**Sensor Monitoring System**

Rishab Shah

Final Project Report

ECEN 5613 Embedded System Design

May 1, 2021

1 Table of Figures 2

2 Introduction 3

3 Technical Description 4

3.1 Board Design 5

3.1.1 Component Placement 5

3.2 Power supply design 7

3.2.1 Brief Technical Details 8

3.2.2 Challenges & Learnings from Power supply design 8

3.3 AM2320 sensor interfacing 10

3.3.1 Technical Details ( firmware design perspective ) 11

3.3.2 Challenges & learnings from AM2320 11

3.4 SI7021 sensor interfacing 15

3.4.1 Technical Details ( firmware design perspective ) 16

3.4.2 Challenges & Learnings from SI7021 16

3.5 Nokia 5110 LCD interfacing 18

3.5.1 Technical Details ( firmware design perspective ) 19

3.5.2 Challenges & Learnings from Nokia 5110: 19

3.6 UART-0 interfacing 21

3.6.1 Technical Details ( firmware design perspective ) 21

3.6.2 Challenges & Learnings from Interrupt based Circular buffer implementation. 21

3.7 Firmware Design 22

3.7.1 Firmware Design Features 23

3.7.2 Module Description 23

3.7.3 Challenges & Learnings from Firmware development 26

3.8 Software Design 27

3.8.1 Challenges & Learnings from Python 27

3.9 Testing Process 28

4 Results and Error Analysis 34

5 Conclusion 35

6 Future Development Ideas 36

7 Acknowledgements 37

8 References 38

9 Appendices 39

9.1 Appendix - Bill of Materials 39

9.2 Appendix – Schematics 40

9.3 Appendix - Firmware Source Code 40

9.4 Appendix - Software Source Code 40

9.5 Appendix - Data Sheets and Application Notes 41

9.6 Appendix – Picture with Sensor Monitoring System project 41

# Table of Figures

[Figure 1 : Block Diagram of Sensor Monitoring System 4](#_Toc70793902)

[Figure 2 : Top view of the board 5](#_Toc70793903)

[Figure 3 : Bottom view of the board 6](#_Toc70793904)

[Figure 4 : Power supply schematic 7](#_Toc70793905)

[Figure 5 : LM2576 Inductor selection graph 7](#_Toc70793906)

[Figure 6 : L4931 circuit for conversion from +5v to +3.3V 8](#_Toc70793907)

[Figure 7 : AM2320 Temperature and Humidity sensor circuit 10](#_Toc70793908)

[Figure 8 : AM2320: Wakeup sensor I2C frame (Datasheet) 12](#_Toc70793909)

[Figure 9 : AM2320 : Wakeup sensor I2C frame SDA not pulled low 12](#_Toc70793910)

[Figure 10 : AM2320 : Wake up sensor corrected I2C frame using bit bang. 12](#_Toc70793911)

[Figure 11 : AM2320: Wakeup sensor (Datasheet frame) 13](#_Toc70793912)

[Figure 12 : AM2320 : Wake up sensor (Logic Analyzer trace ) 13](#_Toc70793913)

[Figure 13 : AM2320 : Send Read command I2C frame (Datasheet frame ) 13](#_Toc70793914)

[Figure 14 : AM2320 : Send Read command I2C frame (Logic Analyzer trace ) 14](#_Toc70793915)

[Figure 15 : AM2320 : Read Temperature and Humidity (Datasheet frame ) 14](#_Toc70793916)

[Figure 16 : AM2320 : Read Temperature and Humidity (Logic Analyzer trace ) 14](#_Toc70793917)

[Figure 17 : AM2320 : All the three frames, Wakeup, Send Read command, Read data. 14](#_Toc70793918)

[Figure 18 : SI7021 : Temperature and Humidity sensor circuit diagram 15](#_Toc70793919)

[Figure 19 : SI7021 : All 4 frames Humidity Write, Read Request & Temperature Write, Read Request 16](#_Toc70793920)

[Figure 20 : SI7021 : Read request for humidity data from the sensor (Logic Analyzer trace ) 17](#_Toc70793921)

[Figure 21 : SI7021 : Reading of Humidity data from the sensor (Logic Analyzer trace ) 17](#_Toc70793922)

[Figure 22 : SI7021 : Read request for temperature data from the sensor (Logic Analyzer trace ) 17](#_Toc70793923)

[Figure 23 : SI7021 : Reading of temperature data from the sensor (Logic Analyzer trace ) 17](#_Toc70793924)

[Figure 24 : Nokia 5110 : LCD circuit diagram 18](#_Toc70793925)

[Figure 25 : Nokia 5110 : PCD8544 internal architecture block diagram 18](#_Toc70793926)

[Figure 26 : Nokia 5110 : SPI Transaction 19](#_Toc70793927)

[Figure 27 : Nokia 5110 : Reset pulse sequence for LCD during initialization. 20](#_Toc70793928)

[Figure 28 : Nokia 5110 : LCD soldered the mail jumper side to the module. 20](#_Toc70793929)

[Figure 29 : Code snippet : Code architecture/Software architecture overview 22](#_Toc70793930)

[Figure 30 : Code snippet : Initialization routine module 23](#_Toc70793931)

[Figure 31 : Code snippet : Increase/Decrease polling interval. 24](#_Toc70793932)

[Figure 32 : Code snippet : SI7021 temperature and humidity measurement 25](#_Toc70793933)

[Figure 33 : Code snippet : AM2320 Temperature and Humidity Sensor 26](#_Toc70793934)

[Figure 34 : Output : 48.75 mv noise at +5V 28](#_Toc70793935)

[Figure 35 : Output : 4.8V to 5.1V variation 28](#_Toc70793936)

[Figure 36 : Output : 403.75mv noise at +3.3V 29](#_Toc70793937)

[Figure 37 : Output : 3.1v to 3.5v variation 29](#_Toc70793938)

[Figure 38 : Output : LCD showing temperature and Humidity values before testing 30](#_Toc70793939)

[Figure 39 : Output : Value increased due to lighting up of lighter near the sensor 30](#_Toc70793940)

[Figure 40 : Output : Increase in Humidity value when air is blown near sensor 31](#_Toc70793941)

[Figure 41 : Output : Live plotting of temperature and humidity data - 1 31](#_Toc70793942)

[Figure 42 : Output : Live plotting of temperature and humidity data - 2 32](#_Toc70793943)

[Figure 43 : Output : Storage functionality displayed using python 33](#_Toc70793944)

# Introduction

The world has moved today to a data driven approach. To take any control actions or redirect the course of an event feedback is required. The main crux of this data generation is sensors. These are the ones that provide the data to carry out analysis and take the control decision.

I wanted to explore this part of the field as well as learn the I2C protocol using register approach, wanted to use a graphical display for local display as well as transmit the data over UART to perform some data analysis.

Hence, in this project, a temperature & Humidity sensor (AM2320, SI7021), A power-supply is designed to lower it to 3.3V for powering the MSP-EXP432P401R board and a Graphical LCD (Nokia 5110) to display the acquired temperature (T) and humidity (RH) data. The dewpoint was calculated from the obtained T, RH data. Also, a python script was written to acquire data from the laptop’s COM port to show a graphical representation of the Temperature and Humidity data as well as to store it.

# Technical Description

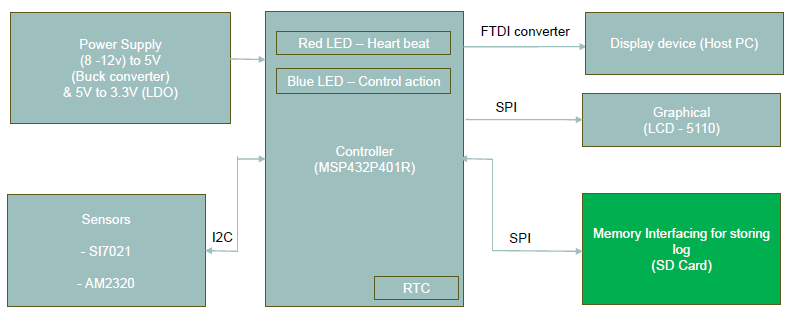


Figure 1 : Block Diagram of Sensor Monitoring System

In this project, a power supply is designed to provide a source of 3.3V to power on MSP.

The data Temperature and Humidity data is received from the SI7021, AM2320 sensors over I2C serial communication. This data is subjected to separate processing. As for the first part, the data is sent directly over to the LCD display over SPI protocol whereas for the second part, the temperature and humidity data is appended to the timestamp generated from the onboard RTC and is sent over UART using FTDI converter to the local PC for analysis. The temperature and humidity data are represented graphically using python as well as it is stored in the PC for diagnostic purpose.

Initially, it was planned (stretched activity) to store data in the SD card but due to time constraints owing to a lot of debugging that part could not be covered.

## Board Design

I had decided to assemble all the components over a perfboard through soldering. The perfboard size chosen was to accommodate the MSP432P401R, the power-supply circuit (9v to 3,3 v) sensors and LCD.

The layout was planned by assigning the upper left corner for power supply circuit and the upper right corner for the sensors.

It was decided by me to bring out +5v and +3.3V pin in the center of the board so that all the different components (sensors, LCD, microcontroller) can draw power by wire wrapping from the source. i.e., the approach was to create a power supply at center of the board with space reserved for additional +5 and +3.3 v supply lines as required in the future.

### Component Placement

In the below picture, Figure 2,

Red 🡪 Buck converter circuit

Yellow 🡪 LDO circuit

Blue 🡪 AM2320 circuit

Orange 🡪 SI7021 circuit

Green 🡪 LCD’s Current Limiting Resistor. The LCD module would have been fitted in the area indicated by green. Since, a damaged LCD was received, and the LCD received from senior was to be returned it was not mounted.

Dark Blue 🡪 Place where MSP would have been mounted.

Red LED 🡪 +5V supply

Green LED 🡪 +3.35V supply

The board design was planned for placement before soldering.

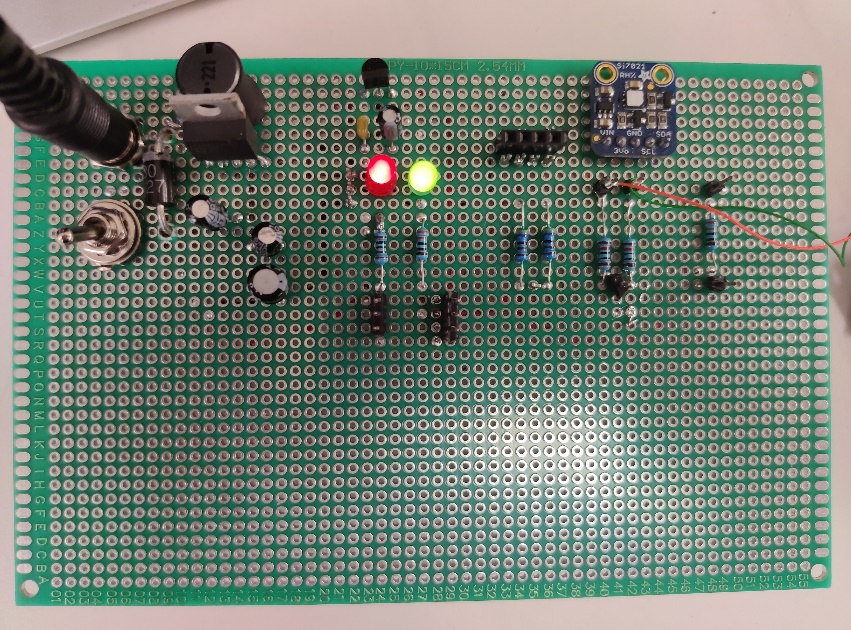
****

Figure 2 : Top view of the board

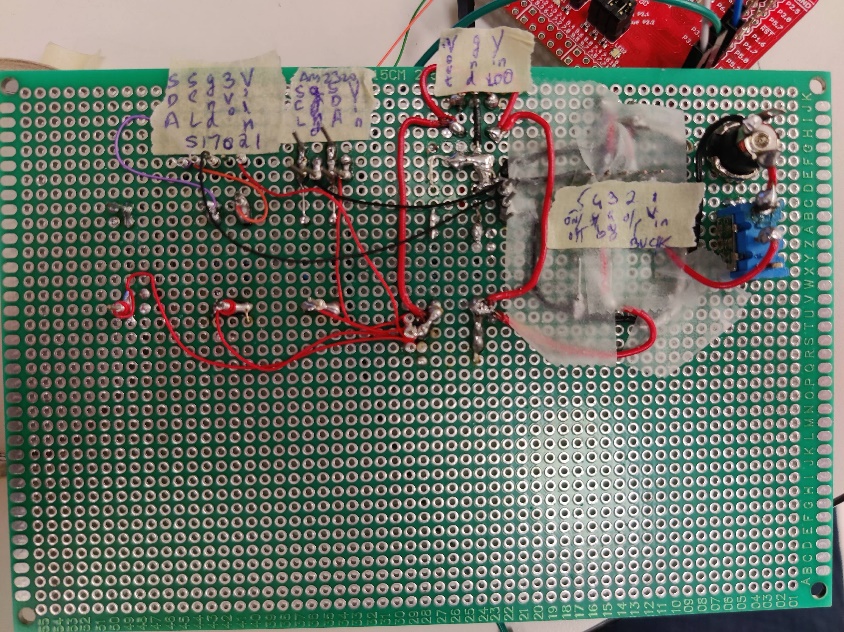


Figure 3 : Bottom view of the board

In the above figure 3, a cello tape layer is added to protect unwanted accidental shorting of Vcc and ground. Normal tape was used as insulation tape was not available.

## Power supply design

Following are the elements of the power supply design.

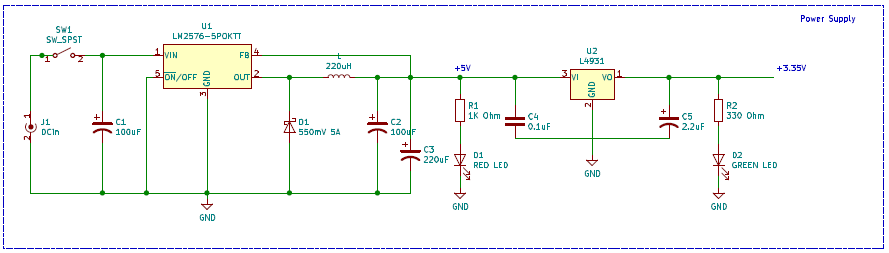


Figure 4 : Power supply schematic

The design was divided into two stages as it can be seen in Figure 4. First the power was dropped from 9V to 5V using a Buck converter and then from 5V to 3.3V using a linear dropout voltage regulator. The buck converter has feedback loop inside tit which gives a drop to 5V when in spite of variation in load resistance.

The reason for choosing a Buck converter + Low voltage dropout regulator was to achieve efficiency and scalability.

The C1 capacitor was chosen for filtering of any noise (bypass capacitor), ripples present at the source. LM2576-5POKTT is a buck converter which lowers the input voltage (Vin > Vout) to a 5V output voltage.

The inductor of 220uH was chosen to provide an output current of 1A [1] The inductor ensures that the current does not rise instantly. When I had decided to place order for power-supply components, I wanted to make a design which would be easily scalable in the future. Hence, went with 1A.

The below figure 5, is taken from LM2576 datasheet [1].

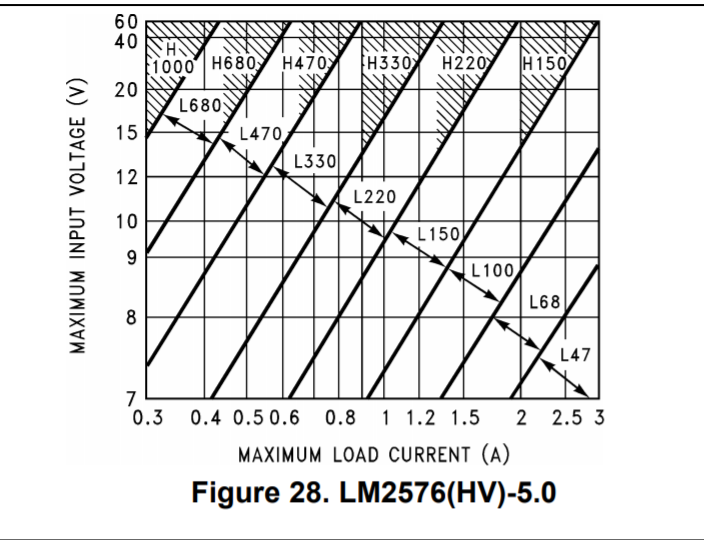


Figure 5 : LM2576 Inductor selection graph

It was recommended to choose a diode such that the reverse voltage rating of the diode must be at least 1.25 times the maximum input voltage. Use a 30-V 31DQ03 Schottky diode, or any of the suggested fast-recovery diodes. [1] Hence, selected an alternate diode with 40v rating SB540FSCT in my design. This diode ensured a low forward voltage drop which will ensure that there is a steady flow of current to the load.

A 330uF capacitor was chosen at the to ensure that a constant voltage supply of 5V is available at the output.

The L4931 was chosen to provide a current of 250 mA and voltage supply of 3.3V to power the MSP. The 0.1 uF capacitor is chosen to provide any noise filtering whereas 2.2 uF capacitor was chosen to provide a constant voltage supply of 3.3V.

The below figure 6 is taken from the datasheet of L4931[2]

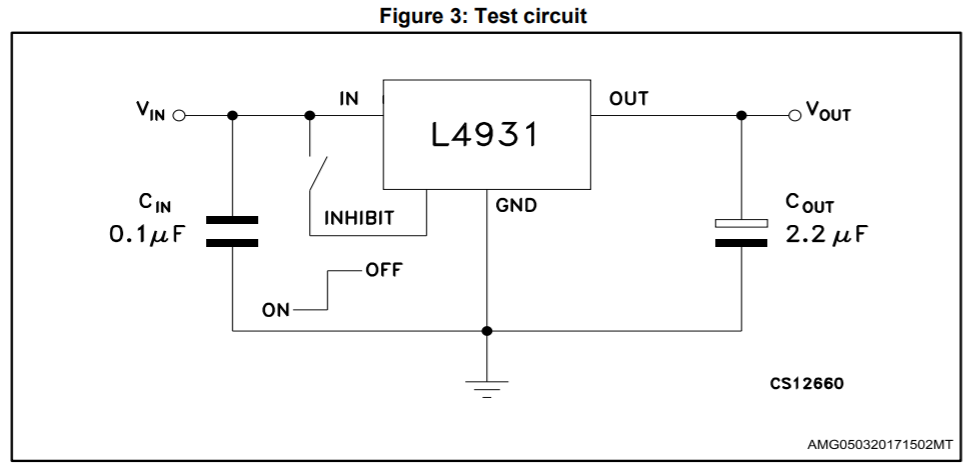


Figure 6 : L4931 circuit for conversion from +5v to +3.3V

I placed the RED Led along with a current limiting resistor of 1K Ohm to indicate 5V whereas used green LED with 330 Ohm as Current limiting resistor to indicate +3.35 V output.

### Brief Technical Details

RED LED +5V indication and current of 1A

GREEN LED +3.35V indication and current of 250mA

The MSP432P401R is powered from this power supply circuit designed using the +3.35V output.

### Challenges & Learnings from Power supply design

1. It was not easy to find out through-hole components. Majority of the components were SMD. A lot of research had to be done to find a through-hole component as I was using a perfboard.
2. The schottky diode chosen had a thick diameter and this parameter was not considered while choosing the diode.
3. The inductor used was a phase inductor. This was a new learning. I realized that the part which is represented by a dot has to be phase aligned with the output and it should be connected to the output of the regulator.
4. There was a requirement of 330 uF at the output of Buck converter. Since, such a large value capacitor was not available I used a parallel combination to achieve the same.
5. In future, I would be ordering capacitor after consideration of derating capability. The order was already placed for capacitor before it could be considered as a factor in design.
6. During the connection of L4931 regulator, I misinterpreted the top view for bottom view, and I made a wrong connection which swapped pin1 and pin 3. I had to redo the entire connections leading to a time loss.
7. It is not easy to create solder tracks on the perfboard. Hence, used thick gauge wires. It consumed a lot of time.

***To summarize, I learnt a lot about how to make proper selection of a component and as well as look for the minute details in the datasheet.***

## AM2320 sensor interfacing

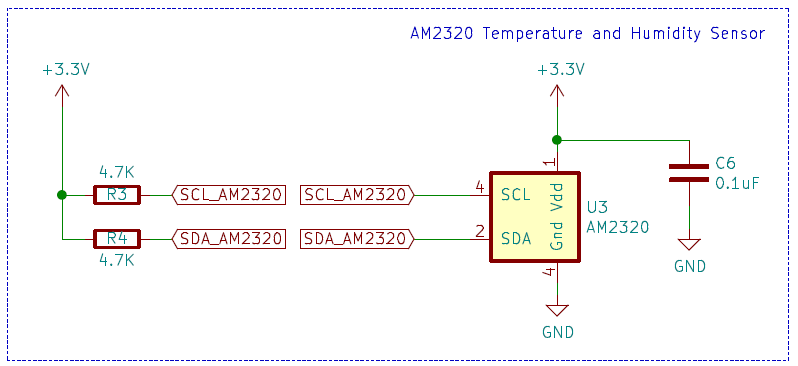


Figure 7 : AM2320 Temperature and Humidity sensor circuit

The AM2320 sensor was chosen for the following capabilitites:[3]

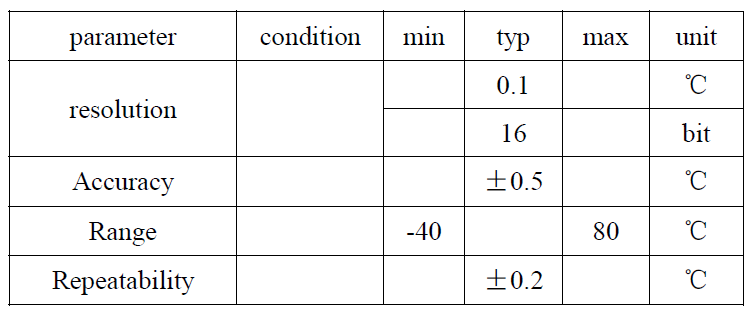
1. The datasheet was crisp in terms of the I2C frames that are to be sent to the sensor.
2. The device went in to sleep automatically when it when there was inactivity for 3 seconds.
3. The baudrate, power supply ratings , Vdd and pull-up resistor value were clearly mentioned.
4. There were ample resources online to figure out any issue in case of debugging.
5. It is also reasonably priced for the features it provides.

The device has sufficient accuracy and resolution for prototype development

The device has following characteristics for Humidity:[3]



The device has following characteristics for Temperature: :[3]



### Technical Details ( firmware design perspective )

Following are the pin connections to the MSP432P401R apart from Vcc and ground:

Clock frequency : 40Khz

P1.6 UCB0SDA SDA of AM2320

P1.7 UCB0SCL SCL of AM2320

AM2320\_I2C\_ADDRESS 0x5C

AM2320\_BITBANG\_ADDRESS 0xB8

Device address : 0xB8 ( In code 0x5C) is used due to the left-shift opertion behaviour of the I2C registers.

AM2320\_FUNCTION\_CODE 0x03

AM2320\_START\_ADDRESS 0x00

AM2320\_REGISTER\_LENGTH 0x04

AM2320\_I2C\_DATA\_READ\_BYTES 7

This was the first device I communicated using I2C based upon register approach. Working with this device, gave me a deeper understanding on debugging and usage of logic analyser and importance of single run mode feature of oscilloscope to analyse the frames.

### Challenges & learnings from AM2320

1. While coding for the I2c I mixed both the bit-banging concept and register address. This caused me to send a 8 bit data along with the read and write bit. This lead to a false addressing of the device. The device was not addressed correctly as incorrect address was sent on the bus. It was later after reading the datasheet diligently, I realised that the MSP’s I2C state machine automatically manages and sends the read and write bit in the frame as per the registers we select.
2. Usually, what address is given in the datasheet of the slave has to be put in the slave address but for this device things were different. The address of the device was 0xB8 were as the value that needs to be fed into the I2C slave register was 0x5C [a]. This was realised after going through the online arduino reference. This is how I started realising that how arduino is used to speed up the development time. I had not used arduino before.
3. The sensor worked properly as soon as the address was corrected and also when corrections were made along the way in registers to send the appropriate instance at which stop bit , nack , waiting for ack has to be configured. This process gave me indight into communicating using I2C registers.
4. Next day, I modified the code to poll continuously at 2 second interval. There were no changes in the register settings or the sequence in which they were set. Still, the behaviour was unreliable. I reliased that the SDA was not pulled low.

Datasheet: Wakeup sequence. :[3]

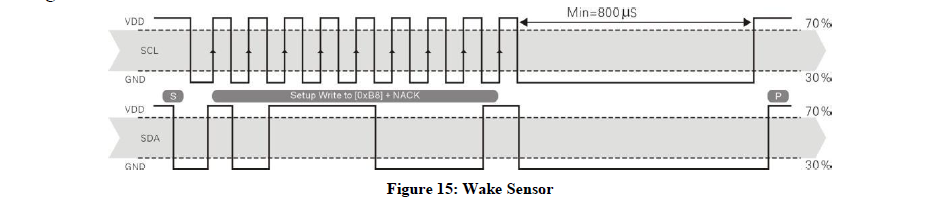


Figure 8 : AM2320: Wakeup sensor I2C frame (Datasheet)

With register approach in below image it can be seen that SDA is not pulled low. SDA is above ,SCL is below in the following logi analyser screenshot

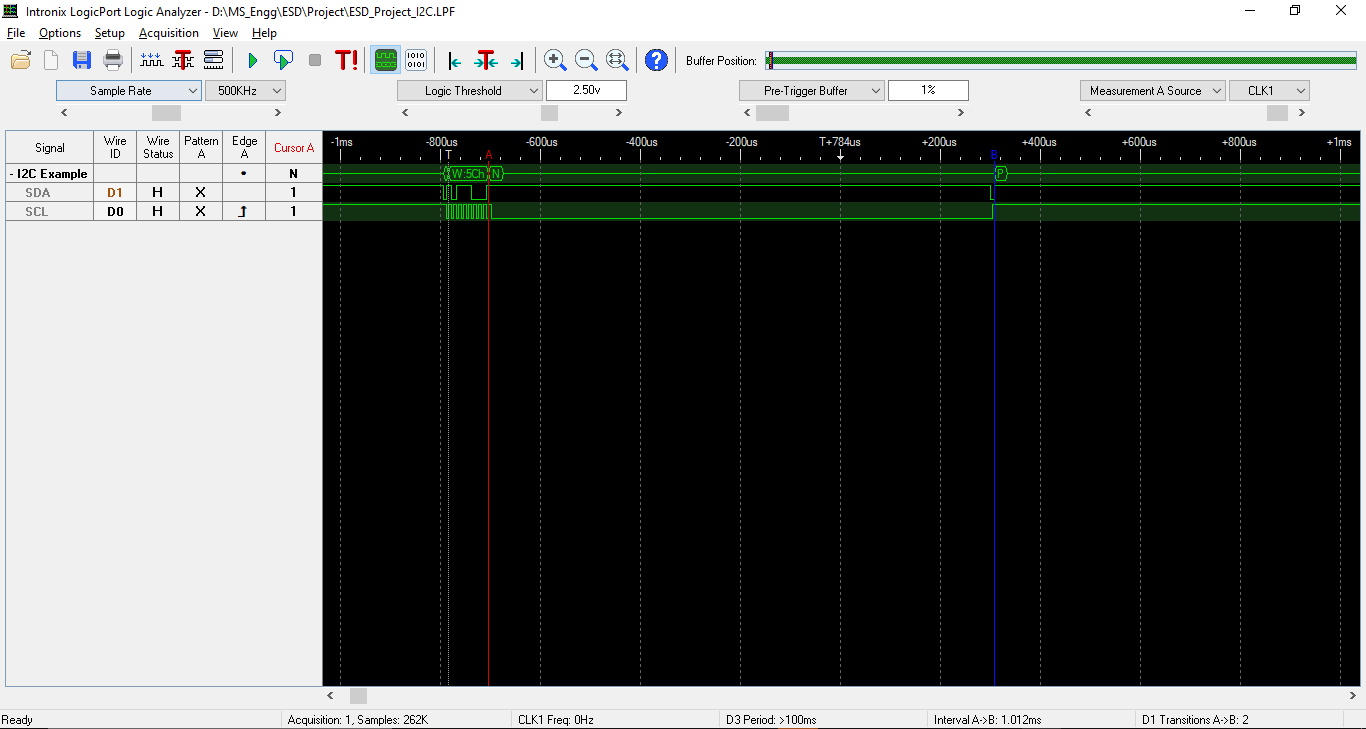


Figure 9 : AM2320 : Wakeup sensor I2C frame SDA not pulled low

To correct the sequence and provide more control, I used the bit-bang method to have precise control over the wake-up frame sequence. With that I got the below frame which matches exactly with the datasheet.

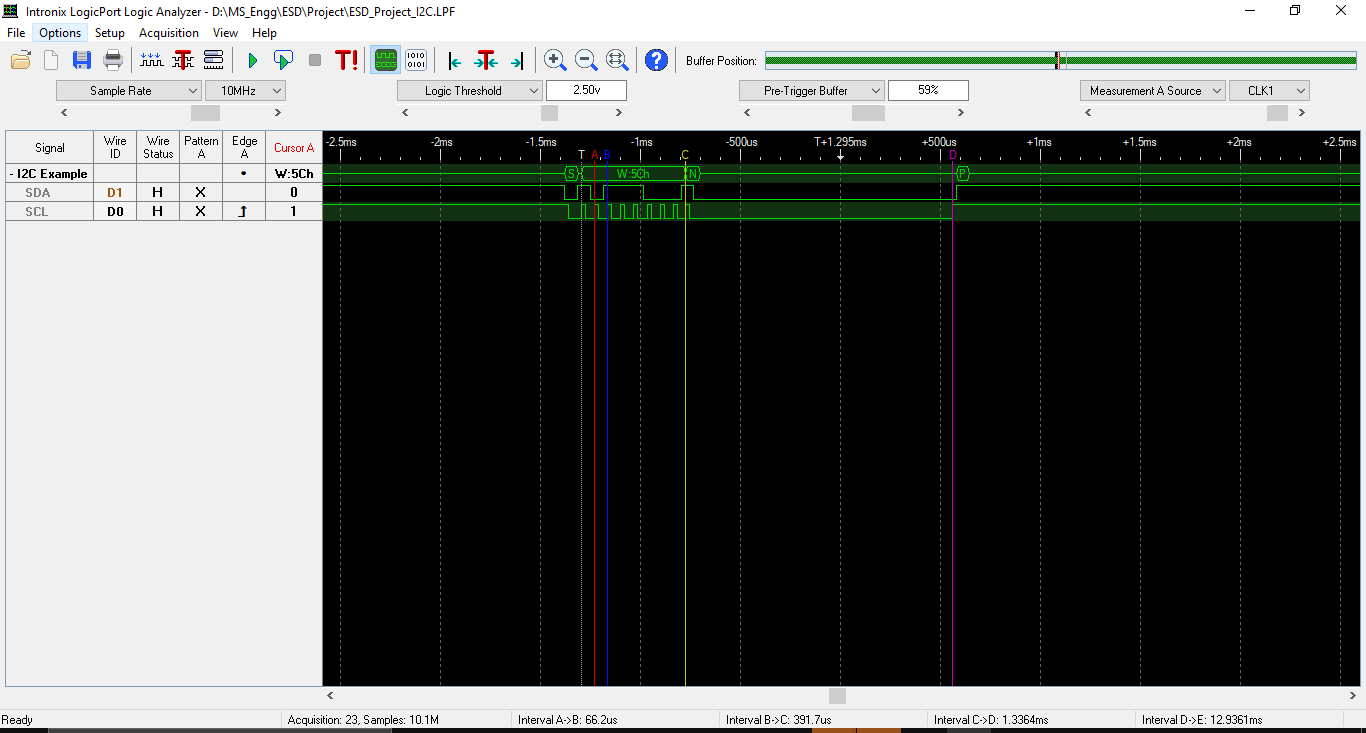


Figure 10 : AM2320 : Wake up sensor corrected I2C frame using bit bang.

After all these corrections and debugging, the frames matched exactly with the Arduino and the datasheet requirements. The AM2320 sensor was functioning properly sometimes.

1. After this the voltage levels were checked at the Vdd, SCL, SDA and the MSP source as well. The connections and voltage were as per requirement in the datasheet. The sensor behavior was still unreliable. Even the I2C operating frequency was reduced.
2. This was all on a breadboard setup. Hence, then decided to move to the perfboard setup as power supply was ready by this time. There was no change in the behavior. The sensor was still unreliable. It would work sometimes.

With less time in hand, moved to the si7021 sensor.

Thus, the implementation code for AM2320 sensor was a combination of bit bang (wakeup sensor) and I2c Register approach (Send read command and read temperature and humidity data) which can be seen in the below diagrams.

Following are the frames which are in sync with the datasheet, which proves that the firmware code for the I2C frames was in sync with datasheet.

1. Wakeup sensor: [3]

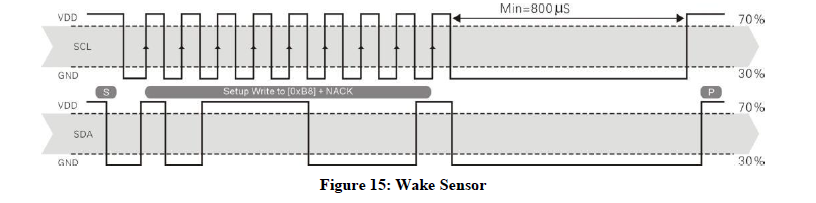


Figure 11 : AM2320: Wakeup sensor (Datasheet frame)

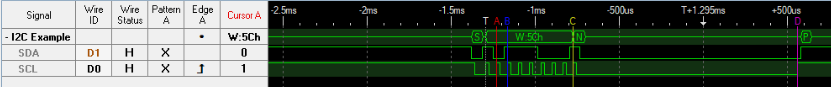


Figure 12 : AM2320 : Wake up sensor (Logic Analyzer trace )

1. Send read command to the sensor: [3]

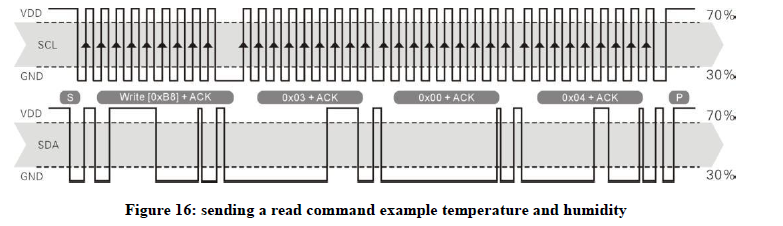


Figure 13 : AM2320 : Send Read command I2C frame (Datasheet frame )

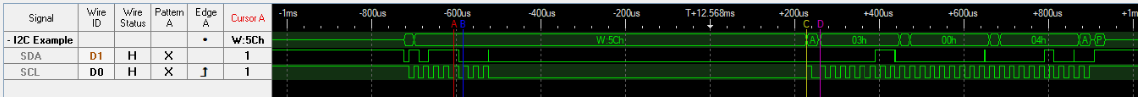


Figure 14 : AM2320 : Send Read command I2C frame (Logic Analyzer trace )

1. Read the temperature and humidity value: [3]

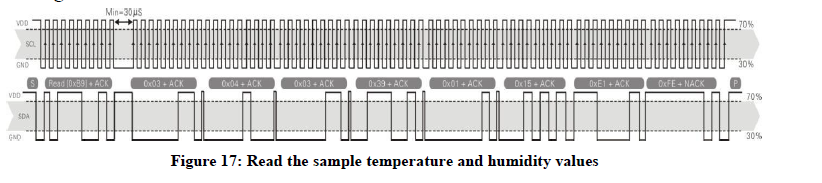


Figure 15 : AM2320 : Read Temperature and Humidity (Datasheet frame )

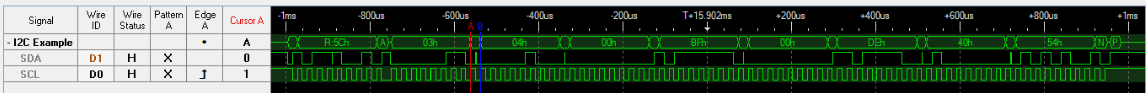


Figure 16 : AM2320 : Read Temperature and Humidity (Logic Analyzer trace )

All the three frames together. First is wakeup frame, second is send read command to the sensor and the third is the read data from the sensor.

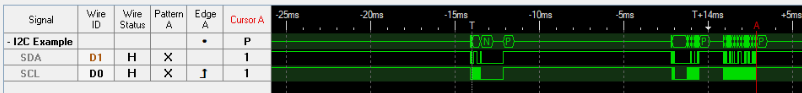


Figure 17 : AM2320 : All the three frames, Wakeup, Send Read command, Read data.

***To summarize, I learnt a lot about how to develop I2C software through this sensor and the software side debugging and also learnt how to snoop and I2C frame through Arduino and analyze it to develop a software in another microcontroller.***

## SI7021 sensor interfacing

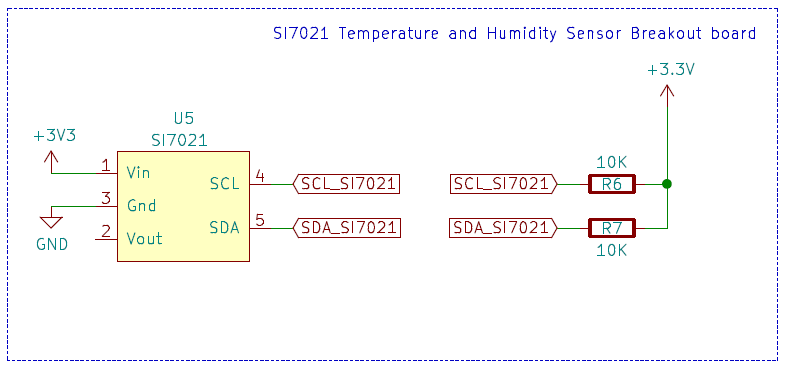


Figure 18 : SI7021 : Temperature and Humidity sensor circuit diagram

The Si7021 is also temperature and Humidity sensor. I used a breakout board in this case as a fallback to AM2320.

Following were the reason for choosing this sensor.[4]

1. Precision Relative Humidity Sensor. ± 3% RH (max), 0–80% RH.
2. High Accuracy Temperature Sensor. ±0.4 °C (max), –10 to 85 °C.
3. I2C Interface with 7-bit slave address of 0x40
4. Low Power Consumption.
5. The datasheet had clearly mentioned the required operating voltage and pulll-up resistor values.
6. Operates at 3.3 V
7. Direct formula from datasheet to calculate temperture and humidity.
8. It also has heater mode to test the sensor

With the knowledge gained in I2C software from working on AM2320 sensor, the code development for SI7021 was very fats and I got it working. I had analysed the frames from arduino code before I started working on the software. I referred the datasheet for address and other parameters.

The following pins were chosen from MSP for communication over I2C. The different sets of pins were chosne because the pull-up resistor values were different also the AM2320 was implemented using bit bang and I2C register combination also had a n unreliable working. This might create complexity during firmware development for SI7021.

### Technical Details ( firmware design perspective )

Following are the pin connections to the MSP432P401R apart from Vcc and ground:

Clock frequency : 100KHz

P6.4 UCB1SDA SDA of SI7021

P6.5 UCB1SCL SCL of SI7021

I2C\_SI7021\_ADDRESS 0x40

SI7021\_MEAS\_RH\_NOHOLD\_CMD 0xF5

SI7021\_MEAS\_TEMP\_NOHOLD\_CMD 0xF3

SI7021\_BUFF\_SIZE (8)

SI7021\_NOOFBYTES\_WRITE (1)

SI7021\_NOOFBYTES\_READ (2)

### Challenges & Learnings from SI7021

1. Once the basic software was ready – Start bit, nack, stop bit etc. The problem was with the reading of the number of bytes. It was not clearly explained from datasheet, how many bytes are to be read and which are the concerned bytes. To solve this issue, I analysed the arduino frames [b] and found the number of bytes to be read exactly. Modified the number of bytes required and thus calculated the values.
2. Direct example from the current project, when AM2320 sensor wasn’t behaving reliably, it took me only 15 to 20 minutes to reconfigure another I2C port to change pin assignment and initialization. My development time was saved.

The pull-up resistor requirement for both the sensors is different. Hence, was required to use another set of I2C port pins and also to reduce complexity during development.

Following are the I2C frames for SI7021:

1. First frame is the read request for humidity, Read humidity data from the sensor (post 20 msec), read request for temperature (post 2 msec) and then read temperature data from the sensor. It was not possible to capture the whole sequence in analyser ina legible manner.

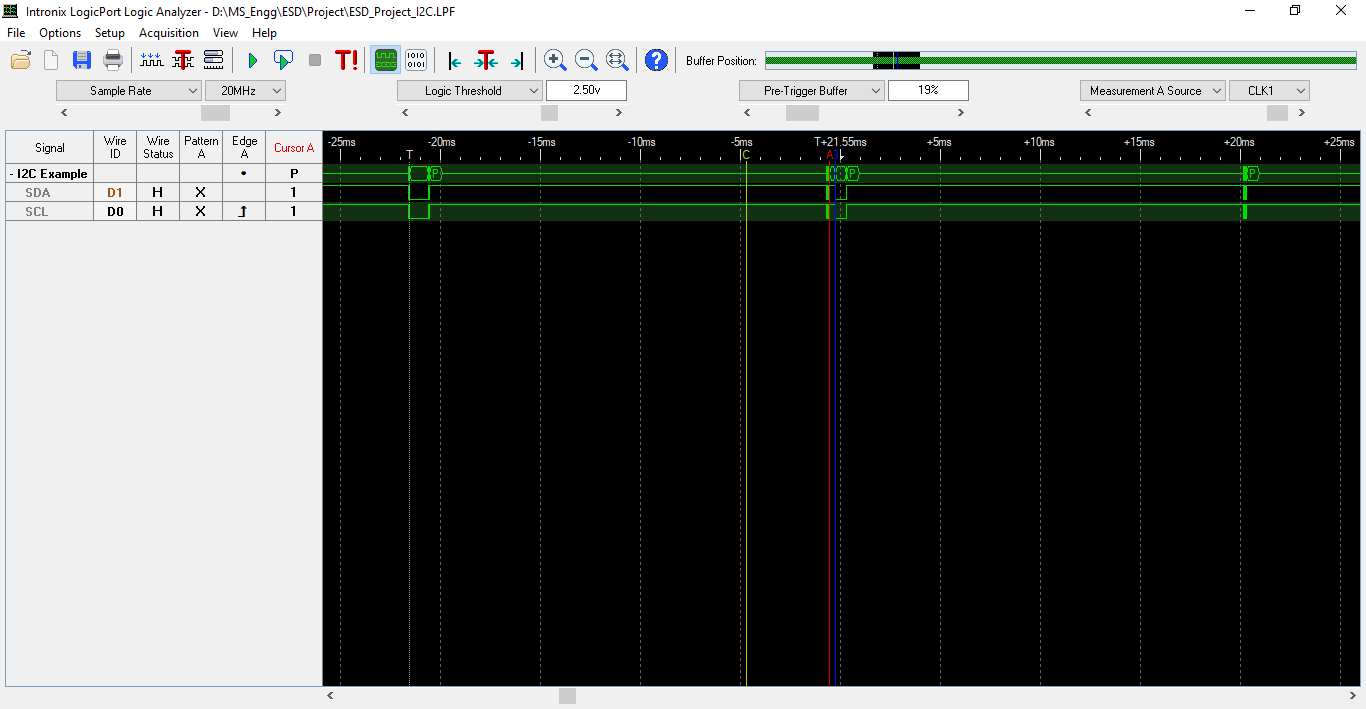


Figure 19 : SI7021 : All 4 frames Humidity Write, Read Request & Temperature Write, Read Request

1. Request to read Humidity frame:

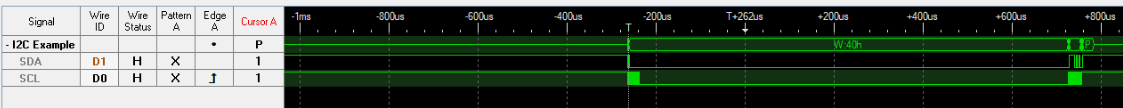
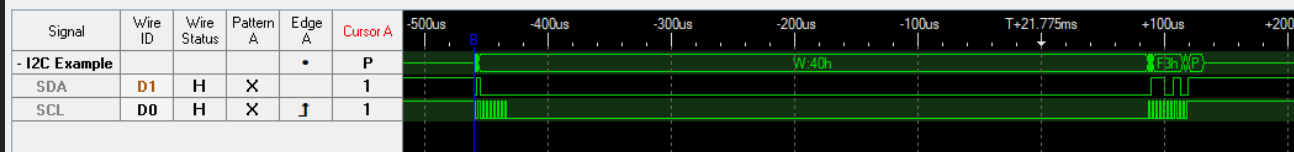


Figure 20 : SI7021 : Read request for humidity data from the sensor (Logic Analyzer trace )

Same frame zoomed in a little so that it is legible:



1. Reading Humidity data from the sensor:

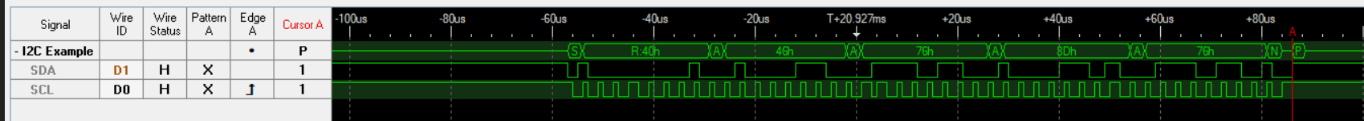


Figure 21 : SI7021 : Reading of Humidity data from the sensor (Logic Analyzer trace )

1. Request to read Temperature frame:



Figure 22 : SI7021 : Read request for temperature data from the sensor (Logic Analyzer trace )

1. Reading Temperature data from the sensor:

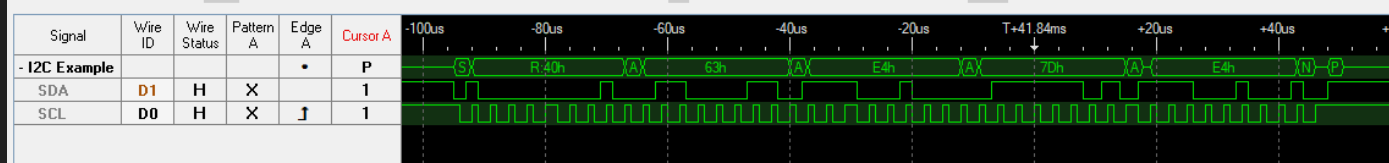


Figure 23 : SI7021 : Reading of temperature data from the sensor (Logic Analyzer trace )

***To summarize, I learnt how to port a software developed for one sensor to another. I had used generic macros in the development of the software for AM2320 code. I made use of these macros in the SI7021 code development and generic read, write functions. This helped me to achieve a faster development time.***

## Nokia 5110 LCD interfacing

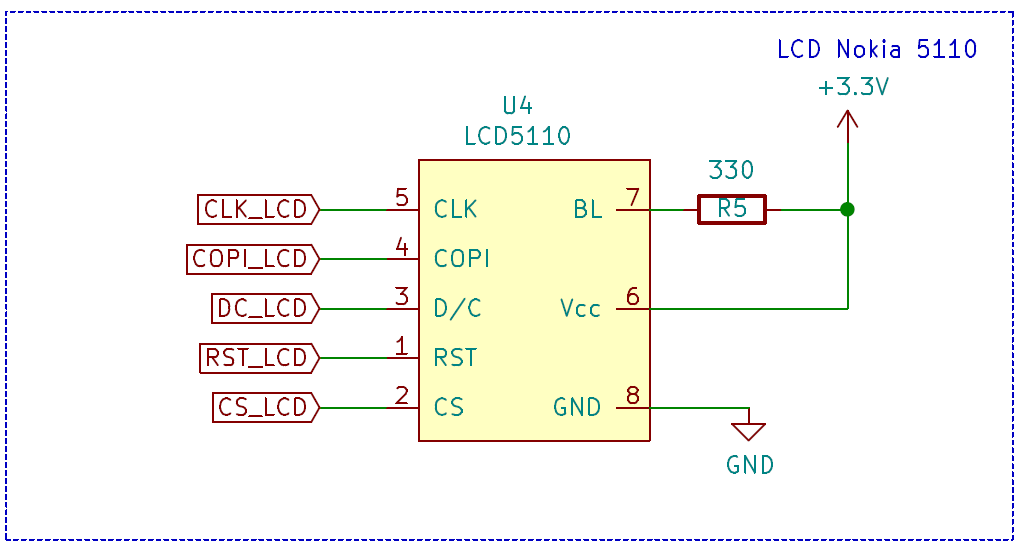


Figure 24 : Nokia 5110 : LCD circuit diagram

Following figure 25, is the internal block diagram of the PCD8544 [5]

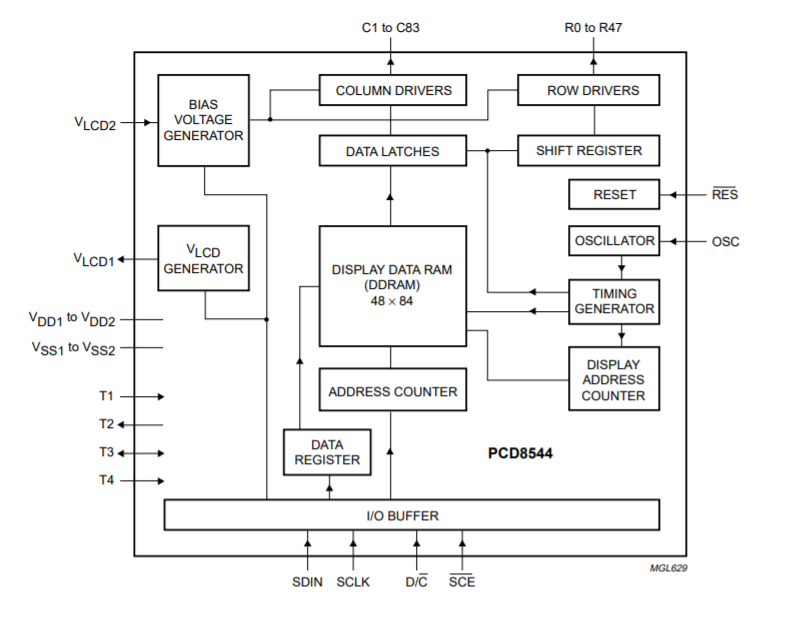


Figure 25 : Nokia 5110 : PCD8544 internal architecture block diagram

The PCD8544 is a low power CMOS LCD controller/driver, designed to drive a graphic display of 48 rows and 84 columns. All necessary functions for the display are provided in a single chip, including on-chip generation of LCD supply and bias voltages, resulting in a minimum of external components and low power consumption. The PCD8544 interfaces to microcontrollers through a serial bus interface. The PCD8544 is manufactured in n-well CMOS technology. [5]

Following are the features of the display.

1. Display data: 48 x 64
2. Supply voltage 2.7 v to 3.3v
3. Serial interface maximum 4.0 Mbits/s
4. External RES (reset) input pin
5. Low power consumption, suitable for battery operated systems

### Technical Details ( firmware design perspective )

Following are the pin connections to the MSP432P401R apart from Vcc and ground:

Clock frequency : 100Khz

P2.4 GPIO pin Data/Command of LCD

P2.6 GPIO pin Chip Select of LCD

P2.7 GPIO pin Reset of LCD

P3.5 UCB2CLK Clock of LCD

P3.6 UCB2SIMO Data in of LCD

The Back light of the LCD is connected to Vcc through a resistor of 330 Ohm for current limiting protection.

### Challenges & Learnings from Nokia 5110:

1. The Chip select GPIO line was not going low in early. This led to a corruption of data and the LCD was not initialized properly. An 8-microsecond delay added for the same to correct.

MOSI

SCLK

CS

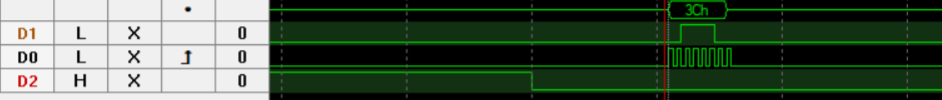


Figure 26 : Nokia 5110 : SPI Transaction

1. The data direction register configuration for the GPIO lines was missed during initialization. This was corrected by adding the appropriate code.
2. The Reset pulse was not generated properly because of which the LCD was not initialized properly during startup. The reset sequence was introduced in the code as per the below timing diagram from datasheet [5]

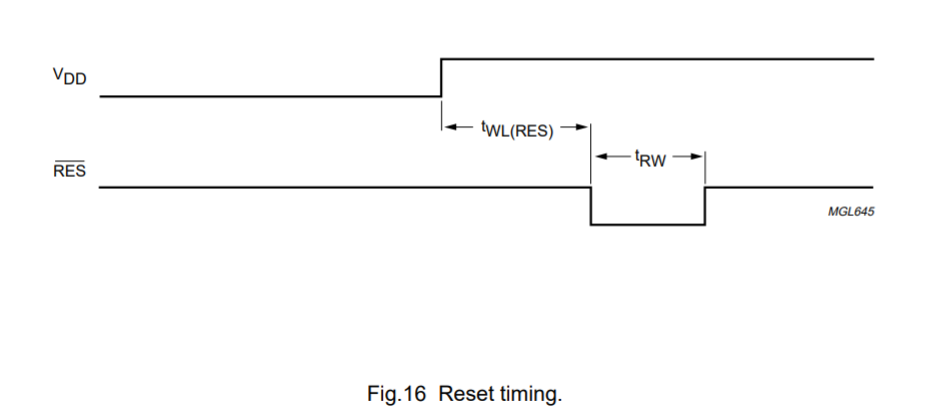


Figure 27 : Nokia 5110 : Reset pulse sequence for LCD during initialization.

1. The Nokia 5110 received was damaged. I verified by running a sample arduino code.[d] Had I worked on this device without testing would have led to a lot of time wastage.

The Nokia 5110 LCD received from senior could not be desoldered also there was no space to wirewrap the pins very fast with the time available in hand. Hence, had to use the below arrangement for connecting to the MSP432P401R by soldering the male jumpers to the LCD.

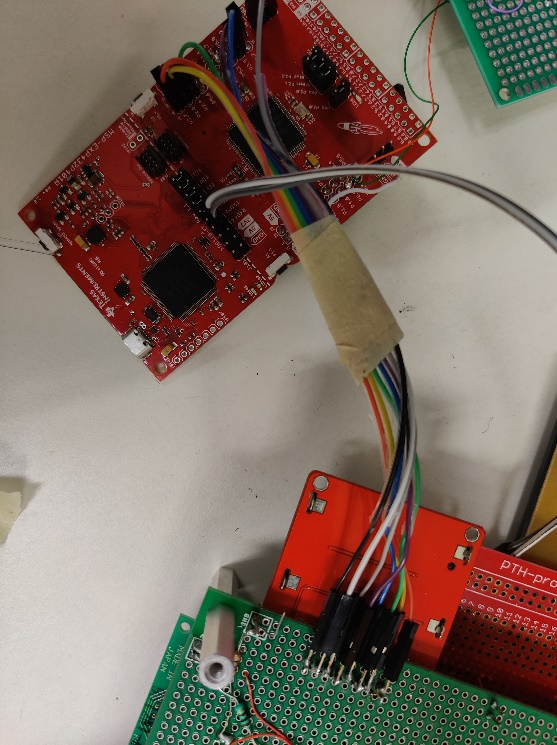


Figure 28 : Nokia 5110 : LCD soldered the mail jumper side to the module.

***To summarize, I learnt how to always recheck your code. A code ported from online reference may not be correct always. All the minute details of initialization need to be considered (GPIO configuration). It is always better to order a device which is Arduino compatible as it will save time because no time is wasted on a damaged component arrived. Lesson learnt is to always check the parts the day they arrive. I was lucky to have found a Nokia 5110 to borrow from a senior o complete the display element. However, due to lack of time could not explore the graphical section.***

## UART-0 interfacing

I wanted to transfer the data to my laptop for storing and displaying the data in a graphical manner so that it is easy to understand to a person.

I wanted this communication to be stable and easily scalable. Hence, I decided to add a circular buffer queue for both the transmitter and receiver buffer along with the UART-0 in interrupt mode.

In designing the circular buffer queue, I referred Making Embedded Systems book[e]

The queue size of each buffer is 256 bytes.

The queue size is easily configurable.

The FTDI chip handles logic level translation from CMOS to TTL logic levels. Thus, it then becomes suitable to be connected to Host PCs comport.

### Technical Details ( firmware design perspective )

Following are the pin connections to the MSP432P401R apart from Vcc and ground:

Baud rate: 9600

P1.2 UCA0RXD Tx of FTDI converter

P1.3 UCA0TXD Rx of FTDI converter

Common ground of MSP and FTDI converter

### Challenges & Learnings from Interrupt based Circular buffer implementation.

1. The transmitter interrupt cannot be enabled continuously. Doing so, caused the tx interrupt to trigger continuously as the TXBUF is empty after the data is transmitted. Hence, the system was modified to enable the Tx interrupt only when the Tx queue is loaded with a data and is not empty.

As soon as the Tx buffer becomes empty the tx interrupt is disabled. This also optimizes the usage of the system resources.

1. The documentation of the FTDI converter did not mention exactly which pinouts were to be used. There were multiple TX and RX pins present. Through continuity check, I came to know that all the Tx and Rx pins are shorted. Hence, for current purpose any RX and TX pin connection would do as the FTDI chip will be powered through laptop.

***To summarize, I learnt to work based upon intuition and logical skills when the documentation is poor.***

## Firmware Design

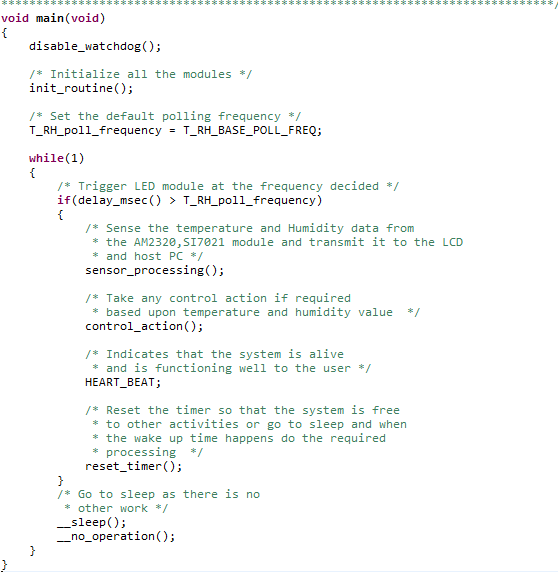


Figure 29 : Code snippet : Code architecture/Software architecture overview

Please refer the features mentioned in the next heading and map with the figure 29.

### Firmware Design Features

The code was architected keeping following features in mind:

1. Modular
2. Readability
3. Sequential in terms of process the data (temperature, humidity) and display it, take control action if any, go to sleep for the time requested and then redo the same process when there is time to wake up.
4. Data security by trying to provide only the required information in the header files i.e., there are only few instances where Macros are defined in the header file. Thus, providing abstraction.
5. The code is also ensured to be easily reconfigurable by providing custom macros due to which it is not required for one to change the code at each and every line to maximum extent possible. Due to time constraints, every pin may not have been assigned a macro, but efforts were taken to provide the same to maximum extent possible. In case of MSP pin damage for I2C communication, one can shift to another set of alternative pins available by changing the pin assignment.

Direct example from the current project, when AM2320 sensor wasn’t behaving reliably, it took me only 15 to 20 minutes to reconfigure another I2C port to change pin assignment and initialization. My development time was saved.

The pull-up resistor requirement for both the sensors is different. Hence, was required to use another set of port pin.

### Module Description

1. There is a separate init\_routine(); module to simplify the design and enhance readability as observed in the below figure 30.

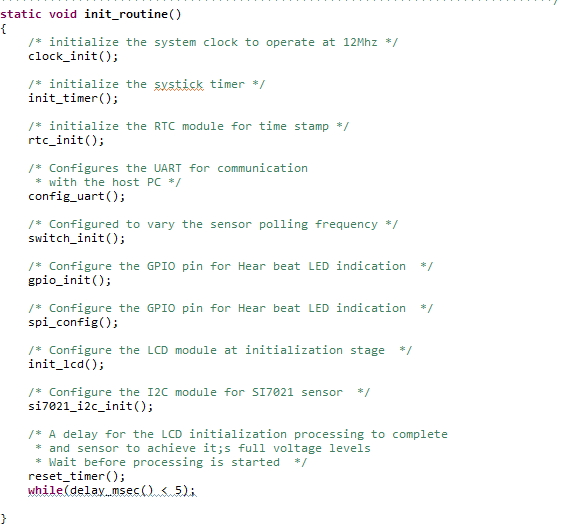


Figure 30 : Code snippet : Initialization routine module

1. From Figure 29, Sensor read rate can be varied from 1 second (default) to 10 second. It depends on the T\_RH\_poll\_frequency. This is for power saving. A systick timer concept is implemented which ensures there is no blocking and only at the polling interval will trigger the read request to the sensor. Rest of the time, the MSP432P401R will continue to sleep.

The below code snippet in figure 31, shows how onboard switches are used to achieve the same.

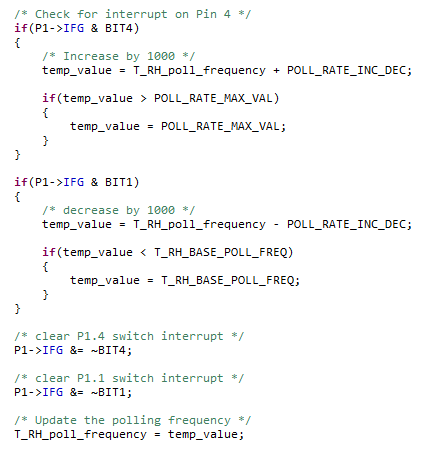


Figure 31 : Code snippet : Increase/Decrease polling interval.

1. From figure 29, the HEART\_BEAT; LED toggles every time it enters the condition. This gives a visual indication to the user that the system is alive and functioning well.
2. From figure 29, the sensor\_processing(); module internally does the write and read request for Temperature and humidity data either through AM2320 or SI7021. Currently, due to AM2320’s unreliable behavior only SI7021 is functioning.
3. The si7021\_temperature\_humidity\_measurement(), is the entry point for temperature and humidity measurement carried for SI7021 sensor.

As it can be seen in the figure 32 , the sequence is request for humidity data, wait for 20 msec, Read humidity data, wait for 300 usec roughly and do the same process for temperature cycle



Figure 32 : Code snippet : SI7021 temperature and humidity measurement

1. The am2320\_temparutre\_humidity\_measurement(); is the entry point for temperature and humidity measurement carried for AM2320 sensor.

As it can be seen in the figure 33 , the sequence is sent wake up sequence, request for temperature and humidity data, wait for 2 msec and then read the temperature and humidity data.

Wait for 10 msec and repeat the whole cycle.

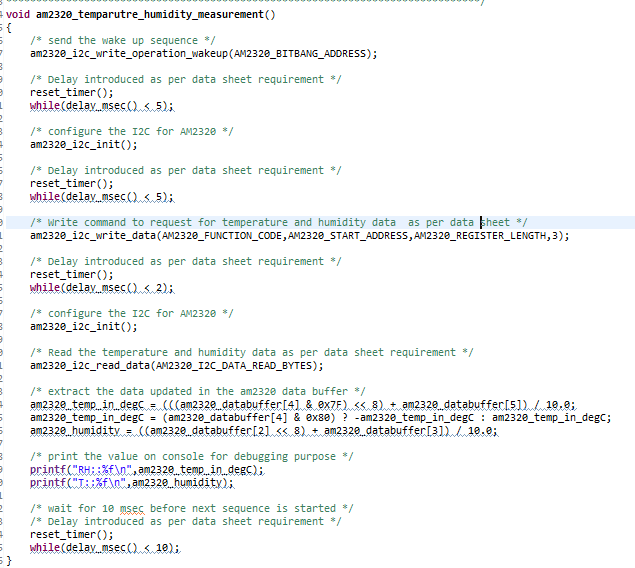


Figure 33 : Code snippet : AM2320 Temperature and Humidity Sensor

1. From figure 29, the control\_action(); module internally takes the control action based upon the variation in temperature values. Currently, as per the climate in the lab, the system is set to turn BLUE Led on when it crosses a TEMP\_THRESHOLD value of 23°C for testing purpose. The purpose of lighting BLUE Led is to demonstrate that It could have been a fan in a cooling plant, or alarm could be fired when a threshold value is crossed.

### Challenges & Learnings from Firmware development

The challenges and learnings during firmware development are mentioned in the section 3.3 to 3.6 individually for better segregation and understanding.

Please refer subsection 3.X.2 of all the modules 3.3 to 3.6

## Software Design

The software component of the project was the python code written. It was my first attempt in using python language. I referred a large number of websites to get it working.

### Challenges & Learnings from Python

1. I had not realized earlier that there is a difference between pip3 and pip command. I was using pip command (a ver 2.X specific command) on a ver 3.x python software installed in my system. I did not receive a proper indication from the python interpreter that it was wrong pip command used. I had left working on this part and moved to the firmware design.
2. Much of the reference code available was for python version 2.x. I learnt to port the code to the version 3.x. I had to use .decode("utf-8"), to solve any encoding issues that may occur. This was not clear to me before.
3. I learnt the concept of delimiter (, and :: ) to distinguish and separate a given string data into separate array components.
4. I learnt the matplotlib library of python to display a simple graph.
5. For development of the graph and delimiter logic, I used code from Lab 3 part 3 on board temperature sensor code. This helped me to look for alternatives to proceed with the development of a particular module.

Initially, I though python would be easy to use and need to only run a set of commands and this part of the project would be over within a few hours to focus on other elements of the project. But I was wrong. There were issues which took time to resolve, mainly because I was not familiar with the language. I had taken up python only after establishing the I2C communication with AM2320 sensor so that I can work effectively for plotting data but the unreliability if the sensor, pushed this activity behind.

***I am glad to have chosen this component. A step towards learning a new language highly relevant in the industry. This is also a good alternative to SD card during prototype development. When the concept has been tested, a local SD card firmware design can be incorporated.***

## Testing Process

Following is the testing process carried out.

1. Power Supply

The Voltage was measured using a multimeter and the noise was measured using the Oscilloscope. The measure feature of the oscilloscope was used to find the maximum and minimum voltage levels detected over a period of time. When measuring the values even the stray signals were captured. Please refer the images below.

1. 5V verification:

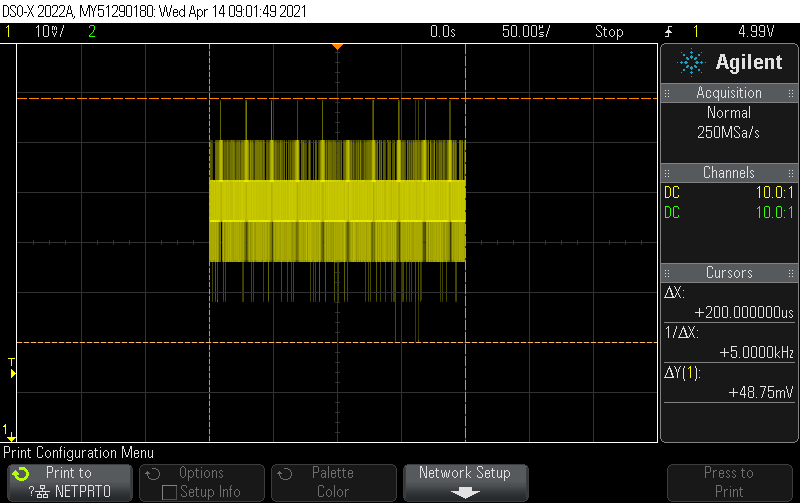


Figure 34 : Output : 48.75 mv noise at +5V

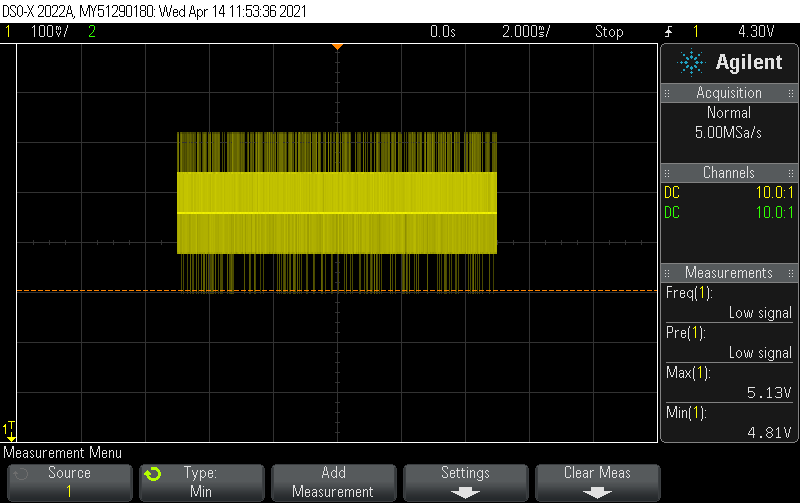


Figure 35 : Output : 4.8V to 5.1V variation

1. 3.3V verification:



Figure 36 : Output : 403.75mv noise at +3.3V

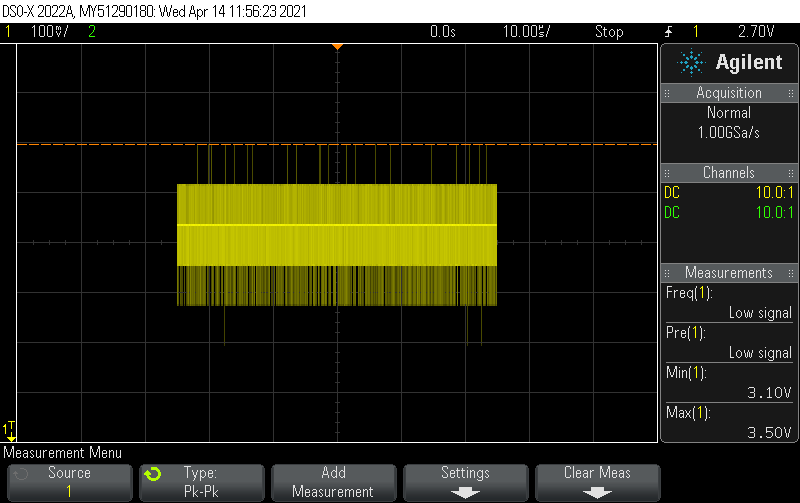


Figure 37 : Output : 3.1v to 3.5v variation

1. Temperature and Humidity Sensor

Below is the image of the TRH value before the lighter was lit and air was blown near the sensor.

Please refer the video from 02:00 min to 02:30 sec interval for this functionality.

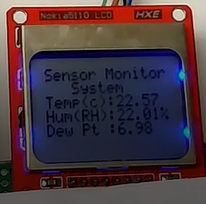


Figure 38 : Output : LCD showing temperature and Humidity values before testing

1. Temperature Value:

A lighter was lit beside the temperature and humidity sensor to check if there is an increase in the value of the temperature. Please refer the video to see the same demonstration.

Video Timestamp: 02 min:00 Sec

After the lighter was lit near the SI7021 sensor.

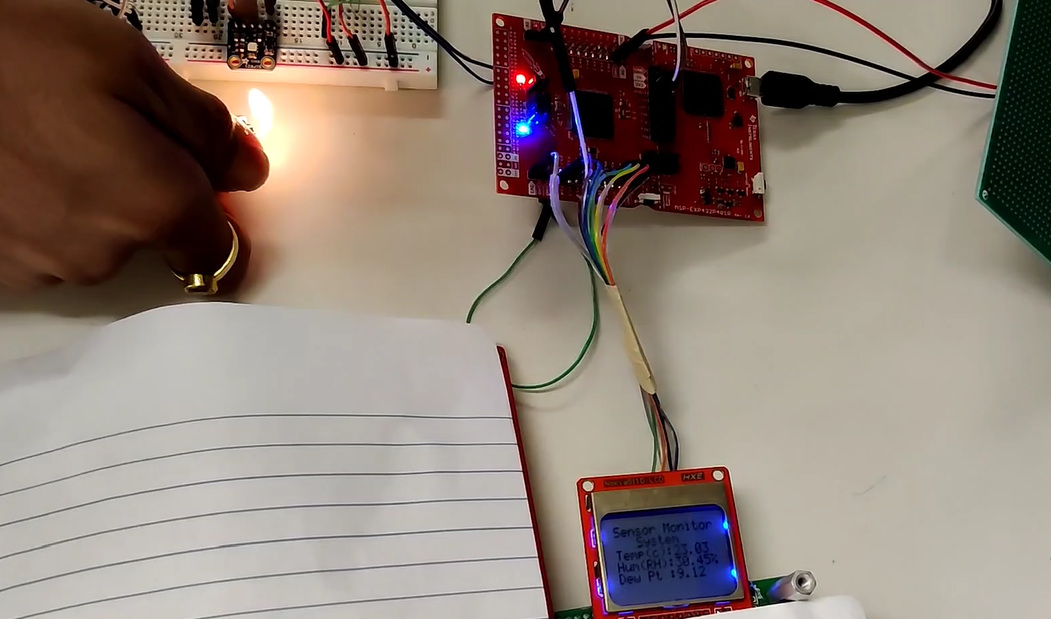


Figure 39 : Output : Value increased due to lighting up of lighter near the sensor

1. Humidity Value:

Air was blown from the mouth on the sensor to check if the humidity would increase. There was definite increase in the humidity value. Please refer the video to see the demonstration.

Video Timestamp: 02 min: 22 sec

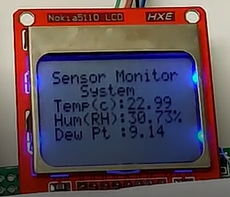


Figure 40 : Output : Increase in Humidity value when air is blown near sensor

1. LCD Nokia 5110

A visual check was done to check if the data written to the LCD was visible at the location requested.

1. Python Graph

The graph was checked to see if it showed an increase (lighter was lit and air was blown near the sensor.) and decrease (after some time) in the temperature and humidity values. The behavior was same as per the expectation.

Please refer the below image below:

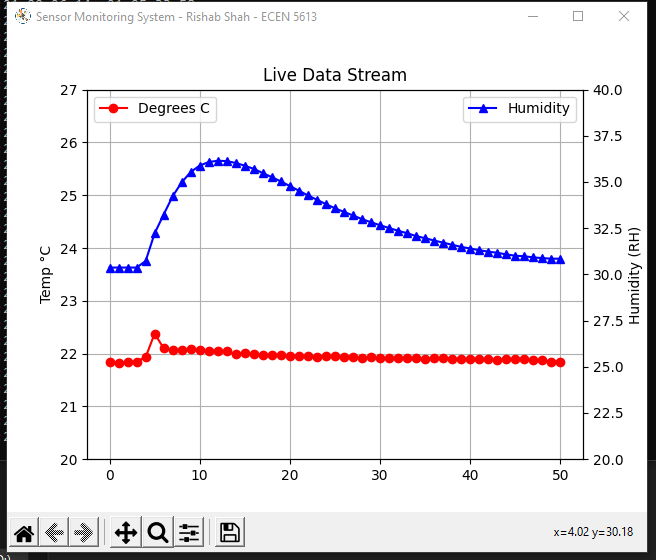


Figure 41 : Output : Live plotting of temperature and humidity data - 1

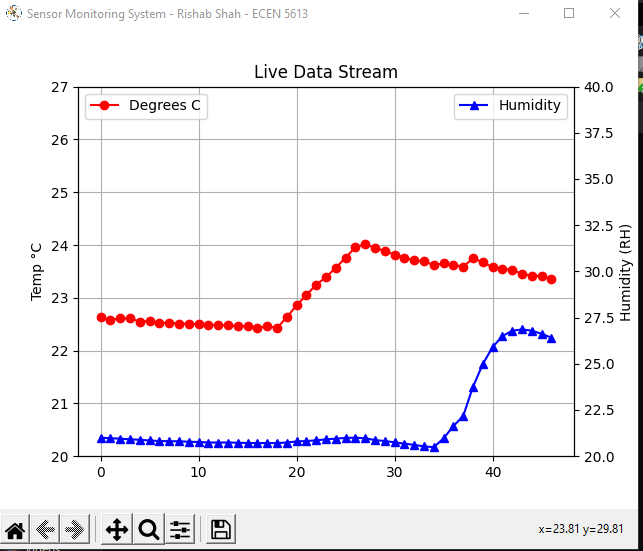


Figure 42 : Output : Live plotting of temperature and humidity data - 2

1. RTC timestamp:

I had used a timer - 0 in my code to generate a tick of 1 msec. Whenever the tick count reached 1000, I printed the RTC timestamp value. I had also arranged to change the printing frequency through the use of the onboard switches.

When P1.4 was pressed the value increased by 1000 and reached a max value of 10 secs and when P1.1 was pressed the value decreased by 1 second and reached a minimum value of 1 second. This was used to change the print frequency.

Whenever the RTC timestamp value was printed at the set frequency, the difference between the previous value and the current value was in sync. i.e., if it was set to 5 second printing frequency, the difference obtained between two consecutive prints was 5 second.

1. Data Storage:

The timestamp value, Temperature value and humidity value was sent over UART to the PC. This data was printed over the terminal. This data was also simultaneously stored in the log file. After the whole temperature and humidity testing process as carried out in the step B, the log value obtained was compared with the printed value. Both the values were same.

Once observe that the data is varying in the log. The logging is done at a read frequency of 2 seconds.

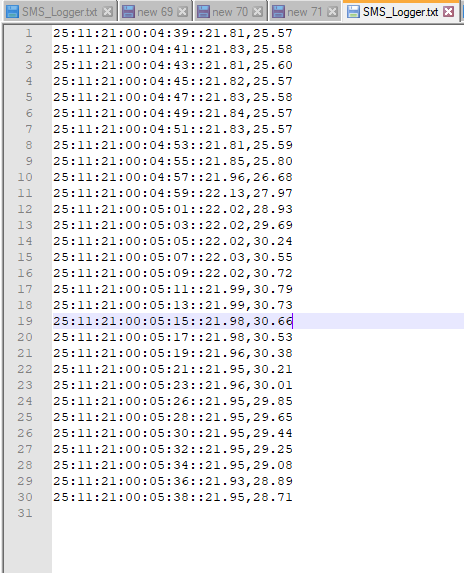


Figure 43 : Output : Storage functionality displayed using python

# Results and Error Analysis

I have covered the challenges associated with each element in the section 2. Here, I am highlighting the major challenges faced in a brief manner for each hardware and software element.

**Hardware:**

Power-supply and component selection: From my background, I had some experience in the software industry. It was only after I started with my master’s and in this coursework, I got exposure to the hardware designing principles and firmware debugging from hardware issue perspective. I had never browsed Digikey, mouser etc. before to the selection of components for that reason I would consider as one of the most difficult aspects. The selection of Buck converter along with LDO for achieving an efficient voltage conversion. I have made mistakes in selection of components. The selection of schottky diode was wrong from the perspective because I had missed to check the thickness of the lead diameter. It was through this process, I understood various design parameters that should be considered while selection of a component, which I would say is a major learning to take away from a hardware course.

Next time, I would know what to look for while selecting the components.

**Software:**

I2C Communication: Previously, in the labs, I had used bit-banging approach to r/w to the EEPROM 24LC16B. I had thought that the register approach may be easier than the bit-banging but that was not the case. The AM2320 sensor gave me insight into various methods of debugging – Hardware – connections, Single mode feature of oscilloscope, Logic analyzer, snooping Arduino code frames through logic analyzer and correct my code timings to make it work, using a combination of bit bang and I2C registers in the same code base to produce the desired I2C frame sequence.

I have learned the different debugging techniques to use during a development.

# Conclusion

In the Sensor Monitoring System project, I had introduced each and every element to achieve a particular functionality and to proceed the development in a step-by-step manner.

The power supply circuit designing was chosen to add a new hardware element in my project and to learn how to select a component and design it to achieve the same.

I had chosen the AM2320 sensor, an I2C based sensor to learn how to code using a register-based approach.

When AM2320 did not run reliably, I moved to SI7021 sensor (a breakout board) arrangement to achieve the same. Thus, learnt when to move to the fallback.

I had chosen Interrupt based UART + Circular Buffer FIFO based implementation element to transmit the data over serial communication to a remote PC for debugging purpose. Using the console printf feature would add latency and will corrupt with the sensor behavior. This becomes especially important when the whole system is integrated, and multiple functionalities are running.

A slight latency would disturb the working and may lead me in a wrong direction while debugging.

The SD card element was chosen as a stretched activity and the aim was to provide a local storage of data and to learn SPI communication. Owing to time-constraints, could not implement this functionality but ensured that the concept of data logging and its significance was highlighted by storing the data in the local PC.

Python is an important language which is currently used in the industry for faster development time and to carry out any data analysis. It is already witnessed that the ML concepts are being used in the embedded software for better result. This was a step taken by me to start learning python so as to be industry ready.

I would like to conclude that I have achieved the objectives and purpose of choosing each and every element and the associated learning that came along with it by implementing either the proposed elements in the PDR or through fallback plan.

# Future Development Ideas

The work on project should be done in a parallel manner. Working on AM2320 sensor to get it working reliably ate up time to work on the LCD and SD card part of the project.

I want to add a BLE module/ Wi-Fi module in the existing project to transmit all the data remotely instead of the current FTDI converter arrangement to transmit the data to the host PC.

I would want to move the existing system to a remote version by using a battery as the DC source and demonstrate the low power mode feature of the device. I would also like to add a method to sense the battery remaining and transmit it to the remote station.

I would also want to use an array of sensors to generate the average temperature and humidity value to provide more accuracy of the values.

Every sensor has information regarding it’s make, model number and various other information. I would want to update the code to transmit such information over remote location to help in the analysis of the sensor’s health status for logging purpose. This can be done by implementation of a state machine and a command processor (similar to bash terminal).

Even before the work on the project is started, my first step is to acquire an Arduino and test the device requested from the manufacturer to ensure that a proper working sensor/device is received.

An automated test-case has to be added which will test if all the pixels of the LCD are working properly. This would increase the reliability of the LCD.

Once the whole system is set up and works properly on a breadboard setup, I will design it on a PCB as soldering is not that convenient on a perfboard.

Connecting a battery source will also help in enhancing the testing capability as the device can be moved around freely to gather the variation of temperature and humidity at various places as it will give insight into how sharp/flat the curve remains when it is tested in the real environment.

# Acknowledgements

I would like to thank professor for designing the labs in a step-by-step incremental manner with difficulty rising at each level. This helped us to constantly raise the bar and instill in me confidence to build a product by my own.

I would like to thank Prayag Desai and Sundar Krishnakumar for their support in guiding me on various stages to complete the project. There were many aspects which were never explored before by me. Their guidance on how to approach each of the elements helped me to work effectively.

I would like to thank Prayag Desai for lending me his SI7021 Temperature and Humidity sensor module to complete the sensor data reading element of the project. I would also like to thank Harsh Rathore for lending me his Nokia 5110 LCD module to complete the display element of the project.

I would like to thank Dhruv Mehta for lending his Arduino Uno board. This helped me to validate the sensors and LCD received and hence reduce my debugging time on a damaged component.

# References

Datasheets:

[1]LM2576T:<https://www.ti.com/lit/ds/symlink/lm2576.pdf?ts=1619692029414&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FLM2576>

[2] L4931: <https://www.st.com/resource/en/datasheet/l4931.pdf>

[3] AM2320: <https://cdn-shop.adafruit.com/product-files/3721/AM2320.pdf>

[4] SI7021 : <https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf>

[5] Nokia 5110 : <https://www.sparkfun.com/datasheets/LCD/Monochrome/Nokia5110.pdf>

[6] Schottky diode: <https://www.onsemi.com/pdf/datasheet/sb580-d.pdf>

[7] Inductor: <https://www.we-online.de/katalog/datasheet/7447221221.pdf>

[8] MSP432P401R Technical Reference manual: <https://www.ti.com/lit/pdf/slau356>

[9] MSP432P401R Pinout diagram datasheet: Provided in the development kit.

[10] FTDI converter: <https://media.digikey.com/pdf/Data%20Sheets/Sparkfun%20PDFs/BOB-13263_Web.pdf>

Firmware - Code references:

[a]: <https://github.com/adafruit/Adafruit_AM2320/blob/master/Adafruit_AM2320.cpp>

[b]: <https://github.com/adafruit/Adafruit_Si7021>

[c]: <http://rohitg.in/2014/11/09/Nokia5510-lcd-with-MSP430/>

[d]: <https://github.com/adafruit/Adafruit-PCD8544-Nokia-5110-LCD-library>

[e]: <https://www.oreilly.com/library/view/making-embedded-systems/9781449308889/>

Software - Python references:

[f]: <https://pypi.python.org/pypi/pyserial>

[g]: <http://embeddedlaboratory.blogspot.com/2016/06/serial-communication-using-python.html>

[h]: <https://makersportal.com/blog/2018/2/25/python-datalogger-reading-the-serial-output-from-arduino-to-analyze-data-using-pyserial>

[i]: <https://www.w3schools.com/python/python_file_handling.asp>

Other Web sources:

Dew Point calculation: <https://iridl.ldeo.columbia.edu/dochelp/QA/Basic/dewpoint.html>

# Appendices

Several appendices have been attached to this report in the order shown below.

## Appendix - Bill of Materials

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| The BOM is as per the components that were used in the project.   |  |  |  |  | | --- | --- | --- | --- | |  | **Part Description** | **Source** | **Cost($)** | | A) | Power Supply Design |  |  | | 1 | [LM2576T-5.0/NOPB-ND](https://www.digikey.com/product-detail/en/texas-instruments/LM2576T-5.0-NOPB/LM2576T-5.0-NOPB-ND/212636) | [www.digikey.com](http://www.digikey.com/) | 2.94 | | 2 | [732-9643-ND](https://www.digikey.com/product-detail/en/w-rth-elektronik/7447221221/732-9643-ND/5870450) | [www.digikey.com](http://www.digikey.com/) | 1.79 | | 3 | [SB540FSCT-ND](https://www.digikey.com/product-detail/en/on-semiconductor/SB540/SB540FSCT-ND/3042709) | [www.digikey.com](http://www.digikey.com/) | 0.53 | | 4 | L4931 | [www.adafruit.com](http://www.adafruit.com/) | 0.95 | | 5 | Red LED | Embedded System Lab | 0.1 | | 6 | Green LED | Embedded System Lab | 0.1 | | 7 | Adapter 9V DC | Embedded System Lab | 0 | | 8 | Power Jack | Embedded System Lab | 1.5 | | 9 | Toggle switch | Embedded System Lab | 1.5 | | B) | Microcontroller |  |  | |  | MSP-EXP432P401R | Embedded System Lab | 20 | | C) | Sensors |  |  | |  | AM2320- 1528-2564-ND | [www.digikey.com](http://www.digikey.com/) | 3.95 | | D) | Display facilitators |  |  | |  | FTDI to USB converter | [www.sparkfun.com](http://www.sparkfun.com/) | 12.95 | |  | Nokia 5110 Sparkfun | [www.sparkfun.com](http://www.sparkfun.com/) | 9.95 | | E) | Borrowed Items/ Donated |  |  | |  | SI7021 | Prayag Desai | 0 | |  | Arduino Uno board | Dhruv Mehta | 0 | |  | Perfboard | Chaitra Suresh | 0 | |  | Breadboard | Chaitra Suresh | 0 | |  | Jumper wires (M-F, F-F, M-M) | Mehul Patel | 0 | |  | Nokia 5110 LCD | Harsh Rathore | 0 | | F) | Resistor Set | [www.amazon.com](http://www.amazon.com/) | 5.57 | |  | Capacitor Set | [www.amazon.com](http://www.amazon.com/) | 5 | | Total: |  |  | 66.83 | | Note: | Not accounted for delivery charges. It comes to around 10 dollars for delivery charges from multiple sources. | | | |  | Not accounted for spare components ordered as a backup | | | |  | Only mentioned the components which were actually used in the project | | | |  | Wires, soldering gun, Solder wire, Wire- wrapping tool, wires etc. were used from the lab | | | |  |  |  |  | | Comments: | I would have not purchased a FTDI converter had I realised early that I could have borrowed another MSP from a senior and send the data over UART to display it on a PC | | | |  | I would have looked for a schottky diode which had a thin diameter suitable for perfboard | | | |  | With this experience, I would try to order maximum components in 1 or 2 orders and try to get equivalent components from the same vendor to sae on delivery charges. | | | |  |  |  |

## Appendix – Schematics

Following are the schematics for the Power supply designed (9v DC to 3.3V DC) MSP432P401R, AM2320, SI7021 and Nokia 5110 (48x 84) display sensor.



## Appendix - Firmware Source Code

Following is the Firmware code for the Sensor Monitoring System Project.



## Appendix - Software Source Code

Python code to capture the data from the COM port for data logging purpose as well as visual display of the readings in a graphical manner.



Also, the log output stored in the laptop:



## Appendix - Data Sheets and Application Notes



## Appendix – Picture with Sensor Monitoring System project

