

**2023-2024 AKADEMİK YILI
BAHAR DÖNEMİ**

BM495 - İLERİ GÖMÜLÜ SİSTEM UYGULAMALARI

Ders Sorumlusu:
ENVER KÜÇÜKKÜLAHLI

**İNSANSIZ KARA ARAÇLARINDA IMU
SENSÖRÜ KULLANIMI PROJESİ**

Hazırlayan:
Ebru AVŞAR

Öğrenci No:
201001077

İÇİNDEKİLER

Sayfa No

ŞEKİLLER LİSTESİ.....	iii
ÖZET	4
1. GİRİŞ.....	5
2. MATERYALLER.....	6
2.1. BNO055 9-DOF IMU SENSÖR	6
2.2. ESP-32S	7
2.3. DİĞER MODÜLLER.....	8
3. Yöntem.....	9
3.1. KODLAMA KISMI.....	9
3.1.1. Kütüphane Dosyaları.....	9
3.1.2. WiFi ve MQTT sunucusu ayarları	10
3.1.3. WiFi ve MQTT bağlantıları için nesneler.....	10
3.1.4. I2C pinleri.....	10
3.1.5. BNO055 sensör nesnesi.....	10
3.1.6. setup_wifi fonksiyonu	10
3.1.7. callback fonksiyonu.....	11
3.1.8. reconnect fonksiyonu	11
3.1.9. Setup Fonksiyonu	12
3.1.10. Loop Fonksiyonu	12
3.2. IOTSTACK	13
3.2.1. Temel Bileşenler ve İşlevler.....	13
3.2.1.1. Docker ve Docker Compose	13
3.2.1.2. IOTstack Yapılandırması	14
3.2.1.3. IoT Hizmetleri	14
3.2.2. IOTstack Kurulumu	14
3.2.3. Portainer	16
3.2.4. Mosquitto	16
3.2.5. Node-RED	16
3.2.5.1. Node-RED Akışları Oluşturma.....	17
3.2.6. InfluxDB.....	19
3.2.7. Grafana	21
3.2.7.1. Grafana ile InfluxDB Entegrasyonu	21
3.2.7.2. Dashboard (Pano) Oluşturma	22
4. SONUÇ	24
5. KAYNAKLAR	25
6. EKLER	26

ŞEKİLLER LİSTESİ

Şekil 1 BNO055 9-DOF IMU SENSÖR	6
Şekil 2 ESP-32S	7
Şekil 3 JUMPER KABLO (Dişi-Dişi, Erkek-Erkek)	8
Şekil 4 GÜÇ KABLOSU	8
Şekil 5 BAĞLANTI ŞEMASI	9
Şekil 6 IOTstack menu.sh	15
Şekil 7 BUILD STACK	15
Şekil 8 Pontainer	16
Şekil 9 Node-RED flows	17
Şekil 10 MQTT node	17
Şekil 11 InfluxDB node	18
Şekil 12 Fonksiyon node	19
Şekil 13 InfluxDB işlemleri	20
Şekil 14 InfluxDB X Şekil 15 InfluxDB Y Şekil 16 InfluxDB Z	21
Şekil 17 Grafana InfluxDB	22
Şekil 18 Grafana	23

ÖZET

İNSANSIZ KARA ARAÇLARINDA IMU SENSÖRÜ KULLANIMI

Ebru AVŞAR

Düzce Üniversitesi

Mühendislik Fakültesi Bilgisayar Mühendisliği Proje Ödevi

Danışman: Enver KÜÇÜKKÜLAHLI

Haziran 2024, 15 sayfa

Bu çalışma, insansız kara araçlarında İnertial Measurement Unit (IMU) sensörlerinin kullanımını incelemektedir. Günümüzde, insansız araçlar çeşitli uygulamalarda yaygın olarak kullanılmaktadır ve bu araçların hareket ve konumlandırma sistemlerinde hassaslık önemlidir. IMU sensörleri, ivmeölçer, jiroskop ve manyetometre gibi öğelerle aracın hareket, konum ve yönelim bilgilerini sağlar. Bu çalışmada, Bosch Sensortec tarafından üretilen BNO055 9-DOF IMU sensörü ve ESP-32S mikrodeneleyici modülü kullanılarak, insansız kara araçlarının doğru ve güvenilir bir şekilde konumlandırılması amaçlanmaktadır.

Anahtar sözcükler: BNO055, ESP32, IOTstack.

1. GİRİŞ

Günümüzde insansız kara araçları, geniş bir uygulama yelpazesinde kullanılmaktadır ve bu araçların başarılı bir şekilde hareket etmeleri ve çeşitli görevleri yerine getirebilmeleri, hassas hareket ve konumlandırma sistemlerine bağlıdır. Bu bağlamda, İnertial Measurement Unit (IMU) sensörleri, insansız kara araçlarının hareket, konum ve yönelim bilgilerini sağlamada kilit bir rol oynamaktadır. Bu proje, insansız kara araçlarında IMU sensörlerinin etkin bir şekilde nasıl kullanılabileceğini incelemeyi amaçlamaktadır.

IMU sensörleri, içerdikleri ivmeölçer, jiroskop ve manyetometre gibi öğelerle aracın ivmesini, hızını, rotasını ve yönünü belirleyerek konumlandırma süreçlerine katkı sağlar. Özellikle aracın denge ve stabilitesini koruma, hassas hareketler ve hızlı manevralar gibi durumlarda önemli olan bu sensörler, aynı zamanda aracın dünya üzerindeki konumunu belirlemesi ve harita tabanlı navigasyon sistemlerine doğruluk kazandırması açısından da kritiktir.

Bu projede kullanılan BNO055 9-DOF IMU sensörü, Bosch Sensortec tarafından üretilen bir mutlak konumlandırma sensörüdür. Gyroskop, ivmeölçer ve manyetometre gibi farklı sensör teknolojilerini birleştirerek geniş bir hareket ve konum bilgisi yelpazesi ölçebilen bu sensör, insansız kara araçlarının doğru ve güvenilir bir şekilde konumlandırılmasına olanak tanır.

Ayrıca, bu projede kullanılan ESP-32S mikrodenetleyici modülü, entegre Wi-Fi ve Bluetooth özellikleri ile dikkat çeker. Bu özellikler sayesinde, kablosuz iletişim yeteneklerinin geniş bir yelpazede projelerde kullanılmasına olanak tanır. ESP-32S, geniş bir GPIO pin aralığına sahip olması ve çeşitli haberleşme protokollerini desteklemesi ile çeşitli sensörlerin, aktüatörlerin ve harici cihazların entegrasyonunu kolaylaştırır.

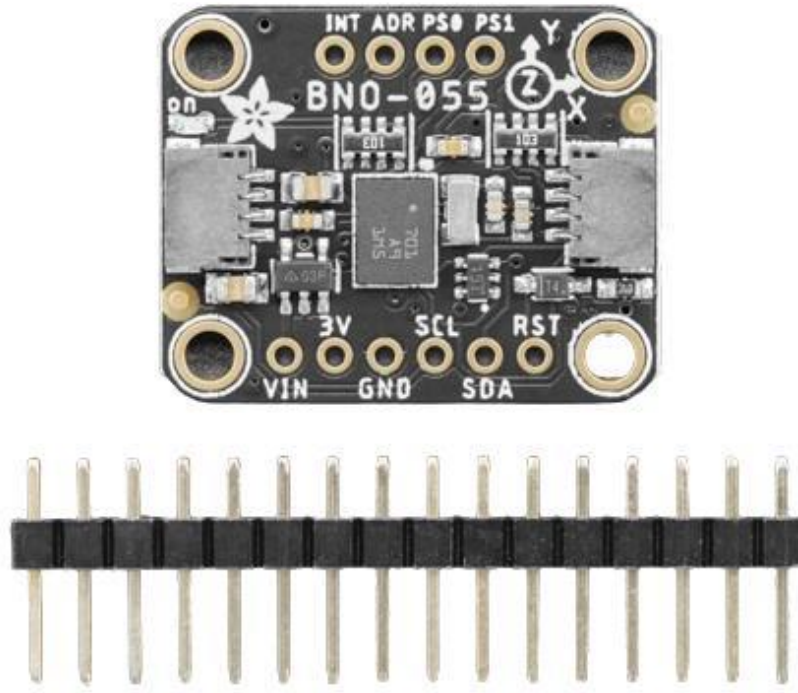
Bu projenin yöntemi, IMU sensörü ve mikrodenetleyici modülü arasındaki bağlantıların fiziksel kurulumunu, kodlama kısmını ve IoTstack adlı bir yazılım yığınının nasıl kullanılacağını kapsar. Kodlama kısmında, gerekli kütüphanelerin ve bağlantıların nasıl kurulacağı, sensör verilerinin nasıl okunacağı ve MQTT protokolü üzerinden nasıl iletilere gönderileceği gibi konular ele alınırken, IoTstack'in kurulumu ve kullanımı da ayrıntılı olarak açıklanmaktadır.

Bu proje, insansız kara araçlarının hassas konumlandırma ve hareket kontrolü için önemli olan IMU sensörlerinin ve uygun mikrodenetleyici modüllerinin etkin bir şekilde nasıl

kullanılabileceğini göstermektedir.

2. MATERYALLER

2.1. BNO055 9-DOF IMU SENSÖR



Şekil 1 BNO055 9-DOF IMU SENSÖR

IMU Sensörü: IMU sensörü, genellikle bir dizi sensörden oluşan bir cihazdır ve bir nesnenin hızını, ivmesini ve eğimini ölçmek için kullanılır. IMU'lar, çeşitli uygulamalarda kullanılan hareket izleme sistemlerinde ve navigasyon sistemlerinde yaygın olarak kullanılır.

IMU, aşağıdaki sensörleri içerebilir:

1. **İvmeölçer:** Bu sensör, bir nesnenin ivmesini ölçer. İvmeölçer, nesnenin hızındaki değişiklikleri algılar.
2. **Jiroskop:** Bu sensör, bir nesnenin dönme hızını ölçer. Nesnenin oryantasyonundaki değişiklikleri algılar.

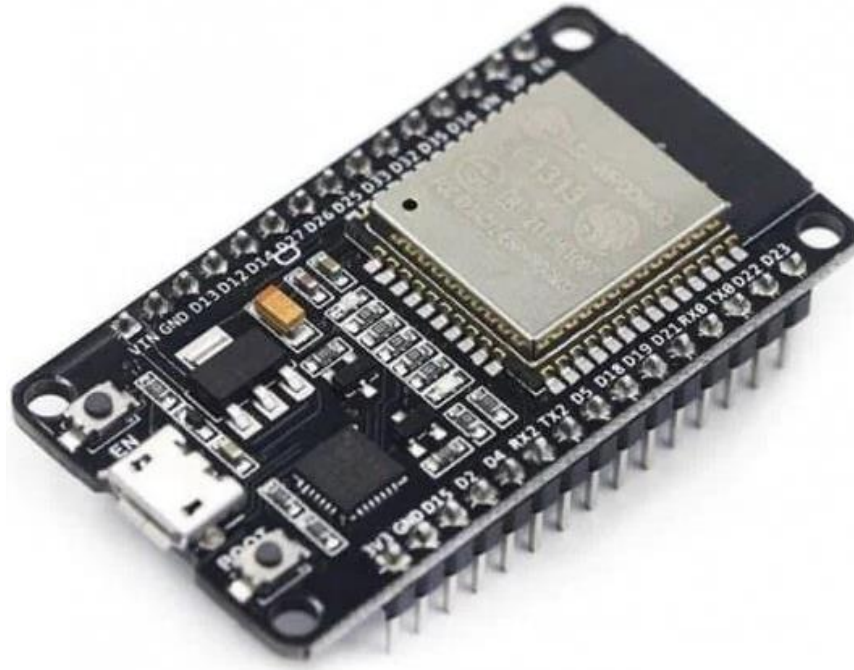
3. **Manyetometre:** Manyetometre, nesnenin yerleşimini belirlemek için dünya üzerindeki manyetik alanı ölçer. Bu, cihazın gerçek kuzeyi belirlemesine yardımcı olabilir.

Bu sensörlerin kombinasyonu, nesnenin konumunu ve hareketini izlemek için kullanılır. Özellikle mobil cihazlarda, giyilebilir teknolojilerde, insansız hava araçlarında (İHA), robotlarda ve diğer benzer uygulamalarda IMU'lar önemli bir rol oynar.

BNO055 Sensörü: BNO055 sensörü, Bosch Sensortec tarafından üretilen bir 9-eksenli mutlak konumlandırma sensörüdür. Bu sensör, gyroskop, ivmeölçer ve manyetometre gibi farklı sensör teknolojilerini birleştirerek geniş bir hareket ve konum bilgisi yelpazesi ölçebilir. Sensör, 3D uzayda hareket ve yönelim bilgilerini belirlemek amacıyla üç ana eksen (X,Y,Z) üzerinde ivme, açısal hız ve manyetik alanı izleyebilir.

BNO055, mutlak konumlandırma sağlayarak başlangıç konumunu referans alır ve cihazın hareketini takip edebilmesini sağlar. Bu, sensörün kullanımını daha etkili hale getirir. Manyetometre bölümü hassas bir şekilde kalibre edilmelidir, bu da sensörün doğru yönlendirme bilgisi sağlamasını sağlar.

2.2. ESP-32S



Şekil 2 ESP-32S

ESP32S, Espressif Systems tarafından geliştirilen bir mikrodenetleyici modüldür. Bu modül, entegre Wi-Fi ve Bluetooth özellikleri ile dikkat çeker, bu da kablosuz iletişim yeteneklerini geniş bir yelpazede projelerde kullanmayı mümkün kılar. İki adet Tensilica LX6 çift çekirdekli mikrodenetleyici içermesi, ESP32S'i daha karmaşık görevleri ve çoklu iş parçacığını eş zamanlı olarak işleme yeteneğine sahip kılar. Ayrıca genellikle 4 MB flaş bellek içerir, bu da program kodu ve diğer veriler için depolama sağlar.

ESP32S, geniş bir GPIO pin aralığına sahiptir, bu da çeşitli sensörler, aktüatörler ve harici cihazların entegrasyonunu kolaylaştırır. Entegre bir ADC sayesinde analog sensörlerle çalışabilir. Ayrıca, birden çok haberleşme protokolünü destekler, bu da çeşitli cihazlar ve modüllerle sorunsuz bir şekilde iletişim kurabilme esnekliği sağlar.

Bu modül, enerji verimliliği sağlayan bir güç yönetimi devresine sahiptir ve pil ile çalışma için uygun hale getirir. Arduino IDE geliştirme ortamı kullanılarak programlanabilir.

Projede kullanılan mikrodenetleyici sayesinde değerleri okunuyor ve yönlendiriliyor.

2.3. DİĞER MODÜLLER

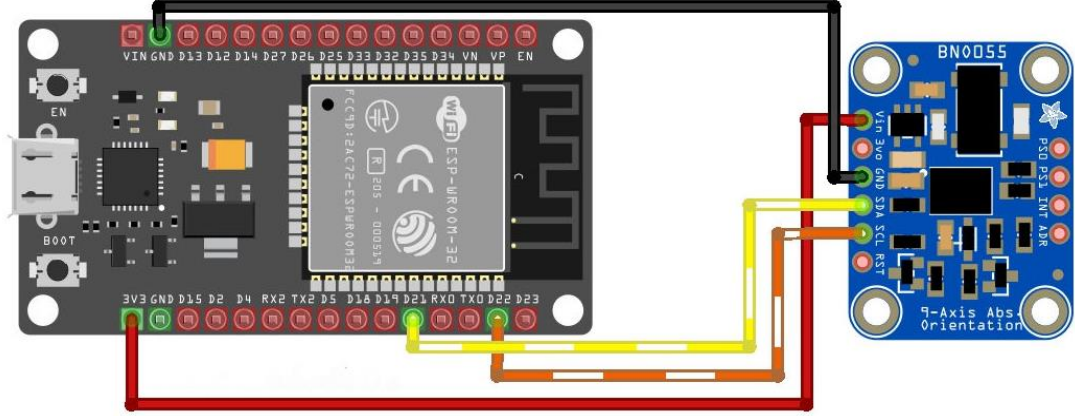


Şekil 3 JUMPER KABLO (Dişi-Dişi, Erkek-Erkek)



Şekil 4 GÜÇ KABLOSU

3. YÖNTEM



Şekil 5 BAĞLANTI ŞEMASI

Proje kapsamında modüllerin güç ve toprak kabloları görseldeki gibi bağlanır.

- IMU Vin: 3v3 pinine bağlantısı yapılır.
- IMU SDA: D21 pinine bağlantısı yapılır.
- IMU SCL: D22 pinine bağlantısı yapılır.
- IMU GND: GND bağlanır.

3.1. Kodlama Kısımı

3.1.1. Kütüphane Dosyaları

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BNO055.h>
#include <utility/imumaths.h>
```

Projede kullanılan çeşitli kütüphaneleri içerir. Bunlar,

- WiFi.h: ESP32 WiFi bağlantısı için.
- PubSubClient.h: MQTT istemcisi için.
- Adafruit_Sensor.h, Adafruit_BNO055.h, utility/imumaths.h: Adafruit BNO055 IMU sensörü için.

3.1.2. WiFi ve MQTT sunucusu ayarları

```
// WiFi ve MQTT ayarları
const char* ssid = "A";
const char* password = "15089443";
const char* mqtt_server = "192.168.179.135";
```

WiFi ve MQTT sunucusu için SSID, şifre ve sunucu IP adresi tanımları.

3.1.3. WiFi ve MQTT bağlantıları için nesneler

```
WiFiClient espClient;
PubSubClient client(espClient);
unsigned long lastMsg = 0;
#define MSG_BUFFER_SIZE (50)
char msg[MSG_BUFFER_SIZE];
```

WiFi ve MQTT bağlantıları için nesneler oluşturuluyor. `lastMsg` değişkeni en son mesajın zamanını saklamak için kullanılıyor. `msg` değişkeni mesajların tutulacağı tampon bellek.

3.1.4. I2C pinleri

```
// BNO055 sensörü için I2C pinleri
#define BNO055_SDA (21)
#define BNO055_SCL (22)
```

BNO055 sensörü için I2C pinleri tanımlanıyor (SDA ve SCL pinleri).

3.1.5. BNO055 sensör nesnesi

```
// BNO055 sensör nesnesi
Adafruit_BNO055 bno = Adafruit_BNO055(55);
```

BNO055 sensör nesnesi oluşturuluyor.

3.1.6. setup_wifi fonksiyonu

```
void setup_wifi() {
  delay(10);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

```

    delay(500);
    Serial.print(".");
}
randomSeed(micros());
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}

```

setup_wifi fonksiyonu WiFi bağlantısını başlatır. Bağlantı sağlanana kadar bekler ve bağlantı sağlandığında IP adresini seri port üzerinden yazdırır.

3.1.7. callback fonksiyonu

```

void callback(char* topic, byte* payload, unsigned int length) {
    Serial.print("Message arrived [");
    Serial.print(topic);
    Serial.print("] ");
    for (int i = 0; i < length; i++) {
        Serial.print((char)payload[i]);
    }
    Serial.println();
}

```

callback fonksiyonu MQTT mesajları geldiğinde çağrılır. Gelen mesajları ve ilgili konuyu seri port üzerinden yazdırır.

3.1.8. reconnect fonksiyonu

```

void reconnect() {
    while (!client.connected()) {
        Serial.print("Attempting MQTT connection...");
        String clientId = "ESP32Client-";
        clientId += String(random(0xffff), HEX);
        if (client.connect(clientId.c_str())) {
            Serial.println("connected");
            client.publish("espStatus", "Connected to MQTT!");
            client.subscribe("espCommand");
        } else {
            Serial.print("failed, rc=");
            Serial.print(client.state());
            Serial.println(" try again in 5 seconds");
            delay(5000);
        }
    }
}

```

reconnect fonksiyonu MQTT sunucusuna bağlanmaya çalışır. Bağlantı sağlanırsa bir

mesaj yayınlar ve belirli bir konuyu dinlemeye başlar. Bağlantı sağlanamazsa 5 saniye bekler ve yeniden dener.

3.1.9. Setup Fonksiyonu

```
void setup() {
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);

  // BNO055 sensörünü başlat
  if (!bno.begin())
  {
    Serial.println("BNO055 sensor not found");
    while (1);
  }
  delay(1000); // Sensörün başlaması için biraz bekle

  // Sensörü kalibrasyon moduna al
  bno.setExtCrystalUse(true);
}
```

setup fonksiyonu cihazın başlangıç ayarlarını yapar:

- Seri haberleşmeyi başlatır.
- WiFi bağlantısını kurar.
- MQTT sunucusunu ve geri arama fonksiyonunu ayarlar.
- BNO055 sensörünü başlatır ve kalibrasyon moduna alır.

3.1.10. Loop Fonksiyonu

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }
  client.loop();

  // Sensör verilerini oku
  sensors_event_t event;
  bno.getEvent(&event);

  // Verileri seri port üzerinden gönder
  Serial.print("X: ");
  Serial.print(event.orientation.x, 4);
  Serial.print("\tY: ");
```

```

Serial.print(event.orientation.y, 4);
Serial.print("\tZ: ");
Serial.print(event.orientation.z, 4);
Serial.println("");

// Verileri MQTT üzerinden yayınla
snprintf (msg, MSG_BUFFER_SIZE, "%f", event.orientation.x);
client.publish("bno055/orientation/x", msg);
snprintf (msg, MSG_BUFFER_SIZE, "%f", event.orientation.y);
client.publish("bno055/orientation/y", msg);
snprintf (msg, MSG_BUFFER_SIZE, "%f", event.orientation.z);
client.publish("bno055/orientation/z", msg);

// Belirli bir süre bekle
delay(500); // Örnek frekans için 500ms bekle
}

```

loop fonksiyonu sürekli olarak çalışır:

- MQTT bağlantısı kesilmişse yeniden bağlanır.
- MQTT istemcisini döngüde tutar.
- BNO055 sensöründen verileri okur.
- Okunan verileri seri port üzerinden ve MQTT üzerinden yayınlar.
- 500 milisaniye bekler ve döngüyü tekrarlar.

3.2. IOTstack

"IOTstack" genellikle, IoT (Nesnelerin İnterneti) cihazlarını yönetmek ve çalıştırmak için kullanılan bir yığın veya çözüm anlamında kullanılır. IOTstack, özellikle Raspberry Pi gibi cihazlarda Docker kullanarak çeşitli IoT servislerini ve araçlarını kolayca dağıtmak ve yönetmek için yapılandırılmış bir sistemdir. Bu sistem, sensör verilerini toplamak, işlemek ve farklı platformlara veya bulut hizmetlerine göndermek için çeşitli araçlar sağlar.

3.2.1. Temel Bileşenler ve İşlevler

3.2.1.1. *Docker ve Docker Compose*

Docker, uygulamaların ve servislerin konteynerler içinde çalıştırılmasını sağlayan bir

platformdur. Docker Compose, birden fazla konteyneri bir araya getirerek tanımlayıp çalıştırmak için kullanılan bir araçtır.

3.2.1.2. *IOTstack Yapılandırması*

Konfigürasyon Dosyaları, Docker Compose dosyaları (docker-compose.yml) ve diğer konfigürasyon dosyaları, hangi servislerin kullanılacağını ve bu servislerin nasıl yapılandırılacağını belirtir. Servis Seçimi, IOTstack, kullanıcılara Node-RED, Mosquitto (MQTT Broker), InfluxDB, Grafana gibi çeşitli servisleri seçme ve yapılandırma imkanı sunar.

3.2.1.3. *IoT Hizmetleri*

Node-RED: Görsel bir araç ile IoT cihazlarını birbirine bağlamak ve otomasyon senaryoları oluşturmak için kullanılır.

Mosquitto: Hafif ve açık kaynaklı bir MQTT brokeridir, MQTT protokolünü kullanarak cihazlar arasında mesajlaşmayı sağlar.

InfluxDB: Zaman serisi veritabanıdır, IoT cihazlarından gelen verilerin saklanması için kullanılır.

Grafana: İzleme ve analitik platformudur, InfluxDB gibi veritabanlarından gelen verileri görselleştirmek için kullanılır.

3.2.2. IOTstack Kurulumu

Curl'ü yüklemek için

```
sudo apt install -y curl
```

Aşağıdaki komutu çalıştırın

```
curl -fsSL https://raw.githubusercontent.com/SensorsIot/IOTstack/master/install.sh |  
bash
```

DHCP'yi kısıtla

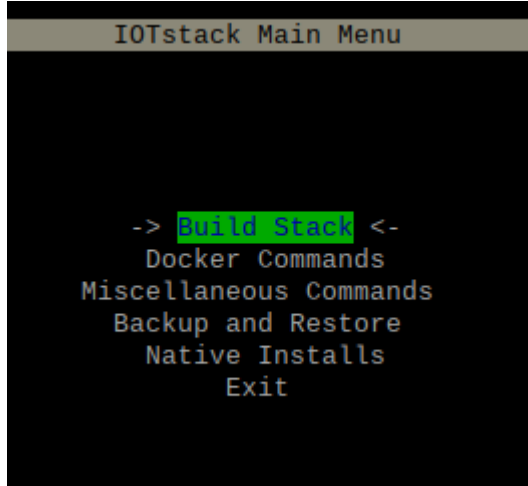
```
sudo bash -c '[ $(egrep -c "^allowinterfaces eth\*,wlan\*" /etc/dhcpd.conf) -eq 0  
&& echo "allowinterfaces eth\*,wlan\*" >> /etc/dhcpd.conf'
```

IOTstack menüsü

İlk docker-compose.yml dosyanızı oluşturmak için

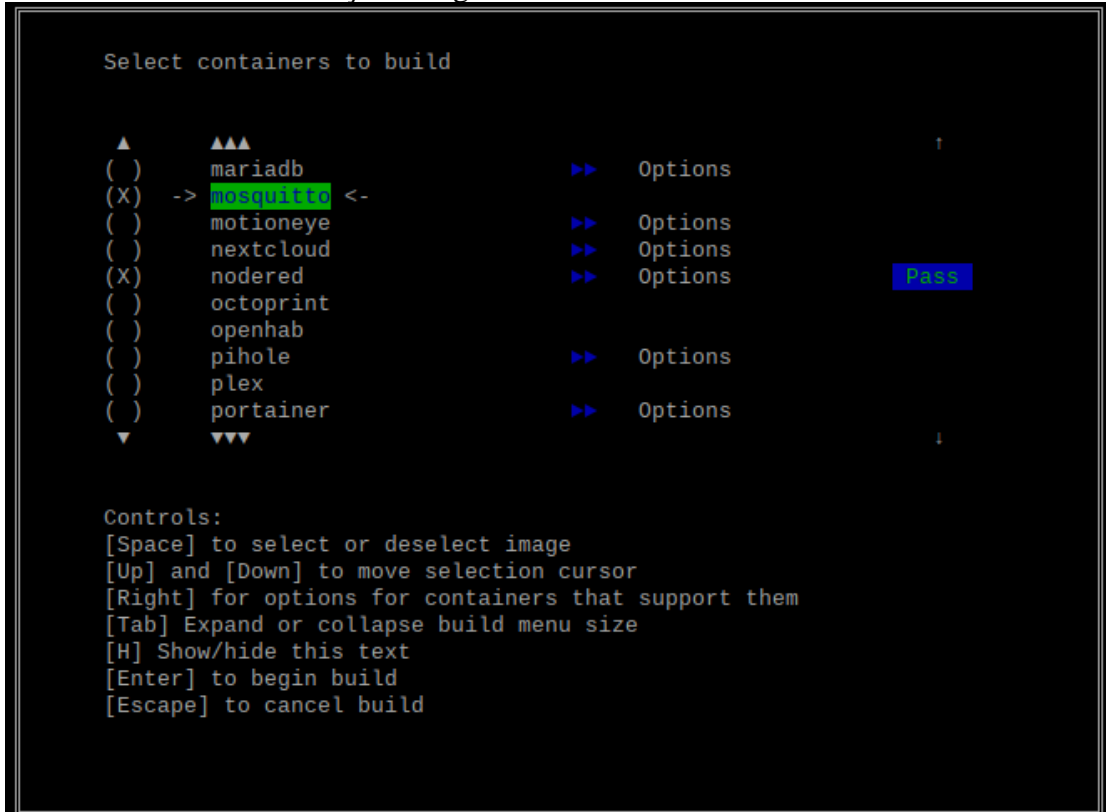
```
cd ~/IOTstack
```

./menu.sh



Şekil 6 IOTstack menu.sh

Bu ekrandan build stacki seçmemiz gerekmektedir



Şekil 7 BUILD STACK

Bu sayfadan proje için ihtiyacımız olan Portainer, Node-RED, Mosquitto, InfluxDB, Grafanayı seçmemiz gerekmektedir. Bu işlemlerden sonra oluşturulan dosyayı açmak için

```
cd ~/IOTstack
```

```
docker-compose up -d
```

3.2.3. Portainer

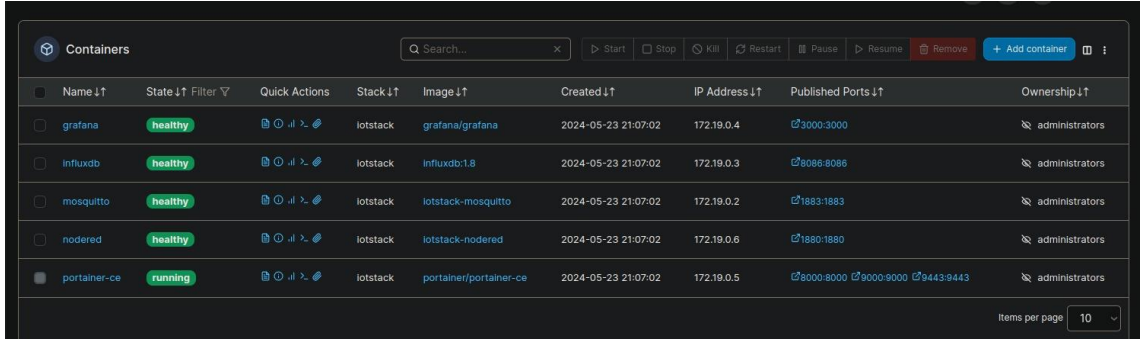
Portainer, Docker konteynerlerini yönetmek için kullanılan bir web tabanlı arayüzdür. IOTstack ile birlikte kullanıldığında, Docker konteynerlerinin kolayca izlenmesi ve yönetilmesi için kullanıcı dostu bir arayüz sağlar.

Portainer'a Erişim için, web tarayıcınızı açın ve localhost ile Portainer'ın portunu kullanarak arayüze erişin:

<http://localhost:9000>

İlk girişte, yönetici kullanıcı adı ve şifreyi belirleyin.

Docker ortamınızı yapılandırmak için "Local" Docker ortamını seçin. Bu, Portainer'ın yerel Docker daemon'unuzu yönetmesine olanak tanır.



The screenshot shows the Portainer web interface with a list of containers. The interface has a dark theme. At the top, there's a search bar and several action buttons: Start, Stop, Kill, Restart, Pause, Resume, Remove, and Add container. Below this is a table with columns: Name, State, Quick Actions, Stack, Image, Created, IP Address, Published Ports, and Ownership. The table lists several containers: grafana (healthy), influxdb (healthy), mosquito (healthy), nodered (healthy), and portainer-ce (running). Each container row has a set of icons for actions like start, stop, kill, restart, pause, resume, and remove. The 'portainer-ce' container is highlighted with a grey background.

Name	State	Quick Actions	Stack	Image	Created	IP Address	Published Ports	Ownership
grafana	healthy	[Icons]	iotstack	grafana/grafana	2024-05-23 21:07:02	172.19.0.4	3000-3000	administrators
influxdb	healthy	[Icons]	iotstack	influxdb:1.8	2024-05-23 21:07:02	172.19.0.3	8086-8086	administrators
mosquitto	healthy	[Icons]	iotstack	iotstack-mosquitto	2024-05-23 21:07:02	172.19.0.2	1883-1883	administrators
nodered	healthy	[Icons]	iotstack	iotstack-nodered	2024-05-23 21:07:02	172.19.0.6	1880-1880	administrators
portainer-ce	running	[Icons]	iotstack	portainer/portainer-ce	2024-05-23 21:07:02	172.19.0.5	8000-8000 9000-9000 9443-9443	administrators

Şekil 8 Pontainer

Sunucuda lazım olan konteynerler aktif ve kullanılabilir durumda olduğunu kontrol edin.

3.2.4. Mosquitto

Mosquitto, hafif bir MQTT mesaj brokeridir. IOTstack ile birlikte kullanıldığında, IoT cihazlarınız arasında verimli ve güvenilir bir mesajlaşma sağlar. 1883 portunu kullanmaktadır. Bu projede ESP ile haberleşmede kullanılacaktır.

3.2.5. Node-RED

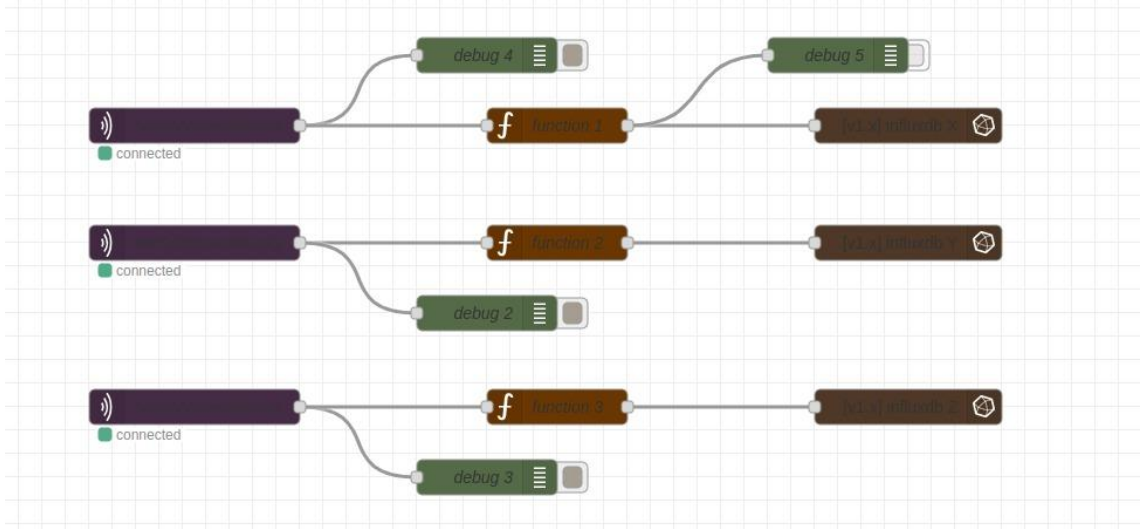
Node-RED, görsel bir programlama aracı olup, IoT cihazlarının birbirine bağlanması ve otomasyon senaryolarının oluşturulması için kullanılır. IOTstack ile birlikte Node-RED'i kurup yapılandırmak oldukça kolaydır.

Node-RED Arayüzüne Erişim, Node-RED'in web tabanlı arayüzüne erişmek için web tarayıcınızı açın ve localhost ile Node-RED'in portunu kullanarak arayüze gidin

<http://localhost:1880>

3.2.5.1. Node-RED Akışları Oluşturma

Node-RED arayüzü üzerinden, çeşitli akışlar (flows) oluşturarak IoT cihazlarınızı birbirine bağlayabilir ve otomasyon senaryoları oluşturulabilmektedir.



Şekil 9 Node-RED flows

3.2.5.1.1. MQTT Girdi (MQTT in) Node'ları:

Bu node'lar MQTT broker'dan belirli bir topic üzerinden mesajları alır. Üç tane MQTT girdi node'u kullanılmamaktadır ve her biri bir topic'e bağlanır. Bu üç node IMU sensöründen gelen X,Y,Z mesajları için ayrı ayrı oluşturulmuştur.

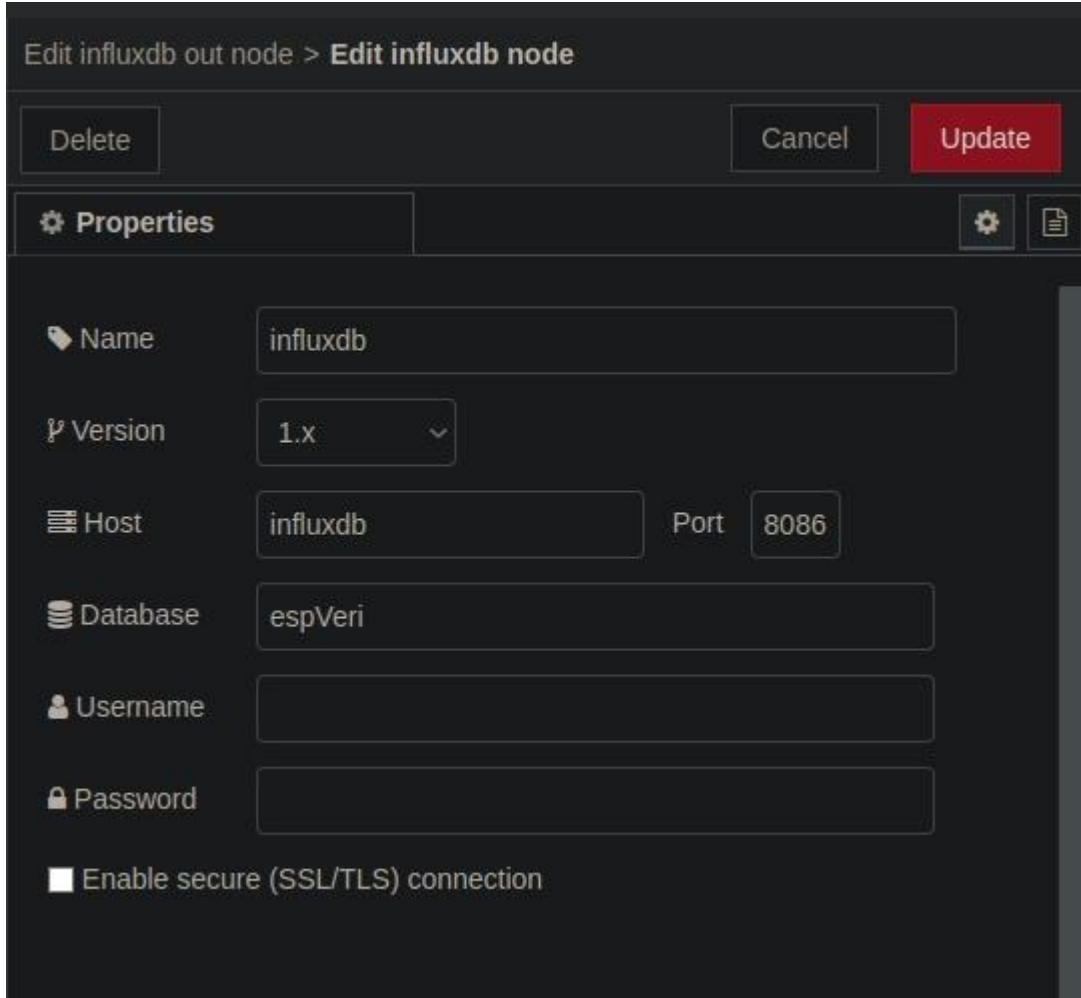
The image shows the configuration dialog for an MQTT node in Node-RED. The dialog has a dark theme and includes the following fields and options:

- Delete** button
- Cancel** button
- Done** button
- Properties** tab (selected)
- Server**: 192.168.179.135:1883
- Action**: Subscribe to single topic
- Topic**: bno055/orientation/x
- QoS**: 2
- Output**: auto-detect (parsed JSON object, string or buff)
- Name**: bno055/orientation/x

Şekil 10 MQTT node

3.2.5.1.2.InfluxDB Node'ları ([v1.x] influxdb):

Bu node'lar, işlenmiş veriyi InfluxDB veri tabanına yazmak için kullanılır. Her bir fonksiyon node'u, işlediği veriyi bağlı olduğu InfluxDB node'una gönderir.



Edit influxdb out node > **Edit influxdb node**

Delete Cancel Update

Properties

Name influxdb

Version 1.x

Host influxdb Port 8086

Database espVeri

Username

Password

☐ Enable secure (SSL/TLS) connection

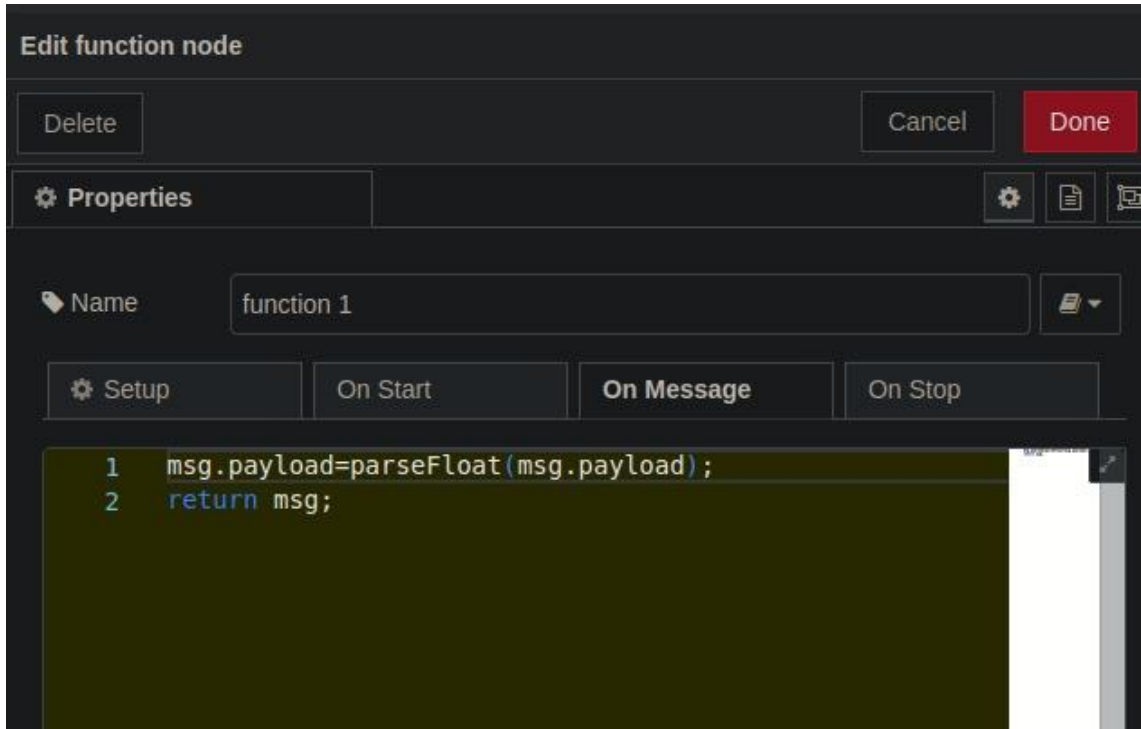
Şekil 11 InfluxDB node

3.2.5.1.3.Debug Node'ları (debug 2, debug 3, debug 4):

Bu node'lar, MQTT girdi node'larından alınan verileri konsolda görüntülemek için kullanılır. Her bir MQTT girdi node'u, gelen mesajları debug node'una gönderir.

3.2.5.1.4.Fonksiyon Node'ları (function 1, function 2, function 3):

Bu node'lar, gelen verileri işlemek için JavaScript kodu çalıştırır. Her bir fonksiyon node'u, bağlı olduğu MQTT girdi node'undan gelen veriyi alır, işleme tabi tutar ve işlenmiş veriyi çıkışa gönderir. Bu işlem gelen string değerleri floata çevirmek için kullanılır.



Şekil 12 Fonksiyon node

Bu akış, MQTT üzerinden gelen verileri alıp işleyerek InfluxDB'ye yazmak için kullanılıyor. Her bir yapı bağımsız olarak çalışıyor ve veri işleme, görüntüleme ve veri tabanına yazma işlevlerini yerine getiriyor.

3.2.6. InfluxDB

InfluxDB, zaman serisi veritabanı olarak kullanılan, performanslı ve açık kaynaklı bir veri tabanıdır. Zaman serisi verisi, belirli zaman damgalarına (timestamps) sahip veri noktalarıdır ve genellikle IoT cihazlarından gelen sensör verileri, performans metrikleri, uygulama izleme verileri gibi sürekli olarak kaydedilen ve analiz edilen verilerdir. InfluxDB, bu tür verileri hızlı bir şekilde yazma, sorgulama ve görselleştirme yetenekleri sağlar. 8086 portunu kullanır.

InfluxDB yi başlatmak için

```
Docker exe -it influxdb influx
```

yazarak influxdb shell açılır.

Kayıtlı olan veritabanlarını görmek için

```
show databases
```

Yeni bir veritabanı oluşturmak için

```
create databases veritabanı_adı
```

Veritabanının oluşup oluşmadığını kontrol etmek için tekrar show databases yazarak kontrol edebilirsiniz.

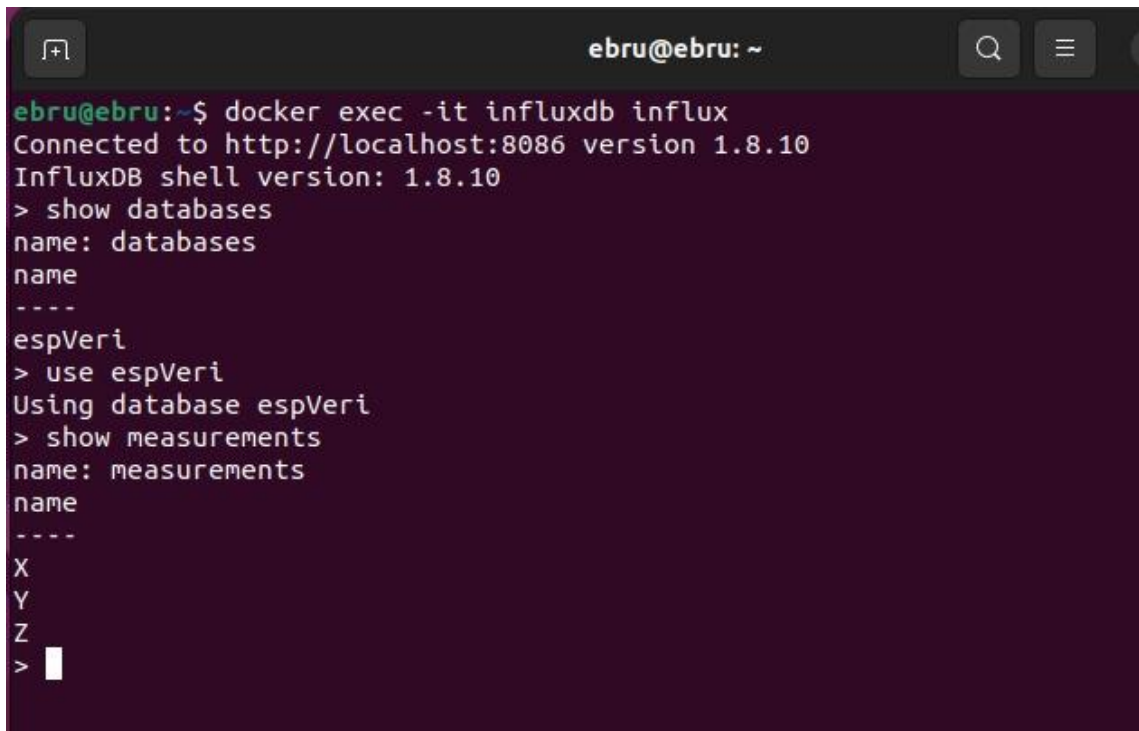
Daha sonra

```
use veritabanı_adı
```

ile oluşturulan veri tabanını kullanabilirsiniz. Veritabanındaki tabloları görmek için

```
show measurements
```

yazarak görüntüleyebilirsiniz.

A terminal window titled 'ebru@ebru: ~' with search and menu icons. The terminal shows the following commands and output:

```
ebru@ebru:~$ docker exec -it influxdb influx
Connected to http://localhost:8086 version 1.8.10
InfluxDB shell version: 1.8.10
> show databases
name: databases
name
----
espVeri
> use espVeri
Using database espVeri
> show measurements
name: measurements
name
----
X
Y
Z
> 
```

Şekil 13 InfluxDB işlemleri

Veritabanındaki tabloların içeriğini görmek için

```
select * from tablo_adı
```

yazarak görüntüleyebilirsiniz.

<pre>> select * from X name: X time value ----- 1716489691437147028 0.31 1716489691962973499 0.44 1716489692360953115 0.62 1716489692845784158 0.62 1716489693382889027 0.69 1716489694049848519 4.06 1716489694417401837 9 1716489694881936972 8.94 171648969526709476 8.31 1716489696141531372 8.63 1716489696554277524 8.63 1716489696959783806 8.38 1716489697589803445 7.37 1716489697983255318 7.62 1716489698604834004 7.56 1716489699114356392 7.06 1716489699849852087 6.75 1716489699960878073 6.56</pre>	<pre>> select * from Y name: Y time value ----- 1716487941855554308 0 1716487942880019024 -49.25 1716487943088217184 -49.25 1716487943905184829 -49.1875 1716487944317071911 -49.3125 1716487944527705438 -49.4375 1716487945338762645 -49.4375 1716487945752391787 -49.5 1716487945954423906 -49.4375 1716487946673208529 -49.375 1716487946880720416 -49.375 1716487947922918014 -49.375 1716487948204726617 -49.3125 1716487948425933207 -49.25 1716487948929667678 -49.1875 1716487949844263654 -49.125 1716487950254406758 -49.125 1716487950866585500 -49.125</pre>	<pre>> select * from Z name: Z time value ----- 1716487941855554822 0 1716487942880077385 -157.625 1716487943088408662 -157.625 1716487943905641733 -157.5625 1716487944317062157 -157.8125 1716487944527925769 -157.625 1716487945338953307 -157.75 1716487945752622972 -157.4375 1716487945954763647 -157.375 1716487946673405970 -157.3125 1716487946880759512 -157.3125 1716487947923077979 -157.25 1716487948204832467 -157.25 1716487948426119595 -157 1716487948929651357 -156.875 1716487949844666800 -156.75 1716487950254514554 -156.6875 1716487950866907113 -156.5625 1716487951060416020 -156.5</pre>
---	---	--

Şekil 14 InfluxDB X

Şekil 15 InfluxDB Y

Şekil 16 InfluxDB Z

3.2.7. Grafana

Grafana, açık kaynaklı bir veri görselleştirme ve izleme aracıdır. IoT projelerinde, cihazlardan toplanan verilerin görselleştirilmesi ve analiz edilmesi için yaygın olarak kullanılır. IOTstack, Grafana'nın kolayca kurulup yapılandırılmasına olanak tanır.

Grafana, çeşitli veri kaynaklarını destekleyen güçlü bir veri görselleştirme ve izleme aracıdır. InfluxDB, Prometheus, Elasticsearch ve MySQL gibi birçok veri kaynağı ile çalışabilir. Kullanıcılar, çeşitli grafikler, çizelgeler ve tablolar ile özelleştirilebilir panolar oluşturabilir. Ayrıca, belirli metrikler için alarm kurabilir ve belirli koşullar karşılandığında bildirim alabilirsiniz. Kullanıcı yönetimi özellikleri ile kullanıcılar ve ekipler için farklı erişim seviyeleri tanımlayabilirsiniz. Grafana'nın zengin eklenti ekosistemi, birçok üçüncü taraf eklentisi ile işlevselliğini genişletmenize olanak tanır.

Grafana Arayüzüne Erişim, Tarayıcınızda <http://localhost:3000> adresine giderek Grafana arayüzüne erişebilirsiniz.

3.2.7.1. Grafana ile InfluxDB Entegrasyonu

InfluxDB Veri Kaynağı Ekleme:

Grafana arayüzünde, sol menüden **Configuration** (yapılandırma) > **Data Sources**

(veri kaynakları) seçeneğine gidin. **Add data source** (veri kaynağı ekle) düğmesine tıklayın ve InfluxDB'yi seçin.

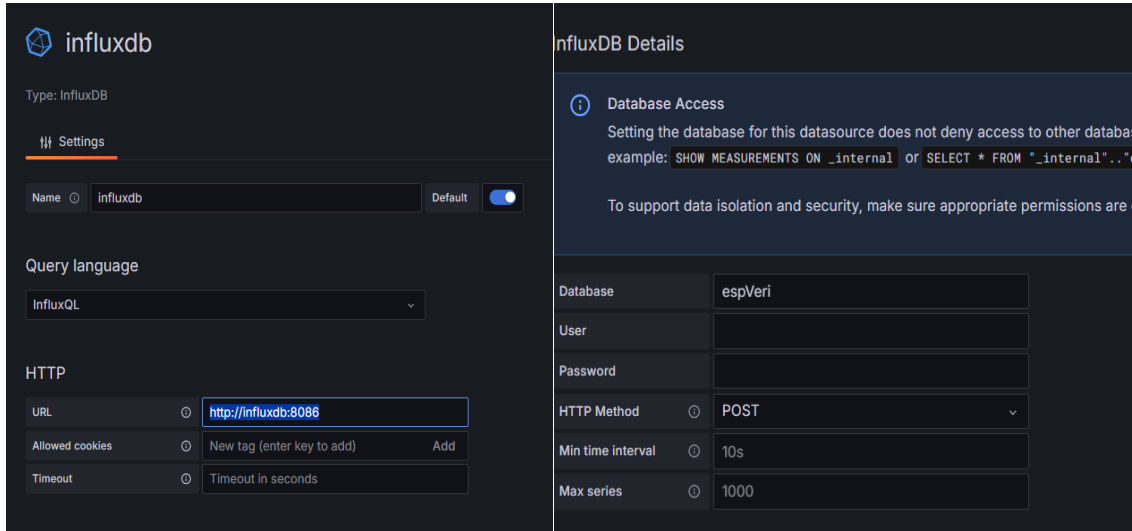
InfluxDB Bağlantı Bilgilerini Girme:

Aşağıdaki bilgileri girerek InfluxDB veri kaynağını yapılandırın:

URL: http://localhost:8086

Database: InfluxDB'deki veritabanınızın adı

HTTP Method: POST



InfluxDB Details	
Database Access Setting the database for this datasource does not deny access to other databases. For example: <code>SHOW MEASUREMENTS ON _internal</code> or <code>SELECT * FROM "_internal".</code> To support data isolation and security, make sure appropriate permissions are set.	
Database	espVeri
User	
Password	
HTTP Method	POST
Min time interval	10s
Max series	1000

Şekil 17 Grafana InfluxDB

Veri Kaynağını Test Etme:

Save & Test düğmesine tıklayarak bağlantıyı test edin. Eğer yapılandırma doğruysa, Grafana InfluxDB'ye başarılı bir şekilde bağlanacaktır.

3.2.7.2. **Dashboard (Pano) Oluşturma**

Grafanalab kullanılarak kullanıcıya grafiksel bir arayüz desteği sağlanılmaktadır. Grafanalab influxdb'den verileri çekerek bunların grafiğini çıkarır.

Yeni Dashboard Ekleme, sol menüden **Create** (oluştur) > **Dashboard** (pano) seçeneğine gidin. **Add new panel** (yeni panel ekle) düğmesine tıklayın.

Panel Yapılandırma, **Panel Title** (panel başlığı) kısmına, göstermek istediğiniz verinin adını yazın. **Data Source** (veri kaynağı) olarak, daha önce eklediğiniz InfluxDB veri kaynağını seçin.

Sorgu Yazma, **Query** (sorgu) sekmesinde, görselleştirmek istediğiniz veriyi almak için

InfluxQL sorgusu yazın.

Paneli Kaydetme, **Apply** (uygula) düğmesine tıklayarak paneli kaydedin. Oluşturduğunuz panel, dashboard'da görünecektir. Diğer panelleri de aynı şekilde oluşturup dashbord'ta gösterilmektedir.



Şekil 18 Grafana

4. SONUÇ

Bu proje, insansız kara araçlarında kullanılan IMU (İnertial Measurement Unit) sensörlerinin önemini vurgulamış ve etkin bir şekilde nasıl kullanılabileceğini ortaya koymuştur. IMU sensörleri, araçların hareket, konum ve yönelim bilgilerini belirlemekte kritik bir rol oynamaktadır. Bu proje kapsamında, BNO055 9-DOF IMU sensörü ve ESP-32S mikrodenetleyici modülü kullanılarak bir prototip oluşturulmuş ve MQTT (Message Queuing Telemetry Transport) ile verilerin iletimi sağlanmıştır.

IMU sensörlerinin kullanım alanları oldukça geniştir. Özellikle insansız kara araçlarında, denge ve stabiliteyi sağlama, hassas hareketler ve hızlı manevralar gibi durumlarda bu sensörlerin kullanımı hayati öneme sahiptir. BNO055 sensörü, gyroskop, ivmeölçer ve manyetometre gibi farklı sensör teknolojilerini bir araya getirerek geniş bir hareket ve konum bilgisi yelpazesi sunar.

Projenin bir diğer önemli bileşeni ise IoTstack altyapısıdır. IoTstack, Docker üzerinde çalışan çeşitli IoT servislerinin kolayca dağıtılmasını ve yönetilmesini sağlayan bir yapıdır. Bu proje kapsamında, Portainer, Node-RED, Mosquitto, InfluxDB ve Grafana gibi IoT servisleri kullanılarak sensör verileri toplanmış, işlenmiş ve görselleştirilmiştir.

Sonuç olarak, bu proje, insansız kara araçlarında IMU sensörlerinin kullanımının önemini vurgulamış ve IoT teknolojisinin sunduğu olanakları kullanarak bir prototip geliştirmiştir. Bu prototip, gelecekteki benzer projeler için bir referans noktası olabilir ve insansız kara araçlarının daha hassas, stabil ve verimli bir şekilde hareket etmesine katkı sağlayabilir.

5. KAYNAKLAR

- [1] <https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor/arduino-code>
- [2] <https://www.direnc.net/9-dof-absolute-orientation-imu-fusion-breakout-bno055-adafruit#:~:text=BNO055%20a%C5%9Fa%C4%9F%C4%B1daki%20sens%C3%B6r%20verilerini%20verebilir,cinsinden%20%C3%BC%C3%A7%20ksen%20'd%C3%B6n%C3%BC%C5%9F%20h%C4%B1z%C4%B1>
- [3] Bosch Sensortec. (2024). BNO055 Absolute Orientation Sensor. <https://www.bosch-sensortec.com/products/smart-sensors/bno055/>
- [4] Espressif Systems. (2024). ESP32S. <https://www.espressif.com/en/products/socs/esp32>
- [5] SensorsIot. (2024). IOTstack. <https://github.com/SensorsIot/IOTstack>
- [6] https://sensorsiot.github.io/IOTstack/Basic_setup/

6. EKLER

