## Introduction

Last week we learned about digital (0 or 1) input to read a switch state. This week we will learn about analog input and output. Analog is a continuously variable signal, with which you are probably more familiar. Time and space are considered continuous variables, at least on the macro-scale. If we measure time or space, we consider any value possible, not just 1 or 0, for example. The microcontroller is a digital device. So, even analog signals are digitized in order to be represented on the computer. *All analog signals on a microcontroller are voltage signals. We will learn about voltage or electric potential this term.*

An example of digitization is when we get the value of a pin that has been set up for analog input using `analog_pin = AnalogIn(board.A0)`. This defines pin A0 as an analog input. We read its value with `analog_pin.value`. The M4 Feather can read electrical voltages between 0-3.3 Volts only, and it digitizes these voltages using a 12-bit analog-to-digital converter. Twelve bits means the 0-3.3 Volt range is split into $2^{12}$ = 4096 integer values. If 0 is the lowest value (0 digital = 0 Volts), then 4096 is the highest value (4096 digital = 3.3 Volts). However, CircuitPython maps these values onto a 16-bit integer. So, even though our measurement device can only resolve 4096 values, the software maps this onto $2^{16}$ = 65536 values. This creates some redundant values, but for now we will simply assume that the microcontroller has 16-bit resolution. Every integer between 0 and 65535 corresponds to a voltage between 0 and 3.3 Volts. The smallest increment that our program can measure is

$$\frac{\Delta V}{\text{digital integers}} = \frac{(3.3-0)\text{Volts}}{65536} = 0.00005\frac{\text{V}}{\text{integer}}$$

Let's think a little about how we can count digitally to create analog values. We will use the first 10 bits, and a bit can be 0 or 1. Each bit corresponds to a power of two. Here are the bit values.

### Binary or Digital bits

| | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit 5 | bit 4 | bit 3 | bit 2 | bit 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| Power of 2 | $2^9$ | $2^8$ | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |
| integer | 512 | 256 | 128 | 64 | 32 | 16 | 8 | 4 | 2 | 1 |

Here is an example of digital counting. I will fill up the lowest bits in a way to count from 0 to 6.

### Binary or Digital Counting

| Integer | bit 10 | bit 9 | bit 8 | bit 7 | bit 6 | bit5 | bit 4 | bit 3 | bit 2 | bit 1 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

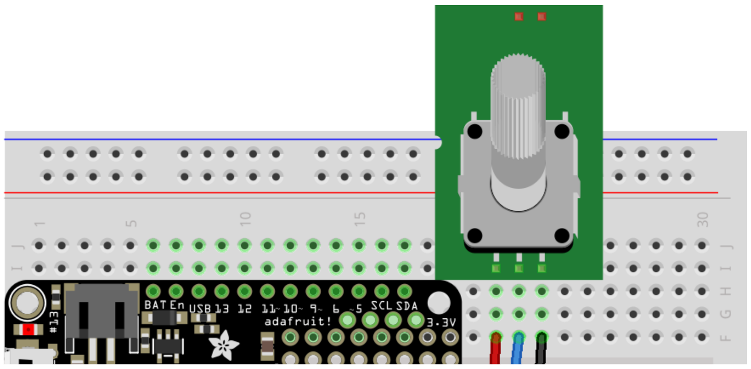
Here is how it works. You simply multiply the 1 or 0 in each bit times the bit value shown in the first table. Then, add them up. If the integer is zero, all the bits have 0 in them, and the sum is 0. If the integer is 1, bit 1 = 1. The bit 1 value is $2^0$=1. So, 1x1=1. If the integer is 6, we have 1 in bits 2 and 3 and zero in all other bits. Bit 2 value is $2^1$=2. Bit 3 value is $2^2$=4. So, the sum is 1x2+1x4 = 6. You try.

- **Make a table like the one above that counts from 6 to 10 in binary.**
- **We have the binary number 1000010001. What is the integer? What is the M4 Feather voltage assuming the 0-3.3 Volt range?**
- **What digital integer and binary number corresponds to 1.00 Volts?**
- **Determine the digital integers and binary values for 0 to 3.3 Volts in 0.5 Volt increments, i.e., 0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.3 Volts. Keep in mind the digital integer and binary values are independent of a voltage. The voltage is a mapping of a measurement onto the digital values due to analog to digital conversion.**
- **How would the measured voltage resolution change if there were fewer bits available in the digital conversion?**

## Part 1: Analog Input from Potentiometer

To explore analog input signals, we will first divide the 3.3 Volts the microcontroller outputs so that a portion is measured on analog pin A0. This division occurs by using a variable resistor or potentiometer (rotary knob). We'll learn more about resistors in chapters 18 and 19. The figure below shows how the potentiometer works. One side is connected to 0 Volts (ground or GND on the microcontroller). The other side is connected to 3.3 Volts on the microcontroller (Vcc in the diagram). A moveable connection is connected to the microcontroller input A0 (output in the diagram). As the knob is turned, the voltage varies according to where this connection is. Closer to Vcc is closer to 3.3 V. Closer to GND is closer to 0 V.

Below is an image of how to wire the circuit.

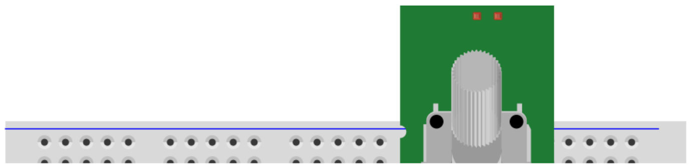


## Part 2: Analog Output to Buzzer

To explore analog output, we will control a buzzer with different sounds. The analog output from the M4 Feather is also not truly analog since it comes from a digital device. However, we will ignore this for now since it is beyond the scope of our experiment. To produce analog output for this experiment, we will use an AudioOut function in CircuitPython. This is done first by an import and then by defining the pin we want to ouptut the audio signal. On the M4 Feather only A0 and A1 can be used for audio output. We are using A0 for the potentiometer. So, we will use A1 for audio.

```
import audioio
audio = audioio.AudioOut(board.A1)
```

There's a bit more to it than that, but we will simply use pre-written code to produce sounds. You will be asked to modify it next week for a different experiment. Below is how to wire the circuitry for both the potentiometer and buzzer.



## Procedure

### Part 1: Analog Input from Potentiometer

The code for the potentiometer is below. This code will read the variable voltage. Test that it works as you turn the rotary knob.

```
"""CircuitPython Essentials Analog In example"""
import time
import board
from analogio import AnalogIn

analog_pin = AnalogIn(board.A0)

while True:
    print(analog_pin.value) # the digital integer
    print(analog_pin.value * 3.3 / 65536) # the converted voltage
    time.sleep(0.1)
```

1. **Explain how the code creates converts between an integer based on 16 bit resolution and a voltage being read.**
2. **Suppose you want to control the amount of time an LED remains lit. Can you connect an LED as we did last week in lab make it stay on for 0 to 3.3 seconds using the voltage of the potentiometer input before turning off for 1 second?**

### Part 2: Analog Output to Buzzer

Okay, now let's set up analog output to make a buzzer make sound based on the value of the potentiometer. Our goal is to make the frequency of the buzzer depend on the voltage of the potentiometer. If the potentiometer reads zero volts, the audio frequency will be 100 Hz. If the potentiometer reads 3.3 volts, the frequency will be 3400 Hz. Therefore, the conversion is

frequency = potentiometer * 1000 + 100

which is the equation of a line. To begin, let's play a single sound at 440 Hz.

```
import audiocore
import audioio
import board
import array
import time
import math

# Generate one period of sine wav.
frequency = 440
length = 8000 // frequency
sine_wave = array.array("h", [0] * length)
for i in range(length):
    sine_wave[i] = int(math.sin(math.pi * 2 * i / length) * (2 ** 15))

audio = audioio.AudioOut(board.A1)
sine_wave = audiocore.RawSample(sine_wave)
audio.play(sine_wave, loop=True)
time.sleep(5)
audio.stop()
```

1. **You should edit the frequency to ensure it behaves as you expect. Explain what you do.**
2. **Modify the code so that frequency gets a value based on the potentiometer. Explain how do you do this?**
3. **How does the buzzer sound change with this frequency? Be specific.**

## Next Week

There is some evidence that humans respond differently to visual and auditory signals. For example, read this article https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4456887/. You measured your reaction time to the visual cue of the LED. We want to know whether reaction time to hearing the buzzer is statistically different from the LED visual cue. Review last week's Arduino code, and propose "pseudo code" for how we might program the buzzer. Pseudo code is plain English language that describes step-by-step what the program should do.

Last week, we also saw how to determine whether a set of data are statistically similar. Explain how we will determine whether visual and auditory reaction times are the same or different. As a reminder, we use the average and the standard error (also known as the standard deviation of the mean).

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}}$$

$$SDM = \frac{\sigma}{\sqrt{N}}$$

To summarize, please prepare the following for next week.

1. **Write pseudo code for programming the buzzer to randomly turn on.**
2. **Explain how to compare data from visual and auditory reaction times.**
    - **What makes data statistically distinguishable?**
    - **What makes data statistically indistinguishable?**
    - **Is there a hard line between distinguishable and indistinguishable? Explain.**

Last modified: Thursday, August 17, 2023, 4:51 PM