OddBotics

The Open Development & Design Platform for Robotics Students

Final Report

Eric Feuvrier-Danziger

Chris Dunkers

Mitch Kosowski

Drew Marschner

5/7/2015

Abstract

Hardware integration is a constant problem in robotics. Building new robots requires time and energy to make sure the new parts are both physically and electrically connected - and then the software problems begin. Creating new internal models, updating software to reflect a new build, and ensuring the robot will still accomplish set goals safely take even more time. We have built a modular robotic platform to address these issues. Our platform will allow students who want to build robots the opportunity not just to build them, which can take an entire semester, but to program them as well.

Our system includes a variety of actuation and sensing modules, with central processing modules that sit between them. We have built wheeled motor modules, ultrasound sensing modules, and two different kinds of camera perception modules. Using our modules, a class can get started programming various advanced robotics topics immediately.

Table of Contents

1. Project Description

Every year high school students involved with FIRST robotics build a robot. Every year they spend almost all of their time physically building. Because of this these robots almost invariably wind up being essentially remote-control cars. The students do not get a chance to become acquainted with topics that involve programming the robot itself such as frame transformations, path planning, or computer vision.

At the same time robotic software has recently become fairly standardized. The Robot Operating System (ROS) is a modular software system that is widely used in academic robotics. It makes starting a robotics project so much easier than starting over from scratch for each project. If there was a similarly modular hardware system available students could build a robot quickly and then be able to focus on the programming aspect of robotics.

Currently there are some robotics platforms available but our team still feels that the market is underserved. At the lower end LEGO Mindstorms is an example of a kit that is cheaper but not very functional. Mindstorms can be used to explore concepts in robotics such as dead reckoning but the precision and accuracy provided by the system makes anything beyond simple demonstration of robotics concepts impossible. At the higher end systems such as the Clearpath Husky offer mobile robotics bases that provide more capability than most users will ever need for a price that few can afford.

Programming is one of the most important ways for students to learn mathematics, teamwork, and build professional skills for the future. Our system will allow them to get started programming quickly on a sophisticated but easy-to-use robotics base while staying within a budget schools can afford.

2. Use Cases

In an Introduction to Robotics undergraduate course Professor Dominic wants to have his students learn about the development and use of a control user interface for a mobile robot [1]. In the assignment students will create a user interface that will allow users to send commands to the robot and receive video feedback. Dominic wants his students to build the robot quickly using the OddBot platform (depicted in Figure 1) as robot design and construction are not the focus of the course. Using OddBot video and locomotion modules the students can quickly assemble their robots without worrying about integrating the hardware themselves. They can instead start programming immediately.

Later in the same course Dominic wants his students to learn how robots localize with dead reckoning. The students can program the OddBot platform to execute movement along specific patterns and see how the actual and theoretical position of the robot differ. The locomotion module's encoders will provide feedback, but the professor wants to establish that feedback is not sufficient and that successful navigation requires feedback from the environment as well.

To expand upon this the professor wants the students to follow a line around a course using the locomotion and sensor modules of the OddBot system. This will allow the students to incorporate all that they have learned in the course and combine odometry and sensor data to accomplish a goal.

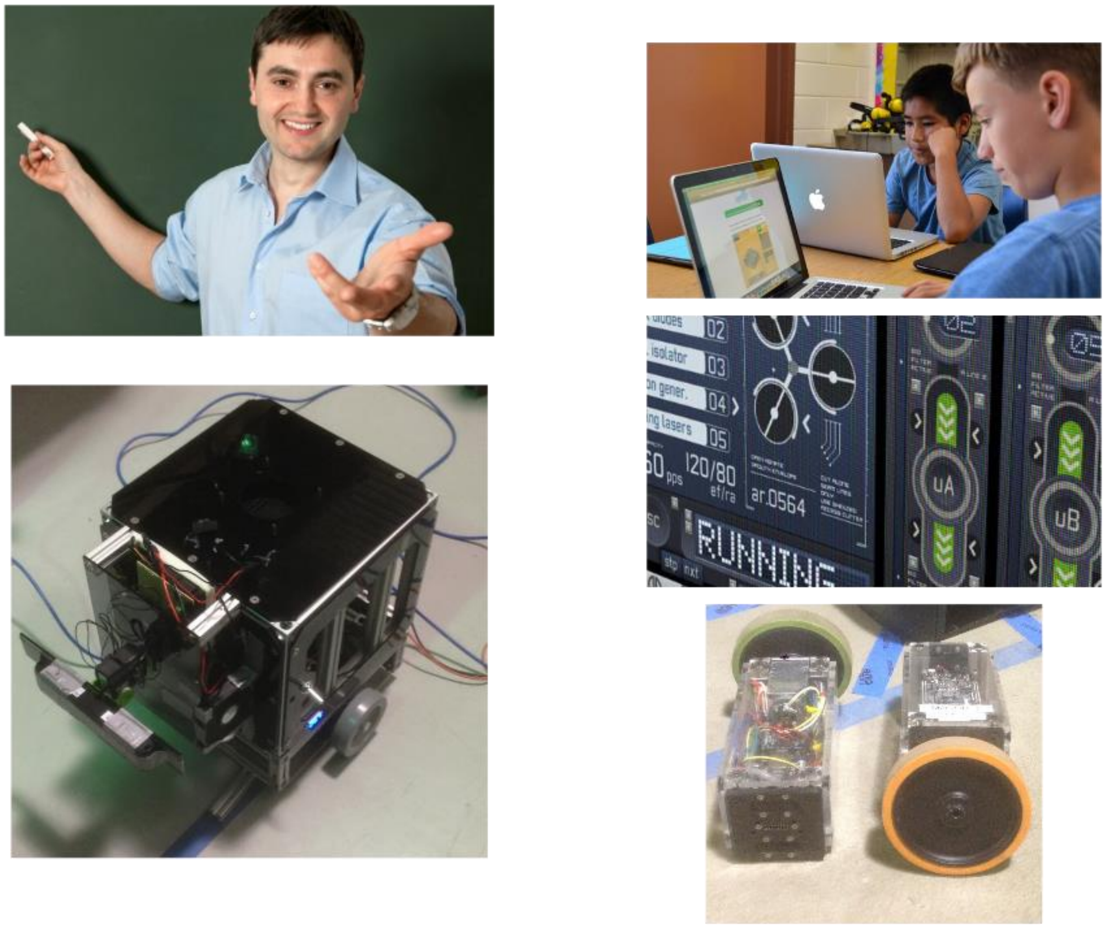


Figure 1: Situational Drawing of the OddBot for a Basic Course

In a different course [2], Professor Davidovic wants to teach his students about frames and transformations. To do this he wants them to program a perception system that gets coordinates for the robot and a goal in the world frame, and translates them into the robot's frame. He also wants them to practice building a module from scratch so they can get practice building in addition to programming.

By using the modular OddBot system his students will be able to immediately both learn math the required math and learn how to apply it. The basics of frame transformations and computer vision are linear algebra and matrices. These concepts can be abstract and hard to understand but using a real physical system to explore them eases students into the learning process and will eventually allow for more mastery and deeper learning of the subjects.

An example of this would be to have Professor Davidovic have his students move the actuation modules of the OddBot from the configuration where driving the motors forward moves the OddBot forward in the x direction in the frame of the robot to the configuration where driving the motors forward moves the OddBot forward in the y direction in the frame of the robot. The OddBot platform allows students to do relatively simple modifications to the programming to see how this affect the performance and behavior of the robot.

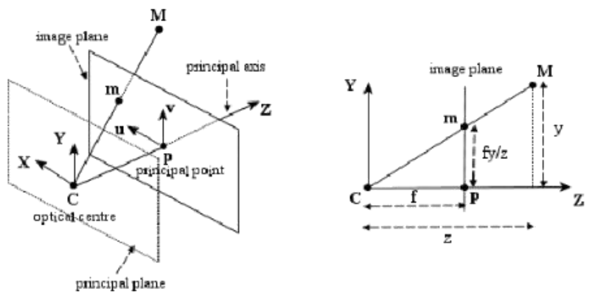
Yet another example is using computer vision with the OddBot system. While the underlying programming can easily become extremely complicated, the basic topics of computer vision (such as feature extraction, triangulation, etc.) are fairly straightforward. One possible assignment could be to have students use a given a template matching program and then be asked to find a way to compute an estimate of the robot's position in the world using only the results of the template matching program. This template matching program could then send coordinates to the OddBot as a trajectory so it can reach a goal. Once again the OddBot platform would allow students to almost immediately begin programming the robot to explore these topics in computer vision without having to worry about the pains of hardware integration.

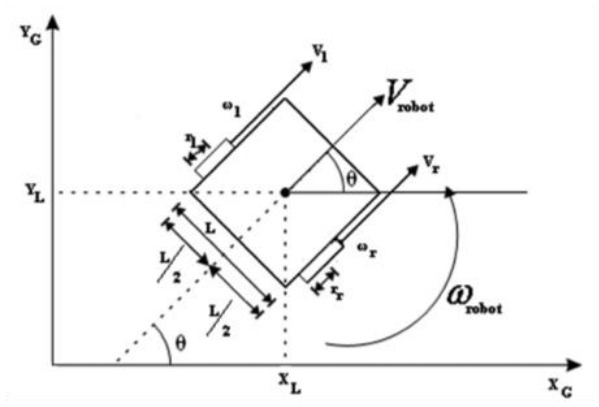
$$AX = B \Rightarrow \mathrm{U}\Lambda V^T X = B$$

$$\Rightarrow \left(\mathrm{U}^T \mathrm{U}\right)\Lambda V^T X = \mathrm{U}^T B \Rightarrow \left(\Lambda^{-1}\Lambda\right)V^T X = \Lambda^{-1}\mathrm{U}^T B \Rightarrow \left(VV^T\right)X = V\Lambda^{-1}\mathrm{U}^T B$$

$$\Rightarrow X = V\Lambda^{-1}\mathrm{U}^T B$$

$$\text{where } \Lambda^{-1} = \begin{bmatrix} \lambda_{11} & 0 & \dots & 0 \\ 0 & \lambda_{22} & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{nn} \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{\lambda_{11}} & 0 & \dots & 0 \\ 0 & \frac{1}{\lambda_{22}} & \dots & 0 \\ \vdots & \dots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda_{nn}} \end{bmatrix}$$





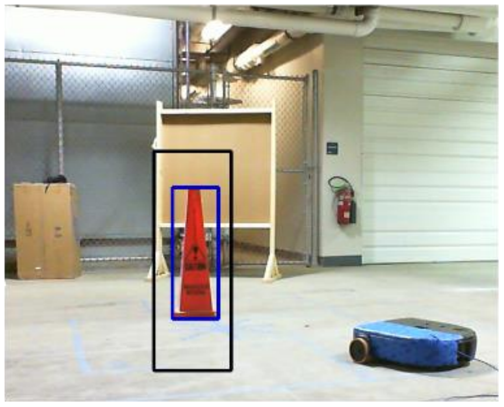


Figure 2: Situational Drawing of the OddBot for an Advanced Course

3. System-Level Requirements

The use cases for our project lead directly into the requirements which in turn inform the functional architecture. Because our project required the design and assembly of a modular robotics platform with many different functionalities we had a long list of requirements. The following requirements are split into mandatory (the minimum threshold for the project) and desired (the objective for the project). These requirements are then split into functional and non-functional subcategories.

3.1 Mandatory Functional Requirements

M.F.1. The robot platform shall stream video with at least 640x480 resolution

Module required: Perception

M.F.2. The robot platform shall have a wide field of view

Module required: Perception

M.F.3. The robot platform shall sense a line

Module required: Perception

M.F.4. The robot platform shall plan a path

Module required: Locomotion and Perception

M.F.5. The robot platform shall locomote to a point with 5 centimeter accuracy

Module required: Locomotion and Perception

M.F.6. The robot platform shall locomote to a pose with 5 degree accuracy

Module required: Locomotion and Perception

M.F.7. The robot platform shall move with a speed of at least 0.1 meter per second

Module required: Locomotion

M.F.8. Mated connection between modules will be able to withstand 10 lbs. of shear force while maintain connection

M.F.9. Mated connection between modules will be able to withstand 10 lbs. of tensile force while maintain connection

M.F.10. Information Architecture shall describe the inputs and outputs of the object

M.F.11. Information Architecture shall describe relevant physical dimensions of the object

M.F.12. Message passing between objects will have a maximum latency of 100 ms

3.2 Mandatory Non-Functional Requirements

M.N.1. The robot platform shall be durable

M.N.2. The robot platform shall have a few standard modules capable of locomotion and perception

M.N.3. The robot platform shall be easy to assemble and disassemble

M.N.4. The robot platform shall be affordable

M.N.5. The connection shall be able to provide 12V/5A of DC power

M.N.6. The connection shall be able to provide 10Mb/s data transmission speed

M.N.7. Connector-to-connector mating time shall be under 1 minute

3.3 Desired Functional Requirement

D.F.1. The robot platform shall be able to perform SLAM accurately

Modules required: Locomotion and Perception

3.4 Desired Non-Functional Requirements

D.N.1. The robot platform shall have many modules

D.N.2. The robot platform shall allow users to construct their own modules easily

D.N.3. The robot platform shall have available tutorials

D.N.4. The robot platform shall feature challenges and competition for users

D.N.5. The robot platform shall have a pleasing aesthetic design

4. Functional Architecture

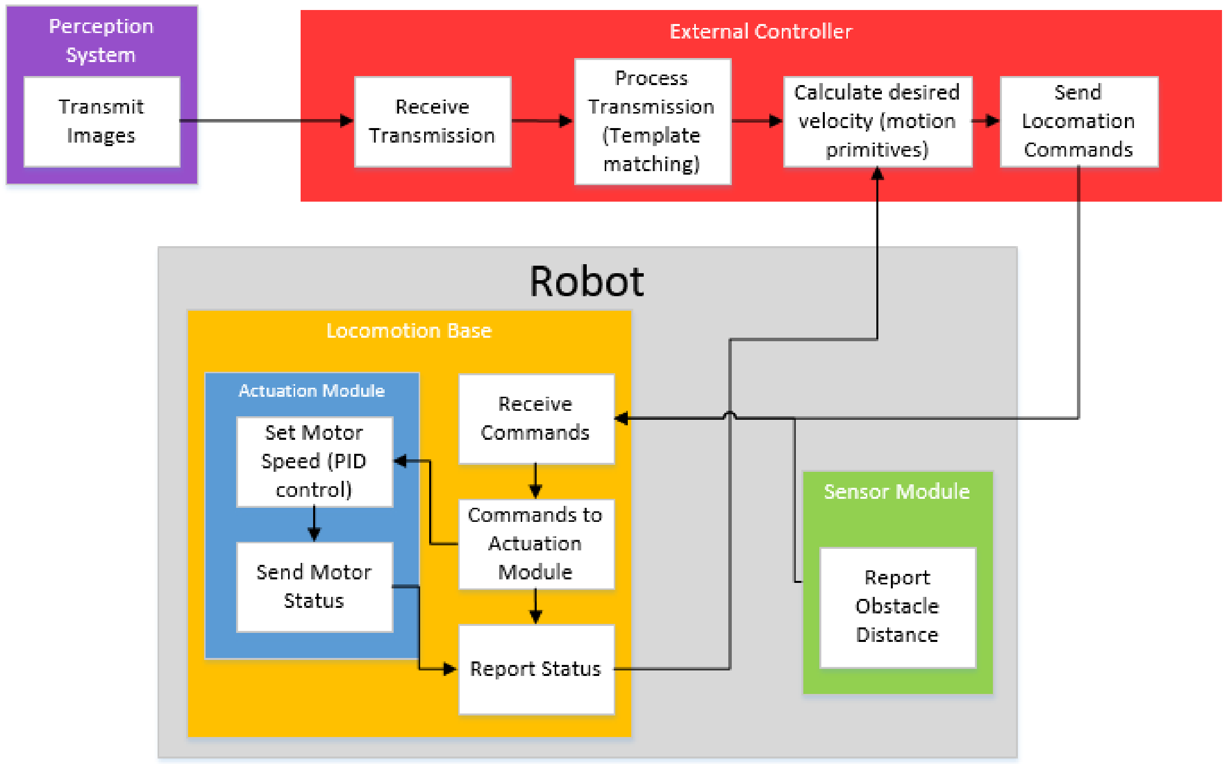


Figure 3: The OddBot Functional Architecture

The functional architecture is intentionally generic due to the multi-functional nature of the OddBot platform. In order to fit the use cases, the robot system must be capable of a wide array of tasks as well as accommodating user-designed modules. The primary functions of this robot are encapsulated in its external controller, perception system, locomotion base, sensor module, and actuation modules.

The OddBot system begins with vision or user input into an external controller. This controller is a computer communicating with the locomotion module over Ethernet. Once the locomotion module has received commands from the external controller it will process the transmission and send appropriate commands to the actuation modules. The actuation modules will have motors and wheels to move the robot as commanded by the user and report motor status back to the external controller through the locomotion module. The perception system will send data gathered by its sensors to the external controller through the locomotion module. The input as images can be changed for dead reckoning to a user provided goal. The external controller acts as the interface between an external perception system or a user and the locomotion base.

5. Trade Study

In the fall we conducted trade studies to identify hardware to incorporate into the base. This included types of modules, processor type for the brain, and the connection protocol.

For type of modules we found that wheeled or tracked modules were going to be the best option given our budget, time, and requirements. In the end we chose wheeled module because of the ease of fabrication compared to a tracked module.

The brain module was an onboard processing unit that dealt with all of the modules. This semester it was switched to be an external controller off board and the locomotion base dealt with module handling. We had chosen and Intel i3 to be the brain because of the computing power and the onboard vision requirements of the robot.

Finally, in the fall semester we had looked at various connection protocols: UART, Ethernet, USB, Wi-Fi, and Bluetooth. We looked at bandwidth, ease of configuration and compatibility. With these metrics Ethernet came out as a clear lead as it had high data transfer rates and was easily compatible with most arm processors.

Keeping these in mind we conducted a few more trade studies to account for the changes made to the base. We conducted trade studies for the new actuation and locomotion base as well as a connector type.

5.1 Actuation module:

For the updated base and the addition of a new actuation module we found that the processing units needed to be reevaluated. Table 1 shows the trade study metrics used to evaluate the processing units.

Table 1: Actuation Module Processing Unit Trade Study

Microcontroller	Cost	Ethernet Speed	Ease of PWM	Documentation	Ease of Interrupts	Size	Total
Weights	0.2	0.1	0.1	0.2	0.1	0.3	1
Udoo [3]	1	5	5	3	4	1	2.5
Raspberry Pi [4]	5	5	3	4	3	5	4.4
BeagleBone Black [5]	4	4	5	4	5	5	4.5
Odroid [6]	5	5	3	3	3	5	4.2

From the trade study we found that the BeagleBone Black, Raspberry Pi and the Odroid all had relatively close scores. Since we had some BeagleBones around the lab we decided that we would start with those. This ended up working well since they had the built in hardware encoder counters which we could use to read the quadrature encoders from the motor.

5.2 Locomotion Base:

Similarly to the actuation module the team also performed a trade study for the choice of microcomputer for the updated locomotion base, seen in Table 2. The choice of parameters is slightly different than the previous study with processing speed and USB port number replacing Ethernet speed, ease of PWM, and ease of interrupts. Size and cost remained the same.

Table 2: Locomotion Base Processing Unit Trade Study

Microcontroller	Cost	Documentation	Size	Processing Speed	USB number	Total
Weights	0.2	0.1	0.3	0.1	0.3	1
Udoo [3]	1	3	1	4	4	2.4
Raspberry Pi [4]	5	4	5	4	4	4.5
BeagleBone Black [5]	4	4	5	3	2	3.6
Odroid [6]	5	3	5	5	4	4.5

From the trade study above it can be seen that the Raspberry Pi and the Odroid scored well above the others. Since we had neither on hand we decided to order the Odroid as it was slightly faster.

5.3 Connector Type

Due to the difficulty of establishing a connection in the fall, we conducted a trade study to determine a better method.

	Active (require control software/circuitry)	Inactive (require no added software/circuitry)
Mechanical	- Servo driven rotary latch - Lead screw driven lock	- Adhesive - Rods, hooks, or pins for mechanical "snap" connection - Threaded connection point - Slide rails - Un-powered mechanical linkage - Spring-driven latches
Magnetic	- Electromagnets - Electro-permanent magnets - Mechanically switched permanent magnets	- Permanent Magnets

Figure 4: Connector Type Comparison Matrix

Table 3: Connector Type Trade Study

Connection Type	Complexity	Cost	Fabrication Time	Reliability	Ease of disassembly	Total
Weights	0.2	0.1	0.2	0.2	0.3	1
Servo driven mechanical rotary latch	1	2	2	5	3	2.7
Lead screw driven lock	5	5	4	5	3	4.2
Rod/Hook/Pin “Snap” Connection	4	4	2	5	5	4.1
Electromagnets	3	1	3	4	5	3.6
Permanent magnets	5	4	5	3	5	4.5

Many different connectors were considered for the second semester. There are several different connection methods which can be roughly broken down into the matrix seen in Figure 4 and quantitatively compared in Table 3.

Active mechanical connections had the advantage of being very strong once locked, only requiring power to switch states from locked to unlocked and vice versa. The disadvantage of these types was that they add bulk, complexity, and require additional code and hardware to lock and unlock.

Inactive mechanical connections were considered simpler to create, more intuitive, and required no additional software. The primary trade-off for these kinds of connections as determined by the trade study was in ease-of-use vs. durability. Simply using screws to secure every module to the other modules would have been very secure, but less use-able for a novice user and would be against the general idea of a “user-friendly modular robot”, as it would have required using tools to disassemble modules. Simpler plastic or metal snaps offered a very easy to operate connection, but little mechanical rigidity. Additionally, fabricating these types of snaps to offer the exact right amount of connection durability would have taken significant time.

Active magnetic connection and locking mechanisms allowed for no moving parts (with the exception of mechanically-switched permanent magnets), could be durable depending on the level of current applied to the electromagnet, and would not have taken up a large amount of room. However the cost and additional electrical and software complexity for controlling these types of connectors was considered too large a commitment [7].

Permanent magnets were a very cheap, low complexity, and small solution. The strength of the magnet would determine how durable the connection is as well as how easy it is to remove the module, and could not be controlled directly through software means. Because of this simplicity in construction and use, permanent neodymium magnets were chosen to mechanically lock modules in addition to module ports that lock in the modules based on a determined module size.

Pogo pins were selected for the electrical connection between modules, as the pin count could be increased or decreased based on design requirements, took up little room, and were easy to wire up.

6. Cyber-Physical Architecture

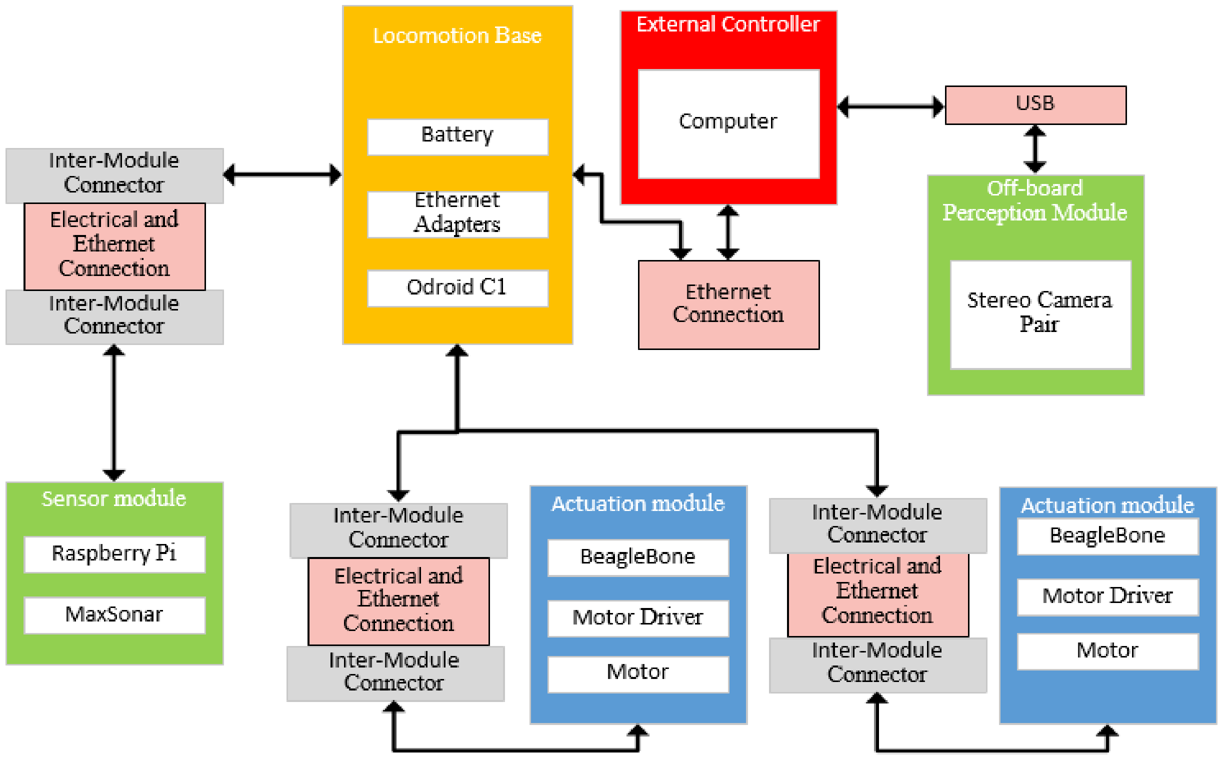


Figure 5: The OddBot Cyber-Physical Architecture

The cyber-physical architecture, shown in Figure 5, was informed by the system trade studies. The locomotion base acts as the central control, taking in commands and interpreting what should be done for the actuation modules. This locomotion module housed a battery which provided power to all of the on-board modules. The central processing was an Odroid, which had 4 Ethernet-to-USB connectors to connect to the modules via Ethernet.

The actuation module included a BeagleBone Black to communicate with the locomotion base, and to calculate the desired motor velocity via an onboard PID loop. A PWM output went to a DC motor driver, which provided power to the DC motors.

The sensor module housed a Raspberry Pi and an ultrasonic range finder that would detect obstacles and tell the locomotion base to turn off the motors, preventing the base from crashing into obstacles.

The perception module included two webcams to perform template matching. The external controller used the predicted robot pose from the perception system to generate a trajectory based on the goal pose. It then calculated the desired velocity of the robot and sent it to the locomotion base.

7. System Description and Evaluation

7.1 Current System and Subsystem Descriptions

7.1.1 Overall System Description

The complete OddBot system can be seen in Figure 6 and Figure 7. The locomotion base is the primary robot structure, the sensor module is located in the front port of the locomotion base, and the two actuation modules are located in the left and right module ports. All modules are connected to the locomotion base through a proprietary electro-mechanical male-female connector pair. Each module port on the locomotion base has a female connector, and each module has a corresponding male connector. Any module with a male connector and the appropriate magnetic polarity can be plugged into any of the four module ports of the locomotion base.

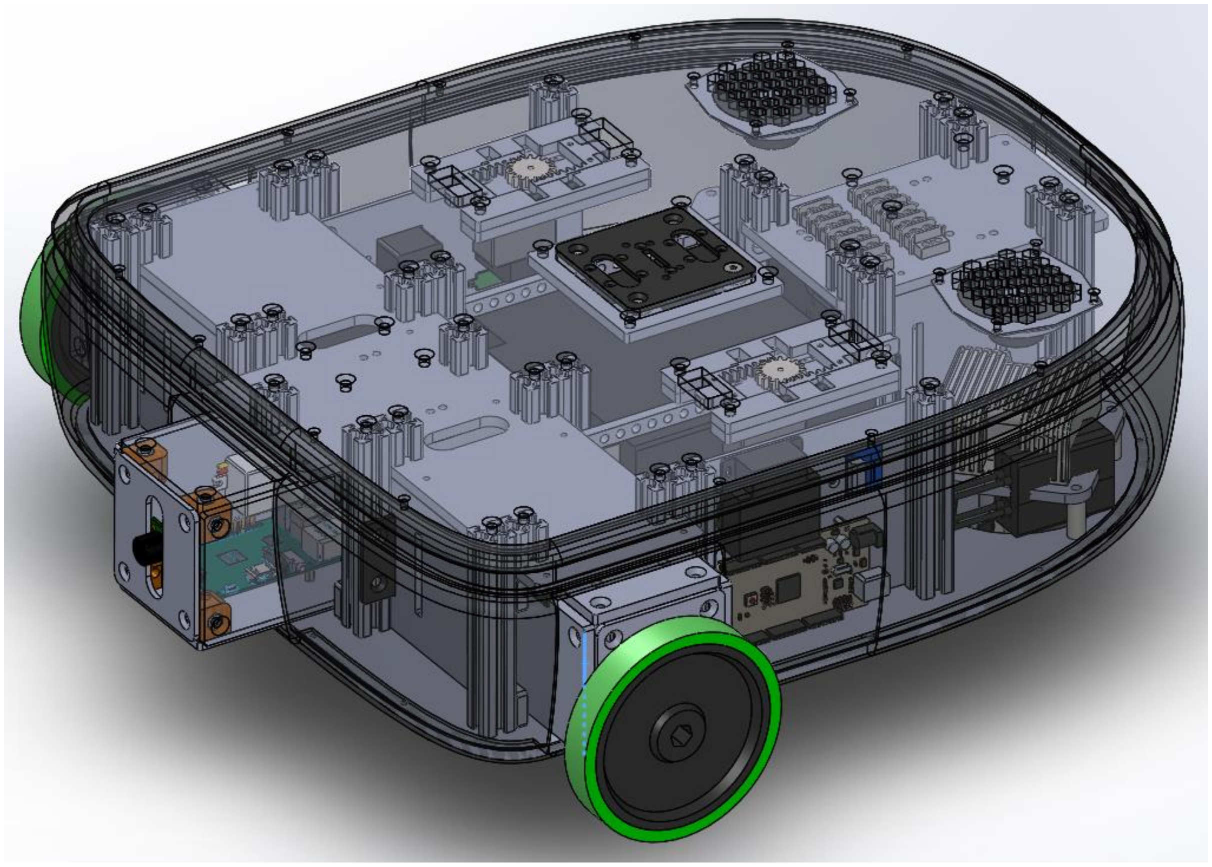


Figure 6: Fully assembled OddBot design

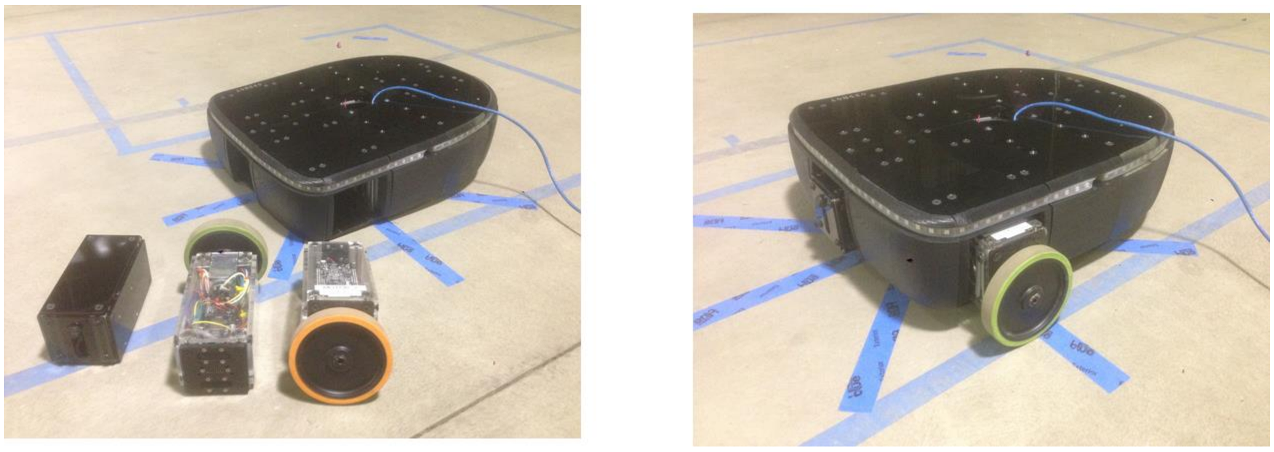


Figure 7: Fully assembled and dis-assembled OddBot robotic system

Figure 8 depicts the general information architecture of the OddBot system. A more detailed explanation of the software in each subsystem is given in sections 7.1.3 – 7.1.6.

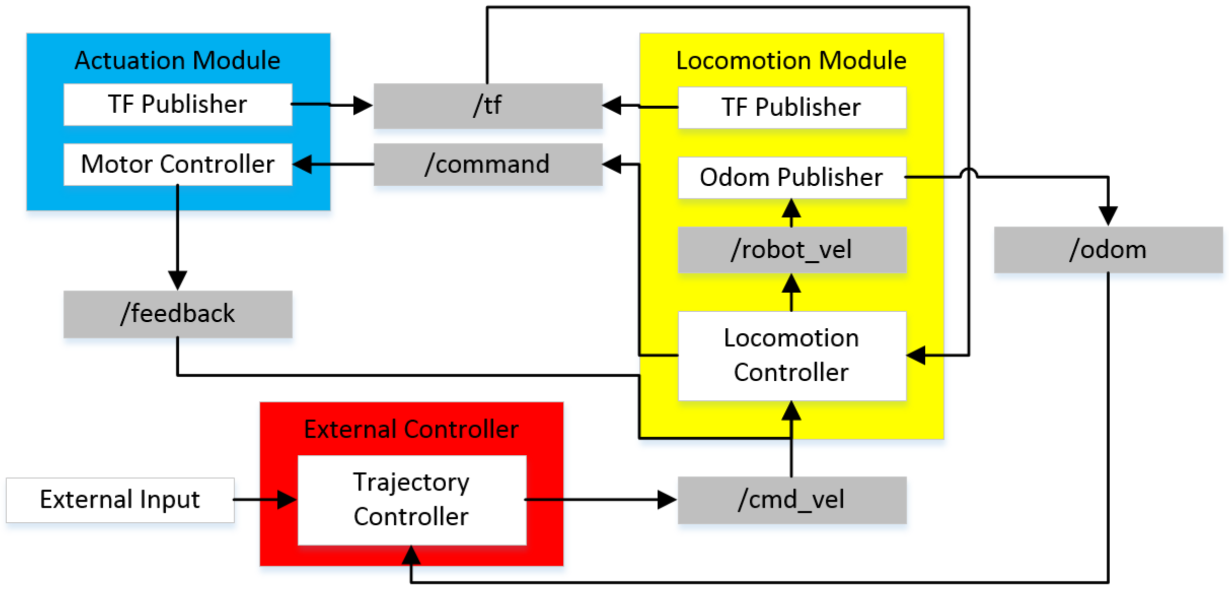


Figure 8: Information Architecture of the OddBot System

7.1.2 Connector Description

Every module has a male electro-mechanical connector that can be used to attach to a corresponding female connector located in one of the four module ports of the locomotion base. Both connectors use neodymium magnets to create a secure mechanical and electrical connection, as well as determine correct module orientation. The electrical connection consists of four ground pins, four 12V pins, and eight pins corresponding to each of the eight wires of an RJ45 Ethernet wire. The electrical connection is executed through male and female pogo pins. Pinouts for the male and female pogo pins can be seen in Figure 9 and Figure 10. A CAD depiction of the male and female connectors can be seen in Figure 11 alongside an assembled male connector.

GND	GND	PWR	PWR	White w/orange stripe	White w/blue stripe	White w/green stripe	White w/brown stripe
GND	GND	PWR	PWR	Orange	Blue	Green	Brown

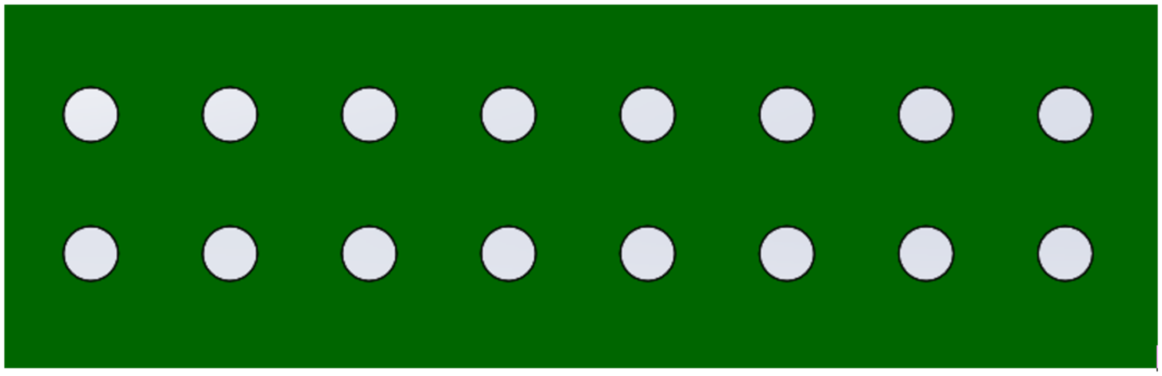


Figure 9: Female Connector Pinout

White w/brown stripe	White w/green stripe	White w/blue stripe	White w/orange stripe	PWR	PWR	GND	GND
Brown	Green	Blue	Orange	PWR	PWR	GND	GND

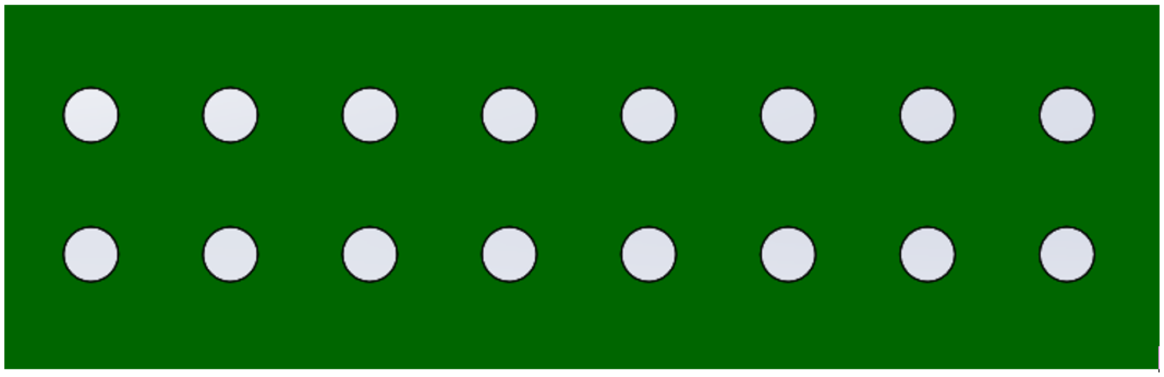


Figure 10: Male Connector Pinout

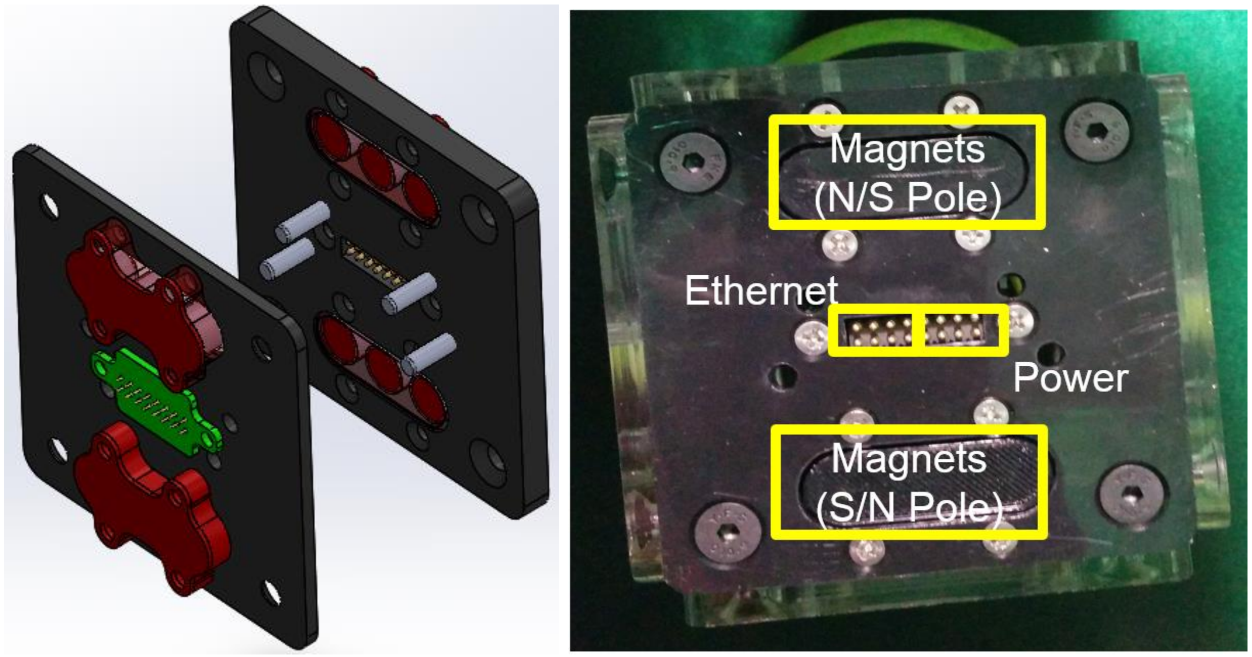


Figure 11: CAD Model of Male and Female Connector and an Annotated Male Connector

7.1.3 Locomotion Base Subsystem

The locomotion base is the central module to the OddBot platform. A CAD model can be seen in Figure 12.To ensure the capability of the entire platform, the following elements are contained within the module: a computing element, battery, and female connectors at every module port. This subsystem is also the primary point of user interaction and currently provides power to the other modules.

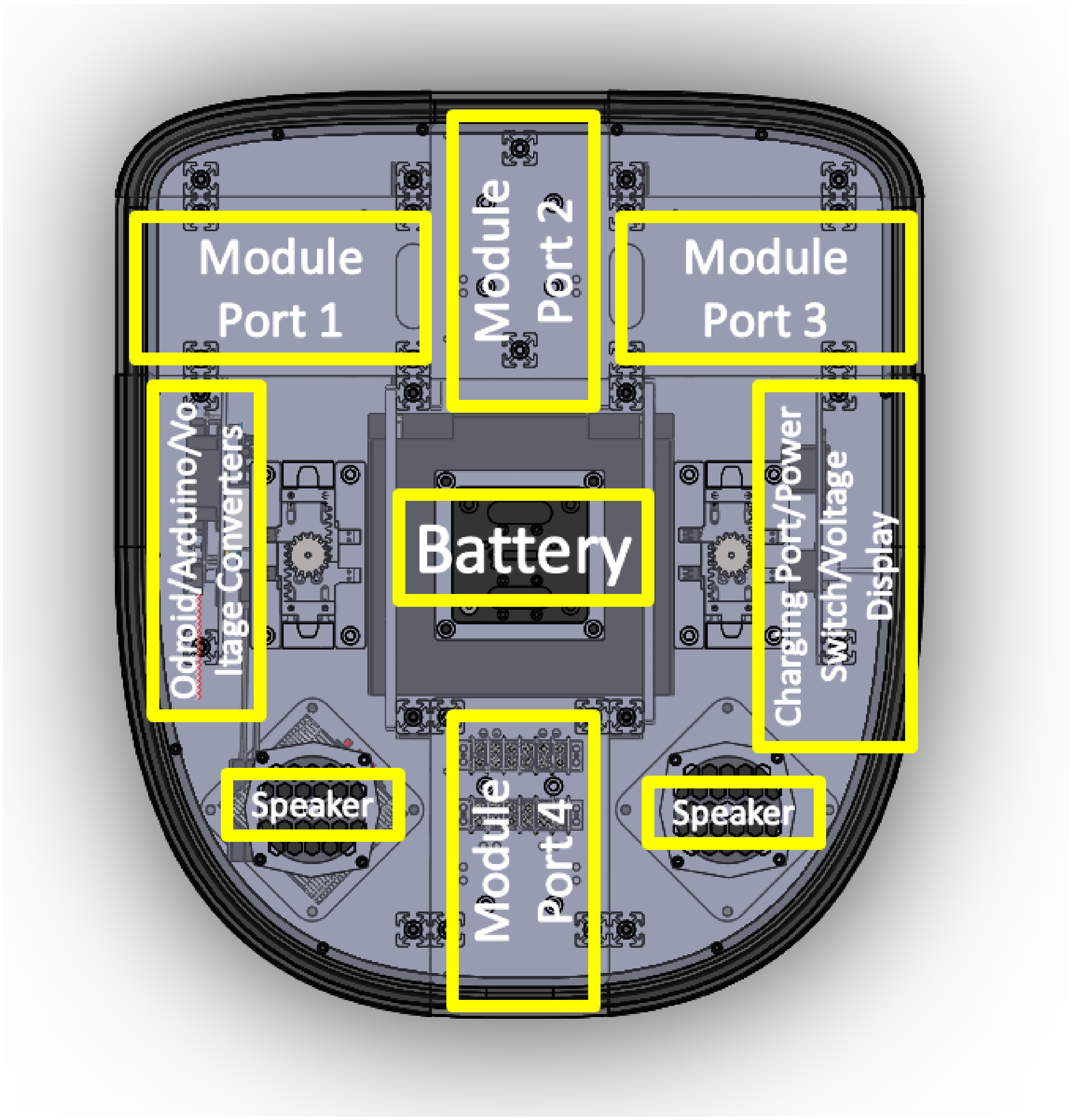

Module Port 1
Module Port 2
Module Port 3
Odroid/Arduino/Voltage Converters
Battery
Charging Port/Power Switch/Voltage Display
Speaker
Module Port 4
Speaker

Figure 12: Top View of Model of Locomotion Base

An Odroid-C1 was determined to be the best computing platform for the locomotion base due to its acceptable computing performance, low power consumption, large number of USB connectors, small form factor, and ability to run ROS/Ubuntu. Each module has an Ethernet and power connection to the locomotion base. Connections between the Odroid and different modules are handled via USB-Ethernet connectors. An Arduino Mega was also included as a controller of speakers and LEDs used for enhanced user interaction.

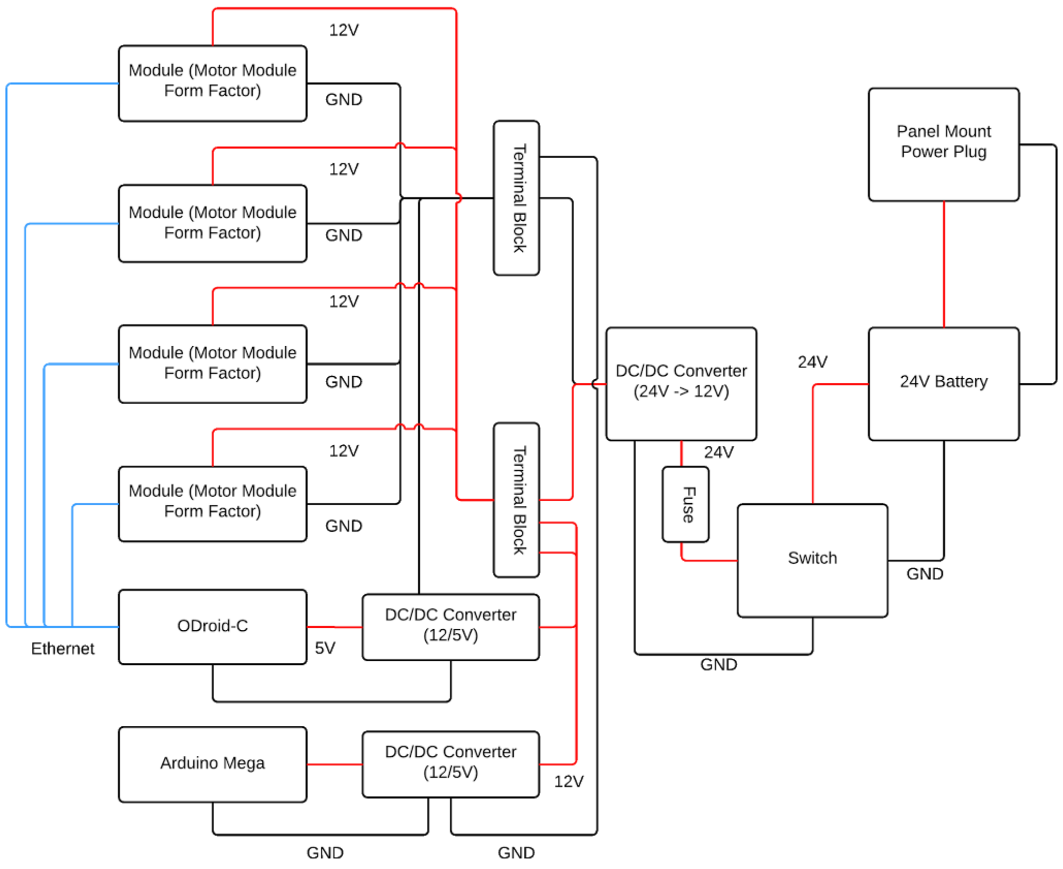


Figure 13: Electrical Subsystem for Locomotion base

The electrical subsystem of the locomotion base can be seen in Figure 13. The locomotion base is powered by a 24V 10Ah Li-ion battery. This battery was chosen as it allowed for several hours of robot uptime. The battery is charged using a panel-mounted power plug, and can be turned on using a high-amp toggle switch. The battery is connected to a 24/12V DC/DC converter, with an on-line fuse to prevent any current surges from damaging the rest of the robot electrical system. The DC/DC Converter leads to a 12V terminal block and a ground terminal block. These terminal blocks provide 12V and ground to all other electrical components of the locomotion base. All four module connection ports are powered by the 12V and ground. Both the Odroid-C1 and Arduino Mega are powered through an additional 12V/5V DC/DC Converters.

The locomotion base was in charge of module management for the connected modules. It had to make sure that only connected modules would receive commands and would not listen to modules which were not connected. To do this the locomotion base would check every 10 seconds if a new actuation module was connected or if one has been removed. It checked based on the following function depiction below, in Figure 14.

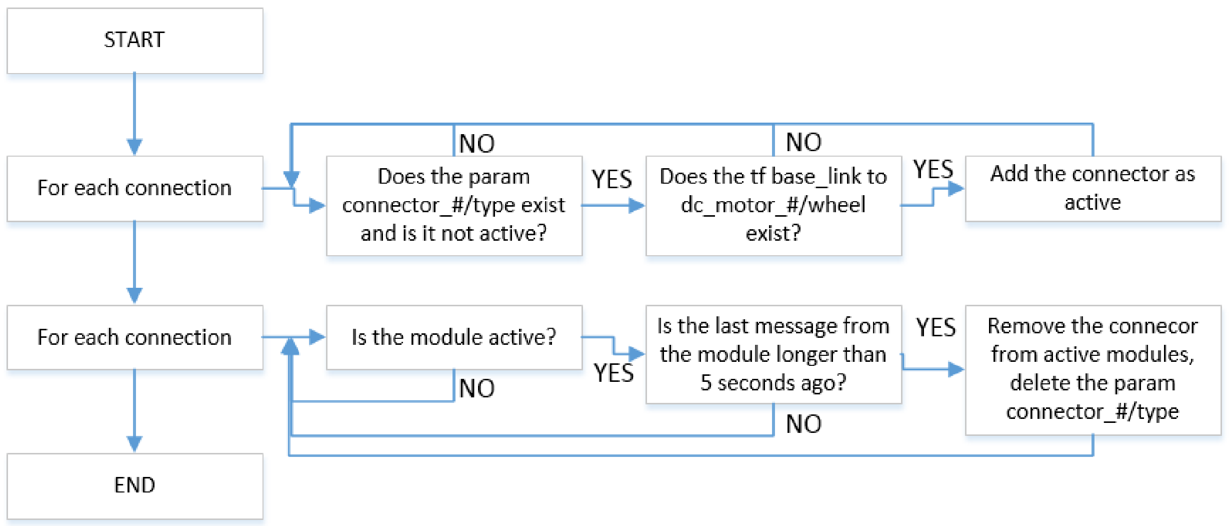

START
For each connection
Does the param connector_#/type exist and is it not active?
NO
YES
Does the tf base_link to dc_motor_#/wheel exist?
NO
YES
Add the connector as active
For each connection
Is the module active?
NO
YES
Is the last message from the module longer than 5 seconds ago?
YES
NO
Remove the connecor from active modules, delete the param connector_#/type
END

Figure 14: Flow diagram of Locomotion Base module check function

If modules were connected then the locomotion base would calculate the desired wheel velocities based on the following equations and the transformation Figure 15 depicted below.

$$wheel_{velocity} = vx * \sin\big(atan2(y,x)\big) + vy * \cos\big(atan2(y,x)\big) + w * r \qquad (1)$$

Vx is the desired linear velocity along the x-axis in meters per second, vy is the desired linear velocity in the along the y-axis in meters per second and w is the desired angular velocity about the z-axis in radians per second. X is the distance of the wheel in meters from the base link along the x-axis, y is the distance of the wheel in meters from the base link along the y-axis, and r is the shortest distance in meters from the base link to the wheel.

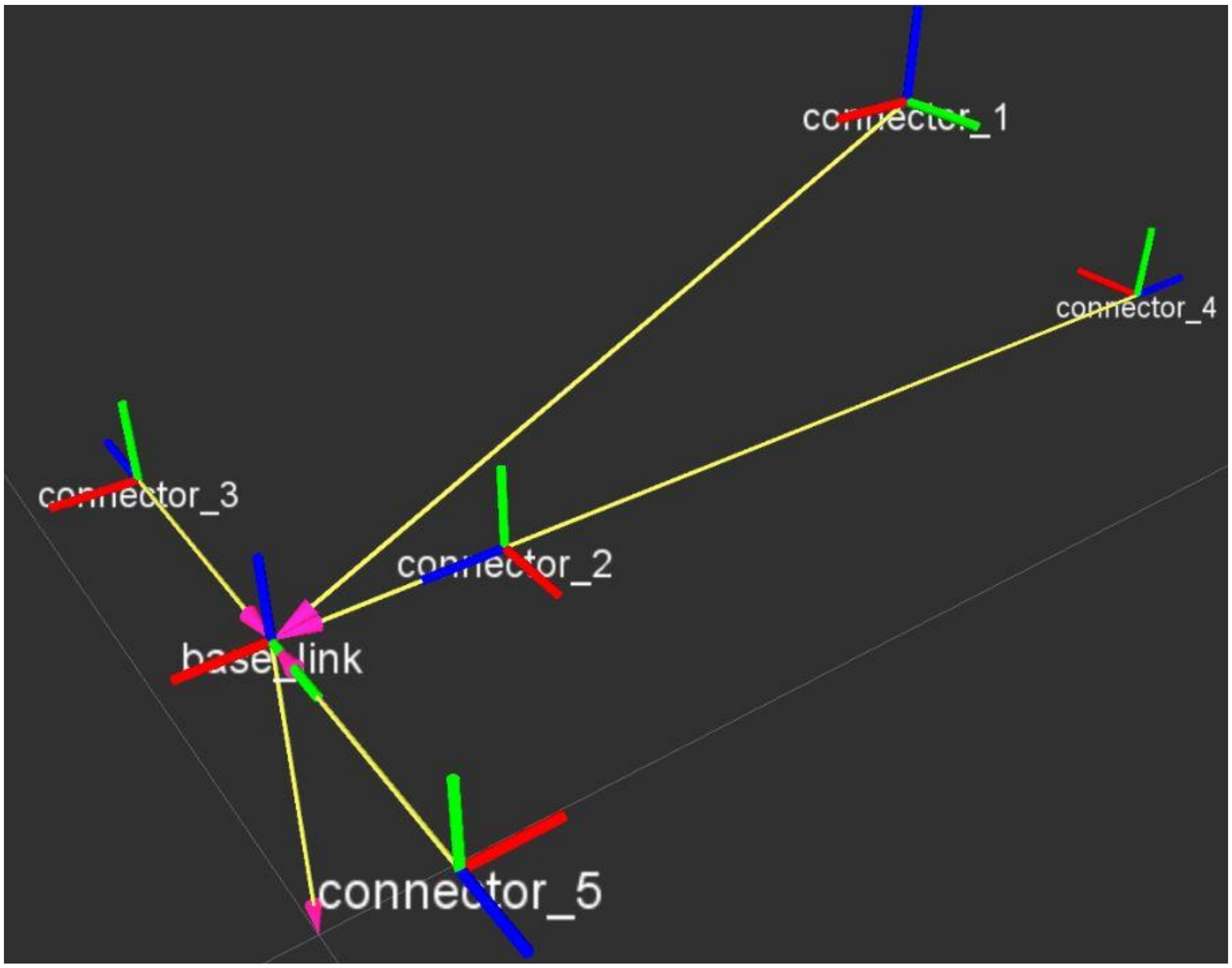


Figure 15: Depiction of the Locomotion Base's Transformation Frame

The transformations were published by the locomotion based are depicted above. When a module was connected, for example the actuation module, a transformation from the connector to the center of the wheel was published. This made it possible to look up the transform from the base_link to the connected wheel and perform the calculations above.

The robot then uses the inverse of the equation above to calculate the robot velocity. These equations are:

$$robot_x = \frac{\left(\sum_{num_{motors}} \frac{wheel_{velocity}}{\sin(atan2(y,x))}\right)}{num_{motors}} \quad (2)$$

$$robot_y = \frac{\left(\sum_{num_{motors}} \frac{wheel_{velocity}}{\cos(atan2(y,x))}\right)}{num_{motors}} \quad (3)$$

$$robot_w = \frac{\left(\sum_{num_{motors}} \frac{wheel_{velocity}}{\mathrm{r}}\right)}{num_{motors}} \qquad (4)$$

Once the robot velocity is acquired, it is used to calculate the robot's odometry. This is calculated based on the robot velocity and performing discrete integration based on the time interval between calculations to estimate the robots actual change in position from when it started. The equations we used to calculate odometry were equation (5) to (11).

$$\Delta t = current_{time_s} - last_{time_s} \qquad (5)$$

$$\Delta x = \left((vx * \cos th) - (vy * \sin(th))\right) * \Delta t \qquad (6)$$

$$\Delta y = \left((vx * \sin(th) - (vy * \cos(th))\right) * \Delta t \qquad (7)$$

$$\Delta th = vth * \Delta t \qquad (8)$$

$$x = x + \Delta x \qquad (9)$$

$$y = y + \Delta y \qquad (10)$$

$$th = th + \Delta t \qquad (11)$$

7.1.4 Actuation Module Subsystem

The actuation modules allowed the locomotion base to move. Two identical actuation modules were built. The primary components of an actuation module are a DC motor, motor driver, BeagleBone Black, and 12V/5V DC converter. A male connector is also necessary in order to interface with the locomotion base, and a wheel to interface with the ground. All components of the actuation module are powered through the 12V electrical connection to the locomotion base through the male connector. The BeagleBone Black runs Ubuntu 14.04 and ROS, sending commands to the motor driver based on command velocities sent to it from the trajectory planner running on the locomotion base, and communicating encoder counts of the motor with the locomotion base. All communication between this module and the locomotion base are performed through the Ethernet pins of the male connector. A breakdown of the actuation module's components can be seen in Figure 16.

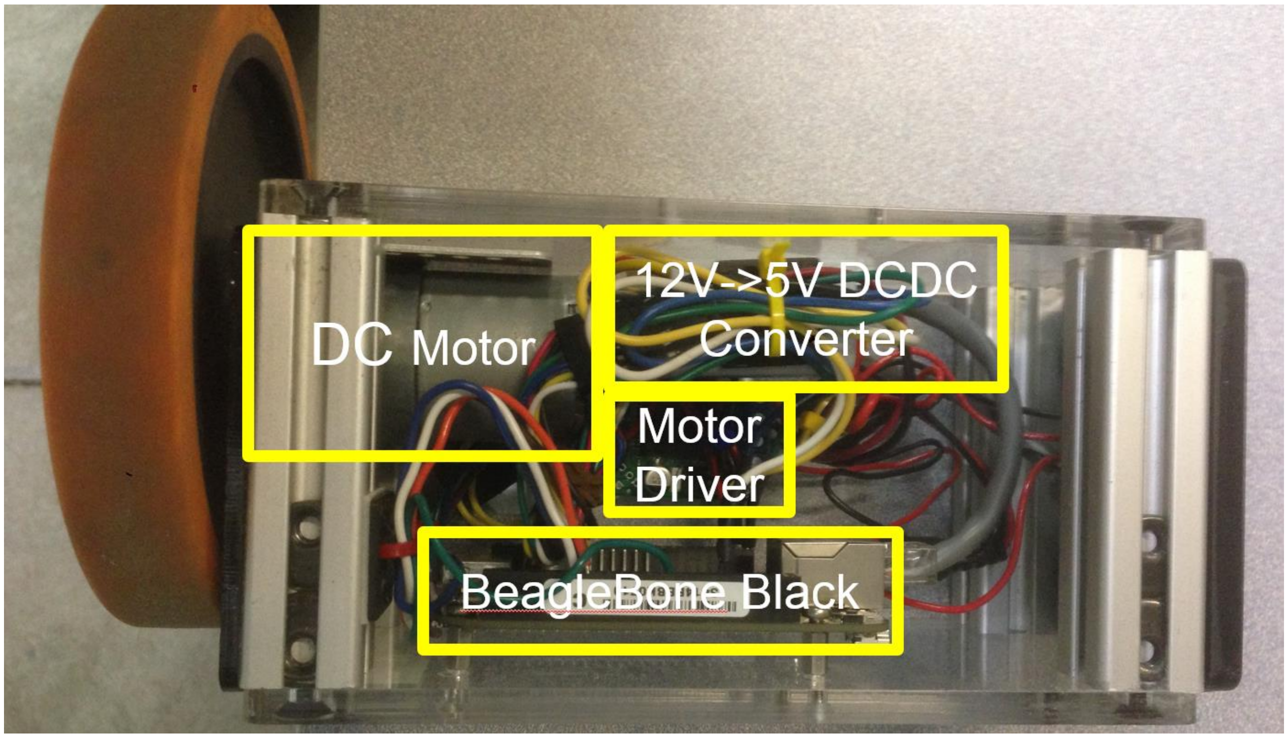


Figure 16: Actuation Module Subsystem

The electrical module receives 12V from the connector. This 12V is used to power the motors. However, it is necessary to step down the voltage to 5V to power the BeagleBone Black. The electrical connections are depicted in Figure 17 and Figure 18.

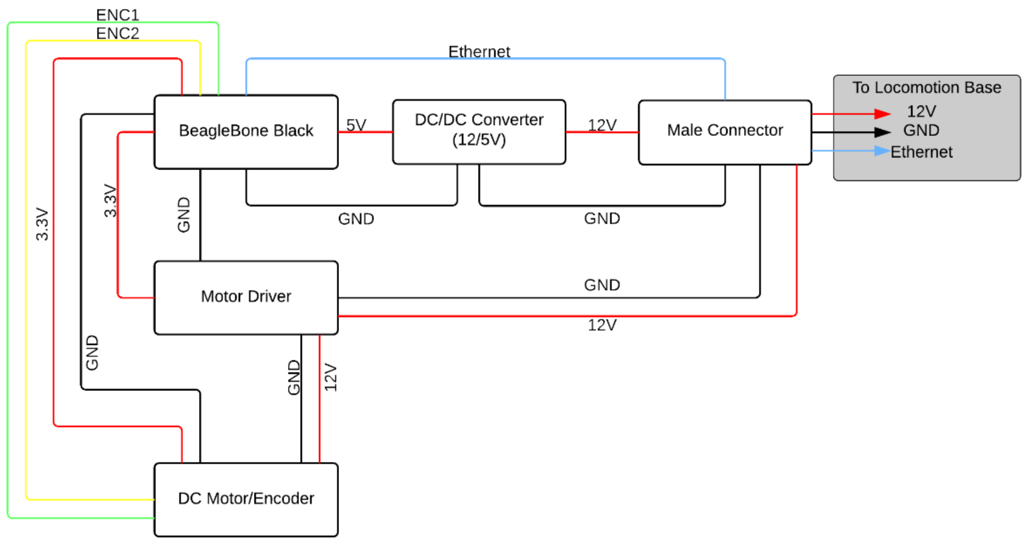


Figure 17: Electrical Schematic of the Actuation Module

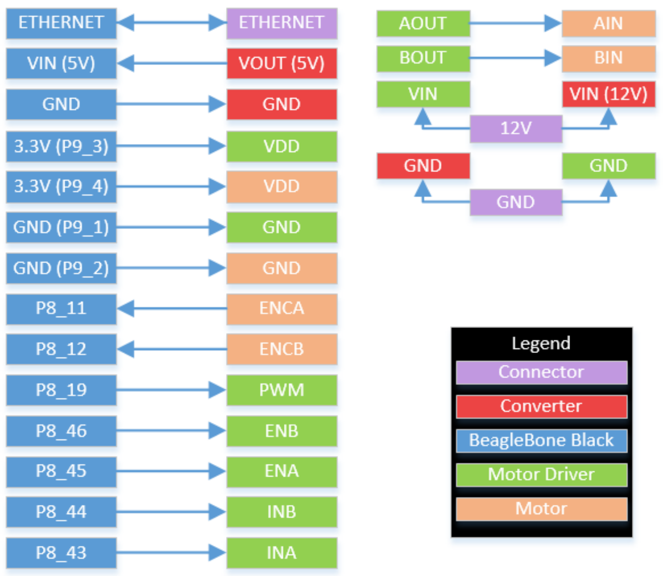
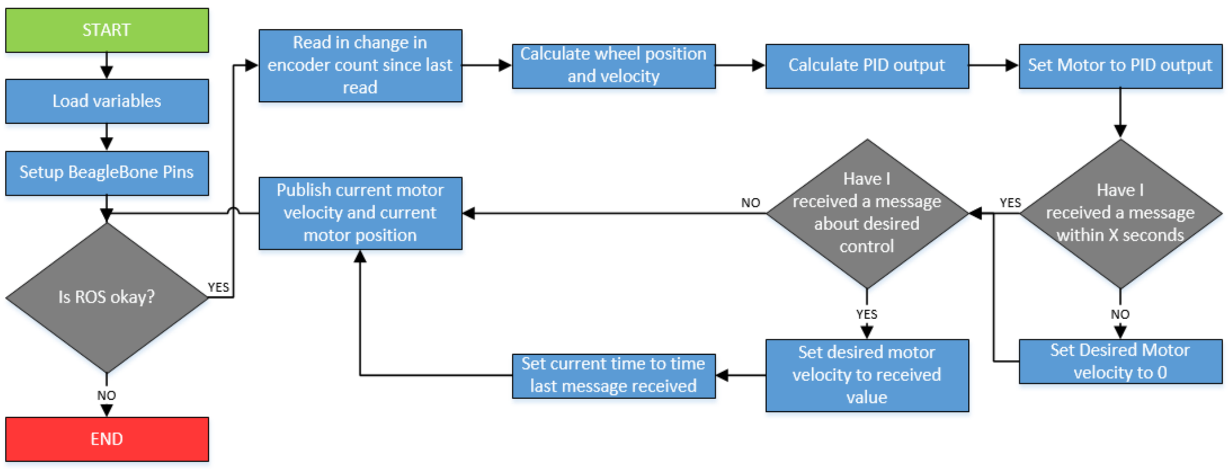


Figure 18: Motor Module Wiring Diagram Pinouts

The software on the actuation module was fairly adaptable. Based on the configuration file the motors could be controlled via a position or velocity controller. For our application we used the velocity controller. Figure 19 depicts the software loop running on the actuation module.



START
Load variables
Setup BeagleBone Pins
Is ROS okay?
YES
NO
END
Read in change in encoder count since last read
Calculate wheel position and velocity
Calculate PID output
Set Motor to PID output
Have I received a message within X seconds
YES
NO
Set Desired Motor velocity to 0
Have I received a message about desired control
NO
YES
Set desired motor velocity to received value
Set current time to time last message received
Publish current motor velocity and current motor position

Figure 19: Software loop for the Actuation Module

The actuation module received a desired wheel velocity. It used PID control to get the wheel as close as possible to that velocity. It then published the actual wheel velocity as feedback. If the module had not received a desired wheel velocity after a specified amount of time the program would set the desired velocity to zero and shut down the module. Throughout this another program was also publishing the appropriate transformations for that module, from the connector, which was used to calculate the wheel velocity in the locomotion module.

7.1.5 Sensor Module Subsystem

The perception module provided a level of sensing to the robot once plugged into the locomotion base. Similarly to the actuation module, it communicated with the locomotion base and was powered by the locomotion base through the pogo pins of the connector. The sensor module consisted of a Raspberry Pi running Ubuntu 14.04 and ROS, a 12V/5V voltage converter, and a MaxSonar ultrasound sensor. The ultrasound sensor sends back distance information to the Raspberry Pi through serial. The Raspberry Pi in turn converts this information into ROS messages which it publishes to the locomotion base. If an obstacle is sensed by the ultrasound sensor below a certain threshold, the OddBot will stop in place until the obstacle is removed. Figure 20 contains a breakdown of the perception module components.

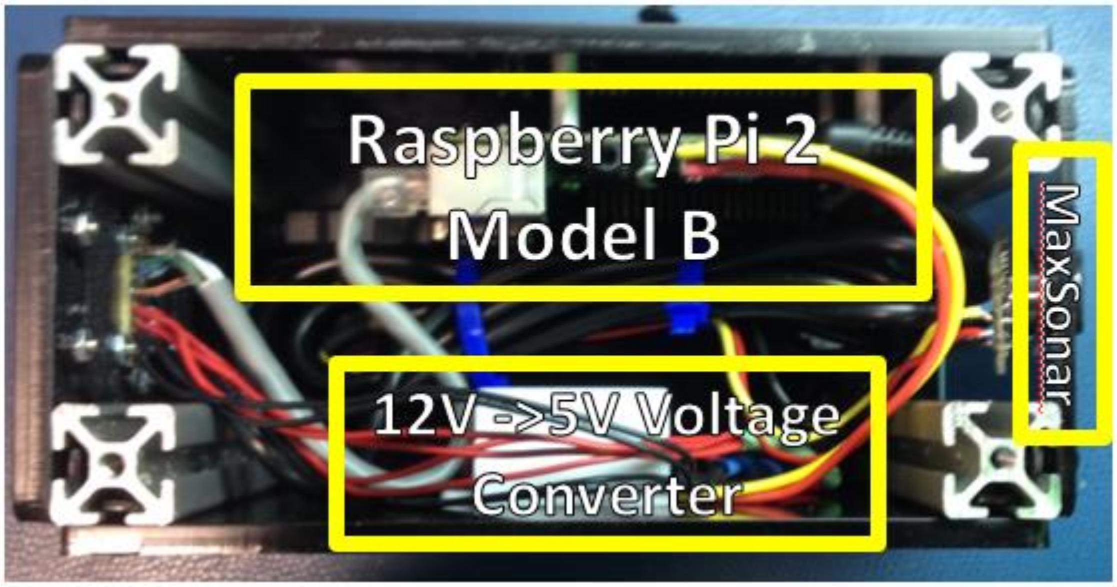


Figure 20: Inside of the Perception Module Depicting Internal Hardware

The electrical layout of the software module was also simple, the 12V input from the connector was stepped down to 5V which supplied the Raspberry Pi and the MaxSonar with 5V power. The electrical drawings can be seen Figure 21 and Figure 22.

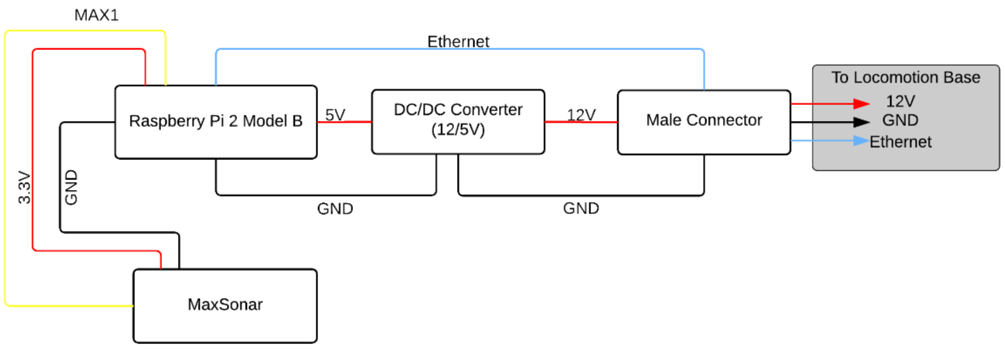


Figure 21: Electrical Schematic for the Sensor Module

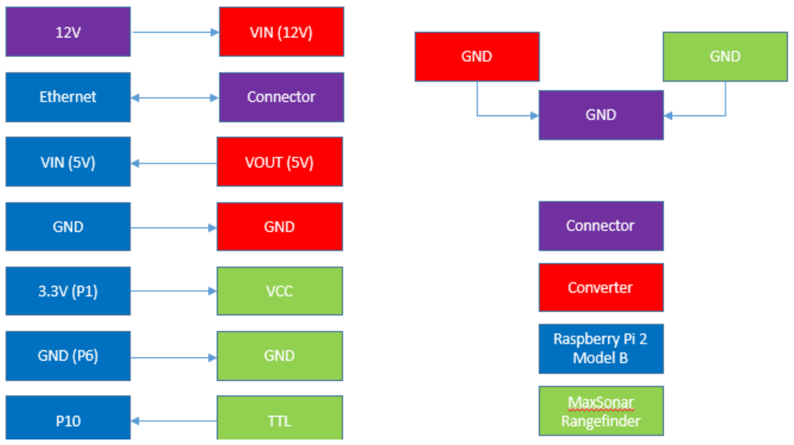


Figure 22: Sensor Module Pinout

The sensor module used a simple software loop running one ROS node publishing sensor data. It provided information about the surrounding environment but did not take in any information. The software loop that ran onboard can be seen in Figure 23.

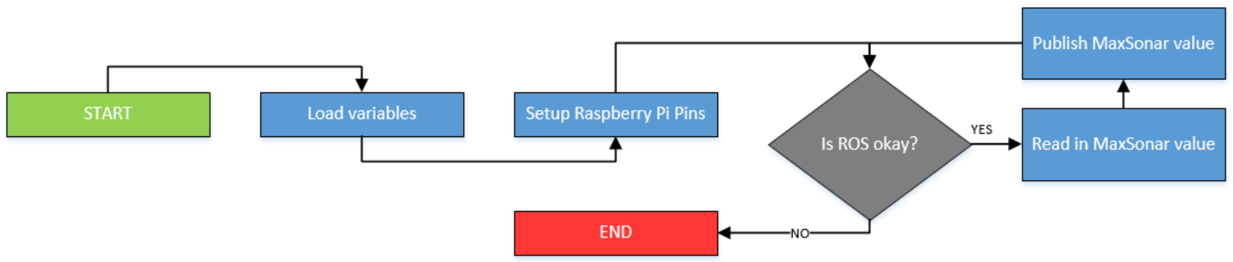


Figure 23: Sensor Module Software Loop

7.1.6 External Controller Subsystem

The external controller can either take in user-given goals or goals given by the perception system. These are desired poses, position and orientation, of the robot. The external controller publishes to the command velocity topic. This topic contains the desired velocity of the robot. The desired velocity is calculated using the desired goal, motion primitives, and the robot's odometry. An example of motion primitives can be seen in the Figure 24.

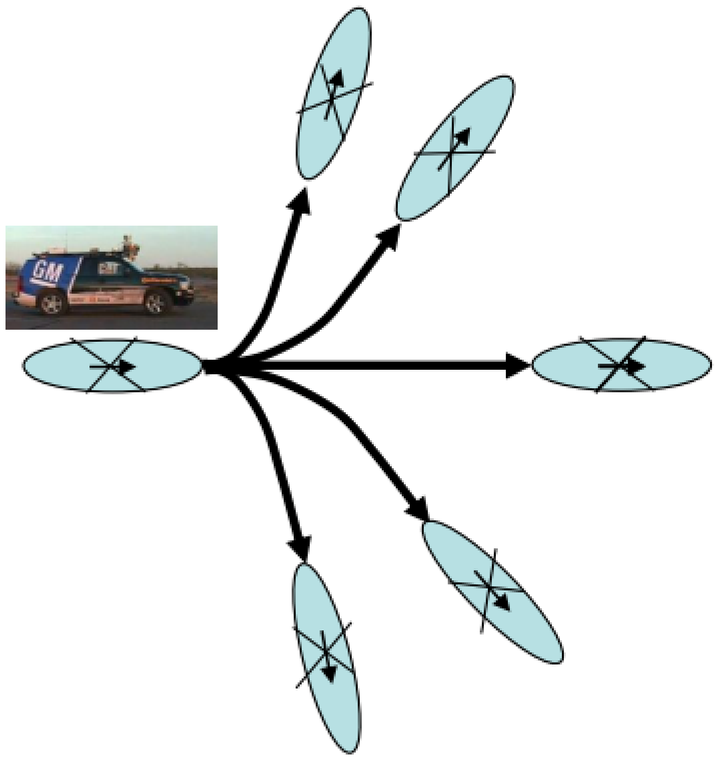


Figure 24: Example Motion Primitives for a Differential Drive Robot [8]

Using the odometry we could calculate the relative location of the goal from our current location. We could then compare a subset of the pre-calculated primitives, based on our current velocity, to the desired goal path. We compared to a subset of the primitives to limit the max change in velocity the robot could choose to execute each iteration. The smallest cumulative distance between the primitive waypoints and the goal path waypoints was chosen. This allows the robot to correct for small changes of the robot in the execution of the path. The software loop for the execution of the motion primitives can be seen in the Figure below.

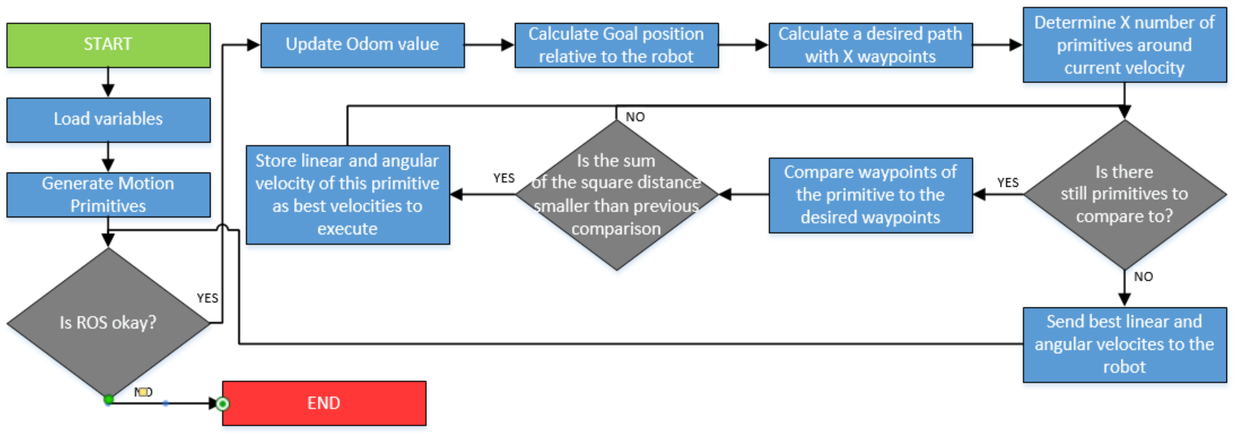

START
Load variables
Generate Motion Primitives
Is ROS okay?
YES
NO
END
Update Odom value
Calculate Goal position relative to the robot
Calculate a desired path with X waypoints
Determine X number of primitives around current velocity
Is there still primitives to compare to?
YES
Compare waypoints of the primitive to the desired waypoints
Is the sum of the square distance smaller than previous comparison
YES
NO
Store linear and angular velocity of this primitive as best velocities to execute
NO
Send best linear and angular velocites to the robot

Figure 25: Software Loop for the External Controller

Using motion primitives would correct the robot after it rotated when it initially started to move, which was a result of the two wheels not starting perfectly at the same time.

7.2 Modeling, Analysis, and Testing

7.2.1 Connection method

The type and number of neodymium magnets used in the male and female connectors was chosen to allow for a sturdy but hand-removable module connection. Six magnets rated at a pull-force of 9.5 lbs. are used in the male and female connector plates. In an ideal connection, the total tensile force needed to remove a module would be:

9.5lbs. * 6 = 57lbs. (magnet-magnet attraction strength is equal to magnet-iron attraction strength)

Due to spacing between the magnets, the actual tensile connection strength was measured at ~20lbs. Axial connection strength was significantly higher than the tensile connection strength as every module was locked in axially by the structure of the locomotion base.

7.2.2 Battery

The 24V 10Ah battery used in the locomotion base was acquired based on requirements from last semester's robot design. The total available energy to the system (assuming perfect efficiency) was calculated to be:

24V * 10Ah * 3600 sec/hr. = 864,000 J

To ensure that it met all electrical requirements we calculated the robot power requirements as can be seen in Table 4. The total average power draw of the robot was 16.5W. Thus the theoretical uptime of the robot on one battery charge was 864,000J/16.5W = 52363 sec, or roughly 14.5 hours. In reality the robot was able to run on average for 6-8 hours on a single battery charge, which was well within the acceptable operating time for testing and demonstration.

Table 4: Average Power Draw of Major Components on the OddBot

Component	**Average Power Draw (W)**
Odroid-C1	0.75
Arduino Mega	0.5
Actuation Module: BeagleBone Black, Motor	6.75 (with motor running under load)
Actuation Module: BeagleBone Black, Motor	6.75 (with motor running under load)
Sensor Module: Raspberry Pi, MaxSonar	0.75
LED Strip	1
Total	16.5

7.2.3 Latency tests

We also investigated alternatives to ROS. We tested the highest rate at which data sent from the BeagleBone could be received by another computer. We tried to find an alternative to ROS as ROS is built for a predefined robot configuration. With our modular system, Ethernet and networking quickly became a difficult aspect of the robot. We wanted another system which was capable of delivering comparable data transfer rates to ROS. To do this we tested REST, Google Protocol and ROS. First we tried to set up Google Protocol but found that a publish-subscribe system was not readily available. We then tried REST, but found that the transfer rate was only 2Hz between the BeagleBone Black and another PC. We finally tested ROS and found that we could publish at 200Hz. To ensure minimal time delay down the project we ended up with ROS.

7.2.4 PID tests

One thing that we needed to do was to tune the PID value. We ended up trying to tune the PID controller. In the end the I-value was very small. This meant that the controller was practically a PD controller. We tuned for the fastest response time, we ended up with a response plot seen in the figure below.

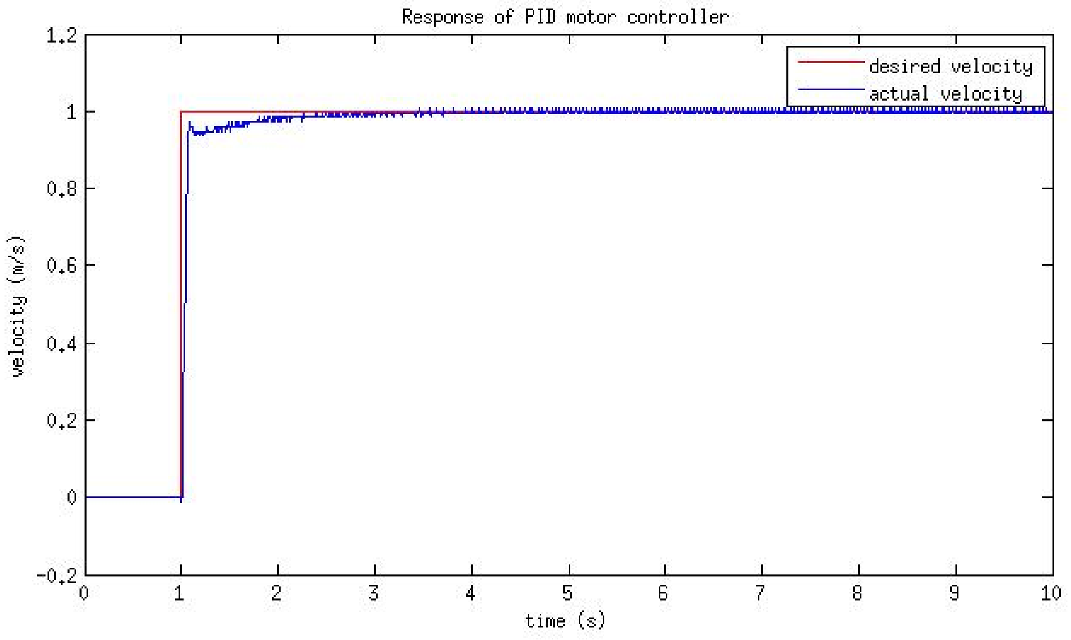


Figure 26: Response Plot of the Velocity PID Controller in the Actuation Module

These values provided a high response rate with zero oscillation with a change in velocity up or down. Some gains were fine when the motor sped up but when the desired velocity was set to zero the motors would oscillate. In the end we had gains of P: 300, I: 0.001, and D: 5.5.

7.2.5 CV template testing

We decided to use template matching to find the robot in the visual field of the cameras. This was originally to bind it for triangulation, so we could be sure we only got feature points within the robot on the image. We also wanted to make sure we could estimate the robot's rotation, since this would feed into the eventual path we wanted it to follow to the goal.

We tested several forms of template matching to find ways of gauging the strongest response. We wanted to find the strongest response not only among the rotation templates, but among scales of those templates because the robot could be farther away from the camera and so be smaller in the image. From testing we found that template matching using Pearson's Coefficient was the best at finding the robot in the frame. We also found that we needed to normalize the results from template matching to the area of the template, so that we could find the best result among scales.

This testing was not robust enough, however, as we continued to get false positives when the scene was changed (objects or people entered the field of view of the cameras) and the scene changed almost every day during the SVE.

7.2.6 CV Triangulation

We initially wanted to use the cameras to triangulate the position of the robot in such a way that it could give us a reasonable estimate of its position in x and y on the ground. We ran tests to find if we could repeatedly find an XYZ position of the robot using triangulation, but we found that there was not enough correspondence.

To get these positions we first found SURF features inside the 'found' robot bounding boxes, and then matched them to the other image on the second camera (also within its bounding box). We then computed triangulation using the projection matrices we had from calibrating the cameras. Unfortunately even when we manually entered in extrema points from the 2D plane (the edges of our box on the ground) we could not get XYZ coordinates that could be transformed into a 2D position.

7.3 Performance Evaluation against the Fall Validation Experiment

To show the capabilities of the modular robotics platform we conducted the following series of tests.

Part A – Test assembly

1. The robot will be laid out into separate modules.
2. One person will start the stop watch.
3. The motor modules will then be attached to the locomotion module.
4. The actuation and sensor modules will then be attached to the locomotion base.
5. Once the connections (mechanical and electrical) are made the timer can be stopped and time recorded.
6. Success conditions: <2 minute assembly time (same time requirement as the fall because the system has more modules)

Results

For the Spring Validation Experiment, the team was not able to achieve assembly/boot-up time of less than two minutes. For the encore, the team was able to achieve assembly/boot-up time of slightly over two minutes with video of a successful run of 1:20.

Part B – Test dead reckoning accuracy

1. The perception module will then be removed from the robot as it is not needed for dead reckoning.
2. The robot will travel straight for 1 meter then perform a 180 degree turn.
3. The motor modules will be removed and placed in the two slots 90 degrees offset from their previous location in the locomotion module.
4. Then the robot will again travel straight for 1 meter then perform a 180 degree turn.
5. For both tests <10 centimeter and <10 degrees discrepancy between desired and actual distances and poses, respectively, will determine success.

Results

For the Spring Validation Experiment, the team just met the position specification by traveling approximately 90 centimeters. The team failed the orientation specification with a pose of approximately 15 degrees in the first test but met this specification with approximately 10 degrees in the second test. For the encore, we passed this test in the first configuration of actuation modules but the second configuration of actuation modules provided a result that was slightly off.

Part C – Test remote perception trajectory planner

1. Place the perception module on a desk in the basement.
2. Turn on perception module.
3. Observe the perception module’s visual field.
4. Place the robot (locomotion, and motor modules) and goal object on the ground in the perception module’s visual field.
5. Power the robot and make sure it is communicating with the perception module.
6. Start the perception module trajectory planner.
7. Success condition: Robot position accuracy of <10 centimeter after goal declared accomplished by the perception module.

Results

For the Spring Validation Experiment, we did not have a successful test of the remote perception trajectory planner due to a combination of odometry problems with latency. For the Encore, the team corrected these issues to successfully achieve <10 cm error on the first test with worse results on the second and succeeding tries.

7.4 Strength and Weakness of the OddBot

Strong points: General information architecture to accommodate many modules, easy to connect connector, aesthetically pleasing, long up-time

From the spring to the fall our team completely redesigned and built a new robot from the ground up along with a new information architecture to allow for communication across the platform. A locomotion base, two new actuation modules, and a sensor module were all designed, built, and demonstrated to all have their required functionality allowing for the successful completion of the majority of the tests set out for the SVE in the encore. These modules were all relatively easy to slide into the locomotion base.

Furthermore the information architecture is general so many types of additional sensing and actuation in different configurations are supported by the platform. Different configurations of the same actuation modules in the locomotion base were shown in the SVE to demonstrate the modularity of the information architecture of the system. In addition the new, sleeker OddBot is much more aesthetically pleasing than the previous boxy design and has a much lower center of gravity making it virtually impossible for the robot to tip over. The system has a long functional uptime of over six hours.

Weak points: Connection reliability, multiple configurations are not similarly capable, remote perception system not accurate enough, command delays

We were able to successfully demonstrate a modular robotics system but experienced reliability issues with our connection method. One of the primary weak points for the fall was that it was fairly difficult to make the modular connections (but these connections were reliable). For the spring we made a connection method using magnets so it only took seconds to make the connection. However this came at the expense of reliability of the connection leading to the failure of the assembly/boot-up part of the SVE test. For the encore the team was able to refine the design of the connection method so that we just missed the two minute assembly time but the reliability was still unsatisfactory especially with the actuation modules. The sensor module connection was consistently fairly reliable but the actuation modules have motors that contact the ground which prove to cause the connection to drop out frequently.

For the encore the team demonstrated dead reckoning of the robot platform in two configurations of actuation modules in the locomotion base but the second configuration was considerably less accurate than the first primarily due to the slight misplacement of the caster wheels.

When testing the remote perception trajectory planner we were able to achieve robot positional accuracy of <10 cm from the goal state but not reliably. One of the issues was that the robot would go into a state where the goal was directly in front of the robot and would go into a loop of moving slightly to the left and slightly to right without moving forward.

The information architecture is general by design which leads to longer time delays than a hard coded solutions. In addition the complexity of the modularity increases the possible error points of the system.

Refinement: Reliability of connections, multiple locomotion bases, caster wheels, communication protocol, reliability of remote perception system, hard shutoff of modules

Future work would be to improve the reliability of connection method possibly by using dowel pins, quarter turn fasteners, or some other more mechanically stable method. More modules of different types (both actuation and sensor) could be created to further increase the functionality and demonstrate more modularity of the system. These new modules would need to be built and tested to confirm they work properly with the information architecture. Furthermore different locomotion bases of different sizes and capabilities (such as more or less module slots) could be designed and built that would use the current information architecture.

The placement of the caster wheels could be refined to improve the dead reckoning of the robot in different configurations of actuation modules. The remote perception trajectory planner code could be refined to remove the case of the robot getting stuck in an infinite loop. We could investigate different types of connection methods other than the Ethernet or using software methods to better confirm that communication has been established across the platforms. We could explore using batteries or a large capacitor on the modules to avoid turning the microcomputers off and on abruptly.

8. Project Management

8.1 Scheduling Evaluation

Honest and open dialogue about progress is necessary - we met every week and made sure we were on track. This worked really well for us. Work was split into either two or four parts, and this meeting gave us a chance to formally talk about progress in a way we didn't do while working together. The times we slacked on this we noticeably fell behind, so these meetings were very useful. Spend more time helping team members develop new skills - one thing we worked on in the beginning of the semesters were trying to make sure team members got to develop new skills. As workload increased, this became less possible and we focused on our strengths. More work to ensure team members could grow would be optimal, but would also require more time.

Working in pairs was good - We tried to split work between two sub-teams as much as possible. This allowed us to focus on separate areas while still working with another person. This still required that we apportion individual tasks to each member of the sub-team, but allowed more flexibility and ensured that there was always someone to go to the machine shop or check over code.

Check constantly to remap out the plan for the semester - One problem we had was making sure the build was done by deadlines. We did not have the ability to build earlier enough the first semester, and took too long perfecting the build second semester. This caused integration issues. One option is being realistic about the time it will take to build, and then apportioning some time to build a robust simulator (which is not likely within the scope of these projects, but could be useful). Another would be to simplify the build and the project's physical components by a lot, so more time can be spent on software. We did this out of necessity first semester, however, and wound up with an underperforming robot. Sometimes there are no good answers.

8.2 Budget

8.2.1 Final Parts List

Table 5 is a refined parts list. For the full parts list and cost see Appendix A

Table 5: Condensed Parts List for the OddBot

Generalized Item	Cost
Connectors and Various Electronics	$1,393.37
Microcomputers	$1,001.06
Robot Frame	$789.08
Battery	$406.99
Actuation and Sensing	$326.88
Total	$3,917.38

8.2.2 Budgeting Process Evaluation

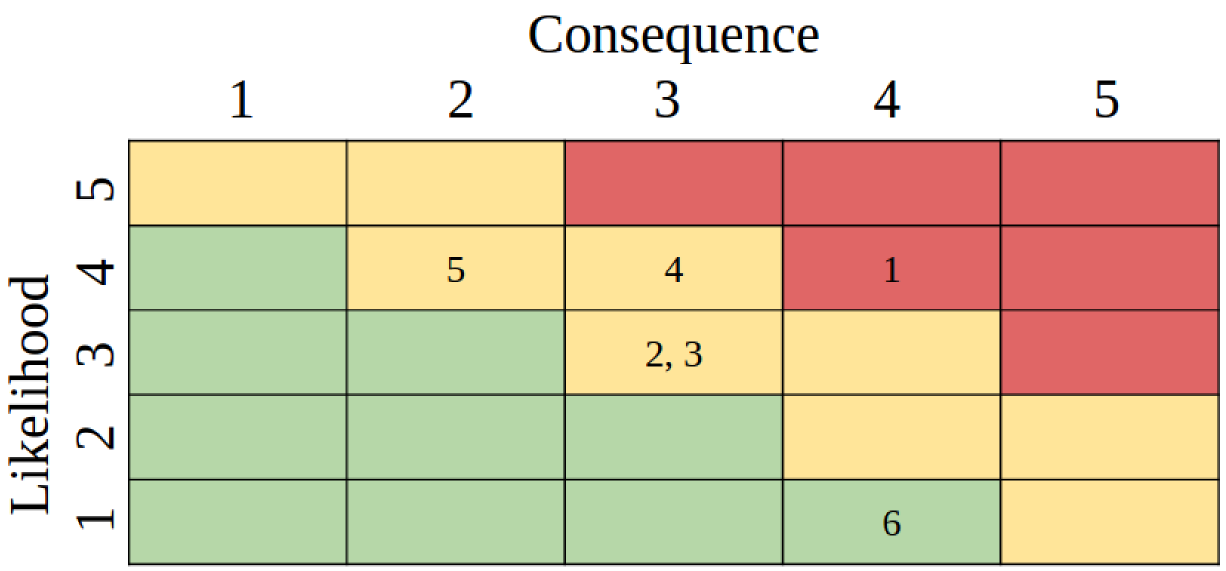
The overall budgeting process varied from the fall to spring semester. In the fall the team was not sure of the direction of the project and one of our biggest budgeting failures was buying many parts to test them out when we could have scavenged parts from the lab or further refined the scope of the project before ordering parts. In the spring the team took a more directed approach where we made absolutely certain that we needed a part before we ordered it and so we were able to finish the design of the new robot while staying under budget.

8.3 Risk Management Process Evaluation

The project risks identified in the system development review are shown in Table 6 and Figure 27.

Table 6: Table of Risks

#	Risk	Reqt.	Type	Likelihood (1 - 5)	Consequence (1 - 5)	Reduction Plan
1	Fabrication Time	All reqts.	Schedule	4	4	Weekly meetings, limit scope
2	Computer Vision Integration	All reqts.	Technical	3	3	Work with CV Professor and TA's
3	Connector robustness	MN1, MN3	Technical	3	3	Use screws, more magnets, or other mounting hardware to secure connection
4	Motor strength	MF7, MF11	Technical	4	3	Get a new motor driver board or new motor
5	Money	All reqts.	Schedule	4	2	Beg, Borrow, Steal, Limit Scope
6	ROS Networking Issues	All reqts.	Technical	2	4	Simplify network architecture, shell scripts



Likelihood \ Consequence	1	2	3	4	5
5	🟨	🟨	🟥	🟥	🟥
4	🟩	🟨 5	🟨 4	🟥 1	🟥
3	🟩	🟩	🟨 2, 3	🟨	🟥
2	🟩	🟩	🟩	🟨	🟨
1	🟩	🟩	🟩	🟩 6	🟨

Figure 27: Risk Matrix

The highest likelihood and consequence risk identified was the fabrication time of the robot. The entire system built in the previous semester was scrapped so the team had to start fresh. While this allowed a level of freedom in designing in lessons learned, it created a design and fabrication time crunch. The team successfully mitigated this risk by limiting the scope and complexity of the robotic system and regularly staying on top of deadlines. By splitting the team into two separate groups one which focused on robot design and fabrication and one which focused on robot control and computer vision, the team was able to program significant portions of the system before it was completely constructed.

The computer vision aspect of the system was also noted as a risk as no one on the team has expertise in computer vision and it was uncertain how much could be accomplished in time for the SVE. By getting as much advice as possible from TAs and professors we were able to create a robot template matching system that could create a trajectory for the robot to follow to the goal position.

Connector robustness was a risk that had much more impact on the reliability of the system than the team expected. Connector designs were tested iteratively and performed well, but it wasn’t clear how reliable connections between the locomotion base and individual modules would be until the locomotion base had been built. Once the locomotion base was built reliability issues with the connection began occurring. By slightly modifying the design of the module ports we created a very tight fit for individual modules which prevented some reliability issues. However guaranteeing an electrical and network connection between modules with a consistent boot time under 2 minutes was still an issue.

Motor strength was cited as a risk as poorly sized motors caused significant issues in the previous semester. This semester we used larger and more powerful motors which performed consistently.

Due to the team needing to completely rebuild the robot, there were concerns about running out of money. At the time of the System Development Review over 70% of the budget had already been spent. Fortunately we were able to use components that were already in the MRSD inventory such as motors and wheels. The team also simplified the design to avoid using any high-cost items and reuse other high-cost items acquired during the first semester such as the battery. Furthermore we took a refined approach to make sure to order only parts that we were fairly certain would go into the final build of the robot.

The primary risks of the project were accurately captured although the likelihood and consequence of certain risks were not accurately assessed. The reliability issues introduced by adding swappable modules caused continuous integration headaches throughout the project. Failure points ranged from the mechanical and electrical connection of the modules to networking problems and Odroid/BeagleBone Black failures. Due to these problems the system was never as reliable as desired however it was able to perform almost all specified requirements for the SVE Encore.

9. Conclusions

9.1 Lessons Learned

For this project we worked hard on creating milestones, delivering on time, and building a quality physical robot and quality information architecture. The robot ultimately still suffered reliability issues that will require a rework of the connector and the information architecture. What could we have done better?

Our first lesson was to build the robot faster. We had a schedule and kept to it, but building it faster would have allowed us to see some of the problems that were down the road. Part of it was out of our hands - the machine shop course only finished a few weeks before we had to have the system completed. Part of it was just poor planning - we could have found a way to approximate the loads on the mounts and motors and put it on the floor to better stress test the system.

Our second lesson was to buy more of what we needed. As many of the teams in this program have seen you can never buy enough. At one point in the project we needed ten power-pull connectors and therefore bought ten. As soon as the connectors arrived, we found that we actually needed twelve. This played out over several items, and cost time that we did not have to spare. We should have bought extra the first time, and saved ourselves hassle and precious time.

Our third lesson was to build the robot better. We tried to cut cost by buying items we thought were only slightly better than what we would need. This was rational as we wanted to conserve our budget, but ultimately came back to bite us. We should have had a worst case scenario, and then estimated our need at 150% of that. Then our motors and mounts would have been able to withstand the weight of the full robot, and could overcome the resistance of the floors and poor casters.

Our fourth lesson directly contradicted the third. Once we desired to rebuild the robot to be better and connect faster, we over-engineered without constant testing, resulting in a very robust base and module system that still suffered connection issues.

Our fifth lesson was to better define use cases and requirements specific to a task. Building a modular robot leaves a lot of possibility to what could theoretically be accomplished. Narrowing the scope of the project was difficult and took a lot of time. Designing with a specific goal would have made performance metrics clearer to instructors and ourselves and would allow us to focus better on the design process.

9.2 Future Work

We will apply these lessons to our future endeavors. If we were to continue this project, we would iterate on the connector design again to make something more robust. We would also do more to ensure reliability in software of the modules, so if they dropped connection due to a hardware issue they could come back online. Generally, we will make an effort to build correctly the first time but to also make sure to do more iterative testing for hardware and software.

10. References

[1] H. Choset, "16311 Introduction to Robotics," [Online]. Available: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/current/labs/lab07/index.html.

[2] Worcester Polytechnic Institute, "Robotics Engineering," 2014. [Online]. Available: https://www.wpi.edu/academics/catalogs/ugrad/rbecourses.html.

[3] "Udoo Quad," [Online]. Available: http://shop.udoo.org/usa/product/udoo-quad.html. [Accessed 2014].

[4] "Raspberry Pi," [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-2-model-b/.

[5] "BeagleBone Black Rev C - 4GB Flash - Pre-installed Debian," [Online]. Available: http://www.adafruit.com/product/1876. [Accessed 2014].

[6] "Odroid-C1," [Online]. Available: http://ameridroid.com/products/odroid-c1.

[7] A. N. Knaian, "Electropermanant Magnetic Connectors and Actuators," MIT, 2010.

[8] "Kinematic Constraints and Motion Primitives," [Online]. Available: source: http://sbpl.net/node/48.

11. Appendix

11.1 Appendix A: Final Parts List

Table 7: Final Parts List for the OddBot

Order Number	Part Name	Cost
1	BeagleBone Black Rev C	$64.47
2	300Mbps Wifi Wireless-N USB Micro Mini Adapter with High Gain Antenna	$15.99
3	Soundsoul 24k Gold Connector Banana Plugs, Open Screw Type	$16.99
4	MSI Computer Corp. Mini ITX DDR3 1600 LGA 1150 Motherboards H81I	$69.99
5	Crucial RAM	$39.74
6	SAMSUNG 840 EVO MZ-7TE120BW 2.5" 120GB SATA III TLC Internal Solid State Drive (SSD)	$85.85
7	Intel Core i3-4130T Haswell Dual-Core 2.9GHz LGA 1150 35W Desktop Processor Intel HD Graphics 4400 BX80646I34130T	$127.49
8	D-Link 5-Port EasySmart Gigabit Ethernet Switch - Lifetime Warranty (DGS-1100-05)	$48.93
9	Mini-Box picoPSU-160-XT High Power 24 Pin Mini-ITX Power Supply	$57.00
10	M350 Universal Mini-ITX enclosure by Mini-Box Black	$39.44
11	Cable Matters USB 2.0 to 10/100 Fast Ethernet Adapter in Black	$10.99
12	Devantech RD02 - 12 Volt Robot Drive System	$217.00
13	HerkuleX DRS-0101 Smart Robot Servo	$79.98
14	UDOO Quad	$146.00
15	UDOO Quad	$146.00
16	TP-LINK TL-WR841N Wireless N300 Home Router, 300Mpbs, IP QoS, WPS Button	$19.99
17	24V 10AH, STARKPOWER 'ULTRAENERGY" LITHIUM ION BATTERY (LIFEPO) ENERGY STORAGE BATTERY (Shipping only, got this part refunded)	$10.34
18	24V LIFEPO CHARGER 5A	$59
19	Logic Level Converter Bi-Directional	$18.96
20	Samsung Electronics 16GB PRO Micro SDHC with Adapter Upto 90MB/s Class 10 Memory Card (MB-MG16DA/AM)	$53.37
21	Universal Plate for HerkuleX Servo 20pk	$4.00
22	U2 Bracket for HerkuleX Servo 4pk	$3.00
23	Aluminum Metric T-Slotted Framing System, Single, 20mm Face Width, Solid, 10' Long	$132.20
24	Aluminum Metric T-Slotted Framing System Corner Connector, 3-Way, for 20 mm Extrusion	$152.80
25	High-Load Compression Spring .300" Length, .60" OD, .009" Wire Thickness, Packs of 1	$4.84
26	Metric Pan Head Phillips Machine Screw Zinc-Plated STL, M3 Size, 25mm Length, .5mm Pitch, Packs of 100	$3.28
27	Heatsink TO-220	$12.69
28	Optically Colored Cast Acrylic Sheet, 1/8" Thick, 24" x 24", Gray	$26.39
29	Powerizer LiFePO4 Battery: 24V 10Ah (240Wh, 40A rate) --- Replace SLA 24V 10Ah with 4 times cycle life, lighter weight -- UN 38.3 Passed (DGR)	$347.99
30	SMAKN® 0.28" Mini 2 Wire Dc Voltmeter 3.0~30v Blue Digital Display Voltmeter Gauge LED Voltage Detector	$6.99
31	Aluminum Metric T-Slotted Framing System, 90 Degree Bracket, Single, 2-Hole, for 20mm Extrusion	$62.08
32	Zinc-Plated Steel End-Feed Fastener, for 20mm Aluminum Metric T-Slotted Framing System, packs of 4	$20.40

33	Type 316 Stainless Steel Flat-Head Socket Cap Screw, M5 Size, 12mm Length, .80mm Pitch, packs of 50	$9.63
34	Black-Oxide Class 12.9 Socket Head Cap Screw, Alloy Steel, M2 Thread, 8mm Length, 0.4mm Pitch, packs of 100	$12.68
35	Black-Oxide Class 12.9 Socket Head Cap Screw, Alloy Steel, M2 Thread, 12mm Length, 0.4mm Pitch, packs of 100	$11.12
36	Class 04 Steel Thin Hex Nut - DIN 439B, Zinc Plated, M2x0.4 Thread Size, 4mm Wide, 1.2mm High, packs of 100	$3.64
37	Build-Your-Own Push-in Connector, Kit for 14-10 AWG, 45 Amps, Black, packs of 5	$5.22
38	Build-Your-Own Push-in Connector, Kit for 14-10 AWG, 45 Amps, Blue, packs of 5	$5.22
39	Build-Your-Own Push-in Connector, Kit for 14-10 AWG, 45 Amps, Red, packs of 5	$5.22
40	Optically Colored Cast Acrylic Sheet, 1/4" Thick, 12" x 12", Black	$37.68
41	M3-ATX-HV 95 Watt - Smart Automotive Carputer Power Supply	$64.50
42	Arctic Silver 5 Thermal Compound 3.5 Grams	$7.89
43	Heatsink and Clip for TO-247 BLK	$12.13
44	Scythe Mini KAZE ULTRA 40mm Silent Mini Fan (SY124020L)	$15.50
45	HOSSEN® DC/DC Converter Regulator 24V Step Down to 12V 20A 240W	$19.97
46	CONN RECPT 13POS JAM NUT W/PINS	$24.93
47	CONN PLUG 13POS STRAIGHT W/SCKT	$37.55
48	VNH5019 Motor Driver Carrier	$28.90
49	Raspberry PI Model A+ 256MB RAM	$34.05
50	CRC Dry Graphite Lube, 10 oz Aerosol Can, Black	$7.08
51	ODROID-C1	$42.90
52	Metric Type 316 Stainless Steel Dowel Pin	$8.13
53	ODROID-C1	$42.90
54	DC Plug and Cable Assembly 2.5mm	$1.45
55	BeagleBone Black Rev C	$59.88
56	VNH5019 Motor Driver Carrier	$29.90
57	Mill-Max Connector	$30.56
58	Mill-Max Connector	$31.15
59	Neodymium Magnets 3/8 in x 1/8 in Disc N42	$32.65
60	Optically Clear Cast Acrylic Sheet	$7.25
61	KEEDOX DC/D Converter 12V Step Down to 5V 3A Power Supply Module	$15.40
62	Various McMaster parts	$100.61
63	Metric 316 Stainless Steel Flat Head Phillips Machine Screw	$8.33
64	Samsung Electronics 16GB PRO Micro SDHC with Adapter Upto 90MB/s Class 10 Memory Card (MB-MG16DA/AM)	$19.99
65	White Delrin Acetal Resin Sheet	$8.98
66	24" x 48" - Clear Acrylic Plexiglass Sheet - 1/4" Thick Cast	$52.99
67	Compact End-Feed Fastener, M% Thread Size for Aluminum T-Slotted Framing Extrusion	$31.84
68	Mill-Max Connector	$37.27
69	Mill-Max Connector	$83.77
70	KEEDOX DC/D Converter 12V Step Down to 5V 3A Power Supply Module	$15.40
71	Various McMaster parts	$33.89
72	Black Delrin Acetal Resin Sheet, 1/4" Thick, 12" x 12"	$29.11
73	Various McMaster parts	$69.74
74	Interpower 8301213 IEC 60320 C14 Screw Mount Power Inlet with Quick Disconnects, IEC 60320 C14 Socket Type, Black, 10A/15A Rating, 250VAC Rating	$2.61
75	Various McMaster parts	$79.20

76	Neodymium Magnets 3/8 in x 1/8 in Disc N42	$34.93
77	HOSSEN® DC/DC Converter Regulator 24V Step Down to 12V 20A 240W	$19.95
78	1/4" x 24" x 48" - Black Plexiglass Acrylic Sheet - #2025	$57.99
79	Blade-Style Low-Voltage Fuse Standard, 32V AC/DC, 10 Amps, Packs of 5	$3.28
80	Black Delrin ® Acetal Resin Sheet 1/4" Thick, 12" X 12"	$29.11
81	Black Alloy Steel Flat-Head Socket Cap Screw Class 10.9, M5 Size, 12mm Length, .80mm Pitch, Packs of 100	$7.59
82	Black Alloy Steel Flat-Head Socket Cap Screw Class 10.9, M5 Size, 10mm Length, .80mm Pitch, Packs of 100	$7.59
83	Black Alloy Steel Flat-Head Socket Cap Screw Class 10.9, M5 Size, 8mm Length, .80mm Pitch, Packs of 100	$6.57
84	Black Alloy Steel Flat-Head Socket Cap Screw Class 10.9, M5 Size, 6 mm Long, 0.8 mm Pitch, Packs of 25	$8.00
85	Cable Matters USB 2.0 to 10/100 Fast Ethernet Adapter in Black	$10.99
86	ODROID-C1	$43.90
87	SUPERNIGHT DC 12V 24V to 5V 10A 50W Converter Step Down Regulator for Car Low Voltage Transformer	$10.99
88	Addressable RGB 120-LED Strip, 5V, 2m	$54.85
89	Stereo 2.8W Class D Audio Amplifier - I2C Control AGC - TPA2016	$20.53
90	Speaker - 3" Diameter - 8 Ohm 1 Watt	$3.90
91	Metric Flat Head Phillips Machine Screw Zinc-Plated STL, M3 Size, 20mm Length, .5mm Pitch, Packs of 100	$3.60
92	Class 04 Steel Thin Hex Nut - DIN 439B Zinc Plated, M3X0.5 Thread Sz, 5.5mm Wd, 1.8mm Ht, Packs of 100	$3.10
93	Raspberry Pi 2 - Model B - ARMv7 with 1G RAM	$48.86
94	DC Plug and Cable Assembly 2.5mm	$7.40
95	KEEDOX DC/D Converter 12V Step Down to 5V 3A Power Supply Module	$7.69
96	.003 in x 8.5 in x 11 in matte two sides - 10 Sheet Pack Light Diffuser Films (at the bottom of the web page)	$23.77
97	KEEDOX DC/D Converter 12V Step Down to 5V 3A Power Supply Module	$7.69
98	DC Barrel Jack Plug - Male	$17.24
99	Metric Flat Head Phillips Machine Screw Zinc-Plated STL, M2 Size, 8mm Length, .4mm Pitch, Packs of 100	$3.17
100	Metric Aluminum Female Threaded Hex Standoff 4.5mm Hex, 20mm Length, M3 Screw Size	$6.60
101	Steel Hex Nut Low-Strength, M2X0.4 Thread Size, 4mm Wd, 1.6mm Ht, Packs of 100	$1.04
102	Metric Flat Head Phillips Machine Screw Zinc-Plated Steel, M4 Size, 16mm Length, .7mm Pitch	$3.60
103	Type 316 Stainless Steel Hex Nut, M3x0.5 Thread Size, 5.5mm Wide, 2.4mm High, packs of 50	$3.08
104	Type 316 Stainless Steel Hex Nut, M5x0.8 Thread Size, 8mm Wide, 4mm High, packs of 50	$4.12
105	Optically Clear Cast Acrylic Sheet, 7/16" Thick, 12" x 24"	$47.45
106	DC Barrel Jack Adapter - Female	$16.84
107	MB1013 HRLV-MaxSonar-EZ1	$64.73
108	Metric Flat Head Phillips Machine Screw Zinc-Plated STL, M3 Size, 12mm Length, .5mm Pitch, Packs of 100	$3.20
109	Steel Hex Nut Low-Strength, M3X0.5 Thread Sz, 5.5mm Wd, 2.4mm Ht, Packs of 100	$1.04
	Total	$3,917.38