IoT Hardware & Firmware

Building an IoT Kit

By Liam Osman

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

14 November 2023

IEEE COMPUTER SOCIETY

Student Branch Chapter at the University of South Florida

UNIVERSITY of SOUTH FLORIDA
College of Engineering

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:

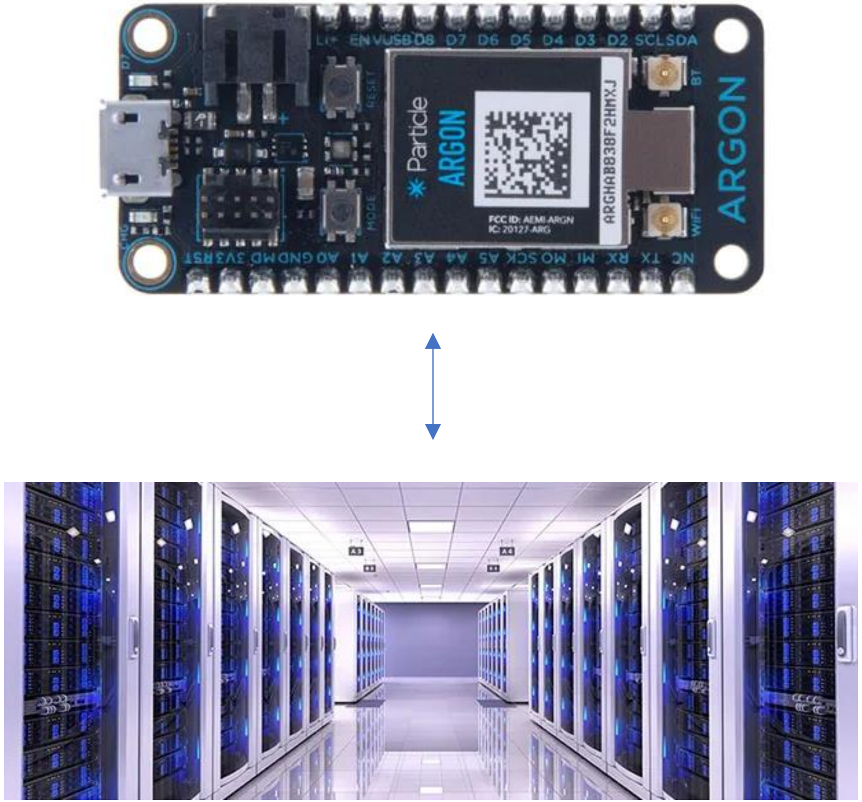
- Nordic Semiconductor nRF
- Microchip Atmega

Development Boards:

- Particle Argon
- Arduino BLE

Platforms:

- Particle IO
- Amazon Web Services
- Cisco IoT



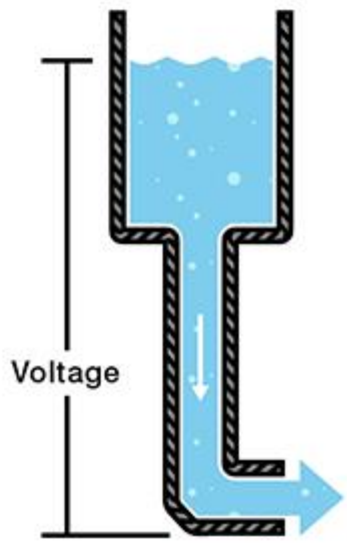
Hardware

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)



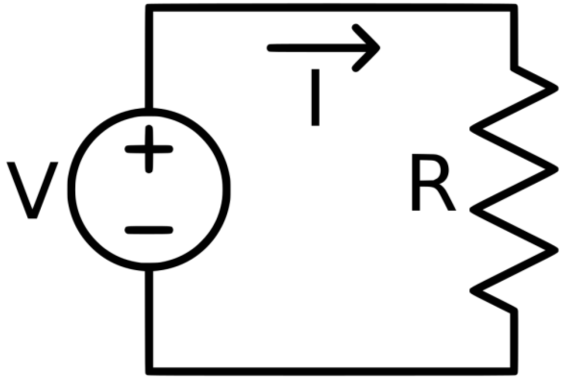
Resistance

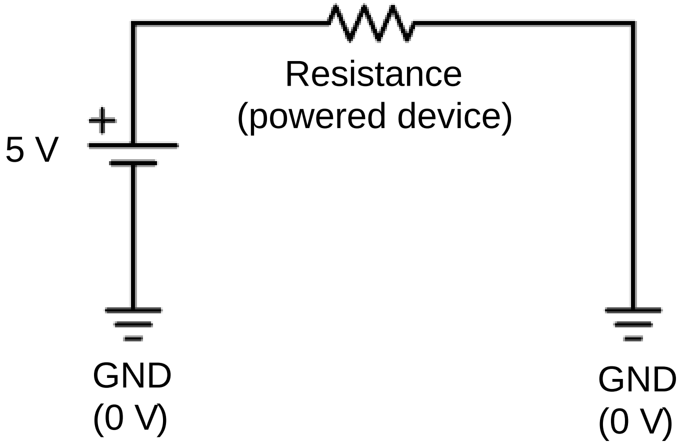
- Things we power have / need *Resistance*
- Similar to a garden hose faucet
- Having no resistance causes short circuit
- Your resistance will “spend” your energy

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

Current

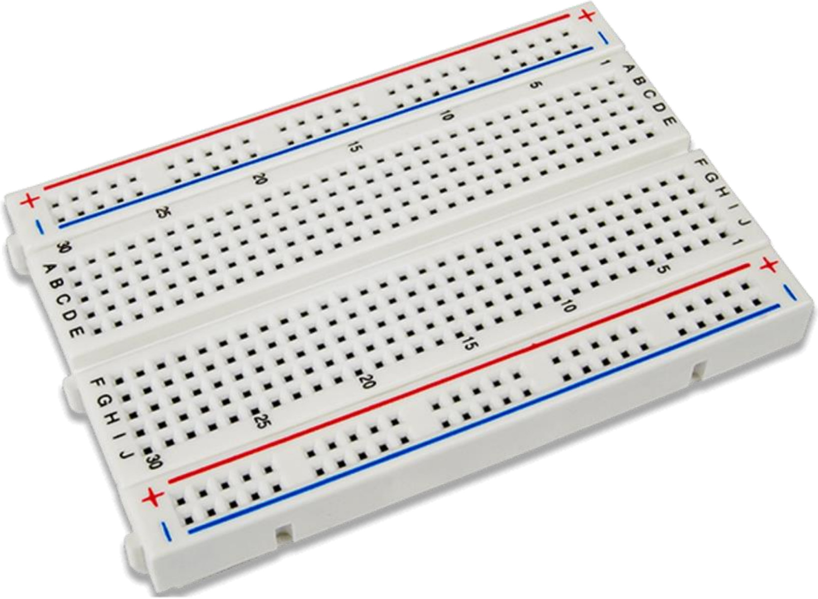
- Current is like the water which flows in our garden hose
- The base reference that voltage relates to is called *ground*
- We have to “drain” our voltages to ground voltage to get current



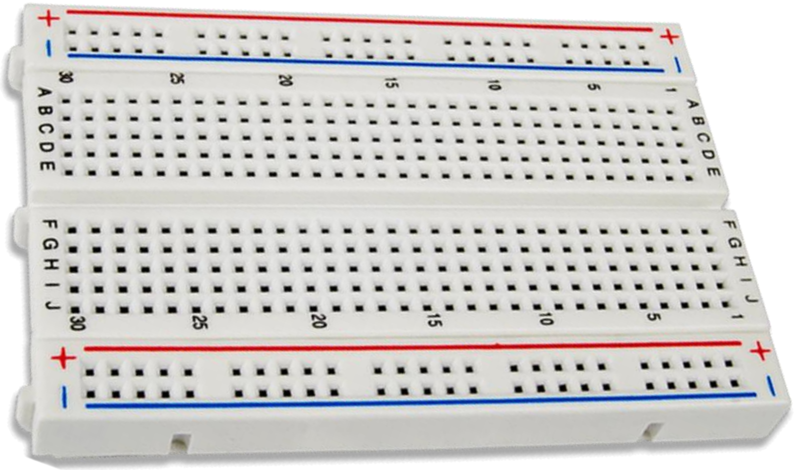


Breadboard

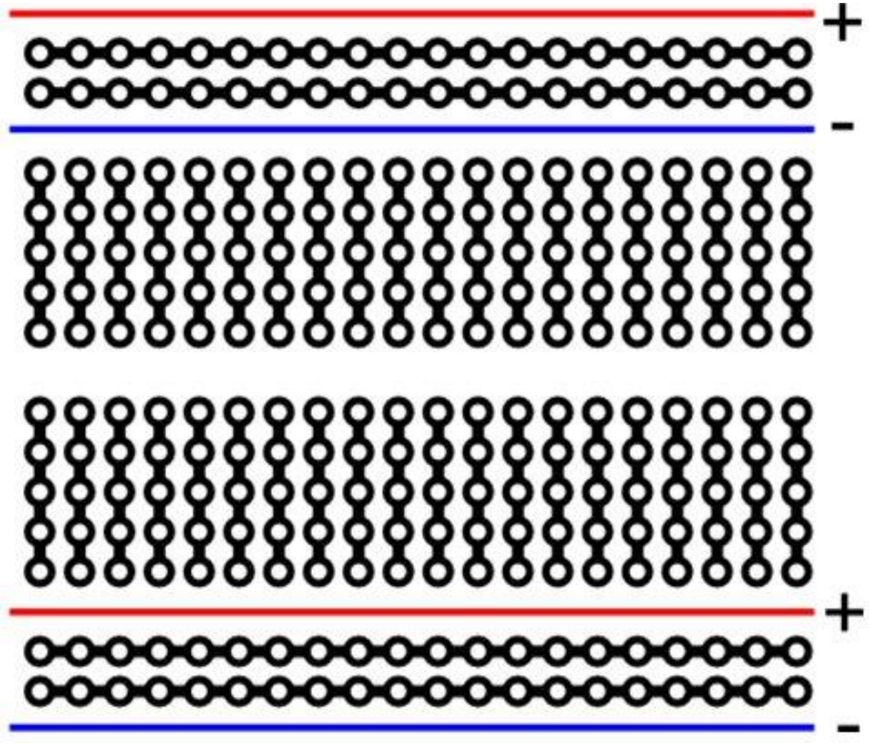
- Help develop without soldering
- Easy swapping of components
- Hidden “wires” to connect parts
- Slot in pins to make a circuit



Breadboard



=



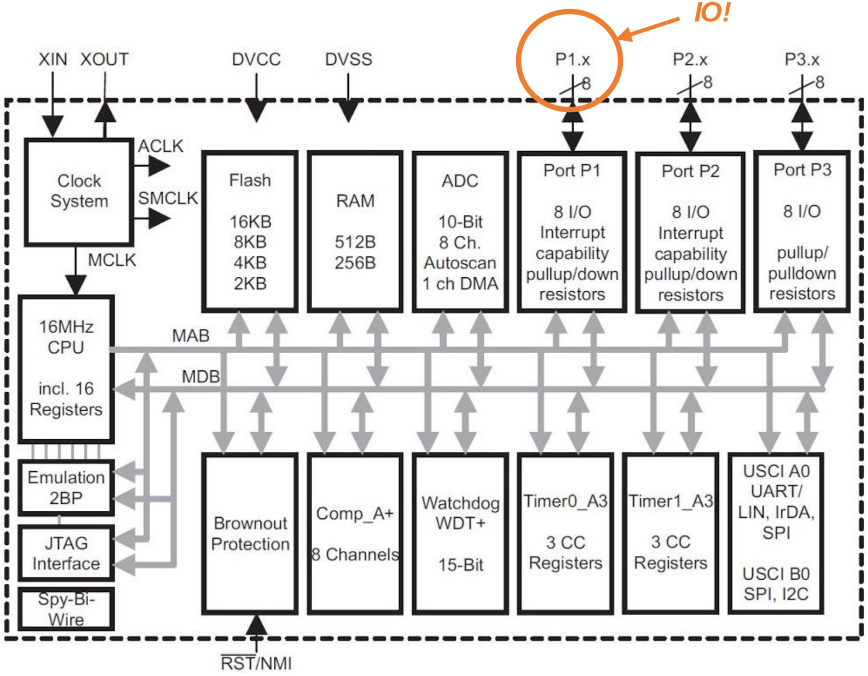
Firmware

Firmware

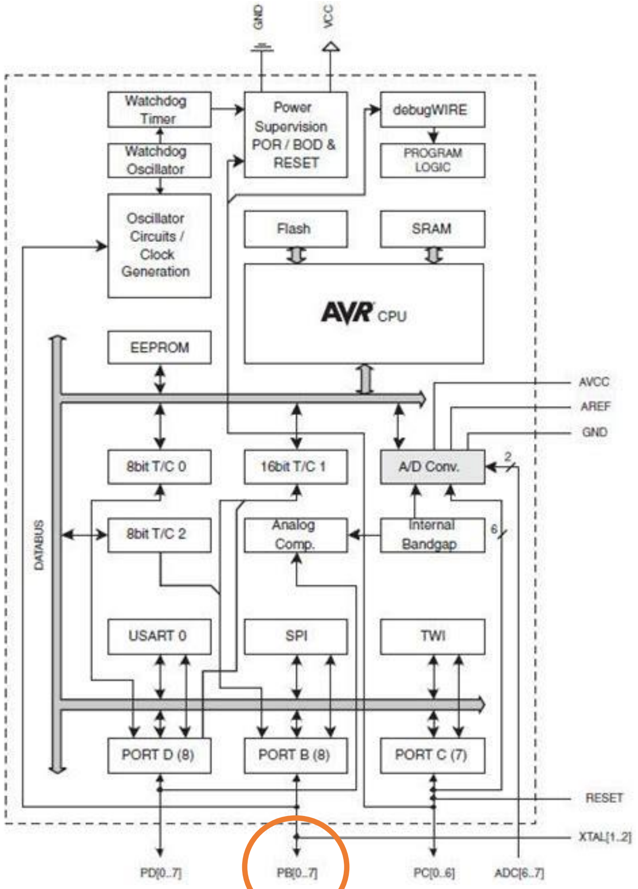
- Code that directly controls hardware
  - Memory, interrupts, data transfer, boot process
- Most often programmed in C, but Rust use is growing
- Firmware is everywhere!
  - BIOS, SSDs, keyboards, IoT devices etc.
- To write firmware, you must first understand the hardware
- Every processor architecture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

Chip Architectures

IO!



Texas Instruments MSP430

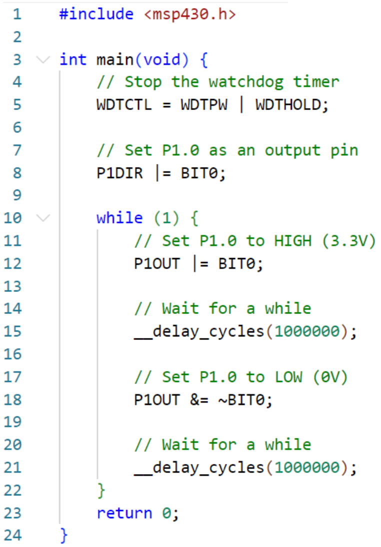


Microchip/Atmel AVR

IO!

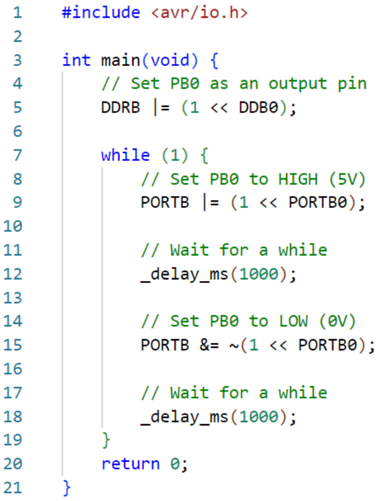
Embedded C

- 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

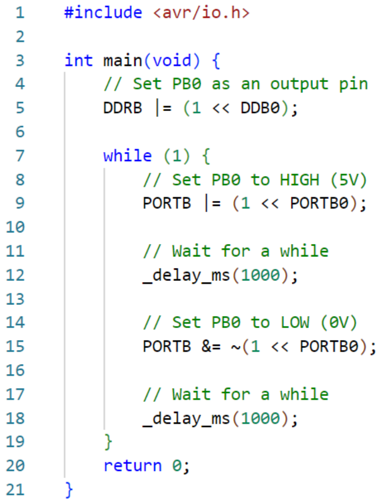


```c
#include <avr/io.h>

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

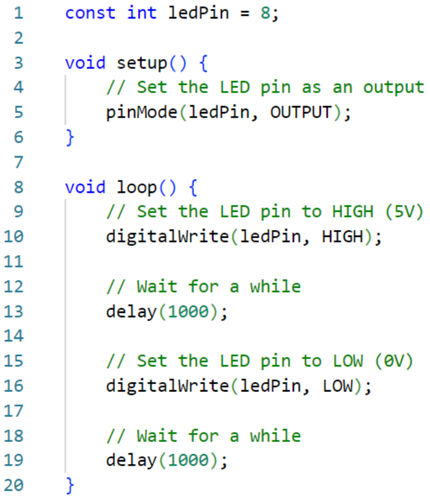
Firmware Frameworks

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



```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;
}
```

Embedded C AVR



```c
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

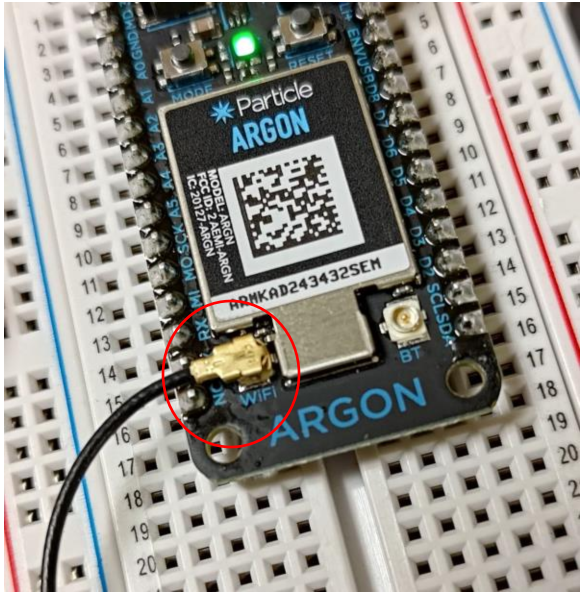
Board Setup

Getting Ready

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

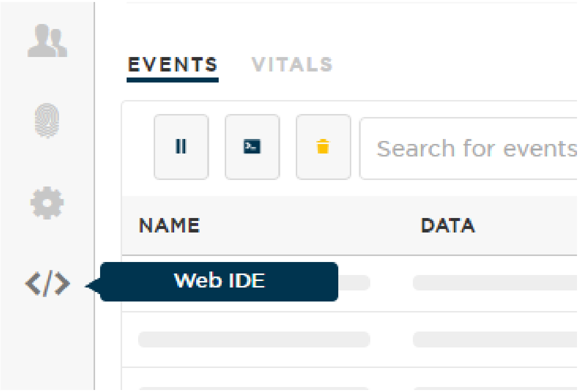
Updating

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

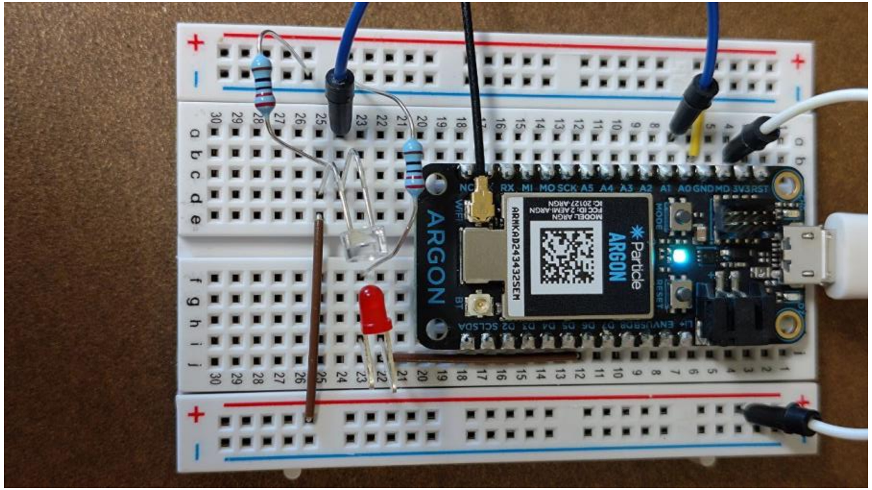
1. “Or create a new product” and give a name
2. “Add to product” and gave a device name
3. “Name Device”
4. Choose Wi-Fi network (Not school Wi-Fi, try phone hotspot)
5. Activate Device
6. “Go to Console”
7. Open Web IDE

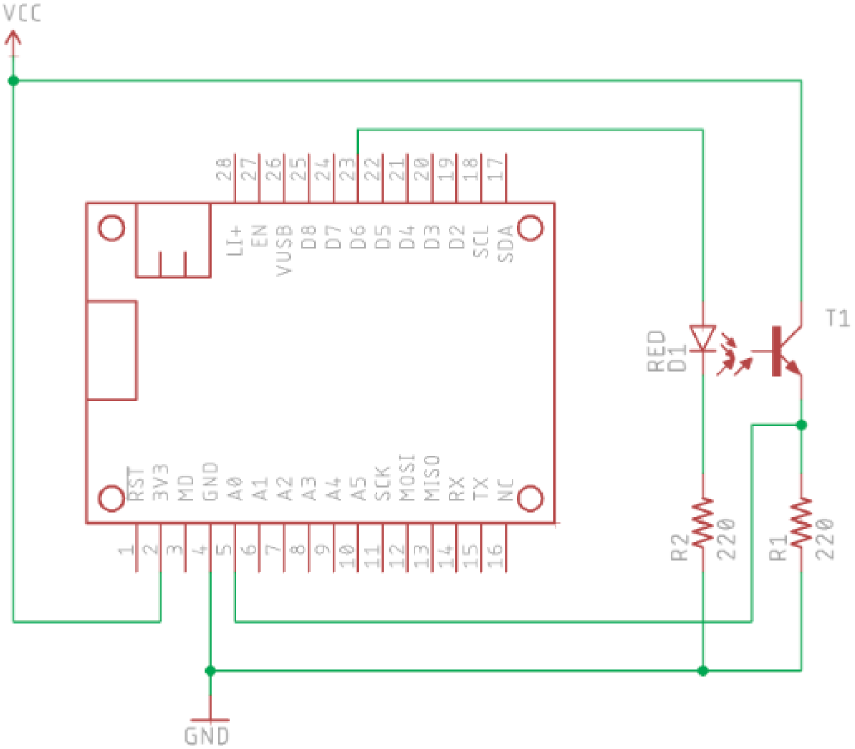


Circuit Assembly

3v3 voltage goes to:

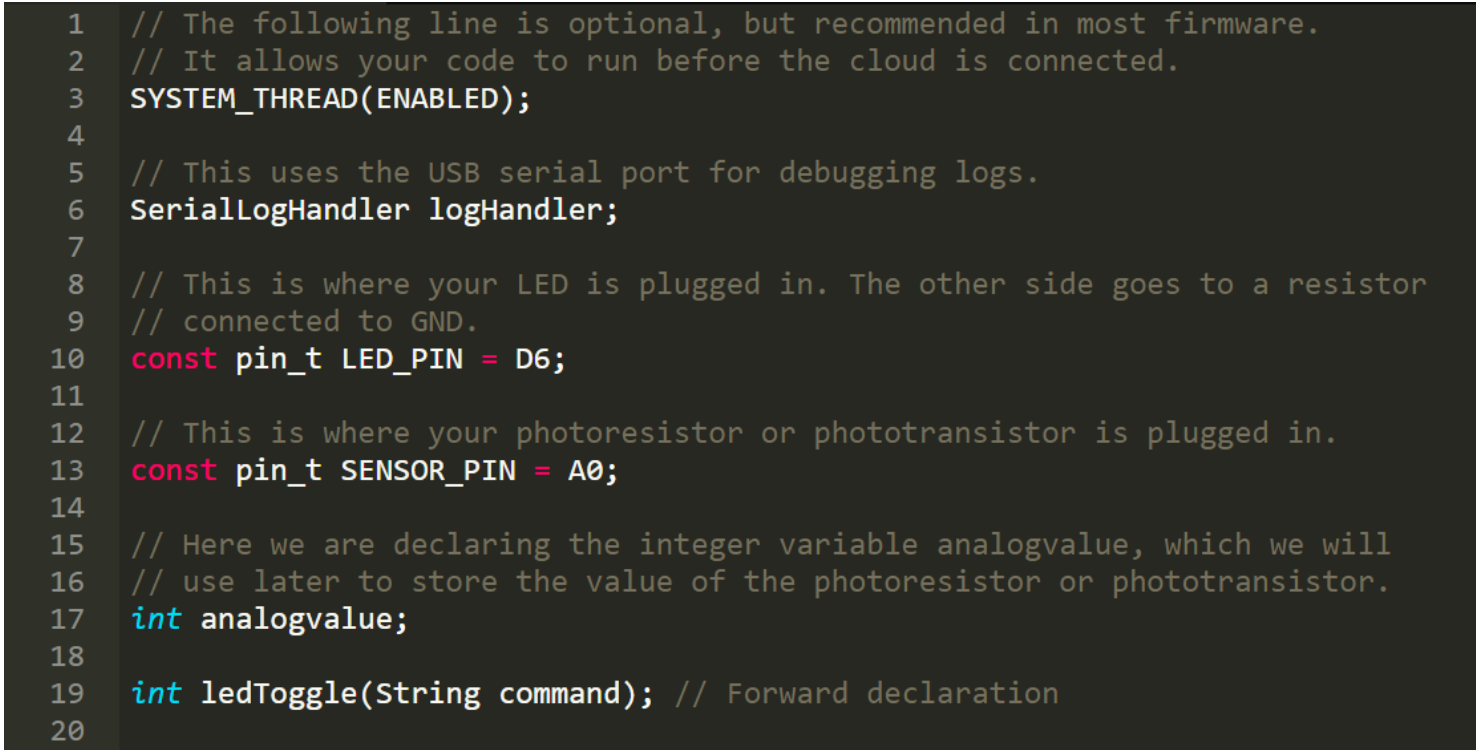
- Long leg of LED
- Short leg of phototransistor





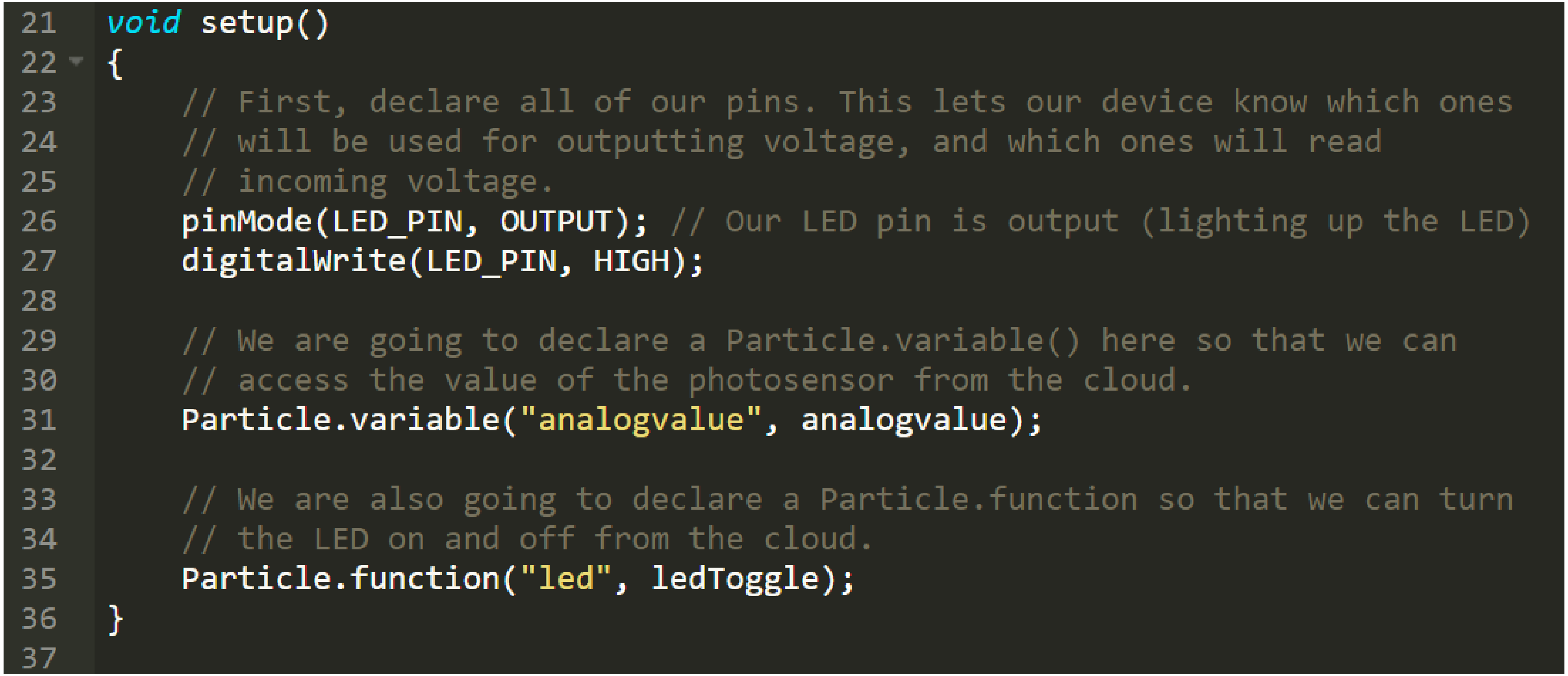
Coding

Code – Beginning



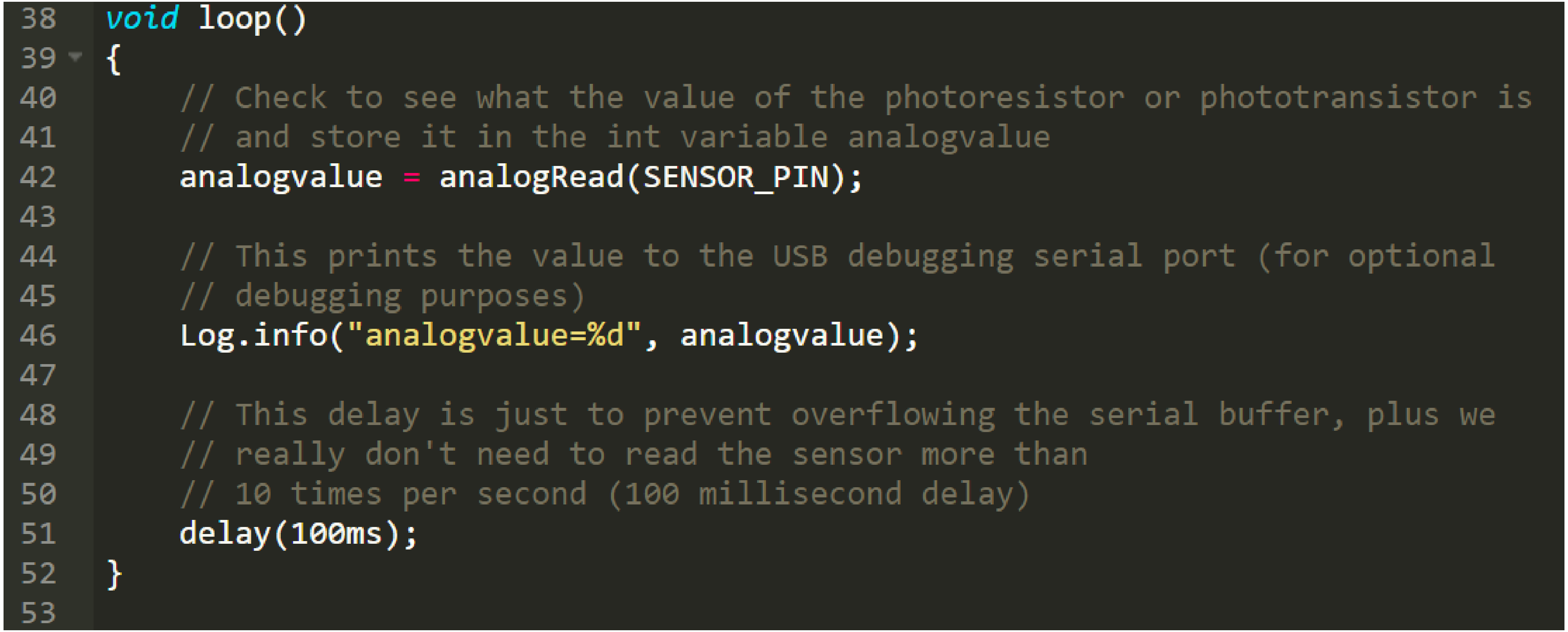
```
// The following line is optional, but recommended in most firmware.
// It allows your code to run before the cloud is connected.
SYSTEM_THREAD(ENABLED);

// This uses the USB serial port for debugging logs.
SerialLogHandler logHandler;

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

// This is where your photoresistor or phototransistor is plugged in.
const pin_t SENSOR_PIN = A0;

// Here we are declaring the integer variable analogvalue, which we will
// use later to store the value of the photoresistor or phototransistor.
int analogvalue;

int ledToggle(String command); // Forward declaration

```

Code - Setup



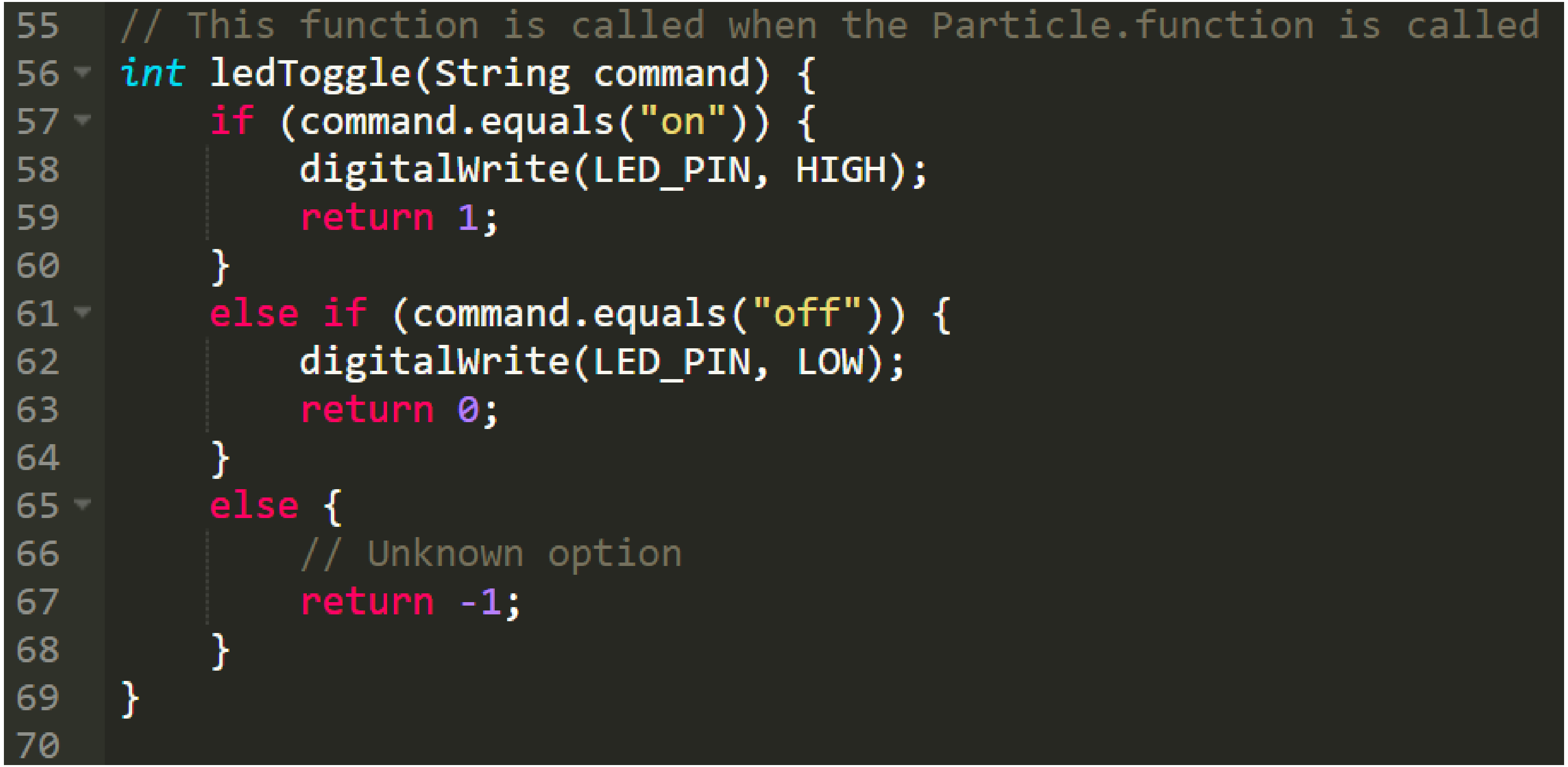
```
void setup()
{
    // First, declare all of our pins. This lets our device know which ones
    // will be used for outputting voltage, and which ones will read
    // incoming voltage.
    pinMode(LED_PIN, OUTPUT); // Our LED pin is output (lighting up the LED)
    digitalWrite(LED_PIN, HIGH);

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

    // We are also going to declare a Particle.function so that we can turn
    // the LED on and off from the cloud.
    Particle.function("led", ledToggle);
}

```

Code - Loop



```
void loop()
{
    // Check to see what the value of the photoresistor or phototransistor is
    // and store it in the int variable analogvalue
    analogvalue = analogRead(SENSOR_PIN);

    // This prints the value to the USB debugging serial port (for optional
    // debugging purposes)
    Log.info("analogvalue=%d", analogvalue);

    // This delay is just to prevent overflowing the serial buffer, plus we
    // really don't need to read the sensor more than
    // 10 times per second (100 millisecond delay)
    delay(100ms);
}

```

Code – LED Toggle Function



```
// This function is called when the Particle.function is called
int ledToggle(String command) {
    if (command.equals("on")) {
        digitalWrite(LED_PIN, HIGH);
        return 1;
    }
    else if (command.equals("off")) {
        digitalWrite(LED_PIN, LOW);
        return 0;
    }
    else {
        // Unknown option
        return -1;
    }
}

```

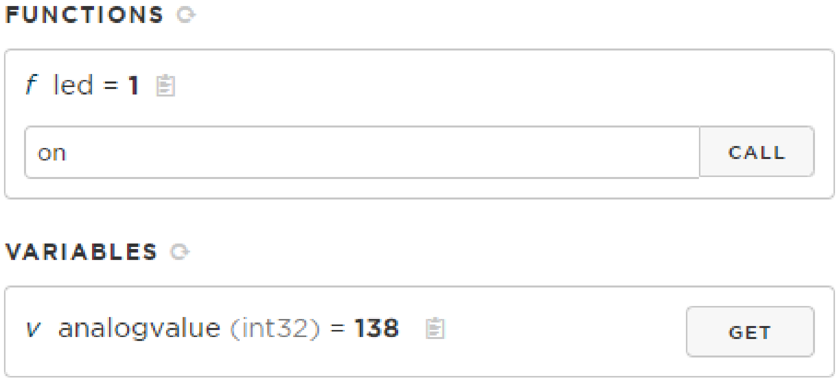
Code – Dashboard Interface

You can now:

- Turn the LED on and off
- See how much light is detected

Also possible:

- Creating events to register things happening and send notifications
- Using a mobile app to access the dashboard and notifications remotely



FUNCTIONS

f led = 1

on

VARIABLES

v analogvalue (int32) = 138

Full original tutorial and code are available at

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

Questions?

IEEE COMPUTER SOCIETY

Student Branch Chapter at the University of South Florida

UNIVERSITY of SOUTH FLORIDA

College of Engineering

Next Events

Conclusion