Timing and Analog Input

CSE 132

Simple Timing

- Use Thread.sleep() in Java
  - Argument is integer number of milliseconds before the method returns

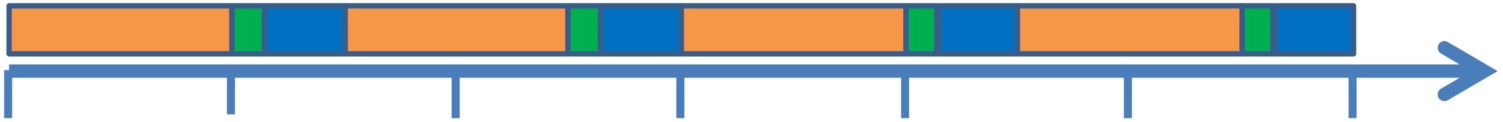
```
for (int i=0; i < endTime; i++) {
    Thread.sleep(1000);
    System.output.println(i + “ seconds have elapsed”);
}
```

- Use delay() on Arduino
  - Same approach as in Java

Effects of Simple Timing

- What are possible issues with this code?

```
while (true)
    wait for 1 second    🟧
    do some work         🟩
    output results       🟦
end while
```



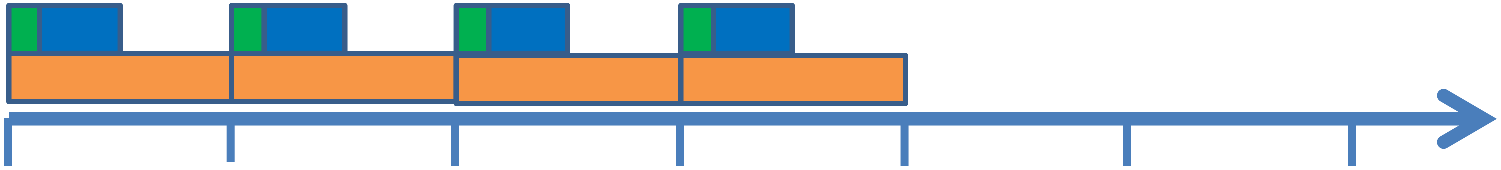
Better Timing

- Use a free-running timer
  - unsigned long millis()
  - Returns # of milliseconds since reset
  - Rolls over to zero after about 50 days
- Now we can use delta time techniques

```
while (true)
    if (millis() > loopEndTime) then
            loopEndTime += deltaTime
            do some work
    end if
end while
```

Impact of Delta Timing

```
while (true)
    if (millis() > loopEndTime) then
        loopEndTime += deltaTime    🟧
        do some work                🟩
        output results              🟦
    end if
end while
```



What if work has delays?

```
while (true)
    if (millis() > loopEndTime) then
            loopEndTime += deltaTime
            do some work
    end if
end while
```

Especially if work takes longer than deltaTime!

Think Like a Finite-State Machine

```
  while (true)
➡     if (millis() > loopEndTime) then
          loopEndTime += deltaTime
          do some work
      end if
  end while
```

Do some (but not all) of the work

Remember “state” information (in one or more variables)

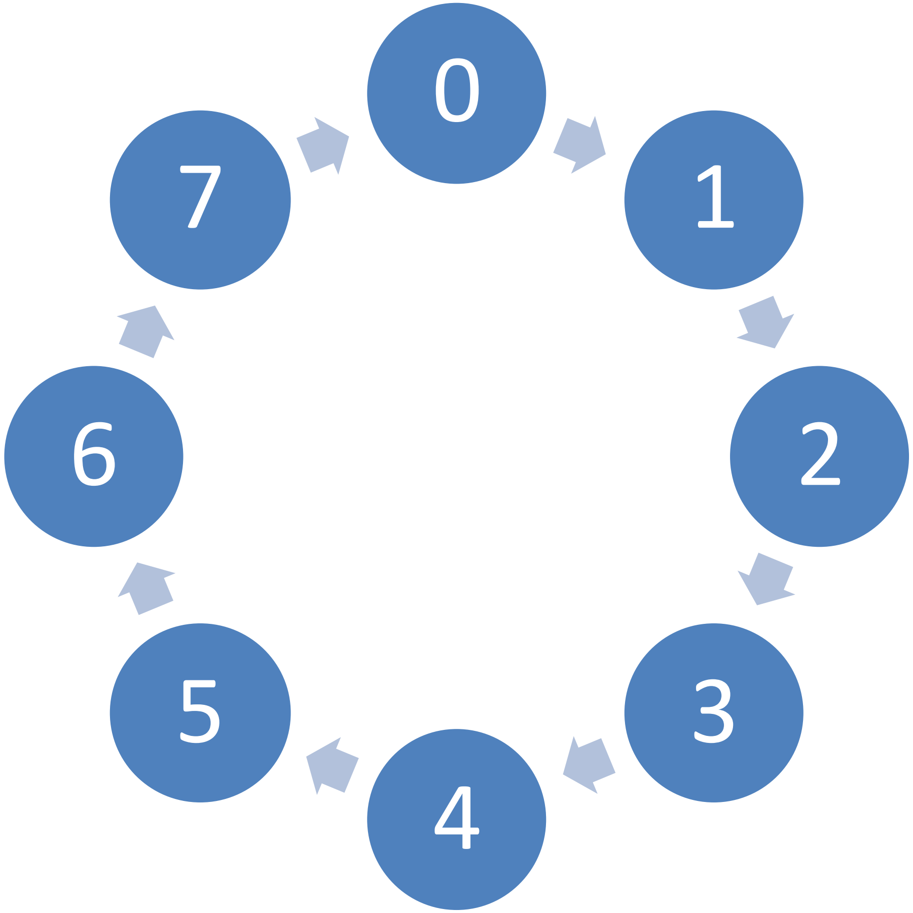
Inside delta time conditional if, add switch statement

Finite State Machine (FSM)

- Useful concept for today's studio software
- Used extensively in hardware and software systems design and analysis
- Explicitly enumerate (i.e., list) all of the "states" that our design can have, and articulate:
  - What happens (e.g., is output) in each state
  - What state is next under what conditions
- "States" represent what our design wishes to remember

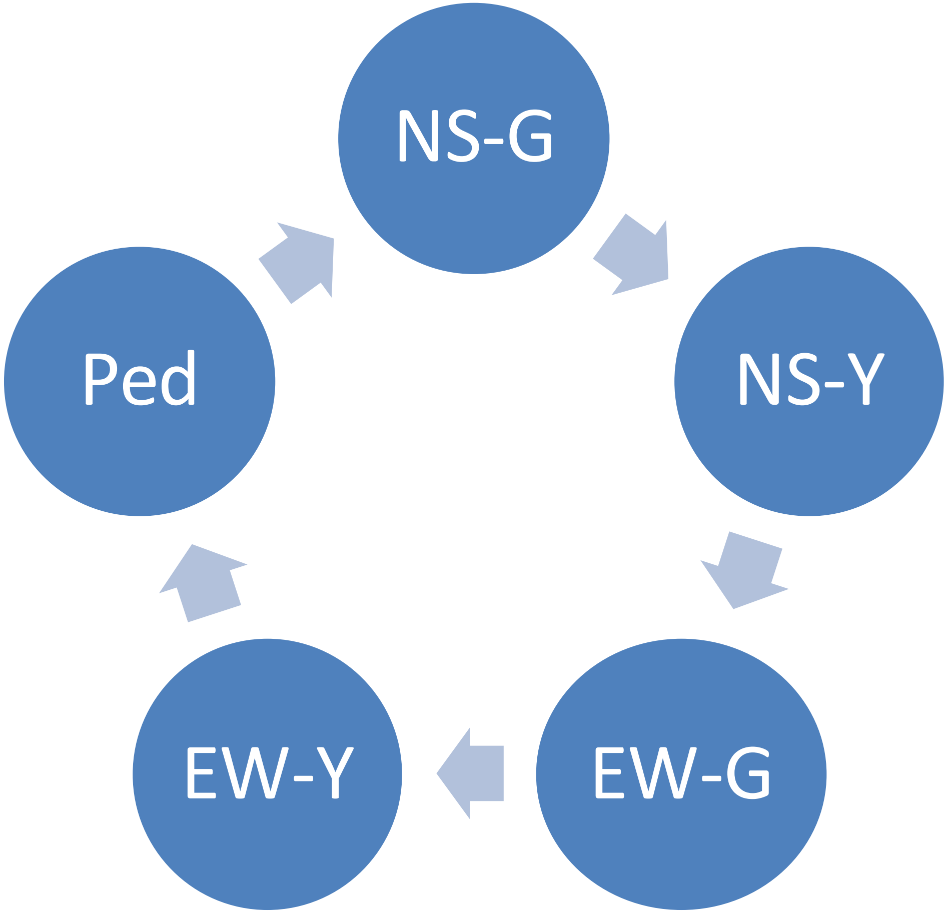
FSM Diagram

- A 3-bit counter cycles from 0 to 7, and then roles over back to 0
- Consider each count value to be a “state”
- In each state, output is simply value of count
- In each state, next state is value+1



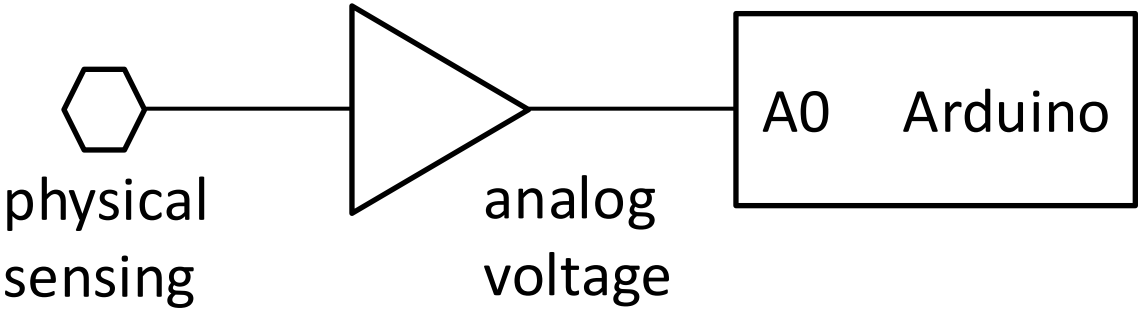
Stoplight Controller

- NS-G: North/South Green
- NS-Y: North/South Yellow
- EW-G: East/West Green
- EW-Y: East/West Yellow
- Ped: Pedestrian Walk

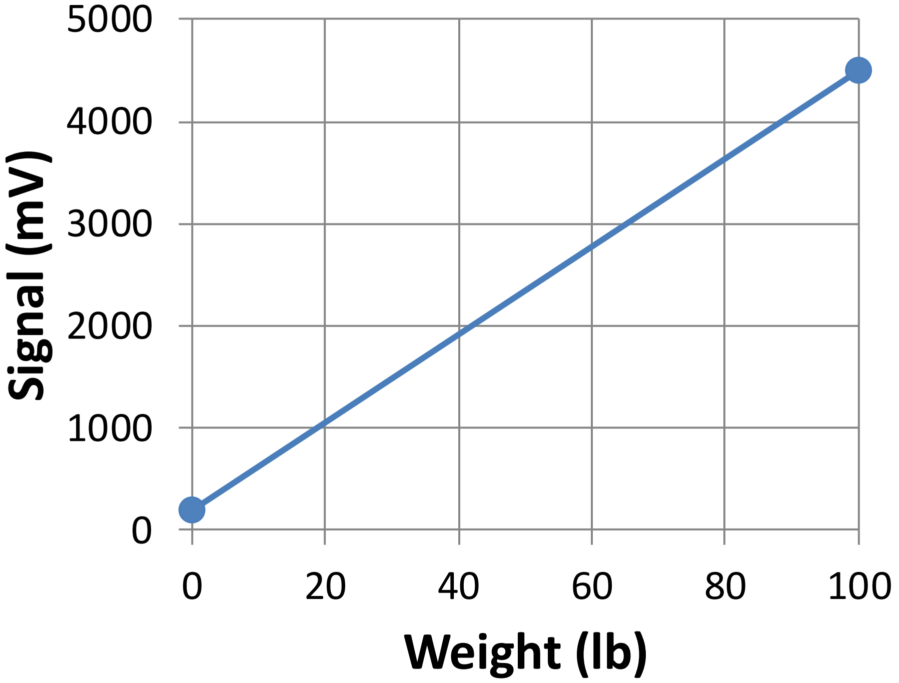


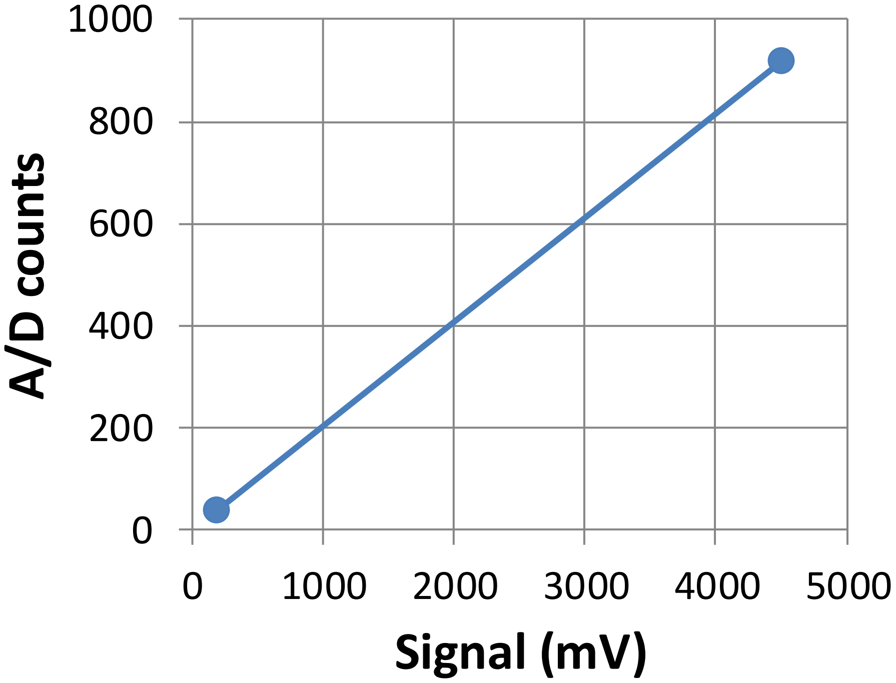
Analog to Digital Conversion

- Convert physical property to voltage signal
- A/D converter on Arduino converts voltage signal to digital representation
  - 10-bit A/D converter has range 0 to $2^{10} - 1$ (0 to 1023) for voltage range 0 to V_{REF}



Understanding Ranges





$$signal = m \times weight + b$$

$$counts = m \times signal + b$$

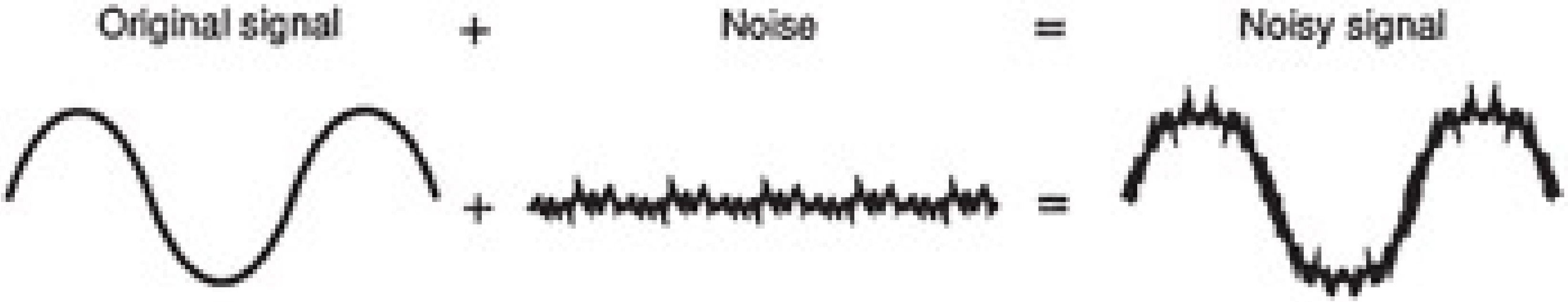
$$signal = 43\frac{mV}{lb} \times weight + 200mV$$

$$counts = 0.2\frac{cnt}{mV} \times signal + 0$$

$$counts = 8.6\frac{cnt}{lb} \times weight + 40$$

$$weight = 0.116\frac{lb}{cnt} \times counts - 4.65$$

Noisy Analog Signals



- Noise is ever present in analog signals
- For stable signal, quick fix is to average several readings

$$avg = \frac{1}{N}\sum_{1}^{N} A/D\ input_i$$

Quiz Time

- Go to Canvas and answer the single question for Quiz 3A

- What data type does millis() return?
  - unsigned long
  - unsigned int
  - char
  - float