REPORT ON

PROXIMITY METER

Abstract

- In this project our objective was to develop a proximity sensor which can measure the distance to a given object accurately and display it on an LCD screen and to indicate when a given distance typing on a keypad to the instrument is reached.
- The accuracy achieved by the instrument is a key component and it is designed to achieve a high accuracy.
- The proximity sensor was tested using many different obstacles such as those with rough, black and smooth surfaces and those which are large is size and vice versa. The test results are carefully observed plotted and measures have been taken to standardize those test results.
- 16F887 microcontroller PIC is used as the main component where it is programmed with the MPLAB IDE. Apart from the microcontroller, an Ultrasonic sensor and an IR sensor is used to detect the target object. Furthermore, a laser beam is used to make it easy for the sensors to align the target object. A buzzer indicates that the proximity meter has arrived at the specified distance.
- MPLAB is used to program the device as the IDE which contains C language. Further Proteus 8.0 professional was used to simulate the circuits and the enclosure was initially designed using SolidWorks.
- The structure of the proximity sensor is a rectangular box with an LED screen, keypad, ultrasonic sensor, IR sensor, laser, a reset button and power indicate LED seen from outside the enclosure.
- There are two functionalities of the proximity meter. One is to measure the distance to a given object. The other is sensing a given distance. Both sensors will take n consecutive values and those readings with a large deviation to the previous value is omitted. The mean of the accurate readings is taken for processing.
- Many problems were encountered during the course of this project. From the very beginning many ideas were put forward and a lot of reasoning was done to reject the rest and chose one. Furthermore, there were serious issues regarding the sensors and we had set backs with some components getting burnt. Technical difficulties were there in soldering the PCB as well. With the help of the instructors we were able to overcome many of the issues and it became a real learning curve for all of us.

Table of Content

1.0 Introduction

- Microcontrollers which are like small computers with a memory and a processor, play a primary role in many autonomous systems. In this project a simple proximity meter is designed using a microcontroller to identify a distance to an obstacle within a given range.
- The proximity meter should comply with following specification
    1. Consists of a proximity sensor, a digital display, input buttons/keypad and a warning light/alarm.
    2. The meter should be able to sense up to a maximum distance of 2m.
    3. The distance must be displayed on screen.
    4. The user should be able to input any desired distance and the device must be able to indicate when the required distance is reached using the warning light/alarm.
- We identified the suitable PIC microcontroller and the necessary IR and Ultrasonic sensors for the project.

Micro controllers

This is the most popular controlling system due its low cost, simple power requirements and the ability to be programmed using simple software a simple hardware interface. Once programmed the microcontroller acts like a mini-computer. There is a wide range of micro controllers serving different interfacing capabilities.

Ultrasonic sensors

Sensor based vision system is used in simple autonomous systems and ultrasonic sensors are used to calculate distance to an object. The sensor sends a signal and calculate the time spent to receive the signal reflected from the object.

IR sensors

An infrared sensor is an electronic device, which emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detects the motion. These types of sensors measures only infrared radiation

Laser

The laser beam is directly pointed at the object to which we have to measure the distance, the two sensors can focus on the object without less effort. The two sensors can align the target object without effort and in less time with the coordinates received from the laser beam.

- When the device is switched on it asks the user to select either of the following functionalities.
    1. Measure the distance to a given target object
    2. Indicate a distance when a user inputs a distance value.

1. When a target object is identified by the proximity sensor, it will initialize the Ultrasonic sensor, n number of consecutive readings will be taken comparing each of the reading with the previous one. If there is a considerable deviation with the previous reading, that particular reading is avoided. It keeps taking readings until the n number of close readings are completed. Afterwards the average of those n readings will be taken.

   If the proximity meter encounters a reading whose value is between 20-150 cm, the IR sensor is initialized and the above process is continued until n readings are take whose average is calculated then. The mean values from both the sensors are taken and the mean of those two values are displayed as output.
2. When the second functionality is activated it asks the user to enter distance within 2m. As soon as the user enters a distance, first it activates the Ultrasonic sensor first and the previously mentioned process is continued. The meter keeps getting the readings. As the readings are from 20cm to 150cm the IR sensor is activated and the previous steps are continues. When the user input distance matches the readings from the meter, a buzzer is activated letting the user that the object has reached the input distance.

2.0 Methodology

2.1 List of electronic components used

- 1 16F887 microcontroller PIC
- 1 HC-SR04 Ultrasonic Ranging Module
- 1 Sharp GP2Y0A02YK0F IR Range Sensor
- 1 16*2 Liquid Crystal Display (HD44780 Driver IC)
- 1 LM 7805 voltage regulator
- 1 20MHz Crystal Oscillator
- 1 4*4 membrane key pad (16 keys)
- 4 10k resistors
- 1 100k resistor
- 1 150k resistor
- 2 0.1uF capacitors
- 4 470uF capacitors
- 2 22pF capacitors
- 1 RGB LED
- 2 green LEDs
- 2 push button switches
- 1 switch

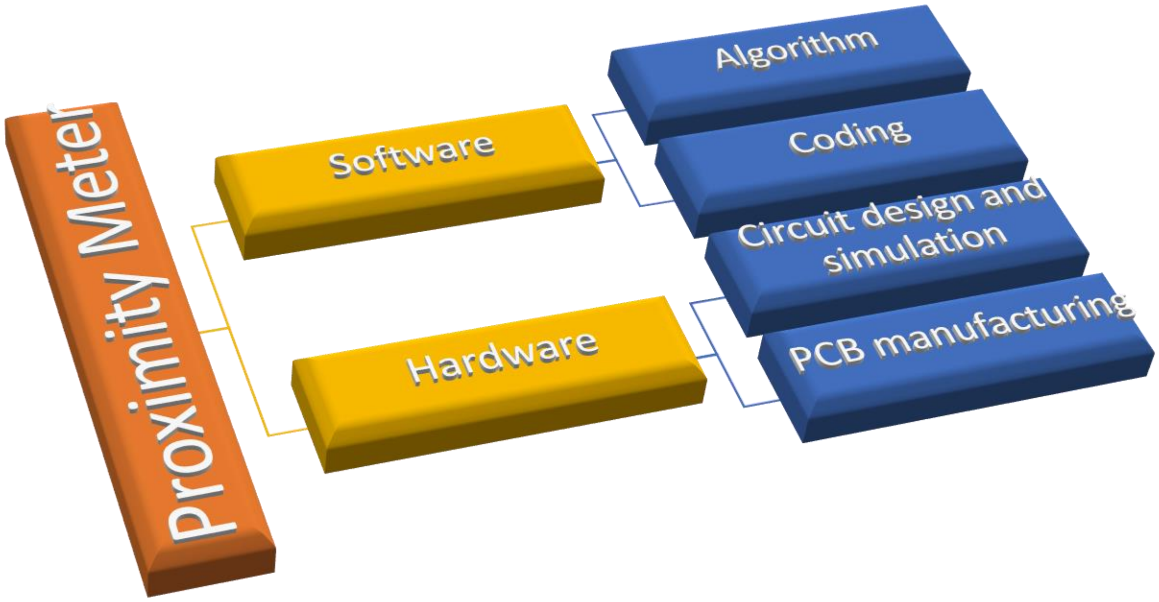
Other components

- Female headers
- Male headers
- Multimeter
- Soldering iron, solder wires
- Sucker

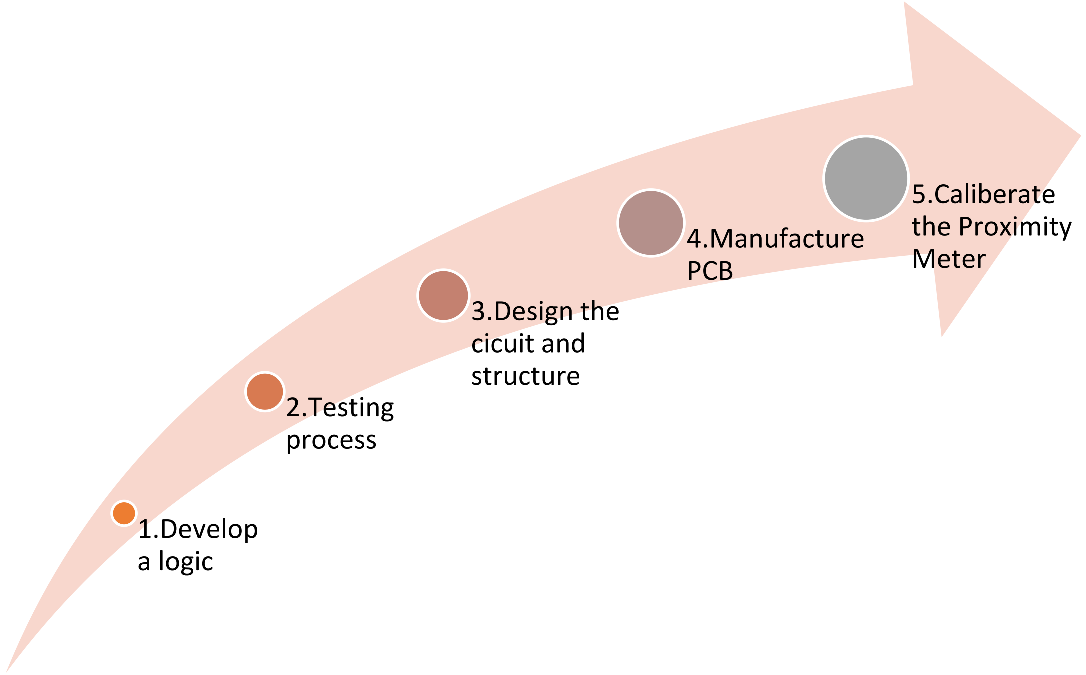
Other items

- Copper plates
- Thinner
- Ferric Chloride

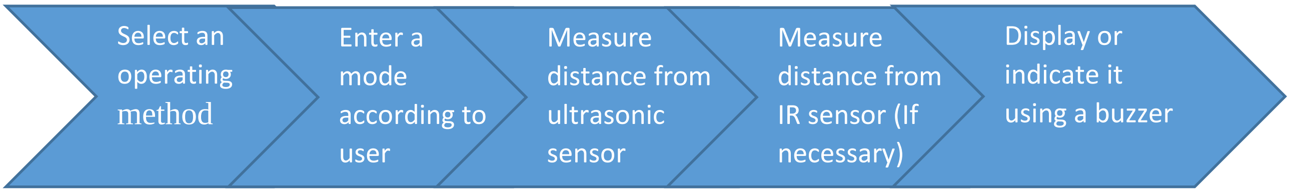
2.2 Approach



2.3 Steps of the Project

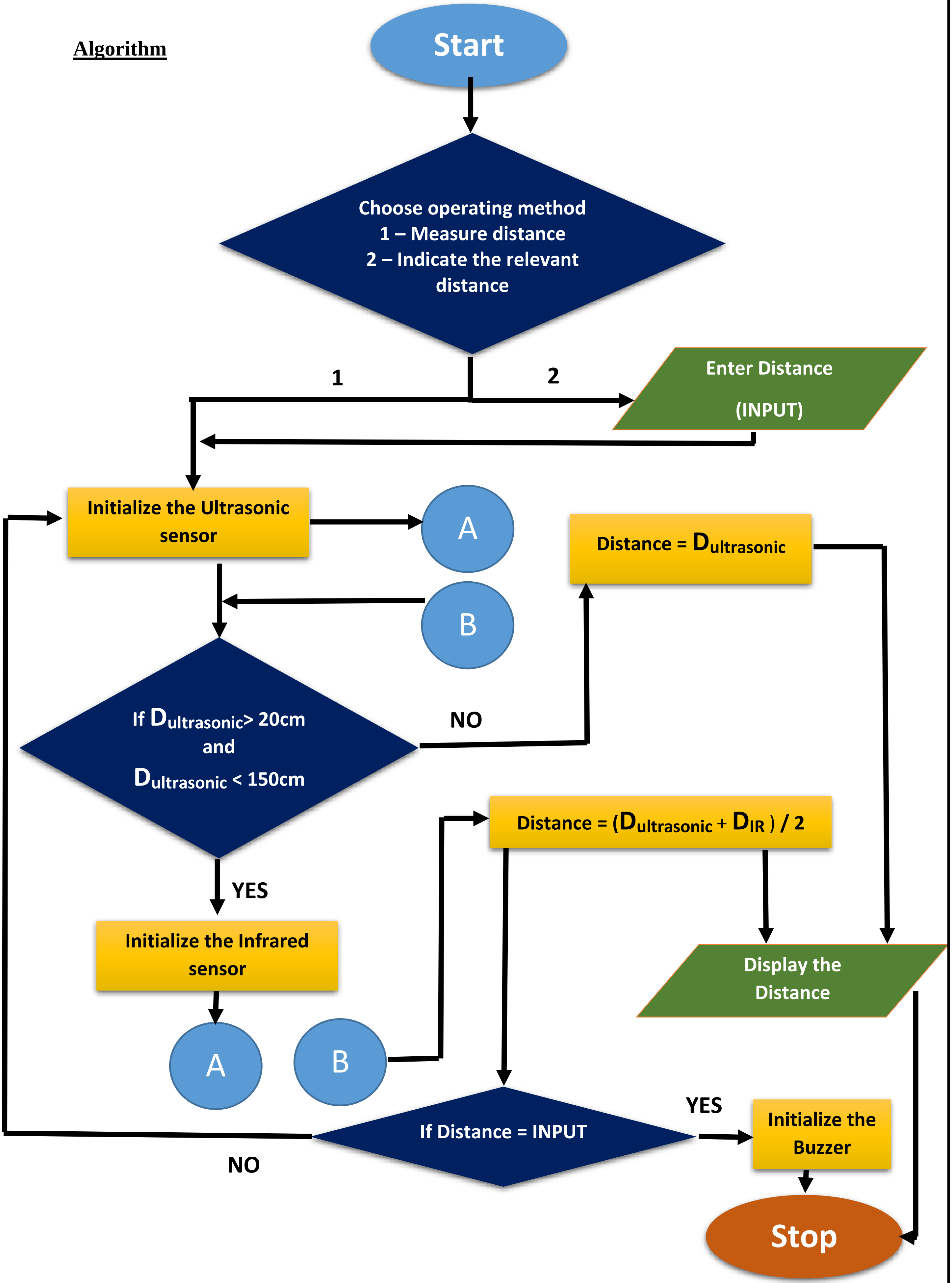

1.Develop a logic
2.Testing process
3.Design the cicuit and structure
4.Manufacture PCB
5.Caliberate the Proximity Meter

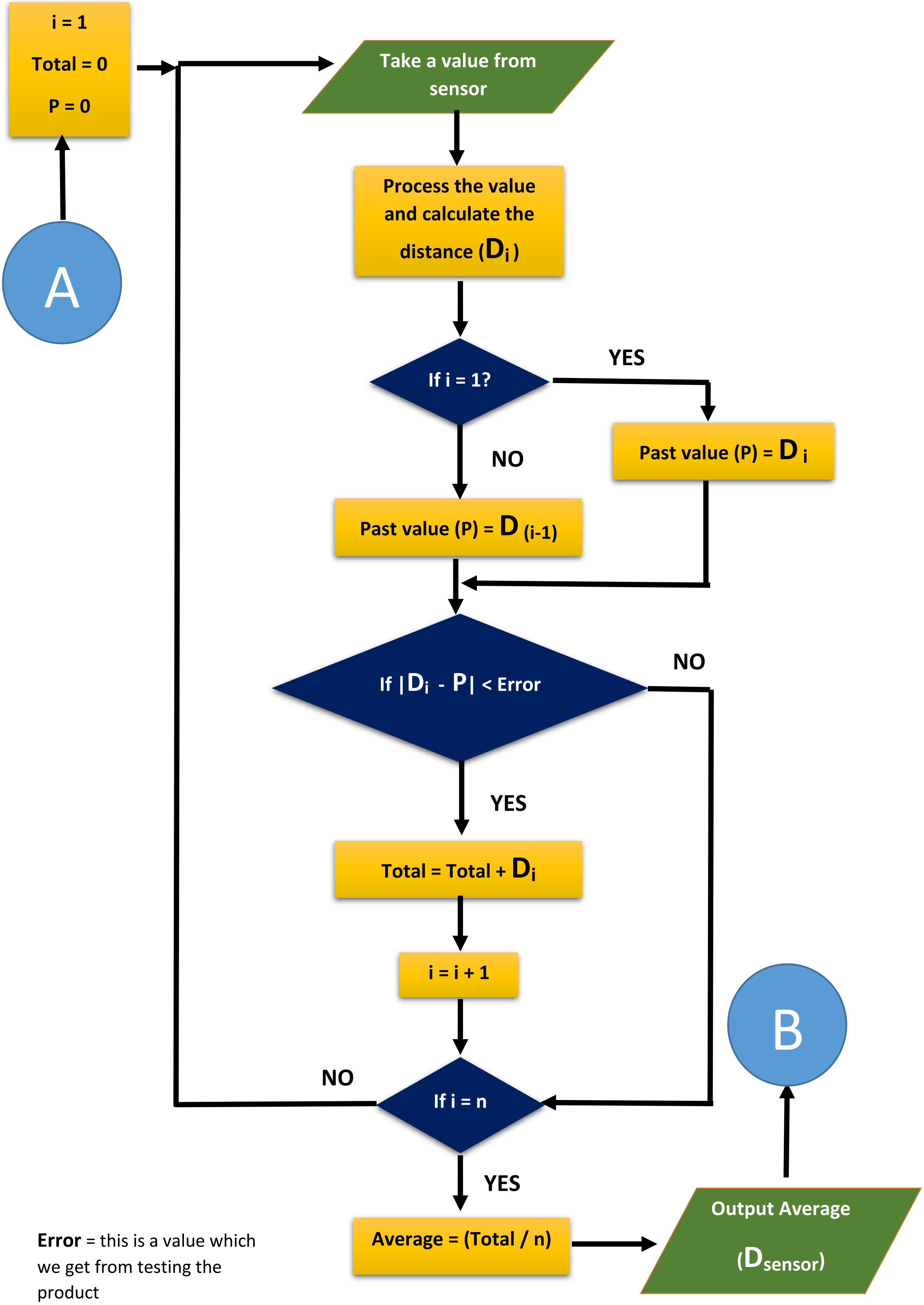
2.4 Logic and main parts of the code


Select an operating method
Enter a mode according to user
Measure distance from ultrasonic sensor
Measure distance from IR sensor (If necessary)
Display or indicate it using a buzzer

Main Logic

Algorithm


Start
Choose operating method
1 – Measure distance
2 – Indicate the relevant distance
1
2
Enter Distance
(INPUT)
Initialize the Ultrasonic sensor
A
B
Distance = $D_{ultrasonic}$
If $D_{ultrasonic}$> 20cm and $D_{ultrasonic}$ < 150cm
NO
YES
Distance = ($D_{ultrasonic}$ + D_{IR}) / 2
Initialize the Infrared sensor
Display the Distance
A
B
If Distance = INPUT
YES
NO
Initialize the Buzzer
Stop



Error = this is a value which we get from testing the product

Results

All the following test runs were done in different environments to emphasize the working condition of the prototype product.

PROCESS –

1. First we measure the distance with the interval of 10cm.
2. Compare it with the actual distance.
3. We took the critical point of each object according to the size and surface.
4. Repeat above steps for 4 different size objects and 3 different type surfaces.
5. At last we came up with the accuracy interval. (Using tested data)

According to the size -

Object number	Object 1	Object 2	Object 3	Object 4
Dimensions of the object	5cmx5cm	8cmx8cm	10cmx10cm	12cmx12cm

According to the surface -

1. Smooth
2. Rough
3. Black
4. White

} Dimensions – 10cmx10cm

Discussion

- Since this is our 1st project involving micro-controllers we had to learn everything from the very beginning: simulating circuits using Proteus, MPLAB with XC8 compiler and hardware designing with SolidWorks.

  Initially the idea was to use MikroC which uses C to program the PIC. Even though programming with C is easy because of the availability of many libraries and just calling a function is sufficient, we went for MPLAB which uses C with XC8 compiler because it gives us the opportunity to analyse the codes in depth. With MPLAB we could learn to manipulate registers ourselves.
- While discussions with the team members, we came up with many proposals with regard to the design of the project of which some had to be rejected due to various reasons.
- First the idea was to use only the Ultrasonic sensors but as we developed the project it was realized that its accuracy was low. Even though the sensor had the capability of sensing far away objects, due to the ultrasonic wave diverging as the distance increases, the chances of sensing nearby objects which is not the accurate goal increase. Therefore, due to the inaccuracy of the readings, the idea had to be neglected.
- Next, we thought of using IR sensors. It was a very good solution for our previous difficulty since the IR beam diverges very little. As the datasheet of the IR sensor says the effective sensing range of an IR sensor is from 20cm to 150 cm. Since we were assigned to sense the distance of objects up to 2 meters, the idea of using only the IR sensor had to be abandoned as well.
- Then we came up with the idea of using both the sensors simultaneously depending on the object distance. The inputs are taken from both the sensors and they are compared, and the decisions are taken by a complex algorithm. This proves to be the best of the suggestions which provided solutions for the problems encountered in the previous two suggestions. Therefore, we continued with the idea of using both Ultrasonic and IR sensors.
- To enhance the performance of the device, the suggestion to use a laser beam was put forward. After some discussion that proved to be a good suggestion as the laser

beam is directly pointed at the object to which we have to measure the distance, the two sensors can focus on the object with less effort. The two sensors can align the target object without effort and in less time.

- Since we were instructed to create a marketable product, we developed a prototype of an enclosure using SolidWorks which suits the proximity sensor. The price of recreating the prototype using a suitable material was money consuming and the cost of the product seemed very high. As a result we were forced to abandon the idea and use a pre-produced enclosure with a considerably low cost.

Problems faced in developing the project

- The PIC micro-controller got burnt once.
- The power provided to the circuit was not sufficient for the IR sensor and the LCD display to function properly.
- While manufacturing the PCB there were a couple of short circuits as two tracks were touching each other. Furthermore a few damaged tracks were identified with disconnections.
- The copper pads of the PCB were not large enough therefore while drilling, the pad disappeared completely.
- Several problems were faced during soldering. Free flowing of lead around the pads did not happen and instead lead turned into balls as soon as the heat was transferred.
- The PCB got burned in the process of soldering.

Identified causes for the problems

- The pins of the PIC were identified incorrectly therefore power was supplied to the wrong pin as a result the PIC got burnt.

- The current drawn by the LCD display and the IR sensor (40mA) was too large therefore one 7805 power supply was insufficient, therefore we were forced to use two 7805s one for the LCD display and the IR sensor and the other for the rest of the circuit.

- The damages in the tracks were identified due the shortcomings of the screen printing process and due to insufficient removal of copper after dissolving them in Ferric Chloride, short circuits were visible.

- The copper pads were not sufficiently large due to selecting a wrong footprint while making the layout using Proteus. The correct foot print had to be chosen to rectify the issue.

- The selected soldering lead contained the wrong lead carbon ratio of 60/37. As a result, when heat was transferred the lead melted into a ball rather than free flowing across the copper pads. As a solution soldering lead with lead carbon ratio of 60/30 was selected.

- The power consumed by the soldering iron (40W) was too high therefore it got heated to a very high temperature. The temperature could not be controlled in the soldering iron we selected. Therefore, as a remedy we used the soldering iron with a manually controllable temperature at the Workshop.

Acknowledgement

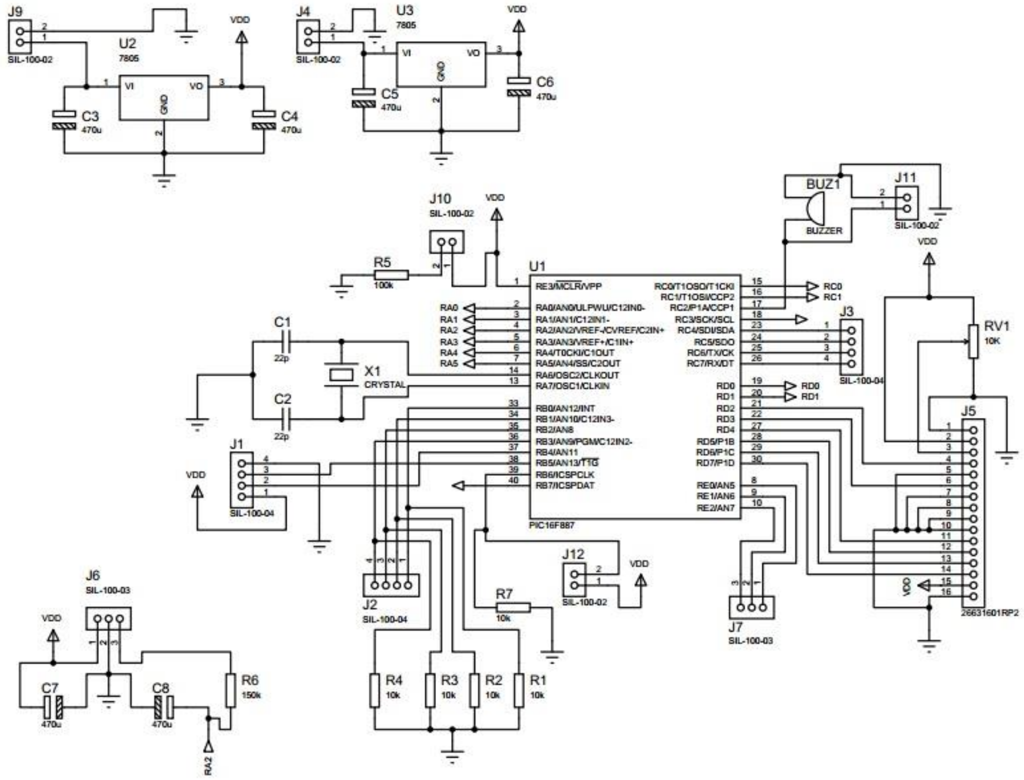
- The completion of this project Proximity Meter was not an easy task since we were new to all the concepts of microcontroller programming. We had to learn everything from scratch and since this is our first project involving microcontrollers, we encountered many problems. There were many individuals behind us from the beginning up to the end of the project.
- First and the foremost our gratitude must go for our supervisor Mr.Yasith Amarasinghe who was giving us advice and forcing us to do self-study on the subject of microcontroller programming. His advice during the mid-evaluation was of great help to identify and rectify certain issues and to do improvements on the project.
- Also our sincere gratitude should be extended to Mr.Janith Kalpa for conducting a pre project lecture on microcontrollers. At a time, we were confused with a big project in front of us, the lecture was of immense help. Furthermore, we are thankful to all the lecturers and instructors who were ready to share their knowledge with us.
- We are thankful to all the lab personals for allowing us to utilize the workshop and other instruments and helping us out with technical problems.
- Our thanks are extended to all the members of the ENTC family who shared their ideas and resources with us during the process. Without them the completion of this project wouldn’t have been a reality. They were always with us at the times of failure.
- We developed team spirit in working together as a team everybody contributing with their maximum potentials and capabilities. It was a great experience.

Reference

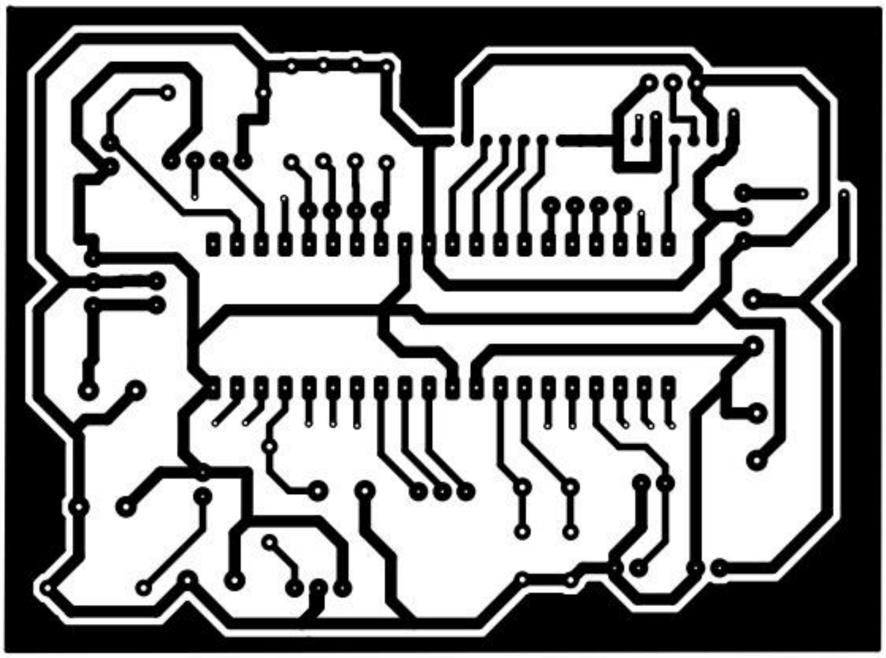
1. Competition Rules and guidelines- sent by Mr.Yasith Amarasinghe
2. http://www.microchip.com/wwwproducts/en/PIC16F887
3. http://learn.mikroe.com/ebooks/piccprogramming/chapter/pic16f887-basic-features/
4. http://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
5. https://www.pololu.com/product/136
6. https://www.newbiehack.com/MicrocontrollerTutorial.aspx
7. http://www.theengineeringprojects.com/2013/03/a-complete-tutorial-on-how-to-use-proteus-isis-ares.html
8. https://electrosome.com/category/tutorials/pic-microcontroller/mikroc/

Appendices

7.1 Schematic diagram -1

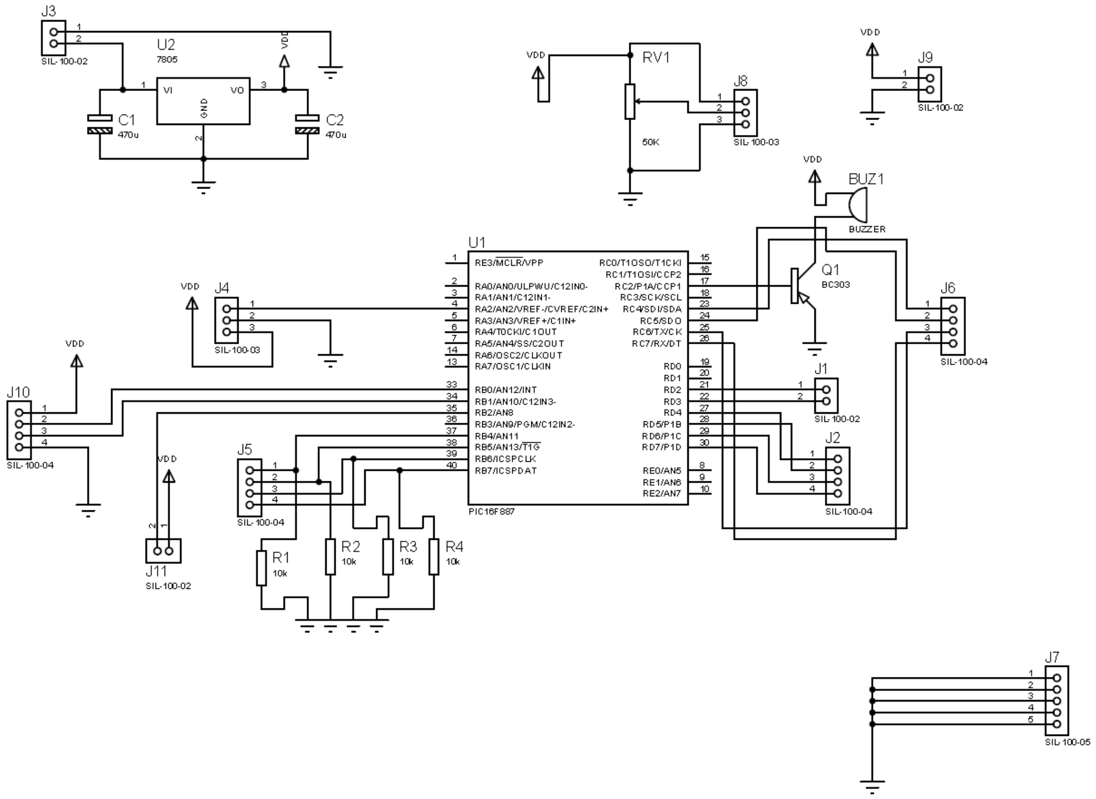


7.2 PCB Layout-1

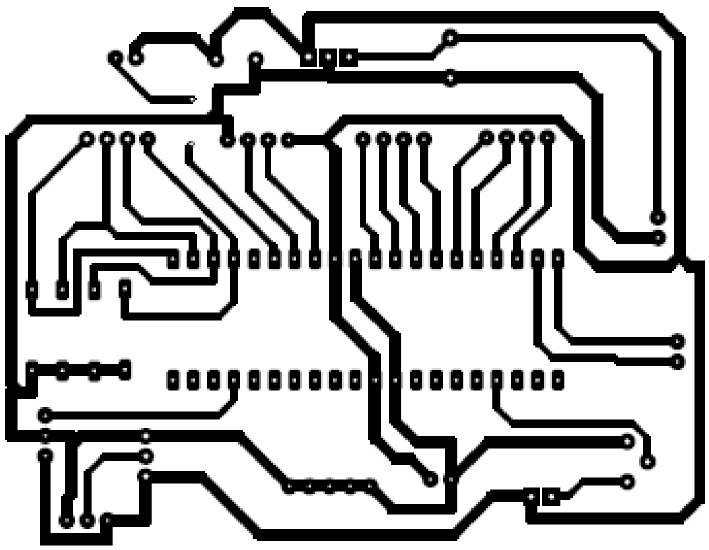


Our first PCB design doesn't work properly. Therefore we have to come up with another PCB layout

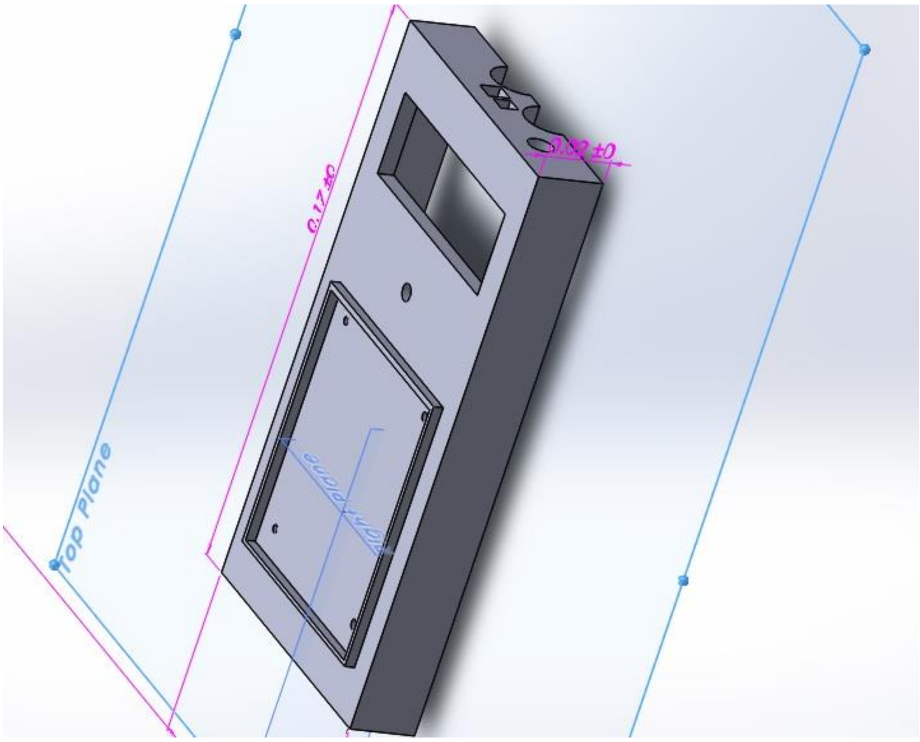
7.3 Schematic diagram -2

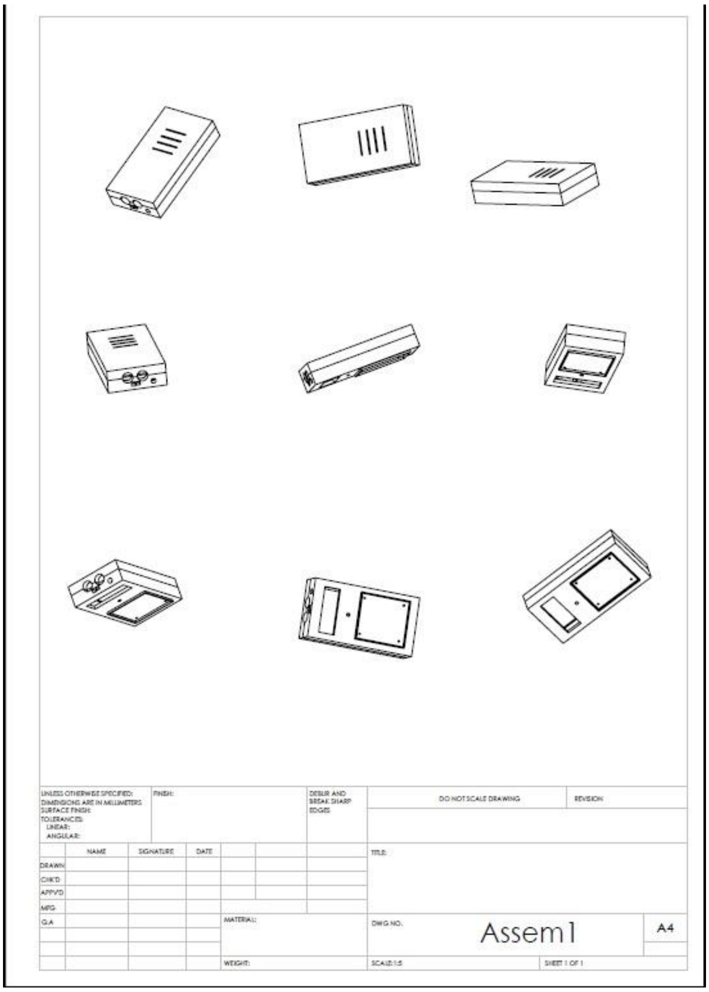


7.4 PCB Layout -2



7.4 Enclosure Prototype





7.5 Enclosure selected

7.6 Codes

```
#define _XTAL_FREQ 8000000

#include <xc.h>
#include "lcd.h";
#include "keypad.h";
#include <stdio.h>
#include <stdlib.h>
#include "U_SONIC.h";

// CONFIG1
#pragma config FOSC = HS         // Oscillator Selection bits (HS oscillator: High-speed crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = ON        // RE3/MCLR pin function select bit (RE3/MCLR pin function is MCLR)
#pragma config CP = OFF          // Code Protection bit (Program memory code protection is disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
#pragma config IESO = OFF         // Internal External Switchover bit (Internal/External Switchover mode is disabled)
#pragma config FCMEN = OFF       // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock Monitor is disabled)
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital I/O, HV on MCLR must be used for programming)

// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to 4.0V)
```

```
#pragma config WRT = OFF         // Flash Program Memory Self Write Enable bits (Write
protection off)



int a,r;
char str[4],str1[4];
int menu=1;
int mode;
char key;

void buzz(int fre)
{
   if (fre<2)RC2=1;
   else RC2=0;

}

void interrupt ISR_HANDLER()
{
   INTCONbits.GIE=0;
   if(INTCONbits.RBIF == 1)
   {

      if(PORTBbits.RB2==1)T1CONbits.TMR1ON=1;
      if(PORTBbits.RB2==0)
      {
         T1CONbits.TMR1ON=0;
         a=(TMR1L|(TMR1H<<8))/58.82;
      }



      //if (PORTBbits.RB0==1)
```

```
        //{
          //menu=1;
        //}
    }
    INTCONbits.RBIF=0;
    INTCONbits.GIE=1;
    }



void main(void) {

  Keypad_Init();
  Lcd_Config();
  Lcd_Init();
  Lcd_Clear();
  Lcd_Set_Cursor(1,1);
  Lcd_Write_String("WELCOME");
  __delay_ms(2000);
  Lcd_Clear();
  INTCONbits.GIE=1;
  INTCONbits.PEIE=1;
  INTCONbits.RBIF=0;
  INTCONbits.RBIE=1;
  //IOCBbits.IOCB0=1;
  //PORTBbits.RB0=1;
  Usonic_Init();
  //TRISBbits.TRISB0=1;
  TRISCbits.TRISC2=0;
  PORTCbits.RC2=0;

  while(1)
```

```
{   int back=0;
    if (menu){
      Lcd_Clear();
      Lcd_Set_Cursor(1,1);
      Lcd_Write_String("Please Select a");
      Lcd_Set_Cursor(2,1);
      Lcd_Write_String("Operating Mode");
      __delay_ms(3000);
      Lcd_Clear();
      Lcd_Set_Cursor(1,1);
      Lcd_Write_String("1.Measure Distance");
      Lcd_Set_Cursor(2,1);
      Lcd_Write_String("2.Proximity Alert");
    }
    while (menu)
    {
     mode=readKeyboard()-48;
     if (mode==1 | mode==2)
     {
        menu=0;
        break;
     }
    }
    if (mode==1){


      Lcd_Clear();
      Lcd_Set_Cursor(1,1);
      Lcd_Write_String("1.Measure Distance");
      __delay_ms(2000);
      Lcd_Clear();
      while(!menu)
      {
```

```
                if (back==67){

                menu=1;

    }
      Ping();
      sprintf(str,"%dcm",a);
      Lcd_Clear();
      Lcd_Set_Cursor(1,1);
      Lcd_Write_String(str);
          back=readKeyboard();
__delay_ms(100);
   }
}
else if (mode==2){

   r=ReadInt();
while(!menu)
{
   if (back==67){

menu=1;

   }

Lcd_Clear();
Lcd_Set_Cursor(1,1);
Lcd_Write_String("Entered=");
Lcd_Set_Cursor(1,9);
sprintf(str1,"%dcm",r);
Lcd_Write_String(str1);
Ping();
```

```
        sprintf(str,"%dcm",a);
        Lcd_Set_Cursor(2,1);
        Lcd_Write_String(str);
        buzz(abs(r-a));

        back=readKeyboard();
        __delay_ms(100);

        }

    }





  }
  }
```

```
int u;
void Usonic_Init(){
    TRISBbits.TRISB2=1;
    TRISBbits.TRISB1=0;
    T1CON=0x10;
    IOCBbits.IOCB2=1;
}
void Ping(){
      TMR1H=0x00;
      TMR1L=0x00;

      PORTBbits.RB1=0;
      __delay_us(2);
      PORTBbits.RB1=1;
      __delay_us(10);
      PORTBbits.RB1=0;

      __delay_ms(50);
}
#include <math.h>
#include<stdlib.h>

float tol=0.1;
int avg=20;
void Sharp_Init()
{
  ANSEL=0x00;
  ANSELH=0x00;
  TRISA=0xFF;



  TRISAbits.TRISA2=1;
  ANSELbits.ANS2=1;

  ADCON1bits.ADFM=1;
  ADCON1bits.VCFG0=0;
  ADCON0bits.ADCS=0b101;

  ADCON0bits.CHS=2;
  ADCON0bits.ADON=1;
}

int analogRead(int channel)
{   int raw;
    float volt;

    ADCON0bits.CHS=channel;
    ADCON0bits.GO=1;
    while(ADCON0bits.GO==1)
      {
         raw=(ADRESH<<8)|ADRESL;
      }
    ADCON0bits.GO=0;

    return raw;                             //(61.573*pow(volt, -1.1068));
}

int Process(int channel)
{
    int p=0;
    int sum=0;
    int previousV=0;

    for (int i=0; i<avg; i++){
        __delay_ms(10);
        int foo=analogRead(channel);

        //if (abs(previousV-foo)<=50){

           // previousV=foo;
            sum=sum+foo;
            p++;

        }
   // }




    int accurateV=sum/p;
    float volt=(accurateV*5.0)/1023.0;
    return (int)(61.573*pow(volt, -1.1068));
}
#include <math.h>
void Keypad_Init()
{
  TRISB=0b11110000;
  PORTB=0x00;
  ANSEL=0;
  ANSELH=0;
  TRISC=0;
  PORTC=0;
}

char findKey(unsigned short a, unsigned short b)
{
```

```
if(b == 0)
{
  if(a == 3)
   return 'A';
  else if(a == 2)
   return '3';
  else if(a == 1)
   return '2';
  else if(a == 0)
   return '1';
}
else if(b == 1)
{
  if(a == 3)
   return 'B';
  else if(a == 2)
   return '6';
  else if(a == 1)
   return '5';
  else if(a == 0)
   return '4';
}
else if(b == 2)
{
  if(a == 3)
   return 'C';
  else if(a == 2)
   return '9';
  else if(a == 1)
   return '8';
  else if(a == 0)
   return '7';
```

```
 }
 else if(b == 3)
 {
   if(a == 3)
    return 'D';
   else if(a == 2)
    return '.';
   else if(a == 1)
    return '0';
   else if(a == 0)
    return '#';
 }
 return ' ';
}

char readKeyboard()
{
 unsigned int i = 0;
 for(i=0;i<4;i++)
 {
  if(i == 0)
  {PORTC=0;
  PORTCbits.RC4=1;}
  else if(i == 1)
  {PORTC=0;
  PORTCbits.RC5=1;}
  else if(i == 2)
  {PORTC=0;
  PORTCbits.RC6=1;}
  else if(i == 3)
  {
    PORTC=0;
  PORTCbits.RC7=1;}

  if(RB4)
    return findKey(0,i);
  if(RB5)
   return findKey(1,i);
  if(RB6)
   return findKey(2,i);
  if(RB7)
   return findKey(3,i);
 }
 return ' ';
}
int ReadInt()
{   int key=0;
    int i=0;
    int dist=0;
    Lcd_Clear();
    int buff[4]={0,0,0,0};
    Lcd_Set_Cursor(1,1);
    Lcd_Write_String("Input Distance:");
    __delay_ms(300);
    while(1)
    {
            key=readKeyboard();

            if (key==65) break;
            else if (key==66 & (i+8)>7)
            {
                i=i-1;
                Lcd_Set_Cursor(2,i+8);
                Lcd_Write_Char(' ');
            buff[i]=0;
            __delay_ms(300);
         }
         else if (key>=48 & key<=57)
         {
            buff[i]=key-48;
            Lcd_Set_Cursor(2,i+8);
            Lcd_Write_Char(key);
            i++;
            __delay_ms(300);
         }
   }
   for(int j=0;j<i;j++)
   {
      dist=dist+pow(10,i-j-1)*buff[j];
   }
   return dist;
}
//LCD Functions Developed by electroSome,edited by HP
void Lcd_Config()
{
   TRISD = 0x00;
   #define RS RD2
   #define EN RD3
   #define D4 RD4
   #define D5 RD5
   #define D6 RD6
   #define D7 RD7
}


void Lcd_Port(char a)
```

```
{
        if(a & 1)
                D4 = 1;
        else
                D4 = 0;

        if(a & 2)
                D5 = 1;
        else
                D5 = 0;

        if(a & 4)
                D6 = 1;
        else
                D6 = 0;

        if(a & 8)
                D7 = 1;
        else
                D7 = 0;
}
void Lcd_Cmd(char a)
{
        RS = 0;             // => RS = 0
        Lcd_Port(a);
        EN  = 1;             // => E = 1
     __delay_ms(4);
     EN  = 0;             // => E = 0
}

void Lcd_Clear()
{
```

```
            Lcd_Cmd(0);
            Lcd_Cmd(1);
}

void Lcd_Set_Cursor(char a, char b)
{
            char temp,z,y;
            if(a == 1)
            {
              temp = 0x80 + b - 1;
                        z = temp>>4;
                        y = temp & 0x0F;
                        Lcd_Cmd(z);
                        Lcd_Cmd(y);
            }
            else if(a == 2)
            {
                        temp = 0xC0 + b - 1;
                        z = temp>>4;
                        y = temp & 0x0F;
                        Lcd_Cmd(z);
                        Lcd_Cmd(y);
            }
}

void Lcd_Init()
{
  Lcd_Port(0x00);
   __delay_ms(20);
  Lcd_Cmd(0x03);
            __delay_ms(5);
  Lcd_Cmd(0x03);
```

```
            __delay_ms(11);
  Lcd_Cmd(0x03);
  /////////////////////////////////////////////////////
  Lcd_Cmd(0x02);
  Lcd_Cmd(0x02);
  Lcd_Cmd(0x08);
  Lcd_Cmd(0x00);
  Lcd_Cmd(0x0C);
  Lcd_Cmd(0x00);
  Lcd_Cmd(0x06);
}

void Lcd_Write_Char(char a)
{
   char temp,y;
   temp = a&0x0F;
   y = a&0xF0;
   RS = 1;             // => RS = 1
   Lcd_Port(y>>4);             //Data transfer
   EN = 1;
   __delay_us(40);
   EN = 0;
   Lcd_Port(temp);
   EN = 1;
   __delay_us(40);
   EN = 0;
}

void Lcd_Write_String(char *a)
{
            int i;
            for(i=0;a[i]!='\0';i++)
		Lcd_Write_Char(a[i]);
	}

	//void Lcd_Shift_Right()
	//{
	//	Lcd_Cmd(0x01);
	//	Lcd_Cmd(0x0C);
	//}
	//
	//void Lcd_Shift_Left()
	//{
	//	Lcd_Cmd(0x01);
	//	Lcd_Cmd(0x08);
	//}
```