

|  |
| --- |
| Final Project  ETEC-222 |
|  |
| ETEC222  Centennial College  Authored by: Anand Venugopalapanicker  Student ID: 301090911 |



## Introduction

A temperature data logger measures the value of temperature over time and saves the data in the EEPROM. This project is about measuring the temperature value, notify when the temperature goes beyond threshold value on LCD and save the date in the EEPROM. The project is carried out using the development board Microstick II with PIC24HJ128GP502 microcontroller and MPLABX IDE is used to program the microcontroller.

|  |
| --- |
| Problem Statement Large industries such as Pharmaceutical, Health care, Agri-food, Environmental, Logistics, and transportation use temperature data loggers to gain insight into temperature levels. Thus, with the desired need, we have designed and implemented the temperature data logger and indicator to measure the temperature value, save time-stamped temperature value in the EEPROM and notify the high and low variations in temperature from the ‘user-set’ temperature value. Block Diagrams P  I  C  2  4  H  J  1  2  8  G  P  5  0  2  KEYPAD  CRYSTAL OSCILLATOR  LCD  24LC512  EEPROM  DS1631  LED1    LED2 |
| Circuit DiagramProposed SolutionAs a solution to the above-mentioned problem, we have implemented the project which includes the display of current temperature, save data along with a timestamp to the EEPROM, enable users to input high and low threshold temperature value through the keypad and display the values in LCD. It will also display 20 logged or saved data from the EEPROM to the UART.Block Diagram and Implementation The block diagram consists of PIC24HJ128GP502 Microcontroller, LCD, Keypad,  Temperature sensor, Crystal Oscillator, and EEPROM. The PIC24H 16-bit device family employs a powerful 16-bit architecture, ideal for applications that rely on high-speed, repetitive computations, as well as control. PIC24HJ128GP502 microcontroller has 28 pins, 128 Kbytes of Flash memory, 8 Kbytes of RAM and 21 I/O pins. The microcontroller performs the reading of temperature values from a temperature sensor and stores the value in the EEPROM memory. The recorded or saved temperature values can be observed in Teraterm through UART.  The temperature sensor used in this project is DS1631. It is a digital temperature sensor that provides 9, 10, 11, or 12-bit temperature readings over a -55°C to +125°C range. It communicates with the PIC24HJ128GP502 microcontroller through the I2C communication protocol and provides current temperature values. 16\*2 LCD module is used in this project. It is connected to Port B I/O pins of the microcontroller. The 32-Khz Crystal oscillator is used for getting time-stamped temperature data. The keypad is configured as low for rows and high for columns. The user can input low and high threshold temperature values through the keypad. **Work Done by Anand Venugopalapanicker**  * Checked individual components work along with its program * Tried to merge RTCC code with EEPROM and succeeded * When my partner succeeded in combining RTCC code with DS1631, I combined it with my current code * Tried to combine LCD code with the current code and succeeded * Wrote code for keypad input by modifying the keypad code and adding my custom code sections. Combined this code with LCD code to display the keys pressed * Modified the keypad code with the interrupt to get the pressed key values and show it on the LCD * Added custom code to handle the user inputs from the keypad and for user threshold temperature prompting * Changed the Circuit by changing the pin assignments in the code as well as in the circuit to accommodate all the components in the project. Allocated other pins for LCD to get the SDA and SCL pins allocated for I2C channel for 24LC512 EEPROM and DS1631 temperature sensor * Did the trial and error method many times to understand the output and to check whether the circuit and code works well * Wrote code for checking the current temperature with threshold temperature values and display the notification. * Finalized the prototype non-blocking code with all components working * Helped my partner in understanding the code * Checked whether DS1631 program is working perfectly * Build and compile RTCC code and checked its working with the crystal * Tried to build the circuit combining DS1631 and RTCC * Collected data for the report during the period * Created the flow diagram for the project * Prepared the project report  Firmware DesignFlow chartCode Modules and Explanation  1. **Module 1**   Int main()  {…….  u8\_i = 2;  au8\_buf[u8\_i++] = u\_RTCC.u8.date;  au8\_buf[u8\_i++] = u\_RTCC.u8.month;  au8\_buf[u8\_i++] = u\_RTCC.u8.yr;  au8\_buf[u8\_i++] = u\_RTCC.u8.hour;  au8\_buf[u8\_i++] = u\_RTCC.u8.min;  au8\_buf[u8\_i++] = u\_RTCC.u8.sec;  au8\_buf[u8\_i++] = (i16\_temp & 0xFF00) >> 8;  au8\_buf[u8\_i++] = i16\_temp & 0xFF;  ……………. }   * Declaring a variable for the indexing of the buffer address. * Allocating the values for Date, Month, Year, Hour, Minute, Second, and Temperature to the consecutive buffers for storing it into the EEPROM. * We use 2 consecutive buffers of 8 bits to store a 16bit temperature value.  1. **Module 2**   typedef enum {  KEY\_PAD\_DEFAULT\_OP,  KEY\_PAD\_EEPROM\_DISLAY\_OP,  KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP,  KEY\_PAD\_LOW\_TEMP\_INPUT\_OP,  } KEYPADOP;   * Declaring custom enumerators for the custom states for the current program. * These states are used in switching between high and low-temperature threshold value setting code parts. * The states for displaying saved value from EEPROM (KEY\_PAD\_EEPROM\_DISLAY\_OP), setting high-temperature value and output on the LCD (KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP), and for setting high-temperature value and its output on the LCD (KEY\_PAD\_LOW\_TEMP\_INPUT\_OP).  1. **Module 3**   // for handling the User input  void handleUserInput (char \*userData,  int userDataSize,  int printHeader,  int inputComplete,  int op)  { char string[100];  string[0] = '\0';  printf("handleUserInput - %d, %d\n", userDataSize, printHeader);  if (printHeader) //Display the Temperature threshold prompt  { writeLCD(0x01,0,0,1); // clear display, move cursor to home  if (op == KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP)  { outStringLCD("T(high): ?");  } else if (op == KEY\_PAD\_LOW\_TEMP\_INPUT\_OP) {  outStringLCD("T(low): ?");  }  } else {  writeLCD(0xC0,0,0,1); // cursor to 2nd line.  int index = 0;  // Get data in the userData buffer and print the values  for (index = 0; index < userDataSize; index++)  { printf("handleUserInput userData[%d] = %d\n", userData[index], index);  sprintf(string,"%s%c", string, userData[index]);  }  outStringLCD(string);  }  if (inputComplete) // if key-press input is done, start to display confirmation//  { int mul = 1;  int input = 0;  int index = 0;  // Convert ASCII value to get Decimal value and print  for (index = userDataSize - 1; index >= 0; index--)  { input += (int)(userData[index] - '0') \* mul;  mul \*= 10;  }  printf("user entered limit = %d\n", input);  initLCD();  if (op == KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP)  { sprintf(string, "T(high) set as %d", input);  outStringLCD(string);  DELAY\_MS(500);  //printf("Temp is: %4.4f (C)\n",(double) f\_tempC);  DELAY\_MS(500);  temperatureLimit\_high = input;  } else if (op == KEY\_PAD\_LOW\_TEMP\_INPUT\_OP) {  sprintf(string, "T(low) set as %d", input);  outStringLCD(string);  DELAY\_MS(500);  //printf("Temp is: %4.4f (C)\n",(double) f\_tempC);  DELAY\_MS(500);  temperatureLimit\_low = input;  }  }  }   * The handleUserInput() function is for taking the input from the keypad and to set the value for high and low-temperature value by the user. * Using the states defined in the typedef enum function, we can switch between different needs. * Depending on the values we give to the arguments in the handleUserInput() function, the program execution will enter into the specific if loops such as if(printHeader) or if(inputComplete) and so on. * The if(printHeader) loop will help us to prompt for the T(high) and T(low) values. * It's else’ part will help us to store the input value of temperature value to the string and show the string on the LCD 2nd line. * The if(inputComplete) loop is for prompting the confirmation of T(high) and T(low) values as 'set'. * The ‘for’ loop inside the if(inputComplete) loop will help us to get the decimal value of the ASCII character we input through the keypad for T(high) and T(low) values. The ‘mul’ variable will be used to multiply it with the input keypad value to get the ones and tenth place value of a 2 digit number that we enter through the keypad.  1. **Module 4**   void printLatestReadingsFromEEPROM(int numReadings)  { printf("testing 3 \n");  uint8\_t au8\_buf[BLKSIZE+2];  uint16\_t u16\_MemAddr = 0;  uint8\_t index;  for (index = 0; index < numReadings; index++)  {// read and check if it is written  memReadLC515(EEPROM, u16\_MemAddr, au8\_buf); // do read  double temp\_inC;  // print time-stamp from EEPROM  printf(" %2x/ %2x/ %2x, %02x:%02x:%02x\n",(uint16\_t)au8\_buf[0],(uint16\_t)au8\_buf[1],(uint16\_t)au8\_buf[2],(uint16\_t)au8\_buf[3],(uint16\_t)au8\_buf[4], (uint16\_t)au8\_buf[5]);  int16\_t i16\_temp = au8\_buf[6] << 8 | au8\_buf[7]; /\*read temp. from buffer and store to variable\*/  temp\_inC = i16\_temp;  temp\_inC = temp\_inC/256; //get temp. in oC  printf("Temp from eeprom is: %4.4f (C)\n", temp\_inC); // print temp. in oC  u16\_MemAddr += BLKSIZE;  }  }   * The printLatestReadingsFromEEPROM(int numReadings) function is to print the last saved 20 temperature loggings from EEPROM to the UART (Teraterm) terminal. * Inside the ‘for’ loop, there is a memReadLC515() function that reads the saved data from the EEPROM address. It will store the ‘read-value’ to the int16\_t i16\_temp variable (for temperature) to do the calculation to get the temperature in degree Celsius and Fahrenheit and display it on the Terminal.  1. **Module 5**   void \_ISR \_T3Interrupt (void)  {  switch (e\_isrState)  {  case STATE\_WAIT\_FOR\_PRESS:  if (KEY\_PRESSED() && (u8\_newKey == 0)) {  //ensure that key is sampled low for two consecutive interrupt periods  e\_isrState = STATE\_WAIT\_FOR\_PRESS2;  }  break;  case STATE\_WAIT\_FOR\_PRESS2:  if (KEY\_PRESSED()) {  // a key is ready  u8\_newKey = doKeyScan(); /\* do key-scan and store the pressed key value to u8\_newKey\*/  e\_isrState = STATE\_WAIT\_FOR\_RELEASE;  } else {  e\_isrState = STATE\_WAIT\_FOR\_PRESS;  }  break;  case STATE\_WAIT\_FOR\_RELEASE:  //keypad released  if (KEY\_RELEASED()) {  printf("key released %c\n", u8\_newKey);  e\_isrState = STATE\_WAIT\_FOR\_PRESS;  if (u8\_newKey == '\*') //check if the key pressed is '\*'  { keyPadOp = KEY\_PAD\_EEPROM\_DISLAY\_OP;  printLatestReadingsFromEEPROM(20);  keyPadOp = KEY\_PAD\_DEFAULT\_OP;  } else if (u8\_newKey == '#') //check if the key pressed is ‘#’  {if (keyPadOp == KEY\_PAD\_DEFAULT\_OP)  { processingUserinput=1;  keyPadOp = KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP;  handleUserInput(userData, userDataSize,1, 0, keyPadOp);  } else if (keyPadOp == KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP) {  handleUserInput(userData, userDataSize,0, 1, keyPadOp);  userDataSize = 0;  // start processing temp low  keyPadOp = KEY\_PAD\_LOW\_TEMP\_INPUT\_OP;  handleUserInput(userData, userDataSize,1, 0, keyPadOp);  } else if (keyPadOp == KEY\_PAD\_LOW\_TEMP\_INPUT\_OP) {  handleUserInput(userData, userDataSize,0, 1, keyPadOp);  userDataSize = 0;  keyPadOp = KEY\_PAD\_DEFAULT\_OP; //  DELAY\_MS(200);  processingUserinput=1;  }  } else {  if ((keyPadOp == KEY\_PAD\_HIGH\_TEMP\_INPUT\_OP) || (keyPadOp == KEY\_PAD\_LOW\_TEMP\_INPUT\_OP)) //check status of keyPadOp  { userData[userDataSize++] = u8\_newKey; //get new key value to buffer  handleUserInput(userData, userDataSize, 0, 0, keyPadOp);  } }  u8\_newKey = 0;  }  break;  default:  e\_isrState = STATE\_WAIT\_FOR\_PRESS; //assign e\_isrState to initial default state  break; }  \_T3IF = 0; //clear the timer interrupt bit  }   * This interrupt function definition gives us the chance to watch for the keypad interrupts and act accordingly with the corresponding key presses. * The function runs based on switch – case statement. * The status of ‘e\_isrState’ will switch the cases accordingly. * We tried to get the benefit of the interrupt by, checking the interrupt status register value and jump into the corresponding cases. * Using different ‘if’ statements, we manage to call the handleUserInput() function with changes in the argument values to do different functions that it meant to do. * The ‘if’ is used to check the key pressed is ‘\*’ or ‘#’ and do the corresponding task assigned if the key was either of the above. Only if the processingUserinput =1. * If the key pressed is other than ‘\*’ or ‘#’, the execution will go to the ‘else’ part and get the key value to the ‘userData[userDataSize++]’ buffer for setting the temperature threshold values. * We used this function to get values from keypad by calling the handleUserInput() function in the ‘case STATE\_WAIT\_FOR\_RELEASE:’ case and checking the status of ‘keyPadOp’ is done for differentiating between the temperature high and low inputting sections from the keypad. * For example, ‘handleUserInput(userData, userDataSize,0, 1, keyPadOp);’ function call will go to the function definition and only enter into the ‘if(inputComplete)’ check loop because in the arguments, only the ‘inputComplete’ has the true condition (1). i.e., the ‘handleUserInput()’ will only depend on the true condition of arguments userData, userDataSize, inputComplete , and keyPadOp.  1. **Module 6**   void showProjectHeader(){  initLCD();  outStringLCD("#Final Project#");  }   * This is a simple module for printing the string #Final Project# on the LCD display after clearing the display.  1. **Module 7**   if (processingUserinput==0) {  Green\_LED2 = 0;  Yellow\_LED1 = 0;  if(f\_tempC <= temperatureLimit\_low) // check if current temperature is lower than userlimit  { writeLCD(0x01,0,0,1); // clear display, move cursor to home  outStringLCD("TEMP is LOW");  Yellow\_LED1 = 1;  DELAY\_MS(500);  Green\_LED2 = 0;  }  if(f\_tempC >= temperatureLimit\_high) // check if current temperature is greater than userlimit  { writeLCD(0x01,0,0,1); // clear display, move cursor to home  outStringLCD("TEMP is HIGH");  Green\_LED2 = 1;  DELAY\_MS(500);  Yellow\_LED1 = 0;  }  char string[100];  string[0] = '\0';  writeLCD(0x01,0,0,1); // clear display, move cursor to home  sprintf(string, "Temp is %4.2f C", f\_tempC);  outStringLCD(string);//}  writeLCD(0xC0,0,0,1); // cursor to 2nd line.  sprintf(string, "%2x/%1x/%2x,%02x:%02x:%02x",(uint16\_t) u\_RTCC.u8.date,(uint16\_t) u\_RTCC.u8.month,  (uint16\_t) u\_RTCC.u8.yr, (uint16\_t) u\_RTCC.u8.hour, (uint16\_t) u\_RTCC.u8.min, (uint16\_t) u\_RTCC.u8.sec );  outStringLCD(string);  }   * This module will firstly check for whether the user is giving any input through Keypad. The enter into this loop only if processinUserinput = 0. * This checking ‘if’ statements are for comparing the instant room temperature with the threshold temperature value set by the user. * Whenever the current temperature goes beyond the T(high) set point, the project will notify as “TEMP is HIGH” as well as turn ON the LED2. * Whenever the current temperature goes lower than T(low) set point, the project will notify as “TEMP is LOW” as well as turn ON the LED1. * If the current temperature is between the T(high) and T(low), the project will display the project name on the LCD and current temperature on the Teraterm Terminal.  1. **Module 8**   #define CONFIG\_Green\_LED2() CONFIG\_RA1\_AS\_DIG\_OUTPUT()  #define Green \_LED2 (\_RA1) //led2 state  #define VREF (3.3);//led1 state  #define CONFIG\_Yellow\_LED1() CONFIG\_RA0\_AS\_DIG\_OUTPUT()  #define Yellow\_LED1 (\_RA0) //led1 state  #define VREF (3.3);   * Defining the LED1(Yellow) and LED2(Green) for the high and low temperature indication. * Pins RA0 and RA1 are configured as OUTPUT. * Reference voltage is set as 3.3V for the pins. (For LEDs)  1. **Module 9**   loopCount++;  if(loopCount == 15){  memWriteLC515(EEPROM,u16\_MemAddr, au8\_buf); // do write  u16\_MemAddr = u16\_MemAddr + BLKSIZE;  loopCount=0;  }  DELAY\_MS(50);   * This loop will initiate one counter and count till 15 with a delay of 50 ms. * When it reaches the count, one memory write is done.  Screenshots and ResultsA screenshot of a computer  Description automatically generatedTimestamp initialization  1. A screenshot of a computer     Description automatically generated**Low and High Threshold value of temperature**  Current Temperature with timestamp  Display the last 20, logged or saved temperature data from the EEPROM**Project Hardware**Conclusion A temperature data logger project was demonstrated using PIC24HJ128GP502 microcontroller, Temperature sensor, LCD, Keypad, EEPROM and Crystal Oscillator. The project displays the current temperature and notify the temperature status on LCD whether it is high or low upon comparing with the threshold value. Users can also view the 20 logged or saved temperature data from the EEPROM on the Teraterm. Lessons learnt and future work We have learned the following ideas from making this project:   * To integrate the codes of different modules * Interfacing IC’s with microcontroller * Learned about when to use the different interfaces including I2C, SPI, and UART   For the future work, we can implement GSM technology or use services such as ThingSpeak or AWS to send the data to remote locations for monitoring, automation and regulation of devices. References  * <http://ww1.microchip.com/downloads/en/DeviceDoc/70293G.pdf> * <https://datasheets.maximintegrated.com/en/ds/DS1631-DS1731.pdf> * <http://ww1.microchip.com/downloads/en/devicedoc/21754m.pdf> * Reese, R. B., Bruce, J. W., & Jones, B. A. (2009). Microcontrollers: from assembly language to C using the PIC24 family. Nelson Education. |
|  |
|  |