Videolink Gitlablink

# Smart Light

## NEW BUSINESS OPPORTUNITY

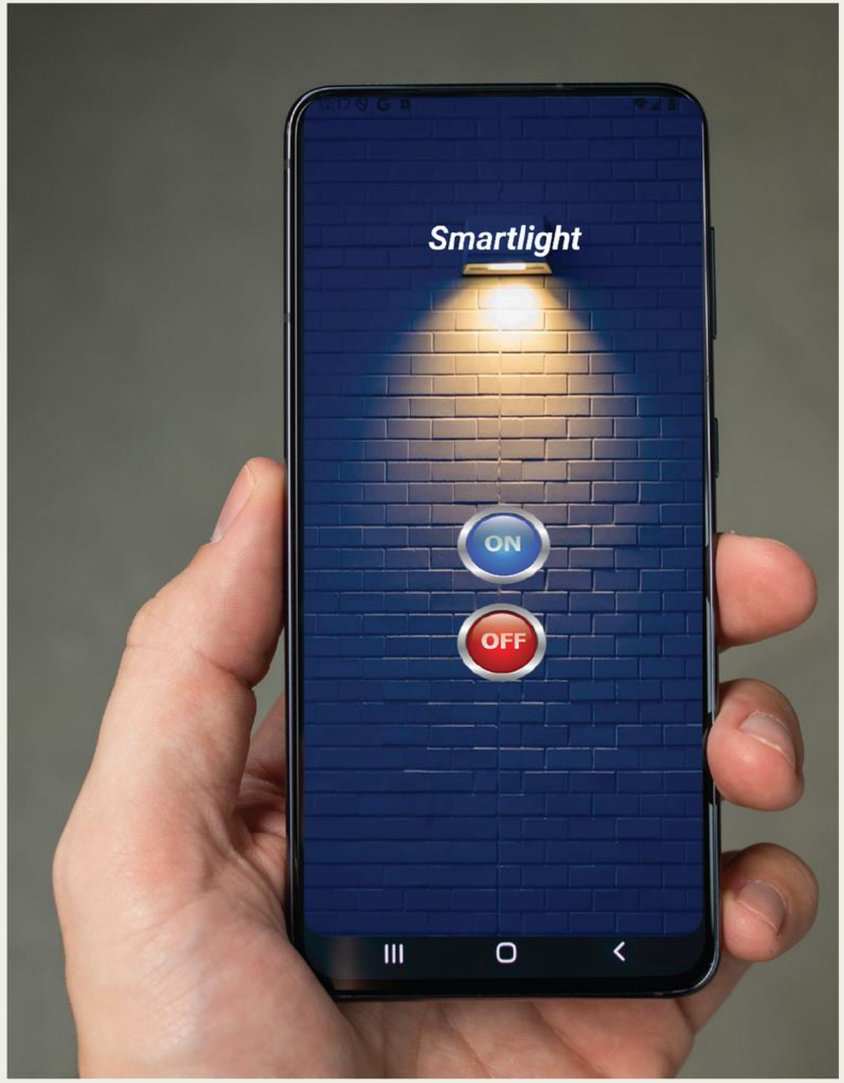
GROUP K

KHANH DO (TEAM LEADER)
JIAXIN LI (UI - ANDROID APP)
OUMAIMA OUBIHI (BACK-END)
OSKARI LAITINEN (RASBERRY PI PIO)

# PURPOSE AND GOAL

## PURPOSE

The purpose of this project is to design and build a smart light system that can be controlled remotely via a mobile application. The smart light will provide users with the convenience of controlling lighting in their home environment from their smartphone, either through Wi-Fi.



# COST AND BENEFITS

## Cost

To estimate the cost of a smart light IoT project, we will break down the total workload into hardware, software, and development time. We'll also list stakeholders and give an overall project cost estimation in euros (€).

**1. Hardware Costs**

| Component | Quantity | Estimated Unit Price (€) | Total (€) |
|---|---|---|---|
| Raspberry Pi Pico W (or ESP32) | 1 | €7 - €10 | €10 |
| RGB LED (with controller) | 1 | €5 | €5 |
| Resistors (if needed) | Assorted | €0.50 | €0.50 |
| Power Supply | 1 | €10 | €10 |
| PCB Board and Wires | 1 set | €3 | €3 |
| Enclosure (plastic or 3D printed) | 1 | €10 | €10 |

**Total Hardware Cost:** ≈ €38.50

# COST AND BENEFITS

## Cost

2. Software Costs

If using open-source tools and libraries, the software cost would be minimal. The main costs here relate to development and testing.

- Development Tools (IDE, libraries): €0 (assuming open-source libraries)
- Mobile App Development (Custom app or Web-based): If developing in-house, the cost will include developer time (estimated below).

# COST AND BENEFITS

## Cost

3. Development and Labor Costs
We’ll break this down by role, with the average European rates.

| Role | Hourly Rate (€) | Hours | Total (€) |
|---|---|---|---|
| Software Developer (IoT/Pico) | €50 | 50 | €2,500 |
| Mobile App Developer | €50 | 60 | €3,000 |
| UI/UX Designer | €40 | 30 | €1,200 |
| Electrical Engineer | €55 | 20 | €1,100 |
| Project Manager | €60 | 20 | €1,200 |
| Testing and QA | €45 | 20 | €900 |

**Total Labor Costs:** ≈ €9,900

# COST AND BENEFITS

## Cost

**4. Other Costs**

| Other Cost Items | Estimated Total (€) |
|---|---|
| Marketing and User Feedback (if applicable) | €500 |
| Prototyping and Iteration | €500 |
| Server hosting (optional) | €100/year |

**Total Other Costs:** ≈ €1,100

---

**5. Total Project Cost**

**Estimated Total:**

- **Hardware**: €38.50
- **Labor**: €9,900
- **Other Costs**: €1,100

**Total Estimated Cost: ≈ €11,038.50**

# COST AND BENEFITS

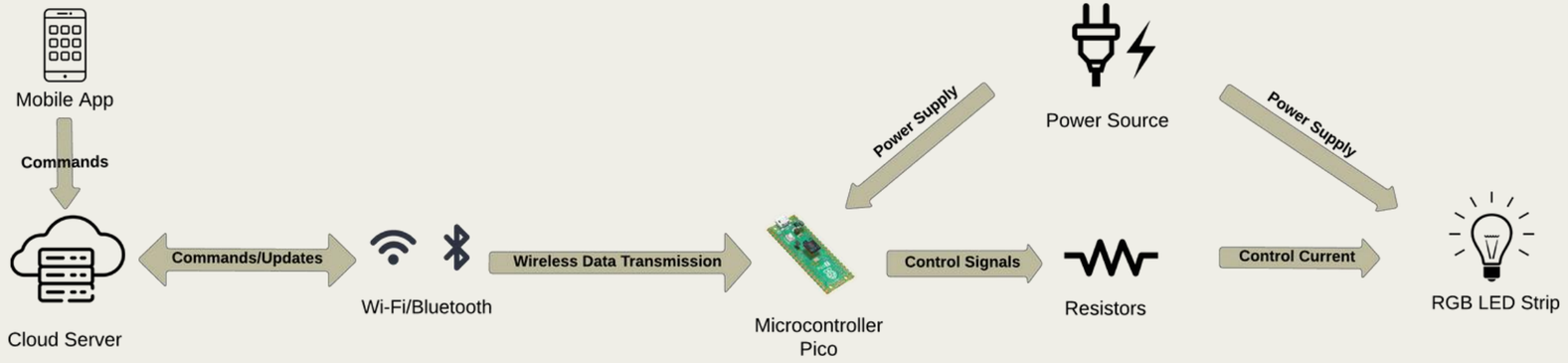
## Benefits

The product owner or investors gain profit through the sales of smart light devices and possibly through long-term services (e.g., smart home integration, data services).
Revenue Streams:
Device sales: Sell the smart lights to consumers or wholesale to retailers.
Subscription model: Offer premium services such as advanced app features, remote access (cloud-based control), or integration with smart home ecosystems like Amazon Alexa, Google Home, etc.
Partnerships: Collaborate with other companies in the IoT or smart home sectors to cross-sell products or services.
Return on Investment (ROI):
With a production cost of approximately €40 per unit, a market price of €80-€100 provides a healthy margin.
Recurring revenue models like monthly subscriptions (€2-5/month) for premium services could lead to long-term profits.

# HOW TO MAKE SMART LIGHT SYSTEM

## Steps:

1. Set Up a virtual private server (VPS) to Store and manage the Light's Status.
2. Create an android app that can connect to VPS for sending request to change the light status.
3. Program the Raspberry pi Pico W set to connect it to VPS to get status of the light and responds accordingly.
4. Test and Run the System



# Set Up a virtual private server (VPS) to Store and manage the Light’s Status.

---

## Steps:

1. **Connect to Azure VM**
2. **Install NodeJS on Azure VM**
3. **Set up a project on VM, and create a file app.js**
   - Create a simple API server listening on the port 3001.
   - Create 3 API end-points
     a. set-light-on: Turns the light on and returns the message "Light is ON."
     b. set-light-off: Turns the light off and returns the message "Light is OFF."
     c. get-light-status: Returns the current status of the light

**4. Open Port 3001 in Azure**

**5. Start the Server on VM**

# Set Up a virtual private server (VPS) to Store and manage the Light’s Status.

---

## Steps:

6. **Access the API Endpoints.** (using web browers, access from everywhere)

**Turn the light on**:

- URL: `http://20.93.3.161:3001/set-light-on`

**Turn the light off**:

- URL: `http://20.93.3.161:3001/set-light-off`

**Check the light status**:

- URL: `http://20.93.3.161:3001/get-light-status`

7. **Keep the Server Running**.

To ensure the server runs even after we disconnect from the SSH session, use **pm2**

# Set Up a virtual private server (VPS) to Store and manage the Light’s Status.

## Code:

```
### **Step 4: Open Port 3000 in Azure**
1. Log in to the Azure portal.
2. Navigate to VM's **Networking** settings.
3. Add a new **Inbound Port Rule**:
   - **Destination Port**: `3000`
   - **Protocol**: TCP
   - **Action**: Allow
   - **Priority**: Set appropriately (e.g., `1000`).
   - **Name**: `Allow-Port-3000`


        General Rule Configuration for HTTP (Port 3000)**:
        | **Field**                 | **Value**                                                                |
        |---------------------------|--------------------------------------------------------------------------|
        | **Source**                | `Any`                                                                    |
        | **Source IP addresses/CIDR ranges** | `*` (Allow all IP addresses) or specify IP in CIDR
        24`). |
        | **Source Port Ranges**    | `*` (Any)                                                                |
        | **Destination**           | `Any`                                                                    |
        | **Destination Port Ranges** | `3000` (The port our API server is running on)                         |
        | **Protocol**              | `TCP` (Because HTTP uses TCP)                                            |
        | **Action**                | `Allow`                                                                  |
        | **Priority**              | A low value, e.g., `100` (higher priority than other rules).             |
        | **Name**                  | A meaningful name, e.g., `Allow_HTTP_API`                                |
```

# Create an android app that can connect to VPS for sending request to change the light’s status.

---

## Steps:

1. **Design the UI with 2 buttons (On and Off)**
2. **Code the JAVA file to make sure the app can control the status of a light by making HTTP GET requests to a server, and to update the app's UI accordingly.**

Here's a detailed explanation:

- turnLightOn(View view): Sends a GET request to the endpoint /set-light-on to turn the light on and updates the app's background.
- turnLightOff(View view): Sends a GET request to the endpoint /set-light-off to turn the light off and updates the app's background.

- changeBackground(boolean status): Updates the background based on the light's status (true for ON, false for OFF). The implementation of changeBackground changes the UI appearance.

# Set Up a virtual private server (VPS) to Store and manage the Light’s Status.

## Code:

MainActivity.java | activity_main.xml | AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission android:name="android.permission.INTERNET" />


    <application
        android:networkSecurityConfig="@xml/network_security_config"
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="smartlight_final"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Smartlight_final"
        tools:targetApi="31">
        <activity
            android:name=".MainActivity"
            android:exported="true"
            android:theme="@style/Theme.Smartlight_final">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHE
            </intent-filter>
        </activity>
    </application>

</manifest>
```

xml

MainActivity.java | activity_main.xml | AndroidManifest.xml | data_extraction_rules.xml

```java
public class MainActivity extends AppCompatActivity {


    1 usage
    public void turnLightOn(View view) {
        sendRequest( urlString: "http://20.93.3.161:3001/set-light-on");
        changeBackground( isLightOn: true);
    }

    1 usage
    public void turnLightOff(View view) {
        sendRequest( urlString: "http://20.93.3.161:3001/set-light-off");
        changeBackground( isLightOn: false);
    }

    2 usages
    private void sendRequest(String urlString) {
        new Thread(() -> {
            try {
                URL url = new URL(urlString);
                HttpURLConnection connection = (HttpURLConnection) url.openConnection();
                connection.setRequestMethod("GET");
                connection.getResponseCode(); // Trigger the request
                connection.disconnect();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }).start();
    }
    2 usages
    private void changeBackground(boolean isLightOn) {
        if (isLightOn) {
```

java

# Program the Raspberry pi Pico W set to connect it to VPS to get status of the light and responds accordingly

## Steps:

1. **Set Up Wi-Fi and HTTP Communication (Use the `urequests` library in MicroPython to send HTTP requests from the Pico to the server.)**
2. **Write Code for Polling Light Status and deal with PIR sensor**

Here's a detailed explanation:

- The Raspberry Pi Pico will use “polling” to check the light’s current status by calling the `/get-light-status` endpoint.
- If it receives a "true" status, it turns the light on; if "false," it turns the light off

# Set Up a virtual private server (VPS) to Store and manage the Light’s Status.

## Code:

```
# Connect to Wi-Fi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect('iPhone', 'tintinnemonemo')

while not wlan.isconnected():
    pass

# Function to check light status from server
def check_light_status():
    try:
        response = urequests.get("http://20.93.3.161:3001/get-light-status")
        data = response.json()
        response.close()
        return data["light_on"]
    except Exception as e:
        print("Error checking light status:", e)
        return None

# Function to update light status on the server
def update_light_status_on_server(light_status):
    try:
        url = "http://20.93.3.161:3001/set-light-on" if light_status else "http://20.93.3.161:3001/set-light-off"
        response = urequests.post(url)
        response.close()
    except Exception as e:
        print("Error updating light status:", e)
```

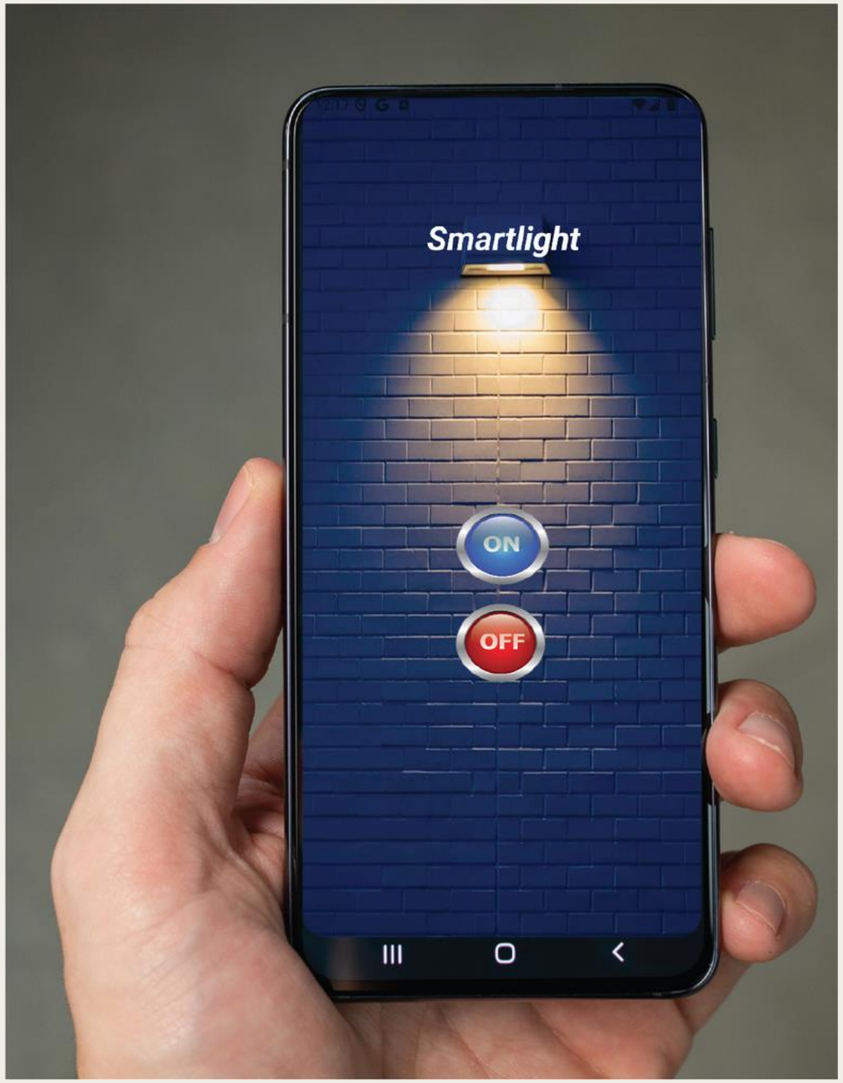
```
# Polling loop for PIR sensor and server
light_status = False

while True:
    try:
        # Step 1: Check light status from the server
        server_light_status = check_light_status()
        if server_light_status is not None and server_light_status !=
            light_status = server_light_status
            if light_status:
                led.high()
            else:
                led.low()

        # Step 2: Handle PIR sensor input
        if pir_sensor.value() == 1:
            if not light_status:  # If the light is off, turn it on
                light_status = True
                led.high()
                update_light_status_on_server(light_status)
        else:
            if light_status:  # If the light is on, turn it off
                light_status = False
                led.low()
                update_light_status_on_server(light_status)

        time.sleep(1)  # Poll every second
    except Exception as e:
        print("Unexpected error in main loop:", e)
```

# Test and Run the System



# Kiitos

Video link

Gitlab link