Lecture 03.10

Analog Signals and Data Acquisition

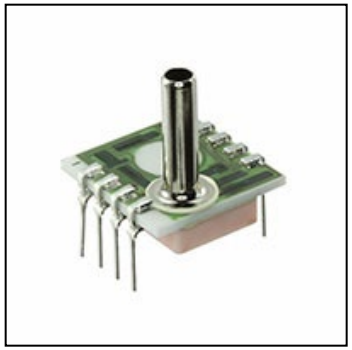
Sensors

- Our ultimate goal is to study some phenomena by making measurements of some physical quantity during the balloon flight
- So we need to:
  1. Measure something
  2. Convert that to an electrical signal
  3. Digitize the signal
  4. Record the digitized measurement
- We will use some kind of sensor to accomplish steps 1 and 2
- In this case, a sensor is a device that measures a physical property and generates an output signal corresponding to the measured quantity

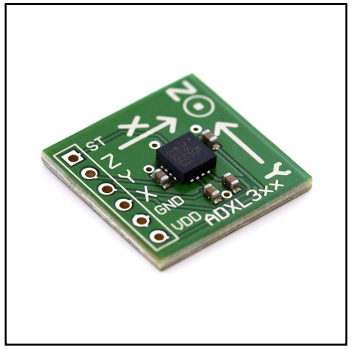
Example Sensor Inputs: Physical Variables

- Temperature
- Pressure
- Humidity
- Light Intensity

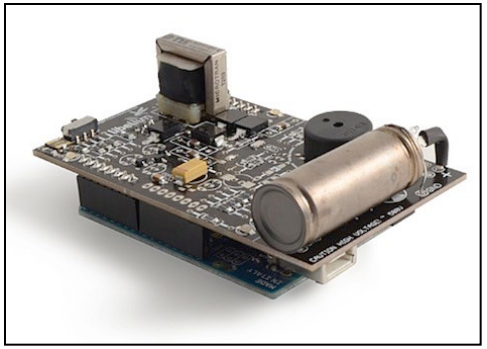
- Radioactive Particles Counts
- Acceleration
- Altitude
- Electric or Magnetic field strength



Pressure Sensor



Accelerometer



Geiger Counter

Electrical Signal Outputs

- Example output signals, the electrical signal can vary in:
  - Voltage (Most common)
  - Current (ex. Photodiodes)
  - Variations with time (common when transmitting over radio)
    - Pulse Width
    - Frequency
    - Number of pulses (like a radiation counter)

- The most common output you will see is an analog voltage output, just like the diode temperature sensor we built

Transfer Function and Span

Transfer Function: The mathematical relationship between the measured quantity and the output signal

Span: The width of possible values the sensor is being used for (both in physical quantity and voltage)

Example Diode Temperature sensor:

- Transfer Function:

V_{diode}=-0.0014*T+0.7724V

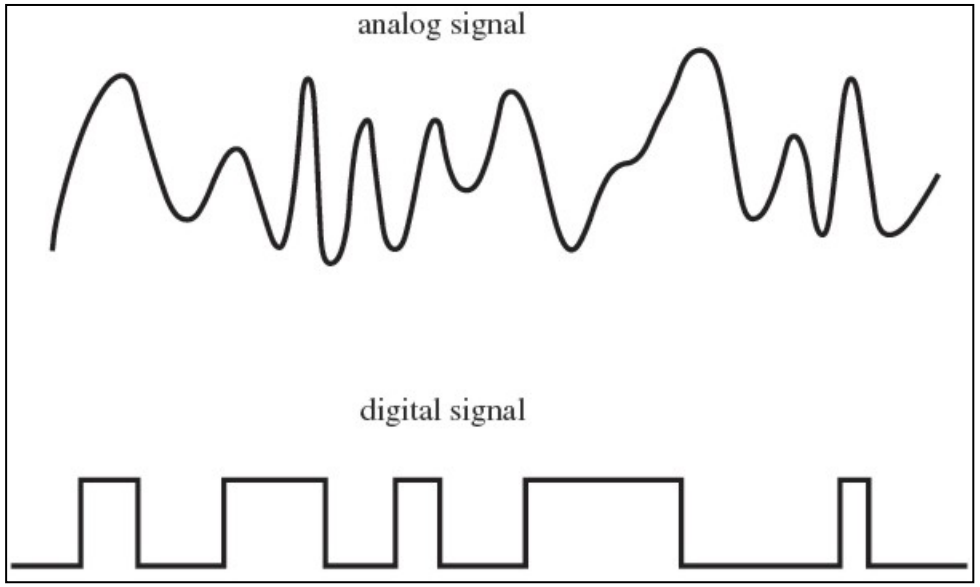
Notice this is the reverse of the calibration equation which would go from voltage to Temperature

- Minimum Value: -50 °C (0.842V)
- Maximum Value: +50 °C (0.702)
- Temperature Span: 100 °C
- Voltage Span: 0.14V

Digital vs Analog

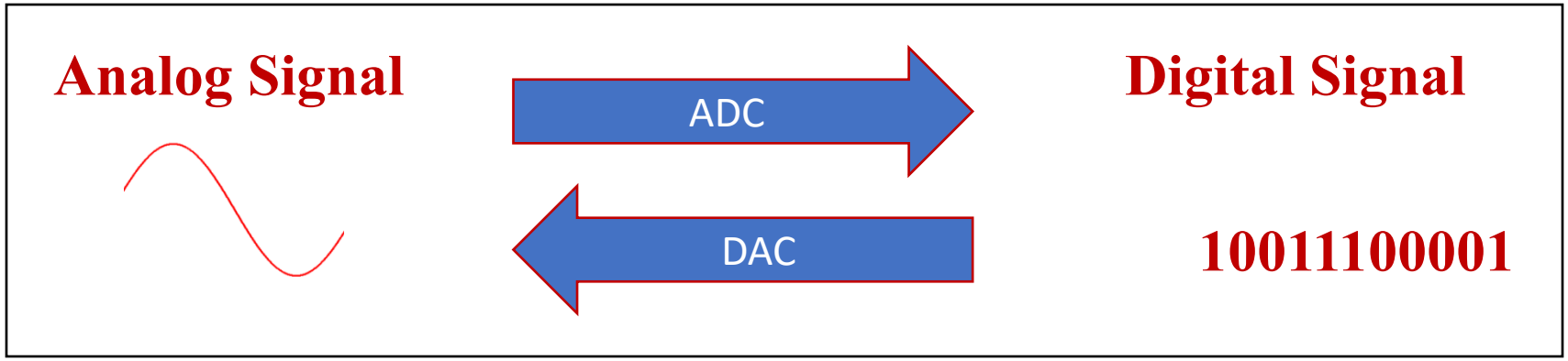
Our temperature signal is an analog signal

- It is smooth & continuous
- So it can take any value between the minimum and maximum
- Even though we can only measure ~0.001, we assume it takes values in between 0.001 and 0.002
- This is contrasted with Digital Signals
  - Can only take values from a finite set of values
  - For example: 0V or 5V



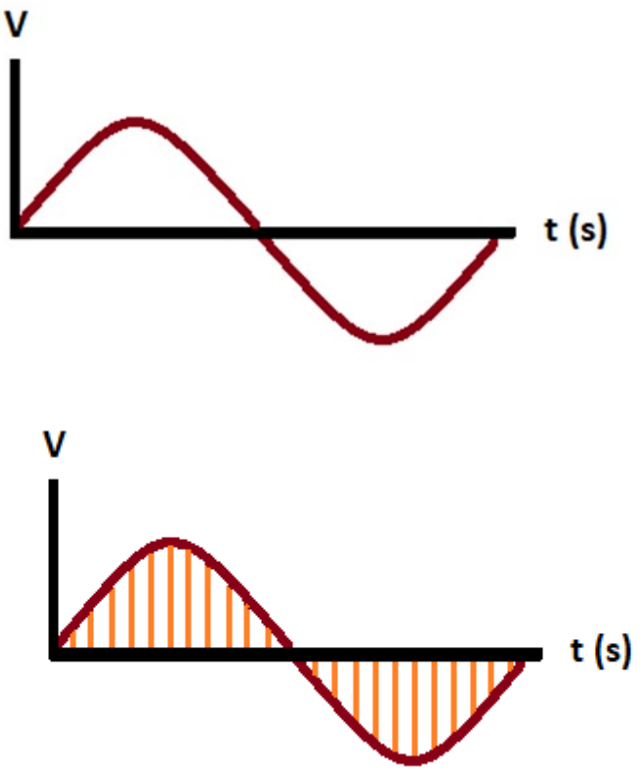
Converting between Analog and Digital Signal

- We need a way to convert the analog voltage into a binary number so we can store or manipulate the data on the Arduino.
- The device that does this is called an Analog to Digital Converter or **ADC**
- There is also the reverse, Digital to Analog Converter or **DAC**



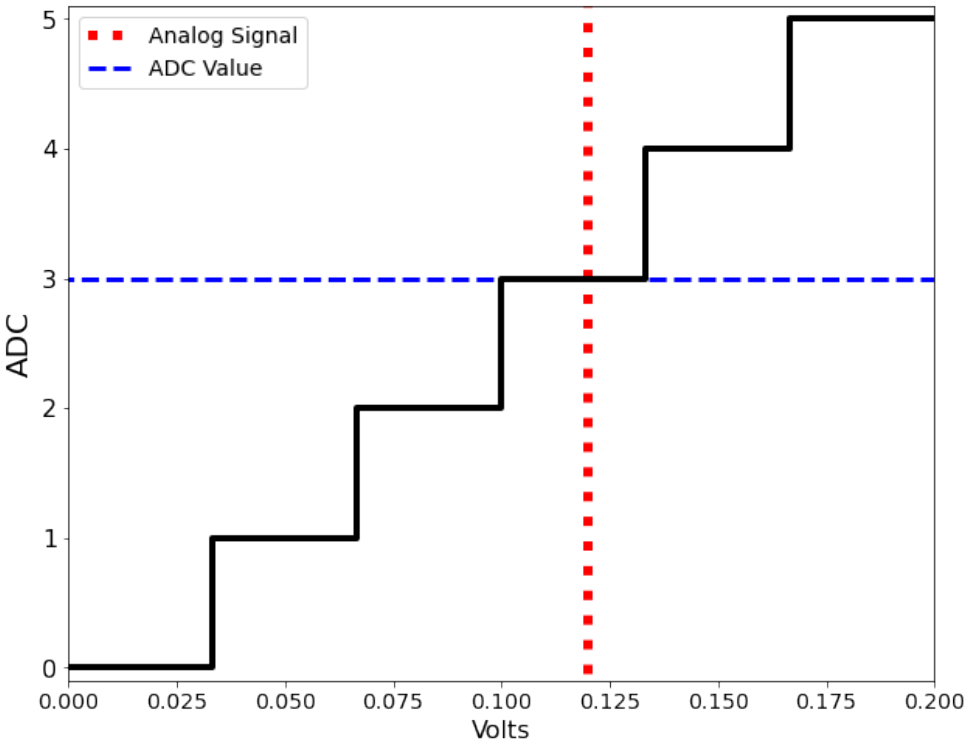
How Does ADC Work

- We begin with a varying analog signal; in this case, a sine wave
- At a single point in time the voltage is measure, this is usually called a sample
- Each sample gets converted to a number based on the amplitude of its voltage



ADC Conversion

- The takes its voltage range and splits it up into discrete sections (called bins) that are numbered 0, 1, 2, 3, etc.
  - Normally this is linear and increasing with voltage
- The sample voltage is compared to the minimum and maximum values for each bin to find out which one it falls in
- ADC then sends that bin number to the microcontroller or whatever device is trying to read the voltage
- The returned value is just a unitless number
  - Usually this will be called ADC units or Bins



ADC Resolution

- The number of integer bins is called the resolution of the ADC
- Typically, this is given in bits; the Arduino Mega has a 10 10-bit ADC
- To calculate the number of ADC bins, use the formula 2^N, where N is the number of bits
- $2^{10} = 1024$, the Arduino ADC has 1024 possible values
  - Typically, the first bin is numbered 0 so the ADC values are from 0-1023

Voltage Range and Resolution

- We next need to know the voltage range of the ADC
- Arduino Mega ADC is 0-5V
  - So the ADC should read 0 ADC units at 0V and 1023 at 5V
  - 0 and 5 values are not exact, might actually be something 0.002V and 4.997V, for example
- Using the voltage range and ADC resolution, we can calculate the voltage resolution – The smallest voltage change you can measure with the ADC
- $R_{\mathrm{v}} = \frac{V_{max} - V_{min}}{N_{\mathrm{bins}}}$ $R_{\mathrm{v}} = \frac{5V - 0V}{1024\ ADC} = 0.004883 \frac{V}{\mathrm{ADC}}$
- So bin 0 would be 0V to 0.004883V, 1 would be 0.004883V to 0.009766, etc.

Physical Unit Range and Resolution

- The range and resolution are important because they set the limits on what you can measure with a combination of a sensor and ADC
- Using the calibration equation, we can calculate the minimum, maximum, and resolution in physical units
- Consider the diode we talked about before

  V_{diode}=-0.0014*T+0.7724V, we need the inverse:

  T= -714.3 *V_{diode}+551.7C
- If we multiply the slope by the voltage resolution, we can get the temperature resolution (since we are talking about a change up or down, we do not care about sign)

$$.004883 \frac{V}{\text{ADC}} * 714.3 \frac{^\circ C}{V} = 3.48^\circ C$$

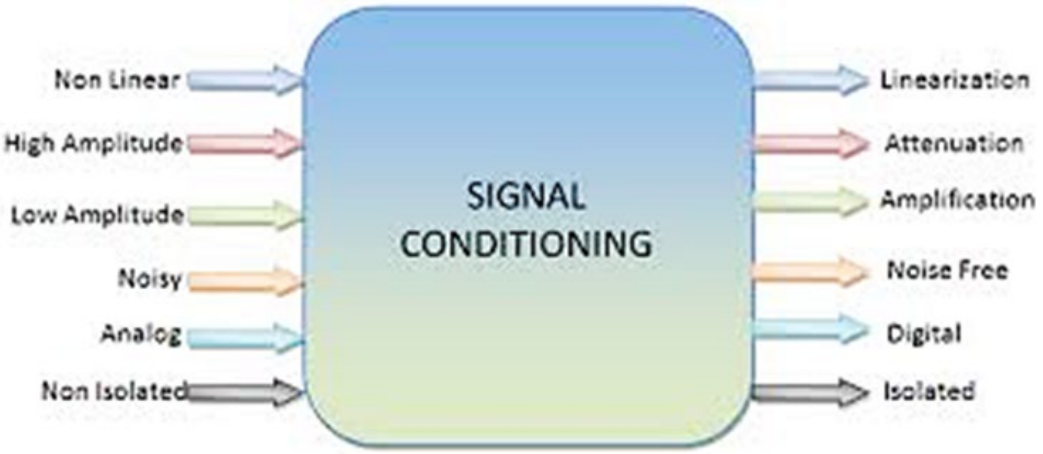
- If we plug in 0V and 5V we can predict the minimum and maximum temperatures
- 551.7 $^\circ C$ at 0V and -3019.7 $^\circ C$ at 5V (we would run into issues physical issues with the diode before reaching those temperatures so we would be limited by that)

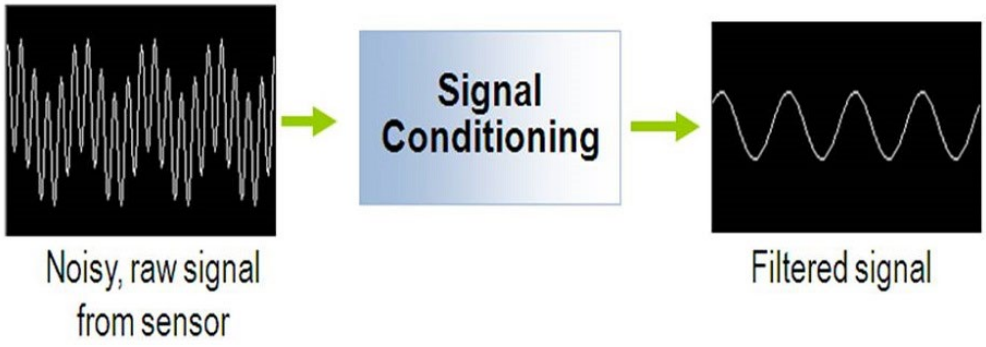
Range and Resolution vs Requirements

- For our actual payload, you will establish a set of minimum requirements for each of your sensors based on goals and objectives
- A typical temperature requirement a range of -50 $°C$ to +50 $°C$ with a resolution of at least 1 $°C$
- The minimum temperature of -3000 $°C$ exceeds the requirement of -50 $°C$
- The maximum temperature of 714 $°C$ exceeds the requirement of +50 $°C$
- But the resolution of ~3.5 $°C$ is worse than the 1 $°C$

Signal Conditioning

- These and other issues are solved by **Signal Conditioning Circuits**
- Signal Conditioning circuits are between the Sensor and the ADC
  - Sometimes the sensor and signal conditioning are all combined in a single component
- The amplifying circuits we built are a type of signal conditioning





Conditioned Range and Resolution

- Does the amplified diode built meet the requirements, take an example amplified calibration

  T= -7.673 *V_{out}+52.88 °C

- Resolution

$$.004883 \frac{V}{\text{ADC}} * 7.673 \frac{^{\circ}C}{V} = 0.0375^{\circ}C$$

- 52.88 °C at 0V and 14.5°C at 5V
- The minimum temperature of 14.5 °C does not meet the requirement of -50 °C
- The maximum temperature of 52 °C exceeds the requirement of +50 °C
- The resolution of ~0.0375 °C is better than the 1 °C
- We would want to adjust the amplifying circuit until we meet all 3 requirement (that is why your gain and offset should be adjustable)
- Notice how you must balance the range and resolution, a lower gain gives a worse resolution but a wide range

Sampling rate

- Besides the range and resolution, we also need to consider how often the ADC should sample the sensor
- This is called the sample rate
  - The ADC requires a small amount of time, called the Conversion Time, for each measurement.
  - That sets the upper limit on the sample rate
  - To select a sample rate, you determine the fastest change you expect to see, and then double that for you sample rate
  - The doubling is called the Nyquist-Shannon sampling theore,

Choosing Sampling Rate: Example

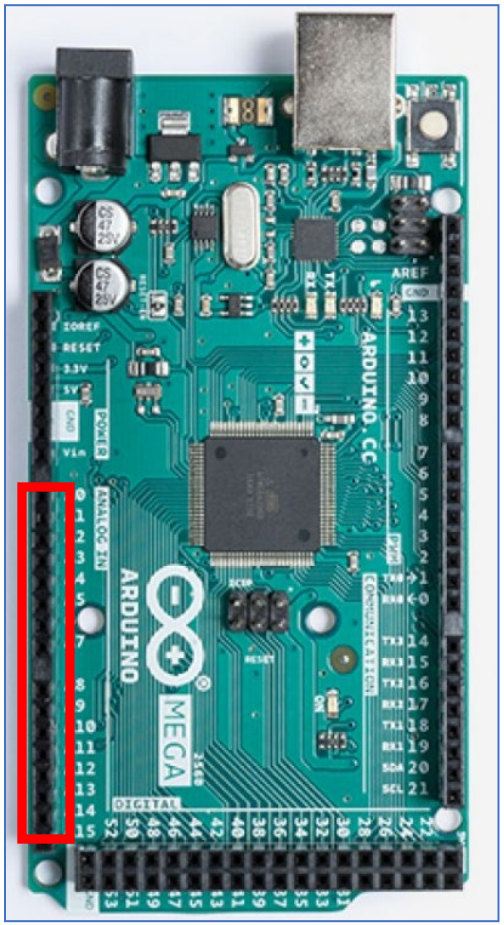
- The standard atmosphere shows temperature changes with altitude at a max of ~3°C/1000 ft
- The average scientific balloon rises at ~1000 ft/min
- This means you expect to experience temperature changes up to 3°C/min
- We want to be able to measure changes of +/- 1 $°C$(matching the resolution)

$$1°C \ * \frac{60\ s}{3\ °C} = 20\ seconds,$$

$$Double\ that\ for\ Nyquist, sample\ every\ 10\ seconds$$

Arduino ADC

- The Mega provides sixteen ADC Pins
- Pins are numbered A0 to A15
- Each ADC Pin is referenced to the GND pin as 0V
- Each is a 10-bit ADC
- 0-5V Range
- Each pin might have slightly different performance
  - 0 ADC at 0.002V on A2, 0 ADC at 0.004V on A3 for example
- Conversion time ~104 uSeconds



Arduino Mega Analog pins are located below the power header.

Arduino ADC Function

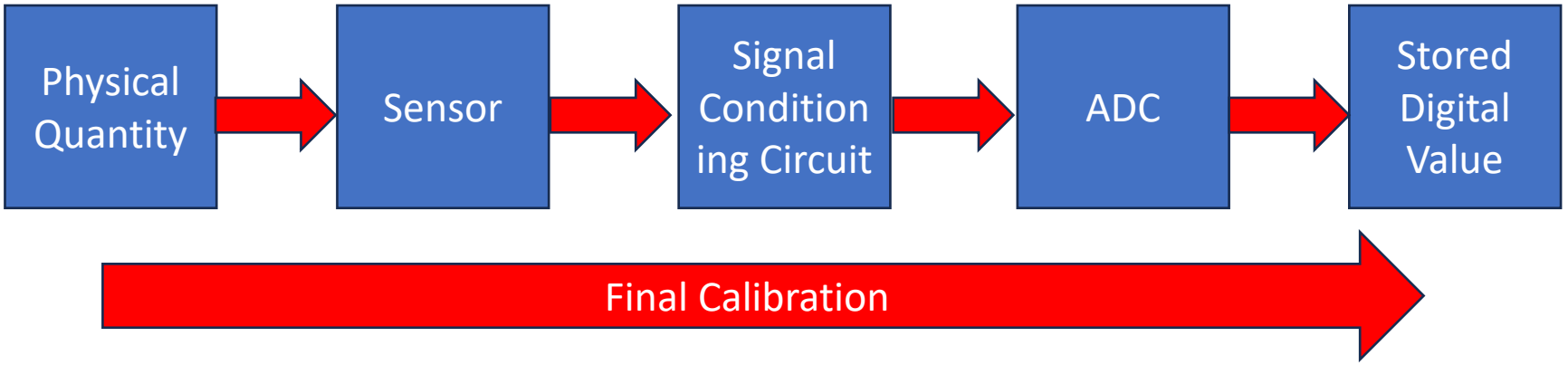
https://docs.arduino.cc/language-reference/en/functions/analog-io/analogRead/

- Single function **analogRead(pin)**

- Constants are defined for each of the analog pins to allow the use of A0 to A15 instead of a pin number
  - analogRead(A3)
- When called, returns the ADC bin as an integer from 0 to 1023
  - **int ADC_reading = analogRead (A2); //example usage**
- Even though ADC is 10-bit, a 16-bit variable is required to store it

Calibration

- When setting up the readout circuitry, you will likely have several intermediate calibrations at each step
- These will allow you to set up offset and gain, or determine if your ADC will meet your requirements



What you actually need is the final conversion between the stored number and the physical measurement, so you should always calibrate the completed system

Digital to Analog Converters (DACs)

- A DAC works the reverse of the ADC; you send the DAC an integer number, and the DAC converts that to an output voltage
- Some Arduinos include a built-in DAC, but the Arduino Mega does not
- Instead, it uses a technique called Pulse Width Modulation to give an analog like signal on the digital pins

How PWM Works

https://docs.arduino.cc/language-reference/en/functions/analog-io/analogWrite/

- The function outputting an analog signal is **analogWrite(pin, value)**
- A pulse repeats on the output pin predefined period
- The integer value determines what percent of the time the pulse on and off (this is called the duty cycle)
  - At 0, it is at 0% so the pulse is always off
  - At 255, is at 100% so the pulse is always one
- So, on average, you get a voltage between 0 and 5V
  - While not a true analog signal for many applications, this average will work the same