ELECTRONICS VIRTUAL LAB USING

PYTHON GUI

Final Year Project Report

Submitted by: -

- Akash Dutta (123180702003)
- Goutam Sutradhar (123180702026)
- Dibyendu Paul (123180702024)
- Debabrata Debnath (123180702023)
- Paramita Chandra (123180702041)

Done under the supervision of:-

Prof. Arindam Banerjee

JIS COLLEGE OF ENGINEERING

Kalyani, Nadia

Maulana Abul Kalam Azad University of Technology

May 2022

Department of

Electronics and Communication Engineering

DECLARATION

We hereby declare that the project entitled **"***ELECTRONICS VIRTUAL LAB USING PYTHON GUI***"** submitted for the B. Tech. (ECE) degree is our original work and the project has not formed the basis for the award of any other degree, diploma, fellowship, or similar titles.

Akash Dutta
ECE, 4th Year
123180702003

Goutam Sutradhar
ECE, 4th Year
123180702026

Dibyendu Paul
ECE, 4th Year
123180702024

Debabrata Debnath
ECE, 4th Year
123180702023

Paramita Chandra
ECE, 4th Year
123180702041

Date:- _____, May 2022

Place: JIS College of Engineering
Kalyani, Nadia

JIS College of Engineering
Block 'A', Phase-III, Kalyani, Nadia, 741235
Phone: +91 33 2582 2137, Telefax: +91 33 2582 2138
Website: www.jiscollege.ac.in, Email: info@jiscollege.ac.in

CERTIFICATE

This is to certify that the students mentioned below have completed their project entitled "ELECTRONICS VIRTUAL LAB USING PYTHON GUI" under the guidance of Mr. Arindam Banerjee in partial fulfilment of the requirements for the award of Bachelor of Technology in Electronics and Communication from JIS College of Engineering (An Autonomous Institute) is an authentic record of their work carried out during the academic year 2021-2022 and to the best of our knowledge, this work has not been submitted elsewhere as a part of the process of obtaining a degree, diploma, fellowship, or any other similar title.

Students Concerned:

Akash Dutta	(123180702003)
Goutam Sutradhar	(123180702026)
Dibyendu Paul	(123180702024)
Debabrata Debnath	(123180702023)
Paramita Chandra	(123180702041)

Signature of External ***ExpertSignature of HOD*** ***Signature of the Supervisor***

-: ACKNOWLEDGEMENT:-

We wish to express our profound and sincere gratitude to Mr. Arindam Banerjee Department of Electronics and Communication Engineering, JISCE, Kalyani who guided us into the intricacies of this project nonchalantly with matchless magnanimity.

We thank *Dr. Biswarup Neogi* sir, Head-of -Department, ECE JISCE for extending his support and letting us carry out the project along with the coursework.

We would be falling behind in our duty if I don't acknowledge the cooperation rendered during various stages of image interpretation, editing, typing, and printing works.

We are indebted to all our Teachers, supporters, and all others who played their respective roles in the completion of our project report and guided us the way through, for their constant, cooperation and help. Words of gratitude are not enough to describe the accommodation and fortitude that they have shown throughout our Endeavour.

By: -

Akash Dutta

Goutam Sutradhar

Dibyendu Paul

Debabrata Debnath

Paramita Chandra

-: Table of Contents :-

Abstract

In this project, a desktop application has been developed to perform Virtual Laboratory experiments for Electronics and Communication Engineering. The main aim is on simulating various sensors, and digital and analog electronic circuits. This experiment can be performed at home on a PC or laptop. This application can be downloaded to perform virtual experiments. In the application, there are various provisions for setting the parameters and giving specific input values.

When the program is run then the output is generated on a separate LCD screen or in a separate window. The desktop application has been developed strictly in Python language. Tkinter Module is used to build the User interface. Moreover, the experiment simulations are tested with real-life equipment and the results are found to be promising.

❖ *Chapter I:* **Introduction**

1.1. General

COVID-19 is one of the worst stricken pandemics humanity has ever seen. It changed the way society worked. From organizations and institutes to people's day-to-day life it had a substantial impact everywhere. One of the worst-hit segments of society is the student. They are detached from the hands-on practical sessions, essential for any discipline, for a very long time. Of numerous possible ways, one effective solution to make the curriculum more productive, interactive, and easily accessible to students is to switch the methodology and bring the system more to virtual platforms. Even though the Pandemic is slowly turning endemic, and a return to the conventional Education systems is prevailing, the inclusion of Virtual platforms should be cultivated and developed alongside as per requirements.

In this project, we have tried to develop an offline simulation-based virtual lab application for Electronics and Communication Engineering students to obtain technical hands-on knowledge about the experiments.

The desktop application is strictly built in the Python Programming language-based GUI - Tkinter. Other modules that were used are NumPy, Matplotlib, etc. Our primary focus is on the basic electronics Sensors and Digital electronics-based experiments. Although any electronic device can never be replicated to a virtual entity Keeping in mind the intricacies of an electronic system, we have tried to optimize every effective parameter for the experiments to attain the maximum precision possible. For most experiments, we have attempted to validate the results with actual constituents in the lab and obtained promising outcomes.

1.2. About The Project

The principal notion of building the virtual platform is to bring the technical knowledge and experience of Electronics Devices and Circuitry practicals just a few clicks away to the users. The functions, mechanisms, and logical operations of the simulated practicals are so built to replicate the workings of the real elements to the best level. All the Parameters that can determine the output or even have influence over the experiment have been integrated into the interface.

The project on the whole is built using Python Programming Language from scratch. The reason for choosing Python for programming the simulations is because of its easy syntax structure, mathematical operation capabilities, versatility, and extensive library support. The complex calculative operations of the elements and circuits can be easily integrated into the simulations using Python Language.

The primary modules that were used in programming were Tkinter, Numpy, Matplotlib, OS, Canvas, etc. All the stated modules have very distinct functions and their usage has been kept to the minimum possible.

Taking into consideration the problems of connectivity and the easy access of the virtual lab to the students the simulations are made to be operated completely in offline mode and require no installation on the system. The principal focus is to generate commonly used Sensors and Digital logic Simulations in this project, as they form the very base of the Electronics technical knowledge. Also, some analog electronics topics were also integrated. We also look forward to adding more Subjects, topics, and experiments to make headway to the next level of the platform.

❖ *Chapter II:* **Project Discussion**

2.1. Modules and Elements Used

Modular programming is the process of subdividing a computer program into separate sub-programs to make the code easier, understandable, efficient, and developer friendly.

In this project, many pre-built python modules are used for complex computations, plotting output, array operations, and building the UI.

The various modules that were used in the code are Tkinter, Scipy, Numpy, OS, Matplotlib, etc. Tkinter is used to build the User Interface, Scipy, and Numpy for easy computations of the output calculations and other array operations, and Matplotlib for plotting the graphical output. The module OS is used to get the location paths used for importing the circuit diagrams and respective pictures to integrate with the interface.

As the interface was aimed to work in offline mode the modules are also chosen to work in offline mode and don't require any connectivity.

For the majority of the experiments the commands below were used to call upon all the required modules or a part of those:

```
1. from tkinter import *
2. import os.path
3. from tkinter import Canvas,Frame
4. import numpy as np
5. import matplotlib.pyplot as plt
G. from scipy import signal
```

Code Section 1: Importing Modules

Apart from the Modular Programming, various other Graphics elements were also used to make the interface more user friendly and give a pinch of realism to the platform. Some of the elements like on-off switches, circuit diagrams, output bulbs, LEDs, etc. are directly imported as pictures and not created manually.

2.2. User Interface and Design

The user experience of an application depends on the usability, accessibility, credibility and desirability of the Interface. In this project, all the aspects of operational comfort have been taken care of, with respect to the application structure and design.

The application utilizes numerous windows with various geometry to resolve its various needs. Each experiment on the application creates its own separate window for simulation.

For most cases, the main window of the experiment is set to be in Standard Definition (720 x 576) pixels, or a fusion between Standard Definition and High Definition (1280 x 720) pixels. This ensures the proper operation of the program even in small screens. The codes used for building the windows are:

```
1. tk=Tk()
2. tk.title("NAME OF THE WINDOW")
3. tk.geometry("WxH")
4. element.grid(row = 0,column=0,sticky=N,pady=10,pady=10)

                                OR

   element.pack(side=LEFT,expand=True,fill=BOTH)

5. tk.mainloop()
```

Code Section 2: Building UI

The code (1) and (5) helps in creating the required window for the program. Tk() starts the instance for the root window and tk.mainloop() halts the program.

All the other codes that work on the program go in between these two lines. Code (2) is used for setting the name of the experiment that has been simulated and Code (3) helps set the dimension for the window. Here, W and H refer to the width and height of the window in pixels respectively.

Some experiments create separate windows for the output such as PWM simulation. In such cases, the output window is created by separate modules which generate the output eg. Matplotlib. For the rest experiments, the outputs are generated in the main windows. Experiments that require a separate object dragging arena for simulation use the sub-modules Canvas for generating the arena. All the elements engaged by the program are placed on the window using the methods .grid() or .pack(). The program can use only one method at a time
,either .grid() or .pack() and not both.

2.3. Program Structure

The efficiency and developer understandability of a program depends on the Program structure. Programs for such complex simulations and computations require desirable, versatile, and developer-friendly formation for easy readability and proper working.

Program Parts: - The different programs developed in the project follow nearly the same structure for operation. This helped in developing various experiments from the same subject or topic easily and reliably. Based on the need of the experiments each structure is altered accordingly. A typical program structure example is described below:

```
1.   from tkinter import *                                          ┐
2.   ……                                                           │ Part 1
3.   ……                                                           ┘
4.   bool state=True                                               ┐
5.   int x                                                        │ Part 2
G.   ……                                                           │
7.   ……                                                           ┘
8.   def function_1():                                            ┐
9.       conditions and logics:                                   │
10.      configure.element_1(parameters,……,     )                │
         (modify elements already created)                        │ Part 3
11.      calculations:                                            │
12.      ……                                                       │
13.      ……                                                       │
14.      return value                                             ┘
15.  tk=Tk()                                                      ┐
1G.  tk.title("WINDOW NAME")                                      │
17.  tk.geometry("WIDTH x HEIGHT")                                │
18.  ……                                                           │
19.  ……                                                           │ Part 4
20.  element_1=create Button/Label/Slider(command=function_1(…,…,…))  │
21.  element_1.grid()                                             │
22.  ……                                                           │
23.  ……                                                           │
24.  tk.mainloop()                                                ┘
```

Code Section 3: Program Structure

Description:-

Part 1: Part 1 of the structure is used to call and import the modules, submodules, packages, and libraries needed for the program.

Part 2: This part of the program is used to declare variables and set values. Also, all the images, graphics, and other files are loaded in this part and set to variables that are called and used later in the program. The default values of the parameters are also set here.

Part 3: The part of the program consists of all the functions and their operations. All the logic, and conditional statements, for the output and return values are computed here. The button triggers and commands also use the functions that are described here. The truth tables, sensor values, plot generating functions, etc. are described here. This part also creates modifications to the widgets that are generated in part 4.

Part 4: This is one of the main parts of the structure where the user interface is built. Codes for the widgets necessary for the simulation are written here. Elements in this part also call upon the variables and functions described in part 3.The return values originated in part 3 are produced as output through widgets here.

All the structural qualities of the project that are described here are a generalized version of maximum programs and do not strictly follow the same format. Exceptions are acquired in all the programs.

2.4. Widgets Used

Tkinter is Python's standard GUI (Graphical User Interface) package. Tkinter provides a variety of common GUI elements which we can use to build out interfaces – such as buttons, menus, and various kinds of entry fields and display areas.

Label: Tkinter Label is a widget that is used to implement display boxes where one can place text or images. The text displayed by this widget can be modified at any part of the program at any time. It is also used to perform tasks such as underlining the part of the text and spanning the text across multiple lines.

Slider: A slider is a Tkinter object with which a user can set a value by moving an indicator. Sliders can be vertically or horizontally arranged. A slider is created with the Scale method(). allows the user to select a numerical value by moving a knob along a scale of a range of values. The minimum and maximum values can be set as parameters, as well as the resolution.

Button: Button widgets represent a clickable item in the applications. Typically, one can use a text or an image to display the action that will be performed when clicked. The button can also be used to trigger certain functions described earlier. To invoke a function or a method of a class automatically when the button is clicked, you assign its command option to the function or method. This is called the command binding in Tkinter.

Canvas: The Canvas widget supplies graphics facilities for Tkinter. Among these graphical objects are lines, circles, images, and even other widgets. With this widget, it's possible to draw graphs and plots, create graphics editors, and implement various kinds of custom widgets. The canvas has two coordinate systems: the window system (left top corner x=0,y=0) and the canvas coordinate system that defines where items are drawn.

Spinbox: Python Tkinter Spinbox is a type of entry widget with up-down arrows. A range of data can be put inside the spinbox and the user can navigate using the up & down arrow. This widget is an alternative to the Entry widget when the user wants to enter a numeric value within a specific range.

❖ *Chapter III:* **Simulation**

3.1. Sensor Simulation

In this project various sensor simulations have been built like the Flame Sensor, Infrared sensor, TMP36 sensor, LM35 sensor, Soil Moisture Sensor, etc. Each sensor is assigned a separate interface for simulation.

Inputs or triggers for every sensor are very different. Various ways for giving inputs to sensors are: setting slider values, mouse movements in the canvas arena, button triggers, etc. For example, the Infrared sensor simulation uses Mouse movement to set the object location which is being detected by the sensor. Similarly, in the soil moisture sensor, the moisture level value and the threshold are set using two separate sliders.

The input values once set are then fetched and fed to output generating functions. The outputs are shown using output LEDs, Microcontroller screens, and Multimeter Screens based on the sensor.

The output generating functions are so developed to replicate the real sensors.In maximum cases, the same equations which are used to generate outputs in real sensors are used but in a reversed way to form the same inputs. For example, in the case of a real TMP36 sensor the output is generated with respect to the analog pin reading using the following equations:

Temperature=(analogRead/1024*5)-0.5)*100

The equation used to print the analog read in the simulation interface from room temperature is

analogRead=((Temperature/100)+0.5)/5*1024

For other sensors which do not follow specific relation with respect to the varied inputs arbitrary equations are formed based on the graphs they form in input vs output. The equations are built to follow the graphs types like linear, logarithmic, quadratic, etc. within the same range.

3.2. Logic Simulation

The simulation for some digital electronics lab experiments like encoder, decoder, mux, demux, etc. require logic building functions for proper output generation. These logic functions form the very base of both Combinational and Sequential Circuits.

Combinational Circuit:- A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs and they have no memory. Combinational Circuits are In combinational circuits the inputs are first taken and are registered to variables. The variables are then fed to the functions created specifically for that logic circuit, as parameters. For logics that can be easily represented using Boolean Functions, the output generation functions use those boolean functions to determine when the output turns high(1). Those high signals are simulated by the output simulation devices such as Microcontroller screens, Multimeter screens, LEDs, etc. But for those logic circuits which require a complex or excessive number of boolean functions compared to the number of inputs/output, instead of functions, they are fed with output data for each particular input using If….Else statement to generate outputs. These outputs are then simulated by the output devices.

Sequential Circuits:- A sequential circuit is specified by a time sequence of inputs, outputs, and internal states. The output of a sequential circuit depends not only on the combination of present inputs but also on the previous outputs. Unlike combinational circuits, sequential circuits include memory elements with combinational circuits.For sequential circuits along with logic functions, memory elements are also created using Python lists. These lists act as memory elements to store required inputs and also take out them whenever specified. The inputs along with the memory values are fed to the Output generating functions to get the outputs. The output devices simulate the output value when next triggered.

3.3. Memory Elements

Some circuit simulations, especially Digital sequential circuits, require special memory elements to store the input data. The outputs of such circuits depend on both present and past inputs. To attain these requirements special functions were to work as a legitimate memory.

For Simple sequential circuits like SR flip flop, JK flip flop, D flip flop, etc no special functions for memory elements are required. These applications can be built to work on simple truth tables. But for complex circuits like shift registers - PISO, PIPO, SISO, and SIPO special memory functions were needed as they first store all the data as a queue and then the outputs move out one by one.

The memory functions can be easily accomplished using the concept of Queue using Python. This can be easily implemented using the queue class.

Firstly Using python an empty list is created when the program is initiated using the function “list_1=[]”

Every time the circuit is in LOAD mode and a new input (high or low) is loaded a new element is appended to the queue using the “.append()” command. With Each clock trigger, a number of elements are added to the list, based on the register type that is being simulated.

When the circuit mode is set to READ mode, with each clock trigger one or more elements from the last index of the list (first entered) are read and then taken out using the ".pop(-1)” function.

The input and output elements are only 0 and 1, 0 refers to low state and 1 refers to a high state. If the read element is high then the LED glows else the LED remains off. In case the built queue is empty the LED remains turned off as default.

❖ *Chapter IV:* **Testing**

4.1. Test and Validation

Testing is an integral part of project development. It helps to eliminate errors and bugs. In this case, it helped to eliminate the bugs in both software and simulation result levels and made the platform act more real. For the developed project two levels of testing have been performed. Those are:

Simulation Testing: In this level of testing the accuracy of the sensor simulated outputs are being compared with the outputs of the actual sensors. For reference, the simulation of the Pulse width Modulation generation experiment was tested by generating the Pulse width Modulation using an Arduino. Both the simulations when compared produced almost the same output and no discrepancies were found. A snap of the generated wave is attached.

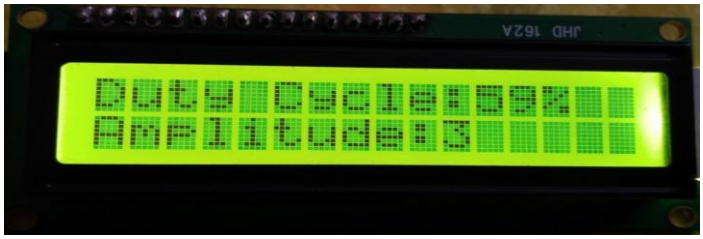

Figure 1: Microcontroller Screen

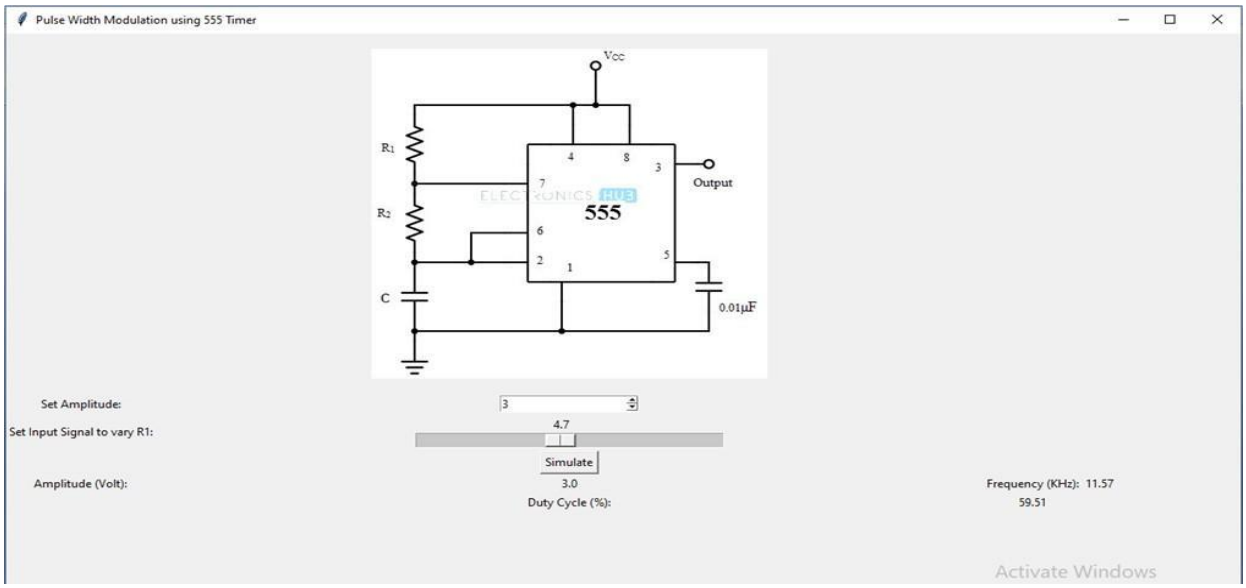

Figure 2: Simulation Window for PWM

Similarly for other sensor simulations, testing was done with the same input levels and their output pin volts were compared but all seemed to produce promising results at any defined range of inputs. All the experiments produced extremely similar results.

Software Testing: In this part, all the software-related tests were done. The platform developed for simulation was tested on various devices considering multiple parameters. The various tests were to check compatibility to various display dimensions, program functioning in multiple devices with various specifications, Working in low-end devices, Multiple programs operating at a time, etc. All the tests were concluded with promising results.

❖ *Chapter V*: **Experiment Simulation Examples**

5.1. Pulse Width Modulated Wave Generation

Discussion:- With regard to this Experiment, a simulation has been developed to perform a virtual laboratory experiment for the generation of Pulse Width Modulation (PWM). There are provisions for parameter selection to modify the pulse width of a square pulse. When the program is brought into action there is a primary window in which the user is able to set the amplitude and the value of the resistor "R_1" and the waveform is generated in a separate window.

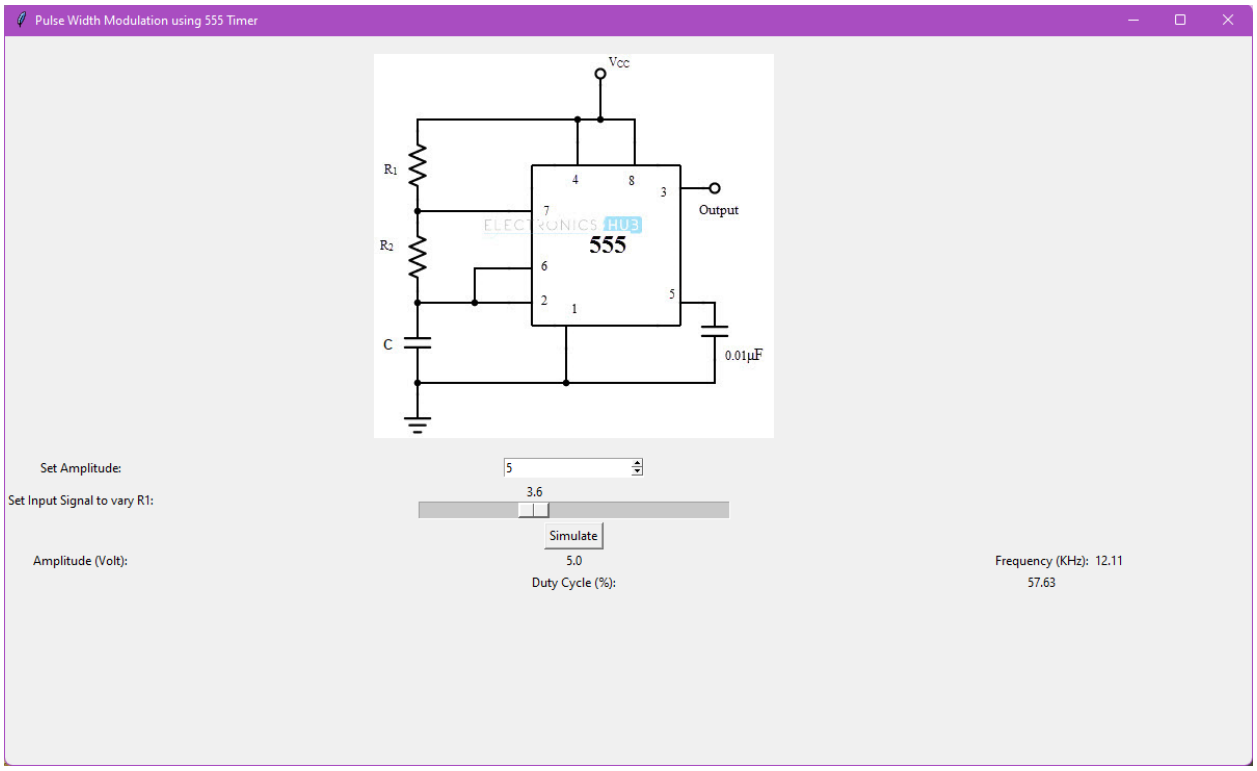

Figure 3: Simulation Window for PWM

555 Timer IC:- In this application, the pulse width modulation is designed to generate using 555 Timer IC. In the 555 timer circuit, a square wave can be generated with the time period given by the following equation.

$T=0.69*(R_1+(2*R_2))*C$

Where R_1 and R_2 are the values representing the two resistors respectively. C represents the value of the capacitor on the circuit.

Here the high and the low time intervals can be expressed with the following equations as follows.

$T_{HIGH} = 0.69*(R_1+R_2)*C$

$T_{LOW} = 0.69*R_2*C$

There is another term in PWM which is very important and that is the duty cycle. The duty cycle can be defined as follows.

Duty Cycle$=T_H/T\times100\%$

$=(0.69*(R_1+R_2)*C)/(0.69*(R_1+2*R_2)*C)*100\%$

$=((R_1+R_2))/((R_1+2*R_2))*100\%$

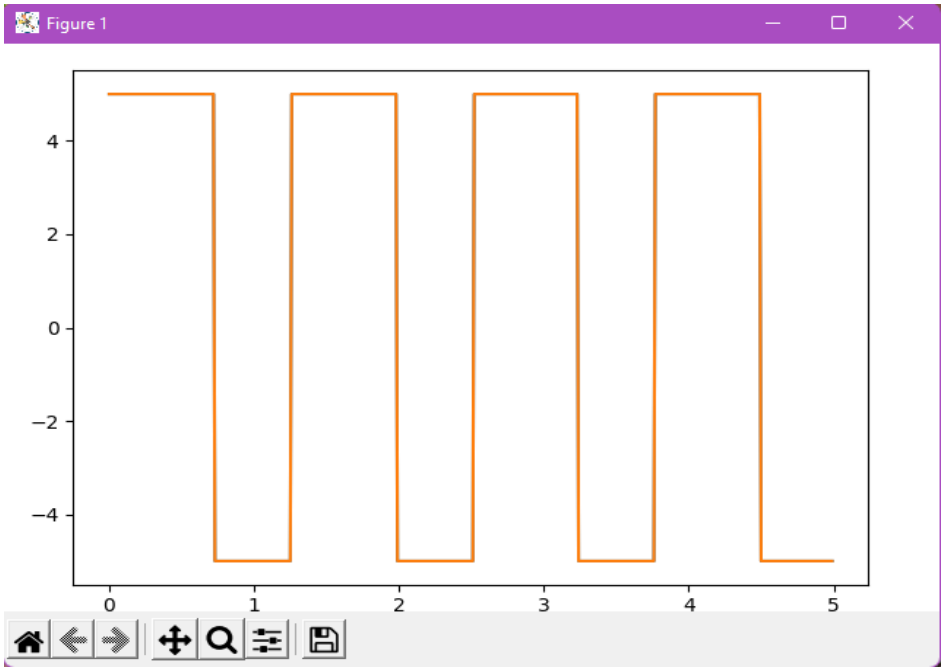

Figure 4: Output Window for PWM

To achieve a perfect square wave, R_2 is taken to be far greater than R_1 i.e.. But in PWM, we need to vary the pulse width. Therefore R_1 needs to be varied. Therefore in the application, there is a provision for varying R_1.

The value of the amplitude required for generating the PWM wave can be set using the spinbox. The Resistor R_1 can be varied using the slider from a range of 0.1 kΩ to 10 kΩ. Finally, the wave can be generated by triggering the simulate button which will then generate a separate window for the constructed wave. The output can be moved, zoomed in and out, and saved using the respectively provided buttons alongside the output window.

Along with the graphical output we also get the Duty Cycle % and frequency of the generated wave as output in the main window.

5.2. Flame Sensor

A flame sensor is designed to detect and respond to the presence of a flame or fire, allowing flame detection. Infrared or wideband infrared flame detectors monitor the infrared spectral band for specific patterns given off by hot gases.

Discussion:- The interface for the Flame sensor consists of a circuit diagram, a microcontroller screen, an output bulb, and an arena for placing the flame. In the arena, the user is able to move the flame using the pointer with regard to the distance from the sensor head. The inference from the calculated output is then shown in the Microcontroller Display.

Two separate switches are deployed in the simulation, one works as an on/off switch and the latter is used as a trigger to manually give high input to the circuit.

The circuit used in the Flame sensor has an IR sensor that constantly detects the presence of infrared radiations. Basically, it can be used in two modes - Digital and Analog mode. In digital mode, the microcontroller is programmed to detect if there is any fire or not. If the circuit detects flame the output is (1) else it responds as (0). Whereas in Analog mode the circuit gives an Analog value between 0 to 1024 based on the intensity of the infrared rays obtained. The intensity increases based on the proximity of the flame.

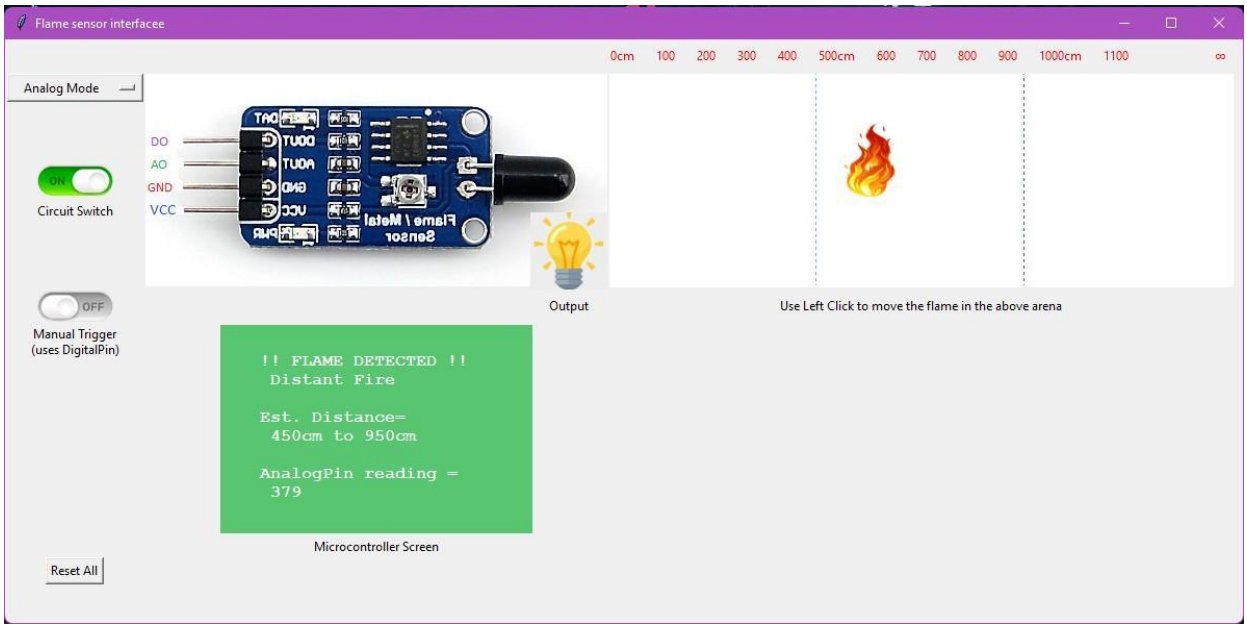

Figure 5: Simulation Window for Flame Sensor (Analog Mode)

A dedicated “Reset All” button is also provisioned which easily resets all input and output values to default, to avoid the hindrance of closing the application and re-running it again.

Analog Mode:- In analog mode the arena is divided into three parts based on the distance from the sensor (0cm – 450cm, 450cm – 900cm, 900 – ∞). Depending on the pointer location in the arena the analog pin output value is generated which ranges from 0 to 1024 and the distance is then calculated between the flame and sensor. Based on the distance the inference is displayed on the Microcontroller screen.

Digital Mode:- Whereas in the Digital mode the arena for the flame is divided only into two parts namely - Detectable range and Undetectable range. If the flame is in detectable range the output is 1 else the output is 0.

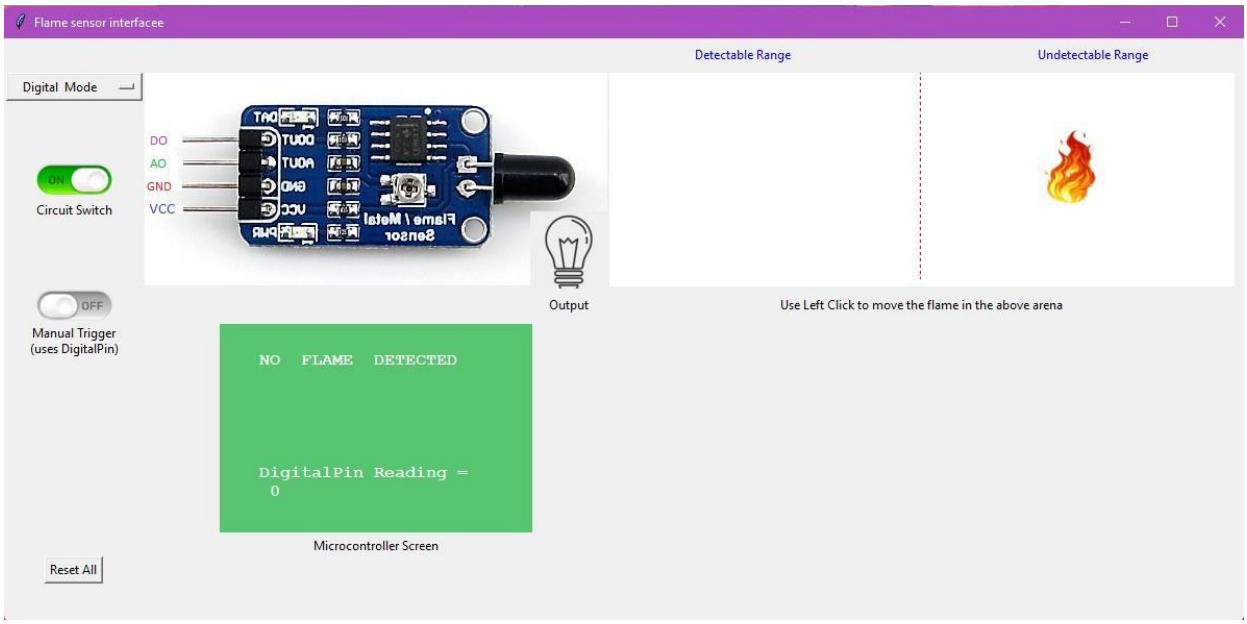

Figure 6: Simulation Window for Flame Sensor (Digital Mode)

Apart from the microcontroller screen output a separate bulb output is also provided to get digital feedback based on the output of the circuit.

5.3. BCD to 7-Segment Display

In this experiment, we simulate a circuit using IC7447 to convert Binary Coded Decimals (BCD) numbers to Decimal Numbers using a 7-Segment Display. A seven-segment display is a digital module specialized to present numerical information through LEDs arranged in the shape of numbers.

Discussion:- The IC7447 converts 4-bit BCD to 8-bit values, where the output of the decoder is connected to that of the seven segments.

IC7447:-The IC7447 has 16 pins, out of which 4 (A_0, A_1, A_2, A_3) take the 4-bit BCD numbers. The inputs $\overline{BI}$ /, $\overline{RBI}$ and $\overline{LT}$ are usually connected to 5v. There are two other pins for the VCC and Ground. The other 7 pins (a, b, c, d, e, f, g) go to the 7-segment display as inputs.

7-Segment Display:-The 7 segment display has a total of 10 inputs, where 8 of the 10 inputs to the display correspond to a LED segment. Pin 3 and 8 of the 7-segment Display are internally connected to form a common pin. This pin is connected to GND or 5V depending upon the type of the display. Out of 10, the 8 pins (a, b, c, d, e, f, g, and DP) are connected to digital pins of the IC7447. By controlling each LED on the segment connected, numbers can be generated.

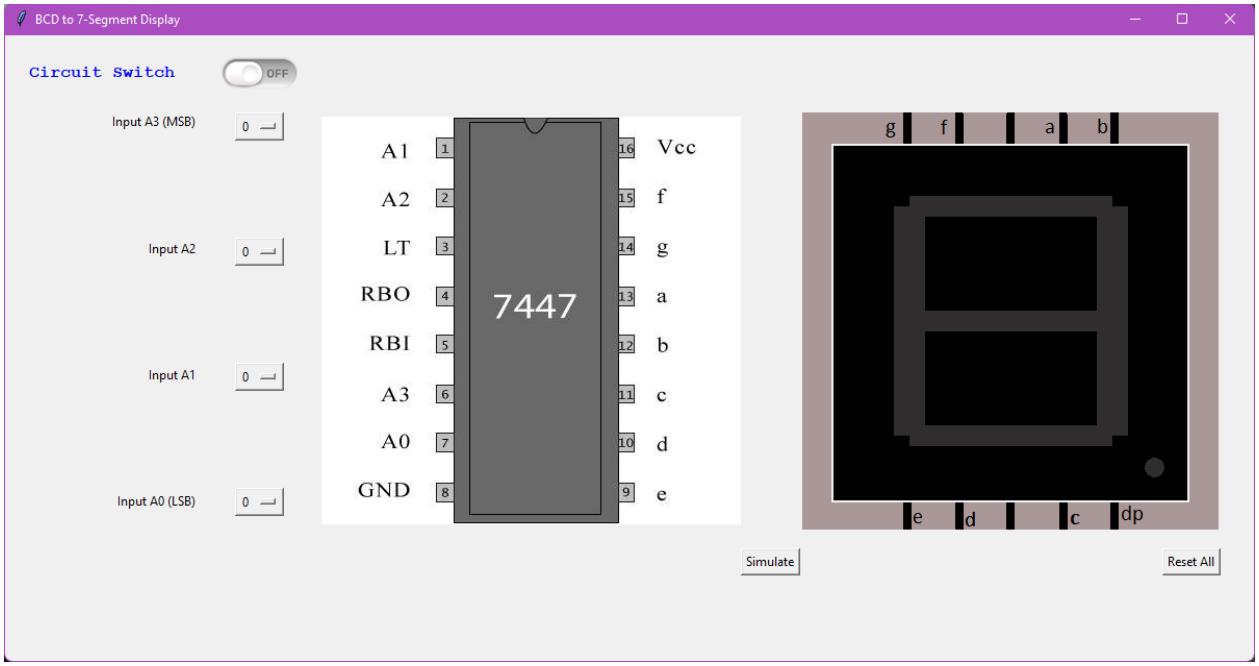

Figure 7: Simulation Window for BCD to 7-segment Display

With 4-bits we can form 16 Decimal numbers (0-15), but in this simulation, we have omitted the numbers from 10 to 15 as those require two separate 7-segment displays for representation.

Decimal Digit	Input Lines				Output Lines							Display Pattern
	A_0	A_1	A_2	A_3	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	0
1	0	0	0	1	0	1	1	0	0	0	0	1
2	0	0	1	0	1	1	0	1	1	0	1	2
3	0	0	1	1	1	1	1	1	0	0	1	3
4	0	1	0	0	0	1	1	0	0	1	1	4
5	0	1	0	1	1	0	1	1	0	1	1	5
6	0	1	1	0	1	0	1	1	1	1	1	6
7	0	1	1	1	1	1	1	0	0	0	0	7
8	1	0	0	0	1	1	1	1	1	1	1	8
9	1	0	0	1	1	1	1	1	0	1	1	9

Table 1: Truth Table for BCD to 7-segment Display

The relation between the inputs (BCD) and the outputs (7-segment display) can be generated from the Truth Table.

- a = F1 (A_0, A_1, A_2, A_3) = $\sum$m (0, 2, 3, 5, 7, 8, 9)
- b = F2 (A_0, A_1, A_2, A_3) = $\sum$m (0, 1, 2, 3, 4, 7, 8, 9)
- c = F3 (A_0, A_1, A_2, A_3) = $\sum$m (0, 1, 3, 4, 5, 6, 7, 8, 9)
- d = F4 (A_0, A_1, A_2, A_3) = $\sum$m (0, 2, 3, 5, 6, 8)
- e = F5 (A_0, A_1, A_2, A_3) = $\sum$m (0, 2, 6, 8)
- f = F6 (A_0, A_1, A_2, A_3) = $\sum$m (0, 4, 5, 6, 8, 9)
- g = F7 (A_0, A_1, A_2, A_3) = $\sum$m (2, 3, 4, 5, 6, 8, 9)

An On/off Switch button is provided which acts as a Circuit button for the whole circuit and controls the power supply.

On the interface the inputs can be given using the four drop down menus (A_0,A_1,A_2,A_3). These inputs are computed and then mapped to the LEDs of the 7 segment display. The output can be then easily generated using the "Simulate" button.

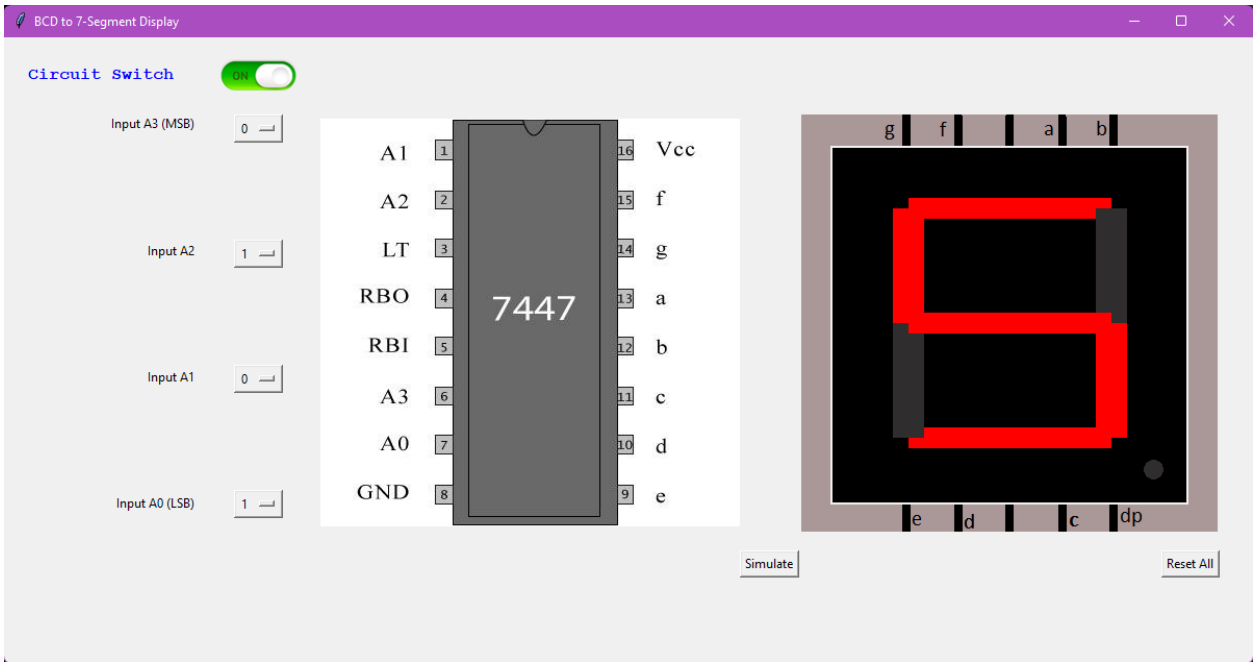

Figure 8: Simulation for BCD to 7-segment Display (Circuit on)

A dedicated "Reset All" button is also provisioned on the interface to reset all input and output values, so that the user does not have to re-run the program to get a fresh start.

5.4. 1:4 Demultiplexer using logic gates

A Demultiplexer is a combinational logic circuit that receives the information on a single input line and transmits the same information over one of ‘n’ possible output lines based on the Select Line inputs.

Discussion:- In this Experiment Simulation two NOT gates and four 3input AND gates are used to form a 1:4 demultiplexer. Based on the input and the select line states, the output of a particular data line is decided . If the output is high (1) the output LED turns Green or else the LED remains Red. A dedicated “Circuit Switch” is provisioned to manage the power supply for the circuit. If the circuit button is turned off , even though the inputs may not be low (0), the circuit will not yield any output and thus all the LEDs will stay low (0) or red.

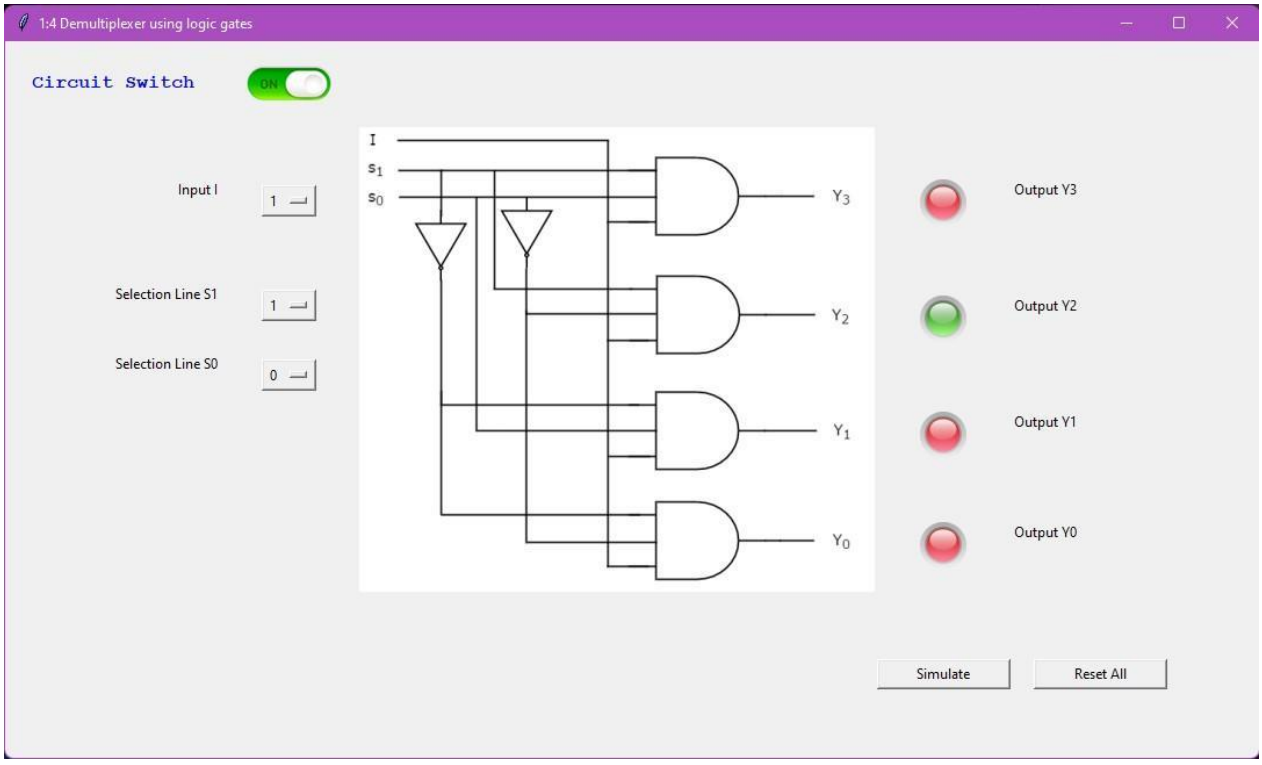

Figure 9: 1:4 Demultiplexer

The 3 inputs namely I ,S_0 and S_1 are accepted through the 3 dropdown menus provided on the left side of the interface .The Simulate command triggers the demux function which maps down the three inputs and generates the output for each of the data lines Y_0,Y_1,Y_2 and Y_3. By default all the LEDs are set to low (0) and remain red . If the output generated by the demux function for any LED is high (0) then the LED turns green while the rest remains red.

Truth Table and Expressions:- For mapping the input data to the output pins the Truth table for 1:4 demultiplexer is used:

S_1	S_0	I	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	0	0
0	0	1	0	0	0	1
0	1	0	0	0	0	0
0	1	1	0	0	1	0
1	0	0	0	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	0	0
1	1	1	1	0	0	0

Table 2: Truth Table for 1:4 Demultiplexer

The boolean Expression for each of the output are:

- $Y_0 = S_1' S_0' I$
- $Y_1 = S_1' S_0 I$
- $Y_2 = S_1 S_0' I$
- $Y_3 = S_1 S_0 I$

The LEDs turn green whenever the result for the above equations yields high (1).

A dedicated “Reset All” button is also provisioned on the right side of the interface to easily reset all input and output value to default low (0), re-initiate the demux function and to avoid the hindrance of closing the application and re-running it again.

5.5. Parity Bit Generator

A Parity Generator is a combinational logic circuit that generates the parity bit in the transmitter. On the other hand, a circuit that checks the parity in the receiver is called a Parity Checker.

Discussion:- In this Experiment Simulation, three XOR gates and four input gates are used to form a parity generator. Based on the input and the select line states, the output of a particular data line is decided. If the output is high (1) the output LED turns Green or else the LED remains red.

A dedicated "Circuit Switch" is provisioned to manage the power supply for the circuit. If the circuit button is turned off, even though the inputs may not be low, the circuit will not yield any output and thus all the LEDs will stay low (0) or red.

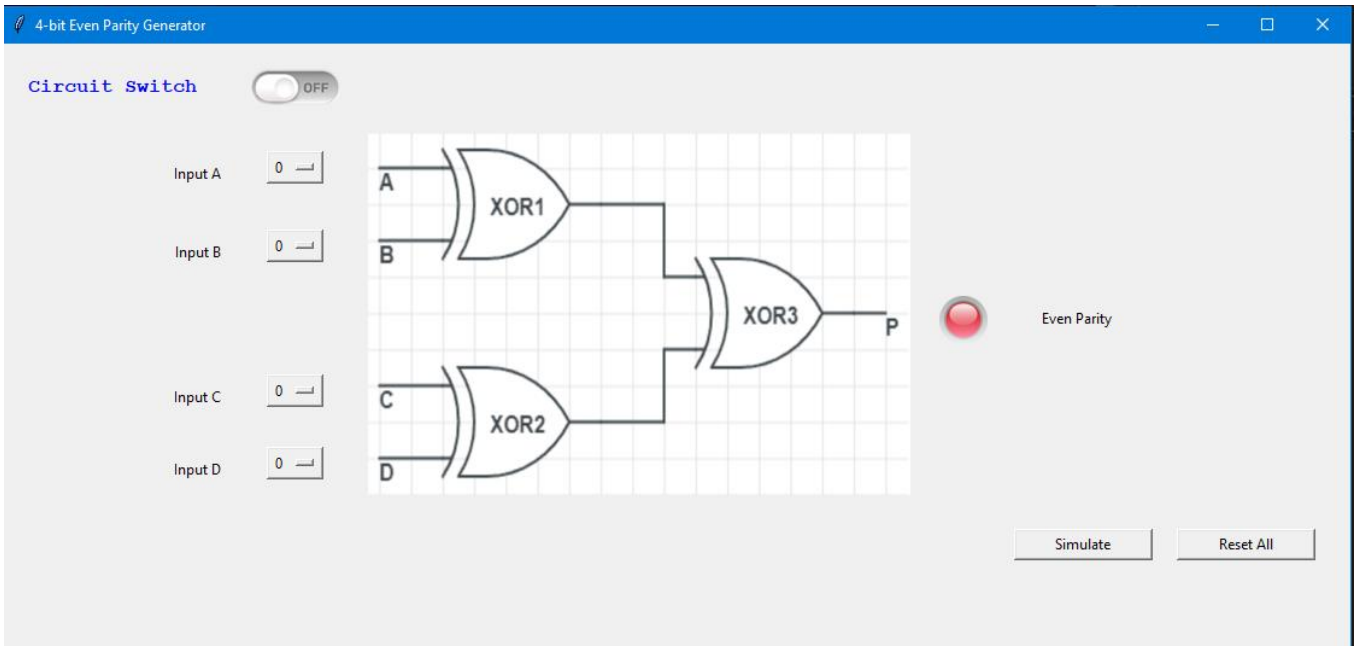

Figure 10: Even Parity generator

The 4 inputs namely A, B, C, and D are accepted through the dropdown menus

provided on the left side of the interface. The Simulate command triggers the function which maps down the four inputs and generates the output for the data line

P. By default all the LEDs are set to low (0) and remain red. If the output generated by the function for any LED is high, then the LED turns green while the rest remains red.

Truth Table and Expressions:-

A	B	C	D	P
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

Table 3: Truth Table for even parity generator

The Boolean Expression for each of the output are:

- $P = A \oplus B \oplus C \oplus D$

The LEDs turn green whenever the result for the above equations yields high (1).

A dedicated “Reset All” button is also provisioned on the right side of the interface to easily reset all input and output values to default low (0), re-initiate the function, and avoid the hindrance of closing the application and re-running it again.

5.6 JK Flip-Flops

A circuit that has two stable states is treated as a flip-flop. These stable states are used to store binary data that can be changed by applying varying inputs. The flip flops are the fundamental building blocks of the digital system. Flip flops and latches are examples of data storage elements. In the sequential logical circuit, the flip flop is the basic storage element. The latches and flip flops are the basic storage elements but are different in working. There are different types of Flip-flops available in the application but here we have used JK Flip-Flop to show the example.

Discussion:- In this Experiment Simulation four NAND gates and two inputs and a CLK are used to form a JK flip-flop. In J-K flip flop, if both of its inputs are different, the value of J at the next clock edge is taken by the output Y. If both of its input is low, then no change occurs, and if high at the clock edge, then from one state to the other, the output will be toggled. The JK Flip Flop is a Set or Reset Flip flop in the digital system. Based on the input and the select line states, the output of a particular data line is decided. If the output is high (1) the output LED turns Green or else the LED remains red.

A dedicated “Circuit Switch” is provisioned to manage the power supply for the circuit. If the circuit button is turned off, even though the inputs may not be low, the circuit will not yield any output and thus all the LEDs will stay low (0) or red.

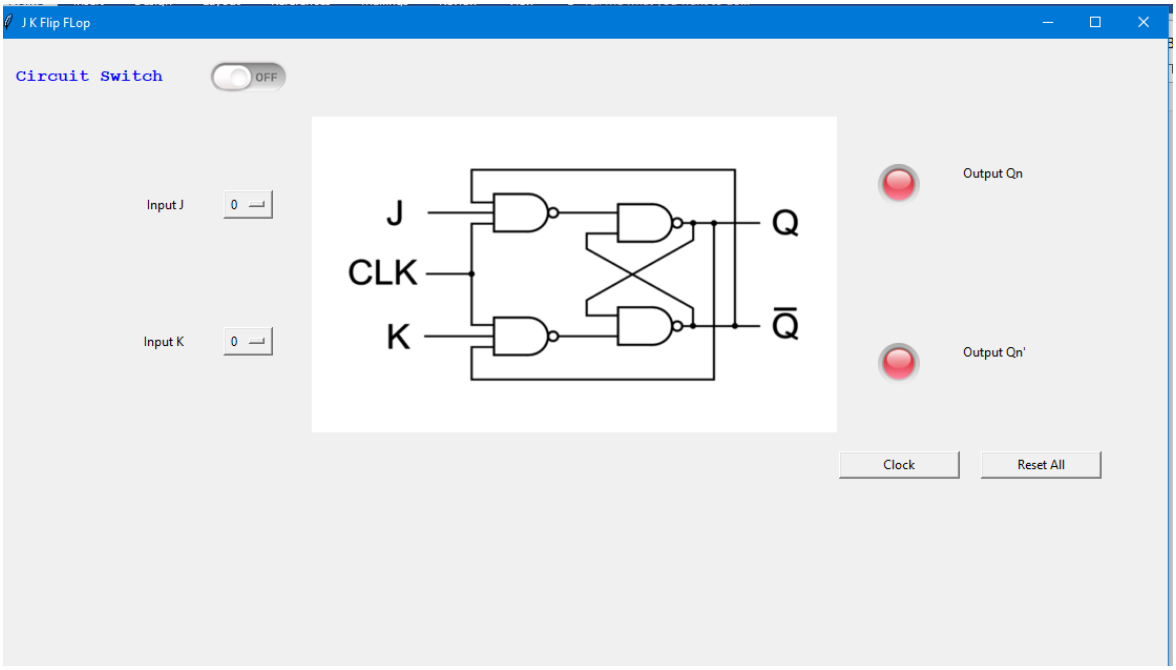

Figure 11: JK flip flop

The 2 inputs namely J, K are accepted through the dropdown menus provided on the left side of the interface. The Simulate command triggers the function which maps down the two inputs and generates the output for the data lines Qn and Qn'. By default, all the LEDs are set to low (0) and remain red. If the output generated by the function for any LED is high, then the LED turns green while the rest remains red.

The LEDs turn green whenever the result for the above equations yields high (1).

A dedicated "Reset All" button is also provisioned on the right side of the interface to easily reset all input and output values to default low (0), re-initiate the function, and avoid the hindrance of closing the application and re-running it again.

Truth Table and Expressions:-

J	K	CLK	Qn	Qn’
0	0	HIGH	0	0
0	0	LOW	1	1
0	1	HIGH	0	0
0	1	LOW	1	0
1	0	HIGH	0	1
1	0	LOW	1	1
1	1	HIGH	0	1
1	1	LOW	1	0

Table 4: Truth Table for JK flip flop

❖ *Chapter VI:* **Conclusion**

6.1. Discussion and Summary

In this project, a virtual platform for Electronics experiments simulation have been developed. A desktop application was developed to create a virtual laboratory that can be accessed offline by the students. Moreover, the reading of the experimental data was compared with the real-life output values of the sensors, which came out similar to that of the simulated.

The application was developed fully in the python programming language, and Tkinter and other modules were used to build the interface. Outputs generated during simulation are shown in simulated LCD screens or in separate windows. The work was totally new to the best of the knowledge of the contributors and so could not be compared with any other existing works. Although utmost care has been taken to replicate the original experiments and results there may be some small discrepancies in results due to undesired errors.

6.2. Future Prospects

The project has been developed keeping in mind the necessities of a student who was otherwise barred from the conventional Lab curriculums during the pandemic situation. Along with this, the platform can also be used by teachers for presentation and explanation purposes. This can benefit a broad spectrum of students and teachers to make them more familiar with the virtual world.

It is a small-sized offline simulation-based application for performing regular College level Electronics experiments. But if needed this can also be integrated into any site as an online-based service (server-based).

The key potential of the project lies in its flexibility, wide usability, and precise and non-erroneous outputs.

With further developments, the application can be easily modified for experiments on other domains and subjects.

Fine tuning the experiments can lead to more realistic outcomes and more precise outputs can be generated.

The application can also be integrated with hardware like Arduino or raspberry pi so that they can act as dedicated experiments kits for the users. This is the initial phase of this idea, so there is always hope to make it more fine-tuned and eliminate the bugs .

6.3. Bibliography

1. IIT Kharagpur, "Virtual Lab," [Online]. Available: https://www.vlab.co.in/participating-institute-iit-kharagpur.

2. IIT Guwahati, "Virtual Lab," [Online]. Available: https://www.vlab.co.in/participating-institute-iit-guwahati.

3. IIT Bombay, "Virtual Lab," [Online]. Available: http://vlabs.iitb.ac.in/vlab/.

4. Python Tkinter GUI Documents, "Python Docs," . Available: https://docs.python.org/3/library/tkinter.html

5. Sensor Knowledge, "Last minute Engineers," [Online]. Available: https://lastminuteengineers.com/

6. Bug Solving, "Stackoverflow" [Online].Available: https://stackoverflow.com/

6.4. About the Contributors

Akash Dutta student of fourth-year student in the Department of Electronics and Communication Engineering, JIS College of Engineering. His research interest is in Automation and Embedded Systems.

Debabrata Debnath student of fourth-year student in the Department of Electronics and Communication Engineering, JIS College of Engineering. His research interest is in IoT, Automation, and Embedded Systems.

Gautam Sutradhar student of fourth-year student in the Department of Electronics and Communication Engineering, JIS College of Engineering. His research interest is in Robotics, Automation, and Python Programming.

Dibyendu Paul student of fourth-year student in the Department of Electronics and Communication Engineering, JIS College of Engineering. His research interest is in Automation and Embedded Systems.

Paramita Chandra student of fourth-year student in the Department of Electronics and Communication Engineering, JIS College of Engineering. Her research interest is in Automation and Embedded Systems.

Arindam Banerjee received the Ph.D., M.Tech, and B.E. degrees respectively in Computer Science and Engineering, Electronics and Communication Engineering, and Electronics and Communication Engineering from Jadavpur University, Maulana Abul Kalam Azad University of Technology, and Burdwan University respectively in 2019, 2008 and 2004, respectively. His research interest is in VLSI, Computer Architecture, Reversible Logic, and Computing, etc.