Software Specification - Warbots Team 620

Version 4

Table of Contents

[1. Roster 4](#_Toc802915)

[2. Change Log 4](#_Toc802916)

[3. Mission Requirements 5](#_Toc802917)

[3.1. Mission Requirements 5](#_Toc802918)

[3.2. Scenarios for Scoring 5](#_Toc802919)

[4. System Requirements 5](#_Toc802920)

[5. Subsystem Requirements 6](#_Toc802921)

[5.1. Drive Train 6](#_Toc802922)

[5.2. Up/Down Mechanism 6](#_Toc802923)

[5.3. Driver Interface 6](#_Toc802924)

[5.4. Tracking System Requirements 7](#_Toc802925)

[5.5. Alignment System Requirements 7](#_Toc802926)

[6. Use Cases 7](#_Toc802927)

[7. System Architecture 8](#_Toc802928)

[7.1. System Architecture 8](#_Toc802929)

[7.2. Layers 8](#_Toc802930)

[7.3. Interfaces 9](#_Toc802931)

[7.3.1. Raspberry Pi 9](#_Toc802932)

[7.4. Functions 9](#_Toc802933)

[7.5. Subsystems 9](#_Toc802934)

[8. Design 11](#_Toc802935)

[8.1. Tracking System 11](#_Toc802936)

[8.2. DriveTrain 11](#_Toc802937)

[8.3. Hatch Grab System - preliminary 12](#_Toc802938)

[8.4. Cargo System - preliminary 12](#_Toc802939)

[8.5. Up/Down Mech - preliminary 12](#_Toc802940)

[9. Patterns 12](#_Toc802941)

[10. Robot Configuration 12](#_Toc802942)

[10.1. User Settings 12](#_Toc802943)

[10.1.1. Driver 12](#_Toc802944)

[10.1.2. Co-driver 13](#_Toc802945)

[10.2. Robot Settings 13](#_Toc802946)

[11. Sensors and Actuators 13](#_Toc802947)

# 

# 1. Roster

* Zachary Carlini
* Chase Bowen
* Ethan Wolman
* Keiran Khan
* Sam Parker
* Chamin Lim
* Bryan Morillo
* Steven Le
* Sareena Miley
* Grace Kim
* Audrey Zhang
* Tommy Nguyen
* Alex Gemond

# 2. Change Log

01/09/19 Initial Draft - Zachary

01/10/19 Added Layers and Use Cases - Zachary

01/12/19 Added Tracking and Alignment Subsystems - Zachary

01/16/19 Consolidated Subsystems and added methods - Zachary

01/19/19 Version 2 - Zachary

01/24/19 Ordered Categories and Sections - Chamin

01/29/19 Version 3 - Chamin

02/04/19 Version 4 and added diagrams - Chamin

02/05/19 Added Robot Configurations

# 3. Mission Requirements

## 3.1. Mission Requirements

1. Shall be able to control every subsystem
2. Shall have a 12 second round trip for delivering the cargo and hatches
3. Shall deliver 2 hatches during the 15-second sandstorm period
4. Shall jump down from the 6-inch platform to start the sandstorm
5. Shall be able to climb the tallest HAB platform in 10 seconds
6. Shall be able to pick up hatches from the ground in 1 second at 10 feet per second
7. Shall be able to deliver hatches and cargo to every port
8. Shall be able to control the robot autonomously during the sandstorm and with operator control during the rest of the match
9. The robot will be able to operate in "aim-bot" mode to assist the driver with game object delivery
10. Shall be able to allow the driver to configure variables without recompiling

## 3.2. Scenarios for Scoring

1. Load a Hatch from Loading Station (Load and automatically back away))
2. Load a Hatch from the floor at speed (Prepare/Pickup)
3. Load a Ball from Loading Station (Load and wait to back away)
4. Load a Ball from the floor at speed (Prepare/Pickup)
5. Load a Ball from the floor in Corral
6. Score a Hatch at Rocket Level (1|2|3)
7. Score a Ball at Rocket Level (1|2|3)
8. Score a Ball in a specific side of a rocket
9. Score a Ball at Cargo Ship
10. Climb HAB (L2 or L3)

# 4. System Requirements

1. Shall be able to control drive train
2. Shall be able to control up/down mechanism
3. Shall be able to control the vision tracking system
4. Get a robot to be able to accelerate 4 feet per second to a maximum velocity of 10 feet per second
5. Get a robot to be able to turn 180 degrees in 2 seconds
6. Travel 15 feet in a straight line
7. Shall be able to operate code when the field changes
8. Shall be able to reach the ball port with the up/down mechanism in 3 seconds
9. Shall be able to launch the ball into the ball port with a vertical accuracy of 1 inch
10. Shall be able to align with tape with an accuracy of 1 inch

# 5. Subsystem Requirements

## 5.1. Drive Train

1. Shall be able to control the drive train
2. Shall be able to accelerate at a rate of at least 4 fps^2
3. Shall be able to have an end velocity of 10 fps
4. Shall be able to assist the driver in driving straight forward with a tolerance of one degree per 15 feet TBR
5. Shall be able to turn 180 degrees in one second with an acceleration and deceleration of 90 degrees per second^2 autonomously or in teleoperate mode
6. Shall be able to turn 90 degrees in one second with an acceleration and deceleration of 45 degrees per half second ^ 2 during autonomous or teleoperate mode
7. Shall be able to align with the tape with an accuracy of one inch with the capability to autocorrect

## 5.2. Up/Down Mechanism

1. Shall be able to reach every port on the rocket ship
2. Shall be able to command particular elevations
3. Shall be able to raise the up/down mech from the lowest position to the highest one in 4 seconds
4. Shall be able to retrieve a hatch or a cargo ball from the loading station
5. Shall be able to pick up a hatch or cargo ball from the ground
6. Shall be able to deposit a hatch or cargo ball into a port
7. Shall be able to lower the up/down mech from the highest position to the lowest one in 4 seconds
8. Shall be able to reach every horizontal distance from the center of the robot?

## 5.3. Driver Interface

1. Shall be able to adjust all key variables without recompiling, including:
   1. Driving dead zones
   2. Driving Sensitivity
   3. Weapons officer dead zones
   4. Weapons officer sensitivity
2. Shall be able to communicate instructions from the controller, through the computer, to the robot
3. Shall be able to control velocity, up/down mech, tracking system direction, aimbot target locking, HAB mechanism, and rocket port level
4. Shall be able to maintain continuous communication between the robot and driver station
5. Shall be able to display information from the robot to the driver
6. Camera feed of 30 fps, 480p resolution, FOV between 40 and 120 degrees (80-100 preferred).
7. Controller vibration feedback for acceleration greater than 4 feet/second^2 TBR
8. Controller vibration feedback for entering and exiting aim-bot mode
9. Dashboard light to indicate whether the driver is currently manually controlling the robot
10. “Sensitive joystick,” precise requirement TBD
11. Automatic turning using the D-pad with:
    1. Up: 180˚ turn
    2. Left: 90˚ turn left
    3. Right: 90˚ turn right
12. Shall be able to switch control schemes without recompiling the code
13. Shall be able to support control with an Xbox controller or a Logitech gamepad.

## 5.4. Tracking System Requirements

1. The tracking system shall be able to illuminate retroreflective tape
2. The tracking system shall be able to search for and recognize illuminated retro-reflective tape from 9 feet TBR
3. The tracking system shall be able to identify individual sets of strips of retroreflective tape
4. The tracking system shall be able to identify the target port via input from the control system target
5. The tracking system shall be able to identify the angle to the tape strips’ center point to within ± 0.5°

## 5.5. Alignment System Requirements

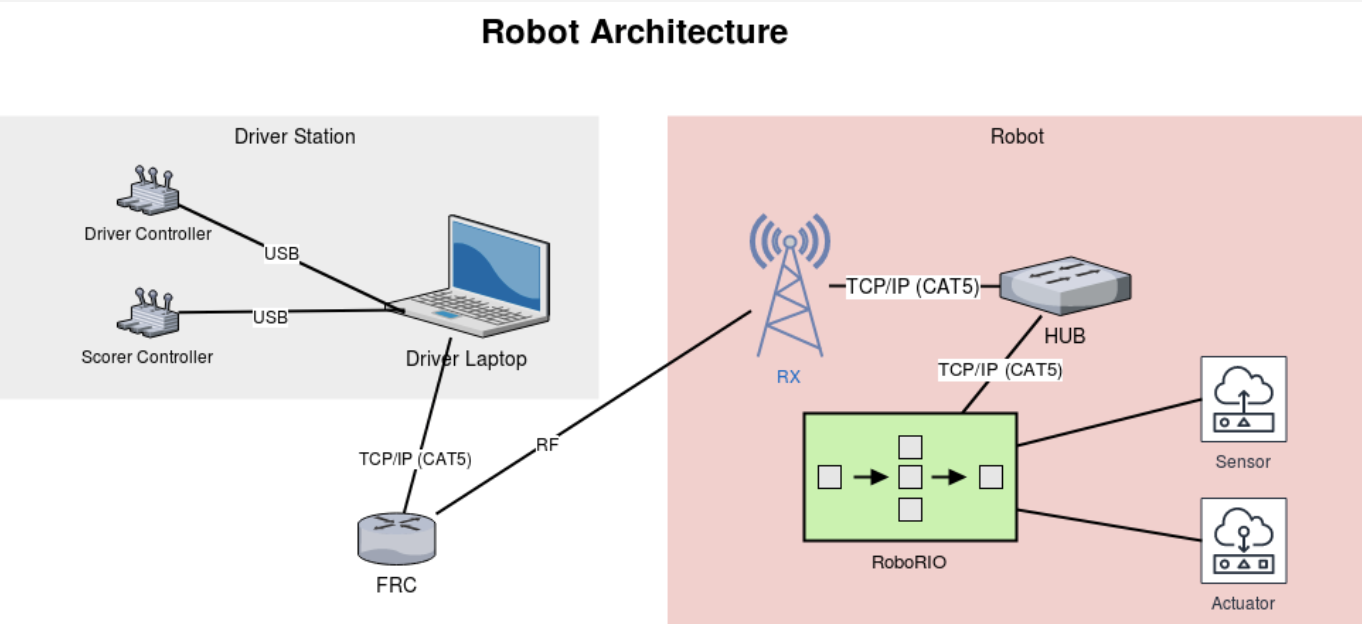
1. Shall be able to get a feed from the camera and recognize and track the angle of the white line on the ground
2. The alignment will be able to detect the line on the ground and take over from the tracking system
3. Shall be able to deliver a game object within one inch of the edge of the line (2 inches from the center)
4. Shall be able to recognize when the cargo or hatch has been dropped off, and then back up and turn around

# 6. Use Cases

1. DriveWithJoysticks
2. Exit Stations
3. Acquire target
4. Align with port
5. Deposit Hatch (Level 1|2|3)
6. Deposit Cargo (Level 1|2|3)
7. Load Hatch
8. Load Cargo
9. Pick up Cargo from Floor
10. Pick up Hatch from Floor
11. Climb Level 2
12. Climb Level 3

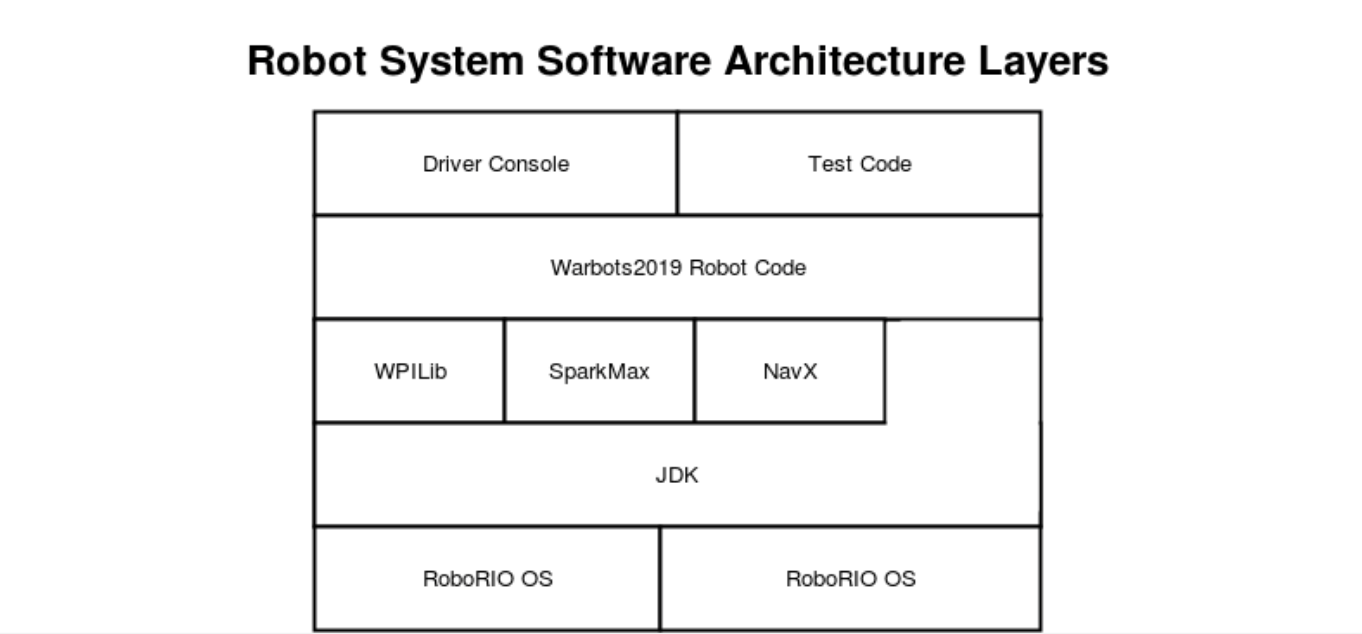
# 

# 7. System Architecture



## 7.1. System Architecture

## 7.2. Layers



1. Driver
2. Driver Station
3. Commands
4. Subsystems
   1. Drivetrain
   2. Tracking System
   3. FRC Libraries
5. Drivers
6. Robot Hardware

## 7.3. Interfaces

1. Robot/Drive Station
2. Hardware/Subsystem
3. Camera/Drivetrain
4. Robot/Scoring Mechanism
5. Drivers/Drive Station
6. Alignment System/Drivetrain
7. Vision System/Drivetrain

### 7.3.1. Raspberry Pi

## 7.4. Functions

1. Drive/Turn/Lift
2. Instructions from Driver
3. Give instructor feed from the camera
4. Storing Data Permanently
   1. Data Logging
   2. Calibration
   3. Test Results
5. Acquire Robot Position

## 7.5. Subsystems

1. Drive train
2. Alignment System
3. Up/Down Mechanism
4. Drive station
5. Motor Control
6. Drive train
   1. Wheels
   2. Motors
   3. Gears
7. Up/Down Mechanism
   1. Position Tracker
   2. Activator
   3. Grab Mechanism
8. Drive station
   1. Xbox Controller
   2. Driver
   3. Laptop
9. Driver Station Program
   1. Dashboard
10. Target Tracking
    1. Camera
    2. Reflective tape
    3. Lighting
    4. Light Relay
    5. Raspberry Pi
11. Alignment Tracking
    1. Camera
    2. White tape
    3. Raspberry Pi
    4. Gyroscope
12. Motor Control
    1. Neo-motors (x4)
    2. Spark Max (x4)
    3. Wheel Encoders

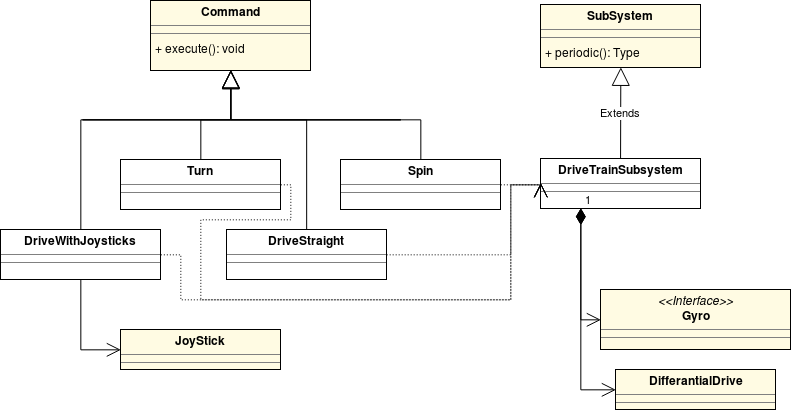
# 

# 8. Design

## 8.1. Tracking System

* getAzimuthTarget – get azimuth to the reflective target
* getDistanceTarget – get distance to reflective target
* getElevationTarget – get elevation to reflective target
* isLocked – check if the tracking system is locked on reflective target
* Lock – set tracking system lock on currently visible reflective target
* getTargetX – get X position of reflective target
* getTargetY – get Y position of reflective target
* setMode – set mode
* getAzimuthLine – get azimuth to start of a line
* getOrientationLine – get the orientation of a line
* isLineVisible – check if the line is in view
* getConfidence – get confidence factor of line in view

## 8.2. DriveTrain



* drive (vel, acc) – start driving at specified velocity and acceleration
* turn (vel, acc) – start turning at specified velocity and acceleration
* driveDistance (dist, vel, acc) – drive specified distance
* turnAngle (ang, vel, acc) – turn specified angle
* stop (decel) – stop at a specified deceleration
* getDistanceTravelled – query for distance traveled
* getAngle – query for angle turned
* isAtTargetSpeed – check if the drivetrain is at a target velocity
* driveStraight (vel, acc) – start driving straight (using gyro)
* setMode – set mode
* extendClimber – extend Climber mechanism
* retractClimber – retract Climber mechanism

## 8.3. Hatch Grab System - preliminary

* Load
* pickUpFromFloor
* deposit
* hasCargo
* setMode

## 8.4. Cargo System - preliminary

* Load
* pickUpFromFloor
* deposit
* hasCargo
* setMode

## 8.5. Up/Down Mech - preliminary

* moveUp
* moveDown
* getPosition
* gotoPosition
* calibrate
* setMode

# 9. Patterns

1. Command-based robot
2. PID Feedback Loop
3. Model View Controller
4. Finite-state Machines

# 10. Robot Configuration

## 10.1. User Settings

### 10.1.1. Driver

* A:
* B:
* X:
* Y:
* Start:
* Back:
* Right Bumper:
* Left Bumper:
* Right Stick Button:
* Left Stick Button:

### 10.1.2. Co-driver

* A:
* B:
* X:
* Y:
* Start:
* Back:
* Right Bumper:
* Left Bumper:
* Right Stick Button:
* Left Stick Button:

## 10.2. Robot Settings

* DeadzoneX(horizontal - left joystick)
* DeadzoneY(vertical - left joystick)
* Subsystem class names(drivetrain, lift mechanism, scoring mechanism)
* Algorithm variables
  + PID constants
  + Automation coefficients
* Physical Properties
  + Wheel diameters
  + Gear ratios

# 11. Actuators and Sensors

## 11.1. Tracking System - preliminary

### Actuators

* 1 Vertical servo motor for camera mount – controlled through network tables
* 1 Horizontal servo motor for camera mount – controlled through network tables

### Sensors

* 1 Azimuth angle to target – retrieved through network tables
* 1 Distance to target – retrieved through network tables

## 11.2. Drive Train

### Actuators

* 4 Drive wheel motors – controlled with Spark Max controllers
* 1 Scissor lift motor – controlled with Talon controller

### Sensors

* 4 Drive wheel encoders – retrieved as quadrature encoders through DIO port
* 2 Scissor lift limit