Uygulama Adı:	MQTT Protokolü İle IoT Uygulaması	No:	

Uygulamanın Tanıtımı:

Esp8266 modülüne sahip Wemos D1 Mini IoT cihazı ile belirli verileri MQTT protokolü ile adafruit IoT platformuna gönderen uygulama.

Şekil 1 IoT Cihazı ve Adafruit Mqtt Broker çalışma biçimi

Ekipman Listesi ve Kullanılan Teknolojiler:

- Wemos D1 mini ya da (Arduino + Esp8266 modülü)
- Adafruit IoT platformu
- MQTT protokolü

Kullanılan Teknolojilere Yönelik Teknik Bilgiler:

Wemos D1 Mini

Arduino geliştirme ortamı (IDE), Arduino bootloader (Optiboot), Arduino kütüphaneleri, AVRDude (Arduino üzerindeki mikrodenetleyici programlayan yazılım) ve derleyiciden (AVR-GCC) oluşur. Arduino yazılımı bir geliştirme ortamı (IDE) ve kütüphanelerden oluşur. IDE, Java dilinde yazılmıştır ve Processing adlı dilin ortamına dayanmaktadır. Kütüphaneler ise C ve C++ dillerinde yazılmıştır ve AVR-GCC ve AVR Libc. ile derlenmiştir.

Wemos D1 kartını Ardunio IDE'nizde tanımlı kartlar arasına ekleyebilmek için **Dosya > Tercihler** sekmesindeki ekranda "**Ek Devre Kartları Yöneticisi URLleri**" kutusuna aşağıda verilen linki ekleyiniz.
`http://arduino.esp8266.com/stable/package_esp8266com_index.json`

Esp8266

Kolayca wireless ağlara bağlanmayı sağlayan modül. esp8266-01'den başlayıp esp8266-12'ye kadar giden versiyonları bulunuyor. Kendi firmware'inizi yazıp yükleyerek başka hiçbir şeye ihtiyaç duymadan uygulama geliştirebiliyoruz. AT+ ile başlayan komutları göndererek bağlanılabilir Wi-Fi ağlarının listelenmesi, Wi-Fi adı ve şifresinin gönderilmesiyle ağa bağlanılması, ağ üzerinden bir sunucuyla TCP bağlantısı kurup istemci olarak veri alışverişi yapılması, yine TCP üzerinde server olarak kullanılması gibi işlemler yapılabiliyor.

Wemos D1 mini kartında ESP8266 kütüphanelerini eklemek için Ardunio IDE'de Araçlar > Kart > Kart Yöneticisi ekranından ESP8266 aratıp, kurunuz.

MQTT

MQTT (Message Queuing Telemetry Transport), yayınlama ve abone olma mantığına dayanan telemetry mesajlasma protokolüdür. Makineler arası haberleşmede kullanılmaktadır. Benzer protokollerden ayrılan en önemli özelliği ise hafif (lightweight) olması ve bu sayede bir çok platformda rahatlıkla kullanılabilmesidir.

MQTT Server portu 1883'tür.

Adafruit IoT Platformu ile MQTT haberleşme protokolü kullanarak haberleşebilmek için aşağıdaki linkte verilen kütüphaneyi Ardunio uygulamamıza Taslak > library ekle > . ZIP Kitaplığı Ekle seçeneği ile eklemeliyiz.

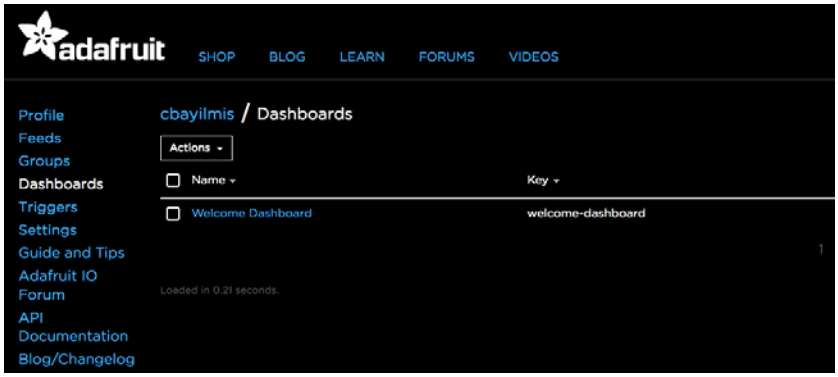
https://github.com/adafruit/Adafruit_MQTT_Library

adafruit.io (Dashboard) IoT Platformu (Web Servis Teknolojisi)

Uygulamanın web üzerinden kontrolü ve kolay yönetilebilmesi için IoT platformu olarak adafruit kullanacağız. adafruit IoT platformu grafik, buton, harita, resim vb. arayüzlerin hızlı bir şekilde kullanılabilmesini sağlamaktadır.

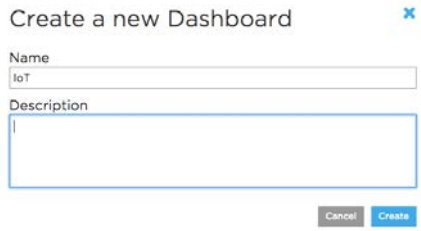
MQTT gibi IoT haberleşme (web servis) protokollerini destekler.

io.adafruit.com adresinden üye olunduktan sonra Şekil ….'de görüldüğü üzere https://io.adafruit.com/kullaniciadi/dashboards adresindeki arayüz aracılığıyla IoT uygulamanıza yönelik paneli (*dashboard*) oluşturabilirsiniz.



Şekil 2: Adafruit.io ilk giriş Dashboard arayüzü

IoT uygulamamızın kontrolü için yeni bir panel (dashboard) oluşturmak için Action sekmesinden Create a New Dashboard ile yeni dashboard'un adını "IoT" olarak tanımladık. Bu arayüzden yeni dashboard oluşturmak mümkünken mevcut dashboard üzerinden düzenlemeler yapabilirsiniz.



Şekil 3: Yeni bir dashboard oluşturma

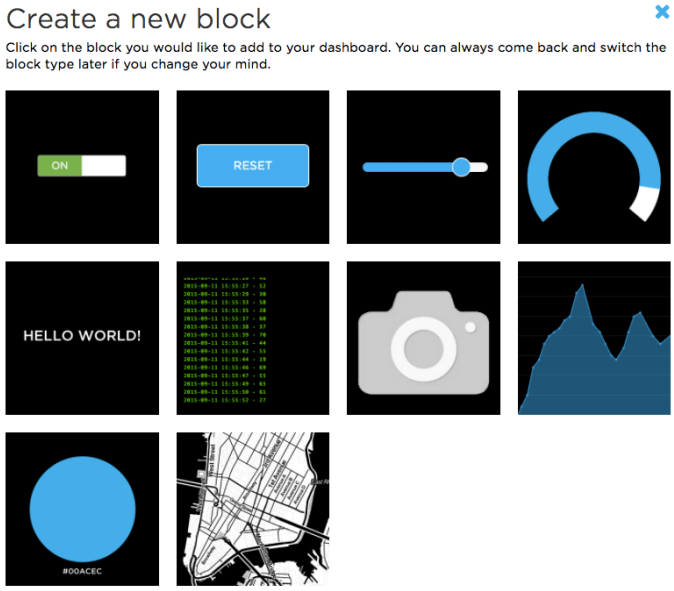
Şekil 4: Oluşturulan dashboard lar

IoT dashboard sekmesini tıklayarak panelimizi uygulamamıza göre özelleştirebiliriz. (https://io.adafruit.com/kullaniciadi/dashboards/iot)

Şekil 5: Oluşturulan IoT dashboard

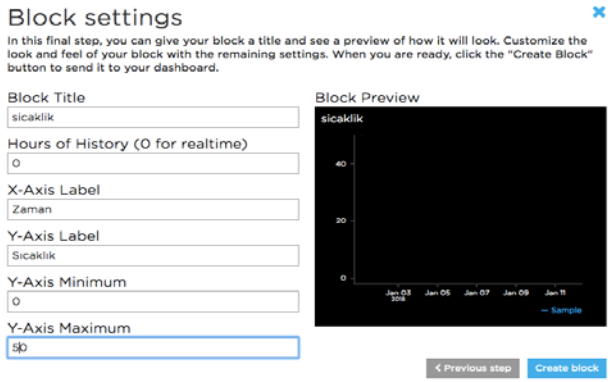
Kilit sekmesi, panelin görünürlük (visibility) değerini göstermektedir. Kapalı kilit bu değerin private (sadece kullanıcı tarafından erişilebilir) olduğunu gösterir. İlgili butona tıklayıp public (herkes tarafından erişilebilir) olarak ayarlanabilir.

artı sekmesi ile ise dashboard ta görülmesini istediğimiz buton, grafik, slider, harita vb. bloklar eklenebilir.



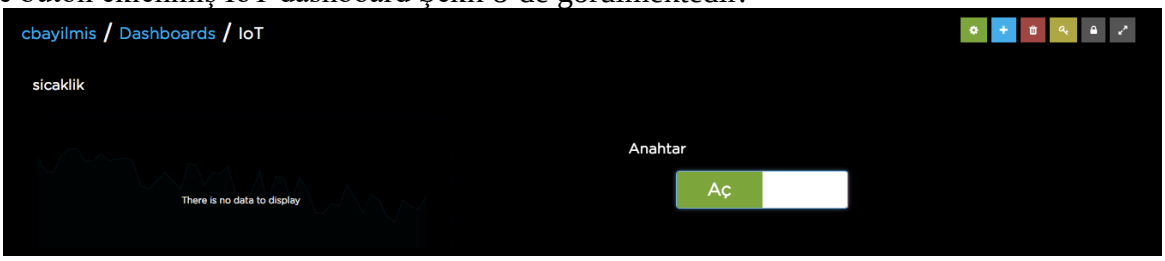
Şekil 6: Oluşturulabilecek Blok Tipleri

Grafik (chart) blok ekleme işlemleri Şekil 7'da görülmektedir.



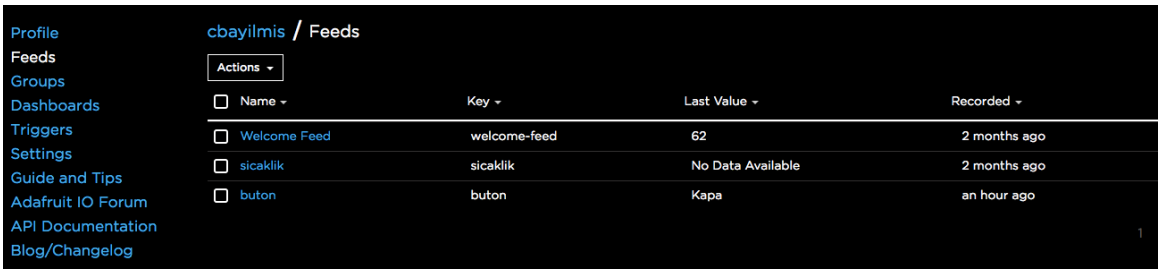
Şekil 7: Grafik blok ekleme işlemi

Grafik ve buton eklenmiş IoT dashboard Şekil 8'de görülmektedir.



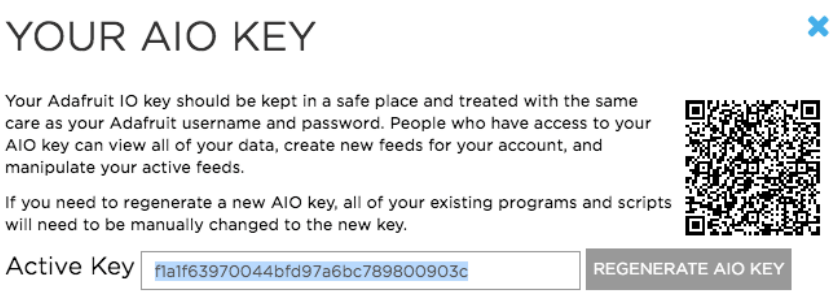
Şekil 8: IoT dashboard eklenen bloklar

Edit sekmesi ile panele yerleştirilmiş, bloklar düzenlenebilir.
Adafruit kullanıcı sayfasımızdan Feeds sekmesinden bloklarımıza ait feed (besleme) isimlerini görebiliriz. Feed isimlerini blokları oluştururken veriliyordu. Feed isimleri Ardunio kod kısmında IoT panelimize veri göndermek ya da veri almak için kullanılacaktır.



Şekil 9: Feed işlemleri

Anahtar sekmesi ile ise IoT panelimize erişmek üzere bize özgü AIO Anahtara erişilir.



Şekil 10: AIO Anahtarı

Uygulama Adımları

1. Adafruit.io da "sau" isimli yeni bir grup oluşturun
2. "sau" isimli grubun altına "mqtt" isimli yeni bir feed oluşturun
3. Wemos cihazında uygulama kodlarını yazın.
4. Adafruit.io da "mqtt" adlı feede tıklayarak veri akışını izleyin

Uygulamanın Wemos D1 Mini Kodları

```
#include <ESP8266WiFi.h>
#include "Adafruit_MQTT.h"
#include "Adafruit_MQTT_Client.h"

/************************ WiFi Access Point *********************************/

#define WLAN_SSID       "KablosuzAgAdi"
#define WLAN_PASS       " KablosuzAgSifresi"

/************************ Adafruit.io Setup *********************************/

#define AIO_SERVER      "io.adafruit.com"
#define AIO_SERVERPORT  1883                   // use 8883 for SSL
#define AIO_USERNAME    "AdafruitKullaniciAdi"
#define AIO_KEY         "ad8a48d5cb5c423183e7c80cdcf3f407" //Adafruit AIO Key


// Create an ESP8266 WiFiClient class to connect to the MQTT server.
WiFiClient client;

Adafruit_MQTT_Client mqtt(&client, AIO_SERVER, AIO_SERVERPORT, AIO_USERNAME, AIO_KEY);

/************************** Feeds ***********************************/

// Setup a feed called 'saumqtt' for publishing.
// Notice MQTT paths for AIO follow the form: <username>/feeds/<feedname>
Adafruit_MQTT_Publish feed = Adafruit_MQTT_Publish(&mqtt, AIO_USERNAME "/feeds/sau.saumqtt");

// Setup a feed called 'onoff' for subscribing to changes.
Adafruit_MQTT_Subscribe onoffbutton = Adafruit_MQTT_Subscribe(&mqtt, AIO_USERNAME
"/feeds/onoff");

/************************* Sketch Code ***********************************/

// Bug workaround for Arduino 1.6.6, it seems to need a function declaration
// for some reason (only affects ESP8266, likely an arduino-builder bug).
void MQTT_connect();

void setup()
{
    Serial.begin(115200);
    delay(10);

    Serial.println(F("Adafruit MQTT demo"));

    // Connect to WiFi access point.
    Serial.println(); Serial.println();
    Serial.print("Connecting to ");
    Serial.println(WLAN_SSID);
```

```
    WiFi.begin(WLAN_SSID, WLAN_PASS);
    while (WiFi.status() != WL_CONNECTED)
    {
        delay(500);
        Serial.print(".");
    }
    Serial.println();

    Serial.println("WiFi connected");
    Serial.println("IP address: "); Serial.println(WiFi.localIP());

    // Setup MQTT subscription for onoff feed.
    mqtt.subscribe(&onoffbutton);
}

uint32_t x = 0;

void loop()
{
    // Ensure the connection to the MQTT server is alive (this will make the first
    // connection and automatically reconnect when disconnected).  See the MQTT_connect
    // function definition further below.
    MQTT_connect();

    // this is our 'wait for incoming subscription packets' busy subloop
    // try to spend your time here

    Adafruit_MQTT_Subscribe* subscription;
    while ((subscription = mqtt.readSubscription(5000)))
    {
        if (subscription == &onoffbutton)
        {
            Serial.print(F("Got: "));
            Serial.println((char*)onoffbutton.lastread);
        }
    }

    // Now we can publish stuff!
    Serial.print(F("\nSending feed val "));
    Serial.print(x);
    Serial.print("...");
    if (!feed.publish(x++))
    {
        Serial.println(F("Failed"));
    }
    else
    {
        Serial.println(F("OK!"));
    }

    // ping the server to keep the mqtt connection alive
    // NOT required if you are publishing once every KEEPALIVE seconds
    /*
    if(! mqtt.ping()) {
      mqtt.disconnect();
    }
    */
}

// Function to connect and reconnect as necessary to the MQTT server.
```

```
// Should be called in the loop function and it will take care if connecting.
void MQTT_connect()
{
    int8_t ret;

    // Stop if already connected.
    if (mqtt.connected())
    {
        return;
    }

    Serial.print("Connecting to MQTT... ");

    uint8_t retries = 3;
    while ((ret = mqtt.connect()) != 0)
    { // connect will return 0 for connected
        Serial.println(mqtt.connectErrorString(ret));
        Serial.println("Retrying MQTT connection in 5 seconds...");
        mqtt.disconnect();
        delay(5000);  // wait 5 seconds
        retries--;
        if (retries == 0)
        {
            // basically die and wait for WDT to reset me
            while (1) ;
        }
    }
    Serial.println("MQTT Connected!");
}
```

KAYNAKLAR

Doç. Dr. Cüneyt BAYILMIŞ ve Doç. Dr. Kerem KÜÇÜK, **"Nesnelerin İnternet'i: Teori ve Uygulamaları"**, Papatya Yayınevi, 2019.