



NESNELERİN İNTERNETİ VE UYGULAMALARI

IoT Projesi Raporu: Bitki Bakım Düzeneği

Katılımcılar ve Grup Bilgisi

- Öğrenci 1: [Aleyna Çakır, G231210370, 2A]
- Öğrenci 2: [Sıla Çanga, G231210372, 2A]
- Öğrenci 3: [Ahmet Kürşat Sonkur, B211210010, 1A]

1. Proje Tanımı ve Amaç Bu proje, IoT tabanlı bir bitki bakım sistemi geliştirmeyi hedeflemektedir. Başlangıçta evcil hayvanlar için otomatik mama besleme sistemi (pet feeder) olarak tasarlanan bu proje, malzeme kaynaklı teknik aksaklılıklar nedeniyle bitki bakım sistemine dönüştürülmüştür. Bu yeni sistem, bitkilerin sularlanması, ortam nemi ve sıcaklığının izlenmesi gibi temel işlemleri otomatik hale getirmeyi hedeflemektedir.

2. Kullanılan Malzemeler

- **NodeMCU:** Sistemin IoT platformuna entegrasyonu ve sensör verilerinin işlenmesi için kullanılan mikrodenetleyici.
- **Raindrops Modülü:** Yağmur veya sulama durumunu algılamak için kullanılmıştır.
- **Nem Sensörü:** Toprak nem düzeyini izlemek ve sulama gereksinimini belirlemek için.
- **Sıcaklık Sensörü:** Ortam sıcaklığını izlemek.
- **Ağırlık Sensörü:** Bitkinin sulama durumundan kaynaklı ağırlık değişimlerini izlemek.
- **Breadboard:** Elektronik bileşenlerin birbirine bağlanması ve prototip oluşturulması için kullanılan platform.

3. Sistem Tasarımı ve Çalışma Prensibi Bu IoT sistemi, bitki bakımı için bir dizi sensör ve kontrol mekanizmasından oluşmaktadır. Sistem, aşağıdaki bileşenlerden oluşur:

- **Sensörler:** Raindrops, nem ve sıcaklık sensörleri bitkinin ihtiyaçlarını algılamak için veri toplar.
- **Veri Toplama ve İşlem:** Toplanan veriler bir mikrodenetleyici tarafından analiz edilir ve sulama veya diğer işlemler otomatik olarak tetiklenir.
- **Ağ Bağlantısı:** Sistemin IoT platformuna entegrasyonu sağlanarak uzaktan izleme ve kontrol mümkün kılınmıştır.

4. Kod ve Yazılım Geliştirme Sistemin çalışmasını sağlamak için bir mikrodenetleyiciye yüklenmiş kod geliştirilmiştir. Bu kod, sensörlerden gelen verileri okur ve uygun yanıtı şu mantıkla gerçekleştirir:

1. Toprak nem sensörü, sıcaklık sensörü, Raindrops Module verilerini oku.
2. Nem seviyesini ve sıcaklığı al verileri uygulama üzerinden göster.
3. Yağmur yaşıyorsa "Yağmur var!", yağmıyorsa "Yağmur yok." şeklinde uygulamadan göster. Yağmur başladığında mail ve uygulama bildirimini uyarı gönderiyor.
4. Ortam sıcaklığı uygulama ortamında belirtilen sıcaklıkların altına ya da üzerine çıktıığında bildirim ve mail olarak bilgi gönderimi sağlanır.

Kodun tam detayları ek dosyada paylaşılacaktır.

- 5. Sonuç ve Geliştirme Alanları**: Bu proje, bitki bakımını otomatik hale getirerek hem zaman tasarrufu sağlamış hem de bitki sağlığını iyileştirmiştir. Gelecekte şu alanlarda iyileştirme yapılabilir:
- Güneş Enerjisi Kullanımı**: Sistemi daha sürdürülebilir kılmak.
- Ek Sensörler**: Hava kalitesi sensörü veya pH sensörü gibi ek bileşenler eklenebilir.
- Mobil Uygulama Entegrasyonu**: Uzaktan kontrol ve izleme kolaylaştırılabilir.
- Otomatik Gübreleme**: Toprağa gübre eklenmesi gerekiğinde otomatik bir sistem entegre edilebilir.
- Gelişmiş Veri Analitiği**: Sensörlerden alınan verilerin uzun vadeli analiziyle bitki büyümesi ve sağlığına yönelik tahminler yapılabilir.
- Kameralı İzleme**: Bitkilerin büyümeye sürecini gözlemlemek için kamera modülü eklenebilir.
- Wi-Fi ve Bluetooth Entegrasyonu**: Daha fazla cihaz uyumluluğu ve esneklik için bağlantı seçenekleri artırılabilir.

6. Teknik Çizimler ve Görseller

- Sistem şemaları ve sensör yerleşim diyagramları.
- Sensörlerden alınan verilerin grafiksel sunumu.

6. Maliyet Analizi

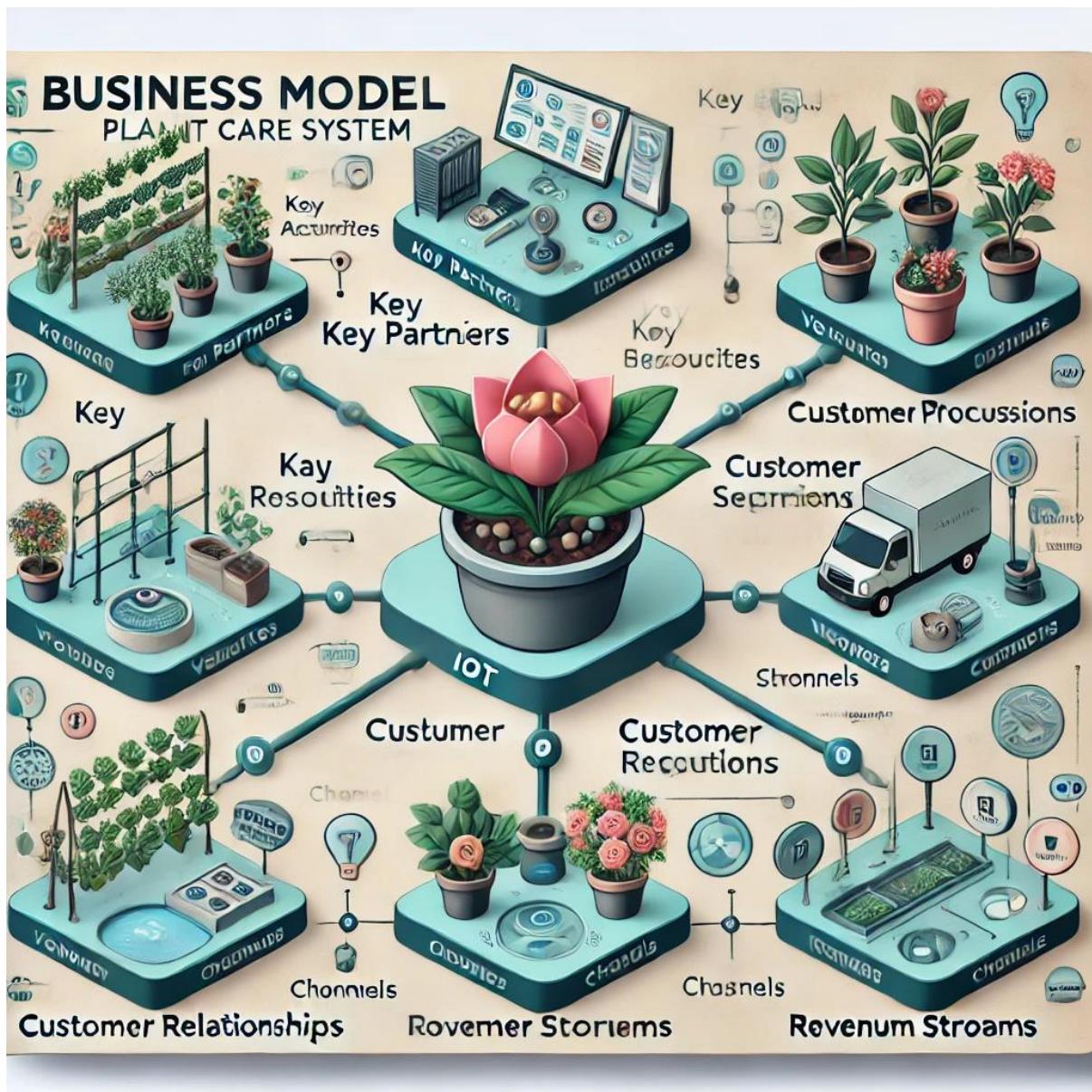
Malzeme	Adet	Birim Fiyat (₺)	Toplam Fiyat (₺)
NodeMCU	2	130	260
Raindrops Modülü	1	100	100
Nem ve Sıcaklık Sensörü	1	40	40
Güç Modülü	1	25	25
Ağırlık Sensörü	1	100	100
Breadboard	2	50	100
Kablolar	20	3	60
Toplam			685

7. İş Modeli (Business Canvas)

Key Partners (Ana Ortaklar)	Key Activities (Ana Faaliyetler)	Key Resources (Ana Kaynaklar)	Value Propositions (Değer Önerileri)
- Elektronik bileşen tedarikçileri	- Sistem tasarımlı, sensör ve mikrodenetleyici entegrasyonu	- Elektronik bileşenler (sensörler, mikrodenetleyiciler)	- Bitki bakımını otomatikleştirerek zaman tasarrufu sağlama.
- Yazılım geliştiriciler ve IoT platform sağlayıcıları	- Yazılım geliştirme ve IoT platformu entegrasyonu	- Yazılım ve IoT platformu altyapısı	- Sağlıklı bitki büyümesi için doğru bakım çözümleri sunma.
- Lojistik ve dağıtım ortakları	- Ürün üretimi ve dağıtım	- Ar-Ge ve tasarım yetenekleri	- Kullanıcı dostu, otomatik sulama, ortam izleme ve uzaktan kontrol.
	- Müşteri desteği ve eğitim materyalleri sağlama	- Üretim ve lojistik altyapısı	

Customer Segments (Müşteri Segmentleri)	Customer Relationships (Müşteri İlişkileri)	Channels (Kanallar)	Revenue Streams (Gelir Akışları)
- Evde bitki yetiştiren bireyler	- Evde bitki yetiştirenler: Otomatik bakım ve çevrimiçi destek.	- E-ticaret platformları (Amazon, Trendyol vb.)	- Ürün satışları: Bitki bakım sistemleri, sensörler ve cihazlar.
- Seracılardan ve botanik severler	- Seracılardan ve botanik severler: Özelleştirilmiş bakım çözümleri, teknik destek ve danışmanlık.	- Sosyal medya ve çevrimiçi pazarlama	- Premium üyelik ve abonelikler: Uzaktan izleme ve ek özellikler için ödeme.
		- Doğrudan satış ve uzmanlıkla ilgili web siteleri	- Teknik destek ve danışmanlık hizmetleri.
		- Mobil uygulama ile uzaktan izleme ve kontrol.	

Cost Structure (Maliyet Yapısı)
- En yüksek maliyetler: Elektronik bileşenler (sensörler, mikrodenetleyiciler), yazılım geliştirme
- Üretim ve lojistik masrafları
- Ar-Ge ve tasarım maliyetleri
- Müşteri destek ve teknik hizmetler



8. Büyük Veri ve IoT Kullanımı Bitki bakım sisteminden elde edilen veriler uzun vadede analiz edilerek şu tür faydalar sağlanabilir:

- Bitkilerin ihtiyaçlarına göre özelleştirilmiş bakım önerileri.
- Bölgesel bitki yetiştirmeye trendleri.
- Toprak, nem ve sıcaklık verilerinden iklimsel tahminler.

Bu tür analizler için büyük veri altyapısı olarak şu teknolojiler önerilmektedir:

- **Apache Hadoop** veya **Apache Spark** gibi büyük veri işleme araçları.
- **AWS IoT Analytics** veya **Google Cloud IoT Core** gibi bulut tabanlı IoT hizmetleri.

9. Kod ve Yazılım Geliştirme Sistemin çalışmasını sağlamak için bir mikrodenetleyiciye yüklenmiş kod geliştirilmiştir. Bu kod, sensörlerden gelen verileri okur ve uygun yanıtı şu mantıkla gerçekleştirir:

1. Toprak nem sensörü verilerini oku.
2. Nem seviyesine göre sulama pompasını çalıştır.
3. Ortam sıcaklığı kritik seviyenin altına düşerse uyarı ver.

Tüm kodlar şu şekildedir.

```
[22:44, 29.12.2024] Kürşat: #include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>

// Blynk bilgileri
#define BLYNK_TEMPLATE_ID "TMPL62MSI-Md8"
#define BLYNK_TEMPLATE_NAME "PetFeeder"
#define BLYNK_AUTH_TOKEN "a8j31LB47vlRTLrca-MOCwq2gdF5bVXRA"

#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

// Wi-Fi bilgileri
const char* ssid = "ks";
const char* password = "123456abc";

// Firebase bilgileri
#define FIREBASE_HOST "https://esp8266-db-aab7b-default.firebaseio.com/"
#define FIREBASE_SECRET "J640JSa7pDPmgUm8LDRBuPARzKDFwqNjohmXYoRx"

FirebaseConfig firebaseConfig;
FirebaseAuth firebaseAuth;
FirebaseData firebaseData;

// DHT11 ve Raindrop sensörleri
#define DHTPIN D2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
#define RAINDROP_PIN D1

// Zamanlayıcılar ve durum bilgileri
```

```
BlynkTimer timer;
bool isRaining = false;
String lastRainStatus = ""; // Firebase'e gereksiz veri göndermeyi önlemek için

// Firebase ve Blynk'e veri gönderme
void sendSensorData() {
    // Sensör verilerini oku
    float h = dht.readHumidity();
    float t = dht.readTemperature();
    if (isnan(h) || isnan(t)) {
        Serial.println("DHT sensör verisi okunamadı!");
        return;
    }

    // Yağmur durumu oku
    int rainStatus = digitalRead(RAINDROP_PIN);
    String rainStatusText = (rainStatus == 0) ? "Yağmur Var" : "Yağmur Yok";

    // Seri monitöre yaz
    Serial.print("Nem: ");
    Serial.print(h);
    Serial.print(" %\tSıcaklık: ");
    Serial.print(t);
    Serial.print(" °C\tYağmur Durumu: ");
    Serial.println(rainStatusText);

    // Firebase'e yalnızca yağmur durumu değiştiğinde veri gönder
    if (rainStatusText != lastRainStatus) {
        lastRainStatus = rainStatusText;

        FirebaseJson json;
        json.set("temperature", t);
        json.set("humidity", h);
        json.set("rainStatus", rainStatusText);

        if (Firebase.pushJSON(firebaseData, "/rainEvents", json)) {
            Serial.println("Veri Firebase'e gönderildi!");
        } else {
            Serial.println("Firebase'e veri gönderilemedi: " + firebaseData.errorReason());
        }
    }
}
```

```
// Blynk'e veri gönder
Blynk.virtualWrite(V5, h); // Nem
Blynk.virtualWrite(V6, t); // Sıcaklık
Blynk.virtualWrite(V7, rainStatusText); // Yağmur durumu

// Yağmur bildirimi (yalnızca değiştiğinde)
if (rainStatus == 0 && !isRaining) {
    isRaining = true;
    Blynk.logEvent("rain_alert", "⚠ Yağmur Tespit Edildi!");
} else if (rainStatus != 0 && isRaining) {
    isRaining = false;
}

yield(); // Watchdog Timer sıfırlama
}

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

    // Wi-Fi'ye bağlan
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Wi-Fi Bağlantısı Sağlandı!");

    // Firebase yapılandırması
    firebaseConfig.host = FIREBASE_HOST;
    firebaseConfig.signer.tokens.legacy_token = FIREBASE_SECRET;
    Firebase.begin(&firebaseConfig, &firebaseAuth);
    Firebase.reconnectWiFi(true);

    // Blynk'i başlat
    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);

    // Her 10 saniyede bir sensör verilerini gönder
    timer.setInterval(10000, sendSensorData);
}
```

```
}

void loop() {
    Blynk.run();
    timer.run();
}

[23:00, 29.12.2024] Kürşat: #include <ESP8266WiFi.h>
#include <FirebaseESP8266.h>
#include <BlynkSimpleEsp8266.h>
#include <DHT.h>

// Wi-Fi bilgileri
const char* ssid = "ks";
const char* password = "123456abc";

// Firebase bilgileri
#define FIREBASE_HOST "https://esp8266-db-aab7b-default-rtdb.firebaseio.com/"
#define FIREBASE_SECRET "J640JSa7pDPmgUm8LDRBuPARzKDFwqNjohmXYoRx"

FirebaseConfig firebaseConfig;
FirebaseAuth firebaseAuth;
FirebaseData firebaseData;

// Blynk bilgileri
#define BLYNK_TEMPLATE_ID "TMPL62MSI-Md8"
#define BLYNK_TEMPLATE_NAME "PetFeeder"
#define BLYNK_AUTH_TOKEN "a8j31LB47vlRTLrca-MOCwq2gdF5bVXRA"

// DHT11 ve Raindrop sensörleri
#define DHTPIN D2
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
#define RAINDROP_PIN D1

// Zamanlayıcılar ve durum bilgileri
BlynkTimer timer;
bool isRaining = false;
String lastRainStatus = "";

// Firebase'e veri gönderme fonksiyonu
void sendFirebaseData() {
```

```
Serial.println("sendFirebaseData başladı.");

float h = dht.readHumidity();
float t = dht.readTemperature();
if (isnan(h) || isnan(t)) {
    Serial.println("DHT sensör verisi okunamadı!");
    return;
}

int rainStatus = digitalRead(RAINDROP_PIN);
String rainStatusText = (rainStatus == 0) ? "Yağmur Var" : "Yağmur Yok";

if (rainStatusText != lastRainStatus) {
    lastRainStatus = rainStatusText;

    FirebaseJson json;
    json.set("temperature", t);
    json.set("humidity", h);
    json.set("rainStatus", rainStatusText);

    if (Firebase.pushJSON(firebaseData, "/rainEvents", json)) {
        Serial.println("Veri Firebase'e gönderildi!");
    } else {
        Serial.println("Firebase'e veri gönderilemedi: " + firebaseData.errorReason());
    }
}

json.clear(); // Belleği temizle
}

yield(); // Watchdog sıfırlama
Serial.println("sendFirebaseData tamamlandı.");
}

// Blynk'e veri gönderme fonksiyonu
void sendBlynkData() {
    Serial.println("sendBlynkData başladı.");

    float h = dht.readHumidity();
    float t = dht.readTemperature();
    if (isnan(h) || isnan(t)) {
        Serial.println("DHT sensör verisi okunamadı!");
```

```
return;
}

int rainStatus = digitalRead(RAINDROP_PIN);
String rainStatusText = (rainStatus == 0) ? "Yağmur Var" : "Yağmur Yok";

Blynk.virtualWrite(V5, h); // Nem
Blynk.virtualWrite(V6, t); // Sıcaklık
Blynk.virtualWrite(V7, rainStatusText); // Yağmur durumu

if (rainStatus == 0 && !isRaining) {
    isRaining = true;
    Blynk.logEvent("rain_alert", "⚠ Yağmur Tespit Edildi!");
} else if (rainStatus != 0 && isRaining) {
    isRaining = false;
}

yield(); // Watchdog sıfırlama
Serial.println("sendBlynkData tamamlandı.");
}

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

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.print(".");
    }
    Serial.println("Wi-Fi Bağlantısı Sağlandı!");

    firebaseConfig.host = FIREBASE_HOST;
    firebaseConfig.signer.tokens.legacy_token = FIREBASE_SECRET;
    Firebase.begin(&firebaseConfig, &firebaseAuth);
    Firebase.reconnectWiFi(true);

    Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);

    timer.setInterval(10000, sendData);
}
```

```
    timer.setInterval(5000, sendBlynkData);
}

void loop() {
    Blynk.run();
    timer.run();
}
```

10. Sonuç ve Geliştirme Alanları Bu proje, bitki bakımını otomatik hale getirerek hem zaman tasarrufu sağlamış hem de bitki sağlığını iyileştirmiştir. Gelecekte şu alanlarda iyileştirme yapılabilir:

- **Güneş Enerjisi Kullanımı:** Sistemi daha sürdürülebilir kılmak.
- **Ek Sensörler:** Hava kalitesi sensörü veya pH sensörü gibi ek bileşenler eklenebilir.
- **Mobil Uygulama Entegrasyonu:** Uzaktan kontrol ve izleme kolaylaştırılabilir.
- **Otomatik Gübreleme:** Toprağa gübre eklenmesi gereğinde otomatik bir sistem entegre edilebilir.
- **Gelişmiş Veri Analitiği:** Sensörlerden alınan verilerin uzun vadeli analiziyle bitki büyümesi ve sağlığına yönelik tahminler yapılabilir.
- **Kameralı İzleme:** Bitkilerin büyümeye sürecini gözlemlemek için kamera modülü eklenebilir.
- **Wi-Fi ve Bluetooth Entegrasyonu:** Daha fazla cihaz uyumluluğu ve esneklik için bağlantı seçenekleri artırılabilir.

