# Robotics 311 : How to build robots and make them move

Prof. Elliott Rouse

GSI Yves Nazon MS

Fall 2022

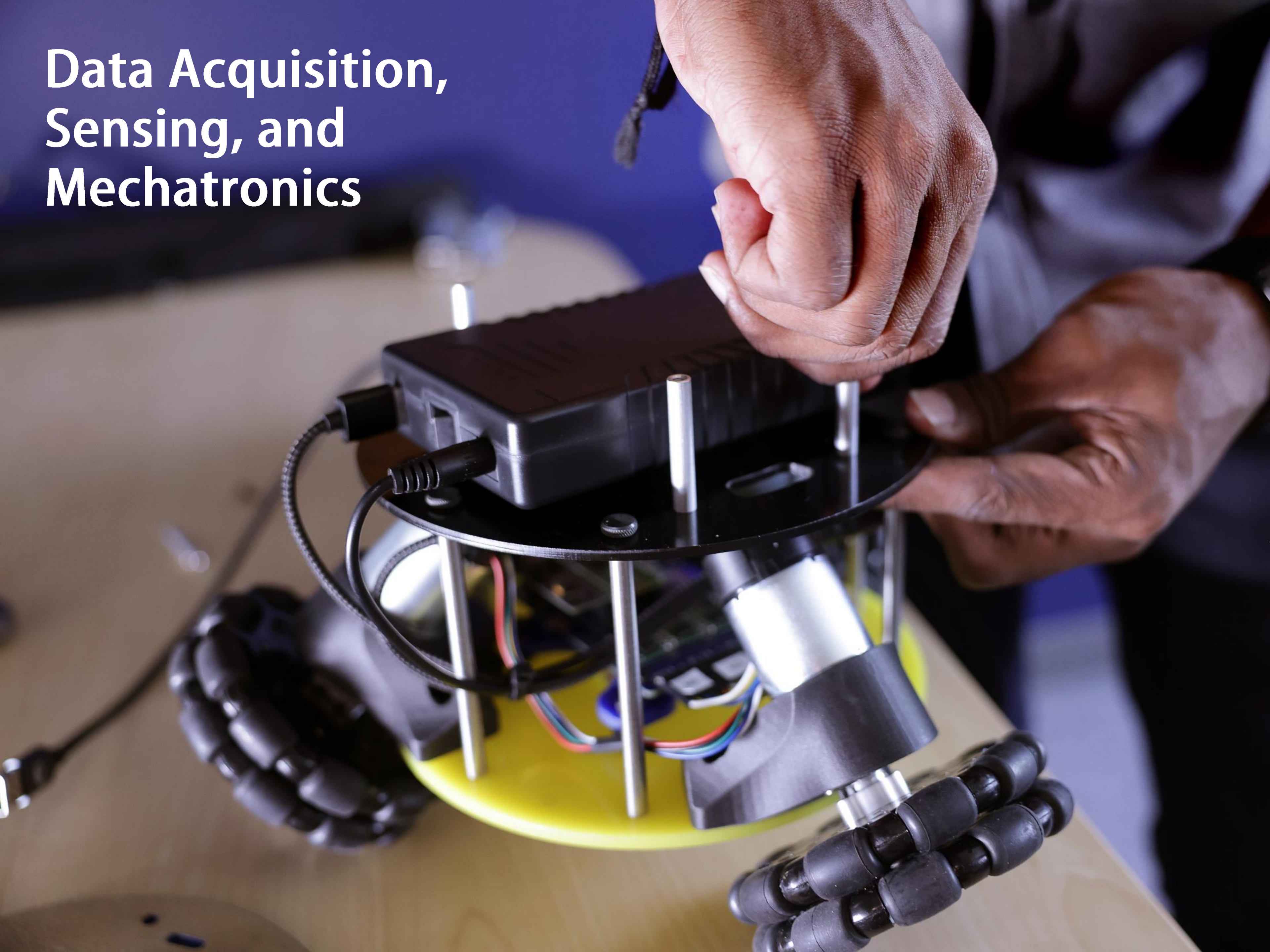
Photo credit: Brenda Ahearn / U-M CoE

# ROB 311 – Lecture 16

- Review IMUs
- Review encoders
- Discuss digital communication
- Discuss omni wheels

- Announcements
  - HW 4 will be posted today!
  - Midterm exam – 11/8 (2 weeks from today)

# Data Acquisition, Sensing, and Mechatronics



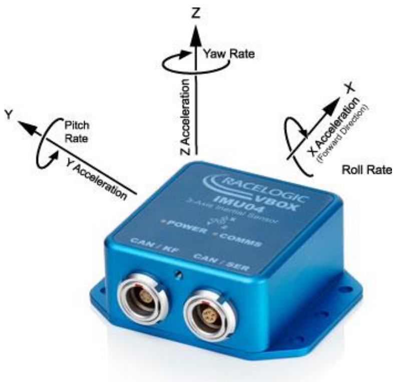
# Sensors

- Many different types of sensors are used in robotic applications
- These sensors convert a physical process into a digital value
- Using sensor characteristics, we can convert the digital value back into physical units
- Common types of sensors:
  - Inertial Measurement Unit – used to tell position / orientation information
    - 3-axis accelerometer
    - 3-axis rate gyroscope
    - 3-axis magnetometer

    IMUs are sometimes described by their number of axes or DOFs (e.g. 6-axis or 9-axis)
    Magnetometers are a more recent addition to IMUs; older versions are 6-axis
  - Encoders - used to sense changes in angular displacement
  - Load sensing – force or torque, used in many robotic applications
  - Vision / LIDAR – used for understanding the environment
  - Many others… In this class, we will focus on IMUs and encoders

# Inertial Measurement Units

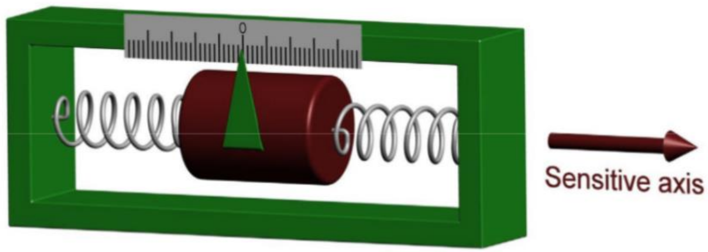
- While there are differences in IMU accuracy and precision, their functions are all relatively similar
- They report $x$, $y$, and $z$ acceleration and $x$, $y$, $z$ angular velocities
- This information can be fused to obtain global changes to position and orientation—proprietary algorithm within the IMU
  - We use this approach from our IMU, which provides position and orientation of the ball-bot
- Most IMUs can have their accel / velocity ranges scaled, depending on use
  - This provides optimal resolution when scaled correctly




# IMU Mechanism of Action

- Accelerometers measure all external forces, including gravity
- Conceptually, an accelerometer is a spring-mass-damper system

Inside an accelerometer—
modern versions are called MEMS
(micro electromechanical system)




- The force applied is related to the mass, damping, and elasticity of the element
- At equilibrium (zero velocity) the acceleration is ${kx}/{m}$
- Three orthogonal axes are used to obtain omnidirectional measurements
- Mechanical / MEMS accelerometers are low-pass, measuring < 500 Hz
- MEMS gyroscopes operate similarly, with a vibrating mass along the radius which deflects according to angular velocity

Piezoelectric accelerometers can sense up to 100 kHz!

# IMU Drift

| Grade | Accelerometer Bias Error [mg] | | Horizontal Position Error [m] 1s | 10s | 60s | 1hr |
|---|---|---|---|---|---|---|
| Navigation | 0.025 | | 0.13 mm | 12 mm | 0.44 m | 1.6 km |
| Tactical | 0.3 | | 1.5 mm | 150 mm | 5.3 m | 19 km |
| Industrial | 3 | | 15 mm | 1.5 m | 53 m | 190 km |
| Automotive | 125 | | 620 mm | 60 m | 2.2 km | 7900 km |

Bias error—error in MEMS mass—and its effect on integrated position measurement error

| Accelerometer Misalignment [mg] | | Horizontal Position Error [m] 1s | 10s | 60s | 1hr |
|---|---|---|---|---|---|
| 0.05° | | 4.3 mm | 0.43 m | 15 m | 57 km |
| 0.1° | | 8.6 mm | 0.86 m | 31 m | 110 km |
| 0.5° | | 43 mm | 4.3 m | 150 m | 570 km |
| 1° | | 86 mm | 8.6 m | 310 m | 1100 km |

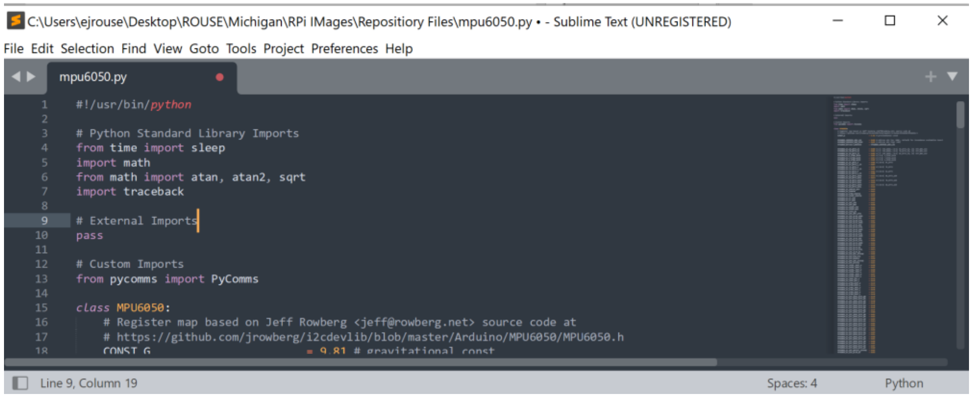
Mounting misalignment error and its effect on integrated position measurement error

| Grade | Gyro Angle Random Walk (ARW) [deg/√hr] | | Horizontal Position Error [m] 1s | 10s | 60s | 1hr |
|---|---|---|---|---|---|---|
| Navigation | 0.002 | | 0.01 mm | 0.1 mm | 1.3 mm | 620 m |
| Tactical | 0.07 | | 0.1 mm | 3.2 mm | 46 m | 22 km |
| Industrial | 3 | | 10 mm | 0.23 m | 3.3 m | 1500 km |
| Automotive | 5 | | 20 mm | 0.45 m | 6.6 m | 3100 km |

Gyroscopic random walk (drift) and its effect on integrated position measurement error

# Ball-Bot IMU

- We are using an MPU-6050 knockoff, which was originally created by Invensense
- We use its internal Digital Motion Processor (DMP) which determines the position and orientation for us
  - Why is this hard?
  - We only know linear acceleration and angular velocity—obtaining position / orientation requires integration, which causes drift
  - Complementary / Kalman filters are used to combine the accel / velocity signals inside the IMU
- In our system, the Pico acquires the IMU data—so, interacting with the IMU has been abstracted from you
- A Raspberry Pi can also use this IMU, but it needs a driver / API
- MPU-6050 Python driver / API uploaded to Canvas as an example



# Encoders

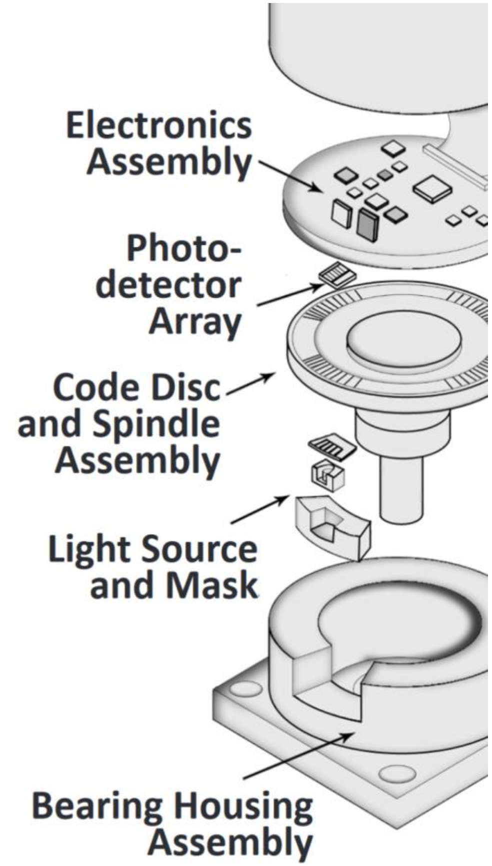
- Encoders are used to measure change in angular position
- Extremely common in robotics and often required for low-level motor control
- Two categories of encoders: relative and absolute
  - Relative: only total traveled distance can be known, but not global orientation
  - Absolute: angle measured relative to global rotation—can measure distance traveled in relation to a fixed global coordinate system
- Many different mechanisms
  - Optical – very common, low cost, and what we're using on our ball-bots
  - Magnetic – very common, low-ish cost, what I use in my research

  We will discuss these

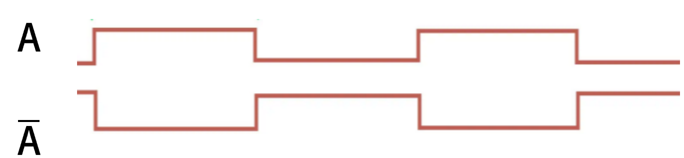
  - Hall effect – common, low-cost, but very coarse
  - Potentiometers – common, low cost, not robust, analog

# Optical Encoders

- Optical encoders used interruption of light to detect rotary motion
- A rotating disk inside encodes opaque lines or patterns
- When rotated, a photodetector senses the pulsing light
- This information is then converted and sent out as pulses for the motor driver
- Signals are often sent with the digital opposite, usually noted with a bar
    - Signals A and $\bar{A}$
- Pros: High resolution, resistant to magnetic interference, low cost
- Cons: Complex / fragile, larger size

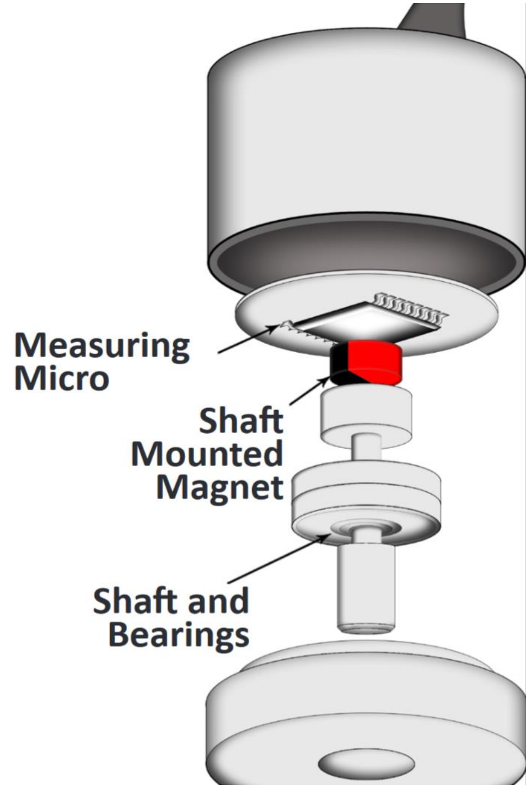



Example encoder signals—high voltage is usually 5V or 3.3V



# Magnetic Encoders

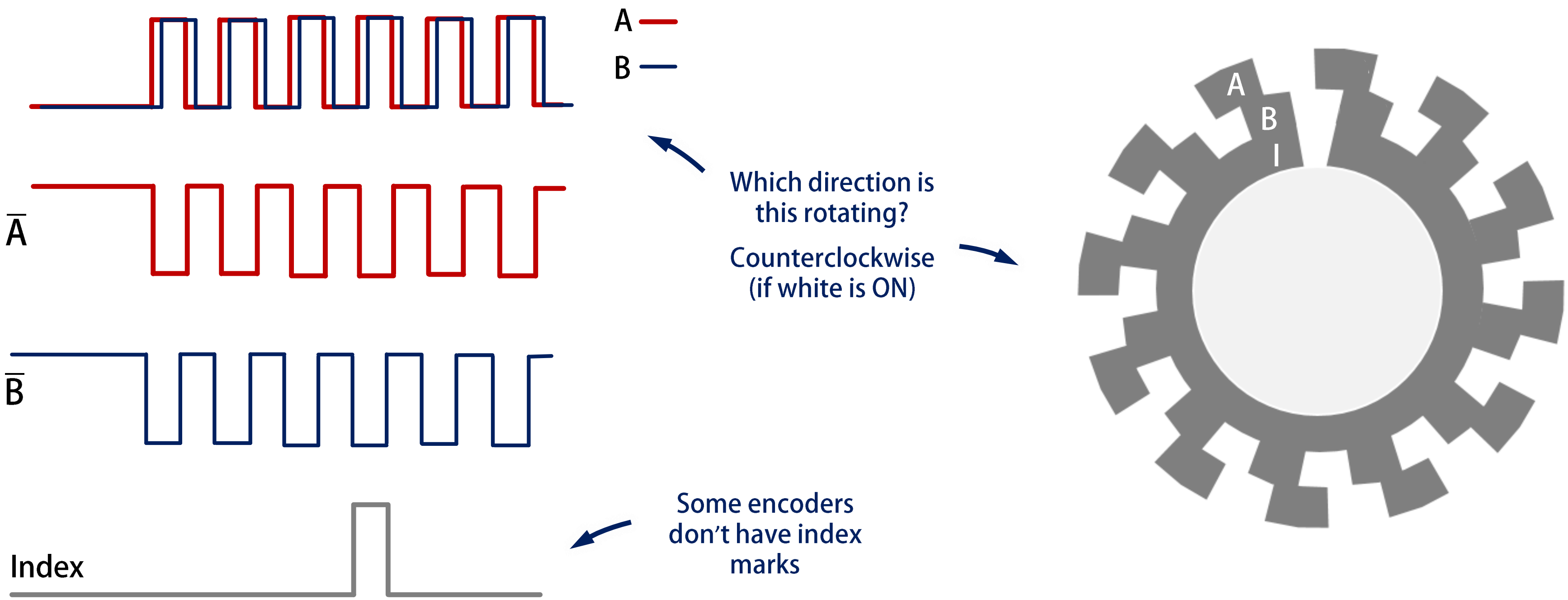
- Magnetic encoders use the change in local magnetic fields to sense rotation
- Typically based on the Hall-Effect, where a magnetic field across a plate creates a voltage
- Magnet is diametrically magnetized and at a specified location (~1-3 mm away from sensor)
- Outputs can have many forms (sine/cosine, digital, pulses)
- Pros: High resolution, absolute, low cost, simple
- Cons: ~susceptible to magnetic interference
- I use the AMS AS5048 encoder in my designs ([link](link))




AMS AS5048

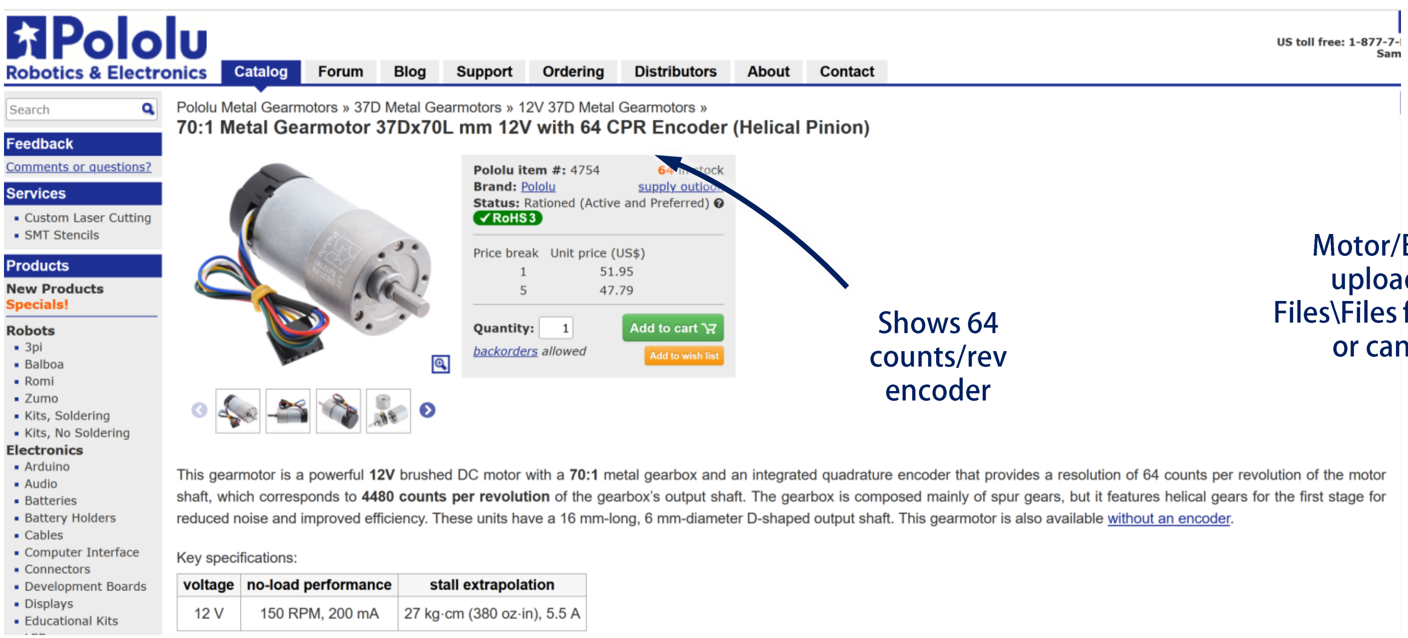
# Quadrature Decoding (Relative)

- Many motors use relative encoders, including ours!
  A key question is knowing the direction a motor is going
- This is accomplished using quadrature decoding: two encoders are positioned a small phase shift from each other
- The signal that rises first is used to know direction
- Indexing is used to know absolute position from a relative encoder



# Quadrature Decoding (Relative)

- Described by ‘counts / revolution’
- When digital negatives are used, the encoder counts can be multiplied (e.g. 4x)
- Encoders are often discussed / purchased with a motor
- For Pololu, encoder wiring is described with the motors datasheet
- Find the color ‘pinout’ for the encoder Pololu 37D motor.




Shows 64 counts/rev encoder

Motor/Encoder datasheet uploaded to Canvas at: Files\Files for Lecture\Lecture 16 or can be found online


M | ROBOTICS

# Quadrature Decoding (Relative)

- [Pololu 37D datasheet]()

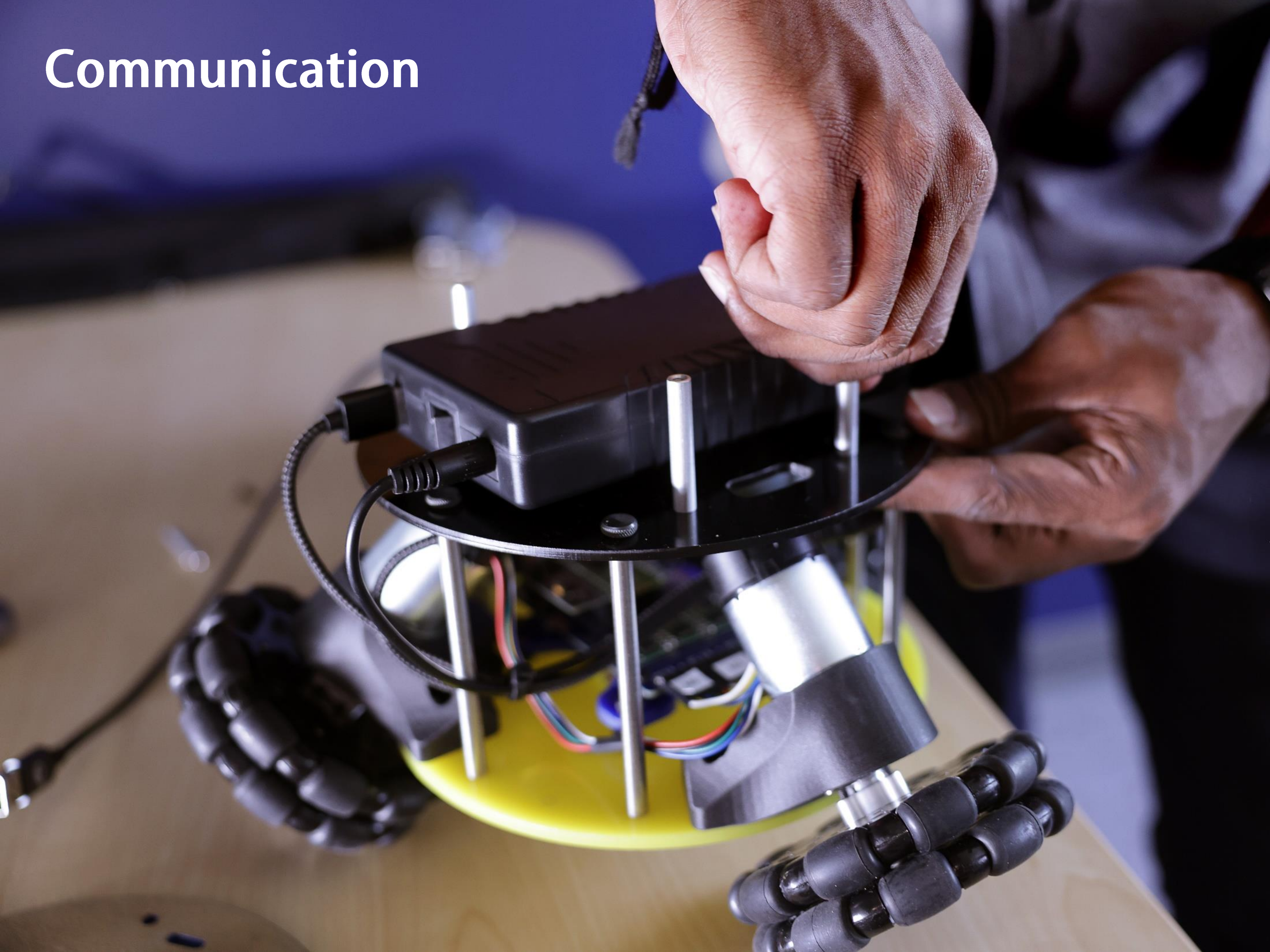
## 37D Metal Gearmotors

Pololu

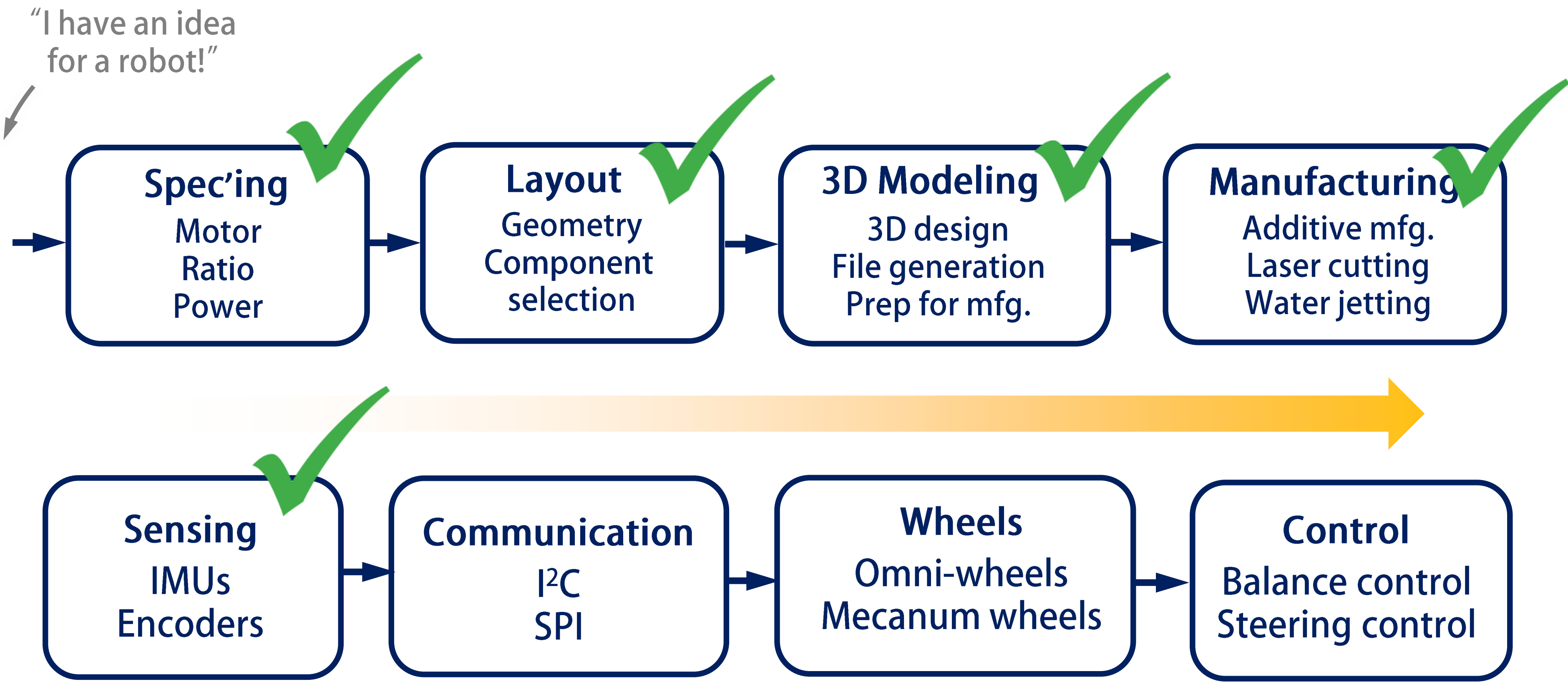
This describes how we plugged the motors into the Pico board

| Lead Color | Function |
|---|---|
| Red | Motor power |
| Black | Motor power |
| Green | Encoder ground |
| Blue | Encoder Vcc (3.5 V to 20 V) |
| Yellow | Encoder A output |
| White | Encoder B output |

# Communication



# Reminder on Where We Are

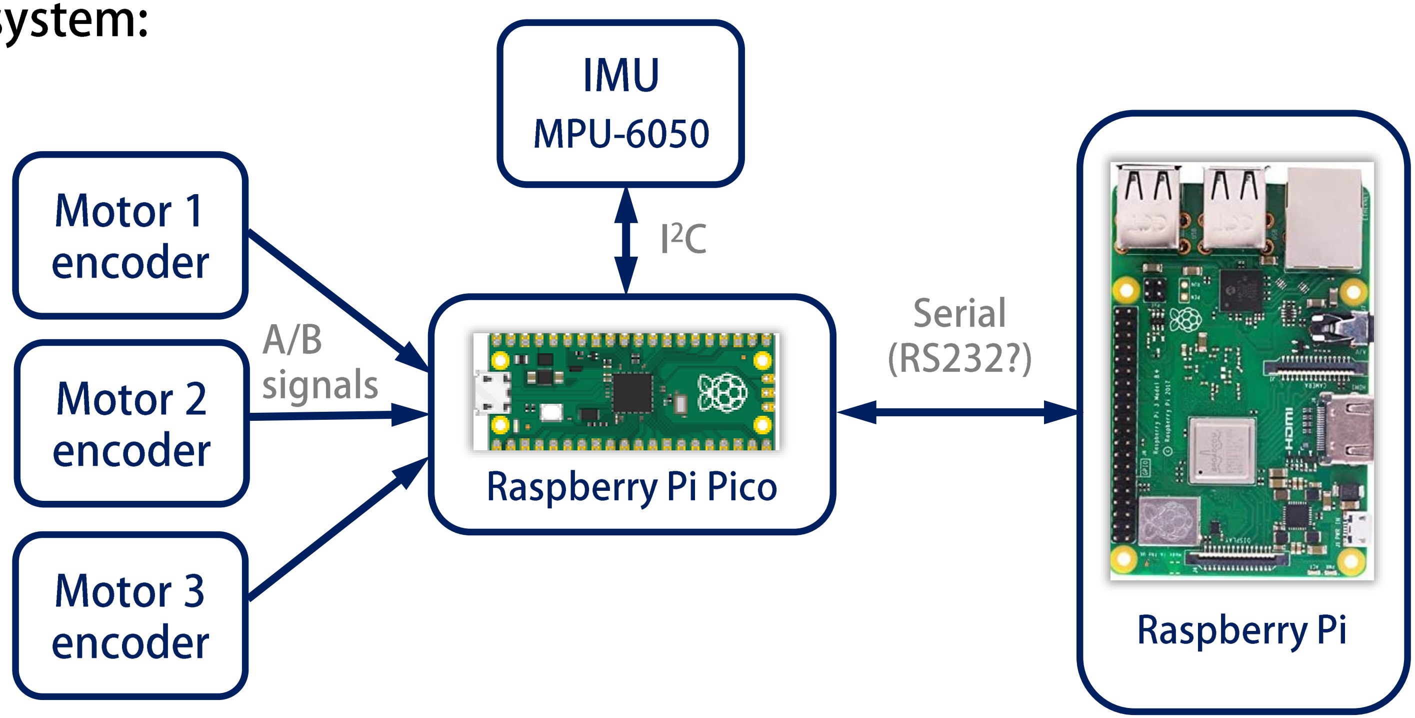



- Lets take a minute to remember where we are in the process
- At this point, we can (hopefully) see a framework to *build a robot and make it move*

Our system doesn't use SPI but it's good to know

# Communication and High-Level Control

- We need to choose a platform for 'high-level' control
- These take on many forms (RPi, Arduino, Beaglebone, Laptop, PIC, etc)
- Raspberry Pis are common controllers—extensive online community
- Choosing sensors / communication / high-level control often relies on what you've seen
- Our system:



# Digital Numbering

- Digital communication comes from descriptions of integer numbers as 1s and 0s
- Review of base-10 numbers:

$$4951_{10} = (4 \times 10^3) + (9 \times 10^2) + (5 \times 10^1) + (1 \times 10^0)$$

- Most common formats are:
  - Binary – base 2 {0 , 1} – helpful for digital communication
  - Hexadecimal – base 16 {0, 1, ..., 9, A, ..., F} - helpful for byte-centered math
- One 0 or 1 is a *bit*, 8 bits is a *byte*
- Convention: $2_{10} = 10_2 = 0\text{b}10$
- Binary to decimal conversion is easy, decimal to binary is harder

$$0\text{b}110 = 110_2 = (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 6_{10}$$

- Hexadecimal is another common base seen in mechatronics
- Convention: $16_{10} = 10_{16} = 0\text{x}10$

$$0\text{xA}5 = \text{A}5_{16} = (10 \times 16^1) + (5 \times 16^0) = 165_{10}$$

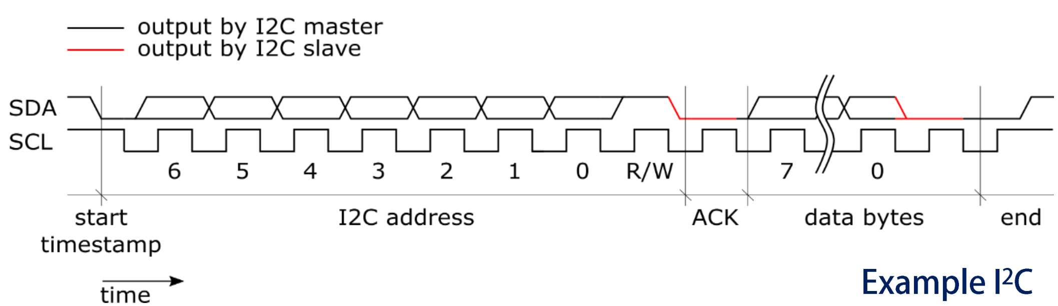
Many online format converters exist, like this one

# Digital Communication

- Digital communication comes in many forms
  - Inter-Integrated Circuit bus ($I^2C$ or I2C)
  - Serial Peripheral Interface (SPI or "spy")
  - Universal Serial Bus (USB)

We will discuss these

- Usually a data line and clock line that communicate information and timing
- Different sensors / chips on the bus need to know they are being communicated with – buses handle this differently
- We'll begin with look into $I^2C$ as an example
  - 4 wires
  - SDA – data line
  - SCL – clock line
  - Power / ground
- Chips are defined by their $I^2C$ address

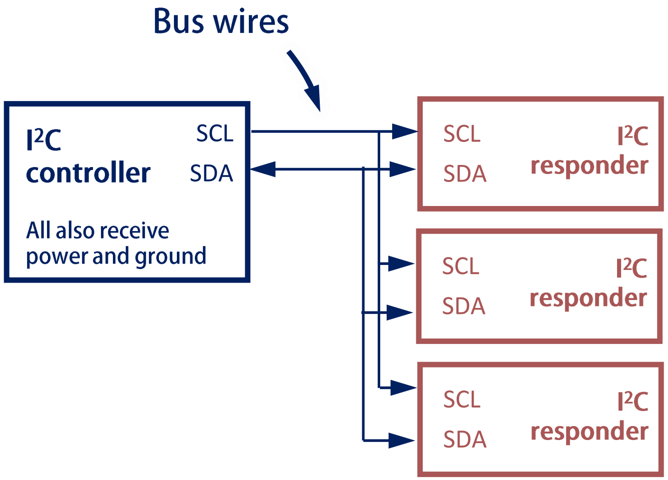


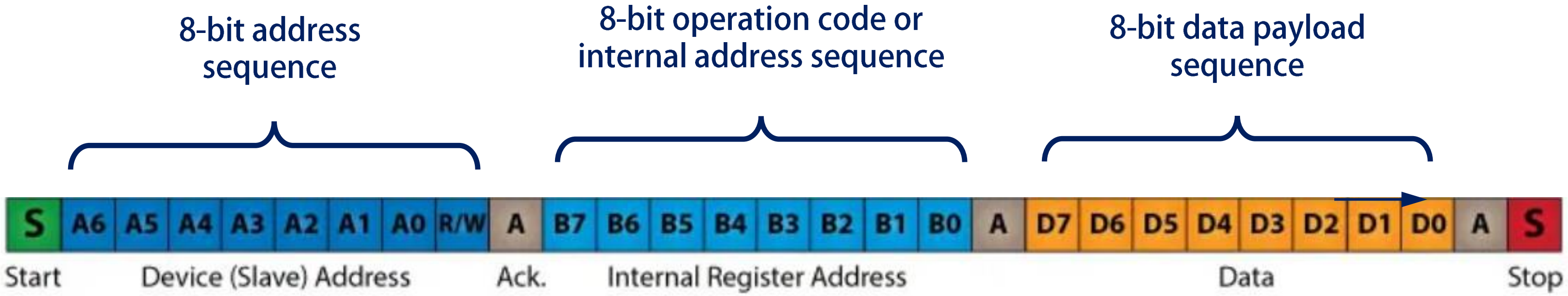

Example $I^2C$ communication

# $I^2C$ Communication

- Each sensor / chip on the $I^2C$ bus is assigned an address
- The address is usually set in hardware
  - Sometimes there are a few options
- The Controller or Master communicates with the Responders or Slaves
- This is defined in each message
- The address information and message structure is provided in the sensor datasheet

Sensors are addressed like a house on a street

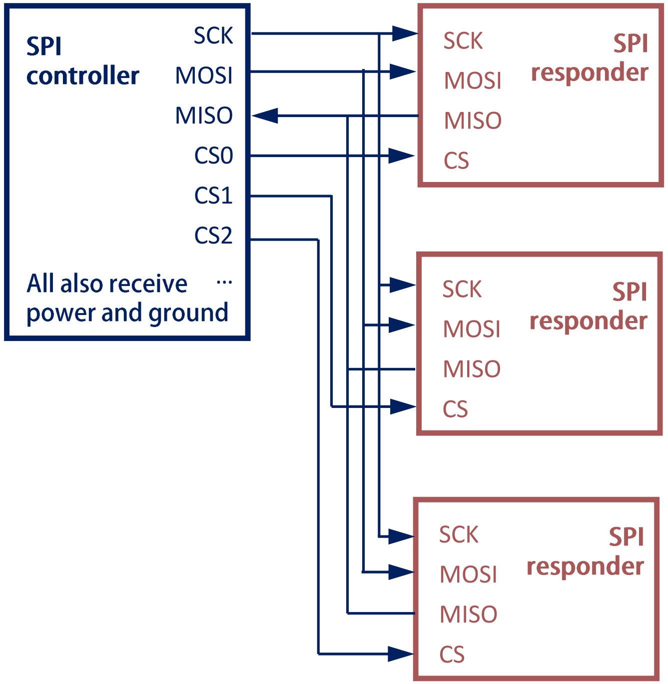


# I²C / SPI Communication

- I²C enables multiple controllers and multiple responders
- Speed is regulated by the clock line
- Half-duplex – only one sensor can communicate at a time
- Older and slower communication (typ. ~400 kB/sec)
- Mainly used for short distances—up to ~1 m unshielded

- **Serial Peripheral Interface (SPI)**
- 5+ wire bus (inc. power / ground)
- MISO – Data line – Master in, Slave out
- MOSI – Data line – Master out, Slave in
- SCK – clock line
- Slave select / Chip select – line for each chip
- Power / ground



# SPI Communication

- SPI does not use addresses – instead, it uses the chip select lines to tell each responder that they are receiving a message
- Full duplex – bidirectional communication possible with two data lines
- Only supports a single controller / master
- Faster communication (<1 MB/sec) – 2x+ that of $I^2C$
- More recently developed

- We've already talked about software drivers / APIs, lets look more closely now that we know about communication

blinkm.thingm.com

**BLINKM DATASHEET**
BlinkM
BlinkM MinM
BlinkM MaxM

**Description**

BlinkM is a "Smart LED": a networkable and programmable full-color RGB LED for hobbyists, industrial designers, prototypers, and experimenters. It is designed to allow the easy addition of dynamic indicators, displays, and lighting to existing or new projects.

The BlinkM family of Smart LEDs makes it easy to add dynamic stand-alone lighting no matter what your project.

If you've used up all your microcontroller PWM channels controlling RGB LEDs and still want more, BlinkM is for you.

BlinkM | BlinkM MinM | BlinkM MaxM

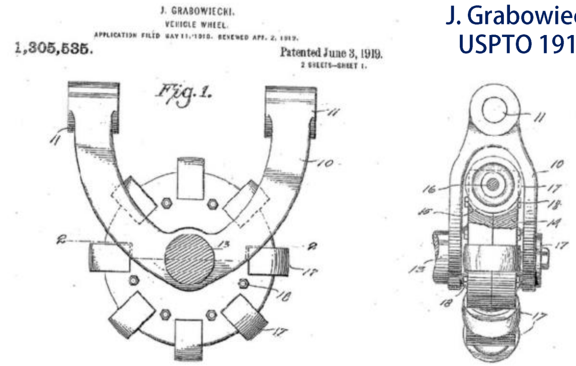
[BLINKM Datasheet](#) uploaded to Canvas

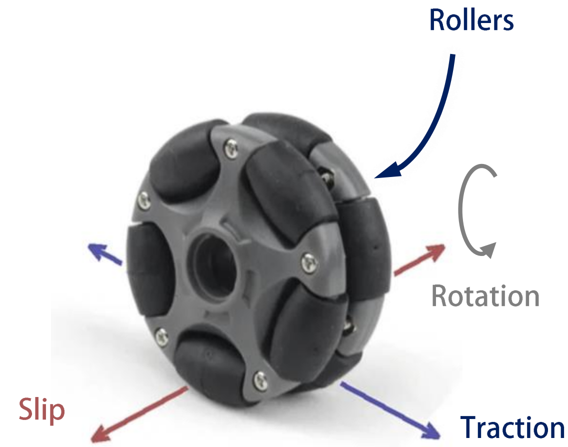
Quiz!

# Omni Wheels

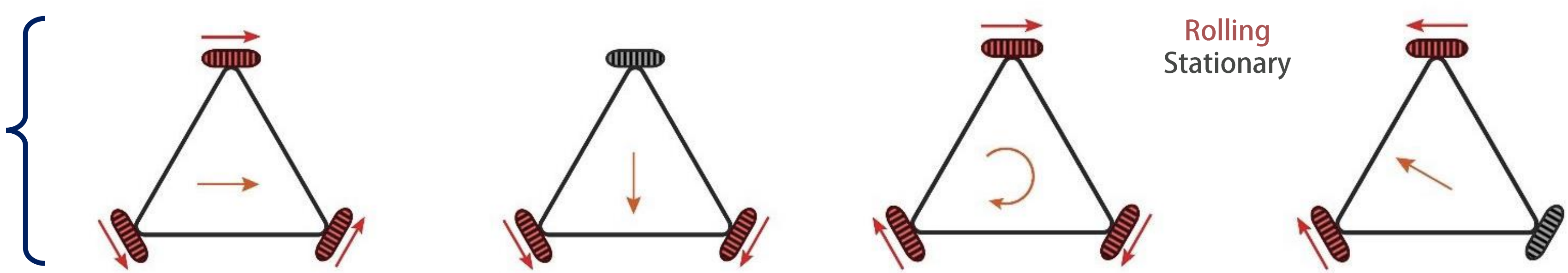
- Small disks / rollers surround the circumference of an outer wheel
- Rollers roll perpendicular to the turning direction
- In some configurations, omni-wheels enable full specification of position and orientation of a robot (holonomic)
- Usually placed in an angled configuration
- Rolling contact on ball—slipping on inside roller
- Many sizes / shapes with different number of rollers / smoothness

J. Grabowiecki
USPTO 1919




Omni-wheel holonomic motion





M | ROBOTICS

# Mecanum Wheels

- Four wheeled systems can use mecanum wheels
- Enable holonomic motion
- They provide lateral motion with canted rollers
- Torque provides two known components of force

Example mecanum wheels

Force vectors caused by motor torque