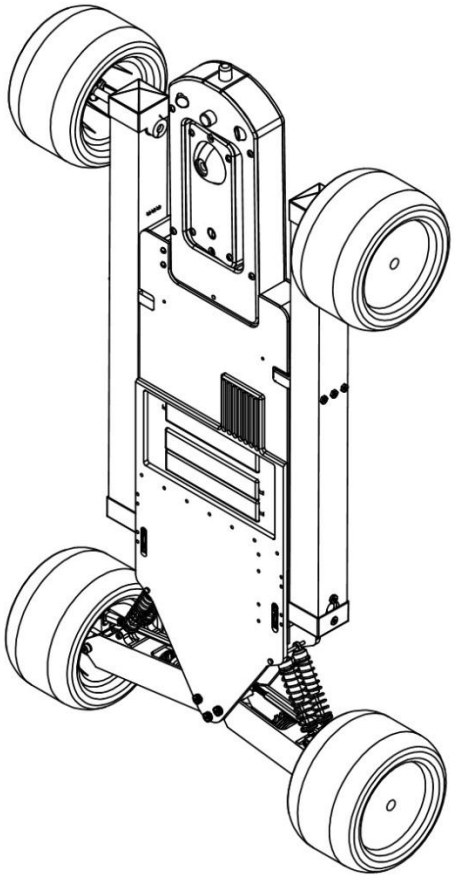
# ISUS
# An Autonomous Off-Road Balancing Robot

## High Level Software and Components Implementation



by

Kent Williams

Computer Engineering

California Polytechnic State University - San Luis Obispo

June 2012

# Table of Contents

# Table of Figures



## Appendix C

# Chapter 1: Introduction

## Problem Summary

The current day off-road and multi-terrain robotic vehicles for use in the military settings are almost entirely continuous track driven given their proven record of being the most aggressive available mobile robot locomotion system. The downside of a continuous tracked locomotion system is its lack of efficiency, its fast rate of wear, and the noise it produces against hard surfaces. When a mobile robotic vehicle is deployed in the current day battle setting, it often travels the majority of is distance on paved or smooth dirt and rarely encounters terrain worthy of its traction. The most beneficial aspect of the continuous track drive and the reason it has been chosen as the preferred drive system, is its ability to turn on the spot without the need for forward or backward locomotion. Currently there are no worthy competitive locomotion system that can provide the off-road capabilities of a continuous track system with turn on the spot capabilities. This project aimed to design and construct a vehicle that harnessed the off-road capabilities as well as the maneuverability of a continuous track robot.

## Client Overview

The ISUS Mobile Robot (MR) was design with a specific capabilities set for rough terrain reconnaissance and investigation in urban environments. Its unique ability to travel efficiently on rough terrain while remaining agile makes it an appropriate choice for disaster recovery and bomb disposal missions. Potential clients include military branches containing bomb disposal squads, public service protection and emergency agencies.

# Chapter 2: Product Definition

## Need statement

There exist a need for a more efficient off-road mobile robotic platform that has the versatility as well as the aggressive off-road capabilities of a continuous track drive system.

## Objectives

The ISUS MR was designed with maximum locomotion efficiency and versatility in mind. With a suspension that can handle off-road terrain, the ISUS MR is able to travel quickly on only two wheels allowing it to maximize its efficiency until encountering challenging rubble at which time it can fall onto all four wheels for maximum stability and aggressive locomotion.

## Requirements

The ISUS MR can be broken down into three key component groups that make this mobile robot a sound machine.

### Mechanical Structure

The ISUS MR has a very unique mechanical structure given its nonstandard method of locomotion. Its default mode of driving is upright given the increase in power efficiency, but the robot contains two sets of rough terrain wheels on which the robot starts from at initialization. The forward most set of wheels are bound to a set of moveable arms which allow the robot to articulate itself into the upright position, but also drive on four wheels for the most challenging terrain. The mechanical structure of the ISUS MR is almost entirely machined aluminum, making it a light but strong overall material for the vehicle. The shape of the vehicle is slender in order to limit the amount of mass about the balancing drivetrain axis and to lessen the impact supported by the wheel axles when falling onto the second set of wheels.

### Sensor Suite

The ISUS MR encompasses the necessary sensor suite to enable it to navigate an unknown environment and proceed through it while not impacting obstacles. For location awareness the ISUS MR utilizes a GPS receiver, optical encoders for the drive motors, and accelerometers which together allow the robot to determine its relative position for accurate navigation. For obstacle avoidance and feature detection the ISUS MR utilizes four sonar range sensors and a high resolution USB camera. The fusion of these sensors makes the ISUS MR a capable autonomous robot for critical missions.

### Capable Computing Power

The ISUS MR contains two main computing units, one responsible for low level control over the motors and balancing algorithm needed to keep the vehicle upright, and the second for high level autonomy and communication with the user. Both the computers communicate between each other over serial communication, allowing constant updates between each of the critical systems.

### Constraints

The ISUS MR is an extremely capable and versatile machine, but it is still prone to untested scenarios and contains some inherent limitations. Although the advanced suspension and balancing algorithms have been well designed, they will potentially fail after an excessive side rolling thrust.  Also, the main power source is delivered by battery cells which limit the amount of time the vehicle can operate, but it still remains an appreciable amount of time.

## Criteria

This particular mobile robotic vehicle falls under a niche class of sophisticated machines with a particular goal in mind. The goal is to reach a destination in a quick and efficient manner with little to no human assistance, and be able to report back information or the status of a missions focus. The vehicles must be extremely robust and be able to handle rough to extreme terrain and remain a reliable resource for those using it. The unit itself must also be compact and light enough so that an average size person can lift it and position it if needed.

# Chapter 3: Design

## Team Process

The ISUS MR was a joint project between Tyson Messori and Kent Williams. Tyson Messori has made the vehicles' mechanical structure as well as its off-road balancing capabilities the focus of his Master's Thesis in Mechanical Engineering at Cal Poly. Kent Williams has designed and implemented the computing internals as well as the sensor suite needed to enable the robot with autonomous capabilities and a substantial communication link between the vehicle and the operator for his Computer Engineering Senior Project at Cal Poly. Together, the efforts for the design and construction of this vehicle have been shared and individually pursued.

## System Architecture

### High Level Computing

The high level computing system encompasses an embedded Linux computer and the services it provides as well as the top level communication with the lower level computing device. It is within this system that the overall decisions for locomotion and navigation are decided upon for the vehicle.

### Low Level Microcontroller

The low level microcontroller manages the drive motors, actuators, monitoring and range sensors, and connection with the onboard inertial measurement unit for use in balancing and localization.

### Power Management

The vehicle has a single power source consisting of eight LiFePO4 cell batteries. This power is managed and distributed via two power switches, one for low, and one for high power.

### Sensors

The vehicles' sensors are interfaced directly to either the high level computing device or the low level microcontroller. Some of the sensors data is shared between the high level computer and the low level microcontroller.

### Communications

The vehicle's primary communication means with the operator is over WI-FI, which the high level computer provides access to using an ad hoc network.

## Hardware Block Diagram

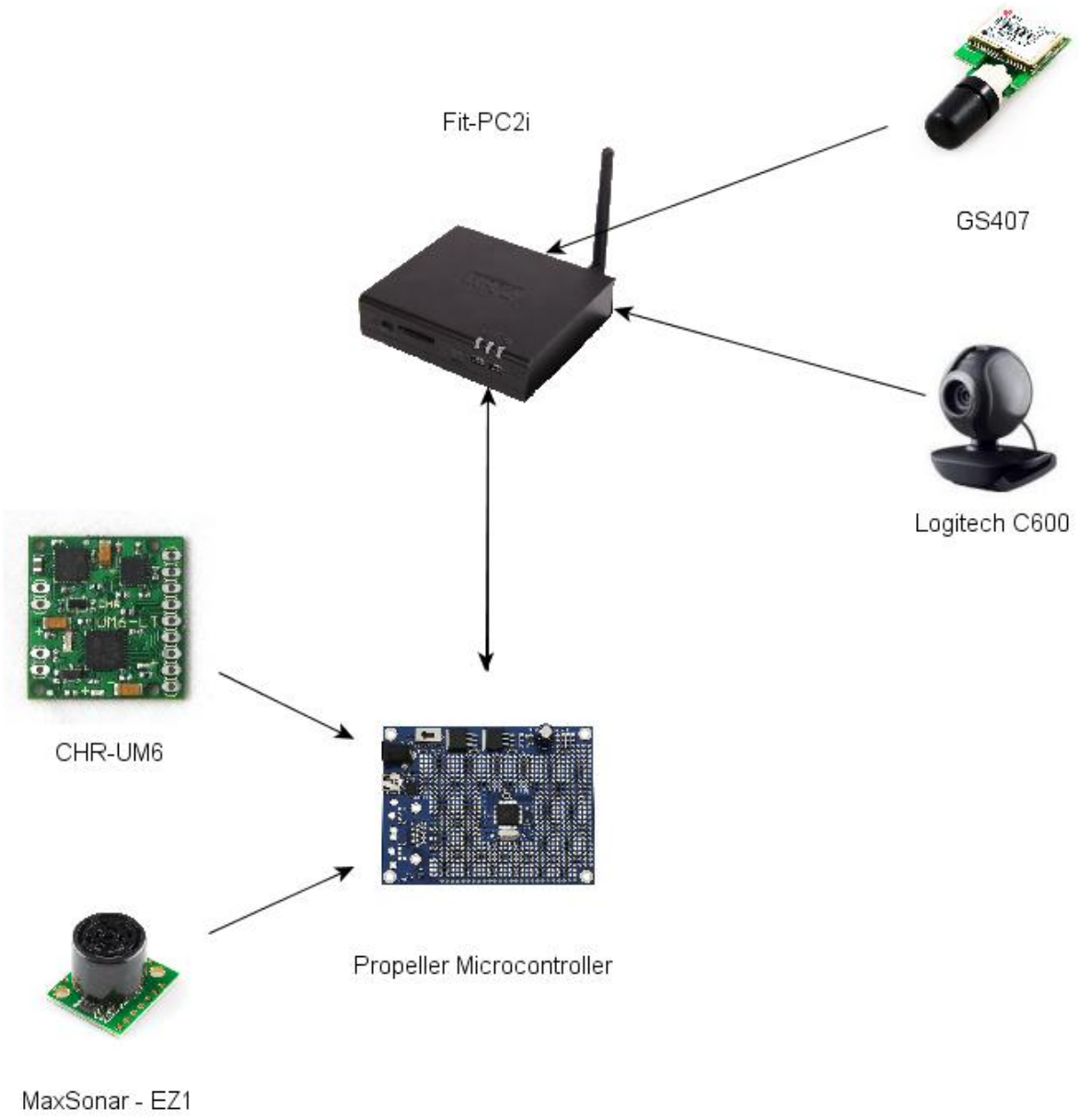



**Figure 1.** Computer Systems and Major Sensors

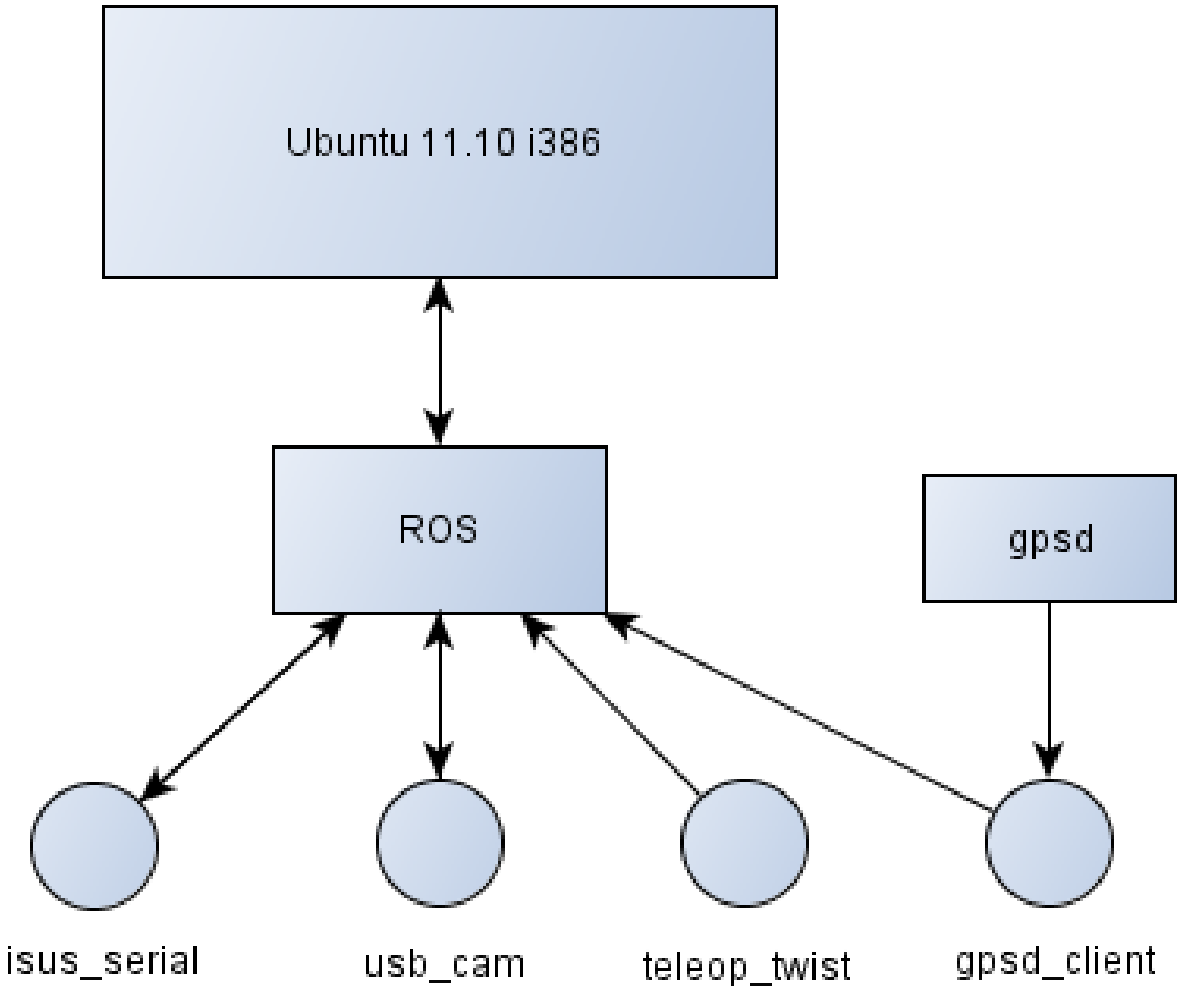



**Figure 2.** Overall High Level Software Diagram

## Software Algorithms

The software architecture of the ISUS MR was divided into two major systems. The low level computing system which is responsible for the balancing algorithm, handling of odometric calculations, and locomotion control and the high level system which was responsible for localization, navigation, and obstacle avoidance.
The following will explain the workings of the support programs that were needed to implement a dead reckoning localization algorithm with computer vision obstacle avoidance. These systems would allow the vehicle to successfully navigate to a GPS waypoint coordinates if supplied.

## High Level Software Components

The following software components are run on the CompuLab Fit-PC2i computer with the following specifications; 2GHz Intel Atom Z550, 2GB DDR2 SDRAM, 64GB Solid State SATA hard drive. This computer is interfaced with the high resolution USB camera and the GS407 GPS over TTL with conversion to USB.

### Host Operating System

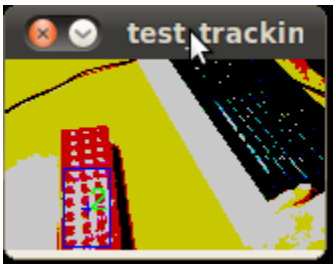
The high level computer is running Ubuntu 11.10 i386, with Linux kernel 3.0.0-12. The wireless network was configured utilizing iwconfig and wpa_supplicant.

### Robotic Operating System (ROS)

The high level computing system utilizes ROS to coordinate the sharing of sensor data, communicate and distribute data from the lower level microcontroller, and handle locomotion control via user inputs and the autonomous operations of obstacle avoidance and waypoint navigation.

### USB Camera Image Driver for ROS

In order to retrieve images from the USB camera, a USB webcam driver for ROS was used. This allowed images to be pulled from the device and converted to ROS message format for transferring over ROS topics for vision processing. Refer to Figure 4 in appendix C to examine this code.

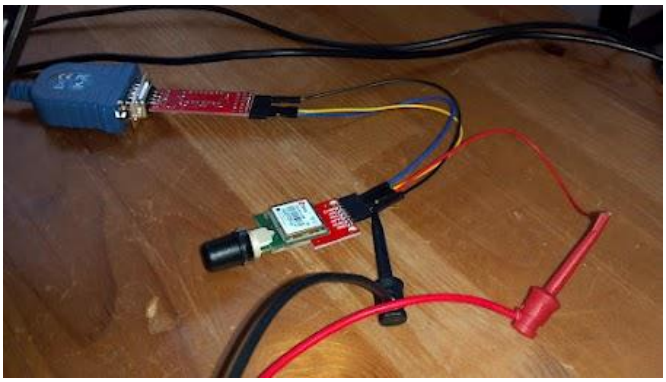



### Serial Communication with Microcontroller

In order to establish communication with our Propeller microcontroller we had to write a custom serial protocol since there was none available in the *Spin* language that the Propeller chip runs on.  On the high level computer, we wrote a custom ROS serial node to establish the serial connection and share data between the computers.

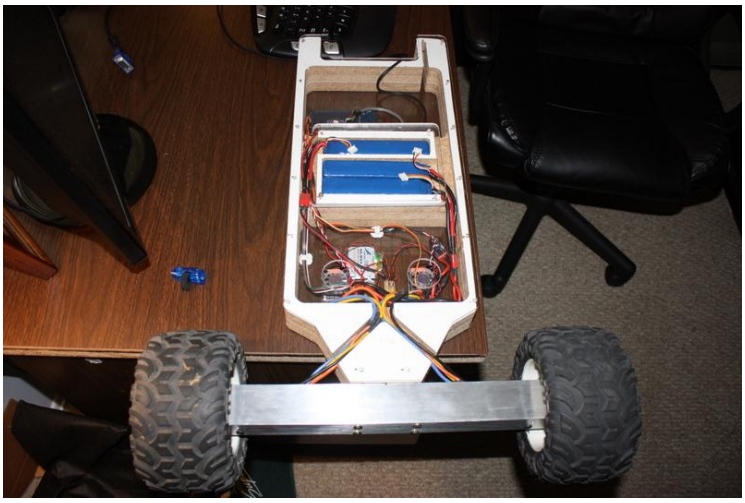
### GPS Coordinate Data Collection

To collect GPS coordinate data from our GPS antenna device, two services were used. The first is an open source GPS data collecting daemon for linux called gpsd. The second service was a ROS node, gpsd_client, which talked with gpsd and converted the coordinate data into ROS typed message format to publish over a topic.



## Mechanical Design

The mechanical structure of this vehicle was designed and constructed by Tyson Messori for completion of his Master's Thesis in Mechanical Engineering at Cal Poly. The details of the mechanical design will not be explored in this document.



## Bill of Materials

Refer to Figure *Appendix A.*

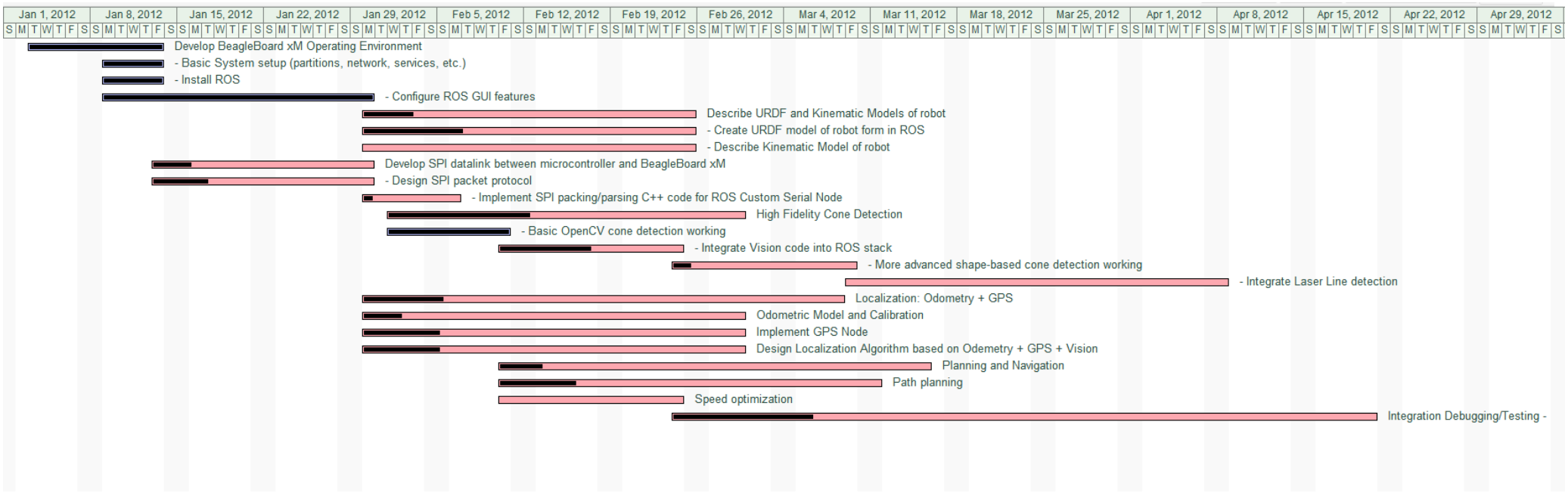
# Appendix A – Bill of Materials

| | Part Description | Part Number | Cost | Quantity | Total Cost | Part Weight (lb) | Total Weight (lb) | Vendor | Buyer | Recieved? |
|---|---|---|---|---|---|---|---|---|---|---|
| Color Key: | | | Already Purchased | | | Phase II Items | | | | |
| Catagory | Part Name | Description | Cost ea. | Quantity | Total Cost | Total Weight (lb) | | Vendor | | |
| Chassis Materials | Aluminum for chassis | .25"X24"X14" | $0 | 1 | $0 | 7 | 7 | | Tyson | Yes |
| | High Density Polyethylene | 1.5"X24"X24" | $103 | 1 | $0 | | 0 | Mr. Plastic | Tyson | Yes |
| | Aluminum Square Tube 6061 T6 | 2"X2" X 72" | $30 | 1 | $0 | | 0 | | Tyson | Yes |
| | Polycarbonate | .25"X24"X14" | $50 | 1 | $0 | | 0 | | Tyson | Yes |
| Machining Work | Shock Brackets, Pivot Blocks, Aluminum Plates, Aluminum A-Arms, Sensor Module | | $400 | 1 | $0 | | 0 | | Tyson | Yes |
| Main Chassis | Acetal (Delrin) Plastic for chassis | 3"X12"X24" | $400 | 1 | $400 | 5.4 | 5.4 | | | |
| | Aluminum Square Tube 6061 T6 | 2.5"X2.5" X 72" | $70 | 1 | $70 | 6 | 6 | | | |
| | Screw 6-32 2.25" LG | 90275A160 | $5 | 1 | $5 | | 0 | | | |
| | Screw 6-32 0.5" LG | 90275A148 | $2 | 1 | $2 | | 0 | | | |
| | Nut 6-32 Press Fit | 95117A433 | $8 | 2 | $16 | | 0 | | | |
| | Slotted Cam Latch | 12265A51 | $11 | 1 | $11 | | 0 | | | |
| | Rubber Seal for Battery Door | 1867T24 | $6 | 1 | $6 | | 0 | | | |
| Battery Door | Hinge For Battery Door | 1581A214 | $3 | 1 | $3 | | 0 | | | |
| | Rubber Washer | 90130A007 | $4 | 1 | $4 | | 0 | | | |
| | Tracks | Vex tracks | $0 | 1 | $0 | 0 | 0 | | Tyson | Yes |
| | Gear Boxes 1/20 ratio | http://banebots.com/pc/P60K-STOCK/P60K-44-0004 | $56 | 2 | $0 | 0 | 0 | | Tyson | Yes |
| Main Drive Motors | Low Backlash Gearmotors 150 watt | Midwest Motion: MMP S22-348H-24V GP52-007 | $439 | 2 | $878 | 3.2 | 6.4 | | | |
| | Motor Controller | Pololu 24V22A Motor Driver | $65 | 2 | $130 | 1 | 2 | | | |
| | Low Backlash Universal Joint | 2524K3 | $116 | 2 | $232 | 0.5 | 1 | | | |
| | 10" Heavy Duty ATR wheels | http://www.superdroidrobots.com/shop/item.aspx?itemid=1132 | $85 | 2 | $170 | 3.91 | 7.82 | | | |
| | Shoulder Bolt 1/4" | 93996A834 | $5 | 4 | $19 | | 0 | | | |
| | Nylon Bushing 1/4" (pack of 5) | 6389K231 | $2 | 2 | $4 | | 0 | | | |
| | Shoulder Bolt 1/8" for horn | 93996A123 | $3 | 2 | $7 | | 0 | | | |
| | 4-40 Nut for 1/8" shoulder bolt 100 pack | 90480A005 | $1 | 1 | $1 | | 0 | | | |
| Drive Suspension | Aluminum Block 2.25"X2.25"X12" | Online Metals.com | $35 | 1 | $0 | 0.5 | 0.5 | | Tyson | |
| | HS-7955TG Servo for steering | Servo City | $90 | 1 | $0 | 0.5 | 0.5 | | Tyson | |
| | 1/2" Bearings Abec 3 double shielded | A7Y55-PSS11250M | $20 | 4 | $80 | | 0 | | | |
| | 10-32 Helicoils (pack of ten) - need tool | 91732A231 | $4 | 1 | $4 | | 0 | | | |
| | Rod Ends | 6072k145 | $7 | 4 | $26 | | 0 | | | |
| | Screw 10-24 3/4" LG for Rod End | 91306A348 | $11 | 1 | $11 | | 0 | | | |
| | 1/4" Pin for Tilt Lever Arm | 98380A537 | $7 | 1 | $7 | | 0 | | | |
| | Spring Pin 5/8" LG, 3/16" Dia | 95755A315 | $10 | 1 | $10 | | 0 | | | |
| | Slotted Disk Shaft Coupling 1/2" ID | 9889T204 | $16 | 4 | $63 | | 0 | | | |
| | Acetel Disk for above shaft coupling | 59985K63 | $5 | 2 | $10 | | 0 | | | |
| | Wheels | Pro Line 6.5" Diameter | $0 | 2 | $0 | 0 | 0 | | Tyson | |
| Sensor Module | 5/8" X 6" Hollow Tube | http://servocity.com/html/hollow_aluminum_tubing.html | $3 | 1 | $0 | | 0 | Servo City | Tyson | |
| | 5/8" Calmping Hubs | http://servocity.com/html/clamping_tube_hub.html | $7 | 1 | $0 | 0.1 | 0.1 | Servo City | Tyson | |
| | 5/8" ID Nylon Bearing | 6294K227 | | | $0 | | 0 | | Tyson | |
| | 3/8" X 1" precision shaft | Servo City | | | $0 | | 0 | | Tyson | |
| | 3/8" ID bushing | Servo City AGB-16 | | | $0 | | 0 | | Tyson | |
| | 5-40 Screws for clamping hub | 91255A123 | $10 | 1 | $0 | | 0 | Servo City | Tyson | |
| | 32P 30 tooth gear | http://servocity.com/html/32_pitch_acetal_hub_gears__3_1.html | $4 | 1 | $0 | | 0 | Servo City | Tyson | |
| | 32P 60 tooth gear | http://servocity.com/html/32p_hs-805_815_gears.html | $7 | 1 | $0 | | 0 | Servo City | Tyson | |
| | Micro Servo | http://servocity.com/html/hs-85mg__mighty_micro.html | $30 | 1 | $0 | 0.1 | 0.1 | | Tyson | |
| | Plug for upper module 3/4" diameter | 9688K27 | $10 | 1 | $0 | | 0 | | Tyson | |
| | Plug for lower module 9/16" diameter | 9688K33 | $8 | 1 | $0 | | 0 | | Tyson | |
| | Screw 1.75" LG 18-8 for shaft lock | 92949A205 | $10 | 1 | $0 | | 0 | | Tyson | |
| | Screw .5" LG #8 | 90184A224 | $10 | 1 | $0 | | 0 | | Tyson | |
| | 4.5" Wheels for bumpers | http://www.superdroidrobots.com/shop/item.aspx?itemid=107 | $19 | 1 | $19 | | | | | |
| Paddle Assembly | Square Finishing Plug for Tube Ribbed, Fits 2-1/2" Tube OD, 2.232"-2.334" Tube ID 10 pack | 9565K92 | $8 | 1 | $8 | | | | | |
| | Locking Collars | 57445K73 | $6 | 4 | $24 | | | | | |
| | 4mm Spring Pin | | $10 | 1 | $10 | | | | | |

| | Part Description | Part Number | Cost | Quantity | Total Cost | Part Weight (lb) | Total Weight (lb) | Vendor | Buyer | Recieved? |
|---|---|---|---|---|---|---|---|---|---|---|
| | M4 Screw | | | | $0 | | | | | |
| Paddle Servo Mechanisms | Screw 6-32 .375 LG 100 pack | 91783A146 | $5 | 1 | $5 | | 0 | | | |
| | Fab Steel Disk | P266_Steel_Break_Disk | $100 | 1 | $0 | | | | | |
| | Fab Aluminum Lever | P267_Steel_Break_Lever | $100 | 1 | $0 | | | | | |
| | Nut 6-32 100 pack | 91841A007 | $4 | 1 | $4 | | 0 | | | |
| | Set Screw 6-32 30 pack | 90251A144 | $10 | 1 | $10 | | 0 | | | |
| | Motor with 1:353 gearbox | IG52-04 24VDC 010 RPM Gear Motor | $140 | 1 | $140 | | 0 | | | |
| | 3/8" Shoulder Bolt 1 5/8" lg shaft | 94498A560 | $6 | 2 | $13 | | 0 | | | |
| | 1/4-20 Nut for Paddle Shoulder Bolts | 97135A210 | $3 | 1 | $3 | | 0 | | | |
| | 1/4-20 T-Nut for Paddle Shaft Collar | 90594A025 | $10 | 1 | $10 | | 0 | | | |
| | 1/1-20 Hex Screw .75" lg shaft | 91205A540 | 9.95 | 1 | $10 | | 0 | | | |
| | 3/4" Shaft collar to mount paddles | 9677T1 | 7.43 | 2 | $15 | | 0 | | | |
| | 3/4" Steel Tube for paddles | | 35 | 1 | $35 | 0.55 | 0.55 | | | |
| | 1/4-20 screws for shaft collar | 93615A410 | 1 | 6.4 | $6 | | 0 | | | |
| | 3/4" Needle Bearing | 5905K135 | 9.99 | 1 | $10 | | 0 | | | |
| | 3/4" Bronz Bushing | 2868T178 | 1.3 | 2 | $3 | | 0 | | | |
| | 2.5 Module Spur Gear 14 tooth 12 mm bore PD: 35 FW: 18 Material: Steel | A1C 2MYK25014 | 28.59 | 1 | $29 | | | | | |
| | Hollow Shaft OD: 20 mm ID: 14 mm Material 52100 Steel | S40LPSM14200400 | 34.09 | 1 | $34 | | | | | |
| | 3/16" 1.25 LG Shoulder Bolt for Gears | 93996A735 | 6.29 | 2 | $13 | | 0 | | | |
| | 2.5 Module Spur Gear 25 tooth 20 mm bore PD: 62.5 mm FW: 18 mm Material: Steel | A1C22MYK25025A | 54.07 | 1 | $54 | | | | | |
| | 6-32 1"LG screw for potentiometer holder | 91251A153 | 9.83 | 1 | $10 | | 0 | | | |
| | 1/8" Spring Pin 2000 double sheer strength | 95755A239 | 12.3 | 1 | $12 | | 0 | | | |
| Power Electronics | | | | | | | 0 | | | |
| | | | | | | | 0 | | | |
| | Drive Motors Novak Brushless 21.5 turn | Comes with motor controller and encoders. http://www.teamnovak.com/products/esc/goat_3s/index.html | 199 | 2 | $0 | 0 | 0 | Banebots | Tyson | |
| | Hex Hub for Drive Wheels | http://www3.towerhobbies.com/cgi-bin/wti0001p?&I=LXTZT3&P=7 | $12 | 1 | $0 | 0 | 0 | Tower Hobbies | | |
| | Track Drive Motors, RS550 6 in-lb running torque 260 rpm, 1/64 ratio Note: we would need encoders for these | http://www.superdroidrobots.com/shop/item.asp?itemid=1000&catid=7 | $130 | 0 | $0 | | 0 | | | |
| | Syren Motor Controller for Paddle Motor | http://www.dimensionengineering.com/SyRen25.htm | $75 | 1 | $0 | | 0 | | | |
| | 11.1V Lithium 25X50X146mm | http://www.valuehobby.com/product_details.php?category_id=15&item_id=226 | $49 | 6 | $0 | 0 | 0 | | | |
| | Battery Charger | http://www.valuehobby.com/product_details.php?category_id=13&item_id=110 | $42 | 2 | $0 | | 0 | | | |
| | 5V Isolated DC DC Converter | http://www.valuehobby.com/product_details.php?category_id=37&item_id=241 | $20 | 2 | $0 | 0.1 | 0.2 | | | |
| | Headway Batteries | http://stores.headway-headquarters.com/-strse-109/lifepo4-Headway-batteries%2C-40160s/Detail.bok?category=BATTERIES | $30 | 8 | $238 | 1.05 | 8.4 | | | |
| | Buss Bars for Headway Batteries | | $1 | 8 | $9 | 0.01 | 0.08 | | | |
| | BMS for Headway Batteries | http://stores.headway-headquarters.com/-strse-108/XJI-24V-BMS-8/Detail.bok?category=BMS%2FPCM%2FPCB | $89 | 1 | $89 | | | | | |
| | Chargeer for Headway Batteries | http://stores.headway-headquarters.com/-strse-87/24v25a-lifepo4-charger%2C-lifepo4/Detail.bok?category=CHARGERS | $176 | 1 | $176 | | | | | |
| | Battery Charger for Headway Batteries | http://www.robotmarketplace.com/products/0-THP1430C.html | $229 | 1 | $0 | | 0 | | | |
| | Power Supply For Headway Battery Charger | http://www.robotmarketplace.com/products/0-AC-DC-516W.html | $119 | 1 | $0 | | 0 | | | |
| | Screws for battery buss bars M5-8 | 91287A081 | $7 | 1 | $0 | | 0 | | | |
| | Ring Terminals for Battery Ends | | | | | | 0 | | | |
| | | | | | | | | | | |
| Logic Level Electronics | GuruPlug Server Standard | http://www.globalscaletechnologies.com/p-31-guruplug-server-standard.aspx | $122 | 1 | $0 | | 0 | Global Scale Technologies | Taylor | |
| | CHR-UM6 9d0f IMU | http://www.pololu.com/catalog/product/1255 | $150 | 1 | $0 | | | | | |
| | 12-bit 8 channel A/D chip | http://search.digikey.com/scripts/DkSearch/dksus.dll?Detail&name=MCP3208-BI/P-ND | $4 | 4 | $0 | | 0 | | | |

| | Part Description | Part Number | Cost | Quantity | Total Cost | Part Weight (lb) | Total Weight (lb) | Vendor | Buyer | Recieved? |
|---|---|---|---|---|---|---|---|---|---|---|
| | E-stop button | | | | $0 | | 0 | | | |
| | Ethernet Jack | | | | $0 | | 0 | | | |
| | FTDI chip for Guru Plug | | | | $0 | | 0 | | | |
| | RC Reciever | Futaba | $0 | 1 | $0 | | 0 | | | |
| | Propeller Microcontroller | | | | $0 | | 0 | | | |
| | Fit PC | | $275 | 1 | $486 | | | | | |
| | Solid State Hard Drives | | $105 | 2 | $210 | | | | | |
| | 3 Sonar | http://www.sparkfun.com/commerce/product_info.php?products_id=9491 | $49 | 2 | $0 | | 0 | | Ryan | |
| | 1 GPS | http://www.sparkfun.com/commerce/product_info.php?products_id=9436 | $80 | 1 | $0 | 0.1 | 0.1 | | Mike | |
| | 4 spools of wire | http://www.sparkfun.com/commerce/product_info.php?products_id=8022 http://www.sparkfun.com/commerce/product_info.php?products_id=8027 http://www.sparkfun.com/commerce/product_info.php?products_id=8023 http://www.sparkfun.com/commerce/product_info.php?products_id=8026 | $10 | 1 | $0 | | 0 | | | |
| | "9DOF" IMU + Programmer | http://www.sparkfun.com/commerce/product_info.php?products_id=9623 http://www.sparkfun.com/commerce/product_info.php?products_id=8772 | $143 | 1 | $0 | 0.1 | 0.1 | | Scott | |
| | Logitech C600 webcam | http://www.bhphotovideo.com/c/product/647523-REG/Logitech_960_000395_C600_Webcam.html | $80 | 1 | $0 | | 0 | B&H Photo | Thomas | |
| | Quarton 635nm Laser Module Line | VLM-635-27-LPA-ND | $48 | 1 | $0 | | 0 | Digi-Key | David | |
| | General Shipping and Handling | | | | $100 | | | | | |
| | | | | Total: | $3,953 | Total Weight: | 46.25 | | | |
| | Money provided by CP Connect: | | | | 2580 | | | | | |
| | Money Pledged By Robotics Club | | | | 700 | | | | | |
| | Out of Pocket So Far | | | | 673 | | | | | |
| | | | | | | | | | | |
| Receipt Totals | Source Company | Order Description | Total | Date | | Re-Imbursed ? | | | | |
| | Pololu Order 1 | 2X High-Power Motor Controller, 2X DC-DC, 1X IMU, Connectors | $344 | 1/22/2012 | Tyson | Not yet | | | | |
| | Mcmaster Order 1 | 2X Universal Joint | $253 | 1/24/2012 | Tyson | Not yet | | | | |
| | Mcmaster Order 2 | 4X Precision Bearings for drive shafts | $121 | 1/24/2012 | Tyson | Not yet | | | | |
| | Amazon | Main Processor, Solid State Hard Drives | $889 | 1/30/2012 | Tyson | Not yet | | | | |
| | Midwest Motion Products | 2X Drive Motors | $896 | 2/7/2012 | Tyson | Not yet | | | | |
| | Headway Headquarters | 8X Battery Cells, Charger, Connectors | $551 | 2/29/2012 | Tyson | Not yet | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | | | | | | | | | |
| | | Total Spent So Far: | $2,853 | | | | | | | |
| | | Total Remaining To Spend: | $427 | | | | | | | |

# Appendix B




**Project Gantt chart**

# Appendix C

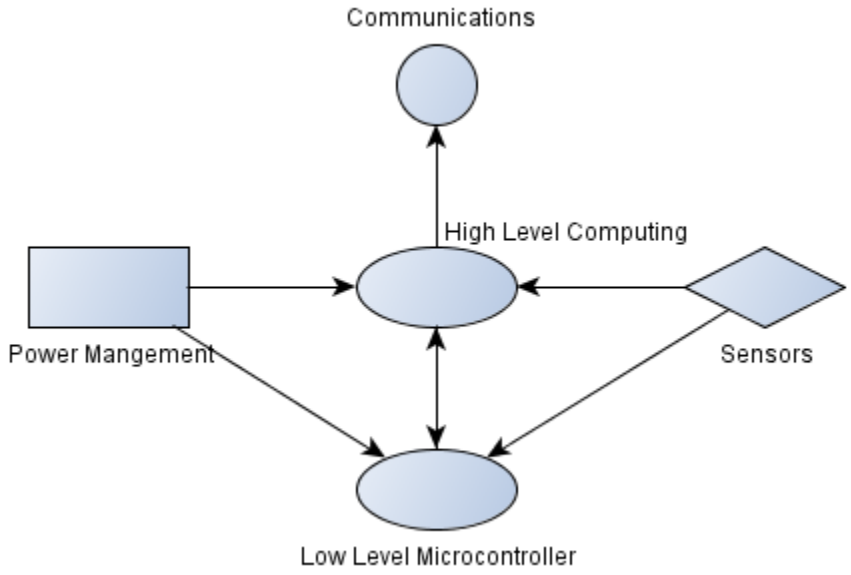



**Figure 1**. System Architecture Block Diagram

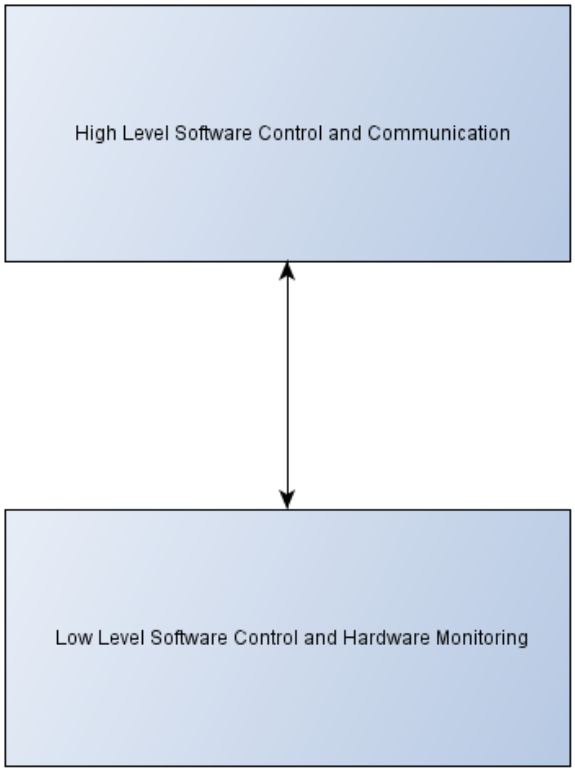



**Figure 2.** Software Relationship Diagram

```
#!/usr/bin/env python

__author__ = "Kent Williams"

import roslib; roslib.load_manifest('isus_driver')
import rospy
import serial, time


from geometry_msgs.msg import Twist

def get_command(target):
    cmdbyte=chr(0x48) #cmdByte Tag for linear x drive velocity
    serialBytes = cmdbyte+chr(target & 0x1F)+chr((target >> 5) & 0x7F)
    print serialBytes
    return serialBytes

class IsusDriver(object):
    def __init__(self):
        rospy.init_node('isus_driver')
        rospy.on_shutdown(self.ShutdownCallback)
        rospy.Subscriber("/cmd_vel",Twist, self.SteeringCallback)
        port='/dev/ttyUSB0'
        self.ser=[]
        self.timeoutmax=1000
        self.timeout=self.timeoutmax
        try:
            self.ser.append(serial.Serial(port))
            self.ser[0].open()
            self.ser[0].write(chr(0x00))
            self.ser[0].flush()
            print "opened port successfully"
        except serial.serialutil.SerialException as e:
            rospy.logerr(rospy.get_name()+": Error opening or initialising port "+port)
            exit(1)

        self.GetConfirmation()

    def GetConfirmation(self):
      self.ser[0].flush()
      self.ser[0].write('\x3F')
      print 'wrote 0x3F'
      inbyte=self.ser[0].read()
      print inbyte
      if inbyte == '\x72':
         print 'Got Confirmation Byte'
      else:
        print 'Shutting Down'
        exit(1)


    def cmdToMotor(self,linear):
        dcmd=linear
        dcmd*=100.0
        return int(dcmd)

    def run(self):

        while not rospy.is_shutdown():
          time.sleep(0.02)
          self.timeout-=20
          if self.timeout<0:
            self.ser[0].write(get_command(0))
            self.timeout=self.timeoutmax


    def ShutdownCallback(self):
        rospy.loginfo("Shutting down")
        if hasattr(self, 'ser'):
```

```
            self.ser[0].write(get_command(0))
            self.ser[0].write(chr(0xA2))
            self.ser[0].close()

    def SteeringCallback(self,data):
        self.timeout=self.timeoutmax
        dcmd = self.cmdToMotor(data.linear.x)
        logstr=": motion cmd:  Linear=" + "str(data.linear.x)"
        logstr+=" cmd=("+str(dcmd)+")"
        rospy.loginfo(rospy.get_name()+logstr )
        #print "received motor speed: "
        self.ser[0].write(get_command(dcmd))

#Init and run
IsusDriver().run()
```

**Figure 3.** ISUS MR Locomotion Driver for ROS

```
/*********************************************************************
 *
 * Software License Agreement (BSD License)
 *
 *  Copyright (c) 2009, Robert Bosch LLC.
 *  All rights reserved.
 *
 *  Redistribution and use in source and binary forms, with or without
 *  modification, are permitted provided that the following conditions
 *  are met:
 *
 *   * Redistributions of source code must retain the above copyright
 *     notice, this list of conditions and the following disclaimer.
 *   * Redistributions in binary form must reproduce the above
 *     copyright notice, this list of conditions and the following
 *     disclaimer in the documentation and/or other materials provided
 *     with the distribution.
 *   * Neither the name of the Robert Bosch nor the names of its
 *     contributors may be used to endorse or promote products derived
 *     from this software without specific prior written permission.
 *
 *  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
 *  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
 *  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
 *  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
 *  COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
 *  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
 *  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
 *  LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
 *  CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 *  LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN
 *  ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE
 *  POSSIBILITY OF SUCH DAMAGE.
 *
 *********************************************************************/
#include <stdio.h>
#include <iostream>

#include <ros/ros.h>
#include <sensor_msgs/fill_image.h>
#include <usb_cam/usb_cam.h>
#include <self_test/self_test.h>
#include <image_transport/image_transport.h>

class UsbCamNode
{
public:
  ros::NodeHandle node_;
  sensor_msgs::Image img_;
```

```
  std::string video_device_name_;
  std::string io_method_name_;
  int image_width_,image_height_;
  std::string pixel_format_name_;
  bool autofocus_;

  sensor_msgs::CameraInfo info_;

  ros::Time next_time_;
  int count_;

  usb_cam_camera_image_t* camera_image_;

  image_transport::CameraPublisher image_pub_;

  UsbCamNode() :
      node_("~")
  {
    image_transport::ImageTransport it(node_);
    image_pub_ = it.advertiseCamera("image_raw", 1);

    node_.param("video_device", video_device_name_, std::string("/dev/video0"));
    node_.param("io_method", io_method_name_, std::string("mmap")); // possible values: mmap,
read, userptr
    node_.param("image_width", image_width_, 640);
    node_.param("image_height", image_height_, 480);
    node_.param("pixel_format", pixel_format_name_, std::string("mjpeg")); // possible values:
yuyv, uyvy, mjpeg
    node_.param("autofocus", autofocus_, false); // enable/disable autofocus

    {
      XmlRpc::XmlRpcValue double_list;
      info_.height = image_height_;
      info_.width = image_width_;

      node_.param("camera_frame_id", img_.header.frame_id, std::string("head_camera"));
      info_.header.frame_id = img_.header.frame_id;

      node_.getParam("K", double_list);
      if ((double_list.getType() == XmlRpc::XmlRpcValue::TypeArray) &&
          (double_list.size() == 9)) {
        for (int i=0; i<9; i++) {
          ROS_ASSERT(double_list[0].getType() == XmlRpc::XmlRpcValue::TypeDouble);
          info_.K[i] = double_list[i];
        }
      }

      node_.getParam("D", double_list);

      if ((double_list.getType() == XmlRpc::XmlRpcValue::TypeArray)) {
        info_.D.resize(double_list.size());
        for (int i=0; i<double_list.size(); i++) {
          ROS_ASSERT(double_list[0].getType() == XmlRpc::XmlRpcValue::TypeDouble);
          info_.D[i] = double_list[i];
        }
      }

      node_.getParam("R", double_list);

      if ((double_list.getType() == XmlRpc::XmlRpcValue::TypeArray) &&
          (double_list.size() == 9)) {
        for (int i=0; i<9; i++) {
          ROS_ASSERT(double_list[0].getType() == XmlRpc::XmlRpcValue::TypeDouble);
          info_.R[i] = double_list[i];
        }
      }

      node_.getParam("P", double_list);

      if ((double_list.getType() == XmlRpc::XmlRpcValue::TypeArray) &&
          (double_list.size() == 12)) {
```

```
        for (int i=0; i<12; i++) {
          ROS_ASSERT(double_list[0].getType() == XmlRpc::XmlRpcValue::TypeDouble);
          info_.P[i] = double_list[i];
        }
      }
    }

    printf("usb_cam video_device set to [%s]\n", video_device_name_.c_str());
    printf("usb_cam io_method set to [%s]\n", io_method_name_.c_str());
    printf("usb_cam image_width set to [%d]\n", image_width_);
    printf("usb_cam image_height set to [%d]\n", image_height_);
    printf("usb_cam pixel_format set to [%s]\n", pixel_format_name_.c_str());
    printf("usb_cam auto_focus set to [%d]\n", autofocus_);

    usb_cam_io_method io_method;
    if(io_method_name_ == "mmap")
      io_method = IO_METHOD_MMAP;
    else if(io_method_name_ == "read")
      io_method = IO_METHOD_READ;
    else if(io_method_name_ == "userptr")
      io_method = IO_METHOD_USERPTR;
    else {
      ROS_FATAL("Unknown io method.");
      node_.shutdown();
      return;
    }

    usb_cam_pixel_format pixel_format;
    if(pixel_format_name_ == "yuyv")
      pixel_format = PIXEL_FORMAT_YUYV;
    else if(pixel_format_name_ == "uyvy")
      pixel_format = PIXEL_FORMAT_UYVY;
    else if(pixel_format_name_ == "mjpeg") {
      pixel_format = PIXEL_FORMAT_MJPEG;
    }
    else {
      ROS_FATAL("Unknown pixel format.");
      node_.shutdown();
      return;
    }

    camera_image_ = usb_cam_camera_start(video_device_name_.c_str(),
        io_method,
        pixel_format,
        image_width_,
        image_height_);

    if(autofocus_) {
      usb_cam_camera_set_auto_focus(1);
    }

    next_time_ = ros::Time::now();
    count_ = 0;
  }

  virtual ~UsbCamNode()
  {

    usb_cam_camera_shutdown();
  }

  bool take_and_send_image()
  {
    usb_cam_camera_grab_image(camera_image_);
    fillImage(img_, "rgb8", camera_image_->height, camera_image_->width, 3 * camera_image_-
>width, camera_image_->image);

    img_.header.stamp = ros::Time::now();
    info_.header.stamp = img_.header.stamp;
    image_pub_.publish(img_, info_);
    return true;
```

```
  }


  bool spin()
  {
    while (node_.ok())
    {
      if (take_and_send_image())
      {
        count_++;
        ros::Time now_time = ros::Time::now();
        if (now_time > next_time_) {
          std::cout << count_ << " frames/sec at " << now_time << std::endl;
          count_ = 0;
          next_time_ = next_time_ + ros::Duration(1,0);
        }
      } else {
        ROS_ERROR("couldn't take image.");
        usleep(1000000);
      }
//      self_test_.checkTest();
    }
    return true;
  }
};

int main(int argc, char **argv)
{
  ros::init(argc, argv, "usb_cam");
  UsbCamNode a;
  a.spin();
  return 0;
}
```

**Figure 4.** USB Camera ROS Node

```
#!/usr/bin/env python
import roslib; roslib.load_manifest('teleop_twist_keyboard')
import rospy

from geometry_msgs.msg import Twist

import sys, select, termios, tty

msg = """
Reading from the keyboard  and Publishing to Twist!
---------------------------
Moving around:
   u    i    o
   j    k    l
   m    ,    .

q/z : increase/decrease max speeds by 10%
w/x : increase/decrease only linear speed by 10%
e/c : increase/decrease only angular speed by 10%
anything else : stop

CTRL-C to quit
"""

moveBindings = {
               'i':(1,0),
               'o':(1,-1),
               'j':(0,1),
               'l':(0,-1),
               'u':(1,1),
               ',':(-1,0),
```

```
                '.':(-1,1),
                'm':(-1,-1),
               }

speedBindings={
                'q':(1.1,1.1),
                'z':(.9,.9),
                'w':(1.1,1),
                'x':(.9,1),
                'e':(1,1.1),
                'c':(1,.9),
              }

def getKey():
        tty.setraw(sys.stdin.fileno())
        select.select([sys.stdin], [], [], 0)
        key = sys.stdin.read(1)
        termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
        return key

speed = .5
turn = 1

def vels(speed,turn):
        return "currently:\tspeed %s\tturn %s " % (speed,turn)

if __name__=="__main__":
        settings = termios.tcgetattr(sys.stdin)

        pub = rospy.Publisher('cmd_vel', Twist)
        rospy.init_node('teleop_twist_keyboard')

        x = 0
        th = 0
        status = 0

        try:
                print msg
                print vels(speed,turn)
                while(1):
                        key = getKey()
                        if key in moveBindings.keys():
                                x = moveBindings[key][0]
                                th = moveBindings[key][1]
                        elif key in speedBindings.keys():
                                speed = speed * speedBindings[key][0]
                                turn = turn * speedBindings[key][1]

                                print vels(speed,turn)
                                if (status == 14):
                                        print msg
                                status = (status + 1) % 15
                        else:
                                x = 0
                                th = 0
                                if (key == '\x03'):
                                        break

                        twist = Twist()
                        twist.linear.x = x*speed; twist.linear.y = 0; twist.linear.z = 0
                        twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = th*turn
                        pub.publish(twist)

        except:
                print e

        finally:
                twist = Twist()
                twist.linear.x = 0; twist.linear.y = 0; twist.linear.z = 0
                twist.angular.x = 0; twist.angular.y = 0; twist.angular.z = 0
                pub.publish(twist)
```

```
termios.tcsetattr(sys.stdin, termios.TCSADRAIN, settings)
```

**Figure 5.** Teleoperation ROS Node

```
/* Author: Rob Thomson, Ken Tossell     */
/* License BSD                          */
/* URL: http://ros.org/wiki/gpsd_client */


#include <ros/ros.h>
#include <gps_common/GPSFix.h>
#include <gps_common/GPSStatus.h>
#include <sensor_msgs/NavSatFix.h>
#include <sensor_msgs/NavSatStatus.h>
#include <libgpsmm.h>

using namespace gps_common;
using namespace sensor_msgs;

class GPSDClient {
  public:
    GPSDClient() : privnode("~"), gps(NULL), use_gps_time(true), check_fix_by_variance(true) {}

    bool start() {
      gps_fix_pub = node.advertise<GPSFix>("extended_fix", 1);
      navsat_fix_pub = node.advertise<NavSatFix>("fix", 1);

      privnode.getParam("use_gps_time", use_gps_time);
      privnode.getParam("check_fix_by_variance", check_fix_by_variance);

      std::string host = "localhost";
      int port = 2947;
      privnode.getParam("host", host);
      privnode.getParam("port", port);

      char port_s[12];
      snprintf(port_s, 12, "%d", port);

      gps_data_t *resp = NULL;

#if GPSD_API_MAJOR_VERSION >= 5
      gps = new gpsmm(host.c_str(), port_s);
      resp = gps->stream(WATCH_ENABLE);
#elif GPSD_API_MAJOR_VERSION == 4
      gps = new gpsmm();
      gps->open(host.c_str(), port_s);
      resp = gps->stream(WATCH_ENABLE);
#else
      gps = new gpsmm();
      resp = gps->open(host.c_str(), port_s);
      gps->query("w\n");
#endif

      if (resp == NULL) {
        ROS_ERROR("Failed to open GPSd");
        return false;
      }

      ROS_INFO("GPSd opened");
      return true;
    }

    void step() {
#if GPSD_API_MAJOR_VERSION >= 5
      gps_data_t *p = gps->read();
#else
      gps_data_t *p = gps->poll();
```

```
#endif
      process_data(p);
    }

    void stop() {
      // gpsmm doesn't have a close method? OK ...
    }

  private:
    ros::NodeHandle node;
    ros::NodeHandle privnode;
    ros::Publisher gps_fix_pub;
    ros::Publisher navsat_fix_pub;
    gpsmm *gps;

    bool use_gps_time;
    bool check_fix_by_variance;

    void process_data(struct gps_data_t* p) {
      if (p == NULL)
        return;

      if (!p->online)
        return;

      process_data_gps(p);
      process_data_navsat(p);
    }


#if GPSD_API_MAJOR_VERSION >= 4
#define SATS_VISIBLE p->satellites_visible
#elif GPSD_API_MAJOR_VERSION == 3
#define SATS_VISIBLE p->satellites
#else
#error "gpsd_client only supports gpsd API versions 3+"
#endif

    void process_data_gps(struct gps_data_t* p) {
      ros::Time time = ros::Time::now();

      GPSFix fix;
      GPSStatus status;

      status.header.stamp = time;
      fix.header.stamp = time;

      status.satellites_used = p->satellites_used;

      status.satellite_used_prn.resize(status.satellites_used);
      for (int i = 0; i < status.satellites_used; ++i) {
        status.satellite_used_prn[i] = p->used[i];
      }

      status.satellites_visible = SATS_VISIBLE;

      status.satellite_visible_prn.resize(status.satellites_visible);
      status.satellite_visible_z.resize(status.satellites_visible);
      status.satellite_visible_azimuth.resize(status.satellites_visible);
      status.satellite_visible_snr.resize(status.satellites_visible);

      for (int i = 0; i < SATS_VISIBLE; ++i) {
        status.satellite_visible_prn[i] = p->PRN[i];
        status.satellite_visible_z[i] = p->elevation[i];
        status.satellite_visible_azimuth[i] = p->azimuth[i];
        status.satellite_visible_snr[i] = p->ss[i];
      }

      if ((p->status & STATUS_FIX) && !(check_fix_by_variance && isnan(p->fix.epx))) {
        status.status = 0; // FIXME: gpsmm puts its constants in the global
                           // namespace, so `GPSStatus::STATUS_FIX' is illegal.
```

```
        if (p->status & STATUS_DGPS_FIX)
          status.status |= 18; // same here

        fix.time = p->fix.time;
        fix.latitude = p->fix.latitude;
        fix.longitude = p->fix.longitude;
        fix.altitude = p->fix.altitude;
        fix.track = p->fix.track;
        fix.speed = p->fix.speed;
        fix.climb = p->fix.climb;

#if GPSD_API_MAJOR_VERSION > 3
        fix.pdop = p->dop.pdop;
        fix.hdop = p->dop.hdop;
        fix.vdop = p->dop.vdop;
        fix.tdop = p->dop.tdop;
        fix.gdop = p->dop.gdop;
#else
        fix.pdop = p->pdop;
        fix.hdop = p->hdop;
        fix.vdop = p->vdop;
        fix.tdop = p->tdop;
        fix.gdop = p->gdop;
#endif

        fix.err = p->epe;
        fix.err_vert = p->fix.epv;
        fix.err_track = p->fix.epd;
        fix.err_speed = p->fix.eps;
        fix.err_climb = p->fix.epc;
        fix.err_time = p->fix.ept;

        /* TODO: attitude */
      } else {
          status.status = -1; // STATUS_NO_FIX
      }

      fix.status = status;

      gps_fix_pub.publish(fix);
    }

    void process_data_navsat(struct gps_data_t* p) {
      NavSatFixPtr fix(new NavSatFix);

      /* TODO: Support SBAS and other GBAS. */

      if (use_gps_time)
        fix->header.stamp = ros::Time(p->fix.time);
      else
        fix->header.stamp = ros::Time::now();

      /* gpsmm pollutes the global namespace with STATUS_,
       * so we need to use the ROS message's integer values
       * for status.status
       */
      switch (p->status) {
        case STATUS_NO_FIX:
          fix->status.status = -1; // NavSatStatus::STATUS_NO_FIX;
          break;
        case STATUS_FIX:
          fix->status.status = 0; // NavSatStatus::STATUS_FIX;
          break;
        case STATUS_DGPS_FIX:
          fix->status.status = 2; // NavSatStatus::STATUS_GBAS_FIX;
          break;
      }

      fix->status.service = NavSatStatus::SERVICE_GPS;
```

```
      fix->latitude = p->fix.latitude;
      fix->longitude = p->fix.longitude;
      fix->altitude = p->fix.altitude;

      /* gpsd reports status=OK even when there is no current fix,
       * as long as there has been a fix previously. Throw out these
       * fake results, which have NaN variance
       */
      if (isnan(p->fix.epx) && check_fix_by_variance) {
        return;
      }

      fix->position_covariance[0] = p->fix.epx;
      fix->position_covariance[4] = p->fix.epy;
      fix->position_covariance[8] = p->fix.epv;

      fix->position_covariance_type = NavSatFix::COVARIANCE_TYPE_DIAGONAL_KNOWN;

      navsat_fix_pub.publish(fix);
    }
};

int main(int argc, char ** argv) {
  ros::init(argc, argv, "gpsd_client");

  GPSDClient client;

  if (!client.start())
    return -1;


  while(ros::ok()) {
    ros::spinOnce();
    client.step();
  }

  client.stop();
}
```

**Figure 6.** gpsd_client ROS Node

| Cable Definition Table | | | | | | |
|---|---|---|---|---|---|---|
| **Cable Span** | **Description** | **Starting Connector** | **Conductors** | **Guage (AWG)** | **Ending Connector** | **Current Capacity (A)** |
| Top chassis cavity to bottom chassis cavity signal wire | Motor 1 Encoder | P1 | 5 | 28 | M1 ENCODER | 1.4 |
| | Motor 2 Encoder | P2 | 5 | 28 | M2 ENCODER | 1.4 |
| | Steering Servo | P3 | 3 | 20 | STEERING SERVO | 3 |
| | Tilt Servo | P4 | 3 | 20 | TILT SERVO | 3 |
| | Break Servo | P5 | 3 | 20 | BREAK SERVO | 1 |
| Battery wire | Battery to BMS | B1 | 9 | 28 | BMS | 0.1 |
| | Battery to Motor Controllers | B2 | 6 | 18 | M1, M2, M3 | 16 |
| | Battery to DC-DC Converters | B3 | 4 | 20 | DC1, DC2 | 9 |
| Sensor Pod Wire | Camera Tilt Servo | S1 | 3 | 20 | CTILT SERVO | 9 |
| | Camera Pan Servo | S2 | 3 | 20 | CTILT SERVO | 9 |
| | Camera USB | S3 | 4 | 28 | CUSB | 1.4 |
| | GPS USB | S4 | 4 | 28 | GPS USB | 1.4 |
| | Sonar (3) | S5 | 9 | 28 | SONAR1, SONAR2, SONAR3 | 1.4 |
| | WIFI Antenna | S6 | 2 | ? | WIFI | 1.4 |
| | RC Reciever Antenna | S7 | 3 | 28 | RECIEVER | 1.4 |

**Figure 7.** Cable Definition Table

| Component Power Usage Table | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Component** | **Required Voltage (V)** | **Max Required Continous Current Rating (A)** | **Power Regulator** | **Source Battery** | **Max Continuous WATT Power Draw** | **Minimum Battery Life (hr):** | **Sensor Signal Voltage** |
| **High Power Ground** | | | | | | | |
| Motor Controller 1 (drive 1) | 24 | 7.5 | | 15A-Hr 25.6V Lipo | 180 | | 5 |
| Motor Controller 2 (drive 2) | 24 | 7.5 | | 15A-Hr 25.6V Lipo | 180 | | 5 |
| Motor Controller 3 (arm actuator) | 24 | 2.85 | | 15A-Hr 25.6V Lipo | 68.4 | | 5 |
| Tilt Servo | 7.4 | 2.5 | 7.4V DC-DC Converter | 15A-Hr 25.6V Lipo | 18.5 | | 7.4 |
| Steering Servo | 7.4 | 2.5 | 7.4V DC-DC Converter | 15A-Hr 25.6V Lipo | 18.5 | | 7.4 |
| Camera Tilt Servo | 7.4 | 2.5 | 7.4V DC-DC Converter | 15A-Hr 25.6V Lipo | 18.5 | | 7.4 |
| | | 25.35 | | | | | |
| **Low Power Ground** | | | | **High Power Total:** | **483.9** | **0.793552387** | |
| Breaking Servo | 5 | 0.5 | 5V DC-DC Converter | 5A-Hr 11.1V Lipo | 2.5 | | 5 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Camera Pan Servo | 5 | 0.5 | 5V DC-DC Converter | 5A-Hr 11.1V Lipo | 2.5 | | 5 |
| Parallax Propeller Board | 5 | 0.1 | 5V DC-DC Converter | 5A-Hr 11.1V Lipo | 0.5 | | |
| Fit PC | 11.1 | 0.6 | 11.1 Volt Battery | 5A-Hr 11.1V Lipo | 6.66 | | |
| IMU | 5 | 0.05 | Propeller Proto Board | 5A-Hr 11.1V Lipo | 0.25 | | 3.3-5 |
| Camera | 5 | 0.1 | Fit PC Regulator | 5A-Hr 11.1V Lipo | 0.5 | | 5 |
| GPS | 5 | 0.1 | Fit PC Regulator | 5A-Hr 11.1V Lipo | 0.5 | | 5 |
| Sonar (4) | 3.3 | 0.08 | Propeller Proto Board | 5A-Hr 11.1V Lipo | 0.264 | | 3.3 |
| Temperature Sensors | 3.3 | 0.01 | Propeller Proto Board | 5A-Hr 11.1V Lipo | 0.033 | | |
| | | **2.04** | | **Low Power Total:** | **13.707** | **4.049026045** | |
| **25.6V to 7.4V DC-DC Converter Min Watt Rating:** | **55.5** | | | | | | |
| **11.1V to 5V DC-DC Converter Min Watt Rating:** | **5.5** | | | | | | |

**Figure 8.** Component Power Usage Table

| DC-DC Converter Table | | |
|---|---|---|
| **Required Voltage Converters to communicate between sensors:** | | |
| **Device** | **Voltage Conversion** | **# of Channels** |
| Motor Control (PWM TX Only) | 3.3V to 5V | 3. |
| Servo Control (PWM TX Only) | 3.3V to 7.4V | 3 |
| Servo Control (PWM TX Only) | 3.3V to 5V | 2 |
| Sonar (RX Only) | 5V to 3.3V | 4 |

| Power Switch Requirements | | |
|---|---|---|
| Device | Voltage and current switched | # of poles |
| Low Power Switch | 11.1V, 2A | 1 |
| High Power Switch | 25.6V, 7.5A (switch will use motor controller reset line to disable main drive motors) | 1 |

**Figure 9.** DC-DC Converter Table and Power Switch Requirements

**Background Information: Clock Frequency: 80Mhz. Speed at which each processor core can access common memory: 10Mhz**

**PROPELLER CHIP TASK DISTRIBUTION AND PIN ASSIGNMENTS**

| COG | TASK | REQUIRED IO LINES (GPIO) | IO PIN ASSIGNMENTS | FREQUENCY (Hz) | SERVICE DURATION (LINES OF CODE) | APPROX CYCLES | DEPENDS ON |
|---|---|---|---|---|---|---|---|
| 1 | Initialize Cogs/Variable Space | | | Once | 2 | 20 | |
| | Gather IMU/PC Commands/Encoder data | | | | 2 | 20 | |
| | Run Balancing Algorithm | | | 200 | 40 | 400 | IMU Data, Quad Encoders |
| | Incorperate velocity/turning commands | | | 200 | 40 | 400 | |
| | Run robot tilt and camera tip calculation | | | 50 | 20 | 200 | |
| | Command Drive Motors | | | 200 | 2 | 20 | |
| | Command robot tilt and camera tip servos | | | 50 | 2 | 20 | |
| | | | Max Frequency Based on Cycles: | 9259.259259 | Total Cycles: | 1080 | |
| 2 | Get data from IMU (1 X RX) | 2 | P0-1 | 9600 kBps | 20 | 200 | IMU sensor board |
| | Parse data and load into variable cach | | | | 10 | 100 | |
| | | | | | | 0 | |
| 3 | Read servo command variables<br>Generate 5 hobby servo compatible PWM signals<br>3 lines will be converted to 7.4V signals<br>2 lines will be converted to 5V signals | 5 | P8-12 | 20 kHz | 50 | 500 | PC Commands Tip and Tilt Algorithm |
| 4 | Record Quadrature encoder data from 3 quadrature encoders at up to 20 kHz<br>Check to see if the quadrature encoders can be run at 3.3V | 6 | P2-7 | 20 kHz | 50 | 500 | Quadrature encoders |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 5 | Motor Control PWM generation for 3 motor contorllers at 20 kHz. PWM & Direction Lines all 3 lines will need to be converted to 5V signals | 6 | P16-21 | 20 Khz | 50 | 500 | Commands from balancing algorithm |
| 6 | Load IR absolution position sensor, arm position commands, IMU rotation acceleration data | | | 100 | 10 | 100 | IR Absolution position sensor data coming from ADC |
| | Look for falling event and trigger quick reaction arm movement accordingly | | | 100 | 5 | 50 | IMU rotation acceleration data |
| | Run Arm Actuator control algorithm (including current control) | | | 100 | 20 | 200 | Commands from PC |
| | Command Arm Actuator Drive Motor | | | 100 | 5 | 50 | |
| | Communicate with 4 sonar sensors (Read Pulse Widths of 4 lines after triggering in succession). Trigger and PWM read lines for each sensor. | 8 | P22-29 | 10 kHz | 40 | 400 | Sonar sensors |
| | Read RC reciever E-Stop signal (1 soft off signal - other channel will control a PWM controlled relay) | 1 | P15 | 10 kHz | 20 | 200 | RC Reciever signals |
| | | | Max Frequency Based on Cycles: | 10000 | Total Cycles: | 1000 | |
| 7 | Look for Commands/Telemetry requests from PC, Perform Serial parsing, load data into cach variables. | 1 | P31 | A-Syncronous | 100 | 1000 | Serial communication from PC |
| | SPI communication with ADC. 2 voltage sensors, 3 current sensors, 2 temperature sensors, 1 IR absolution position sensor on Arm actuator. | 3 | | 50 | 40 | 400 | Analog sensor input |
| | Check temperature and voltage values and trigger error buzzer if | 0 | P1 | 50 | 10 | 100 | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | out of spec.<br><br>Send telemetry data to PC. | 1 | P30<br>Max Frequency Based on Cycles: | 6666.666667 | Total Cycles: | 1500 | |
| 8 | Viewport Debugging/Monitoring | 0 | | 200 | 100 | 1000 | Variables from balancing algorithm |

| | |
|---|---|
| **TOTAL REQUIRED IO:** | **33** |
| TOTAL AVAILABLE IO: | **32** |

**Figure 10.** Propeller Task and Pin Assignments

| PC SERIAL COMMUNICAION PACKET PROTOCOL | | | |
|---|---|---|---|
| **GROUP** | **DATA DESCRIPTION** | **# OF BYTES** | **MIN FREQ (Hz)** |
| **Packets Sent From Microcontroller to PC** | | | |
| **1** | Wheel encoder ticks | 12 (3 X 32bit numbers) | 50 |
| | IMU Orientation | 6 (3 X 16bit angle numbers) | 50 |
| | Servo positions | 5 (5 X 8bit angle numbers) | 50 |
| | Sonar Ranges | 4 (4 X 8bit range numbers) | 50 |
| **Data Description:** | Left32, Right32, Arm32, X16, Y16, Z16, Steering8, RobotTilt8, Break8, CameraTilt8, CameraPan8, | 27 Bytes Total | |
| **2** | Temperatures | 3 (3 X 8bit temperature numbers) | 10 |
| | Currents | 3 (3 X 8 bit current numbers) | 10 |
| | Voltages | 2 (2 X 8 bit voltage numbers) | 10 |
| | Error Codes | 1 (1 X 8 bit error code number) | 10 |
| | | 9 Bytes Total | |
| **Packets Sent From PC To Microcontroller** | | | |
| **1** | Robot Velocity Command | 1(1 X 8bit number) | 50 |
| | Robot Turning Radius Command | 1(1 X 8bit number) | 50 |
| | Steering Servo Angle | 1(1 X 8bit number) | 50 |
| | Camera Pan Angle | 1(1X 8bit number) | 50 |
| | Arm Actuator Angle | 1(1X 8bit number) | 50 |
| | | 5 Bytes Total | |

| **Voltage Conversion PIN Talley** | |
|---|---|
| **OUT** | |
| **IO PINs 3.3V to 5V:** | 6 for Motor Control, 2 for Servos, (8 total) |
| **IO Pins 3.3 to 7.4V:** | 3 for Servos |
| **IN** | |
| **IO Pins 5V to 3.3V:** | 4 for Sonar Sensor |

**Figure 11.** PC to Microcontroller Communication Packet Protocol Definition

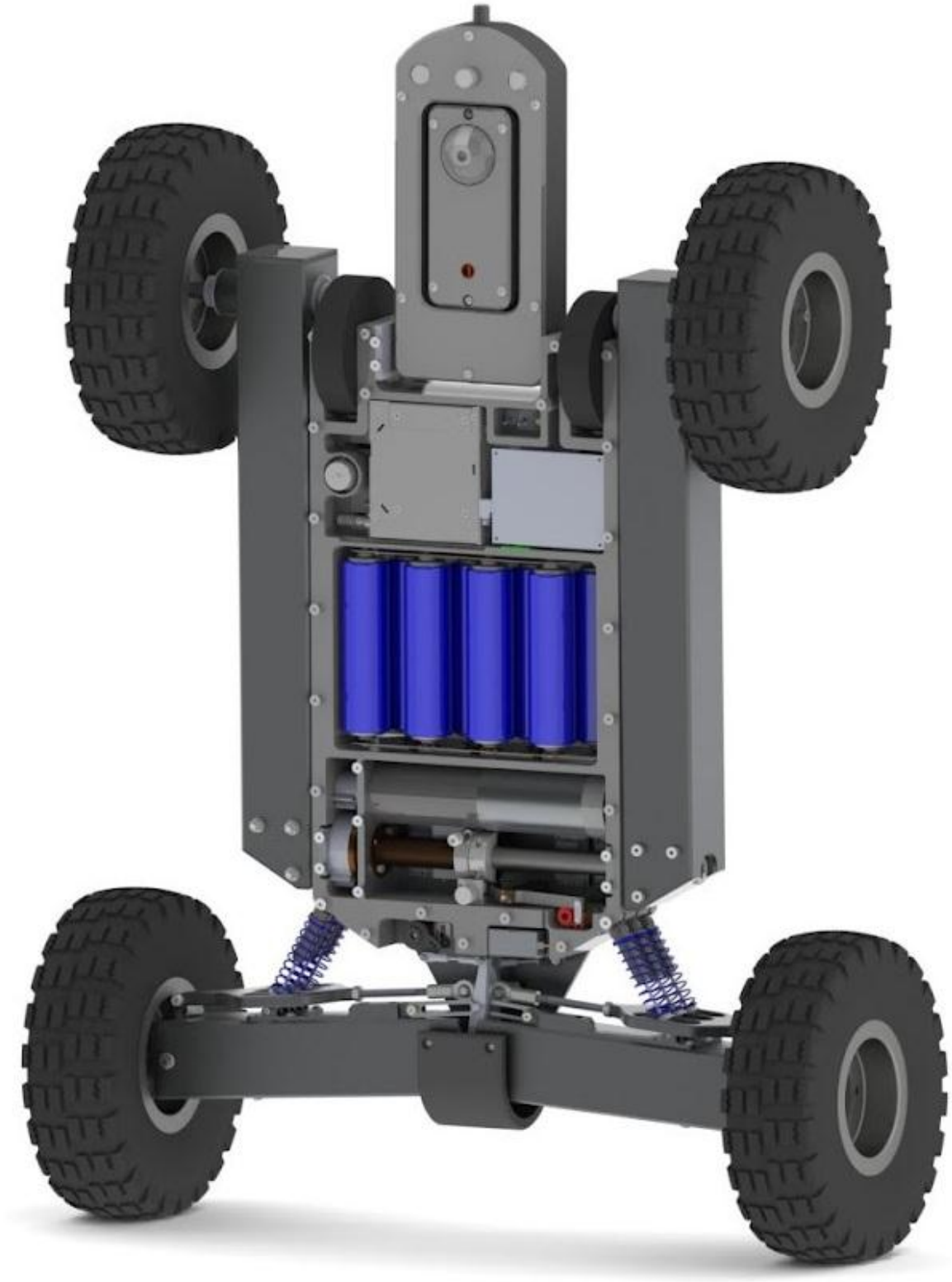


**Figure 12**. ISUS Robot Front, Cover Removed

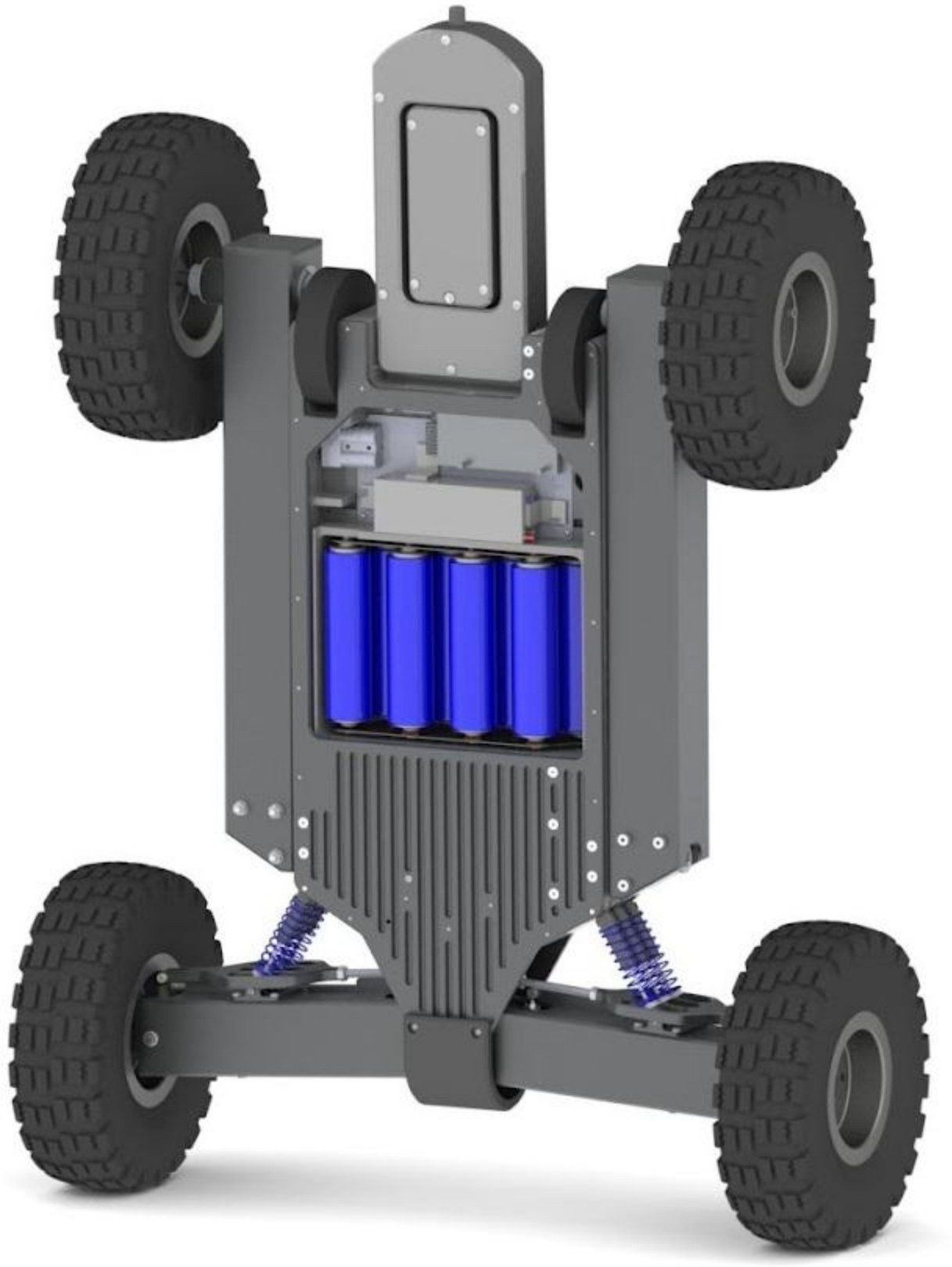


**Figure 13**. ISUS Robot Back, Cover Removed

# Appendix D

## Project Contributions of Kent Williams

The ISUS MR was a joint project between Tyson Messori and Kent Williams. My contributions to this project were the design and implementation of the high level software and components needed to autonomize this vehicle as well as make it accessible to the operator. The problem description was well understood from the beginning and as such the selection of components was evaluated early on. In order to determine the appropriate components for this vehicle, past experience and knowledge from prior robotic projects was called upon. The functionality needed could be broken into two pieces, that of the vision, gps, and sonar sensors, for observing the environment and providing localization capabilities. The second piece was the communication aspect between the operator, the autonomy system, and the control communication with the lower level microcontroller. Originally, I had thought an additional computer would be needed to spread the load of vision processing to a dedicated embedded arm linux board. After reviewing our options for the main vehicle computer, it became apparent that all the high level software services could be successfully run on the fit-PC2i. The next stage in development was to choose a software architecture for retrieving the necessary data from the sensors and allowing the exchange of that data between the localization and navigation algorithms. For this need, the Robotic Operation System was chosen as the most suitable infrastructure for sensors to communicate data with the high level algorithms. All of the high level services components needed for autonomy are contained inside ROS. Refer to figure 2, for the overall high level software diagram. This report has documented these efforts and described the domain in which I contributed.