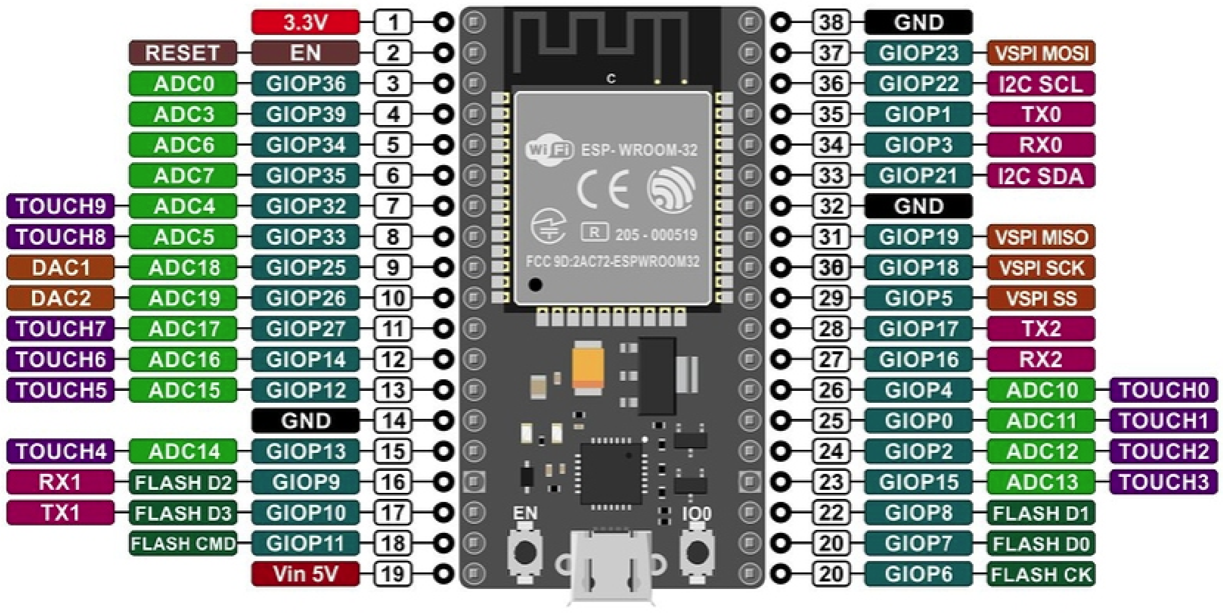
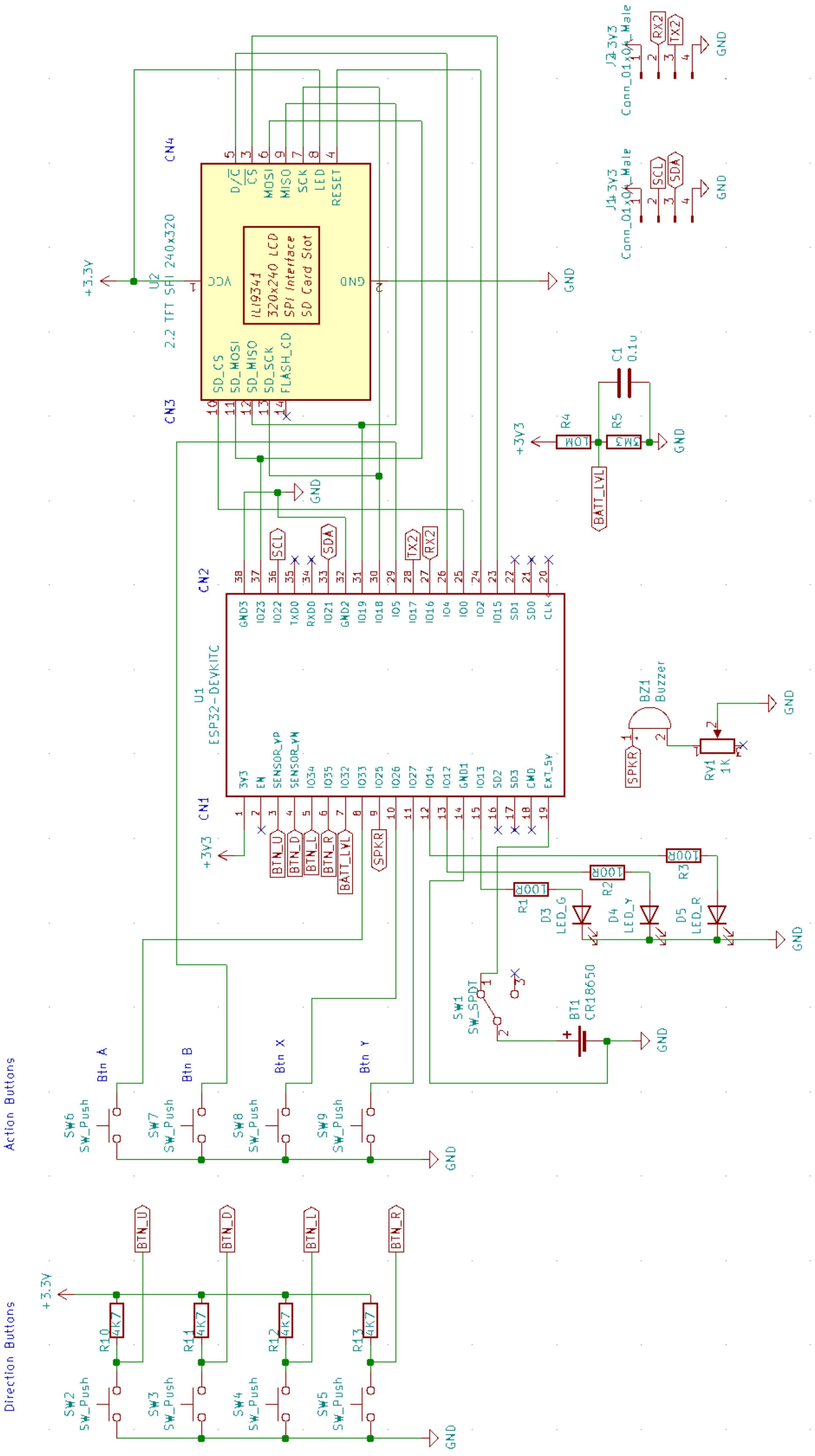
Badge My IoT Life
Lab Guide



ID	Item	Qty/Badge	Notes
U1	ESP 32 DevkitC	1	should have 38 pins
U2	2.2" TFT LCD ILI9341 SPI	1	
	18650 rechargeable battery	1	
BT1	Battery holder	1	
	Custom PCB - v1.2	1	
SW2-SW9	Tactile momentary buttons	8	normally open
BZ1	Mini speaker/transducer	1	passive, polarized
SW1	SPDT slide switch	1	power on/off
RV1	1K trim pot	1	speaker volume, 500 – 2,000 Ohm will work fine
R10-R13	4K7 resistors - 1/4 watt	4	pullup resistors for direction buttons
R1-R3	100R resistors - 1/4 watt	3	current limiters for LEDs
R4	10M resistor - 1/4 watt	1	voltage divider for battery level
R5	3M3 resistor - 1/4 watt	1	voltage divider for battery level
D3-D5	LEDs (1 red, 1 yellow, 1 green)	3	5mm, any colors will work
C1	0.1uF ceramic capacitor	1	marked 104
	USB Battery Charger – TP4056	1	
	Male and Female header pins	1x40 m, 2x40 f	allows ESP32 & TFT to be removed, I2C/serial jumpers
	3D printed hanger	1	allows badge to be worn with a lanyard

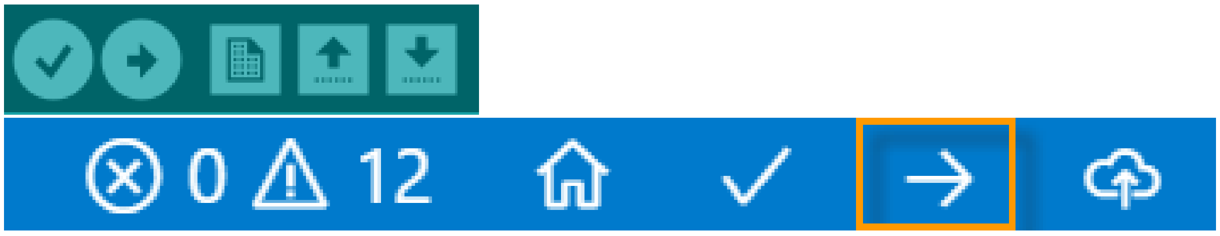

U1
ESP32-DEVKITC
CN1
CN2
3V3
EN
SENSOR_VP
SENSOR_VN
IO34
IO35
IO32
IO33
IO25
IO26
IO27
IO14
IO12
GND1
IO13
SD2
SD3
CMD
EXT_5V
GND3
IO23
IO22
TXD0
RXD0
IO21
GND2
IO19
IO18
IO5
IO17
IO16
IO4
IO0
IO2
IO15
SD1
SD0
CLK
U2
2.2 TFT SPI 240x320
CN3
CN4
VCC
GND
D/C
CS
MOSI
MISO
SCK
LED
RESET
SD_CS
SD_MOSI
SD_MISO
SD_SCK
FLASH_CD
ILI9341
320x240 LCD
SPI Interface
SD Card Slot
+3.3V
+3V3
GND
SCL
SDA
TX2
RX2
SPKR
BTN_U
BTN_D
BTN_L
BTN_R
BAT_LVL
J1 3V3
Conn_01x04_Male
J2 3V3
Conn_01x04_Male
C1
0.1u
R4
10M
R5
3M3
BZ1
Buzzer
RV1
1K
R1
100R
R2
100R
R3
100R
D3
LED_G
D4
LED_Y
D5
LED_R
SW1
SW_SPDT
BT1
CR18650
Action Buttons
Btn A
Btn B
Btn X
Btn Y
SW6
SW_Push
SW7
SW_Push
SW8
SW_Push
SW9
SW_Push
Direction Buttons
SW2
SW_Push
SW3
SW_Push
SW4
SW_Push
SW5
SW_Push
R10
4K7
R11
4K7
R12
4K7
R13
4K7

Lab #1 :: Hello World!

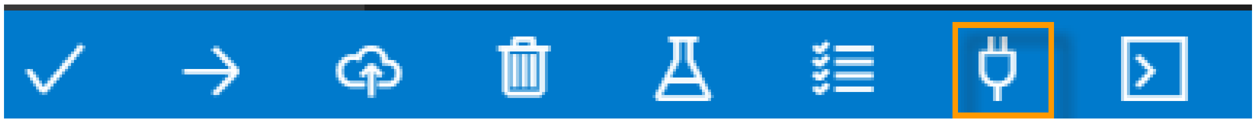
1. Create a new project with a name like “Hello_Badge” for the NodeMCU-32S
2. In the setup() function, enter:

```
    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:
5. In Arduino IDE, select **Tools -> Serial Monitor.**
   In PlatformIO, click the Serial Monitor icon.



6. Run the app again and verify that your “Hello Badge” message is shown in the Serial Monitor window:
7. **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`

Lab #2 :: Hello, My Name Is

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.
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:

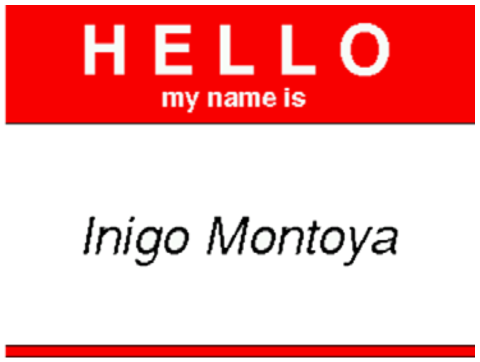
   tft.setRotation(3);

   This will set the TFT rotation to landscape, like the badge.
8. Before setup(), add:

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

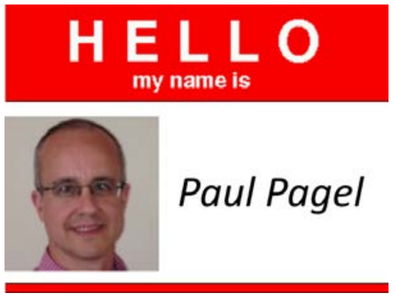
10. Upload the program. You should see something similar to:



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

Lab #3 :: Add Your Picture

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.
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.
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

Lab #4 :: Adding LEDs and Buttons

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

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);
   ```

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.

Lab #5 :: The Basic Bro

If you haven't already done so, download the badge code repo from GitHub to your laptop:
https://github.com/DigiTorus86/ESP32Badge

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.
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.

Lab #6 :: 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);
   ```
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 & 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 = "";
   ```

Upload sketch to verify that the badge is able to get on the WiFi and retrieve the current time and weather for Sandusky.

Optional: Follow the steps in the code to customize the city for the forecast API.

Lab #7 :: 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.

   PERIPHERAL STEPS

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.

   CENTRAL STEPS

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.

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 LEDs will pulse on send/receive data.

7. Start pushing puttons on the peripheral badge. The corresponding indicator on the central badge display should change accordingly.

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

Lab #8 :: 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.
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);`
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.