CMPE 490: Final Report

Automated Life Preserver Launcher

Curtis Sand, curtissand@gmail.com

Zach Byrne, zbyrne@ualberta.ca

Declaration of Original Content

The design elements of this project and report ore entirely the original work of the authors and have not been submitted for credit in any other course except as follows:

The Point of Interest algorithm for finding the target in the picture was suggested by Dr. Elliott.

[1] The triangulation method described in the Design and Description of Operations Section.

[6] The Schematics for the Stepper Motor Control Circuit was provided by the L297 Data-sheet.

The wait() function in alpl.c was developed with the aid of Josh Ibach-Mackeen

The Taylor Polynomials for sine and cosine were acquired from Wolfram Alpha's website.

Abstract

When you're a fisherman off the coast of Labrador or a passenger on a Caribbean cruise there is always a chance of falling overboard. In a man overboard situation the two main things that another crew member must attempt to do are keep a finger pointed at the victim and to get a life preserver to the victim. We believe that there is room for improvement in this situation. To this end we developed a prototype of an Automatic Life Preserver Launcher (ALPL). The prototype utilizes certain simplifications such as a non-moving target and that a bright light and a standard digital cameras can stand in for infra-red cameras and an infra-red panic beacon. The ALPL uses two cameras and a technique called binocular range finding to calculate the distance to the target. A Nerf launcher is used as a stand in for a life preserver launcher. In the current iteration the target is assumed to be the brightest point of light in the picture; in order to ensure success at this stage, a relatively low level of ambient light is required. Overall the ALPL prototype is reasonably accurate considering the accuracy of the Nerf launcher and the simplifications that were used to make developing the system possible within the allotted four months.

Table of Contents

Functional Requirements

For the Automated Life Preserver Launcher to be considered a 100% successful project we would expect the following functionality: A button is pressed to alert the machine of an overboard crew member and the system automatically acquires the infra-red target, aims and accurately fires a life saving flotation device at the location of the crew member. Due to scope issues and availability of mechanical parts, we made a few simplifications and two very important assumptions to make a proof-of-concept prototype of this system attainable.

The first of the two assumptions that we made is that the target is stationary. Motion tracking, and compensation for motion in targeting would be a requirement if this were to be mounted on a boat; however, we were attempting to show that a system that uses an infra-red beacon as a target can calculate the distance of the target, determine an appropriate ballistic trajectory and fire a projectile with reasonable accuracy. The second assumption that we made is that the infra-red beacon that we will use is the brightest infra-red source within the view of the camera.

Some simplifications in the setup that we made include using a toy Nerf launcher to stand in for a life preserver launcher, a white LED to stand in for a human infra-red source, and that targeting time is not an issue. We wanted show that it is possible to use pictures from two parallel facing cameras to calculate the distance of an object and to subsequently calculate the appropriate trajectory for a projectile to hit the target.

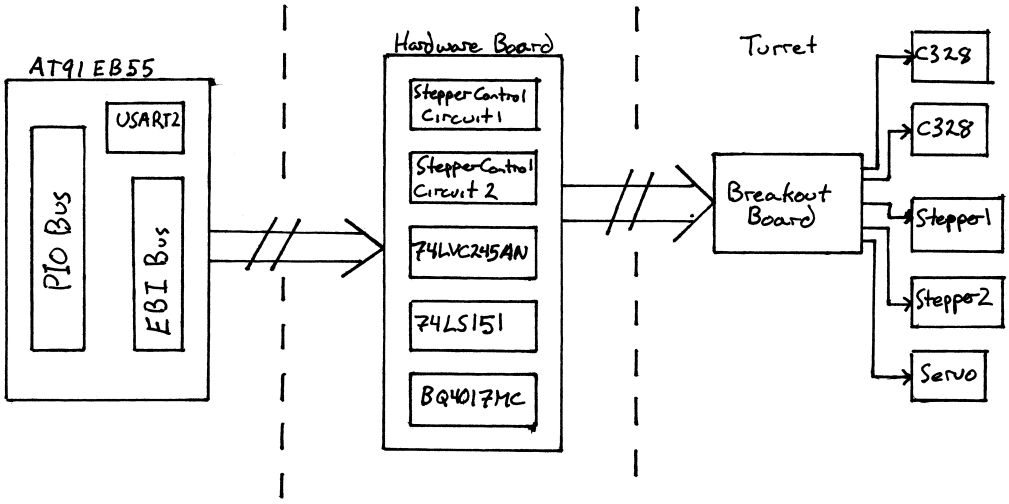
In the end we managed to successfully meet our self imposed minimum requirements. We were able to reliably find the target within the pictures as long as the target was the only point source of light in the picture and the ambient light level was fairly low. See the Design and Description of Operation for more information on this. From there the calculation of the distance to the target was fairly simple and was also fairly accurate as long as we were careful to keep the two camera's square with each other. In the best case we could calculate the distance within 3 or 4 centimeters of the actual distance, but in the worst case we were within a couple of meters.

The biggest source of error was the ballistic model that we used to calculate the angle of trajectory to fire the Nerf launcher. Due to time constraints the model that was used only produced an angle based on the distance to the target. Thus we ignored the height of the target with respect to the turret. We used experimental data to calculate a parabola representing the flight of the dart. This formula could easily be used to extend the model to account for the height of the target but we didn't manage to get this working in time.

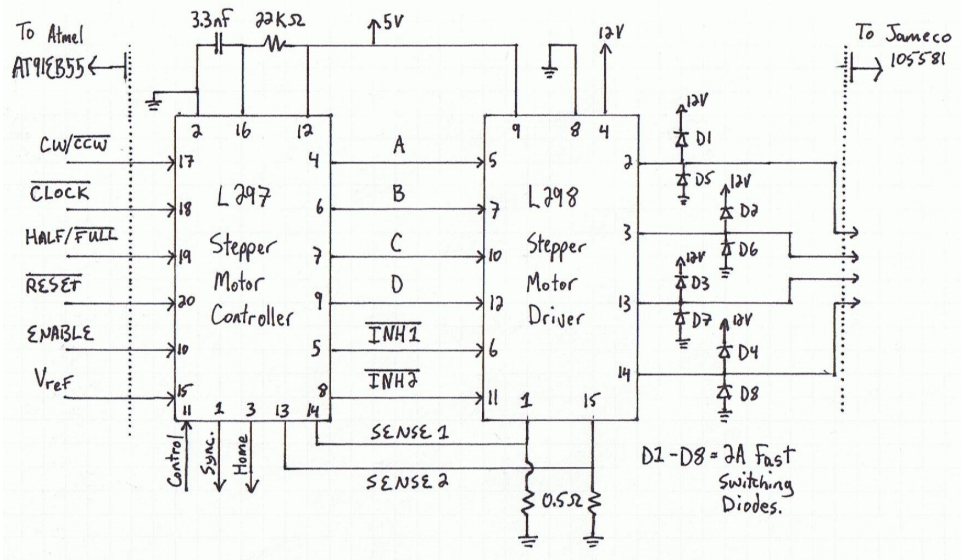
Overall we consider the project a success because, given our assumptions and simplifications we were able to reliably hit the target within a reasonable margin of error.

Design and Description of Operation

The Automated Life Preserver Launcher (ALPL) was broken down into three functional modules or sections. The first section is the interface between the two cameras and the micro-controller The goal of this section is to successfully acquire a single frame from each camera and store it temporarily in RAM on the micro-controller. The second section is a software section that analyzes the two images from the cameras and identifies the target, calculates the distance to the target and then, based on the location of the target, calculates the appropriate angle to aim the turret. The third section was the turret section which took a signal from the micro-controller and then adjusted the turret to the appropriate trajectory.



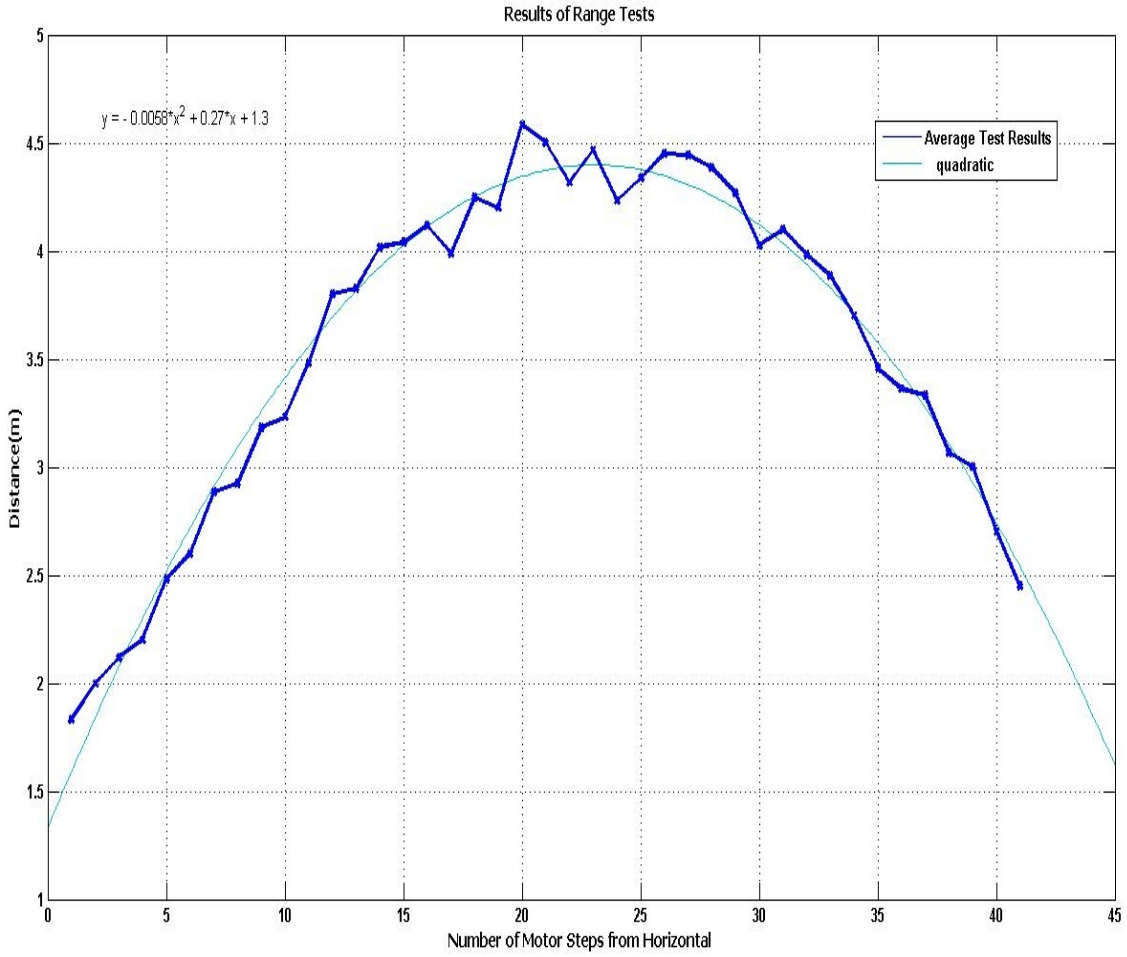
The ALPL System used the Atmel AT91EB55 micro-controller as its brains. Connected through a multiplexer to UART 2 on the Atmel micro-controller were the two SEN-09334 JPEG Color Cameras. Finally connected to the IO bus of the Atmel micro-controller were the 2 Stepper Motor Control Circuits and a servo. The control circuits (shown below) take input from the micro-controller and drive the stepper motors. For a full size view of the Stepper Motor Control Circuit see the Schematics section. Due to the possibility of the hobby servo sending up to 5V of noise back through its PWM line and that the multiplexer ran at 5V, both the servo's PWM line and the output from the multiplexer was put through a buffer that limited the voltage back to the Atmel micro-controller to 3.3V.



In order for the system to accurately calculate the distance of the target, the two cameras were securely mounted so that they faced in a parallel direction 28.5 centimeters apart from each other. In this way the difference in the location of the target in the images from the two cameras was used to calculate both the horizontal position and the distance to the target. The centroid of the light source was found by iterating through the pixels of each image one by one and finding the average coordinates of all pixel locations with a brightness above a threshold value. In the formula below d is the distance to the target. Offset is the number of pixels from center that the target is found in the pictures. P is a constant with units of Degrees per Pixel. P is found by dividing the Field of View of the camera by the number of pixels in the picture. This gives us a very rough measure of the real world from the camera. From the data-sheet of the SEN-09334 the Field of View is 57 degrees diagonally and the resolution of the picture is 80x60pixels. This gives us P=0.57. In general the number of degrees per pixel is not constant across the entire Field of View but it was accurate enough for our purposes.

$$d = \frac{l}{(\frac{1}{\tan(a)}) + (\frac{1}{\tan(b)})} \quad \text{where} \quad a, b = 90 - (offset \times P)$$

To calculate a ballistic model we gathered data about how far the Nerf launcher could shoot the dart at the different angles that the turret was able to produce. The raw data is provided below. Next we put this data into Matlab and used a curve fit algorithm to provide us with the formula for a parabola representing the distance the dart will fly given the angle of trajectory. See the Software Design section for more on how this parabola was used.



	Run 1	Run 2	Run 3					
Steps	Dist.	Dist.	Dist.	Additional Shots to Minimize				Average Di
0	1.95	1.75	1.80	Effect of Odd Datapoints.				1.83
1	2.00	1.80	2.20					2.00
2	2.10	2.00	2.25					2.12
3	2.30	2.10	2.20					2.20
4	2.45	2.40	2.60					2.48
5	2.60	2.60	2.60					2.60
6	3.00	2.95	2.70					2.88
7	3.10	2.80	2.80	3.00				2.93
8	3.20	3.15	3.20					3.18
9	3.30	3.20	3.20					3.23
10	3.60	3.45	3.40					3.48
11	4.00	3.60	4.00	3.60				3.80
12	4.00	3.55	3.95	3.80				3.83
13	4.05	4.00	4.00					4.02
14	3.60	3.80	4.45	4.25	3.95	4.20		4.04
15	4.20	4.10	4.05					4.12
16	3.95	4.00	4.00	4.00				3.99
17	4.20	4.35	4.20					4.25
18	3.80	4.05	4.25	4.50	4.15	4.45		4.20
19	4.50	4.80	4.45					4.58
20	4.50	4.60	4.40					4.50
21	4.30	4.30	4.20	4.50	4.40	4.20		4.32
22	4.40	4.50	4.50					4.47
23	4.40	4.30	4.05	4.10	4.30	4.25		4.23
24	4.20	4.40	4.45	4.30				4.34
25	4.30	4.55	4.45	4.50				4.45
26	4.40	4.80	4.60	4.20	4.45	4.20		4.44
27	4.45	4.20	4.50					4.38
28	4.20	4.20	4.40					4.27
29	3.70	4.00	4.40	4.00				4.03
30	4.20	4.25	4.20	3.80	4.20	3.95		4.10
31	4.00	3.95	4.00					3.98
32	4.05	3.80	3.80					3.88
33	3.75	3.75	3.60					3.70
34	3.40	3.60	3.60	3.20	3.55	3.40		3.46
35	3.40	3.20	3.60	3.25				3.36
36	3.40	3.25	3.35					3.33
37	3.40	2.80	3.00					3.07
38	3.00	2.80	3.20					3.00
39	2.80		2.60					2.70
40	2.45							2.45

results in:

$$distance = -0.0058 \times steps^2 + 0.27 \times steps + 1.3$$

The design of the turret was a simple two axis design so that there were both azimuth and elevation articulation. The turret's initial position was determined by assuming that the system would be started up with the azimuth at it's center and the elevation at it's lowest possible angle. Thus when the system started up the first task was to raise the launcher to horizontal by moving the elevation motor up 21 half-steps. To fire the Nerf launcher we used a hobby servo lined up in such a way that rotating the arm would press against the mechanical trigger that was on the Nerf launcher. To reload the Nerf launcher we were required to physically pull the plunger back.

Parts List

The datasheets for all of the parts will be included with the source code in the accompanying tarball.

- 2 digital cameras: C328-7640(sparkfun:SEN-09334) JPEG Color Camera with a UART Interface
  - $54.95US from sparkfun.com
  - data-sheet: http://www.sparkfun.com/datasheets/Sensors/Imaging/C328.pdf
  - manual: http://www.sparkfun.com/datasheets/Sensors/Imaging/C328_UM.pdf
  - requires some parts (from digikey) for making connections:
    - WM-8289-ND -- 2mm connection Receptacle housing (4 position female) http://parts.digikey.com/1/parts/1210042-conn-rcpt-hsng-2mm-4pos-single-51090-0400.html
      - $0.33 from digikey.com
    - WM-6050-ND -- crimps, female 24-30AWG, 2mm http://parts.digikey.com/1/parts/271044-conn-term-female-24-30awg-tin-50212-8100.html
      - $3.54/10units from digikey.com
- 2x Stepper Motors
  - Jameco 105581 12V Stepper Motor
  - The Data Sheet is no longer available online so see Nancy Minderman.
- 1x servo
  - Hitec HS635HB:
    - data-sheet: http://www.hitecrcd.com/product_file/file/56/HS635.pdf
    - User Manual: http://www.hitecrcd.com/product_file/file/55/Servomanual.pdf
- 2x L297 Stepper Motor Controller
  - Data-sheet: http://www.st.com/stonline/books/pdf/docs/1334.pdf
- 2x L298 Stepper Motor Driver
  - Data-sheet: http://www.st.com/stonline/books/pdf/docs/1773.pdf
- 1x sn74LVC245AN Buffer IC
- 1x 54/74151A Multiplexer IC
- 1x bq4017mc-70 2Mb SRAM IC

Datasheet

Min. Range: ~1.8 meters

Max Range: ~4.5 meters

Max Horizontal Articulation: 340 degrees.

Min Horizontal Articulation (1 half step): 0.6 degrees

Max Vertical Articulation: -28 degrees to 72 degrees from horizontal

Min Vertical Articulation (1 half step): 1.8 degrees

Width of the cameras: 28.5 centimeters

Power Source Data:

Vss	Min. Current Draw	Avg. Current Draw	Max.Current Draw	Max. Power Draw
3.3V	0.6mA	13mA	30mA	99mW
5.0V	130mA	160mA	270mA	1.35W
12.0V	0.0A	0-.75A-1.3A	1.5A	18W
				19.5W

- Note: The total Max Power Draw of 19.5W ignores the power drawn by the Atmel Micro-controller.
- According to the DC power supply for the Atmel Micro-controller it can draw a maximum of 6.0V * 600mA = 3.0W

System I/O to/from the Microcontroller

Camera1-TxD	PA21TXD2
Camera1-RxD	PA22RXD2
Camera2-TxD	PA21TXD2
Camera2-RxD	PA22RXD2
Camera-Select	PB18
Stepper1-CW/CCW	PB2
Stepper1-Clock	PB6/AD0TRIG
Stepper1-Half/Full	PB3/IRQ4
Stepper1-Reset	PB4/IRQ5
Stepper1-Enable	PB0
Stepper2-CW/CCW	PB2
Stepper2-Clock	PB7/AD1TRIG
Stepper2-Half/Full	PB3/IRQ4
Stepper2-Reset	PB4/IRQ5
Stepper2-Enable	PB1
Servo-Control	PA23/TIOA1
RAM-CE	CS2

RAM-WE	NWRO/NWE
RAM-OE	NRO/NOE
RAM-D0	D0
RAM-D1	D1
RAM-D2	D2
RAM-D3	D3
RAM-D4	D4
RAM-D5	D5
RAM-D6	D6
RAM-D7	D7
RAM-A0	A0/NLB
RAM-A1	A1
RAM-A2	A2
RAM-A3	A3
RAM-A4	A4
RAM-A5	A5
RAM-A6	A6
RAM-A7	A7
RAM-A8	A8
RAM-A9	A9
RAM-A10	A10
RAM-A11	A11
RAM-A12	A12
RAM-A13	A13
RAM-A14	A14
RAM-A15	A15
RAM-A16	A16
RAM-A17	A17
RAM-A18	A18
RAM-A19	A19
RAM-A20	A20

Software Design

The software for our system is fairly self explanatory. The system is split up into a number of separate libraries that are used by the main.c. There is only one internal interrupt used in the system. All the interrupt handler does is post to a semaphore that is also used by our wait() function, which sets up a timer and counter in order to pause for a given number of clock cycles. This wait() function is used by any part of our system that requires delays in execution. We found that the provided at91_wait_open() function was too unreliable to use successfully for the final demo. The libraries that we've created are each in charge of a particular part of the system. Also we have one library that is a catch all for functions that didn't have an obvious place elsewhere in the system.

The main() function initializes all of the I/O lines and then pauses for 8 seconds before calling run_alpl() which runs the system. The reason we pause for 8 seconds is because we had de-bounce issues with the push buttons on the board. Once run_alpl() is called it follows the following step by step algorithm:

1. place the turret into position 0,0.
2. synchronize camera 1
3. initialize camera 1
4. take a picture with camera 1
5. remove the header from the picture
6. find the centroid in the picture
7. find the angle offset of the centroid from center
8. free the space used by the picture
9. move the azimuth motor to center the Nerf launcher on the target
10. synchronize camera 2
11. initialize camera 2
12. take a picture with camera 2
13. remove the header from the picture
14. find the centroid in the picture
15. find the distance to the target
16. calculate the number of steps above horizontal required to reach the target
17. move the elevation motor to the calculated angle
18. fire the Nerf launcher
19. move both stepper motors back to position 0,0
20. return

Due to the fact that the Atmel micro-controller does not have floating point functional units and there wasn't a math library readily available we opted to write our own exponential, sine, cosine and tangent functions so that we could do the required math in software. For the sine, cosine functions we used their respective Taylor Polynomials to calculate the values. The tangent function used a floating point division of the sine and cosine functions.

$$sine(x) = 1.0 - pivot^2/2.0 + pivot^4/24.0 - pivot^6/720.0 + pivot^8/40320.0$$

$$cosine(x) = 0.0 - pivot + pivot^3/6.0 - pivot^5/120.0 + pivot^7/5040.0 - pivot^9/362880.0$$

$$where\ x\ is\ degrees,\ pivot = (x \times \pi/180) - \pi/2$$

To calculate the distance we created a small lookup table using the formula for the parabola that is described in the Design and Description of Operation Section. The table was essentially an array of about 40 integers. The index of the array was the number of half-steps above horizontal and the contents of the array was the distance to the target in decimeters. In this way the software was able to quickly search the lookup table for a best solution to decide how to aim the turret in order to hit the target.

Test Plan

Our test plan utilized the modular design of the system so that we were able to test each of the subsystems separately. The cameras were tested on a separate solder-less breadboard, as was the servo. The stepper motors were tested using the DC Power Supply, the Signal Generator, and the Oscilloscope. Even the multiplexer was tested using the DC Power Supply and the Signal Generator and the Oscilloscope. Once the functionality of each of the separate subsystems were verified they were wired together on the same prototyping board. Next a series of software tests were used to test physical systems by rotating the servo and stepper motors according to user input. The software was tested ad-hoc using a number of test cases. We found that the sine, cosine and tangent functions were least accurate around an angle of 0. After integrating all of the subsystems we tested the system as a whole by running it in various light conditions and with the target at various distances.

Results of Experiments

After reading about the range finding formula we decided to try a dry run through. Both of us have Asus eeepcs with the same camera mounted on the screen, so we set them next to each other and took a single picture (at 640x480 resolution) of a bright white LED. Next we wrote a python script (see appendix for source code) that implements the point of interest search for the brightest spot in the jpeg described in the Design and Description of Operation Section. Next the script takes the x location of the LED in each picture and uses the formula described in the Design and Description of Operation section to find the distance to the LED. The distance between the cameras was measured at 27.5cm and we physically measured the distance to the LED to be 1.91m. At first we used the value of P that we calculated for the SEN-09334 camera and the script returned a result of 3.2m as the distance to the LED. After searching for a more appropriate Field of View (75 degrees horizontal) for the camera that was used we tried the script with the value of P changed to 0.12 degrees per pixel. After this change the script returned 1.87m. Thus we are confident that we can calculate the distance to a target with reasonable accuracy given a picture from two cameras mounted parallel to each other.

Citations

[1] Triangulation, Available: http://en.wikipedia.org/wiki/Triangulation, Last Modified: February 2nd, 2010, Last Accessed: February 4, 2010

[2] Hitec HS422 Data-sheet, Available: http://www.robotshop.ca/PDF/hs422.pdf

[3] Hitec HS635HB Data-sheet, Available: http://www.hitecrcd.com/product_file/file/56/HS635.pdf

[4] COMedia SEN-09334 JPEG UART Camera Data-sheet, Available: http://www.sparkfun.com/datasheets/Sensors/Imaging/C328.pdf

[5] L298 Stepper Motor Driver Data-sheet, Available: url=http://www.st.com/stonline/books/pdf/docs/1773.pdf

[6] L297 Stepper Motor Controller Data-sheet, Available: url=http://www.st.com/stonline/books/pdf/docs/1334.pdf

Quick Start Guide

1. Flash the Atmel micro-controller with the provided alpl_ROM (or alternatively alpl_ram) ELF executable. To do this follow the tutorial written by Nancy Minderman for the 2010 Winter CMPE 490 class.
2. Connect the Wire Wrapped Prototype Board to the Atmel micro-controller with ribbon cables. Only the JPAD/DA, JPCOMM, JPTIMER headers need to be connected with this version of the software.
    - Set channel 1 of the DC Power Supply to 3.3V and limit the channel to around 0.50A. Connect the positive line of channel 1 to the Blue power post on the Prototype Board.
    - Set channel 2 of the DC Power Supply to 12.0V and limit the channel to around 1.60A. Connect the positive line of channel 2 to the Red power post on the Prototype Board.
    - Connect the positive line of channel 3 (5.0V limited at 3.0A) to the Green power post on the Prototype Board.
    - Ensure that the Ground connection and the Negative line of all three channels on the DC Power Supply are common. Then connect this node to the Black power post on the Prototype Board.
3. Connect the ribbon cable between the Turret and the Wire Wrapped Prototype Board. Pin 1 on the breakout board (glued to the turret) is labeled as the pin at the back left with respect to the Nerf Launcher. Pin 1 on the Wire Wrapped Prototype Board is in the top left or closest pin to the power posts.
4. Orient the turret so that the pin on the rotating structure lines up with the black mark at the center of its rotation and that the Nerf Launcher is loaded and resting on the stop (pointing as far down as it can).
5. Finally double check all the connections, turn on the DC Power Supply and then plug in the Atmel micro-controller. The software will count to eight and turn on all of the eight LEDs on the micro-controller if it is working properly.

Future Work

Additions and extensions that could be added to the ALPL are many. Due to the use of infra-red light for targeting this type of system could be used to fight fires, or track vehicles with a video camera. An obvious next step from using an infra-red beacon and and standard digital camera would be to use an infra-red camera and image detection to recognize a human. As mentioned earlier, motion tracking would be a logical next step for a system such as this. Another extension that could be added would be an actual life preserver launcher with a cable and a winch for retrieving the target and multiple target recognition and prioritization so that a multiple man-overboard situation could be dealt with.

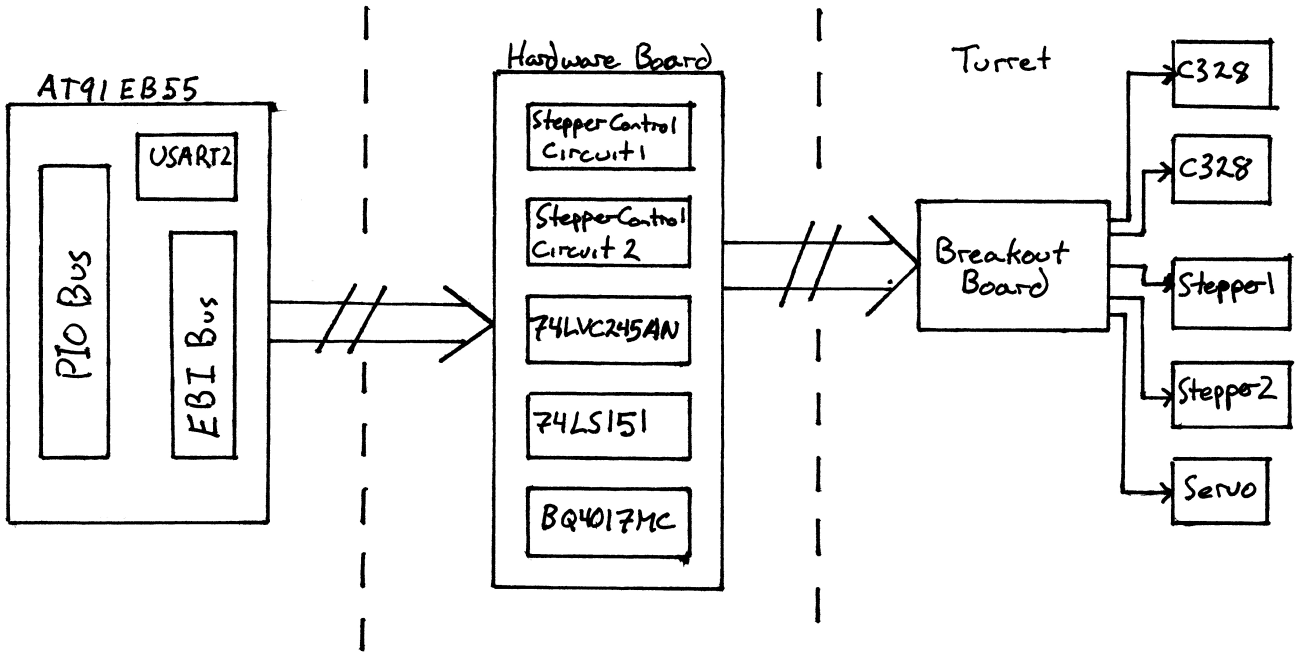
If we were to continue developing the system from its current state, a few of the modifications that we would make first are: better ballistic model, RTOS to speed up the execution time of the system. Next we would work on a better algorithm for locating the target within the picture. For example we could tune the algorithm to find multiple distinct light sources within the picture and then from there identify the actual target by its specific size and/or color. The implementation of the RTOS would allow us to continually take pictures and keep track of the location of the target. This would facilitate a faster execution time because when the system is triggered the system could already know the distance to the target and all that would be required is to aim and fire the turret. An RTOS could also allow us to

implement tracking of the object over time because the continual data from the cameras would allow us to track the target over time.

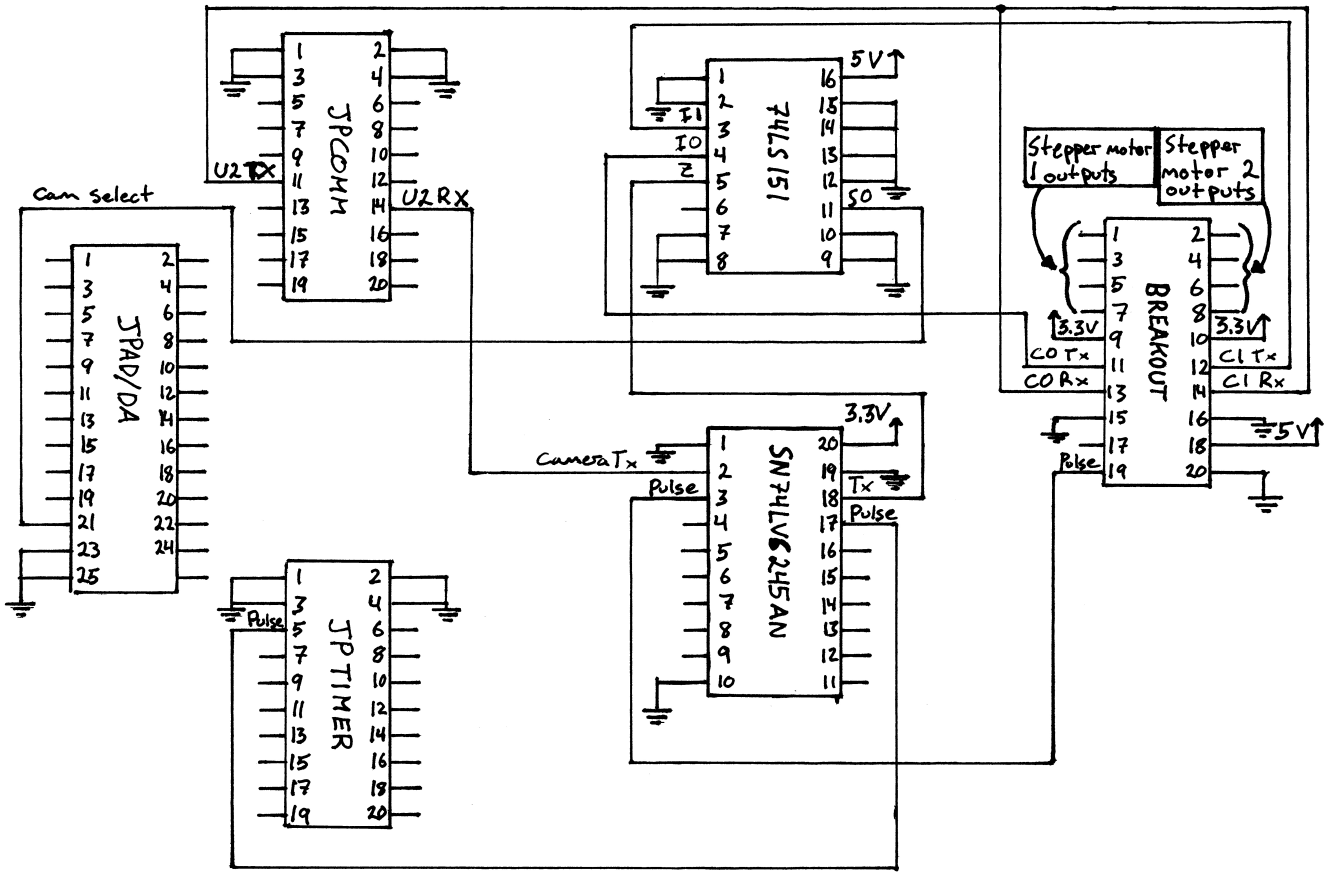
Other things that we would be nice to add would be a slot sensor for each of the stepper motors so that we could automatically determine the orientation of the turret. Also a mechanism by which we could reload the Nerf launcher would be a nice feature so that we could deal with the multiple man overboard situation mentioned earlier. Another big improvement would be to add some sort of gearing system or belt/chain drive to the elevation motor so that we would be able to achieve both smoother motion and increase the resolution of the elevation motor to achieve a more accurate trajectory angle.

Hardware Documentation

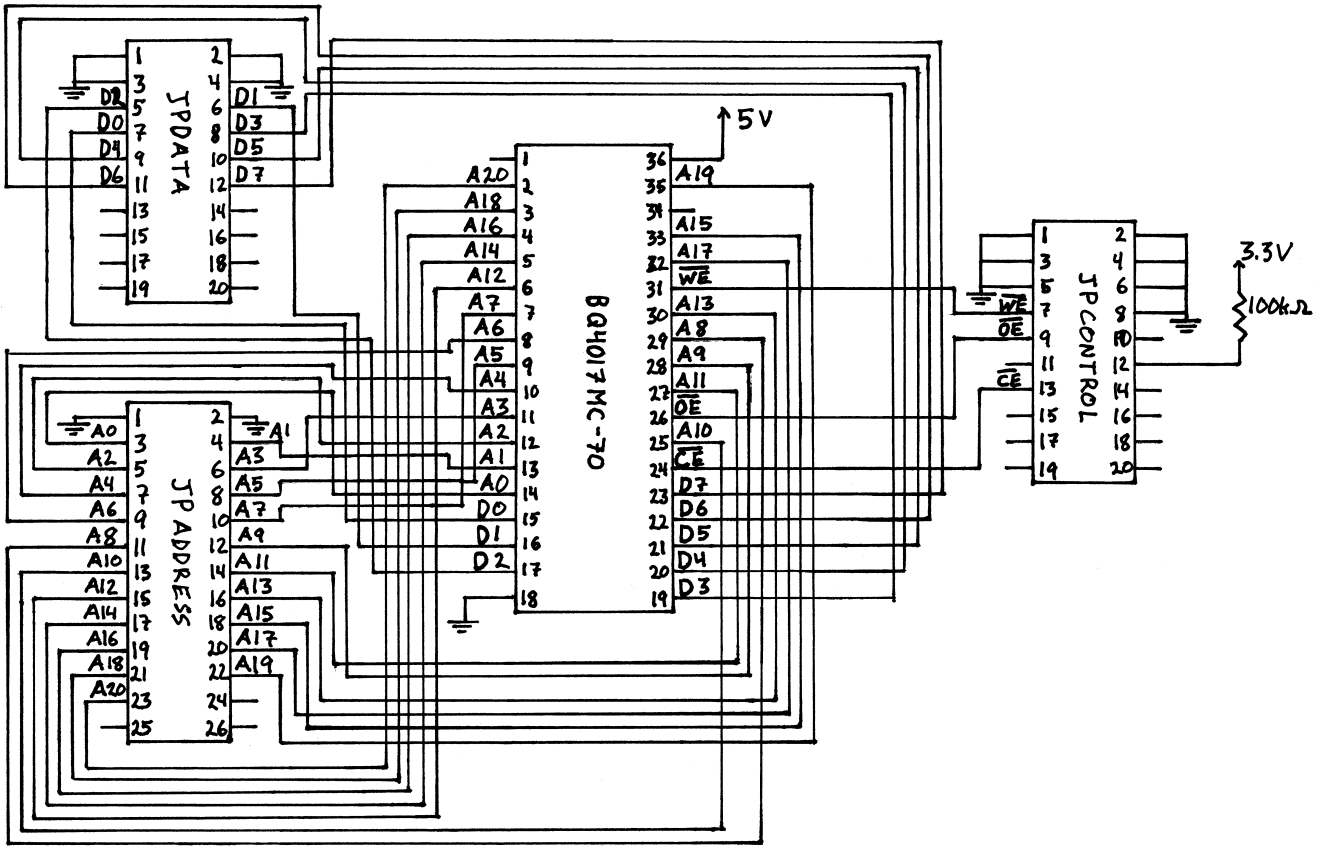
High Level Hardware Block Diagram



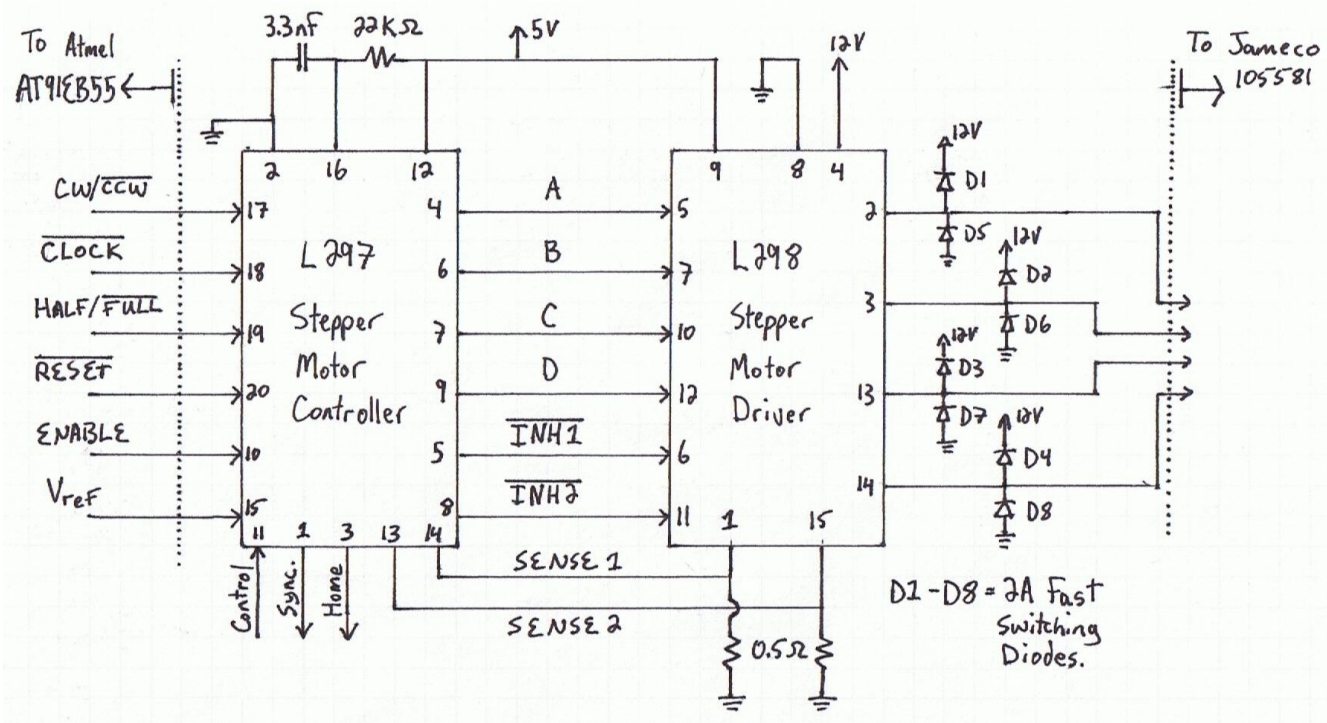
Camera and Servo Wiring Diagram



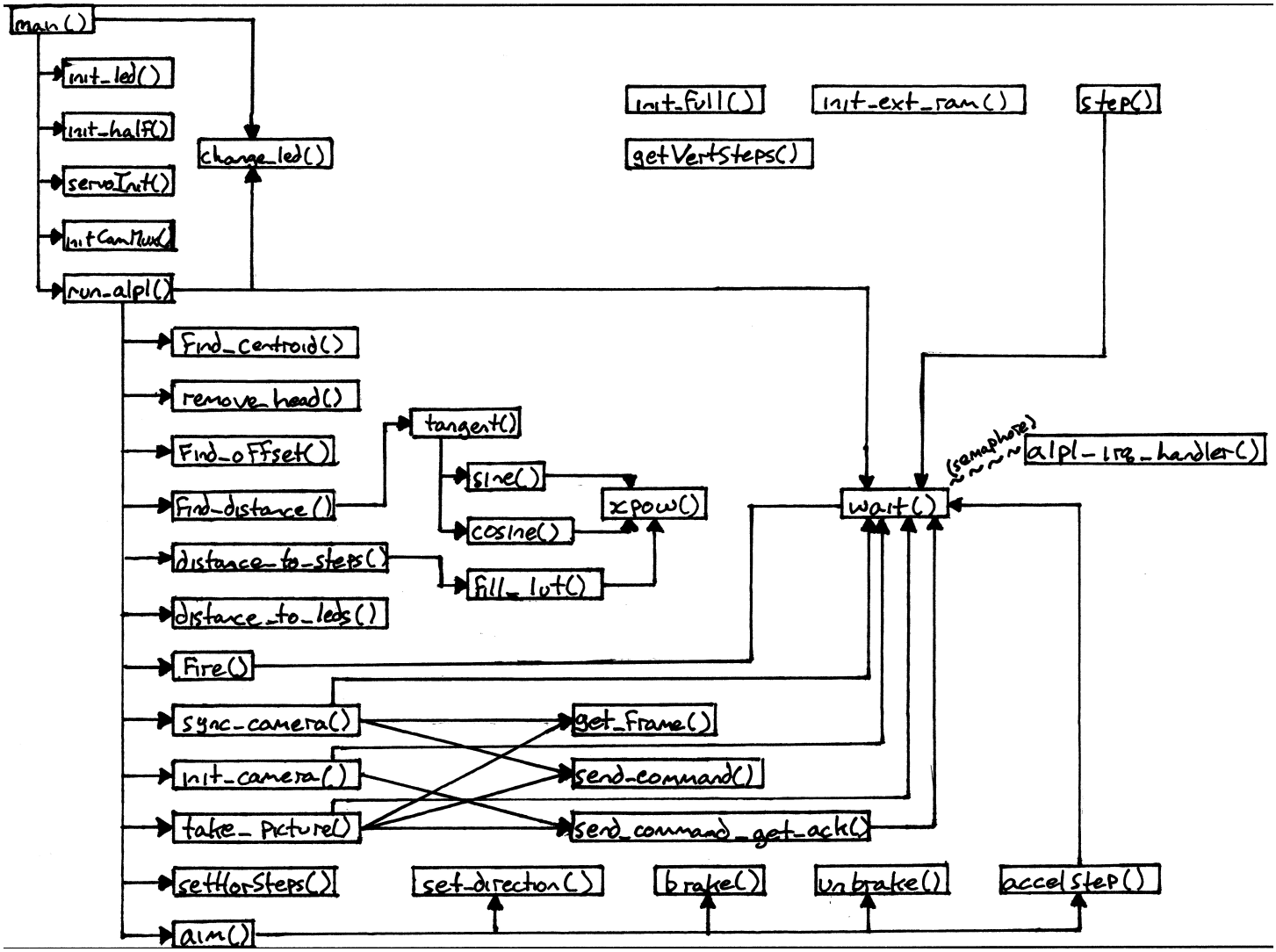
External Ram Wiring Diagram



Stepper Motor Control Circuit Diagram



Graphical Hierarchy of Source Code



Source Code Index

main.c: Executes the high level system algorithm.(T)

alpl.h: defines a number of system wide constants. (T)

alpl.c: general library for functions that didn't fit into their own library. (T)

camera.h: defines a number of values for the camera driver. (T)

camera.c: the camera driver. (T)

image.h: defines a number of values used in the image driver (T)

image.c: contains the functions that manipulate the images from the cameras (T)

servo.h: defines values used by the servo driver (T)

servo.c: the servo driver (T)

stepper.h: defines values used by the stepper motor driver (T)

stepper.c: the stepper motor driver (T)

tests.c: contains a number of functions that were used in testing the systems (T)

rangeFind.py: Test program for the range finding algorithm. (T)

resize: a C program to expand the memdump from the Atmel Board into simple RGB data (T)

develop.py: Test program to turn rgb data from resize into a jpg. (T)

spot.py: a script that searches a jpeg for the average position of bright pixels above a threshold. (T)

makepic: a bash script that calls wget, resize and develop.py to automate developing process. (T)