Semester Final Report

2016 RASC-AL Robo-Ops Competition - University of Wyoming, Computer Science Team

Richard Yang - Computer Science Team Lead
Nasser Alawami
Mohammed Busaleh
Brian Moore
Ross Petrutiu

Outline:

1. Introduction
2. Hardware Setup
3. Connectivity and Networking
4. Robot Operating System
5. Remote Sensory
6. Outreach and Media

1. Introduction

“RASC-AL Exploration Robo-Ops Competition (i.e., Robo-Ops) is an engineering competition sponsored by NASA and managed by the National Institute of Aerospace. In this exciting competition, undergraduate and graduate students are invited to create a multi-disciplinary team to build a planetary rover prototype and demonstrate its capabilities to perform a series of competitive tasks in field tests at the NASA Johnson Space Center’s Rock Yard in May 2016. Allowable rover systems can include a single rover, multiple independently controlled rovers, or a parent rover with supplementary/scouting micro-rovers, all of which combined together are required to meet packaging and mass constraints.”

The University of Wyoming Robo-Ops team was one of the eight qualifying teams selected to receive a $10,000 award and full participation in the Robo-Ops competition in Houston, TX. The UW Robo-Ops team is composed of three divisions, Computer Science, Mechanical Engineering, and Electrical Engineering. Our entire team also presented a poster at the Fall College of Engineering and Applied Sciences Senior Design Symposium and will present at the symposium in the spring.

Group Members:
Richard Yang (ryang3@uwyo.edu) – Computer Science Team Lead
Nasser Alawami (nalawami@uwyo.edu)
Mohammed Busaleh (mbusaleh@uwyo.edu)
Brian Moore (bmoore16@uwyo.edu)
John Ross Petrutiu (jpetruti@uwyo.edu)

This document covers all of the Computer Science division’s duties and progress for this semester.

2. Hardware Setup

Motherboard:
Supermicro Mini ITX A1SRI-2558F-O
CPU: Intel Atom C2558 Processor
RAM: Kingston Technology 2x8GB 1600MHz DDR3L
Hard Disk: Samsung 850 EVO 250GB 2.5-Inch SATA III Internal SSD
Expansion Slots: 1x PCI-Express 2.0 x8 Slot
Ports: 4x USB 3.0 Ports and 2x USB 2.0

Computer Science was tasked with selecting new hardware (if necessary) for the rover. The previous year team for Robo-Ops used a Beaglebone chip as a motherboard of the robot. It has many limitations. The limitations are: it has only 2xUSB ports which is not enough because we are using at least 4xUSB for the Cameras, it has only one thread where we need more than that in order to control and stream form servo controllers and the cameras, and slow processing. The idea of having a motherboard with decent processing power is to have all the image processing on board instead of doing them through the network because of the limitation of the internet speed. Also, the power consumption by the mother is low which is 15 watts, which will survive for more than an hour with 12v battery. Currently, Ubuntu 14.04 is installed on the motherboard.

Servo Controller:
The Phidget servo controller is supposed to connect between the motherboard and the other hardware such as the arm and the wheels. Phidget has many libraries in ROS, which should be explained explicitly in the ROS section of this document.

Supermicro Mini ITX A1SRI-2558F-O

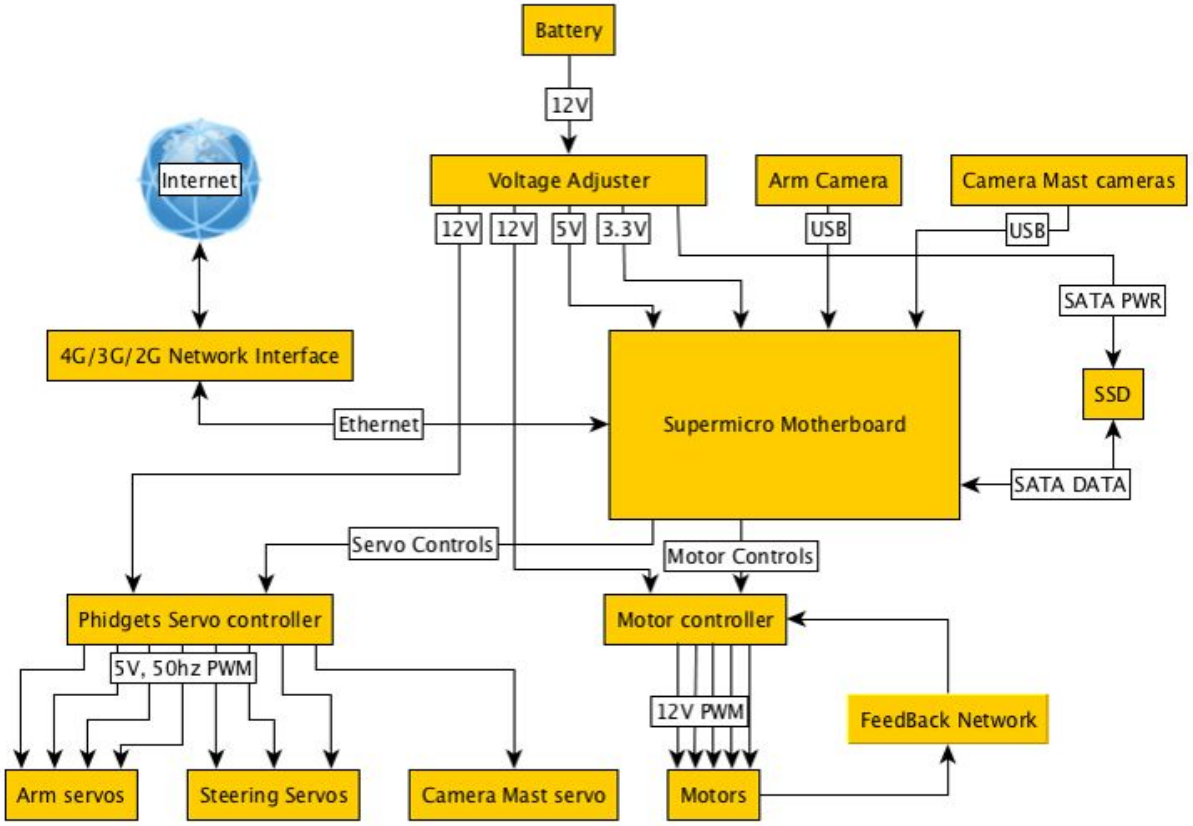


Diagram of the rover components and connections

3. Connectivity and Networking

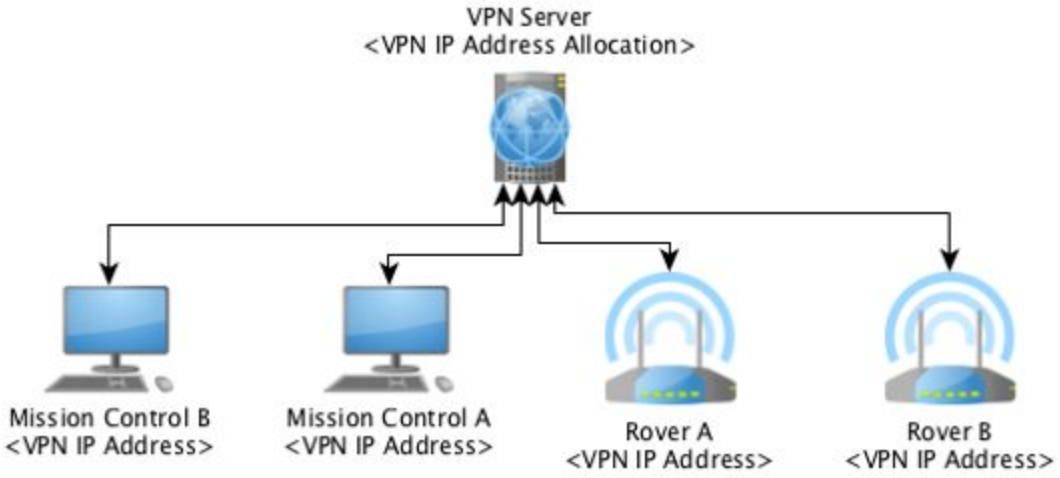
Networking:

Important in allowing *"Mission Control"* to remotely connect to the rover and control it from the UW Campus while it is at the NASA Johnson Space Center. One of the first obstacles that came up with connecting to the rover was finding out what the external IP address of the rover is to allow us to initially set up an SSH connection to it. So to accomplish this, we developed a script that will get run automatically on the rover when it turns on, and it will go visit a website that returns the external IP address. The script will then save that IP address into a text file and then upload that text file to a remote server where we can then download it. Which brought about another issue of figuring out how to allow the rover to secure copy the file, but we were able to set up ssh-keys so that the rover would be able to connect over SSH without having a user enter a password in. Then after some failed testing, and trying to connect to the rover with it on an android hotspot and setting up IP table rules, we figured out that most firewalls would cause us some issues. Where most firewalls won't allow traffic through that wasn't initiated by a host on the inside, which was causing a lot of issues for us to SSH into the rover or to SSH from the rover to campus. So after some trial and error we were able to figure out that we could set

up a Reverse SSH connection from the rover to our external server, where the server would be listening on port 2210 for any incoming SSH connections, then it would “route” that traffic back to the rover on the link that it established from the inside of the firewall. We would then from campus be able to SSH to port 2210 on the external server, which would do the same thing as with the rover, and set up the connection from the inside of the firewall which would allow Mission Control and the rover to communicate.

Connectivity:

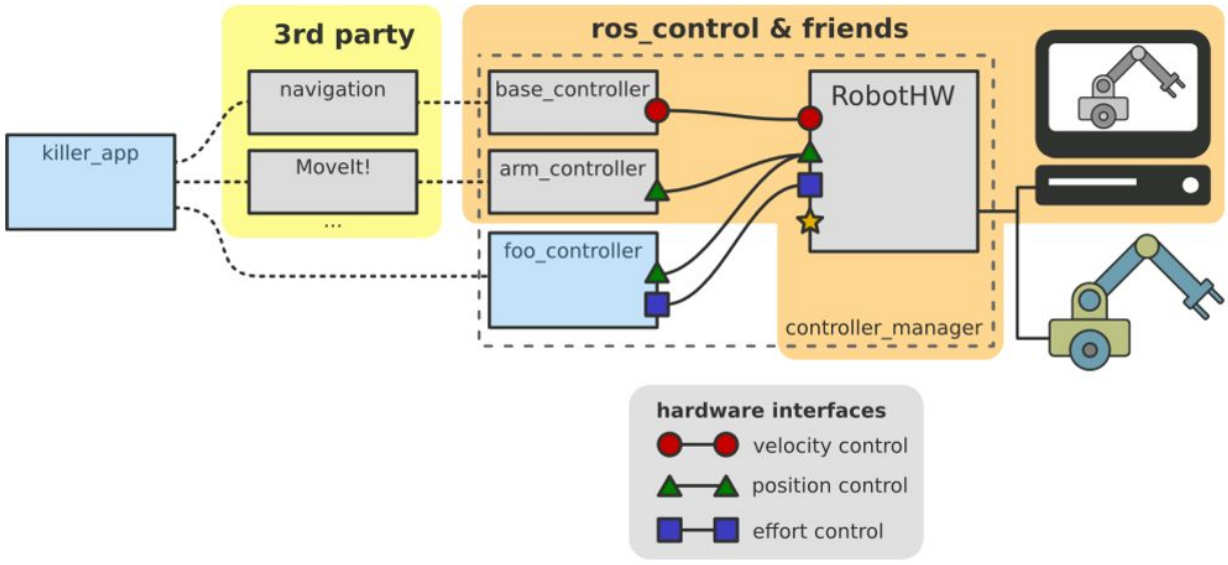
The connectivity aspect piggybacks off of the Networking side of the project, where before were able to actually communicate with the rover all of the networking scripts need to get ran on the rover so that it can establish a connection with our remote server. Then once the rover has connected to the server Mission Control can SSH to port 2210 on the server. In addition, at Mission Control there is going to be an Xbox controller connected to one of the usb ports on the computer, which is currently used to control the rover by sending commands through the ROS packages over the network. Though one of the main issues that we are working through with the controller is that the control for the arm on the rover is very touchy and will not always do what we want. Also once the connection is established between the rover and Mission Control, the rover will be sending a live video feed back to Mission Control so that we are able to see where we are driving the rover and if there are any rocks to pick up. The rover will also send a second video feed to our website on our server, so that anyone can tune in and watch what we are seeing from the rover.



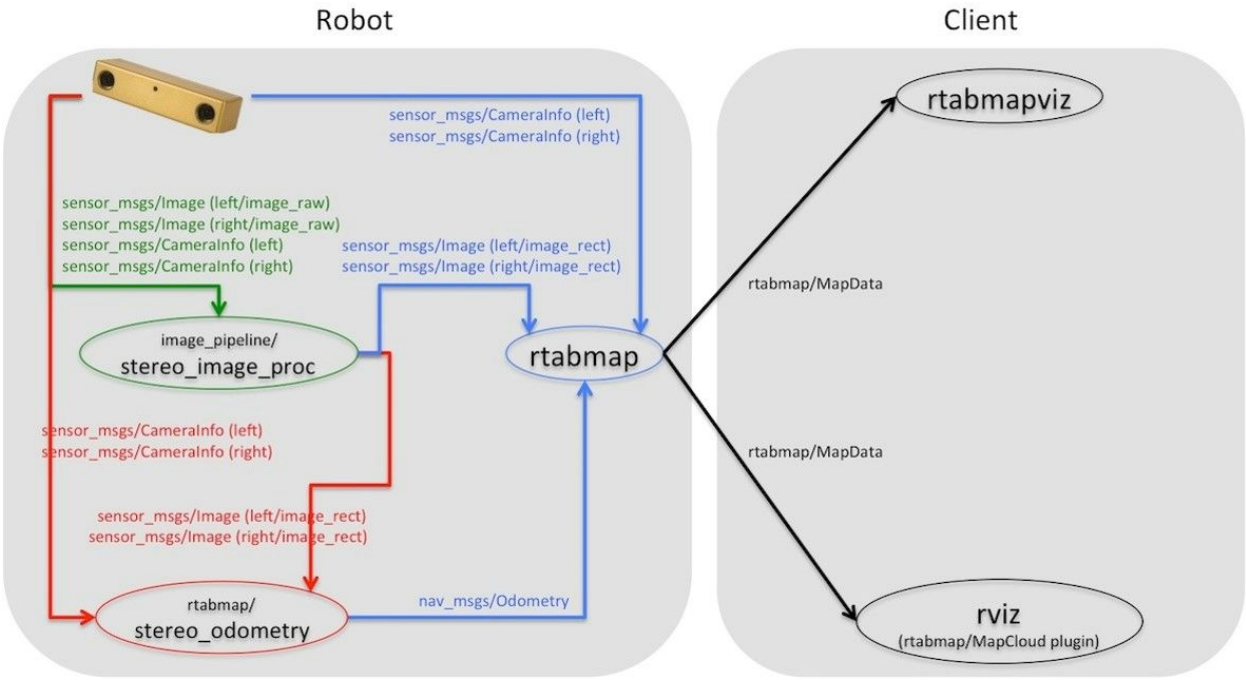
4. Robot Operating System (ROS)

ROS is a set of libraries that helps to create robot applications. It is built infrastructure for any robotic purposes. It saves the time of dealing with hardware installation and drivers installation too. The language that would be used to program the robot is C++ and the Python (if needed). Here some details about the ROS and the libraries we are using:

• ROS version used is Indigo.
• Currently used libraries are:
–Phidget: it has all the functions for controlling the servo
–Phidget_drivers: it consist the drivers of phidget
–Catkin: is the library that initialize ROS environment in order compiles other ROS libraries.
• Future libraries will be used:
–uvc_camera: it a library that allows streaming through ROS instead of using the Ubuntu. This library might be used to keep our work comprehensive
–robot_localization: navigation purposes
–nmea_navsat_driver.: the driver for the navigation chip.
–rqt: it is a library that creates GUI for the ROS in order to ease the control.
–***stdr_simulator*: it has 3d rigid libraries and simulation libraries we might need for the image processing and checking the rover status.**



Current ROS configuration for the arm controller



Blueprint for utilizing additional ROS packages in the future

5. Remote Sensory

An important component for this project is the ability for the rover to communicate sensory information to mission control. Currently, we are considering to employ the following sensors on the rover to relay such information:

Cameras:

Four Genius 1080p 120-degree Ultra Wide Angle Webcams are currently installed on the rover. The only down-side to these cameras is that they are manual focus, but once correctly set, they provide full resolution.
Two cameras are installed on a mast that extends from the chassis of the rover. The remaining two cameras are installed on the gripper arm. The feed from these cameras are handled by the Ubuntu libraries FFServer and FFMPEG, which provide software compression and streaming capabilities on the rover motherboard. We first start an instance of the FFServer on the rover motherboard. Port 8090 on the motherboard is used to display an HTTP webpage with the status of all camera feeds and streams. FFMPEG allows us to generate streams in the following formats: MPEG, MP2, OGG, RM, RA, MPJPEG, ASF, SWF, and AVI. Currently, we've tried two stream formats: MPJPEG and ASF. MPJPEG provides stills of the camera in JPEG format, updated every second. These stills can be accessed from the stream status page. ASF is a video stream format that can be viewed through VLC player. Both of these work nicely with our set-up. In addition, we also have to configure FFServer to accept camera feeds and actually stream them. We have bash scripts to create an instance of FFServer as well as configure each camera feed.
Next semester, we will use the camera feeds to generate a 3D map through a ROS package called RTAB-Map. This will provide a live 3D model of the environment to us at mission control.

GPS Sensor:

We plan to utilize a GPS sensor in conjunction with our camera streams to determine the location of the rover during the competition. It will also serve as a back-up, when any of the cameras fail during the competition we will still have information for the rover to go back to its initial position.

Inertial Measurement Unit (IMU):

The IMU is a sensory board that provides an accelerometer, temperature, and gyroscope information. We plan to have another dynamic webpage that will self-update every second from the sensory board.

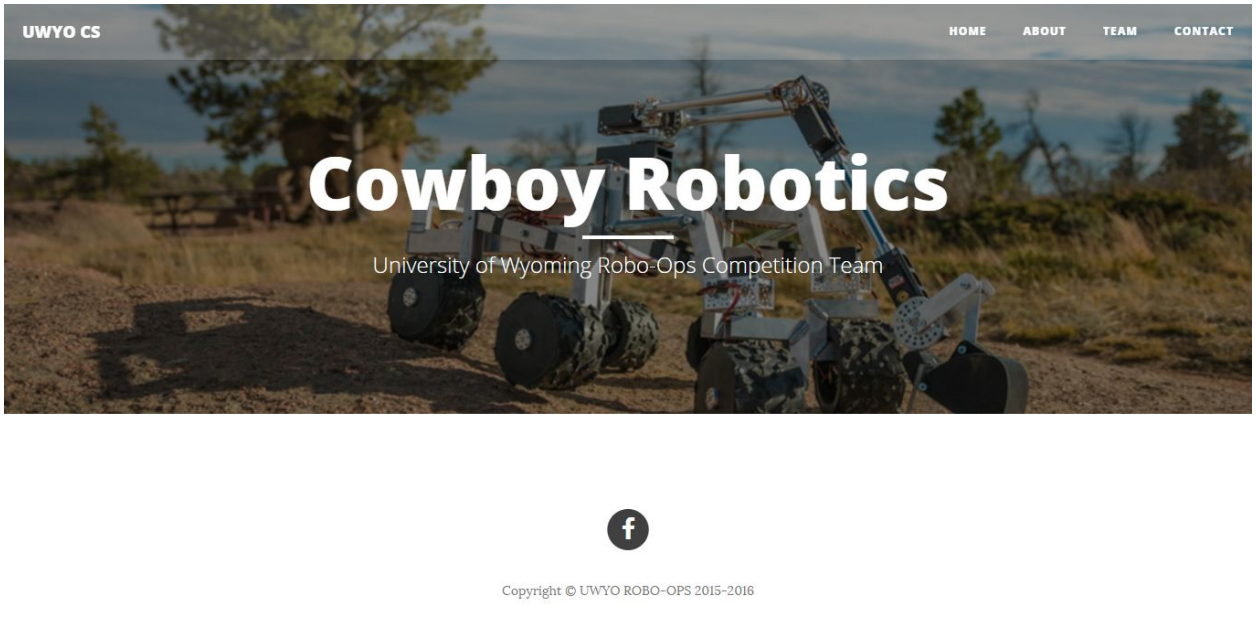
6. Outreach and Media

One of the completion requirement is “public engagement”. So, we have created a Facebook page to post updates and pictures. Also, under the public engagement, we are required to provide a Livestream of our camera feed during the completion using one of the following websites: YouTube or UStream.

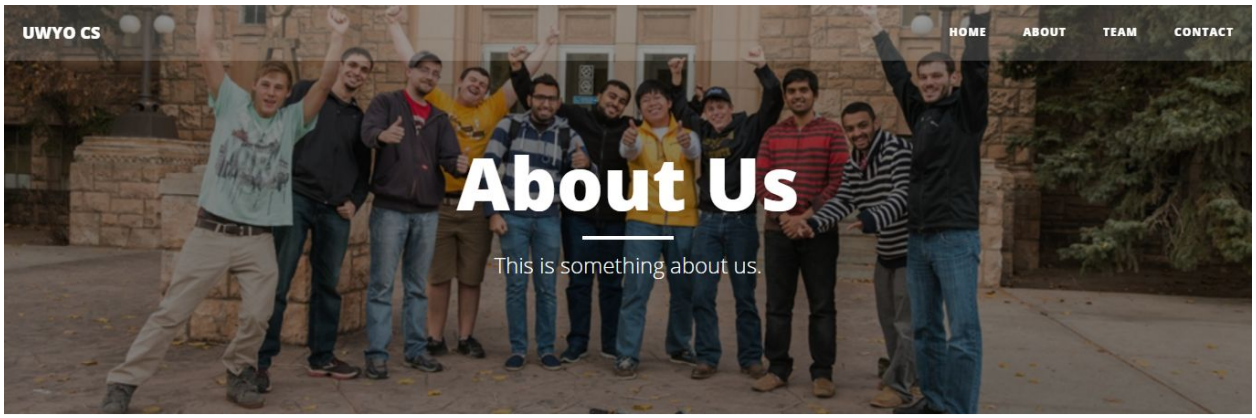
Therefore, we have created a website by using a template from Bootstrap since it is compatible with different screen sizes. We have made the template fit our needs by adding some pages and removing others. Using the website would be a more official way to achieve the public engagement requirement. Also, we will be using our website to post our updates, pictures and livestream our camera feed, which are installed on the rover.

Here are screenshots of the website version (0.2).

First page: the (home) page:
We made it very simple with a picture of the rover and a navigation tab on the top. Also, we have added a link to our Facebook page at the footer and we will add a link to our GitHub after we organize our code/s.



Second page: the (about or about us) page:
It has a group picture of all teams, with a short paragraph stating our goal. Also, we have added a link to the university article, which has more details about the completion and the team members.

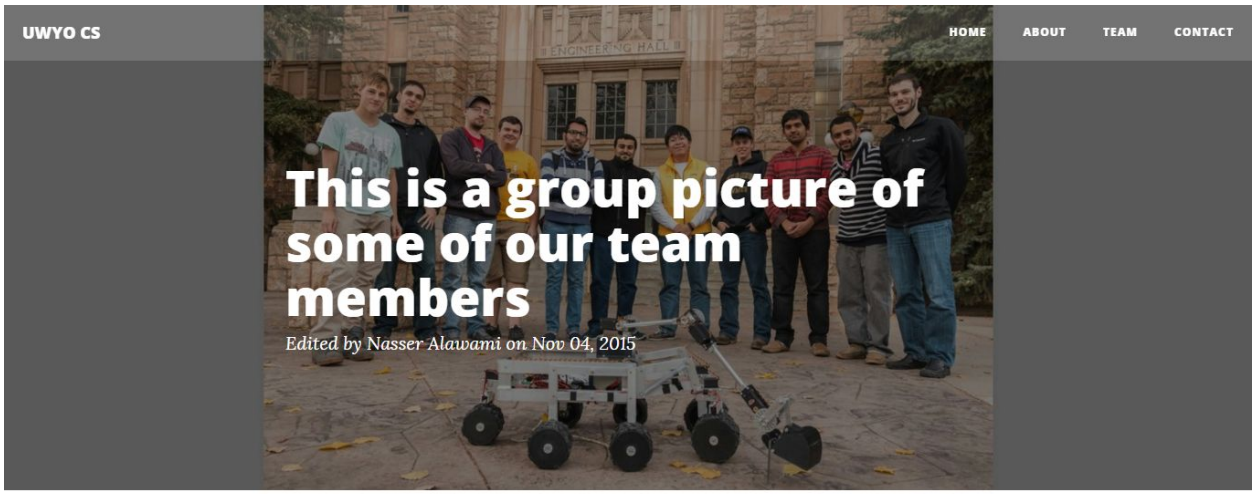


We are groups of senior class students from different departments and different cultures. Teamed up to compete in the NASA 2016 Robotics Tele-Operations Competition.

Our university team is one of 8 other teams that were chosen to compete in the NASA 2016 Robotics Tele-Operations Competition.

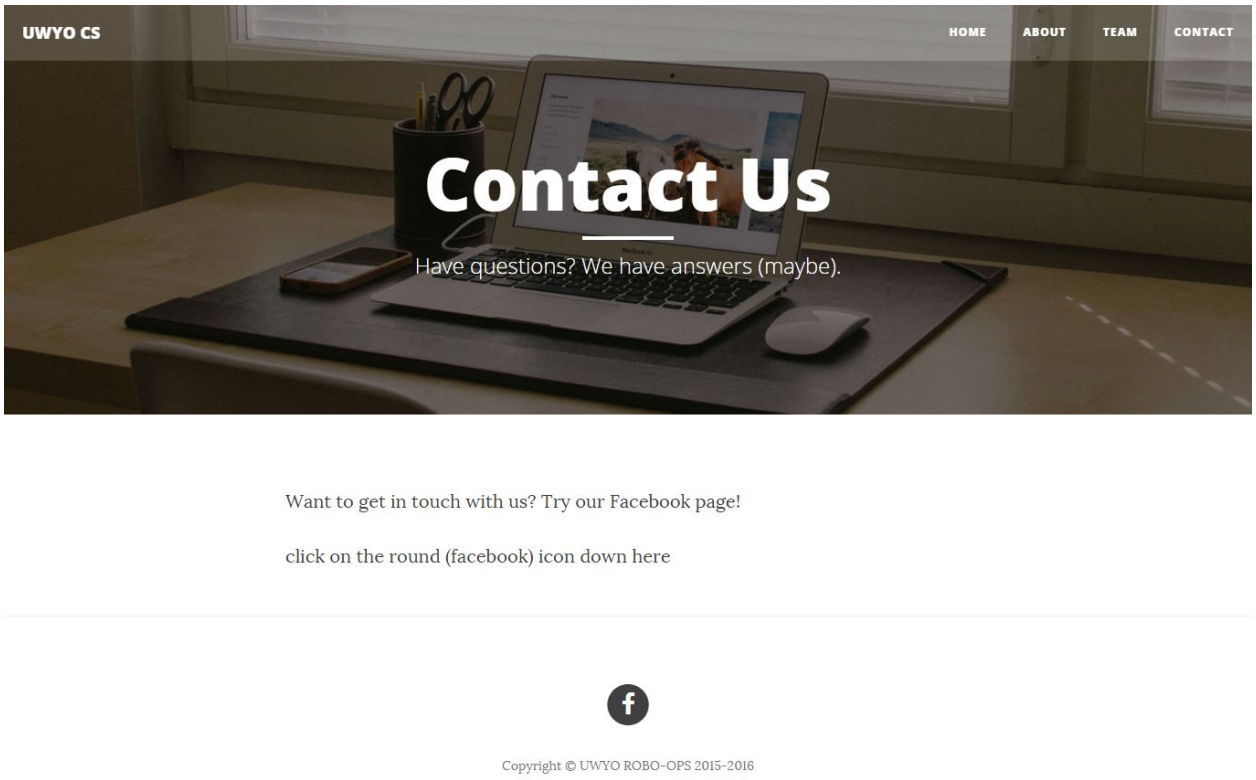
Read more

Third page: the team page:
Originally we created that page to add the team's member's information, but we decided to change it and add a blog page so we can post our weekly updates. As it is required by the completion rules. We would add the team member's information in the blog page.



The page will be updated soon!

Fourth page: the (contact us) page:
A page that has ways to contact our team. We are planning on creating an email for the team, which the people who are interested can contact us easily.



The team has decided to use (WordPress) since it provides an easy way to make a blog page and updated it frequently. That is, instead of dealing with all the HTML scripting because not all the team members have knowledge or experience with HTML scripting and CSS. Also, we will setup the YouTube Livestream and the link to on the website.