Design & Analysis of IoT Software Systems - SOFE4610U

Assignment 3

CRN: 44432

Group Number 8

Group Member 1

Name: Alden O'Cain

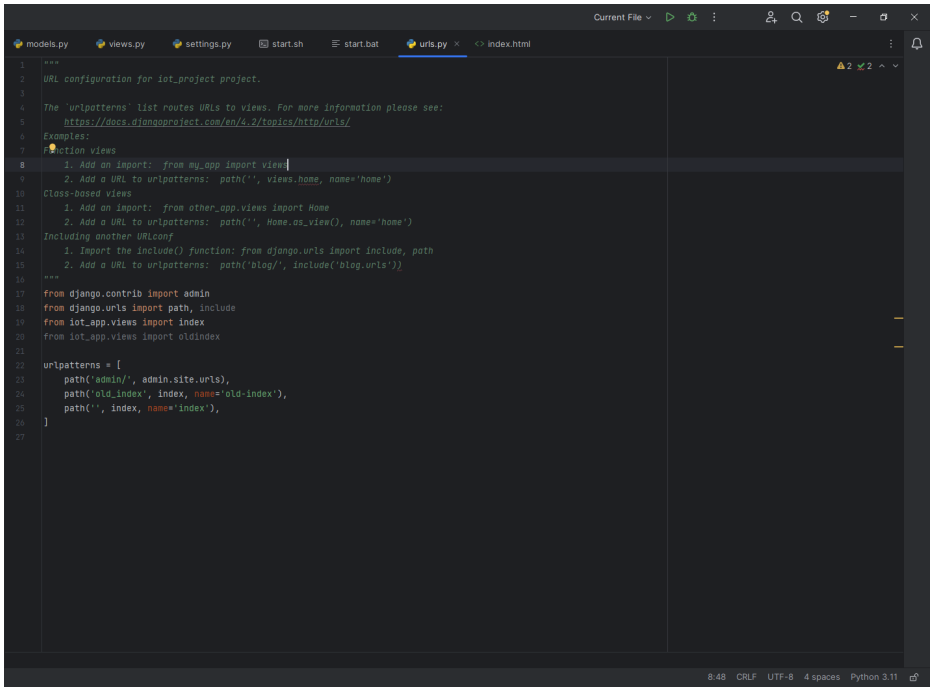
Student ID: 100558599

Group Member 2

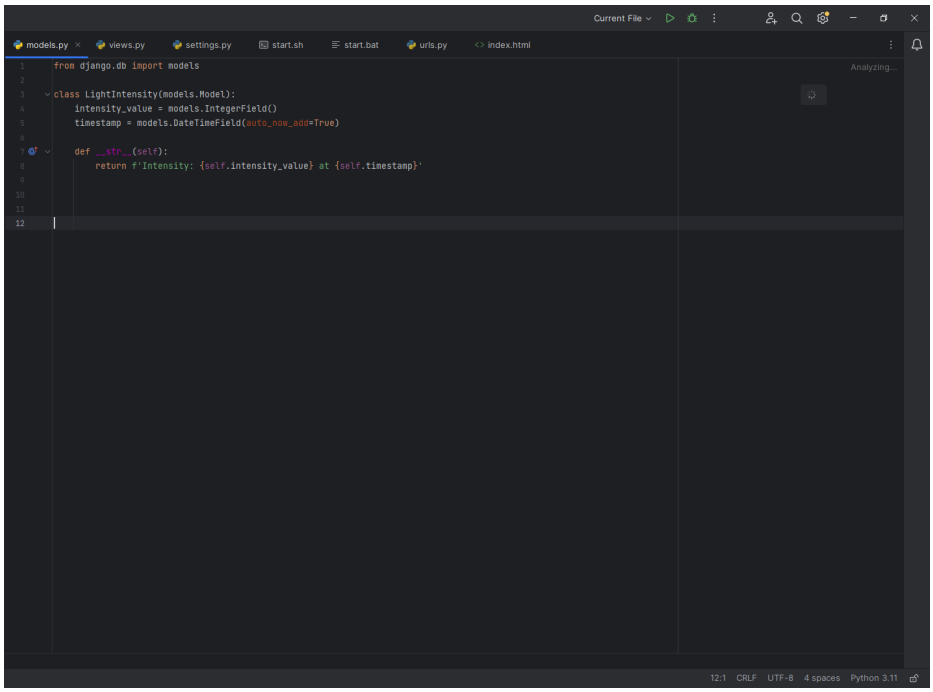
Name: Liam Rea

Student ID: 100743012

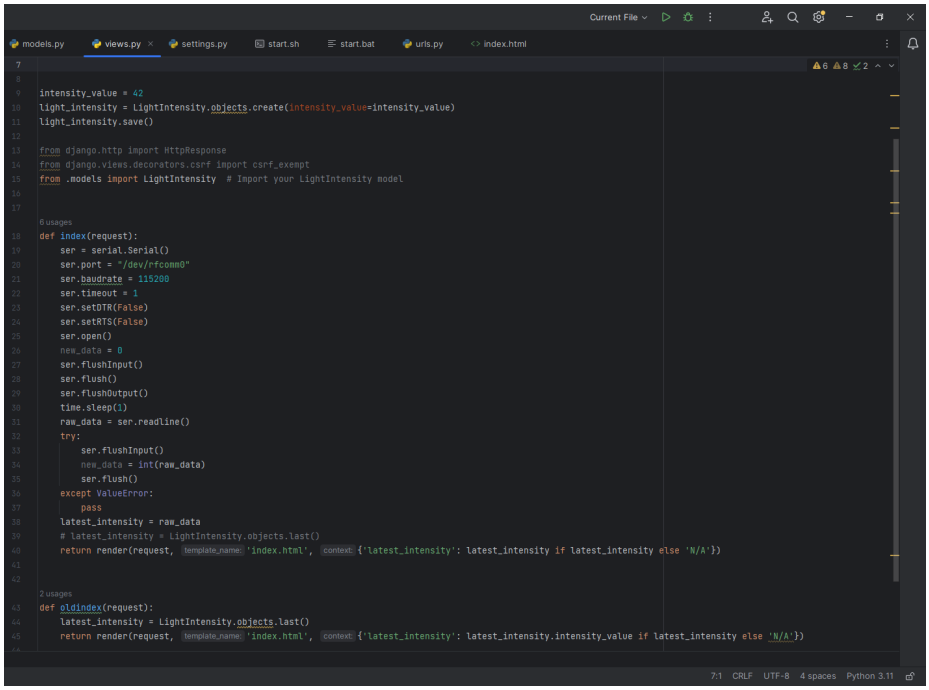
Project Repository



Defining URL routes in the Django app

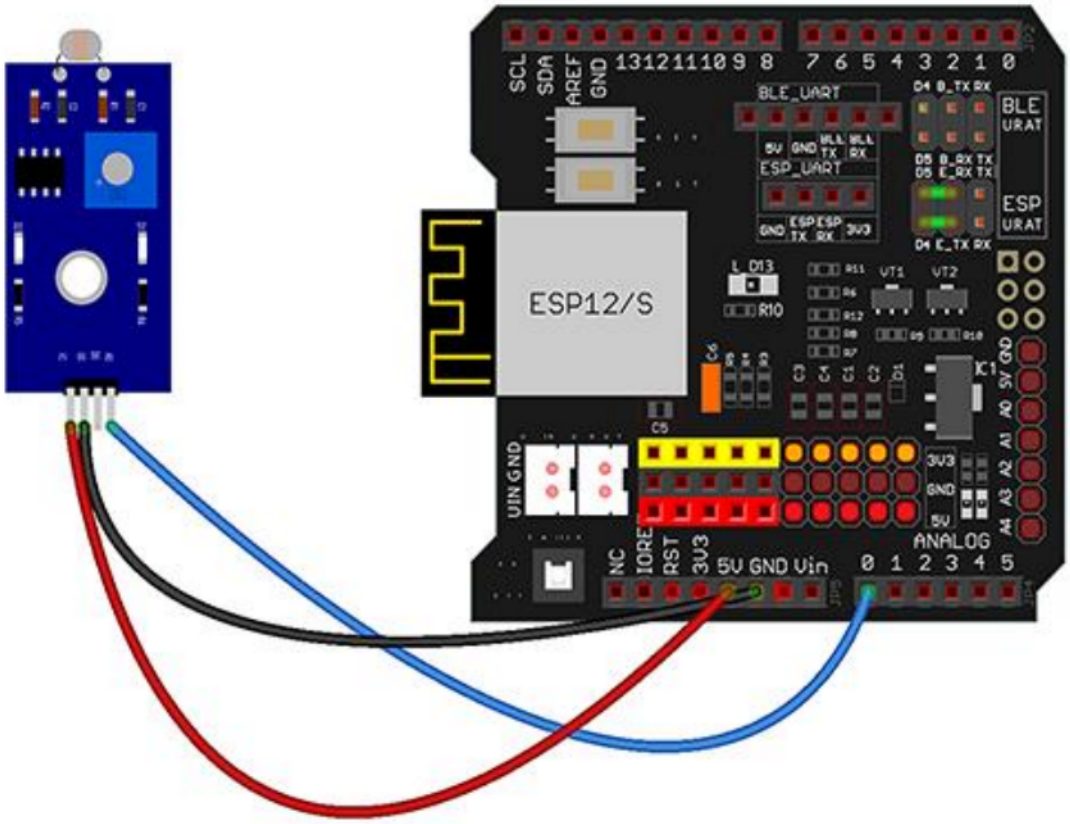


Creating a model for light intensity data

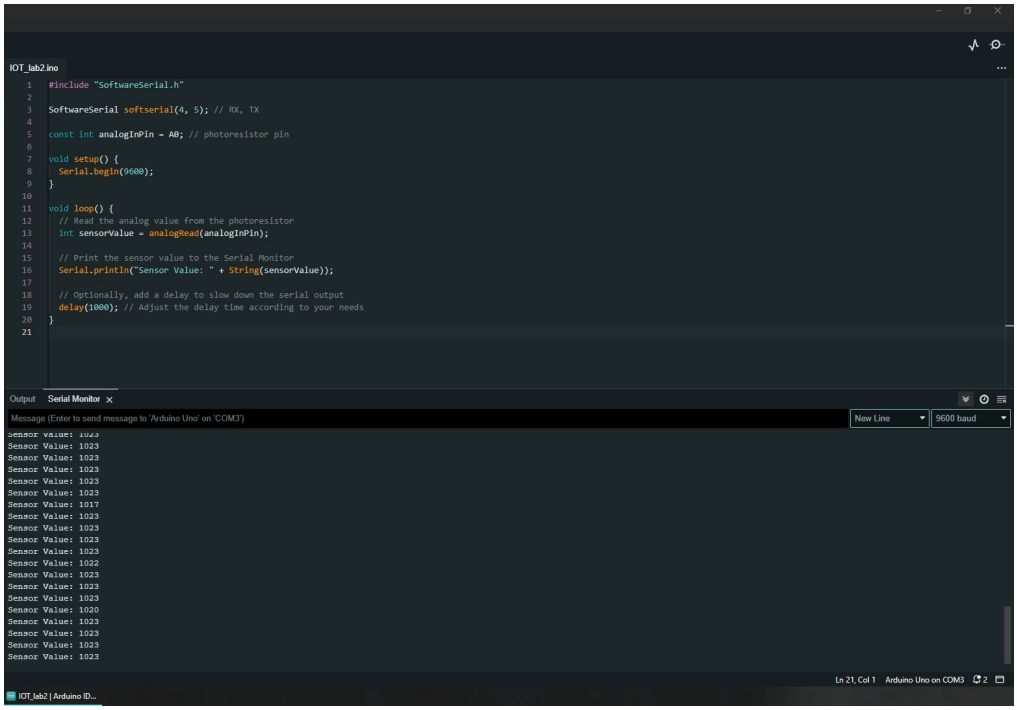


Creating a view for the index page (oldIndex displays a static value)

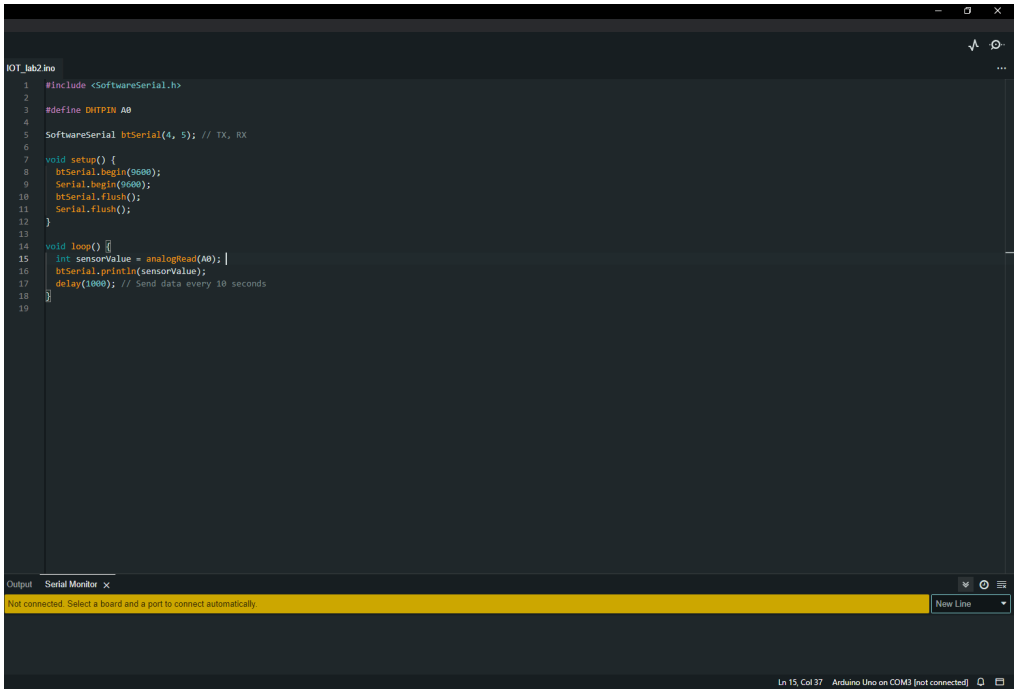
Old index displays a static value to test the HTML template



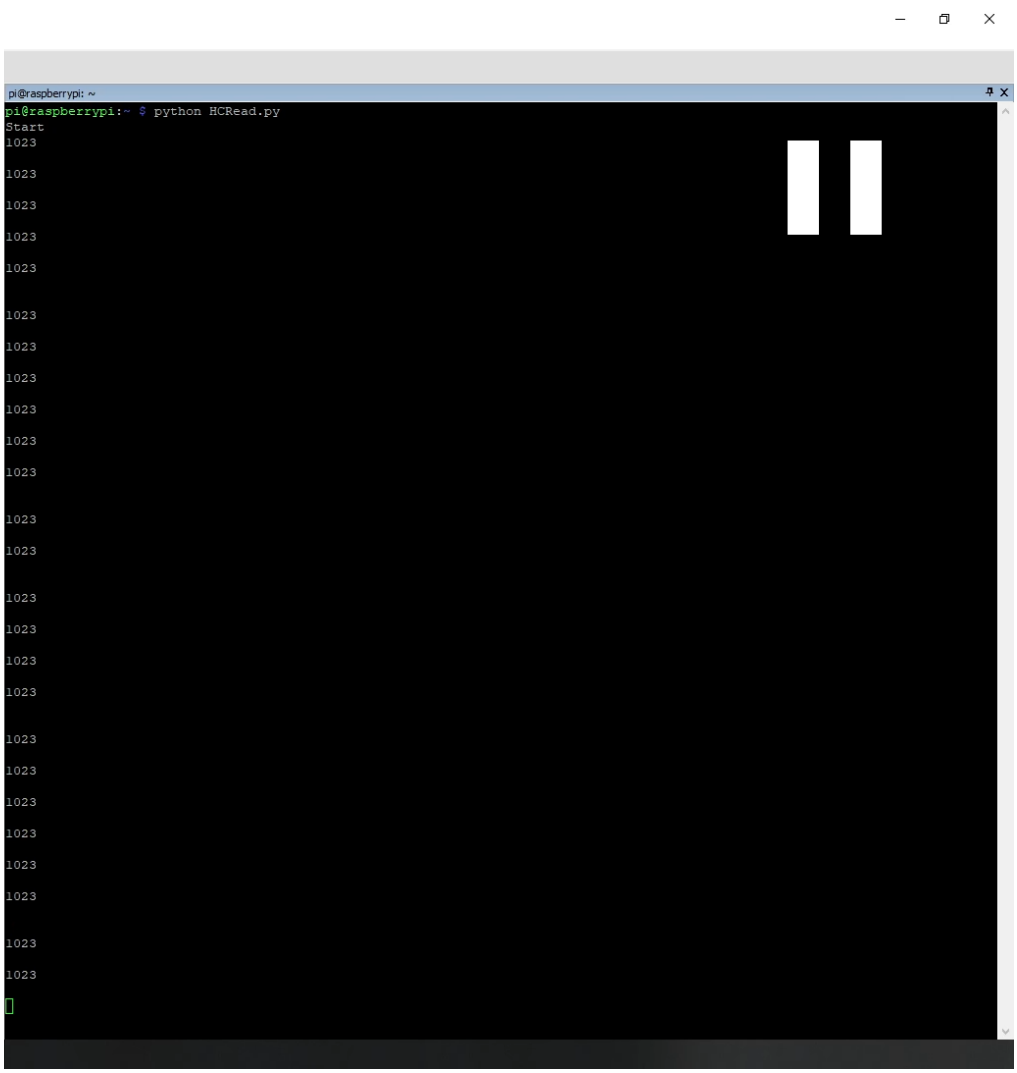
Wiring the photoresistor sensor to the Arduino WiFi shield



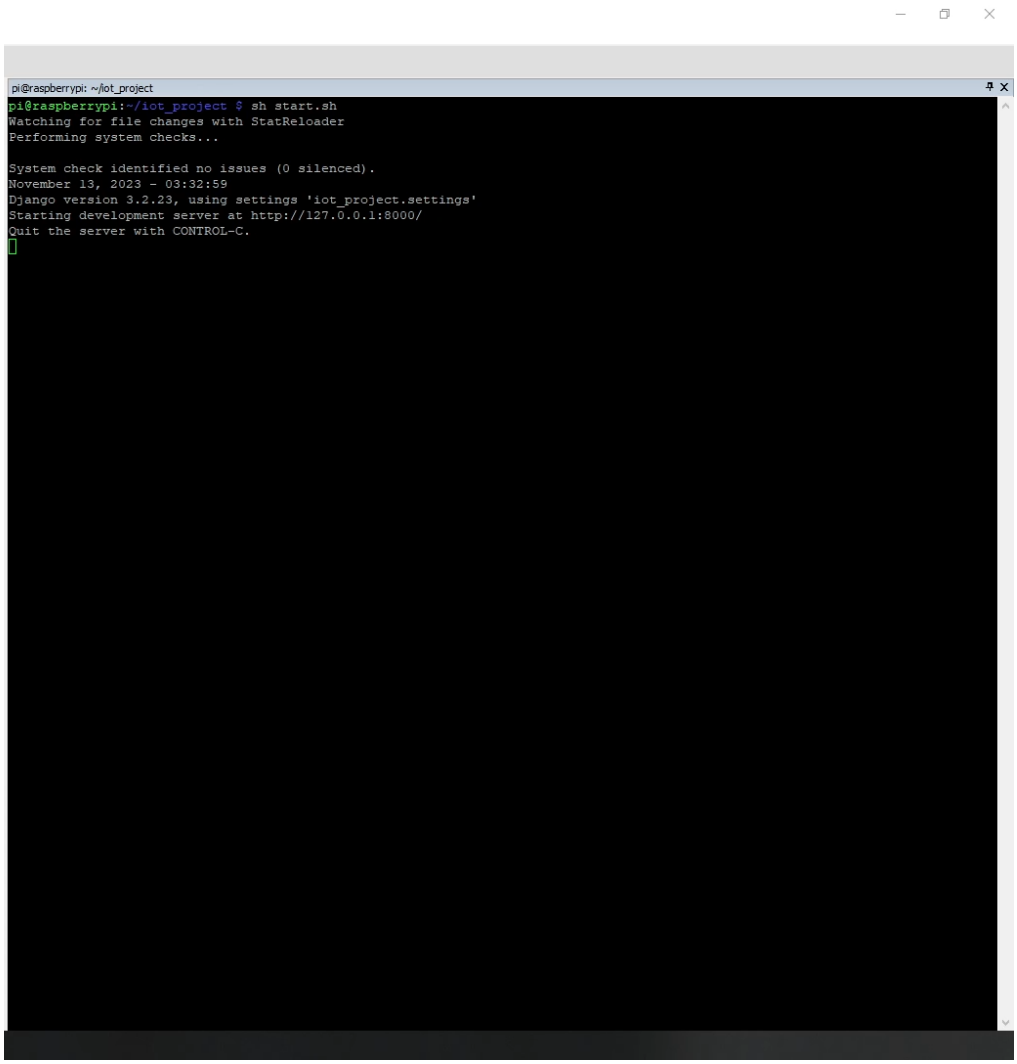
Testing reading the sensor and printing to the Serial Monitor



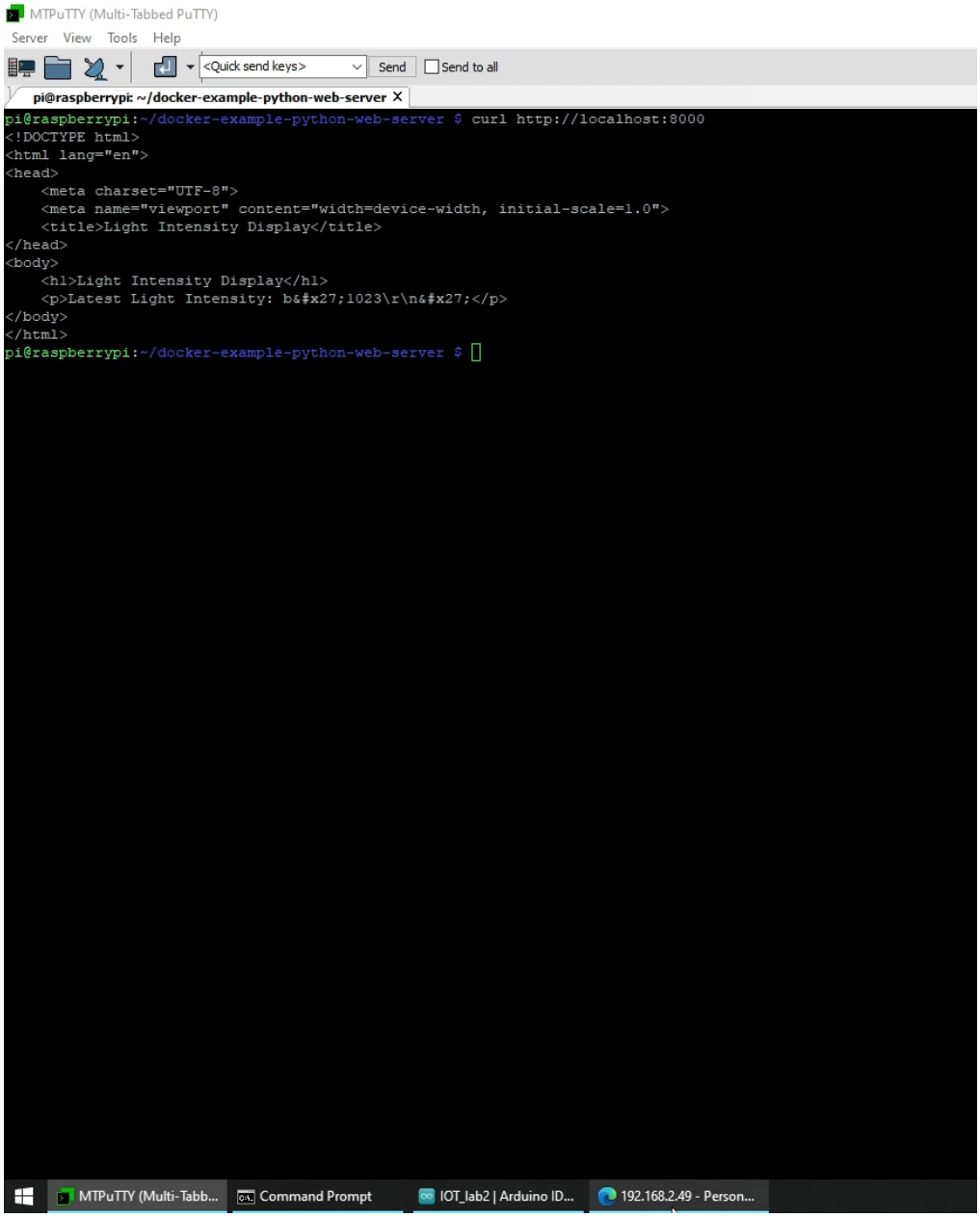
Updating the Arduino sketch to send the sensor value through bluetooth



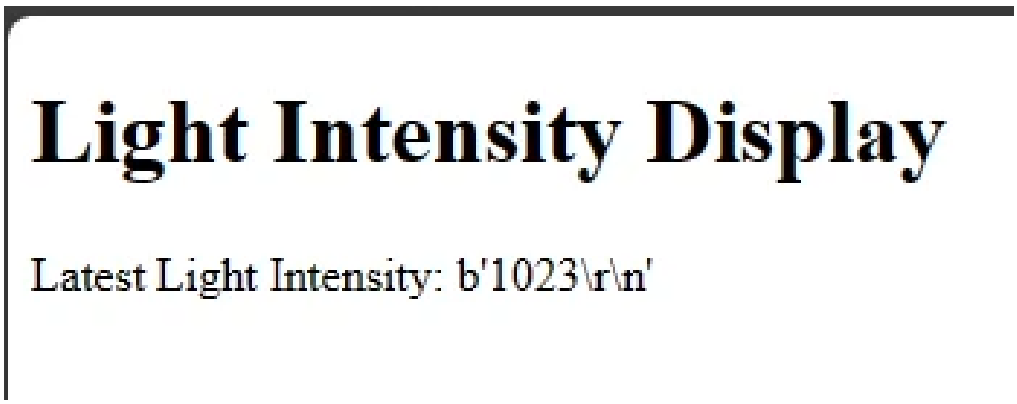
Testing capturing the photoresistor sensor values using a simple python script



Running the updated Django server on the Raspberry Pi

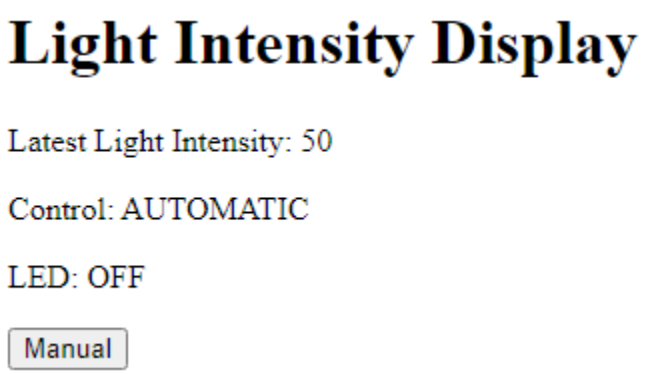


Testing the output of the Django web server locally on the Raspberry pi using curl after incorporating reading the bluetooth data inside the Django view

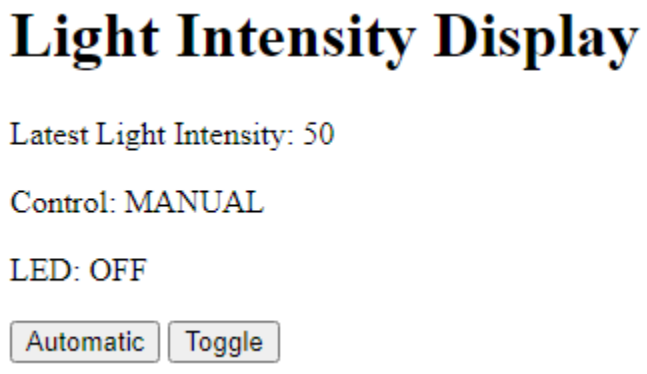


Django server displaying the correct values to another computer on the network after changing the Django settings to bind to the private IP address of the Raspberry Pi

Implementing Manual and Automatic Control of a Light



Simulating **(due to faulty photoresistor sensor)** a low value (brightness) resulting in AUTO light OFF



Simulating **(due to faulty photoresistor sensor)** a low value (brightness) resulting in MANUAL mode with the ability to toggle the light

Light Intensity Display

Latest Light Intensity: 11000

Control: AUTOMATIC

LED: ON

Manual

Simulating **(due to faulty photoresistor sensor)** a high value (darkness) resulting in AUTO light ON

Light Intensity Display

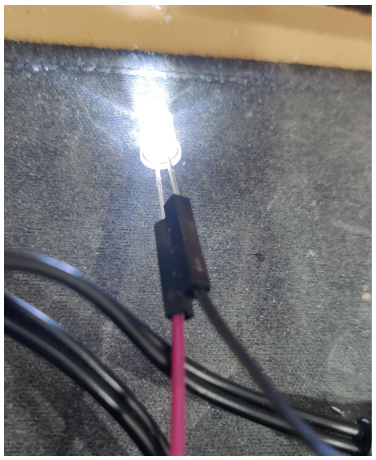
Latest Light Intensity: 11000

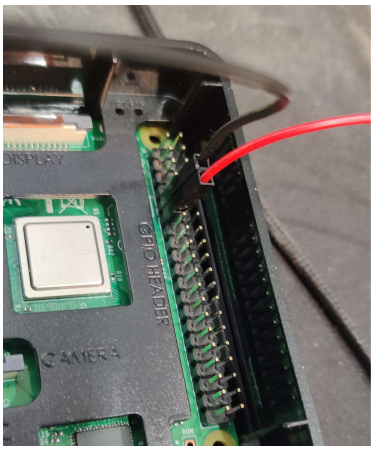
Control: MANUAL

LED: OFF

Automatic Toggle

Simulating **(due to faulty photoresistor sensor)** a high value (darkness) resulting in MANUAL mode with the ability to toggle the light





LED connected to Raspberry Pi with a 200 ohm resistor

```
<body>
    <h1>Light Intensity Display</h1>
    <p>Latest Light Intensity: {{ latest_intensity }}</p>

    <form method="POST" action="{% url 'index' %}">
        {% csrf_token %}
        <p>Control: {{ mode }}</p>
        <p>LED: {{ led_value }}</p>

        {% if mode != "MANUAL" %}
            <button type="submit" name="mode-manual" value="manual">Manual</button>
        {% endif %}
        {% if mode != "AUTOMATIC" %}
            <button type="submit" name="mode-automatic" value="automatic">Automatic</button>
        {% endif %}
        {% if mode != "AUTOMATIC" %}
            <button type="submit" name="toggle" value="toggle">Toggle</button>
        {% endif %}
    </form>
</body>
```

Updated HTML template with conditional visibility of buttons and display of state values

```
if request.method == 'POST':
    global led_value
    global mode

    if 'mode-manual' in request.POST:
        mode = "MANUAL"
    if 'mode-automatic' in request.POST:
        mode = "AUTOMATIC"

    if 'toggle' in request.POST:
        global led_state
        led_state = not led_state
        if mode == "MANUAL":
            if led_state:
                print("[LED-MANUAL] Attempting to turn LED on!")
                led_value = "ON"
                GPIO.output(led_pin, GPIO.HIGH)
            else:
                print("[LED-MANUAL] Attempting to turn LED off!")
                led_value = "OFF"
                GPIO.output(led_pin, GPIO.LOW)
```

Handling POST requests to change AUTO/MANUAL as well as toggle LED requests

```python
# high values are low light, low values are high light
LI_LOWER_LIMIT = 100    # when BELOW this value, turn the light off (when automatic is enabled)
LI_UPPER_LIMIT = 10000  # when ABOVE this value, turn the light on (when automatic is enabled)

# USED TO CHECK AUTOMATIC BEHAVIOUR TO THRESHOLD VALUES
SIMULATE = False
SIMULATE_MDOE = "DARK"
# SIMULATE_MDOE = "LIGHT"
if SIMULATE_MDOE == "DARK":
    SIM_VALUE = 11000
elif SIMULATE_MDOE == "LIGHT":
    SIM_VALUE = 50
```

Declaration of example threshold values for low/high light and simulation code for faulty sensor

```python
if mode == "AUTOMATIC":
    if latest_intensity > LI_UPPER_LIMIT:  # SENSOR IS READING DARK - TURN ON LED
        led_state = not led_state
        print("[LED-AUTO] Attempting to turn LED on!")
        led_value = "ON"
        GPIO.output(led_pin, GPIO.HIGH)
    elif latest_intensity < LI_LOWER_LIMIT:  # SENSOR IS READING LIGHT - TURN OFF LED
        led_state = not led_state
        print("[LED-AUTO] Attempting to turn LED off!")
        led_value = "OFF"
        GPIO.output(led_pin, GPIO.LOW)
```

Code for AUTO handling sensor readings outside of thresholds and actuating LED accordingly