Sakarya Üniversitesi

Bilgisayar ve Bilişim Bilimleri

Bilgisayar Mühendisliği Bölümü

Nesnelerin İnterneti ve Uygulamaları

2024-2025 Güz Dönemi

Proje Adı: Yapay Zekâ Tabanlı Sesli Asistan

Hazırlayanlar

Ali Kerem Kol/ B221210042

Muhammed Baha Bakan/ B221210050

Ders Grubu: 1/A

2. İçindekiler

1. Kapak Sayfası
2. İçindekiler
3. Giriş
4. Literatür Araştırması
5. Donanım ve Yazılım Tasarımı
    - 5.1. Donanım Bileşenleri
    - 5.2. Donanım Tasarımı
    - 5.3. Cihaz Resmi
    - 5.4. Yazılım Yapısı
    - 5.5. UML Diyagramı
6. Çalışma Süreci
    - 6.1. Sistem Akışı
    - 6.2. API Kullanımı
    - 6.3. Kullanıcı Deneyimi
7. Maliyet Analizi
8. IoT Uygulaması için İş Fikri
9. Büyük Veri Kullanımı
10. Sonuç ve Öneriler

---

3. Giriş

IoT (Nesnelerin İnterneti) ve yapay zekâ, modern teknolojinin en kritik iki bileşeni olarak öne çıkmaktadır. Bu iki teknoloji, kullanıcıların cihazlarla daha etkili ve sezgisel bir şekilde etkileşim kurmasını sağlar. Projemiz, IoT teknolojisi ile yapay zekâ tabanlı bir sesli etkileşim cihazı geliştirerek, kullanıcılara doğal dil kullanarak kontrol imkânı sunmayı amaçlamaktadır. Bu çalışmanın sonucunda hem kullanıcı deneyimi gelişecek hem de IoT ve yapay zekâ entegrasyonu konusunda önemli bir adım atılacaktır.

---

4. Literatür Araştırması

Sesli Etkileşim ve Yapay Zekâ

Sesle etkileşimli cihazlar, insan-makine etkileşimini daha sezgisel hale getirir. Bu cihazlar, doğal dil işleme ve ses tanıma algoritmalarının entegrasyonuyla, kullanıcı komutlarını algılayıp yanıt verebilir. Amazon Alexa, Google Assistant gibi örnekler, bu alanda başarılı uygulamalardır.

Kullanılan Teknolojiler

1. **Speech-to-Text (STT):** Kullanıcı sesini metne çevirir.

    - Microsoft AZURE Speech-To-Text API'si kullanıldı.
2. **Text-to-Speech (TTS):** Metni sese çevirerek kullanıcıya geri döner.
    - Microsoft AZURE Text-To-Speech API'si kullanıldı.
3. **Yapay Zekâ:** Kullanıcının sorusunu yanıtlayıp cevabı kullanıcıya geri döner.
    - Google Gemini API'si kullanıldı.
4. **Server:** API bağlantıları ve cevapları için yönetmek için.
    - PHP server kullanıldı.

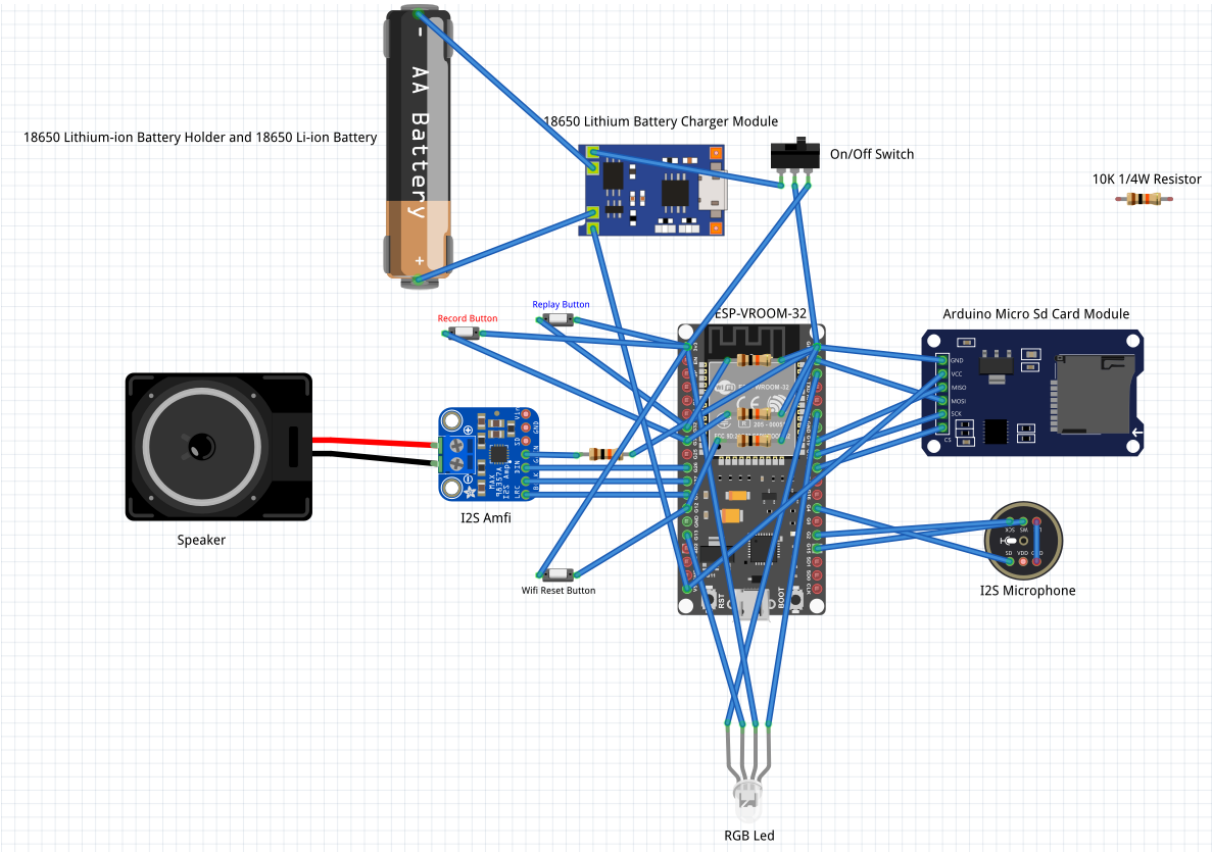
---

5. Donanım ve Yazılım Tasarımı

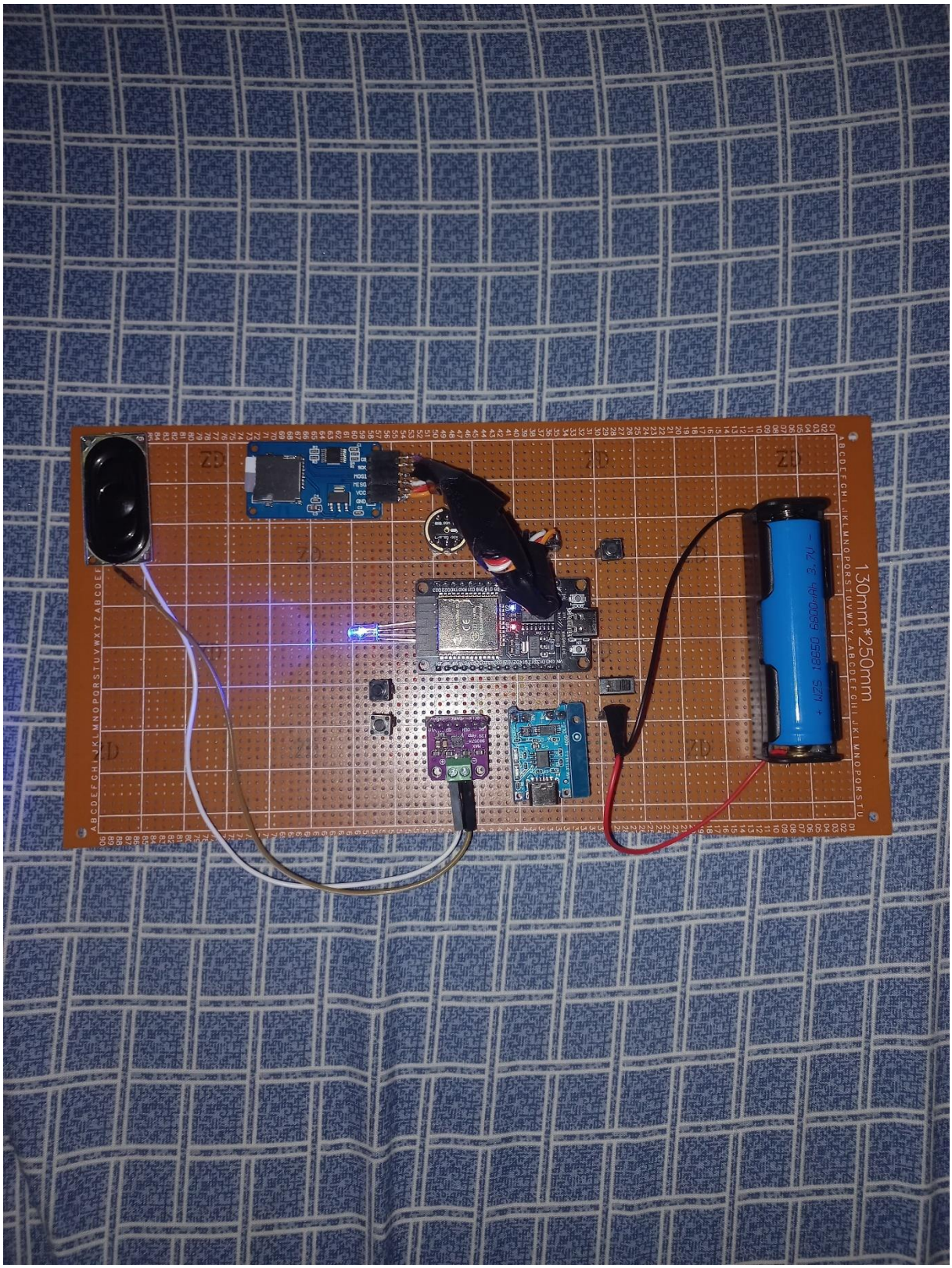
5.1. Donanım Bileşenleri

- **ESP-VROOM-32:** IoT bağlantısı ve temel işlemci birimi.
- **I2S Mikrofon:** Ses girdisi alır.
- **SD Kart Modülü:** Veri depolama birimi.
- **Hoparlör ve Amplifikatör:** Ses çıkışı sağlar.
- **Lityum Pil ve TP4056 Şarj Modülü:** Cihazın enerjisini sağlar.

5.2. Donanım Tasarımı



5.3. Cihaz Resmi



5.4. Yazılım Yapısı

- **Programlama Dili ve Platform:** Cihazın kaynak kodu C++ ile yazılıp Arduino IDE de derlenmiştir. Server için PHP kullanılmıştır.

- Ardunio Kodları:

Esp32-sesli-asistan.ino:

```ino
#define REPLAY_PIN 32

extern const char* yanitFilePath;

#define RED_LED 21
#define GREEN_LED 13
#define BLUE_LED 22

void setup() {
  Serial.begin(115200);
  pinMode(REPLAY_PIN, INPUT_PULLUP);

  pinMode(RED_LED, OUTPUT);
  pinMode(GREEN_LED, OUTPUT);
  pinMode(BLUE_LED, OUTPUT);

  renkAyarla(255,0,0);

  wifiSetup();
  micSetup();
  spkSetup();
}

void renkAyarla(int kirmizi, int yesil, int mavi){
  kirmizi = 255-kirmizi;
  yesil = 255-yesil;
  mavi = 255-mavi;
  analogWrite(RED_LED, kirmizi);
  analogWrite(GREEN_LED, yesil);
  analogWrite(BLUE_LED, mavi);
}

void loop() {
  wifiLoop();
  micLoop();

  if (digitalRead(REPLAY_PIN) == HIGH) { // Butona basıldıysa
    delay(200); // debounce için kısa bekleme
    playAudio(yanitFilePath);
  }
}
```

Mictoserver.ino:

```ino
#include <WiFi.h>
#include <HTTPClient.h>
#include <driver/i2s.h>
#include <SD.h>
#include <FS.h>

// Mikrofon ve buton pinleri
#define I2S_WS 15
#define I2S_SD 4
#define I2S_SCK 2
#define BUTTON_PIN 33

// SD kart pinleri
#define SD_CS 5

extern String sunucuIP;

// Server URL
String serverURL;

// Değişkenler
bool recording = false;
File audioFile;

void micSetup() {

  pinMode(BUTTON_PIN, INPUT_PULLUP);

  pinMode(SD_CS, OUTPUT);

  if (!SD.begin(SD_CS)) {
    Serial.println("SD kart başlatılamadı!");
    renkAyarla(255,0,0);
    while (true);
  }
  Serial.println("SD kart hazır.");

  // SD kart başlat
  if (!SD.begin(SD_CS)) {
    Serial.println("SD kart başlatılamadı!");
    while (true);
  }
  Serial.println("SD kart hazır.");

  // I2S yapılandırma
  i2s_config_t i2s_config = {
    mode = (i2s_mode_t)(I2S_MODE_MASTER | I2S_MODE_RX),
```

```ino
    .sample_rate = 16000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_ONLY_LEFT,
    .communication_format = (i2s_comm_format_t)(I2S_COMM_FORMAT_I2S | I2S_COMM_FORMAT_I2S_MSB),
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = false,
    .tx_desc_auto_clear = true,
    .fixed_mclk = 0
  };

  i2s_pin_config_t pin_config = {
    .bck_io_num = I2S_SCK,
    .ws_io_num = I2S_WS,
    .data_out_num = I2S_PIN_NO_CHANGE,
    .data_in_num = I2S_SD
  };

  // I2S başlat
  i2s_driver_install(I2S_NUM_0, &i2s_config, 0, NULL);
  i2s_set_pin(I2S_NUM_0, &pin_config);
  i2s_zero_dma_buffer(I2S_NUM_0);
}

void micLoop() {
  int buttonState = digitalRead(BUTTON_PIN);

  if (buttonState == HIGH && !recording) {
    // Kayıt başla
    Serial.println("Recording started...");
    recording = true;
    audioFile = SD.open("/audio.raw", FILE_WRITE);
    if (!audioFile) {
      Serial.println("SD kartta dosya açılamadı!");
      recording = false;
    }
  } else if (buttonState == LOW && recording) {
    // Kayıt durdur ve server'a gönder
    Serial.println("Recording stopped, sending data...");
    recording = false;
    audioFile.close();
    sendAudioData();
    Serial.println(serverURL);
  }

  if (recording) {
```

```ino
    readAndSaveChunk();
  }
}

void readAndSaveChunk() {
  renkAyarla(0,255,0);
  const int bufferSize = 1024;
  int32_t buffer[bufferSize];
  size_t bytesRead;

  i2s_read(I2S_NUM_0, (void*)buffer, bufferSize, &bytesRead, portMAX_DELAY);
  audioFile.write((uint8_t*)buffer, bytesRead);
}

void sendAudioData() {
  serverURL = "http://" + sunucuIP + "/esp32-phpserver/upload.php";
  Serial.println("Server URL: ");
  Serial.println(serverURL);

  File file = SD.open("/audio.raw", FILE_READ);
  if (!file) {
    Serial.println("Dosya açılamadı!");
    return;
  }

  WiFiClient client;
  HTTPClient http;
  http.begin(client, serverURL);
  http.addHeader("Content-Type", "application/octet-stream");

  size_t fileSize = file.size();
  Serial.printf("Gönderilecek dosya boyutu: %d bytes\n", fileSize);

  int httpResponseCode = http.sendRequest("POST", &file, fileSize);

  if (httpResponseCode > 0) {
    Serial.printf("HTTP Response code: %d\n", httpResponseCode);
  } else {
    Serial.printf("Error code: %s\n", http.errorToString(httpResponseCode).c_str());
  }

  http.end();
  file.close();
  SD.remove("/audio.raw"); // Dosyayı gönderildikten sonra sil

  delay(1000);
  renkAyarla(0,0,255);
  return;
```

Sdcard.ino:

```
#include <SdFat.h>

File audioFile;

void startRecording() {
  String fileName = "/audio.wav";
  audioFile = SD.open(fileName, FILE_WRITE);
  if (!audioFile) {
    Serial.println("SD karta yazma hatası!");
    return;
  }
  Serial.println("Kayıt başladı: " + fileName);
}

void stopRecording() {
  if (audioFile) {
    audioFile.close();
    Serial.println("Kayıt durduruldu ve dosya kapatıldı.");
  }
}

void writeToSD(int16_t* buffer, size_t size) {
  if (audioFile) {
    audioFile.write((uint8_t*)buffer, size);
  }
}
```

serverToSpeak:

```
#include <SD.h>
#include <HTTPClient.h>
#include <SdFat.h>
#include <SPI.h>
#include <driver/i2s.h>

// Sunucu bilgileri
const char* yanitFilePath = "/yanit.wav"; // SD kart üzerindeki dosya yolu
extern String sunucuIP;

String downloadURL;

// SD kart pinleri
#define SD_CS 5

// I2S pinleri
#define I2S_LRC 14  // LRC pini
#define I2S_BCLK 27 // BCLK pini
#define I2S_DIN 26  // DIN pini

void sphSetup() {
  pinMode(SD_CS, OUTPUT);

  if (!SD.begin(SD_CS)) {
    Serial.println("SD kart başlatılamadı!");
    while (true);
  }
  Serial.println("SD kart hazır.");

  // I2S ayarları
  i2s_config_t i2s_config = {
    .mode = i2s_mode_t(I2S_MODE_MASTER | I2S_MODE_TX),
    .sample_rate = 16000,
    .bits_per_sample = I2S_BITS_PER_SAMPLE_16BIT,
    .channel_format = I2S_CHANNEL_FMT_RIGHT_LEFT,
    .communication_format = I2S_COMM_FORMAT_I2S_MSB,
    .intr_alloc_flags = ESP_INTR_FLAG_LEVEL1,
    .dma_buf_count = 8,
    .dma_buf_len = 64,
    .use_apll = false,
    .tx_desc_auto_clear = true,
    .fixed_mclk = 0
  };

  i2s_pin_config_t pin_config = {
    .bck_io_num = I2S_BCLK,
    .ws_io_num = I2S_LRC,
    .data_out_num = I2S_DIN,
    .data_in_num = I2S_PIN_NO_CHANGE
  };

  // I2S başlat
  i2s_driver_install(I2S_NUM_1, &i2s_config, 0, NULL);
  i2s_set_pin(I2S_NUM_1, &pin_config);
  i2s_set_clk(I2S_NUM_1, 16000, I2S_BITS_PER_SAMPLE_16BIT, I2S_CHANNEL_MONO);
}

void sphRun() {
  if (downloadResponse()) {
    playAudio(yanitFilePath);
  }
}

bool downloadResponse() {
  downloadURL = "http://" + sunucuIP + "/esp32-phpserver/yanit.wav";

  if (WiFi.status() == WL_CONNECTED) {
    HTTPClient http;
    http.begin(downloadURL);
    int httpResponseCode = http.GET();

    if (httpResponseCode == HTTP_CODE_OK) {
        // Dosya oluşturma
        File file = SD.open(yanitFilePath, FILE_WRITE);
        if (!file) {
            Serial.println("Dosya SD kartta oluşturulamadı.");
            http.end();
            return false;
        }

        // HTTP yanıt akışını al
        WiFiClient* stream = http.getStreamPtr();
        size_t totalBytesRead = 0; // İndirilen toplam bayt miktarını takip eder
        uint8_t buffer[512]; // Daha büyük bir tampon boyutu kullanın

        while (http.connected() && (totalBytesRead < http.getSize() || http.getSize() == -1)) {
            size_t availableBytes = stream->available();
            if (availableBytes > 0) {
                size_t bytesToRead = availableBytes > sizeof(buffer) ? sizeof(buffer) : availableBytes;
                size_t bytesRead = stream->readBytes(buffer, bytesToRead);
                file.write(buffer, bytesRead);
                totalBytesRead += bytesRead;
            } else {
                delay(1); // Biraz bekle ile bir sonraki kontrol
            }
        }

        file.close();
        Serial.printf("Yanıt dosyası başarıyla indirildi. Toplam %lu bayt.\n", totalBytesRead);
        http.end();
        return true;
    } else {
        Serial.printf("HTTP isteği başarısız: %d\n", httpResponseCode);
        http.end();
    }
  } else {
    Serial.println("WiFi bağlantısı kesildi.");
  }

  return false;
}

void playAudio(const char* filePath) {
  if (!SD.exists(filePath)) {
    Serial.println("Çalınacak dosya bulunamadı.");
    return;
  }

  File audioFile = SD.open(filePath);
  if (!audioFile) {
    Serial.println("Ses dosyası açılamadı.");
    return;
  }
  resMkparla(255,255,7);
  Serial.print("Download URL: ");
  Serial.println(downloadURL);
  Serial.println("[DEbUG] WAV dosyası çalınıyor...");
  uint8_t buffer[1024];
  while (audioFile.available()) {
    size_t bytesRead = audioFile.read(buffer, sizeof(buffer));
    size_t bytesWritten = 0;
    i2s_write(I2S_NUM_1, buffer, bytesRead, &bytesWritten, portMAX_DELAY);

    if (bytesWritten < bytesRead) {
        Serial.printf("[ERROR] I2S yazma hatası: %d bayt yazıldı, %d bayt okundu.\n", bytesWritten, bytesRead);
    }
  }
  audioFile.close();
  Serial.println("[DEbUG] Oynatma tamamlandı.");
  resMkparla(0,0,255);
}
```

wifiAyar.ino:

```
#include <WiFi.h>
#include <ESPAsyncWebServer.h>
#include <EEPROM.h>
#include <DNSServer.h>

enum class : DNS_PORT = 53;
WebServer dnsServer;

String sunucuIP = "";

#define BUTTON_PIN 12

// WI-FI ayarları
const char *ssid = "Kamuzi Sesli Asistan";
const char *password = "12345678";

// web sunucu
AsyncWebServer server(80);

// WI-FI bilgilerini EEPROM'dan oku
String storedSSID;
String storedPassword;
String storedIP[

void setup() {
  pinMode(BUTTON_PIN, INPUT_PULLUP);

  // EEPROM'u başlat
  EEPROM.begin(512);

  // WI-FI bilgilerini EEPROM'dan oku
  storedSSID = readEEPROM(0);
  storedPassword = readEEPROM(32); // 32'nci yerdeki şifreye başlangıç adresi
  storedIP = readEEPROM(64); // 32'nci yerdeki başlangıç adresi

  // Eğer WI-FI bilgileri varsa, bu bilgileri bağla ve bağlan
  if (storedSSID.length() > 0 && storedPassword.length() > 0 && storedIP.length() > 0) {
    Serial.println("EEPROM'dan okunan WI-FI bilgileri:");
    Serial.print("SSID: ");
    Serial.println(storedSSID);
    Serial.print("Şifre: ");
    Serial.println(storedPassword);

    sunucuIP = storedIP;

    // WI-FI'ye bağlanmaya çalış
    WiFi.begin(storedSSID.c_str(), storedPassword.c_str());
```

```
    Serial.println("WI-FI bağlanıyor...");
    unsigned long startAttemptTime = millis();
    while (WiFi.status() != WL_CONNECTED) {
      if (millis() - startAttemptTime >= 20000) { // 20 saniye deneme süresi
        Serial.println("WI-FI bağlanılamadı.");
        break;
      }
      delay(100);
    }
    if (WiFi.status() == WL_CONNECTED) {
      Serial.println("Bağlandı, kaydediliyor");
      renkAyarla(0,0,255);
      Serial.println("IP: " + WiFi.localIP().toString());
    } else {
      Serial.print("WI-FI bağlantı durumu: ");
      renkAyarla(255,0,0);
      Serial.println(WiFi.status()); // Hata kodunu yazdır
      startAccessPoint();
    }
  } else {
    // WI-FI bilgileri yoksa AP moduna geç
    startAccessPoint();
    renkAyarla(255,90,0);
  }

  // Web sunucusunu WI-FI ayarlarını alma sayfası olsun
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    String html = R"rawLiteral(
      <html lang="tr-TR">
      <head>
          <meta charset="UTF-8">
          <meta name="viewport" content="width=device-width, initial-scale=1.0">
          <title>Kamuzi Sesli Asistan</title>
          <style>
              * {margin: 0; padding: 0;}
              div.kapsayıcı {
                  width: 100%;
                  height: 100%;
                  display: flex;
                  align-items: center;
                  justify-content: center;
              }
              div.kapsayıcı div.form {
                  width: 300px;
                  height: auto;
                  display: flex;
                  flex-direction: column;
                  align-items: center;
              }
              div.kapsayıcı div.form form label {
                  font-family: Arial,Helvetica Neue,Helvetica,sans-serif;
                  font-size: 18px;
              }
              div.kapsayıcı div.form form input[type=text], div.kapsayıcı div.form form input[type=password] {
                  width: 200px;
                  margin: 0px 0px;
                  padding: 10px;
                  line-height: 24px;
                  font-family: Arial,Helvetica Neue,Helvetica,sans-serif;
                  font-size: 24px;
                  border: 1px solid black;
                  border-radius: 5px;
              }
              div.kapsayıcı div.form form input[type=submit] {
                  width: 100%;
                  padding: 10px;
                  line-height: 24px;
                  font-family: Arial,Helvetica Neue,Helvetica,sans-serif;
                  font-size: 18px;
                  cursor: pointer;
                  border: 1px solid black;
                  border-radius: 5px;
                  background-color: white;
                  margin-bottom: 5px;
              }
              div.kapsayıcı div.form form input[type=submit]:hover {
                  background-color: black;
                  color: white;
              }
          </style>
      </head>
      <body>
          <div class="kapsayıcı">
              <div class="form">
                  <form action="/save" method="POST">
                      <label> Kablosuz Ağ Adı </label>
                      <input type="text" name="ssid">
                      <label> Kablosuz Ağ Şifresi </label>
                      <input type="password" name="password">
                      <label> Sunucu IP (Tor) </label>
```

```
                      <input type="text" name="ip">
                      <input type="submit" value="Gönder">
                  </form>
              </div>
          </div>
      </body>
      </html>
    )rawLiteral";
    request->send(200, "text/html", html);
  });

  // WI-FI ayarlarını kaydetme
  server.on("/save", HTTP_POST, [](AsyncWebServerRequest *request){
    String newSSID = request->arg("ssid");
    String newPassword = request->arg("password");
    String newSunucuIP = request->arg("ip");

    saveEEPROM(newSSID, 0);
    saveEEPROM(newPassword, 32); // Şifreyi EEPROM'un 32. adresinden başlat
    saveEEPROM(newSunucuIP, 64); // IP'yi EEPROM'un 64. adresinden başlat

    request->send(200, "text/html", "<meta charset=\"UTF-8\">Kablosuz ağ bilgileri kaydedildi. ESP32 yeniden başlatılıyor...</br>");
    delay(2000);
    ESP.restart();
  });

  server.begin();
}

void startAccessPoint() {
  Serial.println("AP modunda başlatılıyor...");
  WiFi.softAP(ssid, password);
  Serial.println("Access Point başlatıldı");
  Serial.println("IP: " + WiFi.softAPIP().toString());

  // DNS server'ı başlat ve tüm domainleri ESP32'nin IP adresine yönlendir
  dnsServer.start(DNS_PORT, "*", WiFi.softAPIP());

  // Captive portal yönlendirme
  server.onNotFound([](AsyncWebServerRequest *request) {
    request->redirect("/");
  });
}

String readEEPROM(int addr) {
  String data = "";
  for (int i = addr; i < addr + 32; i++) {
```

```
    char c = EEPROM.read(i);
    if (c == '\0') break; // Null terminator'a geldiğimizde okumayı durdur
    data += c;
  }
  return data;
}

void saveEEPROM(String data, int addr) {
  for (int i = 0; i < data.length(); i++) {
    EEPROM.write(addr + i, data[i]);
  }
  EEPROM.write(addr + data.length(), '\0'); // Null terminator ekle
  EEPROM.commit();
}

void clearEEPROM() {
  for (int i = 0; i < 96; i++) { // 32 byte SSID ve 32 byte şifre ve 32 byte IP için 96 byte
    EEPROM.write(i, 0); // EEPROM'dan tüm verileri sıfırla
  }
  EEPROM.commit();
}

void wifiLoop()
{
  if (digitalRead(BUTTON_PIN) == HIGH) { // Butona basıldıysa
    delay(200); // Debounce için kısa bekleme
    clearEEPROM(); // WI-FI bilgilerini sil
    ESP.restart(); // ESP32'yi yeniden başlat
  }

  dnsServer.processNextRequest();
}
```

- PHP Kodları:

  Api.php:

```php
<?php
// Azure API bilgileri
$speechRegion = "eastus";
$speechKey = "7IifyM3BYEwYxywcQQgx7KB5tQWZ9MCjGHTjBD51CYNwA4uA3e3QQJ99ALACYeBjFXJ3wAAAYACDET jSU";
$language = "tr-TR";

// Çalışma dizinindeki ilk .wav dosyasını bul
$wavFile = __DIR__ . "/kayit.wav";

if (empty($wavFile)) {
    die("Çalışma dizininde .wav dosyası bulunamadı.\n");
}

// Azure Speech API isteği için URL
$url = "https://$speechRegion.stt.speech.microsoft.com/speech/recognition/conversation/cognitiveservices/v1?language=$language&format=detailed";

// API isteği için başlıklar
$headers = [
    "Ocp-Apim-Subscription-Key: $speechKey",
    "Content-Type: audio/wav"
];

// .wav dosyasını okuyarak API'ye gönder
$audioData = file_get_contents($wavFile);

$options = [
    "http" => [
        "header"  => implode("\r\n", $headers),
        "method"  => "POST",
        "content" => $audioData
    ]
];

$context = stream_context_create($options);
$response = file_get_contents($url, false, $context);

if ($response === FALSE) {
    die("Azure API çağrısı başarısız.\n");
}
```

```php
$responseData = json_decode($response, true);

// "DisplayText" objesini al
if (isset($responseData['DisplayText'])) {
    $result = $responseData['DisplayText'];
} else {
    die("Azure API yanıtında DisplayText bulunamadı.\n");
}

// Gemini AI API bilgileri
$geminiApiKey = "AIzaSyCMEc1BUU3Qp38Dd0rmBLdIBbXxF8SqA1"; // Burada kendi API anahtarınızı kullanın
$geminiApiUrl = "https://generativelanguage.googleapis.com/v1beta/models/gemini-1.5-flash:generateContent?key=$geminiApiKey";

// Gemini API'ye gönderilecek veri
$geminiPayload = [
    "contents" => [
        [
            "parts" => [
                ["text" => $result."\r\nverecegin cevap maksimum 200 karakter uzunluğunda olsun"]
            ]
        ]
    ],
    "generationConfig" => [
        "maxOutputTokens" => 800,
        "topP" => 0.8,
        "topK" => 10
    ]
];


// Gemini API isteği
$geminiHeaders = [
    "Content-Type: application/json"
];

$options = [
    "http" => [
        "header"  => implode("\r\n", $geminiHeaders),
```

```php
        "method"  => "POST",
        "content" => json_encode($geminiPayload)
    ]
];

$context = stream_context_create($options);
$geminiResponse = file_get_contents($geminiApiUrl, false, $context);

if ($geminiResponse === FALSE) {
    die("Gemini API çağrısı başarısız.\n");
}

$geminiResponse = json_decode($geminiResponse, true);

// "text" objesini al
if (isset($geminiResponse['candidates'][0]['content']['parts'][0]['text'])) {
    $text = $geminiResponse['candidates'][0]['content']['parts'][0]['text'];
} else {
    echo "Text objesi bulunamadı.\n";
}

// Azure TTS API isteği
$ttsUrl = "https://$speechRegion.tts.speech.microsoft.com/cognitiveservices/v1";
$ttsHeaders = [
    "Ocp-Apim-Subscription-Key: $speechKey",
    "Content-Type: application/ssml+xml",
    "X-Microsoft-OutputFormat: riff-16khz-16bit-mono-pcm",
    "User-Agent: curl"
];

$ssml = "<speak version='1.0' xml:lang='$language'><voice xml:lang='$language' xml:gender='Male' name='tr-TR-AhmetNeural'>$text</voice></speak>";

$options = [
    "http" => [
        "header"  => implode("\r\n", $ttsHeaders),
        "method"  => "POST",
        "content" => $ssml
```

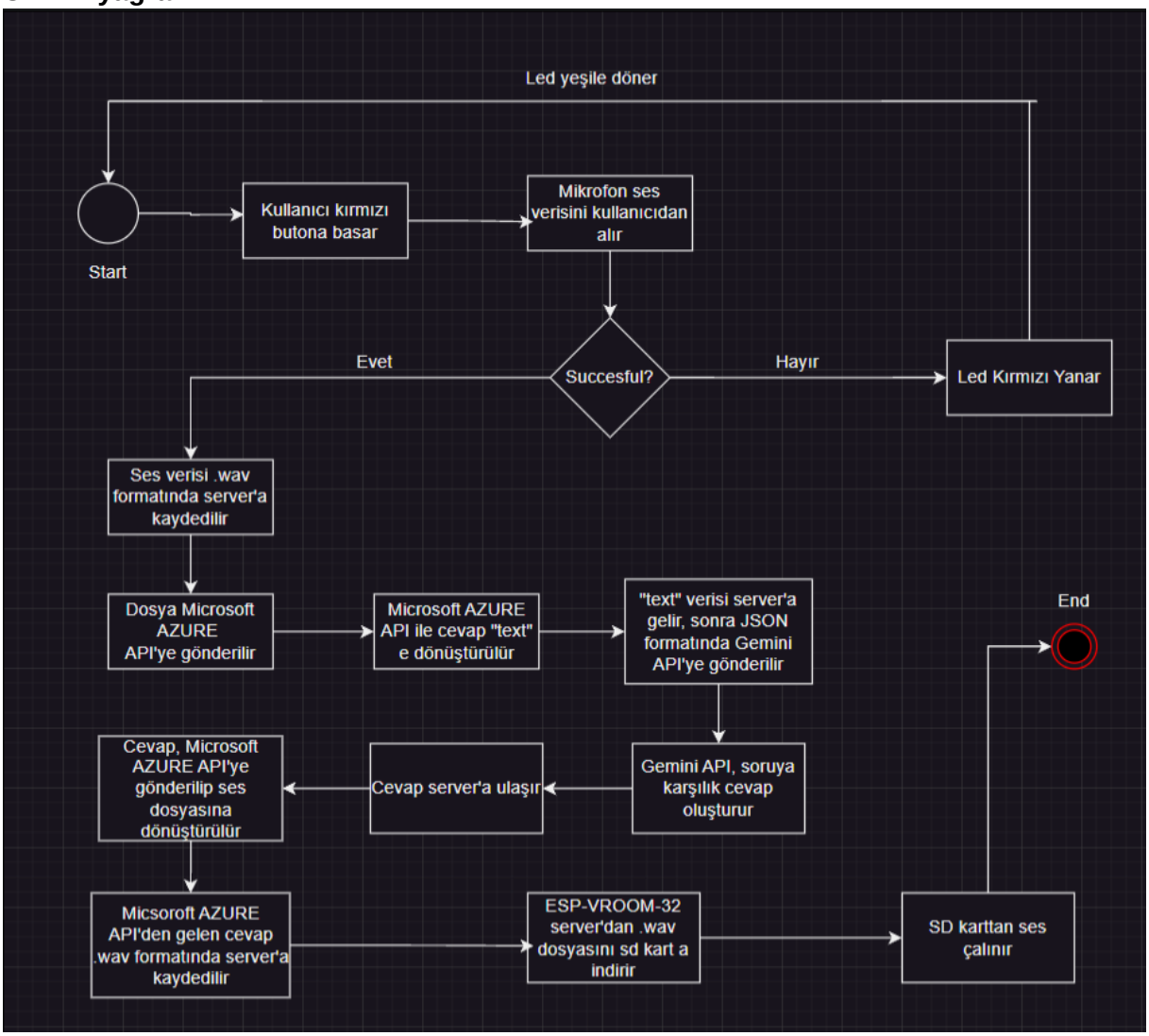
```php
    ]
];

$context = stream_context_create($options);
$ttsResponse = file_get_contents($ttsUrl, false, $context);

if ($ttsResponse === FALSE) {
    die("Azure TTS API çağrısı başarısız.\n");
}

// Yanıtı "yanıt.wav" dosyasına yaz
$filePath = __DIR__ . "/yanit.wav";
if (file_put_contents($filePath, $ttsResponse) === FALSE) {
    die("Yanıt dosyaya yazılamadı.\n");
}

echo "Ses dosyası kaydedildi: $filePath\n";
?>
```

Upload.php

```php
<?php
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $inputData = file_get_contents('php://input');

    // Dosya ismi ve yolu
    $fileName = 'kayit.wav';
    $filePath = __DIR__ . '/' . $fileName;

    // WAV başlığı oluşturma
    $sampleRate = 16000;
    $bitsPerSample = 16;
    $numChannels = 1;
    $byteRate = $sampleRate * $numChannels * $bitsPerSample / 8;
    $blockAlign = $numChannels * $bitsPerSample / 8;

    $header = pack('N', 0x52494646); // "RIFF"
    $header .= pack('V', 36 + strlen($inputData)); // File size - 8
    $header .= pack('N', 0x57415645); // "WAVE"
    $header .= pack('N', 0x666d7420); // "fmt "
    $header .= pack('V', 16); // Subchunk1Size (16 for PCM)
    $header .= pack('v', 1); // AudioFormat (1 for PCM)
    $header .= pack('v', $numChannels); // NumChannels
    $header .= pack('V', $sampleRate); // SampleRate
    $header .= pack('V', $byteRate); // ByteRate
    $header .= pack('v', $blockAlign); // BlockAlign
    $header .= pack('v', $bitsPerSample); // BitsPerSample
    $header .= pack('N', 0x64617461); // "data"
    $header .= pack('V', strlen($inputData)); // Subchunk2Size

    // WAV dosyasını yazma
    file_put_contents($filePath, $header . $inputData);

    echo "File saved successfully: $fileName";

    include("api.php");
} else {
    echo "Invalid request method";
}
?>
```

5.5. UML Diyagramı

- **UML Diyagramı**



---

6. Çalışma Süreci

6.1. Sistem Akışı

1. Kullanıcı sesli komut verir.
2. Mikrofon ses verisini toplar.
3. Veriler Microsoft AZURE API'ye gönderilir.
4. API yanıtı metin olarak server'a döner.
5. Yanıt, JSON formatında Gemini API'ye gönderilir.
6. Gemini API, soruya karşılık cevap oluşturup cevabı server'a gönderir.
7. Metin, tekrardan Microsoft AZURE API'ye gönderilerek, sese çevrilir ve .wav dosyasına dönüştürülür ve server'a kaydedilir.
8. ESP-VROOM-32 .wav dosyasını SD Kart a indirir ve sesi çalar.

6.2. API Kullanımı

Gemini API'si kullanılmış, JSON formatında veri alışverişi yapılmıştır. Şu adımlar takip edilmiştir:

- API anahtarının entegre edilmesi.
- Gemini API, gelen JSON formatındaki verinin içindeki soruya göre cevap üretir.
- Cevabı server' a gönderir.

6.3. Kullanıcı Deneyimi

---

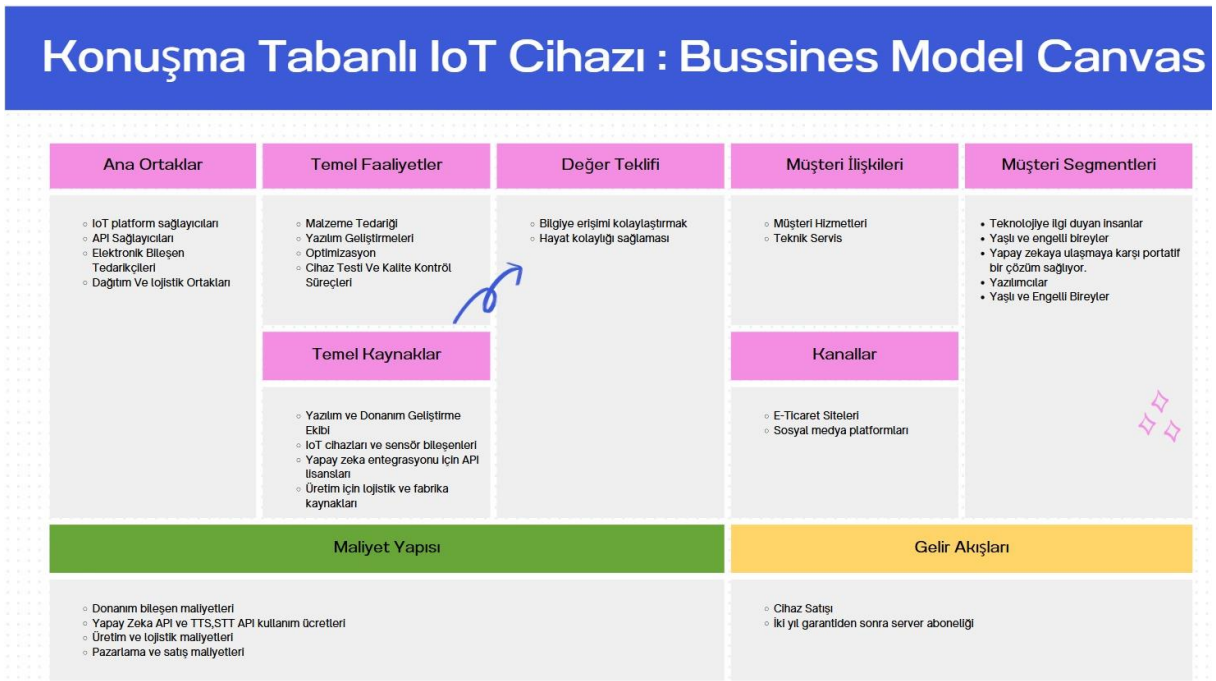
7. Maliyet Analizi

Bileşen	Adet	Birim Fiyat (TL)	Toplam Fiyat (TL)
➢ ESP-VROOM-32	1	295 TL	295 TL
➢ Kingston 32GB Mikro SD Kart	1	200 TL	200 TL
➢ MAX98357 I2S Amfi	1	135 TL	135 TL
➢ INMP441 Mems I2S Omnidirectional Microphone Module	1	110 TL	110 TL
➢ Speaker	1	115 TL	115 TL
➢ 18650 Li-iyon pil	1	100 TL	100 TL
➢ Copper Board	1	50 TL	50 TL
➢ Lehim (2 Metre)	3	30 TL	30 TL
➢ Arduino Micro Sd Kart Modülü	1	25 TL	25 TL
➢ Jumper Kablo	40	0,50 TL	20 TL

➢ TP4056 18650 Lityum Pil Şarj Modülü USB Tip-C	1	15 TL	15 TL
➢ 18650 Lityum İyon Pil Yuvası	1	10 TL	10 TL
➢ 5mm RGB Led	1	5 TL	5 TL
➢ 6x6 4.3mm Tach Button	3	1,30 TL	4 TL
➢ On/Off Switch 180 Derece	1	2 TL	2 TL
➢ 10K 1/4W Direnç	4	0,12 TL	0,50 TL
		Toplam:	**1,101,50 TL**

8. IoT Uygulaması için İş Fikri

8.1 Business Model Canvas



9. Büyük Veri Kullanımı ve Altyapısı

9.1. IoT Uygulamasıyla Elde Edilen Büyük Veri

IoT tabanlı yapay zekâ ile sesli etkileşim cihazımız, doğal olarak kullanıcı davranışlarını ve etkileşimlerini içeren büyük bir veri kaynağı oluşturur. Bu veriler, sistemin kullanıcı deneyimini geliştirmesi ve değer yaratması için çeşitli şekillerde analiz edilebilir.

Projemizden elde edilebilecek büyük veri kaynakları şunlardır:

- Ses Komutları: Kullanıcıların cihaza verdiği sesli komutlar ve konuşma kalıpları.
- Komut Kullanım Sıklığı: Kullanıcıların sık kullandığı özellikler ve belirli saatlerdeki kullanım yoğunlukları.
- Cihaz Yanıt Verileri: Yapay zekanın ürettiği yanıtların kayıtları.
- Kullanıcı Alışkanlıkları: Belirli görevlerin kullanım zamanları ve cihazla etkileşim süreleri.

9.2. Hayatımızı Kolaylaştırabilecek Anlamlı Bilgiler

Büyük veri analizleri ile aşağıdaki anlamlı bilgiler elde edilebilir:

1. **Kullanıcı Profillemesi:**
    - Kullanıcıların sık kullandığı özelliklere göre kişiselleştirilmiş öneriler sunulabilir.
2. **Enerji Verimliliği:**
    - Cihazın hangi saatlerde daha fazla kullanıldığı analiz edilerek enerji tüketimi optimize edilebilir.
3. **Geliştirilmiş Kullanıcı Deneyimi:**
    - Kullanıcı geri bildirimlerinden yola çıkarak cihazın yazılım ve donanımında iyileştirmeler yapılabilir.
4. **Toplumsal Trend Analizi:**
    - Farklı kullanıcı gruplarının cihaz kullanım eğilimleri analiz edilerek, gelecekteki ürün geliştirme süreçlerine yön verilebilir.

9.3. Büyük Veri Altyapısı ve Teknolojileri

Bu verilerin etkin bir şekilde toplanması, saklanması ve analiz edilmesi için aşağıdaki büyük veri altyapı ve teknolojileri önerilir:

- **Veri Depolama:**
    - MySQL veya MariaDB: PHP tabanlı sunucu uygulamanızla kolayca entegre olabilecek, ilişkisel veri tabanı çözümleri. Kullanıcı komutları, yanıtlar ve cihaz kullanım kayıtlarının yapılandırılmış bir şekilde saklanması için uygundur.
    - CSV veya JSON Formatında Loglama: Verilerin basit bir biçimde yedeklenmesi ve taşınabilir formatlarda saklanması.
- **Veri İşleme:**
    - Basit veri işleme ve analiz fonksiyonları PHP'nin yerleşik özellikleri veya ek kütüphaneleri kullanılarak gerçekleştirilebilir.

    - PHP Data Objects (PDO): Veritabanına güvenli erişim ve veri işleme.
- **Veri Analizi:**
    - Python (Pandas, NumPy): Veri setlerini işlemek ve analiz etmek için esnek araçlar.
    - Tableau veya Power BI: Kullanıcı eğilimlerini görselleştirmek ve analiz sonuçlarını paylaşmak için etkili platformlar.
- **Veri Güvenliği ve Anonimleştirme:**
    - **SSL/TLS Sertifikaları:** Sunucu üzerinden yapılan tüm veri iletişimlerinin şifrelenmesini sağlamak için kullanılmalıdır.
    - **Veri Maskeleme:** Kullanıcıların özel bilgilerinin anonimleştirilmesi veya maskeleme tekniklerinin uygulanması.

9.4. Gerekçeler

Bu altyapıları ve teknolojileri seçmenin nedenleri:

1. **Uyumluluk:** PHP, IoT cihazlarından gelen HTTP veya MQTT gibi protokollerle kolayca entegre edilebilir.
2. **Kolay Yönetim:** PHP ve MySQL gibi araçlar, açık kaynaklıdır ve kolay bir öğrenme eğrisine sahiptir.
3. **Esneklik:** PHP ile yazılmış bir sunucu, veri işleme ihtiyaçlarını karşılamak için Python veya diğer dillerle kolayca entegre edilebilir.
4. **Ekonomiklik:** Açık kaynak teknolojiler kullanıldığı için altyapı maliyetleri düşer.

---

10. Sonuç ve Öneriler

Proje, IoT ve yapay zekâ entegrasyonunda başarılı bir örnek sunmuş, ancak gecikme sorunlarını azaltmak ve ses kalitesini iyileştirmek için ek çalışmalar gereklidir.