DR AUTOBUS

Written by:
Penelope Herrera

EEL 5666C
Intelligence Machine Design Laboratory
University of Florida
Electrical and Computer Engineering
Fall 2003

Table of Contents

Abstract

The purpose of this paper is to give a documented report on the design, development and realization of DR Autobus. DR Autobus is a project design to look at the future of the transportation Industry and the public service. It is a robot prototype whose main purpose is to perform basic functions of a public bus but without the need of a driver, thus, being completed autonomous.
As a public bus prototype AUTOBUS simulated the basic functions of the pick-up, drop-off, message display and announcement with its behaviors of line following for navigation through streets; obstacle avoidance through its IR and bump switches sensory system; message display through an on LDC and passenger interaction through a voice record/play IC.
The realization of this motor vehicle was not trivial since many constraints such as steering need consideration. In Autobus steering was accomplished by Ackerman steering and this paper will discuss all the tasks accomplished in this project so far; including constraints, advantages, changes of ideas and future work that needs to be finish.

Executive Summary

The purpose of DR Autobus is to perform some functions of public transportation bus. It should pick-up passenger and drop-off passengers at four stops place on white ground with a long piece of black electrical tape as a road, the robot should follow this line to find the bus shelters. In order to perform these functions, a physical platform was built, sensors and programming were implemented. Though, during the course of the class some features have not yet worked properly.

The platform was design to look like a bus, with two sub-sections. The first level holds the battery pack, the LCD display and on its floor the CDS cell sits.

The second the level holds of the bus contains the circuit test board, stands off, wires and all electronics. The circuit test board had to be used during the testing and demonstration because the Protel design did not get routed correctly, it has two interconnections; thus it could not be gut on time for Demo Day. In addition, the third part of the platform was the R/C base with the front dc motor for right and left steering, and the two servo motors for forward and back driving. The small DC motor in the front was controlled with an H-bridge from National Semiconductors. This chip required a PWM to work.

One of the behaviors that is implemented is object avoidance. Object avoidance is accomplished with two GD2P120 IR sensors place on the front of the robot. The other behavior that is need in DR Autobus was line following. The hardware was accomplished with a line tracking circuit, but for Demo day the program to run this device still had bugs that needed to be corrected.

A lot code was written just to be able to interact with the LCD display and to obtained data from the IR sensors. The most successful behavior program was object avoidance.

Due to time constraints the other features of the Robot were not completed but can be easily incorporated to it.

Introduction

Transportation is a basic need in our daily lives. Many of us drive our cars or take the bus to school, work or home. Now image one day getting on a bus and not finding a driver, and when it is time to get off you simply press a button and then the bus stops. One may wonder, "How does this work," a bus that can drive autonomously? In the 20th Century technology took giant steps towards the world of robotics and automation, and car makers are currently working on models that drive themselves autonomously.

I took on the challenge on the designing a robot which I called DR Autobus. DR Autobus is a four passenger bus that performs basic functions of a public service bus; the pick-up and drop-off of passengers in bus stop shelters and LCD message display.

One of the important features that Autobus needs to perform to realize its functions is navigation. Thus, utilizing the behavior of line following Autobus will drive up the four assigned bus stops that will be placed several feet apart from each other. In addition, Autobus needs to be able to avoid pedestrians walking on the street, thus, object avoidance is another important behavior needed in the robot.

Indeed, this paper will talk about the whole project development cycle of making this bus prototype. When building a robot a designer runs through the stages of integrating the system, building the platform, programming and experimenting. Dr Autobus stages are presented along with all the information about its actuation, sensors, behavior information, experimental results and conclusions.

In the future a full and realistic consideration of the realization of a prototype function will be made. All of the actual function will be finish along with other functions such as driving with other vehicles, stop lights and paying method; can be explore for a possible realization. Also, the voice chip part of the special sensor suite of the robot needs to be incorporated.

Integrated System

In order to achieve the goals introduced previously several components need to be integrated into a whole functional system. The system individual components can be broken into the following:

- Microcontroller: Atmel 128
- Actuation Components: DC motor and Servos
- Sensors: IR Sensors, CDS cells, Bump Switches, Line Tracking Circuit.
- Interactive Display: LCD

All physical components need to be integrated to the Atmel 128 microcontroller. This microcontroller was found in the AVR freaks website, within the LetAtwork II development board. In DR Autobus all components are asserted by signals from the 128, thus, I put careful thought on its features, so that every component I need to attach to it can be done. The features used in the Atmel 128 for Autobus are among the PWM system, AD system, Interrupt system and General Purpose Ports.

PWM System

The Atmel 128 has three (16-bit) Timers/Counters and six PWM channels. This is very good because during the testing of the actuation components I burned one of the PWM channels. Thus, is very handy to have available more than two PWM channels.

A/D System

The 128 has eight-channel, ten-bit ADC with eight single –ended channels. Two of the sensory components, the CDS cells and IR sensors were attached to this port. The CDS aided in the people counting operation of DR Autobus. In order to display "Full Bus," the bus system needed to find a way to count up to top passenger value; which in this case is four. Also, to avoid hitting objects the IR sensors were attached to this A/D port. Since a wide range of values was obtained from the IR rangers, it was vital to attach the sensors this port.

Interrupt System

In addition, for object avoidance a second sensor component is used. Two bump switches are placed to the front of the bus; and corresponding signals are attached to the two of the eight interrupt pins in the Atmel 128. Also, a signal from Port D in the processor is attached to a third interrupt pin for the people counting process needed to display "Full Bus"

General Purpose Ports

The other components of DR Autobus were simply attached to general purpose ports. The ports were configure as input or output pins since signals to components like the LCD and line tracking circuit just needed to be either asserted or be read.

Indeed, all processing was handle by the processor and wiring signals for testing purposes was trivial. But the programming so that all circuit components do work properly and in cohesion was the most challenging part of integration. Figure 1 below gives a graphical view on how this integrated system was accomplished:

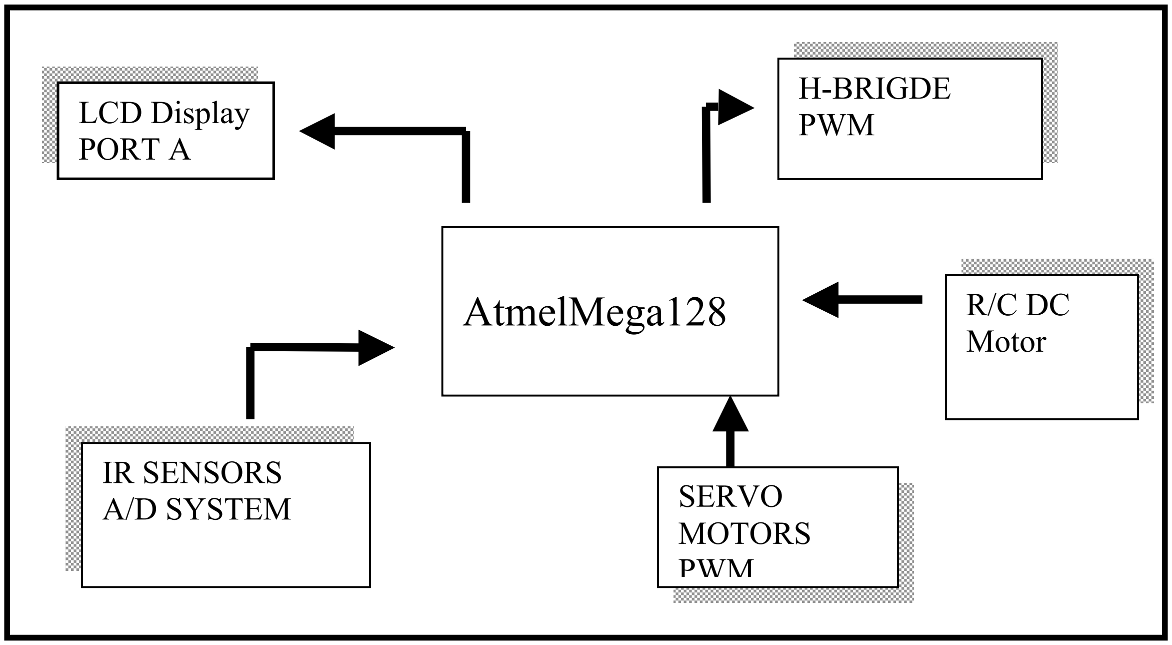


Figure 1

Mobile Platform

The mobile platform was designed in AutoCAD 2003 (the design can be seen in Appendix B) and it looks similar to an English Double Decker bus as seen in Figure 2. The material utilized was 1/8" balsa wood. Two sheets were provided by IMDL and an extra sheet was purchase in Florida Office Supply.

The overall physical part of the Platform can be divided into three parts:

- Upper Level: The top of the bus has an opening so that the top inside portion of the bus can be access. In the outside surrounding the windows of the bus are made with clear plastic adhesive paper. Inside there four seats (CDS cells) for the passengers.
- Lower Level: Inside it contains the circuit board and battery for the system power.
- Base: Tyco R/C ford mustang toy base cut in half. The front portion was used for the actuation of the bus, and four wood stands off were placed on top it to mount the wood red platform.

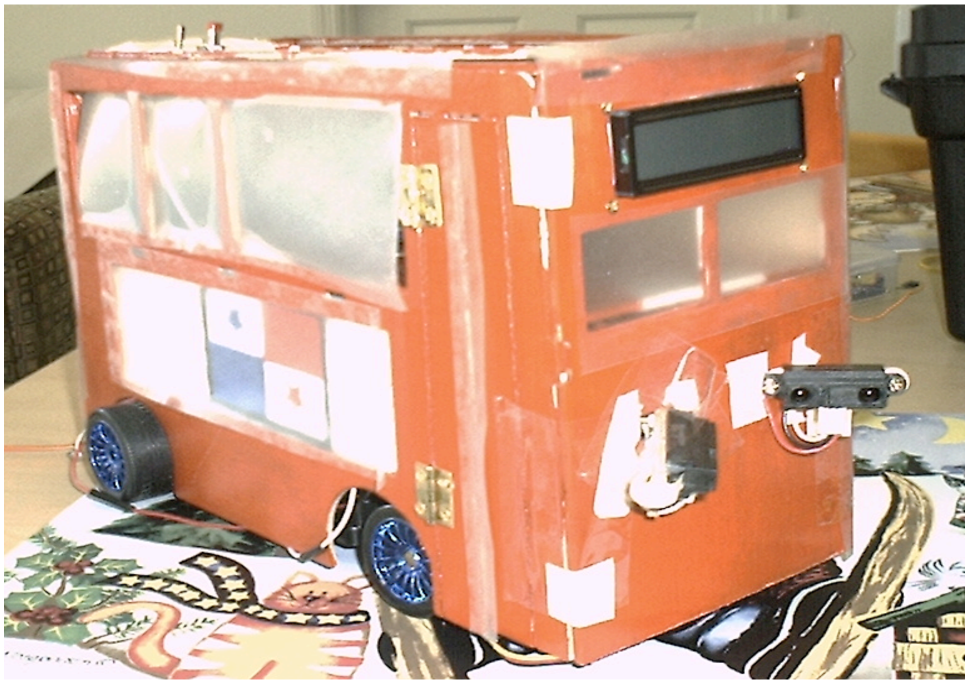


Figure 2

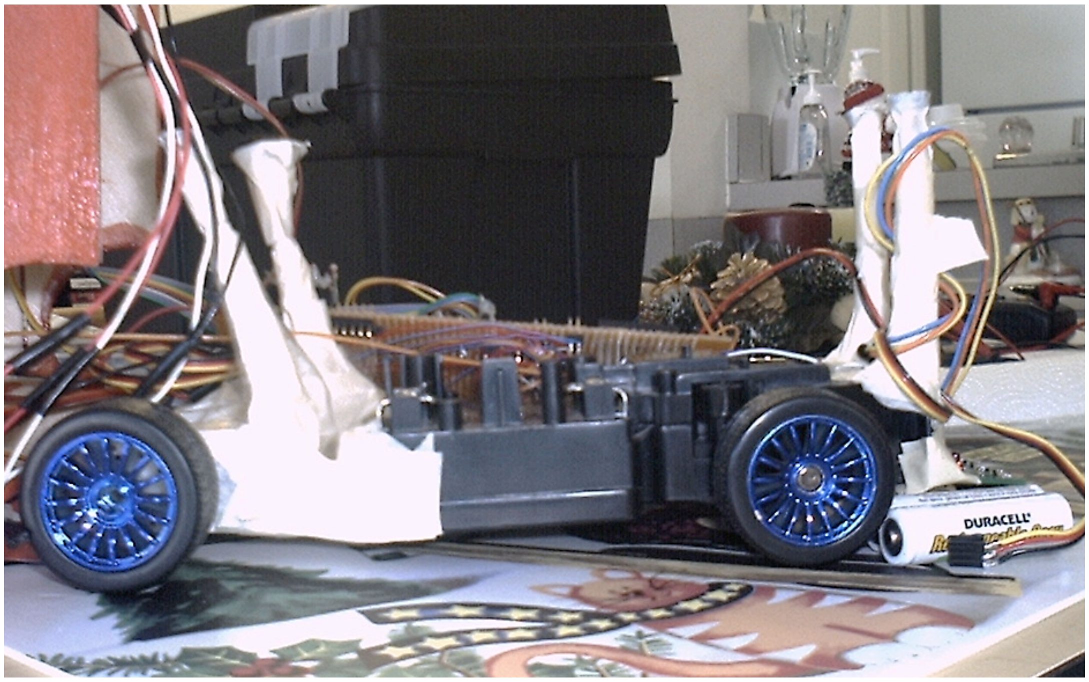
Actuation

DR Autobus has four wheels and they all have to be controlled. Thus, I purchased an Tyco R/C Ford Mustang car from ToysRus and removed the top portion so that the base (seen in figure 3) which has two steering boxes that can be used to steer the bus left, right, forward and back.

The circuit was cut out and back portion which contained a DC motor for the back wheels of the car was removed. Initially, this motor was intended to be used. But its specs were unknown to me. After a month of trying to figure this out, I decided not to use. Thus, the taped white section seen on Figure 3 was the removed back gear box that was removed; and then two pieces of wood were taped in there to replace that section along with two servo motors as the one seen in Figure 4.

These are Ball-bearing Metal Gear Servo Motor. This motor is significantly smaller, lighter, and higher-torque compared to a standard servo motor. They have two ball bearings and metal gears and were used to drive the robot back and forward. In addition, the servos were hacked for continuous rotation and PWM signal controlled its Speed.

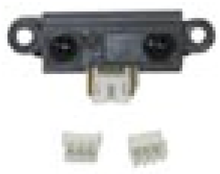
For the front Wheels the R/C gear box was used, and this gear had a DC motor with a stalling of about 1.5 A. The motor is small and it was controlled using a 3A H-bridge from National Semiconductors. The H-bridge required a minimum of 12V of power, thus, the bus had to hold two battery pack which were placed in the first level of the bus.



RCA hacked base with two Servos
Figure 3

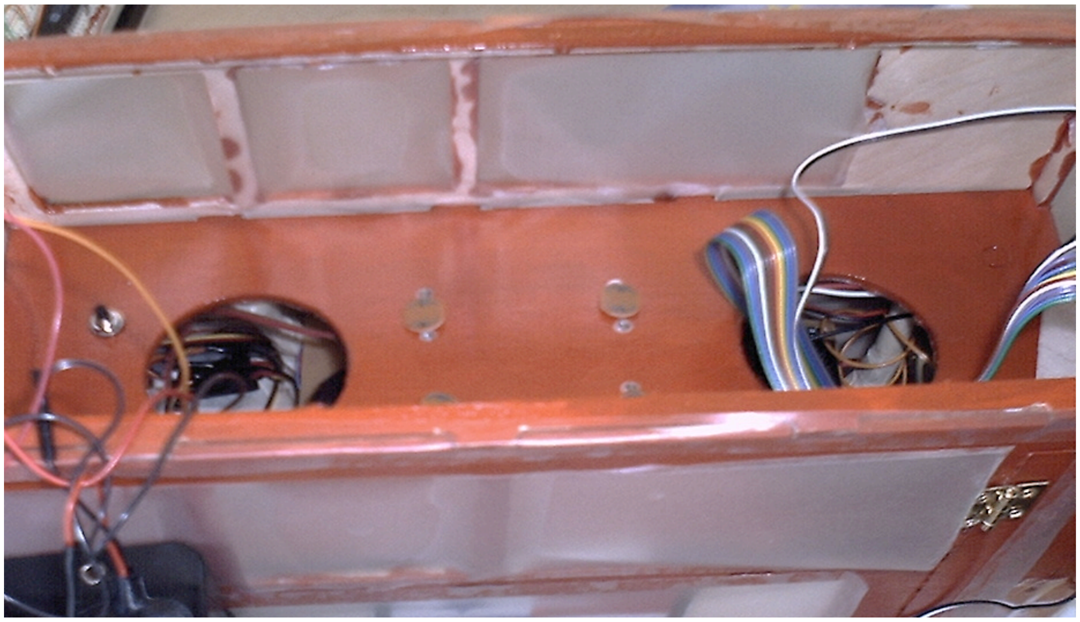
Sensors

- GP2D12 IR: They are short range infrared distance measuring sensor. Accurately determines range to target between 10cm and 80cm.
  Vendor: Mark III Robot Store
  Part Number SHARP GP2D12
  In DR Autobus they are placed in the front in a 45 degree line position with respect to the bus front.



SHARP GP2D12

- CDS Cells: Their resistance increases or decreases with exposure to or coverage from light.
  Vendor: Provided by IMDL Lab.



CDS Cell on DR AUTOBUS

- <u>Line Tracking Circuit:</u> used to track black line on a white background.
  Part Number: TRA-01
  Vendor: Lynxmotion

- Voice Playback/Record Chip: used to record and playback messages from the pet's owner. Easy to use by itself or with a controller.
  Part Number: 141655
  Vendor: Jameco

Jameco Part #141655

Behaviors

OBJECT AVOID

The first behavior object the robot uses to be aware of its environment is avoidance. Autobus uses the IR rangers to basically see in front of it as it drives. It will always try to go forward until either an IR sensor detects an object too close to the front of the robot. The sensors are placed in front as soon they reach a threshold distance the bus retrieves back for a few seconds, turns to a side and keeps on driving in that direction. The 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 not detected via the IR system the robot will run over the object, thus, bumps switches have to be put in place for future work.

LINE FOLLOWING

The second behavior is tracking and following a line. This behavior is done by using the line tracking circuit.

LCD DISPLAY

LCC display is not necessarily a behavior but is another way that the robot interacts with its environment. DR Autobus communicates every task it performs, from hardware initializations to direction is suppose to follow when it does the line following. Also, when the IR sensors are not able to see an object and it hits it displays "Dialing 911."

Experimental Layout and Results

The layout was very simple, a row of 8.5 X 11 white sheets tape one after another, a black electrical tape strip. During testing the line following program was downloaded to the bus and placed on this layout, and it seemed not to follow the line completely at first. Later on I discovered it did not follow the line at all. The error seemed to be incorrect reading of the input port to which the line tracking circuit was attached. Other components such as were tested on proto-boards and they worked pretty well. Object avoidance was the one feature that the robot does well. Although, is it obvious through testing that bump switches need to be added to the design.

Also, having and LCD as a debugging tool was a good feature of testing and experimenting. Since many times I could observed that the hard was failing because the LCD display the right software functions.

Conclusion

At the beginning of the semester I thought I had chosen a simple idea, therefore a simple robot. It turned out that the building this robot was not trivial at all. I was very challenging to learn concepts and put them into practice in the same semester and the biggest draw back was precisely the lack of experience when it comes to building a robot.

I didn't take into consideration a lot of details that are involved in this project such as different power consumption levels of each individual device, steering precision, handling new devices and time constraints.

When it comes to power consumption certain devices require a lot power. I used a 12V power supply because that the minimum that the H-bridge required. Thus, for other devices to work a voltage regulator was needed and I tried to supply 5V to the devices through only one voltage regulator. Actually, pretty much every device needed its own voltage regulator and as a result many H-bridges and two pins from my last processor blew up.

Another concept learned is using an RCA for steering purposes. The steering box in the R/C car I purchased turned left and right in a very sharp way. This is not good a lot for line following purposes.

Also, a Protel design in very early stages would have reduced the amount of damaged Atmel's. Again because of inexperience a PCB board was not completed routed and I was not able to use it. It would have saved me a lot of money and time to design and work in it in early stages, since at many times I switched power and ground on the processor and they resulted damaged. H-bridges were other sensitive devices. They needed no less than 12V, and with any little less they do not work; and the Atmel 128 cannot be program while they have power, otherwise they also get damaged.

In conclusion, if I had enough time I would have finish my robot. I spent too much time trying to make the R/C base completely front and back, and it wasn't until the last minute until I used another resource (servos). I also spent quite a bit of time just waiting for new boards after I had damaged one.

In the future, I will pursue to finish my robot with all the experience I have gain, I made a lot of mistakes in the process, and as an undergraduate student this is a big eye opener to see the skills I lack and need to gain as engineer. I think I accomplish my goal which was to be prepared for senior design. IMDL is the hardest class I have ever taken in my life, but it the best and most educational.

References

LetAtWorkII with Atmel Mega128

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

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

Line Tracking Circuit

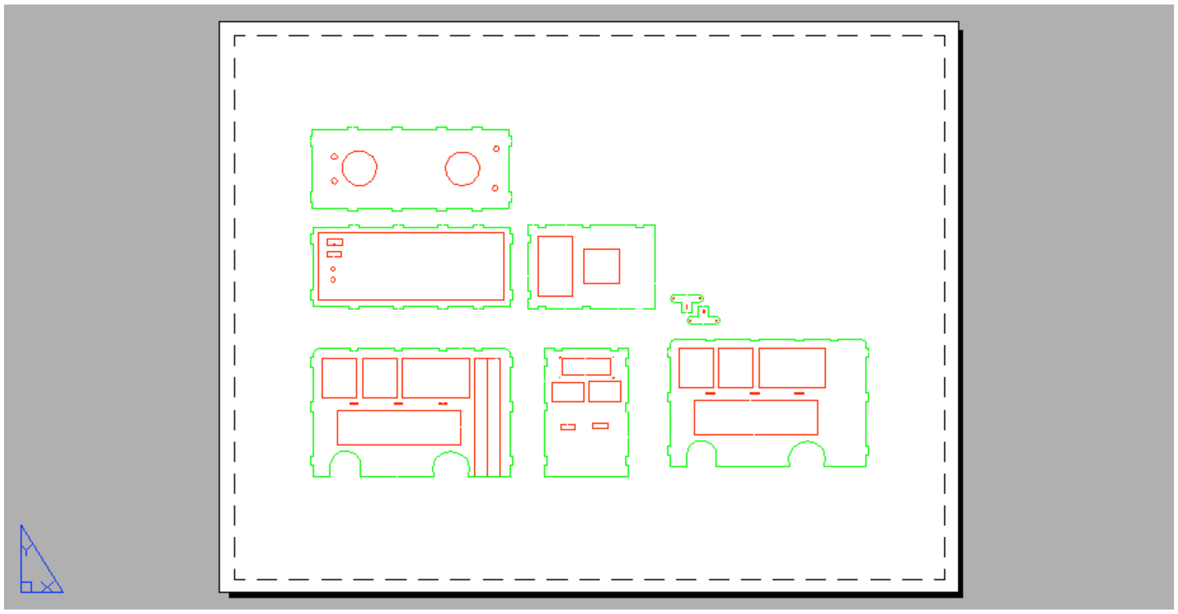
Lynxmotion -
http://www.lynxmotion.com/Product.aspx?productID=57&CategoryID=8

Appendix A

PARTS LIST

5	LetAtWork II Module – Contains a Atmel Mega128 running at 16MHz Available from www.akida.com
1	PCB board design.
3	11x17 sheets of balsa wood with AutoCAD designed cut out at IMDL
1	Can of red pain from Wal-Mart
2	GWS standard servos from Mark III
3	TYCO R/C car purchased at ToysRus.
2	16x2 LCD with LED
2	Sharp GP2D12 IR Ranger
	Female headers provided by IMDL
	Male headers provided by IMDL
1	SPST switch provided by IMDL - power disconnect
2	Push button from Radioshack
1	2' long Ribbon Cable section provided by IMDL
	Hot Glue
5	.01_F Capacitors provided by IMDL
1	15_ resistor provided by IMDL
1	10kohm sip resistor provided by IMDL
3	"bump switches" provided by IMDL
	Crazy glue and Wood glue
6	Voltage regulators 5V 1A output

Appendix B



APPENDIX C

```
/*
 * lcd.h
 *
 */
#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 );
 void lcd_send_command(uint8_t );
****************************************************************
****************************************************************
****
/*
 * lcd.c
 *
 * Author: Max Billingsley and David Wynacht
 * Adapted by: Penelope Herrera
  * 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();
 }
*****************************************************************
*****************************************************************
****
/*Program to read and write to the LCD
 #include <inttypes.h>
#include <avr/io.h>
#include "lcd.h"
#include "lcd.c"
void delay( int len){
         while(len)
                  {
                  len--;
                  int k;
                  for(k=0;k<1567;k++)
                  asm("nop");
                  }
}
/**************Main***********************************************
**/
         void main (void){
                  delay(10);
                  lcd_set_ddr();
                  lcd_init();
```

```
                delay(100);
                lcd_send_str("Init Done");
                delay(2000);
                lcd_send_command(0x00);
                lcd_send_command(0x01);
                delay(100);
                lcd_send_str("Ready to Use. ");
                lcd_send_command(0xc0);
                while(1){;}
        }
**************************************************************
**************************************************************
****//*Written by: Penelope Herrera.           Created on: 10/27/03
//Code to run motors forward and backward.
//A brake function is included.
//Code requires a PWM, this function drives forward and right
//**************Include statements**************************/
#include <inttypes.h>
#include <avr/io.h>
#include "lcd.h"
#include "lcd.c"
void front_motor_pwm_init(void){
TCCR1A=0xA2;            //Set COM1A1, COM1B1, WGM11
TCCR1B=0x12;            //Set CS11, WGM13
ICR1H=0x4e; //Top value for 750HZ Frequency
ICR1L=0x20;
DDRB=0xFF;
OCR1C=0x1388;
}
```

```
void drive_left(void){       // To drive forward PWM, DIR and Brake signals are
asserted
DDRC=0xff;
front_motor_pwm_init();                                    // Turn on PWM
signal=>PWM pin 5 in H-bridge
PORTC=0x80;                                           //   PORTC0=>DIR
pin(V=high) ; PORTC1=>BRAKE(V=L)
}
void drive_right(void){      // To drive forward PWM, DIR and Brake signals are
asserted
DDRC=0xff;
front_motor_pwm_init();                                    // Turn on PWM
signal=>PWM pin 5 in H-bridge
PORTC=0x00;                                           //   PORTC0=>DIR
pin(V=high) ; PORTC1=>BRAKE(V=L)
}
*******************************************************************
*******************************************************************
****
//*Written by: Penelope Herrera.              Created on: 10/27/03
//Code to run motors forward and backward.
//A brake function is included.
//Code requires a PWM

//**************Include statements**************************/
#include <inttypes.h>
#include <avr/io.h>
#include "lcd.h"
#include "lcd.c"
void front_motor_pwm_init(void){
        TCCR1A=0xA2;            //Set COM1A1, COM1B1, WGM11
        TCCR1B=0x12;            //Set CS11, WGM13
        ICR1H=0x4e; //Top value for 750HZ Frequency
        ICR1L=0x20;
        DDRB=0xFF;
        OCR1C=0x1388;
        }

void forward_drive (void){
                TCCR1A=0xA2;            //Set COM1A1, COM1B1, WGM11
                TCCR1B=0x12;            //Set CS11, WGM13
                ICR1H=0x4e; //Top value for 500HZ Frequency
                ICR1L=0x20;
                DDRB=0xFF;
                OCR1A=0x220; // Pin A14 goes to left servo
                OCR1B=0x9d0; // Pin A15 goes to right servo
        }

        void back_drive (void){
                TCCR1A=0xA2;            //Set COM1A1, COM1B1, WGM11
                TCCR1B=0x12;            //Set CS11, WGM13
                ICR1H=0x4e; //Top value for 500HZ Frequency
                ICR1L=0x20;
                DDRB=0xFF;
                OCR1A=0x9d0; // Pin A14 goes to left servo
                OCR1B=0x220; // Pin A15 goes to right servo
        }
        void brake_drive (void){
                TCCR1A=0xA2;            //Set COM1A1, COM1B1, WGM11
                TCCR1B=0x12;            //Set CS11, WGM13
                ICR1H=0x4e; //Top value for 500HZ Frequency
                ICR1L=0x20;
                DDRB=0xFF;
                OCR1A=0x05dc; // Pin A14 goes to left servo
                OCR1B=0x05dc; // Pin A15 goes to right servo
        }

        void drive_left(void){// To drive forward PWM, DIR and Brake signals are
        asserted
        DDRC=0xff;
        front_motor_pwm_init();                                 // Turn on
        PWM signal=>PWM pin 5 in H-bridge
        PORTC=0x80;     // PORTC0=>DIR pin(V=high);
        PORTC1=>BRAKE(V=L)
        }
        void drive_right(void){     // To drive forward PWM, DIR and Brake
signals     are asserted
        DDRC=0xff;
        front_motor_pwm_init();   // Turn on PWM signal=>PWM pin 5 in H-
bridge
```

```
        PORTC=0x00; // PORTC0=>DIR pin(V=high) ; PORTC1=>BRAKE(V=L)
        }
****************************************************************
****************************************************************
****
#include <inttypes.h>
#include <avr/io.h>
#include "lcd.h"
#include "lcd.c"
uint8_t ir_value;
uint8_t x;
uint8_t temp;
void delay(int len){    //This is a delay that will give milliseconds
        while(len)
        {
        len--;
        int k;
        for (k=0; k<1567; k++)
        asm ("nop");
        }
}
void ir_init (void){
        ACSR=0x80;
        ADMUX= 0x60; //when initializing the A/D system I chose to the AVCC
and     external capacitor; and chanel 0
        ADCSRA=0XE0; // enable the A/D, start the conversions, free runing
mode
        }
void display_left_ir_data(void){
        ADMUX= 0x60;                    // Choosing chanel 0
        ADCSRA=0XE0;                    // enable the A/D, start the conversions,
        free runing mode
        delay(200);
        lcd_send_str("LEFT ");
        delay(200);
        lcd_send_command(0x10);
        delay(200);
        ir_value=ADCH; // read the high byte of ADC only 8 bit presicion
        x=ir_value;
        x>>=4;
        temp= x+0X30;
        lcd_send_byte(temp);
        ir_value &=0X0f;
        x=ir_value;
        temp= x+0X30;
        lcd_send_byte(temp);
        delay(100);
        }

void display_right_ir_data(void){
        ADMUX= 0x61;                    // Choosing chanel 0
        ADCSRA=0XE0;                    // enable the A/D, start the conversions,
free runing mode
        delay(200);
        lcd_send_str("RIGHT ");
        delay(200);
        lcd_send_command(0x10);
        delay(200);
        ir_value=ADCH; // read the high byte of ADC only 8 bit presicion
        x=ir_value;
        x>>=4;
        temp= x+0X30;
        lcd_send_byte(temp);
        ir_value &=0X0f;
        x=ir_value;
        temp= x+0X30;
        lcd_send_byte(temp);
        delay(100);
        }
/***Main Program: Should display the left and right IR Values on LDC**/
void main (void){
        lcd_set_ddr();
        lcd_init();
        delay(200);
        lcd_send_str("Init Done");
        delay(300);
        lcd_send_command(0x01);
        while (1){
        display_left_ir_data();
        delay(200);
        lcd_send_command(0xC0);
        delay(200);
        display_right_ir_data();
        delay(200);
        lcd_send_command(0x02);
        }
        }
```

```
*****************************************************************
*****************************************************************
****
#include <inttypes.h>
#include <avr/io.h>
#include "lcd.h"
#include "lcd.c"
uint8_t ir_value;
uint8_t x;
uint8_t temp;
uint8_t IR_left;
uint8_t IR_right;
void delay(int len){    //This is a delay that will give milliseconds
        while(len)
        {
        len--;
        int k;
        for (k=0; k<1567; k++)
        asm ("nop");
        }
}
void PWM_Init (void){
        TCCR1A=0xA2;                //Set COM1A1, COM1B1, WGM11
        TCCR1B=0x12;                //Set CS11, WGM13
        ICR1H=0x4e; //Top value for 750HZ Frequency
        ICR1L=0x20;
        DDRB=0xFF;
        TCCR3A=0xA2;                //Set COM1A1, COM1B1, WGM11
        TCCR3B=0x12;                //Set CS11, WGM13
        ICR3H=0x4e;
        ICR3L=0x20;
        DDRE=0xFF;

}
void drive_left(void){       // To drive forward PWM, DIR and Brake signals are
asserted
        OCR1C=0x1388;
        OCR3A=0x3000;
        PORTC=0x80;                                              // PORTC0=>DIR
pin(V=high) ; PORTC1=>BRAKE(V=L)
}
void Brake(void){ // To drive forward PWM, DIR and Brake signals are asserted
        OCR1A=0x5dc; // Pin A14 goes to left servo
        OCR1B=0x5dc; // Pin A15 goes to right servo
        PORTC=0x40;                                               // PORTC0=>DIR
        pin(V=high) ; PORTC1=>BRAKE(V=L)
}
void drive_right(void){       // To drive forward PWM, DIR and Brake signals are
asserted
        OCR1C=0x1388;
        OCR3A=0x3000;
        PORTC=0x00;
}
void drive_forward (void){
                OCR1A=0x220; // Pin A14 goes to left servo
                OCR1B=0x9d0; // Pin A15 goes to right servo
        }
void drive_backward (void){
                OCR1A=0x9d0; // Pin A14 goes to left servo
                OCR1B=0x220; // Pin A15 goes to right servo
        }
void ir_init (void){
        ACSR=0x80;
        ADMUX= 0x60; //when initializing the A/D system I chose to the AVCC
and     external capacitor; and chanel 0
        ADCSRA=0XE0; // enable the A/D, start the conversions, free runing
mode
        }
void display_left_ir_data(void){
        lcd_send_str("LEFT ");
        delay(200);
        lcd_send_command(0x10);
        delay(200);
        ir_value=ADCH; // read the high byte of ADC only 8 bit presicion
        x=ir_value;
        x>>=4;
        temp= x+0X30;
        lcd_send_byte(temp);
        ir_value &=0X0f;
        x=ir_value;
        temp= x+0X30;
        lcd_send_byte(temp);
        delay(100);
        }
void display_right_ir_data(void){
        lcd_send_str("RIGHT ");
        delay(200);
        lcd_send_command(0x10);
        delay(200);
        ir_value=ADCH; // read the high byte of ADC only 8 bit presicion
        x=ir_value;
        x>>=4;
        temp= x+0X30;
        lcd_send_byte(temp);
        ir_value &=0X0f;
        x=ir_value;
        temp= x+0X30;
        lcd_send_byte(temp);
        delay(100);
        }
void get_ir_data(void){
        ADMUX= 0x60; // Choosing chanel 0
        ADCSRA=0XE0;// enable the A/D, start the conversions, free runing mode
        delay(200);
        IR_left=ADCH; // read the high byte of ADC only 8 bit presicion
        ADMUX= 0x61; // Choosing chanel 0
        ADCSRA=0XE0;// enable the A/D, start the conversions, free runing mode
        delay(200);
        IR_right=ADCH; // read the high byte of ADC only 8 bit presicion
        }
void object_avoid(void){
if ((IR_left > 68) && (IR_right >68)){
        Brake();
        delay(60);
        drive_backward();
        delay(50);
        lcd_send_str("I hit a pedestrain");
        delay(200);
        lcd_send_command(0xc0);
        delay(200);
        lcd_send_str("Dialing 911");
        delay(500);
        lcd_send_command(0x01);
        Brake();
        delay(50);
        drive_left();
        delay(40);
        drive_forward();
        }
        else {if (IR_left >68){
        Brake();
        delay(60);
```

```
	drive_backward();
	delay(50);
	lcd_send_str("I hit a pedestrain");
	delay(200);
	lcd_send_command(0xc0);
	lcd_send_str("Dialing 911");
	delay(200);
	lcd_send_command(0x01);
	Brake();
	delay(50);
	drive_right();
	delay(40);
	drive_forward();
	}
	else {if (IR_right >68){
	Brake();
	delay(60);
	drive_backward();
	delay(50);
	lcd_send_str("I hit a pedestrain");
	delay(500);
	lcd_send_command(0x01);
	delay(200);
	lcd_send_str("Dialing 911");
	delay(500);
	lcd_send_command(0x01);}
	Brake();
	delay(50);
	drive_left();
	delay(40);
	drive_forward();
}}}

/***Main Program: Should display the left and right IR Values on LDC**/
void main (void){
	DDRC=0xf8;
	lcd_set_ddr();
	lcd_init();
	delay(200);
	lcd_send_str("Init Done");
	delay(300);
	lcd_send_command(0x01);
	ir_init();
	PWM_Init();
```

```
        lcd_send_str("PWM Init");
        delay(300);
        lcd_send_command(0x01);
        drive_forward();
        while (1){
        lcd_send_str("ROUTE 20");
        delay(300);
        lcd_send_command(0x01);
        delay(300);
        get_ir_data();
        display_left_ir_data();
        delay(200);
        lcd_send_command(0xC0);
        delay(200);
        display_right_ir_data();
        delay(200);
        lcd_send_command(0x02);
        object_avoid();
        }
        }
****************************************************************
****************************************************************
****
//*Written by: Penelope Herrera.                Created on: 10/27/03
//Code to do line following
//It run motors forward,backward, left and right.
//A brake function is included.
//Code requires a PWM, AD system and input from line following circuit

//**************Include statements***************************/
#include <inttypes.h>
#include <avr/io.h>
#include "lcd.h"
#include "lcd.c"
uint8_t right;
uint8_t left;
uint8_t center;
uint8_t IR_left;
uint8_t IR_right;
uint8_t x;
uint8_t temp;
void delay(int len){    //This is a delay that will give milliseconds
        while(len){
        len--;
        int k;
        for (k=0; k<1567; k++)
        asm ("nop");
        }
}
void PWM_Init (void){
        TCCR1A=0xA2;              //Set COM1A1, COM1B1, WGM11
        TCCR1B=0x12;              //Set CS11, WGM13
        ICR1H=0x4e; //Top value for 750HZ Frequency
        ICR1L=0x20;
        DDRB=0xFF;

        TCCR3A=0xA2;              //Set COM1A1, COM1B1, WGM11
        TCCR3B=0x12;              //Set CS11, WGM13
        ICR3H=0x4e;
        ICR3L=0x20;
        DDRE=0xFF;
}
void drive_left(void){      // To drive forward PWM, DIR and Brake signals are
asserted
        OCR1C=0x1388;
        OCR3A=0x3000;
        PORTC=0x80;                                          // PORTC0=>DIR
pin(V=high) ; PORTC1=>BRAKE(V=L)
}
void Brake(void){      // To drive forward PWM, DIR and Brake signals are
asserted
        OCR1A=0x5dc; // Pin A14 goes to left servo
        OCR1B=0x5dc; // Pin A15 goes to right servo
        PORTC=0x40;                                          // PORTC0=>DIR
pin(V=high) ; PORTC1=>BRAKE(V=L)
}
void drive_right(void){      // To drive forward PWM, DIR and Brake signals are
asserted
        OCR1C=0x1388;
        OCR3A=0x3000;
        PORTC=0x00;
}
void drive_forward (void){
        OCR1A=0x220; // Pin A14 goes to left servo
        OCR1B=0x9d0; // Pin A15 goes to right servo
        }
void drive_backward (void){
        OCR1A=0x9d0; // Pin A14 goes to left servo
```

```
         OCR1B=0x220; // Pin A15 goes to right servo
         }
void line_following(void){
         right=PIND;
         right &=0x01;
         center=PIND;
         center &=0x02;
         left=PIND;
         left &=0x03;
         lcd_send_str("ENTERING Line F");
         delay(500);
         lcd_send_command(0x02);
         delay(200);

         lcd_send_str("Line= ");
         delay(500);
         lcd_send_command(0x10);
         delay(200);
         x=PIND;
         x>>=4;
         temp= x+0X30;
         lcd_send_byte(temp);
         PIND &=0X0f;
         x=PIND;
         temp= x+0X30;
         lcd_send_byte(temp);
         delay(100);
 if ((center=0) && (left>0) && (right>0)){
          drive_forward();
          lcd_send_command(0x00);
         lcd_send_command(0x01);
          lcd_send_str("Going forward now");
         delay(500);
         lcd_send_command(0x00);
         lcd_send_command(0x01);
          }
else {if((center=0) && (right=0) && (left>0)){
         drive_right();
         lcd_send_str("Line to the right");
         delay(500);
         lcd_send_command(0x00);
         lcd_send_command(0x01);}
else {if((center=0) && (left=0) && (right>0)){
         drive_left();
         lcd_send_str("Line to the left");
         delay(500);
         lcd_send_command(0x00);
         lcd_send_command(0x01);}
else {if((center>0) && (right=0) && (left>0)){
         drive_right();
         lcd_send_str("Line to the right");
         delay(500);
         lcd_send_command(0x00);
         lcd_send_command(0x01);}
else {if((center>0) && (left=0) && (right>0)){
         drive_left();
         lcd_send_str("Line to the left");
         delay(500);
         lcd_send_command(0x00);
         lcd_send_command(0x01);}
          }}}}}
 void main (void){
         DDRC=0x00;
         lcd_set_ddr();
         lcd_init();
         delay(200);
         lcd_send_str("Init Done");
         delay(300);
         lcd_send_command(0x00);
         lcd_send_command(0x01);
         PWM_Init();
         lcd_send_str("PWM Init");
         delay(300);
         lcd_send_command(0x00);
         lcd_send_command(0x01);
         drive_forward();
         while (1){
          lcd_send_str("Route 20");
          delay(800);
          lcd_send_command(0xc0);
          delay(200);
          line_following();
          }}
```