Reporte Practica 5 - ADC Integration

Diseño en sistemas de chip

Diego Armando Limón de León A01638247

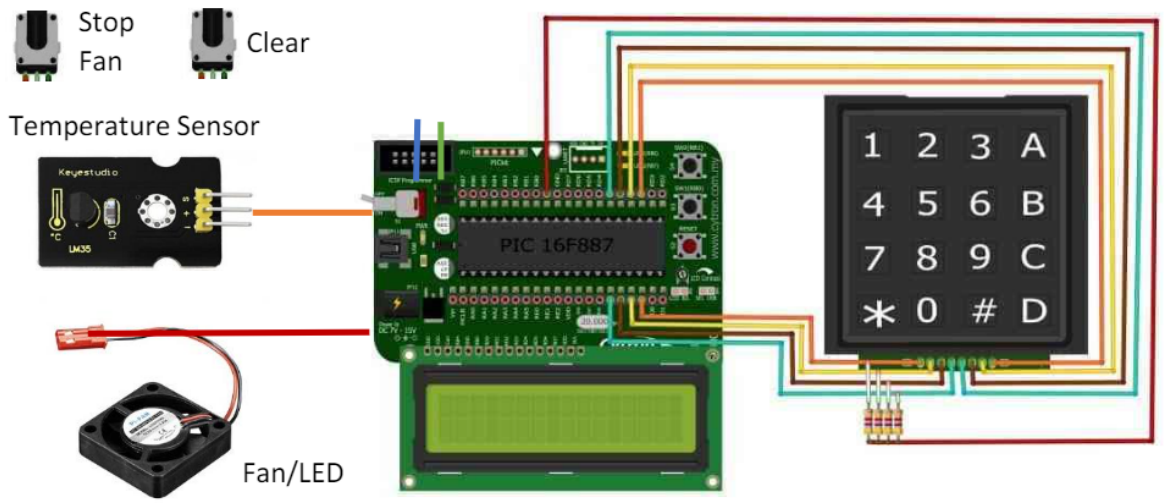
Jose Miguel Figarola Prado A01632557

Giancarlo Franco Carrillo A01638108

Tecnológico de Monterrey

xx de mayo del 2021

In this lab the idea is to integrate aspects of the four modules we have seen so far: LCD display, 4x4 matrix keyboard and ADC(and UART). Thus,before proceeding, be sure that your code for the LCD and the keyboard works properly. You can use either the 8-or 4-bit option for the LCD code.



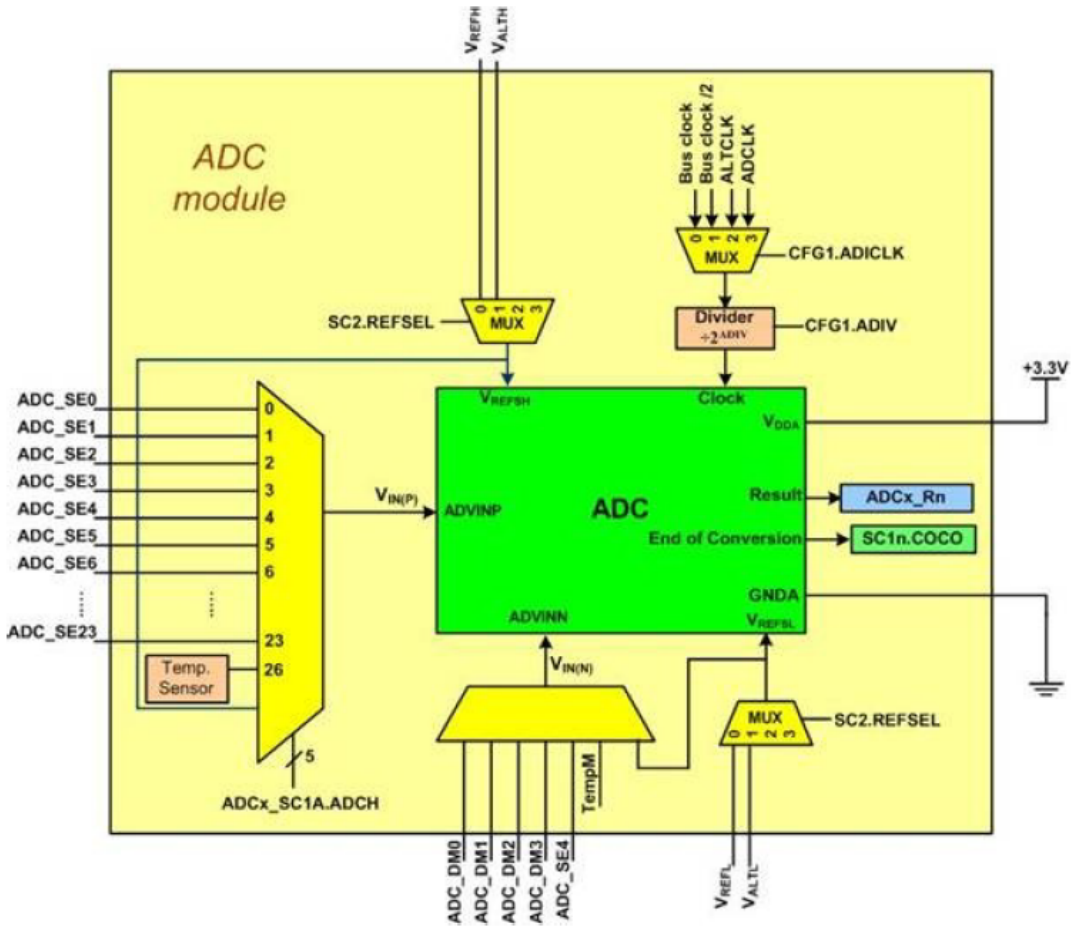
The lab will be divided into four parts, described as follows:

Part 1. ADC Part 1.

In this part, you should integrate the ADC and an internal sensor. We saw that the ADC has an internal temperature sensor attached to one of the channels of the ADC. The idea in this part of the lab is to retrieve the value through a program running in your MCU. Instead of using the LEDs as it was the case of the example seen in class, you need to convert the integer obtained from the ADC to display the temperature in the LCD screen through polling.

Analisis: In this part of the practice we need to show the internal temperature of the KL25z board on the LCD. As part of our debug we used another code seen in class where a

potentiometer controls a LED color output but now in the code the internal temperature controls the LED color output. So it changes the color depending on the temperature shown in the LCD. The channel must be linked to the 26th pin where you can find the internal temperature sensor. as shown in the block diagram below.



The first lines of the code, we begin by including the TextLCD module used in other practices to initialize the LCD and its functions. By lines 12 to 15 we introduce the functions that will be used for this project and will be described later in this section. Basically we get the result from the ADC converter (line 25) and we display it on the LCD (line 29), for aesthetic purposes and because it is not necessary to check the temperature every now and then, we add a delay by line 30.

```
#include "mbed.h"
#include "TextLCD.h" //Peter Dresher 2010 RW

TextLCD lcd(PTD0, PTD2, PTD3, PTD4, PTD5, PTD6, PTD7); // rs, rw, e, d4-d7
void ADC0_init(void);
void LED_set(int s);
void LED_init(void);
void delayMs(int n);

int main (void)
{
    short int result;
    LED_init(); /* Configure LEDs */
    ADC0_init(); /* Configure ADC0 */
    while (1) {
        ADC0->SC1[0] = 26; /* start conversion on channel 26 temperature */
        while(!(ADC0->SC1[0] & 0x80)) { } /* wait for COCO */
        result = ADC0->R[0]; /* read conversion result and clear COCO flag */
        LED_set(result); /* display result on LED */
        lcd.cls();
        lcd.locate(0,0);
        lcd.printf("%d\n",result);
        delayMs(500);

    }
}

```

The ADCO_init function is in charge of the analog to digital converter.

```
void ADC0_init(void)
{
    SIM->SCGC6 |= 0x8000000; /* clock to ADC0 */
    ADC0->SC2 &= ~0x40; /* software trigger */
    /*CLKDIV/4, LS time, single ended 12 bit, bus clock */
    ADC0->CFG1 = 0x40 | 0x10 | 0x04 | 0x00;
}
```

LED_init function is in charge initializing the internal LEDs that will be used in the code when the data is received.

```
void LED_init(void)
{
    SIM->SCGC5 |= 0x400; /* enable clock to Port B */
    SIM->SCGC5 |= 0x1000; /* enable clock to Port D */
    PORTB->PCR[18] = 0x100; /* make PTB18 pin as GPIO */
    PTB->PDDR |= 0x40000; /* make PTB18 as output pin */
    PORTB->PCR[19] = 0x100; /* make PTB19 pin as GPIO */
    PTB->PDDR |= 0x80000; /* make PTB19 as output pin */
    PORTD->PCR[1] = 0x100; /* make PTD1 pin as GPIO */
    PTD->PDDR |= 0x02; /* make PTD1 as output pin */
}
```

LED_set function receives the result from the ADC converter and by this given result we can manage to change the color from the internal LEDs. So every result gives a different color.

```
void LED_set(int s)
{
    if (s & 1) /* use bit 0 of s to control red LED */
        PTB->PCOR = 0x40000; /* turn on red LED */
    else
        PTB->PSOR = 0x40000; /* turn off red LED */

    if (s & 2) /* use bit 1 of s to control green LED */
        PTB->PCOR = 0x80000; /* turn on green LED */
    else
        PTB->PSOR = 0x80000; /* turn off green LED */

    if (s & 4) /* use bit 2 of s to control blue LED */
        PTD->PCOR = 0x02; /* turn on blue LED */
    else
        PTD->PSOR = 0x02; /* turn off blue LED */
}
```

Finally the delayMs function uses the internal clock to ironically make a delay.

```
void delayMs(int n)
{
    int i;
    SysTick->LOAD = 41940 - 1;
    SysTick->CTRL = 0x5; /* Enable the timer and choose sysclk as the clock source */

    for(i = 0; i < n; i++) {
        while((SysTick->CTRL & 0x10000) == 0)
            /* wait until the COUTN flag is set */
        {
        }
    }
    SysTick->CTRL = 0;
    /* Stop the timer (Enable = 0) */
}
```

Part 2. ADC Part 2:

Simple thermostat. As with the previous part, we will be using the internal ADC, but this time connecting a sensor. It will be optimal if you use something similar to the LM35 but any other sensor is fine (even a potentiometer). We will be using the keyboard, LCD, timers, interrupt and UART. The code in this part should proceed as follows:

1. When you start your application, the next message should be displayed.

Set the desired temperature

2. As with the previous code you should be able to read the temperature from the keyboard and display it on the LCD. Then, “set” the “air conditioner” (a LED, but if you have a simple fan it would be nice). In order to start sensing the temperature (using the ADC) you should press the # or * button. The fan is activated for a predefined time (let say 10 seconds, for demonstration purposes)using a counter, and re-activated every minute to “maintain the temperature” (this is how these systems usually work)Temperature SensorFan/LEDStopFanClear

3. The message should be displayed for the normal execution of the program (only changing the values of the ADC), and the system should keep activating the fan (or LED). However,if the value surpasses the desired temperature(by 5 degrees let’s say), the system should activate another output (“alarm”) and activate the fan for a longer time (2 mins). After this

time, if the temperature is still higher than the set temperature, it should go again for two minutes. Otherwise, it moves to normal mode.

4. The fan can be put in idle mode at any moment through a pushbutton via an interrupt, as in the previous lab. The idea is that you still show the temperature but you don't activate the fan in this mode.Whenever you want, press the # button in the keyboard to resume.

5. Reset the system with a second button and an interrupt, in which case the code should go to the first step and ask again for the temperature.

Análisis: In this part it is required to connect a DHT11 sensor to obtain temperature and humidity as input. The user must be able to set the desired temperature by displaying a message on the LCD as the main menu. The code should be able to control the temperature by using a fan or a LED. IF the temperature surpases te disired one, an alarm must be triggered.

First of all we initialize the libraries required, the variables and functions there are going to be used within the code, also it is declared an object to control the fan but it requires a specific controller, so for demonstration purposes the program turns on the blue led of the board to simulate the fan.

```
//code implementation for part 2
#include "mbed.h"

#include "TextLCD.h" //library to lcd with rw
#include "Keypad.h" //library to keypad
#include "DHT11.h" //library to control tmp sensor 2018
#include "DcFan.h" //library to control fan cooler

#define c 262
#define d 294
//buttons_init
DigitalIn B(PTA1);
DigitalIn G(PTA2);

//initialize sensors and actuators
TextLCD lcd(PTD0, PTD2, PTD3, PTD4, PTD5, PTD6, PTD7); // rs, rw, e, d4-d7
Keypad Kpad(PTC5, PTC6, PTC10, PTC11, PTC7, PTC0, PTC3, PTC4); // col 1-4, row 1-4
DHT11 DHT(PTA4); //DHT11 sensor init
DcFan myfan(PTA13, 1.0); //fan init port and activepwm
PwmOut buzzer(PTA12); //buzzer init
void LED_init(void);
//various functions
int isA_number(char key);
bool keyflag(char key);
void setTMP(void);
void alert1(void);
//time functions
void timer(int count,char type);
void delayMs(int n);

int data; //global variable for sensor data
```

The main function clears the variable and calls the setTPM() function that is in charge of setting the desired temperature.

```
int main() {
    LED_init();

    while(1){
        //reset all variables
        lcd.cls();
        pos = 0;
        temp = 0;
        key = '\0';
        setTMP(); //call the function for setting temperature
        delayMs(500);
        data = DHT.readData();
```

```
123 void setTMP(void){
124      //ask user to set ideal temperature
125     lcd.locate(0,0);
126     lcd.printf("Set the desired temperature:");
127     delayMs(20);
128      while(key != '#'){ //set # as enter key
129         delayMs(50);
130         key = Kpad.ReadKey();
131         delayMs(20);
132         num = isA_number(key);
133         delayMs(20);
134         //we do this code only if there is a numerical char
135         if(num != -1 && keyflag(key)) { //condition + flag
136             temp = temp*10 + num;
137             lcd.locate(12+pos,1);
138             lcd.printf("%d \n", num); //display the seconds
139             pos += 1; //get to the next position
140         }
141     }//we have now set the ideal temperature
142 }
```

As stated before, it is important to check the ideal temperature and while it is on the ideal temperature it is important to keep it like that so we turn on the red LED in order to do that.

```
        if(data != DHT11::OK){ //verify connection
            lcd.cls();
            lcd.locate(0,0);
            lcd.printf("Error! \r\n");
        }
        else{//chek ideal temp in a range of -+5 degrees
            while(temp+5>DHT.readTemperature() && temp-5<DHT.readTemperature()){
                lcd.cls();
                delayMs(20);
                lcd.locate(0,0);
                lcd.printf("T:%d H:%d",DHT.readTemperature(),DHT.readHumidity());
                delayMs(2000);
                timer(15,'t'); //wait one minute to activate fan (15 sec to demonstration)
                delayMs(500);
                if(G){//set ideal temperature, we should go out this loop
                    break;
                }
                else if(B){//no fan
                    lcd.cls();
                    delayMs(20);
                    lcd.locate(0,0);
                    lcd.printf("T:%d H:%d",DHT.readTemperature(),DHT.readHumidity());
                    delayMs(5000);
                }else{//keep temperature
                    PTB->PCOR |= 0x40000; /* turn on red LED */
                    myfan.speed(1.0); //fan at medium capacity
                    timer(5,'c'); //set cool for 5 sec
                    myfan.speed(0.0); //stop fan
                    PTB->PSOR |= 0x40000; /* turn off red LED */
                }
            }//we get out if temperature read by sensor is 5degrees over ideal temp
```

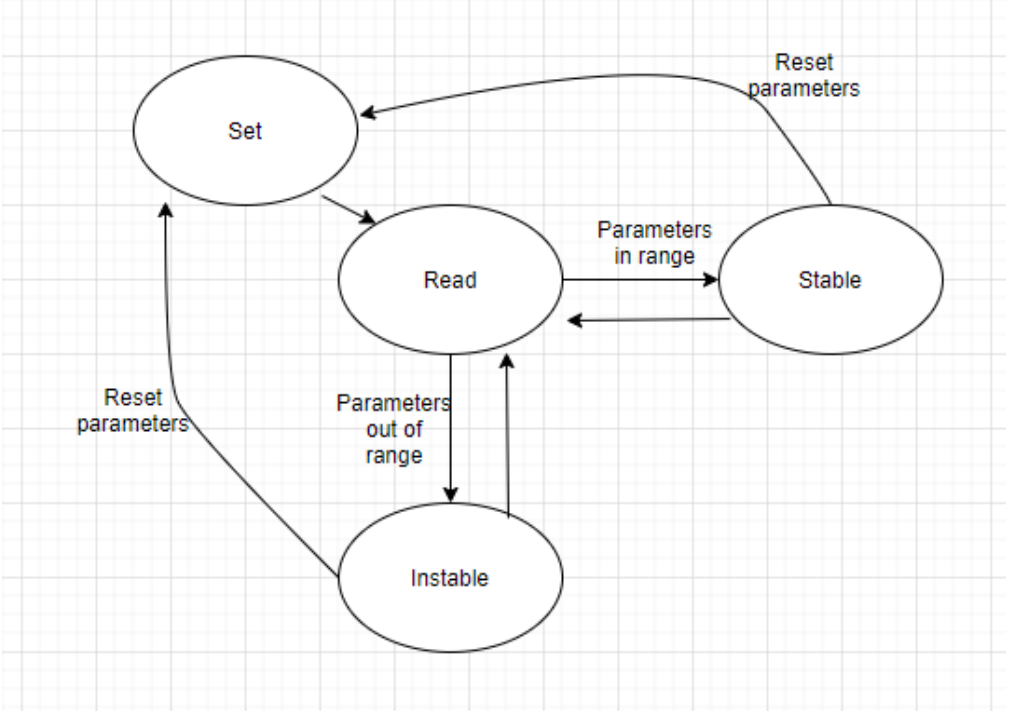
While the temperature is within +5 and -5 out of the ideal temperature, we set the required triggers to change the temperature and as stated before, an alarm is set with the alarm() function.

```
            alert1();//temperature out of range
            //cool until we get back in the range of ideal temp
            while(temp+5<=DHT.readTemperature() || temp-5>=DHT.readTemperature()){
                if(G){//set ideal temperature, we should go out this loop
                    break;
                }
                else if(B){
                    lcd.cls();
                    lcd.locate(0,0);
                    lcd.printf("T:%d H:%d",DHT.readTemperature(),DHT.readHumidity());
                    delayMs(2000);
                }else{
                    if(temp+5<=DHT.readTemperature()){ //heat secuence
                            PTB->PCOR |= 0x40000; /* turn on red LED */
                            timer(15,'h'); //set heat for 15 sec demonstration purposes
                            PTB->PSOR |= 0x40000; /* turn off red LED */
                    }
                    if(temp-5>=DHT.readTemperature()){//cool secuence
                        myfan.speed(1.0); //fan at medium capacity
                        timer(15,'c'); //set cool for 15 sec
                        myfan.speed(0.0); //stop fan
                    }
                }//else
            }//delicate
        }//sensor data fine
    } //go back no normal mode, first checking if sensor is fine
}

```

```
void alert1(void){//activate alarm
    buzzer.period(1/float(c));     // set PWM period
    buzzer = 0.5;                   // set duty cycle
    wait(1.0);
    buzzer.period(1/float(d));     // set PWM period
    buzzer = 0.5;                   // set duty cycle
    wait(1.0);
    buzzer = 0;
}
```

Finite State Machine



As shown in the FMS we have implemented interruptions by polling, we can reset values without resetting all the program, but also we are able to "pause" the fan (blue led of the board for demonstration purposes), we also added a "heating" sequence as to maintain the temperature if that's the case.

Video: For a better visualization of the activity, a Youtube video was made and you can see it in the following link: https://youtu.be/onxpZG0auUY

Requirements for the report 1.

Include the code for each of the functions of your code.The code should be commented and the report should include a short description of each function and how it works, including images of the registers that have been configured for enabling the different functionalities in your code(you should include the image of the registers as seen in class and in your comment code out which bits have been configured for certain purposes)

2. Provide a state machine or a flow diagram for the entire code(only applies for the second part)

3. For the last part, provide a schematic view of the connections of your design.4. Attach a short video demonstrating the system working. Alternatively, you can share the link to the video in your google drive for me to evaluate it.