EE419- Embedded Systems

Assignment # 2

Complex Engineering Activity

Section: EE-8A

Group Members

Aneeq Asghar	17L-4525
Muhammad Hasan	17L-4548

Department of Electrical Engineering
National University of Computer and Emerging Sciences, Lahore

Contents

List of Tables

List of Figures

5.1. Introduction

The problem was to design a home automation system using TM4C Controller. The system design has the ability to detect an intruder, abnormal temperature condition inside a house and any smoke that is hazardous to human life. It also observes the temperature and smoke in the surroundings and on detecting the abnormal activity, an alarm should be turned on for a few seconds to notify the resident. The alarm could be turned off through a PC.

5.1.1. Input and output devices

The I/O devices for our system are defined in the table below:

Device	Type	Purpose
TM4C123GH6PM	Input / Output	Interface sensors, process input signals and generate Output
Temperature sensor	Input	Monitor temperature continuously
Buzzer	Output	Alarms the user if abnormality in temperature or smoke is observed
Smoke sensor	Input	Monitor smoke continuously
Jumper wires	-	Connect input and output devices
Breadboard	-	Circuit connections

Table 1. I/O Devices

5.1.2. Technique for Interfacing of Components

Interfacing can be defined as transferring data between microcontrollers and interfacing peripherals such as sensors, microprocessors and motors. The TM4C microcontroller will be acting as the main micro processing unit for our system. It will be the central hub for taking inputs, processing them and sending outputs.

The Input/Output pins on the Texas Tiva microcontrollers have a wide range of alternative functions including:

- Analog to digital converter (ADC)
- Synchronous serial interface (SSI)
- Timer
- Pulse width modulation (PWM)
- Universal asynchronous receiver/transmitter (UART)

The sensors used in our system, temperature and smoke sensor that will be interfaced with the TM4C microcontroller. The connectivity of these sensors to the respective pins of the microcontroller and its output pins selected can be summarized below:

1. As the temperature sensor gives an analog output, it is connected to one of the analog to digital converter modules available, ADC input 0. This module is present at the port E, pin 3 of the microcontroller, and hence the temperature sensor will be connected here.

2. The smoke sensor gives an analog output. So, it is connected to one of the analog to digital converter modules available, ADC input 1. This module is present at the port E, pin 2 of the microcontroller, and hence the smoke sensor will be connected here.

3. We also wish to turn on an alarm if the temperature or the smoke exceeds the limit. The alarm is operated by sending pulses to it, and thus we can use one of the pulse width modulation modules, PWM0, to send waves of a certain frequency to it. We will be using generator A of PWM0 which is available on port C, pin 4 of the microcontroller, where the buzzer can be connected.

4. For displaying the status on the PC, we use the universal asynchronous receiver/transmitter, UART module 0. UART0 is connected to the computer through a USB port and its receiver (RX) and transmitter (TX) pins are available at port A pin 0 and 1 respectively.

5.1.3. Software Tools

- The KEIL MICROVISION 4 Code Interface was used for programming the microcontroller of the launchpad because not only it is easy to use but also assists in project structure with access to all included files and dependencies, source code editing, program debugging, run-time environment, monitoring peripherals that are currently simulated.

- The Tera Term software was used to communicate the launchpad TM4C with the PC for displaying the output of the processed input signal. Tera Term is an open source, terminal emulator, software implemented and supports serial port connections.

5.2. Sensor Selection

The sensors needed while designing the system are:

- Temperature sensor; used for monitoring the temperature
- Smoke sensor; used for monitoring the smoke

Since the temperature and smoke sensor gives an analogue output, it is connected to the analogue to digital converter pins of the TM4C123GH6PM.

5.2.1. Temperature Sensor:

Temperature of the surrounding can be measured through temperature sensors whose purpose is to continuously sense the temperature and generate an electrical signal. The conditions that must be fulfilled by the sensor includes operating range should be in between 3.3V to 5V, should be integrate-able with TM4C microcontroller, should give output either analog or digital signal. Comparison of different temperature sensors is as shown below:

Module	Range (º C)	Output type	Operating Voltage (V)	Cost/ Rs	Available
AD590	-55 - 100	Analog	4-30	780	No
MLX90614	-70 - 380	Digital	8-16	950	No
DHT-11	0-50	Digital	3 - 5	180	Yes
LM34	-55 - 150	Analog	3-30	180	Yes
DHT-22	-40 - 80	Digital	3.3 – 5.5	450	Yes

Table 2. Temperature Sensors Comparison

From comparison of different temperature sensors, we used LM34 because it is cost effective, easily available in the market, can be easily interfaced with 10- or 12-Bit ADC module and integrated with TM4C microcontroller. It gives the output temperature in ° F. It is preferred over DHT-11 because it measures the temperature range from 0 to 50° C as temperature in some parts of Pakistan ranges from 45 to 50 ° C which is normal in the respective place. Whereas, LM34 measures temperature from -55 to 150° C. DHT-11 only gives digital signal at output while LM34 gives analog and digital signal at output as well. Also, DHT-11 measures not only temperature but humidity level in the environment, on the other hand LM34 only measures temperature. That is why LM34 was used in the designed system.

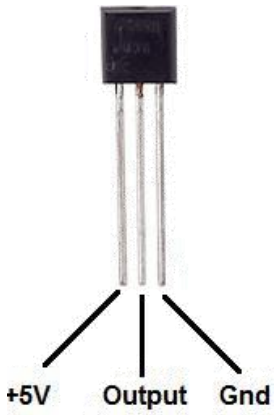


Figure 1. LM34 Temperature Sensor

5.2.2. Smoke sensor:

A smoke sensor is used to detect any kind of gas leakage such as methane or any kind of combustible gas leakage. A smoke detector would also give us an analog value of smoke like short circuit smoke or any kind of extreme level of smoke which is hazardous to human breathing. The sensor that we need to integrate in our system must be compatible with the TM4C microcontroller. Smoke of the surrounding can be measured through smoke sensors whose purpose is to continuously sense the smoke and generate an electrical signal. It should give output either analog or digital signal. The sensor also needs to be cost effective and should be easily available in the market. Its voltage consumption should be ideally 3.3V as the microcontroller works at this voltage. An external battery and voltage regulator would be needed, otherwise.

Keeping these specifications in mind, the possible types of smoke sensors are noted in the table below.

Sensor	**Operating temperature (º C)**	**Output type**	**Availability**	**Cost / Rs**	**Power supply / V**
TPS8804	-40 – 85	Analog	Yes	322.86	2.6 - 15.6
MQ-2	-40 – 85	Digital/ Analog	Yes	180	5

MQ-135	-40 – 85	Digital	Yes	90	5+

Table 3. Smoke Sensors Comparison

We selected TPS8804 as it provides interfacing with our TM4C123GH6PM and it provides us with a Analog output. Analog output can be converted to digital output using ADC built in our TM4C123GH6PM. We did not select MQ-2 or MQ-135 as both of them operate at a voltage of 5+ volts whereas, TM4C123GGH6PM operates at 3.3V.

TPS8804 Features:

- Photo Chamber AFE
  - Dual 8-bit programmable current LED drivers
  - Temperature compensation of LED current
  - Ultra-low offset op-amp for photodiodes
  - Programmable and by passable gain stage
- Carbon Monoxide Sensor AFE
  - Ultra-low offset gain stage
  - Programmable gain and reference
- Power Management
  - Programmable LDO for external microcontroller
- SLC interface transmitter and receiver
- Ultra-low power consumption
- I2C serial interface
- Wide input voltage range

Figure 2. TPS8804 Smoke Sensor

5.3. Design and Development of System

5.3.1. Block Diagram

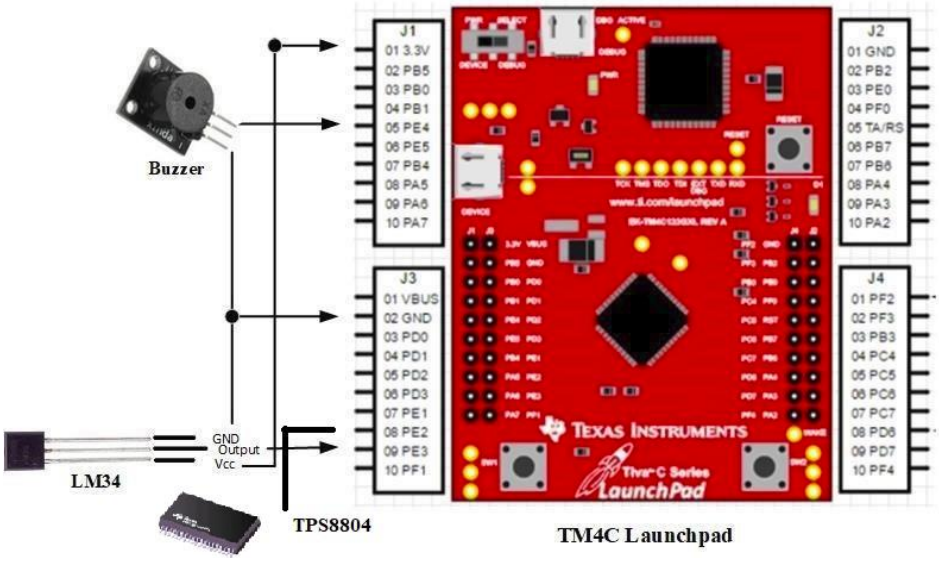


Figure 3. Block Diagram

5.3.2. Flow Chart of Software working

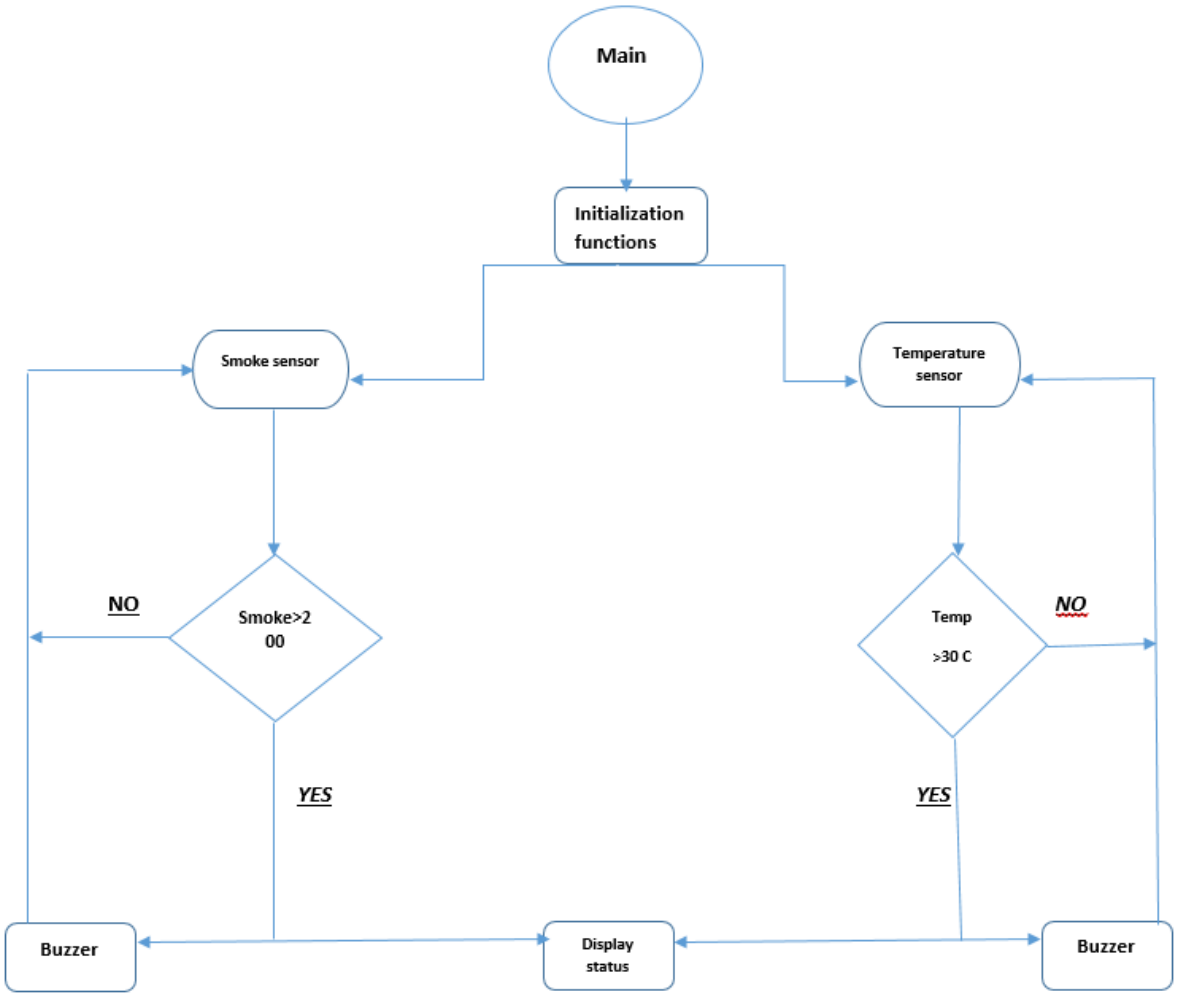


Figure 4. Flowchart

5.3.3. On-Chip Modules Used and their Programming

1. **UART0:**

To use UART0 module, we firstly need to enable port A. For this purpose, we firstly enable the clock, then configure port 0 and 1, and enable them as digital output pins. Finally, we enable their alternative functionality.

To configure UART0, firstly CTL register was cleared, to disable it for initialization, then IBRD and FBRD registers were set. For this the following calculations were done for 9600-baud rate:

$16MHz/16 = 1MHz$, $1MHz/9600 = 104.167$

Thus, $IBRD = 104$ (whole number part) $FBRD$ = (0.167 x 64) + 0.5 = 11(fraction part)

We also enable system clock and then set LCRH register value as 0x60 so that the input is 8-bit (since we are transmitting character by character, which are 8 bits), there is no parity, there is 1stop bit and there is no FIFO. Finally, we enable UART0, TXE and RXE.

2. **ADC0 and ADC1:**

Since ADC0 and ADC1 are used, we firstly need to enable port E, pin 3 for ADC0 and port E, PIN 2 for ADC1 respectively.

For this purpose, we firstly enable the clock and enable the pin for each module as digital output. We then enable its alternative functionality and use analog function as the alternate.

For enabling the ADC modules, firstly the ADC clock is enabled. Sequencer 3 was used, as only 1 sample was needed for each module. It was interrupt driven so ADCx_SSCTL3 was set to 6 and its interrupt was given a priority of 1.

ADC0Seq3_Handler() and ADC1Seq3_Handler() are called when an interrupt occurs.

3. **PWM:**

To use PWM we firstly need to enable port E, pin 4. For this purpose, we firstly enable the clock, configure it and enable the pin as digital output. We then enable its alternative functionality. We use PWM module 1 and its generator A for generating the pulses to run the buzzer. To do this, firstly, clock was enabled and then Load and CMPA were set using the following calculations, with a frequency of 1KHz:

T = $1 / 1KHz = 1msec$

Load= $1ms / 62.5ns$ = 16000

CMPA= (100% $Duty\ Cycle$) x LOAD = (100% − 50%)x16000 = 50% x16000 = 8000

4**. UART Interrupt:**

In order to turn on the buzzer with the PWM module unless the user turns it off using the PC, we used on-chip timer UART0_Handler interrupt. The action on calling this interrupt is performed internally without any external signal. The PWM0_3_LOAD_R register set to 16000, which was used for loading the value. PWM0_3_CPMA_R is set to 8000 if the masked interrupt status is not equal to zero. As soon as ‘A’ is pressed from the keyboard of the PC. Interrupt is cleared and the buzzer switches off by disabling the PWM.

5.3.4. Code

```
#include <stdio.h>
#include <stdint.h>
#include "C:\ti\TivaWare_C_Series-2.1.4.178\inc\tm4c123gh6pm.h"
#include <string.h>

float result_temperature;
float result_smoke;
char x;
volatile int a;

void EnableInterrupts(void);
void UART0_init(void);
char UART0Rx(void);
void UART0Tx(char c);
void sendString(char*ptr);
void adc0_seq3_init();
void ADC0Seq3_Handler();
void adc1_seq3_init();
void ADC1Seq3_Handler();
void portF_init();
void intinit(void);
void delayms(int n);
void PWM_init();
void GPIOPortF_Handler(void);
void UART0_Handler();


void EnableInterrupts(void)
{
  _asm("   cpsie i\n") ;
}
void UART0_init(void)
{
    SYSCTL_RCGCUART_R   |=  1;     /*   provide clock to UART0  */
    SYSCTL_RCGCGPIO_R  |=  0x01;    /*  enable clock to GPIOA   */

    /*  UART0 initialization  */
    UART0_CTL_R  =  0;              /* disable UART0  */
    UART0_IBRD_R =  104;          /*  16MHz/16=1MHz,  1MHz/104=9600 baud rate */
    UART0_FBRD_R =  11;          /*  fraction part   */
    UART0_CC_R      =   0;            /*  use system clock  */
    UART0_LCRH_R = 0x60;         /*  0x08-bit, no parity, 1-stop bit, no FIFO  */
    UART0_CTL_R  =  0x301;       /*  enable UART0, TXE , RXE   */

     /*  UART0 TX0 and RX0 use PA0 and PA1. Set them up.  */
```

```
    GPIO_PORTA_DEN_R  =  0x03;       /*  Make PAO and PAI as digital */
    GPIO_PORTA_AMSEL_R  &=  ~0x03;   //  No analog on PA1-0
    GPIO_PORTA_AFSEL_R  =  0x03;   /*  Use PA0,PA1 alternate fucntion  */
    GPIO_PORTA_PCTL_R  =  0x11;    /* configure PA0 and PA1 for UART  */


    NVIC_EN0_R  =0x20;
    NVIC_PRI1_R=0<<5;
    UART0_IM_R=0X10;
    EnableInterrupts();
}
char UART0Rx(void)
{
                while ((UART0_FR_R&0x0010)!=0) {} ;
                return ( (char) (UART0_DR_R&0xFF) ) ;
}
 void UART0Tx(char  c)
{
                while((UART0_FR_R &0x20)!=0){};
   UART0_DR_R  =  c;
}
void sendString (char*ptr)
{
        while (*ptr != 0)/* if not end of string */
        {
        UART0Tx(*ptr++); /* send the character through UART0 */
        }
}


void adc0_seq3_init() {
       /* enable clocks  */
     SYSCTL_RCGCGPIO_R|=0x10;
     SYSCTL_RCGCADC_R |=  1;       /* enable clock to ADC0 */

     GPIO_PORTE_AFSEL_R  |=0x08;
     GPIO_PORTE_DEN_R  &=~0x08;
     GPIO_PORTE_AMSEL_R  |=0x08;

      /* initialize ADC0  */
     ADC0_ACTSS_R   &=  ~0x08;        /* disable SS3 during configuration */
     ADC0_EMUX_R   &=  ~0xF000;

     ADC0_SSMUX3_R  =  0x00;         /* get input from channel  0  */  // AINO
     ADC0_SSCTL3_R   |=   0x06;       /* take chip result, set flag at 1st sample  */
```

```
    NVIC_EN0_R   |=0X00020000;
    ADC0_IM_R  |=  0x08;
    ADC0_ACTSS_R  |=  0x08;          /*enable ADC0  sequencer  3  */
    EnableInterrupts ();
}
void ADC0Seq3_Handler()
{
        result_temperature = ADC0_SSFIFO3_R * 330/4096;
        ADC0_ISC_R = 0x08;           /* clear completion flag*/
}
void adc1_seq3_init()
{
        SYSCTL_RCGCGPIO_R |= 0x10; /* enable clock to PORTE */
        SYSCTL_RCGCADC_R |= 2; /* Enable Clock to ADC0*/
        GPIO_PORTE_AFSEL_R |= 0x04;
        GPIO_PORTE_DEN_R &=~ 0x04;
        /* INITIALIZE ADC1 */
        ADC1_ACTSS_R &=~ 0x08;  /* Disable SS3 during configuration */
        ADC1_ACTSS_R &=~ 0xF000;

        ADC1_SSMUX3_R = 0x01;   /* Get input from channel 1 AIN1 */
        ADC1_SSCTL3_R = 0x06;   /* Take chip result, set flag at 1st sample */

        NVIC_EN1_R |= 0x00080000;
        ADC1_IM_R |= 0x08;
        ADC1_ACTSS_R |= 0x08; /* Enable ADC1 SS3 */

        EnableInterrupts();
}

void ADC1Seq3_Handler(){
        result_smoke = ADC1_SSFIFO3_R * 330/4096.0;
        ADC1_ISC_R = 0x08; /* Clear Completion Flag */
}

void portF_init()
{
        SYSCTL_RCGCGPIO_R |= 0x20; /* Enable Clock of PORTF */
        GPIO_PORTF_LOCK_R = 0x4C4F4348;
        GPIO_PORTF_CR_R = 0x10; /* Assigns pin to SSI2 */
        GPIO_PORTF_DEN_R = 0x1F;
        GPIO_PORTF_DIR_R = 0x0E; /* PF1,PF2, PF3 INPUT */
        GPIO_PORTF_PUR_R =0x10; /* PF4 input*/
}

void intinit(void){
```

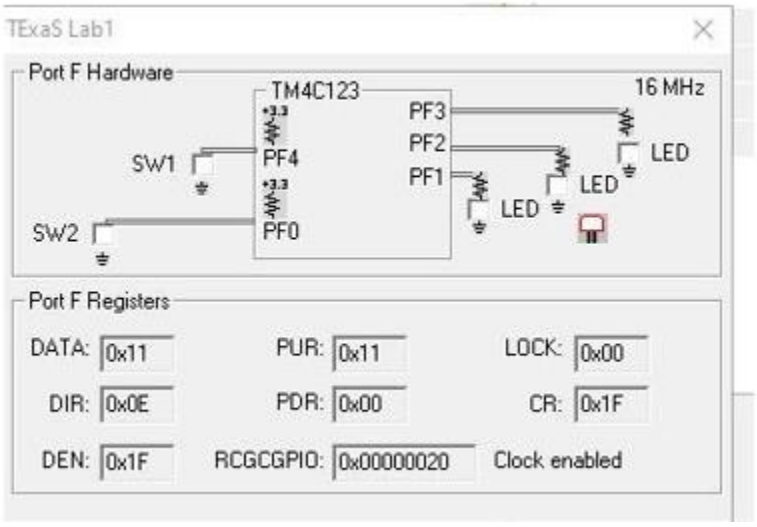
```
	GPIO_PORTF_IS_R &=~ 0x10;
	GPIO_PORTF_IEV_R &=~0x10;
	GPIO_PORTF_ICR_R |= 0x10;
	NVIC_EN0_R |= 0x40000000;
	NVIC_PRI7_R= 1<<21;
	GPIO_PORTF_IM_R = 0x10;
	EnableInterrupts();
}

void delayms(int n)
{
	int j;
	for (a=0;a<n;a++)
	{
		for (j=0;j<3180;j++){}
	}
}

void PWM_init()
{
	SYSCTL_RCGCPWM_R |= 1;  //ENABLE CLOCK FROM PWM0
	SYSCTL_RCGCGPIO_R |= 0x04; //ENABLE PORTC
	SYSCTL_RCC_R  &=~ 0x00100000; //NO PRE-DIVIDE FOR PWM CLOCK
	GPIO_PORTC_AFSEL_R = 0x10; // ENABLE ALTERNATE FUNCTION FOR PC4
	GPIO_PORTC_PCTL_R   &=~ 0x000F0000;
	GPIO_PORTC_PCTL_R    |=  0x00040000; /* ENABLE PWM FOR PC4 */
	GPIO_PORTC_DEN_R |= 0x10;

	PWM0_3_CTL_R = 0;
	PWM0_3_GENA_R = 0x0000008C;
	PWM0_3_CTL_R = 1;
	PWM0_ENABLE_R = 0x40;
	delayms(5000);
}

void GPIOPortF_Handler(void){
	volatile int readback;
	char* str = "INTRUDER ALERT : Press A to turn off the alarm";
	sendString(str);
	str="\n";

	PWM_init();
	PWM0_3_LOAD_R = 16000;
	while(GPIO_PORTC_MIS_R != 0)
	{
		delayms(1000);
```

```
                PWM0_ENABLE_R =0x00;
                GPIO_PORTF_ICR_R = 0x11;
                readback = GPIO_PORTF_ICR_R;
        }
}
void UART0_Handler(){
        volatile int readback;
        char c=UART0_DR_R;
        PWM0_ENABLE_R =0x00;
        a=1000;
        GPIO_PORTF_ICR_R = 0x10;
        readback = GPIO_PORTF_ICR_R;
}

int main(){

        portF_init();
        intinit();
        UART0_init();
        adc0_seq3_init();
        adc1_seq3_init();

        while(1)
        {
                char buffer[16];
                char *str="Room temperature is: ";
                char * degree ="Degree Celsius";
                char * str1 ="and the Smoke level is:  ";

                ADC0_PSSI_R |= 0x08;    /* Start a conversion sequence 3 */

                result_temperature = ADC0_SSFIFO3_R * 330/4096;
                /* clear completion flag */

                ADC1_PSSI_R |= 0x08;  /* START A CONVERSION SEQUENCE 3 */
                result_smoke = ADC1_SSFIFO3_R * 330/4096;
                /* clear completion flag */

                sendString(str);
                sprintf(buffer, "%f\r\n", (result_temperature-32) * 5/9 );
                sendString(buffer);
                sendString(str1);
                sprintf(buffer, "%f\r\n" , result_smoke);
                sendString(buffer);
                delayms(1000);
```

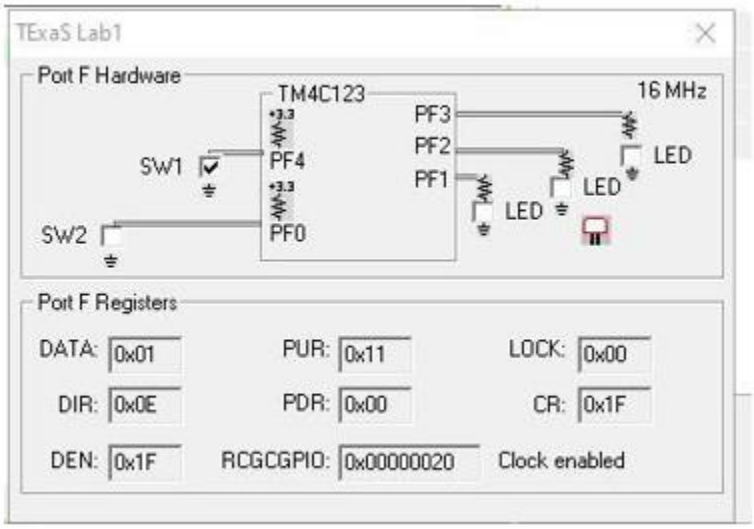
```
            if( ((result_temperature-32)* 5/9 ) > 30 ){
                PWM_init();
                PWM0_3_LOAD_R =16000;
                while(GPIO_PORTC_MIS_R != 0){
                    PWM0_3_CMPA_R = 8000;
                }
                delayms(1000);
                PWM0_ENABLE_R = 0x00;
            }

            else if (result_smoke >200 )
            {
                PWM_init();
                PWM0_3_LOAD_R =16000;
                while(GPIO_PORTC_MIS_R != 0){
                    PWM0_3_CMPA_R = 8000;
                }
                delayms(1000);
                PWM0_ENABLE_R = 0x00;
            }
        }
        return 0;
}
```

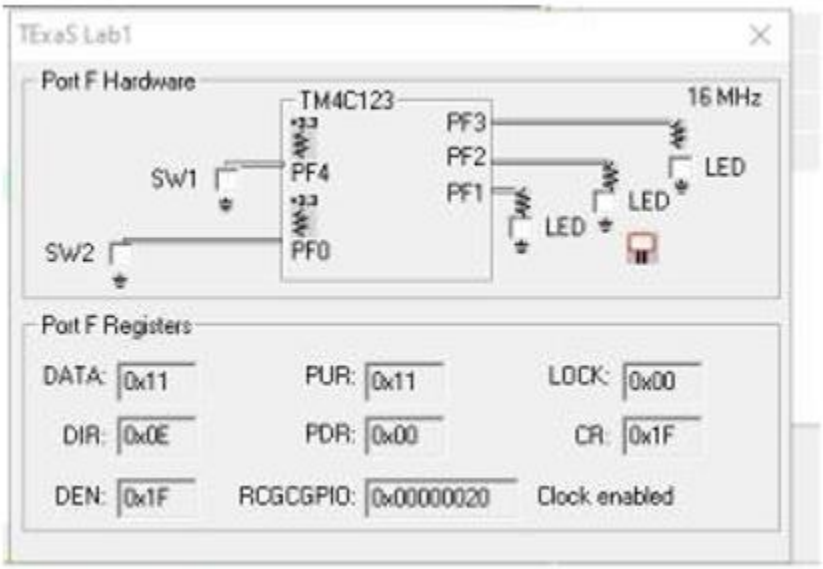
5.4. Simulation of System

- Initially:



- On smoke greater than 200 or temperature greater than 30-degree Celsius detection:




TExaS Lab1
Port F Hardware
TM4C123
16 MHz
SW1
PF4
SW2
PF0
PF3
PF2
PF1
LED
LED
LED
Port F Registers
DATA: 0x11
PUR: 0x11
LOCK: 0x00
DIR: 0x0E
PDR: 0x00
CR: 0x1F
DEN: 0x1F
RCGCGPIO: 0x00000020
Clock enabled

5.5. Limitations and Recommendations

No	Limitations	Recommendations
1	The smoke sensor has a specific range of operating	A sensor with a wider range of smoke detection should be used
2	The temperature sensor has a specific range for operating	A sensor with a wider range of temperature detection should be Used
3	The system is not persistent to hazardous conditions like; short circuit or water resistance	The components used must be persistent to such conditions
4	TPS8804 detects Carbon monoxide only.	The component must have the ability to detect all other gases that are hazardous to human life.

Table 4. Limitations and Recommendations

Cost Analysis

Components	Price (Rs)
Buzzer (KY006)	75
LM34	180
TPS8804	323
Total	**578**

Table 5. Cost Analysis