Letter to the judges,

The Purdue IEEE ROV Team has always focused its media outreach following the competition as opposed to before. We believe, and our media contacts agree, that it is more interesting once we can say we know our final placement at the international level. It is also more useful in that it attracts interested individuals, new team members, and new funding when we need it most. Therefore, we have not reached out to many news outlets yet this year. However, we do plan on contacting the local news outlets soon after the competition.

Here are a few examples of our news attention following last year's ROV competition:
https://engineering.purdue.edu/IE/Spotlights/ieee-rov-team-places-2nd-at-the-mate-international-rov-competition

http://insgc-bc.blogspot.com/2012/08/purdue-mate-rov-entry-world-famous.html

http://www.firstpost.com/topic/organization/purdue-university-purdue-2012-rov-potatos-video-Po8qVXX_Of4-90715-19.html

http://catalog.e-digitaleditions.com/i/98762/24

Two media agencies decided to follow through on our story right before our departure for Seattle:

- WBAA, a public radio station based on the Purdue campus
- Purdue *Exponent*, an independent student newspaper

Results of those stories, as well as follow-up articles and other new contacts, will be available after the competition.

Edison Submersibles

Contact: Michael Hayashi
Tel: (309) 370-4274
E-mail: mhayashi@purdue.edu

FOR IMMEDIATE RELEASE

PURDUE STUDENTS COMPETE IN INTERNATIONAL ROV COMPETITION
Aiming for First Place Internationally with Brand New Vehicle

The Purdue IEEE ROV Team has been competing in the MATE (Marine Advanced Technology Education) Center's ROV (Remotely Operated Vehicle) International Competition for four years. This year they are looking for a win with their latest innovative ROV from the clever minds of its engineers, the *Model N*.

The MATE competition pits school teams against each other to see who can create the most efficient, well-designed, and cost-effective underwater robots to complete an assigned mission. The mission changes every year and has included submarine rescue missions, oil well capping operations and underwater volcano exploration. This year's mission from the MATE Center focuses on seafloor observation networks. The tasks revolve around deploying new instruments and servicing old equipment in one such network designed to measure tectonic activity off the coast of Oregon.

This year the Purdue team, Edison Submersibles (formerly Aperture Aquatics), has built an ROV that is capable of completing the mission in the shortest amount of time. With greater access to machining shops and funding, the team was able to machine all the custom-designed ROV components to the necessary high standards. This includes a lightweight, anodized aluminum frame and custom-designed waterproof enclosure for on-board electronics and pneumatic systems. With the vehicle's power board and control systems designed and mounted by the students, ROV *Model N* represents a logical successor to the team's legacy of power, sophistication, and performance. Perhaps most impressive is the fact that the team designed and constructed the vehicle from the ground up and in house, from the layout of the frame to the software, and including the design and layout of the circuit boards.

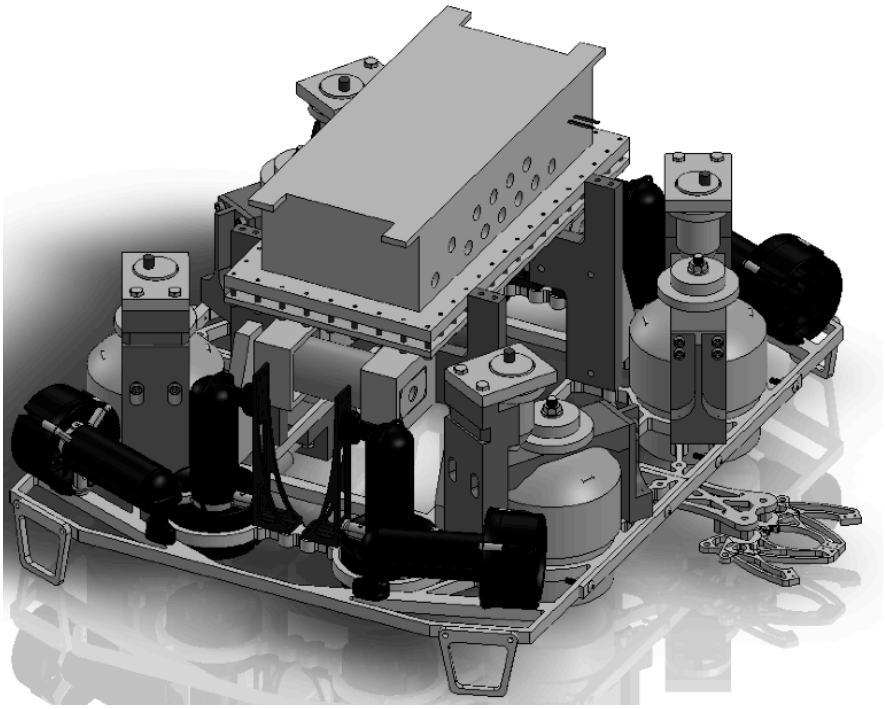
The Purdue IEEE ROV Team would like to thank the Indiana Space Grant Consortium, the Purdue Office of the Provost, and SolidWorks Corporation for their support that makes our team competitive at the International Competition in Seattle, WA. With all the advancements the team has made over the past few years, and with a second place finish in 2012, Edison Submersibles sees this year as their shining chance to seize victory at last.

Edison Submersibles

Purdue University

West Lafayette, Indiana, USA

ROV Model N



Michael Hayashi

Craig Cainkar
Jack McCormack
Amanda Burgoon Skelton
Nick Molo
Houston Fortney

Chandrahas Reddy
Kevin Rockwell
Reid Fouch
Flor Albornoz
Dharna Pahuja

Edison Submersibles

Purdue University IEEE ROV Team

West Lafayette, Indiana, USA

Craig Cainkar
PR & Advertising
Professional Writing
Senior
Communications
1st Year Competing

Jack McCormack
Mechanical Engineering
Senior
Mechanical Team Co-lead
3rd year competing

Amanda Burgoon Skelton
Biochemistry
Senior
Mechanical Team Co-lead
1st Year Competing

Nick Molo
Computer Engineering
Senior
Electrical Team Lead
2nd Year Competing

Houston Fortney
Electrical Engineering
Junior
Electrical Team
2nd Year Competing

Chandrahas Reddy
Computer Engineering
Sophomore
Electrical
1st Year Competing

Kevin Rockwell
Computer Engineering
Junior
Electrical Team
1st Year Competing

Reid Fouch
Electrical Engineering
Technology Education
Senior
Electrical Team
2nd Year Competing

Flor Albornoz
Electrical Engineering
Sophomore
Electrical
1st Year Competing

Dharna Pahuja
Computer Engineering
Junior
Electrical
1st Year Competing

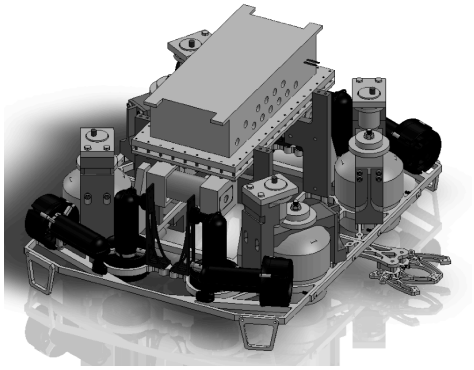
Distance Traveled: 2915 KM

Year Competing: 5th Year

2012 Finish: 2nd Place

Range of Grade Levels:
Sophmore to Senior

Michael Hayashi
Electrical Engineering
Junior
Team Captain
2nd Year Competing



Total Cost of ROV: $20,758.90 USD

Total Weight in Air: 19.0 kg

Overall Dimensions: 71 cm x 69 cm x 51 cm

Primary Materials: aluminum, stainless steel, polyoxymethylene (delrin), high-density polyethylene, polycarbonate

Special Features: immense power to weight ratio, lightweight MIC 6 aluminum frame, high-accuracy length measuring tool, single fixed manipulator for simplicity, entirely custom onboard electronics system and software

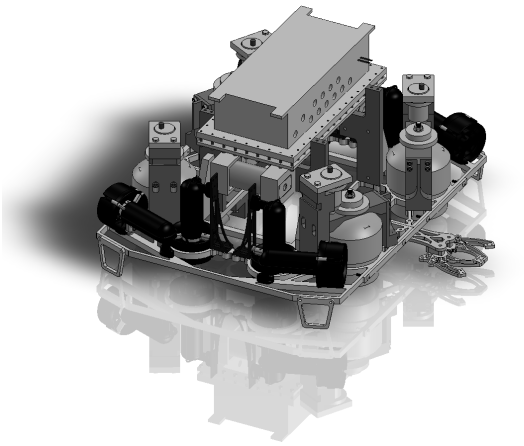
Key Safety Features: shrouds on all propellers, plastic propellers, complete FEA analysis with a safety factor of 2, all electrical wiring sealed with liquid electrical tape and heat shrink, multiple fuses throughout electrical system, emergency quick release pneumatic valve

Model N

2013 Technical Report

Purdue University
IEEE ROV Team
West Lafayette, IN

Technical Report: ROV Model N



Team Members
Mentor-None

Name	Position	Major
Michael Hayashi	Team Captain	Electrical Engineering
Jack McCormick	Mechanical Team Co-Lead	Mechanical Engineering
Amanda Burgoon Skelton	Mechanical Team Co-Lead	Biochemistry
Nick Molo	Electrical Team Lead	Computer Engineering
Houstin Fortney	Electrical Team	Electrical Engineering
Reid Fouch	Electrical Team	Electrical Engineering
Chandrahas Reddy	Electrical Team	Computer Engineering
Kevin Rockwell	Electrical Team	Computer Engineering
Flor Albornoz	Electrical Team	Electrical Engineering
Dharna Pahuja	Electrical Team	Computer Engineering
Craig Cainkar	Communications	Professional Writing

2013 Technical Report

Abstract

Edison Submersibles, formerly Aperture Aquatics, designed and constructed ROV *Model N* to meet and exceed the requirements set forth by the 2013 MATE International ROV Competition. This includes installing the Scientific Interface Assembly and replacement Acoustic Doppler Current Profiler, leveling the secondary node, and deploying a transmissometer to measure ocean turbidity. At 71 cm long, 69 cm wide, and 51 cm tall, ROV *Model N* is capable of performing all these tasks with peerless speed and maneuverability.

Designed with reliability, stability, and speed in mind, ROV *Model N* is capable of maneuvering with six degrees of freedom. It has four thrusters for horizontal movement and four thrusters for vertical movement. The payload tools have been designed specifically for the mission and include a main gripper, turning motor leveling system, and spring-loaded measuring tool. All of the electronic hardware responsible for power management, vehicle movement, and sensor data collection, has been designed and fabricated from the ground up. The on-board and base station software was designed and developed by the company. Although it was a significant challenge to custom design electronic hardware, ROV *Model N* is fully functional.

The remainder of this document covers the design process and specifications of Edison Submersibles' ROV *Model N*. Also included are an expense report and a reflection on the issues that arose in the design process.

Michael Hayashi
Electrical Engineering
Junior
Team Captain
2nd Year Competing

Craig Cainkar
PR & Advertising
Professional Writing
Senior
Communications
1st Year Competing

Jack McCormack
Mechanical Engineering
Senior
Mechanical Team Co-lead
3rd year competing

Amanda Burgoon Skelton
Biochemistry
Senior
Mechanical Team Co-lead
1st Year Competing

Nick Molo
Computer Engineering
Senior
Electrical Team Lead
2nd Year Competing

Houston Fortney
Electrical Engineering
Junior
Electrical Team
2nd Year Competing

Chandrahas Reddy
Computer Engineering
Sophomore
Electrical
1st Year Competing

Kevin Rockwell
Computer Engineering
Junior
Electrical Team
1st Year Competing

Reid Fouch
Electrical Engineering
Technology Education
Senior
Electrical Team
2nd Year Competing

Flor Albornoz
Electrical Engineering
Sophomore
Electrical
1st Year Competing

Dharna Pahuja
Computer Engineering
Junior
Electrical
1st Year Competing

Table of Contents

2013 Technical Report

Fig. 1 - The very first design sketch of ROV Model N.

Mission Summary

Task 1: Complete Primary Node and Install Secondary Node

The Scientific Interface Assembly (SIA) is carried down in ROV *Model N*'s manipulator, and the pilot lowers it into position on the Backbone Interface Assembly (BIA). Using the hook on the top of the manipulator, the pilot may grab the Cable Termination Assembly (CTA) from the seafloor, hold it in an upright orientation with a hook on the top side of the manipulator, and insert it into the bulkhead connector on the BIA. With these steps complete, ROV *Model N* may remove the pin anchoring the secondary node to the elevator with the manipulator and retrieve the secondary node. By fixing the end of the string to the red pipe at the base of the BIA, the pilot may move the ROV in the direction of the deployment site and read the distance on the base station from the gray code encoder on the spring-loaded measuring tool. Once the secondary node is in the correct deployment site, the ROV will land on top of it such that each of the turning motors aligns with the four handles on the legs of the instrument. By using accelerometer data, a software script will be run that auto-levels the secondary node with manual adjusts following under the direction of the co-pilot. The pilot then moves ROV *Model N* off the secondary node to open the BIA door, retrieve the secondary node cable connector using the manipulator and attached hook, and insert it into the appropriate bulkhead connector.

Task 2: Construct and Install Transmissometer

ROV *Model N* takes the transmissometer down from the surface in the manipulator. The ROV rests the transmissometer on the simulated ocean vent using the guideposts to achieve alignment of the transmitter and receiver through the medium. Continuously gathering data from the internal lock-in amplifier, the transmissometer will sit in that position for the duration of the mission run. Using a Vernier voltage probe, relative opacity is plotted against time on the company's custom GUI.

Task 3: Replace Acoustic Doppler Current Profiler (ADCP)

Power will be disconnected from the suspended mooring platform by using the manipulator to remove the platform connector from the

bulkhead connector. The fast-acting pneumatic manipulator is next used to turn the handle in order to unlock the hatch before opening the hatch itself. The ROV removes the ADCP from its cradle and returns it to the surface. The mission crew exchanges the old ADCP for a new one. The ROV then returns to the mooring platform to install the new ADCP. The incredible power-to-weight ratio of the ROV allows a two-way trip to the surface in a short amount of time. The pilot uses the manipulator to close the hatch and relock it using the handle. By grabbing the platform connector using the hook on the manipulator, the connector may be re-inserted into the bulkhead connector with the proper orientation in short order.

Task 4: Remove Biofouling

During the completion of the other tasks, the ROV may return video showing biofouling on the seafloor equipment and instruments. The pilot may position the ROV directly in front of its forward thrusters and activate "tornado mode." The "tornado mode" is a programmed vector thrust configuration that does not displace the ROV, but spins the forward motors quickly. The turbulence from the front thrusters will safely detach the biofouling organisms without requiring the pilot to release any object in the manipulator. Once all mission tasks are complete and all biofouling is removed from the equipment, the ROV may return to the surface.

Design Rationale

Shape and Frame

The shape of ROV *Model N* was controlled by the mission requirements. In order to have the leveling system for the secondary node require only a single docking, the turning motors were spaced so as to mount over the legs of the secondary node. In order to clear the center of the secondary node, the electronics box was mounted atop four posts so that it could have a clear view of the leveling bubble on the instrument. The next decision was to have the manipulator centered at the front of the vehicle at the extreme lower end to minimize difficulties that could arise when picking up objects from the seafloor. The measuring tool was mounted on the side of the vehicle to prevent interference with the manipulator and turning motors. Wanting to keep the the length and width of the vehicle as compact as possible, the thrusters and accumulators filled the

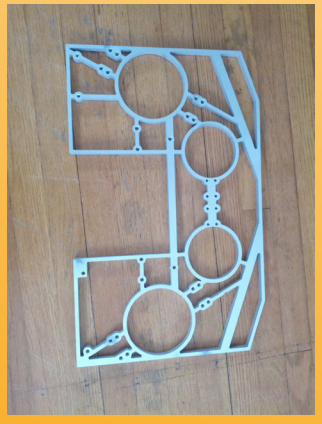


Fig. 2 - One frame half immediately after machining.

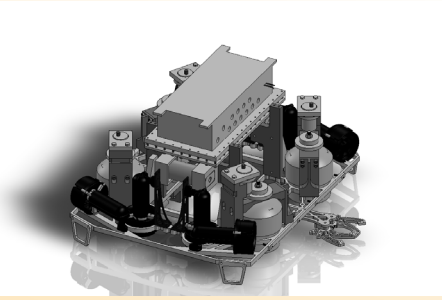
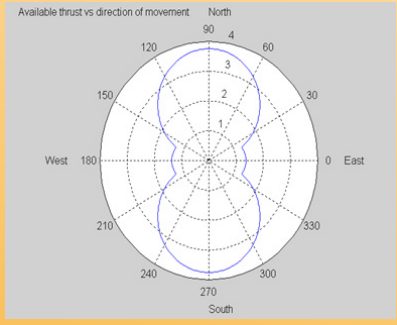


Fig. 3 - Frame assembly.



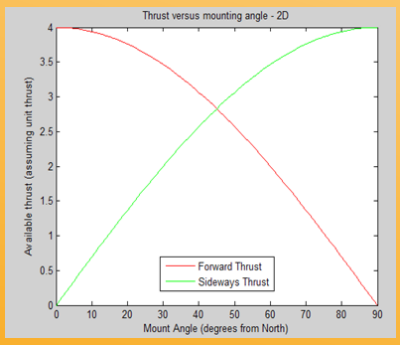


Fig. 4 -Thrust directional analysis.

remaining space by taking advantage of the vertical space available.

The frame of ROV *Model N* was cut by a water jet from a single piece of .635-cm thick Mic 6 aluminum. The material was chosen for its light weight and durability to avoid warping. The intricate shape was designed in Solidworks only after the position of all other attached items were finalized. This design paradigm allows the company to design each tool for its purpose in the mission rather than be forced to design to a pre-existing infrastructure. To reduce the complexity of the design when only a few payload tools are needed, a single-layer frame was implemented rather than a more costly dual-plane frame as used on Aperture Aquatics' 2012 vehicle, ROV *PotaTOS*.

To protect against corrosion, every stainless steel bolt that goes through the aluminum frame is covered in anti-corrosion Tef-Gel brand fastener lubricant. The head of the bolt is also spaced away from the frame by a washer. To further protect the vehicle, every aluminum piece on *Model N* is anodized. To protect the bottom of the frame and tools from the ground, the vehicle is raised on four skids that are 6.8 cm high. These feet are trapezoids with a flat bottom which allow *Model N* to move in any direction while on the floor.

Thrusters

ROV *Model N* uses eight new, custom-designed Seabotix SBT166 thrusters for propulsion. Each thruster is only 17-cm long and produces a maximum of 28.4 N of thrust. While the acquisition of new thrusters presented a significant expense to the company, this purchase was deemed necessary when the thrusters used over the past three years were deemed unreliable for another year of use. These custom-designed thrusters are made to run off 50 Vdc, which eliminated the need for expensive DC/DC conversion boards as used in past years. Because these thrusters communicate between their built-in motor drivers and the microcontroller using RS-485 serial communication, the team could remove their own motor driver boards and utilize a more straightforward communication protocol. Overall, the decision to use brand-new thrusters allowed for reduced electronics and software complexity without sacrificing motor performance.

The four horizontal thrusters are placed at a 20 degree offset to achieve the best balance between turn speed and forward thrust. This angle was determined by performing a thruster analysis at

many different angles until a desirable result was found. The vertical thrusters are placed at each corner of the vehicle. The use of four thrusters for movement along the z-axis not only gives the vehicle improved vertical thrust, but also allows for pitch and roll control. Pitch and roll control gives the pilot greater maneuverability which is needed for the fixed manipulator when picking up objects on the seafloor.

The output of each of the thrusters is determined by the base station software's thruster mixing algorithm, and is discussed in the software section of this report.

Cameras

There are three cameras on-board ROV *Model N*: the main front camera, the rear facing camera, and the down facing camera. The combination of these views allows for a composite image of most of the payload tools on the vehicle. The electronics system on ROV *PotaTOS* can transmit two of the three cameras at any given moment, with the front camera constantly sending video. Switching between the cameras is done by the pilot on the handheld controller. All three cameras use the same Super Circuits Wide Angle Board Cameras inside custom HDPE waterproof enclosures, with the exception of the down facing camera which is housed directly in the electronics box. Analog cameras were chosen for added reliability, lower cost, and simplicity. These particular cameras were chosen for their low weight, small size, and high quality.

The custom enclosures are made with a 3D printer from Taulman 618. This material was chosen after testing Delrin, Aluminum, and HDPE together in the past. Aluminum worked as a watertight material, but was too heavy. Delrin leaked through the side walls of the enclosure when at depths below approximately 35 m. HDPE was both watertight and lighter than either the Delrin or Aluminum. Taulman 618 was eventually chosen for its superior durability and water tightness. These camera enclosures are also used to hold the Transmissometer TX board and Transmissometer RX board.

Electronics Box

With the desire to implement the leveling system for the secondary node in only a single docking, the shape and placement of the electronics enclosure on ROV *Model N* needed to be completely

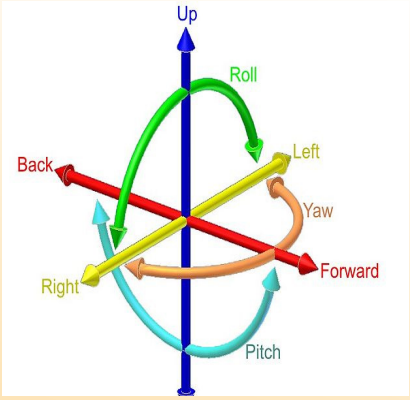


Fig. 5 - A visual representation of six axis movement. (Image courtesy of Wikipedia.org)

Fig. 6 -Testing the O-ring seal on the camera enclosures.

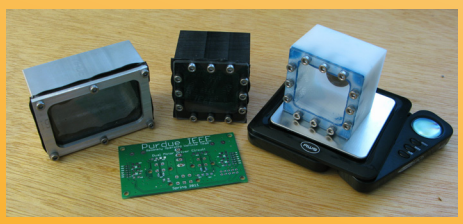


Fig. 7 - Camera prototypes.

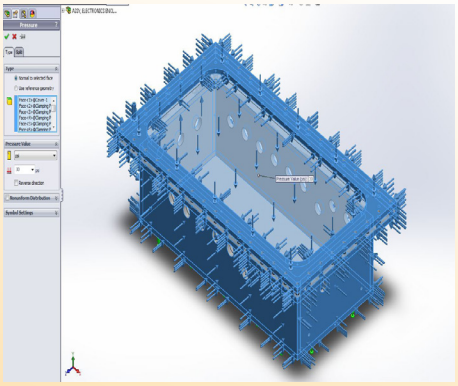


Fig. 8 - Pressure Test on Electronics Box at 208 kPa.

Fig. 9 - The electronics box being machined.

redesigned. After much deliberation, it was decided to create a rectangularly shaped electronics box made from 9.525 millimeter thick aluminum sheets with a transparent, 6.35 millimeter thick polycarbonate, downwards-facing lid. By raising the electronics box on thick stilts, the bulk of the secondary node could fit underneath the box while the handles on the secondary node legs fit under the turning motor shrouds. This design has the benefit of offering a large volume to fit all water-sensitive parts while protecting a camera that can be used to monitor the leveling bubble seafloor instrument during the deployment process.

Inside the volume provided by the box, the entire onboard electronics system lies protected by the forty-six bolts affixing the lid to the rest of the box. While accessing the electronics may prove more cumbersome than in years past, the simplicity of the overall design saves machining costs because welding the aluminum sheets is more cost-effective than milling carriers and designing round polycarbonate tubes as the company has done before. Twenty-four holes provide all the connections necessary for the electrical systems, pneumatic solenoids, and camera housed within the electronics box. In order to save these vulnerable components in the chance of a leak, water-detection systems will be employed using software.

Buoyancy and Ballast

Because ROV *Model N* was designed entirely in Solidworks, accurate volume and mass estimations were available to allow for buoyancy calculations. Because the electronics box is large, there is enough enclosed air to almost balance ROV *Model N*, though the vehicle will still sink slightly. In order to achieve a slightly positive buoyancy, a small amount of hydrostatic proof foam is cut to fit in the crevices of ROV *Model N* and painted for a proper seal and finish. This step will take place two weeks prior to traveling to the international competition to ensure that all modifications that might need to be made can be completed beforehand. The frame of *Model N* has a hole on each corner for .64 cm diameter bolts. These bolts can be ladened with enough stainless steel washers to finely tune the vehicle to a neutral buoyancy. This method of ballast also offers an easy method of fine tuning balance to assure that the vehicle sits level in the water.

$$F_{net} = 0 = mg - \rho_f \mathbf{V}_{disp} g$$

Manipulator

Model N has one fixed manipulator. Having multiple manipulators leads to extra cameras, extra weight and off-vehicle-center mounting of the gripper, which is more challenging for the pilot. This manipulator is used to transport the SIA and the tools to their receptacles. Instead of depending on a moving manipulator, the ROV's thruster layout enables the ROV to pitch up and down without much difficulty in controlling the vehicle. The *Model N*'s manipulator is coated with an ultra-soft polyurethane also know as Sorbothane. It is a nonporous material with a fluid-like consistency and is very durable and elastic in nature.

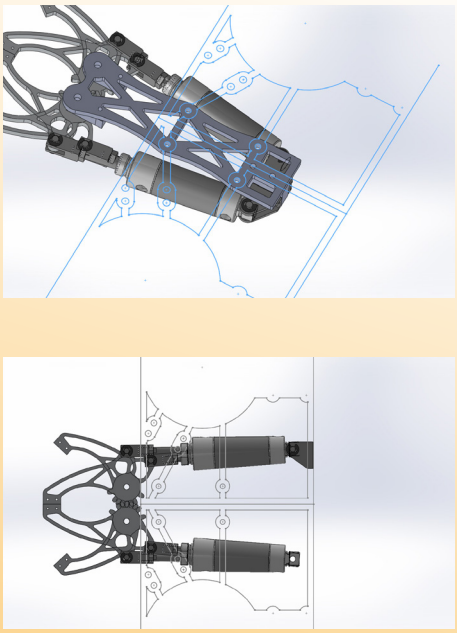


Fig. 10 - Manipulator MK2 with frame layout.

The manipulator was made from aluminum due to its strength and lightness. It is powered by dual pneumatic pistons. Initially during the design process a single piston manipulator was considered but then later ruled out, as having dual pneumatic pistons made the system more reliable. If one piston fails, the other piston still assures the functioning of the manipulator.

Turning Motors

The turning motor leveling system was dimensioned to fit over the legs of the secondary node, with the bulk of the instrument fitting under the electronics box. Built around a bilge pump, each turning motor consists of a plastic shroud, two tines, and a 3:1 drive gear system all attached to an ABS vertical mount. The two tines fit over the handles on the secondary node legs such that all four rest underneath the turning motor shrouds. The gearing increases the output torque enough to turn the handles. A software script may then be activated that automatically levels the vehicle according to accelerometer data. The copilot may then fine tune the secondary node using manual control over the four turning motors until the leveling bubble is in the correct location. Once the secondary node is level, the ROV may lift straight off the instrument, having deployed the device with only a single docking.

Measuring Tool

ROV *Model N* is equipped with a measuring tool consisting of a 7.62-cm spool with a string wrapped around it. In the fully-wound state,

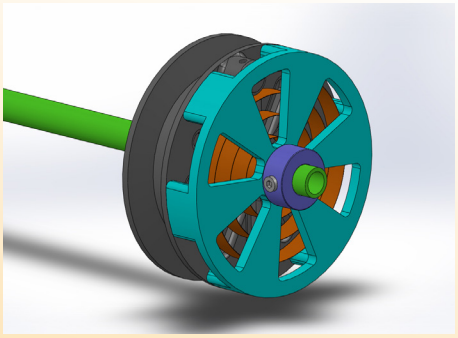


Fig. 11 - Spool assembly, 4 inch

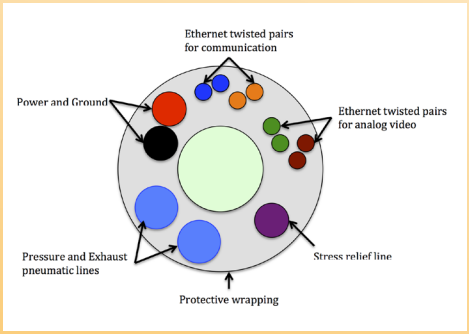


Fig. 12 - Diagram illustrating the innards of the Model N*'s tether.*

a tab slightly protrudes from the vehicle such that it can easily latch onto the red pipe at the end of the Backbone Interface Assembly. As the vehicle moves towards the direction of the secondary node deployment site, the spring-loaded spool unwinds while keeping the string taut. A twelve-notch gray code encoder can measure the length of string in the water down to within 2 cm. With this information, the pilot may find the correct deployment site for the secondary node. With the toggling of a switch on the interface, the pilot may reverse the direction of the vehicle and still track the length of string unwound from the spool. By heaving the vehicle upwards, the pilot may then release the tab from the red pipe.

Tether

ROV *Model N* is equipped with a 23 m long neutrally buoyant tether for power, communication, and pneumatic air supply. For power, the tether has a twisted pair of 10 AWG wire. Communication is handled through an Ethernet wire, using two twisted pairs for analog video transmission and another two twisted pairs for serial communication. There are two pneumatic tubes in the tether: one to send pressurized air to the ROV and another to vent air used by the system to the atmosphere. To make the tether neutrally buoyant, there is a strip of foam along its length with a cross section of .92 cm x 1.3 cm. To protect the tether and make it easier to manage, there is a snake-skin like wrap on the tether.

Pneumatic System

The pneumatic system on ROV *Model N* is used to drive two of the vehicle's tools: the manipulator and the backup measuring system, which is a piston-driven pin release for a fixed-length measuring rod. A compressor is plugged directly in to the tether. The other end of the tether is plugged in to a pressure accumulator on the ROV. This accumulator assures that pressure supplied to the tools does not drop if both tools are actuated simultaneously due to the time it would take to send pressurized air over the length of the tether. This air then travels in to a pneumatic solenoid bank. This bank has a valve for each tool and is controlled by the ROV's electrical system (which takes in controls from the Xbox 360 controller). Air from these valves is then directed to appropriate pneumatic pistons to drive each tool. Once air is used, it is vented to another pressure accumulator. This accumulator serves the same purpose as the other, assuring a stable supply of pressure. The air then travels from this accumulator up the

tether again to a venting silencer at the surface. This silencer quiets the pneumatic system without negatively impacting performance. The pneumatic system vents at the surface instead of at the ROV to increase the pressure differential since the pressure created by surrounding water at depth would reduce the effective pressure of the system.

Transmissometer

The company-designed transmissometer is an independent system from ROV *Model N* that receives 12 Vdc power from the surface. Its tether consists of a power line, ground line, and five lines of CAT5 ethernet cable. The body of the transmissometer consists of a rigid frame of ABS plastic made negatively buoyant by attached weights and held upright by sealed foam. The frame then rests against the alignment guides on the simulated ocean vent. Two of the company's camera boxes are positioned facing each other through the target area to house the sensing apparatus.

The transmissometer transmitter (TX) board is mounted inside one of the camera boxes on the frame. The TX board consists of three parallel infrared LED's that shine out of the box. The LED's are controlled by two signals at their positive and negative ends.

The transmissometer receiver (RX) board is mounted inside the other camera box. The RX board receives power from the surface and sends an output signal from a phototransistor. As the infrared light from the TX board is attenuated through the target medium, the phototransistor releases more electrons from the base-collector junction, increasing the current through the device. This increase in current lowers the voltage of the output signal due to the biasing of the phototransistor. The output signal is passed through a voltage follower to avoid degrading the signal.

The transmissometer surface board receives the signal from the RX board and compares it to a signal in-phase and at the same frequency as the pulse sent to toggle the LED's on the TX board. By passing the two signals through a lock-in amplifier, only a DC signal will remain that is associated with that specific frequency, even in a noisy environment with other signals of similar frequency. By passing the DC signal to a Vernier Voltage probe, the changes in this DC signal may be interpreted as relative ocean turbidity and graphed against time using custom software.

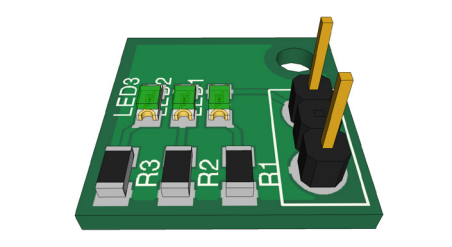


Fig. 13 - Transmissometer TX board.

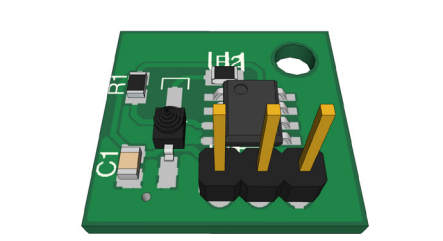


Fig. 14 - Transmissometer RX board.

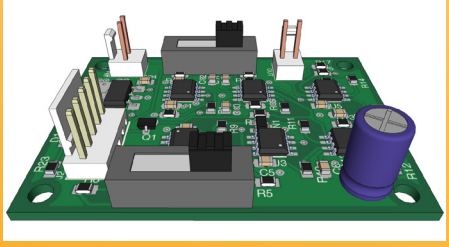


Fig. 15 - Transmissometer surface board.

Fig. 16 - The main board resting on the microcontroller adaptor board.

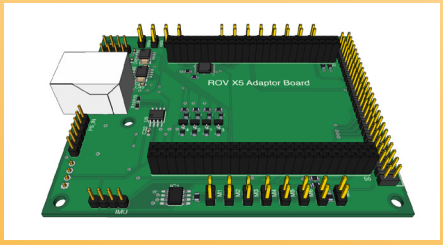


Fig. 17 - The microcontroller adaptor board.

Electronics

The electronics system of the Edison Submersibles ROV *Model N* is designed to provide complete user control over all of the ROV subsystems, including thrusters, video cameras, pneumatic actuators, and leveling system. The electronics system includes the main microcontroller board, microcontroller adaptor board, application board, power board, and interface adaptor board.

The main microcontroller board is the central point of communication and control for the subsea electronics system. The board is a modified discovery board from STMicroelectronics. The processor used is a STM32-F4 that utilizes a Cortex M4 series ARM processor. This processor is responsible for sending and receiving data from the tether, parsing data from sensors, interpreting and calculating the necessary thruster commands, controlling the pneumatic actuators, managing the video multiplexing, and activating the turning motors with PWM. The tether communication protocol used is RS-485. It offers increased range over RS-232 and provides noise immunity due to the differential pairs used. The internal communications for subsystem to subsystem communication is I2C. The IMU and the magnetometer act as slave devices on the I2C bus, and will transmit the necessary sensor data when requested to by the main processor. This sensor data can include internal enclosure temperature, 24V bus voltage and current, and IMU positioning data from an accelerometer, gyroscope, and magnetometer. Thruster speeds and directions from the user input device (XBox 360 controller) are interpreted by the microcontroller and then are sent out on the motor bus.

The microcontroller adaptor board acts as a shield for the main board and a connection to the application board. The eight thrusters receive their instructions from this board. All input and output signals are routed to GPIO pins on the main board through the adaptor board. The RS-485 communication block resides here, and the two 2-to-1 video multiplexers are located here as well. Four Darlington pairs serve as water alert sensors that alert the pilot to water in the electronics box if there is an increase in conductivity between any of the sets of two leads positioned as suspect leaking points in the otherwise dry chamber.

The application board contains system components pertinent only to this year's mission theme of Ocean Observing Systems. It receives 12 V for operating the turning motors and 3.3 V for logic from the power board. The application board receives the two motor select signals, direction signal, and PWM from the main board through a header connecting to the adaptor board. The select lines operate a 4-to-1 multiplexer that sends motor enable signals to each of the half-bridge motor drivers for the high-side voltage. A common low-side driver receives the PWM signal. The separation of the high-side and low-side voltage to the connected motors allows for direction control over PWM with just five half-bridge motor drivers.

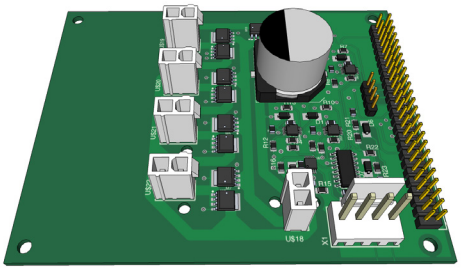


Fig. 18 - The application board.

The power breakout board helps to simplify the internal wiring and +48V power distribution to the main board and each of the DC/DC modules. This board has one input connector for the +48V wiring from the tether and fourteen individually fused outputs. Linear regulators create a +12V, +5V, and +3.3V output power line from the +48V provided over the tether, each of which is temperature-monitored. The connection to the microcontroller is voltage and current regulated. Each of the fast-blow fuses is connected to an alert signal that allows for rapid identification of any blown fuse on the power board. The first output of the device send power to the microcontroller adaptor board to power the thrusters. A second output is used for a +12V, +5V, and +3.3V connection for the main microcontroller board. A third output sends +12V to the solenoid bank, cameras, and IMU. Another output sends data about any alerts about voltage, current, temperature, or blown fuses that have been raised. The remaining three connectors are left open for debugging or future development. Each channel voltage also has a LED indicator to help to quickly determine the status of the fuse.

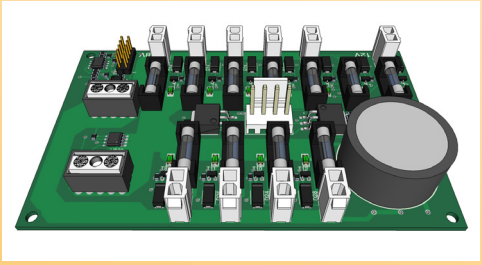


Fig. 19 - The power board.

The interface adapter board acts as the topside interface between the RS-485 and video signals that come from the ROV and the computer and video monitors. It is responsible for converting the RS-485 data into USB, and it breaks out the multiplexed video lines with simple RCA video connectors.

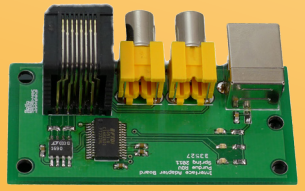


Fig. 20 - Interface adapter board.

Software

The control system of ROV *Model N* is meant to give the user an easy and intuitive way of controlling the vehicle. The control system has two distinct parts; the on-board software and the base station software. The base station is the brains of the operation. It takes user

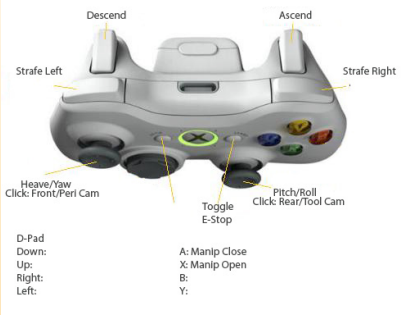


Fig. 21 - Xbox 360 controller mapping.

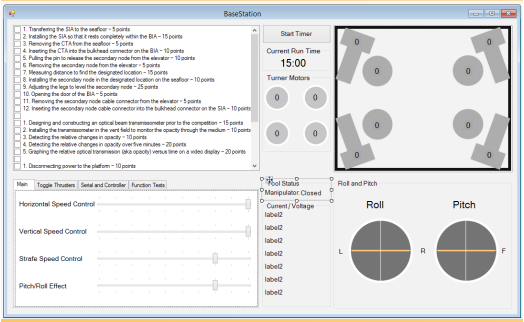


Fig. 22 - Graphical User Interface displaying relevant sensor data and thrust vectors.

input in the form of an Xbox 360 controller and computes thruster vectors for the vehicle's movement, as well as the position of the pneumatic systems and which video feeds are currently displayed. The base station was written in C++ and designed to communicate at a high baud rate to reduce lag in the movement of the vehicle. Using readily available libraries for the game pad controller, we were able to easily interface the controller with our software. It also displays relevant sensor data and thrust vectors to the pilot via a graphical user interface, as can be seen in the GUI screenshot. The base station software is responsible for core tasks such as communication with the vehicle processor, interpreting sensor data, controlling pneumatic actions and movement of the vehicle.

The on-board vehicle software is written in embedded C. The main ARM processor code is compiled using the EWARM libraries.

The main controller board talks to the integrated accelerometer allowing us to preform auto leveling actions as well as position holds. This allows much more fine grain control of the vehicle. Incorporated into our designs this year is a H20 sensing circuit. In the case of enclosure failure we can notify the pilot and preform an emergency shutdown of the system to prevent as much damage as possible. The main board utilizes the RS-485 serial protocol to communicate with the base station, receiving control values for the motors, pneumatics, and cameras on board the vehicle, as well as sending sensor data back up to the user. Any sensor data received is sent back to the base station and displayed on the graphical user interface, such as positional data and electrical voltages and currents. To control the solenoids, the processor takes the received instructions and assembles a byte of data to serve as a solenoid status byte. This byte is then output via the serial peripheral interface to the solenoid shift register, which then activates the appropriate solenoid. The motors are all controlled by a half duplex RS-485 bus. Each of the motors are individually assigned addresses and can be sent speed vectors and respond with information like motor temperature, current RPM and current power draw. The processor also uses general purpose digital outputs to create logical high and low switches to control which camera view is selected by the video switchers.

Reflections

Challenges

In the interest of reducing manufacturing costs, the Purdue IEEE ROV Team decided that an alternative to the round polycarbonate electronics enclosures that the team has become well-known for over the past two years must be found. After investigating a few possibilities, the team decided that a welded aluminum electronics box with a polycarbonate lid would be the most effective solution for its cost and for the reduced size of the electrical systems. Even though it was decided to put the lid on the underside of the box to see the mission area and to minimize water damage in case of a breach, leaking was still a primary concern for the team. The team tested several different thicknesses of polycarbonate lids both in Solidworks and in a pool to achieve a manageable dimension that would allow several bolts to hold it in position without flexing under the pressure at extreme depths. Dual weld lines were made to provide extra protections to the electronics box at the sides. All bolts and connection holes were sealed with 2-hour dry Loctite brand marine epoxy, the only reliable sealant that the team has ever found for a pressurized seal.

Troubleshooting Techniques

After initial population of the Application PCB, the motor driver channels were not functional. Troubleshooting began with verification of expected signals from the microcontroller lines. The problem was then isolated to the driver ICs. Despite all of the expected input signals, the expected output was not being produced. The next step was to verify every single component in the circuit for build errors. No problems were identified.

However, it was then discovered that the circuit would intermittently operate correctly. It was determined that a bad solder joint was the root cause of the issue. Sometimes the driver was not being enabled due to a low voltage lockout condition or possibly the enable line itself. After reflowing the solder on the whole board, the problem has been resolved.

The bad solder joint was likely due to the very small size of the particular motor driver IC that was employed. The overall process could have been improved by avoiding chips that are small where

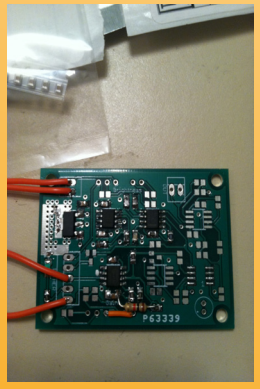


Fig. 23 - Electronics trouble-shooting.

there are larger chips with acceptable functionality. Further, population of multiple boards could have helped to more quickly isolate a construction issue from a design issue.

Lessons Learned and Skills Gained

The experience of building ROV *Model N* has taught both new and returning members many important skills. Most important of which is team work. No course can teach the ability for electrical and mechanical engineers to work together to make an end product that is up to professional standards. The Purdue team always treats its vehicles as test beds for learning. The electronics team learned how to code software with an ARM processor to control all of the vehicle's onboard systems.. There were a few new electrical members whose previous skill set ranged from designing some simple PCB's to never having done much soldering at all. The electronics team also refined its ability to design PCB's for professional printing, etching, and silk screening. The team struggled to design this year's vehicle with a shortage of mechanically focused team members. The team had to practice good project management skills to maintain forward progress among team members of vastly different skill levels. A returning challenge for the mechanical group this year was working with an outside machine shop. Because the team had enough funds to outsource some of its more complex pieces, an on-campus, University-affiliated machine shop was contracted. The mechanical group worked on refining its Solidworks designs to allow for cost-saving manufacturing. By working with a professional machinist to communicate clearly the engineers' intent, tolerances and other important criteria were reworked. The iterative and continuing nature of this process surprised everyone on the team because this year's team members had never gone through this process before.

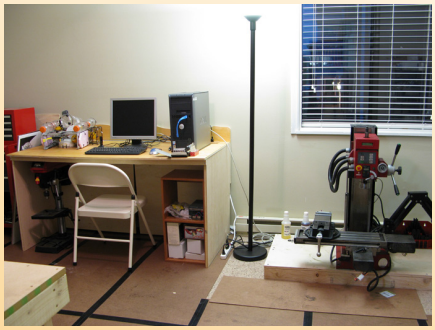


Fig. 24 - The team work-shop.

Future Improvements

There are a few improvements to consider on ROV *Model N*. While the vehicle is more than capable of completing its mission tasks, there a few luxuries that have been requested under this year's new pilot: depth holding, controllable pitch/roll stabilization, and faster vehicle response. The first two may be accomplished by software assistance. Better ROV responsiveness would be accomplished by reducing the weight of the vehicle, which may be accomplished by either reducing or replacing the heavy aluminum electronics box raised on thick stilts. Despite not having found a reliable, wide-angle digital camera

that can be feasibly waterproofed, the team would enjoy being able to implement more digital signal processing to aid in the propulsion of the vehicle and manipulation of objects underwater. Despite the tantalizing nature of these changes, the team must seriously consider the benefit of these features against the cost and implementation time that they would require.

Group Reflection

This is the fifth year for the Purdue IEEE ROV team. The team began in 2009 with four team members, a $500 budget, and a 5th place international finish. With the team size increasing to seven, nine, and seventeen and the budget increasing to $6,000, $14,700, and then $24,000 in 2012; the team rose in its final placement at the International Competition to 4th, 2nd, and 2nd just last year. Dedicated to earning 1st place this year, the Purdue IEEE ROV Team returns in force with eleven members and a $24,742 budget.

The team experienced its first transition of the captaincy and coped with changes in experience, management styles, and communication methods. With many of the past stalwarts of the team graduated and moved away, the team wrestled with its own inexperience designing and constructing ROV's. However, with the strong legacy of the team and mentoring from those past team members, the team came to terms with its abilities and still produced a superb ROV. The team has learned more collectively this year than any other year since its inception. With a year of growth, the returning members of the team next year are setting their sights higher and firmer.

Team Safety

The Purdue IEEE ROV team considered safety during both construction and operation of ROV *Model N*. Operation of all power tools required the use of OSHA approved glasses. The use of advanced tools owned by the team, such as a drill press and horizontal band saw, are safer than hand drills and hack saws when used properly because the work piece is not held by the user. Operation of the vehicle is made safe through many measures. Pneumatically, the ROV has a pressure regulator and an emergency quick release valve that can empty the system in a few seconds. Every part of the pneumatic system is rated to at least 689 kPa, well above the operating pressure of 270 kPa. Electronically, the ROV has a large array of onboard fuses and systems designed to not cause damage if communication is lost

with the surface. There is also a main inline fuse on the tether at the surface. Electrical system activations on the vehicle are slightly staggered over approximately two seconds to reduce high inrush at vehicle start up. Mechanically, there are as few sharp edges as possible throughout the ROV. There are a couple points on the ROV's tools that team members know to be aware of when in operation, such as the gripper and measuring rod pin release. In the team's five years of operation, only a few minor scrapes and no serious injures have occurred.

Fig. 25 - The Celebrate Science Indiana IEEE Booth

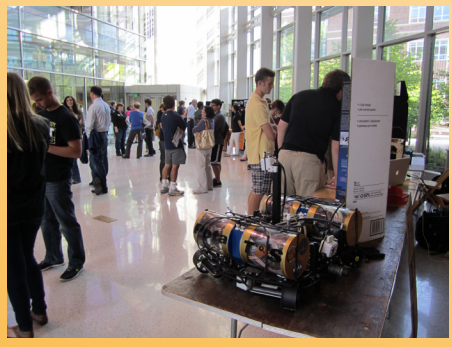


Fig. 26 - ROV PotaTOS and materials at PSEF NEXT Day.

Outreach

The Purdue IEEE ROV Team believes that educating grade school students about the benefits of STEM (Science, Technology, Engineering, and Mathematics) and the Marine Engineering industry is important. The students the team reaches are often unaware of the multiple opportunities these fields could have for them and how to access these benefits for themselves. The team has utilized its 2012 ROV, *PotaTOS*, throughout the year as an ambassador for STEM learning at events such as Hobart High School EXPO and on campus grade school visits such as PSEF NEXT Day. These events put the vehicle and the team in front of thousands of young students from Indiana. This year, the Purdue team worked with the Eli Whitney School in Chicago in an advisory capacity as part of its long-term commitment to a mentoring partnership. While this was mainly limited to email exchanges, the team hopes that this partnership can continue to grow stronger in the coming years. Finally, the team has made an attempt to educate as many Purdue students as possible about the Marine Engineering Industry. Most students on campus have no knowledge of what opportunities are available to them and presenting the vehicle at many public events is helping to give the field necessary exposure in the absence of other outlets.

Expense Report

Item	QTY If Applicable	Expense Cost USD / Unit	Donations Cost USD / Unit	Income Monetary
ROV Construction				
Waterjet Aluminum Frame		$450.00		
Assorted Machining (from Local Machine Shop)		$2,500.00		
Raw Aluminum, Stainless Steel, and Plastic Stock		$1,920.00		
Stainless Steel Bolts & Washers		$146.00		
Glues and Tef-Gel Corossion Protection		$65.00		
Bearings		$110.00		
Pneumatic Systems (Solenoids, Fittings, Tubing, Pistons)		$207.40		
Electronics Box O-Rings	2	$6.25		
Seabotix SBT166 Custom Thrusters (25% Discount)	8	$1,103.15	$365.51	
Board Cameras (from Supercircuits)	3	$93.41	$15.42	
Reused Tether Materials (Power Line, Ethernet, Pneumatic Tubes)		$370.00		
Binder-USA Waterproof Connections		$825.80		
Laptop to Drive ROV (Previously Owned)		$1,000.00		
Reused Video Displays (televisions)	2	$147.66		
Circuit Board Printing (from Advanced Circuits)		$161.77	$500.00	
Custom Electronics (Microcontroller, Fuses, Connectors, etc.)		$532.47		
Transmissometer Probes (from Vernier)		$84.00		
Turning Motors		$248.65		
Other/Travel				
Flying to Seattle (8 people)		$3,368.80		
Hotel in Seattle (5 Rooms)		$2,203.00		
Rental Minivan		$932.38		
SolidWorks	20		$1,300.00	
Competition Fee			$100.00	
Prototyping Costs (Electrical and Mechanical)		$650.00		
Poster		$150.00		
Presentation Materials		$90.00		
Power Tools and Accessories				
Set of Mission Props for Practice		$248.73		
Air Compressor		$242.99		
Donations				
Purdue Office of the Provost				$5,000.00
Indiana Space Grant Consortium				$7,500.00
Purdue Engineering Student Council				$4,800.00
Purdue IEEE Student Chapter				$4,640.41
Northrop Grumman				$1,000.00
Purdue College of Engineering				$1,500.00
Purdue Student Government				$301.65
Summary **Please Note: Quantity is taken in to account before values are summed**				
ROV Construction		$17,788.54	$2,970.36	
Other/Travel		$6,953.52	$26,100.00	
Donations				$24,742.06
TOTAL		$24,742.06	$29,070.36	$24,742.06
Total Remaining Balance	**$0.00**			

2013 Technical Report

Acknowledgements

Monetary Donations

and
Purdue Student Government

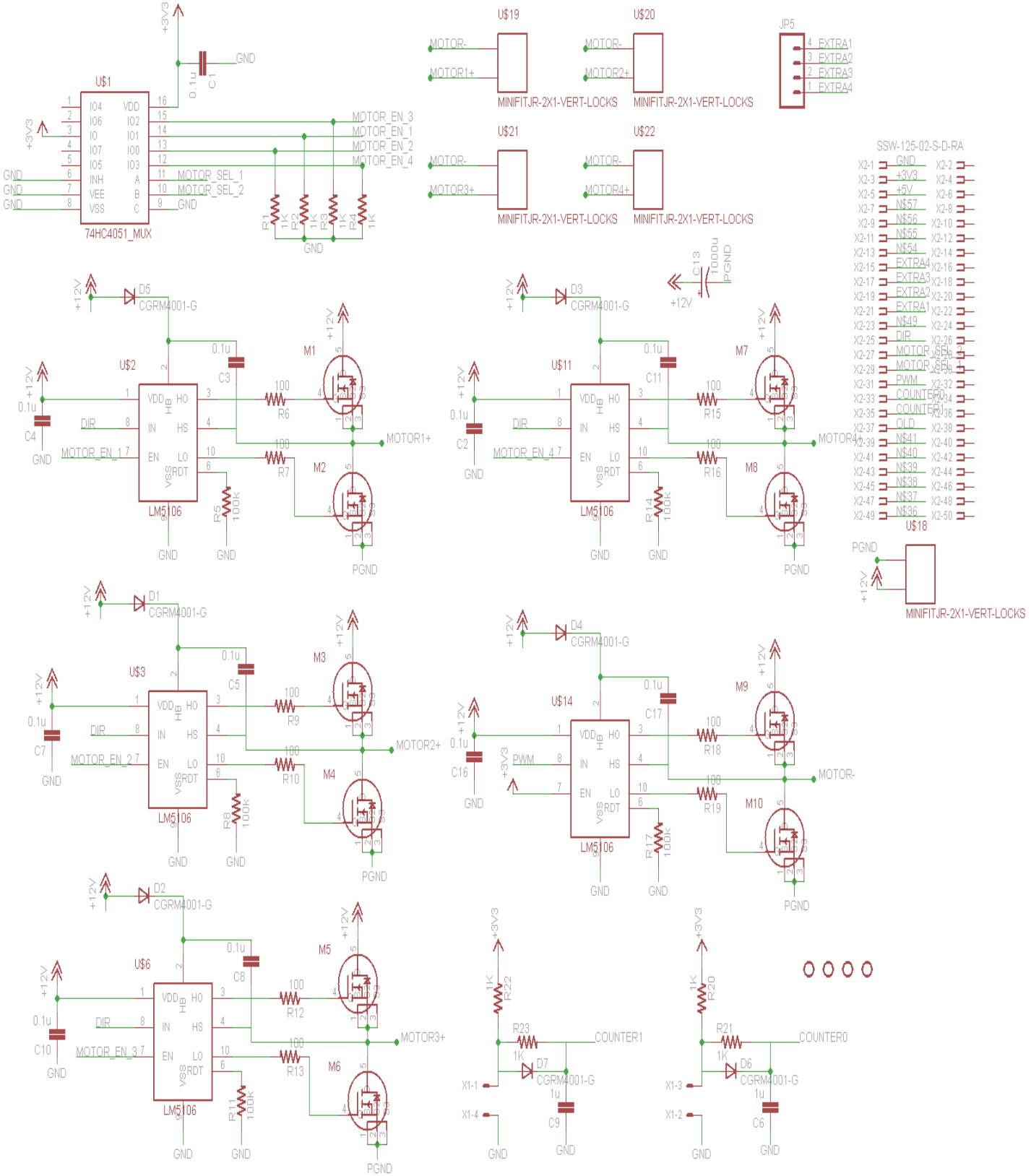
Discounts or Non-Monetary Donations

Edison Submersibles would also like to thank:
All the volunteer judges of the MATE Competition
MATE Center for providing us with this opportunity
Shedd Aquarium for hosting the Midwest Regional
Veteran members Clayton Kleppinger, Clement Lan, and Rob Swanson for mentoring new electronics members
Lawrence Goldstein for mentoring our new mechanical members
Seth Baklor and Spencer Julian for aiding the new captain
Purdue Research Machining Services for being such a great machining resource
Purdue School of Electrical and Computer Engineering for offering guidance in fund raising
Dr. Mark Johnson for use of the Computer Engineering Senior Design Lab
Eta Kappa Nu, Beta Chapter for use of the HKN Lab
Ross Howard for his facilitation of purchases
Friends and Family for putting up with us during the build season

2013 Technical Report

Appendix A - Electrical System Block Diagrams

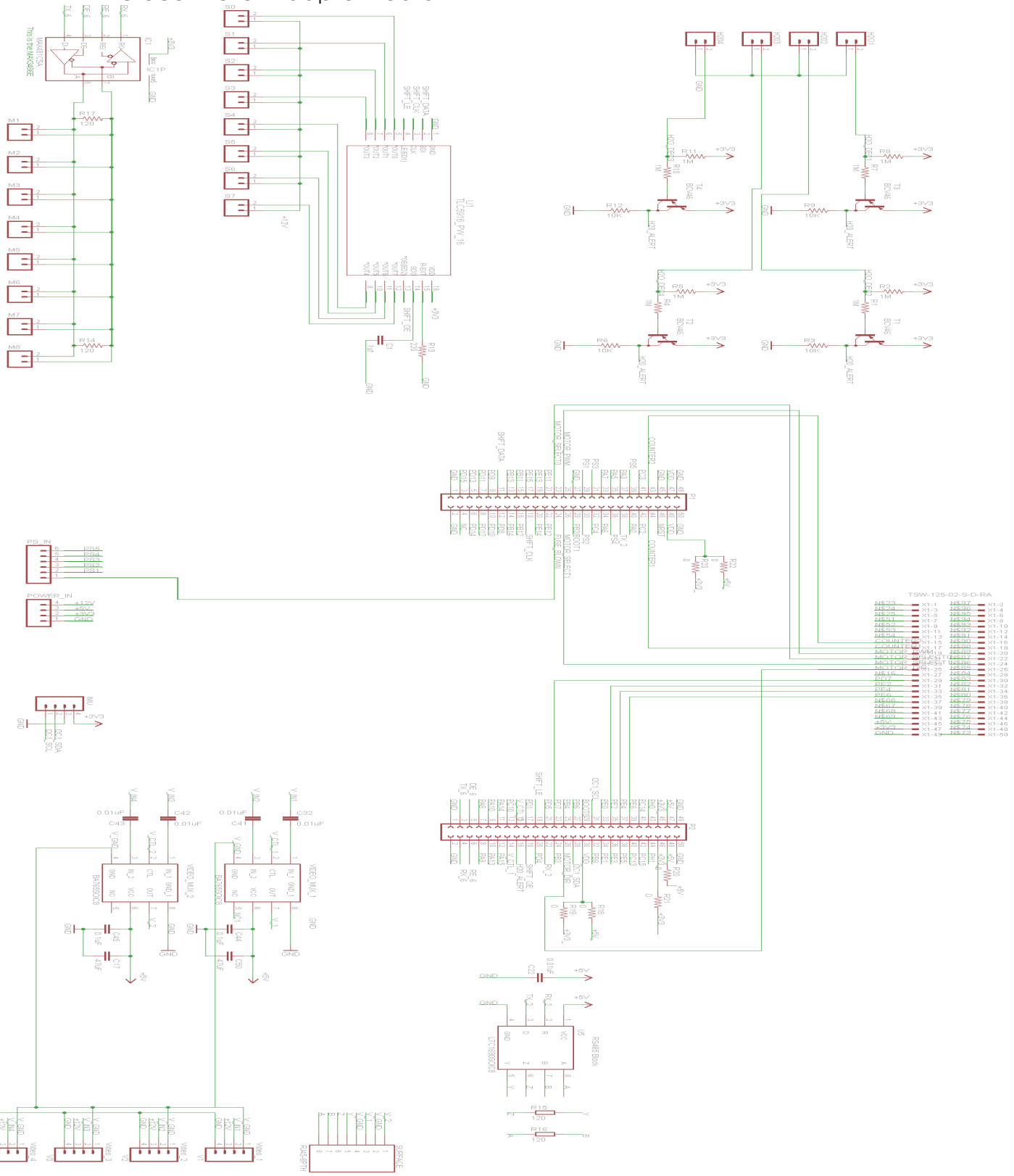
Application Board



2013 Technical Report

Appendix A - Electrical System Block Diagrams

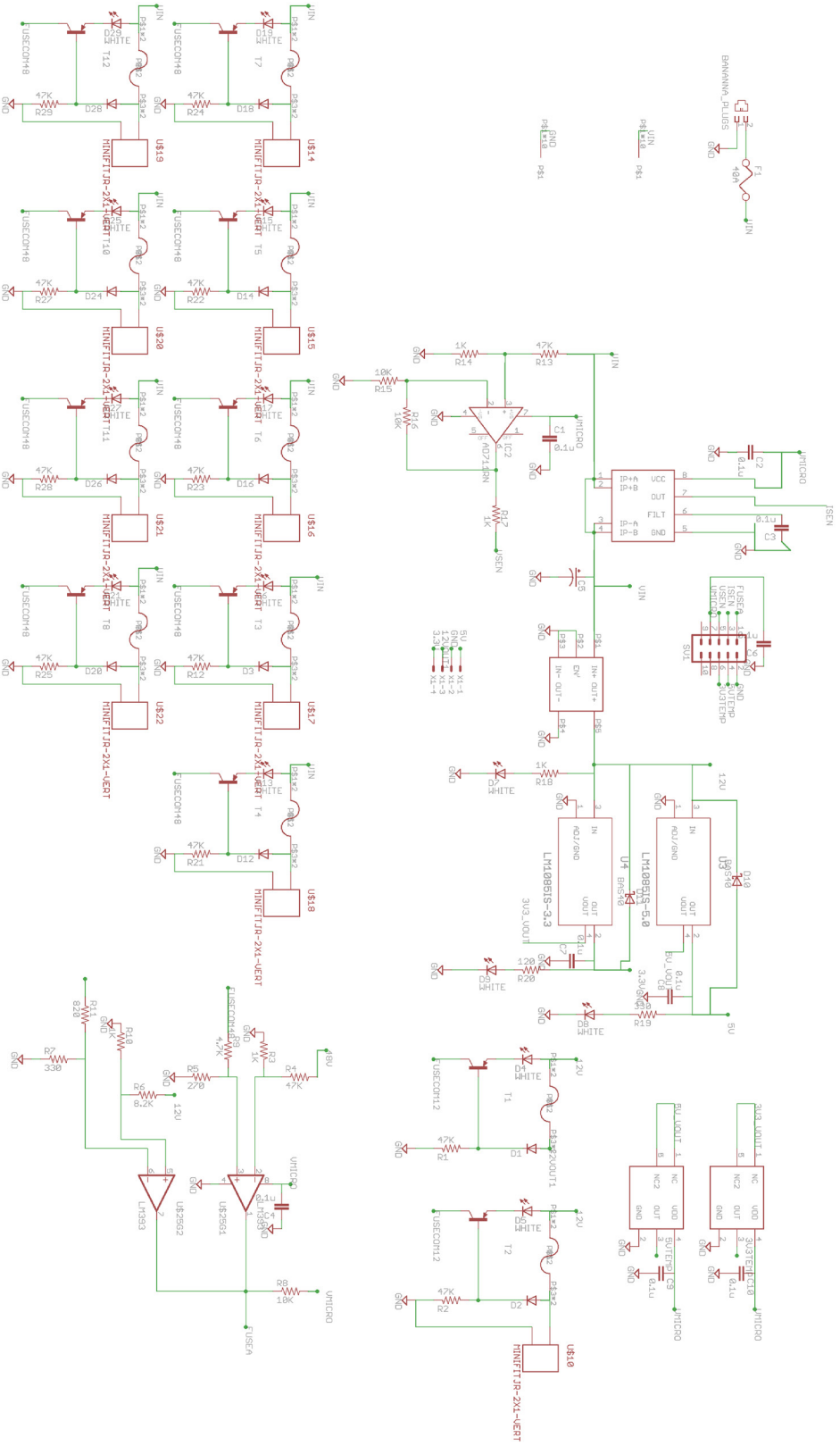
Microcontroller Adapter Board



2013 Technical Report

Appendix A - Electrical System Block Diagrams

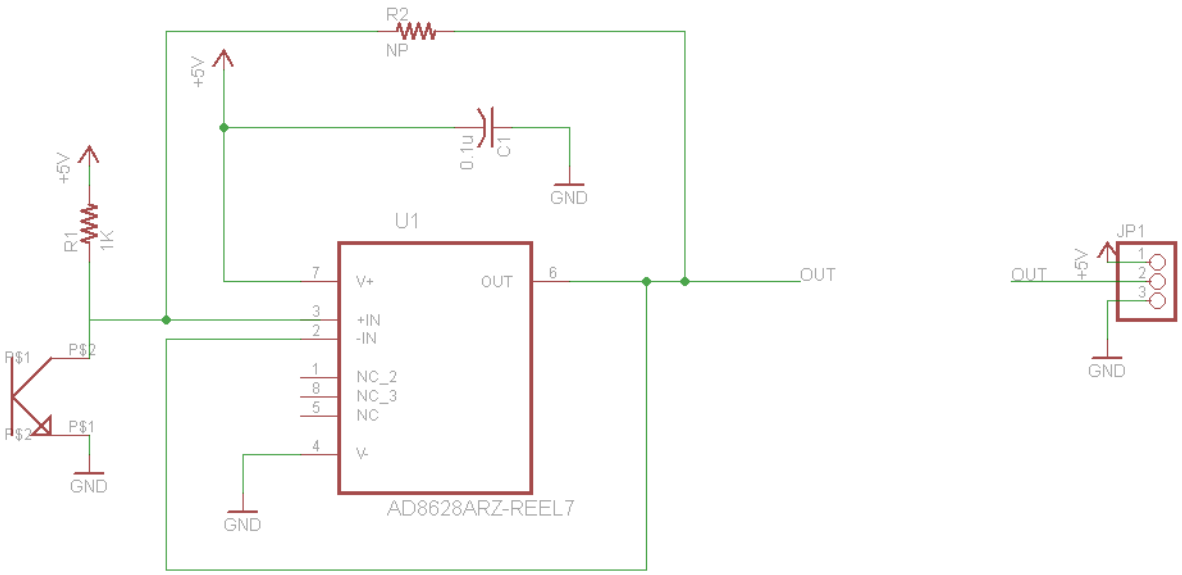
Power board



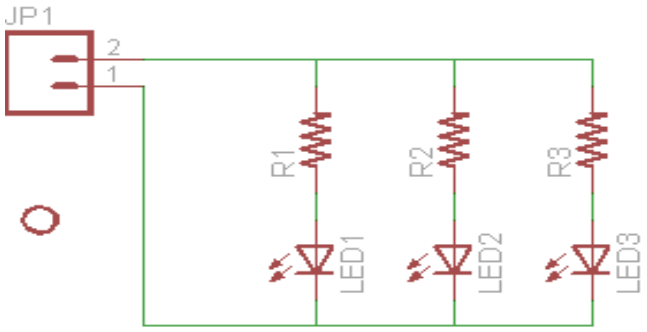
2013 Technical Report

Appendix A - Electrical System Block Diagrams

Transmissometer RX Board



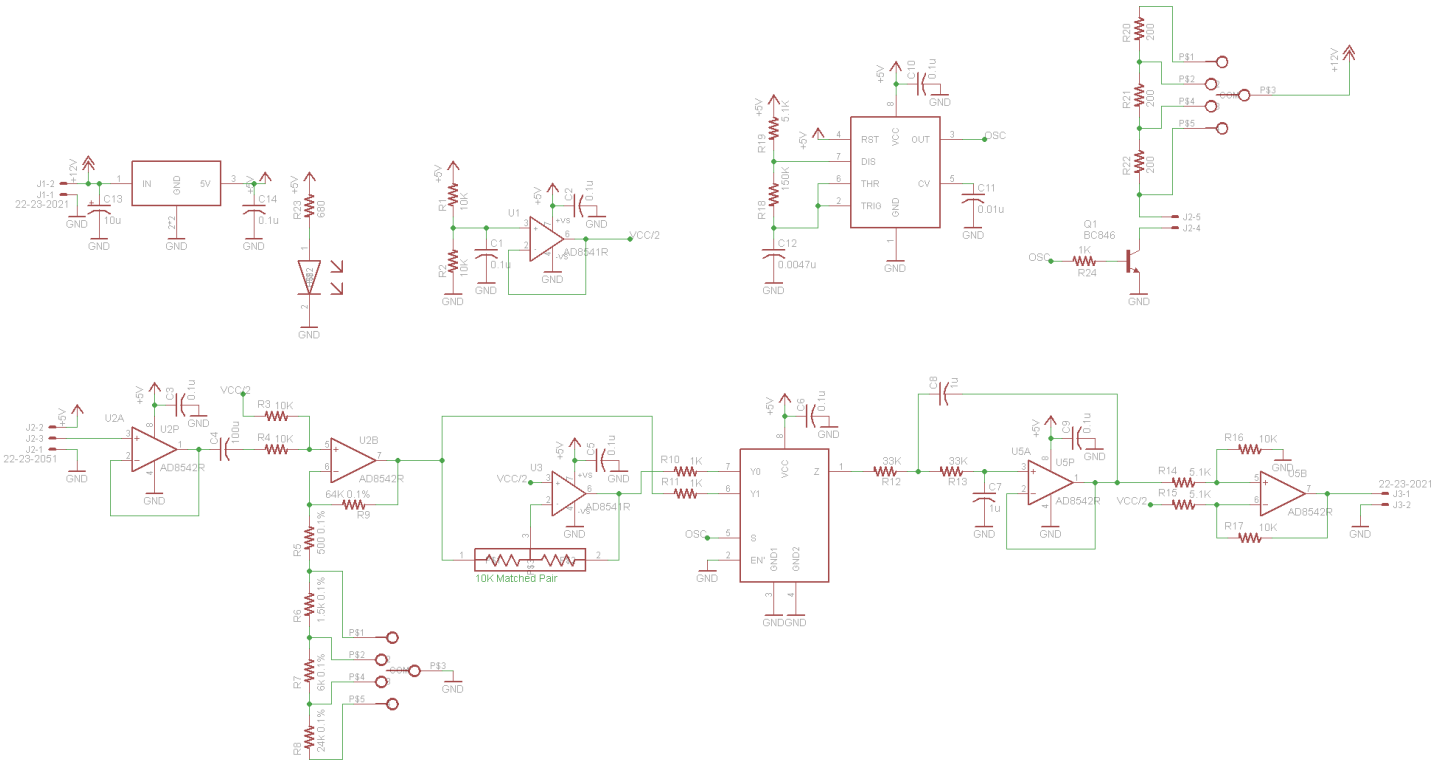
Transmissometer TX Board



2013 Technical Report

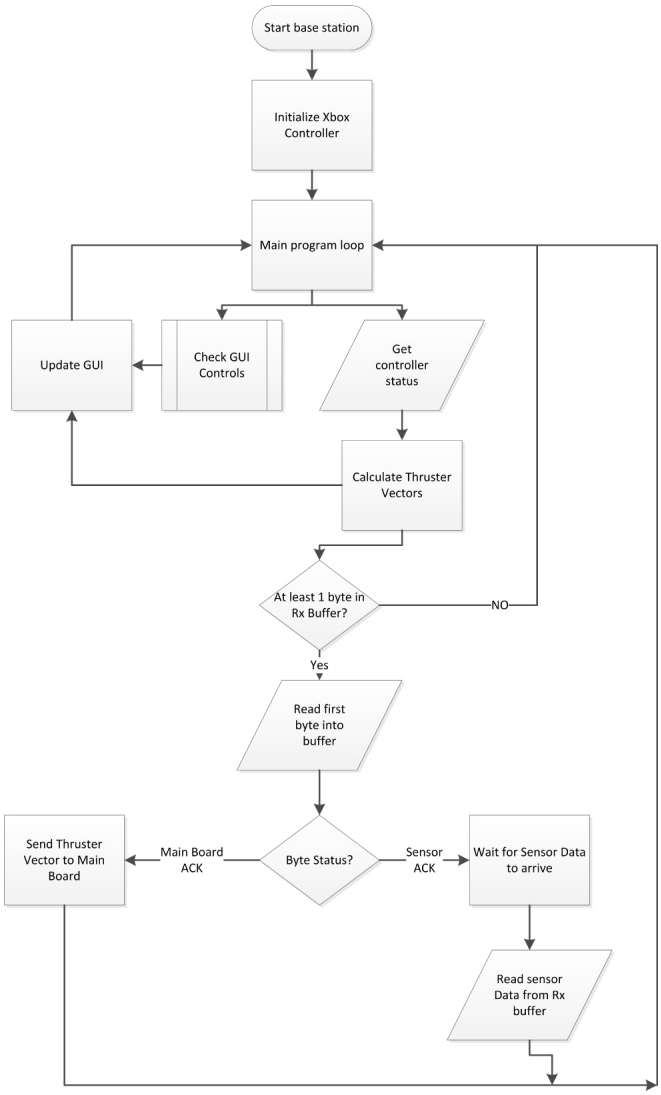
Appendix A - Electrical System Block Diagrams

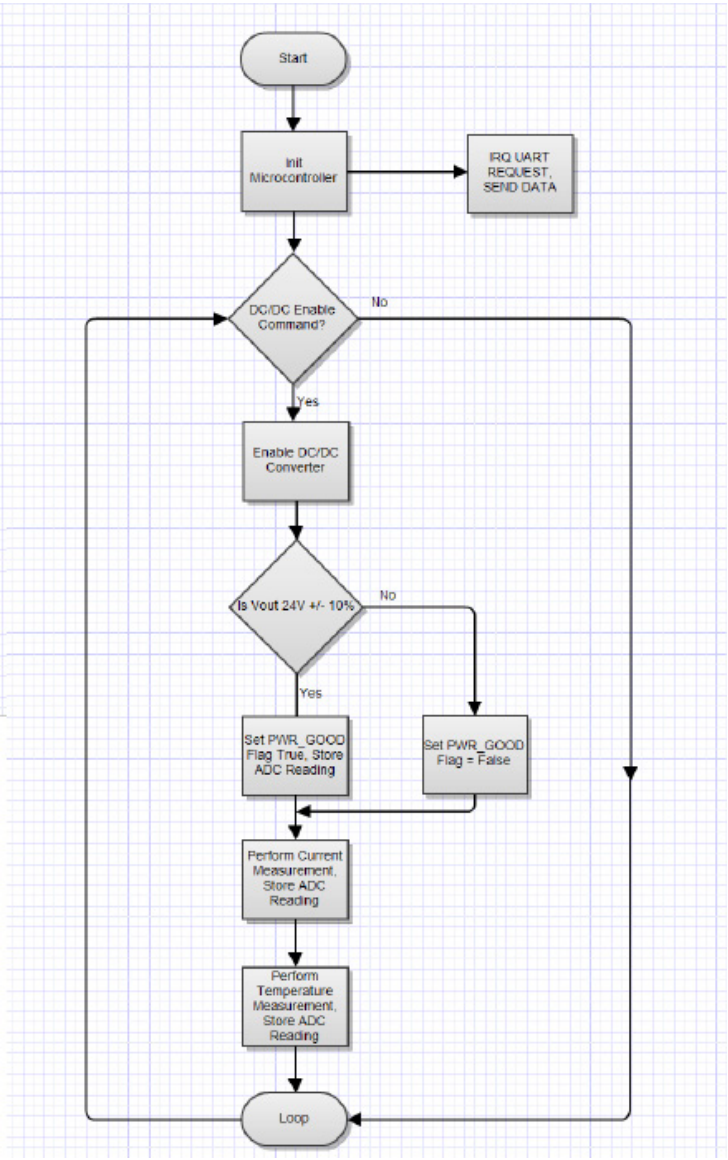
Transmissometer Surface Board



Appendix B - Software Flowcharts

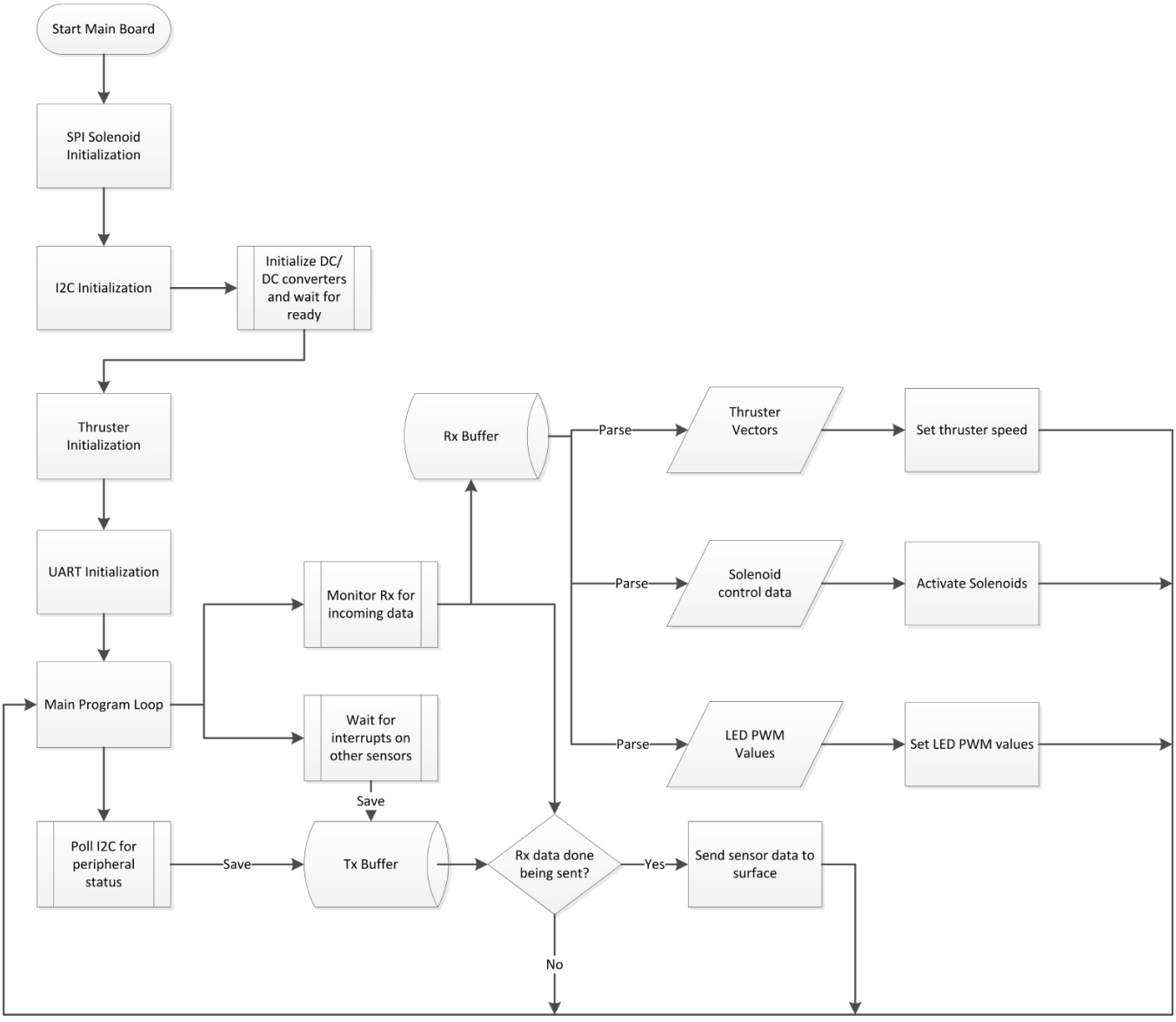
Basestation & DC/DC





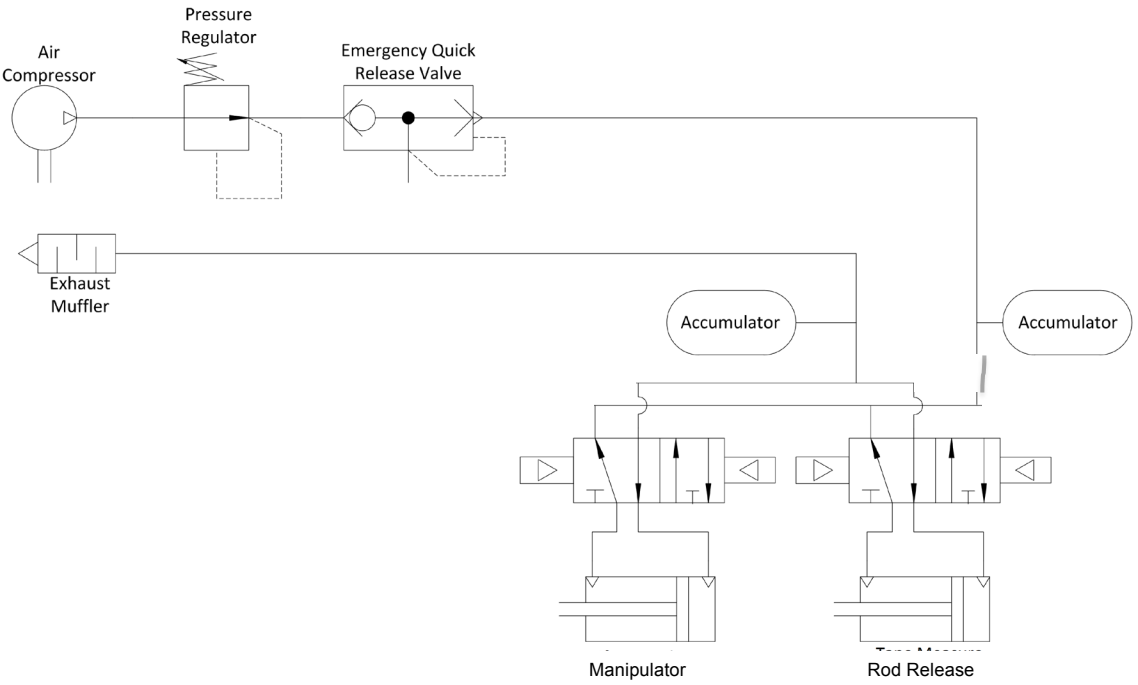
Appendix B - Software Flowcharts

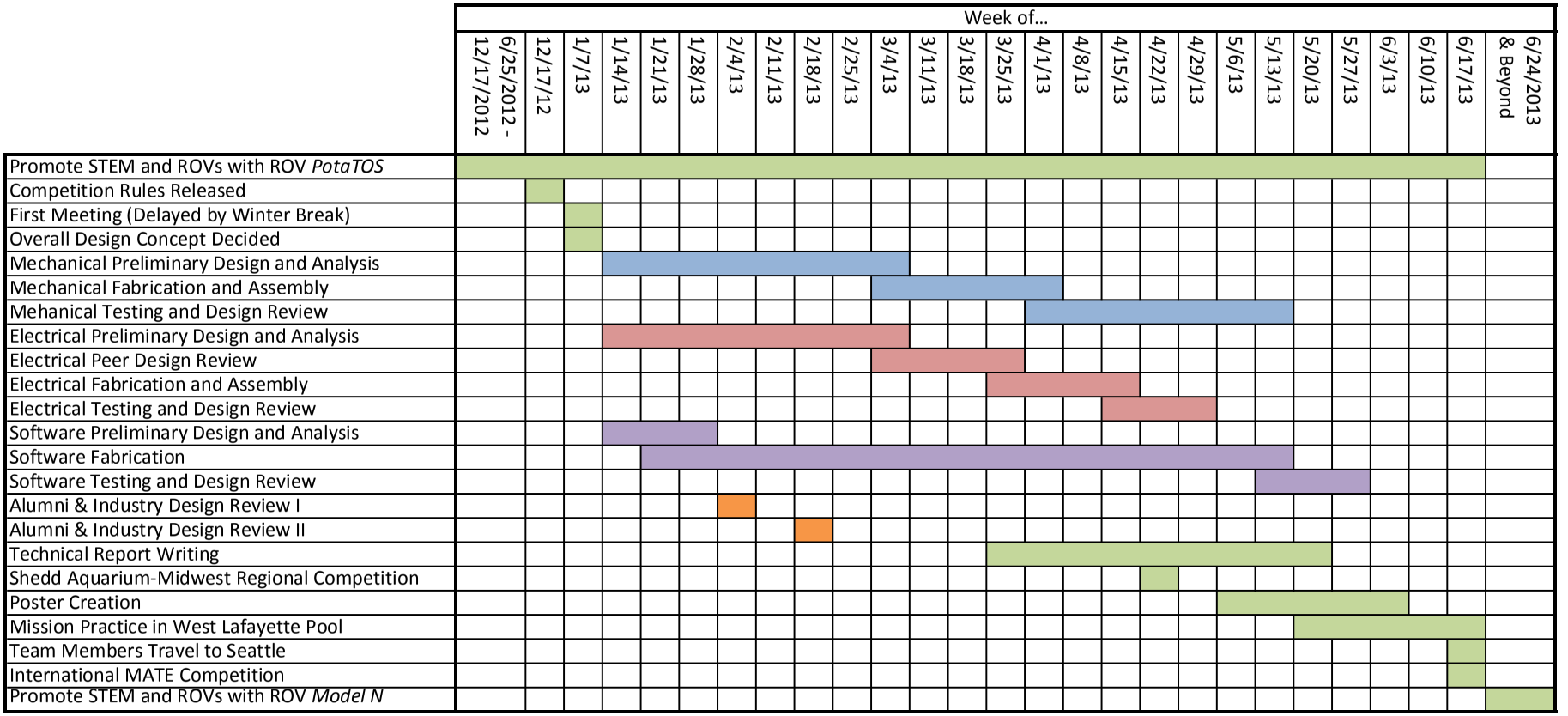
Main Board



Appendix C - Pneumatic System Diagram

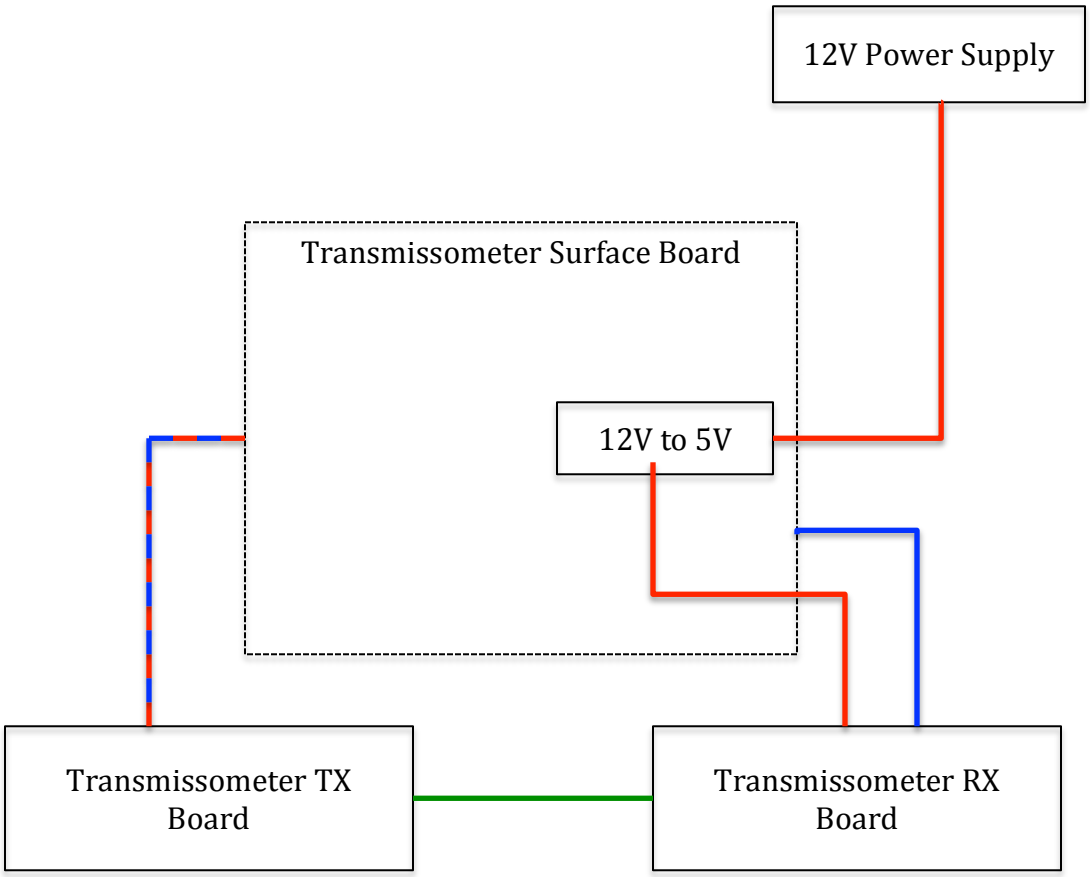
Pneumatic System



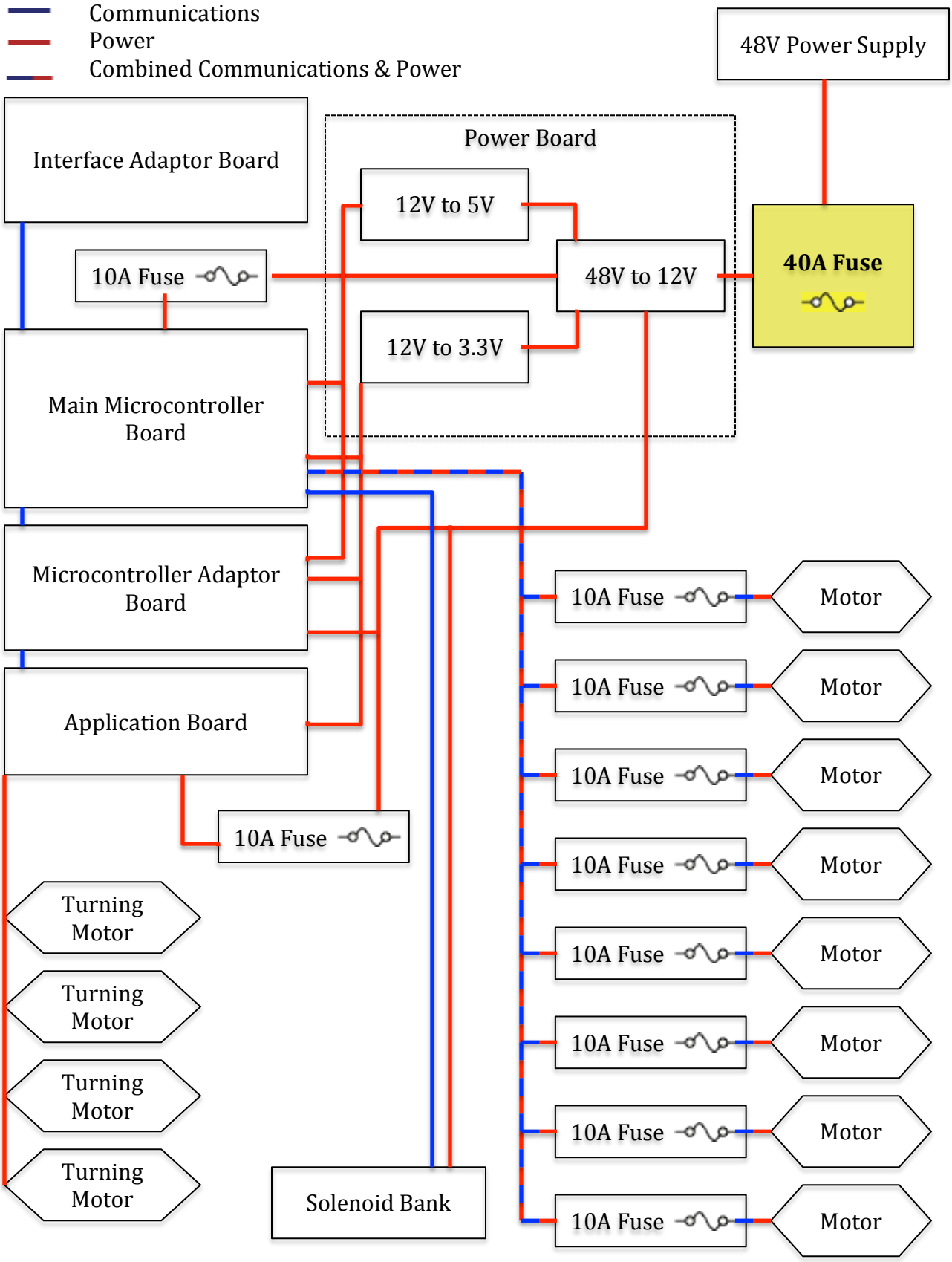

Week of...
6/25/2012 - 12/17/2012
12/17/12
1/7/13
1/14/13
1/21/13
1/28/13
2/4/13
2/11/13
2/18/13
2/25/13
3/4/13
3/11/13
3/18/13
3/25/13
4/1/13
4/8/13
4/15/13
4/22/13
4/29/13
5/6/13
5/13/13
5/20/13
5/27/13
6/3/13
6/10/13
6/17/13
6/24/2013 & Beyond
Promote STEM and ROVs with ROV PotaTOS
Competition Rules Released
First Meeting (Delayed by Winter Break)
Overall Design Concept Decided
Mechanical Preliminary Design and Analysis
Mechanical Fabrication and Assembly
Mehanical Testing and Design Review
Electrical Preliminary Design and Analysis
Electrical Peer Design Review
Electrical Fabrication and Assembly
Electrical Testing and Design Review
Software Preliminary Design and Analysis
Software Fabrication
Software Testing and Design Review
Alumni & Industry Design Review I
Alumni & Industry Design Review II
Technical Report Writing
Shedd Aquarium-Midwest Regional Competition
Poster Creation
Mission Practice in West Lafayette Pool
Team Members Travel to Seattle
International MATE Competition
Promote STEM and ROVs with ROV Model N

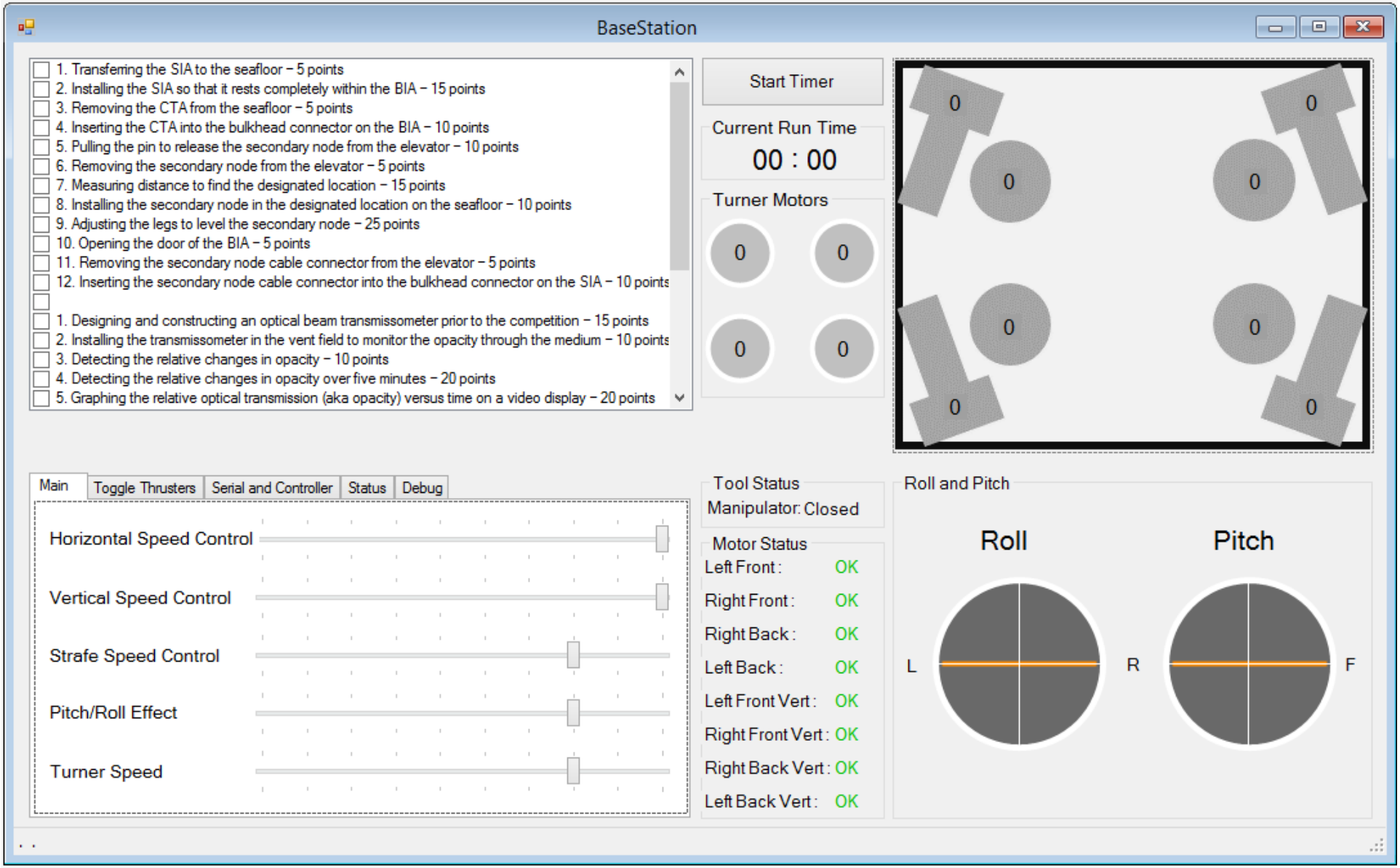
Transmissometer Electronics

Communications
Power
Combined Communications & Power
Light

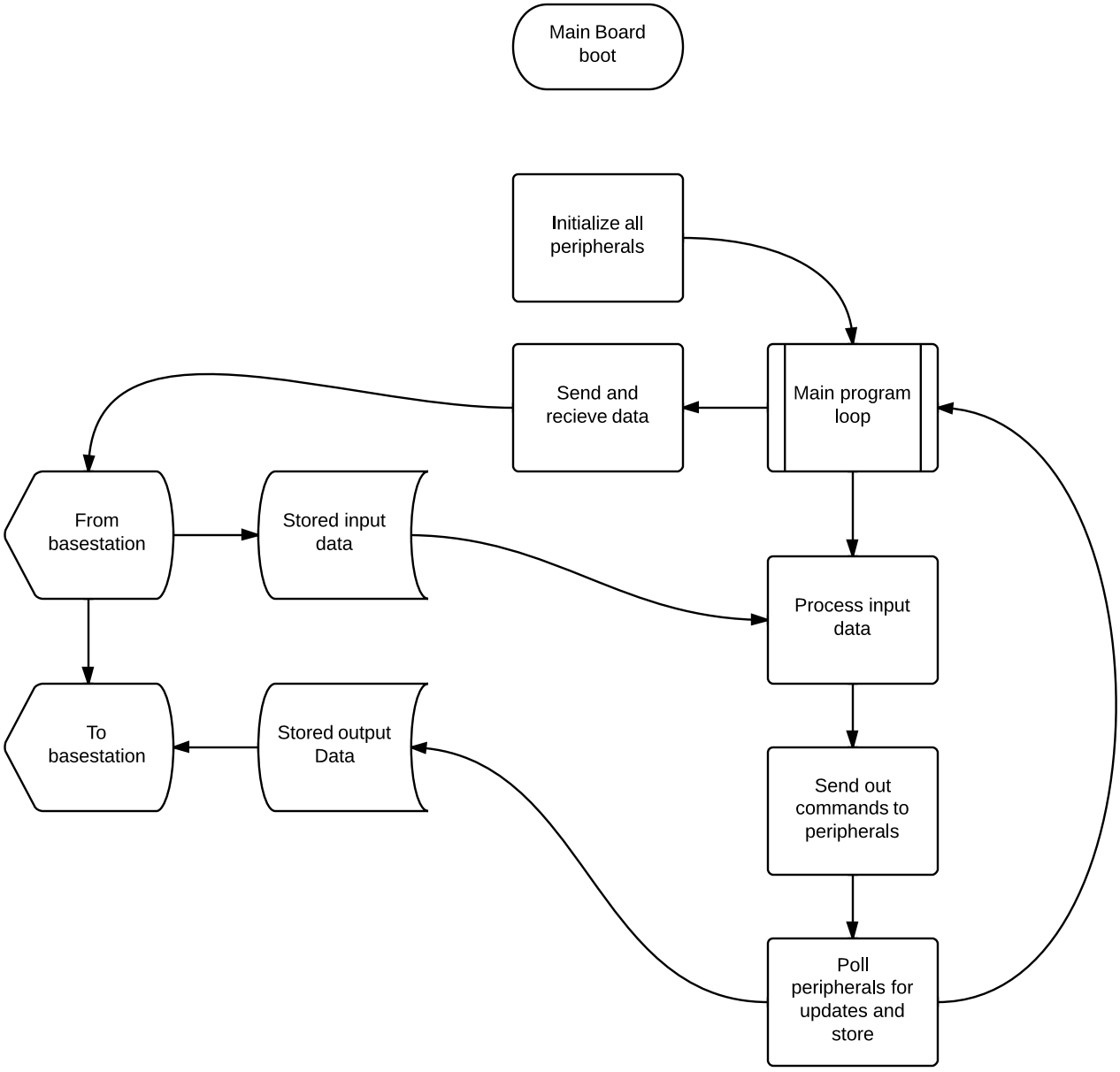


Electronics

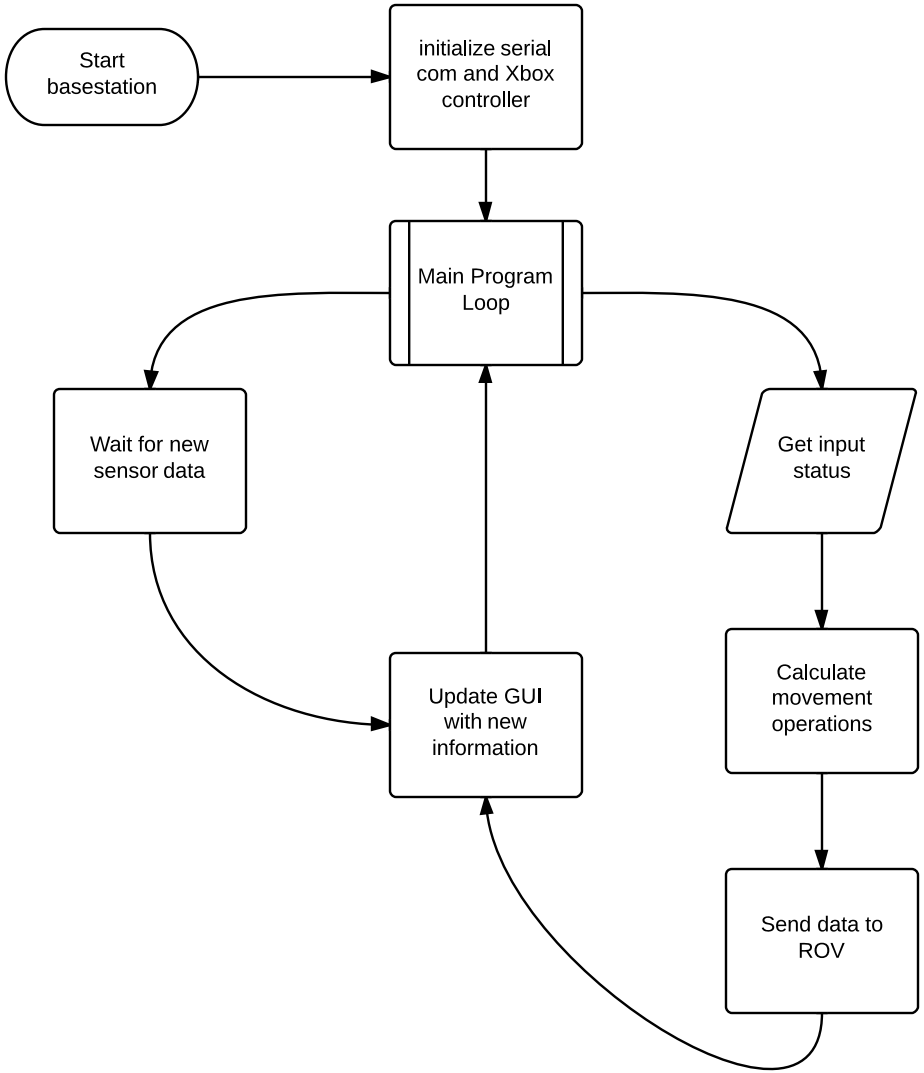



BaseStation
1. Transferring the SIA to the seafloor – 5 points
2. Installing the SIA so that it rests completely within the BIA – 15 points
3. Removing the CTA from the seafloor – 5 points
4. Inserting the CTA into the bulkhead connector on the BIA – 10 points
5. Pulling the pin to release the secondary node from the elevator – 10 points
6. Removing the secondary node from the elevator – 5 points
7. Measuring distance to find the designated location – 15 points
8. Installing the secondary node in the designated location on the seafloor – 10 points
9. Adjusting the legs to level the secondary node – 25 points
10. Opening the door of the BIA – 5 points
11. Removing the secondary node cable connector from the elevator – 5 points
12. Inserting the secondary node cable connector into the bulkhead connector on the SIA – 10 points
1. Designing and constructing an optical beam transmissometer prior to the competition – 15 points
2. Installing the transmissometer in the vent field to monitor the opacity through the medium – 10 points
3. Detecting the relative changes in opacity – 10 points
4. Detecting the relative changes in opacity over five minutes – 20 points
5. Graphing the relative optical transmission (aka opacity) versus time on a video display – 20 points
Start Timer
Current Run Time
00 : 00
Turner Motors
0
0
0
0
0
0
0
0
0
0
0
0
Main
Toggle Thrusters
Serial and Controller
Status
Debug
Horizontal Speed Control
Vertical Speed Control
Strafe Speed Control
Pitch/Roll Effect
Turner Speed
Tool Status
Manipulator: Closed
Motor Status
Left Front : OK
Right Front : OK
Right Back : OK
Left Back : OK
Left Front Vert : OK
Right Front Vert : OK
Right Back Vert : OK
Left Back Vert : OK
Roll and Pitch
Roll
Pitch
L
R
F

Main Board


Main Board boot
Initialize all peripherals
Main program loop
Send and recieve data
From basestation
Stored input data
Process input data
To basestation
Stored output Data
Send out commands to peripherals
Poll peripherals for updates and store

Basestation



```
1  ------------------------------------------------ Abstract.h
…  ------------------------------------------------
2  #ifndef ABSTRACT_H_
3  #define ABSTRACT_H_
4
5  #include "functions.h"
6
7  //Debug Leds - PD12, PD13, PD14, PD15
8  #define LED_0_PIN                    GPIO_Pin_12
9  #define LED_0_PORT                   GPIOD
10 #define LED_0_CLK                    RCC_AHB1Periph_GPIOD
11
12 #define LED_1_PIN                    GPIO_Pin_13
13 #define LED_1_PORT                   GPIOD
14 #define LED_1_CLK                    RCC_AHB1Periph_GPIOD
15
16 #define LED_2_PIN                    GPIO_Pin_14
17 #define LED_2_PORT                   GPIOD
18 #define LED_2_CLK                    RCC_AHB1Periph_GPIOD
19
20 #define LED_3_PIN                    GPIO_Pin_15
21 #define LED_3_PORT                   GPIOD
22 #define LED_3_CLK                    RCC_AHB1Periph_GPIOD
23
24
25 //4x GPIO - for water detection  - Input floating: PD0, PC12, PC11, PC10
26 #define WATER_0_PIN                  GPIO_Pin_0
27 #define WATER_0_PORT                 GPIOD
28 #define WATER_0_CLK                  RCC_AHB1Periph_GPIOD
29
30 #define WATER_1_PIN                  GPIO_Pin_12
31 #define WATER_1_PORT                 GPIOC
32 #define WATER_1_CLK                  RCC_AHB1Periph_GPIOC
33
34 #define WATER_2_PIN                  GPIO_Pin_11
35 #define WATER_2_PORT                 GPIOC
36 #define WATER_2_CLK                  RCC_AHB1Periph_GPIOC
37
38 #define WATER_3_PIN                  GPIO_Pin_10
```

```
39 #define WATER_3_PORT                GPIOC
40 #define WATER_3_CLK                 RCC_AHB1Periph_GPIOC
41
42
43 //2x GPIO - Turner selection - Output push pull: PE8, PE7
44 #define TURNER_SEL_0_PIN            GPIO_Pin_8
45 #define TURNER_SEL_0_PORT           GPIOE
46 #define TURNER_SEL_0_CLK            RCC_AHB1Periph_GPIOE
47
48 #define TURNER_SEL_1_PIN            GPIO_Pin_7
49 #define TURNER_SEL_1_PORT           GPIOE
50 #define TURNER_SEL_1_CLK            RCC_AHB1Periph_GPIOE
51
52
53 //MUX CONTROL - 2x GPIO - for camera selection PC11, PC12
54 #define MUX_0_PIN                   GPIO_Pin_11
55 #define MUX_0_PORT                  GPIOC
56 #define MUX_0_CLK                   RCC_AHB1Periph_GPIOC
57
58 #define MUX_1_PIN                   GPIO_Pin_12
59 #define MUX_1_PORT                  GPIOC
60 #define MUX_1_CLK                   RCC_AHB1Periph_GPIOC
61
62
63 //1x GPIO for fuse blown - Input floating:PE10
64 #define FUSE_DECT_PIN               GPIO_Pin_10
65 #define FUSE_DECT_PORT              GPIOE
66 #define FUSE_DECT_CLK               RCC_AHB1Periph_GPIOE
67
68
69 //PB5 for Turner  DIR
70 #define TURNER_DIR_PIN              GPIO_Pin_5
71 #define TURNER_DIR_PORT             GPIOB
72 #define TURNER_DIR_CLK              RCC_AHB1Periph_GPIOB
73
74
75 //1x PWM - Turner motors - TIM1: PE9, PE11, PE13, PE14
76 //UNIVERSAL
77 #define TURNER_PWM_TIM_RCC          RCC_APB2Periph_TIM1
```

```
78  #define TURNER_PWM_TIM_AF           GPIO_AF_TIM1
79  #define TURNER_PWM_TIM              TIM1
80
81  //TURNER1
82  #define TURNER_PWM_1_PIN            GPIO_Pin_9
83  #define TURNER_PWM_1_PORT           GPIOE
84  #define TURNER_PWM_1_CLK            RCC_AHB1Periph_GPIOE
85  #define TURNER_PWM_1_CH             1
86  #define TURNER_PWM_1_SOURCE         GPIO_PinSource9
87  //TURNER2
88  #define TURNER_PWM_2_PIN            GPIO_Pin_11
89  #define TURNER_PWM_2_PORT           GPIOE
90  #define TURNER_PWM_2_CLK            RCC_AHB1Periph_GPIOE
91  #define TURNER_PWM_2_CH             2
92  #define TURNER_PWM_2_SOURCE         GPIO_PinSource11
93
94  //TURNER3
95  #define TURNER_PWM_3_PIN            GPIO_Pin_13
96  #define TURNER_PWM_3_PORT           GPIOE
97  #define TURNER_PWM_3_CLK            RCC_AHB1Periph_GPIOE
98  #define TURNER_PWM_3_CH             3
99  #define TURNER_PWM_3_SOURCE         GPIO_PinSource13
100
101 //TURNER4
102 #define TURNER_PWM_4_PIN            GPIO_Pin_14
103 #define TURNER_PWM_4_PORT           GPIOE
104 #define TURNER_PWM_4_CLK            RCC_AHB1Periph_GPIOE
105 #define TURNER_PWM_4_CH             4
106 #define TURNER_PWM_4_SOURCE         GPIO_PinSource14
107
108
109 //Usart2: PD6-RX, PA2-TX
110 #define USART2_TIM_RCC              RCC_APB1Periph_USART2
111 #define USART2_TIM_AF               GPIO_AF_USART2
112
113 #define USART2_RX_PIN               GPIO_Pin_6
114 #define USART2_RX_PORT              GPIOD
115 #define USART2_RX_CLK               RCC_AHB1Periph_GPIOD
116
```

```
117 #define USART2_TX_PIN                GPIO_Pin_2
118 #define USART2_TX_PORT               GPIOA
119 #define USART2_TX_CLK                RCC_AHB1Periph_GPIOA
120
121
122 //Usart6: PC7-RX, PC6-TX - Motor bus also needs PC8, PC9
123 #define USART6_ENABLE_PIN            GPIO_Pin_8
124 #define USART6_ENABLE_PORT           GPIOC
125 #define USART6_ENABLE_CLK            RCC_AHB1Periph_GPIOC
126
127 #define USART6_DISABLE_PIN           GPIO_Pin_9
128 #define USART6_DISABLE_PORT          GPIOC
129 #define USART6_DISABLE_CLK           RCC_AHB1Periph_GPIOC
130
131 #define USART6_TIM_RCC               RCC_APB1Periph_USART2
132 #define USART6_TIM_AF                GPIO_AF_USART2
133
134 #define USART6_RX_PIN                GPIO_Pin_7
135 #define USART6_RX_PORT               GPIOC
136 #define USART6_RX_CLK                RCC_AHB1Periph_GPIOC
137
138 #define USART6_TX_PIN                GPIO_Pin_6
139 #define USART6_TX_PORT               GPIOC
140 #define USART6_TX_CLK                RCC_AHB1Periph_GPIOC
141
142
143 //4x Extra GPIO: PD7, PE6, PE4, PE2
144 #define GPIO_0_PIN                   GPIO_Pin_7
145 #define GPIO_0_PORT                  GPIOD
146 #define GPIO_0_CLK                   RCC_AHB1Periph_GPIOD
147
148 #define GPIO_1_PIN                   GPIO_Pin_6
149 #define GPIO_1_PORT                  GPIOE
150 #define GPIO_1_CLK                   RCC_AHB1Periph_GPIOE
151
152 #define GPIO_2_PIN                   GPIO_Pin_4
153 #define GPIO_2_PORT                  GPIOE
154 #define GPIO_2_CLK                   RCC_AHB1Periph_GPIOE
155
```

```
156 #define GPIO_3_PIN                  GPIO_Pin_2
157 #define GPIO_3_PORT                 GPIOE
158 #define GPIO_3_CLK                  RCC_AHB1Periph_GPIOE
159
160
161 //solenoid
162 //enable held low for operating - active low
163 //regclr held high for working, low for all zero
164 //serial- in for datas
165 //clock high then low for clock inbetween each data bit
166
167 #define SER_EN_PIN                      GPIO_Pin_2
168 #define SER_EN_PORT                     GPIOE
169 #define SER_EN_CLK                      RCC_AHB1Periph_GPIOE
170
171 #define SER_CLR_PIN                     GPIO_Pin_3
172 #define SER_CLR_PORT                    GPIOE
173 #define SER_CLR_CLK                     RCC_AHB1Periph_GPIOE
174
175 #define SER_IN_PIN                      GPIO_Pin_2
176 #define SER_IN_PORT                     GPIOA
177 #define SER_IN_CLK                      RCC_AHB1Periph_GPIOA
178
179 #define SER_CLK_PIN                     GPIO_Pin_3
180 #define SER_CLK_PORT                    GPIOA
181 #define SER_CLK_CLK                     RCC_AHB1Periph_GPIOA
182
183
184 #endif
185
186
187 ---------------------------------------------- Functions.h
  … ------------------------------------------------
188
189
190 #ifndef FUNCTIONS_H_
191 #define FUNCTIONS_H_
192
193 #include "stm32f4xx.h"
```

```
194 #include "stm32f4xx_conf.h"
195 #include <stm32f4xx_dma.h>
196 #include <stm32f4xx_gpio.h>
197 #include <stm32f4xx_i2c.h>
198 #include <stm32f4xx_it.h>
199 #include <stm32f4xx_rcc.h>
200 #include <stm32f4xx_syscfg.h>
201 #include <stm32f4xx_spi.h>
202 #include <stm32f4xx_tim.h>
203 #include <stm32f4xx_usart.h>
204
205 #include "abstract.h"
206
207 #include <stdio.h>
208
209 #define USARTBUFFSIZE   20
210
211 typedef struct{
212     uint8_t in;
213     uint8_t out;
214     uint8_t count;
215     uint8_t buff[USARTBUFFSIZE];
216 }FIFO_TypeDef;
217
218 void BufferInit(__IO FIFO_TypeDef *buffer);
219 ErrorStatus BufferPut(__IO FIFO_TypeDef *buffer, uint8_t ch);
220 ErrorStatus BufferGet(__IO FIFO_TypeDef *buffer, uint8_t *ch);
221 ErrorStatus BufferIsEmpty(__IO FIFO_TypeDef buffer);
222
223 extern volatile FIFO_TypeDef U2Rx, U2Tx;
224 extern volatile uint8_t flag2;
225 void Usart2Init(void);
226 void Usart2Put(uint8_t ch);
227 uint8_t Usart2Get(void);
228
229 extern volatile FIFO_TypeDef U6Rx, U6Tx;
230 extern volatile uint8_t flag6;
231 void Usart6Init(void);
232 void Usart6Put(uint8_t ch);
```

```
233 uint8_t Usart6Get(void);
234
235 #include "functions.h"
236
237 /*************************** X5 Code
    ******************************************/
238 void PinOutput(const uint16_t PIN, GPIO_TypeDef* PORT,const uint32_t
    CLK);
239
240 void PinInput(const uint16_t PIN, GPIO_TypeDef* PORT,const uint32_t CLK);
241
242 void Mux_Init();
243
244 void TurnerSelectInit();
245
246 void TurnerDirInit();
247
248 void WaterInit();
249
250 void FuseInit();
251
252 void TurnerPWMInit();
253
254 void NVIC_Config(void);
255
256 void LedInit();
257
258 void Usart2Init();
259
260 void Usart6Init();
261
262 void MUXControl(volatile uint8_t M0, volatile uint8_t M1);
263
264 void delay(int i);
265
266 uint16_t GenChecksum(uint16_t Address, uint16_t Command);
267
268 void UpdateAddress(uint16_t OldAddress, uint16_t NewAddress);
269
```

```
270 void MotorControl(uint16_t Address, uint16_t Speed);
271
272 void RequestStatus(uint16_t Address);
273
274 /****************** Usart & Buffer Functions ******************/
275 void Usart2Put(uint8_t ch);
276
277 uint8_t Usart2Get(void);
278
279 void Usart6Put(uint8_t ch);
280
281 uint8_t Usart6Get(void);
282
283 void BufferInit(__IO FIFO_TypeDef *buffer);
284
285 ErrorStatus BufferPut(__IO FIFO_TypeDef *buffer, uint8_t ch);
286
287 ErrorStatus BufferGet(__IO FIFO_TypeDef *buffer, uint8_t *ch);
288
289 ErrorStatus BufferIsEmpty(__IO FIFO_TypeDef buffer);
290
291
292 /****************** Function that initializes all pin types (I think)
… ******************/
293 void PinInit(const uint16_t PIN, GPIO_TypeDef* PORT,const uint32_t CLK,
… const uint8_t CH, const uint8_t SOURCE, TIM_TypeDef* TIM, const uint32_t
… TIM_RCC, const uint8_t TIM_AF, int INPUT);
294
295
296
297 #endif
298
299 ---------------------------------------------- Functions.c
… ----------------------------------------------
300
301 #include "functions.h"
302
303 /*************************** X5 Code
… ******************************************/
```

```
304 void PinOutput(const uint16_t PIN, GPIO_TypeDef* PORT,const uint32_t
  … CLK){
305
306   RCC_AHB1PeriphClockCmd(CLK, ENABLE);
307
308   GPIO_InitTypeDef GPIO_InitStructure;
309   GPIO_InitStructure.GPIO_Pin =  PIN;
310   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
311   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
312
313   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
314   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
315
316   GPIO_Init(PORT, &GPIO_InitStructure);
317 }
318
319 void PinInput(const uint16_t PIN, GPIO_TypeDef* PORT,const uint32_t CLK){
320
321   RCC_AHB1PeriphClockCmd(CLK, ENABLE);
322
323   GPIO_InitTypeDef GPIO_InitStructure;
324   GPIO_InitStructure.GPIO_Pin =  PIN;
325   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
326   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
327
328   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
329   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
330
331   GPIO_Init(PORT, &GPIO_InitStructure);
332 }
333
334 void Mux_Init(){
335   PinOutput(MUX_0_PIN, MUX_0_PORT, MUX_0_CLK);
336   GPIO_ResetBits(MUX_0_PORT, MUX_0_PIN);
337   PinOutput(MUX_1_PIN, MUX_1_PORT, MUX_1_CLK);
338   GPIO_ResetBits(MUX_1_PORT, MUX_1_PIN);
339 }
340
341 void TurnerSelectInit(){
```

```
342   PinOutput(TURNER_SEL_0_PIN, TURNER_SEL_0_PORT, TURNER_SEL_0_CLK);
343   GPIO_ResetBits(TURNER_SEL_0_PORT, TURNER_SEL_0_PIN);
344   PinOutput(TURNER_SEL_1_PIN, TURNER_SEL_1_PORT, TURNER_SEL_1_CLK);
345   GPIO_ResetBits(TURNER_SEL_1_PORT, TURNER_SEL_1_PIN);
346 }
347
348 void TurnerDirInit(){
349   PinOutput(TURNER_DIR_PIN, TURNER_DIR_PORT, TURNER_DIR_CLK);
350 }
351
352 void WaterInit(){
353   PinInput(WATER_0_PIN, WATER_0_PORT, WATER_0_CLK);
354   PinInput(WATER_1_PIN, WATER_1_PORT, WATER_1_CLK);
355   PinInput(WATER_2_PIN, WATER_2_PORT, WATER_2_CLK);
356   PinInput(WATER_3_PIN, WATER_3_PORT, WATER_3_CLK);
357 }
358
359 void FuseInit(){
360   PinInput(FUSE_DECT_PIN, FUSE_DECT_PORT, FUSE_DECT_CLK);
361 }
362
363 void TurnerPWMInit(){
364   /* GPIOD clock enable */
365   RCC_APB2PeriphClockCmd(TURNER_PWM_TIM_RCC, ENABLE);
366
367   GPIO_InitTypeDef GPIO_InitStructure;
368
369   /*------------------------- GPIO Configuration
…  ----------------------------*/
370   GPIO_InitStructure.GPIO_Pin = TURNER_PWM_1_PIN;
371   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
372   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
373   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
374   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
375   GPIO_Init(TURNER_PWM_1_PORT, &GPIO_InitStructure);
376
377   /* Connect TIMx pins to AF */
378   GPIO_PinAFConfig(TURNER_PWM_1_PORT, TURNER_PWM_1_SOURCE,
…  TURNER_PWM_TIM_AF);//need pinsource not just pin
```

```
379
380   TIM_OCInitTypeDef  TIM_OCInitStructure;
381   TIM_TimeBaseInitTypeDef  TIM_TimeBaseStructure;
382   uint16_t Period;
383
384   Period = 1000000 / 2000; // 20 KHz for 1MHz prescaled
385        //100000
386   /* Time base configuration */
387   TIM_TimeBaseStructure.TIM_Prescaler = ((SystemCoreClock / 10000000) /
…  2) - 1; // Get clock to 1 MHz on STM32F4
388   TIM_TimeBaseStructure.TIM_Period = Period - 1;
389   TIM_TimeBaseStructure.TIM_ClockDivision = 0;
390   TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
391   TIM_TimeBaseInit(TURNER_PWM_TIM, &TIM_TimeBaseStructure);
392
393   /* Enable TIMx Preload register on ARR */
394   TIM_ARRPreloadConfig(TURNER_PWM_TIM, ENABLE);
395
396   /* TIM PWMXZ Mode configuration */
397   TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;
398   TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;
399   TIM_OCInitStructure.TIM_Pulse = Period / 2; // 50%
400   TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;
401
402   /* Output Compare PWM1 Mode configuration: Channel1 PD.12 */
403   TIM_OC1Init(TURNER_PWM_TIM, &TIM_OCInitStructure);
404   TIM_OC1PreloadConfig(TURNER_PWM_TIM, TIM_OCPreload_Enable);
405
406   /* TIMx enable counter */
407   TIM_Cmd(TURNER_PWM_TIM, ENABLE);
408   TURNER_PWM_TIM->CCR1 = 0;
409 }
410
411 void NVIC_Config(void){
412
413   NVIC_InitTypeDef NVIC_InitStructure;
414
415   /* Enable the USARTx Interrupt */
416   NVIC_InitStructure.NVIC_IRQChannel = USART2_IRQn;
```

```
417   NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
418   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
419   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
420   NVIC_Init(&NVIC_InitStructure);
421 }
422
423 void LedInit(){
424   PinOutput(LED_0_PIN, LED_0_PORT, LED_0_CLK);
425   PinOutput(LED_1_PIN, LED_1_PORT, LED_1_CLK);
426   PinOutput(LED_2_PIN, LED_2_PORT, LED_2_CLK);
427   PinOutput(LED_3_PIN, LED_3_PORT, LED_3_CLK);
428 }
429
430 void Usart2Init(){ //fix this
431
432   BufferInit(&U2Rx);
433   BufferInit(&U2Tx);
434
435   NVIC_Config();
436
437   RCC_APB1PeriphClockCmd(RCC_APB1Periph_USART2, ENABLE);
438   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOB, ENABLE);
439
440   GPIO_InitTypeDef GPIO_InitStructure;
441
442   GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_10 | GPIO_Pin_11;//
443   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
444   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
445   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
446   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
447   GPIO_Init(GPIOB, &GPIO_InitStructure);
448
449   GPIO_PinAFConfig(GPIOB, GPIO_PinSource10, GPIO_AF_USART2);
450   GPIO_PinAFConfig(GPIOB, GPIO_PinSource11, GPIO_AF_USART2);
451
452   USART_InitTypeDef USART_InitStructure;
453
454   USART_InitStructure.USART_BaudRate = 115200;
455   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
```

```
456   USART_InitStructure.USART_StopBits = USART_StopBits_1;
457   USART_InitStructure.USART_Parity = USART_Parity_No;
458   USART_InitStructure.USART_HardwareFlowControl =
  … USART_HardwareFlowControl_None;
459
460   USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
461
462   USART_Init(USART2, &USART_InitStructure);
463
464   USART_Cmd(USART2, ENABLE);
465
466   USART_ITConfig(USART2, USART_IT_RXNE, ENABLE);
467   USART_ITConfig(USART2, USART_IT_TXE, DISABLE);
468 }
469
470 void Usart6Init(){ //fix this
471   PinOutput(USART6_ENABLE_PIN, USART6_ENABLE_PORT, USART6_ENABLE_CLK);
472   PinOutput(USART6_DISABLE_PIN, USART6_DISABLE_PORT, USART6_DISABLE_CLK);
473
474   BufferInit(&U6Rx);
475   BufferInit(&U6Tx);
476
477   NVIC_Config();
478
479   RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART6, ENABLE);
480   RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOC, ENABLE);
481
482   GPIO_InitTypeDef GPIO_InitStructure;
483
484   GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_6 | GPIO_Pin_7;
485   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
486   GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
487   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
488   GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;
489   GPIO_Init(GPIOC, &GPIO_InitStructure);
490
491   GPIO_PinAFConfig(GPIOC, GPIO_PinSource6, GPIO_AF_USART6);
492   GPIO_PinAFConfig(GPIOC, GPIO_PinSource7, GPIO_AF_USART6);
493
```

```
494   USART_InitTypeDef USART_InitStructure;
495
496   USART_InitStructure.USART_BaudRate = 115200;
497   USART_InitStructure.USART_WordLength = USART_WordLength_8b;
498   USART_InitStructure.USART_StopBits = USART_StopBits_1;
499   USART_InitStructure.USART_Parity = USART_Parity_No;
500   USART_InitStructure.USART_HardwareFlowControl =
  … USART_HardwareFlowControl_None;
501
502   USART_InitStructure.USART_Mode = USART_Mode_Tx | USART_Mode_Rx;
503
504   USART_Init(USART6, &USART_InitStructure);
505
506   USART_Cmd(USART6, ENABLE);
507
508   USART_ITConfig(USART6, USART_IT_RXNE, ENABLE);
509   USART_ITConfig(USART6, USART_IT_TXE, DISABLE);
510 }
511
512 void MUXControl(volatile uint8_t M0, volatile uint8_t M1){
513   if(M0 == 1){
514     GPIO_SetBits(MUX_0_PORT, MUX_0_PIN);
515   }else{
516     GPIO_ResetBits(MUX_0_PORT, MUX_0_PIN);
517   }
518
519   if(M1 == 1){
520     GPIO_SetBits(MUX_1_PORT, MUX_1_PIN);
521   }else{
522     GPIO_ResetBits(MUX_1_PORT, MUX_1_PIN);
523   }
524 }
525
526 void delay(int i){    //100 000 is about 1/4th a second, 100 000 000 is
  … about 2.5 sec
527   while (i-- > 0) {
528     asm("nop");       /* This stops it optimising code out */
529   }
530 }
```

```
531
532 uint16_t GenChecksum(uint16_t address, uint16_t command){
533   return (address+command)&0xFF;
534 }
535
536 void UpdateAddress(uint16_t OldAddress, uint16_t NewAddress){
537   //Getting things set up
538   int OldAddress1 = ((OldAddress & 0xF0) >> 4);
539   if(OldAddress1 >= 0x0A){
540       OldAddress1 += 0x37;
541   }else{
542       OldAddress1 += 0x30;
543   }
544
545   int OldAddress2 = ((OldAddress & 0x0F)) ;
546   if(OldAddress2 >= 0x0A){
547       OldAddress2 += 0x37;
548   }else{
549       OldAddress2 += 0x30;
550   }
551
552   int NewAddress1 = ((NewAddress & 0xF0) >> 4);
553   if(NewAddress1 >= 0x0A){
554       NewAddress1 += 0x37;
555   }else{
556       NewAddress1 += 0x30;
557   }
558
559   int NewAddress2 = ((NewAddress & 0x0F)) ;
560   if(NewAddress2 >= 0x0A){
561       NewAddress2 += 0x37;
562   }else{
563       NewAddress2 += 0x30;
564   }
565
566   GPIO_SetBits(USART6_ENABLE_PORT, USART6_ENABLE_PIN);
567   GPIO_SetBits(USART6_DISABLE_PORT, USART6_DISABLE_PIN);
568
569   //Start !
```

```
570   USART_SendData(USART6, 0x24);
571   delay(3000);
572
573   //Old address byte
574   USART_SendData(USART6, OldAddress1);
575   delay(3000);
576
577   USART_SendData(USART6, OldAddress2);
578   delay(3000);
579
580   //Change address command
581   USART_SendData(USART6, 0x30);//0
582   delay(3000);
583
584   USART_SendData(USART6, 0x42);//B
585   delay(3000);
586
587   //New Address byte
588   USART_SendData(USART6, NewAddress1);
589   delay(3000);
590
591   USART_SendData(USART6, NewAddress1);
592   delay(3000);
593
594   //checksum byte - Same as new address
595   USART_SendData(USART6, NewAddress1);
596   delay(3000);
597
598   USART_SendData(USART6, NewAddress1);
599   delay(3000);
600
601   //end $
602   USART_SendData(USART6, 0x21);
603   delay(3000);
604
605   GPIO_ResetBits(USART6_ENABLE_PORT, USART6_ENABLE_PIN);
606   GPIO_ResetBits(USART6_DISABLE_PORT, USART6_DISABLE_PIN);
607 }
608
```

```
609 void MotorControl(uint16_t Address, uint16_t Speed){
610   //Getting things set up
611   int Address1 = ((Address & 0xF0) >> 4);
612   if(Address1 >= 0x0A){
613       Address1 += 0x37;
614   }else{
615       Address1 += 0x30;
616   }
617 
618   int Address2 = ((Address & 0x0F)) ;
619   if(Address2 >= 0x0A){
620       Address2 += 0x37;
621   }else{
622       Address2 += 0x30;
623   }
624 
625   int Speed1 = ((Speed & 0xF0) >> 4);
626   if(Speed1 >= 0x0A){
627       Speed1 += 0x37;
628   }else{
629       Speed1 += 0x30;
630   }
631 
632   int Speed2 = ((Speed & 0x0F));
633   if(Speed2 >= 0x0A){
634       Speed2 += 0x37;
635   }else{
636       Speed2 += 0x30;
637   }
638 
639   int Checksum = GenChecksum(Address, Speed);
640   int Checksum1 = ((Checksum & 0xF0) >> 4);
641   if(Checksum1 >= 0x0A){
642       Checksum1 += 0x37;
643   }else{
644       Checksum1 += 0x30;
645   }
646 
647   int Checksum2 = ((Checksum & 0x0F));
```

```
648   if(Checksum2 >= 0x0A){
649       Checksum2 += 0x37;
650   }else{
651       Checksum2 += 0x30;
652   }
653
654   GPIO_SetBits(USART6_ENABLE_PORT, USART6_ENABLE_PIN);
655   GPIO_SetBits(USART6_DISABLE_PORT, USART6_DISABLE_PIN);
656
657   //Start !
658   USART_SendData(USART6, 0x24);
659   delay(3000);
660
661   //address byte
662   USART_SendData(USART6, Address1);
663   delay(3000);
664
665   USART_SendData(USART6, Address2);
666   delay(3000);
667
668   //speed byte
669   USART_SendData(USART6, Speed1);
670   delay(3000);
671
672   USART_SendData(USART6, Speed2);
673   delay(3000);
674
675   //nothing byte
676   USART_SendData(USART6, 0x30);
677   delay(3000);
678
679   USART_SendData(USART6, 0x30);
680   delay(3000);
681
682   //checksum byte
683   USART_SendData(USART6, Checksum1);
684   delay(3000);
685
686   USART_SendData(USART6, Checksum2);
```

```
687   delay(3000);
688
689   //end $
690   USART_SendData(USART6, 0x21);
691   delay(3000);
692
693   GPIO_ResetBits(USART6_ENABLE_PORT, USART6_ENABLE_PIN);
694   GPIO_ResetBits(USART6_DISABLE_PORT, USART6_DISABLE_PIN);
695 }
696
697 void RequestStatus(uint16_t Address){
698   //Getting things set up
699   int Address1 = ((Address & 0xF0) >> 4);
700   if(Address1 >= 0x0A){
701       Address1 += 0x37;
702   }else{
703       Address1 += 0x30;
704   }
705
706   int Address2 = ((Address & 0x0F)) ;
707   if(Address2 >= 0x0A){
708       Address2 += 0x37;
709   }else{
710       Address2 += 0x30;
711   }
712
713   GPIO_SetBits(USART6_ENABLE_PORT, USART6_ENABLE_PIN);
714   GPIO_SetBits(USART6_DISABLE_PORT, USART6_DISABLE_PIN);
715
716   //Start !
717   USART_SendData(USART6, 0x24);
718   delay(3000);
719
720   //address byte
721   USART_SendData(USART6, Address1);
722   delay(3000);
723
724   USART_SendData(USART6, Address2);
725   delay(3000);
```

```
726
727   //checksum byte
728   USART_SendData(USART6, Address1);
729   delay(3000);
730
731   USART_SendData(USART6, Address2);
732   delay(3000);
733
734   //end $
735   USART_SendData(USART6, 0x21);
736   delay(3000);
737
738   GPIO_ResetBits(USART6_ENABLE_PORT, USART6_ENABLE_PIN);
739   GPIO_ResetBits(USART6_DISABLE_PORT, USART6_DISABLE_PIN);
740 }
741
742
743
744 /****************** Usart & Buffer Functions ******************/
745 void Usart2Put(uint8_t ch){
746   //put char to the buffer
747   BufferPut(&U2Tx, ch);
748   //enable Transmit Data Register empty interrupt
749   USART_ITConfig(USART2, USART_IT_TXE, ENABLE);
750 }
751
752 uint8_t Usart2Get(void){
753   uint8_t ch;
754   //check if buffer is empty
755   while (BufferIsEmpty(U2Rx) ==SUCCESS);
756   BufferGet(&U2Rx, &ch);
757   return ch;
758 }
759
760 void Usart6Put(uint8_t ch){
761   //put char to the buffer
762   BufferPut(&U6Tx, ch);
763   //enable Transmit Data Register empty interrupt
764   USART_ITConfig(USART6, USART_IT_TXE, ENABLE);
```

```
765 }
766
767 uint8_t Usart6Get(void){
768   uint8_t ch;
769   //check if buffer is empty
770   while (BufferIsEmpty(U6Rx) ==SUCCESS);
771   BufferGet(&U6Rx, &ch);
772   return ch;
773 }
774
775 //typedef enum {ERROR = 0, SUCCESS = !ERROR} ErrorStatus;
776 void BufferInit(__IO FIFO_TypeDef *buffer){
777
778   buffer->count = 0;//0 bytes in buffer
779   buffer->in = 0;//index points to start
780   buffer->out = 0;//index points to start
781 }
782
783 ErrorStatus BufferPut(__IO FIFO_TypeDef *buffer, uint8_t ch){
784   if ( ch == 0x12){
785     buffer->in=0;
786   }
787   //if(buffer->count==USARTBUFFSIZE)
788   //  return ERROR;//buffer full
789   buffer->buff[buffer->in++]=ch;
790   //buffer->count++;
791   if(buffer->in==USARTBUFFSIZE)
792     buffer->in=0;//start from beginning
793   return SUCCESS;
794 }
795
796 ErrorStatus BufferGet(__IO FIFO_TypeDef *buffer, uint8_t *ch){
797   if(buffer->count==0)
798     return ERROR;//buffer empty
799   *ch=buffer->buff[buffer->out++];
800   buffer->count--;
801   if(buffer->out==USARTBUFFSIZE)
802     buffer->out=0;//start from beginning
803   return SUCCESS;
```

```
804 }
805
806 ErrorStatus BufferIsEmpty(__IO FIFO_TypeDef buffer){
807   if(buffer.count==0)
808     return SUCCESS;//buffer full
809   return ERROR;
810 }
811
812
813 /****************** Function that initializes all pin types (I think)
… ******************/
814 void PinInit(const uint16_t PIN, GPIO_TypeDef* PORT,const uint32_t CLK,
… const uint8_t CH, const uint8_t SOURCE, TIM_TypeDef* TIM, const uint32_t
… TIM_RCC, const uint8_t TIM_AF, int INPUT){  //8 inputs to function
815     //enable RCC on gpio pin NOT TIMER THINGS
816     //AKA need to do any GPIO at all
817     //Gotta do RCC stuff first.
818     if(TIM == TIM14){
819         RCC_APB1PeriphClockCmd(TIM_RCC, ENABLE);
820
821     }else if(TIM != NULL){
822         if(TIM_RCC == RCC_APB2Periph_TIM9 | TIM_RCC ==
… RCC_APB2Periph_TIM8 ){ //timer 9 is special.
823             RCC_APB2PeriphClockCmd(CLK, ENABLE);
824
825         }else{
826
827             RCC_APB1PeriphClockCmd(CLK, ENABLE);
828         }
829     }
830
831     RCC_AHB1PeriphClockCmd(CLK, ENABLE);
832
833     //Start initinalizing GPIO
834     GPIO_InitTypeDef GPIO_InitStructure;
835     GPIO_InitStructure.GPIO_Pin =  PIN;
836
837     //dynamic gpio stuff
838     if( INPUT == 1 ){ //1 for out,2 for AF,  0 or else for input
```

```
839
840         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
841         GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
842     }else if( INPUT == 2){ // for things that need Alternate outputs. aka
    … pierphials
843
844         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF;
845         GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
846
847     }else{
848         GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
849     }
850
851     //pretty sure always the same gpio stuff
852     GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;    //I think the
    … same in all cases
853     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_100MHz;  //Same as above
854
855     //stuff for pwm NOTE* Remember to exclude other timers for other
    … periphials
856     if (TIM == TIM8){ //this part for LED init
857
858     uint16_t PrescalerValue = (uint16_t) (SystemCoreClock /8400000) - 1;
    … //work on this
859
860         TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
861         TIM_OCInitTypeDef TIM_OCInitStruct;
862
863         //this part
864         TIM_TimeBaseStructInit( &TIM_TimeBaseInitStruct );
865         TIM_TimeBaseInitStruct.TIM_ClockDivision = 0;
866         TIM_TimeBaseInitStruct.TIM_Period =2000-1;   // 0..999
867         TIM_TimeBaseInitStruct.TIM_Prescaler = PrescalerValue; // Div 240
868         TIM_TimeBaseInit( TIM, &TIM_TimeBaseInitStruct );
869         //to this part need to be fixed to generate the correct pwm
    … frequency
870
871         TIM_OCStructInit( &TIM_OCInitStruct );
872         TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
```

```
873         TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1; //has nothing to
    … do with channel
874         // Initial duty cycle equals 0%. Value can range from zero to
    … 1000.
875         TIM_OCInitStruct.TIM_Pulse = 500; // 0 .. 1000 (0=Always Off,
    … 1000=Always On)
876 
877         if(CH == 1){
878             TIM_OC1Init( TIM, &TIM_OCInitStruct ); //add in IF statement
    … here
879         }else if(CH == 2){
880             TIM_OC2Init( TIM, &TIM_OCInitStruct );
881         }else if(CH == 3){
882             TIM_OC3Init( TIM, &TIM_OCInitStruct );
883         }
884             TIM_Cmd( TIM, ENABLE );
885     }else if (TIM != 0 && TIM != TIM14){ //then not a pin that has a
    … timer!
886 
887     uint16_t PrescalerValue = (uint16_t) ((SystemCoreClock /2) / 45000) -
    … 1;   //work on this
888 
889         TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStruct;
890         TIM_OCInitTypeDef TIM_OCInitStruct;
891 
892         //this part
893         TIM_TimeBaseStructInit( &TIM_TimeBaseInitStruct );
894         TIM_TimeBaseInitStruct.TIM_ClockDivision = 0;
895         TIM_TimeBaseInitStruct.TIM_Period =2000 - 1;   // 0..999
896         TIM_TimeBaseInitStruct.TIM_Prescaler = PrescalerValue; // Div 240
897         TIM_TimeBaseInit( TIM, &TIM_TimeBaseInitStruct );
898         //to this part need to be fixed to generate the correct pwm
    … frequency
899 
900         TIM_OCStructInit( &TIM_OCInitStruct );
901         TIM_OCInitStruct.TIM_OutputState = TIM_OutputState_Enable;
902         TIM_OCInitStruct.TIM_OCMode = TIM_OCMode_PWM1; //has nothing to
    … do with channel
903         // Initial duty cycle equals 0%. Value can range from zero to
```

```
1000.
        TIM_OCInitStruct.TIM_Pulse = 500; // 0 .. 1000 (0=Always Off,
1000=Always On)

        if(CH == 1){
            TIM_OC1Init( TIM, &TIM_OCInitStruct ); //add in IF statement
here
        }else if(CH == 2){
            TIM_OC2Init( TIM, &TIM_OCInitStruct );
        }else if(CH == 3){
            TIM_OC3Init( TIM, &TIM_OCInitStruct );
        }
            TIM_Cmd( TIM, ENABLE );
    }


    if( INPUT == 2){
            GPIO_PinAFConfig(PORT, SOURCE, TIM_AF); //changes for every
pin
    }

    //initinalize the GPIO's
    GPIO_Init(PORT, &GPIO_InitStructure);

}

------------------------------------------------ Main.c
--------------------------------------------------

#include "functions.h"
volatile FIFO_TypeDef U2Rx, U2Tx, U6Rx, U6Tx;
volatile uint8_t flag=0;


int main(void){

        SystemInit();

        LedInit();
```

```
938         Usart6Init();
939 
940     while(1){
941 
942         //LF, RF, LB, RB, Nothing , RFV, LBV, RBV, nothing, LFV
943 
…   MotorControl(U2Rx.buff[1],U2Rx.buff[2],U2Rx.buff[3],U2Rx.buff[4],0,U2Rx.
…   buff[6],U2Rx.buff[7],U2Rx.buff[8],0,U2Rx.buff[5]);
944         ShiftControl(U2Rx.buff[9], U2Rx.buff[10],U2Rx.buff[11],
…   U2Rx.buff[12]);
945         MUXControl(U2Rx.buff[13],U2Rx.buff[14]);
946     }
947 }
948 
```

```
1  ---------------------------------------- XBoxController.h
…  ----------------------------------------------
2
3  //Copyright 2013 Purdue University ROV Team
4  //Code by Clement Lan, Nick Molo
5
6  //Please ask permission (clement.lan@gmail.com, nmolo@purdue.edu)
7  //before using this code
8
9  #pragma once
10 #include <Windows.h>
11 #include <XInput.h>
12
13 #pragma comment(lib, "XInput.lib")
14 #define INPUT_DEADZONE  ( 0.24f * FLOAT(0x7FFF) )
15 #define TRIGGER_THRESH 20
16
17 class XBoxController
18 {
19 public:
20     XBoxController(int id);
21     XINPUT_STATE GetState();
22     bool isConnected();
23     void vibrate(int leftVal, int rightVal);
24 private:
25     int controllerID;
26     XINPUT_STATE controllerState;
27 };
28
29 ---------------------------------------- BaseStation.h
…  ---___-------------------------------------------
30
31 //Copyright 2013 Purdue University ROV Team
32 //Code by Clement Lan, Nick Molo
33
34 //Please ask permission (clement.lan@gmail.com, nmolo@purdue.edu)
35 //before using this code
36
37 #pragma once
```

```
38 #include "XBoxController.h"
39
40
41 #define INPUT_DEADZONE  ( 0.24f * FLOAT(0x7FFF) )
42 #define TRIGGER_THRESH 20
43 #define ACK 13
44
45 namespace ROV_BaseStation {
46
47     using namespace System;
48     //using System::Math;
49     using namespace System::ComponentModel;
50     using namespace System::Collections;
51     using namespace System::Windows::Forms;
52     using namespace System::Data;
53     using namespace System::Drawing;
54     using namespace System::IO::Ports;
55
56
57
58
59     /// <summary>
60     /// Summary for BaseStation
61     /// </summary>
62     public ref class BaseStation : public System::Windows::Forms::Form
63     {
64     public:
65         BaseStation(void);
66
67     protected:
68         /// <summary>
69         /// Clean up any resources being used
70         /// </summary>
71         ~BaseStation()
72         {
73             if (serialPort1->IsOpen) serialPort1->Close();
74             if (components)
75             {
76                 delete components;
```

```
77              }
78          }
79
80      public: System::IO::Ports::SerialPort^  serialPort1;
81      private: System::Windows::Forms::Timer^  timer1;
82      private: System::Windows::Forms::Label^  estopDisp;
83      private: System::Windows::Forms::Label^  recValues;
84      private: System::Windows::Forms::ComboBox^  CPortBox;
85      private: System::Windows::Forms::Label^  serialLabel;
86      private: System::Windows::Forms::Button^  button2;
87      private: System::Windows::Forms::GroupBox^  ToolPanel;
88      private: System::Windows::Forms::Label^  ManipStatus;
89      private: System::Windows::Forms::Label^  label1;
90      private: System::Windows::Forms::Label^  RFThrust;
91      private: System::Windows::Forms::Label^  LFThrust;
92      private: System::Windows::Forms::Label^  LBThrust;
93      private: System::Windows::Forms::Label^  RBThrust;
94      private: System::Windows::Forms::Label^  RBVThrust;
95      private: System::Windows::Forms::Label^  LBVThrust;
96      private: System::Windows::Forms::Label^  RFVThrust;
97      private: System::Windows::Forms::Label^  LFVThrust;
98      private: System::Windows::Forms::Panel^  panel1;
99      private: System::Windows::Forms::GroupBox^  groupBox1;
100     private: System::Windows::Forms::TabPage^  Serial;
101     private: System::Windows::Forms::TabPage^  ThrusterToggle;
102     private: System::Windows::Forms::CheckBox^  DisableLBV;
103     private: System::Windows::Forms::CheckBox^  DisableRBV;
104     private: System::Windows::Forms::CheckBox^  DisableRFV;
105     private: System::Windows::Forms::CheckBox^  DisableLFV;
106     private: System::Windows::Forms::CheckBox^  DisableLB;
107     private: System::Windows::Forms::CheckBox^  DisableRB;
108     private: System::Windows::Forms::CheckBox^  DisableRF;
109     private: System::Windows::Forms::CheckBox^  DisableLF;
110     private: System::Windows::Forms::TabPage^  Main;
111     private: System::Windows::Forms::Label^  label8;
112     private: System::Windows::Forms::Label^  label7;
113     private: System::Windows::Forms::Label^  label6;
114     private: System::Windows::Forms::Label^  label5;
115     private: System::Windows::Forms::TrackBar^  PitchRollBar;
```

```
116     private: System::Windows::Forms::TrackBar^  StrafeSpeedCTL;
117     private: System::Windows::Forms::TrackBar^  VertLimiter;
118     private: System::Windows::Forms::TrackBar^  HorizLimiter;
119     private: System::Windows::Forms::TabControl^  tabControl1;
120     private: Microsoft::VisualBasic::PowerPacks::ShapeContainer^
  … shapeContainer2;
121     private: Microsoft::VisualBasic::PowerPacks::OvalShape^  ovalShape1;
122     private: Microsoft::VisualBasic::PowerPacks::OvalShape^  ovalShape2;
123     private: Microsoft::VisualBasic::PowerPacks::Printing::PrintForm^
  … printForm1;
124     private: Microsoft::VisualBasic::PowerPacks::LineShape^  Roll;
125     private: System::Windows::Forms::Label^  label10;
126     private: System::Windows::Forms::Label^  label9;
127     private: Microsoft::VisualBasic::PowerPacks::LineShape^  Pitch;
128     private: Microsoft::VisualBasic::PowerPacks::LineShape^  lineShape6;
129     private: Microsoft::VisualBasic::PowerPacks::LineShape^  lineShape5;
130     private: Microsoft::VisualBasic::PowerPacks::LineShape^  lineShape4;
131     private: Microsoft::VisualBasic::PowerPacks::LineShape^  lineShape3;
132     private: System::Windows::Forms::Label^  label13;
133     private: System::Windows::Forms::Label^  label12;
134     private: System::Windows::Forms::GroupBox^  groupBox4;
135     private: System::Windows::Forms::Label^  label15;
136     private: System::Windows::Forms::GroupBox^  groupBox3;
137     private: System::Windows::Forms::Label^  label14;
138     private: System::Windows::Forms::Label^  C2Status;
139     private: System::Windows::Forms::Label^  C1Status;
140     private: System::Windows::Forms::Button^  ControlCheck;
141     private: System::Windows::Forms::CheckedListBox^  checkedListBox1;
142     private: System::Windows::Forms::Label^  label18;
143     private: System::Windows::Forms::Label^  label17;
144     private: System::Windows::Forms::Label^  label16;
145     private: System::Windows::Forms::Label^  label11;
146     private: System::Windows::Forms::Label^  label4;
147     private: System::Windows::Forms::Label^  label3;
148     private: System::Windows::Forms::Label^  label19;
149     private: System::Windows::Forms::TabPage^  Debug;
150     private: System::Windows::Forms::Label^  label21;
151     private: System::Windows::Forms::Label^  label20;
152     private: System::Windows::Forms::Button^  TimerStart;
```

```
153     private: System::Windows::Forms::GroupBox^  groupBox2;
154     private: System::Windows::Forms::Label^  TimeSec;
155     private: System::Windows::Forms::Timer^  timer2;
156     private: System::Windows::Forms::GroupBox^  groupBox6;
157     private: System::Windows::Forms::Label^  label24;
158     private: System::Windows::Forms::Label^  label23;
159     private: System::Windows::Forms::Label^  label22;
160     private: System::Windows::Forms::Label^  turner1;
161     private: Microsoft::VisualBasic::PowerPacks::ShapeContainer^
  … shapeContainer1;
162     private: Microsoft::VisualBasic::PowerPacks::OvalShape^  ovalShape6;
163     private: Microsoft::VisualBasic::PowerPacks::OvalShape^  ovalShape5;
164     private: Microsoft::VisualBasic::PowerPacks::OvalShape^  ovalShape4;
165     private: Microsoft::VisualBasic::PowerPacks::OvalShape^  ovalShape3;
166     private: System::Windows::Forms::Label^  label25;
167     private: System::Windows::Forms::TrackBar^  TurnerBar;
168     private: System::Windows::Forms::Label^  SerialDebug;
169     private: System::Windows::Forms::Button^  SerialForce;
170     private: System::Windows::Forms::StatusStrip^  statusStrip1;
171     private: System::Windows::Forms::ToolStripStatusLabel^
  … SerialLabel2;
172     private: System::Windows::Forms::ToolStripStatusLabel^  PolarTest;
173     private: System::Windows::Forms::TabPage^  Status;
174     private: System::Windows::Forms::GroupBox^  groupBox5;
175     private: System::Windows::Forms::Label^  label34;
176     private: System::Windows::Forms::Label^  label33;
177     private: System::Windows::Forms::Label^  label32;
178     private: System::Windows::Forms::Label^  label31;
179     private: System::Windows::Forms::Label^  label30;
180     private: System::Windows::Forms::Label^  label29;
181     private: System::Windows::Forms::Label^  label28;
182     private: System::Windows::Forms::Label^  label27;
183     private: System::Windows::Forms::Label^  label26;
184     private: System::Windows::Forms::Label^  label2;
185     private: System::Windows::Forms::DataGridView^  dataGridView1;
186     private: System::Windows::Forms::DataGridViewTextBoxColumn^  Motor;
187     private: System::Windows::Forms::DataGridViewTextBoxColumn^
  … Temperature;
188     private: System::Windows::Forms::DataGridViewTextBoxColumn^
```

```
188… Voltage;
189      private: System::Windows::Forms::DataGridViewTextBoxColumn^
  … Current;
190      private: System::Windows::Forms::DataGridViewTextBoxColumn^
  … FaultStatus;
191      private: System::Windows::Forms::Button^  TimerReset;
192      private: System::Windows::Forms::Label^  label36;
193      private: System::Windows::Forms::Label^  TimeMin;
194
195
196  private: //function prototypes
197          private: //function prototypes
198          System::ComponentModel::IContainer^  components;
199          System::Void SendThrusterVec();
200          System::Void timer1_Tick(System::Object^  sender,
  … System::EventArgs^  e);
201          System::Void timer2_Tick(System::Object^  sender,
  … System::EventArgs^  e);
202          System::Void GenerateThrusterVec(int* inputs);
203          System::Void CheckControllerStatus();
204          System::Void UpdateConnection();
205          System::Void UpdateGuiThrust();
206          System::Void AdjustVector();
207          int GetHorizNonZero();
208          int GetVertNonZero();
209          float GetThrust(float x, float y) { return
  … (x+y)/GetHorizNonZero()*60; }
210          float GetThrustV(float x, float y, float z) { return
  … (x+y+z)/GetVertNonZero()*60; }
211          System::Void UpdateStatButton_Click(System::Object^  sender,
  … System::EventArgs^  e) {
212              CheckControllerStatus();
213          }
214          System::Void VertLimiter_ValueChanged(System::Object^  sender,
  … System::EventArgs^  e) {
215              SpeedDivider_V = (float)(this->VertLimiter->Value)/10;
216          }
217          System::Void StrafeSpeedCTL_ValueChanged(System::Object^
  … sender, System::EventArgs^  e) {
```

```
218             StrafeSpeed = (float)(this->StrafeSpeedCTL->Value)/10;
219         }
220         System::Void HorizLimiter_ValueChanged(System::Object^  sender,
… System::EventArgs^  e) {
221             SpeedDivider_H = (float)(this->HorizLimiter->Value)/10;
222         }
223         System::Void ConnectButton_Click(System::Object^  sender,
… System::EventArgs^  e) {
224              UpdateConnection();
225         }
226         System::Void serialPort1_DataReceived(System::Object^  sender,
… System::IO::Ports::SerialDataReceivedEventArgs^  e);
227         System::Void GenerateMainThrusterVec(void);
228     private: //variables
229     //Objects
230         XBoxController *controller0;
231         XBoxController *controller1;
232         String ^thrustbackstr;
233         String ^mbstring;
234         String ^out;
235     //Arrays
236         int *thrust_vec;
237         int *polar_vec;
238         int *prev_thrust_vec;
239         int *inputs;
240         int *Bbuttonaverage;
241         int *Abuttonaverage;
242         int *Ybuttonaverage;
243         char *rcvstr;
244         int *DupAverage;
245         int *DdownAverage;
246         int* DleftAverage;
247         int *Xbuttonaverage;
248         int *DrightAverage;
249         short *accel_data;
250         short *gyro_data;
251         short *mag_data;
252         int *pb_data;
253         char *faultdata;
```

```
254     //Booleans
255         bool cam_toggleF, cam_toggleR, manip_open, estop, speed;
256         bool vecstatus;
257         bool strafing;
258         bool prox;
259         bool serialstart;
260         bool passcodeOK;
261         bool unitState;
262         bool missiontime;
263     //Floats
264         float SpeedDivider_H;
265         float SpeedDivider_V;
266         float StrafeSpeed;
267         float PitchRollFactor;
268         float depth, depthoffset;
269         float nLX, nLY, nRY, nRX, nZ, LX, LY;
270     //Ints
271         int red;
272         int green;
273         int blue;
274         int bCount;
275         int lsCount;
276         int rsCount;
277         int startCount;
278         int selectCount;
279         int Dupcount;
280         int Ddowncount;
281         int Dleftcount;
282         int Drightcount;
283         int xcount;
284         int yCount;
285         int test_counter;
286         int test_counter2;
287         int runtimeus;
288         long fifteentimer;
289         int prevrange;
290         int timercount;
291         /// <summary>
292         /// Required designer variable.
```

```
293         /// </summary>
294 
295 
296 
297 #pragma region Windows Form Designer generated code
298         /// <summary>
299         /// Required method for Designer support - do not modify
300         /// the contents of this method with the code editor.
301         /// </summary>
302         void InitializeComponent(void)
303         {
304             this->components = (gcnew
    System::ComponentModel::Container());
305             System::ComponentModel::ComponentResourceManager^  resources
    = (gcnew
    System::ComponentModel::ComponentResourceManager(BaseStation::typeid));
306             this->timer1 = (gcnew
    System::Windows::Forms::Timer(this->components));
307             this->serialPort1 = (gcnew
    System::IO::Ports::SerialPort(this->components));
308             this->estopDisp = (gcnew System::Windows::Forms::Label());
309             this->recValues = (gcnew System::Windows::Forms::Label());
310             this->CPortBox = (gcnew System::Windows::Forms::ComboBox());
311             this->serialLabel = (gcnew System::Windows::Forms::Label());
312             this->button2 = (gcnew System::Windows::Forms::Button());
313             this->ToolPanel = (gcnew
    System::Windows::Forms::GroupBox());
314             this->ManipStatus = (gcnew System::Windows::Forms::Label());
315             this->label1 = (gcnew System::Windows::Forms::Label());
316             this->RFThrust = (gcnew System::Windows::Forms::Label());
317             this->LFThrust = (gcnew System::Windows::Forms::Label());
318             this->LBThrust = (gcnew System::Windows::Forms::Label());
319             this->RBThrust = (gcnew System::Windows::Forms::Label());
320             this->RBVThrust = (gcnew System::Windows::Forms::Label());
321             this->LBVThrust = (gcnew System::Windows::Forms::Label());
322             this->RFVThrust = (gcnew System::Windows::Forms::Label());
323             this->LFVThrust = (gcnew System::Windows::Forms::Label());
324             this->panel1 = (gcnew System::Windows::Forms::Panel());
325             this->groupBox1 = (gcnew
```

```
325… System::Windows::Forms::GroupBox());
326              this->label21 = (gcnew System::Windows::Forms::Label());
327              this->label20 = (gcnew System::Windows::Forms::Label());
328              this->label19 = (gcnew System::Windows::Forms::Label());
329              this->label10 = (gcnew System::Windows::Forms::Label());
330              this->label9 = (gcnew System::Windows::Forms::Label());
331              this->shapeContainer2 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::ShapeContainer());
332              this->lineShape6 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::LineShape());
333              this->lineShape5 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::LineShape());
334              this->lineShape4 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::LineShape());
335              this->lineShape3 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::LineShape());
336              this->Pitch = (gcnew
  … Microsoft::VisualBasic::PowerPacks::LineShape());
337              this->Roll = (gcnew
  … Microsoft::VisualBasic::PowerPacks::LineShape());
338              this->ovalShape1 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::OvalShape());
339              this->ovalShape2 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::OvalShape());
340              this->Serial = (gcnew System::Windows::Forms::TabPage());
341              this->groupBox4 = (gcnew
  … System::Windows::Forms::GroupBox());
342              this->ControlCheck = (gcnew
  … System::Windows::Forms::Button());
343              this->C2Status = (gcnew System::Windows::Forms::Label());
344              this->C1Status = (gcnew System::Windows::Forms::Label());
345              this->label14 = (gcnew System::Windows::Forms::Label());
346              this->label15 = (gcnew System::Windows::Forms::Label());
347              this->groupBox3 = (gcnew
  … System::Windows::Forms::GroupBox());
348              this->label13 = (gcnew System::Windows::Forms::Label());
349              this->label12 = (gcnew System::Windows::Forms::Label());
350              this->ThrusterToggle = (gcnew
  … System::Windows::Forms::TabPage());
```

```
351             this->DisableLBV = (gcnew
… System::Windows::Forms::CheckBox());
352             this->DisableRBV = (gcnew
… System::Windows::Forms::CheckBox());
353             this->DisableRFV = (gcnew
… System::Windows::Forms::CheckBox());
354             this->DisableLFV = (gcnew
… System::Windows::Forms::CheckBox());
355             this->DisableLB = (gcnew
… System::Windows::Forms::CheckBox());
356             this->DisableRB = (gcnew
… System::Windows::Forms::CheckBox());
357             this->DisableRF = (gcnew
… System::Windows::Forms::CheckBox());
358             this->DisableLF = (gcnew
… System::Windows::Forms::CheckBox());
359             this->Main = (gcnew System::Windows::Forms::TabPage());
360             this->label25 = (gcnew System::Windows::Forms::Label());
361             this->TurnerBar = (gcnew
… System::Windows::Forms::TrackBar());
362             this->label8 = (gcnew System::Windows::Forms::Label());
363             this->label7 = (gcnew System::Windows::Forms::Label());
364             this->label6 = (gcnew System::Windows::Forms::Label());
365             this->label5 = (gcnew System::Windows::Forms::Label());
366             this->PitchRollBar = (gcnew
… System::Windows::Forms::TrackBar());
367             this->StrafeSpeedCTL = (gcnew
… System::Windows::Forms::TrackBar());
368             this->VertLimiter = (gcnew
… System::Windows::Forms::TrackBar());
369             this->HorizLimiter = (gcnew
… System::Windows::Forms::TrackBar());
370             this->tabControl1 = (gcnew
… System::Windows::Forms::TabControl());
371             this->Status = (gcnew System::Windows::Forms::TabPage());
372             this->dataGridView1 = (gcnew
… System::Windows::Forms::DataGridView());
373             this->Motor = (gcnew
… System::Windows::Forms::DataGridViewTextBoxColumn());
```

```
374             this->Temperature = (gcnew
… System::Windows::Forms::DataGridViewTextBoxColumn());
375             this->Voltage = (gcnew
… System::Windows::Forms::DataGridViewTextBoxColumn());
376             this->Current = (gcnew
… System::Windows::Forms::DataGridViewTextBoxColumn());
377             this->FaultStatus = (gcnew
… System::Windows::Forms::DataGridViewTextBoxColumn());
378             this->Debug = (gcnew System::Windows::Forms::TabPage());
379             this->TimerReset = (gcnew System::Windows::Forms::Button());
380             this->SerialForce = (gcnew
… System::Windows::Forms::Button());
381             this->SerialDebug = (gcnew System::Windows::Forms::Label());
382             this->printForm1 = (gcnew
… Microsoft::VisualBasic::PowerPacks::Printing::PrintForm(this->components
… ));
383             this->groupBox5 = (gcnew
… System::Windows::Forms::GroupBox());
384             this->label34 = (gcnew System::Windows::Forms::Label());
385             this->label33 = (gcnew System::Windows::Forms::Label());
386             this->label32 = (gcnew System::Windows::Forms::Label());
387             this->label31 = (gcnew System::Windows::Forms::Label());
388             this->label30 = (gcnew System::Windows::Forms::Label());
389             this->label29 = (gcnew System::Windows::Forms::Label());
390             this->label28 = (gcnew System::Windows::Forms::Label());
391             this->label27 = (gcnew System::Windows::Forms::Label());
392             this->label26 = (gcnew System::Windows::Forms::Label());
393             this->label18 = (gcnew System::Windows::Forms::Label());
394             this->label17 = (gcnew System::Windows::Forms::Label());
395             this->label16 = (gcnew System::Windows::Forms::Label());
396             this->label11 = (gcnew System::Windows::Forms::Label());
397             this->label4 = (gcnew System::Windows::Forms::Label());
398             this->label3 = (gcnew System::Windows::Forms::Label());
399             this->label2 = (gcnew System::Windows::Forms::Label());
400             this->checkedListBox1 = (gcnew
… System::Windows::Forms::CheckedListBox());
401             this->TimerStart = (gcnew System::Windows::Forms::Button());
402             this->groupBox2 = (gcnew
… System::Windows::Forms::GroupBox());
```

```
403             this->label36 = (gcnew System::Windows::Forms::Label());
404             this->TimeMin = (gcnew System::Windows::Forms::Label());
405             this->TimeSec = (gcnew System::Windows::Forms::Label());
406             this->timer2 = (gcnew
  … System::Windows::Forms::Timer(this->components));
407             this->groupBox6 = (gcnew
  … System::Windows::Forms::GroupBox());
408             this->label24 = (gcnew System::Windows::Forms::Label());
409             this->label23 = (gcnew System::Windows::Forms::Label());
410             this->label22 = (gcnew System::Windows::Forms::Label());
411             this->turner1 = (gcnew System::Windows::Forms::Label());
412             this->shapeContainer1 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::ShapeContainer());
413             this->ovalShape6 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::OvalShape());
414             this->ovalShape5 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::OvalShape());
415             this->ovalShape4 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::OvalShape());
416             this->ovalShape3 = (gcnew
  … Microsoft::VisualBasic::PowerPacks::OvalShape());
417             this->statusStrip1 = (gcnew
  … System::Windows::Forms::StatusStrip());
418             this->PolarTest = (gcnew
  … System::Windows::Forms::ToolStripStatusLabel());
419             this->SerialLabel2 = (gcnew
  … System::Windows::Forms::ToolStripStatusLabel());
420             this->ToolPanel->SuspendLayout();
421             this->panel1->SuspendLayout();
422             this->groupBox1->SuspendLayout();
423             this->Serial->SuspendLayout();
424             this->groupBox4->SuspendLayout();
425             this->groupBox3->SuspendLayout();
426             this->ThrusterToggle->SuspendLayout();
427             this->Main->SuspendLayout();
428             (cli::safe_cast<System::ComponentModel::ISupportInitialize^
  … >(this->TurnerBar))->BeginInit();
429             (cli::safe_cast<System::ComponentModel::ISupportInitialize^
  … >(this->PitchRollBar))->BeginInit();
```

```
430             (cli::safe_cast<System::ComponentModel::ISupportInitialize^
... >(this->StrafeSpeedCTL))->BeginInit();
431             (cli::safe_cast<System::ComponentModel::ISupportInitialize^
... >(this->VertLimiter))->BeginInit();
432             (cli::safe_cast<System::ComponentModel::ISupportInitialize^
... >(this->HorizLimiter))->BeginInit();
433             this->tabControl1->SuspendLayout();
434             this->Status->SuspendLayout();
435             (cli::safe_cast<System::ComponentModel::ISupportInitialize^
... >(this->dataGridView1))->BeginInit();
436             this->Debug->SuspendLayout();
437             this->groupBox5->SuspendLayout();
438             this->groupBox2->SuspendLayout();
439             this->groupBox6->SuspendLayout();
440             this->statusStrip1->SuspendLayout();
441             this->SuspendLayout();
442             //
443             // timer1
444             //
445             this->timer1->Enabled = true;
446             this->timer1->Interval = 10;
447             this->timer1->Tick += gcnew System::EventHandler(this,
... &BaseStation::timer1_Tick);
448             //
449             // serialPort1
450             //
451             this->serialPort1->BaudRate = 115200;
452             this->serialPort1->PortName = L"COM5";
453             this->serialPort1->ReadBufferSize = 256;
454             this->serialPort1->WriteBufferSize = 256;
455             this->serialPort1->DataReceived += gcnew
... System::IO::Ports::SerialDataReceivedEventHandler(this,
... &BaseStation::serialPort1_DataReceived);
456             //
457             // estopDisp
458             //
459             this->estopDisp->AutoSize = true;
460             this->estopDisp->Font = (gcnew
... System::Drawing::Font(L"Microsoft Sans Serif", 25,
```

```
460… System::Drawing::FontStyle::Regular,
…  System::Drawing::GraphicsUnit::Point,
461                 static_cast<System::Byte>(0)));
462             this->estopDisp->ForeColor = System::Drawing::Color::Red;
463             this->estopDisp->Location = System::Drawing::Point(12, 292);
464             this->estopDisp->Name = L"estopDisp";
465             this->estopDisp->Size = System::Drawing::Size(0, 39);
466             this->estopDisp->TabIndex = 5;
467             //
468             // recValues
469             //
470             this->recValues->AutoSize = true;
471             this->recValues->Font = (gcnew
…  System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
…  System::Drawing::FontStyle::Regular,
…  System::Drawing::GraphicsUnit::Point,
472                 static_cast<System::Byte>(0)));
473             this->recValues->Location = System::Drawing::Point(86, 24);
474             this->recValues->Name = L"recValues";
475             this->recValues->Size = System::Drawing::Size(56, 16);
476             this->recValues->TabIndex = 6;
477             this->recValues->Text = L"Serial In";
478             //
479             // CPortBox
480             //
481             this->CPortBox->FormattingEnabled = true;
482             this->CPortBox->Items->AddRange(gcnew cli::array<
…  System::Object^  >(11) {L"COM0", L"COM1", L"COM2", L"COM3", L"COM4",
…  L"COM5",
483                 L"COM6", L"COM7", L"COM8", L"COM9", L"COM10"});
484             this->CPortBox->Location = System::Drawing::Point(356, 12);
485             this->CPortBox->Name = L"CPortBox";
486             this->CPortBox->Size = System::Drawing::Size(134, 21);
487             this->CPortBox->TabIndex = 0;
488             this->CPortBox->Text = L"COM5";
489             //
490             // serialLabel
491             //
492             this->serialLabel->AutoSize = true;
```

```
493             this->serialLabel->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
494                 static_cast<System::Byte>(0)));
495             this->serialLabel->Location = System::Drawing::Point(86,
… 52);
496             this->serialLabel->Name = L"serialLabel";
497             this->serialLabel->Size = System::Drawing::Size(66, 16);
498             this->serialLabel->TabIndex = 8;
499             this->serialLabel->Text = L"Serial Out";
500             //
501             // button2
502             //
503             this->button2->Location = System::Drawing::Point(356, 82);
504             this->button2->Name = L"button2";
505             this->button2->Size = System::Drawing::Size(134, 35);
506             this->button2->TabIndex = 9;
507             this->button2->Text = L"Toggle Serial";
508             this->button2->UseVisualStyleBackColor = true;
509             this->button2->Click += gcnew System::EventHandler(this,
… &BaseStation::button2_Click);
510             //
511             // ToolPanel
512             //
513             this->ToolPanel->BackColor =
… System::Drawing::Color::Transparent;
514             this->ToolPanel->Controls->Add(this->ManipStatus);
515             this->ToolPanel->Controls->Add(this->label1);
516             this->ToolPanel->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
517                 static_cast<System::Byte>(0)));
518             this->ToolPanel->Location = System::Drawing::Point(534,
… 334);
519             this->ToolPanel->Name = L"ToolPanel";
520             this->ToolPanel->Size = System::Drawing::Size(143, 43);
521             this->ToolPanel->TabIndex = 0;
```

```
522             this->ToolPanel->TabStop = false;
523             this->ToolPanel->Text = L"Tool Status";
524             //
525             // ManipStatus
526             //
527             this->ManipStatus->AutoSize = true;
528             this->ManipStatus->Location = System::Drawing::Point(77,
    … 20);
529             this->ManipStatus->Name = L"ManipStatus";
530             this->ManipStatus->Size = System::Drawing::Size(51, 16);
531             this->ManipStatus->TabIndex = 3;
532             this->ManipStatus->Text = L"Closed";
533             //
534             // label1
535             //
536             this->label1->AutoSize = true;
537             this->label1->BackColor =
    … System::Drawing::Color::Transparent;
538             this->label1->Location = System::Drawing::Point(2, 19);
539             this->label1->Name = L"label1";
540             this->label1->Size = System::Drawing::Size(81, 16);
541             this->label1->TabIndex = 0;
542             this->label1->Text = L"Manipulator:";
543             //
544             // RFThrust
545             //
546             this->RFThrust->AutoSize = true;
547             this->RFThrust->BackColor =
    … System::Drawing::SystemColors::ControlDark;
548             this->RFThrust->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 12,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
549                 static_cast<System::Byte>(0)));
550             this->RFThrust->ForeColor =
    … System::Drawing::SystemColors::ActiveCaptionText;
551             this->RFThrust->Location = System::Drawing::Point(317, 25);
552             this->RFThrust->Name = L"RFThrust";
553             this->RFThrust->Size = System::Drawing::Size(18, 20);
```

```
554             this->RFThrust->TabIndex = 17;
555             this->RFThrust->Text = L"0";
556             //
557             // LFThrust
558             //
559             this->LFThrust->AutoSize = true;
560             this->LFThrust->BackColor =
… System::Drawing::SystemColors::ControlDark;
561             this->LFThrust->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 12,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
562                 static_cast<System::Byte>(0)));
563             this->LFThrust->ForeColor =
… System::Drawing::SystemColors::ActiveCaptionText;
564             this->LFThrust->Location = System::Drawing::Point(40, 25);
565             this->LFThrust->Name = L"LFThrust";
566             this->LFThrust->Size = System::Drawing::Size(18, 20);
567             this->LFThrust->TabIndex = 16;
568             this->LFThrust->Text = L"0";
569             //
570             // LBThrust
571             //
572             this->LBThrust->AutoSize = true;
573             this->LBThrust->BackColor =
… System::Drawing::SystemColors::ControlDark;
574             this->LBThrust->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 12,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
575                 static_cast<System::Byte>(0)));
576             this->LBThrust->ForeColor =
… System::Drawing::SystemColors::ActiveCaptionText;
577             this->LBThrust->Location = System::Drawing::Point(40, 261);
578             this->LBThrust->Name = L"LBThrust";
579             this->LBThrust->Size = System::Drawing::Size(18, 20);
580             this->LBThrust->TabIndex = 15;
581             this->LBThrust->Text = L"0";
582             //
```

```
583             // RBThrust
584             //
585             this->RBThrust->AutoSize = true;
586             this->RBThrust->BackColor =
…   System::Drawing::SystemColors::ControlDark;
587             this->RBThrust->Font = (gcnew
…   System::Drawing::Font(L"Microsoft Sans Serif", 12,
…   System::Drawing::FontStyle::Regular,
…   System::Drawing::GraphicsUnit::Point,
588                 static_cast<System::Byte>(0)));
589             this->RBThrust->ForeColor =
…   System::Drawing::SystemColors::ActiveCaptionText;
590             this->RBThrust->Location = System::Drawing::Point(317, 261);
591             this->RBThrust->Name = L"RBThrust";
592             this->RBThrust->Size = System::Drawing::Size(18, 20);
593             this->RBThrust->TabIndex = 14;
594             this->RBThrust->Text = L"0";
595             //
596             // RBVThrust
597             //
598             this->RBVThrust->AutoSize = true;
599             this->RBVThrust->BackColor =
…   System::Drawing::SystemColors::ControlDark;
600             this->RBVThrust->Font = (gcnew
…   System::Drawing::Font(L"Microsoft Sans Serif", 12,
…   System::Drawing::FontStyle::Regular,
…   System::Drawing::GraphicsUnit::Point,
601                 static_cast<System::Byte>(0)));
602             this->RBVThrust->ForeColor =
…   System::Drawing::SystemColors::ActiveCaptionText;
603             this->RBVThrust->Location = System::Drawing::Point(273,
…   199);
604             this->RBVThrust->Name = L"RBVThrust";
605             this->RBVThrust->Size = System::Drawing::Size(18, 20);
606             this->RBVThrust->TabIndex = 13;
607             this->RBVThrust->Text = L"0";
608             //
609             // LBVThrust
610             //
```

```
611             this->LBVThrust->AutoSize = true;
612             this->LBVThrust->BackColor =
…   System::Drawing::SystemColors::ControlDark;
613             this->LBVThrust->Font = (gcnew
…   System::Drawing::Font(L"Microsoft Sans Serif", 12,
…   System::Drawing::FontStyle::Regular,
…   System::Drawing::GraphicsUnit::Point,
614                 static_cast<System::Byte>(0)));
615             this->LBVThrust->ForeColor =
…   System::Drawing::SystemColors::ActiveCaptionText;
616             this->LBVThrust->Location = System::Drawing::Point(82, 199);
617             this->LBVThrust->Name = L"LBVThrust";
618             this->LBVThrust->Size = System::Drawing::Size(18, 20);
619             this->LBVThrust->TabIndex = 12;
620             this->LBVThrust->Text = L"0";
621             //
622             // RFVThrust
623             //
624             this->RFVThrust->AutoSize = true;
625             this->RFVThrust->BackColor =
…   System::Drawing::SystemColors::ControlDark;
626             this->RFVThrust->Font = (gcnew
…   System::Drawing::Font(L"Microsoft Sans Serif", 12,
…   System::Drawing::FontStyle::Regular,
…   System::Drawing::GraphicsUnit::Point,
627                 static_cast<System::Byte>(0)));
628             this->RFVThrust->ForeColor =
…   System::Drawing::SystemColors::ActiveCaptionText;
629             this->RFVThrust->Location = System::Drawing::Point(273, 86);
630             this->RFVThrust->Name = L"RFVThrust";
631             this->RFVThrust->Size = System::Drawing::Size(18, 20);
632             this->RFVThrust->TabIndex = 11;
633             this->RFVThrust->Text = L"0";
634             //
635             // LFVThrust
636             //
637             this->LFVThrust->AutoSize = true;
638             this->LFVThrust->BackColor =
…   System::Drawing::SystemColors::ControlDark;
```

```
639             this->LFVThrust->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 12,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
640                 static_cast<System::Byte>(0)));
641             this->LFVThrust->ForeColor =
… System::Drawing::SystemColors::ActiveCaptionText;
642             this->LFVThrust->Location = System::Drawing::Point(82, 86);
643             this->LFVThrust->Name = L"LFVThrust";
644             this->LFVThrust->Size = System::Drawing::Size(18, 20);
645             this->LFVThrust->TabIndex = 10;
646             this->LFVThrust->Text = L"0";
647             //
648             // panel1
649             //
650             this->panel1->BackgroundImage =
… (cli::safe_cast<System::Drawing::Image^
… >(resources->GetObject(L"panel1.BackgroundImage")));
651             this->panel1->BackgroundImageLayout =
… System::Windows::Forms::ImageLayout::Center;
652             this->panel1->Controls->Add(this->LFThrust);
653             this->panel1->Controls->Add(this->RBThrust);
654             this->panel1->Controls->Add(this->LBThrust);
655             this->panel1->Controls->Add(this->RBVThrust);
656             this->panel1->Controls->Add(this->RFThrust);
657             this->panel1->Controls->Add(this->LBVThrust);
658             this->panel1->Controls->Add(this->LFVThrust);
659             this->panel1->Controls->Add(this->RFVThrust);
660             this->panel1->Location = System::Drawing::Point(683, 12);
661             this->panel1->Name = L"panel1";
662             this->panel1->Size = System::Drawing::Size(374, 307);
663             this->panel1->TabIndex = 1;
664             //
665             // groupBox1
666             //
667             this->groupBox1->BackColor =
… System::Drawing::Color::Transparent;
668             this->groupBox1->Controls->Add(this->label21);
669             this->groupBox1->Controls->Add(this->label20);
```

```
670             this->groupBox1->Controls->Add(this->label19);
671             this->groupBox1->Controls->Add(this->label10);
672             this->groupBox1->Controls->Add(this->label9);
673             this->groupBox1->Controls->Add(this->shapeContainer2);
674             this->groupBox1->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
675                 static_cast<System::Byte>(0)));
676             this->groupBox1->Location = System::Drawing::Point(683,
… 334);
677             this->groupBox1->Name = L"groupBox1";
678             this->groupBox1->Size = System::Drawing::Size(373, 269);
679             this->groupBox1->TabIndex = 10;
680             this->groupBox1->TabStop = false;
681             this->groupBox1->Text = L"Roll and Pitch";
682             //
683             // label21
684             //
685             this->label21->AutoSize = true;
686             this->label21->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 10,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
687                 static_cast<System::Byte>(0)));
688             this->label21->Location = System::Drawing::Point(349, 140);
689             this->label21->Name = L"label21";
690             this->label21->Size = System::Drawing::Size(16, 17);
691             this->label21->TabIndex = 3;
692             this->label21->Text = L"F";
693             //
694             // label20
695             //
696             this->label20->AutoSize = true;
697             this->label20->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 10,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
698                 static_cast<System::Byte>(0)));
```

```
699             this->label20->Location = System::Drawing::Point(179, 140);
700             this->label20->Name = L"label20";
701             this->label20->Size = System::Drawing::Size(18, 17);
702             this->label20->TabIndex = 3;
703             this->label20->Text = L"R";
704             //
705             // label19
706             //
707             this->label19->AutoSize = true;
708             this->label19->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 10,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
709                 static_cast<System::Byte>(0)));
710             this->label19->Location = System::Drawing::Point(8, 141);
711             this->label19->Name = L"label19";
712             this->label19->Size = System::Drawing::Size(16, 17);
713             this->label19->TabIndex = 3;
714             this->label19->Text = L"L";
715             //
716             // label10
717             //
718             this->label10->AutoSize = true;
719             this->label10->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 15.75F,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
720                 static_cast<System::Byte>(0)));
721             this->label10->Location = System::Drawing::Point(244, 40);
722             this->label10->Name = L"label10";
723             this->label10->Size = System::Drawing::Size(60, 25);
724             this->label10->TabIndex = 2;
725             this->label10->Text = L"Pitch";
726             //
727             // label9
728             //
729             this->label9->AutoSize = true;
730             this->label9->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 15.75F,
```

```
730… System::Drawing::FontStyle::Regular,
…  System::Drawing::GraphicsUnit::Point,
731                 static_cast<System::Byte>(0)));
732             this->label9->Location = System::Drawing::Point(63, 40);
733             this->label9->Name = L"label9";
734             this->label9->Size = System::Drawing::Size(49, 25);
735             this->label9->TabIndex = 1;
736             this->label9->Text = L"Roll";
737             //
738             // shapeContainer2
739             //
740             this->shapeContainer2->Location = System::Drawing::Point(3,
…  18);
741             this->shapeContainer2->Margin =
…  System::Windows::Forms::Padding(0);
742             this->shapeContainer2->Name = L"shapeContainer2";
743             this->shapeContainer2->Shapes->AddRange(gcnew cli::array<
…  Microsoft::VisualBasic::PowerPacks::Shape^  >(8) {this->lineShape6,
744                 this->lineShape5, this->lineShape4, this->lineShape3,
…  this->Pitch, this->Roll, this->ovalShape1, this->ovalShape2});
745             this->shapeContainer2->Size = System::Drawing::Size(367,
…  248);
746             this->shapeContainer2->TabIndex = 0;
747             this->shapeContainer2->TabStop = false;
748             //
749             // lineShape6
750             //
751             this->lineShape6->BorderColor =
…  System::Drawing::Color::White;
752             this->lineShape6->Name = L"lineShape6";
753             this->lineShape6->X1 = 273;
754             this->lineShape6->X2 = 273;
755             this->lineShape6->Y1 = 65;
756             this->lineShape6->Y2 = 195;
757             //
758             // lineShape5
759             //
760             this->lineShape5->BorderColor =
…  System::Drawing::Color::White;
```

```
761             this->lineShape5->Name = L"lineShape5";
762             this->lineShape5->X1 = 95;
763             this->lineShape5->X2 = 95;
764             this->lineShape5->Y1 = 65;
765             this->lineShape5->Y2 = 195;
766             //
767             // lineShape4
768             //
769             this->lineShape4->BorderColor =
    System::Drawing::Color::White;
770             this->lineShape4->Name = L"lineShape4";
771             this->lineShape4->X1 = 209;
772             this->lineShape4->X2 = 339;
773             this->lineShape4->Y1 = 130;
774             this->lineShape4->Y2 = 130;
775             //
776             // lineShape3
777             //
778             this->lineShape3->BorderColor =
    System::Drawing::Color::White;
779             this->lineShape3->Name = L"lineShape3";
780             this->lineShape3->X1 = 30;
781             this->lineShape3->X2 = 160;
782             this->lineShape3->Y1 = 130;
783             this->lineShape3->Y2 = 130;
784             //
785             // Pitch
786             //
787             this->Pitch->BorderColor =
    System::Drawing::Color::DarkOrange;
788             this->Pitch->BorderWidth = 4;
789             this->Pitch->Name = L"Pitch";
790             this->Pitch->X1 = 214;
791             this->Pitch->X2 = 334;
792             this->Pitch->Y1 = 130;
793             this->Pitch->Y2 = 130;
794             //
795             // Roll
796             //
```

```
797             this->Roll->BorderColor =
... System::Drawing::Color::DarkOrange;
798             this->Roll->BorderWidth = 4;
799             this->Roll->Name = L"Roll";
800             this->Roll->X1 = 35;
801             this->Roll->X2 = 155;
802             this->Roll->Y1 = 130;
803             this->Roll->Y2 = 130;
804             //
805             // ovalShape1
806             //
807             this->ovalShape1->BackColor =
... System::Drawing::Color::DimGray;
808             this->ovalShape1->BackStyle =
... Microsoft::VisualBasic::PowerPacks::BackStyle::Opaque;
809             this->ovalShape1->BorderColor =
... System::Drawing::Color::White;
810             this->ovalShape1->BorderWidth = 5;
811             this->ovalShape1->Location = System::Drawing::Point(30, 65);
812             this->ovalShape1->Name = L"ovalShape1";
813             this->ovalShape1->Size = System::Drawing::Size(130, 130);
814             //
815             // ovalShape2
816             //
817             this->ovalShape2->BackColor =
... System::Drawing::Color::DimGray;
818             this->ovalShape2->BackStyle =
... Microsoft::VisualBasic::PowerPacks::BackStyle::Opaque;
819             this->ovalShape2->BorderColor =
... System::Drawing::Color::White;
820             this->ovalShape2->BorderWidth = 5;
821             this->ovalShape2->Location = System::Drawing::Point(209,
... 65);
822             this->ovalShape2->Name = L"ovalShape2";
823             this->ovalShape2->Size = System::Drawing::Size(130, 130);
824             //
825             // Serial
826             //
827             this->Serial->Controls->Add(this->groupBox4);
```

```
828             this->Serial->Controls->Add(this->groupBox3);
829             this->Serial->Location = System::Drawing::Point(4, 22);
830             this->Serial->Name = L"Serial";
831             this->Serial->Padding = System::Windows::Forms::Padding(3);
832             this->Serial->Size = System::Drawing::Size(508, 243);
833             this->Serial->TabIndex = 3;
834             this->Serial->Text = L"Serial and Controller";
835             this->Serial->UseVisualStyleBackColor = true;
836             //
837             // groupBox4
838             //
839             this->groupBox4->Controls->Add(this->ControlCheck);
840             this->groupBox4->Controls->Add(this->C2Status);
841             this->groupBox4->Controls->Add(this->C1Status);
842             this->groupBox4->Controls->Add(this->label14);
843             this->groupBox4->Controls->Add(this->label15);
844             this->groupBox4->Location = System::Drawing::Point(7, 140);
845             this->groupBox4->Name = L"groupBox4";
846             this->groupBox4->Size = System::Drawing::Size(495, 94);
847             this->groupBox4->TabIndex = 13;
848             this->groupBox4->TabStop = false;
849             this->groupBox4->Text = L"Controller";
850             //
851             // ControlCheck
852             //
853             this->ControlCheck->Location = System::Drawing::Point(355,
    … 26);
854             this->ControlCheck->Name = L"ControlCheck";
855             this->ControlCheck->Size = System::Drawing::Size(134, 45);
856             this->ControlCheck->TabIndex = 12;
857             this->ControlCheck->Text = L"Check for Controllers";
858             this->ControlCheck->UseVisualStyleBackColor = true;
859             this->ControlCheck->Click += gcnew
    … System::EventHandler(this, &BaseStation::ControlCheck_Click);
860             //
861             // C2Status
862             //
863             this->C2Status->AutoSize = true;
864             this->C2Status->Font = (gcnew
```

```
System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
            this->C2Status->Location = System::Drawing::Point(137, 55);
            this->C2Status->Name = L"C2Status";
            this->C2Status->Size = System::Drawing::Size(75, 16);
            this->C2Status->TabIndex = 3;
            this->C2Status->Text = L"Controller 2";
            //
            // C1Status
            //
            this->C1Status->AutoSize = true;
            this->C1Status->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
            this->C1Status->Location = System::Drawing::Point(137, 26);
            this->C1Status->Name = L"C1Status";
            this->C1Status->Size = System::Drawing::Size(75, 16);
            this->C1Status->TabIndex = 2;
            this->C1Status->Text = L"Controller 1";
            //
            // label14
            //
            this->label14->AutoSize = true;
            this->label14->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
                static_cast<System::Byte>(0)));
            this->label14->Location = System::Drawing::Point(10, 55);
            this->label14->Name = L"label14";
            this->label14->Size = System::Drawing::Size(121, 16);
            this->label14->TabIndex = 1;
            this->label14->Text = L"Controller 2 Status :";
            //
            // label15
```

```
895             //
896             this->label15->AutoSize = true;
897             this->label15->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
898                 static_cast<System::Byte>(0)));
899             this->label15->Location = System::Drawing::Point(10, 26);
900             this->label15->Name = L"label15";
901             this->label15->Size = System::Drawing::Size(121, 16);
902             this->label15->TabIndex = 0;
903             this->label15->Text = L"Controller 1 Status :";
904             //
905             // groupBox3
906             //
907             this->groupBox3->Controls->Add(this->label13);
908             this->groupBox3->Controls->Add(this->label12);
909             this->groupBox3->Controls->Add(this->button2);
910             this->groupBox3->Controls->Add(this->serialLabel);
911             this->groupBox3->Controls->Add(this->recValues);
912             this->groupBox3->Controls->Add(this->CPortBox);
913             this->groupBox3->Location = System::Drawing::Point(6, 6);
914             this->groupBox3->Name = L"groupBox3";
915             this->groupBox3->Size = System::Drawing::Size(496, 128);
916             this->groupBox3->TabIndex = 12;
917             this->groupBox3->TabStop = false;
918             this->groupBox3->Text = L"Serial";
919             //
920             // label13
921             //
922             this->label13->AutoSize = true;
923             this->label13->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
924                 static_cast<System::Byte>(0)));
925             this->label13->Location = System::Drawing::Point(11, 52);
926             this->label13->Name = L"label13";
927             this->label13->Size = System::Drawing::Size(69, 16);
```

```
928             this->label13->TabIndex = 11;
929             this->label13->Text = L"Serial Out:";
930             //
931             // label12
932             //
933             this->label12->AutoSize = true;
934             this->label12->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
935                 static_cast<System::Byte>(0)));
936             this->label12->Location = System::Drawing::Point(19, 24);
937             this->label12->Name = L"label12";
938             this->label12->Size = System::Drawing::Size(59, 16);
939             this->label12->TabIndex = 10;
940             this->label12->Text = L"Serial in:";
941             //
942             // ThrusterToggle
943             //
944             this->ThrusterToggle->AutoScroll = true;
945             this->ThrusterToggle->Controls->Add(this->DisableLBV);
946             this->ThrusterToggle->Controls->Add(this->DisableRBV);
947             this->ThrusterToggle->Controls->Add(this->DisableRFV);
948             this->ThrusterToggle->Controls->Add(this->DisableLFV);
949             this->ThrusterToggle->Controls->Add(this->DisableLB);
950             this->ThrusterToggle->Controls->Add(this->DisableRB);
951             this->ThrusterToggle->Controls->Add(this->DisableRF);
952             this->ThrusterToggle->Controls->Add(this->DisableLF);
953             this->ThrusterToggle->Location = System::Drawing::Point(4,
… 22);
954             this->ThrusterToggle->Name = L"ThrusterToggle";
955             this->ThrusterToggle->Size = System::Drawing::Size(508,
… 243);
956             this->ThrusterToggle->TabIndex = 2;
957             this->ThrusterToggle->Text = L"Toggle Thrusters";
958             this->ThrusterToggle->UseVisualStyleBackColor = true;
959             //
960             // DisableLBV
961             //
```

```
962             this->DisableLBV->AutoSize = true;
963             this->DisableLBV->Location = System::Drawing::Point(113,
  … 153);
964             this->DisableLBV->Name = L"DisableLBV";
965             this->DisableLBV->Size = System::Drawing::Size(132, 17);
966             this->DisableLBV->TabIndex = 7;
967             this->DisableLBV->Text = L"Disable Left Back Vert";
968             this->DisableLBV->UseVisualStyleBackColor = true;
969             //
970             // DisableRBV
971             //
972             this->DisableRBV->AutoSize = true;
973             this->DisableRBV->Location = System::Drawing::Point(254,
  … 153);
974             this->DisableRBV->Name = L"DisableRBV";
975             this->DisableRBV->Size = System::Drawing::Size(139, 17);
976             this->DisableRBV->TabIndex = 6;
977             this->DisableRBV->Text = L"Disable Right Back Vert";
978             this->DisableRBV->UseVisualStyleBackColor = true;
979             //
980             // DisableRFV
981             //
982             this->DisableRFV->AutoSize = true;
983             this->DisableRFV->Location = System::Drawing::Point(254,
  … 83);
984             this->DisableRFV->Name = L"DisableRFV";
985             this->DisableRFV->Size = System::Drawing::Size(138, 17);
986             this->DisableRFV->TabIndex = 5;
987             this->DisableRFV->Text = L"Disable Right Front Vert";
988             this->DisableRFV->UseVisualStyleBackColor = true;
989             //
990             // DisableLFV
991             //
992             this->DisableLFV->AutoSize = true;
993             this->DisableLFV->Location = System::Drawing::Point(113,
  … 83);
994             this->DisableLFV->Name = L"DisableLFV";
995             this->DisableLFV->Size = System::Drawing::Size(131, 17);
996             this->DisableLFV->TabIndex = 4;
```

```
997             this->DisableLFV->Text = L"Disable Left Front Vert";
998             this->DisableLFV->UseVisualStyleBackColor = true;
999             //
1000            // DisableLB
1001            //
1002            this->DisableLB->AutoSize = true;
1003            this->DisableLB->Location = System::Drawing::Point(33, 192);
1004            this->DisableLB->Name = L"DisableLB";
1005            this->DisableLB->Size = System::Drawing::Size(110, 17);
1006            this->DisableLB->TabIndex = 3;
1007            this->DisableLB->Text = L"Disable Left Back";
1008            this->DisableLB->UseVisualStyleBackColor = true;
1009            //
1010            // DisableRB
1011            //
1012            this->DisableRB->AutoSize = true;
1013            this->DisableRB->Location = System::Drawing::Point(353,
    … 192);
1014            this->DisableRB->Name = L"DisableRB";
1015            this->DisableRB->Size = System::Drawing::Size(117, 17);
1016            this->DisableRB->TabIndex = 2;
1017            this->DisableRB->Text = L"Disable Right Back";
1018            this->DisableRB->UseVisualStyleBackColor = true;
1019            //
1020            // DisableRF
1021            //
1022            this->DisableRF->AutoSize = true;
1023            this->DisableRF->Location = System::Drawing::Point(353, 45);
1024            this->DisableRF->Name = L"DisableRF";
1025            this->DisableRF->Size = System::Drawing::Size(116, 17);
1026            this->DisableRF->TabIndex = 1;
1027            this->DisableRF->Text = L"Disable Right Front";
1028            this->DisableRF->UseVisualStyleBackColor = true;
1029            //
1030            // DisableLF
1031            //
1032            this->DisableLF->AutoSize = true;
1033            this->DisableLF->Location = System::Drawing::Point(39, 45);
1034            this->DisableLF->Name = L"DisableLF";
```

```
1035            this->DisableLF->Size = System::Drawing::Size(109, 17);
1036            this->DisableLF->TabIndex = 0;
1037            this->DisableLF->Text = L"Disable Left Front";
1038            this->DisableLF->UseVisualStyleBackColor = true;
1039            //
1040            // Main
1041            //
1042            this->Main->Controls->Add(this->label25);
1043            this->Main->Controls->Add(this->TurnerBar);
1044            this->Main->Controls->Add(this->label8);
1045            this->Main->Controls->Add(this->label7);
1046            this->Main->Controls->Add(this->label6);
1047            this->Main->Controls->Add(this->label5);
1048            this->Main->Controls->Add(this->PitchRollBar);
1049            this->Main->Controls->Add(this->StrafeSpeedCTL);
1050            this->Main->Controls->Add(this->VertLimiter);
1051            this->Main->Controls->Add(this->HorizLimiter);
1052            this->Main->Location = System::Drawing::Point(4, 22);
1053            this->Main->Name = L"Main";
1054            this->Main->Padding = System::Windows::Forms::Padding(3);
1055            this->Main->Size = System::Drawing::Size(508, 243);
1056            this->Main->TabIndex = 0;
1057            this->Main->Text = L"Main";
1058            this->Main->UseVisualStyleBackColor = true;
1059            //
1060            // label25
1061            //
1062            this->label25->AutoSize = true;
1063            this->label25->Font = (gcnew
   … System::Drawing::Font(L"Microsoft Sans Serif", 10,
   … System::Drawing::FontStyle::Regular,
   … System::Drawing::GraphicsUnit::Point,
1064                static_cast<System::Byte>(0)));
1065            this->label25->Location = System::Drawing::Point(9, 201);
1066            this->label25->Name = L"label25";
1067            this->label25->Size = System::Drawing::Size(96, 17);
1068            this->label25->TabIndex = 9;
1069            this->label25->Text = L"Turner Speed";
1070            //
```

```
1071            // TurnerBar
1072            //
1073            this->TurnerBar->BackColor =
    … System::Drawing::SystemColors::Window;
1074            this->TurnerBar->LargeChange = 1;
1075            this->TurnerBar->Location = System::Drawing::Point(164,
    … 189);
1076            this->TurnerBar->Minimum = 1;
1077            this->TurnerBar->Name = L"TurnerBar";
1078            this->TurnerBar->Size = System::Drawing::Size(338, 45);
1079            this->TurnerBar->TabIndex = 8;
1080            this->TurnerBar->TickStyle =
    … System::Windows::Forms::TickStyle::Both;
1081            this->TurnerBar->Value = 8;
1082            //
1083            // label8
1084            //
1085            this->label8->AutoSize = true;
1086            this->label8->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 10,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1087                static_cast<System::Byte>(0)));
1088            this->label8->Location = System::Drawing::Point(9, 155);
1089            this->label8->Name = L"label8";
1090            this->label8->Size = System::Drawing::Size(107, 17);
1091            this->label8->TabIndex = 7;
1092            this->label8->Text = L"Pitch/Roll Effect";
1093            //
1094            // label7
1095            //
1096            this->label7->AutoSize = true;
1097            this->label7->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 10,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1098                static_cast<System::Byte>(0)));
1099            this->label7->Location = System::Drawing::Point(9, 111);
1100            this->label7->Name = L"label7";
```

```
1101            this->label7->Size = System::Drawing::Size(140, 17);
1102            this->label7->TabIndex = 6;
1103            this->label7->Text = L"Strafe Speed Control";
1104            //
1105            // label6
1106            //
1107            this->label6->AutoSize = true;
1108            this->label6->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 10,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1109                static_cast<System::Byte>(0)));
1110            this->label6->Location = System::Drawing::Point(9, 66);
1111            this->label6->Name = L"label6";
1112            this->label6->Size = System::Drawing::Size(149, 17);
1113            this->label6->TabIndex = 5;
1114            this->label6->Text = L"Vertical Speed Control";
1115            //
1116            // label5
1117            //
1118            this->label5->AutoSize = true;
1119            this->label5->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 10,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1120                static_cast<System::Byte>(0)));
1121            this->label5->Location = System::Drawing::Point(9, 20);
1122            this->label5->Name = L"label5";
1123            this->label5->Size = System::Drawing::Size(166, 17);
1124            this->label5->TabIndex = 4;
1125            this->label5->Text = L"Horizontal Speed Control";
1126            //
1127            // PitchRollBar
1128            //
1129            this->PitchRollBar->BackColor =
    … System::Drawing::SystemColors::Window;
1130            this->PitchRollBar->LargeChange = 1;
1131            this->PitchRollBar->Location = System::Drawing::Point(164,
    … 144);
```

```
1132            this->PitchRollBar->Minimum = 1;
1133            this->PitchRollBar->Name = L"PitchRollBar";
1134            this->PitchRollBar->Size = System::Drawing::Size(338, 45);
1135            this->PitchRollBar->TabIndex = 3;
1136            this->PitchRollBar->TickStyle =
    … System::Windows::Forms::TickStyle::Both;
1137            this->PitchRollBar->Value = 8;
1138            this->PitchRollBar->Scroll += gcnew
    … System::EventHandler(this, &BaseStation::PitchRollBar_Scroll);
1139            //
1140            // StrafeSpeedCTL
1141            //
1142            this->StrafeSpeedCTL->BackColor =
    … System::Drawing::SystemColors::Window;
1143            this->StrafeSpeedCTL->LargeChange = 1;
1144            this->StrafeSpeedCTL->Location = System::Drawing::Point(164,
    … 99);
1145            this->StrafeSpeedCTL->Minimum = 1;
1146            this->StrafeSpeedCTL->Name = L"StrafeSpeedCTL";
1147            this->StrafeSpeedCTL->Size = System::Drawing::Size(338, 45);
1148            this->StrafeSpeedCTL->TabIndex = 2;
1149            this->StrafeSpeedCTL->TickStyle =
    … System::Windows::Forms::TickStyle::Both;
1150            this->StrafeSpeedCTL->Value = 8;
1151            this->StrafeSpeedCTL->Scroll += gcnew
    … System::EventHandler(this, &BaseStation::StrafeSpeedCTL_Scroll);
1152            //
1153            // VertLimiter
1154            //
1155            this->VertLimiter->BackColor =
    … System::Drawing::SystemColors::Window;
1156            this->VertLimiter->LargeChange = 1;
1157            this->VertLimiter->Location = System::Drawing::Point(164,
    … 54);
1158            this->VertLimiter->Minimum = 1;
1159            this->VertLimiter->Name = L"VertLimiter";
1160            this->VertLimiter->Size = System::Drawing::Size(338, 45);
1161            this->VertLimiter->TabIndex = 1;
1162            this->VertLimiter->TickStyle =
```

```
System::Windows::Forms::TickStyle::Both;
            this->VertLimiter->Value = 10;
            this->VertLimiter->Scroll += gcnew
System::EventHandler(this, &BaseStation::VertLimiter_Scroll);
            //
            // HorizLimiter
            //
            this->HorizLimiter->BackColor =
System::Drawing::SystemColors::Window;
            this->HorizLimiter->LargeChange = 1;
            this->HorizLimiter->Location = System::Drawing::Point(164,
9);
            this->HorizLimiter->Minimum = 1;
            this->HorizLimiter->Name = L"HorizLimiter";
            this->HorizLimiter->Size = System::Drawing::Size(338, 45);
            this->HorizLimiter->TabIndex = 0;
            this->HorizLimiter->TickStyle =
System::Windows::Forms::TickStyle::Both;
            this->HorizLimiter->Value = 10;
            this->HorizLimiter->Scroll += gcnew
System::EventHandler(this, &BaseStation::HorizLimiter_Scroll);
            //
            // tabControl1
            //
            this->tabControl1->Controls->Add(this->Main);
            this->tabControl1->Controls->Add(this->ThrusterToggle);
            this->tabControl1->Controls->Add(this->Serial);
            this->tabControl1->Controls->Add(this->Status);
            this->tabControl1->Controls->Add(this->Debug);
            this->tabControl1->Location = System::Drawing::Point(12,
334);
            this->tabControl1->Name = L"tabControl1";
            this->tabControl1->SelectedIndex = 0;
            this->tabControl1->Size = System::Drawing::Size(516, 269);
            this->tabControl1->TabIndex = 0;
            //
            // Status
            //
            this->Status->Controls->Add(this->dataGridView1);
```

```
1195            this->Status->Location = System::Drawing::Point(4, 22);
1196            this->Status->Name = L"Status";
1197            this->Status->Padding = System::Windows::Forms::Padding(3);
1198            this->Status->Size = System::Drawing::Size(508, 243);
1199            this->Status->TabIndex = 5;
1200            this->Status->Text = L"Status";
1201            this->Status->UseVisualStyleBackColor = true;
1202            //
1203            // dataGridView1
1204            //
1205            this->dataGridView1->ColumnHeadersHeightSizeMode =
    … System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::
    … AutoSize;
1206            this->dataGridView1->Columns->AddRange(gcnew cli::array<
    … System::Windows::Forms::DataGridViewColumn^  >(5) {this->Motor,
1207                this->Temperature, this->Voltage, this->Current,
    … this->FaultStatus});
1208            this->dataGridView1->Location = System::Drawing::Point(3,
    … 3);
1209            this->dataGridView1->Name = L"dataGridView1";
1210            this->dataGridView1->Size = System::Drawing::Size(499, 234);
1211            this->dataGridView1->TabIndex = 0;
1212            //
1213            // Motor
1214            //
1215            this->Motor->HeaderText = L"Motor";
1216            this->Motor->Name = L"Motor";
1217            this->Motor->ReadOnly = true;
1218            this->Motor->Width = 50;
1219            //
1220            // Temperature
1221            //
1222            this->Temperature->HeaderText = L"Temperature";
1223            this->Temperature->Name = L"Temperature";
1224            //
1225            // Voltage
1226            //
1227            this->Voltage->HeaderText = L"Voltage";
1228            this->Voltage->Name = L"Voltage";
```

```
1229            //
1230            // Current
1231            //
1232            this->Current->HeaderText = L"Current";
1233            this->Current->Name = L"Current";
1234            //
1235            // FaultStatus
1236            //
1237            this->FaultStatus->HeaderText = L"Fault Status";
1238            this->FaultStatus->Name = L"FaultStatus";
1239            //
1240            // Debug
1241            //
1242            this->Debug->Controls->Add(this->TimerReset);
1243            this->Debug->Controls->Add(this->SerialForce);
1244            this->Debug->Controls->Add(this->SerialDebug);
1245            this->Debug->Location = System::Drawing::Point(4, 22);
1246            this->Debug->Name = L"Debug";
1247            this->Debug->Padding = System::Windows::Forms::Padding(3);
1248            this->Debug->Size = System::Drawing::Size(508, 243);
1249            this->Debug->TabIndex = 4;
1250            this->Debug->Text = L"Debug";
1251            this->Debug->UseVisualStyleBackColor = true;
1252            //
1253            // TimerReset
1254            //
1255            this->TimerReset->Location = System::Drawing::Point(396,
    … 71);
1256            this->TimerReset->Name = L"TimerReset";
1257            this->TimerReset->Size = System::Drawing::Size(106, 66);
1258            this->TimerReset->TabIndex = 2;
1259            this->TimerReset->Text = L"Reset Timer";
1260            this->TimerReset->UseVisualStyleBackColor = true;
1261            this->TimerReset->Click += gcnew System::EventHandler(this,
    … &BaseStation::TimerReset_Click);
1262            //
1263            // SerialForce
1264            //
1265            this->SerialForce->Location = System::Drawing::Point(396,
```

```
6);
            this->SerialForce->Name = L"SerialForce";
            this->SerialForce->Size = System::Drawing::Size(106, 37);
            this->SerialForce->TabIndex = 1;
            this->SerialForce->Text = L"I Dont need to stinkin serial!";
            this->SerialForce->UseVisualStyleBackColor = true;
            this->SerialForce->Click += gcnew System::EventHandler(this,
&BaseStation::SerialForce_Click);
            //
            // SerialDebug
            //
            this->SerialDebug->AutoSize = true;
            this->SerialDebug->Location = System::Drawing::Point(6, 18);
            this->SerialDebug->Name = L"SerialDebug";
            this->SerialDebug->Size = System::Drawing::Size(41, 13);
            this->SerialDebug->TabIndex = 0;
            this->SerialDebug->Text = L"label26";
            //
            // printForm1
            //
            this->printForm1->DocumentName = L"document";
            this->printForm1->Form = this;
            this->printForm1->PrintAction =
System::Drawing::Printing::PrintAction::PrintToPrinter;
            this->printForm1->PrinterSettings =
(cli::safe_cast<System::Drawing::Printing::PrinterSettings^
>(resources->GetObject(L"printForm1.PrinterSettings")));
            this->printForm1->PrintFileName = nullptr;
            //
            // groupBox5
            //
            this->groupBox5->Controls->Add(this->label34);
            this->groupBox5->Controls->Add(this->label33);
            this->groupBox5->Controls->Add(this->label32);
            this->groupBox5->Controls->Add(this->label31);
            this->groupBox5->Controls->Add(this->label30);
            this->groupBox5->Controls->Add(this->label29);
            this->groupBox5->Controls->Add(this->label28);
            this->groupBox5->Controls->Add(this->label27);
```

```
1300            this->groupBox5->Controls->Add(this->label26);
1301            this->groupBox5->Controls->Add(this->label18);
1302            this->groupBox5->Controls->Add(this->label17);
1303            this->groupBox5->Controls->Add(this->label16);
1304            this->groupBox5->Controls->Add(this->label11);
1305            this->groupBox5->Controls->Add(this->label4);
1306            this->groupBox5->Controls->Add(this->label3);
1307            this->groupBox5->Controls->Add(this->label2);
1308            this->groupBox5->Font = (gcnew
  … System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
  … System::Drawing::FontStyle::Regular,
  … System::Drawing::GraphicsUnit::Point,
1309                static_cast<System::Byte>(0)));
1310            this->groupBox5->Location = System::Drawing::Point(534,
  … 381);
1311            this->groupBox5->Name = L"groupBox5";
1312            this->groupBox5->Size = System::Drawing::Size(143, 222);
1313            this->groupBox5->TabIndex = 11;
1314            this->groupBox5->TabStop = false;
1315            this->groupBox5->Text = L"Motor Status";
1316            //
1317            // label34
1318            //
1319            this->label34->AutoSize = true;
1320            this->label34->ForeColor =
  … System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
  … System::Byte>(0)),
  … static_cast<System::Int32>(static_cast<System::Byte>(192)),
1321
  … static_cast<System::Int32>(static_cast<System::Byte>(0)));
1322            this->label34->Location = System::Drawing::Point(101, 200);
1323            this->label34->Name = L"label34";
1324            this->label34->Size = System::Drawing::Size(26, 16);
1325            this->label34->TabIndex = 9;
1326            this->label34->Text = L"OK";
1327            //
1328            // label33
1329            //
1330            this->label33->AutoSize = true;
```

```
1331            this->label33->ForeColor =
    … System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
    … System::Byte>(0)),
    … static_cast<System::Int32>(static_cast<System::Byte>(192)),
1332
    … static_cast<System::Int32>(static_cast<System::Byte>(0)));
1333            this->label33->Location = System::Drawing::Point(101, 174);
1334            this->label33->Name = L"label33";
1335            this->label33->Size = System::Drawing::Size(26, 16);
1336            this->label33->TabIndex = 8;
1337            this->label33->Text = L"OK";
1338            //
1339            // label32
1340            //
1341            this->label32->AutoSize = true;
1342            this->label32->ForeColor =
    … System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
    … System::Byte>(0)),
    … static_cast<System::Int32>(static_cast<System::Byte>(192)),
1343
    … static_cast<System::Int32>(static_cast<System::Byte>(0)));
1344            this->label32->Location = System::Drawing::Point(101, 148);
1345            this->label32->Name = L"label32";
1346            this->label32->Size = System::Drawing::Size(26, 16);
1347            this->label32->TabIndex = 7;
1348            this->label32->Text = L"OK";
1349            //
1350            // label31
1351            //
1352            this->label31->AutoSize = true;
1353            this->label31->ForeColor =
    … System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
    … System::Byte>(0)),
    … static_cast<System::Int32>(static_cast<System::Byte>(192)),
1354
    … static_cast<System::Int32>(static_cast<System::Byte>(0)));
1355            this->label31->Location = System::Drawing::Point(101, 122);
1356            this->label31->Name = L"label31";
1357            this->label31->Size = System::Drawing::Size(26, 16);
```

```
1358            this->label31->TabIndex = 6;
1359            this->label31->Text = L"OK";
1360            //
1361            // label30
1362            //
1363            this->label30->AutoSize = true;
1364            this->label30->ForeColor =
... System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
... System::Byte>(0)),
... static_cast<System::Int32>(static_cast<System::Byte>(192)),
1365
... static_cast<System::Int32>(static_cast<System::Byte>(0)));
1366            this->label30->Location = System::Drawing::Point(101, 96);
1367            this->label30->Name = L"label30";
1368            this->label30->Size = System::Drawing::Size(26, 16);
1369            this->label30->TabIndex = 5;
1370            this->label30->Text = L"OK";
1371            //
1372            // label29
1373            //
1374            this->label29->AutoSize = true;
1375            this->label29->ForeColor =
... System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
... System::Byte>(0)),
... static_cast<System::Int32>(static_cast<System::Byte>(192)),
1376
... static_cast<System::Int32>(static_cast<System::Byte>(0)));
1377            this->label29->Location = System::Drawing::Point(101, 70);
1378            this->label29->Name = L"label29";
1379            this->label29->Size = System::Drawing::Size(26, 16);
1380            this->label29->TabIndex = 4;
1381            this->label29->Text = L"OK";
1382            //
1383            // label28
1384            //
1385            this->label28->AutoSize = true;
1386            this->label28->ForeColor =
... System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
... System::Byte>(0)),
```

```
1386… static_cast<System::Int32>(static_cast<System::Byte>(192)),
1387
   … static_cast<System::Int32>(static_cast<System::Byte>(0)));
1388             this->label28->Location = System::Drawing::Point(101, 44);
1389             this->label28->Name = L"label28";
1390             this->label28->Size = System::Drawing::Size(26, 16);
1391             this->label28->TabIndex = 3;
1392             this->label28->Text = L"OK";
1393             //
1394             // label27
1395             //
1396             this->label27->AutoSize = true;
1397             this->label27->ForeColor =
   … System::Drawing::Color::FromArgb(static_cast<System::Int32>(static_cast<
   … System::Byte>(0)),
   … static_cast<System::Int32>(static_cast<System::Byte>(192)),
1398
   … static_cast<System::Int32>(static_cast<System::Byte>(0)));
1399             this->label27->Location = System::Drawing::Point(101, 18);
1400             this->label27->Name = L"label27";
1401             this->label27->Size = System::Drawing::Size(26, 16);
1402             this->label27->TabIndex = 2;
1403             this->label27->Text = L"OK";
1404             //
1405             // label26
1406             //
1407             this->label26->AutoSize = true;
1408             this->label26->Location = System::Drawing::Point(0, 200);
1409             this->label26->Name = L"label26";
1410             this->label26->Size = System::Drawing::Size(96, 16);
1411             this->label26->TabIndex = 1;
1412             this->label26->Text = L"Left Back Vert :";
1413             //
1414             // label18
1415             //
1416             this->label18->AutoSize = true;
1417             this->label18->Location = System::Drawing::Point(0, 174);
1418             this->label18->Name = L"label18";
1419             this->label18->Size = System::Drawing::Size(106, 16);
```

```
1420            this->label18->TabIndex = 0;
1421            this->label18->Text = L"Right Back Vert :";
1422            //
1423            // label17
1424            //
1425            this->label17->AutoSize = true;
1426            this->label17->Location = System::Drawing::Point(0, 148);
1427            this->label17->Name = L"label17";
1428            this->label17->Size = System::Drawing::Size(105, 16);
1429            this->label17->TabIndex = 0;
1430            this->label17->Text = L"Right Front Vert :";
1431            //
1432            // label16
1433            //
1434            this->label16->AutoSize = true;
1435            this->label16->Location = System::Drawing::Point(0, 122);
1436            this->label16->Name = L"label16";
1437            this->label16->Size = System::Drawing::Size(95, 16);
1438            this->label16->TabIndex = 0;
1439            this->label16->Text = L"Left Front Vert :";
1440            //
1441            // label11
1442            //
1443            this->label11->AutoSize = true;
1444            this->label11->Location = System::Drawing::Point(0, 96);
1445            this->label11->Name = L"label11";
1446            this->label11->Size = System::Drawing::Size(69, 16);
1447            this->label11->TabIndex = 0;
1448            this->label11->Text = L"Left Back :";
1449            //
1450            // label4
1451            //
1452            this->label4->AutoSize = true;
1453            this->label4->Location = System::Drawing::Point(0, 70);
1454            this->label4->Name = L"label4";
1455            this->label4->Size = System::Drawing::Size(79, 16);
1456            this->label4->TabIndex = 0;
1457            this->label4->Text = L"Right Back :";
1458            //
```

```
1459            // label3
1460            //
1461            this->label3->AutoSize = true;
1462            this->label3->Location = System::Drawing::Point(0, 44);
1463            this->label3->Name = L"label3";
1464            this->label3->Size = System::Drawing::Size(78, 16);
1465            this->label3->TabIndex = 0;
1466            this->label3->Text = L"Right Front :";
1467            //
1468            // label2
1469            //
1470            this->label2->AutoSize = true;
1471            this->label2->Location = System::Drawing::Point(0, 18);
1472            this->label2->Name = L"label2";
1473            this->label2->Size = System::Drawing::Size(68, 16);
1474            this->label2->TabIndex = 0;
1475            this->label2->Text = L"Left Front :";
1476            //
1477            // checkedListBox1
1478            //
1479            this->checkedListBox1->FormattingEnabled = true;
1480            this->checkedListBox1->Items->AddRange(gcnew cli::array<
    … System::Object^  >(29) {L"1. Transferring the SIA to the seafloor – 5
    … points",
1481                L"2. Installing the SIA so that it rests completely
    … within the BIA – 15 points", L"3. Removing the CTA from the seafloor – 5
    … points",
1482                L"4. Inserting the CTA into the bulkhead connector on
    … the BIA – 10 points", L"5. Pulling the pin to release the secondary node
    … from the elevator – 10 points",
1483                L"6. Removing the secondary node from the elevator – 5
    … points", L"7. Measuring distance to find the designated location – 15
    … points",
1484                L"8. Installing the secondary node in the designated
    … location on the seafloor – 10 "
1485                L"points", L"9. Adjusting the legs to level the
    … secondary node – 25 points",
1486                L"10. Opening the door of the BIA – 5 points", L"11.
    … Removing the secondary node cable connector from the elevator – 5
```

```
points",
                L"12. Inserting the secondary node cable connector into
the bulkhead connector on t"
                L"he SIA - 10 points", L"", L"1. Designing and
constructing an optical beam transmissometer prior to the compet"
                L"ition - 15 points",
                L"2. Installing the transmissometer in the vent field to
monitor the opacity throug"
                L"h the medium - 10 points", L"3. Detecting the relative
changes in opacity - 10 points",
                L"4. Detecting the relative changes in opacity over five
minutes - 20 points", L"5. Graphing the relative optical transmission
(aka opacity) versus time on a vide"
                L"o display - 20 points",
                L"", L"1. Disconnecting power to the platform - 10
points", L"2. Turning the handle to unlock the hatch - 10 points", L"3.
Opening the hatch - 10 points",
                L"4. Removing the ADCP from the mooring platform - 10
point", L"5. Installing the new ADCP into the mooring platform - 10
points",
                L"6. Closing the hatch - 10 points", L"7. Turning the
handle to lock the hatch - 10 points", L"8. Reconnecting power to the
platform - 10 points",
                L"", L"1. Locate five areas of biofouling and removing
all biofouling organisms - 5 poin"
                L"ts each"});
            this->checkedListBox1->Location = System::Drawing::Point(12,
12);
            this->checkedListBox1->Name = L"checkedListBox1";
            this->checkedListBox1->Size = System::Drawing::Size(516,
274);
            this->checkedListBox1->TabIndex = 12;
            //
            // TimerStart
            //
            this->TimerStart->Font = (gcnew
System::Drawing::Font(L"Microsoft Sans Serif", 9.75F,
System::Drawing::FontStyle::Regular,
System::Drawing::GraphicsUnit::Point,
```

```
1507                static_cast<System::Byte>(0)));
1508            this->TimerStart->Location = System::Drawing::Point(534,
    … 11);
1509            this->TimerStart->Name = L"TimerStart";
1510            this->TimerStart->Size = System::Drawing::Size(143, 39);
1511            this->TimerStart->TabIndex = 13;
1512            this->TimerStart->Text = L"Start Timer";
1513            this->TimerStart->UseVisualStyleBackColor = true;
1514            this->TimerStart->Click += gcnew System::EventHandler(this,
    … &BaseStation::TimerStart_Click);
1515            //
1516            // groupBox2
1517            //
1518            this->groupBox2->Controls->Add(this->label36);
1519            this->groupBox2->Controls->Add(this->TimeMin);
1520            this->groupBox2->Controls->Add(this->TimeSec);
1521            this->groupBox2->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 10,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1522                static_cast<System::Byte>(0)));
1523            this->groupBox2->Location = System::Drawing::Point(534, 57);
1524            this->groupBox2->Name = L"groupBox2";
1525            this->groupBox2->Size = System::Drawing::Size(143, 50);
1526            this->groupBox2->TabIndex = 14;
1527            this->groupBox2->TabStop = false;
1528            this->groupBox2->Text = L"Current Run Time";
1529            //
1530            // label36
1531            //
1532            this->label36->AutoSize = true;
1533            this->label36->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 16,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1534                static_cast<System::Byte>(0)));
1535            this->label36->Location = System::Drawing::Point(65, 19);
1536            this->label36->Name = L"label36";
1537            this->label36->Size = System::Drawing::Size(18, 26);
```

```
1538            this->label36->TabIndex = 2;
1539            this->label36->Text = L":";
1540            //
1541            // TimeMin
1542            //
1543            this->TimeMin->AutoSize = true;
1544            this->TimeMin->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 16,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
1545                static_cast<System::Byte>(0)));
1546            this->TimeMin->Location = System::Drawing::Point(35, 21);
1547            this->TimeMin->Name = L"TimeMin";
1548            this->TimeMin->Size = System::Drawing::Size(36, 26);
1549            this->TimeMin->TabIndex = 1;
1550            this->TimeMin->Text = L"00";
1551            //
1552            // TimeSec
1553            //
1554            this->TimeSec->AutoSize = true;
1555            this->TimeSec->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 16,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
1556                static_cast<System::Byte>(0)));
1557            this->TimeSec->Location = System::Drawing::Point(77, 21);
1558            this->TimeSec->Name = L"TimeSec";
1559            this->TimeSec->Size = System::Drawing::Size(36, 26);
1560            this->TimeSec->TabIndex = 0;
1561            this->TimeSec->Text = L"00";
1562            //
1563            // timer2
1564            //
1565            this->timer2->Interval = 1000;
1566            this->timer2->Tick += gcnew System::EventHandler(this,
… &BaseStation::timer2_Tick);
1567            //
1568            // groupBox6
1569            //
```

```
1570            this->groupBox6->Controls->Add(this->label24);
1571            this->groupBox6->Controls->Add(this->label23);
1572            this->groupBox6->Controls->Add(this->label22);
1573            this->groupBox6->Controls->Add(this->turner1);
1574            this->groupBox6->Controls->Add(this->shapeContainer1);
1575            this->groupBox6->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 10,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
1576                static_cast<System::Byte>(0)));
1577            this->groupBox6->Location = System::Drawing::Point(534,
… 113);
1578            this->groupBox6->Name = L"groupBox6";
1579            this->groupBox6->Size = System::Drawing::Size(143, 163);
1580            this->groupBox6->TabIndex = 15;
1581            this->groupBox6->TabStop = false;
1582            this->groupBox6->Text = L"Turner Motors";
1583            //
1584            // label24
1585            //
1586            this->label24->AutoSize = true;
1587            this->label24->BackColor = System::Drawing::Color::Silver;
1588            this->label24->Font = (gcnew
… System::Drawing::Font(L"Microsoft Sans Serif", 12,
… System::Drawing::FontStyle::Regular,
… System::Drawing::GraphicsUnit::Point,
1589                static_cast<System::Byte>(0)));
1590            this->label24->ForeColor =
… System::Drawing::SystemColors::ActiveCaptionText;
1591            this->label24->Location = System::Drawing::Point(102, 115);
1592            this->label24->Name = L"label24";
1593            this->label24->Size = System::Drawing::Size(18, 20);
1594            this->label24->TabIndex = 19;
1595            this->label24->Text = L"0";
1596            //
1597            // label23
1598            //
1599            this->label23->AutoSize = true;
1600            this->label23->BackColor = System::Drawing::Color::Silver;
```

```
1601            this->label23->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 12,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1602                static_cast<System::Byte>(0)));
1603            this->label23->ForeColor =
    … System::Drawing::SystemColors::ActiveCaptionText;
1604            this->label23->Location = System::Drawing::Point(22, 115);
1605            this->label23->Name = L"label23";
1606            this->label23->Size = System::Drawing::Size(18, 20);
1607            this->label23->TabIndex = 18;
1608            this->label23->Text = L"0";
1609            //
1610            // label22
1611            //
1612            this->label22->AutoSize = true;
1613            this->label22->BackColor = System::Drawing::Color::Silver;
1614            this->label22->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 12,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1615                static_cast<System::Byte>(0)));
1616            this->label22->ForeColor =
    … System::Drawing::SystemColors::ActiveCaptionText;
1617            this->label22->Location = System::Drawing::Point(102, 40);
1618            this->label22->Name = L"label22";
1619            this->label22->Size = System::Drawing::Size(18, 20);
1620            this->label22->TabIndex = 19;
1621            this->label22->Text = L"0";
1622            //
1623            // turner1
1624            //
1625            this->turner1->AutoSize = true;
1626            this->turner1->BackColor = System::Drawing::Color::Silver;
1627            this->turner1->Font = (gcnew
    … System::Drawing::Font(L"Microsoft Sans Serif", 12,
    … System::Drawing::FontStyle::Regular,
    … System::Drawing::GraphicsUnit::Point,
1628                static_cast<System::Byte>(0)));
```

```
1629            this->turner1->ForeColor =
    … System::Drawing::SystemColors::ActiveCaptionText;
1630            this->turner1->Location = System::Drawing::Point(22, 40);
1631            this->turner1->Name = L"turner1";
1632            this->turner1->Size = System::Drawing::Size(18, 20);
1633            this->turner1->TabIndex = 18;
1634            this->turner1->Text = L"0";
1635            //
1636            // shapeContainer1
1637            //
1638            this->shapeContainer1->Location = System::Drawing::Point(3,
    … 19);
1639            this->shapeContainer1->Margin =
    … System::Windows::Forms::Padding(0);
1640            this->shapeContainer1->Name = L"shapeContainer1";
1641            this->shapeContainer1->Shapes->AddRange(gcnew cli::array<
    … Microsoft::VisualBasic::PowerPacks::Shape^  >(4) {this->ovalShape6,
1642                this->ovalShape5, this->ovalShape4, this->ovalShape3});
1643            this->shapeContainer1->Size = System::Drawing::Size(137,
    … 141);
1644            this->shapeContainer1->TabIndex = 0;
1645            this->shapeContainer1->TabStop = false;
1646            //
1647            // ovalShape6
1648            //
1649            this->ovalShape6->BackColor =
    … System::Drawing::Color::Silver;
1650            this->ovalShape6->BackStyle =
    … Microsoft::VisualBasic::PowerPacks::BackStyle::Opaque;
1651            this->ovalShape6->BorderColor =
    … System::Drawing::Color::White;
1652            this->ovalShape6->BorderWidth = 4;
1653            this->ovalShape6->Location = System::Drawing::Point(83, 80);
1654            this->ovalShape6->Name = L"ovalShape6";
1655            this->ovalShape6->Size = System::Drawing::Size(50, 50);
1656            //
1657            // ovalShape5
1658            //
1659            this->ovalShape5->BackColor =
```

```
1659… System::Drawing::Color::Silver;
1660            this->ovalShape5->BackStyle =
…  Microsoft::VisualBasic::PowerPacks::BackStyle::Opaque;
1661            this->ovalShape5->BorderColor =
…  System::Drawing::Color::White;
1662            this->ovalShape5->BorderWidth = 4;
1663            this->ovalShape5->Location = System::Drawing::Point(2, 80);
1664            this->ovalShape5->Name = L"ovalShape5";
1665            this->ovalShape5->Size = System::Drawing::Size(50, 50);
1666            //
1667            // ovalShape4
1668            //
1669            this->ovalShape4->BackColor =
…  System::Drawing::Color::Silver;
1670            this->ovalShape4->BackStyle =
…  Microsoft::VisualBasic::PowerPacks::BackStyle::Opaque;
1671            this->ovalShape4->BorderColor =
…  System::Drawing::Color::White;
1672            this->ovalShape4->BorderWidth = 4;
1673            this->ovalShape4->Location = System::Drawing::Point(83, 6);
1674            this->ovalShape4->Name = L"ovalShape4";
1675            this->ovalShape4->Size = System::Drawing::Size(50, 50);
1676            //
1677            // ovalShape3
1678            //
1679            this->ovalShape3->BackColor =
…  System::Drawing::Color::Silver;
1680            this->ovalShape3->BackStyle =
…  Microsoft::VisualBasic::PowerPacks::BackStyle::Opaque;
1681            this->ovalShape3->BorderColor =
…  System::Drawing::Color::White;
1682            this->ovalShape3->BorderWidth = 4;
1683            this->ovalShape3->Location = System::Drawing::Point(2, 6);
1684            this->ovalShape3->Name = L"ovalShape3";
1685            this->ovalShape3->Size = System::Drawing::Size(50, 50);
1686            //
1687            // statusStrip1
1688            //
1689            this->statusStrip1->Items->AddRange(gcnew cli::array<
```

```
1689… System::Windows::Forms::ToolStripItem^  >(2) {this->PolarTest,
  … this->SerialLabel2});
1690              this->statusStrip1->Location = System::Drawing::Point(0,
  … 609);
1691              this->statusStrip1->Name = L"statusStrip1";
1692              this->statusStrip1->Size = System::Drawing::Size(1067, 22);
1693              this->statusStrip1->TabIndex = 16;
1694              this->statusStrip1->Text = L"statusStrip1";
1695              //
1696              // PolarTest
1697              //
1698              this->PolarTest->Name = L"PolarTest";
1699              this->PolarTest->Size = System::Drawing::Size(10, 17);
1700              this->PolarTest->Text = L".";
1701              //
1702              // SerialLabel2
1703              //
1704              this->SerialLabel2->Name = L"SerialLabel2";
1705              this->SerialLabel2->Size = System::Drawing::Size(10, 17);
1706              this->SerialLabel2->Text = L".";
1707              //
1708              // BaseStation
1709              //
1710              this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
1711              this->AutoScaleMode =
  … System::Windows::Forms::AutoScaleMode::Font;
1712              this->ClientSize = System::Drawing::Size(1067, 631);
1713              this->Controls->Add(this->statusStrip1);
1714              this->Controls->Add(this->groupBox6);
1715              this->Controls->Add(this->groupBox2);
1716              this->Controls->Add(this->TimerStart);
1717              this->Controls->Add(this->checkedListBox1);
1718              this->Controls->Add(this->groupBox5);
1719              this->Controls->Add(this->groupBox1);
1720              this->Controls->Add(this->estopDisp);
1721              this->Controls->Add(this->ToolPanel);
1722              this->Controls->Add(this->panel1);
1723              this->Controls->Add(this->tabControl1);
1724              this->Name = L"BaseStation";
```

```
1725            this->RightToLeftLayout = true;
1726            this->Text = L"BaseStation";
1727            this->Load += gcnew System::EventHandler(this,
    … &BaseStation::BaseStation_Load);
1728            this->ToolPanel->ResumeLayout(false);
1729            this->ToolPanel->PerformLayout();
1730            this->panel1->ResumeLayout(false);
1731            this->panel1->PerformLayout();
1732            this->groupBox1->ResumeLayout(false);
1733            this->groupBox1->PerformLayout();
1734            this->Serial->ResumeLayout(false);
1735            this->groupBox4->ResumeLayout(false);
1736            this->groupBox4->PerformLayout();
1737            this->groupBox3->ResumeLayout(false);
1738            this->groupBox3->PerformLayout();
1739            this->ThrusterToggle->ResumeLayout(false);
1740            this->ThrusterToggle->PerformLayout();
1741            this->Main->ResumeLayout(false);
1742            this->Main->PerformLayout();
1743            (cli::safe_cast<System::ComponentModel::ISupportInitialize^
    … >(this->TurnerBar))->EndInit();
1744            (cli::safe_cast<System::ComponentModel::ISupportInitialize^
    … >(this->PitchRollBar))->EndInit();
1745            (cli::safe_cast<System::ComponentModel::ISupportInitialize^
    … >(this->StrafeSpeedCTL))->EndInit();
1746            (cli::safe_cast<System::ComponentModel::ISupportInitialize^
    … >(this->VertLimiter))->EndInit();
1747            (cli::safe_cast<System::ComponentModel::ISupportInitialize^
    … >(this->HorizLimiter))->EndInit();
1748            this->tabControl1->ResumeLayout(false);
1749            this->Status->ResumeLayout(false);
1750            (cli::safe_cast<System::ComponentModel::ISupportInitialize^
    … >(this->dataGridView1))->EndInit();
1751            this->Debug->ResumeLayout(false);
1752            this->Debug->PerformLayout();
1753            this->groupBox5->ResumeLayout(false);
1754            this->groupBox5->PerformLayout();
1755            this->groupBox2->ResumeLayout(false);
1756            this->groupBox2->PerformLayout();
```

```
1757            this->groupBox6->ResumeLayout(false);
1758            this->groupBox6->PerformLayout();
1759            this->statusStrip1->ResumeLayout(false);
1760            this->statusStrip1->PerformLayout();
1761            this->ResumeLayout(false);
1762            this->PerformLayout();
1763
1764        }
1765 #pragma endregion
1766     private: System::Void BaseStation_Load(System::Object^  sender,
   … System::EventArgs^  e) {
1767             }
1768     private: System::Void button2_Click(System::Object^  sender,
   … System::EventArgs^  e) {
1769                 UpdateConnection();
1770              }
1771     private: System::Void HorizLimiter_Scroll(System::Object^  sender,
   … System::EventArgs^  e) {
1772                 SpeedDivider_H = (float)(HorizLimiter->Value)/10;
1773              }
1774     private: System::Void StrafeSpeedCTL_Scroll(System::Object^  sender,
   … System::EventArgs^  e) {
1775                 StrafeSpeed = (float)(StrafeSpeedCTL->Value)/10;
1776              }
1777     private: System::Void VertLimiter_Scroll(System::Object^  sender,
   … System::EventArgs^  e) {
1778                 SpeedDivider_V = (float)(VertLimiter->Value)/10;
1779              }
1780     private: System::Void PitchRollBar_Scroll(System::Object^  sender,
   … System::EventArgs^  e) {
1781                 PitchRollFactor = (float)(PitchRollBar->Value)/10;
1782              }
1783     private: System::Void ControlCheck_Click(System::Object^  sender,
   … System::EventArgs^  e) {
1784                 CheckControllerStatus();
1785              }
1786 private: System::Void SerialForce_Click(System::Object^  sender,
   … System::EventArgs^  e) {
1787             if(!serialstart){
```

```
1788                    serialstart = true;
1789                }else{
1790                    serialstart = false;
1791                }
1792            }
1793
1794    private: System::Void TimerStart_Click(System::Object^  sender,
    … System::EventArgs^  e) {
1795                if(!missiontime){
1796                    missiontime = true;
1797                    timer2->Enabled = true;
1798                    TimerStart -> Text = "Pause Timer";
1799                }else{
1800                    missiontime = false;
1801                    timer2->Enabled = false;
1802                    TimerStart -> Text = "Start Timer";
1803                }
1804            }
1805    private: System::Void TimerReset_Click(System::Object^  sender,
    … System::EventArgs^  e) {
1806                 missiontime = false;
1807                 timer2->Enabled = false;
1808                 TimerStart -> Text = "Start Timer";
1809                 timercount = 0;
1810                 TimeMin -> Text = "0";
1811                 TimeSec -> Text = "0";
1812            }
1813    };
1814    }
1815
1816    ---------------------------------------- BaseStation.cpp
    … ------------------------------------------------
1817
1818    //Copyright 2013 Purdue University ROV Team
1819    //Code by Clement Lan, Nick Molo
1820
1821    //Please ask permission (clement.lan@gmail.com, nmolo@purdue.edu)
1822    //before using this code
1823
```

```
1824
1825 #include "stdafx.h"
1826 #include "BaseStation.h"
1827
1828
1829 ROV_BaseStation::BaseStation::BaseStation(void) {//needs cleaning
1830 //------------------- Component Initializations
   …  -------------------------------
1831     InitializeComponent();
1832     vecstatus = false; //needs to be as early as possible to prevent
   …  exceptions when reading from serialport
1833     controller0 = new XBoxController(0);
1834     controller1 = new XBoxController(1);
1835     UpdateConnection();
1836     CheckControllerStatus();
1837     String ^ out = "";
1838 //--------------------- Array Allocation
   …  ---------------------------------------
1839     inputs = (int *)malloc(sizeof(int)*10);
1840     thrust_vec = (int *)malloc(sizeof(int)*11);
1841     polar_vec = (int *)malloc(sizeof(int)*2);
1842     prev_thrust_vec = (int *)malloc(sizeof(int)*11);
1843     gyro_data = (short *)malloc(sizeof(short)*4);
1844     accel_data = (short *)malloc(sizeof(short)*3);
1845     mag_data = (short *)malloc(sizeof(short)*3);
1846     pb_data = (int *)malloc(sizeof(int)*3);
1847     rcvstr = (char *)(malloc(sizeof(char)*9));
1848     faultdata = (char *)malloc(sizeof(char)*2);
1849 //--------------------- Array Initialization
   …  -----------------------------------
1850     for (int i = 0; i < 11; i++) {
1851         thrust_vec[i] = 0;
1852         prev_thrust_vec[i] = 0;
1853     }
1854     for (int i = 0; i < 3; i++) {
1855         pb_data[i] = 0;
1856         gyro_data[i] = 0;
1857         accel_data[i] = 0;
1858         mag_data[i] = 0;
```

```
1859     } gyro_data[3] = 0;
1860
1861     for(int i = 0; i < 8; i++){
1862         rcvstr[i] = 0;
1863     }
1864     faultdata[0] = 0x00;
1865     faultdata[1] = 0x00;
1866 //------------------- Integer/Float Initializations
    …----------------------------
1867
1868     SpeedDivider_H = (float)(HorizLimiter->Value)/10;
1869     SpeedDivider_V = (float)(VertLimiter->Value)/10;
1870     StrafeSpeed = (float)(StrafeSpeedCTL->Value)/10;
1871     PitchRollFactor = (float)(PitchRollBar->Value)/10;
1872     nLX = 0; nLY = 0;
1873     nRX = 0; nRY = 0;
1874     LX = 0; LY = 0;
1875     nZ = 0;
1876     depth = 0;
1877     runtimeus = 0;
1878     depthoffset = 0;
1879     fifteentimer = 900000;
1880     prevrange = 0;
1881     startCount = 0;
1882     selectCount = 0;
1883     bCount = 0;
1884     yCount = 0;
1885     timercount = 0;
1886
1887 //----------------------- Booleans Initialization
    …-----------------------------
1888     cam_toggleF = false;//For the front/ periscope camera
1889     cam_toggleR = false; //For the rear/ tool camera
1890     manip_open = false; //True if open false if closed
1891     estop = false; //ESTOP!!!!
1892     vecstatus = true; //set true at end of setup so program knows its OK
    … to put values into arrays
1893     prox = false;
1894     unitState = false;
```

```
1895        serialstart = false;
1896        bool missiontime = false;
1897    }
1898
1899    System::Void ROV_BaseStation::BaseStation::CheckControllerStatus() {
1900        //check controller0 connection status
1901        if (!controller0->isConnected()) {
1902            this->C1Status -> Text = gcnew String("Controller
    …   Disconnected");
1903            this->C1Status -> ForeColor = Color::Red;
1904        }else{
1905            this->C1Status -> Text = gcnew String("Controller Connected");
1906            this->C1Status -> ForeColor = Color::Black;
1907        }
1908    }
1909    System::Void ROV_BaseStation::BaseStation::timer1_Tick(System::Object^
    …   sender, System::EventArgs^  e) { //Needs cleaned
1910
1911        inputs[0] =
    …   (short)(controller0->GetState().Gamepad.sThumbLX*SpeedDivider_H); //left
    …   stick x
1912        inputs[1] =
    …   (short)(controller0->GetState().Gamepad.sThumbLY*SpeedDivider_H); //left
    …   stick y
1913        inputs[2] =
    …   (short)(controller0->GetState().Gamepad.sThumbRX*SpeedDivider_V*
    …   PitchRollFactor); //right stick x
1914        inputs[3] =
    …   (short)(controller0->GetState().Gamepad.sThumbRY*SpeedDivider_V*
    …   PitchRollFactor); //right stick y
1915        inputs[4] =
    …   (short)(controller0->GetState().Gamepad.bLeftTrigger*SpeedDivider_V);
    …   //left trigger
1916        inputs[5] =
    …   (short)(controller0->GetState().Gamepad.bRightTrigger*SpeedDivider_V);
    …   //right trigger
1917
1918        WORD wbuttons0 = controller0->GetState().Gamepad.wButtons;
1919
```

```
1920    if (wbuttons0 & XINPUT_GAMEPAD_START) { //Toggle E-Stop
1921        startCount++;
1922        if(!estop && startCount >= 25){
1923            estop = true;
1924            estopDisp -> Text = gcnew String("ESTOP");
1925            startCount = 0;
1926        }
1927        else if(estop && startCount >= 30){
1928            estop = false;
1929            estopDisp -> Text = gcnew String("");
1930            startCount = 0;
1931        }
1932    }
1933    if (wbuttons0 & XINPUT_GAMEPAD_BACK) { //Toggle Speed mode
1934        selectCount++;
1935        if(!speed && selectCount >= 25){
1936            speed = true;
1937            estopDisp -> Text = gcnew String("LOW SPEED");
1938            HorizLimiter->Value=8;
1939            VertLimiter->Value=8;
1940            StrafeSpeedCTL->Value=8;
1941            SpeedDivider_H = (float)(HorizLimiter->Value)/10;
1942            SpeedDivider_V = (float)(VertLimiter->Value)/10;
1943            StrafeSpeed = (float)(StrafeSpeedCTL->Value)/10;
1944            selectCount = 0;
1945        }
1946        else if(speed && selectCount >= 30){
1947            speed = false;
1948            estopDisp -> Text = gcnew String("");
1949            HorizLimiter->Value=10;
1950            VertLimiter->Value=10;
1951            StrafeSpeedCTL->Value=10;
1952            SpeedDivider_H = (float)(HorizLimiter->Value)/10;
1953            SpeedDivider_V = (float)(VertLimiter->Value)/10;
1954            StrafeSpeed = (float)(StrafeSpeedCTL->Value)/10;
1955            selectCount = 0;
1956        }
1957    }
1958
```

```
1959
1960
1961
1962
1963        if (wbuttons0 & XINPUT_GAMEPAD_X) {//opens manip
1964            if (manip_open == false) {
1965                manip_open = true;
1966                ManipStatus->Text = gcnew String("Open");
1967            }
1968        }
1969
1970        if (wbuttons0 & XINPUT_GAMEPAD_A) {//closes manip
1971            if (manip_open == true) {
1972                manip_open = false;
1973                ManipStatus->Text = gcnew String("Closed");
1974            }
1975        }
1976
1977        if ( wbuttons0 & XINPUT_GAMEPAD_RIGHT_THUMB ) {//Camera set 1
1978            lsCount++;
1979            if(lsCount >= 25){
1980                cam_toggleF = !cam_toggleF;
1981                lsCount = 0;
1982            }
1983        }
1984        if ( wbuttons0 & XINPUT_GAMEPAD_LEFT_THUMB) {//Camera Set2
1985            rsCount++;
1986            if(rsCount >= 25){
1987                cam_toggleR = !cam_toggleR;
1988                rsCount = 0;
1989            }
1990        }
1991
1992
1993      //check to see if both shoulders are pressed
1994        if ( wbuttons0 & XINPUT_GAMEPAD_LEFT_SHOULDER && wbuttons0 &
   … XINPUT_GAMEPAD_RIGHT_SHOULDER ) {
1995            wbuttons0 = wbuttons0&(~XINPUT_GAMEPAD_LEFT_SHOULDER);
1996            wbuttons0 = wbuttons0&(~XINPUT_GAMEPAD_RIGHT_SHOULDER);
```

```
1997        }
1998
1999      //check for "deadzone"
2000        for (int i = 0; i < 4; i++) {if (abs(inputs[i]) < INPUT_DEADZONE)
    … inputs[i] = 0; }
2001        for (int i = 4; i < 6; i++) {if (abs(inputs[i]) < TRIGGER_THRESH)
    … inputs[i] = 0; }
2002        for (int i = 7; i < 9; i++) {if (abs(inputs[i]) < TRIGGER_THRESH)
    … inputs[i] = 0; }
2003
2004        GenerateThrusterVec(inputs);
2005        AdjustVector();
2006        UpdateGuiThrust();
2007        if(serialstart){
2008            SendThrusterVec();
2009        }
2010        else if(!serialstart){
2011            this -> serialLabel -> Text = "Serial Disconnected";
2012        }
2013        out = "";
2014
2015    }
2016
2017    System::Void ROV_BaseStation::BaseStation::timer2_Tick(System::Object^
    … sender, System::EventArgs^  e) {
2018        timercount++;
2019        int sec = timercount % 60;
2020        int min = ((timercount - sec) / 60) % 60;
2021        this -> TimeMin -> Text = min.ToString();
2022        this -> TimeSec -> Text = sec.ToString();
2023    }
2024
2025    //actually generates the thruster vectors based on input
2026    System::Void
    … ROV_BaseStation::BaseStation::GenerateMainThrusterVec(void){
2027    //generate Polar vector
2028    //  polar_vec[0] = Math.Sqrt(Math.Pow(LX, 2) + Math.Pow(LY, 2));
2029    //  polar_vec[1] = Math.Atan2(LY, LX);
2030
```

```
2031
2032    //calculate normalized values
2033      nLX = inputs[0]/(float)(32767);
2034      nLY = inputs[1]/(float)(32767);
2035      nRX = inputs[2]/(float)(32767);
2036      nRY = inputs[3]/(float)(32767);
2037      nZ = ((float)(inputs[5]-inputs[4]))/255;
2038
2039      //[LF, RF, LB, RB, LFV, RFV, LBV, RBV, Cap, toggles, debug]
2040      //calculate horizontal thrusters
2041      //LF = (ForwardBack+LeftRight-LStrafe/2+RStrafe/2)/Nonzeros*255
2042      thrust_vec[0] = (int)GetThrust(nLX, nLY);
2043      //RF = (ForwardBack-LeftRight+LeftStrafe/2-RStrafe/2)/Nonzeros*255
2044      thrust_vec[1] = (int)GetThrust(-nLX, nLY);
2045      //LB = (ForwardBack+LeftRight+LStrafe/2-RStrafe/2)/Nonzeros*255
2046      thrust_vec[2] = (int)GetThrust(nLX, nLY);
2047      //RB = (ForwardBack-LeftRight-LStrafe/2+RStrafe/2)/Nonzeros*255
2048      thrust_vec[3] = (int)GetThrust(-nLX, nLY);
2049      //Front thrusters are backwards.
2050      //calculate vertical thrusters: LFV, RFV, LBV, RBV
2051      //LFV
2052      thrust_vec[4] = (int)GetThrustV(nZ, nRX, -nRY);
2053      //RFV
2054      thrust_vec[5] = (int)GetThrustV(nZ, -nRX, -nRY);
2055      //LBV
2056      thrust_vec[6] = (int)GetThrustV(nZ, nRX, nRY);
2057      //RBV
2058      thrust_vec[7] = (int)GetThrustV(nZ, -nRX, nRY);
2059
2060      for (int i = 0; i < 8; i++) {
2061          if (thrust_vec[i] > 0) thrust_vec[i] += 40;
2062          else if (thrust_vec[i] < 0) thrust_vec[i] -= 40;
2063      }
2064
2065      WORD wbuttons0 = controller0->GetState().Gamepad.wButtons;
2066      if ( wbuttons0 & XINPUT_GAMEPAD_RIGHT_SHOULDER ) {
2067          thrust_vec[0] += (int)(100*StrafeSpeed);
2068          thrust_vec[1] -= (int)(100*StrafeSpeed);
2069          thrust_vec[2] -= (int)(100*StrafeSpeed);
```

```
2070            thrust_vec[3] += (int)(100*StrafeSpeed);
2071            //thrust_vec[3] = -thrust_vec[3];
2072        strafing = true; //set strafe to true so you know that it needs to
    … be averaged in
2073        } else if ( wbuttons0 & XINPUT_GAMEPAD_LEFT_SHOULDER ) {
2074            thrust_vec[0] -= (int)(100*StrafeSpeed);
2075            thrust_vec[1] += (int)(100*StrafeSpeed);
2076            thrust_vec[2] += (int)(100*StrafeSpeed);
2077            thrust_vec[3] -= (int)(100*StrafeSpeed);
2078            //thrust_vec[3] = -thrust_vec[3];
2079        strafing = true;
2080        } else strafing = false;
2081
2082        if (strafing == true) {
2083                for (int i = 0; i < 4; i++) {
2084                    if (thrust_vec[i] > 100) thrust_vec[i] = 100;
2085                    else if (thrust_vec[i] < -100) thrust_vec[i] = -100;
2086                }
2087        }
2088
2089        //flip front motors
2090        thrust_vec[0] = -thrust_vec[0];
2091        thrust_vec[1] = -thrust_vec[1];
2092        //for(int i = 0; i<8; i++){
2093            //if (thrust_vec[i] == 12) thrust_vec[i] =13;
2094        //}
2095    }
2096
2097    System::Void ROV_BaseStation::BaseStation::UpdateConnection() {
2098    try {
2099            this->serialPort1->PortName = this->CPortBox->Text;
2100            this->serialPort1->Open();
2101        } catch (Exception ^) {}
2102        if (serialPort1->IsOpen) {
2103            serialstart = true;
2104            this->serialLabel->ForeColor = Color::Black;
2105            this->serialLabel -> Text = gcnew String("");
2106        }
2107        if (!serialPort1->IsOpen) {
```

```
2108            serialstart = false;
2109            this->serialLabel -> Text = gcnew String("Serial Disconnected");
2110            this->serialLabel->ForeColor = Color::Red;
2111        }
2112    }
2113
2114    //checks things and makes sure you dont overcurrent the motors, Limits
    … things
2115    System::Void ROV_BaseStation::BaseStation::GenerateThrusterVec(int*
    … inputs) {
2116
2117        GenerateMainThrusterVec();
2118
2119        int MaximumChangeBar = 10;
2120        for (int i = 0; i < 9; i++) {
2121            if (abs(thrust_vec[i]-prev_thrust_vec[i]) > MaximumChangeBar) {
2122                if (thrust_vec[i] >= prev_thrust_vec[i]) { //i.e. going from
    … 40->70
2123                    thrust_vec[i] = prev_thrust_vec[i]+MaximumChangeBar;
2124                } else if (thrust_vec[i] < prev_thrust_vec[i]){ //i.e. going
    … from 60->45
2125                    thrust_vec[i] = prev_thrust_vec[i]-MaximumChangeBar;
2126                }
2127            }
2128        }
2129
2130        if (DisableLF->Checked == true) thrust_vec[0] = 0;
2131        if (DisableRF->Checked == true) thrust_vec[1] = 0;
2132        if (DisableLB->Checked == true) thrust_vec[2] = 0;
2133        if (DisableRB->Checked == true) thrust_vec[3] = 0;
2134        if (DisableLFV->Checked == true) thrust_vec[4] = 0;
2135        if (DisableRFV->Checked == true) thrust_vec[5] = 0;
2136        if (DisableLBV->Checked == true) thrust_vec[6] = 0;
2137        if (DisableRBV->Checked == true) thrust_vec[7] = 0;
2138
2139        if (estop) { //go into standby mode (disable pins)
2140            for (int i = 0; i < 9; i++) { //for each of the 9 motors
2141                thrust_vec[i] = 0;//set 8th bit on each byte w/ 50% duty
    … cycle
```

```
2142            }
2143        }
2144
2145        for (int i = 0; i < 11; i++) { //copy into prev_thrust_vec
2146            prev_thrust_vec[i] = thrust_vec[i];
2147        }
2148
2149    }
2150
2151    System::Void ROV_BaseStation::BaseStation::UpdateGuiThrust() {
2152        LFThrust->Text = gcnew String(Convert::ToString(thrust_vec[0]));
2153        RFThrust->Text = gcnew String(Convert::ToString(thrust_vec[1]));
2154        LBThrust->Text = gcnew String(Convert::ToString(thrust_vec[2]));
2155        RBThrust->Text = gcnew String(Convert::ToString(thrust_vec[3]));
2156        LFVThrust->Text = gcnew String(Convert::ToString(thrust_vec[4]));
2157        RFVThrust->Text = gcnew String(Convert::ToString(thrust_vec[5]));
2158        LBVThrust->Text = gcnew String(Convert::ToString(thrust_vec[6]));
2159        RBVThrust->Text = gcnew String(Convert::ToString(thrust_vec[7]));
2160        this -> recValues -> Text = out;
2161    }
2162
2163
2164    System::Void ROV_BaseStation::BaseStation::AdjustVector() {
2165        //50% is the center, i.e. 0 thrust
2166        for (int i = 0; i < 8; i++) {
2167            if(thrust_vec[i] > 0 && thrust_vec[i] < 40){
2168                thrust_vec[i] = 40;
2169            }
2170            if(thrust_vec[i] < 0 && thrust_vec[i] > -40){
2171                thrust_vec[i] = -40;
2172            }
2173        }
2174      //thrust_vec[9] = (int)((this->thrust_vec[9])/100*(254));
2175    }
2176
2177
2178    int ROV_BaseStation::BaseStation::GetHorizNonZero(){
2179        int n = 0;
2180        for(int i = 0; i < 2; i++)
```

```
2181     {
2182         if (abs(this->inputs[i]) > 0) n++;
2183     }
2184 
2185     if (n == 0) return 1; else return n;
2186 }
2187 
2188 
2189 int ROV_BaseStation::BaseStation::GetVertNonZero(){
2190     int n = 0;
2191     for (int i = 2; i < 6; i++) {
2192         if (abs(this->inputs[i]) > 0) n++;
2193     }
2194     if (n == 0) return 1; else return n;
2195 }
2196 
2197 System::Void ROV_BaseStation::BaseStation::SendThrusterVec() {
2198   //copy vector into a char array
2199     array<unsigned char> ^sendarr = gcnew array<unsigned char>(19);
2200     sendarr[0] = (char)(12);
2201     sendarr[18] = (char)(13);
2202     for(int i = 1; i < 9; i++) {
2203         sendarr[i] = (char)(this->thrust_vec[i-1]);
2204         if (i <= 8) sendarr[i]*(Convert::ToInt32(estop));
2205     }
2206     sendarr[9] = (char)this->manip_open;
2207 //  sendarr[10] = (char)this->tape_down;
2208 //  sendarr[11] = (char)this ->patch_open;
2209 //  sendarr[12] = (char)this->lift_open;
2210     sendarr[13] = (char)this->cam_toggleF;
2211     sendarr[14] = (char)this->cam_toggleR;
2212     sendarr[15] = (char)this->red;
2213     sendarr[16] = (char)this->green;
2214     sendarr[17] = (char)this->blue;
2215 
2216     String ^ out;
2217     for(int i = 0; i < 19; i++){
2218         out = out+" "+(char)sendarr[i];
2219     }
```

```
2220      this -> serialLabel ->Text = out;
2221      this -> SerialDebug -> Text = out;
2222      this -> SerialLabel2 -> Text = out;
2223      if (this->serialPort1->IsOpen) {
2224          this->serialPort1->Write(sendarr, 0, 19);
2225      }
2226  }
2227
2228  System::Void
    …  ROV_BaseStation::BaseStation::serialPort1_DataReceived(System::Object^
    …  sender, System::IO::Ports::SerialDataReceivedEventArgs^  e) {
2229    int data;
2230    if(serialstart){
2231        if (this->serialPort1->BytesToRead > 0) {
2232          data = (char)(this->serialPort1->ReadByte());//read leading byte
2233          //this->proxSensor->BackColor=Color::Green;
2234          if (data == (char)(18)) {
2235
2236              while(this->serialPort1->BytesToRead < 3);
2237                 //rcvstr[0] = (char)(13);
2238              for (int i = 0; i < 2; i++) {
2239                      rcvstr[i] = (this->serialPort1->ReadByte());
2240              }
2241              if(rcvstr[0] == 48){
2242                  prox = false;
2243              }
2244              else if(rcvstr[0] == 49){
2245                  prox = true;
2246              }
2247              for(int i = 0; i < 2; i++){
2248                  out = out+" "+(char)rcvstr[i];
2249              }
2250          }
2251        }
2252    }
2253  }
```