Team 2537
Control Systems 2016

The Space RAIDers

Control Systems Executive Summary

Key Features:

- Sensor Network
- Command Based Architecture
- PID Control
- Course Correction
- Vision Software
- Visual Aids for Drivers

Sensor Network

- Sensors Controller receives sensor information from IMU's, Lidar/Ultrasonic, and Infrared Sensors.
- Controller sends out data to subsystems that need it e.g. the Arm Mechanism's Subsystem receives data about the arm's angle.

Command Based Architecture

- Underlying structure for robot code.
- "Command" classes are programmed for certain functions and told to run by a master Scheduler.
- "Subsystem" classes provides information on the condition of a specific mechanism.

PID Control

- Method of motor control that minimizes errors in approaching a speed or position.
- Used to: spin up shooter flywheels, pivot arm mechanism, pivot shooter mechanism.
- Sets closed loop on Talon Breakout Boards to minimize latency.

Course Correction

- Autonomous code that makes the robot drive in a straight line despite outside forces.
- The robot gets its angle when it initializes and corrects for any changes while driving to keep on course.

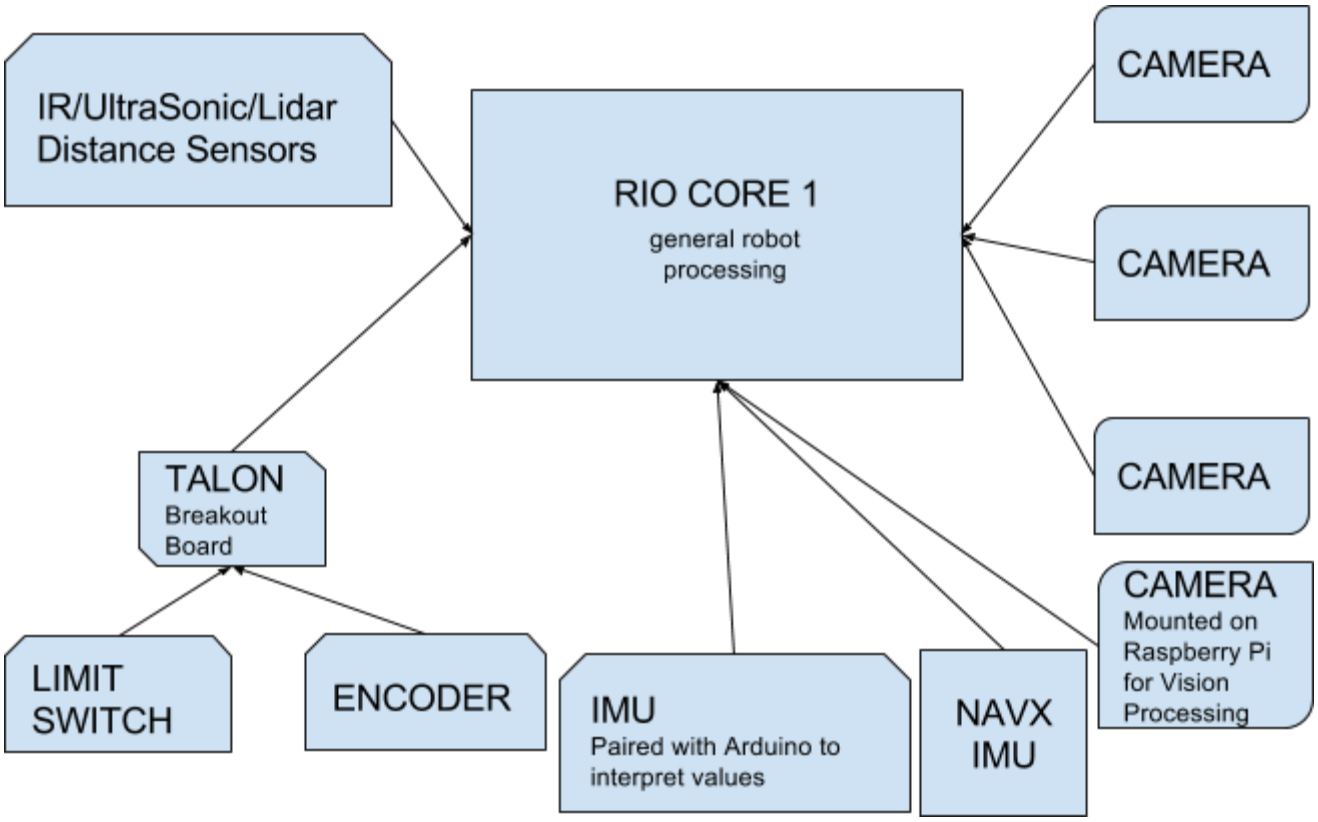
Visual Aids

- Camera system displays three vertical lines on the shooter camera screen which provide assistance for shooting boulders and toggles between the three cameras.

Vision Processing

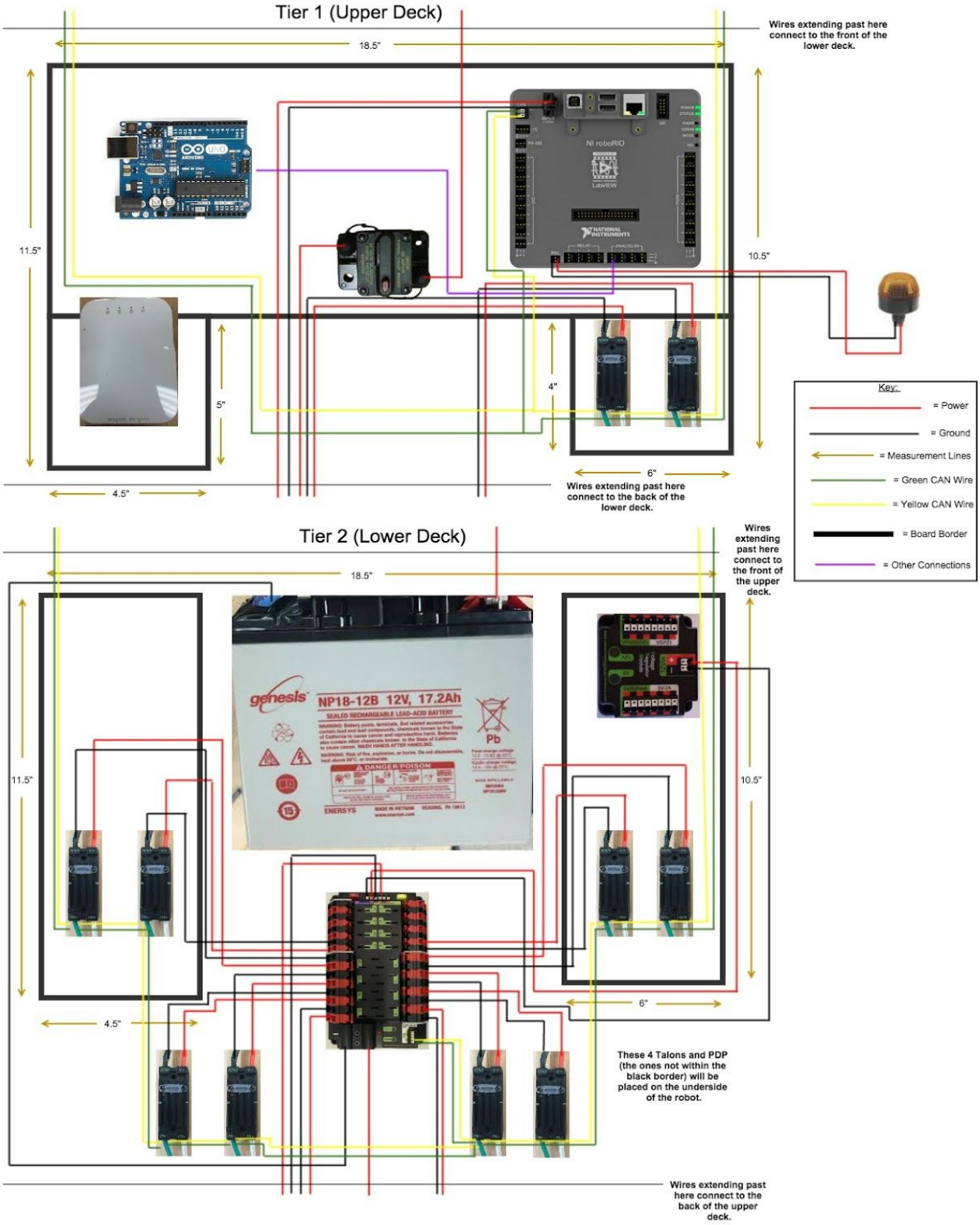
- A method of adjusting the robot to accurately and autonomously shoot boulders into a high goal.
- Shines a light onto the retro reflective tape reflective tape and looks for goals
- Provides feedback to the robot as to where the robot should be located and how the shooter should be angled.

Sensor Electrical Data Flow Map

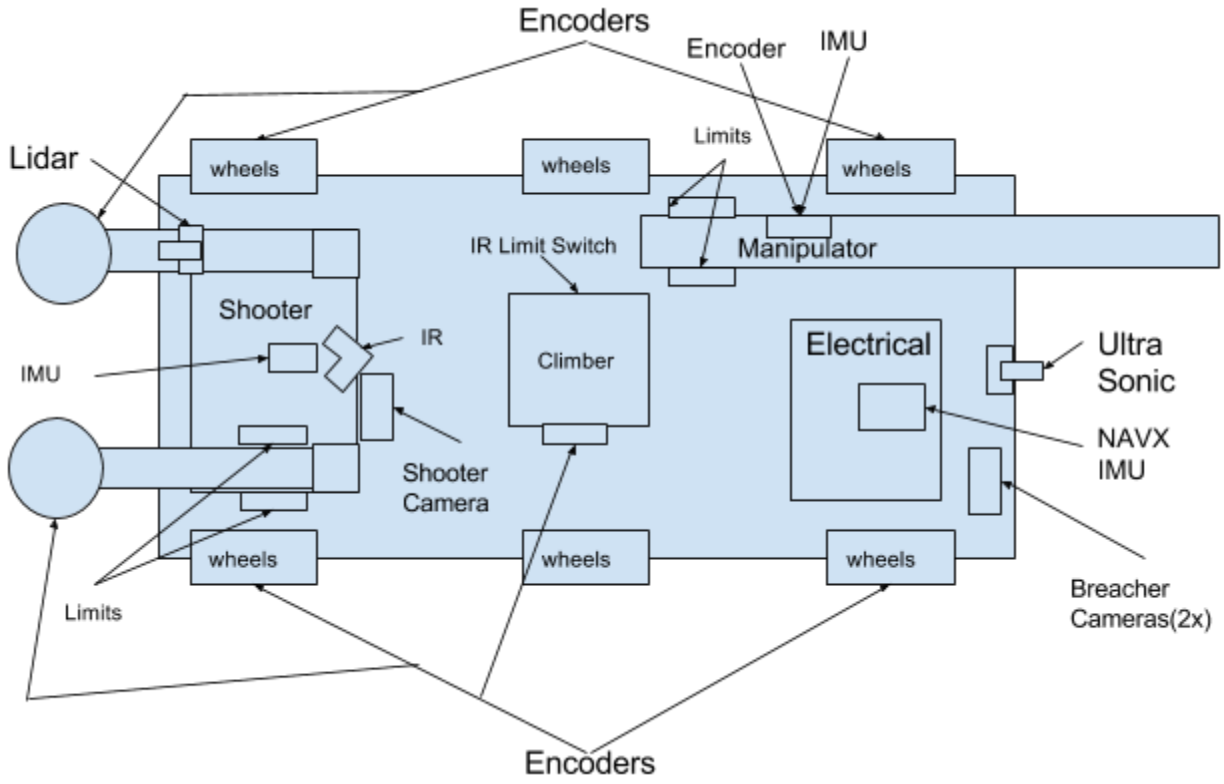


Data flow from various sensors to the RoboRIO

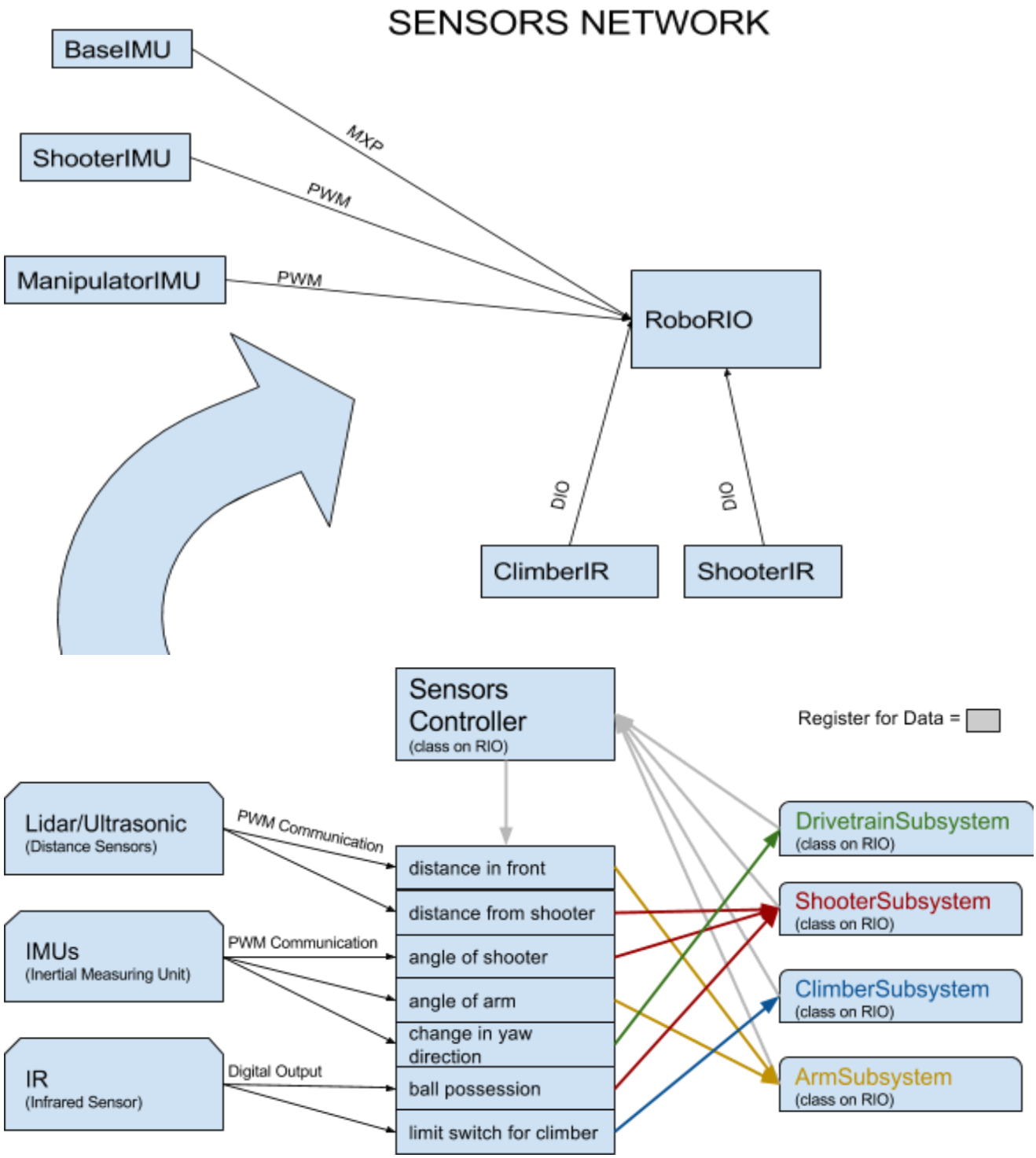
Electrical Board Layout



Sensors Breakdown

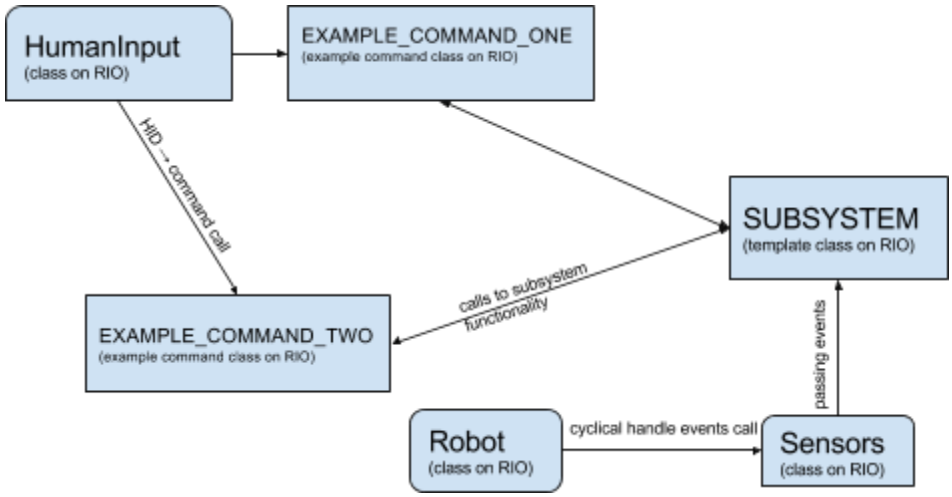


Primary Sensor	**Failsafe Sensor**	**Measure**
Encoders	Ammeters	Drive Wheels
Ultrasonic	-	distance to defense
Lidar	-	distance to goal (mounted on shooter)
IR	-	boulder in possession
IR	Encoder	climbing tick counter
IMU	Limit Switches	angle/range of shooter pivot
Encoder	Ammeters	speed of shooter flywheels
Encoder	Linereader	climb height
IMU	Limit Switches	angle/range of arm
Cameras	-	vision & driver assists



Internal to the RoboRIO, the sensor framework applies the listener/observer design pattern to distribute sensor data to the mechanism subsystems. Subsystems register with the sensors controller and receive only data that they wish to receive.

Command Based Architecture

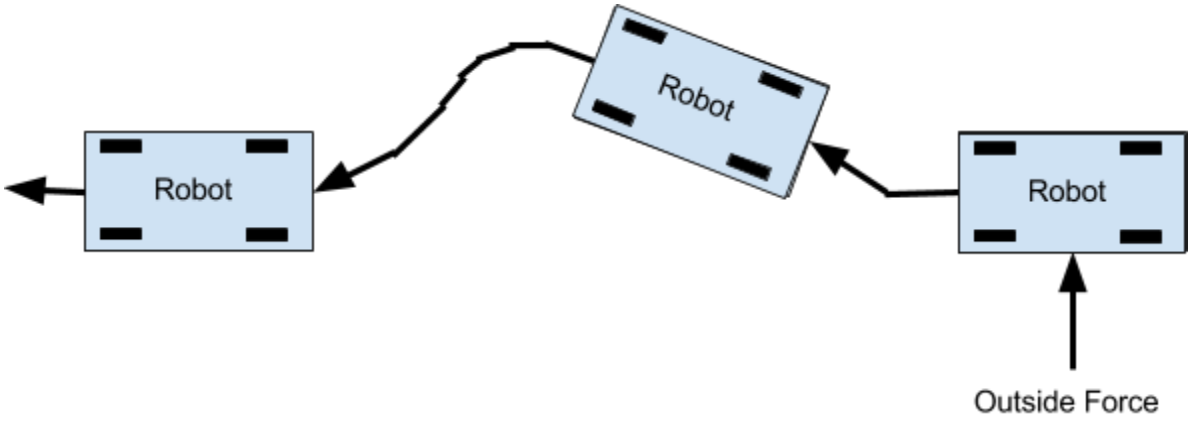


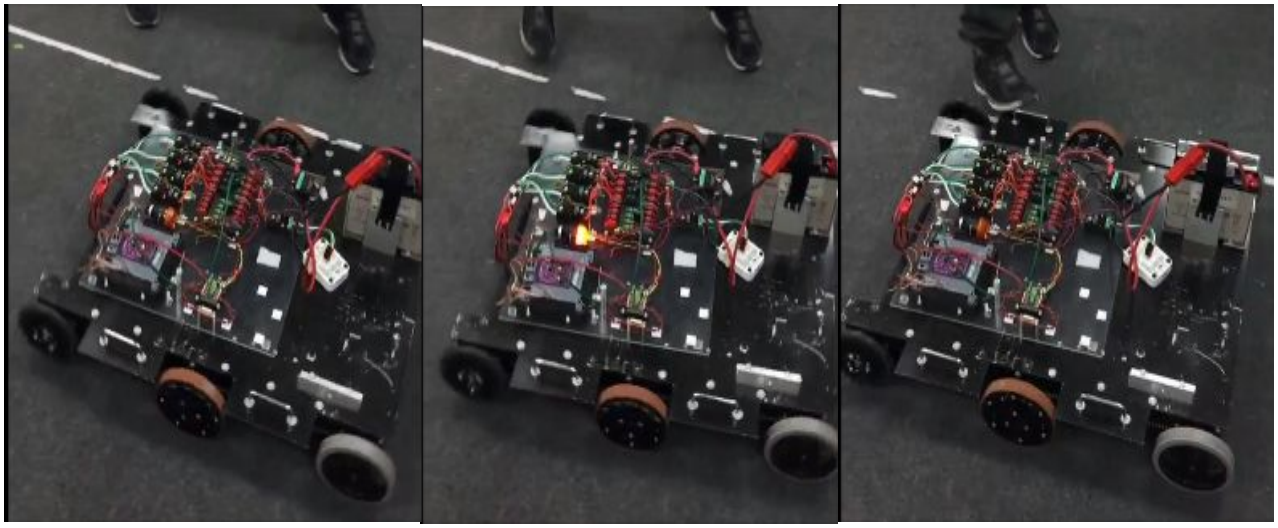
The Command Based coding architecture is the basis for operation on the RoboRIO. In order to implement the Command Based architecture, classes detailing specific actions are created, assigned to button presses, and scheduled with the master Scheduler class. The subsystem provides information about the state of the robot and access to the physical components of the robot. Finally the Sensors class reads electrical signals from Digital Input Output(DIO) ports and distributes the pertinent information to specific subsystems so that a subsystem can retrieve sensor values that are relevant to its subsystem.

PID Control

PID is a method of closed loop motor control that allows us to move the motors to a setpoint whether that be a speed or a position. The robot allows us two implementations of PID Control - the Talon's implementation and WPILib's implementation. The Talon's implementation works through the Talon breakout boards, making it relatively simple to use. This implementation is used to control the angle of the arm position based on feedback from an encoder. It is also used on the shooter to set the flywheels to a certain speed based on feedback from a quad encoder to keep the flywheels at a consistent speed while shooting and harvesting. A second implementation of PID Control is the WPILib implementation, which specifies the input device, the PID values, and the output device. This implementation is used on the shooter pivot because the shooter pivot does not have an encoder that plugs into a breakout board, but rather an IMU that plugs directly into the RoboRIO.

Course Correction





Course correction is code used in autonomous that makes the robot drive in a straight line while correcting for unwanted rotation. First the robot gets a distance to travel from the autonomous plan chooser and angle when it initializes. Then, the robot drives until it reaches the specified distance, and whenever the robot's current angle is different than robot's initial angle the robot turns until it reaches the initial angle. The robot gets the angle with the navX-MXP IMU, and calculates distance travelled using encoders.

Camera System (Visual Aids)

Three vertical lines allow guidance for on the shooter screen. The center line is for shooting, while the other two line show where the arms will fall and where the breacher lands.

Camera System (Vision Processing)

A green light is shone on the reflective tape around the goal before taking a picture. A color filter applied to the image isolates all green reflective surfaces. Contour lines are calculated based on a binary threshold of the of the isolated objects. If more than one object is detected, binary threshold objects are sent to a neural network to detect which object is the reflective tape. A neural network is a mathematical emulation of the supervised learning system in humans. Through a training process, a neural network learns and is optimized to certain problems otherwise difficult or impossible to solve algorithmically. Although this step is not necessary, it makes the algorithm more reliable and effective than without the security and flexibility of artificial intelligence. The neural network is trained on a series of binary threshold images of the reflective tape around the goal as well as a series of randomized binary threshold objects, so that it may see examples of objects that it should not categorize as the goal. After the reflective tape is detected, the edges are converted to an angle to inch ratio, which allows us to calculate the position of the target relative to the robot. With this data, the robot is able to be positioned on the x axis by rotating itself and on the z axis by manipulating the shooter angle. The necessary angle is calculated based on experimental data of the parabola made by a boulder propelled by flywheels running at a constant speed. This parabola is virtually constant, and therefore we can calculate the function needed to intersect the goal.