ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Output 2: Online Course for Microcontrollers: syllabus, open educational resources

Practice leaflet: Module_2-6 Communication - ADC

Lead Partner: International Hellenic University (IHU)

Authors: Theodosios Sapounidis [IHU], Aristotelis Kazakopoulos [IHU], Aggelos Giakoumis [IHU], Sokratis Tselegkaridis [IHU]

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

Declaration

This report has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

Funding Disclaimer

This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Executive summary

In this Module we will use PIC18F4550 with serial communication and Analog to Digital Converter.

Chapter 1: **Overview**

Table 1. Overview

Title / short summary	**6. Serial communication and Analog to Digital Converter**
Expected learning outcomes	• Students will be able to use two-way serial communication with the PIC18F4550 • Students will be able to design and implement simple circuits with serial communication • Students will be able to handle analog signals with the PIC18F4550 • Students will be able to design and implement simple circuits with analog signals • Students will be able to connect the PIC18F4550 to the TMP36 temperature sensor • The student will be able to load and animate a microcontroller program in the Proteus Design Suite
Keywords	Serial communication, ADC, TMP36
Duration	The duration of the module_2-6 is 3 hours • Presentation of the module_2-6 by the teacher, 30 minutes • 1st activity, Serial communication and LEDs, 50 minutes • 2nd activity, Analog to Digital Converter, 50 minutes • 3rd activity, Thermometer (TMP36), 50 minutes
Involved	**The teacher:** Presents the slides associated with the module_2-6 and answers question

	The students: Draw circuits in Proteus Schematic, write programs in C language, load programs to a microcontroller and run the simulation using the Proteus Design Suite
Assignment	At the end of the Module_2-6 will be given: • Open Project
Educational tools and equipment	• **Material:** PC • **Software:** CCS C compiler, Proteus Design Suite
Prerequisites / pre-existing knowledge	• The student must be familiarized with the Proteus Design Suite (link1) • The student must be completed Module_2-1, Module_2-2, Module_2-3, Module_2-4 and Module_2-5
Educational content	• CCS C Compiler manual (C Compiler Reference Manual) • MICROCHIP, PIC18F2455/2550/4455/4550 Data Sheet • Module_2-6 slides • Module_2-6 Evaluation leaflet • Module_2-6 Open project leaflet • Module_2-6 Programs, Schematic Proteus (Compressed folder)
Tips	***Tip1***. *The microcontroller and the monitor must have the same baud rate* ***Tip2***. *ADC resolution is 10 bit (number range 0~1023)*

	***Tip3**. The temperature of the TMP36 sensor is given by the formula* T (^{o}C) = (Vout (mV) – 500) / 10

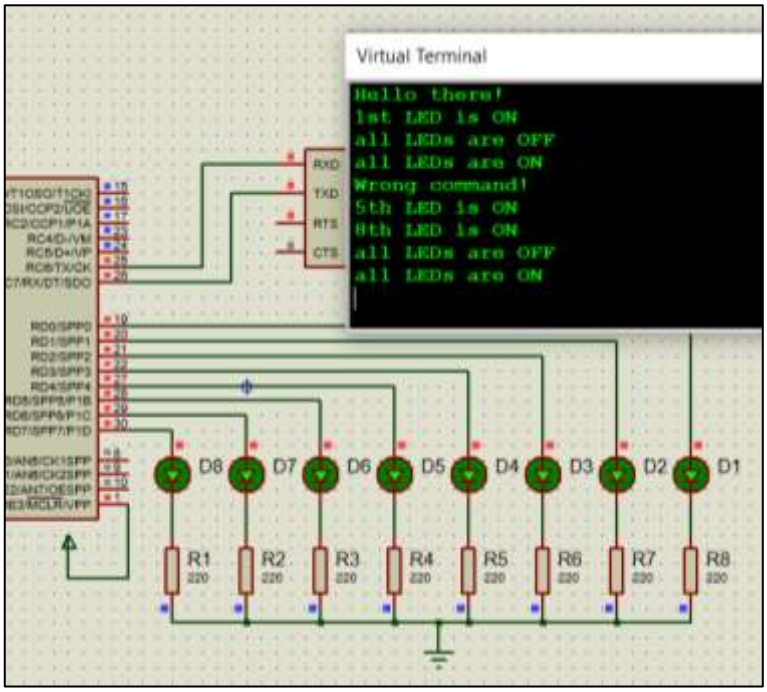
Chapter 2: **Activities**

2.1 Activity 1. Serial and LEDs

This activity uses serial communication between the PIC18F4550 and a serial monitor. 8 LEDs are connected to the PORTD.

Table 2. Activity 1

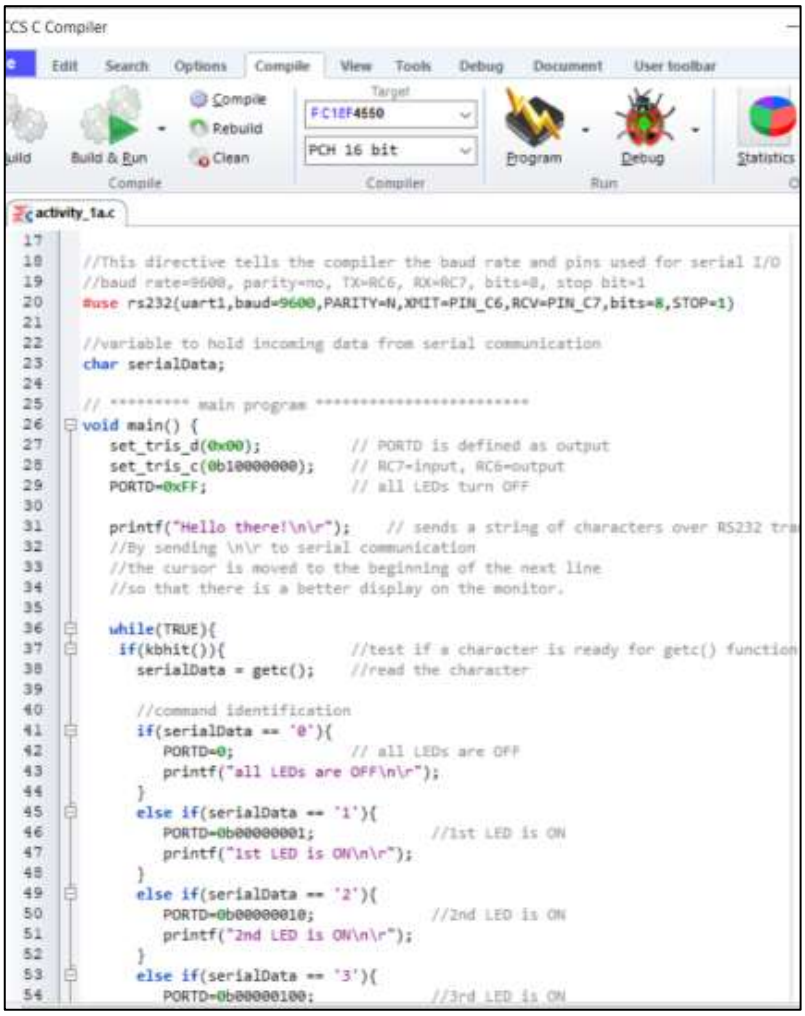
Activity 1[a] (30 minutes)	The PIC18F4550 reads the serial port and activates the appropriate LED. The commands are: • '0' => all LEDs turned OFF • '1' => 1st LED turned ON • '2' => 2nd LED turned ON • '3' => 3rd LED turned ON • '4' => 4th LED turned ON • '5' => 5th LED turned ON • '6' => 6th LED turned ON • '7' => 7th LED turned ON • '8' => 8th LED turned ON • '9' => all LEDs turned ON • Anything else is considered a wrong command **Step 1.** The circuit is drawn in the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code. **Step 4.** The machine code is loaded to the microcontroller. **Step 5.** The animation is activated.

Step 1 (10 minutes)	Draw the circuit of the picture in the Proteus Design Suite.  *Figure 1. Serial and LEDs*

Step 2
(10 minutes)
	Write in CCS C Compiler the program in C language
	
	`#include <main.h> // the file main.h with the`
	`// initial settings is included.`
	`// This file must be placed in the same`
	`// folder with the project.`
	`// Also the 18F4550.h file must exist`
	`// in the same folder with the project`
	
	`#byte PORTD =0xF83`
	`// We attribute to the memory position 0xF83 the name PORTD.`
	`// This means that we define a 8 bit variable whose value`
	`// will be stored to the memory position F83h.`
	`// The memory position F83h is the PORTD data register.`
	
	`#byte PORTC=0xF82 // F82h is the position or PORTC data register`
	`// at the data memory of the microcontroller`

```
                        //   SFR   Special   Function
 Register


//This directive tells the compiler the baud rate
and pins used for serial I/O

//baud rate=9600, parity=no, TX=RC6, RX=RC7, bits=8,
stop bit=1

#use rs232(uart1,baud=9600, PARITY=N, XMIT=PIN_C6,
RCV=PIN_C7, bits=8, STOP=1)


//variable to hold incoming data from serial
communication

char serialData;


// ********* main program ***********************

void main() {

   set_tris_d(0x00);          // PORTD is defined as
output

   set_tris_c(0b10000000);           //   RC7=input,
RC6=output

   PORTD=0xFF;                // all LEDs turn OFF


   printf("Hello there!\n\r");    // sends a string
of characters over RS232 transmission pin (TX)

   //By sending \n\r to serial communication

   //the cursor is moved to the beginning of the next
line

   //so that there is a better display on the
monitor.


   while(TRUE){

    if(kbhit()){              //test if a character
is ready for getc() function

      serialData = getc();    //read the character


      //command identification

      if(serialData == '0'){

         PORTD=0;             // all LEDs are OFF

         printf("all LEDs are OFF\n\r");

      }

      else if(serialData == '1'){

         PORTD=0b00000001;             //1st LED is
ON

         printf("1st LED is ON\n\r");

      }
```

```
        else if(serialData == '2'){
            PORTD=0b00000010;              //2nd LED is
ON
            printf("2nd LED is ON\n\r");
        }
        else if(serialData == '3'){
            PORTD=0b00000100;              //3rd LED is
ON
            printf("3rd LED is ON\n\r");
        }
        else if(serialData == '4'){
            PORTD=0b00001000;              //4th LED is
ON
            printf("4th LED is ON\n\r");
        }
        else if(serialData == '5'){
            PORTD=0b00010000;              //5th LED is
ON
            printf("5th LED is ON\n\r");
        }
        else if(serialData == '6'){
            PORTD=0b00100000;              //6th LED is
ON
            printf("6th LED is ON\n\r");
        }
        else if(serialData == '7'){
            PORTD=0b01000000;              //7th LED is
ON
            printf("7th LED is ON\n\r");
        }
        else if(serialData == '8'){
            PORTD=0b10000000;              //8th LED is
ON
            printf("8th LED is ON\n\r");
        }
        else if(serialData == '9'){
            PORTD=0xFF;                     //all LEDs
are ON
            printf("all LEDs are ON\n\r");
        }
        else{
            printf("Wrong command!\n\r");
        }
```

	} } }
Step 3 (4 minutes)	Compile the program in C in order to create the program in the microcontroller machine code (hex file).  *Figure 2. CCS C Compiler, translation to machine code (hex file)*
Step 4 (1 minute)	Load to the microcontroller the hex file (program in machine code) that was created from the CCS C Compiler.
Step 5 (5 minutes)	Run the simulation and check the correct operation of the circuit.

Activity 1[b] (20 minutes)	The PIC18F4550 reads the serial port and sets the frequency at which all LEDs flash. By default the frequency is 5Hz. The commands are: • 'a' => f=4Hz • 'b' => f=2Hz • 'c' => f=10Hz • 'd' => f=1Hz • Anything else => f=5Hz **** The circuit is the same as activity 1a **** **Step 1.** The program in C language is written. **Step 2.** The program is compiled with the use of CCS C compiler to the microcontroller machine code. The machine code is loaded to the flash memory of the microcontroller. **Step 3.** The animation is activated.

Step 1 (10 minutes)	Write in CCS C Compiler the program in C language
	
	`#include <main.h> // the file main.h with the`
	`// initial settings is included.`
	`// This file must be placed in the same`
	`// folder with the project.`
	`// Also the 18F4550.h file must exist`
	`// in the same folder with the project`
	
	`#byte PORTD =0xF83`
	`// We attribute to the memory position 0xF83 the name PORTD.`
	`// This means that we define a 8 bit variable whose value`

```
                        // will be stored to the memory
position F83h.
                        // The memory position F83h is
the PORTD data register.


#byte PORTC=0xF82    // F82h is the position or PORTC
data register
                        // at the data memory of the
microcontroller
                        //   SFR   Special   Function
Register


//This directive tells the compiler the baud rate
and pins used for serial I/O
//baud rate=9600, parity=no, TX=RC6, RX=RC7, bits=8,
stop bit=1
#use rs232(uart1, baud=9600, PARITY=N, XMIT=PIN_C6,
RCV=PIN_C7, bits=8, STOP=1)


//variable to store incoming data from serial
communication
char serialData;
//variable to store the time (ms) of the pulses
//Toff=Ton. By default f=5Hz => T=0.2s => Ton=100ms
int Ton=100;


// ********* main program ************************
void main() {
   set_tris_d(0x00);          // PORTD is defined as
output
   set_tris_c(0b10000000);           //   RC7=input,
RC6=output
   PORTD=0xFF;                // all LEDs turn OFF


   printf("Hello there!\n\r");    // sends a string
of characters over RS232 transmission pin (TX)
   //By sending \n\r to serial communication
   //the cursor is moved to the beginning of the next
line
   //so  that  there  is  a  better  display  on  the
monitor.


   while(TRUE){
    PORTD=0x00;          //all LEDs turned OFF
    delay_ms(Ton);       //wait for Ton ms
```

```
        PORTD=0xFF;          //all LEDs turned ON
        delay_ms(Ton);      //wait for Ton ms | Toff=Ton

        if(kbhit()){              //test if a character
    is ready for getc() function
          serialData = getc();    //read the character



          //command identification
          if(serialData == 'a'){
             //f=4Hz => T=250ms => Ton=125ms
             Ton=125;
             printf("4Hz\n\r");
          }
          else if(serialData == 'b'){
             //f=2Hz => T=500ms => Ton=250ms
             Ton=250;
             printf("2Hz\n\r");
          }
          else if(serialData == 'c'){
             //f=10Hz => T=100ms => Ton=50ms
             Ton=50;
             printf("10Hz\n\r");
          }
          else if(serialData == 'd'){
             //f=1Hz => T=1s => Ton=500ms
             Ton=500;
             printf("1Hz\n\r");
          }
          else{
             printf("Wrong command!\n\r");
             //f=5Hz => T=0.2s => Ton=100ms
             Ton=100;
             printf("5Hz\n\r");
          }
        }
      }
    }
```

Step 2 (5 minutes)	Use the CCS C Compiler to translate the programm from C language to the microcontroller machine code. Load to the microcontroller the hex file (machine code) that was created from the CCS Compiler.
Step 3 (5 minutes)	Run the simulation and check the correct operation of the circuit.

2.2 Activity 2. Analog to Digital Converter

The purpose of this activity is for the microcontroller to use the built-in Analog to Digital Converter.

Table 3. Activity 2

Activity 2[a] (30 minutes)	The PIC18F4550 • read the analog voltage of a potentiometer and convert it to a value (0~1023) • display the value on a LCD **Step 1.** The circuit is drawn in the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code. The machine code is loaded to the flash memory of the microcontroller. **Step 4.** The animation is activated.
Step 1 (10 minutes)	Draw the circuit of the picture at the Proteus Design Suite.

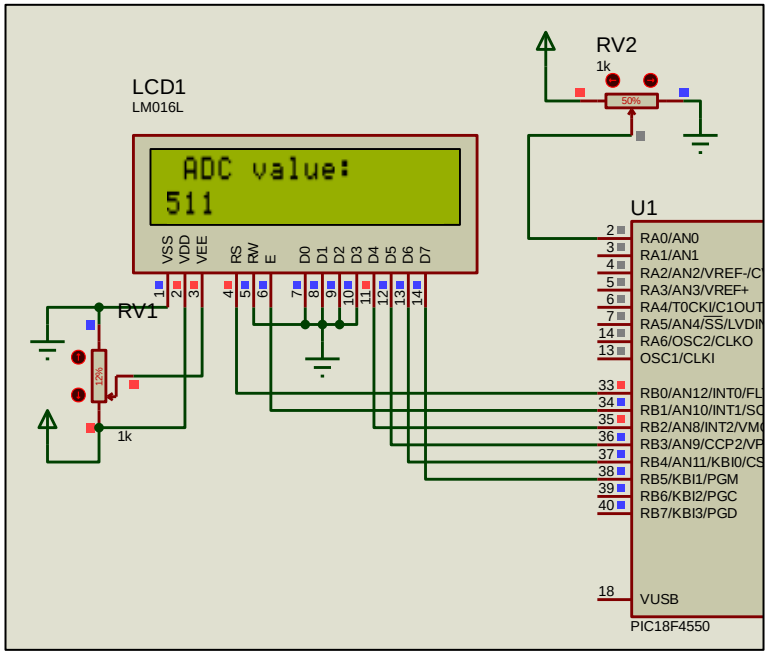


Figure 3. ADC and LCD 16x2

***Tip**. The MCLR pin must go to Vcc.*

Step 2
(10 minutes)

Write in CCS Compiler the program in C language

```
#include <main.h> // the file main.h with the
                  // initial settings is included.
                  // This file must be placed in
the same
                  // folder with the project.
                  // Also the 18F4550.h file must
exist
                  // in the same folder with the
project

#include <flex_lcd.h> // The h file of the lcd
driver
                      // should be in the same
folder where we will save our program.
                      // The #define LCD_DB4
PIN_B4 etc statements in flex_lcd.c
                      // should be checked and
possibly modified.
                      // These statements
determine the pins of the microcontroller
```

```
                        // that are connected to LCD
16x2.


#byte PORTB =0xF81
                       // We attribute to the memory
position 0xF81 the name PORTB.
                       // This means that we define
a 8 bit variable whose value
                       //  will  be  stored  to  the
memory position F81h.
                       // The memory position F81h
is the PORTD data register.



//ADC = 10 Bit => values: 0~1023
unsigned int16 ADC_value;


// ********* main program ************************
void main() {
   set_tris_b(0x00);        //PORTB is defined as
output
   lcd_init();           //initialization routine
for the LCD 16x2
   setup_adc(ADC_CLOCK_DIV_8);         // Set ADC
conversion time to 8Tosc
   setup_adc_ports(AN0);             // Set RA0 as
analog pin
   set_adc_channel(0);                   // Select
channel 0 (analog input 0)
   while(TRUE){
      delay_ms(1000);             //wait for 1 sec
      lcd_putc("\f");           //clear the screen
      lcd_putc(" ADC value:");    //send a message
to the LCD
      lcd_gotoxy(1,2);            //first position
of second line
      ADC_value=read_adc();      //read value from
ADC
      printf(lcd_putc,"%Lu",ADC_value);     //send
adc value to the LCD
   }
}
```

Step 3 (5 minutes)	Use the CCS C Compiler to translate the programm from C language to the microcontroller machine code. Load to the

	microcontroller the hex file (machine code) that was created from the CCS Compiler.
Step 4 (5 minutes)	Run the simulation and check the correct operation of the circuit.

Activity 2[b] (20 minutes)	The PIC18F4550 • read the analog voltage of a potentiometer and convert it to a value (0~1023) • display the value of analog voltage on a LCD **** The circuit is the same as activity 2a **** **Step 1.** The program in C language is written. **Step 2.** The program is compiled with the use of CCS C compiler to the microcontroller machine code. The machine code is loaded to the flash memory of the microcontroller. **Step 3.** The animation is activated.

Step 1
(10 minutes)
	Write in CCS Compiler the program in C language
	
	`#include <main.h> // the file main.h with the`
	`// initial settings is included.`
	`// This file must be placed in the same`
	`// folder with the project.`
	`// Also the 18F4550.h file must exist`
	`// in the same folder with the project`

```
#include <flex_lcd.h> // The h file of the lcd driver
                      // should be in the same folder where we will save our program.
                      // The #define LCD_DB4 PIN_B4 etc statements in flex_lcd.c
                      // should be checked and possibly modified.
                      // These statements determine the pins of the microcontroller
                      // that are connected to LCD 16x2.


#byte PORTB =0xF81
                     // We attribute to the memory position 0xF81 the name PORTB.
                     // This means that we define a 8 bit variable whose value
                     // will be stored to the memory position F81h.
                     // The memory position F81h is the PORTD data register.


unsigned int16 ADC_value; //ADC = 10 Bit => values: 0~1023
float voltage; //variable to store the potentiometer's analog voltage


// ********* main program ************************
void main() {
   set_tris_b(0x00);      //PORTB is defined as output
   lcd_init();           //initialization routine for the LCD 16x2
   setup_adc(ADC_CLOCK_DIV_8);         // Set ADC conversion time to 8Tosc
   setup_adc_ports(AN0);             // Set RA0 as analog pin
   set_adc_channel(0);                  // Select channel 0 (Analog input 0)
   while(TRUE) {
      delay_ms(1000);          //wait for 1 sec
      lcd_putc("\f");          //clear the screen
      lcd_putc(" Voltage:");   //send a message to the LCD
```

	lcd_gotoxy(1,2); //first position of the second line
	ADC_value=read_adc(); //read value from ADC
	voltage = ((float)(ADC_value*5)/1024); //convert adc value to analog voltage
	printf(lcd_putc,"%f",voltage); //send a message to the LCD
	}
	}

Step 2 (5 minutes)	Use the CCS C Compiler to translate the programm from C language to the microcontroller machine code. Load to the microcontroller the hex file (machine code) that was created from the CCS Compiler.
Step 3 (5 minutes)	Run the simulation and check the correct operation of the circuit.

2.3 Activity 3. Thermometer (TMP36)

The purpose of this activity is for the microcontroller to use the built-in ADC and read the TMP36 sensor.

The PIC18F4550:

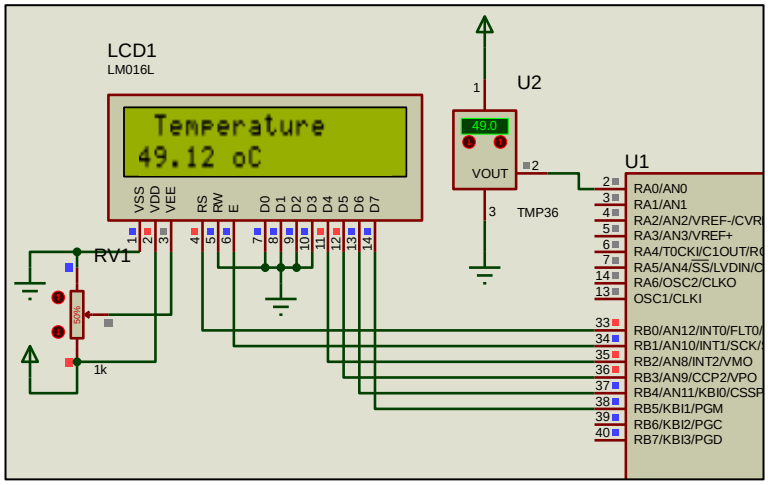
- reads the analog voltage of the sensor and converts it to a value (0~1023)
- converts the ADC value to analog voltage
- converts analog voltage to temperature (°C)
- displays the temperature on a LCD
- turns ON/OFF the 8 LEDs, according to the following table

Table 4. Temperature and LEDs

Temperature (°C)	Active LEDs
< 0	None
< 10	LED1
< 20	LED1 ~ LED2
< 30	LED1 ~ LED3
< 40	LED1 ~ LED4

< 50	LED1 ~ LED5
< 60	LED1 ~ LED6
< 70	LED1 ~ LED7
> 70	LED1 ~ LED8

Table 5. Activity 3

Activity 3rd (50 minutes)	**Step 1.** The circuit is drawn at the Proteus Design Suite. **Step 2.** The program in C language is written. **Step 3.** The program is compiled with the use of CCS C compiler to the microcontroller machine code (the hex.file is created). The program in machine code is loaded to the microcontroller. **Step4.** The animation is activated. **Step5.** Modification and discussion.
Step 1 (15 minutes)	Draw the circuit of the picture in the Proteus Design Suite.  *Figure 4(a). TMP36, LEDs and LCD 16x2*

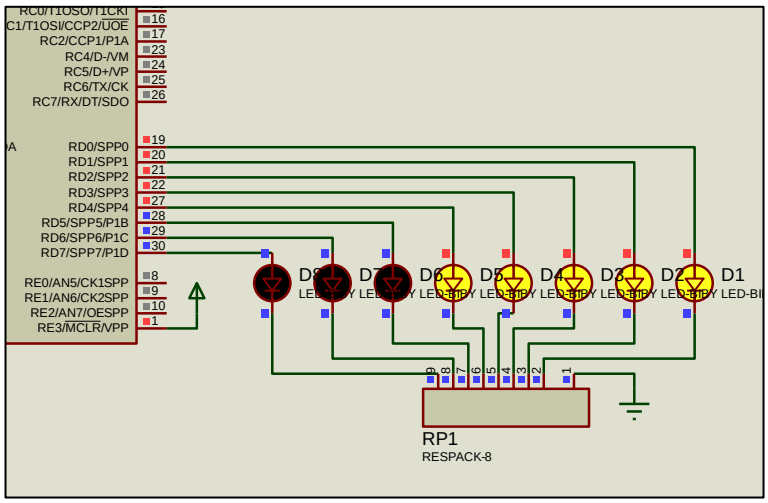


Figure 4(b). TMP36, LEDs and LCD 16x2

Step 2
(20 minutes)

Write in CCS C Compiler the program in C language

```
#include <main.h> // the file main.h with the
                  // initial settings is included.
                  // This file must be placed in
the same
                  // folder with the project.
                  // Also the 18F4550.h file must
exist
                  // in the same folder with the
project


#include <flex_lcd.h> // The h file of the lcd
driver
                      // should be in the same
folder where we will save our program.
                      // The #define LCD_DB4
PIN_B4 etc statements in flex_lcd.c
                      // should be checked and
possibly modified.
                      // These statements
determine the pins of the microcontroller
                      // that are connected to LCD
16x2.


#byte PORTB =0xF81
                     // We attribute to the memory
position 0xF81 the name PORTB.
                     // This means that we define
a 8 bit variable whose value
```

```
                        //  will  be  stored  to  the
memory position F81h.
                        // The memory position F81h
is the PORTD data register.


#byte PORTD =0xF83
                        // We attribute to the memory
position 0xF83 the name PORTD.
                        // This means that we define
a 8 bit variable whose value
                        //  will  be  stored  to  the
memory position F83h.
                        // The memory position F83h
is the PORTD data register.


unsigned int16 ADC_value; //ADC = 10 Bit => values:
0~1023
float voltage; //variable to store the sensor's
(TMP36) analog voltage
float temperature; //variable to store sensor's
temperature


// ********* main program ************************
void main() {
   set_tris_b(0x00);       //PORTB  is  defined  as
output
   set_tris_d(0x00);       //PORTD  is  defined  as
output
   lcd_init();            //initialization routine
for the LCD 16x2
   setup_adc(ADC_CLOCK_DIV_8);         //  Set  ADC
conversion time to 8Tosc
   setup_adc_ports(AN0);              // Set RA0 as
analog pin
   set_adc_channel(0);                    // Select
channel 0 (Analog input 0)
   while(TRUE) {
      delay_ms(1000);           //wait for 1 sec
      lcd_putc("\f");           //clear the screen
      lcd_putc(" Temperature"); //send a message
to the LCD
      lcd_gotoxy(1,2);            //first position
of the second line
      ADC_value=read_adc();     //read value from
ADC
      voltage    =    ((float)(ADC_value*5)/1024);
//convert adc value to analog voltage
```

```
        voltage*=1000; //convert V to mV
        temperature   =   (voltage-500)/10;//convert
sensor's voltage (mV) to temperature (oC)
        printf(lcd_putc,"%f       oC",temperature);
//send a message to the LCD


        //turn ON / OFF LEDs according to temperature
        if(temperature<0){
           PORTD=0b00000000;
        }
        else if(temperature<10){
           PORTD=0b00000001;
        }
        else if(temperature<20){
           PORTD=0b00000011;
        }
        else if(temperature<30){
           PORTD=0b00000111;
        }
        else if(temperature<40){
           PORTD=0b00001111;
        }
        else if(temperature<50){
           PORTD=0b00011111;
        }
        else if(temperature<60){
           PORTD=0b00111111;
        }
        else if(temperature<70){
           PORTD=0b01111111;
        }
        else{
           PORTD=0b11111111;
        }
    }
}
```

Step 3 (5 minutes)	Compile the program in order to create the hex.file (program in machine code). Load the program (hex.file) to the microcontroller.

Step 4 (5 minutes)	Run the simulation and check the correct operation of the circuit.
Step 5 (5 minutes)	Suggested modifications and discussion: • can the LCD show temperature in Kelvin and Celsius at the same time?

Chapter 3: **Recapitulation**

☞ The schematic of the circuits was drawn with Proteus Design Suite

☞ A serial communication and analog to digital converter were used to implement applications such as a simple thermometer with TMP36 sensor.

☞ The programs in C was written in CCS C compiler.

☞ The programs in C was compiled to the microcontroller machine code (hex file).

☞ The machine code was “loaded” to the microcontroller and the animation was activated.

References

CCS C Compiler Manual. Ccsinfo.com. (2021). Retrieved from https://www.ccsinfo.com/downloads/ccs_c_manual.pdf.

PIC18F2455/2550/4455/4550 Data Sheet. Ww1.microchip.com. (2006). Retrieved from https://ww1.microchip.com/downloads/en/devicedoc/39632c.pdf.

Proteus Tutorial : Getting Started with Proteus PCB Design (Version 8.6). Youtube.com. (2017). Retrieved from https://www.youtube.com/watch?v=GYAHwYUUs34.

Simple LED Circuits. Electronics Hub. (2017). Retrieved from https://www.electronicshub.org/simple-led-circuits/.

Appendix. Figures with high resolution

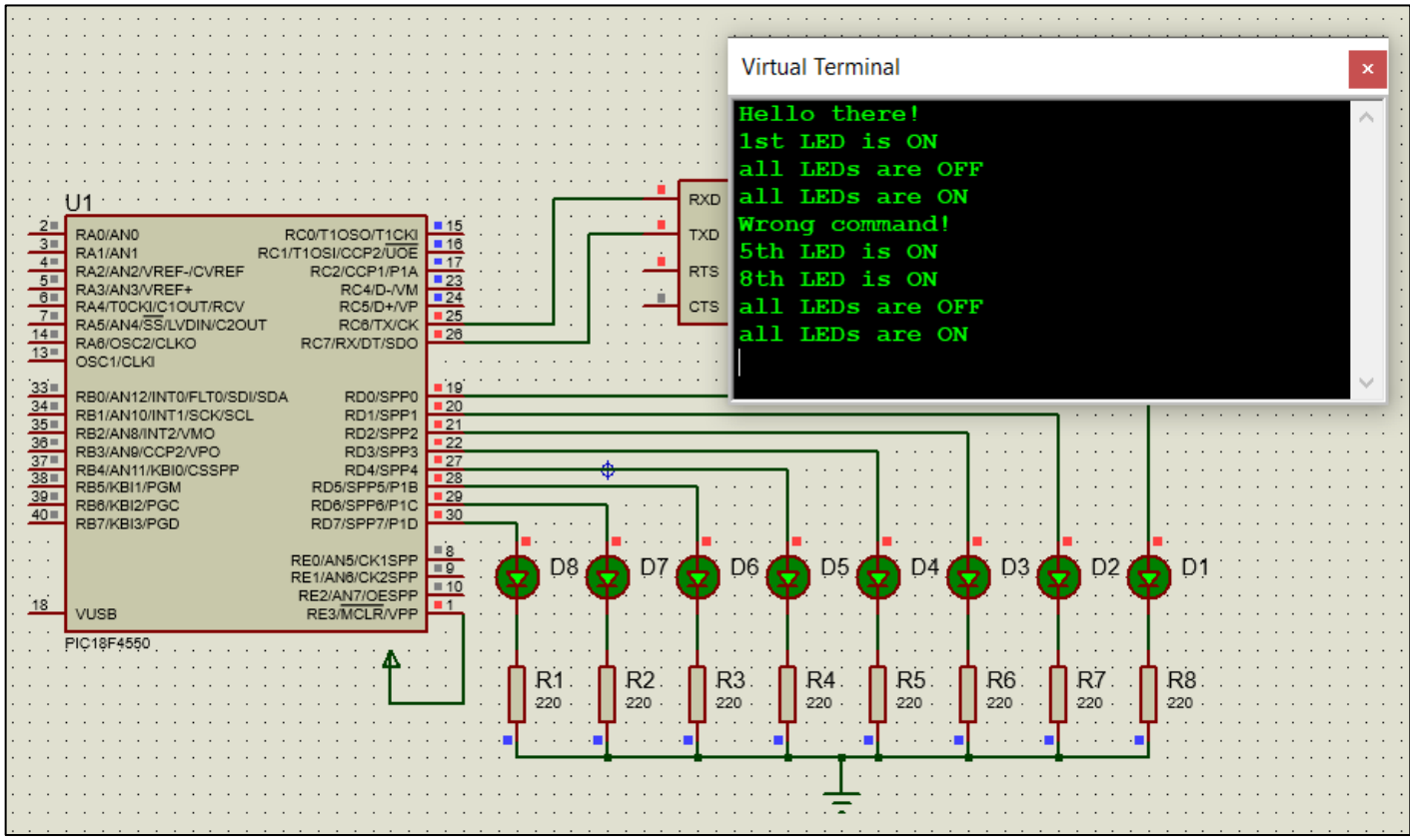


Figure 1. Serial and LEDs

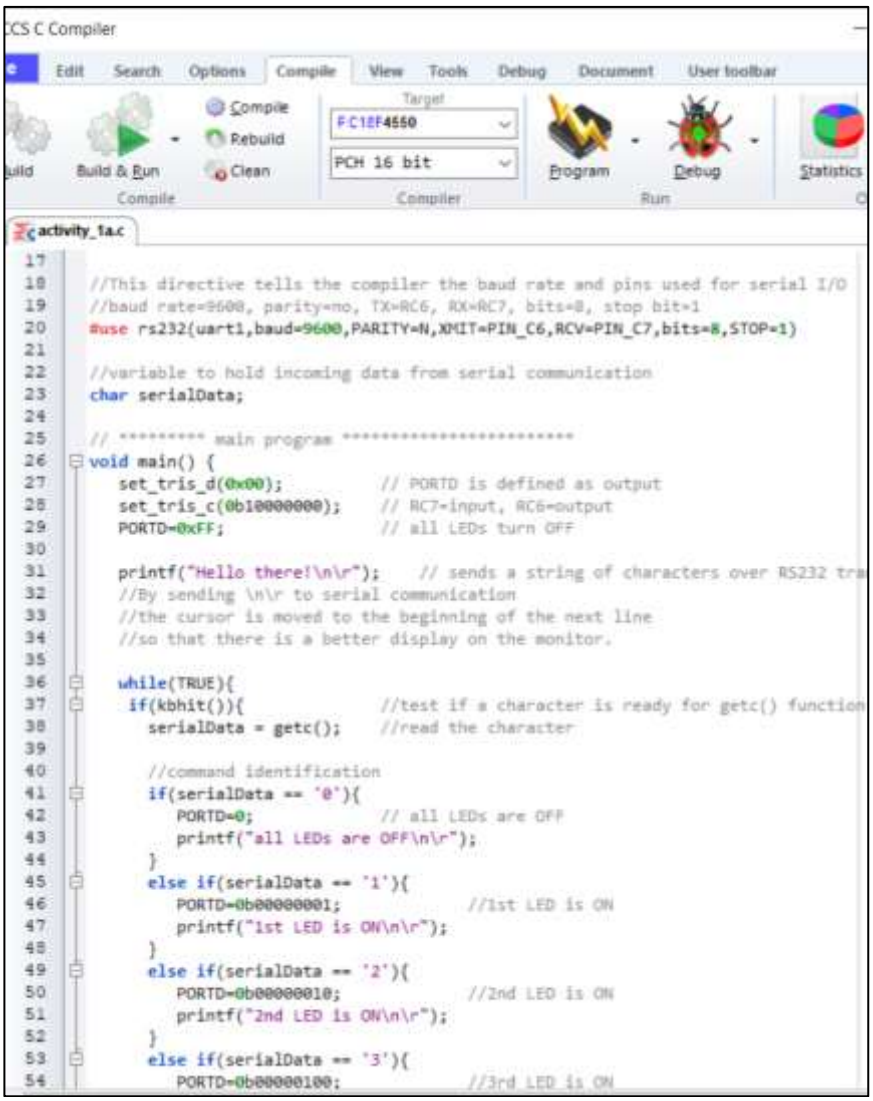


Figure 2. CCS C Compiler, translation to machine code (hex file)

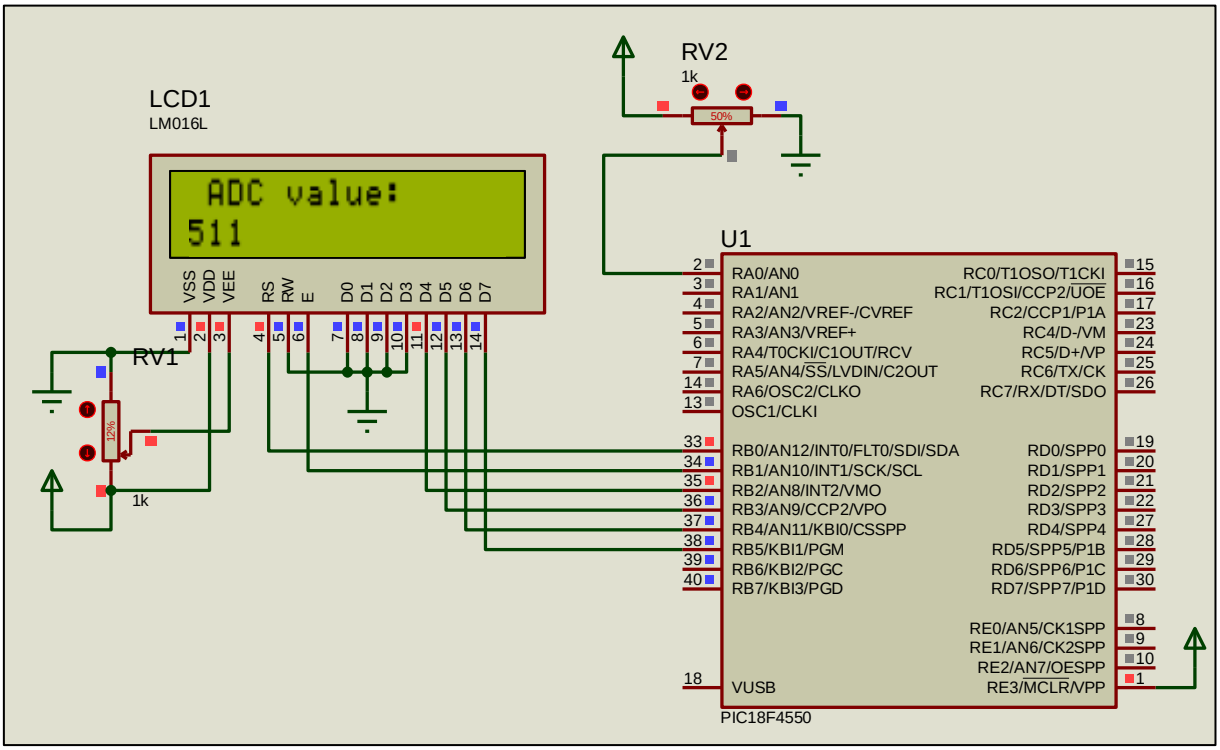


Figure 3. ADC and LCD 16x2

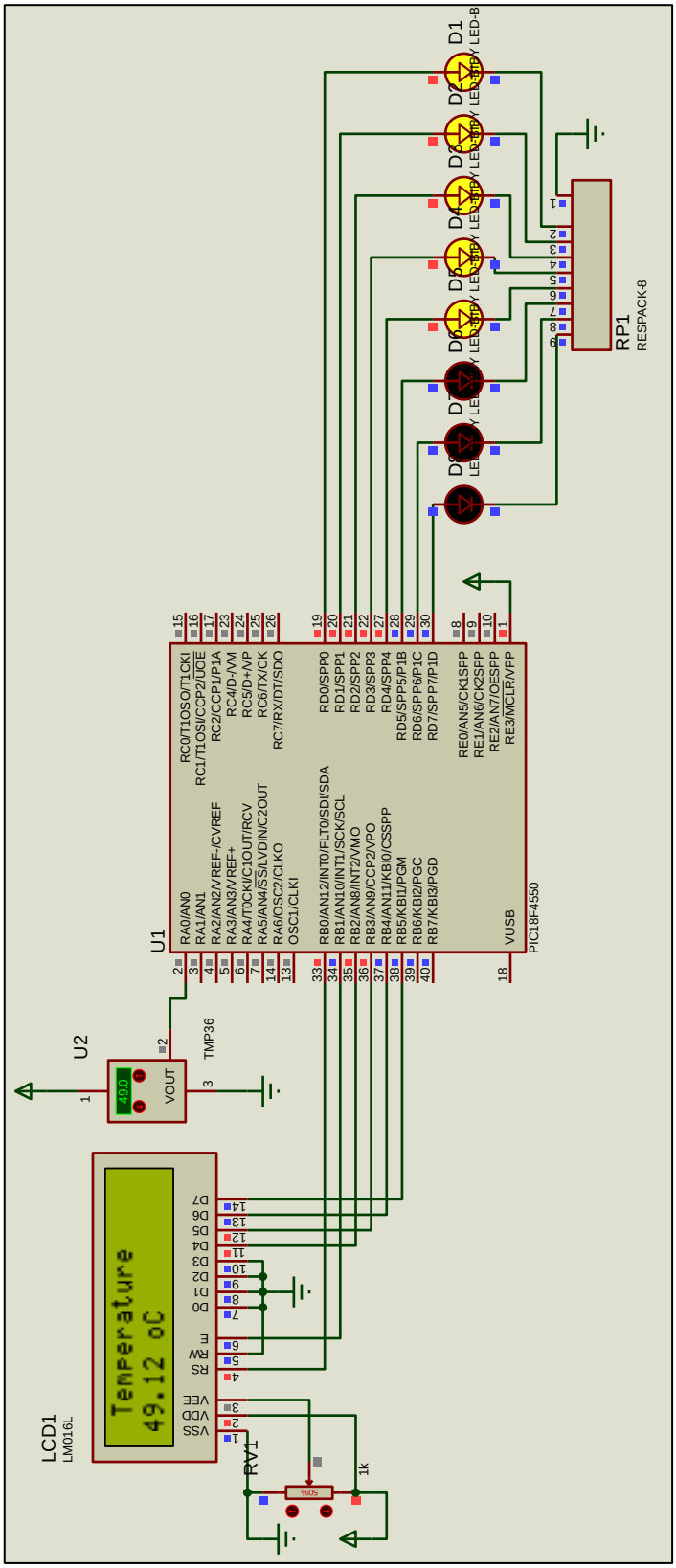


Figure 4. TMP36, LEDs and LCD 16x2