Sunshine Celebrator

Haichao Xing

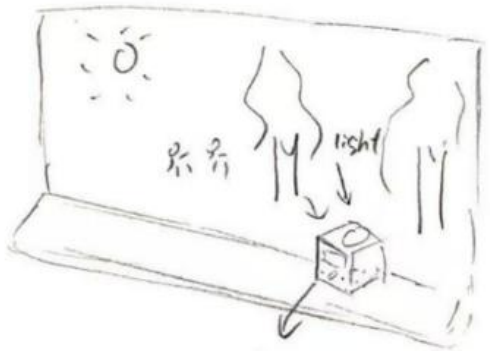
Github link: https://github.com/Maris-26/Sunshine-Celebrator.git

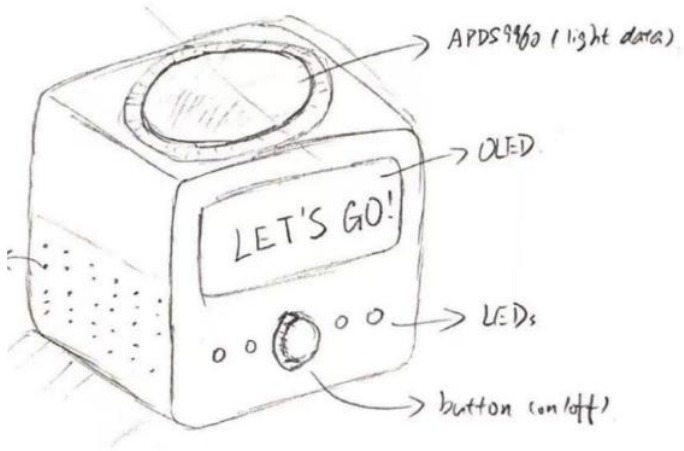
Current Problems

Seattle’s winters are long, dark, and often cloudy, making sunshine feel rare and special. The short daylight hours and constant overcast skies make it easy to **miss those fleeting moments of sunlight**.

People want a way to **instantly notice when the sun is shining** so they don’t miss those bright, uplifting moments.

Solution proposal



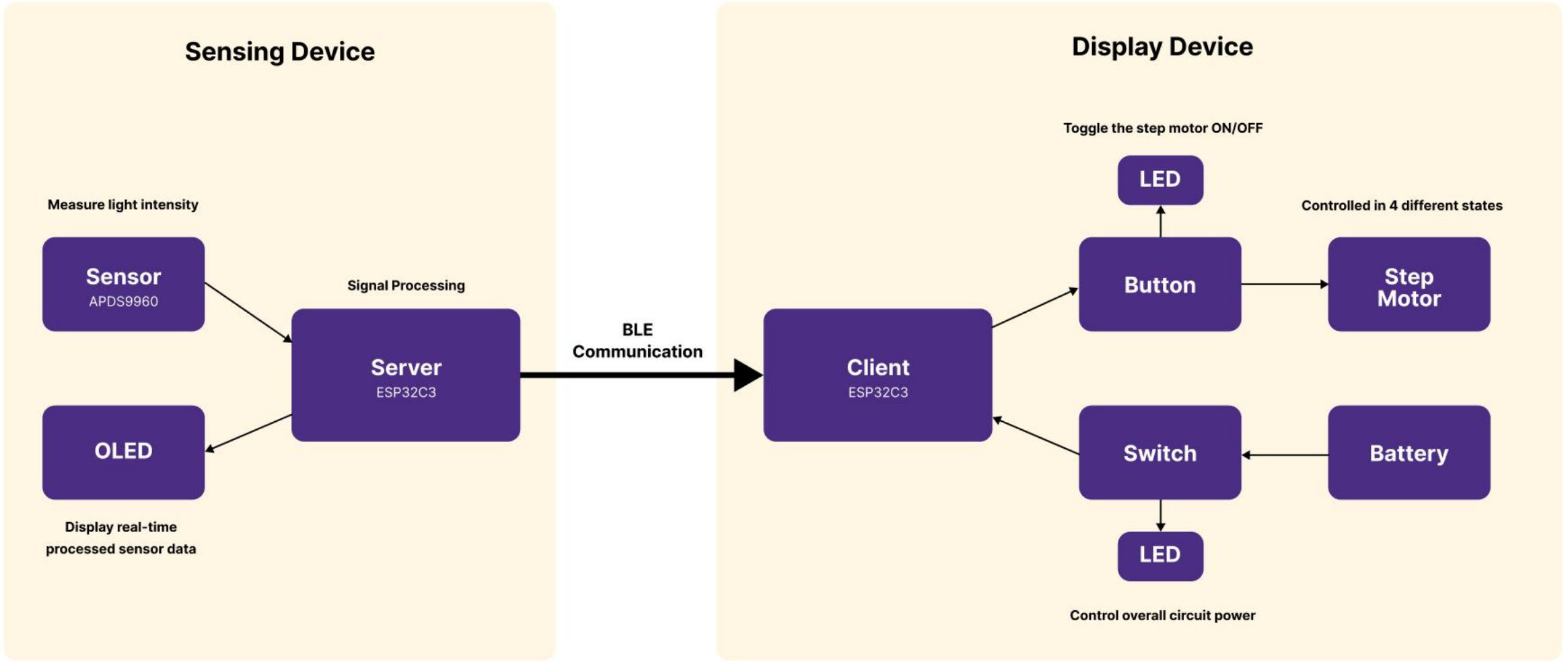


Sunshine Celebrator enhances the experience of bright days by **detecting sunlight levels and visually displaying real-time changes.**

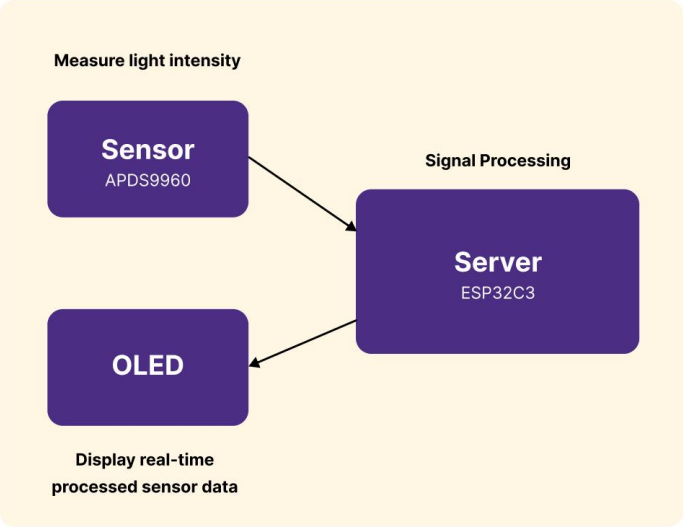
Using an APDS9960 light sensor, the device processes light intensity and responds through an OLED display and a stepper motor.

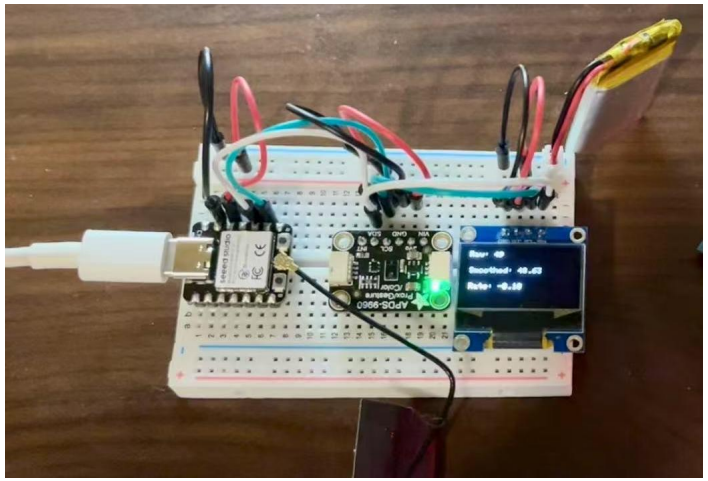
Users can place the sensing device on a windowsill while keeping the display device indoors, allowing them to instantly see and appreciate when the sun is shining.

System Architecture



Sensing Device





Signal Processing:

- ***Moving Average*** → Reduces noise by averaging last 5 sensor readings
- ***Exponential Weighted Moving Average*** → Smooths fluctuations with a 0.3 weight factor
- ***Rate Calculation*** → Measures the rate of change to detect intensity variations

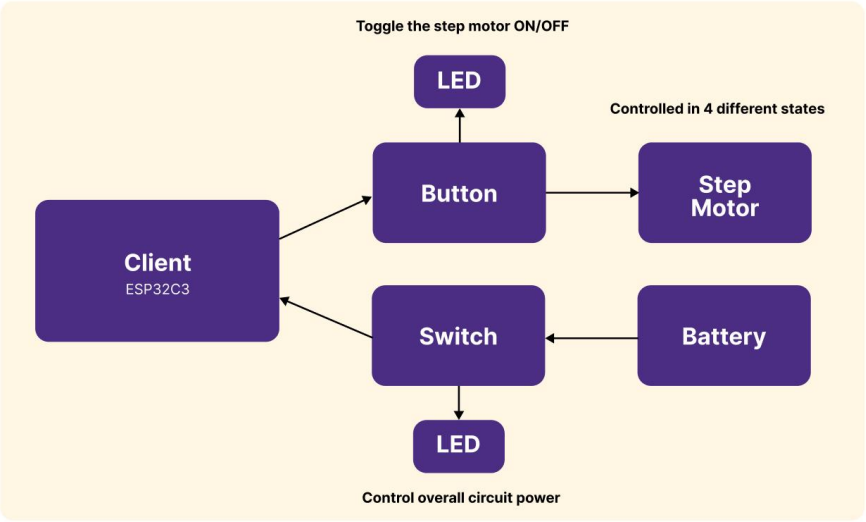
Processed sensor data (OLED):

- Raw → Direct sensor readings without processing
- Smoothed → More stable data with noise reduction
- Rate → Helps the client determine how fast the light intensity is changing

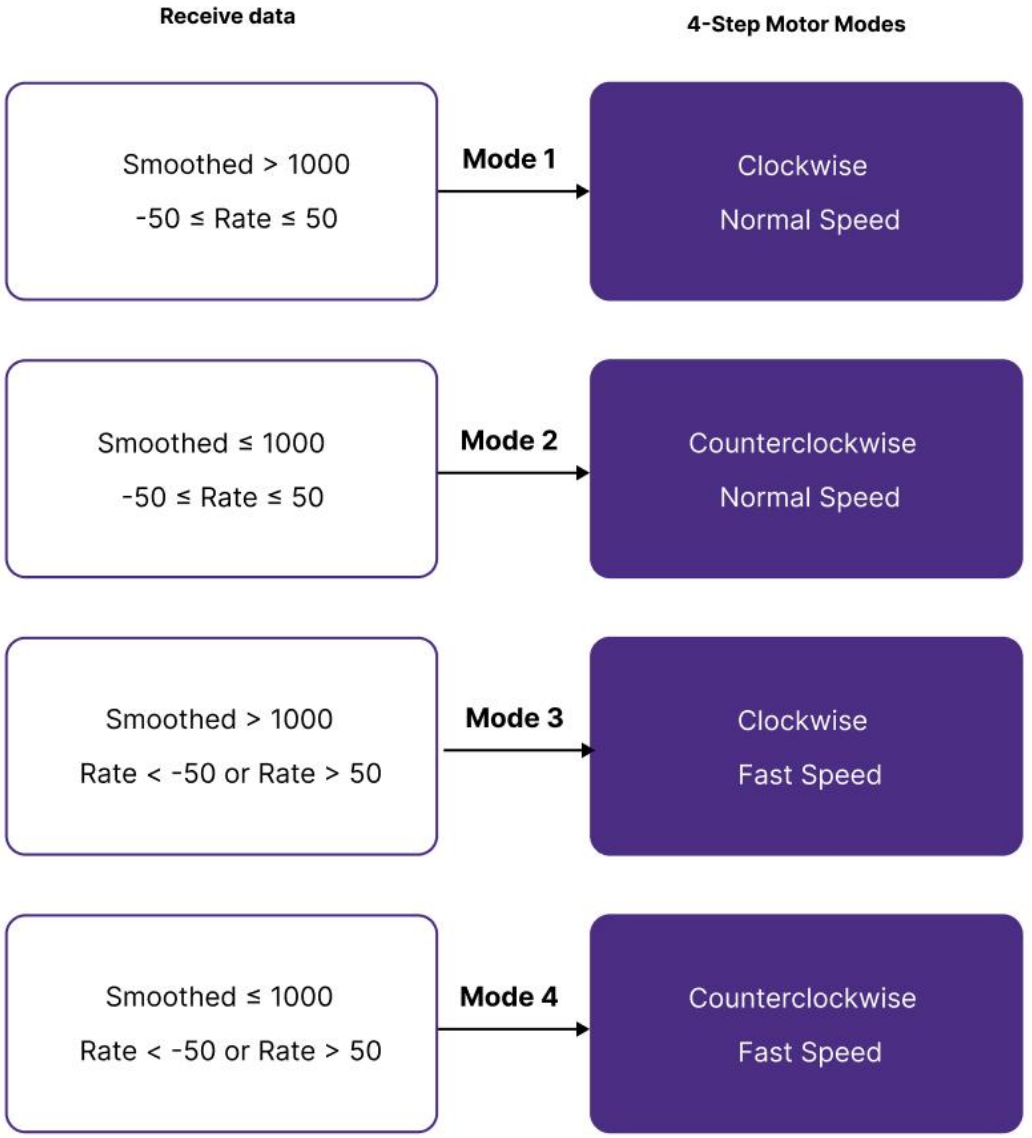
Effect:

- Reduces BLE Data Transmission Load

Display Device

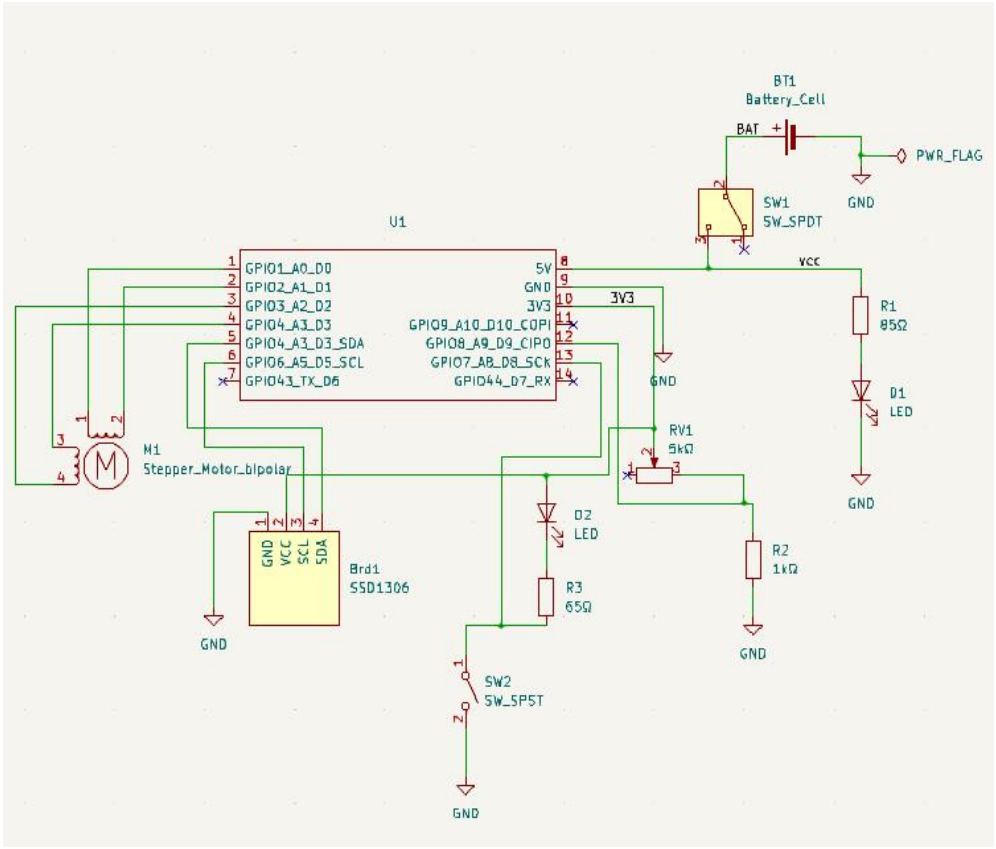


- The stepper motor cycles through 4 predefined states, determined by Smoothed Light Intensity (Smoothed) and Rate of Change (Rate)

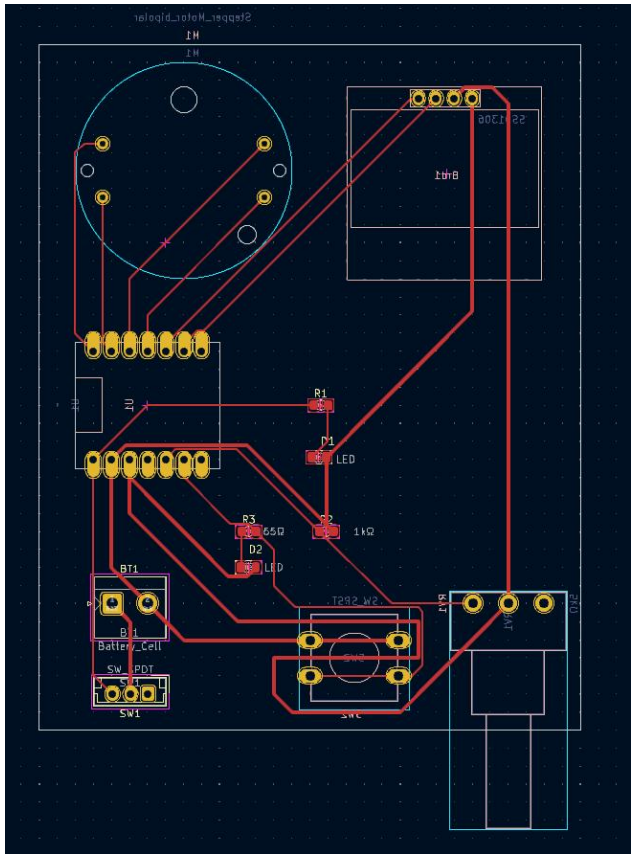


Display Device

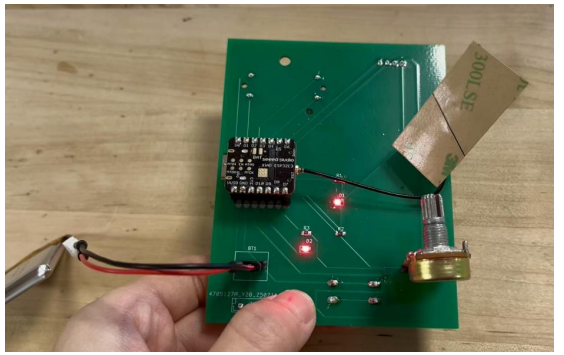
Project Schematic

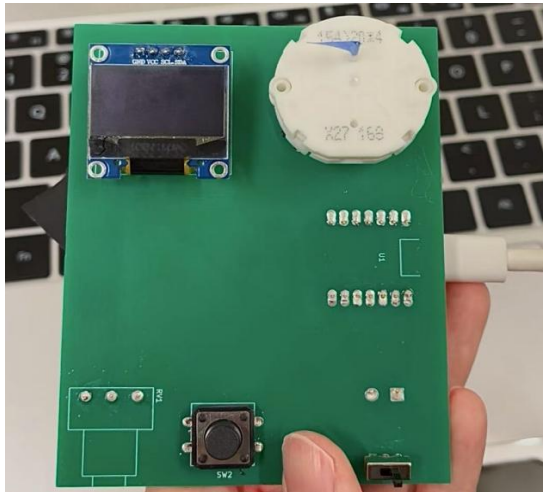


PCB Editor

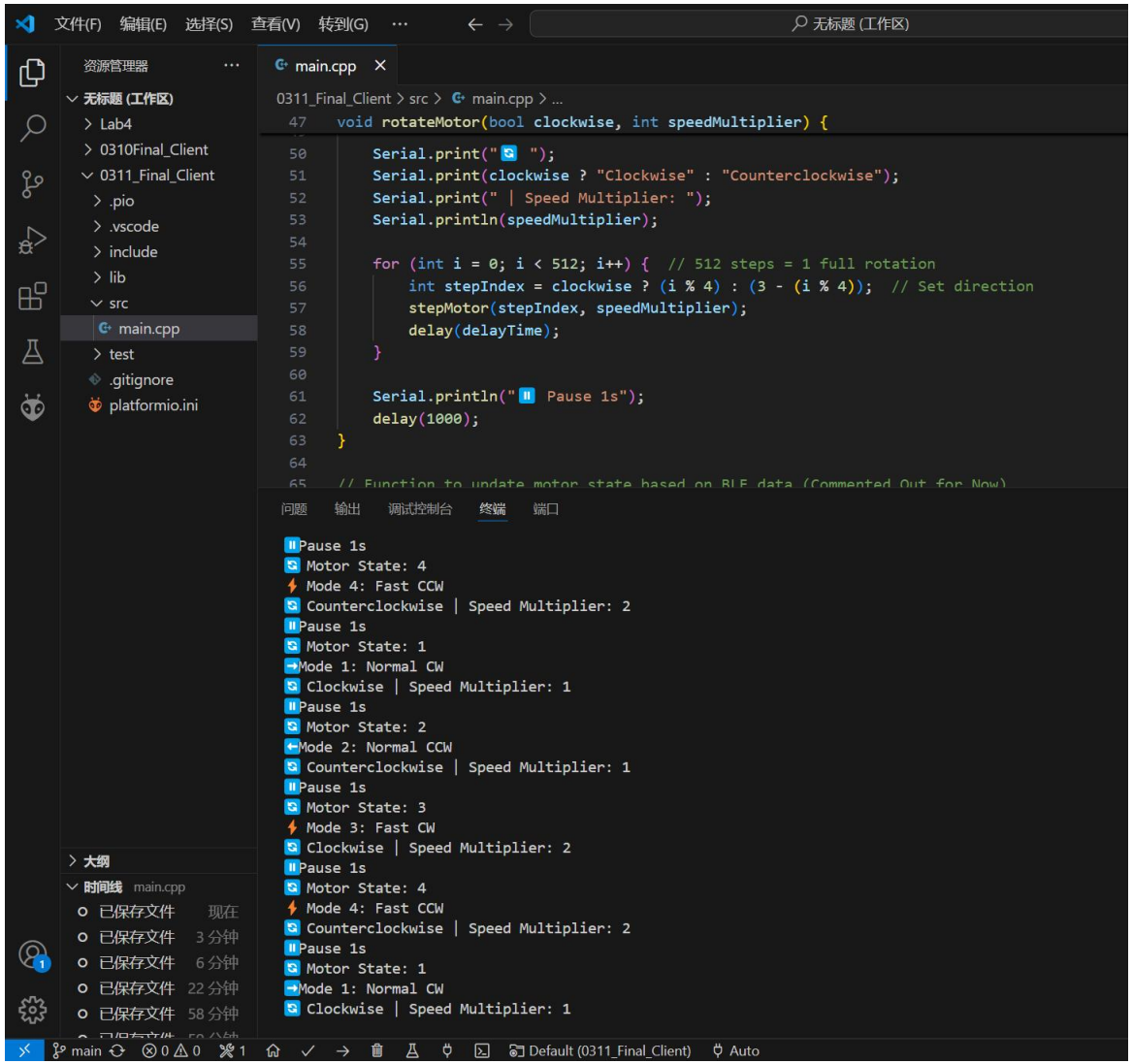


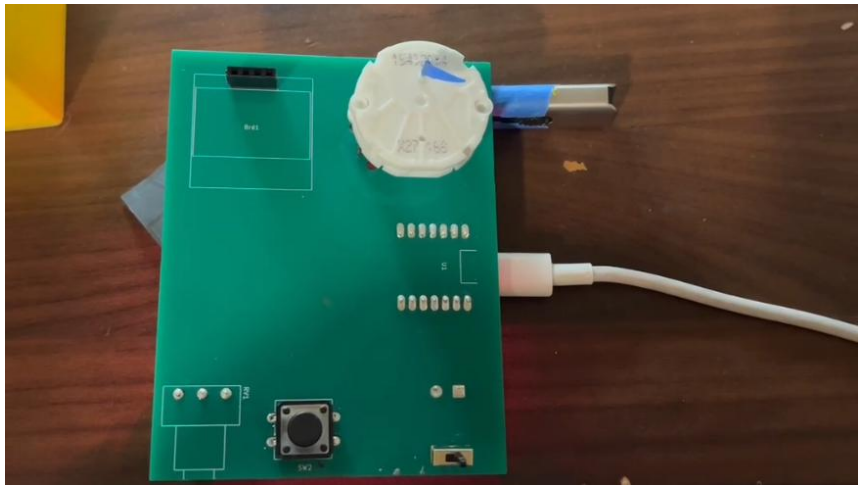
Printed PCB

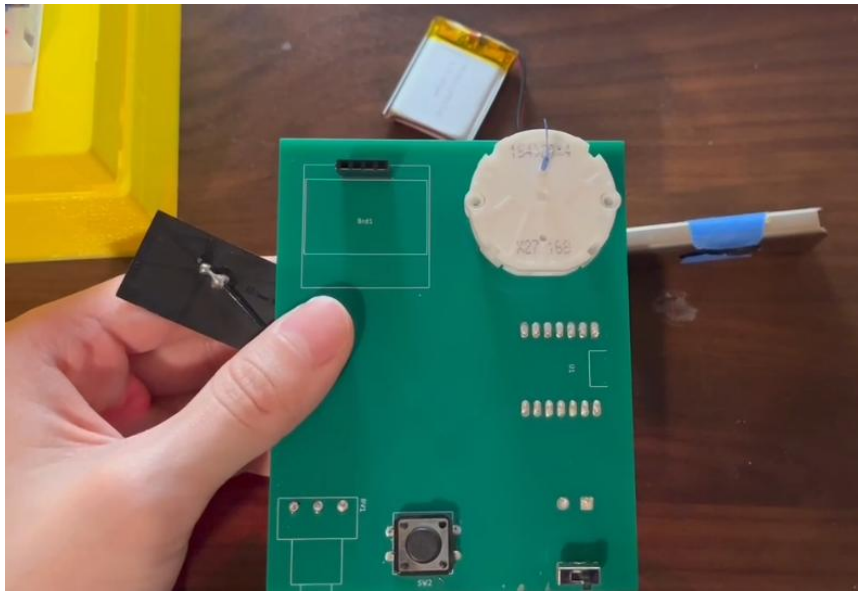




Display Device

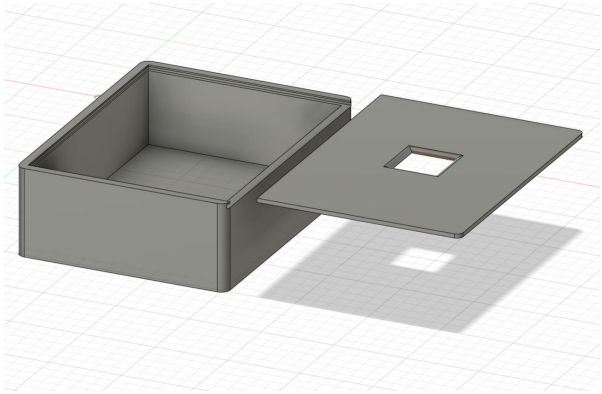


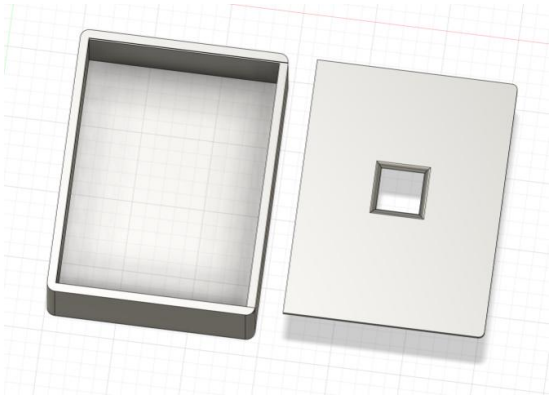




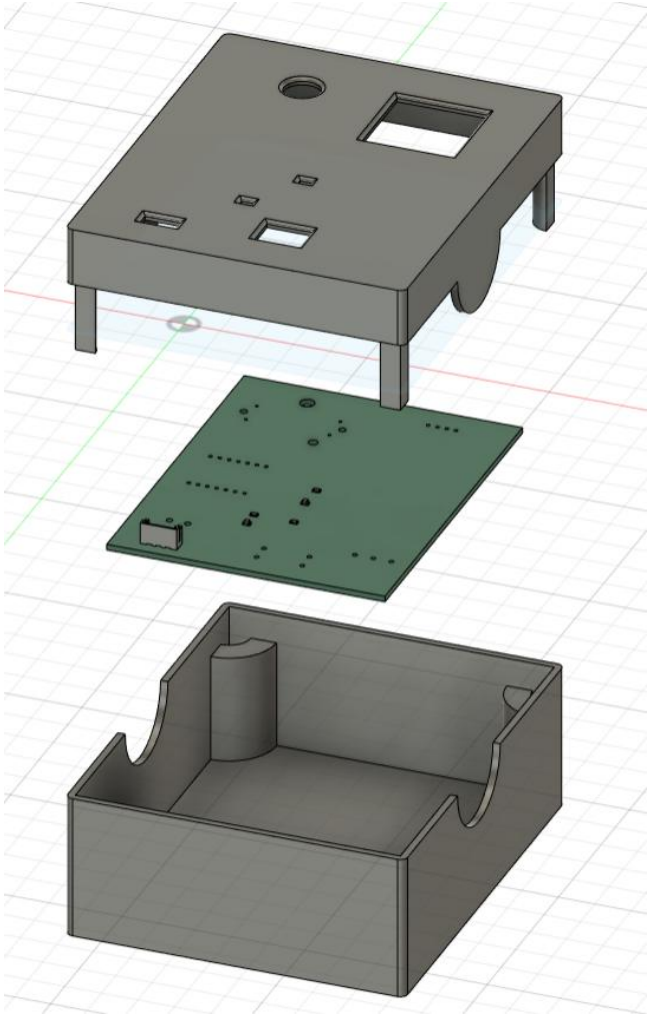
Enclosure Design

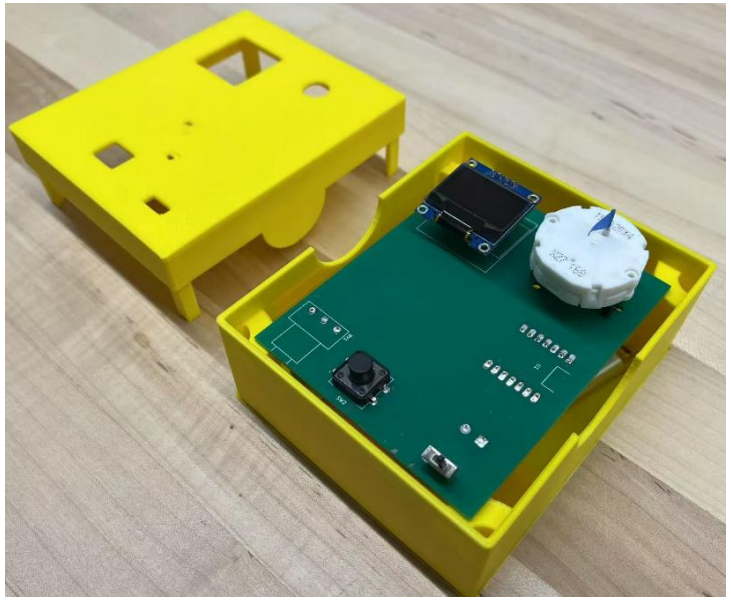
For Sensing Device





For Display Device





BLE Communication

Server -- succeed

sketch_mar11a | Arduino IDE 2.3.4

File Edit Sketch Tools Help

XIAO_ESP32C3

sketch_mar11a.ino

```
#include <Arduino.h>
#include <Wire.h>
#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEServer.h>
#include <Arduino_APDS9960.h>  // 确保使用正确的库

// 创建 BLE UUID
#define SERVICE_UUID        "35b88443-045d-4687-8fc4-a79c946baab8"
#define CHARACTERISTIC_UUID "c5cc25dd-2467-4829-87bc-55ccf7b0b622"

BLEServer *pServer = NULL;
BLECharacteristic *pCharacteristic = NULL;
bool deviceConnected = false;

// **正确声明 APDS9960 传感器**
APDS9960 apds(Wire, -1);  // 传入 I2C 和中断引脚（-1 代表不使用中断）

class MyServerCallbacks : public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
        deviceConnected = true;
        Serial.println("Client Connected!");
    }

    void onDisconnect(BLEServer* pServer) {
```

Output Serial Monitor ×

Message (Enter to send message to 'XIAO_ESP32C3' on 'COM7')

```
Light Intensity (C): 394
Light Intensity (C): 391
Light Intensity (C): 219
Light Intensity (C): 103
Light Intensity (C): 103
Light Intensity (C): 103
```

Client -- succeed

client | Arduino IDE 2.3.4

File Edit Sketch Tools Help

XIAO_ESP32C3

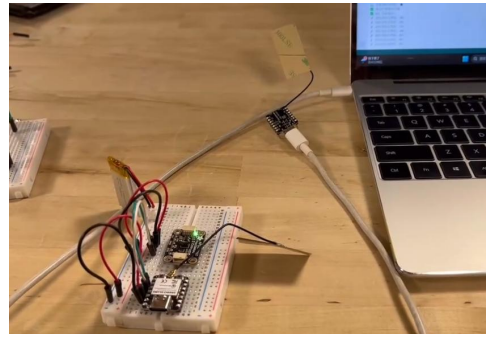
client.ino

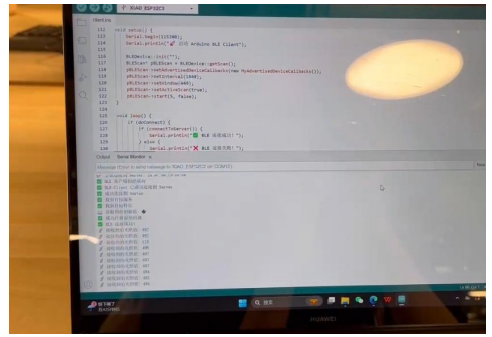
```
void setup() {
    Serial.begin(115200);
    Serial.println("🚀 启动 Arduino BLE Client");

    BLEDevice::init("");
    BLEScan* pBLEScan = BLEDevice::getScan();
    pBLEScan->setAdvertisedDeviceCallbacks(new MyAdvertisedDeviceCallbacks());
    pBLEScan->setInterval(1349);
    pBLEScan->setWindow(449);
    pBLEScan->setActiveScan(true);
    pBLEScan->start(5, false);
}

void loop() {
    if (doConnect) {
        if (connectToServer()) {
            Serial.println("✅ BLE 连接成功！");
        } else {
            Serial.println("❌ BLE 连接失败！");
```

Output Serial Monitor ×

Message (Enter to send message to 'XIAO_ESP32C3' on 'COM15')

```
🔗建成功
✅ BLE Client 已成功连接到 Server

✅ 找到目标服务
✅ 找到目标特征
📖 读取到的初始值: h
✅ 成功注册通知回调
✅ BLE 连接成功！
📡 接收到的光照值: 104
📡 接收到的光照值: 104
📡 接收到的光照值: 104
📡 接收到的光照值: 107
```



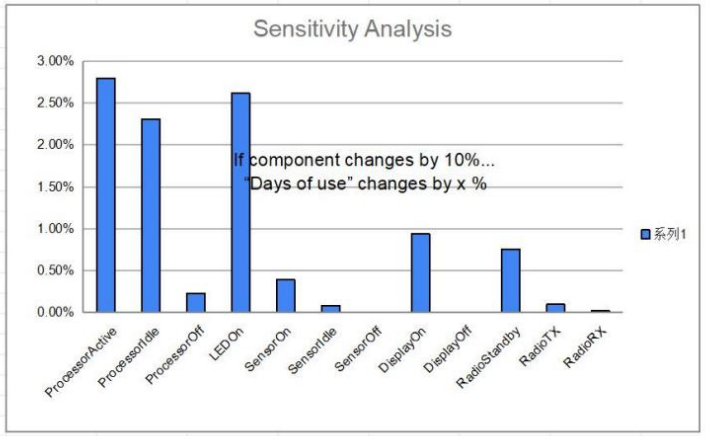


The recording of a successful BLE connection between the server and client is also on GitHub.

Battery & Budget Summary

Battery Life Estimate

System Parameters (defined by hardware) form the datasheets		Profiles (usage of each component mode - defined by software and usage) "off"	"sensing"	"interactive"
Processor (ESP32C3)				
Active	80 mW	0%	20%	25%
Idle	20 mW	0%	75%	70%
Sleep	1 mW (same as default)	100%	5%	5%
		(same as default)		
LED				
On	15 mW	70%	0%	30%
Sensor (APDS9960)				
On	5 mW (same as default)	0%	50%	50%
Idle	1 mW (same as default)	0%	50%	50%
Off	0 mW	100%	0%	0%
		(same as defau	(same as defau	(same as default)
Display (OLED)				
On	15 mW	0%	0%	100%
Off (leakage)	0 mW	100%	100%	0%
		(same as defau	(same as defau	(same as default)
Radio (BLE)				
Data Rate	300 bps (same as default)	0%	0%	0%
Standby Power	5 mW (same as default)	0%	97%	97%
TX Power	30 mW	0%	2%	2%
RX Power	10 mW (same as default)	0%	1%	1%
		(same as defau	(same as defau	(same as default)
		14	6	4 hours/day typical usage
Battery				
Capacity	1000 mAh			
Nominal Voltage	3.7 V			
Regulator Efficiency	90% (same as default)			



Total power in profile (mw)		Maximum Time
"off"	11.5 mW	289.6 hours
"sensing"	39.6 mW	84.1 hours
"interactive"	62.1 mW	53.6 hours
Effective Battery Capacity	3330 mW*h	
Days of Use	**5.15 days**	
Hours of Use	**123.52 hours**	

REFLECTIONS : WHAT DID YOU LEARN FROM ANALYZING YOUR POWER. TALK ABOUT SOME POTENTIAL TRADEOFFS.

Budget Summary

$9.9

1 more*ESP32C3

$16.58

2*3.7V1000mAh Battery

$0

OLED, APDS9960, PCB, Step Motor, Switch, Button, LED...

Total

$26.48

Future Work

1. Improve BLE Stability

Optimize the Bluetooth connection between the server and client to ensure more reliable real-time data transmission.

2. Processed sensor data (OLED)

Improve motor response consistency and reduce any unintended fluctuations in movement.

3. Effect:

Enhance OLED visualization by adding real-time feedback and historical sunlight trends for better user interaction.

Thank You