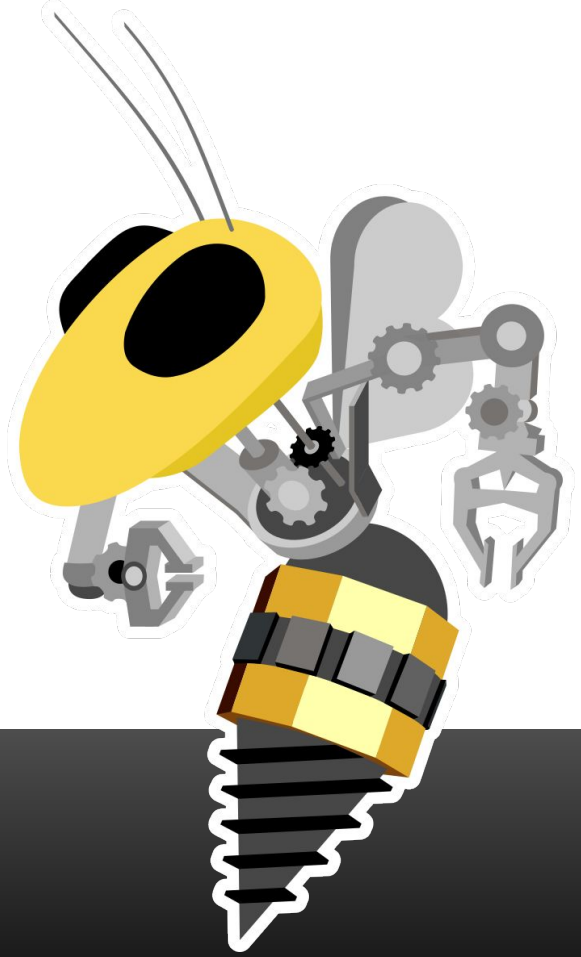
Welcome!

Firmware Training
Week 4



RoboJackets
Competitive Robotics at Georgia Tech

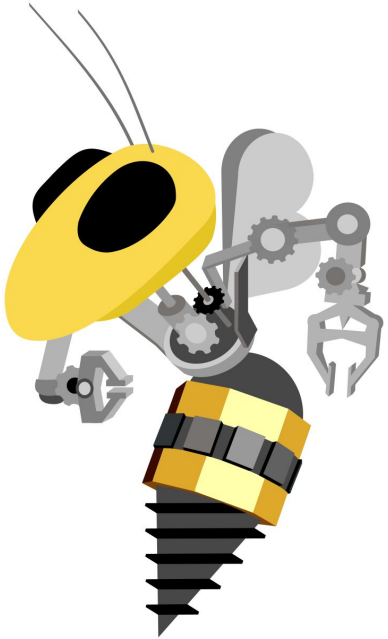
www.robojackets.org

Last Week

- Registers and Timers
- PWM
- Binary and Hexadecimal
- Bitwise Operations
- Masking

This Week

- More C++
- Common Communication Protocols
  - SPI, I2C, UART, USB
- Sensors
- I2C implementation in ATMega328p
- Lab



C vs C++

C vs C++

- For a long time, C was used over C++ on MCUs
  - Largely had to do with lack of C++ compilers for MCUs
- In recent years, C++ has gained popularity for MCUs
- C:
  - Procedural, small language, “easier to read” assembly
- C++:
  - Object oriented, huge feature set, can do everything C can, can make code easier to understand
- Interesting Podcast: C vs C++ for Embedded Systems

C++ Software Training Videos

- Week 0 Playlist
  - What is C++, variables, objects, loops
- Week 1 Playlist
  - Namespaces, const, strings, header files
- Week 2 Playlist
  - Classes, inheritance,
- Week 3 Playlist
  - References, Pointers, Ownership
- Week 4 Playlist
  - Concurrency & Threads (advanced)

C++ Software Training Videos (cont)

- Week 5 Playlist
  - Lambda expressions, Function Pointers (advanced)
- Week 6 Playlist
  - Templates
- Week 8 Playlist
  - Iterators, Basic Algorithms

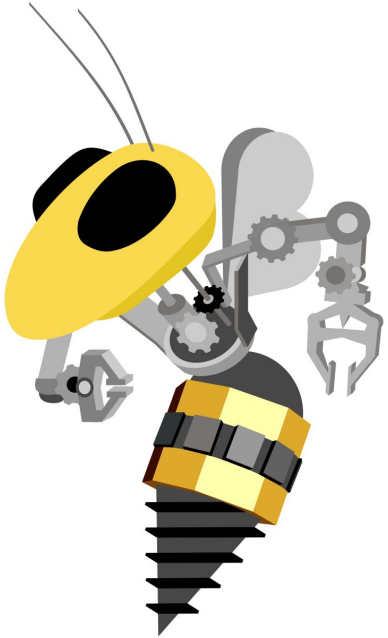
Core C++ Ideas

C++ Basics:
- Variables
- Objects
- Loops
- Keywords (static, const, final, volatile,…)
- Strings
- Iterators
- C++ structure (source, header files)

OOP (Object Oriented Programming)
- Classes & Objects
- Inheritance
- Polymorphism
- Namespaces
- Templates

Advanced Topics
- Concurrency & Threads
- Lambda Expressions
- Function pointers

Memory
- Pointers
- References
- Ownership

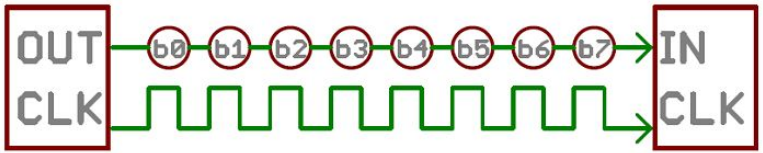


Communication Protocols

Now we're talking

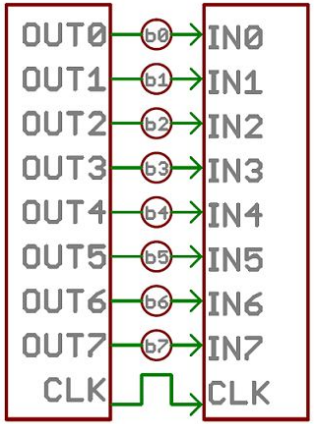
Serial vs Parallel Communication

Serial



- Stream data one bit at a time
- Example: I2C, USB, SPI, UART

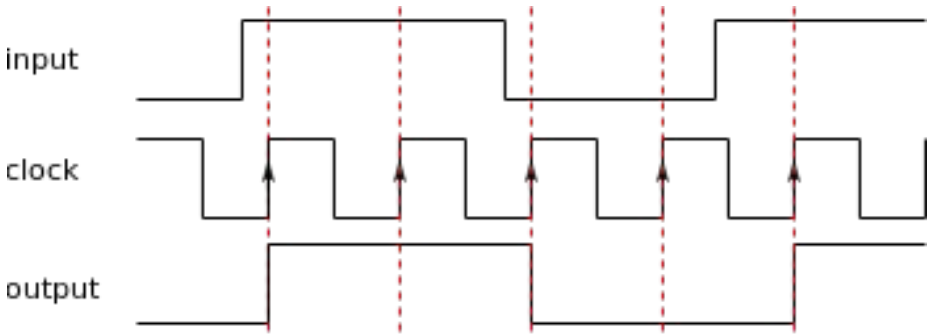
Parallel



- Many bits of data sent at the same time through different wires
- Example: PCI

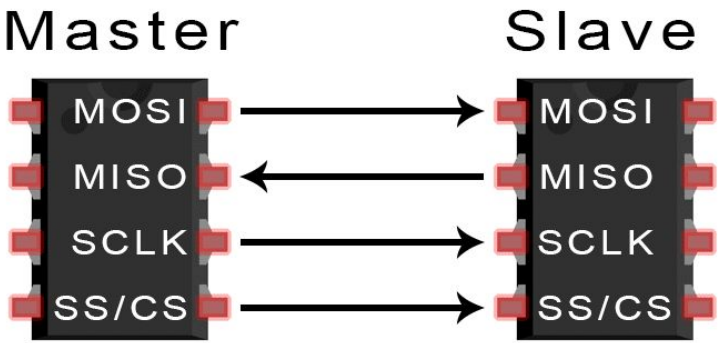
Clock Signals

- Square waves of known frequency (baud rate)
- Edge used to sync data reading across devices
- Synchronous
  - Uses clock
- Asynchronous
  - Does not use clock



SPI (Serial Peripheral Interface)

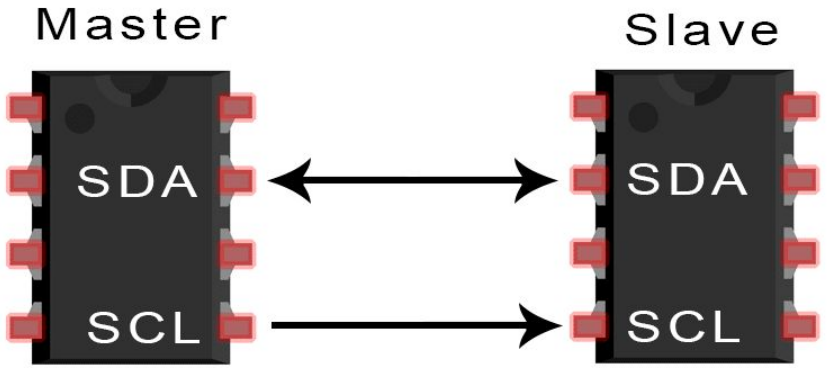
- Full Duplex (data goes both ways at any time)
  - All devices share 3 lines
- Unique Chip Select line per device
- Master controls CLK



MOSI: Master Out Slave In
MISO: Master In Slave Out
SCLK/SCK: Serial Clock
SS/CS: Slave Select/Chip Select

I2C (Inter-Integrated Circuit) / TWI (Two Wire Interface)

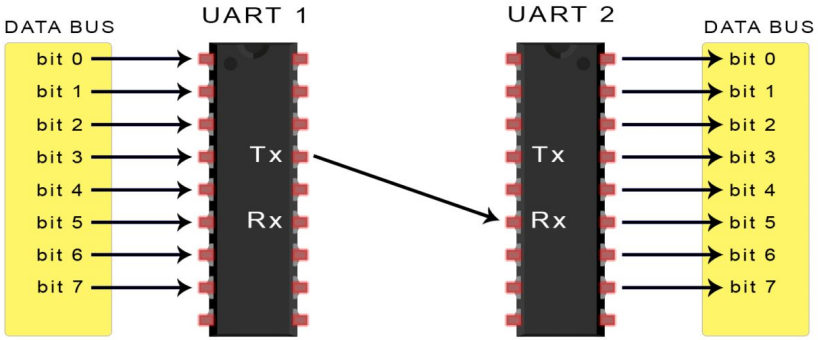
- Synchronous, Half Duplex (one way at a time)
- Uses only two wires
- Sends data in ‘frames’
- Any device can claim master by controlling SCL



SDA: Serial Data
SCL: Serial Clock

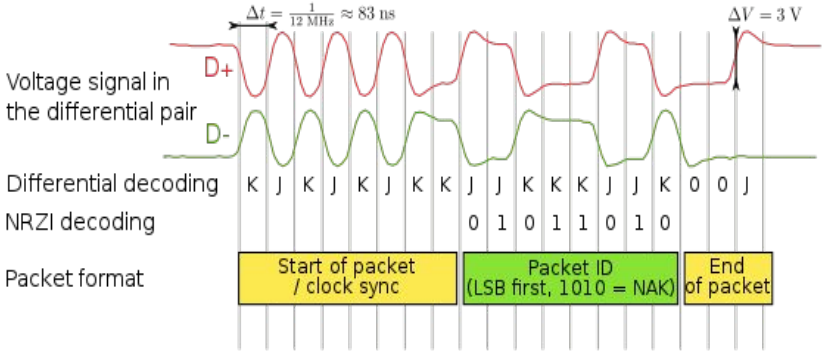
UART (Universal Asynchronous Receiver/Transmitter)

- Asynchronous
- Full Duplex
- Uses 2 wires
- Need same baud rate



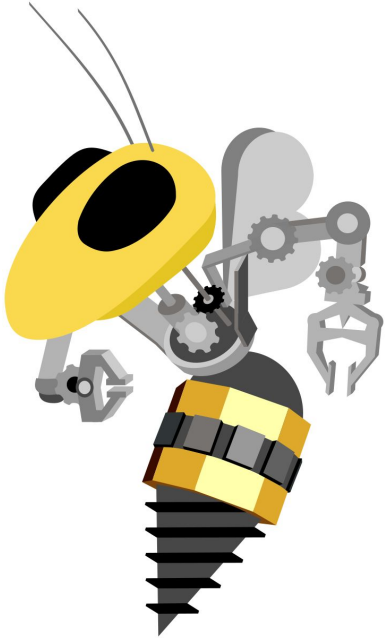
USB (Universal Serial Bus)

- Differential Pair signal
  - Difference between D+,D-
- Defines rate in “clock sync” phase
- Useful for computer-device communication



Protocol Comparison

Protocol	Needed Wires (min)	Clock?	Speed	Number of Devices (max)
SPI	4	Yes	Fast	A lot (need additional wire per)
I2C	2	Yes	Slow	127
UART	2	No, set same rate	Slow	2 (without more hardware)
USB	2	No, data sync	Fast	2 (without more hardware)



Sensors

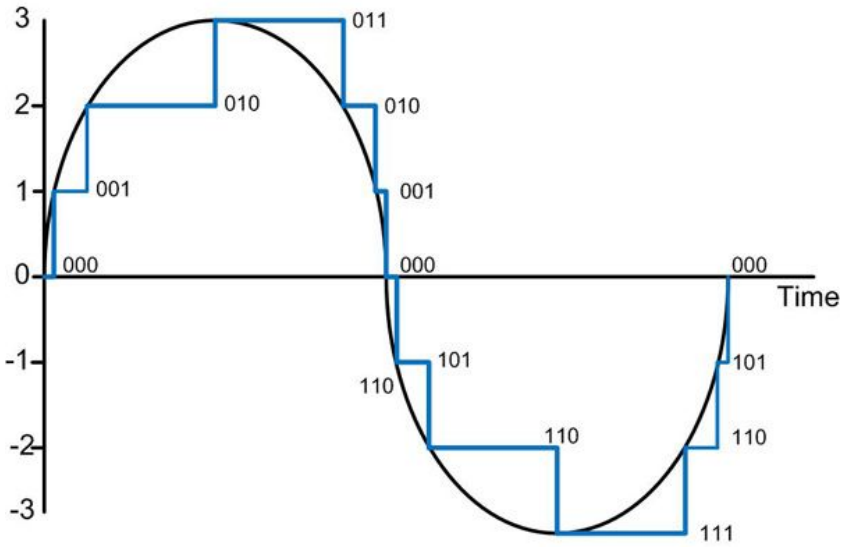
How many are there again… ?

What is a Sensor?

- Converts a real-world quantity into an electrical signal
  - Used to measure “state variables” of robot or system
  - Can be Digital or Analog Signal

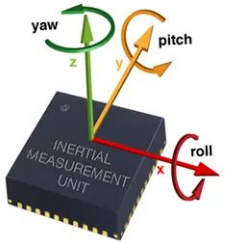
Types of Signals

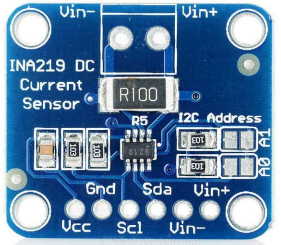
- Analog Signal
  - Outputs voltage correlated to reading
- Digital Signal
  - Communicates binary representation of reading



Types of Sensors

- IMU
  - Accelerometer
  - Gyroscope
  - Compass
- ToF/Distance
- Encoder/Potentiometer
- Current Sensor





Types of Sensors

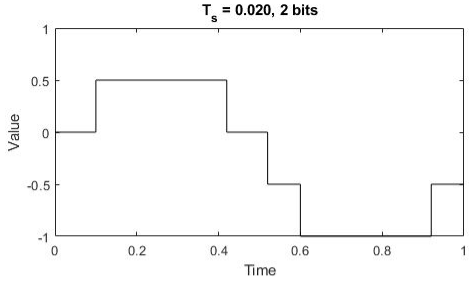
- Camera
- LIDAR
- GPS

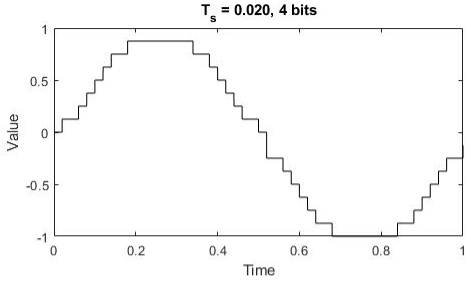
Sample Rate

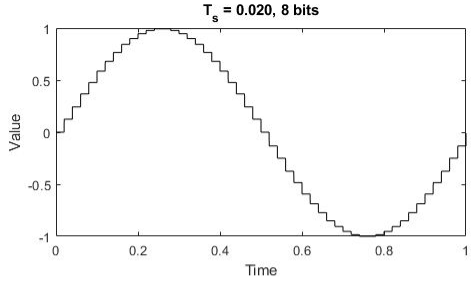
- Sample Rate f_s
  - How many times a sensor can read per second
- f_s must be at least twice the max frequency of the sampled signal
  - Failure to do so may result in aliasing (which distorts the signal)
  - Thanks Mr. Nyquist!

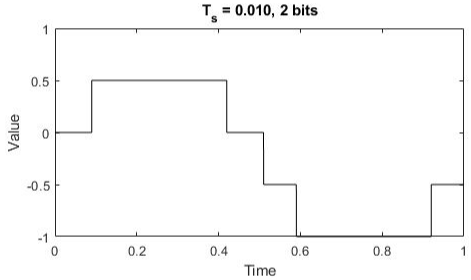
Bit Depth

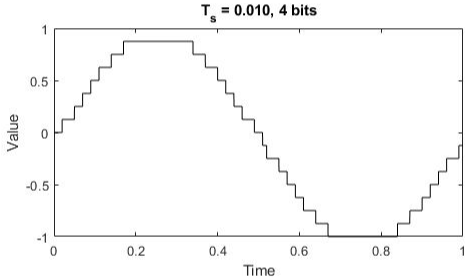
- Quantization
  - Divides sensor’s range into discrete number of “steps”
- Bit depth
  - Number of steps (2^number of bits)

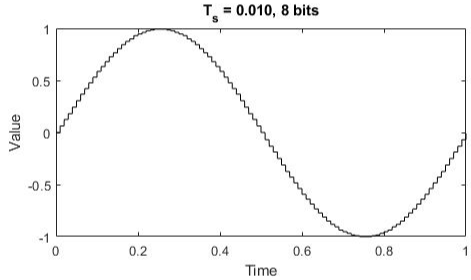

T_s = 0.020, 2 bits
Value
Time

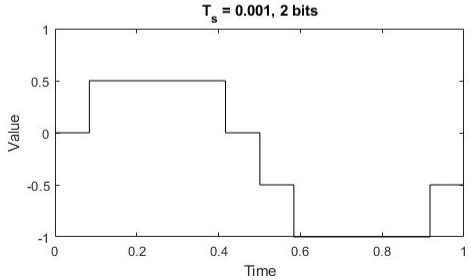

T_s = 0.020, 4 bits
Value
Time

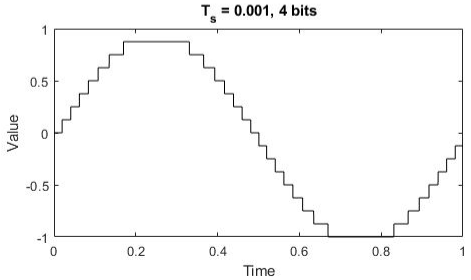

T_s = 0.020, 8 bits
Value
Time

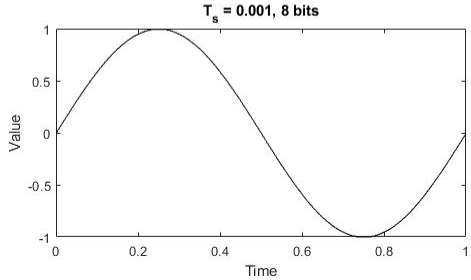

T_s = 0.010, 2 bits
Value
Time


T_s = 0.010, 4 bits
Value
Time


T_s = 0.010, 8 bits
Value
Time

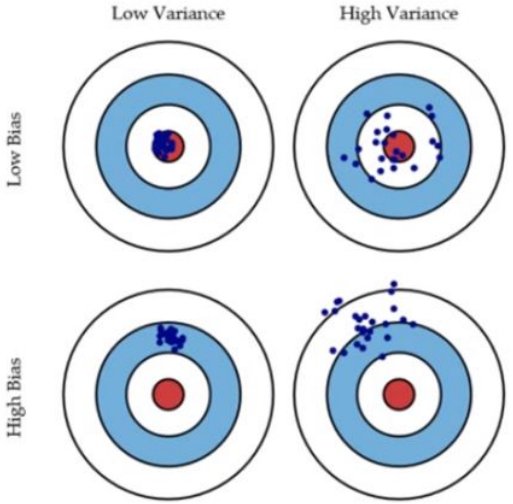

T_s = 0.001, 2 bits
Value
Time


T_s = 0.001, 4 bits
Value
Time


T_s = 0.001, 8 bits
Value
Time

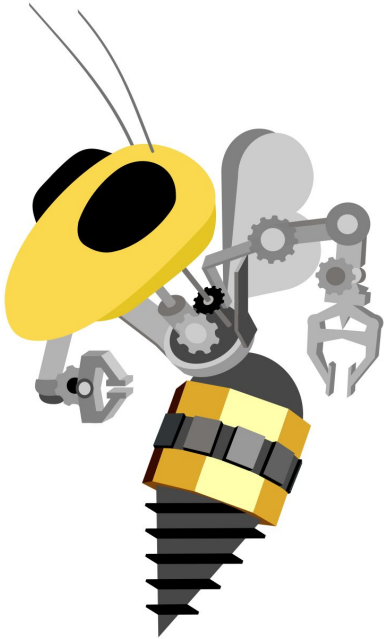
Error

- Incorrect readings which cloud sensor measurements
  - Error terms are added to “true” measurement of state variable
- Types of error:
  - Noise - Zero mean, high variance
  - Bias - Nonzero mean, low variance
- Sources of error
  - Temperature, EMI, vibration, Quantization



Reducing Error

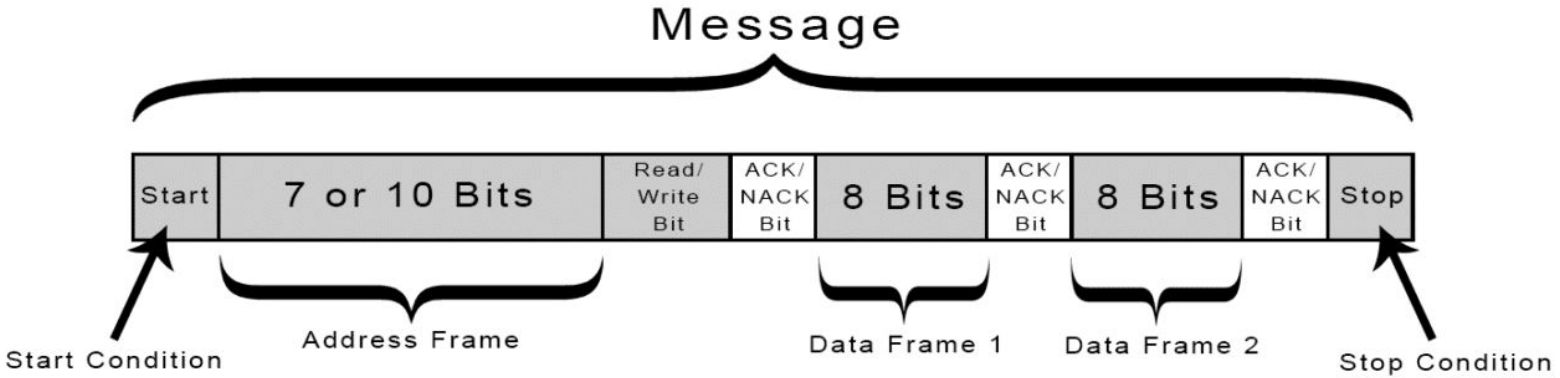
- Calibration
  - Take a series of measurements of a known parameter across sensor range
  - Calculate offset and subtract from subsequent measurements
- Noise
  - Redundant measurements can be averaged to cancel noise effects (ensemble averaging)
  - Estimation/Filtering algorithms can be run to nullify noise in certain frequency ranges



Implementing I2C/TWI

Let's Talk

I2C Packet Format



Atmega328p I2C Module

- Buffer of values to be sent and received are stored in RAM of Arduino
- Libraries handle moving those buffered values to commands on the I2C bus
  - Fill, send, and read from the queue
  - Timing functions to let system know when data is ready

Wire

- Arduino Library for I2C / TWI communication
- Use library:
  - `#include <Wire.h>`
- Need to use specific pins for I2C on most AVR-based Arduinos

I2C Device Organization

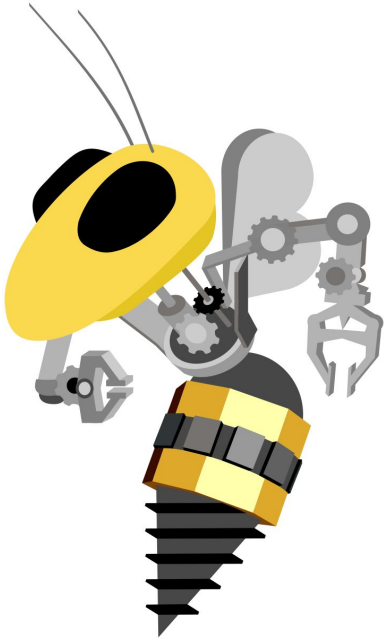
- Devices contain a bank of control and data registers which can be read or written
- Register address indicates next location of reading or writing
  - Master can set the value of the current address and write or request data at that location

Reading and Writing

- **Reading**: send memory location to slave device and sends STOP condition
  - Device responds with the data stored in the requested memory location
- **Writing**: Send memory location and value to write
- In both cases: device automatically increments register address

I2C Example Code

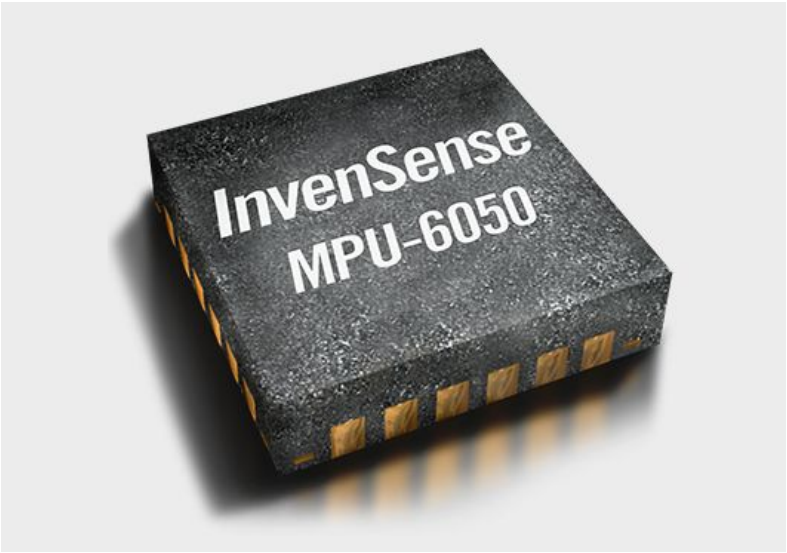
```
Wire.beginTransmission(DEV_ADDR);
Wire.write(REQ_REG); //Queues write
Wire.endTransmission(false); //Sends write
Wire.requestFrom(ADDR, NUM);

while(Wire.available()){
   uint8_t c = Wire.read();
}
```



Lab Time

MPU-6050

- Type of IMU (Inertial Measurement Unit)
  - One of the most common types of sensors in robotics
  - Contains a 3-axis accelerometer and 3-axis gyroscope
- Communicates over I2C



Lab Info

- Initiate Communication with Sensor
- Read from Who Am I register to confirm address
- Write to Power Management Register to turn on
- Read from Acceleration registers to measure force in all directions
- Convert accelerations in 3 axes into angle measurement
- Light up LEDs to make a digital level