# IoT Hardware & Firmware

## Building an IoT Clock

**By Liam Osman**

IEEE Computer Society – Student Branch Chapter
University of South Florida
Tampa, FL, USA

4 October 2024

IEEE COMPUTER SOCIETY

*Student Branch Chapter at the University of South Florida*

UNIVERSITY of SOUTH FLORIDA

College of Engineering

# Checkin

# Agenda

- IoT Recap
- Hardware
  - Voltage / Resistance / Current
  - Breadboard and Connecting Pins
- Firmware
  - Definition
  - Platforms
- Board Setup
  - Registration
  - Circuit Building
- Coding

# IoT Recap

# Internet of Things

Network of physical objects or “things” that are embedded with sensors, software, and other technologies to collect and exchange data with other devices and systems over the internet.

# Examples of IoT Devices

- Fitness tracking devices
- Hospital call buttons

- Smart light bulbs
- Smoke Detectors (w/ internet)

- Automatic traffic lights
- Car computers (w/ internet)

# IoT Systems

Microcontrollers:

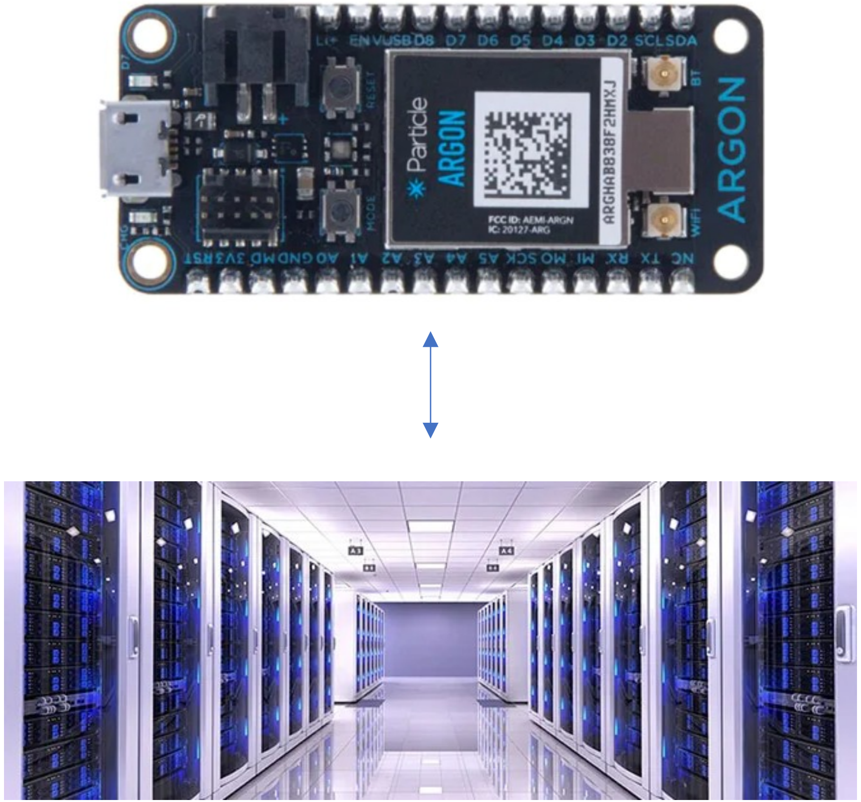
- Nordic Semiconductor nRF
- Microchip Atmega

Development Boards:

- Particle Argon
- Arduino BLE

Platforms:

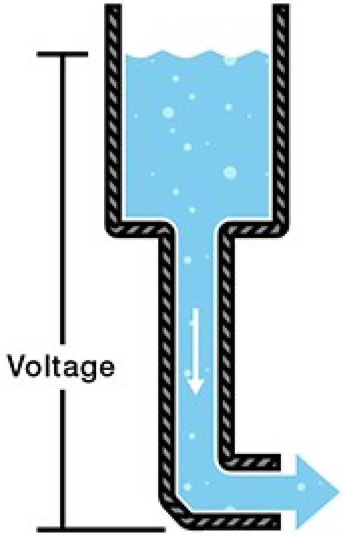
- Particle IO
- Amazon Web Services
- Cisco IoT



# Hardware

# Voltage

- Voltage is *Potential Energy*
- Similar to pipe water pressure
- All our hardware needs voltage



Common DC voltage levels:

- 5 V (max provided by most USB)
- 3.3 V (for lower power hardware)
- 3 V (coin cell battery)
- 1.5 V (AA & AAA battery)

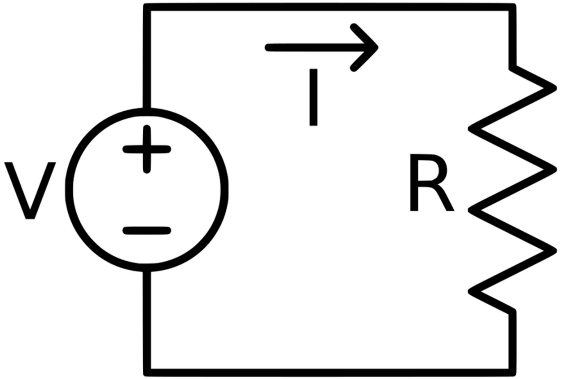
# Resistance

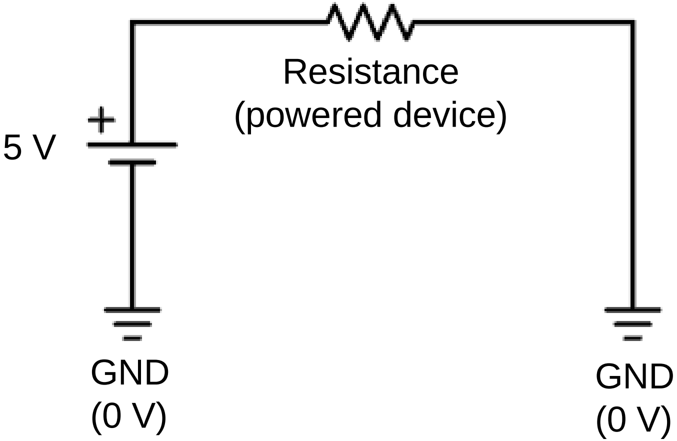
- Things we power have / need *Resistance*
- Similar to a garden hose faucet
- Having no resistance causes short circuit
- Your resistance will “spend” your energy

$$P = \frac{V^2}{R}$$

# Current

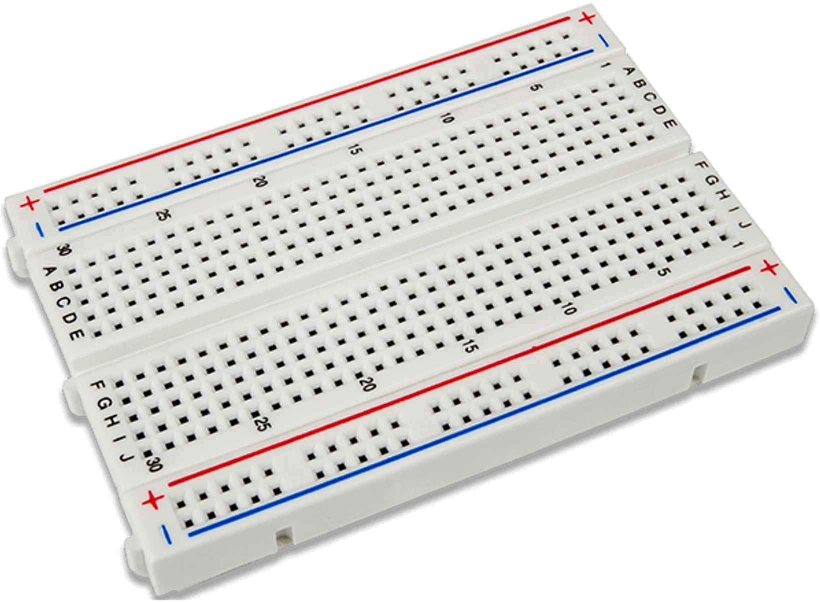
- Current is like the water which flows in our garden hose
- The base reference that voltage relates to is called *ground*
- We have to “drain” our voltages to ground voltage to get current



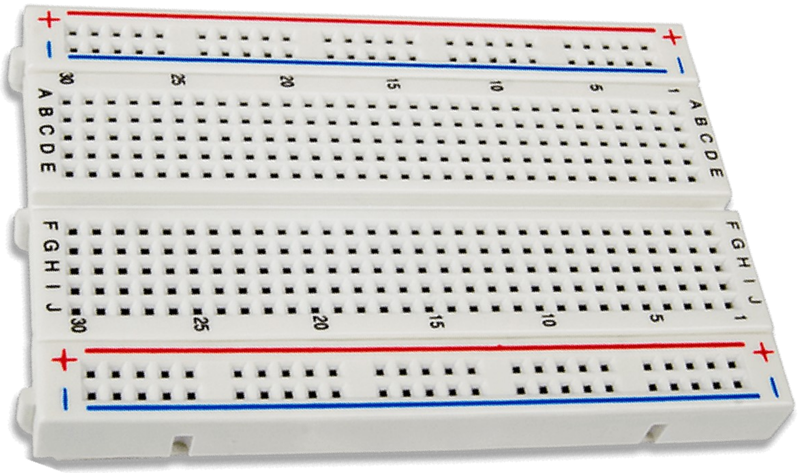


# Breadboard

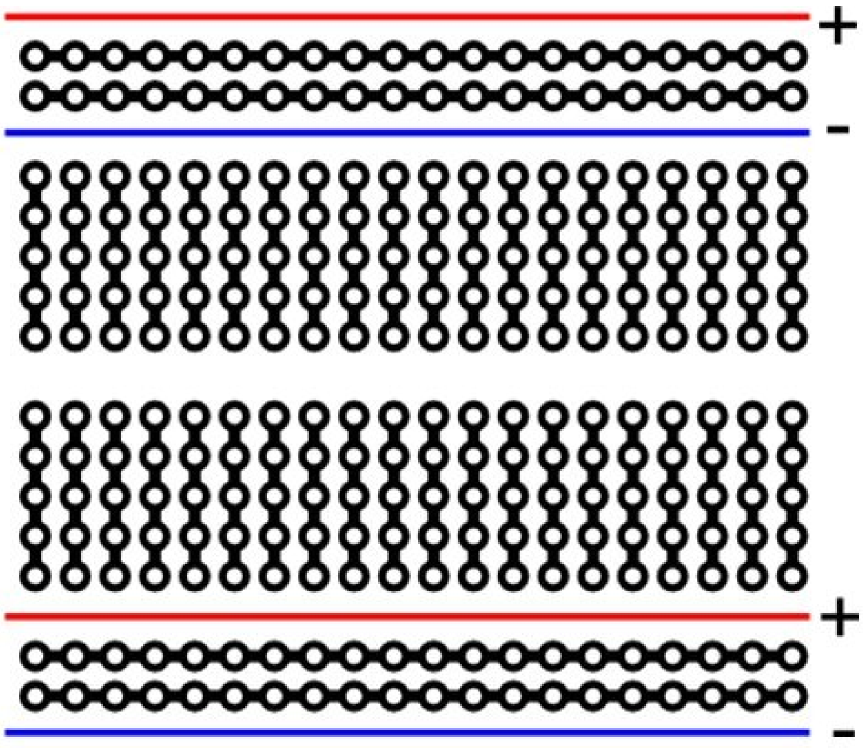
- Help develop without soldering
- Easy swapping of components
- Hidden “wires” to connect parts
- Slot in pins to make a circuit



# Breadboard



=

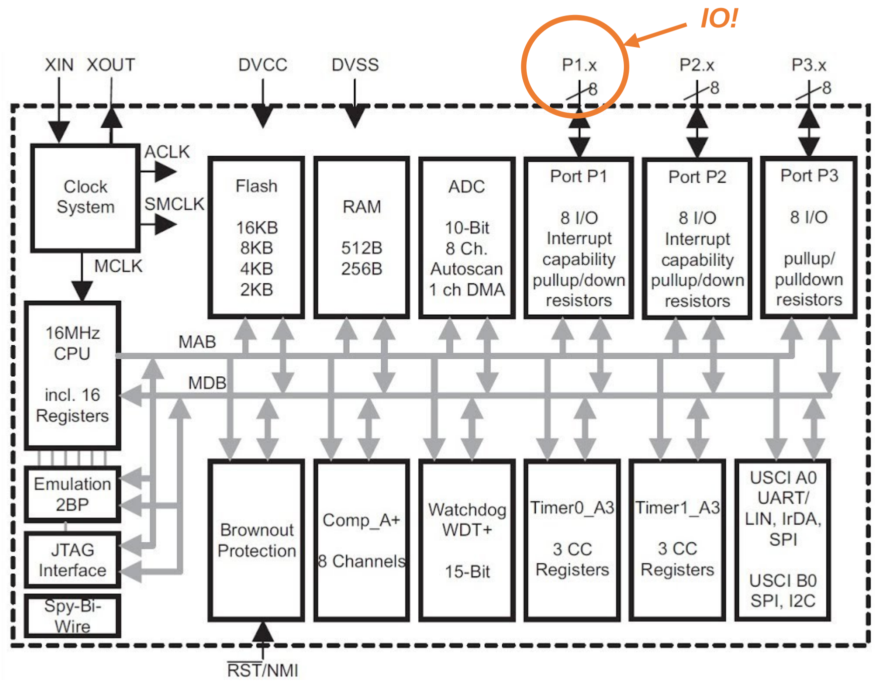


# Firmware

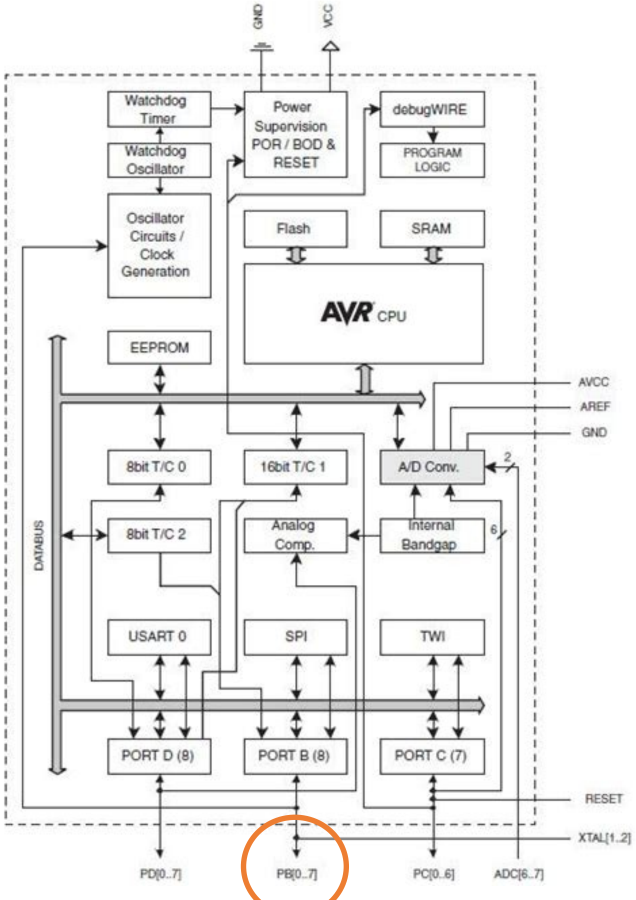
# Firmware

- Code that directly controls hardware
  - Memory, interrupts, data transfer, boot process
- Most often programmed in C, but Rust use is growing
- Firmware is everywhere!
  - BIOS, SSDs, keyboards, IoT devices etc.
- To write firmware, you must first understand the hardware
- Every processor architecture requires specific firmware

# Chip Architectures



*Texas Instruments MSP430*

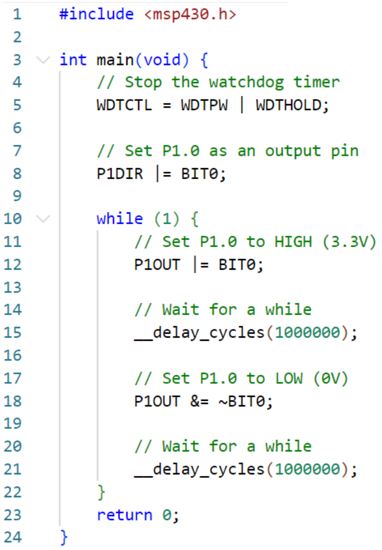


*Microchip/Atmel AVR*

IO!

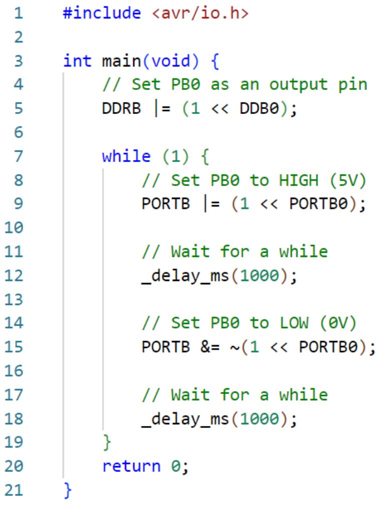
# Embedded C

- The same thing as C, but with compiler-specific macros
- No libraries, you must read the chip’s documentation!



```c
#include <msp430.h>

int main(void) {
    // Stop the watchdog timer
    WDTCTL = WDTPW | WDTHOLD;

    // Set P1.0 as an output pin
    P1DIR |= BIT0;

    while (1) {
        // Set P1.0 to HIGH (3.3V)
        P1OUT |= BIT0;

        // Wait for a while
        __delay_cycles(1000000);

        // Set P1.0 to LOW (0V)
        P1OUT &= ~BIT0;

        // Wait for a while
        __delay_cycles(1000000);
    }
    return 0;
}
```

*Texas Instruments MSP430*

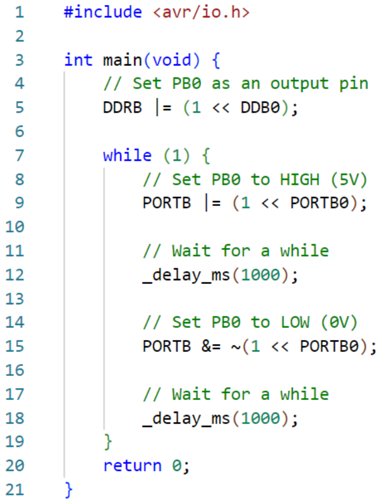


```c
#include <avr/io.h>

int main(void) {
    // Set PB0 as an output pin
    DDRB |= (1 << DDB0);

    while (1) {
        // Set PB0 to HIGH (5V)
        PORTB |= (1 << PORTB0);

        // Wait for a while
        _delay_ms(1000);

        // Set PB0 to LOW (0V)
        PORTB &= ~(1 << PORTB0);

        // Wait for a while
        _delay_ms(1000);
    }
    return 0;
}
```

*Microchip/Atmel AVR*

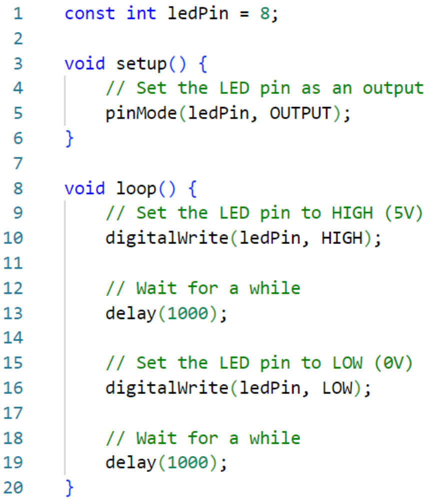
# Firmware Frameworks

- Abstract the manual bitwise operations with a header file
- The Arduino framework is widely used and works with most chips



```
#include <avr/io.h>

int main(void) {
    // Set PB0 as an output pin
    DDRB |= (1 << DDB0);

    while (1) {
        // Set PB0 to HIGH (5V)
        PORTB |= (1 << PORTB0);

        // Wait for a while
        _delay_ms(1000);

        // Set PB0 to LOW (0V)
        PORTB &= ~(1 << PORTB0);

        // Wait for a while
        _delay_ms(1000);
    }
    return 0;
}
```

*Embedded C AVR*



```
const int ledPin = 8;

void setup() {
    // Set the LED pin as an output
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // Set the LED pin to HIGH (5V)
    digitalWrite(ledPin, HIGH);

    // Wait for a while
    delay(1000);

    // Set the LED pin to LOW (0V)
    digitalWrite(ledPin, LOW);

    // Wait for a while
    delay(1000);
}
```

*Arduino Framework*

# Firmware to Cloud

- When working with IoT applications you will typically use frameworks
- Particle uses the Arduino framework with added functionality, such as cloud variables and functions
  - Cloud variables and functions can be accessed from the web interface!
- NuvIoT and Arduino also offer cloud specific libraries for IoT applications

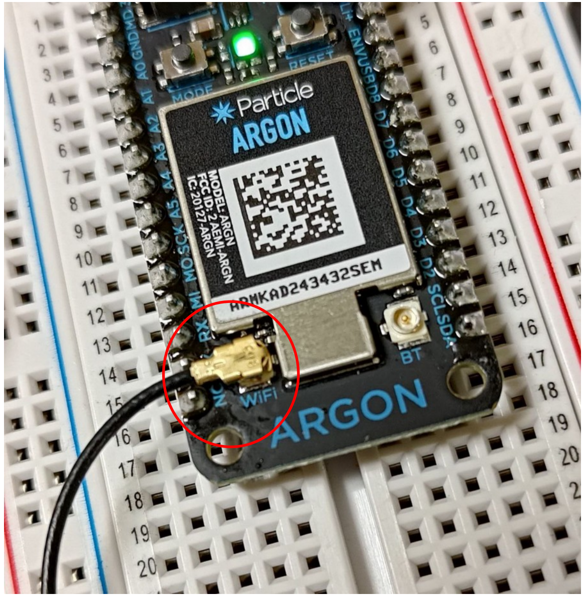
# Board Setup

# Getting Ready

1. docs.particle.io/quickstart/argon/
2. “Set up your Argon”
3. “Get Started”
4. Make Account (required)

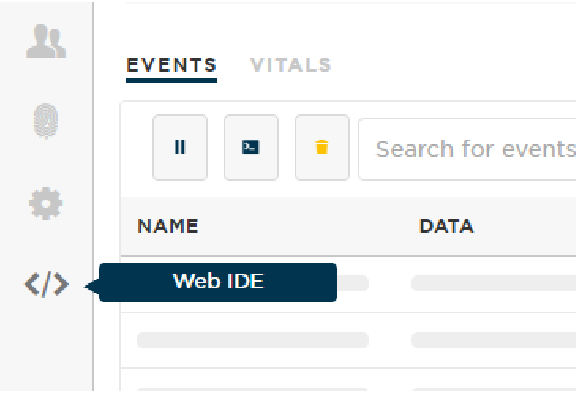
# Updating

1. Attach antenna to Argon board “Wi-Fi” port
2. “Start setting up my device”
3. Attach board with USB cable to laptop
4. “Select Device,” pick device, and “connect”
5. “Continue,” pick device, and “connect”
6. “Continue” and “Update Device”



# Registering

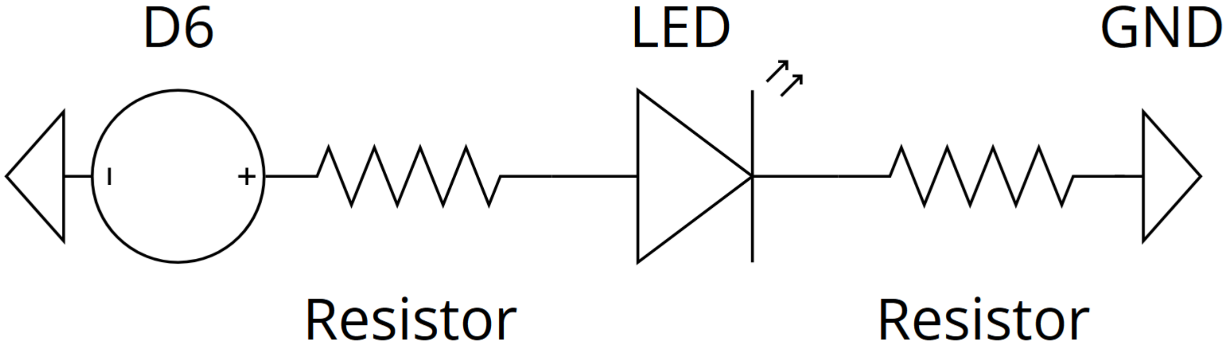
1. “Or create a new product” and give a name
2. “Add to product” and gave a device name
3. “Name Device”
4. Choose Wi-Fi network (Not school Wi-Fi, try phone hotspot)
5. Activate Device
6. “Go to Console”
7. Open Web IDE

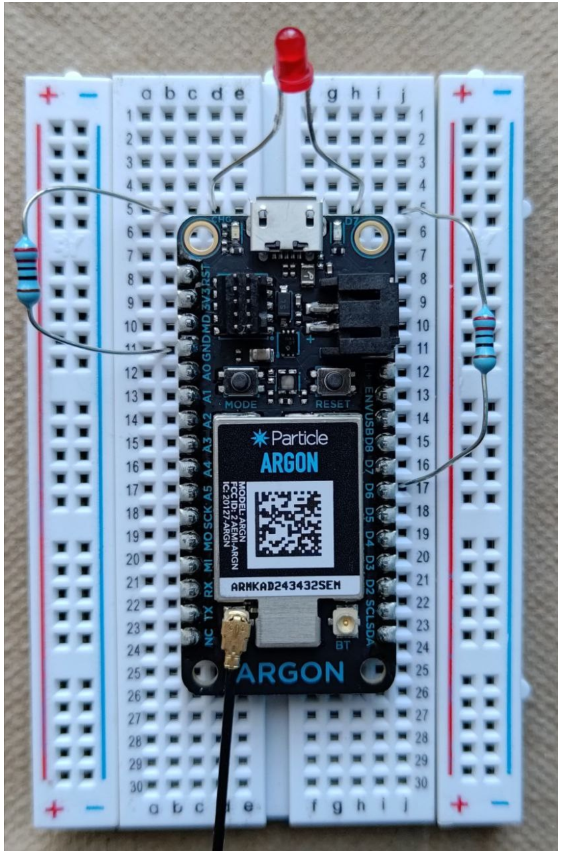


# Circuit Assembly

**3v3 voltage to long leg of LED**

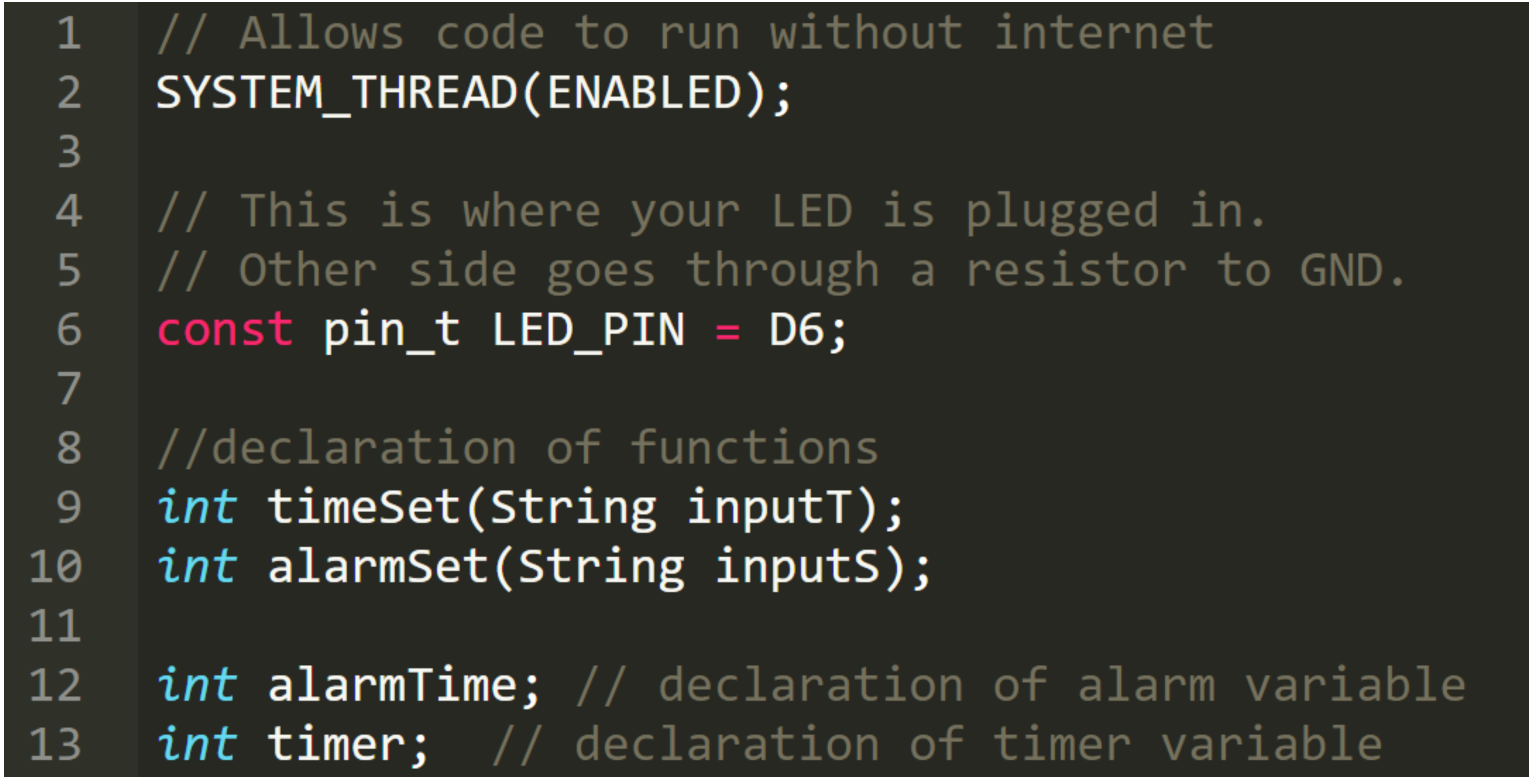
**GND to short leg of LED**





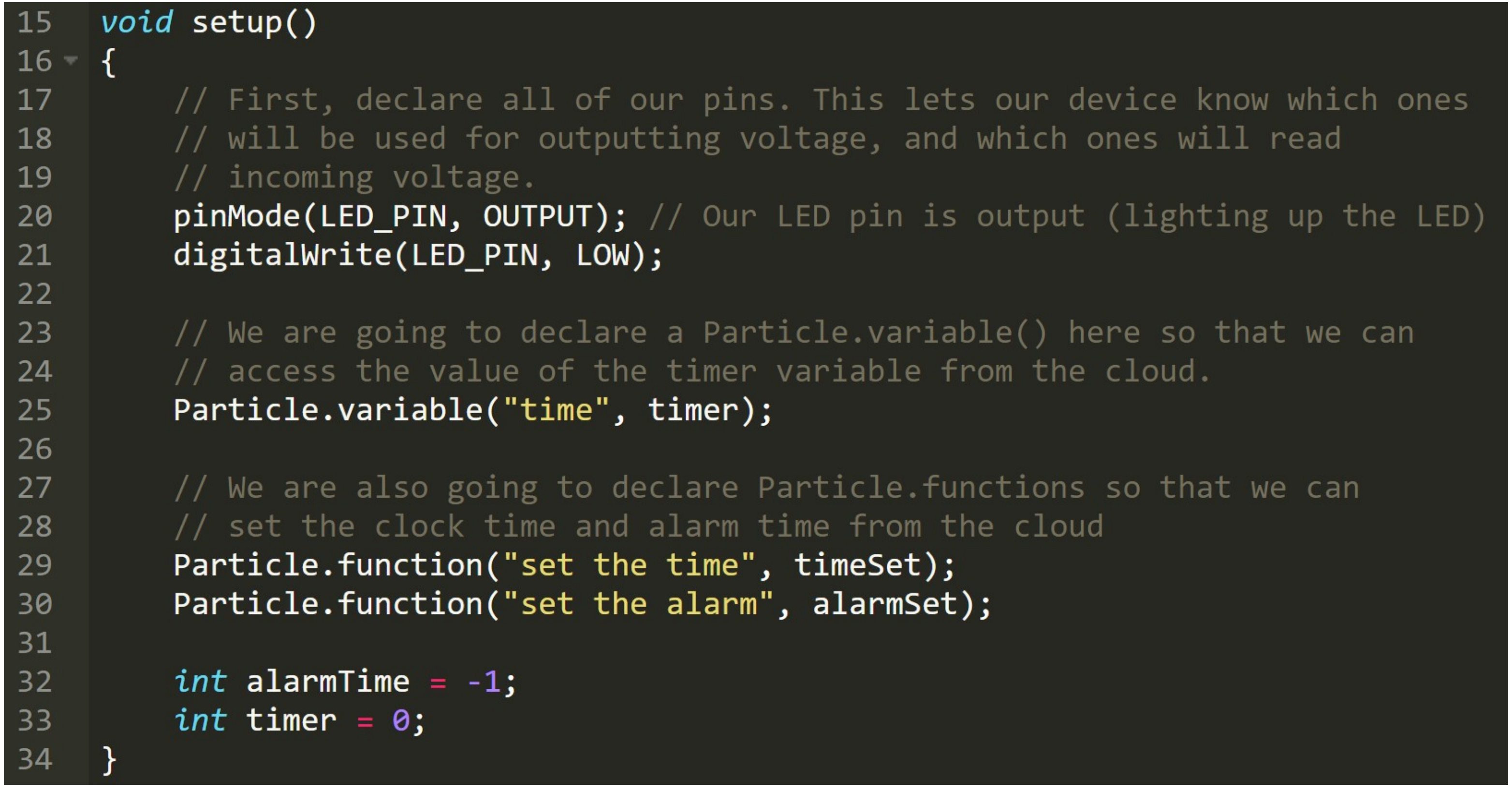
# Coding

# Code – Beginning



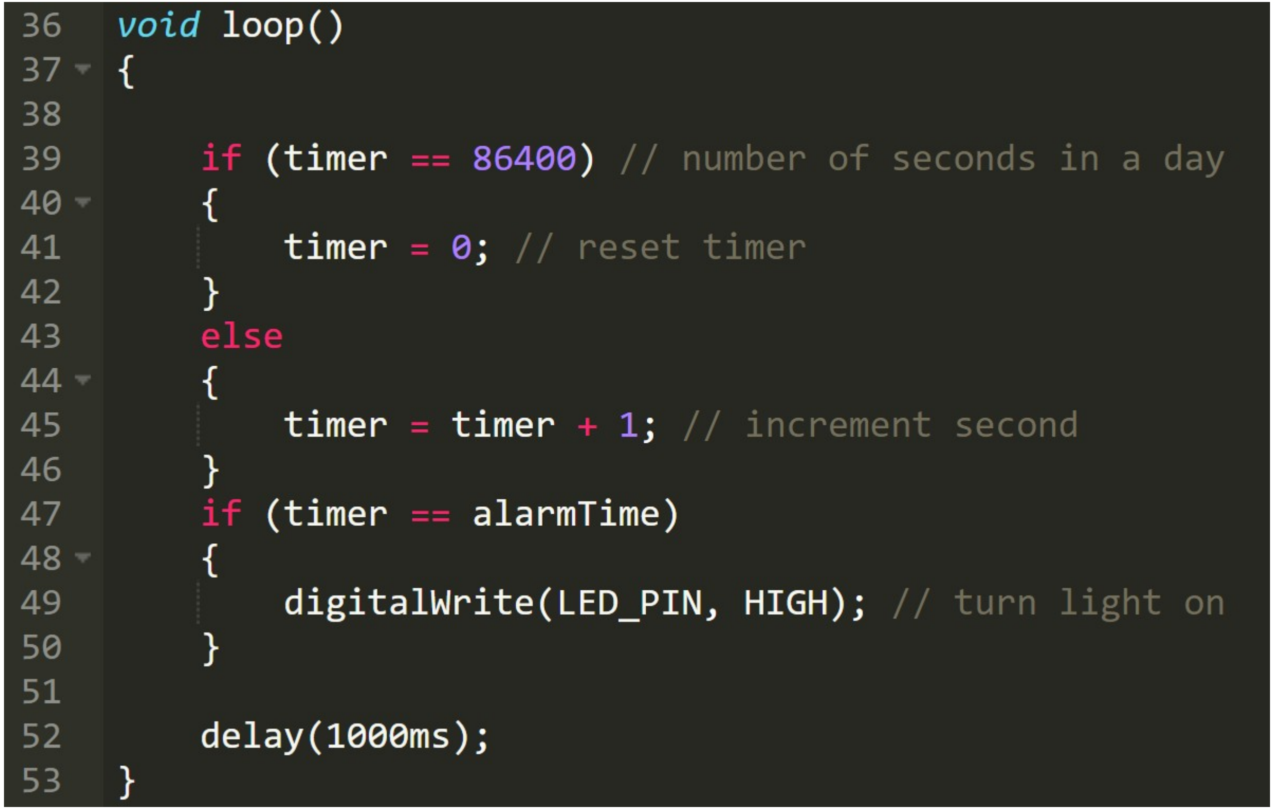
```
// Allows code to run without internet
SYSTEM_THREAD(ENABLED);

// This is where your LED is plugged in.
// Other side goes through a resistor to GND.
const pin_t LED_PIN = D6;

//declaration of functions
int timeSet(String inputT);
int alarmSet(String inputS);

int alarmTime; // declaration of alarm variable
int timer;  // declaration of timer variable
```

# Code - Setup



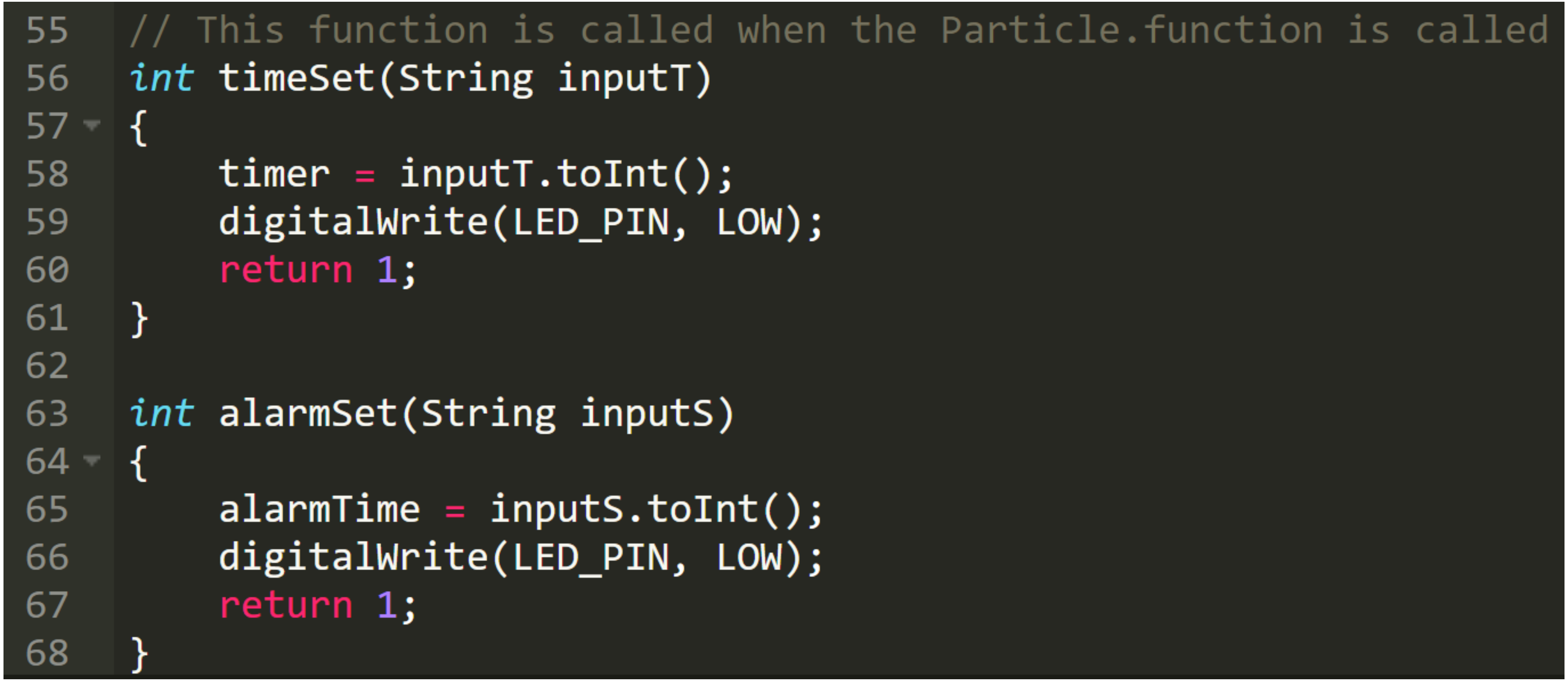
```
void setup()
{
    // First, declare all of our pins. This lets our device know which ones
    // will be used for outputting voltage, and which ones will read
    // incoming voltage.
    pinMode(LED_PIN, OUTPUT); // Our LED pin is output (lighting up the LED)
    digitalWrite(LED_PIN, LOW);

    // We are going to declare a Particle.variable() here so that we can
    // access the value of the timer variable from the cloud.
    Particle.variable("time", timer);

    // We are also going to declare Particle.functions so that we can
    // set the clock time and alarm time from the cloud
    Particle.function("set the time", timeSet);
    Particle.function("set the alarm", alarmSet);

    int alarmTime = -1;
    int timer = 0;
}
```

# Code - Loop



```
void loop()
{

    if (timer == 86400) // number of seconds in a day
    {
        timer = 0; // reset timer
    }
    else
    {
        timer = timer + 1; // increment second
    }
    if (timer == alarmTime)
    {
        digitalWrite(LED_PIN, HIGH); // turn light on
    }

    delay(1000ms);
}
```

# Code – Functions



```
// This function is called when the Particle.function is called
int timeSet(String inputT)
{
    timer = inputT.toInt();
    digitalWrite(LED_PIN, LOW);
    return 1;
}

int alarmSet(String inputS)
{
    alarmTime = inputS.toInt();
    digitalWrite(LED_PIN, LOW);
    return 1;
}
```

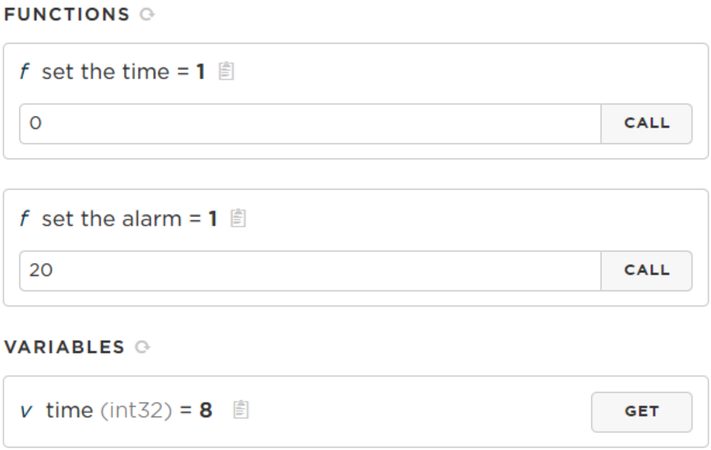
# Code – Dashboard Interface

You can now:

- Set the clock and timer remotely
- Check the current time

Also possible:

- Change output to hours & minutes
- Using a mobile app to access the dashboard and notifications remotely



FUNCTIONS

*f* set the time = **1**

0

*f* set the alarm = **1**

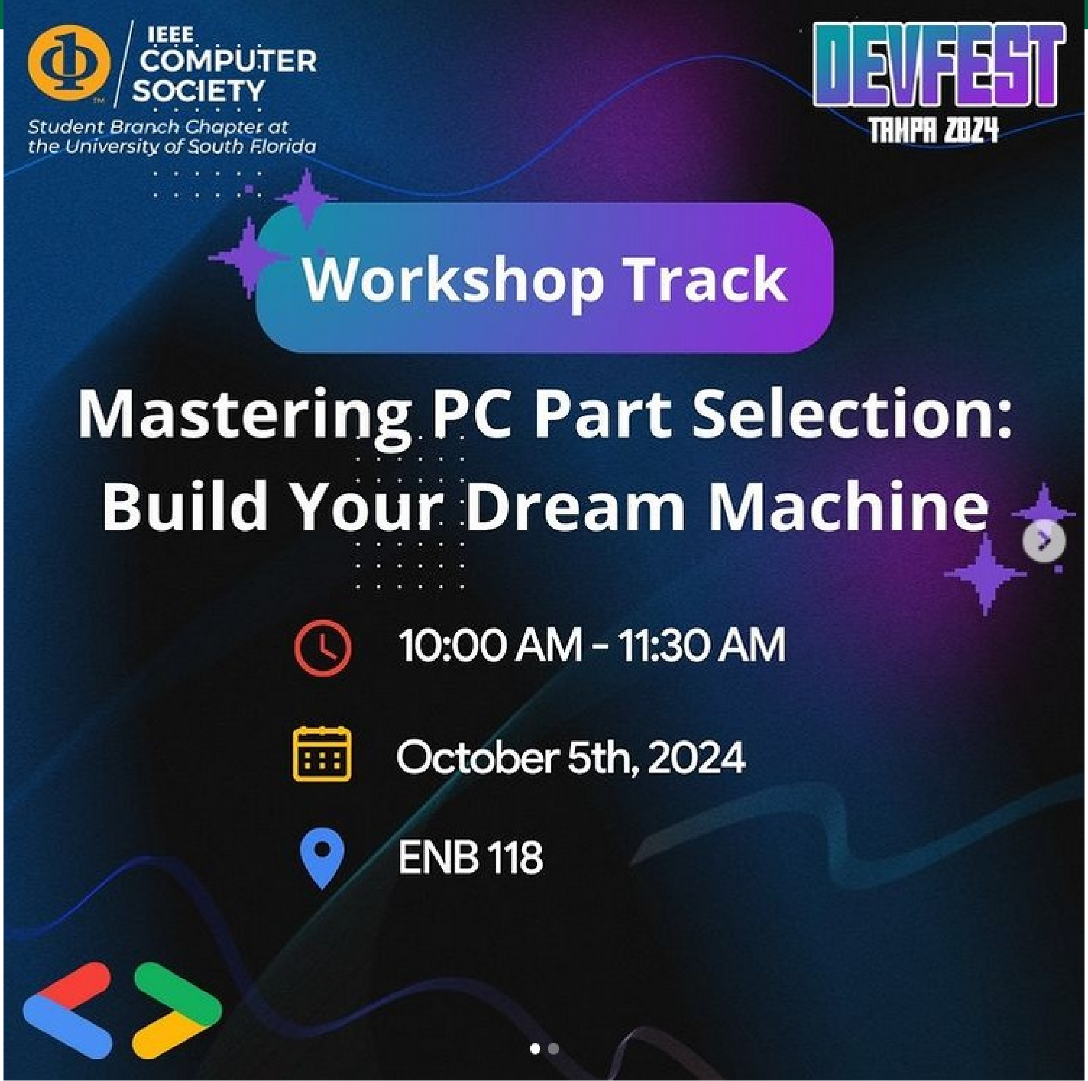
20

VARIABLES

*v* time (int32) = **8**

More tutorials and code are available at

https://docs.particle.io/getting-started/hardware-tutorials/hardware-examples/



IEEE COMPUTER SOCIETY

Student Branch Chapter at the University of South Florida

DEVFEST TAMPA 2024

## Workshop Track

# Mastering PC Part Selection: Build Your Dream Machine

10:00 AM - 11:30 AM

October 5th, 2024

ENB 118

Publix chicken & vegetarian wraps provided!

# Introduction into Embedded Systems Workshop

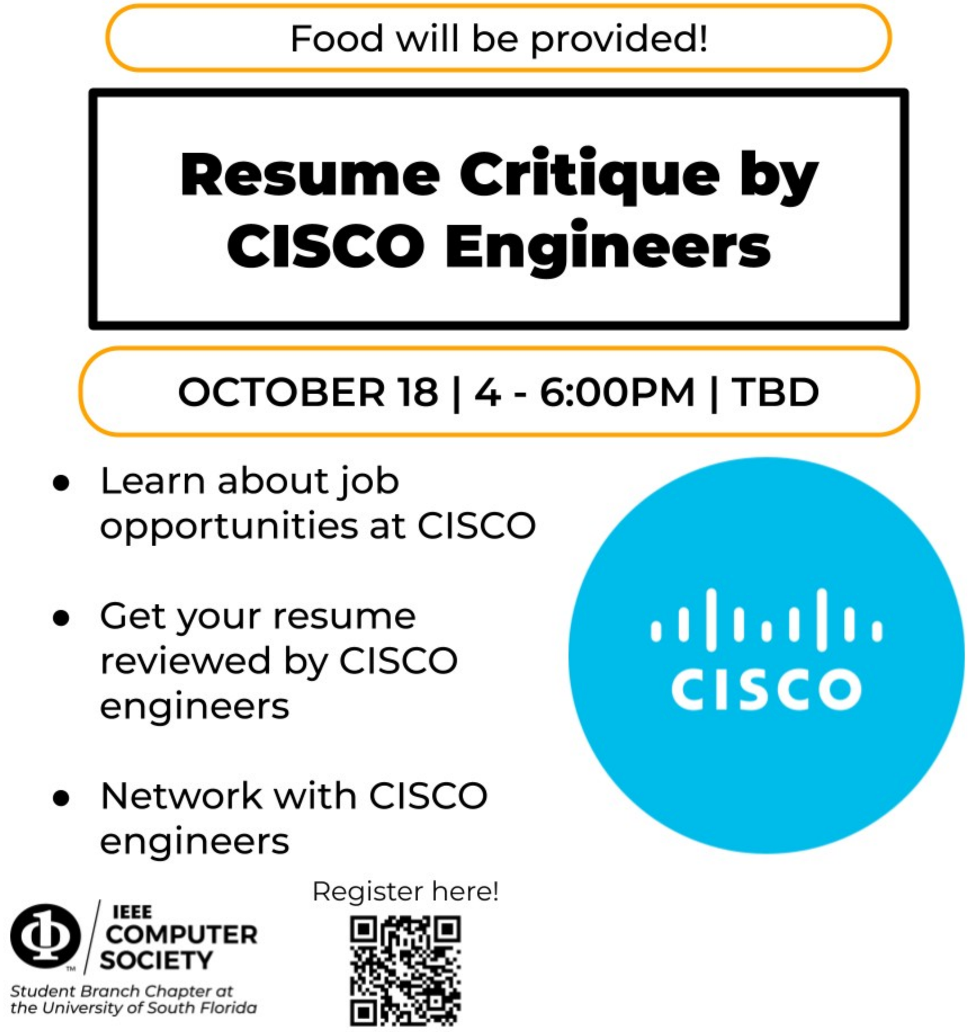
OCTOBER 25 | 4 - 5:30PM | TBD

- Overview of Embedded Applications within today's society
- Assembly & Embedded C
- Showcase of simple microcontroller project w/ MSP430

Register here!

IEEE COMPUTER SOCIETY

*Student Branch Chapter at the University of South Florida*



Food will be provided!

# Resume Critique by CISCO Engineers

OCTOBER 18 | 4 - 6:00PM | TBD

- Learn about job opportunities at CISCO
- Get your resume reviewed by CISCO engineers
- Network with CISCO engineers

CISCO

IEEE COMPUTER SOCIETY

*Student Branch Chapter at the University of South Florida*

Register here!

# Questions?

IEEE
COMPUTER
SOCIETY

*Student Branch Chapter at*
*the University of South Florida*

UNIVERSITY of
SOUTH FLORIDA
College of Engineering