Project Report
EEE 301 & 302
Digital Electronics
Department of Electrical and Electronic Engineering

Semester: Summer 2021

Project Title

Digital security system

Sec: 02
Group No: 11

Group members

Name	ID	Email/Phone
Tanvir Mahmud Mahim	19121012	tanvir.mahmud.mahim@g.barcu.ac.bd
Tahmid Zarif Ul Hoq Sayor	19121015	tahmid.zarif.ul.hoq.sayor@g.bracu.ac.bd
Fardin Islam	19121002	fardin.islam@g.bracu.ac.bd
Nuren Tahsin	18321047	nuren.tahsin@g.bracu.ac.bd

Abstract:

Security of personal data or assets is an essential need in this modern world. Thereover, having a convenient way of securing one's personal info or assets has become a basic need nowadays. Although there exists a whole bunch of tools to provide such facilities. However, Common analog tools are generally vulnerable and arduous to use. Luckily a password protected based digital protection system can solve such issues. Again, such systems will not only be convenient but also ensure stronger protection.

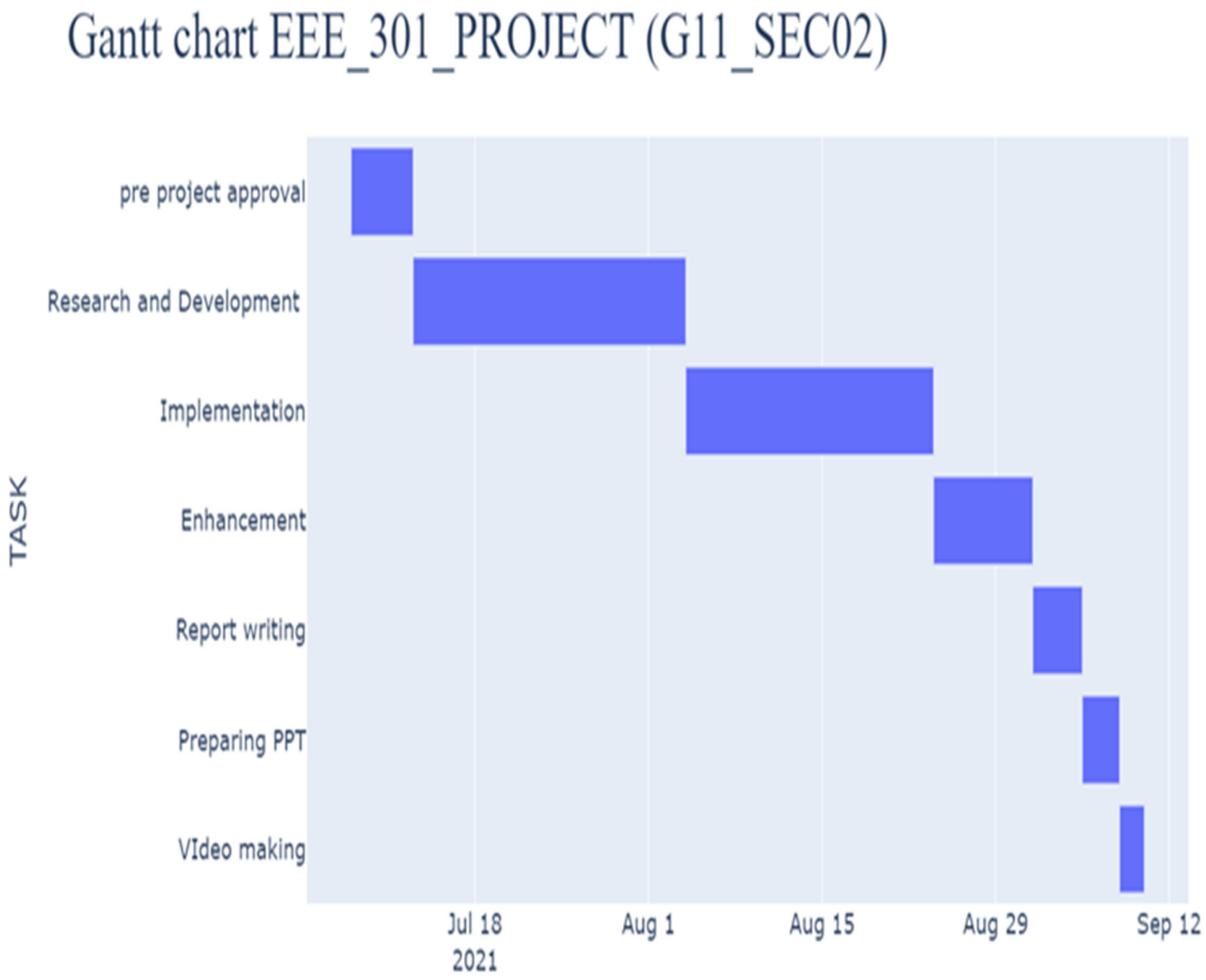
Objectives:

Our objective is to make a digital system that will contain digital security options such as password protected in-order to provide higher and alternative security for sensitive data and assets.

Motivation:

In order to prevent digital theft, security scams and other unethical problems our team have been motivated to build a digital password security system as Passwords provide the first line of defense against unauthorized access to your asset, computer and personal information.

Gantt Chart:


Gantt chart EEE_301_PROJECT (G11_SEC02)
TASK
pre project approval
Research and Development
Implementation
Enhancement
Report writing
Preparing PPT
VIdeo making
Jul 18
2021
Aug 1
Aug 15
Aug 29
Sep 12

Responsibilities:

TASK	START	END	Responsible person
pre project approval	7/8	7/13	All members
Research and Development	7/13	8/4	All members
Implementation	8/4	8/24	Tanvir Mahmud Mahim, Tahmid Zarif Ul Hoq Sayor
Enhancement	8/24	9/1	Nuren tahsin, Fardin Islam
Report writing	9/1	9/5	Tanvir Mahmud Mahim, Tahmid Zarif Ul Hoq Sayor
Preparing PPT	9/5	9/8	Nuren tahsin, Fardin Islam
Video making	9/8	9/10	Tanvir Mahmud Mahim

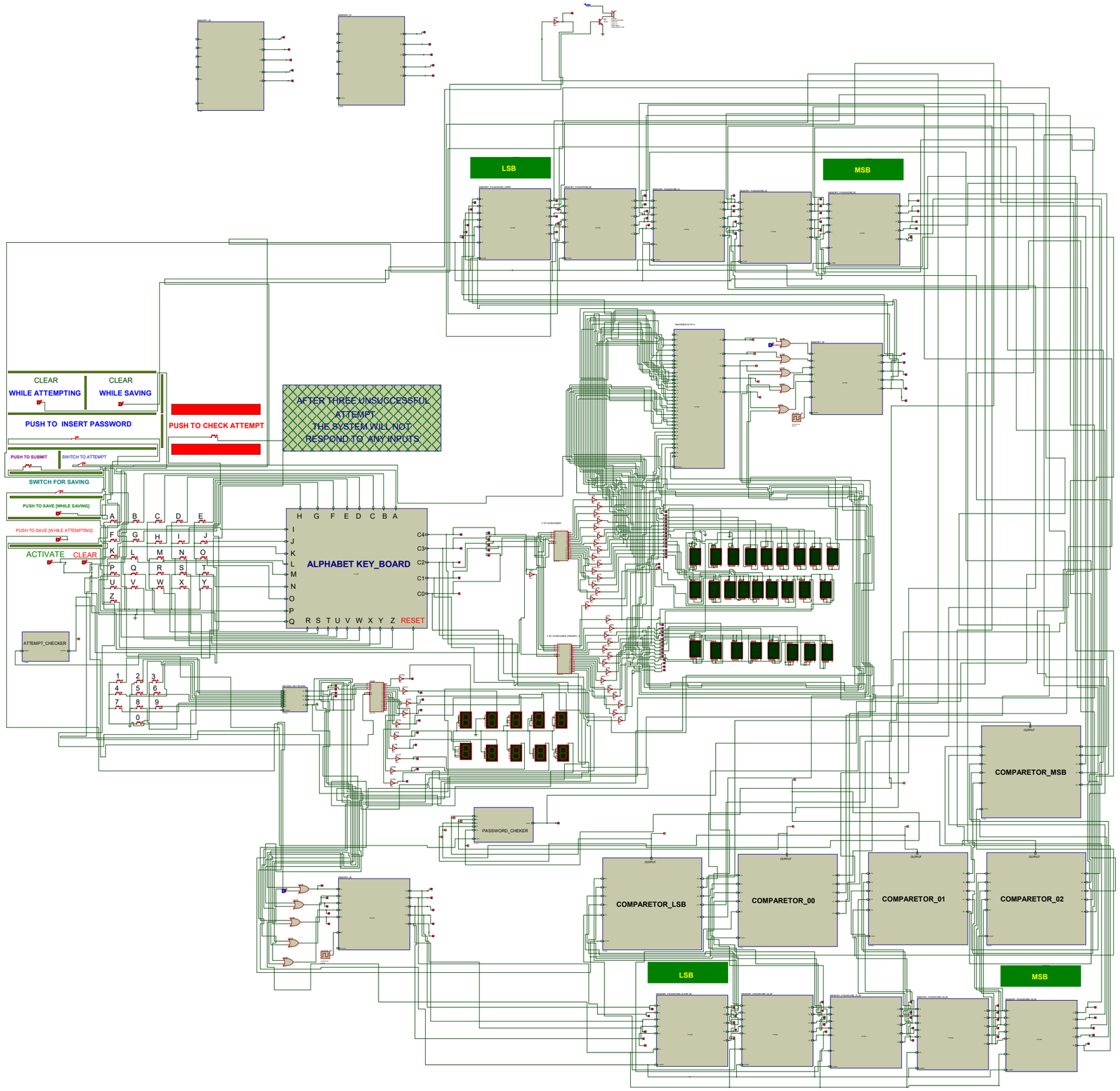
Specifications and Requirements [Equipment List]:

The specifications we used in our project are:

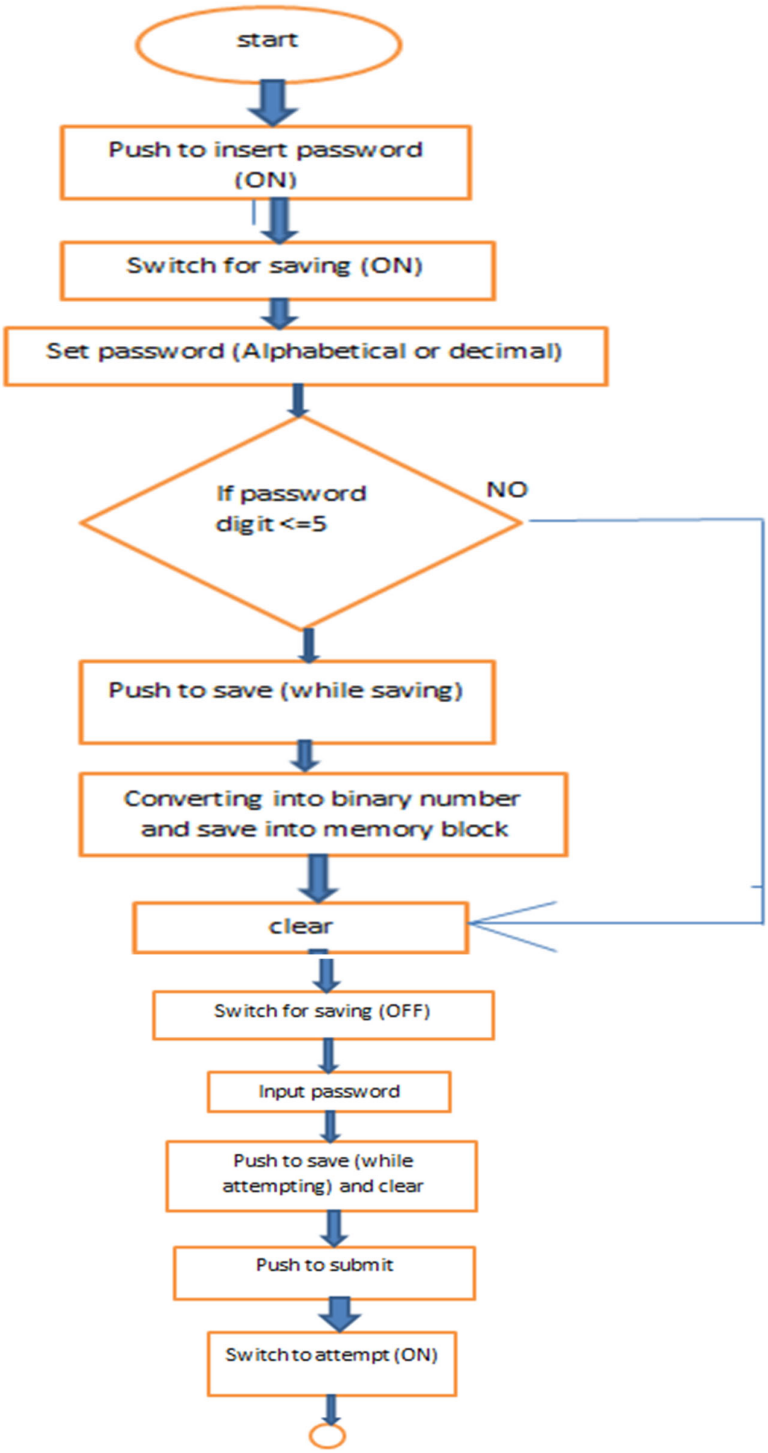
- Counters
- Registers
- Flipflops
- Encoders
- Decoders
- Comparator
- Checker
- 7 segment and 14 segment displays
- Logic Toggle
- Logic Probe
- Logic gates (AND, OR, XOR, XNOR, NOT, NAND, NOR)
- BJT
- Push Button, Relay, Switch, Buzzer

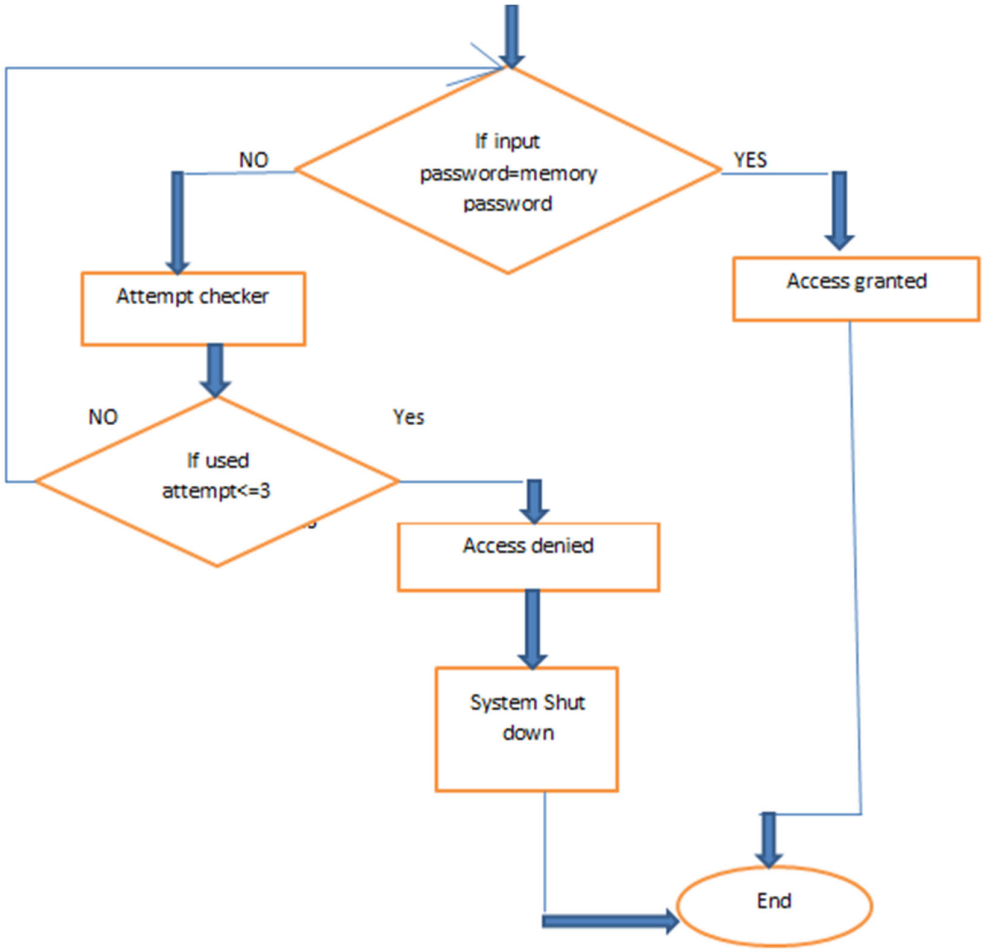
To get a overall visualization here is the birds-eye view of our systems design structure:

[Note: We have attached a high DPI design structure with key details at the end of this design paper]


LSB
MSB
CLEAR
WHILE ATTEMPTING
CLEAR
WHILE SAVING
PUSH TO INSERT PASSWORD
PUSH TO CHECK ATTEMPT
AFTER THREE UNSUCCESSFUL ATTEMPT
THE SYSTEM WILL NOT RESPOND TO ANY INPUTS
PUSH TO SUBMIT
SWITCH TO ATTEMPT
SWITCH FOR SAVING
PUSH TO SAVE [WHILE SAVING]
PUSH TO SAVE [WHILE ATTEMPTING]
ACTIVATE
CLEAR
ALPHABET KEY_BOARD
RESET
ATTEMPT_CHECKER
PASSWORD_CHEKER
COMPARETOR_MSB
COMPARETOR_LSB
COMPARETOR_00
COMPARETOR_01
COMPARETOR_02
LSB
MSB

System Design Flow:





Methodology:

Ideas of software implementation of the system design flow diagram in Proteus:

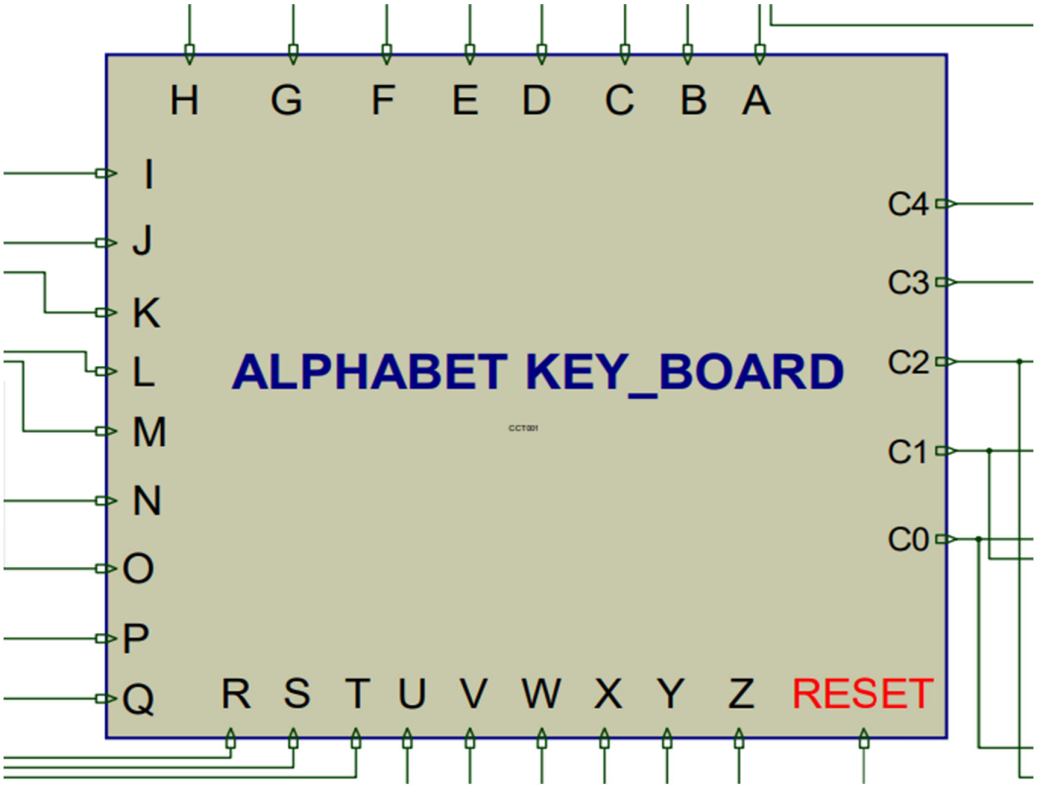
We have followed the above flow chart to come up with a logical circuit which can be implemented in Proteus. We have to keep inserting the password switch on while setting our password, which can be alphabetical (A-Z ,5 bit binary number) or decimal number (from 0-9 4-bit binary number). We have to keep our switch for the save button on to save our inserted password into memory. The

password we have set will be saved in our memory password block (memory password) which contains 5 shell (sub circuit) which implies that we can have the highest 5-digit password. Each sub circuit is made of D flip flop, BJT, register, relay switch. Working as a PIPO register. Then we will insert an input password to compare with the memory password. For this purpose, we have comparator blocks which have 5 comparator cells (sub circuit) which use X-NOR gates to compare the inserted password with the previously set password. Then the user input password will be saved into the memory block. We also have a password checker to check whether the user input and the memory password matches or not. If yes then the system will grant access. If not, then our buzzer will beep. We will get 3 attempts to correct our password. If we can't make it correct then within this attempt then the system will shut down. To count our attempts, we have an attempt checker which is made of D flip flop, OR gate, And gate, relay BJT, register. We have 26 display (14 segment display for each) to show from A-Z which are connected with 4 to 16 decoder and 4 to 16 priority decoder which converts them into binary number and its output is connected with 32 to 5 encoders. We have a decimal keyboard which is connected with 4 to 16 decoders converting them into binary numbers and its output is connected with 10 displays (7 segment display for each) to show 0-9 numbers. We have alphabetical keyboard and inside this a single alphabet is represented in 5-bit binary number by using five D flip flop (PIPO), relay, BJT, register and in decimal keyboard a single decimal number is represented in 4-bit binary number by using four D flip flop (PIPO), relay, BJT, register. Thus, we have made our expected logic circuit.

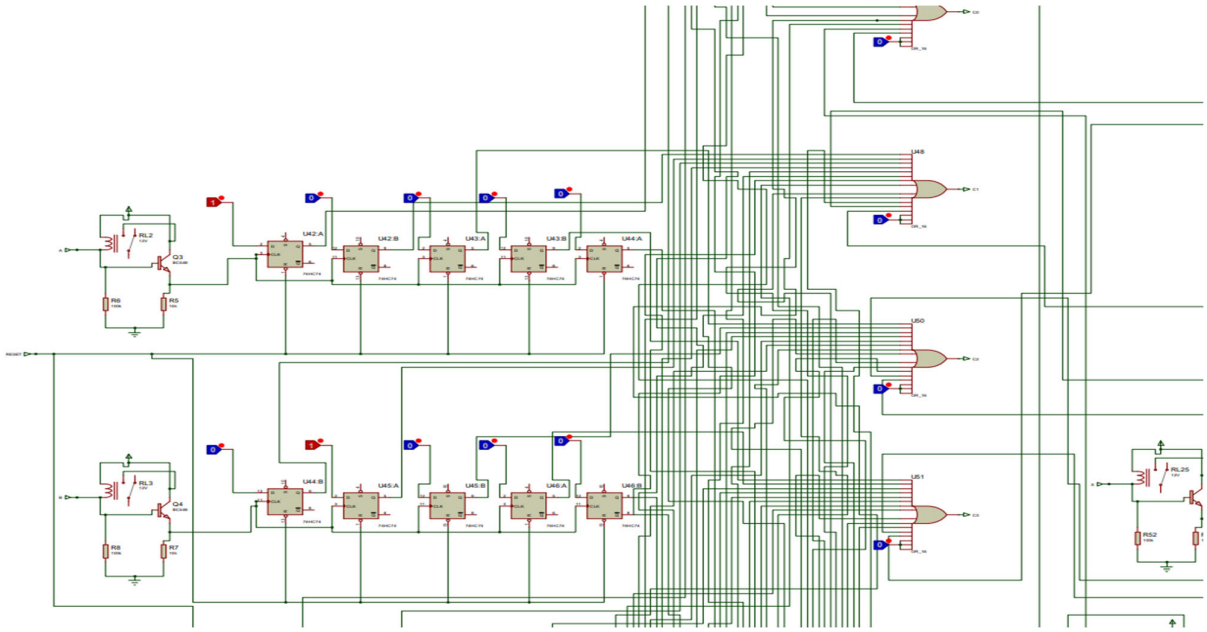
Input Keyboard:

Here we are using two keyboards using push buttons one is for alphabetical inputs another for numerical inputs. All these characters are encoded to binary equivalent and then sent for saving, comparing, checking purpose. Please do that for simplicity. We used the same binary representation for both Numerical & Alphabetical keyboards. Hence, we cannot use both the keyboards simultaneously.

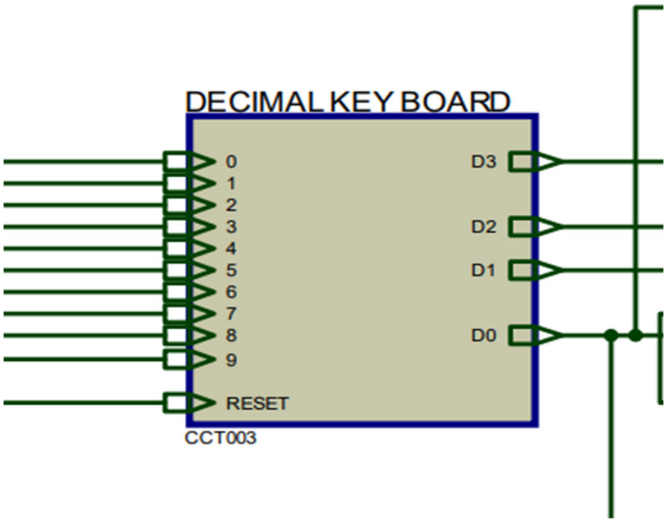
Alphabetical keyboard:



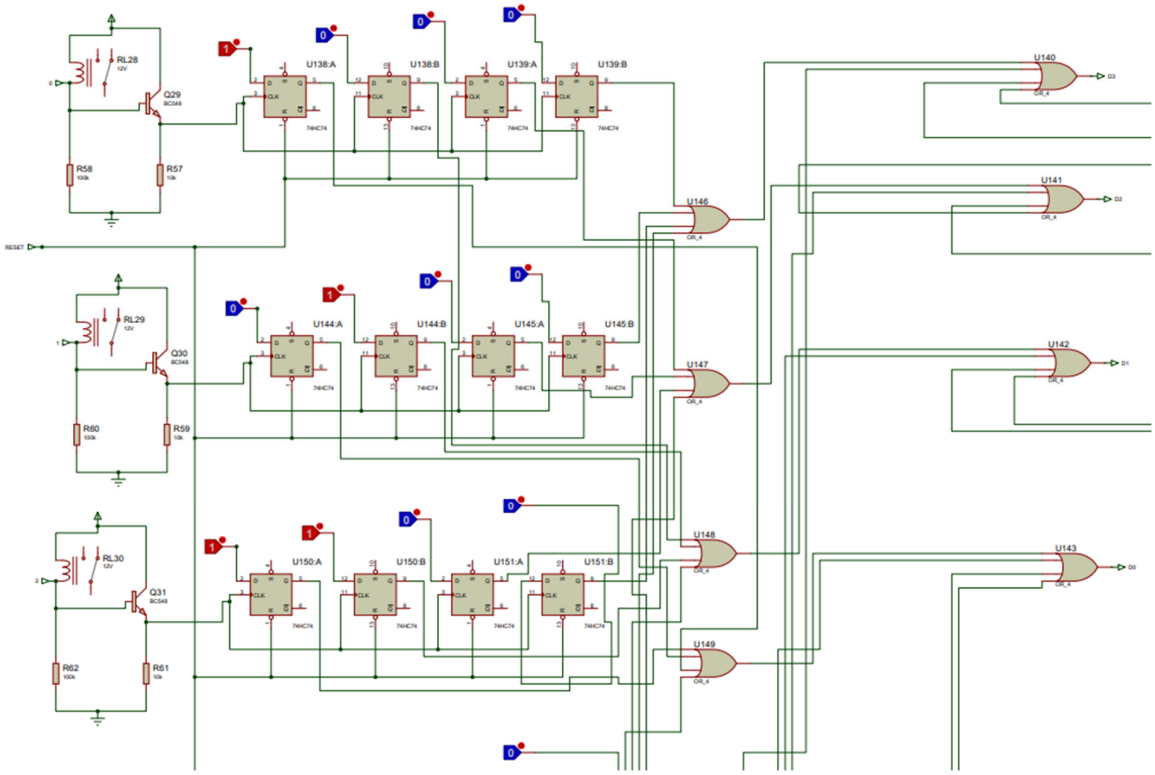
Sub Circuit Segment for Alphabetical Keyboard:



DECIMAL Keyboard:

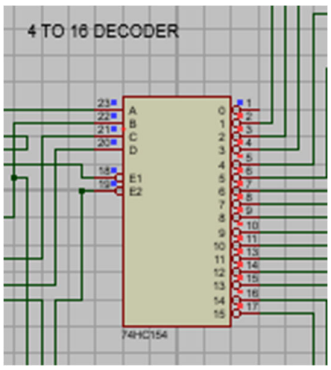


Sub Circuit Segment for Decimal Keyboard:



Decoder:

As the keyboard inputs are encoded. We use decoders to decode the characters to Display in monitor. Hence, we use 4 to 16 Decoders. As the alphabetical keyboard characters are encoded in 5 bits binary thus, we use two 4 to 16 decoders to make it 5 to 32 decoders.



Here we are just showing one 4 to 16 decoder truth table to justify our case of logic and also drawing a 5-to-32-bit output decoder is not feasible as well.

4 to 16 Decoder:

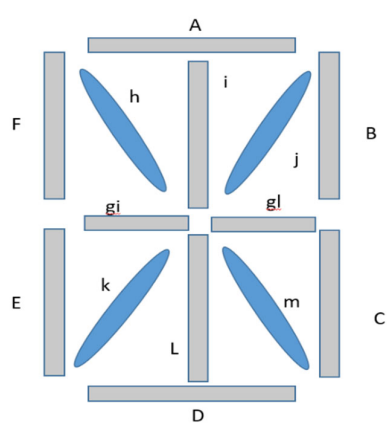
I_4	I_3	I_3	I_0	O_0	O_1	O_2	O_3	O_4	O_5	O_6	O_7	O_8	O_9	O_{10}	O_{11}	O_{12}	O_{13}	O_{14}	O_{15}
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Display:

After decoding we assign a display for each character. Specifying the lights to make that letter, here we showed the 14-segment display connection in order to display the alphabetical characters through the decoded signal of input 5-bit representation which we just showed above.

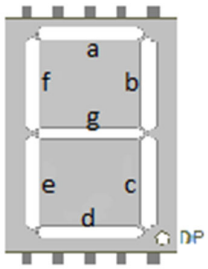
For 14-segment display**:**



I_4	I_3	I_2	I_1	I_0	a	b	c	d	e	f	h	i	j	g1	m	l	k	g2	KEY
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	hold
0	0	0	0	1	1	1	1	0	1	1	0	0	0	1	0	0	0	1	A
0	0	0	1	0	1	1	1	1	1	1	0	0	0	1	0	0	0	1	B
0	0	0	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	C
0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1	0	D
0	0	1	0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	1	E
0	0	1	1	0	1	0	0	0	1	1	0	0	0	1	0	0	0	1	F
0	0	1	1	1	1	0	1	0	1	1	0	0	0	1	0	0	1	0	G
0	1	0	0	0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	H
0	1	0	0	1	1	0	0	1	0	0	0	1	0	0	0	1	0	0	I
0	1	0	1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	J
0	1	0	1	1	0	0	0	0	1	1	0	0	1	1	1	0	0	0	K
0	1	1	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	L
0	1	1	0	1	0	1	1	0	1	1	1	0	1	0	0	0	0	0	M
0	1	1	1	0	0	1	1	0	1	1	1	0	0	0	1	0	0	0	N
0	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	O

1	0	0	0	0	1	1	0	1	1	0	0	0	0	1	0	0	0	1	P
1	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	Q
1	0	0	1	0	1	1	0	0	1	1	0	0	0	1	1	0	0	1	R
1	0	0	1	1	1	0	1	1	0	1	0	0	0	1	0	0	0	1	S
1	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	T
1	0	1	0	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	U
1	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	V
1	0	1	1	1	0	1	1	0	1	1	0	0	0	0	1	0	1	0	W
1	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	1	0	X
1	1	0	0	1	0	0	0	0	0	0	1	1	0	0	0	1	0	0	Y
1	1	0	0	0	1	0	0	1	0	0	0	1	0	0	0	0	1	0	Z

For seven segment displays:



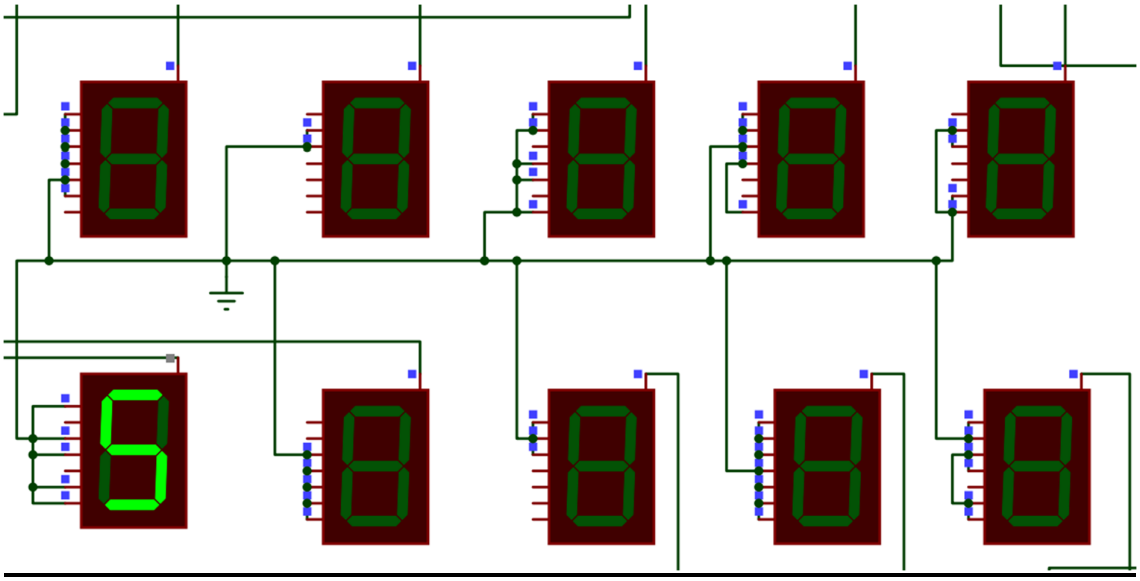
7-Segment

The numerical keyboard we used 7 segment displays and this table is the connection circuitry for the 7-segment display.

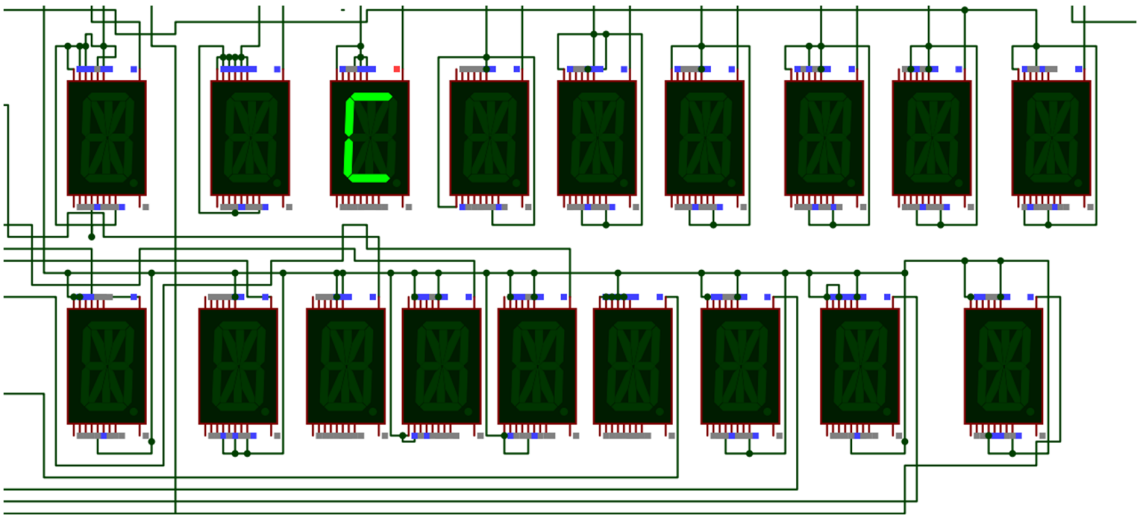
KEY	I_3	I_2	I_1	I_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0

8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

Display for seven segments:

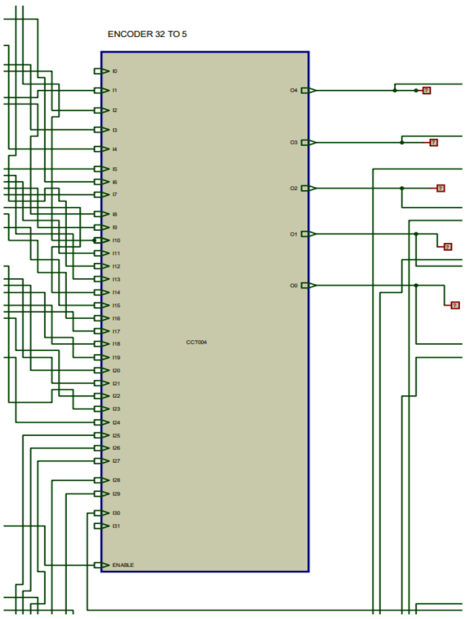


Display for 14 segments:

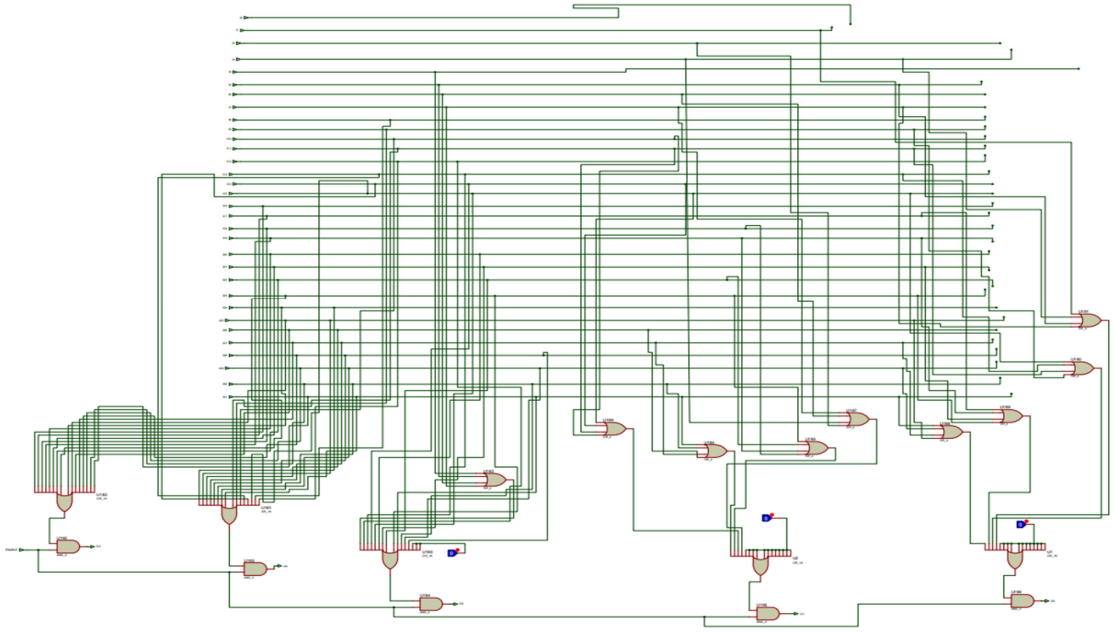


Encoder:

After our display purpose is done, we encode the characters again to put them in memory. This encoding significantly reduces the number of register flip-flops we use to store the password.



Sub circuit for encoder:



32 to 5 Encoder Truth table:

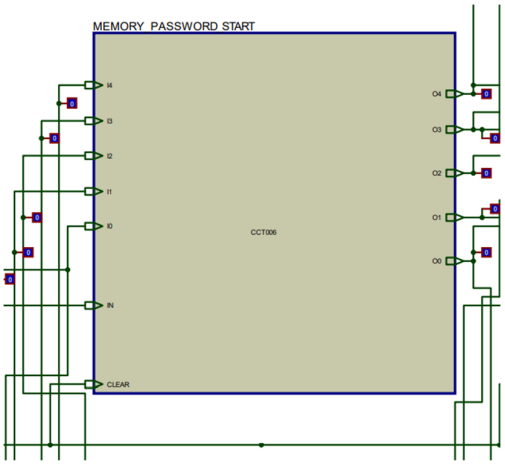
Since the encoder has 32 inputs it is not feasible to draw the truth table for the 32-to-5-bit encoder therefore we showed the conversion in decimal representation which will be the out of the encoder after the 32bit are encoded to 5 bits.

Input (Decimal Equivalent having four-bit representation)	O4	O3	O2	O1	O0
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0
17	1	0	0	0	1
18	1	0	0	1	0

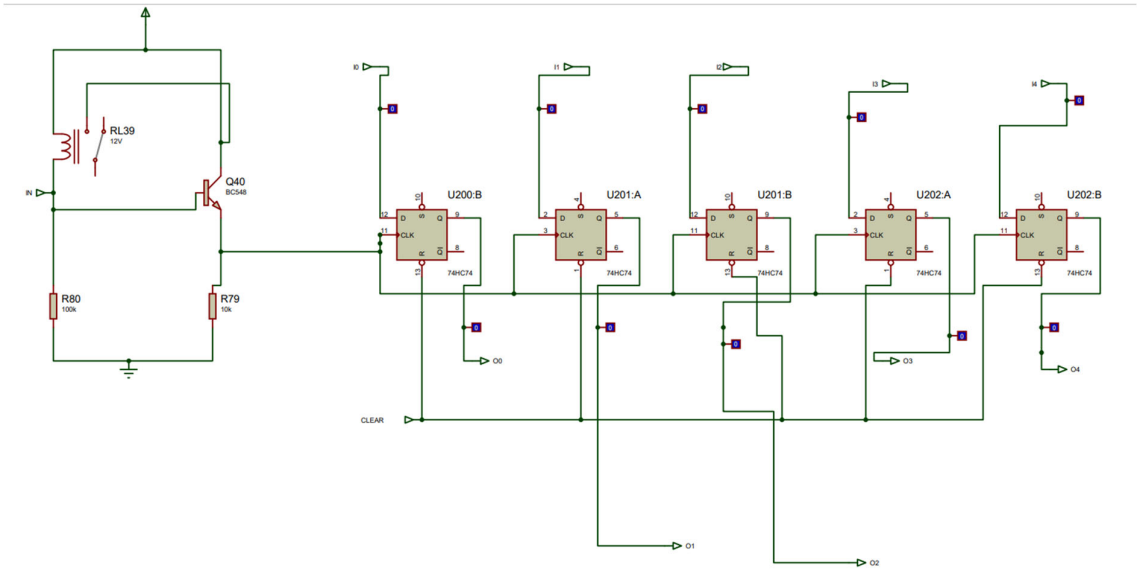
19	1	0	0	1	1
20	1	0	1	0	0
21	1	0	1	0	1
22	1	0	1	1	0
23	1	0	1	1	1
24	1	1	0	0	0
25	1	1	0	0	1
26	1	1	0	1	0
27	1	1	0	1	1
28	1	1	1	0	0
29	1	1	1	0	1
30	1	1	1	1	0
31	1	1	1	1	1

Register [Memory Cells]:

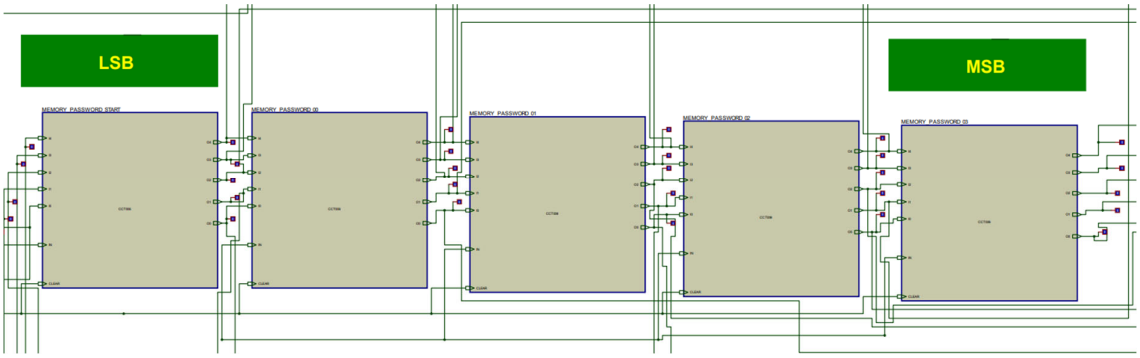
Registers are used to store binary data. Each flip flop can contain a binary bit. Here, as we have to save our passwords also, we have to take user inputs and store them hence we need memory elements. Our PIPO registers are doing exactly that.



Sub-Circuit Built Using PIPO registers:



As we are to store up to 5 digits password hence, we use 5 of these PIPO registers working in a set by shift mechanic



We used the same MEMORY BLOCKS for saving purposes for both saving the password and taking user input.

Comparator:

As our MEMORY blocks are done, we are set to compare whether the user input matches the previously saved password or not.
Hence, we use our comparator blocks to check that. This consists of an X-nor gate.
As we know for 2 input X-NOR gates we get an output only when two inputs are the same. In other words when the user input is a correct password.

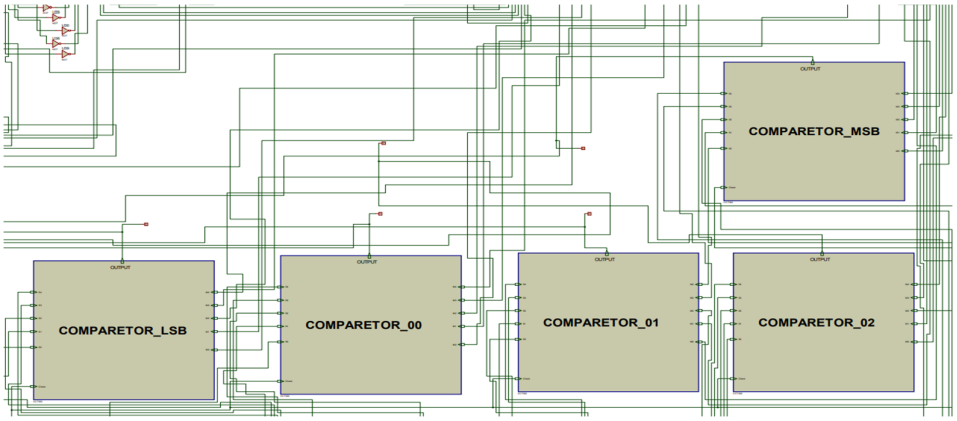
X-NOR gate truth table:

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

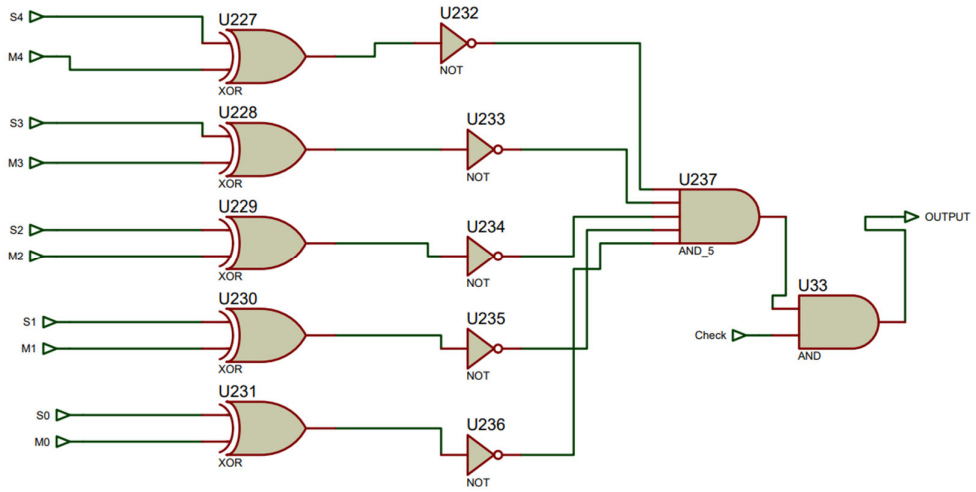
As we can see the output is high only when A & B are equal. In other words, we get an output when user inputs match the password. And hence our system unlocks.

COMPARATOR BLOCKS:

5 blocks to compare up to five digits



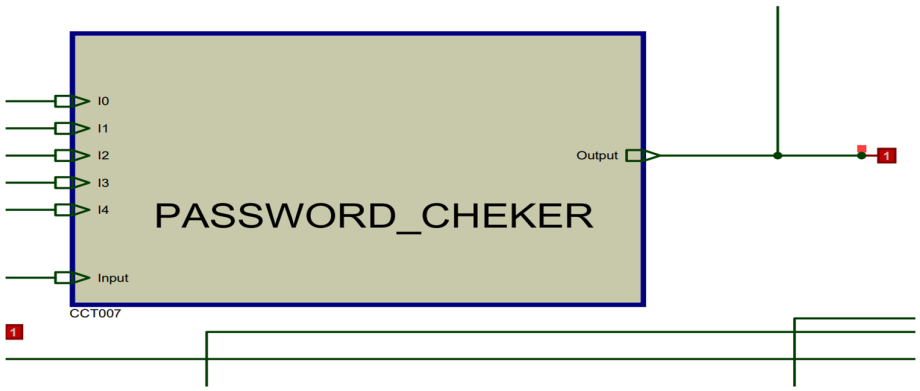
Sub circuit:



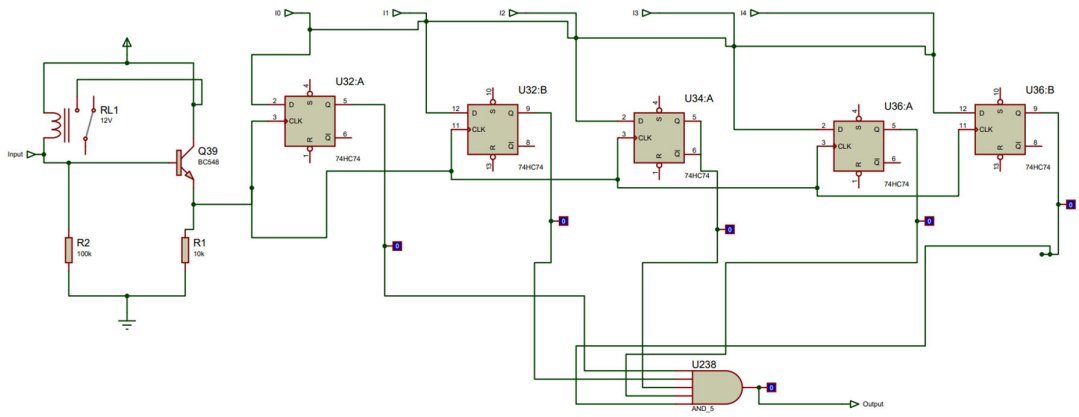
The Child Sheet shows every comparator block compares all Five Bits. As our Letters are encoded in 5 bits.

<u>Password Checker:</u>

Our password checker block uses the Output of all five comparator blocks. Here we use an and gate. Hence, when all the Comparator out is high, we get the feedback that our password is correct.
However, here we used PIPO registers too. This is because we want the password checker to check only when a clock pulse is given. That means when the user submits the password (clock pulse) the password checker starts doing its operation and shifts the comparator outputs to password checking and gate.

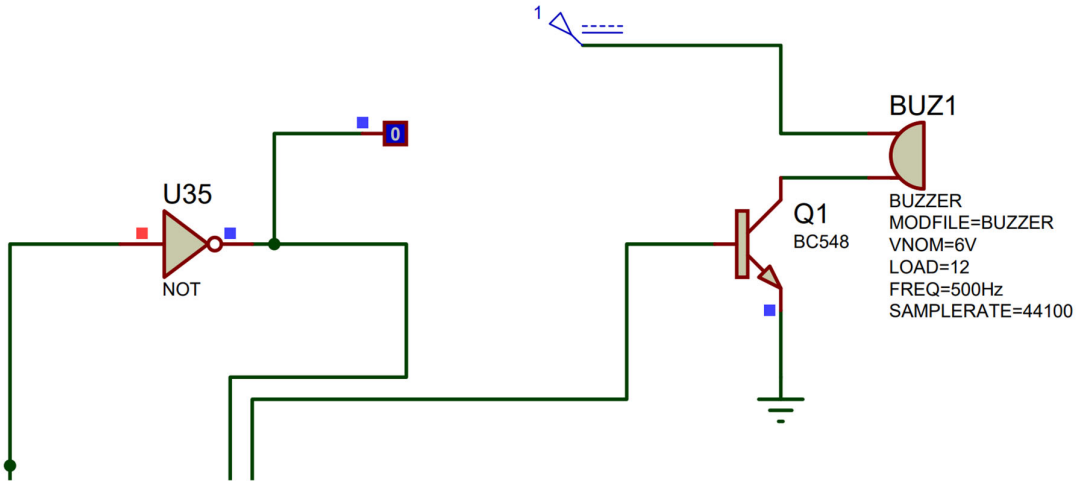


Sub Circuit:



Buzzer:

The buzzer works as Haptic Feedback in our circuit and buzzes for wrong attempts.



The buzzer buzzes every time the circuit is complete (Switch to attempt is pressed). However, it is also connected to the output of the password checker. Hence, when the password is correct it doesn't buzz

Attempt counter:

A good security system cannot let anyone attempt the wrong password as many times as they want. So, we use D flip flops to count the number of wrong attempts and deactivate the system when the wrong attempt limit exceeds.

D (DATA) Flip Flop Truth Table [When the clock is high]:

D (DATA)	Qn+1	Action
0	0	Memory
1	1	Memory

D (DATA) flip flop characteristic table:

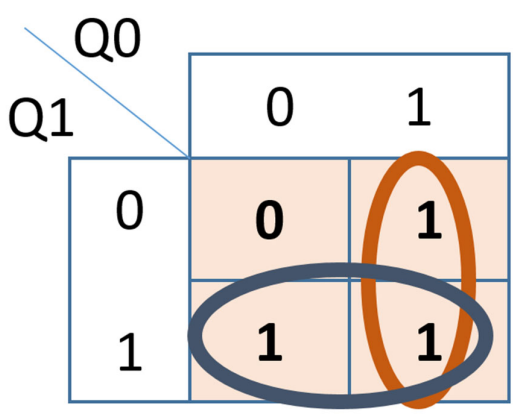
Qn	D (Data)	Qn+1
0	0	0
0	1	1
1	0	0
1	1	1

Circuit characteristic table:

Present State		FF input		Next stage	
Q1	Qo	D1 Q1 Q1+	Do Q0 Q0+	Q1+	Q0+
0	0	0	1	0	1
0	1	1	0	1	0
1	0	1	0	1	1
1	1	1	1	1	1

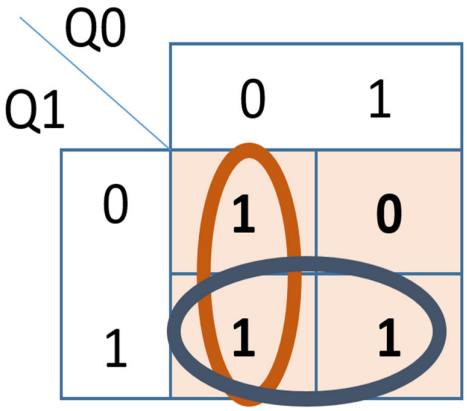
K map for our counter:

For Flip Flop D_1



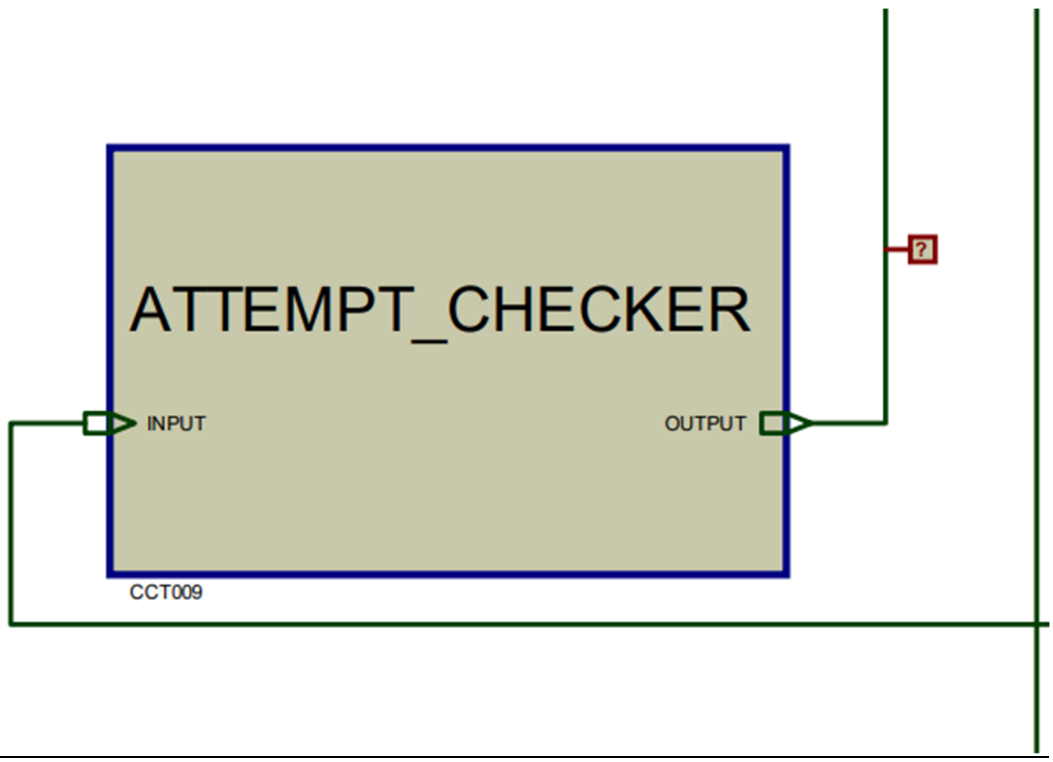
So, $D_1 = Q_0 + Q_1$

For Flip flop D_0

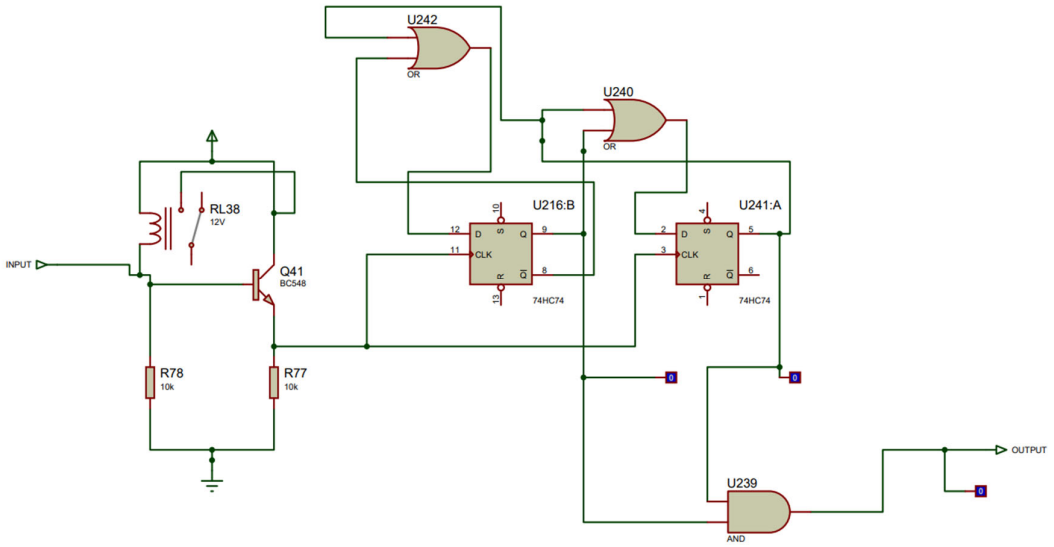


Hence, $D_0 = Q_0' + Q_1$

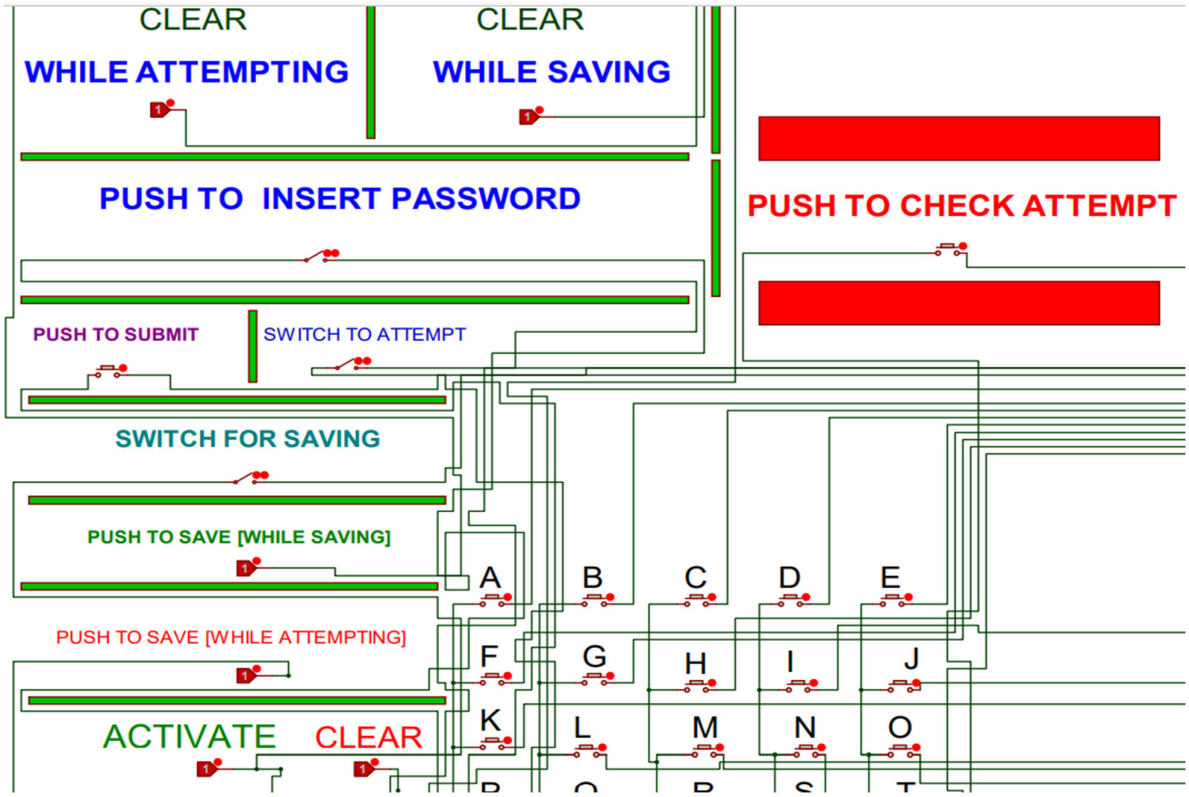
Attempt checker block:



Sub circuit:



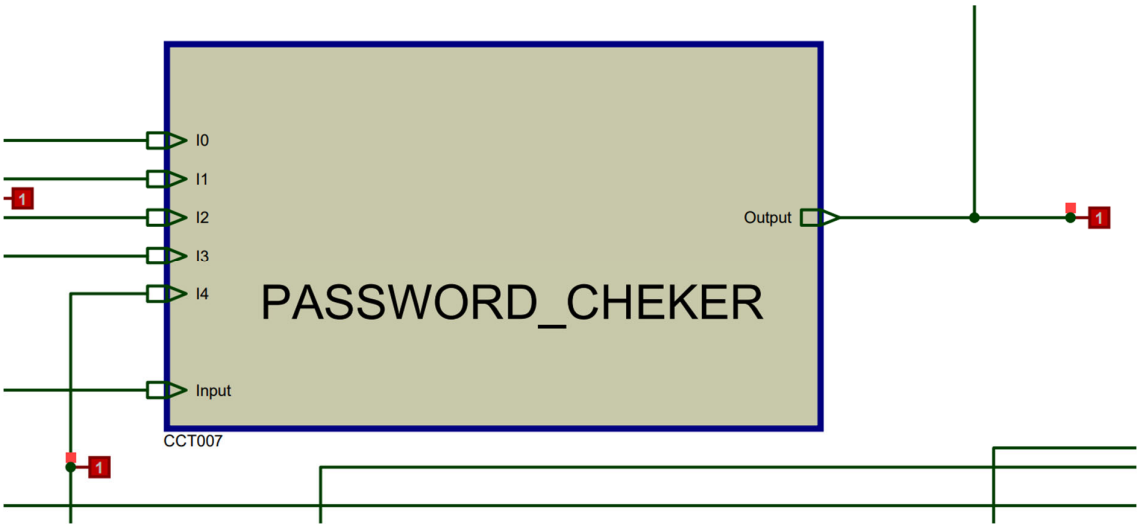
Activation Switches:

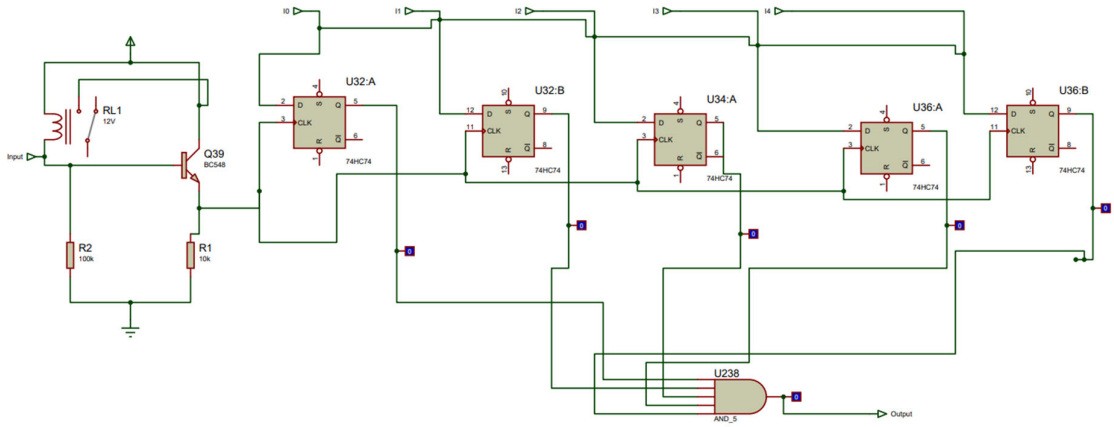


Here, as we can see there are a bunch of different switches and we have a couple more to it. Instead of explaining one by one on how they work let's focus on how to make them work. First let's talk about how to activate the system. The activation switch works something like a power on button as without this the whole system will shut down. The clear button next to it is the clear switch for keyboards and clears the previous input. The push to save buttons are basically clocks for the memory cell clocks. One for Saved password cells and another for input password cells. Switch for saving is actually the activator of the encoder that encodes the characters for storing purpose. If the Push to insert button is on it activates the comparator. Push to submit is nothing but the clock of the password checker. Switch to attempt completes the buzzer circuit. And push to check count the attempt numbers.

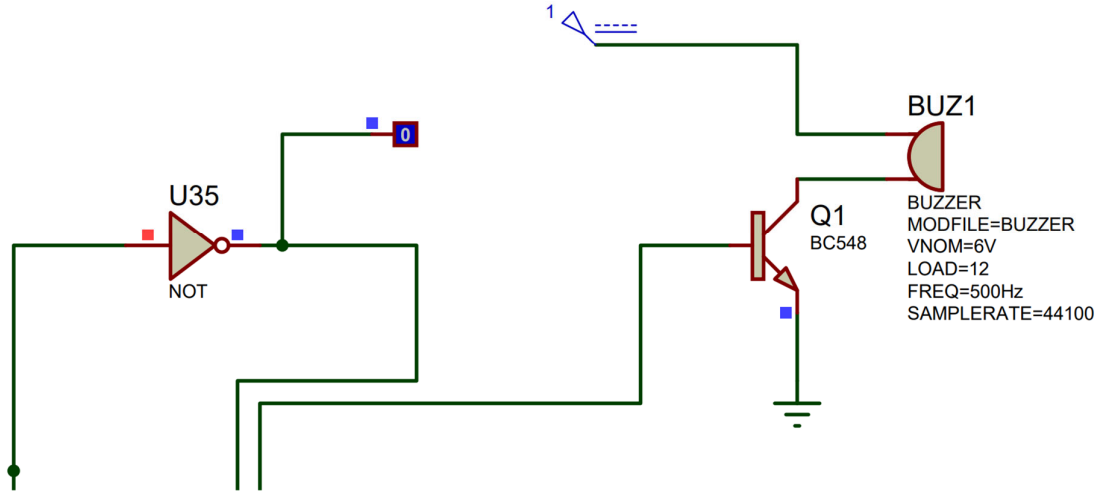
Results and Discussion:

Here our password systems work just fine. First let's observe whether our memory section works or not. As I set the password as 'DB' the memory cells stored just that. D (00100) & B (00010). Now if I open the password saving switch and go to insert mode and put the same password and attempt to check then all the comparator outputs are shown high. (Rest of the three digits being 00000 matches too) And as all the comparator output is high the password checker gives positive feedback too.

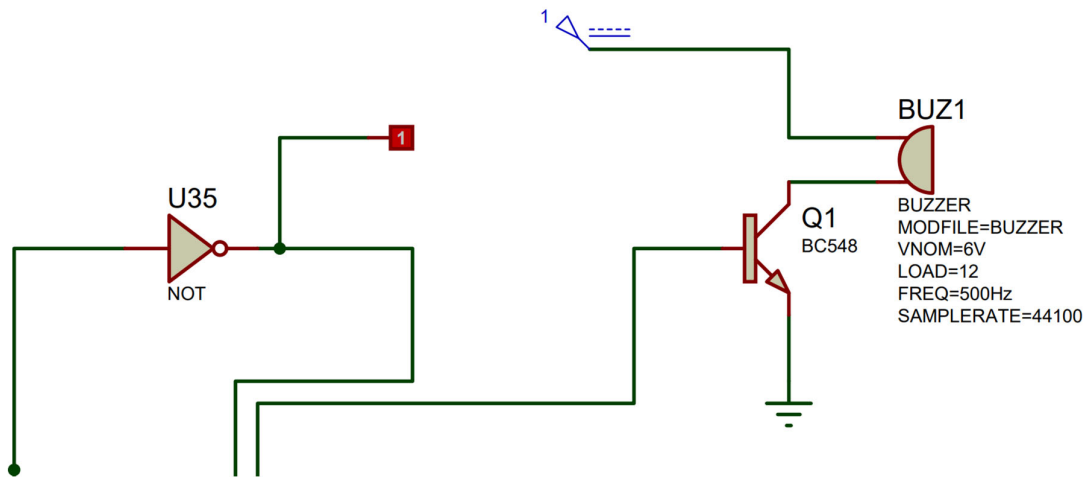




Now when the password is correct the buzzer doesn't buzz as the output of the password checker is inverted.



However, when the password is incorrect the buzzer buzzes,

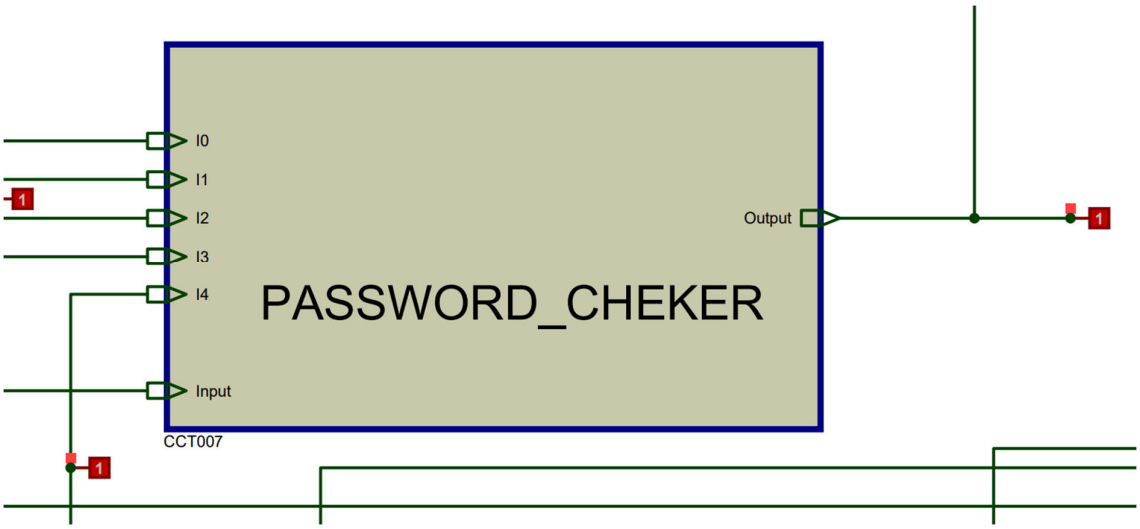


Also, other switches work just fine. Clear buttons (While attempting or saving) clears all memory blocks in that specific area. This really helps as users might mistakenly put wrong inputs and feel the need to reset it.
Again, our activation key works as a master key and without that our system won't run.

Project Validation:

Our project was all about designing digital password protected systems. So, the key goal here is to make sure it unlocks when the user inputs the correct password. And when the input is wrong it raises an alarm to notify the owner or the alarm may work as haptic feedback too.

In our system when the user puts the correct password. The password checker provides high outs. This output may be connected to a lock & when the output is high it will unlock the locker.



This shows the basic working mechanic of our system is working just fine.

Now when the password is incorrect then the system will buzz means it will raise an alarm. When the password is incorrect the password checker shows the output is wrong.

Alternative Design:

As the time did not permit us, we could not provide the optimal design in our design project. However, simple tweaks can make our design more efficient and optimized. We consider these tweaks being our alternate design which will basically enhance our circuit.

These tweaks include:

6 bits encoding:

As our characters are encoded up to 5 bits hence there is an overlap between Alphabetical & Numerical keyboards. Therefore, we can use both to input passwords or if we do so there would exist an alternate way to unlock the system.

But 6 bits encoding of the characters will give us enough space to include both numbers and characters. We could use Ascii codes too for encoding purposes.

Less User Inputs:

Some of the buttons we use can be merged together to make the work of the user less tedious. For example, we have to push the submit button to input our password as well as we have to clear the keyboards. These keys can be merged together as well.

Impact:

In this 21st century, digital theft, security scams and other unethical problems have increased. Such issues have motivated our team to build a digital password security system as Passwords provide the first line of defense against unauthorized access to your asset, computer and personal information. Though cyber-criminals are increasingly using automated password cracking tools to identify passwords in a matter of seconds and also weak passwords can lead to a data breach, a strong password can protect our system from hackers, malicious software and also protect from accessing our personal information. By implementing our idea, we will be able to ensure safety and prevention of illegal activities by averting identity theft, potential fraud and saving users from potential intruders.

Future Work:

Although the security system we have designed is successful in checking whether the correct password has been given as input and we have successfully added alphabets and numbers, there are still possible chances of further improvement which we can implement in near future. The system we implemented can only use the alphabet or number at once. Also, we are using same memory shell for alphabet and numerical number. For instance, we can further modify our system

that can take both the alphabet and number together by making a complex design which will have different binary numbers for alphabets and decimal numbers and also can add an option for special characters (&, %, $, #, @ etc.). Furthermore, instead of so many switches, we can further implement only one button to activate and clear the system and only one switch for saving and attempting purpose. We could have used fewer OR-gates and logic probes in the sub circuits of the digital password security system.

Conclusion:

Our password system deals with a system that is able to check a 5-digit decimal number or alphabet. Our system consists of activating, clearing, saving, attempting and other switches to start the system, setting and saving our memory password. Our alphabets are connected with 2 decoders and its output is connected to 32 to 5 encoder and decimal numbers are connected with 4 to 16 decoder which is converting our password into binary numbers. We have also built a comparator and password checker to compare our memory password and user input password. We have also built an attempt checker which counts the number of attempts left to match the required password before the system shuts down. In our system only 3 attempts are possible to enter the correct password. All our sub circuits are made of D flip flop working as PIPO registers.

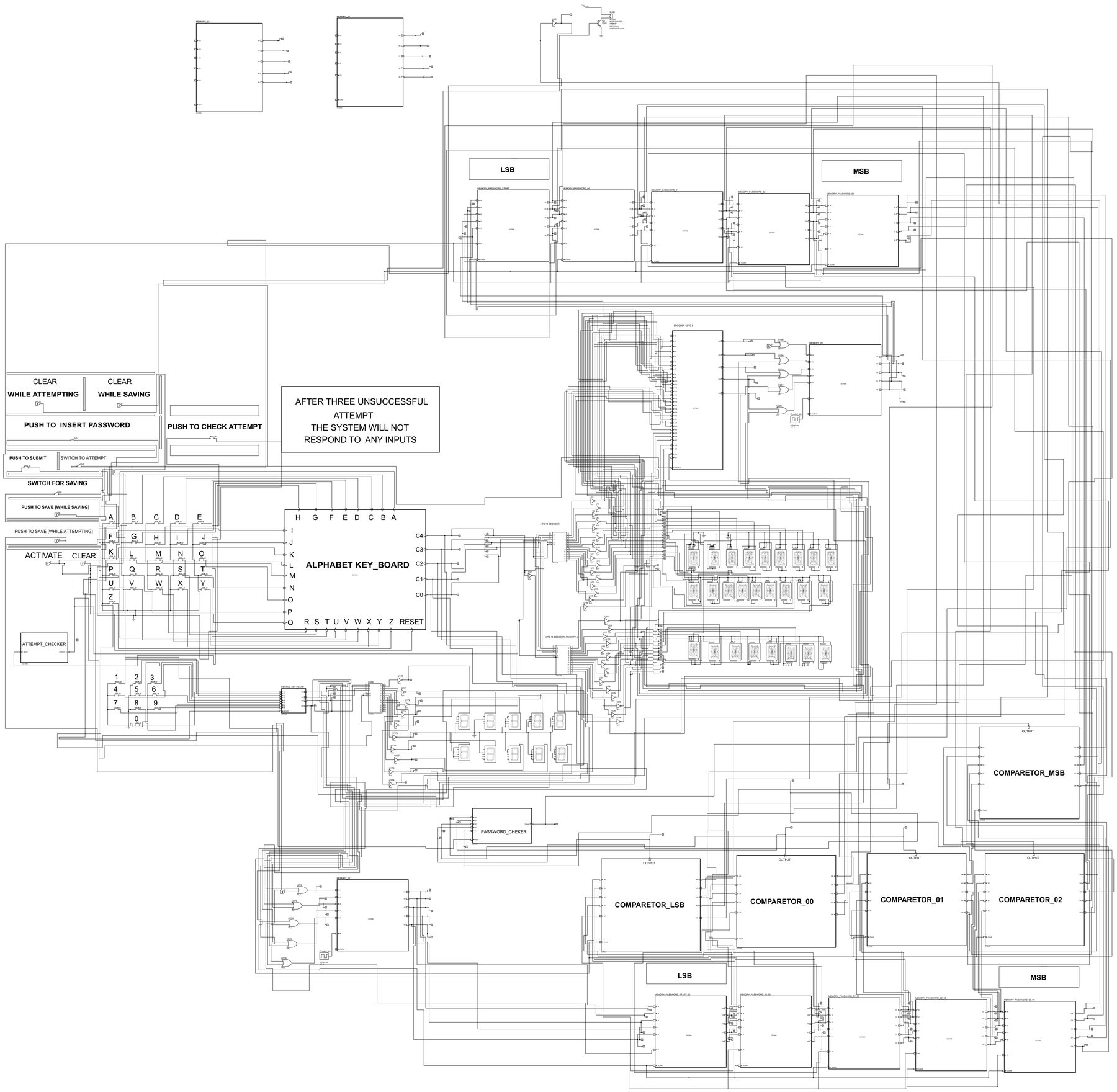
Reference:

1. https://www.youtube.com/watch?v=M0mx8S05v60&list=PLBlnK6fEyqRjMH3mWf6kwqiTbT798eAOm
2. https://www.researchgate.net/
3. https://www.circuitstoday.com/proteus-software-introduction
4. https://componentsearchengine.com/library/proteus?gclid=CjwKCAjwhOyJBhA4EiwAEcJdcTMr4kbKMpVVFJZuZrHOXzIN9SEUTI13cNnzuNqb2p1h3gqhCi8hphoCzBcQAvD_BwE
5. https://bd.izito.ws/ws?q=proteus%20library&asid=iz_bd_gc1_01&mt=b&nw=g&de=c&ap=&ac=17228&cid=13261636033&aid=120644392497&locale=en_BD&gclid=Cjw

KCAjwhOyJBhA4EiwAEcJdcYRnqElPL-3jDUGneW-19Zh5PG9aO1Z0ebe-gQmZe_ZtgV2ptDeb4BoCM5YQAvD_BwE

Project files [drive link]:

https://drive.google.com/drive/folders/1OowGobyuhhZCYoKwzzojj1tRbR93isWs?usp=sharing


LSB
MSB
CLEAR
WHILE ATTEMPTING
CLEAR
WHILE SAVING
PUSH TO INSERT PASSWORD
PUSH TO CHECK ATTEMPT
AFTER THREE UNSUCCESSFUL ATTEMPT
THE SYSTEM WILL NOT RESPOND TO ANY INPUTS
PUSH TO SUBMIT
SWITCH TO ATTEMPT
SWITCH FOR SAVING
PUSH TO SAVE [WHILE SAVING]
PUSH TO SAVE [WHILE ATTEMPTING]
ACTIVATE
CLEAR
ALPHABET KEY_BOARD
ATTEMPT_CHECKER
PASSWORD_CHEKER
COMPARETOR_MSB
COMPARETOR_LSB
COMPARETOR_00
COMPARETOR_01
COMPARETOR_02
LSB
MSB