Greenhouse Control System

Embedded Systems Development: 600085

Group Report By Scott Kikumu and Gabriel Coventry

Word Count: 1898

Introduction

The group report aims to describe the main functionality of the greenhouse control system. The system will be responsible for maintaining climate temperatures in a desired room by integrating with and controlling heating and cooling systems.

Project description and requirements:

The interfaces that should be used are the real time clock, the thermometer, the LCD panel, the 4x4 button matrix, the buzzer and the I/O ports embedded on the board. All of these interfaces were used in the project. A low level driver should be coded to interface with the device and the creation of an API which contains the functions used to initialize the device and can be used for other activities. It should be able to set the current time and date as well as the temperature. It should allow the user to set the date, time and temperature as well and should have trigger thresholds. It should also allow for two different thresholds so that if the heating drops too low it will start heating or if it goes too high it will start air circulating instead. An alarm should sound when surpassing thresholds and the user should be able to turn it off. There should be day and night operation modes and the day should began at 6:30 and end at 7:30. Nighttime will be out of this range. The temperature resolution can be to at least 1 dp.

Code:

Code listing can be found in the appendices under Code Listing.

Hardware and Pin Mappings

The separate hardware components used on the PIC Board in this project are:

- Thermometer
- LCD 128x64 Display
- Buzzer
- Onboard Button Matrix
- Realtime Timekeeping Chip

Ports Used

The thermometer usually mapped to RA0 was mapped to RE0 in order to avoid conflicts with the LCD display which has its reset mapped to RA0.

The LCD128x64 was mapped to Port A and Port D, specifically in port A, RA0, RA2, RA3, RA4 and RA5. RA0 is the port responsible for resetting the LCD, clearing and refreshing the screen. RA2 is responsible for PSB which decides whether the LCD is in Serial Mode or Parallel mode, which is the mode used for our project. RA3 is responsible for the pullup function which then tells the LCD to start accepting data. RA4 is responsible for deciding whether the data is being written or read. RA5 is responsible for telling the LCD if the incoming data is actual data or a command. Port D on the LCD and the pins that encompasses are used for data, whether it be reading or writing data or it could be issuing a command such as clearing the screen.

The Buzzer has been remapped to RE1 due to the fact the native pin it is mapped to (RA0) conflicts with both the Thermometer and the LCD display which would not allow for concurrent use of any of these components

The Onboard Button Matrix uses Port C. Jumper 16 is set to the pull up position meaning that detecting the button presses is done by detecting which pins have been pulled up. Port C has been split in half with the first 4 pins (RC0-RC3) set as input and the higher pins (RC4-RC7) have been set as output. This is because in order to detect the button presses on the Onboard Button Matrix, we have to be able to set pins but also detect pins that have changed for example, in regards to the below diagram, to detect if k12 had been pressed we would pull down RC1 and listen to RC4 to see if it has also been pulled down, if so then the button has been pressed as it would connect RC1 and RC4 together in the circuit, pulling down RC4 in the process.

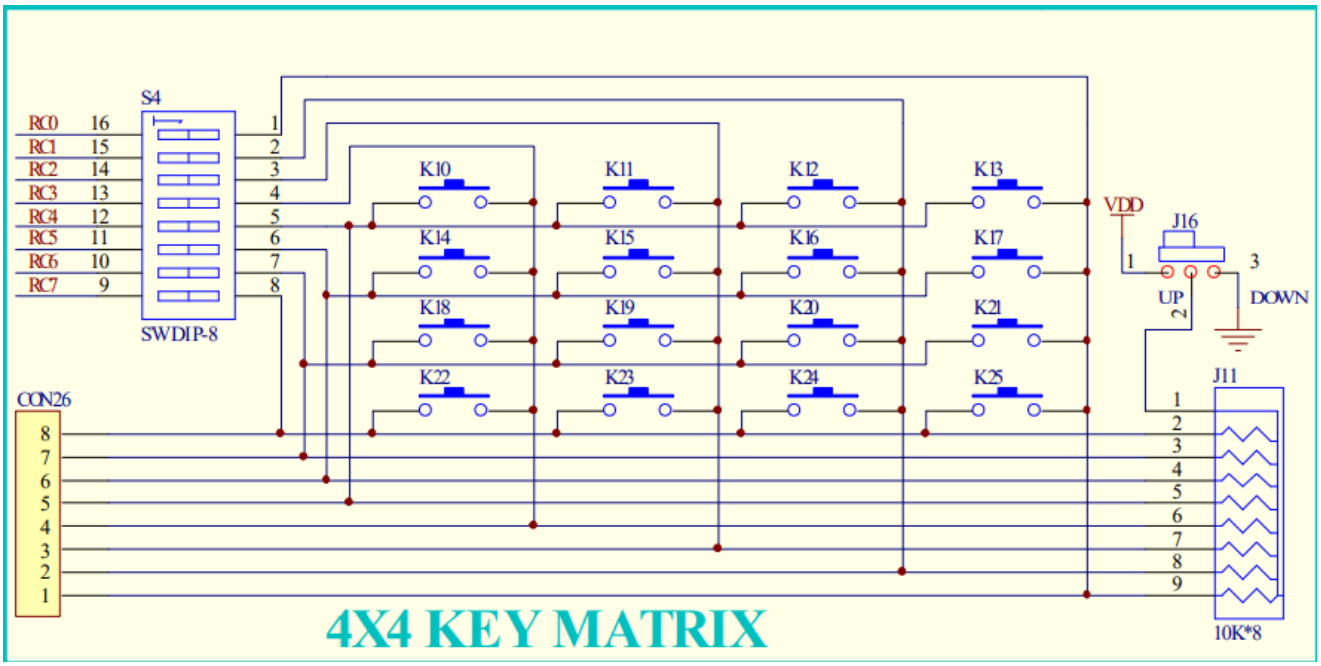


Figure 1 (QL200_SCH, n.d.)

The Timekeeping Chip uses 3 pins, RB0, RB4 and RB5. RB0 is used for sclk(Serial Clock Input), this is used to synchronize data movement, so this pin will be used for commands and data when set to high, since it is only 1 bit and not a byte it will do this serially, bit by bit to create a byte. Once a byte is created it is set back to low. RB4 is used for the Input/Output function, this decides whether or not the data is being read by the clock or written. The RB5 pin is used for resetting the clock when pulled low it resets and disables the clock and when pulled high it enables it again.

Drivers:

for the functions associated with these drivers please refer to the Functions section of the appendices

- Button_driver.h
  This driver is responsible for observing key presses and detecting which has been pressed as well as which menu to switch to.

- Clock_driver.h
  Clock driver is for getting the time and sending commands to the clock

- Day_of_week_driver.h
  this header file is responsible for displaying the day of the week associated with the day number

- LCD_driver.h
  this driver is responsible for controlling the LCD display such as sending commands and display information.

- Math_driver.h
  This driver is responsible for setting up for arithmetic used throughout the program.

- Thermometer_driver.h
  This driver is responsible for setting up the thermometer as well as communicating with it via commands and retrieving the temperature

- Threshold_driver.h
  This driver is responsible for dictating the temperature thresholds and alerting the buzzer when it is reached.

- Buzzer_driver.h

  this driver is responsible for activating the buzzer went sent a signal.

- SystemHeader.h

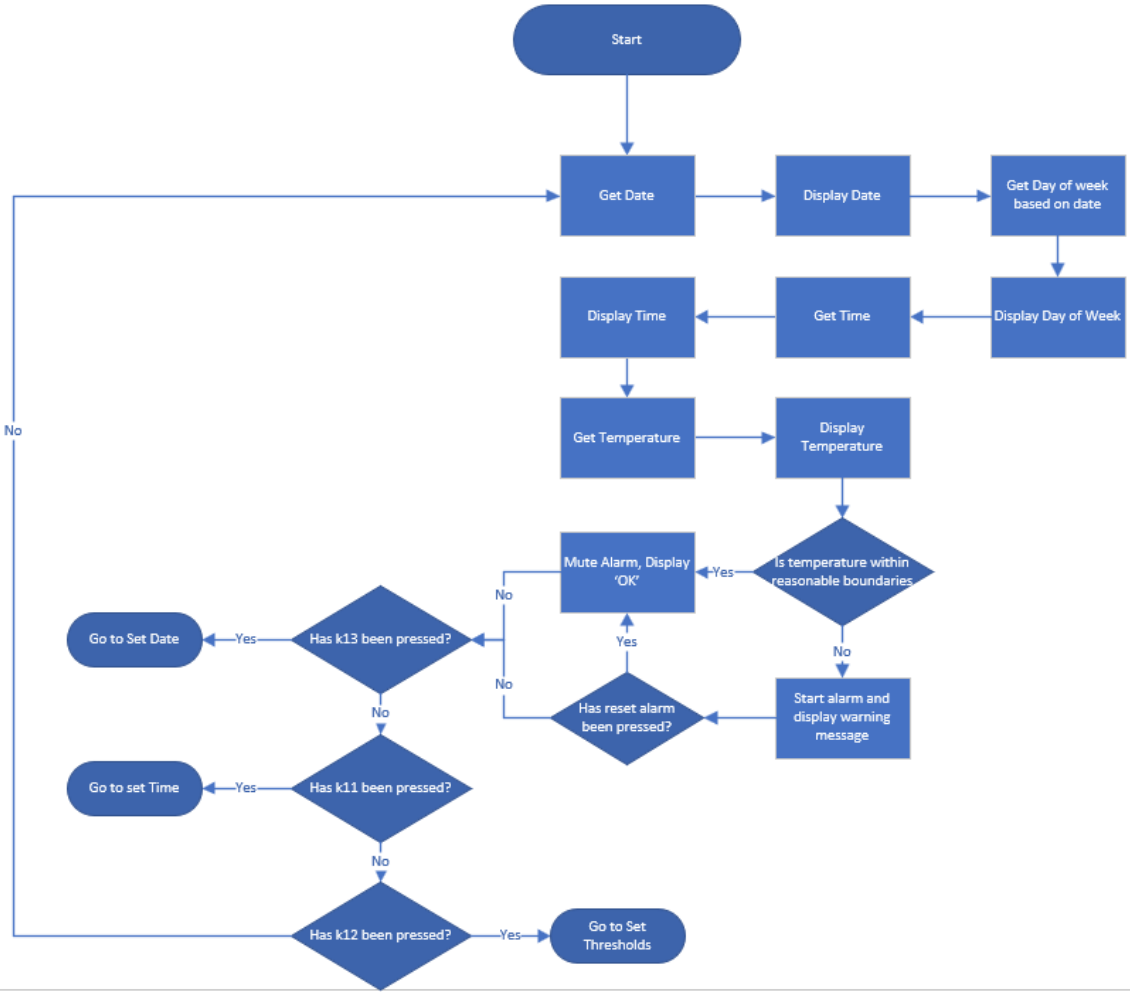
  This driver is responsible for initialising pins and ports for other drivers and functions.

Testing:

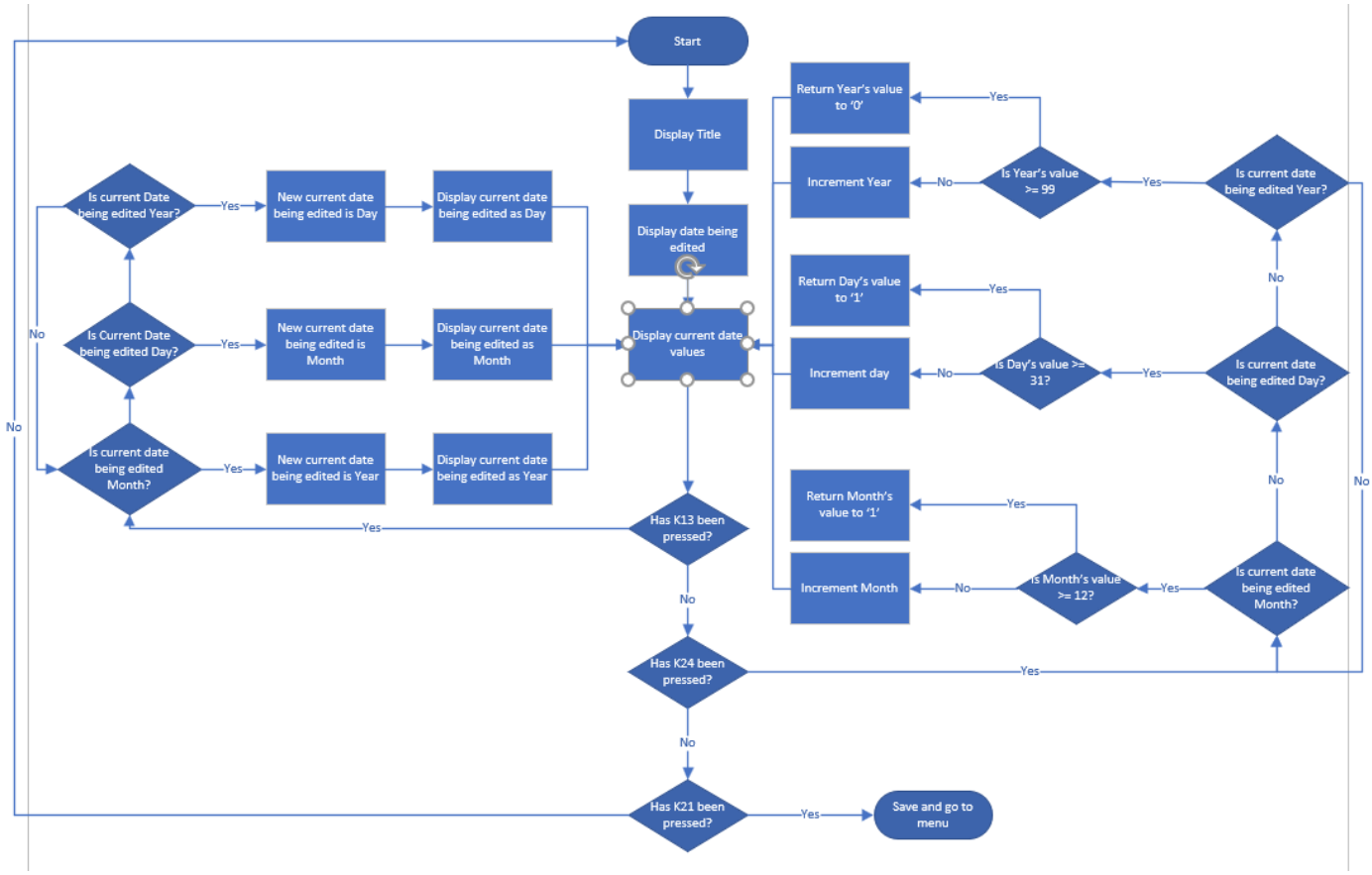
Testing table can be found in the appendices under Testing Table.

Logic Flow

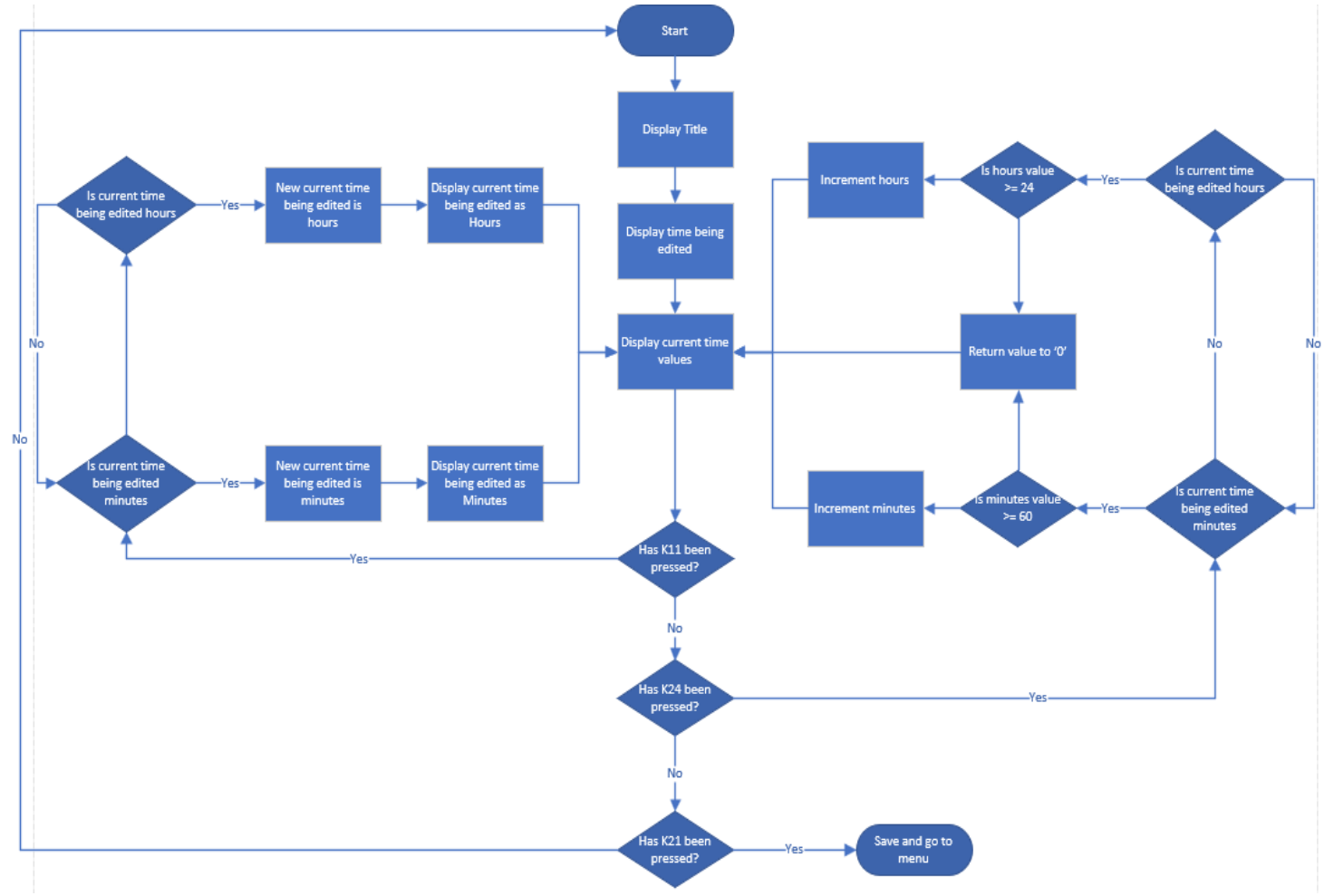
Menu



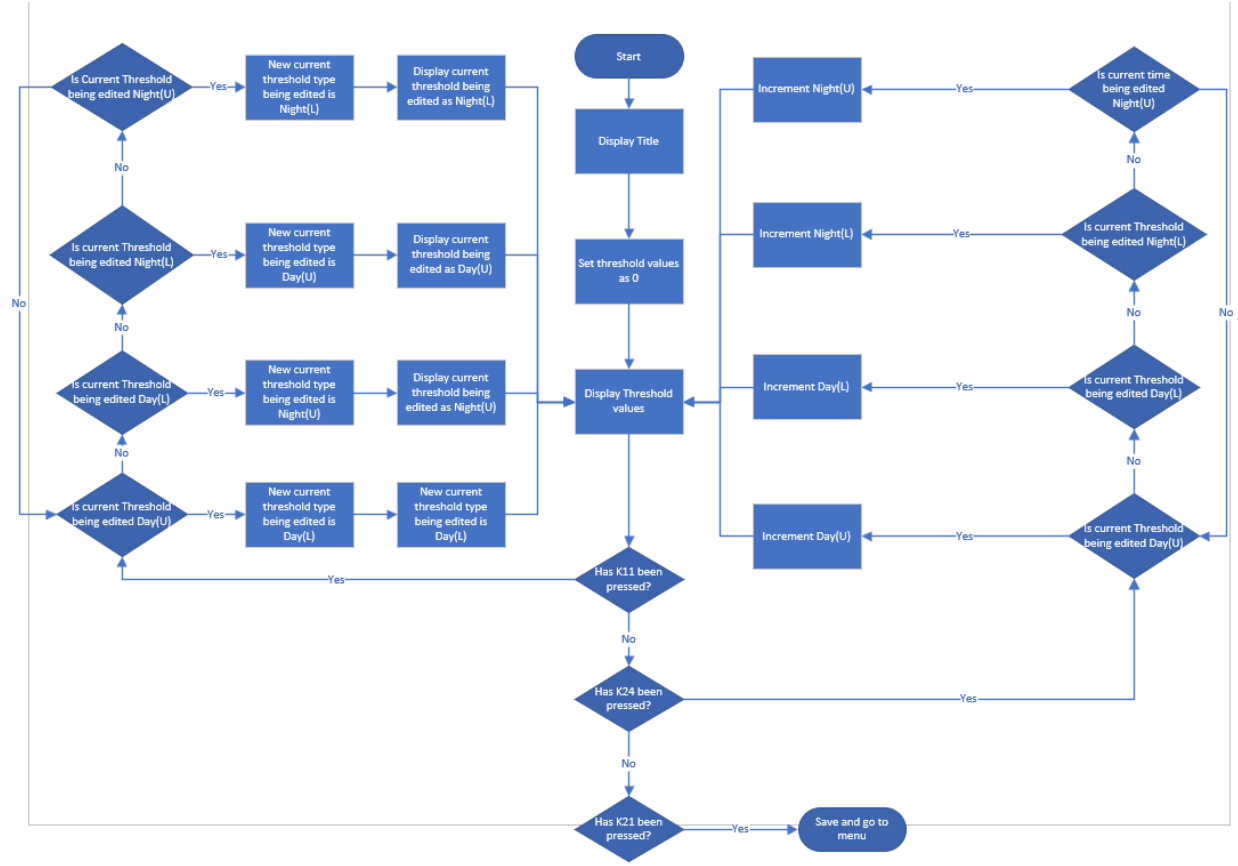
Set Date


Start
Display Title
Display date being edited
Display current date values
Is current Date being edited Year?
Yes
New current date being edited is Day
Display current date being edited as Day
Is Current Date being edited Day?
Yes
New current date being edited is Month
Display current date being edited as Month
Is current date being edited Month?
Yes
New current date being edited is Year
Display current date being edited as Year
No
No
No
Yes
Has K13 been pressed?
No
Has K24 been pressed?
Yes
No
Has K21 been pressed?
Yes
Save and go to menu
Return Year's value to '0'
Yes
Increment Year
No
Is Year's value >= 99
Yes
Is current date being edited Year?
No
Return Day's value to '1'
Yes
Increment day
No
Is Day's value >= 31?
Yes
Is current date being edited Day?
No
Return Month's value to '1'
Yes
Increment Month
No
Is Month's value >= 12?
Yes
Is current date being edited Month?
No
No

Set Time


Start
Display Title
Display time being edited
Display current time values
Is current time being edited hours
Yes
New current time being edited is hours
Display current time being edited as Hours
No
Is current time being edited minutes
Yes
New current time being edited is minutes
Display current time being edited as Minutes
No
Has K11 been pressed?
Yes
No
Has K24 been pressed?
Yes
No
Has K21 been pressed?
Yes
Save and go to menu
Is current time being edited hours
Yes
Is hours value >= 24
Yes
Increment hours
Return value to '0'
No
Is current time being edited minutes
Yes
Is minutes value >= 60
Yes
Increment minutes
No

Set Threshold



Critical Evaluation

The PIC is not capable of many complex tasks, although very simple is perfect for one specific job such as maintaining a temperature in a greenhouse. It would not be able to handle many complex tasks and would end up running out of memory. It is also very portable and very modular emphasizing its usefulness on specific jobs. It can be integrated in many systems and environments making it very useful for simple jobs. It also has very low power consumption due to its small design and architecture making it much more useful for simple programs, like traffic light systems or remote controls, if something that had more standard computer architecture was used for these instead it would end up consuming much more power and would be too powerful and way too costly for something so simple.

Lessons learned from programming for the PIC are that careful and efficient programming is required. The PIC cannot handle too much and only has limited space, for example instead of using floats or doubles we had use chars to make up each digit as it stored it much more efficiently due to the complicated calculations required for floating point operations. It also had a couple of hardware components that could only communicate using one pin. The PIC requires a lot of efficiency, It needs thorough testing and fool proofing in order to make sure that it continues running smoothly any number of small issues could mean that the PIC will not run properly, the most prominent of them all is memory.

Contribution breakdown

Contribution table can be found in appendices under Contribution Table

Operating Manual

Navigating the Main Menu

As seen in the image below the main menu displays the current temperature, the date, the time and the current day to go with the date. in order to change any of these settings you can press the following keys on the button matrix.

To Edit the Date press K13 (Please refer to figure 1 in the hardware and pin mappings section)

To Edit the Temperature Thresholds Press K12 (Please refer to figure 1 in the hardware and pin mappings section)

To Edit the Time Press K11 (Please refer to figure 1 in the hardware and pin mappings section)

To Reset the Alarm Press K10 (Please refer to figure 1 in the hardware and pin mappings section)

Red:

this line displays the current temperature to 2 decimal places.

Blue:

this line displays the date set by the user.

Orange:

This line shows the time set by the user.

Purple:

This displays whether or not the temperature has gone above or below defined thresholds, it is within safe boundaries it will display as ‘OK’, if it is too warm it will display ‘WARM’ or too cold it will display ‘COLD’ as well as an audible alarm which can be muted using the reset button which is as listed above

Green:

This line displays the current day attributed to the date set by the user.

Editing the Date

once in the date menu you will have a display as seen below. it will show the date you are editing whether it be day, month or year and the actual date that will be saved when you exit. you can increment the date, change which value you are editing or save and exit back to the main menu with the following keys.

To increment the current date value press K24 (Please refer to figure 1 in the hardware and pin mappings section)

To change the date value being edited press K13 (Please refer to figure 1 in the hardware and pin mappings section)

To save and exit to the menu press K21 (Please refer to figure 1 in the hardware and pin mappings section)

To exit without saving press K25 (Please refer to figure 1 in the hardware and pin mappings section)

Editing the Time

Once in the time menu you will have a display like seen below, it will show you which time value you are editing as well as the time you are changing to. You can increment the time value you are editing, you can change the time value you are editing and you can also save and exit back to the menu. You can do these things with the following keys:

To increment the current time value press K24 (Please refer to figure 1 in the hardware and pin mappings section)

To change the time value being edited press K11 (Please refer to figure 1 in the hardware and pin mappings section)

To Save and exit back to the main menu press K21 (Please refer to figure 1 in the hardware and pin mappings section)

To Exit to the Menu without Saving press K25 (Please refer to figure 1 in the hardware and pin mappings section)

Red:

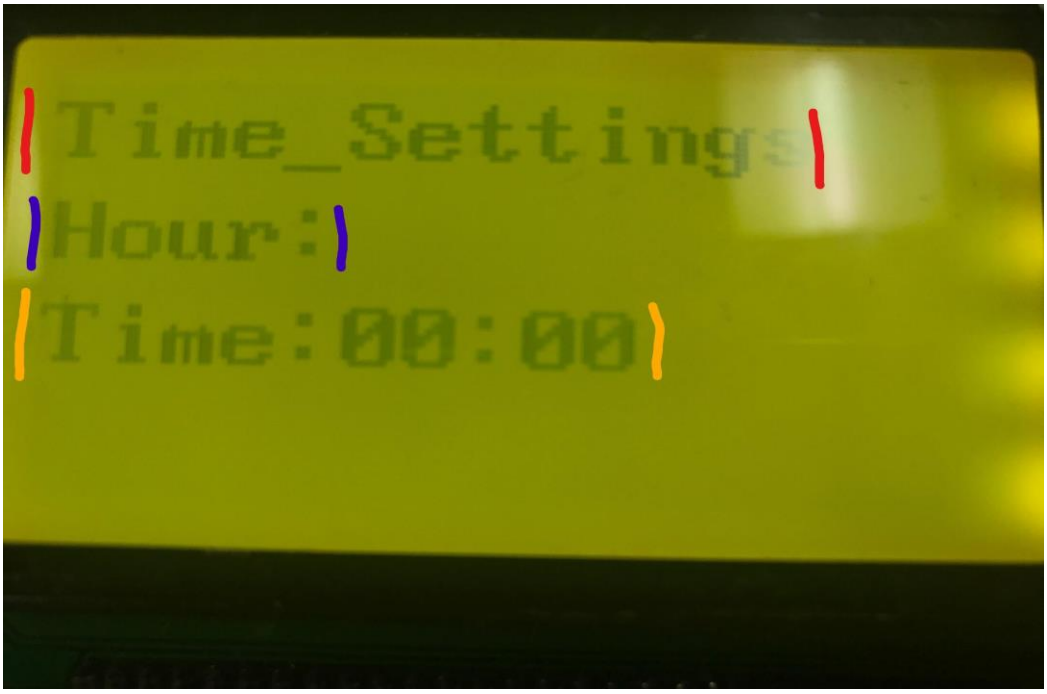
Displays the Date Settings title annotating to the user which menu they are in.

Blue:

Displays which current date value is getting edited which is currently the months

Orange:

Shows the current date set and what will be displayed when the user saves.



Red:

Time Settings Title to annotate which menu you are in.

Blue:

This displays which time measurement is being edited, in this case hours.

Orange:

Shows the current time settings that the user has designated, this changes when the user starts to increment and shows the time that will be displayed when saved.

Editing the Threshold

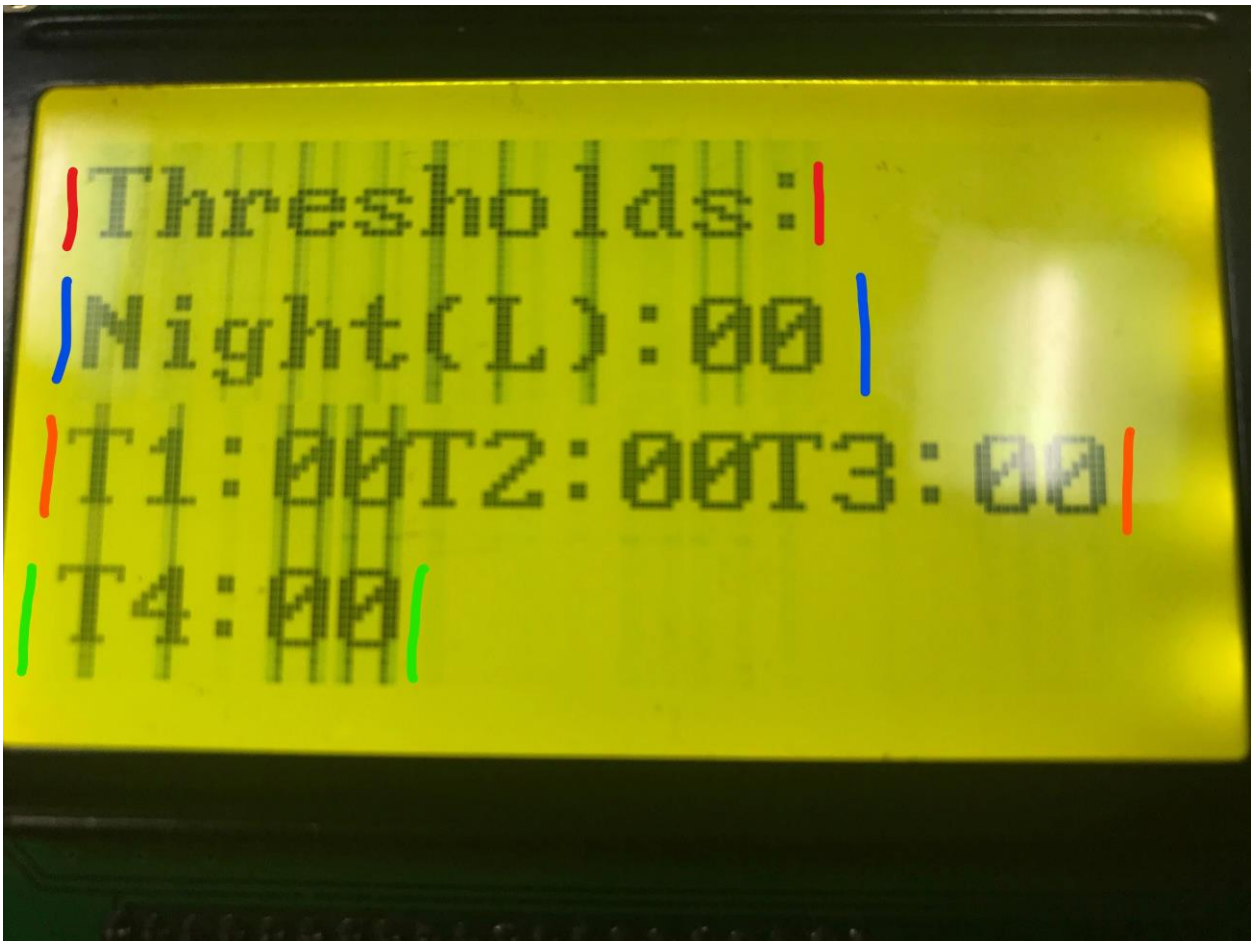
Once in the threshold menu there is a display as seen below. It shows the Threshold type you are editing as well as it's old value and the new value the user is incrementing it to. You can save and exit with the values you have inputted but **beware:** if you set the values to 0 then the system will believe that they are the thresholds and that the heating systems are failing. You can also exit without saving.

To increment the threshold value press K24 (Please refer to figure 1 in the hardware and pin mappings section)

To change the threshold type press K12 (Please refer to figure 1 in the hardware and pin mappings section)

To save and exit press K21 (Please refer to figure 1 in the hardware and pin mappings section)

To exit without saving press K25 (Please refer to figure 1 in the hardware and pin mappings section)



Red:

This displays the title of the menu the user

Blue:

Has different modes tied to it which include:

Night(L) : Indicator showing the user is currently setting a lower temperature threshold for nighttime

Night(U) : Indicator showing the user is currently setting a higher temperature threshold for nighttime

Day(L) : Indicator showing the user is currently setting a lower temperature threshold for daytime

Day(U) : Indicator showing the user is currently setting a higher temperature threshold for daytime

Orange and Green:

TI: Indicator for Day(U) showing value is being changed

T2:Indicator for Day(L) showing value is being changed

T3:Indicator for Night(U) showing value is being changed

T4:Indicator for Night(L) showing value is being changed

Surpassing the Threshold

When the threshold has been surpassed an audible warning will set off, this is because going either too high or too low has triggered a threshold and an alarm. In case of false alarm it can be reset with the press of K10 on key matrix this will silence the alarm. Returning the temperature to normal boundaries will also silence the alarm this can be done with the automatic activation of heaters and coolers.

To reset the alarm press K10. (Please refer to figure 1 in the hardware and pin mappings section)

Red:

This displays the current temperature which in this case has surpassed the set threshold.

Blue:

This displays the current set date.

Orange:

This displays the current set time.

Purple:

This displays a warning message telling the user which threshold they have reached. In this case the temperature is too high.

Green

This displays the day of the week associated with the date.

share code:

https://drive.google.com/file/d/1AFV4qSvhnWvQSGCptzNmc_5k-k0nrpbo/view?usp=sharing

Appendices

Contribution Table:

	Scott	Gabriel
Interface	x	
Thermometer		x
LCD 128x64 Panel	x	
Buzzer		x
Onboard Button Matrix	x	
Timekeeping Chip	x	
I/O Ports	x	
counter logic		x

Testing Table:

Test No.	Test Description	Expected Result	Actual Result	Pass/Fail (P/F)
1	Test to see if breaching the threshold displays warning message.	The left hand corner of the LCD should display WARM when the temperature is too high.	The LCD displays WARM perfectly fine.	P
2	Test to see if breaching the threshold sets of the alarm.	The buzzer should sound signalling the threshold is too high.	It sounds when the threshold is exceeded	P
3	Test to see if returning to an accepted temperature range displays 'OK' again on the LCD and silences the buzzer	The Buzzer should stop making noise and the LCD should display 'OK'	The LCD displays properly and the buzzer does not sound	P
4	Test to see if resetting the buzzer while above threshold will silence the alarm.	The buzzer should no longer make a noise.	The buzzer no longer sounds	P
5	Test to see if K13 leads to the Date_Settings menu	Pressing this button should lead the display to the Date_Settings menu	It does in fact transition to the Date_Settings menu	P
6	Test To see if K13 changes the date value being edited whilst in the Date_Settings menu	Pressing this button should change the value being edited between day, month and year	it transitions between the date values as expected	P
7	Test to see if K24 increments Month, Day and Year	Pressing this button should increment any of the date values that are currently selected	it increments on each of the 3 date values fine.	P
8	Test to see if the chosen date settings save when the button K21 is pressed	It should save keeping the data and displaying it correctly on the main menu	It displays correctly for generic values and has saved	P
9	Test to see if erroneous values save properly for month on Date_settings	Erroneous values such as 0 should not save	It saves as 0 which is not the expected result as there is no 0 month	F
10	Test to see if Month can	It should save and display 9	Strange results, tested a	P

	display boundary numbers correctly, i.e 9 and 10	and 10 perfectly fine	few times and worked perfectly except one time where it displayed 11 instead of 10, why was this test different I would call it a pass as it could be the pic board acting strangely	
11	Test to see if Year displays boundary numbers correctly, i.e 9 and 10 or 19 and 20	It should display these numbers with no issues	These numbers display perfectly fine	P
12	Test to see if Day within the Date_settings can accept erroneous values	Day should not be able to accept 0 as there is no 0th day in a month	It can save and display 00 which is not ideal	F
13	Test to see if Day within Date_settings can accept boundary values and display them correctly such as 9 and 10 or 19 and 20	It should display the numbers with no problems	it displays th values with no problems	P
14	Test to see if pressing K25 within the Date_Settings menu returns to the main menu	This button should return the user to the main menu without saving the changes	It does not save the values and simply returns to the main menu	P
15	Test to see if pressing K11 brings the user to the Time_Settings	This button should transition the LCD over to the Time_Settings	It goes to the time settings without any issues	P
16	Test to see if the button K21 saves the data edited in the Time_Settings menu	It should transition back to the		

Functions:

A list of functions used in Main.c

File	Function	Description

Button_driver.h	thermometer_threshhold_settings()	Allows the user to display and use the threshold settings. It displays the current threshold and allows the user to increment the temperature, it allows the user to set a day threshold as well as a nighttime threshold and allows for it to be saved.
Button_driver.h	date_settings()	Allows the user to change the date displayed on the LCD. The user can change the values they are changing to days, months or years. The user can then increment or decrement these values, if for example the user tries to go above 12 months it resets to 1. This function also allows the user to save these settings.
Button_driver.h	time_settings()	This function allows the user to change the time that is displayed on the LCD. The user can switch between the values they are changing to seconds, minutes or hours and can increment these values. If they go above any set limits such as trying to go to 60 minutes or 25 hours then it will set back to one. This function allows the user to save these settings to be displayed on the LCD
Button_driver.h	Initialise_buttons()	This function initializes all of the variables and pins that are used in the source file. Port A is set to digital and half of Port C is set to input while the other half is set to output. It also makes all of the variables used later on equal to 0 incase they carry over values they should not.
Clock_driver.h	ds1302 init()	This function initializes the clock pins and variables such as resetting the clock by pulling RA5 low and high again. It also sets serial clock input to low to stop it from doing anything and sends the control command to the clock and enables writing to it before resetting again.
Clock_driver.h	Set_time()	The set time function sends the write command to the clock and then using the function time_write_1 writes the time values into the table bit by bit until it makes a byte.
Clock_driver.h	Get_time()	Get time does the opposite of set time it sends the read command to the timekeeping chip before reading bit by bit the time from the table.
Clock_driver.h	display_clock()	Display clock displays the table values to the LCD using the write_char function.
Clock_driver.h	display_date()	Display date does the same as display clock but with

		the date instead of the time, it takes the table data and uses the write_char function to write to the LCD.
day_of_week_driver.h	Display_day()	This function will display a certain day on the LCD based on the number that is passed into it.
lcd_driver.h	Init()	This initializes many of the ports and pins used in the project such as setting port A as digital, Port D and E as output and setting Pin RA0 as input as well as opening port B for being pulled up.
lcd_driver.h	Lcd_init()	This initializes the LCD screen such as setting port A as output and telling the LCD to write left to right , display the cursor and set it to Parallel Mode.
lcd_driver.h	write_char()	This function writes a character to the LCD via Port D.
lcd_driver.h	write_cmd()	This function writes a command to the LCD via port D.
Math_driver.h	modulus_func()	the modulus_func() function performs a modulus onto the two values passed into it.
Thermometer_driver.h	Init_temp()	This function initializes Port E used with the thermometer. It sets Port E to output and sets the value of Port E to 0.
Thermometer_driver.h	display_temp()	display_temp displays the temperature value which is saved as 4 different values, 1 for each digit as it is to 2 decimal places, using the write_char function
Thermometer_driver.h	get_temp()	get_temp gets the temperature from the thermometer using the write_byte and read_byte functions. It firstly tells the thermometer to ignore ROM matching and then sends the temperature convert command. It reads the least significant half of the byte first and then reads the most significant half and then puts the digits it receives into values in order to store the temperature. Each digit is saved separately for later use.
Thermometer_driver.h	delay_temp()	This is a delay function that freezes the program for a specified amount of time depending on the 2 values passed in.

Threshold_driver.h	set_upper_threshold()	This function sets the upper threshold for the thermometer and will set the buzzer alarm off if exceeded, this will then trigger the cooler as well as displaying ‘WARM’ to the LCD.
Threshold_driver.h	set_lower_threshold()	If in the lower threshold it will display ‘OK’ on the LCD as well as silencing the buzzer. when in this ‘OK’ state it will also turn off the buzzer.
Threshold_driver.h	cold_threshold()	This function sets the lower threshold for the thermometer and will set of the buzzer alarm if exceeded, this will also trigger the heater as well as displaying ‘COLD’ to the LCD.
Threshold_driver.h	reset_alarm()	this function will reset the alarm with a specific button press, it will do this by turning off the alarm and displaying ‘RESET’ on the LCD

Code List for Main Logic:

```
#include "SystemHeader.h"

#include "LCD_driver.h"

#include "Thermometer_driver.h"

#include "clock_driver.h"

#include "Button_driver.h"

#include "Threshold_driver.h"

#include "Buzzer_driver.h"

#include "Day_of_week_driver.h"

#include "math_driver.h"

void main() {

  TRISB=0X02;

  init_temp();

  set_time();

  ds1302_init();


  init();                    //I/O init
```

```
lcd_init();                    //LCD init
init_temp();                   //Initialise temperature variables
initialise_buttons();          //Initialise button variables
var1 = 3;
var2 = 4;
var_night_low = 4;
var_night_high = 2;
cold_high = 1;
cold_low = 9;

while(1){
get_temp();                    //get temperature variables from device
get_time();                    //get time variables from hardware
write_cmd(0x80);
write_char('T');
write_char('e');
write_char('m');
write_char('p');
write_char(':');
display_temp();                //Display tempperature to screen

write_cmd(0x90);
write_char('D');
write_char('a');
write_char('t');
write_char('e');
write_char(':');
write_char(' ');
```

```
display_date();                 //Display date to screen


write_cmd(0x88);
write_char('T');
write_char('i');
write_char('m');
write_char('e');
write_char(':');
display_clock();                //Display clock settings
thermometer_threshhold_settings();   //access thermometer settings
date_settings();                //access date settings
time_settings();                //access time settings
write_cmd(0x98);
//day_threshhold
 if((temp_high+'0')<= 1 && (temp_low + '0') < 8){
alarm__ = set_lower_threshold(var1,var2);    //hot threshold
alarm__ = set_upper_threshold(var1,var2);    //normal threshold
alarm__ = cold_threshold(cold_high,cold_low);
}
//night_threshhold
if((temp_high+'0') >= 1 && (temp_low + '0') >= 8){
alarm__ = set_lower_threshold(var_night_high,var_night_low);
alarm__ = set_upper_threshold(var_night_high,var_night_low);
alarm__ = cold_threshold(cold_high_night,cold_low_night);
}
alarm__ = reset_alarm();
set_beep_threshhold(alarm__);
write_char(' ');
write_char('H');
```

```
    write_char(':');
    write_char(heater_state);
    write_char('C');
    write_char(':');
    write_char(cooler_state);
    for(int i = 0; i < 3; i++)
    write_char(' ');
    day_counter = modulus_func(day_low,8);
    display_day(day_counter);
}
}
```

Bibliography:

QL200_SCH. (n.d.). [ebook] Available at: https://canvas.hull.ac.uk/courses/54451/files/2071097?module_item_id=291630 [Accessed 13 Dec. 2019].