**West Virginia University – Statler College of Engineering and Mineral Resources**

***Lane Department of Computer Science and Electrical Engineering***

# 2016 Mercury Robot Challenge

Technical Report

Robot CEJ

*Group Members*

Edwin Mann

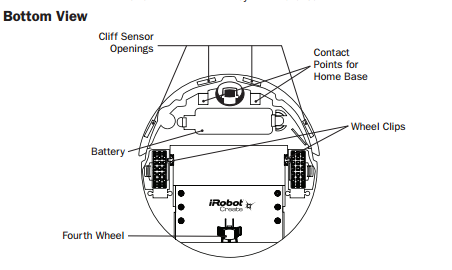
Olujide Jacobs

Caleb Azonsi

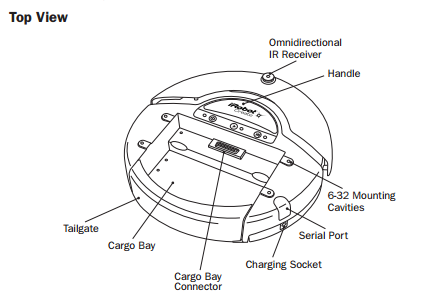
March 16, 2016

# Introduction

This section will give an overview of our robot and the design we came up with. In order to create a robot for this competition we used a Roomba as our base. The Roomba is can be very easily manipulated and has many built-in features that we found to be advantageous towards our goals for this competition. The Roomba has an Open Interface which can be sent set commands in order to control its movement and other features. It can also be connected to by the computer with a serial port which allows for easy manipulation. Along with choosing the Roomba as our robot’s base, we also decided on using a catapult as our method of propulsion for this competition. We thought that this would be the best way to launch an object the furthest distance. The following report will go into more detail on all of the modifications made to the Roomba in order to make it a robot of our own.



Figure



Figure

# Communication systems

## *TeamViewer*

Our goal is to be able to communicate with our robot and control it from at least fifty miles away. In order to accomplish this task we used a program called “TeamViewer”. In order for this connection to work, we needed to mount a tablet on the robot to communicate with. The tablet is then able to be connected to from our computer using the TeamViewer application. The tablet contains the software we created in order to control the robot. The “TeamViewer” application allows us to control the screen of the tablet and run our robot software from there. The tablet uses a serial connection to the robot in order to communicate.

TeamViewer uses a UDP pinholeing connection. How this works is, first, our computer has a TCP connection to the main TeamViewer server. When we want to make a connection between our computer and the tablet, the machine tells the main server its intention. The main server then gives the IP address of the tablet to our computer. Our computer then begins sending UDP packets to the tablet. The tablet is then signaled that our computer intended to connect and is given its IP. The client then also begins sending UDP packets to our computer.

## *Software Communication System*

We also have a communication system contained in our software. This connection is between the Operator Control Unit (OCU) and the Robot Control Unit (RCU). The Operator Control Unit controls all of the information on the computer and then sends that information to the Robot Control Unit for it to interpret and then give commands to the robot. For example, the Operator Control Unit handles any of the controller’s buttons being pressed. It then sends this information to the Robot Control Unit. The RCU controls all of the robot’s elements such as the arm, wheels, etc. For example, when a button on the controller is pressed, the RCU receives this information from the OCU and then will tell the robot to move depending on which button is pressed.

## *Camera Connection*

Another connection we used was between our computer and the camera mounted on our robot. We use an IP connection in order to see the visual feed from the camera. Using the AXIS IP Utility application, we are able to setup an IP address with the camera and use the online application to view the feed.

# Video Feedback System

In order to be able to control the robot from at least fifty miles away, we needed to create a video feedback system. In order to accomplish this we mounted a 2.7mm Fujinon Fish Eye camera to the robot. This camera comes with an application where we are able to see the feed on our computer. This application also allows us to zoom in and change our view in all directions. The Camera uses power from the robot itself in order to work while mounted on the robot.

As stated previously, the online application for the camera uses an IP connection. In order to use it, we setup an IP address using AXIS IP Utility. We then connect to this IP on the computer and we are able to use the application with very quick visual response time. The use of this camera and its application allows us to easily navigate without actually seeing the robot. The fish eye lens allows us to see a one hundred and eighty degree view of our surroundings. And its zoom feature allows us to click on any part of the video feed and instantly zoom in on the area in which we clicked.

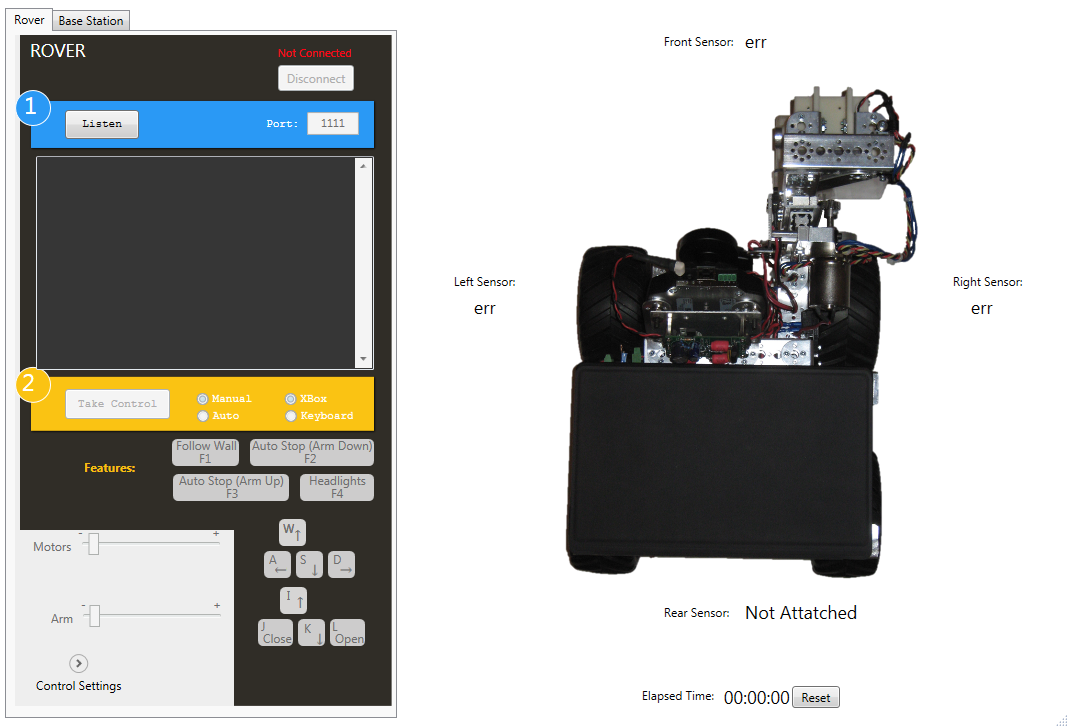


Figure

# Driver Interface

Our main goal when setting up the driver interface was to make it very user friendly. With our interface we can either use a keyboard or an Xbox Controller to control our robot. For the competition, we will be using the computer keyboard. The interface allows you to both drive the robot and control the arm. It also allows you to release the arm when you are ready to throw an object. It can tell you the values of the sensors and has a visual aid in order for you to know which sensor each value corresponds to on the robot.

As stated before, the device we will be using in order to control the robot will be a computer keyboard. The software will make the robot preform different actions depending on which key is pressed on the keyboard. The featured keys and their corresponding robot action can be seen in the picture below.



Figure

# Drive Configuration

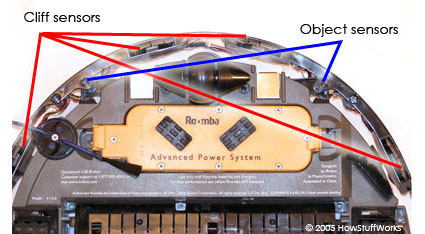
Our robot has a specific configuration that differs slightly from that used by the original Roomba. The Roomba originally contained five motors but our robot only has two. We removed the other three motors as they were unnecessary and we could use the voltage being sent to these motor for other objectives that pertained to the project. Our robot also contains three wheels. Two of these wheels are motorize and able to be controlled by our software. The third doesn’t have a motor and is used for stability and turning ability.

In order to make the robot move we sent signals to each of the motor in the wheels. Each wheel can be turned individually. This keeps our turning radius to a minimum. The speed of the wheels can be adjusted using the interface. In this case, both wheels are affected by the change in speed to prevent from one wheel turning faster than the other.

# Subsystems

## *Sensors*

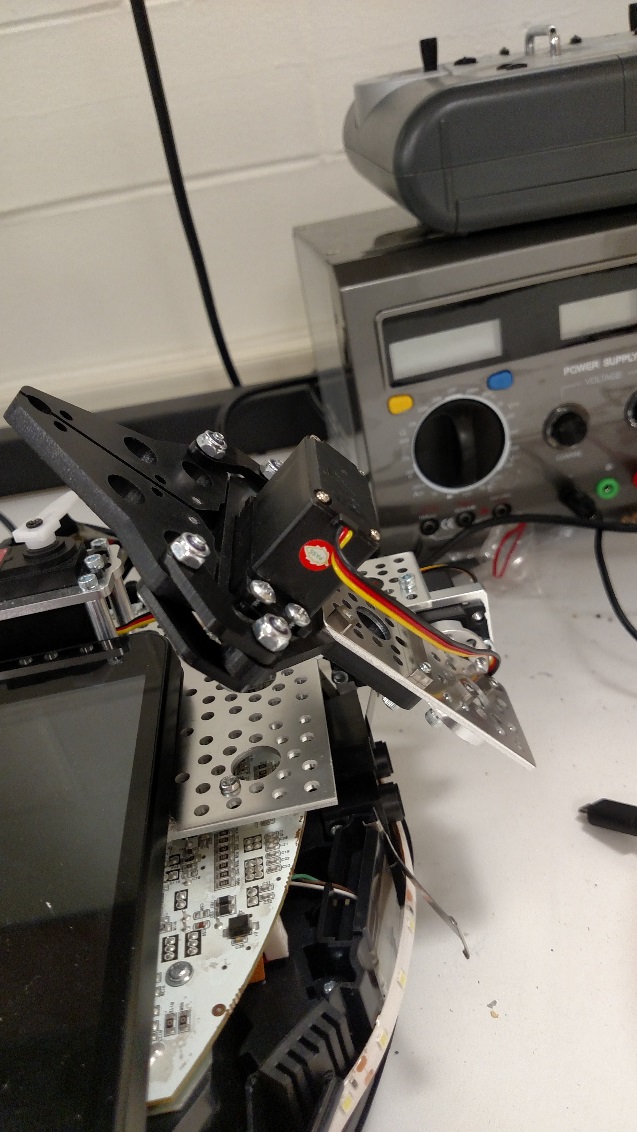
The Roomba comes equip with its own sensors. There are four “cliff sensors” and two “object sensors”. We are able to use these sensors to our advantage. Our software can read the values of these sensors in order to recognize its surroundings and automatically redirect its chosen path. It can also use these sensors to initiate the automatic breaking system within our software.



Figure

## *Arm*

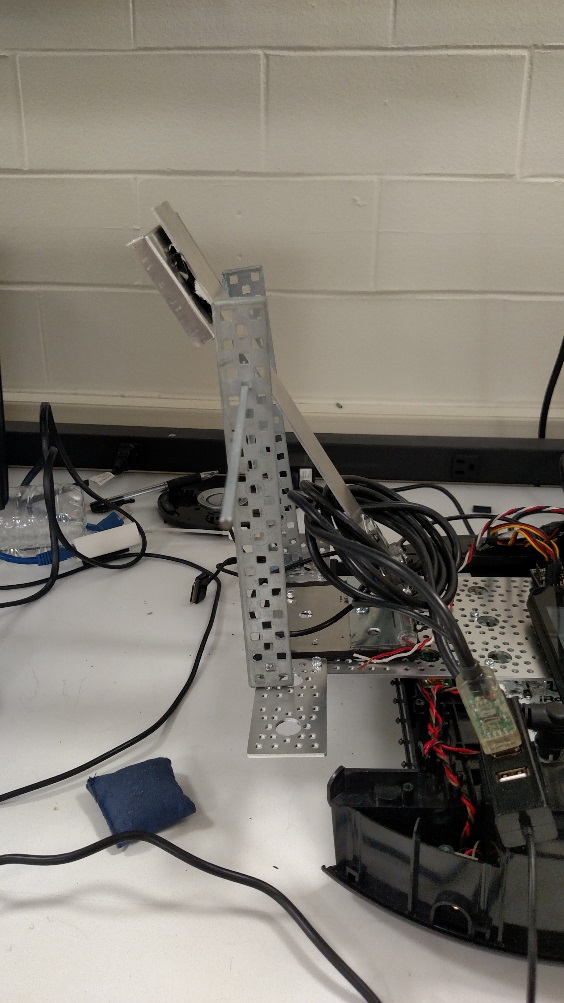
The arm uses many different parts in order to accomplish its task. It uses two Llitec HS-311 servo motors. These motors allow for the arm to move down and open in order to pick up objects. One of these motors is located below the catapult platform and allows for the “grabber” to be raised and lowered. The second of these motors connected to the “grabber” and allows for it to open and close. The “grabber” we used was designed specifically to work with our servos. The servos allow for the arm and the “grabber” to move smoothly and accomplish the task at hand.



Figure

## *Catapult*

In order for our object to propel objects through the air, we came up with a catapult design. The catapult uses a mouse trap spring in order to create the tension needed for propulsion. This spring is mounted to the robot and a metal lever with a container on the top. The correlation of the arm to the metal container is such that the arm is able to easily place and object into this container. Like the arm, the catapult also uses a servo motor. In this case, the motor is connected to a small plastic tab. This tab is used to hold the arm of the catapult in the “ready” positon. When the object is in the container and ready to be launched, the catapult servo moves the plastic tab which releases the arm. The servo for the catapult can be controlled by our software allowing the user to launch release the arm whenever they are ready.



Figure

## *LED’s*

In order for our robot to be controlled in the dark, we placed LED’s on the front. The system for this is quite simple. The LED’s are wired to the robot which is where they get their power from. The software programs the robot to only allow a certain amount of voltage to be sent to the LED’s. These can also be turned off and on using the keyboard controls.

# Power Subsystem

Our robot is powered by a rechargeable NiMH battery. The battery can produce around 18-volts. This voltage is used to power the wheels, camera, arm, LED’s, sensors and everything else used in our robot. Some of the features of our robot already had power running to them when due to the setup of the original Roomba. But we did have to make modifications in order to power other features of our robot. In order to accomplish this, we had to remove three different motors from the Roomba. This gave us the ability to use the voltage being sent to these motors. We used this voltage in order to power the LED’s, arm and camera. The LED’s use the voltage originally sent to the “spinning side brush” of the Roomba. Both the camera and arm use the voltage that was originally being sent to the vacuum portion of the Roomba. Both voltage sources are able to produce up to sixteen volts. Using commands sent to the Roomba, we were able to adjust the voltage output at both to our exact specifications at both of these sources.