Department of Electronic and Telecommunication Engineering

University of Moratuwa

EN 1093 – Laboratory Practice I

WEIGHT SCALE WITH LOAD CELL

K.R.Y Nagasinghe 180411K

K. Navaneethan 180415C

T.M.S Nayanajith 180417J

N. Neminathan 180418M

This is submitted as a partial fulfillment for the module
EN1093: Laboratory Practice I
Department of Electronic and Telecommunication Engineering
University of Moratuwa

06th March 2020

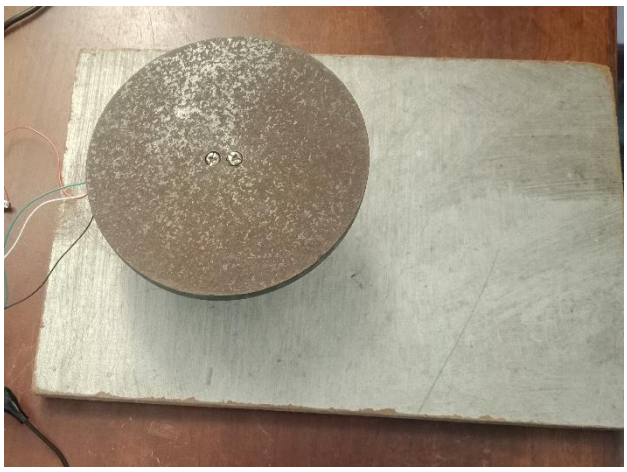
CONTENTS

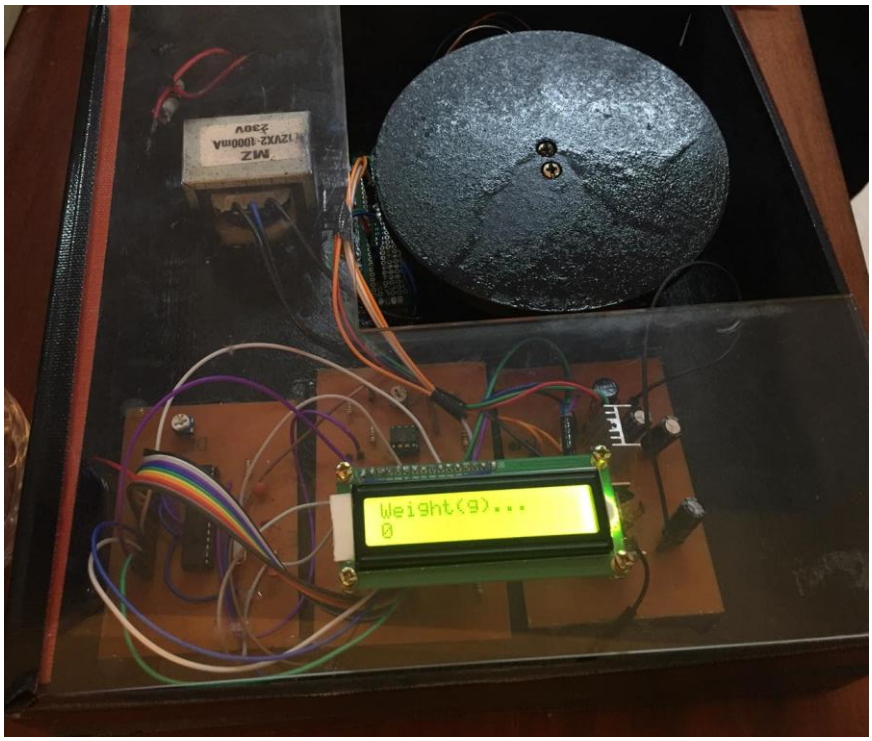
ABSTRACT

The objective of our project is to design a weight scale using loadcells, which can measure weight in 0-5 kg range with 10g accuracy in measuring.

Weight scale has been designed using the AtMega328p microcontroller and we have used Atmel Studio platform to program it by C language. We have used Orcad to design our circuits and to design the layouts of the PCB and also, we have used Multisim to simulate those circuits such as Power supply circuit, Amplifying circuit and Microcontroller circuit.

Product enclosure at mid evaluation





FINAL PRODUCT

INTRODUCTION

In our project, a weight scale has been designed using a load cell. The weight scale should satisfy the following specifications.

1. Specifications of the weight scale

- Weight scale should be designed with analog circuit components and a microcontroller.
- It should display the weight of the load.
- Weight scale must be AC powered.
- Weight scale should be able to measure weights in 0-5 kg range.
- Should be able to measure weights with 10g of accuracy.

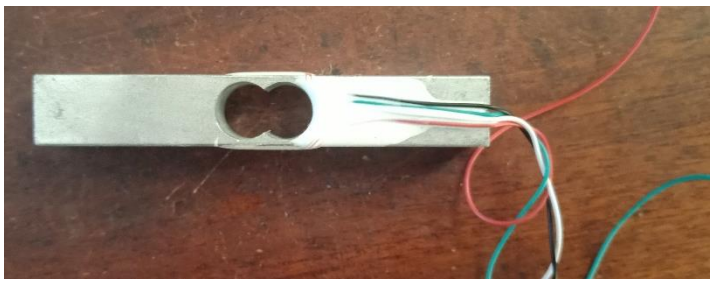
2. Scientific Literature

First we studied about load cell and its functionalities.

We have used Orcad to design circuits in module EN 1070 and Multisim to simulate circuits for a project in module EN 1013. So, we have used Orcad and Multisim to design circuits in this project too. But we don't have any experience in microcontroller and its programming. So, we studied about microcontrollers and C programming for ADC part for this project.

Load Cell

Load cell is a transducer which converts force (Eg: strain, pressure) into electric signals. Here we use loadcell to convert the force applied by the weight into the form of electric voltage. We have used strain gauge load cell here which performs according to a specific circuit called 'Wheatstone Bridge'. First, we tried 5 kg load cell. Then we replaced it with 10 kg load cell, as we are in a need to measure weights in range 0-5 kg including 5kg too. We could have used two 5 kg load cells instead of 10kg load cell. We didn't use that method as circuits become much more complicated.



Microcontroller

Micro controller is also called as minicomputer, which can perform a task in a circuit according to the way we program it. Programming a micro controller can be done with software. Microcontroller is programmed using Atmel Studio. Programming was done by mounting the microcontroller on a Atmega32 development board. An USBASP was used as the programmer.

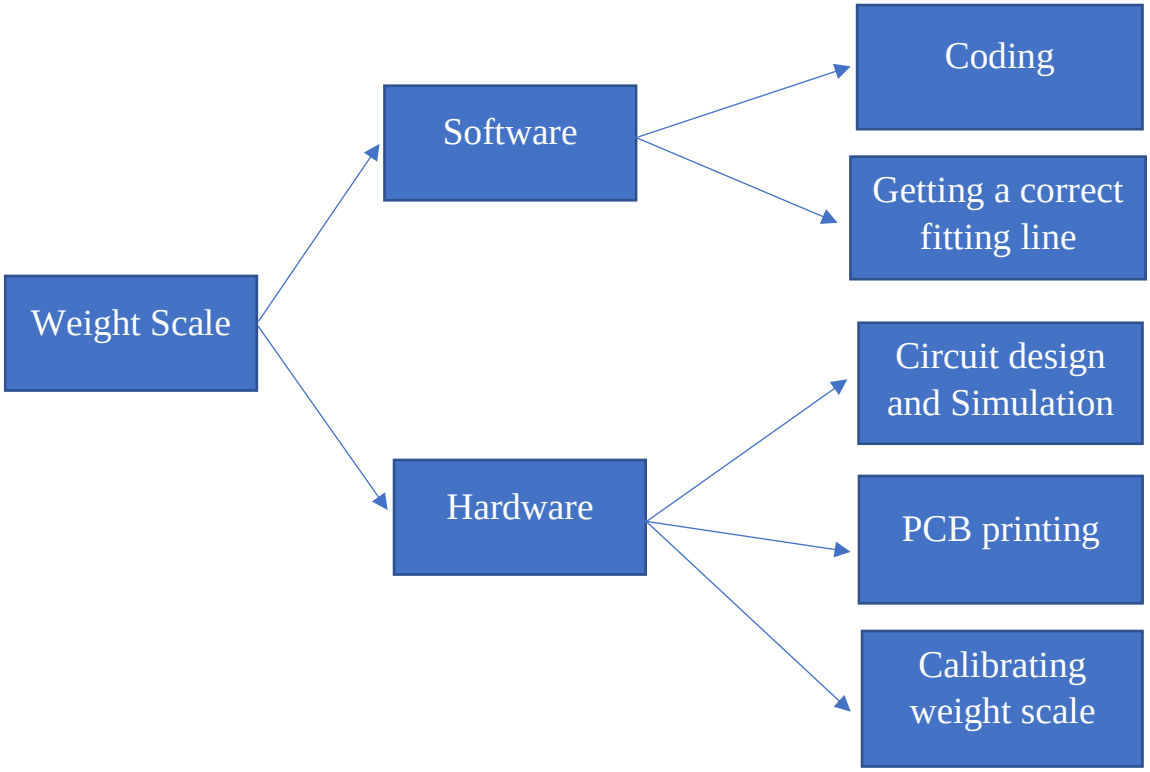
METHOD

List of components used

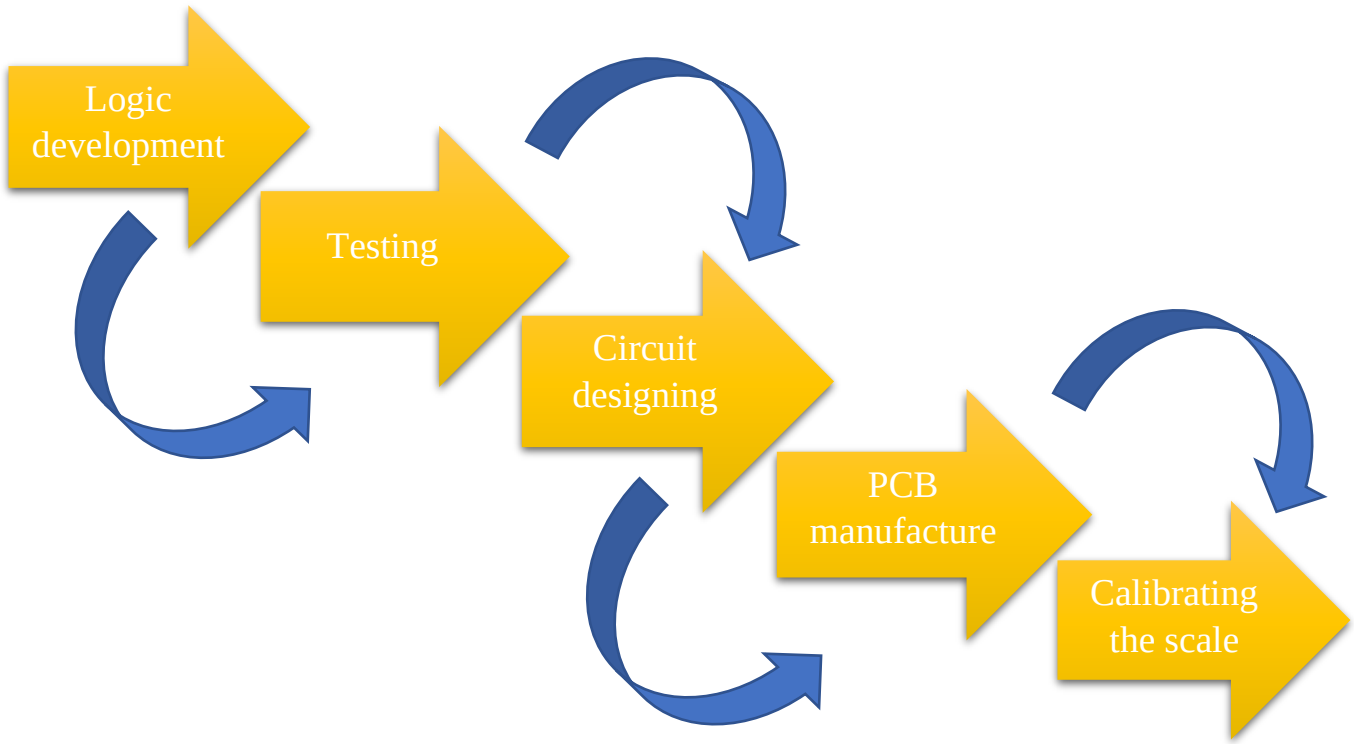
- Power Supply Circuit
    - LM 7810, LM 7910, LM 7805 regulators with heat sink.
    - 3 - 470µF, 3 - 0.01µF capacitors.
    - 3 – 1kΩ resistors.
    - Center tapped transformer 230 V
    - Rectifier Bridge 2W10
- Amplifier with adder Circuit
    - 3 - UA741 Op-Amplifiers for Instrumental amplifier and UA741 for adder circuit.
    - 5 – 5kΩ, 4 – 1kΩ, 1MΩ resistors.
    - 5kΩ variable resistor for gain and 1kΩ variable resistor for offset purpose.
- Microcontroller circuit- ADC part
    - AT-Mega 328p microcontroller.
    - 16 MHz oscillator.
    - 10 kΩ variable resistor.
    - 1kΩ resistor.
    - 2 - 22pF, 100nF capacitors.
- Other components and equipment we used
    - Load cell - 10 kg
    - LCD display(16 pin)

APROACH

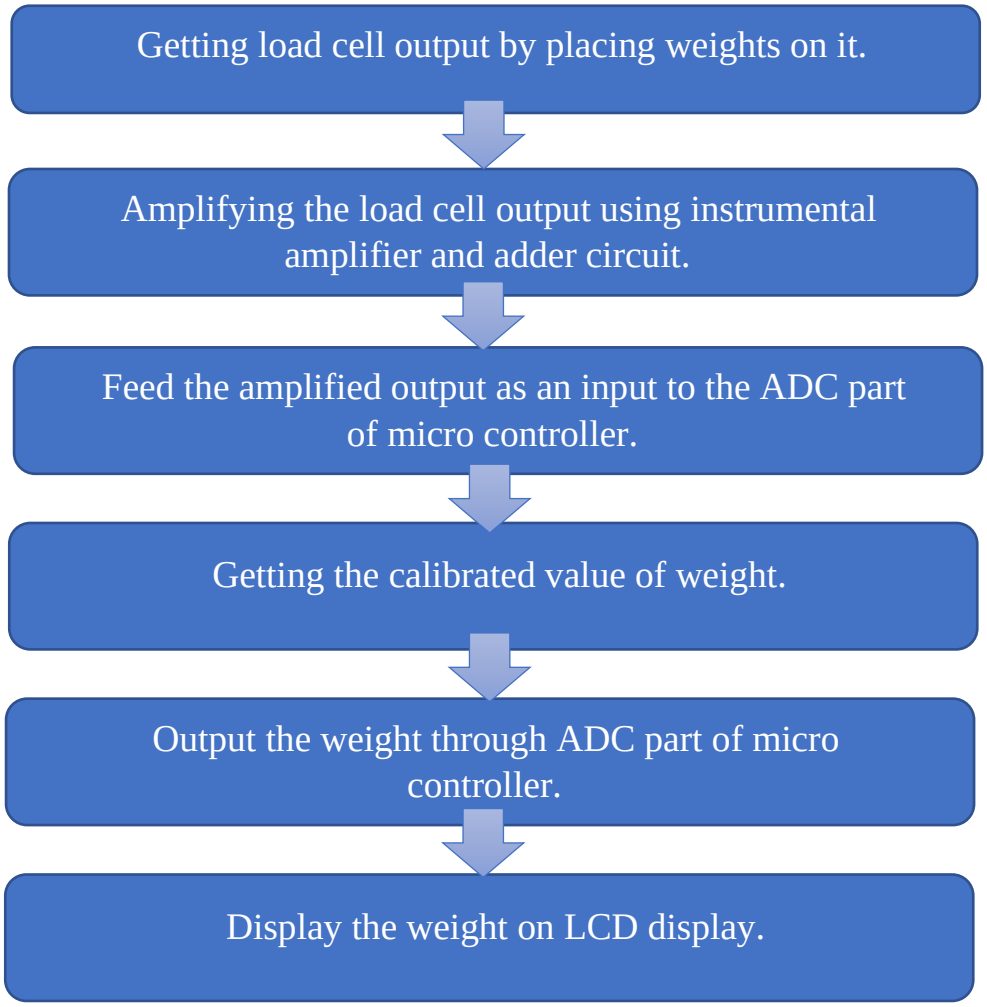
Basically, there are 2 sectors to be considered in any project. Such as software and hardware.



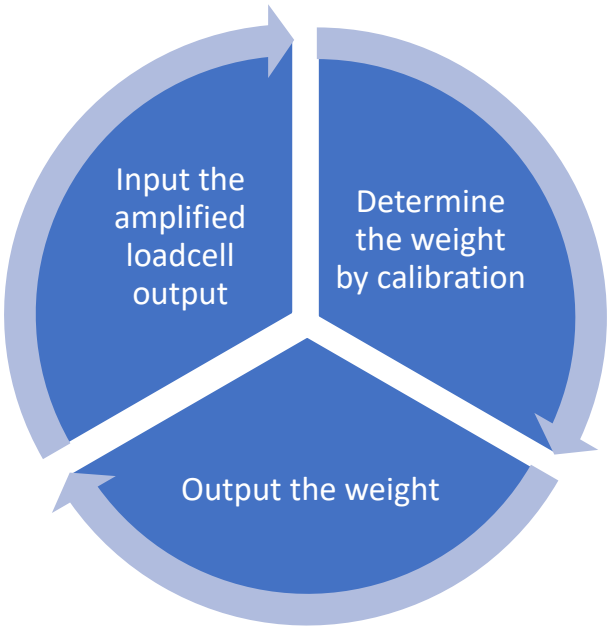
Main steps of the project



Logic



Main parts of our code



RESULTS

In order to calibrate our weight scale, we have tested it with several weights and note down its corresponding amplified voltage input to the microcontroller.

Real weight (kg)	Voltage input to microcontroller(V)		Real weight (kg)	Voltage input to microcontroller(V)
0	0.1		2.659	1.99
0	0.095		2.727	2
0.07	0.14		3	2.14
0.12	0.17		3.025	2.25
0.19	0.22		3.16	2.28
0.28	0.28		3.264	2.3
0.38	0.34		3.343	3.475
0.466	0.405		3.688	3.85
0.57	0.48		4	4.095
0.659	0.54		4.07	4.23
0.727	0.59		4.12	4.26
1	0.78		4.19	4.30
1.16	0.89		4.28	4.33
2	1.54		4.38	4.33
2.07	1.58		4.466	4.435
2.12	1.61		4.57	4.45
2.19	1.66		4.659	4.47
2.28	1.73		4.727	4.54
2.38	1.80		5	4.9
2.466	1.86		5	4.8
2.57	1.92			

We have plot the above readings in a weight-voltage graph to get best fitting line using **least square rule.**

Then the above equation **Y = 0.9847X – 0.2028** is used for calibrating the weight scale using the microcontroller. We have used least square rule as most of the readings we got is nearly linear and least square rule is the easiest method to find the equation of a line.

Sometimes we got different voltage outputs for same weight, so we used all that data so, we can get a mean value for that weight. We get different values for same weight due to various reasons.

- Affecting of noise and loose connection of wires
- Variating of input 230v.
- Load cell getting a permanent bend after some time
- High sensitivity of the load cell

We got an accuracy of 20 grams after calibrating carefully reducing lots of errors, but we couldn't get the 10 grams accuracy due to above mentioned reasons. (Sometimes this accuracy may change up to 80 grams due to high variating of the 230V power input or permanent bending of load cell after some measurements.)

Discussion

As we all were new to these kinds of projects, we have faced some difficulties during this project.

Problems we have faced during the project

- Several difficulties have been faced while designing and simulating the circuits as well as in making PCB layout too.
- When we have tested our circuits in bread board, we found an unexpected voltage drop in it.
- When we have just only amplified the loadcell output, we got outputs within the range including negative values also. (-2V to 4V)
- We faced difficulties while soldering the PCB.
- After assembling all PCBs, the weight scale didn't work as we expected.
- Output readings of our scale started to vary with time, even after we have corrected the code and our circuits as much as possible.

Reasons we have found for those issues mentioned above and some solutions we made to correct those

- First, we have used 5V regulators in power supply. After observing the voltage drop, we have replaced them with 10V regulators in order to increase the voltage given to the amplifier circuit.
- We have included an adder circuit to avoid the negative values in our amplified outputs.
- After manufactured the PCB, we found that we forgot to mirror it before printing so we remade the PCBs again.
- We have faced bad soldering, loose connection problems. We have rechecked every problem and due to some reasons, we have remade the PCB for our project.
- Output is varying due to noise, so we put capacitors between load cell and ground, microcontroller power and ground to minimize noise as much as possible.

Acknowledgement

As we were beginners, we had a big fear about this project at starting time. The encouragements and advices that we have received from everyone helped us in many ways.

First, we should thank our supervisor Ms. Theshani Nurada to lead us in a correct path to do this project this much successfully.

We would like to mention our gratitude towards our lecturers and instructors who encouraged us by clearing doubts and issues we had in our project.

Then we are in responsible place to thank personal in-charge of laboratories and workshop, for allowing us to use the space and equipments for our project and for helping us in technical issues.

We should thank our beloved batch friends who always showed their love by sharing their knowledge with us which were required in our project.

Further, we thank all people who have helped us in achieving this success.

BIBLIOGRAPHY

[1] Ramith Hettiarachchi (2018 Nov.) ATMega Tutorial Series 2 – (configuring USBasp with atmel studio) [Online].

Available: https://blog.ramith.fyi/static/atmega32-tutorial-series-2configuring-usbasp-with-atmel-studio/index.html

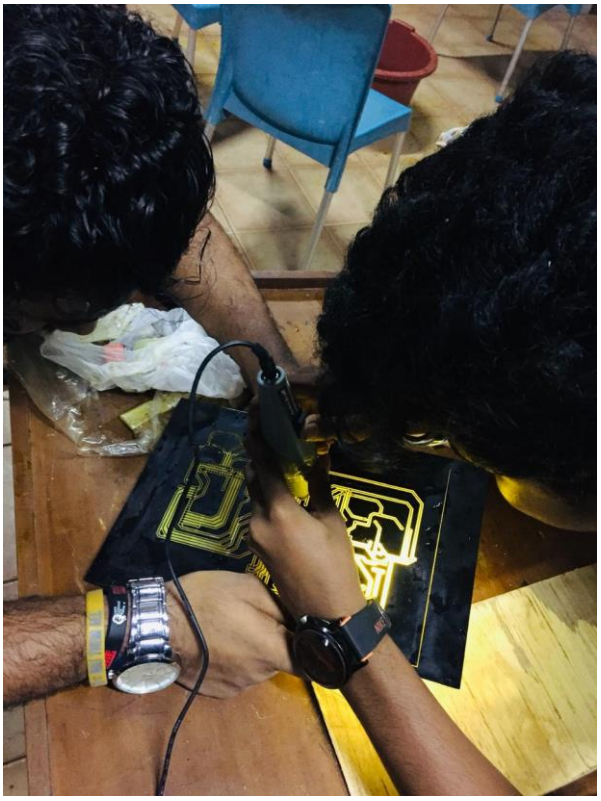
[2] Sparkfun getting started with load cells

Available: https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all

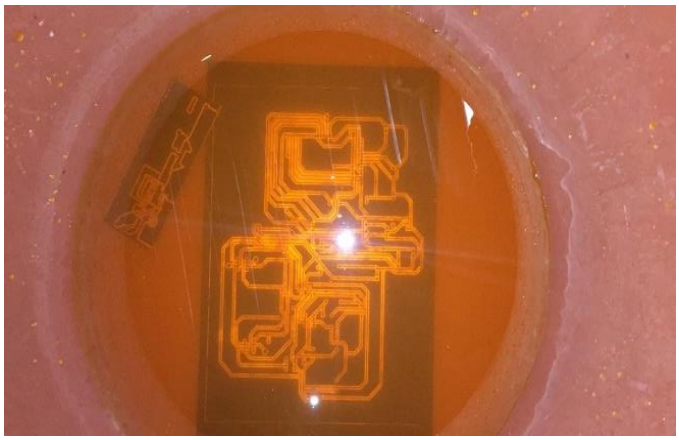
[3] All about circuits

Available: https://www.allaboutcircuits.com/projects/how-to-build-weighing-scale-wheatstone-bridge-strain-gauge-microcontroller/

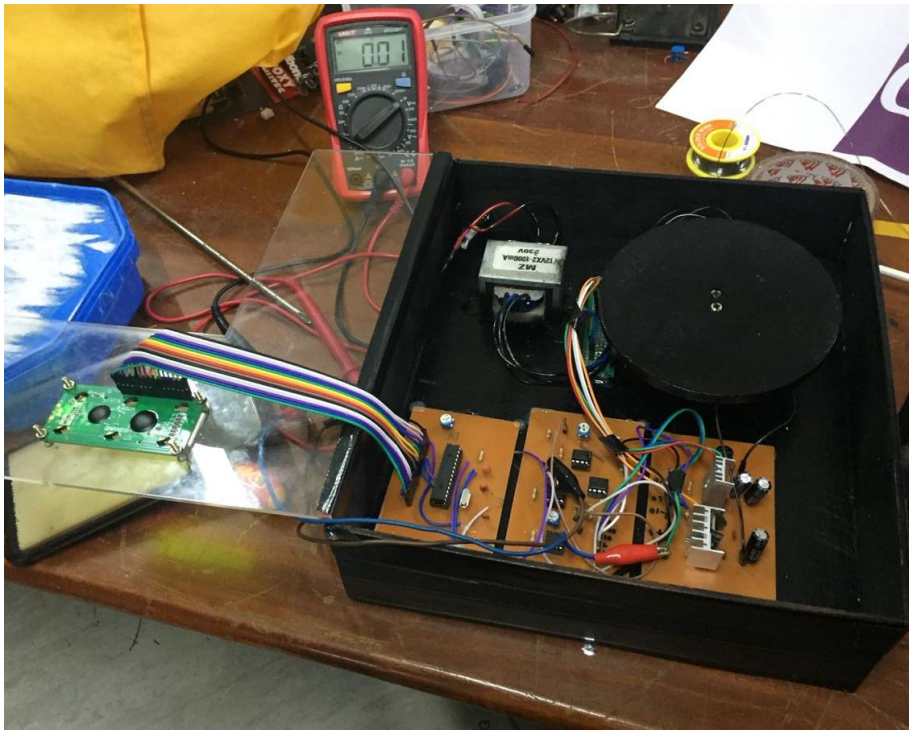
Appendix



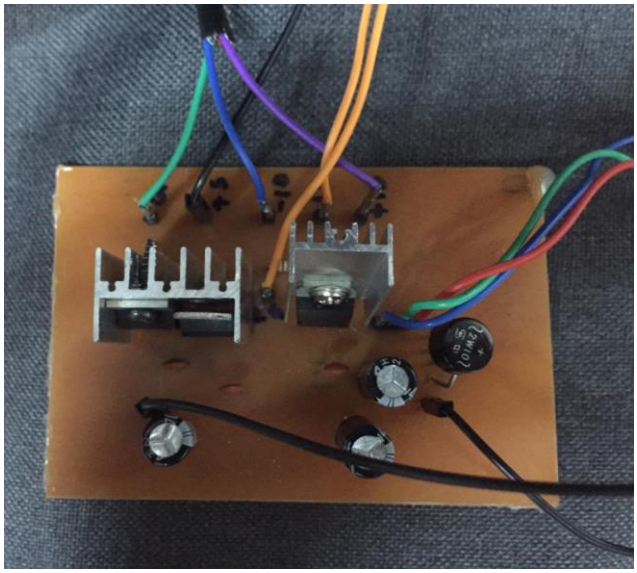
DRILLING



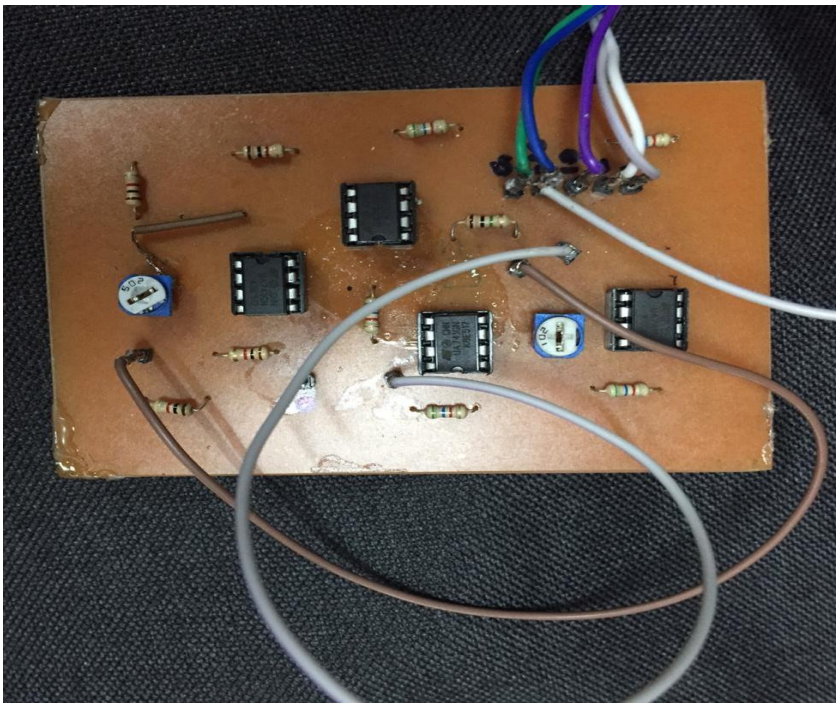
ETCHING



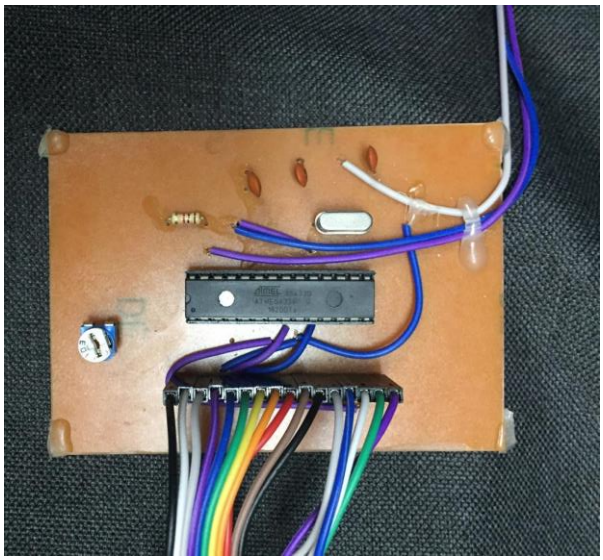
CALIBRATING



Power supply circuit

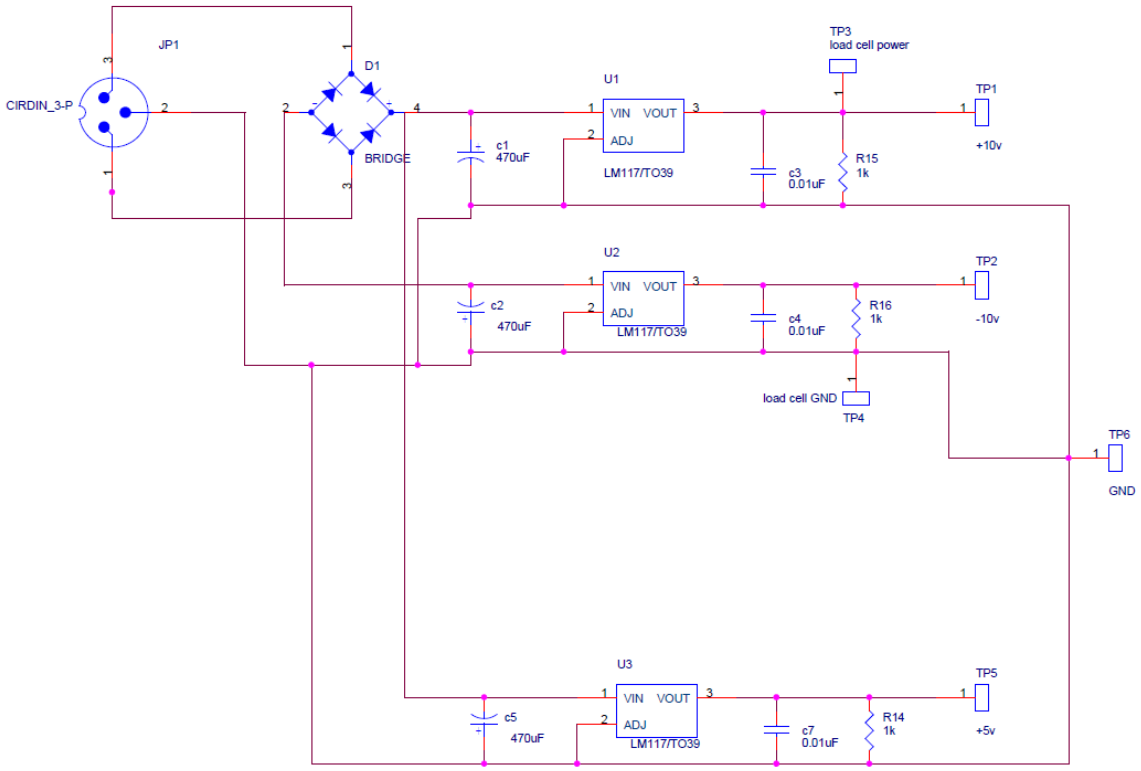


Amplifier and adder circuit

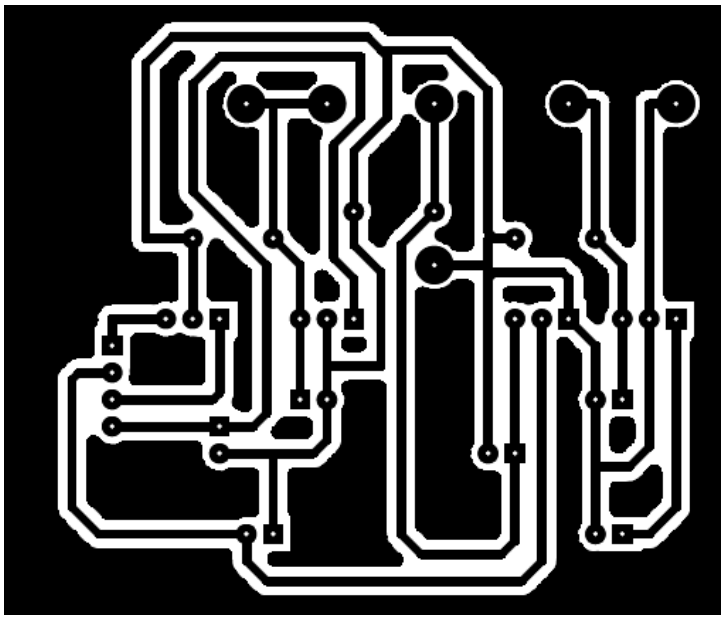


microcontroller circuit

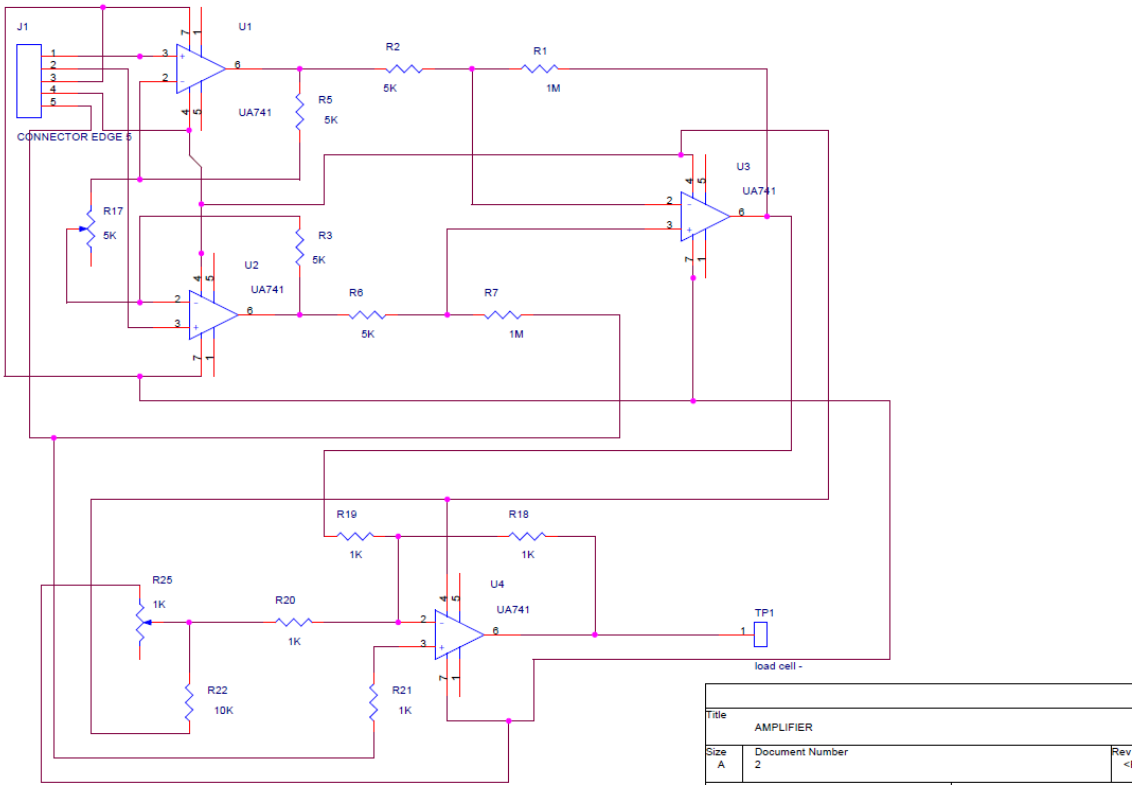
Appendix 1: Power supply schematic diagram



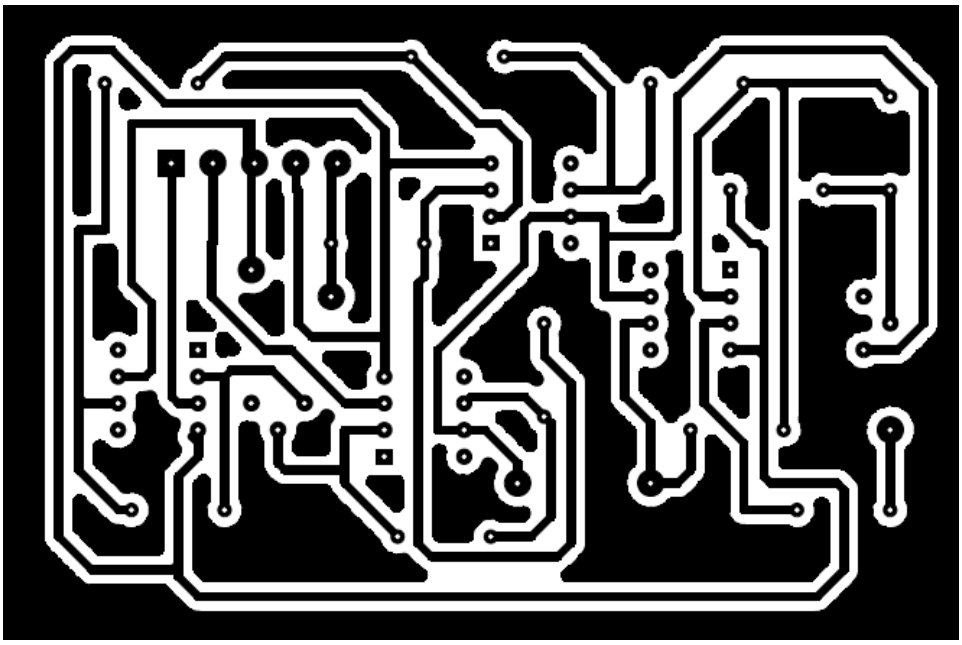
Appendix 2: Power supply layout



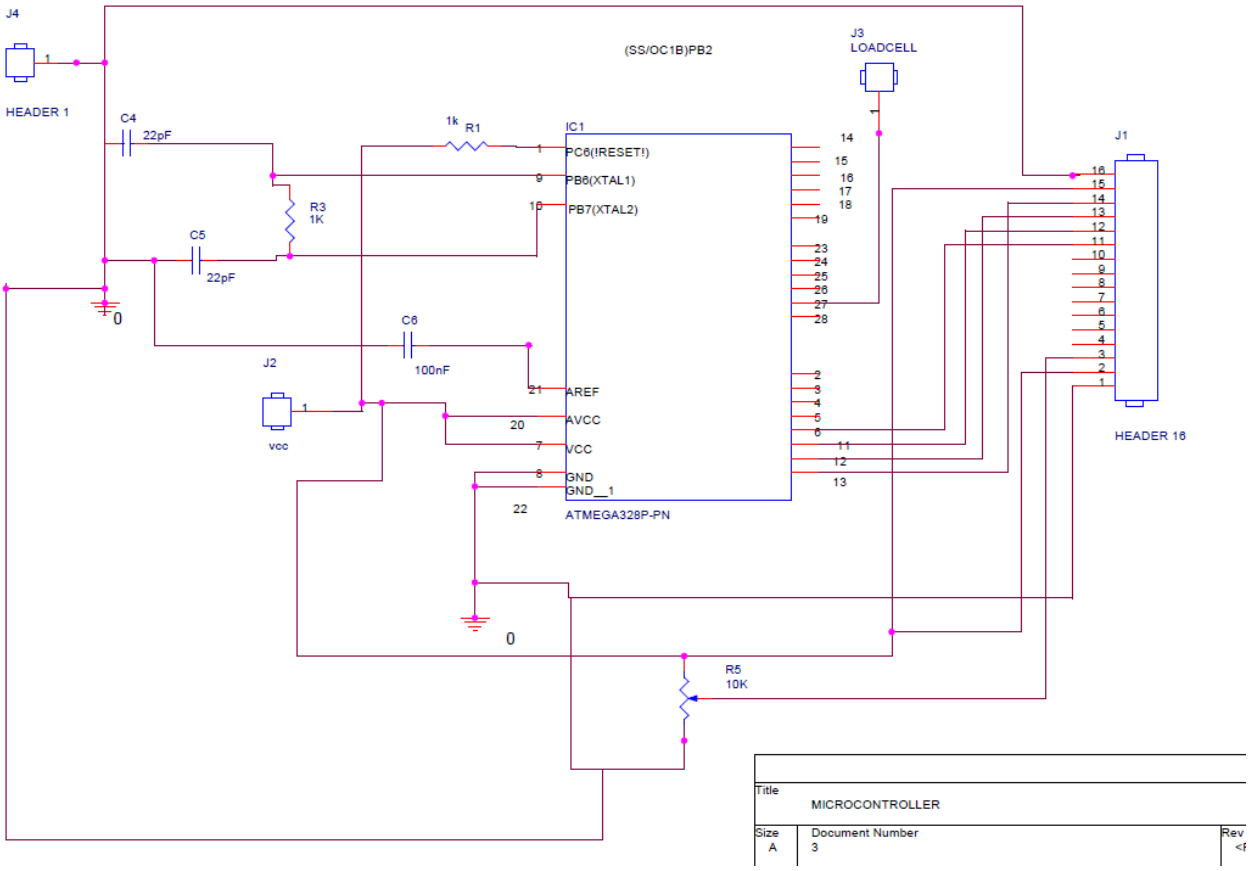
Appendix 3: Amplifier with adder circuit schematic diagram



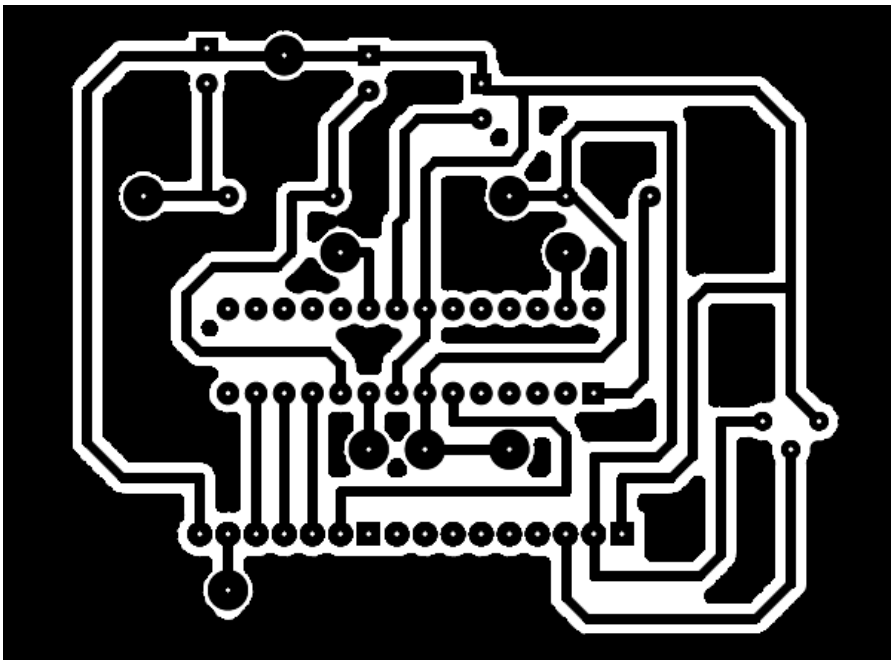
Appendix 4: Amplifier with adder layout



Appendix 5: Microcontroller ADC circuit schematic diagram



Appendix 6: Microcontroller layout



Appendix 7: Microcontroller Coding

```
#define F_CPU 800000UL
#define D4 eS_PORTD4
#define D5 eS_PORTD5
#define D6 eS_PORTD6
#define D7 eS_PORTD7
#define RS eS_PORTB6
#define EN eS_PORTB7

#include <stdlib.h>
#include <avr/io.h>
#include <util/delay.h>


#include "lcd.h"



void Adcinit()
{
	DDRC= 0b00000000;	// make port c as input
	ADMUX=0b01000000;	 // using Avcc with external capacitor at vref and 10bit
	//ADMUX=0b01100000;
	ADCSRA = (1<<ADEN);

}

float AdcRead(int a)
{
	unsigned short x;
	//ADMUX=0b01100000 + a;
	ADMUX=0b01000000 + a;
	ADCSRA = ADCSRA|(1<<ADSC);
	while (ADCSRA & (1<<ADSC));
	//x=ADCH;
	x = ADC;
	return x ;
}

float y_axis;
float y_axiss;
float i;
char* str1 = "Weight(Kg)";
```

P.T.O

```
int main(void)
{
	DDRB=0xFF;
	DDRD=0xFF;
	Adcinit();
	Lcd4_Init();
	 while (1)
	 {



		i=1;
		y_axiss=0;

		while (i<50){
			y_axis=AdcRead(4);
			y_axiss=(y_axiss+y_axis)/2;
			y_axiss=y_axis;
			i=i+1;
		}
		//y_axiss=y_axiss/i;
		char *y="000000";
		y_axiss=(y_axiss/1.023)*5;
		itoa(y_axiss,y,10);
		Lcd4_Set_Cursor(1,0);
		Lcd4_Write_String(str1);
		Lcd4_Set_Cursor(2,0);
		Lcd4_Write_String(y);
		_delay_ms(200);
		Lcd4_Clear();

	}


}
```

lcd.h library

```
#define eS_PORTB0 0
#define eS_PORTB1 1
#define eS_PORTB2 2
#define eS_PORTB3 3
#define eS_PORTB4 4
#define eS_PORTB5 5
#define eS_PORTB6 6
#define eS_PORTB7 7
#define eS_PORTC0 8
#define eS_PORTC1 9
#define eS_PORTC2 10
#define eS_PORTC3 11
#define eS_PORTC4 12
#define eS_PORTC5 13
#define eS_PORTC6 14
```

```
#define eS_PORTD0 15
#define eS_PORTD1 16
#define eS_PORTD2 17
#define eS_PORTD3 18
#define eS_PORTD4 19
#define eS_PORTD5 20
#define eS_PORTD6 21
#define eS_PORTD7 22


#ifndef D0
#define D0 eS_PORTC0
#define D1 eS_PORTC1
#define D2 eS_PORTC2
#define D3 eS_PORTC3
#endif


#include<util/delay.h>

void pinChange(int a, int b)
{
   if(b == 0)
   {
        if(a == eS_PORTB0)
        PORTB &= ~(1<<PB0);
      else if(a == eS_PORTB1)
        PORTB &= ~(1<<PB1);
      else if(a == eS_PORTB2)
        PORTB &= ~(1<<PB2);
      else if(a == eS_PORTB3)
        PORTB &= ~(1<<PB3);
      else if(a == eS_PORTB4)
        PORTB &= ~(1<<PB4);
      else if(a == eS_PORTB5)
        PORTB &= ~(1<<PB5);
      else if(a == eS_PORTB6)
        PORTB &= ~(1<<PB6);
      else if(a == eS_PORTB7)
        PORTB &= ~(1<<PB7);
      else if(a == eS_PORTC0)
        PORTC &= ~(1<<PC0);
      else if(a == eS_PORTC1)
        PORTC &= ~(1<<PC1);
      else if(a == eS_PORTC2)
        PORTC &= ~(1<<PC2);
      else if(a == eS_PORTC3)
        PORTC &= ~(1<<PC3);
      else if(a == eS_PORTC4)
        PORTC &= ~(1<<PC4);
      else if(a == eS_PORTC5)
        PORTC &= ~(1<<PC5);
        else if(a == eS_PORTC6)
          PORTC &= ~(1<<PC6);
      else if(a == eS_PORTD0)
        PORTD &= ~(1<<PD0);
      else if(a == eS_PORTD1)
        PORTD &= ~(1<<PD1);
      else if(a == eS_PORTD2)
        PORTD &= ~(1<<PD2);
      else if(a == eS_PORTD3)
        PORTD &= ~(1<<PD3);
```

```
      else if(a == eS_PORTD4)
        PORTD &= ~(1<<PD4);
      else if(a == eS_PORTD5)
        PORTD &= ~(1<<PD5);
      else if(a == eS_PORTD6)
        PORTD &= ~(1<<PD6);
      else if(a == eS_PORTD7)
        PORTD &= ~(1<<PD7);
   }
   else
   {
      if(a == eS_PORTB0)
        PORTB |= (1<<PB0);
      else if(a == eS_PORTB1)
        PORTB |= (1<<PB1);
      else if(a == eS_PORTB2)
        PORTB |= (1<<PB2);
      else if(a == eS_PORTB3)
        PORTB |= (1<<PB3);
      else if(a == eS_PORTB4)
        PORTB |= (1<<PB4);
      else if(a == eS_PORTB5)
        PORTB |= (1<<PB5);
      else if(a == eS_PORTB6)
        PORTB |= (1<<PB6);
      else if(a == eS_PORTB7)
        PORTB |= (1<<PB7);
      else if(a == eS_PORTC0)
        PORTC |= (1<<PC0);
      else if(a == eS_PORTC1)
        PORTC |= (1<<PC1);
      else if(a == eS_PORTC2)
        PORTC |= (1<<PC2);
      else if(a == eS_PORTC3)
        PORTC |= (1<<PC3);
      else if(a == eS_PORTC4)
        PORTC |= (1<<PC4);
      else if(a == eS_PORTC5)
        PORTC |= (1<<PC5);
      else if(a == eS_PORTC6)
        PORTC |= (1<<PC6);
      else if(a == eS_PORTD0)
        PORTD |= (1<<PD0);
      else if(a == eS_PORTD1)
        PORTD |= (1<<PD1);
      else if(a == eS_PORTD2)
        PORTD |= (1<<PD2);
      else if(a == eS_PORTD3)
        PORTD |= (1<<PD3);
      else if(a == eS_PORTD4)
        PORTD |= (1<<PD4);
      else if(a == eS_PORTD5)
        PORTD |= (1<<PD5);
      else if(a == eS_PORTD6)
        PORTD |= (1<<PD6);
      else if(a == eS_PORTD7)
        PORTD |= (1<<PD7);
   }
}
```

```
//LCD 8 Bit Interfacing Functions
void Lcd8_Port(char a)
{
   if(a & 1)
   pinChange(D0,1);
   else
   pinChange(D0,0);

   if(a & 2)
   pinChange(D1,1);
   else
   pinChange(D1,0);

   if(a & 4)
   pinChange(D2,1);
   else
   pinChange(D2,0);

   if(a & 8)
   pinChange(D3,1);
   else
   pinChange(D3,0);

   if(a & 16)
   pinChange(D4,1);
   else
   pinChange(D4,0);

   if(a & 32)
   pinChange(D5,1);
   else
   pinChange(D5,0);

   if(a & 64)
   pinChange(D6,1);
   else
   pinChange(D6,0);

   if(a & 128)
   pinChange(D7,1);
   else
   pinChange(D7,0);
}
void Lcd8_Cmd(char a)
{
   pinChange(RS,0);             // => RS = 0
   Lcd8_Port(a);             //Data transfer
   pinChange(EN,1);             // => E = 1
   _delay_ms(1);
   pinChange(EN,0);             // => E = 0
   _delay_ms(1);
}

void Lcd8_Clear()
{
   Lcd8_Cmd(1);
}

void Lcd8_Set_Cursor(char a, char b)
{
   if(a == 1)
   Lcd8_Cmd(0x80 + b);
```

```
   else if(a == 2)
   Lcd8_Cmd(0xC0 + b);
}

void Lcd8_Init()
{
   pinChange(RS,0);
   pinChange(EN,0);
   _delay_ms(20);
   ////////////// Reset process from datasheet /////////
   Lcd8_Cmd(0x30);
   _delay_ms(5);
   Lcd8_Cmd(0x30);
   _delay_ms(1);
   Lcd8_Cmd(0x30);
   _delay_ms(10);
   ////////////////////////////////////////////////////
   Lcd8_Cmd(0x38);    //function set
   Lcd8_Cmd(0x0C);    //display on,cursor off,blink off
   Lcd8_Cmd(0x01);    //clear display
   Lcd8_Cmd(0x06);    //entry mode, set increment
}

void Lcd8_Write_Char(char a)
{
   pinChange(RS,1);             // => RS = 1
   Lcd8_Port(a);             //Data transfer
   pinChange(EN,1);             // => E = 1
   _delay_ms(1);
   pinChange(EN,0);             // => E = 04
   _delay_ms(1);
}

void Lcd8_Write_String(char *a)
{
   int i;
   for(i=0;a[i]!='\0';i++)
   Lcd8_Write_Char(a[i]);
}

void Lcd8_Shift_Right()
{
   Lcd8_Cmd(0x1C);
}

void Lcd8_Shift_Left()
{
   Lcd8_Cmd(0x18);
}
//End LCD 8 Bit Interfacing Functions

//LCD 4 Bit Interfacing Functions

void Lcd4_Port(char a)
{
   if(a & 1)
   pinChange(D4,1);
   else
   pinChange(D4,0);

   if(a & 2)
   pinChange(D5,1);
```

```
   else
   pinChange(D5,0);

   if(a & 4)
   pinChange(D6,1);
   else
   pinChange(D6,0);

   if(a & 8)
   pinChange(D7,1);
   else
   pinChange(D7,0);
}
void Lcd4_Cmd(char a)
{
   pinChange(RS,0);             // => RS = 0
   Lcd4_Port(a);
   pinChange(EN,1);            // => E = 1
   _delay_ms(1);
   pinChange(EN,0);             // => E = 0
   _delay_ms(1);
}

void Lcd4_Clear()
{
   Lcd4_Cmd(0);
   Lcd4_Cmd(1);
}

void Lcd4_Set_Cursor(char a, char b)
{
   char temp,z,y;
   if(a == 1)
   {
      temp = 0x80 + b;
      z = temp>>4;
      y = (0x80+b) & 0x0F;
      Lcd4_Cmd(z);
      Lcd4_Cmd(y);
   }
   else if(a == 2)
   {
      temp = 0xC0 + b;
      z = temp>>4;
      y = (0xC0+b) & 0x0F;
      Lcd4_Cmd(z);
      Lcd4_Cmd(y);
   }
}

void Lcd4_Init()
{
   Lcd4_Port(0x00);
   _delay_ms(20);
   ////////////// Reset process from datasheet /////////
   Lcd4_Cmd(0x03);
   _delay_ms(5);
   Lcd4_Cmd(0x03);
   _delay_ms(11);
   Lcd4_Cmd(0x03);
   /////////////////////////////////////////////////////
   Lcd4_Cmd(0x02);
```

```
   Lcd4_Cmd(0x02);
   Lcd4_Cmd(0x08);
   Lcd4_Cmd(0x00);
   Lcd4_Cmd(0x0C);
   Lcd4_Cmd(0x00);
   Lcd4_Cmd(0x06);
}

void Lcd4_Write_Char(char a)
{
   char temp,y;
   temp = a&0x0F;
   y = a&0xF0;
   pinChange(RS,1);             // => RS = 1
   Lcd4_Port(y>>4);             //Data transfer
   pinChange(EN,1);
   _delay_ms(1);
   pinChange(EN,0);
   _delay_ms(1);
   Lcd4_Port(temp);
   pinChange(EN,1);
   _delay_ms(1);
   pinChange(EN,0);
   _delay_ms(1);
}

void Lcd4_Write_String(char *a)
{
   int i;
   for(i=0;a[i]!='\0';i++)
   Lcd4_Write_Char(a[i]);
}

void Lcd4_Shift_Right()
{
   Lcd4_Cmd(0x01);
   Lcd4_Cmd(0x0C);
}

void Lcd4_Shift_Left()
{
   Lcd4_Cmd(0x01);
   Lcd4_Cmd(0x08);
}
//End LCD 4 Bit Interfacing Functions
```