NORTH SOUTH UNIVERSITY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

SPRING 2022

CSE 331

Microprocessor Interfacing & Embedded System

Project Report

'Analysis of Power Consumption of different Microcontrollers'

Faculty: **Dr. Dihan Md. Nuruddin Hasan**
Section: **2**
Group: **6**

Names and IDs of the Group Members: **Remarks:**

Margub Hussain Musawi	1821295642	
Md Abu Jubair Dihan	1821320042	
Md. Rifat Ahmed	1931725042	
Ashfe Asade Simon	1911962642	
Saif Noor	1912510642	

Table of Contents

1. Introduction

1.1 Main Objective

- This project's primary objective was to compare the power consumption of several microcontroller boards (which are available for sale in the market) for a unique situation.
- In the special case, the microcontrollers will use a 4 bit logic input.
- The microcontroller would then use an encryption method in accordance with a truth table that was provided (Table 2.1), and would then produce a 4 bit value that could be displayed with LEDs.
- The purpose of this experiment is to measure the amount of current that the circuit draws under various input and system conditions. For a 4 bit input, there are 16 possible inputs (ranging from 0000 to 1111); for each of these combinations, the circuit's current is measured. This is performed for at least 2 microcontroller boards with various specifications.
- Data for all the boards has been collected, and a graphical analysis has been done to provide a comparison between the two boards.
- All of the microcontrollers' input conditions must be kept constant in order to produce accurate results. A circuit with a constant power supply, as an example.

2. Method of Derivation

2.1 Truth Table

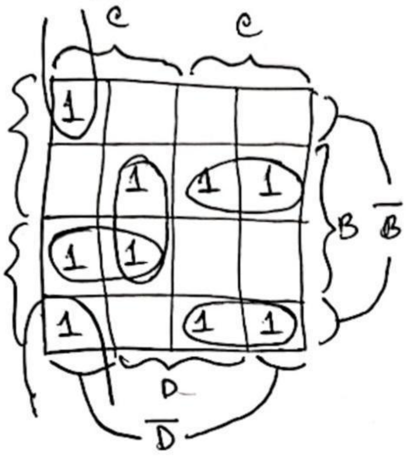
Input				Output			
I3	**I2**	**I1**	**I0**	**O3**	**O2**	**O1**	**O0**
0	0	0	0	1	1	0	1
1	0	0	0	1	0	1	1
0	1	0	0	0	1	0	1
1	1	0	0	1	0	1	1
0	0	1	0	0	0	0	0
1	0	1	0	1	0	1	1
0	1	1	0	1	0	0	0
1	1	1	0	0	1	1	0
0	0	0	1	0	0	0	1
1	0	0	1	0	1	0	0
0	1	0	1	1	0	0	0
1	1	0	1	1	0	1	1
0	0	1	1	0	0	1	1
1	0	1	1	1	1	1	0
0	1	1	1	1	0	0	1
1	1	1	1	0	0	1	0

Table 2.1: Truth Table

2.2 Boolean Expression using Karnaugh Map (K-Map)

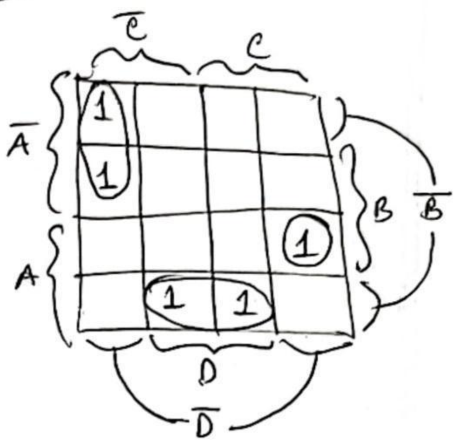
We have to derive the Boolean expression for the outputs according to the Truth Table (Table 2.1). So for each output we will get one Boolean expression. The Boolean expression were derive using K-map.

O3



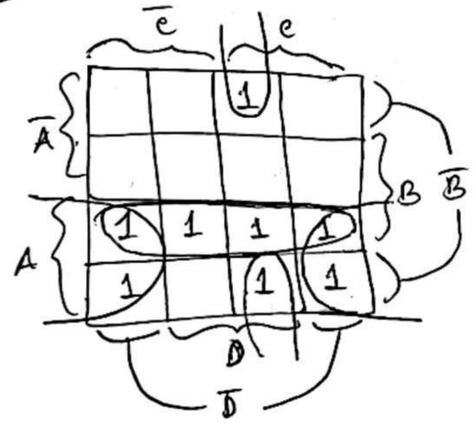
$O3 = AB\overline{C} + A\overline{B}C + \overline{A}BC + B\overline{C}D + \overline{B}\,\overline{C}\,\overline{D}$

O2



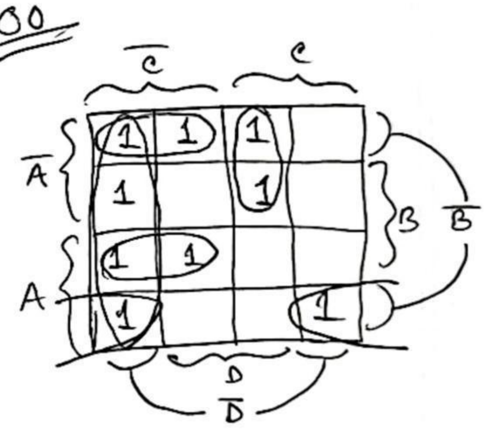
$O2 = A\overline{B}D + \overline{A}\,\overline{C}\,\overline{D} + ABC\overline{D}$

O1



$O1 = AB + A\overline{D} + \overline{B}CD$

O0



$O0 = \overline{C}\,\overline{D} + AB\overline{C} + \overline{A}\,\overline{B}\,\overline{C} + A\overline{B}\,\overline{D} + \overline{A}CD$

<u>The Boolean expressions are:</u>

O0 = C'D' + ABC' + A'B'C' + AB'D' + A'CD

O1 = AB + AD' + B'CD

O2 = AB'D + A'C'D' + ABCD'

O3 = ABC' + AB'C + A'BC + BC'D + B'C'D'

Note: Here, A = I3, B = I2, C = I1 & D = I0

2.3 LogiSim Simulation

To confirm that the Boolean expressions derived are working, we build the logic circuit in LogiSim (specification in Appendix A) and tested the expressions. And the expressions are working fine. These expression will be implemented in the microcontroller.

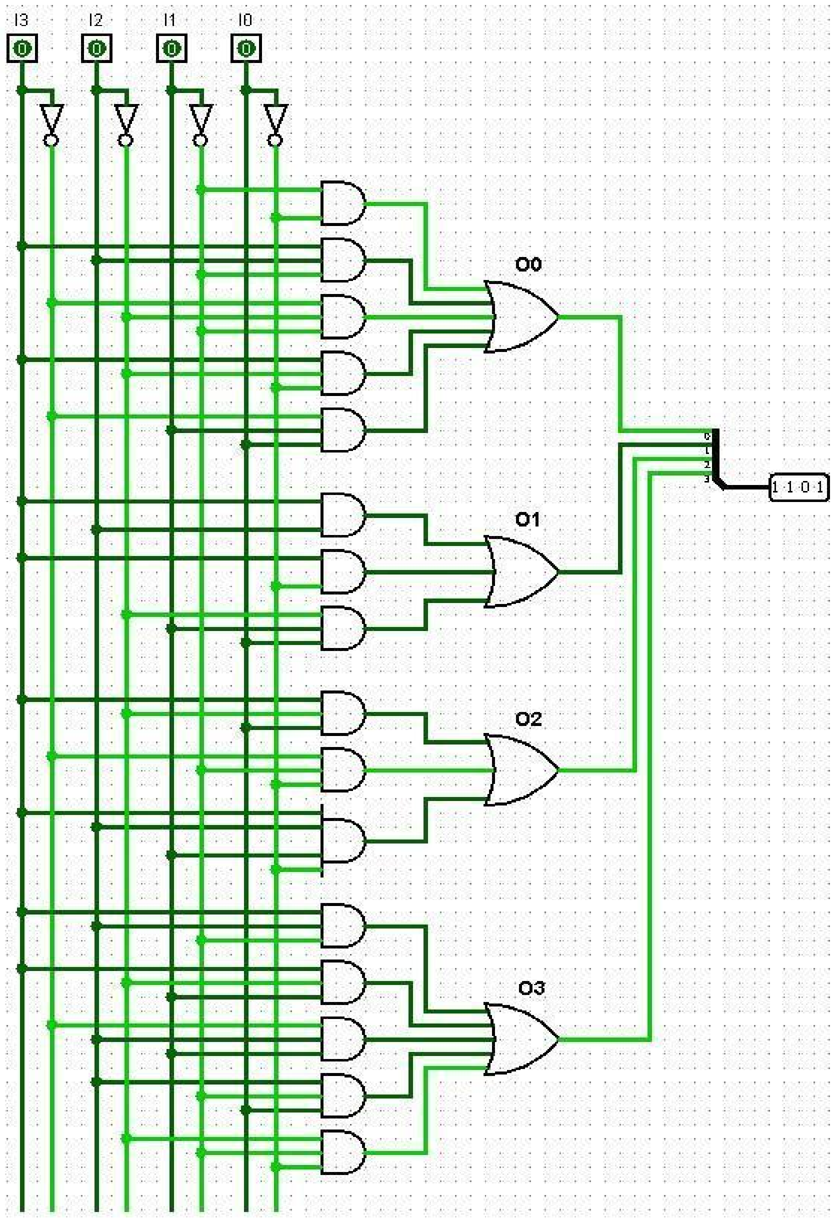


Figure 2.2: Logic Circuit for the derived Boolean Expressions

3. Circuit Design & Simulation

3.1 Circuit Design Procedure

Every commercial microcontroller board has some specific number of General Purpose Input Output (GPIO) pins which can be used to handle digital inputs and outputs. Some GPIO pins are Digital & a few are Analog. We will be using the digital GPIO pins. For digital I/O the pins can operate at only two states: Logic 1 (High) & Logic 0 (Low).

Circuit Connection Methodology:

- A single 9V constant power supply is used to power the whole system.
- We need 8 Digital GPIO pins of the microcontroller: 4 for digital input & 4 for digital output.
- We connected 4 input pins with 4 toggle switches (4-bit dip switch) with 4 pull down resistors (22 KΩ each) to provide the logic input sequence.
- We connected 4 LEDs to the 4 digital output pins through 330Ω resistors so that the LED doesn't burn.
- This procedure is followed for all the microcontroller boards being used for the experiment.

We have done the experiment using 2 microcontroller boards:

1. Arduino Uno R3 (16 MHz)
2. NodeMCU V2 (80MHz)

(Specifications of these boards are given in Appendix B).

3.2 Simulation Procedure

Simulation was done using Proteus 8 Professional (Appendix A). We cannot simulate Arduino directly in Proteus, cause the specific libraries for it was needed to be installed first.

NodeMCU V2 cannot be simulated using Proteus as the simulation library for the board was not present.

Then the code needed to be written to implement the Boolean expression that we had derived. Then the code was converted into the hex file using Arduino IDE (Appendix A) and loaded into Proteus to operate the circuit in Proteus.

After performing the simulation of the two boards (Uno & NodeMCU) we saw that our code was working fine and we were getting our desired output. The snapshots of our circuit diagram for the three boards is given in the following section.

3.3 Program Logic Flow Chart

All the codes for the design are given in Appendix C. All the codes for the three boards use this same concept. A small difference is there for NodeMCU V2 in defining the pins.

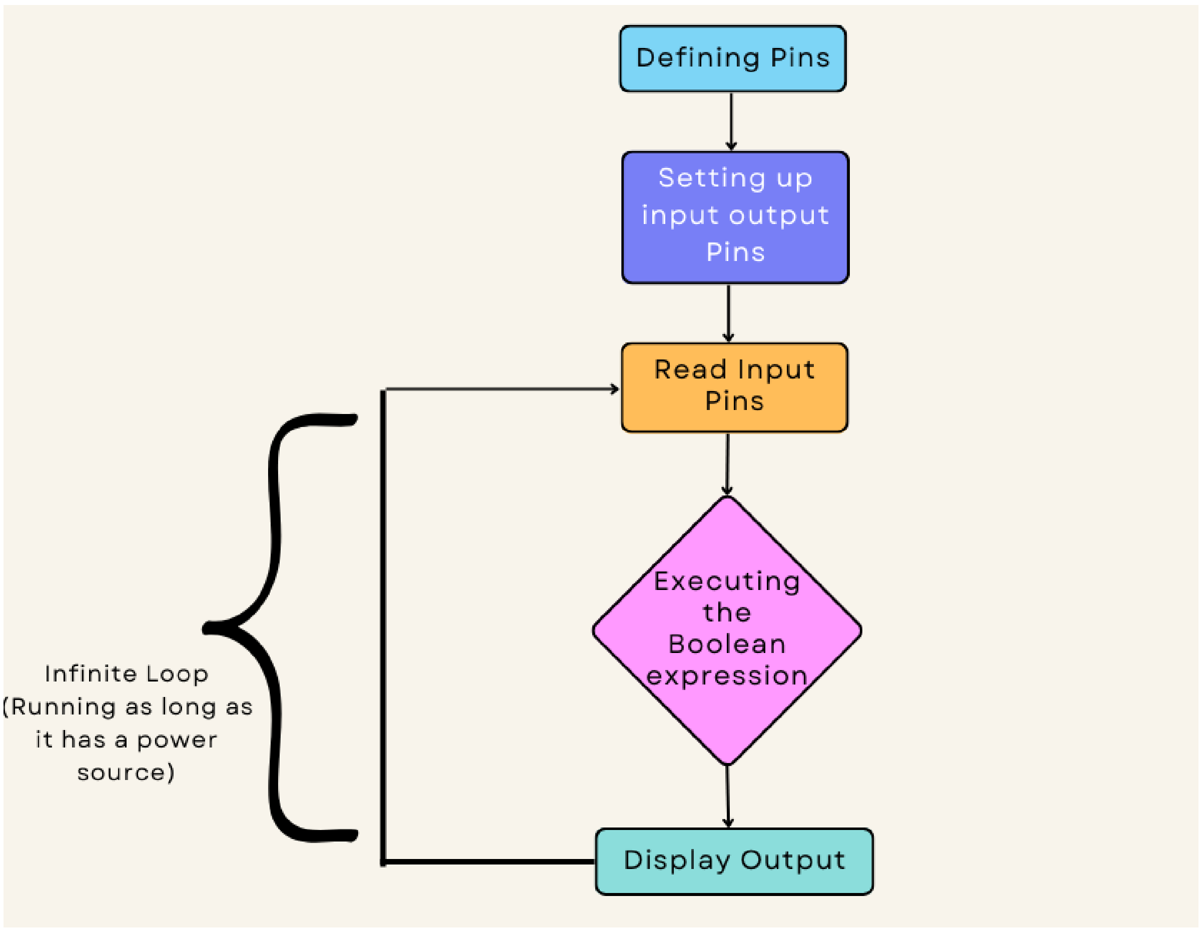
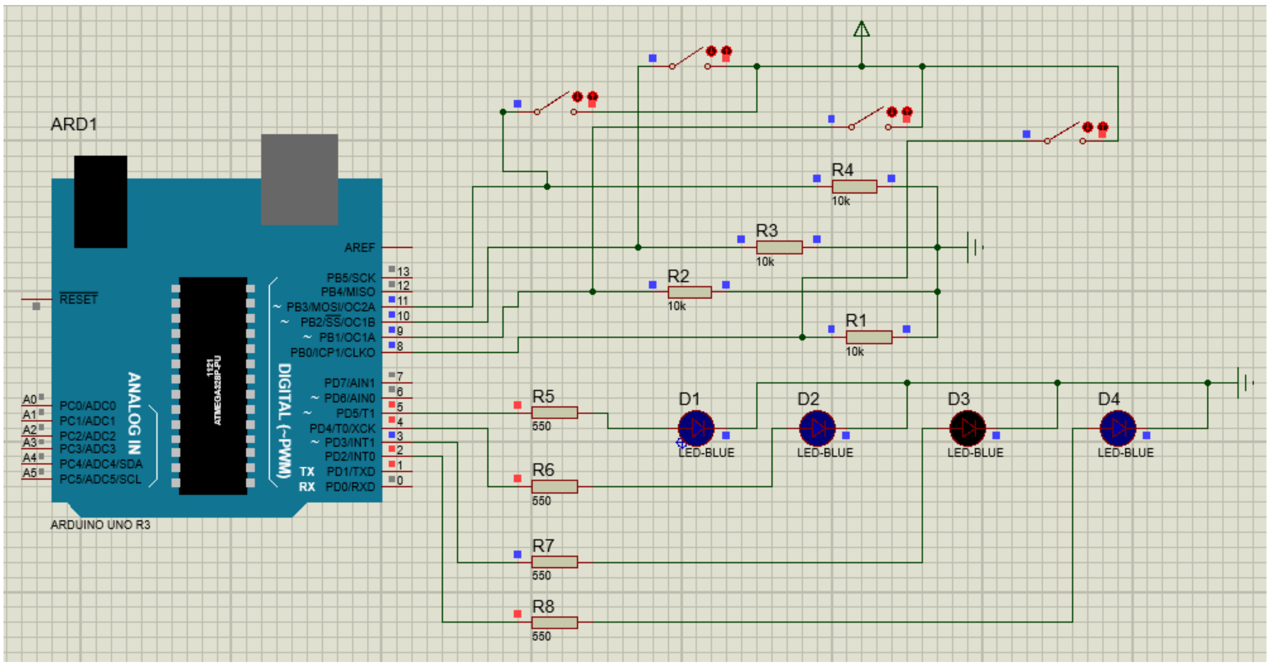

Defining Pins
Setting up input output Pins
Read Input Pins
Executing the Boolean expression
Display Output
Infinite Loop (Running as long as it has a power source)

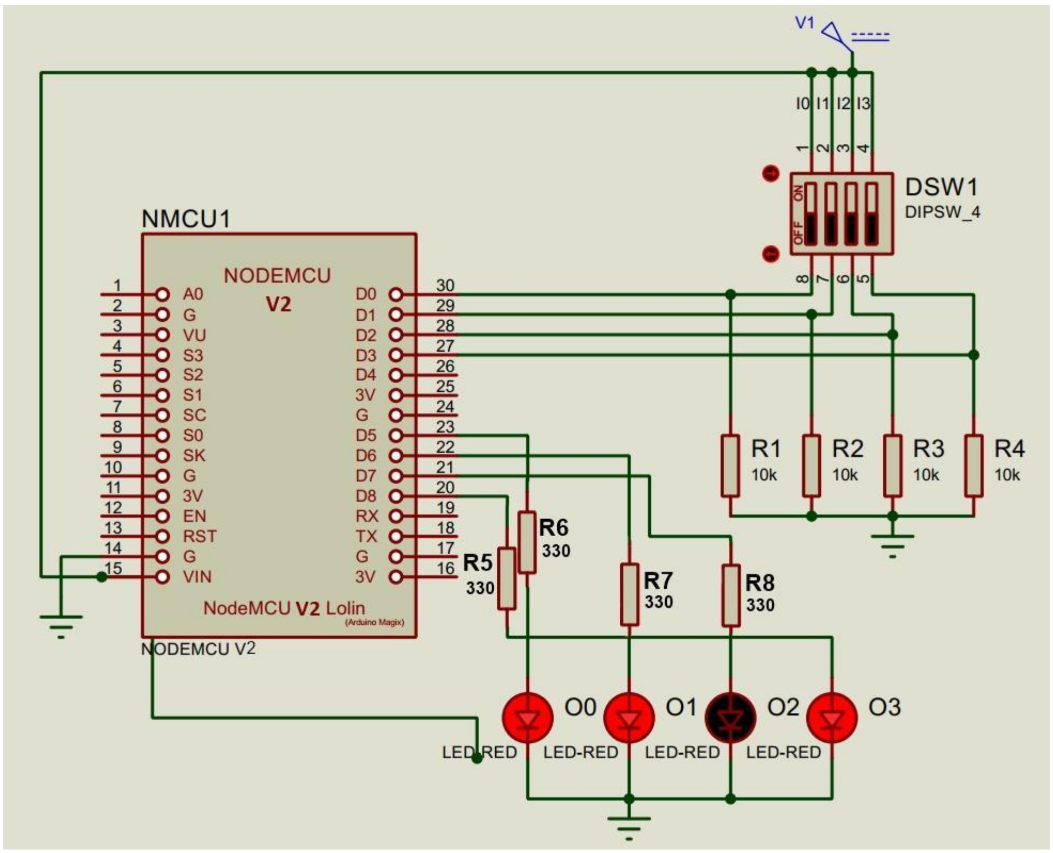
Figure 3.3: Program Logic Flow Chart

3.4 Arduino Uno R3 Simulation



Circuit 3.4: Circuit diagram for Arduino Uno R3

3.5 NodeMCU V2 Schematic



Circuit 3.5: Circuit diagram for NodeMCU V2

4. Circuit Operation

4.1 Working Principle

- The whole system is powered by a 9V ($\boldsymbol{V_{cc}}$) power supply.
- Current from the $\boldsymbol{V_{cc}}$ is branched off in two directions:
    1. Vin pin of the microcontroller boards.
    2. Into the Switch branch (*Input Side of the Circuit*).
- In the switch branch there are 4 sub branches which represent the 4 logic inputs and each branch has a 22 KΩ pull down resistor. The pull down resistors keep the input at ground level (Logic Low) when the switch is off.
- When the switch is on, current flows in all the 4 branches. Internal resistance of the input pins is much higher (in the range of MΩ) than 10 KΩ so a small current flow into the input pins. The rest of the current flows through the resistor to ground.
- Then the output LEDs are connected through 330Ω resistors to 4 GPIO pins of microcontrollers so that the LEDs don't burn.
- Power delivered in the Vin pins is used to power the LEDs.
- The microprocessor processes the input values received from the switch branch and delivers the appropriate logic to the output pins, hence light up the LEDs accordingly.

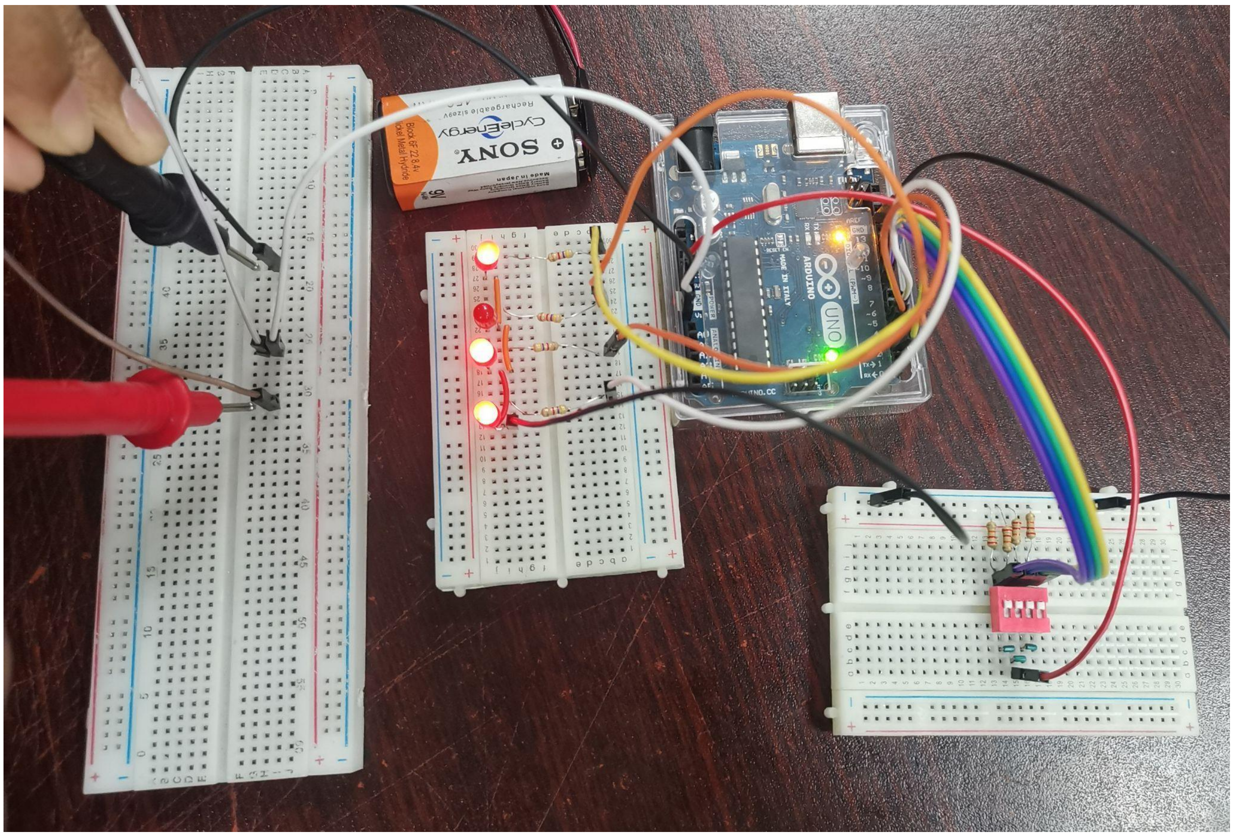
5. Hardware Implementation

5.1 Equipment & Components Used

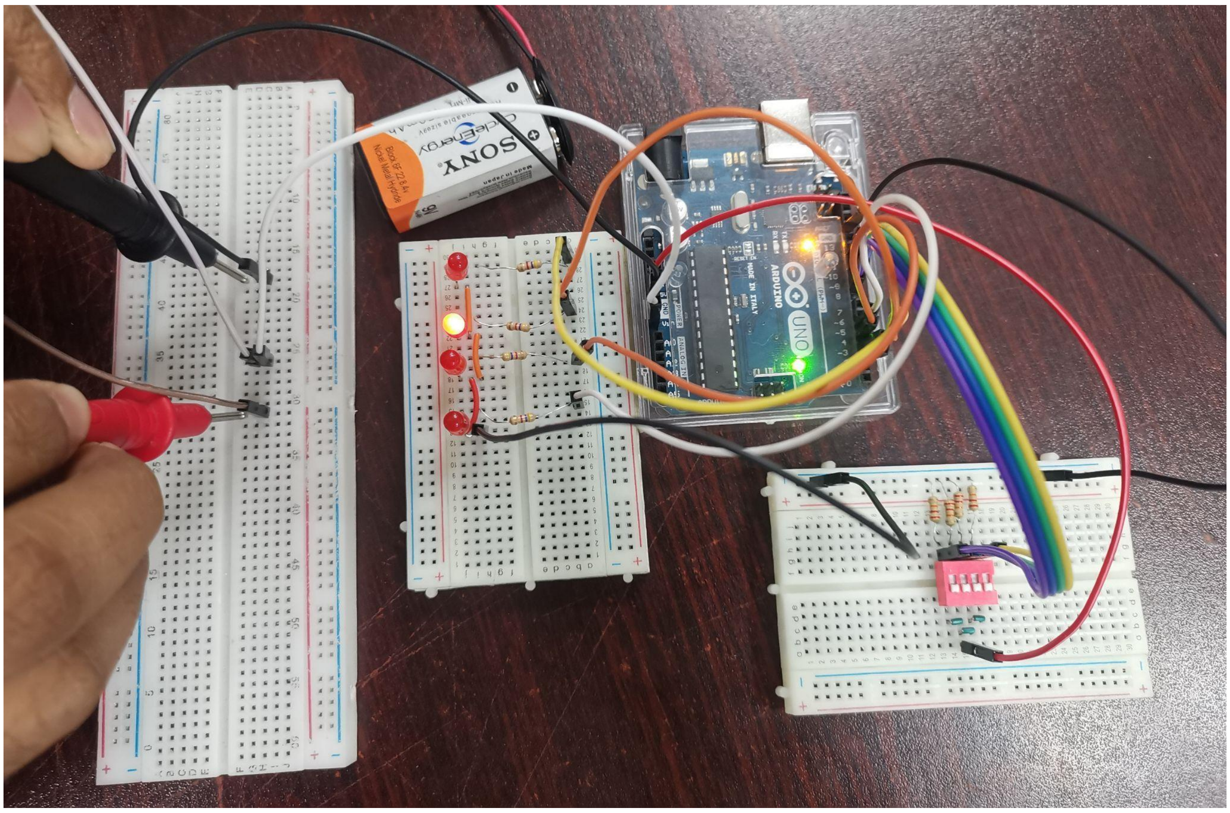
Serial No.	Name	Quantity
1	9 Volt Rechargable Battery	1
2	Breadboard	2
3	4-bit DIP Switch	1
4	Resistor (330 Ω)	4
5	Resistor (22 KΩ)	4
6	LED (Red)	4
7	Arduino Uno R3	1
9	NodeMCU V2 (ESP8266)	1
10	Digital Multimeter	1
11	Jumper Wires	As required

Table 5.1.1: Equipment & Components used

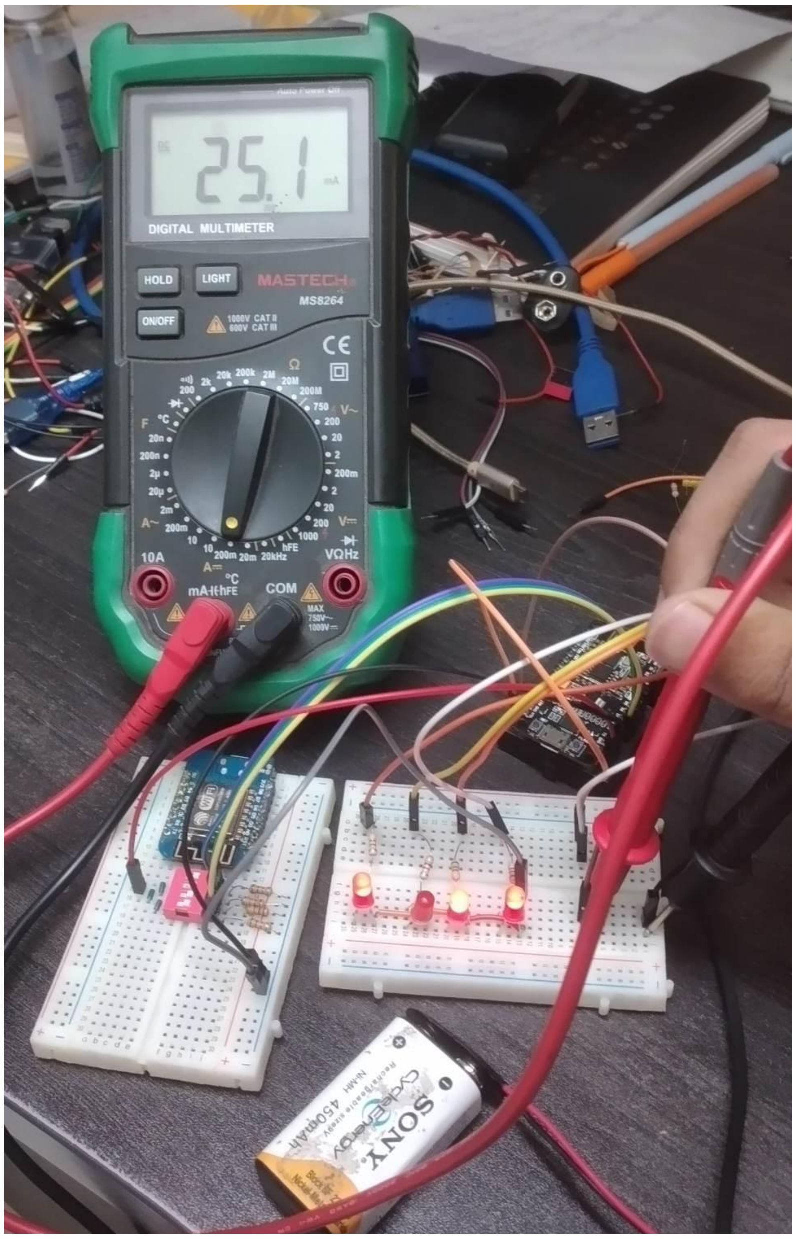
5.2 Arduino Uno R3 Implementation



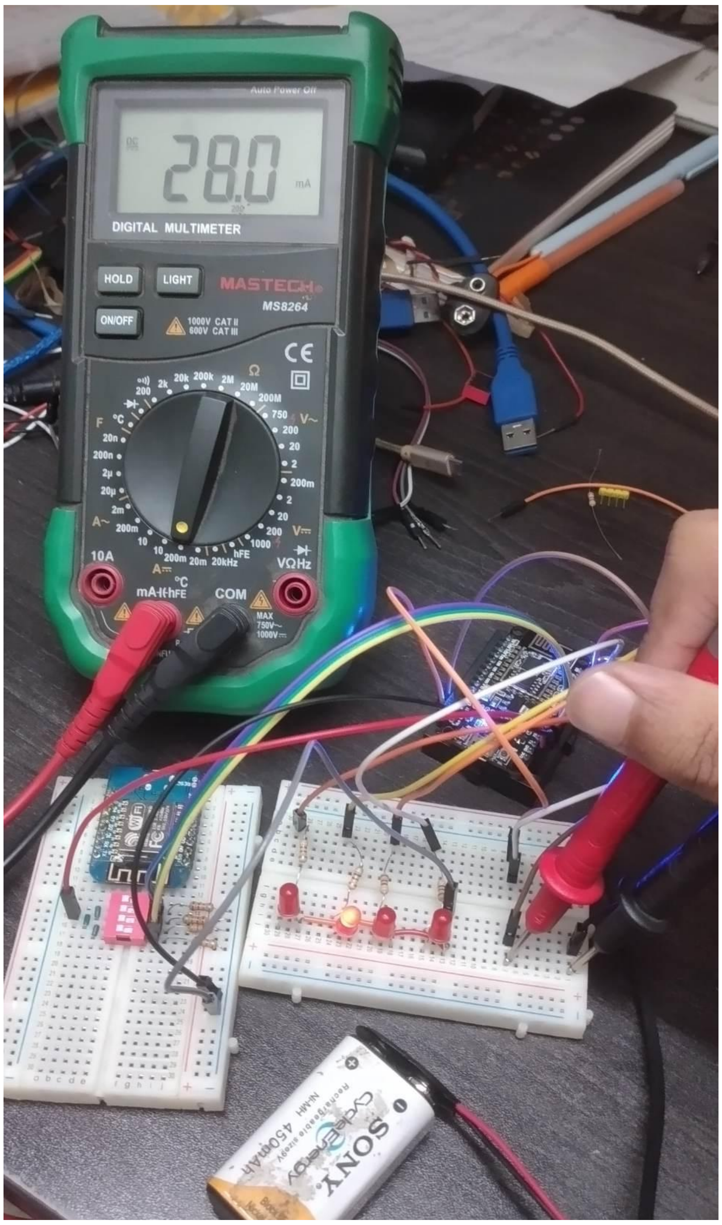
Circuit 5.2.1: Implementation of Arduino Uno for input **0000**


Circuit 5.2.2: Implementation of Arduino Uno for input **1111**

5.3 NodeMCU V2 Implementation

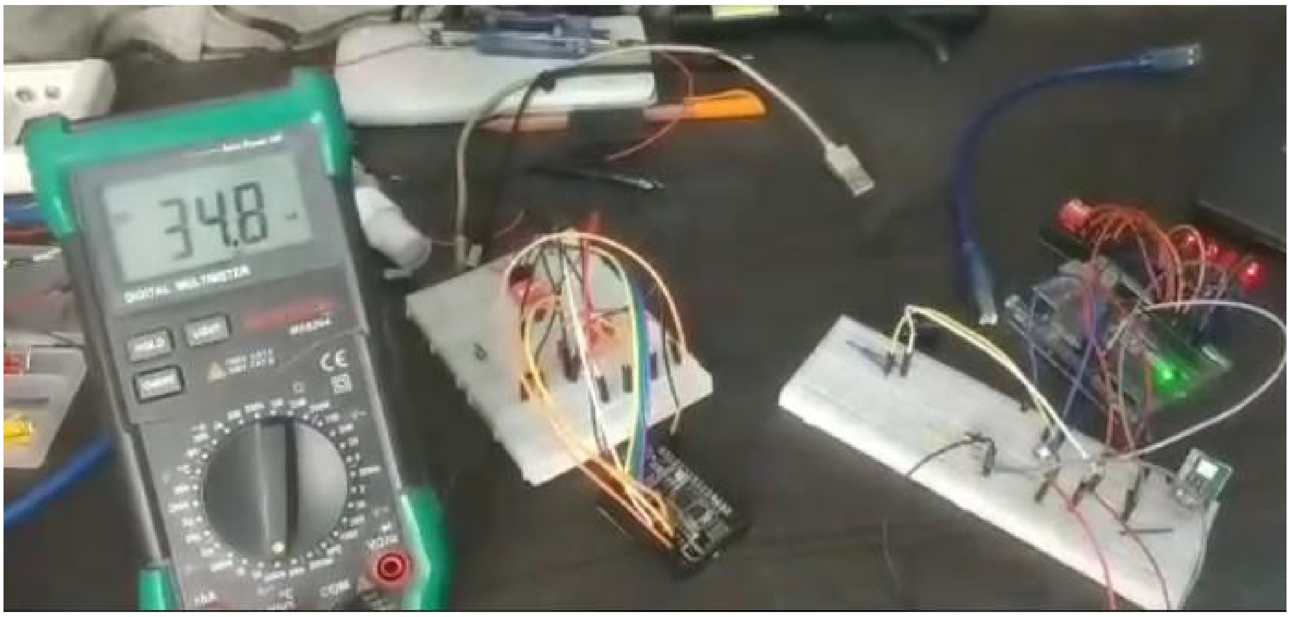


Circuit 5.3.1: Implementation of NodeMCU V2 for input **0000**

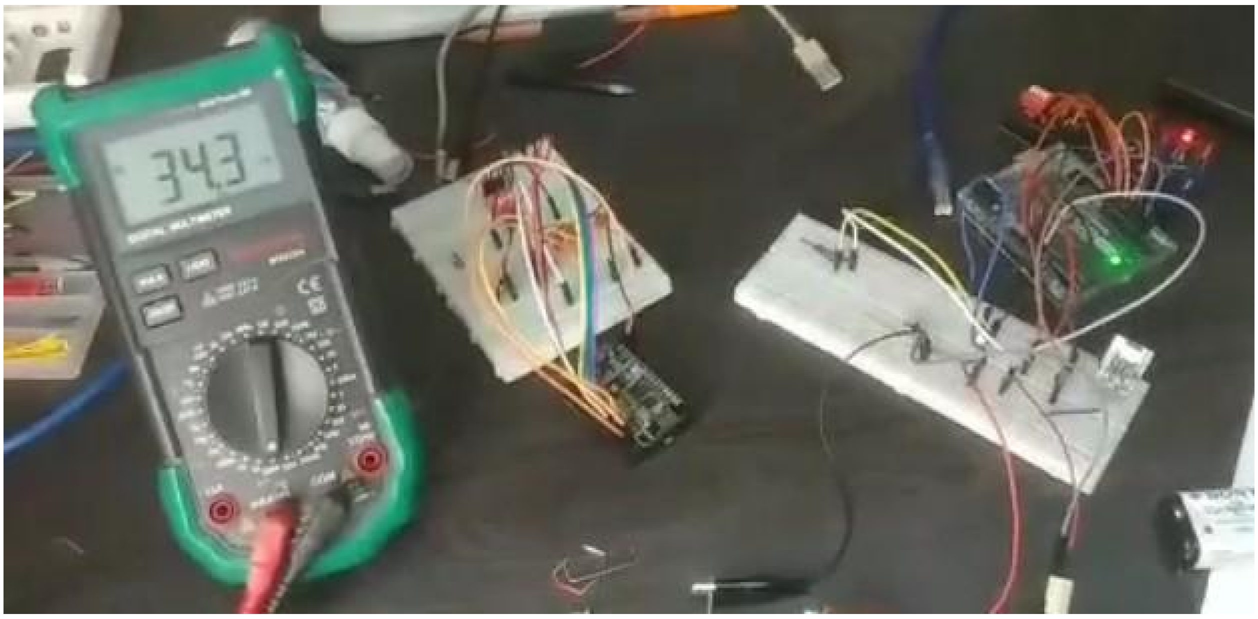


Circuit 5.3.2: Implementation of NodeMCU V2 for input **1111**

5.4 Arduino Uno R3 (Deep Sleep) Implementation



Circuit 5.4.1: Implementation of Arduino Uno R3 (Deep Sleep) output side current for 0000



Circuit 5.4.2: Implementation of Arduino Uno R3 (Deep Sleep) output side current for 1111

6. Data Collection & Graphical Analysis

6.1 Data Collection Procedure

1. Since we are powering a fixed 9V constant power supply using a rechargeable battery, the power can be found by measuring sure the current using the formula:

$$\boldsymbol{P = V \times I}.$$

2. For rigorous analysis, current is measured in different situations and positions (explained below).
3. The most important focus was given to the fact that the experimental environment remains constant for all the boards.

6.2 Input Side Power Consumption

Arduino Uno R3 Input Side Current & Power Consumption

Input Logic	Output Logic	Current (mA)	Avg. Current (mA)	Voltage (V)	Power (mW)
0000	1101	43.1	43.2	9	388.8
		43.2			
		43.2			
1000	1011	43.3	43.3	9	389.7
		43.2			
		43.3			
0100	0101	42.6	42.6	9	383.4
		42.5			
		42.6			
1100	1011	43.6	43.6	9	392.4
		43.5			
		43.6			
0010	0000	40.4	48.4	9	435.6
		40.5			
		40.4			
1010	1011	43.6	43.6	9	392.4
		43.5			
		43.6			
0110	1000	42.3	42.3	9	380.7
		42.2			
		42.3			
1110	0110	43.1	43.1	9	387.9
		43.2			
		43.1			
0001	0001	42.0	42.0	9	378.0
		42.1			
		42.0			

1001	0100	42.4	42.4	9	381.6
		42.3			
		42.4			
0101	1000	42.4	42.4	9	381.6
		42.3			
		42.4			
1101	1011	44.0	44.0	9	396.0
		44.1			
		44.0			
0011	0011	42.8	42.8	9	385.2
		42.7			
		42.8			
1011	1110	44.1	44.1	9	396.9
		44.2			
		44.1			
0111	1001	43.4	43.4	9	390.6
		43.3			
		43.4			
1111	0010	42.8	42.8	9	385.2
		42.7			
		42.8			

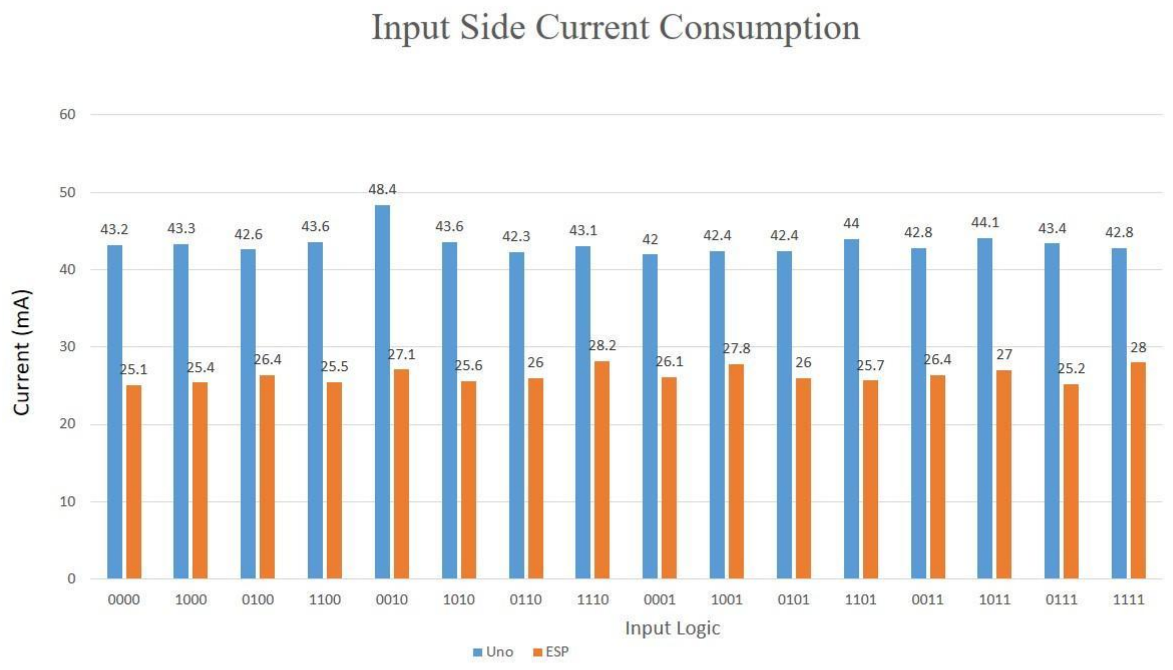
Table 6.3.1: Arduino Uno R3 Input Side Current & Power Consumption

ESP8266 NodeMCU V2 Input Side Current & Power Consumption					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	25.1	25.1	9	225.9
		25.2			
		25.1			
1000	1011	25.4	25.4	9	228.6
		25.3			
		25.4			
0100	0101	26.4	26.4	9	237.6
		26.3			
		26.4			
1100	1011	25.5	25.5	9	229.5
		25.4			
		25.5			
0010	0000	27.1	27.1	9	243.9
		27.2			
		27.1			
1010	1011	25.6	25.6	9	230.4
		25.5			
		25.6			
0110	1000	26.0	26.0	9	234.0

		26.1			
		26.0			
1110	0110	28.2	28.2	9	253.8
		28.3			
		28.2			
0001	0001	26.1	26.1	9	234.9
		26.2			
		26.1			
1001	0100	27.8	27.8	9	250.2
		27.7			
		27.8			
0101	1000	26.0	26.0	9	234.0
		26.1			
		26.0			
1101	1011	25.7	25.7	9	231.2
		25.6			
		25.7			
0011	0011	26.4	26.4	9	237.6
		26.3			
		26.4			
1011	1110	27.0	27.0	9	243.0
		27.1			
		27.0			
0111	1001	25.2	25.2	9	226.8
		25.1			
		25.2			
1111	0010	28.0	28.0	9	252.0
		28.1			
		28.0			

Table 6.3.2: ESP8266 NodeMCU V2 Input Side Current & Power Consumption

Graphical Analysis:



Graph 6.3.3: Input Side Current Consumption Comparison between the boards

- Here we can see that **Arduino Uno** has the **highest power consumption** in this case.
- We initially thought NodeMCU will consume the highest power since it had much higher frequency than Arduino UNO however, we can see NodeMCU V2 consumes much low currents in this case.
- Maximum current and power consumption in Arduino UNO was for the input logic 0010 and minimum for 0001 and for NodeMCU V2 the maximum consumption was for the input 1110 and minimum for 0000.

7. Question & Answers

7.1 Arduino Uno R3

1. **What is the clock frequency of the microcontroller used?**
   Answer: 16 MHz.

2. **What is the data bus width of the microcontroller used?**
   Answer: 8 bit.

3. **What is the size of your hex file generated? Attach the hex codes in your report.**
   Answer: 8.53 KB. Hex file is attached in Appendix D.

4. **Can the project be implemented by using interrupt?**
   Answer: Yes, since all the pins of the Arduino Uno R3 can be used as interrupts.

5. **Is the main routine required to be an infinite loop? Provide explanation in favor of your answer.**
   Answer: Yes the main routine is required to be an infinite loop. This is because we need to continuously check the input status and at the same time provide a constant output logic all the time. If it were not an infinite loop the program would execute and the finish and after that if you change the input logic no output will be given.

6. **Is there any difference between level triggered and edge triggered operation for the given project?**
   Answer: Since we are using a toggle switch, there is no difference between level triggering and edge triggering.

7. **Is the project referring encryption or decryption from input to output?**
   Answer: This is like an encryption circuit. We are applying an encryption algorithm to an information we have. This algorithm can be used to decrypt and find the original information using the appropriate decryption key.

7.2 NodeMCU V2

1. **What is the clock frequency of the microcontroller used?**
   Answer: 80 MHz.

2. **What is the data bus width of the microcontroller used?**
   Answer: 32 bit.

3. **What is the size of your hex file generated? Attach the hex codes in your report.**
   Answer: We were not able to generate the hex file for this board in the Arduino IDE.

4. **Can the project be implemented by using interrupt?**
   Answer: Yes, pin D0-D8 support interrupt for the NodeMCU. So it can be used.

5. **Is the main routine required to be an infinite loop? Provide explanation in favor of your answer.**
   Answer: Yes the main routine is required to be an infinite loop. This is because we need to continuously check the input status and at the same time provide a constant output logic all the time. If it were not an infinite loop the program would execute and the finish and after that if you change the input logic no output will be given.

6. **Is there any difference between level triggered and edge triggered operation for the given project?**
   Answer: Since we are using a toggle switch, there is no difference between level triggering and edge triggering.

7. **Is the project referring encryption or decryption from input to output?**
   Answer: This is like an encryption circuit. We are applying an encryption algorithm to an information we have. This algorithm can be used to decrypt and find the original information using the appropriate decryption key.

8. References

1. Project materials provided by Dr. Dihan Md. Nuruddin Hassan
2. https://www.elprocus.com/what-is-arduino-uno-r3-pin-diagram-specificationand-applications/
3. https://www.elprocus.com/esp8266-wi-fi-module/
4. https://www.make-it.ca/nodemcu-details-specifications/
5. https://circuitdigest.com/microcontroller-projects/arduino-interrupt-tutorialwith-examples
6. https://comparecamp.com/arduino-ide-review-pricing-pros-cons-features/
7. https://en.wikipedia.org/wiki/Proteus_Design_Suite

Appendix A: Software Specifications

Arduino IDE

Arduino IDE is an open-source tool that makes it possible for users to write as well as upload code to a work environment in real-time. Since the written code will be moved to the cloud, it's frequently used by those who need an additional level of redundancy. Arduino IDE offers full compatibility to any Arduinobased software board. The software can easily be deployed in any Linux, Mac, or Windows operating systems. Most of its parts are written within JavaScript for seamless compilation and editing. While the tool's main aim is based on code writing, it offers several noteworthy functionalities. For instance, Arduino IDE lets users share important project information to company stakeholders. Users are given the freedom to make internal layouts and schematic modifications when needed. Comprehensive guides are available for those who need help in the installation process. Tutorials are present for users who have little experience dealing with the tool's framework. Arduino IDE is highly rated by users for its ease of use. It can conduct complex processes while keeping computer resources to a minimum. The tool makes it easy for users to access their libraries. At the same time, it offers updated support for the latest Arduino boards, which can help users with their sketches using the latest IDE version.

Website: https://www.arduino.cc/en/software

Proteus 8 Professional

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.

Website: https://www.labcenter.com/downloads/

LogiSim

When learning computer architecture and logic circuits, you will need a realworld, graphical example of what you are studying. Text and diagrams only go so far. A helpful tool for designing and simulating logic circuits is Logisim.

Because the tool lets you create large circuits from smaller circuits, you can design entire CPUs using Logisim. Further, the tool will run on any computer!

The interface itself is very intuitive and the use of color-coding of wires and elements allows for easy analysis and testing of circuits. You can also save the completed file as an image, or as a .circ file (core to Logisim). Website: http://www.cburch.com/logisim/download.html

Appendix B: Hardware Specification

Arduino Uno R3

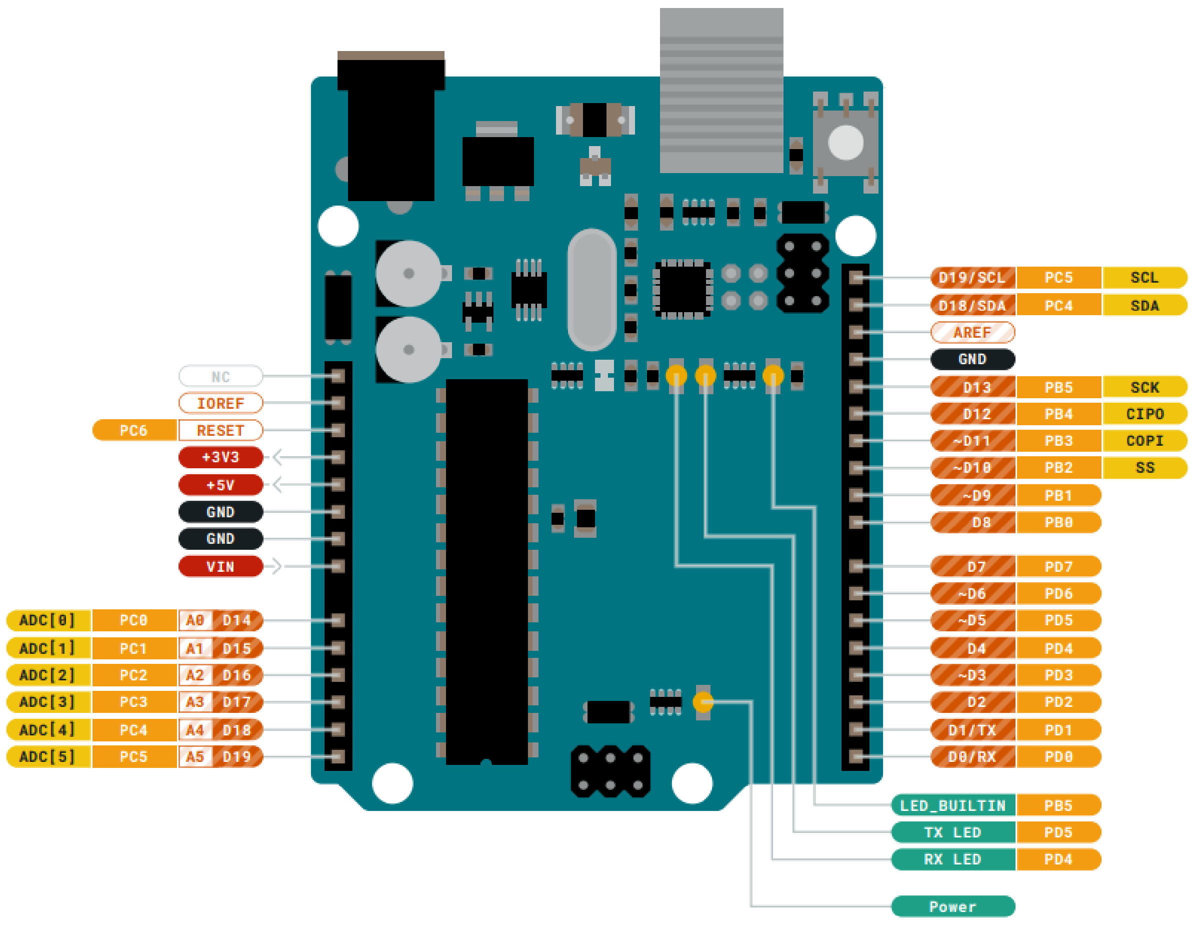
The Arduino UNO R3 is frequently used microcontroller board in the family of an Arduino. This is the latest third version of an Arduino board and released in the year 2011. The main advantage of this board is if we make a mistake we can change the microcontroller on the board. The main features of this board mainly include, it is available in DIP (dual-inline-package), detachable and ATmega328 microcontroller. The programming of this board can easily be loaded by using an Arduino computer program. This board has huge support from the Arduino community, which will make a very simple way to start working in embedded electronics, and many more applications. Please refer the link to know about Arduino – Basics, and Design.

Arduino Uno R3 Specifications

The Arduino Uno R3 board includes the following specifications.

- It is an ATmega328P based Microcontroller
- The Operating Voltage of the Arduino is 5V
- The recommended input voltage ranges from 7V to 12V
- The i/p voltage (limit) is 6V to 20V
- Digital input and output pins-14
- Digital input & output pins (PWM)-6
- Analog i/p pins are 6
- DC Current for each I/O Pin is 20 mA
- DC Current used for 3.3V Pin is 50 mA
- Flash Memory -32 KB, and 0.5 KB memory is used by the boot loader
- SRAM is 2 KB
- EEPROM is 1 KB
- The speed of the CLK is 16 MHz
- In Built LED
- Length and width of the Arduino are 68.6 mm X 53.4 mm
- The weight of the Arduino board is 25 g

Pin Diagram:


NC
IOREF
PC6
RESET
+3V3
+5V
GND
GND
VIN
ADC[0] PC0 A0 D14
ADC[1] PC1 A1 D15
ADC[2] PC2 A2 D16
ADC[3] PC3 A3 D17
ADC[4] PC4 A4 D18
ADC[5] PC5 A5 D19
D19/SCL PC5 SCL
D18/SDA PC4 SDA
AREF
GND
D13 PB5 SCK
D12 PB4 CIPO
~D11 PB3 COPI
~D10 PB2 SS
~D9 PB1
D8 PB0
D7 PD7
~D6 PD6
~D5 PD5
D4 PD4
~D3 PD3
D2 PD2
D1/TX PD1
D0/RX PD0
LED_BUILTIN PB5
TX LED PD5
RX LED PD4
Power

NodeMCU V2

An ESP8266 Wi-Fi module is a SOC microchip mainly used for the development of end-point IoT (Internet of things) applications. It is referred to as a standalone wireless transceiver, available at a very low price. It is used to enable the internet connection to various applications of embedded systems.

Espressif systems designed the ESP8266 Wi-Fi module to support both the TCP/IP capability and the microcontroller access to any Wi-Fi network. It provides the solutions to meet the requirements of industries of IoT such as cost, power, performance, and design.

It can work as either a slave or a standalone application. If the ESP8266 Wi-Fi runs as a slave to a microcontroller host, then it can be used as a Wi-Fi adaptor to any type of microcontroller using UART or SPI. If the module is used as a standalone application, then it provides the functions of the microcontroller and Wi-Fi network.

The ESP8266 Wi-Fi module is highly integrated with RF balun, power modules, RF transmitter and receiver, analog transmitter and receiver, amplifiers, filters, digital baseband, power modules, external circuitry, and other necessary components. The ESP8266 Wi-Fi module is a microchip shown in the figure below.

A set of AT commands are needed by the microcontroller to communicate with the ESP8266 Wi-Fi module. Hence it is developed with AT commands software to allow the Arduino Wi-Fi functionalities, and also allows loading various software to design the own application on the memory and processor of the module.

The processor of this module is based on the Tensilica Xtensa Diamond Standard 106 micro and operates easily at 80 MHz. There are different types of ESP modules designed by third-party manufacturers. They are,

ESP8266-01 designed with 8 pins (GPIO pins -2)

ESP8266-02 designed with 8 pins (GPIO pins -3)

ESP8266-03 designed with 14 pins (GPIO pins- 7)

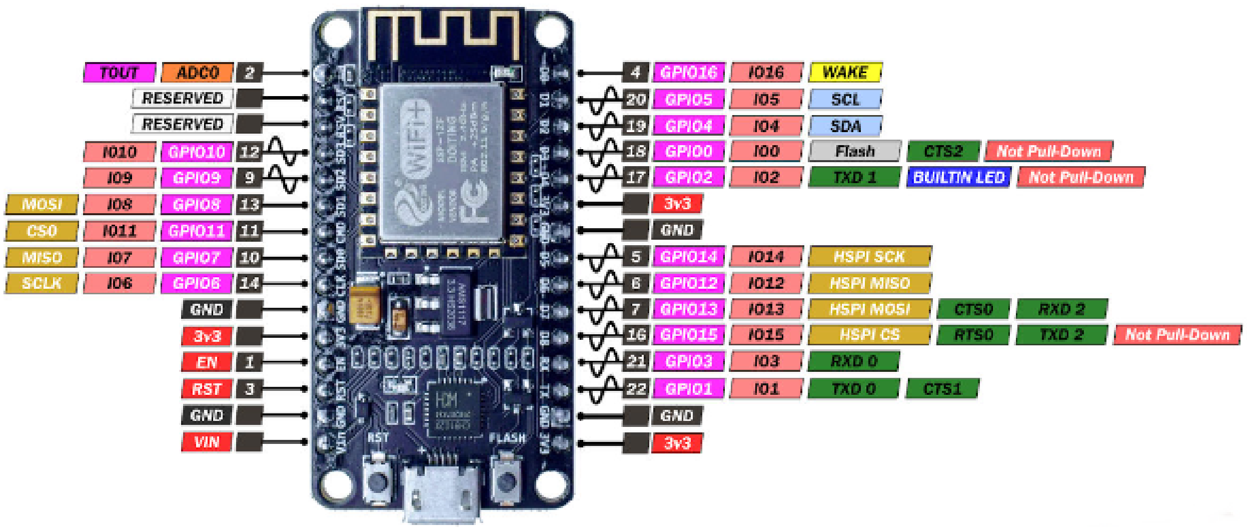
ESP8266-04 designed with 14 pins (GPIO pins- 7)

The ESP8266 Wi-Fi module comes with a boot ROM of 64 KB, user data RAM of 80 KB, and instruction RAM of 32 KB. It can support 802.11 b/g/n Wi-Fi network at 2.4 GHz along with the features of I2C, SPI, I2C interfacing with DMA, and 10-bit ADC. Interfacing this module with the microcontroller can be done easily through a serial port. An external voltage converter is required only if the operating voltage exceeds 3.6 Volts. It is most widely used in robotics and IoT applications due to its low cost and compact size.

NodeMCU V2 Specifications

- It is a powerful Wi-Fi module available in a compact size at a very low price.
- It is based on the L106 RISC 32-bit microprocessor core and runs at 80 MHz
- It requires only 3.3 Volts power supply
- The current consumption is 100 m Amps
- The maximum Input/Output (I/O) voltage is 3.6 Volts. It consumes 100 mA current
- The maximum Input/Output source current is 12 mA
- The frequency of built-in low power 32-bit MCU is 80 MHz
- The size of flash memory is 513 kb
- It is used as either an access point or station or both
- It supports less than 10 microAmps deep sleep
- It supports serial communication to be compatible with several developmental platforms such as Arduino
- It is programmed using either AT commands, Arduino IDE, or Lua script
- It is a 2.4 GHz Wi-Fi module and supports WPA/WPA2, WEP authentication, and open networks.
- It uses two serial communication protocols like I2C (Inter-Integrated Circuit) and SPI (Serial Peripheral Interface). It provides 10- bit analog to digital conversion
- The type of modulation is PWM (Pulse Width Modulation)
- UART is enabled on dedicated pins and for only transmission, it can be enabled on GPIO2.
- GPIO pins – 17
- Memory Size of instruction RAM – 32 KB
- The memory size of instruction cache RAM – 32 KB
- Size of User-data RAM- 80 KB
- Size of ETS systems-data RAM – 16 KB

Pin Diagram:


TOUT ADC0 2
RESERVED
RESERVED
IO10 GPIO10 12
IO9 GPIO9 9
MOSI IO8 GPIO8 13
CS0 IO11 GPIO11 11
MISO IO7 GPIO7 10
SCLK IO6 GPIO6 14
GND
3v3
EN 1
RST 3
GND
VIN
4 GPIO16 IO16 WAKE
20 GPIO5 IO5 SCL
19 GPIO4 IO4 SDA
18 GPIO0 IO0 Flash CTS2 Not Pull-Down
17 GPIO2 IO2 TXD 1 BUILTIN LED Not Pull-Down
3v3
GND
5 GPIO14 IO14 HSPI SCK
6 GPIO12 IO12 HSPI MISO
7 GPIO13 IO13 HSPI MOSI CTS0 RXD 2
16 GPIO15 IO15 HSPI CS RTS0 TXD 2 Not Pull-Down
21 GPIO3 IO3 RXD 0
22 GPIO1 IO1 TXD 0 CTS1
GND
3v3
RST
FLASH

Appendix C: Codes

Arduino Uno R3

```
/*Configured for the (gpio) pinout of esp8266 D1 mini (clone) (need a
pulldown registor in D0 to use with sd card module).

Arduino uno pinout is shown below*/


#define in1 11 // switch1

#define in2 10 // switch2

#define in3 9 // switch3

#define in4 8 // switch4


#define out1 5  //led1

#define out2 4  //led2

#define out3 3  //led3

#define out4 2 //led4


boolean val1 = false;      // variable to store the read value

boolean val2 = false;      // variable to store the read value

boolean val3 = false;      // variable to store the read value

boolean val4 = false;      // variable to store the read value


void setup() {

  Serial.begin(9600);

  delay(500);


  pinMode(out1, OUTPUT);  // sets the digital pin 5 as output of
arduino uno

  pinMode(out2, OUTPUT);  // sets the digital pin 4 as output of
arduino uno

  pinMode(out3, OUTPUT);  // sets the digital pin 3 as output of
arduino uno

  pinMode(out4, OUTPUT);  // sets the digital pin 2 as output of
arduino uno
```

```
  pinMode(in4, INPUT);    // sets the digital pin 11 as input of
arduino uno
  pinMode(in3, INPUT);    // sets the digital pin 10 as input of
arduino uno
  pinMode(in2, INPUT);    // sets the digital pin 9 as input of arduino
uno
  pinMode(in1, INPUT);    // sets the digital pin 8 as input of arduino
uno

  digitalWrite(out4, LOW);
  digitalWrite(out3, LOW);
  digitalWrite(out2, LOW);
  digitalWrite(out1, LOW);

  delay(500);
}

void loop() {

  /*Boolean Function Implementation*/

  val1 = (!digitalRead(in3)*!digitalRead(in4))+
         (digitalRead(in1)*digitalRead(in2)*!digitalRead(in3))+
         (!digitalRead(in1)*!digitalRead(in2)*!digitalRead(in3))+
         (digitalRead(in1)*!digitalRead(in2)*!digitalRead(in4))+
         (!digitalRead(in1)*digitalRead(in3)*digitalRead(in4));

  val2 = (digitalRead(in1)*digitalRead(in2))+
         (digitalRead(in1)*!digitalRead(in4))+
         (!digitalRead(in2)*digitalRead(in3)*digitalRead(in4));

  val3 = (digitalRead(in1)*!digitalRead(in2)*digitalRead(in4))+
         (!digitalRead(in1)*!digitalRead(in3)*!digitalRead(in4))+

(digitalRead(in1)*digitalRead(in2)*digitalRead(in3)*!digitalRead(in4));
```

```
  val4 = (digitalRead(in1)*digitalRead(in2)*!digitalRead(in3))+
         (digitalRead(in1)*!digitalRead(in2)*digitalRead(in3))+
         (!digitalRead(in1)*digitalRead(in2)*digitalRead(in3))+
         (digitalRead(in2)*!digitalRead(in3)*digitalRead(in4))+
         (!digitalRead(in2)*!digitalRead(in3)*!digitalRead(in4));

  /* sets the LED to the button's value*/
  digitalWrite(out4, val1);
  delay(10);
  digitalWrite(out3, val2);
  delay(10);
  digitalWrite(out2, val3);
  delay(10);
  digitalWrite(out1, val4);
  delay(10);
}
```

NodeMCU V2

```
/*Configured for the (gpio) pinout of esp8266 D1 mini (clone) (need a
pulldown registor in D0 to use with sd card module).
Arduino uno pinout is shown below*/

#define in1 11 // switch1
#define in2 10 // switch2
#define in3 9 // switch3
#define in4 8 // switch4

#define out1 5  //led1
#define out2 4  //led2
#define out3 3  //led3
#define out4 2 //led4

boolean val1 = false;      // variable to store the read value
boolean val2 = false;      // variable to store the read value
boolean val3 = false;      // variable to store the read value
boolean val4 = false;      // variable to store the read value

void setup() {
  Serial.begin(9600);
  delay(500);
  /*
  pinMode(in1, INPUT_PULLUP);
  pinMode(in2, INPUT_PULLUP);
  pinMode(in3, INPUT_PULLUP);
  pinMode(in4, INPUT_PULLUP);

  pinMode(out1, INPUT_PULLUP);
  pinMode(out2, INPUT_PULLUP);
  pinMode(out3, INPUT_PULLUP);
  pinMode(out4, INPUT_PULLUP);
  */
  pinMode(out1, OUTPUT);  // sets the digital pin 5 as output of
arduino uno
  pinMode(out2, OUTPUT);  // sets the digital pin 4 as output of
arduino uno
  pinMode(out3, OUTPUT);  // sets the digital pin 3 as output of
arduino uno
  pinMode(out4, OUTPUT);  // sets the digital pin 2 as output of
arduino uno


  pinMode(in4, INPUT);    // sets the digital pin 11 as input of
arduino uno
  pinMode(in3, INPUT);    // sets the digital pin 10 as input of
arduino uno
  pinMode(in2, INPUT);    // sets the digital pin 9 as input of arduino
uno
  pinMode(in1, INPUT);    // sets the digital pin 8 as input of arduino
uno


  digitalWrite(out4, LOW);
  digitalWrite(out3, LOW);
  digitalWrite(out2, LOW);
  digitalWrite(out1, LOW);
  /*
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
  */
  delay(500);
}


void loop() {
          /*Boolean Function Implementation*/


  val1 = (!digitalRead(in3)*!digitalRead(in4))+
         (digitalRead(in1)*digitalRead(in2)*!digitalRead(in3))+
         (!digitalRead(in1)*!digitalRead(in2)*!digitalRead(in3))+
         (digitalRead(in1)*!digitalRead(in2)*!digitalRead(in4))+
         (!digitalRead(in1)*digitalRead(in3)*digitalRead(in4));

  val2 = (digitalRead(in1)*digitalRead(in2))+
         (digitalRead(in1)*!digitalRead(in4))+
         (!digitalRead(in2)*digitalRead(in3)*digitalRead(in4));

  val3 = (digitalRead(in1)*!digitalRead(in2)*digitalRead(in4))+
         (!digitalRead(in1)*!digitalRead(in3)*!digitalRead(in4))+
(digitalRead(in1)*digitalRead(in2)*digitalRead(in3)*!digitalRead(in4));

  val4 = (digitalRead(in1)*digitalRead(in2)*!digitalRead(in3))+
         (digitalRead(in1)*!digitalRead(in2)*digitalRead(in3))+
         (!digitalRead(in1)*digitalRead(in2)*digitalRead(in3))+
         (digitalRead(in2)*!digitalRead(in3)*digitalRead(in4))+
         (!digitalRead(in2)*!digitalRead(in3)*!digitalRead(in4));

  /* sets the LED to the button's value*/
  digitalWrite(out4, val1);
  delay(10);
  digitalWrite(out3, val2);
  delay(10);
  digitalWrite(out2, val3);
  delay(10);
  digitalWrite(out1, val4);
  delay(10);
}
```

Arduino Uno R3 (Deep Sleep)

```
#define LED_PIN 3

/*Arduino uno pinout is shown below*/

#define in1 11 // switch1
#define in2 10 // switch2
#define in3 9 // switch3
#define in4 8 // switch4

#define out1 7  //led1
#define out2 6  //led2
#define out3 5  //led3
#define out4 4 //led4

boolean val1 = false;      // variable to store the read value
boolean val2 = false;      // variable to store the read value
boolean val3 = false;      // variable to store the read value
boolean val4 = false;      // variable to store the read value

void setup() {
  //Save Power by writing all Digital IO LOW - note that pins just need
to be tied one way or another, do not damage devices!
  //for (int i = 0; i < 20; i++) {
  //if(i != 2)//just because the button is hooked up to digital pin 2
  pinMode(i, OUTPUT);
  //}

  attachInterrupt(0, digitalInterrupt, FALLING); //interrupt for waking
up
  pinMode(LED_PIN, OUTPUT);

  pinMode(out1, OUTPUT);  // sets the digital pin 5 as output of
arduino uno
```

```
  pinMode(out2, OUTPUT);  // sets the digital pin 4 as output of
arduino uno
  pinMode(out3, OUTPUT);  // sets the digital pin 3 as output of
arduino uno
  pinMode(out4, OUTPUT);  // sets the digital pin 2 as output of
arduino uno


  pinMode(in4, INPUT);    // sets the digital pin 11 as input of
arduino uno
  pinMode(in3, INPUT);    // sets the digital pin 10 as input of
arduino uno
  pinMode(in2, INPUT);    // sets the digital pin 9 as input of arduino
uno
  pinMode(in1, INPUT);    // sets the digital pin 8 as input of arduino
un




  //Disable ADC - don't forget to flip back after waking up if using
ADC in your application ADCSRA |= (1 << 7);
  ADCSRA &= ~(1 << 7);
  //SETUP WATCHDOG TIMER
  WDTCSR = (24);//change enable and WDE - also resets
  WDTCSR = (33);//prescalers only - get rid of the WDE and WDCE bit
  WDTCSR |= (1<<6);//enable interrupt mode
  //ENABLE SLEEP - this enables the sleep mode
  SMCR |= (1 << 2); //power down mode
  SMCR |= 1;//enable sleep


}


void loop() {
  //attachInterrupt(0, digitalInterrupt, FALLING); //interrupt for
waking up
```

```
  digitalWrite(LED_PIN, HIGH);
  delay(1000);
  digitalWrite(LED_PIN, LOW);

  //Enable ADC converter
  //ADCSRA |= (1 << 7);
  /*Boolean Function Implementation*/
  val1 = (!digitalRead(in3)*!digitalRead(in4))+
         (digitalRead(in1)*digitalRead(in2)*!digitalRead(in3))+
         (!digitalRead(in1)*!digitalRead(in2)*!digitalRead(in3))+
         (digitalRead(in1)*!digitalRead(in2)*!digitalRead(in4))+
         (!digitalRead(in1)*digitalRead(in3)*digitalRead(in4));

  val2 = (digitalRead(in1)*digitalRead(in2))+
         (digitalRead(in1)*!digitalRead(in4))+
         (!digitalRead(in2)*digitalRead(in3)*digitalRead(in4));

  val3 = (digitalRead(in1)*!digitalRead(in2)*digitalRead(in4))+
         (!digitalRead(in1)*!digitalRead(in3)*!digitalRead(in4))+
(digitalRead(in1)*digitalRead(in2)*digitalRead(in3)*!digitalRead(in4));

  val4 = (digitalRead(in1)*digitalRead(in2)*!digitalRead(in3))+
         (digitalRead(in1)*!digitalRead(in2)*digitalRead(in3))+
         (!digitalRead(in1)*digitalRead(in2)*digitalRead(in3))+
         (digitalRead(in2)*!digitalRead(in3)*digitalRead(in4))+
         (!digitalRead(in2)*!digitalRead(in3)*!digitalRead(in4));


   /* sets the LED to the button's value*/
  digitalWrite(out4, val1);
  digitalWrite(out3, val2);
  digitalWrite(out2, val3);
  digitalWrite(out1, val4);
```

```
    for(int i=0; i<1; i++){

    //Save Power by writing all Digital IO LOW - note that pins just
need to be tied one way or another, do not damage devices!

    for (int i = 0; i < 20; i++) {

    if(i != 2)//just because the button is hooked up to digital pin 2

    pinMode(i, OUTPUT);

    }


    //BOD DISABLE - this must be called right before the __asm__ sleep
instruction

    MCUCR |= (3 << 5); //set both BODS and BODSE at the same time

    MCUCR = (MCUCR & ~(1 << 5)) | (1 << 6); //then set the BODS bit and
clear the BODSE bit at the same time

    __asm__  __volatile__("sleep");//in line assembler to go to sleep


    }

}


void digitalInterrupt(){

  //needed for the digital input interrupt

}


ISR(WDT_vect){

  //DON'T FORGET THIS!  Needed for the watch dog timer.  This is called
after a watch dog timer timeout - this is the interrupt function called
after waking up

}// watchdog interrupt
```

Hex Codes:

Arduino Uno R3:

```
:100000000C9462000C948A000C948A000C948A0070
:100010000C948A000C948A000C948A000C948A0038
:100020000C948A000C948A000C948A000C948A0028
:100030000C948A000C948A000C948A000C948A0018
:100040000C944A020C948A000C94BA020C94940208
:100050000C948A000C948A000C948A000C948A00F8
:100060000C948A000C948A00000000000024002700F1
:100070002A0000000000250028002B0000000000DE
:1000800023002600290004040404040404040202DA
:100090000202020203030303030301020408102007
:1000A0004080010204081020010204081020000012
:1000B0000000080002010000030407000000000027
:1000C0000000000CD0511241FBECFEFD8E0DEBFCDBFAD
:1000D00011E0A0E0B1E0E4E0FCE002C005900D9288
:1000E000A231B107D9F721E0A2E1B1E001C01D9230
:1000F000AB3BB207E1F710E0C2E6D0E004C02197C5
:10010000FE010E94FA05C136D107C9F70E94EC0230
:100110000C9400060C940000AF92BF92CF92DF9235
:10012000EF92FF920F931F93CF93DF936C017B01AC
:100130008B01040F151FEB015E01AE18BF08C0173D
:10014000D10759F06991D601ED91FC910190F081B0
:10015000E02DC6010995892B79F7C501DF91CF9173
:100160001F910F91FF90EF90DF90CF90BF90AF90D5
:100170000895FC01538D448D252F30E0842F90E0AD
:10018000821B930B541710F0CF9608950197089592
:10019000FC01918D828D981761F0A28DAE0FBF2F5B
:1001A000B11D5D968C91928D9F5F9F73928F90E0B1
:1001B0008958FEF9FEF0895FC01918D828D981720
:1001C00031F0828DE80FF11D858D90E008958FEF5D
```

```
:1001D0009FEF0895FC01918D228D892F90E0805C26

:1001E0009F4F821B91098F739927089582E191E0B7

:1001F0000E94EA0021E0892B09F420E0822F089573

:1002000080E090E0892B29F00E94F60081110C9487

:1002100000000895FC01A48DA80FB92FB11DA35AA9

:10022000BF4F2C91848D90E001968F739927848F16

:10023000A689B7892C93A089B1898C918370806439

:100240008C93938D848D981306C00288F389E02DDA

:1002500080818F7D80830895EF92FF920F931F938B

:10026000CF93DF93EC0181E0888F9B8D8C8D981369

:100270001AC0E889F989808185FF15C09FB7F89475

:10028000EE89FF896083E889F989808183708064C1

:1002900080839FBF81E090E0DF91CF911F910F910C

:1002A000FF90EF900895F62E0B8D10E00F5F1F4F1B

:1002B0000F731127E02E8C8D8E110CC00FB607FC2A

:1002C000FACFE889F989808185FFF5CFCE010E94B8

:1002D0000A01F1CFEB8DEC0FFD2FF11DE35AFF4F1B

:1002E000F0829FB7F8940B8FEA89FB898081806246

:1002F000CFCFCF93DF93EC01888D8823B9F0AA8903

:10030000BB89E889F9898C9185FD03C0808186FDD0

:100310000DC00FB607FCF7CF8C9185FFF2CF80811F

:1003200085FFEDCFCE010E940A01E9CFDF91CF9189

:1003300008958333081F028F4813099F08230A9F05B

:100340008958730A9F08830C9F08430B1F48091E5

:1003500080008F7D03C0809180008F778093800024

:10036000089584B58F7784BD089584B58F7DFBCFC4

:100370008091B0008F778093B00008958091B00095

:100380008F7DF9CFCF93DF93282F30E0F901E2552D

:10039000FF4F8491F901E656FF4FD491F901EA57D6

:1003A000FF4FC491CC23A1F081110E949901EC2F41

:1003B000F0E0EE0FFF1FE458FF4FA591B491EC91D0

:1003C000ED2381E090E009F480E0DF91CF91089582
```

```
:1003D00080E090E0FACF1F93CF93DF93282F30E097

:1003E000F901E255FF4F8491F901E656FF4FD49190

:1003F000F901EA57FF4FC491CC23A9F0162F8111C0

:100400000E949901EC2FF0E0EE0FFF1FEE58FF4F16

:10041000A591B4918FB7F894EC91111108C0D095C3

:10042000DE23DC938FBFDF91CF911F910895DE2BE8

:10043000F8CFCF93DF9390E0FC01E656FF4F249175

:100440008A579F4FFC0184918823D1F090E0880F58

:10045000991FFC01E859FF4FA591B491FC01EE589A

:10046000FF4FC591D49161110EC09FB7F8948C9144

:10047000E22FE0958E238C932881E223E8839FBFAF

:10048000DF91CF9108958FB7F894EC91E22BEC9324

:100490008FBFF6CF1F920F920FB60F9211242F939A

:1004A0003F938F939F93AF93BF938091B7019091A8

:1004B000B801A091B901B091BA013091B60123E021

:1004C000230F2D3758F50196A11DB11D2093B601BC

:1004D0008093B7019093B801A093B901B093BA018A

:1004E0008091B2019091B301A091B401B091B50196

:1004F0000196A11DB11D8093B2019093B301A09309

:10050000B401B093B501BF91AF919F918F913F918D

:100510002F910F900FBE0F901F90189526E8230F74

:100520000296A11DB11DD2CF1F920F920FB60F924E

:1005300011242F933F934F935F936F937F938F93E8

:100540009F93AF93BF93EF93FF9382E191E00E945B

:100550000A01FF91EF91BF91AF919F918F917F9190

:100560006F915F914F913F912F910F900FBE0F9020

:100570001F9018951F920F920FB60F9211242F9370

:100580008F939F93EF93FF93E0912201F0912301CA

:100590008081E0912801F091290182FD1BC09081AA

:1005A00080912B018F5F8F7320912C01821741F076

:1005B000E0912B01F0E0EE5EFE4F958F80932B01D2

:1005C000FF91EF919F918F912F910F900FBE0F9000
```

```
:1005D0001F9018958081F4CF789484B5826084BD93

:1005E00084B5816084BD85B5826085BD85B5816037

:1005F00085BD80916E00816080936E0010928100B5

:10060000809181008260809381008091810081606F

:100610008093810080918000816080938000809130

:10062000B10084608093B1008091B00081608093BC

:10063000B00080917A00846080937A0080917A0083

:10064000826080937A0080917A00816080937A0042

:1006500080917A00806880937A001092C100E091C6

:100660002201F091230182E08083E0911E01F0914C

:100670001F011082E0912001F09121018FEC808315

:1006800010922A01E0912601F091270186E08083F3

:10069000E0912401F0912501808180618083E091C7

:1006A0002401F0912501808188608083E0912401FC

:1006B000F0912501808180688083E0912401F09190

:1006C000250180818F7D808361E085E00E94190291

:1006D00061E084E00E94190261E083E00E94190257

:1006E00061E082E00E94190260E08BE00E94190242

:1006F00060E08AE00E94190260E089E00E9419022D

:1007000060E088E00E94190280E0E82E80E0F82E88

:1007100088E00E94C2018C0189E00E94C201089F0A

:10072000E001099FD00D189FD00D112488E00E9490

:10073000C2018C018BE00E94C20121E030E0892BD4

:1007400011F030E020E0209FC001219F900D309FEC

:10075000900D1124C80FD91F89E00E94C2018C019D

:100760008AE00E94C20121E030E0012B11F030E06C

:1007700020E0289F8001299F100D389F100D112423

:100780008BE00E94C2019C01029FC001039F900D5B

:10079000129F900D11248C0F9D1F99249394892BE7

:1007A00009F4912C9092B10188E00E94C201EC0101

:1007B00089E00E94C20121E030E0892B11F030E095

:1007C00020E02C9F80012D9F100D3C9F100D1124C7
```

```
:1007D0008BE00E94C201089FE001099FD00D189F85

:1007E000D00D112488E00E94C2018C018AE00E9491

:1007F000C20141E050E0012B11F050E040E021E067

:1008000030E0892B11F030E020E0429F8001439FCF

:10081000100D529F100D11248BE00E94C20121E0A7

:1008200030E0892B11F030E020E0209FC001219FB3

:10083000900D309F900D1124C80FD91F88E00E94A1

:10084000C2018C0189E00E94C201089F6001099FDA

:10085000D00C189FD00C11248AE00E94C201C89EBF

:100860008001C99E100DD89E100D11248BE00E94AE

:10087000C20121E030E0892B11F030E020E0209F20

:10088000C001219F900D309F900D11248C0F9D1F52

:10089000D1E0892B09F4D0E0D093B00188E00E9428

:1008A000C2018C0189E00E94C201089F6001099F7A

:1008B000D00C189FD00C11248AE00E94C20121E0C4

:1008C00030E0892B11F030E020E02C9D80012D9D3F

:1008D000100D3C9D100D112488E00E94C2015C01A6

:1008E00089E00E94C20121E030E0892B11F030E064

:1008F00020E02A9D60012B9DD00C3A9DD00C112444

:100900008AE00E94C2019C01C29EC001C39E900D5C

:10091000D29E900D1124080F191F88E00E94C20179

:100920006C0189E00E94C20121E030E0CD2811F085

:1009300030E020E0289F6001299FD00C389FD00C28

:1009400011248AE00E94C2019C01C29EC001C39E84

:10095000900DD29E900D1124080F191F89E00E945E

:10096000C2015C018AE00E94C20121E030E0892BD3

:1009700011F030E020E02A9D60012B9DD00C3A9DC3

:10098000D00C11248BE00E94C2019C01C29EC001C8

:10099000C39E900DD29E900D1124080F191F89E05F

:1009A0000E94C2016C018AE00E94C20141E050E055

:1009B000CD2811F050E040E021E030E0892B11F02B

:1009C00030E020E0429F6001439FD00C529FD00C4A
```

```
:1009D00011248BE00E94C20121E030E0892B11F04C
:1009E00030E020E02C9DC0012D9D900D3C9D900D90
:1009F0001124800F911FC1E0892B09F4C0E0C0933E
:100A0000AF018AE00E94C2018C018BE00E94C2010A
:100A100041E050E0012B11F050E040E021E030E0F7
:100A2000892B11F030E020E0429F8001439F100DA0
:100A3000529F100D112488E00E94C2015C0189E0E0
:100A40000E94C201A89E6001A99ED00CB89ED00C45
:100A500011248AE00E94C20121E030E0892B11F0CC
:100A600030E020E02C9DC0012D9D900D3C9D900D0F
:100A70001124080F191F88E00E94C2016C0189E04F
:100A80000E94C20141E050E0CD2811F050E040E06A
:100A900021E030E0892B11F030E020E0429F60013E
:100AA000439FD00C529FD00C11248AE00E94C201B7
:100AB00021E030E0892B11F030E020E02C9DC001D6
:100AC0002D9D900D3C9D900D1124080F191F88E05D
:100AD0000E94C2015C0189E00E94C20121E030E075
:100AE000892B11F030E020E02A9D60012B9DD00C75
:100AF0003A9DD00C11248BE00E94C20121E030E02D
:100B0000892B11F030E020E02C9DC0012D9D900D2F
:100B10003C9D900D1124080F191F88E00E94C2010E
:100B20006C018AE00E94C20121E030E0CD2811F082
:100B300030E020E0289F6001299FD00C389FD00C26
:100B400011248BE00E94C2019C01C29EC001C39E81
:100B5000900DD29E900D1124080F191F61E0012BFA
:100B600009F460E082E00E94EB01692D83E00E94BD
:100B7000EB016D2F84E00E94EB016C2F85E00E9459
:100B8000EB01E114F10409F4C3CD0E94F6008823BF
:100B900009F4BECD0E940000BBCDE2E1F1E013827A
:100BA000128288EE93E0A0E0B0E084839583A68370
:100BB000B78384E091E09183808385EC90E0958712
:100BC000848784EC90E09787868780EC90E0918B17
```

```
:100BD000808B81EC90E0938B828B82EC90E0958B04
:100BE000848B86EC90E0978B868B118E128E138E01
:100BF000148E0895EE0FFF1F0590F491E02D0994D7
:040C0000F894FFCF96
:100C0400000000002C018C00B9007901EA00C80042
:020C1400DC0002
:00000001FF
```

Appendix D: Resources

Logisim Circuit:

https://drive.google.com/file/d/1ttv15-gZJoX4U4tNvVLAhXA-UDtSzq6Q

Proteus Simulation Circuit:

https://drive.google.com/drive/folders/1z0YclDoYtbE4uaSmyheVelC7qNoAdNtS