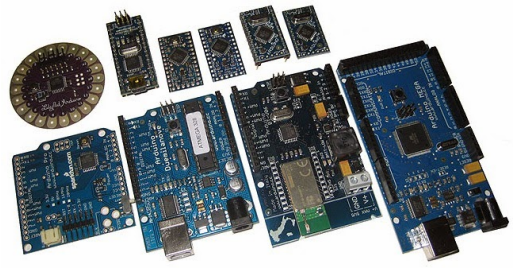
Arduino

Arduino is a hardware and software company that designs a variety of microprocessors and controllers. They are best known for their AVR-based boards such as the Arduino Uno, Nano, Mega, and more. Developed by Atmel in the late 90s, the AVR series of microcontrollers is a cheap and compact microcontroller solution that can be useful in a wide variety of electronics projects. AVR chips can be difficult for beginners to program on their own, but Arduino boards make it much easier to program and interface with them. Since the project is open source, other companies can make their own versions of Arduino boards, sometimes with extra features.

An ATMEGA328P, as used in the Arduino Uno



A few Arduino-based boards

The Arduino can send and receive signals using the connections on the board. On the Arduino Uno, you may interface with the controller using 14 digital connections and 6 analog connections. The digital pins can be used to send or receive high or low signals, such as receiving a high signal from a button press or sending a high signal to turn on an LED. The analog pins, however, can be used to measure the voltage of an incoming signal or, in certain cases, send a specific voltage using a PWM signal.

Analog-to-Digital Conversion

Analog vs. Digital:

Analog data is data which exists as a continuous signal (denoted by a continuous sine wave) that represents physical measurements. The world is “in” analog; that is information and data are constantly flowing and changing. Analog represents more detailed and refined information. However, analog is very susceptible to noise (interference) and also tends to have a lower quality signal than equivalent digital signals.

Digital data is data which exists only in binary (0’s and 1’s) denoted by a square wave. This means that the signals are time-separated and non-continuous. This type of signal is extremely

easy for computers to read and write in. Digital represents discrete numbers (0 and 1) to represent information. Digital signals are less susceptible to noise since there are only two states which they are represented in and generally have better quality signals.

ADC:
An Analog-to-Digital Converter (commonly abbreviated ADC) is a device which can convert varying analog voltages into digital signals so that they can be more easily read by digital devices. This allows us to collect information from real-world instruments and then analyze it digitally.

How does an ADC work (in an arduino)?:
The ADC in an arduino uses the analog voltage to charge an internal capacitor, then measures the time it takes to discharge across an internal resistor. The microcontroller monitors the number of clock cycles that pass before the capacitor is completely discharged. The number of cycles is returned by the microcontroller in the form of a binary number. This technique is called successive approximation, since it takes several clock cycles to approximate the input voltage.

I2C will not be discussed in this workshop

<u>Potential Applications</u>

1. Get Temperature
    a. A thermistor is a variable resistor whose resistance changes depending on its temperature
    b. Measure the voltage coming from a thermistor and use that to calculate the temperature of the thermistor
    c. Maybe use that information to trigger some other operation with the Arduino
2. Get the position of a dial
    a. A potentiometer is a variable resistor whose resistance changes depending on the position of a dial or fader
    b. Measure the voltage coming from the pot to calculate the position of the fader/dial
    c. Could use that information to change some parameter (ie. Volume, brightness, etc)
3. Take in audio information
    a. Take an audio signal and process it with the Arduino
    b. Could make an audio visualizer or sound level meter by interpreting the data from the adc

<u>Deliverables</u>

1. Pot example circuit
    a. Simple pot to analog in circuit
    b. Code which outputs percentage on serial out
2. Volume meter circuit

a. Mic in to analog in circuit
b. Code gets volume from signal and lights LEDs accordingly

Start with internal adc, can be migrated to adafruit later, if we end up using the Adafruit adcs