Table of Contents

[Mechanical build notes 4](#_Toc2183741)

[OpenCV Vision with Network tables on RasPi 4](#_Toc2183742)

[Setup 4](#_Toc2183743)

[Misc links 4](#_Toc2183744)

[OpenCV 4](#_Toc2183745)

[Draw on image 5](#_Toc2183746)

[Trig 5](#_Toc2183747)

[Recommendations 5](#_Toc2183748)

[Many guides and articles for Python & image processing 5](#_Toc2183749)

[Vision processing on desktop for simulation and network tables 5](#_Toc2183750)

[Pi Vision including setup, project build and deploy. 5](#_Toc2183751)

[GRIP vision 6](#_Toc2183752)

[Vision/Network table examples 6](#_Toc2183753)

[OpenCV on RoboRIO 6](#_Toc2183754)

[FRC networking 6](#_Toc2183755)

[FRC networking basics 6](#_Toc2183756)

[Change RasPi mDNS name 6](#_Toc2183757)

[Troubleshooting 7](#_Toc2183758)

[FRC IP networking at events 7](#_Toc2183759)

[Network tables 7](#_Toc2183760)

[Using network tables 7](#_Toc2183761)

[Misc notes 7](#_Toc2183762)

[LOOK AT SHUFFLEBOARD !!! It can display tons of info including network tables 8](#_Toc2183763)

[Data recording & playback 8](#_Toc2183764)

[Displaying camera 8](#_Toc2183765)

[Graphs 8](#_Toc2183766)

[Displaying command based state information 8](#_Toc2183767)

[UDP messaging 8](#_Toc2183768)

[TCP messaging 8](#_Toc2183769)

[PID tuners etc… 8](#_Toc2183770)

[Misc stuff 9](#_Toc2183771)

[Start here ... 9](#_Toc2183772)

[Great article on programming for FRC. READ THIS!!! 9](#_Toc2183773)

[Systems as PID input 9](#_Toc2183774)

[Talon resources 9](#_Toc2183775)

[Misc notes. READ ME ! 9](#_Toc2183776)

[Make sure Rio is imaged correctly for 2019 9](#_Toc2183777)

[Understand the new Phoenix tuner application 9](#_Toc2183778)

[New project test example 9](#_Toc2183779)

[Quadrature and limit switch sensor testing 10](#_Toc2183780)

[Sensor setup and testing 10](#_Toc2183781)

[Use the plot feature to 'see' things happen :) 10](#_Toc2183782)

[Motors, testing and calibration, READ ME ! 10](#_Toc2183783)

[Almost complete example of motor, sensor, drive, display info etc... here 10](#_Toc2183784)

[Motor 'followers' are described here 10](#_Toc2183785)

["Ramping" is discussed here 10](#_Toc2183786)

[Can read the following at any time... 10](#_Toc2183787)

[Sensor checking/debug 11](#_Toc2183788)

[See "Recommended Procedure" 11](#_Toc2183789)

["Motion Magic Control Mode" might be better for the lift, but not 100% sure yet 11](#_Toc2183790)

[Make sure we understand how to check, analyze and clear faults 11](#_Toc2183791)

[Examples exists here for the following... 11](#_Toc2183792)

[Motion profile generator and example code 11](#_Toc2183793)

[Motion Profiling 12](#_Toc2183794)

[Spline fitting 12](#_Toc2183795)

[Motion profiling article 12](#_Toc2183796)

[Motion planning video presentation 12](#_Toc2183797)

[Custom PID sources 12](#_Toc2183798)

[Good Talon blog with simulation environment 12](#_Toc2183799)

[Example motion profile from SteamWorks 12](#_Toc2183800)

[Useful function for periodic message display… 13](#_Toc2183801)

[Itterative vs Timed vs Command 13](#_Toc2183802)

[Command based joystick control 13](#_Toc2183803)

[Groups of commands 13](#_Toc2183804)

[Running commands whilst button pressed or held down. 13](#_Toc2183805)

[Using commands during auto & teleop 13](#_Toc2183806)

[Default/auto switching between joystick and commands 13](#_Toc2183807)

[Synchronizing commands 13](#_Toc2183808)

[Limit switches and commands 14](#_Toc2183809)

[High level overview 14](#_Toc2183810)

[Team 5940 code using PathWeaver 14](#_Toc2183811)

[Mechanisms, parts and components 14](#_Toc2183812)

[Single articulated mechs 14](#_Toc2183813)

[Cascade lifter mechs 14](#_Toc2183814)

[Bearing options 14](#_Toc2183815)

[Vex versa-blocks 14](#_Toc2183816)

[Linear actuator 14](#_Toc2183817)

[Competition robot parts (limited and expensive!!) 14](#_Toc2183818)

[Interesting COTS discussion 14](#_Toc2183819)

[CUI encoders 14](#_Toc2183820)

[CheesyVision - Huhh!!! Is this even leagal ??? 14](#_Toc2183821)

[Git software revision control and software management 15](#_Toc2183822)

[Make sure Git is installed on the computer you want to develop on 15](#_Toc2183823)

[Make sure you have an account on GitHub 15](#_Toc2183824)

[Creating a new Git repository from existing code on your computer 16](#_Toc2183825)

[‘Cloning’ code to your computer from a GitHub repository 16](#_Toc2183826)

[Making sure you have the latest code 17](#_Toc2183827)

[FTC teams 17](#_Toc2183828)

[FRC teams 17](#_Toc2183829)

[Checking in changes and pushing to Github 17](#_Toc2183830)

[FTC teams 17](#_Toc2183831)

[FRC team 18](#_Toc2183832)

[Advanced topics 18](#_Toc2183833)

[Switching between ‘accounts’ 18](#_Toc2183834)

[Username and password wrong 18](#_Toc2183835)

[Persistent username/password 18](#_Toc2183836)

[Forcing your changes to github 19](#_Toc2183837)

[Writing testable code 19](#_Toc2183838)

[Adding test ‘commands’ to Smart Dashboard 19](#_Toc2183839)

# Mechanical build notes

1. A sharpie is NOT the correct marker to mark lengths, holes etc…!!! Use either a scribe or a propelling pencil (or a very sharp regular pencil/marker).
2. Use a square for all markings. Scribe clearly the entire width of the material to be cut.
3. Plan all pieces before making a single cut.
4. Check all measurements again before cutting/drilling.
5. Make sure to account for blade kerf (blade thickness) when marking multiple sections on one segment. Preferred method is to mark and cut one at a time to ensure exact kerf size is accounted for.
6. Check all measurements again before cutting/drilling.
7. When cutting make sure the blade is completely on the correct side of the length measurement markings. Do this by…
   1. making sure the saw is unplugged and turned OFF.
   2. bring the blade down to contact material to be cut.
   3. move the material so that the edge of the blade aligns with the scrap side of the measurement mark.
   4. Check all measurements again before cutting/drilling.
8. Punch all holes before drilling.
9. Check all measurements again before cutting/drilling.
10. Drill all holes with a pilot hole before final drill.

# OpenCV Vision with Network tables on RasPi

## Setup

<https://wpilib.screenstepslive.com/s/currentCS/m/85074> WPILib setup

<https://github.com/wpilibsuite/FRCVision-pi-gen/releases> Pre-built image

<https://github.com/oscarrobotics/VisionOnPi2016>

<https://www.youtube.com/watch?v=ZNIlhVzC-4g> \*\*

<https://github.com/wpilibsuite/FRCVision-pi-gen> Generate FRCVision RasPi image

## Misc links

<https://medium.com/@rosbots/ready-to-use-image-raspbian-stretch-ros-opencv-324d6f8dcd96>

<https://www.pyimagesearch.com/2018/09/26/install-opencv-4-on-your-raspberry-pi/>

<https://www.pyimagesearch.com/2015/12/14/installing-opencv-on-your-raspberry-pi-zero/>

<https://tutorial.cytron.io/2017/08/16/raspberry-pi-zero-w-pi-camera-application/>

<https://www.hackster.io/phfbertoleti/easily-compiling-opencv-in-raspberry-pi-178e3a>

## OpenCV

<https://docs.opencv.org/master/>

<https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/hough_lines/hough_lines.html>

<https://www.geeksforgeeks.org/line-detection-python-opencv-houghline-method/>

<https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_feature2d/py_features_harris/py_features_harris.html> - corner detection

<https://pysource.com/2018/03/07/lines-detection-with-hough-transform-opencv-3-4-with-python-3-tutorial-21/>

<https://docs.opencv.org/3.0-beta/modules/line_descriptor/doc/tutorial.html>

<https://docs.opencv.org/3.4/dc/ddd/group__line__descriptor.html>

<https://stackoverflow.com/questions/45322630/how-to-detect-lines-in-opencv>

<https://stackoverflow.com/questions/52816097/line-detection-with-opencv-python-and-hough-transform>

<https://stackoverflow.com/questions/14184147/detect-lines-opencv-in-object>

<http://felix.abecassis.me/2011/09/opencv-morphological-skeleton/> -Reduce to thin line

<https://stackoverflow.com/questions/16665742/a-good-approach-for-detecting-lines-in-an-image>

<https://stackoverflow.com/questions/47389128/opencv-houghline-only-detect-one-line-in-image>

<https://stackoverflow.com/questions/42153379/detecting-line-by-color-using-opencv-in-python>

<https://stackoverflow.com/questions/49993616/multiple-line-detection-in-houghlinesp-opencv-function>

<https://medium.com/@mrhwick/simple-lane-detection-with-opencv-bfeb6ae54ec0>

<http://www.robindavid.fr/opencv-tutorial/chapter5-line-edge-and-contours-detection.html>

<http://www.aishack.in/tutorials/hough-transform-basics/>

### Draw on image

<https://stackoverflow.com/questions/18632276/how-to-draw-a-line-on-an-image-in-opencv/18633964>

### Trig

<http://geomalgorithms.com/a02-_lines.html>

## Recommendations

Configure PI to use RAM, not SDCard.

Make FS read only if possible. (<http://hallard.me/raspberry-pi-read-only>, <https://learn.adafruit.com/read-only-raspberry-pi/overview>)

Shutdown gracefully.

Backup SD card.

Make sure Pi booted and running before access.

Good power supply.

Ensure force HDMI set on

## Many guides and articles for Python & image processing

<https://www.pyimagesearch.com/>

## Vision processing on desktop for simulation and network tables

<https://www.youtube.com/watch?v=QIGn90hKnv8>

## Pi Vision including setup, project build and deploy.

Don’t forget to change the hostname!!

Part 4 pulls everything together including an application pushing data to network tables.

Part 5 is code analysis.

Part 6 - GRIP

Part 7 - Debugging

<https://github.com/Team997Coders/BB2018BallFindingVision>

<https://www.youtube.com/watch?v=Zgt2vMSxNbs>

# GRIP vision

<https://wpilib.screenstepslive.com/s/currentCS/m/vision/l/463566-introduction-to-grip>

<https://wpilib.screenstepslive.com/s/currentCS/m/vision/l/672730-generating-code-from-grip>

<https://wpiroboticsprojects.github.io/GRIP/#/>

# Vision/Network table examples

<https://github.com/frc4646/frc4646-2016-competition-code/blob/master/src/Subsystems/VisionCalculation.cpp>

<https://github.com/oscarrobotics/VisionOnPi2016>

<https://github.com/GarnetSquadron4901/rpi-vision-processing/blob/master/wait_for_shutdown.py>

<https://github.com/frc5687/pi-tracker>

<https://github.com/robotpy/pynetworktables/tree/master/samples>

# OpenCV on RoboRIO

<https://wpilib.screenstepslive.com/s/currentCS/m/vision/l/669166-using-the-cameraserver-on-the-roborio>

<https://wpilib.screenstepslive.com/s/currentCS/m/vision>

<http://first.wpi.edu/FRC/roborio/release/docs/java/edu/wpi/first/vision/package-summary.html>

<https://github.com/Team2168/2168_Vision_Example>

<https://github.com/robotpy/roborio-opencv>

<https://github.com/WPIRoboticsProjects/opencv-installer>

<https://wpilib.screenstepslive.com/s/currentCS/m/vision/l/682117-strategies-for-vision-programming>

<https://usfirst.collab.net/sf/frs/do/viewRelease/projects.wpilib/frs.sample_programs.2017_c_java_vision_sample?_message=1483834990405>

# FRC networking

## FRC networking basics

<https://wpilib.screenstepslive.com/s/4485/m/13503/l/696075-networking-basics>

<https://wpilib.screenstepslive.com/s/4485/m/24193/l/319135?data-resolve-url=true&data-manual-id=24193>

<https://wpilib.screenstepslive.com/s/currentCS/m/troubleshooting/l/319135-ip-networking-at-the-event>

<https://www.chiefdelphi.com/t/unable-to-connect-to-raspberry-pi-at-competition/157559>

## Change RasPi mDNS name

<https://www.howtogeek.com/167195/how-to-change-your-raspberry-pi-or-other-linux-devices-hostname/>

ex. roboRIO-TEAM-FRC.local

ex. Vision-TEAM-FRC.local

Change RasPi to static IP

<https://www.modmypi.com/blog/how-to-give-your-raspberry-pi-a-static-ip-address-update>

This is my regular advice every time this issue comes up.

Move to Static IP’s

DS 10.TE.AM.5

RoboRio 10.TE.AM.2

RPi 10.TE.AM.10 (Although .11 works too.)

DS 10.TE.AM.5 with mask= 255.0.0.0 (MANDATORY)

RoboRio 10.TE.AM.2 with mask 255.0.0.0 (to ensure lan-ARP results get resolved every time)

RPi 10.TE.AM.10 (/8 mask suggest for consistency)

RPi2 10.TE.AM.11 (/8 mask suggest for consistency)

## Troubleshooting

The most common issue is to have a mix of static and DHCP configured devices. This should be less problematic with the 2018 configuration, but should still be avoided.

Another common issue is using a subnet mask of 255.255.255.0 on the DS PC. This configuration will not communicate with the FMS system which is on a 10.0.100 address.

## FRC IP networking at events

<https://wpilib.screenstepslive.com/s/4485/m/24193/l/319135?data-resolve-url=true&data-manual-id=24193>

# Network tables

DON’T USE MDNS !!! USE STATIC IP!!!

## Using network tables

<https://robotpy.readthedocs.io/en/latest/guide/nt.html>

<https://pynetworktables.readthedocs.io/en/latest/>

## Misc notes

NOTE : Network tables are SLOW!!! Try using setUpdateRate(); for 10mS instead of 100mS?

<https://github.com/frc5687/pi-tracker> Network tables and UDP examples

<https://github.com/frc5687/2016-Outlier2/tree/auto/%23198-PoseHistory>

<https://robotpy.readthedocs.io/en/latest/guide/nt.html#networktables-guide> \*\*

<https://github.com/robotpy/pynetworktables/tree/master/samples>

<https://www.chiefdelphi.com/t/network-tables-and-the-raspberry-pi/153462>

<https://pynetworktables.readthedocs.io/en/latest/>

<https://wpilib.screenstepslive.com/s/currentCS/m/vision/l/479908-reading-array-values-published-by-networktables>

<https://wpilib.screenstepslive.com/s/3120/m/7912/l/80205-writing-a-simple-networktables-program-in-c-and-java-with-a-java-client-pc-side>

### LOOK AT SHUFFLEBOARD !!! It can display tons of info including network tables

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/814689-tour-of-shuffleboard> \*\*

### Data recording & playback

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/1021944-controlling-data-recording>

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/822285-using-record-and-playback>

### Displaying camera

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/831042-displaying-camera-streams>

### Graphs

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/822288-working-with-graphs>

### Displaying command based state information

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/831050-working-with-commands-and-subsystems>

# UDP messaging

<http://einsteiniumstudios.com/using-the-roborio-with-the-beaglebone.html>

<https://www.baeldung.com/udp-in-java>

<https://systembash.com/a-simple-java-udp-server-and-udp-client/>

<https://stackoverflow.com/questions/10556829/sending-and-receiving-udp-packets-using-java>

<https://wiki.python.org/moin/UdpCommunication>

# TCP messaging

<https://www.chiefdelphi.com/t/frc-java-tcp-client/138061/12>

# PID tuners etc…

<https://wpilib.screenstepslive.com/s/currentCS/m/shuffleboard/l/831044-testing-and-tuning-pid-loops>

use setPIDTarget();

<https://wpilib.screenstepslive.com/s/currentCS/m/smartdashboard/l/255413-pid-tuning-with-smartdashboard>

Good video

<https://www.youtube.com/watch?v=yqD9iHiR3j8>

Team 2168 video series PID tuning

<http://team2168.org/index.php/resources/programming/217-pid-control-tutorial>

<https://www.youtube.com/watch?v=KBh54PAvoxs>

Team 5584 PID tuning guides

Talk about closed loop velocity control. Also follow suggested CTRE Talon software guide instructions (section 12.4)

<http://www.versiontree.com/icrobotics/first/123-pid-tuning-motor-control>

Check CTRE Talon software user guide for examples on tuning both speed and position.

# Misc stuff

## Start here ...

<https://phoenix-documentation.readthedocs.io/en/latest/ch01_PhoeSoftRefManual.html>

READ ALL CHAPTERS!!!

Chapter 16 is where closed loop motor control is introduced. READ ALL PREVIOUS CHAPTERS THOUGH!!! READ THE ENTIRE SECTION BEFORE 'PLAYING AROUND' !!!

Read the FAQs here

<https://phoenix-documentation.readthedocs.io/en/latest/ch20_FAQ.html>

## Great article on programming for FRC. READ THIS!!!

<https://media.readthedocs.org/pdf/frc-pdr/latest/frc-pdr.pdf>

See 7.4.3 Cascade Elevator advice

Article also has great section on scouting and "Introduction to Data Analysis"

Chapter 12 covers OpenCV !!!

## Systems as PID input

To use vision (or any other 'sensor' system as an input to a PID see this article... <https://wpilib.screenstepslive.com/s/3120/m/7912/l/79828-operating-the-robot-with-feedback-from-sensors-pid-control>

## Talon resources

<http://www.ctr-electronics.com/talon-srx.html#product_tabs_technical_resources>

\*\* note about 2019 firmware !!! Also has motion profile generator

\*\*\*\*\*\*\*Check the "FOLLOW THESE INSTRUCTIONS" section !!!

<https://phoenix-documentation.readthedocs.io/en/latest/index.html>

Example code

<https://github.com/CrossTheRoadElec/Phoenix-Examples-Languages>

Very simple Test Drive of robot using Talons

<https://phoenix-documentation.readthedocs.io/en/latest/ch13_MC.html>

Note, example is only single channel. Tank requires 2

## Misc notes. READ ME !

### Make sure Rio is imaged correctly for 2019

<https://phoenix-documentation.readthedocs.io/en/latest/ch05_PrepWorkstation.html>

### Understand the new Phoenix tuner application

(<https://phoenix-documentation.readthedocs.io/en/latest/ch03_PrimerPhoenixSoft.html>

### New project test example

<https://phoenix-documentation.readthedocs.io/en/latest/ch05a_CppJava.html>

Read this, there is a lot of good info about the new VS environment. Note "ControlMode" parameter !!! READ EVERYTHING. This section goes on to describe configuring the hardware, which is also necessary!!!

### Quadrature and limit switch sensor testing

<https://phoenix-documentation.readthedocs.io/en/latest/ch12_BringUpCANifier.html>

### Sensor setup and testing

<https://phoenix-documentation.readthedocs.io/en/latest/ch14_MCSensor.html>

### Use the plot feature to 'see' things happen :)

\*\*Before you enable the DS, spin the Joystick axis so it reaches the X and Y extremities are reached. USB Gamepads calibrate on-the-fly so if the Gamepad was just inserted into the DS, it likely has not auto detected the max mechanical range of the sticks.

Reset the motor controllers then DOCUMENT EXACTLY how each controller is configured for each position on the robot.

<https://phoenix-documentation.readthedocs.io/en/latest/ch13_MC.html>

New in 2019 is the ability to set all these parameters from software. This is recommended to ensure the controllers are really configured correctly, just in case the controller has been replaced/swapped etc...

A general recommendation is to:

Configure all devices during robot-bootup using the API,

Use Tuner to dial values quickly during testing/calibration.

Export the settings so they are not lost.

Update your software config values so that Tuner is no longer necessary.

## Motors, testing and calibration, READ ME !

<https://phoenix-documentation.readthedocs.io/en/latest/ch13_MC.html>

Send debug info to the console with e.g. System.out.println("stick:" + stick);

### Almost complete example of motor, sensor, drive, display info etc... here

<https://phoenix-documentation.readthedocs.io/en/latest/ch14_MCSensor.html>

### Motor 'followers' are described here

<https://phoenix-documentation.readthedocs.io/en/latest/ch13_MC.html>

SetNeutralMode() can be different on follower motors to give partial braking.

### "Ramping" is discussed here

<https://phoenix-documentation.readthedocs.io/en/latest/ch13_MC.html>

This is CRITICAL for smooth driving !!! configOpenLoopRampRate & configClosedLoopRampRate. See also promoting of low settings. Especially useful for motors to ensure minimum drive. Deadband also plays into this.

Consider "enableVoltageCompensation" if we see variations in performance.

### Can read the following at any time...

Quadrature Encoder Position, Velocity, Index Rise Count, Pin States (A, B, Index)

Analog-In Position, Analog-In Velocity, 10bit ADC Value,

Battery Voltage, Current, Temperature

Fault states, sticky fault states,

Limit switch pin states

Applied Throttle (duty cycle) regardless of control mode.

Applied Control mode: Voltage % (duty-cycle), Position/Velocity closed-loop, or slave follower.

Brake State (coast vs brake)

Closed-Loop Error, the difference between closed-loop set point and actual position/velocity.

Sensor Position and Velocity, the signed output of the selected Feedback device (robot must select a Feedback device, or rely on default setting of Quadrature Encoder).

We can use "Soft Limits" on the lifter to control the max/min position when under driver control and not a pre-set position. "Sensor Phase" is critical here.

### Sensor checking/debug

<https://phoenix-documentation.readthedocs.io/en/latest/ch14_MCSensor.html>

We might need to change the sensor sample window and/or rolling average window size if motors are moving fast

<https://phoenix-documentation.readthedocs.io/en/latest/ch14_MCSensor.html>

### See "Recommended Procedure"

We probably want to use "Position Closed-Loop Control Mode" for the lifter. Start here for the tuning procedure

<https://phoenix-documentation.readthedocs.io/en/latest/ch16_ClosedLoop.html>

### "Motion Magic Control Mode" might be better for the lift, but not 100% sure yet

We MIGHT want to use "Velocity Closed-Loop Control Mode" for the drive train, but I don't think it is necessary. Ramp mode should give smooth control. Velocity mode would be good for a distance shooter (frisbee, 'steam' etc....)

"Motion Profile Control Mode" might be useful for 'getting close' in auto mode, then use vision to home in, then profile again to move back to the hatch pickup etc...

### Make sure we understand how to check, analyze and clear faults

<https://phoenix-documentation.readthedocs.io/en/latest/ch17_Faults.html>

### Examples exists here for the following...

<https://github.com/CrossTheRoadElec/Phoenix-Examples-Languages/tree/master/Java>

DriveStraight\_AuxQuadrature - Drive straight based on encoder

DriveStraight\_AuxPigeon

\*\*Label the devices/controllers appropriately so there is no guessing which device ID is what.

# Motion profile generator and example code

<https://github.com/vannaka/Motion_Profile_Generator/releases>

Interesting post about PID and arms, to account for gravity. Might help FTC Mechanicats

Speed controlled object

<https://wpilib.screenstepslive.com/s/currentCS/m/java/l/599702-driving-motors-with-pwm-speed-controller-objects>

<https://www.chiefdelphi.com/t/speed-control-with-talon-srx-and-encoder/149271>

Lift will use motion magic. Make sure to read the Talon SRM motion magic control section, especially about setting F parameter correctly!!! “Motion Magic Closed-Loop Walkthrough”!!!

Auto MIGHT use Motion Profile mode . Read Motion Profile Reference Manual. Check the complete example including display feedback in section 6.7.2.

# Motion Profiling

<https://github.com/juchong/Motion_Profile_Generator>

## Spline fitting

<https://github.com/Team254/TrajectoryLib>

<https://www.chiefdelphi.com/t/pic-parametric-quintic-spline-trajectory/178825> (Nice images)

## Motion profiling article

<https://www.chiefdelphi.com/t/motion-profiling/115133>

## Motion planning video presentation

<https://www.youtube.com/watch?v=8319J1BEHwM>

<https://docs.google.com/presentation/d/1xjtQ5m3Ay4AYxS_SfloF2n_vWZnCU25aXZuu9A59xPY/pub?start=false&loop=false&delayms=3000#slide=id.g76b62f478_0_111>

See ~12m for explanation of PID.

~15m for motion profiling

~35:30 for ‘following the trajectory’ implementation.

~38:30 talks about tuning the PID

~42 Talks about not even needing PID for trajectory following

~42 Tuning methodology

Set loops to 200Hz (5ms) but make sure accurate. Rio might not be accurate. Use getFPGATimestamp() to check.

Use vision ONLY to calculate goals, use higher update rate sensors in the control loops.

Team 254 has released pre-computed and on the fly code. Also web server to show information real time.

# Custom PID sources

<https://wpilib.screenstepslive.com/s/currentCS/m/java/l/599721-operating-the-robot-with-feedback-from-sensors-pid-control>

configSelectedFeedbackSensor seems to be used to supply non-integrated sources for PID control. Not 100% sure though.

# Good Talon blog with simulation environment

<https://www.systemvision.com/blog/controlling-motors-talon-srx-february-1-2017>

Good blog on simulating for FRC

<https://www.systemvision.com/blog/first-robotics-frc-motor-modeling-may-6-2016>

# Example motion profile from SteamWorks

<https://github.com/CrossTheRoadElec/FRC-Examples-STEAMWORKS/blob/master/JAVA_MotionProfileExample/src/org/usfirst/frc/team3539/robot/GeneratedMotionProfile.java>

# Useful function for periodic message display…



# Itterative vs Timed vs Command

<https://wpilib.screenstepslive.com/s/currentCS/m/java/l/599697-choosing-a-base-class>

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241900-simple-subsystems>

## Command based joystick control

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241902-creating-simple-commands>

## Groups of commands

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241903-creating-groups-of-commands>

## Running commands whilst button pressed or held down.

Specifically, see “whileHeld” & “cancelWhenPressed”

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241904-running-commands-on-joystick-input>

## Using commands during auto & teleop

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241905-running-commands-during-the-autonomous-period>

## Default/auto switching between joystick and commands

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241907-default-commands>

## Synchronizing commands

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241908-synchronizing-two-commands>

## Limit switches and commands

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/241909-using-limit-switches-to-control-behavior>

## High level overview

<https://wpilib.screenstepslive.com/s/currentCS/m/cpp/l/277232-scheduling-commands>

# Team 5940 code using PathWeaver

<https://github.com/BREAD5940/frc-java-command-codebase>

Check \frc-java-command-codebase\src\main\java\frc\robot\commands\auto\actions for detailed code for both path following and motion profile following.

# Mechanisms, parts and components

## Single articulated mechs

<https://www.youtube.com/watch?v=B1bLacxONlY>

<https://www.youtube.com/watch?v=5SRyYz-tFxQ> \*\* Motor mount info

## Cascade lifter mechs

<https://www.youtube.com/watch?v=wZ6a6dc4BGg>

## Bearing options

<https://www.youtube.com/watch?v=G_HG1_oCbXk>

## Vex versa-blocks

<https://www.vexrobotics.com/bearingblocks.html>

<https://www.idesignsol.com/217-5852>

<http://www.wcproducts.net/217-3436>

<http://www.wcproducts.net/217-3634/>

<http://www.wcproducts.net/217-4155>

## Linear actuator

<http://dartactuators.com/>

## Competition robot parts (limited and expensive!!)

<https://www.competitionrobotparts.com/>

## Interesting COTS discussion

<https://www.chiefdelphi.com/t/greyt-universal-cascade-elevator-and-powercube-claw/162345/59>

## CUI encoders

<https://www.digikey.com/product-detail/en/AMT102-V/102-1307-ND/827015>

# CheesyVision - Huhh!!! Is this even leagal ???

Apparently since they got an award for it !!!

<https://www.chiefdelphi.com/t/team-254-presents-cheesyvision/136529>

<https://github.com/Team254/CheesyVision>

# Git software revision control and software management

## Make sure Git is installed on the computer you want to develop on

1. Open a command prompt
2. Type “git”. If you get a message saying “’git’ is not recognized as an internal or ….” Message then you need to install git
   1. In a web browser navigate to <https://git-scm.com/download/win>
   2. Follow the instructions to install git
   3. Once installed go back to 1)

## Make sure you have an account on GitHub

You need an account at [www.githib.com](http://www.githib.com) to be able to make changes to the items stored on GitHub. Once you have an account the project leader for the particular project will need to give you permission to be able to ‘push’ changes. Initially there will be a ‘team’ account but ultimately each member should have their own account so it is easy to track who made what changes ☺

Miscellaneous repositories are available here…

<https://github.com/WoodrowRobocats>

There are multiple repositories for different projects such as the PiArcade, Button Box controller, test code, robot code etc…

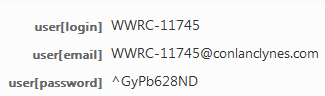
The initial FRC account details are shown below…



<https://github.com/WWRC-FRC>

<https://github.com/WWRC-FRC/2019-Main-Code.git>

FTC teams currently have only their current robot code repositories as outlined below.



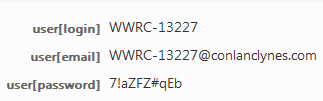
<https://github.com/WWRC-11745>

<https://github.com/WWRC-11745/FTC-2018-2019.git>



<https://github.com/WWRC-11761>

<https://github.com/WWRC-11761/FTC-2018-2019.git>



<https://github.com/WWRC-13227>

<https://github.com/WWRC-13227/FTC-2018-2019.git>

## Creating a new Git repository from existing code on your computer

If you have a project on your computer that you want to start tracking with git and storing in the github cloud you need to create a repository on github and also a local repository, then link them together.

1. Log in to the correct github account at <https://github.com/> for your team.
2. Create a new repository on GitHub for a new project with no files (i.e. do NOT include readme.md). Make a note of the URL name given as the repository URL for use later on.
3. On computer containing the project…
   1. Open command prompt in the directory you wish to store on Github. You can easily do this in windows by navigating to the directory with Windows Explorer, then typing “cmd” in the address bar. You should then have a DOS command window open where you can type commands.
   2. Type ‘git init’ This creates a local ‘repository’ to keep track of your changes.
   3. Type ‘git add .’ This adds all files from the current directory to the repository staging area.
   4. Type ‘git commit -m "Initial commit"’. This actually stores the files in a git repository.
   5. Type ‘git remote add origin *URL*’ where URL is the one given when you created the repository above. This links the repository to GitHub.  
      e.g. ‘git remote add origin https://github.com/WWRC-FRC/Documents.git’
   6. ‘git push -u origin master’ This will store, or ‘push’ all your files in the cloud on github.   
      NOTE: You will possibly/probably be asked to enter the username and password for the github account. Enter your personal details if you have a personal account, or the team details from above. If you have issues with your password see later “Username and password wrong”
   7. Summary…  
      git init  
      git add .  
      git commit -m "Initial commit"  
      git remote add origin *URL*git push -u origin master

## ‘Cloning’ code to your computer from a GitHub repository

If you want to work on code on your computer at home, or basically anywhere you will need to initially ‘clone’ the code from github to the computer you want to work on. This is easily done with the following procedure

1. Create a directory on your computer where you want to put the code.
2. Open a command prompt in the directory (see above for instructions).
3. Type ‘git clone *URL*’ where URL is the source URL for the github repository you want to ‘clone’. You can find this easily by opening the github web page for the project you want to clone and clicking on the  button which will then give you a URL. Do not use the “Open in Desktop” or “Download ZIP” options.

## Making sure you have the latest code

Before editing code which is under git control you should really make sure that other people have not been making changes. To do this you should ‘pull’ the latest code before starting to work on code. Note, if you start working on code before ‘pulling’ then you can still run these commands to merge your changes with those already on github. Take care with this though, multiple people should not be working on the exact same functionality without close collaboration to make sure you don’t break each other’s code.

### FTC teams

1. Open a command window in the directory containing your code
2. Type ‘git pull’. This will download any changes from the github storage and merge them with your local files. If you have made any local changes then all will be merged together.

### FRC teams

1. Select the Source Control icon  
   
2. Click on the … to bring up the git menu.  
   
3. Then click “pull”. This will download any changes from the github storage and merge them with your local files. If you have made any local changes then all will be merged together.

## Checking in changes and pushing to Github

Whenever you make changes to your code that you want to “commit” to local source control and “push” to the cloud storage you use the following operations…

### FTC teams

1. Open a command window in the directory containing your code
2. If you have added any new files then type ‘git add .’
3. Type ‘git commit .’
4. Type ‘git push’

### FRC team

1. Select the Source Control icon  
   
2. The side panel should then show files which are different to ones currently checked in. In the box that says “Message” type something meaningful to describe what changes you made. This can be as long as you need to accurately describe everything. Once finished hit CTRL+Enter to ‘commit’ your changes. After this the filenames you changed should then be cleared from the list. At this point in time you have made a copy of your files and stored them locally on the computer (‘committed’ them). You can go back to this version of the files at any time should you need to. This is also why descriptive messages are critical since otherwise you won’t know why these files were stored in this current state.
3. Finally we are going to ‘push’ the files to github so that if the laptop is damaged/lost/stolen then you can recover your work. Click on the … to bring up the git menu.  
   
4. Then click “push”. This should then push your changes to github.

## Advanced topics

### Switching between ‘accounts’

This is an advanced topic ! Sometimes it is desirable to switch between different user accounts. In theory you can use the following git command to switch to a different identity.

git remote set-url origin <https://USERNAME@github.com/USERNAME/PROJECTNAME.git>

### Username and password wrong

Sometimes it is necessary to do the following git commands to ‘forget’ the current user/password. This usually only happens if multiple usernames are being accessed.

git config --global --unset-all credential.helper

git config --unset-all credential.helper

git config --system --unset credential.helper

You might also need to delete git from Windows’s ‘Credentials manager’. In the Windows search box type “credential” and you should see the credentials manager appear. Select this then delete anything github related in the generic credentials section.

### Persistent username/password

You can make your username/password persistent for a period of time using the following command…

git config --global credential.helper 'cache --timeout 7200'

This will set the time period to 7200 seconds before you need to re-enter your username/password.

### Forcing your changes to github

If for some reason you get errors when you try to push your changes to github and you get messages talking about “failed to push some refs” then you can force your version of life, the universe and everything out to github with the following command…

‘git push --force -u origin master’

Note though that this is VERY bad practice. Don’t do this unless you understand exactly what the consequences are.

The most common occurrence of this error is if you have deleted a file locally that still exists on github. In this situation make sure the deleted file really can be deleted.

There are better ways to handle deleted files which will be outlined at a later date.

# Writing testable code

Writing testable code will allow us to, well, test, our code without needing a robot. This is great for checking sequences of events occur correctly, that calculations are correct etc…

This requires code to be written in a way that allows it to be testable though.

For example, a project created from the FRC templates immediately instantiates real hardware objects such as motor controllers. At this point in time the code tries to talk to the hardware. Obviously this is an issue if we don’t actually have any hardware connected yet, ot worse still, if we are simulating on a desktop!!!

The solution is to structure our code so that hardware objects are only ‘constructed’ when running on the real robot. When not on the robot we will create simulation models for things that need them, and ‘wrapper’ code that needs to interact with real hardware.

Good slides on what/why/how testing is good are here…

<https://speakerdeck.com/frc4931/writing-testable-robot-code-for-frc>

Excellent video on automated testing. This goes way beyond what we need though.

<https://www.youtube.com/watch?v=vmRFiF9hd2E>

You need this VSCode plugin installed: <https://www.youtube.com/redirect?event=video_description&v=vmRFiF9hd2E&redir_token=HVMmWucIUSGTg1OhSYVRJnWe4n58MTU1MTM4MjM1NUAxNTUxMjk1OTU1&q=https%3A%2F%2Fmarketplace.visualstudio.com%2Fitems%3FitemName%3Dvscjava.vscode-java-test>

Check at 4:00 in the video for info on how to construct objects.

Check at 9:00 for info on Mockito

Check at 13:00+ish for some wpilib requirements to stop exceptions and other requirements.

Other videos from Chuck Benedict from Team 997

<https://www.youtube.com/watch?v=iyifKF1jocI> – Excellent intro to testability – Untestable Command Robot

<https://www.youtube.com/watch?v=rbSPkhAgLk0> – Testable Command Robot

<https://www.youtube.com/watch?v=Dq8Lc6wJkGM>

## Adding test ‘commands’ to Smart Dashboard

<https://wpilib.screenstepslive.com/s/3120/m/8564/l/88513-how-to-write-an-easy-to-test-robot-program>