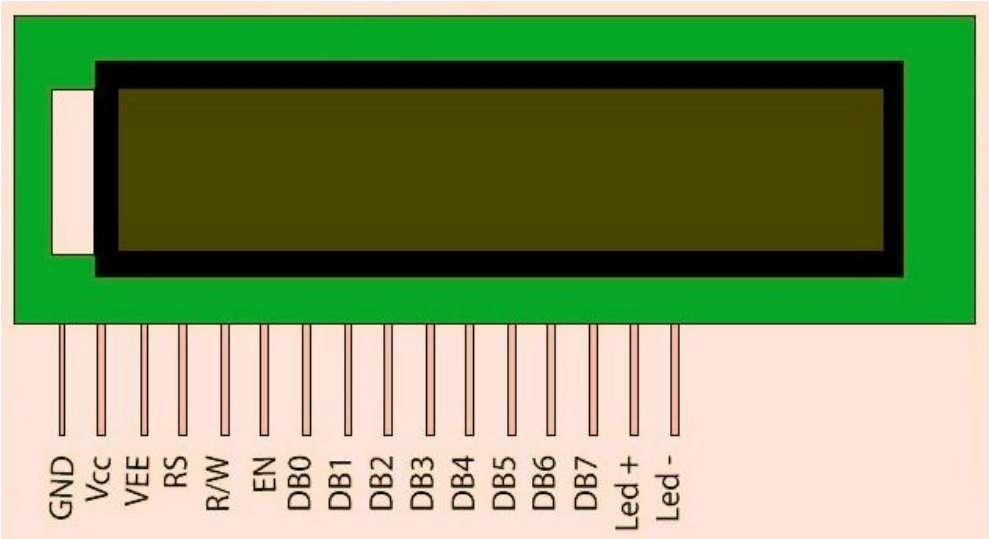
Signal Conditioning and Data Processing

To complete the whole system design, integration of individual parts was the first on the list of things to be done. There are following individual modules have been developed:

1. Signal conditioning circuit using two instrumentation amplifiers (INA129)
2. Power supply circuit to get dual power supply (+15 V and -15 V) from 220 V AC supply
3. Power supply circuit to get +7V for load cell
4. Microcontroller ATmega for analog to digital conversion of output obtained after signal conditioning
5. LCD to display any alphanumeric characters (interfaced using the microcontroller)

The initial task was to combine the individual modules number 4 and 5 mentioned above (i.e. to display the digital converted value on LCD)
The following diagram shows LCD connections to the microcontroller (We are using a 16 x 2 LCD for our purpose):



To simply run the LCD using the ATmega 2560 microcontroller, the following code was used:

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins. Refer to the LCD 16x2 Diagram.
//Also refer to arduino liquid crystal library webpage where assignment of lcd pins and their
//connection is mentioned in detail.

LiquidCrystal lcd(48,46,44,42,40,38,36,34,32,30,28);
void setup() {
  // set up the LCD's number of columns and rows:
  pinMode(52, OUTPUT);      // set pin to OUTPUT, Vcc pin
  digitalWrite(52, HIGH);
```

```
  pinMode(50, OUTPUT);       // set pin to OUTPUT
  digitalWrite(50, LOW);       // turn on pullup resistors to set this to 0Volts, Vdd
  lcd.begin(16, 2);                  //initialize the lcd using this command
  // Print a message to the LCD.
  lcd.print("a"); //display any value
  lcd.setCursor(0, 1);
  lcd.print("bb"); //display any value in the next line

void loop() {
 /*nothing is written here since this code only initialises the lcd ones with the above given
values and those values remain on the lcd until power is switched off. To get dynamically
changing values continuously, the lcd.print command can be written here and variables can
be used. See the next code */
}
```

Now, to decrease the number of pins used and also switch on the backlight of the LCD the following code was used :

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(48,44,34,32,30,28); //Only 4 data pins used now, refer to the arduino liquid
//crystal library. The rest of the code is similar to previous one.

void setup() {
  // set up the LCD's number of columns and rows:
  pinMode(52, OUTPUT);       // set pin to OUTPUT
  digitalWrite(52, HIGH);
  pinMode(46, OUTPUT);       // set pin to OUTPUT
  digitalWrite(46, LOW);         //RS pin is also grounded in this case of 4 data pins
  pinMode(50, OUTPUT);       // set pin to OUTPUT
  digitalWrite(50, LOW);// turn on pullup resistors
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("a"); //display anything
  lcd.setCursor(0, 1);
  lcd.print("bb"); //display anything

  //setting backlight, by setting the last two pins of the LCD to be high and low respectively.
  pinMode(24, OUTPUT);       // set pin to OUTPUT
  digitalWrite(24, LOW);
  pinMode(26, OUTPUT);       // set pin to OUTPUT
 digitalWrite(26, HIGH);
}
```

```
void loop() {

}
```

Using the above code, the LCD was now glowing it's backlight and also only 4 data pins were being utilised to display anything to the LCD. Now, to combine this code with the ADC code previously developed, some modifications were made. Now, number 4 and number 5 modules mentioned above have been combined.

```
#include <LiquidCrystal.h>

LiquidCrystal lcd(48,44,34,32,30,28); //using 4 data pins d4-d7.

//RW is gnd, RS and EN are in the func.

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
        // set up the LCD's number of columns and rows:
  pinMode(52, OUTPUT);      // set pin to OUTPUT
  digitalWrite(52, HIGH);
  pinMode(46, OUTPUT);      // set pin RS to OUTPUT and GND
  digitalWrite(46, LOW);
  pinMode(50, OUTPUT);      // set pin to OUTPUT
  digitalWrite(50, LOW);

  //setting backlight
  pinMode(24, OUTPUT);      // set pin to OUTPUT
  digitalWrite(24, LOW);
  pinMode(26, OUTPUT);      // set pin to OUTPUT
  digitalWrite(26, HIGH);

}
```

```
void loop() {


  weight= 1.027*analogRead(inputPin); //1.027 is the slope obtained from readings
//(Calibration factor)
  // add the reading to the total:
  Serial.println(weight);
```

```
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print(weight);
  lcd.setCursor(5, 0);
  lcd.print(grams);
  delay(2);       // delay in between reads for stability, can be increased for better readability
}
```

The sampling rate of the ADC is very fast and hence the values being displayed change continuously and are not visible statically for better aesthetic display to the user. Also, the readings are not very accurate and change continuously. Hence, we proposed the solution to this problem in terms of a moving average filter.
This moving average filter would make the values more stable on the display and also this would provide for post-processing of the data that we are obtaining through the signal conditioning and digital conversion. This post processing is done by taking a moving average of a fixed number of samples. This improved the final result and the output value of the weight as well. The following code implements the moving average filter and can be considered as the final code that can be used for weight measurement and display system.

```
#include <LiquidCrystal.h>

const int numReadings = 10;

int readings[numReadings];   // the readings from the analog input
int readIndex = 0;              // the index of the current reading
int total = 0;              // the running total
int average = 0;                // the average

int inputPin = A0;

// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(48,44,34,32,30,28); //using 4 data pins d4-d7.

//RW is gnd, RS and EN are in the func.

void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
        // set up the LCD's number of columns and rows:
  pinMode(52, OUTPUT);      // set pin to OUTPUT
  digitalWrite(52, HIGH);
  pinMode(46, OUTPUT);      // set pin RS to OUTPUT and GND
  digitalWrite(46, LOW);
  pinMode(50, OUTPUT);      // set pin to OUTPUT
  digitalWrite(50, LOW);// turn on pullup resistors
```

```
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
          readings[thisReading] = 0;
  }
  //setting backlight
  pinMode(24, OUTPUT);      // set pin to OUTPUT
  digitalWrite(24, LOW);
  pinMode(26, OUTPUT);      // set pin to OUTPUT
  digitalWrite(26, HIGH);

}

// the loop routine runs over and over again forever:
void loop() {

  // subtract the last reading:
  total = total - readings[readIndex];
  // read from the sensor:
  readings[readIndex] = 1.027*analogRead(inputPin); //1.027 is the slope obtained from
readings // (Calibration factor)
  // add the reading to the total:
  total = total + readings[readIndex];
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...
  if (readIndex >= numReadings) {
          // ...wrap around to the beginning:
          readIndex = 0;
  }

  // calculate the average:
  average = total / numReadings;
  // send it to the computer as ASCII digits
  Serial.println(average);
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print(average);
  lcd.setCursor(5, 0);
  lcd.print(grams);
  delay(2);       // delay in between reads for stability
}
```

On testing the whole setup we could obtain the weight values on the LCD successfully. All of the above codes ran on ATmega 2560 Arduino Mega microcontroller.

To decrease the cost and to make the overall system compact, Arduino Nano board was instead used. The final code that worked after all modifications is given below. This code is similar to the ones described in detail above.

```
#include <LiquidCrystal.h>

const int numReadings = 30; /* 30 readings are being averaged out for better accuracy. This can be increased for even better accuracy but that would be at the cost of higher memory usage and time. Point to note here is that Arduino Nano doesn't have enough memory so this value should be increases with caution. */

int readings[numReadings];   // the readings from the analog input
int readIndex = 0;           // the index of the current reading
int total = 0;               // the running total
int average = 0;             // the average

int inputPin = A7;           /*the pin number of arduino nano where input from signal
                             conditioning is being taken*/
int groundPin = A6;          /*the ground pin is also taken as another input to adc's second channel so as to diminish the effects of noises that ground might be carrying. This value is subtracted from the original output to lead to a better final value. */



// initialize the library with the numbers of the interface pins

LiquidCrystal lcd(4, 6, 9, 10, 11, 12); //using 4 data pins d4-d7. see arduino nano pin diagram                                        //for connection details



//RW is gnd, RS and EN are in the func.

/*
  AnalogReadSerial
  Reads an analog input on pin 0, prints the result to the serial monitor.
  Attach the center pin of a potentiometer to pin A0, and the outside pins to +5V and ground.

 */

// the setup routine runs once when you press reset:
void setup() {
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600); /*this is for debugging purposes only, that is when the microcontroller
                    board is connected to the computer, the values can be viewed on a serial port
*/
  // set up the LCD's number of columns and rows:
```

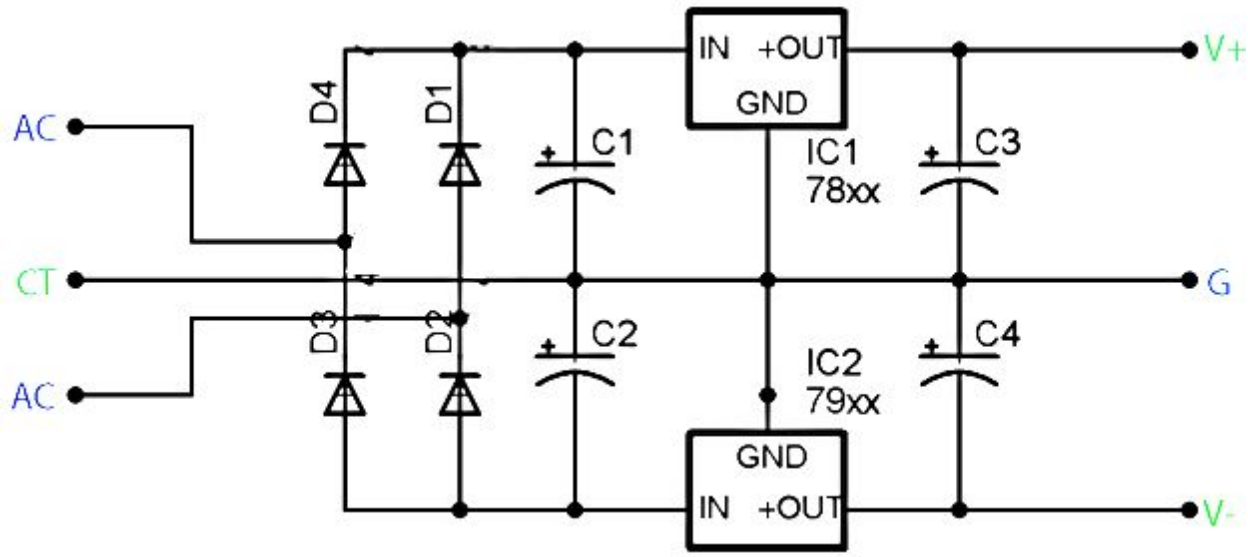
```
  pinMode(2, OUTPUT);         // set pin to OUTPUT
  digitalWrite(2, HIGH);
  pinMode(5, OUTPUT);         // set pin RS to OUTPUT and GND
  digitalWrite(5, LOW);
  pinMode(3, OUTPUT);         // set pin to OUTPUT
  digitalWrite(3, LOW);// turn on pullup resistors
  for (int thisReading = 0; thisReading < numReadings; thisReading++) {
          readings[thisReading] = 0;
  }

}

void loop() {

  // subtract the last reading:
  total = total - readings[readIndex];
  // read from the sensor: slope calculated is mentioned in this line.
  readings[readIndex] = 1.035445373 * (analogRead(inputPin)-analogRead(groundPin));
  // add the reading to the total:
  total = total + readings[readIndex];
//  total = max(total, readings[readIndex]);
  // advance to the next position in the array:
  readIndex = readIndex + 1;

  // if we're at the end of the array...
  if (readIndex >= numReadings) {
          // ...wrap around to the beginning:
          readIndex = 0;
  }

  // calculate the average:
  average = total / numReadings;
  average=average-10; ..offset is subtracted here. change accordingly.


  // send it to the computer as ASCII digits
  //  Serial.println(average); //for debugging purposes only, i.e. to view on computer, the
values.

  // Print a message to the LCD.
  lcd.begin(16, 2);
  lcd.print(average);
  lcd.setCursor(5, 0);
  lcd.print("gm");
  delay(100);              // delay in between reads for stability
}
```

Next,

Soldering the power supply circuit to supply the following voltages:

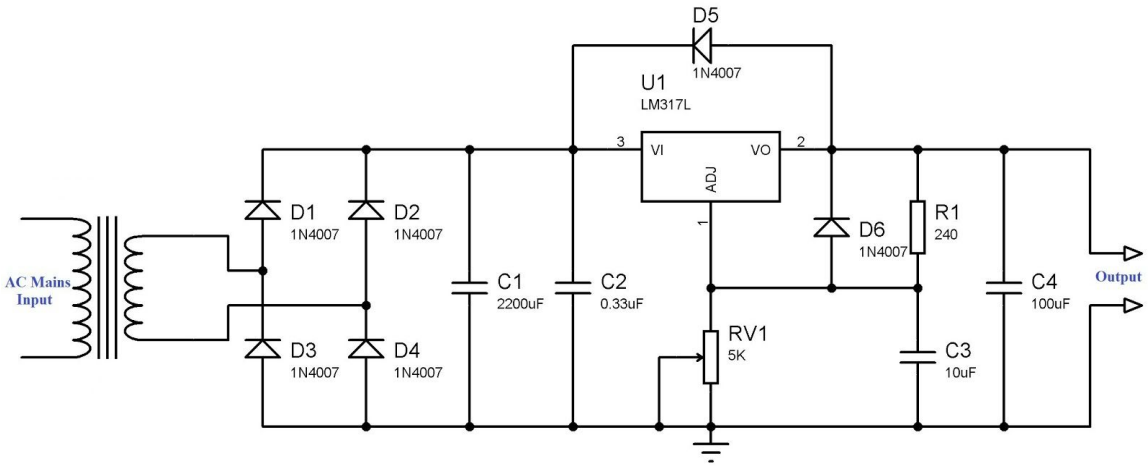
a. +15V for INA129 positive supply
b. -15V for INA129 negative supply
c. +7V for load cell and micrcontroller supply

IC7812 and IC7912 were used respectively for the two voltages. These voltage regulator ICs provide constant voltage and can supply a maximum current of 1.2A. These specifications are suitable for our instrumentation application of weight measurement using load cell.

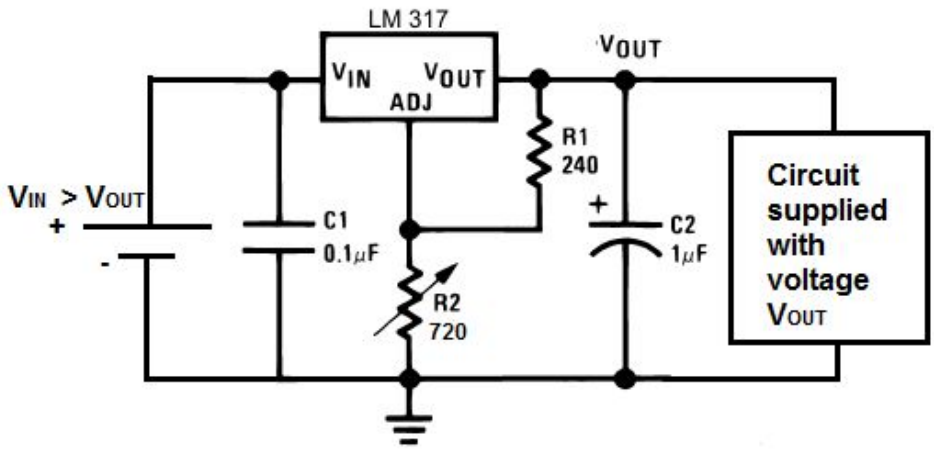


In the above diagram, C1 and C2 are polar capacitances of value 1000uF and C3 and C4 are nonpolar of value 0.1uF. When xx=15, V+=15V and V-=-15V.

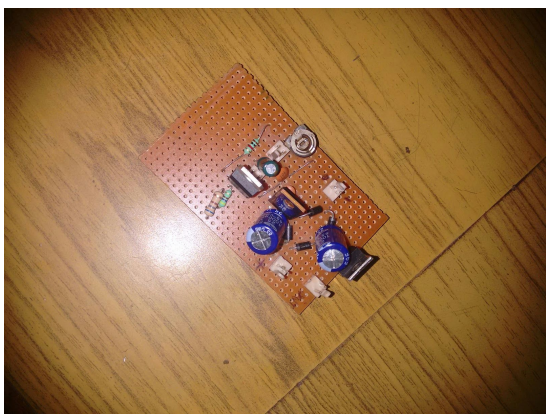
Similar circuit using LM317 can be used to obtain any voltage between the range of 1V - 30V by adjusting the resistance. We used this to get +7V for load cell and microcontroller supply. The following circuit can be used:



But. since we have already rectified the AC signal to get +15 DC voltage output from the first circuit, we need not rectify the signals again for +7V. So, the 15V DC output obtained in the first part can be used to supply +7V DC output using LM317. The above circuit in this case can be replaced by:



The breadboard simulation of the above circuit was done and was taken for soldering on a PCB. The following images show the soldered circuit which is yet to be tested rigorously for problems.



Possible Modifications:

1. Low pass filtering using RC on the output of the sensor

2. Decoupling the sensor by using a capacitor across its power supply
3. In the microcontroller code, implementing a mode filtering function to improve the stability of the readings and accuracy
4. Using 1.1V reference voltage so as to measure weights of lesser resolutions as well.
5. Implementing higher order butterworth (or other) filters to improve the stability of the data.

Conclusion:

The aim of the project was satisfied and the weights are being displayed as required. Though, accuracy and resolution improvements are to be done to make the whole system design perfectly working.