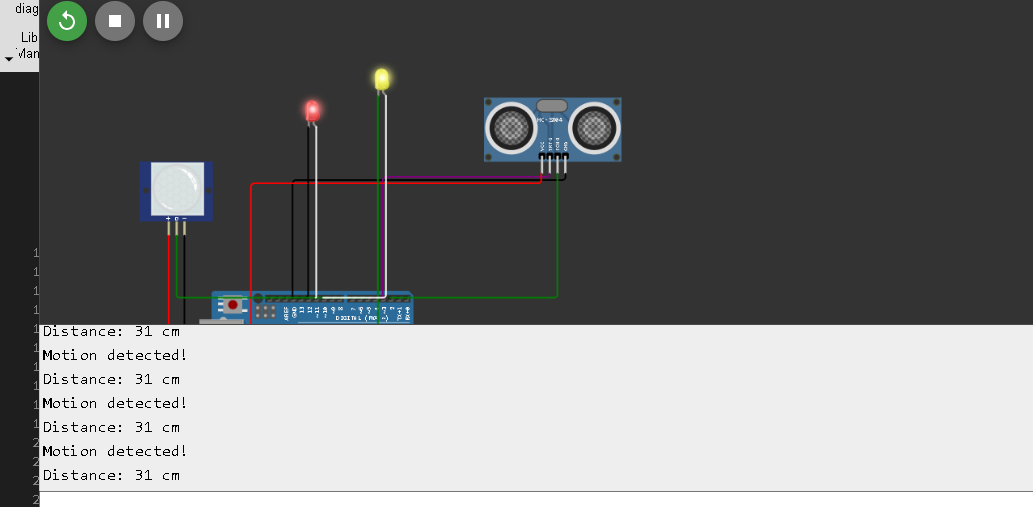
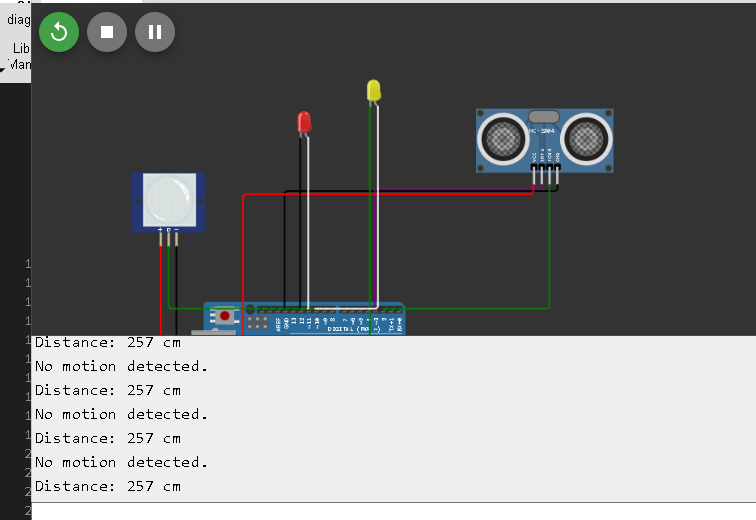
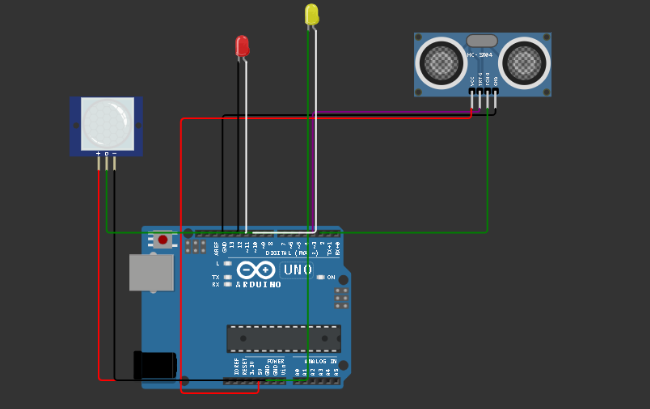
**SMART PUBLIC RESTROOM**

Here the simulation and the output simple basic circuit with the pir and ultra sonic sensors in the simulation.

This picture denotes the simulation that occurs in the pir and ultra sonic sensor the pir sensor detects the motion and the ultra sonic sensor detects the distance .the pir sensor detects the motion the light gets turn on. if the motion does not detected it gets turned off. The minimum distance that set is detected and the distance is below or above the value the light gets turned on . if it doesn’t attain the required distance it gets turned off. The below picture show the no motion detected and the distance is set to above level.





This is the basic circuit of the sensor and Arduino board pir senor ground positive and negative terminals are connected out center and vcc are connected to board, then the ultrasonic sensor is connected vcc, trigger, echo, ground terminals are connected to the board. The Arduino board got power supply and the led are connected to ultra-sonic and pir to detect the motion and distance.

Coding for this simulation in python

import RPi.GPIO as GPIO

import time

# Pin Definitions

pir\_pin = 5 # PIR sensor pin

trig\_pin = 3 # Ultrasonic sensor Trig pin

echo\_pin = 2 # Ultrasonic sensor Echo pin

pir\_led = 11 # LED for PIR sensor

ultrasonic\_led = 10 # LED for Ultrasonic sensor

# Setup GPIO

GPIO.setmode(GPIO.BOARD)

GPIO.setup(pir\_pin, GPIO.IN)

GPIO.setup(trig\_pin, GPIO.OUT)

GPIO.setup(echo\_pin, GPIO.IN)

GPIO.setup(pir\_led, GPIO.OUT)

GPIO.setup(ultrasonic\_led, GPIO.OUT)

def read\_pir():

pir\_value = GPIO.input(pir\_pin)

GPIO.output(pir\_led, pir\_value) # Turn PIR LED on if motion is detected

if pir\_value == GPIO.HIGH:

print("Motion detected!")

else:

print("No motion detected.")

def read\_ultrasonic():

GPIO.output(trig\_pin, GPIO.LOW)

time.sleep(0.2)

GPIO.output(trig\_pin, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trig\_pin, GPIO.LOW)

pulse\_start = time.time()

while GPIO.input(echo\_pin) == GPIO.LOW:

pulse\_start = time.time()

pulse\_end = time.time()

while GPIO.input(echo\_pin) == GPIO.HIGH:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150 # Speed of sound is 34300 cm/s

distance = round(distance, 2)

GPIO.output(ultrasonic\_led, distance < 50) # Turn Ultrasonic LED on if distance < 50cm

print(f"Distance: {distance} cm")

# Main loop

try:

while True:

read\_pir()

read\_ultrasonic()

time.sleep(1) # Delay for readability, adjust as needed

except KeyboardInterrupt:

GPIO.cleanup()

According to pin numbers in the program the sensors are connected and executed and simulated.

# *Coding for cleanliness and occupancy sensor for real time data analysis*

import requests

import time

# Function to read sensor data (replace with actual sensor readings)

def read\_sensors():

occupancy = 1 # Example occupancy value (1 means occupied, 0 means empty)

cleanliness = 0.8 # Example cleanliness value (0.8 means 80% clean)

return occupancy, cleanliness

# Function to send data to the platform

def send\_data(occupancy, cleanliness):

url = "http://your\_platform\_endpoint.com/api/update\_data" # Replace with actual API endpoint

payload = {

"occupancy": occupancy,

"cleanliness": cleanliness

}

try:

response = requests.post(url, json=payload)

response.raise\_for\_status()

print(f"Data sent successfully. Response: {response.status\_code}")

except requests.exceptions.RequestException as e:

print(f"Failed to send data: {e}")

# Main loop to continuously send data

while True:

occupancy, cleanliness = read\_sensors() # Read sensor data

send\_data(occupancy, cleanliness) # Send data to the platform

time.sleep(60) # Wait for 60 seconds before sending the next update

for occupancy and cleanliness sensor python code.