Project

Building Energy Optimization

Aim

Our goal is to build a website that monitors power consumption in various departments and labs, provides visualizations of power usage, detects power losses, and gives users recommendations to save power.

Workflow :-

https://drive.google.com/file/d/1TTr3YMh0WQ9UsSHSC6ailqaJvP6r2aUq/view?usp=sharing

Benefits of our project

1) By monitoring and analyzing power usage, the system helps us to identify areas where energy is being wasted, leading to more efficient use of energy and reduced consumption and hence reduced electricity .
2) Visualizations and detailed reports provide insights into energy consumption patterns, helping decision-makers to allocate resources more effectively and plan for future energy needs.
3) Real-time data collection allows users to monitor power usage instantly, enabling them to take immediate action to prevent energy loss.

Technologies

Front End :

1) HTML
2) CSS
3) Javascript (Node.js)
4) React.js

BackEnd :

1) .Net Framework (C#)

Database :

1) MySql
2) mongoDB

Hardware :

1) ESP32 or Arduino with Wi-Fi capability
2) ACS712 Current Sensor
3) PIR Motion Sensors
4) Door/Window Sensors
5) Temperature Sensors
6) Connecting wires, resistors, and other electronics essentials

Recommendation :

1. Integrating OpenAI API

Observation :

Task 1) How to Calculate the Power consumption of a Device.

To calculate power consumption using the ACS712 sensor, we need to measure both the current flowing through our system and the voltage supplied to it.

Calculate the Current for Each Reading

Using the formula:

$$\text{Current (Amps)} = \frac{(\text{Measured Voltage} - V_{\text{offset}})}{\text{Sensitivity}}$$

V_{offset} (Voltage with no current) = 2.5V

Current Calculations for each timestamp c1,c2,.. …. … cN .

Calculate Instantaneous Power for Each Reading

Using the formula:

$$\text{Power (Watts)} = \text{Voltage (Volts)} \times \text{Current (Amps)}$$

Power Calculations for each timestamp:

Calculate Total Power Consumption:

Take Sum of all Power Readings

Convert to Watt-Hours:

$$\text{Total Power Consumption (Wh)} = \left(\text{Average Power (W)} \times \frac{\text{Total Time (Seconds)}}{3600}\right)$$

Circuit Diagram :-
https://app.diagrams.net/#G1dFoMlwQe7tX8dXTp_n8bUYXpcwvqbKv1#%7B%22pageId%22%3A%22bELQv2NfKL-ixonXnF-M%22%7D

Task 2) How To Detect Human Presence in a Room:

System Overview: The system detects human presence in a room using multiple PIR sensors, which are connected to ESP8266 modules. These ESP8266 modules communicate wirelessly with a central ESP32, which then sends the data to a server for further analysis or monitoring.

Components:

1. **PIR Sensors:**
    - **Function:** Detects motion and presence of humans in various locations within the room.
    - **Placement:** Positioned at different spots to cover the entire room effectively.
2. **ESP8266 Transmitters:**
    - **Function:** Each PIR sensor is connected to an ESP8266 module.
    - **Operation:** The ESP8266 modules transmit data (0 or 1) indicating the absence or presence of motion, respectively.
    - **Data Transmission:** The data is wirelessly transmitted to a central ESP8266 receiver.
3. **Central ESP8266 Receiver:**
    - **Function:** Receives data from all ESP8266 transmitter modules.
    - **Operation:** Forwards the aggregated data to the ESP32.
4. **ESP32 (Central Controller):**
    - **Function:** Processes the data received from the central ESP8266 module.
    - **Logic:**
        - If all PIR sensors report "0", the ESP32 determines that no one is in the room.
        - If any PIR sensor reports "1", the ESP32 detects human presence in the room.
    - **Data Transmission:** Acts as a gateway to send the processed data to a server for further actions, analysis, and monitoring.

-

Communication Flow:-

PIR Sensor → ESP8266 → ESP32 → Server

Circuit Diagram :-

https://app.diagrams.net/#G1FQjwuEs9PZbbyX0ozxnq-dSu0K11Z6yt#%7B%22pageId%22%3A%22_Ib6XqYBlwlNwOGxmKqh%22%7D

Task 3) How To Detect Doors and Windows Are open

System Overview: The system detects when doors and windows are open using magnetic reed switches. These switches determine the physical state (open/closed) of the door or window by measuring the proximity of a magnet. When the door or window opens, the reed switch's state changes, signaling that it is open.

Magnetic Reed Switches:

- **How it Works:**
    - A magnetic reed switch consists of two parts: the switch and a magnet.
    - The reed switch is placed on the stationary frame (door/window frame), and the magnet is placed on the moving part (door/window).
    - When the door/window is closed, the magnet is near the reed switch, keeping the circuit closed, which indicates the door/window is closed.
    - When the door/window opens, the magnet moves away from the reed switch, opening the circuit and signaling that the door/window is open.

System Components:

1. **Magnetic Reed Switches:**
    - **Placement:** Installed on the frame and moving part of each door/window.
    - **Connection:** Wired to an ESP8266 module to monitor the open/closed states.
2. **ESP8266 Modules:**
    - **Function:** Each magnetic reed switch is connected to its own ESP8266 module.
    - **Operation:** The ESP8266 modules transmit data (0 or 1) indicating the state of the door/window (0 for closed, 1 for open).
3. **Central ESP8266 Receiver:**
    - **Function:** Receives data from all ESP8266 modules connected to the magnetic reed switches.
    - **Operation:** Forwards the aggregated data to an ESP32 for processing.
4. **ESP32 (Central Controller):**
    - **Function:** Processes the data received from the central ESP8266 module.
    - **Data Transmission:** Sends the processed data to a server for further processing or alerting, similar to the human presence detection system.

Magnetic Reed Switch (detects state change) → ESP8266 (sends data) → Central ESP8266 Receiver → ESP32 (central controller processes data) → Server (optional: for analysis or monitoring)

Task 4) How monitor environmental conditions and ensure that temperature power consumption is less (like fans and air conditioners are no need to run if temperature is normal)

Overview: The system monitors temperature and humidity using DHT22 sensors to optimize power consumption by controlling devices like fans and air conditioners. By ensuring these devices operate only when needed, the system helps reduce unnecessary energy usage and improves efficiency.

Sensors and Components Used:

1. **DHT22 (Temperature & Humidity Sensor):**
    - **Temperature Range:** -40°C to 80°C
    - **Temperature Accuracy:** ±0.5°C
    - **Humidity Range:** 0% to 100%
    - **Humidity Accuracy:** ±2-5%
    - **Function:** Provides accurate temperature and humidity readings.
    - **Connection:** Each DHT22 sensor is connected to an ESP8266 module.
2. **ESP8266 Modules:**
    - **Function:** Each ESP8266 module is connected to a DHT22 sensor.
    - **Operation:** Transmits temperature and humidity data from the sensor to a centralized ESP32.
    - **Data Transmission:** Sends data wirelessly to a central ESP32 for processing.
3. **Central ESP32 Controller:**
    - **Function:** Receives data from all ESP8266 modules.
    - **Data Management:** Sends temperature and humidity data to a server for logging and analysis.

Communication Flow:-

DHT22 Sensor → ESP8266 (transmits data) → Central ESP32 (processes data and controls devices) → Server (for logging and analysis)

Task 5) How to Detect Machines is Idle Mode

The ACS712 sensor is used to determine if a machine is in idle mode by measuring its power consumption.

Step 1:- Connect the ACS712 sensor to measure current.

Step 2:- Read and **convert** the analog voltage to current.

current = (analog_reading - V_offset) / Sensitivity   " where `V_offset` is the voltage when no current is flowing (typically half of the supply voltage) and `Sensitivity` is the sensor's sensitivity in mV/A."

Step3:- Set a threshold to differentiate between idle and active states.

Step4:- Implement logic to detect idle mode based on the current measurement.

Components List:

- **1x ESP32 (Centralized Microcontroller)**
- **3x ESP8266 Modules for PIR Sensors**
- **1x ESP8266 Module for the Door Sensor**
- **1x ESP8266 Module for the Window Sensor**
- **2x ESP8266 Modules for Temperature Sensors**
- **1x ESP8266 Module for ACS712 Sensor**
- **1x ACS712 Sensor (for power consumption measurement)**
- **Power Supply:** 5V power supplies or battery packs for each ESP8266 module and the ESP32

Installation and Configuration:

1. **Sensor Nodes Setup:**
    - **PIR Sensors (3 Units):**
        - Connect each PIR sensor to an ESP8266 module using GPIO pins.
        - Program each ESP8266 to read motion data and send it to the ESP32 via HTTP requests or MQTT.
    - **Door Sensor (1 Unit):**
        - Connect the magnetic reed switch to an ESP8266 module.
        - Program the ESP8266 to send door status (open/close) to the ESP32.
    - **Window Sensor (1 Unit):**
        - Connect the magnetic reed switch to another ESP8266 module.
        - Program the ESP8266 to send window status to the ESP32.
    - **Temperature Sensors (2 Units):**
        - Connect each temperature sensor (DHT22 or DS18B20) to an ESP8266 module.
        - Program each ESP8266 to read temperature data and send it to the ESP32.
    - **ACS712 Sensor (1 Unit):**
        - Connect the ACS712 sensor to an ESP8266 module.
        - Program the ESP8266 to read the current measurement from the ACS712 and send it to the ESP32.
2. **Centralized ESP32 Setup:**
    - **Role:** Collect data from all sensor nodes and send it to the server.
    - **Connection:**
        - **Wi-Fi:** Connect to the same Wi-Fi network as the sensor nodes.
    - **Programming:**
        - Write code to handle incoming data from the ESP8266 modules.

        - Process the data from PIR sensors, door sensors, window sensors, temperature sensors, and ACS712.
        - Forward processed data to the server for logging and analysis.
3. **Power Supply:**
    - **For All ESP8266 Modules and ESP32:**
        - Use 5V power supplies or battery packs.
        - Ensure stable power to each module to maintain reliable operation.
4. **Server Setup:**
    - **Database:**
        - Set up a database (e.g., MySQL) on the server to store sensor data.
    - **Web Interface:**
        - Create a web dashboard using HTML, CSS, and JavaScript to visualize sensor data in real-time.
        - Include visualizations for temperature, humidity, motion detection, door/window status, and power consumption.

Details About Sensors and Microcontroller

1. ESP32 (Centralized Microcontroller)

- **Wi-Fi Range**: Around 50-100 meters indoors, depending on obstacles like walls, and up to 200 meters outdoors.
- **Bluetooth Range**: Approximately 10-50 meters, depending on the Bluetooth class and obstructions.
- **Power**: Typically operates on 3.3V, though a 5V power supply is often used to regulate power for connected peripherals.

2. ESP8266 (For Sensor Nodes)

- **Wi-Fi Range**: Similar to ESP32, around 50-100 meters indoors, and up to 200 meters outdoors.
- **Power**: Operates on 3.3V, with power provided via 5V supplies or battery packs.

3. PIR Sensors (Connected to ESP8266)

- **Model**: Typically HC-SR501 or equivalent.
- **Detection Range**: 5-7 meters (depends on the sensor's sensitivity settings).
- **Field of View**: 100-120 degrees.
- **Power**: Operates on 5V, supplied through the ESP8266 module.

4. Magnetic Reed Switch (For Door/Window Sensors)

- **Range**: Detects proximity of a magnet, typically effective up to 1-2 cm.
- **Power**: Passive sensor, connected to GPIO pins of the ESP8266.

5. Temperature Sensors (DHT22 or DS18B20)

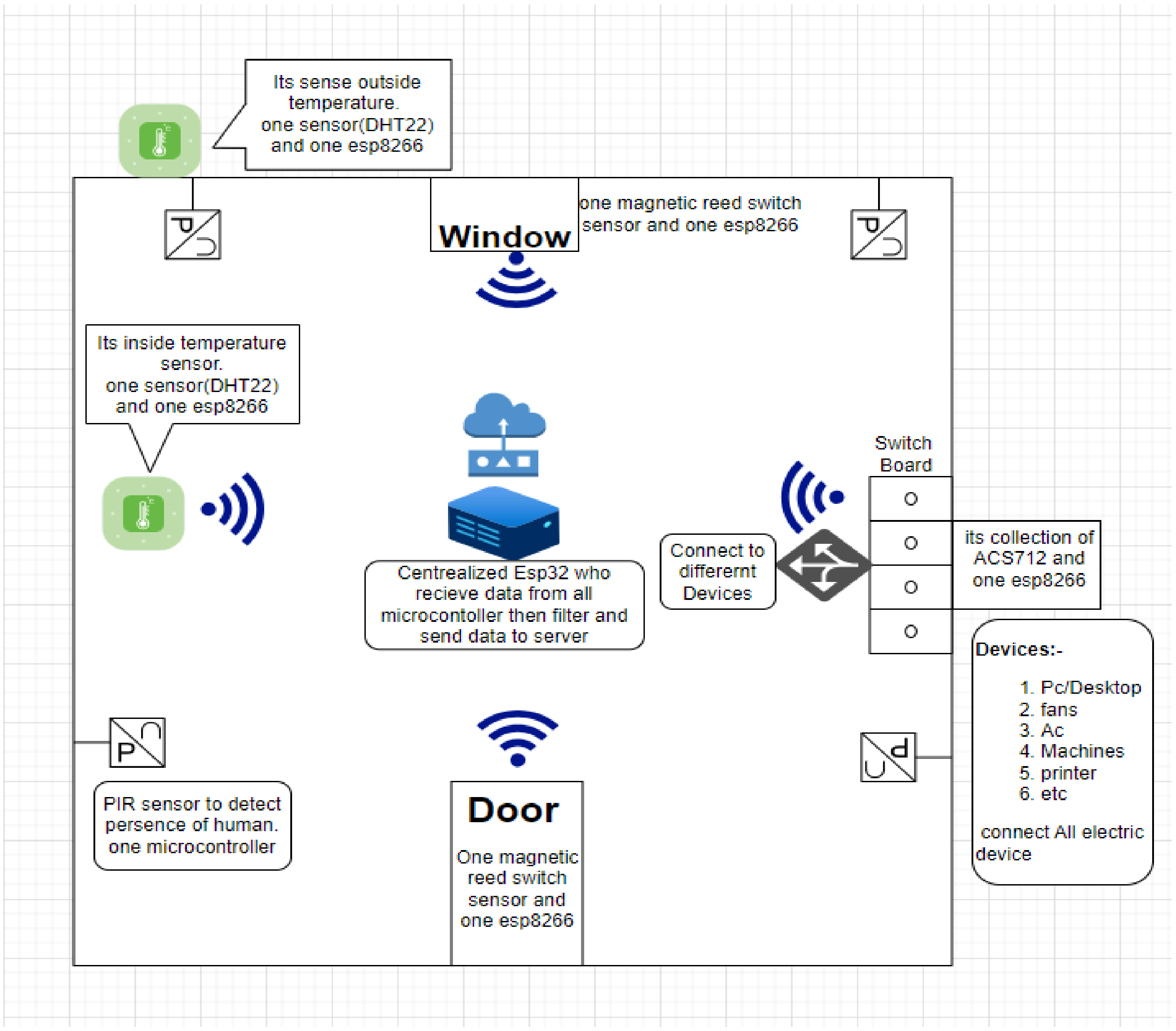
- **DHT22 (Humidity and Temperature Sensor)**:
  - **Temperature Range**: -40°C to 80°C.
  - **Accuracy**: ±0.5°C.
  - **Humidity Range**: 0-100% RH.
  - **Accuracy**: ±2-5% RH.
  - **Power**: 3.3V-5.5V.
- **DS18B20 (Temperature Sensor)**:
  - **Temperature Range**: -55°C to 125°C.
  - **Accuracy**: ±0.5°C.
  - **Power**: 3.0V-5.5V.
  - **Range**: Wired communication can reach up to 100 meters with proper wiring.

6. ACS712 Current Sensor (For Power Consumption)

- **Range**:
  - There are different variants like ACS712-05B, ACS712-20A, and ACS712-30A for measuring 5A, 20A, and 30A currents, respectively.
  - Accuracy: Typically 1.5% error or better.
- **Power**: Operates at 5V, connected to the ESP8266 module via analog input.
- **Range of Current Measurement**:
  - For example, ACS712-05B can measure from -5A to 5A, while the ACS712-30A can measure from -30A to 30A.

Complete setup:-

https://app.diagrams.net/#G15kbNFpSzcfQqqZrMsPqMHlC2x8w7mOGP#%7B%22pageId%22%3A%22uUR8m7Itm5GvlA4v2K2-%22%7D


Its sense outside temperature. one sensor(DHT22) and one esp8266
Window
one magnetic reed switch sensor and one esp8266
Its inside temperature sensor. one sensor(DHT22) and one esp8266
Centrealized Esp32 who recieve data from all microcontoller then filter and send data to server
Switch Board
Connect to differernt Devices
its collection of ACS712 and one esp8266
Devices:-
1. Pc/Desktop
2. fans
3. Ac
4. Machines
5. printer
6. etc
connect All electric device
P
PIR sensor to detect persence of human. one microcontroller
Door
One magnetic reed switch sensor and one esp8266