IEEE SoutheastCon 2019 Hardware Competition Team

AUBURN UNIVERSITY
SAMUEL GINN COLLEGE OF ENGINEERING

SPARC
Student Projects & Research Committee

Senior Design:
Matthew Castleberry, Joe Hinely, Josh Jablonowski, Nia Perkins

SPARC:
William Dovre, Alex Jones, Noah Niedzwiecki, Kevin Ogden, Rebecca Thomas, William Yates

Objective

Construct autonomous mobile robot to compete in 3 minute round
Scoring Criteria
- 5 points for leaving home base
- 10 points for returning to home base
- 25 points for raising flag
- Up to 15 points per debris move to colored corner
- 5 points per revolution around center

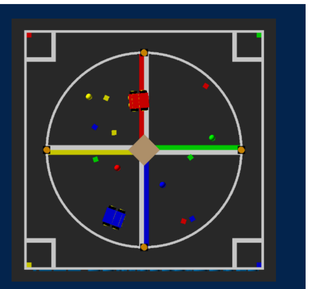

Figure 1: Vertical view of playing field

Mechanical

We constructed a 4 wheel differential drive robot. Each side has a motor powering the wheels with a chain and sprocket system. There is an opening with a movable flap in the center section on the bottom for the debris to be collected. The robot fits within the 9" x 9" x 11" size requirement. It was constructed with an 8020 aluminum frame and custom PLA+ 3D printed brackets.

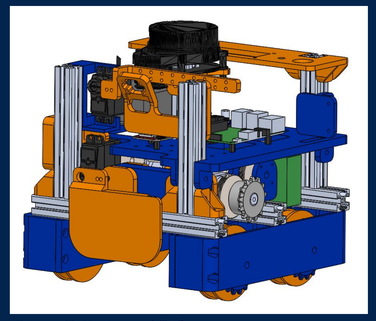

Figure 2: Mechanical CAD

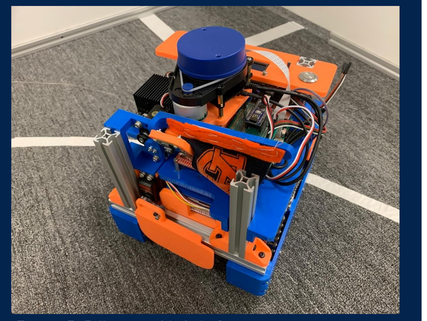

Figure 3: Fully assembled platform

Sensors

We selected 3 different sensors to help the robot be aware of its surroundings in the arena: a visual camera, a 2D LIDAR, and rotary encoders. The camera used is the Raspberry Pi Camera Module. The raw images are processed on entirely on a Raspberry Pi to find the debris objects on the field. The LIDAR is a relatively low cost 2D LIDAR system. The YDLIDAR uses a spinning laser rangefinder to return a 2D point cloud at about 8 Hz. The rotary encoders monitor the rotation of each drive motor. The LIDAR and encoders are used for localizing the robot in the field.

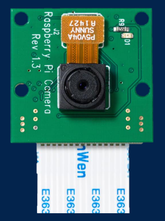

Figure 4: Camera module

Figure 5: YDLIDAR

Figure 6: Rotary encoders

Electrical Hardware

The robot is controlled using two coprocessing Raspberry Pi 3B+ connected via an ethernet cable. A Teensy microcontroller polls the encoders and sends the velocity information to the main Pi. Once the Pi's determine a desired path, an Arduino microcontroller instructs the motor controllers of the corresponding commands. Two different batteries are used to prevent the motors from causing a brownout situation for the Raspberry Pi's.

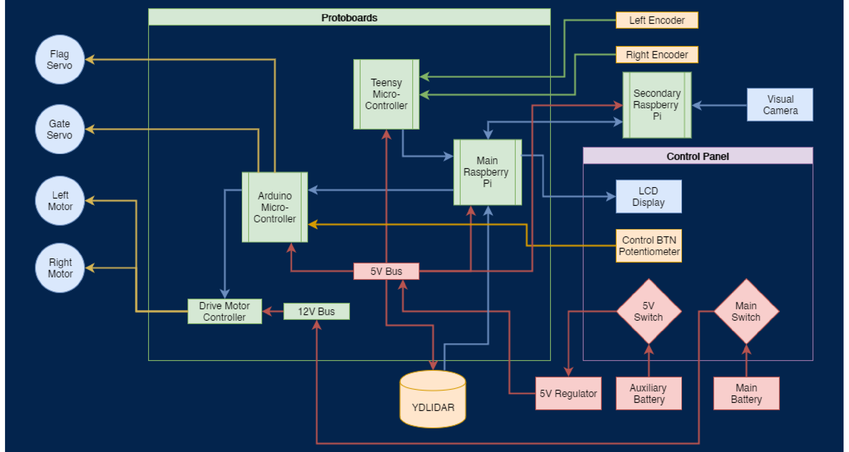

Figure 7: Simplified electrical hardware block diagram

ROS

Due to the intricate software required to process the selected sensors, we chose the open source framework Robot Operating System (ROS). It consists of a distributed network of nodes. By combining our custom nodes with publicly available nodes, we were able to implement a way to transform the raw collected data to commands to control the robot.

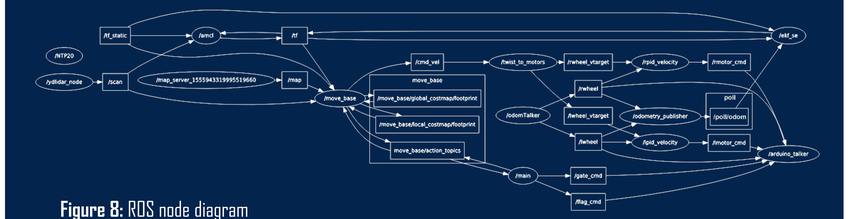

Figure 8: ROS node diagram

Localization

The LIDAR and encoder data is received on the main Raspberry Pi via USB. Through ROS, this data is processed along with encoder data to determine an x,y position and orientation of the robot with respect to the field. The Adaptive Monte-Carlo Localization (AMCL) algorithm compares the current point cloud to a prerecorded map.

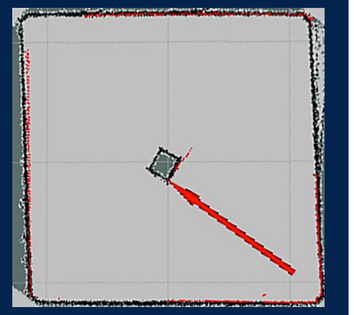

Figure 9: Robot position and orientation indicated on arena map

Object Detection

A visual camera image is processed using the computer vision library OpenCV. Through our OpenCV algorithms, the robot can determine an x,y position in meters of each debris object with respect to itself. This is possible because the debris has known sizes that can be compared to the recorded pixel sizes. The algorithm could determine the positions of all 4 colors at approximately 10Hz.

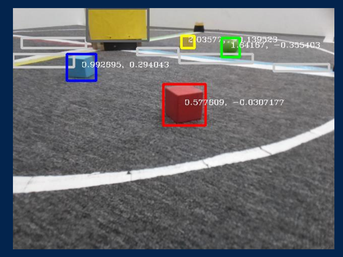

Figure 10: Objects identified and analyzed via OpenCV visual processing

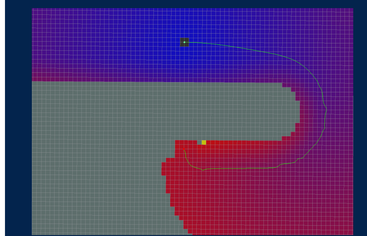

Figure 11: Example visualization of navigation pathfinding

Navigation

Once the robot has a coordinate of its current location and its desired location, it can then calculate a path to drive from where it is to where it wants to go. ROS has several different packages that we tried including Dynamic Window Approach, Find the Carrot, and Move Basic. Ultimately Move Basic was the easiest to implement but it is not as smooth as the alternatives.

Integration

While each individual aspect operated nicely when configured properly, the Raspberry Pi's were not computationally powerful enough to handle the load. ROS and OpenCV are very processing intensive. Adding a second Pi helped distribute the load but increased development time dramatically as well as reduced consistency. In order to properly accomplish the tasks with this approach, a more powerful processor is required such as an Intel NUC.

Figure 12: Field algorithm testing

Figure 13: Full SPARC and senior design team at SoutheastCon 2019 in Huntsville, AL