Instrumentation System Design Laboratory Report

Date: October 8th 2015

Tasks for the week:

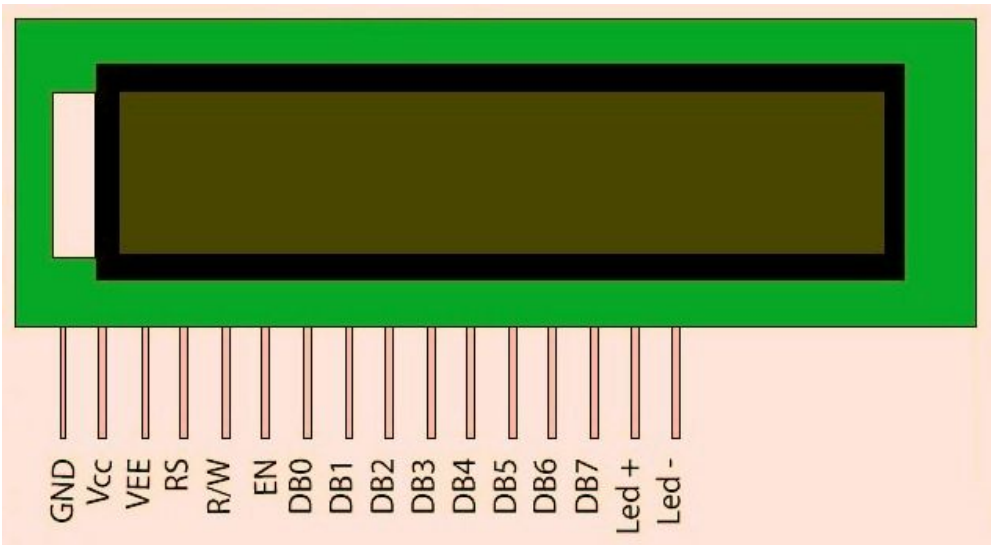
1. Interfacing of LCD with microcontroller
2. Complete setup working using
    a. Power supply circuit to power the instrumentation amplifier and load cell
    b. Microcontroller and LCD interfacing to display the weight values

Work done:

To complete the whole system design, integration of individual parts was the first on the list of things to be done in the week. There are following individual modules that we have developed up till now:

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)

For this week, 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 being used right now, the following code was used:

```
#include <LiquidCrystal.h>

// initialize the library with the numbers of the interface pins
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
  digitalWrite(52, HIGH);
  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 any value
  lcd.setCursor(0, 1);
  lcd.print("bb"); //display any value in the next line

void loop() {

}
```

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

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
```

```
  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, we found that 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, we made some modifications and achieved the following code for the microcontroller. 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);

}



// the loop routine runs over and over again forever:
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
}
```

On using the above code, there is a problem that we faced. 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. But, we were faced with another problem when we went for testing another day when one of the instrumentation amplfier ICs stopped working. On further testing, we found that due to one of the IC which was not working, the second IC was also rendered useless and now we need to procure new instrumentation amplifiers to move ahead with the work.

Discussions and Problems Faced:

1. LCD connection can be done using only 4 data pins of the LCD and it was implemented in the code.
2. For better readability, the backlight (which was initially switched off) was switched on by making appropriate changes to the code
3. The final output data needs filtering to obtain a stable output display on the LCD
4. A moving average filter was chosen and applied for every 10 samples of digital values.
5. Also, a delay of 2 seconds was added to the code to attain stability in the display.
6. The whole setup was working, but due to some fault in INA129, the progress was halted and solutions need to be looked into for it.

Tasks for the weeks to come:

1. Fixing the instrumentation amplifier problem OR procuring new ICs.
2. Soldering the whole setup to make it permanent.
3. Procuring our own transformer (+18V 0V -18V)
4. Rigorous readings, manual and report writing.

Instrumentation System Design Laboratory Report

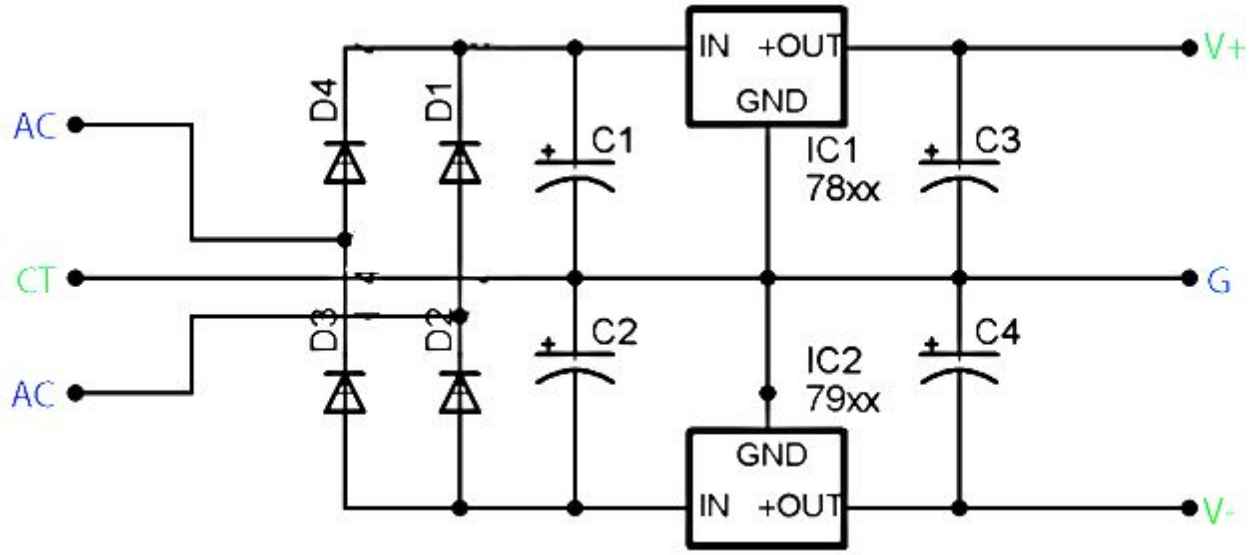
Date: October 14th 2015

Tasks for the week:

1. 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

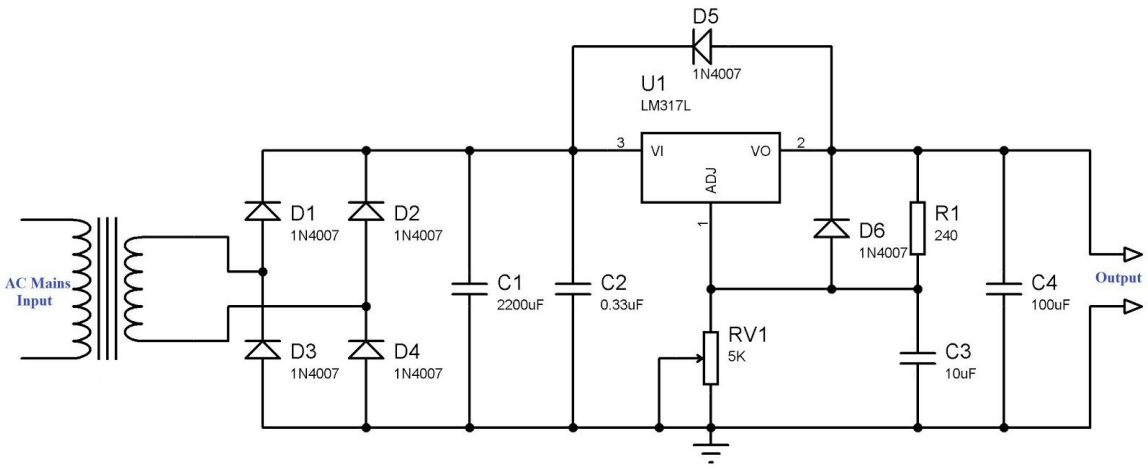
Progress:

The following circuit was checked on a breadboard for +15V and -15V. 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.

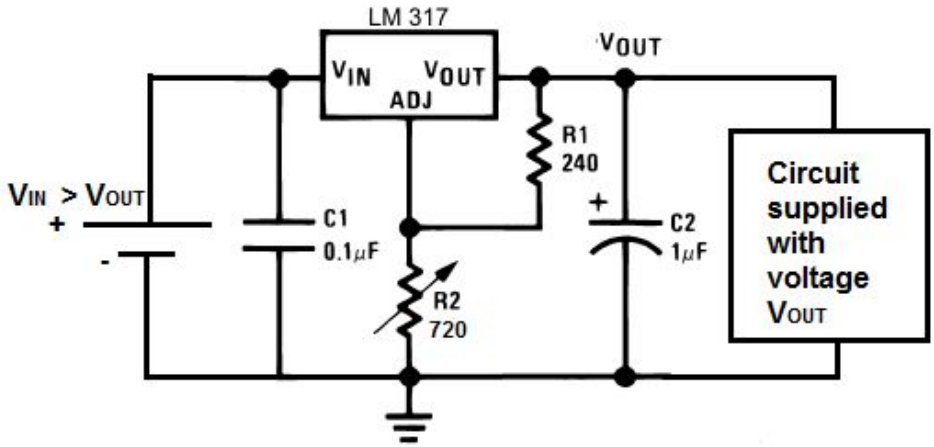


In the above diagram, C1 and C2 are polar capacitances of value 1000uF and C3 and C4 are nonpolar of value 0.1uF. When xx=15, V+=15V and V-=-15V.

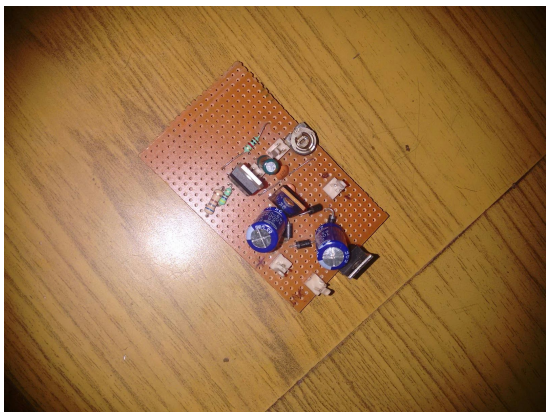
Similar circuit using LM317 can be used to obtain any voltage between the range of 1V - 30V by adjusting the resistance. We used this to get +7V for load cell and microcontroller supply. The following circuit can be used:

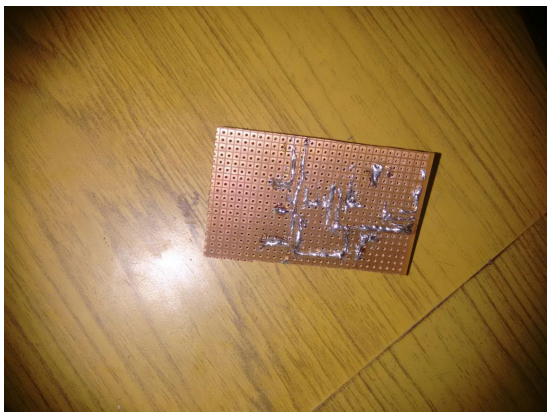


But. since we have already rectified the AC signal to get +15 DC voltage output from the first circuit, we need not rectify the signals again for +7V. So, the 15V DC output obtained in the first part can be used to supply +7V DC output using LM317. The above circuit in this case can be replaced by:



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.





Tasks for next week:

1. Soldering the signal conditioning circuit along with the microcontroller to complete the whole setup on PCB.
2. Checking rigorously for problems the power supply circuit soldered this week and using it to actually supply the power required while testing.

Problems faced and Discussions:

1. Redundancy of rectification for the two kinds of voltages was a problem which was solved by regulating the higher of the two voltages (+15V) to the lower voltage (+7V). Accordingly, circuit modifications were made.
2. Optimization of component placement to minimize the solder space was a problem at hand before soldering. Also, it was a challenge to not use any soldered jumper wires on the PCB. Appropriate placement of the components was achieved as shown in the pictures above to complete the soldering.
3. Testing of the power circuit with varying load requirements of current have to be checked and if the ICs heat up then a heat sink might have to be used with the ICs used. Conceptually, minimal current (in the range of mA) will be withdrawn but still checking the circuit for robustness is necessary and hence would be one of the targets for the following week.

Lab Report: 4th Nov. 2015

Problems faced:

1. Various difficulties in the final circuits on PCB
    a. Capacitor dysfunctional: Replaced by new one.
    b. IC 7912 for negative voltage stopped functioning : Replaced by a new one.
    c. Due to excessive heating at times, there were problems : Heat sinks are now in place.
    d. LCD stopped working when soldered on PCB : Debugging complete
    e. Microcontroller code having various issues : Debugging still left.
    f. Offset voltage : Adjusted by adjusting gain properly and taking care of the power supply.

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.

Conclusions:

The project is almost complete with only thing that is left now is to improve the data stability on the LCD and to improve the resolution of the whole device. Both of these tasks are very much achievable and can be posted as an aim of an experiment to students.
Here is a picture showing the complete setup: