Photon Analog IO -- Output

44-440/640-IoT

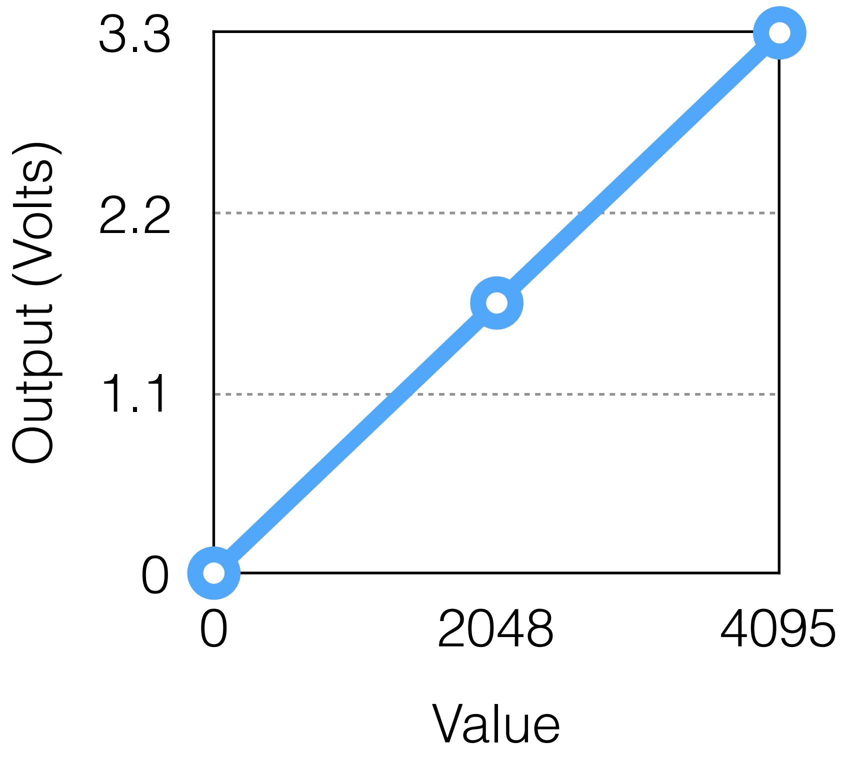
Objectives

- Students will be able to:
    - explain the difference between analog output and pulse width modulation
    - explain the difference between buzzers and speakers
    - explain the purpose of safe mode and how to put your Photon into safe mode
    - define pulse width modulation and duty cycle
    - perform various output operations using PWM

True Analog Output: DAC

- On 2 pins, **analogWrite()** can generate a genuine* analog output, a voltage that varies from 0.0V - 3.3V as the digital value supplied varies from 0-4095
- DAC = Digital to Analog Converter
- void analogWrite(int *pin*, int *value*)
  - *pin* - true analog works on pins **DAC1** (A6, DAC) & **DAC2** (A3)
  - *value* - 0-4095
- Q: What could we use this for? Ponder this for a moment before looking at the next slide.

*Point to ponder: is this *really, truly* analog? 🤔



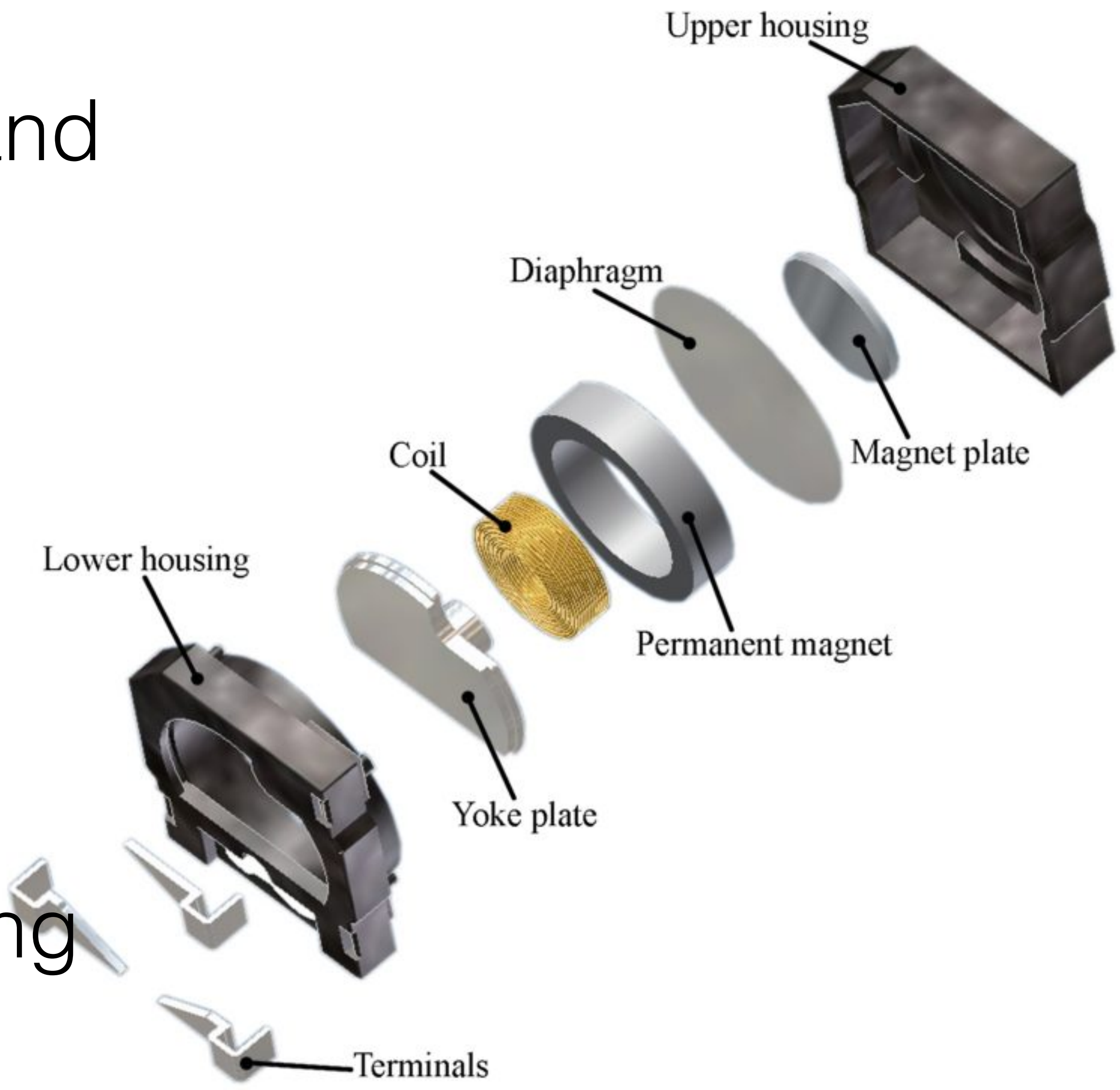
Uses of True Analog Output: DAC

- Playing music 🙂
- Driving motors

The Buzz on Buzzers and Speakers

- A buzzer works with direct current. Apply a voltage, and a an oscillating transistor circuit causes it to buzz.

- A speaker has a magnetic coil that acts on a metal plate. As electricity runs through the coil, it pushes and pulls the plate, vibrating the air and creating sound



Magnetic Buzzer Construction

A Modicum of Code

```
// https://community.particle.io/t/photon-play-sampled-music-wav-file-using-dac/19849/4

//SYSTEM_THREAD(ENABLED)
SYSTEM_MODE(MANUAL)

#include "SparkIntervalTimer/SparkIntervalTimer.h"
#include <math.h>

const int MIDPOINT = 2048;                  // zenter point for DAC
const int MAX_VOL  = 1600;                  // amplitude around (+/-) center point (no more than 2047! to avoid clipping)
const int SAMPLE_PERIODE = 1000000/44100;   // µs between samples (44.1kHz)
const int SAMPLES = 50;
uint16_t  sine[SAMPLES];                    // one periode over 50 samples is aprox. 880Hz

volatile int r = 0;                         // sample to play on right channel
volatile int l = 25;                        // sample to play on left channel (slightly offset)

IntervalTimer tPlayer;

void setup() {
  pinMode(DAC1, OUTPUT);
  pinMode(DAC2, OUTPUT);

  for (int i=0; i < SAMPLES; i++)
  { // precalc one periode sine wave
    double x = 2.0 * M_PI * i / SAMPLES;
    sine[i] = MIDPOINT + MAX_VOL * sin(x);
  }

  tPlayer.begin(playSample, SAMPLE_PERIODE, uSec);
}

void loop() {
}

void playSample()
{
    analogWrite(DAC1, sine[r++]);
    analogWrite(DAC2, sine[l++]);

    r %= SAMPLES;
    l %= SAMPLES;
}
```

https://community.particle.io/t/photon-play-sampled-music-wav-file-using-dac/19849/3

https://github.com/pkourany/SparkIntervalTimer

ICE: A Spectacular Speaker

1. Create a new VSC project, Music Maestro
2. Copy the code from the previous slide, which came from here: https://community.particle.io/t/photon-play-sampled-music-wav-file-using-dac/19849/4
3. Explain to a partner what the code is doing
4. Install the SparkIntervalTimer library
5. Add a buzzer: + to DAC, - to ground
6. Run the program (and listen very, very carefully)
7. Explain what tPlayer.begin() is doing
8. Print out sine

The code disables the connection to the cloud - you will have to put it into safe mode before refreshing.

1. Hold down reset & setup;
2. Release reset until it flashes magenta
3. Release setup

Safe mode disables the firmware so you can flash new software

A Non-Use of True Analog Output: DAC

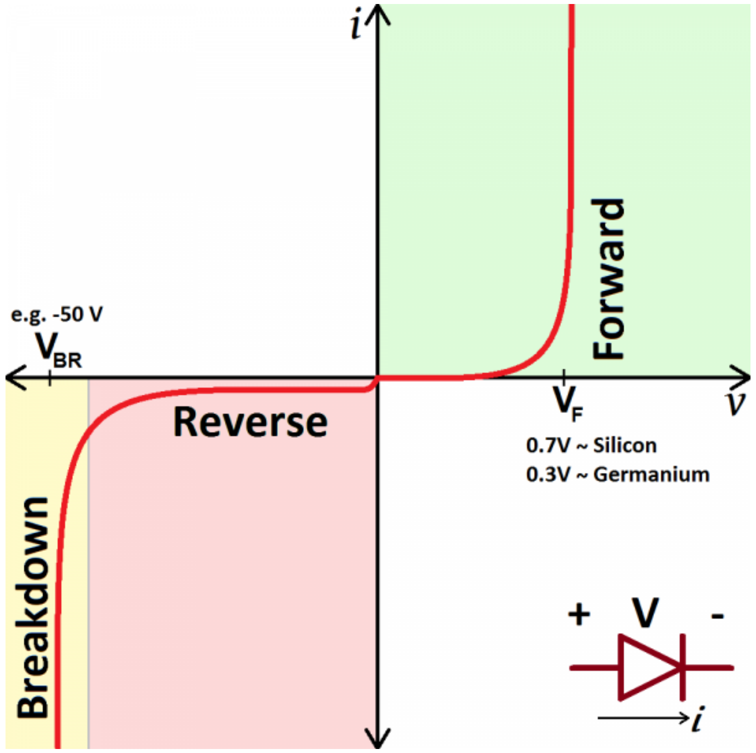
- **Not** dimming LEDs 🙁
- Why not LEDs? Review what you know about LEDs before looking at the next slide.

The Problem with True Analog Output

- For an LED to light, voltage must exceed Vf (approx 2.0V).

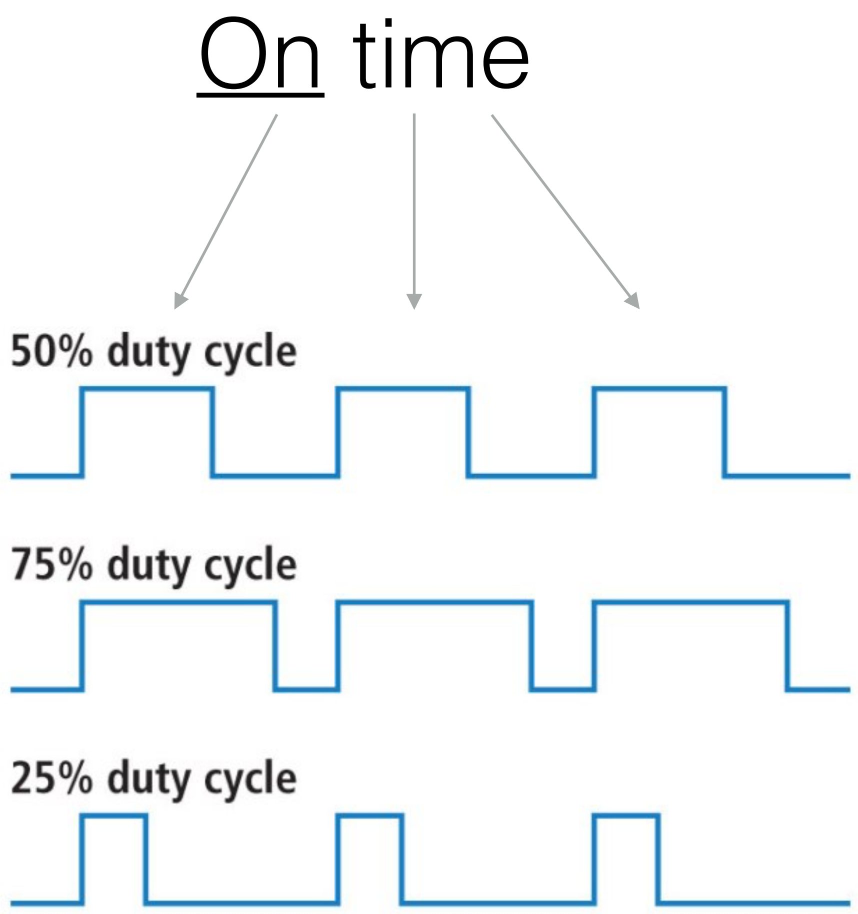
- Suppose we wanted an LED to be 20% brightness.

- Setting the brightness to, say, 20%, would mean setting the output voltage to 3.3 * 0.20 = 0.66V, i.e., the LED would stay dark.



Pulse Width Modulation

- Pulse width modulation (PWM) is a **conjuring trick** that allows a digital signal to simulate an analog signal.
- Uses include controlling the brightness of an LED, driving a motor at various speeds, and controlling servos
- PWM works because, while a digital signal by definition can only be LOW or HIGH, the **time** that it occupies those two values can be controlled in an analog fashion.
- **On time** is the *time* when the signal is HIGH*.
- **Duty cycle** is the *percentage of the time* that a signal is high.
- This is a conjuring trick is because the LED is either on or off — the voltage is either 3.3V or 0.0V — there is no intermediate brightness. But if it is on for only, say, 25% of the time, it will appear just 25% as bright.
- The frequency of the signal needs to be high, otherwise you will be see the flickering.The Photon default is 500 Hz.



On time

50% duty cycle

75% duty cycle

25% duty cycle

*"on time" (as opposed to "on time") is also something that all students must usually be, in this course, in order to achieve a good grade!

A PWM Example

- Suppose we wanted 20% brightness.
- With PWM, setting the duty cycle to 20% would mean that the output voltage would be 3.3V 20% of the time: it would appear to be 20% of its max brightness.
- To put it another way, the *average* voltage is 3.3 x 0.20 = 0.66 V — but as discussed a few slides back, if we applied 0.66 V, the LED would not light

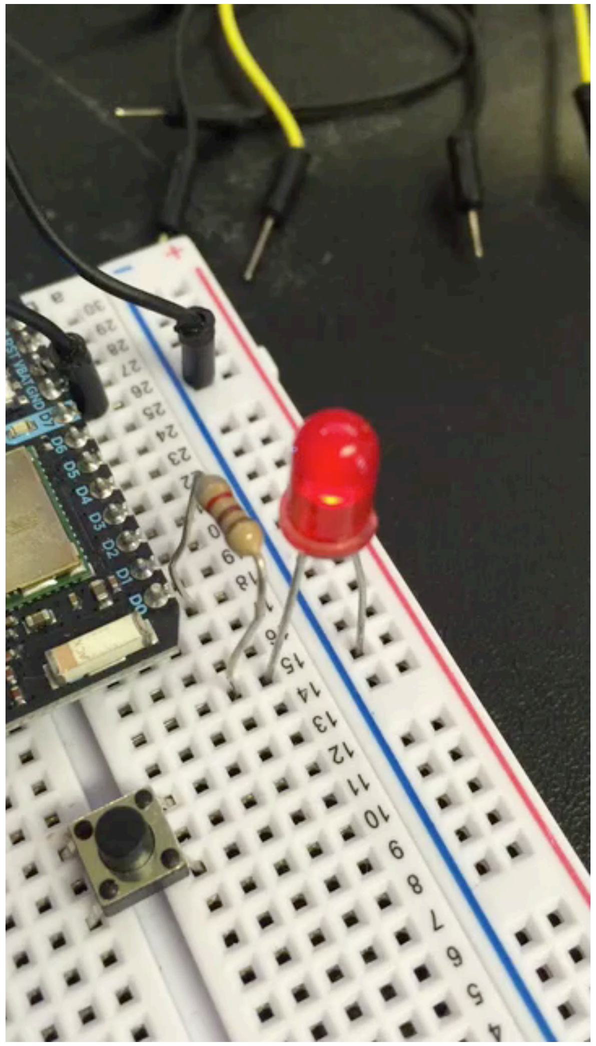
PWM and the Photon

- **pinMode(*pin*, OUTPUT)** // must be called **before** analogWrite()
- **void analogWrite(*pin*, *value* [,*frequency*])**
  - *pin* - PWM works on pins **D0-D3, A4, A5, WKP, RX, and TX** (but A5/D2 and A4/D3 are linked: they cannot both be used for PWM output at the same time)
  - *value* - duty cycle - 0 (always off) to 255 (always on)
  - *frequency* - 1Hz - 65535 Hz (default is 500 Hz)
- When analogWrite() is called, it will generate a square wave with the indicated frequency/duty cycle until another call to analogWrite() (the same behavior as digitalWrite())

Ignore pin assignments at your peril 😱

ICE: A Half Bright Idea?

- Create a circuit with one LED connected to D0. Using analogWrite(), cause the LED to appear at 50% brightness.
- Experiment 1: Modify your code so that the frequency is 1 Hz. What do you expect to see? Flash your Photon and verify (or debunk) your prediction
- Experiment 2:Repeat the above (code, predict, flash), this time with 20 Hz.



ICE: A Half Bright Idea? - Solution

```
int analogAmount = 128;
int externalLED = D1;

void setup() {
   pinMode(externalLED,OUTPUT);
   analogWrite(externalLED, analogAmount /, * 1 */);

}

// the loop can remain empty, because the effect of analogWrite()
// persists until the program is restarted
void loop() {

}
```

What Does This Do?

```
int ledRed = D0;
int ledGreen = D1;
int redBrightness = 30;
int greenBrightness = 100;
int deltaBrightness = 10;
int switcher = D4;

void setup() {
    pinMode(switcher,INPUT_PULLUP);
    pinMode(ledRed,OUTPUT);
    pinMode(ledGreen,OUTPUT);
    analogWrite(ledRed, redBrightness);
    analogWrite(ledGreen, greenBrightness);
}

void loop() {
    // each time you push a switch, increase the red and green brightness
    if(digitalRead(switcher) == LOW){
        redBrightness += deltaBrightness;
        greenBrightness += deltaBrightness;

        redBrightness %= 255;
        greenBrightness %= 255;

       analogWrite(ledRed, redBrightness);
       analogWrite(ledGreen, greenBrightness);
  }
}
```

Exercises

- Set up 3 LEDs, and have them cyclically vary in brightness: red from 0-100, blue from 100-0, green randomly.

Resources

- https://learn.sparkfun.com/tutorials/analog-vs-digital
- https://learn.sparkfun.com/tutorials/pulse-width-modulation
- http://www.eetimes.com/document.asp?doc_id=1274544
- https://www.sparkfun.com/products/7950
- https://community.particle.io/t/photon-play-sampled-music-wav-file-using-dac/19849/3
- https://docs.particle.io/reference/firmware/electron/#analogwrite-pwm-
- https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor
- http://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf
- https://www.tigoe.com/pcomp/code/controllers/input-output/analog-output/