Week5: Sensors

Dan Goldsmith

November 7, 2017

Overview

Talking about Sensors

- A bit of electronics
- How do we get the micro-controller to talk to our sensors?
- Different types of sensors

Sensors (Physical / Background) Stuff

What is a Sensor?

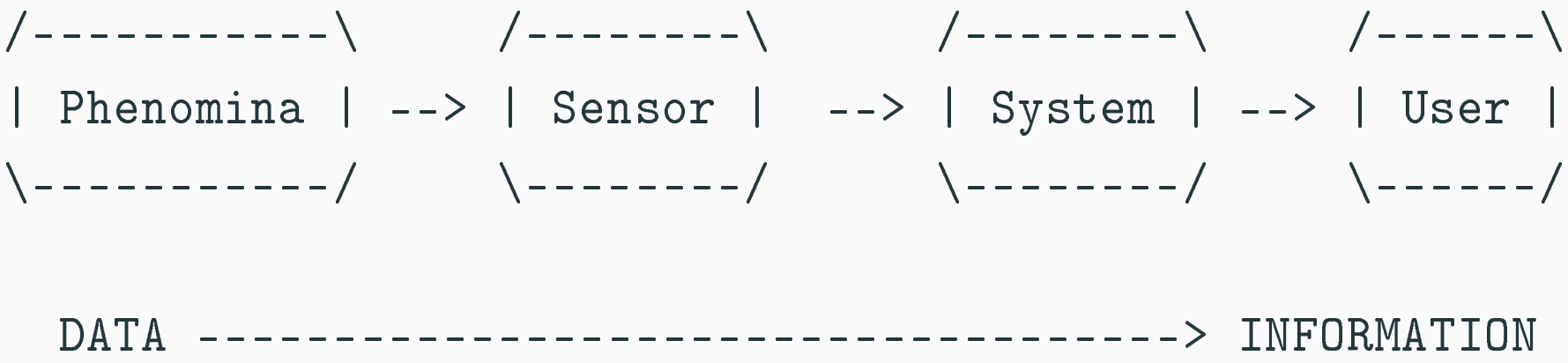
A sensor is a converter that measures a physical quantity and converts it into a signal that can be read by an observer or instrument.

What is a Sensor?

While the wikipedia statement is accurate. . .

- We are learning about pervasive computing
  - Using electronics to measure and process phenomena
  - Therefore it is important that our sensors convert to a signal that can be understood by a computer.

Data flows



Analog or Digital?

- We live in an Analog world:
  - Infinite Spectrum of Colors we can see
  - Infinite Number of Sounds / Tones we can here
- Digital signals are Discrete or Finite meaning there are a limited set of values (2, 255, 1024 etc.)

Analog or Digital?

23:59:59

Signals

- In computing signals are defined as time-varying elements that convey information
  - Generally this is voltage (However some use Current)
- Used to send and receive information between devices
- Can be transmitted over Wire / Radio etc.

Differences

Analog

- Continuous
- Infinite range of values (higher accuracy?)
- Can be harder to work with electronically

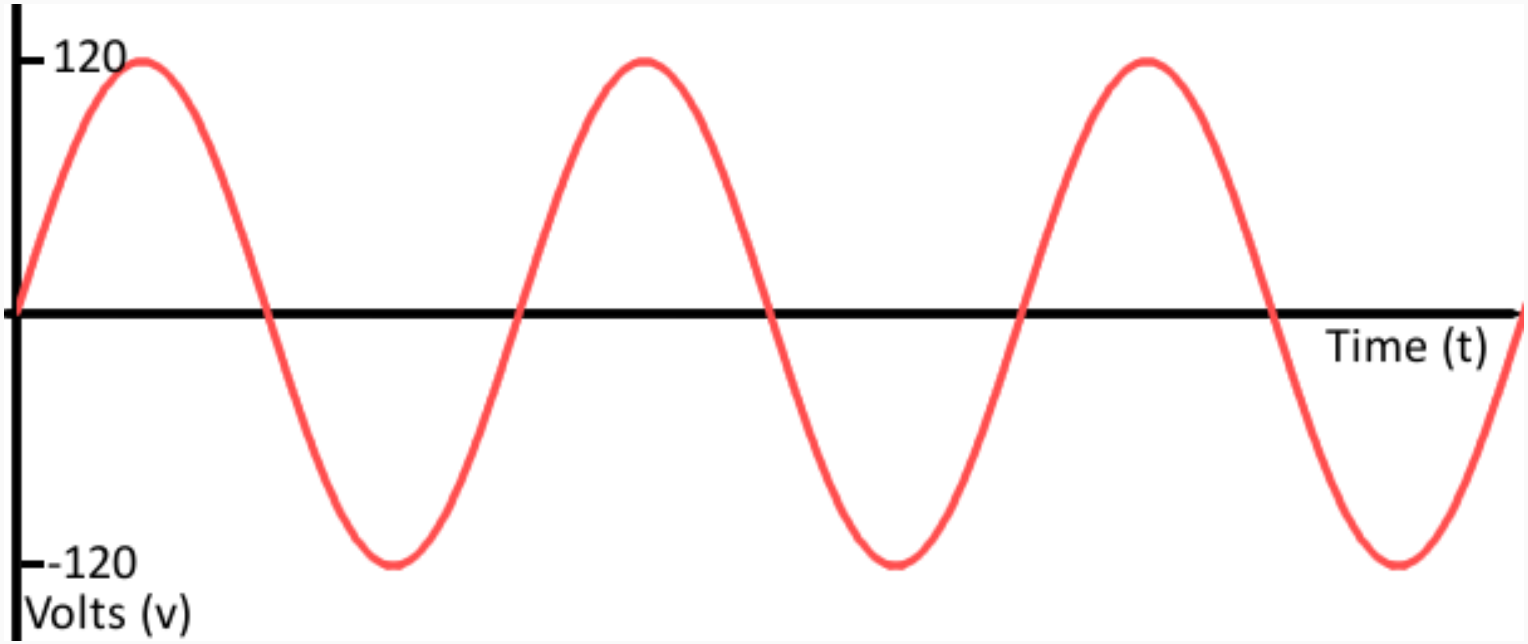
Digital

- Discrete
- Finite range of values
- Not as exact but easier to work with

Analog Sensors and Signals

Analog Signals

- Time-VS-Signal graph should be smooth, with an "infinite" number of states
- Can also have negative values



Analog Sensors

- Return an output (usually a voltage) that varies depending on the state it is measuring
- Think of the Temperature sensor in the simulator:
  - Low temperature == Low Voltage
  - High temperature == High Voltage

Analog to Digital conversion

- But aren't computers Binary??
- We need to convert these values to Digital before the micro controller can understand them.
  - Process know as Analog To Digital Conversion (ADC)
  - Uses a separate IC
  - A2D converters accuracy is given a number of Bits (the resolution)

Analog to Digital Conversion:

- A number of 'slots' given by the Resolution of the ADC
- Voltage is mapped between the High and Low Reference voltages
    - Generally between 0 and the voltage the circuit is running at

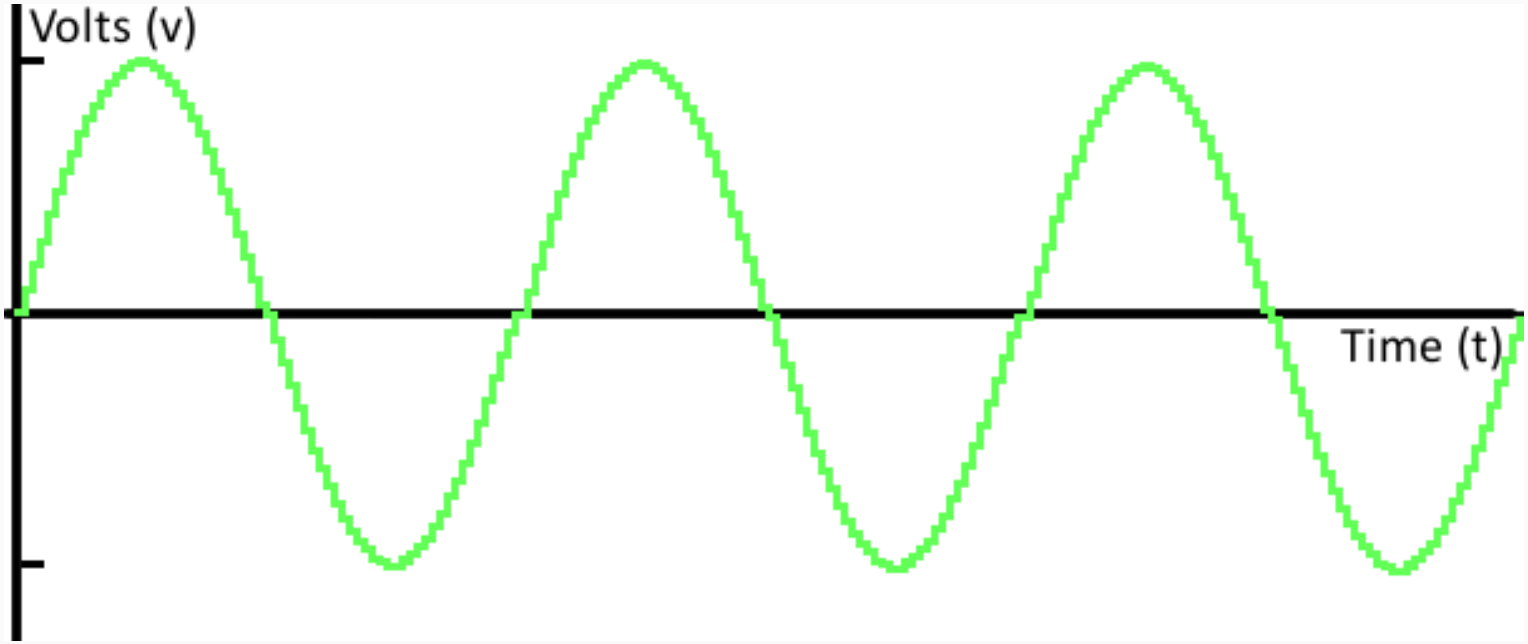
Analog to Digital conversion:

Consider the 8 bit ADC on the Arduino:

- Has $2^8 == 256$ 'slots'
- Voltage range of 0-5v
    - Thus each 'slot' is 0.019v

What happens to our Signal

- The signal is now quantized
- Steps between values



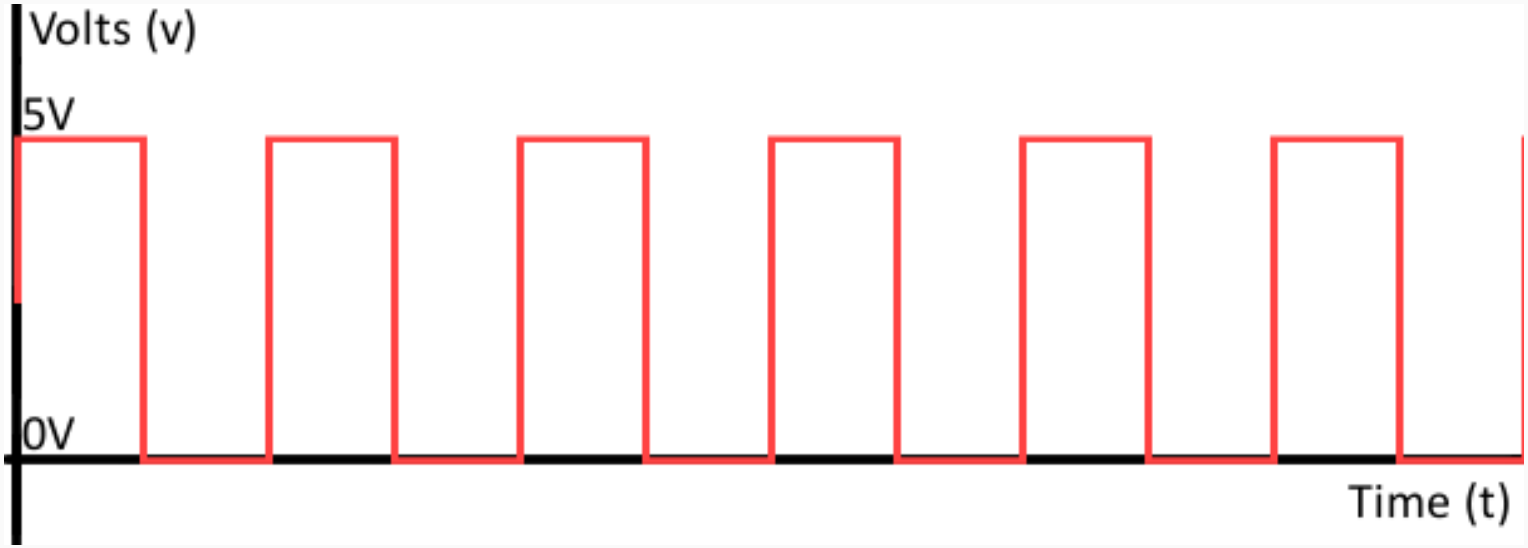
Factors to Consider with Analog sensors

- Generally easy to get the voltage (read from an ADC pin)
- Will need converting from voltage to value
- Depending on the output voltages of the sensor(s) we can lose accuracy

Digital Signals

Digital Signals

- "Binary" ON-OFF signal
- No negative values



Digital Sensors

- Talk in the same binary "language" as our Microcontroller
- If its just On and Off are we stuck with switches?
  - Sensor has processing capabilities
  - We ask the sensor to provide the value

Digital Sensors

- Moving the ADC onto the Sensor has some benefits
  - Resolution can be defined by the sensor (rather than other hardware)
  - Removes the need for a separate ADC in micro-controller
  - Any conversion to "real world" units can be done on the device.

Communicating with Digital sensors

- Switches are easy, just On and Off.
- Well understood Standards for Digital Communication.
  - Serial
  - SPI
  - I^2C
- Most of these are "wrapped" in Arduino Function Calls

"Serial" Communications

- Data is transmitted as a steam of bits (01001010...)
- Sender dictates the speed data is transmitted (asynchronous)
- Data is encoded using well known scheme
- Examples:
    - Sending data between Arduino and PC (Ascii Encoded)

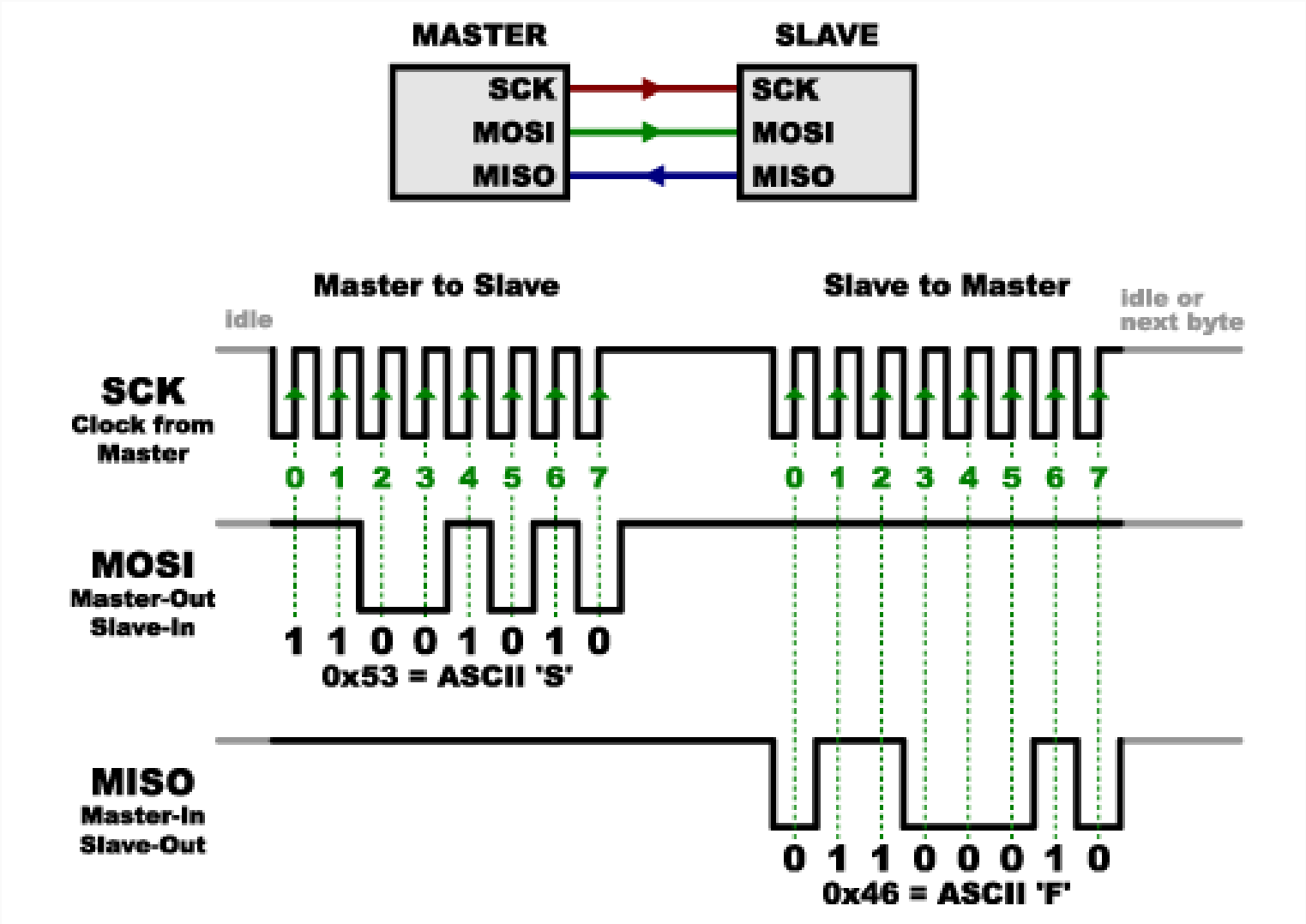
Considerations of Serial

- Benefits
  - A simple method of sending between two devices
  - Only 2 wires required
- Drawbacks
  - Crossed Wires (Tx -> Rx not Tx -> Tx)
  - Baud Rate mismatch
  - Bus Contention (running more than one device)

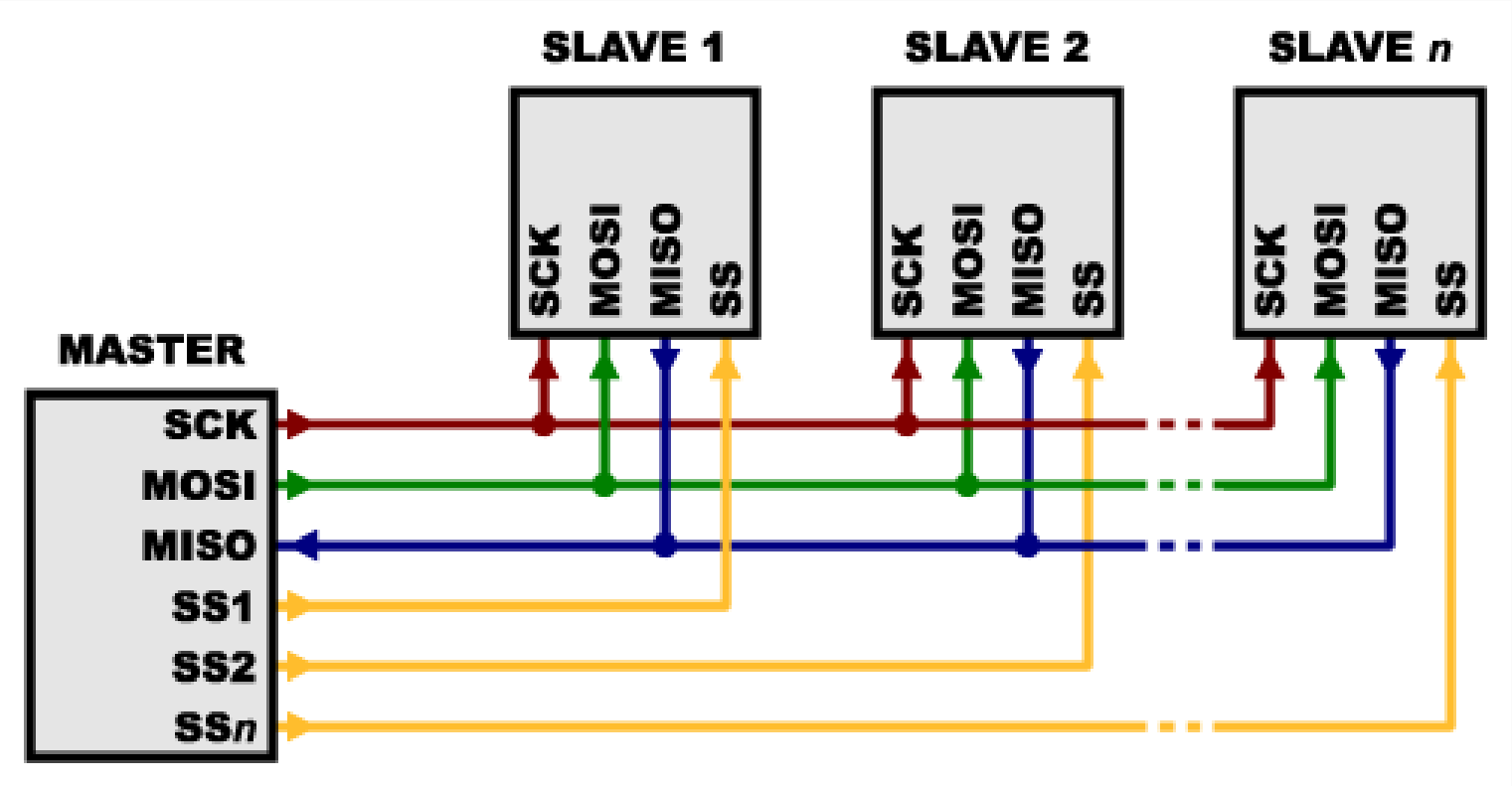
SPI

- Uses separate lines for data and a clock
  - Ensures both sides are in sync
- Allows multiple devices using chip select lines

SPI


MASTER
SLAVE
SCK
MOSI
MISO
SCK
MOSI
MISO
Master to Slave
Slave to Master
idle
idle or
next byte
SCK
Clock from
Master
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
MOSI
Master-Out
Slave-In
1 1 0 0 1 0 1 0
0x53 = ASCII 'S'
MISO
Master-In
Slave-Out
0 1 1 0 0 0 1 0
0x46 = ASCII 'F'

Multiple Slaves



Considerations of SPI

- Pros
  - Faster than Serial
  - Simple to implement in hardware
  - Supports multiple devices
- Cons
  - Requires many signal lines
  - Master must control communication
  - Communication need to be well defined in advance

I^2C / TWI

- Inter-Integrated Circuit Protocol / Two wire Interface.
  - Designed to allow multiple "slave" sensors
  - Requires less wires that SPI
  - Uses slave addresses to control communication

Considerations of I^2C

- Pros
  - Two Wires
  - Multiple Master / Slave
  - Complex Communication possible
- Cons
  - More Complex IC than SPI
  - Proprietary (though Phillips have "opened" it)?
  - Needs consistent logic levels (ie all 5v)

Dont Panic

- You shouldn't have to write a driver:
  - Most of the Sensors we have are Analog
  - Simple Libraries / Drivers for the Digital Sensors

Examples of Sensors