

Final Report for Valet-bot

By David Wynthacht

**University of Florida
Electrical and Computer Engineering
IMDL, EEL 5666C
Summer 2003**

Table of Contents

Abstract	- 3 -
Executive Summary	- 4 -
Introduction	- 6 -
Integrated System	- 7 -
Mobile Platform	- 8 -
Special Platform	- 10 -
Actuation	- 10 -
Sensors	- 11 -
Behaviors	- 12 -
Experimental Layout and Results	- 13 -
Conclusion	- 15 -
Documentation	- 16 -
Appendices	- 17 -
Appendix A	- 17 -
The main program for object avoidance and color following.....	- 17 -
Appendix B	- 25 -
The code for the space finding and parking.....	- 25 -
Appendix C	- 30 -
The header for the servos.....	- 30 -
The code for the servos.	- 31 -
Appendix D	- 33 -
The header for the LCD.	- 33 -
The code for the LCD.....	- 34 -
Appendix E	- 37 -
The PCB schematic.	- 37 -
The PCB layout.	- 38 -
Appendix F	- 39 -
The AutoCAD board 1:	- 39 -
The AutoCAD board 2:	- 40 -
Appendix G	- 41 -
The Parts list:	- 41 -

Abstract

The topic of this paper is the conclusion of a project designed to use computer vision to assist people in find parking spaces and parking the car for them. This paper will show how this task was accomplished and the various methods used to complete this task. In addition the various behaviors to both test and demonstrate that robot is functional, such as color following and object avoidance. The paper will discuss the use of IR and bump to understand the world around the robot as well as using a CMU cam as the computer vision portion of the project. The platform and its consideration are also discussed below. Also, the future of were this robot can be improved upon is include. This system, although is not portable to a real car, is the first step in to a journey where parking frustrations maybe a thing of the past. As well as, a lot small fender benders could be voided.

Executive Summary

The purpose of the robot is to find a “Parking space” and park in that space. The robot also has two other behaviors, one of which is follow a color object around, and the other is to avoid object as the robot moves forward.

Small description of the robot; in order to do avoidance “bump” sensors and IR sensors are used to detect the objects around the robot. Computer vision is used to find the parking space via color detection. If the camera can see the color of the garage wall then there is no car there. Parking is handled via preprogrammed movement. The processor that controls it all is the Atmel Mega 128 found on the LetAtWork board. This board is used as the heart of the robot.

Special Sensor description is such; the sensor is the CMU Cam. The CMU cam will be used to find the parking space using color detection. The CMU module has, microcontroller, power regulation, serial communication port via TTL or RS232 at 115k baud, and a color camera. Integration of the sensor delta with the mounting, rotation, and a clear field of view. This was accomplished by placing the camera on the top of the robot connected to a servo.

The sensor data is in an ASCII string that contains various data about the tracked object. For this project # of pixels is used to see if an object is being tracked or not. While the mx value I can tell if the tracked color is to the right, left, or center of the vehicle.

Future adaptation with the current platform design can be used for many more behaviors. Line following or light following can be implemented with no change to the

board and minimal change to the platform. Other Color related behaviors can be added with external change to the system. To improve the reliability of the camera lights might be added to the outside of the camera enclosure. This would send a more uniform light and perhaps more consistent values as the robot changes venue.

All in all the robot meets the goals set out for it, all that remains is to clean up the few loose end left at the end of the semester. The simplistic nature of the robot design and code made of a successful robot with plenty of room to grow.

Introduction

Have you ever been in a parking lot with no time to get where your going, and find that parking lot filled? Have you ever thought that having an extra pair of eyes searching for that elusive spot? No this isn't a metaphor, this is a real situation that at some point we all face. Well it's not a pair of eye's but it can look. It's my robot, trying to make the impossible seem possible.

The main idea of my robot is to find a parking space and park in that space for the robot owner, an automated Valet if you will. So some of the tasks the robot needs to accomplish are to find an available space, save that space. The robot will traverse the parking lot searching for a space to park in. The parking space will be identified by using computer vision. Parking will be handled by more traditional sensors. The robot will also have the traditional object avoidance and then it will follow colored objects. If a human needs to direct the car where to go the this function will suffice.

The remaining text will provide the insight on how this robot accomplished its tasks. First we'll look at how the system is put together, then will look at the structure of the robot. Next, we'll look at accessories need to demonstrate the robots abilities, then a look at the moving parts of the robot and how it gets around. From there we'll examine how the robot sees it's world, and then move in to how the robot interacts with our world. Lastly, we'll look at how the robot preformed and maybe what we can do to improve the robot. That should cover all the operations and goings on of the robot.

Integrated System

There will be many integrated systems on the robot. Of course, integrating object avoidance with the other behaviors is one. Also, integrating the CMU cam with the main processor was one of the most challenging tasks. Also the platform with the computer hardware was a new experience.

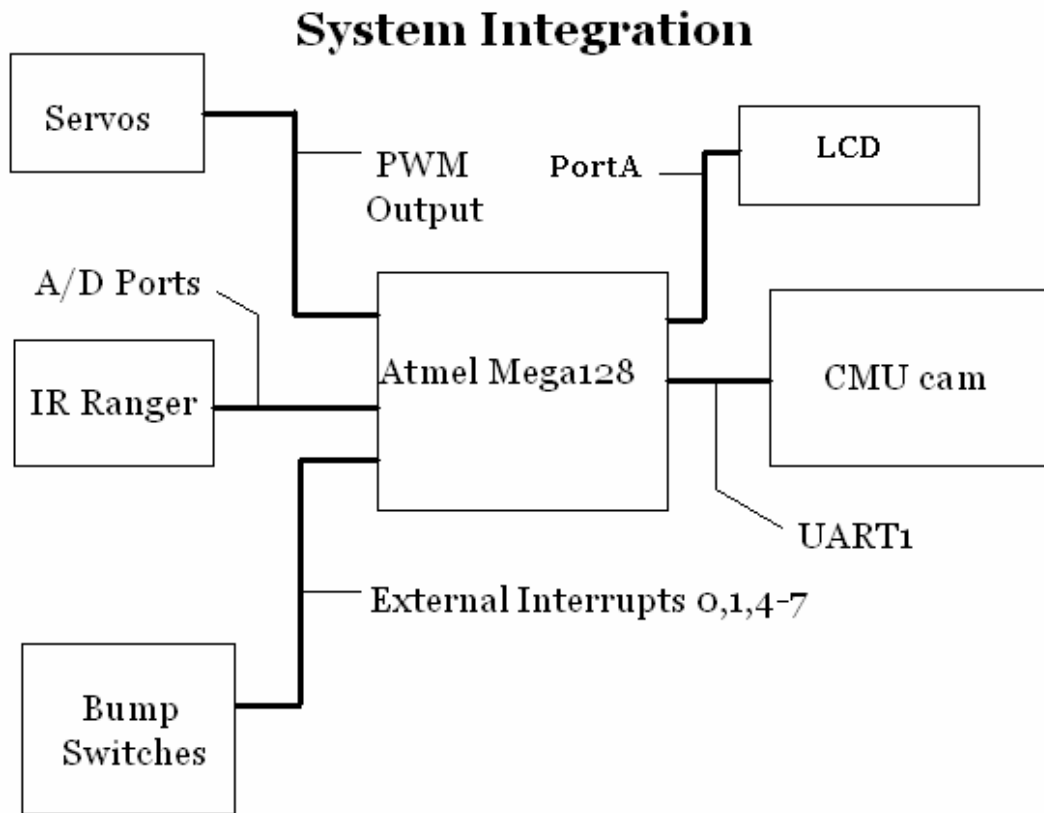
Starting with the frame or platform, this had to serve the project for all aspects of its operation. The main concerns were finding a balance between the wheels and support bolt; placement of the CMU cam and dealing with its movement; and creating a car like structure that allowed for the behaviors like object avoidance. Mostly, the platform was kept very simple except for the bumpers. Though they were not overly complicated they were the hardest thing to properly incorporate.

The control system was structured around the LetAtwork II module. This is a single processor design (excluding the vision process), that kept the foot print small and organized. Creating a PCB board helped reduce the errors in the system as well as keep the system orderly. See appendixes G & H for the PCB layout and schematic. The processor handled all of the servo, bump, LCD, input, communications, and program execution.

The vision system was handled completely by the CMU cam, which housed the only other processor. Simple commands were relayed via serial communication and then processed data was fed back to the main processor via the same serial link.

An LCD was used as output and user interaction, along with the bump switches. The bump switches allow the user to choose which program they would like to execute. This simple setup can be used for all kinds of user menus or program selection.

The integrated systems were implemented as shown in the following diagram:



(c) 2003 David Wyancht

Mobile Platform

The platform was made out of balsa wood with some wood glue. Most of the pieces either fit by friction or had a lock system in place. The shape was more of a car shape, a circle elongated with front and back bumpers and straight sides. The profile

looked much like any car. The inside was mostly hollow which allowed for more room for electronics. The bulk of the weight rested on the bottom piece which was supported by two servos with wheels and on 5/16 bolt. This allowed the general structure and design to be simplistic. The pieces had an interlocking design so very little gluing was needed. For the bottom piece two bars were used to lock the bottom on. These bars attached the side and gravity coupled with friction kept the bottom secure. The rest held together by friction forces. See appendixes I & J for the AutoCAD layout.

The platform was finished off by spray painting the about three coats of red paint on the external surfaces. The internal surfaces were sprayed with an ultra flat black which turned out to be conducted with resistance in the range from 100kΩs. to 1MegaΩ depending on the distance. This did cause some problems with the bump switches, so the bump switches needed to be isolated from the paint. The external surfaces were also sprayed with a clear coat to add a glossy look. The bumpers and wheel hubs were painted silver to give a chrome look and also to match the bolt support and some of the screw that were visible. This gave a nice appearance to the robot, kind of sports car look.

Things to watch out for on the next platform construction are cutting out a notch for the servo wires so you don't have to take the servo apart to place or remove the servo from the robot. Having a specific battery placement system, and make the external port a little more accessible, for example UART0 should have been a little closer to the edge of the robot. Also, make a holder for the robot when you are working on the internal components this way undue stress will not be exerted on sensitive parts. Just a few things to keep in mind the next time around.

Special Platform

A second platform was constructed for the purpose to act as a parking garage. This was constructed out of partial board the size of a sheet of plywood. This was sawed in half and one of the halves was sawed in thirds. The thirds were attached to the other half in a perpendicular fashion to give the walls of the garage. The walls were attached using some nails, wood glue and masking tape. The floor of the garage was spray painted white and the inside of the side were painted blue. The blue was found to show up better with the CMUcam. For the purpose of the demonstration cardboard boxes were used to simulate other cars. The platform served it's purpose well.

Actuation

The actuation consists of 3 servos. Two servos are used to move the robot around and one servo for the camera. The camera servo is used to look to the right and left of the robot. The two servos for movement are hack for continuous rotation. The third servo is unhacked and only needs to turn 180 degrees. Most cars have four motors for locomotion and one to steer the car. This excludes the window motors and the solenoids to start the car and unlock doors, etc. In my design I have chosen to use only two wheels for both motion and direction. This differential system is simpler to implement in the time provided. Locomotion and rotating the camera is the only actuation needed to complete the goals of the system. See appendixes C & D for the servo code.

Sensors

Valet-bot has an array of six bump switches, three IR rangers, and a CMU cam. The sensors are used to “see” object in front and to the side of the robot, feel anything that contacts the front and back of the robot, and to find a vacant parking spot.

The tactile sensor are “bump” switches connected to the external interrupt pins of the microprocessor via a pull down network. Each bump switch has its own interrupt associated with it. The bump switches also double as a buttons to select which program to execute.

There are two type of vision sensors, the first of which is the IR rangers. There were two GD2P120 and one GD2P12. The 120’s are short range detectors with a range of about 2 to 24 inches. The 12 is a long range from about 5 to 48 inches. The 12 was set in the center of the robot to look for objects directly in front of the robot. The 120’s were mounted on the front side edges of the robot on about a 30 degree angle from the side edge. This would allow the sensors to see both what’s next to the robot but also what is in the front but off to the side of the robot. The combination of the angle and the short range allow the robot to get in to tight space like a parking space. The data that the IR rangers send is an analog voltage level indicate the distance of the reflecting object. This is connect to the A/D port so the data can be converted to integers and thus be manipulated in the program.

The last sensor is the CMUcam color detecting and tracking sensor. The CMUcam is mounted on the end of a servo at the top of the robot. This allows the robot

to look around. The CMUcam is controlled and sends data serially. The data is in the form of ASCII characters, and in several formats. I chose the M packet format because this format gave me the center of a color blob in the x coordinate and how large the color blob was. These are the value that the program looks at to determine where it needs to go or if there is a parking space open. I have made a more detail analysis of the CMUcam and how I used it with Valet-bot titled “Sensor for Valet-bot”

Behaviors

The robot has four main behaviors that dictate it's existence. The first behavior is to avoid objects in it's environment. The robot will follow a colored object around demonstrating that the camera can be controlled and it's data used to direct the robot. Next the robot will find parking spaces using computer vision. After the robot locates a space the robot will park in that space. All of these behaviors should combined to create a complete robot that will serve it's function as a valet.

The first behavior object avoidance uses the IR rangers and the bump switches to interpret its environment. It will always try to go forward until either an IR sensor detects something too close to the robot or a bump sensor is triggered. If a bump sensor is tripped it operate on an external interrupt so not matter what the robot is doing it will react to the bump appropriately. However, the IR sensors are connected to the analog to digital converter; because of this the IR sensor must be scanned continually to see if something is close by. If something is detected via the IR system the robot will veer away from the detected object.

The second behavior is tracking and following a color object. This behavior starts out by setting up the camera and then waiting for the color to come before it. When the color is spotted (found by the pixel value above 0) then the robot looks at the center point and determines if it need to go left right or forwards. If the robot losses the tracked color then the robot stops and waits for it to again pass in front of it.

The next behavior is finding a parking space. This is accomplished by starting at the beginning of the parking lot then the robot turns the camera to the right and takes values from the camera if it detect the color blue then in parks in that spot. If it does not find the color then it looks to theleft and again takes readings to determine if the color is present. If it is present then it will park in that location, if not then it will go forward to the next space; currently a specific distance; and again start the procedure all over again. This will loop until the robot finds a space, or the user resets or powers off the robot.

The last behavior is the parking behavior, once the robot finds a space the robot must the turn into that spot and park. This is accomplished by determining which side the found spot is on and the turning in pace toward that spot. After the turn the robot moves forward for a predetermined time. Currently the robot does not care were the other cars are in the parking lot. So, it will just park in the first available spot.

Experimental Layout and Results

The layout was simple provide an environment that the robot can function in and see if the robot will perform the tasks assigned to it. The actual testing occurred in the special platform described above. A parking scenario was setup and the robot was let

loose. It failed about ninety percent of the time with small periods where the robot performed as expected. Most of the error seems to come from two places one was the variable lighting condition and the other was from heat build up. After the heat build up was discovered then robot was allowed to cool and functionality returned but at a reduced performance. In the end it did find a parking space and then park in that space, it also followed a color object though the last performance was much diminished compared to previous trials. The one thing that the robot does well is avoid objects. It is not afraid to get close to things it seems to feel that it can take on anything.

Conclusion

The robot will perform the tasks it was designed to do. The performance can be improved however. Object avoidance is just fine and needs nothing changed at this time. Color following could use a better camera to find the color. Finding a parking space only needs a few improvements basically have a better idea of where the spaces are, and get a better camera for detecting colors. The parking which it does just fine needs to be more complicated to have a more intelligent approach. Data from the IR sensors should be used in conjunction with data from the camera. Some other improvements that are needed are better controlled lighting conditions and more control over the cameras calibration features.

There are some improvements that I have been meaning to get to added to the robot. The first one is the blink, braking and headlight system, I have the capabilities on the PCB board. Also line following or light follow can easily be added to the robot, the hardware is already in place for that to work. The last would probably get a different camera system..

The grand conclusion is that this is the single largest learning experience I have had at his University. This is an experience that I not only take professional knowledge away but also personal knowledge as well.

Documentation

LetAtWorkII with Atmel Mega128

Akida - http://www.opticompo.com/emb/atwork_en.html

CMUcam

Seattle Robotics - <http://www.seattlerobotics.com/cmucam.htm>

Sharp GP2D120

Mark III - <http://www.junun.org/MarkIII/Info.jsp?item=37>

Sharp GP2D12

Mark III - <http://www.junun.org/MarkIII/Info.jsp?item=9>

GWS Standard Servo

Mark III - <http://www.junun.org/MarkIII/Info.jsp?item=18>

16x2 LCD

Mark III - <http://www.junun.org/MarkIII/Info.jsp?item=35>

Appendices

Appendix A

The main program for object avoidance and color following.

```
/*
*****
*****
** Valet-bot (a.k.a. SpaceSaver-bot) Program **
**                                     By David Wynacht                               **
**          IMDL EEL 5666C, Summer 2003,          **
**          University of Florida                    **
*****
*****
```

This program was produced by David Wynacht
© Copyright 2003

Project : Valet-Bot
Version : 1
Date : 8/7/2003
Author : David Wynacht
Company : University of Florida,
 Electrical & Computer Engineering
 IMDL EEL 5666C - Summer 2003

Comments:

```
Chip type      : ATmega128
Program type   : Application
Clock frequency : 16.000000 MHz
Memory model   : Small
External SRAM size : 0
Data Stack size : 1024
*****8*/
```

```
#include <inttypes.h>
#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#include <stdio.h>
#include "servo.h"
#include "servo.c"
#include "lcd.h"
#include "lcd.c"
```

```
volatile uint16_t IR_center;
```

```

volatile uint16_t IR_left;
volatile uint16_t IR_right;
volatile int go=0;
volatile char camdata;
volatile char camstr[30];
volatile int pack=0;
volatile int indx=29;
volatile unsigned char
data0[3],data1[3],data2[3],data3[3],data4[3],data5[3],data6[3],data7[3];
volatile int mx,my,x1,y1,x2,y2,pix,conf;

```

```

SIGNAL(SIG_UART1_RECV)
{
    camdata=UDR1;
    if(pack==0)
    {
        if(camdata=='M')
        {
            pack=1;
            indx=29;
            int j;
            for(j=0;j<=30;j++){
                camstr[j]=0;}
            camstr[indx--]=camdata;
        }
    }
    else
    {
        if(camdata=='r')
        {
            pack=0;
        }
        else
        {
            camstr[indx--]=camdata;
        }
    }
    sei();
}

```

```

SIGNAL(SIG_INTERRUPT0)
{
    cli();
    servo_delay(1000);
    if(go==1){

```

```

    done=1;
    while(done==1){
    servo_set(0,-25);
    }
    servo_delay(1000);}
    else{
    go=1;
    }
    sei();
}

```

```

SIGNAL(SIG_INTERRUPT1)
{
    cli();
    servo_delay(100);
    if(go==1){
    done=1;
    while(done==1){
    servo_set(25,-25);
    }
    servo_delay(1000);}
    else{
    go=2;}
    sei();
}

```

```

SIGNAL(SIG_INTERRUPT4)
{
    servo_delay(100);
    if(go==1){
    done=1;
    while(done==1){
    servo_set(25,0);
    }
    servo_delay(1000);}
    sei();
}

```

```

SIGNAL(SIG_INTERRUPT5)
{
    servo_delay(100);
    if(go==1){
    done=1;
    while(done==1){
    servo_set(0,25);
    }
}

```

```

        servo_delay(1000);}
        sei();
    }

SIGNAL(SIG_INTERRUPT6)
{
    servo_delay(100);
    if(go==1){
        done=1;
        while(done==1){
            UDR0=0x32;
            servo_set(-25,25);
        }
        servo_delay(1000);}
    sei();
}

SIGNAL(SIG_INTERRUPT7)
{
    servo_delay(100);
    if(go==1){
        done=1;
        while(done==1){
            servo_set(-25,0);
        }
        servo_delay(1000);}
    sei();
}

void cmucomm(char *s)
{
    while (*s){
        UDR1=*s++;
        servo_delay(400);}
}

void init_cmucam(void)
{
    UBRR1L=0x08;
    UCSR1B=0x98;
    UCSR1C=0x06;
    //cmucomm("\rTC 80 170 90 170 90 170\r");
    cmucomm("\rTW\r");
}

void int_sensors(void)
{

```

```

    //See pages 242-245
    ACSR=0x80;
    ADMUX=0x43; //Selects voltage range and channel
    ADCSRA=0x80; //Enables A/D, Starts Conversion, Selects Free Running
Conversion
}

void get_sensors(void)
{
    ADMUX=0x42;
    ADCSRA=0xE0;
    servo_delay(2);
    IR_center= ADC;
    ADMUX=0x40;//41
    ADCSRA=0xE0;
    servo_delay(2);
    IR_left= ADC;
    ADMUX=0x41;//43
    ADCSRA=0xE0;
    servo_delay(2);
    IR_right= ADC;
}

int main(void)
{
    cli();
    int_sensors();
    DDRD=(DDRD && 0xfc);
    DDRE=(DDRE && 0x0f);
    servo_delay(100);
    EIMSK=0xF3;
    EICRA=0x00;
    EICRB=0x00;
    lcd_set_ddr();
    lcd_init();
    sei();
    lcd_send_str("Waiting for");
    lcd_send_command(0xc0);
    lcd_send_str("input.");
    while(go==0)
        {};
    if(go==1)
    {
        lcd_send_command(0x00);
        lcd_send_command(0x01);
        lcd_send_str("Object Avoid.");
    }
}

```

```

        lcd_send_command(0xc0);
        lcd_send_str("Init . . ");
        servo_init();
        lcd_send_str("Ready.");
    }
    while(go==1)
    {

        get_sensors();
        if(IR_center > 150 )
        {
            servo_set(-25,25);
        }
        else
        {
            if (IR_left > 70 && IR_right >70)
            {
                servo_set(-25,25);
            }
            else
            {
                if (IR_left >70)
                {
                    servo_set(25,25);
                }
                else
                {
                    if (IR_right >70)
                    {
                        servo_set(-25,-25);
                    }
                    else
                    {
                        servo_set(25,-25);
                    }
                }
            }
        }
    }
};
if(go==2)
{
    lcd_send_command(0x00);
    lcd_send_command(0x01);
    lcd_send_str("Color Follow");
    lcd_send_command(0xc0);
    lcd_send_str("Init . . ");
}

```

```

    init_cmucam();
    servo_init();
    lcd_send_str("Ready.");
}
while(go==2)
{
    int l=27;
    int x=0;
    while(camstr[l]!=' ' && l>=0)
    {
        data0[x++]=camstr[l-];
    }
    if (x==1) {mx= (data0[0]-48);}
    if (x==2) {mx= (((data0[0]-48)*10)+(data0[1]-48));}
    if (x==3) {mx= (((data0[0]-48)*100)+((data0[1]-48)*10)+(data0[2]-48));}
    x=0;
    l=l-1;
    while(camstr[l]!=' ' && l>=0)
    {
        data1[x++]=camstr[l-];
    }
    //if (x==1) {my= (data1[0]-48);}
    //if (x==2) {my= (((data1[0]-48)*10)+(data1[1]-48));}
    //if (x==3) {my= (((data1[0]-48)*100)+((data1[1]-48)*10)+(data1[2]-
48));}

    x=0;
    l=l-1;
    while(camstr[l]!=' ' && l>=0)
    { data2[x++]=camstr[l-];}
    //if (x==1) {x1= (data2[0]-48);}
    //if (x==2) {x1= (((data2[0]-48)*10)+(data2[1]-48));}
    //if (x==3) {x1= (((data2[0]-48)*100)+((data2[1]-48)*10)+(data2[2]-
48));}

    x=0;
    l=l-1;
    while(camstr[l]!=' ' && l>=0)
    { data3[x++]=camstr[l-];}
    //if (x==1) {y1= (data3[0]-48);}
    //if (x==2) {y1= (((data3[0]-48)*10)+(data3[1]-48));}
    //if (x==3) {y1= (((data3[0]-48)*100)+((data3[1]-48)*10)+(data3[2]-
48));}

    x=0;
    l=l-1;
    while(camstr[l]!=' ' && l>=0)
    { data4[x++]=camstr[l-];}
    //if (x==1) {x2= (data4[0]-48);}

```

```

//if (x==2) {x2= (((data4[0]-48)*10)+(data4[1]-48));}
//if (x==3) {x2= (((data4[0]-48)*100)+((data4[1]-48)*10)+(data4[2]-
48));}

x=0;
l=l-1;
while(camstr[l]!=' '&& l>=0)
{data5[x++]=camstr[l-];}
//if (x==1) {y2= (data5[0]-48);}
//if (x==2) {y2= (((data5[0]-48)*10)+(data5[1]-48));}
//if (x==3) {y2= (((data5[0]-48)*100)+((data5[1]-48)*10)+(data5[2]-
48));}

x=0;
l=l-1;
while(camstr[l]!=' '&& l>=0)
{data6[x++]=camstr[l-];}
if (x==1) {pix= (data6[0]-48);}
if (x==2) {pix= (((data6[0]-48)*10)+(data6[1]-48));}
if (x==3) {pix= (((data6[0]-48)*100)+((data6[1]-48)*10)+(data6[2]-48));}

//x=0;
//l=l-1;
//while(camstr[l]!=' '&& l>=0)
//{data7[x++]=camstr[l-];}
//if (x==1) {conf= (data7[0]-48);}
//if (x==2) {conf= (((data7[0]-48)*10)+(data7[1]-48));}
//if (x==3) {conf= (((data7[0]-48)*100)+((data7[1]-48)*10)+(data7[2]-
48));}

if(pix>0)
{
    if(mx<=50 && mx>=30){servo_set(25,-25); /*UDR0='c';*/}
    else
    {
        if(mx>50){servo_set(25,0); /*UDR0='r';*/}
        if(mx<30){servo_set(0,-25); /*UDR0='l';*/}
    }
}
else
{
    servo_set(0,0);
}
}
return 1;
}

```

Appendix B

The code for the space finding and parking.

/******

This program was produced by David Wynacht

© Copyright 2003

Project : Parking Code

Version : 1

Date : 5/31/2003

Author : David Wynacht

Company : IMDL

Comments: This is the program that the robot uses to park.

Chip type : ATmega128

Program type : Application

Clock frequency : 16.000000 MHz

Memory model : Small

External SRAM size : 0

Data Stack size : 1024

*****/

```
#include <inttypes.h>
```

```
#include <avr/io.h>
```

```
#include <avr/signal.h>
```

```
#include <avr/interrupt.h>
```

```
#include <stdio.h>
```

```
#include "lcd.h" //Max Billingsly with some change by me.
```

```
#include "lcd.c"
```

```
#include "servo.h"
```

```
#include "servo.c"
```

```
volatile char camdata;
```

```
volatile char camstr[30];
```

```
volatile int pack=0;
```

```
volatile int indx=29;
```

```
volatile unsigned char
```

```
data0[3],data1[3],data2[3],data3[3],data4[3],data5[3],data6[3],data7[3];
```

```
volatile int mx,my,x1,y1,x2,y2,pix,conf;
```

```
SIGNAL(SIG_UART1_RECV)
```

```
{
```

```
    camdata=UDR1;
```

```
    if(pack==0)
```

```
    {
```

```
        if(camdata=='M')
```

```
        {
```

```
            pack=1;
```

```

        indx=29;
        int j;
        for(j=0;j<=30;j++){
            camstr[j]=0;}
        camstr[indx--]=camdata;
    }
}
else
{
    if(camdata=='r')
    {
        pack=0;
    }
    else
    {
        camstr[indx--]=camdata;
    }
}

sei();
}
void cmucomm(char *s)
{
    while (*s){
        UDR1=*s++;
        servo_delay(400);}
}
void init_cmucam(void)
{
    UBRR1L=0x08;
    UCSR1B=0x98;
    UCSR1C=0x06;
    cmucomm("\rCR 19 32\r");//Turns off the auto gain.
    cmucomm("\rTW\r");
    //cmucomm("\rTC 80 170 90 170 90 170\r"); //Looks for either blue or white
    can't remember which.
}

void main(void){
    lcd_set_ddr();
    lcd_init();
    servo_setcam(80);
    servo_delay(2000);
    servo_setcam(0);
    servo_delay(2000);
    servo_setcam(-80);
}

```

```

    lcd_send_str("Init . . ");
    init_cmucam();
    servo_init();
    lcd_send_str("Ready.");
    int odd=0;
    sei();
while(1)
{
    int f=0;

    int l=27;
    int x=0;
    while(camstr[l]!=' ' && l>=0)
    { data0[x++]=camstr[l-];}
    if (x==1) {mx= (data0[0]-48);}
    if (x==2) {mx= (((data0[0]-48)*10)+(data0[1]-48));}
    if (x==3) {mx= (((data0[0]-48)*100)+((data0[1]-48)*10)+(data0[2]-48));}
    x=0;
    l=l-1;
    while(camstr[l]!=' '&& l>=0)
    { data1[x++]=camstr[l-];}
    //if (x==1) {my= (data1[0]-48);}
    //if (x==2) {my= (((data1[0]-48)*10)+(data1[1]-48));}
    //if (x==3) {my= (((data1[0]-48)*100)+((data1[1]-48)*10)+(data1[2]-48));}
    x=0;
    l=l-1;
    while(camstr[l]!=' '&& l>=0)
    { data2[x++]=camstr[l-];}
    //if (x==1) {x1= (data2[0]-48);}
    //if (x==2) {x1= (((data2[0]-48)*10)+(data2[1]-48));}
    //if (x==3) {x1= (((data2[0]-48)*100)+((data2[1]-48)*10)+(data2[2]-48));}
    x=0;
    l=l-1;
    while(camstr[l]!=' '&& l>=0)
    { data3[x++]=camstr[l-];}
    //if (x==1) {y1= (data3[0]-48);}
    //if (x==2) {y1= (((data3[0]-48)*10)+(data3[1]-48));}
    //if (x==3) {y1= (((data3[0]-48)*100)+((data3[1]-48)*10)+(data3[2]-48));}
    x=0;
    l=l-1;
    while(camstr[l]!=' '&& l>=0)
    { data4[x++]=camstr[l-];}
    //if (x==1) {x2= (data4[0]-48);}
    //if (x==2) {x2= (((data4[0]-48)*10)+(data4[1]-48));}
    //if (x==3) {x2= (((data4[0]-48)*100)+((data4[1]-48)*10)+(data4[2]-48));}
    x=0;

```

```

l=l-1;
while(camstr[l]!=' '&& l>=0)
{data5[x++]=camstr[l-];}
//if (x==1) {y2= (data5[0]-48);}
//if (x==2) {y2= (((data5[0]-48)*10)+(data5[1]-48));}
//if (x==3) {y2= (((data5[0]-48)*100)+((data5[1]-48)*10)+(data5[2]-48));}
x=0;
l=l-1;
while(camstr[l]!=' '&& l>=0)
{data6[x++]=camstr[l-];}
if (x==1) {pix= (data6[0]-48);}
if (x==2) {pix= (((data6[0]-48)*10)+(data6[1]-48));}
if (x==3) {pix= (((data6[0]-48)*100)+((data6[1]-48)*10)+(data6[2]-48));}

//x=0;
//l=l-1;
//while(camstr[l]!=' '&& l>=0)
//{data7[x++]=camstr[l-];}
//if (x==1) {conf= (data7[0]-48);}
//if (x==2) {conf= (((data7[0]-48)*10)+(data7[1]-48));}
//if (x==3) {conf= (((data7[0]-48)*100)+((data7[1]-48)*10)+(data7[2]-48));}

lcd_send_command(0x00);
lcd_send_command(0x01);
lcd_send_str("PIX: ");
lcd_send_str(data6);

//servo_delay(100);

//if(pix>0){
//if(mx<=50 && mx>=30){servo_set(25,-25); /*UDR0='c';*/}
//else{
//if(mx>50){servo_set(25,0); /*UDR0='r';*/}
//if(mx<30){servo_set(0,-25); /*UDR0='l';*/}
//}}
//else{
//servo_set(0,0);
//}
int k;
if(pix>0 && odd==1)
{
for(k=0;k<5000;k++){
servo_set(10,10);}
for(k=0;k<9000;k++){
servo_set(10,-10);}
servo_setcam(0);

```

```

while(1){servo_set(0,0);}
}else{
if(pix>0 && odd==0){
for(k=0;k<5000;k++){
servo_set(-10,-10);}
for(k=0;k<9000;k++){
servo_set(10,-10);}
servo_setcam(0);
while(1){servo_set(0,0);}
}else{
if(odd==1){
servo_setcam(80);
servo_delay(3000);
int g;
for(g=0;g<5000;g++){
servo_set(10,-10);}
for(f=0;f<5000;f++){
servo_set(0,0);}
odd=0;}
else{
servo_setcam(-80);
servo_delay(3000);
odd=1;}}
}
return 1;
}

```

Appendix C

The header for the servos.

```
/******
```

```
This program was produced by  
David Wynacht  
© Copyright 2003
```

```
Project : Servo  
Version : 1  
Date   : 5/31/2003  
Author : David Wynacht  
Company : IMDL  
Comments:
```

```
Chip type      : ATmega128  
Program type   : Application  
Clock frequency : 16.000000 MHz  
Memory model   : Small  
External SRAM size : 0  
Data Stack size : 1024  
*****/
```

```
//#ifndef servo_H  
//#define servo_H  
#include <avr/io.h>  
#include <inttypes.h>  
void servo_delay(int len);  
void servo_delayms(int len);  
void servo_init(void);  
void servo_set(signed int left, signed int right);  
void serv0_setcam(signed int center);  
  
//#endif
```

The code for the servos.

```
/******
```

This program was produced by David Wynacht

© Copyright 2003

Project : Servo

Version : 1

Date : 5/31/2003

Author : David Wynacht

Company : IMDL

Comments:

Chip type : ATmega128

Program type : Application

Clock frequency : 16.000000 MHz

Memory model : Small

External SRAM size : 0

Data Stack size : 1024

```
*****/
```

```
#include <inttypes.h>
```

```
#include <avr/io.h>
```

```
#include <stdio.h>
```

```
volatile int done=0;
```

```
void servo_delayms(int len){
```

```
    while(len)
```

```
        { //increments of 10 micro seconds
```

```
            len--;
```

```
            int k;
```

```
            for(k=0;k<16;k++)//1597
```

```
                asm("nop");
```

```
            }
```

```
}
```

```
void servo_delay(int len) {
```

```
    while(len)
```

```
        { //increments of 10 micro seconds
```

```
            len--;
```

```
            int k;
```

```
            for(k=0;k<1567;k++)//1597
```

```
                asm("nop");
```

```
            }
```

```
}
```

```
void servo_init(void)
```

```

{
    // Timer/Counter 1 initialization
    // Clock source: System Clock
    // Clock value: Timer 1 Scaled by 1/8
    // Mode: PhPWM top=20000
    // OC1A output: Con.
    //Timer counter control Reg
    TCCR1A=0xAA; //(1010 0010)
    TCCR1B=0x12; //(0001 0010)

    //Output compare reg 1a A14
    //high byte    set duty cycle
    //low byte      220=5%ccw 9D0=10%cw    5DC=7.5%stop
    //input capture reg 1
    ICR1H=0x4E; //sets period using the top equation
    ICR1L=0x20;
    OCR1A=0x05d2;
    OCR1B=0X05dc;
    OCR1C=0x05dc;
    DDRB=0xe0;
}
void servo_set(signed int left, signed int right)
{
    //int j=100;
    uint16_t new_left;
    uint16_t new_right;
    new_left=1490 + ( 10*left);
    new_right=1500 + ( 10*right);
    if(((OCR1A < .9*new_left)||((OCR1A > 1.1*new_left))||((OCR1B <
.9*new_right)||((OCR1B > 1.1*new_right)) )
    {
        OCR1A=(9.9*OCR1A + .1*new_left)/10;
        OCR1B=(9.9*OCR1B + .1*new_right)/10;
    }
    else
    {
        OCR1A=new_left;
        OCR1B=new_right;
        done=0;
    }
}
void servo_setcam(signed int center)
{
    OCR1C=1500 + (10*center);
}

```

Appendix D

The header for the LCD.

```
/*
 * lcd.h
 *
 * Author: Max Billingsley
 * Adapted by: David Wynthacht
 */

#define LCD_PORT PORTA
#define LCD_DDR  DDRA

#define ENABLE 0x40

/* function prototypes */

void lcd_set_ddr(void);
void lcd_init(void);
void lcd_delay(void);
void lcd_send_str(char *s);
void lcd_send_byte(uint8_t val);
void lcd_send_command(uint8_t val);
```

The code for the LCD.

```
/*
 * lcd.c
 *
 * Author: Max Billingsley
 * Adapted by: David Wynthacht
 */

/*
 * LCD_PORT0 - DB4
 * LCD_PORT1 - DB5
 * LCD_PORT2 - DB6
 * LCD_PORT3 - DB7
 * LCD_PORT4 - RS
 * LCD_PORT5 - r/w
 * LCD_PORT6 - EN
 *
 * RS: Register Select:
 *
 * 0 - Command Register
 * 1 - Data Register
 *
 */

#include <inttypes.h>
#include <avr/io.h>

#include "lcd.h"

/* entry point */

void lcd_init(void)
{
    lcd_send_command(0x43);
    lcd_send_command(0x43);
    lcd_send_command(0x43);
    lcd_send_command(0x42);
    lcd_send_command(0x42);
    lcd_send_command(0x48);
    lcd_send_command(0x40);
    lcd_send_command(0x0f);
    lcd_send_command(0x00);
}
```

```

        lcd_send_command(0x01);
    }

void lcd_set_ddr(void)
{
    LCD_DDR = 0xff;
}

void lcd_delay(void)
{
    uint16_t time1;

    for (time1 = 0; time1 < 2000; time1++);
}

void lcd_send_str(char *s)
{
    while (*s) lcd_send_byte(*s++);
}

void lcd_send_byte(uint8_t val)
{
    uint8_t temp = val;

    val >>= 4;
    val |= 0x10; /* set data mode */
    LCD_PORT = val;

    lcd_delay();

    LCD_PORT |= ENABLE;
    LCD_PORT &= ~ENABLE;

    temp &= 0x0f;
    temp |= 0x10; /* set data mode */
    LCD_PORT = temp;

    lcd_delay();

    LCD_PORT |= ENABLE;
    LCD_PORT &= ~ENABLE;

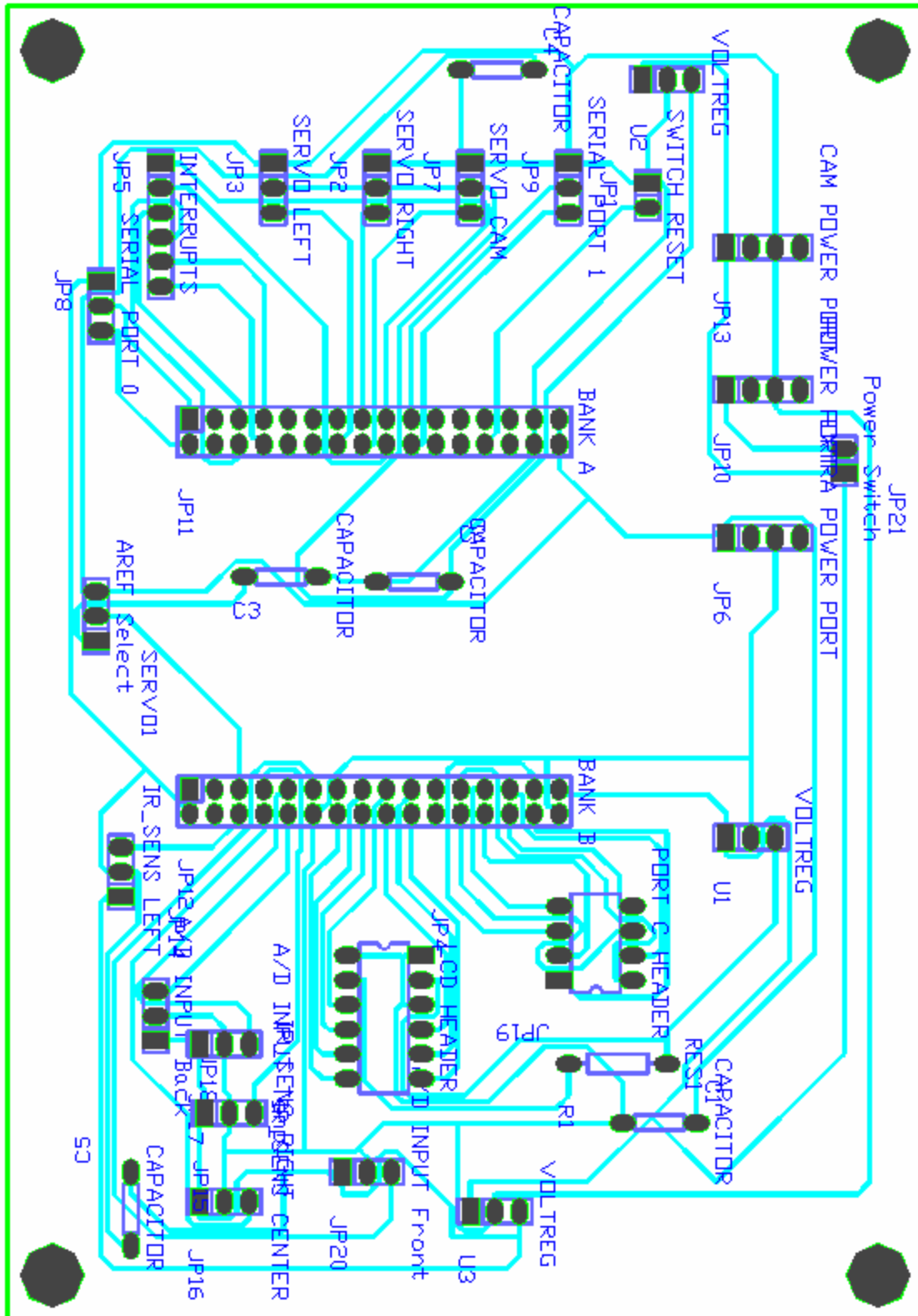
    lcd_delay();
}

void lcd_send_command(uint8_t val)

```

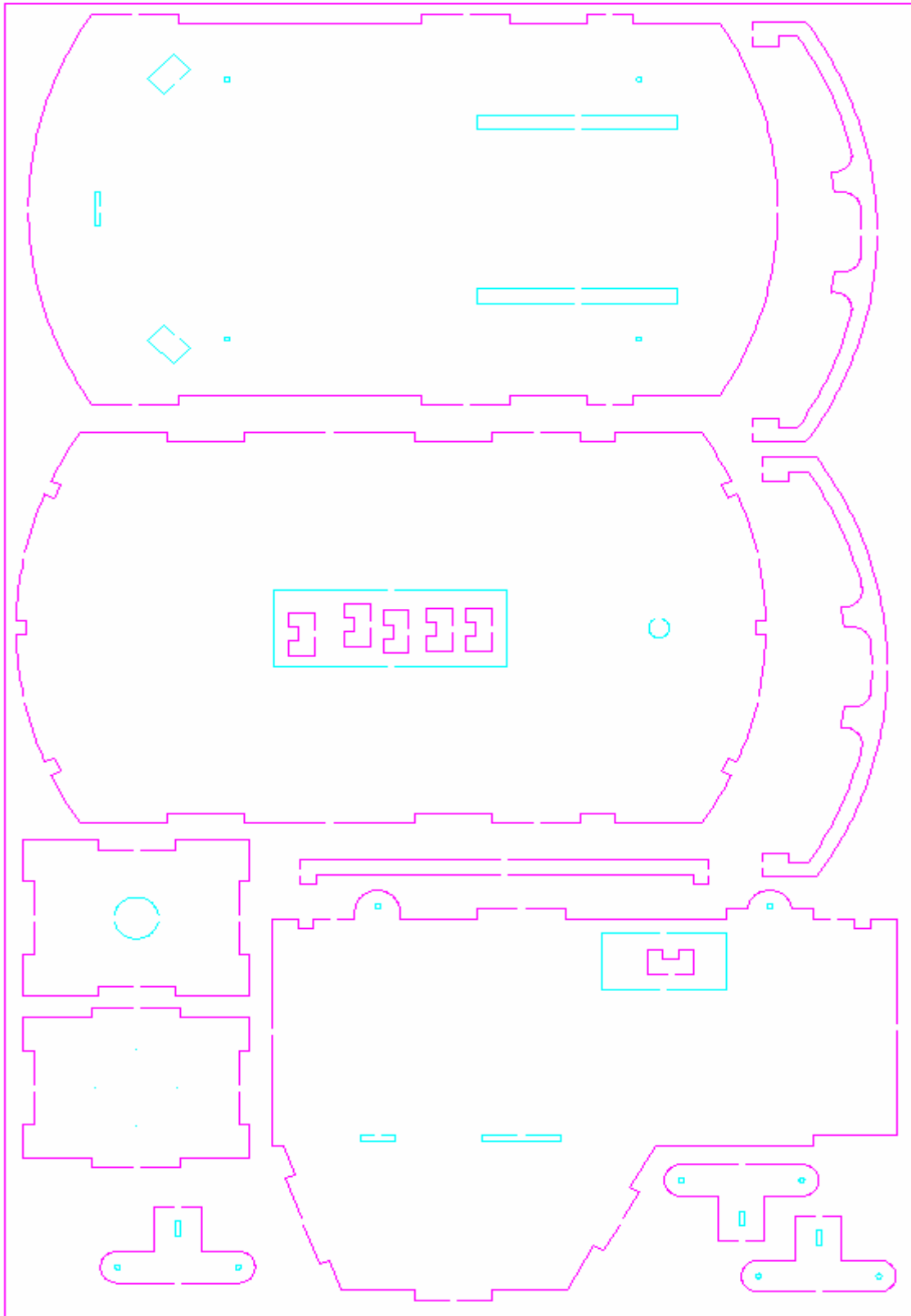
```
{  
    uint8_t temp = val;  
  
    val >>= 4;  
    LCD_PORT = val;  
  
    lcd_delay();  
  
    LCD_PORT |= ENABLE;  
    LCD_PORT &= ~ENABLE;  
  
    temp &= 0x0f;  
    LCD_PORT = temp;  
  
    lcd_delay();  
  
    LCD_PORT |= ENABLE;  
    LCD_PORT &= ~ENABLE;  
  
    lcd_delay();  
}
```


The PCB layout.

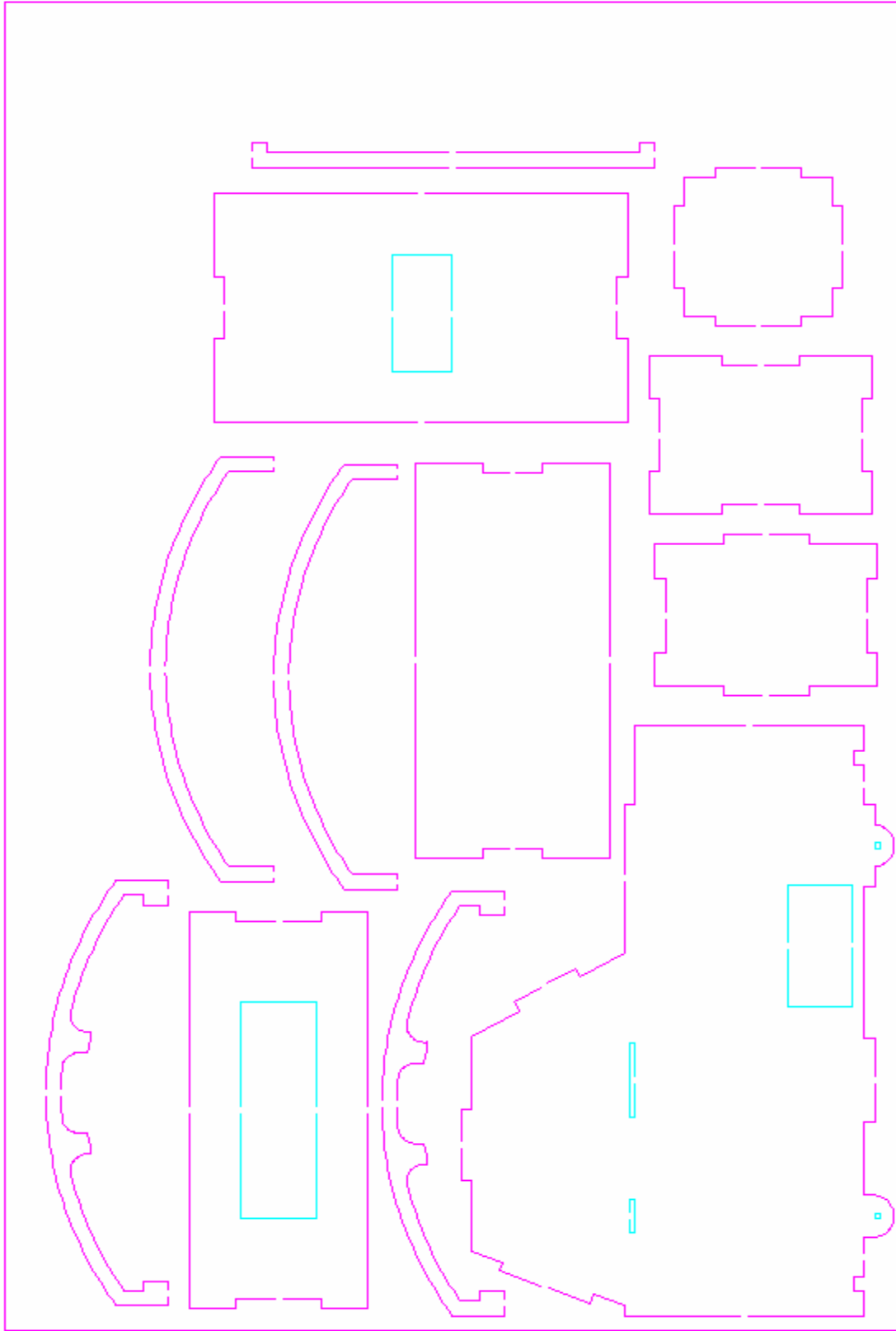


Appendix F

The AutoCAD board 1:



The AutoCAD board 2:



Appendix G

The Parts list:

- 1 LetAtWork II Module – Contains a Atmel Mega128 running at 16MHz Available from www.akida.com for \$69.
- 1 PCB board – Custom made and cut out at IMDL.
- 2 11x17 sheets of balsa wood with AutoCAD designed cut out at IMDL
- 7 Cans of spray paint, 1 red, 1 silver, 1 blue, 1 clear coat, 1 black, and 2 white from Wal-Mart for \$2.95 each.
- 1 Sheet of pressed board size of plywood sheet from Lowe’s for \$ 5.99.
- 1 CMUcam from Seattle Robotics for \$109 assembled and tested.
- 3 GWS standard servos from Mark III for \$10.50 each.
- 2 4.6” wheels from All Electronics for \$1.50 each
- 2 16x2 LCD with LED backlight using Hitachi HD44780U LCD Controller one was damaged and is now the “ground effect” light. The other is the user display. From Mark III for \$8.50 each
- 2 Sharp GP2D120 IR rangers for close range, from Mark III for \$8.25.
- 1 Sharp GP2D12 IR Ranger for long range, from Mark III for \$8.25.

Many Female headers provided by IMDL

Many Male headers provided by IMDL

- 1 SPST switch provided by IMDL - power disconnect
- 1 Push button switch provided IMDL – Reset switch
- 1 2’ long Ribbon Cable section provided by IMDL

A lot of Hot Glue mainly black but some clear provided by IMDL

- 5 .01 μ F Capacitors provided by IMDL
- 1 150 resistor provided by IMDL
- 1 10kohm sip resistor provided by IMDL

6 "bump switches" provided by IMDL

Some wood glue provided by Shuguang Feng

Some Super glue provided by Seth Lakritz

Many 4/40 screws and nuts provided by IMDL

1 set of 5/16 bot, 2 nuts, and one end cap from Lowe's under \$2.

3 Voltage regulators 5V 1A output.

A lot of Solder provided by IMDL

A lot of help, support, and laughs from friends, family, REUers and MIL members.

A lot of patience, loss of sleep, frustration, blood, sweat, and tears, appreciation.