatorBytes = read32(file);
        uint32_t imageOffset = read32(file); // Start of image data
        uint32_t headerSize = read32(file);
        uint32_t width = read32(file);
        uint32_t height = read32(file);
        uint16_t planes = read16(file);
        uint16_t depth = read16(file); // bits per pixel
        uint32_t format = read32(file);
        if ((planes == 1) && ((format == 0) || (format == 3)))
        { // uncompressed is handled, 565 also
            Serial.print("File size: ");
            Serial.println(fileSize);
            Serial.print("Image Offset: ");
            Serial.println(imageOffset);
            Serial.print("Header size: ");
            Serial.println(headerSize);
            Serial.print("Bit Depth: ");
            Serial.println(height);
            // BMP rows are padded (if needed) to 4-byte boundary
            uint32_t rowSize = (width * depth / 8 + 3) & ~3;
            if (depth < 8)
                rowSize = ((width * depth + 8 - depth) / 8 + 3) & ~3;
            if (height < 0)
            {
                height = -height;
                flip = false;
            }
            uint16_t w = width;
            uint16_t h = height;
            if ((x + w - 1) >= display.width())
                w = display.width() - x;
            if ((y + h - 1) >= display.height())
                h = display.height() - y;
            valid = true;
            uint8_t bitmask = 0xFF;
            uint8_t bitshift = 8 - depth;
            uint16_t red, green, blue;
            bool whitish, colored;
            if (depth == 1)
                with_color = false;
            if (depth <= 8)
            {
                if (depth < 8)
                    bitmask >>= depth;
                file.seek(54); //palette is always @ 54
                for (uint16_t pn = 0; pn < (1 << depth); pn++)
                {
                    blue = file.read();
                    green = file.read();
                    red = file.read();
                    file.read();
                    whitish = with_color ? ((red > 0x80) && (green > 0x80) && (blue > 0x80)
                    colored = (red > 0xF0) || ((green > 0xF0) && (blue > 0xF0));
                    if (0 == pn % 8)
                        mono_palette_buffer[pn / 8] = 0;
                    mono_palette_buffer[pn / 8] |= whitish << pn % 8;
                    if (0 == pn % 8)
                        color_palette_buffer[pn / 8] = 0;
                    color_palette_buffer[pn / 8] |= colored << pn % 8;
                }
            }
            display.fillScreen(GxEPD_WHITE);
            uint32_t rowPosition = flip ? imageOffset + (height - h) * rowSize : imageOffse
            for (uint16_t row = 0; row < h; row++, rowPosition += rowSize)
            { // for each line
                uint32_t in_remain = rowSize;
                uint32_t in_idx = 0;
                uint32_t in_bytes = 0;
                uint8_t in_byte = 0; // for depth <= 8
                uint8_t in_bits = 0; // for depth <= 8
                uint16_t color = GxEPD_WHITE;
```

```
if (in_idx >= in_bytes)
{ // ok, exact match for 24bit also (size IS multiple of 3)
    in_bytes = file.read(input_buffer, in_remain > sizeof(input_buffer)
    in_remain -= in_bytes;
    in_idx = 0;
}
switch (depth)
{
case 24:
    blue = input_buffer[in_idx++];
    green = input_buffer[in_idx++];
    red = input_buffer[in_idx++];
    whitish = with_color ? ((red > 0x80) && (green > 0x80) && (blue > 0
    colored = (red > 0xF0) || ((green > 0xF0) && (blue > 0xF0));
    break;
case 16:
{
    uint8_t lsb = input_buffer[in_idx++];
    uint8_t msb = input_buffer[in_idx++];
    if (format == 0)
    { // 555
        blue = (lsb & 0x1F) << 3;
        green = ((msb & 0x03) << 6) | ((lsb & 0xE0) >> 2);
        red = (msb & 0x7C) << 1;
    }
    else
    { // 565
        blue = (lsb & 0x1F) << 3;
        green = ((msb & 0x07) << 5) | ((lsb & 0xE0) >> 3);
        red = (msb & 0xF8);
    }
    whitish = with_color ? ((red > 0x80) && (green > 0x80) && (blue > 0
    colored = (red > 0xF0) || ((green > 0xF0) && (blue > 0xF0));
}
break;
case 1:
case 4:
case 8:
{
    if (0 == in_bits)
    {
        in_byte = input_buffer[in_idx++];
        in_bits = 8;
    }
    uint16_t pn = (in_byte >> bitshift) & bitmask;
    whitish = mono_palette_buffer[pn / 8] & (0x1 << pn % 8);
    colored = color_palette_buffer[pn / 8] & (0x1 << pn % 8);
    in_byte <<= depth;
    in_bits -= depth;
}
break;
}
if (whitish)
{
```

```
                            color = GxEPD_RED;
                        }
                        else
                        {
                            color = GxEPD_BLACK;
                        }
                        uint16_t yrow = y + (flip ? h - row - 1 : row);
                        display.drawPixel(x + col, yrow, color);
                    } // end pixel
                }     // end line
                Serial.print("loaded in ");
                Serial.print(millis() - startTime);
                Serial.println(" ms");
            }
        }
        file.close();
        if (!valid)
        {
            Serial.println("bitmap format not handled.");
        }
    }

    void displayInit(void)
    {
        static bool isInit = false;
        if (isInit)
        {
            return;
        }
        isInit = true;
        display.init();
        display.setRotation(1);
        display.eraseDisplay();
        display.setTextColor(GxEPD_BLACK);
        display.setFont(&DEFALUT_FONT);
        display.setTextSize(0);

        if (SDCARD_SS > 0)
        {
            display.fillScreen(GxEPD_WHITE);
    #if !(TTGO_T5_2_2)
            SPIClass sdSPI(VSPI);
            sdSPI.begin(SDCARD_CLK, SDCARD_MISO, SDCARD_MOSI, SDCARD_SS);
            if (!SD.begin(SDCARD_SS, sdSPI))
    #else
            if (!SD.begin(SDCARD_SS))
    #endif
            {
                displayText("SDCard MOUNT FAIL", 50, CENTER_ALIGNMENT);
            }
            else
            {
                displayText("SDCard MOUNT PASS", 50, CENTER_ALIGNMENT);
                uint32_t cardSize = SD.cardSize() / (1024 * 1024);
        }
    }

    bool setPowerBoostKeepOn(int en)
    {
        Wire.beginTransmission(IP5306_ADDR);
        Wire.write(IP5306_REG_SYS_CTL0);
        if (en)
            Wire.write(0x37); // Set bit1: 1 enable 0 disable boost keep on
        else
            Wire.write(0x35); // 0x37 is default reg value
        return Wire.endTransmission() == 0;
    }

    void setup()
    {
        Serial.begin(115200);
        delay(500);

    #ifdef ENABLE_IP5306
        Wire.begin(I2C_SDA, I2C_SCL);
        bool ret = setPowerBoostKeepOn(1);
        Serial.printf("Power KeepUp %s\n", ret ? "PASS" : "FAIL");
    #endif

    // It is only necessary to turn on the power amplifier power supply on the T5_V24 board.
    #ifdef AMP_POWER_CTRL
        pinMode(AMP_POWER_CTRL, OUTPUT);
        digitalWrite(AMP_POWER_CTRL, HIGH);
    #endif

    #ifdef DAC_MAX98357
        AudioGeneratorMP3 *mp3;
        AudioFileSourcePROGMEM *file;
        AudioOutputI2S *out;
        AudioFileSourceID3 *id3;

        file = new AudioFileSourcePROGMEM(image, sizeof(image));
        id3 = new AudioFileSourceID3(file);
        out = new AudioOutputI2S();
        out->SetPinout(IIS_BCK, IIS_WS, IIS_DOUT);
        mp3 = new AudioGeneratorMP3();
        mp3->begin(id3, out);
        while (1)
        {
            if (mp3->isRunning())
            {
                if (!mp3->loop())
                    mp3->stop();
            }
            else
            {
                Serial.printf("MP3 done\n");
                break;
        if (SPEAKER_OUT > 0)
        {
            ledcSetup(CHANNEL_0, 1000, 8);
            ledcAttachPin(SPEAKER_OUT, CHANNEL_0);
            int i = 3;
            while (i--)
            {
                ledcWriteTone(CHANNEL_0, 1000);
                delay(200);
                ledcWriteTone(CHANNEL_0, 0);
            }
        }
        SPI.begin(SPI_CLK, SPI_MISO, SPI_MOSI, -1);
        if (!FILESYSTEM.begin())
        {
            Serial.println("FILESYSTEM is not database");
            Serial.println("Please use Arduino ESP32 Sketch data Upload files");
            while (1)
            {
                delay(1000);
            }
        }
        if (!loadBadgeInfo(&info))
        {
            loadDefaultInfo();
        }

        if (esp_sleep_get_wakeup_cause() == ESP_SLEEP_WAKEUP_UNDEFINED)
        {
            showMianPage();
        }

        WebServerStart();

        button_init();
    }

    void loop()
    {
        button_loop();
    }
```

Now if you press the button 2(IO39), you can able to toggle the displays. If you want to change the contents, open serial monitor and wait for few seconds to see the IP Address. Copy and paste it in a browser and hit enter.

You can able to see a form. Type in your details and submit. In the telephone field, other than this number '18503000000' all other number shows malformed number. I have checked the