Team 2

Ioannis Papakis

Jeeseop Kim

Ashrarul Haq Sifat

I. Design objectives and Constraints

Our unmanned ground vehicle is a platform provided by the Computational Systems Laboratory of Virginia Tech, for the course of Experimental Robotics. It is a pre-built metal structure with four DC motors coupled with each wheel.

The main purpose of the first project was to integrate into the existing platform mechanical and electrical equipment so that the vehicle can sense its environment, interact with it and be tele-operated.

The second phase focused on perception and improvement of the mechatronics system. The main goal was to use SLAM packages and map the environment as well as localize the robot in it. We also had to tag an AR code and track it with the camera. All these actions had to be included in RVIZ.

The final goal and competition included teleoperation, mapping, localizing, detecting the AR code and putting it on the map created by the robot.

Restrictions in the overall implementation had to do with space available, data transfer rate from mini pc to base station and processing power of the mini pc.

II. Hardware system

The basis of the mechanical system is the metal structure with the four motors and wheels. We incorporated there a 3D printed structure that has the purpose of accommodating the electronics and batteries as well as the robotic arm. The design is built in such a way that it can make use in the best way possible of the available space, so since space was limited, it was decided that the height of the structure can be utilized to accommodate the parts. The printed structure has three layers each of which serves to include every electrical component.

The electrical components include the batteries, one for the motor and the Lidar and one for the UGV PC. These are placed on the bottom layer so that they don't put pressure on the structure due to their weight. In the middle layer, the mini pc is positioned so that it easily accessible through ports and if needed easily removed. The top layer has a dual purpose of accommodating the Arduino and controller on the downside and robotic arm on the upside. All parts are installed inside or on top of the printed structure except for the camera and Lidar. The Lidar had to have a front view access and the camera should be able to be in a position that provides a high level view and is not obscured by other objects.

A significant improvement was the change of orientation of the arm by 180 degrees, in order to maintain the same workspace but at the same time have a better field of view with the camera.

Following are a depiction of the actual system and an illustration of the data and power transfer system.

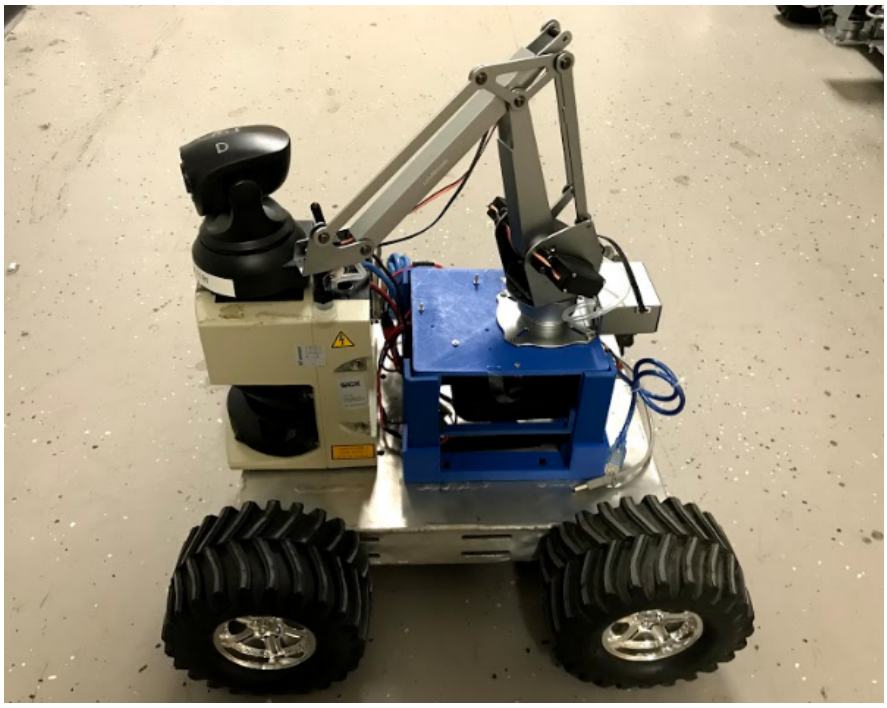

Figure 1: Left side view of UGV assembly

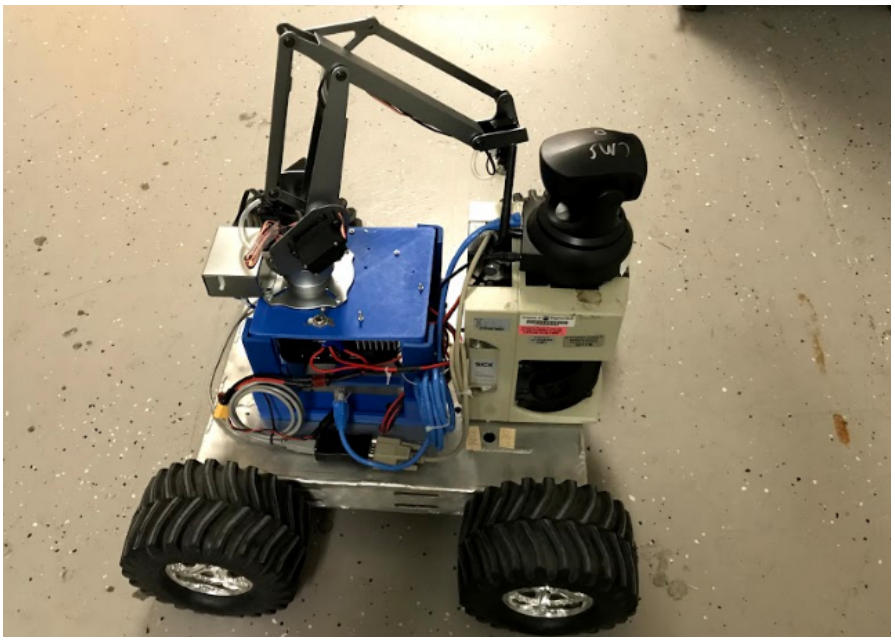

Figure 2: Right side view of UGV assembly

Figure 3: 3D printed structure with inside components

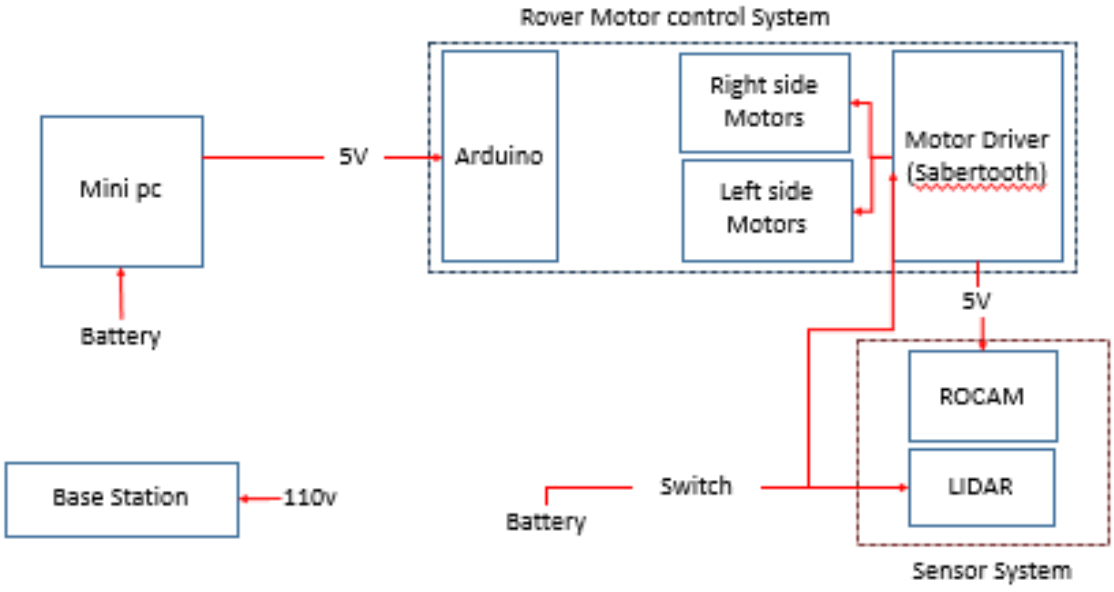

Figure 4: Power line system diagram

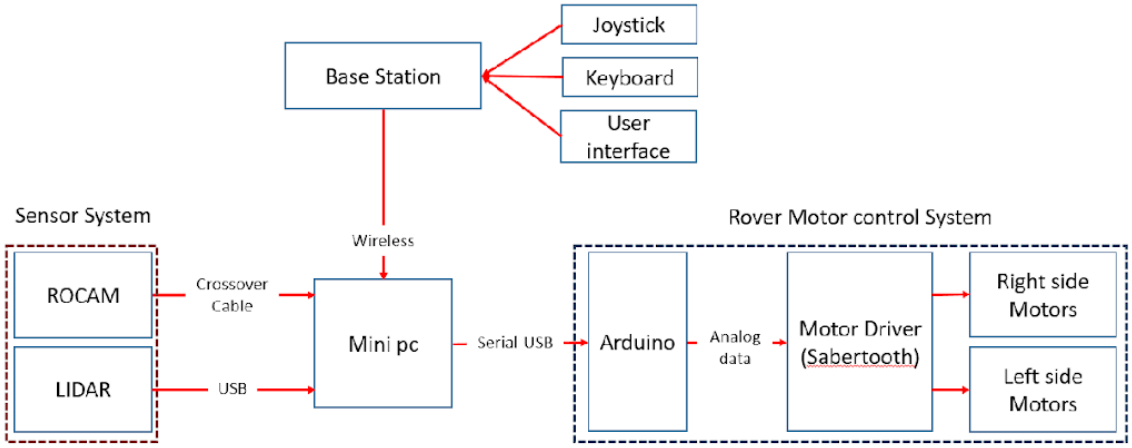

Figure 5: Data line system diagram

III. Experimental validation

This part included the testing of the SLAM package Hector and the tracking of the AR tag. As it can be seen in the picture below, SLAM was performed at the basement of Randolph building. Hector SLAM could map the environment sufficiently well but without odometry measurements, errors were occurring in depicting the actual orientation of the robot.

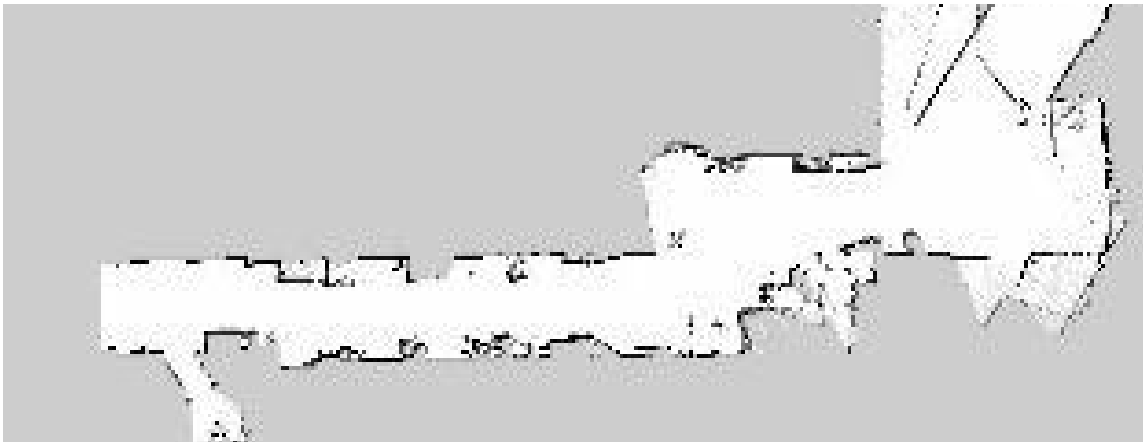

Figure 6:Mapping of Randolph basement

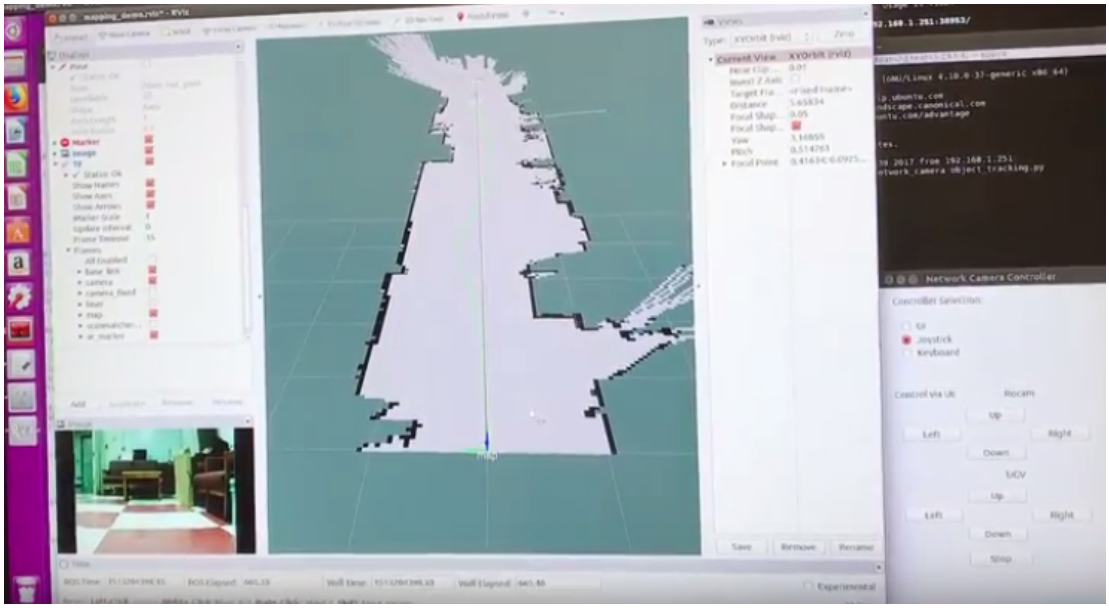

Figure 7: Mapping in Third person view (Rviz)

Also AR tagging and tracking was possible while the UGV was moving.

Figure 8: Tracking moving target, UGV fixed

Figure 9:Tracking fixed target, UGV moving

IV. Conclusions and Future Work

The conclusions we have to make concerning the overall path to the goal of creating a UGV for mapping and sensing its environment are the following. The ROS packages provide an easy to use platform with many capabilities. The mapping is done accurately enough for our purpose, tracking also fast and the platform all together behaves and responds fast. The wifi connection was producing some obstacles in the fast and accurate implementation, since many times communication was lost between base station and mini pc due to wifi lost connection.

Future work outside the framework of this course could include odometry and other additional localizing systems for better estimation. Also, wifi connection stabilization would be an important aspect of improvement. Finally, the one system component that was not utilized could be utilized for many purposes and that is the robotic arm. One application would be to find the target, map it on the RVIZ map and then use the arm to grasp an object that has the AR code on it. In this way all system capabilities would be combined and localizing would be used in parallel with camera telemanipulation to create a very powerful result. That would be the traversing of the environment, the mapping, the target searching and telemanipulation of it. The application of this could have many different implementations.

V. Manual

The below manual includes every operational capability the system has.

Hardware Check list

- Check the connections at mini PC

→ Mini PC power cable

→ Crossover cable for ROCAM

→ USB cable for Arduino

→ USB cable for LIDAR

→ USB wifi router(3.0 port works good)

- Check the connections in ROCAM

→ ROCAM Power cable

→ Crossover Cable

- Check the connections in LIDAR

→ LIDAR Power cable

→ LIDAR Data cable(USB)

- Check the connections between motor-driver(Sabertooth) and Arduino

ii). Mini pc Setup (ID: team2 , password: 1)

- Turn on the Mini PC

iii). Base Station Setup (ID: team2 , password: password)

- Open the terminal

- $ssh team2 (ssh into miniPC)

- $roslaunch sys_launcher ugv.launch

- Open the terminal

- $rosrun rviz rviz

→ Display ROCAM image and LIDAR data

- Open the terminal

- $rosrun gui_team2 uibk_Ioannis.py

- Open the terminal (for joystick)

- $ls /dev/input/

- $sudo chmod 777 /dev/input/js0

- $rosparam set joy_node/dev “/dev/input/js0”

- $rosrun joy joy_node

- Open the terminal (for operating ROCAM using keyboard)

- $rosrun teleop_twist_keyboard rocam_keyboard.py

  - Open the terminal (for operating UGV using keyboard)

  - For mapping and camera tracking

    Mapping

    - rosrun sicktoolbox_wrapper sicklms _port:=/dev/ttyUSB0
    - roslaunch hector_slam_launch tutorial.launch
    - rosrun map_server map_saver -f mapnameonlywithoutfileextension

    Tracking

    - roslaunch network_camera network_camera.launch
    - rosrun object_tracking.py