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 17

- Review digital communication
- Discuss omni wheels
- Break to work on ball-bots

- Announcements
  - HW 4 posted—don't wait until the last minute
  - Due 11/10 at class start
  - Lab plot assignment posted to Canvas—due by next lab
  - Midterm exam – 11/8
  - Review class – 11/3

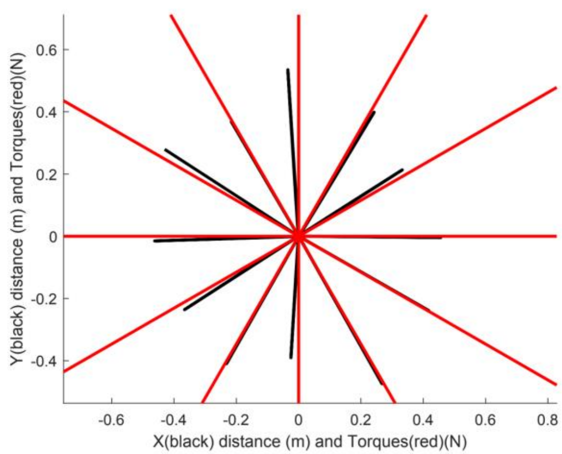
Lab 9

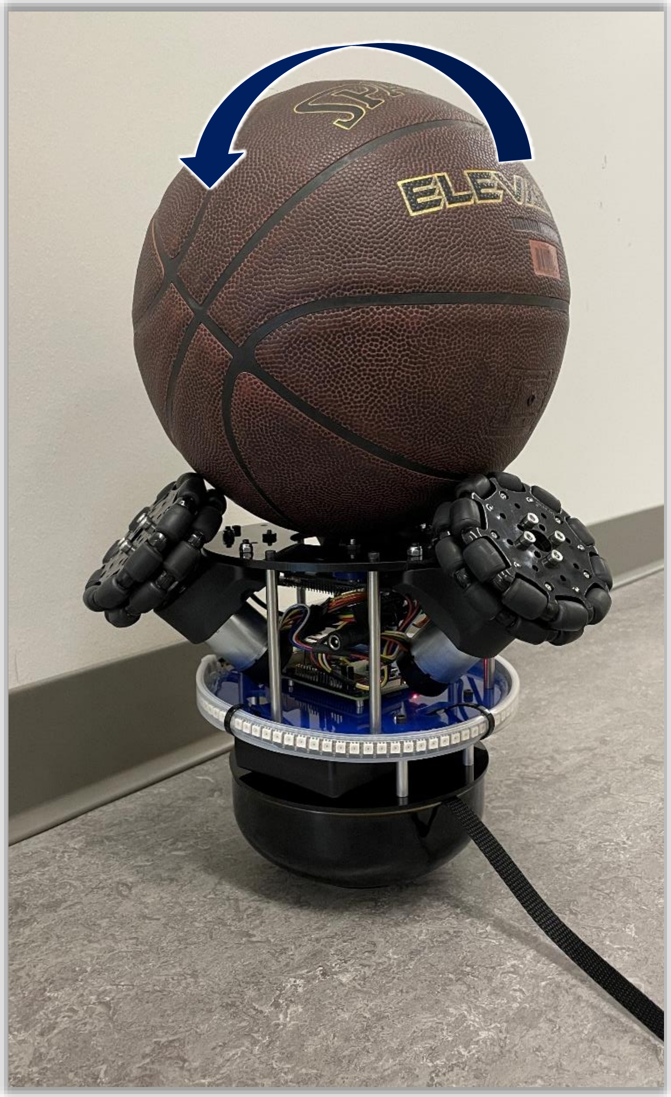
- Yesterday's lab provided our first ability to send power to the motors

```
commands['motor_1_duty'] = T1
commands['motor_2_duty'] = T2
commands['motor_3_duty'] = T3
ser_dev.send_topic_data(101, commands)
```

- Your goal was to rotate the ball using torques applied in ~8 – 10 directions
- Plot the difference in motion direction between the applied torque and motion

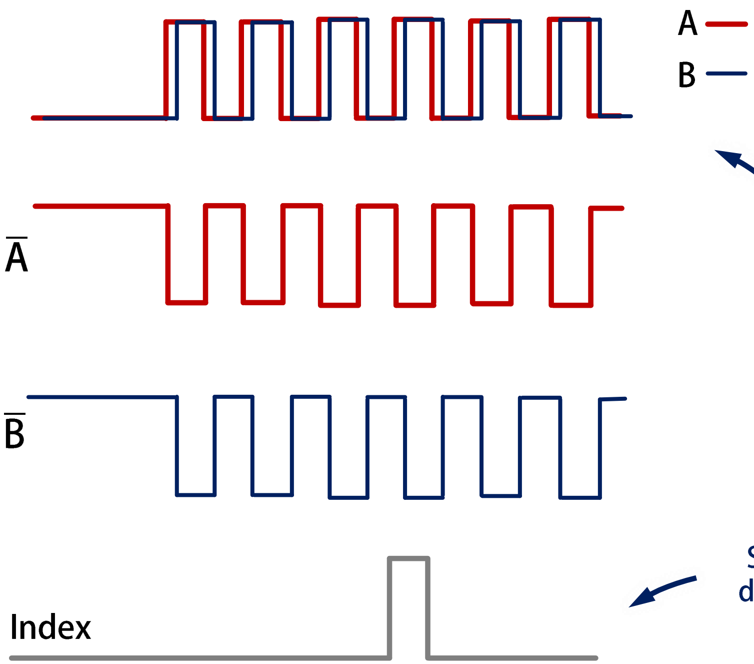
Your plot should look like this





Quadrature Decoding (Relative)

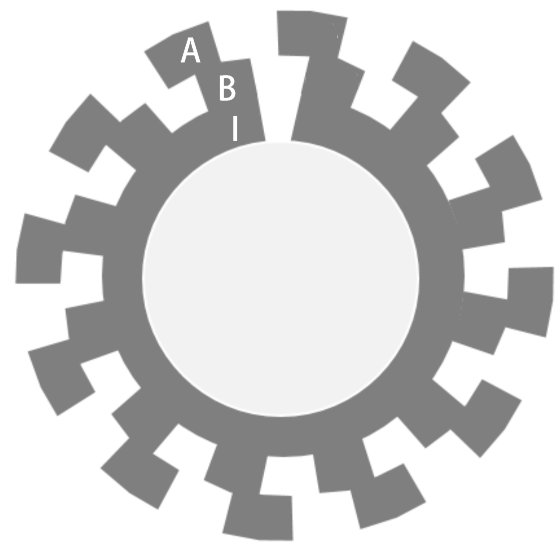
- Many motors use relative encoders, including ours!
  A key question is knowing the direction a motor is going
- This is accomplished using quadrature decoding: two encoders are positioned a small phase shift from each other
- The signal that rises first is used to know direction
- Indexing is used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



Which direction is this rotating?

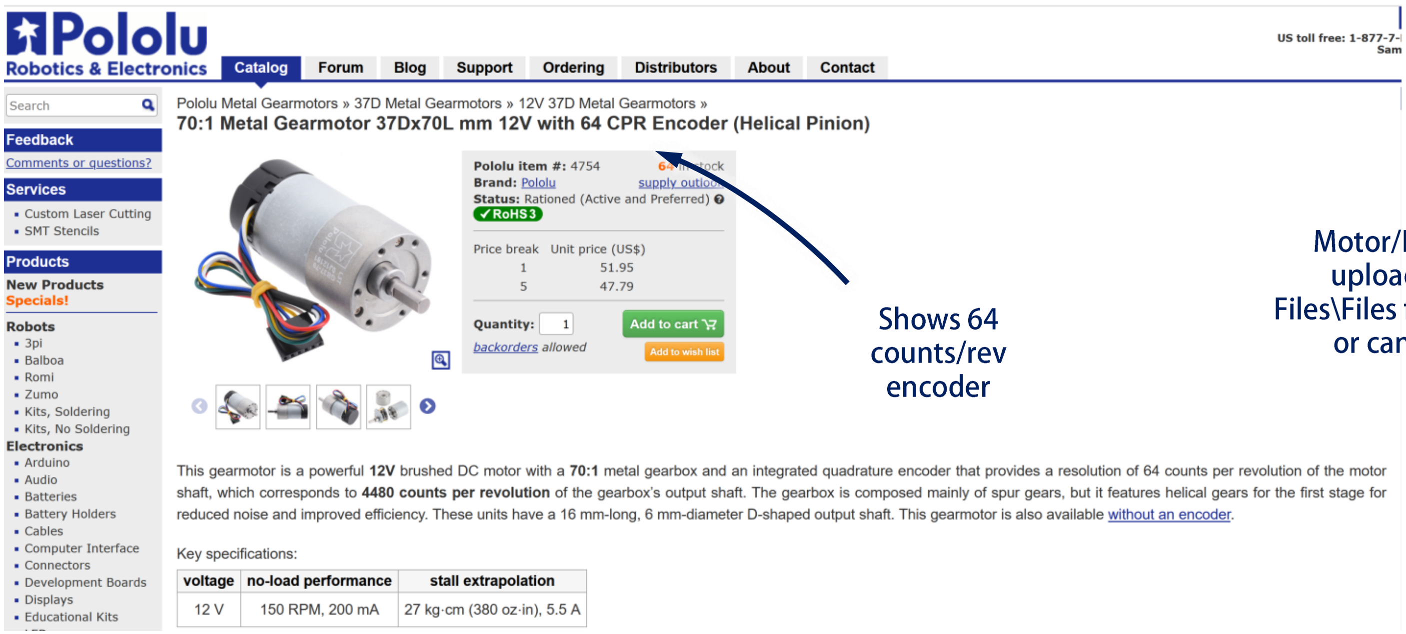
Counterclockwise (if white is ON)

Some encoders don’t have index marks



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]

Quadrature Decoding (Relative)

- [Pololu 37D datasheet]

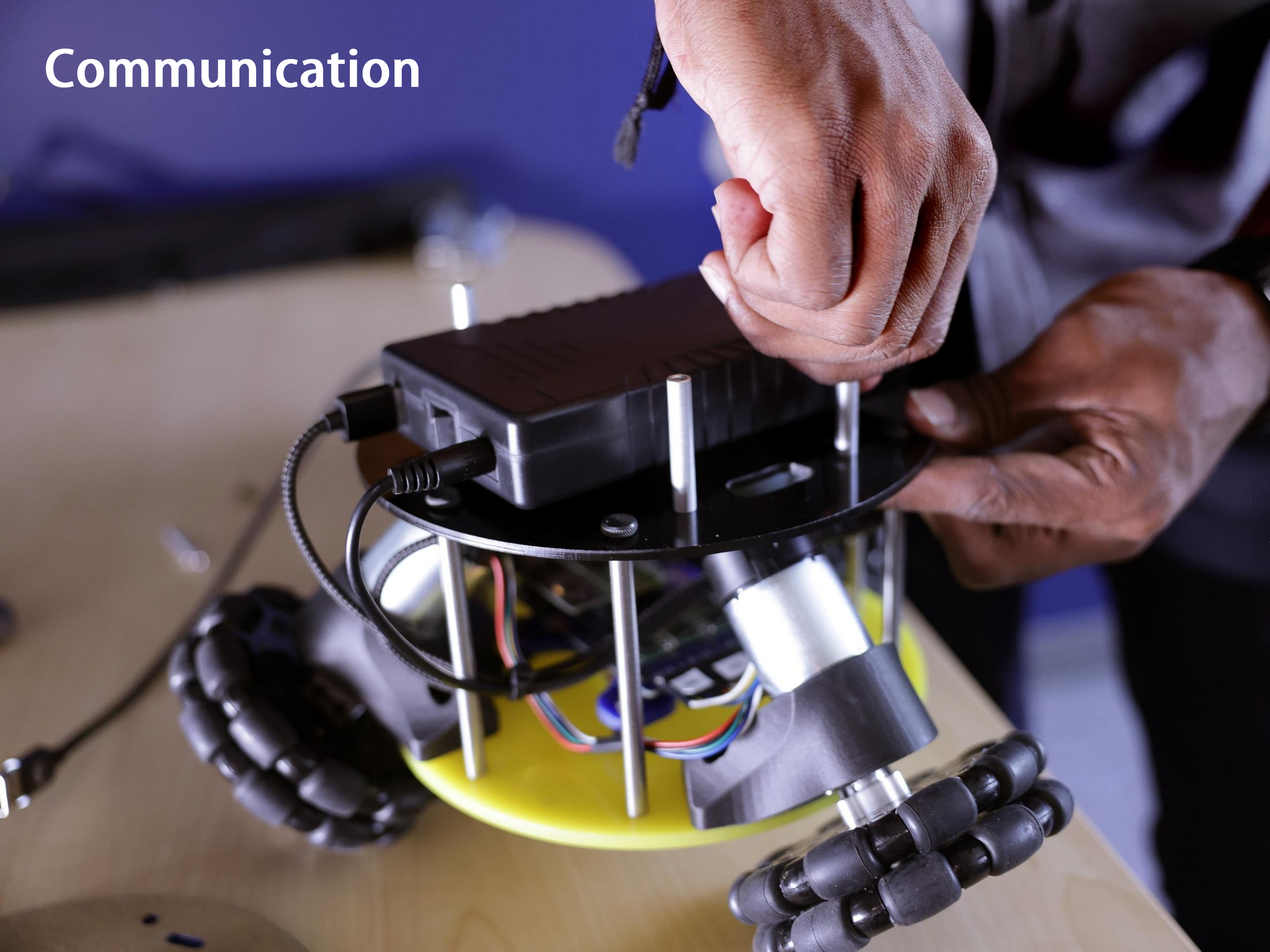
37D Metal Gearmotors

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

"I have an idea for a robot!"

Spec'ing
Motor
Ratio
Power

Layout
Geometry
Component selection

3D Modeling
3D design
File generation
Prep for mfg.

Manufacturing
Additive mfg.
Laser cutting
Water jetting

Sensing
IMUs
Encoders

Communication
I^{2}C
SPI

Wheels
Omni-wheels
Mecanum wheels

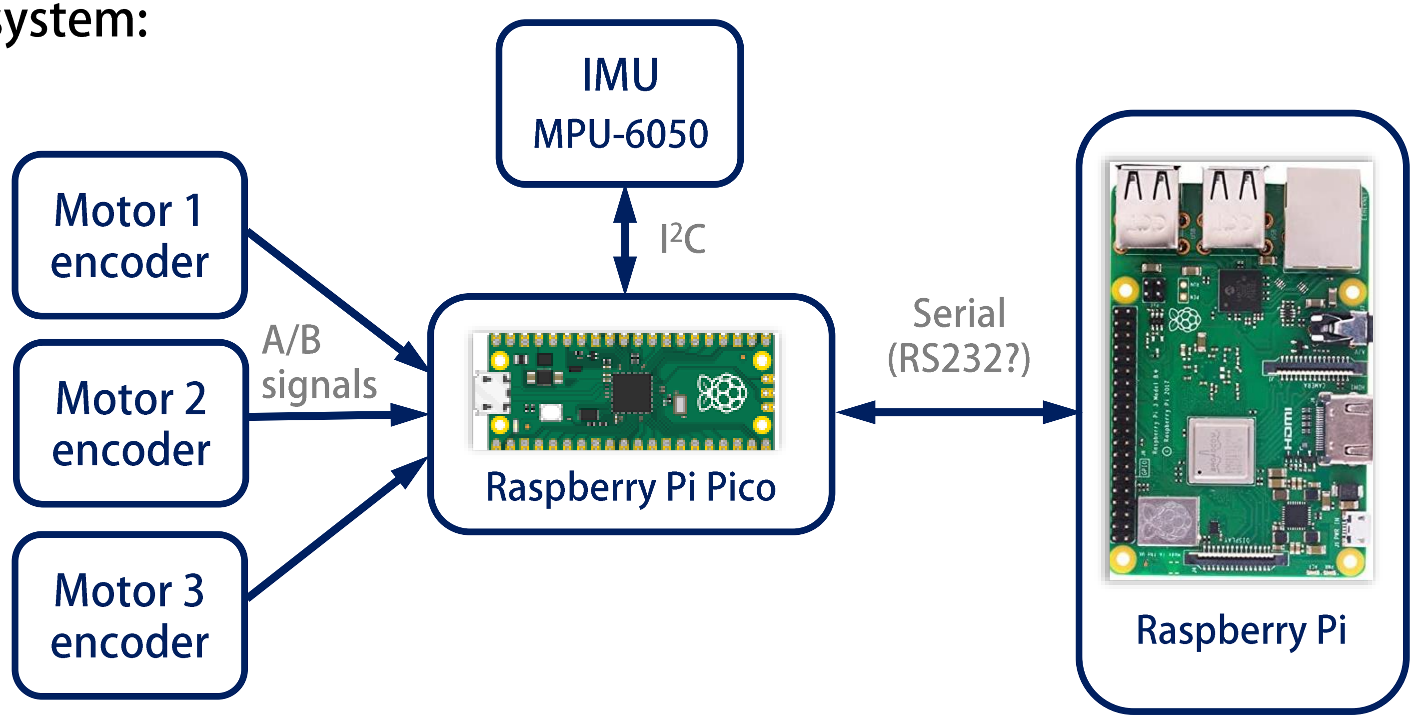
Control
Balance control
Steering control

- 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 or mecanum wheels, but they're 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:

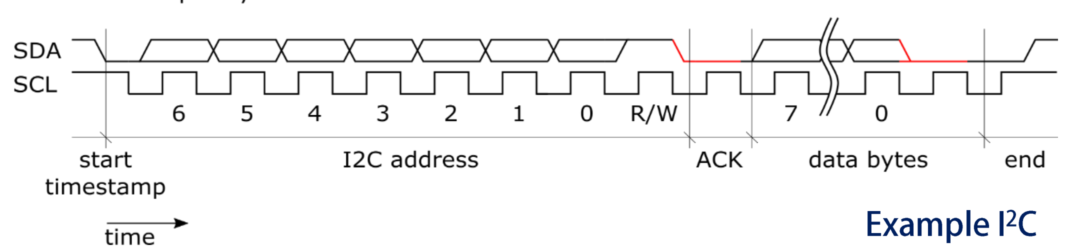


Digital Communication

- 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

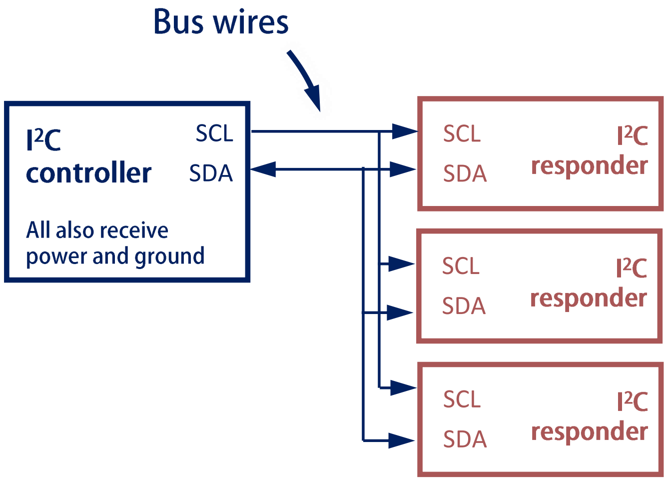


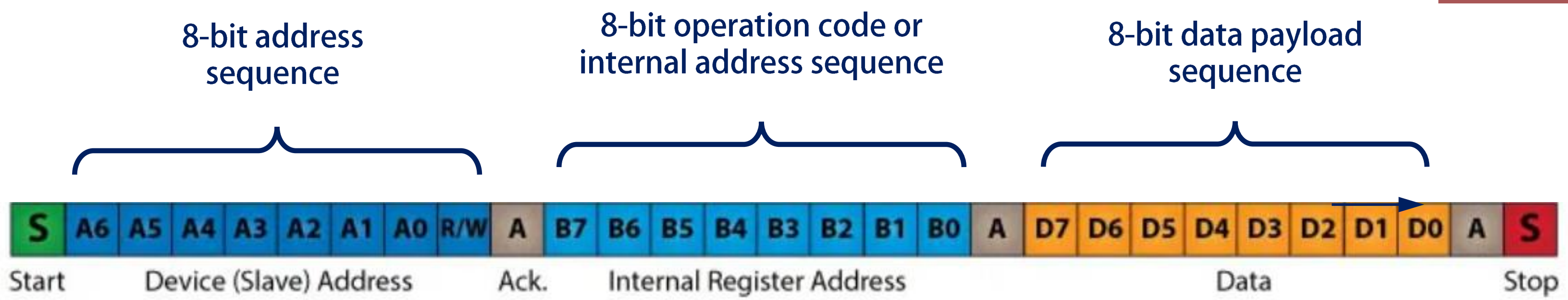
Example I^2C communication

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

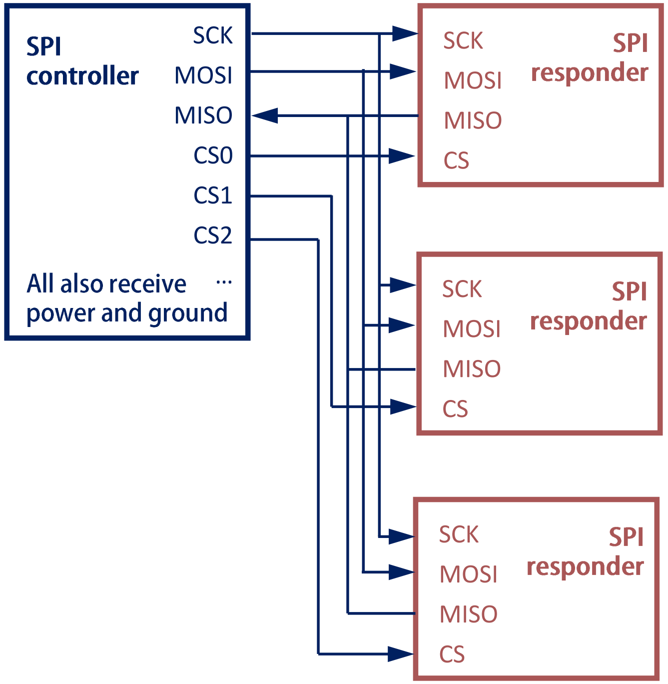




I^2C / SPI Communication

- I^2C 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^{2}C
- More recently developed

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

BLINKM DATASHEET
BlinkM
BlinkM MinM
BlinkM MaxM

blinkm.thingm.com

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.

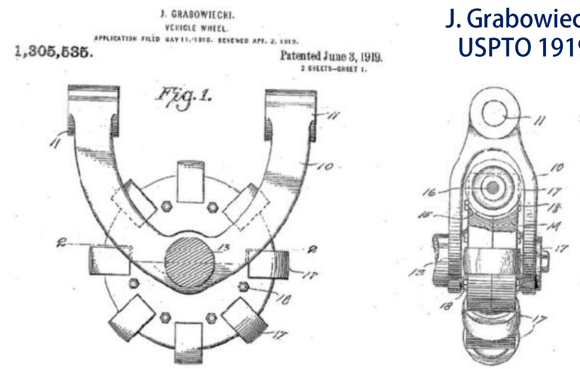
BlinkM | BlinkM MinM | BlinkM MaxM

[BLINKM Datasheet](#) uploaded to Canvas

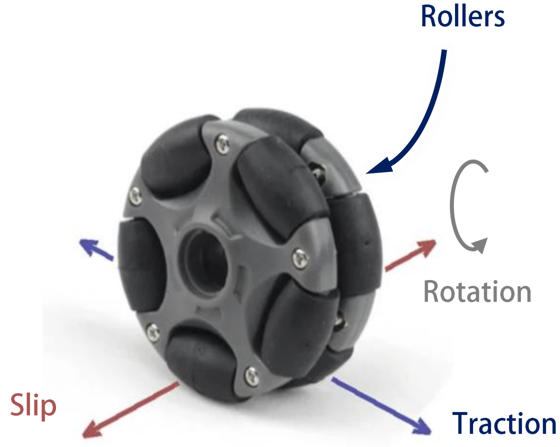
Quiz!

Omni Wheels

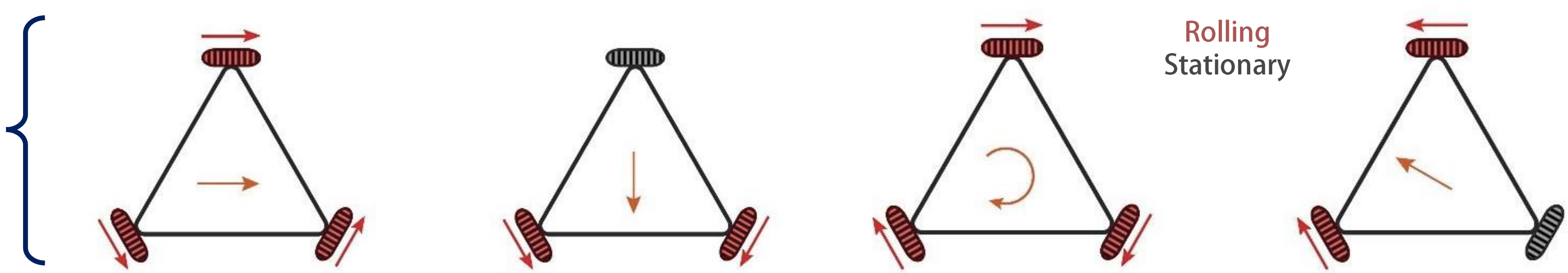
- Small disks / rollers surround the circumference of an outer wheel
- Rollers roll perpendicular to the turning direction
- In some configurations, omni-wheels enable full specification of position and orientation of a robot (holonomic)
- Usually placed in an angled configuration
- Rolling contact on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



J. Grabowiecki
USPTO 1919



Omni-wheel holonomic motion

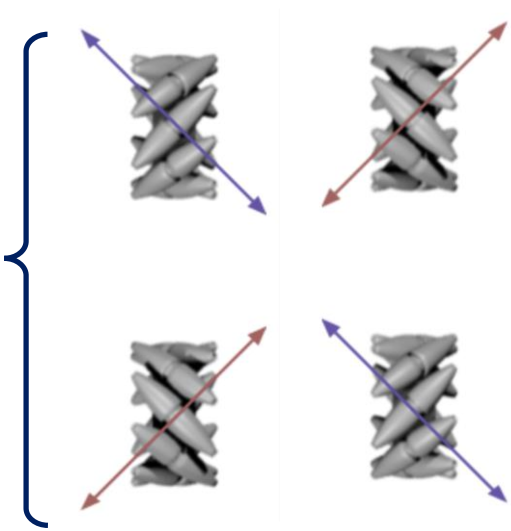


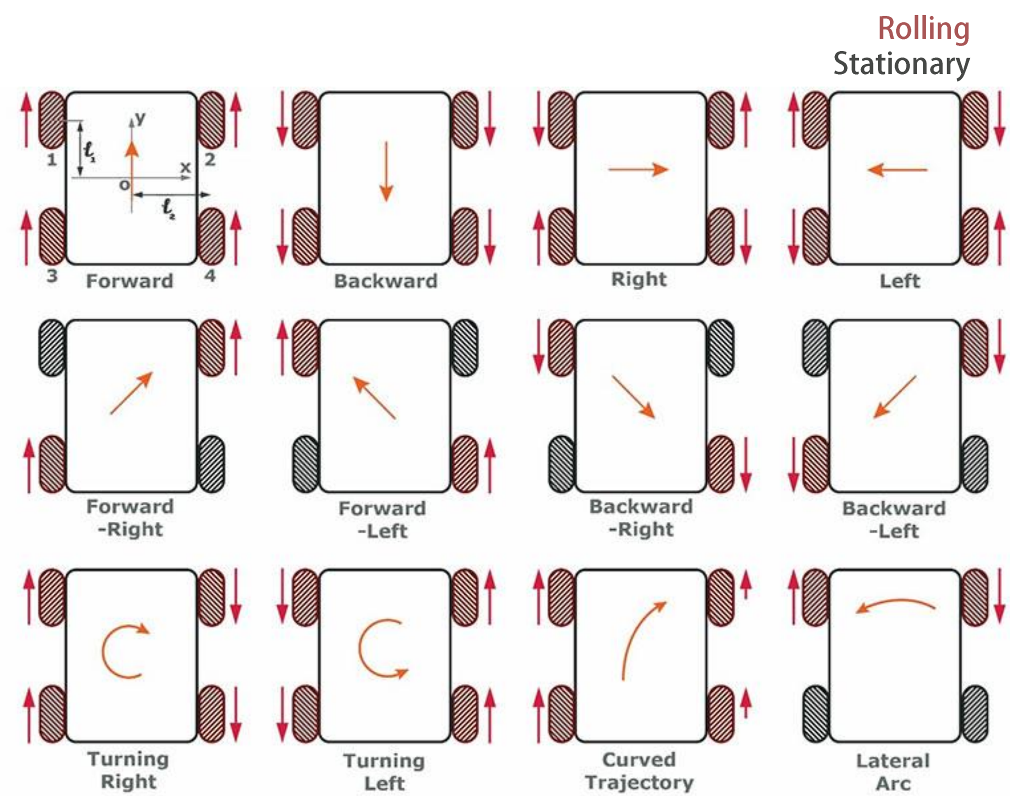
Mecanum Wheels

- 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

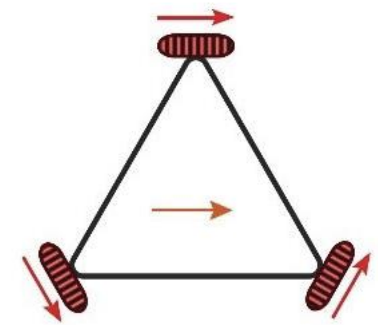
Force vectors caused by motor torque





Omni-Wheel Movement Example

- Set up the equations to move the body at an 80 angle from the x-axis at 1 m/s with no rotation of its body—find the tangential velocity of each wheel for this movement to be possible



Three wheels, each applying a force tangential to their angular velocity direction

We want the sum of these velocities to move the body at a specific angle

$$v_t \sin(80°) = v_1 \sin(\theta_1) + v_2 \sin(\theta_2) + v_3 \sin(\theta_3)$$

$$v_t \cos(80°) = v_1 \cos(\theta_1) + v_2 \cos(\theta_2) + v_3 \cos(\theta_3)$$

$$0 = v_1 + v_2 + v_3$$

v_t= 1 m/s, and the angles are defined by the geometry. These equations could be solved for v_1, v_2, and v_3

Where We Are

"I have an idea for a robot!"

Spec'ing
Motor
Ratio
Power

→ **Layout**
Geometry
Component selection

→ **3D Modeling**
3D design
File generation
Prep for mfg.

→ **Manufacturing**
Additive mfg.
Laser cutting
Water jetting

Sensing
IMUs
Encoders

→ **Communication**
I^2C
SPI

→ **Wheels**
Omni-wheels
Mecanum wheels

→ **Control**
Balance control
Steering control

- This is much of the content for how to design and build a robot
- We will begin control next, and the class make take a more hybrid lab / lecture format.