Department project on illumination automation

Problem Statement:

Illumination automation refers to the process of automating the control of lighting systems in the institution .

Problem:

Inefficient and manual control of lighting systems leads to energy waste in the institution.

Objective:

To develop an illumination automation system that provides efficient control of lighting systems to reducing energy consumption.

Scope:

The illumination automation system will control lighting systems in institutional buildings. The system will have the ability to turn lights on and off automatically by detecting the motion of the people inside the room/lab.

Constraints:

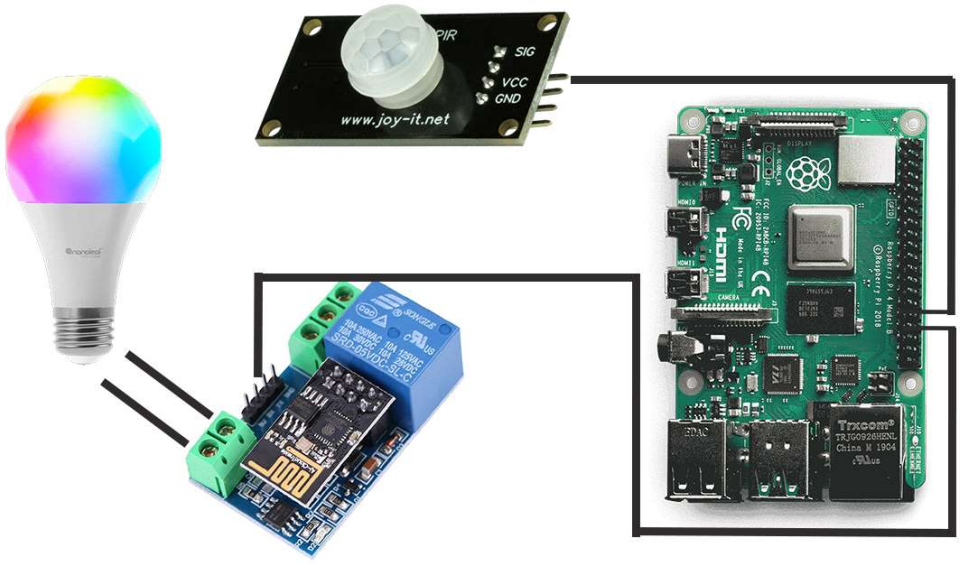
The illumination automation system must be user-friendly and cost-effective. The system must also adhere to industry standards and regulations for energy efficiency.

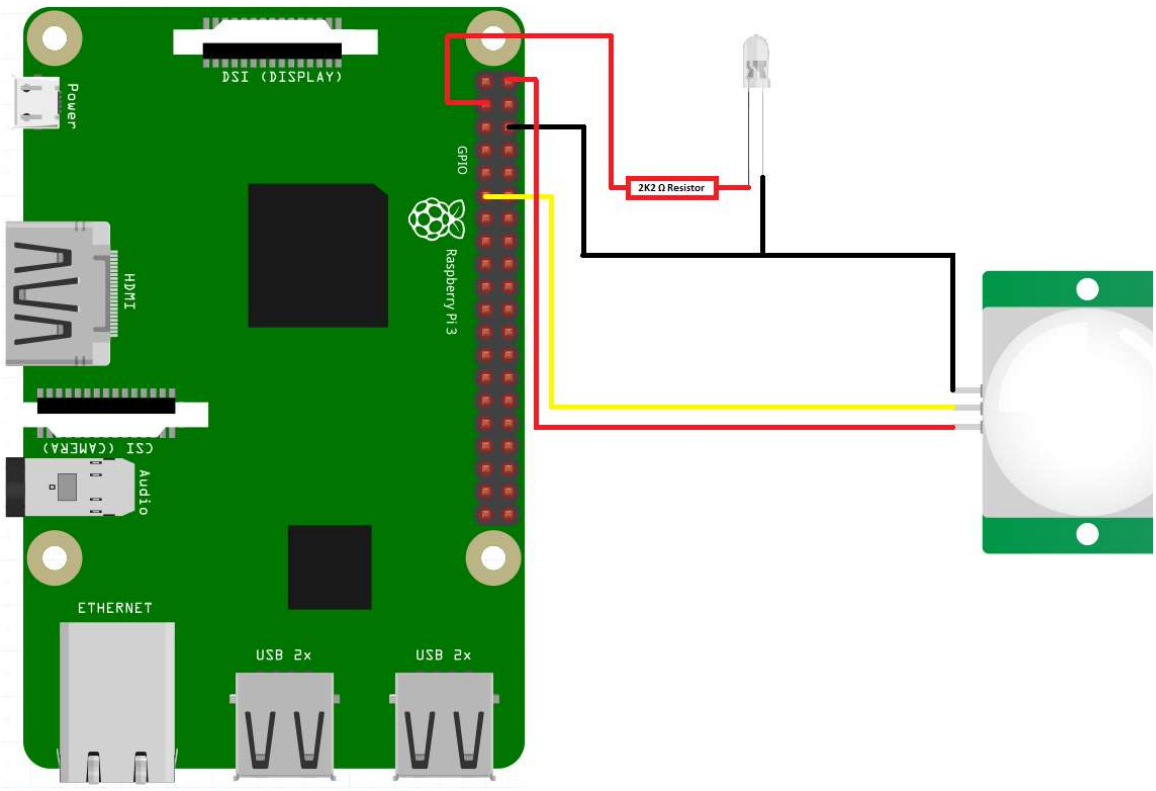
System Requirements:

- Raspberry Pi kit x1 (Lab included).
- PIR sensor x2.
- Wifi Relay Module 2 channels.
- Jumpers and Connecting wires.

Working:

The Sensors installed across the room will detect presence of Motion, the PIR's send signals to Raspberry Pi. The Microcontroller process the signals and sends commands to connected Relay switch. The Relay switch is directly connected to the Light power lines. The Relay switch also can be accessed with a Smartphone, thus this implementation improves the efficency of problem statement.


PIR
SIG
VCC
GND
www.joy-it.net

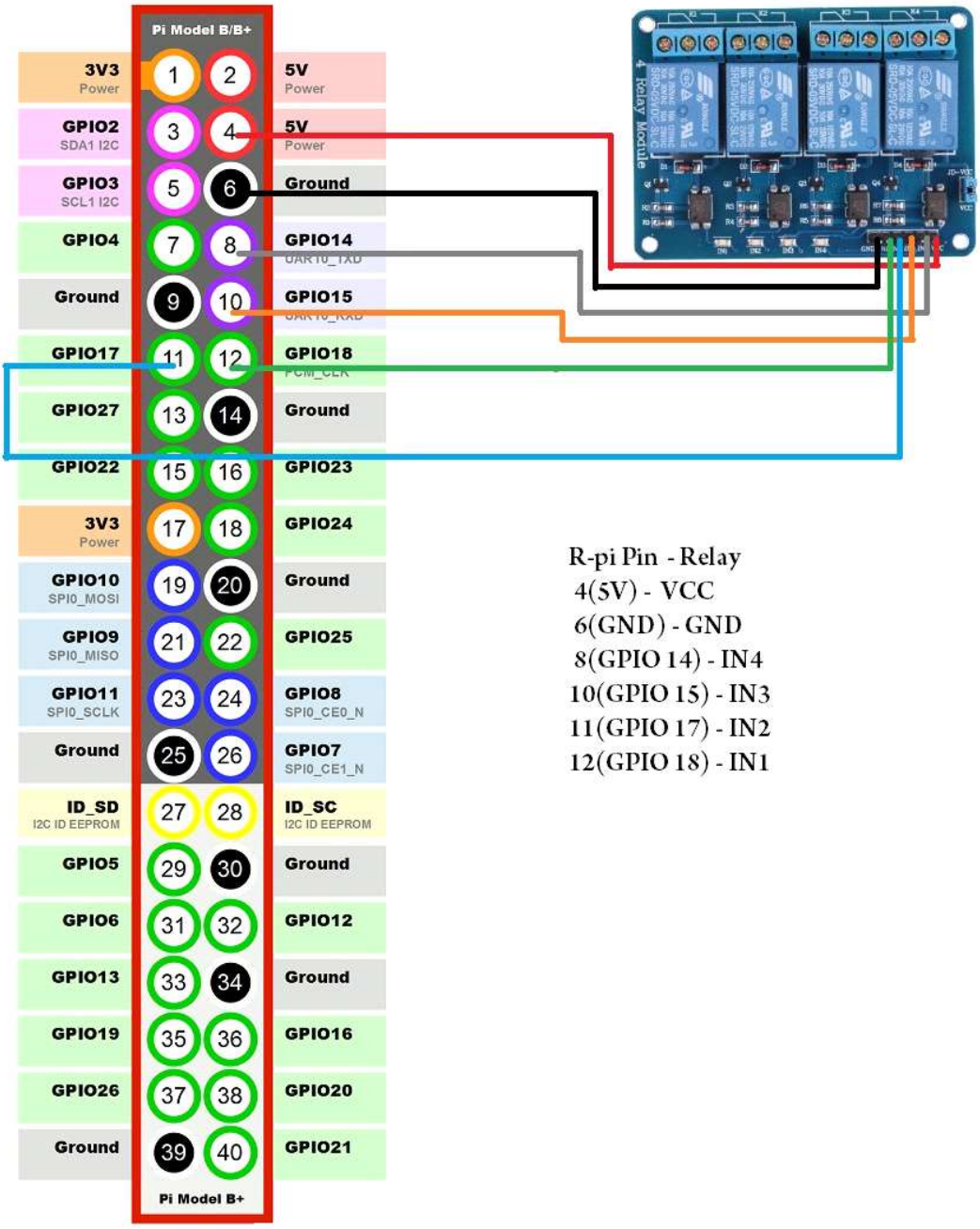



```
import RPi.GPIO as GPIO
import time
import json

# set up GPIO pins for PIR sensors
pir_pins = [23, 24]  # example PIR pins
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
for pin in pir_pins:
    GPIO.setup(pin, GPIO.IN)

# set up GPIO pins for relays
relay_pins = [18, 17, 15, 14]  # example relay pins
for pin in relay_pins:
```

```
    GPIO.setup(pin, GPIO.OUT)

# create empty list
arr = []
# for loop from 0 to 3
for x in range(0, 4):
    # putting all relays in on state if GPIO is not setup
    GPIO.output(relay_pins[x], 1)
    # putting the relay state in the empty list
    arr.append(not GPIO.input(relay_pins[x]))

# set up variables for detecting person and turning off lights
person_detected = False
time_since_person_left = 0
max_time_since_person_left = 60  # example time in seconds, adjust as needed


while True:
    # read PIR sensor inputs
    pir_input = []
    for pin in pir_pins:
        pir_input.append(GPIO.input(pin))

    # detect if a person is present and turn on the lights if necessary
    if any(pir_input):
        person_detected = True
        time_since_person_left = 0
        GPIO.output(relay_pins[0], 0)  # turn on first relay
    else:
```

```
    person_detected = False

# turn off the lights if the person has been gone for too long
if not person_detected:
    time_since_person_left += 0.1  # increment time since person left
    if time_since_person_left >= max_time_since_person_left:
        GPIO.output(relay_pins[0], 1)  # turn off first relay

# read current relay states and update arr list
arr = []
for x in range(0, 4):
    arr.append(not GPIO.input(relay_pins[x]))

# print the list of state's in JSON format
print(json.dumps({0: arr[0], 1: arr[1], 2: arr[2], 3: arr[3]}))

time.sleep(0.1)  # wait for a short time before repeating the loop
```

This code sets up two PIR sensor inputs and four relay outputs, and adds a person_detected variable to keep track of whether a person is in the room, and a time_since_person_left variable to keep track of how long it has been since the person left the room. If a person is detected, the lights are turned on and the time_since_person_left variable is reset to zero. If a person is not detected, the time_since_person_left variable is incremented by the loop time (0.1 seconds), and if it exceeds a certain threshold (max_time_since_person_left), the lights are turned off. You can adjust the max_time_since_person_left variable to set the amount of time that should elapse before the lights are turned off after the person leaves the room.



System:

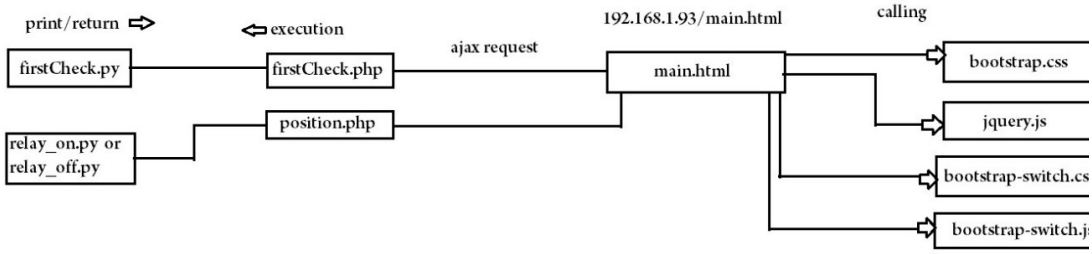
- The **index.html** file is the main file for the web interface. It includes some CSS and JavaScript files that are necessary for displaying the web page and creating interactive elements such as the Bootstrap switches.

- When a user interacts with the web page (e.g. toggling a switch), jQuery is used to make an AJAX POST request to the **update.php** file.
- The **update.php** file receives the POST request and executes a Python script (**firstcheck.py**) using the **exec()** function.
- The **firstcheck.py** script reads the current state of the GPIO pins and returns the values to the **update.php** file as a JSON object.
- The **update.php** file receives the JSON object and sends it back to the web page as the response to the original AJAX POST request.
- Finally, the web page updates the display to reflect the new state of the switches based on the values returned by the Python script.

(Simply its like a counter where the PHP and python interacts with each other to know the current state)

So the overall flow is: user interaction -> AJAX POST request -> PHP script execution -> Python script execution -> JSON response -> web page update.

This allows for a feedback loop between the user, the web interface, the PHP backend, and the Python script that controls the GPIO pins. By using AJAX and JSON, the web page can update in real-time without requiring a full page refresh, which provides a more seamless and responsive user experience.



Testing:

Here are some testing cases.

- Verify that the system can detect when a person enters the room by walking in front of the PIR sensor. This can be done by observing the state of the switches on the web page before and after the person enters the room.
- Verify that the system can detect when a person exits the room by walking out of the range of the PIR sensor. This can be done by observing the state of the switches on the web page before and after the person exits the room.
- Test that the lights turn on and off as expected when a person enters or exits the room.
- Test that the system can handle multiple people entering and exiting the room at the same time. For example, you could have two people enter the room at the same time and see if the lights turn on for both of them.
- Test that the system can handle different lighting scenarios, such as turning on only certain lights when a person enters the room during the day versus at night.
- Verify that the system can handle different numbers and types of sensors, such as adding additional PIR sensors or integrating other types of sensors like temperature or humidity sensors.
- Test that the web interface is responsive and updates in real-time when a user interacts with the switches.
- Verify that the system is robust and can recover from errors, such as a lost internet connection or a power outage.

4 Array relay Web App

4 Array relay

Light 1 ON

Light 2 ON

Light 3 ON

Light 4 ON

Relay 1 is

Relay 2 is

Relay 3 is

Relay 4 is