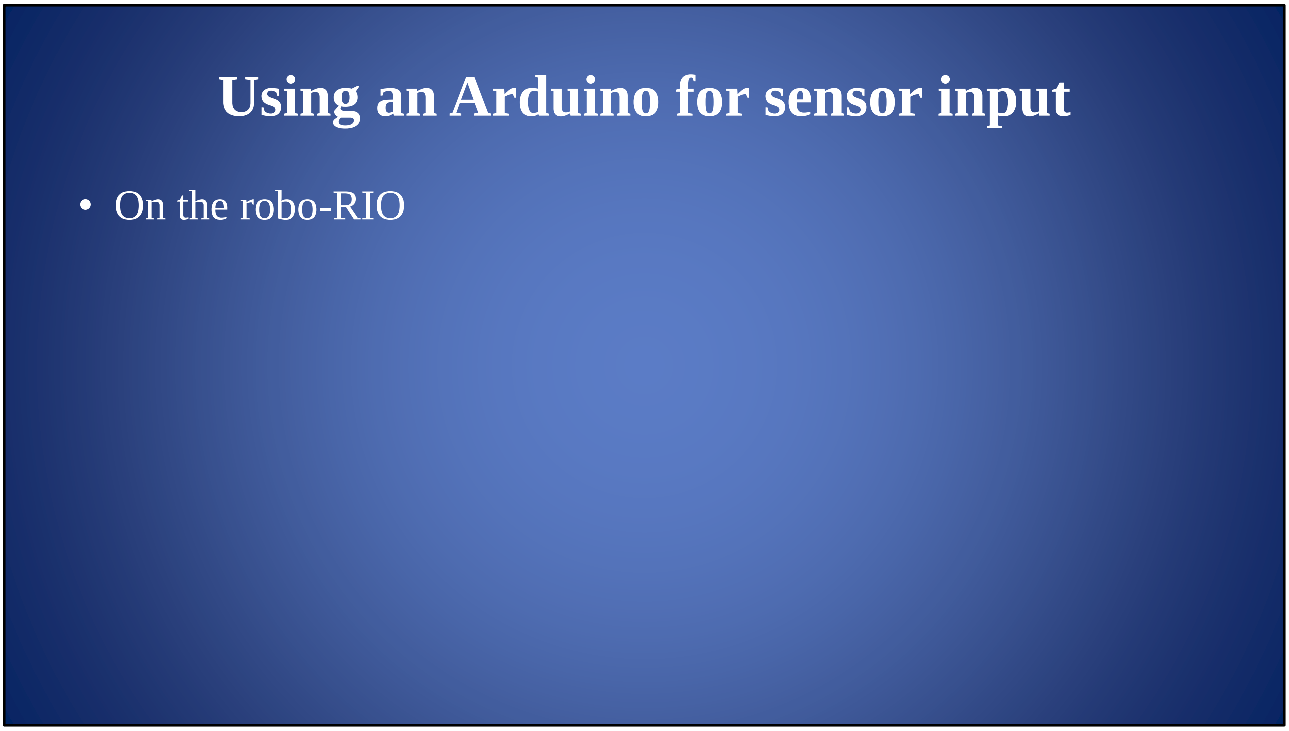
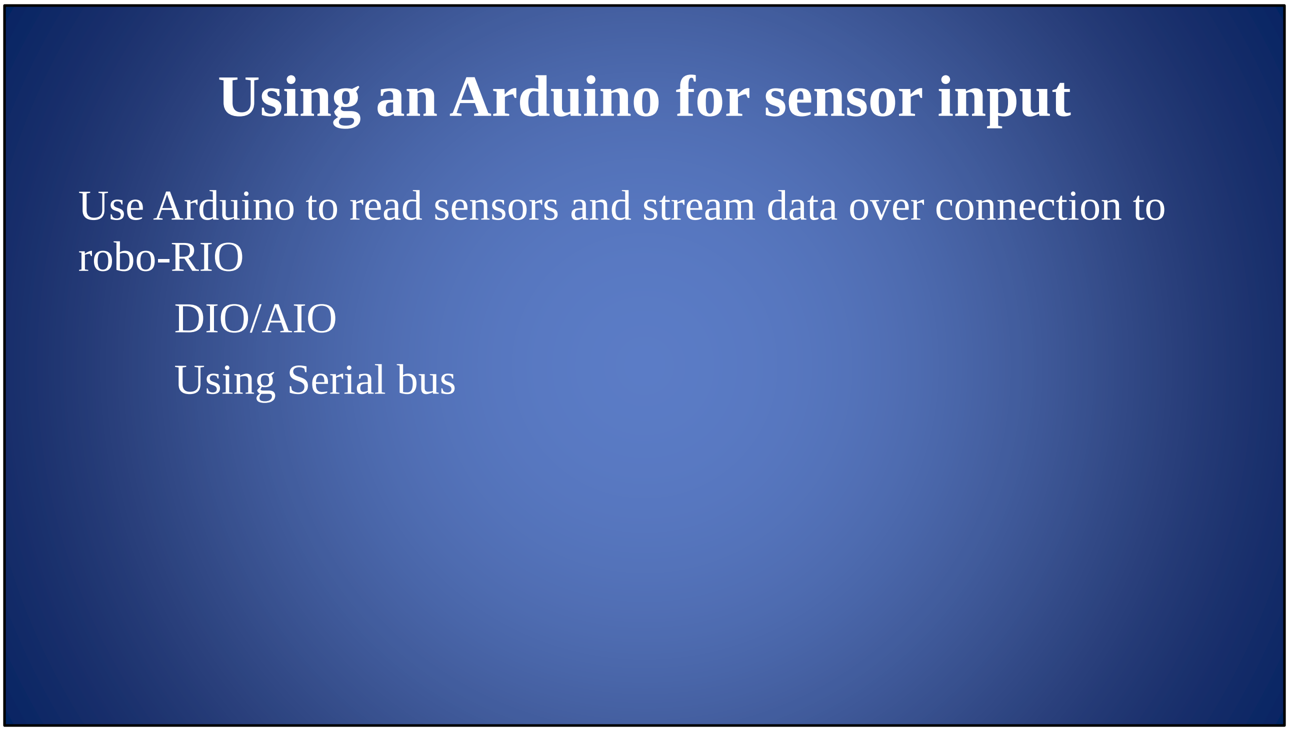
Advanced LabVIEW

frclabviewtutorials.com/workshop



Using an Arduino for sensor input

- On the robo-RIO



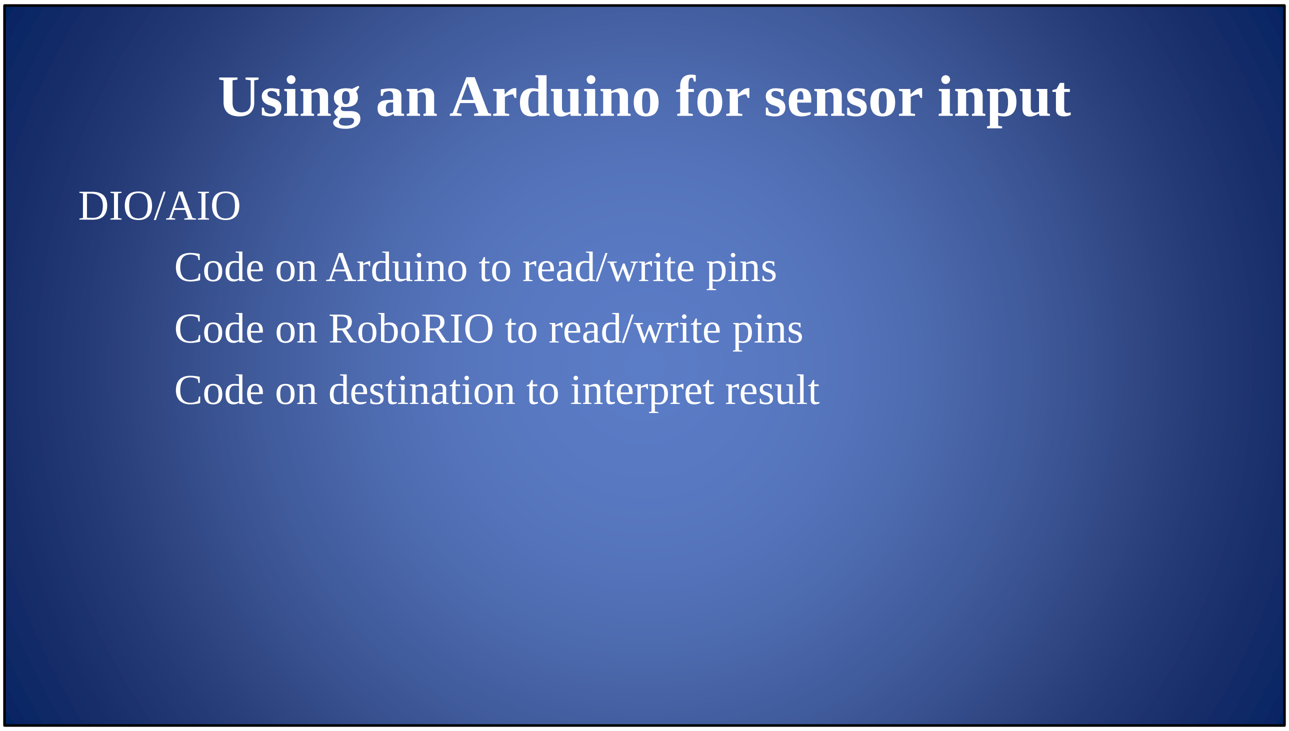
Using an Arduino for sensor input

Use Arduino to read sensors and stream data over connection to robo-RIO

- DIO/AIO
- Using Serial bus

Connecting DIO or AIO lines to and from an Arduino and the RoboRIO can provide a simple interface – useful for a small finite set of states to communicate (i.e., Breakaway LED status in Recycle Rush – 2 DO for type and 1 AO for height).

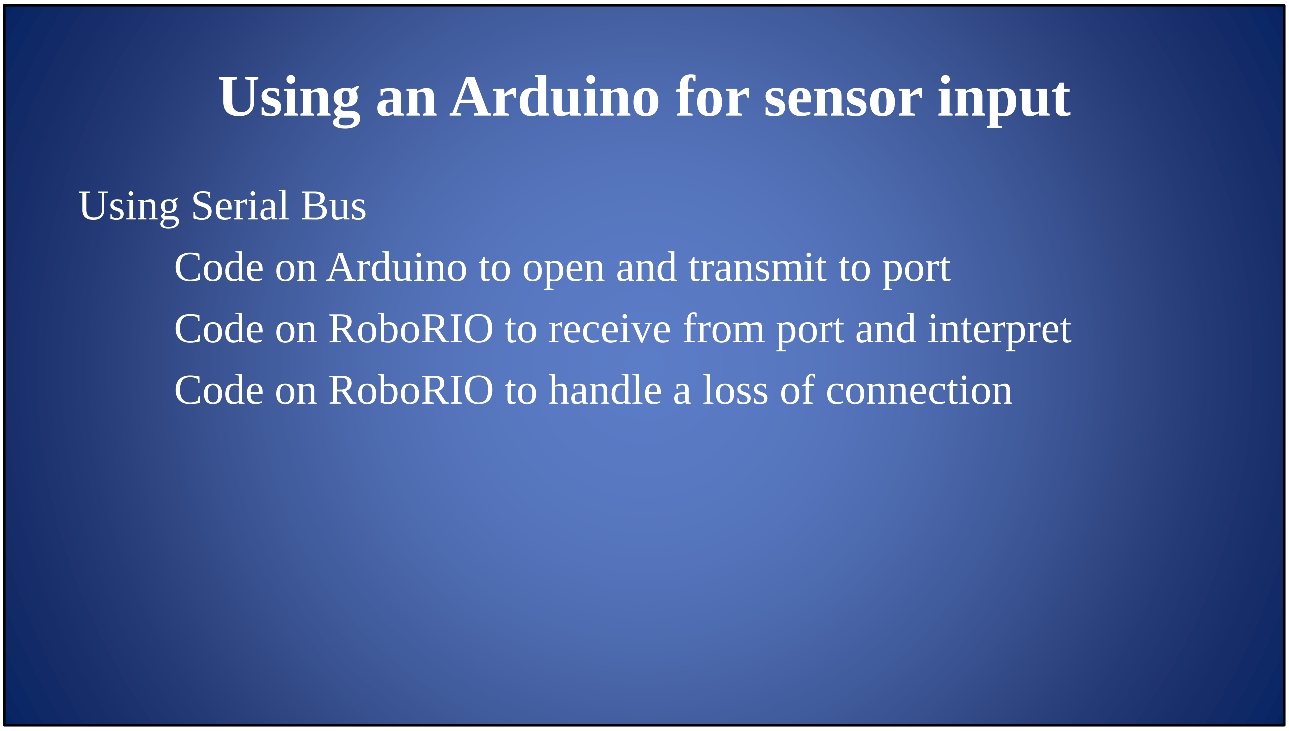
Serial bus is a tad harder to code, but allows for infinite states to be communicated (while only consuming one of the serial ports on the RIO).



Using an Arduino for sensor input

DIO/AIO

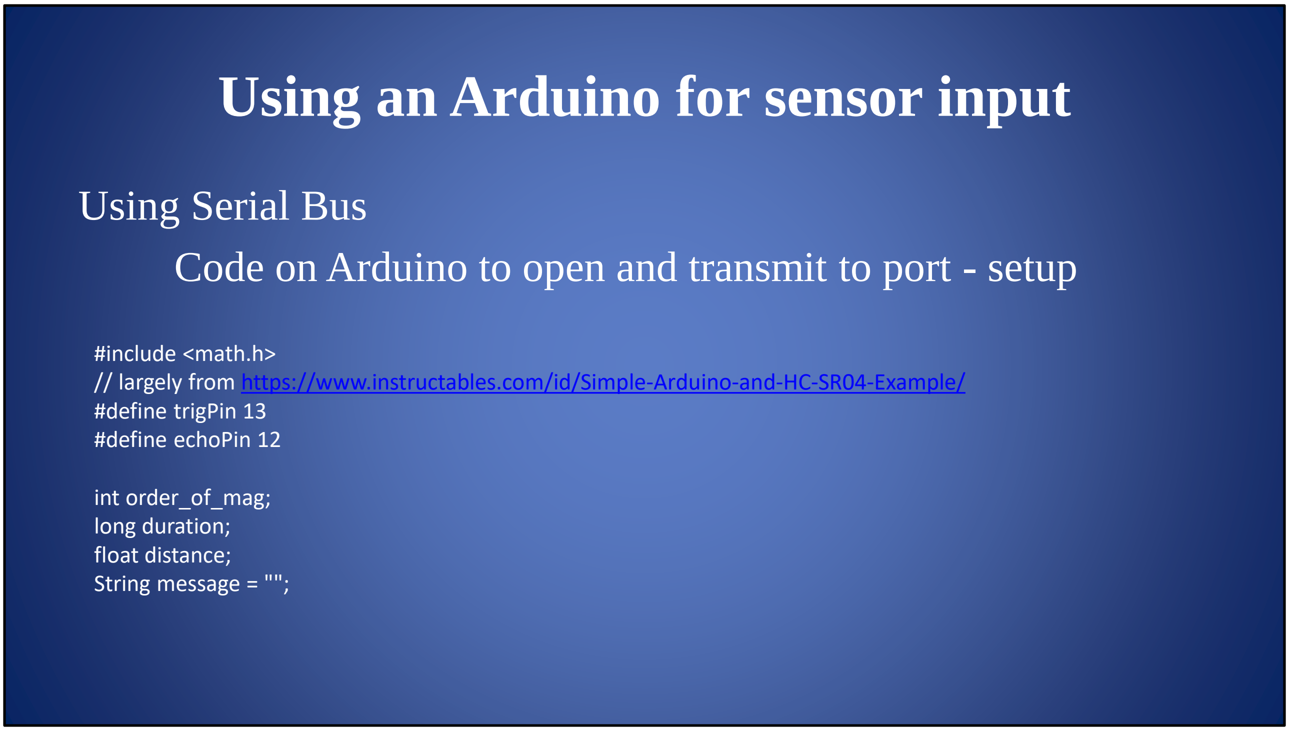
- Code on Arduino to read/write pins
- Code on RoboRIO to read/write pins
- Code on destination to interpret result



Using an Arduino for sensor input

Using Serial Bus

- Code on Arduino to open and transmit to port
- Code on RoboRIO to receive from port and interpret
- Code on RoboRIO to handle a loss of connection



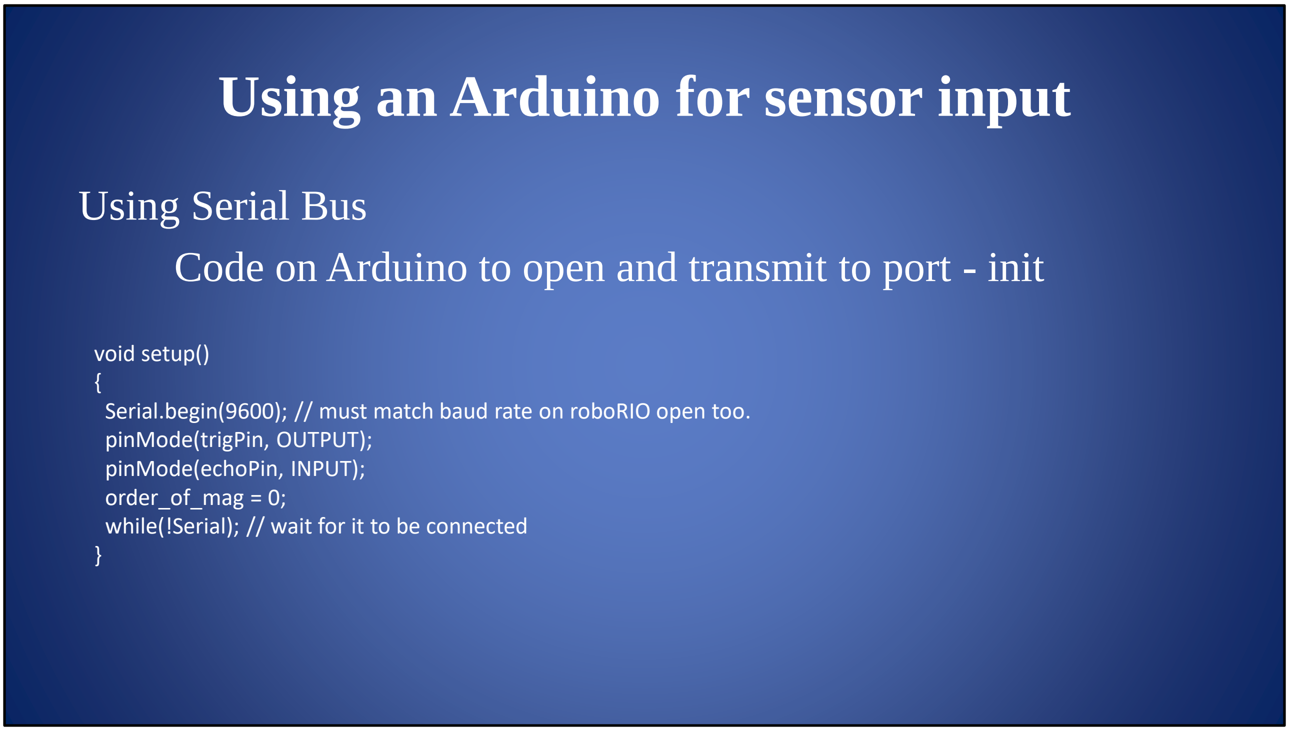
Using an Arduino for sensor input

Using Serial Bus

Code on Arduino to open and transmit to port - setup

```
#include <math.h>
// largely from https://www.instructables.com/id/Simple-Arduino-and-HC-SR04-Example/
#define trigPin 13
#define echoPin 12

int order_of_mag;
long duration;
float distance;
String message = "";
```

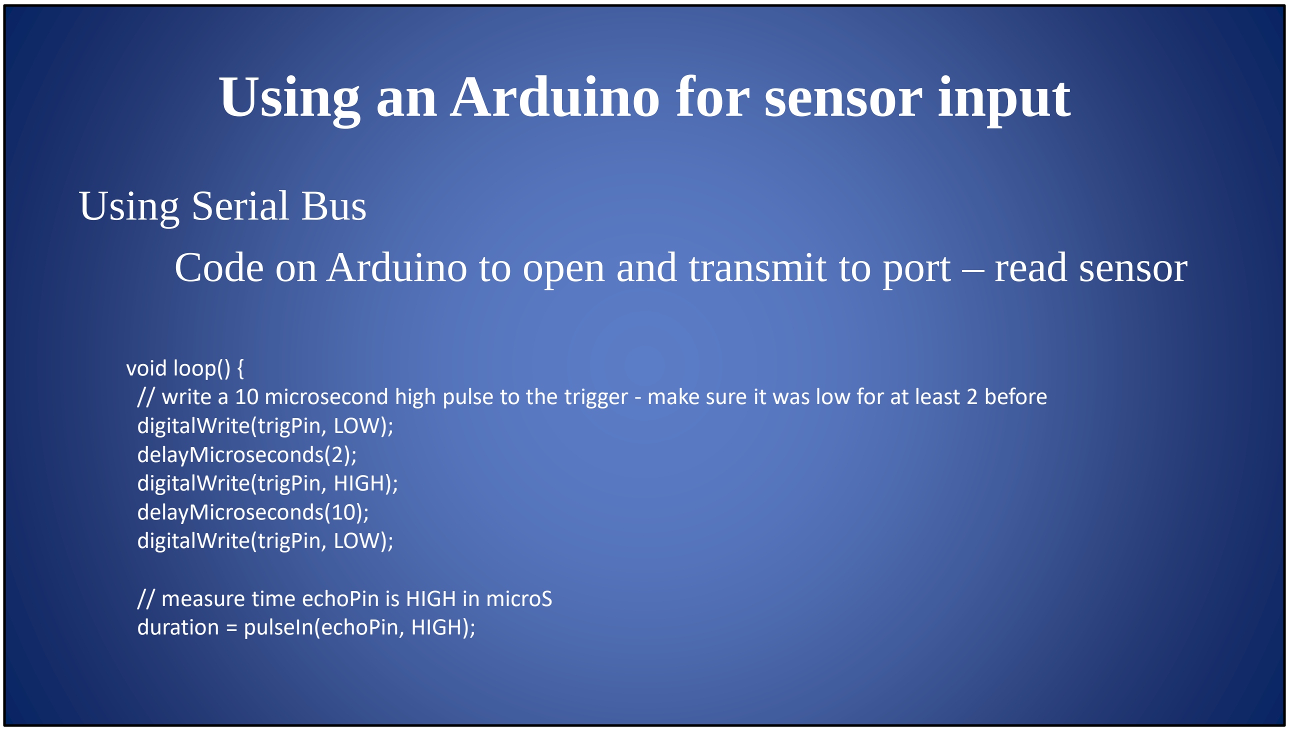


Using an Arduino for sensor input

Using Serial Bus

Code on Arduino to open and transmit to port - init

```
void setup()
{
  Serial.begin(9600); // must match baud rate on roboRIO open too.
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  order_of_mag = 0;
  while(!Serial); // wait for it to be connected
}
```



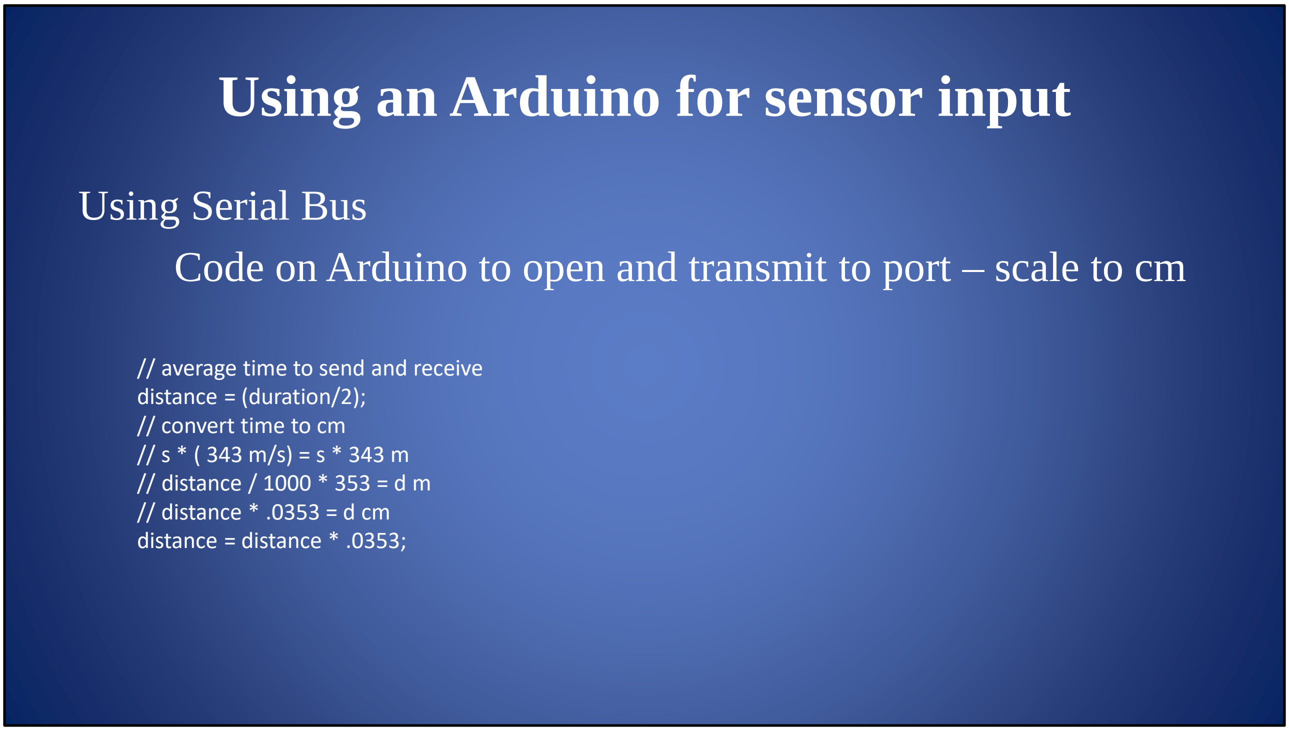
Using an Arduino for sensor input

Using Serial Bus

Code on Arduino to open and transmit to port – read sensor

```
void loop() {
 // write a 10 microsecond high pulse to the trigger - make sure it was low for at least 2 before
 digitalWrite(trigPin, LOW);
 delayMicroseconds(2);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);

 // measure time echoPin is HIGH in microS
 duration = pulseIn(echoPin, HIGH);
```

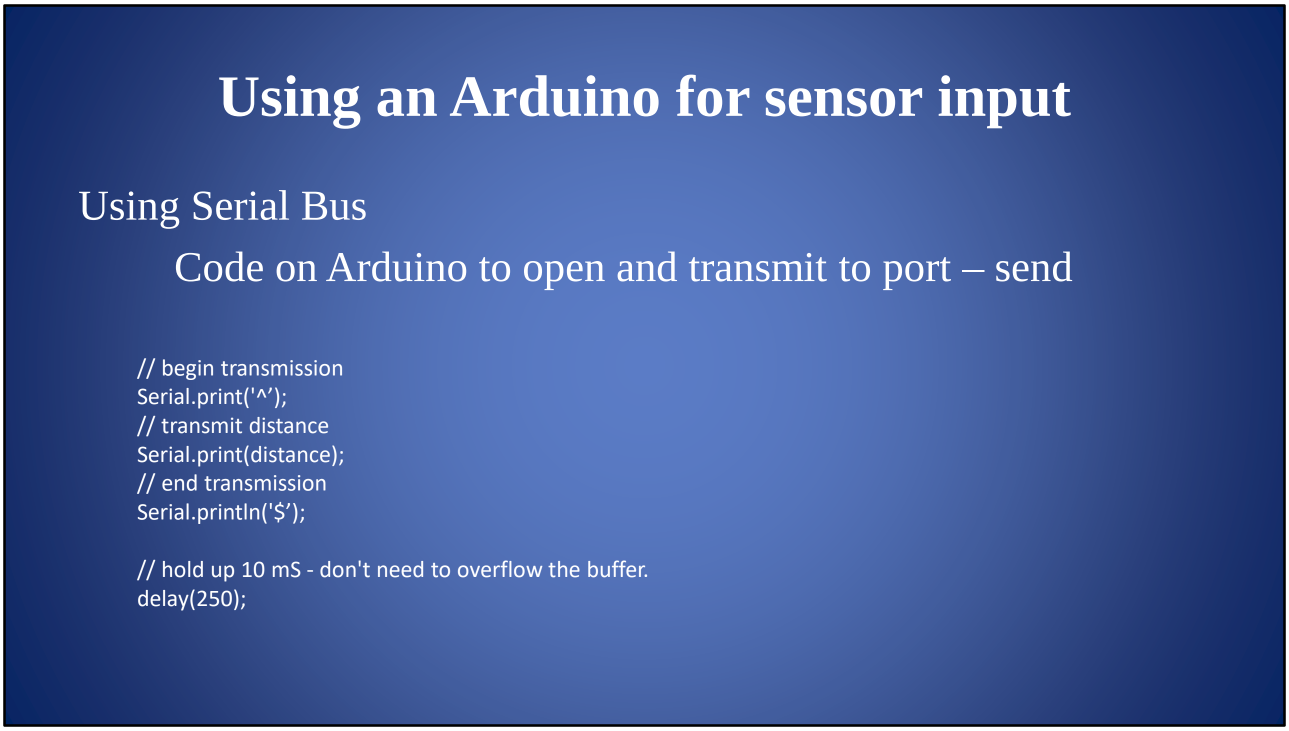


Using an Arduino for sensor input

Using Serial Bus

Code on Arduino to open and transmit to port – scale to cm

```
// average time to send and receive
distance = (duration/2);
// convert time to cm
// s * ( 343 m/s) = s * 343 m
// distance / 1000 * 353 = d m
// distance * .0353 = d cm
distance = distance * .0353;
```



Using an Arduino for sensor input

Using Serial Bus

Code on Arduino to open and transmit to port – send

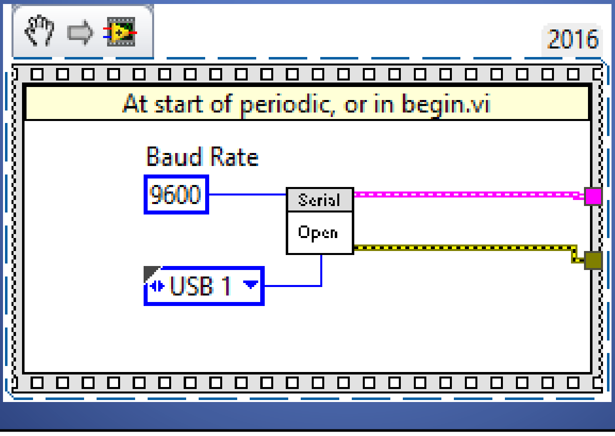
```
// begin transmission
Serial.print('^’);
// transmit distance
Serial.print(distance);
// end transmission
Serial.println('$’);

// hold up 10 mS - don't need to overflow the buffer.
delay(250);
```

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to open port

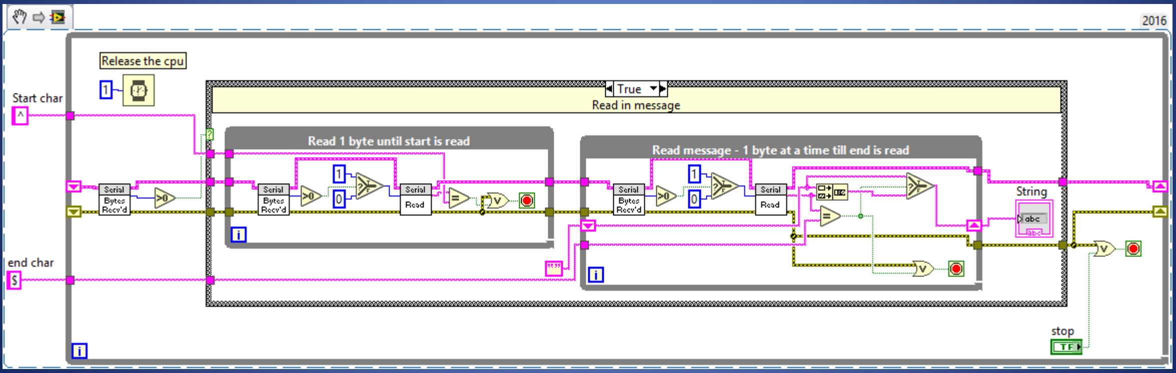


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

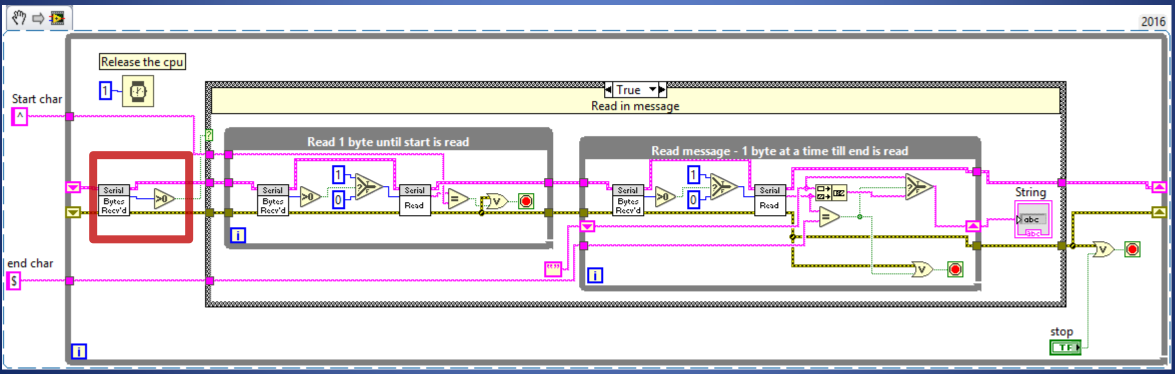


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

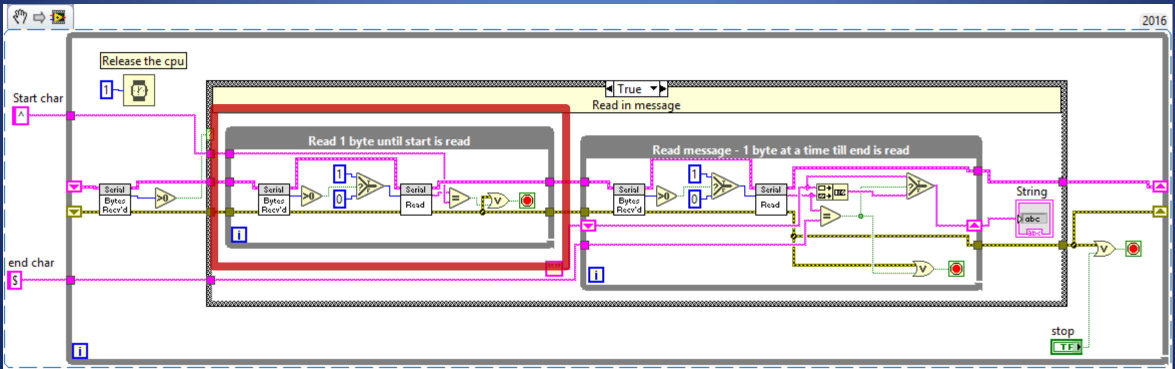


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

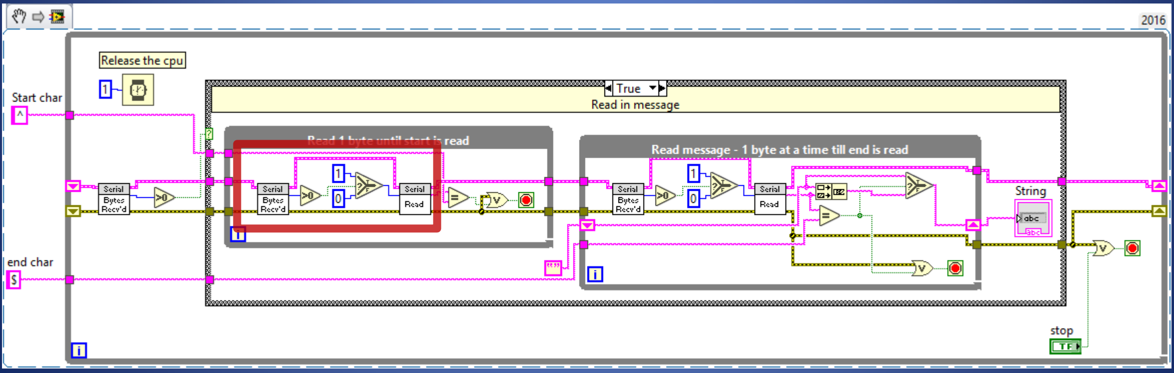


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

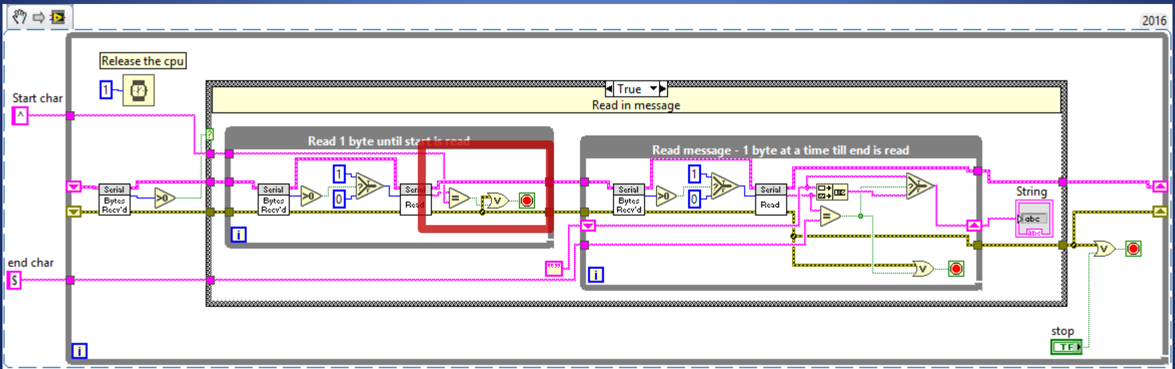


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

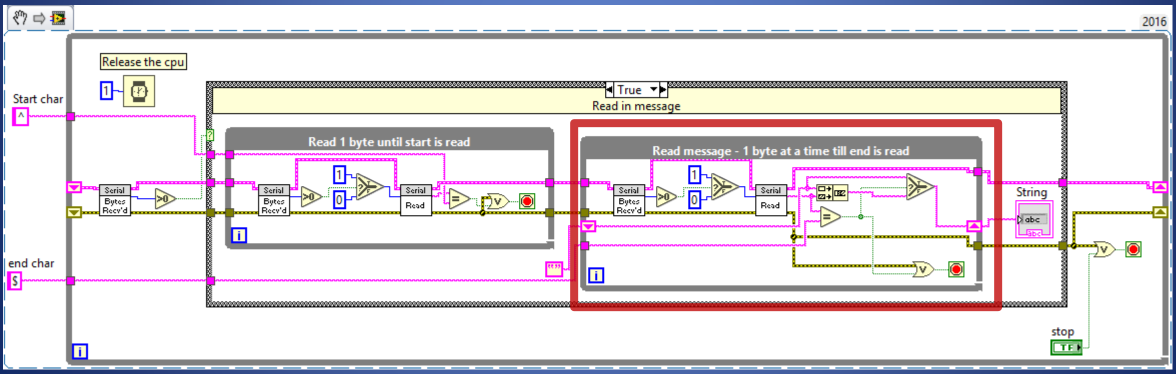


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

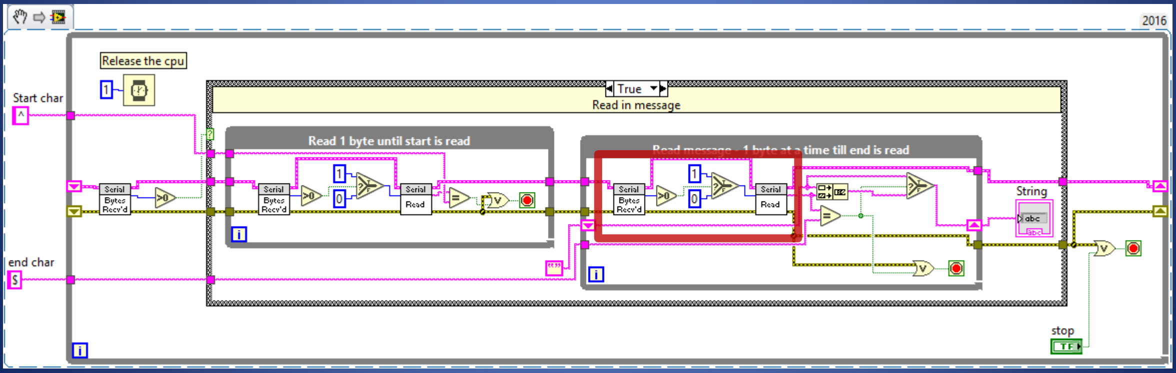


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

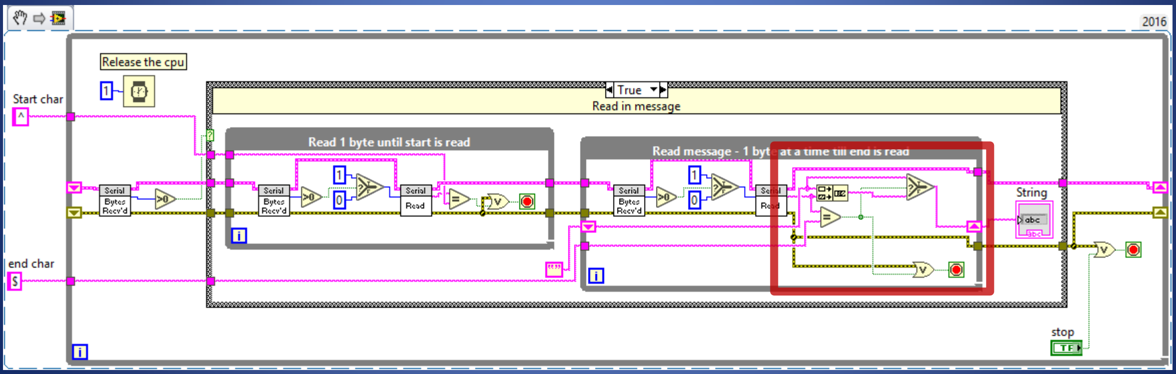


Make sure baud rate matches – select port

Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available

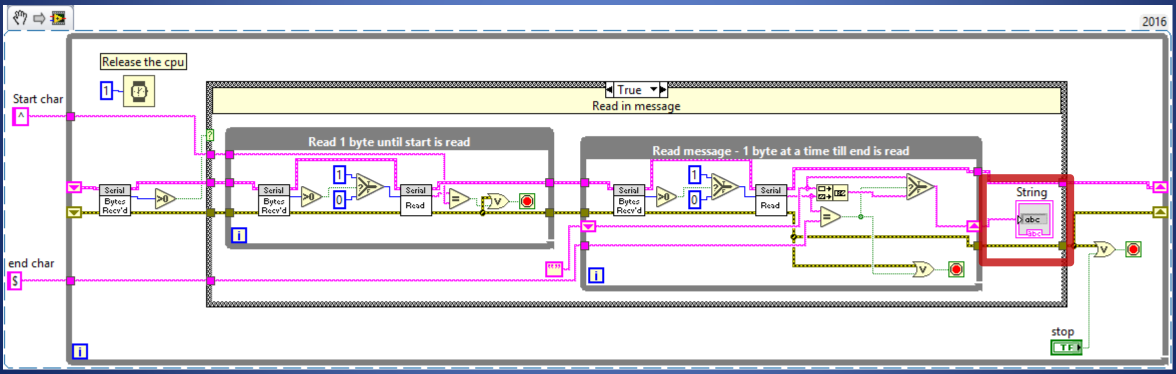


Make sure baud rate matches – select port

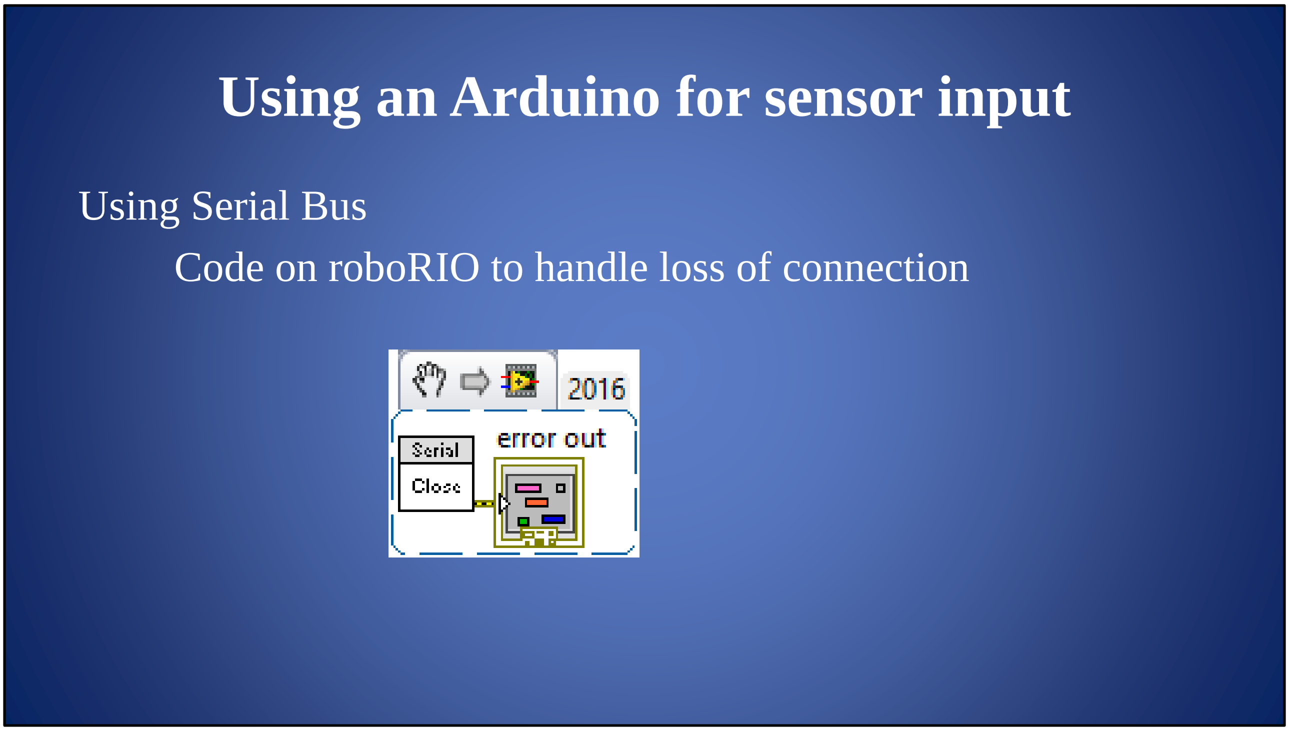
Using an Arduino for sensor input

Using Serial Bus

Code on roboRIO to receive when available



Make sure baud rate matches – select port

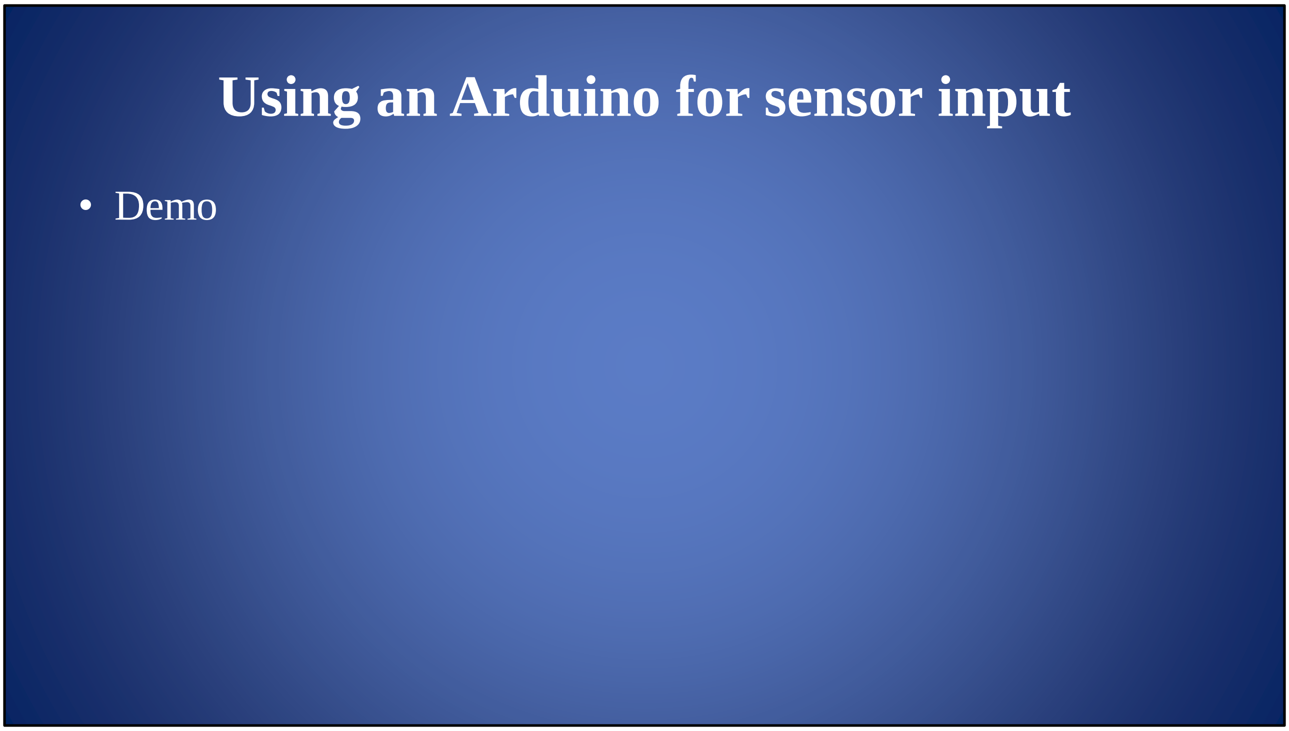


Using an Arduino for sensor input

Using Serial Bus

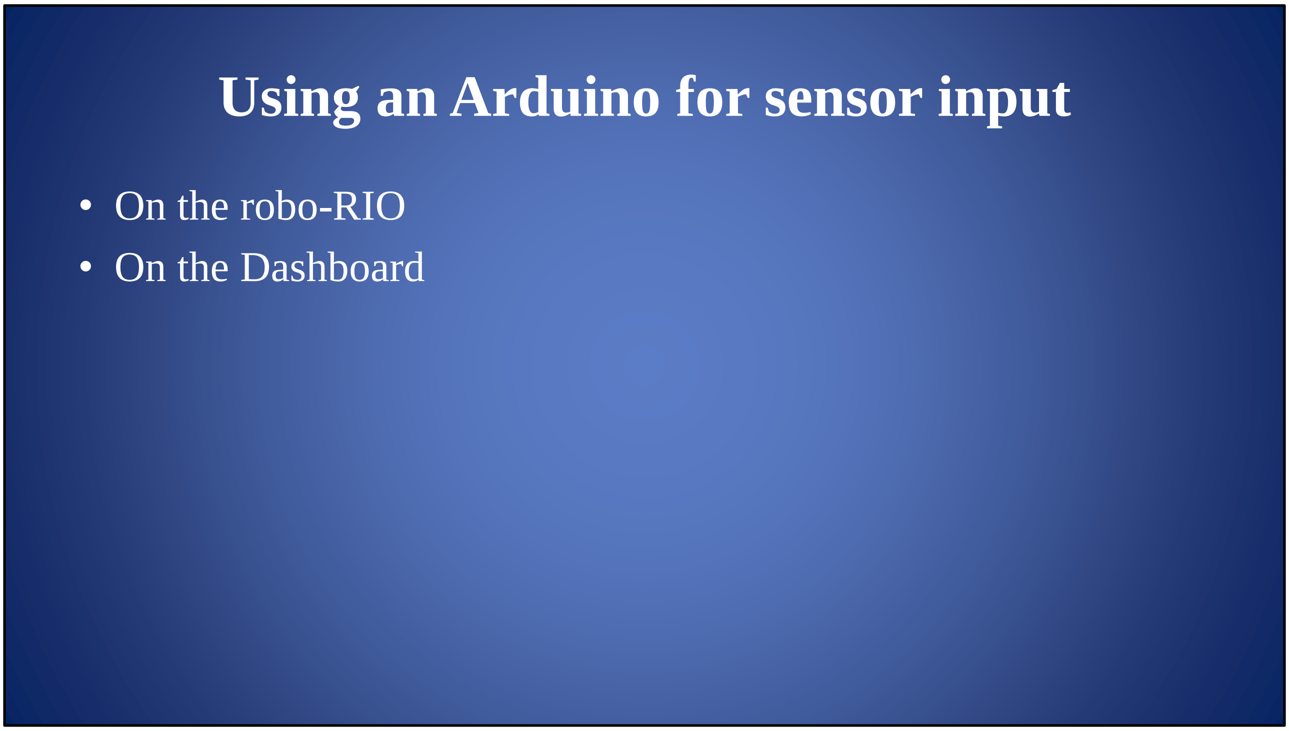
Code on roboRIO to handle loss of connection

Make sure baud rate matches – select port



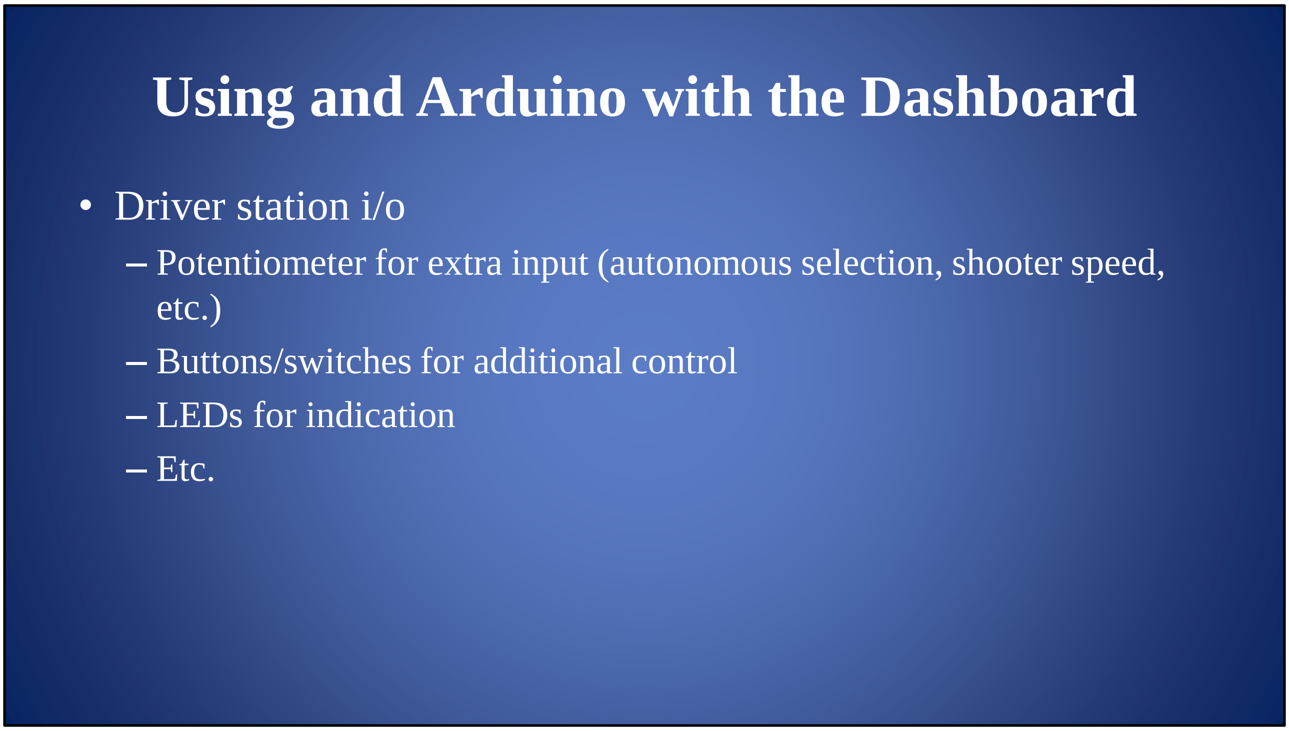
Using an Arduino for sensor input

- Demo



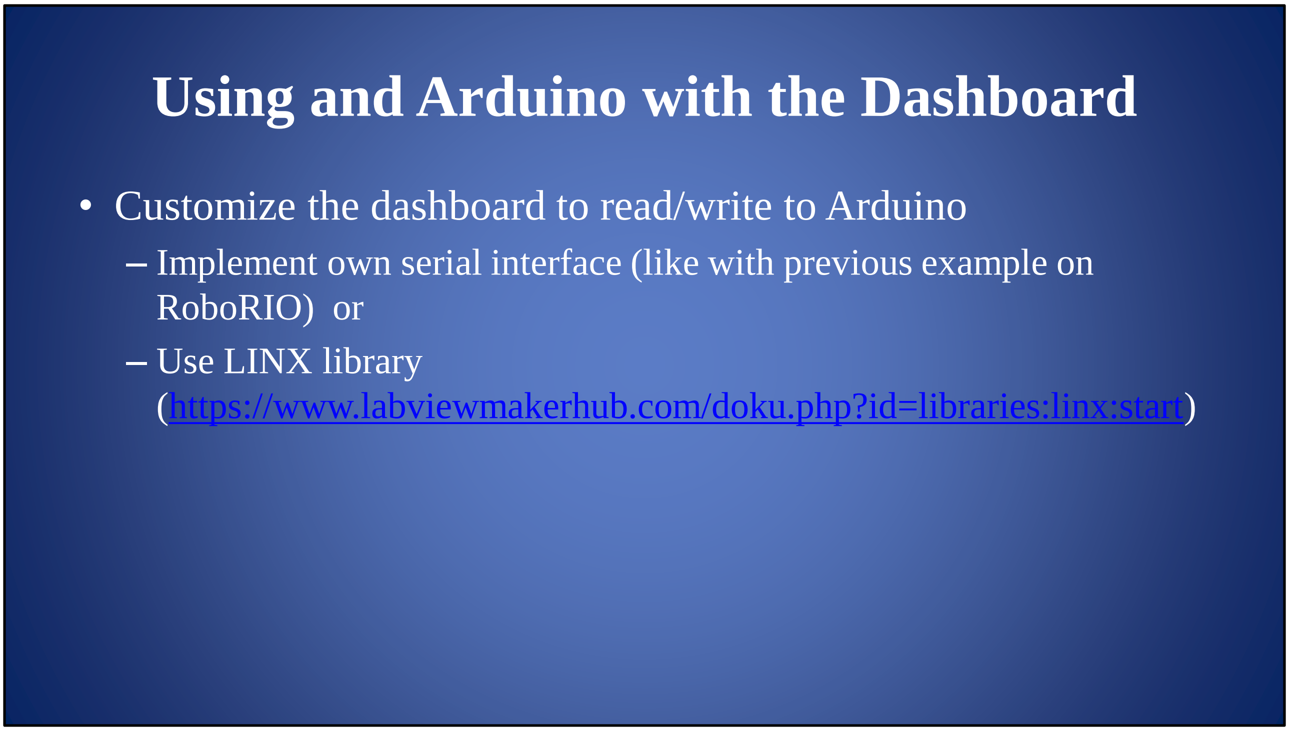
Using an Arduino for sensor input

- On the robo-RIO
- On the Dashboard



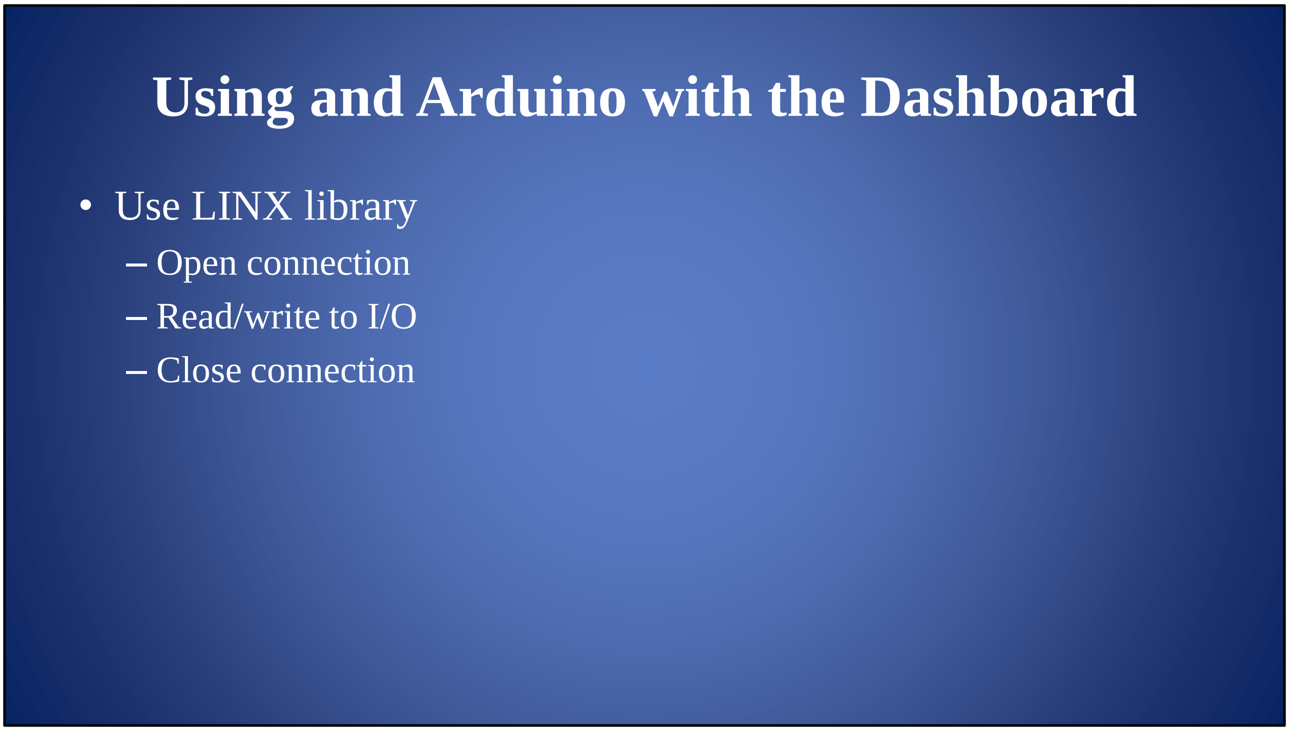
Using and Arduino with the Dashboard

- Driver station i/o
  - Potentiometer for extra input (autonomous selection, shooter speed, etc.)
  - Buttons/switches for additional control
  - LEDs for indication
  - Etc.



Using and Arduino with the Dashboard

- Customize the dashboard to read/write to Arduino
  - Implement own serial interface (like with previous example on RoboRIO) or
  - Use LINX library (https://www.labviewmakerhub.com/doku.php?id=libraries:linx:start)

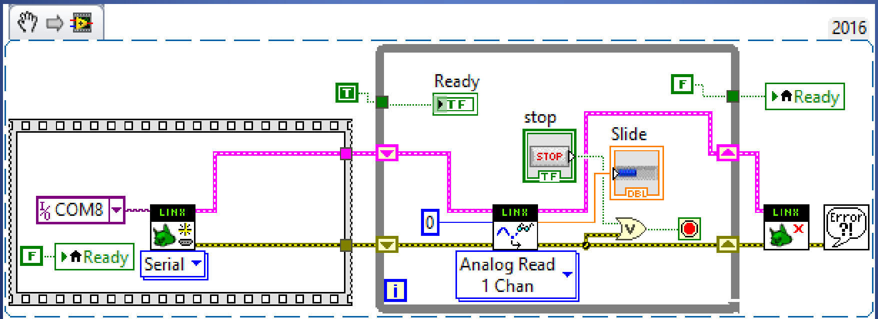


Using and Arduino with the Dashboard

- Use LINX library
  - Open connection
  - Read/write to I/O
  - Close connection

Using and Arduino with the Dashboard

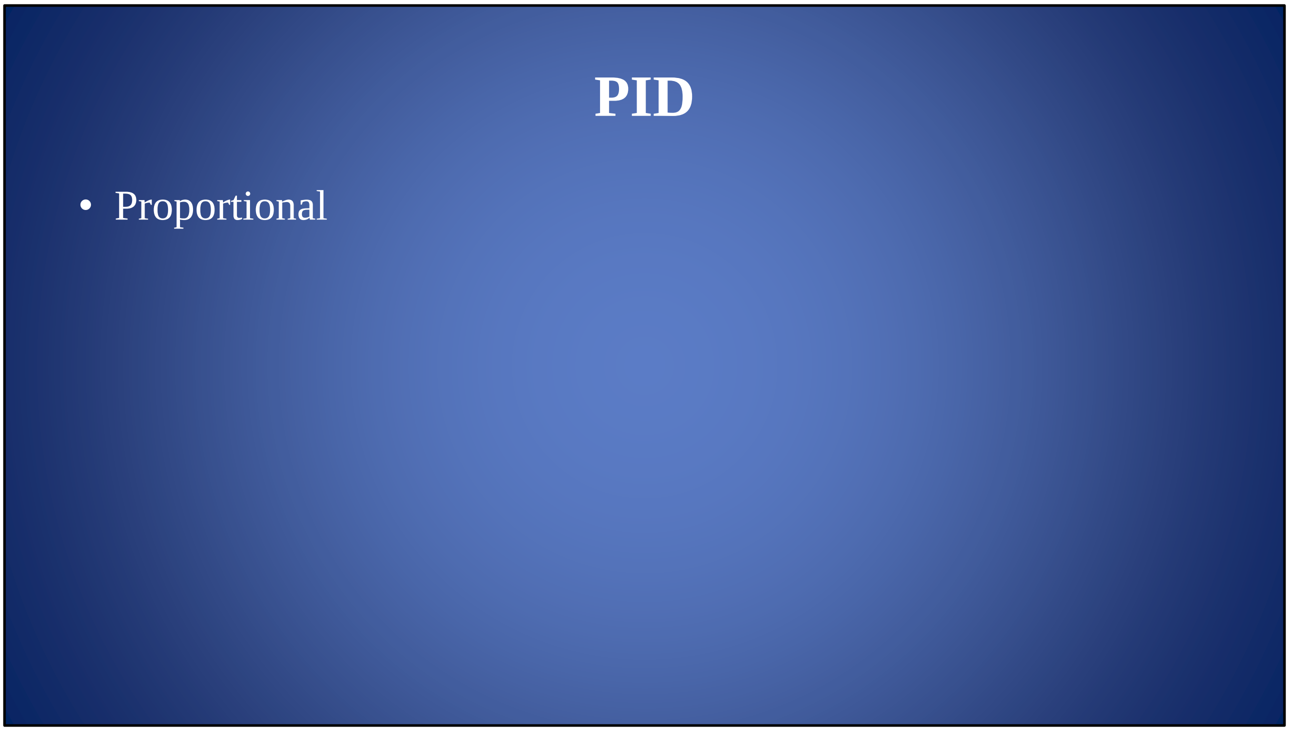
- Use LINX library
  - Open connection
  - Read/write to I/O
  - Close connection



Works in built exe with/without pressing stop. Need to press stop in dev (will leave the port reserved).

Using and Arduino with the Dashboard

- Demo



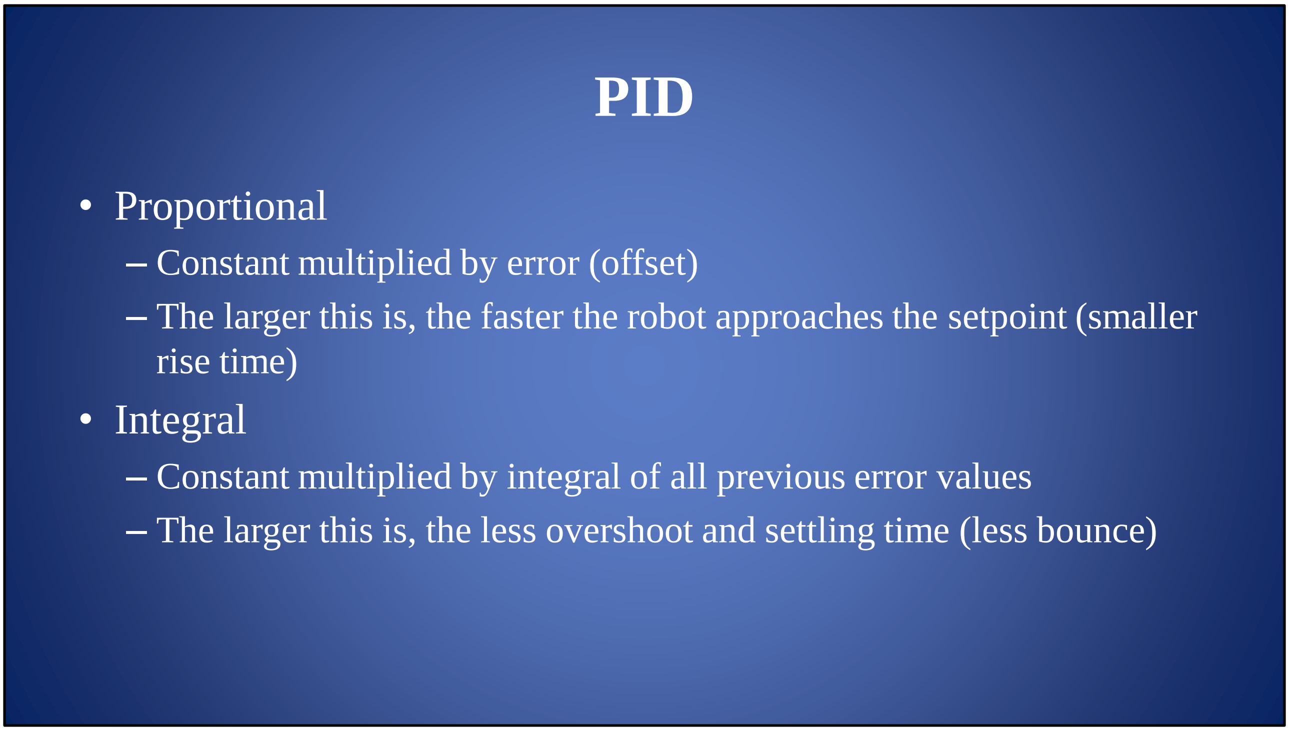
PID

- Proportional

https://docs.google.com/viewer?a=v&pid=sites&srcid=aGFyZGluZy5lZHV8dGVhbS0zOTM3fGd4OjUyNzdiNzRkNjkxNjA3MGM
https://www.youtube.com/watch?v=JEpWlTl95Tw
https://www.youtube.com/watch?v=UR0hOmjaHp0
http://robotics.stackexchange.com/questions/167/what-are-good-strategies-for-tuning-pid-loops

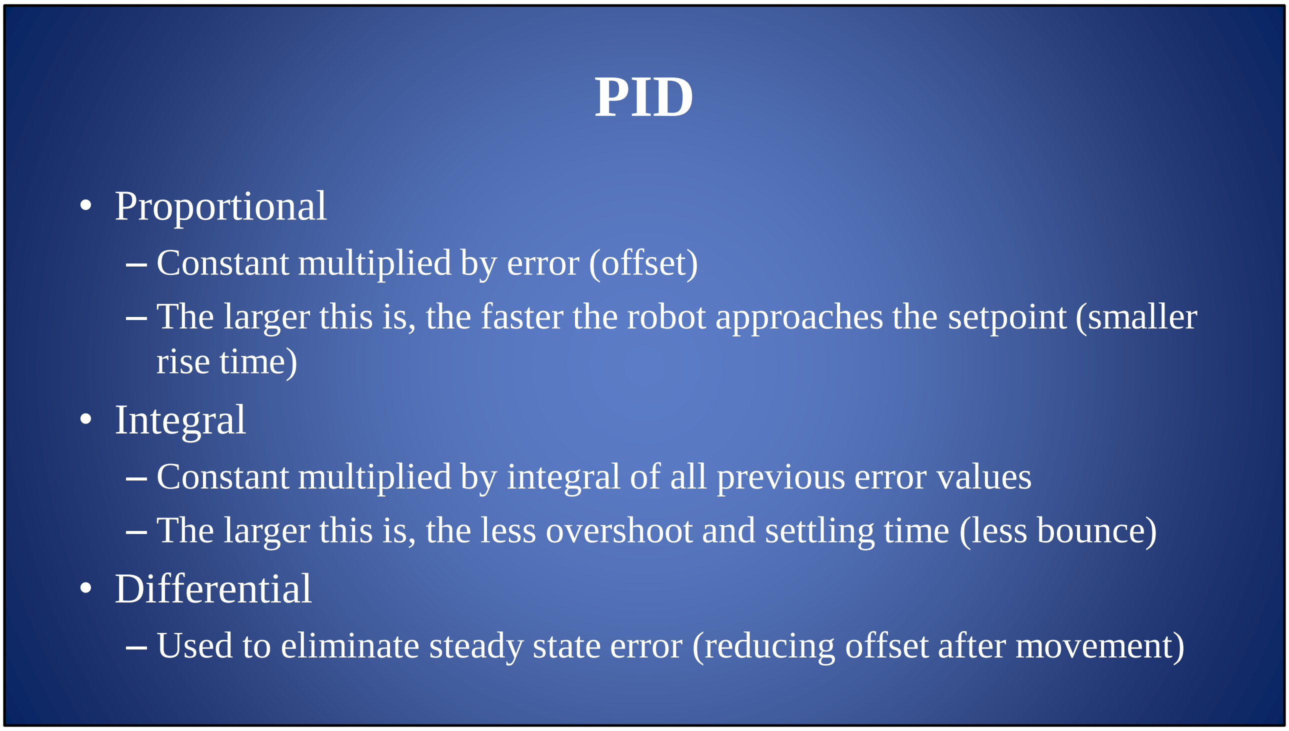
PID

- Proportional
  - Constant multiplied by error (offset)
  - The larger this is, the faster the robot approaches the setpoint (smaller rise time)



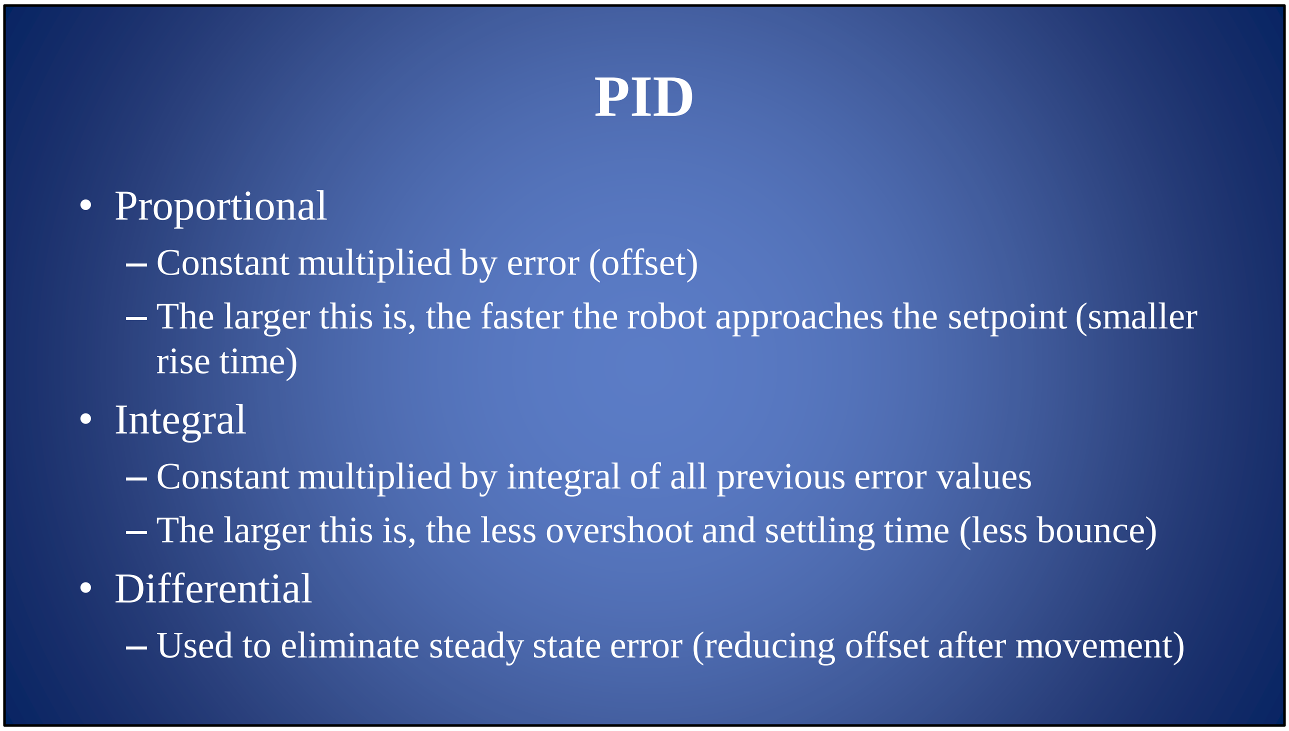
PID

- Proportional
  - Constant multiplied by error (offset)
  - The larger this is, the faster the robot approaches the setpoint (smaller rise time)
- Integral
  - Constant multiplied by integral of all previous error values
  - The larger this is, the less overshoot and settling time (less bounce)



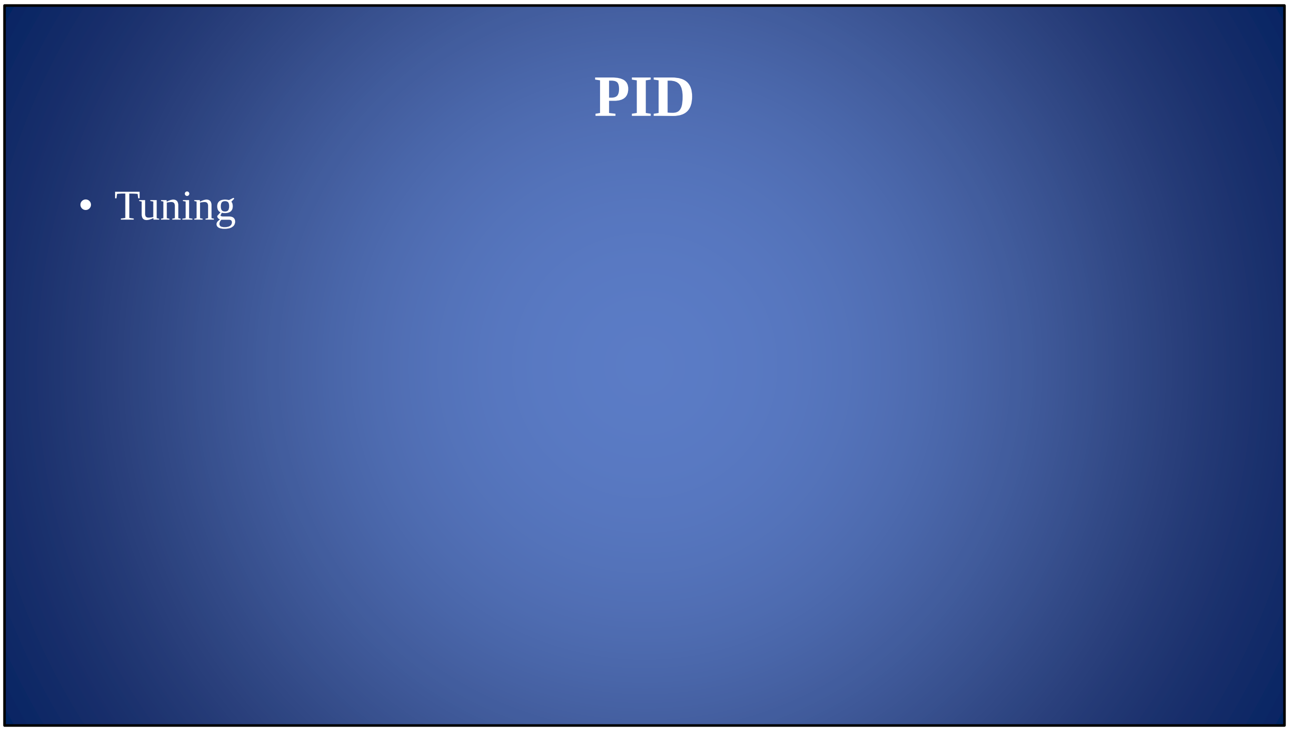
PID

- Proportional
  - Constant multiplied by error (offset)
  - The larger this is, the faster the robot approaches the setpoint (smaller rise time)
- Integral
  - Constant multiplied by integral of all previous error values
  - The larger this is, the less overshoot and settling time (less bounce)
- Differential
  - Used to eliminate steady state error (reducing offset after movement)



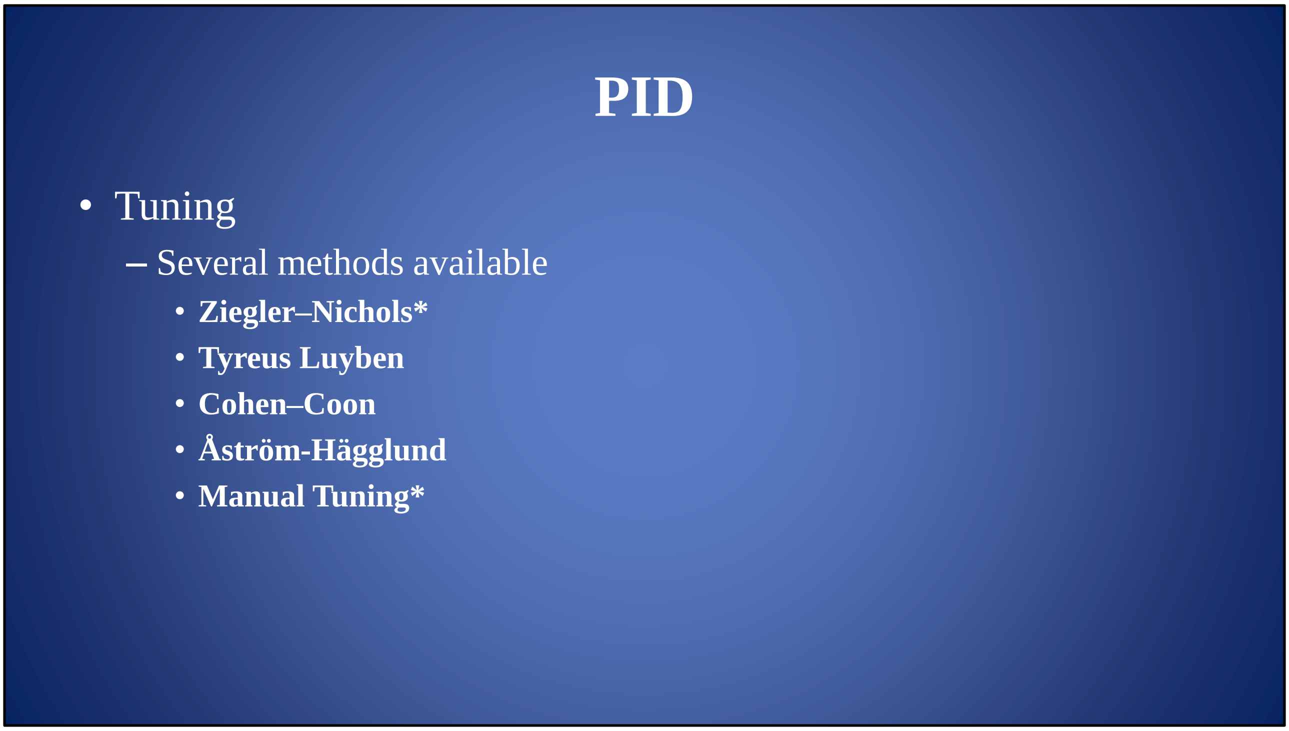
PID

- Proportional
  - Constant multiplied by error (offset)
  - The larger this is, the faster the robot approaches the setpoint (smaller rise time)
- Integral
  - Constant multiplied by integral of all previous error values
  - The larger this is, the less overshoot and settling time (less bounce)
- Differential
  - Used to eliminate steady state error (reducing offset after movement)



PID

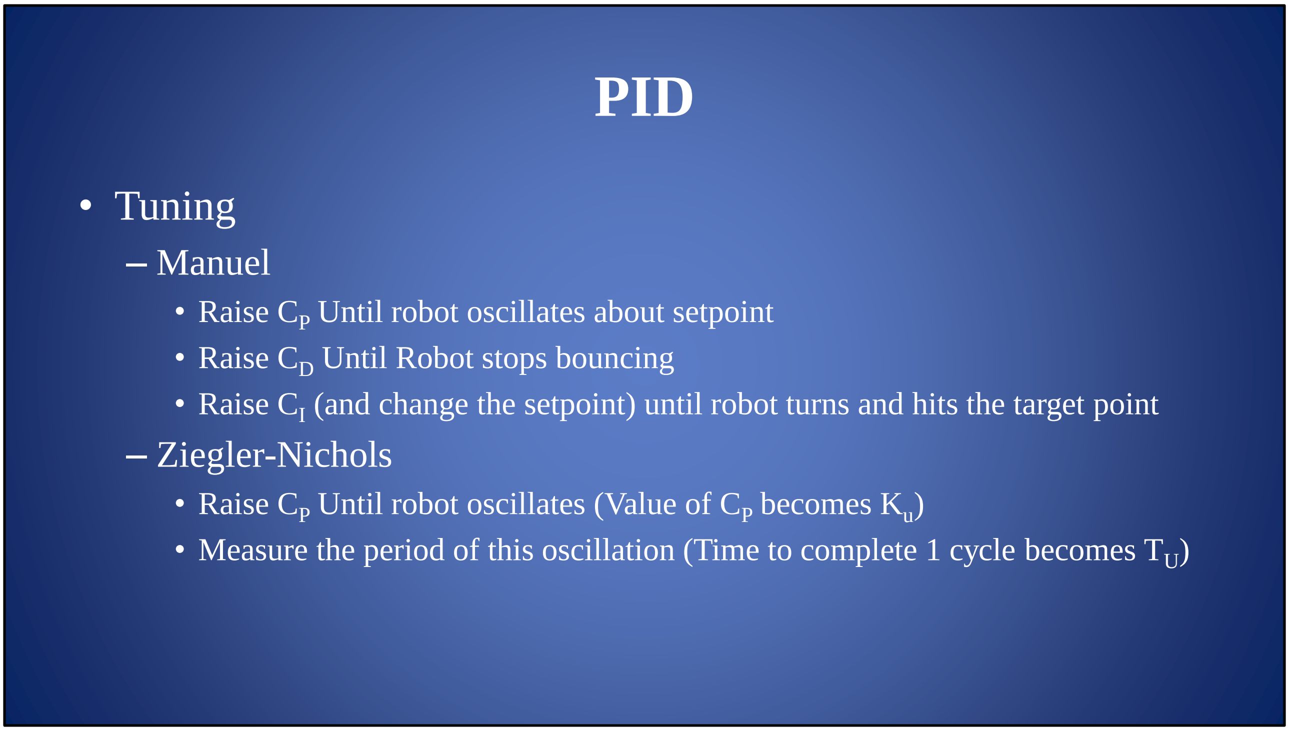
- Tuning



PID

- Tuning
  - Several methods available
    - **Ziegler–Nichols***
    - **Tyreus Luyben**
    - **Cohen–Coon**
    - **Åström-Hägglund**
    - **Manual Tuning***

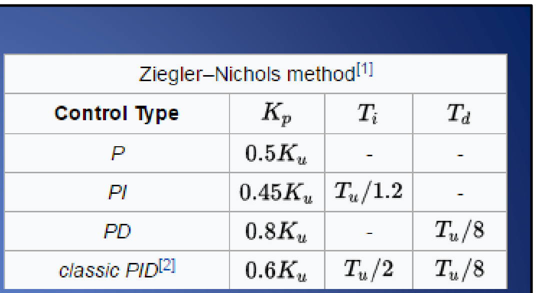
http://faculty.mercer.edu/jenkins_he/documents/TuningforPIDControllers.pdf#page=6
https://www.youtube.com/watch?v=JEpWlTl95Tw
https://www.youtube.com/watch?v=UR0hOmjaHp0
http://robotics.stackexchange.com/questions/167/what-are-good-strategies-for-tuning-pid-loops
Ziegler-Nichols: http://robotsforroboticists.com/pid-control/
Manual (page 16):
https://docs.google.com/viewer?a=v&pid=sites&srcid=aGFyZGluZy5lZHV8dGVhbS0zOTM3fGd4OjUyNzdiNzRkNjkxNjA3MGM
http://www.ni.com/white-paper/3782/en/



PID

- Tuning
  - Manuel
    - Raise C_P Until robot oscillates about setpoint
    - Raise C_D Until Robot stops bouncing
    - Raise C_I (and change the setpoint) until robot turns and hits the target point
  - Ziegler-Nichols
    - Raise C_P Until robot oscillates (Value of C_P becomes K_u)
    - Measure the period of this oscillation (Time to complete 1 cycle becomes T_U)

PID

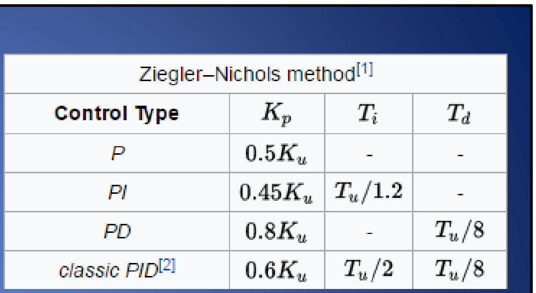


Ziegler–Nichols method[1]

Control Type	K_p	T_i	T_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
classic PID[2]	$0.6K_u$	$T_u/2$	$T_u/8$

- Tuning
  - Manuel
    - Raise C_P Until robot oscillates about setpoint
    - Raise C_D Until Robot stops bouncing
    - Raise C_I (and change the setpoint) until robot turns and hits the target point
  - Ziegler-Nichols
    - Raise C_P Until robot oscillates (Value of C_P becomes K_u)
    - Measure the period of this oscillation (Time to complete 1 cycle becomes T_U)

PID



Ziegler–Nichols method[1]

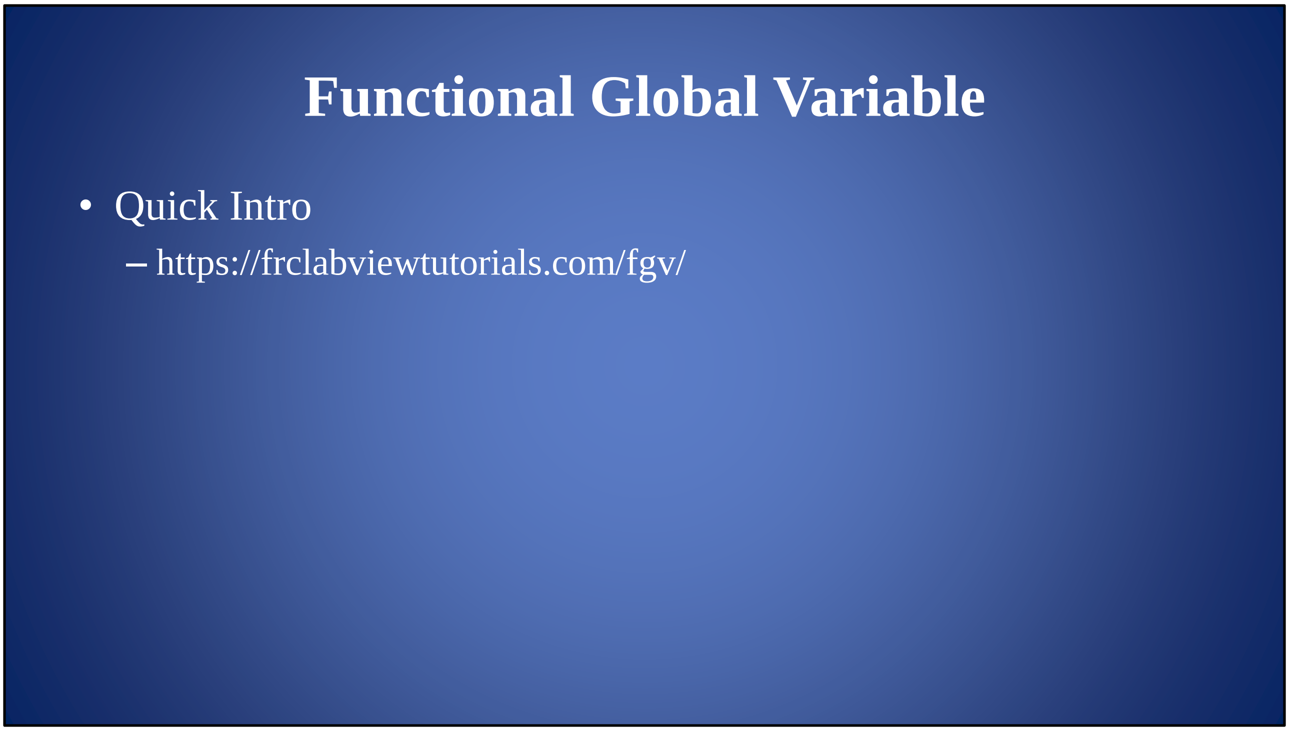
Control Type	K_p	T_i	T_d
P	$0.5K_u$	-	-
PI	$0.45K_u$	$T_u/1.2$	-
PD	$0.8K_u$	-	$T_u/8$
classic PID[2]	$0.6K_u$	$T_u/2$	$T_u/8$

- Tuning
  - Manuel
    - Raise C_P Until robot oscillates about setpoint
    - Raise C_D Until Robot stops bouncing
    - Raise C_I (and change the setpoint) until robot turns and hits the target point
  - Ziegler-Nichols
    - Raise C_P Until robot oscillates (Value of C_P becomes K_u)
    - Measure the period of this oscillation (Time to complete 1 cycle becomes T_U)

PID

- Demo

Functional Global Variable

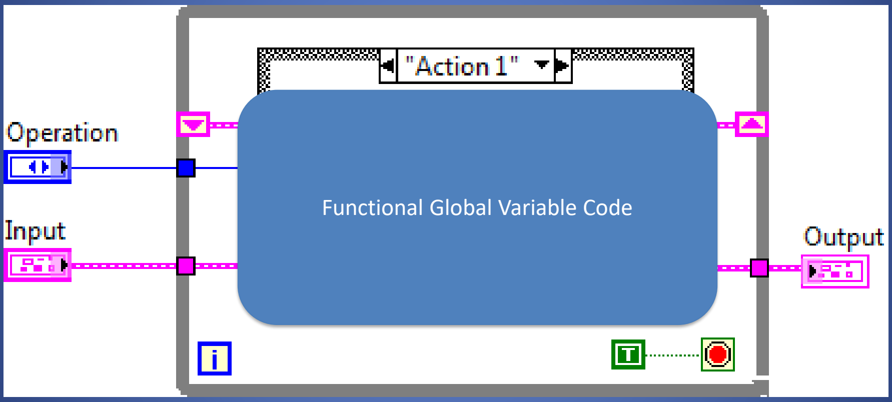


Functional Global Variable

- Quick Intro
  - – https://frclabviewtutorials.com/fgv/

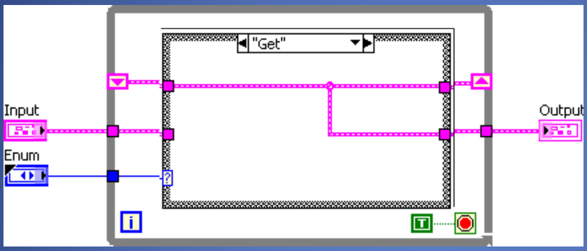
demo

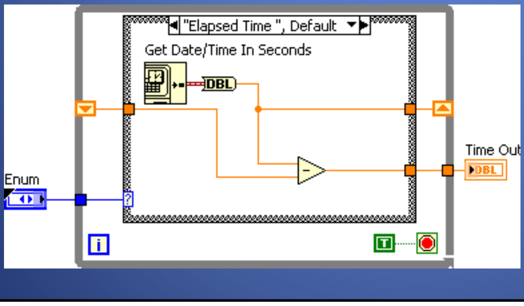
FGV



"Action 1"
Operation
Input
Functional Global Variable Code
Output
i
T

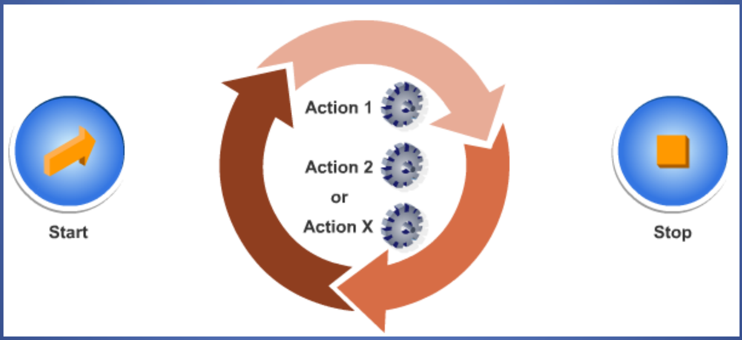
Implementing An FGV





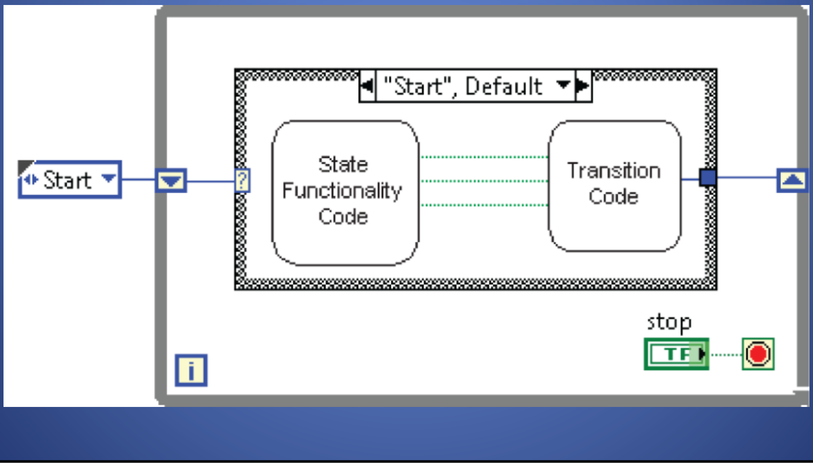
Architectures

- State Machine



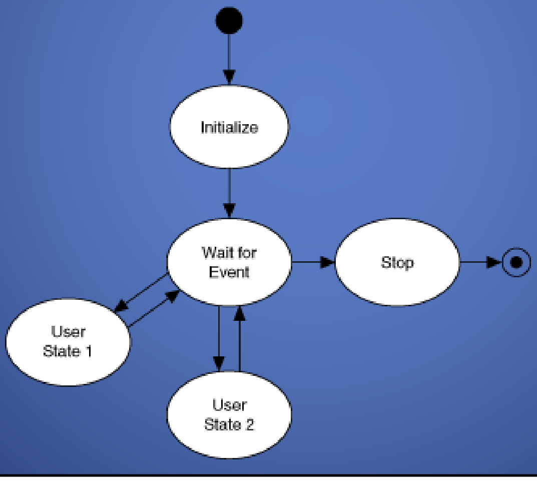
Architectures

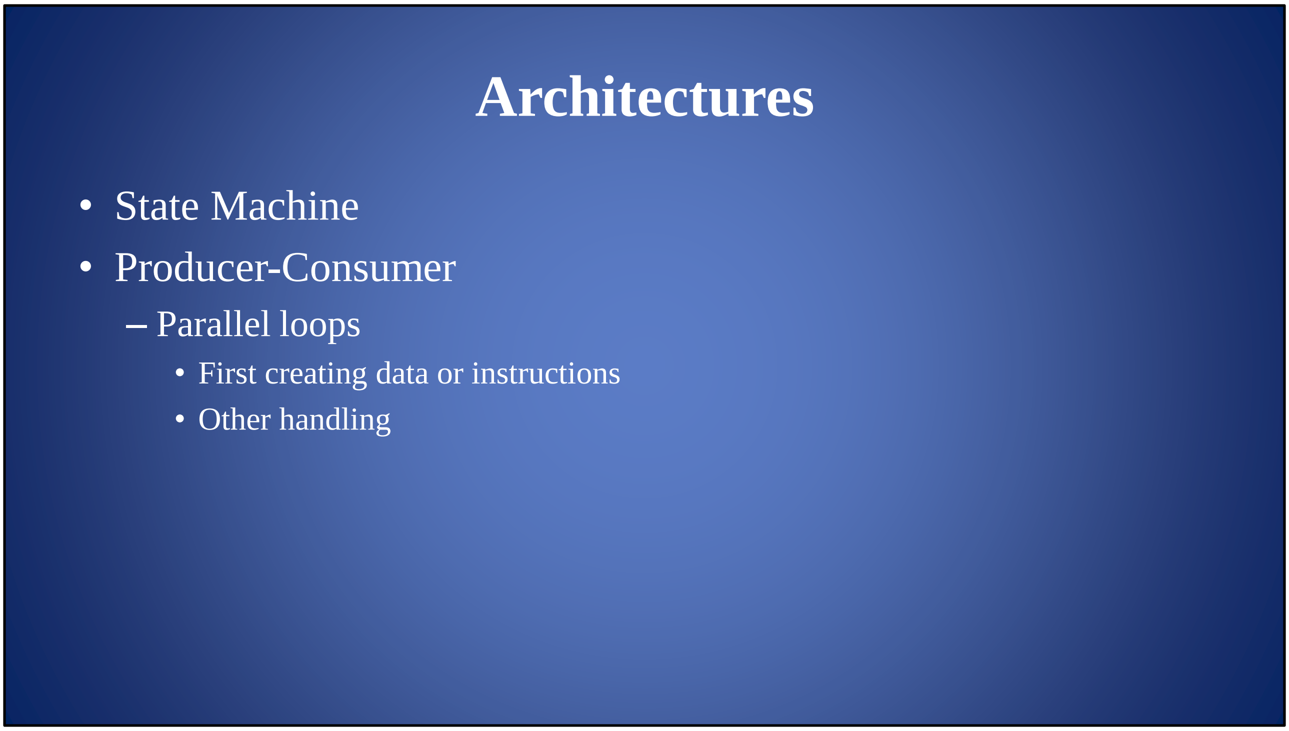
- State Machine



Architectures

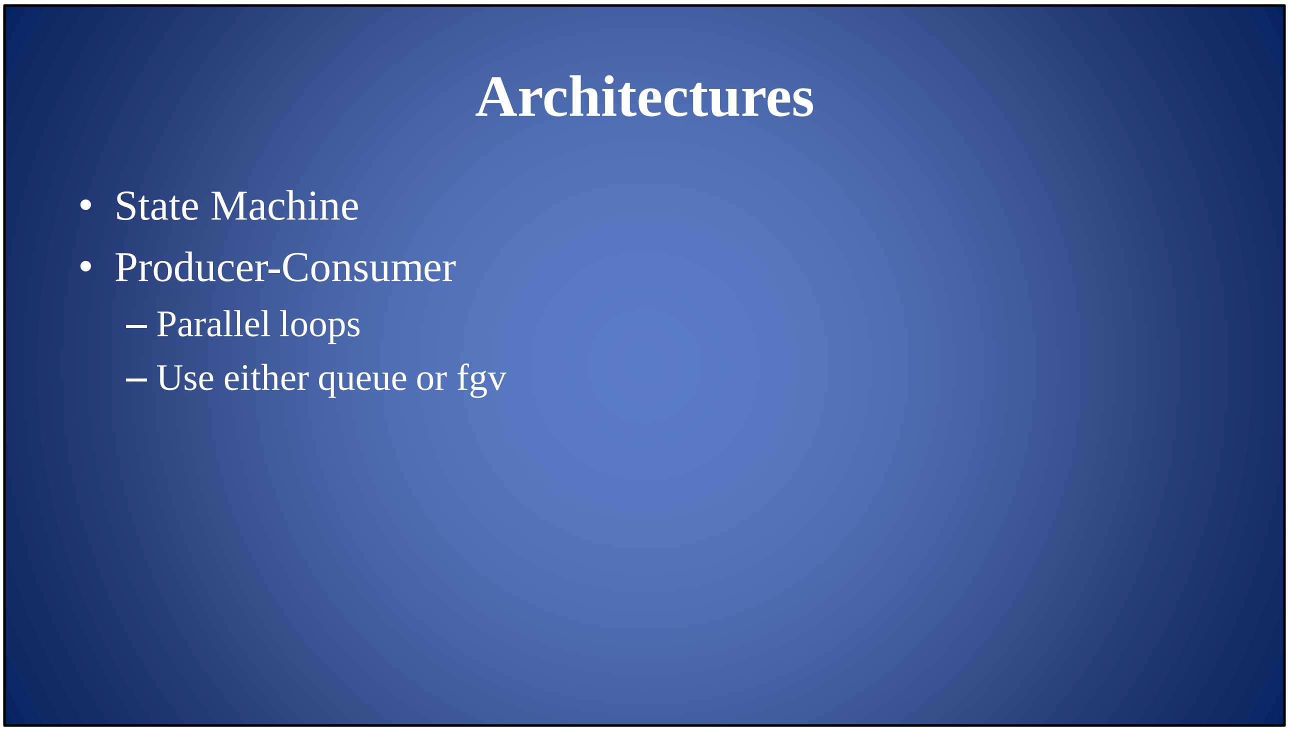
- State Machine





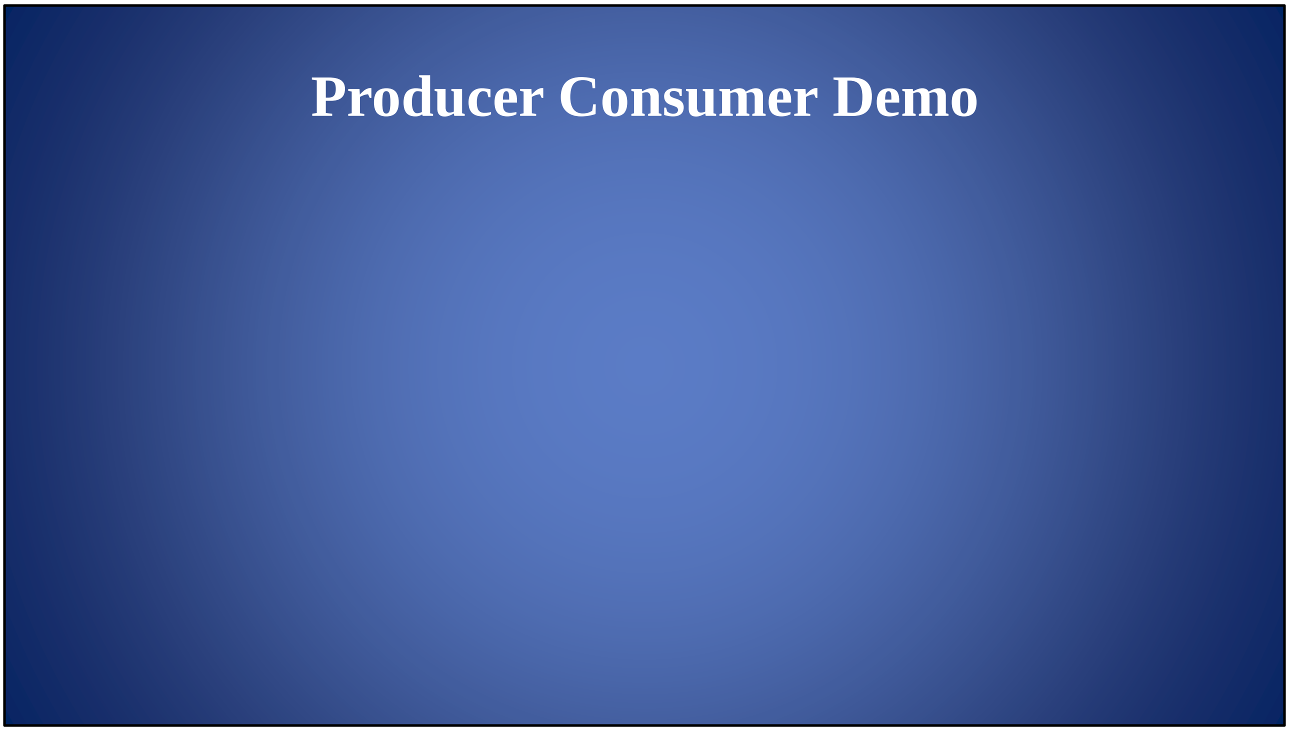
Architectures

- State Machine
- Producer-Consumer
  - – Parallel loops
    - First creating data or instructions
    - Other handling



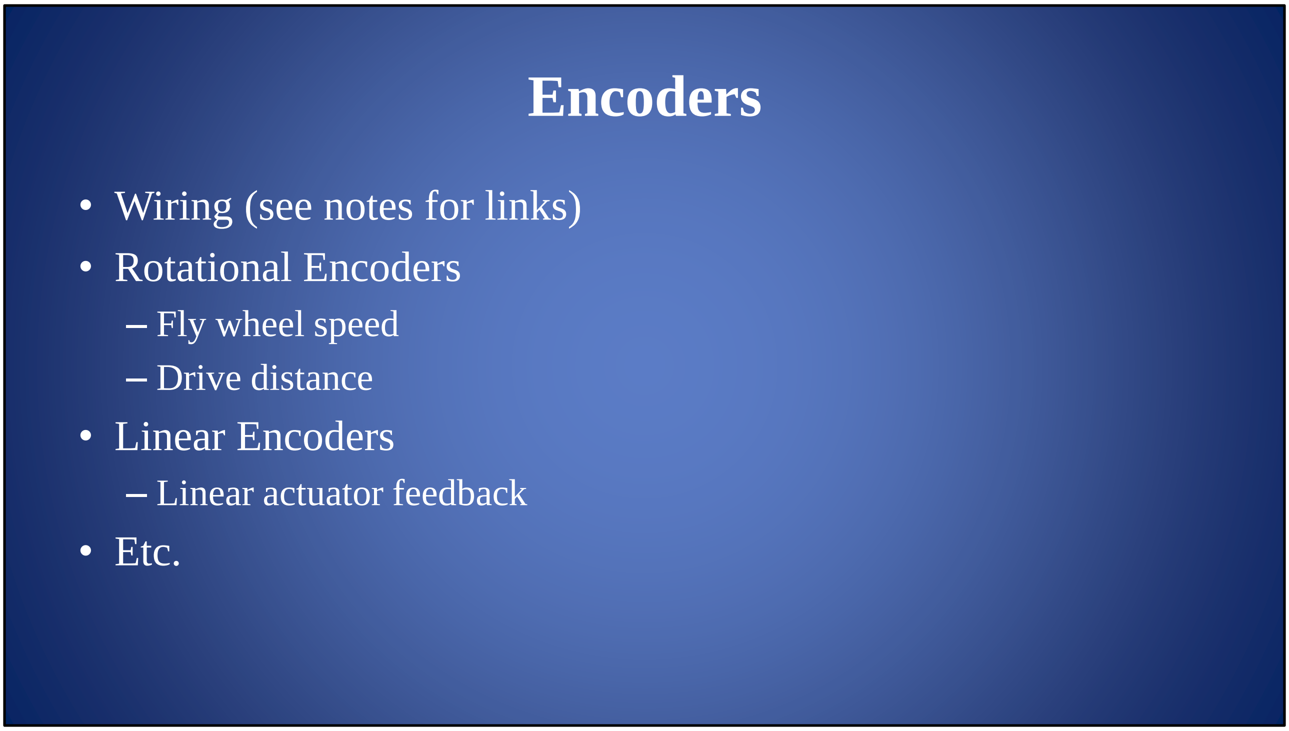
Architectures

- State Machine
- Producer-Consumer
  - Parallel loops
  - Use either queue or fgv



Producer Consumer Demo

Queue and FGV

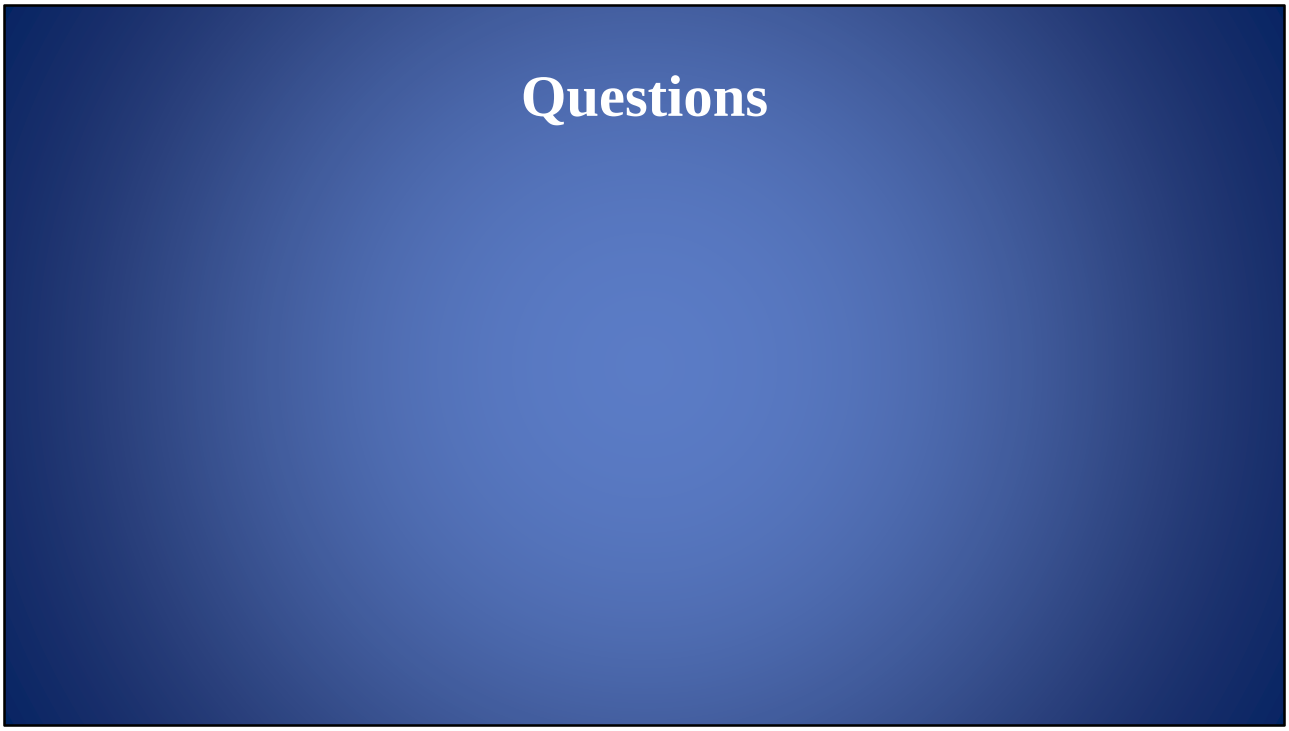


Encoders

- Wiring (see notes for links)
- Rotational Encoders
  - Fly wheel speed
  - Drive distance
- Linear Encoders
  - Linear actuator feedback
- Etc.

https://www.chiefdelphi.com/forums/showthread.php?t=133263
https://www.andymark.com/encoder-p/am-3314.htm
https://www.andymark.com/product-p/am-2992.htm

Encoders – Fly Wheel monitor demo



Questions