Project Title: EcoSmart

IoT-Based Smart Energy Management System

Team Member 1: Jose Carlos Rodriguez Saavedra
jrodriguezsaav2186@sdsu.edu

Team Member 2: Azza Laz
alaz7039@sdsu.edu

ABSTRACT

Our project, EcoSmart, is designed to help reduce unnecessary energy consumption by automatically managing lighting in indoor environments. The system uses a light-dependent resistor (LDR) to detect ambient brightness and an accelerometer (LSM6DSO) to detect motion. The ESP32-based TTGO board processes sensor inputs and activates a simulated light source (LED) accordingly. If the room is dark or motion is detected, the LED turns on; otherwise, it remains off to save energy. Real-time feedback is provided via the onboard TFT display, and the system logs status updates to a Flask server hosted on an AWS EC2 instance. This project demonstrates how IoT devices can provide scalable, energy-efficient solutions for homes and buildings.

INTRODUCTION

Motivation/Background (3 pts)
The motivation of our project was to build a system that helps reduce energy waste in buildings and homes where lights are often left on. In many cases, lights remain active even when a room is unoccupied or when natural daylight is sufficient. This not only wastes electricity but also contributes to higher utility costs and environmental strain. By combining motion detection and ambient light sensing, we aim to create an automated lighting control system that makes real-time decisions to optimize energy usage. Our approach improves upon basic timer- or light-only systems by using dual-sensor logic to ensure lighting is only active when truly needed.

Project Goals (6 pts)
The primary goal of EcoSmart is to develop a functional and intelligent IoT-based lighting system that:

- Uses motion and light sensors to determine when lights should be on or off
- Provides real-time feedback through an OLED display
- Sends status updates to a cloud server (AWS EC2) via WiFi for logging into a CSV file
- Minimizes energy waste in indoor spaces
- Demonstrates an accessible, low-cost solution for smart home or office applications

Assumptions (3 pts)

- The system will be used in small to medium-sized rooms where one light source is sufficient.
- The ESP32 device will be connected to a stable 2.4GHz WiFi network.
- Sensor placement is assumed to cover the effective detection area for motion and ambient light.

- We assume user interaction with the system is minimal, and it operates autonomously once installed.

SYSTEM ARCHETICTURE

The EcoSmart system integrates hardware, software, and cloud components to enable real-time monitoring and reporting of environmental conditions. The architecture consists of:

1. **ESP32 TTGO LoRa32**: Serves as the main microcontroller, responsible for reading data from sensors, processing it, displaying output on the TFT screen, and transmitting data via WiFi and Bluetooth.

2. **Sensors**: Includes a light sensor (LDR) for ambient light detection and an onboard accelerometer (LSM6DSO) for motion detection via I²C communication.

3. **Display Module**: A TFT_eSPI display connected over SPI to visually present sensor readings and device status.

4. **WiFi Connectivity**: The ESP32 connects to a wireless network and transmits sensor data to a cloud server hosted on an AWS EC2 instance. Data is sent using HTTP GET requests to a Flask web server listening on port 5000.

5. **Bluetooth Connectivity**: In parallel, the ESP32 broadcasts sensor readings over Bluetooth for local device pairing and monitoring using a mobile app.

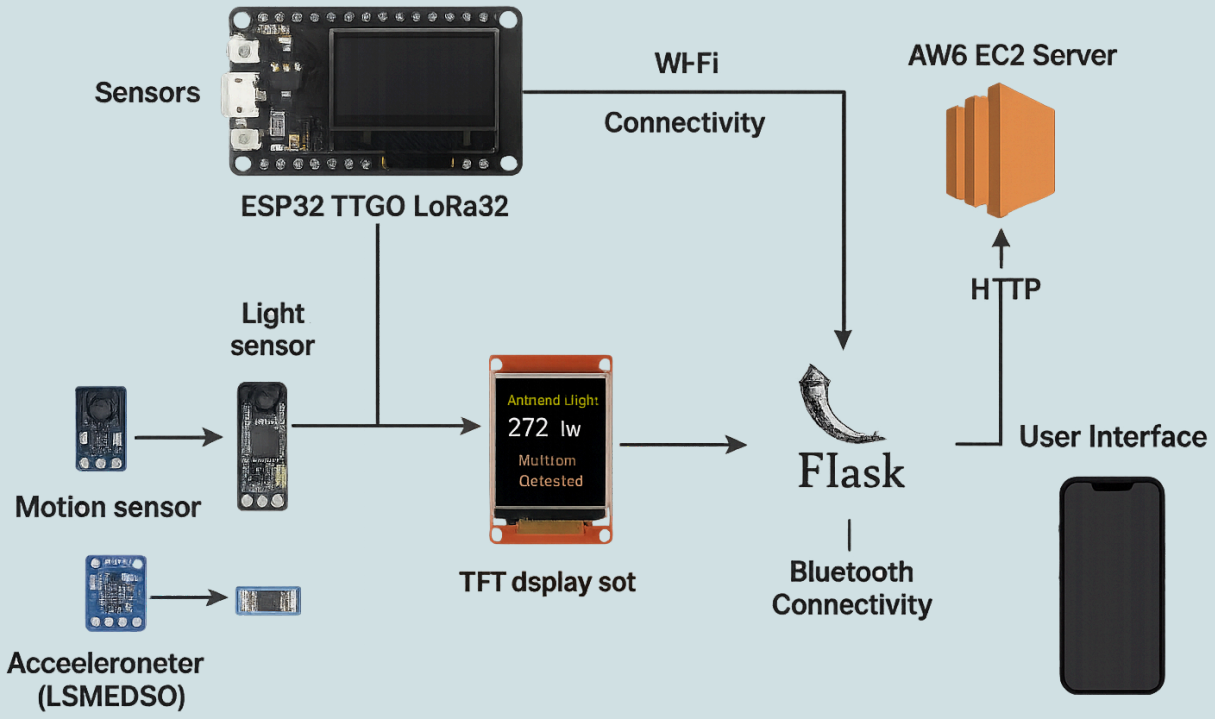
6. **AWS EC2 Server**: Hosts a Flask application that receives incoming data, parses it, and logs it to a CSV file for storage and later analysis.

7. **User Interface**: Users can view data in real-time on the ESP32's display, receive Bluetooth notifications on a mobile device, or access logged data from the cloud server for further analytics.

This architecture provides a flexible, multi-channel approach to data acquisition, visualization, and cloud storage, enabling both local and remote monitoring.

DIAGRAM BELOW

ARCHITECTURE DIAGRAM



Final List of Hardware Components:

Component	Quantity
TTGO LoRa32 (ESP32 Dev Board)	1
Light Sensor (photoresistor)	1
LSM6DSO Accelerometer	1
LED	1
Resistor (for sensors)	1
Wires (Dupont cables)	9

PROJECT IMPLEMENTATION (30 PTS)

Tasks/Milestones Completed (15 pts)

- Integrated and programmed LDR and motion sensors on the ESP32 for environment monitoring
- Connected the ESP32 to Wi-Fi and AWS EC2 cloud server for remote data logging and monitoring
- Configured Bluetooth communication on the ESP32 for wireless data transmission to mobile devices

- Set up and customized the TFT display to visualize real-time sensor readings and device status
- Developed and tested the demo, including live data logging to a CSV file and visualization of system behavior

Task Completed	**Team Member**
Sensor integration & coding	Azza & Carlos
Wi-Fi & AWS cloud connection	Carlos
Bluetooth communication setup	Azza
Display (TFT) configuration	Azza
Demo & data logging setup	Carlos

Challenges/Roadblocks (5 pts)

- Sensor Library Issues: The default LSM6DSO libraries didn't work reliably on ESP32. We solved this by writing direct I2C communication to retrieve motion data manually.
- OLED + WiFi Conflicts: We encountered timing issues when the ESP32 tried to run WiFi and update OLED simultaneously. We resolved this by adding delays and restructuring loop logic.
- I2C Wiring: Initially, incorrect pin assignments caused failed sensor readings. Re-checking datasheets and using GPIO 21/22 resolved the issue.

Tasks Not Completed

Task	**Reason**
Web dashboard visualization	Time constraints; prioritized core functionality
Cloud database integration	AWS EC2 logging was used instead
Advanced data analytics	Basic CSV logging implemented instead

WEAK POINTS / FUTURE WORK

1. Physical Enclosure & Deployment: The current prototype is built on a breadboard. For real-world use, the system should be housed in a durable enclosure and mounted for better sensor coverage.
2. Multi-Room Scalability: Currently designed for single-room control, the system could be extended to manage multiple rooms/zones with individual logic.
3. Enhanced Analytics & Dashboard: Integrating a web or mobile dashboard for visualizing historical sensor data, usage trends, and energy savings would improve usability and user engagement.
4. Power Optimization: Implementing power-saving modes such as deep sleep could extend battery life in remote or off-grid use cases.

SOURCE CODE

https://github.com/carlssrodriguez/EcoSmart.git