



## SoutheastCon Team A Final Report



| Date      | Revision | Comments      |
|-----------|----------|---------------|
| 4/16/2015 | 1        | First edition |
|           |          |               |
|           |          |               |
|           |          |               |

### Team Members

Nils Bjerén  
Ryan-David Reyes  
Julian Velasquez  
Kurt Marsman  
Chris Lewis  
Donovan Carey  
James Pace

### Faculty Advisors

Dr. Bruce Harvey  
Dr. Michael Frank  
Dr. Linda DeBrunner  
Dr. Carl Moore



## Contents

|   |    |
|---|----|
| Executive Summary .....   | 3  |
| 1 Introduction .....  | 3  |
| 1.1 Acknowledgements .....  | 3  |
| 1.2 Problem Statement.....  | 3  |
| 1.2.1 Problem .....   | 3  |
| 1.2.2 Solution.....   | 4  |
| 1.3 Operating Environment.....  | 4  |
| 1.4 Intended Use(s) and Intended User(s).....                               | 4  |
| 1.5 Assumptions and Limitations .....                                       | 4  |
| 1.6 End Product and Other Deliverables.....                                 | 5  |
| 2 System Design.....  | 6  |
| 2.1 Overview of the System.....   | 6  |
| 2.2 Major Components of the System.....                                     | 7  |
| 2.2.1 Primary Microcontroller and Propulsion Subsystem .....                | 7  |
| 2.2.2 Line Following Subsystem.....   | 8  |
| 2.2.3 Etch-A-Sketch and Playing Card Subsystem.....                         | 9  |
| 2.2.4 Simon Says and Rubik’s Cube Subsystem.....                            | 10 |
| 2.3 Subsystem Requirements.....   | 11 |
| 2.3.1 Requirements Specification for Chassis .....                          | 11 |
| 2.3.2 Requirements Specification for Main Controller.....                   | 12 |
| 2.3.3 Requirements Specification for Propulsion .....                       | 12 |
| 2.3.4 Requirements Specification for Power.....                             | 12 |
| 2.3.4 Requirements Specification for Line Following .....                   | 13 |
| 2.3.5 Requirements Specification for Arm 1: Etch-a-Sketch/Playing Card..... | 13 |
| 2.3.6 Requirements Specification for Arm 2: Simon Says/Rubik’s Cube .....   | 14 |
| 2.4 Performance Assessment.....   | 14 |

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

|   |    |
|---|----|
| 2.5 Pending Design Decisions .....                    | 14 |
| 2.5.6 Custom PCB Option .....                         | 14 |
| 3 Design of Major Components .....                    | 15 |
| 3.1 Chassis .....                                     | 15 |
| 3.2 Main Controller .....                             | 15 |
| 3.3 Propulsion .....                                  | 19 |
| 3.4 Power .....                                       | 25 |
| 3.4.1 Power Analysis .....                            | 25 |
| 3.4.2 Battery Type .....                              | 26 |
| 3.5 Line following .....                              | 27 |
| 3.6 Arm 1: Etch-a-Sketch/Playing Card .....           | 31 |
| 3.7 Arm 2: Simon Says/Rubik's Cube .....              | 38 |
| 4 Test Plan .....                                     | 43 |
| 4.1 System and Integration Test Plan .....            | 43 |
| 4.2 Test Plan for Major Subsystems .....              | 43 |
| 4.2.1 Completed Test Example .....                    | 44 |
| 4.2.2 Pending Test Example .....                      | 45 |
| 4.3 Summary of Test Plan Status .....                 | 46 |
| 5 Schedule .....                                      | 47 |
| 5.1 External Deadlines .....                          | 47 |
| 5.2 Project Functionality Schedule .....              | 47 |
| 5.3 Final schedule compliance and justification ..... | 48 |
| 6 Budget Estimate .....                               | 49 |
| 7 Conclusion .....                                    | 51 |
| Appendix A: Testing Plans .....                       | 52 |
| Appendix B: User's Guide .....                        | 71 |
| Setup/Configuration .....                             | 72 |
| Operation .....                                       | 72 |
| Appendix C: Software .....                            | 73 |

## Executive Summary

IEEE SoutheastCon is the annual conference for Region 3 of the Institute of Electrical and Electronics Engineers. This event includes several competitions, one of which is the hardware competition. The purpose of the project outlined in the present report was to compete in, and win, the 2015 SoutheastCon Hardware Competition. In order to do this, an autonomous robot was designed in accordance with the competition rules.

This year's competition, held in Ft. Lauderdale, had a "road trip" theme. The robot needed to navigate along a course represented by a white line on a black background. Along the course, four different classic road trip toys were to be "played with." These included a Rubik's Cube, the Simon Says game, an Etch-a-Sketch, and a deck of playing cards. The robot should complete the course as quickly as possible, completing the challenges without error in less than 5 minutes.

Team 1A's robot employed a combination of custom designed components in order to complete the challenges. A robot that will reliably be able to complete a majority of the tasks in the competition was built by March 20th, 2015 and competed in the Hardware Competition on April 11th. Within this document is an outline of the design process, as well as a risk assessment, testing plan, and user manual.

## 1 Introduction

### 1.1 Acknowledgements

The team would like to gratefully acknowledge Ramiro Velasquez and INNOVAtek for their generous \$500 donation towards the project, as well as the \$750 provided by the FAMU/FSU College of Engineering. In addition, the continued advice and suggestions of the ECE and ME Senior Design faculty have been invaluable in the design process.

### 1.2 Problem Statement

#### 1.2.1 Problem

The purpose of this project is to build an autonomous robot that will win the 2015 SoutheastCon Hardware Competition. In order to complete this task, the robot will have to be able to move along a white line on a black background, as well as complete four different "road trip" themed challenges. These challenges are: twisting one row of a Rubik's Cube 180 degrees, playing Simon Says for 15 seconds, drawing "IEEE" on an Etch-a-Sketch, and picking up a single playing card from a deck of 52. In addition to this, the robot must fit within a 1' by 1' by 1' box at the beginning and end of the course.

### **1.2.2 Solution**

As per the rules of the competition, an autonomous robot was designed and constructed from scratch. It consists of one “interface” on either side of the robot, each having the ability to play two games. One interface plays the Simon Says game and twists the Rubik’s Cube. The other writes IEEE on the Etch-a-Sketch and picks up the playing card. The white line is followed using an array of infrared sensors. Mecanum wheels are used to facilitate navigation along the track.

## **1.3 Operating Environment**

The environment in which the product will be used is the game board of the 2015 SoutheastCon Hardware Competition. The game board features multiple white branch lines off one main white line. In Ft. Lauderdale, the competition happened in a conference room with very dim lights. The local competition happened in the atrium with very bright lights. Both locations were very loud. .

## **1.4 Intended Use(s) and Intended User(s)**

The intended use of the project is to successfully build a robot from scratch that can compete in SoutheastCon 2015. The robot will have to be able to autonomously start, navigate the track, play Simon for 15 seconds, draw “IEEE” on an Etch-a-Sketch, rotate any row of a Rubik’s cube 180 degrees, and pick up a card from a deck of cards, taking the card to the finish line. The intended users of the project will be the engineers who built it, as they will be the ones taking the robot to the competition. The whole FAMU-FSU Electrical and Computer Engineering Department will be represented by the team and its robot.

## **1.5 Assumptions and Limitations**

The design is based on the following assumptions. Branches and corners on the white line will be deterministic, i.e. if sensed correctly, it is impossible to mistake a branch for a corner and vice versa. Sufficient time will be allotted between runs during the competition to charge/replace batteries, as well as to replace sticky surfaces (for example on the Etch-a-Sketch arm). The robot will not need to function for more than a total of 30 minutes before the battery can be fully recharged. As the robot is made according to the most up to date competition rules, a major assumption of the current design is that the rules will not change ahead of the competition. Another assumption is that toys of the same build and SKU have consistent operating parameters. For example, it is assumed that the torque required to turn the knobs on the Etch-A-Sketch the team purchased for testing will be the close to the torque needed to turn the knobs on the Etch-A-Sketch used at competition.

The limitations imposed by the competition are as follows. The robot shall be completely autonomous, requiring no human input other than placement in the starting position. The final design must fit within a 1’ by 1’ by 1’ box before starting and after finishing the course. At no point during the competition can the toys be “hidden” from the audience. No flammable liquids, high pressures, or otherwise dangerous

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

items must be part of the design. Finally, the entire course must be completed in less than 5 minutes. Self-imposed limitations include using a total of two subsystems (not more) in order to interface with four different games, as well as relying exclusively on microcontrollers rather than more complex devices such as the Raspberry Pi.

## **1.6 End Product and Other Deliverables**

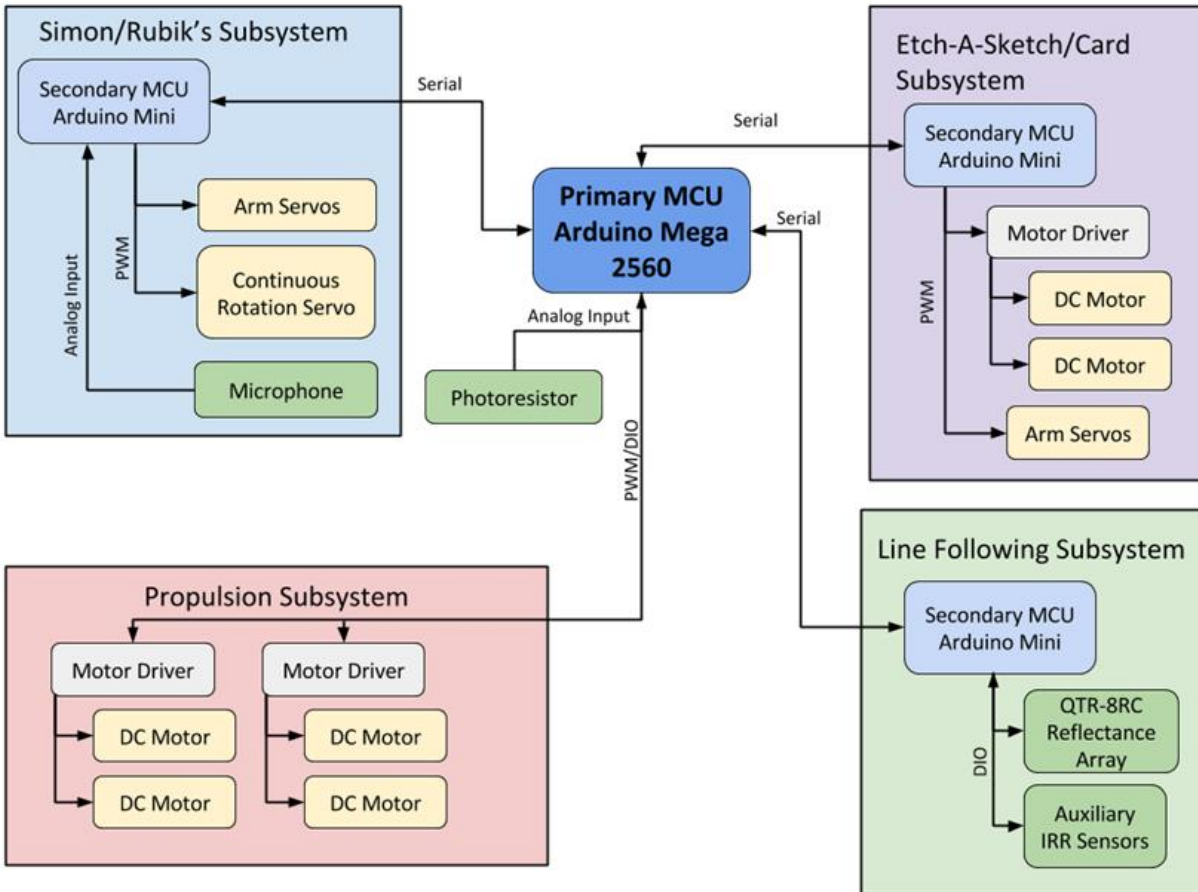
The end product is a fully autonomous robot that is capable of successfully completing the 2015 SoutheastCon Hardware competition challenges within the 5 minute time limit. This robo has the following capabilities:

- Navigate according to a white line
- Twist one row of a Rubik's Cube
- Write "IEEE" on an Etch-a-Sketch
- Play Simon Says for 15 seconds
- Pick up and carry a single playing card

The only deliverable product for this project is the complete autonomous robot. The robot will need to be finished before the local competition in March. In addition to this, there are six milestone reports and presentations according to the requirements of the Senior Design course.

## 2 System Design

### 2.1 Overview of the System



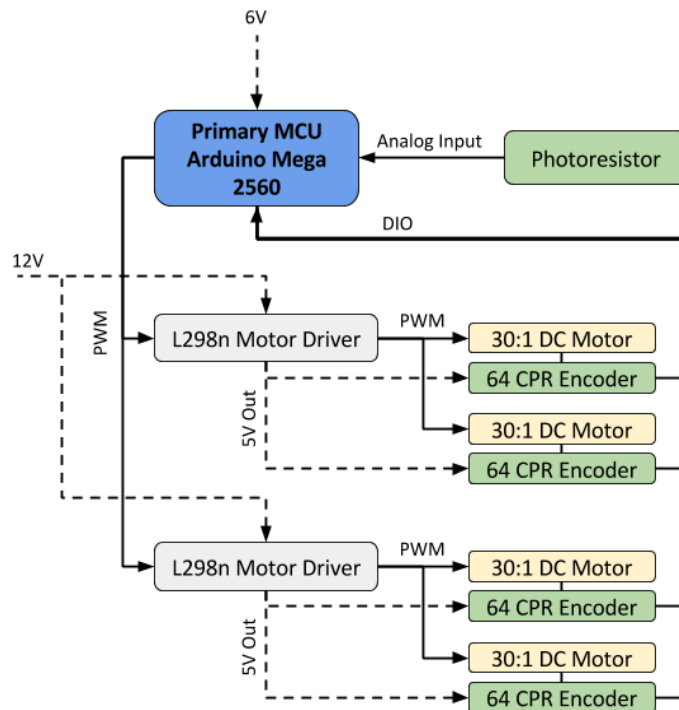
**Figure 2.1: Top level diagram of the system**

As seen in Figure 2.1, there are multiple microcontrollers set up in a primary-secondary configuration, otherwise known as a master-slave configuration. The primary microcontroller is an Arduino Mega 2560, which is in charge of all the major subsystems of the robot. The subsystem it directly oversees is the propulsion subsystem, which consists of brushed DC motors with encoders. The other subsystems are delegated to secondary microcontrollers, each being an Arduino Mini with an ATmega 328p. They communicate with the primary microcontrollers over a serial connection. All subordinate microcontrollers have a dedicated communication line, as this was trivial to implement. The Line Following Subsystem consists of an Arduino Mini, two 8-sensor rows of IRR sensors, and another set of 2 reflectivity sensors. The Simon Says and Rubik's Cube subsystems consists of an Arduino Mini, a servo to lift the arm, a

continuous rotation servo to rotate the Rubik's Cube, and a microphone to distinguish the Simon Says Game cues. The Etch-A-Sketch and Card Subsystem is comprised of an Arduino Mini and two DC motors for rotating the Etch-A-Sketch knobs, and a motor connected to a power screw for lifting the arm. For the power subsystem, an 11.1 V battery will supply the drive motors directly, and the same will also be regulated down to 6V in order to power servos, smaller DC motors, and the microcontrollers.

## 2.2 Major Components of the System

### 2.2.1 Primary Microcontroller and Propulsion Subsystem



**Figure 2.2: Propulsion Subsystem**

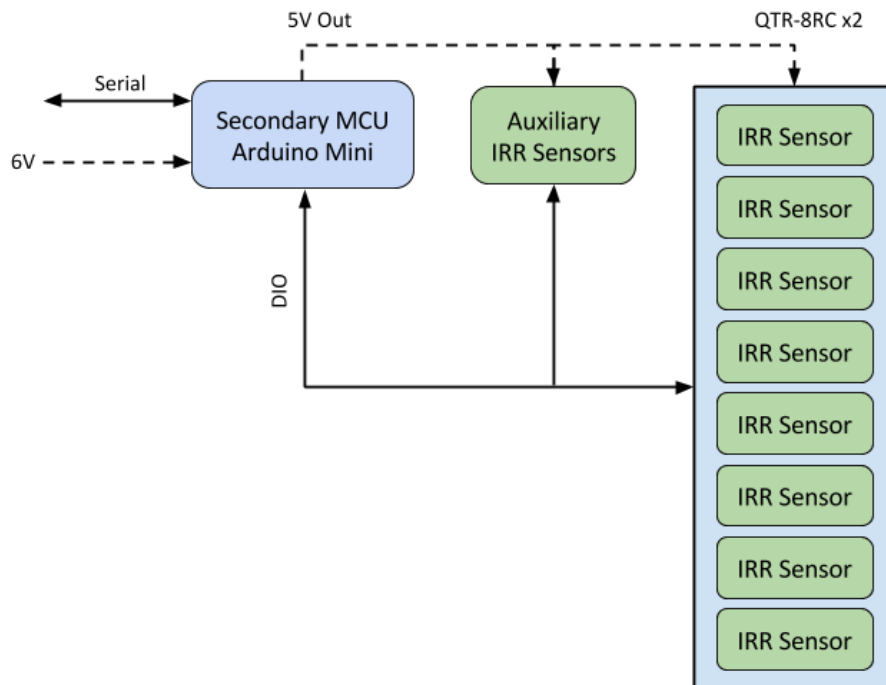
The propulsion subsystem consists of the primary Arduino Mega 2560, four DC motors with encoders, two motor drivers, and a photoresistor. The Arduino Mega is in charge of the overall state and logic of the robot, determining when it is time to follow a line or play a game depending on the sensor data communicated. The four DC motors are required since omnidirectional movement was desired, thus each motor independently drives a Mecanum Wheel. The MCU sends a PWM signal to two Solarbotics L298n Dual motor drivers, each of which amplify the PWM signal to drive two motors with 4A maximum output current. The L298n motor drivers are powered directly by the 11.1V high current battery, and they contain a 5V regulated output which is used to power the encoders on the motors. Each motor is a 30:1



gear ratio motor with 64 counts per revolution quadrature encoders connected to the motor shaft with a free run current of 300mA. This results in 1920 counts per revolution of the output shaft, which drives a Mecanum wheel. The quadrature encoder output is sampled by the MCU for use in velocity PID control of the motors.

The photoresistor is used to sense the start LED. The start of a heat is signaled by an LED turning off underneath the chassis of the robot. The photoresistor is placed in a pull-down resistor configuration, such that the MCU will read 0V from the voltage divider when the LED is ON, and 5V when the LED is OFF, using the analog input.

### 2.2.2 Line Following Subsystem



**Figure 2.3: Line Following Subsystem**

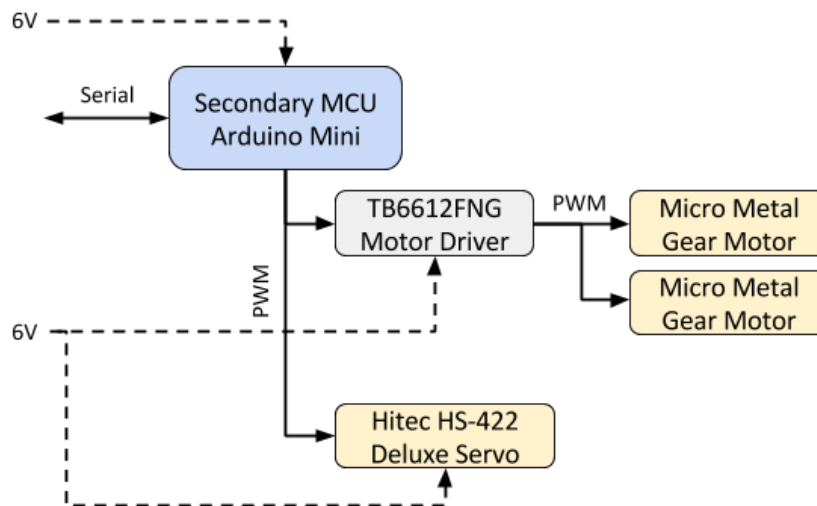
The Primary Arduino Mega has to sample the eight encoder lines and perform quadrature decoding and counting, perform PID calculations, calculate PWM control, and enact state decisions of the robot, which is a lot to do for a single microcontroller. Hence, the Line Following subsystem was assigned a dedicated Arduino Mini Microcontroller to help offload the processing work from the main Primary Arduino Mega. The Arduino Mini is powered by the 11.1V battery, and the sensors are powered by the 5V out pin

provided by the Arduino Mini. The code samples the IRR sensors sequentially, thus the Arduino will never have current overdraw from the output pin as only one will be powered at a time.

The main task of the Secondary Arduino Mini is to sample both the rows of QRE 113 Reflectivity sensors and determine the current line following status from the data obtained. This is accomplished by sending a pulse on the Digital Input/Output line associated with a particular sensor. This causes the IR LED to light up, and the reflected light charges a capacitor via a phototransistor. The MCU then measures the amount of time the capacitor takes to discharge. The discharge time of the capacitor is correlated with the amount of IR light reflected back from the surface, thus indicating the presence of a line. The grid configuration was chosen due to its versatility in determining the orientation of the line, detecting the existence of branches along the main line, as well as determining how far into solid white square the robot has ventured. Branch detection is augmented by the usage of auxiliary IRR sensors placed along the outer edges of the chassis, as branch detection with the main IRR sensors alone proves difficult.

The Arduino Mini communicates with the Arduino Mega through the TTL Serial communication protocol provided by an on board UART chip. The Mini samples the IRR sensors, and determines a velocity vector which the robot must achieve to correctly orient itself on the line. The Mini sends this information once it receives a request from the Mega.

### 2.2.3 Etch-A-Sketch and Playing Card Subsystem



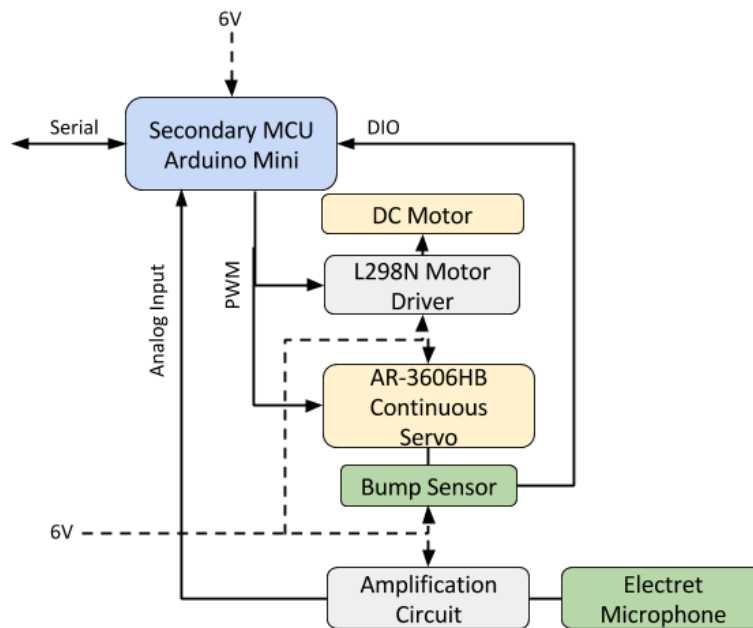
**Figure 2.4: Etch-A-Sketch and Playing Card Subsystem**

The Etch-A-Sketch and Playing Card Subsystem is controlled by an Arduino Mini, powered by the 9V battery. Overall, the system consists of a DC Motor to raise and lower the manipulator, and two DC

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

motors attached to the arm to turn the knobs of the Etch-A-Sketch. The arm lifting motor and a TB6612FNG motor driver are powered from a 5V regulated power source. The TB6612FNG motor driver allows up to a 1A peak current draw for the micro gear motors. This is more than required to drive the knobs on the Etch-A-Sketch. The Micro Gear motors are small motors that will be placed normal to the knob's surface. These produce about 1.7 kg-cm of torque, which is enough to spin the knob when they are adhered to the surface. To pick up the playing card, adhesive will be used on the edge of the end-effector. This Secondary MCU communicates with the Primary MCU via Serial TTL interface provided by the onboard UART chip. This MCU stays in a 'wait' state until the Arduino Mega commands it to begin playing the game. Once finished, the Etch-A-Sketch MCU reports that it is finished, and the Arduino Mega will have the robot continue down the course.

### 2.2.4 Simon Says and Rubik's Cube Subsystem



**Figure 2.5: Simon Says and Rubik's Cube Subsystem**

Overall, this system consists of one servo, DC motor, and a microphone. The DC motor is used to raise and lower the arm, while the servo is used to turn the Rubik's Cube and hit the buttons on the Simon Says game. An Arduino Mini controls this subsystem, and is powered by the regulated 6V from the main battery. It waits for the Arduino Mega to send the command to play either of the games, and once finished with the game, reports back to the Arduino Mega so that the robot can continue down the track. The Arduino Mini controls the servo using a PWM signal and the stepper by using a motor driver and rapidly turning on and off certain pins.

The servo is an AR-3606HB Continuous Servo, which accepts a PWM input for velocity control. A continuous servo is required since 360 degree rotation is needed to press all the buttons on the Simon Says. To ensure that the servo only moves in 90 degree increments, an 'encoder' was created using several a bump sensors. The end-effector will hit the bump sensors as it spins, allowing for control with precision down to 45 degree increments.

An Electret microphone is used to distinguish the game cues that the Simon Says game provides. Each color in the game sequence is accompanied by a distinct sound, which the MCU uses wave edge detection to identify the frequency. This setup requires an amplification circuit that provides 490x amplification, and this is accomplished through the use of operational amplifiers powered by the 5V regulated source. The MCU then samples it using its on board analog to digital converter.

## **2.3 Subsystem Requirements**

In the following sections, the requirements for the robot's subsystems are outlined. These are a few general notes that apply to the microcontrollers:

As no one single part of this project is computationally very heavy, the memory capacity and clock frequency of the microcontrollers were not taken into consideration, as most microcontrollers are more than powerful enough for the purposes of this project, especially with the distributed master/slave architecture. This has held up during testing. Therefore, specifications such as RAM, clock frequency, and similar are not listed for the MCUs. Rather, the focus is on digital and analog IO, as well as communication protocols.

Further, in this project, microcontrollers are assumed to draw negligible power compared to the rest of the system. As an example, one Arduino Uno draws roughly 45 mA of current by itself at 5 V. It would take this MCU more than 10 hours to deplete a cheap rechargeable 9V battery. Therefore, there are no microcontroller power specifications. The microcontrollers simply draw power from the regulated 6V.

### **2.3.1 Requirements Specification for Chassis**

The chassis is what keeps the robot physically together. The main formal requirement for the chassis, as dictated by the competition rules, is that the robot must fit within a 1'x1'x1' box at the beginning and end of any competition run. Thus the two manipulators mounted on the robot must be able to erect themselves on the chassis in such a way as to limit the amount of overhang they create. The main body of the robot will consist of several layers, each providing space for mounting different electronics or mechanical components.

Most of the space on the bottom layer of the robot is used to mount the motors needed for propulsion and the battery. The next layer will be used to mount the electronics for the propulsion. A third layer will

house electronics and motors for the end effectors, with the top of the robot being left almost purely as a mounting area for the end-effector for the Rubik's Arm.

Further, the chassis needs room and mounting points for all other systems of the robot.

### **2.3.2 Requirements Specification for Main Controller**

The main controller interfaces with the rest of the subsystems and is responsible for primary control of the robot. Seeing as RS232 Serial is the team's current preferred communications protocol, and there are 3 "slave" subsystems, three different RS232 interfaces are required. In order to interface with the motor encoders, 8 digital inputs are needed. The controller also needs 12 digital output pins, 4 of which are PWM capable in order to provide velocity commands to the propulsion motors. The Arduino Mega was chosen as it meets all of the above requirements.

### **2.3.3 Requirements Specification for Propulsion**

The main goal of the propulsion system is to propel the robot along the white line to each of the challenges. For the present solution to the competition, the robot requires omnidirectional movement. This will be accomplished through the use of four Mecanum wheels.

For reliable movement in any direction, a sufficient understanding of the kinematics of Mecanum wheels is required. This must be implemented in a PID control algorithm that will run on the main controller. The PID needs to be reliable enough to provide consistent motion according to the provided commands.

Four independent DC motors with encoders are required (it is impossible to achieve PID control without feedback). For power, motor drivers are needed. The motor drivers chosen for this subsystem are the Solarbotics L298n Compact motor drivers. This system will be controlled by the main controller, the requirements of which are described in section 2.3.2. The 4 propulsion motors need 1.2A continuous current total, allowing for peak current draw up to 4.5A total.

### **2.3.4 Requirements Specification for Power**

The Power System is a vital part of the robot in the sense that none of the required task cannot be done without sufficient and reliable power. Every component and device on the robot has both a voltage and current requirement. It is important that these required voltages and currents are supplied by a reliable power supply and that the power supply are able to sustain these requirements for the duration of the track run.

In order to ensure sufficient reliability by the power supplies to the system, certain precautions were taken into consideration. Because of the current spikes in motors and servos, it was advised that two separate power supplies be used for the microcontrollers/sensors and for the motors/servos. Another precaution taken is to do the power analysis on the assumption of worst case scenario. This includes using the max

time of 5 minutes for the calculation of the duration that the power supplies must supply power. This also includes using the max values of the current and voltages when calculating the amount of power needed. This will allow for sufficient enough power to the system.

#### **2.3.4 Requirements Specification for Line Following**

The line following system needs to be able to differentiate between a line segment, a corner, a branch, and the white squares that contain the challenges. The sensors involved in this system need to differentiate between black and white on a surface 0.5 inches away. At this stage, the sensor network consists a row of 8 IRR sensors, along with auxiliary edge sensors (single IRR sensors) for branch/corner detection.

The microcontroller associated with this subsystem, an Arduino Mini, has the following minimum requirements: 18 digital IO ports for interfacing with the IRR sensors, and a 9600-baud two-wire interface for RS232 Serial communication with the master controller. Potentially additional digital IO for interfacing with a gyroscope or additional IRR sensors.

#### **2.3.5 Requirements Specification for Arm 1: Etch-a-Sketch/Playing Card**

This manipulator has to have the ability to both write IEEE across the Etch-a-Sketch and pick up the playing card, reliably completing both challenges the arm is responsible for. A worm gear design, using a high torque motor, machine bolt, and two guide rails will be used to lift and lower the manipulator onto the etch-a-sketch toy.

The requirements for playing the Etch-a-Sketch is lowering the manipulator onto the game, aligning the game both longitudinally and latitudinally by driving the robot to the game, using a gripper to grab the knobs of the toy and place the game in the desired position. This data allows the robot to lower the manipulator with the motors accurately onto the knobs, and will be used to write IEEE across the screen of the toy.

In order to carry the playing card across the finish line the robot is only required to recognize the deck of cards and align the manipulator with the stack, lower the manipulator with the force that will allow the adhesive to grab the card, pick the card up with the manipulator using the worm gear screw design, and carry it across the finish line to complete the game.

A separate microcontroller will be used for this manipulator, specifically the Arduino Mini. This will allow the robot to keep the primary/secondary architecture that has been established. In this case, the Arduino Mega is the master and the Arduino Mini controlling this manipulator serves as the secondary. This microcontroller has the duty of controlling all of the motors and servos that the manipulator system uses. It has a serial connection to the master microcontroller and will perform the tasks required for each game when ordered by the master Arduino.

### **2.3.6 Requirements Specification for Arm 2: Simon Says/Rubik's Cube**

The Simon Says/Rubik's Cube arm needs to be able to interface reliably with those two challenges. A chain drive system controlled by a stepper motor is required in order to raise and lower the arm, which will provide the vertical movement that will be used to hit the buttons on the Simon Says game. This will also allow for the system to hang the manipulator on the inside of the robot to begin each round of the game. This saves space keeping the robot within the required dimensions.

For the Rubik's Cube, the requirements are as follows: the Rubik's Cube must be held in place such that only the top row has freedom to rotate, and the arm must be able to grab and twist the top row of the Rubik's Cube exactly 180 degrees. A power screw system with a custom designed end effector attached to a platform has been chosen. The gripper holds the bottom two rows while the end effector twists the top one.

In order to play the Simon Says, the subsystem must be able to properly detect the sequence, and actuate the correct buttons accordingly. The sequence can be detected by sound, light, or color, the first being the preferred method for this design. Thus, a microphone with an amplifier circuit is required. The MCU must be capable of sampling the data from the microphone using an analog to digital converter, and determine the frequency of a particular sound.

For control, in keeping with the primary/secondary architecture of this project, the subsystem requires a dedicated microcontroller. This device must have at least the following: 1 analog input port with an associated 10-bit (minimum) analog-to-digital converter, 2 digital output ports capable of providing a PWM output for control of the two servo motors, and a two-wire interface for RS232 Serial communication with the primary controller at least 9600 baud. Thus, the Arduino Mini was deemed sufficient to control this subsystem.

## **2.4 Performance Assessment**

The finished robot, design as specified, works reliably to complete all tasks.

## **2.5 Pending Design Decisions**

### **2.5.6 Custom PCB Option**

The only pending design decision currently being considered is the custom PCB. The components used in the design are spread across several printed circuit boards. One clean-up measure would be to consolidate several subsystems on a single PCB. This would reduce wire clutter and improve the overall organization of the robot.

### 3 Design of Major Components

This section outlines the various subsystems included in this project. There are seven in total, as shown in section 2. The subsystems were chosen in order to divide the engineering problem into manageable smaller pieces, but also in order to parallelize the design process as much as possible. Another prominent feature is challenge consolidation. This is why the decision fell on using 2 different subsystems for manipulation, rather than 4, to complete all the challenges.

#### 3.1 Chassis

The chassis provides the structure for every other component of the robot to attach to. It is important that the chassis is structurally sound, while providing plenty of room for everything else to attach. When designing the chassis, it is also important to consider the human factors of the design, including the aesthetics and the ability for maintenance and modification after the robot is manufactured. The biggest constraint on the chassis design is the size limit imposed in the rules. The whole robot at the start and end of the competition must be able to fit within a 1ft by 1ft by 1ft cube. To do this, while still leaving room for the wheels, the chassis will only be 7 inches long by 7 inches wide. This forces the manipulators to be designed such that when they are erect, they do not protrude from the sides of the chassis more than two inches.

To provide enough area for various components to mount, the chassis will be built in four layers, one for the motors and battery, two others for electronics, and a top layer to mount the arms.

Structurally, the chassis will be in two main parts. The main structure will be an aluminum frame that will provide structural rigidity; aluminum is light, strong, and easy to machine. The forces involved with this size of robot are not large enough to require steel for support, which is also much harder to work with. The second component of the chassis will be polycarbonate sides and polycarbonate “shelves” for the various components to sit in. The siding does not have to be structurally rigid, and mainly serves a decorative purpose, improving the aesthetics. Clear acrylic will be purchased, showing off the team’s electronics and wiring handiwork. For the “shelves” the acrylic will have some structural requirements, and acrylic was especially chosen to be just structurally sound enough to serve this role.

#### 3.2 Main Controller

The main controller serves as the hub of communications between the different secondary microcontrollers in the robot. RS232 Serial is the preferred communications protocol, which requires three different RS232 UART interfaces on the microcontroller, one for each subsystem. Communication between the Primary microcontroller and the Secondary microcontrollers is encoded using the Consistent Overhead Byte Stuffing (COBS) Algorithm. This was done to prevent error due to noise, and allow the encoding of any value without having to add additional escape sequences for values that match the header byte. The COBS Algorithm only adds one extra byte per 255 bytes.



In order to interface with the motors, a total of 20 Digital I/O pins are required. 4 of the pins are PWM pins for velocity control of the motors, 8 pins configure the direction of the motor, and another 8 are used for quadrature counting. The Arduino Mega was chosen as it meets all of the above requirements.

**Table 3.1 Arduino Mega 2560 Specifications**

| Parameter              | Value |
|------------------------|-------|
| Operating Voltage      | 5V    |
| Input Voltage          | 7-12V |
| Input Voltage Max      | 20V   |
| Digital I/O Pins       | 54    |
| PWM Pins               | 14    |
| Analog input Pins      | 16    |
| DC Current per I/O Pin | 40mA  |
| UARTs                  | 4     |

The Arduino Mega oversees one particular subsystem: the propulsion subsystem. Details on how the Arduino Mega interfaces with this subsystem will be listed in the following section. The Arduino Mega is also required to sense when the heat begins. It uses a photoresistor in pull down configuration connected to an analog input to sense when the robot is required to begin the heat. The Mega will be switched on via a manual switch on the board, and will begin polling the photoresistor. Since the photoresistor is in pull down configuration, the MCU will measure close to zero volts at the analog input port when the LED is shining. Once the LED shuts off, the analog input port will measure 5V, and the robot will change states into 'Line Following State', begin down the track, and cease polling the photoresistor.

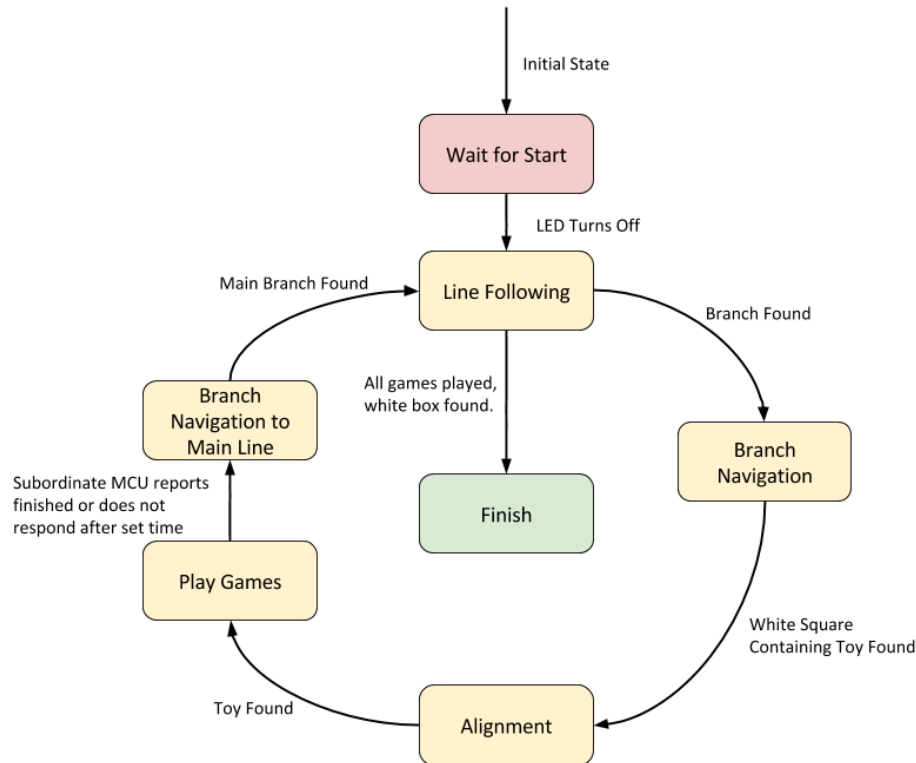
As such, the Arduino Mega is required to determine the overall state of the robot. A list of states and their descriptions can be seen in Table 3.2. Figure 3.1 details the overall state machine. The Arduino Mega is also required to communicate with its several subsystems to complete each task. A dedicated microcontroller tasked with polling the line sensors was implemented to reduce the amount of computations the Arduino Mega is required to do, as it already undertakes several time-sensitive functions. The two subordinate MCUs that control each of the arms are implemented to increase the

robustness of the system. If either subordinate microcontroller ceased functioning and could not play a particular game, then the Arduino Mega can continue down the track and finish the rest of the heat.

**Table 3.2: List of Robot States**

| State             | Description  |
|-------------------|--|
| Wait For Start    | This is the robot's initial state after being powered on. The robot will continuously poll the photoresistor until the start LED is turned off, upon which it will move forward onto the line and transition to Line Following.  |
| Line Following    | The Arduino Mega requests data from the Secondary Line Following MCU. The Line Following MCU polls the sensors and determines the orientation of the robot on the line and the existence of branches. The Line Following MCU then provides a direction vector that the robot must correct in. The Mega then computes the required velocities for each wheel and moves in that direction. The Secondary MCU also notifies the Mega about branches, which will transition to Branch Navigation State. Once the robot has finished playing 4 games, and it finds a white patch, it will stop in the Finish State. |
| Branch Navigation | The Secondary MCU has determine that there is a branch from the main line leading to a toy, and has commanded the robot to follow the branch towards it. Once the robot gets over the white box enclosing the toy, the robot transitions to Alignment.   |
| Alignment         | The order of the games is known a priori, so the robot knows which side is required to line up with the toy. The robot then extends the required manipulator and uses IR LED/Detector pairs to   |

|                                |   |
|--------------------------------|---|
|                                | determine whether the toy is within its grasp. When it is, it transitions to Play Game mode.  |
| Play Game                      | As mentioned above, the order of the games is known a priori, so the robot knows which game is required to play. The robot signals the correct subordinate microcontroller to play the game once aligned. The Mega will wait for a set time for the subordinate microcontroller to finish. If the microcontroller does not finish within that set time, it is considered to have failed and the robot will continue the heat. Ideally, the subordinate microcontroller will report when it is finished to the Mega. |
| Branch Navigation to Main Line | The robot will back up until it is over the line. It will then use the same line following algorithm to navigate to the main line and enter Line Following.   |
| Finish                         | The robot enters this state once it has played 4 games, and then comes across a full white square indicative of the finish line. The robot will then cease motion.  |



**Figure 3.1: FSM diagram for the robot**

### 3.3 Propulsion

Omni Directional control was desired in this project, to aid in alignment of the toys and following perpendicular lines and branches. Thus the Mecanum Omni Directional Wheel platform was chosen for this project, which requires 4 motors to individually actuate each wheel. The Mecanum wheels can be seen in Figure 3.2. The motors chosen can provide 8 kg-cm of torque and run at 350 rpm, which is adequate for movement. These motors can be seen in Figure 3.3. They have a free run current of 300mA, which means that 1.2A are required to run all these motors. To meet this requirement, the Solarbotics L298n Compact motor driver was chosen, as it is a dual motor driver that can provide up to 4A of output current, which can be seen in Figure 3.4. The pins in use, described in the following paragraphs may be found in Table 3.3.

The Arduino Mega requires 4 PWM pins to provide velocity control for each of the motors. Each motor takes a PWM input in addition to two additional inputs which determine the direction in which to spin in. This functionality is provided by two additional Digital I/O pins hooked up to the motor driver, which sums to 3 pins for speed and direction per motor. The two direction pins specify the direction of the motor by toggling the internal H-Bridge on the motor driver. If both pins are set high or low, the motor is forced

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

to brake. If one pin is high and the other is low, the motor will turn rotate in a particular direction - clockwise, for example. If the polarities of the pins are reversed, then the motor will rotate in the opposite direction - counter-clockwise.

The Arduino Mega performs quadrature decoding and counting for each of the four motors. Eight Digital Input pins are used to sample the outputs of the Encoder A and B lines for each motor. The encoder lines of each motor are connected to a set of pins which are configured to trigger a single pin change interrupt. The MCU runs a piece of code that performs quadrature counting every time a logic change is seen on those particular pins. The code used can be seen in Listing 3.1. The encoder counts are used as feedback for the motor PID algorithm.

**Listing 3.1: Quadrature Decoding/Counting code**

```
//port K interrupt vector
ISR(PCINT2_vect) {
    static const int8_t rot_states[] = //lookup table of rotation states
    {0, -1, 1, 0, 1, 0, 0, -1, -1, 0, 0, 1, 0, 1, -1, 0};
    static uint8_t AB[NUM_MOTORS] = {0x03, 0x03, 0x03, 0x03}; //encoder
AB status
    uint8_t status = PINK; // read port status

    for (int i = 0; i < NUM_MOTORS; ++i) {
        // check for rotary state change button1
        AB[i] <<= 2; // save previous state
        AB[i] |= (status >> 2*i) & 0x03; // add current state
        motor_encoders[i] += rot_states[AB[i] & 0x0f];
    }
}
```

**Table 3.3: Pinout for Primary Arduino Mega**

| Function          | Pin |
|-------------------|-----|
| Motor 0 PWM       | 4   |
| Motor 0 Input 1   | 22  |
| Motor 0 Input 2   | 23  |
| Motor 0 Encoder A | 2   |
| Motor 0 Encoder B | 28  |
| Motor 1 PWM       | 5   |
| Motor 1 Input 1   | 24  |
| Motor 1 Input 2   | 25  |
| Motor 1 Encoder A | 20  |
| Motor 1 Encoder B | 26  |
| Motor 2 PWM       | 6   |
| Motor 2 Input 1   | 51  |
| Motor 2 Input 2   | 50  |
| Motor 2 Encoder A | 21  |
| Motor 2 Encoder B | 48  |
| Motor 3 PWM       | 7   |
| Motor 3 Input 1   | 53  |
| Motor 3 Input 2   | 52  |
| Motor 3 Encoder A | 18  |
| Motor 3 Encoder B | 46  |

|                              |    |
|------------------------------|----|
| Start LED                    | A0 |
| Line Following Serial Tx     | 14 |
| Line Following Serial Rx     | 15 |
| Etch-A-Sketch/Card Serial Tx | 16 |
| Etch-A-Sketch/Card Serial Rx | 17 |
| Simon/Rubik's Tx             | 20 |
| Simon/Rubik's Rx             | 19 |

A velocity PID was implemented that runs in a 200 Hz timer interrupt. The timer interrupt ensures that the algorithm will be computed at a fixed time step, and eliminates the time dependence (delta time) of the PID algorithm. The PID algorithm outputs the required PWM, while using the encoder data as feedback. This code can be seen in Listing 3.2. The forward kinematics for the system were derived and mimicked in code. As a result, a vector that contains the velocity in the x and y direction and the angular velocity of the whole system can be specified, and the MCU will compute the required velocities of each individual wheel.

**Listing 3.2: PID Algorithm and Interrupt**

```
//interrupt handler for the timer compare
ISR(TIMER1_COMPA_vect) {
    float current_error;

    for (int i = 0; i < NUM_MOTORS; ++i) {
        motors[i].encoder_value = motor_encoders[i];

        //calculate new position based on velocity
        motors[i].command_position += SAMPLE_TIME *
motors[i].command_velocity;

        //calculate PID
        current_error = motors[i].command_position -
motors[i].encoder_value;
        fixedUpdatePID(motor_pid_data[i], current_error);
    }
}
```

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

```
//pwm is absolute value of output
motors[i].pwm = round(fabs(motor_pid_data[i].pid_output));
motors[i].pwm = constrain(motors[i].pwm, 0, 255);

//if output is < 0 switch directions
if (motor_pid_data[i].pid_output < 0)
    setMotorDirection(motors[i], DIRECTION_1);
else
    setMotorDirection(motors[i], DIRECTION_2);

//write to pwm
analogWrite(motors[i].pwm_pin, motors[i].pwm);
}
} //end interrupt handler

//fixed update PID is meant to be called at constant time intervals,
//therefore it does not need delta_t
void fixedUpdatePID(pid_data &pid, const float &current_error) {
    float error_differential = 0;

    pid.integral_error += current_error;
    pid.integral_error = constrain(pid.integral_error, -
pid.integral_guard,
    pid.integral_guard);

    error_differential = (current_error - pid.previous_error);

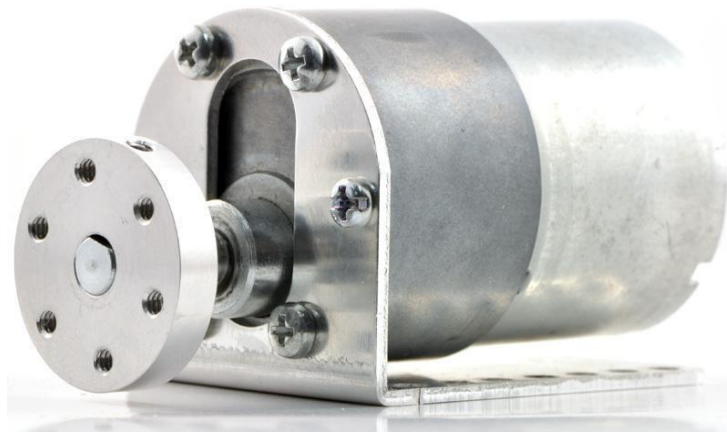
    pid.pid_output = (pid.proportional_gain * current_error) +
        (pid.integral_gain * pid.integral_error) +
        (pid.derivative_gain * error_differential);

    pid.previous_error = current_error;
} //end pidControl()
```

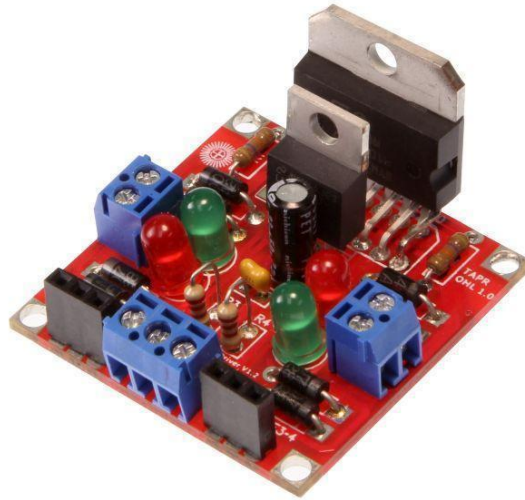




**Figure 3.2: Mecanum Omnidirectional Wheel**



**Figure 3.3: Pololu 30:1 Motor**



**Figure 3.4: Solarbotics L298n Motor Driver**

### 3.4 Power

For the first developed design, the engineers broke up the robot into two different sections in order to efficiently provide power; the first section being the propulsion. The propulsion system will have an independent battery because the motors for propulsion draw more current than any other motor in the robot. The other battery will be used to power the rest of the subsystems and manipulators. The currently implemented design takes details of the power consumption of the robot into consideration. When the robot is navigating the white lines, it uses the propulsion system and when it is using the manipulators it is using a different system. The robot is designed so these two power systems don't get used at the same time, therefore one battery with enough capacity will be able to handle running the robot.

#### 3.4.1 Power Analysis

**Table 3.4: Power Analysis**

|              | Quantity | Max Voltage | Average Current | Peak Current  | Average Power | Peak Power |
|--------------|----------|-------------|-----------------|---------------|---------------|------------|
| Motors       | 2        | 6V          | 40 mA(free-run) | 0.7 A (stall) | 240 mW        | 4.2 W      |
| Motor Driver | 1        | 5.5 V       | 1.2 A           | 3.2 A         | 6.6 W         | 17.6 W     |

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

|                                    |   |        |        |     |         |        |
|------------------------------------|---|--------|--------|-----|---------|--------|
| 180 degree Servo                   | 1 | 6 V    | 150 mA |     | 0.9 W   |        |
| Pololu Motors                      | 4 | 12 V   | 300 mA | 1 A | 3.6 W   | 12 W   |
| Motor Drivers                      | 2 | 14.8 V | 600 mA | 4 A | 8.88 W  | 59.2 W |
| Power HD continuous rotation servo | 1 | 6 V    | 900 mA |     | 5.4 W   |        |
| Total:                             |   |        | 5.13 A |     | 45.54 W |        |

The power of electronics such as sensors and microcontrollers is considered to be negligible, it will not be added to the power analysis. These systems will be powered by a 9 V D-type battery.

### 3.4.2 Battery Type

The engineers had to decide the correct type of battery from many different types that are out on the market. Some of them include but are not limited to Lithium Polymer (LiPo), Lithium Iron Phosphate (LiFePo), Nickel Zinc (NiZN), and Nickel Metal Hydride (NiMH). The categories that were the most important in choosing a battery were: cost, weight, capacity, and size. After carefully considering all the options the engineers could choose from, it was decided that the best fit for robot was a LiPo battery. The LiPo chosen for the propulsion was a Rhino 2150 mAH 20C LiPo Pack. The battery is composed of 4 cells each of 3.7 V, for a total voltage of 14.8 V. The constant discharge of the battery is 20C with a burst rate of 30C for 15 seconds. The dimensions of the battery are 113x35x27 mm, the weight is 215 g.



Figure 3.5: Rhino 2150 High Discharge Li-Po Battery

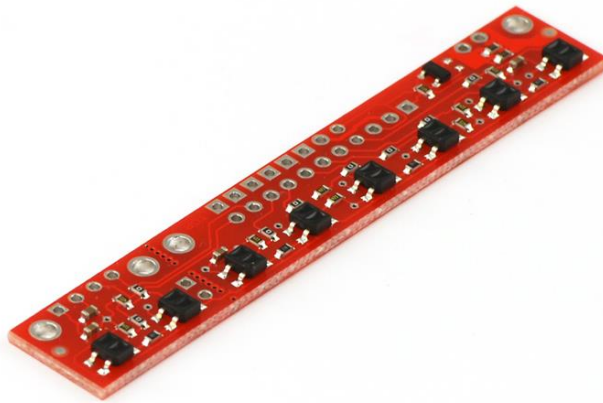
### 3.5 Line following

As part of the competition, the robot will need to navigate according to a white line on a black surface. The “traditional” way to accomplish this in a small robotics setting such as the present is by using infrared reflectance (IRR) sensors. These are able to distinguish between black and white because the amount of infrared light that is reflected back to the source depends on the color of the surface on which it is incident. This information is converted to an analog or a digital signal that can be read by a microcontroller. As there was no need to reinvent the wheel, this approach was chosen for this project as well. The IRR sensors are arranged in a pattern such that the direction in which the robot should travel can be detected by reading the sensor data.



**Figure 3.6: IRR Sensor Breakout Board**

There are several commercially available pre-constructed arrays of IRR sensors, such as the Pololu QTR-8RC pictured below. Buying an off-the-shelf array presents the first of two available options.



**Figure 3.7: Pololu QTR-8RC Reflectance Sensor Array**

The second option was to construct a sensor array from scratch. This approach was pursued initially, but was found to be impractical. The current line following system uses a front-mounted QTR-8RC Sensor Array with several stand-alone IRR sensors at the edges of the robot.

These sensors will be polled by a dedicated Arduino Mini Pro, which aggregates the sensor data, makes navigation decisions, and transmits these over RS-232 Serial to the main microcontroller. The full specifications of the Arduino Mini can be found in Table 3.5, and the pins in use can be found in Table 3.6. This removes a lot of computation from the main controller. The transmission will occur “on demand,” meaning that the main controller will request navigation information from the line following subsystem, and only then proceed to transmit data. The data will be in the form of an X- and a Y-value corresponding to directional velocity along the two axes.

**Table 3.5: Arduino Mini Specifications**

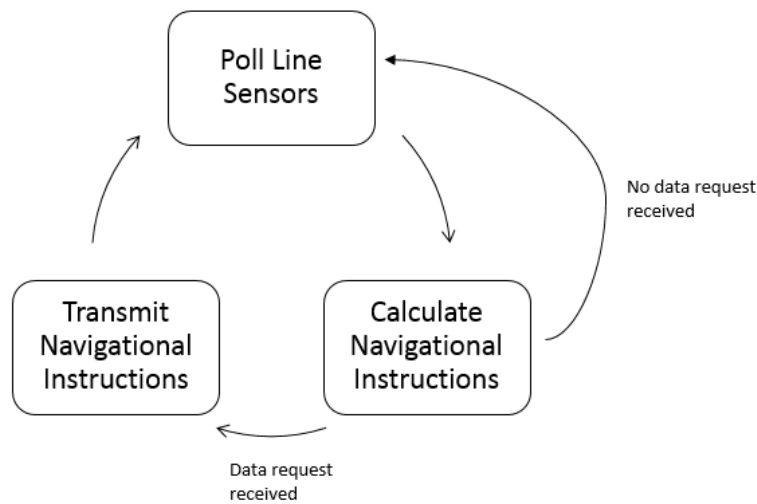
| Parameter              | Value |
|------------------------|-------|
| Operating Voltage      | 5V    |
| Input Voltage          | 7-9V  |
| Digital I/O Pins       | 14    |
| PWM Pins               | 6     |
| Analog input Pins      | 8     |
| DC Current per I/O Pin | 40mA  |
| UARTs                  | 1     |

**Table 3.6: Pins in use on Secondary Arduino Mini**

| Function              | Pin |
|-----------------------|-----|
| Primary MCU Serial Tx | Tx0 |
| Primary MCU Serial Rx | Rx0 |
| Row IRR Sensor 0      | 2   |
| Row IRR Sensor 1      | 3   |
| Row IRR Sensor 2      | 4   |
| Row IRR Sensor 3      | 5   |
| Row IRR Sensor 4      | 6   |

|                        |    |
|------------------------|----|
| Row IRR Sensor 5       | 7  |
| Row IRR Sensor 6       | 8  |
| Row IRR Sensor 7       | 9  |
| Auxiliary IRR Sensor 0 | 10 |
| Auxiliary IRR Sensor 1 | 11 |

The code that is implemented to accomplish this behavior is summarized on the flowchart below.



**Figure 3.8: Line Sensor Logic Flowchart**

The current state of this block is one of continued testing. The interface with the main controller works reliably, and navigation at corners and along straight lines have both been accomplished. Reliable branch navigation is a work in progress.

Further improvements may include, as deemed necessary, the inclusion of additional IRR sensors in order to improve resolution at corners and branches, as well as a gyroscope in order to help the robot stay aligned. The latter would be controlled by the subsystem microcontroller, and angular information would be submitted as an extra data point to the main controller along with the X- and Y-velocities.

The code used to poll the IRR sensors individually is included below:

**Listing 3.3: IRR Sensor Code**

```
int readQD(int QRE_PinNum){
    //Returns value from the QRE1113
    //Lower numbers mean more reflective
    //More than 3000 means nothing was reflected.
    int diff = 0;

    pinMode( QRE_PinNum, OUTPUT );
    digitalWrite( QRE_PinNum, HIGH );
    delayMicroseconds(10);
    pinMode( QRE_PinNum, INPUT );

    long time = micros();

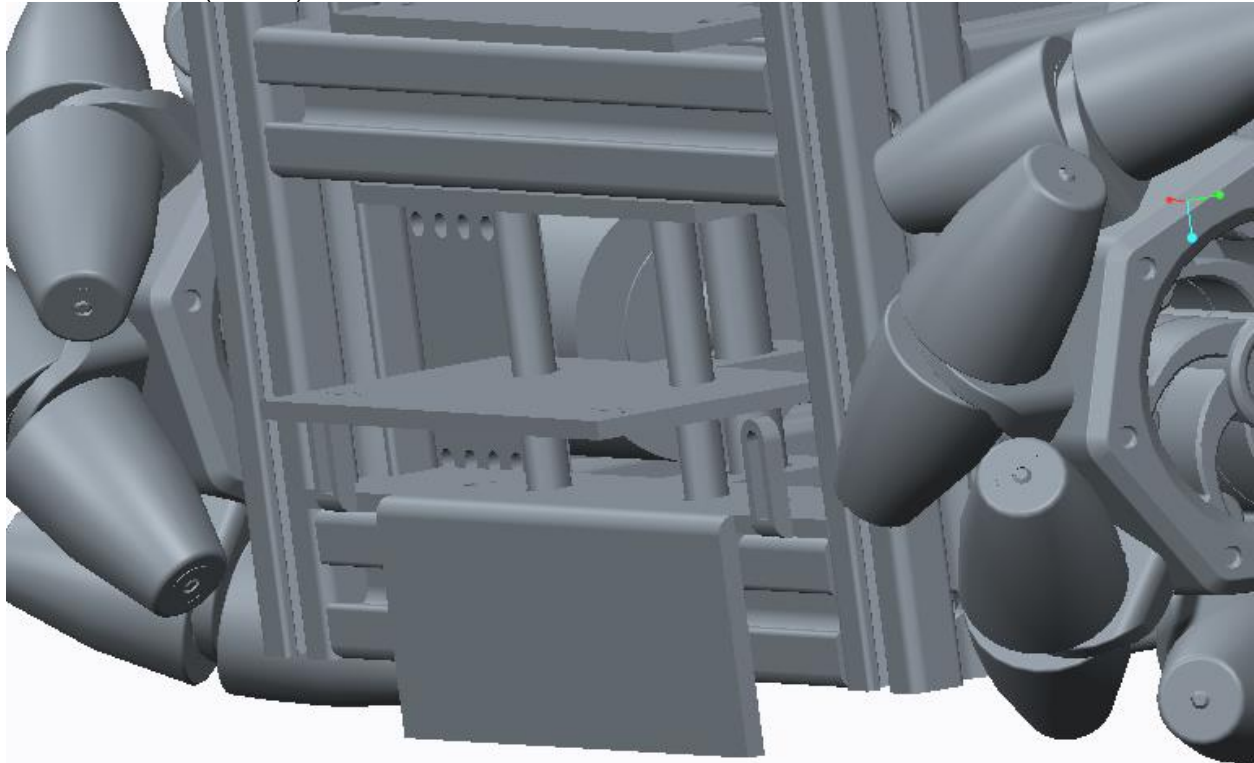
    //time how long the input is HIGH, but quit after 3ms as nothing
    happens after that
    while (digitalRead(QRE_PinNum) == HIGH && micros() - time < 3000);
    diff = (micros() - time);

    return diff;
}
```

### 3.6 Arm 1: Etch-a-Sketch/Playing Card

Both the Etch-a-Sketch and Playing Card challenges require manipulation up and down (in the z-axis). Because of this symmetry, it was decided that both challenges could be solved through the use of a single arm. As stated in the previous milestones, the general design is to use a sticky surface, mounted to motors, which will use high friction to manipulate the toys. It has proven challenging to align the toys and the arm properly. Understanding this, the fabrication of a more forgiving arm with semi-self-aligning functionalities has become the priority. Whereas navigational alignment is difficult, building a self-aligning manipulator somewhat ameliorates this problem. A gripper design using two servos with fabricated attachments are used to grip both knobs of the etch-a-sketch toy and align the game against the back plate with exact position. A worm gear design for the manipulator is used to interface the motors with the knobs of the toy. By approaching the knobs of the toy in a downward motion the robot can reliably assure that the knobs are interfacing with the toy directly. The worm gear system works by using a high torque motor to spin a screw attached through the end effector. Guide rails are used through the end effector to ensure that the manipulator responds vertically due to relative motion instead of in a circular motion. This allows the manipulator to travel along the z-direction direction. This increases precision. Pictures of the current prototype of the arm are below.





**Figure 3.9: ProE Model of the proposed End-Effector Assembly**

The general algorithm used for this design is as follows. First the robot will arrive at the toy and run the toy into the back plate. The robot will then use the gripper system to grip the knobs of the toy and align the toy with the back plate. The manipulator will then be lowered until the motors are aligned accurately with the knobs of the toy. The motors will then be used to complete the challenge of writing IEEE onto the screen of the toy. After completing the challenge the manipulator will be pulled up and the robot will continue to the rest of the challenges.

The picking up of the card shouldn't impose much more difficulty, though the manipulator has been altered from the original plan. It is still the intention of the team for the robot to approach the stack of cards, lower the manipulator with the adhesive between the two motors used for the Etch-a-Sketch game, and pick up the card to be carried across the finish line.

To control the geared motors and the drive motor on the worm gear, the robot uses an Arduino Mini which connects back to the Main Controller to receive the initial start signal. Several Arduino functions

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

are used for controlling the motor attached to the worm gear, as well as the motors. To control the servos on the gripping system, the Arduino Servo Library is used.

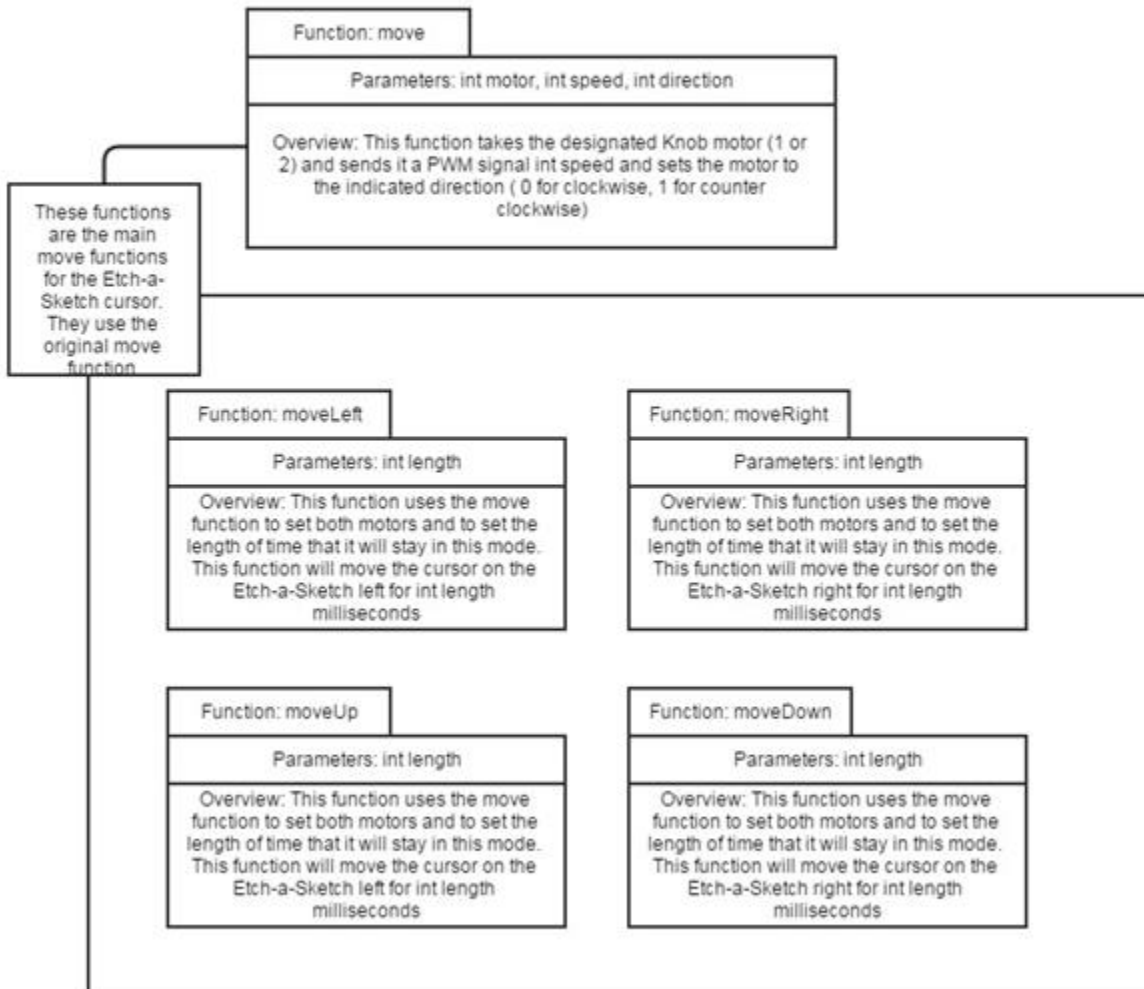
In the following table are the pin assignments for this subsystem:

**Table 3.7: Arm 1 Pin Assignments**

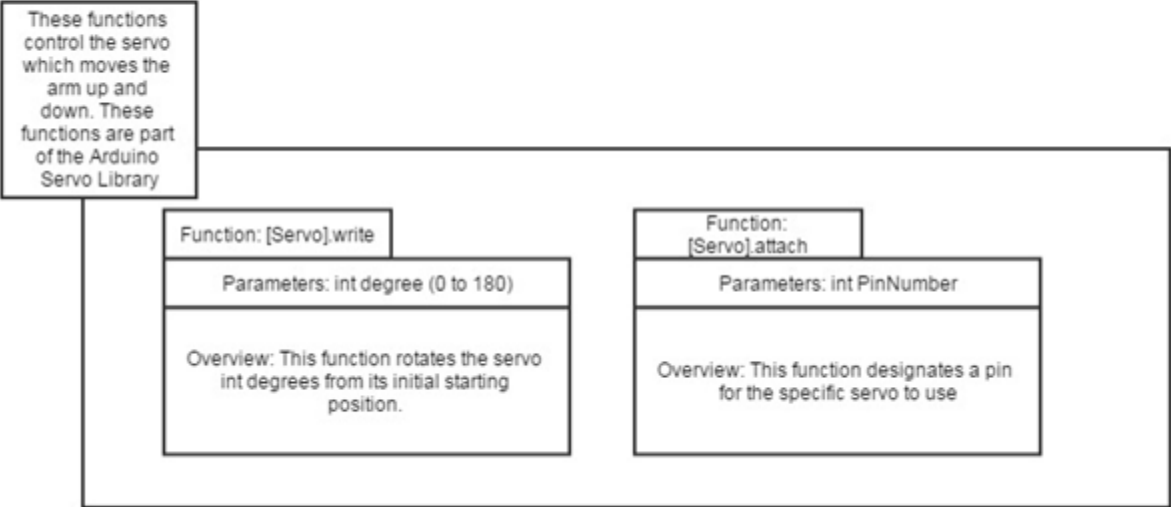
| Function                          | Pin   |
|-----------------------------------|-------|
| Primary MCU Serial Tx             | Tx0   |
| Primary MCU Serial Rx             | Rx0   |
| Arm Servo Control                 | 14    |
| Horizontal Knob Motor Speed (PWM) | 3     |
| Vertical Knob Motor Speed (PWM)   | 5     |
| Horizontal Knob Motor Direction   | 8,9   |
| Vertical Knob Motor Direction     | 11,12 |

For the Etch-a-Sketch challenge, all controls of the knob motors and servo are hardcoded to generate the letters using open-loop control.

Here is a general overview of all the functions that are used in the code:



**Figure 3.10: Movement Function Table**



**Figure 3.11: Arm Servo Function Table**

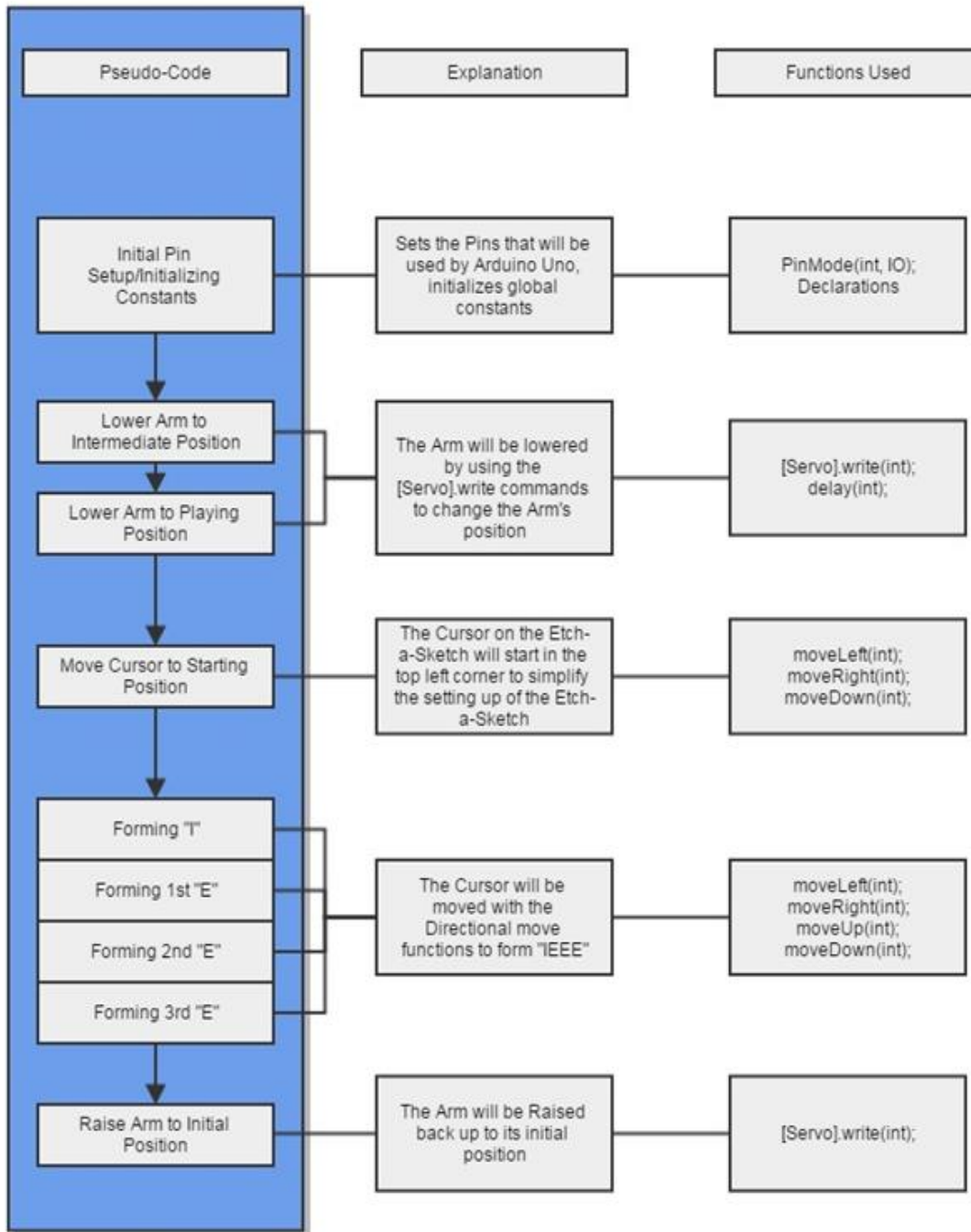


Figure 3.12: Etch-A-Sketch Arm Program Flow Chart

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

Currently, much testing has already been done for this arm and the team feels confident enough to move on from the prototype design into a more precise and more permanent model.

The tests for the Geared Motors for spinning the knobs of the Etch-a-Sketch have been successful using a hub and double-sided tape. Also the test for the worm gear system to lift the end effector was successful. The test for the preliminary version of the Etch-a-sketch code was successful as well, by simulating the perfect alignment with the Motors perfectly placed and stuck on the knobs although the testing was done at 4.5V and the delay factor (constant which controls duration of all delays in the code) will have to be adjusted if a different voltage is used.

With the first prototype design arm (not shown) the test for controlled descent of the arm onto the Etch-a-Sketch and operation of the knobs (a full operational test) was not successful as the axis' of rotation from the knob spinning motors to the knobs did not match up. The team concluded that the first prototype design was not constructed well enough for the precise alignment that was needed for the test.

The second prototype model (not shown) was made to test the idea of a framed alignment system (consisting of the back plate and wing pieces), to finalize all the dimensions of the arm in relation to the actual height of the robot and to give a more precise and well-constructed model to spot potential problems. Though this arm worked slightly better than the first prototype; interfacing with the etch-a-sketch knobs at an angle was a very fragile design because the motors and the knobs have to have the same axis of rotation for the design to work correctly, and the team could not get this axis of rotation lined up correctly using the arm.

The most recent prototype model was made based off of the importance of interfacing the knobs and the motors accurately. This is done by simply using a worm gear system where the major screw is connected to the external motor. When the motor is turned the end effector falls and rises along the guide rails placed to prevent the end effector from turning with the motor. This concept of the system has proved to work.

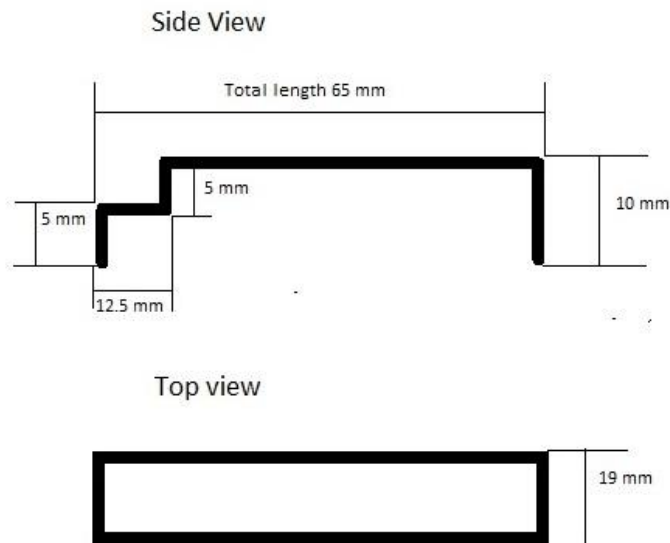
The major concerns with this design revolve around the amount of force that this system provides. Because the double sided tape will be used as the primary interface between the motors and knobs it is important that the system have enough force for the tape to stick. If this is not proven through testing alternatives must be assessed. These alternatives include adding dead weight to the manipulator, and using a different adhesive for interfacing. The alignment issues from the previous designs have been improved immediately due to the gripper subsystem and the vertical interfacing of the motors and the knobs.

The team has decided to continue on with a more permanent model that will have precisely crafted pieces to determine whether an optimally crafted prototype can be built to do the complete challenge. This full operational test of the prototype manipulator will determine if a change in the adhesive, or a heavier end effector is needed.

### 3.7 Arm 2: Simon Says/Rubik's Cube

With two of the competition's challenges being covered by the first arm, the second arm will complete the remaining two: Simon Says and the Rubik's Cube. What these games have in common is that they both require 180 to 360 degree rotational movement in the horizontal plane. For the Rubik's Cube, exactly one row needs to be rotated 180 degrees. For the Simon Says game, the four colored buttons are located at 90 degree intervals in a circle. Therefore, unless four separate button actuators are used, the actuator must be able to move at least 270 degrees to reach every button.

In order to integrate solutions to both these challenges into a single system, a custom manipulator was designed as can be seen below.



NOTE: Not to scale

**Figure 3.13: Custom Rubik's Cube/Simon Says Manipulator**

This system allows the inner notch to touch the buttons of the Simon Says game, whereas the outer ends are far enough apart to grab onto and twist the top row of the Rubik's Cube. Vertical motion (in order to

push the Simon buttons or touch the Cube) will be provided by lowering and raising the arm with which the servo is attached to the body of the robot.

The motion is provided by an AR-3606HB continuous rotation servo rated for 93oz-in of torque at 6 V. These specifications were appropriate for the design because the torque is greater than what is required by a Rubik's cube by a safe margin, speed is not a concern, and price varies little between different models in this class of servos.



[www.pololu.com](http://www.pololu.com)

**Figure 3.14: Continuous Rotation Servo**

In order to provide accurate motion in 90-degree steps, a rudimentary motor encoder was created using the same model IRR sensor as described in section 3.5. Four white patches were placed at 90-degree intervals around the manipulator, and the IRR sensor was affixed to the level part of the servo, facing downwards. Thus, every time the sensor input goes from “black” to “white,” 90 degrees of rotation have occurred. This system was tested, and it works reliably.

There were multiple ideas for delivering the end effector onto both games successfully. The first was a chain drive driven by a large stepper motor. This idea was scrapped because it lacked mechanical reliability. The second was a custom made arm consisting of 2 extension and angling servos and the end effector servo; however this idea was scrapped for the same reason because it proved to be quite fragile when implemented. The third and final design was a power screw design almost identical to the Etch-A-Sketch arm. This allowed for the end effector and rotation servo to be lowered onto each game relatively straight and able to push down on the game when needed. The simplicity, robustness, and alignment of this design was the reason for its continued use.



EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

The Rubik's Cube will be held in place by two pincers that protrude from the chassis. The distance between these is less than the diagonal of the Rubik's Cube to prevent the entire cube from rotating while the manipulator is turning the top row of the cube. The pincer mechanism is actuated by two servos that, when not in use, rest in a flared out position that makes the unit relatively flush with the exterior of the chassis and keeps the whole robot within the 1ft x 1ft x 1ft size constraint at the beginning and end of the round. Each arm of the mechanism is tall enough to cover the bottom two rows of the cube, preventing them from turning, and have a protrusion at the end that further prevents the entire cube from rotating when being manipulated by the robot.

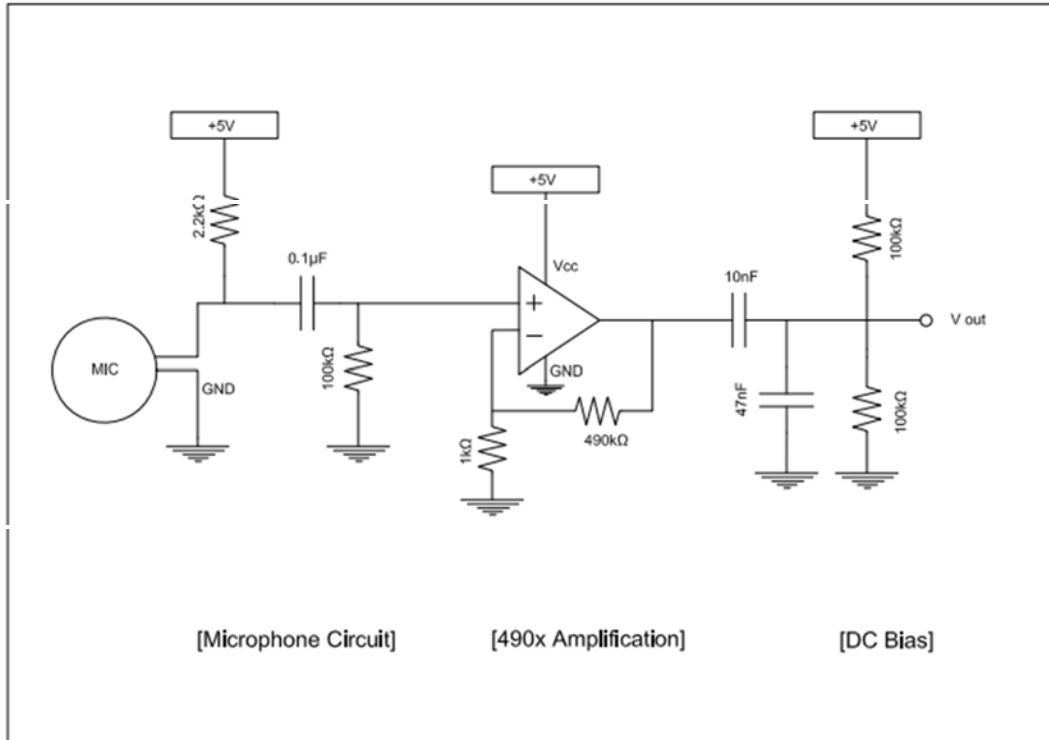
Since the Simon Says and the Rubik's Cube are played on the same side of the robot, the pincer must be able to hold both games. The problem with this is that the Simon and the Rubik's Cube are different sizes and heights. The implemented design allows for the pincer to hold the cube efficiently and the protrusion on the end of each arm fits over the clip portion of the Simon Says.

For detecting the Simon Says' sequence of buttons, several systems were considered. It was decided that light and color would not be reliable enough for the competition environment, and thus sound detection was chosen. The current system includes an Electret microphone as shown below. This is included in an amplifier circuit, and a DC bias, so as to make its output readable by a microcontroller. The microcontroller performs frequency counting and constructs the proper sequence.



**Figure 3.15: Electret Microphone**

# Microphone Circuit



**Figure 3.16: Microphone Amplification/Biasing Circuit**

For the sound detection scheme, a bank of digital band pass filters was used to filter out the desired frequencies and remove as much noise as possible. The desired frequencies were as follows:

**Table 3.8: Simon Says Sound Frequencies**

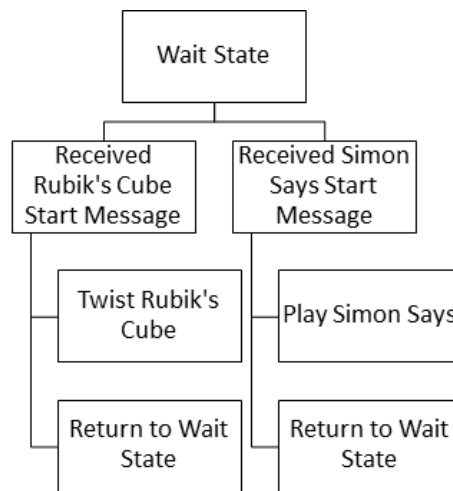
|        |               |
|--------|---------------|
| Blue   | 348Hz & 90Hz  |
| Green  | 422Hz & 279Hz |
| Red    | 329Hz         |
| Yellow | 210Hz         |

Because of microcontroller constraints, the sampling frequency was 1kHz; the 90Hz peak for blue was actually higher than what the system could construct without some form of aliasing. This occurrence was

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
 ECE Team 1A (ME 29)

taken advantage of by detecting the aliasing at 90Hz and the other peak at 3448Hz. Once all of the bandpass filters were constructed, the system was tested and was proven to be roughly 75% accurate under the correct conditions.

The control for this system is performed by a dedicated Arduino microcontroller. This will be responsible for controlling the servo and the arm to which it is attached. It will wait until it receives an instruction to play a game by the main controller. Then, it will enter one of two states, depending on whether it's time to play the Simon Says game or twist the Rubik's Cube. The subsystem logic is outlined in the flowcharts below.



**Figure 3.17: Arm 2 High Level State Overview**

In the following table are the pin assignments for this subsystem:

**Table 3.9: Arm 2 Pin Assignments**

| Function                | Pin |
|-------------------------|-----|
| Primary MCU Serial Tx   | Tx0 |
| Primary MCU Serial Rx   | Rx0 |
| Microphone Analog Input | A0  |
| Servo PWM output        | 10  |
| IRR Sensor Encoder I/O  | 9   |

This subsystem is currently in a state of continued testing. The servo successfully rotates a single row of the Rubik's Cube, and the custom-designed manipulator successfully pushes each button.

The plans to reuse the design from Arm 1 was reevaluated when it was realized that the design was not ideal due to spacing on the chassis. In order to fit in the required space it was realized that the manipulator for this arm must have the capability to start the round inside the robot, and come outside the chassis as the system is needed to complete the challenges. This led to the idea of the chain drive system which is used to carry the end effector to the outside of the chassis to perform the challenges and back on the inside of the chassis to end the round within the required dimensions.

## 4 Test Plan

In broad strokes, the test plan for this project consists of four main phases:

The initial component test consists of testing each newly bought component as it comes in, in order to reduce the possibility of error stemming from faulty parts. This is done continuously and is not described in detail in this section.

The second phase is testing each subsystem independently. This means making sure the subsystem works as desired by itself, taking overall system integration out of the equation. The third phase is system integration testing. At this stage, each system is integrated into the robot and tested as part of a whole. This lets the team iron out any inter-system issues. The final phase is the full run. Each system will have been tested by itself at this point, as well as mounted onto the robot.

A tabular summary of all tests, as well as current status, can be found in Section 4.3.

### 4.1 System and Integration Test Plan

For the system, the most important test is that the system works well as a whole. Each of the subsystems will be mounted to the robot and tested in turn. Once each subsystem works well when integrated with the robot, the focus will be on making a full run be reliable. The system integration and system test plans are available as forms in the appendix, and are omitted here for formatting reasons. Please see the next section for examples of test forms.

### 4.2 Test Plan for Major Subsystems

Each subsystem will be tested, both by itself, and after integrated with the robot. In this section, two examples of test forms, one completed and one pending, are included as subsections below. Please see the forms in the appendix, as well as the table in Section 4.3, for overall testing status.

### 4.2.1 Completed Test Example

#### Scheduled Test Reporting Form

|  |                   |
|--|-------------------|
| Test: Rubik's Cube: Torque Test              |                   |
| Tester Name: Nils Bjerer, / Julian Velasquez |                   |
| Date: 10/1/2014                              | Test Result: Pass |
| Time: 6:30 PM                                | Notes: Ref: Video |
| Location: ME Senior Design Room              |                   |

#### Test Objective:

It had been determined early on that a high amount of torque is required to turn a row on a Rubik's Cube. This test is designed to make sure that the continuous rotation servo together with the end-effector have enough torque to complete this task. The servo will also need to be able to turn the row 180 degrees.

#### Test Description/Requirements:

The Rubik's Cube's bottom two rows will be held in place while the end-effector is placed on the top row. The servo will then turn the top row 180 degrees.

#### Success Condition:

The top row of the Rubik's Cube will be turned 180 degrees while the rest remains stationary.

#### Results:

Originally, this test failed - the torque of the servo was high enough, but the team was unable to make the servo move 180 degrees reliably. As this was a continuous rotation servo, there is only open loop control of position in terms of speed. A simple encoder was assembled out of a leftover IRR sensor that fixed the problem (see video). After this fix, the test was passed.

#### Reason for Failure:

Originally - lack of control over CR Servo.

#### Recommended Fix:

Home-made encoder - Implemented.

### 4.2.2 Pending Test Example

#### Scheduled Test Reporting Form

|                      |                      |
|----------------------|----------------------|
| Test: Full Run       |                      |
| Tester Name:         |                      |
| Date:                | Test Result: Pending |
| Time:                | Notes:               |
| Location: Test Track |                      |

**Test Objective:**

Have the robot autonomously complete a full run of the course, successfully completing all four challenges.

**Test Description/Requirements:**

The robot will start at the right time (when the red LED turns off), and follow the line to each challenge. It will complete each challenge as per the rules. After the last challenge, the robot will navigate to the finish line and stop.

**Success Condition:**

The robot finishes the track, completing all four challenges. The maximum amount of non-time points are obtained.

**Results:**

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

This test requires all subsystems to be complete before it can be performed completely.

### 4.3 Summary of Test Plan Status

The project is now approaching integration testing. Most of the subsystems are either finished, or near finished. The focus between the time of writing and the internal school competition on March 20th will be to make sure the robot operates well as a coherent system. Please refer to the following table for an overview of completed and pending tests. All the corresponding forms can be found in the appendix.

**Table 4.1**

| Test                                       | Outcome |
|--|---------|
| Simon Says: Sequence Recreation (software) | Pass    |
| Simon Says: Sequence Recreation (hardware) | Pass    |
| Rubik's Cube: Torque Test                  | Pass    |
| Rubik's Cube: Alignment and holding        | Pass    |
| Etch-a-Sketch: Writing IEEE                | Pass    |
| Etch-a-Sketch: Alignment and Holding       | Pass    |
| Playing Card: Pick up and hold             | Pass    |
| Motors: Accurate PID Control               | Pass    |
| Line Following: Basic line correction      | Pass    |
| Line Following: Branch and return          | Pass    |
| Line Following: Stop at Finish Line        | Pass    |
| Structure: Robot fits within size          | Pass    |
| Integration: Rubik's Cube                  | Pass    |
| Integration: Etch-a-Sketch                 | Pass    |
| Integration: Simon Says                    | Pass    |
| Integration: Playing Card                  | Pass    |

|                               |      |
|-------------------------------|------|
| Full run                      | Pass |
| Integration: Start on LED OFF | Pass |

## 5 Schedule

### 5.1 External Deadlines

**Table 5.1: Spring Semester External Deadlines**

| Task                             | Intended Completion Date/Deadline |
|----------------------------------|-----------------------------------|
| Midterm Hardware Software Review | 2/26/2015                         |
| School Level Competition         | 3/20/2015                         |
| SoutheastCon 2015                | 4/9/2015                          |
| ECE Design Fair                  | 4/9/2015                          |
| ME Open House and Presentation   | 4/16/2015 or 4/17/2015            |

The most important external deadline for the team this semester is the internal competition on March 20th. The team must have the robot completed by this date so the team can successfully win the local competition and make it to SoutheastCon. Other important deadlines include the Midterm Hardware/Software Review and this report, as well as final presentations for both the Mechanical and Electrical Engineering Departments

### 5.2 Project Functionality Schedule

The project fell significantly off track from the originally optimistic/aggressive schedule the team set for itself as shown in Table 5.2. The biggest consequence of the team being off the original schedule is that there is significantly less time for time improvement and system level testing than originally desired. The team originally scheduled four months for these tasks, which is luckily almost certainly overkill.

**Table 5.2: Original Project Functionality Schedule**

| Task Name              | Team Members | Duration   | Time Frame             |
|------------------------|--------------|------------|------------------------|
| Propulsion             | 1            | 2 months   | Sept. 1st – Oct. 31st  |
| Chassis Design         | 4            | 1 month    | Sept. 1st – Sept. 30th |
| Line Following         | 2            | 2 weeks    | Sept. 1st – Sept. 15th |
| Simon/Rubik's arm      | 3            | 1.5 months | Sept. 1st – Oct. 15th  |
| Etch-a-sketch/Card arm | 3            | 1.5 months | Sept. 1st – Oct. 15th  |
| Circuit Development    | 3            | 3 weeks    | Sept. 29th – Oct. 20th |
| Integrate all systems  | 7            | 1 month    | Oct. 21st – Nov. 19th  |
| System Level Testing   | 7            | 1 month    | Nov. 19th – Dec. 17th  |



EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

|                   |   |          |                        |
|-------------------|---|----------|------------------------|
| Time Improvement  | 7 | 3 months | Jan. 5th – April 8th   |
| SoutheastCon 2015 | 7 | 4 days   | April 9th – April 12th |

Table 5.3 shows a more realistic schedule for the remainder of the semester, which will allow for two weeks each for Integration and Time Improvement as well as plenty of time for the remaining design and building of the Etch-A-Sketch and Rubik’s Cube arms. The most important conclusion that can be drawn from this table is that the team, while off from the original optimistic schedule, will still have time to finish the project successfully.

**Table 5.3 New Project Functionality Schedule**

| Task Name                             | Team Members | Duration | Deadline  |
|---------------------------------------|--------------|----------|-----------|
| Final Etch-A-Sketch Build             | 3            | 3 wks    | 2/20/2015 |
| Final Rubik’s Build                   | 2            | 3 wks    | 2/20/2015 |
| Integrate New Chassis with Propulsion | 2            | 2 wks    | 2/20/2015 |
| Integration                           | 7            | 2 wks    | 3/06/2015 |
| Time Improvement                      | 7            | 2 wks    | 3/20/2015 |
| Internal Competition                  | 7            | 1 day    | 3/20/2015 |
| Time Improvement                      | 7            | 2.5 wks  | 4/09/2015 |
| SoutheastCon                          | 7            | 4 days   | 4/09/2015 |

### 5.3 Final schedule compliance and justification

The team deviated significantly from the two schedules mentioned above. The main reason for this is that a high bar was set for stability. Rather than have a system work intermittently, the team wanted to make sure it would work every time, even if that required sacrifices in terms of speed. However, a robot that could reliably navigate and perform three of the four tasks (Etch-a-Sketch, Rubik’s Cube, and playing card) was ready by the local competition. This robot beat the other team, and as such fulfilled the only requirement at that time. Three weeks later, the team competed in Ft. Lauderdale with a robot that could perform the same three tasks reliably, as well as the fourth (Simon Says) with some degree of repeatability (achieved 8 and 11 seconds during the competition). Therefore, even though the above

schedules were not followed to the letter, it is safe to say that the project was completed successfully on time.

## 6 Budget Estimate

**Table 6.1: Milestone 1 budget estimate**

| Category               | Cost       |
|------------------------|------------|
| Wheels                 | \$80.00    |
| Motors                 | \$300.00   |
| Batteries/Chargers     | \$150.00   |
| Microcontrollers       | \$300.00   |
| Electronics            | \$200.00   |
| Misc. Mechanical Parts | \$170.00   |
| Total:                 | \$1,200.00 |

**Table 6.2: Currently specified parts cost**

| Purpose   | Items                     | Cost     |
|---|---------------------------|----------|
| Simon Frequency Analysis                              | Miscellaneous Electronics | \$11.47  |
| Simon/Rubik's Manipulator                             | Servo Motor               | \$19.40  |
| Etch-a-sketch manipulator                             | Miscellaneous equipment   | \$131.35 |
| Etch-a-sketch manipulator                             | Gear mounts               | \$20.35  |
| Battery Charger for Propulsion/Manipulators Batteries | Battery Charger           | \$40.08  |
| Batteries   | Power                     | \$83.25  |
| Propulsion  | Motor Drivers             | \$67.95  |

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

|                  |  |           |
|------------------|--|-----------|
| Propulsion       | Miscellaneous Equipment  | \$51.89   |
| Propulsion       | Motors for Propulsion  | \$177.25  |
| Propulsion       | Mecanum Wheels   | \$71.76   |
| Aluminium        | Chassis  | \$271.89  |
| Sensors          | Line Following (note: cost high because of overnight shipping) | \$88.01   |
| Microcontrollers | Logic  | 99.94     |
|                  | Total:   | \$1154.59 |

**Table 6.3: Estimated personnel cost**

| Personnel              | Total Hours per Semester | Total Hours Worked Both Semesters | Total Base Salary (\$30 per hour) | Total Salary + Fringe Rate of (29%) |
|------------------------|--------------------------|-----------------------------------|-----------------------------------|-------------------------------------|
| 1 Group Member         | 192                      | 384                               | \$11,520                          | \$14,860.8                          |
| Whole Team (7 Members) | 1,344                    | 2,688                             | \$80,640                          | \$104,025.6                         |

**Table 6.4: Estimated direct and overhead costs**

| Direct Cost | Direct Cost + Fringe Rate + Overhead rate (45%) | Direct Cost + Overhead Rate (45%) |
|-------------|---|-----------------------------------|
| \$81,148.01 | \$151,573.74                                    | \$117,664.62                      |

**Table 6.5: Total project cost**

| Category  | Expense (\$) |
|---|--------------|
| Supplies and Small Items (Current)              | \$1154.59    |
| Direct Cost + Fringe Rate + Overhead rate (45%) | \$151,573.74 |
| Total Project Cost:                             | \$152,773.74 |

In total, the team spent \$1154.59 on the project, well within the total budget of \$1250.

## 7 Conclusion

The goal of this project was to win the 2015 IEEE SoutheastCon Hardware Competition. With that in mind, an autonomous robot was designed from scratch. As detailed in this report, the system has been broken up into seven components: Chassis, Control, Line Following, Power, Propulsion, and two interface systems for the game challenges.

The budget for this project was \$1,250, and the team has stayed well within this limitation. This has been accomplished through careful specification and selection of parts before purchase, as well as extensive prototyping with already available components.

The robot was finished on time, and was able to compete in both the local and regional competitions.

In order to compete at SoutheastCon, the robot first had to beat Team 1B's robot in the internal competition. This took place on March 20<sup>th</sup>, 2015. The competition went mostly without a hitch, with the robot making it to the finish every time, and completing three out of four tasks (purposefully skipping Simon Says) two out of three times. During one of the runs, the robot did have some issues releasing the games after playing them, but this did not affect the outcome of the competition. As the other team's robot never made it to the finish, the qualification for SoutheastCon was fairly comfortable.

At SoutheastCon, the robot competed against 38 other teams (of which 35 were in the school competition). The team found themselves in the middle of a very competitive (top) part of the bracket and performed very well compared to most of the other teams that were there. After the first heat, FAMU-FSU was number 4, dropping to 5<sup>th</sup> and finally 7<sup>th</sup> place after the second and third rounds. All in all, the team is very happy with this position, and feels it appropriately reflects a full year of hard work and dedication.

## Appendix A: Testing Plans

### Scheduled Test Reporting Form

|   |                          |
|---|--------------------------|
| <b>Test:</b> Simon Says: Sequence Recreation (software) |                          |
| <b>Tester Name:</b> Julian Velasquez, Nils Bjerer       |                          |
| <b>Date:</b> 10/01/2014                                 | <b>Test Result:</b> Pass |
| <b>Time:</b> 7:00 PM                                    | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room                  |                          |

**Test Objective:**

To determine whether the sound sensor for the Simon Arm can accurately determine what button pattern has been given and then store this information for later use. (Where later use specifically would be hitting the buttons in the real competition).

**Test Description/Requirements:**

The microphone will be set up near the Arduino with the approximately the distance between it and the robot that will be seen in competition the game will then be played by a human. The robot will sense which buttons have been chosen by the toy and save then display this data to the user.

**Success Condition:**

The robot displays each of the buttons selected by the toy correctly. For each sequence of buttons, the display shows the entire sequence correctly.

**Results:**

The robot passed this test.

**Reason for Failure: N/A**

**Recommended Fix: N/A**

**Other Comments:**

### Scheduled Test Reporting Form

|   |                          |
|---|--------------------------|
| <b>Test:</b> Simon Says: Sequence Recreation (hardware) |                          |
| <b>Tester Name:</b> Nils Bjerer, Julian Velazquez       |                          |
| <b>Date:</b> 03/25/2015                                 | <b>Test Result:</b> Pass |
| <b>Time:</b> 13:00                                      | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room                  |                          |

**Test Objective:**

Determine if the manipulator is capable of accurately rotating to each position corresponding to the four colors of the Simon game and press each button correctly according to the button sequence sampled by the microphone circuit.

**Test Description/Requirements:**

When the Simon Says game is being played by the robot, the correct buttons must be pressed by the manipulator according to the order in which the game played the sounds. Since the buttons are not incredibly small, there exists some margin of error.

**Success Condition:**

The servo must rotate to the correct position and the chain drive must lower the servo down to press the button; It must then lift it up to press the next button. The buttons must be pressed in the correct order to pass this test.

**Results:**

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|   |                          |
|---|--------------------------|
| <b>Test:</b> Rubik's Cube: Torque Test            |                          |
| <b>Tester Name:</b> Nils Bjerer, Julian Velasquez |                          |
| <b>Date:</b> 10/1/2014                            | <b>Test Result:</b> Pass |
| <b>Time:</b> 6:30 PM                              | <b>Notes:</b> Ref: Video |
| <b>Location:</b> ME Senior Design Room            |                          |

**Test Objective:**

It had been determined early on that a high amount of torque is required to turn a row on a Rubik's Cube. This test is designed to make sure that the continuous rotation servo together with the end-effector have enough torque to complete this task. The servo will also need to be able to turn the row 180 degrees.

**Test Description/Requirements:**

The Rubik's Cube's bottom two rows will be held in place while the end-effector is placed on the top row. The servo will then turn the top row 180 degrees.

**Success Condition:**

The top row of the Rubik's Cube will be turned 180 degrees while the rest remains stationary.

**Results:**

Originally, this test failed - the torque of the servo was high enough, but the team was unable to make the servo move 180 degrees reliably. As this was a continuous rotation servo, there is only open loop control of position in terms of speed. A simple encoder was assembled out of a leftover IRR sensor that fixed the problem (see video). After this fix, the test was passed.

**Reason for Failure:**

Originally - lack of control over continuous rotation Servo.

**Recommended Fix:**

Home-made encoder - Implemented.

**Other Comments:**

### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Rubik's Cube: Alignment and Holding |                          |
| <b>Tester Name:</b> Christopher Lewis            |                          |
| <b>Date:</b> 1/20/2015                           | <b>Test Result:</b> Pass |
| <b>Time:</b> 3:00                                | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room           |                          |

**Test Objective:**

Determine if the pincer mechanism designed to align the Rubik's Cube and hold it stationary is capable of doing so.

**Test Description/Requirements:**

Place the Rubik's Cube in multiple probably positions and orientations in front of the robot. Tell the robot to grip onto the Rubik's Cube, and then try to turn one row of the Rubik's Cube.

**Success Condition:**

The test will be considered passed if:

1. The robot can grip onto the Rubik's cube AND put the cube in the right position for the gripper to be able to grab it AND
2. If the right amount of torque is applied to turn one row of the Rubik's cube, the rest of the Cube must stay stationary.

**Results:**

The test failed. When the amount of torque necessary to turn one row was applied to the Rubik's cube, the whole cube turned. Also, alignment was only shown to work in limited possible positions.

**Reason for Failure:**

Not enough torque was given by the servos on the pincers.

**Recommended Fix:**

Purchase servo with more torque.

**Other Comments:**

Stronger servos were implemented, and they fixed the problem.



### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Etch-a-Sketch: Writing 'IEEE' |                          |
| <b>Tester Name:</b> Chris Lewis            |                          |
| <b>Date:</b> 10/08/2014                    | <b>Test Result:</b> Pass |
| <b>Time:</b>                               | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room     |                          |

**Test Objective:**

To determine if the stick tape can actually grab the knobs, if the DC motors moving the Etch-A-Sketch knobs have enough torque, and to verify the pattern for IEEE programmed in the robot is correct.

**Test Description/Requirements:**

The Etch-A-Sketch manipulator assembly will be placed on top of the Etch-A-Sketch. The Etch-A-Sketch program will then run, hopefully writing IEEE on the Etch-A-Sketch.

**Success Condition:**

This test will be a success if the letters "IEEE" appear legibly on the Etch-A-Sketch screen.

**Results:**

The test passed.

**Reason for Failure:**

N/A

**Recommended Fix:**

N/A

**Other Comments:**

A video of this test is posted on the team's Google Drive.

### Scheduled Test Reporting Form

|   |                          |
|---|--------------------------|
| <b>Test:</b> Etch-A-Sketch: Alignment and Holding |                          |
| <b>Tester Name:</b> Christopher Lewis             |                          |
| <b>Date:</b> 2/27/2014                            | <b>Test Result:</b> Pass |
| <b>Time:</b> 03:00                                | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room            |                          |

**Test Objective:**

Determine if the holding and alignment mechanism for the Etch-A-Sketch is capable of properly aligning the game to the robot and allowing the motors to freely turn the knobs of the game.

**Test Description/Requirements:**

Place the Etch-A-Sketch in front of the mechanism and actuate the servos that operate the arms. Check to see if the EAS is properly aligned with the motors that turn the knobs. Repeat the test for several orientations of the EAS to ascertain its effectiveness.

**Success Condition:**

EAS aligned properly no matter which reasonable orientation is tested and the knobs the game are able to freely rotate

**Results:**

The holding mechanism passed the test

**Reason for Failure:** N/A

**Recommended Fix:** N/A

**Other Comments:**

This test was passed using a prototype mechanism. The test will need to be repeated once a final product is built.

### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Playing Card: Pick Up and Hold                  |                          |
| <b>Tester Name:</b> Kurt Marsman, Chris Lewis, Donovan Carey |                          |
| <b>Date:</b> 3/18/2014                                       | <b>Test Result:</b> Pass |
| <b>Time:</b> 21:00   | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room                       |                          |

**Test Objective:**

Determine whether or not the Etch-A-Sketch manipulator is capable of picking a card up off of the top of a deck of 52 playing cards and then hold it.

**Test Description/Requirements:**

Place a deck of cards under the EAS manipulator and lower it onto the deck. Raise the manipulator up off of the deck and check to see whether or not a single card stuck to the manipulator.

**Success Condition:**

A single card picked up and held onto by the manipulator

**Results:**

The card stayed on the manipulator, so the test was passed.

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|   |   |
|---|---|
| <b>Test:</b> Motors: Accurate PID Control |   |
| <b>Tester Name:</b> Nils Bjerer           |   |
| <b>Date:</b> 1/30/2015                    | <b>Test Result:</b> Pass                                |
| <b>Time:</b> 4:30 PM                      | <b>Notes:</b> Corresponds to new chassis.<br>Ref: Video |
| <b>Location:</b> Test Track               |   |

**Test Objective:**

The objective of this test is to determine whether the encoder-based PID motor control is sufficient for accurate control of the robot.

**Test Description/Requirements:**

The robot will be fed direction commands via USB cable from a computer. Various reasonable speeds will be tested, and the robot will be manually driven forwards, backwards, sideways, as well as spin around its vertical axis.

**Success Condition:**

The robot is able to move in the commanded direction with minimal slip or delay.

**Results:**

The robot was able to move in all directions (including to either side) as commanded. As can be seen in the corresponding video, next to no slip occurred when moving in any direction.

**Reason for Failure:**

N/A

**Recommended Fix:**

N/A

**Other Comments:**

This was under testing and refinement on the wooden prototype chassis. This passed test was conducted after the transition to the final aluminum chassis.

### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Line Following: Basic Line Correction |                          |
| <b>Tester Name:</b> Ryan-David Reyes, Nils Bjerer  |                          |
| <b>Date:</b> 1/11/2015                             | <b>Test Result:</b> Pass |
| <b>Time:</b> 5:00 PM                               | <b>Notes:</b>            |
| <b>Location:</b> Test Track                        |                          |

**Test Objective:**

This test is intended to test whether or not the basic line following algorithm of the robot is sufficient.

**Test Description/Requirements:**

The robot will be placed on a white line on a black surface, configured simply to follow the white line. It will try to autonomously stay on top of the white line.

**Success Condition:**

The robot is able to navigate down the white line, correcting in order to maintain the proper heading, leaving the line only when it ends.

**Results:**

This test was a success. The robot is able to reliably stay on a white line.

**Reason for Failure:**

N/A

**Recommended Fix:**

N/A

**Other Comments:**

While line following in a vacuum is trivial, the challenge will be making sure branching is reliable.

### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Line Following: Branch and Return |                          |
| <b>Tester Name:</b> Nils Bjerer, Ryan Reyes    |                          |
| <b>Date:</b> 2/20/15                           | <b>Test Result:</b> Pass |
| <b>Time:</b> 18:00                             | <b>Notes:</b>            |
| <b>Location:</b> Test Track                    |                          |

**Test Objective:**

The objective of this test is to make sure the robot can navigate down each branch as needed in order to find and complete the challenges. More important than the initial branching is the ability to proceed in the correct direction once returning to the main line.

**Test Description/Requirements:**

The robot will be configured to turn down every branch, stop at the white “challenge” square (to simulate playing a game), then navigate back down the branch and proceed with the course. It should be able to do this for each of the four game branches on the course in the same run.

**Success Condition:**

The robot is able to correctly branch in order to play a game, as well as know which way to go when it comes back to the main line.

**Results:**

The robot was successfully able to navigate down and back from each branch and refind the main line to finish the rest of the run

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|   |                          |
|---|--------------------------|
| <b>Test:</b> Line Following: Stop at Finish Line  |                          |
| <b>Tester Name:</b> Ryan-David Reyes, Nils Bjerer |                          |
| <b>Date:</b> 1/19/2015                            | <b>Test Result:</b> Pass |
| <b>Time:</b> 10:00 AM                             | <b>Notes:</b> Ref: Video |
| <b>Location:</b> Test Track                       |                          |

**Test Objective:**

The objective of this test is to make sure that the robot is able to successfully stop at the finish line after completing the course.

**Test Description/Requirements:**

The robot will be placed on the line, just after the starting box. It will proceed to navigate the course, configured to count branches, but not navigate down each one, staying on the main line. The test concerns whether the robot is able to stop at the finish line after “seeing” four branches, representing having played (or attempted) four games.

**Success Condition:**

The robot stops at the finish line after autonomously navigating the entire course.

**Results:**

The robot navigated the course, counting four branches (visible on the team’s debug software), and stopped at the finish line.

**Reason for Failure:**

N/A

**Recommended Fix:**

N/A

**Other Comments:**

While this test has been passed, it is important to emphasize that continued improvement and testing is necessary as more subsystems are added to the robot.

### Scheduled Test Reporting Form

|  |   |
|--|---|
| <b>Test:</b> Structure: Robot Fits Within Size Constraints |   |
| <b>Tester Name:</b> James Pace                             |   |
| <b>Date:</b> 1/30/2015                                     | <b>Test Result:</b> Fail  |
| <b>Time:</b> 6:00 PM                                       | <b>Notes:</b> Will be retested as additional components are added to the frame. |
| <b>Location:</b> ME Senior Design Room                     |   |

**Test Objective:** To make sure the robot fits within the 1 ft by 1 ft by ft size constraint stated in the competition rules.

**Test Description/Requirements:** For this test, a 1 ft by 1 ft plate will be used. The robot will be placed on the center of this plate. The robot will then be taken off the plate and will instead be placed on a flat surface with the plate placed perpendicular to the ground, with one side of the plate touching the ground.

**Success Condition:** The test will be considered a success if:

1. no part of the robot hangs off the plate when the robot is set on the plate AND
2. no part of the robot sticks above the plate when the plate is placed perpendicular to the ground.

**Results:**

The robot failed this test. The wheels on one side of the robot stuck approximately one tenth of an inch off the side the of the plate on one side of the plate, with the wheels on the other side just on the other edge. The robot had approximately another 5 inches of room along the axis that ran from the front to the back of the robot (without the end effectors attached), and approximately 5 inches in the direction of the axis that runs up and down.

**Reason for Failure:**

The size of the robot was checked before it was manufactured by measuring a model of the chassis in Creo. Creo estimated the width of the robot to be around 11.8 in, which is off from reality by approximately 0.3 in. This difference is most likely caused by the various surfaces of the Creo model being mated incorrectly/inaccurately. For example, the front of the motor bracket in Creo was mated with the front of the extrusion in the model, not by lining up the holes with the extrusion, which is more realistic. While the simulation is close to reality, small errors like this can quickly build up causing errors similar to what was measured in the test.

**Recommended Fix:**

To fix this, washers will added between the motors and the motor brackets. This will pull the motors and wheel assembly in towards the chassis, shrinking the robot along that direction.

**Other Comments:**

This test will be repeated as new elements are added to the chassis.



### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Integration: Rubik's Cube               |                          |
| <b>Tester Name:</b> Christopher Lewis, Donovan Carey |                          |
| <b>Date:</b> 03/09/2014                              | <b>Test Result:</b> Pass |
| <b>Time:</b> 15:00                                   | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room               |                          |

**Test Objective:**

The objective of this test is to verify that the "Rubik's Cube" capability is successfully integrated with the robot.

**Test Description/Requirements:**

The robot will grab, align, and hold the Rubik's Cube. Then, it will turn exactly one row of the Rubik's Cube 180 degrees before letting go of it. The cube will be prepositioned within reasonable reach of the alignment mechanism, but the robot must autonomously complete the task.

**Success Condition:**

The robot is able to autonomously twist one row of a Rubik's Cube 180 degrees.

**Results:**

This test succeeded. The whole "twist Rubik's Cube" system works well.

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|   |                          |
|---|--------------------------|
| <b>Test:</b> Integration: Etch-a-Sketch |                          |
| <b>Tester Name:</b> Chris Lewis         |                          |
| <b>Date:</b> 3/01/15                    | <b>Test Result:</b> Pass |
| <b>Time:</b> 5:00am                     | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room  |                          |

**Test Objective:**

The objective of this test is to verify that the “Etch-a-Sketch” capability is successfully integrated with the robot.

**Test Description/Requirements:**

With the Etch-a-Sketch subsystem attached, the robot will write ‘IEEE’ on an Etch-a-Sketch. It should be able to align the toy and actuate the knobs. The Etch-a-Sketch will be placed within reasonable reach of the alignment system, but the robot needs to complete the task autonomously.

**Success Condition:**

The robot is able to autonomously write ‘IEEE’ on an Etch-a-Sketch.

**Results:**

Etch-A-Sketch works successfully.

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Integration: Simon Says   |                          |
| <b>Tester Name:</b> Kurt Marsman       |                          |
| <b>Date:</b> 4/01/15                   | <b>Test Result:</b> Pass |
| <b>Time:</b> 12:00pm                   | <b>Notes:</b>            |
| <b>Location:</b> ME Senior Design Room |                          |

**Test Objective:**

The objective of this test is to verify that the "Simon Says" capability is successfully integrated with the robot.

**Test Description/Requirements:**

After attaching the Simon Says subsystem to the robot, it will play the game for 15 seconds without error. The Simon Says will be prepositioned for the alignment system, but the robot will autonomously play the game.

**Success Condition:**

The robot is able to navigate to the Simon Says game and play it for 15 seconds without error.

**Results:**

Robot can play game for full time.

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|  |                          |
|--|--------------------------|
| <b>Test:</b> Integration: Playing Card |                          |
| <b>Tester Name:</b> James Pace         |                          |
| <b>Date:</b> 3/19/15                   | <b>Test Result:</b> Pass |
| <b>Time:</b> 3:00pm                    | <b>Notes:</b>            |
| <b>Location:</b> Test Track            |                          |

**Test Objective:**

The objective of this test is to verify that the “playing card challenge” capability is successfully integrated with the robot.

**Test Description/Requirements:**

The robot will autonomously navigate from its starting position to the deck of cards. It will then pick up a single card and carry it to the finish line, where it will stop. This test will be carried out as a part of the “full run” test in order to ensure cross-subsystem compatibility.

**Success Condition:**

The robot is able to pick up and carry a card to the finish line as part of a successful run.

**Results:**

Robot successfully picked up playing card.

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

### Scheduled Test Reporting Form

|                                  |                          |
|----------------------------------|--------------------------|
| <b>Test:</b> Sense Start LED     |                          |
| <b>Tester Name:</b> Kurt Marsman |                          |
| <b>Date:</b>                     | <b>Test Result:</b> Pass |
| <b>Time:</b>                     | <b>Notes:</b>            |
| <b>Location:</b> Test Track      |                          |

**Test Objective:**

To test whether the MCU can correctly recognize the LED's status using the on board photoresistor.

**Test Description/Requirements:**

The photoresistor will be placed above the LED, and the MCU will provide information on the status of the LED - whether it is on or off.

**Success Condition:**

The MCU can correctly determine when the LED is on or off.

**Results:**

The MCU can correctly sense the state of the LED using the photoresistor.

**Reason for Failure:**

N/A

**Recommended Fix:**

N/A

**Other Comments:**

N/A

### Scheduled Test Reporting Form

|  |                            |
|--|----------------------------|
| <b>Test:</b> Integration: Start at LED OFF |                            |
| <b>Tester Name:</b> Kurt Marsman           |                            |
| <b>Date:</b> 4/01/15                       | <b>Test Result:</b> Passed |
| <b>Time:</b> 2:00pm                        | <b>Notes:</b>              |
| <b>Location:</b> Test Track                |                            |

**Test Objective:**

Determine whether or not the robot is successfully able to begin navigating the course and completing games when the red LED under the starting box is turned off.

**Test Description/Requirements:**

The robot will be placed over the starting LED and the LED will be turned on. The robot will then constantly check the voltage value across an onboard photoresistor for a sudden rise to indicate that the red LED has in fact been turned off. Once the light has been turned off, the robot will proceed to move towards the line, and engage in line following mode.

**Success Condition:**

Successful completion of the "start at LED off" test entails successfully starting and moving along the test track when the start LED is turned off.

**Results:**

Robot moved consistently after the LED was turned off.

**Reason for Failure:**

**Recommended Fix:**

**Other Comments:**

Required the additon of a skirt around the photoresistor.

### Scheduled Test Reporting Form

|                                      |  |
|--------------------------------------|--|
| <b>Test:</b> Full Run                |  |
| <b>Tester Name:</b> All Team Members |  |
| <b>Date:</b> 3/20/2015               | <b>Test Result:</b> Passed   |
| <b>Time:</b> 9:00                    | <b>Notes:</b> Tested during local competition.<br>Updated during SoutheastCon. |
| <b>Location:</b> Test Track          |  |

**Test Objective:**

Have the robot autonomously complete a full run of the course, successfully completing all four challenges.

**Test Description/Requirements:**

The robot will start at the right time (when the red LED turns off), and follow the line to each challenge. It will complete each challenge as per the rules. After the last challenge, the robot will navigate to the finish line and stop.

**Success Condition:**

The robot finishes the track, completing all four challenges. The maximum amount of non-time points are obtained.

**Results:**

The robot completed all challenges within three minutes. The robot could not complete Simon, but did get 11 points from the judges out of 15.

**Reason for Failure:**

It was decided to not pursue Simon.

**Recommended Fix:**

**Other Comments:**

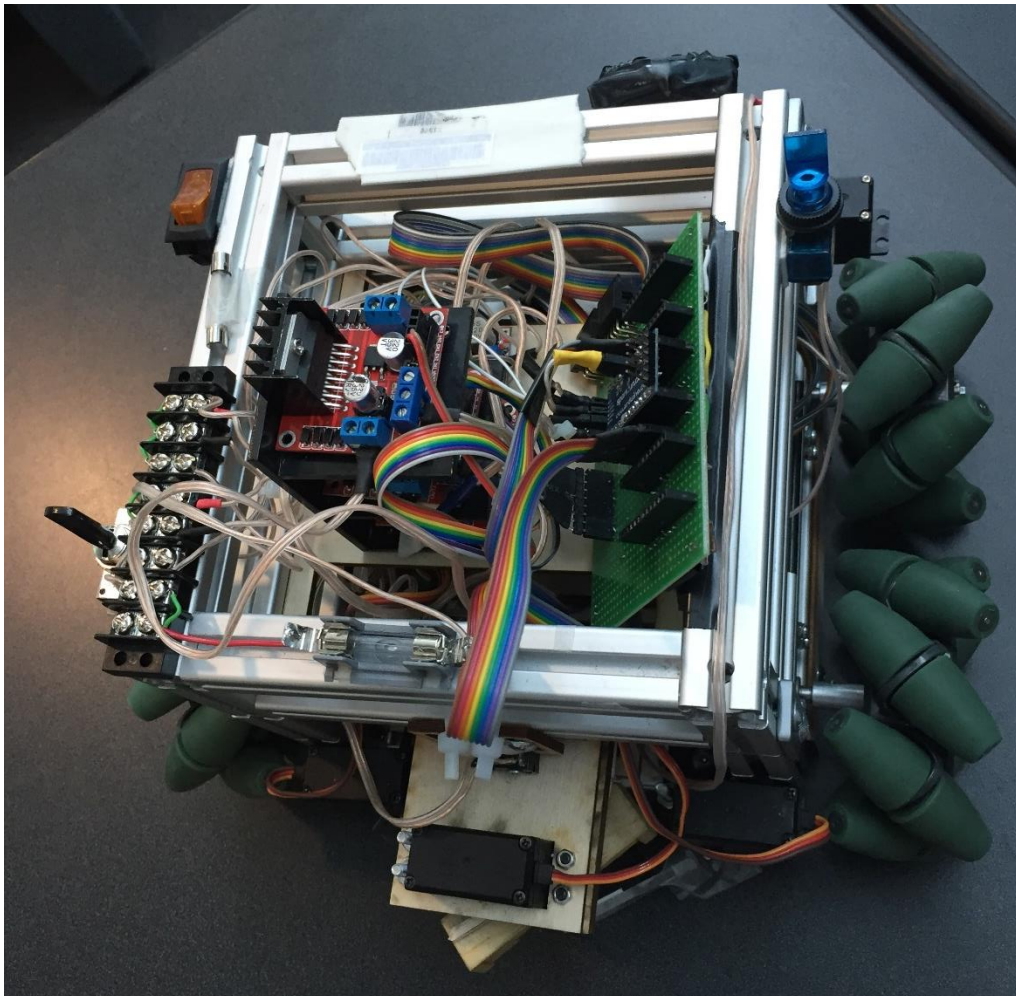
This test requires all subsystems to be complete before it can be performed completely.

## Appendix B: User's Guide

# SoutheastCon 2015 Hardware Competition

# User Manual

ECE Team #1A, ME Team #29





## **Setup/Configuration**

Before each of the runs please make sure to check every single item in this check list:

### **Steps to take**

1. Tighten wheels
2. Tighten T-nuts
3. Tighten both power screws
4. Replace and/or check tape
5. Make sure Rubik's/Simon manipulator fits snug
6. Line sensor power/line sensors on
7. Check voltage on battery
8. Make sure Arduinos are on
9. make sure servos move when excited
10. align all the games inside the boxes

## **Operation**

Ensure that Lithium Polymer battery is at a voltage of atleast 9.6 V. (Note that the voltage drop of the battery is exponential as each individual cell drops from 4.2 V to 3.2 V) An ideal voltage for the battery will be between 10.5 V to 12.6 V.

Make sure all three switches are in the off position, opposing the red tape which marks the on position.

Place the robot over the red LED which should be on, turn the orange switch to the ON position indicated by the red tape. After the switch is in the ON position an orange LED should shine in the second layer. This LED indicates that the photo resistor "sees" the red LED under it.

After the orange LED is on, turn the black switch to the ON position. This switch controls the drive system of the robot. Without this switch the robot will not be able to produce any movement.

The final switch that should be put in the ON position is the red switch. This switch sends the main MCU a signal that will let the MCU know that it should be looking for a state change in the red LED in order to begin the run.

EEL4914C/49415C: Senior Design Spring 2015 Final Report  
ECE Team 1A (ME 29)

## Appendix C: Software

Please see the attached .zip file for all the software used in this designed. This approach was considered cleaner than including all the code as text. Please contact Nils Bjerer at [nab13@gmail.com](mailto:nab13@gmail.com) if there are any issues accessing or interpreting any piece of software.