# Student Name: Cooper St. Martin

# Student ID: 000858219

The work in this report is my own. I did not share or copy another person’s or thing’s work.

# Analog to Digital (A/D) Conversion

1. You would like to measure speed with your IoT device. You located a sensor (Sensor A) that generates 0-3.3 Volts for a 0-10m/s input. The ESP32 converter is 12 bits wide and accepts a maximum input voltage of 3.3 Volts. What is the speed per bit that you can expect from this A/D converter’s digitized output? Show your work.

(2 Marks)

To get the distance/bit I need to get two things, the speed per volt, and then volt per bit.

Speed per volt = max speed / max voltage

Volt per bit = voltage range of the sensor / controller resolution

To get speed/bit/second i just multiply these together, plug in the numbers we have and we get:

Speed per bit = (10 / 3.3) \* (3.3 / 4096) ~= 0.00244 m/s/b

1. Assume same scenario as the previous question except this time your sensor (Sensor B) produces 0-2 volts for a 0-20m/s input. What is the speed per bit that you can expect from this A/D converter’s digitized output? Show your work. Which sensor is more accurate: A or B?

(3 Marks)

Same logic applies here, apply the formula and plug in the numbers.

Speed per bit = (20 / 3.3) \* (2 / 4096) = 0.00295 m/b/s

So sensor A has a resolution of 0.00244 m/s/b while B has a resolution of 0.00295m/b/s so sensor A is more accurate.

# Sources of Signal Conversion Error

Signals in the physical world rarely behave in an ideal fashion. And the A/D convertors we use to digitize signals trade off accuracy, precision, and cost depending on the application.

Some courses of conversion error were discussed in the lab. Here is an [interesting article](https://www.monolithicpower.com/en/learning/mpscholar/analog-to-digital-converters/adc-errors-and-compensation/sources-of-errors-in-adcs) discussing additional sources of error.

1. List and describe 5 sources of error:

(2 Marks)

1. Offset Errors. Results from a disparity of the real and theoretical outputs.
2. Gain Errors. Results from a divergence from the real and ideal gradient of the ADC.
3. Linearity Errors. In theory the ADC should map the analogue input linearly to the output but due to resolution or real world limitation the steps might not be perfectly linear
4. Quantization errors. Digital representation is only so accurate, perfect matches between this and the analogue input might not be possible
5. Noise. Outside occurrences that interfere with either the sensor or the controller from being accurate. This could be wind on a thermometer, etc.
6. You probably observed that your variable resistor does not behave exactly like other people’s resistors in the lab. If you turn the resistor completely in one direction, you may see a digitized value of 4 and someone else might see a value of 6. And even when you are not turning the resistor, the digitized value is not completely steady. These errors are caused by manufacturing variances in the part and limitations of the A/D converter in the ESP32.

Imagine you are developing an IoT device, and you must deal with components that show variability like this. How could you compensate for this issue in software?

(2 Marks)

Given the range of the resistor not being full calibrating the software so it compensates for the real max/min of the sensor not the theoretical max/min is one possible solution

With the unstable read outs from the resistor you could record an average of the outputs given a certain time frame then return that instead. This would avoid a lot of the jittery behaviour that’s attributed to the sensor’s build quality rather than user interaction.