مركز حمدان بن راشد
آل مكتوم للموهبة والإبداع
Hamdan Bin Rashid Al Maktoum
Centre for Giftedness and Creativity

تحدي
علوم
المستقبل

(شرح الأكواد لمشروع IOT_Energy_Gaurd لفريق Shingames)

شرح الكود الأول: التحكم والمراقبة عن طريق ESP 32.

الكود يعمل على التحكم ومراقبة استهلاك الطاقة في الأجهزة المنزلية باستخدام لوحة ESP32 المتصلة بمنصة Blynk. يتم استخدام حساسات تيار ACS712 لقياس استهلاك الطاقة، ومستشعر IR للكشف عن وجود الأشخاص في الغرفة. كما يُرسل البيانات إلى **Raspberry Pi** عند تفعيل سويتش معين ليتم استخدامها في تدريب نموذج ذكاء اصطناعي.

تعريف المكتبات اللازمة:

يتم تضمين المكتبات الضرورية لتمكين ESP32 من الاتصال بشبكةWi-Fi ، التعامل مع منصةBlynk ، وقراءة بيانات الاستهلاك من حساسات التيار.ACS712

```
#define BLYNK_TEMPLATE_ID "TMPL4rmwiLFqh"
#define BLYNK_TEMPLATE_NAME "IoT Energy Guard"
#define BLYNK_AUTH_TOKEN "gBtkOsEL-vg-gi3pJNl168LDgj0GM068"
#define BLYNK_PRINT Serial
```

تعريف معرفاتBlynk :

يتم تحديد معرف القالب، اسمه، ورمز المصادقة الذي يتيح الاتصال بمنصةBlynk .

```
#include <WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp32.h>
#include "ACS712.h"
```

إعدادات شبكة الواي فاي:

يتم تحديد اسم الشبكة وكلمة المرور للاتصال بها.

```
char ssid[] = "NCD";   // اسم شبكة الواي فاي
char pass[] = "N123456n";   // كلمة مرور شبكة الواي فاي
```

تعريف المتغيرات والمنافذ:

يتم تحديد المنافذ المتصلة بالمفاتيح وحساس الأشعة تحت الحمراء.

```
const int switchPins[] = {13, 2, 5, 12};
const int switchPin4 = 4;
const int irPin = 25;
```

تعريف حساسات التيار المتردد:

يتم إنشاء كائنات حساسات التيار وتحديد المعايير الخاصة بها.

```
ACS712 currentSensors[] = {
    ACS712(34, 5.0, 1023, 30),
    ACS712(35, 5.0, 1023, 30),
    ACS712(32, 5.0, 1023, 30),
    ACS712(33, 5.0, 1023, 30)
    };
```

منافذ Blynk الافتراضية:

يتم تحديد المنافذ التي ستُستخدم لعرض بيانات التيار وحالة الغرفة.

```
const int gaugePins[] = {V0, V2, V5, V12};
const int statusPin = V1;
const int totalConsumptionPin = V20;
const int predictionPin = V21;
```

إعداد النظام في setup():

هذه الدالة مسؤولة عن تهيئة النظام قبل بدء التشغيل الفعلي، حيث تقوم بعدة مهام تشمل بدء الاتصال التسلسلي لتمكين المراقبة عبر المنفذ التسلسلي، ثم الاتصال بمنصة Blynk باستخدام بيانات المصادقة الخاصة بها. كما تقوم بإعداد المنافذ الرقمية المتصلة بالمفاتيح بوضع INPUT_PULLUP لمنع التداخلات الكهربائية، بالإضافة إلى إعداد حساس الأشعة تحت الحمراء (IR) لمراقبة وجود الأشخاص. وأيضًا، تقوم بضبط حساسات التيار على نقطة المنتصف تلقائيًا لتوفير قراءات دقيقة.

```
void setup() {
    Serial.begin(115200);
    Blynk.begin(auth, ssid, pass, "blynk.cloud", 80);
    delay(1000);
    for (int i = 0; i < 4; i++) {
        pinMode(switchPins[i], INPUT_PULLUP);
    }
    pinMode(switchPin4, INPUT_PULLUP);
    pinMode(irPin, INPUT);


    for (int i = 0; i < 4; i++) {
        currentSensors[i].autoMidPoint();
    }
}
```

تنفيذ الكود في loop():

هذه الدالة تعمل بشكل مستمر وهي مسؤولة عن تشغيل النظام الفعلي، حيث تقوم بعدة مهام رئيسية تتضمن تشغيل منصة Blynk للتفاعل مع الأجهزة الأخرى عبر الإنترنت، وقراءة حالة المفتاح switchPin4 للتحكم في بدء أو إيقاف إرسال البيانات. في حال تم تنشيط جمع البيانات، تقوم الدالة بقراءة بيانات حساس الأشعة تحت الحمراء وإرسالها إلى منصة Blynk لتحديد وجود شخص في الغرفة، ثم قراءة بيانات حساسات التيار الأربعة وإرسالها إلى المنصة، بالإضافة إلى حساب إجمالي استهلاك الطاقة وإرساله إلى Blynk. كما تقوم الدالة بطباعة البيانات المجمعة على المنفذ التسلسلي، وأخيرًا، تستقبل أي بيانات متوقعة من جهاز Raspberry Piالمسؤول عن التنبؤات وتقوم بإرسالها إلى منصة Blynk.

```
void loop() {
    Blynk.run();
    int switchState = digitalRead(switchPin4);
    if (switchState == LOW && lastSwitchState == HIGH) {
        sentA = !sentA;
        startPrintingData = !startPrintingData;
        Serial.println("A");
        delay(300);
    }
```

```
    lastSwitchState = switchState;
    if (startPrintingData) {
        int irState = digitalRead(irPin);
        Blynk.virtualWrite(statusPin, irState == HIGH ? "There is someone in the room" :
"No one in the room");


        String valuesToPrint = "";
        double totalConsumption = 0.0;
        for (int i = 0; i < 4; i++) {
            int switchState = digitalRead(switchPins[i]);
            double current = (switchState == HIGH) ?  currentSensors[i].getCurrentAC() :
0.0;
            Blynk.virtualWrite(gaugePins[i], current);
            totalConsumption += current;
            valuesToPrint += String(current, 3) + ",";
        }
        Blynk.virtualWrite(totalConsumptionPin, totalConsumption);
        valuesToPrint.remove(valuesToPrint.length() - 1);
        Serial.println(valuesToPrint);
        delay(1000);
    }
    if (Serial.available()) {
          String receivedData = Serial.readStringUntil('\n');
          receivedData.trim();
          int firstComma = receivedData.indexOf(',');
          int secondComma = receivedData.indexOf(',', firstComma + 1);

          if (firstComma != -1 && secondComma != -1) {
            double nextDay = receivedData.substring(0, firstComma).toFloat();
            double nextWeek = receivedData.substring(firstComma + 1,
secondComma).toFloat();
            double nextMonth = receivedData.substring(secondComma + 1).toFloat();

            Blynk.virtualWrite(nextDayPin, nextDay);
            Blynk.virtualWrite(nextWeekPin, nextWeek);
            Blynk.virtualWrite(nextMonthPin, nextMonth);
          }
}
```

شرح الكود الثاني: التنبؤ باستخدام خوارزمية SGD

يهدف هذا الكود إلى تطوير نظام يقوم بجمع بيانات الاستهلاك الكهربائي من أربع حساسات عبر اتصال تسلسلي (Serial) باستخدام ESP32. ثم يقوم بتحليل البيانات، تدريب نماذج تنبؤية باستخدام مكتبة *Scikit-learn*، والتنبؤ بقيم استهلاك الكهرباء لليوم التالي، الأسبوع التالي، والشهر التالي. يتم تحديث النماذج بشكل تدريجي (Incremental Training) وإرسال النتائج التنبؤية إلى ESP32 لإجراء مزيد من العمليات أو العرض.

المكتبات المستخدمة:

- **مكتبة (Serial)**: للتعامل مع الاتصال التسلسلي بين Raspberry pi 3 وESP32.
- **مكتبة (numpy)**: لتسهيل التعامل مع القيم العددية والمصفوفات.
- **مكتبة (SGDRegressor)**: من مكتبة sklearn، وهي خوارزمية تستخدم للتعلم التدريجي.
- **مكتبة (Deque)**: لتخزين نافذة بيانات الحساسات بشكل دوري (Sliding Window) مع حجم ثابت.
- **مكتبة (Time)**: لإدارة فترات الانتظار وتوقيت المهام.
- **مكتبة (matplotlib.pyplot)**: لرسم بيانات التوقعات بشكل ديناميكي.
- **مكتبة (sklearn.exceptions.NotFittedError)**: لمعالجة استثناءات محاولة التنبؤ قبل تدريب النموذج.

آلية عمله والخوارزمية المتبعة

الخطوة الأولى: إعداد النماذج وتهيئة البيانات

في بداية الكود، يتم إنشاء أربعة نماذج انحدار تدريجي (SGDRegressor) لكل غرفة على حدة، بالإضافة إلى نموذج خامس يمثل استهلاك المنزل بالكامل. يتم استخدام مكتبة make_pipeline لربط SGDRegressor مع StandardScaler لضمان تطبيع البيانات قبل التدريب. هذه النماذج تتعلم بشكل تدريجي مع كل دفعة جديدة من البيانات.

```
room_models = [make_pipeline(StandardScaler(), SGDRegressor(learning_rate='constant',
eta0=0.0001)) for _ in range(4)]
    house_model = make_pipeline(StandardScaler(), SGDRegressor(learning_rate='constant',
eta0=0.0001))
```

الخطوة الثانية: إعداد الاتصال التسلسلي

إعداد الاتصال التسلسلي مع ESP32 يتم إنشاء اتصال تسلسلي مع ESP32 عبر منفذ USB باستخدام مكتبة .serial ينتظر البرنامج إدخال الحرف "A" من ESP32 لبدء استقبال البيانات.

```
print("Waiting for 'A' from ESP32 to start receiving sensor data...")
while True:
   if ser.in_waiting > 0:
        data = ser.readline().decode('utf-8').strip()
        if data == 'A':
           print("Received 'A', starting data collection...")
           break
```

الخطوة الثالثة: جمع البيانات ومعالجتها

يتم قراءة البيانات بشكل مستمر منESP32 ، والتحقق من صحة البيانات لضمان أنها تحتوي على 4 قيم فقط. يتم تحويل القيم إلى أعداد عشرية (float) لضمان دقة التنبؤات.

```
if ser.in_waiting > 0:
    data = ser.readline().decode('utf-8').strip()
    try:
       sensor_values = list(map(float, data.split(',')))  # Change int to float
       if len(sensor_values) != 4:
           print(f"Invalid data format: {data}")
                       continue
   except ValueError as e:
       print(f"Error processing data: {e}. Data: {data}")
```

الخطوة الرابعة: التدريب التدريجي للنماذج

بعد جمع البيانات، يتم تحديث كل نموذج تدريجيًا باستخدام .partial_fitيتم تدريب كل نموذج على القيم المستقبلة من الغرفة نفسها، بينما يتم تدريب نموذج المنزل على المتوسط العام لجميع الغرف.

```
def update_models(sensor_values, room_models, house_model):
    """Update room and house models with new sensor data."""
    sensor_array = np.array(sensor_values).reshape(1, -1)  # Shape (1, 4)

    for i in range(4):
        X_room = np.array([[sensor_values[i]]])  # Single feature input
        target_room = sensor_values[i]  # Predict itself for now

        # Fit StandardScaler before first training iteration
        if not hasattr(room_models[i].named_steps['sgdregressor'], 'coef_'):
            room_models[i].named_steps['standardscaler'].fit(X_room)

        room_models[i].named_steps['sgdregressor'].partial_fit(X_room, [target_room])

    target_house = np.mean(sensor_values)  # Predict the average of all rooms

    if not hasattr(house_model.named_steps['sgdregressor'], 'coef_'):
        house_model.named_steps['standardscaler'].fit(sensor_array)

    house_model.named_steps['sgdregressor'].partial_fit(sensor_array, [target_house])
```

الخطوة الخامسة: التنبؤ بقيم المستقبل

يتم استخدام نموذج المنزل لإجراء التنبؤات المستقبلية للاستهلاك اليومي، الأسبوعي، والشهري باستخدام معاملات مختلفة.

```
def predict_values(house_model, sensor_values):
    """Predict values for next day, week, and month."""
    X_house = np.array(sensor_values).reshape(1, -1)
    try:
        house_prediction = house_model.predict(X_house)
        next_day_prediction = house_prediction[0] * 1.1 * 12 * 24
        next_week_prediction = house_prediction[0] * 1.5 * 12 * 24 * 7
        next_month_prediction = house_prediction[0] * 2.0 * 12 * 24 * 30
        return next_day_prediction, next_week_prediction, next_month_prediction
    except NotFittedError:
        print("House model not yet trained.")
        return None, None, None
```

الخطوة السادسة: عرض التوقعات بصريًا

يتم رسم التوقعات بشكل ديناميكي باستخدام مكتبة matplotlib، ويتم تحديث الرسم بعد كل دورة تدريب.

```
plt.ion()
    fig, ax = plt.subplots()
    lines = ax.plot([], [], [], [], [], [])
    ax.set_xlabel("Time")
    ax.set_ylabel("Predicted Values")
    ax.legend(["Next Day", "Next Week", "Next Month"])
```

الخطوة السابعة: إرسال التوقعات

إرسال التوقعات إلى ESP32، يتم إرسال التوقعات إلى ESP32 في صيغة نص مفصول بفواصل.

```
   predicted_values = f"{next_day:.2f},{next_week:.2f},{next_month:.2f}"
   ser.write((predicted_values + "\n").encode('utf-8'))
   print("Predicted values sent to ESP32:", predicted_values)
```

ملاحظات عامة

الخوارزمية تتبع نهجًا ديناميكيًا يعتمد على التحديث التدريجي للبيانات والنماذج. يتم الاستفادة من بيانات الحساسات في الوقت الفعلي لتدريب النماذج بشكل تدريجي والتنبؤ بالقيم المستقبلية بناءً على اتجاهات البيانات المجمعة.

كود التحكم والمراقبة من قبل ESP32

كود تدريب AI