Badge My IoT Life - Part II

Paul Pagel

Badge My IoT Life – Part II

GitHub repo:
https://github.com/DigiTorus86/ESP32Badge

Slide Decks and Labs (PDF):
https://github.com/DigiTorus86/ESP32Badge/tree/master/documents

Session Objectives

- What can the ESP32 badge do?
- How can I incorporate my own graphics and digital media?
- How do I build a personalized badge application?
- How can I use the wireless capabilities of the ESP32?
- What are some next steps I can take?

Out of Scope

This session will not:

- Dive deep into the low-level details of MCU architecture or interface protocols
- Require expertise with C or C++ (though it will help)
- Make you an EE

Disclaimers

- Not a Mac guy
- Provided_code != production_code

Safety

- Lead is a common component in electronics, so wash hands before eating and avoid rubbing eyes.
- Keep your work area clear, especially of open liquid containers.
- Some components can be damaged by static shock, so grounding yourself after walking around is generally a good idea.
- **If you see something that looks unsafe, call a stop so it can be evaluated and addressed.**

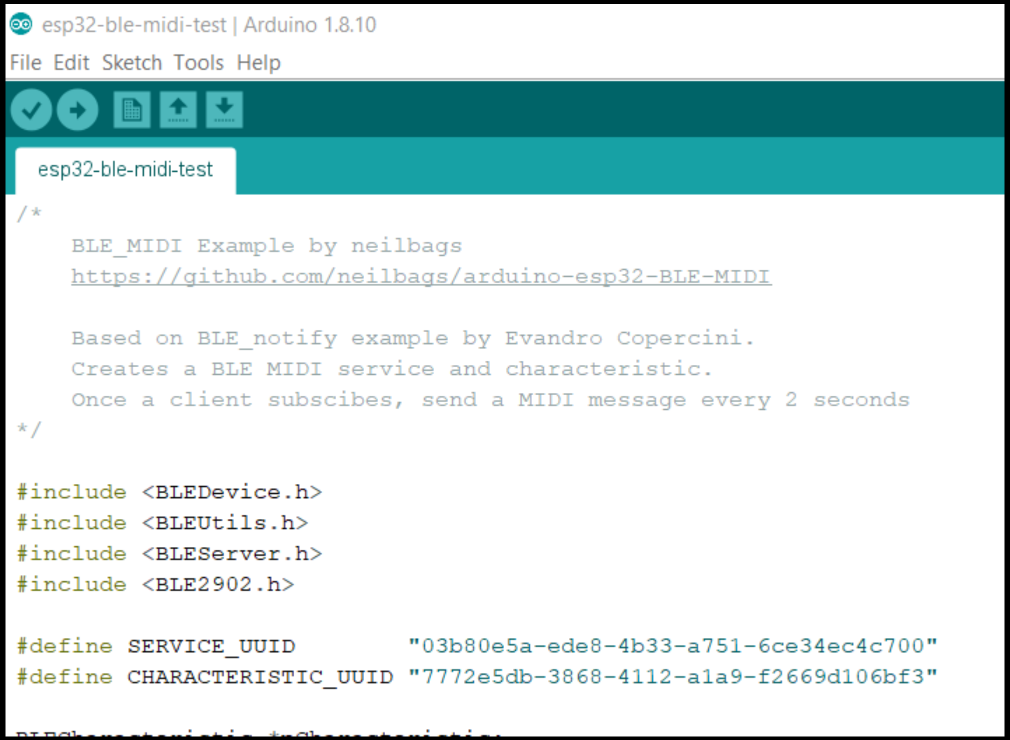
Microcontroller Development Environments

A Quick Overview

Arduino IDE

Pros

- Simple to use
- Very mature
- Wide device and library support
- Many tutorials available



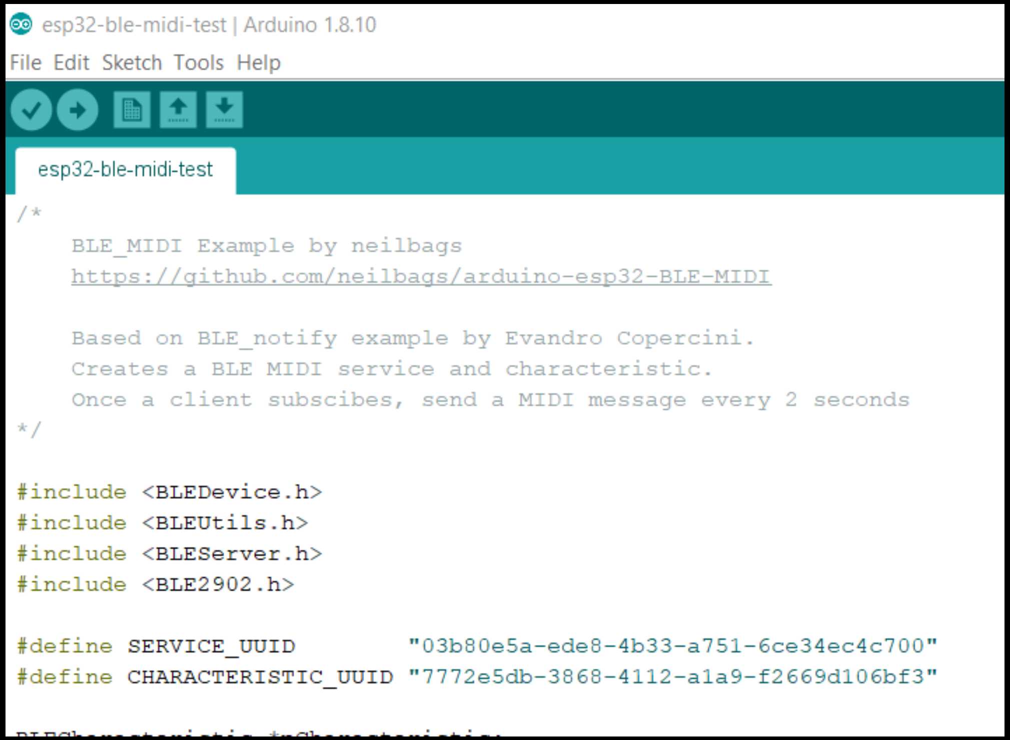
esp32-ble-midi-test | Arduino 1.8.10
File Edit Sketch Tools Help
esp32-ble-midi-test

```
/*
    BLE_MIDI Example by neilbags
    https://github.com/neilbags/arduino-esp32-BLE-MIDI

    Based on BLE_notify example by Evandro Copercini.
    Creates a BLE MIDI service and characteristic.
    Once a client subscibes, send a MIDI message every 2 seconds
*/

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <BLE2902.h>

#define SERVICE_UUID        "03b80e5a-ede8-4b33-a751-6ce34ec4c700"
#define CHARACTERISTIC_UUID "7772e5db-3868-4112-a1a9-f2669d106bf3"
```

Arduino IDE

Cons

- UI is a bit dated
- No auto-complete
- Limited debugging capabilities
- New "Pro" version looks like it will address some of these issues, but is still in alpha

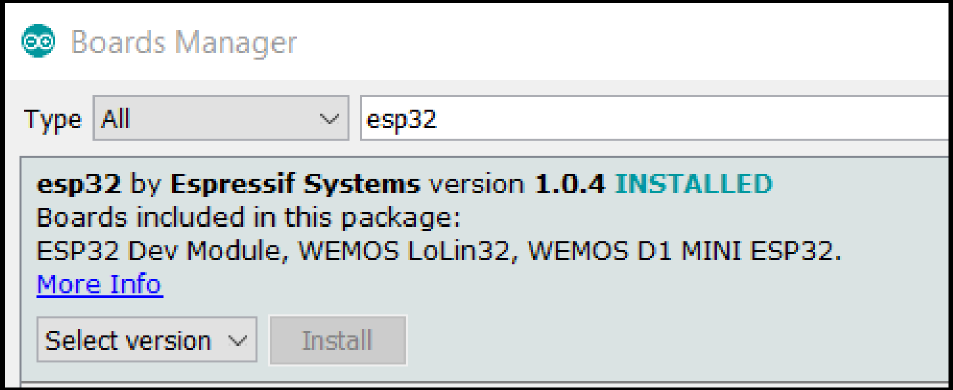

esp32-ble-midi-test | Arduino 1.8.10
File Edit Sketch Tools Help
esp32-ble-midi-test

```
/*
   BLE_MIDI Example by neilbags
   https://github.com/neilbags/arduino-esp32-BLE-MIDI

   Based on BLE_notify example by Evandro Copercini.
   Creates a BLE MIDI service and characteristic.
   Once a client subscibes, send a MIDI message every 2 seconds
*/

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <BLE2902.h>

#define SERVICE_UUID        "03b80e5a-ede8-4b33-a751-6ce34ec4c700"
#define CHARACTERISTIC_UUID "7772e5db-3868-4112-a1a9-f2669d106bf3"
```

Arduino IDE Verification

Go to Tools -> Board and verify that **NodeMCU-32S** is in the list

If it is not, go to Tools -> Board -> Boards Manager… and verify that you have version 1.0.4 or higher of the esp32 board toolchain installed:

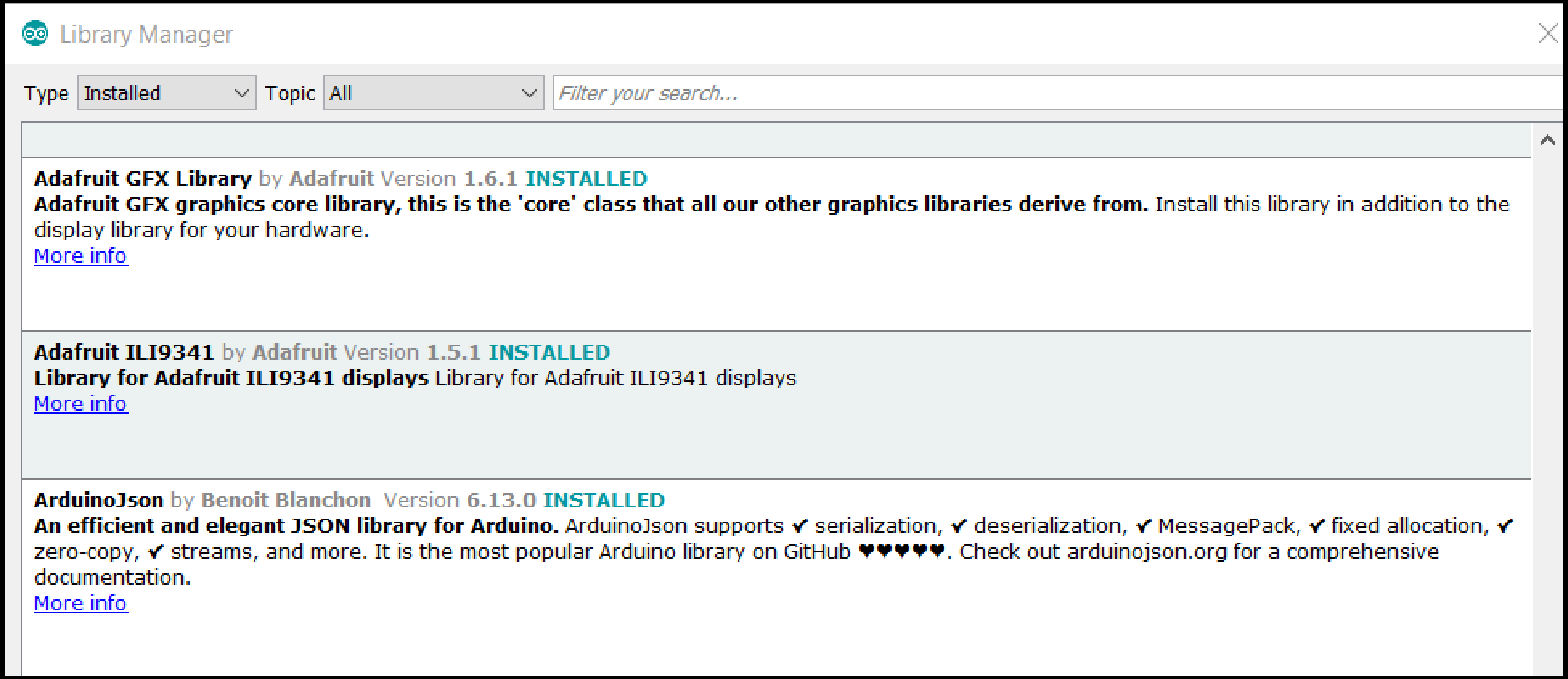


esp32

esp32 by **Espressif Systems** version **1.0.4** **INSTALLED**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32, WEMOS D1 MINI ESP32.
More Info

Arduino IDE Verification

1. Go to Tools-> Manage Libraries
2. Select Type "Installed" and verify these libraries are shown as INSTALLED:



Adafruit GFX Library by **Adafruit** Version **1.6.1** **INSTALLED**
Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from. Install this library in addition to the display library for your hardware.
More info

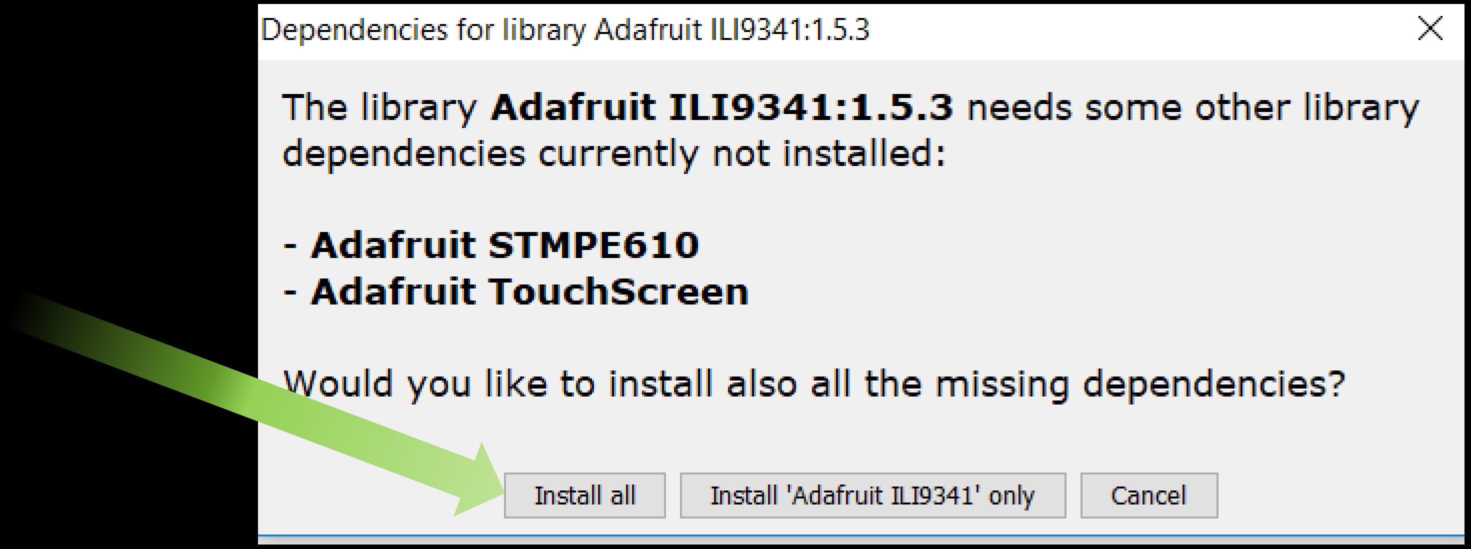
Adafruit ILI9341 by **Adafruit** Version **1.5.1** **INSTALLED**
Library for Adafruit ILI9341 displays Library for Adafruit ILI9341 displays
More info

ArduinoJson by **Benoit Blanchon** Version **6.13.0** **INSTALLED**
An efficient and elegant JSON library for Arduino. ArduinoJson supports ✔ serialization, ✔ deserialization, ✔ MessagePack, ✔ fixed allocation, ✔ zero-copy, ✔ streams, and more. It is the most popular Arduino library on GitHub ♥♥♥♥♥. Check out arduinojson.org for a comprehensive documentation.
More info

Library Dependencies

Some of the libraries have recent updates (within the last month.) They are not needed for this workshop.

If you do upgrade to the latest versions, make sure to install the new additional library dependencies with them.


Dependencies for library Adafruit ILI9341:1.5.3

The library **Adafruit ILI9341:1.5.3** needs some other library dependencies currently not installed:

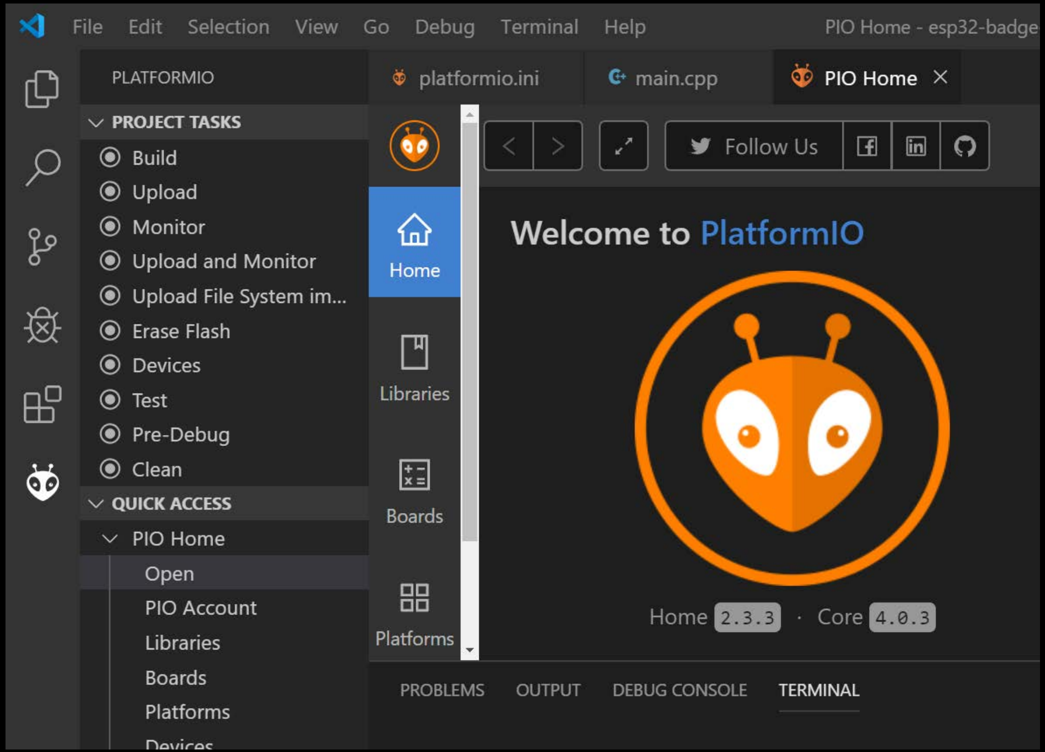
- **Adafruit STMPE610**
- **Adafruit TouchScreen**

Would you like to install also all the missing dependencies?

PlatformIO

Pros

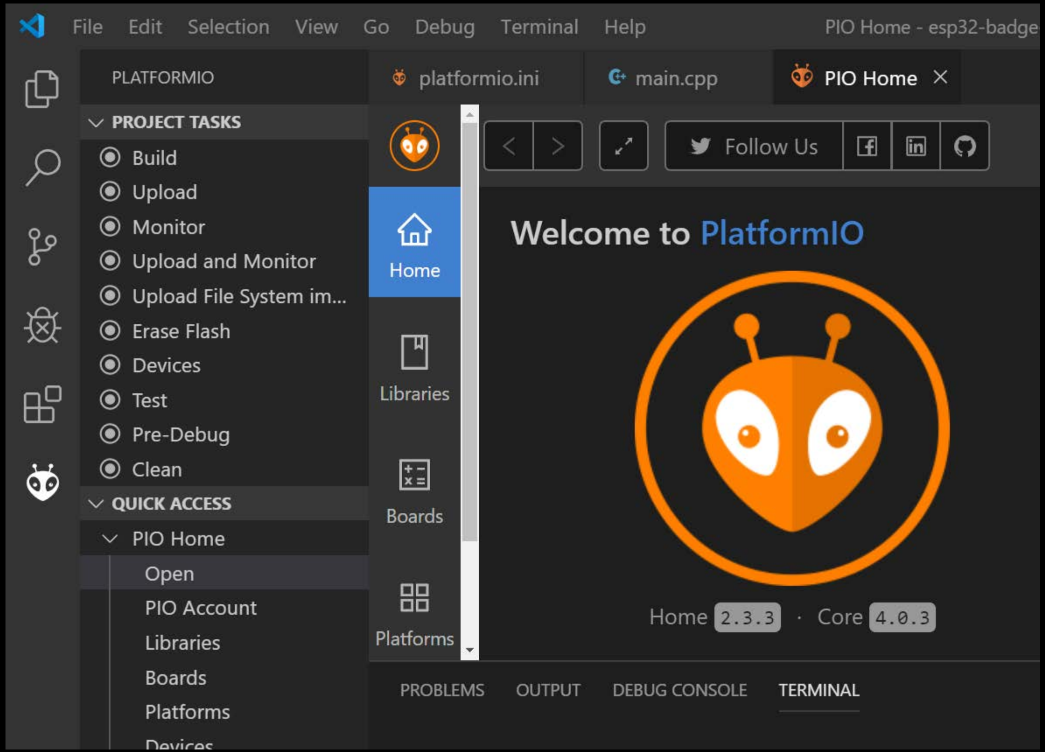
- Integrates with VS Code
- Modern UI
- Auto-complete
- Available integrated debugging
- Supports frameworks other than Arduino



PlatformIO

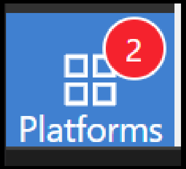
Cons

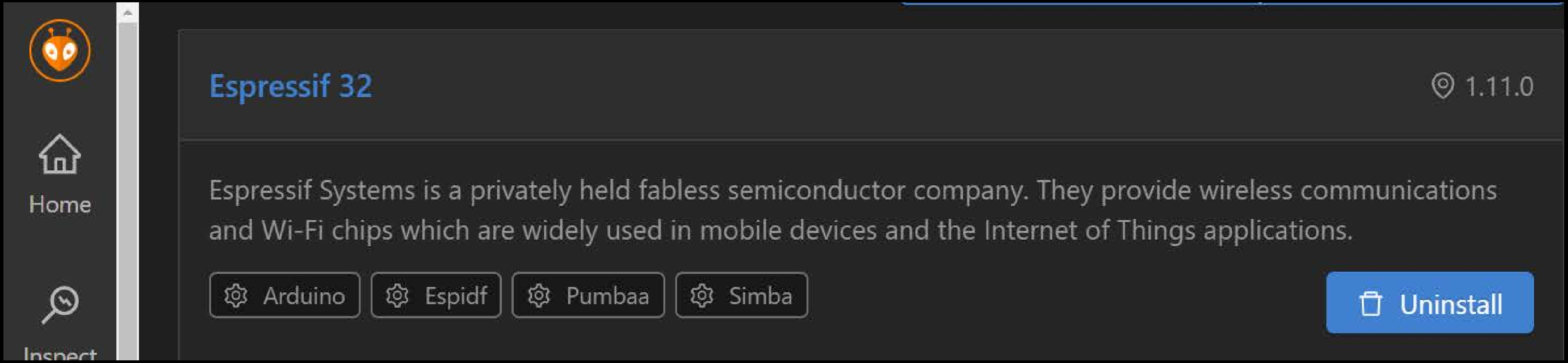
- Can be a bit more confusing for people not used to VS Code
- Fewer training resources available



PlatformIO Setup Verification

1. Click the Platforms icon
2. Verify that Espressif 32 is shown



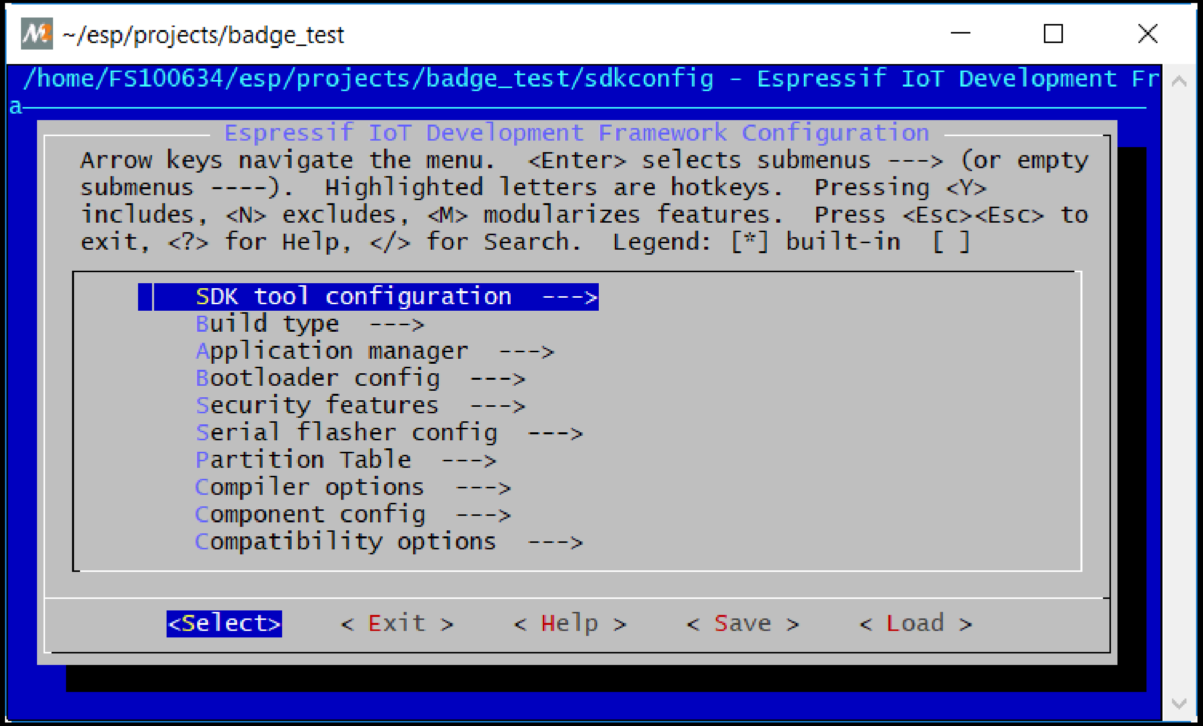


Espressif Systems is a privately held fabless semiconductor company. They provide wireless communications and Wi-Fi chips which are widely used in mobile devices and the Internet of Things applications.

Espressif ESP-IDF

IoT Development Framework

- Make and GCC based build system
- Uses a different set of hardware abstraction libraries than Arduino
- Requires more code but provides more flexibility



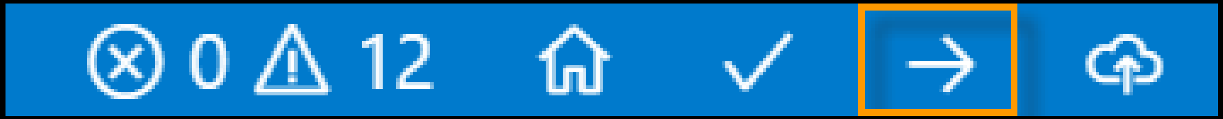
```
~/esp/projects/badge_test
 /home/FS100634/esp/projects/badge_test/sdkconfig - Espressif IoT Development Fr
a
               Espressif IoT Development Framework Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

          SDK tool configuration  --->
          Build type  --->
          Application manager  --->
          Bootloader config  --->
          Security features  --->
          Serial flasher config  --->
          Partition Table  --->
          Compiler options  --->
          Component config  --->
          Compatibility options  --->

      <Select>    < Exit >    < Help >    < Save >    < Load >
```

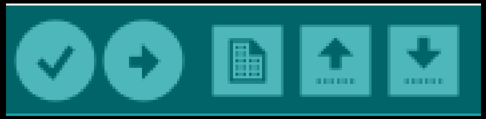
Lab #1
Hello World!

The Time-Honored Classic

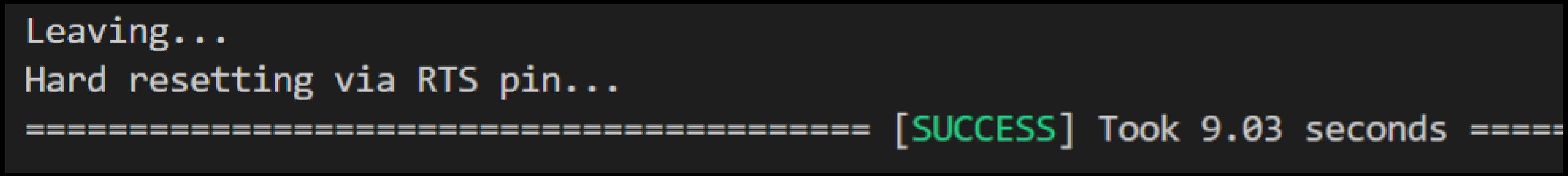
Hello Badge

1. Create a new project with a name like "Hello_Badge" for the NodeMCU-32S
2. In the setup function, enter:
   ```
   Serial.begin(9600);
   Serial.println("Hello Badge");
   ```
3. Click the Upload (right arrow) icon:





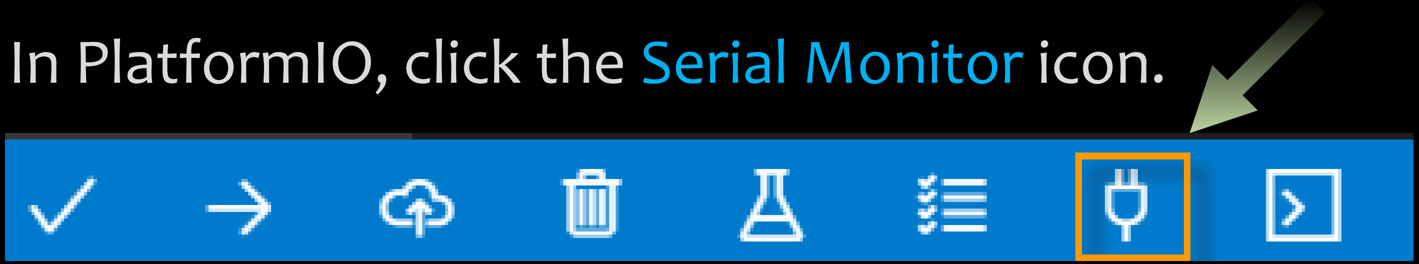
4. Verify that it builds correctly and uploads to the board:



```
Leaving...
Hard resetting via RTS pin...
========================================== [SUCCESS] Took 9.03 seconds =====
```

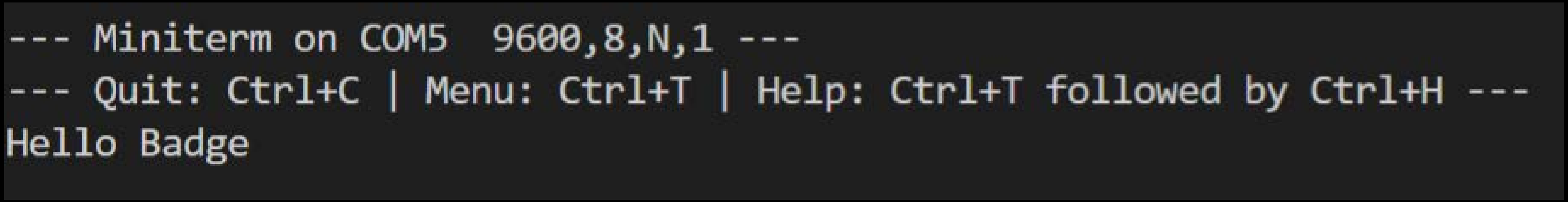
Serial Monitor

5. In PlatformIO, click the Serial Monitor icon.



   In Arduino IDE, select **Tools -> Serial Monitor**.

6. Run the app again and verify that your “Hello Badge” message is shown in the Serial Monitor window:

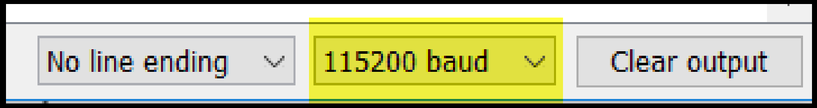


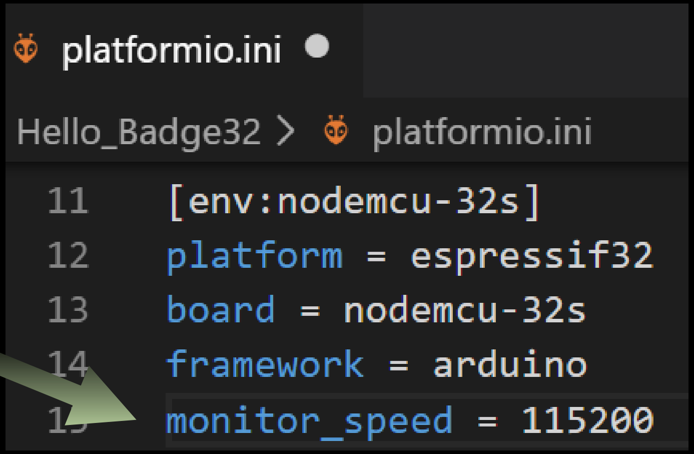
```
--- Miniterm on COM5  9600,8,N,1 ---
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H ---
Hello Badge
```

About the Serial Monitor

Your serial begin parameter must match the Serial Monitor setting!

- Try changing the speed param in the code to: `Serial.begin(115200);`
- In the Arduino IDE, this is selected from a dropdown on the serial monitor window.
- In PlatformIO, open **platformio.ini** and enter: `monitor_speed = 115200`





```
[env:nodemcu-32s]
platform = espressif32
board = nodemcu-32s
framework = arduino
monitor_speed = 115200
```

My Program Won’t Load!!!

- Try again
- Hold down the **Boot** button on the ESP32 when you see:
  Connecting … ____…____…
- Verify the USB port is correct and operational (Device Manager)
- Power cycle the ESP32
- Swap out the cable (just because it works for power, does not mean it will work correctly for data)
- Restart the IDE

Troubleshooting

Invalid character compiler error

- Often happens when copying and pasting from a presentation-oriented medium (PowerPoint, Acrobat, web page, etc.) into the code editor.
- Strip out all white space in the affected area and retype (or retype the whole area)

Missing or garbled serial monitor output

- Ensure that the Serial.begin() parameter value matches the Serial Monitor speed
- Common speeds are 115200 and 9600

End of Hello World Exercise

If everything worked correctly, please turn your cup upside down.

If you’re getting stuck, please ask for help!

Colors, Fonts, & Drawing

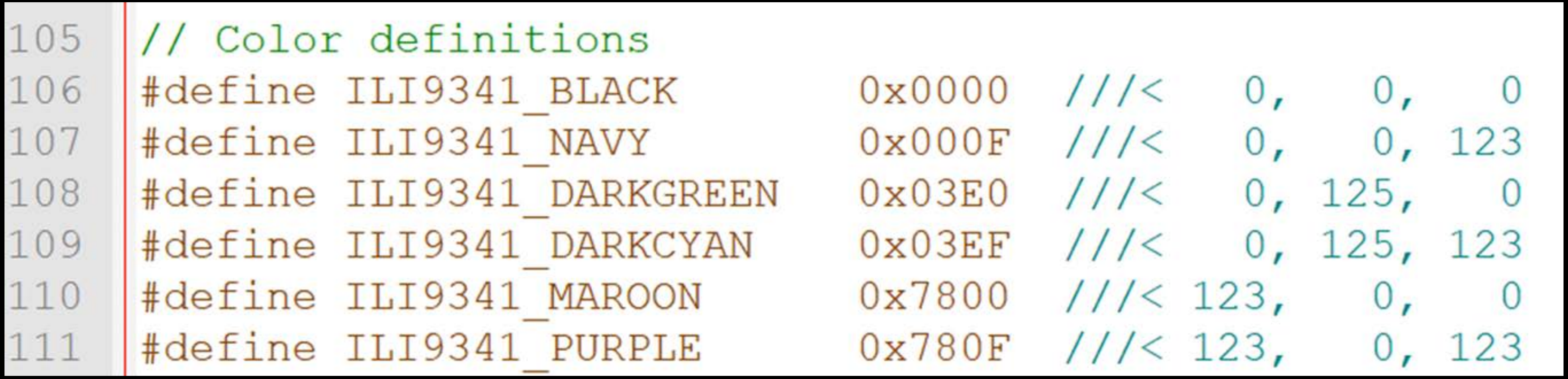
Hey, I’ve Got Something to Say!

Colors

Colors are in RGB 565 format – 16 bits per pixel

R4 : R3 : R2 : R1 : R0 : G5 : G4 : G3 : G2 : G1 :G0 : B4 : B3 : B2 : B1 : B0

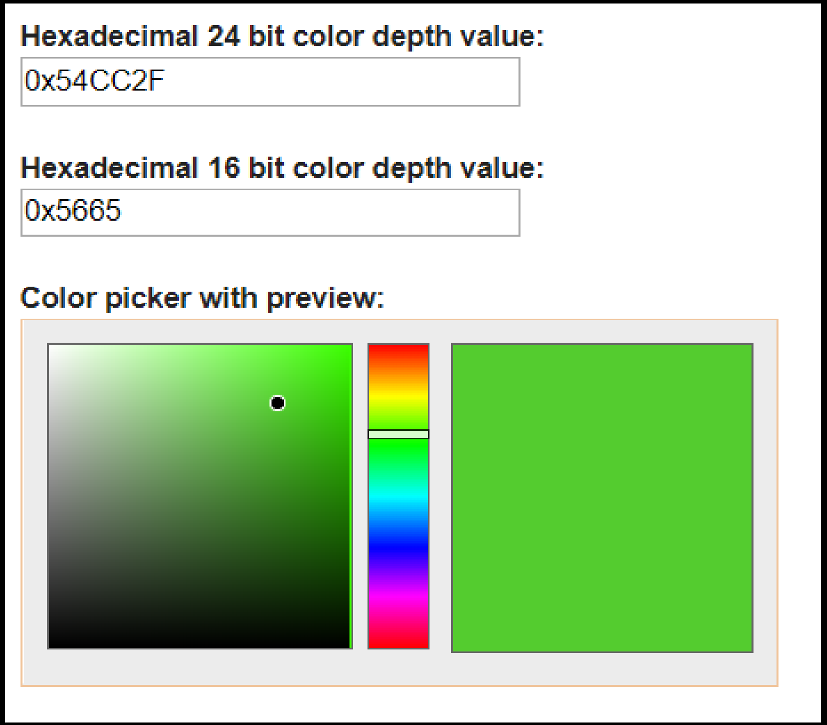
Predefined colors can be found in
Arduino\libraries\Adafruit_ILI9341\Adafruit_ILI9341.h



```
// Color definitions
#define ILI9341_BLACK        0x0000  ///<   0,   0,   0
#define ILI9341_NAVY         0x000F  ///<   0,   0, 123
#define ILI9341_DARKGREEN    0x03E0  ///<   0, 125,   0
#define ILI9341_DARKCYAN     0x03EF  ///<   0, 125, 123
#define ILI9341_MAROON       0x7800  ///< 123,   0,   0
#define ILI9341_PURPLE       0x780F  ///< 123,   0, 123
```

Colors

Online RGB 565 color picker:
https://ee-programming-notepad.blogspot.com/2016/10/16-bit-color-generator-picker.html


Hexadecimal 24 bit color depth value:
0x54CC2F

Hexadecimal 16 bit color depth value:
0x5665

Color picker with preview:

Fonts

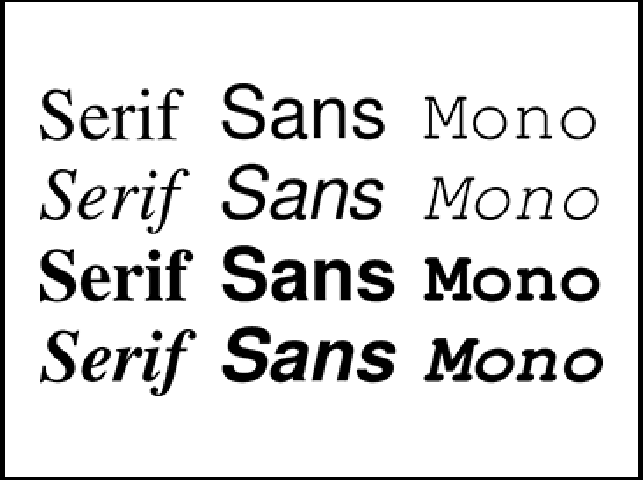
There are over 50 different fonts included with the Adafruit_GFX_Library

To use a font, simply include the header file after the GFX include:

```
#include <Fonts/FreeSansBold24pt7b.h>
```

...and then set it on the TFT:

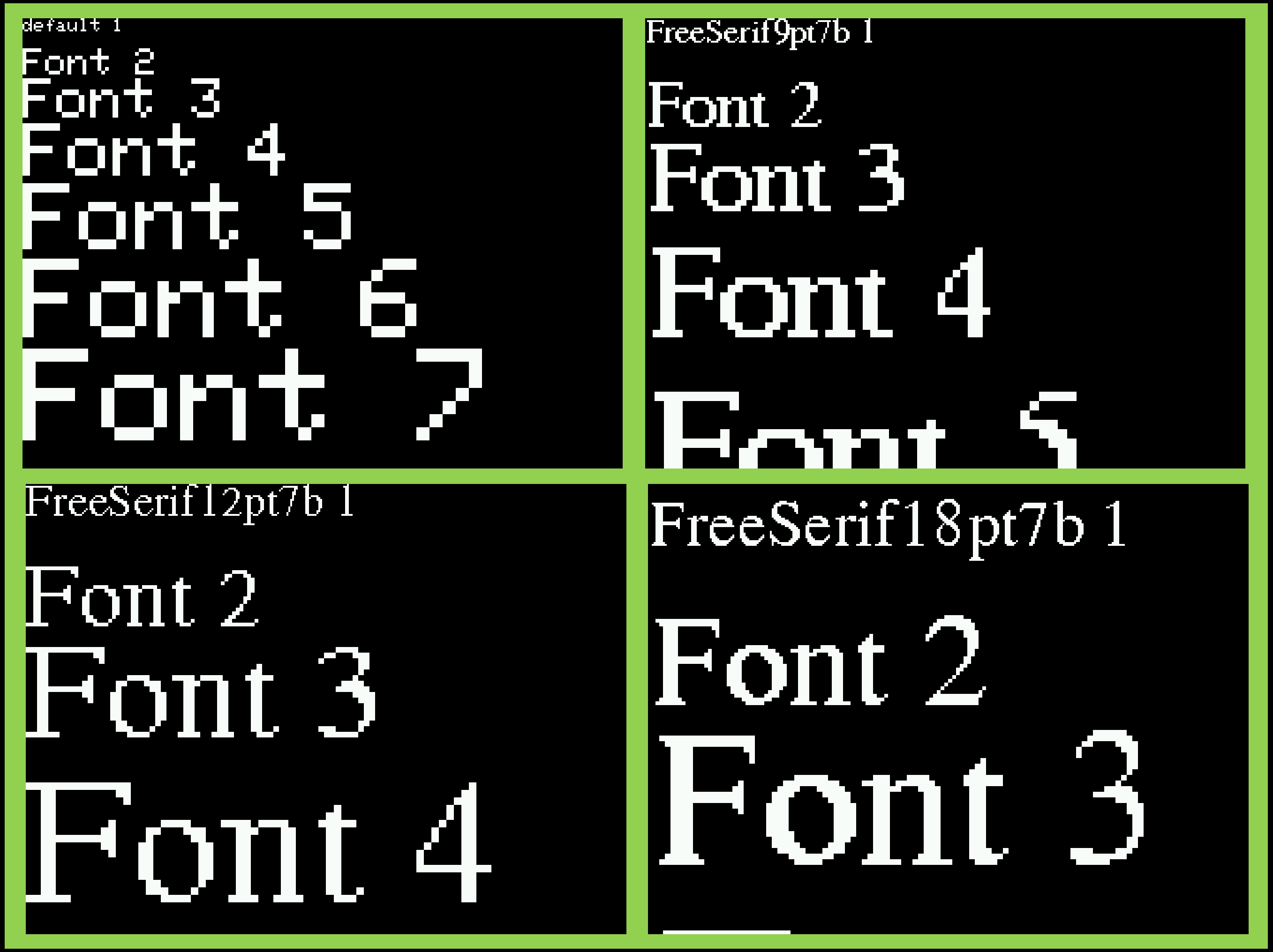
```
tft.setFont(&FreeSansBold24pt7b);
```



Fonts

- The Adafruit GFX Library also contains source for a utility to convert TTF fonts to C header files.
- Each font included increases your program size, so only pull in the fonts you need.
- Use the `tft.setTextSize()` method to set a scale factor for your font (2 is 2x bigger than font size 1, 3 is 3x, etc.)

Fonts - Scaling


default 1
Font 2
Font 3
Font 4
Font 5
Font 6
Font 7

FreeSerif9pt7b 1
Font 2
Font 3
Font 4

FreeSerif12pt7b 1
Font 2
Font 3
Font 4

FreeSerif18pt7b 1
Font 2
Font 3

There isn't any anti-aliasing, so a big font scaled at 1 or 2 will look a lot smoother than a small font scaled by 4 or 5.

Font scalings shown are proportionate to the TFT's 320x240 display.

Drawing

As with most graphics libraries, the Adafruit GFX library provides functions for drawing common shapes, both filled and unfilled:

• Line:	`drawLine()`
• Rectangle:	`drawRect(), fillRect()`
• Round Rectangle:	`drawRoundRect(), fillRoundRect()`
• Circle:	`drawCircle(), fillCircle()`
• Triangle:	`drawTriangle(), fillTriangle()`

See Adafruit_GFX.h for full API

Lab #2
Hello, My Name Is

Without the Sticker

Adding TFT Output

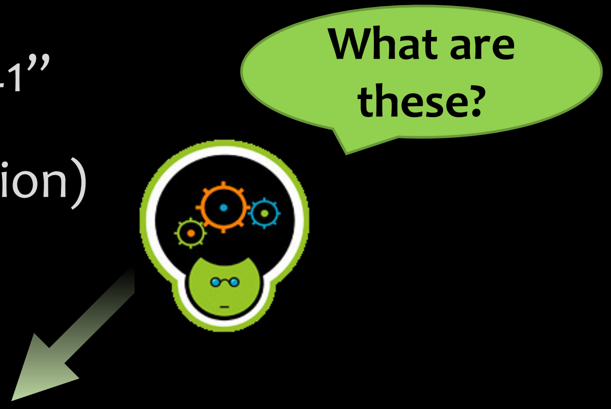
1. Click on Libraries and add “Adafruit GFX” and “Adafruit ILI9341”

2. Add the includes (try typing them to see the intellisense in action)
   ```
   #include <Adafruit_GFX.h>
   #include <Adafruit_ILI9341.h>
   ```

3. Add the TFT constructor before the setup() function:
   ```
   Adafruit_ILI9341 tft = Adafruit_ILI9341(15, 4, 23, 18, 2, 19);
   ```

4. Add the following inside your setup() function:
   ```
   tft.begin();
   tft.fillScreen(ILI9341_BLACK);
   tft.setTextColor(ILI9341_WHITE);
   tft.print(“Your Name");
   ```

5. Upload the sketch. You should see the screen go dark and display your name – though it may not look quite like you’d expect. We’ll fix that shortly.



Troubleshooting

Library not found

- Make sure the library has been installed into your environment
- Use <> around the path for global includes that are not local to your project

Adding Nicer TFT Output

6. Add the following font includes after the ILI9341 library include:

```
#include <Fonts/FreeSansBold24pt7b.h>
#include <Fonts/FreeSansBold9pt7b.h>
#include <Fonts/FreeSansOblique18pt7b.h>
```

7. In setup() , replace the statements after tft.begin() with:

```
tft.setRotation(3);
```

   This will set the TFT rotation to landscape, like the badge.

8. Before setup(), add:

```
const char* my_name  = “Your Name Here";
```

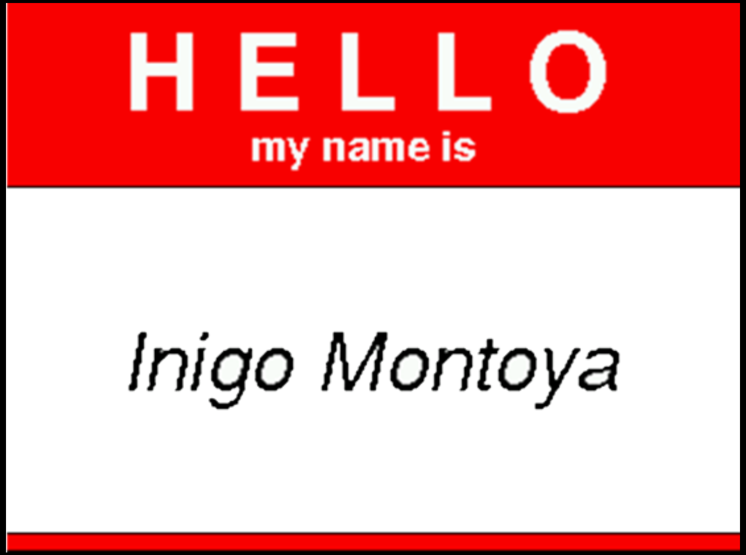
Adding Nicer TFT Output

9. Add the following to setup(), after the existing statements:

```
tft.setFont(&FreeSansBold24pt7b);
tft.setTextColor(ILI9341_WHITE);
tft.setCursor(50, 46);
tft.print("H E L L O");
tft.setFont(&FreeSansBold9pt7b);
tft.setTextColor(ILI9341_WHITE);
tft.setCursor(106, 68);
tft.print("my name is");
int16_t  x1, y1;
uint16_t wd, ht;
tft.setFont(&FreeSansOblique18pt7b);
// Get the size of the displayed name using this font to center
tft.getTextBounds(my_name, 10, 50, &x1, &y1, &wd, &ht);
tft.setCursor(160 - (wd / 2), 120 + ht);
tft.setTextColor(ILI9341_BLACK);
tft.print(my_name);
```

Adding Nicer TFT Output

10. Upload the program. You should see something similar to:



If you get stuck, check out the esp32-badge-hello sketch:

https://github.com/DigiTorus86/ESP32Badge/blob/master/hello-badge/esp32-badge-hello/esp32-badge-hello.ino

Adding Digital Images

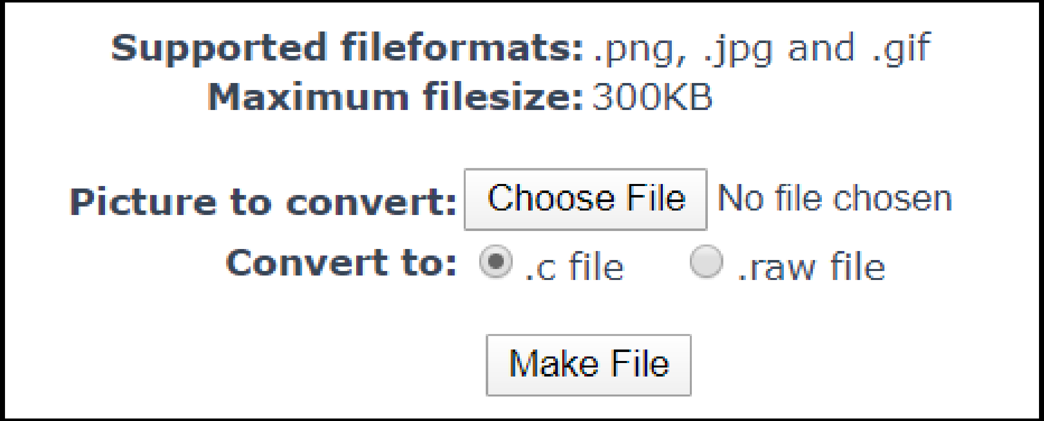
Picture This...

Preparing Images

1. Edit the picture in the image editing software of your choice (GIMP, Paint.NET, PhotoShop, etc.)
2. Final dimensions should not exceed 320 x 240 pixels.
3. Save the image as a PNG or JPG file.
4. Convert the image file to an array in a C include file.

Exporting Images to C - Online

Use an online image conversion utility:
http://rinkydinkelectronics.com/t_imageconverter565.php


Supported fileformats: .png, .jpg and .gif
Maximum filesize: 300KB

Picture to convert: Choose File No file chosen
Convert to: [✓ .c file] .raw file

Make File

Exporting Images to C - Python

Use the img-rgb565.py utility in the scripts folder of the Github repo.

Requires the Pillow Python Image Library. To install:

```
pip install Pillow
```

Usage:

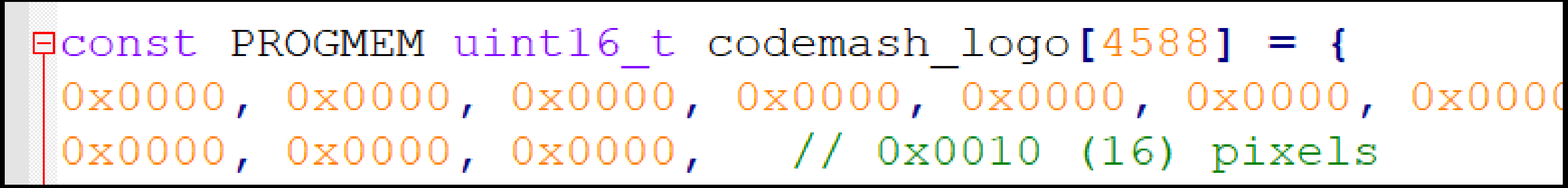
```
python img-rgb565.py mypicture.png
```

Replace with your file and path.

A new “mypicture.h” file will be created in the same folder.

Displaying Bitmaps

Use: `const PROGMEM uint16_t` *`<resource name>`*`[`*`<size>`*`] = {` *`<hex array data>`* `};`



```
const PROGMEM uint16_t codemash_logo[4588] = {
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x000
0x0000, 0x0000, 0x0000,   // 0x0010 (16) pixels
```

Use the **drawRGBBitmap()** function to display the bitmap image:

```
tft.drawRGBBitmap(x,
                  y,
                  (uint16_t *)bitmap_var,
                  width,
                  height);
```

Lab #3
Add Your Picture

Selfie sticks at the ready…

Get an Image to Edit

1. Take a selfie using your laptop camera or cellphone, or select some other image you'd like to use and open it in your image editing software.
2. Crop and/or scale the image to approximately 120 x 120 pixels.
3. Save the new image as a JPG or PNG file.
4. Convert the image to C using either an online utility or Python script. http://rinkydinkelectronics.com/t_imageconverter565.php
5. Copy the result file to your Hello_Badge sketch folder or create a new include file (tab) in your sketch and add the C image code to it.

Get an Image to Edit

6. Add the include below the others in your main file, adjusting the filename as needed:
   ```
   #include “mypicture.h”
   ```

7. Add the call to display your picture at the end of setup():

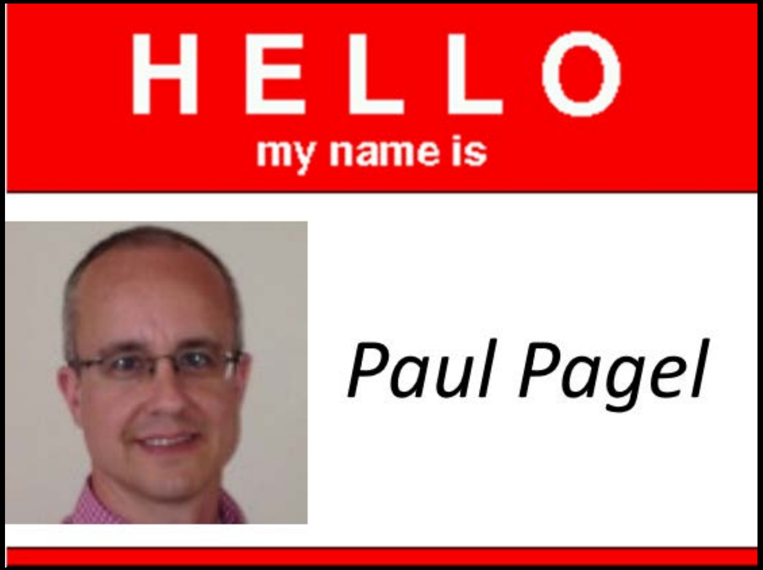
   ```
   tft.drawRGBBitmap(0,  100,  &mypicture, 120, 120);
   ```

8. Adjust the name display cursor x coordinate:
   ```
   tft.setCursor(220 - (wd / 2), 120 + ht);
   tft.setTextColor(ILI9341_BLACK);
   tft.print(my_name);
   ```

9. If you have a long name, you may want to just use your first name or break it into two different print() calls.

Adding Nicer TFT Output

10. Upload the program. You should see something similar to:



If you get stuck, check out the esp32-badge-hello-pic sketch:
https://github.com/DigiTorus86/ESP32Badge/tree/master/hello-badge/esp32-badge-hello-pic

GPIO Pins

The Ins and Outs

GPIO Pins

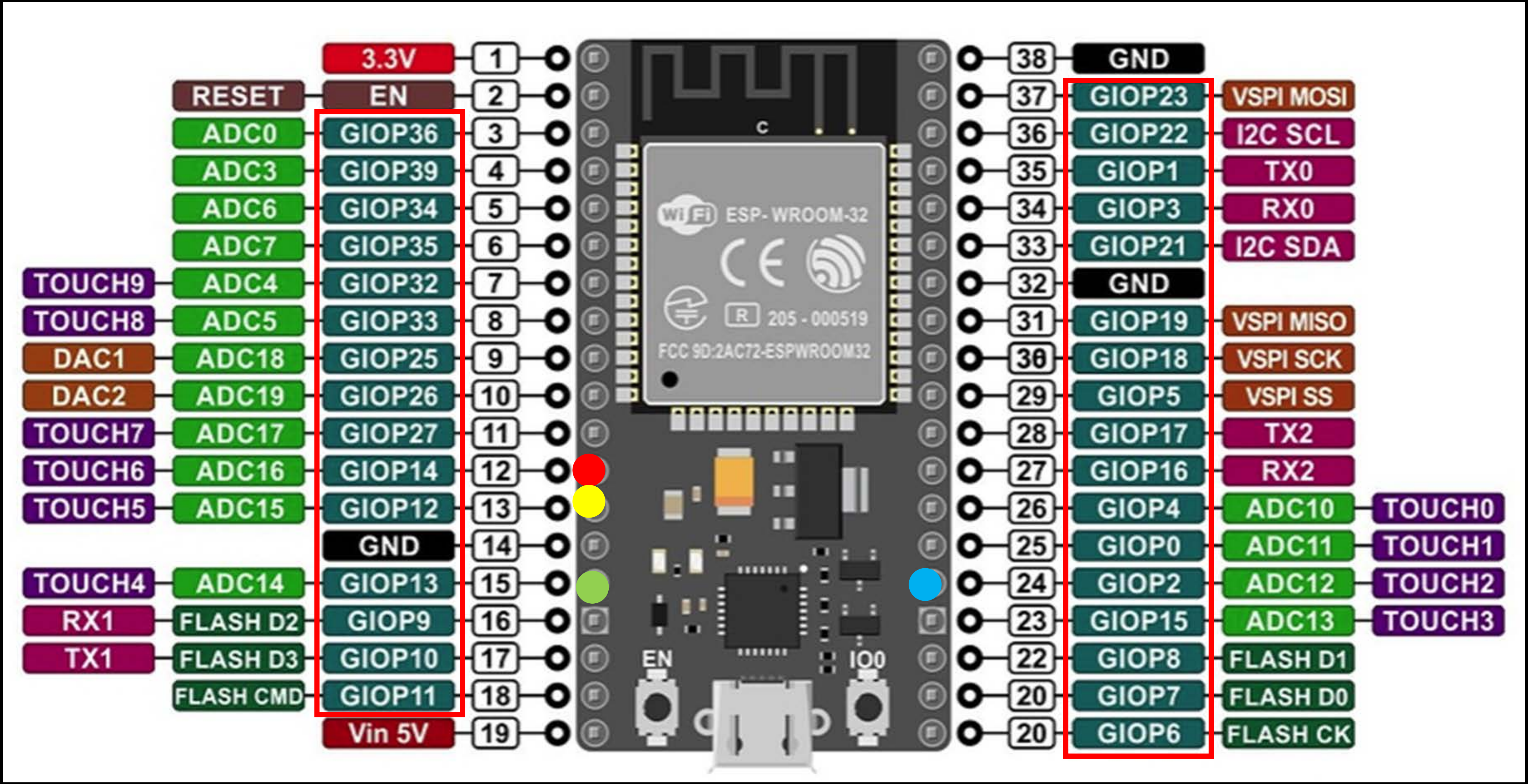
General Purpose Input Output Pins allow the MCU to read and write from connected switches, sensors, and components.

GPIO pins should usually be designated using the pinMode() function in your setup() as:

• OUTPUT	set the output voltage, i.e. turning on a light
• INPUT	read the input voltage, i.e. reading a sensor value
• INPUT_PULLUP	read the input voltage, but use the MCU’s internal pullup resistors

Example:

```
pinMode(12, OUTPUT);
```


3.3V
1
RESET
EN
2
ADC0
GIOP36
3
ADC3
GIOP39
4
ADC6
GIOP34
5
ADC7
GIOP35
6
TOUCH9
ADC4
GIOP32
7
TOUCH8
ADC5
GIOP33
8
DAC1
ADC18
GIOP25
9
DAC2
ADC19
GIOP26
10
TOUCH7
ADC17
GIOP27
11
TOUCH6
ADC16
GIOP14
12
TOUCH5
ADC15
GIOP12
13
GND
14
TOUCH4
ADC14
GIOP13
15
RX1
FLASH D2
GIOP9
16
TX1
FLASH D3
GIOP10
17
FLASH CMD
GIOP11
18
Vin 5V
19
38
GND
37
GIOP23
VSPI MOSI
36
GIOP22
I2C SCL
35
GIOP1
TX0
34
GIOP3
RX0
33
GIOP21
I2C SDA
32
GND
31
GIOP19
VSPI MISO
30
GIOP18
VSPI SCK
29
GIOP5
VSPI SS
28
GIOP17
TX2
27
GIOP16
RX2
26
GIOP4
ADC10
TOUCH0
25
GIOP0
ADC11
TOUCH1
24
GIOP2
ADC12
TOUCH2
23
GIOP15
ADC13
TOUCH3
22
GIOP8
FLASH D1
20
GIOP7
FLASH D0
20
GIOP6
FLASH CK
ESP-WROOM-32
205 - 000519
FCC 9D:2AC72-ESPWROOM32
EN
IO0

GPIO Pins

The digitalWrite() function sets the value of a pin:

```
digitalWrite(12, HIGH);
```

The digitalRead() gets the current value of a pin:

```
int val = 0;
val = digitalRead(36);
```

When set as INPUT_PULLUP, a pin will read HIGH when the connected switch or button is open, and LOW when it is closed/pressed.

```
// evaluates True if button on GPIO36 is pressed
if(digitalRead(36) == LOW)
```

Lab #4
Adding LEDs and Buttons

Let there be lights!

Meet the LEDs

1. Add the following at the end of setup():

```
pinMode(14, OUTPUT);  // red
pinMode(12, OUTPUT);  // yellow
pinMode(13, OUTPUT);  // green
```

2. Inside loop(), add:

```
digitalWrite(14, HIGH);
digitalWrite(12, HIGH);
digitalWrite(13, HIGH);
delay(250);
digitalWrite(14, LOW);
digitalWrite(12, LOW);
digitalWrite(13, LOW);
delay(250);
```

Meet the LEDs

3. Upload the program. The 3 badge LEDs should now be blinking on and off every quarter second.

Pushing Some Buttons

4. Add the following at the end of setup():
```
pinMode(36, INPUT_PULLUP); // Up button
pinMode(39, INPUT_PULLUP); // Down button
```

5. Add the following at the start of loop():
```
static uint16_t delay_ms = 250;
if(digitalRead(36) == LOW) // up pressed
   delay_ms = (delay_ms > 100 ? delay_ms - 100 : 10);
if(digitalRead(39) == LOW) // down pressed
   delay_ms = (delay_ms < 5000 ? delay_ms + 100 : 5000);
```

6. Change both delay(250) to:
```
   delay(delay_ms);
```

Pushing Some Buttons

7. Upload the program. The 3 badge LEDs should still be blinking on and off every quarter second.
8. Press the Up button on the badge. The LEDs should blink faster, eventually just appearing to stay on.
9. Press the Down button. The LEDs should blink slower.

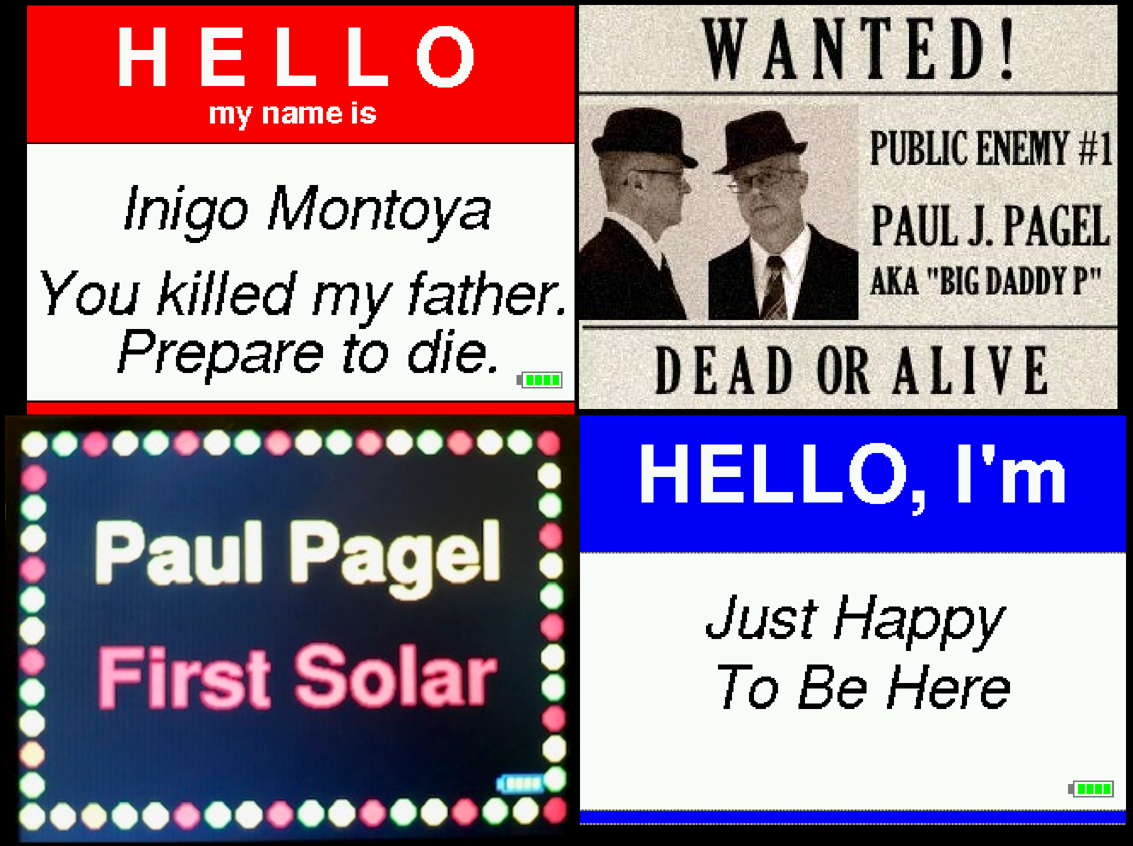
If you get stuck, check out the esp32-badge-hello-btn sketch:

https://github.com/DigiTorus86/ESP32Badge/tree/master/hello-badge/esp32-badge-hello-btn

Pre-made Badge Apps

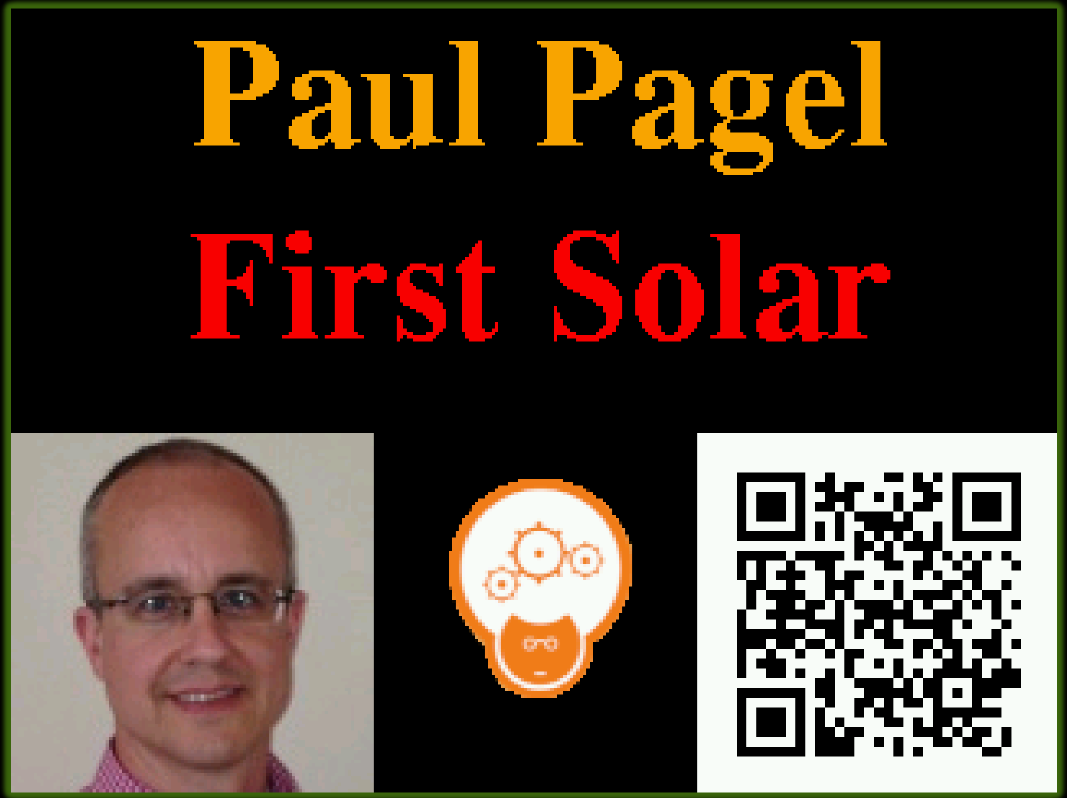
What’s on the Menu

Basic Bro’ Badge



- Simple and direct, this is the simplest and fastest way to generate a personalized badge.
- No multimedia or programming skills required.
- Contains a number of different display modes that are selectable at runtime

The QR Code Badge



- Dynamically-generated QR code contains the network web address for the ESP32.
- Scanning it with your smartphone will open the web page generated by the ESP32

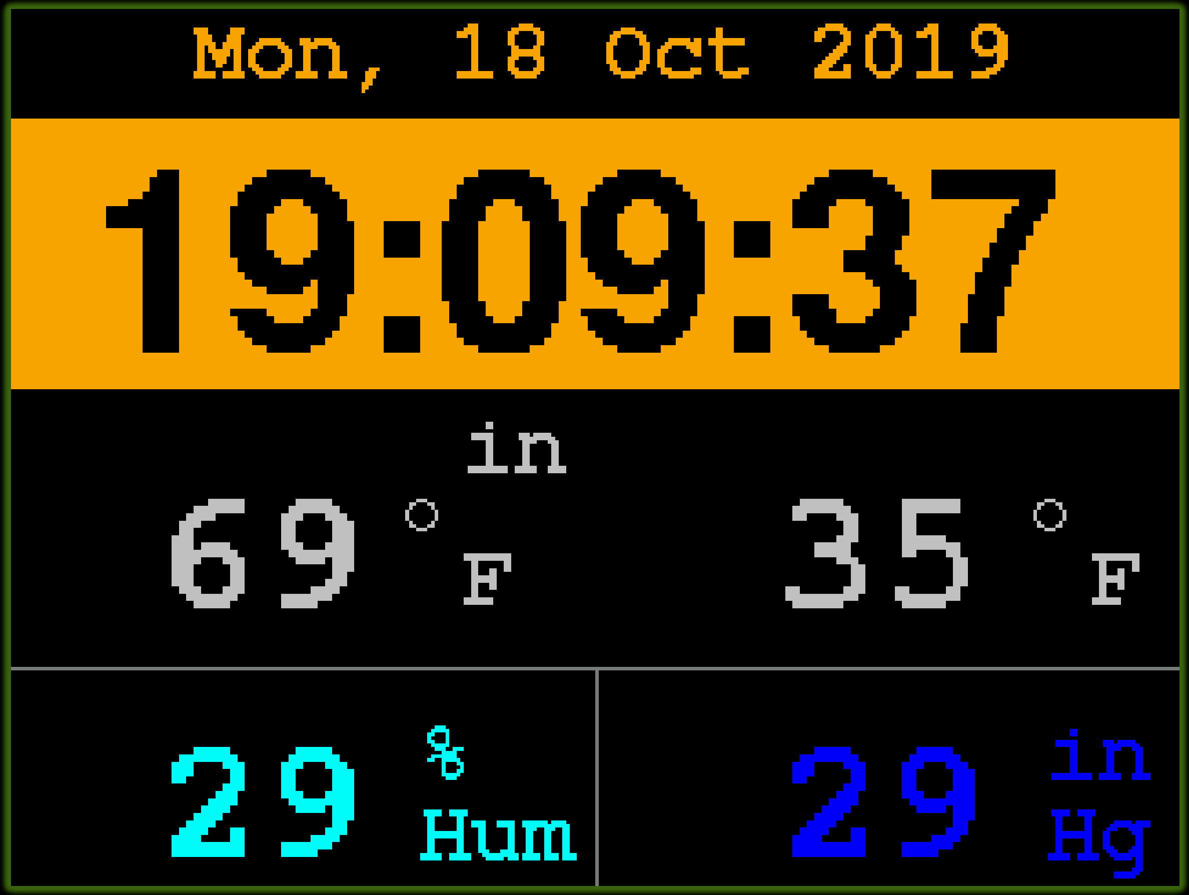
  (This assumes the smartphone is on the same WiFi network)
- Larger QR code displayed when the B button is pressed.

The Escaped Wizard

HAVE YOU SEEN THIS WIZARD?
Approach with Extreme Caution!
Do Not Attempt to Use Magic Against This Man!

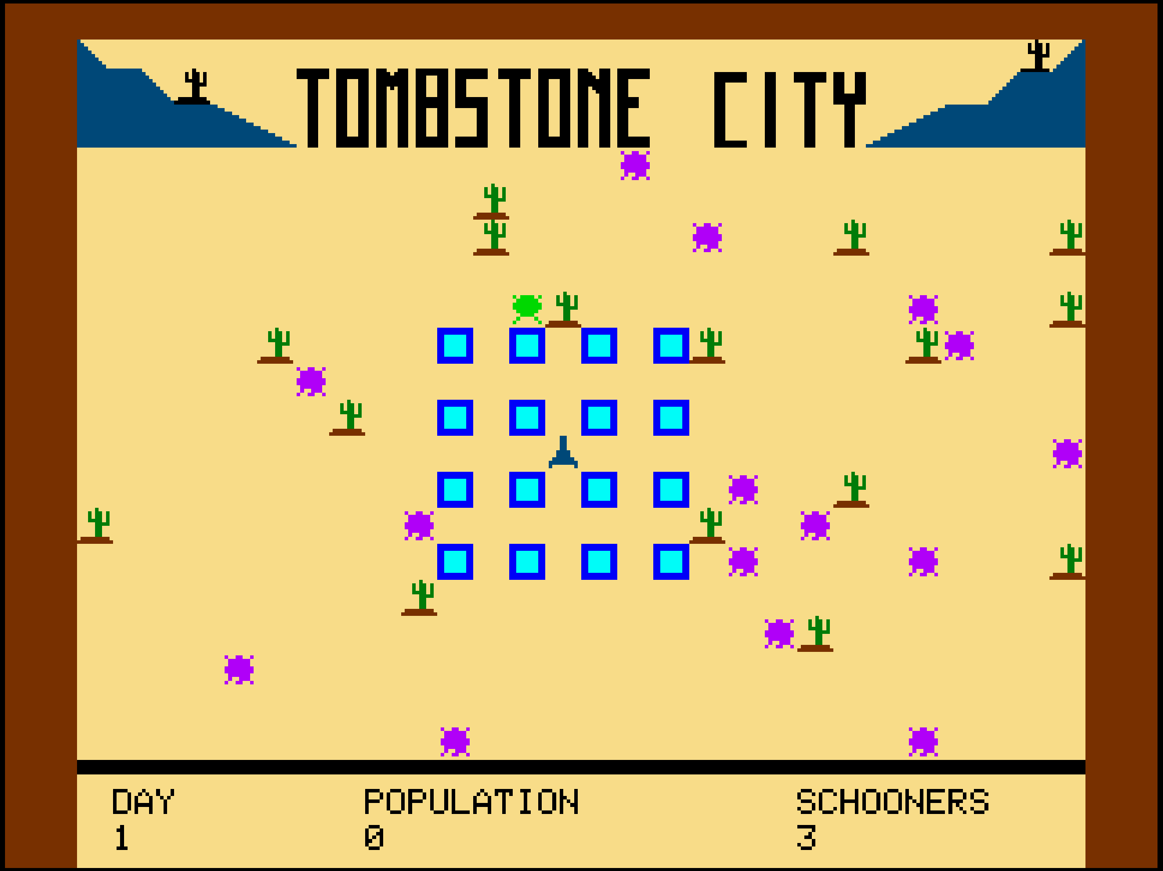
- Displays a series of time-lapse photos to generate an animation effect
- Some photo-editing skills and software required.

Time Lord Badge



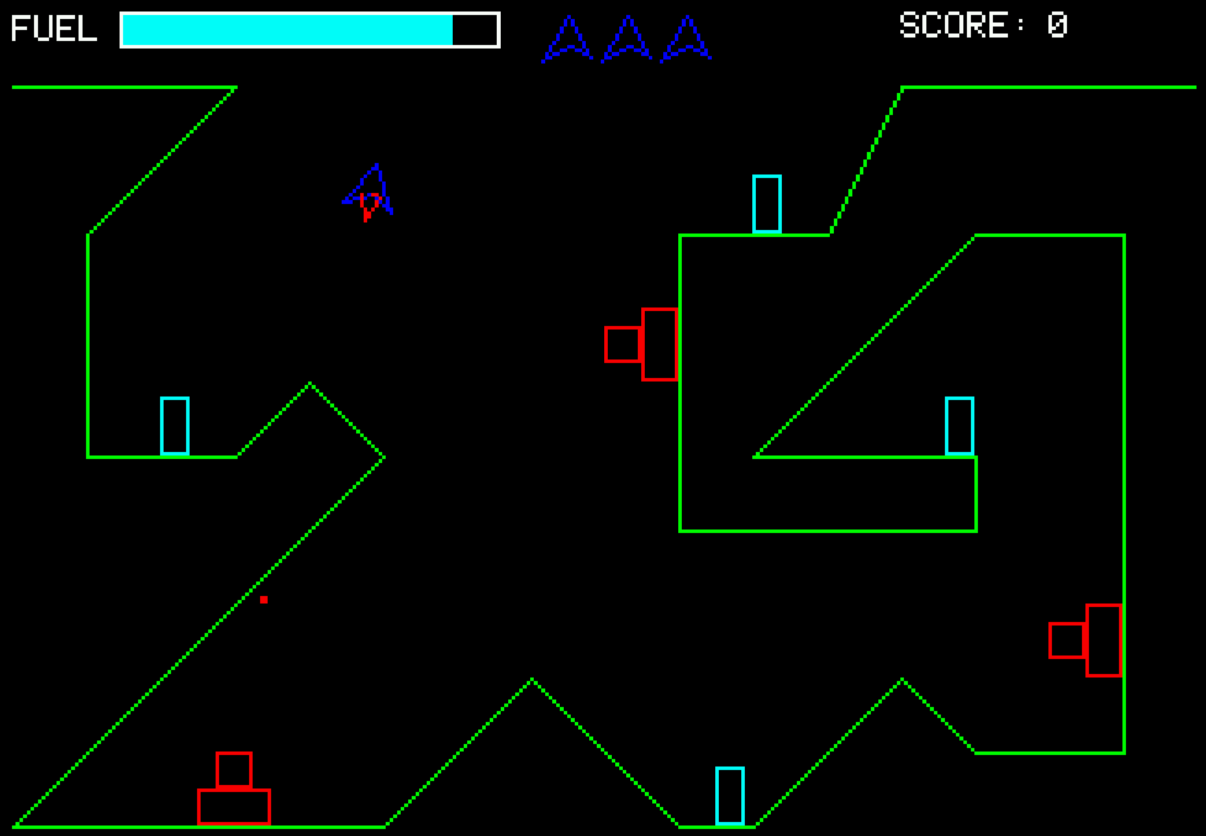
- Displays current date, time, and weather conditions.
- If no sensors available, uses online forecast info.
- Uses ESP32's onboard Real Time Clock (RTC)
- Syncs time via Network Time Protocol (NTP)
- Requires additional hardware for full features.

Tombstone City Game



- An homage to the classic Texas Instruments game created in 1981 by John C. Plaster
- Uses PWM audio for the simple sound effects and melodies

Gravitack Game



- Inspired by the color vector coin-op Gravitar released by Atari in 1982
- Uses 8 bit WAV files for the sound effects
- Comes with 6 levels

Badge Framework

A simple starter sketch (esp-32-badge-framework-only) that provides:

- The library includes needed for the TFT
- GPIO Pin usage definitions
- TFT and GPIO setup code
- Main loop that maps button pin states to “pressed” and “released” variables

Lab #5
The Basic Bro

Are you basic?

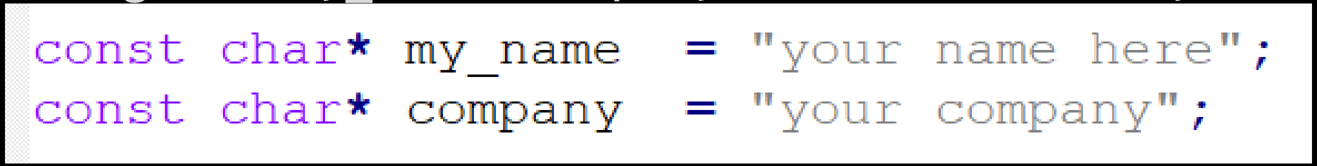
Get the Code

If you haven’t already done so, download the badge code repo from GitHub to your laptop:

https://github.com/DigiTorus86/ESP32Badge

The Basic Bro Badge

1. Load the **esp32-badge-basic** sketch into the IDE
2. Build it without making any changes. This will verify that you have the needed libraries in place.
3. Change the my_name/company values to whatever you want



```
const char* my_name   = "your name here";
const char* company   = "your company";
```

4. Upload the sketch to your badge and verify that it loads and runs correctly.
5. Use the X and Y buttons to cycle through the badge display modes.

The Basic Bro Badge

6. Make some customizations:
   - Open HelloRotate.h and edit the message values. The wittier the better!
   - Change the _COLOR_HEADER values in Hello.h and HelloRotate.h to your favorite colors.

7. Create a 320 x 240 graphic to replace the picture in gangster.h.
   Follow the same steps as the previous lab for image conversion.

8. Upload and verify everything still works.

HTTP Client / Server

Putting the “Hyper” in Hypertext

HTTP Client

- ESP32 client library supports standard HTTP request methods (GET, POST, PUT, PATCH)
- Supports HTTPS requests.
- HTTPClient library: https://github.com/espressif/arduino-esp32/blob/master/libraries/HTTPClient/src/HTTPClient.h
- Used in the ESP32 Badge Time and Temperature sketch.
- Recognize that running the WiFi and Bluetooth radios causes the ESP32 to consume more power.

HTTP Server

- ESP32 library provides lightweight HTTP server capabilities.
- Note that running an HTTP server continuously will drain your battery.
- WiFiServer
  https://github.com/espressif/arduino-esp32/tree/master/libraries/WiFi/src
- Useful for serving up content from the ESP32, such as sensor data or camera images.
- Also useful for configuring and controlling the ESP32.

Lab #6
QR Code & Time Badges

With HTTP Server Control

QR Code Badge

1. Ensure your phone is charged and on the CodeMash WiFi.

2. Load the esp32-badge-qrcode sketch.

3. Enter the appropriate values for name, company, and WiFi in the main file:
```
const char* my_name  = "";
const char* company  = "";
const char* ssid     = "";
const char* password = "";
```

4. Optional: replace headshot.h with your 120x120-ish picture created in the previous lab. In drawBadge(), adjust the dimensions or name of the bitmap as needed:

```
tft.drawRGBBitmap(0, 130, (uint16_t *)headshot, 111, 111);
```

QR Code Badge

5. Upload the app to your badge, and wait for the yellow LED to turn on, then off – indicating that the badge is connected to WiFi.
6. Press the B button to display the large QR code (smaller one may work, but is more subject to lighting, reflection, and camera issues.)
7. Use the camera or QR scan app on your phone to scan the QR code.
8. Use the button displayed in the webpage on your phone to turn the green LED on and off.

IDEA: Create a new version of the sketch with the QR code containing the URL for your personal or company website.

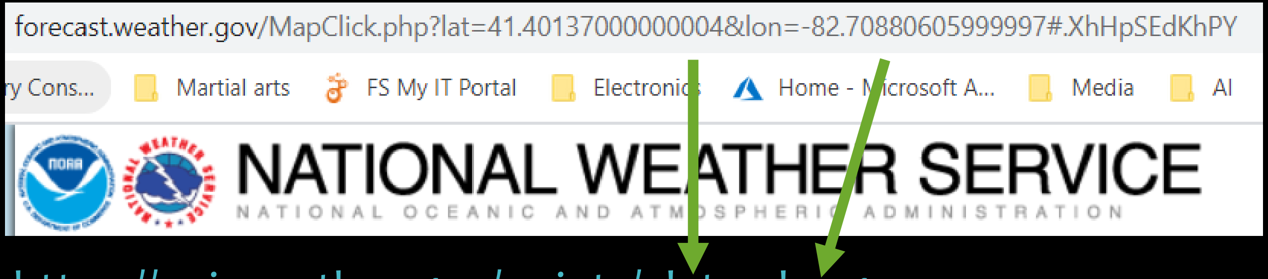
Time and Weather Badge

1. Load the esp32-badge-time-weather sketch.
2. Enter the appropriate values for WiFi in the main file:
   ```
   const char* ssid     = "";
   const char* password = "";
   ```
3. Upload sketch to verify that the badge is able to get on the WiFi and retrieve the current time and weather for Sandusky.

Weather - Customized Forecast

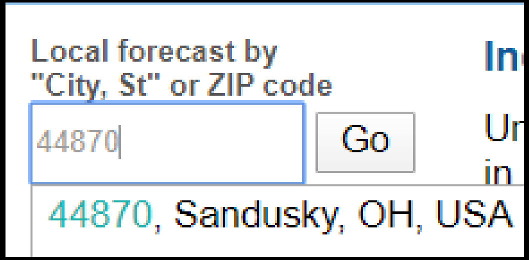
OPTIONAL: Customizing the Forecast API URL
https://www.weather.gov/documentation/services-web-api

1. Plug in your zipcode and select the city if multiple results
2. Copy the latitude and longitude from the result URL to the one below:


forecast.weather.gov/MapClick.php?lat=41.40137000000004&lon=-82.70880605999997#.XhHpSEdKhPY

https://api.weather.gov/points/<lat>,<long>

3. Copy the URL value in the properties->forecast node of the JSON response
from the points URL and put this in your forecast_api_url variable assignment

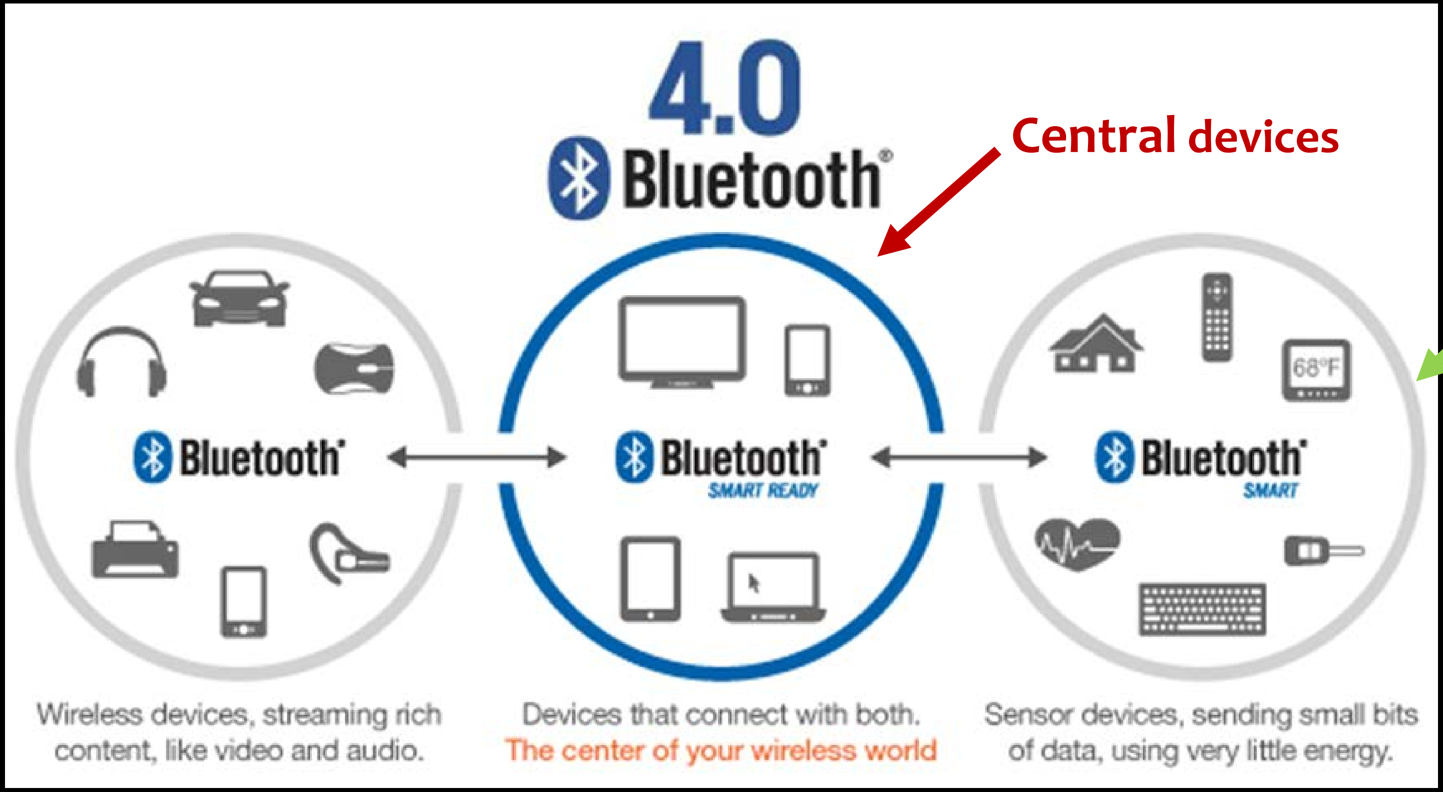

Local forecast by "City, St" or ZIP code
44870

44870, Sandusky, OH, USA

Bluetooth Low Energy

On the BLEeding Edge

Bluetooth Low Energy (BLE)

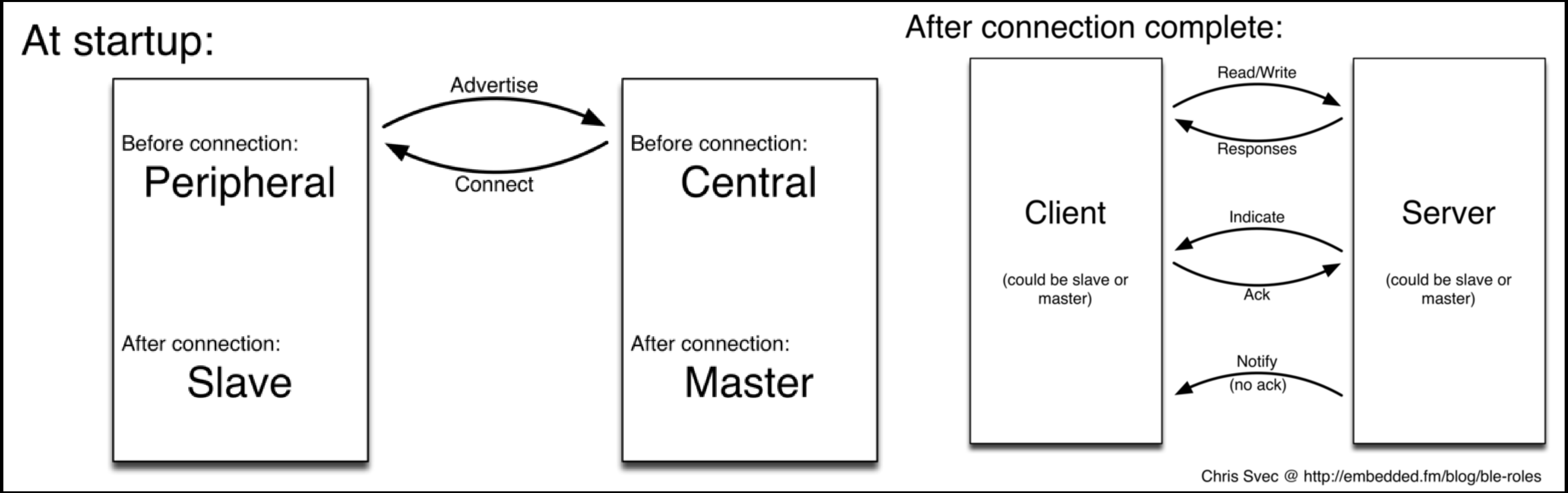

4.0
Bluetooth
Central devices
Bluetooth
Bluetooth SMART READY
Bluetooth SMART
68°F
Wireless devices, streaming rich content, like video and audio.
Devices that connect with both. The center of your wireless world
Sensor devices, sending small bits of data, using very little energy.

Peripheral devices

Peripherals pair with only 1 central device at a time

- BLE is a lightweight subset of the Bluetooth classic specification
- Highly interoperable with mobile computing platforms

BLE Roles



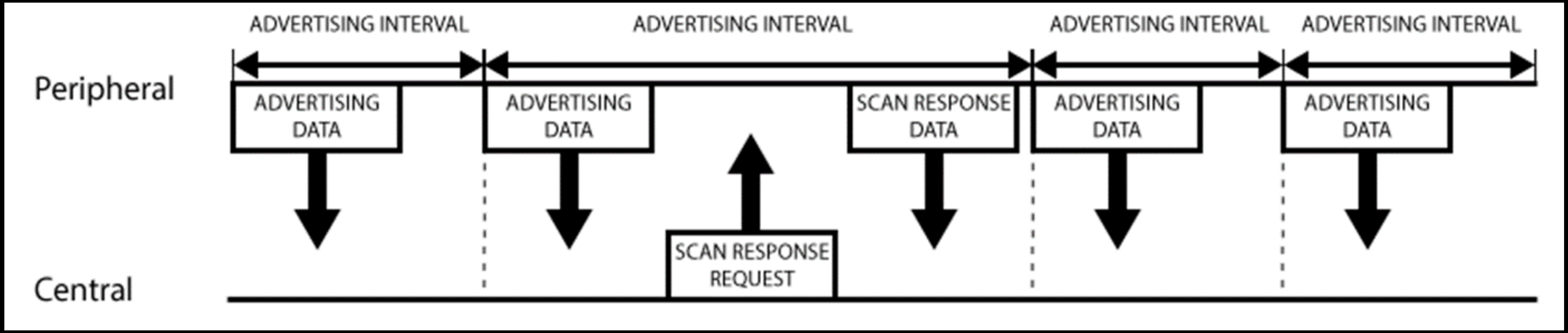
The ESP32 can play any of the roles shown.

https://embedded.fm/blog/ble-roles

BLE – Terms and Concepts

GAP – Generic Access Profile

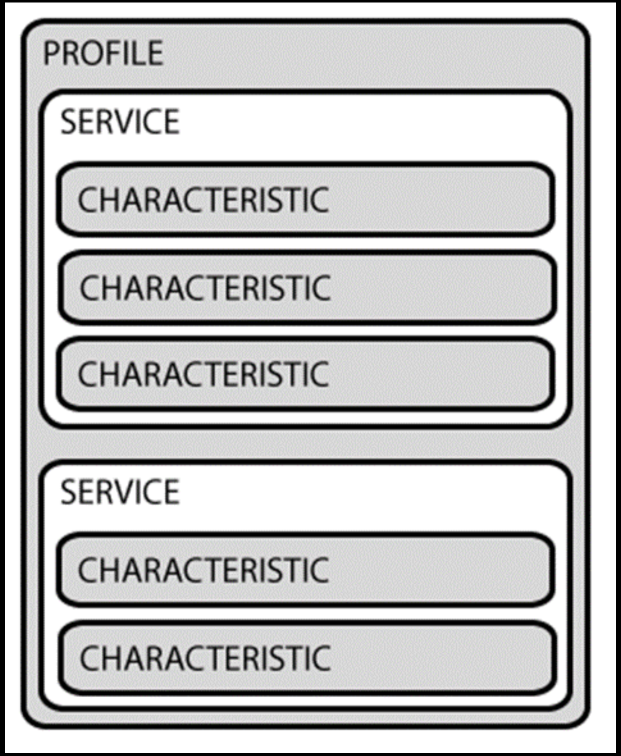
- Controls advertising (visibility) and connection (interaction) of Bluetooth devices
- Server sends out advertising “beacons” on a configurable interval
- Sends back additional data when requested by a central “client” device



https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction

BLE – Terms and Concepts

- GATT specifies the structure in which profile data is exchanged
- Service: logical group of characteristics with a Universally Unique ID (UUID)
  - Can be standard/predefined or custom
- Characteristic: single datapoint with a UUID
  - Can be read from or written to
  - Can be mandatory or optional
  - Also encapsulates metadata (security, representation, etc.)
- Master device = central, often GATT Client
- Slave device = peripheral, often GATT Server



https://learn.adafruit.com/introduction-to-bluetooth-low-energy/introduction

BLE Development

- ESP32 uses several different BLExxx.h libraries for implementing BLE client/server functionality
- Server Mode Process:
  - Create server
  - Add Characteristic
  - Start Advertising
  - Handle connect/disconnect events in callback functions

- May be useful to install a BLE app on your smartphone for verification and debugging purposes.
- Client Mode Process
  - Create client
  - Scan for server devices
  - Pair with server
  - Handle connect/disconnect/data received events in callback functions

LightBlue® Explorer 4+
The go-to BLE development tool
Punch Through
★★★★☆ 4.3, 148 Ratings
Free

Adafruit Bluefruit LE Connect 4+
Adafruit Industries
★★★★☆ 4.3, 6 Ratings
Free

Lab #7
BLE Remote Control

Robot Not Included

BLE Remote Control

1. Talk to your partner(s) and decide who is going to be “peripheral” and who is going to be “central” for the first iteration. Everyone should get a chance to do both roles.

   Peripheral is the badge acting as the controller (just like an Xbox™ controller)

   Central is the badge being controlled (like your Xbox™ console)

   Execute instructions 2-4 on <u>one</u> of the following two slides depending on your role.

BLE Remote Control - Peripheral

2. Open the esp32-badge-ble-peripheral sketch and upload it to your badge.
3. The screen should look very similar to the hardware test app that was pre-loaded on your ESP32. However, there should be a 6-byte address line in orange at the bottom.

   ADDR: xx:xx:xx:xx:xx:xx

4. Copy the address to a piece of paper or show the screen to your central partner when they are ready to configure the central sketch. Include the colon separators.

BLE Remote Control - Central

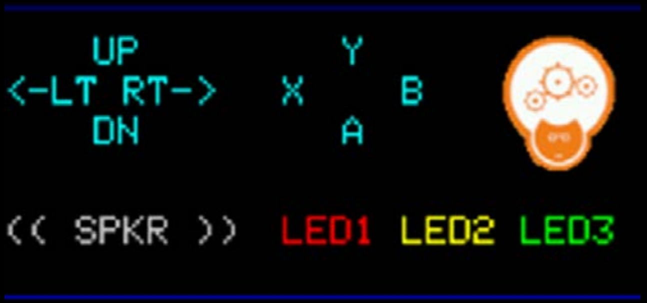
2. Open the esp32-badge-ble-central sketch.
3. Get the MAC address from your peripheral partner’s display and change the value of the address variable at the top of the sketch to match it:

   ```
   String esp32_peripheral_address = “enter address here";
   ```

4. Upload the sketch to your badge.

BLE Remote Control

5. The central badge should initially display list of detected BLE devices.
6. When/if it finds the configured address, it should pair – at which point the green light on both badges should come on. The yellow LED will pulse when the badges send/receive data.
7. Start pushing puttons on the peripheral badge. The corresponding indicator on the central badge display should change accordingly.



```
    UP          Y
<-LT RT->    X     B
    DN          A

<< SPKR >>  LED1 LED2 LED3
```

8. Switch roles and repeat steps 2-7 until everyone has had a turn with each role.

Adding Sound

Preparing Audio

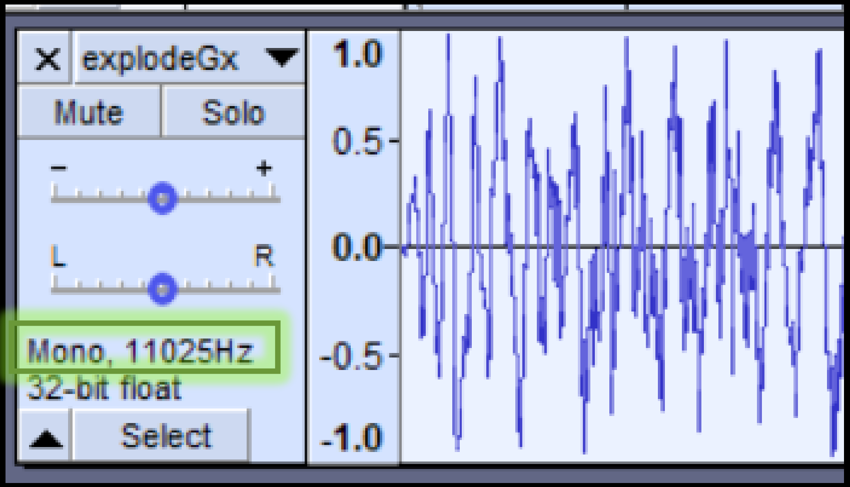
Export audio file as a mono 8bit PCM at a sample rate of 44100 Hz or less.

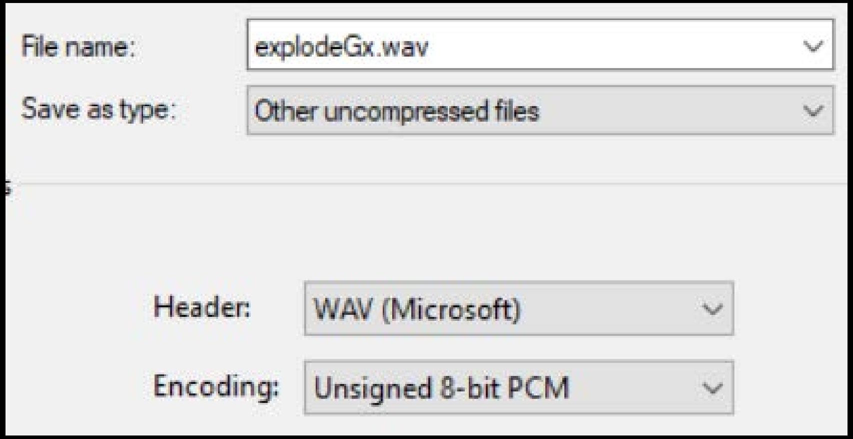
Suggest **11025 Hz** to decrease size since the badge speaker is not high fidelity.

Use: **Tracks->Resample** menu option

File->Export as 8-bit WAV.

- Save as type: other uncompressed files.
- Header: WAV (Microsoft)
- Encoding: Unsigned 8-bit PCM




File name: explodeGx.wav
Save as type: Other uncompressed files
Header: WAV (Microsoft)
Encoding: Unsigned 8-bit PCM

Preparing Audio

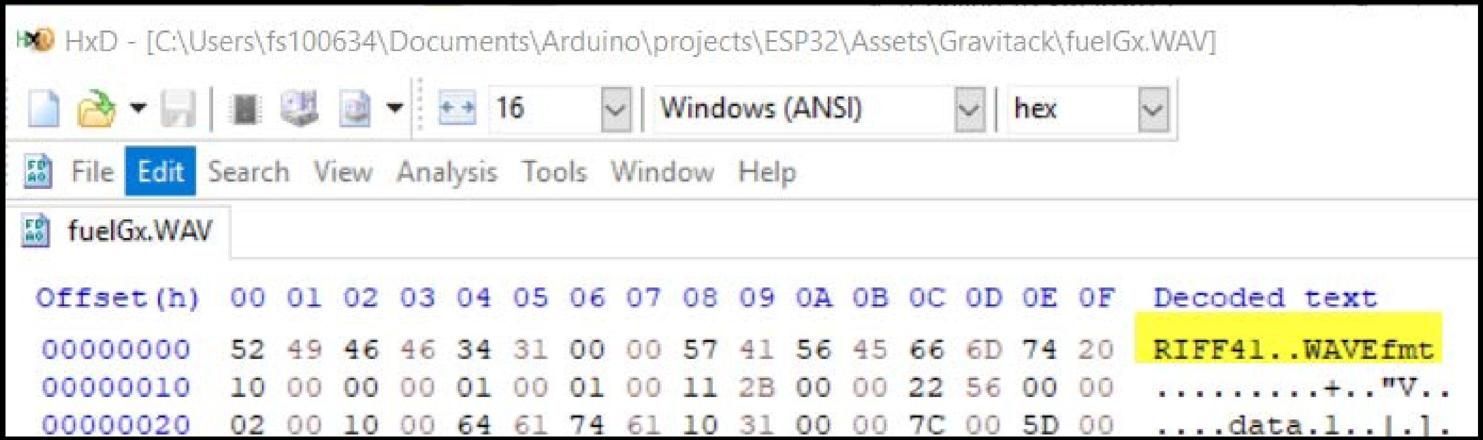
Audio Sample Editing Tips

- SHORT: Trim samples to shortest possible length
- LOUD: Adjust samples to be as loud as possible without clipping
- FREQ: Keep frequencies in the mid to upper range (no bass)

Preparing Audio – App Export to C

Use HxD or similar utility to export the audio file as C code

1. Open the WAV file in HxD and verify RIFF/WAVfmt as shown below
2. File -> Export -> C
3. Rename file to .h and include in your sketch
4. Use: `const PROGMEM unsigned uint8_t` *`<resource name>`*`[`*`<size>`*`] =` `{` *`<hex array data>`* `};`



HxD - [C:\Users\fs100634\Documents\Arduino\projects\ESP32\Assets\Gravitack\fuelGx.WAV]

16 | Windows (ANSI) | hex

```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000000  52 49 46 46 34 31 00 00 57 41 56 45 66 6D 74 20  RIFF41..WAVEfmt
00000010  10 00 00 00 01 00 01 00 11 2B 00 00 22 56 00 00  .........+.."V..
00000020  02 00 10 00 64 61 74 61 10 31 00 00 7C 00 5D 00  ....data.1..|.].
```

Preparing Audio – Script Export to C

Use the scripts\wav-pcm8.py script to generate the .h files.

Usage:

```
python wav-pcm8.py myaudio.wav
```

The file “myaudio_wav.h” will be created in the same folder.

Playing Digital Audio

To begin playing a digital audio clip:

- Download and install the XT DAC Audio library (v4.2 or higher)
  https://www.xtronical.com/the-dacaudio-library-download-and-installation/
- Include the library and the wave header file

```
#include "XT_DAC_Audio.h";
#include “myaudio_wav.h"
```

- Call the **Play()** function to initiate the sound

```
DacAudio.Play(&myaudio_wav);
```

Fill the audio buffer periodically to continue playing the sound:

```
DacAudio.FillBuffer();
```

Lab #8
WAV Player

Lets make some noise!

WAV Player

1. Download and install the XT DAC Audio library (v4.2 or higher)
   https://www.xtronical.com/the-dacaudio-library-download-and-installation/
2. Open the esp32-badge-wav-play sketch from the hello-badge folder.
3. Upload to your badge.
4. Press the B button and verify you can hear the sound clip. You may need to rotate the trimpot counterclockwise to increase volume.
5. Download or record a WAV file for your badge.
6. Follow the steps from the previous slides to save the WAV file as an 8-bit 11KHz mono PCM sample.
7. Export the file to a C array using either the HxD app or the wav-pm8.py script and add it to your sketch folder.

WAV Player

8. Add the include directive for your sound file.
9. Add the XT_Wav_Class player declaration for this sample:

```
XT_Wav_Class mywav_play(mywav_wav);
```

Replace with your wav array variable name.

10. Add the following to loop() after btnA_pressed = true:

```
DacAudio.Play(&mywav_play);
```

11. Modify the while condition in loop():

```
while (audio_play.Playing || mywav_play.Playing)
```

12. Upload the sketch and verify the sound is played when the A button is pressed.

Ideas and Next Steps

Some of the ideas might even be good ones…

Topics

Sensors

ADC/DAC

Common Protocols

- I2C
- SPI
- UART

Motor Control

SD Cards and Data Storage

Project-Driven Discovery

Find something you want to do (real-world application)

…something that excites you

…something that you can accomplish in a 2-3 week sprint

…and figure out how to do it.

CodeMash Badge Contest

Categories

- Laugh Out Loud
- Artistic Achievement
- Technical Wizardry

Bring submissions to the Maker Space before noon on Friday for consideration. CodeMash staff will take a short video, along with your name(s) and email. Must be present Friday afternoon to win.

Questions

Resources

AdaFruit GFX Library
https://learn.adafruit.com/adafruit-gfx-graphics-library/overview

565 Image Converter
http://rinkydinkelectronics.com/t_imageconverter565.php

Online Color Picker
https://ee-programming-notepad.blogspot.com/2016/10/16-bit-color-generator-picker.html

Contact Info for Paul Pagel:

- Email: pjpagel86@gmail.com

Thank You!