North South University

Department of Electrical and Computer Engineering

Project Report

CSE331

Section: 04 || Group: 09

Semester: Summer_2023

Project Name: *Implementation of an encryption table using a microcontroller*

Submitted to: Dr. Dihan Md Nuruddin Hasan

Submission date: 31-10-2023

Name	NSU ID	Email
Shahan Ahmed	2013168042	shahan.ahmed@northsouth.edu
Md. Tajwar Munim Turzo	2012717042	munim.turzo@northsouth.edu
Sadia Hassan Chowdhury	2014105042	sadia.chowdhury1@northsouth.edu
Shikder Motasim Billah	2011342042	shikder.billah@northsouth.edu
Nazia Kamal	2011075042	nazia.kamal10@northsouth.edu

Table of Content

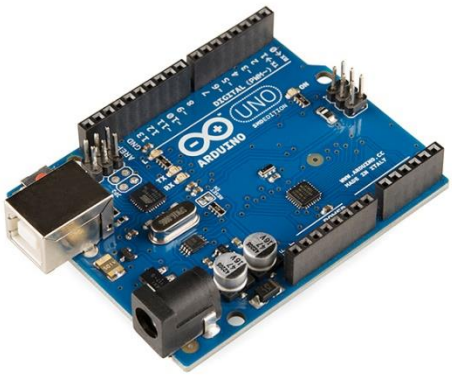
General Description

Abstract:

Arduino is a development board that contains a microcontroller. Arduino boards are relatively inexpensive compared to other microcontroller platforms and their programming environment is easy-to-use. We are going to implement the given encryption table using Arduino. We will use a single pole, and double throw switch to configure the inputs for high and low conditions. So, we are using LEDs to represent the corresponding output statuses.

Arduino UNO:

The Arduino Uno is a popular and widely used microcontroller development board that is part of the Arduino family of open-source hardware and software platforms. It was designed to provide an easy and accessible way for both beginners and experienced engineers to create and prototype a wide range of electronic projects. Here is a general description of the Arduino Uno:



1. **Microcontroller:** The heart of the Arduino Uno is an ATmega328P microcontroller, which is based on the AVR architecture. It runs at 16 MHz and has 32 KB of flash memory for storing your program code, 2 KB of SRAM for data storage, and 1 KB of EEPROM for non-volatile data storage.
2. **Digital and Analog I/O:** The Arduino Uno features 14 digital input/output pins (of which 6 can be used as PWM outputs) and 6 analog input pins. These pins can be used to interface with various sensors, displays, and other components.
3. **Power Supply:** The board can be powered through a USB connection to a computer or an external DC power source. It operates at 5V and can be powered via USB or a 7-12V DC power supply. The built-in voltage regulator ensures a stable 5V supply for the microcontroller.
4. **USB Interface:** The Arduino Uno is equipped with a USB interface for programming and serial communication with a computer. It uses the standard USB-to-serial converter chip (usually the ATmega16U2 or CH340), making it easy to upload code from the Arduino Integrated Development Environment (IDE).

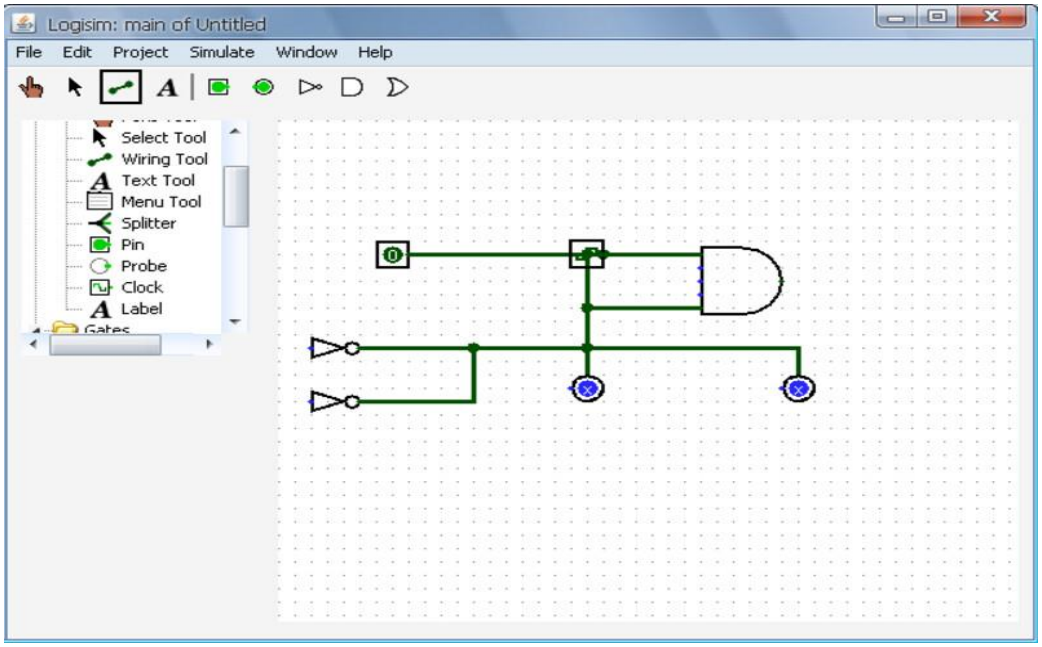
Arduino IDE:

The Arduino Integrated Development Environment (IDE) is a software application that provides a user-friendly platform for programming, compiling, and uploading code to Arduino microcontroller boards. It is a fundamental tool for anyone working with Arduino-based projects. Here's a general description of the Arduino IDE:

1. **Cross-Platform Compatibility:** The Arduino IDE is available for various operating systems, including Windows, macOS, and Linux. This cross-platform compatibility ensures that users can work on their projects regardless of their preferred computing environment.
2. **Open Source:** The Arduino IDE is an open-source software, which means it's freely available to the public. The source code is accessible and can be modified by users, contributing to the collaborative and supportive Arduino community.
3. **User-Friendly Interface:** The IDE provides a straightforward and user-friendly interface, making it accessible to both beginners and experienced developers. It consists of a text editor for writing code, as well as menus and buttons for managing and controlling the development process.
4. **Code Editing:** The integrated text editor in the Arduino IDE offers features such as syntax highlighting, auto-indentation, and code suggestions to help users write and debug their code more efficiently. This editor simplifies the coding process for those new to programming.
5. **Library Support:** The Arduino IDE includes a vast library of pre-written code, known as libraries, that simplifies interfacing with various sensors, modules, and components. These libraries can be easily included in your projects, saving time and effort.
6. **Board and Port Selection:** Users can select the specific Arduino board they are using, along with the communication port to which their board is connected. This ensures that the IDE compiles and uploads the code to the correct target.
7. **Serial Monitor:** The Arduino IDE includes a serial monitor that allows users to communicate with the Arduino board, send and receive data, and monitor the output of their programs. This is invaluable for debugging and troubleshooting.
8. **Compile and Upload:** The IDE provides buttons for compiling the code into a binary file and uploading it to the Arduino board. It automates many of the technical aspects, making it easy to get code running on the hardware.

Logisim:

Logisim is a popular, open-source digital circuit design and simulation tool that provides a user-friendly environment for creating, editing, and simulating digital logic circuits. It is commonly used in educational settings, particularly for teaching digital electronics and computer architecture concepts. Here's a general description of Logisim:



1. **Graphical Circuit Design:** Logisim offers a graphical interface that allows users to design digital logic circuits by arranging and connecting various logic gates and components. Users can create complex circuits by dragging and dropping components onto the workspace.
2. **Component Library:** Logisim includes a library of digital components, including logic gates (AND, OR, XOR, etc.), multiplexers, flip-flops, registers, arithmetic circuits, and various input/output components. Users can select from these pre-built components to construct their circuits.
3. **Custom Components:** Users can create custom components and subcircuits, allowing for the design and reuse of specific circuit modules. This feature is especially useful when working on larger and more intricate projects.
4. **Simulation Capabilities:** Logisim provides simulation tools that allow users to test and verify the functionality of their digital circuits. Users can set input values, apply clock signals, and observe the behavior of the circuit in real-

time. It's an invaluable feature for debugging and understanding how a circuit operates.

5. **User-Friendly Interface:** Logisim's interface is designed to be intuitive and user-friendly, making it accessible to both beginners and experienced circuit designers. The drag-and-drop interface simplifies the process of creating and editing circuits.
6. **Real-Time Feedback:** When simulating a circuit, Logisim provides real-time feedback on the state of each component, enabling users to observe signal propagation and logic changes as they occur in the circuit.
7. **Textual Notation:** For advanced users, Logisim allows for the creation of circuits using a textual notation called the "JAR" file format, which can be edited using a text editor. This feature is handy for more precise and efficient circuit design.
8. **Export and Import:** Logisim supports the export and import of circuit designs, making it easy to share projects with others or use designs in different contexts.

Equipment:

Requirements	
Hardware Requirement	**Software Requirement**
Microcontroller: Arduino UNO Input: 4-Channel Push Button Switch Output: LED (Number of 4) Jumper wires	Arduino IDE Proteus Logisim

Method of Derivation:

Truth Table

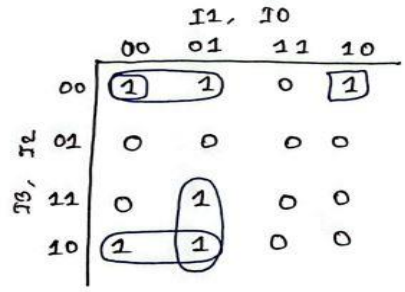
Input				Output			
I3	I2	I1	I0	O3	O2	O2	O1
0	0	0	0	0	1	0	1
1	0	0	0	1	1	1	1
0	1	0	0	0	0	1	0
1	1	0	0	1	1	0	0
0	0	1	0	1	0	1	1
1	0	1	0	1	0	1	0
0	1	1	0	1	1	0	0
1	1	1	0	0	0	0	0
0	0	0	1	0	1	0	1
1	0	0	1	1	0	0	1
0	1	0	1	0	0	1	0
1	1	0	1	1	1	1	1
0	0	1	1	0	1	0	0
1	0	1	1	1	0	1	0
0	1	1	1	1	1	1	0
1	1	1	1	0	0	0	0

We sequentially sorted the table:

	INPUT				**OUTPUT**			
	I3	I2	I1	I0	Q3	Q2	Q1	Q0
0	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**1**
1	**0**	**0**	**0**	**1**	**0**	**1**	**0**	**1**
2	0	0	1	0	1	0	1	1
3	0	0	1	1	0	1	0	0
4	0	1	0	0	0	0	1	0
5	0	1	0	1	0	0	1	0
6	0	1	1	0	1.	1	0	0
7	0	1	1	1	1.	1	1	0
8	1	0	0	0	1.	1	1	1
9	1	0	0	1	1.	0	0	1
10	1	0	1	0	1.	0	1	0
11	1	0	1	1	1.	0	1	0
12	1	1	0	0	1.	1	0	0
13	1	1	0	1	1.	1	1	1
14	1	1	1	0	0	0	0	0
15	1	1	1	1	0	0	0	0

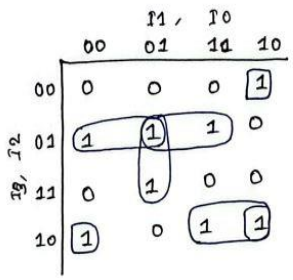
Derived result: (By using K-map)

Sum of products –



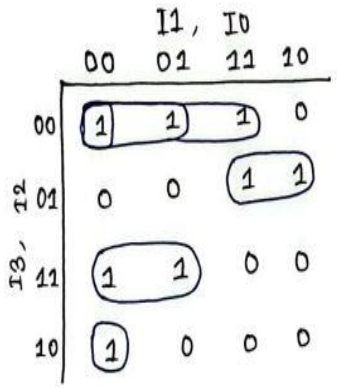
Q0= $\overline{I3}\,\overline{I2}\,\overline{I0} + \overline{I2}\,\overline{I1} + I3\,\overline{I1}\,I0$

Sum of Products –



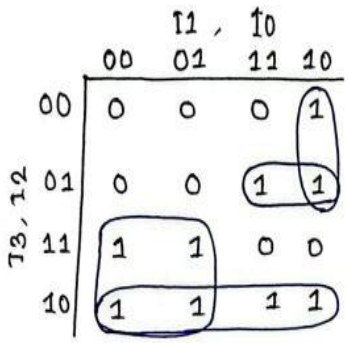
Q1= $\overline{I2}\,I1\,\overline{I0} + \overline{I3}\,I2\,\overline{I1} + \overline{I3}\,I2\,I0 + I2\,\overline{I1}\,I0 + I3\,\overline{I2}\,\overline{I0} + I3\,\overline{I2}\,I1$

Sum of products –



Q2= $\overline{I3}\,\overline{I2}\,\overline{I1} + \overline{I2}\,\overline{I1}\,\overline{I0} + \overline{I3}\,\overline{I2}\,I0 + \overline{I3}\,I2\,I1 + I3\,I2\,\overline{I1}$

Sum of products –



Q3= $\overline{I3}\,\overline{I1}\,\overline{I0} + \overline{I3}\,I2\,I1 + I3\,\overline{I2} + I3\,\overline{I1}$

Circuit Diagram:

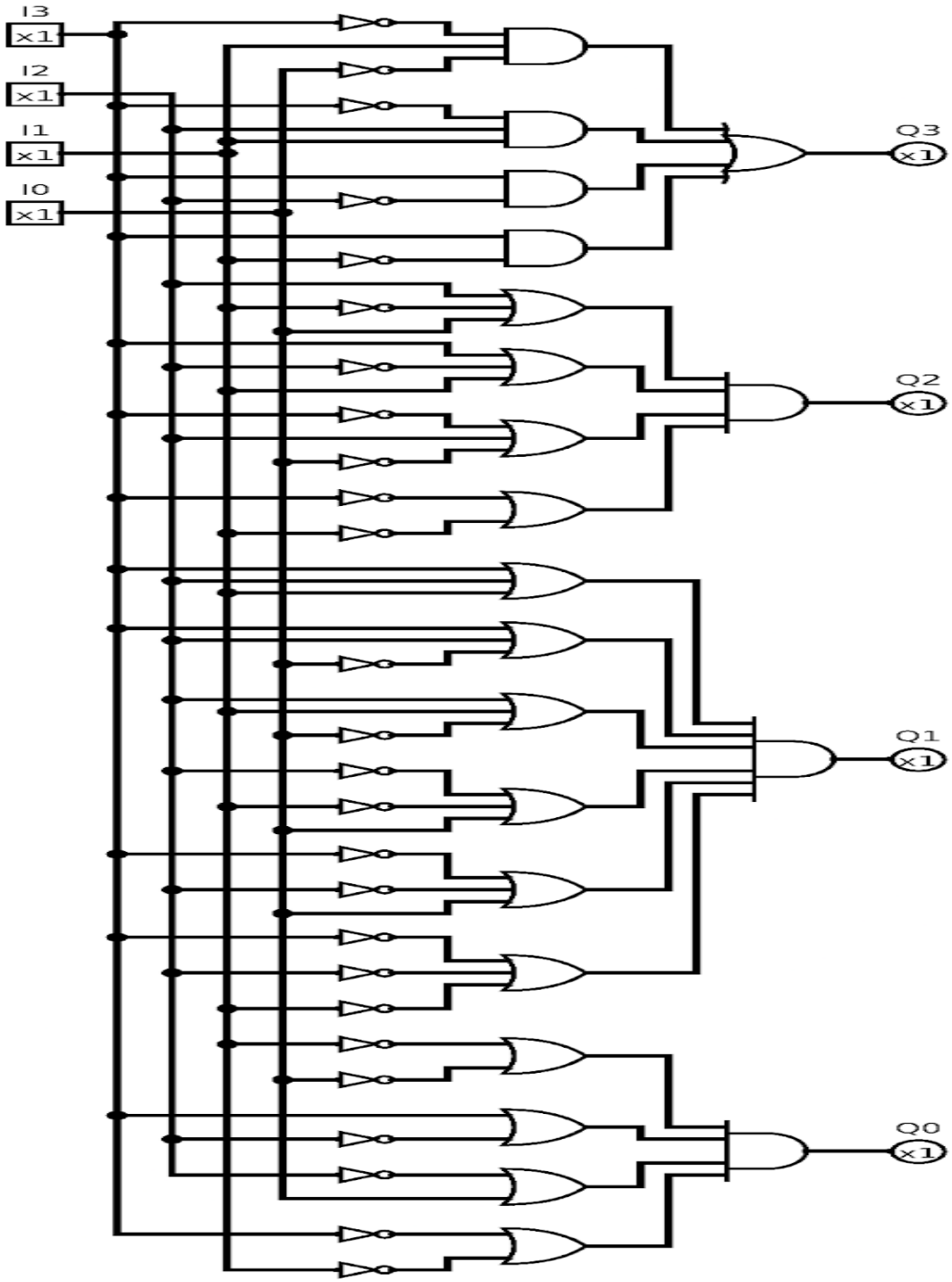


Figure 1 Circuit Diagram for the Given Input and Outputs by using logic gate.

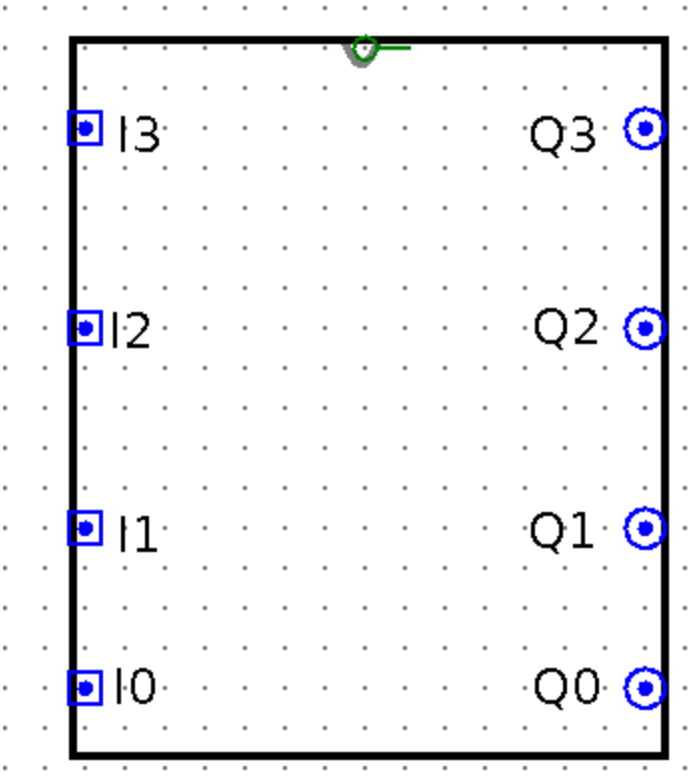


Figure 2 Subcircuit of Figure 1 Circuit Diagram for the Given Input and Outputs by using

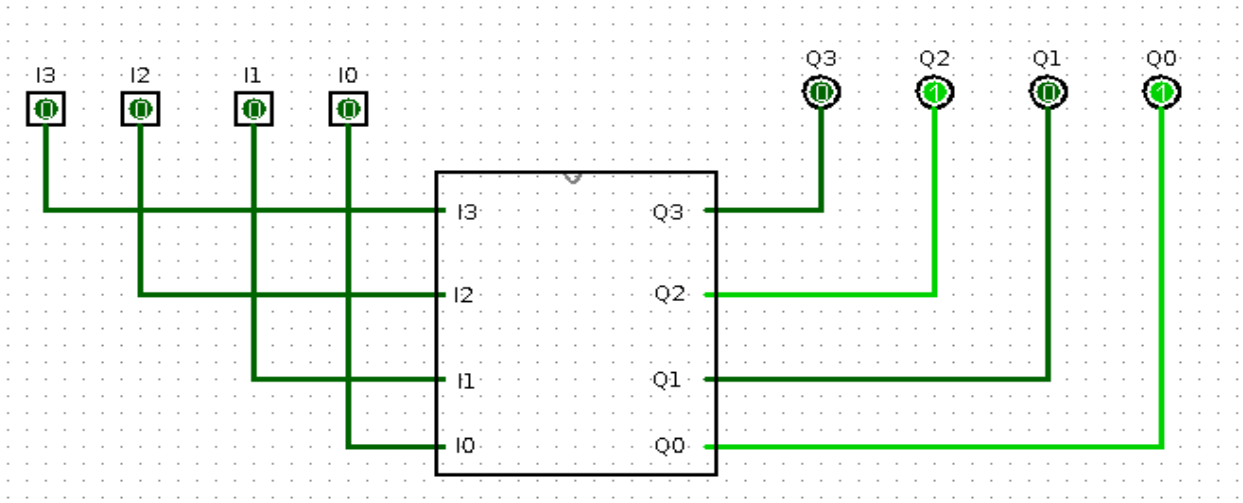


Figure 3 Circuit diagram for the given Input and Output by using subcircuit

Circuit Operation principle:

Our project is designed with Arduino. Arduino is a microcontroller board based on the ATmega328P. It is an open-source electronics platform that is easy to use both software and hardware. It has 14 digital input/output pins, where 6 can be used as PWM outputs, 6 analog inputs, a 16 MHz ceramic resonator (CSTCE16M0V53-R0), a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with an AC-to-DC adapter or battery to get started.

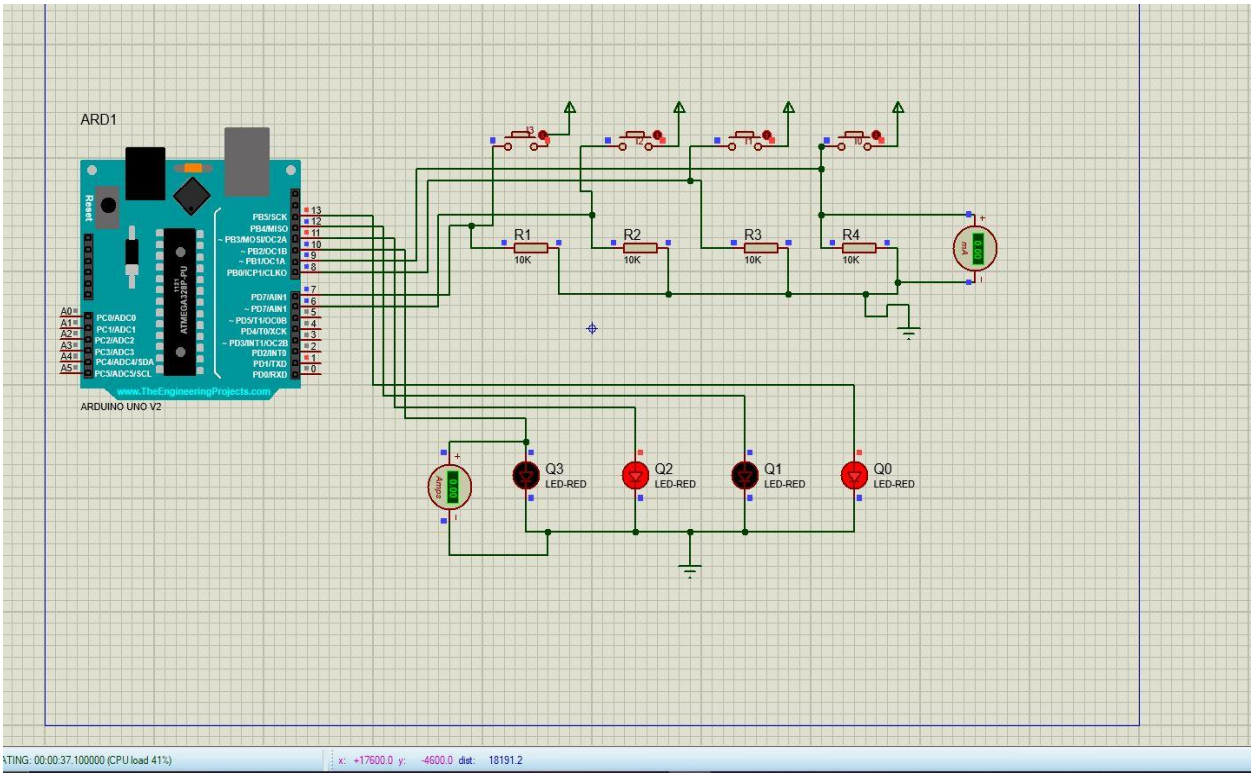


Figure 4 Proteus Diagram for the Given Input and Outputs.

In-Circuit Design we used:

Arduino UNO I/O pin 7-PD0-input button – I3

Arduino UNO I/O pin 6-PD1-input button – I2

Arduino UNO I/O pin 8-PD2-input button – I1

Arduino UNO I/O pin 9-PD3-input button – I0

Arduino I/O pin13-PB2-output ledQ0

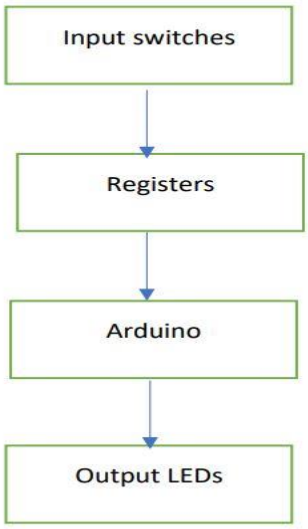
Arduino I/O pin11-PB2-output ledQ1

Arduino I/O pin12-PB2-output ledQ2

Arduino I/O pin13-PB2-output ledQ3

Circuit operation in Arduino is simple. In the Input section, we connected Switches with Resistors. Resistors are used to control the flow of current into other components. In Output, we don't use any switches. The whole operation is done in the Registers of Arduino Board where our code is saved. all the connections are done, we can use the Input Switches which will function according to our code and will give an expected result in output.

Flow Chart:



Arduino Code:

Method-1

```
int LED1=13;
int LED2=12;
int LED3=11;
int LED4=10;

int button1=9;
int button2=8;
int button3=6;
int button4=7;



void setup() {
Serial.begin(9600);
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
pinMode(LED4, OUTPUT);

pinMode(button1,INPUT);
pinMode(button2,INPUT);
pinMode(button3,INPUT);
pinMode(button4,INPUT);







}

void loop() {
// digitalWrite(LED1,HIGH);
// digitalWrite(LED2,HIGH);
// digitalWrite(LED3,HIGH);
// digitalWrite(LED4,HIGH);



int value1=digitalRead(button1);
int value2=digitalRead(button2);
int value3=digitalRead(button3);
int value4=digitalRead(button4);

int B_ZERO = ((value4==1 && value3==1 && value2==1 && value1==0) ||
(value4==1 && value3==1 && value2==1 && value1==1)); //14, 15 ALL LED OFF

int B_TWO = (value4==0 & value3==1 & value2==0 & value1==0) | (value4==0 &
value3==1 & value2==0 & value1==1);// 4,5 LED: 0,0,1,0
int B_FOUR = (value4==0 & value3==0 & value2==1 & value1==1);
```

```
int B_FIVE = (value4==0 & value3==0 & value2==0 & value1==0) | (value4==0
& value3==0 & value2==0 & value1==1);
int B_NINE= (value4==1 & value3==0 & value2==0 & value1==1);
int B_TEN = (value4==1 & value3==0 & value2==1 & value1==0) | (value4==1 &
value3==0 & value2==1 & value1==1);
int B_ELEVEN= (value4==0 & value3==0 & value2==1 & value1==0);
int B_TWELVE = (value4==0 & value3==1 & value2==1 & value1==0) |
(value4==1 & value3==1 & value2==0 & value1==0);
int B_FOURTEEN= (value4==0 & value3==1 & value2==1 & value1==1);
int B_FIFTEEN = (value4==1 & value3==0 & value2==0 & value1==0) |
(value4==1 & value3==1 & value2==0 & value1==1);



// Serial.println(B_ZERO);

if (B_ZERO==1)
{
digitalWrite(LED4,LOW);
digitalWrite(LED3,LOW);
digitalWrite(LED2,LOW);
digitalWrite(LED1,LOW);
}
else if (B_TWO==1)
{
digitalWrite(LED4,LOW);
digitalWrite(LED3,LOW);
digitalWrite(LED2,HIGH);
digitalWrite(LED1,LOW);
}

else if (B_FOUR==1)
{
digitalWrite(LED4,LOW);
digitalWrite(LED3,HIGH);
digitalWrite(LED2,LOW);
digitalWrite(LED1,LOW);
}

else if (B_FIVE==1)
{
digitalWrite(LED1,LOW);
digitalWrite(LED2,HIGH);
digitalWrite(LED3,LOW);
digitalWrite(LED4,HIGH);
}

else if (B_NINE==1)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,LOW);
digitalWrite(LED3,LOW);
digitalWrite(LED4,HIGH);
```

```
}

else if (B_TEN==1)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,LOW);
digitalWrite(LED3,HIGH);
digitalWrite(LED4,LOW);
}

else if (B_ELEVEN==1)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,LOW);
digitalWrite(LED3,HIGH);
digitalWrite(LED4,HIGH);
}
else if (B_TWELVE==1)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,HIGH);
digitalWrite(LED3,LOW);
digitalWrite(LED4,LOW);
}

else if (B_FOURTEEN==1)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,HIGH);
digitalWrite(LED3,HIGH);
digitalWrite(LED4,LOW);
}

else if (B_FIFTEEN==1)
{
digitalWrite(LED1,HIGH);
digitalWrite(LED2,HIGH);
digitalWrite(LED3,HIGH);
digitalWrite(LED4,LOW);
}
}
```

```
/home/shahan/.arduino15/packages/arduino/tools/avr-gcc/7.3.0-atmel3.6.1-arduino7/bin/avr-size -A /tmp/arduino/sketches/
Sketch uses 2368 bytes (7%) of program storage space. Maximum is 32256 bytes.
Global variables use 184 bytes (8%) of dynamic memory, leaving 1864 bytes for local variables. Maximum is 2048 bytes.
```

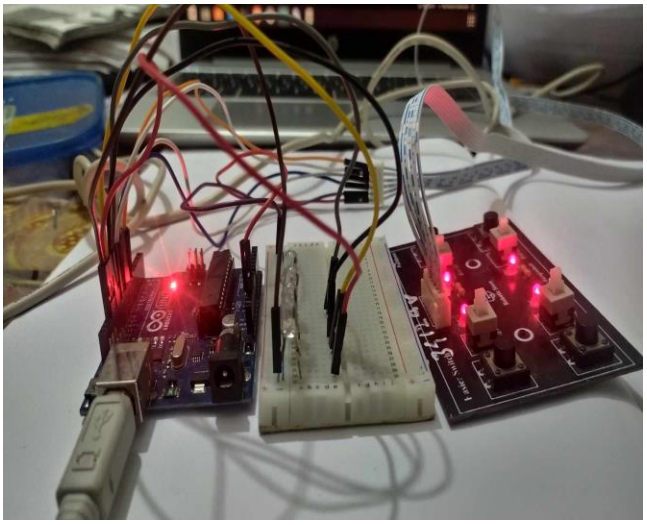
Method-2

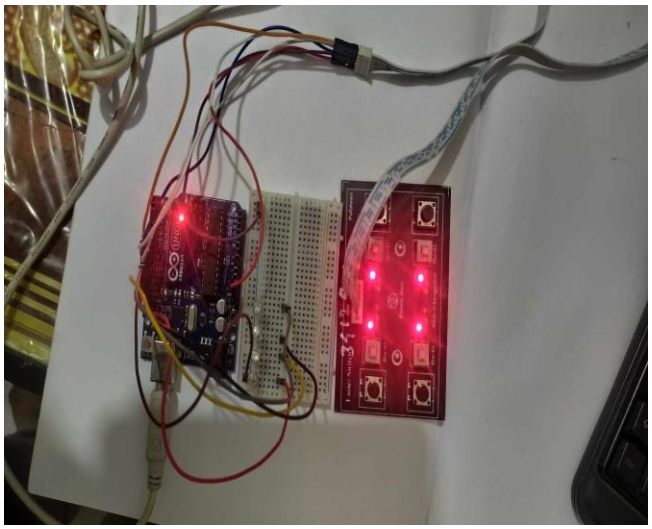
```
int LED1=13;
int LED2=12;
int LED3=11;
int LED4=10;

int button1=9;
int button2=8;
int button3=6;
int button4=7;


void setup() {
Serial.begin(9600);
pinMode(LED1, OUTPUT);
pinMode(LED2, OUTPUT);
pinMode(LED3, OUTPUT);
pinMode(LED4, OUTPUT);

pinMode(button1,INPUT);
pinMode(button2,INPUT);
pinMode(button3,INPUT);
pinMode(button4,INPUT);




}

void loop() {
int value1=digitalRead(button1);
int value2=digitalRead(button2);
int value3=digitalRead(button3);
int value4=digitalRead(button4);

int expression1= (!value4 & value2 & !value1) | (!value4 & value3 &
value2) | (value4 & value2) | (value4 &!value1);

int expression2= (!value4 & !value3 & !value3) | (!value3 & !value2 &
!value1) | (!value4 & !value3 & value1) | (!value4 & value3 & value2) |
(value4 & value3 & !value2);

int expression3= (!value3 & value2 & !value4) | (!value4 & value3 &
!value2) | (!value4 & value3 & value1) | (value3 & !value2 & value1) |
(value4 & !value3 & ! value1) | (value4 & !value3 & value2);

int expression4= (!value4 & !value3 & !value1) | (!value3 & !value2) |
(value4 & !value2 & value1);

digitalWrite(LED1,expression1);
```

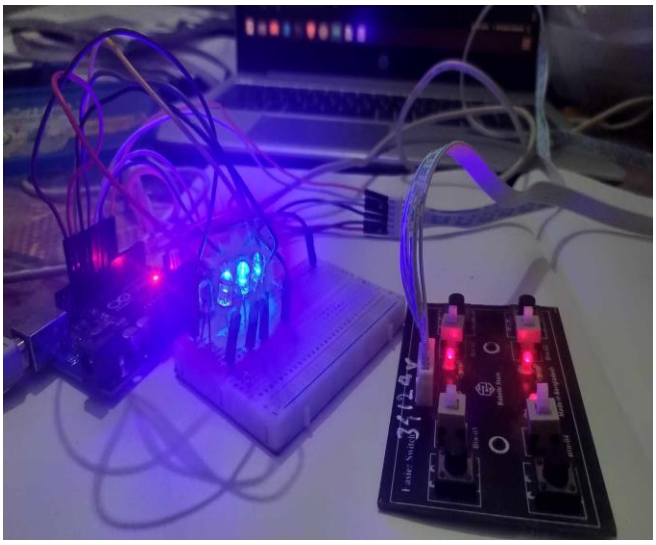
```
digitalWrite(LED2,expression2);
digitalWrite(LED3,expression3);
digitalWrite(LED4,expression4);


}
```

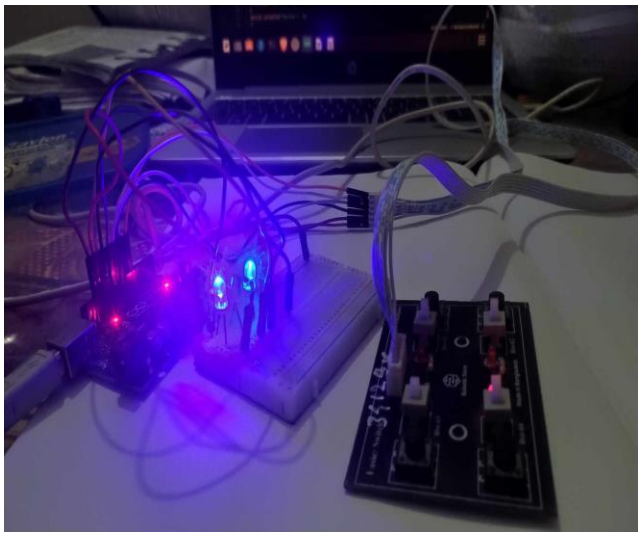
```
/home/shahan/.arduino15/packages/arduino/tools/avr-gcc/7.3.0-atmel3.6.1-arduino7/bin/avr-size -A /tmp/arduino/sketches/2A4283DF3EC7B029D6B9C39EF5CE512
Sketch uses 2230 bytes (6%) of program storage space. Maximum is 32256 bytes.
Global variables use 184 bytes (8%) of dynamic memory, leaving 1864 bytes for local variables. Maximum is 2048 bytes.
```

Hardware Implementation:









Result and Analysis:

	Method 1	Method 2
Maximum program space	32256 bytes	32256 bytes
Sketch uses	**2368 bytes (7%) of program storage space**	**2230 bytes (6%) of program storage space**
Maximum Global variables	2048 bytes	2048 bytes
Dynamic memory	184 bytes (8%)	184 bytes (8%)
Local variables	1864 bytes	1864 bytes

Conclusion:

The combinational circuit under discussion pertains to the utilization of memory through 16 distinct inputs with varying output requirements. Initially, this circuit was simulated using the Proteus simulator and subsequently implemented on real hardware using a microcontroller. This report outlines two distinct methodologies, with one clearly demonstrating a more efficient memory usage than the other.

In the first methodology, four push switches serve as input, while four LEDs serve as output indicators. The problem was addressed primarily through a series of conditional "if-else" statements. This approach involved formulating specific output configurations for each of the 16 potential input combinations. While this method produced the correct results, it was accompanied by a significant drawback: the substantial memory allocation required to store the numerous conditional expressions. Given the multitude of input possibilities, this approach resulted in a substantial memory overhead.

Conversely, the second methodology employed a more streamlined approach to address the same problem. It utilized Karnaugh Maps (K-maps) to simplify the output equations. Once the K-maps were analyzed and minimized, only four distinct equations were derived to determine the desired output. This elegant approach led to a significant reduction in memory consumption, as it required a smaller number of instructions and yielded improved execution time.

In summary, the first methodology, while functional, proved to be memory-intensive due to the proliferation of conditional expressions required to cover all input scenarios. The second methodology, with its utilization of K-maps, efficiently reduced memory overhead and optimized execution, making it a more favorable choice for this particular combinational circuit.

Reference:

1. https://docs.arduino.cc/learn/programming/memory-guide

2. https://roboticsbackend.com/arduino-millis-vs-micros/#:~:text=completely%20open%20source.-,When%20to%20use%20Arduino%20millis()%20vs%20micros(),not%2C%20just%20use%20millis().

3. https://electronicslovers.com/2015/04/how-to-use-virtual-terminal-in-proteus.html