NAAN MUDHALVAN PHASE-3 ASSESSMENT

Course Name : Internet Of Things

Project Title : Smart Public Restroom

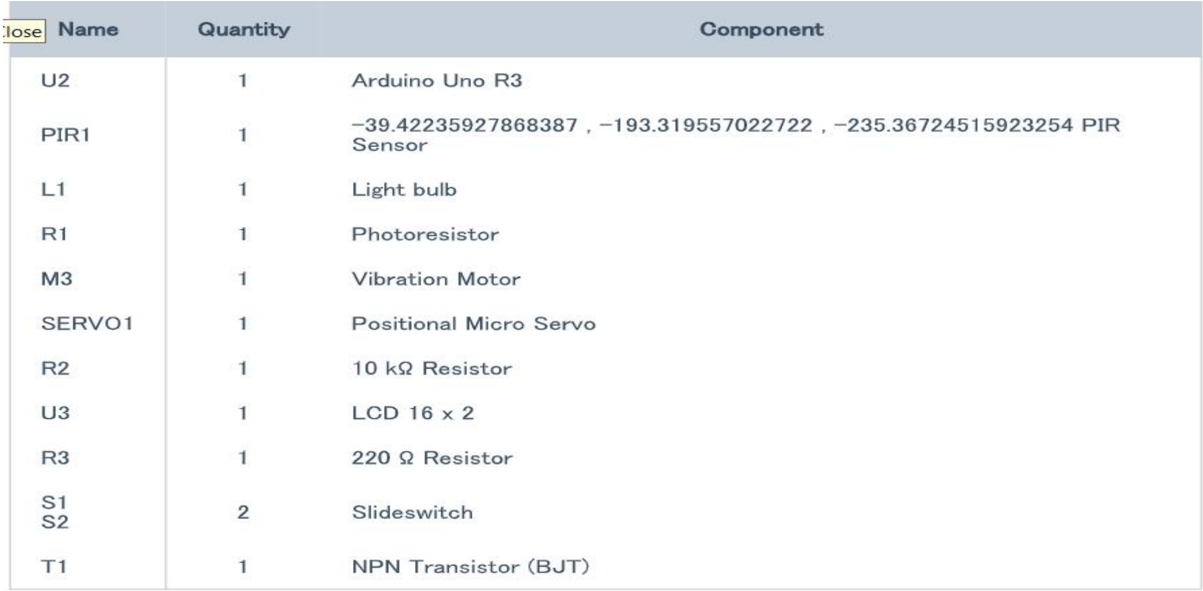
Team Name : Team Spartans

Team Members :

Name	AU Register No.	Naan Mudhalvan ID
Vishnu Kumar S R	721221106121	au721221106121
Suryamoorthi K	721221106108	au721221106108
Nithishkumar S	721221106065	au721221106065
Hema Varshini S	721221106035	au721221106035

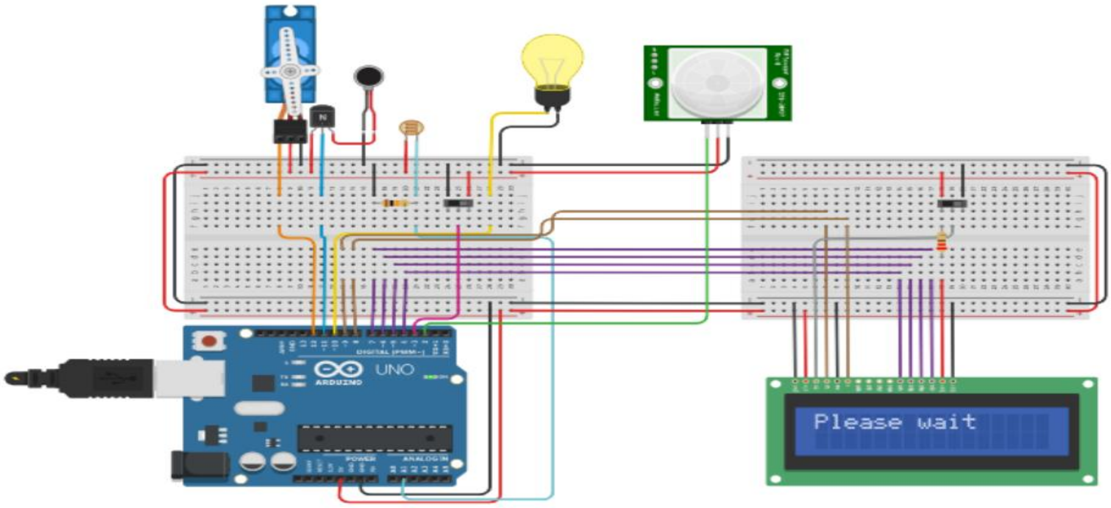
Simulator used : Wokwi Simulator

Components used :



Name	Quantity	Component
U2	1	Arduino Uno R3
PIR1	1	−39.42235927868387 , −193.319557022722 , −235.36724515923254 PIR Sensor
L1	1	Light bulb
R1	1	Photoresistor
M3	1	Vibration Motor
SERVO1	1	Positional Micro Servo
R2	1	10 kΩ Resistor
U3	1	LCD 16 x 2
R3	1	220 Ω Resistor
S1 S2	2	Slideswitch
T1	1	NPN Transistor (BJT)

Circuit Diagram:



Working :

The provided Python code is designed to simulate a simple system using a Raspberry Pi that controls a servo motor and an LED based on the status of a push-button switch, simulating a restroom's occupancy status. Here's how the code works:

Initialization (Setup):

- It imports the necessary libraries: RPi.GPIO and time.
- It defines the GPIO (General-Purpose Input/Output) pins used for the button, LED, and servo motor.
- It sets the GPIO mode to use the Broadcom SOC channel numbering.
- It configures the buttonPin as an input and ledPin as an output.
- It initializes the servo motor by configuring the servoPin and setting up a PWM (Pulse Width Modulation) object with a frequency of 50 Hz. The PWM signal will be used to control the servo motor's position.

- It starts the servo motor at its initial position (0 degrees).

Loop:

- The code enters a continuous loop using a while loop with True as the condition.

Button State Detection:

- Inside the loop, it reads the state of the buttonPin using GPIO.input(buttonPin) and stores it in the buttonState variable.

Restroom Status Control:

- If the button is pressed (buttonState is HIGH), it considers the restroom as occupied.
- It turns on the LED (sets ledPin to HIGH) to indicate that the restroom is occupied.
- It uses PWM to set the servo motor to a specific duty cycle (7.5%) to open the door (rotate the servo to 90 degrees).

Restroom Vacant:

- If the button is not pressed (buttonState is LOW), it considers the restroom as vacant.
- It turns off the LED (sets ledPin to LOW) to indicate that the restroom is vacant.
- It uses PWM to set the servo motor to a different duty cycle (2.5%) to close the door (return the servo to 0 degrees).

Delay:

- The code includes a small delay of 0.1 seconds using time.sleep(0.1) to prevent rapid toggling of the servo and LED due to switch bounce.

Cleanup on KeyboardInterrupt:

- The code is designed to handle a KeyboardInterrupt (Ctrl+C) gracefully. When the user interrupts the program, it stops the servo motor and cleans up the GPIO pins using servo.stop() and GPIO.cleanup(), respectively.
- In summary, the code continuously monitors the state of a button switch. When the button is pressed, it turns on an LED and rotates a servo to open a door, simulating an occupied restroom. When the button is released, it turns off the LED and closes the door by returning the servo to its initial position, simulating a vacant restroom. The code keeps running in a loop until the user interrupts it, and it ensures proper cleanup of resources upon exit.

Program:

```
//Given code
#include <Servo.h>
const int buttonPin = 7;
const int ledPin = 2;
const int servoPin = 9;  // Digital pin for the servo
int buttonState = 0;
Servo doorServo;
void setup() {
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);
  doorServo.attach(servoPin);  // Attaching the servo to the pin
}
void loop() {
  buttonState = digitalRead(buttonPin);
  if (buttonState == HIGH) {
    // Restroom is occupied
    digitalWrite(ledPin, HIGH);
    // Open the door (rotate the servo)
    doorServo.write(90);  // Angle to open the door
  }
else {
    // Restroom is vacant
    digitalWrite(ledPin, LOW);
    // Close the door (return the servo to its initial position)
    doorServo.write(0);  // Angle to close the door
  }
}
//Python code
import RPi.GPIO as GPIO
import time
# Define the GPIO pins
buttonPin = 7
ledPin = 2
servoPin = 9
# Set the GPIO mode and setup the pins
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(ledPin, GPIO.OUT)
GPIO.setup(buttonPin, GPIO.IN)
# Initialize the servo
GPIO.setup(servoPin, GPIO.OUT)
servo = GPIO.PWM(servoPin, 50)  # 50 Hz PWM frequency
servo.start(0)  # Start with the servo at its initial position

try:
    while True:
        buttonState = GPIO.input(buttonPin)
        if buttonState == GPIO.HIGH:
            # Restroom is occupied
            GPIO.output(ledPin, GPIO.HIGH)
            # Open the door (rotate the servo)
            servo.ChangeDutyCycle(7.5)  # Duty cycle for 90 degrees

        else:
            # Restroom is vacant
            GPIO.output(ledPin, GPIO.LOW)
            # Close the door (return the servo to its initial position)
            servo.ChangeDutyCycle(2.5)  # Duty cycle for 0 degrees

        time.sleep(0.1)
```

```
except KeyboardInterrupt:
    servo.stop()
    GPIO.cleanup()
```