Time & Temperature Display

EE 3501 EMEBEDDED SYSTEMS PROJECT

Caleb Winfrey
FALL 2020 | KSU ID:

Objective:

The goal of this project is to design a clock that can be set by the user multiple times if needed, be able to tell the user when an invalid value for hours, and/or minutes are inserted. The clock will also report the temperature in Fahrenheit when a button on the board is pressed and start at a default value of 12:00:00 on startup just like a normal clock.

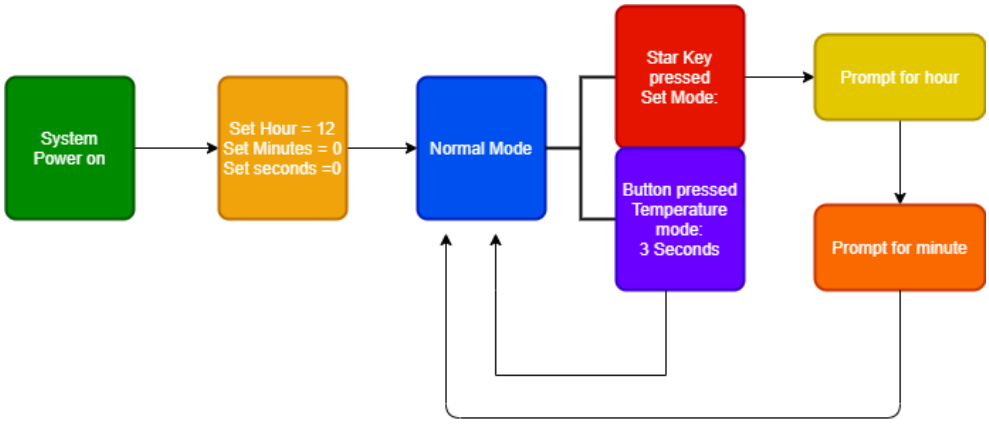
Apparatus List:

The equipment used in building the hardware of the clock, and testing:

1. Discovery Board - DISCO-L476VG
2. Keypad – Adafruit 419
3. Temperature Sensor – Texas Instruments LM35DZ
4. USB A to USB B Mini Cable – Molex
5. 3 Solderless Bread Boards (various sizes) – RexQualis KD-BK-4
6. Various Jumper Wires
7. A PC for console interface and the complier.

Design Specification Plan:

At the beginning of the designing process, it was useful to visualize what portions of the code needed to be executed at what time. Below is a simple flowchart outlining what functions needed to be executed and how the flow control should be structured.



The flow chart is ignoring the parts of the code that check for user error, but those will be explained.

The given requirements stated that the clock have two main modes, the normal mode, and the set mode. On powerup the clock is set to default values and continues forever until the star key is pressed. Then the LCD prompts the user for the hour, and then the minute after the values are entered and the star key is pressed. Then the code returns automatically to normal mode at the set time. The device will also display the temperature when the button on the board is pressed and return to the clock function after 3 seconds.

Before any clock functions can used a use of a while loop that runs when the green LED on the board is off which will be useful for the temperature function. For the normal mode, three if statements for the seconds, minutes, and hours where each returned its value to default if it reached or exceeded the max value. 59 for minute and second, 13 for the hour. A delay function for the loop to count once every second is used. After the if statements iterate the data is formatted as a string, the display is cleared and then prints the time is displayed on the LCD. A keypad scan function is also called to write whatever key is pressed to a string which is used to check if the star key is pressed. This was chosen for its simplicity as it did not need any special time functions and the if statement could cascade off each other and it could reset easily.

For the set mode, an if statement is used within the same while function as stated before, to check if the star key is pressed, if it is then it executes to prompt the user and then using another while loop continuously runs a keypad scan function and places the integer value of what was pressed on the keypad into a variable “hourVal” an if/else statement after the function call checks if the hourVal is within or equal to 1 and 12 to break out of the while loop, else it display’s “ERROR” on the LCD for a bit before restarting the loop. After breaking out of the hour while loop the LCD displays the hour input to the user. The code moves to the next while loop that deals with the minute prompt. The minute loop is nearly identical to the hour loop but stores the user input into a variable called “minVal” and its if statement checks if the user enters an integer between or equal to 0 and 59. After it is all done the values stored in the temporary variables are set as the hour and minute variables used in the if statements for the normal mode and the while loop repeats.

The set mode described above was used because the flow control is simple and does not need anything incredibly complex to run, only some simple if and while statements are needed. The set and normal modes being run in the same while loop allows the set values to be moved to the normal mode easily. One aspect of the design was to have the LCD “mirror” whatever was pressed on the keypad onto the screen, so the keypad scan function has code to run the display too.

One key detail is there are two keypad scan functions outside of the main function one that has all the mirroring and data conversion for the set mode and one that is a lighter version that only needs to output a character for the normal mode to exit when star is pressed. This makes the coding easier and is reasonably stable and allows for the set mode to just call the same keypad function twice.

The main keypad function mirrors anything that is pressed by using a do/while loop that checks what is pressed on the keypad, the keypad scans and then displays that value to the LCD by converting the character to a string and then displaying the string. There are two arrays one for the raw user input used output the total value input by the user, and the other used for the mirroring and to compare if the number sign is pressed to break the loop. At the end of the function, the user input is displayed, converted into a form the LCD uses and then the string is converted into an integer which gets rid of the “#” so the clock can use it.

For the temperature operation a separate function used outside of the main function and is controlled by an if statement inside the clock function while loop. If the green LED turns on the LCD is cleared and the temperature function is called where the sensor is read and the Celsius reading off the sensor is converted to Fahrenheit, converted to a character string from integer value and displayed on the LCD for 3 seconds before returning to normal mode.

The temperature function depended on the green LED to be on as the button does not directly control the temp function itself but an ISR that interrupts the program anytime the button is pressed. An ISR was chosen because it makes sure no mater what the clock is doing it will display the temp when the button is pressed and is reasonably reliable to function. The ISR simply toggles the green and red LEDs as the proven code from a previous lab could be used and it given visual confirmation that the ISR is working correctly which is useful for testing.

Test Plan:

While building the program, there were constant small tests being conducted so ensure there were no major problems developing while functions and features were being implemented. Small scale pilot tests of using the LCD library with the used of the keypad scan function were done to nail down the most stable way to display the user input, and parse that data back to the main function. Similar tests were done with the ISR and temperature function. Which by a lot of trial and error the tests lead to the development of the keypad and temperature functions used in this project. Once the keypad/temperature function and LCD display were satisfactory in reliability the program was ready to be expanded to include new functionality to meet the design specs this was done through an iterative process that would add code, test it and amend it until a desirable and stable result was observed which was a lot of trial and error.

The first, second and third iterations was to get the keypad and LCD to work together this included testing how the keypad function could be written to work as a 4X3 keypad, however this proved to be unstable and the function was kept as a 4x4. Another aspect of the testing phase of interactions was to parse an integer value from the keypad function to the main function this was tested by outputting a digit to the main function from the keypad function and outputting the value and that value multiplied by 3 to see if expected results are shown. The “out of range” features were tested too, and if/else statements worked well since the keypad function output was an integer. Once the output was what was expected the iteration was complete. Fourth iteration was to the clock to just run-in normal mode when the board was powered. Starting at 12:00:00 meant that the default values had to be first stated then a series of if statements would run and then the values converted to a string to be output on LCD. The “rollover” from 12:59:59 to 1:00:00 was tested too and it stacked up to the requirements well as all that was needed to do was to set the hour value back to 1 when it reached 13.

The fifth iteration was to get the set mode to work and to return to the normal mode which included starting up at normal mode, then with press of star the set mode would run and then return to normal mode once complete. This meant copying, simplifying, and modifying the keypad function to output a character string which would be used by an if statement to see if the star key is pressed. Iteration 6 was using the ISR and temperature sensor function (which was written in one of the pilot tests) testing as the program was in normal mode, while the initial tests worked, the clock would "lose" time when normal mode would return. Since the amount of time on the temperature operation was known, the simplest way to ensure time would be accurate was to add seconds to the seconds variable when normal mode starts again.

The final test (iteration 7) was to see how it all worked together, the digital out pins needed to be changed to make room for the temperature sensor which is analog. After testing the new keypad config worked the clock was tested multiple times at different values placed into set mode several times and the temperature mode was tested multiple times. First the clock was powered on and it started always at 12:00:00 and began counting the seconds. Next the star key was pressed, and the clock went into set mode, first for the hour value "15" was input to see if it would spot the error and it did after displaying error it returned to enter the hour again and "12" was entered which the calculator displayed back. Then the minute prompt came up and a number "72" was entered and the error displayed like before and then went back to the input. "59" was then entered and the clock went back to normal mode at 12:59:00. The clock ran for a minute and it did restart back to 1:00:00. After some time, the user button was pressed to start the ISR and display the temperature in Fahrenheit, which it did well.

Source Code:

The source code will be placed at the end of this report as it is formatted better when printed to pdf from the compiler itself instead of copy and pasted in.

Conclusion:

This project had it challenges as getting the keypad, the clock, temperature sensor, and the LCD to work together required thought into how to structure the program in a way that was stable, could be debugged and did not require to reinvent the wheel. One advantage to the design of the keypad function was that it could be transplanted to another project if needed although it might need to be clean up for as the conversion is a bit of a mess. Another advantage is the clock is simple and works in the main function so there is not a lot of data parsing needing to be done so simple keypad scan function, some while loops and if statements make the normal and set modes and work decently well. One disadvantage is the clock functions are not as modular as expected at the beginning of the project, so it is kind of large. Another advantage/disadvantage is the use of the ISR controls the LEDs to trigger parts of the code to start/stop, this is good for stability and it works, however it does mean there is some time drift on the clock's accuracy every time the button is pressed even though it has been mitigated. Overall, the design meets the specifications laid out on the instructions and is stable.

```cpp
//Final Version 1.0 Release

/*********************************************************************************************
   EE 3501 Embedded Systems project Code
   Student Name: Caleb Winfrey
*********************************************************************************************/

// Including Libraries and OS.
//============================================================================================
#include "mbed.h" // for Mbed OS libraries
#include "LCD_DISCO_L476VG.h" //for the LCD display driver {from the DISCO_L476VG_GlassLCD example project}
#include <iostream> // basic input and output for the console (mainly used for debugging and testing)
#include <string>//the library to make the use of strings easier
//============================================================================================

//Initializing the GPIO and initiallising some key variables for the keypad function and the ISR.
//--------------------------------------------------------------------------------------------
LCD_DISCO_L476VG lcd; //Initializing the LCD library

DigitalOut ledG(LED1);//Initializing the green LED on the board.

DigitalOut ledR(LED2);//Initializing the red LED on the board.

DigitalOut ROW[4]={PE_11,PE_10,PE_12,PE_13};  // for the rows of the keypad functions

DigitalIn COL[4]= {PA_3, PB_6, PB_7, PD_0};   // for the columns on the keypad functions

char userInput[6]; // the storage array for the user input on the kepad functions

int count = 0; // the intiallized variable used in the kepad function (counts up for the functions)

AnalogIn LM35(PA_0); // the initallization of an analog in pin for the temp display.

InterruptIn button(USER_BUTTON); // Initializing the ISR with the user buttion on the board.

Ticker t; // for the ticker on the ISR.
//--------------------------------------------------------------------------------------------

/*********************************************************************************************
!    DEVELOPER NOTE: While testing the ISR it was found it worked better when the ISR functions
 !   were BEFORE the main and forward declarations of the program.
  !  The main purpose of the ISR is to trigger a hardward change (LED power state)
   ! which will be used as way to flow control the temperature function.
*********************************************************************************************/

//ISR functions:
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void toggle1() // for the rise command, toggles each LED's state at once.
{
    ledR = 0; //toggles Red LED off

    ledG = 1;//toggles Green LED on
}

void toggle2() // for the fall command, toggles each LED's state and then detaches the ticker.
{
   ledR = 1;//toggles Red LED on

   ledG = 0;//toggles Green LED off

 t.detach(); //datching the ISR so the program can return to normal mode.

}

void tickTime() //for controlling when the LED is switch back to red when the button is released.
{

    t.attach(&toggle2, 1); // the command for the delay to switch.
}
//~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

// Forward declarations of the functions outside of main.
//============================================================================================
void tempFunction(); // for powering on the temp sensor and displaying the temperature in Fahrenheit

void keyFunction(int &digit); //the kepad function for intaking the Hour and Minute values in set mode

void keyFunction2(string &set); //the 2nd keypad function that constantly runs while in normal mode to see if the star key is pressed.

//============================================================================================

/*SPACED FOR PDF PUBLISH*/
```

```
// the main function:
//_____________________________________________________________________________________________________________________________
int main(){
     char hourDisp[2];// for initializing final "HOUR" display values.
     char minDisp[2];// for initializing final "Minute" display values.
     uint8_t DisplayedString[7] = {0}; // the initial array that allows the LCD library display text and integers.

     ledG = 0; //for use with the interrupts
     ledR = 0; //for showing when the interupt is done and if the interupt has been pressed already (useful for debug)

     //~~~~~~~~~~~~~~~below is the Normal Mode portion of the code. The program starts here and returns here after set or the ISR.~~~~~~~~~~~~~
     int hr = 12, min = 0, sec = 0; // setting the initial values for hour, minutes and seconds so the clock would begin at 12:00 on powerup.

     char display[8]; //initiallizing a array for use in displaying info.
     string setKey = ""; //initiallizing a string

    button.rise(&toggle1); // button is pressed.
    button.fall(&tickTime); // button is released.

    while (ledG == 0){ // the main clock function, is on when green LED is off.

    wait(1); //waiting 1 second between loops
    sec++; // adding 1 to second
    if (sec >= 59) { // if statment stating for seconds variable to restart and add 1 to the minute variable
        sec = 0;     // when the seconds count reaches 59.
        min ++;
        if (min >= 59) { // if statment stating for minutes variable to restart and add 1 to the hours variable
            min = 0;     // when the minutes count reaches 59.
            hr ++;
            if (hr >= 13) { // if statment stating for hours variable to restart back to 1.
                hr = 1;     // when the hours count reaches 13.
            }
        }
    }

    sprintf(display,"%02d%02d%02d",hr,min,sec); // for formating the data for the LCD display
    cout << display << endl; // console out for debug
    lcd.Clear(); // Command to clear display before displaying.
    lcd.DisplayString((uint8_t *) display); //displays the time.

    keyFunction2(setKey); //Runs this function to see if the star key is pressed so it can go into set mode.

    cout << setKey << endl; // console out for debug.

    if (ledG == 1) { /*if statement for when the Green LED is turned on the program switches the the temperature function.
                       This is why the ISR is used as a hardware interupt, it's stable and flow controls well. */
        lcd.Clear(); //clear lcd
        tempFunction(); //calls the temperature function.
        wait(2); // waits 2 seconds. (there's already about a 1 second deltay with the ISR)
        ledG = 0; //to make sure the the green LED is off after the temperature function is done.
        sec= sec + 2; //adding lost time

        }
    //~~~~~~~~~~~~~~~~~~ below is the portion of the code that controls the set mode of the clock program. ~~~~~~~~~~~~~~~~~~~~~~~~~~~

    if (setKey == "*"){ // the if statement that checks if the return of the second keybad function is returning a star.

     lcd.Clear();
     lcd.DisplayString((uint8_t *) "HOUR"); //prompts user for the hour on the LCD.
     wait(2);
     int hourVal = 0; //for initiallizing return value from function
     while(1){ //the while loop that keeps the program running the keypad function running until number within range is entered
     keyFunction(hourVal);//calling the keypad function
     if (hourVal >= 1 && hourVal <= 12){ // if statment to make sure the user inputs a valid number.
        break;
        }else {// if the user enters a number that is not in range the LCD displays error and then returns to while loop.
            lcd.DisplayString((uint8_t *) "ERROR");
            cout << "Error hour value is out of range" << endl; // for console debug
            wait(2);
            }
      }
     sprintf(hourDisp, "%d", hourVal); // converts integer value(s) from function into a single string
     lcd.DisplayString((uint8_t *) hourDisp);// displays the total value that the user entered for example "1" and "2" outputs "12"
     wait(1);

     lcd.Clear();
     lcd.DisplayString((uint8_t *) "MIN"); //prompts user for the minute on the LCD.
     wait(2);

     int minVal = 0; //for initiallizing return value from function
     while(1){//the while loop that keeps the program running the keypad function running until number within range is entered
     keyFunction(minVal);//calling the keypad function
     if (minVal >= 0 && minVal <= 59){ // if statment to make sure the user inputs a valid number.
        break;
        }else {// if the user enters a number that is not in range the LCD displays error and then returns to while loop.
            lcd.DisplayString((uint8_t *) "ERROR");
            cout << "Error minute value is out of range" << endl; // for console debug
            wait(2);
            }
      }
     sprintf(minDisp, "%d", minVal); // converts integer value(s) from function into a single string
     lcd.DisplayString((uint8_t *) minDisp); // displays the total value that the user entered for example "5" and "9" outputs "59"

     hr = hourVal; min = minVal; sec = 0; // stores the new values for hour and minute into the clock variables seconds are also
    }                                     // reset to 0 before returning to normal mode
  }
}//end of main function
//_____________________________________________________________________________________________________________________________
```

```
//Modular functions outside of the main function
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
unsigned char key_map [4][4] = { // the array for the different characters on the physical keypad that the keypad functions use.
        {'1', '2', '3', 'A'}, //ROW 1
        {'4', '5', '6', 'B'}, //ROW 2
        {'7', '8', '9', 'C'}, //ROW 3
        {'*', '0', '#', 'D'}, //ROW 4
//      COL1  COL2 COL3 COL4
    };

void keyFunction(int &digit){ // the keypad scan function that is used for the set mode.

int count = 0; //initiallizing the count varable.

    COL[0].mode(PullUp); // setting the columns HIGH
    COL[1].mode(PullUp);
    COL[2].mode(PullUp);
    COL[3].mode(PullUp);

    char key[4]= {'0'}; //default character for initialization.

    char str[2]; //for initiallising the string output for mirroring

    char out[2]; // for initiallsing a character array for converting characters to integers

    char display[2]; // for help displaying the output of the function.

    char userInput[2]; // the array storing the total what the user inputs.

    string trim = ""; //for use in trimming the "#"

    string compare = ""; // for use to break out of the do while loop

    do{ // start of the keypad scan, will stop when "#" is pressed
        for(int row=0;row<4; row++){
            ROW[0]=1; ROW[1]=1; ROW[2]=1; ROW[3]=1; ROW[row]=0;
            wait(0.01);
            for(int col=0; col<4; col++){
                if(COL[col]==0){
                    key[0]=key_map[row][col];

                    cout << key <<endl; // console out for debug

                    userInput[count]=key_map[row][col]; // for storing and converting to string

                    count++;
                    if (count ==5)
                    {
                        count= 0;
                        }

                        wait(0.05);

                        while (COL[col]==0);
                        }
                        }

                        sprintf(str, "%s",key); // for converting the unsigned char to string.

                        lcd.Clear();

                        lcd.DisplayString((uint8_t *) str); //outputs what the user enters.

                        compare = str; // intermediate step to help with breaking out of the loop when "#" is pressed

                        }


   } while ( compare != "#"); //breaks out of the loop when # is pressed


    cout << userInput << endl; // console out for debug

    sprintf(out, "%s",userInput); //for converting the input into a char

    trim = out; // for converting char into string

    cout << trim << endl; // console out for debug

    lcd.DisplayString((uint8_t *) out); /* since the "#" wont show up on display this will be okay
                                           for just showing what the user put in. must be before all the trimming */
    digit = stoi( trim ); // for converting string to integer (will be output into the main function)
}









/*SPACED FOR PDF PUBLISH*/
```

```
/*****************************************************************************************************************************************************************
!    DEVELOPER NOTE: The function below is used only for the program to see the star key is pressed, while it is very simular to the first keypad scan function
  !  it only needs to return a single character string for the main function to go into set mode. and it doesn't need to output anything to the LCD
*****************************************************************************************************************************************************************/


void keyFunction2(string &set){ // this is the keypad scan function to monitor when star is pressed to go into set mode

int count = 0; // initiallizing count variable

    COL[0].mode(PullUp); // setting the columns HIGH
    COL[1].mode(PullUp);
    COL[2].mode(PullUp);
    COL[3].mode(PullUp);

    char key[4]= {'0'};//default character for initialization.

    char str[2]; //for initiallising the string output for mirroring

        for(int row=0;row<4; row++){
            ROW[0]=1; ROW[1]=1; ROW[2]=1; ROW[3]=1; ROW[row]=0;
            wait(0.01);
            for(int col=0; col<4; col++){
                if(COL[col]==0){
                    key[0]=key_map[row][col];

                    cout << key <<endl; // console out for debug

                    userInput[count]=key_map[row][col]; // for storing and converting to string

                    count++;

                    if (count ==5)
                    {
                        count= 0;
                        }

                        wait(0.05);

                        while (COL[col]==0);
                        }
                        }

                        sprintf(str, "%s",key); // for converting the unsigned char to string.

                        set = str; //seting the user input string into the output of the function.

                        }
}

void tempFunction() // the tempurature sensor function, is triggered by the ISR through the push of the userbutton
{

    int value; //initializing the value varable

    value = ((LM35.read()*3276)/100); //taking the raw data of the sensor and seting to celcius

    uint8_t DisplayedString[7] = {0}; //initiallizing the character array for the LCD library.

    int fVal = (value * (1.8) + 32); //converting to fahrenheit

    char stringVal[8]; // array for he output to the LCD

    sprintf(stringVal, "%d F", fVal);  // converts integer values to strings

    lcd.Clear();

    lcd.DisplayString((uint8_t *) stringVal); //display temp

    printf("Temp: %d degree c\r\n", value); // console out for debug

    printf("Temp: %d degree F\r\n", fVal); // console out for debug

}

// End of program code.
```