TITLE:

ENERGY OPTIMIZATION IN SMART BUILDINGS USING IOT AND AI

Objective

To develop a smart system that uses IoT sensors and AI algorithms to monitor and optimize energy consumption in smart buildings, aiming to reduce energy waste, lower costs, and enhance occupant comfort and sustainability.

Overview

This project combines IoT (Internet of Things) and AI (Artificial Intelligence) to intelligently manage energy systems in buildings. IoT devices gather real-time data on temperature, occupancy, lighting, and power usage. AI processes this data to make intelligent decisions—like adjusting HVAC and lighting based on predicted patterns—to optimize energy efficiency.

Implementation

A. Hardware Components

- ESP32 / Raspberry Pi: Acts as the IoT hub.
- Sensors: DHT11/DHT22 (temperature & humidity)
- PIR sensor (occupancy detection)
- LDR (light intensity)
- Energy meters (for real-time consumption)
- Actuators: Smart switches/relays for controlling lights, fans, HVAC.

B. Software Stack

- Data Collection: Python/Node.js scripts to read from sensors.

- Communication Protocol: MQTT over Wi-Fi to publish data.
- Backend: Flask/Django server or Firebase to store and manage data.
- Database: MongoDB/Firebase Realtime DB.
- AI Models: Regression models for consumption prediction.
- Decision Trees or Reinforcement Learning for control decisions.
- Dashboard: Web app (React/HTML+JS) to monitor and control systems.

C. Flow Diagram

- Sensors collect real-time data.
- Data sent to server via MQTT.
- AI processes data for optimization.
- Control signals sent back to actuators.
- Dashboard shows current stats, predictions, and alerts.

CHALLENGES AND SOLUTIONS:

CHALLENGES:

- Unreliable sensor data
- Real time decision making
- Data security
- Power outages
- Complex AI modelling

SOLUTIONS:

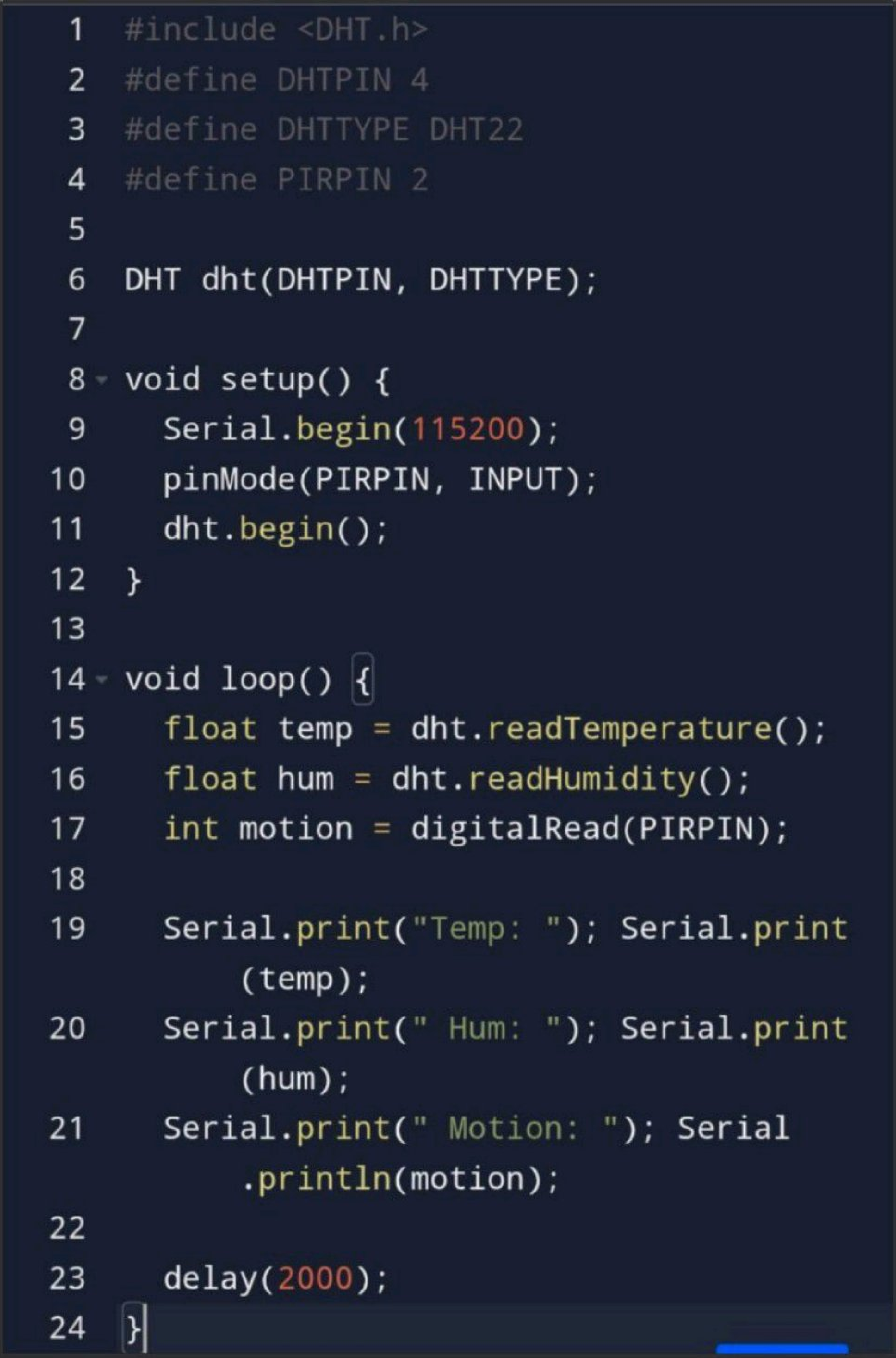
- Use filtering techniques like Kalman filter or moving averages
- Implement edge computing on ESP32 or Pi for quick actions
- Encrypt data in transit, TLS for MQTT, use authentication

- Use backup power for critical devices and cloud data syncing
- Start with simple models, move to deep learning/reinforcement learning

Outcomes

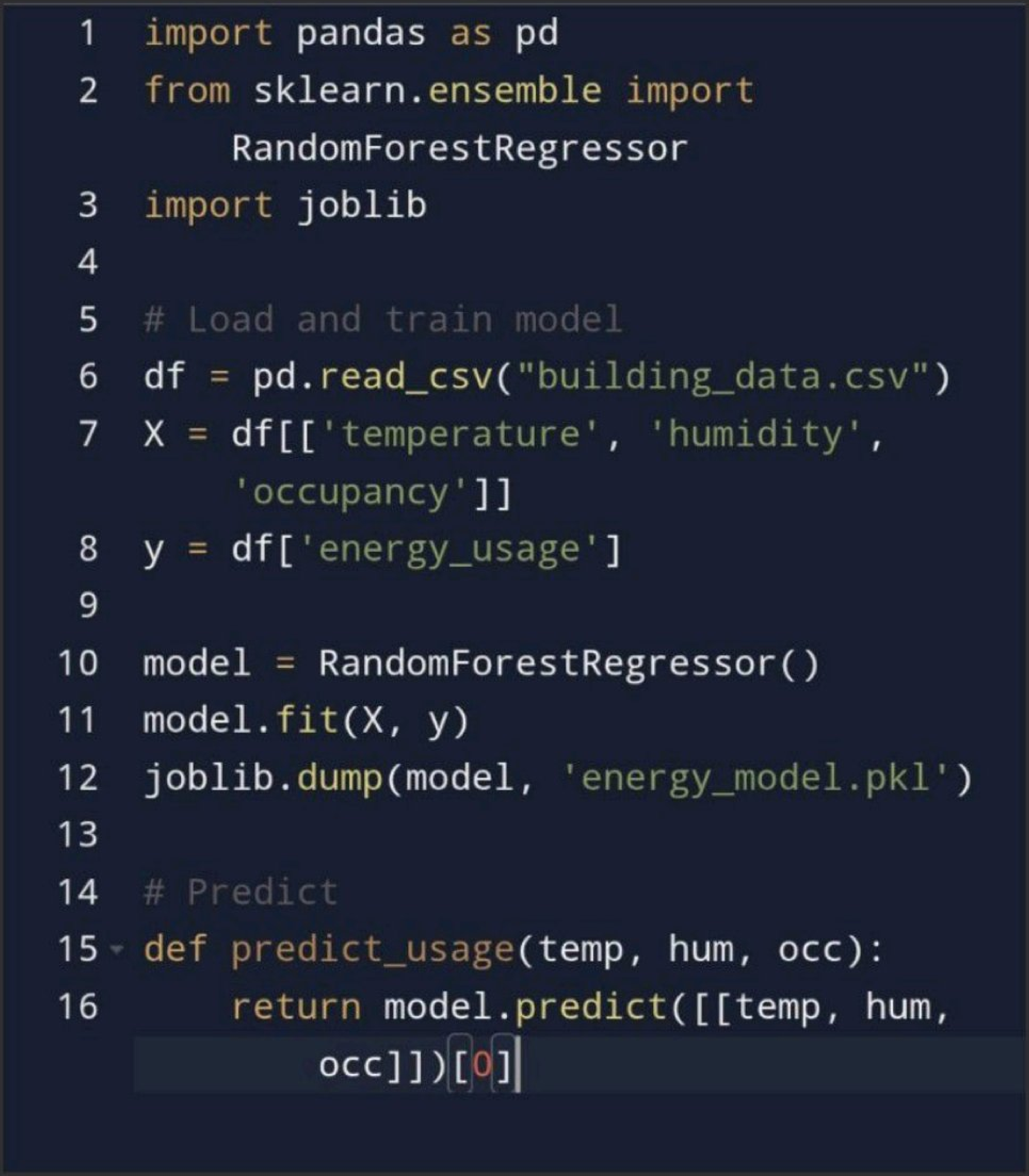
- 20–40% reduction in energy usage.
- Dynamic and adaptive energy control.
- Real-time monitoring and remote control.
- Data-driven insights for facility managers.
- Scalable to various building sizes.

IOT (ESP32 WITH DHT22 AND PIR) – ARDUINO C



```cpp
#include <DHT.h>
#define DHTPIN 4
#define DHTTYPE DHT22
#define PIRPIN 2

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(115200);
  pinMode(PIRPIN, INPUT);
  dht.begin();
}

void loop() {
  float temp = dht.readTemperature();
  float hum = dht.readHumidity();
  int motion = digitalRead(PIRPIN);

  Serial.print("Temp: "); Serial.print(temp);
  Serial.print(" Hum: "); Serial.print(hum);
  Serial.print(" Motion: "); Serial.println(motion);

  delay(2000);
}
```

AI MODEL (PYTHON – PREDICT ENERGY USAGE)



```python
import pandas as pd
from sklearn.ensemble import RandomForestRegressor
import joblib

# Load and train model
df = pd.read_csv("building_data.csv")
X = df[['temperature', 'humidity', 'occupancy']]
y = df['energy_usage']

model = RandomForestRegressor()
model.fit(X, y)
joblib.dump(model, 'energy_model.pkl')

# Predict
def predict_usage(temp, hum, occ):
    return model.predict([[temp, hum, occ]])[0]
```

PROJECT PROGRESS:

- Requirements gathering, architecture planning
- Sensor integration, hardware setup
- Backend setup, data logging

- AI model training and validation
- Automation /control logic and dashboard UI
- Testing, optimization, documentation