Final Project Design Report:
Voice Controlled Robot Car

Team VR:
Name: Chris Do
Email: chrisdo@ufl.edu
Phone: 8135271966

Name: Kyle Stevenson
Email: relakyle@ufl.edu
Phone: 8132051287

Abstract
The idea for this project is to build a robotic car and use it as a platform for testing voice recognition. The system allows the user to input voice commands through a microphone or use directional buttons to control the movement and other functions of the car. The user interface controller consists of a graphic LCD, a joystick, control buttons and a voice recognition module connected to a microcontroller. The remote controller is wireless connected to a microcontroller on the robot car. Added to the car are IR object detection sensors that will detect obstructions in its path and signal the user interface controller. The user will have the option to change the path of the car manually or configure automatic object avoidance. The LCD displays the relative speed and direction of the car as well as current obstructions of the sensors.
Table of Contents

List of Tables and Figures ................................................................. 3
Project Features/Objectives ........................................................... 4
Components/Concept Selection ...................................................... 6
Summary of costs ........................................................................... 7
Diagrams ..................................................................................... 9
Flowcharts .................................................................................. 11
Separation of Labor ..................................................................... 12
Gantt Chart ................................................................................ 14
Appendix .................................................................................... 15
Code ........................................................................................... 16
References .................................................................................. 32
List of Tables and Figures

Pictures

- Picture 1. XBee Wireless Chip.................................................................4
- Picture 2. Robot Car.............................................................................4
- Picture 3. Sharp IR Sensor.................................................................4

Diagrams

- Figure 1. Top-Level Design View..........................................................8
- Figure 2. Car Controller Diagram.........................................................9
- Figure 3. Remote Controller Diagram..................................................9

Tables

- Table 1. Summary of Costs.................................................................8
- Table 2. Separation of Responsibility...................................................10
- Table 3. Separation of Labor.............................................................10

Gantt Chart

- Gantt Chart..........................................................................................11
Project Features/Objectives

The voice-controlled device finds applications in the advancement of unmanned vehicles. Unmanned vehicles are becoming popular in military, police and rescue operations in which the situation is dangerous or inaccessible by human contact. The unmanned devices can be programmed or designed for several different applications or missions. The car will be able to detect objects inside that structure that might have been previously unknown by the user. The ability to control the device with voice commands can give the user hands-free convenience allowing for multi-tasking.

Wireless Transmission

We are using X-Bee chips in order to control the Robot car remotely. These are 2.4GHz chips that have a rated signal range of 1600m. There is an X-Bee chip on the car, and on the remote. The remote controller uses the X-Bee to transmit speed and direction commands and configuration options to the robotic car. The car will transmit the status of the object sensors back to the remote controller.

Robot Car

The car is a simple kit composed of two DC motors and a chassis. A microprocessor and motor controller is added to give us control over the function of the motors. The speed and direction of the car can be controlled by specific commands sent to the motor controller. The car can also detect objects and avoid them, using IR sensors. The logic on the car is powered with a 9V battery regulated to 5V, while the motors are powered by two AA batteries.
Voice Recognition

The user is able to control the direction of the car using voice commands that can be trained by the user. The chip purchased was the HM2007. This chip records various voice patterns for a certain command, and when that pattern is detected by the chip, it outputs a corresponding code to the microprocessor. The input is very sensitive to background noise and a simple bandpass filter between 300 – 3000Hz is added to the front end of the microphone input on the chip.

Remote Control

The car is controlled wirelessly with a remote that we designed. This remote has a joystick to control the car manually, as well as the option to control the car using voice commands. The remote also includes a graphic LCD to display the car control program including configuration options as well as direction and speed. A microphone connected to the remote to receive the voice commands. The remote is powered with a 9V battery regulated to 5V.
Components/Concept Selection

The following is a component list for the final design.

Atmega324p

- Cost: ~$8 each
- Qty.: 2

The Atmega324p is the device used for the two microcontrollers needed in this design. This microcontroller was chosen because of availability and previous experience with the device. The Atmega provides a serial output port that will be useful for communicating with the motor controller. The two controllers that are described is the user interface controller and the RC car controller. The Atmega324P was chosen over the Atmega32 because it has two USART’s on it. We needed one for the X-Bee chip and one for the motor controller.

XBee Pro

- Cost: ~$40 each
- Qty.: 2

The option for wireless chips were between Nordic, XBee, and Bluetooth chips. Bluetooth chips were more expensive, and we have no experience with them. Between the Nordic and XBee's, the XBee chips are easier to program and offer a further signal range that was beneficial to our project.

HM2007

- Cost: ~$25
- Qty.: 1

The HM2007 is the voice recognition chip that will be used to receive voice commands. When researching voice recognition chips, the HM2007 and the VR Stamp module were the only two chips that were available. We chose the HM2007 because of the simplicity of the programming instructions and its basic functionality was the closest to what we will be using it for.

Motor Controller

- Cost: ~$55 (sold with Motor Gearbox and Chassis)
- Qty.: 1

The motor controller will be used to output the necessary signals to operate the motion of the car. It was chosen to provide an interface between the Atmega324p microcontroller on the car to the gear motors. The motor controller we are using was chosen because is was conveniently included in a kit with a motor gearbox so we knew it would work.
Twin-Motor Gearbox Kit/Chassis

Cost: ~$55 (so with Motor Controller)  
Qty: 1

The twin-motor gearbox kit includes two motors, wheels and axles, ball caster assembly, and the motor controller. The motors will be used to drive the wheels to give motion to the car. This particular kit was chosen because it contained simple, essentially bare-bone components which will allow us to better customize the car to fit our needs. A basic chassis will also be used to hold the gearbox components and later used to attach the electrical components. The car design is meant to be small and compact in order to have more options for maneuverability.

LCD/Control Buttons

Cost: ~$20

The LCD will provide a visual interface from the remote control to the user. It will consist of a square screen used to display requests and warnings from the program. The LCD displays the sensor readings and will display whether or not an object is in front of the sensors and which sensor is being read. The LCD must be large enough to effectively display the map to the user but must also be simple enough to interface with our microcontroller. The LCD we chose was a graphic 128x64 LCD. Control push buttons and a joystick are used to receive user inputs.

Sensors

Cost: ~$10  
Qty.: 4

The primary sensor to be used on the car is a proximity sensor to detect when the car comes in close contact with an object. The sensors we chose are the Sharp IR sensors, and are positioned with one in the back, one in the front, and one on the front right area, and one in the front left area.
## Summary of Costs

<table>
<thead>
<tr>
<th>PART</th>
<th>UNIT COST</th>
<th>QTY.</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmega324p</td>
<td>$8.00</td>
<td>2</td>
<td>$16.00</td>
</tr>
<tr>
<td>Xbee Pro</td>
<td>$40.00</td>
<td>2</td>
<td>$80.00</td>
</tr>
<tr>
<td>HM2007</td>
<td>$25.00</td>
<td>1</td>
<td>$25.00</td>
</tr>
<tr>
<td>Motor Controller/Gearbox Kit</td>
<td>$55.00</td>
<td>1</td>
<td>$55.00</td>
</tr>
<tr>
<td>LCD/Input Controls</td>
<td>$25.00</td>
<td>1</td>
<td>$25.00</td>
</tr>
<tr>
<td>Sensors</td>
<td>$10</td>
<td>4</td>
<td>$40.00</td>
</tr>
<tr>
<td><strong>TOTAL ESTIMATED COSTS</strong></td>
<td></td>
<td></td>
<td><strong>$241.00</strong></td>
</tr>
</tbody>
</table>

Table 1. Summary of Costs
Diagrams

The following diagrams describe the overview of the design and the basic data flow between the different components.

**Top-Level View**

![Top-Level Design Diagram](image1)

**Figure 1. Top-Level Design Diagram**

**Car Controller**

![Car Controller Diagram](image2)

**Figure 2. Car Controller Diagram**
Figure 3. Remote Controller Diagram
Flowcharts

Remote Control software flow chart

Interrupted State

Data received from car?

Update LCD

Train

Display Voice command codes

Wait for User input

Y

Back

N

Command

Voice Recognition ON/OFF

Joystick direction

Voice command

Stop

Forward

Reversal

Left

Right

Update Speed

Transmit direction and speed

Update Aim Input

Update Object Avoidance Option

Update LCD

Main Menu

Startup
Separation of Labor

The following table is a preliminary list of each partners responsibilities, although, a lot of the work will be done together. This list is subject to change.

<table>
<thead>
<tr>
<th>Chris</th>
<th>Kyle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Research</td>
<td></td>
</tr>
<tr>
<td>Ordering Parts</td>
<td></td>
</tr>
<tr>
<td>Robot car assembly</td>
<td></td>
</tr>
<tr>
<td>X-Bee chip coding</td>
<td>Voice Recognition Chip Coding</td>
</tr>
<tr>
<td>Object Detection</td>
<td>LCD coding</td>
</tr>
<tr>
<td>Build Prototype</td>
<td></td>
</tr>
<tr>
<td>Test/Debug Prototype</td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Separation of Responsibility

<table>
<thead>
<tr>
<th>Item</th>
<th>Chris Do</th>
<th>Kyle Stevenson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Atmel Programming</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Voice Chip Programming</td>
<td>10%</td>
<td>90%</td>
</tr>
<tr>
<td>PCB Robot Design</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>Xbee Programming</td>
<td>90%</td>
<td>10%</td>
</tr>
<tr>
<td>Motor Controller Programming</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>Breadboard prototyping</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>PCB Remote Design</td>
<td>60%</td>
<td>40%</td>
</tr>
<tr>
<td>Sensor Programming</td>
<td>50%</td>
<td>50%</td>
</tr>
<tr>
<td>LCD Interface</td>
<td>50%</td>
<td>50%</td>
</tr>
</tbody>
</table>

Table 3. Separation of Labor
Appendix

Robot side view

Robot top view

Remote control
Car C-Code
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <inttypes.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <util/LCD.h>
#define  RESET_PORT PORTD
#define  RESET_DDR DDRD
#define  LEFT 0xF0
#define  RIGHT 0xF1
#define  BACK 0xF2
#define  FWD 0xF3
#define  HONK 0xF4
#define  BUZZER_PORT PORTB
#define  BUZZER_DDR DDRB
#define  BUZZER_PIN 0
#define  CH_DIR 0xFA
#define  CH_SPD 0xFC

unsigned char SPEED, DIR, AUTO=0;
unsigned char f,l,r,b;
int sensor_fr, sensor_lf, sensor_rt, sensor_bk;

void USART_Init( unsigned int baud )
{
  /* Set baud rate */
  UBR0H = (unsigned char)(baud>>8);
  UBR0L = (unsigned char)baud;
  /* Enable receiver and transmitter */
  UCSR0B = (1<<RXEN0)|(1<<TXEN0);
  /* Set frame format: 8data, 1stop bit */
  UCSR0C = (3<<UCSZ00);
}

void USART0_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSR0A & (1<<UDRE0)) );
  /* Put data into buffer, sends the data */
  UDR0 = data;
}

void USART1_Transmit( unsigned char data )
{
  /* Wait for empty transmit buffer */
  while ( !( UCSR1A & (1<<UDRE1)) );
  /* Put data into buffer, sends the data */
  UDR1 = data;
}

unsigned char USART_Receive( void )
{
  /* Wait for data to be received */
}
while (!(UCSR1A & (1<<RXC1)))
/* Get and return received data from buffer */
return UDR1;
}

void CONFIG_motor()
{
    USART0_Transmit(0x80);
    USART0_Transmit(0x02);
    USART0_Transmit(0x06);
}

void MOTOR_init()
{
    RESET_DDR |= 0b01000000;  // set portd.0 to output
    RESET_PORT |= 0x00; // set portd.0 low
    _delay_ms(20);  // delay
    RESET_PORT |= 0b01000000; // set portd.0 high
}

void motion(unsigned char SPEED, unsigned char SPDIRA, unsigned char SPDIRB)
{
    USART0_Transmit(0x80);
    USART0_Transmit(0x00);
    USART0_Transmit(SPDIRA);
    USART0_Transmit(SPEED);
    USART0_Transmit(0x80);
    USART0_Transmit(0x00);
    USART0_Transmit(SPDIRB);
    USART0_Transmit(SPEED);
}

// Provide the duration and period of buzzer signal in ms
void BUZZ(float duration, float period)
{
    long int cycles;
    float half_period; // Initialize variables

cycles=duration/period; // Compute the number of cycles to loop toggling the pin
half_period = period/2; // Compute a half cycle period

BUZZER_DDR = (1 << BUZZER_PIN) | BUZZER_DDR; // Set the port for the buzzer output

for (i=0;i<cycles;i++) // Toggle the speaker the appropriate number of cycles
{
    _delay_ms(half_period); // Wait a half cycle to toggle port pin
    BUZZER_PORT = (1 << BUZZER_PIN) | BUZZER_PORT; // Set the port pin
    _delay_ms(half_period); // Wait a half cycle to clear the port pin
    BUZZER_PORT = ~(1 << BUZZER_PIN) & BUZZER_PORT; // Clear the port pin
}

return; // Return to the main program
}

void ADC_INIT(void)
{
    ADMUX=0x00;
    ADCSRA=0b10000110;
}

int ADC_START(unsigned char channel)
{
    ADMUX = channel;  // Start conversion
    ADCSRA |= (1 << ADSC);  // wait for conversion to complete
    return ADC;
}

unsigned char SPDIRA = 0x01;
unsigned char SPDIRB = 0x03;

ISR(USART1_RX_vect)
{
    char cmd;
    cmd=UDR1;
    if((cmd&0x80)==0x80)
    {
        if(cmd==0xAC)
        {
            AUTO=1;
        }
        else if(cmd==0xAD)
        {
            AUTO=0;
        }
        else if(cmd==HONK)
        {
            LCD_STRING("HONKED");
            BUZZ(75,0.5);
        }
        else
        {
            DIR = cmd;
            switch (DIR)
            {
                case LEFT:
                    SPDIRA = 0x02;
                    SPDIRB = 0x01;
                    break;

                case RIGHT:
                    SPDIRA = 0x03;
                    SPDIRB = 0x00;
                    break;

                case FWD:
                    SPDIRA = 0x01;
                    SPDIRB = 0x03;
                    break;

                case BACK:
                    SPDIRA = 0x00;
                    SPDIRB = 0x02;
                    break;
            }
        }
    }
    else if((cmd&0x80)==0x00)
    {
        SPEED = cmd;
    }
}

void Check_Sensors(){
    sensor_fr = ADC_START(0);
sensor_lf = ADC_START(1);
sensor_rt = ADC_START(2);
sensor_bk = ADC_START(3);

// char lcd_f[20], lcd_l[20], lcd_r[20], lcd_b[20];
sprintf(lcd_f, "%d", sensor_fr);
sprintf(lcd_l, "%d", sensor_lf);
sprintf(lcd_r, "%d", sensor_rt);
sprintf(lcd_b, "%d", sensor_bk);
    LCD_COMMAND(LCD_CLEAR_HOME);
    LCD_STRING("F:");
    LCD_STRING(lcd_f);
    LCD_ADDR(0x0A);
    LCD_STRING("L:");
    LCD_STRING(lcd_l);
    LCD_ADDR(0x40);
    LCD_STRING("R:");
    LCD_STRING(lcd_r);
    LCD_ADDR(0x4A);
    LCD_STRING("B:");
    LCD_STRING(lcd_b);
    _delay_ms(100);

f = (sensor_fr > 400 ? 1:0);
l = (sensor_lf > 400 ? 1:0);
r = (sensor_rt > 400 ? 1:0);
b = (sensor_bk > 400 ? 1:0);

int main(void)
{
    USART_Init(51);
    _delay_ms(20);
    LCD_INIT();
    _delay_ms(500);
    LCD_STRING("Starting...");
    _delay_ms(1000);

    ADC_INIT();
    DDR_A=0x00;
    _delay_ms(20);

    SPEED = 0x00;

    f = 0;
l = 0;
r = 0;
b = 0;
    int x=0;
    int beeped=0;
    int x1=0;

    // MOTOR_init();
    // delay_ms(20);
    // CONFIG_motor();
    sei();
    while(1)
    {

        Check_Sensors();
        x = ((f<<3) | (l<<2) | (r<<1) | b);
if(x!=x1)
    USART1_Transmit(x);
x1=x;

if ((DIR == FWD) & ((f==1) | (l==1) | (r==1)) & (SPEED!=0))
{
    if(AUTO==1)
    {
        if(sensor_lf>sensor_rt)
            {motion(SPEED, 0x03, 0x00);}  
        else {motion(SPEED, 0x02, 0x01);}  
    }
    else{
        motion(0x00, SPDIRA, SPDIRB);
        LCD_ADDR(0x00);
        LCD_STRING("OBJECT DETECTED");
        if(beeped==0){
            BUZZ(75.0.5);
            beeped=1;
        }
    }
}

else if ((DIR == BACK) & (b==1) & (SPEED!=0))
{
    if(AUTO==1)
    {
        motion(00, 0x02, 0x01);
    }
    else{
        motion(0x00, SPDIRA, SPDIRB);
        LCD_ADDR(0x00);
        LCD_STRING("OBJECT DETECTED");
        if(beeped==0){
            BUZZ(75.0.5);
            beeped=1;
        }
    }
}

else{
    motion(SPEED, SPDIRA, SPDIRB);
    beeped=0;

    if(SPEED==0)
    {
        LCD_ADDR(0x00);
        LCD_STRING("Stopped   ");
    }
    else{
        switch(DIR){
        case LEFT:
            LCD_ADDR(0x00);
            LCD_STRING("Turning LEFT ");
            break;
        case RIGHT:
            LCD_ADDR(0x00);
            LCD_STRING("Turning RIGHT ");
            break;
        case FWD:
            LCD_ADDR(0x00);
            LCD_STRING("Going       ");
            break;
        default:
            LCD_ADDR(0x00);
            LCD_STRING("Invalid command");
        }
    }
}
LCD_ADDR(0x00);
LCD_STRING("Going FORWARD ");
break;

case BACK:
  LCD_ADDR(0x00);
  LCD_STRING("Going BACKWARD ");
  break;

}
Remote control C-Code
#include <inttypes.h>
#include <util/delay.h>
#include <avr/io.h>
#include <stdlib.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include "displaygraphic.c"

#define LEFT 0xF0
#define RIGHT 0xF1
#define BACK 0xF2
#define FWD 0xF3

void Interrupt_Init(){
    PCICR = (1<<3);
    PCMSK3 = (1<<7);
}

//USART Initialization
void USART_Init( unsigned int baud )
{
    /* Set baud rate */
    UBRR0H = (unsigned char)(baud>>8);
    UBRR0L = (unsigned char)baud;
    /* Enable receiver and transmitter */
    UCSR0B = (1<<RXEN0)|(1<<TXEN0)|(1<<RXCIE0);
    /* Set frame format: 8data, 1stop bit */
    UCSR0C = (3<<UCSZ00);
    return;
}

void USART0_Transmit( unsigned char data )
{
    /* Wait for empty transmit buffer */
    while ( !( UCSR0A & (1<<UDRE0)) );
    /* Put data into buffer, sends the data */
    UDR0 = data;
    return;
}

unsigned char USART0_Receive( void )
{
    /* Wait for data to be received */
    while ( !(UCSR0A & (1<<RXC0)) );
    /* Get and return received data from buffer */
    return UDR0;
}

// ADC Initialization
void ADC_INIT(void)
{
    ADMUX = (1<<5);
    ADCSRA = 0b10000110; // ADC clock prescalar of 64
}

uint8_t ADC_START(unsigned char channel)
{
    ADMUX = channel|(1<<5);
    ADCSRA |= (1 << ADSC); // Start conversion
    while(ADCSRA & (1 << ADSC)); // wait for conversion to complete
}
return ADCH;
}

//LCD Sensor Displays
void LCD_senfr(uint8_t bit){
    uint8_t x,y;
    y=6;
    while(y<8){
        for (x=58;x<74;++x){
            lcd_setbit(x,y,bit);
            y++;
        }
        lcd_flush();
    }
}

void LCD_senlf(uint8_t bit){
    uint8_t x,y,c;
    y=14;
    for(x=48;x<56;++x){
        for(c=0;c<2;++c){
            lcd_setbit(x,y,bit);
            y--;
        }
        y++;
    }
    lcd_flush();
}

void LCD_senrt(uint8_t bit){
    uint8_t x,y,c;
    y=6;
    for(x=76;x<84;++x){
        for(c=0;c<2;++c){
            lcd_setbit(x,y,bit);
            y++;
            y--;
        }
    }
    lcd_flush();
}

void LCD_senbk(uint8_t bit){
    uint8_t x,y;
    y=43;
    while(y<45){
        for(x=58;x<74;++x){
            lcd_setbit(x,y,bit);
            y++;
        }
    }
    lcd_flush();
}

//Data Bus External Interrupt Vector
uint8_t word;
ISR(PCINT3_vect){
    word = PINC;
}

uint8_t f,l,r,b;

//USART Receive Interrupt Vector
ISR(USART0_RX_vect){
    uint8_t flag;
    flag = UDR0;
    f = ((flag&0x08)>>3);
    l = ((flag&0x04)>>2);
    r = ((flag&0x02)>>1);
b = ((flag&0x01));

uint8_t x,y;

y=1;
while(y<9){
    for(x=58;x<74;++x){
        lcd_setbit(x,y,0);
        y++;
    }
    lcd_flush();
}

y=43;
while(y<51){
    for(x=58;x<74;++x){
        lcd_setbit(x,y,0);
        y++;
    }
    lcd_flush();
}

if (f==1){
    LCD_senfr(1);
}

if (l==1) {
    LCD_senlf(1);
}else{
    LCD_senlf(0);
}

if (r==1) {
    LCD_senrt(1);
}else{
    LCD_senrt(0);
}

if (b==1){
    LCD_senbk(1);
}

//LCD Main Menu Display
void display_main(){
    lcd_display_car();
    lcd_set_cursor(3,10);
    lcd_putstr("VR");
    lcd_set_cursor(7,0);
    lcd_putstr("Train");
    lcd_set_cursor(7,18);
    lcd_putstr("Cmd");
}

uint8_t spd;
uint16_t cnt=0;

void Update_Speed(){
    spd = ADC_START(3);
    lcd_set_cursor(1,0);
    lcd_putstr("SPD ");
    lcd_putstr((spd/28)+48);
    cnt=0;
}

void TRAIN(){
    lcd_clear();
    lcd_set_cursor(7,0);
    lcd_putstr("Back");
    lcd_set_cursor(0,0);
lcd_putstr("01 - FWD");
 lcd_set_cursor(1,0);
 lcd_putstr("05 - REV");
 lcd_set_cursor(2,0);
 lcd_putstr("10 - RHT");
 lcd_set_cursor(3,0);
 lcd_putstr("15 - LFT");
 lcd_set_cursor(4,0);
 lcd_putstr("20 - STOP");
 lcd_set_cursor(5,0);
 lcd_putstr("25 - HORN");
 lcd_set_cursor(5, 12);
 lcd_putstr="# - Train");
 lcd_set_cursor(6,12);
 lcd_putstr("* - Clear");

DDRD &= ~0b10000000; // set portd.7 to input
DDRC &= ~0xFF; // set portc input

Interrupt_Init();

sei();

while(1){

    lcd_set_cursor(3,15);
    lcd_putch((word>>4)+48);
    lcd_set_cursor(3,16);
    lcd_putch((word&0x0F)+48);

    if((PINA&0x10)==0x10){
        break;
    }
}

cli();
word=0;
 lcd_clear();
display_main();
}

//LCD Direction Displays
void LCD_FWD(uint8_t bit){

    int8_t y;
    uint8_t x,s,d;

    if ((f==0)&(l==0)&(r==0)){
        d=16;
        s=58;
        y=8;
        while(y>0){
            for (x=s; x<(d+58); ++x){
                lcd_setbit(x,y,bit);
                s++; d--; y--;
            }
        }
    }

    lcd_flush();
}

void LCD_BACK(uint8_t bit){

    int8_t y;
    uint8_t x,s,d;

    if(b==0){
        d=16;
        s=58;
        y=43;
    }
while(y<51){
    for (x=s; x<(d+58); ++x){
        lcd_setbit(x,y,bit);
    }
    s++; d--; y++;
}lcd_flush();
}

void LCD_RHT(uint8_t bit){
    int8_t y;
    uint8_t x,s,d;
    d=16;
    s=18;
    x=83;
    while(x<91){
        for (y=s; y<(d+18); ++y){
            lcd_setbit(x,y,bit);
        }
        s++; d--; x++;
    }lcd_flush();
}

void LCD_LFT(uint8_t bit){
    int8_t y;
    uint8_t x,s,d;
    d=16;
    s=18;
    x=48;
    while(x>40){
        for (y=s; y<(d+18); ++y){
            lcd_setbit(x,y,bit);
        }
        s++; d--; x--;
    }lcd_flush();
}

void Joystick(){
    uint8_t updnlfrt;
    updnlfrt = ADC_START(0);
    if (updnlfrt>0xF0){
        USART0_Transmit(FWD);
        _delay_us(20);
        spd = ADC_START(3);
        USART0_Transmit(spd>>1);
        LCD_FWD(1);
        while (updnlfrt>0xF0){
            updnlfrt=ADC_START(0);
            cnt++;
            if (cnt==4000){
                Update_Speed();
                USART0_Transmit(spd>>1);
            }
        }
        USART0_Transmit(0x00);
        LCD_FWD(0);
    }
}
else if (updn<0x0F) {
    USART0_Transmit(BACK);
    _delay_us(20);
    spd = ADC_START(3);
    USART0_Transmit(spd>>1);
}

LCD_BACK(1);

while (updn<0x0F) {
    updn=ADC_START(0);
    cnt++;
    if (cnt==4000) {
        Update_Speed();
        USART0_Transmit(spd>>1);
    }
}

USART0_Transmit(0x00);

LCD_BACK(0);
}

ifrt = ADC_START(1);

if (ifrt<0x0F) {
    USART0_Transmit(RIGHT);
    _delay_us(20);
    spd = ADC_START(3);
    USART0_Transmit(spd>>1);
}

LCD_RHT(1);

while(ifrt<0x0F) {
    ifrt=ADC_START(1);
    cnt++;
    if (cnt==4000) {
        Update_Speed();
        USART0_Transmit(spd>>1);
    }
}

USART0_Transmit(0x00);

LCD_RHT(0);
}

else if (ifrt>0xF0) {
    USART0_Transmit(LEFT);
    _delay_us(20);
    spd = ADC_START(3);
    USART0_Transmit(spd>>1);
}

LCD_LFT(1);

while(ifrt>0xF0) {
    ifrt=ADC_START(1);
    cnt++;
    if (cnt==4000) {
        Update_Speed();
        USART0_Transmit(spd>>1);
    }
}

USART0_Transmit(0x00);
void Voice()
{
    if (word==0x01)
    {
        USART0_Transmit(FWD);
        _delay_us(20);
        spd = ADC_START(3);
        USART0_Transmit(spd>>1);

        lcd_set_cursor(3,1);
        lcd_putstr("FWD ");

        LCD_FWD(1);
        LCD_BACK(0);
        LCD_LFT(0);
        LCD_RHT(0);
    }
    if (word==0x05)
    {
        USART0_Transmit(BACK);
        _delay_us(20);
        spd = ADC_START(3);
        USART0_Transmit(spd>>1);

        lcd_set_cursor(3,1);
        lcd_putstr("BACK ");

        LCD_FWD(0);
        LCD_BACK(1);
        LCD_LFT(0);
        LCD_RHT(0);
    }
    if (word==0x10)
    {
        USART0_Transmit(RIGHT);
        _delay_us(20);
        spd = ADC_START(3);
        USART0_Transmit(spd>>1);

        lcd_set_cursor(3,1);
        lcd_putstr("RHT ");

        LCD_FWD(0);
        LCD_BACK(0);
        LCD_LFT(0);
        LCD_RHT(1);
    }
    if (word==0x15)
    {
        USART0_Transmit(LEFT);
        _delay_us(20);
        spd = ADC_START(3);
        USART0_Transmit(spd>>1);

        lcd_set_cursor(3,1);
        lcd_putstr("LFT ");

        LCD_FWD(0);
        LCD_BACK(0);
        LCD_LFT(1);
        LCD_RHT(0);
    }
}
if (word==0x20){
    USART0_Transmit(0x00);
    lcd_set_cursor(3,1);
lcd_putstr("STOP");
    LCD_FWD(0);
    LCD_BACK(0);
    LCD_LFT(0);
    LCD_RHT(0);
}
if (word==0x25){
    USART0_Transmit(0xF4);
    lcd_set_cursor(3,1);
lcd_putstr("HORN");
    _delay_ms(500);
    USART0_Transmit(0xF4);
    word=0;
}
void CMD() {
    USART_Init(51);
lcd_clear();
lcd_display_car();
lcd_set_cursor(3,10);
lcd_putstr("VR");
lcd_set_cursor(7,0);
lcd_putstr("Back");
uint8_t AC=0;
uint8_t VR=0;
sei();
while(1){
    if(VR==0){
        Joystick();
    }else if (VR==1){
        Voice();
    }
    if((PIN&A&0b10000000)==0b10000000){
        _delay_ms(5);
        USART0_Transmit(0xF4);
    }
    if((PIN&A&0x20)==0x20){
        AC=AC^1;
        _delay_ms(1000);
        if (AC==0){ USART0_Transmit(0xAD); }
        else if (AC==1){ USART0_Transmit(0xAC);}
    }
    if((PIN&A&0x40)==0x40){
        VR=VR^1;
        _delay_ms(1000);
    }
    if((PIN&A&0x10)==0x10){break;}}
lcd_set_cursor(7,7);
if (AC==0){
    lcd_putstr("AutoOff");
} else if (AC==1){
    lcd_putstr("AutoOn ");
}

lcd_set_cursor(7,16);
if (VR==0){
    lcd_putstr("VrOff");
    PCICR = (0<<3);
    PCMSK3 = (0<<7);
    lcd_set_cursor(3,1);
    lcd_putstr(" ");
    USART0_Transmit(0x00);
    LCD_FWD(0);
    LCD_BACK(0);
    LCD_LFT(0);
    LCD_RHT(0);
} else if (VR==1){
    lcd_putstr("VrOn ");
    Interrupt_Init();
}

Update_Speed();
}

UCSR0B = (0<<RXCIE0);
cli();
lcd_clear();
display_main();

int main(void){
    DDRA &= ~(0xFF);
    DDRC &= ~(0xFF);
    DDRD &= ~(0x80);
    lcd_init();
    _delay_ms(1000);
    ADC_INIT();
    //USART_Init(51);
    lcd_set_cursor(3,6);
    lcd_putstr("Starting");
    _delay_ms(1000);
    lcd_putstr(",");
    _delay_ms(1000);
    lcd_putstr(",");
    _delay_ms(1000);
    lcd_putstr(",");
    _delay_ms(1000);
    lcd_clear();
    lcd_set_cursor(3,7);
    lcd_putstr("TEAM VR");
    lcd_set_cursor(5, 5);
    lcd_putstr("Kyle Stevenson");
    lcd_set_cursor(6, 5);
    lcd_putstr("Chris Do");
    _delay_ms(3000);
    lcd_clear();
display_main();

while(1){

if ((PINA&0x10)==0x10)
    TRAIN();
else if ((PINA&0x40)==0x40)
    CMD();
}

return 0;
References

1. Datasheets for Parts: Atmega324p, HM2007, XBee Pro, Motor Controller
2. Sparkfun.com
3. LCD Display code. http://jormungand.net/dev/avrlcd/