North South University

Department of Electrical & Computer Engineering

Project Report

CSE331

Section: 09

Group: 05

Submission Date: 01.01.2023

Submitted to: Dr. Dihan Md. Nuruddin Hasan (DMH)

Group Member

Abraham Kaikobad	2013983642
Parsa Alam	2011902642
Rian Ahmed	2013650042
Mahir Ayaan Begh Jeet	2013778642

Table of Contents

Abstract

We were given an encryption table for this project and have to construct it using a microcontroller. We are also instructed to set up the inputs for high and low circumstances using a single pole and double-throw switch. In addition, we are obliged to employ LEDs to symbolize the table's matching output.

The given encryption table:

Input				**Output**			
I_3	I_2	I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1	0	1
1	0	0	0	1	1	0	0
0	1	0	0	1	1	1	0
1	1	0	0	0	0	0	1
0	0	1	0	0	1	1	1
1	0	1	0	0	0	1	0
O	1	1	0	0	1	1	0
1	1	1	0	0	1	1	0
0	0	0	1	1	1	0	0
1	0	0	1	0	0	0	0
0	1	0	1	0	1	1	0
1	1	0	1	1	0	1	1
0	0	1	1	0	0	0	0
1	0	1	1	0	0	1	1
0	1	1	1	1	0	1	1
1	1	1	1	0	0	1	1

General Description:

2.1 Arduino UNO

The ideal board to start with when learning about electronics and programming is the Arduino UNO. The most reliable board to start with if this is someone's first time fiddling with the platform is the UNO.

having fun with. The UNO is the Arduino family's most popular and well-documented board.

An ATmega328P-based microcontroller board is the Arduino UNO. Six analog inputs, a 16 MHz ceramic resonator, 14 digital input/output pins (six of which can be used as PWM outputs), a USB port, a power jack, an ICSP header, and a reset button are all included. It comes with everything required to support the microcontroller; to get started, just use a USB cable to connect it to a computer, or an AC-to-DC adapter or battery to power it. Without too much concern, you can experiment with your UNO; if something goes wrong, you can replace the chip for a few dollars and start over.

Replaceable chip: The classic high-performance, low-power AVR microcontroller.

ATmega328P: The ATmega328P can easily be replaced, as it is not soldered to the board.

EEPROM: The ATmega328P also features 1kb of EEPROM, a memory that is not erased when powered off.

BatteryConnector:

The Arduino UNO features a barrel plug connector that works great with a

standard 9V battery

It doesn't connect to a computer using a conventional USB port. It instead includes a type B Micro USB. Depending on the Atmega board, flash memory can be 16 or 32 KB in size.

Atmega168 has 16 KB of flash memory, while Atmega328 has 32 KB of flash memory. Code is stored on flash memory. A bootloader uses 2KB of the whole flash memory's memory. The Arduino Nano comes with 2KB of SRAM memory. A 1KB EEPROM serves as its memory.

2.2 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. The software used for writing, compiling & uploading code to Arduino boards is called Arduino IDE (Integrated Development Environment). It is a cross-platform software that is available for every Operating System like Windows, Linux, macOS. This software can be used with any Arduino board.

2.3 Proteus 8 PRO

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards. We simulate our circuit in Proteus 8 Pro using Arduino Uno, resistors, single pole double throw switches, LED lights.

2.4 Logisim

Logisim is a logic simulator that permits circuits to be designed and simulated using a graphical user interface.

It is free and open-source software compatible with multiple platforms like Windows, macOS, and Linux. The ease of Logisim comes from its interactive graphical user interface developed with the Java Swing GUI library. Logisim enables users to create circuits of any scale, including sub-circuits as well as complex structures. Color-coded wires impart clarity to the representation. Logisim lets a user simulate his logical circuits and edit circuits while the simulation is running. Therefore, a user can immediately rectify flaws without the need for separate edits and loading.

3.Equipment's

We use a variety of software and hardware to finish the projects, those are

1. Logisim

2. Arduino Uno

3. Proteus 8 Pro

4. 4-bit SPDT switch

5. 4 x LED light

6. Jumper Wire

7. Breadboard

8. 4 x 10k ohm resistor

4.Method of Derivation

We used the given table of Group-5 and we built it into a truth table for inputs and outputs. Then we used the concept of SOP to produce the KMAPs below and then found the logical expression for each of the outputs using KMAP.

4.1 Truth Table:

Input				Output			
I_3	I_2	I_1	I_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1	0	1
1	0	0	0	1	1	0	0
0	1	0	0	1	1	1	0
1	1	0	0	0	0	0	1
0	0	1	0	0	1	1	1
1	0	1	0	0	0	1	0
O	1	1	0	0	1	1	0
1	1	1	0	0	1	1	0
0	0	0	1	1	1	0	0
1	0	0	1	0	0	0	0
0	1	0	1	0	1	1	0
1	1	0	1	1	0	1	1
0	0	1	1	0	0	0	0
1	0	1	1	0	0	1	1
0	1	1	1	1	0	1	1
1	1	1	1	0	0	1	1

4.2 Derived Result:

We use the K-Map to derive the expression for the output.

KMAP:

Below are the K-maps for the output of our encryption table, where A, B, C, and D is the input.

After using the KMAP we find the following expressions:

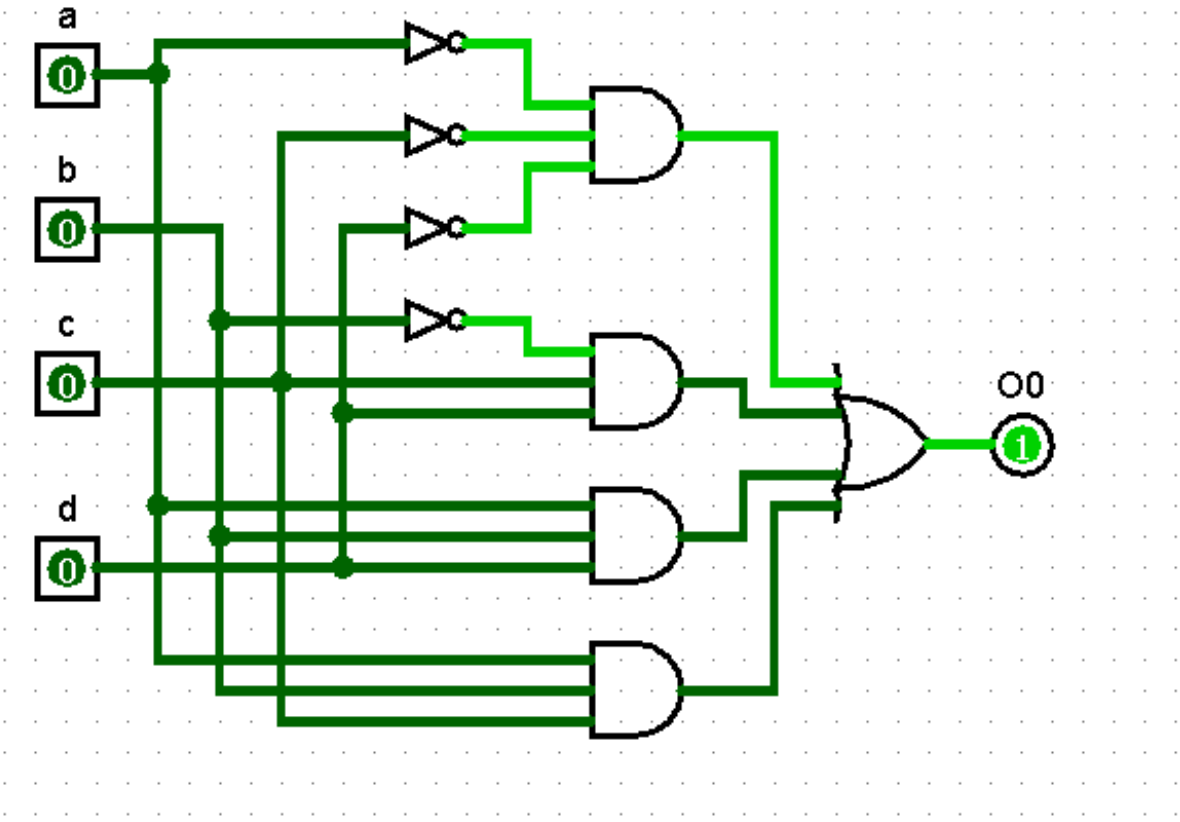
O_0= A'C'D' + B'CD + ABD + ABC

O_1 = CD' + A'B + AC + BD

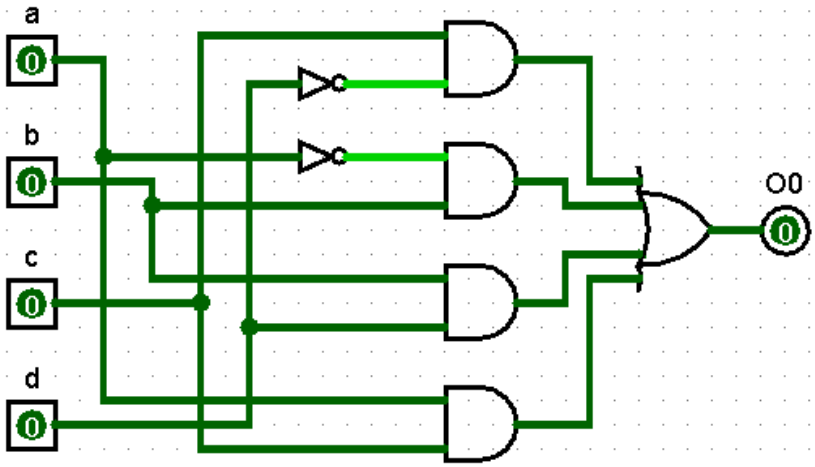
O_2 = A'B'C' + A'D' + A'BC + B'D'

O_3= A'B'C'D + A'B'CD' + AB'C'D' + AB'CD + ABCD'

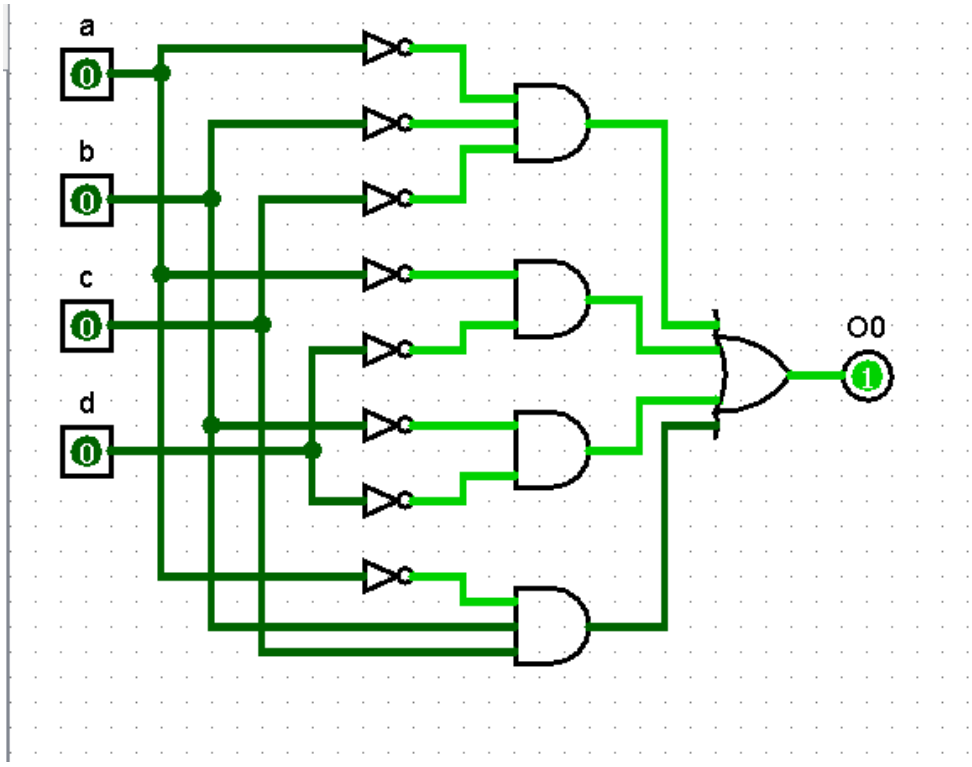
Logical Circuit Diagram of O_0



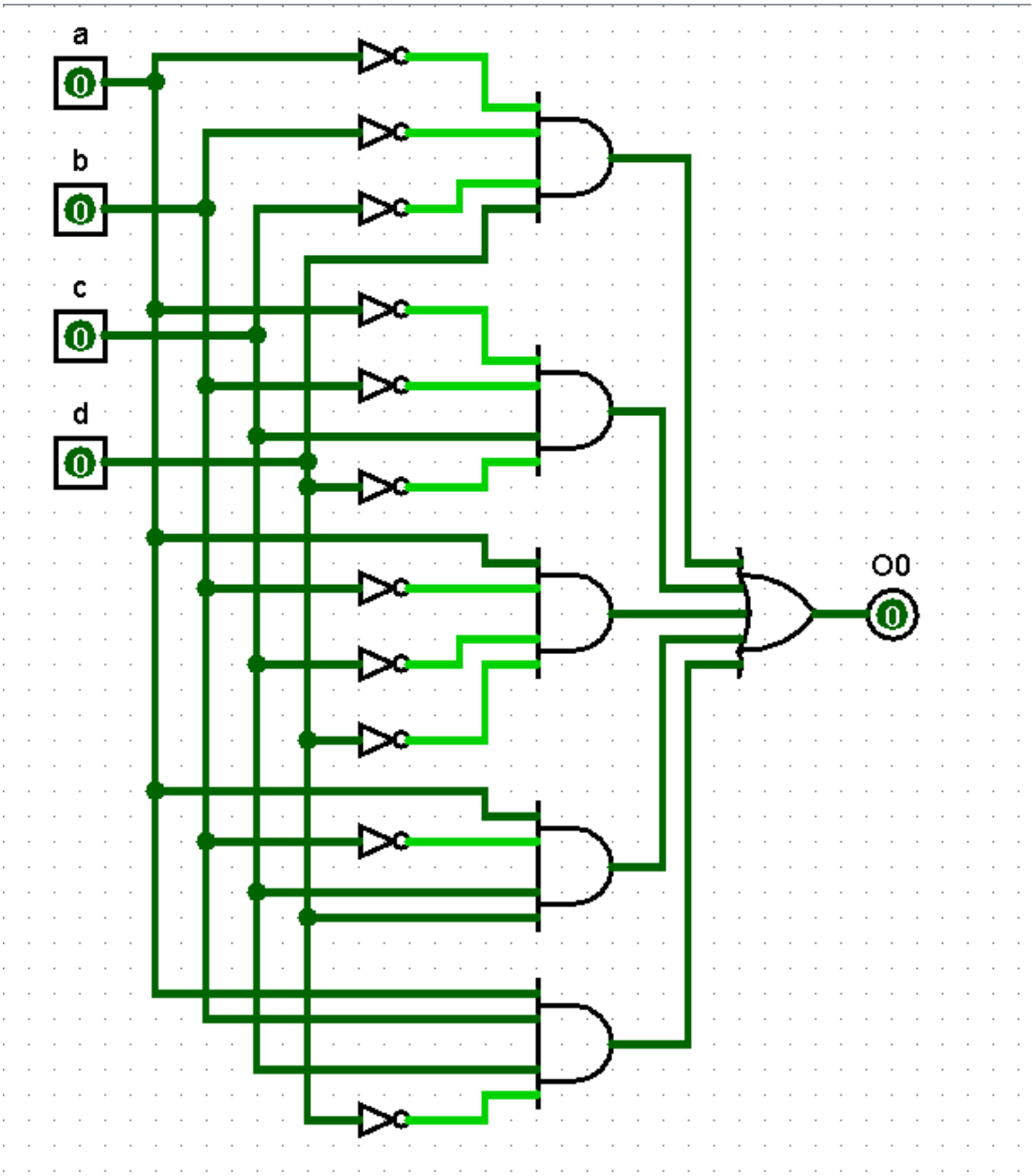
Logical Circuit Diagram of O1



Logical Circuit Diagram of O2

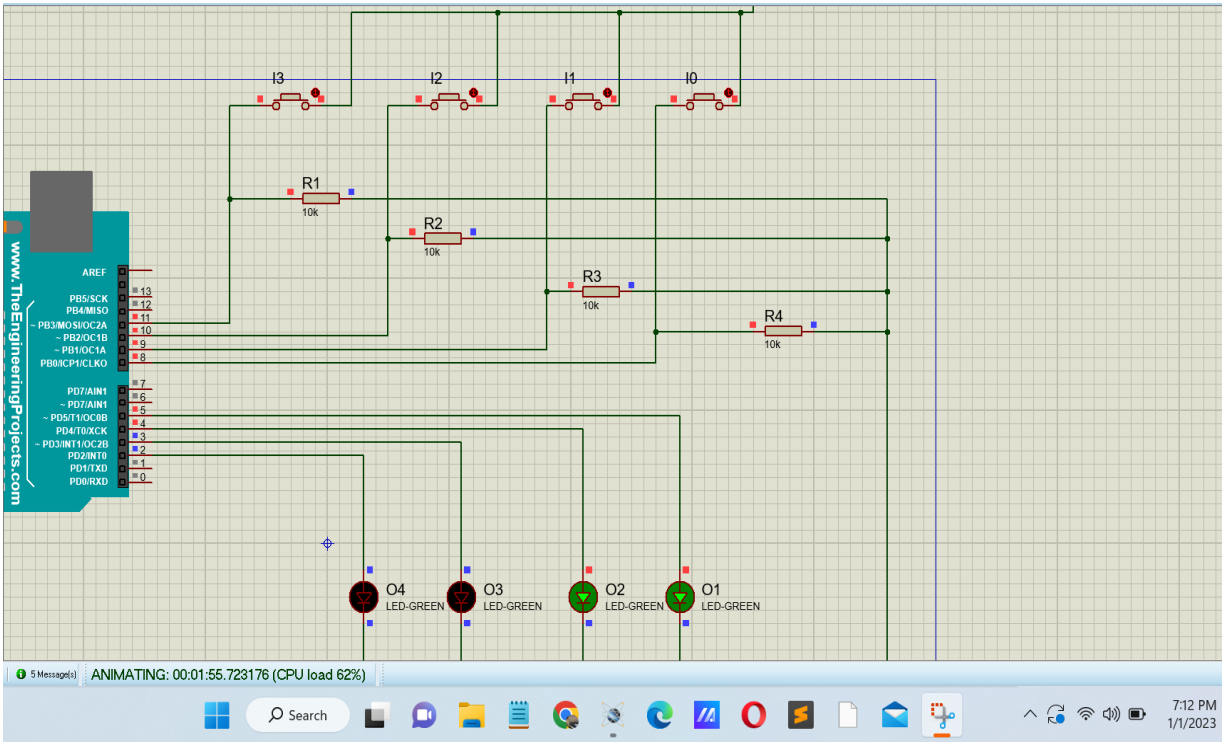


Logical Circuit Diagram of O3



Circuit Diagram with Values of Electrical Components

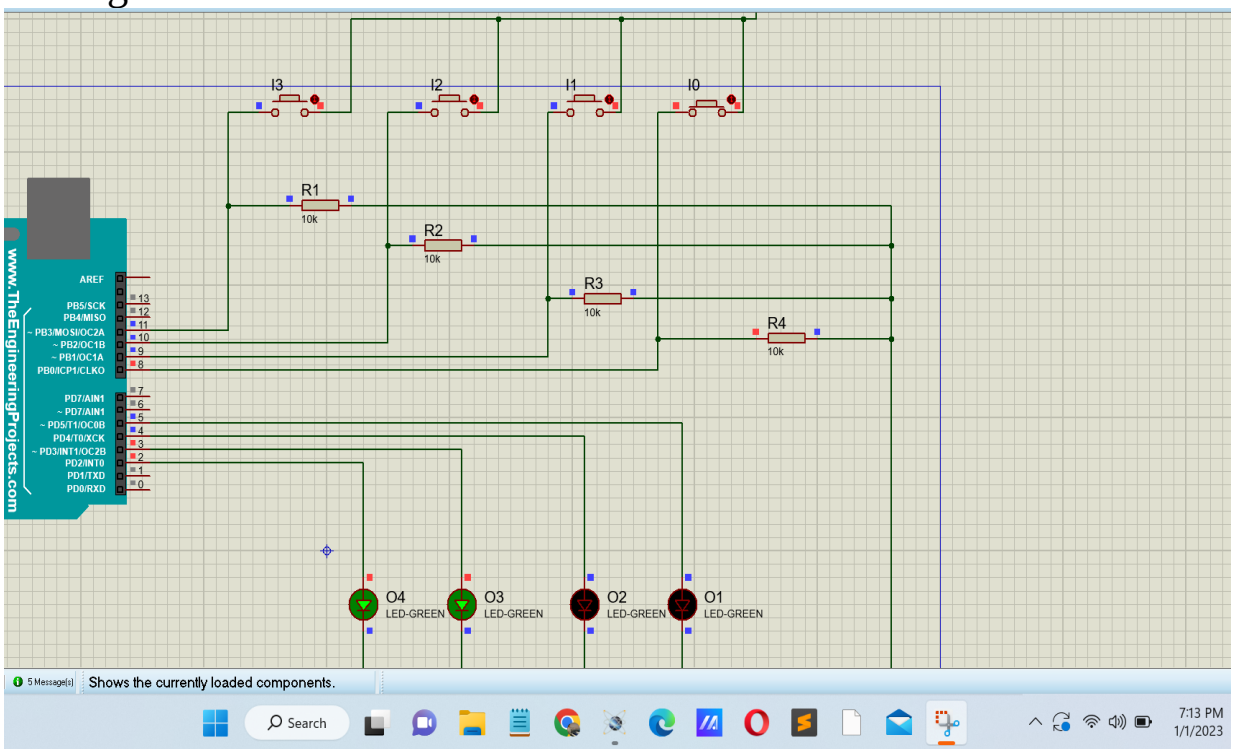
5.1 Figure 1:



Here we can see the circuit in Proteus 8,

for the inputs are **I_3=1, I_2=1, I_1=1, I_0=1** the LED outputs are **O_4=0, O_3=0, O_2=1, O_1=1**.

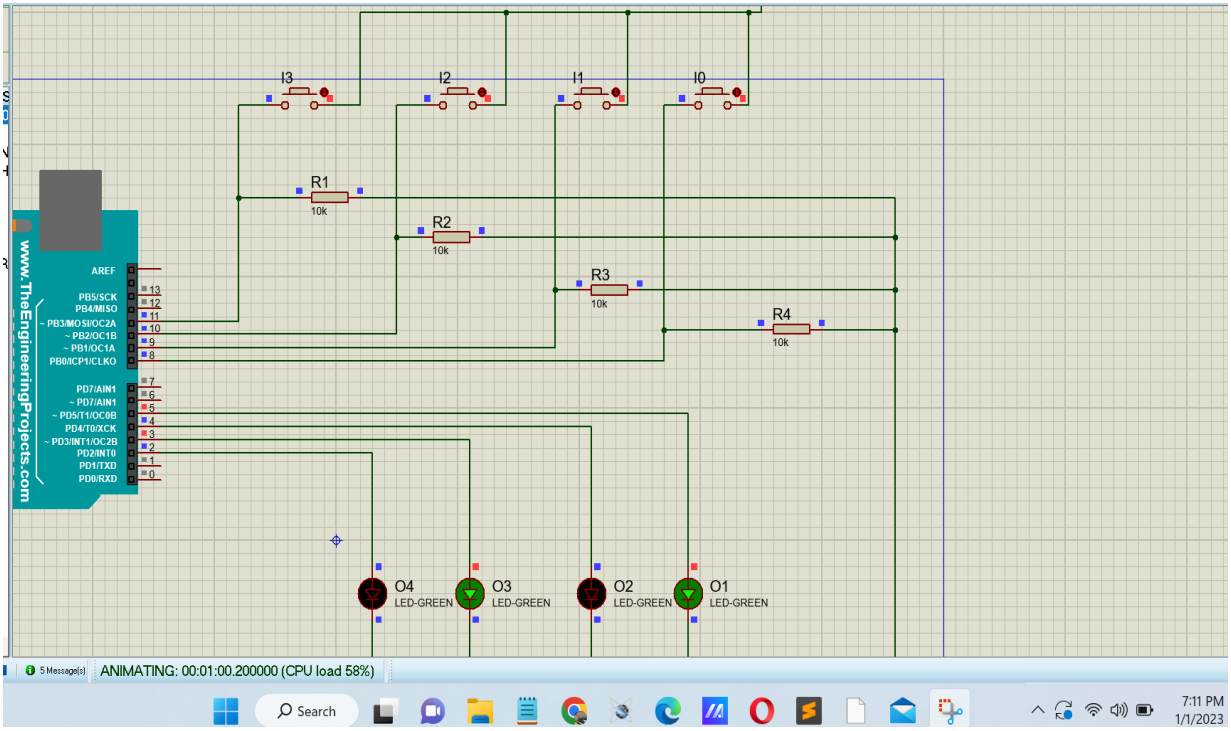
5.2 Figure 2:



Here we can see the circuit in Proteus 8,

for the inputs are **I_3=0, I_2=0 I_1=0, I_0=1** the LED outputs are **O_4=1, O_3=1, O_2=0, O_1=0**.

5.3 Figure 3:



Here we can see the circuit in Proteus 8,

for the inputs are **I_3=0, I_2=0, I_1=0, I_0=0** the LED outputs are **O_3=0, O_3=1, O_2=0, O_0=1**.

Circuit Operation Principles:

We get our output functions using the K-map and the equations are,

$\mathbf{O_0}$= I_3'I_2'I_0' + I_2'I_1' + I_3 I_2' I_0

$\mathbf{O_1}$ = I_3' I_1 + I_1I_0 + I_3I_2 + I_3 I_2' I_1' + I_3 I_1' I_0'

$\mathbf{O_2}$ = I_3'I_2'I_1'I_0 + $I_2I_1I_0$' + I_3I_1+ I_3I_2

$\mathbf{O_3}$= I_3'I_2'I_1 + I_3'I_2I_1' + $I_3I_1I_0$' +$I_3I_2I_0$

The next step is to build the circuit using logic gates and implement them on Logisim. Then we used Proteus 8 pro to build the hardware circuit diagram, the components used in proteus are **Arduino Uno**, this is the microcontroller we used in the simulation.

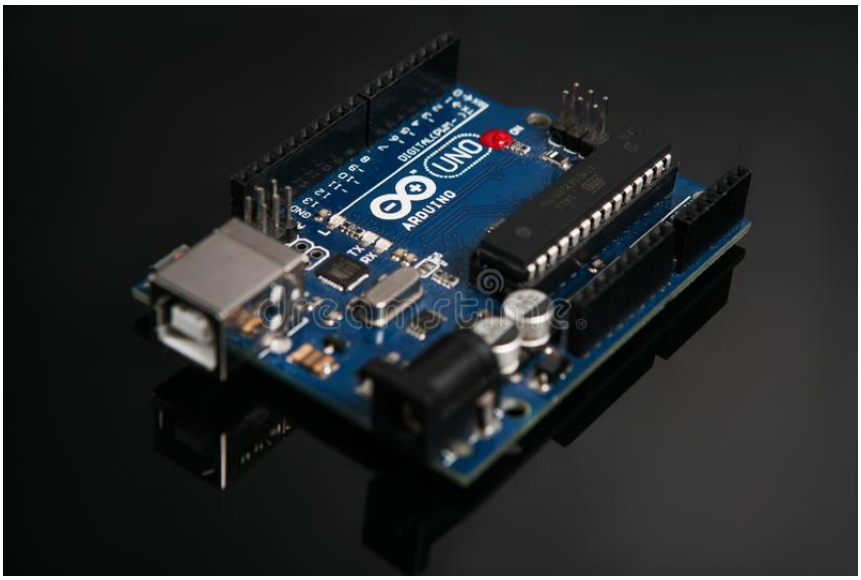


Figure: Arduino UNO

Push Switch:

A push button switch is a mechanical device that controls an electrical circuit by physically pressing a button to activate an internal switching mechanism. They come in various shapes, sizes, and configurations depending on the demands of the design.

We utilized a push switch with two pins. The left pin is connected to the other buttons, while the first pin is connected to the Arduino's output pin. Each switch is interconnected with the others as well as independently attached to the relevant Arduino input pins 2,3 and 4.

Figure: Switch

Resistors:

We used 4 10K -ohm resistors to connect with the switches and connect with the LED outputs.

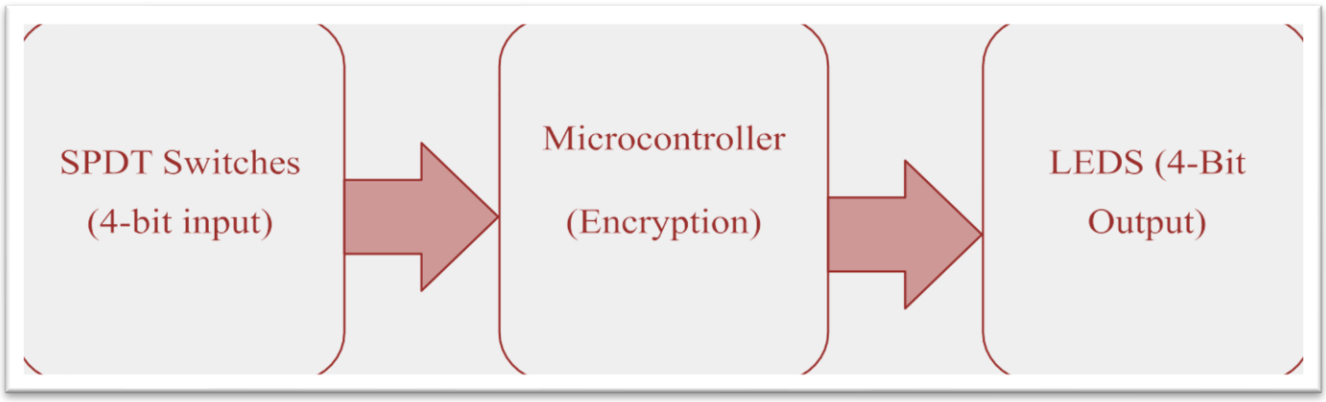
Figure: Resistor

LED'S:

We connected 4 LED outputs O_0, O_1, O_2, O_3 to Arduino Uno pin no D11, D10, D9, D8, respectively. After building the circuit we uploaded the Arduino code to the Nano and simulate the project. We uploaded a video on Drive to demonstrate the Proteus simulation of the circuit.

Figure: LEDs

Flow Diagram:



Arduino Program

Arduino CODE:

```
int in_0 = 8;
int in_1 = 9;
int in_2 = 10;
int in_3 = 11;
int out_0 = 5;
int out_1 = 4;
int out_2 = 3;
int out_3 = 2;
int A = 0;
int B = 0;
int C = 0;
int D = 0;

void setup() {
pinMode(in_0, INPUT);
pinMode(in_1, INPUT);
pinMode(in_2, INPUT);
pinMode(in_3, INPUT);
pinMode(out_0, OUTPUT);
pinMode(out_1, OUTPUT);
pinMode(out_2, OUTPUT);
pinMode(out_3, OUTPUT);
}

void loop() {
```

```
A = digitalRead(in_0);
B = digitalRead(in_1);
C = digitalRead(in_2);
D = digitalRead(in_3);

if ((!A && !C && !D ) || (!B && C && D) || (A && B && D) || (A && B && C)){
digitalWrite(out_0, HIGH);
} else {
digitalWrite(out_0, LOW);
}

if ((C && !D) || (!A && B) || (A && C) || (B && D)){
digitalWrite(out_1, HIGH);
} else {
digitalWrite(out_1, LOW);
}

if ((!A && !B && !C ) || ( !A && !D) || (!A && B && C) || (!B && !D)){
digitalWrite(out_2, HIGH);
} else {
digitalWrite(out_2, LOW);
}

if ((!A && !B && !C && D) || (!A && !B && C && !D) || (A && !B && !C && !D) || (A &&
!B && C && D) ||( A && B && C && !D)){
digitalWrite(out_3, HIGH);
} else {
digitalWrite(out_3, LOW);
}

}
```

Question and Answer

Question 1: What is the clock frequency of the microcontroller used?

Answer: We used Arduino UNO, and the clock frequency of the microcontroller is 16MHz.

Question 2: What is the data bus width of the microcontroller used?

Answer: The data bus width of Arduino UNO is 8-bit.

Question 3: What is the size of your hex file generated? Attach the hex codes in your report.

Answer: The hex file size is 3.43KB. The hex file is attached below:

```
:100000000C9461000C9473000C9473000C947300B6
:100010000C9473000C9473000C9473000C94730094
:100020000C9473000C9473000C9473000C94730084
:100030000C9473000C9473000C9473000C94730074
:100040000C9426010C9473000C9473000C947300B0
:100050000C9473000C9473000C9473000C94730054
:100060000C9473000C947300000000000240027001F
:100070002A00000000000250028002B0000000000DE
:1000800023002600290004040404040404040202DA
:1000900002020202030303030303030301020408102007
:1000A000408001020408102001020408102000012
:1000B000000800020100000304070000000000027
:1000C000000011241FBECFEFD8E0DEBFCDBF21E07E
:1000D000A0E0B1E001C01D92A131B207E1F70E949A
:1000E00070010C9468020C940000833081F028F4B5
:1000F000813099F08230A9F008958730A9F08830D6
:10010000C9F08430B1F4809180008F7D03C080916C
:1001100080008F7780938000089584B58F7784BDA9
:10012000089584B58F7DFBCF8091B0008F77809349
:10013000B00008958091B0008F7DF9CFCF93DF9309
:10014000282F30E0F901E255FF4F8491F901E6567E
:10015000FF4FD491F901EA57FF4FC491CC23A1F08E
:1001600081110E947500EC2FF0E0EE0FFF1FE458A4
:10017000FF4FA591B491EC91ED2381E090E009F45B
:1001800080E0DF91CF91089580E090E0FACF1F9357
:10019000CF93DF93282F30E0F901E255FF4F849190
:1001A000F901E656FF4FD491F901EA57FF4FC49188
:1001B000CC23A9F0162F81110E947500EC2FF0E0DE
:1001C000EE0FFF1FEE58FF4FA591B4918FB7F89433
:1001D000EC91111108C0D095DE23DC938FBFDF9125
:1001E000CF911F910895DE2BF8CFCF93DF9390E04E
:1001F000FC01E656FF4F24918A579F4FFC018491E2
:100200008823D1F090E0880F991FFC01E859FF4F37
:10021000A591B491FC01EE58FF4FC591D4916111A5
:100220000EC09FB7F8948C91E22FE0958E238C93AB
:100230002881E223E8839FBFDF91CF9108958FB794
```

```
:10024000F894EC91E22BEC938FBFF6CF1F920F92B4
:100250000FB60F9211242F933F938F939F93AF93D9
:10026000BF9380910D0190910E01A0910F01B0916B
:10027000100130910C0123E0230F2D3758F5019622
:10028000A11DB11D20930C0180930D0190930E01CF
:10029000A0930F01B0931001809108019091090182
:1002A000A0910A01B0910B010196A11DB11D80938F
:1002B000080190930901A0930A01B0930B01BF912B
:1002C000AF919F918F913F912F910F900FBE0F9003
:1002D0001F90189526E8230F0296A11DB11DD2CFBD
:1002E000789484B5826084BD84B5816084BD85B511
:1002F000826085BD85B5816085BD80916E0081601D
:1003000080936E0010928100809181008260809 3C2
:10031000810080918100816080938100809180 00C4
:1003200081608093800080 91B10084608093B100EF
:100330008091B00081608093B00080917A008460E9
:1003400080937A0080917A00826080937A00809115
:100350007A00816080937A0080917A00806880932F
:100360007A001092C10060E088E00E94F50060E031
:1003700089E00E94F50060E08AE00E94F50060E0FC
:100380008BE00E94F50061E085E00E94F50061E0ED
:1003900084E00E94F50061E083E00E94F50061E0E6
:1003A00082E00E94F50080E0A82E80E0B82E88E070
:1003B0000E949E006C01909307018093060189E0E2
:1003C0000E949E007C0190930501809304018AE0C5
:1003D0000E949E008C019093030180930201 8BE0A8
:1003E0000E949E00EC01909301018093 0001C114D2
:1003F000D10439F40115110591F5892B81F060E0E4
:100400000FC0E114F10431F401151105C1F3209777
:10041000B1F305C0892B19F40115110581F361E0D1
:1004200085E00E94C70001151105E9F0209719F138
:10043000C114D10401F5E114F104E9F460E01CC039
:10044000E114F10429F142C0EF2829F4012B61F5F0
:10045000CD2B19F42BC0012BD9F760E028C0E11493
:10046000F10469F6D4CFC114D10429F3E114F104E5
:1004700029F3209719F361E084E00E94C700C114BA
:10048000D104F1F6E114F104E1F40115110511F0C4
:100490002097E1F461E083E00E94C700CD28A1F637
:1004A000EF28D9F6012BA1F6CD2BB9F261E082E05D
:1004B0000E94C700A114B10409F479CF0E94000082
:1004C00076CF209739F30115110521F760E0E3CFCE
:0404D000F894FFCFCE
:00000001FF
```

Question 4: Can the project be implemented by using interrupt?

Answer: Yes, an interruption can be used to implement the project. The start and stop operations of the CPU can be controlled using interrupts. Interrupts can be produced by the Arduino UNO's pins 2 and 3. Additionally, there is a feature known as digital PinInterrupt (pin). However, if an interrupt is produced, neither new data nor new output will occur.

Question 5: Is the main routine required to be an infinite loop? Provide an explanation in favor of your answer.

Answer: Yes, an infinite loop must be used for the main routine. We regularly get 4-bit data, which we encrypt. The machine must therefore continue to operate in a loop. After receiving input, the program would get stuck without a loop and stop providing constant feedback. An infinite loop was, therefore, necessary.

Question 6: Is there any difference between level-triggered and edge-triggered operation for the Given project?

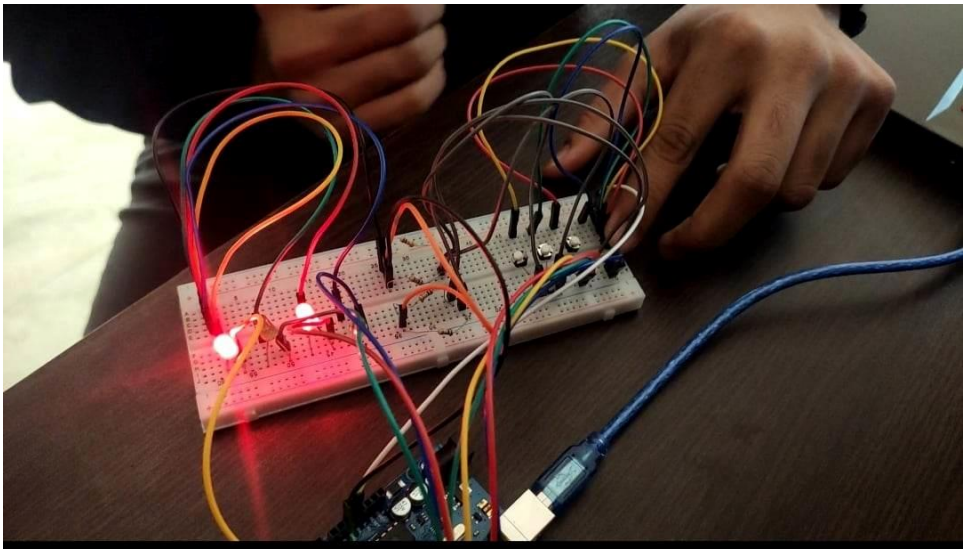
Answer: A level-triggered interrupt module always initiates an interrupt when the level of the interrupt source is asserted. An edge-triggered interrupt module generates an interrupt when the level of the interrupt source shifts from inactive to active. As a result, an edge-triggered interrupt module can only have one interrupt per edge. In contrast, a level-triggered interrupt module can have several interrupts while the interrupt level is asserted. Since continuous input would not be possible with an edge-triggered interrupt module, this project would not work.

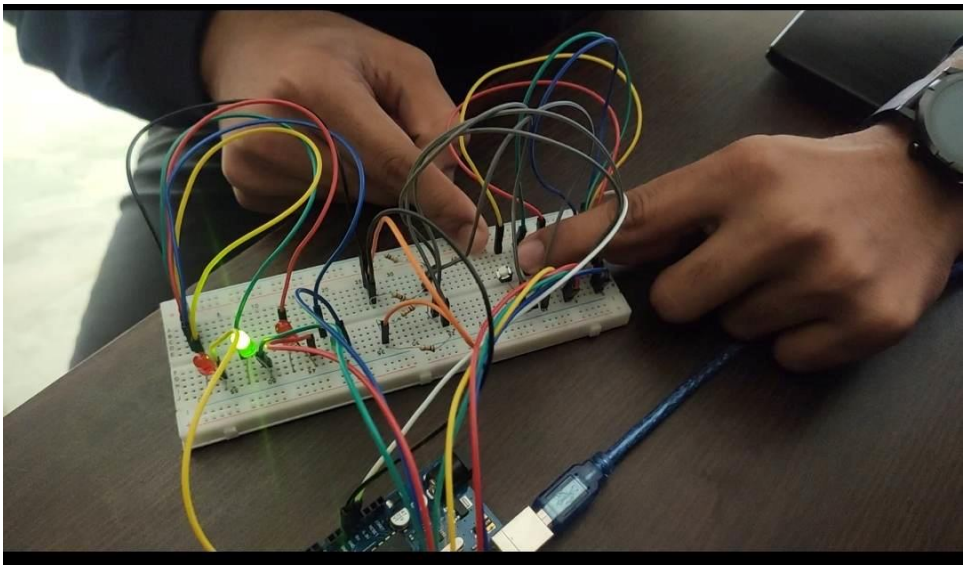
Question 7: Is the project referring to encryption or decryption from input to output?

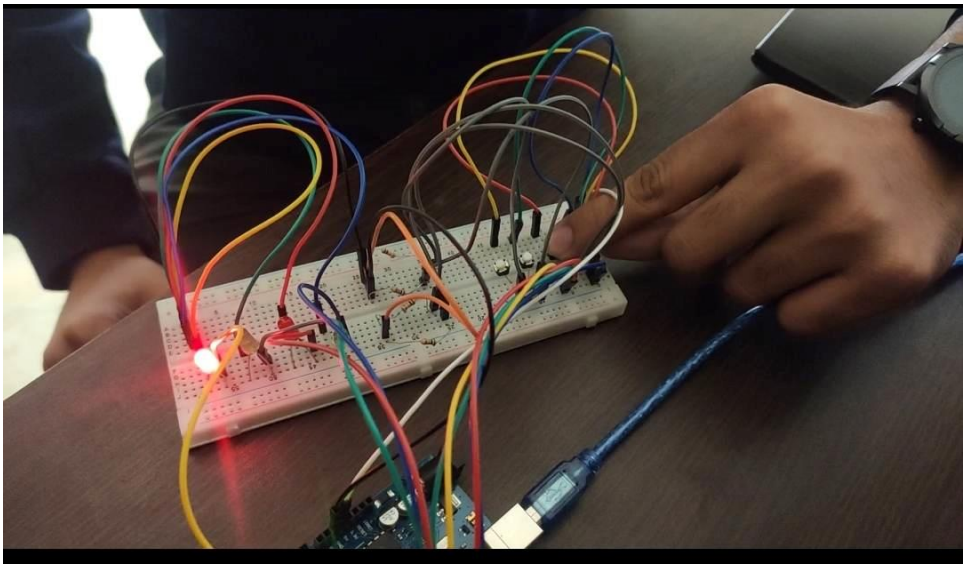
Answer: The input and output lengths are the same, and the encryption table is available. The information we've received is being encrypted. The output is the input's encrypted data.

Hardware Implementation

Here is our hardware implementation:







References

https://en.wikipedia.org/wiki/Proteus_Design_Suite
https://en.wikipedia.org/wiki/Arduino_Uno
https://store.roboticsbd.com/components/761-push-button-switch-robotics-bangladesh.htmlhttps://pediaa.com/what-is-the-difference-between-edge-and-level-triggering/

YouTube link

Software video: https://youtu.be/QG-c88jUI_A

Hardware video: https://youtu.be/uhuZLTOMr-4