SRM Institute of Science and Technology

College of Engineering and Technology

Department of Electronics and Communication Engineering

18ECO109J-Embedded System Design Using Raspberry Pi
2023-2024 (Odd Semester)

Mini Project Report

Name	**: Kamya Ojha**
Register No.	**: RA2111003010343**
Day / Session	**: 3**
Venue	**: TP1117**
Project Title	**: WEB-ENABLED RASPBERRY PI DOOR LOCK**
Lab Supervisor	**: Suganthi Brindha G**
Team Members	**: 1)Ganesh K (RA2111003010298)**
	2)Palaash Surana (RA2111003010319)
	3) Kamya Ohja (RA2111003010343)

Particulars	**Max. Marks**	**Marks Obtained**
Objective & Description	05	
Algorithm,Flowchart,Program	20	
Demo verification	10	
Viva	10	
Report	05	
Total	**50**	

REPORT VERIFICATION

Date :

Staff Name :

WEB-ENABLED RASPBERRY PI DOOR LOCK

OBJECTIVE:

To create a web-controlled door lock system using a Raspberry Pi, solenoid lock, and Flask web framework for enhanced home security and convenience.

ABSTRACT:

This project focuses on enhancing home security and convenience through the implementation of a Wi-Fi-controlled door lock system. Leveraging the capabilities of a Raspberry Pi and the Flask web framework, we have developed a user-friendly web interface accessible from anywhere in the world, provided port forwarding is enabled on the router. This system utilizes a 12V solenoid lock, a relay module, and various jumper wires to automate the operation of the door lock.

Solenoid locks, as opposed to traditional key-based locks, enable automatic latch control by applying voltage. The low-voltage solenoid activates to pull the latch, securing or releasing the door, with the latch maintaining its position until the next activation. The solenoid lock in this project operates at 12V, although it can function at 9V, albeit with slower response times. Such locks find applications in remote areas where automation is desired without requiring manual effort.

The circuit connection between the Raspberry Pi and the solenoid door lock is straightforward. Given that the solenoid lock requires 12V to operate and the Raspberry Pi GPIO pins provide only 3.3V, we use an external 12V power source in conjunction with a relay module to trigger the lock. The GPIO 18 pin of the Raspberry Pi connects to the input pin of the relay module, while the

VCC and GND pins of the relay module are linked to the Raspberry Pi's 5V and GND pins, respectively. On the other side, the solenoid lock's GND pin connects to the Common (COM) terminal of the relay module, and its Positive pin connects to the 12V power supply's positive terminal. The negative terminal of the 12V power supply is connected to the Normally Open (NO) terminal of the relay module.

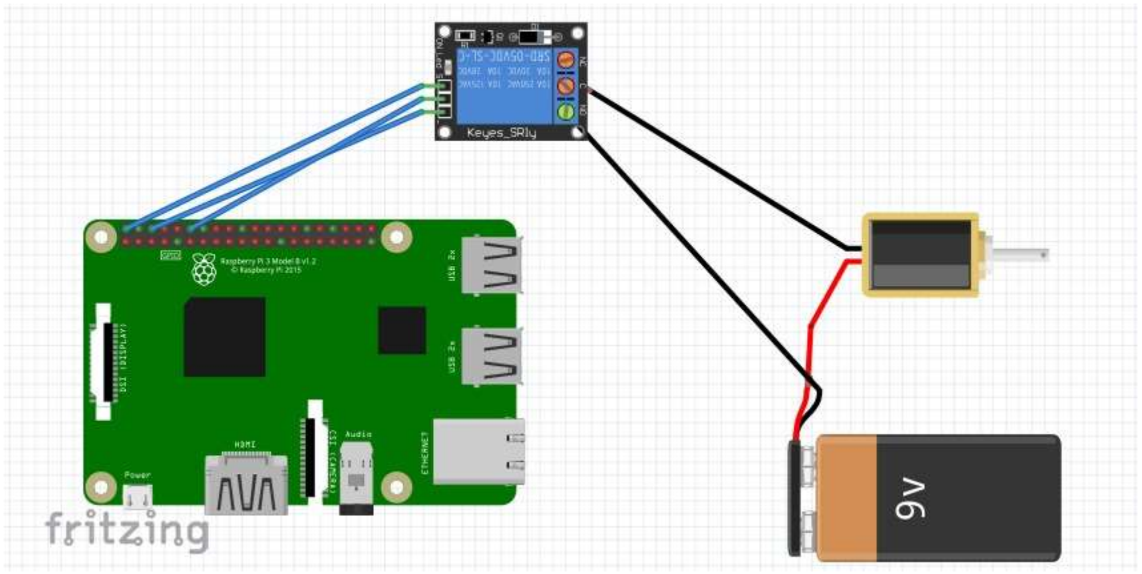
This project exemplifies the fusion of Raspberry Pi's robust processing capabilities and built-in Wi-Fi functionalities with the Flask web framework, enabling users to remotely control a solenoid door lock via a simple web interface. The result is an innovative solution that combines security and automation for modern homes.

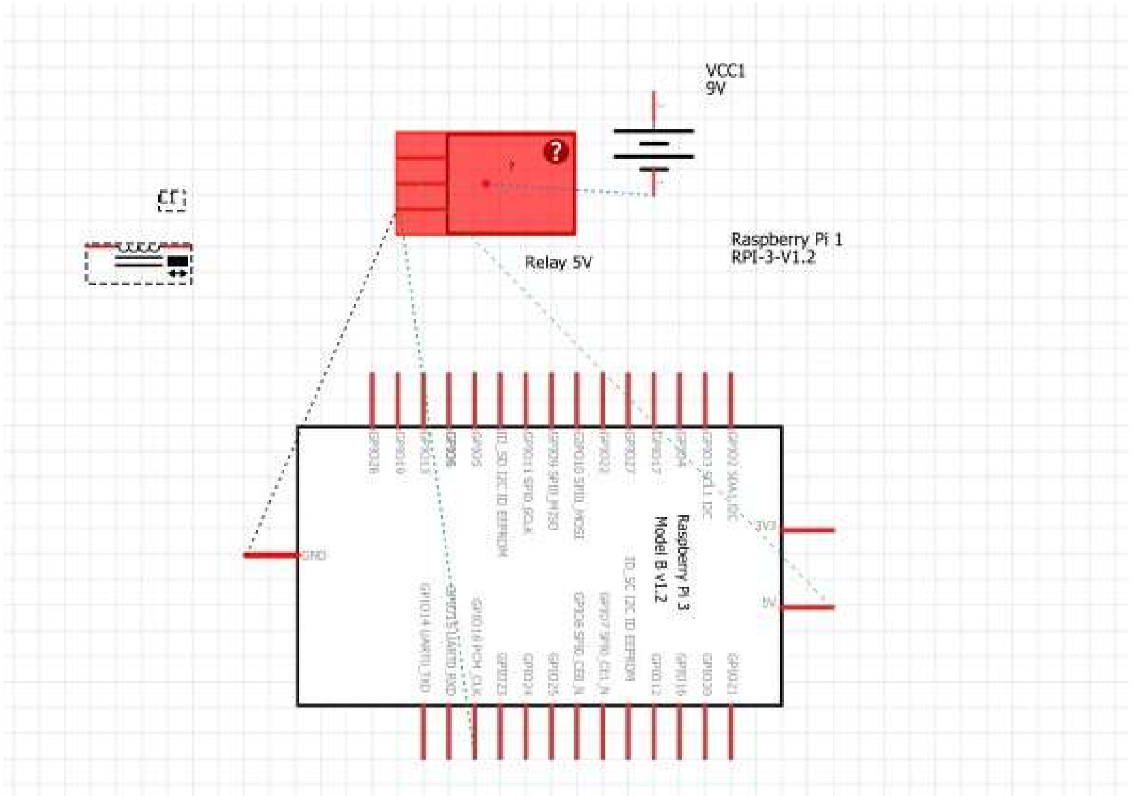
HARDWARE / SOFTWARE REQUIRED:

- Raspberry Pi 4
- M/M and M/F jumper wires
- Relay Module
- 12v Solenoid Lock

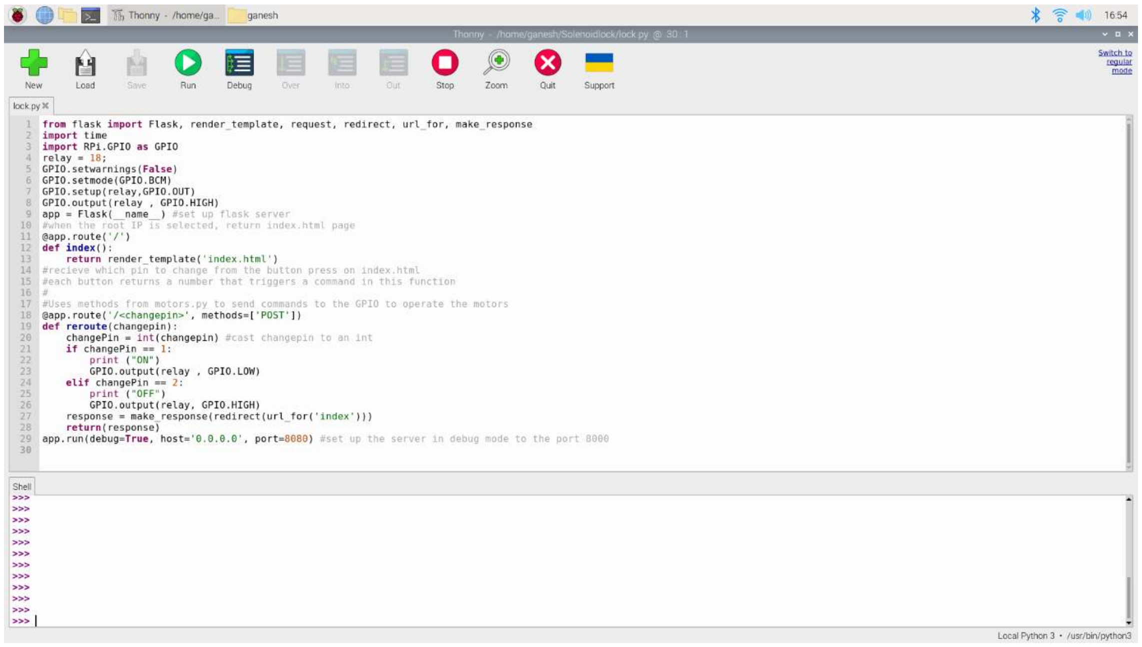
BLOCK DIAGRAM/CONNECTION DIAGRAM:

The connection diagram illustrates the interface between the Raspberry Pi and a 12V solenoid lock, facilitated by a relay module, enabling remote control via a web interface.It showcases the integration of low-voltage GPIO pins, a relay module, and an external power source to efficiently operate the solenoid lock while ensuring security and convenience.


Keyes_SRly
SRD-05VDC-SL-C
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
ETHERNET
DSI (DISPLAY)
CSI (CAMERA)
HDMI
Audio
Power
9V
fritzing


VCC1
9V
Relay 5V
Raspberry Pi 1
RPI-3-V1.2
Raspberry Pi 3
Model B v1.2
GND
3V3
5V

CODE: (Python)



```python
from flask import Flask, render_template, request, redirect, url_for, make_response
import time
import RPi.GPIO as GPIO
relay = 18;
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(relay,GPIO.OUT)
GPIO.output(relay , GPIO.HIGH)
app = Flask(__name__) #set up flask server
#when the root IP is selected, return index.html page
@app.route('/')
def index():
    return render_template('index.html')
#recieve which pin to change from the button press on index.html
#each button returns a number that triggers a command in this function
#
#Uses methods from motors.py to send commands to the GPIO to operate the motors
@app.route('/<changepin>', methods=['POST'])
def reroute(changepin):
    changePin = int(changepin) #cast changepin to an int
    if changePin == 1:
        print ("ON")
        GPIO.output(relay , GPIO.LOW)
    elif changePin == 2:
        print ("OFF")
        GPIO.output(relay, GPIO.HIGH)
    response = make_response(redirect(url_for('index')))
    return(response)
app.run(debug=True, host='0.0.0.0', port=8080) #set up the server in debug mode to the port 8000

```

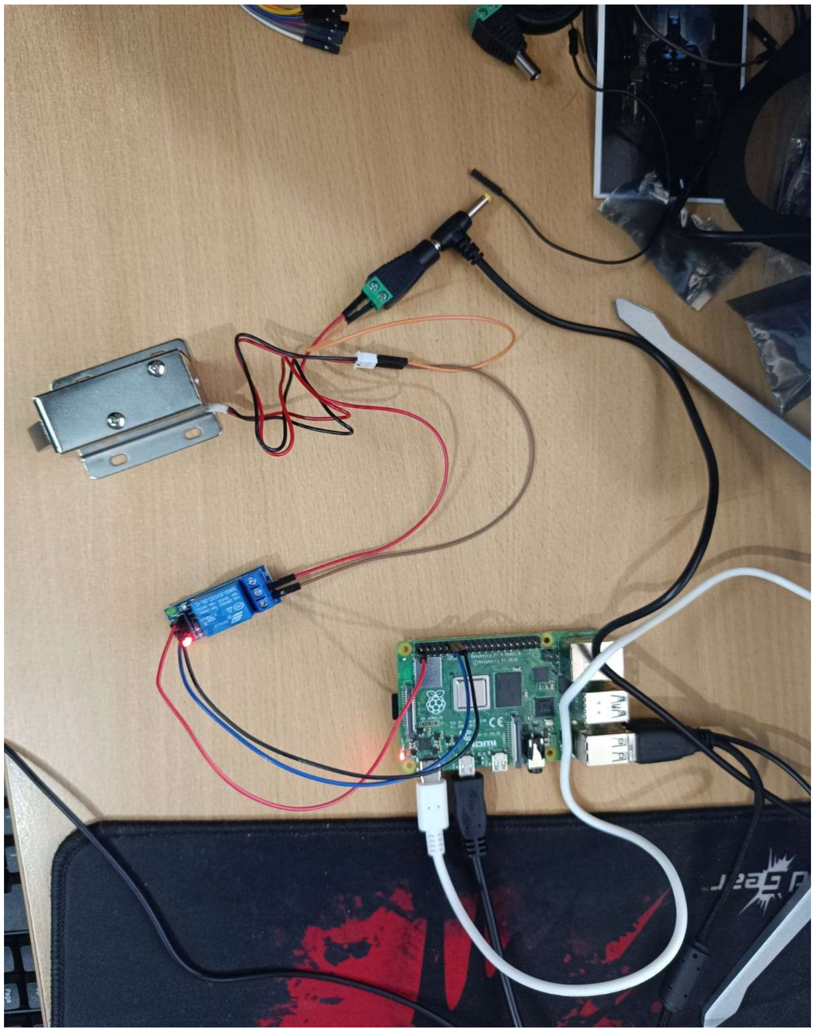
```
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
>>>
```

(HTML)

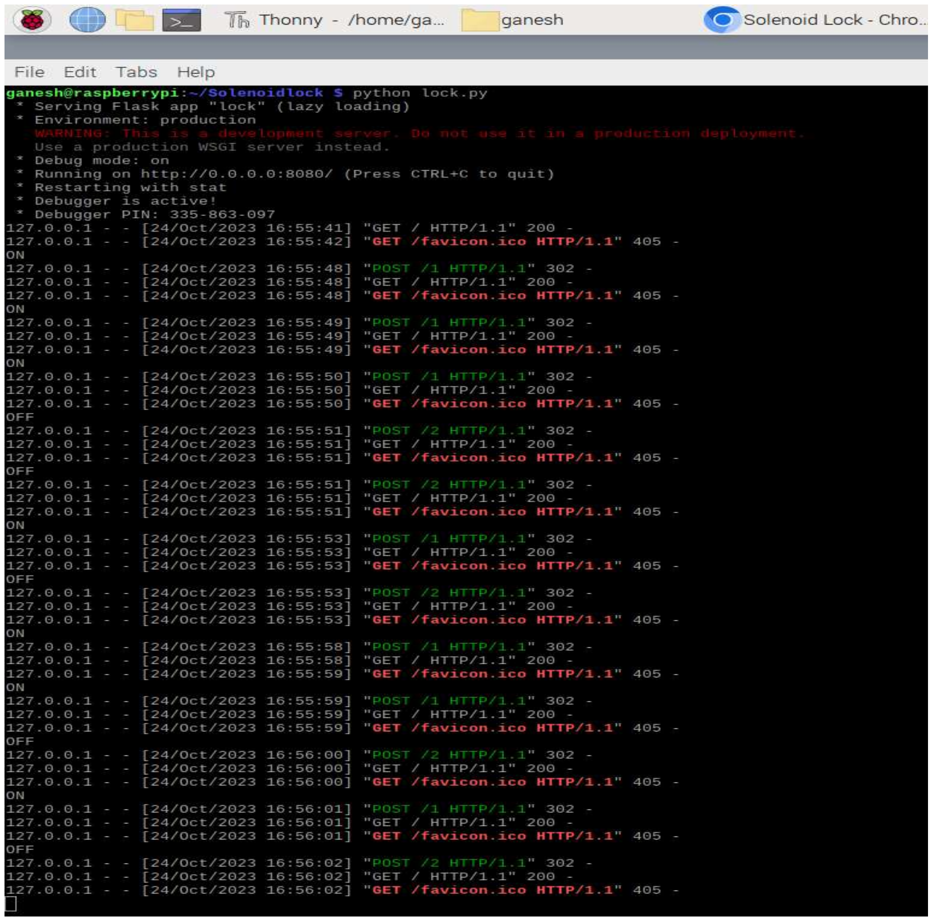
```html
<!DOCTYPE html>

<html>

        <head>

                <title>Solenoid Lock</title>

                <meta name="viewport" content="width=device-width, initial-
scale=1">

        </head>

        <body>

            <h2><center>IoT Based Solenoid Door Lock Using Raspberry Pi
4<center></h2>

        <form action="/1" method="POST">

                    <p align=center><button
style=width:90px;height:30px;background-color:red;color:white; <button
id="on" class="solenoid">ON</button> <P>

                      </br>

                    </form>
                    <form action="/2" method="POST">
                         <p align=center><button
style=width:90px;height:30px;background-color:black;color:white; <button
id="off" class="solenoid">OFF</button> <P>
                         </br>
                         </form>
    </body>
</html>
```

Output:

Connections:

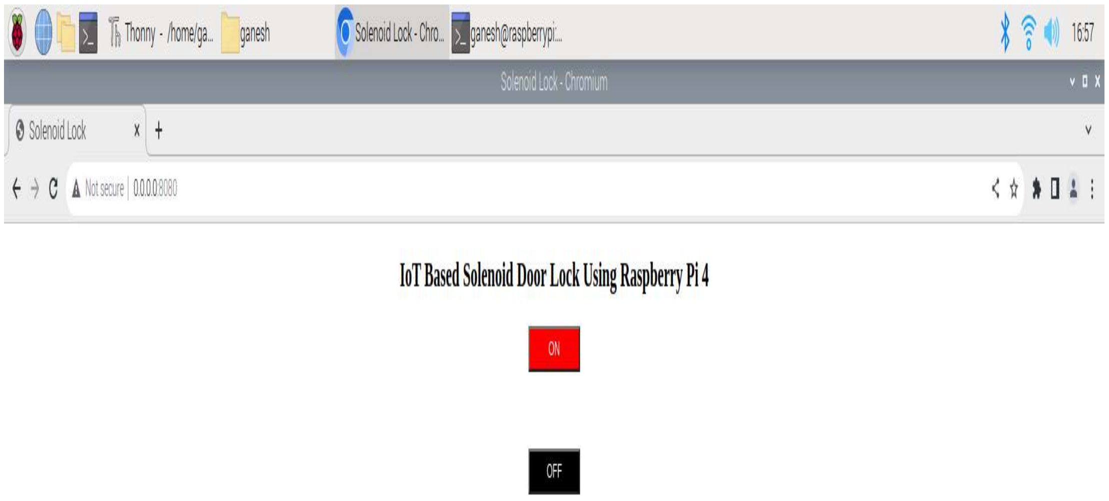


Terminal:



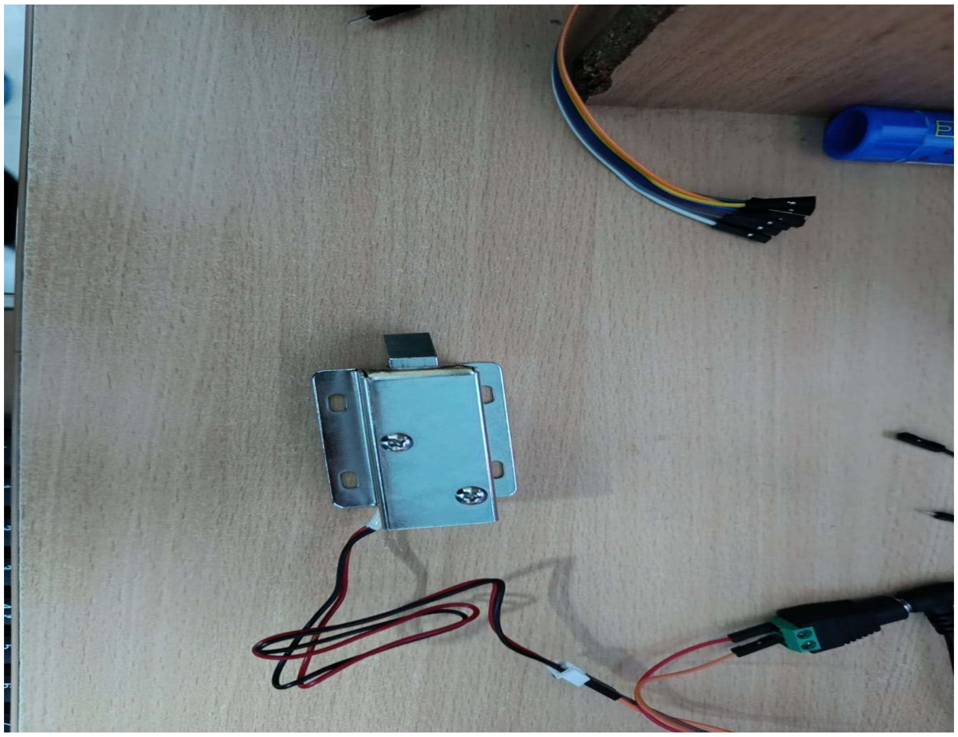
```
ganesh@raspberrypi:~/Solenoidlock $ python lock.py
 * Serving Flask app "lock" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://0.0.0.0:8080/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 335-863-097
127.0.0.1 - - [24/Oct/2023 16:55:41] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:42] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:55:48] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:48] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:48] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:55:49] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:49] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:49] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:55:50] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:50] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:50] "GET /favicon.ico HTTP/1.1" 405 -
OFF
127.0.0.1 - - [24/Oct/2023 16:55:51] "POST /2 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:51] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:51] "GET /favicon.ico HTTP/1.1" 405 -
OFF
127.0.0.1 - - [24/Oct/2023 16:55:51] "POST /2 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:51] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:51] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:55:53] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:53] "GET /favicon.ico HTTP/1.1" 405 -
OFF
127.0.0.1 - - [24/Oct/2023 16:55:53] "POST /2 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:53] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:53] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:55:58] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:58] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:59] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:55:59] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:55:59] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:55:59] "GET /favicon.ico HTTP/1.1" 405 -
OFF
127.0.0.1 - - [24/Oct/2023 16:56:00] "POST /2 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:56:00] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:56:00] "GET /favicon.ico HTTP/1.1" 405 -
ON
127.0.0.1 - - [24/Oct/2023 16:56:01] "POST /1 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:56:01] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:56:01] "GET /favicon.ico HTTP/1.1" 405 -
OFF
127.0.0.1 - - [24/Oct/2023 16:56:02] "POST /2 HTTP/1.1" 302 -
127.0.0.1 - - [24/Oct/2023 16:56:02] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [24/Oct/2023 16:56:02] "GET /favicon.ico HTTP/1.1" 405 -
```

WEBSITE:



IoT Based Solenoid Door Lock Using Raspberry Pi 4

DOOR LOCKED:



DOOR UNLOCKED: