Autonomous Navigator Mobile Robot Upgrade

David Sansoucy

Sansoucy, David, "Autonomous Navigator Mobile Robot Upgrade" (2022). *Thinking Matters Symposium*. 27.
https://digitalcommons.usm.maine.edu/thinking-matters-symposium/2022/poster_presentations/27

Autonomous Navigator Mobile Robot Upgrade

By: David Sansoucy, Under Supervision of Dr. Carlos Luck

EGN402 Senior Design Project, Engineering Department, Fall 2021

Robot Arm Selection

While updating the platform in Spring 2020, Belle-Isle and Werner attached a MicroBot Teachmover robot arm, determined a control scheme for it in ROS, and built a GUI to send commands to it. However, that arm had several drawbacks, including its high weight and high power requirements. The main objective for this project was, keeping within the current hardware and software frameworks, to mount a modern, lightweight robotic arm onto the mobile platform and control it. The table in Figure 2 shows several candidate arms and their performance capabilities. The engineering decision matrix in Figure 3 was used to score each arm with regards to a particular attribute or capability. The Tinkerkit Braccio arm won with the highest score.

Arm	MicroBot Teachmover	Tinkerkit Braccio	PhantomX Pincher	Yahboom Dofbot
DOF	5	5	4	5
Max Reach (cm)	44.45	43	31	35
Max Payload, at distance (gram)	453.6, at max reach	150, at 32 cm	100, at 15 cm	200
Motor Type	stepper	servo	servo	servo
Gripper Type	parallel jaws	near parallel jaws	parallel jaws	near parallel jaws
Arm Weight (gram)	4000	792	550	1256
Cost, $	N/A	218.90	379.95	379.99

Figure 2: Robot arms comparison

	DOF	Max Reach	Max Payload	Power Required	Arm Weight	Cost	Total
Tinkerkit Braccio	3	3	2	3	2	3	16
PhantomX Pincher	2	1	1	2	3	2	11
Yahboom Dofbot	3	2	3	2	1	1	12

Figure 3: Engineering decision matrix

Electrical Changes

A powered USB hub was added to expand the RPi's IO capability and an electromechanical relay module was added. The arm and hub are powered through this module, so they don't waste power when not needed. Figure 4 to the right shows the arm's electrical panel and a schematic of its power system.

Abstract

The mobile robot platform has been developed over the course of 10 years at USM. This project aimed to build on the Robot Operating System (ROS) frameworks developed by the previous 2 teams with the main goal of mounting a modern robotic arm onto the platform. An arm was selected among several candidates and bench tested with an open-source joint angle control software package. The package was then integrated into the existing mobile robot's code. From a remote Linux PC, the arm can be controlled manually, or it can run programed movements defined in Python scripts. The arm mounts to the robot on a height adjustable and modular platform as shown in Figure 1. The arm gives future teams more capabilities as the robot can now manipulate objects in its environment.

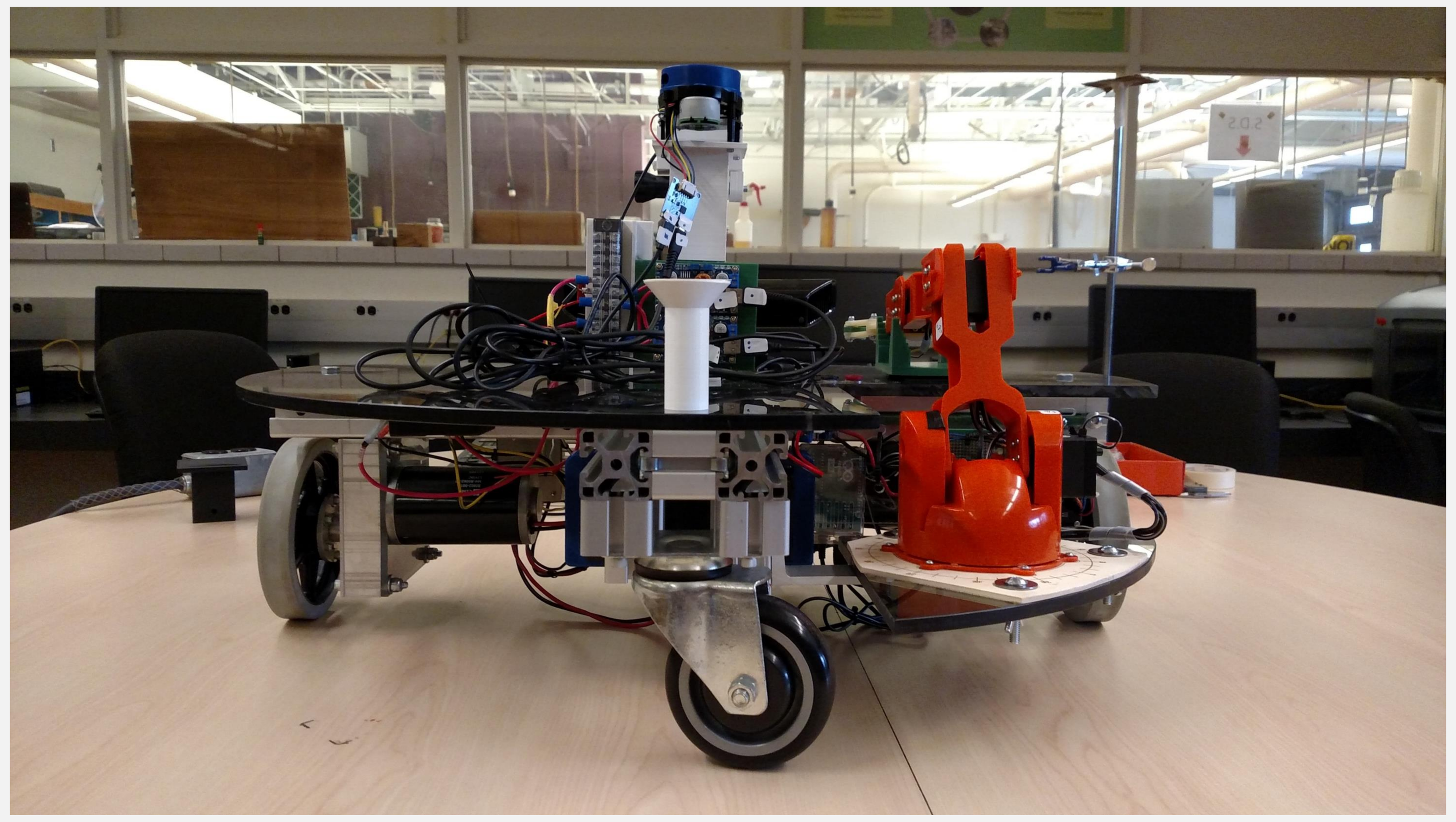

Figure 1: Braccio arm mounted on mobile robot, shown in the stowed position

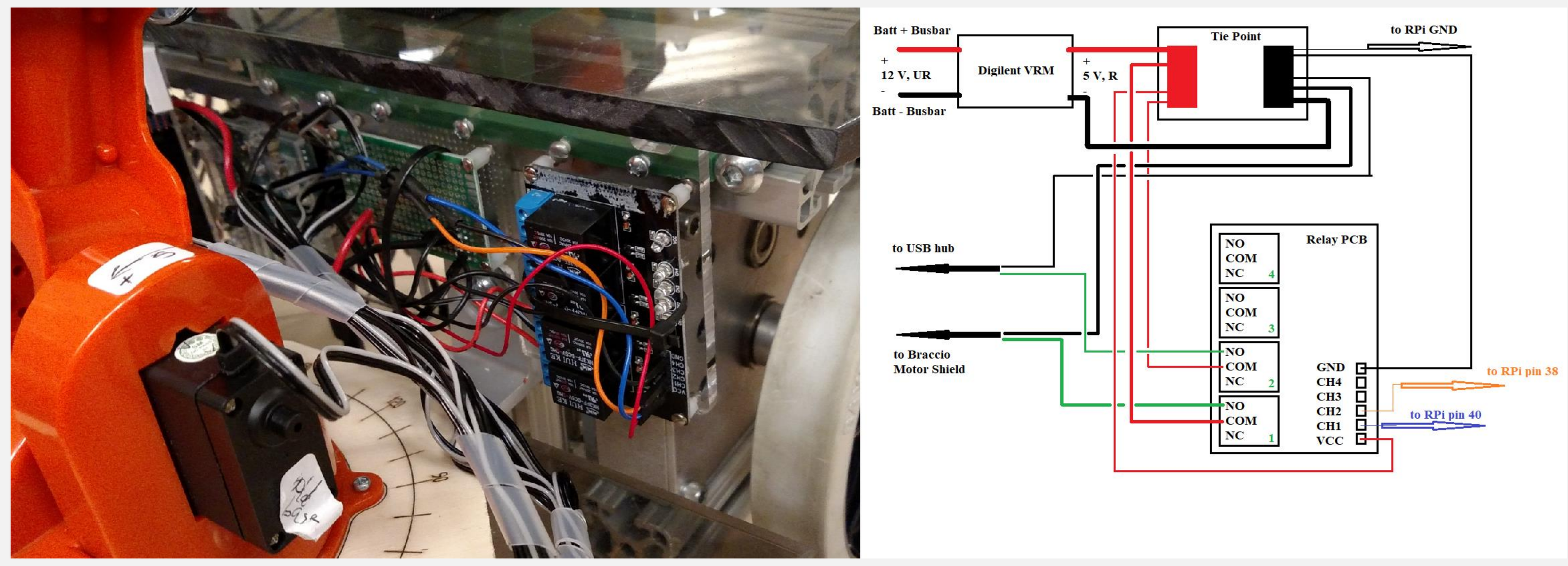

Figure 4: Arm's electrical panel (left) and schematic (right)

Next Steps

Rviz, MoveIt!, or Gazebo could be used to define a safe working envelope. A program that used inverse kinematics could pick up items that were recognized using the webcam.

Software Development

After the robot was assembled, the initial testing was done directly with the native Arduino code. All the joint motions were verified and some servos were adjusted as needed. Repeatability tests and power consumption tests were also done. An open-source GitHub package was the starting point for using ROS with the arm. This package enabled a Linux machine running ROS Kinetic Kame to control the Braccio using the RViz joint_state_publisher GUI as in Figure 5. Modifications were made to allow the package to run on a Raspberry Pi and to verify that arm movement commands sent wirelessly from the Linux PC correctly caused the Braccio to move. Finally, the code was integrated into the mobile robot's existing code and some demo programs consisting of Python scripts were made. Figure 6 shows the system architecture.

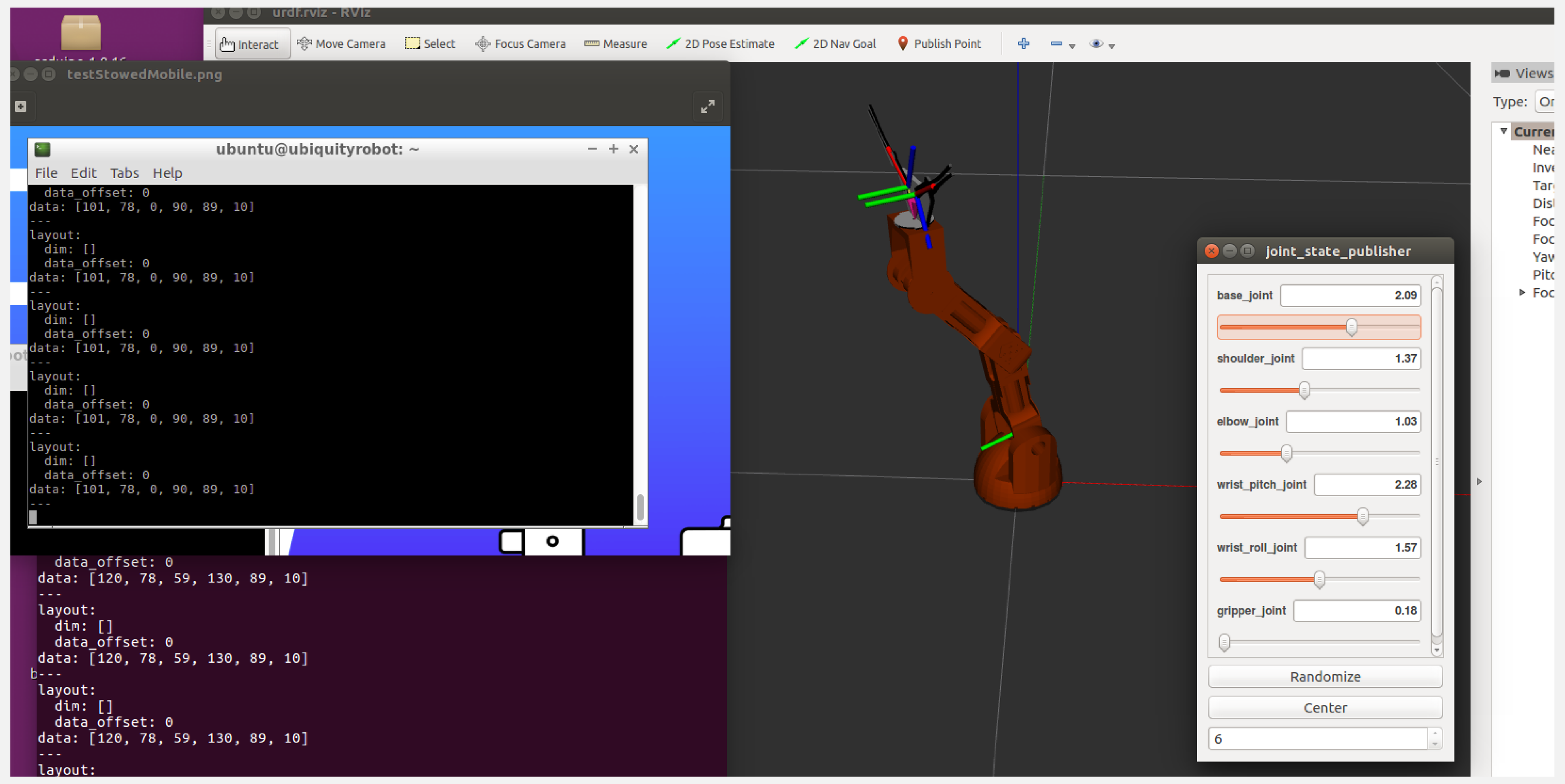

Figure 5: Rviz environment showing arm model and control GUI

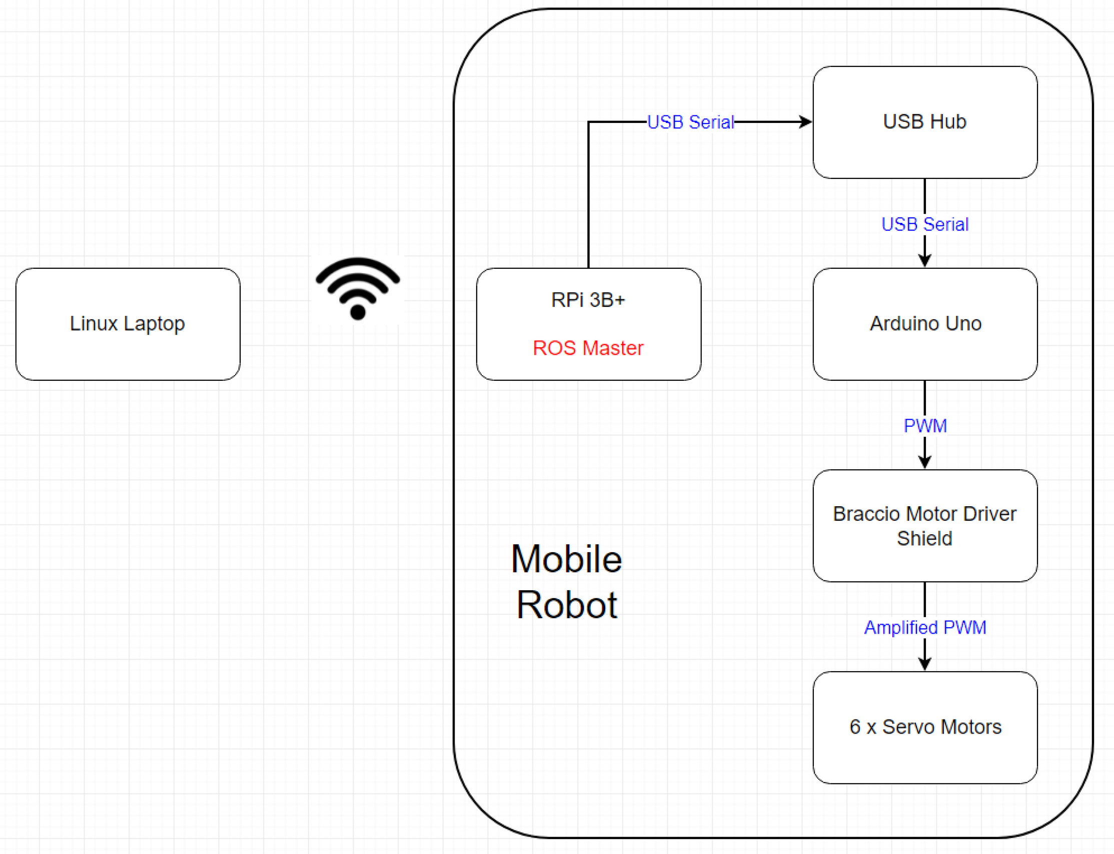

Figure 6: System architecture