Robotics Organization

Plans

Plan

1. Pre-Season: Teaching new programmers and relearning ourselves
    - 1.1. Way around the robot and software (what do all the parts of the robots mean, how to turn it on/off)
        - 1.1.1. Go over document
        - 1.1.2. Give a visual tour of the robot
        - 1.1.3. Look at software being used and how it works in conjunction with the robot
    - 1.2. Code foundations of a robot upward (getting a better understanding of what the code is doing)
        - 1.2.1. Big steps:
            - 1.2.1.1. Reprogramming Waffle
            - 1.2.1.2. Starting a robot from scratch
            - 1.2.1.3. Tuning up Archie
        - 1.2.2. Small steps:
            - 1.2.2.1. Getting a light blinking
            - 1.2.2.2. Setting up a WPILIB Project
                - 1.2.2.2.1. Make a subsystem (part of the robot) and command
                - 1.2.2.2.2. Building and deploying code
            - 1.2.2.3. Learn about control loops
    - 1.3. 2024 repo (GitHub repository, “giant filing cabinet”)
2. Build Season
    - 2.1. Grab vegetables
    - 2.2. Accurate figure-8 path
    - 2.3. Path to go somewhere, turn around, and go back
    - 2.4. Reset the encoders
    - 2.5.
    - 2.6. Get a working digital simulator (label the constants)
    - 2.7. Tweak PathPlanner
    - 2.8. ~~Make wheels realign themselves by default~~
    - 2.9. Suck-in and push-out
    - 2.10. Angles for shooting
    - 2.11. Climbing

To-do:

1. FIx turn coding

Auto

1. Key
    - 1.1. Reset
    - 1.2. Arm
        - 1.2.1. Arm base position
        - 1.2.2. Intake wheels
        - 1.2.3. Flywheels
    - 1.3. Drive
    - 1.4. Camera
2. Routine
    - 2.1. Reset
        - 2.1.1. Reset wheels to 0
        - 2.1.2. Reset arm to 0
    - 2.2. Detect location
        - 2.2.1. Find numbers
            - 2.2.1.1. If a4 & b3
                - 2.2.1.1.1. Red
            - 2.2.1.2. If a8 & b7
                - 2.2.1.2.1. Blue
        - 2.2.2. Find position
            - 2.2.2.1. a > b
                - 2.2.2.1.1. Left
            - 2.2.2.2. a < b
                - 2.2.2.2.1. Right
            - 2.2.2.3. a = b
                - 2.2.2.3.1. Center
    - 2.3. Run corresponding program
        - 2.3.1. Move arm to shoot position
        - 2.3.2. Shoot note
        - 2.3.3. Drive back to note
        - 2.3.4. Move arm to pick up position
        - 2.3.5. Pick up note
        - 2.3.6. Drive back behind white line
        - 2.3.7. Drive to shoot location
        - 2.3.8. Move arm to shoot position
        - 2.3.9. Shoot note

How To

How to set a robot set up with code:

1. Flashing the Rio to make sure it has firmware (give them an operating system)
2. Set up the radio (a wifi access point, allows you to connect from the computer to the robot without a USB cord)
3. Make sure motor controllers and sensors that you're using are connected (CANBUS, a 2-wire cord connection like USB)
4. Deploy code to the robot

How to drive Archie:

1. Controller 1: Drive
    a. Left joystick: Strafing
    b. Right joystick: Rotate
2. Controller 2:
    a. Left bumper: Eject
    b. Right bumper: Pick up

How to set wheel offsets:

1. Create a boolean for the booting status:
    a. Boot to zero = true (boot to absolute position = false)
    b. Use motor offsets = false
2. Record the angle of the wheels (in degrees) in the wheel offsets, then:
    a. Boot to zero = false (boot to absolute position = true)
    b. Use motor offsets = true (set the offsets, in constants, to the recorded values)
3. Invert one side of the wheels (if necessary)
4. To realign wheels: Go to Teleop when disabled then re-enable
    a. Elsewise, redeploy
    b. Elsewise, turn off and on the robot

How to wet up Spark Max Controllers:
(Note: If the message "No devices detected" appears, unplug then replug the cable.)

1. Plug USB to USB-C from the computer into Spark Max Controller
2. Open (and update) REV Hardware Client
3. Download the latest firmware (as prompted by REV)
4. Locate the Spark Max on the side panel ("USB" will be visible underneath to show the Spark Max you are currently plugged into)
5. Under "Update" press "Update Firmware"
6. Under "Basic" set the CAN ID to the corresponding number then press "Burn Flash"
7. Repeat for every Spark Max Controller

How to use GitHub:

1. Build robot code
2. "git status"
3. "git add [any new files]"
4. "git commit -am "[the name you want the commit]""
5. "git push" to make it accessible from another computer
6. To git from another computer: "git pull"

What's Robotics?

A robot consists of

1. Physical elements
    - 1.1. Output (physical movement)
        - 1.1.1. Motors (move robot and parts)
            - 1.1.1.1. Continuous motion (set velocity or acceleration)
            - 1.1.1.2. Servo (set angle)
        - 1.1.2. Motor controllers (control the voltage passed to the motors) [Ex: Can Spark Max, CTRE Talon, CTRE Falcon]
        - 1.1.3. Pistons (move pneumatic parts of the robot)
        - 1.1.4. Solenoids (control pistons)
    - 1.2. Robot input (uncontrollable)
        - 1.2.1. Camera (see from the robot's perspective; detects shapes, colors, april tags, etc.)
        - 1.2.2. LimeLight (camera that detects and tracks reflectivity)
        - 1.2.3. Encoders (measure revolutions, the speed the motors turn)
        - 1.2.4. Gyros (measure of acceleration: rate of change of velocity [rate of change of position], orientation)
        - 1.2.5. Sensors (distance sensors, lidar [Light Detection and Ranging], etc.)
        - 1.2.6. Limit switch (a button that reads from the motor controller, can stop a robot from overextending)
    - 1.3. Human input (controllable)
        - 1.3.1. Buttons (boolean trigger [0 or 1]) [Ex: A, B, X, Y, joystick buttons, bumpers]
        - 1.3.2. Trigger (1 axis value [double from 0 to 1])
        - 1.3.3. Joystick (2 axis values, x and y)
        - 1.3.4. D-pad/POV (angle [integer from 0 to 8, times 45])
2. Code
    - 2.1. Output (giving info to physical elements)
        - 2.1.1. Physical change
            - 2.1.1.1. Pass a value to the motor controllers (motors [double])
            - 2.1.1.2. Changing a solenoid state (open/close a piston [boolean])
            - 2.1.1.3. PID controller (passes a value to the motor controllers [distance from a target, where you want to go]) [smart with the movement of the robot, corrects for error in the output]
        - 2.1.2. Digital change
            - 2.1.2.1. Changing a camera setting (LimeLight)
            - 2.1.2.2. Updating values in SmartDashboard
            - 2.1.2.3. Turn lights on/off
            - 2.1.2.4. Calibration of sensors
    - 2.2. Input (uncontrollable, receiving info from physical elements)
        - 2.2.1. Reading sensor values
        - 2.2.2. Reading controller values
    - 2.3. Processing
        - 2.3.1. Kinematics (move the arm up 1 m, forward 1 m; calculate 2 angles; give them to a PID controller)

2.3.2. Image processing (retroreflective tape; LimeLight's input; trigonometry for distance; movement to PID controller)
2.3.3. Sequencing actions (move arm down; grab object; move arm up and forward; drop object)
2.3.4. Motion profiling (positions moved-to based on sub-positions or using PathWeaver)

What makes up a WPILIB project in Visual Studio Code:

1. Subsystems (access part of the robot)
2. Commands (tell subsystems what to do with parts of the robot)
3. Gradle (build system, collects all written code and sends it to the robot)

Code:

1. Software that connects physical controllers to physical/digital output (mapping human input to robot output)
2. Drive train (how the robot moves around the field and orients)
3. Camera input
4. Parts of the robot

Integrated circuits:

1. Relationship between multiple devices and how they interact with each other

PID Constants:

1. Position
   1.1. The more offset, the more correction
   1.2. As it gets closer to the target, the less correcting it does
   1.3. High correction at start, low to no correction near target = can slightly miss target
2. Integral
   2.1. Progressively increases correction when closer to the target = might oscillate on target
3. Derivative
   3.1. Decreases the amount of correction to avoid to much correction at the start

Logins

Usernames and Passwords:

- General
  - New password: 159!
  - Old password: 159GoAlpine!
- GitHub
  - Name: FRCTeam159
  - Username: Alpine-Robotics
  - Password: AlpineRobotics159

Emails

Email:

(Dear XXX,

Hello, I hope you are doing well!) Insert your own greeting here.

My FIRST robotics team, Alpine Robotics, is looking forward to a new build season and robot challenge, starting January 2024!

We're using December 5th, Colorado Gives Day, to raise funds to support our mission of engaging students in STEM (science, technology, engineering, and math) education by designing, building, and programming a robot to participate in FIRST® Robotics competitions.

Although **donations can be made at any time** through the Colorado Gives website, **Alpine Bank has offered to match every dollar earned**, one for one, from now through the end of this year (up to $500). We would like to raise as much money as we can by this time, to help offset our costs for the upcoming season, so all donations are greatly appreciated!

If you would like to help, (click here) [re-insert link https://www.coloradogives.org/organization/FIRSTTeam159AlpineRobotics] or go to our website at [re-insert link https://alpinerobotics.org/]. You can also visit [re-insert link www.coloradogives.org] and search "FIRST Team 159 Alpine Robotics", and our donation site will pop up.

Thank you for supporting our team!

Sincerely,

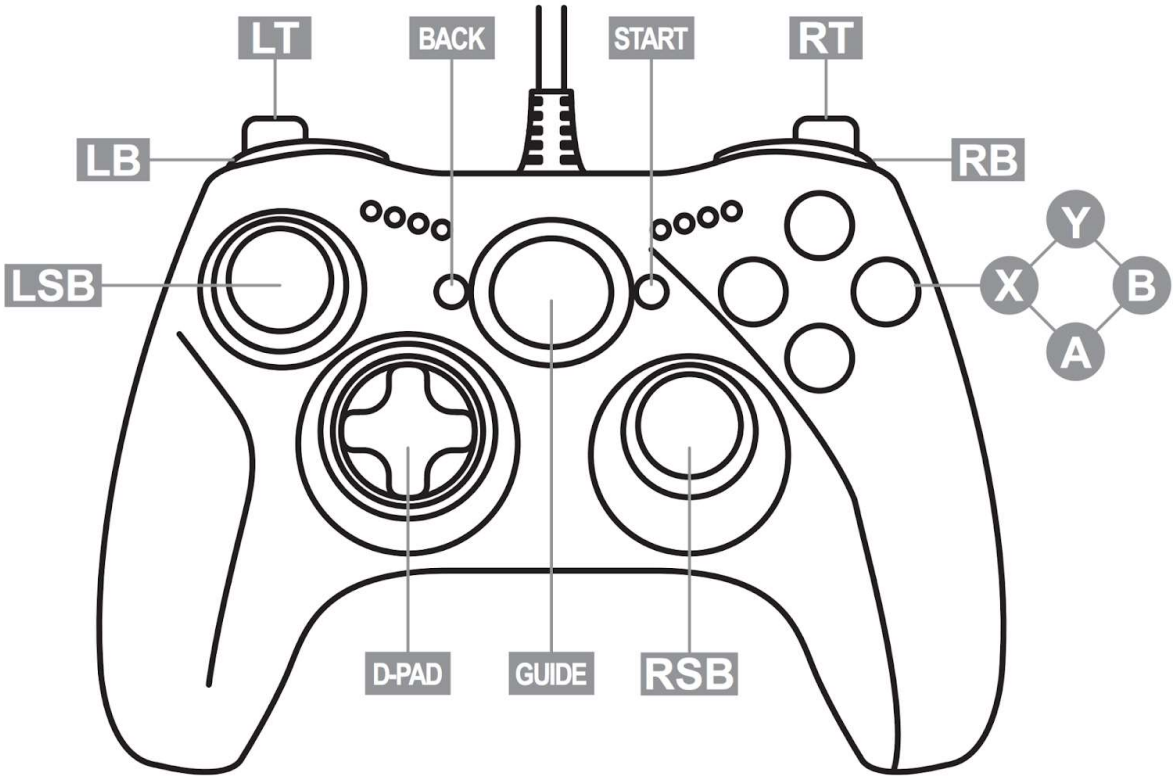
Your Name Here
More about our team:

Poudre School District's Alpine Robotics, Team 159, is the oldest FIRST Robotics Competition (FRC) team in Colorado, started at Poudre High School in 1998, by a very dedicated teacher, Steve Sayers. We have traveled to competitions all across the country and have since inspired numerous teams to form in Colorado.

Our student-led team works with mentors to design, build, and program robots to compete in a FIRST Robotics, game-style competition. Each year our robots must shoot projectiles, climb structures, balance on boards, and strategically deploy cargo to score game points. Throughout our history, we have won awards for supporting FIRST's values, having exceptional mentors, and doing well in the competition. Our team has supported other FRC and middle school FIRST Lego League teams, and has demonstrated our bots at local schools and businesses to increase support for STEM education.

As of 2019, Alpine Robotics has been a 501(c)(3) organization thanks to a group of parents and mentors lending their support. This has allowed us to be able to participate in grant funding and other opportunities that may not be open to school districts. After competition was limited in 2020, we rely more than ever on our sponsors, businesses in our community, and individuals to raise the funds needed to continue the team.

2024 Button Layout



- Movement
    - LS: Forward, backward, and strafe
    - RS: Rotate in place
- Arm Movement:
    - LT: Arm down
    - RT: Arm up
- Arm Positions
    - A: Pickup
    - B: Amp
    - X: Speaker
- Note
    - LB: Pick up
    - RB: Shoot

Plan:

1. ~~Write down button layout~~
2. Get autonomous fully simulated w/ commands
    a. Create a command for moving the arm to a position
    b. Fix init to be “setAngle”