

## Objectives

The purpose of this lab session is to study the way to make a measurement by using the ADC, and then show the outcome by using LCD display.

## Methodology

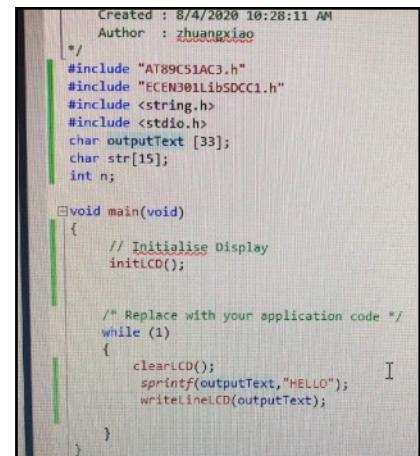
### LCD Display

The equipments I used in this lab session were microcontroller and LCD display.

1) Connected LCD display into microcontroller. Plug the data line into port 0, and the control lines into port 4.

The program code shown in figure 1. I initially met an issue, the LCD display repeated the same character string without stopping. And I found that it was because I didn't write `clearLCD()`; at the beginning of while loop, to clear the display after every loops were done.

There should be a `delay(30000);` at the end of while loop, otherwise the machine does not have enough time to print all of the characters.



```
Created : 8/4/2020 10:28:11 AM
Author : zhuangxiaobin

*/
#include "AT89C51AC3.h"
#include "ECEN301libSDCC1.h"
#include <string.h>
#include <stdio.h>
char outputText [33];
char str[15];
int n;

void main(void)
{
    // Initialise Display
    initLCD();

    /* Replace with your application code */
    while (1)
    {
        clearLCD();
        sprintf(outputText, "HELLO");
        writelineLCD(outputText);
    }
}
```

Fig. 1

By changing the outputText content, not only "HELLO", but also something else can be displayed. (Do not over the outputText size [33])

The result shown in figure 2.

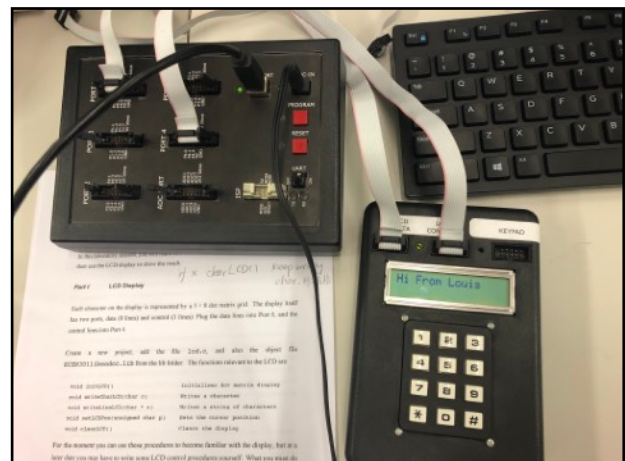


Fig. 2

2) base on previous session, add numbers into the outputText. Code shown in figure 3.  
Define two integer: *month* and *day*. Assign month = 8, day = 4. The result would be "The date is 4/8".

Here in the figure 4, the outcome was different with code. Because the LCD displayed the previous information which I used for testing the code. The corresponding code was `sprintf(outputText, "Today is %d/%d", month, day);`

Same with the previous, `delay()` needed to be set at the end of loop to give it enough time to print characters. The `delay()` was missing in figure 3.

```
#include "AT89C51AC3.h"
#include "ECEN301LibSDCC1.h"
#include <string.h>
#include <stdio.h>
char outputText [33];
char str[15];
int n;
int month;
int day;

void main(void)
{
    // Initialise Display
    initLCD();

    /* Replace with your application code */
    while (1)
    {
        clearLCD();
        sprintf(outputText, "The date is %d/%d", day, month);
        writelineLCD(outputText);
    }
}
```

Fig. 3

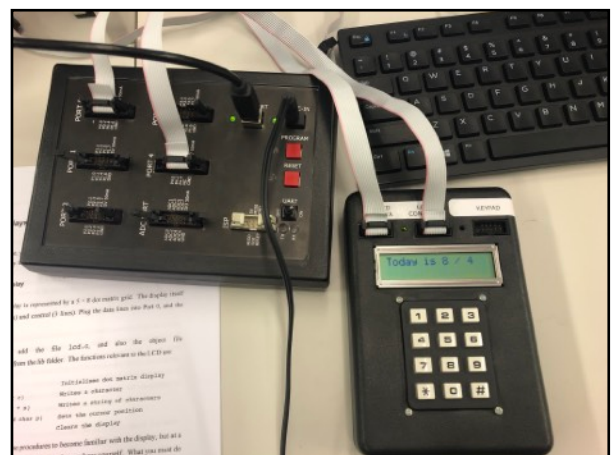


Fig. 4

3) in this session, I generated a counter. Main code shown in figure 5.

There was a different thing with the previous program, defined an integer *count* and assigned 0 to it. And because of `count++` the integer would increased 1 when a loop was done.

The connection and display result shown in figure 6.

```
void main(void)
{
    initLCD();

    while (1)
    {
        clearLCD();
        // (1)
        // (2)
        // (3)
        sprintf(outputText, "Count is = %i", count);
        writelineLCD(outputText);
        count++;
        //delay(32000) //take 17 second to count 100 number
        delay(10000); //take 6 second to count 100 number
    }
}
```

Fig. 5



Fig. 6

I found that the counter speed depended on the delay length. For example, if the delay was 10000, it would take around 6 second to count 100 numbers; if the delay was 30000, it would take around 17 second to count 100 numbers. Which means the delay length and count speed are linear.

## Analogue to Digital Conversion

In this lab session, I used ADC port to convert analogue signal to digital signal and showed on LCD display. The equipments I used were microcontroller, LCD display, breadboard and resistance box. The real connection shown in figure7. (because there were not enough wire, I used inappropriate colour of wire. )

In case the connection is too messy to understand, the schematic shown in figure 8.

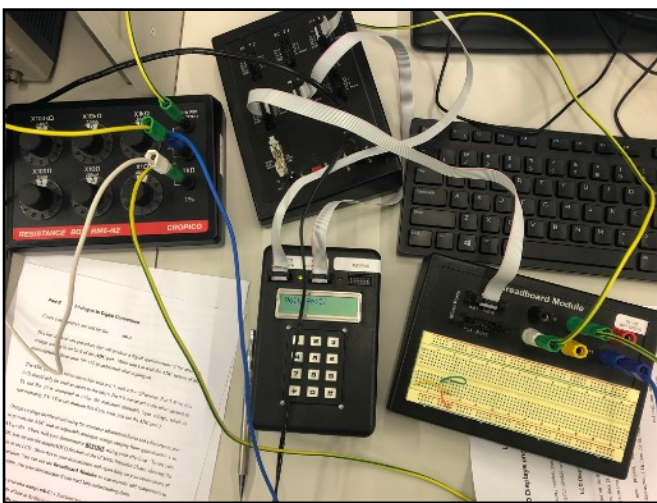


Fig. 7

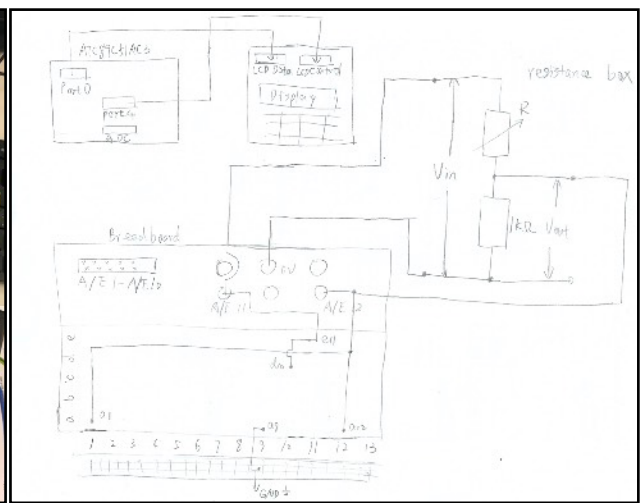


Fig. 8

I wrote my own sampleADC() function, instead of simply using sampleADC() in the library. Which shown in figure 9. Figure also include main function.

```

unsigned char Sample_ADC()
{
    static unsigned char Sample;
    ADCF = 0x01;
    ADCON = 0x20; // Enable the ADC

    ADCON &= 0xF8;

    ADCON |= 0x00;
    ADCON |= 0x08;
    // Wait flag end of conversion
    while ((ADCON & 0x10) != 0x10) {}
    /* Do Nothing */
    ADCON &= 0xEF;
    Sample = (ADCH << 2) + (ADCL);
    return Sample;
}

/** Main Function *****/
void main()
{
    initLCD();
    while (1)
    {
        clearLCD();
        val = Sample_ADC();
        sprintf(outputText, "Voltage: %i", val);
        writelineLCD(outputText);
        delay(30000);
    }
}

```

Fig. 9



R (variable) ohm	Vout V	Digital number (LCD)
0	3	255
...	...	...
77	...	254
...	...	...
436	2.22	0
437	...	255
...	...	...
1180	...	0

Table 1

To observed how ADC ran, I recorded some data which shown in table 1.

The digital number had range from 0-255, which would repeated after the number ran out of.

The variable R needed to increase 436 ohm to decrease digital number into 0. However, when the digital number repeated again, resistance needed to alter 743 ohm to reach same outcome.

Which means the relationship between value of R and value of Digital number is not linear.

### Temperature measurement

In this lab session, design a temperature logger.

First, tested the thermistor. When left the sensor under room temperature (21°C), resistance = 10.79k ohm; when held by finger, the temperature increased, resistance dropped to 7.50k ohm. When finger released, R backed to the previous value.

Wire and equipment were set up as in figure 10.

Base on previous lab session, I replaced variable resistance into thermistor, and replaced fixed resistance into 12k ohm resistance.

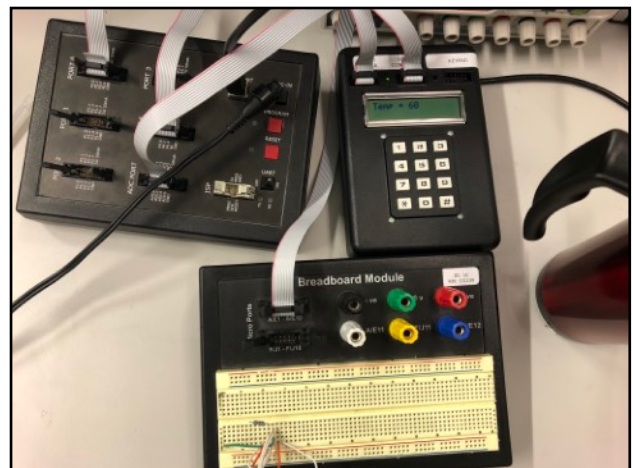


Fig. 10

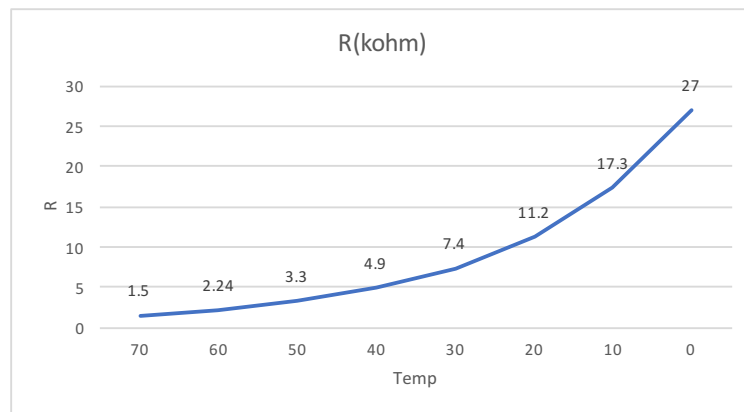


Fig. 11

By measuring thermistor's resistance and corresponding temperature, drew a graph shown in figure 11.

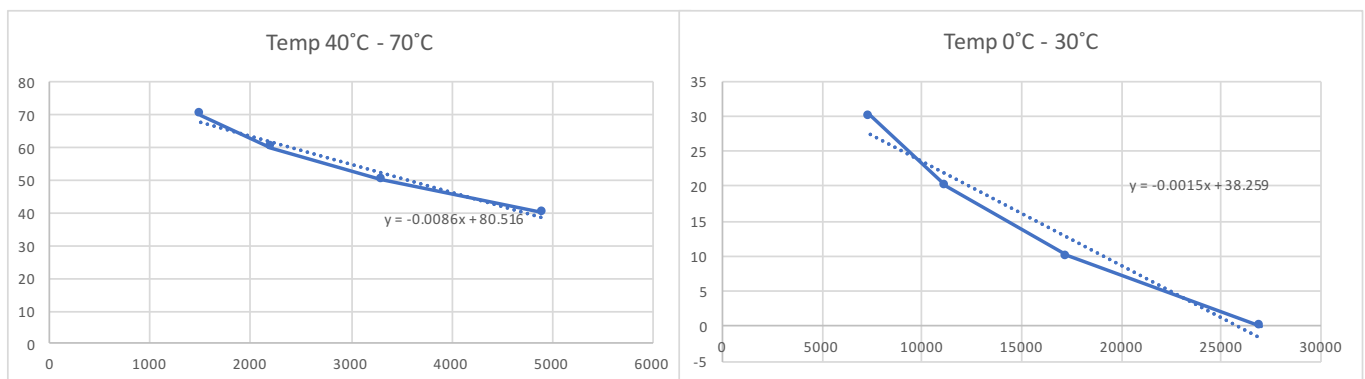


Fig. 12

Break the curve in figure 11 into two regions, and generate equation for both curve:  $y = -0.0086x + 80.516$ ,  $y = -0.0015x + 38.259$ . which shown in figure 12.

These two equation would be used for writing code.

```

ECEN301LibSDCC1.lib  main.c  X
#include "AT89C51AC3.h"
#include "ECEN301LibSDCC1.h"
#include <string.h>
#include <stdio.h>
char outputText [33];
char str[15];
int val;
int Temp;
float fval = 0.0;
unsigned char Sample_ADC()
{
    static unsigned char Sample;
    ADCF = 0x01;
    ADCON = 0x20; //enable the adc
    ADCON &= 0xF8;
    ADCON |= 0x08;
    while ((ADCON & 0x10) != 0x10)
        /* Do Nothing */;
    ADCON &= 0xEF;
    Sample = (ADCH<<2)+(ADDL);
    return Sample;
}

void main(void)
{
    // Initialise Display
    initLCD();

    /* Replace with your application code */
    while (1)
    {
        clearLCD();
        //val = Sample_ADC();
        val = sampleADC();
        fval = (float)val;

        float vref = fval*0.003;
        float R1 = 12000*(3.3/vref)-12000;

        if(val > 680){
            Temp = (-0.0006*R1)+80.516;
        }
        else{
            Temp = (-0.0015*R1)+38.259;
        }

        sprintf(outputText,"Temp = %i", Temp);
        writeLCD(outputText);
        delay(30000);
    }
}

```

Fig. 13

Code shown in figure 13 could achieve function of temperature measurement. Here was an issue I met: when I defined a *float fval* without assigning value, compiler would report an error. So I had to assign an initial value to *fval*.

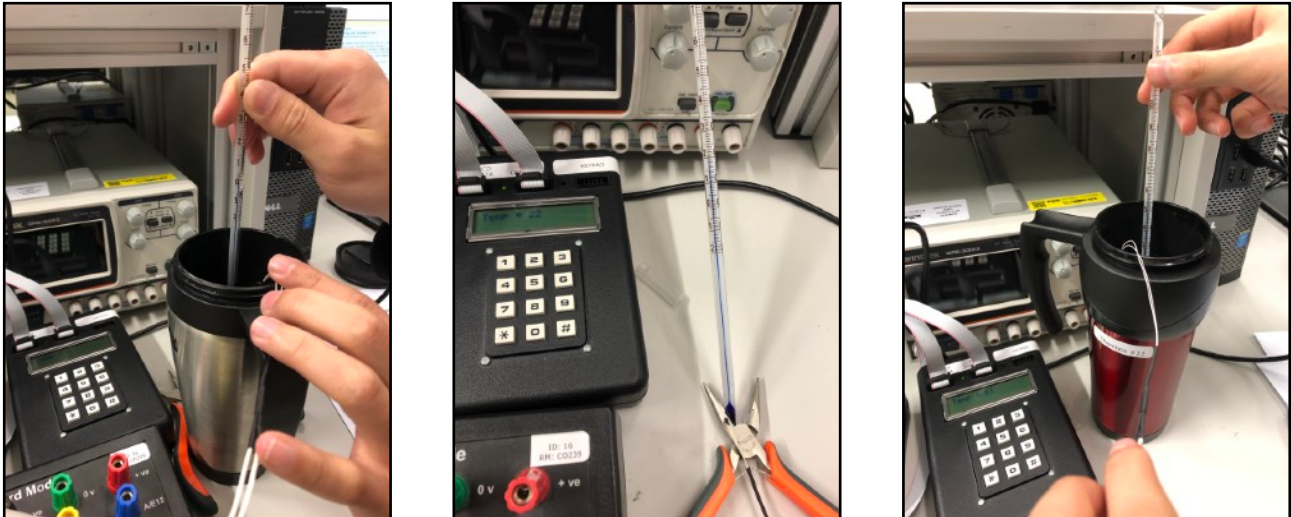


Fig. 14

To display the standard deviation and mean of readings, I modified the code.

Used a `getTemp()` function to change Analogue to digital. I multiplied 100, then over 100 at last, is because the accuracy requirement.

More details shown in figure 15.

Fig. 15

Figure 16 showed display result of STD and mean. Under room temperature condition, the STD = 2. Which means the temperature did not much change.

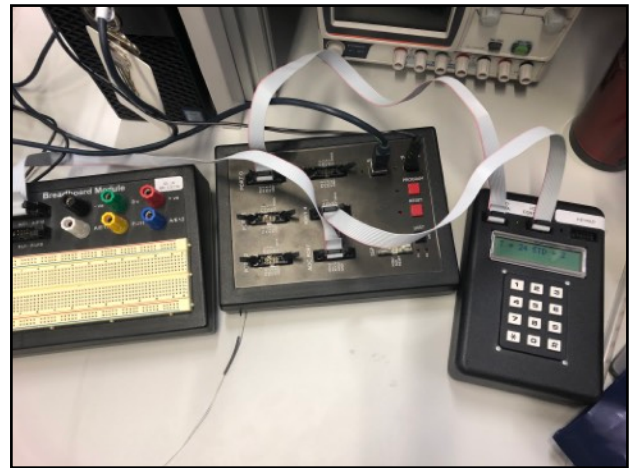


Fig. 16

To find maximum value of temperature, for loop can be used:

```
int maximum;  
int i;  
maximum = Temperature[0];  
for (i = 1; i < 5; i++) {  
    if (Temperature[i] > maximum){  
        maximum = Temperature[i];  
    }  
}
```

For minimum value:

```
int minimum;  
minimum = Temperature[0];  
for (i = 1; i < 5; i++){  
    if(Temperature[i] < minimum){  
        minimum = Temperature[i]  
    }  
}
```

With these two function, maximum and minimum value can be found, then display on LCD. However, the microcontroller doesn't have enough memory. If I add these two functions into my code, the file can not be created.

## Questions

### Q1: ADCON:

bits 0-2: Selection of channel to convert

bit 3: set to start an A/D conversion; clears by hardware after completion of the conversion

bit 4: set by hardware when ADC result is ready to be read. This flag can generate an interrupt. Must be cleared by software.

bit 5: set to enable ADC. Clear for standby mode (power dissipation 1uW)

bit 6: set to put in idle mode during conversion. Clear to cover without idle mode.

bit 7: no meaning

### ADCF:

bits 0-7: set to use P1.x as ADC input. Clear to use P1.x as standard I/O port.