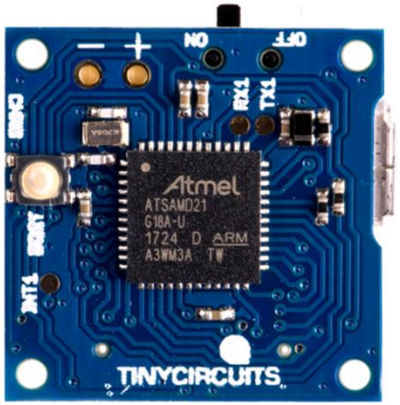
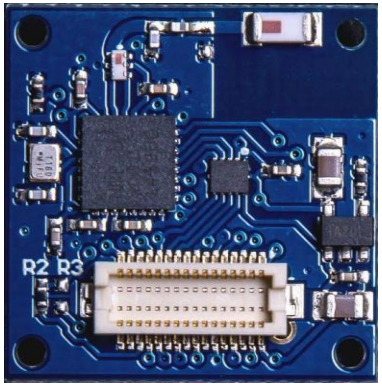
CSE190 Winter 2025

Lecture 16

Interfacing with The Analog World





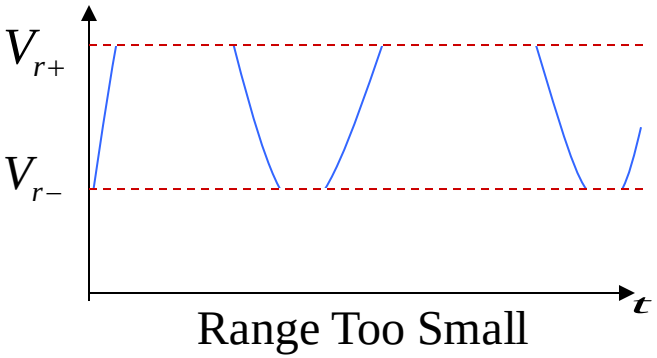
Wireless Embedded Systems

Aaron Schulman

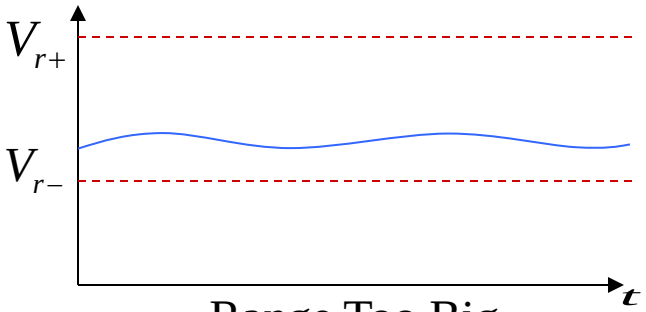
2nd digital problem: Choosing the ADC's range (amplitude)

- Fixed (discrete) # of bits (e.g. 8-bit ADC)
- Span a particular continuous input voltage range
- What do the sample values represent?
  - Some fraction within the range of values

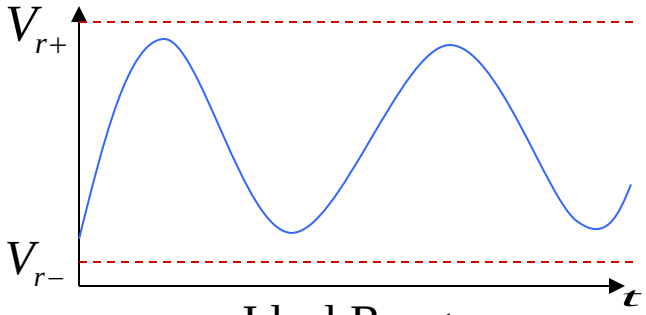
  → What range to use?



Range Too Small



Range Too Big



Ideal Range

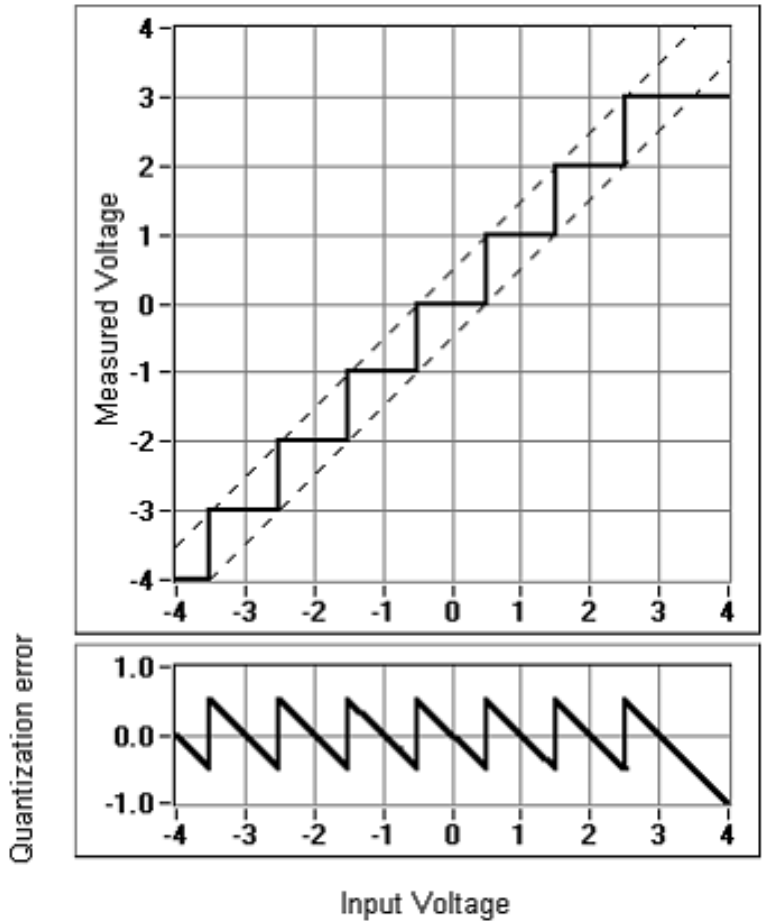
Choosing the granularity

- Resolution
  - Number of discrete values that represent a range of analog values
  - 12-bit ADC
    - 4096 values
    - Range / 4096 = Step

  Larger range ➔ less info / bit

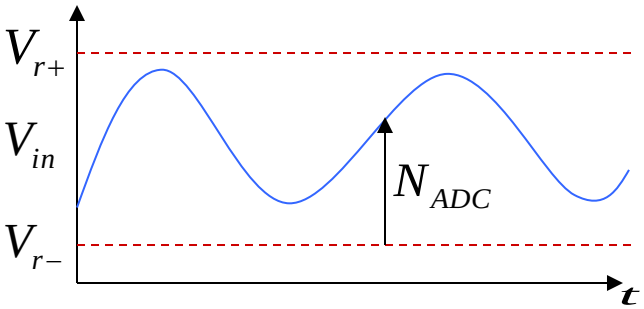
- Quantization Error
  - How far off discrete value is from actual
  - ½ LSB → Range / 8192

  Larger range ➔ larger error



Converting between voltages, ADC counts, and engineering units

- Converting: ADC counts ⇔ Voltage



$$N_{ADC} = 4095 \times \frac{V_{in} - V_{r-}}{V_{r+} - V_{r-}}$$

$$V_{in} = N_{ADC} \times \frac{V_{r+} - V_{r-}}{4095}$$

- Converting: Voltage ⇔ Engineering Units

$$V_{\text{TEMP}} = 0.00355(\text{TEMP}_{\text{C}}) + 0.986$$

$$\text{TEMP}_{\text{C}} = \frac{V_{\text{TEMP}} - 0.986}{0.00355}$$

A note about sampling and arithmetic*

- Converting values in fixed-point MCUs

$$V_{\text{TEMP}} = N_{ADC} \cdot \frac{V_{r+} - V_{r-}}{4095} \qquad \text{TEMP}_{\text{C}} = \frac{V_{\text{TEMP}} - 0.986}{0.00355}$$

```
float vtemp = adccount/4095 * 1.5;
float tempc = (vtemp-0.986)/0.00355;
```

→ vtemp = 0! Not what you intended, even when vtemp is a float!

→ tempc = -277 C

- Fixed point operations
  - Need to worry about underflow and overflow

- Floating point operations
  - They can be costly on the embedded system

Try it out for yourself…

You need to scale integers so there are no < 1 fractions

```
$ cat arithmetic.c
#include <stdio.h>

int main() {

  int adccount = 2048;
  float vtemp;
  float tempc;

  vtemp = adccount/4095 * 1.5;
  tempc = (vtemp-0.986)/0.00355;

  printf("vtemp: %f\n", vtemp);
  printf("tempc: %f\n", tempc);
}

$ gcc arithmetic.c

$ ./a.out
vtemp: 0.000000
tempc: -277.746490
```

Oversampling (sampling faster than Nyquist)

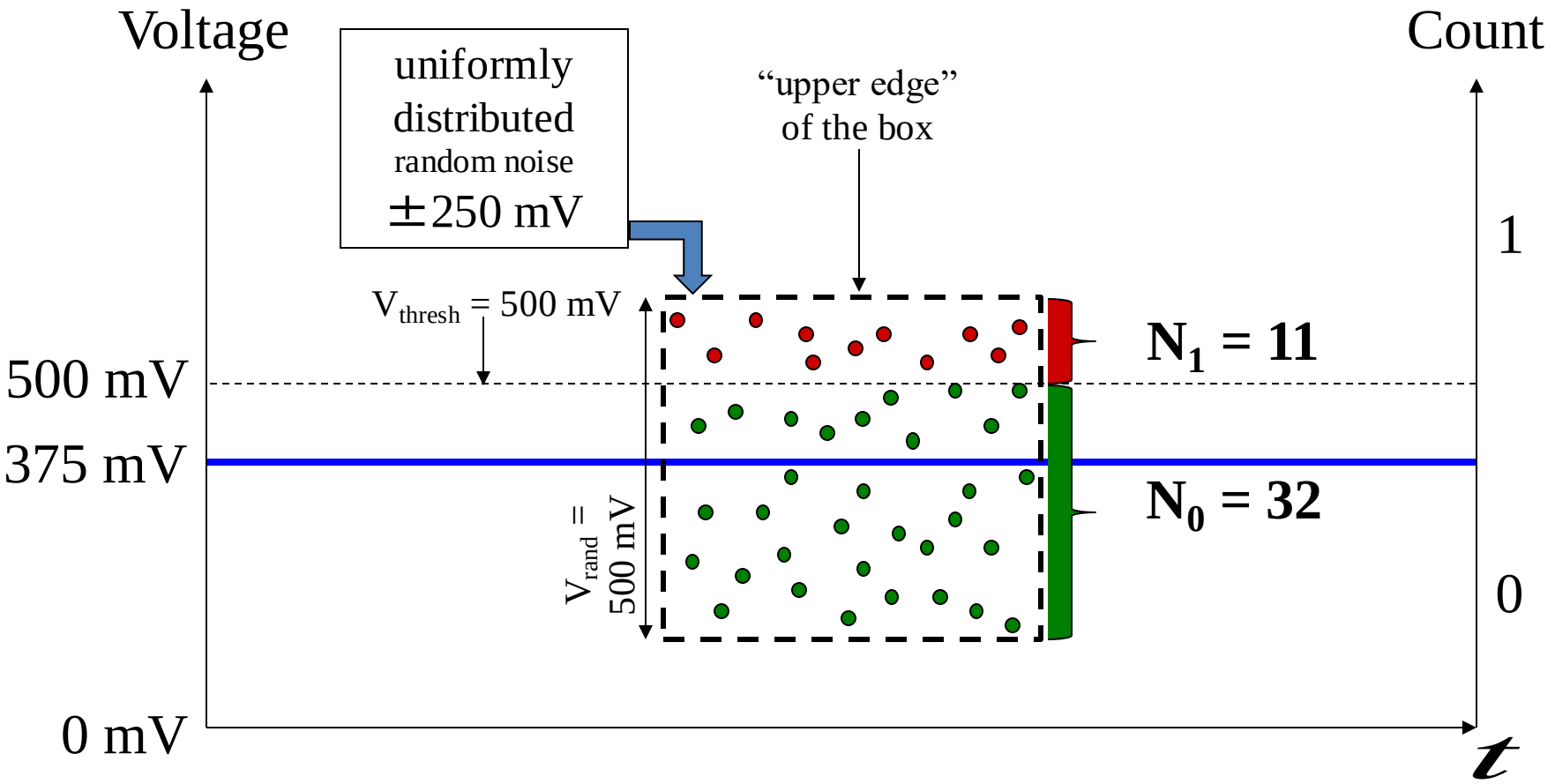
One interesting trick in analog+digital is that you can use oversampling to help reduce the impact of quantization error.

- Let's look at an example of oversampling plus dithering to get a 1-bit converter to do a much better job…

Oversampling a 1-bit ADC w/ noise & dithering (cont)

- How to get more than 1-bit out of a 1-bit ADC?
- Add some noise to the input
- Do some math with the output
- Example
  - 1-bit ADC with 500 mV threshold
  - Vin = 375 mV → ADC count = 0
  - Add ±250 mV uniformly distributed random noise to Vin
  - Now, roughly
    - 25% of samples (N_1) ≥ 500 mV → ADC count = 1
    - 75% of samples (N_0) < 500 mV → ADC count = 0

Oversampling a 1-bit ADC w/ noise & dithering (cont)



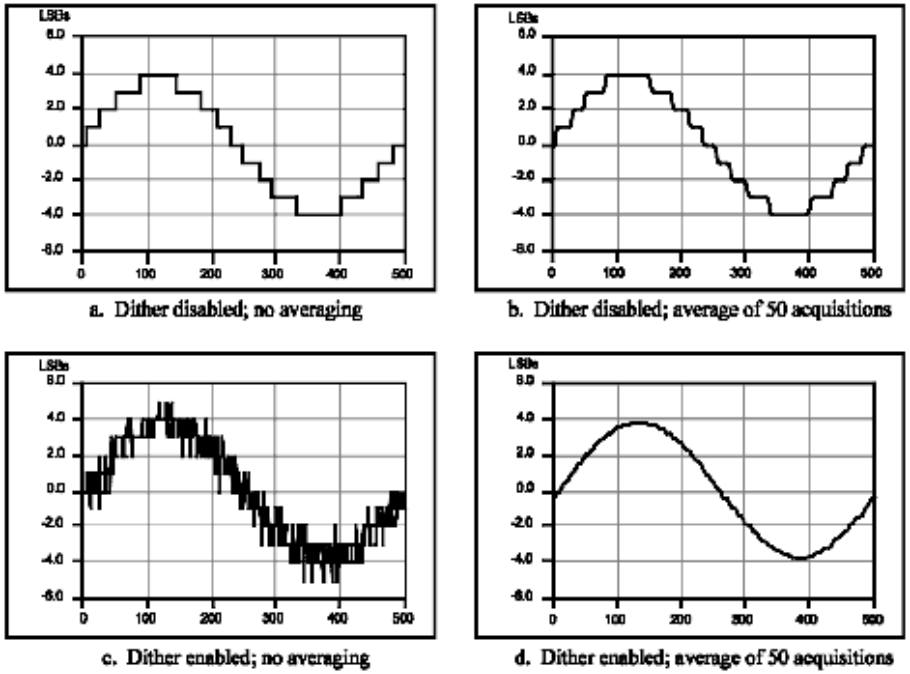
Note:

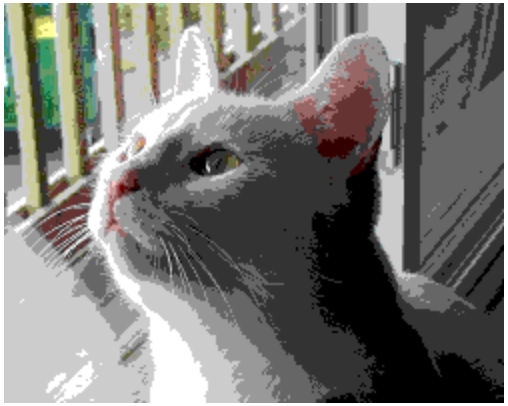
N_1 is the # of ADC counts that = 1 over the sampling window

N_0 is the # of ADC counts that = 0 over the sampling window

Can use dithering to deal with quantization error in images too

- Dithering (introducing noise)
  - Quantization errors can result in large-scale patterns that don't accurately describe the analog signal
  - Oversample and dither
  - Introduce random (white) noise to randomize the quantization error.





Direct Samples

Dithered Samples

Selection of a DAC (digital to analog converter)

- **Error/Accuracy/Resolution:** Quantizing error represents the difference between an actual analog value and its digital representation. Ideally, the quantizing error should not be greater than ± 1⁄2 LSB.

Output Voltage Range -> Input Voltage Range

- **Output Settling Time** -> Conversion Time
- **Output Coding** (usually binary)