Alarm Clock
EN1093 - Semester 2 Project

Wickramasinghe N.L. 190688X*, Widanagamage T.R. 190691B†,
Widanapathiranage D.N. 190692E‡, Wijegunawardana C.H.W. 190696U§
Department of Electronic and Telecommunication Engineering
University of Moratuwa, Sri Lanka
Email: *wickramasinghenl.19, †widanagamagetr.19, ‡widanapathiranagedn.19, §wijegunawardanachw.19@uom.lk

***Abstract*—Our objective was to design and implement an alarm clock that includes set time, date, and alarms throughout the day with the features of delete, change, and selecting an alarm tone. Initially, the complete code was implemented using Arduino C++. We transferred it into AVR C++ at the latter part of the project. For testing purposes, Proteus software was used. In parallel to the code, schematics and PCB layout were developed while having an idea of the conceptual design. The enclosure design was developed using Solidworks, including all the features needed to set the alarm for the user. Further, the physical product was developed to get final results, correcting all the impairments beforehand using proteus simulation. Throughout this project, we gained programming skills, improved logical thinking, PCB design skills, enclosure designing skills, and improved teamwork skills.**

I. Introduction

In this paper, we have discussed the development of a digital alarm clock. Our objective was to create an alarm clock with the functions of,

- Set current date and time
- Set an alarm with the features of change, delete, and selecting a tone
- Factory reset

We navigated through various challenges to identify and experience the whole process of developing a product from scratch to the final complete product.

The significant parts of the alarm clock are keypad input, LCD, alarm timing, and audio output. Complete firmware for the parts was developed separately and combined to develop the final firmware using AVR C++. The ATMEGA644P microcontroller was used to develop this product.

In this paper, we have included the methods of the development of,

- Firmware
  - Algorithms for each of the subsystems key input, menu display, alarm timing, and audio output
  - Final Algorithm
- PCB Design
  - Components used
  - Schematic Diagrams
  - PCB layout
- Enclosure Design

The objective of our project was to create a digital alarm clock that had a smooth user interface while having all the functions required in an alarm clock. We also worked on creating an accurate PCB that required very little space and, finally, creating a sleek and modern enclosure design for the product.

II. Method

A. *Firmware Development*

The task was to come up with an algorithm to implement an alarm clock with the following functions. The user should be able to add multiple alarms and add favorable tones to each of those alarms.

1) Set current time
2) List the existing alarms
3) Change or delete an alarm
4) Set a new alarm
5) Select an alarm tone
6) Factory reset

The system was planned to be implemented on ATMega328p microcontroller which has 32kB memory. But the memory was found to be insufficient for the developed algorithm mainly due the memory taken up to store multiple tones for alarms and other arrays. Therefore, the microcontroller was upgraded to an ATMega644p which has 64kB of memory and has a DIP package. The firmware was developed for this microcontroller using Microchip Studio. The algorithms were tested in several initial stages on Proteus simulation software. During the prototype development, the code was uploaded to the microcontroller using USBasp open-source programmer via the AVRDude interface.

A keypad was used to take number inputs from the user and an LCD was used to display the user interface of the clock. A sound system comprised of a buzzer was used to generate sounds for the alarms. The alarm clock keeps track of time and date using the RTC IC DS3231 which communicates with the microcontroller through the I2C interface.

The firmware development was initially divided into 4 main sub-systems in which each member developed a library of functions to perform a unique set of tasks. Based on these firmware libraries, four independent micro-products were developed to demonstrate each of the set of functions that will later be combined to create the final product. The sub-systems and each of their functions are in Table I.

Sub-system	Functions
Key Input	Read number input from a 4x3 keypad Detect button pressings
Menu Display	Display messages generated by the microcontroller on the 16x2 LCD
Alarm Timing	Set time and date Set/Delete alarms Detect alarms
Audio Output	Select multiple tones Drive a buzzer to ring a tone

Table I
SUB-SYSTEMS AND FUNCTIONS.

The microproducts were implemented in two stages. The team members developed the algorithms to implement each function and tested the algorithm using codes written in Arduino C++. These codes were tested on Arduino Uno boards using Proteus software. After satisfactory results the firmware was developed in AVR C++ for the ATMega microcontroller. All the tasks were written modularized into functions and all the inter-related functions were combined into header files(.h and .cpp files). The microproduct implementation was done using these header files.

Algorithms

1) Key Input: This sub-system provides two main functions from Table I. The keypad read function will wait for a key-input from the user when the function is called. Once the user presses a key in the keypad, the function returns the pressed key as an integer. Figure 1 shows the algorithm of that function.

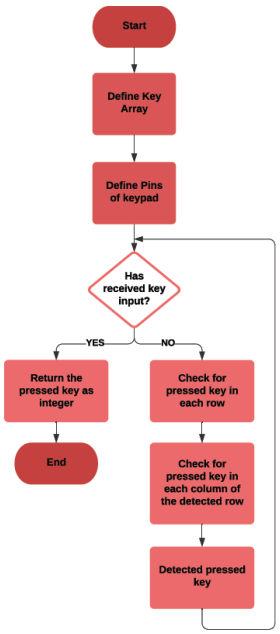

Figure 1. Keypad Read Algorithm

The button press function checks if a switch has been pressed and return True if yes, False if not.The algorithm is illustrated in Figure 2.

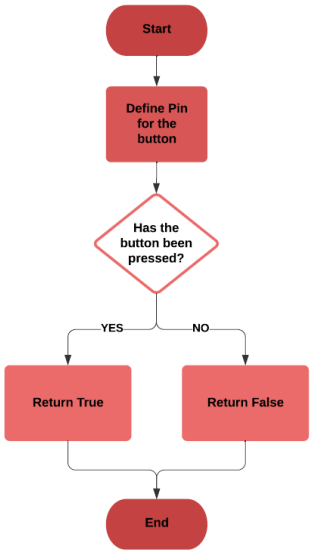

Figure 2. Button Press Algorithm

2) Menu Display: Two types of functions were written for controlling and displaying messages on the LCD.

- Control Functions
- Display Functions

The Table 2 and Table 3 shows the control functions and display functions respectively.

Function Name	Task Performed
LCD_Command(cmd)	Sending unique commands to setup the LCD
LCD_Char(char)	Sending unique characters to setup the LCD
LCD_Init()	Initial setting up of the LCD
LCD_String(string)	Sending a string to LCD
LCD_String_xy(row,col,string)	Sending a string to LCD with (x,y) position
LCD_Clear()	Clear LCD

Table II
DISPLAY FUNCTIONS OF LCD

3) Alarm Timing: The firmware for this sub-system was written based on objected oriented programming. Three classes were created with their own variables and methods. The classes are

- Time
- DS3231
- Alarm

a) Class Time: This class consists of the following set of variables that define the Time/Date. A single constructor method exists to create an instance of class Time.

- Hour
- Minute
- Second
- Year
- Month
- Day

Function	Snapshot
Home	17:56 12/08/2021
Menu	1>D&T 2>ALARM 3>RESET #>BACK
Invalid Input	INVALID INPUT
Set Time Menu	1>TIME 2>DATE #>BACK
Alarm List	1>12:45 2>08:00 3>ADD *>Next
Alarm Menu	1>SET 2>DELETE 3>TONE #>BACK
Select Tone	1>HARRYP 2>STARW 3>GOT 4>GREENL
Setting Digits	17:MM ^ #>RESET
Confirm Time/Date/Alarm	17:55 *>SET #>RESET

Table III
DISPLAY FUNCTIONS OF LCD

b) Class DS3231: This sub-system interacts with the RTC DS3231 IC to maintain the timekeeping. As aforementioned, the microcontroller communicates with this IC through I2C communication. This class contains the variables and methods required for that purpose and the functions for setting/reading time and date to/from the registers of the RTC IC. The necessary registers of the IC were referred from it's datasheet. [1] The following necessary set of functions that interface the I2C communication[2] and encoding/decoding functions were written. They are provided in Table 4.

Function Name	Task Performed
begin	Initialize pins and bit rate and enable TWI
burst_read	Send address to start reading and read from that address
write_register	Send address to write value and write to that address
decode	Convert hex representation to integer value
encode	Convert integer value to hex representation

Table IV
I2C FUNCTIONS

Then these functions in Table 5 were used to write the following functions to be used in the timekeeping.

Function Name	Task Performed
set_Time	Write time to necessary registers in the IC
set_Date	Write date to necessary registers in the IC
get_Time	Read Time/date from registers and return the time as an object of Time
get_TimeStr	Return a string of time
get_DateStr	Return a string of date

Table V
TIMEKEEPING FUNCTIONS

Using the above three classes, five central functions were written to perform the following tasks.

Function Name	Task Performed
Timing_set_time	Read input from keypad and set time
Timing_set_date	Read input from keypad and set date
Timing_set_alarm	Read input from keypad and set an alarm
Timing_delete_alarms	Delete all alarms
Reset_time	Factory reset

Table VI
ALARM TIMING FUNCTIONS

The algorithm that was being used to set time/date/alarm is given in Figure 3.

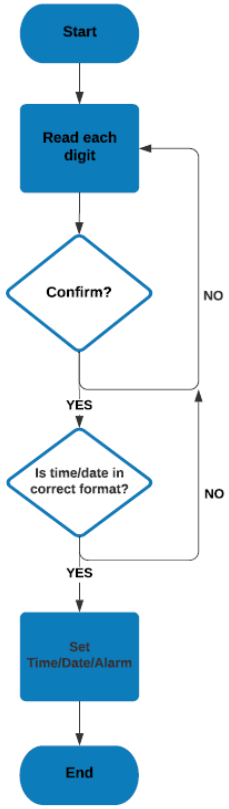

Figure 3. Set Time/Date/Alarm Algorithm

4) Audio Output: All the notes with the durations were taken from a github repo [3].They were modified to create two seperate header files. One header file "notes.h" defined the frequencies of each note of music used in the tones. The second header file "tones.h" defined two arrays with the notes and their respective durations for each of four tones as follows.

- Harry Potter
- Star Wars
- Game of Thrones
- Green de Leaves

A class Tone was created with the following variables and two functions.

- Array of note

- Array of durations
- Length
- Rate

The following functions in Table 7 were implemented to perform the respective tasks.

Function Name	Task Performed
Play_note	Play the note for the respective duration on the buzzer
Audio_play	Play the tone until the user stops the alarm

Table VII
AUDIO OUTPUT FUNCTIONS

The following Figure 4 and Figure 5 show the algorithms for play_note and audio_play functions.

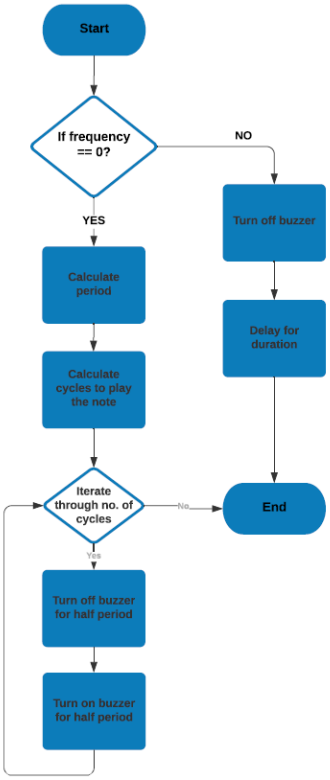

Figure 4. Play Note Algorithm

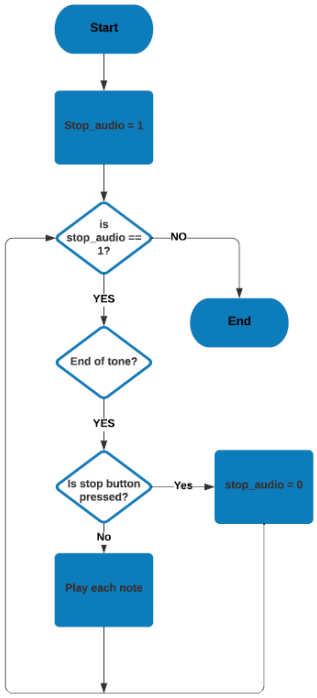

Figure 5. Audio Play Algorithm

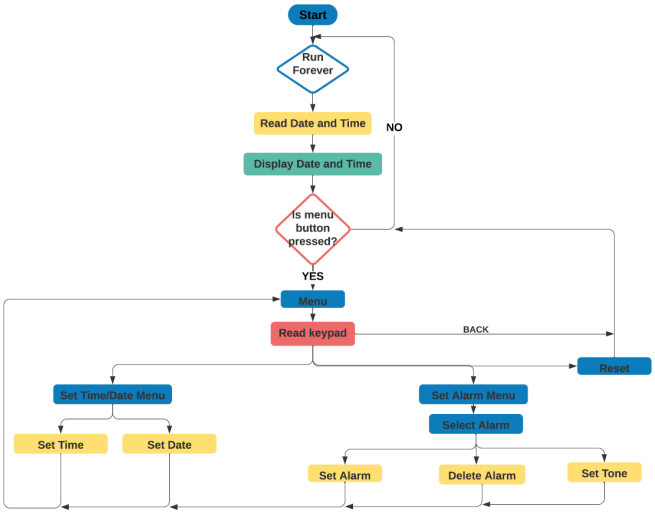

Figure 6. Main Algorithm

Final Product

The algorithm of the final product implemented a user interface of a menu system for the alarm clock according to the following algorithm in Figure 6. It makes use of functions of the above explained header files.

The checks for alarm times were implemented using software interrupts. The reason for using interrupts was to detect alarms even when the user is in the middle of the menu. The Timer Overflow Interrupt of 16-bit Timer/Counter 1 of ATMega644p was used in this case. The interrupt service routine (ISR) was written to iterate through each alarm in the alarm list and check whether the time is equal to any of the active alarms. If found, its ring function will be called.

Between the menu, an error message was programmed to be displayed whenever the user entered an invalid input.

The AVR C++ codes of the firmware of the alarm clock can be found at the GitHub link in the appendix.

B. *PCB Design*

1) List of components used:

a) Microcontroller: According to the Atmega644p microcontroller datasheet, 0.1 uF bypass capacitor was provided to connect VCC and GND. [4]

- Atmega644P microcontroller
- 0.1 uF capacitor
- Reset switch
- 10k resistor

b) Power regulator: LM7805 has been selected as the regulator and according to the datasheet the two capacitor values were chosen. [5]

- 2 pin header for the power supply.
- LM7805
- 0.33 uF capacitor
- 0.1 uF capacitor

c) Programmer:

- 10 pin header for the USBASP

d) Buttons:

- 2 pin header *2 for the menu button and stop button.
- 10k resistor

e) Keypad:

- 7 pin connector for the keypad
- 10k resistor*4

f) LCD display:

- 12 pin connector for the lcd display
- 1k resistor

g) RTC clock: According to the DS3231 datasheet values of the resistors have been decided. [1]

- DS3231 IC
- 10k resistors*2
- 2 pin header for the coin cell battery

h) Buzzer:

- 1N4001 diode
- BC547 transistor
- 220 resistor
- 2 pin header for the buzzer

i) Oscillator:

- 16 MHz crystal oscillator
- 22 μF capacitor*2

Reasoning for use of particular components

Microcontroller: Atmega644P microcontroller was chosen for the project due to several reasons. First, the program was implemented using 328P microcontroller and the memory of that was not enough. Atmega328P has 32 Kbyte flash memory, 1 Kbyte EEPROM and 2 Kbyte RAM. Then the microcontroller was upgraded to Atmega644P [Figure 7, Figure 8] because it has 64 Kbyte flash memory, 2 Kbyte EEPROM, 4 Kbyte RAM and 40 pins. [4]

Keypad: 4*3 keypad was chosen instead of 4*4 keypad because only '*' button and '♯' button were taken other than the number buttons for the control of going back and set functions.

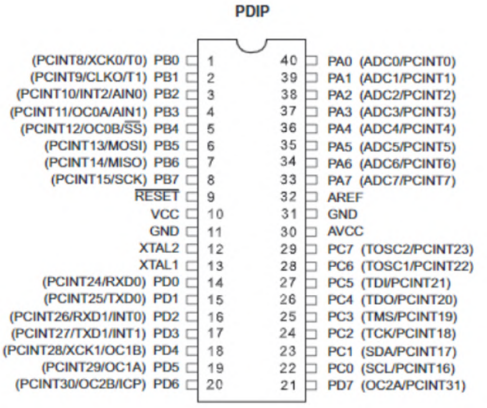

Figure 7. Pinout644P

Figure 8. Atmega644P

Switches: The purpose was to minimize the number of switches in the enclosure for the convenience of the user. Even through a button of the keypad can be taken for the menu button, a separate switch is provided because the user will be used menu button several times and if there's a separate switch user can recognize it easily as menu button other than providing a keypad number as the menu button. There is another switch for stop alarm because if it is given as a keypad button, user gets difficult to off while awake in sleep. Therefore, a separate button is provided for the stop alarm, then user can easily switch off.

Real Time Clock(RTC) IC: After comparing DS1307 and DS3231, DS3231 module was selected. DS3231 IC is more accurate than DS1307 IC but it is existing only as a SMD component. DS3231 has been chosen because it's not an issue for soldering. [1]

Buzzer circuit: A transistor has been used for the amplifying purpose and a diode has been used to make the protection for the buzzer.

Batteries: There are two batteries, 1.5V, 4AAA batteries are used as main voltage supply and a silver coin cell battery is used as a backup battery for the RTC module. Even though there are two batteries, the user doesn't always have to replace the coin cell battery, because the coin cell battery provides current to the RTC module only if there isn't a main voltage supply. The required current for the DS3231 IC module is given in the datasheet as 200 uA. The capacity of a silver coin cell battery is 200 mAh. If coin cell battery provides 20 hours of current per month, the user can use one coin cell battery

for about 4 years without replacing it. The required voltage for the circuits is 5V. Then it should supply 5v or more than that. The batteries available in the market are AA and AAA batteries. Therefore four 1.5v batteries must be provided. The size of the AA batteries is quite larger than AAA batteries therefore the enclosure battery holder has to be enlarged. As a result, 4 AAA batteries were selected.

2) PCB layout: The Schematic diagram and the PCB Layout made for the design of PCB can be found in Figure 19 and Figure 20 respectively.

Considerations in PCB design

- The board size, neatness and the shortest routing path were generally considered.
- The placement of bypass capacitor(C3) near to the micro controller according to the data sheet.
- Minimized routing traces through the oscillator and placed near to the micro controller.
- Capacitors of the oscillator, placed near to it (C4, C5).
- The same value of resistors placed in line for the convenience of soldering purpose.
- All header connectors were placed close to edges of the board because of the convenience of connecting wires.
- Capacitors of the regulator, placed near to it. (C1, C2)

The dimensions of the final PCB were 61.6 mm x 45.3 mm. The top and bottom of the 2D and 3D views can be found in Figures 21, 22 and Figures 23, 24 respectively.

C. *Enclosure design*

Designing the enclosure is one of most important parts in the process of product development. A product enclosure should have the characteristics of

- Strength – To withstand the stresses during normal use
- Functionality – Have the necessary components for proper functionality of the product
- Attractive – A sleek, aesthetic, and modern design to attract potential buyers
- Easy to handle – The product should be able to be operated by anyone without prior experience about the particular product

The goal of our team was to create a suitable enclosure for the housing of the electronics of our digital alarm clock. Our aim was to create an enclosure that would have the necessary strength for day to day use. It should also have the requirements to be easily opened for repairs and maintenance while having a very unique and aesthetic design.

Our enclosure had to be able to house the following components within the enclosure.[Figure 9]

- 2 x Momentary switch
- 1 x Printed circuit board along with the electronics
- 1 x Coin cell battery holder
- 1 x 4xAAA Battery holder
- 1 x Buzzer
- 1 x 4x3 Matrix keypad
- 1 x LCD screen display
- Necessary Wires

(a) PCB (b) Keypad (c) 4xAAA Battery Pack (d) Momentary Switch (e) Coin cell battery holder (f) Buzzer (g) LCD

Figure 9. Components to be placed within the Enclosure

Figure 10 shows the initial sketches we had planned for the enclosure design of the digital alarm clock.

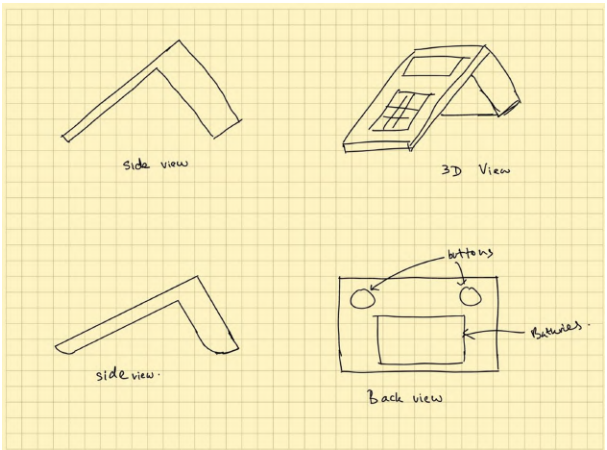

Figure 10. Initial sketches

Our Final enclosure is split into 3 separate parts.

- Bottom part
- Top part
- Battery cover

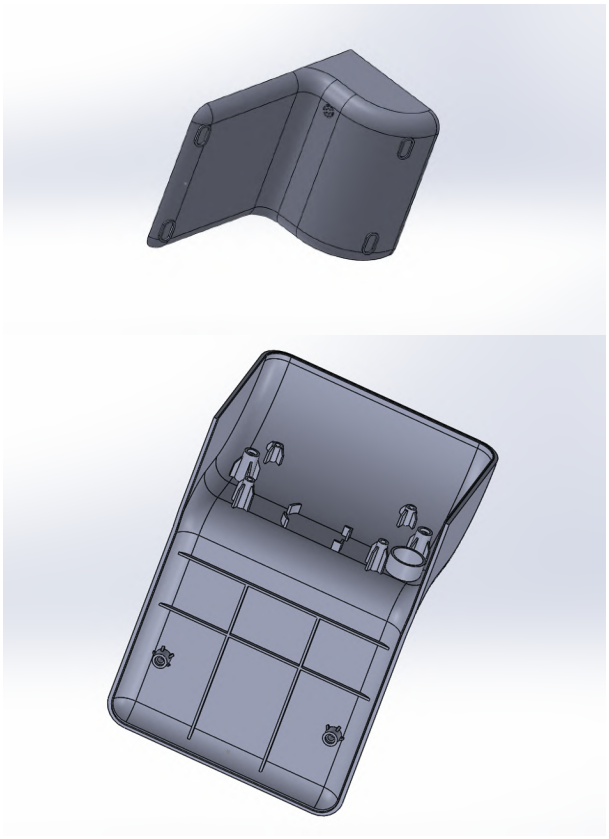

Figure 11. Bottom part

1) Bottom part [Figure 11]: The bottom part gets connected to the top part using four mounting bosses. Also, lips and grooves are etched onto the edges where the two parts meet. It also contains ribs for additional support and high strength. It also helps to maintain shape under high-stress conditions. The bottom part houses the coin cell battery, PCB, and the buzzer. We created a separate slot to accommodate the coin cell battery. The PCB is mounted on 4 mounting bosses. A circular slot is separately carved to accommodate the buzzer. A vent is created on the bottom side to pass the sound of the buzzer outside. 4 straight slots are etched onto the bottom surface, where rubber pads will be attached later on. The rubber pads on the bottom will help stabilize the product and help stop slipping.

2) Top part [Figure 12]: The top part too includes the required mounting bosses to connect to the bottom part along with the lips and grooves. It houses the 4x3 matrix keypad, LCD, 4xAAA battery holder, and the two momentary switches. Separate holes are cut out for the momentary switches to be attached. The holes have a carving above them so that the user can easily identify the function of the switch. A 1mm depth slot is carved out in the front face to accommodate the 4x3 matrix keypad. A small slot is cut at the bottom to get the wires inside the enclosure. A large window is cut at the top part of the front face for the LCD screen. Also a large slot is created at the back to accommodate the 4XAAA battery

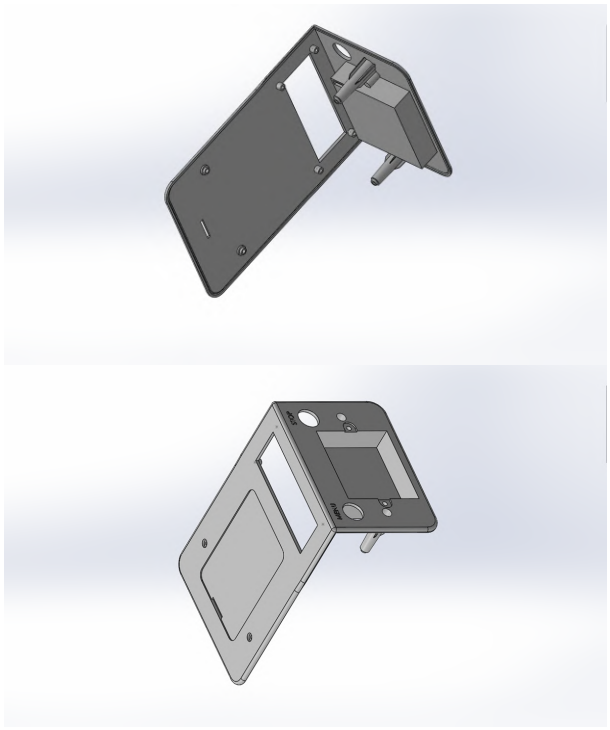

Figure 12. Top part

holder. It also contains the holes required to attach the cover of the battery compartment.

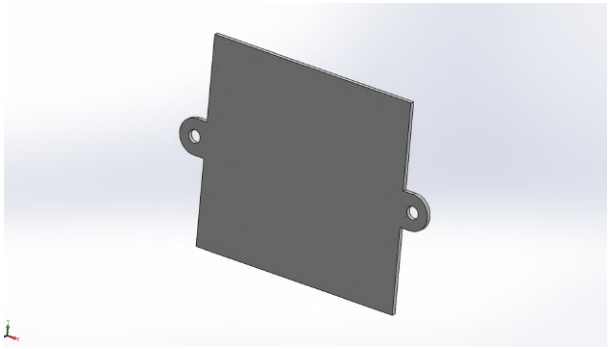

Figure 13. Battery cover

3) Battery cover [Figure 13]: A separate cover is created in order to cover the 4xAAA battery holder. The cover is fastened on to the top part using two screws.

The bottom part was colored with blue and the top with white for the final product. We used SolidWorks software for the creation of the 3D models. [Figure 14]

III. Results

The alarm clock design was tested using the schematic in [Figure 15] using Proteus simulation software. The design was tested for setting time and date, setting multiple alarms, deleting alarms, detecting the alarms, stopping the alarm and resetting functions. All four tones were tested successfully.

Figure 14. Final Enclosure

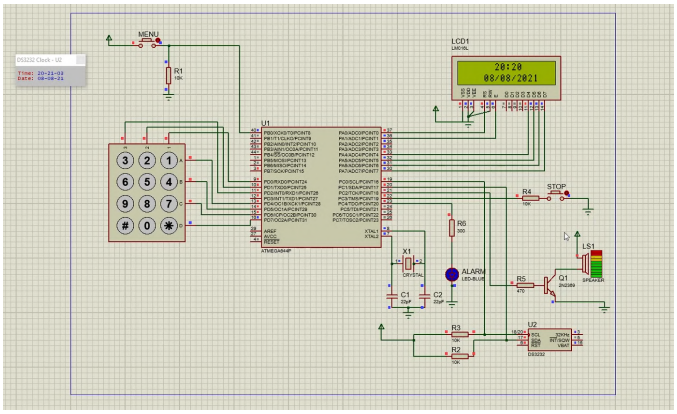

Figure 15. Proteus Simulation

A recording of the Proteus Simulation could be found by following the Google Drive link in the appendix. After the algorithm and the schematic were confirmed using Proteus, the circuit was implemented on breadboard to be evaluated on real-life aspects. The circuit performed as expected giving similar results as in Proteus simulation. [Figure 16] shows the breadboard testing.

Then the PCB was manufactured from JLCPCB from China and the components were purchased from Arrow Electronics. [Figure 17] [Figure 25] [Figure 26] The components were hand soldered to the PCB and tested using a prototype in [Figure 18]

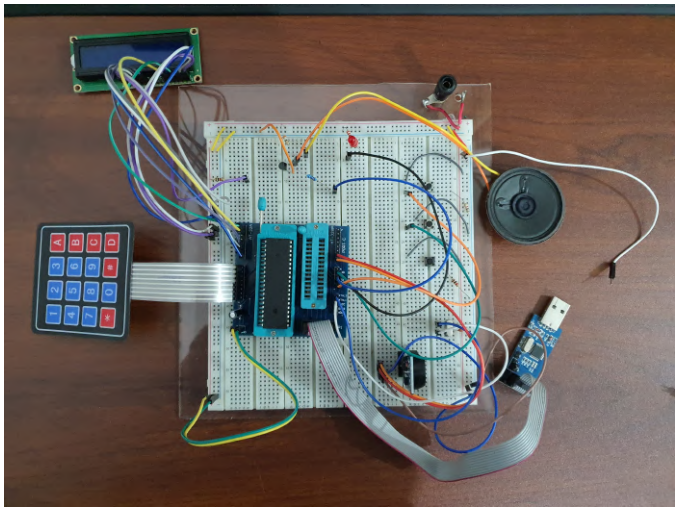

Figure 16. Breadboard Testing

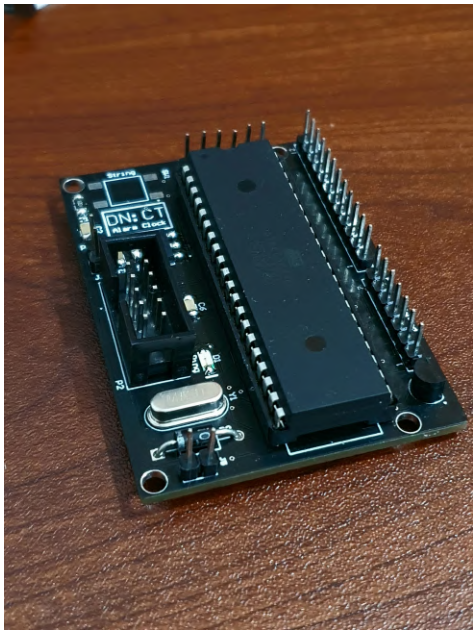

Figure 17. The Fabricated PCB

IV. Discussion

There were several challenges we faced during the project of designing the alarm clock. During the development of the firmware, it was required to combine modularized code written by several people into a single project. To make it easier, a flowchart was drawn for the central algorithm and the functions were modified into a common code writing style. The codes were well documented to be understood by other collaborators. After making the circuit on the breadboard, the Port C of the micro-controller happened to be not working properly, supposedly due to the continuous I2C communication through the same port. The components on Port C were shifted to unoccupied pins in Port B. However, this error did not occur in the Proteus simulation which emphasized the

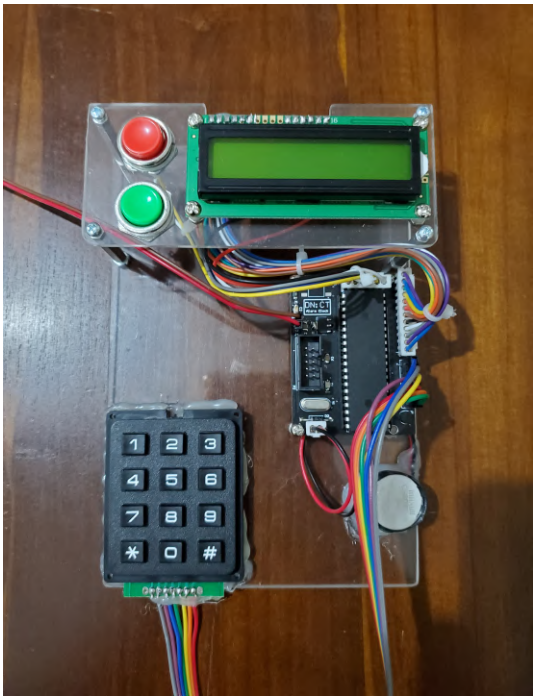

Figure 18. Prototype

importance of testing the circuit in physical conditions.

Compacting components to get a smaller PCB was a challenge when designing our PCB. Specially, the upgraded micro-controller Atmega644P is considerably larger than the previous Atmega328P which in turn increased the form factor. The final dimensions of the PCB were 61.6 mm x 45.3 mm, which is satisfactory. The PCB was manufactured by JLCPCB. Therefore, it was required to adhere to the rules given by JLCPCB. The pads around the pins of the transistor in its footprint were too close to each other, violating clearance rules. To solve this, the size of the pins of the transistor were taken from it's data sheet and the footprint was adjusted. When placing the components, the components which are taller in height like headers were placed only on one side of the PCB to avoid being required to keep space on both sides of the PCB when it is being mounted in the enclosure. Also, even though the coin cell battery can be inserted on the PCB itself, it was placed in the enclosure because the purpose was to make PCB smaller in size.

Software, such as SolidWorks, helped us to easily design unique and functional enclosures for our electronic products. Most of the 3D models for the components we used could be found online, which was very helpful when deciding the required dimensions for the enclosure. Although we planned to 3D print and assemble our final product, we were unable to do it due to not being able to gather the some of the components within the given time frame. But, the the error analysis method given by the SolidWorks software itself showed no errors with the final enclosure [Figure 14].

There are several improvements that could be made to the design of this alarm clock. The LCD could be connected to the micro-controller using an I2C multiplexer, which will reduce the number of pins required. Also, in our current design, 4 AAA batteries were used to supply 5V power to the electronic components. But this makes the alarm clock bulky and inconvenient for the user who will have to replace 4 batteries when they drain out. Instead, a boost converter could be integrated into the circuit to draw power from a single AAA battery. However, this was not implemented in the current stage of development, because it was not in our knowledge scope yet.

V. Conclusion

Design of the alarm clock for our semester 2 project for the module EN1093 helped us in gaining many technical skills as engineers. Developing firmware for an electronic product using ATMega micro-controllers, designing a schematic and a PCB layout for an electronic circuit using Altium, fabricating and soldering a PCB, designing a commercially viable enclosure for an electronic product using Solidworks and testing an electronic product for design verification were the most valuable skills we gained during the project. We adhered to the important stages of the product development life cycle and gained social skills of working as a team to achieve a common goal. Our design of the alarm clock performed all the functions of the project requirements, contained an easily comprehensible user interface using an LCD and a keypad, and possessed a stable and attractive enclosure.

Acknowledgment

We would like to express our sincere gratitude to our supervisors Mr. Kithmin Wickramasinghe, Mr. Abarajithan Gnaneswaran and Mr. Pasan Dissanayaka, for their enthusiasm, insightful comments, helpful information, and practical advice. All of them helped tremendously during the completion of this project. The project would not have been possible to complete without the support and guidance from the supervisors. We thank our parents for the financial support given to us in completing this project.

References

[1] Maxim Integrated, "Ds3231 datasheet," https://pdf1.alldatasheet.com/datasheet-pdf/view/336170/MAXIM/DS3231.html, accessed 15-07-2021.

[2] California State University, Long Beach, "Serial communications and i2c," http://web.csulb.edu/~hill/ee346/Lectures/20%20C++%20ATmega%20I2C%20Serial%20Comm.pdf/, accessed 15-07-2021.

[3] R. Couto, "arduino-songs," https://github.com/robsoncouto/arduino-songs, 2021.

[4] Microchip Studio, "Atmega644pa datasheet," https://pdf1.alldatasheet.com/datasheet-pdf/view/313648/ATMEL/ATmega644PA.html, accessed 15-07-2021.

[5] Texas Instruments, "Lm7805 datasheet," https://pdf1.alldatasheet.com/datasheet-pdf/view/838007/TI1/LM7805.html, accessed 15-07-2021.

VI. Appendix

Link to GitHub Repository:
https://github.com/nima34366/alarm_clock

The following Google Drive link contains the following documents related to the project:
https://drive.google.com/drive/folders/1GlEg0UCcHJv6p5PrgpzKblIw5QNwTWeP?usp=sharing

- Microchip Project File
- Altium Project File
- Solidworks File
- Bill of Material
- Proteus Simulation
- Prototype Simulation

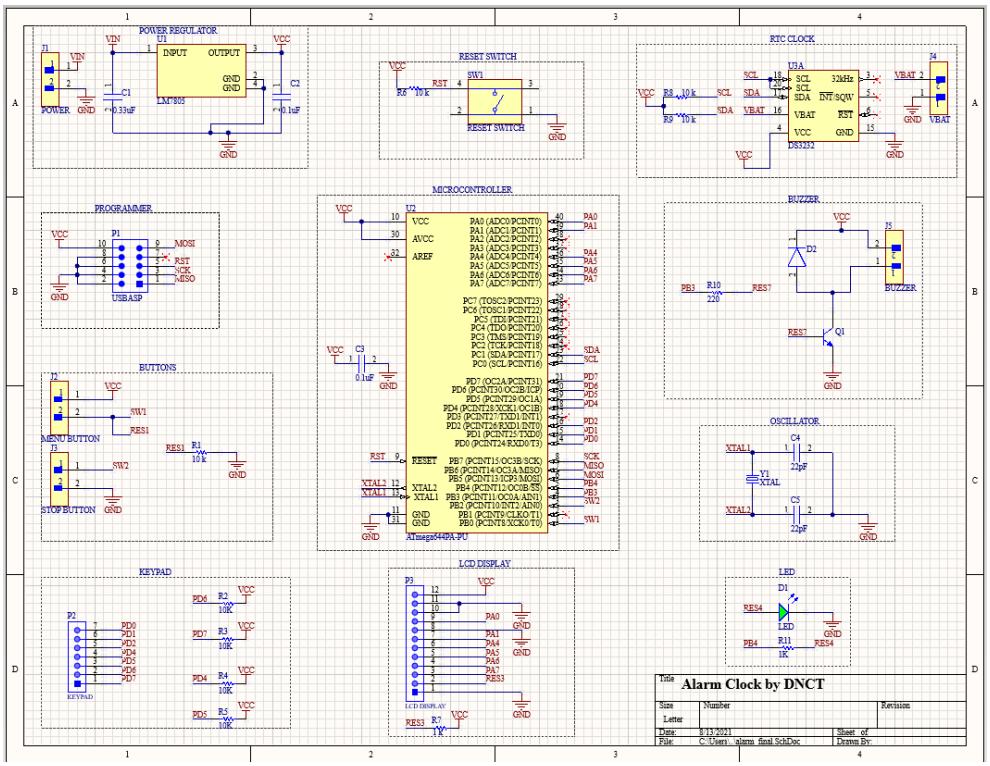

Figure 19. Schematic diagram

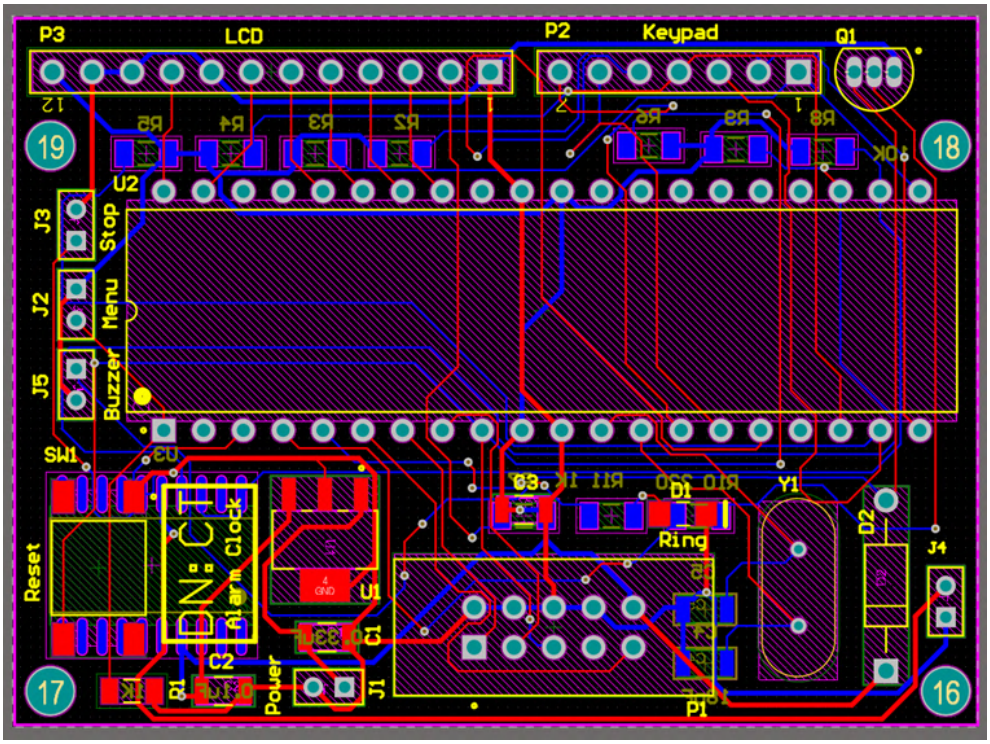

Figure 20. Layout diagram

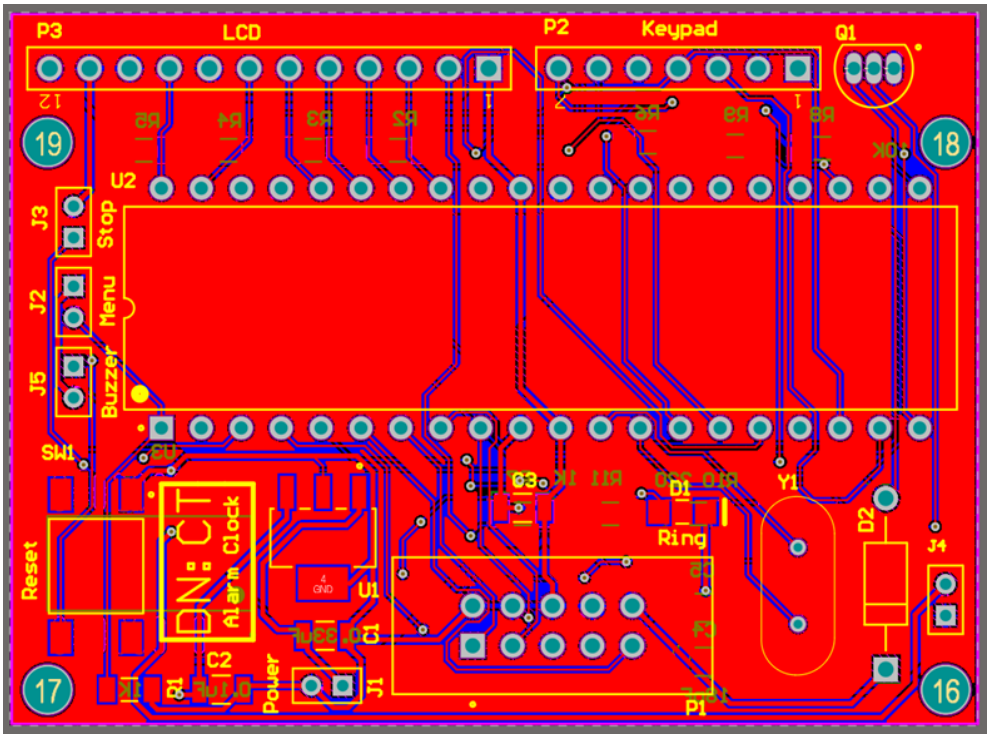

Figure 21. Top

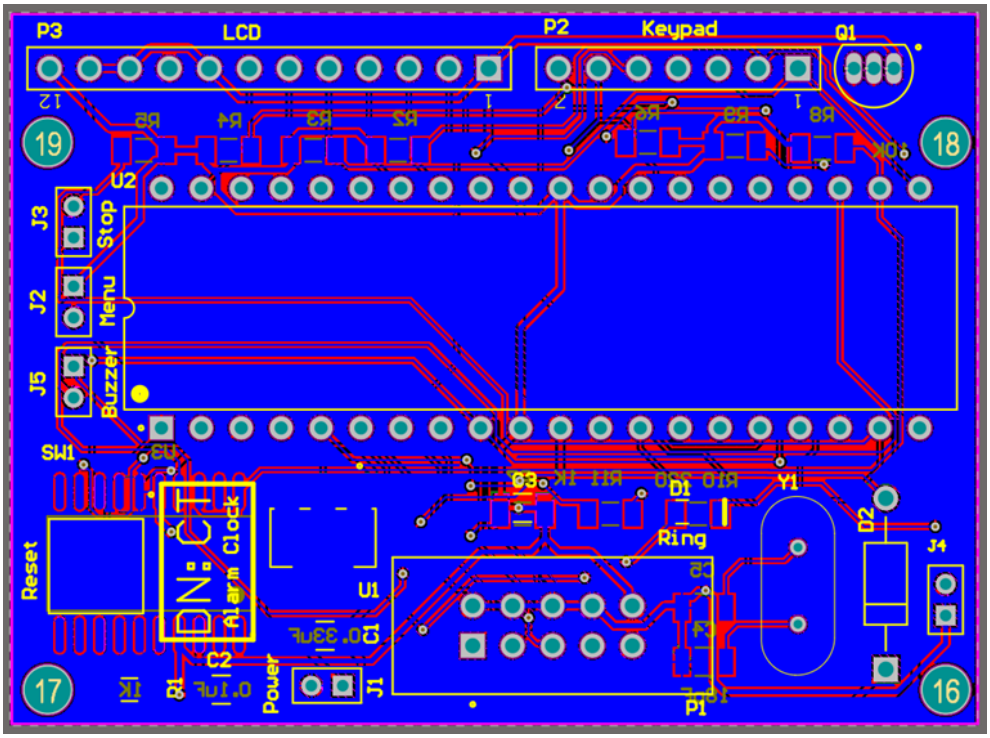

Figure 22. Bottom

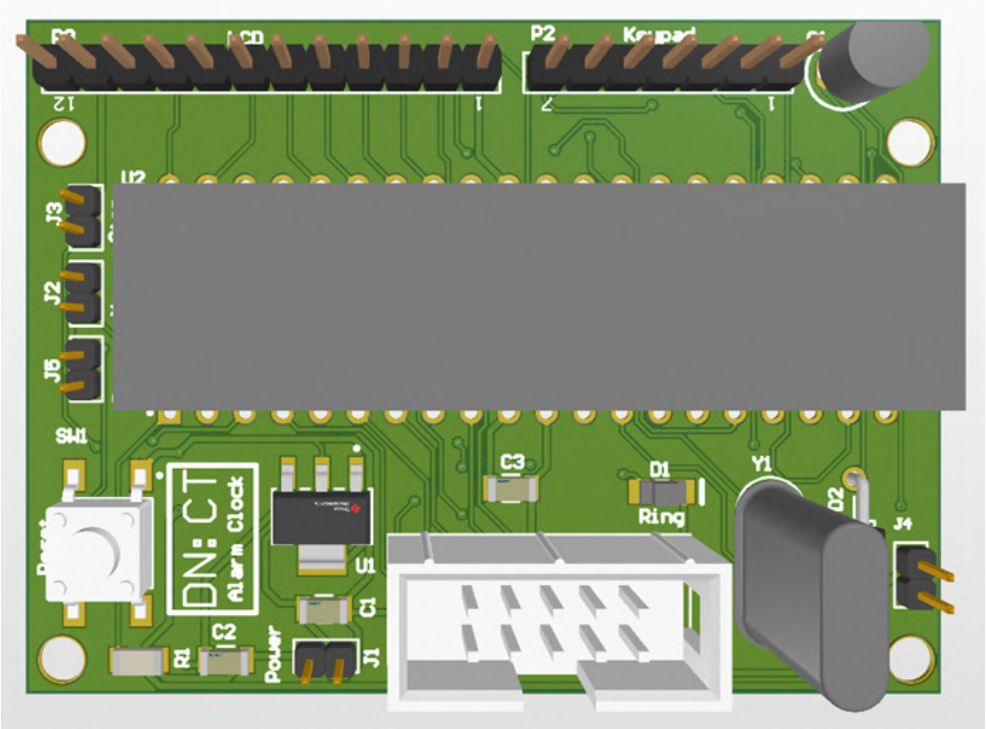

Figure 23. Top

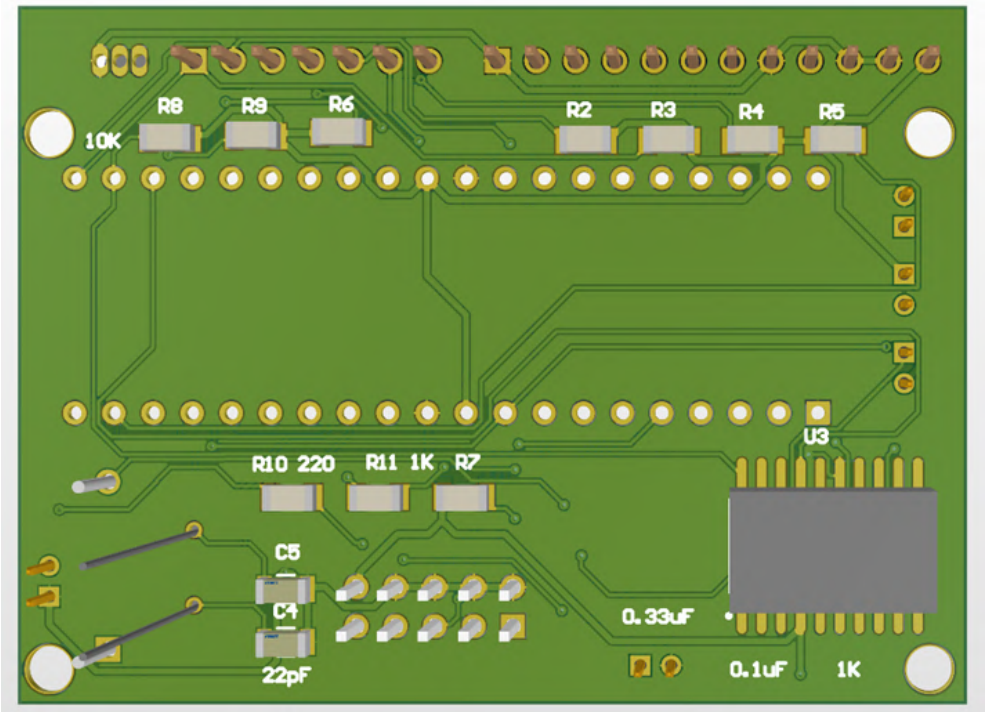

Figure 24. Bottom

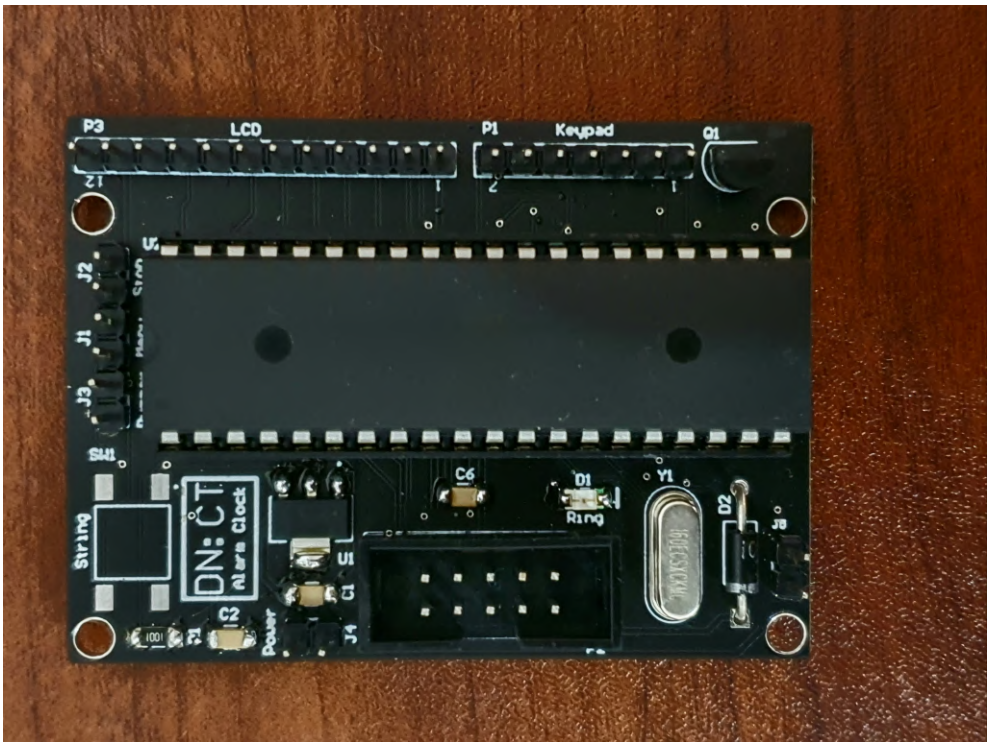

Figure 25. Top of Fabricated PCB

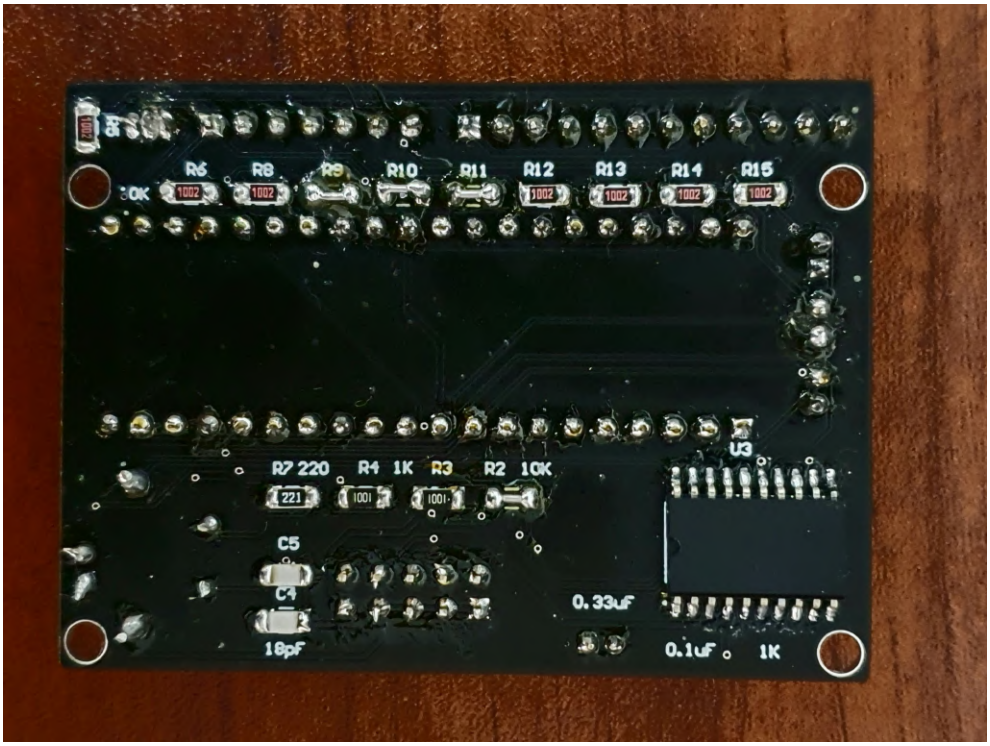

Figure 26. Bottom of Fabricated PCB