Using Sensors in your FTC Robot

09/07/2019

Overview of this Presentation

- What sensors are and why we use them
- Types of sensors and what they do
- Hardware connections for sensors
- Configuring sensors in your software
- Best practices for sensor use
- Advanced Topics

Sensors

Sensors measure something about the physical world and put this information into a useful form for your robot's software.

Am I touching the wall?

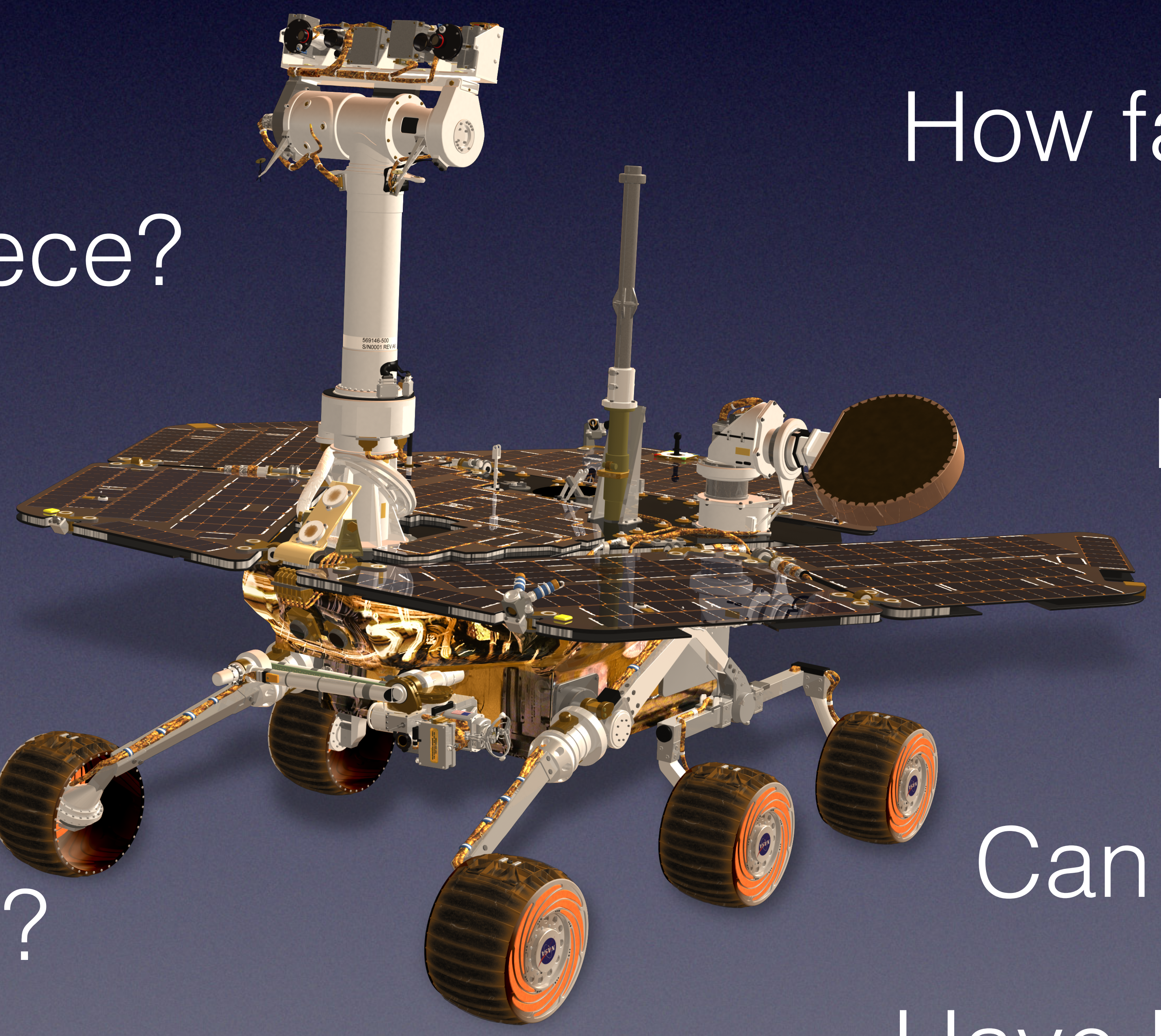
Am I holding a game piece?

What color is the block?

Is the battery voltage too low?

Have I crossed the white line?

How far away is the scoring zone?



How fast is the flywheel spinning?

Is my grabber fully deployed?

How far have I driven?

Can I see the navigation target?

Have I turned 90 degrees yet?

* not a representative FTC robot

What can sensors measure?

Lots of stuff!

- Distance traveled
- Proximity to a target
- Rotation speed of a shaft
- Angles of levers
- Status of robot mechanisms
- Physical forces (acceleration, compass heading)
- Light & Vision
- Passage of Time

Common FTC Sensors

Distance	Proximity	Physical	Advanced
Motor Encoders	IR Distance Sensor	Touch Sensor	Camera
	Ultrasonic Sensor	Accelerometer, Gyroscope	
	Touch Sensor	Color	
		Potentiometer	

Motor Encoders

- Probably the most useful FTC sensor for robot odometry.
    - Use these in your drivetrain to drive precise distances
    - Use in your manipulators, elevators, grabbers
- Motor encoders measure *relative* distances.
    - In other words, encoders can only tell you how far you’ve gone from some starting point.
- The Rev Expansion hubs have software to help you coordinate motors and encoders.

Most common FTC Sensors

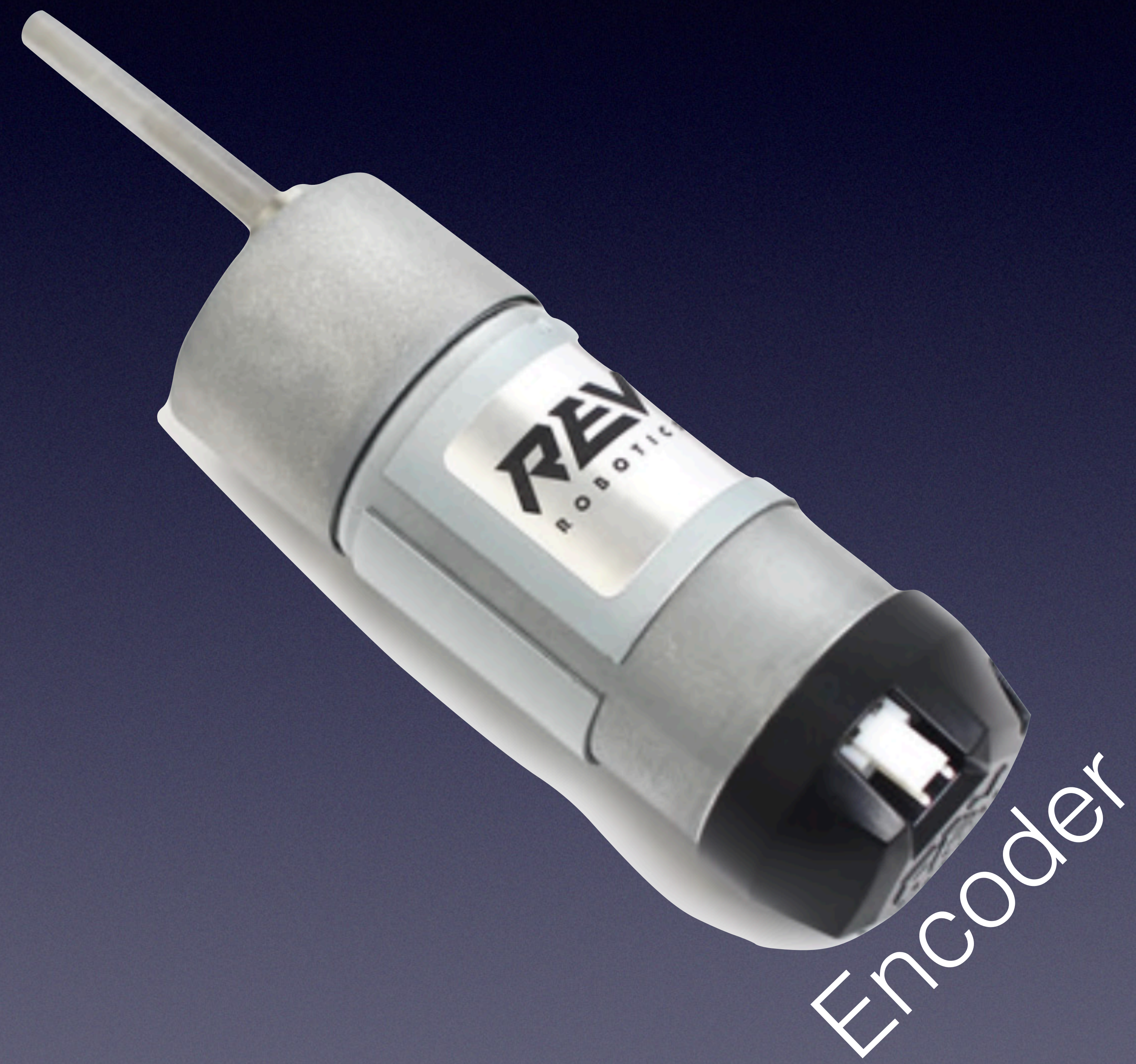
- Motor Encoders
- IMU (Accelerometer & Gyroscope)
- Touch Sensor
- Color Sensor

The Most Common FTC Sensors

Sensor Type	Common Usage
Motor Encoder	Counts shaft rotations, can measure distance robot has traveled, amount of elevator lift, or winch windings, speed of flywheel
Accelerometer/ Gyroscope	Measures robot's compass heading, tilt of robot on any axis.
Touch Sensor, Magnetic Limit	Can detect robot touching a surface Mechanism in a particular state (intake in, game piece acquired, etc).
IR Distance, Ultrasonic	Can measure absolute distances to a target, usually a wall or surface
Color Sensor	Can provide rudimentary color information of a very nearby object. Can detect navigation tapes on the field floor.
Camera	Advanced Sensor. Can identify objects in captured images.

Motor Encoders

- Most common FTC sensor
- Built-in to most Andymark and Rev motors
  - Andymark Neverest, Rev Core Hex, Rev HD Hex.
- Measures the rotation of the motor's shaft.
  - Note that this is *not the output shaft*.
  - You must consider the gear reduction to get correct measurements!



Motor Encoders

Table 1: Core Hex Motor (REV-41-1300) Encoder Specifications

Core Hex Motor (REV-41-1300) Reduction	72:1
Free Speed (RPM)	125
Cycles per Rotation of the Encoder Shaft	4 (1 Rise of Channel A)
Counts per Rotation of the Output Shaft	288 (72 Rises of Channel A)

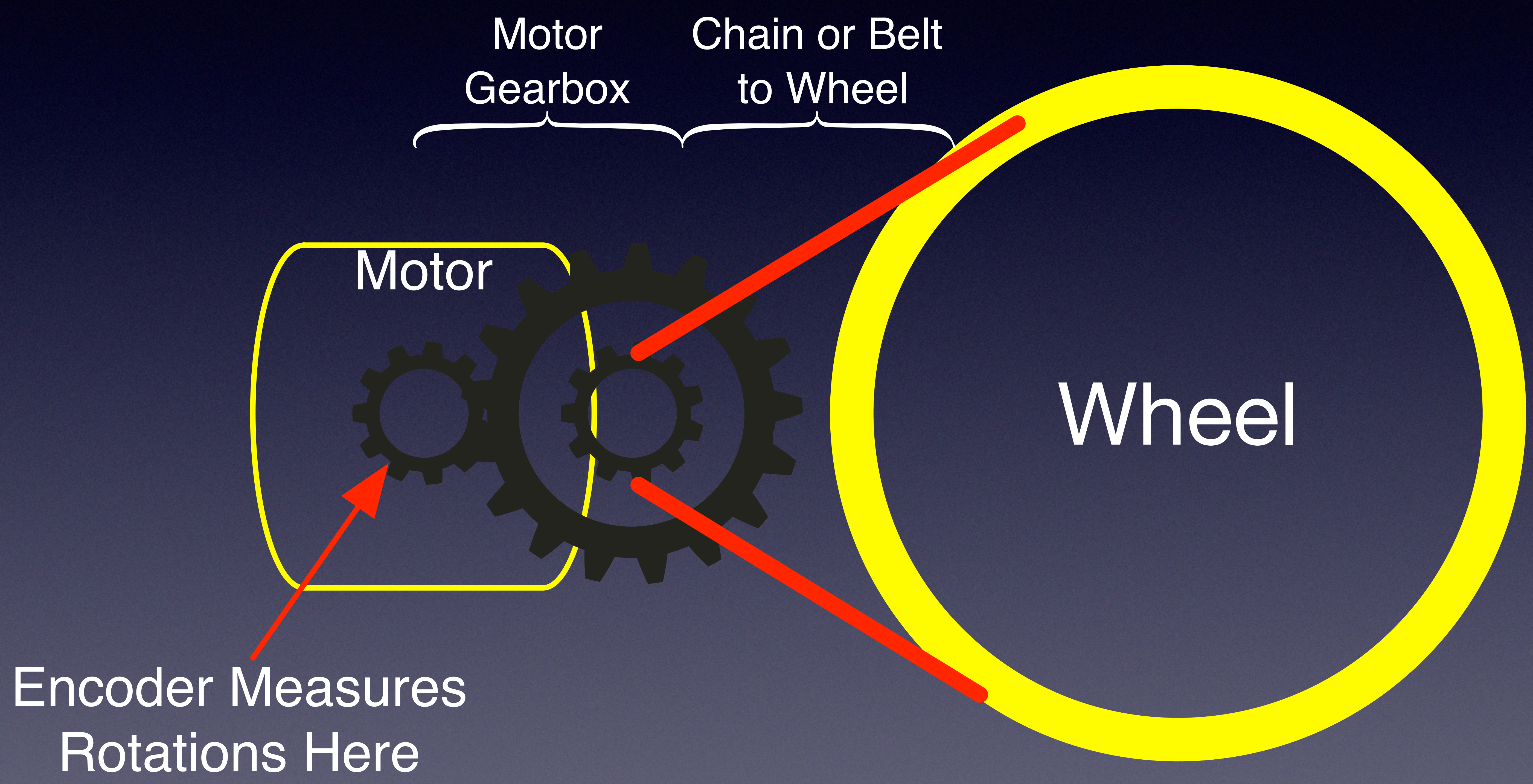
Table 2: HD Hex Motor (REV-41-1301) Encoder Specifications

HD Hex Motor (REV-41-1301) Reduction	40:1	20:1
Free Speed (RPM)	150	300
Cycles per Rotation of the Encoder Shaft	28 (7 Rises of Channel A)	28 (7 Rises of Channel A)
Counts per Rotation of the Output Shaft	1120 (280 Rises of Channel A)	560 (140 Rises of Channel A)

- Read the Specifications!
- Rev conveniently does part of the math for you.
- For example, each full rotation of a 40:1 motor is 1120 pulses.
- These pulses are the same values you will see in software.

So… how do we know how far our robot has driven?

Motor Encoders - more math!



- Consider:
  - 40:1 Rev Motor
    - (1120 pulses per output shaft rotation)
  - 2:1 reduction in chain
    - (1120 * 2 = 2240 pulses per wheel rotation)
  - 3" wheel diameter
    - (Pi * 3.0 = distance per rotation)
    - 9.42" circumference

Pulses/Inch = 2240 PPR / 9.42" = 237

How many pulses would you measure if robot moved 5"?

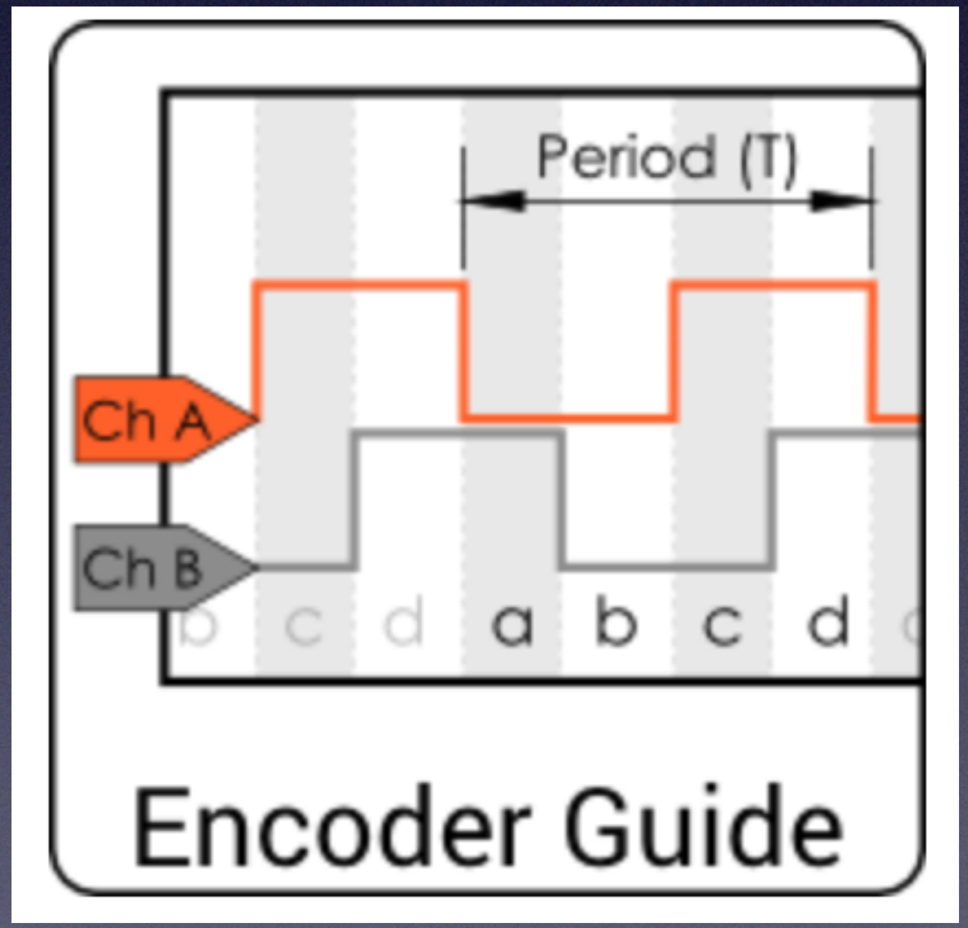
Encoder Sensors are Special

- The Rev hub can automatically control motors using the corresponding encoders.
- Software can specify the distance, speed, or power and the expansion hub will control the motor for you, using data from the encoder.
  - This is sometimes called “encoder drive”

The Rev Expansion Hub

The Rev Guides

Go to: http://www.revrobotics.com/resources/



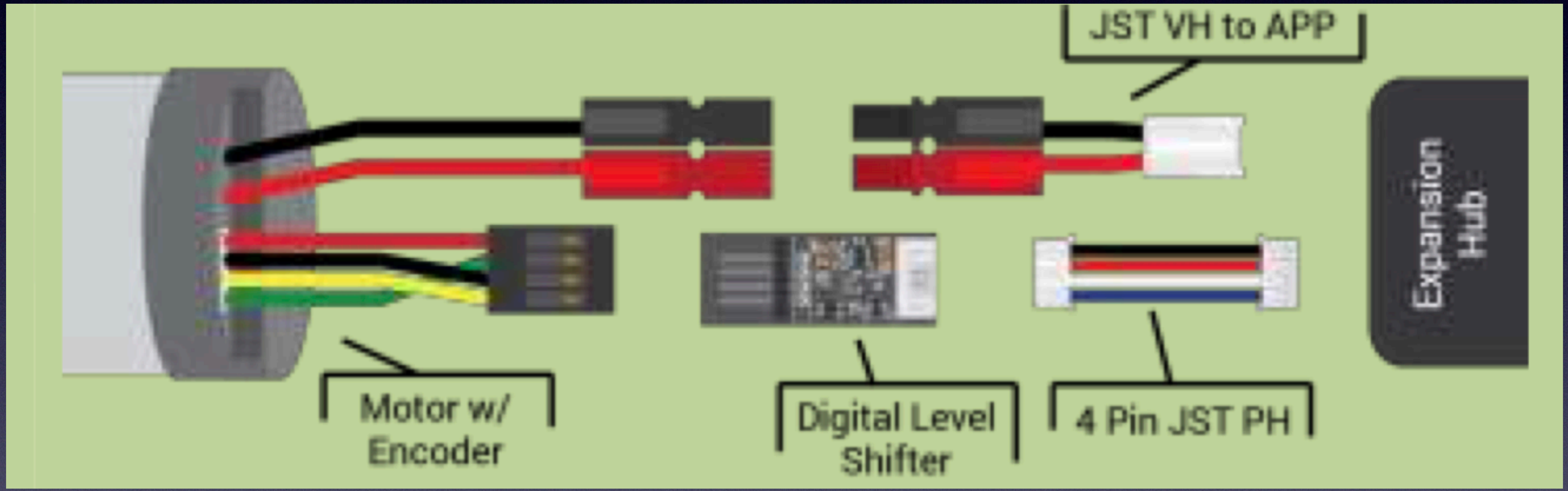
Sensor Connections (Rev)

- If you use all Rev motors and sensors, hookup is easy.
  - Use supplied JST cables or fabricate equivalent ones
  - Connect to available port based on sensor type
  - The accelerometer/gyro is built into your expansion hub, no connections needed!

Touch Sensor	Potenti-ometer	Distance Sensor	Magnetic Switch	Color Sensor	Encoder
					
Digital Port	Analog Port	I2C Port	Digital Port	I2C Port	Motor Encoder Port
1 or 0	0.0 to 1.0 (double)	Distance in cm	1 or 0	R,G,B (double)	32-bit integer

Andymark Motor Encoders

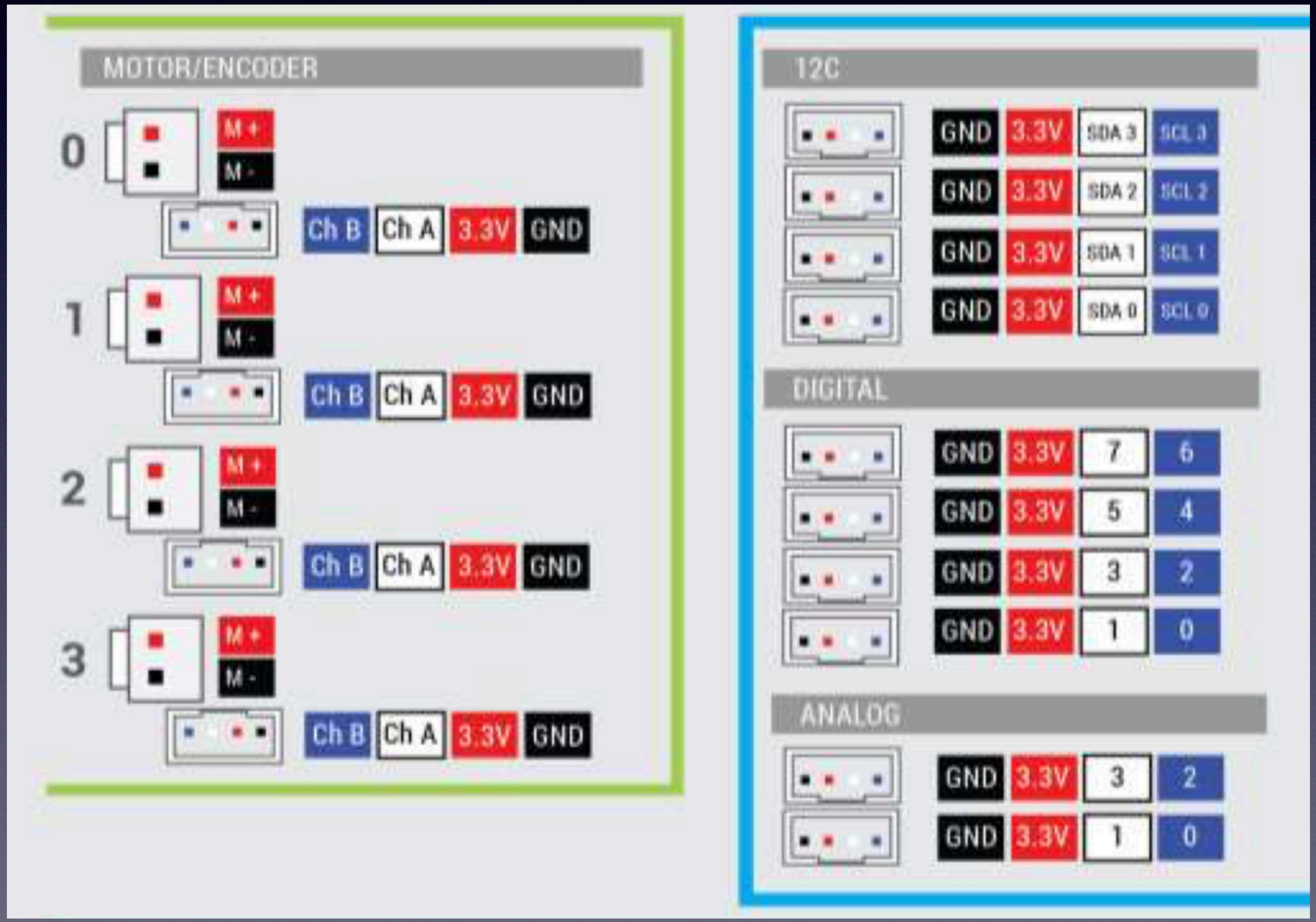
(not needed if you use Rev motors)



- Rev level shifter must be installed (REV-31-1389)
- Andymark encoders are 5V, Expansion hub is 3.3V

Using Non-Rev Sensors

- Diagram colors correspond to wire colors on standard Rev cables
- Cut a Rev sensor wire to attach non-Rev sensors
- *DO NOT SHORT 3.3V+GND TOGETHER! IT CAN DESTROY YOUR EXPANSION HUB!*



Mechanical Switches

Connect ‘NO’ to either white or blue wire

Connect ‘COM’ to black wire

Standard 4-pin Rev sensor cable
cut in half
INSULATE THE RED WIRE,
DO NOT ALLOW IT TO SHORT

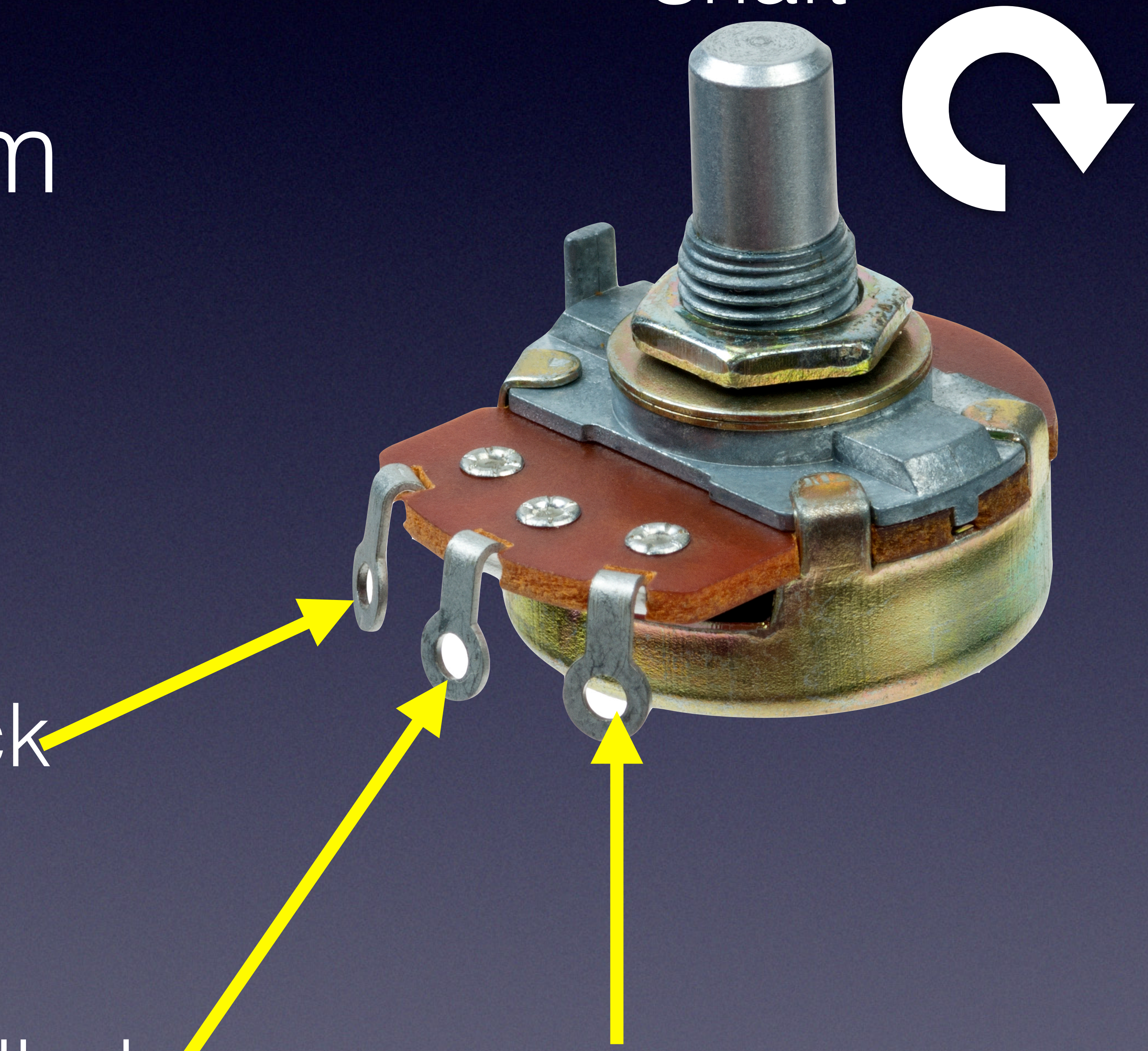
Sensor will read 0 if switch is pressed, 1 if switch is released

Potentiometers

Also known as : “Variable Resistors”

- Good for measuring the angle of a mechanism
- Shaft usually turns about 270 degrees

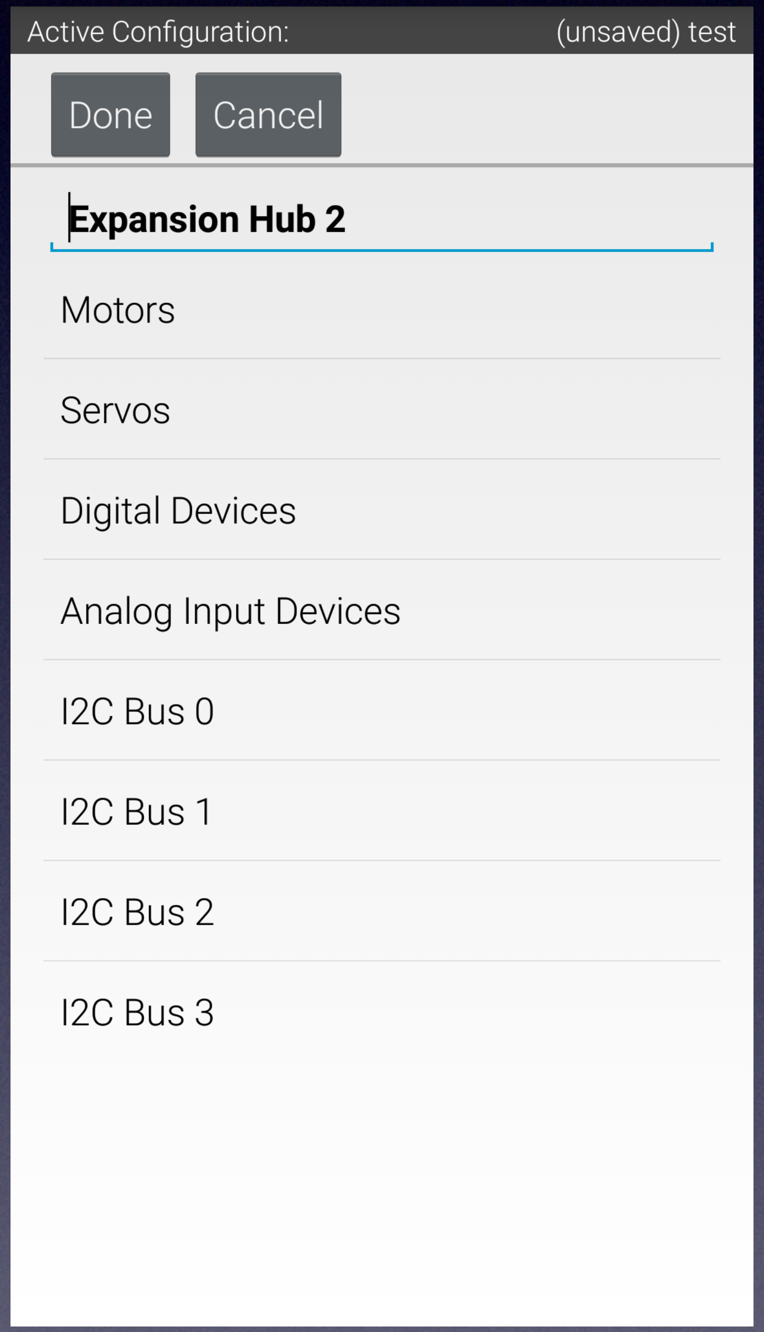
Standard 4-pin Rev sensor cable cut in half



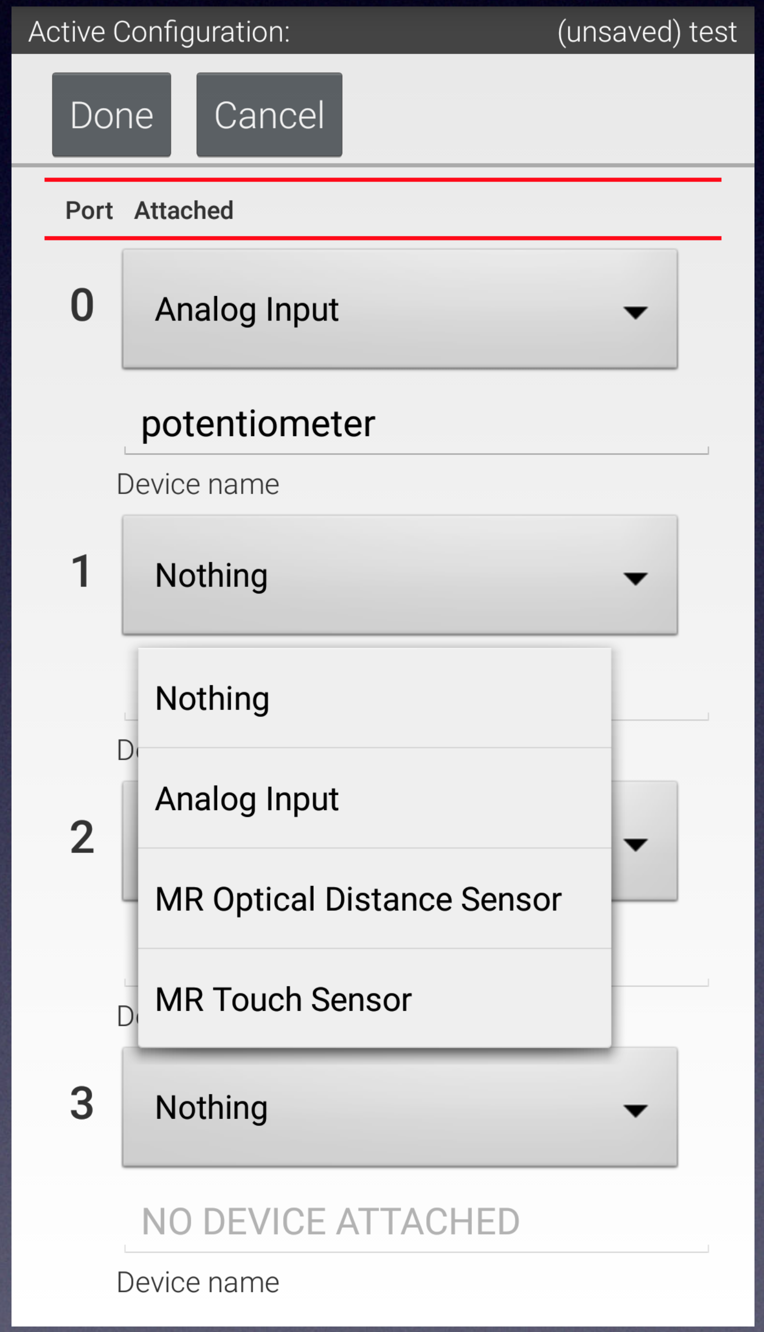
I2C Sensors

- Rev Hub supports only 3.3v Sensors
  - However, all Rev-branded sensors are 3.3v!
  - Use a Rev level shifter module if you use a non-Rev sensor
- I2C Sensors have *addresses*. This means there's a unique number that identifies a specific sensor attached to an expansion port
  - This address will appear in your software.

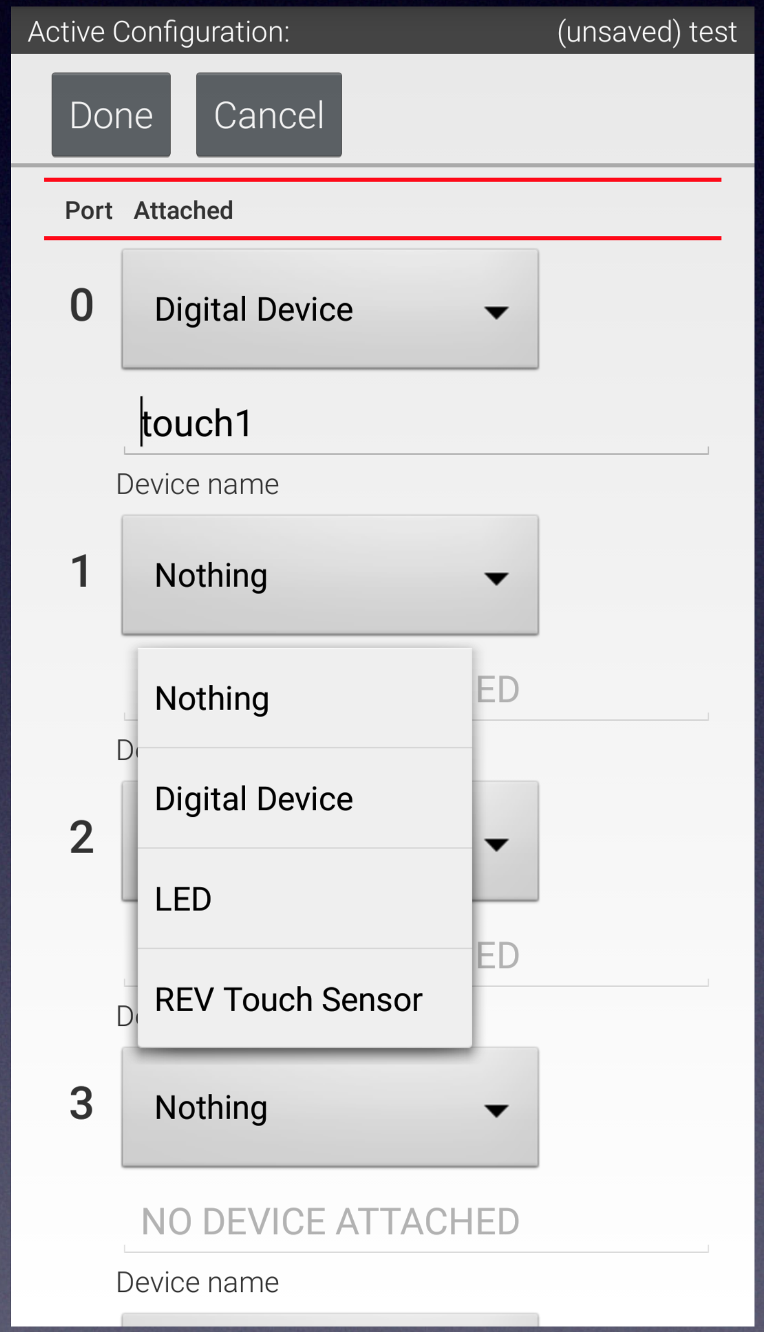
Configuring Sensors on RC



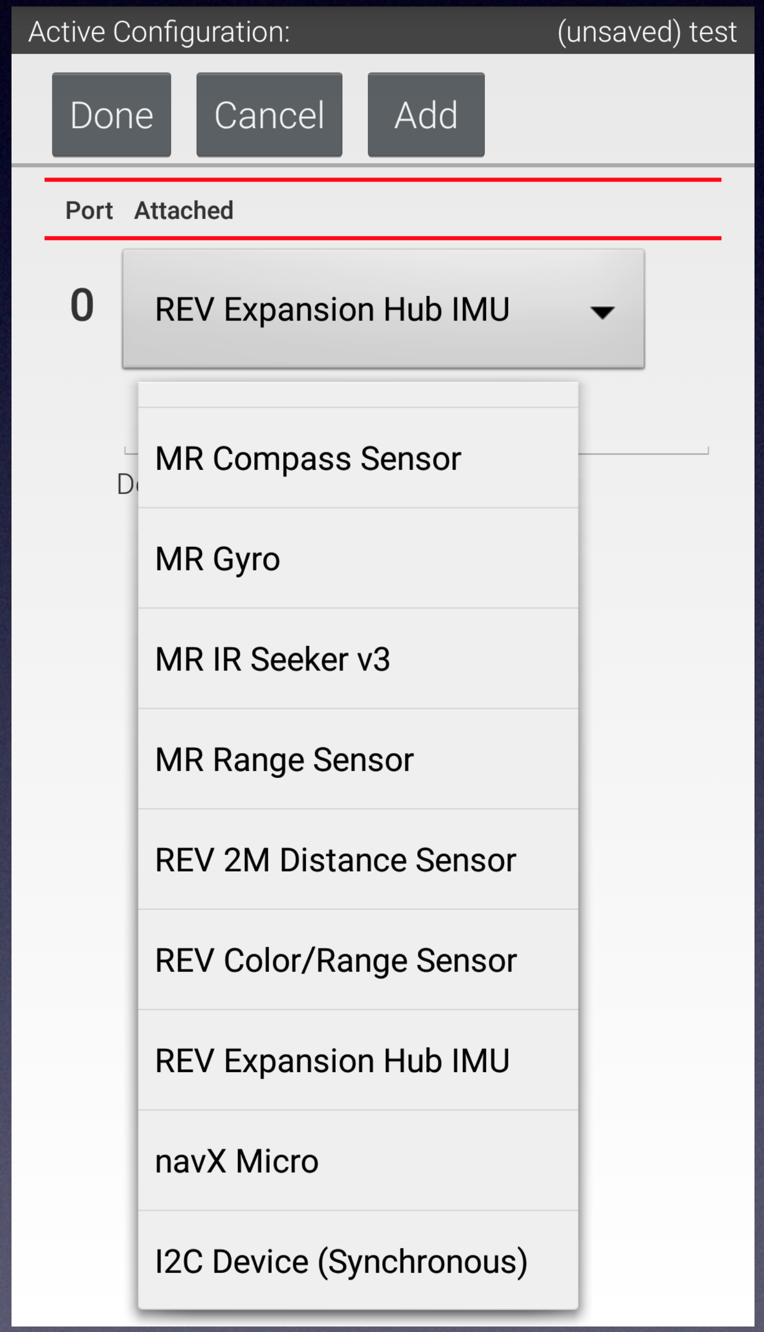
Sensor Interfaces



Analog Sensors



Digital Sensors

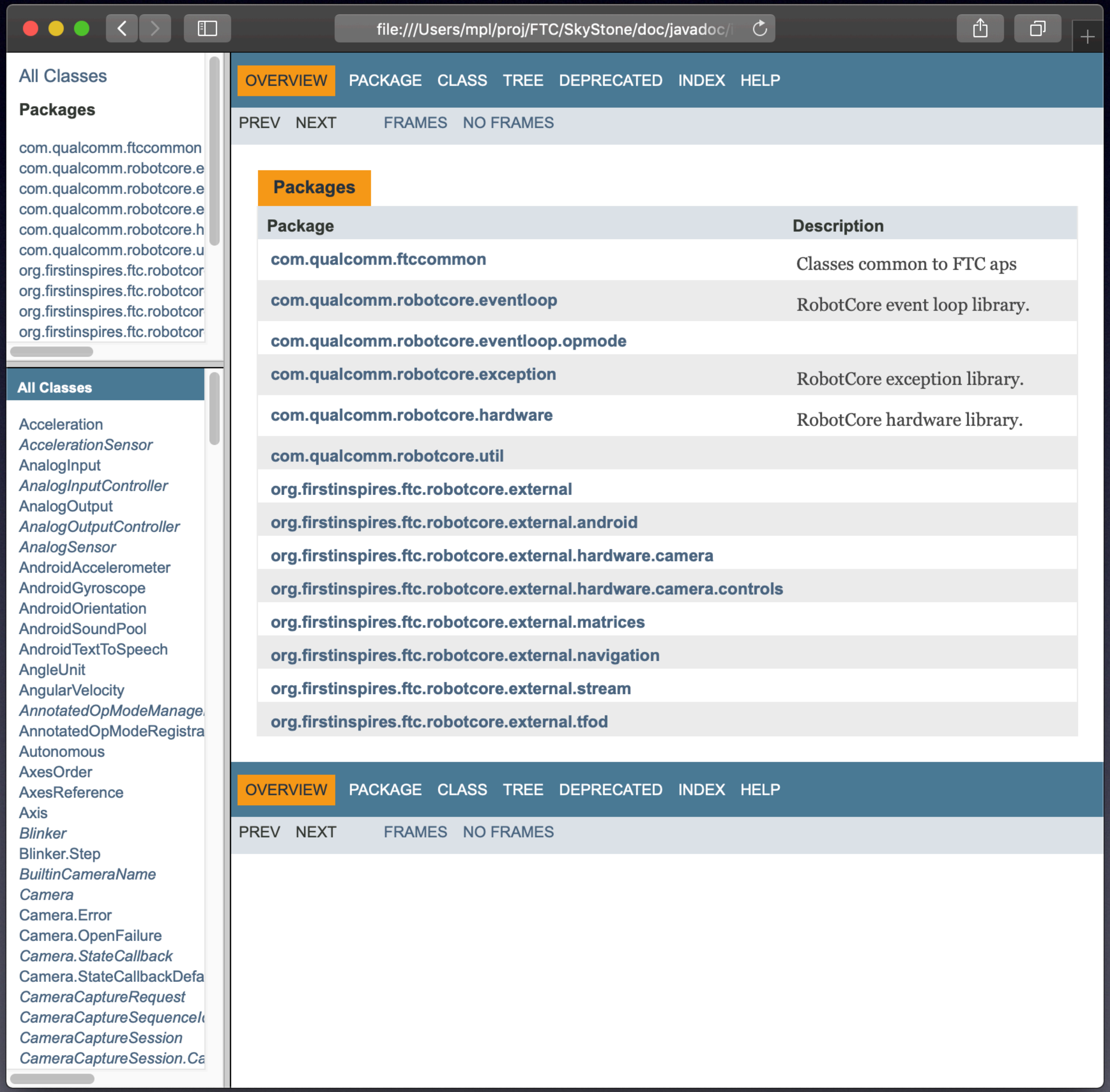


I2C Sensors

Adding Sensors to Software

- Each sensor type has a unique Java class
  - See FTC SDK documentation for a complete list!
- Create the object by looking up sensor in the *hardware map*.
- Later in your code, use this object to read the sensor.
  - Methods for each sensor type are also in the SDK documentation

Read the docs!



- Documentation is in your SDK
- Open *doc/javadoc/index.html* in your web browser

Java Classes for Sensors

	Rev Touch	Rev Mag Limit	Rev IMU	Potenti-ometer	Rev 2M Distance	Rev Color V2
Java Import	rev.RevTouchSensor	rev.RevTouchSensor	bosch.BNO055IMU	hardware.AnalogInput	rev.Rev2mDistanceSensor	hardware.AnalogInput
Java Class	TouchSensor	TouchSensor	BNO055IMU	AnalogInput	Rev2mDistance Sensor	ColorSensor
Expansion Port	Digital	Digital	I2C	Analog	I2C	I2C
Sensor Type	Rev Touch Sensor	Rev Touch Sensor	Rev Expansion Hub IMU	Analog Input	Rev 2M Distance Sensor	Rev Color/Range Sensor

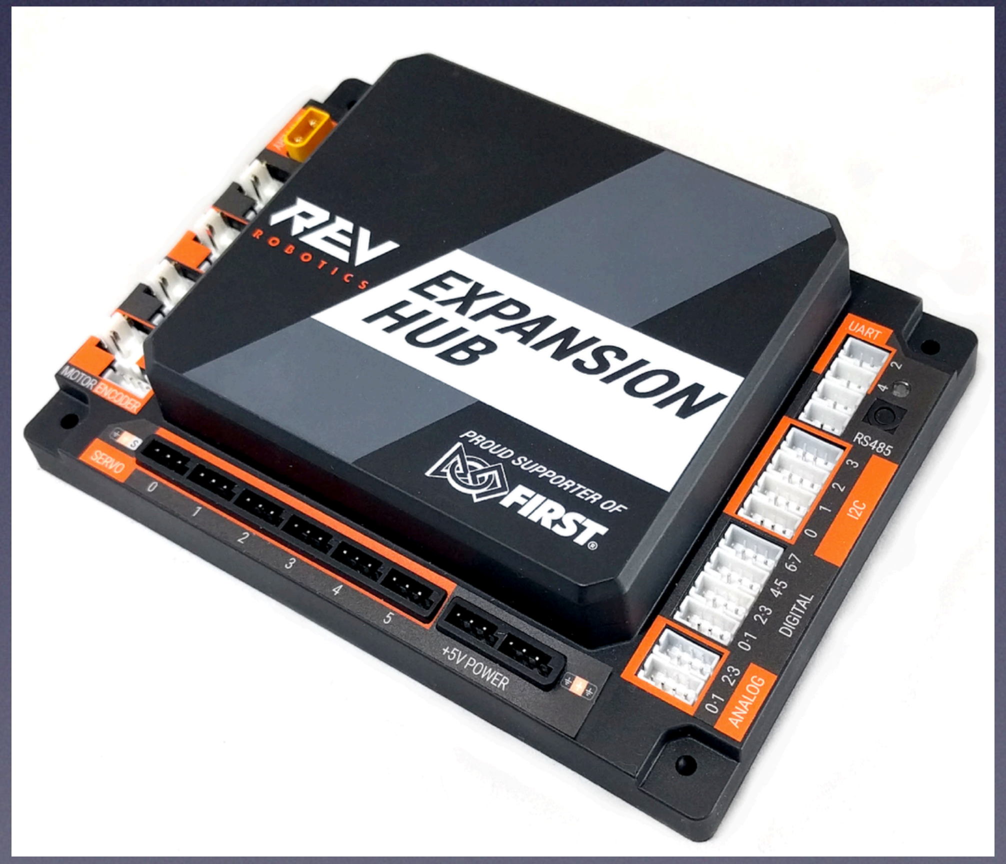
Four Steps

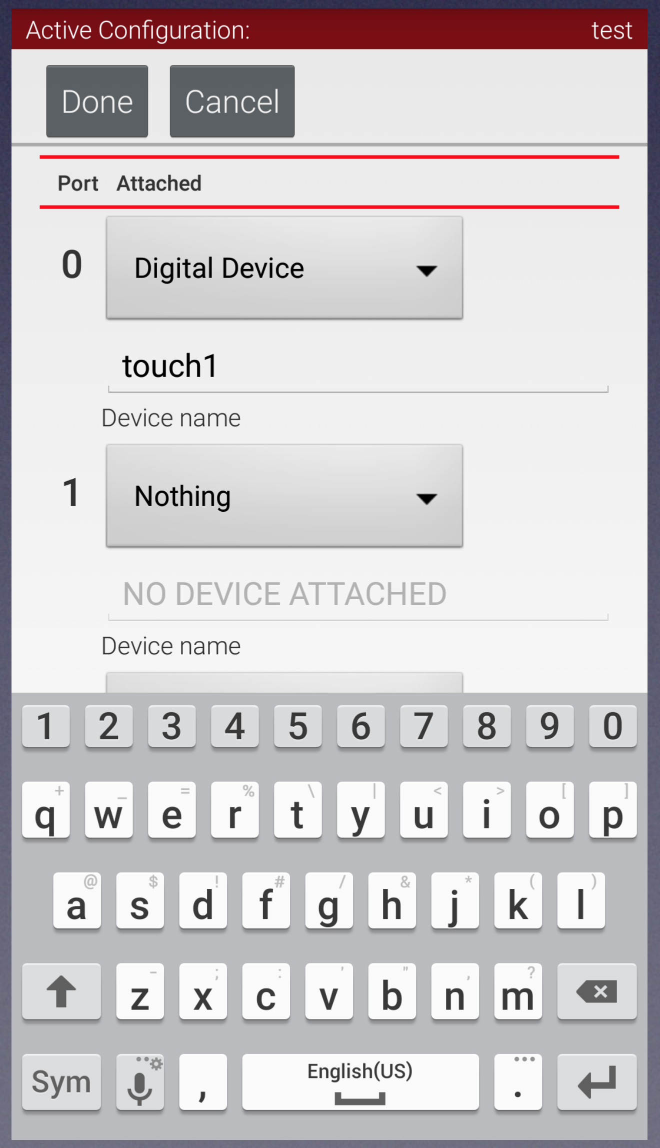
Connect the sensor to your robot's expansion hub

Name your sensor in the robot controller phone

In your *robot init* code, create the sensor object using the same name

In teleop or autonomous code, use the object to access the sensor





Robot Init Code

```
private TouchSensor touchSensor;

touchSensor = hardwareMap.get(TouchSensor.class, "touch1");
```

Declare sensor object
Create object from hardware map

Teleop/Autonomous

```
boolean touchValue = touchSensor.isPressed();
```

Access sensor data

Sensor Noise

Problems You May See

- Sensors are *not* perfect!
    - Occasionally you may get readings that make no sense
        - Color sensors occasionally return unusual values
        - Gyros suffer from *drift*.
        - Distance sensors may seem out of range
    - Even touch sensors *bounce* while the mechanical switch settles.
    - Exception: Encoders are usually noise-free

Things to Try

- For analog sensors, average several recent readings
- Discard obvious outliers.
- For digital sensors, read until you get the same value a few times in a row

Absolute vs Relative Sensors

Absolute

- Returns a specific reading based on the current physical world (will generally be the same reading after robot is restarted)
  - Distance Sensor
  - Color Sensor
  - Touch Sensor
  - Potentiometer

Relative

- Returns a value representing a change from a previous reading
  - Encoder. You don't really know where the robot was at the previous reading. FTC robots typically want to know the distance from some previous reading.
  - Gyro. Although it returns the absolute compass heading, this information is not very useful. FTC robots want to know the *changes* in the heading.

Sensor Characterization

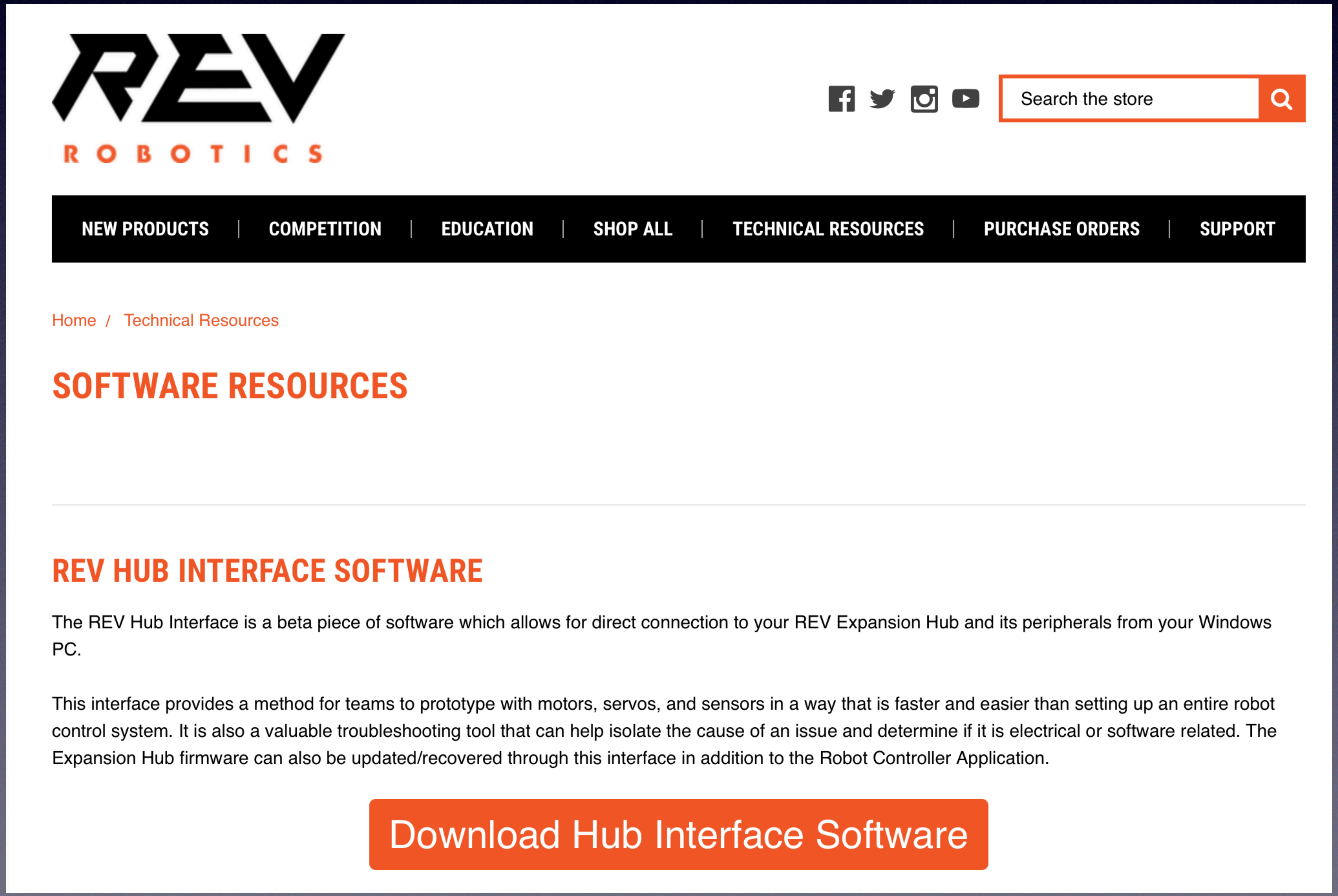
Test your sensors!

- Make a test program
  - Log sensor data to a file
  - Write sensor data to telemetry
  - Test sensors, particularly analog, color, distance, gyro
  - Understand the measurements.
  - Observe and filter noise

Touch Sensor	Potenti-ometer	Distance Sensor	Magnetic Switch	Color Sensor	IMU
					
Switch Bounce	Noise, out of range	Noise, out of range	Switch Bounce	Noise from External lights	Compass drift
Read multiple times	Average recent readings	Average recent readings	Read multiple times	Pre-measure targets	Initialize Sensor properly

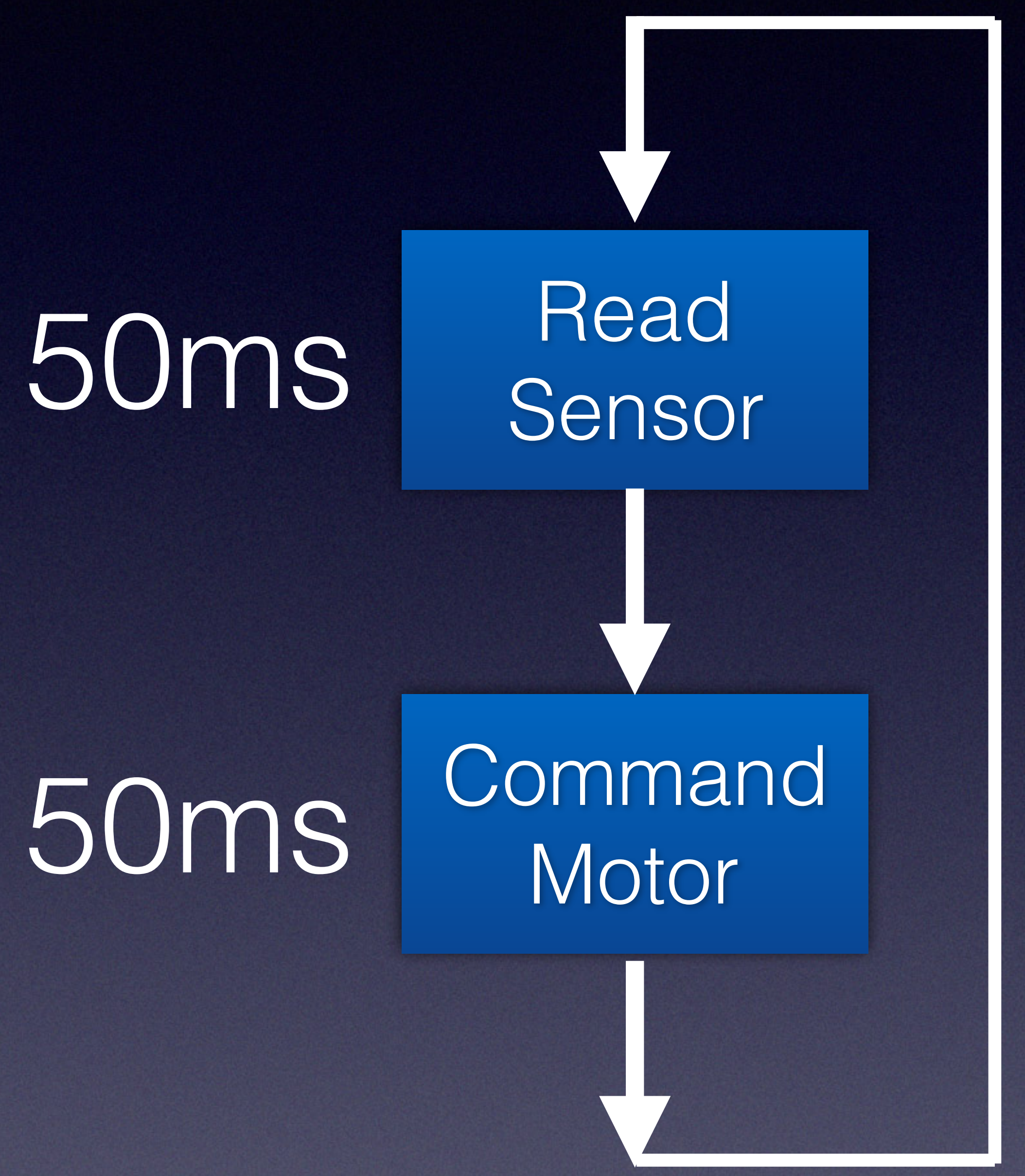
Rev Hub Interface

Test your sensors without a robot controller, using a Windows PC



Sensor Latency

- Sensor readings are *not* instantaneous!
- A sensor doesn't tell you what is happening now, it tells you what happened in the recent past.
- Reading sensors in a tight loop will slow other things down
- Taking multiple measurements slows things down further
- I2C sensors are the slowest to read, including the gyro



* examples for illustration, not real latencies

If a wheel is turning at 20RPM (0.33/sec), it completes a rotation every 330ms.
By the time you can stop the motor, the wheel has made 1/3 extra turn!

Special Consideration: Gyro

Import the Rev gyro as a Bosch BNO055IMU

```
import com.qualcomm.hardware.bosch.BNO055IMU;
```

You must initialize this sensor!

```
imu = hardwareMap.get(BNO055IMU.class, "imu");

BNO055IMU.Parameters parameters = new BNO055IMU.Parameters();
parameters.angleUnit           = BNO055IMU.AngleUnit.DEGREES;
parameters.accelUnit           = BNO055IMU.AccelUnit.METERS_PERSEC_PERSEC;
imu.initialize(parameters);
```

The compass heading's method isn't obvious…

```
public double theHeading()
{
    return imu.getAngularOrientation().firstAngle;
}
```

Special Consideration: Encoder

The encoder is actually part of the *DCMotor* class.

The *getPosition()* method reads the current encoder count.
The *setMode()* method enables encoder modes.

DcMotor.RunMode	Purpose
STOP_AND_RESET_ENCODER	Stop motor & reset position to 0
RUN_TO_POSITION	Motor will run at specified power until encoder reaches a given position.
RUN_USING_ENCODER	Motor will attempt to hold a constant speed using the encoder.
RUN_WITHOUT_ENCODER	Motor is run without using the encoder.

Questions?