ECE323: FINAL PROJECT REPORT

System Integration using keypad, LCD, GPIO, ADC, Time Using LPC1114, keil C and C#

Adeyemo Joel

15th of December 2020

**Contents**

[Background 1](#_gjdgxs)

[Goals and Objectives 1](#_30j0zll)

[System Requirements, design specifications 1](#_1fob9te)

[Design and Implementation 1](#_4d34og8)

[Results 2](#_17dp8vu)

[Development Journal 2](#_3rdcrjn)

[Analysis and Discussion 3](#_26in1rg)

[Conclusion 3](#_lnxbz9)

[Acknowledgement 3](#_35nkun2)

[Appendix A 3](#_1ksv4uv)

# Background

This project is about building an embedded system that does multiple things using ARM Microcontroller. The ARM Microcontroller is a one of ARM7TDMI Cortex M chip architecture, designed by ARM ltd, to use little energy but a medium computation and processing power process. The Microcontroller used is an LPC1114FDH28/102 chip which is characterized by the limited amount of register compared to other chips, about four GPIO (General Input/Output) ports, SWDIO(Data I/O pin) port used for debugging, ADC(Analog digital converter), 32 and 16 bit count timer ,SSELO and about 16 PIO ports which can be repurposed for different things. Diagram of the chip is shown in the Appendix.

The system built for this project also included a LCD used to display data received to the LPC1114, a keypad used to send input data to the microcontroller, an ST-link that enables the program to be downloaded on the chip and be debugged, UART used to received and send byte data, an LED that use the system clock to tell the user how the data is entered and an LM35 temperature sensor. The data is coded using C/C# and assembly language.

The use of Exception handling (ie interrupt, Match event), subroutines and generation of pulses using timers is implemented in the design project.

# Goals and Objectives

The objective of this project is to take the temperature of a control unit and wirelessly transmit the temperature a GUI computer host using LPC1114 chip, keypad, LCD, UART using Bluetooth, C and C# GUI and LM35

The system is able to collect the current temperature of a place using the LM35 temperature sensor, then transmit that analog value through ADC (Analog digital converter) to the chip, and display that value on the LCD screen and at the same time on the GUI(Graphic User interface).

The system is also able to control the rate at which the temperature is sampled ie Sampling frequency, average that temperature and display that temperature at a particular period. The user inputs the value for sampling frequency (rate of collection of temp in Hz ), Display period (1 temp in how many seconds),and the average number(Average of the total number of collected temperatures) using both the keypad and GUI. They are all controlled using system timers and interrupts.

The basic operation of the system is, firstly the user inputs values for sampling frequency, Display period and Average num using the keypad or GUI. The chip receives those numbers, using timers, interrupt, event handling and voltage levels the LM35 temperature sensor collects the current temperature of the room and sends it back to the chip. The LPC1114 chip then converts it to a digital value using bits , then immediately sends that value to be displayed on the LCD and the GUI through the UART. From the GUI, the temperature gotten can be graphed. The collected temperature alongside the real time from the computer, the inputted sampling frequency, Display period and Average number can be sent and saved on an excel spreadsheet. This sampling process keeps on going until the user stops the sampling frequency using the “\*” from the keypad or GUI.

# System Requirements, design specifications

1. Once the system is powered on, “ECE 323 F2020” is on the first line of the LCD and the programmer's name must be on the second line.
2. After 5 seconds the LCD and GUI terminal must display the following

* A : Sampling frequency
* B :Display Period
* C: Average length
* D:Start, \*: Return

1. When the system is powered on, the Bluetooth module (UART) must be ready for connection
2. After A is pressed from the keypad or keyboard, “Enter Sampling frequency” must be displayed on the terminal or LCD.

4.1 Enter (Using keypad or Keyboard) and display sampling frequency in the second line LCD and the computer terminal

4.2 After B is pressed from keypad or keyboard, “Enter Display Period” must be displayed on the terminal or LCD

4.3 Enter and display the Display period on the second line of the LCD and computer terminal. it must be from 0.1 to 10 seconds (100ms to 10000ms). It must be controlled by a timer

4.4 After C is pressed, “Enter Average number” must be displayed

4.5 Enter and display the Average Number N the second line of the LCD and computer terminal. It must range from 1 to 1000. The initial average number for the display can be ADC value before entering the while loop

4.6 Press D button to start the sampling and display the average temperature

4.7 The temperature value must be sampled from the LM35 temperature sensor at the sampling frequency using a timer control.

4.8 Use a timer to control the display time and use another timer to control the ADC sampling time

4.9 N samples are averaged and to be ready to be displayed on the LCD or GUI terminal. If the sampling frequency is too slow and average is too high and the display rate is too fast, use whatever the calculated temperature for the display previously.

4.10 Repeat steps 4.7 to 4.9, so that the temperature is continuously sampled and displayed to show the temperature change

4.11 Pushing \* at any point in time return you to the main menu i.e. Step 1and 2.

4.12 The entering can be entered by “#” from the keypad or keyboard.

1. Based on Step 4, the temperature values must be able to plot continuously using C# GUI
2. Based on Step 5, there are text boxes for sampling frequency, display period and average number.

6.1 There are buttons to transmit the entered number to the LPC1114

6.2 There is a button to enable the system to run with set parameters in the GUI text boxes.

6.3 There is a button to enable continuously plotting of the temperatures

6.4 There is a button to stop the plot

1. The plotted temperature data can be stored in an Excel File. The format should include the time and temperature values with set parameters.

7.1 The displayed temperature should include one digit after decimal point on the LCD and terminal

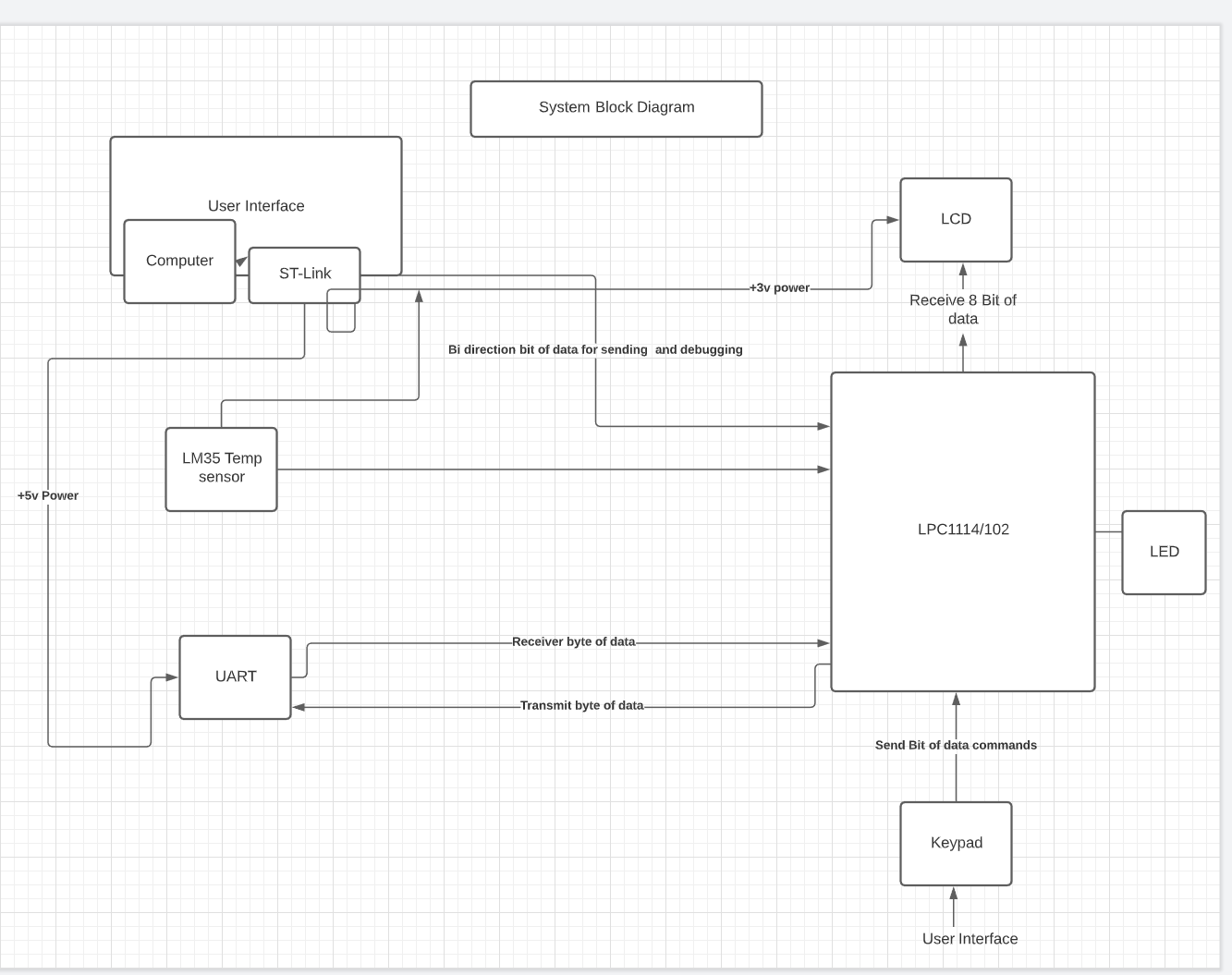
7.2 The time stamp in the excel file should be the real time of the computer : year:hour:min:sec

# **Design and Implementation**

Describe your design of the system. Contents can be how you derive certain constants used in the program; the system’s operation in block diagram, state diagram, schematics, flow charts, event charts; calibrations and hardware construction.

.

## **System Blocks**



ST Link

1. USed for debugging and downloading the program

Computer

1. Gives power to the whole system
2. Place where the code is written and includes the GUI interface

LM35 Temperature sensor

1. Using voltage levels to sense the temperature of the room

UART

1. Transmit byte of data from the Microcontroller chip to the computer GUI terminal
2. Receive byte of data from the GUI terminal and send it to the Microcontroller.

Keypad

1. used for input commands and character to the LPC1114 chip

LCP1114/102 chip

1. A microcontroller that has input and output pins, can convert temperatures using ADC built in, and Has built in timer that can simulate real time clock
2. Send and receive data to the entire system

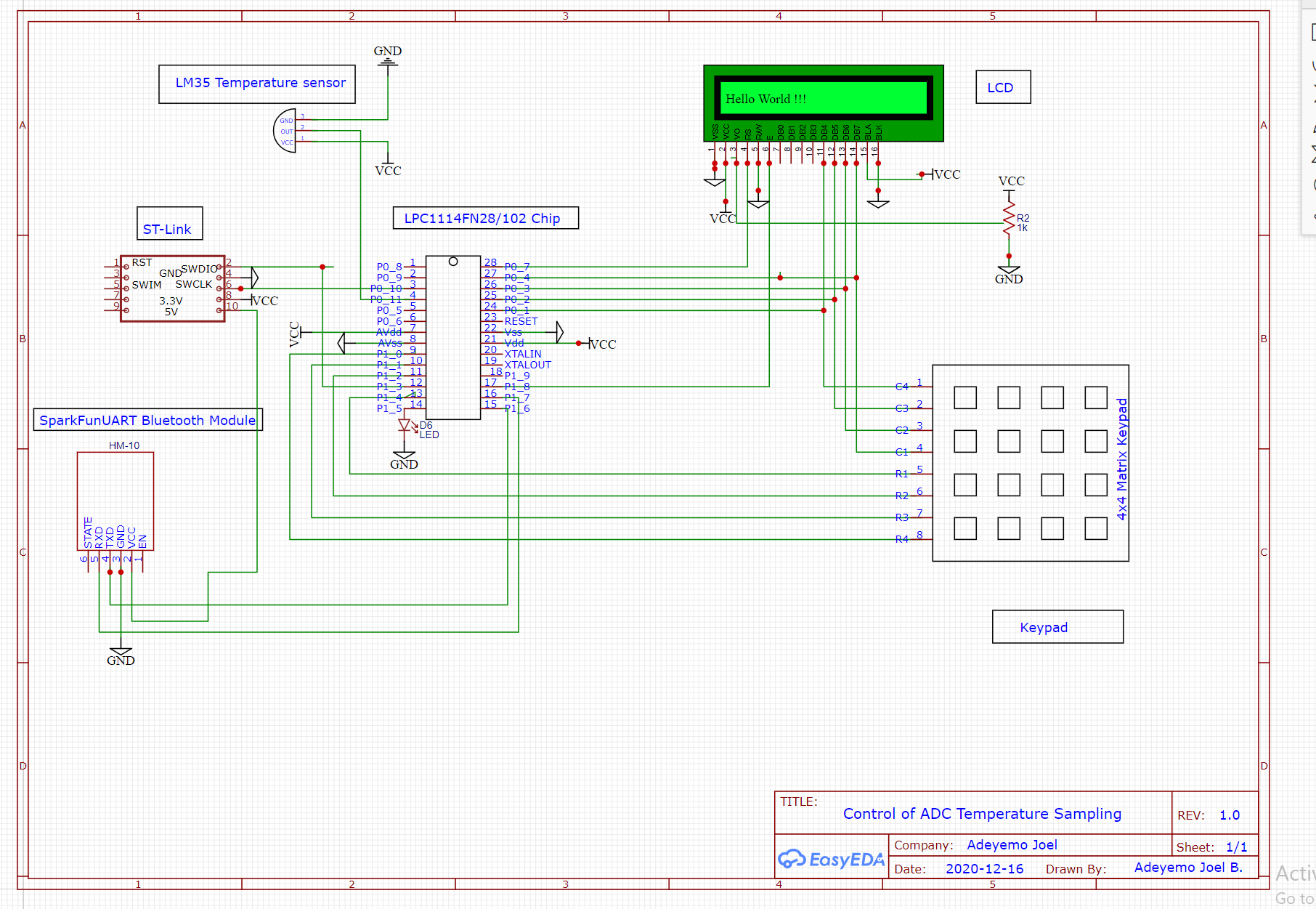
LCD

1. Display the inputted signals and characters, after being received from the Microcontroller.

LED

1. Simple diode that can be controlled using GPIO pins

## **Schematics**



ST-Link

1. SWDIO is connected to the chip which is a bi-directional wire that is used to both debug and send data to the chip
2. SWCLK is connected to the chip as the system clock
3. 3.3 V connection used to power the whole chip and other components
4. 5V connection to power the UART Bluetooth module

Spark Fun UART Bluetooth Module

1. TXD(Serial Transmit data i.e. output) from the UART is connected to the RX(Received) of the chip to transmit and receive byte data from GUI back to the chip
2. RXD(Serial Receive data ie Input) from the UART is connected to the TX(Transmit) of the chip to receive and transmit byte of data from the chip to the GUI
3. VCC is connected to the 5v of the ST-Link

LM35 Temperature Sensor

1. Out of the sensor is connected to the chip (PIO4\_P0\_11) which ADC control for the pin has been enabled. The temperature sensor measures 10mv for 1oC increase.

LCD

1. Pin 3 on the LCD is connected to a potentiometer to increase or reduce the contrast of display
2. Pin 4 on the LCD is connected to the Register selection (P0\_7) of the chip i.e. if Rs=0 then the instruction common code register is selected and if Rs=1 then the data register is selected
3. Pin 6 on the LCD is connected to the Enable(P1\_8) of the chip which enables after data has been sent
4. Pin15 and Pin 16 on the LCD is connected to VCC and GND respectively to turn the LED of the LCD on
5. Pin 11 to 14 on LCD is connected to (P0\_1 to P0\_4) respectively to each collect 4bits of data at a time (8bit in total)
6. Pint 1 and Pin 5 on LCD is connected to ground

Keypad

1. Pin 1 to Pin 4 are the column pins connected to the chip to help enable interrupt when the keypad is pressed
2. Pin 5 to Pin 8 are the Rows pins connected to the P1\_0,P1\_1,P1\_3,P1\_4 which are configured as general purpose pins and are used for the scanning process to know which key is pressed.

LPC1114FN28/102 Chip

1. P0\_8 to P0\_11 are natural GPIO pins.
2. P0\_11 is configured for ADC input
3. P0\_10 is configured as SWCLK (System Clock)
4. P1\_3 is configured as SWDIO (used for debugging and downloading of program)
5. P1\_5 is first configured as a GPIO (General purpose Input Output) which is now used for blinking of LED to show the display period
6. P1\_6 and P1\_7 are configured as RX and TX respectively which are used to transmit and receive byte data from the chip to UART to GUI terminal and vice versa.
7. P0\_1 to P0\_4 are configured as GPIO pins and used for both LCD and keypad. When the keypad is pressed, the Pins are configured for the keypad and when it is time to send data to the LCD, the pins are configured for LCD.
8. P1\_1,P1\_2,P1\_4,and P1\_0 are first configured as GPIO pins and are used as Rows for the keypad for the scanning process of the keypad.
9. P0\_X means for Port 0
10. P1\_X means for Port 1

## **Software Components**

Design and Code for Initializations

In the main of the keil C code different functions are initialized. These functions include LCD\_init, keypad\_init, timer\_init,UART init and Instruction Display.

Under the LCD\_initialization, the Pins 23 to pin 27 are configured as GPIO pins to enable the passing of bits while Pin 27(configured as GPIO first) is enabled as the register selection ie if the Register selection is 1 then the instruction been sent is in data mode else if register selection is 0 then it is in command mode. The data of the of these pins are first set to low using the code

LPC\_GPIO->DATA &=~(0x1E) // For the data line

LPC\_GPIO->DATA &=~(0x80) // For the RS(Register selection)

LPC\_GPIO->DATA &=~(0x100) // For the Enable pin

Then using the LCD \_command hexadecimal number are sent to the bit line of the LCD (Db4 to DB7) 4bits at a time and at the same time the RS is either 0 or1

For the Keypad initialization, Pins 24,25,26,27(all columns) and Pins 9,10,11,13 are all firstly configured as GPIO pins using LPC\_IOCON-> PIO0\_X depending on the port names of the pins mentioned. Then the direction of the columns are set to 0 to represent inputs while the dir of the Row are set to 1 representing outputs using the LPC\_GPIOX-> DIR where X is either 0 or 1. The level sensitive is also set using LPC\_GPIOX-> IS. The interrupt register for the pointer handler for the keypad is enabled.

For the UART initialization, the Pins 15 and Pin 16 that represent the RX and Tx respectively are configured as that using LPC\_IOCON-> PIO--. The power for the UART and is configured alongside the interrupt for receiving the data. In order to receive and transmit data, the control register is enabled using LPC\_UART-> LCR |=0x3. The configuration of the baud rate is calculated and stored in the LPC\_UART-> .

The Instruction Display simply uses sprintf and LCD\_print function to print the instruction on the LCD and the C# GUI terminal.

Design and Code for keypad control:

An external Pointer handler is used to enable an event or exception handling when the keypad is pressed. Pins 24,25,26,27(all columns) are all firstly configured as GPIO pins so that when the keypad is pressed at any time any one of those pins goes low (0)and the code goes into the interrupt pointer handler.

In the Pointer handler, firstly the interrupt register for the pointer handler is first disables and then the pending Interrupt register is cleared both are done by using NVIC\_DisableIRQ(EINT0\_IRQn) and NVIC\_ClearPendingIRQ(EINT0\_IRQn) then the scanning process begins. For the scanning process the Pins 9,10,11,13 are configured as GPIO.They are used to represent the rows and the DATA of the pins are turned to High(1). One after the other each row is turned to zero to see which key that corresponds to the column that was pressed. When it finds the key it sends the char to “key value”.

In basic terms, the way keypad knows what character is been pressed is that if the column and the rows are both 0 then that is the key i.e. let us say we pressed the key 5 on the keypad firstly the column 2 which is associated with 5 goes to 0 while other columns are still high , then the scanning process begins. Then each row is turned into 0 i.e. (first row would be 0111 and second row would be 1011) to find which row corresponds to the columns selected. It starts from row 1 see if any column is 0, then moves to row 2 sees if any column is 0 which in this case would be the second column which is 5 and it would display 0x35 in the register and wait for debounce (until all the columns are back to high) .

Flow chart design is shown in the Appendix.

Design and Code for Sampling Timer control

From the initialization, the timer is first initialized meaning the 32Bit timer 0 is reset then the system clock that deals with that timer is enabled, it is in timer mode and then the interrupt register for the timer is also enabled. below shows the code that does it:

LPC\_TMR32B0->TCR = 2

LPC\_SYSCON->SYSAHBCLKCTRL |= (1<<9); // Enables clock for 32 bit timer (CT32B0) Page 34

LPC\_TMR32B0->CTCR = 0; // timer mode not counter mode (Page 367)

NVIC\_EnableIRQ(TIMER\_32\_0\_IRQn);

The user first inputs a value for frequency\_init which goes into Match register value using:

LPC\_TMR32B0->MR0 =frequency\_init.

When a user inputs a value from the keypad it goes into the match register above. When the timer starts, by pressing D on the keypad, and after it has gotten to the desired match register number it goes into the timer32\_0\_ handler, collects the Digital values in LPC\_ADC->DR[0] puts it into ADC\_value and converts it to Celsius using temp = 3.33 / (1023 / ADC\_value) \*100.

There is a counter that keeps track if the average number is reached. if the counter is not equal to the average number inputted, it gets out of the handler and starts the counter again. If the counter is equal to the average number, the total temp is divided by the average number and kept into a variable called average using (average = running\_sum / Average\_num) formula. Now the counter is reset and total temp is also reset.

Flow chart design is shown in the Appendix.

Design and Code for Displaying Timer control

The same process as the sampling timer control but instead of timer32\_0\_ handler it is timer32\_1\_ handler. The user also has to input the value for Display period which would be converted to hertz and sent to the Match register for timer 1. When the counter reaches that match register values, it goes into the interrupt handler for the timer 32\_1.

First it checks if the average value is gotten if not it displays the current temperature already samples. If the average value is greater than 0, it first sets the GPIO pins to LCD mode, then uses the function LCD\_print \_number () to display the average temperature value obtained from the sampling timer control.

The process repeats itself until the timer is stopped.

Flow chart design is shown in the Appendix.

## **Application Workflow**

The General user interface (GUI) terminal includes a button that helps with the sampling process. These buttons include Start sampling, stop sampling, Graphing etc.

A) Start Sampling:

Firstly, numbers are inputted to the textbox for sampling frequency, display period and average number and using the enter button, each of the values are sent to the microcontroller. The sampling frequency is sent to the 32-bit timer 0 into MR0 after being divided by 48000000. Also the display period is sent to the 32 bit timer 1into MRO after being first converted from seconds to frequency and then 4800000 is divided by it. The average number gotten is sent to a global variable “Average\_num” .

When the start sampling button is pressed, it send the character “D” to the microcontroller, and when the microcontroller sees the character letter D it starts the timer for both of the 32bits timer 0 and 1 which handles the sampling of the ADC and display of the temperature respectively.

B) Stop Sampling:

When the stop sampling button is pressed from the GUI terminal it sends the character “T” to the microcontroller and when the microcontroller sees the character T from the UART it puts 0 into their individual MR0 for both the 32bit timer 0 and 1.

[Describe how your system handles each operation such as start sampling, stop sampling.]

## **Analog to Digital Conversion**

The LM35 temperature sensor is used for the sampling temperature for this project. The VCC and ground are both connected to pins 1 and 3 as shown in the diagram above and pin 2 is connected to the Pin4 of the chip which is already configured for ADC control.

The LM35 has a 10mv increase for every 1oC. The 10mv gotten is an analog wave which is stored in the 8bit input signal register called AD0[0:7]. Using the AD0 control register, the analog signals are converted into a digital one(i.e. High or Low) with an Op-Amp built into the chip and a reference voltage. The voltage signals obtained are in step size meaning depending on the reference voltage (Vref or input voltage) the register in which the bit is going to be stored would either be 1 or 0.

The step size deals with the resolution or accuracy of the data being received meaning if you want more accuracy you increase the number of the step size. it is represented by the formula Vref/2n, where n is the number of bits. For the project, the n used is 10 meaning 210 =1024. The way the process works is that firstly each of the 8bit AD0 register that has voltage level is goes into an interrupt one after the other and it is converted into a digital version (1 or 0) depending on the fraction of the Vref it falls under (i.e. if the voltage gotten is about 0.25 of the Vref the bit would be read as 1 else it would be 0) and then stored into a 10 bit register. The 10 bits data collected is stored in the register’s ADC DR0 [15:6]. The 10-bit registers are then multiplied by 0x3ff to remove excess bits because 8 bits is all that is needed. The 8bits is converted to hexadecimal then stored as an integer ADC\_value. The formula below would give the actual temp as integer values that we can read.

temp = Vdd / (1023 / ADC\_value) \* 100.

Flow chart design is shown in the Appendix.

## **C# Application**

A) How Control signal is sent From C# application.

Using C#, Bluetooth setting class is created under which \_serial is created also that represents the Serial Port resource in C#. When the button Bluetooth setting is pressed, the port name and Baud Rate are both already entered, and the user has to enter the COM\_number depending on the UART Bluetooth modem used. After the COM\_number is entered, using the connect button and \_serial. open, the UART becomes connected to the chip and the port can now receive or transmit byte of data. When this happens, the flashing red light turns green and the toolstripstatuslabel turns green.

When a character or string is typed on the GUI terminal i.e. “\*”, or “My Name:”, a richTextBox1\_Texchanged eventhandler is triggered. Firstly, the text length gotten and then each character is kept in a character array (ch[0]=send\_ch) and using

\_setting.\_serial.Write(character,index,length) to send each character one after the other through the serial port and bluetooth setting to the chip. Then the Function Write is invoked again, this time just sending a character (”\r”) which signals that you are sending a carriage.

B) How C# interprets Incoming data

Using C# and a visual studio under a new class made,a serialData received handler is created as an event. \_setting.\_serial.DataReceived field is created that represents the data that the data has been received through a port represented by the Serial port object.

A function called DataRecievedHandler receives byte data using SerialPort sp = (SerialPort)sender object and BytesToRead gets the number of byte data received from the serialport.

The byte of data is kept inside a buffer byte array. Which is extracted one after the other using an eventhandler method(update\_richtextbox1) so it has to avoid multiple threads error , then it is converted first into characters and subsequently converted to string. A new richtextbox event is created using eventhandler method in C# to print the string on the terminal and also receive any string on the terminal and send to the chip.

# **Results**

1. Stress test:

The system was made t o run for a long time to make sure that the system doesn't break at any point in time. Since the system didn't break then the result is that the system is very stable.

2. Heat and Cold Test:

While the system was plotting the temperature we held the top of the LM35 sensor very tight and released the temperature sensor. Also placing something very cold like ice close to the temperature sensor. The result of the experiment is that the temperature increases and decreases respectively showing that the temperature sensor is working.

3. Varied Parameter Inputs:

Multiple input for the sampling frequency, average number and Display period to test to see what happens. The result is that the system did not break while inputting those numbers or when the sampling started and that the system is stable.

4. Parameter Input using different source:

Input parameters from the terminal and keypad were tested at random to see if varied input systems would break the system. The result is that the system didn’t break, therefore the system is stable.

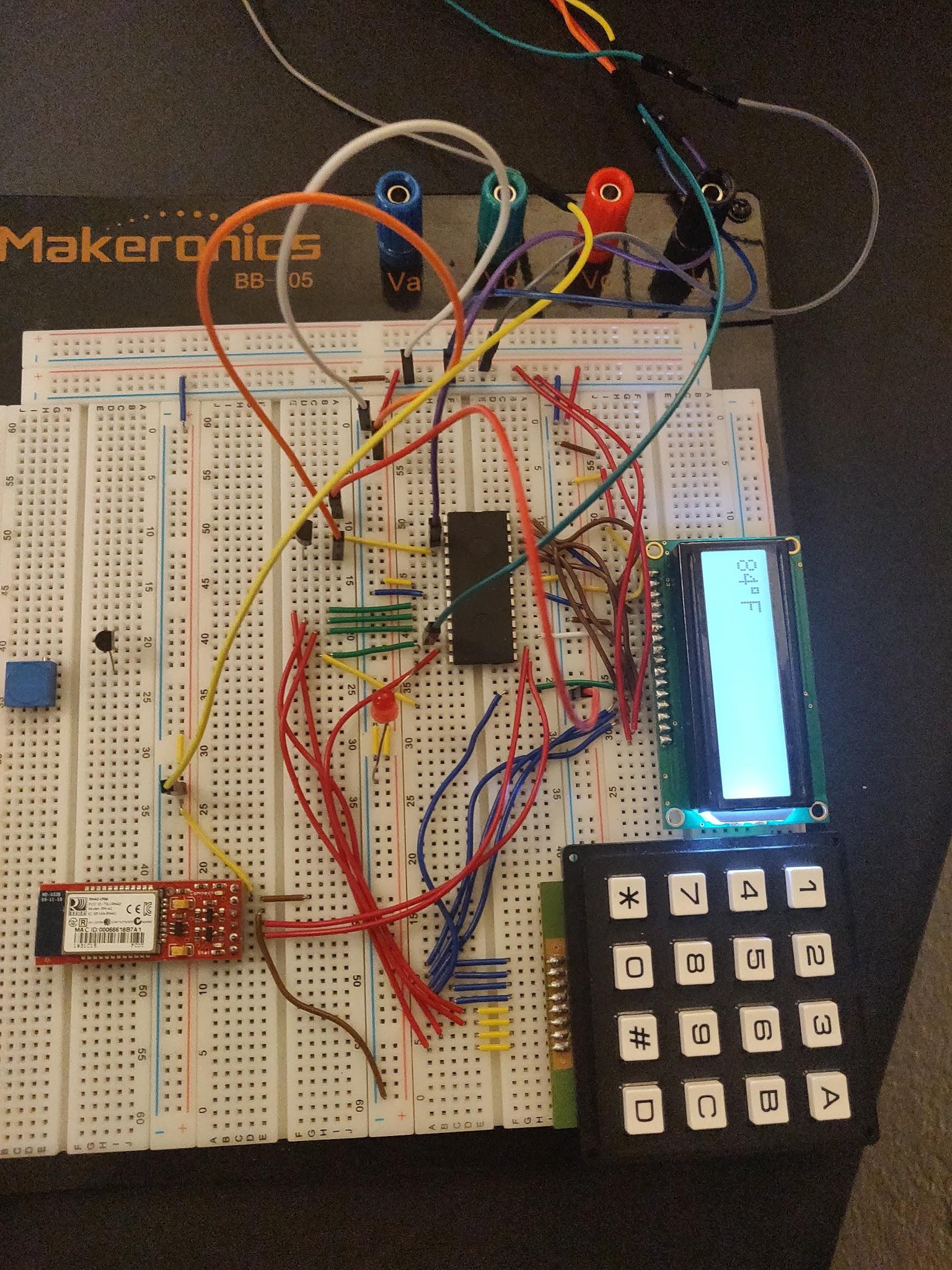


Fig 1.2 ( Photo of Hardware)

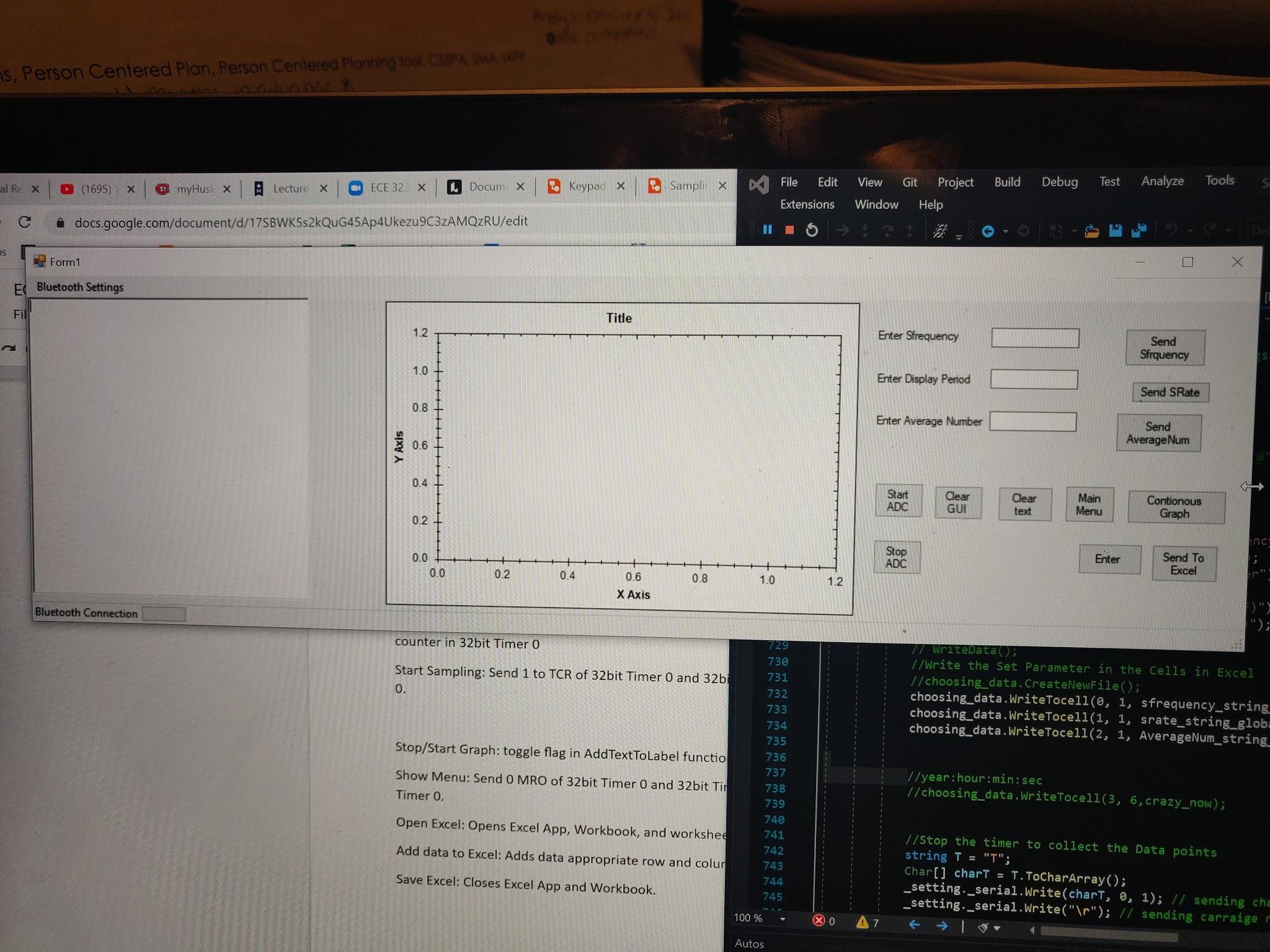


Fig 1.3 (picture of GUI)

# **Development Journal**

Include your development logs here.

|  |  |
| --- | --- |
| Date/Time | Description |
| 11/27/20  - 12 pm to 10pm | Initial reading of the design specification and how to write initial code in C for those specifications |
| 11/28/20  -12pm-4pm | ->Understanding display period, sampling frequency and average number alongside their correlation  -> Writing the code for keypad design specifications like when the numbers are inputted where they should be stored and how to send them to the UART and display on LCD at the same time. |
| 11/29/20  -12pm-2pm | ->Making sure that the inputted value are between bound of the design specification ie 0.1 to 10seconds  ->Break/ Rest day |
| 12/1/20  -10pm -3am | -> Creation of the two timer handler one for sampling through sampling frequency and the other for display using the display period entered  -> To do the above formulas were developed ie (when the period is entered in seconds, it is converted into frequency and then divided by 48000000(1hz using the system clock) to send into the Match Register for the timer) |
| 12/2/20  -10am to 4pm | -> Finding ways to extract numbers from a string with C# using functions like Regex and Remove.  ->Creating string and character arrays |
| 12/3/20  -10am to 5pm | -> Putting a collected temp into a string array and also plotting using list. Add();  -> learning about excel and finding out my computer doesn’t have any space for the download.  -> Creating buttons to send the value for the Sampling frequency, average number and display period to the microcontroller  ->Creating buttons to start and stop the ADC sampling  -> Creating buttons to plot the temperature using flags |
| 12/4/20  -Whole day | -> Worked on excel, first downloaded the excel object from Microsoft office.  -> Test how to create excel class and use a function to write to individual cells.  -> Testing the creation of things like new file, worksheet ,workbooks and save as() functions  -> Started the actual function under the button pressed  -> Firstly sent the Sample frequency, Display period and Average number to the excel when inputted from the GUI |
| 12/5/20  -Whole day | ->When the excel started plotting, using range function in excel the temperature values were either not showing or were showing up in like threes of each other  -> Decided to use for loops to individual put the temperature values in the exact cell i wanted |
| 12/6/20  -10pm -8am | -> Adjust the code in to fit the new design requirements, which incline bi-direction passing of data i.e. enter the values from both the keypad and the GUI terminal and can work both ways |
| 12/7/20  -Whole day | -> temperature plotting started acting up  -> Fixing temperature plotting to by changing the length of byte of data used for counting |
| 12/8/20 | -> Preparation for check off |
| 12/9/20  -1:30pm to 2pm | Check off day |
|  |  |

# Analysis and Discussion

For this project all design specifications are met these include:

1. Design using the initialization when the program is turned on
2. Sampling temperature, converting the temperature using ADC and Display the temperature on the LCD
3. Using the C# GUI terminal to plot temperature gotten through the UART Bluetooth module
4. Using buttons and textbox to input values for the Sampling frequency, Average number and Display period, send them to the Microcontroller through the UART and received the Temperature gotten through the LM35 temperature sensor
5. Sending the values of the temperature, the real time of the computer, and the values of the sampling frequency, average number and display period to an excel file through the use of a button.

These specifications are all met in the program and the results and test backs up the claim. This means the system is very stable. Also included in the code for both keil and C#.There are multiple checkpoints that help to prevent any errors or leakage.

Problems Encountered

1. Figuring out to transfer data in C# to excel is was hard
2. Kept running into out of bound error due to leakage of code
3. Extracting the values of the temperature, and plotting the value using zed graph

Resolution of problems

1. Reading articles and watching videos.
2. Fixed the leakage by reading up on windows components object C# and fixing the problem
3. Using Regex to extract values and plotting using List. Add.
4. Sending float number form the terminal to the UART and back

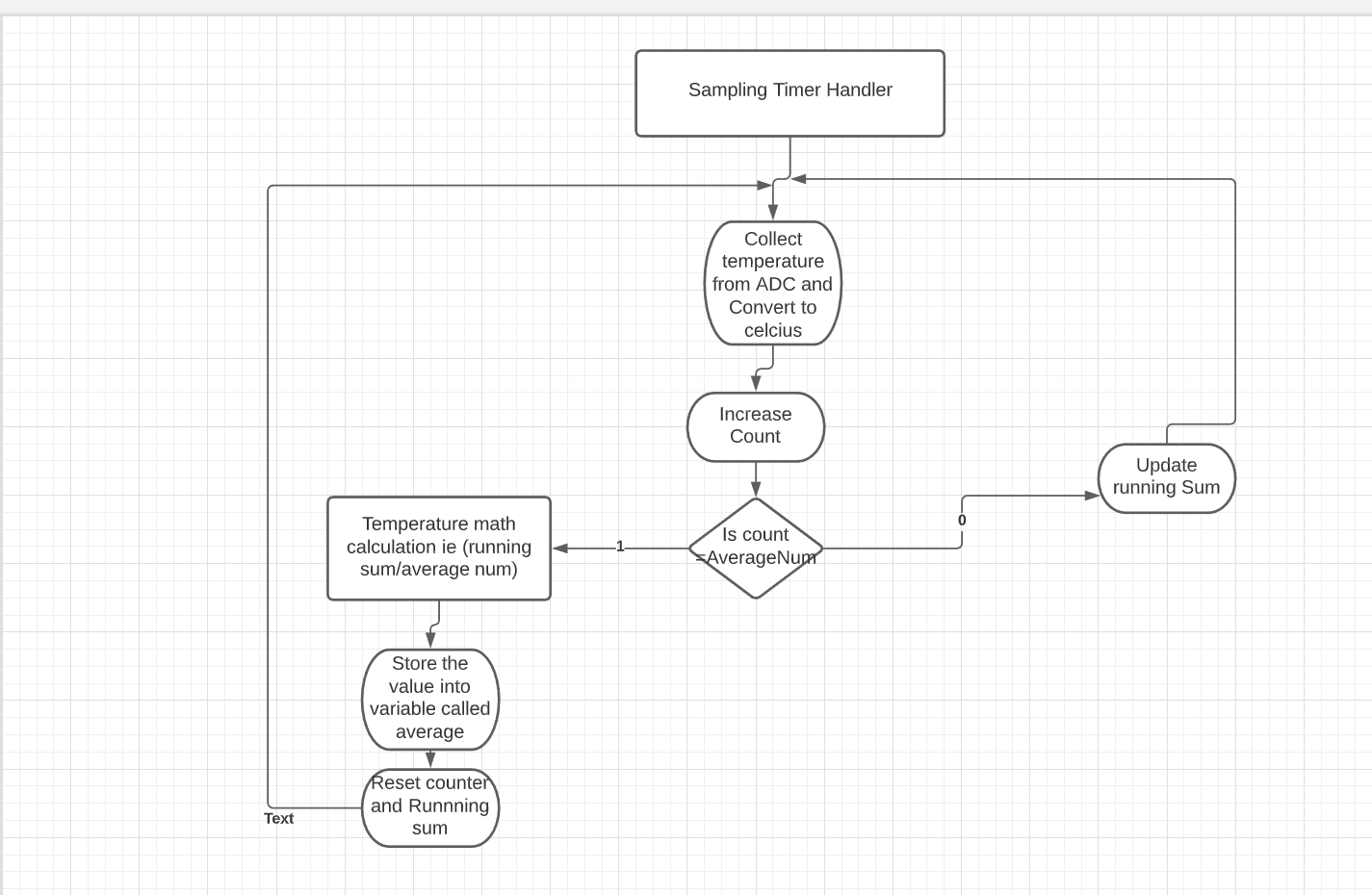
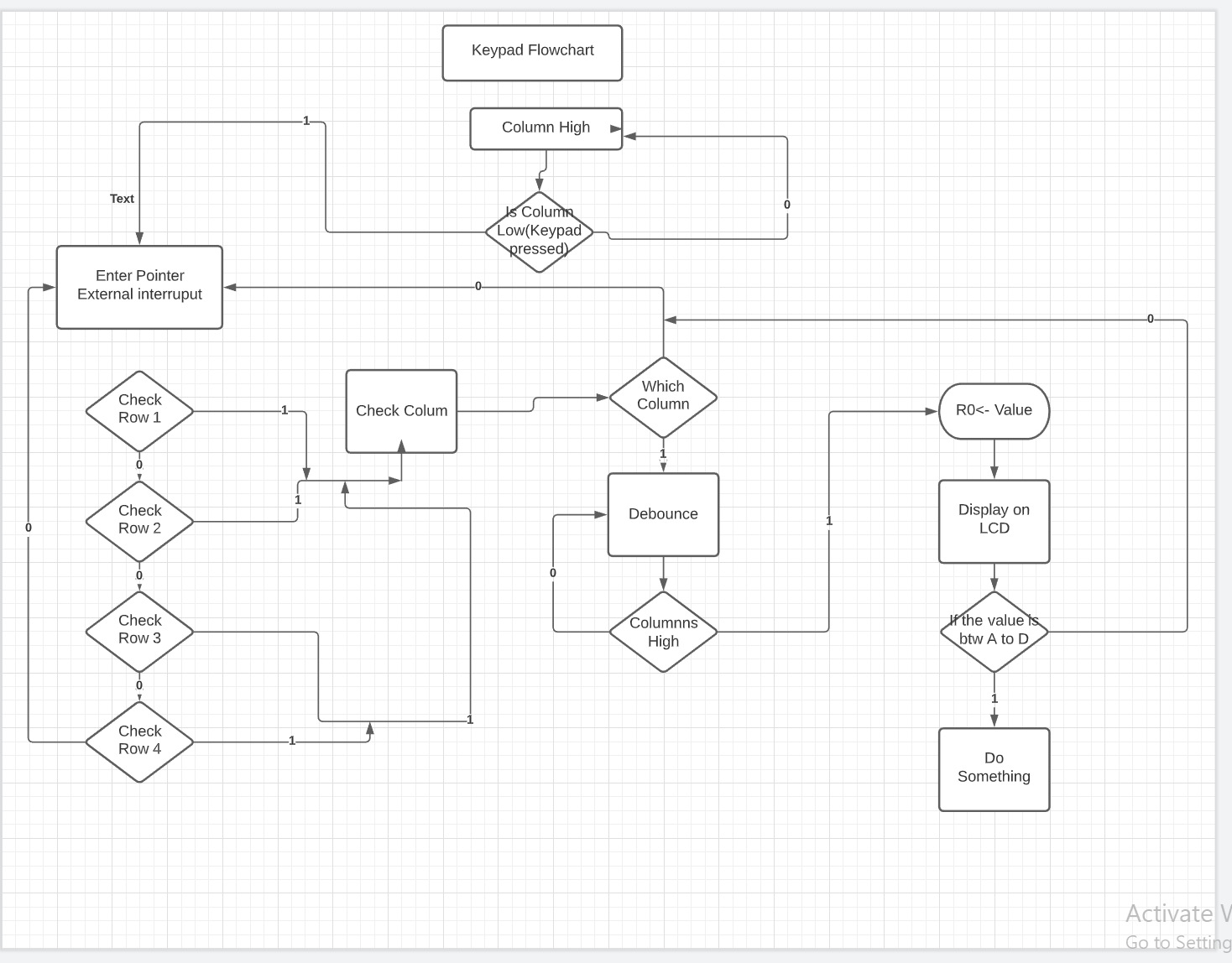
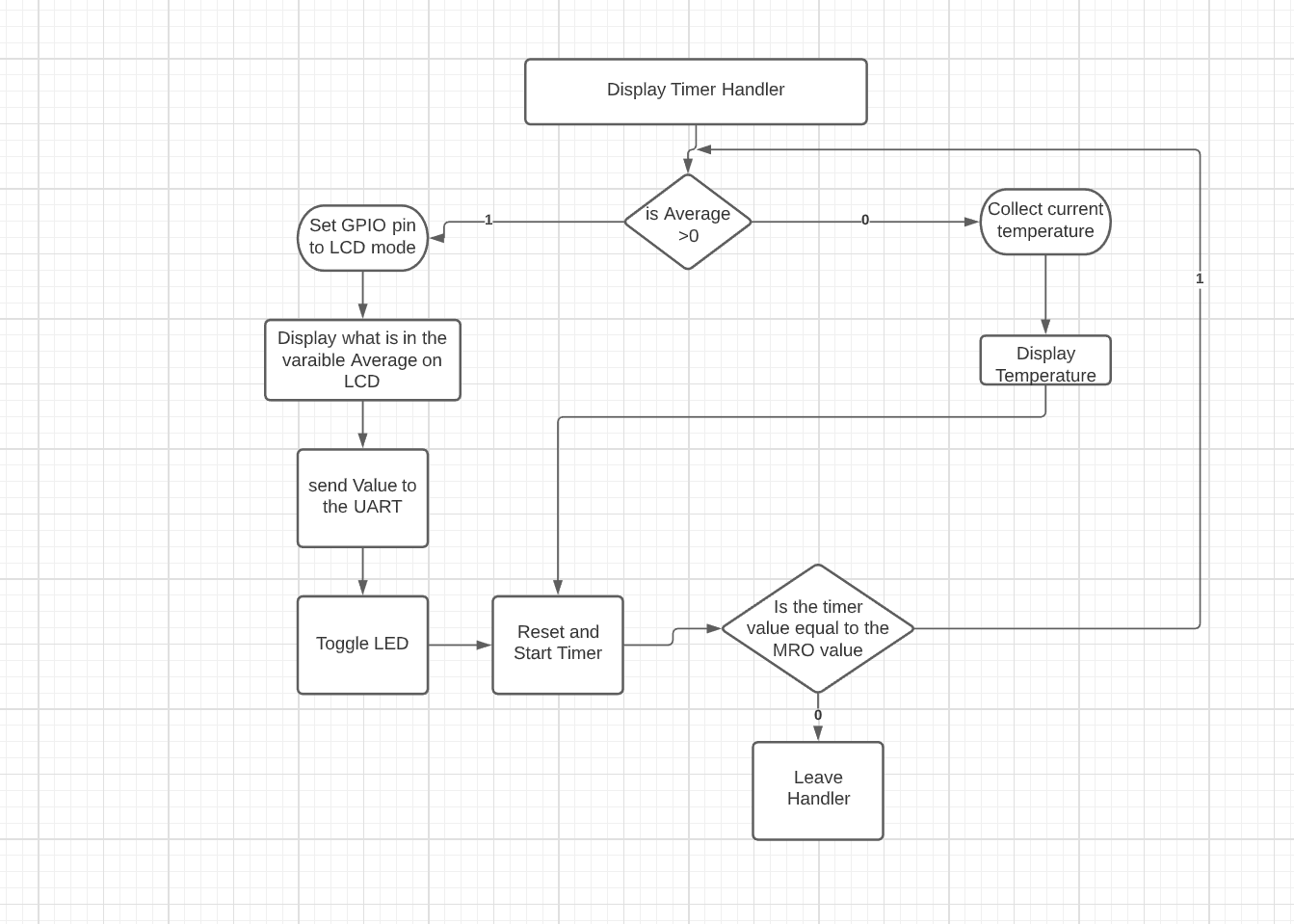
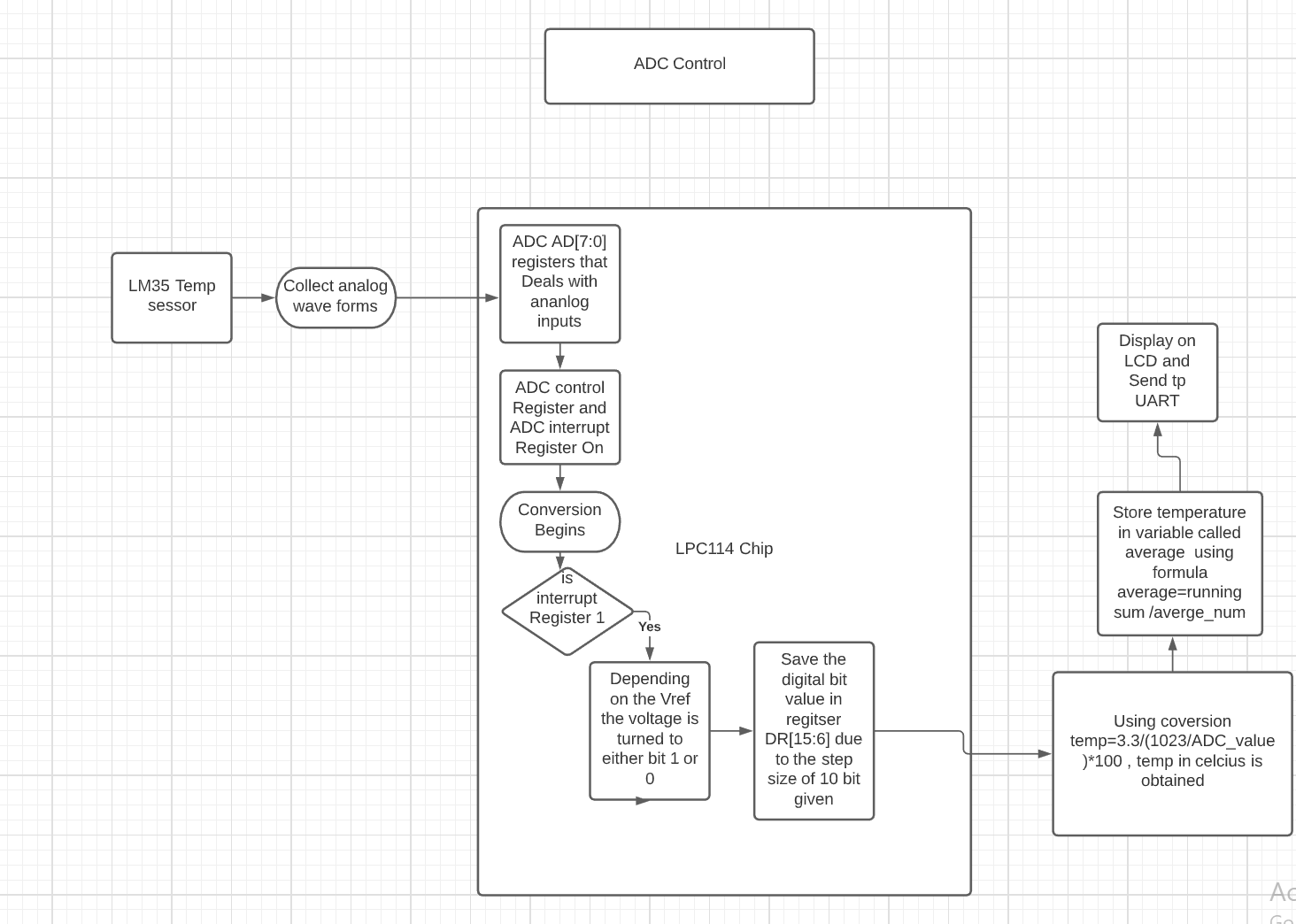
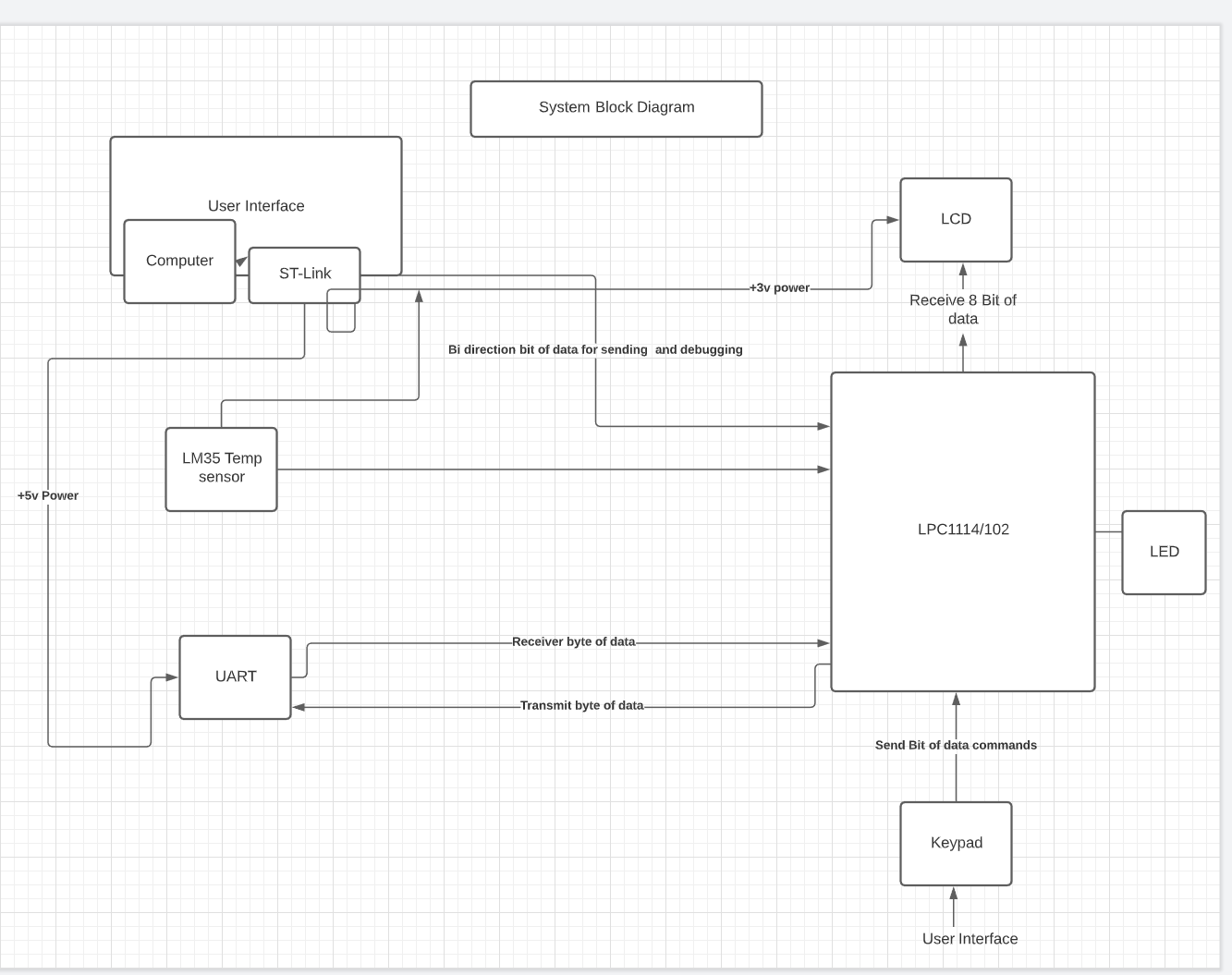
One way to improve the system would be to reduce the specification of the system. The system doesn't have any limitations and all specifications are met and the system is stable.

# Conclusion

The system is designed to interface with the LPC1114 , LCD, keypad, ADC and UART modem. System is fully functional and the system can do ADC sampling depending on the values inputted for the sampling frequency, wirelessly transmit those obtained temperature (in Fahrenheit) through the UART on a GUI interface. The code is written in both C and C#.

# Acknowledgement

# Appendix A



Codes;

Due to the length of the codes, the file of the code is zipped alongside the project Report.