Bob Jones University Robotics Team
Bruin 3

Date Submitted: May 14, 2022

Team Captain:
Esteban Pacquing | epacq326@students.bju.edu

2021-2022 Team Members:

Caleb Talley | ctall434@students.bju.edu
Caleb Whiteley | cwhit118@students.bju.edu
Christopher Zuehlke | czueh332@students.bju.edu
Conrado Ferro | cmart375@students.bju.edu
Conway Duncan | cdunc249@students.bju.edu
Debanhi Flores | dflor912@students.bju.edu
Garrett Jones | gjone119@students.bju.edu
Joshua Heinrich | jhein429@students.bju.edu
Landon Goetz | lgoet353@students.bju.edu
Marcela Barrera | mmart372@students.bju.edu
Steven Platt | splat433@students.bju.edu

Faculty Advisors:
Jeff King | jeking@bju.edu

Statement of Integrity:

I certify that the design and engineering of the vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Faculty Advisor: Jeff King, PhD
Associate Professor
Department of Engineering
Bob Jones University

Faculty Advisor Signature: [illegible] Date: 5-14-22

INTRODUCTION

Students from Bob Jones University have worked to implement and integrate several subsystems to transform a Polaris GEM e2 vehicle into an autonomous vehicle named Bruin 3. This has required work from many different fields of engineering to design the different parts of the vehicle. Students from BJU have been working for several years to transform this vehicle to have fully autonomous capabilities. In 2021 and 2022, students replaced the steering and the braking systems with a better design, as well as getting all the software working. This report focuses on the designs of Bruin3 over the 2021-2022 school year.

ORGANIZATION

Table 1 lists members of the project for 2021-2022 and the number of hours contributed to the project for a total of 1200+ hours.

Table 1. Team Members and hours worked.

Name	Hours
Caleb Talley	90+
Caleb Whiteley	90+
Christopher Zuehlke	125+
Conrado Martinez Ferro	90+
Conway Duncan	90+
Debanhi Flores	90+
Esteban Pacquing	150+
Garrett Jones	150+
Joshua Heinrich	90+
Landon Goetz	90+
Marcela Martinez Barrera	90+
Steven Platt	90+

DESIGN ASSUMPTIONS AND DESIGN PROCESSES

The starting design assumption for the team was that the IGVC rules from 2021 would be appropriately representative of the 2022 rules and represented a good set of user requirements for the vehicle's design. Additionally, interviews with members who were on the 2021 IGVC team yielded some specific areas of improvement to focus the team's attention. After discussion with faculty, the team decided to redesign the braking system and the mechanical steering system. This was accomplished by dividing the overall team into two hardware design teams in the fall and a software design team in the spring.

One significant design assumption or constraint of Bruin3 is that any design changes to the vehicle that resulted in permanent damage or reconfiguration must be minimized. The goal is that the vehicle can be restored to OEM condition in the future with minimal effort.

The design processes used by the student teams during the 2021-2022 school year included standard systems engineering processes such as Requirements Definition, Work Breakdown Structure (WBS), and GANTT chart scheduling. Each sub-team was responsible for defining the key requirements and performance parameters for their respective system, estimating the work required using a WBS with personnel and durations for each task. The WBS was then used to create a planning schedule using a

GANTT chart. The schedule was reviewed during the weekly team meetings with the faculty mentor and any delays were discussed.

The software team used GitLab as a version control system, adopting a standard git workflow that emphasizes configuration management. Within git, a master branch contains the official copy of the Bruin 3 software. This software must be modified slightly for each computer on the vehicle, so each computer has its own secondary master branch. Developers are not allowed to make changes directly to any of the master branches. Instead, development takes place on smaller branches called feature branches. Once the changes on these branches have undergone a review and approval process, they are integrated into the appropriate master branch. This process facilitates configuration management by traking every software change and encouraging code review from other team members.

In 2018 and 2019, students designed most of the hardware and mechanical components of the vehicle. Thus, much of the basic mechanical design for Bruin3 was already in place from the progress made in past years. The 2020 team laid out much of the groundwork for the software. The design process for these components is covered in detail in the 2019 IGVC design report and the 2021 IGVC design report, and many of the details are parallel in this report. The old design had a few unresolved problems. Since the steering system and the braking system both failed to work effectively, the 2021 fall semester team split into two teams and worked to improve or change the systems. The steering was completed in the fall, and the 2022 team picked up where the 2021 team left off and completed the brake system. In addition, they continued to improve the RTK software, writing interfaces for the new hardware systems and developing the RTK integration to an operable state.

INNOVATIONS

Speed Sensing (Optical Flow Odometer and CAN Speedometer)

Typically, rotary wheel encoders are used to determine the speed of the vehicle, but Bruin 3 does not have any wheel encoders, so we had to come up with some replacement sensors to determine speed. First, we used a PX4 Flow Optical Odometer to provide feedback on the speed of the vehicle. Mounted on the rear computer compartment, the odometer determines the speed of the vehicle by capturing images of the ground, rather than with a rotary encoder on the wheels as is traditional. The design includes a CAN bus to obtain speed measurements. This CAN bus received feedback from a speedometer, and we were able to publish that speed information on a ROS topic.

RTK

The main innovation on Bruin3 is the use of RTK (Robot Technology Kernel) software. This software was provided by United States Army CCDC Ground Vehicle Systems Center (formerly TARDEC) and integrated into our vehicle, as part of a grant to help develop RTK. The current vehicle fully uses RTK with a minimum of custom nodes. RTK works alongside the software designed by the team members, but the difference between them is that RTK is the brain that tells the vehicle what to do. According to the information or input that comes in, RTK decides what the next action should be and then the system designed by the team sends signals to the various parts that control the vehicle. Integrating RTK into the vehicle was a challenging task that led to multiple innovations. Several components of RTK were designed for specific sensors, which were not available on Bruin 3. We had to redesign many of the low-level topics coming into RTK and simulate some others that were not being produced. In addition, the entire Drive-By-Wire system within RTK had to be redesigned in order to work with our vehicle and its actuators.

Steering System

The steering team addressed the issues of insufficient motor torque and gear deformation in the steering system. The stepper motor that controls the position of the steering wheel was replaced with a more powerful brushed DC motor, and the 3D-printed pinion and spur gears were replaced with manufactured gears. This did require a motor mount redesign to account for different gear sizes and the new motor. The steering system is protected from exceeding the wheel's turning limit through software that prevents the wheel from approaching the limit on both sides by 5%. The design is further detailed in *Mechanical Design* (pg. 5).

Braking system

The brake team addressed the issue of having inefficient brakes. They reversed the braking system to the original equipment manufacture design and utilized a DC worm geared motor to control the state of the brakes. The motor drove a spool that connected to the brake pedal with a cable. This design gives the user the ability to control the brakes in both autonomous and manual modes. The design is further detailed in *Mechanical Design* (pg. 6).

MECHANICAL DESIGN

Overview

The rear of the vehicle is divided into two sections, the bottom rear and the top rear. The bottom rear of the vehicle contains the PX4-Flow optical odometer . A wooden plate sits on top of the back-chassis frame and supports the two pieces of equipment. The top rear section contains the two CPUs (Fanny and Freddy), a 16-port ethernet box, four power converters, the IMU, and the Lidar ethernet box.

The following sensors are located on the top of the vehicle: LIDAR, Mako camera, GPS unit, and a light beacon. These are attached on top of a metal plate that is secured with brackets via the T-slot feature on the sides of the car.

The front of the vehicle contains the RADAR and the stereo camera. The RADAR is mounted to the front, diagonal beams of the frame. The stereo camera is attached to the vehicle right below the windshield in the charging port area.

Under the hood of the vehicle, several components of the braking system are mounted. The DC worm gear motor used to actuate the brakes is mounted on the left side of the vehicle to be colinear with the path of the brake pedal. In addition, limit switches are mounted to the motor, as shown in Figure 1. The SPDT (single pole double throw) relay to reset the brake pedal's position is located on the right side of the vehicle.

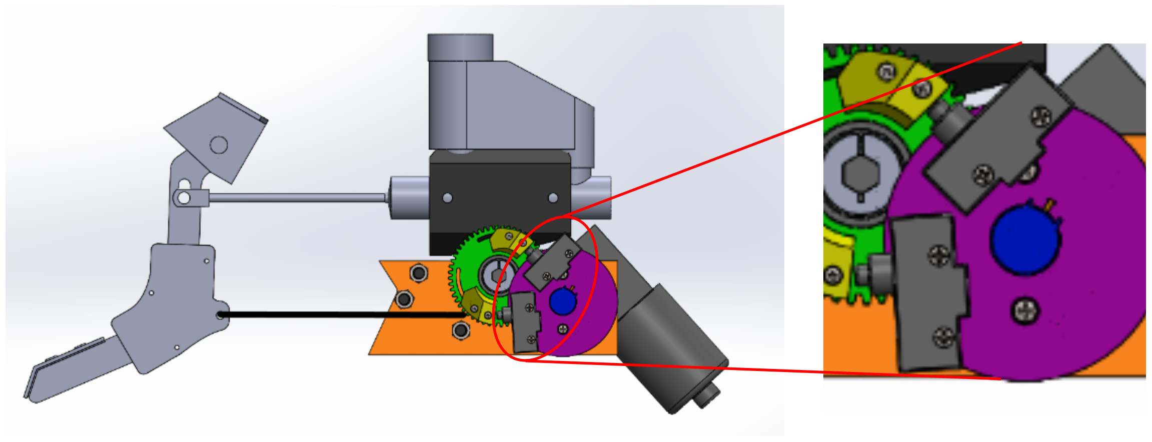

Figure 1: Braking System CAD Model (Limit Switches Emphasized)

Drive-by-wire kit

The team designed and installed their own drive-by wire-kit allowing the computer to control the vehicle's steering, brakes, and acceleration pedal. The system consists of three actuators. The brake actuator is a DC worm gear motor which depresses and releases the brake pedal to activate the OEM hydraulic brake system =. A DC planetary geared motor turns the steering wheel. Finally, an electronic accelerator pedal interface was created by the team that produces the same signals as the accelerator pedal from the factory car.

Suspension

The 2018 GEM e2's front suspension is a MacPherson strut, and the rear suspension is an independent trailing arm. No changes were made to the vehicle's stock suspension.

Weather proofing

Bruin 3 must be protected from severe weather. Concerning the sensors, the three cameras, LIDAR, and Radar, GPS unit, and IMU are self-protected, and thus require no additional protection . The enclosed sensors are the DAC, PX4Flow odometer, and the steering equipment. In the back, a waterproof box protects the vehicle's batteries and various electronics. Soft doors and a back window were purchased to protect the rest of the vehicle.

Steering System

For the design of Bruin 3, we wanted to be able to control the steering through a ROS node. We did this using a DC planetary geared motor with a Phidget DC Motor Controller connected to *Freddy* (see *Electronics Suite Description* pg. 7), which runs a custom node called "steer_drive" (see *Steering and Throttle Interface* pg. 11). This node controls both the steering and the throttle for Bruin 3. The motor drives a pinion gear to rotate the spur gear mounted to the steering wheel shaft as shown in Figure 2. The position of the steering wheel is read by a linear potentiometer located on the front axle. Using this feedback, the motor will rotate towards the desired position. As the position of the steering wheel approaches the desired location, the motor will slow down and stop at the approximate position, allowing for a range of error so that the motor isn't continuously running to find the exact position. To prevent the steering wheel from hitting the edge of the steering range, a function that caps the direction input at 5%

from either edge is integrated in the "steer_drive" node. The gears are enclosed in a plastic 3D-printed casing, shown in Figure 3, as a protective measure for the user.

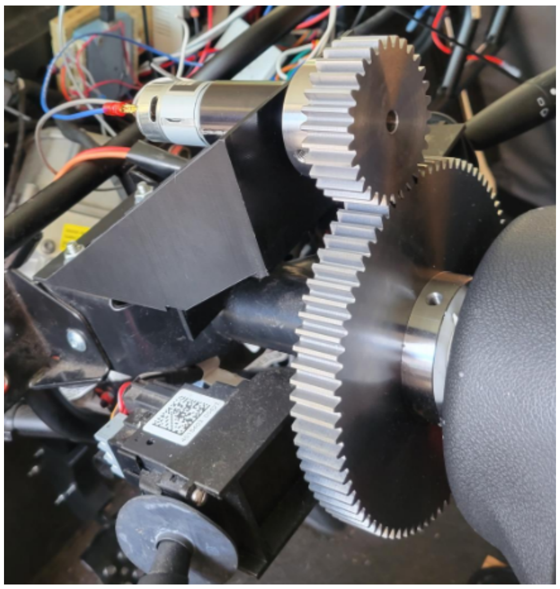
Figure 2. Steering Spur and Pinion Gears

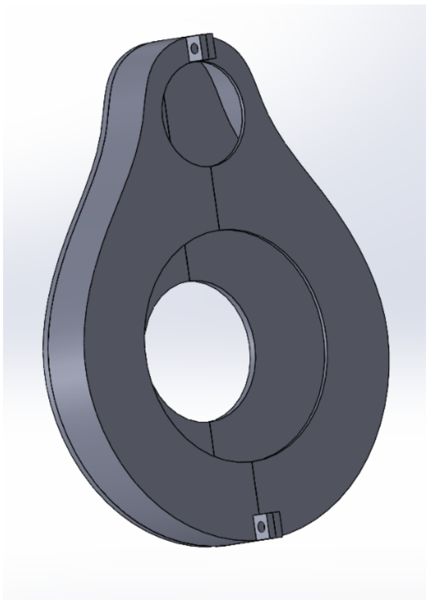
Figure 3. Protective Gear Casing

Braking System

For the design of Bruin 3, we wanted to be able to control the brakes through a ROS node. We did this using a DC worm gear motor with the RoboteQ Motor Controller being connected to *Freddy,* which runs a custom "/brake_control" node (see *Brake Controller Interface* pg. 11). The motor is connected to the brake pedal through a cable that will either depress or release the pedal. The motor controller simply changes the direction of the motor rotation and uses the normally closed limit switches as a rudimentary two state controller, BND (brake not depressed) and BD (brake depressed). The switches are pressed by using protruding columns on a spur gear as shown in Figure 4. A DPDT relay is used to reset the brake's state to BND when the vehicle is in manual mode.

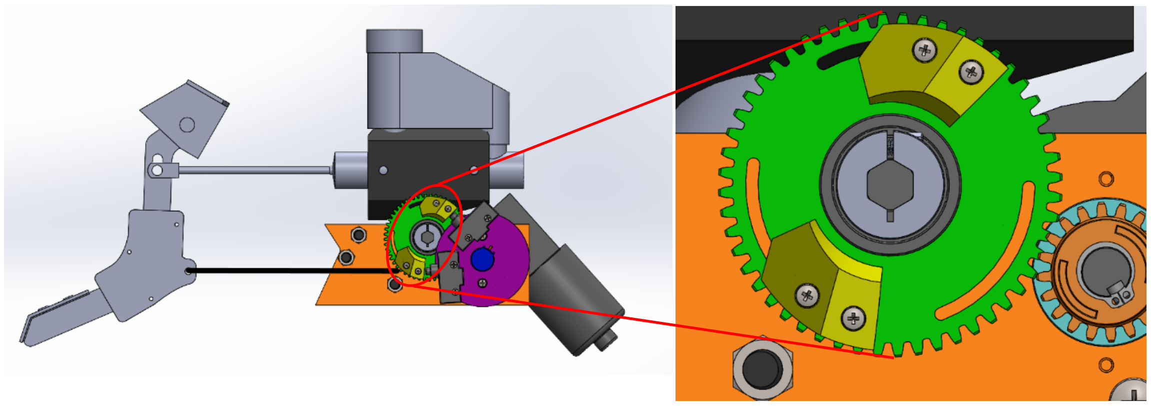
Figure 4. Braking System: CAD Model (Protruding Columns Emphasized)

DESCRIPTION OF ELECTRONIC AND POWER DESIGN

Overview

The vehicle uses a 48V Battery Pack as power source and various converters that power the computers, sensors, and mechanical systems. Additionally, a 12V battery pack serves as a backup battery for the braking system in E-Stop.

The vehicle consists of three main computers and sensors. The sensors include cameras for obstacle detection and localization sensors.

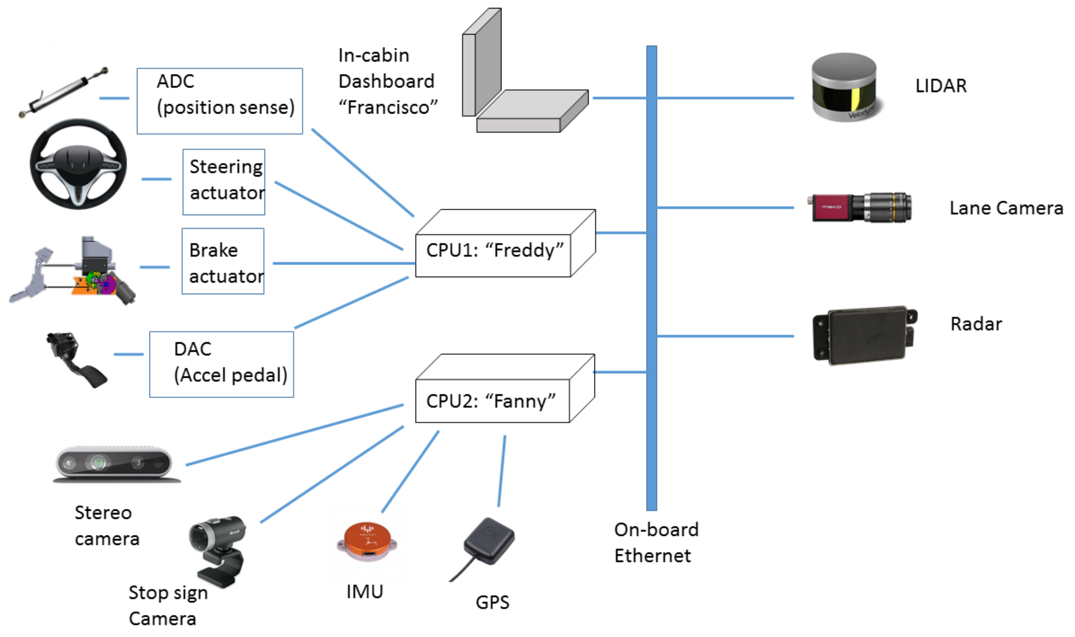

Figure 5. Block Diagram of the Control System.

Power Distribution System

There are two 48-to-12V DC-to-DC converters from which the computers are powered. The computer that processes images, "Fanny," requires a 12V power supply and draws 10A. All the cameras are connected to this computer. The sensors require 12V power supply and draw 3A. The other computer, "Freddy," receives feedback from the other sensors and sends signals to the motor controller and actuators which requires a 12V power supply and draws 5A. There are two additional 48-to-12V DC-to-DC converters from which the steering system, the brake system, and the e-stop are powered. The steering system requires 12V power supply and draws a maximum of 5A. The braking system requires 12V power supply and draws a maximum of 6A. The power distribution is demonstrated in Figure 6.

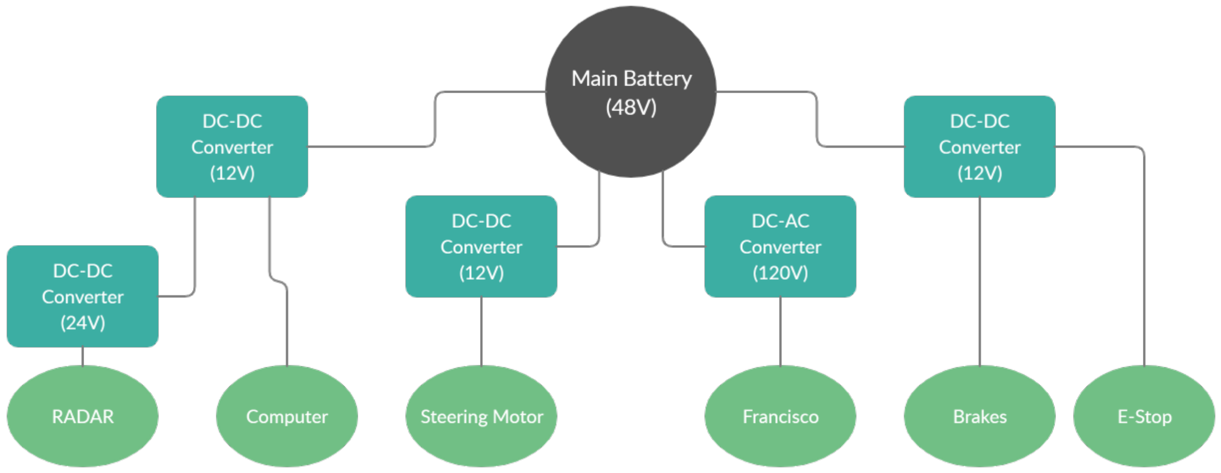

Figure 6. Power Distribution Diagram.

Electronics Suite Description

Computer Hardware

1. *Freddy* is a LINUX PC that runs the actuation nodes.
2. *Francisco* is a LINUX laptop that runs the high-level RTK nodes and provides a software dashboard in the cab of the vehicle.
3. *Fanny* is a LINUX PC that runs sensors and localization nodes.
4. *Whyme* is a Windows laptop that runs the WMI (Warfighter Machine Interface) and can be used by the vehicle's occupants or a remote operator.

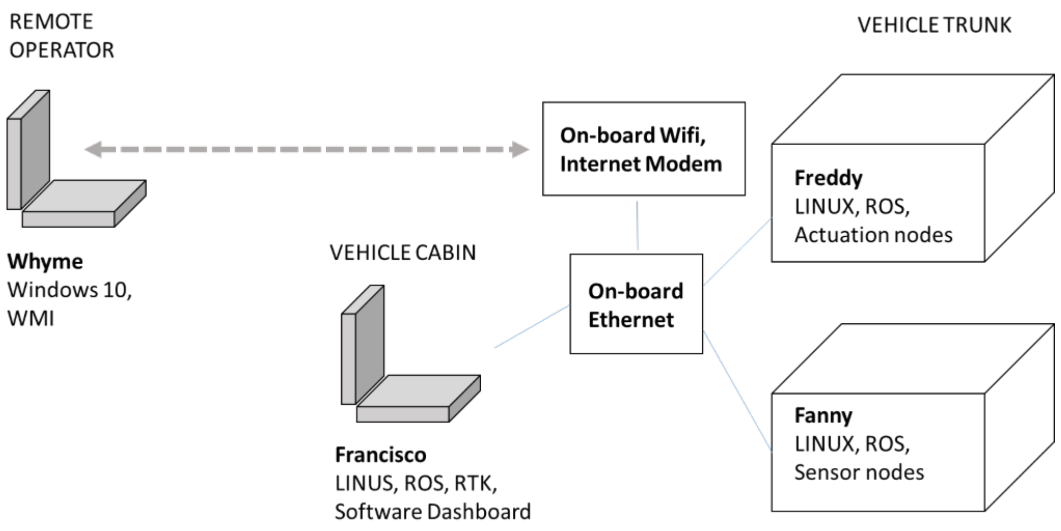

Figure 7. Computer Hardware Connections.

Sensors

1. Cameras
    a. Lane detection camera
    b. Stereo camera
    c. Road sign detection camera
2. LIDAR
3. RADAR
4. Localization sensors
    a. GPS sensor
    b. Odometry sensor
    c. IMU (Inertial Measurement Unit)

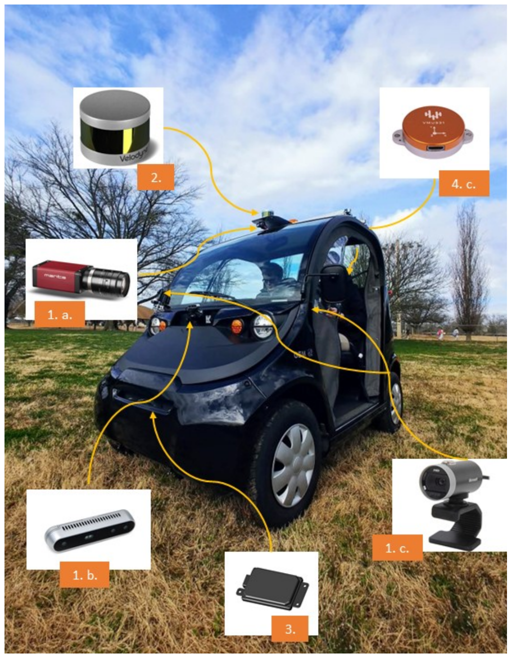

Figure 8. Sensor locations in the vehicle.

Safety Devices

Fuses are inserted between the DC-to-DC converters and computers to prevent short circuits. An E-Stop system is also implemented in the case that the vehicle must be shut down immediately. The back-up battery is integrated into the braking system circuit in the case of failure. The brakes utilize a series of SPST relays to determine the state of the system between the Automatic state and the E-Stop state. When the E-Stop state is triggered, the SPST relays will lose power and switch positions to send current through the motor using the back-up battery to ensure a brake depressed state. There is a 20A fuse to protect the brake system and a 5A fuse to protect the steering system.

CAN-bus Speed Sensor

For Bruin 3's speedometer, we used a Kvaser Leaf v2 to collect and process Bruin 3's CAN bus speedometer data. We implemented Kvaser's Linux drivers to receive the Leaf's CAN data stream. The Kvaser then reads and isolates Bruin3's speed from the stream and sends it to a publishing ROS through a Linux FIFO pipe. Once the publishing ROS node receives Bruin3's speed data from the pipe, it translates the scalar speed values to an odometry message and publishes it to the RTK topic /localization/speed.

Autonomous Transmission Control

Forward-Neutral-Reverse mode was previously controlled through a mechanical switch. Each mode was activated by connecting different ports to create a short circuit to ground. Each of the voltage probes represents a wire that tells the motor to be in reverse mode, neutral mode, or forward mode. After finding a schematic of what was going on inside the switch and the car, we were able to reverse engineer it. The two switches on the left indicate the Forward-Neutral-Reverse (FNR) switch in the car. The 3 voltage probes on the right side indicate the 3 sensor wires to indicate which gear we are in. When the sensor wire is at 0 volts, then we are in that gear. From top to bottom the sensor wires indicate Reverse, Neutral, and Forward gears. Using this information, we were able to control the transmission autonomously through a relay board. We designed a ROS node to change the transmission, based on what RTK requested.

SOFTWARE STRATEGY AND MAPPING TECHNIQUES

Overview

Our software is based on RTK (Robotics Technology Kernel) which is built on top of the ROS (Robot Operating System) framework which is an open-source software package maintained by the Open Source Robotics Foundation (OSRF). In ROS each major function is managed by a separate smaller program called a node. The nodes communicate between each other and perform the various tasks needed for our vehicle to function. RTK is a compilation of ROS nodes managed by the Army CCDC Ground Vehicle Systems Center (GVSC). Our team wrote several custom ROS nodes to implement different features.

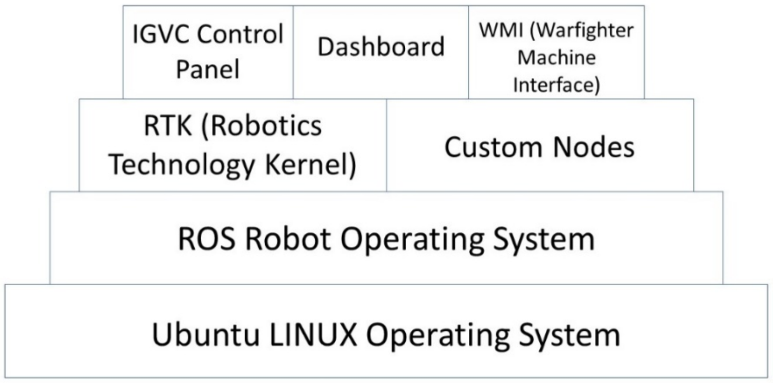
IGVC Control Panel
Dashboard
WMI (Warfighter Machine Interface)
RTK (Robotics Technology Kernel)
Custom Nodes
ROS Robot Operating System
Ubuntu LINUX Operating System

Figure 9. Software Architecture.

First, the localization subsystem of RTK receives sensor inputs from the Sensor subsystem. RTK expects a standard localization sensor suite, including a specific Novatel GPS, Microstrain IMU, RTD rear wheel encoders, and KVH single axis gyroscope. We did not have all these hardware components available, so the sensor part of RTK had to be entirely rewritten. We had a Garmin GPS and used the GPS driver gps_umd from SWRI Robotics to generate sensor messages. In addition, we used the IMU VMU931 from Variense. Variense provides driver code that we used to generate sensor messages from the IMU. We also used this driver to get simulated gyroscope data. Bruin3 does not have wheel encoders, so we used CAN bus messages and a px4flow camera to get speed measurements. The CAN bus reports data from Bruin3's speedometer. The px4flow camera optically detects how fast the ground is moving relative to the camera and uses that data to calculate speed. We used the px4flow ROS node to obtain speed measurements from the flow camera. Most of the Bruin3 sensor messages were not formatted correctly for RTK, so we wrote an additional ROS node to transform each of these topics into the correct RTK format.

Second, RTK has an IOP Bridge subsystem connecting to the OCU (Operator Control Unit). We used the standard OCU for RTK: the WMI software (Warfighter Machine Interface). The IOP Bridge and WMI use JAUS (Joint Architecture for Unmanned Systems) messages to communicate information back and forth.

Third, the motion execution and low-level CAN subsystems had to be rewritten. These are the DBW (Drive by Wire) system that RTK uses to control the low-level actuators on the vehicle. In standard RTK, the motion execution system sends DBW commands to the CAN Bridge to control the vehicle. The actuators on Bruin 3 are very different than the ones that RTK expects, so we had to completely redesign the DBW system. We took the four commands from the Motion Execution System: brake, throttle, steering, and transmission inputs, and wrote our own ROS nodes to implement those commands with the actuators on Bruin3. RTK is also designed to expect several messages from these low-level components indicating that everything is working fine. We had to simulate many of these messages with a stub in order to make the RTK system happy.

Finally, there were several other details in the main part of RTK that we had to change in order to get everything working properly. One of these was the VMS (Vehicle Management System). It was expecting a lot of messages from different parts of the RTK system that weren't being published. We had to change some of those things and simulate different topics in order to get the VMS to do its job.

Obstacle Detection and Avoidance

We used three major components for obstacle detection: a stereo camera, LIDAR, and RADAR.

1. LIDAR is used to create a 3D map of the area around the vehicle as seen in Figure 10.
2. RADAR senses certain obstacles in front of the vehicle, for example, pedestrians and other vehicles.
3. The stereo camera detects obstacles that are in front of the vehicle as seen in Figure 11.

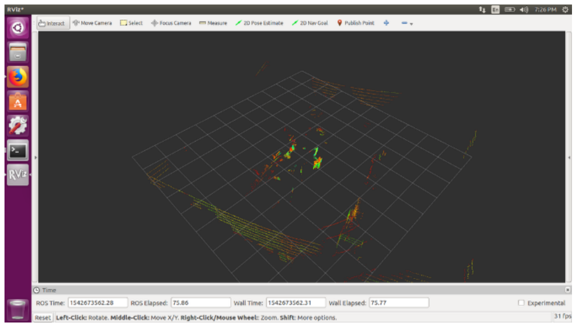

Figure 10. LIDAR sample.

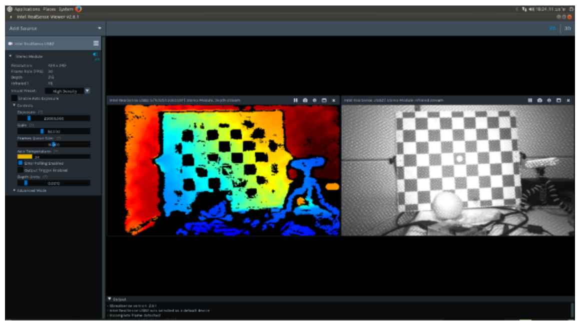

Figure 11. Sample depth image from the RealSense stereo camera.

Lane Following

A camera on the top of the vehicle scans for road marking lines. These marking lines are turned into 3D obstacles and added to the cost map so that the path planner can navigate between them. A high response speed Mako camera was used to gather the information.

Map Generation

The world model module of RTK combines data from the LIDAR, stereo camera, and other sensors to generate a map of the world around the vehicle with the obstacles and other parts of the course. This world model then generates the costmap for the entire situation. The generated costmap indicates the riskiness of different paths the vehicle can take. See example cost map in Figure 12.

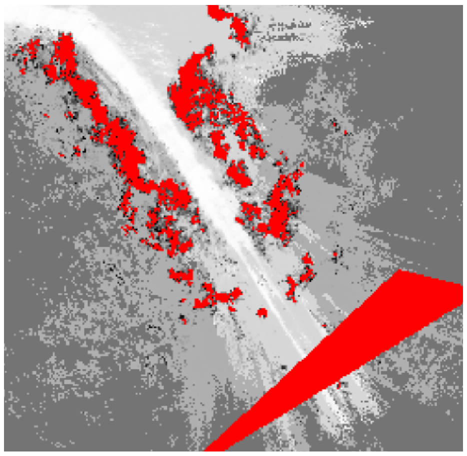
Figure 12. Example of a costmap generated by RTK.

Goal selection and Path Generation

The path planning module of RTK uses the costmap and the A* algorithm to find the path of least total cost. The RTK module uses the sensor information to determine where the vehicle cannot go, like going off the road or crashing, and associates that with a very high cost. The clear road ahead of the vehicle will be assigned a very low cost and that is the path that the vehicle will take. The vehicle will then use this path that RTK plans to drive the motor and steering of Bruin 3. The WMI user interface allows the user to provide their desired destination in the form of a waypoint. The vehicle requires GPS to perform waypoint navigation. The vehicle stays in the lane while the GPS indicates the location to direct the system to head in the right direction.

Steering and Throttle Interface

We wrote a custom ROS node named `steer_drive` to interact with the main drive motor, steering sensor, and steering motor in the DBW system. These interactions are illustrated in Figure 13. To control the drive motor, the node produces an analog signal ranging from 1.53 V to 1.6 V to simulate the acceleration pedal. This signal is generated using a DAC. In addition to this primary signal, the node must produce a second signal whose voltage is 5/9 that of the first. This requirement is built into the vehicle as a safety mechanism to ensure the accelerator is working properly.

The `steer_drive` node operates the steering motor through a digital motor controller. It also collects steering feedback from a linear potentiometer that measures the displacement of the front wheels of the vehicle. This potentiometer enables closed-loop steering control.

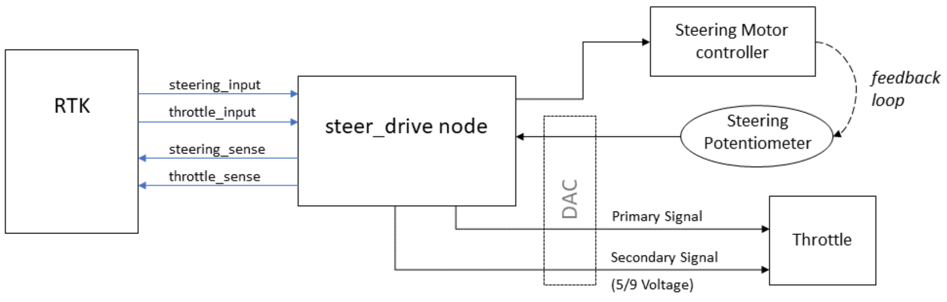

Figure 13. Block Diagram for the Steering and Throttle Interface.

The `steer_drive` node runs on Freddy, which is connected to the steering and drive systems.

Brake Controller Interface

The brake motor controller is connected to Freddy, which runs a custom “brake_control” node. This node subscribes to several topics to determine the wanted state of the brake and communicates to the brake actuator motor controller via a serial port.

The brake control node listens to the “stop_state” ROS message and bruin_manual_mode ROS topics. RTK publishes the “stop_state” which is used to communicate its current operating state to relevant subnetworks in Bruin3’s ROS system. The ROS topic “bruin_manual_mode” allows the vehicle to know if it is under operator control by manually setting it to a boolean value of TRUE or FALSE. Once the desired state has been received, the node sends a command to the motor controller commanding whether the brake should be on or off. Using this logic, the brake will only be used in autonomous mode when the vehicle must come to a complete stop. Slowing will be achieved naturally with the aid of regenerative braking without using the hardware brakes.

DESCRIPTION OF FAILURE MODES, FAILURE POINTS AND RESOLUTIONS

Vehicle failure modes, points, and resolutions

If the steering actuator subsystem fails, the vehicle may attempt to drive into obstacles. The obstacle sensors should detect the obstacles. It will attempt to steer around the obstacles, without success, and then stop the vehicle when it becomes clear that a viable path is no longer available. The human safety driver is also responsible to observe the operation of the vehicle and intervene if the path is toward an obstacle. Although the human safety driver should be able to overcome the torque produced by the steering system’s motor, it is advised to only intervene in case of emergency while in autonomous mode. In the event the user needs to take control of the vehicle, he should activate either the E-Stop or switch the vehicle out of autonomous mode.

If the brake actuator subsystem fails, we are dependent on the human safety driver to stop the vehicle using the brake pedal. The user will be able to utilize the pedal in both autonomous and manual mode. In the unlikely event of failure in the brake pedal, the parking brake may be used by the human safety driver to stop the vehicle.

If the accelerator pedal actuator fails, the vehicle may accelerate out of control. The obstacle detection systems should intervene and attempt to brake the vehicle. If full acceleration and braking are both actuated at the same time, the brakes will be able to stop the vehicle but at a reduced rate. The human safety driver may need to intervene in this situation as well.

If the battery fails, we are dependent on the back-up battery. This battery will continue powering the vehicle estop which will be triggered by loss of power.

If the actuators fail, we are dependent on the e-stop to stop the vehicle, in order to avoid further complications.

If the e-stop communication fails, this will cause an e-stop.

If the communication between the computers fails, the vehicle will stop driving.

All failure prevention strategy

The vehicle health system can detect multiple failure points across the vehicle, and it will stop driving.

The vehicle operation currently requires a human safety driver in the vehicle at all times. The human operator can engage the estop at any time. The operator can brake and steer the vehicle manually in both autonomous mode and manual mode. Refer to the steering section of *Vehicle Failure Modes and Resolutions* (pg. 13) for manual steering in autonomous mode.

Testing

The e-stop system was tested and stops the vehicle. It also engaged when a wire was disconnected from the e-stop system.

The wireless estop system has a specified range of 600 feet, well beyond the IGVC requirement of 100 feet. This range has yet to be tested but will be tested before IGVC.

The torque required to override the steering actuator is given to be 11 Nm of torque which is achievable by the human safety driver. Refer to the steering section of *Vehicle Failure Modes and Resolutions* (pg. 13) for manual steering in autonomous mode.

The manual brakes were tested and are fully functional in autonomous mode.

The vehicle and our on-campus test track are modeled in the Gazebo simulation environment. The vehicle can be driven in the simulated environment using the same software as the real vehicle. The resulting vehicle trajectories can be compared.

The actual vehicle was tested on a grassy field on the BJU campus. On March 12, 2021, we successfully demonstrated driving to a waypoint.

Vehicle safety design concepts

The Polaris E2 vehicle meets all the safety standards for a low-speed electric vehicle (LSEV) including headlamps, tail lamps, stop lamps, reflectors, mirrors, a parking brake, a windshield and seat belts. We have not modified any of the safety features except the brakes as described below.

The maximum speed and path curvature are limited by software drivers with low-level control of the vehicle. These drivers will override any command from the navigation system to exceed the speed or curvature limits.

The vehicle includes a fire extinguisher as required by the IGVC rules.

Four on-board e-stop buttons and a wireless e-stop provide hardware shutdown of all the actuators. The estop buttons activate normally-closed switches, so any hardware fault in the system that results in an open circuit causes an estop.

The steering wheel and brake pedal are fully functional in autonomous mode, giving a safety driver the ability to control the vehicle manually at all times. With the current vehicle we intend to operate the vehicle only with a human safety driver in the driver's seat.

The braking system uses a back-up battery that is independent of the main power system to provide a brake depressed state in the event of an e-stop. The vehicle does not coast after e-stop and will stop even in the case of a total loss of primary system power.

SIMULATIONS EMPLOYED

Simulations in virtual environment

Within RTK's Operator Control Unit, WMI, RTK provides a sample simulation to test out the software and learn how WMI should work. We used a simulation to test the signals for driving the vehicle and testing how the RTK modes should work.

The vehicle is modeled in Gazebo for virtual simulation.

PERFORMANCE TESTING

Component testing, system and subsystem testing, etc.

The vehicle has a specified battery range of 20 to 30 miles. In a worst-case battery life test (hilly terrain, high speed stop-and-go driving) the vehicle reached a "low battery" level after 12 miles. This lowers the range between 8 to 18 miles but is more than adequate for IGVC.

The vehicle can climb a 6-degree (11%) slope easily.

All sensors have been successfully tested in ROS.

INITIAL PERFORMANCE ASSESMENTS

At the time of completion of this report, the RTK is not fully implemented in the vehicle, and the vehicle is not yet capable of performing the full set of IGVC tasks. All the subsystems are installed and tested individually.

MANDATORY UNIT TESTS

Unit test 1: Emergency stop

In the current state of the vehicle, we simulated an E-Stop command at 5 mph. The vehicle stopped 15 feet after the E-Stop was triggered in 3.2 seconds.