WEST VIRGINIA UNIVERSITY

BENJAMIN M. STATLER COLLEGE OF ENGINEERING & MINERAL RESOURCES

LANE DEPARTMENT OF COMPUTER SCIENCE & ELECTRICAL ENGINEERING

**GROUP 6B: Mercury Robot**

**SYSTEM MANUAL**

***CALEB AZONSI***

***EDWIN MANN***

***OLUJIDE JACOBS***

FACULTY MENTOR : DR. POWSIRI KLINKHACHORN

INSTRUCTOR : DR. YENUMULA V. REDDY

Contents

[Abstract 3](#_Toc449559086)

[Introduction 4](#_Toc449559087)

[Executive Summary 4](#_Toc449559088)

[Needs 5](#_Toc449559089)

[Objectives 5](#_Toc449559090)

[Stakeholder Goals 5](#_Toc449559091)

[Conclusion 6](#_Toc449559092)

[Design Achievements 6](#_Toc449559093)

[Complete hardware design 8](#_Toc449559094)

[Schematics 8](#_Toc449559095)

[Hardware Used 9](#_Toc449559096)

[Roomba 9](#_Toc449559097)

[HP Stream 7 10](#_Toc449559098)

[Sensors 11](#_Toc449559099)

[Arm 12](#_Toc449559100)

[Camera 13](#_Toc449559101)

[Catapult 14](#_Toc449559102)

[LED’s 15](#_Toc449559103)

[Power supply 16](#_Toc449559104)

[Complete Hardware Assembly 18](#_Toc449559105)

[During Construction 18](#_Toc449559106)

[Completed 20](#_Toc449559107)

[Complete Software design 21](#_Toc449559108)

[Communication systems 21](#_Toc449559109)

[TeamViewer 21](#_Toc449559110)

[Software Communication System 22](#_Toc449559111)

[Camera Connection 22](#_Toc449559112)

[Video Feedback System 23](#_Toc449559113)

[Driver Interface 23](#_Toc449559114)

[Drive Configuration 24](#_Toc449559115)

[Software Specification 25](#_Toc449559116)

[Complete source code listing 32](#_Toc449559117)

[Appendix 1 – A user Manual 32](#_Toc449559118)

[Starting up Robot 32](#_Toc449559119)

[Shutting down Robot 32](#_Toc449559120)

[Appendix 2 - A Maintenance Manual 33](#_Toc449559121)

[Scheduled Maintenance 33](#_Toc449559122)

[Appendix 3 – Original Design Proposal 33](#_Toc449559123)

[Elevator Pitch 33](#_Toc449559124)

[Summary 34](#_Toc449559125)

[Introduction 34](#_Toc449559126)

[Requirement Specification 35](#_Toc449559127)

[System Design 40](#_Toc449559128)

[**1. Design and Analysis** 40](#_Toc449559129)

[1.1 Overall System architecture 40](#_Toc449559130)

[1.2 Use Case 43](#_Toc449559131)

[1.3 User Interface Specification 44](#_Toc449559132)

[1.4 Database Design 45](#_Toc449559133)

[1.5 Detailed Circuit Diagrams 46](#_Toc449559134)

[1.6 Software Specification to the function level 49](#_Toc449559135)

[2. Testing 59](#_Toc449559136)

[3 Project Plan 64](#_Toc449559137)

[Appendix 4- Summary of Revision 71](#_Toc449559138)

# Abstract

The Mercury Remote Robot Challenge is a worldwide, interscholastic competition for robots. The goal is to plan and assemble a robot equipped for finishing a remote mission. The operator of the robot must be situated no less than 50 miles (80 kilometers) from the robot. The robot must be worked utilizing just the correspondences channel characterized by the Mercury rules. The mission incorporates route of a dim passage, picking a bean bag, climbing and climbing a seesaw, also, launching the beanbag to a mark. Notwithstanding these activities, the robot should likewise illustrate speed sprinting toward the completion line.

The task at hand is to design a robot that is equipped with components capable of motion, arm movement, mass transportation, and sensory. This prototype aims to enable control from distance of at least 50 miles. The prototype should be able to navigate and perform its functions in low light environment while avoiding obstacles like walls, bumps, and barriers. The arm functions of the prototype aims to be able to tilt, pinch, lift, propel, and carry

A bigger picture application of this prototype is very useful for long distance exploration in a case when miners are trapped under a cave and need some sort of communication and physical aid. This robot will be able to assist from a long distance, hereby not endangering the life of the rescuers and also delivering mobile help to victims.

# Introduction

## Executive Summary

We are currently designing a robot that is equipped with a camera which can help the driver see the path of the robot when he/she isn’t physically present; a flashlight which helps to navigate in dark or low-light environments; four-wheels to ensure balance of the robot so it doesn’t tip over when speeding; sensors to help the robot not hit walls and barriers; batteries, chips and wires that help with the functionality of moving the robot and powering it on.

        This robot is very useful for long distance explorations like rescue missions and extra-terrestrial explorations. For instance, miners are trapped under a cave and need some sort of communication and physical aid, this robot will be able to assist from a long distance hereby not endangering the life of the rescuers and also offering an efficient mobile help to the victims.

## Needs

The mercury robot seeks to complete the track while scaling with speed and accuracy. Also to ensure easy operation and user friend interaction with the operator. The robot also seeks to pick up and launch the load with accuracy and power enough to gain maximum points without hitting walls while doing that.

## Objectives

* Choose and Manipulate Vehicle Base
* Hack Roomba
* Build Arm Mechanism
* Program Arm to raise, and lower via control
* Build and program Gripper
* Create shaft plate for shock absorbing
* Build catapult mechanism
* Program catapult to set and release via control
* Hack Roomba Sensors
* Create Circuits to power servos and headlights
* Select Cameras
* Provide video feed from robot to base station
* Program loss of signal pointer to robot
* Finalize Robot trimming and testing

## Stakeholder Goals

Below shows the list of goals set for the robot:

1. To navigate through track in Oklahoma
2. Receive video feed
3. Pick bean bag and launch
4. Sprint at maximum capacity
5. To regain loss of signal

## Conclusion

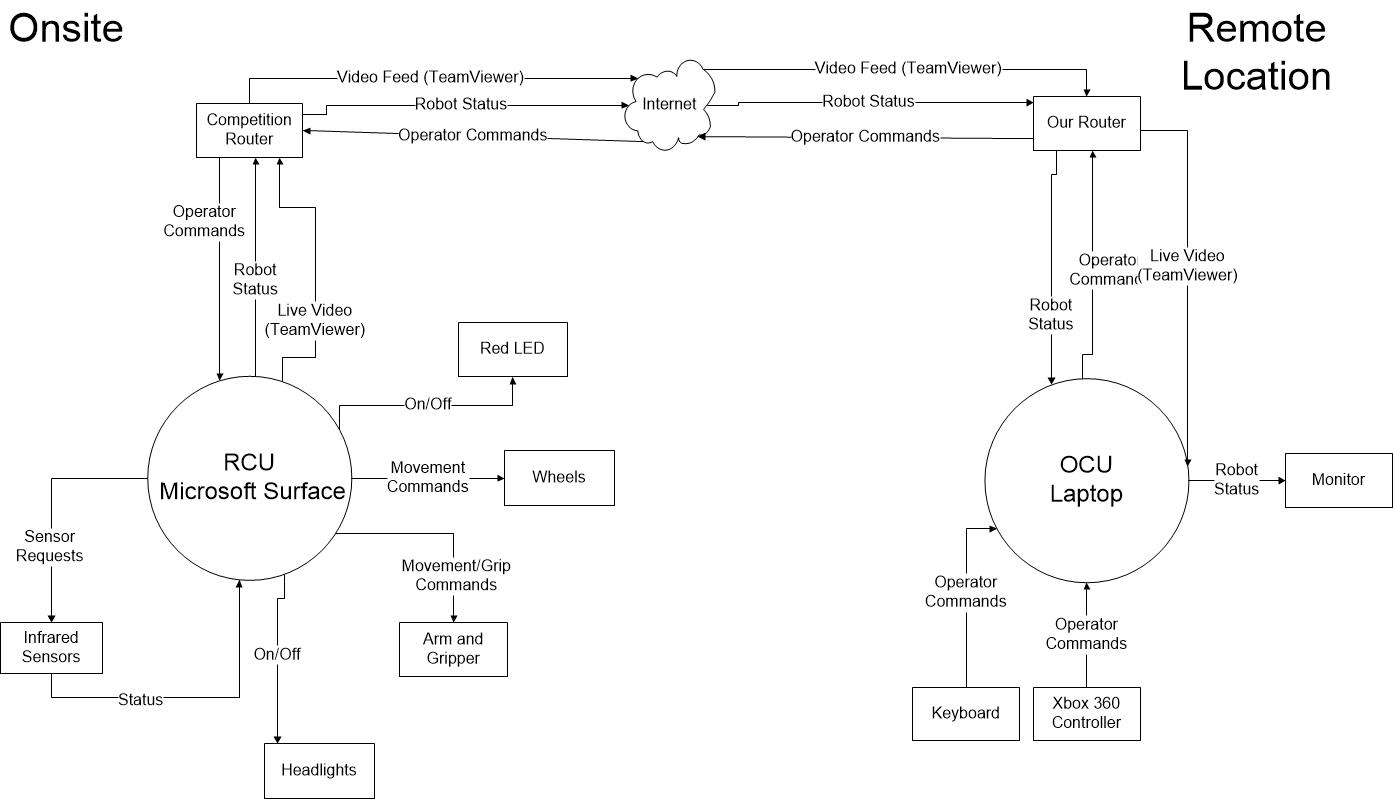
* Construction objectives were met
* We passed the loss of signal test
* Navigated the track, but had signal failure during pick up
* Team Mountaineers took **3rd Place Overall!**

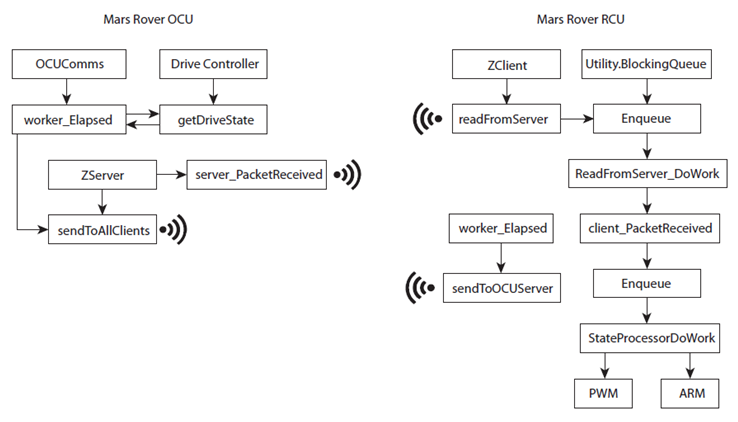
# Design Achievements

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Tasks | Description | Current Achievements |
| 1 | Reassemble Roomba | Assemble Roomba and arm | We currently have a Roomba assembled to our necessities. We added an arm, catapult, lights and a camera. We also made a few modifications to the Roomba in order to accommodate for our necessary additions. |
| 2 | Develop software to communicate | Write/adapt software core to power the Surface and control units. | We currently have software that controls nearly everything we added on the Roomba. The arm and catapult system can be controlled and the Roomba can be driven around using the software. The only additions not controlled by the software are the camera (which was unnecessary) and the lights which are controlled by a switch and a battery pack. |
| 3 | Test communications between OCU and RCU | Ensure data connections | Our Operator Control Unit and Robot Control Unit are able to connect to each other using IP ports |
| 4 | Interface windows tablet to robot and motors |  | We are currently able to run our Robot Control Unit on our tablet which allows the tablet to control the Roomba, Arm and catapult. |
| 5 | Compete in competition | Meet all deadlines for the competition | We were able to finish our robot and software by the competition deadline which allowed us to compete at Oklahoma State University |
| 6 | Place in competition | Finish the course and place in the top 10 | We were unable to accomplish this goal. We ended up with a system failure half way through the course and were unable to finish. We were disqualified for not finishing the course and therefore did not place in the top 10 or at all. |

# Complete hardware design

## Schematics



****Below is the block diagram connecting all hardware used



## Hardware Used

This section will give an overview of the hardware used in the robot and the specifications. 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.

### Roomba

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.

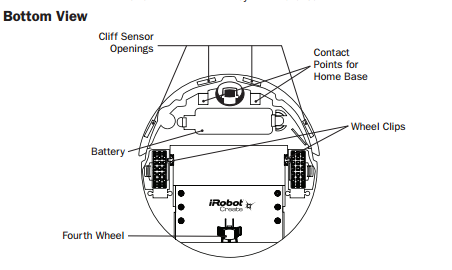


Figure 1

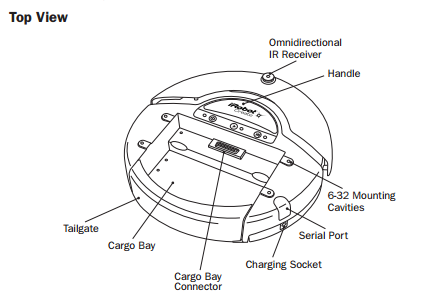


Figure 2

### HP Stream 7

Our main controller unit on the robot is a 7inch tablet. This tablet serves as a receiver unit for all commands sent by the operator. It also send back video feed with is viewed using team viewer and the help of a wide angle camera. The tablet will bear the RCU of the software which will control all functions of the robot , arm and camera.



### 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.

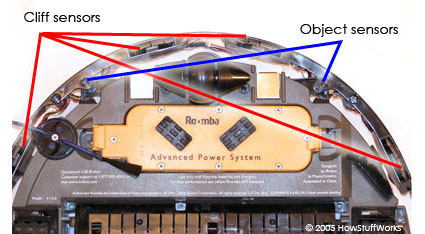


Figure 5

### Arm

The arm uses many different parts in order to accomplish its task. It uses two Llitec HS-311 servo motors and Actobotics gripper kit2 . 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.

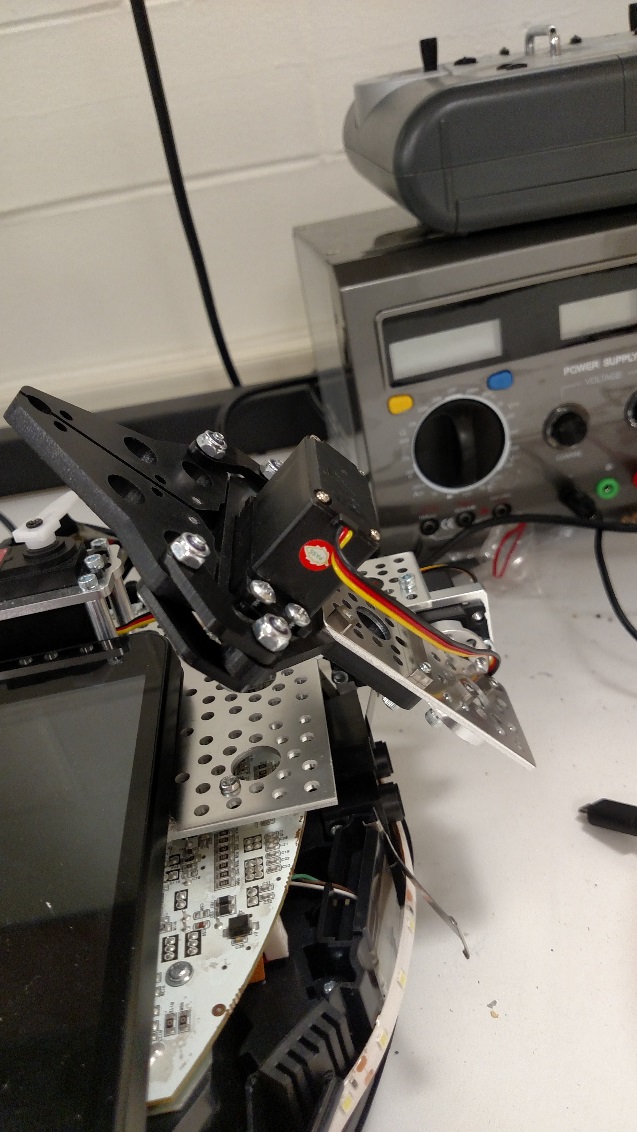


Figure 6

### Camera

For quality video feedback and adequate coverage for driving the robot, we employed the use of a wide angle camera. The camera was connected to the tablet which was then used through a free for all surveillance app to monitor the track while driving. Two cameras were used, one for driving and the other on the rear end to aim for launching the bean bag to the designated mark.



### 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.

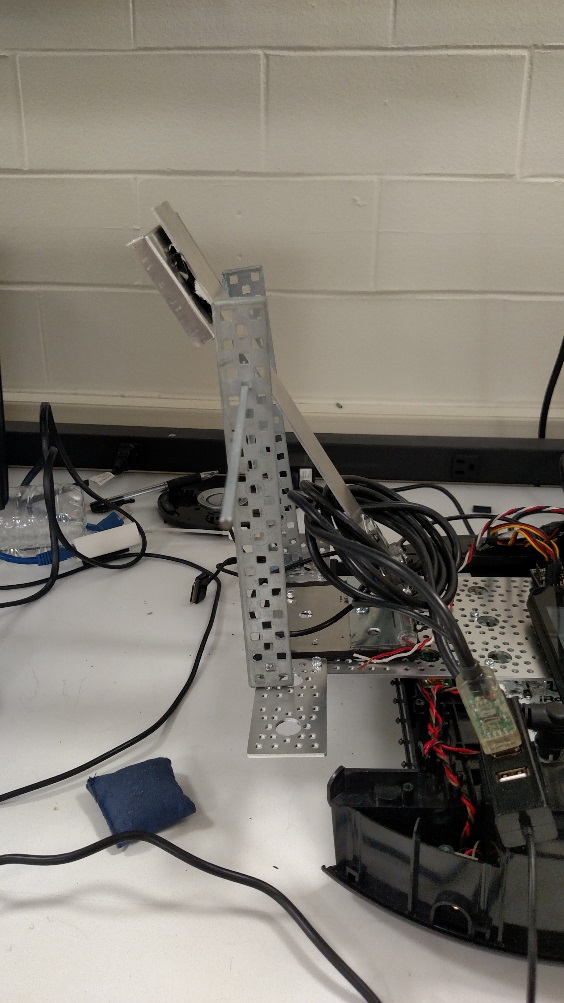


Figure 7

### 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.



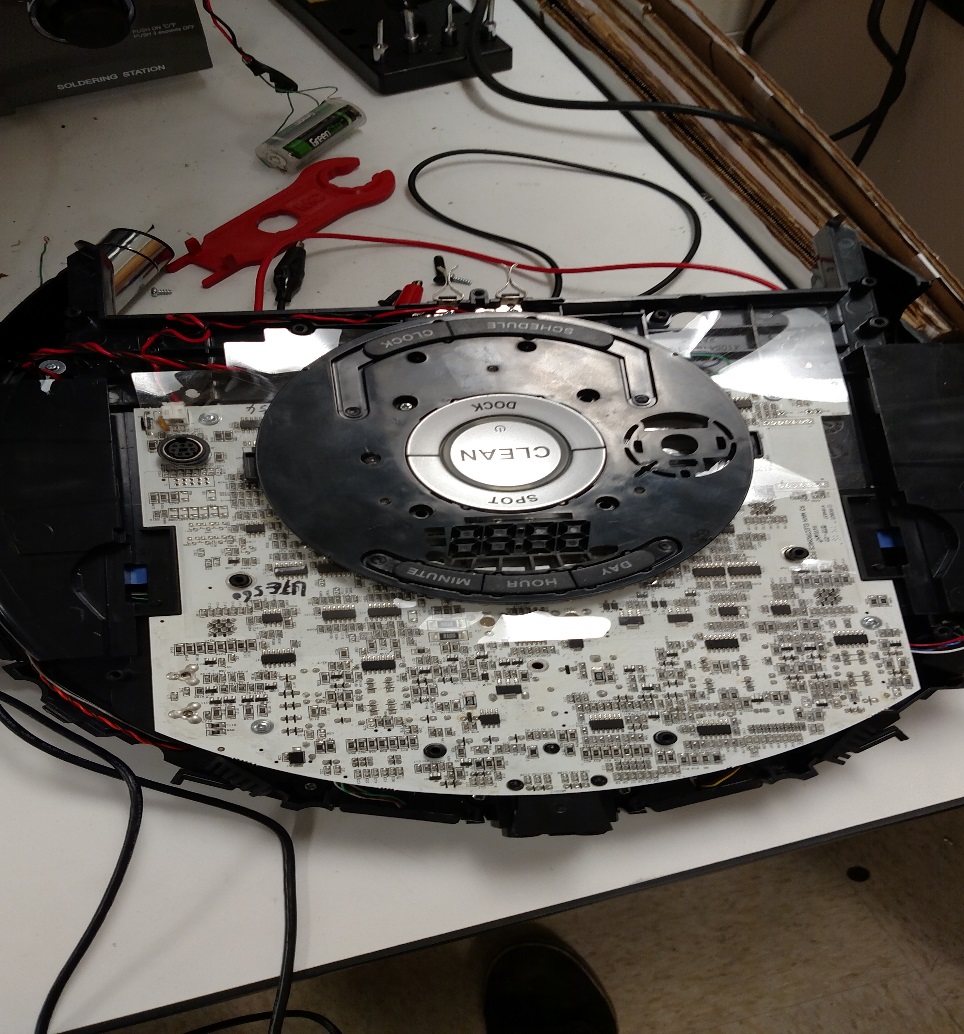
### Power supply

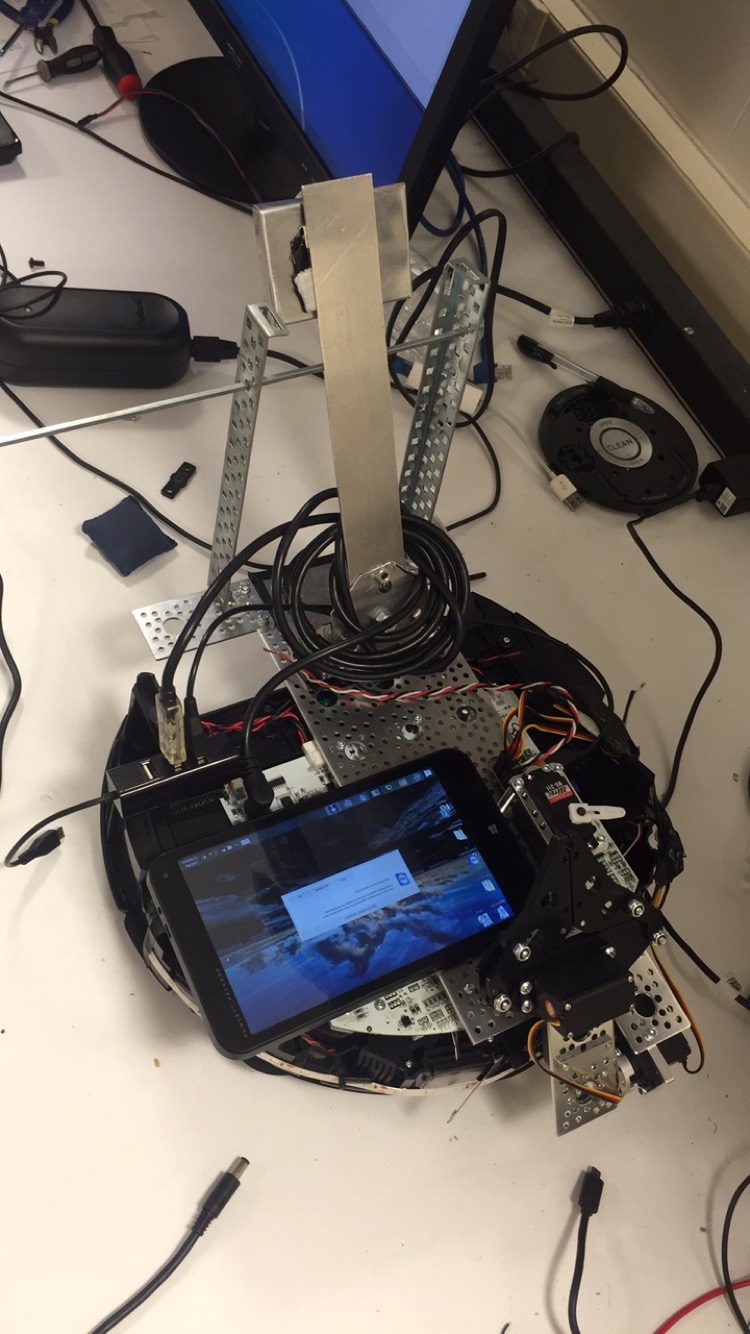
Our robot is powered by a rechargeable NiMH battery. The battery can produce around 18-volts. This voltage is used to power the wheels, camera, arm, LED’s, sensors and everything else used in our robot. Some of the features of our robot already had power running to them when due to the setup of the original 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.



## Complete Hardware Assembly

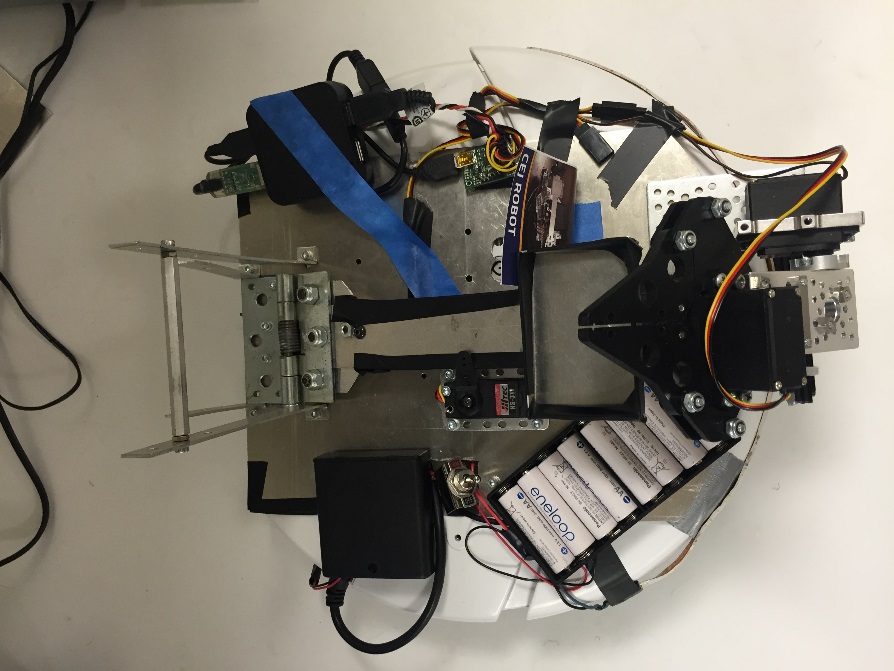
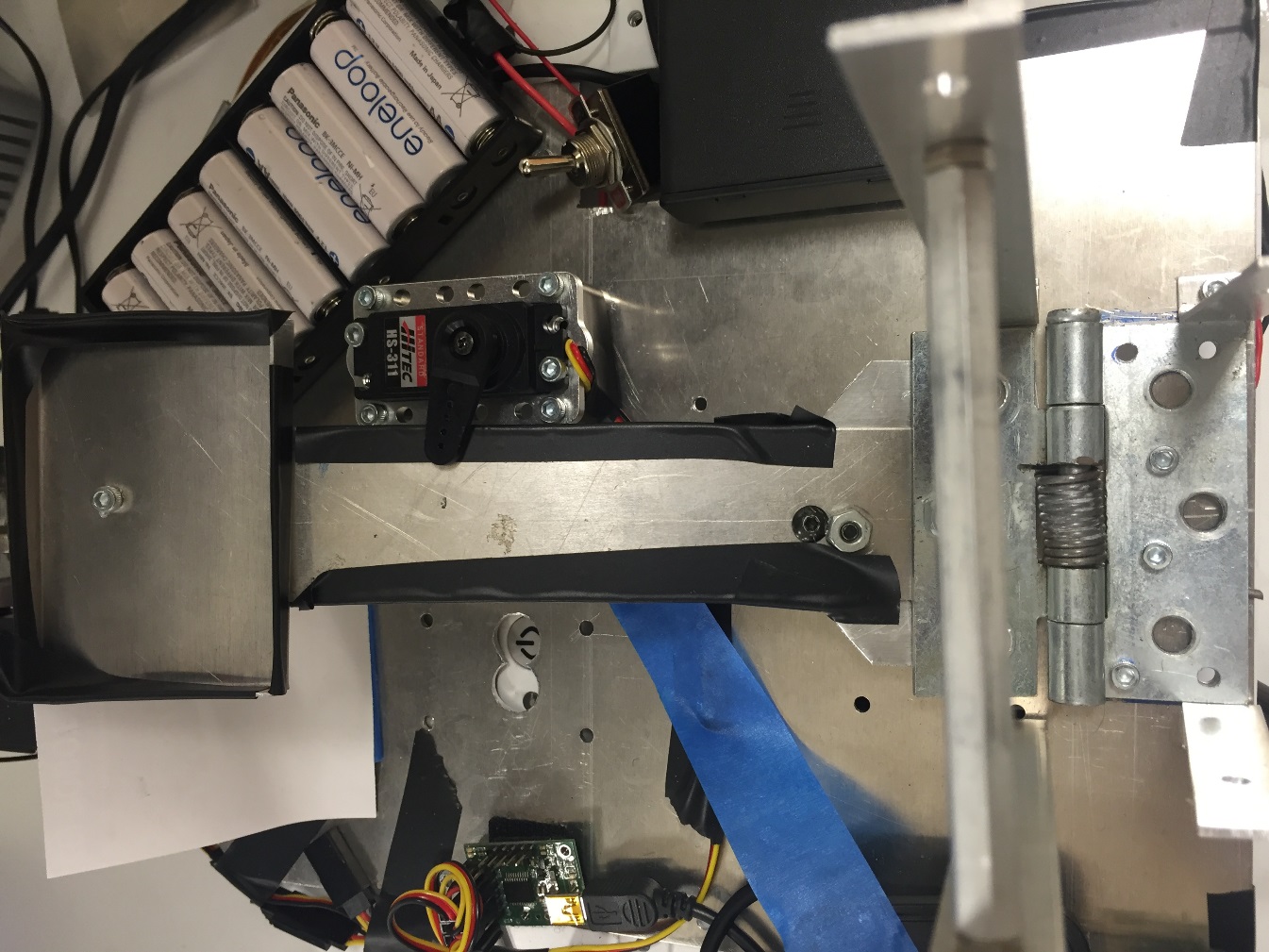
### During Construction

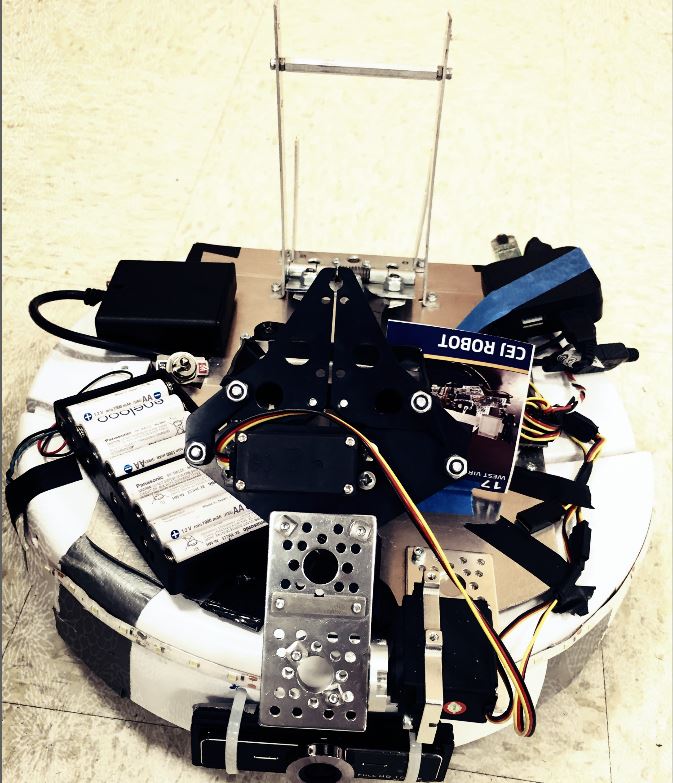
 





### Completed



# Complete Software design

## 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.

## 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.

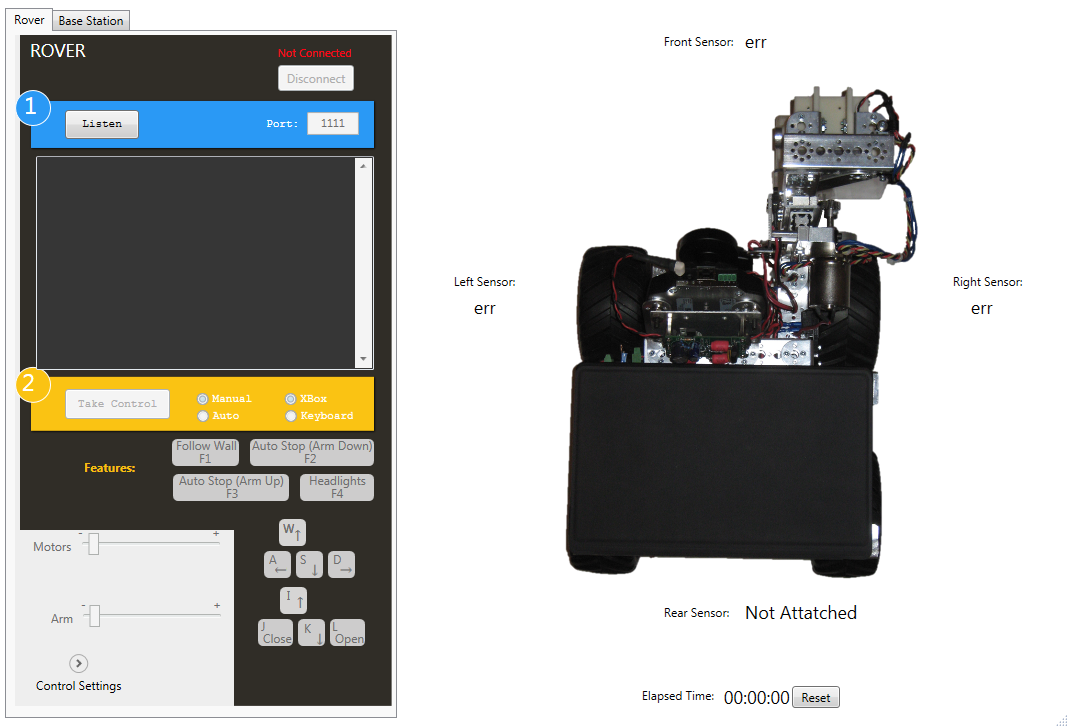


Figure 4

## 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.

## Software Specification

**Name of function:** Move Forward

**Description:** Moves the robot forward

**Inputs:** “W” Key

**Source of input:** Keyboard

**Outputs:** Both wheels move the robot forward

**Destination of output:** Wheel motors

**Processing:** When the driver presses and holds down “W” on the keyboard, the robot moves forward until the key is released.

**Requirements:** Connectivity between the controller and robot car

**Precondition:** “W” key is pressed

**Post condition:** Wheels move the robot forward

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Move Backward

**Description:** Moves the robot backward

**Inputs:** “S” Key

**Source of input:** Keyboard

**Outputs:** Both wheels move the robot backward

**Destination of output:** Wheel motors

**Processing:** When the driver presses and holds down “S” on the keyboard, the robot moves backward until the key is released.

**Requirements:** Connectivity between the controller and robot car

**Precondition:** “S” key is pressed

**Post condition:** Wheels move the robot backward

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Turn Left

**Description:** Robot turns to the left

**Inputs:** “A” Key

**Source of input:** Keyboard

**Outputs:** Both wheels move causing the robot to turn left

**Destination of output:** Wheel motors

**Processing:** When the driver presses and holds down “A” on the keyboard, the robot turns left until the key is released.

**Requirements:** Connectivity between the controller and robot car

**Precondition:** “A” key is pressed

**Post condition:** Wheels turn the robot left

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Turn Right

**Description:** Robot turns to the right

**Inputs:** “D” Key

**Source of input:** Keyboard

**Outputs:** Both wheels move causing the robot to turn right

**Destination of output:** Wheel motors

**Processing:** When the driver presses and holds down “D” on the keyboard, the robot turns right until the key is released.

**Requirements:** Connectivity between the controller and robot car

**Precondition:** “D” key is pressed

**Post condition:** Wheels turn the robot right

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Arm up

**Description:** Allows the arm of the robot to move upward

**Inputs:** “F3” key

**Source of input:** Keyboard

**Outputs:** Arm moves up

**Destination of output:** Motors in the arm

**Processing:** When the driver presses the “F3” key, the arm of the robot moves upward to a resting position above the catapult

**Requirements:** Connectivity between the controller and robot car

**Precondition**: “F3” key is pressed

**Post condition**: Arm moves up

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Arm down

**Description**: Moves the arm down to a resting position

**Inputs:** “F2” key

**Source of input:** Keyboard

**Outputs:** Arm moves downward

**Destination of output:** motors in the arm

**Processing:** When the driver presses the “F2” key on the keyboard, the arm will move down to a set position just above the ground.

**Requirements:** Connectivity between the controller and robot car.

**Precondition:** “F2” key is pressed

**Post condition:** Arm moves downward

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Open/Close Gripper

**Description:** Allows the gripper to open and close in order for it to pick up objects

**Inputs:** “F4” key, current state of gripper

**Source of input:** Keyboard

**Outputs:** Gripper Opens or closes

**Destination of output:** motors in the gripper

**Processing:** When the driver presses the “F4” key on the keyboard, the gripper either opens or closes depending on whether it is currently opened or closed

**Requirements:** Connectivity between the controller and robot car.

**Precondition:** “F4” Key pressed

**Post condition:** Gripper open/close

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Launch/Set

**Description:** Allows catapult arm to either be launched or set

**Inputs:** “F1” key, current state of catapult arm

**Source of input:** keyboard

**Outputs:** Catapult arm is either launched or set

**Destination of output:** Servo holding the arm down

**Processing:** When the driver presses the “F1” key on the keyboard, the arm is either set or launched depending on the current state of the arm

**Requirements:** Good connection in order to view a clear feed

**Precondition:** “F1” key pressed

**Post condition:** Catapult arm launched or set

**Side effects:** Excessive speed is detrimental to the control of the robot

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Desired/Required

**Name of function:** View video feed

**Description:** Displays the video feed from the IP camera for the driver

**Input:** Camera

**Source of input:** Team members

**Outputs:** Video feed to the driver

**Destination of output:** web application

**Processing:** The robot will send video feed data through wireless communication to the driver through a web application

**Requirements:** Connectivity between the controller and robot car

**Precondition**: Driver logged in to web application that displays video on screen

**Post condition:** User views feed

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Desired/Required

**Name of function:** Reconnect

**Description:** Resets the connection between the controller and the robot car

**Input:** Controller

**Source of input:** Driver

**Outputs:** connection reset

**Destination of output:** connection

**Processing:** When the driver pushes the button on the controller that correlates to the this function

**Requirements:** Connectivity between the controller and the car

**Precondition**: Specific button is pushed

**Post condition:** connection reset

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

**Name of function:** Loss of Signal

**Description:** Connection between the Operator Control Unit and Robot Control Unit have been lost

**Input:** Loss of connection

**Source of input: N/A**

**Outputs:** LED on Roomba turns red, any actions being carried out by the Roomba are stopped

**Destination of output:** Roomba

**Processing:** When connection between the OCU and RCU is lost, the red light on the Roomba will turn on and any actions being carried out by the Roomba will be stopped

**Requirements:** Connectivity between the OCU and RCU

**Precondition**: Connection lost

**Post condition:** LED turns red, actions performed by Roomba are stopped

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

# Complete source code listing

Source code is extremely large. Please refer to wiki page for code.

# Appendix 1 – A user Manual

## Starting up Robot

* Before turning on robot, make sure all wires from the robot, Camera, Servos are all connected properly to the USB Hub which is connected to the tablet
* Turn on robot with power button
* Turn on headlights and servos with switches
* Open the OCU application on the PC driving the robot
* Click on "Listen" on the OCU
* Open the RCU on the tablet. The OCU should have listened to the RCU
* Click on "Take Control" on the OCU System Base
* Drive robot with the keyboard keys designated on the System Base

## Shutting down Robot

* Type in "exit" on the RCU. the OCU automatically stops listening and RCU closes.
* Close the OCU from the PC
* Turn off headlights and servo with switches
* Turn off robot

# Appendix 2 - A Maintenance Manual

## Scheduled Maintenance

* Scheduled maintenance should be done once every month
* Replace any worn out tapes used to hold down robot components
* Unscrew basic parts like rear camera, catapult in order to access base for easy cleaning
* Remove front wheels by pulling it out and clean it. to replace, push it back in
* Unscrew bolts under the robot to access circuit board. This is not necessary unless robot needs to be fixed if broken.
* Disconnect al wired from HUB when not in use
* DO NOT drive robot off any surface more than 2 feet from ground. It will most likely DAMAGE robot

# Appendix 3 – Original Design Proposal

# Elevator Pitch

We are currently designing a robot that is equipped with a camera which can help the driver see the path of the robot when he/she isn’t physically present; a flashlight which helps to navigate in dark or low-light environments; four-wheels to ensure balance of the robot so it doesn’t tip over when speeding; sensors to help the robot not hit walls and barriers; batteries, chips and wires that help with the functionality of moving the robot and powering it on.

        This robot is very useful for long distance explorations like rescue missions and extra-terrestrial explorations. For instance, miners are trapped under a cave and need some sort of communication and physical aid, this robot will be able to assist from a long distance hereby not endangering the life of the rescuers and also offering an efficient mobile help to the victims.

# Summary

        The task at hand is to design a robot that is equipped with components capable of motion, arm movement, mass transportation, and sensory. This prototype aims to enable control from distance of at least 50 miles. The prototype should be able to navigate and perform its functions in low light environment while avoiding obstacles like walls, bumps, and barriers. The arm functions of the prototype aims to be able to tilt, pinch, lift, propel, and carry

        A bigger picture application of this prototype is very useful for long distance exploration in a case when miners are trapped under a cave and need some sort of communication and physical aid. This robot will be able to assist from a long distance, hereby not endangering the life of the rescuers and also delivering mobile help to victims.

        This prototype will be competing in the mercury robot challenge taking place in Oklahoma State University.

# Introduction

        The Mercury Remote Robot Test is a worldwide, interscholastic rivalry of mechanical autonomy. The test is to plan and manufacture a robot fit for finishing a remote mission. The administrator of the robot must be situated no less than 50 miles (80 kilometers) from the robot. The robot must be worked utilizing just the interchanges channel characterized by the Mercury tenets .The mission incorporates route of a dim passage, catching a bean pack, climbing and dropping 30 degree grades, and conveying the payload to an objective. Notwithstanding these activities, the robot should likewise exhibit rate sprinting toward the completion line.

The target of the test is to augment your groups score. Likewise, the robot must be in consistence with all standards and have the capacity to perceive Loss of Sign (LOS) keeping in mind the end goal to be pronounced the 2016 Mercury Remote Mechanical autonomy Test Champion. LOS is the condition that keeps the robot from accepting orders from the administrator because of a system issue (not identified with the robot framework). Robots which come up short the LOS test might, best case scenario place second. All robots will take the LOS test amid the morning of the day of the opposition.

# Requirement Specification

***Requirements***

At the season of the start of the match the robot should fit into a 18 in (46 cm) by 18 in square with a vertical cutoff of 12in (30.5 cm) in size volume however may grow to bigger once the run has started.

We urge every one of the members to consider the security regulation required by the robots to ensure different members, open and the venue. There are some standard parts that are utilized for the robot outline that are directed for wellbeing purposes. We maintain whatever authority is needed to preclude (or permit to take part) any group that does not meet the measures of security considered by the Jury. Consider the accompanying:

•  Batteries: You may utilize NiCad, NiMH, SLA batteries or other "safe" batteries. Li+ batteries may be utilized just if the group can show that appropriate charging and low profile off frameworks have been actualize

•Rocket engines, Medieval thrashes, Atomic gadgets (that incorporates both combination and splitting) and any parts that tend to combust, blast, or kick off the end of the world

Segments that may make harm persons or property ought to be stayed away from.

        The opposition gives a 802.11b/g/n Wi-Fi system on the venue. All interchanges between the driver and the robot must utilize this system. The driver must build up a two-route correspondence with the robot. At any rate, the robot must send a pulse sign back to the driver.

The accompanying are the points of interest of the remote system and regulations of its utilization amid the opposition:

1. The rivalry Wi-Fi system will have the ESSID "MERCURY" and no security insurance. This ESSID won't be telecast. If it's not too much trouble guarantee that your framework can unite with a Wi-Fi system without the ESSID show.

2. The Wi-Fi switch giving this system will have an open IP address that will be uncovered to the group upon the arrival of the opposition.

3. Each group is permitted to have at most two organized hosts utilizing the Wi-Fi system. For instance, an IP camera and a Wi-Fi gadget will consider two hosts. A Wi-Fi gadget with a non-IP camera appended just considers one host (for instance, a cell phone giving video sustain and control channel will just consider one host, however it must utilize the Wi-Fi system).

4. The group will need to give data about their arranged gadgets on the online enrollment structure. The group may change this data on the structure any number of times up until the week prior to the opposition. This data incorporates a brief portrayal of every gadget, the Macintosh locations, and the ports every gadget will utilize if an inbound association is required.

5. The organized gadgets will need to utilize DHCP to acquire an IP address. Static IP locations are not permitted and will bring about the group's exclusion if utilized. IP locations are doled out in view of the Macintosh locations of the organized gadgets gave by the group on the enrollment structure.

6. If the group requires an inbound association with an arranged gadget, the group is permitted to have at most three sent ports. The data gave on the enlistment structure will be utilized and the group will be informed of the outside ports allocated to the group a week prior to the opposition.

7. During the group's run, just the group's robot will have entry to the Wi-Fi system. Every other robot will be totally killed.

8. A base station to give non-Wi-Fi remote connection between the robot and the official switch is permitted to be utilized nearby. This remote connection must not utilize the 802.11 standard. The base station must utilize the opposition Wi-Fi system to pick up Web access and the base station will tally towards the two greatest arranged gadgets.

9. Independent Wi-Fi repeaters, spans, specially appointed Wi-Fi systems, and access focuses are not permitted. The main 802.11b/g arrange every gadget may utilize is the official remote system.

The group should likewise pass the "Loss-of-Sign" test that will be performed in the morning of the opposition to be qualified as the opposition champ. The test will be executed as takes after:

1. The group plainly exhibits that the driver can control the robot,

2. The authority switch specialist will then close down the switch and the robot must have the capacity to plainly demonstrate that it is presently encountering lost sign circumstance and stop,

3. After the official switch is restarted, the group must have the capacity to show that the driver can re-setup association with the robot without the group faculty controlling the robot. The robot must demonstrate that association is re-built up by killing the Loss-of-Sign marker, and resume ordinary operation as in point 1.

***Competitive Benchmark***

Score = Tunnel+Pickup+Transit+70Sprint72 - Dist144+1-3\*WC-10\*RP

Where the following are possible scores for each action:

Tunnel={25,SuccessfullynavigatetheTunnelwithoutawallcontact 0,otherwise

Pickup={15,SuccessfullyAcquiresPayloadwithoutanyoutsideassistance 0,otherwise

*Sprint*

***Constraints***

In the accompanying cases the portrayed punishments will be connected.

Touching the robot – If the Robot Handler needs to right the robot amid the run it will bring about a score punishment of 10 focuses and the robot will be put where it cleared out the track or anyplace before that point. In the event that whatever other colleague touches the robot amid the run, the present run will be excluded and in this manner not scored.

Excessive Correspondence – If the judge decides that any colleague at the opposition site is giving headings to the Administrator amid the run, the group may be issued a notice, punishment or be precluded relying upon the degree of the infraction. The main interchanges suggested between the two people amid a run is "Begin when prepared" and "End this run?"

Touching track limits – If the robot comes into contact with the track dividers or traverses the range above taped track limits a punishment of 3 focuses will be deducted from the last score. The punishment will be evaluated every time the robot comes into contact with the limits. Amplified contact can be evaluated numerous punishments in the event that it keeps going longer than three seconds and the robot stays in movement, if a robot keeps on rolling over the dividers as to pick up favorable position the judge will survey a 5 divider punishment for every event . For instance, a robot that stops while touching the limit will just get one punishment however one that drives while touching the divider may get more than one at the judge

References

Oklahoma State University. *Mercury Robot Challenge* (2015) . Retrieved from https://mercury.okstate.edu/content/mercury-challenge

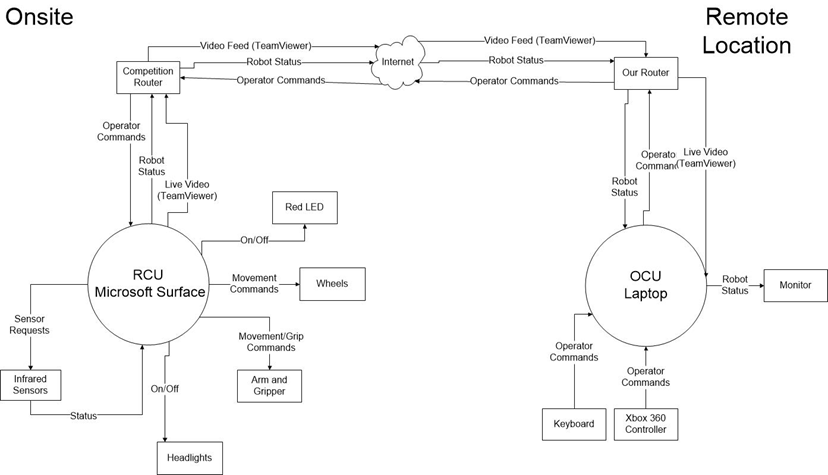
Society of Robots. (2015).*Robot Arm Tutorial*. Retrieved 21 October 2015, from http://www.societyofrobots.com/robot\_arm\_tutorial.shtml

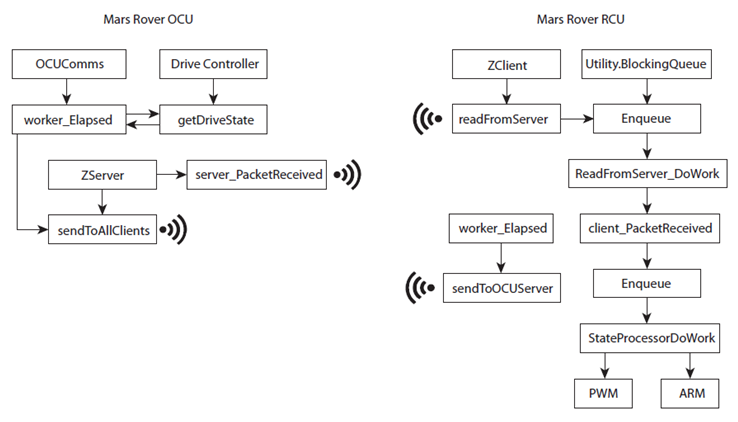
RobotShop (2015). *Robot & Arm Torque tutorial* Retrieved 21 October 2015, from http://www.robotshop.com/blog/en/robot-arm-torque-tutorial-7152

# System Design

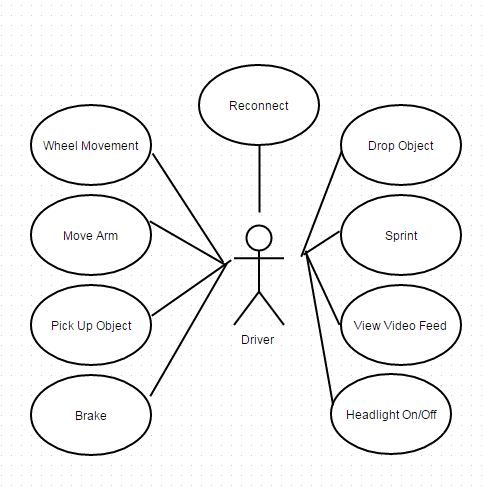
## **1. Design and Analysis**

### 1.1 Overall System architecture

****

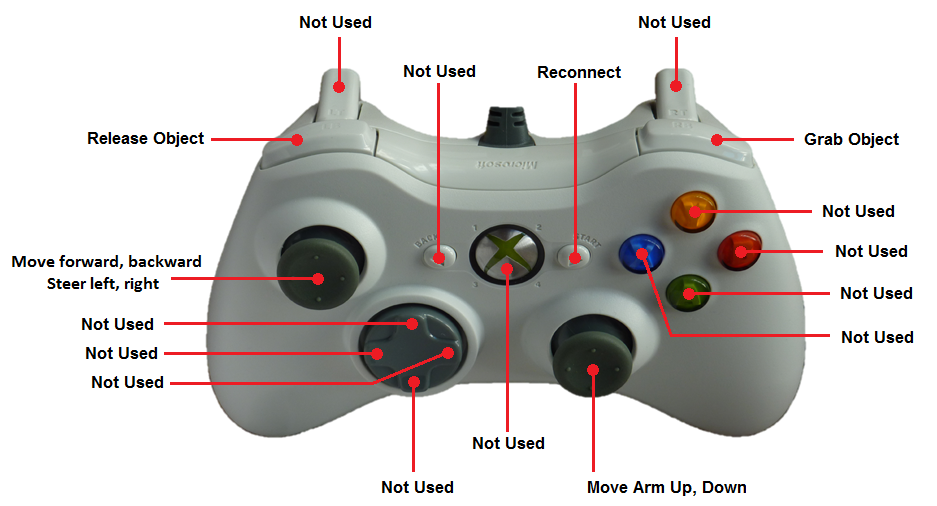
****

### 1.2 Use Case

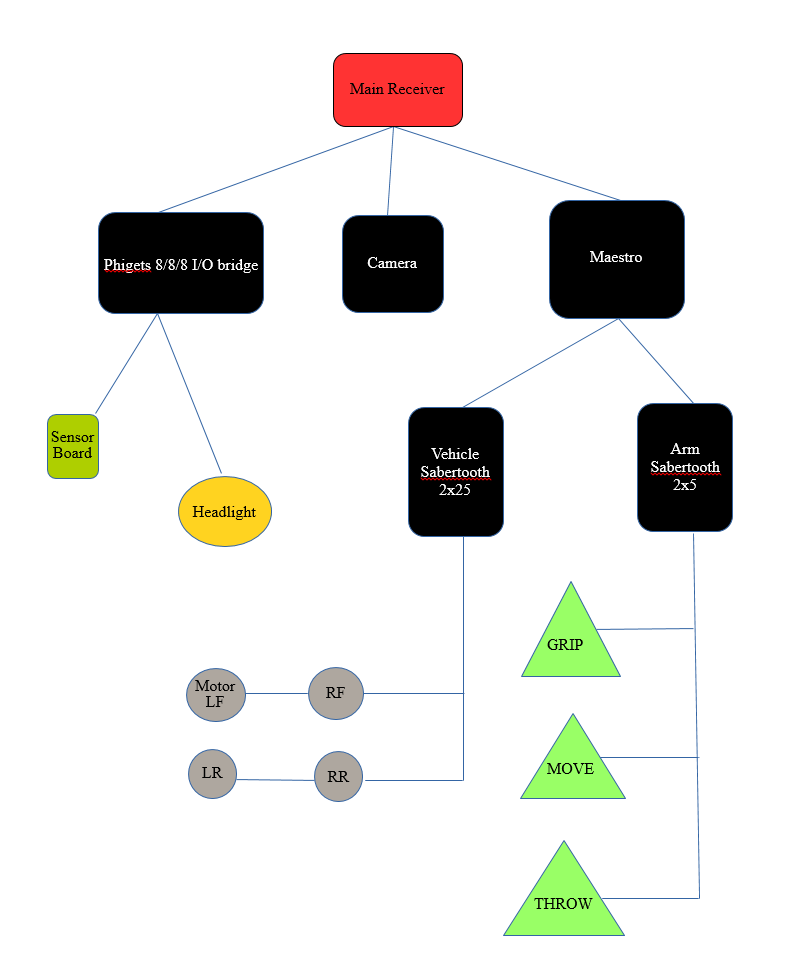
****

### 1.3 User Interface Specification

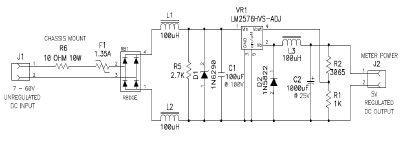
The individual driving the robot is going to have a controller that is already programmed to the robot, and also a screen in order to be able to determine the status of the robot. This is the only means of communication between the robot and the driver. Data will be sent from the robot to the driver through a possible Internet connection.

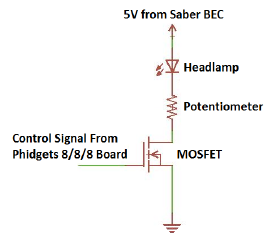


### 1.4 Database Design

****

### 1.5 Detailed Circuit Diagrams





The major individual components are pictured.These will all come together to create the end result; the final product! All Roomba chips and sensors are included in the icreate 2 pack.

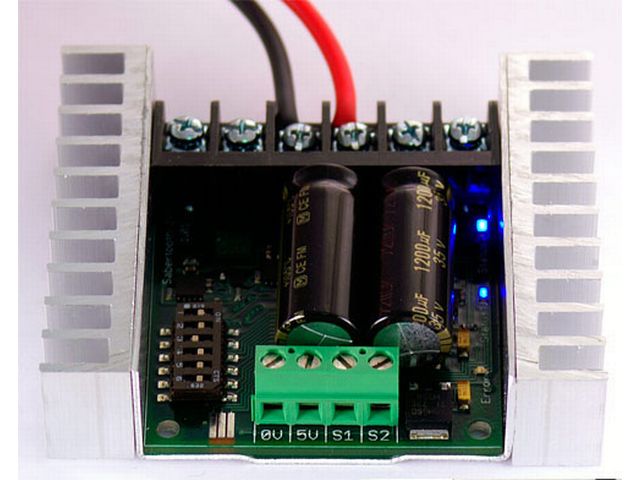
I robot Roomba







Resistors





Robot arm and gripper

### 1.6 Software Specification to the function level

#### 1.6.1 Wheel movement

**Name of function:** Move wheels

**Description:** Allows the robot to move in specific directions

**Inputs:** Controller

**Source of input:** The driver

**Outputs:** Wheels move the robot in various directions

**Destination of output:** Wheel motors

**Processing:** When the driver moves the controller’s left joystick, the robot moves to the direction in which the joystick is moved. Either left, right, forward or backward

**Requirements:** Connectivity between the controller and robot car

**Precondition:** Controller’s left joystick is moved in any direction

**Post condition:** Wheels move the robot around

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

#### 1.6.2 Move Arm

**Name of function:** Move arm

**Description:** Allows the arm of the robot to move

**Inputs:** Controller

**Source of input:** The driver

**Outputs:** Arm moves up or down

**Destination of output:** motors in the arm

**Processing:** When the driver moves the controller’s right joystick, the arm of the robot should move in the direction specified by the driver.

**Requirements:** Connectivity between the controller and robot car

**Precondition**: Controller’s right joystick is moved either up or down

**Post condition**: Arm moves up or down

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

#### 1.6.3 Pick up Object

**Name of function:** Pick up Object

**Description**: Allows the clamps at the robot arm open up, pick object, and close up.

**Inputs:** Controller

**Source of input:** The driver

**Outputs:** Clamp picks up the object

**Destination of output:** motors in the arm

**Processing:** When the driver presses the button on the controller, the clamps on the arm of the robot should either open, slide to pick object, or close.

**Requirements:** Connectivity between the controller and robot car.

**Precondition:** Specific button is pushed

**Post condition:** Object is picked up

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

#### 1.6.4 Drop Object

**Name of function:** Drop Object

**Description:** Allows the clamps at the robot arm open up, drop object, and close up.

**Inputs:** Controller

**Source of input:** The driver

**Outputs:** Clamp drops the object in the drop off zone

**Destination of output:** motors in the arm

**Processing:** When the driver presses the button on the controller, the clamps on the arm of the robot should either open, slide to drop object, or close.

**Requirements:** Connectivity between the controller and robot car.

**Precondition:** Specific button is pushed

**Post condition:** Object is dropped off

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

#### 1.6.5 Sprint

**Name of function:** Sprint

**Description:** Allows the robot reach maximum speed and cover some distance

**Inputs:** Controller

**Source of input:** The driver

**Outputs:** Robot sprints till it reaches the finish line

**Destination of output:** Wheel motors

**Processing:** When the driver moves the controller joystick up and presses the button to reach maximum speed (The Up direction button), the robot should continue sprinting till it's stopped by the driver.

**Requirements:** Good connection in order to view a clear feed

**Precondition:** Specific button is pushed

**Post condition:** Robot reaches maximum speed

**Side effects:** Excessive speed is detrimental to the control of the robot

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Desired/Required

#### 1.6.6 View video feed

**Name of function:** View video feed

**Description:** Displays the video feed from the IP camera for the driver

**Input:** Camera

**Source of input:** Team members

**Outputs:** Video feed to the driver

**Destination of output:** web application

**Processing:** The robot will send video feed data through wireless communication to the driver through a web application **Requirements:** Connectivity between the controller and robot car

**Precondition**: Driver logged in to web application that displays video on screen

**Post condition:** User views feed

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Desired/Required

#### 1.6.7 Brake

**Name of function:** brake

**Description:** Causes the robot car to slow down quickly when the user pushes a specific button on the controller

**Input:** Controller

**Source of input:** Driver

**Outputs:** robot car slows down

**Destination of output:** Brakes in the wheels

**Processing:** When the driver pushes the button on the controller that correlates to the braking function

**Requirements:** Good connection in order to view a clear feed

**Precondition**: Specific button is pushed

**Post condition:** robot car slows down

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Desired/Required

#### 1.6.8 Turn headlight light on/off

**Name of function:** headlight on/off

**Description:** Turns the headlight on and off

**Input:** Controller

**Source of input:** Driver

**Outputs:** headlight turns on or off

**Destination of output:** headlight

**Processing:** When the driver pushes the button on the controller that correlates to the this function

**Requirements:** Connectivity between the controller and the car

**Precondition**: Specific button is pushed

**Post condition:** headlight turns on or off

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

#### 1.6.9 Reconnect

**Name of function:** Reconnect

**Description:** Resets the connection between the controller and the robot car

**Input:** Controller

**Source of input:** Driver

**Outputs:** connection reset

**Destination of output:** connection

**Processing:** When the driver pushes the button on the controller that correlates to the this function

**Requirements:** Connectivity between the controller and the car

**Precondition**: Specific button is pushed

**Post condition:** connection reset

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

#### 1.6.10 Reconnect

**Name of function:**Throw object

**Description:**throws the bean bag

**Input:** Controller

**Source of input:** Driver

**Outputs:**arms throws the object

**Destination of output:**robot arm

**Processing:** When the driver pushes the button on the controller that correlates to the this function

**Requirements:** Connectivity between the controller and the car

**Precondition**: Specific button is pushed

**Post condition:**object thrown

**Side effects:** None

**Responses to abnormal behavior:** Messages displayed

**Necessity:** Required

## 2. Testing

#### Table 2.1 Failure Mode and Effects Analysis

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Description  Number | Item | | Failure | | Effects | | Description Method |
| Identification | Function | Mode | Causes | Device | User |
| 1.1 | Controller | Sends commands to the car | Doesn’t send the commands | Battery is dead | Car won’t receive the commands | None | Battery Notification |
| 1.2 | Controller | Sends commands to the car | Doesn’t send the commands | Lost connection | Car won’t receive the commands | None | Loss of Signal Notification |
| 2.1 | Arm | Picks up and carries the bean bag | Doesn’t react to the commands sent to it | Hardware is not connected | Arm is unable to move | None | Hardware Disconnected Notification |
| 3.1 | Car | Navigates the maze and receives commands from the controller | Doesn’t react to the commands sent to it | Battery is dead | Car will not perform the commands | None | Battery Notification |
| 3.2 | Car | Navigates the maze and receives commands from the controller | Doesn’t react to the commands sent to it | Lost connection | Car will not perform the commands | None | Loss of Signal Notification |
| 4.1 | Connection | Provides connection between the controller and the car | Loss of connection | Can’t connect to the network | Entire system will not be able to send or receive commands | None | Loss of Signal Notification |

**KEY:**

1. Description Number

1.1 Reference to an in depth description of the failure listed in “Description Failure Modes”.

2. Item

2.1 Identification

2.1.1 This is the component in our system that are at risk.

2.2 Function

2.2.1 The propose of the item in our system.

3. Failure

3.1 Mode

3.1.1 What is happens when the part fails.

3.2 Causes

3.2.1 What will cause this failure.

4. Failure Effects

4.1 Device

4.1.1 How this failure will affect our device.

4.2 User

4.2.1 How this failure will affect our user.

5. Failure Detection Methods

5.1 How we will detect that a failure and possibly how we will prevent it from happening

#### 2.2 Description of Failure Modes

**2.2.1  Controller**

**2.2.1.1  Battery is Dead**

The system will tell the user when the battery is about to run out. The user can then charge it or switch the battery with another one that is already charged.

**2.2.1.2  Lost Signal**

The system will tell the user when the system loses connection and can no longer communicate between the controller and the car.

**2.2.2  Arm**

**2.2.2.1  Hardware Has Become Disconnected**

When the arm becomes disconnected or isn’t connected correctly, it will no longer be able to receive commands and will be unable to move.

**2.2.3  Car**

**2.2.3.1  Battery is Dead**

The system will tell the user when the batter is about to run out. The user can then charge it or switch the batter with another one that is already charged.

**2.2.3.2  Lost Signal**

The system will tell the user when the system loses connection and can no longer communicate between the controller and the car.

**2.2.4  Connection**

**2.2.4.1  Loss of Connection**

The system will tell the user when the system loses connection and can no longer communicate between the controller and the car.

## 3 Project Plan

#### 3.1 Work Breakdown Structure

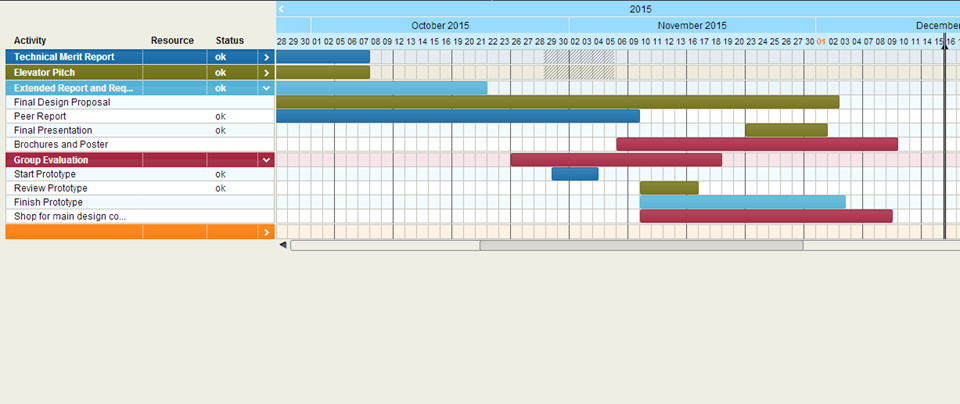
|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Description** | **Contributors** | **Resources** |
| Research | Gather information about the project and how we can accomplish our goals | Group | Books, Internet |
| Collecting Resources for prototype | Gather materials needed in order to build the prototype | Group | Materials supplied by Dr.Klink |
| Assemble Prototype | Build the prototype and assemble all of the hardware | Group | Materials supplied by Dr. Klink |
| Test Prototype | Run tests with the prototype in order to decide on changes for the final design | Group | Practice course |
| Collect materials for Final Design | Gather the materials for the final design | Group | Materials supplied by Dr. Klink |
| Assemble Final Design | Build the final design and assemble the hardware for it | Group | Materials supplied by  Dr. Klink |
| Write software | Create the software needed in order to complete the task at hand | Group | laptop, internet |
| Test Design | Make sure that the final design accomplishes all goals | Group | Practice Course |
| Compete | Competition at Oklahoma State University | Group | Competition course |

#### 3.2 Personnel Assignments

For this project thus far almost all of our work has been done with the entire group present and contributing. Most of the initial research was done by Caleb and Olujide. Most of the initial research dealt with learning about the competition. We needed to know what it was, our goals, the constraints and other valuable information that dealt with the Mercury Robot Challenge. We then used this information in order to plan out how we wanted to accomplish these tasks.

Every milestone we accomplished was a group effort. Every member was present when we worked on them. But the prototype was mainly assembled by Caleb. He collected the materials we needed and put it together. While he did that, Olujide and Edwin worked on the software needed for this project. Some of the software was given to us from the previous year’s groups but there was still work to be done and we had to look through the already written code in order to understand how the system worked.

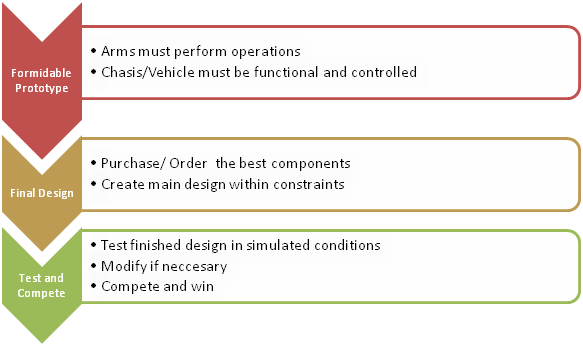
#### 3.3 Gantt Chart

****

#### 3.4 Milestones

|  |  |  |
| --- | --- | --- |
| **Task** | **Lead** | **Date** |
| Elevator Pitch | Group | 10/7/15 |
| Project Technical Merit | Group | 10/7/15 |
| Elevator Pitch Presentation | Group | 10/21/15 |
| Extended Problem Report and Requirements Specification | Group | 10/21/15 |
| Peer Report | Group | 11/4/15 |
| Evaluation of Group Members | Individual | 11/4/5 |
| Final Presentation | Group | 12/2/15 |
| Design Proposal Final | Group | 12/2/15 |
| Idea Expo | Group | 11/7/15 |

#### 3.5 Contingency Plans

****

#### 3.6 Task Assignments

In terms of building this project, there will be many schemes and shifts happening at once. Hardware will be assembled and developed as software will be written. We have divided the major tasks into three parts. Hardware, Motor and Controls, and finally External connections and wireless controls. The below table shows the various segments and the subdivision of tasks assigned to individual members of the group.

**Caleb Azonsi(Hardware)**

* Design and construct the robot frame
* Design and construct Robot Arm
* Choose Appropriate motors
* Assess weight and speed limitations to desired specifications.
* Calibrate ultrasonic sensors to avoid bumps( safety)
* Debugs electrical and mechanical activities.

**Olujide Jacobs(Robot Programming)**

* Programs preliminary robots functions using visual studio
* Write programs to control servos for arm
* Create program to combine all robot functions using visual studio
* Incorporate all other components of robot into the visual studio program(e.g) flashlight camera and sensors
* Write hazard prevention program
* Test and debug robot software frequently

**Edwin Mann(Controls and Serial Connections)**

* Create the controls functions with Xbox controller using visual studio.
* Create wireless connections for all hardware connections.
* Choose Wi-Fi device to be used for control.
* Choose appropriate Arduino controllers.
* Write program code for video feed for video.
* Conduct final robot testing.

#### 3.7 Updated Deadlines

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Tasks** | **Description** | **People** | **Resources** | **Deadline** |
| 1 | Procure material | Receive order for all materials needed | Caleb, Dr. Klink | Servocity Cart | 1/15/16 |
| 2 | Reassemble Roomba | Assemble Roomba and arm | Caleb, Jide | Roomba , Vex kit, hitec servos, Actobotics Grippers | 3/1/156 |
| 3 | Develop software to communicate | Write/adapt software core to power the Surface and control units. | Edwin | Xbox Controller, Laptop | 3/27/16 |
| 4 | Test communications between OCU and RCU | Ensure data connections | Edwin, Jide | Desktop, Window Tablet | 3/14/16 |
| 5 | Interface windows tablet to robot and motors |  | Edwin, Caleb | Window Tablets, motor controllers | 2/20/16 |
| 6 | Develop GUI | Write an interface to control total functionality of robot | Edwin, Jide | Computer, Visual Studio 2013 | 3/6/16 |
| 7 | Prototyping | Build functional system. Add rest of materials to the RC car. | Caleb | RC Car, Controller, Laptop, LEDs, Robotic Arm | 4/20/16 |
| 8 | Testing | Debugging | Jide | Robot, Controller, Laptop | 4/21/16 |
| 9 | Final Testing and Error Correction | Final Modifications | Group | Robot, Controller, Laptop | 4/22/16 |

# Appendix 4- Summary of Revision

Since the design plan didn’t deviate much from the original. The group decided to make very minimal changes to the design and strategy proposed. However, we edited some technicalities in our plans which is shown in the detailed circuit section. We also added the plan to use a Roomba as our main vehicle which makes the task a little easier to deal with. This allows us to focus more on software and arm functions rather than deal with the architecture of the robot.

Our main revision was to our organizational structure. We have been able to allocate specific responsibilities to individual members of the group. This is shown in the final project plan sections.

Finally, task-related deadline have been allocate to cater for recent developments in the completion rules and regulations. This is also highlighted in the Updated Deadline section.