NORTH SOUTH UNIVERSITY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

SPRING 2022

CSE 331/ EEE 332 / EEE 453

Microprocessor Interfacing & Embedded System

Project Report

'Analysis of Power Consumption of different Microcontrollers'

Faculty: Dr. Dihan Md. Nuruddin Hasan
Section: 4
Group: 2

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

Table of Contents

1. Introduction

1.1 Main Objective

- The main objective of this project was to analyze the power consumption of different microcontroller boards (which are commercially available in the market.) for a special case.
- The **special case** is to apply a **4 bit logic input** in the microcontrollers.
- The microcontroller then would apply an **encryption algorithm** according a given **truth table** (**Table 2.1.1**) and give **a 4 bit output** which can be displayed with **LEDs**.
- The **scope** of this experiment is to observe the current drawn by the circuit for each possible logic applied to the **input** and the **total system**. As for 4 bit input, there are **16 possible inputs** (from 0000 to 1111) and current in the circuit is measured for all these combinations. This is done for a minimum of **3 microcontroller boards** with different specification.
- The **data** is collected for all the boards and a **graphical analysis** was done to build a **comparison** between the three boards.
- To get **valid results** all the input conditions of the microcontrollers must be kept **constant**. For example, a constant power supply to the circuit.

2. Method of Derivation

2.1 Truth Table

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

Table 2.1.1: 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 2.1.1**). So for each output we will get one Boolean expression. The Boolean expression were derive using **K-map**.

The Boolean expressions are:

$$\boldsymbol{O0 = (I1 + I2')(I1' + I2)(I0 + I2' + I3')(I0' + I2' + I3)}$$

$$\boldsymbol{O1 = (I0'.I1'.I2') + (I0'.I1.I2.I3') + (I0.I1.I2'.I3)}$$

$$\boldsymbol{O2 = (I0'.I1) + (I1.I3) + (I0.I2) + (I0.I1'.I3')}$$

$$\boldsymbol{O3 = (I0'.I3') + (I2'.I3') + (I1.I3') + (I0'.I1'.I2') + (I0.I2.I3)}$$

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.

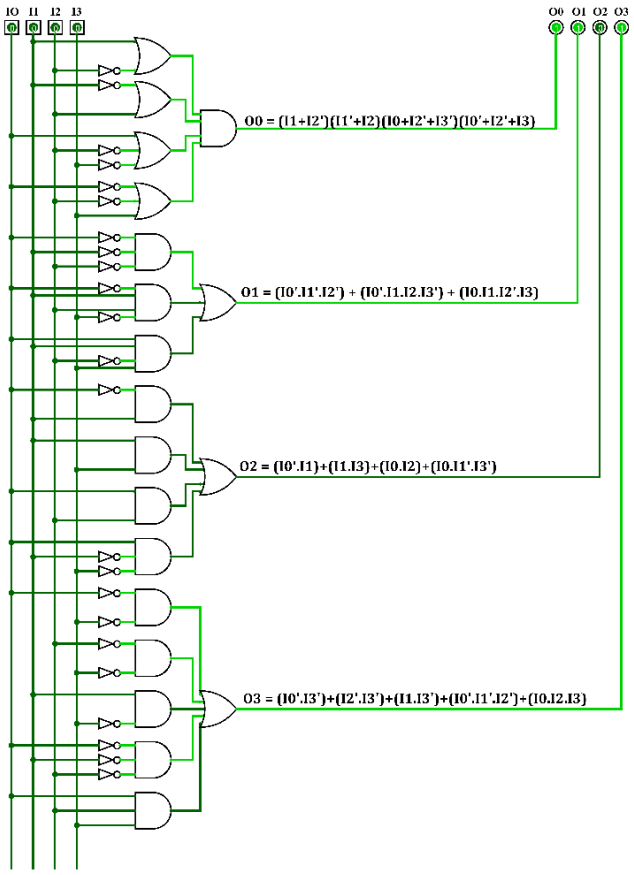


Figure 2.2.1: Logic Circuit for the derived Boolean Expressions.

3. Circuit Design & Simulation

3.1 Circuit Design Procedure

Every commercial microcontroller boards have 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 5V 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** (10 KΩ each) to provide the logic input sequence.
- We connected **4 LEDs** to the 4 digital output pins.
- This procedure is followed for all the microcontroller boards being used for the experiment.

We have done the experiment using 3 microcontroller boards:

1. **Arduino Uno R3 (16 MHz)**
2. **Arduino Nano (16MHz)**
3. **NodeMCU V3 (80MHz)**

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

3.2 Simulation Procedure

Simulation is done using **Proteus 8 Professional** (**Appendix A**). We cannot simulate Arduino directly in Proteus, first we need to install some specific **libraries** which include the simulation libraries for both the Arduino Uno and Nano.

NodeMCU V3 cannot be simulated using Proteus as simulation library for the board is not present.

We have to write our **code** (all codes are given in **Appendix C**) to implement the Boolean expression that we have derived. This code will be converted into a **hex file** by the **Arduino IDE** (**Appendix A**) to be used to simulate the circuit in Proteus.

After performing the simulation of the two boards (Uno & Nano) 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.

Now we will see the logic behind the codes.

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** and the codes for Arduino Uno and Nano are **exactly same**. A small difference is there for NodeMCU V3 in defining the pins.

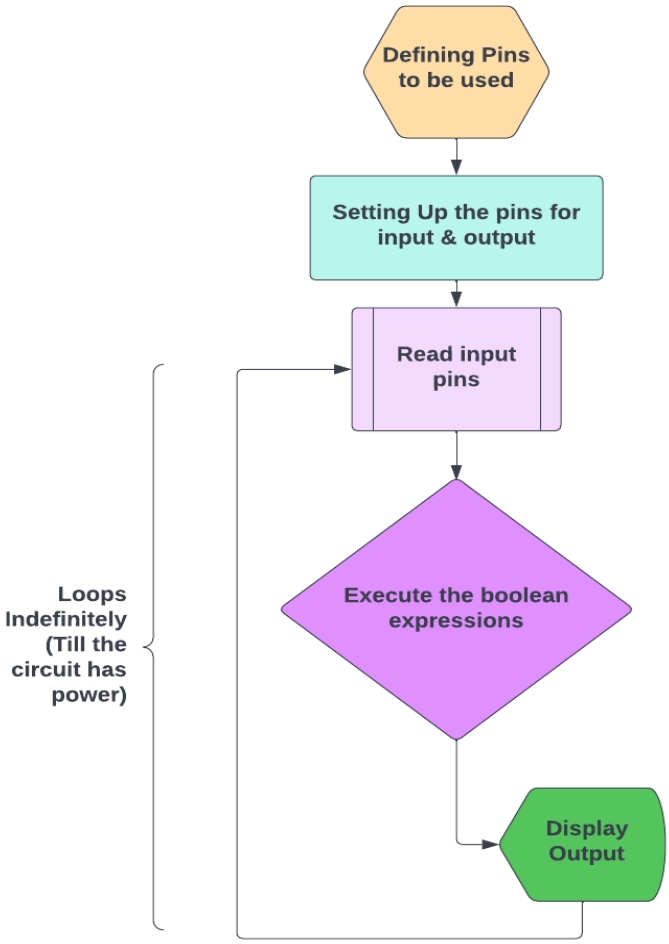
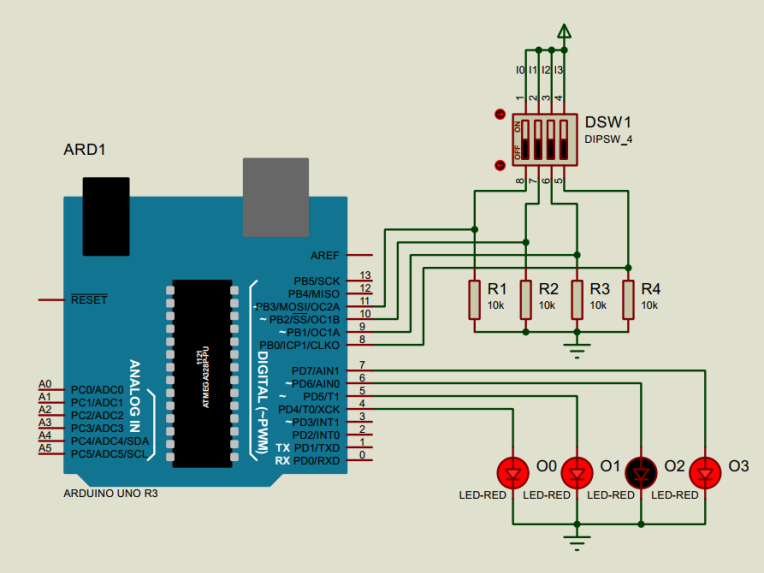


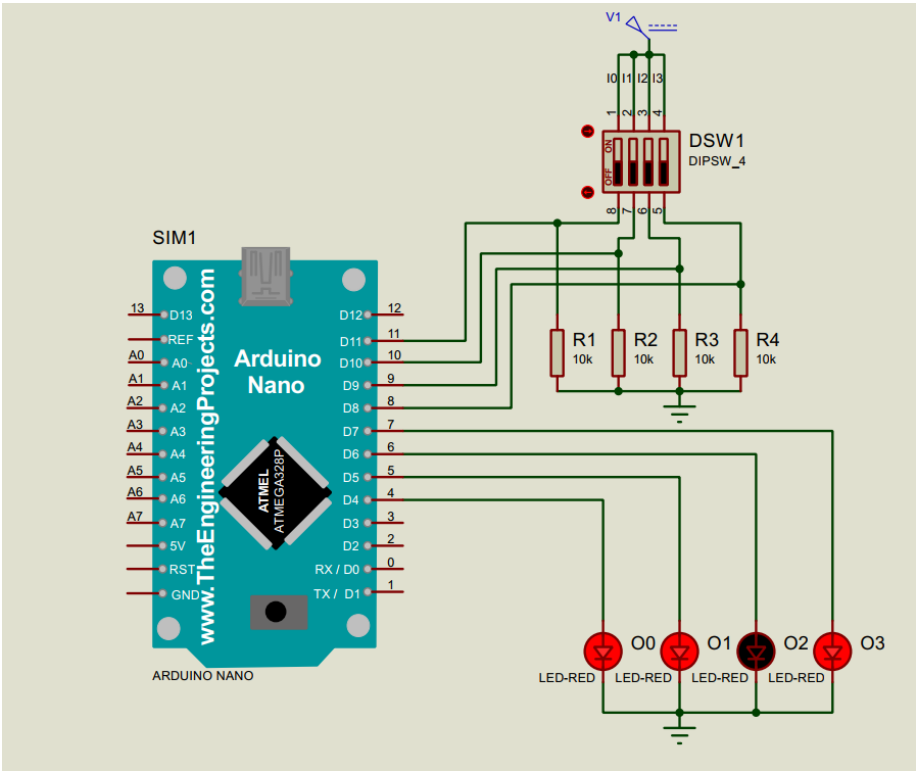
Figure 3.3.1: Program Logic Flow Chart

3.4 Arduino Uno R3 Simulation



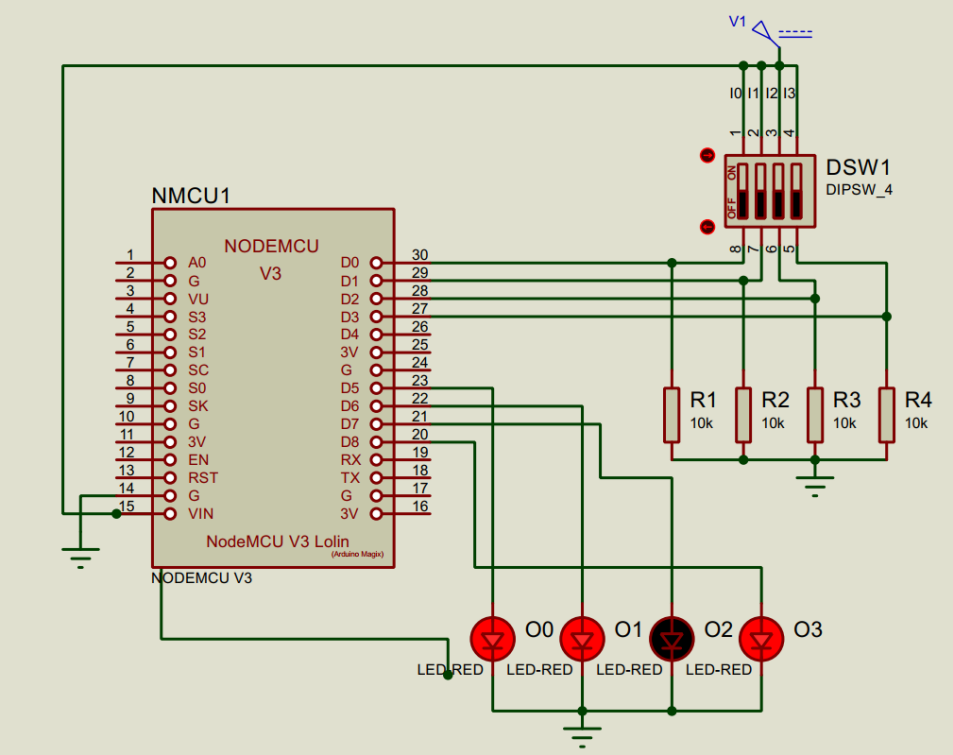
Circuit 3.3.1: Circuit diagram for Arduino Uno R3.

3.5 Arduino Nano Simulation



Circuit 3.4.1: Circuit diagram for Arduino Nano.

3.6 NodeMCU V3 Schematic



Circuit 3.5.1: Circuit diagram for NodeMCU V3.

4. Circuit Operation

4.1 Working Principle

- The whole system is powered by a **5V (V_{cc})** power supply.
- Current from the 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 **10 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 flows into the input pins**. The rest of the current flows through the resistor to ground.
- The output LEDs are connect to 4 GPIO pins of microcontrollers.
- Power delivered in the Vin pins is used to power the LEDs. (**A small amount of current also goes into the board through the input pins**)
- 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.

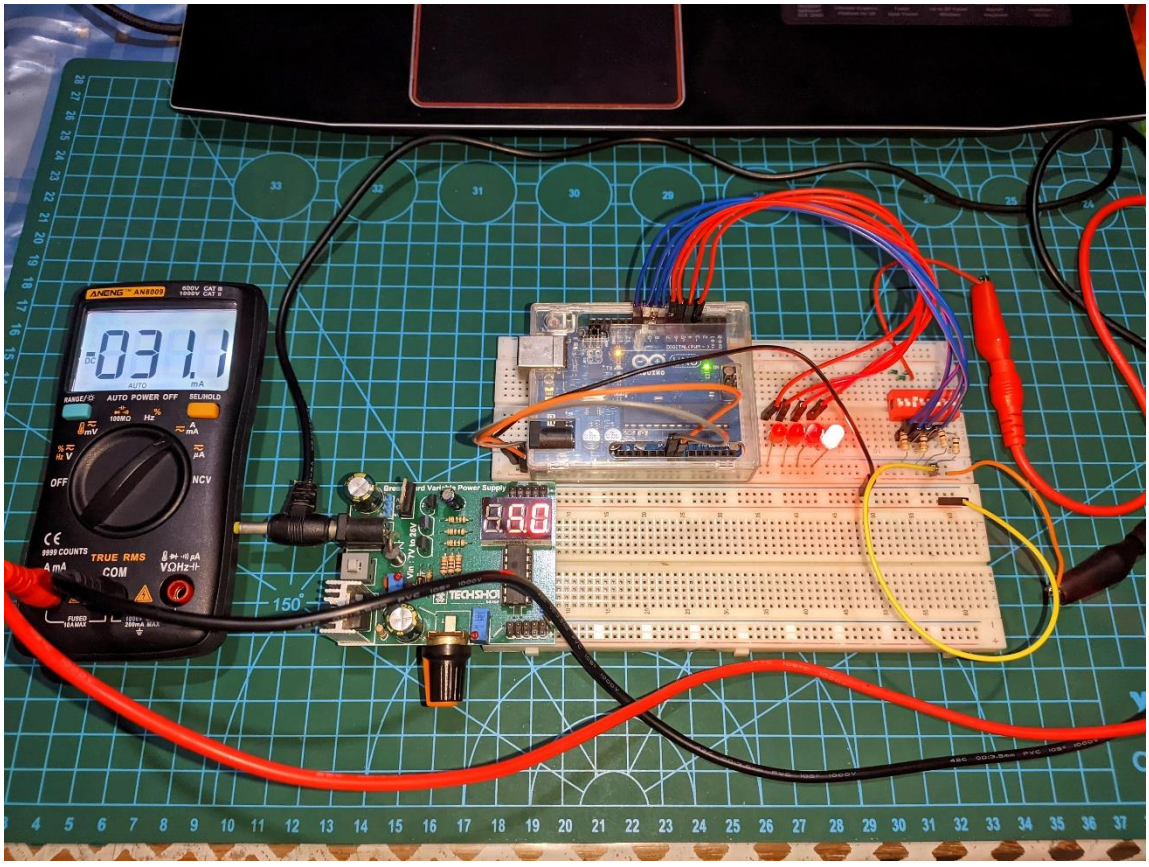
5. Hardware Implementation

5.1 Equipment & Components Used

Serial No.	Name	Quantity
1	Variable DC Breadboard Power Supply	1
2	12 Volt Adapter (To run the PS)	1
3	Breadboard	2
4	8-bit DIP Switch	1
5	Resistor (10 KΩ)	4
6	LED (Red)	4
7	Arduino Uno R3	1
8	Arduino Nano	1
9	NodeMCU V3 (ESP8266)	1
10	ANENG AN8009 Multimeter	1
11	Jumper Wires	As required

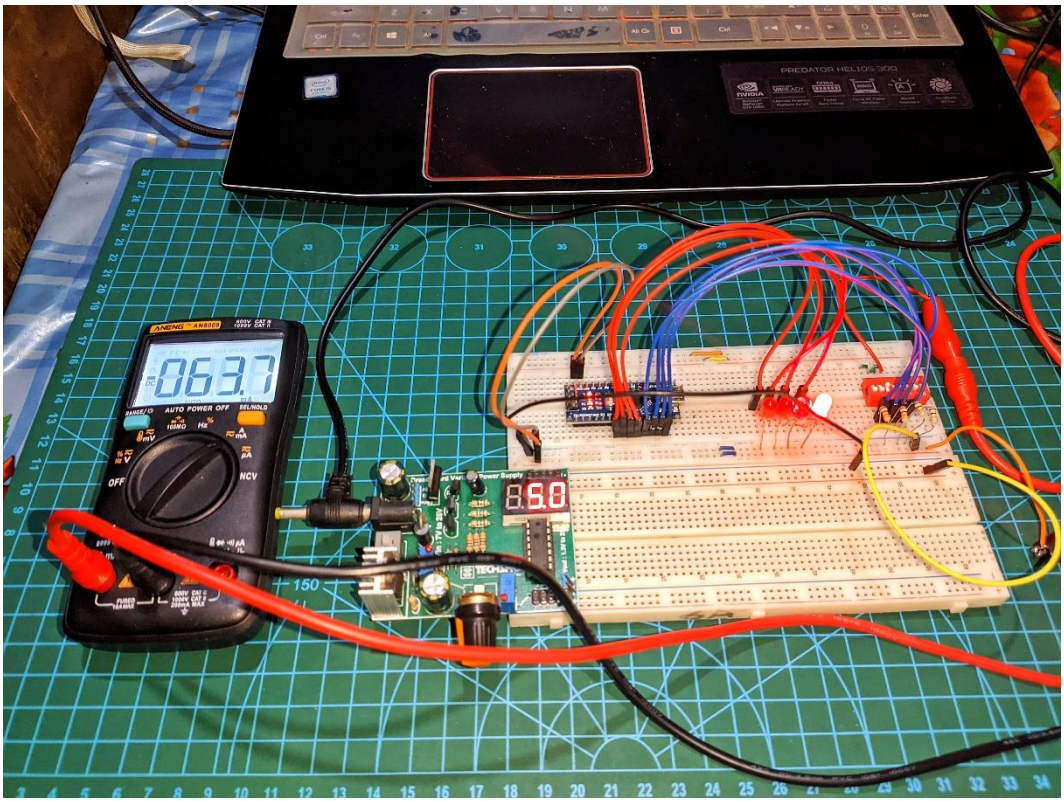
Table 5.1.1: Equipment & Components used.

5.2 Arduino Uno R3 Implementation



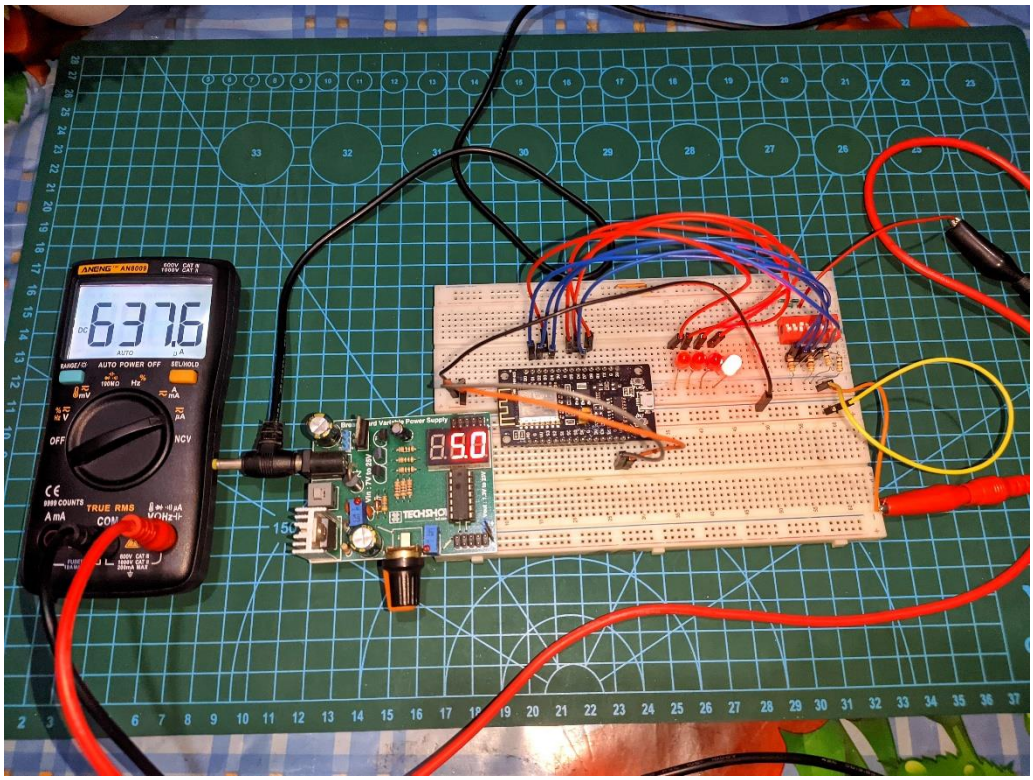
*Circuit 5.2.1: Implementation of Arduino Uno and Current for **0010** as input*

5.3 Arduino Nano Implementation



*Circuit 5.3.1: Implementation of Arduino Nano and Current for **0010** as input.*

5.4 NodeMCU V3 Implementation



*Circuit 5.4.1: Implementation of NodeMCU V3 and Current for **0010** as input.*

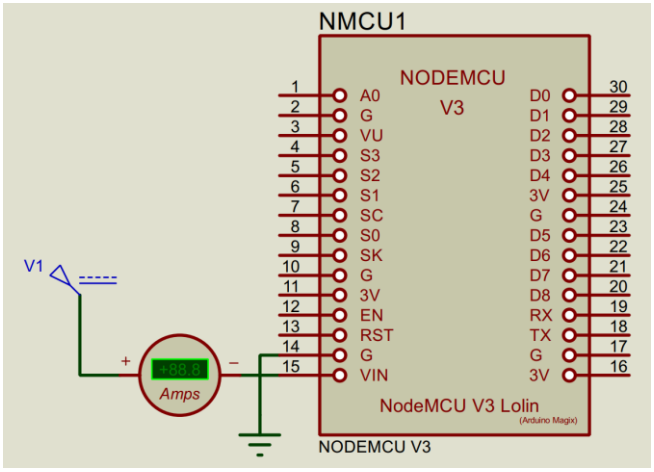
6. Data Collection & Graphical Analysis

6.1 Data Collection Procedure*

1. Since we are powering with a fixed 5V constant power supply, the power can be found by only measure 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.

IDLE State Current:

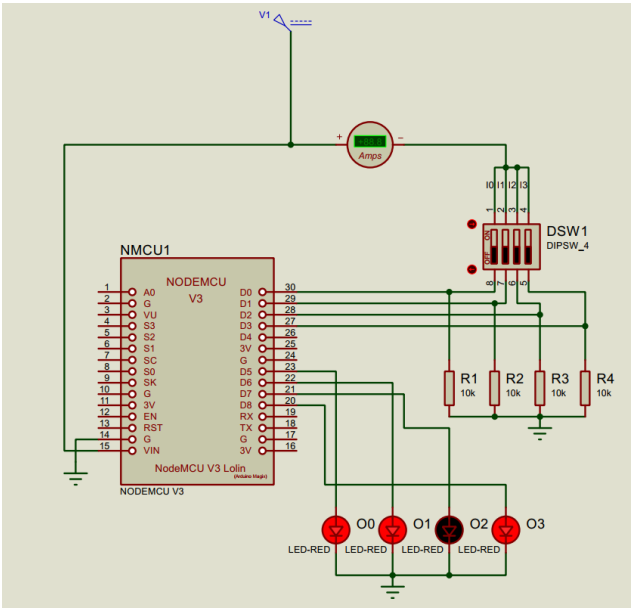
Follow the ammeter in the below circuit diagram.



Circuit 6.1.1: IDLE State Current Measurement

Input Side Current:

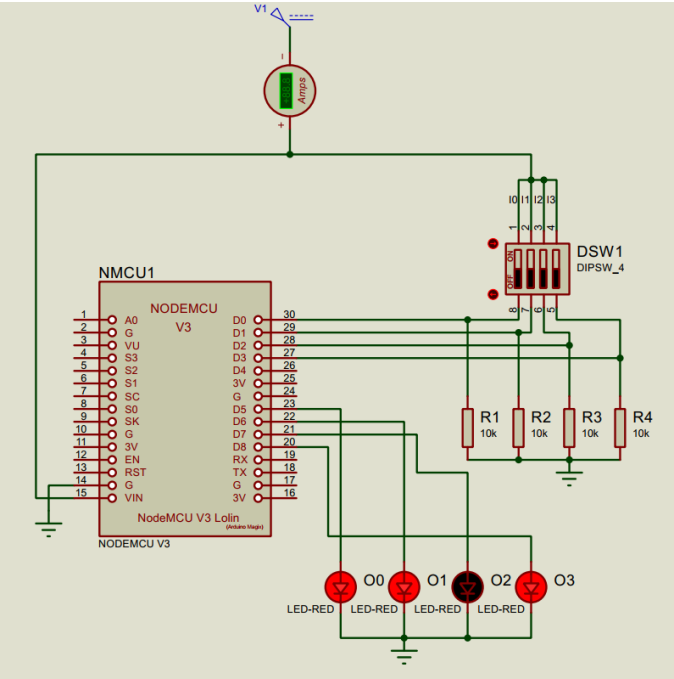
Follow the ammeter in circuit diagram below.



Circuit 6.1.2: Input Side Current Measurement

Total System Current:

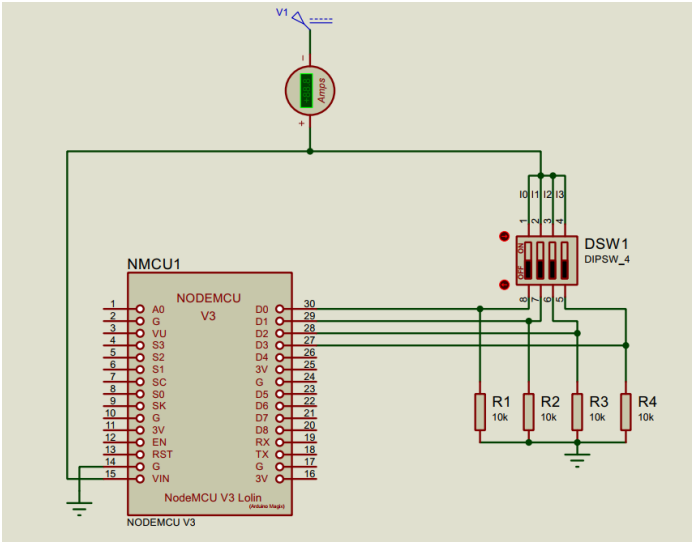
Follow the ammeter in circuit diagram below.



Circuit 6.1.3: Total System Current Measurement

Total System Power without LEDs:

Follow the ammeter in circuit diagram below.



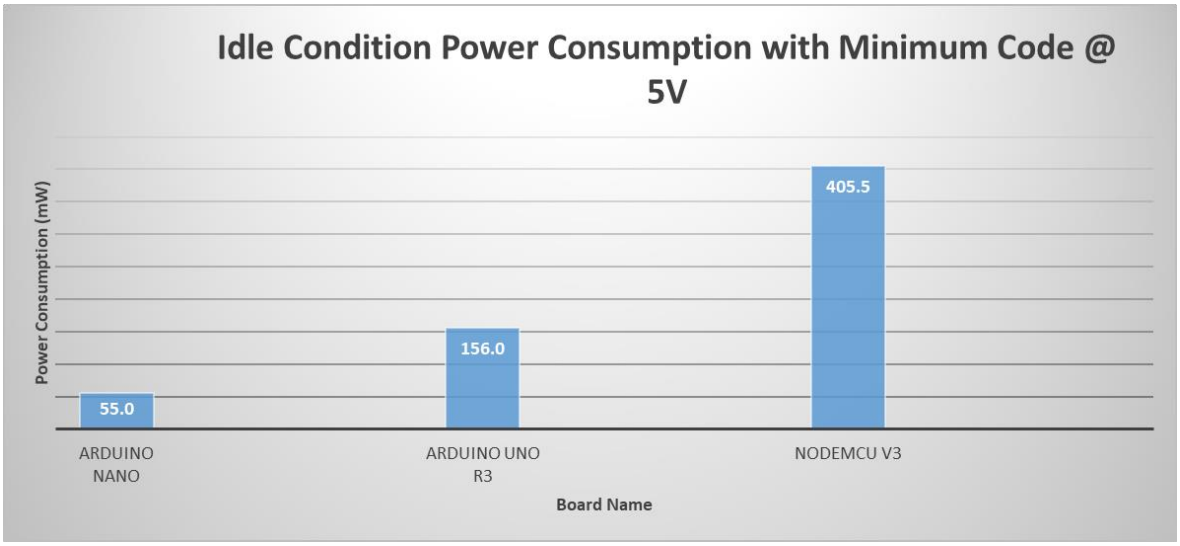
Circuit 6.1.4: Total System without LEDs Current Measurement

6.2 IDLE State Power Consumption

CSE331.4 Group-2 (Spring 2022)						
Idle Condition Power Consumption with Minimum Code Running						
Board Name	**MCU**	**Frequency**	**Current (mA)**	**Avg Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
Arduino Nano	ATmega328	16 MHz	10.9	11.0	5.0	55.0
			11.1			
			11.1			
Arduino Uno R3	ATmega328P	16 MHz	30.3	31.2	5.0	156.0
			31.7			
			31.5			
NodeMCU V3	ESP8266	80 MHz	81.2	81.1	5.0	405.5
			81.1			
			81.1			

Table 6.2.1: IDLE State Power Consumption.

Graphical Analysis



Graph 6.2.2: Comparison between three boards IDLE state power consumption.

- Here we can see that the power consumption in **idle state** is:

$$\boldsymbol{NodeMCU > Uno > Nano}$$

- For this case **NodeMCU** has the **highest power consumption in idle state** because its clock frequency is higher (**80 MHz**, **Appendix B**) than Arduino Uno and Nano (both **16 MHz**).
- Another reason also contributes to this higher power consumption is the fact that NodeMCU has **WiFi** and Arduino Uno and Nano doesn't.
- Although Uno and Nano both have same clock frequency (**16 MHz**), there are more components present in Arduino Uno than Nano. As a result in idle state Uno has to power greater number of components than Nano so power consumption of Uno is higher.

6.3 Input Side Power Consumption

CSE331.4 Group-2 (Spring 2022)					
Arduino Uno R3 Input Side Power Consumption					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Voltage (V)**	**Power (mW)**
0000	1101	0.04 0.05 0.04	0.04	5.0	0.20
0001	1101	51.7 51.9 51.3	51.6	5.0	258.0
0010	0001	26.0 25.9 26.0	26.0	5.0	130.0
0011	0000	22.4 22.3 22.4	22.4	5.0	112.0
0100	0011	29.3 29.4 29.4	29.4	5.0	147.0
0101	0010	36.8 36.7 36.8	36.8	5.0	184.0
0110	1111	68.4 68.5 68.7	68.5	5.0	342.5
0111	0010	39.6 39.5 39.5	39.5	5.0	197.5
1000	1011	31.9 31.5 31.7	31.7	5.0	158.5
1001	1000	39.6 39.9 39.8	39.8	5.0	199.0
1010	0010	63 63.5 63.7	63.4	5.0	317.0
1011	0011	39.4 39.5 39.7	39.5	5.0	197.5
1100	0001	47.5 47.5 47.6	47.5	5.0	237.5
1101	0110	72.8 73.4 73.6	73.3	5.0	366.5
1110	0011	61.8 62 62.2	62.0	5.0	310.0
1111	1011	85.3 85.9 86	85.7	5.0	428.5

Table 6.3.1: Arduino Uno R3 Input Side Power

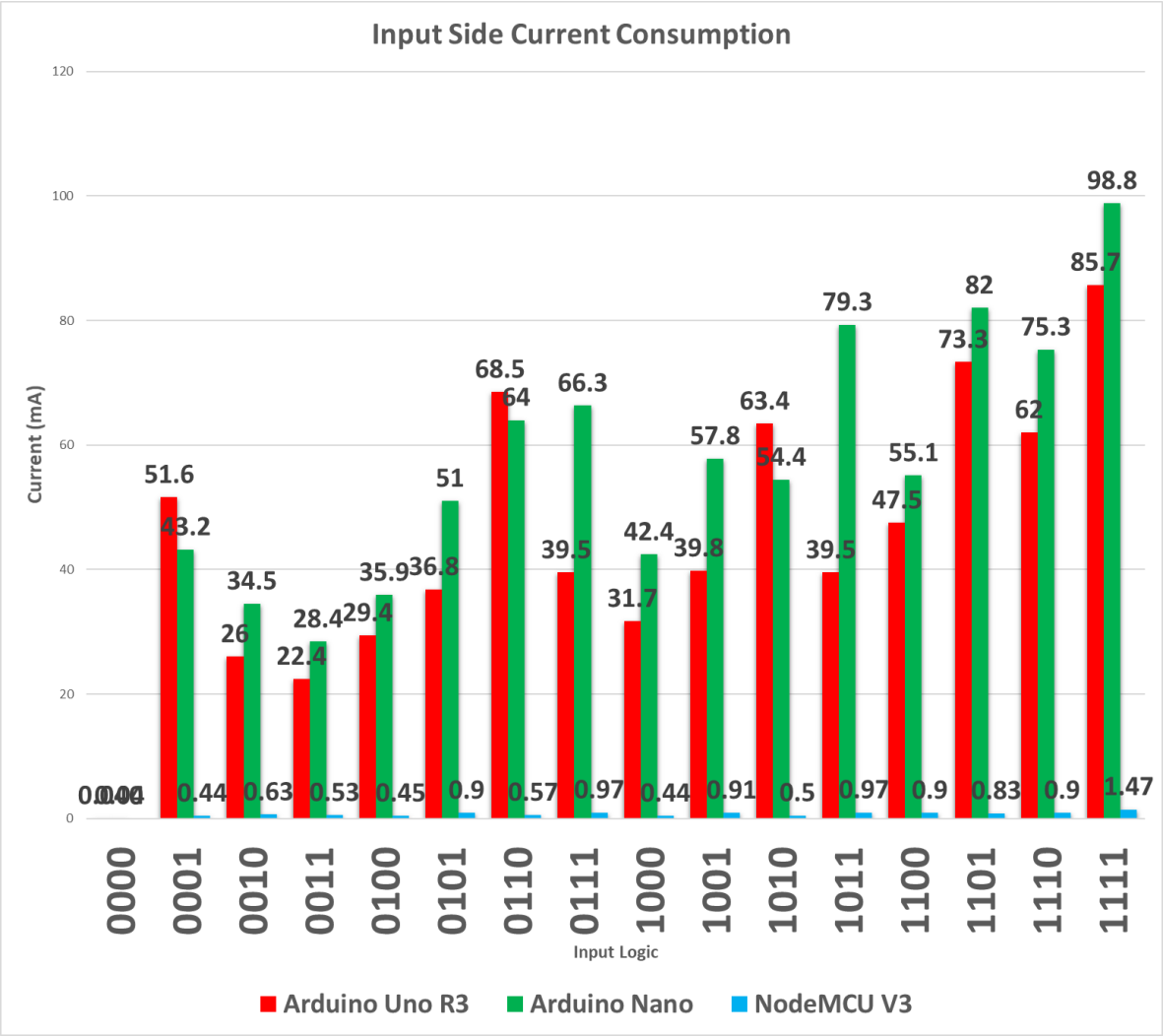
CSE331.4 Group-2 (Spring 2022)					
Arduino Nano Input Side Power Consumption					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Voltage (V)**	**Power (mW)**
0000	1101	0.40	0.43	5.0	2.15
		0.40			
		0.50			
0001	1101	43.3	43.2	5.0	216.0
		43.1			
		43.1			
0010	0001	34.4	34.5	5.0	172.5
		34.6			
		34.4			
0011	0000	28.9	28.4	5.0	142.0
		28.5			
		27.9			
0100	0011	35.8	35.9	5.0	179.5
		35.9			
		36.0			
0101	0010	50.6	51.0	5.0	255.0
		51.2			
		51.2			
0110	1111	64.0	64.0	5.0	320
		63.8			
		64.2			
0111	0010	65.9	66.3	5.0	331.5
		66.4			
		66.5			
1000	1011	42.3	42.4	5.0	212
		42.5			
		42.4			
1001	1000	57.6	57.8	5.0	289.0
		57.8			
		58.1			
1010	0010	54.2	54.4	5.0	272.0
		54.5			
		54.5			
1011	0011	79.2	79.3	5.0	396.5
		79.4			
		79.4			
1100	0001	55.0	55.1	5.0	275.5
		55.2			
		55.2			
1101	0110	81.8	82.0	5.0	410
		82.1			
		82.0			
1110	0011	75.3	75.3	5.0	376.5
		75.4			
		75.3			
1111	1011	98.6	98.8	5.0	494
		98.9			
		99.0			

Table 6.3.2: Arduino Nano Input Side Power Consumption

CSE331.4 Group-2 (Spring 2022)					
NodeMCU V3 Input Side Power Consumption					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	0	0	5.0	0.00
		0			
		0			
0001	1101	0.44	0.44	5.0	2.20
		0.44			
		0.44			
0010	0001	0.63	0.63	5.0	3.15
		0.63			
		0.63			
0011	0000	0.5	0.53	5.0	2.65
		0.6			
		0.5			
0100	0011	0.45	0.45	5.0	2.25
		0.45			
		0.45			
0101	0010	0.9	0.9	5.0	4.50
		0.9			
		0.9			
0110	1111	0.6	0.57	5.0	2.85
		0.5			
		0.6			
0111	0010	1	0.97	5.0	4.85
		0.9			
		1			
1000	1011	0.44	0.44	5.0	2.20
		0.44			
		0.44			
1001	1000	0.91	0.91	5.0	4.55
		0.91			
		0.91			
1010	0010	0.4	0.5	5.0	2.50
		0.5			
		0.6			
1011	0011	1	0.97	5.0	4.85
		0.9			
		1			
1100	0001	0.9	0.9	5.0	4.50
		0.9			
		0.91			
1101	0110	0.8	0.83	5.0	4.15
		0.8			
		0.9			
1110	0011	1	0.9	5.0	4.50
		0.9			
		0.8			
1111	1011	1.5	1.47	5.0	7.35
		1.4			
		1.5			

Table 6.3.3: NodeMCU V3 Input Side Power Consumption

Graphical Analysis:



Graph 6.3.4: Input Side Current Consumption Comparison between the three boards.

- Here we can see that **Arduino Nano** has the **highest power consumption** in this case.
- We initially thought Nano will consume the lowest power however we can see NodeMCU V3 consume such low currents in this case.
- Although Arduino Uno and Nano have almost the exact same specifications, this difference in power might be due to the fact that Arduino Uno has a more power efficient chip which is **ATmega328P** which is a slight upgraded version of ATmega328 which is in the Nano.
- **ATmega328P consumes less power than ATmega328**.
- Notice the difference for when **one switch is ON**, for the logic: 0001, 0010 and 0100. There is a difference in current in the input branch! This difference may be due to the **output side** of the circuit and it is **affecting** the **input side**.

- Maximum consumption is for the logic 1111 and minimum for 0000 for all boards.

6.4 Total System Power Consumption

CSE331.4 Group-2 (Spring 2022)					
Arduino Uno R3 Total Power Consumption with LED					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	114.2 117.8 117.9	116.6	5.0	583.00
0001	1101	141.1 139,4 139.1	140.1	5.0	700.5
0010	0001	88.1 87.7 87.9	87.9	5.0	439.5
0011	0000	46.9 47 47	47	5.0	235.0
0100	0011	123.1 122.5 122.3	122.6	5.0	613.0
0101	0010	101 100.7 100.8	100.8	5.0	504.0
0110	1111	176.4 174.8 174.5	175.2	5.0	876
0111	0010	108.5 108.5 108.6	108.5	5.0	542.5
1000	1011	148.1 148 149.4	148.5	5.0	742.5
1001	1000	97.7 97.9 97.7	97.8	5.0	489.0
1010	0010	106 105.9 106	106	5.0	530.0
1011	0011	148.7 148.8 148.8	148.8	5.0	744
1100	0001	102.9 103.1 103.1	103.3	5.0	516.5
1101	0110	139.5 140.2 140.4	140	5.0	700
1110	0011	145.9 145.8 145.6	145.8	5.0	729.0
1111	1011	176 176 174.6	175.5	5.0	877.5

Table 6.4.1: Arduino Uno R3 Total Power Consumption with LED

CSE331.4 Group-2 (Spring 2022)					
Arduino Nano Total Power Consumption with LED					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	91	91	5.0	455.00
		91.1			
		90.9			
0001	1101	122.7	122.4	5.0	612.0
		122.3			
		122.2			
0010	0001	49.5	49.3	5.0	246.5
		49.1			
		49.2			
0011	0000	33.6	33.6	5.0	168.0
		33.7			
		33.6			
0100	0011	85.7	36.4	5.0	182.0
		86.6			
		86.8			
0101	0010	90.4	90.6	5.0	453.0
		90.7			
		90.6			
0110	1111	118.5	117.9	5.0	589.5
		117.5			
		117.6			
0111	0010	101.9	101.7	5.0	508.5
		101.6			
		101.7			
1000	1011	94.5	92.7	5.0	463.5
		92.3			
		91.4			
1001	1000	87.3	87.7	5.0	438.5
		87.9			
		88			
1010	0010	87.5	87.9	5.0	439.5
		88.1			
		88.1			
1011	0011	134.1	133.7	5.0	668.5
		133.6			
		133.5			
1100	0001	90	90.1	5.0	450.5
		90.3			
		90.1			
1101	0110	131.4	132.8	5.0	664
		133.5			
		133.4			
1110	0011	124.9	124.9	5.0	624.5
		125			
		124.8			
1111	1011	164	163.3	5.0	816.5
		163.3			
		162.6			

Table 6.4.2: Arduino Nano Total Power Consumption with LED

CSE331.4 Group-2 (Spring 2022)					
NodeMCU V3 Total Power Consumption with LED					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	138 137.7 137.9	137.9	5.0	689.50
0001	1101	121 124.3 124.4	123.2	5.0	616.0
0010	0001	98.9 99 98.9	98.9	5.0	494.5
0011	0000	81.3 81.6 82.1	89.6	5.0	448.0
0100	0011	113.5 113.8 114.1	113.8	5.0	569.0
0101	0010	101.2 100.2 100.4	100.6	5.0	503.0
0110	1111	127.6 125.1 125.5	126.1	5.0	630.5
0111	0010	103.6 103.7 103.5	103.6	5.0	518
1000	1011	119.6 118.9 119	119.2	5.0	596
1001	1000	100.5 100.7 100.8	100.7	5.0	503.5
1010	0010	105.1 105.1 105.2	105.1	5.0	525.5
1011	0011	114 113.9 114	114	5.0	570
1100	0001	99 96.6 97.7	97.8	5.0	489
1101	0110	105.2 105.3 105.4	105.3	5.0	526.5
1110	0011	112.7 112.6 112.5	112.6	5.0	563.0
1111	1011	116.3 116.4 116.8	116.5	5.0	582.5

Table 6.4.3: NodeMCU V3 Total Power Consumption with LED

Graphical Analysis

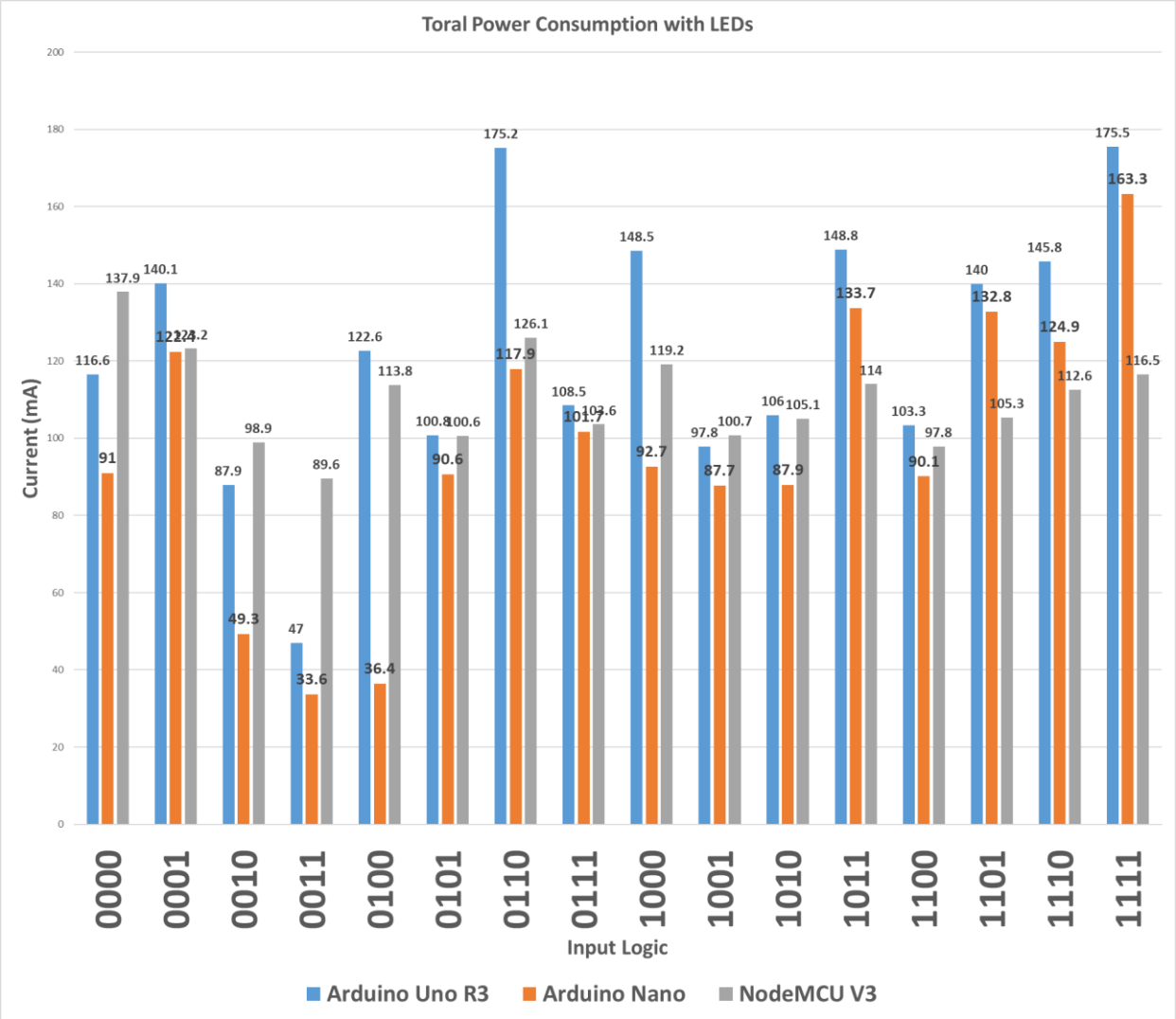


Table 6.4.4: Total System Current with LEDs Consumption Comparison between the three boards.

- Here we can see that **Arduino Uno** has the **highest total power consumption** compared to the others in this case.
- And the Arduino Nano is consuming the lowest total power in most of the cases except few.
- We also the similar trend as before for the 1 switch on situation.
- **Maximum power** is for input logic **1111** for **Arduino Uno** and **Nano**. For **NodeMCU** it is for the logic **0000**.
- **Minimum** is at **0011** for all the boards and this is logical because at this level all the LEDs are off.

6.5 Total System Power Consumption without LEDs

CSE331.4 Group-2 (Spring 2022)					
Arduino Uno R3 Total Power Consumption without LED					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	28.9 29 29.1	29	5.0	145.00
0001	1101	41.7 42 41.8	41.8	5.0	209.0
0010	0001	40.9 41 40.9	40.9	5.0	204.5
0011	0000	43.2 43.1 43.2	43.2	5.0	216.0
0100	0011	41.6 41.5 41.8	41.6	5.0	208.0
0101	0010	44.3 44.2 44.3	44.3	5.0	221.5
0110	1111	48.4 48.5 48.5	48.5	5.0	242.5
0111	0010	51.9 52 52	52	5.0	260
1000	1011	42.7 42.6 42.8	42.7	5.0	213.5
1001	1000	49.7 49.8 49.7	49.7	5.0	248.5
1010	0010	50.3 50.2 49.9	50.1	5.0	250.5
1011	0011	53.9 54.1 54.1	54	5.0	270
1100	0001	49.4 49.6 49.7	49.6	5.0	248
1101	0110	53.4 53.7 53.9	53.7	5.0	268.5
1110	0011	53.9 54.1 54.1	54	5.0	270.0
1111	1011	56.8 56.7 57	56.8	5.0	284

Table 6.5.1: Arduino Uno R3 Total Power Consumption without LEDs

CSE331.4 Group-2 (Spring 2022)					
Arduino Nano Total Power Consumption without LED					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	11.8	11.8	5.0	59.00
		11.7			
		11.8			
0001	1101	27.3	27.4	5.0	137.0
		27.5			
		27.5			
0010	0001	27.4	27.5	5.0	137.5
		27.6			
		27.6			
0011	0000	31.9	32	5.0	160.0
		32			
		32.2			
0100	0011	27.8	27.9	5.0	139.5
		27.9			
		28			
0101	0010	32.8	33.1	5.0	165.5
		33.2			
		33.3			
0110	1111	33.4	33.6	5.0	168
		33.7			
		33.7			
0111	0010	34.9	35.1	5.0	175.5
		35.2			
		35.1			
1000	1011	27.9	28	5.0	140
		28.1			
		28			
1001	1000	33.2	33.3	5.0	166.5
		33.4			
		33.4			
1010	0010	33.4	33.4	5.0	167.0
		33.3			
		33.6			
1011	0011	35	35.2	5.0	176
		35.2			
		35.4			
1100	0001	34.1	34.2	5.0	171
		34.3			
		34.2			
1101	0110	34.4	34.8	5.0	174
		34.8			
		35.1			
1110	0011	34.8	35.2	5.0	176.0
		35.1			
		35.7			
1111	1011	36.3	36.2	5.0	181
		36.1			
		36.3			

Table 6.5.2: Arduino Nano Total Power Consumption without LEDs

CSE331.4 Group-2 (Spring 2022)					
NodeMCU V3 Total Power Consumption without LED					
Input Logic	**Output Logic**	**Current (mA)**	**Avg. Current (mA)**	**Ref. Voltage (V)**	**Power (mW)**
0000	1101	79.7 79.6 79.5	79.6	5.0	398.00
0001	1101	79.9 80 80.1	80	5.0	400.0
0010	0001	80.7 80.8 80.5	80.7	5.0	403.5
0011	0000	82 82.8 81.8	82.2	5.0	411.0
0100	0011	80.8 80.7 81	80.8	5.0	404.0
0101	0010	81.1 80.8 81	81	5.0	405.0
0110	1111	80.7 80.9 80.9	80.8	5.0	404
0111	0010	81.1 81.2 81.3	81.2	5.0	406
1000	1011	79 79 79.3	79.1	5.0	395.5
1001	1000	79.5 80.6 80.4	80.2	5.0	401.0
1010	0010	81.2 81.3 81.1	81.2	5.0	406.0
1011	0011	82.2 82.1 81.9	82.1	5.0	410.5
1100	0001	81.2 81.4 81.7	81.4	5.0	407
1101	0110	82.1 82.1 82.2	82.1	5.0	410.5
1110	0011	82.1 82 81.9	82	5.0	410.0
1111	1011	82.3 82.4 82.3	82.3	5.0	411.5

Table 6.5.3: NodeMCU V3 Total Power Consumption without LEDs

Graphical Analysis

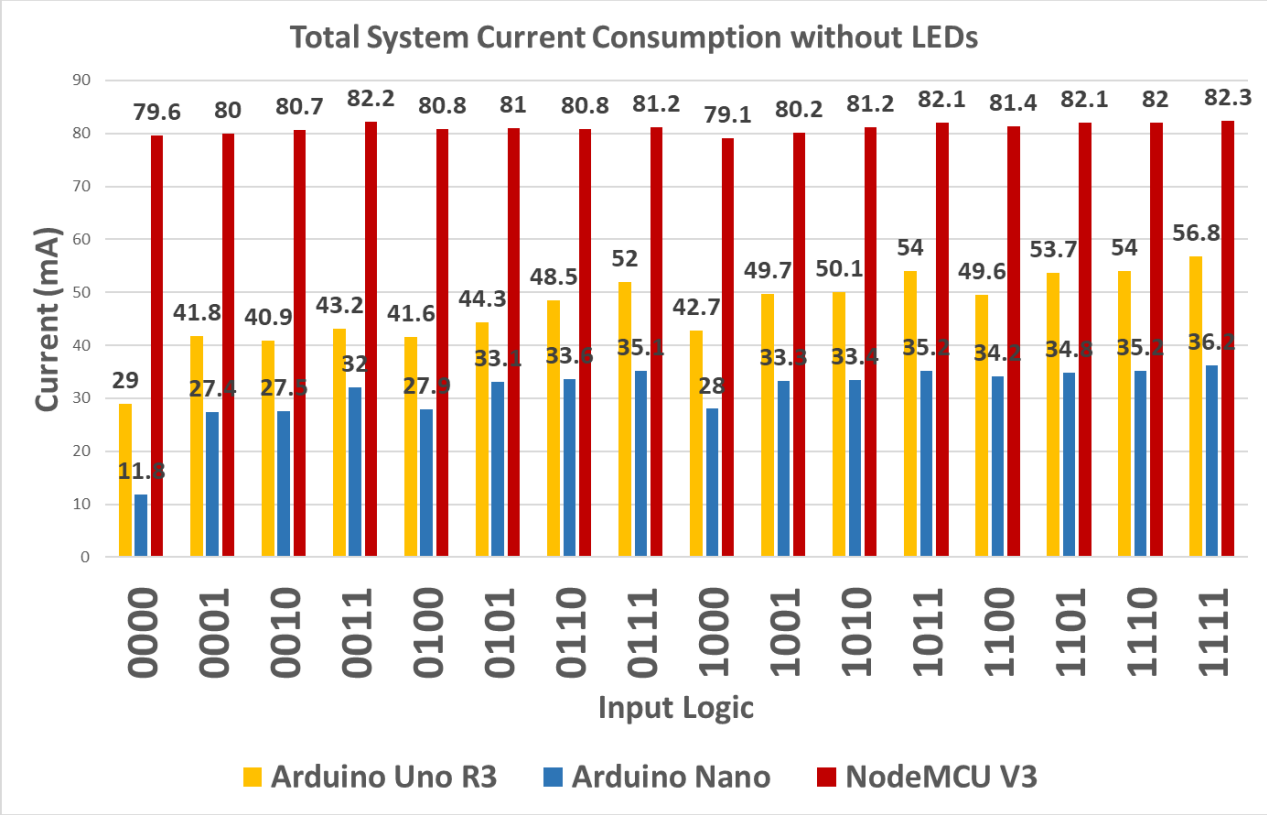


Table 6.5.4: Total System Current without LEDs Consumption Comparison between the three boards.

- For the **0000 input sequence**, we can see the **IDLE current** which is logic.
- As you can see the **trend** of the values are **fairly constant** and **horizontal**.
- This can be due to the effect of **frequency overpowering the logic implementation.**
- The slight difference is due to the difference in **computation complexity** for each logic.
- The maximum is for the input 1111 and this consumes the highest power. So complexity to calculate the output can be said to be the highest.
- Similarly the minimum is for 0000 so complexity to calculate the output can be said to be the lowest.

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: 3.38 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.

7.2 Arduino Nano

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: 3.38 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 Nano 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.

7.3 NodeMCU V3

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.

8. References

1. Project materials provided by Dr. Dihan Md. Nuruddin Hassan.

2. https://www.elprocus.com/what-is-arduino-uno-r3-pin-diagram-specification-and-applications/

3. https://www.elprocus.com/an-overview-of-arduino-nano-board/

4. https://www.elprocus.com/esp8266-wi-fi-module/

5. https://www.make-it.ca/nodemcu-details-specifications/

6.https://techshopbd.com/detail/2060/Breadboard_Variable_Power_Supply_techshop_bangladesh

7. https://www.techtarget.com/whatis/definition/routine

8. https://circuitdigest.com/microcontroller-projects/arduino-interrupt-tutorial-with-examples

9. https://www.electronicwings.com/nodemcu/nodemcu-gpio-interrupts-with-arduino-ide#:~:text=NodeMCU%20based%20ESP8266%20has%20an,except%20the%20D0%2FGPIO16%20pin.

10. https://comparecamp.com/arduino-ide-review-pricing-pros-cons-features/

11. https://en.wikipedia.org/wiki/Proteus_Design_Suite

12. https://study.com/academy/lesson/introduction-to-logisim-setup-overview.html

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 Arduino-based 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 real-world, 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

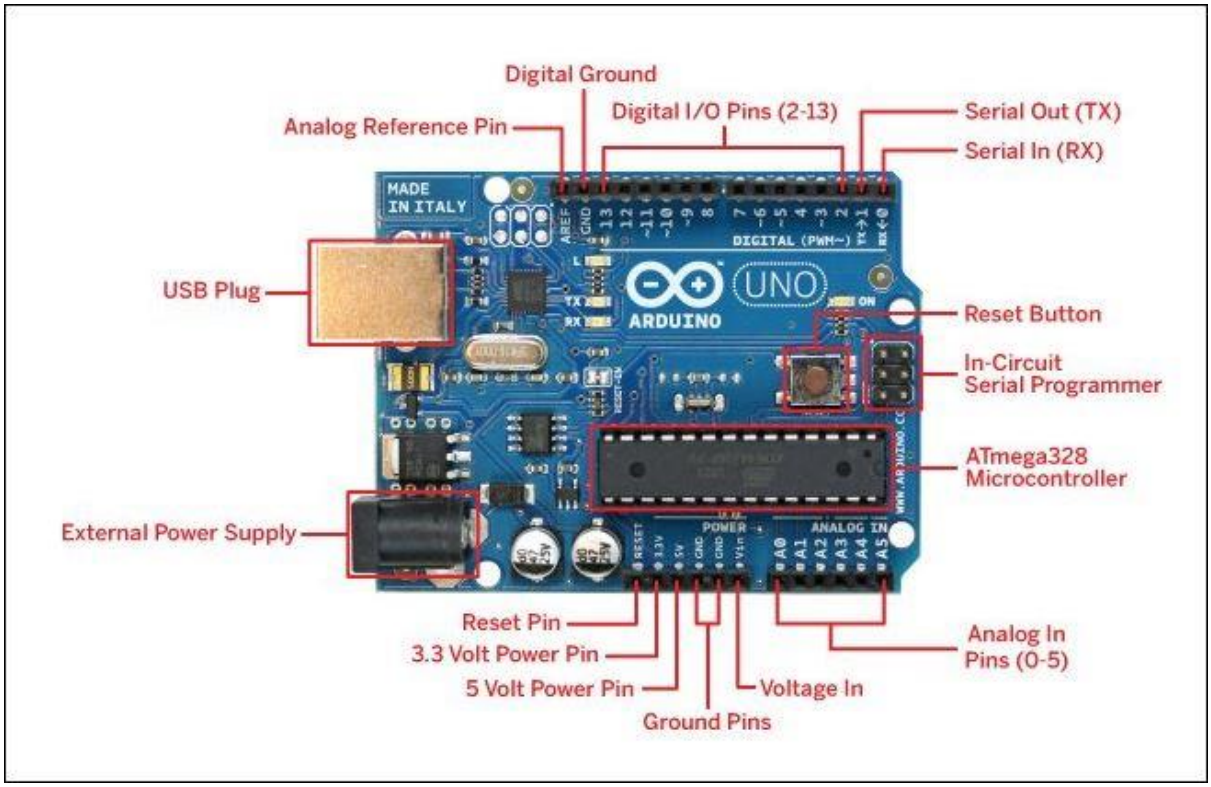
The Arduino UNO R3 is frequently used microcontroller board in the family of an Arduino. This is the latest third version of an Arduino board and released in the year 2011. The main advantage of this board is if we make a mistake we can change the microcontroller on the board. The main features of this 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:


Digital Ground
Analog Reference Pin
Digital I/O Pins (2-13)
Serial Out (TX)
Serial In (RX)
USB Plug
Reset Button
In-Circuit
Serial Programmer
ATmega328
Microcontroller
External Power Supply
Reset Pin
3.3 Volt Power Pin
5 Volt Power Pin
Ground Pins
Voltage In
Analog In
Pins (0-5)
MADE
IN ITALY
UNO
ARDUINO
DIGITAL (PWM~)
POWER
ANALOG IN

Arduino Nano

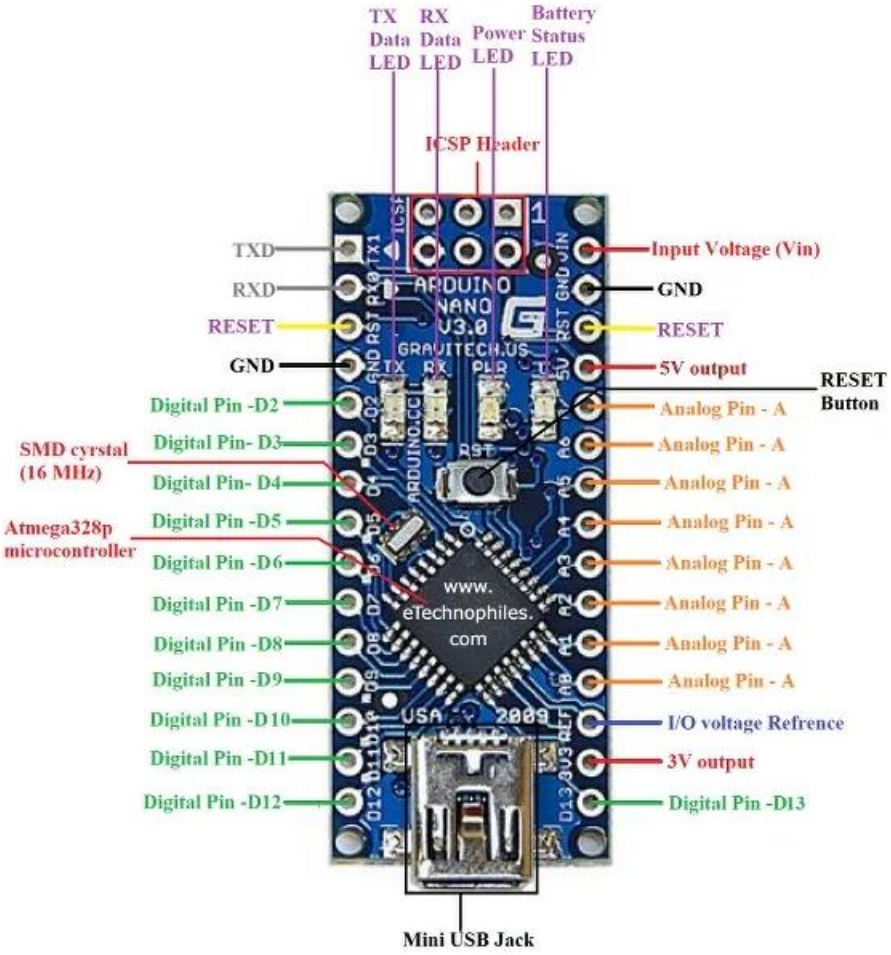
Arduino Nano is one type of microcontroller board, and it is designed by Arduino.cc. It can be built with a microcontroller like Atmega328. This microcontroller is also used in Arduino UNO. It is a small size board and also flexible with a wide variety of applications. Other Arduino boards mainly include Arduino Mega, Arduino Pro Mini, Arduino UNO, Arduino YUN, Arduino Lilypad, Arduino Leonardo, and Arduino Due. And other development boards are AVR Development Board, PIC Development Board, Raspberry Pi, Intel Edison, MSP430 Launchpad, and ESP32 board.

This board has many functions and features like an Arduino Duemilanove board. However, this Nano board is different in packaging. It doesn't have any DC jack so that the power supply can be given using a small USB port otherwise straightly connected to the pins like VCC & GND. This board can be supplied with 6 to 20volts using a mini USB port on the board.

Arduino Nano Specifications

- ATmega328P Microcontroller is from 8-bit AVR family
- Operating voltage is 5V
- Input voltage (Vin) is 7V to 12V
- Input/Output Pins are 22
- Analog i/p pins are 6 from A0 to A5
- Digital pins are 14
- Power consumption is 19 mA
- I/O pins DC Current is 40 mA
- Flash memory is 32 KB
- SRAM is 2 KB
- EEPROM is 1 KB
- CLK speed is 16 MHz
- Weight-7g
- Size of the printed circuit board is 18 X 45mm
- Supports three communications like SPI, IIC, & USART

Pin Diagram:



Pinout of Arduino Nano

NodeMCU V3

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)

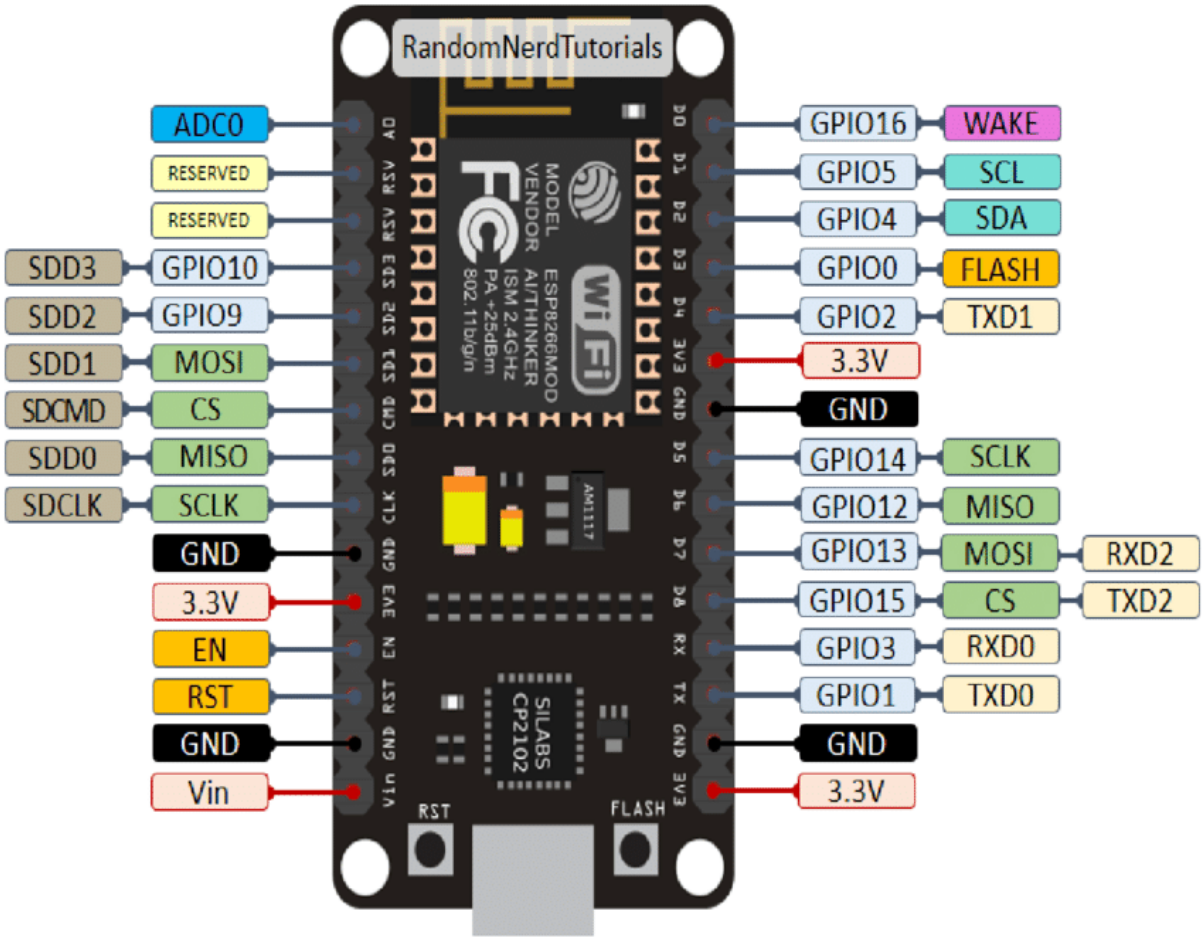
The ESP8266 Wi-Fi module comes with a boot ROM of 64 KB, user data RAM of 80 KB, and instruction RAM of 32 KB. It can support 802.11 b/g/n Wi-Fi network at 2.4 GHz along with the features of I2C, SPI, I2C interfacing with DMA, and 10-bit ADC. Interfacing this module with the microcontroller can be done 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 V3 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.
- It is an IEEE 802.11 b/g/n Wi-Fi module with LNA, power amplifier, balun, integrated TR switch, and matching networks.
- 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:


RandomNerdTutorials
ADC0
RESERVED
RESERVED
SDD3
GPIO10
SDD2
GPIO9
SDD1
MOSI
SDCMD
CS
SDD0
MISO
SDCLK
SCLK
GND
3.3V
EN
RST
GND
Vin
GPIO16
WAKE
GPIO5
SCL
GPIO4
SDA
GPIO0
FLASH
GPIO2
TXD1
3.3V
GND
GPIO14
SCLK
GPIO12
MISO
GPIO13
MOSI
RXD2
GPIO15
CS
TXD2
GPIO3
RXD0
GPIO1
TXD0
GND
3.3V
RST
FLASH

ANENG AN8009 Multimeter

Generic GB Digital Multimeter AN8009 LCD Display 9999 Counts AC/DC Tester-Black.

Features:

1. Operating Temperature: 0 - 40 鈩?br>Operating Humidity: 鈥?5%RH
2. Storage Condition: -20~60 鈩?br>Storage Humidity: 鈥?0%RH
3. DIY Supplies: Electrical
4. Operating Mode: Auto/Manual Ranging
5. Measuring Inductance Range: no
6. Measuring Temperature Range: -20~1000 C/-4F-1832F

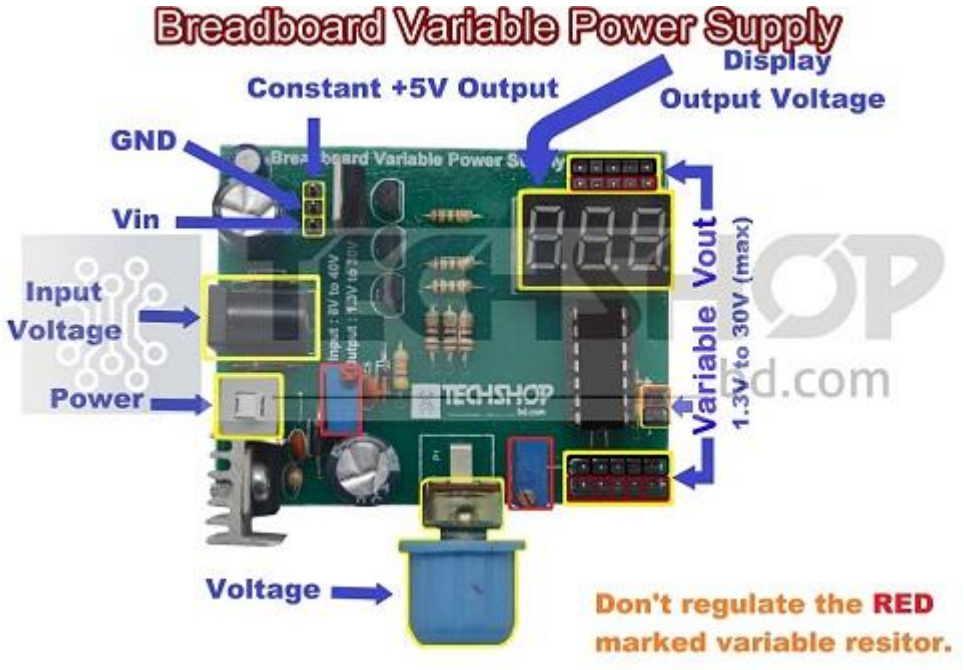
Descriptions:

1. Measuring Capacitance Range: 9.99nF/99.99nF/999.9nF/9.99uF/99.99uF/999.9uF/9.999MF
2. Measuring Voltage Range: 999.9mV/9.999V/99.99V/999.9V
3. Measuring Current Range:60mA/600mA/10A
4. Measuring Resistance Range: 99.99/999.9/9.999k/99.99k/999.9K/999.9M
5. Frequency: 99.99Hz/999.9Hz/9.999KHz/99.99KHz/999.9KHz/9.999MHz
6. Square Wave Output: 50Hz/100Hz/200Hz/300Hz/400Hz/500Hz/600Hz/700Hz/800Hz/ 900Hz/1000Hz/200
7. DC/AC Voltage(mV): 9.999mV/99.99mV
8. DC/AC Voltage(V): 999.9mV/9.999V/99.99V/750ACV(999.9DCV)
9. DC/AC Current (mA&A): 999.9mA/9.999A
10. DC/AC Current (uA): 99.99 渭 A/999.9 渭 A
11. Display Type: Digital Display
12. Diode Test: Yes
13. Continuity: Yes
14. Duty Cycle: 1%-99%
15. Sample Rate: 3 times per second
16. Power: 2 * 1.5V AAA batteries.

Variable Breadboard Power Supply

- Input Voltage:7V to 25V
- Variable Output Voltage: 1.3V to 23V
- Variable Output Current: 500mA (+Vout Pin)
- Constant Output Current: 800mA (+5V Pin)
- Output Current: 2A (Vin Pin)

Pin Diagram:



Appendix C: Codes

Arduino Uno R3

```
// CSE331.4 (Spring 2022)
// Group-2
// Code for Arduino Uno R3:

// Declaring pins of I/O:
//Input Pins:      Output Pins:
  int pin_I0 = 11; int pin_O0 = 4;
  int pin_I1 = 10; int pin_O1 = 5;
  int pin_I2 =  9; int pin_O2 = 6;
  int pin_I3 =  8; int pin_O3 = 7;

// Logical Operators of input:
  int I0 = 0;
  int I1 = 0;
  int I2 = 0;
  int I3 = 0;

void setup() {
  // Pin Mode for Input:   Pin Mode for Outut:
   pinMode(pin_I0, INPUT); pinMode(pin_O0, OUTPUT);
   pinMode(pin_I1, INPUT); pinMode(pin_O1, OUTPUT);
   pinMode(pin_I2, INPUT); pinMode(pin_O2, OUTPUT);
   pinMode(pin_I3, INPUT); pinMode(pin_O3, OUTPUT);
}

void loop() {
  // Reading the input pins:
  I0 = digitalRead(pin_I0);
  I1 = digitalRead(pin_I1);
  I2 = digitalRead(pin_I2);
  I3 = digitalRead(pin_I3);

  // Applying the logical expressions for output:
  // For O0 = (I1+I2')(I1'+I2)(I0+I2'+I3')(I0'+I2'+I3)
  if ( (I1||!I2)&&(!I1||I2)&&(I0||!I2||!I3)&&(!I0||!I2||I3) ){
```

```
    digitalWrite(pin_O0, HIGH);
  }
  else{
    digitalWrite(pin_O0, LOW);
  }
  // For O1 = (I0'.I1'.I2')+(I0'.I1.I2.I3')+(I0.I1.I2'.I3)
  if ( (!I0&&!I1&&!I2)||(!I0&&I1&&I2&&!I3)||(I0&&I1&&!I2&&I3) ){
    digitalWrite(pin_O1, HIGH);
  }
  else{
    digitalWrite(pin_O1, LOW);
  }
  // For O2 = (I0'.I1)+(I1.I3)+(I0.I2)+(I0.I1'.I3')
  if ( (!I0&&I1)||(I1&&I3)||(I0&&I2)||(I0&&!I1&&!I3) ){
    digitalWrite(pin_O2, HIGH);
  }
  else{
    digitalWrite(pin_O2, LOW);
  }
  // For O3 = (I0'.I3')+(I2'.I3')+(I1.I3')+(I0'.I1'.I2')+(I0.I2.I3)
  if ( (!I0&&!I3)||(!I2&&!I3)||(I1&&!I3)||(!I0&&!I1&&!I2)||(I0&&I2&&I3)
){
    digitalWrite(pin_O3, HIGH);
  }
  else{
    digitalWrite(pin_O3, LOW);
  }
}
```

Arduino Nano

```
// CSE331.4 (Spring 2022)
// Group-2
// Code for Arduino Nano:

// Declaring pins of I/O:
//Input Pins:      Output Pins:
  int pin_I0 = 11; int pin_O0 = 4;
  int pin_I1 = 10; int pin_O1 = 5;
  int pin_I2 =  9; int pin_O2 = 6;
  int pin_I3 =  8; int pin_O3 = 7;

// Logical Operators of input:
  int I0 = 0;
  int I1 = 0;
  int I2 = 0;
  int I3 = 0;

void setup() {
  // Pin Mode for Input:   Pin Mode for Outut:
   pinMode(pin_I0, INPUT); pinMode(pin_O0, OUTPUT);
   pinMode(pin_I1, INPUT); pinMode(pin_O1, OUTPUT);
   pinMode(pin_I2, INPUT); pinMode(pin_O2, OUTPUT);
   pinMode(pin_I3, INPUT); pinMode(pin_O3, OUTPUT);
}

void loop() {
  // Reading the input pins:
  I0 = digitalRead(pin_I0);
  I1 = digitalRead(pin_I1);
  I2 = digitalRead(pin_I2);
  I3 = digitalRead(pin_I3);

  // Applying the logical expressions for output:
  // For O0 = (I1+I2')(I1'+I2)(I0+I2'+I3')(I0'+I2'+I3)
  if ( (I1||!I2)&&(!I1||I2)&&(I0||!I2||!I3)&&(!I0||!I2||I3) ){
    digitalWrite(pin_O0, HIGH);
  }
```

```
  else{
    digitalWrite(pin_O0, LOW);
  }
  // For O1 = (I0'.I1'.I2')+(I0'.I1.I2.I3')+(I0.I1.I2'.I3)
  if ( (!I0&&!I1&&!I2)||(!I0&&I1&&I2&&!I3)||(I0&&I1&&!I2&&I3) ){
    digitalWrite(pin_O1, HIGH);
  }
  else{
    digitalWrite(pin_O1, LOW);
  }
  // For O2 = (I0'.I1)+(I1.I3)+(I0.I2)+(I0.I1'.I3')
  if ( (!I0&&I1)||(I1&&I3)||(I0&&I2)||(I0&&!I1&&!I3) ){
    digitalWrite(pin_O2, HIGH);
  }
  else{
    digitalWrite(pin_O2, LOW);
  }
  // For O3 = (I0'.I3')+(I2'.I3')+(I1.I3')+(I0'.I1'.I2')+(I0.I2.I3)
  if ( (!I0&&!I3)||(!I2&&!I3)||(I1&&!I3)||(!I0&&!I1&&!I2)||(I0&&I2&&I3)
){
    digitalWrite(pin_O3, HIGH);
  }
  else{
    digitalWrite(pin_O3, LOW);
  }
}
```

NodeMCU V3

```
// CSE331.4 (Spring 2022)
// Group-2
// Code for NodeMCU V3:

// Declaring pins of I/O:
//Input Pins:
#define pin_I0 D1
#define pin_I1 D2
#define pin_I2 D4
#define pin_I3 D7
//Output Pins:
#define pin_O0 D3
#define pin_O1 D5
#define pin_O2 D6
#define pin_O3 D8

// Logical Operators of input:
  int I0;
  int I1;
  int I2;
  int I3;

void setup() {
  // Pin Mode for Input:
   pinMode(pin_I0, INPUT);
   pinMode(pin_I1, INPUT);
   pinMode(pin_I2, INPUT);
   pinMode(pin_I3, INPUT);
 // Pin Mode for Output:
   pinMode(pin_O0, OUTPUT);
   pinMode(pin_O1, OUTPUT);
   pinMode(pin_O2, OUTPUT);
   pinMode(pin_O3, OUTPUT);
}

void loop() {
  // Reading the input pins:
```

```
  I0 = digitalRead(pin_I0);
  I1 = digitalRead(pin_I1);
  I2 = digitalRead(pin_I2);
  I3 = digitalRead(pin_I3);

  // Applying the logical expressions for output:
  // For O0 = (I1+I2')(I1'+I2)(I0+I2'+I3')(I0'+I2'+I3)
  if ( (I1||!I2)&&(!I1||I2)&&(I0||!I2||!I3)&&(!I0||!I2||I3) ){
    digitalWrite(pin_O0, HIGH);
  }
  else{
    digitalWrite(pin_O0, LOW);
  }
  // For O1 = (I0'.I1'.I2')+(I0'.I1.I2.I3')+(I0.I1.I2'.I3)
  if ( (!I0&&!I1&&!I2)||(!I0&&I1&&I2&&!I3)||(I0&&I1&&!I2&&I3) ){
    digitalWrite(pin_O1, HIGH);
  }
  else{
    digitalWrite(pin_O1, LOW);
  }
  // For O2 = (I0'.I1)+(I1.I3)+(I0.I2)+(I0.I1'.I3')
  if ( (!I0&&I1)||(I1&&I3)||(I0&&I2)||(I0&&!I1&&!I3) ){
    digitalWrite(pin_O2, HIGH);
  }
  else{
    digitalWrite(pin_O2, LOW);
  }
  // For O3 = (I0'.I3')+(I2'.I3')+(I1.I3')+(I0'.I1'.I2')+(I0.I2.I3)
  if ( (!I0&&!I3)||(!I2&&!I3)||(I1&&!I3)||(!I0&&!I1&&!I2)||(I0&&I2&&I3)
){
    digitalWrite(pin_O3, HIGH);
  }
  else{
    digitalWrite(pin_O3, LOW);
  }
}
```

Appendix D: Resources

LogiSim File

Proteus Files