Raider Robotics Controls Notebook

2024-2025 - VEX High Stakes

Contents

Meeting Notes - Our Formatting

Timestamp: [Date] - Author
Attending:

Meeting Goals

- Goals / Directives for the meeting

Discussion

Details of the meeting including resources used, implementation details, and meeting setup details.

Results

- Accomplishments of the meeting

Action Items

- Todos for next meeting

Tuesday Strategy Meeting

Timestamp: [09/17/2024] - [Luke Gagnon, Leigh Goetsch]
Attending: Peter Wolfgramm, Andrew Needham, Miles Trompeter, Landon Van Mersbergen

Meeting Goals

- Tools for developing robot strategies
- Investigate virtual robot simulation options

Discussion

Here is an option for virtual robot simulation recently posted to the vex forums. I think looking at options for virtual robot simulation for controls would be a good idea. bowlerstudio

Here is the pygame repo for testing robot strategies: repo

Wednesday Integration Meeting

Timestamp: [09/18/2024] - Luke Gagnon, Leigh Goetsch
Attending:

Meeting Goals

- Finish Friday task
- Assemble and test odom systems

Discussion

We will be finishing the challenge you were given on Friday: program a robot to move autonomously across the field as creatively as possible

Add a copy of the auton program to a text file in the "autons" folder. Please put your name and auton file name here when done!! :D

Results

name(s)	file name
Andrew	backwards-auton
Miles	swirly-auton

Table 1: Auton files

Tuesday Strategy Meeting

Timestamp: [09/24/2024] - Luke Gagnon, Leigh Goetsch
Attending:

Meeting Goals

- Finish the STEM Center Canvas course for STEM Center Access
- Control Theory: Tracking Robot Position
- Strategy Formulation: Analyzing the Game and Field

Discussion

Please complete the STEM Center Canvas course for STEM Center Access. This will give you STEM Center access for the rest of the year.

Here is the page on odomentry and tracking robot position from the Purdue VexU team

Youtube video series on odometry

For the strategy formulation, we will be analyzing the game and field. We will be putting a picture of the field on the board and discussing path planning and autonomous routines. Add any notes you have to the strat folder, please!!!

Results

- STEM Center Canvas course completed
- Watched youtube video series on odometry
- Analyzed the game and field

Monday Controls Group Meeting

Timestamp: [9/30/24] - Landon & Hunter
Attending:

Results

- Odometry pods are working and calculations appear to be accurate. More precise offset measurements will be needed to improve accuracy.
- Square root input curve on drive is working using voltage control.
- Voltage control allows motors to drive above max speed.
- Due to uneven friction, voltage control adds slight drift and may need to be removed.

Action Items

- SD card for differentiation between bots, images, odometer.
- Implementing path gen for usable auton paths in competition.

Friday Controls Group Accomplishments

Timestamp: [10/04/24] - Landon & Hunter
Attending:

Results

- Added control to toggle claw in op and autonomous control.
- Practiced autonomous routes with claw mechanisms.

Action Items

- Research pros tasks to run processes in parallel.
- Research odometry for autonomous routes.
- Plan autonomous routes for skills and competition.

Strategy Meeting - Project Updates

Timestamp: [10/08/2024] - Luke Gagnon, Leigh Goetsch
Attending: Peter Wolfgramm, Andrew Needham, Miles Trompeter, Landon Van Mersbergen

Meeting Goals

- Review progress on current project objectives
- Discuss next steps for each project area
- Identify any roadblocks or challenges
- Plan for future strategy meetings

Project Updates

The following is a summary of the progress made in each project area and the next steps planned for each group:

Leblib Path Planning

Group: Andrew, Peter, Salvin
Objective: Set up the Leblib path planning library and begin integrating it with autonomous navigation and odometry.
Plan:

- Successfully set up the path planning library on the development environment.
- Initial testing was not conducted yet; plan to test integration with the robot's autonomous systems and odometry tomorrow.

Next Steps:

- Tomorrow, test the library with the robot's autonomous functionalities and odometry data.
- Identify any adjustments or customizations needed for better integration.

Robot Communication

Group: Luke, Leigh, Miles
Objective: Establish a communication protocol between robots and test message reception.
Plan:

- Set up a basic test environment to verify correct message reception between robots.
- Began discussing the communication protocol for inter-robot messaging, focusing on what specific data needs to be shared.

Key Considerations for Communication Protocol:

- Location data: Essential for coordinating movements between robots.

- Object detection data: Could be useful in determining shared understanding of obstacles or objectives.
- Game state: Related to object detection, the state of the game (such as target detection or task progress) may need to be shared.
- Help signal: To allow one robot to request assistance from another if necessary.

Next Steps:

- Finalize the communication protocol, specifying the exact data types to be transmitted.
- Implement and test the protocol between the robots.

Task Abstraction

Group: Not assigned yet
Objective: Develop a task management system for handling asynchronous tasks during autonomous robot operation.
Plan:

- Research two potential approaches: creating a custom task system vs. leveraging the PROS library's task management system.
- Determine how events can be triggered within the system to better manage robot actions.

Next Steps:

- Research both options: creating a custom task system vs. leveraging PROS.
- Determine how events can be triggered within the system to better manage robot actions.

Web Dashboard

Status: Paused / Peripheral
Objective: Update and integrate the internal path planning interface to export JSON data.
Plan:

- The web dashboard interface, developed a few years ago, needs updates to ensure compatibility with our current system.
- Initial review of legacy code has not yet been conducted.

Next Steps:

- Review the legacy code to assess the current implementation.
- Ensure the dashboard can export JSON data for use in other areas of the project.

Website Maintenance

Group: Salvin, Andy
Objective: Modernize the website and update the content.
Plan:

- The website is outdated and has dependency issues with npm, preventing updates in its current state.
- Initial steps will involve resolving these dependency issues to make the site functional in modern environments.

Content Updates Considered:

- Sponsor list.
- Links to relevant GitHub repositories (pneumatic reverse engineering, senior design projects, web dashboard, website).
- Past engineering notebooks, Onshape designs (past robots), and updated tournament statistics.
- Contact information update.

Next Steps:

- Fix the dependency issues.
- Begin content updates as discussed, starting with sponsor lists and repository links.

Wednesday Integration Meeting

Timestamp: [10/09/2024] - Luke Gagnon, Leigh Goetsch
Attending: Miles,

Meeting Goals

- Implement a simple program for exploring VEXlink communication

Discussion

Using the VEXlink communication protocol, we can send signals to the robot. However, we are unable to send signals when the remotes are connected. We wrote some basic tests to try to get the remotes connected.

Results

- Wrote a simple program to test VEXlink communication
- We can send signals, but not with remotes connected
- Wrote some basic tests to try to get to connected

Tuesday Strategy Meeting

Timestamp: [10/22/2024] - Luke Gagnon, Leigh Goetsch
Attending: Peter Wolfgramm, Andrew Needham, Miles Trompeter

Meeting Goals

- Try to figure out the reason behind VEXlink issues
- VEXlink documentation reading
- Get VEX AI team members GHOST project setup and running

Discussion

- VEXlink Docs
- Excerpts:
  - When multiple radios are connected to a V5 brain, the radio in the highest numbered smart port will be used for the controller VEXnet connection, to avoid errors
  - Data Rate:
    - Maximum data rate for manager to worker robot is 1040 bytes/second.
    - Maximum data rate for worker to manager robot is 520 bytes/second.

Results

- VEXlink issues are likely due to the order of the controller and the VEXlink device ports

Action Items

VEXU Team Tasks

- VEXlink
  Leigh, Miles
- LebLib
  Andrew

VEX AI Team Tasks

- Nano -> V5 microcontroller communication:
  - How is the nano and the V5 microcontroller working together?
  - What is being processed on what?
  - Create a simple program to send a message from one to the other
- ROS2 serial robot messaging
  - How is ROS architected?
  - How is this useful in our use-case?
  - Write a publisher and subscriber script (or run the one in GHOST's example)

- robot -> robot messaging (VEXlink)
  - What messages should robots send each other
  - What message protocol do robotic systems usually use?
  - What message protocol should we use?
  - Use VEXlink PROs API to write a simple program to send a message from one robot to another.
- Simulation - GHOST
  Evan, Salvin, Leigh
  - The sim software that GHOST uses is called Gazebo Sim
  - The python that the bash file ./scripts/launch_sim.sh runs can be found at 04_Sim/ghost_sim/launch/
  - There is a typo that causes the world to not load in the sim env, will post the fix later tonight.
  - What information can the sim env store?
  - What is the filetype of the world and the model files?
- Path planning - GHOST
  - GHOST uses https://web.casadi.org/ for path planning
  - What does this library do?
  - What are the input and outputs for their algorithm?
- MSOE Senior Design - Adversarial Strategy VEX Robot Program
  - This is a senior design project from two years ago using RL
  - If you are itching for more project setup, load up this project
  - How does this project represent a robot?
  - What information does the robot store?
  - How does the robot choose actions?

Specific People:

- Nano Setup
  Evan, Salvin
  - Try following the instructions in this repo to setup your nano and run the example code
- Rosie Newbie
  Andrew
  - Go to the AI Club Learning Tree and follow the instructions to practice requesting rosie and running jobs
  - I will have some files for you in the next couple days for this

Wednesday Controls Group Accomplishments

Timestamp: [10/23/24] - Luke Gagnon
Attending: Miles, Gideon

Discussion

port maps:

- left drive: 11, 12, 13
- right drive: 14, 15, 16
- tandem motor: 17, 18
- solo motor: 19

Results

- Library Research project (Luke)
  - Continued investigation of pros vs. custom libraries
- Main Robot project (Luke)
  - Started getting code set up for the current robot project
- Vex Link project (Miles, Gideon)
  - Continued research for vex link for use in Vex AI

Action Items

- Library Research project: Continue to investigate pros vs. custom libraries
  Luke
- Main Robot project: Do whatever design needs
  Luke
- Vex Link project: Continue vex link work
  Miles, Gideon

Friday Controls Group Accomplishments

Timestamp: 10/25/24 - Luke Gagnon
Attending:

Results

- Main Robot project (Luke)
  - Fixed some problems with the current robot project
  - Got pneumatics working on the current robot
  - Looked into setting up odom

Action Items

- All project
  - Touch base with everyone to see what's happening with teams and members who have been busy and might not have been at all meetings

Friday Controls Group LemLib Accomplishments

Timestamp: 11/08/24 - Andrew Needham
Attending:

Results

- LemLib (Andrew)
  - Figured out the old version of LemLib was causing problems when calculating robot position
  - Created a new PROS project with the latest version of LemLib which worked correctly
  - Created constants to initialize EZTemplate and LemLib robot configurations **LemLib Test Project**

```
const vector<int> leftMotorPorts = {-5,8,-9};
const vector<int> rightMotorPorts = {-3,4,-11};
const int motorRPM = 600;
const pros::motor_gearset_e_t motorGearSet =
   pros::motor_gearset_e_t::E_MOTOR_GEAR_600;
const int imuPort = 16;
const float driveWheelDiameter = 3.25;
const float trackingWheelDiameter = 2;
const float driveRPM = 360;
const float driveTrackWidth = 10.75;
const float verticalTrackingWheelOffset = 1.625;
const float horizontalTrackingWheelOffset = 2.75;
const int verticalEncoderPortTop = 3;
const int verticalEncoderPortBottom = 4;
const int horizontalEncoderPortTop = 2;
const int horizontalEncoderPortBottom = 1;
```

  - Created a PROS task to control the claw without interrupting the main leftMotorPorts **LemLib Test Project**

```
bool clawOn = false;
void toggleClaw(){
pros::Task claw_task([&]() {
      if(clawOn){
         Claw2.set_value(0);
         pros::delay(100);
         Claw1.set_value(0);
      }else{
         Claw1.set_value(1);
         pros::delay(100);
         Claw2.set_value(1);
      }
      clawOn = !clawOn;
   });
}
```

Action Items

- All project
  - Continue development of the new PROS project OR update the current vex-base to new version of LemLib
  - Create a GitHub repo for the new project

LemLib implementation

Timestamp: [11/01/24] - Andrew Needham
Attending:

Goals

- Get LemLib odometry values to read the robot position on the field

Methods

LemLib odomentry values all read as NaN Troubleshooted by checking to see what values the encoders read and then checked where the problem first started occuring

Results

- Motor encoder values read correctly, around 1000 units per rotation
- Values from the tracking wheel objects are not reading correctly, will jump between 1000000, -1000000 and 0 seeminly randomly
- Found source code for LemLib TrackingWheel

  TrackingWheel.getDistanceTraveled

```
float lemlib::TrackingWheel::getDistanceTraveled() {
    if (this->encoder != nullptr) {
        return (float(this->encoder->get_value()) * this->diameter *
            M_PI / 360) / this->gearRatio;
    } else if (this->rotation != nullptr) {
        return (float(this->rotation->get_position()) * this->diameter
            * M_PI / 36000) / this->gearRatio;
    } else if (this->motors != nullptr) {
        // get distance traveled by each motor
        std::vector<pros::MotorGears> gearsets =
            this->motors->get_gearing_all();
        std::vector<double> positions =
            this->motors->get_position_all();
        std::vector<float> distances;
        for (int i = 0; i < this->motors->size(); i++) {
            float in;
            switch (gearsets[i]) {
                case pros::MotorGears::red: in = 100; break;
                case pros::MotorGears::green: in = 200; break;
                case pros::MotorGears::blue: in = 600; break;
                default: in = 200; break;
            }
            distances.push_back(positions[i] * (diameter * M_PI) *
                (rpm / in));
        }
        return lemlib::avg(distances);
    } else {
        return 0;
```

```
            }
        }
```

Action Items

- Will try to use old VEX encoder to see if the precision of the encoders are leading to position calculation issues
- Try to find what "‘TrackingWheel.getDistanceTraveled()"‘ does to figure out why it’s not working

CasADi Test Program

Timestamp: [11/05/24] - Andrew Needham
Attending:

Goals

- Create a basic Python program to get familiar with using the CasADi library to generate robot path trajectories

Methods

- Use ChatGPT to generate a basic test program and iterate from there to fit our use case
- Use the CasADi documentation to figure out what inputs and outputs are needed

Results

- Was able to get a basic program that minimizes the distance traveled between two points given a list of obstacles to avoid
- The program optimizes the objective function given a set of constraints and plots the path on a graph
- GitHub repo for CasADi test program: https://github.com/msoe-vex/CasADi-Test.git

Action Items

- Will update the program to minimize the amount of time used to travel from one point to another
- After that, will work on converting the Python program to a C++ implementation

CasADi Test Program

Timestamp: [11/08/24] - Andrew Needham
Attending:

Goals

- Update the python test path program to minimize the amount of time to traveled instead of just the path length

Methods

- Use ChatGPT to generate a basic test program and iterate from there to fit our use case
- Use the CasADi documentation to figure out what inputs and outputs are needed
- Find other existing programs to build off of

Results

- Was able to get a solver working that minimizes the amount of time
- The solver also has constraints for maximum velocity and acceleration
- The robot starts and ends at a complete stop
- The program is very finicky at the moment, any small change in the constraints or staring values can cause the solver to not be able to find a solution
- GitHub repo for CasADi test program: https://github.com/msoe-vex/CasADi-Test.git
- GitHub repo for another users implementation: https://github.com/tianchenji/path-planning.git

Action Items

- Will use the tianchenji/path-planning.git repo to learn how to better implement CasADi
- After that, will work on converting the Python program to a C++ implementation

CasADi Test Program

Timestamp: [11/11/24] - Andrew Needham
Attending:

Goals

- Update the python test path program to follow the laws of physics and come to a complete stop at start and end points

Methods

- Iterate the path planning repo to fit our use case
- Use the CasADi documentation to figure out what inputs and outputs are needed
- Utilize ChatGPT to quickly develop the program

Results

- Was able to get a solver working that follows a smooth path and limits the maximum speed and velocity
- The solver works in about 0.1 seconds so it will be able to calculate paths in real time during the match
- Currently the program has a static time interval and a variable amount of steps to look into
- GitHub repo for CasADi test program: https://github.com/msoe-vex/CasADi-Test.git
- GitHub repo for another users implementation: https://github.com/tianchenji/path-planning.git

Action Items

- Will update the program to change the units to inches so that it can be used with LemLib
- Will research how LemLib controls the robot based on a list of points
- The program can be updated to use a set number of steps and a variable amount of time in between them
- After that, will work on converting the Python program to a C++ implementation

LemLib Test Program

Timestamp: [11/14/24] - Andrew Needham
Attending:

Goals

- Create a LemLib PROS project to control the robot in auton and opcontrol

Methods

- Create a new PROS project
- Install LemLib depot
- Update the program to control the robot in auton and opcontrol

Results

- Was able to get a program working that calculates the position of the robot based of the inputs of tracking wheel encoders
- Added classes to allow selection of auton during the initialize part of the match

Action Items

- Confirm that the auton selector works in a match
- Create a split arcade drive configuration to control the robot in opcontrol
- Tune PID constants and make sure the chassis configuration is correct and tracks robot position correctly

VEX Reinforcement Learning

Timestamp: [1/31/25] - Andrew Needham
Attending:

Goals

- Create a reinforcement learning environment for the VEX robot
- Train the robot to use an optimal strategy to score as many points as possible

Methods

- Use Gymnasium and Stable Baselines3 to create the environment and train the robot

Results

- Was able to create a basic environment with several actions and render each step
- Added code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Add restrictions for actions
- Add more actions
- Update rewards for each action
- Train the model to score as many points as possible in as little time as possible
- Update the structure of the environment to better represent the robot and field

VEX Reinforcement Learning

Timestamp: [2/7/25] - Andrew Needham
Attending:

Goals

- Create a reinforcement learning environment for the VEX robot
- Train the robot to use an optimal strategy to score as many points as possible
- Update the structure of the environment to better represent the robot and field

Methods

- Use Gymnasium and Stable Baselines3 to create the environment and train the robot
- Use Slurm to run the training script on the cluster to train quickly

Results

- Was able to create a basic environment with several actions and render each step
- Updated the reward system to use the score of the field instead of arbitrary rewards assigned to each action based on results
- The reward = score after action - score before action
- After training for 500,000 steps, the robot was able to score about 24 points in 120 seconds
- Updated code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Update the structure of the environment to better represent the robot and field
- Add different colors for rings
- Add wall stakes
- Add probability of actions succeeding
- Update the environment to account for unknown and missing information about elements on the field

VEX Reinforcement Learning

Timestamp: [2/12/25] - Andrew Needham
Attending:

Goals

- Create a reinforcement learning environment for the VEX robot
- Train the robot to use an optimal strategy to score as many points as possible
- Update the structure of the environment to better represent the robot and field

Methods

- Use Gymnasium and Stable Baselines3 to create the environment and train the robot
- Use Slurm to run the training script on the cluster to train quickly

Results

- Added wall stakes to the environment and added actions to move to and ad rings to stakes
- Simplified the observations space and added time remaining to the observation space
- Fixed some of the restrictions on the actions to account for game rules and physical limitations
- Adding a reward for climbing makes the robot reach a local maximum and not explore other actions
- Added small chance of actions failing to simulate the robot not being able to perform an action
- Experimented with the reward system to give points based on estimated final score at the end of the episode
- Noticed that training usually peaked around 2 million steps and slowly started to decline
- It seems that after a certain number of steps, the robot starts to overfit the environment and starts repeating the same actions
- Updated code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Update the structure of the environment to better represent the robot and field
- Add different colors for rings
- Update the environment to account for unknown and missing information about elements on the field
- Experiment with the PPO parameters to see if the robot can learn more effectively

VEX Reinforcement Learning

Timestamp: [2/17/25] - Andrew Needham
Attending:

Goals

- Create a reinforcement learning environment for the VEX robot
- Train the robot to use an optimal strategy to score as many points as possible
- Update the structure of the environment to better represent the robot and field
- Optimize the environment and training process to improve the robot's performance

Methods

- Use Gymnasium and Stable Baselines3 to create the environment and train the robot
- Use Slurm to run the training script on the cluster to train quickly

Results

- Consolidated actions to simplify the number of choices to lead to more optimal strategies
- Added a small penalty to invalid actions to discourage the robot from performing them
- Added an option to randomize the field to help generalize the robot's strategy
- Gave the robot a FOV to determine what objects it can see
- The robot now learns an optimal strategy much more effectively and consistently and can adapt to different field states
- Updated code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Add different colors for rings
- Update the environment depending on the match type (skills or regular match)
- Experiment with the PPO parameters to see if the robot can learn more effectively

VEX Reinforcement Learning

Timestamp: [2/25/25] - Andrew Needham
Attending:

Goals

- Create a reinforcement learning environment for the VEX robot
- Train the robot to use an optimal strategy to score as many points as possible
- Update the structure of the environment to better represent the robot and field
- Optimize the environment and training process to improve the robot's performance
- Integrate the CasADi program to plan optimal paths between positions for each action
- Add options to change hyperparameters and environment settings to optimize training

Methods

- Use Gymnasium and Stable Baselines3 to create the environment and train the robot
- Use Slurm to run the training script on the cluster to train quickly

Results

- Added arguments to change hyperparameters and environment settings to allow for more flexibility in training
- Integrated the CasADi program to plan optimal paths between positions for each action and to calculate realistic estimates for the paths
- Training now takes significantly longer but uses more accurate time costs for paths, this can be turned off to speed up training
- Updated code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Add different colors for rings
- Update the environment depending on the match type (skills or regular match)
- Export functions for the VEX program based on actions that take place to create autonomous programs
- Test the VEX program to ensure paths generated by a list of points work as expected

Path Planner Improvements

Timestamp: [3/10/25] - Andrew Needham
Attending:

Goals

- Improve efficiency and performance of our path planning algorithm for our simulation
- Integrate the path planning algorithm with the VEX robot simulation environment control program

Methods

- Implement A* algorithm to find optimal initial path to optimize
- Export path as a flattened list of points to be used in the VEX program
- Include total distance for better estimate of duration of the path

Results

- The A* algorithm worked effectively to find an initial path for the robot to follow
- The new inital path generation only adds about 0.1 seconds to processing time, which is a sufficiently small amount to not affect path genration in competition
- Generating paths in training slows down training significantly, but this can be turned off to speed up training
- The initial path generated by A* reduced the number of failed paths by about 20% in the training environment
- The path was successfully exported as a flattened list of points, which can be used in the VEX program
- Running the path is not completely accuratue due to PID not being tuned for the new path generation, but it is a good starting point
- Updated code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Fix the PID controller to work with the new path generation
- Ensure the path is being followed correctly in the VEX program and accounting for robot's size

Multi Agent Reinforcement Learning

Timestamp: [4/29/25] - Andrew Needham
Attending:

Goals

- Utilize multi agent simulation to train multiple robots to work together
- Build off of the existing simulation environment to allow for multiple robots to be trained at once

Methods

- Use PettingZoo library to create a multi-agent environment
- Use RlLib from Ray to train the agents in the multi-agent environment
- Test the multi-agent environment with multiple robots to ensure they can work together effectively

Results

- The new environment was successfully created using the PettingZoo library
- Actions are sent to the model in an synchonized format, and each action is performed in chronological order
- After each action, the "robot time" is updated to reflect the time taken for the action to be performed
- Only the robot with the lowest "robot time" is allowed to take an action, which prevents multiple robots from taking actions at the same time
- The multi-agent environment was successfully tested with multiple robots, and they were able to work together effectively
- The agents can learn from randomized environment states to learn generalized game strategies
- The agents don't currently have information about the other agents, so they must learn to work together without communication
- Updated code to GitHub repo: https://github.com/msoe-vex/VEX-AI-Reinforcement-Learning.git

Action Items

- Create script to compile the policy into a compact inferencing model that can be used in the VEX program
- Integrate the compiled model with communication between the VEX program and the Jetson Nano running the model
- Test running the model on the Jetson Nano to ensure it can run in real-time

Reinforcement Learning Model Integration

Timestamp: [5/23/25] - Andrew Needham
Attending:

Goals

- Create a script to compile the policy into a compact inferencing model that can be used in the VEX program
- Integrate the compiled model with communication between the VEX program and the Jetson Nano running the model
- Test running the model on the Jetson Nano to ensure it can run in real-time

Methods

- Use RlLib library to trace the policy from a checkpoint and compile it into a compact inferencing model for Pytorch
- Get information about the current state of the environment and the actions taken by the agents from the VEX program and Intell RealSense camera
- Use the compiled model to predict the next action for each agent based on the current state of the environment
- Send the predicted action to the VEX program to be executed by the robot

Results

- The script to compile the policy into a compact inferencing model was successfully created using RlLib
- The model had to be traced instead of scripted due to the format of the policy and environment using PettingZoo
- The Jetson Nano was able to get information about the current state of the environment and the actions taken by the agents from the VEX program and Intell RealSense camera
- The script was able to run on the Jetson Nano to predict the next action for each agent based on dummy data
- Updated code to GitHub repo: https://github.com/msoe-vex/VAIC$_2$4$_2$5

Action Items

- Ensure the model can run in real-time on the Jetson Nano with the VEX program using information from the Intell RealSense camera
- Test the model in real time with the VEX program to ensure it can predict the next action for each agent based on the current state of the environment

AI Research Meetings Findings

CasADi and GHOST Path planning

Timestamp: [10/29/24] - Andrew Needham
Attending:

Research CasADi library and GHOST implementation of path planning

The goal for this exploration is to determine how to use the CasADi library to generate path trajectories for the robot

Questions Explored

- What is CasADi and what does it do?
- How does GHOST do path planning?
- What are the inputs and outputs of the algorithm?
- How can CasADi be used to generate path trajectories

Methodology

1. Reading through CasADi documentation
2. Checking GHOST public repo for CasADi implementation and path planning code
3. Create a test program to generate path and become familiar with the library

Code Locations

- GHOST VEX-U Repository: link
- GHOST Swerve drive trajectory generation link
- Test program repo: link
- CasADi Robot Path Planning example: link

Findings

- CasADi is a general purpose library with the focus of solving optimization problems given a set of constraints
- Inputs of CasADi is a function of multiple variables and a number of constraints for the function
- The GHOST systems uses serialized messaging to communicate between different classes and parts of the robot

Issues

- Unable to locate where in the GHOST repo CasADi was implemented for a tank drive
- Unable to consistently generate a path that minimizes the time taken to get to the target point

Code Snippets / Commands Used

LemLib implementation

Timestamp: [11/06/24] - Andrew Needham
Attending:

Research LemLib library

Questions Explored

- What is LemLib and what does it do?
- How can odometry be used to control the robot?

Methodology

1. Reading through LemLib documentation to configure the robot
2. Use Jerry.io and/or CasADi to plan paths to be used with LemLib

Code Locations

- LemLib source code repository (stable): link
- LemLib documentation: link
- Test program repo: link

Findings

- LemLib supports odometry that tracks the position of the robot
- PID tuning can be used to get accurate measurements
- LemLib also supports pure pursuit which can be used to create complex paths for the robot to follow

Issues

- Unable to get TrackingWheel values to read correctly

Code Snippets / Commands Used

Beginning With Gazebo

Timestamp: [11/4/24] - Evan Roegner
Attending:

Objective / Focus

The goal for this was to explore the gazebo simulation to determine what its components are, and what role they serve and can do.

Questions Explored

- What information can the sim environment store?
- What is the filetype of the world and model files?

Steps Taken / Methodology

1. Fixed the typo in the start_sim_launch file.
2. Found what files were being used to create the world and the objects/robots in it
3. Looked into those files to determine exactly what they actually were
4. Changed the filepath being used in start sim launch to accept files from a local directory

Paths to Key Files / Code Locations

- start_sim_launch: scripts -> start_sim_launch.py, run from VEXU_GHOST directory with ./scripts/launch_sim.sh
- World files: 04_Sim -> ghost_sim -> worlds
- Robot xacro files: 04_Sim -> ghost_sim -> urdf
- Macro xacro files: 04_Sim -> ghost_sim -> urdf -> macros
- Typo, local directory change, world selection: start_sim_launch Line 70
- Robot selection: start_sim_launch Line 20

Findings / Insights

- Typo and local directory line should be changed to world_file = os.path.join(home_dir, "VEXU_GHOST", "04_Sim", "ghost_sim", "worlds", **INSERT WORLD NAME HERE**")
- start_sim_launch.py launches the simulation when run. launch_setup method contains a single variable that can be changed to switch the robot that is being simulated. This means that robots are not defined in the world, but as their own file which is placed in an existing world by this script. This would be used to change the world the robot is being loaded into (line 70) as well as the robot being loaded (line 20). Also allows you to specify a joystick to use in the simulation.

- .world files are XML files which contains all of the information about a given environment a robot can be placed into. Contains the placement and state of all objects in the environment. Editing these files allow you to add physical objects into the simulation as well as change various attributes about the simulation (gravity, kinematics, etc). Many attributes in the world file can be modified while the program is running (this does not change the actual file, it just changes it in the simulation). This is used in start_sim_launch to create the environment the robot will be placed into.
- xacro files are used to define various macros and the robots that they make up. There are xacro files for robots and separate files for the macros. The macros are xml representations of robot parts
- Macro xacro files Contain information about the various parts of the robot, by default there are 2 types: actuators and sensors. Sensors are anything the robot can use to learn things about its environment and actuators are types of motors. The sensors macro file contains the following:
  - lidar_macro - A simulation of a lidar sensor which shoots raycasts in a circle
  - imu_macro - A simulation of a sensor that gives a robot's acceleration and velocity with simulated uncertainty

  The actuators macro file contains the following:
  - v5_motor_macro - A generic motor with a number of settings similar to an actual motor
  - joint_pid_macro - A generic motor with a PID loop, with settings similar to an actual PID motor

Code Snippets / Commands Used

```
./scripts/launch_sim.sh

world\_file = os.path.join(home\_dir, "VEXU\_GHOST", "04\_Sim", "ghost\_sim",
    "worlds", **INSERT WORLD NAME HERE**")
```

Powering Jetson Orin Nano Over Battery

Timestamp: [11/14/24] - Evan Roegner
Attending:

Objective / Focus

The goal for this was figure out a method of powering the Jetson Orin Nano off of a 5V USB Power bank

Questions Explored

- What ports on the nano accept power?
- What connectors or cables are needed?
- How much power does the Nano need?
- How can USB-A/USB-C power out be converted to a standard that the nano accepts?

Steps Taken / Methodology

1. Nano usb-c plugged into 15V=3A power
2. Watched NVIDIA Jetson on Battery Power video
3. Searched for 5V to 19V step-up converter
4. Looked into potentially using POE

Findings / Insights

- Our battery outputs at 5V, but the nano needs 9V-19V
- We can convert between voltages with a step-up converter
- Most step-up converters on the market that convert from 5V usb to 9V-19V barrel only provide up to 10 watts, which is not enough
- If we get an expansion board, Power over Ethernet is possible, but likely impractical
- Absolute minimum amount of power would be 15 Watts, but more is recommended
- Inner barrel positive, outer is negative

Items needed

- 5V USB -> 9V-19V 5.5mmx2.5mm Barrel, >= 20 Watt output
- Alternatively, generic 5V->9V-19V step-up adaptor with correct connections soldered on would work

Challenges / Issues

- Need to find an applicable power adaptor, or potentially create our own with a generic step-up adapter.

Appendices

This section contains additional information that may be useful for understanding the contents of this notebook. This includes code snippets, data tables, and other supplementary information.