Android Telefon Üzerinden Mesaj Gönderimi ve Sistem Çalışma Prensibi (2. ve 3. Projeler İçin)

Bu projede, kullanıcıdan gelen mesajların ESP32'ye iletilmesi için bir Android telefon kullanılmıştır. Bunun için Google Play Store üzerinden indirilebilen "RestClient" adlı uygulama tercih edilmiştir. Bu uygulama sayesinde, ESP32 üzerinde çalışan HTTP sunucusuna manuel veya sesli olarak mesaj gönderilebilmiştir.

Kullanıcı, RestClient uygulamasını açarak HTTP POST isteği oluşturur. ESP32'nin IP adresi ve mesaj alma endpoint'i (örneğin http://192.168.156.32/mesaj) istek URL'si olarak girilir. Mesaj, "text/plain" içeriğiyle gönderilir. Uygulamada herhangi bir ek JSON yapısına ihtiyaç duyulmadan sade bir metin formatı yeterlidir. Bu sayede işlemler basit ve hızlı bir şekilde gerçekleştirilebilir.

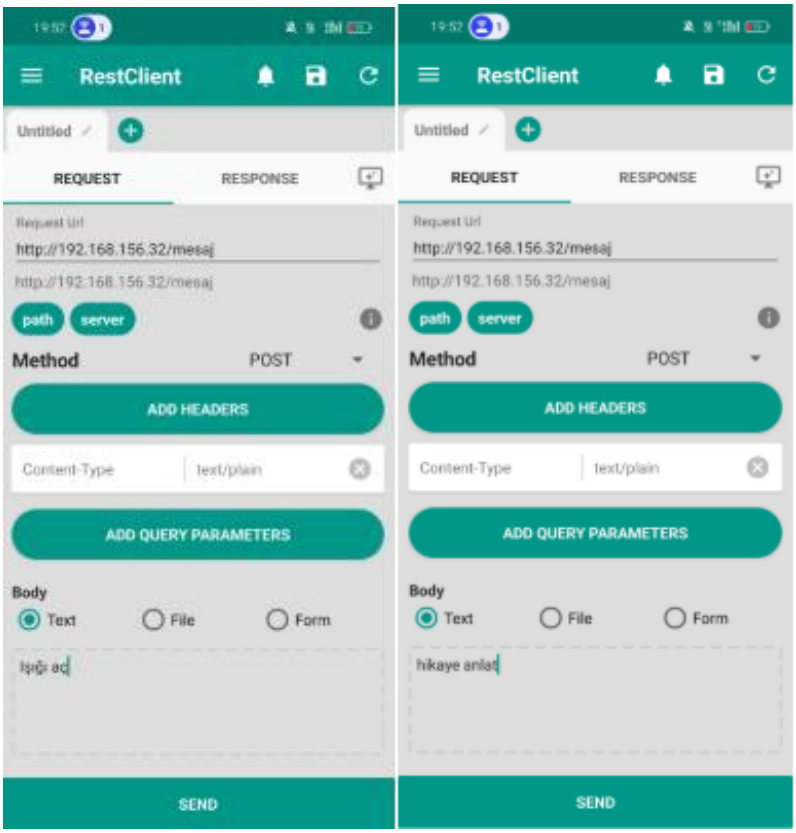
Mesajın manuel olarak yazılması dışında, Android cihazın klavyesinde bulunan mikrofon simgesi kullanılarak Google Asistan yardımıyla sesli giriş de yapılabilmiştir. Google Asistan, kullanıcının sesli komutunu otomatik olarak metne dönüştürmekte ve bu metin doğrudan uygulamanın "Body" (gövde) kısmına yapıştırılabilmektedir. Bu yöntem sayesinde kullanıcı herhangi bir fiziksel klavye girişi olmadan, sadece konuşarak mesaj gönderebilmiştir.

Gönderilen mesaj ESP32'ye ulaştığında sistem şu şekilde çalışmaktadır:

Eğer mesaj içerisinde "ışığı aç" veya "ışığı kapat" gibi önceden belirlenmiş bir komut bulunuyorsa, ESP32 doğrudan LED'i kontrol etmekte ve OLED ekran üzerinden kullanıcıya işlemle ilgili bilgi vermektedir.

Eğer mesaj genel bir metin içeriyorsa, bu metin ESP32 tarafından OpenRouter API'ye gönderilmekte ve gelen yapay zeka yanıtı OLED ekranda satır satır ve harf harf olarak gösterilmektedir.

Bu yöntem sayesinde sistem, hem klasik komut kontrolü hem de yapay zeka tabanlı sohbet veya yanıt gösterimi yapabilen çift amaçlı bir akıllı mikrodenetleyici çözümüne dönüşmüştür. Android cihazın hem kullanıcı arayüzü (RestClient) hem de sesli metin dönüştürücü (Google Asistan) olarak kullanılabilmesi, sistemi daha etkileşimli ve pratik hale getirmiştir.



📄 Proje 1: ESP32 ile Yapay Zeka Yanıt Gösterimi (OLED Üzerinden)

1. Proje Adı

ESP32 ile OpenRouter API Kullanarak Yapay Zeka Yanıtının OLED Ekranda Gösterilmesi

2. Amaç

Bu projenin amacı, ESP32 mikrodenetleyicisi ile internet üzerinden OpenRouter (OpenAI uyumlu) API'sine bağlanarak yapay zekaya bir mesaj gönderip, aldığı yanıtı OLED ekran üzerinde harf harf animasyon şeklinde gösterebilmektir.

3. Gerekli Malzemeler

ESP32 Geliştirme Kartı

SSD1306 OLED Ekran (128x64, I2C)

Bağlantı kabloları

WiFi bağlantısı

Arduino IDE ve gerekli kütüphaneler:

WiFi.h

HTTPClient.h

ArduinoJson.h

Adafruit_GFX.h

Adafruit_SSD1306.h

4. Devre Şeması / Bağlantı Diyagramı

OLED ekran bağlantısı (I2C protokolü):

VCC → ESP32 3.3V

GND → ESP32 GND

SCL → ESP32 GPIO 22

SDA → ESP32 GPIO 21

5. Çalışma Prensibi

ESP32 açıldığında WiFi ağına bağlanır.

Önceden belirlenmiş bir mesaj (örneğin: "Merhaba, nasılsın?") OpenRouter API'sine gönderilir.

API'den alınan yanıt ArduinoJson ile ayrıştırılır.

Yanıttaki Türkçe karakterler sadeleştirilir (ç → c, ğ → g vb.).

Harf harf olacak şekilde OLED ekranda yazdırılır.

Yazı taşarsa ekran temizlenip yazmaya devam edilir.

6. Kod Açıklamaları

sendMessageToGPT(String mesaj): Mesajı API'ye gönderir, yanıtı alır.

yavasYazHarfHarf(String yazi): OLED ekranında yazıyı yavaş yavaş yazdırır.

replaceTurkishChars(String str): Türkçe karakterleri sadeleştirir.

WiFi bağlantısı kurulup ekran durum mesajları verilir (WiFi BAGLANDI! gibi).

Ana mesaj setup() fonksiyonu içerisinde gönderilir ve gösterilir.

7. Karşılaşılan Sorunlar ve Çözümleri

OLED ekran Türkçe karakterleri göstermiyordu Türkçe karakter sadeleştirme fonksiyonu eklendi.

WiFi bağlantısı gecikebiliyordu Bağlantı sırasında kullanıcıya ekranda durum mesajı gösterildi.

API yanıtları JSON hatası verebiliyordu Hatalar kontrol altına alındı ve kullanıcıya bildirim yapıldı.

8. Sonuç

Bu proje sayesinde ESP32 ile hem API entegrasyonu hem de OLED kullanımı öğrenildi. Gerçek zamanlı bir mesajlaşma hissi veren harf harf yazım özelliği ile kullanıcı deneyimi geliştirildi.

9. Geliştirilebilir Yönler

Kullanıcının telefondan mesaj gönderebildiği bir arayüz eklenebilir.

Sesli giriş ve çıkış entegre edilerek konuşan bir yapay zekaya dönüştürülebilir.

Daha büyük ekran veya TFT ekranla zengin arayüzler oluşturulabilir.

Gelen yanıtlar SD karta kaydedilebilir.

📄 Proje 2: ESP32 Üzerinden HTTP POST ile Yapay Zeka Yanıt Gösterimi

1. Proje Adı

ESP32 HTTP Sunucusu ile Kullanıcıdan Alınan Mesajın Yapay Zeka Yanıtının OLED Ekranda Gösterilmesi

2. Amaç

Kullanıcının telefon veya bilgisayardan ESP32'ye mesaj gönderebildiği, gelen bu mesajın OpenRouter API üzerinden GPT-3.5-turbo modeline iletilerek cevabının OLED ekranda gösterildiği bir sistem kurmak.

3. Gerekli Malzemeler

ESP32 Geliştirme Kartı

SSD1306 OLED Ekran (128x64, I2C)

Bağlantı kabloları

WiFi bağlantısı

Arduino IDE ve kütüphaneler:

WiFi.h

WebServer.h

HTTPClient.h

ArduinoJson.h

Adafruit_GFX.h

Adafruit_SSD1306.h

4. Devre Şeması / Bağlantı Diyagramı

OLED ekran bağlantısı (I2C):

VCC → ESP32 3.3V

GND → ESP32 GND

SCL → ESP32 GPIO 22

SDA → ESP32 GPIO 21

5. Çalışma Prensibi

ESP32 açıldığında WiFi ağına bağlanır.

HTTP sunucusu başlatılır ve /mesaj endpoint'i dinlenir.

Kullanıcı, ağ üzerinden (örneğin Postman ya da web arayüzüyle) ESP32'nin IP adresine bir POST isteği gönderir.

Mesaj alınır ve OpenRouter API'ye iletilir.

Gelen yanıt sadeleştirilir (Türkçe karakterler sadeleştirilir).

OLED ekran harf harf yanıtı yazar.

6. Kod Açıklamaları

server.on("/mesaj", HTTP_POST, handlePostMessage);: HTTP POST endpoint tanımı.

handlePostMessage(): Gelen mesajı işler, API'ye gönderir ve ekranda gösterir.

sendMessageToGPT(): GPT modeline mesajı gönderir ve yanıtı alır.

yavasYaz(): OLED ekran üzerinde yazıyı harf harf ve satır satır olarak gösterir.

replaceTurkishChars(): Türkçe karakterleri sadeleştirir.

7. Karşılaşılan Sorunlar ve Çözümleri

HTTP POST üzerinden mesaj alınırken veri eksik gelebiliyordu, hasArg("plain") ile veri kontrolü yapıldı.

Uzun yanıtlar OLED ekrana sığmıyordu, Ekran satır satır ve sayfa sayfa yenilenerek yazı bölündü.

Türkçe karakterler bozuk çıkıyordu, replaceTurkishChars() fonksiyonu eklendi.

8. Sonuç

Bu proje sayesinde ESP32'nin WiFi sunucusu yetenekleri kullanılarak dışarıdan mesaj alma, API ile yapay zeka yanıtı alma ve ekran üzerinde yazı yazdırma başarıyla entegre edildi. Kullanıcı etkileşimi önemli ölçüde artırıldı.

9. Geliştirilebilir Yönler

HTML arayüzü oluşturularak kullanıcı mesajını doğrudan bir form ile gönderebilir.

Yanıt sesli okunarak sesli yapay zeka yardımcısı haline getirilebilir.

OLED yerine TFT ekran kullanılarak görsel zenginlik artırılabilir.

Güvenlik açısından IP ve mesaj sınırlandırmaları eklenebilir.

📄 Proje 3: ESP32 Üzerinden HTTP POST ile Mesaj Alma, Yapay Zeka Yanıtı Gösterme ve LED Kontrolü

1. Amaç

ESP32 üzerinde HTTP sunucusu açarak dışarıdan gelen metin mesajlarını analiz etmek, belirli komutlara göre (ör. ışık aç/kapat) LED kontrolü yapmak ve diğer mesajları OpenRouter API üzerinden GPT modeline gönderip gelen cevabı OLED ekranda göstermek.

2. Özellikler

WiFi ağına bağlanma

HTTP POST üzerinden mesaj alma

Komutları tanıma ve LED’i açıp kapatma

Diğer mesajları yapay zeka API’sine iletme

Gelen yanıtı OLED ekranda satır satır gösterme

Türkçe karakterleri OLED için sadeleştirme

Mesajdaki Türkçe karakterleri doğru algılamak için küçük harfe çevirme işlemi

3. Kullanılan Donanımlar

ESP32

SSD1306 OLED (I2C)

ESP32 üzerindeki dahili LED ya da harici LED (GPIO 5)

4. Bağlantılar

OLED ekran I2C bağlantısı (GPIO 21 SDA, GPIO 22 SCL)

LED bağlıysa GPIO 5 (LED_PIN)

5. Kod ve İşlevler

handlePostMessage(): Gelen mesajı kontrol eder.

“ışığı aç” varsa LED yakar ve OLED mesaj gösterir.

“ışığı kapat” varsa LED kapatır ve OLED mesaj gösterir.

Diğer durumlarda mesajı GPT API’ye gönderir, yanıtı OLED’de gösterir.

sendMessageToGPT(): Mesajı OpenRouter API’ye gönderir ve yanıtı döner.

yavasYaz(): OLED’de yanıtı düzgün satırlara bölerek yavaşça yazdırır.

toLowerTurkish(): Gelen metni Türkçe karakterlere dikkat ederek küçük harfe çevirir.

replaceTurkishChars(): OLED uyumlu hale getirmek için Türkçe karakterleri sadeleştirir.

6. Çalışma Süreci

ESP32 açılır, WiFi’ye bağlanır.

HTTP sunucu başlar ve /mesaj adresi dinlenir.

Mesaj geldiğinde önce küçük harfe çevirilir.

Mesajda “ışığı aç” veya “ışığı kapat” komutları aranır.

Eğer komut varsa LED kontrol edilir ve OLED'de mesaj gösterilir.

Değilse mesaj GPT API'ye gönderilir, yanıt OLED'de gösterilir.

7. Geliştirme Fikirleri

Sesli komutları algılayarak LED ve diğer cihazları kontrol etme

Farklı cihazlar için daha fazla komut ekleme

HTTPS desteği ve güvenlik önlemleri ekleme

Kullanıcı dostu web arayüzü yaparak komut gönderme kolaylığı sağlama

OLED yerine TFT ekran ya da sesli çıktı ekleme

Sonuç

Bu proje, IoT cihaz kontrolü ve yapay zeka entegrasyonunu birleştirerek hem sesli/metin komutlarla cihaz kontrolü hem de gelişmiş sohbet yeteneği sunuyor. Projenin yetenekleri genişletilebilir ve ev otomasyon sistemlerinde veya kişisel asistan uygulamalarında kullanılabilir.

Kaynak Kodlar:

Proje1:

```
#include <WiFi.h>  // ESP32 WiFi kütüphanesi

// Wi-Fi bilgilerini buraya yaz

const char* ssid = "karadas_internet_hayrati";        // Wi-Fi SSID (ağ adı)

const char* password = "mkmk1249";   // Wi-Fi şifresi

void setup() {

  Serial.begin(115200); // Seri haberleşmeyi başlat

  WiFi.begin(ssid, password); // Wi-Fi ağına bağlanmayı dene

  Serial.print("Wi-Fi'ye bağlanıyor");
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("✅ Bağlantı başarılı!");
  Serial.print("💻 IP Adresi: ");
  Serial.println(WiFi.localIP()); // IP adresini yazdır
}
void loop() {
  // Wi-Fi bağlantısı kontrolü
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("❌ Bağlantı koptu! Yeniden bağlanılıyor...");
    WiFi.begin(ssid, password);
  }
  delay(10000); // Her 10 saniyede bir kontrol et
}
```

Proje2:

```
#include <WiFi.h>
#include <WebServer.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
// OLED ayarları
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// WiFi bilgileri
const char* ssid = "karadas_internet_hayrati";
const char* password = "mkmk1249";
// OpenRouter API bilgileri
const char* openai_api_key = "Bearer sk-or-v1-
e11cc2e302f10aa388524554df07e1d968e9545974aec5ebf8ca1733f829e719";
const char* api_url = "https://openrouter.ai/api/v1/chat/completions";
// HTTP sunucu port 80
WebServer server(80);
// Yazı pozisyonu OLED için
int currentLine = 0;
String ekranda = "";
// Fonksiyon prototipleri
String sendMessageToGPT(String mesaj);
void yavasYaz(String yazi);
void handlePostMessage();
String replaceTurkishChars(String str);
void setup() {
  Serial.begin(115200);
  // OLED başlat
  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
```

```
  Serial.println("OLED başlatılamadı");
  while (true);
}
display.clearDisplay();
display.setTextSize(1);
display.setTextColor(SSD1306_WHITE);
display.setCursor(0, 0);
display.println("WiFi baglaniyor...");
display.display();
// WiFi'ye bağlan
WiFi.begin(ssid, password);
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println();
Serial.print("WiFi baglandi! IP: ");
Serial.println(WiFi.localIP());
display.clearDisplay();
display.setCursor(0, 0);
display.println("WiFi BAGLANDI!");
display.println(WiFi.localIP().toString());
display.display();
// HTTP POST endpoint ayarla
server.on("/mesaj", HTTP_POST, handlePostMessage);
```

```
  // Sunucuyu başlat
  server.begin();
  Serial.println("HTTP Sunucu başlatıldı");
}
void loop() {
  server.handleClient();
}
// === Türkçe karakterleri sadeleştirme ===
String replaceTurkishChars(String str) {
  str.replace("ç", "c");  str.replace("Ç", "C");
  str.replace("ğ", "g");  str.replace("Ğ", "G");
  str.replace("ı", "i");  str.replace("İ", "I");
  str.replace("ö", "o");  str.replace("Ö", "O");
  str.replace("ş", "s");  str.replace("Ş", "S");
  str.replace("ü", "u");  str.replace("Ü", "U");
  return str;
}
// === POST isteği ile gelen mesajı işleyen fonksiyon ===
void handlePostMessage() {
  if (server.hasArg("plain")) {
    String mesaj = server.arg("plain");
    Serial.println("Gelen mesaj: " + mesaj);
    String yanit = sendMessageToGPT(mesaj);
    yavasYaz(yanit);
    server.send(200, "text/plain", "Mesaj alındı");
```

```
  } else {
    server.send(400, "text/plain", "Hata: Mesaj yok");
  }
}
// === OpenRouter API'ye mesaj gönderip cevap alan fonksiyon ===
String sendMessageToGPT(String mesaj) {
  HTTPClient http;
  http.begin(api_url);
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Authorization", openai_api_key);
  String jsonData = "{\"model\":\"gpt-3.5-turbo\",\"messages\":[{\"role\":\"user\",\"content\":\"" + mesaj + "\"}]}";
  int httpResponseCode = http.POST(jsonData);
  Serial.println("HTTP kodu: " + String(httpResponseCode));
  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Yanıt:\n" + response);
    DynamicJsonDocument doc(4096);
    DeserializationError error = deserializeJson(doc, response);
    if (error) {
      Serial.println("JSON hatası: ");
      Serial.println(error.c_str());
      return "JSON hatasi!";
    }
    String gptYaniti = doc["choices"][0]["message"]["content"].as<String>();
    return gptYaniti;
```

```
  } else {
    Serial.println("HTTP hatasi: " + http.errorToString(httpResponseCode));
    return "API hatasi!";
  }
}
// === OLED ekranda yavaş yazdırma fonksiyonu ===
void yavasYaz(String yazi) {
  yazi = replaceTurkishChars(yazi);  // Türkçeyi sadeleştir
  display.clearDisplay();
  display.setCursor(0, 0);
  currentLine = 0;
  ekranda = "";
  for (int i = 0; i < yazi.length(); i++) {
    ekranda += yazi[i];
    String satir = ekranda.substring(currentLine * 21, (currentLine + 1) * 21);
    display.setCursor(0, currentLine * 8);
    display.print(satir);
    display.display();
    delay(50);
    if (satir.length() >= 21 || i == yazi.length() - 1) {
      currentLine++;
      if (currentLine >= 7) {
        delay(1000); // Biraz bekle
        display.clearDisplay();
        currentLine = 0;
```

```
      ekranda = "";
    }
  }
 }
}
```

Proje3:

```
#include <WiFi.h>
#include <WebServer.h>
#include <HTTPClient.h>
#include <ArduinoJson.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
// OLED ayarları
#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET    -1
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
// WiFi bilgileri
const char* ssid = "karadas_internet_hayrati";
const char* password = "mkmk1249";
// OpenRouter API bilgileri
const char* openai_api_key = "Bearer sk-or-v1-
e11cc2e302f10aa388524554df07e1d968e9545974aec5ebf8ca1733f829e719";
```

```
const char* api_url = "https://openrouter.ai/api/v1/chat/completions";

// HTTP sunucu port 80

WebServer server(80);

// LED pini

#define LED_PIN 5

// Fonksiyon prototipleri

String sendMessageToGPT(String mesaj);

void yavasYaz(String yazi);

void handlePostMessage();

String toLowerTurkish(String str);

String replaceTurkishChars(String str);

void setup() {

 Serial.begin(115200);

 if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {

  Serial.println("OLED başlatılamadı");

  while (true);

 }

 display.clearDisplay();

 display.setTextSize(1);

 display.setTextColor(SSD1306_WHITE);

 display.setCursor(0, 0);

 display.println("WiFi baglaniyor...");

 display.display();

 WiFi.begin(ssid, password);

 while (WiFi.status() != WL_CONNECTED) {
```

```
    delay(500);
    Serial.print(".");
  }
  Serial.println();
  Serial.print("WiFi baglandi! IP: ");
  Serial.println(WiFi.localIP());
  display.clearDisplay();
  display.setCursor(0, 0);
  display.println("WiFi BAGLANDI!");
  display.println(WiFi.localIP().toString());
  display.display();
  pinMode(LED_PIN, OUTPUT);
  digitalWrite(LED_PIN, LOW);
  server.on("/mesaj", HTTP_POST, handlePostMessage);
  server.begin();
  Serial.println("HTTP Sunucu başlatıldı");
}
void loop() {
  server.handleClient();
}
void handlePostMessage() {
  if (server.hasArg("plain")) {
    String mesaj = server.arg("plain");
    mesaj = toLowerTurkish(mesaj);
    Serial.println("Gelen mesaj: " + mesaj);
```

```
    if (mesaj.indexOf("ışığı aç") >= 0) {
      digitalWrite(LED_PIN, HIGH);
      yavasYaz("Isigi aciyorum");
      server.send(200, "text/plain", "Isik acildi");
    } else if (mesaj.indexOf("ışığı kapat") >= 0) {
      digitalWrite(LED_PIN, LOW);
      yavasYaz("Isigi kapatiyorum");
      server.send(200, "text/plain", "Isik kapandi");
    } else {
      String yanit = sendMessageToGPT(mesaj);
      yavasYaz(yanit);
      server.send(200, "text/plain", "Mesaj alındı ve yanıtlandı");
    }
  } else {
    server.send(400, "text/plain", "Hata: Mesaj yok");
  }
}
String sendMessageToGPT(String mesaj) {
  HTTPClient http;
  http.begin(api_url);
  http.addHeader("Content-Type", "application/json");
  http.addHeader("Authorization", openai_api_key);
  String jsonData = "{\"model\":\"gpt-3.5-turbo\",\"messages\":[{\"role\":\"user\",\"content\":\"" +
mesaj + "\"}]}";
  int httpResponseCode = http.POST(jsonData);
  Serial.println("HTTP kodu: " + String(httpResponseCode));
```

```
  if (httpResponseCode > 0) {
    String response = http.getString();
    Serial.println("Yanıt:\n" + response);
    DynamicJsonDocument doc(4096);
    DeserializationError error = deserializeJson(doc, response);
    if (error) {
      Serial.println("JSON hatası: ");
      Serial.println(error.c_str());
      return "JSON hatasi!";
    }
    String gptYaniti = doc["choices"][0]["message"]["content"].as<String>();
    return gptYaniti;
  } else {
    Serial.println("HTTP hatasi: " + http.errorToString(httpResponseCode));
    return "API hatasi!";
  }
}
// OLED'de düzgün satırlarla yazı yazma
void yavasYaz(String yazi) {
  yazi = replaceTurkishChars(yazi); // Türkçe sadeleştirme
  String ekranda = "";
  int currentLine = 0;
  for (int i = 0; i < yazi.length(); i++) {
    ekranda += yazi[i];
    display.clearDisplay();
```

```
    // Her satır için ayrı yaz
    for (int line = 0; line <= currentLine; line++) {
      int startIdx = line * 21;
      int endIdx = startIdx + 21;
      if (endIdx > ekranda.length()) endIdx = ekranda.length();
      display.setCursor(0, line * 8);
      display.print(ekranda.substring(startIdx, endIdx));
    }
    display.display();
    delay(50);
    // Yeni satıra geç
    if ((ekranda.length() % 21) == 0 && i != 0) {
      currentLine++;
    }
    // 7 satırdan sonra sıfırla
    if (currentLine >= 7) {
      ekranda = "";
      currentLine = 0;
    }
  }
}
// Türkçe küçük harf düzeltme
String toLowerTurkish(String str) {
  str.replace("İ", "i");
  str.replace("I", "ı");
```

```
  str.replace("Ş", "ş");
  str.replace("Ğ", "ğ");
  str.replace("Ü", "ü");
  str.replace("Ö", "ö");
  str.replace("Ç", "ç");
  str.toLowerCase();
  return str;
}
// OLED uyumlu sade Türkçe karakterler
String replaceTurkishChars(String str) {
  str.replace("ç", "c");
  str.replace("ğ", "g");
  str.replace("ş", "s");
  str.replace("ı", "i");
  str.replace("ö", "o");
  str.replace("ü", "u");
  str.replace("Ç", "C");
  str.replace("Ğ", "G");
  str.replace("Ş", "S");
  str.replace("İ", "I");
  str.replace("Ö", "O");
  str.replace("Ü", "U");
  return str;
}
```