**FINAL PROJECT REPORT (FALL 2018)**

Embedded Systems (EEL6935)

 “Automated vehicle speed control system”

**University of South Florida**

Department of Electrical Engineering

Under the Guidance of

**Dr. Alexandro Castellanos**

By

Pragnesh Arjunbhai Patel (U39764176)

Kalpana Joshi (U87530830)

Barkha Bunkar (U69802728)

Sai Krishna Kandagatla (U15974415)

**Aim:**

Here we propose an “***Automated vehicle speed control system***” where the presence of GSM module helps keep a track of the vehicle and further provides real time safety and medical aid for the passengers in the case of an accident.

**Abstract:**

This project is researched and documented so as to be realizable in the form of a practical solution for the impending traffic problem presently observed in many countries. There have been significant investments made by the research community and the industry in the field of automobile to solve problems due to inadequate infrastructure, road safety enforcement and traffic regulation. In this Project, we focus on building an automated module for the vehicle targeting the speed sensitive zones like School zones and Hospital zones for controlling the speed of the vehicles remotely passing through the zones.

A conceptual idea is to place a transmitter at all exit and entry point of the interface of any two zones that transmits a message signal at carrier frequency, indicating the upper limit value of the zone’s speed range into which the vehicle is entering at that moment, to the receiver which gives the message as an input to a pre-programmed microcontroller embedded within the automobile which regulates the speed of the vehicle by controlling the voltage supply to the motor devices. The entire system is small in size and easy to assemble onto an existing vehicle without disturbing its present arrangement.

**Block Diagram:**

A close up of a white wall

Description automatically generated

*Figure1: Block Diagram showing various components to be used in the project*

**Components used:**

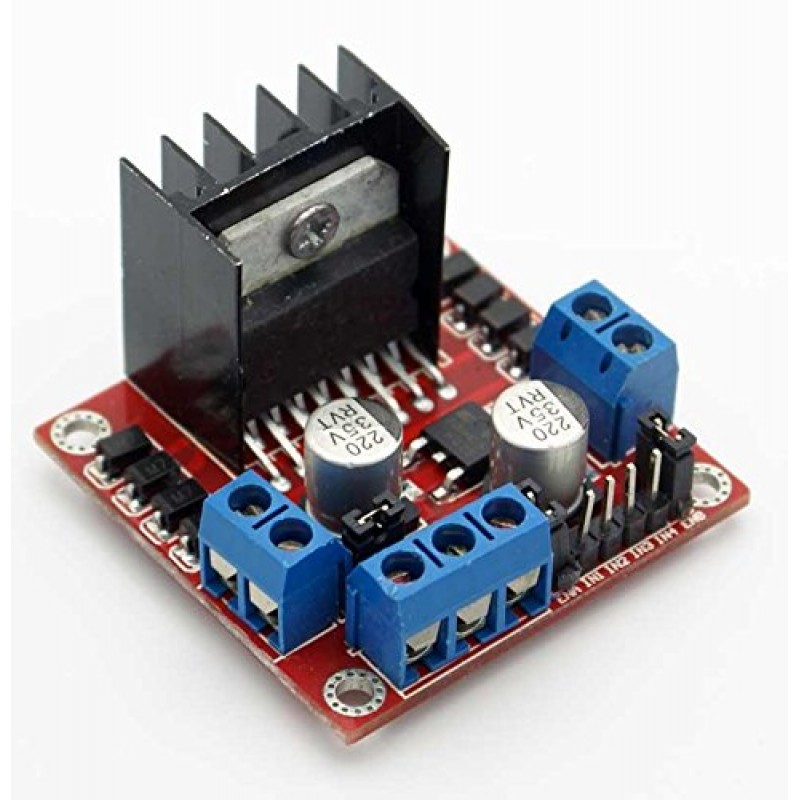
* Microchip PIC18F45K50 microcontroller
* Motor Driver IC
* DC Motor
* Bluetooth Module
* LCD
* RF Module
* Battery & Charger

**BLUETOOTH(HC05):**

A circuit board

Description automatically generatedA Bluetooth module used to control the prototype to give commands indicating it to move forward, backward, left and right.

**H-BRIDGE MOTOR DRIVER(L298):**

****A dual H-bridge to be used on board for motor control of the prototype for movement control. The design would be implemented using a Dual H bridge L293 model. The concept of H-bridge would possibly be implemented keeping in mind about the lecture on Enhanced PWM mode which would make it easy to implement and reduce the component requirements.

**DC MOTOR:**

**A close up of a device

Description automatically generated**DC Motor to drive around the prototype controlled by the Dual H-Bridge or Enhanced PWM mode function of PIC18F45K60. And a relay or PWM signal to control the speed of the Motor.

A circuit board

Description automatically generated**LCD MODULE:**

LCD module to display the zone entering and possibly we will also try to display the speed limits.

**RF MODULE:**

RF module to create the zones. The transmitter will be installed at the zones start and end boundary. The receiver will be mounted on the module. As soon as the prototype enters the zone and receives the signal from the transmitter the receiver picks up the signal and communicates it to the microcontroller that will trigger the relay/PWM to control the speed of the

**Project covering topic:**

* Embedded C programming
* Working on PIC18F45K50
* Working with communication protocols
* Working with Relays, LCD, PWM, interrupts and all the other components involved.
* User Interfacing with Mobile Application.

**Software Using:**

* MPLAB X IDE
* HC-05 Bluetooth Remote Android Application

**Pseudocode:**

**Step 1**: Start the program.

**Step 2**: Declare configuration bits by initializing the internal oscillator FOSC=INTOSCIO, WDTEN= OFF, PBADEN=ON, LVP=ON and generate the code for header file.

**Step 3**: Declare USART header file => #include <xc.h> In this header file we define USART transmitter and receiver functions, and also declare the Baud Rate.

**Step 4**: Then we declare main.c file in which we include configuration bits, header files and USART header files.

**Step 5**: Define variables for PORTD, PORTB and PORTA used in the program.

**Step 6**: Declare the functions required for LCD display, PWM and Bluetooth module.

**Step 7**: Define the functions of USART transmitter and receiver.

**Step 8**: Define Main of the program. Clear the TRISA, TRISB and TRISD.

**Step 9**: Declare oscillator frequency, T2CON and PR2 for the motor control. Call the USART initialize, LCD initialize and display string functions.

**Step 10**: Start an infinite loop where first the Bluetooth module is initialized which in turn also initializes the motor. When the RF module is triggered to 1, it initializes the LCD display functions like “**Caution and School Zone 20mph**” and then automatically PWM is triggered which in turn decreases the speed of the motor. Otherwise it resumes to travel in the normal speed without triggering the PWM.

**Step 11**: The LCD display functions are defined.

**Step 12**: The Bluetooth functions are defined in which the PWM and USART functions were called for the movement of the motor based on the instructions from the Bluetooth.

**Step 13**: The motor/PWM function are defined as:

1= forward

2=backward

3= Left

4= Right

**Step 14**: END the program.

**Code:**

**-----------------------------------------------Header Files-------------------------------------------------**

Configuration\_Bits.h: -

// PIC18F45K50 Configuration Bit Settings

// 'C' source line config statements

// CONFIG1L

#pragma config PLLSEL = PLL4X // PLL Selection (4x clock multiplier)

#pragma config CFGPLLEN = OFF // PLL Enable Configuration bit (PLL

Disabled (firmware controlled))

#pragma config CPUDIV = NOCLKDIV // CPU System Clock Postscaler (CPU uses system clock (no divide))

#pragma config LS48MHZ = SYS24X4 // Low Speed USB mode with 48 MHz system clock (System clock at 24 MHz, USB clock divider is set to 4)

// CONFIG1H

#pragma config FOSC = INTOSCIO // Oscillator Selection (Internal oscillator)

#pragma config PCLKEN = ON // Primary Oscillator Shutdown (Primary oscillator enabled)

#pragma config FCMEN = OFF // Fail-Safe Clock Monitor (Fail-Safe Clock Monitor disabled)

#pragma config IESO = OFF // Internal/External Oscillator Switchover (Oscillator Switchover mode disabled)

// CONFIG2L

#pragma config nPWRTEN = OFF // Power-up Timer Enable (Power up timer disabled)

#pragma config BOREN = SBORDIS // Brown-out Reset Enable (BOR enabled in hardware (SBOREN is ignored))

#pragma config BORV = 190 // Brown-out Reset Voltage (BOR set to 1.9V nominal)

#pragma config nLPBOR = OFF // Low-Power Brown-out Reset (Low-Power Brown-out Reset disabled)

// CONFIG2H

#pragma config WDTEN = OFF // Watchdog Timer Enable bits (WDT disabled in hardware (SWDTEN ignored))

#pragma config WDTPS = 32768 // Watchdog Timer Postscaler (1:32768)

// CONFIG3H

#pragma config CCP2MX = RC1 // CCP2 MUX bit (CCP2 input/output is multiplexed with RC1)

#pragma config PBADEN = OFF // PORTB A/D Enable bit (PORTB<5:0> pins are configured as digital I/O on Reset)

#pragma config T3CMX = RC0 // Timer3 Clock Input MUX bit (T3CKI function is on RC0)

#pragma config SDOMX = RB3 // SDO Output MUX bit (SDO function is on RB3)

#pragma config MCLRE = ON // Master Clear Reset Pin Enable (MCLR pin enabled; RE3 input disabled)

// CONFIG4L

#pragma config STVREN = ON // Stack Full/Underflow Reset (Stack full/underflow will cause Reset)

#pragma config LVP = ON // Single-Supply ICSP Enable bit (Single-Supply ICSP enabled if MCLRE is also 1)

#pragma config ICPRT = OFF // Dedicated In-Circuit Debug/Programming Port Enable (ICPORT disabled)

#pragma config XINST = OFF // Extended Instruction Set Enable bit (Instruction set extension and Indexed Addressing mode disabled)

// CONFIG5L

#pragma config CP0 = OFF // Block 0 Code Protect (Block 0 is not code-protected) #pragma config CP1 = OFF // Block 1 Code Protect (Block 1 is not code-protected) #pragma config CP2 = OFF // Block 2 Code Protect (Block 2 is not code-protected) #pragma config CP3 = OFF // Block 3 Code Protect (Block 3 is not code-protected)

// CONFIG5H

#pragma config CPB = OFF // Boot Block Code Protect (Boot block is not code-protected)

#pragma config CPD = OFF // Data EEPROM Code Protect (Data EEPROM is not code- protected)

// CONFIG6L

#pragma config WRT0 = OFF // Block 0 Write Protect (Block 0 (0800-1FFFh) is not write- protected)

#pragma config WRT1 = OFF // Block 1 Write Protect (Block 1 (2000-3FFFh) is not write- protected)

#pragma config WRT2 = OFF // Block 2 Write Protect (Block 2 (04000-5FFFh) is not write- protected)

#pragma config WRT3 = OFF // Block 3 Write Protect (Block 3 (06000-7FFFh) is not write- protected)

// CONFIG6H

#pragma config WRTC = OFF // Configuration Registers Write Protect (Configuration registers (300000-3000FFh) are not write-protected)

#pragma config WRTB = OFF // Boot Block Write Protect (Boot block (0000-7FFh) is not write-protected)

#pragma config WRTD = OFF // Data EEPROM Write Protect (Data EEPROM is not write- protected)

// CONFIG7L

#pragma config EBTR0 = OFF // Block 0 Table Read Protect (Block 0 is not protected from table reads executed in other blocks)

#pragma config EBTR1 = OFF // Block 1 Table Read Protect (Block 1 is not protected from table reads executed in other blocks)

#pragma config EBTR2 = OFF // Block 2 Table Read Protect (Block 2 is not protected from table reads executed in other blocks)

#pragma config EBTR3 = OFF // Block 3 Table Read Protect (Block 3 is not protected from table reads executed in other blocks)

// CONFIG7H

#pragma config EBTRB = OFF // Boot Block Table Read Protect (Boot block is not protected from table reads executed in other blocks)

# // #pragma config statements should precede project file includes.

// Use project enums instead of #define for ON and OFF.

#include <xc.h>

USART.h :-

// This is a guard condition so that contents of this file are not included

// more than once.

#ifndef USART\_HEADER\_FILE\_H

#define USART\_HEADER\_FILE\_H

#include <xc.h>

void USART\_Init(long);

void USART\_TransmitChar(char);

void USART\_SendString(const char \*);

void MSdelay(unsigned int val);

char USART\_ReceiveChar();

#define F\_CPU 8000000/64 //#define Baud\_value(baud\_rate) (((float)(F\_CPU)/(float)baud\_rate)-1)

#define Baud\_value (((float)(F\_CPU)/(float)baud\_rate)-1)

#endif /\* USART\_HEADER\_FILE\_H \*/

-------------------------------------**-MAIN.C**-------------------------------------------------------------

// Including the necessary Header files //

#include <xc.h>

#include "Configuration\_Bits.h"

#include "USART.h"

// MOTOR Pins

#define M1\_C LATDbits.LATD0 //PIN 0 of PORTD is assigned for Motor1 Clockwise

#define M1\_A LATDbits.LATD1 //PIN 1 of PORTD is assigned for Motor1 Anti Clockwise

#define M2\_C LATDbits.LATD3 //PIN 3 of PORTD is assigned for Motor2 Clockwise

#define M2\_A LATDbits.LATD2 //PIN 2 of PORTD is assigned for Motor2 Anti Clockwise

// LCD Pins

#define RS LATB0 /\*PIN 0 of PORTB is assigned for register select Pin of LCD\*/

#define EN LATB1 /\*PIN 1 of PORTB is assigned for enable Pin of LCD \*/

#define ldata LATB /\*PORTB(PB4-PB7) is assigned for LCD Data (D4-D7) Output\*/

#define LCD\_Port TRISB /\*define macros for PORTB Direction Register\*/

// Declaring the necessary Functions //

void MSdelay\_L(unsigned int ); /\*Generate delay in ms\*/

void LCD\_Init(); /\*Initialize LCD\*/

void LCD\_Command(unsigned char ); /\*Send command to LCD\*/

void LCD\_Char(unsigned char x); /\*Send data to LCD\*/

void LCD\_String(const char \*); /\*Display data string on LCD\*/

void LCD\_String\_xy(char, char , const char \*);

void LCD\_Clear(); // LCD Clear

void mspeedfunc(int); // Motor Speed funtion

void bluetoothint(); // Bluetooth Intialize function

int mspeedval=0;

char BT\_data\_in = '0'; // Input received from BT

int fbtext = 0; //Used to print msg only once in BT

// Initializing the USART //

void USART\_Init(long baud\_rate)

{

float temp;

ANSELC=1;

TRISC6=1; /\*Make Tx pin as output\*/

TRISC7=1; /\*Make Rx pin as input\*/

temp=Baud\_value;

SPBRG=(int)temp; /\* SPBRG=(F\_CPU /(64\*9600))-1\*/

TXSTA=0x20; /\*Transmit Enable(TX) enable\*/

RCSTA=0x90; /\*Receive Enable(RX) enable and serial port enable \*/

}

void USART\_TransmitChar(char out)

{

while(TXIF==0); /\*wait for transmit interrupt flag\*/

TXREG=out;

}

char USART\_ReceiveChar()

{

while(RCIF==0); /\*wait for receive interrupt flag\*/

return(RCREG); /\*receive data is stored in RCREG register and return to main program \*/

}

void USART\_SendString(const char \*out)

{

while(\*out!='\0')

{

USART\_TransmitChar(\*out); // Printing of string at the exact location

out++;

}

}

void MSdelay(unsigned int val)

{

unsigned int i,j; /\*This count Provide delay of 1 ms for 8MHz Frequency \*/

for(i=0;i<=val;i++)

for(j=0;j<81;j++);

}

void main()

{

OSCCON=0x62; /\* use internal oscillator frequency which is set to \* 8MHz \*/

TRISC1 = 0;

TRISC2 = 0; /\* set PORT as output port \*/

PORTCbits.CCP1 = 0;

PORTCbits.CCP2 = 0;

PR2 = 0b00011000;

TRISD = 0; // set PORT as output port

T2CON = 0b00000100;

USART\_Init(9600); /\* initialize USART operation with 9600 baud rate \*/

USART\_SendString(" Ready "); // Send message via Bluetooth

LCD\_Init(); /\*Initialize LCD to 5\*8 matrix in 4-bit mode\*/

LCD\_String\_xy(1,3," WELCOME"); /\*Display string on 1st row, 5th location\*/

LCD\_String\_xy(2,0,"TESLA V2.0"); /\*Display string on 2nd row,1st location\*/

MSdelay(25); //Time granted for Settle before start

while(1)

{

fbtext = 0;

if(RCIF==1) //Check Bluetooth and receive it's value

{

BT\_data\_in = RCREG;

fbtext = 1;

}

if(PORTCbits.RC0 == 1) // If RF signal == True (Entering School Zone)

{

//PWM [Higher value of CCPRxL = Higher Speed or Duty Cycle]

CCPR1L = 0b00001101; //Slow

CCP1CON = 0b00001100;

CCPR2L = 0b00001101; //Slow

CCP2CON = 0b00001100;

MSdelay(1); //Time granted for Settle

if(fbtext == 1)

{

USART\_SendString(" School Zone ");

fbtext = 0;

} // Send message via Bluetooth

LCD\_String\_xy(1,3,"CAUTION!!!"); /\*Display string on 1st row, 5th location\*/

LCD\_String\_xy(2,0,"SchoolZone 20MPH"); /\*Display string on 2nd row,1st location\*/

}

else //if (PORTCbits.RC0 == 0)

{

LCD\_String\_xy(1,3," WELCOME "); /\*Display string on 1st row, 5th location\*/

LCD\_String\_xy(2,0," TESLA V2.0 "); /\*Display string on 2nd row,1st location\*/

CCPR1L = 0b11111111; //100% speed of motor1

CCP1CON = 0b00001100;

CCPR2L = 0b11111111; //100% speed of motor2

CCP2CON = 0b00001100;

MSdelay(1); //Time granted for Settle

}

MSdelay(1); //Time granted for Settle

bluetoothint(); // initialise Bluetooth

}

}

void LCD\_Init()

{

LCD\_Port = 0; /\*PORT as Output Port\*/

MSdelay\_L(15); /\*15ms,16x2 LCD Power on delay\*/

LCD\_Command(0x02); /\*send for initialization of LCD \*for nibble (4-bit) mode \*/

LCD\_Command(0x28); /\*use 2 line and \*initialize 5\*8 matrix in (4-bit mode)\*/

LCD\_Command(0x01); /\*clear display screen\*/

LCD\_Command(0x0c); /\*display on cursor off\*/

LCD\_Command(0x06); /\*increment cursor (shift cursor to right)\*/

}

void LCD\_Command(unsigned char cmd )

{

ldata = (ldata & 0x0f) |(0xF0 & cmd); /\*Send higher nibble of command first to PORT\*/

RS = 0; /\*Command Register is selected i.e.RS=0\*/

EN = 1; /\*High-to-low pulse on Enable pin to latch data\*/

NOP();

EN = 0;

MSdelay\_L(1);

ldata = (ldata & 0x0f) | (cmd<<4); /\*Send lower nibble of command to PORT \*/

EN = 1;

NOP();

EN = 0;

MSdelay\_L(3);

}

void LCD\_Char(unsigned char dat)

{

ldata = (ldata & 0x0f) | (0xF0 & dat); /\*Send higher nibble of data first to PORT\*/

RS = 1; /\*Data Register is selected\*/

EN = 1; /\*High-to-low pulse on Enable pin to latch data\*/

NOP();

EN = 0;

MSdelay\_L(1);

ldata = (ldata & 0x0f) | (dat<<4); /\*Send lower nibble of data to PORT\*/

EN = 1; /\*High-to-low pulse on Enable pin to latch data\*/

NOP();

EN = 0;

MSdelay\_L(3);

}

void LCD\_String(const char \*msg)

{

while((\*msg)!=0)

{

LCD\_Char(\*msg); // Printing of string at the exact location

msg++;

}

}

void LCD\_String\_xy(char row,char pos,const char \*msg)

{

char location=0;

if(row<=1)

{

location=(0x80) | ((pos) & 0x0f); /\*Print message on 1st row and desired location\*/

LCD\_Command(location);

}

else

{

location=(0xC0) | ((pos) & 0x0f); /\*Print message on 2nd row and desired location\*/

LCD\_Command(location);

}

LCD\_String(msg);

}

void LCD\_Clear()

{

LCD\_Command(0x01); /\*clear display screen\*/

}

void MSdelay\_L(unsigned int val)

{

unsigned int i,j;

for(i=0;i<=val;i++)

for(j=0;j<81;j++); /\*This count Provide delay of 1 ms for 8MHz Frequency \*/

}

void bluetoothint()

{

if(BT\_data\_in=='1')

{

mspeedval = 1;

if(fbtext == 1)

{

USART\_SendString(" Back ");//USART\_SendString("Back"); // Send message via Bluetooth

fbtext = 0;

} // Send message via Bluetooth

}

else if(BT\_data\_in=='2')

{

mspeedval = 2;

if(fbtext == 1)

{

USART\_SendString(" Forward ");//USART\_SendString("Forward"); // Send message via Bluetooth

fbtext = 0;

} // Send message via Bluetooth

}

else if(BT\_data\_in=='3')

{

mspeedval = 3;

if(fbtext == 1)

{

USART\_SendString(" Left ");//USART\_SendString("Left"); // Send message via Bluetooth

fbtext = 0;

} // Send message via Bluetooth

}

else if(BT\_data\_in=='4')

{

mspeedval = 4;

if(fbtext == 1)

{

USART\_SendString(" Right ");//USART\_SendString("Right"); // Send message via Bluetooth

fbtext = 0;

} // Send message via Bluetooth

}

else if(BT\_data\_in=='5')

{

mspeedval = 0;

if(fbtext == 1)

{

USART\_SendString(" Stop ");//USART\_SendString("Stop"); // Send message via Bluetooth

fbtext = 0;

} // Send message via Bluetooth

}

else

{

mspeedval = 0;//USART\_SendString("\_Wrong\_"); // Send message via Bluetooth

}

mspeedfunc(mspeedval);

}

void mspeedfunc(mspeedval)

{

if (mspeedval == 0) // Motor Stops

{

M1\_C = 0;

M1\_A = 0;

M2\_C = 0;

M2\_A = 0;

}

else if (mspeedval == 1) // Motor goes Back

{

M1\_C = 1;

M1\_A = 0;

M2\_C = 1;

M2\_A = 0;

}

else if (mspeedval == 2) // Motor goes Forward

{

M1\_C = 0;

M1\_A = 1;

M2\_C = 0;

M2\_A = 1;

}

else if (mspeedval == 3) // Motor goes in Left direction

{

M1\_C = 1;

M1\_A = 0;

M2\_C = 0;

M2\_A = 1;

}

else if (mspeedval == 4) // Motor goes in Right Direction

{

M1\_C = 0;

M1\_A = 1;

M2\_C = 1;

M2\_A = 0;

}

else // Motor Stops

{

M1\_C = 0;

M1\_A = 0;

M2\_C = 0;

M2\_A = 0;

}

}

**Flowchart:**

A close up of a map

Description automatically generated

**Conclusion:**

The implementation of this system into every automobile will solve the problems of traffic congestion, loss life in everyday road accidents and property damage and will also supplementing law enforcement agencies in high speed pursuits.

This project offers user friendly embedded system vehicle which makes a driver much more alert and thus can save hundreds of lives caused every year due to negligence. The presence of GSM module helps keep a track of the vehicle and further provides real time safety and medical aid for the passengers in the case of accidents.

**System’s Picture:**

A close up of a toy

Description automatically generated

A close up of a toy

Description automatically generated

A picture containing ground, floor

Description automatically generated