

Final Report
Aaron Zukley
Autonomous Laser Tank (ALT)
EEL5666C Intelligent Machine Design Laboratory
Instructors: Dr. A. Antonio Arroyo, Dr. Eric M. Schwartz
TAs: Andy Gray, Jake Easterling



Table of Contents

Table of Figures	1
Executive Summary.....	2
Abstract.....	3
Introduction	3
Integrated System.....	4
Mobile Platform	7
Actuation.....	7
Sensors	8
Behaviors.....	9
Experimental Layout and Results.....	9
Fuzzy Logic	9
Special Sensor	12
Conclusion.....	14
Documentation	14
Appendices.....	16
Python Image Processing Code.....	16

Table of Figures

Figure 1: ALT Robot Description Block Diagram	4
Figure 2: Target Tower Description Block Diagram	5
Figure 3: ODROID-C1+ Software Layout Diagram.....	5
Figure 4: Arduino Mega 2560 Software Layout Diagram	6
Figure 5: Arduino Uno Software Layout Diagram	6
Figure 6: ALT Robot Frame Top View	7
Figure 7: Image Processing Hardware: Odroid C1+ with Camera and Laser.....	12
Figure 8: Target Tower with Green Ball for Image processing	12
Figure 9: Color Masking using OpenCV Trackbars	13
Figure 10: Image color masking	13
Figure 11: Contour Detection and Ball Tracking.....	14

Executive Summary

The purpose of this paper is to explain the creation and functionality of the Autonomous Laser Tank (ALT). ALTs mission is to search an area for the target tower placed randomly within an obstacle course. Once the tower is located ALT uses image processing to align with the tower and shoot one of the light sensors on the tower with a laser. When the Laser hits the towers LEDs begin to flash red and the robot moves a safe distance away to await it's next mission.

For the physical construction of both ALT and the target tower, plexiglass acrylic sheets and pvc pipe were used. The plexiglass frame of ALT and the Base of the target tower were cut using a bandsaw and mounting holes were created using a simple hand drill. Each of the plexiglass pieces were combined using metal L brackets. The use of acrylic lead to a more finished looking robot and gave an interesting aesthetic look of being clear. Acrylic also had some issues, being that pieces were easy to crack and all wiring of the system was exposed leading to good wire management being encouraged.

To accomplish the desired behaviors several systems had to be developed. For obstacle avoidance infrared sensors and ultrasonic sensors readings were combined using fuzzy logic to smooth the motion of the robot. For the robot to function a power system had to be created to distribute the voltages from the 3 cell, 5000mah battery to 11.1V, 9V and 5V for the various systems on board. For motion two 50:1 geared Pololu motors were controlled by the Arduino Mega onboard. To target the tower an Odroid C1+ was used with a 720p USB web camera and OpenCV to distinguish the tower from other obstacles in the arena. A wireless network using X-Bees between the target tower and ALT was also created. With these systems ALT is able to implement the desired behaviors.

Along with the creation of ALT the target tower also needed to be created. For control of the target tower an Arduino Uno was used to read the light sensor values, send signals to ALT saying the robot was destroyed and to drive the LED strips that were used to illuminate the ball on top of the tower and to provide visual feedback showing the tower was destroyed. For the image processing a green ball was placed on top of the tower which is what ALT targets. For power, A 9V battery was used.

Abstract

ALT's mission is to search an area for the target tower, shoot it with a laser and move to a location designated with a colored object. The robot part of ALT will start by avoid obstacles while searching for the Target Tower. Once the Target Tower is found, ALT will use a camera to align itself to shoot a laser at the light sensors mounted on the Target Tower. Once the tower is hit, a wireless signal will tell the robot that the target was hit. Finally ALT will search the nearby area for a rendezvous location designated by the colored object.

Introduction

The inspiration for ALT came from a variety of sources. First, as a kid I had a passion for Remote control vehicles and laser tag. With these two pastimes came the desire to make a autonomous laser tag game between two robots. However, due to budget and time constraints the building of two robots seems unfeasible in one semester. Therefore, the plan for ALT is to design one of the robots now and possibly later grow the system into the full laser tag game.

In this paper I will describe the technical requirements of the ALT robot system and what steps will be taken towards the execution of this robot. The paper will begin with an overview of the hardware and software systems, leading into the individual actuator and sensors that will be implemented in this system.

Integrated System

ALT will be a robot driven by two motors attached to a Vex Tank Tread Kit and will use a combination of Bumper, Ultrasonic and Infrared sensors to avoid obstacles. Most of the sensors will be mounted on the front of the vehicle, reserving three ultrasonic sensors behind the vehicle to detect objects if backing up is required. The motors on ALT will be driven by a motor driver controlled by an Arduino Mega, which will also take readings from the Ultrasonic, bumper and Infrared sensors. A 5mW laser will be connected to the Arduino Mega for firing at the Target and a series two x-bee module will tell ALT when the Target Tower is hit by the laser. A camera will be attached to a ODROID-C1+ microcomputer, which will perform image processing for the robot to identify where the target is and where the rendezvous /home location is.

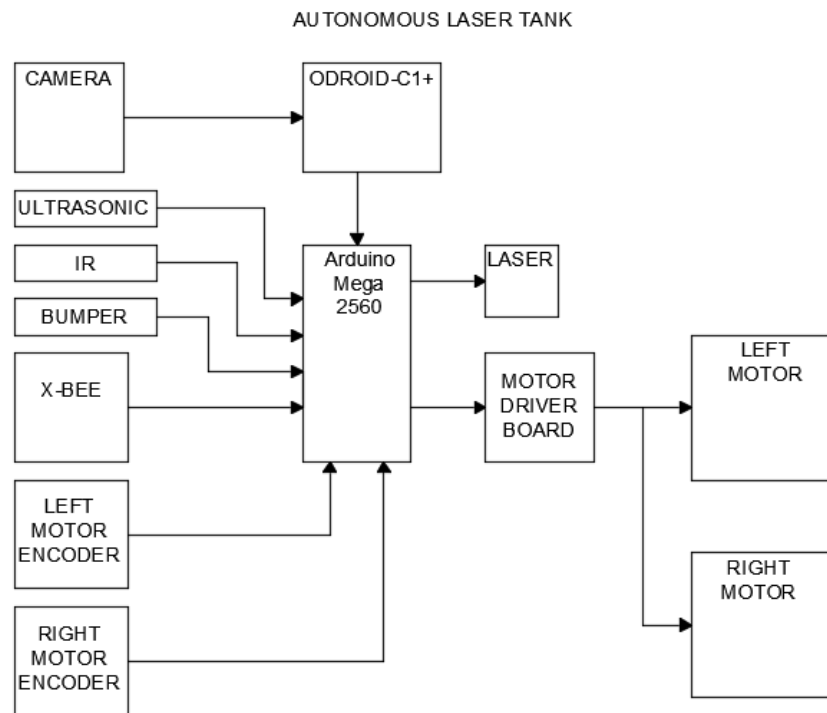


Figure 1: ALT Robot Description Block Diagram

The Target Tower will have six light sensors attached to the sides of the tower at the same height as the laser on ALT. The Light Sensors will have colored covers above and below the light sensors to block external light and to give an indication to ALT of where to aim. When a Sensor is tripped a signal is sent to an Arduino Uno in the tower which passes on the signal to an x-bee wireless module, which communicates to ALT.

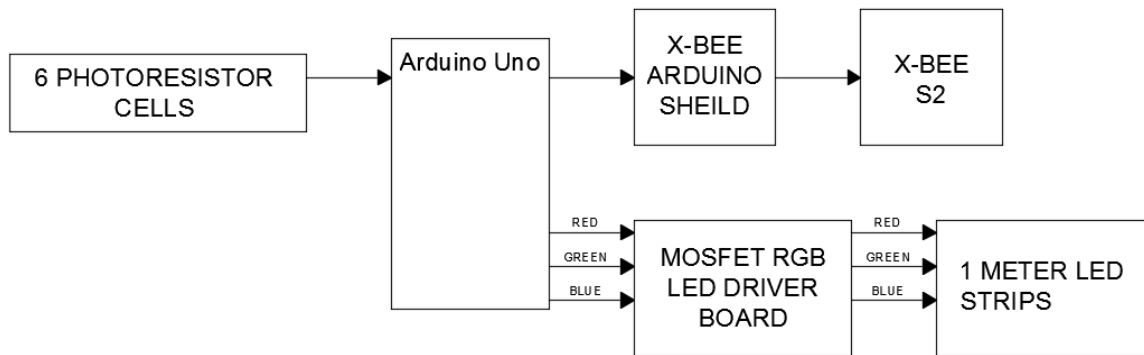


Figure 2: Target Tower Description Block Diagram

The software for this system will be subdivided between the three different controllers, the ODROID-C1+ shown in **Figure 3**, the Arduino Mega 2560 shown in **Figure 4**, and the Arduino Uno shown in **Figure 5**. The ODROID-C1+ is in charge of taking in images from the camera, evaluating the pictures, and determining how the robot needs to respond to shoot the target tower and go to rendezvous location. OpenCV will be used for image processing, using functions such as thresholding and object detection to recognize objects of interest. Once images are processed, commands will have to be made to tell the robot what direction to move. These commands will then be sent through USART to the Arduino mega controlling the rest of the system.

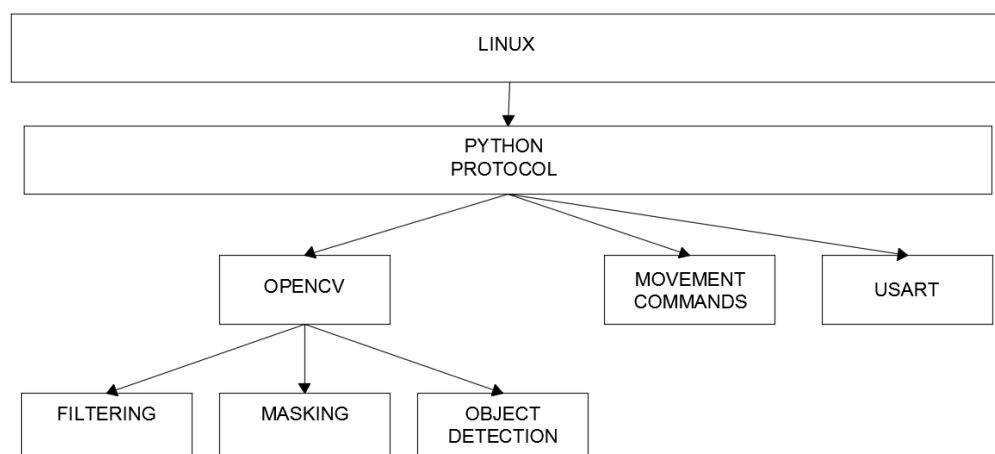


Figure 3: ODROID-C1+ Software Layout Diagram

The Arduino Mega will control most of the systems on the robot. The Arduino Mega will have to read sensors for obstacle avoidance, read commands from the ODROID, listen for confirmation that the laser hits the target tower, shoot the laser and control the motors. The plan is to use the C language in AVR studios and configure the Arduino boot loader, avrdude, to program the board. Using C will provide better control over the system and better execution of interrupts which will be crucial with this many systems on an older board.

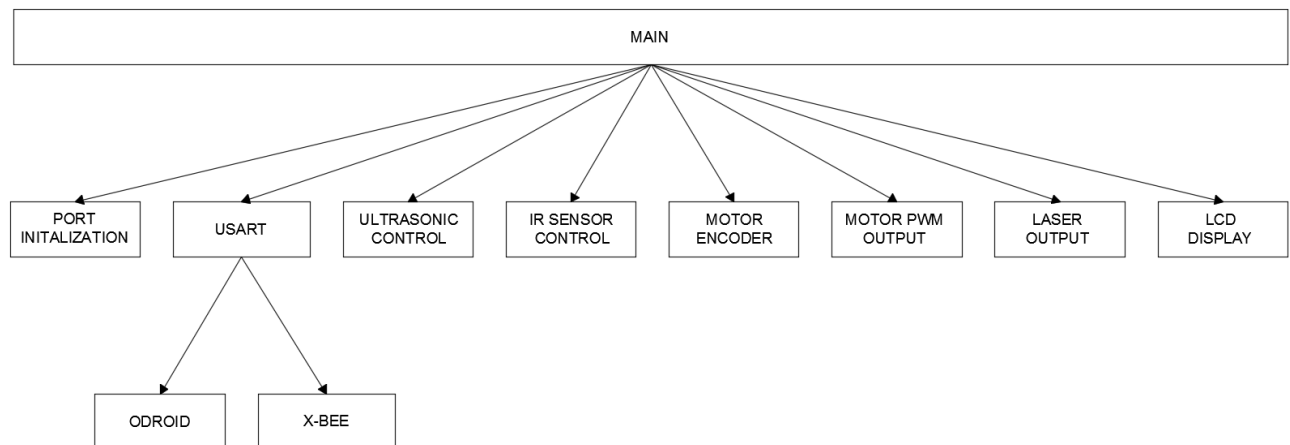


Figure 4: Arduino Mega 2560 Software Layout Diagram

The target tower will have an Arduino Uno to control the reading of the light sensors and to signal ALT that the target has been hit. LEDs may be added as well for visual effect and feedback. The Arduino Uno will also be coded in C. This will allow for the similar libraries used on ALT to be developed at the same time reducing coding time.

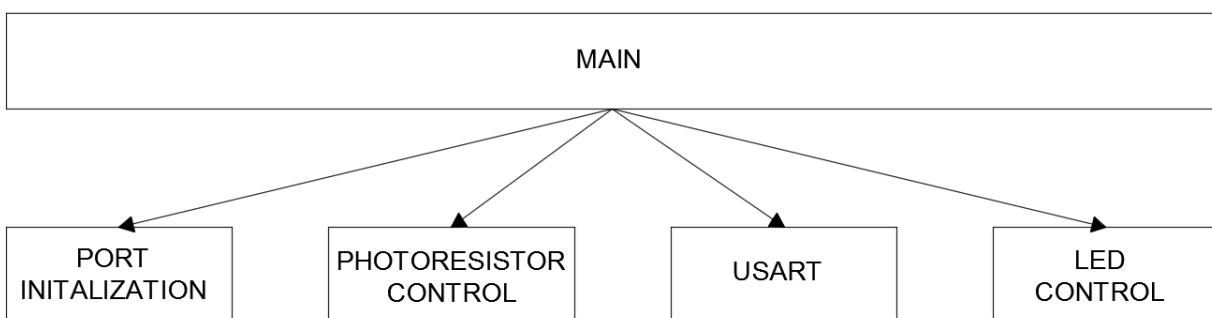


Figure 5: Arduino Uno Software Layout Diagram

Mobile Platform

The chaise for ALT will be made out of acrylic, using L-brackets to connect the individual pieces together. The base of the robot will be made out of eighth inch acrylic and the sides out of sixteenth inch acrylic. The thicker base makes the robot more stiff which should allow for better handling for the robot. The front and back of the robot will be extended to cover the tracks of the robot. With sensors mounted in front of the tracks, it will be less likely that the robot will get stuck on an object without sensing the object.

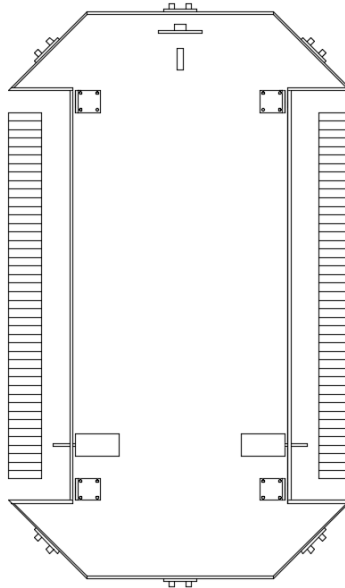


Figure 6: ALT Robot Frame Top View

The motors for the robot will be connected to an axle by a coupler that will connect the motors to the drive wheel of the tank tracks. The tank tracks came from a Vex robotics tank tread kit. The kit came with 170 tread links, 4 driver/idler wheels, 4 double support wheel assemblies, and 2 signal support wheel assemblies. Clamps will be used on the outside of the wheels to keep the wheels from slipping on the axle. Spacers will also be used to keep equal distance between the wheels and the chassis of the robot. Motor encoders came with the motors and will be used to ensure that the motors are behaving in the desired way.

Actuation

Alt will have two Pololu brushed motors geared to 50:1 ratio. At this ratio the motors can rotate at 200rpm and move up to 170oz. These motors are driven by a Pololu Dual H-Bridge VNH3SP30 bidirectional motor driver. These motor drivers have two bit direction control allowing for forward, reverse, and braking. Two PWM signals will be used to control the actual speed of the motors.

Sensors

ALT will have 6 HC-SR04 Ultrasonic sensors, 3 mounted towards the front of the robot and 3 on the back. These sensors can sense between 2cm and 400cm. The main goal of these sensors is to help with object detection and avoidance during the robots mission. These ultrasonic sensors were chosen primarily due to cost. These sensors were drastically less expensive than others I found coming in at three dollars each. One of the downsides to these sensors compared to other ultrasonic sensors is that they require four pins to control rather than three. The three being mounted on the back are for when the robot has to back up from objects being found in front of the robot.

Similarly, ALT will have three SharpGP2Y0A21YK Infrared sensors mounted on the front of the of the robot for obstacle detection and avoidance. The Infrared sensors can measure from 10cm to 80cm making these sensors good midrange sensors for the robot. Due to these sensors being sensitive to sunlight, testing will have to be done to see if outdoor search areas are viable. Things such as light covers may need to be made for these sensors.

ALT will also have two bump switches mounted to the front of the robot. These switches will the final effort is recognizing an object being in the path of the robot.

The Target Tower will need a way to detect when the laser hits the tower. Photoresistors that change resistance with light will sense when the laser is hitting the tower. When there is light on the sensor the resistance of the sensor will be close to $5k\Omega$, and when the sensor is dark the resistance will be close to $200k\Omega$. The Sensors can be set up in a voltage divider configuration and attached to the ADC pins of the Arduino Uno. One of the issues with this sensor is how small the sensor is and how small the beam of the laser is. Efforts will need to be made on the tower to increase the area of effect and robustness of the laser system. Materials such as mirrors or opaque materials may be investigated to widen the affect of the beam. Also pulsing the laser may allow for more robustness in the system, preventing shadows or changing light conditions from affecting the sensors.

Motor encoders are attached to each motor inside of ALT. These encoders count 64 counters per revolution and will be used to monitor the rotation of the motors as the robot runs through the course. These encoders are to help in preventing the robot from drifting or turning unintentionally due to the motors on being synchronized.

ALT will be using an ODROID USB web camera that runs at 30fps at 720p resolution. The camera will look for a specific object and color mounted on top of the target tower to identify the tower and to orient ALT to shoot the laser. The image processing will be done in OpenCV run on the ODROID-C1+. OpenCV will use image processes such as thresholding, edge detection and regions of interest to determine what adjustments are needed to be able to shoot the laser accurately. The camera will also be used in identifying the rendezvous location designated by a colored object.

Behaviors

ALT will be placed in an indoor area. Alt will begin searching the area using a combination of the camera and the obstacle avoidance sensors. Once the camera sees the Target Tower the robot will begin trying to align itself with the tower. Once the robot is aligned a pulsed laser signal will be shot at the target tower. If the laser fails to hit a sensor ALT will try to move closer and realign for another shot. If the laser succeeds in hitting one of the sensors then a wireless signal will tell ALT to begin searching for the location designated by a colored object in the nearby area. Once the designated area is found the robot will move to it and stop next to the object.

Experimental Layout and Results

Fuzzy Logic

For obstacle avoidance, ALT uses the average value given from two sets of fuzzy logic. One fuzzy logic set with the infrared sensors and one set with the front ultrasonic sensors. Fuzzy logic was used to smooth the motion of the robot as the robot searches the obstacle course. Fuzzy logic achieves this smoothing motion by comparing the sensor value to a range of predefined values. The definitions for the predefined values are shown below.

Sensor value definitions

- $VN = \text{Very Near}$
- $N = \text{Near}$
- $F = \text{Far}$
- $VF = \text{Very Far}$
- $VVF = \text{Very Very far}$
- $HL = \text{Hard Left}$
- $L = \text{Left}$
- $S = \text{Strait}$
- $R = \text{Right}$
- $HR = \text{Hard Right}$

Ultrasonic Distance conversion:

$$\text{speed of sound} = \frac{340 \text{ M/S}}{2}$$
$$\text{Distance} = (\text{high level time}) * \frac{340 \text{ M/S}}{2},$$

These ranges of values are defined at set distances for each sensor. For the ultrasonic sensors the time of the echo is converted to defined distances.

Table 1: Ultrasonic Sensor		
Ultrasonic	Time (ms)	Distance
VN	1.04	18
N	1.86	32
F	2.67	46
VF	3.48	60
VVF	4.29	74

The infrared sensors output an analog voltage that is then converted to a distance. Note that both the desired predefined distance values are the same for both the ultrasonic sensors shown in **Table 1** and the infrared sensors shown in **Table 2**.

Table 2: Infrared Sensor			
Infrared	Distance (cm)	Voltage (V)	10 Bit ADC value 5V Ref.
VN	18	2.33	476
N	32	1.80	328
F	46	1.28	261
VF	60	0.75	153
VVF	74	0.23	46

Using the predefined distance values, percentages are created comparing the sensor value to the two closest predefined distance values.

For example a distance reading of 40cm on either set of sensors would result in:

$$N = \frac{IR_{value} - Lower_{value}}{Higher_{value} - Lower_{value}} * 100 = \frac{40cm - 32cm}{46cm - 32cm} * 100 = 53.33\% \text{ near}$$

$$F = 100 - \frac{IR_{value} - Lower_{value}}{Higher_{value} - Lower_{value}} * 100 = 100 - \frac{40cm - 32cm}{46cm - 32cm} * 100 = 46.66\% \text{ far}$$

These percentages are collected for the left and right sensor before being used to find the motor output values mapped in **Table 3**. The percentages of each sensor are combined using the statements shown below.

Table 3: Fuzzy logic							
		Right Sensors					
		Sensors	VN	N	F	VF	VVF
Left Sensors	VN	L	HR	HR	HR	HR	
	N	HL	L	HR	R	R	
	F	HL	HL	S	S	S	
	VF	HL	L	S	S	S	
	VVF	HL	L	S	S	S	

- $HL = RVN \text{ and } (LN \text{ or } LF \text{ or } LVF \text{ or } LVVF)$
- $L = (LVF \text{ and } RN) \text{ or } (LVVF \text{ and } RN)$
- $S = (LVF \text{ or } LVVF) \text{ and } ((RF \text{ or } RVF \text{ or } RVVF) \text{ or } (LF \text{ and } (RVF \text{ or } RVVF)))$
- $R = (RVF \text{ and } LN) \text{ or } (RVVF \text{ and } LN)$
- $HR = LVN \text{ and } (RVN \text{ or } RN \text{ or } RVF \text{ or } RVVF)$

Once the motor direction percentages are found the final motor output value is calculated using a weighted average. This system produced a motor output value between -100 and 100, where -100 was hard left and 100 is hard right. To remove the need to use negative values an offset of 100 could be used to simplify the output.

$$Motor\ output = \frac{(HL * -100) + (L * -20) + (R * 20) + (HR * 100)}{HL + L + S + R + HR} + 100$$

Special Sensor

ALT uses the Odroid-c1+ and the Odroid USB web camera to perform image processing for targeting the Target Tower. On the Odroid-c1+ a python script uses OpenCV, an open source computer vision library to perform image processing techniques.

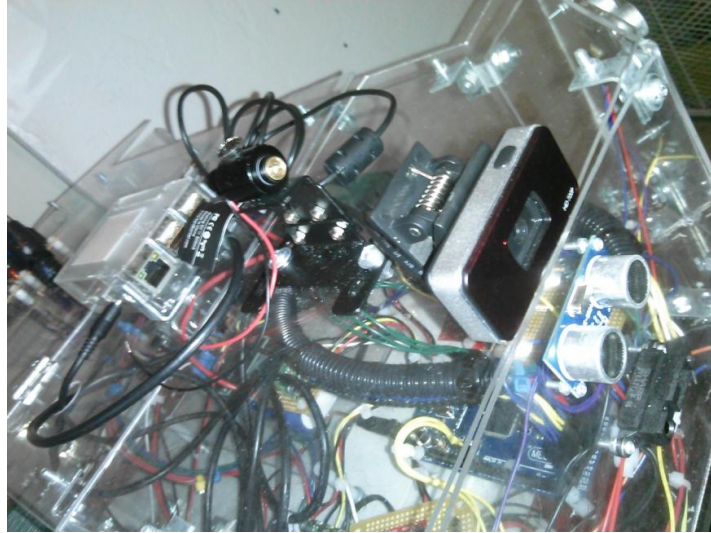


Figure 7: Image Processing Hardware: Odroid C1+ with Camera and Laser

The task for the Image processing is to identify the target tower. To help accomplish this goal a green ball was mounted on top of the target tower, which is what the image processing code helps to identify and track as the robot travels. The target tower is shown in **Figure 8**.



Figure 8: Target Tower with Green Ball for Image processing

Several methods were attempted to identify the green ball such as masking colors, identifying contours, implementing circle detection algorithms and several forms of filtering. The first experiment was with masking colors. A python test code was created to calibrate and test the masking of colors to filter out as much as possible while still leaving the green ball in the image. **Figure 9** shows the task bars used to calibrate the red, green and blue colors in an image.

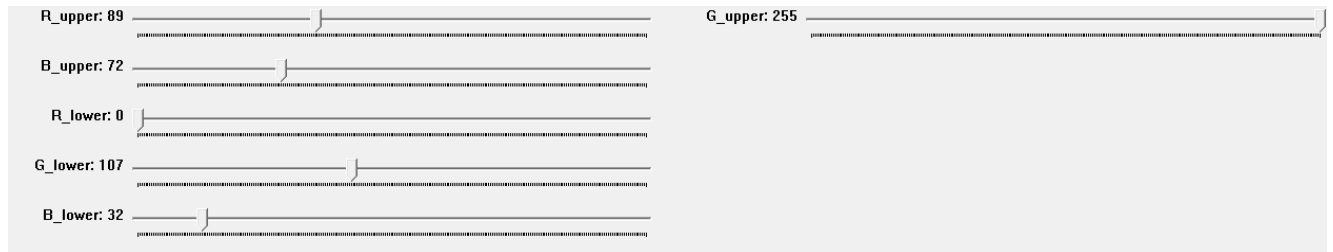


Figure 9: Color Masking using OpenCV Trackbars

The results of masking are shown in **Figure 10**. You can see from **Figure 10** that some of the ball was removed and a little noise is left over, however the vast majority of the remaining image is of the desired ball.

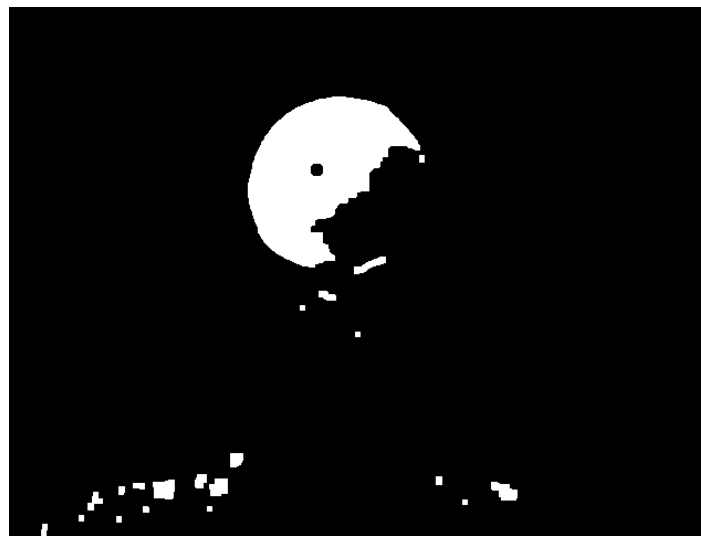


Figure 10: Image color masking

After a little more filtering using erosion and dilation to remove small noise the python script looks for circular contours in the image. Moments were experimented with to capture the center of the object however, with differing lighting conditions and making occasionally taking out part of the desired circle it was found that the center of the remaining pixels in the contour would not always be the center of the ball. Instead the smallest bounding circle for the contour was used to find the center position and to show correct identification of the ball. Hough Circle detection was also tested, however it was found to be slow and often found multiple circles that were not always desired.

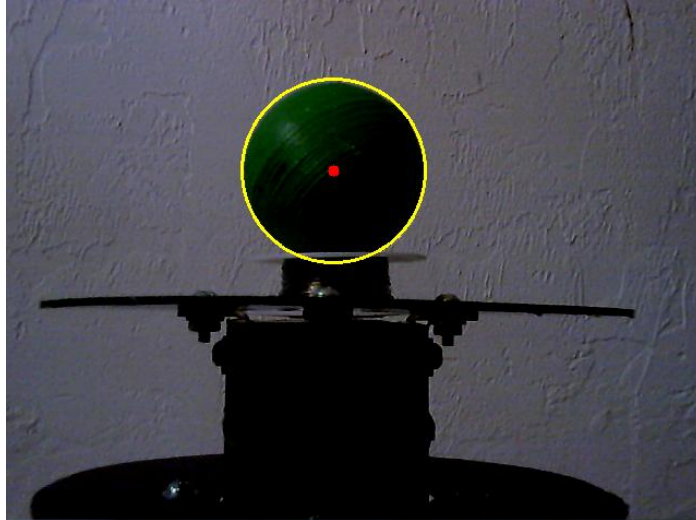


Figure 11: Contour Detection and Ball Tracking

Based off of the determined x coordinate of the center position of the ball a serial command was created telling the robot to either turn left, turn right or remain aligned. This system proved to be functional but separating out the regions of the image into more commands may prove beneficial so that the robot can better adjust depending on how far from center the ball is found to be.

Conclusion

Overall ALT and the target tower worked very well. ALT was able to perform the desired behaviors and complete the mission. I found that with working with lasers, alignment becomes critical. Several times during testing the targeting system I found the laser and camera miss aligned leading to poor performance. I found that Linux and OpenCV were my weakest areas. Starting this project I was new to python and had very little experience in Linux. I found that experience with command prompt and terminal is extremely useful for microcomputers, especially when a monitor is not available. I also found that as I gained experience with Python, my ability to develop better functioning OpenCV scripts improved. It was critical to get a functional work area to be able to program and develop on the microcomputer. I ran into several issues with monitor compatibility which made initial development tedious. I also found out the hard way that microcomputers as the name suggests are small computers and should be treated as such. Controlling the powering up and powering down of the microcomputer needs to be more controlled than a microprocessor. I also ran into some issues with wiring the Arduino Mega. Most of my issues stemmed from having so many thing connected to the Arduino Mega. For example, ALT to have as many Ultrasonic sensors as possible, a multiplexer was used which made it difficult at first to get all of the sensors working.

If I had more time there were several improvements that could have been made to make the system function even better. I suspect better calibration of the sensor readings may have added a slight improvement to the system. I also would have continued to improve the vision system allowing the robot to move towards a better defined final location or created multiple targets to go after. The original idea for this project was to create a laser tag robot. With what has currently been developed the original idea is a very possible improvement to the current system, where the target tower functionality would be combined with the current robot and a second robot would be developed so that laser tag battles and targeting intelligence could be explored.

Documentation

- ODROID-C1+:
http://www.hardkernel.com/main/products/prdt_info.php?g_code=G141578608433&tab_idx=2
- Arduino Mega 2560:
<https://www.arduino.cc/en/Main/ArduinoBoardMega2560>
- Arduino Mega 2560 Pin Out:
<http://3.bp.blogspot.com/-5bIrGV8-TfE/VKSNL21TULI/AAAAAAAAAAk/UC4vz6oc-Wg/s1600/ARDUINO.Mega.Pinout.Diagram.png>
- Arduino Uno:
<https://www.arduino.cc/en/Main/ArduinoBoardUno>
- Arduino Uno Pin Out:
http://marcusjenkins.com/wp-content/uploads/2014/06/ARDUINO_V2.png
- Sharp GP2Y0A21YK:
<http://www.sparkfun.com/datasheets/Components/GP2Y0A21YK.pdf>
- Ball Tracking Example Code:
<http://www.pyimagesearch.com/2015/09/14/ball-tracking-with-opencv/>

Appendices

The project website which contains a list of sensors, actuators, processing boards and more can be found at: <https://sites.google.com/site/zukleyaaron/>

Python Image Processing Code

```
import cv2
import numpy as np
import time
import serial

def nothing(x):
    pass

# For OpenCV2 image display
WINDOW_NAME = 'image'
WINDOW_NAME2 = 'HSV_mask'
WINDOW_NAME3 = 'hsv'
WINDOW_NAME4 = 'res'

cv2.namedWindow('control')

# create trackbars for HSV color change
cv2.createTrackbar('H_upper', 'control', 100, 255, nothing)
cv2.createTrackbar('S_upper', 'control', 255, 255, nothing)
cv2.createTrackbar('V_upper', 'control', 255, 255, nothing)
cv2.createTrackbar('H_lower', 'control', 30, 255, nothing)
cv2.createTrackbar('S_lower', 'control', 0, 255, nothing)
cv2.createTrackbar('V_lower', 'control', 0, 255, nothing)

# create trackbars for RGB manipulation
cv2.createTrackbar('Blue_upper', 'control', 0, 255, nothing)
cv2.createTrackbar('Green_upper', 'control', 0, 255, nothing)
cv2.createTrackbar('Red_upper', 'control', 0, 255, nothing)

cv2.createTrackbar('contrast', 'control', 100, 400, nothing)
cv2.createTrackbar('brighness', 'control', 0, 100, nothing)

# adjust the gamma value for the image
# makes the image look closer to human vision
cv2.createTrackbar('gamma', 'control', 100, 500, nothing)

def track(image):
    # build a lookup table mapping the pixel values [0, 255] to
    # their adjusted gamma values

    gamma = float(cv2.getTrackbarPos('gamma', 'control')/100)

    invGamma = 1.0 / gamma

    table = np.array([((i / 255.0) ** invGamma) * 255
        for i in np.arange(0, 256)].astype("uint8"))

    image = cv2.LUT(image, table)
```

```

# Blur the image to reduce noise
blur = cv2.GaussianBlur(image, (5,5),0)

# Convert BGR to HSV
hsv = cv2.cvtColor(blur, cv2.COLOR_BGR2HSV)

# construct a mask for the color "green", then perform
# a series of dilations and erosions to remove any small
# blobs left in the mask

kernel = np.ones((9,9),np.uint8)

#define range of green colors
upper_green = np.array([100,255,255])
lower_green = np.array([30,0,10])

mask = cv2.inRange(hsv, lower_green, upper_green)
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel)

# Bitwise AND mask and original image
res = cv2.bitwise_and(image,image, mask=mask)

# Assume no centroid
center1 = (-1,-1)

# find contours in the mask and initialize the current
# (x, y) center of the green ball
cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)[-2]

# only proceed if at least one contour was found
if len(cnts) > 0:

    areaArray = []

    num_of_contours = len(cnts)

    # Create an array of contours based on
    for i, c in enumerate(cnts):
        area = cv2.contourArea(c)
        areaArray.append(area)

    #first sort the array by area
    sortdata = sorted(zip(areaArray, cnts), key=lambda x: x[0],
reverse=True)

    for k in range(0,num_of_contours):
        #find the nth largest contour [n-1][1], in this case 2
        contour = sortdata[k][1]

        # find the largest contour in the mask, then use
        # it to compute the minimum enclosing circle and
        # centroid
        ((x, y), radius) = cv2.minEnclosingCircle(contour)

        area = cv2.contourArea(contour)

```

```

        circle_area = 3.14*radius*radius

        if area > circle_area*2/3:
            break

    # Using the Center of the min enclosing circle to define center
    center = (int(x),int(y))

    # only proceed if the radius meets a minimum size
    if radius > 10 and radius < 59 and area > circle_area*2/3:
        # draw the circle and centroid on the frame,
        # then update the list of tracked points
        cv2.circle(image, (int(x), int(y)), int(radius), (0,255,255), 2)
        cv2.circle(image, center, 5, (0,0,255), -1)
        center1 = center
    elif radius > 60 and area > circle_area*2/3:
        # Signal the robot is close to tower
        ser.write(chr(1))
        cv2.circle(image, (int(x), int(y)), int(radius), (0,255,255), 2)
        cv2.circle(image, center, 5, (0,0,255), -1)

    # Draw alignment lines
    cv2.line(image, (0,240), (640,240), (255,0,0), 5)

    # Draw alignment lines
    cv2.line(image, (320,0), (320,480), (255,0,0), 5)

    # Display full-color image
    cv2.imshow(WINDOW_NAME4, res)
    cv2.imshow(WINDOW_NAME3, hsv)
    cv2.imshow(WINDOW_NAME2, mask)
    cv2.imshow(WINDOW_NAME, image)

    if cv2.waitKey(1) & 0xFF == 27:
        center1 = None

    # Return coordinates of centroid
    return center1

#Function to Initialize the Serial Port
def init_serial():
    COMNUM = 1          #Enter Your COM Port Number Here.
    global ser          #Must be declared in Each Function
    ser = serial.Serial()
    ser.baudrate = 9600
    ser.port = COMNUM - 1    #COM Port Name Start from 0

    #ser.port = '/dev/ttyUSB0' #If Using Linux

    #Specify the Timeout in seconds, so that SerialPort
    #Doesn't hangs
    ser.timeout = 10
    ser.open()          #Opens SerialPort

    # print port open or closed
    if ser.isOpen():

```

```

        print 'Open: ' + ser.portstr
#Function Ends Here

#Call the Serial Initialization Function, Main Program Starts from here
init_serial()

# Test with input from camera
if __name__ == '__main__':

    capture = cv2.VideoCapture(1)

    print capture.get(3)
    print capture.get(4)

    # Tell if the video is opened
    print capture.isOpened()

    capture.set(5,1)

    # Change the resolution of the image
    # Set Width
    capture.set(3,640)

    # Set Hight
    capture.set(4,480)

    ser.write(chr(255))

    while True:

        okay, image = capture.read()

        if okay:

            # print track(image)
            (x,y) = track(image)

            if x > 1:
                # Convert x center position to 8 bit value
                out = 255*x/640

                # Serially send value to arduino mega
                ser.write(chr(out))
            else:
                ser.write(chr(0))

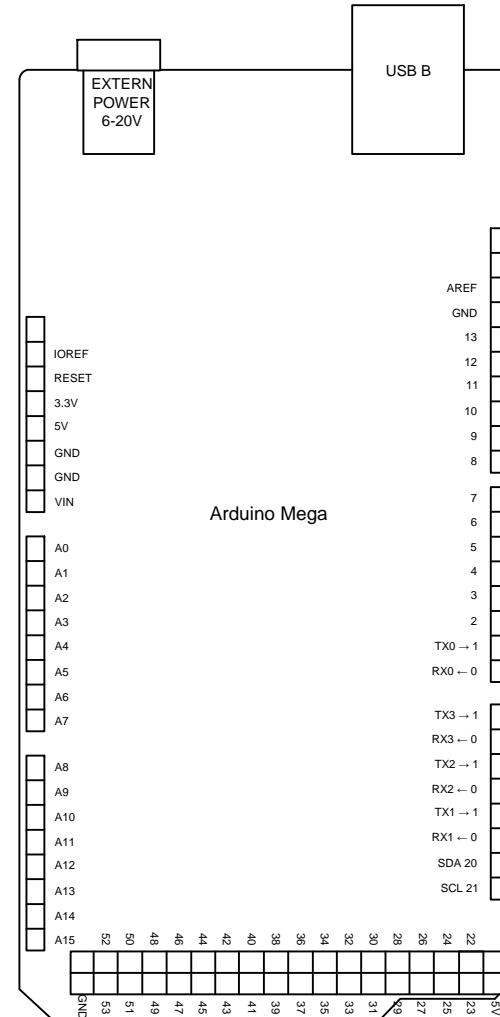
            if not track(image):
                break

            if cv2.waitKey(1) & 0xFF == 27:
                break

        else:
            print('Capture failed')
            break

```

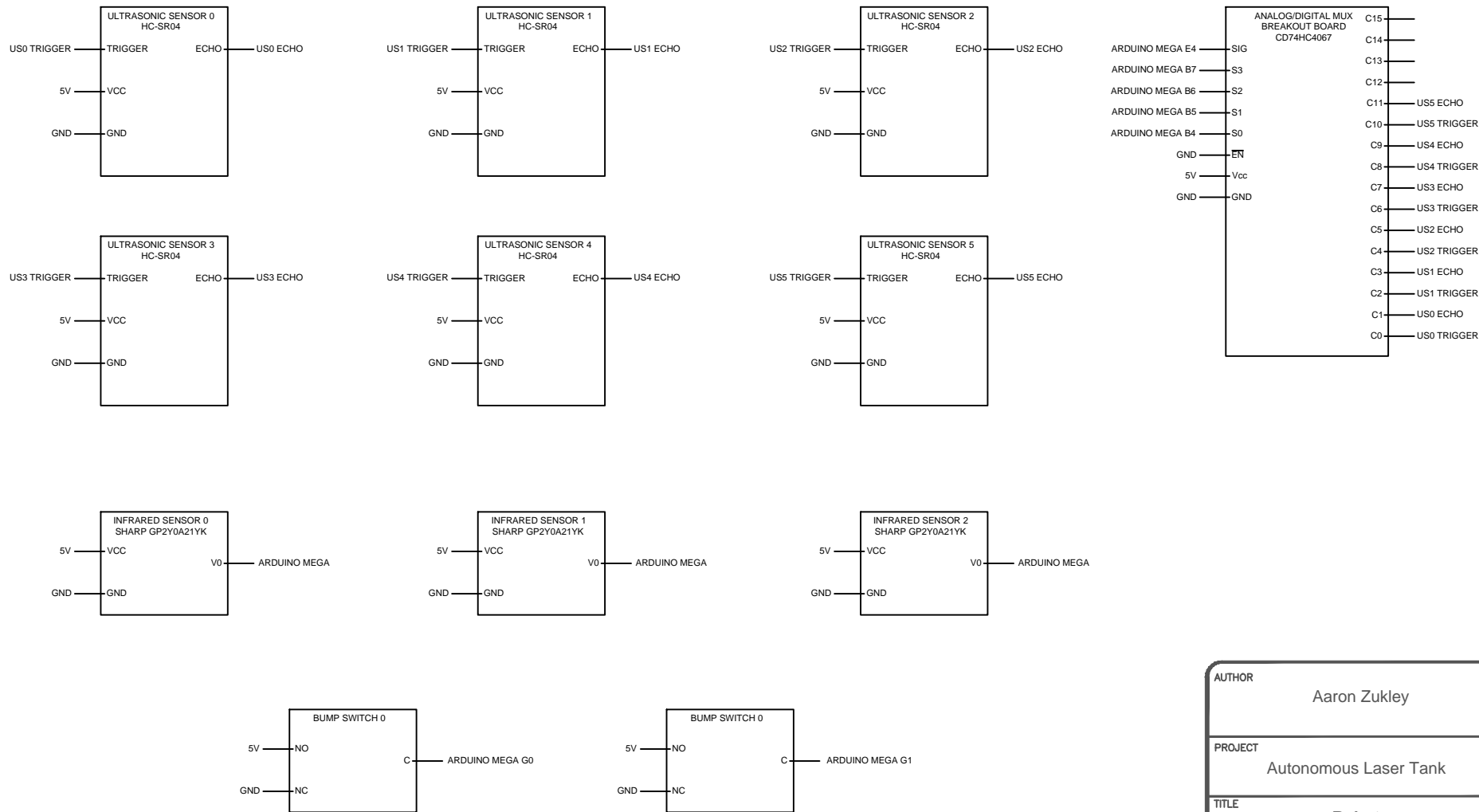

PORT	PIN	ARDUINO IDE PIN	FUNCTION
C	0	37	LED 1 RED
	1	36	LED 1 GREEN
	2	35	LED 1 BLUE
	3	34	LED 2 RED
	4	33	LED 2 GREEN
	5	32	LED 2 BLUE
	6	31	LED 3 RED
A	7	30	LED 3 GREEN
	0	22	LED 3 BLUE
	1	23	LED 4 RED
	2	24	LED 4 GREEN
	3	25	LED 4 BLUE
	4	26	LED 5 RED
L	5	27	LED 5 GREEN
	6	28	LED 5 BLUE
	0	35	LCD DATA 4
	1	36	LCD DATA 5
	2	37	LCD DATA 6
	3	38	LCD DATA 7
	4	39	LCD RS LINE
G	5	40	LCD R/W LINE
	6	41	LCD EN LINE
	0	51	LEFT BUMP SWITCH
K	1	52	RIGHT BUMP SWITCH
	0	A8	RIGHT MOTOR CONTROL BIT 0
	1	A9	RIGHT MOTOR CONTROL BIT 1
	2	A10	LEFT MOTOR CONTROL BIT 0
F	3	A11	LEFT MOTOR CONTROL BIT 1
	0	A0	INFRARED 0
	1	A1	INFRARED 1
D	2	A2	INFRARED 2
	0	21	RIGHT ENCODER A CHANNEL
	1	20	RIGHT ENCODER B CHANNEL
	2	19	LEFT ENCODER A CHANNEL
H	3	18	LEFT ENCODER B CHANNEL
	0	17	X-BEE RX USART
	1	16	X-BEE TX USART
	3	6	LEFT MOTOR PWM SPED CONTROL
J	4	7	RIGHT MOTOR PWM SPEED CONTROL
	0	15	LASER FIRING CONTROL
E	0	0	RESERVED USART SERIAL TO ODROID
	1	1	RESERVED USART SERIAL TO ODROID
	4	2	ULTRASONIC TRIGGER/ECHO TO MUX
B	4	10	ULTRASONIC MUX CHANNEL SELECT 0
	5	11	ULTRASONIC MUX CHANNEL SELECT 1
	6	12	ULTRASONIC MUX CHANNEL SELECT 2
	7	13	ULTRASONIC MUX CHANNEL SELECT 3



AUTHOR			
Aaron Zukley			
PROJECT			
Autonomous Laser Tank			
TITLE			
Robot Arduino Mega 2560			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW		REV. 0	
PROJECT No.			
DRAWING:			

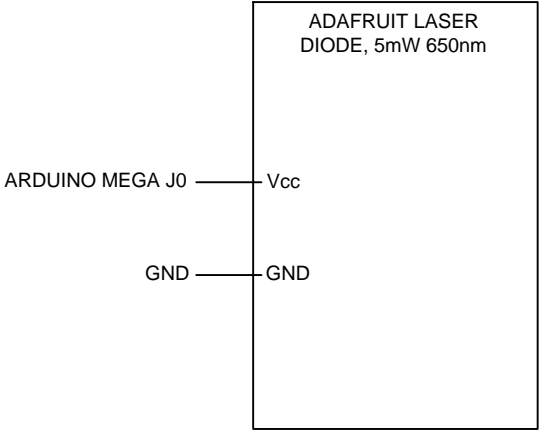
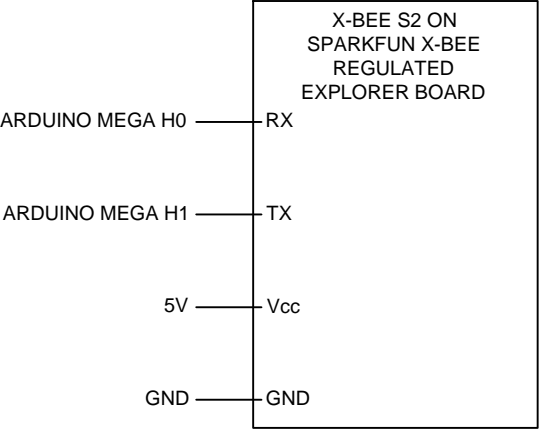
12/06/15	XXX	Rev 0	XXX		
REV	DATE	DES	REVISION DESCRIPTION	CADD	CHK RVW

12/06/15	XXX	Rev 0	XXX		
REV	DATE	DES	REVISION DESCRIPTION	CADD	CHK RVW

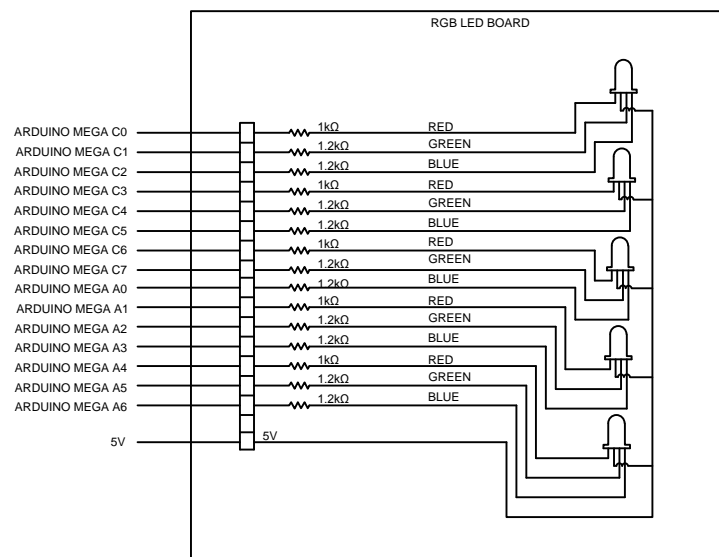
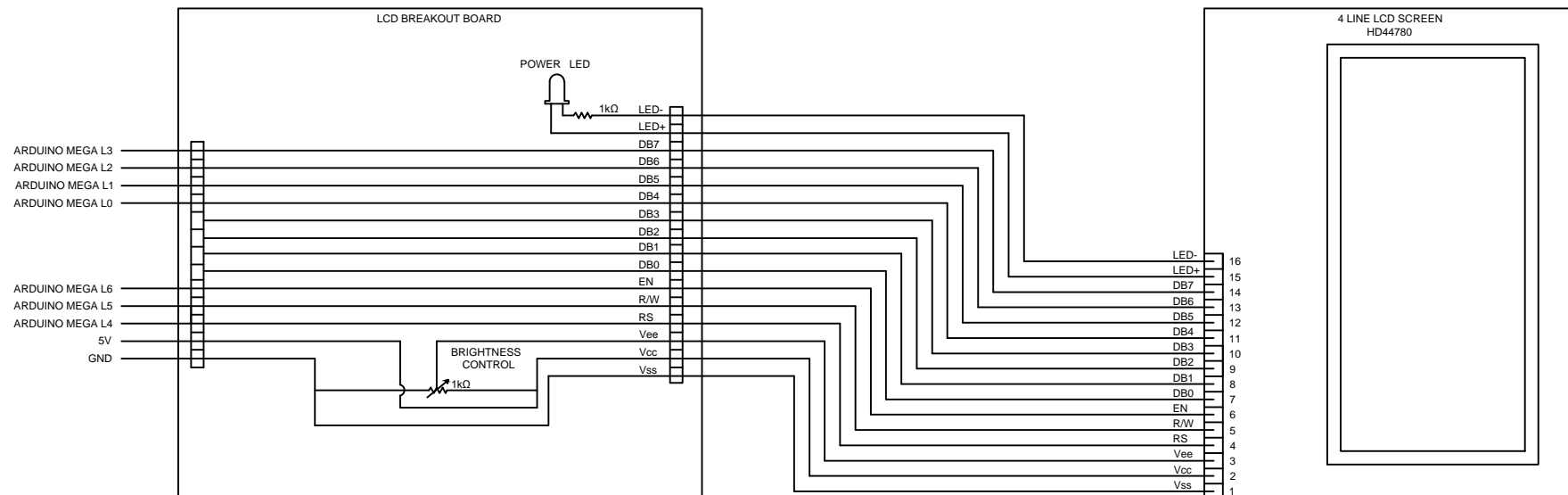


<div>AUTHOR</div> <div>Aaron Zukley</div>			
<div>PROJECT</div> <div>Autonomous Laser Tank</div>			
<div>TITLE</div> <div>Robot Sensors</div>			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW			REV. 0
PROJECT No.			
DRAWING:			

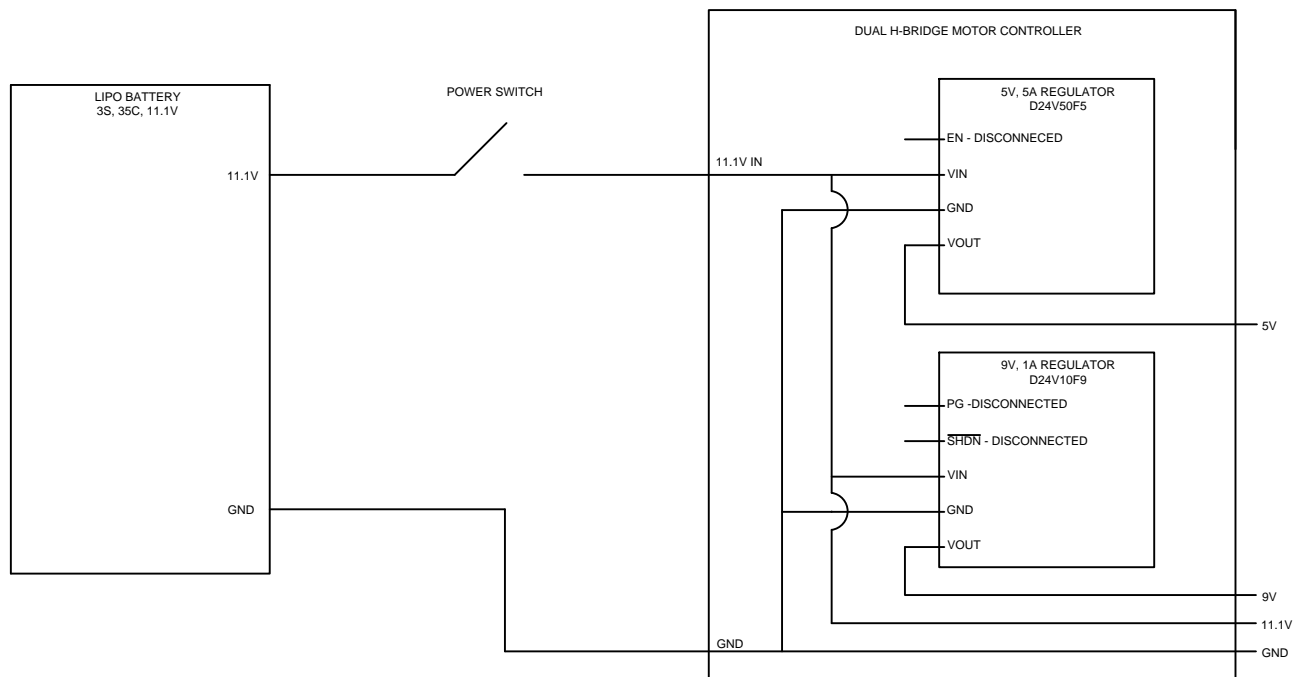
⚠	12/06/15	XXX	Rev 0			XXX
REV	DATE	DES	REVISION DESCRIPTION			CADD CHK RVW



AUTHOR		Aaron Zukley	
PROJECT		Autonomous Laser Tank	
TITLE		Robot X-Bee and Laser System	
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW		REV. 0	
PROJECT No.			
DRAWING:			

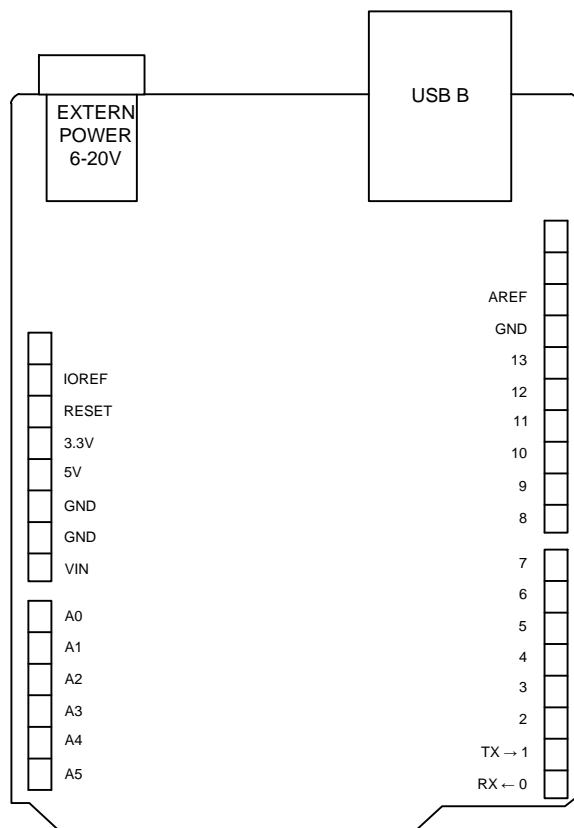
[illegible]

AUTHOR			
Aaron Zukley			
PROJECT			
Autonomous Laser Tank			
TITLE			
Robot Feedback			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW			REV. 0
PROJECT No.			
DRAWING:			

[illegible]


AUTHOR			
Aaron Zukley			
PROJECT			
Autonomous Laser Tank			
TITLE			
Robot Power Distribution			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW			REV. 0
PROJECT No.			
DRAWING:			

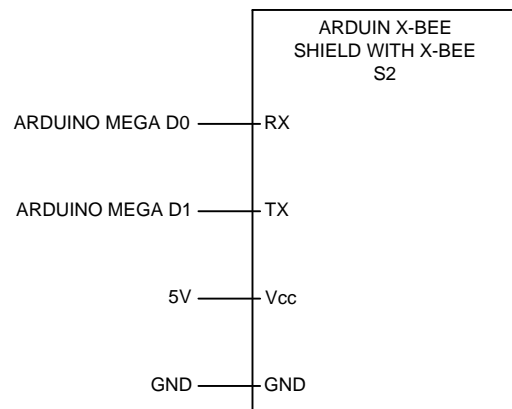
12/06/15	XXX	Rev 0	XXX		
REV	DATE	DES	REVISION DESCRIPTION	CADD	CHK RVW



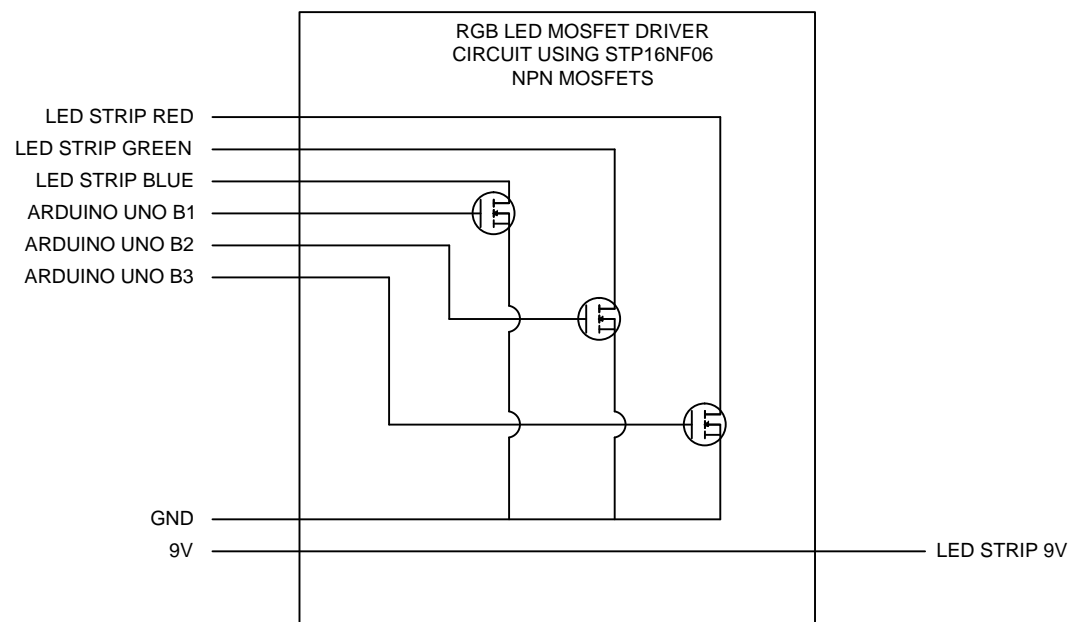
PORT	PIN	ARDUINO IDE PIN	FUNCTION
C	0	A0	CDS CELL 0
	1	A1	CDS CELL 1
	2	A2	CDS CELL 2
	3	A3	CDS CELL 3
	4	A4	CDS CELL 4
B	5	A5	CDS CELL 5
	1	9	PWM OUTPUT TO RED MOSFET
	2	10	PWM OUTPUT TO GREEN MOSFET
D	3	11	PWM OUTPUT TO BLUE MOSFET
	0	0	X-BEE RX
	1	1	X-BEE TX

AUTHOR			
Aaron Zukley			
PROJECT			
Autonomous Laser Tank			
TITLE			
Target Tower Arduino Uno			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW		REV. 0	
PROJECT No.			
DRAWING:			

	12/06/15	XXX	Rev 0			XXX
REV	DATE	DES	REVISION DESCRIPTION			CADD CHK RVW



AUTHOR			
Aaron Zukley			
PROJECT			
Autonomous Laser Tank			
TITLE			
Target Tower X-Bee			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW		REV. 0	
PROJECT No.			
DRAWING:			

[illegible]

AUTHOR			
Aaron Zukley			
PROJECT			
Autonomous Laser Tank			
TITLE			
Target Tower LED Strip Controller			
DESIGN	XXX	12/06/2015	FILE No.
CADD	XXX	12/06/2015	SCALE AS SHOWN
REVIEW			REV. 0
PROJECT No.			
DRAWING:			