Lab Report: Particle Argon Peripherals

Lue Xiong

February 28, 2020

1 Introduction

The focus of the lab is to gain a basic understanding of what analog and digital signals are and how they relate to one another. For this lab, we are using resistors on an analog pin to manipulate the analog values, capture and convert them into digital values. Calculations will be done beforehand to predict the analog-to-digital (ADC) values with a margin of error percentage. The predicted ADC outputs will be compared with the actual readings coming from the Particle Argon. The idea is to get familiar by grounding our understanding with real-world instances.

2 Problem Statement

The lab can be summarized down to five main different parts, which include:

- RGB LED – Utilize Particle Argon's pulse-width modulation capability on pins *D2*, *D3*, and *D4*
- Piezzo Buzzer – Modify the code to play the same sequence as the "two-bits.mp4"
- Switches –
- Servo –
- Analog Temperature Sensor –

3 Process

3.1 Part One: RGB LED

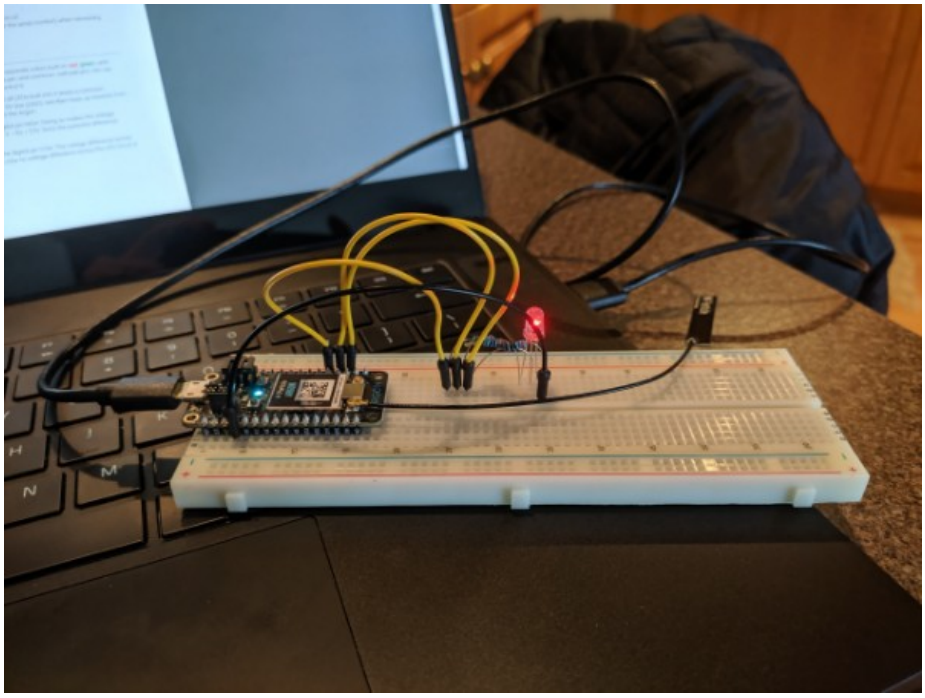

Figure 1: Particle Argon RGB LED

The code below shows how to create the fade effect on the RGB LED by using a function called **fade()**. The function starts with a loop that increments *brightness* by 1 from 0 to 255 with 10 millisecond delays in between. Upon reaching *brightness* value 255, another loop will run and decrement by 1 from 255 to 0 with 10 millisecond delays in between. It is in practice, a rather slow fade but it works nonetheless. Figure 1 is a photo of the components composition on the breadboard.

```
int bPin = D2;
int gPin = D3;
int rPin = D4;

void setup() {
  pinMode(bPin, OUTPUT);
  pinMode(gPin, OUTPUT);
  pinMode(rPin, OUTPUT);
}
```

```
void loop() {
  int delay_in_milliseconds = 10;

  // blink(bPin, delay_in_milliseconds);
  // blink(gPin, delay_in_milliseconds);
  // blink(rPin, delay_in_milliseconds);
  fade(bPin, delay_in_milliseconds);
  fade(gPin, delay_in_milliseconds);
  fade(rPin, delay_in_milliseconds);

  delay(1000);
}

void blink(int pin, int delay_in_milliseconds) {
  digitalWrite(pin, HIGH);
  delay(delay_in_milliseconds);
  digitalWrite(pin, LOW);
  delay(delay_in_milliseconds);
}

void fade(int pin, int delay_in_milliseconds) {
  for(int brightness = 0; brightness < 256; brightness++) {
    analogWrite(pin, brightness);
    delay(delay_in_milliseconds);
  }

  for(int brightness = 255; brightness >= 0; brightness--) {
    analogWrite(pin, brightness);
    delay(delay_in_milliseconds);
  }
}
```

3.2 Part Two: Piezzo Buzzer

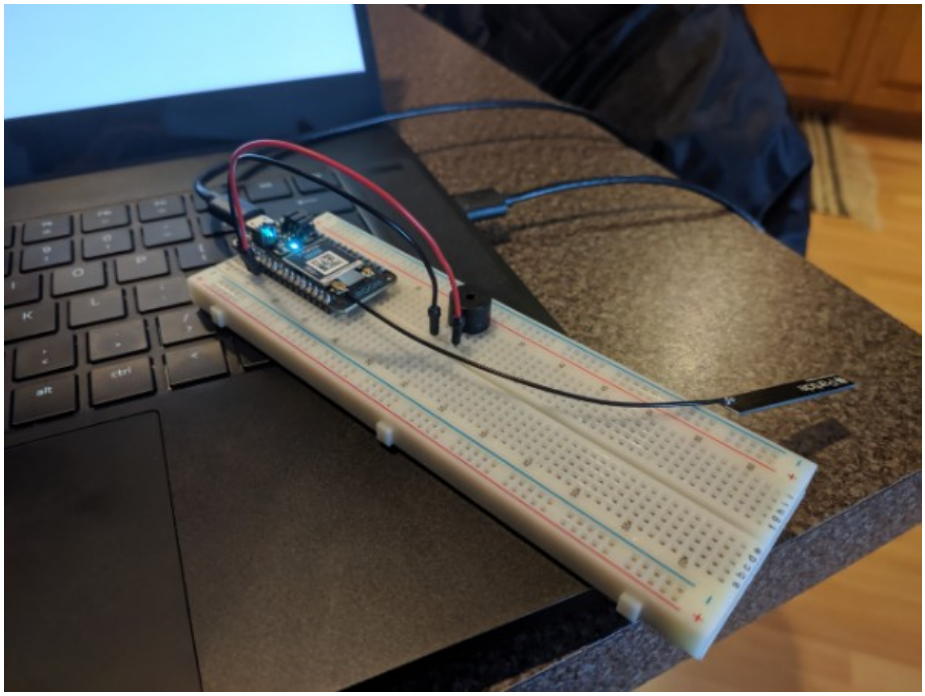

Figure 2: Particle Argon Piezzo Buzzer

The code below shows how to create the two-bits sound on the piezzo buzzer by using a function called **twoBitsBuzz()**. The function sends currents of multiple levels to the piezzo buzzer at arbritrary times in attempt to mimic the "two-bits.mp4" tune. Figure 1 is a photo of the components composition on the breadboard.

```
int buzzPin = A0;

void setup() {
  pinMode(buzzPin, OUTPUT);
  Serial.begin(115200);
}

void loop() {
  Serial.println("Buzzing...");

  // int delayInMilliseconds = 500;
  // buzzEffectOne(buzzPin, delayInMilliseconds);
  // buzzEffectTwo(buzzPin, delayInMilliseconds);
  twoBitsBuzz(buzzPin);
}
```

```
void buzzEffectOne(int buzzPin, int delayInMilliseconds) {
  analogWrite(buzzPin, 128);
  delay(delayInMilliseconds);
  analogWrite(buzzPin, 0);
  delay(delayInMilliseconds);
}

void buzzEffectTwo(int buzzPin, int delayInMilliseconds) {
  for(int i = 0; i < 256; i++) {
    analogWrite(buzzPin, i);
    delay(delayInMilliseconds * 0.2);
    analogWrite(buzzPin, 0);
    delay(delayInMilliseconds * 0.2);
  }
}

void twoBitsBuzz(int buzzPin) {
  analogWrite(buzzPin, 208);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(300);

  analogWrite(buzzPin, 160);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(50);

  analogWrite(buzzPin, 160);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(50);

  analogWrite(buzzPin, 160);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(75);

  analogWrite(buzzPin, 160);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(500);

  analogWrite(buzzPin, 208);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(300);

  analogWrite(buzzPin, 240);
  delay(100);
  analogWrite(buzzPin, 0);
  delay(3000);
}
```

3.3 Part Three: Switches

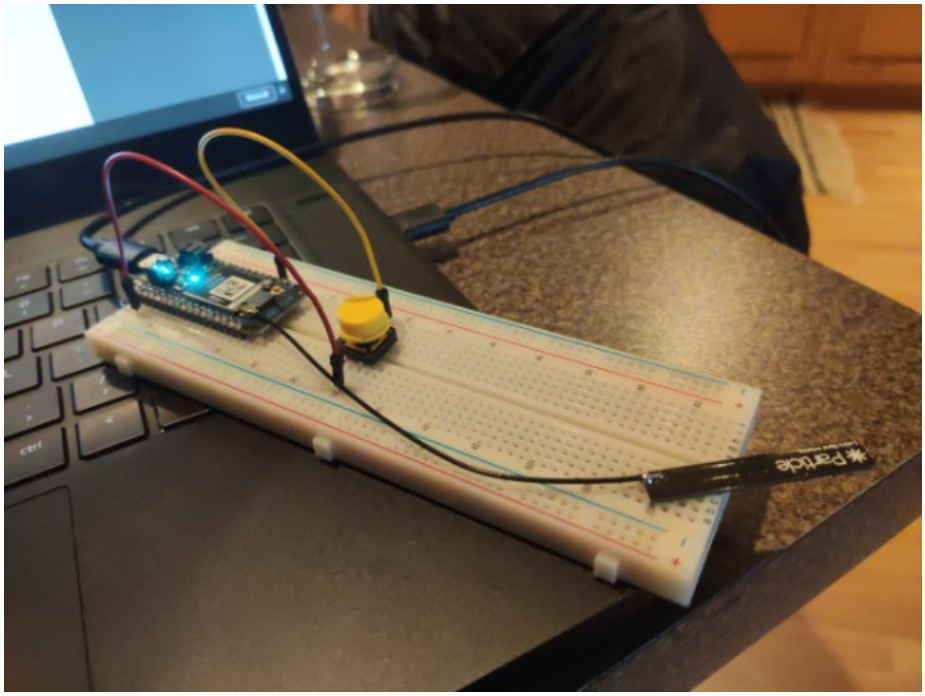

Figure 3: Particle Argon Switches

When the momentary switch is depressed, it engages metal to metal contact which creates a circuit for currents to run through. When it is released from being depressed, the circuit is broken. In this context, assuming the momentary switch is depressed, 3.3 volts will flow from the **3V3** pin to the **D2** pin and keep looping as such. Essentially this is what happens when **mom = 1**.

Parameter	Symbol	Min	Typ	Max	Unit
LiPo Battery Voltage	V_{LiPo}	+3.3		+4.4	V
Supply Input Voltage	V_{3V3}	+3.0	+3.3	+3.6	V
Supply Output Voltage	V_{3V3}		+3.3		V
Operating Current (uC on, Radio ON)	$I_{Li+\ avg}$		8	240	mA
Operating Current (EN pin = LOW)	$I_{disable}$		20	30	uA
Operating Current (uC on, Radio OFF)	$I_{Li+\ avg}$		TBD	TBD	mA
Sleep Current (4.2V LiPo, Radio OFF)	I_{Qs}		TBD	TBD	mA
Deep Sleep Current (4.2V LiPo, Radio OFF)	I_{Qds}		TBD	TBD	uA
Operating Temperature	T_{op}	-20		+60	°C
Humidity Range Non condensing, relative humidity				95	%

Figure 4: Particle Argon Operating Conditions

As shown on Figure 4 – more specifically the **Supply Input Voltage** parameter – the minimum voltage stated is 3.0 volts for inputs like momentary switches. Interestingly enough, the maximum voltage is 3.6 volts, which means that there is a +/- 0.3 volt difference from the typical 3.3 volt.

```
int momPin = D2;

void setup() {
  pinMode(momPin, INPUT_PULLDOWN);
  Serial.begin(115200);
}

void loop() {
  int momState = 0;
  momState = digitalRead(momPin);

  Serial.println("mom:␣");
  Serial.println(momState);

  delay(200);
}
```

The code above produces the outputs on the serial monitor in Figure 5. The **mom = 0** state means that there is no physical connection between the **3V3** and **D2** pin. The **mom = 1** means that the momentary switch is depressed to create a physical connection between **3V3** and **D2** pin.

Figure 5: Particle Argon Switches

3.4 Part Four: Servo

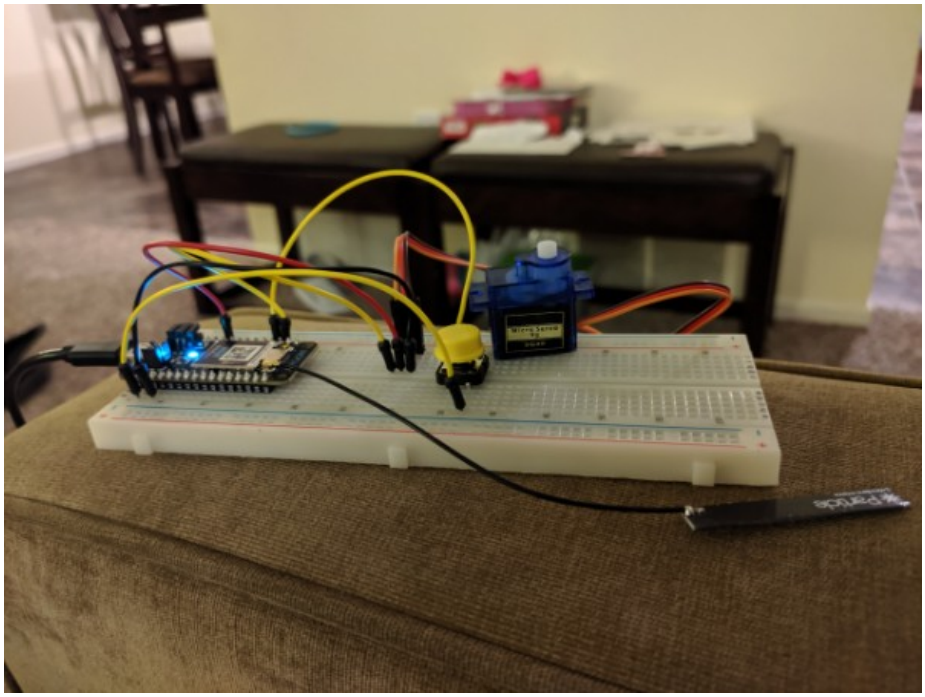

Figure 6: Particle Argon Servo

```
Servo servo;

int momPin = D3;
int position = 0;
```

```
void setup() {
  pinMode(momPin, INPUT_PULLDOWN);

  servo.attach(D2);
  servo.write(0);

  delay(500);

  Serial.begin(115200);
}

void loop() {
  int momState = 0;
  momState = digitalRead(momPin);

  delay(150);

  if(momState == 1) {
    if(position < 181) {
      servo.write(position += 20);
    }

    if(position > 180) {
      servo.write(position -= 180);
    }

    Serial.println("Position:␣");
    Serial.println(String(position));
  }
}
```

3.5 Part Five: Analog Temperature Sensor

4 Discussion of Results

5 Conclusion