**AUTOMATIC BRAKING SYSTEM**

by

**Kefil Tonouewa & Ravyn Dickinson**

**ECE 47100 Final Project Report**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*



**Computer Engineering**



Electrical and Computer Engineering Department

Indianapolis, Indiana

ECE 47100 - Spring 2020

**TABLE OF** **CONTENTS**

[LIST OF FIGURES](#_heading=h.1fob9te) 4

[LIST OF TABLES](#_heading=h.3znysh7) 5

INTRODUCTION6

REQUIREMENT OF PROJECT 7

[1.](#_heading=h.3dy6vkm) THE DESIGN DETAILS 8

[1.1](#_heading=h.1t3h5sf) Hardware Design 8

[1.1.1](#_heading=h.4d34og8)  Schematic Diagram with Explanation 8

[1.2](#_heading=h.2s8eyo1) Software 9

[1.2.1 Data Structure Definition](#_heading=h.17dp8vu) 10

[1.2.2](#_heading=h.3rdcrjn)  Block diagram shows the components and interaction 10

[1.2.3](#_heading=h.26in1rg)  Flowchart shows the operation sequence 10

[2.](#_heading=h.147n2zr) IMPLEMENTATION DETAILS 11

2.1 List of Part and tools 11

2.2 Layout diagram 12

2. 3 Software tools 13

3. EXPERIMENTS RESULTS 13

3.1 Describe the results and compare with the requirements 13

4. DISCUSSION 14

5. CONCLUSION 14

REFERENCE 15

[APPENDIX 16](#_heading=h.3o7alnk)

THE SPECIFICATION OF ALL PARTS 16

COMPLETE SOFTWARE CODE 17

**LIST OF FIGURES**

Figure 1. Schematic Diagram

Figure 2. System Block Diagram

Figure 3. Operation Sequence Flowchart

Figure 4. Layout Diagram

**LIST OF TABLES**

Table 1. Potentiometer vs Speed of Motor

**INTRODUCTION**

The purpose of this project is to showcase an understanding of a few of the concepts that were discussed throughout the semester in ECE 47100. In particular, this project requires that each student implement at least two of the peripherals learned this semester namely: analog to digital converter (ADC), watchdog timer, timer, communication modules (serial port interface (SPI), serial communication interface (SCI)), input capture, output capture, and pulse width modulation (PWM). In the following project, four of the listed peripherals were implemented. Among those we can quote ADC, watchdog timer, timer, and a communication module (Liquid Crystal Display). These peripherals were combined together to work on a car braking system. In order to demonstrate the use and full functionality of the chosen peripherals, components such as an ultrasonic sensor, a stepper motor and a potentiometer were used. All of these components were input into a microcontroller, namely PIC18F452. The software for this project was built using C programming and an integrated development environment called MPLAB X. Debugging the software for this project was made possible by the use of PicKit 4 which is an in-circuit debugger. This report will discuss in-depth project requirements, design, implementation, results and finally a conclusion of the project.

**REQUIREMENT OF PROJECT**

The project requirements are for students to familiarize themselves with a board they are not used to and become familiar with it. To do so, the students will be taking advantage of the material that comes with the evaluation board, such as manuals and schematics. The goal is to implement a project that will utilize at least two components for a one-person team and four components for a two person team.

The application will be modeling an automatic braking system using an ultrasonic sensor, a potentiometer and a stepper motor. The user will be able to turn the potentiometer which will directly affect the speed of the stepper motor. This is symbolic of a brake pedal in a vehicle. If the potentiometer is turned all the way up, then the stepper motor will lower speed. If it is midway, the motor will turn at medium speed and if the potentiometer is turned all the way down, the motor will consequently turn at maximum speed. See the table below.

|  |  |
| --- | --- |
| Potentiometer Value | Stepper Motor Speed |
| 10K | Low |
| 5K | Medium |
| 0K | Maximum |

Table 1. Potentiometer vs Speed of Motor

The ultrasonic sensor will be used to sense the distance between the car and an obstacle. If the distance is lower than a certain amount, the car (stepper motor) will come to an automatic complete stop.

# THE DESIGN DETAILS

In terms of the design details of the project, certain components were to be used. As suggested in the project proposal, these components are liquid crystal display, timer, analog to digital converter, stepper motor and a watchdog timer. The design also included the usage of interrupts instead of the polling method for gathering data on many of these components for efficiency purposes. A more detailed explanation of the hardware and software design will be explained in more detail in the following sections.

## Hardware Design

A detailed description of the hardware design is shown in section 1.1.1.

### Schematic Diagram with Explanation

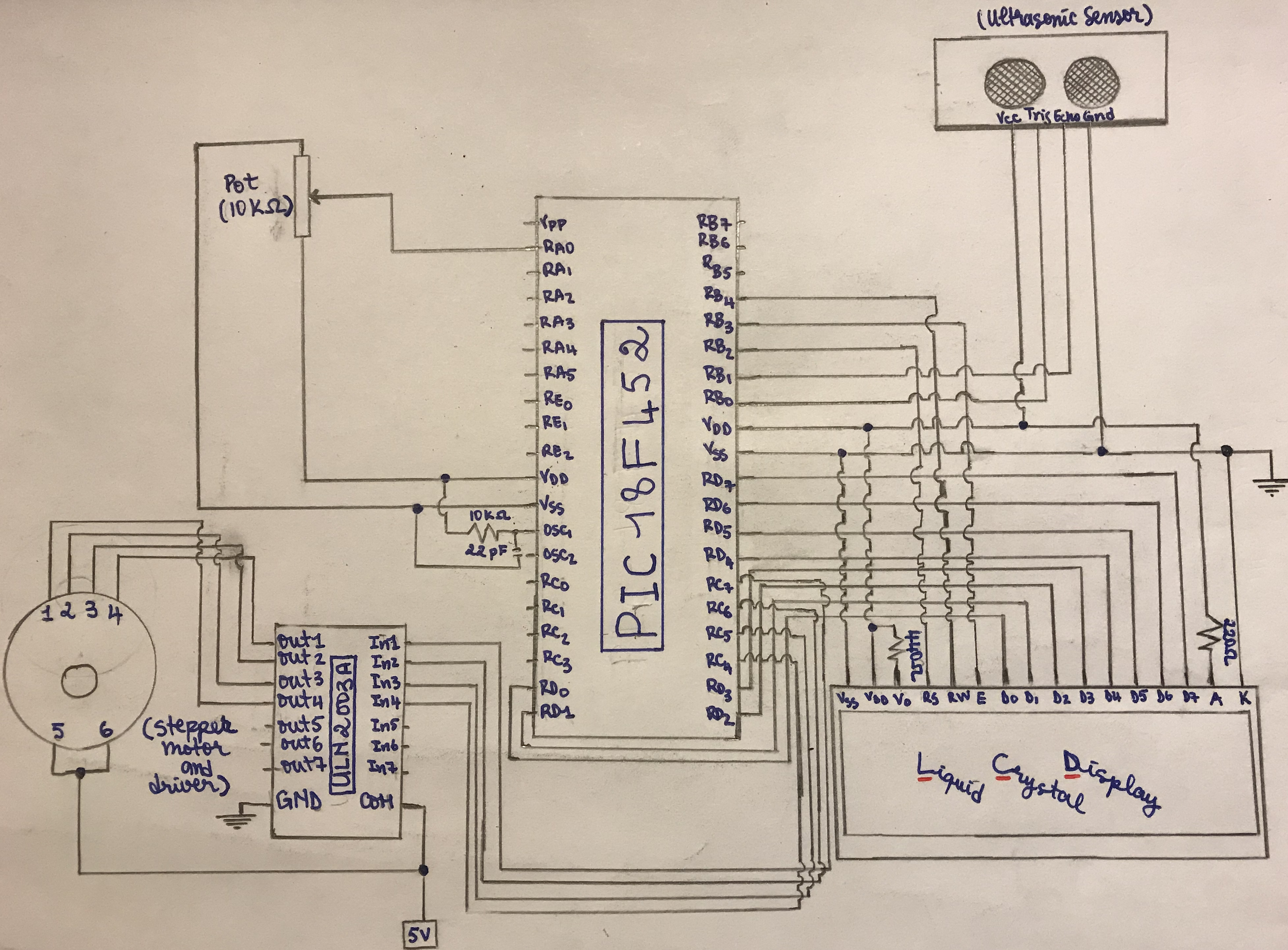


Figure1: Schematic Diagram

As shown on the schematic above, components such as LCD, ultrasonic sensor, potentiometer and stepper motor are being used together with the help of a microcontroller (PIC18F452) to complete the project.

In fact, by visualizing the above schematic, it can be deducted that the 08 data pins of the LCD (D0-D7) are connected to the 08 pins representing PORT D on the microcontroller. The Read-Write pin(RW), the Enable(EN) pin and the Register Select pin(RS) are respectfully connected to PORT B fifth pins, PORT B fourth pin and PORT B third pin on the microcontroller. The anode pin (A) on the LCD is connected to the VDD line on the microcontroller via a 220 ohms resistor (LCD’s LED light control). The cathode (K) on the LCD is connected to the ground of the circuit. The VSS pin on the LCD is connected the ground line on the microcontroller, the VDD on the LCD is connected to the VDD pin on the microcontroller and the Vo pin is connected to the VDD pin on the microcontroller via a 440 ohms resistor.

On the schematic diagram it can also be seen that the Ultrasonic sensor four pins are connected as follows. The GND pin is connected to the ground, the VDD pin is connect to the VDD pin of the microcontroller, the trigger pin is connected to to the the first pin of PORT B on the microcontroller and the echo pin is connected to the second pin of PORT B on the microcontroller.

The potentiometer three pins are connected as follows. The first pin is connected to the VDD pin of the microcontroller, the second pin is connected to PORT A first pin on the microcontroller and the third pin is connected to the ground.

Lastly, according to the schematic, the four pins of the stepper motor are connected to the microcontroller via the last four pins of PORT C.

## Software

The software design for this project was predicated on the header files that existed in pic18f452. Also, after referencing the manual to see how peripherals were configured, all that was left to do was configure said peripherals through code. This included setting bits for which ports would be used for input, output, etc.

### Data Structure Definition

No new data structures were defined for this implementation. Data structures for the values of the ports were already defined in configuration via pic18f452.h.

* + 1. **Block diagram shows the components and interactions**

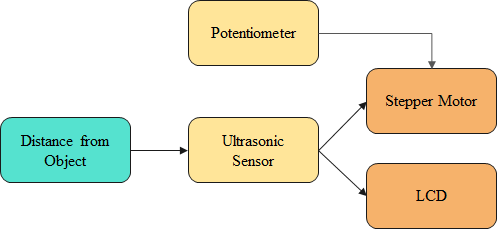


Figure 2. System Block Diagram

The block diagram shown above consists of the main components used to make this project flourish. The speed of the stepper motor is controlled by the ultrasonic sensor and the potentiometer. The distance from the object indirectly feeds into the speed of the stepper motor because it controls the ultrasonic sensor. When the object gets too close, then the stepper motor is halted. However, if the potentiometer is turned to the highest value, then the stepper motor will slow down. This value from the ultrasonic sensor is displayed through the liquid crystal display.

* + 1. **Flowchart shows the operation sequence**

The following figure shows how decisions are being made by the software in this project and what those decisions are affecting in the Hardware.

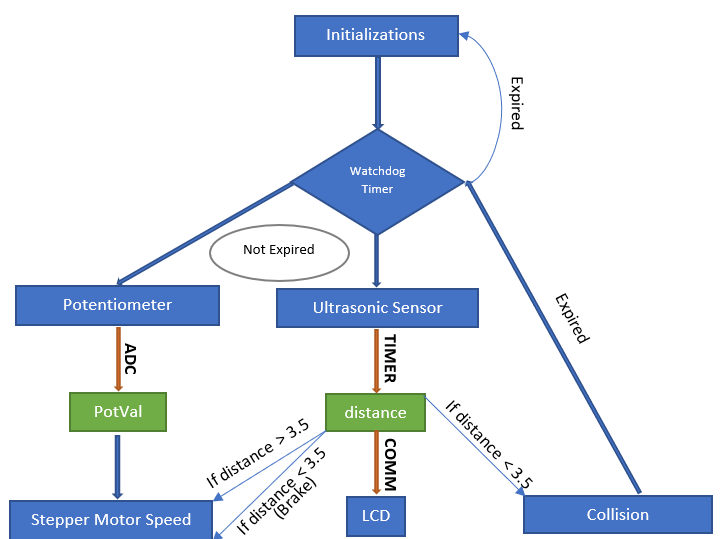


Figure 3. Operation Sequence Flowchart

# IMPLEMENTATION DETAILS

The following subsections will discuss the hardware parts, layout diagram and software tools used for this project.

## List of parts and tools

* PIC18f452
* PicKit 4
* PicKit 3
* Breadboard
* Resistors
* Capacitors
* Crystal Oscillators
* Jumper Wires
* Regular Wires
* Stepper Motor
* Stepper Motor Driver (ULN2003ARG)
* Power Supplies
* Potentiometer
* Ultrasonic Sensor
* Liquid Crystal Display

## Layout Diagram

The layout diagram below shows the connections in the circuit to all external devices used in this project.

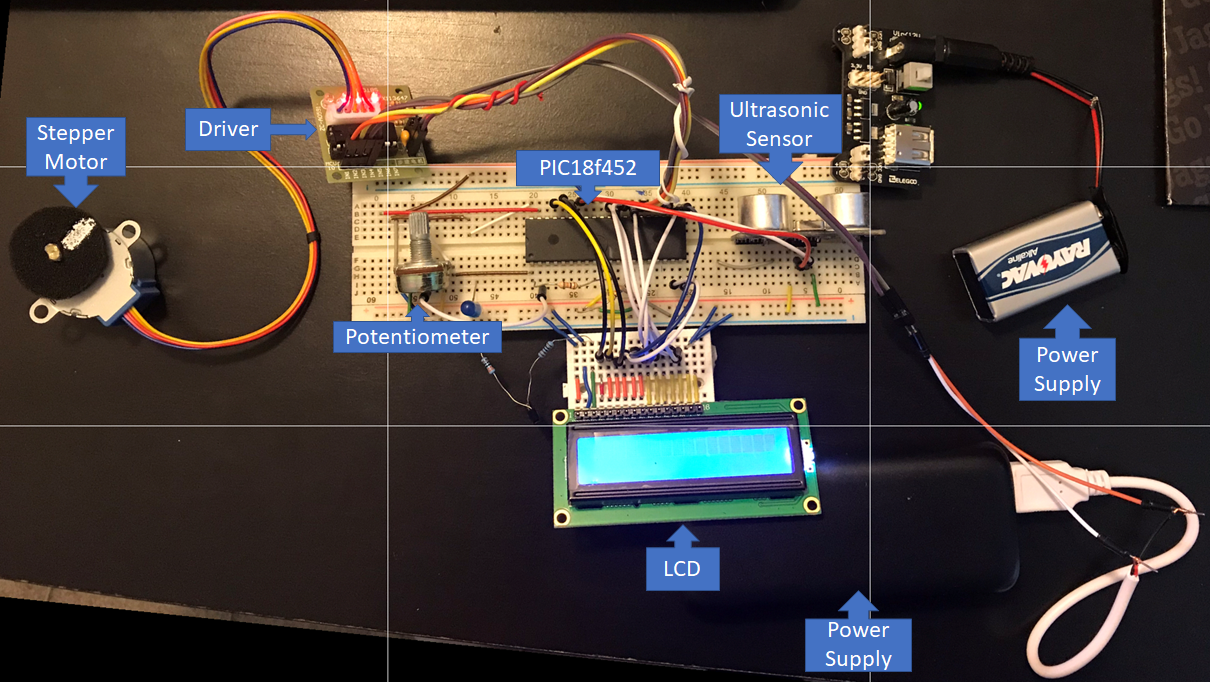


Figure 4. Layout Diagram

LCD: Used to display the distance from the sensor to the object

Stepper Motor: Used to portray the speed

Driver: Used to control stepper motor

Potentiometer: Used to slow down or speed up the stepper motor

Ultrasonic Sensor: Used to sense the distance between itself and object

Power Supplies: Used to provide power

PIC18f452: Used to program the project

## Software tools

* XC8 Compiler version 2.10
* MPLAB IDE version 5.35

# EXPERIMENT RESULTS

The following subsection will explain the various results of the project.

## Describe the results and compare with the requirements

There are a few types of possible results from this project. The following situations will be discussed in this section.

* When an object is greater than 3.5 cm away and the potentiometer is turned to a low value
  + The speed of the motor is high and the LCD displays the distance from the object
* When an object is greater than 3.5 cm away and the potentiometer is turned to a value in the middle
  + The speed of the motor is at medium and the LCD displays the distance from the object
* When an object is greater than 3.5 cm away and the potentiometer is turned to a high value
  + The speed of the motor is low, and the LCD displays the distance from the object
* When an object is less than 3.5 cm away
  + The motor is stopped due to the object being too close to the sensor

# DISCUSSION

This was a decent project that exposed students to different challenges that could be faced in the professional world when they are dealing with programming a microcontroller. During this project numerous challenges were faced. The challenges faced ranged from setting up the hardware to writing software to the right components on board. Those challenges were resolved with the help of some research as well as critical thinking.

# CONCLUSION

In brief, this project has been a good summary of all the materials that were learned in class this semester. Lessons learned include how to correctly wire components together in a circuit, understanding PIC programming and acquiring knowledge on how to configure peripherals. Also, students learned the importance of studying documentation while programming microcontrollers.

**REFERENCES**

Makerguides (2020). How to use a 16x2 character LCD with Arduino. Retrieved from <https://www.makerguides.com/character-lcd-arduino-tutorial/>

Makerguides (2019).28BYJ-48 Stepper Motor with ULN2003 Driver and Arduino Tutorial. Retrieved from <https://www.makerguides.com/28byj-48-stepper-motor-arduino-tutorial/>

Microchip Technology Incorporated. (2006). PIC18F452. Retrieved from <https://www.microchip.com/wwwproducts/en/PIC18F452>

**APPENDIX**

**THE SPECIFICATION OF ALL PARTS**

**PIC18F425**

* Program memory Type: Flash
* Program Memory Size (KB) :32
* CPU Speed (MIPS/DMIPS): 10
* SRAM (B): 1536
* Data EEPROM/HEF (bytes): 256
* Digital Communication Peripherals: 1-UART, 1-SPI, 1-ISC1-MSSP(SPI/I2C)
* Capture/Compare/PWM Peripherals: 2 CCP
* Timers: 1\*8bits, 3\*16bits
* ADC Input: 8 channels, 10bit
* Temperature Range (\*C): -40 to 125
* Operating Voltage Range (V): 2 to 5.5
* Pin Count: 40

**LCD**

* Operating Voltage: 5V
* Controller: Hitachi HD44780 LCD controller
* Screen resolution: 2-lines \* 16 characters
* Character resolution: 5 \* 8 pixels
* Module dimensions: 80 \* 36 \* 12mm
* Viewing are dimensions: 64.5 \* 16.4mm

**Stepper Motor**

* Rated voltage: 5V
* Coil Resistance: 50 Ohms
* Diameter-shaft: 0.197” (5.00 mm)
* Length-shaft and bearing: 0.394” (10 mm)
* Features: Flatted Shaft
* Gear reduction: 1/64
* Steps per revolution: Half step mode: 0.0879DEG, Full step mode: 0.176 DEG
* Steps per revolution: Half step mode: 4096, Full step mode: 2048
* Termination style; Permanent Magnet Gear Motor
* Number of phases: 4

**ULN2003A**

* Output current (single output): 500 mA max
* High sustaining voltage output: 50 V min
* Output clamp diodes
* Inputs compatible with various types of logic
* Package Type-APG: DIP-16pin
* Package Type-AFWG: SOL-16pin

**Ultrasonic Sensor**

* Working Voltage: DC 5 V
* Working Current: 15mA
* Working Frequency: 40Hz
* Max Range: 4m
* Min Range: 2cm
* Measuring Angle: 15 degree
* Trigger Input Signal: 10uS TTL pulse
* Echo Output Signal Input TTL level signal and the range in proportion
* Dimension 45\*20\*15mm

**COMPLETE SOFTWARE CODE**

The following file shows how the project was configured.

*/\**

*\* File: config.h*

*\* Author: Kefil Tonouewa & Ravyn Dickinson*

*\**

*\* Created on March 28, 2020, 9:44 AM*

*\*/*

#ifndef CONFIG\_H

#define CONFIG\_H

#ifdef \_\_cplusplus

**extern** "C" {

#endif

*// CONFIG1H*

#pragma config OSC = RC *// Oscillator Selection bits (RC oscillator)*

#pragma config OSCS = OFF *// Oscillator System Clock Switch Enable bit (Oscillator system clock switch option is disabled (main oscillator is source))*

*// CONFIG2L*

#pragma config PWRT = OFF *// Power-up Timer Enable bit (PWRT disabled)*

#pragma config BOR = OFF *// Brown-out Reset Enable bit (Brown-out Reset disabled)*

#pragma config BORV = 20 *// Brown-out Reset Voltage bits (VBOR set to 2.0V)*

*// CONFIG2H*

#pragma config WDT = OFF *// Watchdog Timer Enable bit (WDT disabled (control is placed on the SWDTEN bit))*

#pragma config WDTPS = 128 *// Watchdog Timer Postscale Select bits (1:128)*

*// CONFIG3H*

#pragma config CCP2MUX = ON *// CCP2 Mux bit (CCP2 input/output is multiplexed with RC1)*

*// CONFIG4L*

#pragma config STVR = ON *// Stack Full/Underflow Reset Enable bit (Stack Full/Underflow will cause RESET)*

#pragma config LVP = ON *// Low Voltage ICSP Enable bit (Low Voltage ICSP enabled)*

*// CONFIG5L*

#pragma config CP0 = OFF *// Code Protection bit (Block 0 (000200-001FFFh) not code protected)*

#pragma config CP1 = OFF *// Code Protection bit (Block 1 (002000-003FFFh) not code protected)*

#pragma config CP2 = OFF *// Code Protection bit (Block 2 (004000-005FFFh) not code protected)*

#pragma config CP3 = OFF *// Code Protection bit (Block 3 (006000-007FFFh) not code protected)*

*// CONFIG5H*

#pragma config CPB = OFF *// Boot Block Code Protection bit (Boot Block (000000-0001FFh) not code protected)*

#pragma config CPD = OFF *// Data EEPROM Code Protection bit (Data EEPROM not code protected)*

*// CONFIG6L*

#pragma config WRT0 = OFF *// Write Protection bit (Block 0 (000200-001FFFh) not write protected)*

#pragma config WRT1 = OFF *// Write Protection bit (Block 1 (002000-003FFFh) not write protected)*

#pragma config WRT2 = OFF *// Write Protection bit (Block 2 (004000-005FFFh) not write protected)*

#pragma config WRT3 = OFF *// Write Protection bit (Block 3 (006000-007FFFh) not write protected)*

*// CONFIG6H*

#pragma config WRTC = OFF *// Configuration Register Write Protection bit (Configuration registers (300000-3000FFh) not write protected)*

#pragma config WRTB = OFF *// Boot Block Write Protection bit (Boot Block (000000-0001FFh) not write protected)*

#pragma config WRTD = OFF *// Data EEPROM Write Protection bit (Data EEPROM not write protected)*

*// CONFIG7L*

#pragma config EBTR0 = OFF *// Table Read Protection bit (Block 0 (000200-001FFFh) not protected from Table Reads executed in other blocks)*

#pragma config EBTR1 = OFF *// Table Read Protection bit (Block 1 (002000-003FFFh) not protected from Table Reads executed in other blocks)*

#pragma config EBTR2 = OFF *// Table Read Protection bit (Block 2 (004000-005FFFh) not protected from Table Reads executed in other blocks)*

#pragma config EBTR3 = OFF *// Table Read Protection bit (Block 3 (006000-007FFFh) not protected from Table Reads executed in other blocks)*

*// CONFIG7H*

#pragma config EBTRB = OFF *// Boot Block Table Read Protection bit (Boot Block (000000-0001FFh) 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.*

#define \_XTAL\_FREQ 1976000

#ifdef \_\_cplusplus

}

#endif

#endif */\* CONFIG\_H \*/*

The following file shows the main part of the project.

*/\**

*\* File: main.c*

*\* Author: King Kefil Tonouewa*

*\**

*\* Created on March 26, 2020, 6:34 PM*

*\*/*

#include <xc.h>

#include <pic18f452.h>

#include <stdint.h>

#include <stdio.h>

#include "config.h"

#define LCD\_DATA LATD *//used to display data onto the lcd screen*

#define LCD\_RS LATBbits.LB2

#define LCD\_RW LATBbits.LB4

#define LCD\_EN LATBbits.LB3

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Global variables\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**static** **uint8\_t** pos\_track = 0;

**static** **uint8\_t** line\_track = 0;

**uint16\_t** PotVal;*//hold result from A/D conversion*

**uint8\_t** Car\_speed;*//hold the car speed(delay)*

**char** Buffer[20];

**uint8\_t** dis;*//*

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Function Prototypes\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**uint8\_t** distance(**void**); *//This function is used to find the distance from the obstacle to the ultrasonic sensor*

**uint8\_t** Speed\_Converter(**void**);*//This function return the delay amount based on the pot value*

**void** **LCD\_clock**(**void**);

**void** **LCD\_command**(**uint8\_t**);

**void** **LCD\_init\_8bits**(**void**);

**uint8\_t** LCD\_set\_pos(**uint8\_t** ,**uint8\_t**);

**void** **LCD\_print**(**uint8\_t**);

**void** **LCD\_print\_string**( **char**\* );

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Interrupt Service Routine\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**void** \_\_interrupt() high\_isr (**void**)

{

INTCONbits.GIE = 0;*//disable further interrupts*

*//we check if the A/D conversion is finished.*

**if**(PIR1bits.ADIF == 1)

{

PotVal = ADRES;*//PotValue get the value of the register that contains the conversion*

Car\_speed = Speed\_Converter();*//convert pot value to delay*

PIR1bits.ADIF =0;*//re-enable interrupt for future interrupts.*

}

INTCONbits.GIE = 1;*//enable interrupts*

}

**void** **Update\_Display**()

{

LCD\_set\_pos(0,0);

LCD\_print\_string("Distance:");

sprintf(Buffer, "%3d",dis );

LCD\_set\_pos(9,0);

LCD\_print\_string(Buffer);

LCD\_set\_pos(13,0);

LCD\_print\_string("cm");

}

*//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

**void** **main**(**void**) {

WDTCONbits.SWDTEN = 1;

**if**(!RCONbits.TO)*//checking if the watchdog timer have been reset*

TRISBbits.RB1 = 1; *//Set PORT B 1st bit as input (ECHO)*

TRISBbits.RB0 = 0; *//set PORT B 2nd bit as an output (TRANSMITER)*

*//TIMER 1 SET UPS*

T1CONbits.T1CKPS0 = 0;

T1CONbits.T1CKPS1 = 0; *//1:1 Prescale value [T1CKPS0=0]*

T1CONbits.TMR1CS = 0; *//Using Internal Clock (Fosc/4)*

*//Stepper motor configuration*

TRISC = 0;*//for stepper motor*

*//Analog to digital conversion (Potentiometer)*

TRISAbits.RA0 = 1; *//Set PORTA as input for the potentiometer*

ADCON1bits.PCFG = 0x0E;*//PORT configuration Control bits. This line set only PORTA 1st bit as an analog pin*

ADCON1bits.ADFM = 1; *//Right justified the result obtained*

ADCON1bits.ADCS2 = 1;

ADCON0bits.ADCS = 0; *//Using the internal clock (FOSC/4) [ADCS2 =1]*

ADCON0bits.CHS = 0; *//Channel 0(AN0)*

ADCON0bits.ADON = 1; *//power up AD Conversion*

PIR1bits.ADIF = 0; *//Conversion in not complete*

PIE1bits.ADIE = 1; *//Enable the A/D interrupt*

*//Interrupts configurations*

RCONbits.IPEN = 1; *//High priority mode*

INTCONbits.GIE =1; *//Enable interrupts*

INTCONbits.PEIE =1; *//Enable peripherals interrupts*

*//Setting the LCD control bits as output*

TRISBbits.RB2 = 0;

TRISBbits.RB4 = 0;

TRISBbits.RB3 = 0;

TRISD = 0x00;*//setting portD as an output for the LCD*

LCD\_RS = 0; *//Register select as 0*

LCD\_RW = 0; *//Read write is 0 because we are only writing for now*

LCD\_EN = 0; *//Set enable to 0*

LATD = 0x00; *//put 0 in PORT D (DATA port)*

LCD\_init\_8bits();*//Initialize the LCD IN 8 bits mode*

\_\_delay\_ms(2000);*//give time for the LCD to warm up*

Update\_Display();

**while**(1)

{

*//while there is no collision*

**while**(distance() > 3.5 )

{

ADCON0bits.GO\_DONE = 1;*//START THE CONVERSION*

LATC = 0b10010000; *//first step*

**for**(**int** j = 0; j<=Car\_speed; j++)

{

\_\_delay\_ms(1);

}

LATC = 0b11000000; *//second step*

**for**(**int** j = 0; j<=Car\_speed; j++)

{

\_\_delay\_ms(1);

}

LATC = 0b01100000; *//third step*

**for**(**int** j = 0; j<=Car\_speed; j++)

{

\_\_delay\_ms(1);

}

LATC = 0b00110000; *//fourth step*

**for**(**int** j = 0; j<=Car\_speed; j++)

{

\_\_delay\_ms(1);

}

CLRWDT();*//clearing watch dog timer since everything is still working properly*

Update\_Display();

}

**while**(1);*//give time to the watch dog timer to stack up*

};

}

**uint8\_t** distance()

{

TMR1 = 0; *//Clear the timer*

*//Trigger the Ultrasonic sensor*

LATBbits.LATB0 = 1;

\_\_delay\_us(13);

LATBbits.LATB0 = 0;

**while**(PORTBbits.RB1 != 1);*//wait for echo to go high*

GIE = 0; *//Disable interrupts the reduce the amount of error in the code*

TMR1ON = 1; *//Turn the timer on*

**while**(PORTBbits.RB1 == 1);*//wait for echo to go back low*

TMR1ON = 0; *//Turn the timer off*

dis = (TMR1/30.5);

GIE = 1;*//re-enable the interrupts*

**return** dis;

}

**uint8\_t** Speed\_Converter()

{

**if**(PotVal <=100)

{

Car\_speed = 6;

}

**else** **if**(PotVal <=205)

{

Car\_speed = 15;

}

**else** **if**(PotVal <=310)

{

Car\_speed = 25;

}

**else** **if**(PotVal <=410)

{

Car\_speed = 40;

}

**else** **if**(PotVal <=510)

{

Car\_speed = 50;

}

**else** **if**(PotVal <=610)

{

Car\_speed = 60;

}

**else** **if**(PotVal <=700)

{

Car\_speed = 70;

}

**else** **if**(PotVal <=800)

{

Car\_speed = 80;

}

**else** **if**(PotVal <=900)

{

Car\_speed = 90;

}

**else** **if**(PotVal <=1000)

{

Car\_speed = 100;

}

**else**

{

Car\_speed =130;

}

**return** Car\_speed;

}

**void** **LCD\_clock**(**void**){

LCD\_EN = 1;

\_\_delay\_ms(1);

LCD\_EN = 0;

\_\_delay\_ms(1);

}

**void** **LCD\_command**(**uint8\_t** command){

LCD\_RS = 0;

LCD\_RW = 0;

LCD\_DATA = command;

LCD\_clock();

\_\_delay\_ms(10);

}

**void** **LCD\_init\_8bits**(**void**){

LCD\_RS = 0;

LCD\_RW = 0;

\_\_delay\_ms(50);

*//Making the LCD an 8bits interface*

LCD\_command(0x30);

LCD\_command(0x30);

LCD\_command(0x30);

*//set function*

*//8bit mode*

*//1 line display*

*//font to 5x8*

LCD\_command(0x34);

*//display off*

*//cursor off*

*//cursor no blink*

LCD\_command(0x08);

*//clear the display and set to home*

LCD\_command(0x01);

*//increment*

*//no shift*

LCD\_command(0x06);

*//display on*

*//cursor off*

*//cursor not blinking*

LCD\_command(0x0C);

}

**uint8\_t** LCD\_set\_pos(**uint8\_t** x,**uint8\_t** y){

**if**(y == 1){

x += 0x40;

}

LCD\_command(0x80 | x);

**return** x;

}

**void** **LCD\_print**(**uint8\_t** x){

**if**(pos\_track > 15){

**if**(line\_track > 0){

LCD\_set\_pos(0,0);

line\_track = 0;

}**else**{

line\_track++;

LCD\_set\_pos(0,1);

}

pos\_track = 0;

}

LCD\_RS = 1;

LCD\_RW = 0;

LCD\_DATA = x;

LCD\_clock();

pos\_track++;

}

**void** **LCD\_print\_string**(**char** \*str){

**uint16\_t** i = 0;

**do**{

LCD\_print(str[i]);

i++;

}**while**(str[i] != '\0');

}