**AboveTheClaw**

**Critical Design Review (CDR)**

**By:**

**Peter Corcoran**

**Evan Gilbert**

**Ben Henson**

**Kevin Houston**

**Aaron McDaniel**

**Terence Staples**

**Leah Watkins**

**2 November 2015**

**EE 498 – Senior Design**

**Department of Electrical and Computer Engineering**

**The University of Alabama at Birmingham**

**Table of Contents**

[Section 1: Outline of System 1-1](#_Toc434233307)

[Section 2: Overview 2-1](#_Toc434233308)

[2.1 Delivery & Storage 2-1](#_Toc434233309)

[2.2 Cargo Retrieval 2-2](#_Toc434233310)

[2.3 Image Processing & Lighting 2-3](#_Toc434233311)

[2.4 Propulsion 2-4](#_Toc434233312)

[2.5 Navigation 2-6](#_Toc434233313)

[2.6 Microcontroller & Logistics 2-7](#_Toc434233314)

[2.7 Power & Chassis 2-8](#_Toc434233315)

[Section 3: Chassis Layout 3-1](#_Toc434233316)

[3.1 Delivery & Storage 3-1](#_Toc434233317)

[3.1.1 CAD Drawings 3-1](#_Toc434233318)

[3.1.2 Issues 3-2](#_Toc434233319)

[3.1.3 Circuit Board Layout 3-3](#_Toc434233320)

[3.2 Cargo Retrieval 3-4](#_Toc434233321)

[3.2.1 CAD Drawings 3-4](#_Toc434233322)

[3.2.2 Issues 3-9](#_Toc434233323)

[3.2.3 Circuit Board Layout 3-10](#_Toc434233324)

[3.3 Image Processing & Lighting 3-13](#_Toc434233325)

[3.3.1 CAD Drawings 3-13](#_Toc434233326)

[3.3.2 Issues 3-14](#_Toc434233327)

[3.3.3 Circuit Board Layout 3-14](#_Toc434233328)

[3.4 Propulsion 3-15](#_Toc434233329)

[3.4.1 CAD Drawings 3-15](#_Toc434233330)

[3.4.2 Issues 3-15](#_Toc434233331)

[3.4.3 Circuit Board Layout 3-15](#_Toc434233332)

[3.5 Navigation 3-17](#_Toc434233333)

[3.5.1 CAD Drawings 3-17](#_Toc434233334)

[3.5.2 Issues 3-18](#_Toc434233335)

[3.5.3 Circuit Board Layout 3-18](#_Toc434233336)

[3.6 Microcontroller & Logistics 3-19](#_Toc434233337)

[3.6.1 CAD Drawings 3-19](#_Toc434233338)

[3.6.2 Issues 3-20](#_Toc434233339)

[3.6.3 Circuit Board Layout 3-20](#_Toc434233340)

[3.7 Power & Chassis 3-21](#_Toc434233341)

[3.7.1 CAD Drawings 3-21](#_Toc434233342)

[3.7.2 Issues 3-24](#_Toc434233343)

[3.7.3 Circuit Board Layout 3-24](#_Toc434233344)

[Section 4: Interfaces 4-1](#_Toc434233352)

[4.1 Delivery & Storage 4-1](#_Toc434233353)

[4.2 Cargo Retrieval 4-3](#_Toc434233354)

[4.3 Image Processing & Lighting 4-4](#_Toc434233355)

[4.4 Propulsion 4-5](#_Toc434233356)

[4.5 Navigation 4-7](#_Toc434233357)

[4.6 Microcontroller & Logistics 4-8](#_Toc434233358)

[4.7 Power & Chassis 4-10](#_Toc434233359)

[Section 5: Schematics 5-1](#_Toc434233360)

[5.1 Delivery & Storage 5-1](#_Toc434233361)

[5.2 Cargo Retrieval 5-2](#_Toc434233362)

[5.3 Image Processing & Lighting 5-3](#_Toc434233363)

[5.4 Propulsion 5-4](#_Toc434233364)

[5.5 Navigation 5-5](#_Toc434233365)

[5.6 Microcontroller & Logistics 5-6](#_Toc434233366)

[5.7 Power & Chassis 5-7](#_Toc434233367)

[Section 6: Software Design 6-1](#_Toc434233368)

[6.1 Delivery & Storage 6-2](#_Toc434233369)

[6.1.1 Representational Layer 6-2](#_Toc434233370)

[6.1.2 Link Layer 6-3](#_Toc434233371)

[6.2 Cargo Retrieval 6-4](#_Toc434233372)

[6.2.1 Representational Layer 6-4](#_Toc434233373)

[6.2.2 Link Layer 6-5](#_Toc434233374)

[6.3 Image Processing & Lighting 6-7](#_Toc434233375)

[6.3.1 Representational Layer 6-7](#_Toc434233376)

[i) Link Layer 6-9](#_Toc434233377)

[6.4 Propulsion 6-10](#_Toc434233378)

[6.4.1 Representational Layer 6-10](#_Toc434233379)

[6.4.2 Link Layer 6-11](#_Toc434233380)

[6.5 Navigation 6-12](#_Toc434233381)

[6.5.1 Representational Layer 6-12](#_Toc434233382)

[6.5.2 Link Layer 6-14](#_Toc434233383)

[6.6 Microcontroller & Logistics 6-15](#_Toc434233384)

[6.6.1 Representational Layer 6-15](#_Toc434233385)

[6.6.2 Link Layer 6-26](#_Toc434233386)

[6.6.3 Settings 6-27](#_Toc434233387)

[6.7 Power & Chassis 6-32](#_Toc434233388)

[6.7.1 Representational Layer 6-32](#_Toc434233389)

[6.7.2 Link Layer 6-32](#_Toc434233390)

[Section 7: Weight Estimates 7-1](#_Toc434233391)

[7.1 Summary 7-1](#_Toc434233392)

[7.2 Delivery & Storage 7-2](#_Toc434233393)

[7.3 Cargo Retrieval 7-3](#_Toc434233394)

[7.4 Image Processing & Lighting 7-7](#_Toc434233395)

[7.5 Propulsion 7-8](#_Toc434233396)

[7.6 Navigation 7-9](#_Toc434233397)

[7.7 Microcontroller & Logistics 7-10](#_Toc434233398)

[7.8 Power & Chassis 7-11](#_Toc434233399)

[Section 8: Energy Estimates 8-1](#_Toc434233400)

[Section 9: Possible Problems & Solutions 9-1](#_Toc434233401)

[9.1 Delivery & Storage 9-1](#_Toc434233402)

[9.2 Cargo Retrieval 9-2](#_Toc434233403)

[9.3 Image Processing & Lighting 9-3](#_Toc434233404)

[9.4 Propulsion 9-4](#_Toc434233405)

[9.5 Navigation 9-5](#_Toc434233406)

[9.6 Microcontroller & Logistics 9-6](#_Toc434233407)

[9.7 Power & Chassis 9-7](#_Toc434233408)

[Section 10: Implementation Plan 10-1](#_Toc434233409)

[10.1 Delivery & Storage 10-1](#_Toc434233410)

[10.2 Cargo Retrieval 10-2](#_Toc434233411)

[10.3 Image Processing & Lighting 10-5](#_Toc434233412)

[10.3.1 Software Coding 10-5](#_Toc434233413)

[10.3.2 Hardware 10-5](#_Toc434233414)

[10.3.3 Wiring 10-5](#_Toc434233415)

[10.3.4 Mounting 10-5](#_Toc434233416)

[10.3.5 Software Loading 10-5](#_Toc434233417)

[10.4 Propulsion 10-6](#_Toc434233418)

[10.5 Navigation 10-7](#_Toc434233419)

[10.6 Microcontroller & Logistics 10-8](#_Toc434233420)

[10.6.1 Software Coding 10-8](#_Toc434233421)

[10.6.2 Execute Test Plan 10-8](#_Toc434233422)

[10.6.3 Chassis Mounting 10-8](#_Toc434233423)

[10.6.4 Wiring 10-8](#_Toc434233424)

[10.6.5 Software Loading 10-8](#_Toc434233425)

[10.7 Power & Chassis 10-9](#_Toc434233426)

[10.7.1 First Floor 10-9](#_Toc434233427)

[10.7.2 Second Floor 10-9](#_Toc434233428)

[10.7.3 Third Floor 10-9](#_Toc434233429)

[10.7.4 Fourth Floor 10-9](#_Toc434233430)

[Section 11: Testing Plan 11-1](#_Toc434233431)

[11.1 Delivery & Storage 11-1](#_Toc434233432)

[11.2 Cargo Retrieval 11-7](#_Toc434233433)

[11.3 Image Processing & Lighting 11-13](#_Toc434233434)

[11.3.1 BeagleBone Connectivity Test 11-13](#_Toc434233435)

[11.3.2 BeagleBone Connectivity via Putty Test 11-13](#_Toc434233436)

[11.3.3 Electronic Switch Test 11-14](#_Toc434233437)

[11.3.4 Image Capture Test 11-15](#_Toc434233438)

[11.3.5 Pixel Recognition Test 11-15](#_Toc434233439)

[11.3.6 QR Code Recognition Test 11-16](#_Toc434233440)

[11.3.7 Size Recognition Test 11-17](#_Toc434233441)

[11.3.8 Size Recognition Test 11-17](#_Toc434233442)

[11.3.9 Lighting Source Test 11-18](#_Toc434233443)

[11.3.10 Lighting Source to Controller Test 11-19](#_Toc434233444)

[11.3.11 Lighting Source to Controller via Variable Resistor Test 11-19](#_Toc434233445)

[11.3.12 BeagleBone to Image Processing Test 11-20](#_Toc434233446)

[11.3.13 Wiring Space Test 11-21](#_Toc434233447)

[11.4 Propulsion 11-22](#_Toc434233448)

[11.5 Navigation 11-26](#_Toc434233449)

[11.6 Microcontroller & Logistics 11-28](#_Toc434233450)

[11.6.1 BeagleBone to Delivery Controller Test 11-28](#_Toc434233451)

[11.6.2 BeagleBone with Delivery Servos Test 11-28](#_Toc434233452)

[11.6.3 BeagleBone to Arm Controller Test 11-29](#_Toc434233453)

[11.6.4 BeagleBone to Lidar Controller Test 11-30](#_Toc434233454)

[11.6.5 BeagleBone to Motor Controller Test 11-31](#_Toc434233455)

[11.6.6 BeagleBone to Image Processing Test 11-32](#_Toc434233456)

[11.7 Power & Chassis 11-34](#_Toc434233457)

[Section 12: Appendix A – Parts List 12-1](#_Toc434233458)

[Section 13: Appendix B – Cable & Connectors 13-1](#_Toc434233459)

[Section 14: Appendix C – Additional Information 14-1](#_Toc434233460)

**List of Figures**

[Figure 1. BeagleBone Black Microcontroller 2-7](#_Toc434233461)

[Figure 2. Front and Top view of Storage Solution 3-1](#_Toc434233462)

[Figure 3. Isometric view of Storage Solution 3-1](#_Toc434233463)

[Figure 4. Front View of Storage Solution 3-2](#_Toc434233464)

[Figure 5. Rear View of Storage Solution 3-2](#_Toc434233465)

[Figure 6. Firgelli Controller Board 3-3](#_Toc434233466)

[Figure 7. Gantry System for Cargo Retrieval 3-4](#_Toc434233467)

[Figure 8. Cargo Retrieval Base Joint 3-4](#_Toc434233468)

[Figure 9. Cargo Retrieval Pincher 3-5](#_Toc434233469)

[Figure 10. Cargo Retrieval Linkage 3-5](#_Toc434233470)

[Figure 11. Cargo Retrieval Servo Shoe 3-6](#_Toc434233471)

[Figure 12. Cargo Retrieval Servo Apron 3-6](#_Toc434233472)

[Figure 13. Cargo Retrieval Shoulder Spacer 3-7](#_Toc434233473)

[Figure 14. Cargo Retrieval Shoulder Spacer 3-7](#_Toc434233474)

[Figure 15. Cargo Retrieval Effector Paddle 3-8](#_Toc434233475)

[Figure 16. Cargo Retrieval Effector Hinge 3-8](#_Toc434233476)

[Figure 17. Cargo Retrieval End Bar 3-9](#_Toc434233477)

[Figure 18. Pin Layout of USB2AX Servo Motor Controller Board 3-11](#_Toc434233478)

[Figure 19. Data and Power Pins of an AX Series Dynamixel servo motor 3-11](#_Toc434233479)

[Figure 20. Big Easy Driver Version 1.2 Motor Controller Board Contact Layout 3-12](#_Toc434233480)

[Figure 21. Multiple LED array 3-13](#_Toc434233481)

[Figure 22. Single LED 3-13](#_Toc434233482)

[Figure 23. Circuit Board Layout 3-14](#_Toc434233483)

[Figure 24. Control Signal Input 3-15](#_Toc434233484)

[Figure 25. Output Terminals for H-Bridge Controller 3-16](#_Toc434233485)

[Figure 26. LIDAR CAD Drawing 3-17](#_Toc434233486)

[Figure 27. Side view of LIDAR System 3-17](#_Toc434233487)

[Figure 28. Rendered Isometric view of LIDAR System 3-18](#_Toc434233488)

[Figure 29. BeagleBone Black CAD Drawing 3-19](#_Toc434233489)

[Figure 30. BeagleBone Black Proto Board 3-19](#_Toc434233490)

[Figure 31. Chassis 3D View 3-21](#_Toc434233491)

[Figure 32. Chassis First Floor 3-22](#_Toc434233492)

[Figure 33. LIDAR Floor 3-22](#_Toc434233493)

[Figure 34. BeagleBone Black Floor 3-23](#_Toc434233494)

[Figure 35. Top Floor 3-23](#_Toc434233495)

[Figure 36. Switching Regulator 3-24](#_Toc434233496)

[Figure 37. 12-volt Regulator 3-25](#_Toc434233497)

[Figure 38. Propulsion Interface Diagram 4-6](#_Toc434233498)

[Figure 39. Microcontroller Hardware Schematic & Interface Diagram 4-9](#_Toc434233499)

[Figure 40. Cargo Retrieval Schematic Diagram 5-2](#_Toc434233500)

[Figure 41. Microcontroller System Schematic 5-6](#_Toc434233501)

[Figure 42. Power distribution circuit 5-7](#_Toc434233502)

[Figure 43. BLAZE Software Layers 6-1](#_Toc434233503)

[Figure 44. Competition boards or Ports 6-1](#_Toc434233504)

[Figure 45. Delivery Controller Class 6-2](#_Toc434233505)

[Figure 46. Cargo Retrieval Class 6-4](#_Toc434233506)

[Figure 47. Image Processing & Lighting Class 6-7](#_Toc434233507)

[Figure 48. Motor Controller Class 6-10](#_Toc434233508)

[Figure 49. Navigation Controller Class 6-12](#_Toc434233509)

[Figure 50. Port Class Diagram 6-15](#_Toc434233510)

[Figure 51. Floor Class Diagram 6-17](#_Toc434233511)

[Figure 52. Boundary Class Diagram 6-17](#_Toc434233512)

[Figure 53. Point Class Diagram 6-18](#_Toc434233513)

[Figure 54. Zone Class Diagram 6-20](#_Toc434233514)

[Figure 55. Robot Controller Class Diagram 6-21](#_Toc434233515)

[Figure 56. Logistics Controller Class Diagram 6-24](#_Toc434233516)

[Figure 57. Route Controller Class Diagram 6-25](#_Toc434233517)

[Figure 58. Belt Configuration 7-6](#_Toc434233518)

**List of Tables**

Table 1. Delivery & Storage Interfaces 4-1

Table 2. Interface for the Cargo Retrieval Module 4-3

Table 3. Image Processing & Lighting Interfaces 4-4

Table 4. Propulsion Interfaces 4-5

Table 5. Navigation Interfaces 4-7

Table 6. Delivery Controller Properties 6-2

Table 7. Delivery Controller Behavior: getStorageColor 6-2

Table 8. Delivery Controller Behavior: deliverContainer 6-3

Table 9. Cargo Retrieval Controller Properties 6-4

Table 10. Cargo Retrieval Controller Behavior: getBlock 6-4

Table 11. Cargo Retrieval Controller Behavior: setPosition 6-5

Table 12. Cargo Retrieval Controller Behavior: open 6-5

Table 13. Cargo Retrieval Controller Behavior: close 6-5

Table 14. Cargo Retrieval Controller Behavior: moveArm 6-5

Table 15. Lighting Controller Properties 6-7

Table 16. Lighting Controller Properties: ON 6-7

Table 17. Camera Controller Properties: takeImage 6-8

Table 18. Image Controller: getColor 6-8

Table 19. Image Controller: getSize 6-8

Table 20. Image Controller: getQR 6-8

Table 21. Image Controller: getPosition 6-8

Table 22. Image Controller: idRailColor 6-8

Table 23. Motor Controller Properties 6-10

Table 24. Motor Controller Behavior: engage 6-10

Table 25. Motor Controller Behavior: brake 6-10

Table 26. Motor Controller Behavior: slower 6-11

Table 27. Motor Controller Behavior: faster 6-11

Table 28. Motor Controller Behavior: getSpeed 6-11

Table 29. Navigation Controller Properties 6-12

Table 30. Navigation Controller Behavior: getCurentPosition 6-12

Table 31. Navigation Controller Behavior: forward 6-13

Table 32. Navigation Controller Behavior: Reverse 6-13

Table 33. Navigation Controller Behavior: turn 6-13

Table 34. Navigation Controller Behavior: navigateTo 6-13

Table 35. Navigation Controller Behavior: NavigateRoute 6-14

Table 36. Navigation Controller Behavior: navigateThroughTunnel 6-14

Table 37. Port Properties 6-15

Table 38. Port Behavior: getTime 6-16

Table 39. Port Behavior: determinePort 6-16

Table 40. Floor Properties 6-17

Table 41. Boundary Properties 6-17

Table 42. Point Behavior: x 6-18

Table 43. Point Behavior: y 6-19

Table 44. Point Behavior z 6-19

Table 45. Point Behavior: theta 6-19

Table 46. Point Behavior: dist 6-19

Table 47. Point Behavior: add 6-19

Table 48. Point Behavior: sub 6-19

Table 49. Point Behavior: move 6-20

Table 50. Zone Properties 6-20

Table 51. Robot Controller Properties 6-21

Table 52. Robot Control Entry Function 6-22

Table 53. Robot Control Start Function 6-22

Table 54. Robot Control Initialize Settings Function 6-22

Table 55. Robot Control System Check Function 6-22

Table 56. Robot Control System Behavior: hardwareCheck 6-22

Table 57. Robot Control System Behavior: sensorCheck 6-23

Table 58. Robot Control System Behavior: readyStateCheck 6-23

Table 59. Robot Control System Behavior: startFromRoute 6-23

Table 60. Logistics Controller Properties 6-24

Table 61. Logistics Controller Behavior: startLogistics 6-24

Table 62. Logistics Controller Behavior: takeInventory 6-24

Table 63. Logistics Controller Behavior: loadInventory 6-25

Table 64. Logistics Controller Behavior: deliverInventory 6-25

Table 65. Route Properties 6-25

Table 66. Robot Settings 6-27

Table 67. Delivery & Storage Weight Table 7-2

Table 68. Weight Budget 7-3

Table 69. Gantry Weight Budget Used to Select Stepper Motor Torque and Weight 7-4

Table 70. Propulsion Wegith Table 7-8

Table 71. Navigation Weight Table 7-9

Table 72. Microcontroller Weight Table 7-10

Table 73. Delivery & Storage Problem/Solutions 9-1

Table 74. Cargo Retrieval Problem/Solutions 9-2

# Outline of System

AboveTheClaw has determined modular design components to assist in the engineering process of the Project.  Modules and responsible team members are delineated below:

1. Delivery/Storage - Leah Watkins
2. Cargo Retrieval - Evan Gilbert
3. Image Processing & Lighting - Aaron McDaniel
4. Propulsion - Kevin Houston
5. Navigation - Terence Staples
6. Microcontroller - Peter Corcoran
7. Logistics - Peter Corcoran
8. Chassis - Ben Henson
9. Power - Ben Henson

The purpose of the project is to build an autonomous robot (AR) capable of completing a course defined by “*Hardware Competition IEEE SoutheastCon 2016*”. The AR is made from a composite of multiple modules, controllers, and functional systems. Figure 1 shows the AR’s system layer diagram (SLD), which defines AR sections, section requirements, and the section’s solution. Specific AR features correspond to requirements of the hardware competition, and are linked to the specified solution.

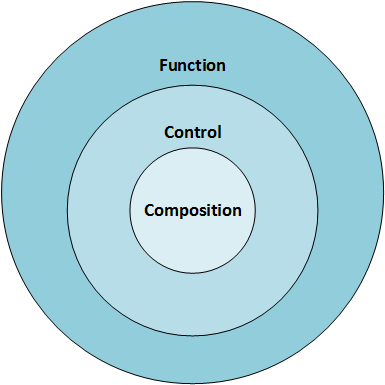
 

Figure 1. Autonomous Robot System Layer Diagram.

# Overview

## Delivery & Storage

Leah Watkins

The Delivery & Storage module was designed with two primary objectives in mind.

1. Carry a sufficient amount of blocks to make only one trip to and from the barge area per delivery zone.
2. Engineer an all-inclusive solution to accommodate each delivery area (Rail, Boat, and Truck)

*Delivery Overview*

There are two different methods of delivery:

Method one is mechanical in nature and uses the arm to lift the back end of a false-bottom plank to be located beneath the stack of blocks.  When the false-bottom is lifted, the blocks tip out and into the appropriate delivery area.  Each stack of blocks is capable of being unloaded one slot at a time with this method as required.  Method one will be used in the delivery of blocks into the Rails area with the elevator lifted at a height of 6”, or the boat with the elevator lowered at the ground.

Method two uses a single linear actuator located at the first floor of the chassis to unload three vertical storage units concurrently. The actuator forces the blocks directly outward and into the truck or boat zone while the elevator is in a fully lowered position.

*Storage Overview*

The solution incorporates the use of an elevator mechanism to lift and lower a series of four vertical storage units.  The elevator system is driven by a single Firgelli L16 linear actuator (LA).   The LA is mounted vertically, parallel to the side of the storage system.  The storage system itself will include the interface point for the LA to push upwards and will be installed on a v-rails system on the chassis.  The v-rails solution will possess one driven and one passive rail to ensure control of the storage unit during travel.  The storage system will be dedicated to storing 4-5 blocks, for a total of 16-20 blocks depending on the barge area from which they were obtained.  The elevator height is adjusted according to the delivery zone, 6” for the rail zone, and fully lowered for the boat and truck.  The elevator systems LA is controlled with pulse width modulation (PWM).  By using an output pin from the master controller, the desired actuator position is encoded as the duty cycle of a 3.3 Volt, 1 kHz square wave on the linear actuator controller (LAC).  The percent duty cycle sets the actuator position to the same percent of full stroke extension. For example, 100% duty cycle represents full extension, and 0% duty cycle represents full retraction.  In order to allow for the LIDAR navigation solution to maintain 360 degrees of visibility, the elevator solution will travel high for the duration of the course except for the boat and truck deliveries where is may be lowered.

The Delivery and Storage solution is modular in nature and lends itself to a straightforward implementation process while successfully meeting each of the primary objectives set forth in the initial design process.

## Cargo Retrieval

Evan Gilbert

An overwhelming 88.636% of the points in the 2016 IEEE SoutheastCon Hardware Competition come directly from block manipulation, so the Cargo Retrieval subsystem is definitely critical to the performance of the vehicle.  Since there exist blocks in different sizes placed at varying heights, and the contest rules cap the global build envelope, this subsystem must also be both adaptive and compact.  A robotic arm with a moveable base solves these issues.  By using a gantry system to relocate the arm, fewer stop-and-go movements are required of the vehicle – the tail doesn’t wag the dog, so to speak.  Additionally, smart servos trained with preset height differentials make it easy to “flagpole” a camera, which relays Image Processing & Lighting (IPL) data to assist with finer, sighted arm control.  And, by incorporating a belt-and-pinion configuration using modular off-the-shelf parts backed by an open source community, it’s possible to design a low-profile gantry that can also be cheaply outfitted with 3D printed nylon and ABS parts.

FEATURES:

* 3D Printed Nylon 6-DOF Arm is light enough to prevent vehicular pitching
* Belt-&-Pinion Gantry allows for a concealed belt with few moving parts
* Parallel-Grip End Effector
* On-Board Feedback Control of Servo Motors allows for trainability
* Wrist Mounted with Webcam so that the IPL module can help guide arm operations
* Cable Management solution(s) not limited to any one growth scenario of the vehicle

DUTIES:

* Articulating a camera such that it can map environmental artifacts
* Retrieving and Relaying to the Delivery & Storage subsystem all wooden blocks
* Actuating BOAT and RAIL Deliveries through Delivery & Storage interfacing

## Image Processing & Lighting

Aaron McDaniel

The image processing system was designed with the objective of determining the size, color, shape, QR code, and position of the blocks for pickup to assist the autonomous robot’s arm in providing all the knowledge needed to grab the block. The image processing system was also designed for a purpose of obtaining the order of the rail bins on the given playing field to assist the logistics aspect of the autonomous robot.

The image processing system will employ a Microsoft HD LifeCam 5000, utilizing an ffmpeg command to capture the image, connected to a BBB via USB. The USB connection will provide power and also transfer information.

The lighting system was designed with the objective of lighting the frame of a picture to assist the autonomous robot’s camera. The lighting design will assist the webcam in lighting up the frame of the picture to take out the delay of auto adjusting brightness level built in to the webcam itself.

The lighting design will consist of six white-wash LEDs and connected to the G-9612 bus bar via an electronic switch for state purposes.

## Propulsion

Kevin Houston

*Overview*

The propulsion system was designed with the objective of propelling the autonomous robot through the initial tunnel and to various points on the track in the fastest and most efficient way possible. The time limit of 300 seconds was observed heavily when considering that the robot needs to be able to successfully travel to various points successfully with repeatable results.

*Motor Choice*

The first step was the choice of DC or AC motors. DC motors were chosen to allow multiple functionality with respect to being able to rotate in reverse through simple means.

The necessary torque was calculated with respect to a proposed wheel diameter of greater than 3” and a weight estimate of 12lbs loaded weight. An overview of the calculations can be seen below:

*torque =*

12lb - 192 oz.

3.15” wheel diameter - 1.57” wheel radius

The resulting torque calculated is 96 oz.-in. Therefore, the motors chosen should have much *greater* torque capability than 96 oz.-in.

Initially, the goal velocity was proposed to complete the course within the 300 seconds allotted. The greater value is gained by using the equation shown below:

(velocity(in/s))/(1in) = (rad/s) \* = rev/s \* 60 = Necessary RPM

The goal velocity set forth was determined to be 18in/s, by using that as the initial value the resulting goal rpm is 172RPM. Therefore, the chosen motors should have not only a torque greater than 96 oz.-in, but a rpm capability greater than 172 rpm.

*Motor Control*

The propulsion system needs a liaison between the motors, the microcontroller and the power source. The choice selected here was to select a capable motor controller for two motors. A RioRand DC Dual Motor Controller will be used to receive PWM signals from the microcontroller, the BeagleBone Black in order to transfer the necessary current from the battery to the motors. The specs that are important about the dual motor controller are its rated current at 15A and its peak current at 30A. These specs are necessary to provide as much current durability for the motors chosen.

*Wheel Implementation*

Since only two motors will be used, the front two wheels will be mounted using a mounted axle support system. The design was to choose a suitable L-Bracket mount in which to place a specified axle for a wheel bearing to rotate about. The parts selected met the specifications for this design. The wheel bearing was chosen to have a suitable mount in which the specified wheel and sprocket may be attached. via #4-40 1” screws. The sprocket will be placed on the outermost part while the wheel will be placed in the middle between the wheel bearing mount and the sprocket. The sprocket is placed on the outermost part as the space is needed to observe the behavior of the chain.

## Navigation

Terence Staples

The navigation system was designed with the objective of determining the autonomous robot’s position on the playing board and using this position to assist locomotion and logistics to get to the needed positions on the playing board.

The navigation system will utilize a LIDAR unit, repurposed from a Neato XV-11 Robotic Vacuum, alongside a SLAM (Simultaneous Localization and Mapping) algorithm to determine and communicated position to the BeagleBone Black microcontroller. The LIDAR unit will be powered and communicate to the BeagleBone Black through an externally powered USB hub.

## Microcontroller & Logistics

Peter Corcoran

The microcontroller and logistics system is used to coordinate all sub-system components. Its purpose is to read inputs from all components, process the inputs, and supply results to the same or another component. A BeagleBone Black (BBB) is used as the core microcontroller in the system. The BeagleBone is a low cost ARM Cortex-A8 based processor. The microcontroller supplies a wide range of interface capabilities, and allows for add-on boards (called capes). The BeagleBone features that are critical to the system design include:

* 1GHz Processor
* 512MB DDR3L
* 2GB Onboard Managed NAND (eMMC)
* USB 2.0 Client Port
* Two 46 Pin Connectors

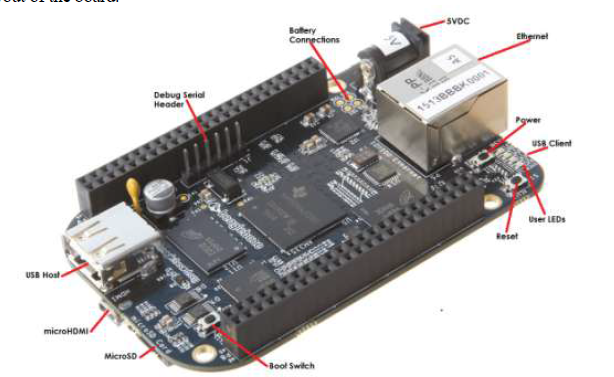


Figure . BeagleBone Black Microcontroller

Although the BeagleBone offer a number of features that support the major robot’s function, additional connections are required to manage other major components. Therefore, a USB hub will also be used to interface components to the BBB. The navigation, arm, and camera systems will be connected via the powered USB hub. The delivery system will be using two linear actuator controller board to interface with the BBB using pulse-width modulation (PWM).

The logistics system running on the microcontroller BLAZE (Birmingham’s Logistics Actuating Zone Evaluator) will employ a representational set of objects that track the robot and environment changes over the course of a “run”. BLAZE will update object state in real-time while tracking robot location, and inventory. BLAZE will also keep logs of system debug information, and be updateable over wireless communications (updates not available during competition)

## Power & Chassis

Ben Henson

The power distribution circuit power all components. It will need to power each device regardless of its specifications. The specifications for each component have been limited to either 5 volts or 12 volts. All 12 volt components will either be associated with pulse width modulation or will have internal voltage regulation. Five volts regulated will be supplied by the power distribution system. This regulated 5 volts will be provided by a synchronous switching step-down regulator. This regulator takes an input voltage of up to 38 V and efficiently reduces it to 5 V. The lithium polymer (lipo) battery that will be used is 14.8 volts. Every voltage regulation device that is used within the robot is compatible with the voltage supplied by the battery. The power distribution circuit will have a slow blow fuse in series with the positive terminal of the battery. A toggle switch will also be placed in series with the positive terminal of the battery. The raw voltage from the battery will have is only voltage terminal block. The 5 volts regulated terminal will have its only terminal block both voltage values will share a common ground terminal.

The chassis will house all aspects of the robot. Each part, every element must be accounted for and space within the chassis must be allocated for it. The chassis will consist of four main floors. The first floor will house the two propulsion motors and the battery. The rest of the first floor will be allocated to the elevator system. The second floor will be completely dedicated to the LIDAR system. This floor is minimal in height, consisting of only 0.7 inches. The third floor will house all processors, controllers, grounding and voltage terminals. The fourth floor will be dedicated to the arm.

# Chassis Layout

## Delivery & Storage

Leah Watkins

### CAD Drawings

The images below represent measurements and dimensions as they pertain to the storage unit. As one can see, there are four vertical units in which blocks will be stored.

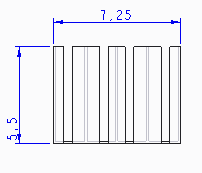
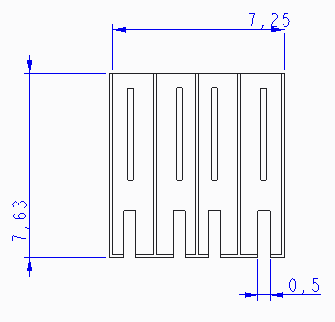


Figure . Front and Top view of Storage Solution

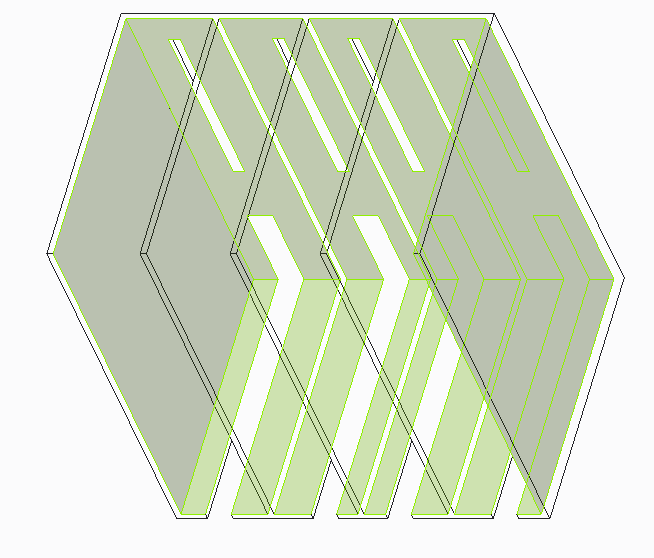


Figure . Isometric view of Storage Solution

The image below represents the element used in Delivery Method #2, the “rake” element. In the second figure, the rake element is identified in red. A small linear actuator is mounted at the center bar to push the element out horizontally. Because the blocks will be pushed from the bottom, the entire stack will be forced outward. This solution was designed primarily for efficient block delivery at the truck.

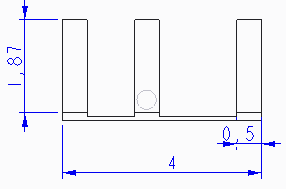


Figure . Front View of Storage Solution

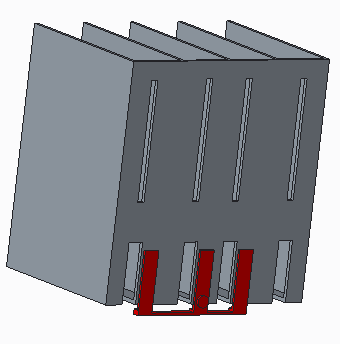


Figure . Rear View of Storage Solution

### Issues

Space is a critical design issue. The storage/delivery solution takes up a majority of the 12x12x12 size constraint leaving little room for other components like hardware and wiring.

Efficient operation of the LIDAR navigation system is also a critical design issue. LIDAR requires a 360-degree field of view. By allowing the storage taco to ride hide throughout most of the course ensures that LIDAR field of view is not obstructed.

### Circuit Board Layout

The figure below represents the LAC board layout. Indicated in red are the two points of interface which pertain to the delivery and storage solution. This controller is currently designed and manufactured by Firgelli for use with their L16 actuator.

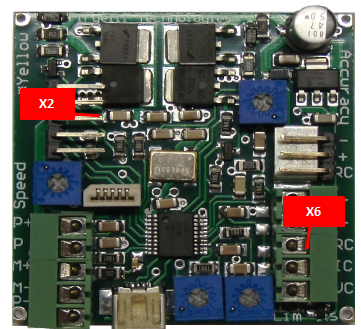


Figure . Firgelli Controller Board

The X2 connector is where the cable from the linear actuator will terminate. The X6 connector is where the controller will interface with the controller and power systems. For additional information on interconnects including specific pin numbers please refer to the connector/cable list included in Appendix B.

## Cargo Retrieval

Evan Gilbert

### CAD Drawings

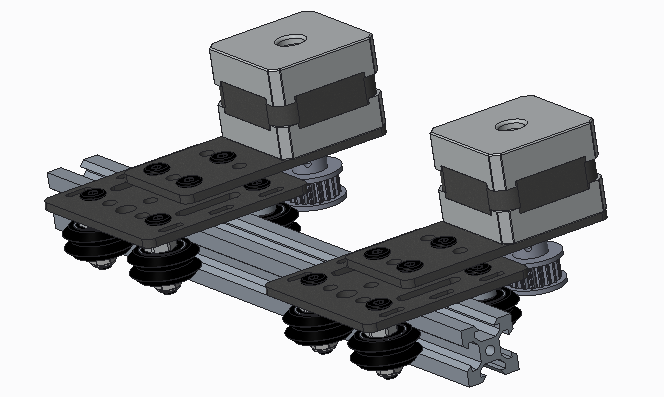


Figure . Gantry System for Cargo Retrieval

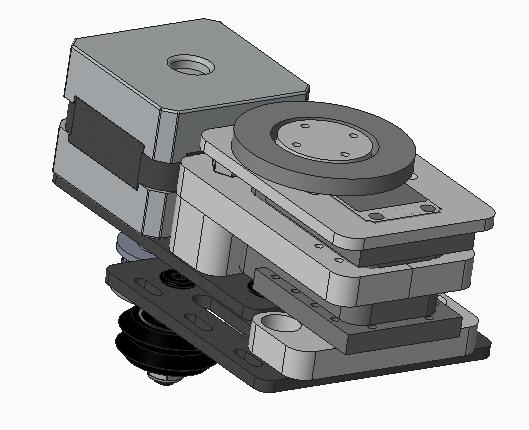


Figure . Cargo Retrieval Base Joint



Figure . Cargo Retrieval Pincher

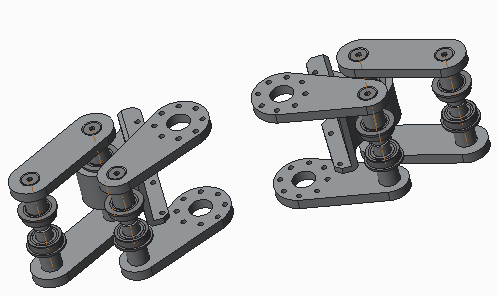


Figure . Cargo Retrieval Linkage

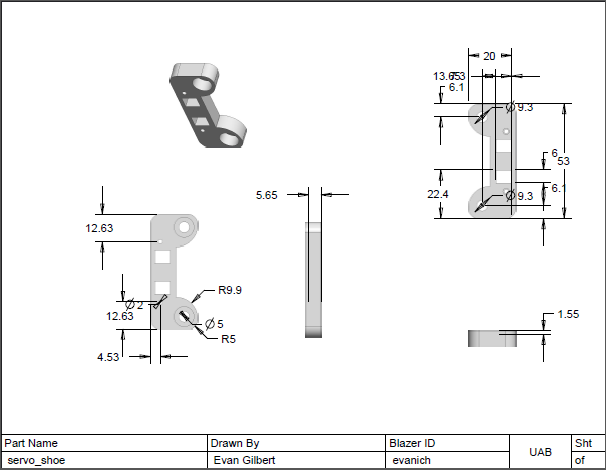


Figure . Cargo Retrieval Servo Shoe

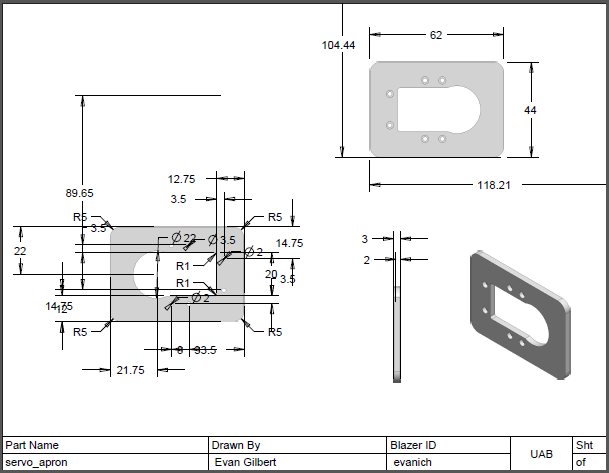


Figure . Cargo Retrieval Servo Apron

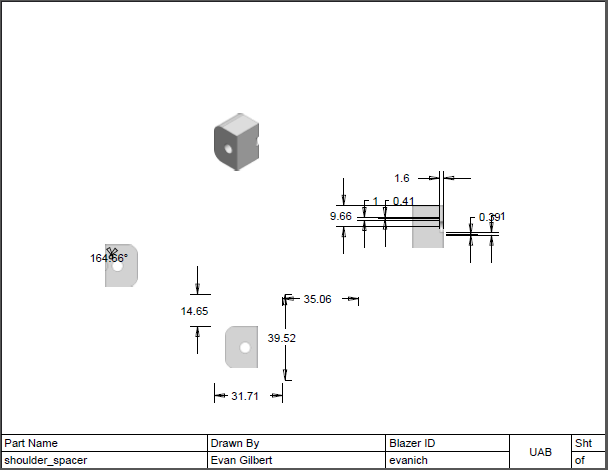


Figure . Cargo Retrieval Shoulder Spacer

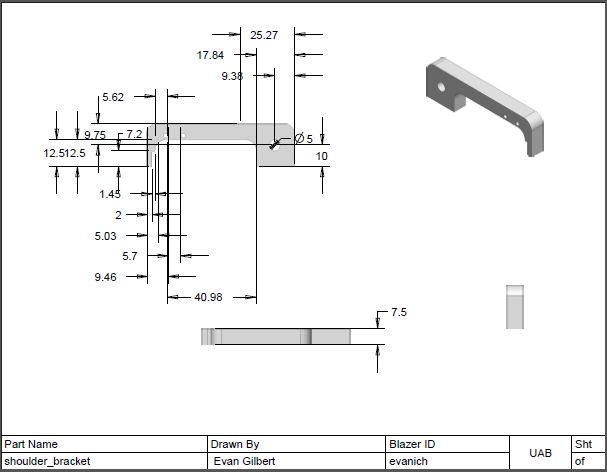


Figure . Cargo Retrieval Shoulder Spacer

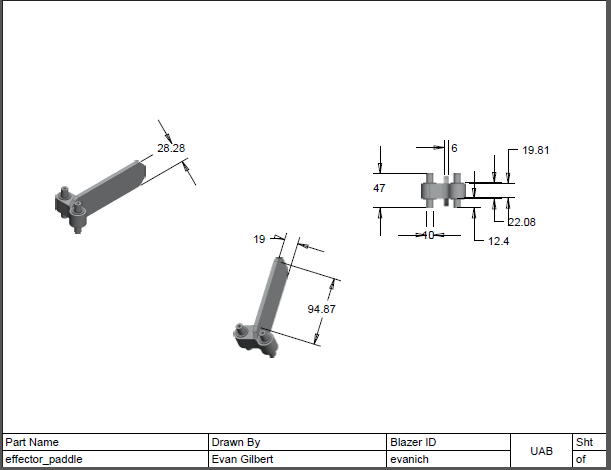


Figure . Cargo Retrieval Effector Paddle

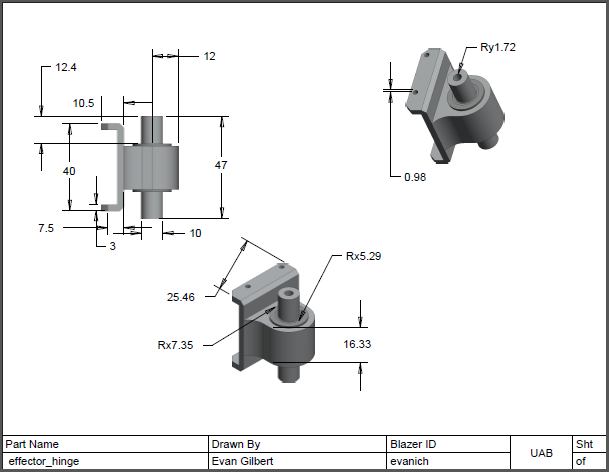


Figure . Cargo Retrieval Effector Hinge

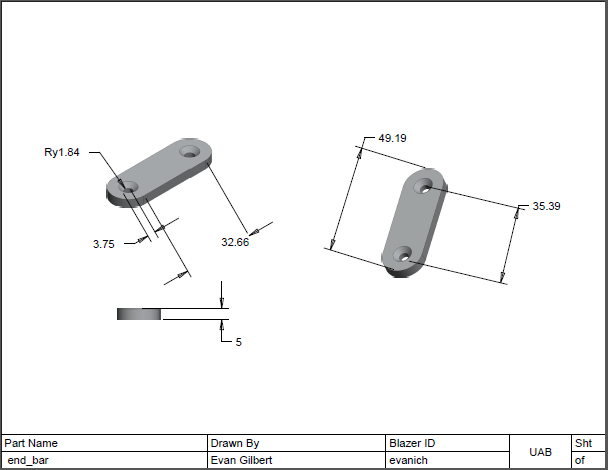


Figure . Cargo Retrieval End Bar

### Issues

***Cable management:*** Gantry mobility includes the movement of all associated wires, none of which must be permitted to move freely without introducing a host of crippling liabilities. Because the vehicle build is subject to change, 3 plausible solutions have been explored: (1) a drag chain can be placed before the gantry, such that it is horizontally arched and suspended just above the LIDAR FOV, where it rendezvouses with the master controller and power elements; (2) the cable bundle can be bound in a flexible wrap and *slacked* toward the elevator, where it is secured to the outwardly facing, unused channel of the v-rail elevator infrastructure; (3) the cable can be fixed to a flagpole and allowed to dangle provided the length of bundled cable does not invite opportunities for entanglement from elements internal and external to the vehicle.

***Shifting Center of Gravity:*** Because a large U-shaped section has been removed from the vehicle rear, carved out for an elevator shaft that routes cargo, and a gantry is to be run on the top floor, opposite the elevator, the primary components of both the gantry and robot arm must be light enough to preclude any operational movements that might pitch the vehicle or introduce some performance instability. Heavier components, where space and heat transfer permit, should be centrally grouped to counterweight the gantry so that when fully extended beyond the vehicle perimeter, the arm does not pose a tipping concern, despite the makeup of low-weight parts. Should such a problem slip through implementation and testing, and a teardown aimed at reorienting the vehicle weight will not suffice, reduced gantry speeds and ballast will be considered.

***Manipulation of Delivery & Storage Actuator Implement:*** Given the use of smart servos, the arm can be trained in a range of gestures by capturing the numeric tags that correspond to the movements of each servo horn, and programmatically registering them in a software solution that brokers the logic driving each actionable sequence of the arm. Finer, or localized, motor control, however, will require some cooperation with the IPL module – the trained “muscle memory” is primarily used to plant the camera nearer to some target area, where image processing is then used to guide the acquisition efforts.

***Collecting all Zone A blocks:*** A gantry, a 300° shoulder joint, and a 300° wrist should allow for easy retrieval of Zone A blocks placed in such a way that the RAIL bins physically obstruct further vehicular advances. Should a time bonus exist that diminishes any incentive to collect the blocks described above, the modular design of the arm will permit a rebuild that excludes the now unnecessary shoulder joint. Instead of tossing the servo, it would be coupled with the existing elbow actuator (now making up the base of the arm) in order to facilitate a speedier volley between pick-up and storage.

### Circuit Board Layout

No custom manufacture of a board is required at this time. What follows is a quick reference for the pin layouts native to the Cargo Retrieval module.

The servo motor controller is shown in Figure 18. Pin Layout of USB2AX Servo Motor Controller Board. The USB points — VBUS, D-, D+, and GND, connect to a powered USB hub that busses traffic between the master controller and its sensors and actuators. (Note: pictured in Figure 1 is a pin layout of board version 3.1a, which is identical to the version 3.2a board pin layout that will be used in this project.) Servo logic is passed, in Figure 11, from the 3-pin TTL cable that connects a chain of servos to board points PB1-3. There is no need to physically reduce to a single wire the 3-pin TTL cable that connects to the 12-V energized actuator chain and this 5-V USB device because this board has been designed with some protective circuitry forethought.

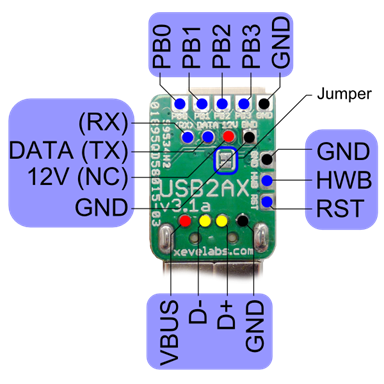


Figure . Pin Layout of USB2AX Servo Motor Controller Board

For quick reference, the servo motor pin layout is illustrated in Figure 19. Data and Power Pins of an AX Series Dynamixel servo motor.



Figure . Data and Power Pins of an AX Series Dynamixel servo motor

The pin layout of the stepper-motor controller is shown in Figure 20. Big Easy Driver Version 1.2 Motor Controller Board Contact Layout. The board features a meaningful redundancy of contacts so that the topology of the circuit can be spec’d to the project hardware: the pin spacing is not fixed, which allows for quicker implementation and also introduces minor improvements to fitting this board within the project real estate. Referring to Figure 20, the motor controller will interface with the microcontroller at the following points, or sister points: EN, M1, M2, M3, STEP, and DIR. The A± and B± contacts will be tied directly to the corresponding 4-wire leads of the stepper motor. Note: hardware not shown in Figure 20 does not exceed the height of the drum capacitor.

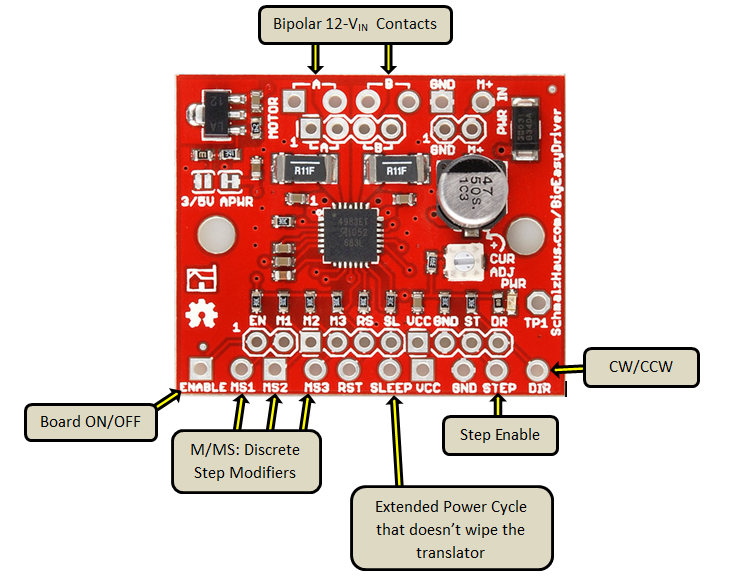


Figure . Big Easy Driver Version 1.2 Motor Controller Board Contact Layout

## Image Processing & Lighting

Aaron McDaniel

### CAD Drawings

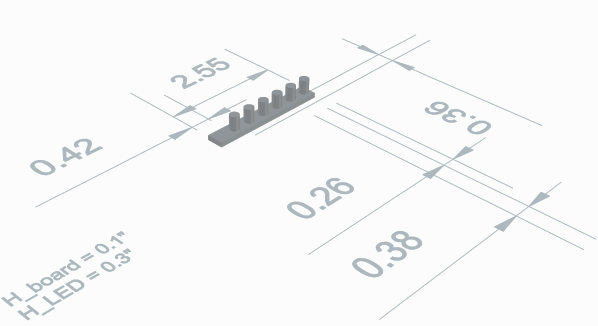


Figure . Multiple LED array

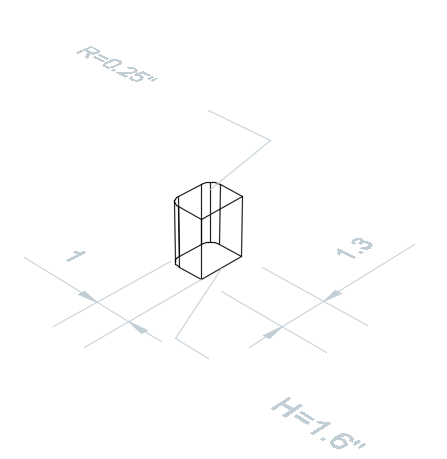


Figure . Single LED

### Issues

Potential issues could arise if the static equations of the arm are not accurate which have a negative effect on the perceived functioning of the arm.

### Circuit Board Layout

There is an electronic transistor switch designed for the state of the light source. The 5V voltage source is connected to an electronic transistor, which in turn is connected to a ground controller through a Zener diode and resistor and the lighting source.

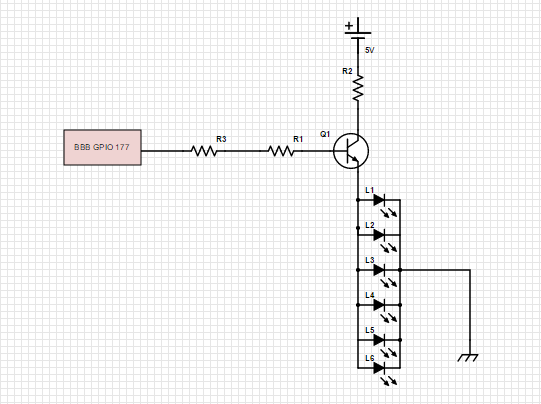


Figure . Circuit Board Layout

## Propulsion

Kevin Houston

### CAD Drawings

### Issues

The potential issue with the propulsion system in reference to its interaction with the chassis is the implementation of the wheel set. The front two wheels will be used as casters; however, it is needed for them to be tied to the back wheels for an all-wheel drive train. The issue arises with the mounting of the front tires in a stable manner. The specification considering the height of the chassis is that it needs to be 0.5” from the ground, the wheels need to be mounted in such a way that they allow the bottom of the chassis to be at that distance designated in the design. S set of two blocks are to be designed and will be 3D printed for the back motors and their mounts to be placed on. Reference Figure X to view the drawing and specifications of the designed blocks for the front wheel mounts and the back mounts respectively. The height of the block is drawn from taking the placement of the DC motor’s shaft while in the mount from the measured distance that is necessary for the bottom of the chassis to be 0.5” from the floor of the track.

### Circuit Board Layout

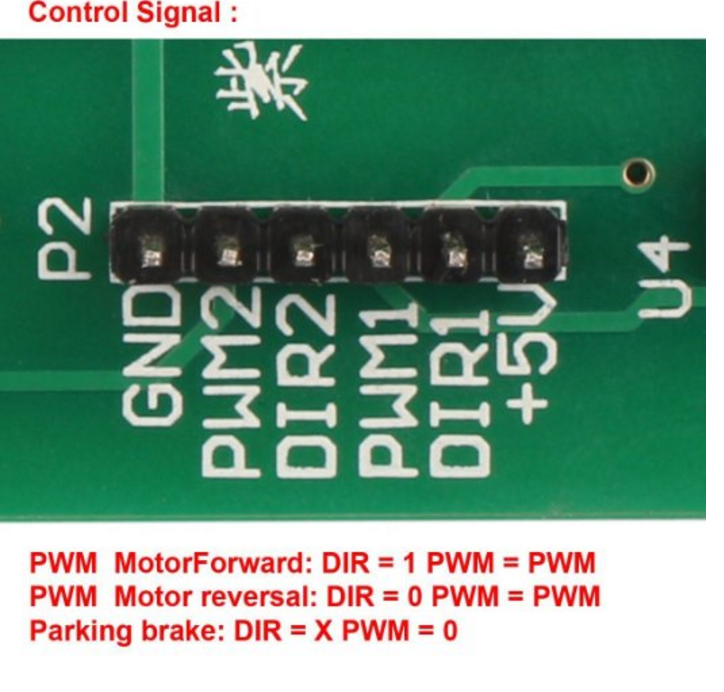


Figure . Control Signal Input

Shown above in Figure 24, the control signal pins for the RioRand dual motor controller are shown. The dual motor controller will receive PWM signals sent from the BeagleBone Black. DIR[#] = 1 and PWM = PWM will set the motor controller to set the motors to move forward. the argument should be PWM Motor Forward. To reverse a specified motor, the DIR [#] = 0 and PWM = PWM, this shall reverse the specified motor. DIR1 and DIR2 will be used for the two separate motors. A parking brake can also be implemented by the signal DIR[#] = X and PWM = 0. This sophisticated controller will be able to receive simple commands for the operations needed of the motors.

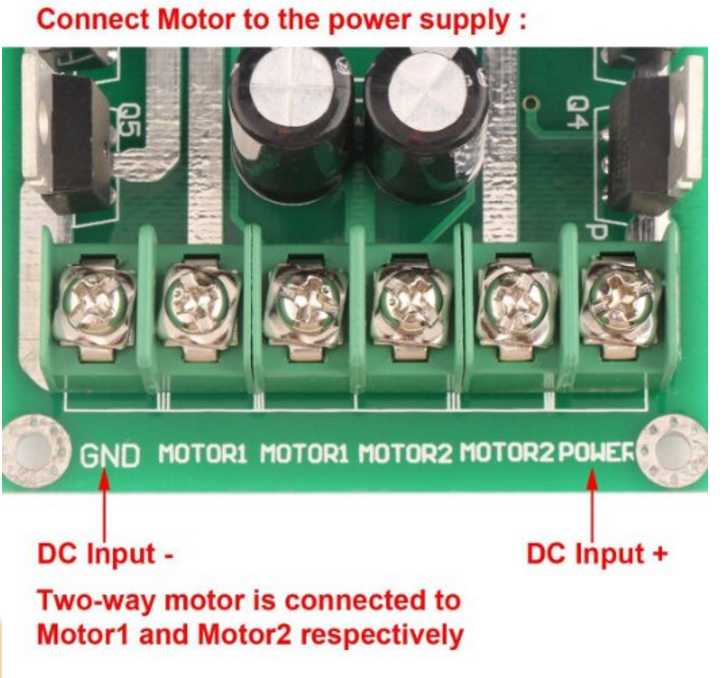


Figure . Output Terminals for H-Bridge Controller

The pin layout for the controller to interact with the battery and motors is shown above. The negative DC input to connect to the battery is at the GND pin, and the positive DC input is at the POWER input. The specified motors will be connected to the respective pins for MOTOR1 and MOTOR2.

## Navigation

Terence Staples

### CAD Drawings

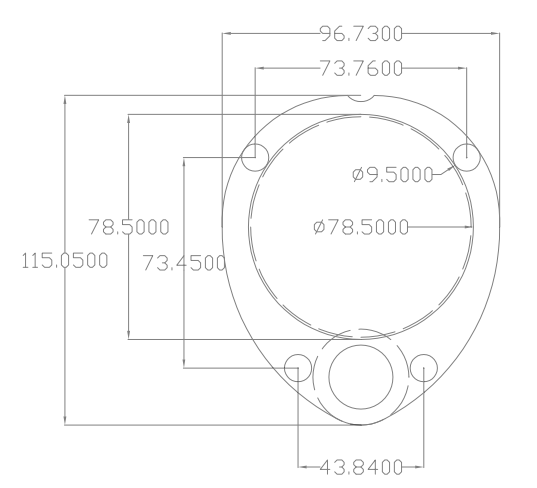


Figure . LIDAR CAD Drawing

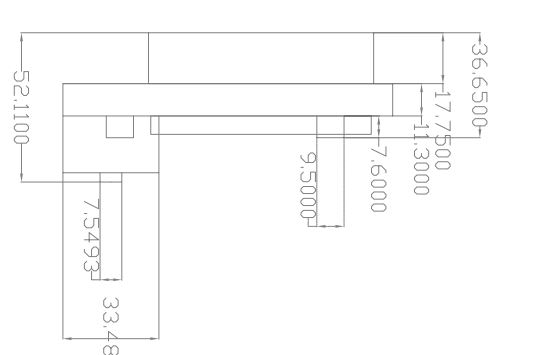


Figure . Side view of LIDAR System



Figure . Rendered Isometric view of LIDAR System

### Issues

The biggest issue of the LIDAR is the data readings are invalid at a certain distance (<=100mm). This can be combated since the data reading are 360 degrees. Another problem is that the LIDAR cannot have a 360-degree view of the board due to other parts of the robot. However, navigation can be achieved without a full 360 degrees.

### Circuit Board Layout

The LIDAR unit includes a teensy board that allows for motor control and interface through USB. The boards dimensions are 1 x 1.75 in.

## Microcontroller & Logistics

Peter Corcoran

### CAD Drawings

CAD drawing of the BBB and the BBB proto-board are for sizing only. No custom CAD drawings are needed.

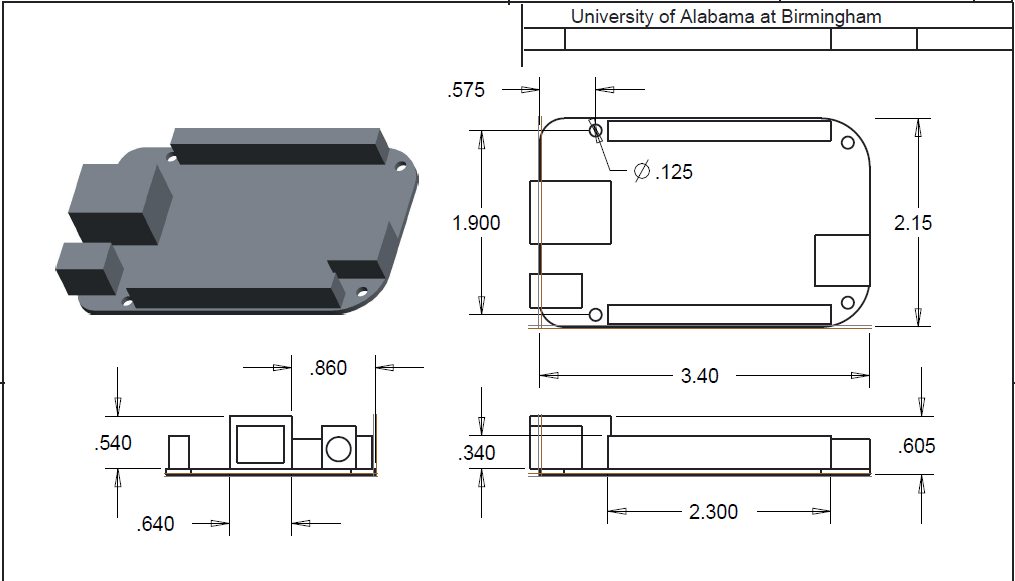


Figure . BeagleBone Black CAD Drawing

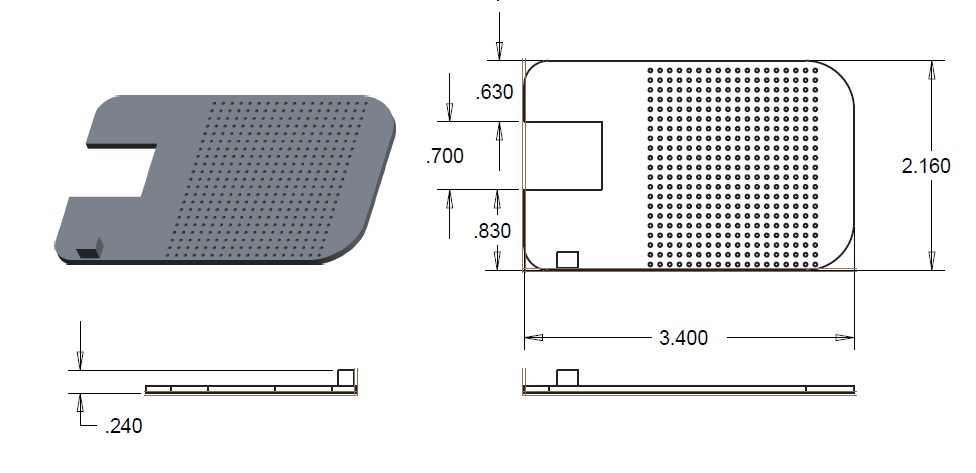


Figure . BeagleBone Black Proto Board

### Issues

The potential issue with the controller system and chassis mounting is how the controller is mounted between floors of the chassis. The controller needs clearance between header pins and the floor above the controller.

### Circuit Board Layout

A proto-board will be used and components and component connectors will be soldered to the board. **Error! Reference source not found.** shows the BBB layout and connections to each other system. The flexibility of using a proto-board allows for easy connection of components.

## Power & Chassis

Ben Henson

### CAD Drawings

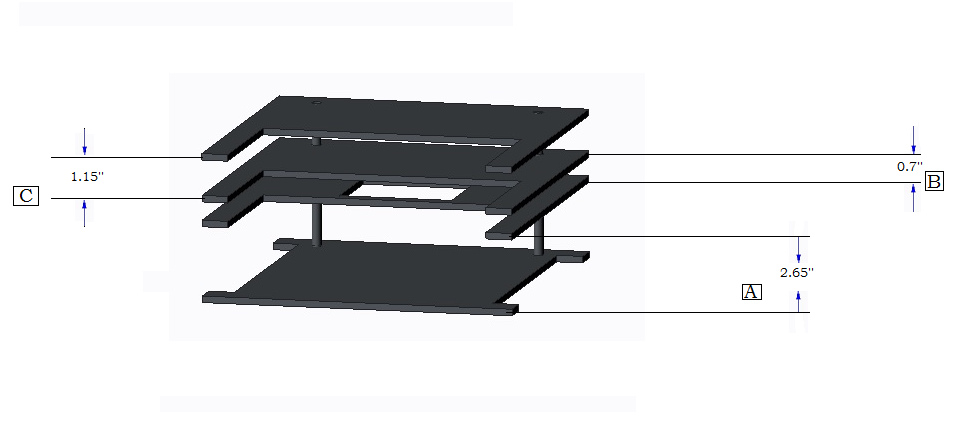


Figure . Chassis 3D View

Figure 31. Chassis 3D View shows the 3 dimensional view of the chassis design. The floors are separated by the letters A, B and C. A shows the distance between the first floor and the LIDAR floor. B shows the distance between the LIDAR floor and the processor floor (BBB). C shows the distance between the processor floor (BBB) and the arm floor.

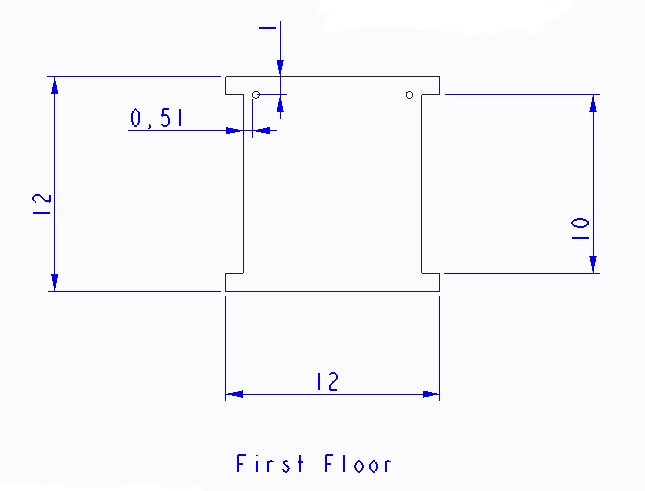


Figure . Chassis First Floor

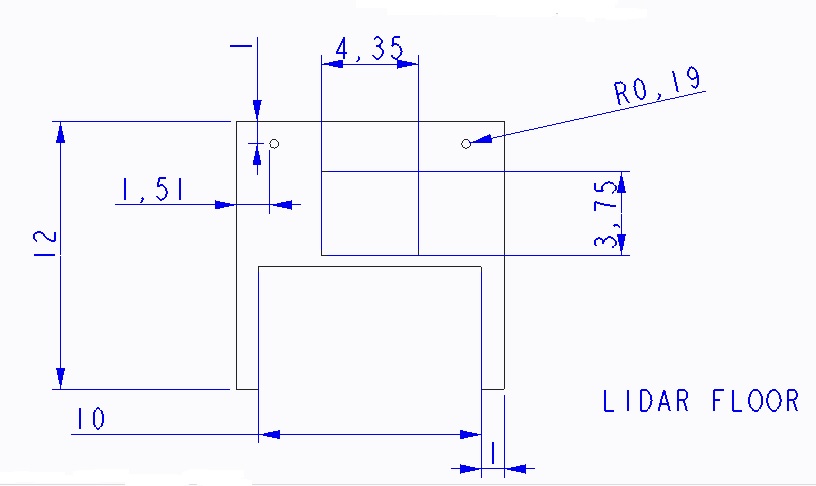


Figure . LIDAR Floor

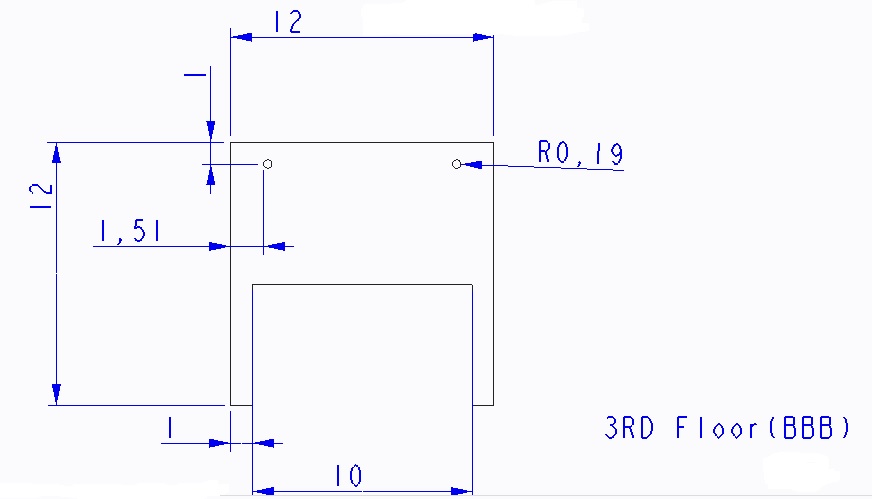


Figure . BeagleBone Black Floor

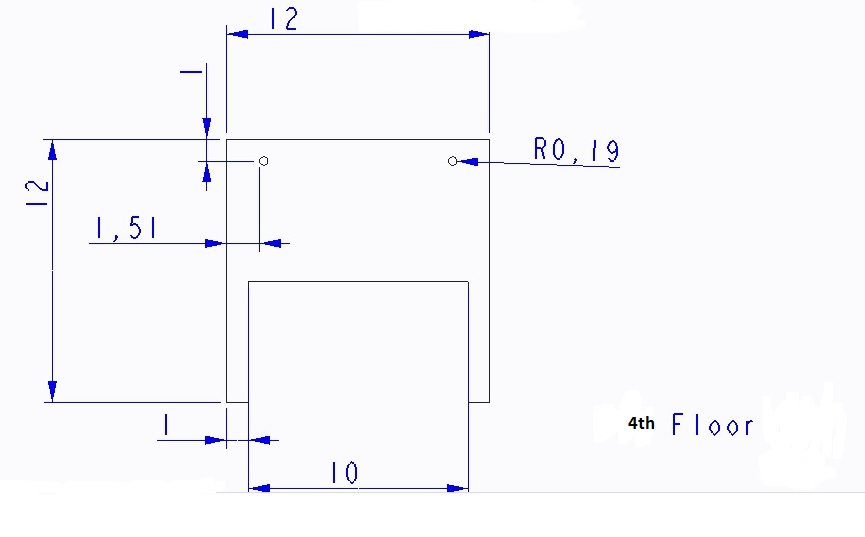


Figure . Top Floor

No CAD drawings are needed for the power supply.

### Issues

The Chassis design presented many issues. The elevator system takes up the majority of the 12x12x12 volume constraint. This left little room of all other components. In addition to this, the LIDAR demanded that it have its own dedicated floor for maximum accuracy. To solve this, the LIDAR was given a 0.7’’ high floor. This depth was possible because all but the emitter was pushed through the floor, protruding into the space that the first floor occupies. The LIDAR also required that that be as few stand offs as possible. Because of this, the passive rail that will guide the elevator up and down and also dub as the stand offs for that side of the chassis. Also, the chassis height must be low enough for the relaxed arm to fold up and yet still be under 12’’ high. This called for the chassis as well and all necessary components to be as low to the ground as possible. To aid in this endeavor, the propulsion motors will be mounted on blocks. This will essentially shift the first floor lower to the ground. This is done to exploit all the what would be wasted space between the first floor of the chassis and the ground. The blocks that the motors will be mounted on will have a height such that the bottom of the first floor is 0.5’’ from the ground.

The power supply system had many issues. Given the space requirements it was hard to find voltage regulators that did not take up too much volume. In addition to that many voltage regulators do not supply enough output current to be beneficial to this design. The wiring was also an issue with respect to the power supply circuit. Because the LIDAR must maintain as much of a 360-degree field of vision as possible, a specific solution was created. The solution was to run all sire going floor to floor either along the threaded rod standoffs or along the v-rail that aids the elevator.

### Circuit Board Layout

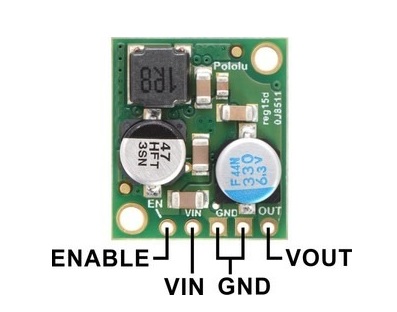


Figure . Switching Regulator

Figure 36. Switching Regulator shown the layout for the 5 volt switching regulator. The input voltage, VIN, powers the regulator. Voltages between 4.5 V and 38 V can be applied to VIN. The output voltage, VOUT, is fixed at 5 volts. The enable pin will not be used in this design; however, its function is to put the board into a low-power state.

The 12-volt regulator used in this design has little documentation posted online. As user manual is promised, however, with the purchase of the unit. This regulator can receive an Input Voltage from 4v to 32V. Its output voltage is variable and can range from 1.2v to 32V. Its default output voltage is 5V. The minimum Voltage Difference is 1V. The output current can range from 0 to 15 amps. It has terminals V-IN, V-OUT and ground. Below is the circuit board layout provided by the manufacture.



Figure . 12-volt Regulator

# Interfaces

## Delivery & Storage

Leah Watkins

The delivery and storage solution has a number of soft and hard interfaces as shown in the table below.

Table . Delivery & Storage Interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Energy | Material | Info | Description |
| Chassis | X | X |  | All Storage and elevator components will be mounted to the chassis. Wiring, power, and grounding elements will also be mounted here. |
| Power | X | X |  | Linear actuator(s) and controller(s) require power and ground. from the battery. |
| Controller/Logistics | X | X | X | The controller will provide a PWM signal and will be connected via 26AWG wire to the Linear Actuator Control Board. |
| Arm | X |  |  | Initiation for successful unloading of blocks for delivery method one is completed by the arm. Mechanical energy is transferred to the storage unit by way of upward lift on a loop to move the false-bottom in the vertical slot and remove the blocks inside . |
| Module | Energy | Material | Info | Description | |
| Chassis | X | X |  | All Storage and elevator components will be mounted to the chassis. Wiring, power, and grounding elements will also be mounted here. | |
| Power | X | X |  | Linear actuator(s) and controller(s) require power and ground. from the battery. | |
| Controller/Logistics | X | X | X | The controller will provide a PWM signal and will be connected via 26AWG wire to the Linear Actuator Control Board. | |
| Arm | X |  |  | Initiation for successful unloading of blocks for delivery method one is completed by the arm. Mechanical energy is transferred to the storage unit by way of upward lift on a loop to move the false-bottom in the vertical slot and remove the blocks inside . | |

## Cargo Retrieval

Evan Gilbert

Table . Interface for the Cargo Retrieval Module

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Energy | Material | Info | Description |
| Delivery & Storage |  | X |  | The arm actuates delivery implements native to the elevator. |
| Chassis |  | X |  | The gantry rail is affixed to a subfloor, which seats additional controllers and wiring. |
| Power | X | X |  | The motors, a stepper driver, and two limit switches require 12 V. The servo controller will need the 5 V made available from the powered USB hub. |
| Image Processing & Lighting |  | X | X | The arm moves the camera: the camera moves the arm. Shared feedback logic. |
| Microcontroller & Logistics | X |  | X | Motor controller logic is communicated to and through the master controller, which also energizes the limit switches. |

## Image Processing & Lighting

Aaron McDaniel

The image process and lighting system obtains both hard interfaces and soft interfaces. The interfaces are shown below.

Table . Image Processing & Lighting Interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Energy | Material | Info | Description |
| Arm |  | X |  | The camera and the light source will be mounted on the arm for ideal sight. |
| Power | X |  |  | Both the camera and the light source will require power from the battery. |
| Controller/Logistics |  | X | X | The camera and the light source will send information via software design to the controller/logistics. The information will be sent from the camera via USB. |

## Propulsion

Kevin Houston

Table . Propulsion Interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Energy | Material | Info | Description |
| Controller | X | X | X | The RioRand DC dual motor controller will receive PWM signals from the BeagleBone controller in order to determine how much current is sent to either motor. The microcontroller will in turn be determining the voltage and velocity of the motors |
| Chassis |  | X |  | The motors will be mounted using a specified motor mount. The mount will in turn be placed on top of a block to allow the chassis to be 0.5” from the floor of the track. The front wheels will be mounted using the same motor mounts, however axels will be placed as motor shafts to allow the front wheels to be mounted in a similar fashion as the back wheels. |
| Power | X | X |  | The battery will be connected to the RioRand dual motor controller via GND and POWER DC inputs as shown in Figure X in 3.43 |

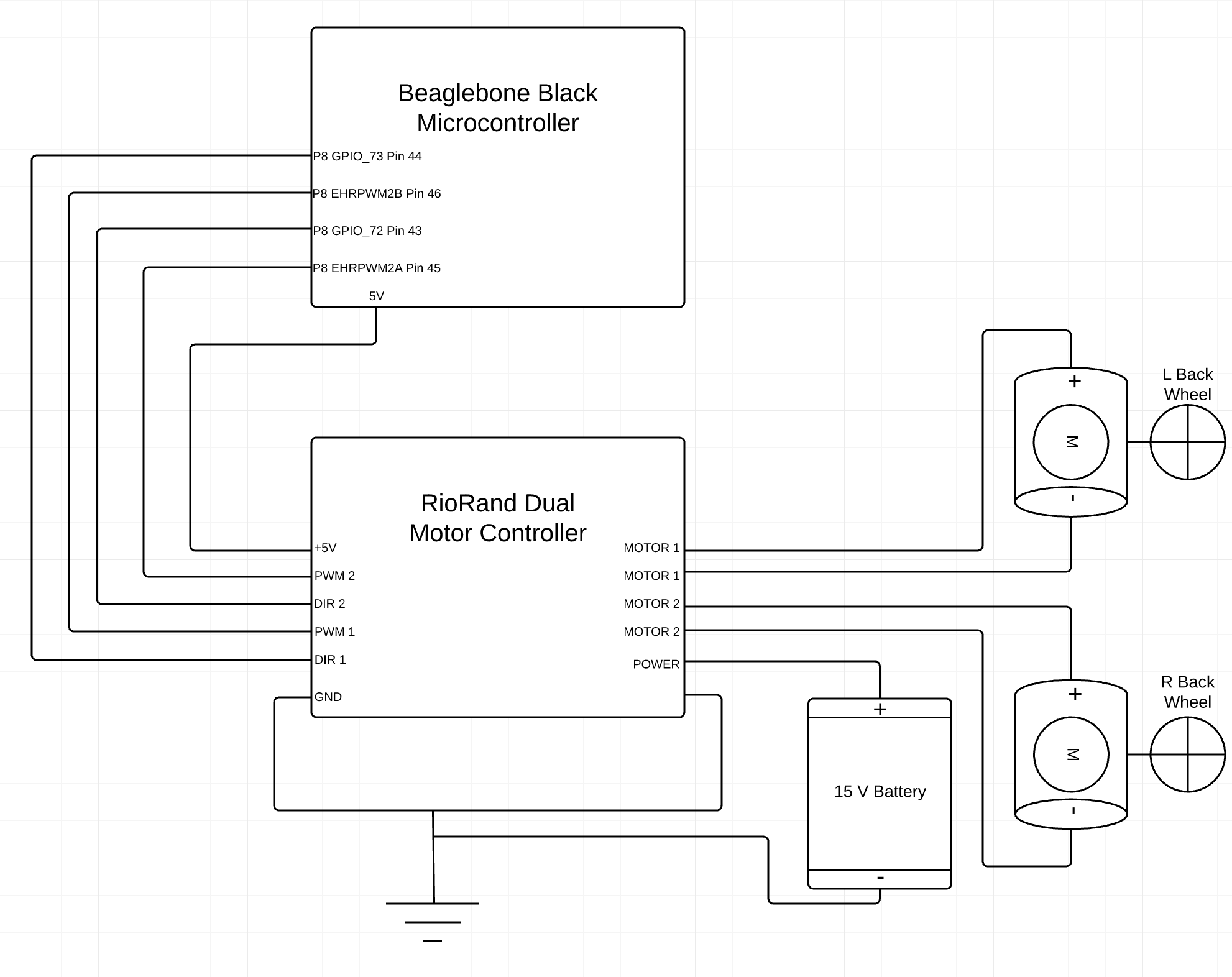


Figure . Propulsion Interface Diagram

## Navigation

Terence Staples

Table . Navigation Interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Energy | Material | Info | Description |
| Controller/Logistics | X |  | X | The LIDAR unit will receive power and communicate position to the BeagleBone Black through the USB hub. |

## Microcontroller & Logistics

Peter Corcoran

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Module | Energy | Material | Info | Description |
| Storage & Delivery |  |  | X | The controller will provide a PWM signal and will be connected through a Linear Actuator Control Board |
| Arm  USB2AX |  |  | X | The controller will be connected to the Arm through a USB2AX controller |
| Power | X | X |  |  |
| Image Processing |  |  | X | The camera and the light source will send information via software design to the controller/logistics. The information will be sent from the camera via USB. |
| Navigation  LIDAR |  |  | X | USB connections via USB hub |
| Propulsion  H-Bridge |  | X | X | Wired connections between microcontroller to h-bridge to motors. PWM Signals control voltage and velocity of motors. |

The microcontroller’s circuit diagram is used to represent the interface connections between robot subcomponents. Figure 39 shows the schematic of the BBB as it relates to other modules.

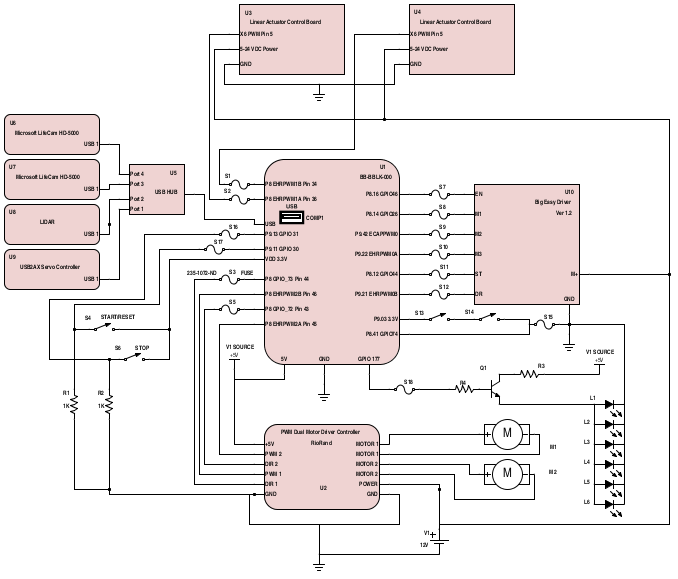


Figure . Microcontroller Hardware Schematic & Interface Diagram

## Power & Chassis

Ben Henson

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Module** | **Energy** | **Material** | **Info** | **Description** |
| Cargo Retrieval | X | X |  | The cargo retrieval system will be mounted on the top floor of the chassis. This floor entire existence is for the arm. The arm will be mounted on a gantry system which in term will be mounted on the chassis. The gantry location will be opposite that of the elevator system and in parallel with the edge of the top floor of the chassis. This module requires energy to power the 3 servo motors as well as the stepper motor. |
| Delivery & Storage | X | X |  | Both the linear actuators and their associated controller require power. Both of these actuators need to be given a space to occupy. The 1st actuator will be on the 1st floor, oriented in parallel with the surface of the floor. The second actuator will be aligned in parallel with the passive v-rails of the elevator system. The Elevator itself is given a 5.5 by 10 inches area on each floor. |
| Propulsion | X | X |  | Propulsion will require power. There is a great energy need here. The wheels will be given a 1 by 10 area on each side of the first floor in which to exist. The motors will not be mounted directly to the first floor of the chassis, but rather on a vertical block so as to lower the height of the chassis and take advantage of what would be wasted space. |
| Navigation | X | X |  | The LIDAR will require energy. It will be given its own floor. The LIDAR will have a new encasing printed that will simply slide into a pre-cut slot in the chassis surface. |
| Microcontroller & Logistics | X | X |  | The microcontroller will require energy. It is given its on floor with as specifically allocated space. This floor will also house all other controllers. |

# Schematics

## Delivery & Storage

Leah Watkins

A schematic is not needed for this module. Module details can be found in Section 6.6 Figure 41. Microcontroller System Schematic.

## Cargo Retrieval

Evan Gilbert

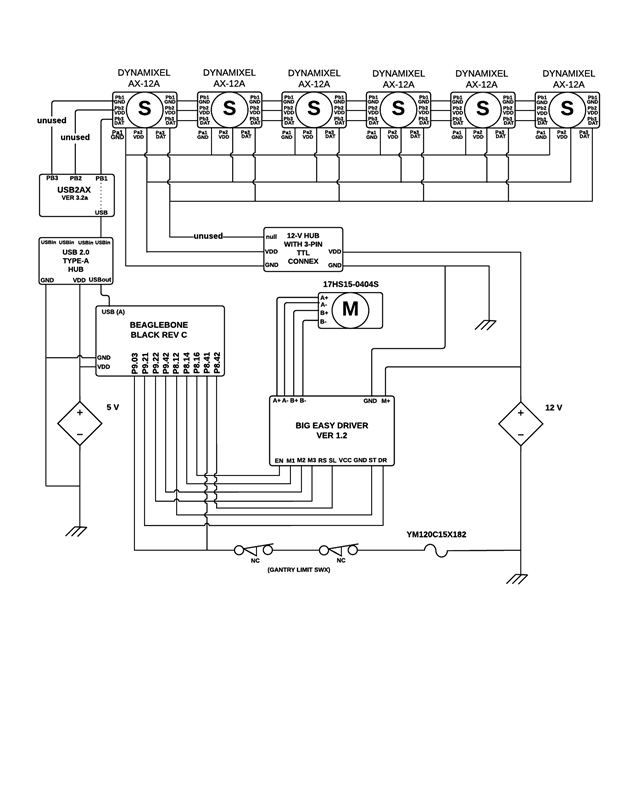


Figure . Cargo Retrieval Schematic Diagram

## Image Processing & Lighting

Aaron McDaniel

A schematic is not needed for this module. Module details can be found in Section 6.6 Figure 41. Microcontroller System Schematic.

## Propulsion

Kevin Houston

A schematic is not needed for this module. Module details can be found in Section 6.6 Figure 41. Microcontroller System Schematic.

## Navigation

Terence Staples

A schematic is not needed for this module. Module details can be found in Section 6.6 Figure 41. Microcontroller System Schematic.

## Microcontroller & Logistics

Peter Corcoran

The microcontroller system schematic is made up of the interface connections the system has to with the other sub-components of the robot system. Figure 41. Microcontroller System Schematic shows the module sub-systems that connect to it. The diagram does not cover the power distribution system, or the full cargo retrieval systems circuit.

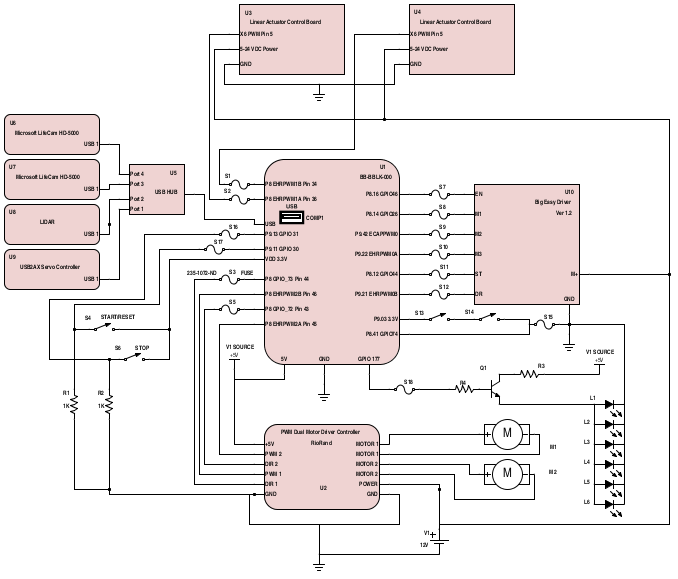
1. 

Figure . Microcontroller System Schematic

## Power & Chassis

Ben Henson

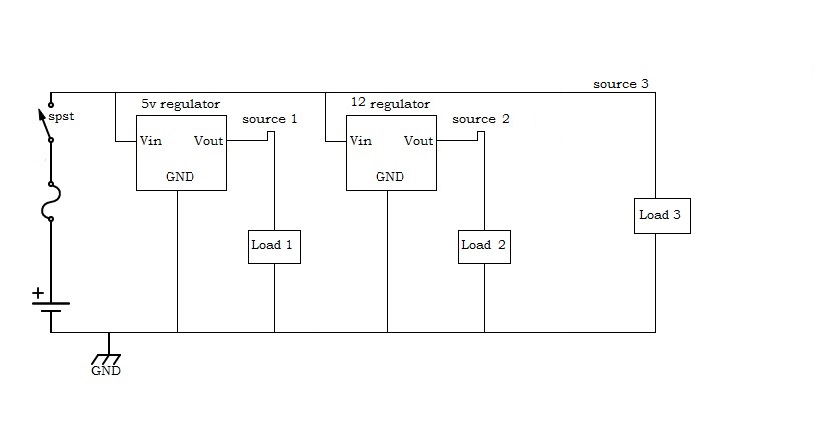


Figure . Power distribution circuit

The power distribution circuit consists of 3 voltage sources. The first is a regulated 5v source.

The 5v source or source 1 will be connected to the USB powered hub, LED array, and the BeagleBone. The USB hub will power the Logitech camera, LIDAR, and the controller arm.

Source two will be connected to the servo motors for the arm as well as its stepper motor. The third source will be the raw voltage from the battery and will be connected to the propulsion motors. It will also power the actuators for the elevator system.

# Software Design

The software being designed for the Robot is of an object oriented design and is called “*Birmingham's Logistics Actuating Zone Evaluator*” or BLAZE. The design focuses on creating an object representation of real-world objects (i.e. the robot, port, zones, etc.) This section details out the software design of each component system.

For the design to be successful multiple layers have been designed to abstract software/hardware interfacing. Meaning, the Representational Layer of the whole system sits on top of a Link Layer, which sits on top of the Physical Layer of the hardware.



Figure . BLAZE Software Layers

Hardware definitions are defined in other sections of this document, so are out scope of this section. Software Design will focus on the Link and Representational Layers.

Before any of the robot sub-systems’ software can be defined the highest hierarchical element must be defined. Deriving from the purpose of this Senior Design project, the highest object should be defined as the “Competition”, however, the first real-world object that all other objects either interact with or are contained within is the abstract port that the robot will be competing on. Therefore, a “*Port*” object will be defined that represents the game board. The “Port” object will be detailed in the “*Microcontroller & Logistics*” section but is briefly introduced here because it will be referenced in sub-system sections.

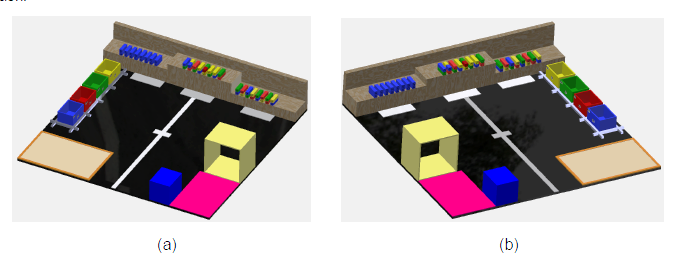


Figure . Competition boards or Ports

## Delivery & Storage

The Delivery & Storage sub-system is a hardware system that is comprised of custom hardware, and linear servos. Delivery & Storage is specific to the robot and does not have representations beyond the robot, but does have behaviors at specific locations of the Port.

### Representational Layer

#### Object: Delivery Controller



Figure . Delivery Controller Class

**Properties**

Table . Delivery Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| isEmpty | Boolean | An indicator showing if the delivery system is empty |  |
| Containers | Container collection | Object storing references to all the containers |  |
| Settings | Settings | Settings object | True |

**Behavior**

Table . Delivery Controller Behavior: getStorageColor

|  |  |
| --- | --- |
| Function | getStorageColor(int containerNumber) |
| Description | Function used to get the color of the blocks being stored in container |
| Accessibility | Public |
| Return Type | Color |
| Arguments | Int containerNumber – the number of the container that is being checked. |
|  |  |

Table . Delivery Controller Behavior: deliverContainer

|  |  |
| --- | --- |
| Function | deliverContainer(int containerNumber) |
| Description | Function used to deliver the blocks from the storage system |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Int containerNumber – the number of the container that is being delivered |
|  |  |

|  |  |
| --- | --- |
| Function | deliverAll() |
| Description | Function used to deliver all blocks at one time |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |

### Link Layer

The Delivery system runs using Firgelli Linear Actuator Control Board that uses PWM signals to control the voltage to the servos. The PWM mode allows control of the actuator using a single digital output pin from the BBB. The desired actuator position is encoded as the duty cycle of a 3.3 Volt, 1 kHz square wave on LAC connector X6 pin 5, where the percent duty cycle sets the actuator position to the same percent of full stroke extension. 100% duty cycle represents full extension, and 0% duty cycle represents full retraction. A library will be used to broker the communication to the Delivery, and can be found online at <https://github.com/derekmolloy/exploringBB/blob/master/chp06/pwm/PWM.h>

## Cargo Retrieval

The Cargo Retrieval sub-system is an electromechanical integration of custom hardware,

hardware system that is comprised of custom hardware, and servos made to function as multiple arms. Cargo Retrieval is specific to the robot and does not have representations beyond the robot, but does have behaviors at specific locations of the Port.

### Representational Layer

#### Object: CargoRetrievalController



Figure . Cargo Retrieval Class

**Properties**

Table . Cargo Retrieval Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Behaviors**

Table . Cargo Retrieval Controller Behavior: getBlock

|  |  |
| --- | --- |
| Function | getBlock(int armIndex) |
| Description | Function used to retrieve a block from a zone, and place into storage |
| Accessibility | Public |
| Return Type | Void |
| Arguments | int armIndex – the number of the arm that should pick-up the block. |
|  |  |

Table . Cargo Retrieval Controller Behavior: setPosition

|  |  |
| --- | --- |
| Function | setPosition(int armIndex, point) |
| Description | Function used set the point in space the arm extends to |
| Accessibility | Public |
| Return Type | Void |
| Arguments | int armIndex – the number of the arm that should pick-up the block. |
|  | Point – position in space |

Table . Cargo Retrieval Controller Behavior: open

|  |  |
| --- | --- |
| Function | open(int armIndex, int width) |
| Description | Function used to open the arm’s gripper |
| Accessibility | Public |
| Return Type | Void |
| Arguments | int armIndex – the number of the arm that should pick-up the block. |
|  | Int width – the width to open the gripper to |

Table . Cargo Retrieval Controller Behavior: close

|  |  |
| --- | --- |
| Function | close(int armIndex, int width) |
| Description | Function used to close the gripper of the arm |
| Accessibility | Public |
| Return Type | Void |
| Arguments | int armIndex – the number of the arm |
|  | Int width – the width to close the gripper to |

Table . Cargo Retrieval Controller Behavior: moveArm

|  |  |
| --- | --- |
| Function | moveArm(int armIndex, point) |
| Description | Function used to move arm to a specific point in space |
| Accessibility | Public |
| Return Type | Void |
| Arguments | int armIndex – the number of the arm that should pick-up the block. |
|  | Point point – the point in space to move arm to |

### Link Layer

The arm uses a set of smart servo to control movements of the arm. The servos have defined SDK for the Linux operating system that can be used with the BBB. The Link Layer communications will take advantage of these predefined APIs.

Dynamixel SDK for Linux

<http://support.robotis.com/en/software/dynamixel_sdk/usb2dynamixel/usb2dxl_linux.htm>

Dynamixel Linux gcc

<http://support.robotis.com/en/software/dynamixel_sdk/usb2dynamixel/linux/gcc.htm>

Dynamixel API Reference

<http://support.robotis.com/en/software/dynamixel_sdk/api_reference.htm>

## Image Processing & Lighting

The Image Processing & Lighting sub-system is a hardware system that is comprised of multiple cameras and LED lighting. Image Processing & Lighting is specific to the robot and does not have representations beyond the robot, but does have behaviors at specific locations of the Port.

### Representational Layer

The lighting controller and the camera controller will simply return an ON/OFF state. The image controller will receive and input of the image taken from the camera controller and return color, size, QR code, position, and the identification of the order of the rail bins.

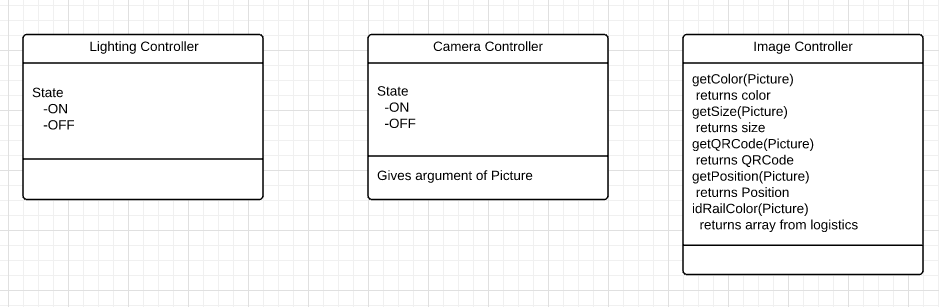


Figure . Image Processing & Lighting Class

Table . Lighting Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Required** |
| **State** | State | State that will determine the functionality of the lighting source. | True |

Table Camera Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Description** | **Required** |
| **State** | State | State that will return the camera’s functionality. | True |

**Behavior**

Table . Lighting Controller Properties: ON

|  |  |
| --- | --- |
| **Function** | **switchOn(int switch)** |
| **Description** | Boolean value to control electronic switch |
| **Accessibility** | Public |
| **Return Type** | void |
| **Arguments** | Int switch – Boolean variable (when true light is on) |

Table . Camera Controller Properties: takeImage

|  |  |
| --- | --- |
| **Function** | **takeImage(int click,image Picture)** |
| **Description** | Function used to control camera |
| **Accessibility** | Public |
| **Return Type** | image |
| **Arguments** | int click - Boolean variable (when true, takes picture)  image Picture - returns image taken |

Table . Image Controller: getColor

|  |  |
| --- | --- |
| **Function** | **getColor(image Picture, int colorType)** |
| **Description** | Function used to obtain each desired color |
| **Accessibility** | Public |
| **Return Type** | int |
| **Arguments** | int colorType- returns type of color |

Table . Image Controller: getSize

|  |  |
| --- | --- |
| **Function** | **getSize(image Picture, int size)** |
| **Description** | Function used to obtain size of block |
| **Accessibility** | Public |
| **Return Type** | int |
| **Arguments** | int size - returns size of block |

Table . Image Controller: getQR

|  |  |
| --- | --- |
| **Function** | **getQR(image Picture, int QR)** |
| **Description** | Function used to obtain QR code |
| **Accessibility** | Public |
| **Return Type** | int |
| **Arguments** | Int QR - returns color affiliated to QR code |

Table . Image Controller: getPosition

|  |  |
| --- | --- |
| **Function** | **getPosition(image Picture, int position)** |
| **Description** | Function used to obtain location in space of block |
| **Accessibility** | Public |
| **Return Type** | int |
| **Arguments** | Int position - returns (x,y) of block |

Table . Image Controller: idRailColor

|  |  |
| --- | --- |
| **Function** | **idRailColor(image Picture, int Colors)** |
| **Description** | Function used to obtain bin colors |
| **Accessibility** | Public |
| **Return Type** | array |
| **Arguments** | Int Colors - returns an array of bin colors |

### Link Layer

The images will be captured by a ffmpeg command. The command ffmpeg will communicate through putty to the BeagleBone and webcam. This ffmpeg command allows the image capture on the BeagleBone.

A library with the ffmpeg build can be found here:

<https://github.com/derekmolloy/boneCV>

## Propulsion

The propulsion sub-system is a hardware system that is comprised of motors and dual motor controller. Propulsion is specific to the robot and does not have representations beyond the robot.

### Representational Layer

#### Object: MotorController



Figure . Motor Controller Class

**Properties**

Table . Motor Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| Settings | Settings | The global settings object that contains the system’s configuration | True |

**Behavior**

Table . Motor Controller Behavior: engage

|  |  |
| --- | --- |
| Function | engage(int motorIndex, int speed) |
| Description | Function used to engage the motor at a specific velocity |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Int motorIndex – the index of the motor to engage (1 or 2 for motor 1 or two) |
|  | Int Speed – 8bit integer value representing the voltage to apply to the motor. (Max Speed in setting is a threshold value that needs to prevent values applied to motor that is higher) |

Table . Motor Controller Behavior: brake

|  |  |
| --- | --- |
| Function | brake(int motorIndex) |
| Description | Function used to slowly decrease the speed of the motor |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Int motorIndex – the index of the motor to engage (1 or 2 for motor 1 or two) |
|  |  |

Table . Motor Controller Behavior: slower

|  |  |
| --- | --- |
| Function | slower(int motorIndex) |
| Description | Function used slow motor by a factor of 10% |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Int motorIndex – the index of the motor to engage (1 or 2 for motor 1 or two) |
|  |  |

Table . Motor Controller Behavior: faster

|  |  |
| --- | --- |
| Function | faster(int motorIndex) |
| Description | Function used to speed up the motor by a factor of 10% |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Int motorIndex – the index of the motor to engage (1 or 2 for motor 1 or two) |
|  |  |

Table . Motor Controller Behavior: getSpeed

|  |  |
| --- | --- |
| Function | getSpeed(int motorIndex) |
| Description | Function used to get the current speed of the motor |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Int motorIndex – the index of the motor to engage (1 or 2 for motor 1 or two) |

### Link Layer

The Motor system runs using a dual h-bridge controller that receives PWM signals to control the voltage to the motors. The PWM signal maximum must be calculating for the maximum voltage of the motors (12V) and the battery supply (15V), which is

(1)

A library will be used to broker the communication to the motors, and can be found online at <https://github.com/derekmolloy/exploringBB/blob/master/chp06/pwm/PWM.h>

## Navigation

The Navigation sub-system is a hardware system that is comprised of a Piccolo Laser Distance Sensor retrieved from a Neato XV11. Navigation is non-specific to the robot and does have representations beyond the robot which includes a map of the Ports.

### Representational Layer

#### Object: NavigationController



Figure . Navigation Controller Class

**Properties**

Table . Navigation Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Behaviors**

Table . Navigation Controller Behavior: getCurentPosition

|  |  |
| --- | --- |
| Function | getCurrentPosition() |
| Description | Function for getting the robot’s current position |
| Accessibility | Public |
| Return Type | Point – a point object that contains the position of the robot |
| Arguments | Na |
|  |  |

Table . Navigation Controller Behavior: forward

|  |  |
| --- | --- |
| Function | Forward(speed, duration, length) |
| Description | Function for making the robot move forward in a straight line. |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Double speed – the speed at which to start moving |
|  | Double duration (optional) – the elapse time to move forward |
|  | Double length (optional) – the distance to move forward |

Table . Navigation Controller Behavior: Reverse

|  |  |
| --- | --- |
| Function | Reverse(speed, duration, length) |
| Description | Function for moving the robot in reverse in a straight line |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Double speed – the speed at which to move in reverse |
|  | Double Duration (optional) – the elapse time to move in reverse |
|  | Double length (optional) – the distance to move in reverse |

Table . Navigation Controller Behavior: turn

|  |  |
| --- | --- |
| Function | turn(targetAngle, radius, duration, dist) |
| Description | Function use to turn the robot over a distance, time or radius |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Double targetAngle – the angle the robot should be in at the end of the turn |
|  | Double radius – the radius at which the robot should turn (0 radius means move in place) |
|  | Double duration (optional) – the elapse time to turn |
|  | Double length (optional ) – the distance of move in an arc |

Table . Navigation Controller Behavior: navigateTo

|  |  |
| --- | --- |
| Function | NavigateTo(Point) |
| Description | Function used to navigate to a specific point on the board. The function will prevent navigation to an area that is already occupied with a known entity. |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Point – The point to navigate to |
|  |  |

Table . Navigation Controller Behavior: NavigateRoute

|  |  |
| --- | --- |
| Function | NavigateRoute(Route) |
| Description | Function used to navigate the robot through a known route (which is made up of multiple waypoints) |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Route – a route object that contains multiple waypoints |
|  |  |

Table . Navigation Controller Behavior: navigateThroughTunnel

|  |  |
| --- | --- |
| Function | navigateThroughTunnel |
| Description | Function that is specific to navigating the robot through the Port’s tunnel. |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |
|  |  |

### Link Layer

The LIDAR data will be obtained though the Debian terminal on the BeagleBone Black. The system is cable of receiving raw data directly from the USB port (cat dev/ttyACM0). This data can then be converted to distance data and utilized by the controller.

## Microcontroller & Logistics

The Microcontroller & Logistics sub-system is a hardware & software system that is comprised of a BeagleBone Black and power USB hub. The BBB is specific to the robot and does have representations beyond the robot, however, the Logistics system contains all non-robot object representations.

### Representational Layer

#### Object: Port

The Port object represent the competition board, and its mirror. There are two singleton Ports: PortA and PortB, representing the two competitions fields that can be presented to the robot.



Figure . Port Class Diagram

**Properties**

Table . Port Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| Floor | Floor | The Floor object for the defined port | True |
| Boundaries | Boundary | The Boundaries of the port | True |
| Type | String | The port type (A or B) | True |
| StartZone | Zone | The start Zone of the port | True |
| Tunnel | Zone | The tunnel of the particular port | True |
| ZoneA | Zone | The object representing Zone A | True |
| ZoneB | Zone | The object representing Zone B | True |
| ZoneC | Zone | The object representing Zone C | True |
| Boat | Zone | The object representing the Boat | True |
| Truck | Zone | The object representing the Truck | True |
| YellowRail | Zone | The object representing the yellow rails | True |
| BlueRail | Zone | The object representing the blue rail | True |
| GreenRail | Zone | The object representing the green rail | True |
| RedRail | Zone | The object representing the red rail | True |
| Timer | Object | An object used to keep time | True |

**Behaviors**

Table . Port Behavior: getTime

|  |  |
| --- | --- |
| Function | getTime() |
| Description | Function used to get the current time of the timer object. Used to ensure robot is able to complete course |
| Accessibility | Public |
| Return Type | Double (seconds passed since start) |
| Arguments |  |

Table . Port Behavior: determinePort

|  |  |
| --- | --- |
| Function | determinePort |
| Description | Function used to determine the port the robot is located at. Based on static points of object on the course, the robot can determine what port it is at dynamically using the Lidar system. |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Void |
|  |  |

#### Object: Floor

The Floor object defines boundaries for navigation lines that are on the floor of the Port.



Figure . Floor Class Diagram

**Properties**

Table . Floor Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| NavigationLine | Boundary | Boundary definition of the Navigation lines painted on the floor | Yes |
| ZoneAMarker | Boundary | Boundary definition of the Zone A Marker painted on the floor | Yes |
| ZoneBMarker | Boundary | Boundary definition of the Zone B Marker painted on the floor | Yes |
| ZoneCMarker | Boundary | Boundary definition of the Zone C Marker painted on the floor | Yes |

**Behaviors**

The floor does not have any behaviors defined.

#### Object: Boundary

A boundary is a set of coordinates that define a location’s outer perimeter.



Figure . Boundary Class Diagram

**Properties**

Table . Boundary Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| Points | Collection<Points> | A collection of points that defined the outer perimeter of the boundary | True |
| IsStatic | Boolean | Indicates the boundary is stationary or is moveable | True |

**Behavior**

|  |  |
| --- | --- |
| Function | Parse(string) |
| Description | Function used to parse a comma delimited string value in the form of “x1,y1,z1,x2,y2,z2….xn,yn,zn” |
| Accessibility | Public |
| Return Type | Boundary |
| Arguments | String |

#### Object: Point



Figure . Point Class Diagram

**Properties**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| xval | Double | X position of the point in reference to Port’s origin |  |
| yval | Double | Y position of the point in reference to Port’s origin |  |
| zval | Double | Z position of the point in reference to Port’s origin |  |
| theta | Double | Theta angle of the point in reference to Port’s origin |  |

**Behaviors**

Table . Point Behavior: x

|  |  |
| --- | --- |
| Function | x() |
| Description | Function used to return the x value of a point |
| Accessibility | Public |
| Return Type | Double |
| Arguments |  |

Table . Point Behavior: y

|  |  |
| --- | --- |
| Function | y() |
| Description | Function used to return the y value of a point |
| Accessibility | Public |
| Return Type | Double |
| Arguments |  |

Table . Point Behavior z

|  |  |
| --- | --- |
| Function | z() |
| Description | Function used to return the z value of a point |
| Accessibility | Public |
| Return Type | Double |
| Arguments |  |

Table . Point Behavior: theta

|  |  |
| --- | --- |
| Function | theta() |
| Description | Function used to retrieve theta angle of the point |
| Accessibility | Public |
| Return Type | Double |
| Arguments |  |

Table . Point Behavior: dist

|  |  |
| --- | --- |
| Function | dist() |
| Description | Function to calculate distance between two points |
| Accessibility | Public |
| Return Type | Double |
| Arguments | Point |

Table . Point Behavior: add

|  |  |
| --- | --- |
| Function | add() |
| Description | Function used to add two points together |
| Accessibility | Public |
| Return Type | Point |
| Arguments | Point b |

Table . Point Behavior: sub

|  |  |
| --- | --- |
| Function | sub() |
| Description | Function used to subtract two point |
| Accessibility | Public |
| Return Type | Point |
| Arguments | Point b |

Table . Point Behavior: move

|  |  |
| --- | --- |
| Function | move() |
| Description | Function is used to move a point’s x & y position |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Double a – added to x value |
|  | Double b – added to y value |
|  | Double c – added to z value |
|  | Double d – added to the theta value |

#### Object: Zone



Figure . Zone Class Diagram

**Properties**

Table . Zone Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| Boundary | Boundary | The defined boundary of the zone | True |
| isDeliveryZone | Bool | Indicator to show if the instance is a delivery zone | True |
| TargetColor | Color | Indicator for the color block the zone contains or should contain | False |
| Inventory | Collection<Blocks> | Collection of Blocks that the zone contains | True |
| QRCode | Qrcode | A qrcode object that identifies the zone | false |
|  |  |  |  |

**Behavior**

Zones do not have any behavior, only state

#### Object: RobotController



Figure . Robot Controller Class Diagram

**Properties**

Table . Robot Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| isSystemReady | Boolean | Indicator that shows system ready status | True |
| isHardwareReady | Boolean | Indicator that shows all hardware is ready | True |
| isSensorsReady | Boolean | Indicator that shows all sensors are ready | True |
| isReady | Boolean | Indicator that shows that robot is ready to begin | True |
| LogisticsControl | LogisticsController | Singleton that controls the logistics of the robot | True |
| DeliveryControl | DeliveryController | Singleton that controls the storage & delivery object | True |
| CargoRetrievalControl | CargoRetrievalController | Singleton that controls the arm | True |
| NavigationControl | NavigationController | Singleton that controls the robot Navigation | True |
| MotorControl | MotorController | Singleton that controls the motors | True |

**Behavior**

Table . Robot Control Entry Function

|  |  |
| --- | --- |
| Function | entry(void \*args) |
| Description | Function acts as an entry point into the Robot Control system. Starts as a new system thread. |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Void \*args |

Table . Robot Control Start Function

|  |  |
| --- | --- |
| Function | Start() |
| Description | Function starts the main robot control loop |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |

Table . Robot Control Initialize Settings Function

|  |  |
| --- | --- |
| Function | initializeSettings(file) |
| Description | Function initializes all system settings and makes them ready to communicate to sub-component controllers |
| Accessibility | Public |
| Return Type | Settings |
| Arguments | File – file name of the settings file to run (system can have multiple settings files. One file can be specified at robot startup.) |

Table . Robot Control System Check Function

|  |  |
| --- | --- |
| Function | systemCheck() |
| Description | Function executes a full system check to ensure everything is ready to operate. |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |

Table . Robot Control System Behavior: hardwareCheck

|  |  |
| --- | --- |
| Function | hardwareCheck() |
| Description | Function checks hardware components to ensure ready state. |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |

Table . Robot Control System Behavior: sensorCheck

|  |  |
| --- | --- |
| Function | sensorCheck(sensor) |
| Description | Function checks each sensor to ensure the sensor is operational |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Sensor – object defines a sensor |

Table . Robot Control System Behavior: readyStateCheck

|  |  |
| --- | --- |
| Function | readyStateCheck() |
| Description | Function checks the “startFrom” variable to determine robot start route.   * If starting from the start, scans toggle switch state that indicate competition start, after start condition is met, the NavigationControl.navigateThroughTunnel() function executes. * If the starting from a pickup/delivery route, the robot will follow the |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |

Table . Robot Control System Behavior: startFromRoute

|  |  |
| --- | --- |
| Function | startFromRoute(Route) |
| Description | Function is used to start the robot from a named route, rather than the start each time. |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Route – Name of the route to start from. Route names are in the settings file. |

#### Object: LogisticsController



Figure . Logistics Controller Class Diagram

**Properties**

Table . Logistics Controller Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| Routes | Route collection | List of routes that the robot uses to solve the competition | True |
| CurrentInventory | Blocks collection | Collection of blocks that represent what the robot is currently carrying in storage | True |
| CurrentRoute | Route | The current route being navigated by the navigation system | NA |
| isZoneADelivered | Boolean | An indicator value that tracks if Zone A has been delivered or not | True |
| isZoneBDelivered | Boolean | And indicator value that tracks if Zone B has been delivered or not | True |
| isZoneCDelivered | Boolean | An indicator value that tracks if Zone C has been delivered or not | True |
| Settings | Settings | An object that contains all system settings | True |

**Behavior**

Table . Logistics Controller Behavior: startLogistics

|  |  |
| --- | --- |
| Function | startLogistics |
| Description | Function used to start the logistics controller. |
| Accessibility | Public |
| Return Type | Void |
| Arguments |  |

Table . Logistics Controller Behavior: takeInventory

|  |  |
| --- | --- |
| Function | takeInventory(Zone) |
| Description | Function used to take the inventory of a zone |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Zone – The zone to take inventory of |

Table . Logistics Controller Behavior: loadInventory

|  |  |
| --- | --- |
| Function | loadInventory(Zone) |
| Description | Function used to take inventory of a zone |
| Accessibility | Public |
| Return Type | Void |
| Arguments | Zone – The zone to load |

Table . Logistics Controller Behavior: deliverInventory

|  |  |
| --- | --- |
| Function | deliverInventory(targetZone) |
| Description | Function used deliver Inventory to a specific zone. |
| Accessibility | Public |
| Return Type | Void |
| Arguments | targetZone – the zone being delivered to (boat, truck, yellow rail, blue rail, green rail, red rail) |

#### Object: Route



Figure . Route Controller Class Diagram

**Properties**

Table . Route Properties

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
| SourceZone | Zone | The zone the robot is “starting” from | False |
| TargetZone | Zone | The target zone the robot is traveling to.   * If not populated the robot will stop at the last waypoint specified in the Waypoints property * If populated the robot will navigate to the target zone | False |
| Waypoints | Point Collection | A collection of point that are on the route.   * If move type is a straight line, the robot will move in a straight line from one point to the next. * If the move type is an arc, the robot will move from one point to the next arcing around the origin at the specified radius * If the move type is a reverse-arc, the robot will move from one point to the next arcing away from the origin at the specific radius |  |
| RouteType | String | The type of route that this defines, navigation, load, delivery, etc | True |
| MoveType | String | The type of move that should be executed, straight line, arc, reverse-arc, etc | True |
|  |  |  |  |

**Behaviors**

#### Object:

**Properties**

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Description | Required |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Behaviors**

### Link Layer

### Settings

BLAZE is a settings controlled system, meaning the settings file used to run the primary blaze application defines the state, behavior, and configuration of BLAZE. The purpose of this approach is to make “tweaking” and configuring the robot easier.

Table . Robot Settings

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Section | Description | Default Value |
| VERSION | System | System Version |  |
| NAME | System | Robot Name | BLAZE |
| CONTROL\_MODE | System | AUTO is fully automated mode  MANUAL is remote control mode. |  |
| START\_MODE | System | START - start Blaze at starting line wait for light  TUNNEL - start Blaze in navigate tunnel  LOAD - start Blaze in loading mode  DELIVERY - starts Blaze in delivery mode  ZONEAL - start Blaze in Zone A Load mode  ZONEBL - start Blaze in Zone B Load mode  ZONECL - start Blaze in Zone C Load mode  BOATD - start Blaze in Boat Delivery mode  YRAILD - start Blaze in Yellow Rail Delivery mode  BRAILD - start Blaze in Blue Rail Delivery mode  GRAILD - start Blaze in Green Rail Delivery mode  RRAILD - start Blaze in Red Rail Delivery mode  TRUCKA - start Blaze in Truck Delivery mode  TRUCKB - start Blaze in Zone C Boat Delivery mode | START |
| CONSOLE\_ENABLED | LOGGING | Enables or disables robot console output | True |
| DEBUG\_ENABLED | LOGGING | Enables or disables robot debug | True |
| DEBUG\_LEVEL | LOGGING | Current Debug Level:  "ERROR",  "WARNING",  "INFO",  "DEBUG",  "DEBUG1",  "DEBUG2",  "DEBUG3",  "DEBUG4" | Debug |
| LOGFILE\_ENABLED | LOGGING |  | True |
| LOGFILE\_NAME | LOGGING | name of the log file to save data to | Blaze\_error.log |
| APPEND\_TO\_LOG | LOGGING | specifies whether to append to the log (true) or overwrite (false) | true |
| robot\_x | ROBOT | Robot width | 12 |
| robot\_y | ROBOT | Robot length | 12 |
| robot\_z | ROBOT | Robot initial height | 12 |
| WP\_TUNNEL\_EXIT | Waypoints | The target point the robot will navigate to in order to exit tunnel | TBD |
| ZONEA\_INVENTORY | Waypoints | The target point the robot will navigate to start taking inventory of Zone A | TBD |
| ZONEB\_INVENTORY | Waypoints | The target point the robot will navigate to start taking inventory of Zone B | TBD |
| ZONEC\_INVENTORY | Waypoints | The target point the robot will navigate to start taking inventory of Zone C | TBD |
| ZONEA\_LOAD\_START | Waypoints | The target point the robot will navigate to start loading Zone A | TBD |
| ZONEB\_LOAD\_START | Waypoints | The target point the robot will navigate to start loading Zone B | TBD |
| ZONEC\_LOAD\_START | Waypoints | The target point the robot will navigate to start loading Zone C | TBD |
| TRUCK\_DELIVERY\_START | Waypoints | The target point the robot will navigate to start delivery of the truck | TBD |
| BOAT\_DELIVERY\_START | Waypoints | The target point the robot will navigate to start delivery of the boat | TBD |
| YRAIL\_DELIVERY\_START | Waypoints | The target point the robot will navigate to start delivery of the YELLOW Rail Zone, updated by QRCode Scan | Dynamic |
| BRAIL\_DELIVERY\_START | Waypoints | The target point the robot will navigate to start delivery of the BLUE Rail Zone, updated by QRCode Scan | Dynamic |
| GRAIL\_DELIVERY\_START | Waypoints | The target point the robot will navigate to start delivery of the GREEN Rail Zone, updated by QRCode Scan | Dynamic |
| RRAIL\_DELIVERY\_START | Waypoints | The target point the robot will havigate to start delivery of the RED Rail Zone, updated by QRCode Scan | Dynamic |
| PORT\_SCALE | Port | Size of port grid in inches | 1 |
| PORTA\_ZONE\_ORDER | Port | Order of Port A Zone resolution | A,B,C |
| PORTB\_ZONE\_ORDER | Port | Order of Port B Zone resolution | C,B,A |
| PA\_START\_BOUND | Port | Set of points (x, y, z) that define the boundary of the item the first point is the closest to the origin, then moving in a clockwise fashion until closed | 76,0,0  ,76,30,0  ,96,30,0  ,96,0,0 |
| PA\_ZONEA\_BOUND | Port | Port A Zone A Boundary |  |
| PA\_ZONEB\_BOUND | Port | Port A Zone B Boundary |  |
| PA\_ZONEC\_BOUND | Port | Port A Zone C Boundary |  |
| PA\_BOAT\_BOUND | Port | Port A Boat Boundary |  |
| PA\_TRUCK\_BOUND | Port | Port A Truck Boundary |  |
| PA\_TRUCKOPENING\_BOUND | Port | Port A Truck Opening Boundary |  |
| PA\_RAIL1\_TOP\_BOUND | Port | Port A Rail 1 (closest to barge) Top boundary |  |
| PA\_RAIL2\_TOP\_BOUND | Port | Port A Rail 2 (2nd away from barge) Top boundary |  |
| PA\_RAIL3\_TOP\_BOUND | Port | Port A Rail 3 (3rd away from barge) top boundary |  |
| PA\_RAIL4\_TOP\_BOUND | Port | Port A Rail 4 top boundary |  |
| PA\_TUNNEL\_ENTRANCE | Port | Port A Tunnel Entrance boundary |  |
| PA\_TUNNEL\_EXIT | Port | Port A Tunnel Exit boundary |  |
| PB\_START\_BOUND | Port | See above |  |
| PB\_ZONEA\_BOUND | Port | Port B Zone A Boundary |  |
| PB\_ZONEB\_BOUND | Port | Port B Zone B Boundary |  |
| PB\_ZONEC\_BOUND | Port | Port A Zone C Boundary |  |
| PB\_BOAT\_BOUND | Port | Port B Boat Boundary |  |
| PB\_TRUCK\_BOUND | Port | Port B Truck Boundary |  |
| PB\_TRUCKOPENING\_BOUND | Port | Port B Truck Opening Boundary |  |
| PB\_RAIL1\_TOP\_BOUND | Port | Port B Rail 1 (closest to barge) Top boundary |  |
| PB\_RAIL2\_TOP\_BOUND | Port | Port B Rail 2 (2nd away from barge) Top boundary |  |
| PB\_RAIL3\_TOP\_BOUND | Port | Port B Rail 3 (3rd away from barge) top boundary |  |
| PB\_RAIL4\_TOP\_BOUND | Port | Port B Rail 4 top boundary |  |
| PB\_TUNNEL\_ENTRANCE | Port | Port B Tunnel Entrance boundary |  |
| PB\_TUNNEL\_EXIT | Port | Port b Tunnel Exit boundary |  |
|  |  |  |  |
| arm\_enabled | Arm | Enable/Di See above sable the hardware | True |
| Motors\_enabled | Motors | Enable/Disable the hardware | True |
| m1\_max\_speed | Motors | The maximum 8 bit integer representation of maximum voltage allowed to motor | 255 |
| m2\_max\_speed | Motors | The maximum 8 bit integer representation of maximum voltage allowed to motor | 255 |
| Camera\_enabled | Cameras | Enable/Disable the hardware | True |
| Lidar\_enabled | LIDAR | Enable/Disable the hardware | True |

## Power & Chassis

The Power & Chassis sub-system is a hardware system that is comprised of batteries, and the material of the robot. As such, the Power & Chassis is specific to the robot and does not have representations beyond the robot.

### Representational Layer

No software representation is needed for the Power and Chassis sub-system.

### Link Layer

No software link layer is needed for the Power and Chassis sub-system.

# Weight Estimates

## Summary

|  |  |
| --- | --- |
| **Module** | **Estimated Weight** |
| Delivery/Storage - Leah Watkins | 3.2 lbs. |
| Cargo Retrieval - Evan Gilbert | 4.75 lbs. |
| Image Processing & Lighting - Aaron McDaniel | 0.08 lbs |
| Propulsion - Kevin Houston | 2.2 lbs |
| Navigation - Terence Staples | 0.44 lbs |
| Microcontroller - Peter Corcoran  Logistics - Peter Corcoran | 0.6 lbs |
| Chassis - Ben Henson  Power - Ben Henson | [4.47(chassis)+2.15(power)]lbs. |
| **TOTAL** | **17.5 lbs** |

## Delivery & Storage

Table . Delivery & Storage Weight Table

|  |  |  |  |
| --- | --- | --- | --- |
| QTY | Part | W, g | Total W, g |
| 2 | Linear Actuator | 74 | 148 |
| 2 | Linear Actuator Controller | 19.5 | 39 |
| 6 | Kydex Sheets | 208 | 1248 |

## Cargo Retrieval

Evan Gilbert

Table . Weight Budget

|  |  |  |  |
| --- | --- | --- | --- |
| QTY | Part | W, g | Total W, g |
| 1 | Thrust Needle Bearing | 21 | 21 |
| 3 | Arm Skate BB | 8.5 | 25.5 |
| 6 | Servo Motor | 54.5 | 327 |
| 1 | Stepper Motor | 290 | 290 |
| 1 | Gantry Plate | 27.6 | 27.6 |
| 1 | Motor Mount | 20.87 | 20.87 |
| 3 | Sm Linkage | 10 | 30 |
| 2 | Lg Linkage | 19 | 38 |
| 1 | Camera Mount | 15 | 15 |
| 1 | Shoulder Housing (BTM) | 45 | 45 |
| 1 | Shoulder Housing (Top) | 20 | 20 |
| 1 | End Effector | 200 | 200 |
| 1 | Wooden Block | 150 | 150 |
| 1 | Camera | 200 | 200 |
| — | Cables | 150 | 150 |
| — | Hardware | 300 | 300 |
| 2 | Limit Switch | 6.4 | 12.8 |
| 1 | Timing Pulley | 8.68 | 8.68 |
| 1 | Timing Belt | 6 | 6 |
| 1 | ~12” V-Rail | 135 | 135 |
| 1 | Motor Controller | 6.52 | 6.52 |
| 1 | Motor Controller Mount | 15 | 15 |
| 1 | Servo Controller | 4.97 | 4.97 |
| 16 | Wheel Skate BB | 4.26 | 68.16 |
| 8 | Delrin Wheel | 2.39 | 19.12 |
|  |  |  | 2136.22 |

Note: The cable and hardware weights were inflated to overcompensate for any minor future additions.

=> A conservative approximation of the subsystem weight currently stands at 2136.22g→ 2150g ≅ 4.74lb → 4.75lb­­

A separate gantry weight budget was used to determine the stall torque required of the bipolar stepper motor driving the belt-and-pinion gantry. Weights of the items constituting the stepper motor payload were inflated as a means of derating the motor torque in order to create an operational torque differential, which did include additional conservative measures. A FBD analysis was conducted wherein the gantry wheels were ignored, and the toothed timing belt was replaced with a flat one. From the torque differential, a favorable weight class and power specifications could be selected. By choosing the lightest power appropriate motor capable of delivering the desired range of torque, the gantry’s shifting center of mass becomes less problematic.

Table . Gantry Weight Budget Used to Select Stepper Motor Torque and Weight

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| QTY | Part | W, g | Derating by W @ 15% inflation, g | Derated Total W, g |
| 1 | Thrust Needle Bearing | 21 | 24.5 | 24.5 |
| 3 | Skate BB | 8.5 | 9.775 | 29.325 |
| 6 | Servo Motor | 54.5 | 62.675 | 376.05 |
| 1 | Stepper Motor | 350 | 402.5 | 402.5 |
| 1 | Gantry Plate | 27.6 | 31.74 | 31.74 |
| 1 | Motor Mount | 20.87 | 24.0005 | 24.0005 |
| 2 | Sm Linkage | 10 | 11.5 | 23 |
| 3 | Lg Linkage | 19 | 21.85 | 65.55 |
| 1 | Camera Mount | 15 | 17.25 | 17.25 |
| 1 | Shoulder Housing (BTM) | 45 | 51.75 | 51.75 |
| 1 | Shoulder Housing (Top) | 20 | 23 | 23 |
| 1 | End Effector | 200 | 230 | 230 |
| 1 | Wooden Block | 150 | 172.5 | 172.5 |
| 1 | Camera | 200 | 230 | 230 |
| — | Servo Cables | 30 | 34.5 | 34.5 |
|  |  |  |  | 1735.3155 |

Note: The approximated camera weight includes the supported weight of the attached USB cable. Also, less than 100g of hardware was excluded from the FBD analysis, yet proved negligible, nonetheless.

=> 1735.3155g → 1740g ≅ 61.37669oz

Proposed Solution: NEMA 17 Bipolar Stepper Motor

→ 12V, 0.4 AMAX, 56.07 oz.in

→ weighs 290g

Adjusting for the 290-g motor => derated weight becomes 1666.3155g → 1670oz ≅ 58.908oz → 59oz

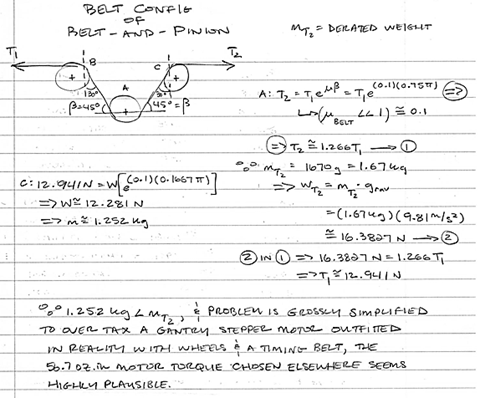


Figure . Belt Configuration

## Image Processing & Lighting

Aaron McDaniel

No additional information needed.

## Propulsion

Kevin Houston

Table . Propulsion Weight Table

|  |  |  |  |
| --- | --- | --- | --- |
| QTY | Part | W, g | Total W, g |
| 2 | 47:1 Pololu Brushed DCMetal Gearmotor | 88 | 176 |
| 4 | Pololu Universal Aluminum 4mm Mounting Hub | 3.2 | 12.8 |
| 4 | Pololu Wheel 80x10mm | 19.84 | 79 |
| 4 | Pololu 25D mm Metal Gearmotor Bracket | 8.5 | 34 |
| 4 | Aluminum Sprockets (0.250) 32T | 19.8 | 79.2 |
| 1 | #25 Single Strand-Riveted Roller Chain, 10 feet (5 feet) | 230 | 230 |
| 1 | RioRand DC Dual Motor controller | 8.5 | 8.3 |
| 2 | Wooden Block | 150 | 300 |
| 2 | Locknut | 17 | 34 |
| 4 | Kavan Wheel Collar | 10 | 40 |
|  |  |  |  |

## Navigation

Terence Staples

Table . Navigation Weight Table

|  |  |  |  |
| --- | --- | --- | --- |
| QTY | Part | W, g | Total W, g |
| 1 | LIDAR unit | 200 | 200 |

## Microcontroller & Logistics

Peter Corcoran

Table . Microcontroller Weight Table

|  |  |  |  |
| --- | --- | --- | --- |
| QTY | Part | W, g | Total W, g |
| 1 | BBB unit | 12g | 12g |
| 1 | Proto-board & components | 36g | 36g |

## Power & Chassis

Ben Henson

|  |  |  |  |
| --- | --- | --- | --- |
| QTY | Part | W, lbs. | Total W, lbs. |
| 1 | Seaboard floor 3 | 1 | 1 |
| 1 | Seaboard floor 3 | 1 | 1 |
| 1 | Seaboard floor 2 | 0.8 | 0.8 |
| 1 | Seaboard floor 1 | 1.37 | 1.37 |
| 2 | Threaded rod(⅜’’) | 0.15 | 0.3 |
| 1 | 12 volt switching regulator | 0.2 | 0.2 |
| 1 | Battery | 1.8 | 1.8 |
| 3 | Terminal Block | 0.05 | 0.15 |
|  |  |  | Total = 6.62 |

# Energy Estimates

|  |  |  |
| --- | --- | --- |
| Component | Operating Voltage | Operating current |
| Arm servo motor 1 | 12v | 900mA |
| Arm servo motor 2 | 12v | 900mA |
| Arm servo motor 3 | 12v | 900mA |
| Arm servo motor 4 | 12v | 900mA |
| Arm servo motor 5 | 12v | 900mA |
| Arm servo motor 6 | 12v | 900mA |
| Logitech c930e | 5v | 500mA |
| Propulsion motor 1 | 12v | 1.37A |
| Propulsion motor 1 | 12v | 1.37A |
| BBB | 5v | 1A |
| LIDAR | 5v | 135mA |
| LED Array | 5v | 0.36A |
| Stepper motor | 12v | 0.4 A |
| USB powered hub | 5v | 2.6A |
| Arm servo controller | 5v | 100mA |
| Stepper motor controller | 12v | 0.4A |
| Linear Servo Controller 1 | 5v-24v | 4A |
| Linear Servo Controller 2 | 5v-24v | 4A |

# Possible Problems & Solutions

This section…

## Delivery & Storage

Leah Watkins

Table . Delivery & Storage Problem/Solutions

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| 1. Losing blocks during transit | Navigation, propulsion and logistics will work together to determine the appropriate speed that the AR must travel to avoid losing blocks during transit. |
| 2. Malfunction of linear actuator | Care will be taken to ensure input values stay within the acceptable range of the part to increase lifespan. |
| 3. Malfunction of mechanical solution for unloading using the arm | The delivery/storage solution will be made of durable materials which will withstand the wear and tear of the loading and unloading of blocks. |
| 4. Space | Each module will make efficient use of space in order to accommodate each solutions as it is designed. |
| 5. LIDAR field of view | The elevator will ride high through a majority of the course in order to ensure the LIDAR can utilize 360 degree field of view. |

## Cargo Retrieval

Evan Gilbert

Table . Cargo Retrieval Problem/Solutions

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| 1. Dead actuator! | A contingency set of inverse kinematics will be used to include possible actuator failure based gestures as a means of compensating for the lost functionality. |
| 2. False positive motor control relay from Image Processing & Lighting module | A lumen reader can be used to determine if certain arm gestures contribute to image processing washout due to abrupt changes in intensity/luminosity. Additionally, the USB2AX servo controller can be used to compare a hanging image processing value against a positional reality check. Essentially, the arm could return to a previous gestural state (like a physical restore point) so that the image processing module can reevaluate the environment. A block-in-hand scenario involves careful backtracking, as nothing would be gained were the arm to prematurely drop its payload as a procedural means of assessing where to drop its payload. The end effector servo must, therefore, be accounted for during the routine outlined above. |
| 3. Unstable/shifting center of mass | In order to avoid adding foreign ballast to the chassis, the heavier components that don’t require special heat shunting placement can be grouped closest to the elevator shaft and beneath the gantry. It is expected that operational gantry speeds will redefine this problem during the implementation phase. |
| 4. Operational speed of arm-to-elevator delivery system actuation leads to block spillage | Should a screen damper/retainer not be used in combination with the elevator states, then the servo speeds will need to be throttled until a routine of smooth, consistent delivery actuation is established. |
| 5. No confirmation of delivery upon actuation of delivery system | The arm can lift the camera such that it can peek inside the elevator compartment and communicate some logical status with the master controller. |

## Image Processing & Lighting

Aaron McDaniel

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| 1. not obtaining correct brightness | Calculate correct output lumens |
| 2. Malfunction of image controller algorithm | Run several tests and debug image processing algorithm. |
| 3. Malfunction of lighting controller algorithm | Run several tests and debug image processing algorithm. |
| 4. Malfunction of camera controller algorithm | Run several tests and debug image processing algorithm. |

## Propulsion

Kevin Houston

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| 1. Robot not able to travel through the tunnel | Recalculate necessary torque and verify the amount of current being sent to each motor/Consider larger wheel diameter |
| 2. Wheels not gaining enough traction | Consider adding high-traction sticky tire coats over back wheels to increase tread. |
| 3. Front wheels not rotating fast enough. | Make sure that the front axles have enough space for the front wheels to rotate around efficiently. |
| 4. Chain between the front and back sprockets falling off track. | Verify that the necessary amount of chain links is used. Consider adding more/less chain links. |

## Navigation

Terence Staples

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| 1. LIDAR not able to connect to BeagleBone through the USB hub | Utilized pin connections on BBB and power the motor externally |
| 2. LIDAR algorithm utilized provides data not consistent with what is being seen on the playing board | Modify code and test until the desired results are obtained |
| 3. Block after placement distort the position reading (x,y) | Code can be modified to mitigate this problem |

## Microcontroller & Logistics

Peter Corcoran

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| Libraries used in Link Layers may not have need functions to make system function | Libraries will need to be tested to ensure functionality will be appropriate for use |
| USB device communication | USB device communication with devices will need to be tested to ensure the systems can talk to each other without collision. |
| LIDAR communication & Link Layer libraries make it difficult to develop concrete solution. | LIDAR communication & Link Layer libraries will need to be identified or custom code needs to be written to take advantage of the technology. |

## Power & Chassis

Ben Henson

|  |  |
| --- | --- |
| **Problem** | **Solution** |
| Voltage regulators may not be able to provide enough current to meet load requirements. | Select regulators with greater current output than necessary so as to avoid this problem. |
| The chassis may tip if it runs over a dropped block | Create the chassis so that it rides very low to the ground. This was a block would simply be pushed out of the way instead of run over. |
| The chassis may be so low to the ground it cannot make it through the tunnel | Let the tunnel lip height decide how high the chassis should be from the ground. |

# Implementation Plan

This section…

## Delivery & Storage

Leah Watkins

Efficient implementation of subsystems is key to ensuring any system is completed in a timely manner. The Delivery & Storage solution lends itself to modularity so its implementation will require few steps to complete. The general steps describing build and implementation are described below.

1. Build Storage “taco” using lightweight durable Kydex material
   1. Measure and cut Kydex panels
   2. Rivet pieces to shape
2. Delivery method one
   1. Measure and cut false-bottom
   2. Attach cord and loop for delivery via arm
   3. Install onto “taco” vertical storage unit
3. Test delivery method number one
4. Build rake mechanism for delivery method number two using lightweight durable Kydex material
   1. Measure and cut Kydex panels
   2. Rivet pieces to shape
   3. Install hardware for linear actuator stroke to nest at back of the rake mechanism
5. Install v-rail system for elevator onto chassis
6. Test linear actuator(s) and controller(s) (uninstalled)
7. Install linear actuator(s) and controller(s) including wiring and mounting hardware
8. Test linear actuator(s) and controller(s) (installed)
9. Test overall solution including delivery methods one and two

## Cargo Retrieval

Evan Gilbert

Week of 11-2-2015

1. Wear eye protection while prepping the v-rail aluminum extrusion.

2. Test v-rail to determine if section is square.

3. Cut v-rail to desired length.

4. Remove metal artifacts with a compressor and a deburring tool. Sand or file metal edges if

necessary.

5. Apply a light stream of cutting fluid across the tap site, and tap v-rail ends with a driver that

matches the desired hardware.

6. Remove any excess cutting fluid from the cut section of v-rail using a rag and/or a

compressor. Wipe dry with a non-fibrous cloth or implement.

7. Screw printed plastic end caps on v-rail ends. (The minimal loss to the traveling distance of

the gantry carriage is somewhat illusory as the placement of the gantry limit switches can be

offset. The point of covering the metal ends is to reduce the possibility of collision hang-ups,

while also indirectly protecting the robot against a hypothetical that involves damage to the

robot followed some human injury.)

8. Assemble the gantry wheels, followed by the gantry carriage plates. Attach wheels to plates.

9. Attach the gantry carriage to the v-rail and calibrate the wheel hardware until a fluid motion is

achieved. Sticking and knocking wheels indicate further calibration is needed.

Week of 11-9-2015

10. Test the stepper motor using a dev board, a controller, and a dedicated power source that can

be operated via a safety toggle switch.

11. Add limit switches to the motor controller circuitry and retest the system.

12. Remove stepper motor from the controller circuit.

13. Remove the gantry carriage from the v-rail and attach the stepper motor to its mount.

14. Attach the v-rail to the chassis, or a standalone mockup of sorts, and determine if the

placement is level.

15. Attach the gantry carriage and a timing belt to the v-rail.

16. Using enough wire slack to prevent fatiguing component leads, wire the motor controller

circuitry, minus the limit switches.

17. Energize the gantry and test operation of for instability factors. Turn off when done.

18. Attach the limit switches to the v-rail polar ends. Energize and retest the system.

19. Test the servos using an appropriate controller, dev board, and power source. Additional

testing may be conducted by programmatically tweaking the servo parameters.

Week of 11-16-2015

20. Stress test multiple copies of parts printed to interface with the gantry. Discard tested parts.

Week of 11-23-2015

21. Remove gantry carriage and attach the shoulder components. (Because the gantry assembly

is pieced together in a unique manner, combining mechanical retention with strict clearances,

and much of the generic (non-designed) parts had to be modeled alongside the custom parts,

it is my intention to eventually supply cartoon assembly instructions à laIkea.)

22. Reattach gantry carriage with shoulder to the v-rail section.

23. Connect servo controller to dev board circuitry.

24. Piece together the arm actuators and linkages, in sequence from shoulder to hand, testing

between each addition.

Week of 11-30-2015

25. Test setup by operating the gantry while digitally posing the arm.

26. Implement a cable management solution and test said solution against the stepper tool path.

Because several cables will be made from spliced unions, buffered by heat shrink tubing, and

the vehicle build will likely experience one or more teardowns, it would be beneficial at this

point to clearly identify bundled wires by means of some sorting scheme. In this way, simple

debugging practices won’t become a chore when dealing with common issues such as shorts

and weak solder joints.

27. Construct the end effector and test using the circuit used to test the other servos.

28. Test the functionality of the image processing module using computational means, to be

compared with results obtained while the camera is mounted to the arm.

29. Disable the stepper motor.

Week of 12-7-2015

30. Use the smart servo SDK to record the numerical markers generated by the desired arm

gestures. Test the arm by feeding it the recorded numerical markers.

31. Secure the controller boards to the chassis. Experiment with printed housing if necessary.

32. Attach the camera mount and camera to the arm. Add the camera cable to the cable

management solution

33. Test the arm movements for gestures that might impinge the operation and/or structural

integrity of the image processing module.

34. Test inputs bussed from the image processing module to the master controller, from where

they are relayed to the smart servos. Do processed image results translate into desirable arm

gestures? Take the time to calibrate this interface, using a range of end effector substitutions

if helpful.

35. Attach primary end effector to arm and run more image processing tests that relate to fine

motor control. Determine which arm gestures work best with the camera, and also adjust the

camera mount angle during these trials.

Week of 12-14-2015

36. Programmatically train the arm to match gestures that correspond with the necessary

actionable events required to reach and manipulate blocks at varying heights. Test build to

dial in a sweet spot that balances speed and precision of execution.

37. Test the validity of response to illegal block configurations. Run drills that deal with

compromised scenarios, as means of assessing the robustness of the system.

38. Run tests to determine how best to actuate the delivery mechanisms built into the

elevator car.

Week of 12-21-2015

39. Program the arm to assist the camera with long range shots, such as knowing how and when

to capture the color-coded order of the RAIL bins.

40. Evaluate which servo-related positional instructions yield consistently satisfactory results.

41. Calibrate the arm such that it can fluidly transition between the desired range of gestural

presets that have been calculated from the arm test data.

Week of 12-30-2015

42. Program emergency gestures that account for unexpected gantry movements, and attempt to

compensate for a non-responsive servo by geometrically moving around it where possible.

43. Lubricate the v-rail and begin running speed trials.

44. Keep all software and firmware updated, unless said update is released so close to a race date

that it could result in a barked software solution with no time to debug the mess.

45. Routinely inspect the build for signs of fatigue.

## Image Processing & Lighting

Aaron McDaniel

### Software Coding

The image processing design will be implemented on the front end by using a command line, ffmpeg command. This will capture the image from the webcam onto the BeagleBone. Once this is complete, the software design should be in a hustle to finish as it gives information to the arm on how to react. The next step would be testing. Several tests will be done as the software is written, but will undergo several tests while just taking pictures when called from the logistics system. Once the communication between the logistics system and the image processing is complete, testing of the camera mounted on the arm should occur. This testing process will allow for an understanding of the arms interpretation of the color, size, QR code, shape, and position of the block in space.

### Hardware

The electronic transistor switch should be built in parallel of the software development so the switch is ready for implementation when the initial software is complete.

### Wiring

The wiring from the camera will implement via USB. The light source will be connected to the G-9612 bus bar via an electronic switch for state purposes.

### Mounting

Next, the camera and the light source will be mounted on the arm.

### Software Loading

The coded software will be loaded onto the BeagleBone black used in the autonomous robot. Several rounds of software testing will take place.

## Propulsion

Kevin Houston

The robot will be best built from the ground up. Once the bottom level of the chassis has been built. The motors will need to be mounted with the correct measurements taken to verify that the robot does not go outside of the 12x12x12 constraint.

1. The blocks for the back wheels will need to be mounted in the selected space.
2. The motor mount will be attached to the top of the specified blocks.
3. The motors will be tested to make sure they are functioning correctly.
   1. Voltage tests
   2. Torque/weight cases.
4. The motors shall be mounted onto the selected mounts.
5. The hubs/wheels/sprocket sets will be created.
   1. The universal wheel hubs will be screwed to the wheels facing the center of the robot.
   2. The back sprockets will be screwed to the outside of the wheels.
6. The specified back two hub/wheel/sprocket sets will be mounted to the two back motors by screwing the hubs using the necessary hex screwed to the D shaped shafts of the motors.
7. The blocks for the axle/wheel mount of the front wheels will be placed in the measured place, allowing space for the storage area as well as the wheel sets to fit correctly.
8. Motor mounts/axle mounts will be placed on top of the blocks.
9. The axles will be placed into the motor mounts acting as shafts for the wheels to mount on.
10. The axles will be screwed into place using locknuts.
11. 5mm washers will be placed onto each axle before the wheel sets to prevent damage to the hub.
12. The chains for each side of the robot will be placed on the back sprockets and the front wheel sets, then the wheel sets will be placed on their respective axles.
13. Another washer will be placed on the outside of the wheels to prevent damage from rubbing against the wheel collars that will secure the wheels in place.
14. Connect the motor controller to the BeagleBone Black
15. Verify the connection, then connect the RioRand to the specified motors.
16. Connect the RioRand to the power source/battery
17. Gather the data output from the BeagleBone black to test and verify the capabilities of the Pololu motors and to match against theoretical calculations.

## Navigation

1. Test the connectivity to the BeagleBone Black through the powered USB hub. Raw data should be able to be viewed through the serial connection. This has been tested and proved to work in a Debian environment but not on the actual BBB.
2. Obtain distance data through BBB. This requires converting the raw data that is usable.
3. Complete the algorithm needed to determine position (See document in Appendix).
4. Obtain desired position results independent of the robot with proper testing.
5. Place the LIDAR unit on the robot.
6. Modify the code to meet logistics’/controller’s requirements
7. Test navigation with the robot until desired results are achieved.

## Microcontroller & Logistics

Peter Corcoran

### Software Coding

The BLAZE software system needs to be completed because it exposes system functions that will be used during system testing. BLAZE has a command mode that allows user entry of specific command that will allow the robot to perform certain system functions.

### Execute Test Plan

The next step in the implementation plan for the microcontroller will be to execute each step item in the Test plan in relationship to the Microcontroller and Logistics system. Multiple rounds of testing may be executed to ensure the configuration of the BBB is correct.

### Chassis Mounting

Next the BBB and related hardware will be mounted on the Chassis system.

### Wiring

Next, the system will be wired to each other sub-system once the sub systems are added to the Chassis

### Software Loading

Multiple rounds of software testing and debugging will be executed to tune the robot. Each route needs to be planned out and loaded via settings file. Debugging the routes will occur over time and the BBB USB connector will need to be exposed to access the board.

## Power & Chassis

Ben Henson

### First Floor

The power and chassis implementation plan will follow a ground up approach. The first floor and all its components will first be implemented. The motor mount blocks will be attached to the chassis. Next, the motors will be fixed to the motor mount blocks. The blocks associated with the passive wheel blocks will then be fixed to the first floor. Then, the wheels, sprockets and chains will be assembled. The Delivery and Storage linear actuator used in delivery method two will be fixed to the surface of floor 1. The associated rake will then be attached the linear actuator. The threaded rod will then be placed. The passive v-rail rail will also be fixed to the floor at this time. The power distribution circuit will also be placed on this floor.

### Second Floor

Floor two will now be created. The second floor will be aligned with the threaded rod and passive rail and fixed at the correct height. The passive rail will be attached to the second floor. The LIDAR box will then be placed in its slot and some adhesive will be used to ensure it does not slip.

### Third Floor

Floor three will now be created. The third floor will be aligned with the threaded rod and the passive rail and fixed at the correct height. The passive rail will be attached to the third floor.

### Fourth Floor

Floor four will now be created. Again the floor will be aligned and fix in place. The passive rail will be attached to the fourth floor. The gantry system will then be attached to this floor. With the completion of the fourth floor the necessary negative space has been created for the elevator. The linear actuator will then be attached within the negative space. Next, the elevator “taco” will be placed in the negative space and will be fixed to the linear actuator.

# Testing Plan

## Delivery & Storage

Leah Watkins

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID / Name** | Delivery Method 1 (without arm) | | | | | | |
| **Date Created** | 10/30/15 | | | | | | |
| **Created By** | Leah Watkins | | | | | | |
| **Tester ID / Name** |  | | | | | | |
| **Test Date** | TBD | | | | | | |
| **Special Prerequisites** | Taco must be built | | | | | | |
| **Step # / Action** | | | **Expected Result** | | **Actual Result** | |
| Loop will be pulled to expel blocks mechanically using false-bottom | | | Block will fall from taco | | TBD | |
| **Test Case Passed Yes** | | | **Test Case Failed** | | | | |
| **Comments:** |  | | | | | | |
|  |  |  | |  | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID / Name** | Delivery Method 1 (with arm) | | | | | |
| **Date Created** | 10/30/15 | | | | | |
| **Created By** | Leah Watkins | | | | | |
| **Tester ID / Name** |  | | | | | |
| **Test Date** | TBD | | | | | |
| **Special Prerequisites** | Taco and arm must be built and installed on the chassis. | | | | | |
| **Step # / Action** | | | **Expected Result** | | **Actual Result** | |
| Loop will be pulled to expel blocks mechanically using the arm | | | Block will fall from taco | | TBD | |
| **Test Case Passed Yes** | | | **Test Case Failed** | | | |
| **Comments:** |  | | | | | |
|  |  |  | |  | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID / Name** | Test LA and LAC (uninstalled) | | | |
| **Date Created** | 10/30/15 | | | |
| **Created By** | Leah Watkins | | | |
| **Tester ID / Name** |  | | | |
| **Test Date** | TBD | | | |
| **Special Prerequisites** | LA and LAC connected together with power source and PWM regulation. | | | |
| **Step # / Action** | | **Expected Result** | **Actual Result** |
| Test LA1 & LAC1 | | Stroke extends and retracts are directed |  |
| Test LA2 & LAC2 | | Stroke extends and retracts are directed |  |
| **Test Case Passed Yes** | | **Test Case Failed** | | |
| **Comments:** |  | | | |
|  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID / Name** | Test LA and LAC (installed) | | | |
| **Date Created** | 10/30/15 | | | |
| **Created By** | Leah Watkins | | | |
| **Tester ID / Name** |  | | | |
| **Test Date** | TBD | | | |
| **Special Prerequisites** | LA2 and LAC2 connected together with power source and PWM regulation installed on the chassis. | | | |
| **Step # / Action** | | **Expected Result** | **Actual Result** |
| Test LA1 & LAC1 | | Stroke extends and retracts are directed |  |
| Test LA2 & LAC2 | | Stroke extends and retracts are directed |  |
| **Test Case Passed Yes** | | **Test Case Failed** | | |
| **Comments:** |  | | | |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID / Name** | Test Delivery Method #2 | | |
| **Date Created** | 10/30/15 | | |
| **Created By** | Leah Watkins | | |
| **Tester ID / Name** |  | | |
| **Test Date** | TBD | | |
| **Special Prerequisites** | Taco, Vrails, LA and LAC pairs installed. | | |
| **Step # / Action** | | **Expected Result** | **Actual Result** |
| Activate horizontal LA | | Stroke extends and retracts are directed to push out blocks simultaneously. |  |
| **Test Case Passed Yes** | | **Test Case Failed** | |
| **Comments:** |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID / Name** | Test Elevator | | |
| **Date Created** | 10/30/15 | | |
| **Created By** | Leah Watkins | | |
| **Tester ID / Name** |  | | |
| **Test Date** | TBD | | |
| **Special Prerequisites** | Taco, Vrails, LA and LAC pairs installed. | | |
| **Step # / Action** | |  |  |
| Activate vertical LA | | Elevator rises and lowers accordingly |  |
| **Test Case Passed Yes** | |  | |
| **Comments:** |  | | |

## Cargo Retrieval

Evan Gilbert

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT1 / Square Rail Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | 11-3-2015 | |
| Special Prerequisites | T-square and packing tape | |
| Step # / Action | Expected Result | Actual Result |
| Step 1 | Blunt metal edges of T-square with a layer of packing tape. |  |
| Step 2 | Slide extrusion along corner of T-square. Repeat for every face. |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT2 / Stepper Motor Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | 11-10-2015 | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | CW/CCW ops work |  |
| Step/Action 2 | micro stepping tolerances are met |  |
| Step/Action 3 | torque rating matches reality |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT3 / Basic Servo Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | servo should hit target returns |  |
| Step/Action 2 | servo should record target values |  |
| Step/Action 3 | rated servo torque should match reality |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT4 /Gantry Carriage Calibration** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | the gantry wheels should neither stick nor jog in place |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT5 / Cable Management Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step 1 | movement of the wire bundle from the gantry components should not impinge on the operations of neighboring subsystems |  |
| Action 2 |  |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT6 / Gantry Homing Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | tripped limit switch should trigger a meaningful routine that can be used to either home or slowly reverse the gantry away from the limit switch |  |
|  |  |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT7 / Servo Network Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | the loss of a single actuator should not disrupt an entire network |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT8 / Gesture Log Test** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites | requires ROBOTIS SDK | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | arm movements should translate into meaningful numerical data |  |
| Step/ Action 2 | arm should be consistent when performing programmed gestures |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **CRT9 / Experiment with Motor Controller Sleep Cycles** | |
| Date Created | 10-27-2015 | |
| Created By | Evan Gilbert | |
| Tester ID / Name | evanich / Evan Gilbert | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | power consumption should plummet in a sleep state |  |
| Step/Action 2 | transitioning between sleep cycles should not disrupt values in running memory |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

## Image Processing & Lighting

Aaron McDaniel

### BeagleBone Connectivity Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 1: BeagleBone Connectivity Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | BeagleBoneBlack with USB connection | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | BeagelBoneBlack should connect to power and commence operation. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### BeagleBone Connectivity via Putty Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 2: BeagleBone Connectivity via Putty Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | BeagleBoneBlack with USB connection | |
| Step # / Action | Expected Result |  |
| Step/Action 1 | BeagleBoneBlack should interface with putty. BeagleBoneBlack and camera should be recognized through putty via lsusb command. |  |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Electronic Switch Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 3: Electronic Switch Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | All elements of electronic switch design obtained | |
| Step # / Action | Expected Result |  |
| Step/Action 1 | Implement electronic switch design to justify design |  |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Image Capture Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 4: Image Capture Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | Ajmcdan / Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites |  | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | ffmpeg command/command should be recognized by the BeagleBone black and obtain an image. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Pixel Recognition Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 5: Pixel Recognition Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Captured Image from webcam via BeagleBoneBlack | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Written software should recognize wanted pixel colors. | TBD |
| Step/Action 2 | Put output in GUI format to ensure pixel recognition. |  |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### QR Code Recognition Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 6: QR Code Recognition Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Captured Image from webcam via BeagleBoneBlack. Pre-saved image of QR code for comparison. | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Written software should recognize wanted QR Code. | TBD |
| Step/Action 2 |  |  |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Size Recognition Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 7: Size Recognition Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Captured Image from webcam via BeagleBoneBlack. | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Written software should recognize wanted size | TBD |
| Step/Action 2 |  |  |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Size Recognition Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 8: Size Code Recognition Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Captured Image from webcam via BeagleBoneBlack. | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Written software should recognize wanted position of blocks. | TBD |
| Step/Action 2 |  |  |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Lighting Source Test

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID / Name | **Test 9: Lighting Source Test** | | |
| Date Created | 10/26/2015 | | |
| Created By | Aaron McDaniel | | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | | |
| Test Date | TBD | | |
| Special Prerequisites | Lighting source constructed | | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Lighting source should power cycle. | TBD |
| Test Case Passed YES | Test Case | | |
| Comments: |  | | |

### Lighting Source to Controller Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 10: Lighting Source to Controller Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Lighting Source Test complete | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Lighting Source connected to controller/Lighting source should turn on without voltage issues. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### Lighting Source to Controller via Variable Resistor Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 11: Lighting Source to Controller via Variable Resistor Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Lighting Source to Controller Test failed | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Designed variable resistor connected to lighting source/ should control voltage from controller to lighting source. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

### BeagleBone to Image Processing Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 12: BeagleBone to Image Processing Test** | |
| Date Created | 10/25/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/ Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be setup and powered on.  SSH connection via Putty to allow command line execution of arguments  Base Image Processing Link Layer code is completed | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect USB Hub to BBB | TBD |
| Step/Action 2 | Cameras connected to hub show up when usbls command is executed in Putty session |  |
| Step/Action 3 | Camera 1 is able to capture image and save it to the BBB disk |  |
| Step/Action 4 | Camera 2 is able to capture image and save it to the BBB disk |  |
| Step/Action 5 | Using BLAZE in command mode, the system is able to identify QRCodes by taking image and returning the QRCode in picture. |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

### Wiring Space Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 13: Wiring Space Test** | |
| Date Created | 10/26/2015 | |
| Created By | Aaron McDaniel | |
| Tester ID / Name | ajmcdan/Aaron McDaniel | |
| Test Date | TBD | |
| Special Prerequisites | Arm/camera and lighting source constructed and wiring connected | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | All wiring should lay in place without obstructing movement of the gantry. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

## Propulsion

Kevin Houston

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 1: 360 Degree Turn Torque** | |
| Date Created | 10/30/2015 | |
| Created By | Kevin Houston | |
| Tester ID / Name | caoimhin/Kevin Houston | |
| Test Date | TBD | |
| Special Prerequisites | Need multiple parts | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Complete wheel arrangement with bottom level of chassis | TBD |
| Step/Action 2 | Apply predicted weight on chassis | TBD |
| Step/Action 3 | Use hanging scale and string to pull chassis around 360 degrees |  |
| Step/Action 4 | Observe force needed in order to accomplish the turn. |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 2: Test Motor Reverse Torque** | |
| Date Created | 10/30/2015 | |
| Created By | Kevin Houston | |
| Tester ID / Name | caoimhin/Kevin Houston | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be connected to the RioRand  BBB should send pwm signals to the RioRand to reverse the current  in the motor.  Verify weight pulled/torque using hanging scale. | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect Beaglebone Black to the RioRand. | TBD |
| Step/Action 2 | Connect the RioRand to the Motor. | TBD |
| Step/Action 3 | Attach a string around the wheel. | TBD |
| Step/Action 4 | Have the Beaglebone send pwm to run the motor in reverse. | TBD |
| Step/Action 5 | Observe the weight pulled by the wheel, Compare against forward torque produced. | TBD |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID / Name | **Test 3: RioRand Power Test** | | |
| Date Created | 10/30/2015 | | |
| Created By | Kevin Houston | | |
| Tester ID / Name | caoimhin/Kevin Houston | | |
| Test Date | TBD | | |
| Special Prerequisites | Small variety of low power test batteries will be connected to the RioRand controller.  Measure the current produced and compare to theoretical values | | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect test battery to RioRand | TBD |
| Step/Action 2 | Measure current across motor connection pins and GND pin of controller | TBD |
| Test Case Passed YES/NO | Test Case | | |
| Comments: |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| Test Case ID / Name | **Test 3: Wheel Arrangement Space test** | | |
| Date Created | 10/30/2015 | | |
| Created By | Kevin Houston | | |
| Tester ID / Name | caoimhin/Kevin Houston | | |
| Test Date | TBD | | |
| Special Prerequisites | Several parts | | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Measure the hub/wheel/sprocket width | TBD |
| Step/Action 2 | Verify that the arrangement meets the 1” chassis space constraint | TBD |
| Step/Action 3 | Attach specified wheel sets on the chassis/mounts | TBD |
| Step/Action 4 | Measure and verify that the wheel arrangement is within the space allocation constraint | TBD |
| Test Case Passed YES/NO | Test Case | | |
| Comments: |  | | |

## Navigation

Terence Staples

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 1: Receive serial data from LIDAR on the BBB** | |
| Date Created | 10/30/2015 | |
| Created By | Terence Staples | |
| Tester ID / Name | tstap11/Terence Staples | |
| Test Date | TBD | |
| Special Prerequisites | The LIDAR unit is connected to the USB hub which is externally powered and is connected to the BeagleBone | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect the USB hub to the external power and to the BeagleBone | The BeagleBone recognized that a 4 port usb hub had been connected |
| Step/Action 2 | Connect the LIDAR unit to the USB hub and make sure it is receiving power | The motor received power and causing the sensor unit to spin |
| Step/Action 3 | Find location of the LIDAR on the BeagleBone | TBD |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 2: Determine position on board using LIDAR independent of the rest of the robot** | |
| Date Created | 10/30/2015 | |
| Created By | Terence Staples | |
| Tester ID / Name | tstap11/Terence Staples | |
| Test Date | TBD | |
| Special Prerequisites | The navigation algorithm needs to be complete to the point of testing | |
| Step # / Action | Expected Result |  |
| Step/Action 1 | Connect the LIDAR unit to the BBB | TBD |
| Step/Action 2 | Run the navigation algorithm on the BBB | TBD |
| Step/Action 3 | Check if the LIDAR can determine the correct position on the playing board | TBD |
| Step/Action 4 | Move LIDAR on playing board | TBD |
| Step/Action 5 | Repeat step 5 | TBD |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

## Microcontroller & Logistics

Peter Corcoran

### BeagleBone to Delivery Controller Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **BeagleBone to Delivery Controller** | |
| Date Created | 10/25/2015 | |
| Created By | Peter Corcoran | |
| Tester ID / Name | TBD | |
| Test Date | TBD | |
| Special Prerequisites | Need multiple parts | |
| Step # / Action | Expected Result | Actual Result |
| Step 1 | Setup connection between BBB & Delivery Controller with test servo. Execute PWM commands from command line of BeagleBone |  |
| Step/Action 2 | Servo should respond to PWM signal |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

### BeagleBone with Delivery Servos Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **BeagleBone with Delivery Servos** | |
| Date Created | 10/25/2015 | |
| Created By | Peter Corcoran | |
| Tester ID / Name | Peter Corcoran | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be setup and powered on. S  SSH connection via Putty to allow command line execution of arguments  Test 14.6.1 should be executed successfully | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect Delivery linear servos | TBD |
| Step/Action 2 | Execute PWM commands from command line of BBB |  |
| Step 3 | Servo should respond to PWM single depending on the duty cycle of the signal. |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

### BeagleBone to Arm Controller Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **BeagleBone to Arm Controller** | |
| Date Created | 10/25/2015 | |
| Created By | Peter Corcoran | |
| Tester ID / Name | Peter Corcoran | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be setup and powered on.  SSH connection via Putty to allow command line execution of arguments  Arm Controller Link Layer code will need to be completed. | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect USB Hub to BBB | TBD |
| Step/Action 2 | Connect USB2AX controller to USB Hub |  |
| Step 3 | Execute usbls command to view if device is available. |  |
| Step 4 | Execute command line command to determine if communication is open to the controller |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

### BeagleBone to Lidar Controller Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **BeagleBone to Lidar Controller** | |
| Date Created | 10/25/2015 | |
| Created By | Peter Corcoran | |
| Tester ID / Name | Peter Corcoran | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be setup and powered on.  SSH connection via Putty to allow command line execution of arguments  Lidar Link Layer code will need to be completed to execute test. | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect USB Hub to BBB | TBD |
| Step/Action 2 | Connect Lidar to USB HUB |  |
| Step 3 | Execute usbls command to view if devices are available. |  |
| Step 4 | Command line command executed will produce list of distances to various points on the field. |  |
| Step 5 | With Port settings defined, the LIDAR should be placed on the Port, with BLAZE in command mode, the console should print out the location of the LIDAR system in inches. |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

### BeagleBone to Motor Controller Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **BeagleBone to Motor Controller Test** | |
| Date Created | 10/25/2015 | |
| Created By | Peter Corcoran | |
| Tester ID / Name | Peter Corcoran | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be setup and powered on.  SSH connection via Putty to allow command line execution of arguments  Motor Controller PWM Link Layer should be completed, BLAZE software written to execute command line parameters | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | System doesn’t burn up when motors are connected to the h-bride, and h-bridge is connected to the BBB | TBD |
| Step/Action 2 | PWM commands entered into Putty session start the motors |  |
| Step 3 | PWM commands entered into Putty session stop the motors |  |
| Step 4 | PWM command entered into Putty session speed/slow motors |  |
| Step 5 | PWM commands entered in Putty session reverse motor directions |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

### BeagleBone to Image Processing Test

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **BeagleBone to Image Processing Controller** | |
| Date Created | 10/25/2015 | |
| Created By | Peter Corcoran | |
| Tester ID / Name | Peter Corcoran | |
| Test Date | TBD | |
| Special Prerequisites | BBB should be setup and powered on.  SSH connection via Putty to allow command line execution of arguments  Base Image Processing Link Layer code is completed | |
| Step # / Action | Expected Result | Actual Result |
| Step/Action 1 | Connect USB Hub to BBB | TBD |
| Step/Action 2 | Cameras connected to hub show up when usbls command is executed in Putty session |  |
| Step/Action 3 | Camera 1 is able to capture image and save it to the BBB disk |  |
| Step/Action 4 | Camera 2 is able to capture image and save it to the BBB disk |  |
| Step/Action 5 | Using BLAZE in command mode, the system is able to identify QRCodes by taking image and returning the QRCode in picture. |  |
| Test Case Passed YES/NO | Test Case | |
| Comments: |  | |

## Power & Chassis

Ben Henson

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 1: Verify correct voltage regulation of D24V50F5** | |
| Date Created | 11/1/2015 | |
| Created By | Ben Henson | |
| Tester ID / Name | Bhenson1 / Ben Henson | |
| Test Date | TBD | |
| Special Prerequisites | Receive parts | |
| Step # / Action | Expected result | Actual result |
| Step/Action 1 | Input a voltage toward the low limit and verify that 5 volts is still output. | TBD |
| Step/Action 2 | Input a voltage toward the middle of the limit and verify that 5 volts is still output. | TBD |
| Step/Action 3 | Input a voltage toward the high limit and verify that 5 volts is still output. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 2: Verify correct voltage regulation of the 12 volt regulator** | |
| Date Created | 11/1/2015 | |
| Created By | Ben Henson | |
| Tester ID / Name | Bhenson1 / Ben Henson | |
| Test Date | TBD | |
| Special Prerequisites | Receive parts | |
| Step # / Action | Expected result | Actual result |
| Step/Action 1 | Input a voltage toward the low limit and verify that 12 volts is still output. | TBD |
| Step/Action 2 | Input a voltage toward the middle of the limit and verify that 12 volts is still output. | TBD |
| Step/Action 3 | Input a voltage toward the high limit and verify that 12 volts is still output. | TBD |
| Test Case Passed YES | Test Case | |
| Comments: |  | |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test 3: Verify that the potentiometer is working of the 12**  **Volt regulator** | |
| Date Created | 11/1/2015 | |
| Created By | Ben Henson | |
| Tester ID / Name | Bhenson1 / Ben Henson | |
| Test Date | TBD | |
| Special Prerequisites | Receive parts | |
| Step # / Action | Expected result | Actual result |
| Step/Action 1 | Vary the potentiometer as the output voltage is connected to the oscilloscope and observe the result. | TBD |
| Test Case Passed YES | Test Case |  |
| Comments: |  |  |

|  |  |  |
| --- | --- | --- |
| Test Case ID / Name | **Test: Test chassis strength** | |
| Date Created | 11/1/2015 | |
| Created By | Ben Henson | |
| Tester ID / Name | Bhenson1 / Ben Henson | |
| Test Date | TBD | |
| Special Prerequisites | Receive parts | |
| Step # / Action | Expected result | Actual result |
| Step/Action 1 | Place weight on chassis that simulates the estimated weight of the chassis. Make sure the structure is capable of holding this amount of weight. | TBD |
| Test Case Passed YES | Test Case |  |
| Comments: |  |  |

# Appendix A – Parts List

# Appendix B – Cable & Connectors

# Appendix C – Additional Information