

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/359921045>

RASPBERRY PI ile NESNELERİN İNTERNETİ (IoT) UYGULAMALARI

Chapter · April 2022

CITATIONS

0

READS

1,003

1 author:



[Murat Altun](#)

Milli Eğitim Bakanlığı

25 PUBLICATIONS 7 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Öğrenci Akademik Performansının Kestirilmesine İlişkin Bir Model Önerisi: Veri Madenciliğine Dayalı Bir Çalışma (Model Proposal Related To Predicting Student Academic Performance: A Study Based On Data Mining) [View project](#)

RASPBERRY PI ile NESNELERİN İNTERNETİ (IoT) UYGULAMALARI

Bu bölümde Raspberry Pi üzerinde Python programlama dili kullanılarak nesnelerin interneti uygulamaları gösterilecektir. Bu uygulamalar, Raspberry Pi'nin GPIO pinlerinden veri almayı (sıcaklık, nem, ışık ...) ve yine bu pinleri kullanarak bağlı devre elemanlarını veya cihazları (led, lamba, kapı kilidi...) kontrol etmeyi sağlayacaktır.

MQTT Kullanımı

Raspberry Pi üzerinde **MQTT** kullanarak cihazların haberleşmesini sağlanabilmektedir. Bunun için sisteme **Mosquitto** kurulur. Terminal açılarak sistemi güncellemek için aşağıdaki komutlar kullanılır. Aşağıdaki komutlar kullanılarak repo bilgileri güncellenir ve uygun durumdaki tüm paketler yükseltilebilir.

```
sudo apt update
```

```
sudo apt upgrade
```

Raspberry Pi üzerine **Mosquitto** (MQTT broker) kurmak için aşağıdaki komutlar kullanılır. mosquitto-clients bilgisayardan hem yayın yapmak (publish) hem de yayına abone (subscribe) olmak için kullanılır.

```
sudo apt install -y mosquitto mosquitto-clients
```

Mosquitto versiyonu kontrol edilerek kurulum doğrulanabilir.

```
mosquitto -v
```

Komut çalıştırıldığında çıktı aşağıdaki gibi olmalıdır.

```
1614365055: mosquitto version 1.5.7 starting
1614365055: Using default config.
1614365055: Opening ipv4 listen socket on port 1883.
1614365055: Error: Address already in use
```

Raspberry açıldığında otomatik olarak **Mosquitto** başlamasını sağlamak için aşağıdaki komut çalıştırılır. Bu komutun bir program aracılığıyla (Python'da yazılan bir kod) yapılan bir yayının veya aboneliğin otomatik olarak başlatılması için kullanılması gerekebilir.

```
sudo systemctl enable mosquitto
```

Aşağıdaki komut kullanılarak çalışma durumu kontrol edilebilir.

```
sudo systemctl status mosquitto
```

Servis sorunsuz çalışıyorsa çıktı aşağıdaki gibi olmalıdır.

```
Active: active (running) since Tue 2021-02-16 00:00:37 +03; 33s ago
Şub 16 00:00:37 raspberrypi systemd[1]: Starting Mosquitto MQTT v3.1/v3.1.1 Broker...
```

Şub 16 00:00:37 raspberrypi systemd[1]: Started Mosquitto MQTT v3.1/v3.1.1 Broker

Mosquitto servis kontrolü ile ilgili diğer komutlar

Servisi durdurmak için aşağıdaki komut kullanılır.

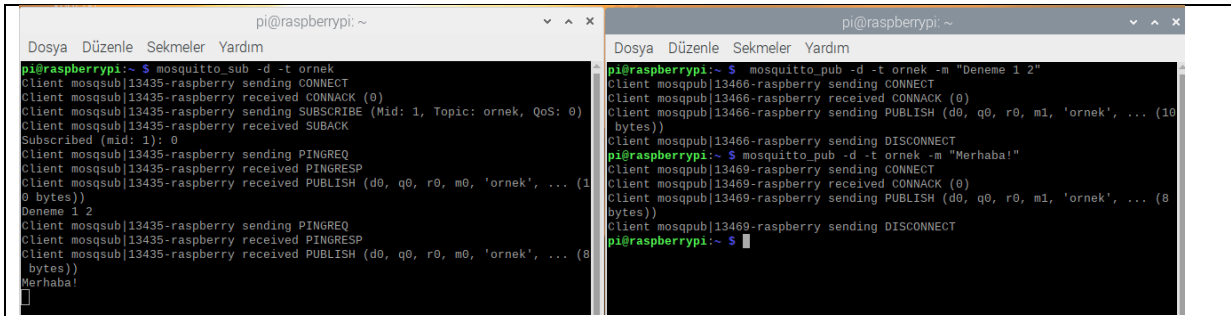
```
systemctl stop mosquitto
```

Servisi otomatik başlatmasını kapatmak için ise aşağıdaki komut kullanılır.

```
systemctl disable mosquitto
```

MQTT Broker Test Etme

Raspberry Pi üzerinde MQTT'yi yerel olarak test etmek için iki terminal penceresi açılır. Pencerelelerden biri yapılan yayına abone olmak (subscribe, Şekil 9.1) için ikincisi ise yayın yapmak (publish, Şekil 9.2) için kullanılmaktadır.



Şekil 9.1. Yayına abone olmak (subscribe)

Şekil 9.2. Yayın yapmak (publish)

<pre>pi@raspberrypi:~ \$ mosquitto_sub -d -t ornek Deneme 1 2 Merhaba!</pre>	<pre>pi@raspberrypi:~ \$ mosquitto_pub -d -t ornek -m "Deneme 1 2" pi@raspberrypi:~ \$ mosquitto_pub -d -t ornek -m "Merhaba!"</pre>
--	---

Soldaki terminal ile **mosquitto_sub** komutu kullanılarak **ornek** konulu yayınına abone olunmuştur. Bu yayın bir sensör verisi veya bir butonun durumunu gösteren bir yayın olabilir. Sağdaki terminal ile sırayla “Deneme 1 2” ve “Merhaba!” mesajları gönderilmiştir. Sağdaki terminalden bir ileti gönderildiğinde soldaki terminalde gönderilen ileti görünmektedir.

Web üzerinde bir **MQTT** yayına bağlanmak için aşağıdaki komut kullanılabilir.

```
pi@raspberrypi:~ $ mosquitto_sub -h broker.mqttdashboard.com -t testtopic/1
```

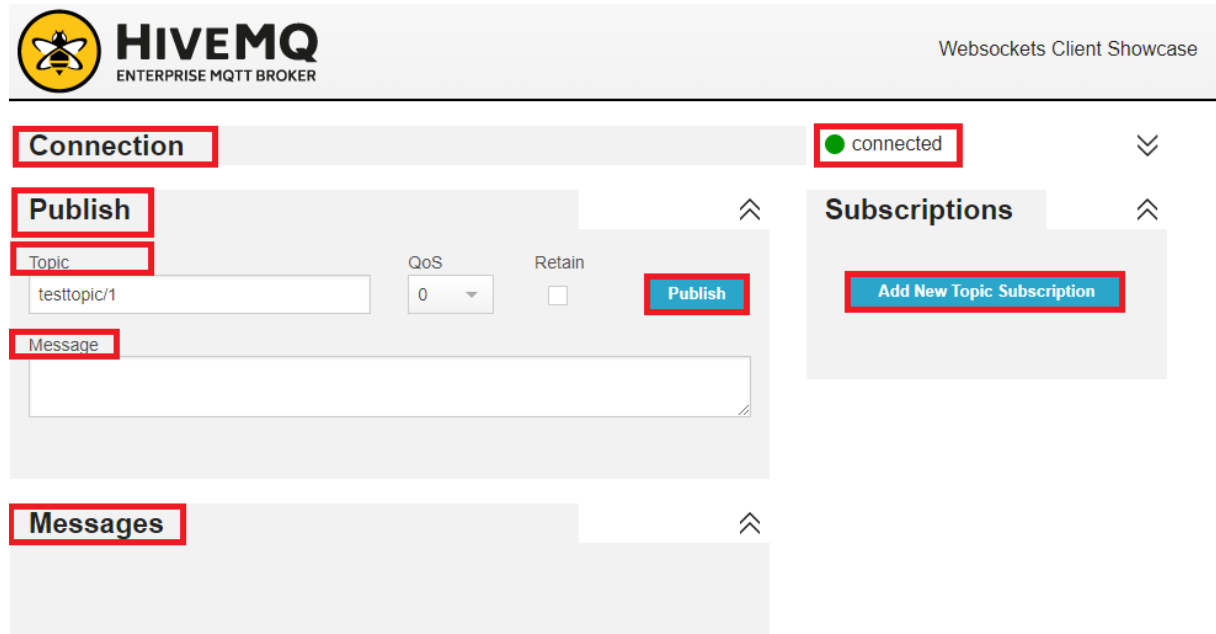
Örnekte **MQTT broker** genel sunucuya bağlanılarak **testtopic/1** yayınına abone olunmaktadır. Bu kanal genel kullanıma açık bir kanaldır. Sunucu üzerinde MQTT yayıncı-abone testleri yapılabilir. Aşağıdaki örnekteki komutla **testtopic** ve altındaki tüm konulara abone olunabilir.

```
pi@raspberrypi:~ $ mosquitto_sub -h broker.mqtttdashboard.com -t testtopic/#
```

İkinci örnekte ise **testtopic/1** konusunda genel kullanıma açık yayın yapılmaktadır.

```
pi@raspberrypi:~ $ mosquitto_pub -h broker.hivemq.com -t "testtopic/1" -m "ON"
```

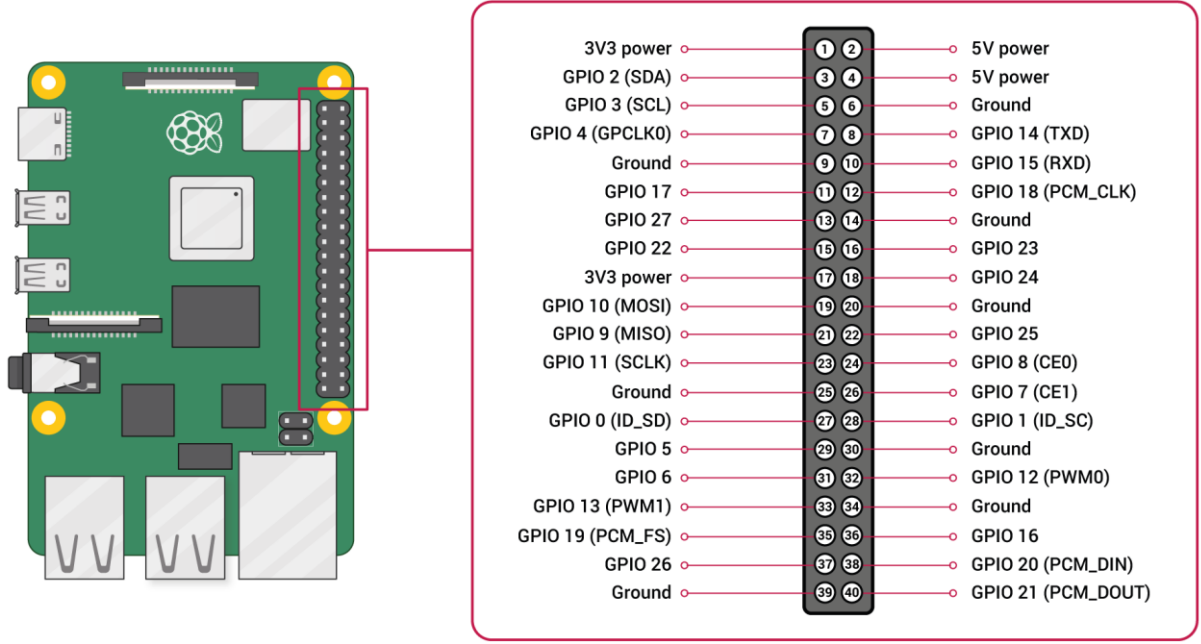
Raspberry Pi üzerinde genel deneme yayını (yukarıdaki örnekte çalıştırılan) **HiveMq** sunucusu (Şekil 9.3, <http://www.hivemq.com/demos/websocket-client/>) üzerinden de takip edilebilir. Sayfada **Add New Topic Subscription** butonuna tıklanarak konu (**topic**) yazılır. Takip edilen konuya ait mesajlar **Messages** bölümünde görünür. Aynı bölümden **Connection** bağlantı ayarlarını yapmak ve bağlanmak için kullanılır. **Publish** bölümü yayın yapmak için kullanılır. Bu bölümden konu adı (**topic**) değiştirilebilir. **Message** bölümünden ileti gönderilebilir. Bu değer, bir sensor değeri veya bir butona bağlı komut veya herhangi bir veri olabilir. Sunucu üzerinden hem yayın yapılabilir hem de bu yayınlara abone olunarak kanaldaki akış takip edilebilir.



Şekil 9.3. HiveMQ üzerinden yayın işlemleri

IoT Uygulamalarında kullanılacak pin dizilimi

Kitaptaki IoT uygulamalarında Raspberry Pi modeli ve programlamada **BCM** (Şekil 9.4) pin dizilimi kullanılmıştır.



Şekil 9.4. Raspberry Pi BCM pin dizilimi

(Kaynak: <https://www.raspberrypi.org/documentation/usage/gpio/>)

Uygulama -1

MQTT ile Buton Kontrolü

Bu uygulamada **MQTT** ile bir butona basılma durumu hem yayınlanmakta hem de abone olarak takip edilmektedir.

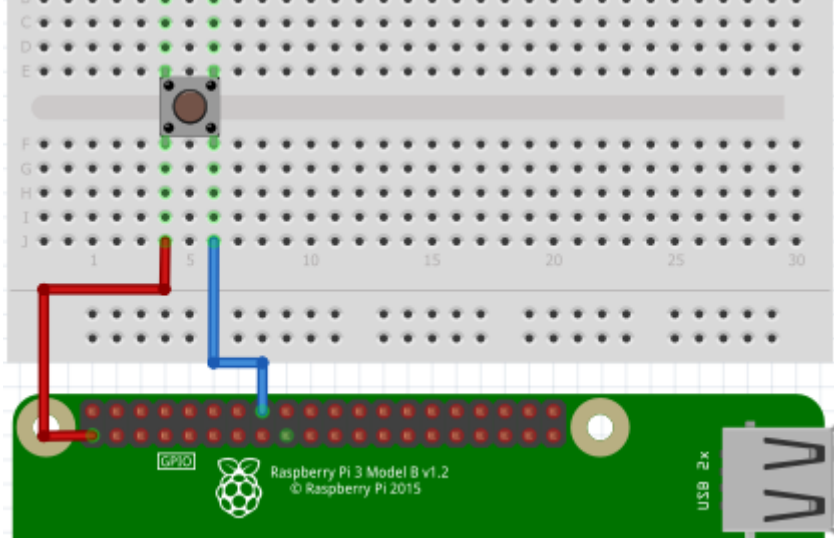
Python üzerinde **MQTT** protokolünü kullanabilmek amacıyla **paho-mqtt** kütüphanesi kullanılacaktır.

Raspberry Pi üzerine terminal kullanılarak **paho-mqtt** paketi kurulur. Paketi kurmak için

```
pip3 install paho-mqtt
```

paho-mqtt kurulumu ile daha fazla bilgiye proje sayfasında (<https://pypi.org/project/paho-mqtt/>) bulunmaktadır.

Devrenin Kurulması



Şekil 9.5. MQTT buton devresi

Kullanılan Malzemeler
Raspberry Pi
Bread board
Buton
Bağlantı kabloları

Devrede GPIO 23 nolu pine takılmış buton bulunmaktadır. Pin girişindeki dalgalanmaları önlemek amacıyla program içinde butonun bağlı olduğu pinin **pull down** direnci aktif edilmiştir.

Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır. Butona basıldığında 23 nolu pin girişi **HIGH**, butona basılı değilse **LOW** değerindedir. Programda butona basıldığında “Ev/buton” konusunda **1** mesajı yayınlanmaktadır. Buton basılı değilken **0** mesajı yayınlanmaktadır. Aynı program içinde hem yayın yapılmakta hem de bu yayına abone olunarak mesajlar okunmaktadır.

```
#Kullanılacak kütüphaneler dahil ediliyor.  
#Paho mqtt kütüphanesinin önceden kurulmuş olması gerekir.  
import paho.mqtt.client as paho  
from time import sleep  
import RPi.GPIO as GPIO  
# GPIO pinleri sıfırlanır.  
GPIO.cleanup()  
#Uyarılar devre dışı bırakılır.  
GPIO.setwarnings(False)  
#BCM pin dizilişi ayarlanır.  
GPIO.setmode(GPIO.BCM)  
#butonun bağladığı pin ayarlanır.  
buton=23  
#Kullanılan Raspberry Pi ip adresi değişkene atanmaktadır.  
#Hata almamak için cihazın ip adresi kontrol edilmelidir.  
broker="192.168.1.41"
```

```
#port ayarı yapılmaktadır.  
port=1883  
#buton giriş pini ve pull down olarak ayarlanır.  
GPIO.setup(buton,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
```

on_connect fonksiyonunun oluşturulması.

```
def on_connect(client1, userdata, flags, rc):  
    print("Bağlandı. Kod: "+str(rc))  
    # bu fonksiyon içinde abone olmak, bağlantı kesildiğinde veya yenilendiğinde  
    abonelikleri de yenilemeyi sağlar.  
    client1.subscribe("Ev/buton")  
    # testtopic/1 konusuna abone olundu.
```

Mesaj yayınlandığında çağrılacak olan on_message fonksiyonunu tanımlanır.

```
def on_message(client1, userdata, msg):  
    #Alınan mesaj utf-8 olarak formatlanır.  
    msg_str=str(msg.payload, "utf-8")  
    #Alınan mesaj yazdırılır.  
    print(msg.topic+" "+msg_str)
```

on_publish fonksiyonunun oluşturulması.

```
def on_publish(client1,userdata,result):  
    #Yayınlanan mesaj yazdırılır.  
    print("Veri yayınlandı. \n")  
    pass
```

Bir mqtt.Client nesnesi oluşturulur ve gerekli ayarlar yapılır.

```
#Yeni bir nesne türetilir.  
client1= paho.Client()  
#Nesnenin ilgili olayları fonksiyonlarla ilişkilendirilir  
client1.on_connect=on_connect  
client1.on_message = on_message  
client1.on_publish = on_publish  
#Bağlantı yapılır. Adres, port ve zaman aşımı parametrelerinin değerleri verilir.  
client1.connect(broker, port, 60)  
#Servis döngüsü başlatılır. Böylece sürekli olarak yayın alınması sağlanır.  
client1.loop_start()
```

Butona basıldığında **Ev/buton** konusunda **1** mesajı iletilir. Butona basılı değilken **0** mesajı iletilmektedir.

```
while True:  
    if GPIO.input(buton)==1:  
        client1.publish("Ev/buton",1)  
    elif GPIO.input(buton)==0:  
        client1.publish("Ev/buton",0)  
    # Butonu okumak için 0.1 saniye bekleme süresi eklenir.
```

sleep(0.1)

Uygulama -1'e ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/7bXO7>) bağlantısı kullanabilir.

Uygulamanın test edilmesi

Program çalıştırıldığında aşağıdaki çıktı alınır. Butona basılmadığında 0 yayınlanır. Butona basıldığında ise 1 yayınlanır. Uygulamada sadece butona basılma durumu kontrol edilmektedir.

Çıktı

Bağlandı. Kod: 0

Veri yayınlandı.

Ev/buton 0

Veri yayınlandı.

Ev/buton 1

Veri yayınlandı.

Uygulama -2

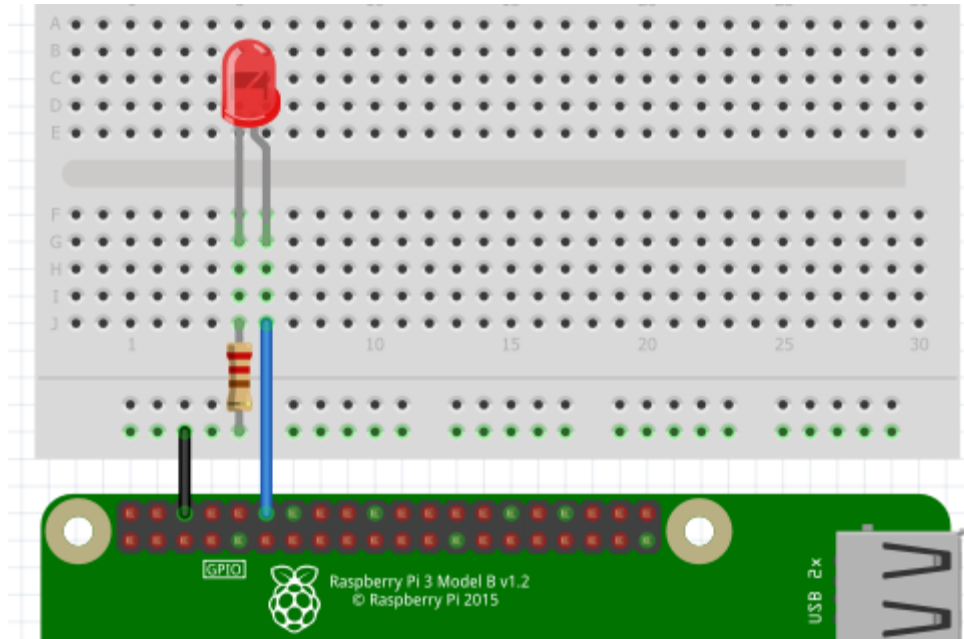
MQTT Broker Aracılığıyla Led Kontrolü

Bu uygulama ile internet üzerinden bir ledin kontrolü (açma/kapama) yapılmaktadır.

Not: **paho-mqtt** sisteme kurulmadıysa Uygulama-1'deki komutlar kullanılarak kurulum yapılmalıdır.

Devrenin Kurulması

MQTT üzerinden led kontrolü devre şeması Şekil 9.6'da gösterilmiştir.



Şekil 9.6. Led devresi

Kullanılan Malzemeler
Raspberry Pi
Bread board
Direnç 220 Ω
Led
Bağlantı kabloları

Devrede ledin anot ucu (BCM dizilişine göre) GPIO 18 nolu pine takılmıştır. Ledin katot ucuna 220 Ω değerinde bir direnç bağlanmıştır.

Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır.

```
#!/usr/bin/env python3
#Kullanılacak kütüphaneler dahil ediliyor.
#Paho mqtt kütüphanesinin önceden kurulmuş olması gerekir.
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
from time import sleep
#GPIO ile ilgili ayarlamalar
GPIO.setwarnings(False)
GPIO.cleanup()
led=18
GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)
```

on_connect fonksiyonunun oluşturulması.

```
def on_connect(client, userdata, flags, rc):
    print("Bağlandı. Kod:"+str(rc))
    client.subscribe("testtopic/1")
```

Mesaj yayınlandığında çağrılacak olan on_message fonksiyonunu tanımlanır.

```
def on_message(client, userdata, msg):
    msg_str=str(msg.payload, "utf-8")
    #Alınan mesaj yazdırılır.
    print(msg.topic+" "+msg_str)
    if msg_str=='ON':
        GPIO.output(led, GPIO.HIGH)
    elif msg_str=='OFF':
        GPIO.output(led, GPIO.LOW)
    #Mesaj ON ise Led açılır OFF ise kapatılır.
    #Burada ON, OFF yerine Aç, Kapa gibi kelimeler de kullanılabilir.
```

Bir mqtt.Client nesnesi türetilir ve gerekli ayarlar yapılır.

```
#Yeni bir nesne türetilir.
client = mqtt.Client()
#Nesnenin ilgili olayları fonksiyonlarla ilişkilendirilir.
client.on_connect = on_connect
```

```
client.on_message = on_message  
#Bağlantı yapılır. Adres, port ve zaman aşımı parametrelerinin değerleri verilir.  
client.connect("broker.hivemq.com", 1883, 60)  
#bağlantı, yayın ve abonelik işlemlerinin sürekli tekrarlanması sağlanır.  
client.loop_forever()
```

Uygulama -2'ye ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/bROWv>) bağlantısı kullanılabilir.

Uygulamanın test edilmesi

Yapılan uygulama yerel olarak terminalden test edilebilir. Ledi açmak için daha önce gösterilen aşağıdaki komut kullanılabilir.

```
pi@raspberrypi:~ $ mosquitto_pub -h broker.hivemq.com -t  
"testtopic/1" -m "ON"
```

Burada yine web sunucu üzerinden kontrol yapıldığı unutulmamalıdır. Ledi kapatmak için de aşağıdaki komut kullanılabilir.

```
pi@raspberrypi:~ $ mosquitto_pub -h broker.hivemq.com -t  
"testtopic/1" -m "OFF"
```

Çıktı

Bağlandı. Kod: 0

testtopic/1 ON

testtopic/1 OFF

Uygulamada, genel bir konuya (testtopic/1) abone olduğundan başka yayıncılar tarafından aynı konuda yapılan yayınlar da alınır.


Örnek:

testtopic/1 hi


testtopic/1 hello

testtopic/1 hello

Uygulama <http://www.hivemq.com/demos/websocket-client/> sayfasından da (Şekil 9.7) test edilebilir.

**HIVEMQ**
ENTERPRISE MQTT BROKER

Websockets Client Showcase

Connection ● disconnected 

Host
broker.mqttdashboard.com

Port
8000

ClientID
clientId-5SXccqU9cV

Connect

Username

Password

Keep Alive
60


Clean Session
☒


Last-Will Topic


Last-Will QoS
0

Last-Will Retain
☐

Last-Will Message


Publish 

Subscriptions 


Messages 


Şekil 9.7. HiveMQ ile uygulamanın test edilmesi

Connect butonuna tıklanarak bağlantı gerçekleştirilir. Bağlantı gerçekleştirildikten sonra **Publish** penceresinden (Şekil 9.8) gerekli ayarlar yapılır. Konu (**Topic**) gerekirse değiştirilebilir. Yukarıdaki koda **client.subscribe("testtopic/1")** satırında **Topic testtopic/1** olarak ayarlanmıştır.

**HIVEMQ**
ENTERPRISE MQTT BROKER

Websockets Client Showcase

Connection ● connected 

Publish 


Topic
testtopic/1

QoS
0


Retain
☐

Publish

Message
ON

Subscriptions 

Add New Topic Subscription

Messages 

Şekil 9.8. HiveMQ ile uygulamanın test edilmesi-2

Message kısmına Python kodunda kontrol edilen **ON** iletisi yazılarak (Şekil 9.8.) **Publish** butonu tıklandığında, kurulan devredeki led yanacaktır. Message bölümüne “OFF” yazıp **Publish** butonu tıklandığında led sönecektir. Bu siteye Raspberry Pi dışındaki başka bir cihazdan da bağlanarak kontrolleri yapılabilir. Web sunucu kanallarının (özelleştirilmedikçe) genel kullanıma açık olduğu unutulmamalıdır. Başka bir yayıncı aynı kanaldan yayın yapabilir ve bilmeyerek de olsa cihazları kontrol edebilir.

Uygulama -3

Akıllı cihaz ile MQTT broker aracılığıyla led kontrolü

Bu uygulamada bir led devresi (Uygulama-2’de oluşturulan) akıllı cihazdaki MQTT uygulaması (Android cihazda) ile kontrol edilecektir. Bu yöntemle Raspberry Pi üzerine bağlı bir cihaz (bu örnekte bir led) akıllı cihaz aracılığıyla MQTT broker aracılığıyla kontrol edilebilmektedir.

Devrenin Kurulması

Uygulama -2’deki devre kullanılmıştır. Devrede Raspberry Pi GPIO 18 pinine bağlı bir led bulunmaktadır.

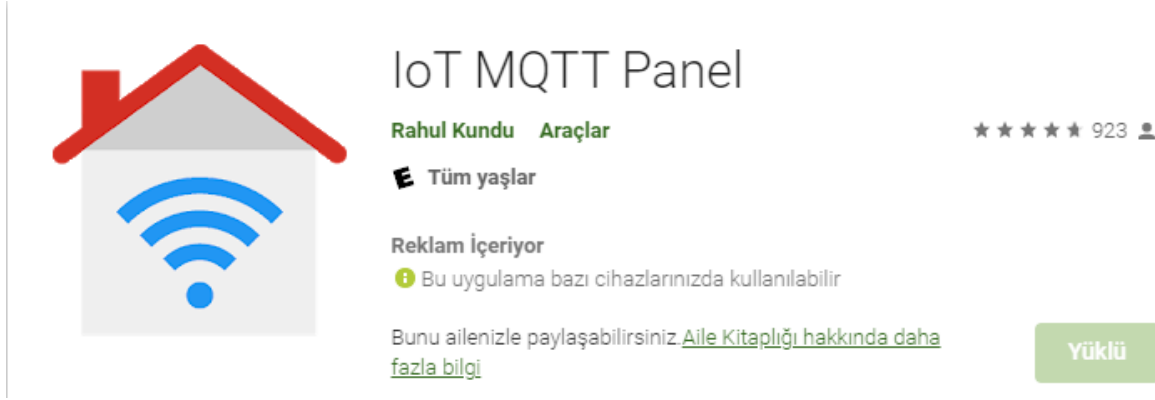
Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır.

```
#!/usr/bin/env python3
import paho.mqtt.client as mqtt
import RPi.GPIO as GPIO
from time import sleep
GPIO.setwarnings(False)
GPIO.cleanup()
led=18
GPIO.setmode(GPIO.BCM)
GPIO.setup(led, GPIO.OUT)
def on_connect(client, userdata, flags, rc):
    print("Connected with result code "+str(rc))
    client.subscribe("testtopic/1")
def on_message(client, userdata, msg):
    msg_str=str(msg.payload, "utf-8")
    print(msg.topic+" "+msg_str)
    #print (type(msg.payload))
    if msg_str=='ON':
        GPIO.output(led, GPIO.HIGH)
    elif msg_str=='OFF':
        GPIO.output(led, GPIO.LOW)
client = mqtt.Client()
client.on_connect = on_connect
client.on_message = on_message
client.connect("broker.hivemq.com", 1883, 60)
client.loop_forever()
```

Akıllı cihaza uygulama kurulması

Raspberry Pi üzerinden ledi kontrol etmek için Android bir cihaza **IoT MQTT Panel** adlı (Şekil 9.9) bir uygulama yüklenmiştir. Bu uygulama ile akıllı cihaz ile internet üzerinden led kontrol edilebilmektedir. Play Store’da bu uygulamaya benzer başka uygulamalar da bulunmaktadır istenirse bu uygulamalar da kullanılabilir.



Şekil 9.9. Iot MQTT panel uygulaması

Uygulama akıllı cihaza yüklendikten sonra bazı ayarlar yapılması gerekmektedir.

1. **Connections** bölümünde **Add Connection** (+ butonuna) tıklanarak yeni bir bağlantı eklenir.
2. Bağlantı bilgileri Şekil 9.10'da olduğu gibi doldurulur. Bağlantı adı (Connection name) ve istemci kimliği (Client Id) değiştirilebilir. Eğer gerekiyorsa gelişmiş ayarlar (Additional options) bölümünde kullanıcı adı ve şifre bilgileri de girilebilir. Bu bilgiler girildikten sonra **Create** butonu tıklanarak yeni bir bağlantı oluşturulur.

Şekil 9.10. Yeni bağlantı ekleme

3. Bağlantı oluşturulduktan sonra kontrolleri gerçekleştirmek ve izlemek için paneller eklenir. Panel eklemek için + butonu tıklanır. Açılan pencereden sırayla **Led indicator**, **Text input** ve **switch** seçilir.

Led indicator eklenirken ayarlar Şekil 9.11'deki gibi olmalıdır. Ayarlarda sırayla panel adı, konu, on mesajı ve off mesajı yer almaktadır. Python kodunda konu **testtopic/1** ve açma kapama için **ON** ve **OFF** mesajları kullanıldığı için bilgiler bu şekilde doldurulmuştur.

Panel name *

Led gösterge

Topic *

testtopic/1

Payload on *

ON

Payload off *

OFF

LED on icon

LED color

#DF0000

LED off icon

LED color

#9E9E9E

☐ Enable notification

☐ Payload is JSON Data

☐ Show received timestamp

QoS

0

Şekil 9.11. Led indicator ayarları

Ayarlar yapıldıktan sonra **text input** eklenerek Şekil 9.12'deki gibi ayarlanır.

Panel name *

Msg

Topic *

testtopic/1

☐ Show sent timestamp

☐ Confirm before publish

☐ Payload is JSON Data

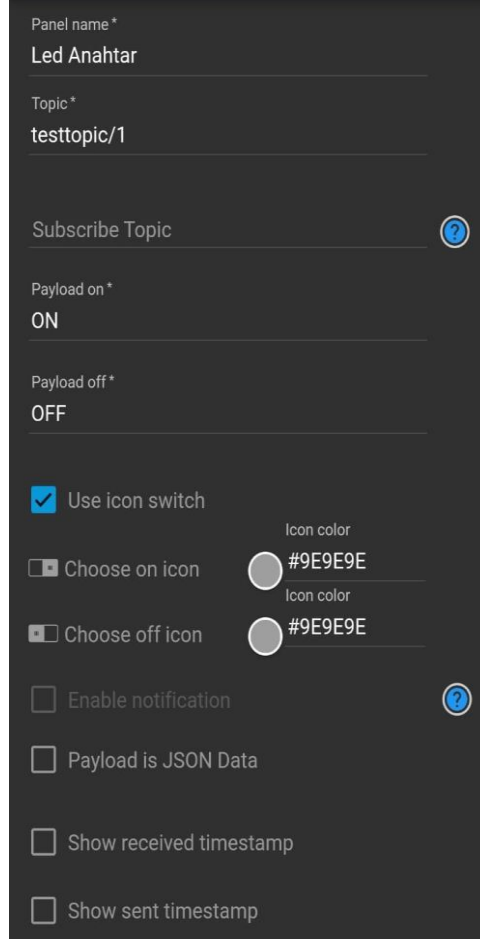
☐ Clear text on publish

QoS

0

Şekil 9.12. Text input ayarları

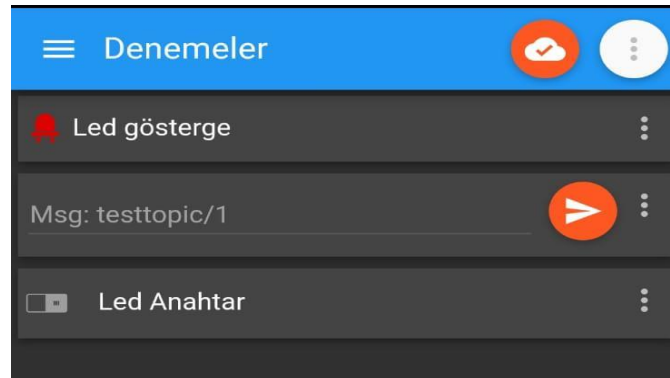
Text input ekledikten sonra **switch paneli** eklenerek Şekil 9.13'deki gibi ayarlanır. Konu (**topic**) Python kodundaki ile aynı olmalıdır.



The image shows a configuration screen for a 'Switch' panel. The 'Panel name' is 'Led Anahtar' and the 'Topic' is 'testtopic/1'. There are fields for 'Payload on' (ON) and 'Payload off' (OFF). A checkbox 'Use icon switch' is checked. Below it, there are options to 'Choose on icon' and 'Choose off icon', both with a color picker set to '#9E9E9E'. There are also checkboxes for 'Enable notification', 'Payload is JSON Data', 'Show received timestamp', and 'Show sent timestamp'.

Şekil 9.13. Switch ayarları

4. Paneller eklendiğinde kontrol ekranı Şekil 9.14'deki gibi görünecektir.



Şekil 9.14 Kontrol panellerinin görünümü

Uygulamanın test edilmesi

Uygulamanın test edilmesi için Python kodları çalıştırılır. Böylece Raspberry Pi ile **testtopic/1** konulu yayın takip edilir. **ON** mesajı alındığında led yanacaktır. **OFF** mesajı alındığında ise led sönecektir. Android cihaz üzerinde uygulama çalıştırılır. Led anahtar tıklanarak **ON** ve **OFF** mesajları broker aracılığıyla gönderilir ve ledin yanıp sönmesi sağlanır. Led göstergede ise ledin durumuna göre ikon görünecektir. **Msg:testtopic/1** bölümüne **On** veya **OFF** yazılarak da led kontrolü yapılabilir.

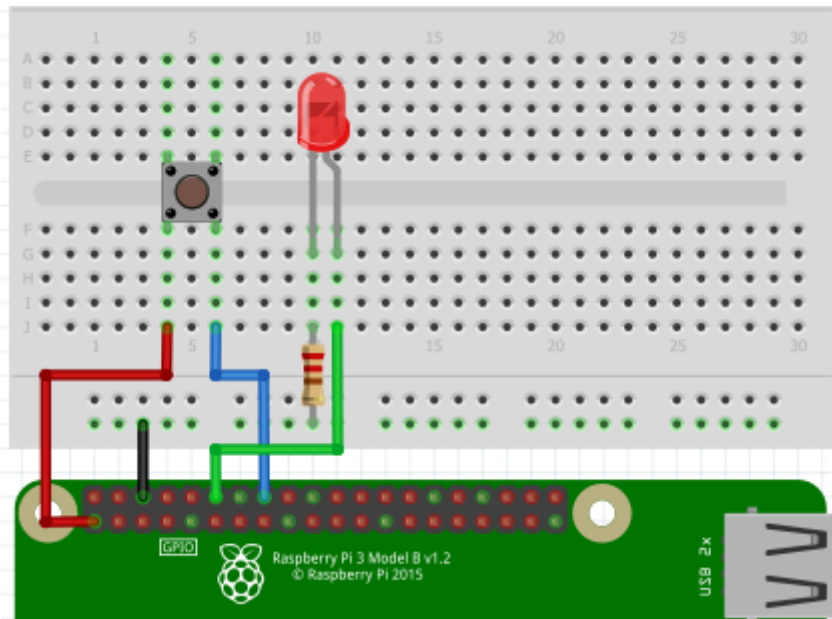
Uygulama -4

Yerel Makinede MQTT ile Buton-Led kontrolü

Bu uygulama ile MQTT üzerinden bir ledin buton ile kontrolü (açma/kapama) yapılmaktadır. Bu uygulamada broker olarak yerel makine tanımlanmıştır. Bu uygulamada normal buton led uygulamasından farklı olarak buton basılma durumu MQTT üzerinden yayınlamakta ve bu yayına abone olunmaktadır. Broker değiştirilerek (koddaki yorum satırı kullanılabilir) kontrol internet üzerinden de kontrol gerçekleştirilebilir.

Devrenin Kurulması

Buton ile led kontrolü devre şeması Şekil 9.15’de gösterilmiştir.



Şekil 9.15. Buton led devresi

Kullanılan Malzemeler
Raspberry Pi
Bread board
Direnç 220 Ω
Buton
Led

Bağlantı kabloları

Devrede ledin anot ucu GPIO 18 nolu pine takılmıştır. Ledin katot ucuna 220 Ω değerinde bir direnç bağlanmıştır. Buton ise GPIO 23 nolu pine takılmıştır.

Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır. Programa öncelikle gerekli kütüphaneler dahil edilir. Programda **broker="192.168.1.41"** pasif hale getirilerek **#broker= "broker.hivemq.com"** satırı aktif hale getirilirse kontrol işlemi broker aracılığıyla web üzerinden gerçekleştirilebilir.

```
import paho.mqtt.client as paho
from time import sleep
import RPi.GPIO as GPIO
buton=23
led=18
# Hata almamak için cihazın ip adresi kontrol edilmelidir.
broker="192.168.1.41"
#broker= "broker.hivemq.com"
port=1883
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
GPIO.setup(buton,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(led, GPIO.OUT)
```

MQTT servisiyle ilgili fonksiyonlar (On_connect on_mesage ve on_publish) tanımlanır. Uygulama -1'den farklı olarak butona basılma yayını alındığında led yanacaktır.

```
def on_connect(client1, userdata, flags, rc):
    print("Bağlandı kod: "+str(rc))
    client1.subscribe("Ev/buton")
def on_message(client1, userdata, msg):
    msg_str=str(msg.payload, "utf-8")
    print(msg.topic+" "+msg_str)
    #Butona basılma mesajı alındıysa led yanar.
    if int(msg_str)==1:
        GPIO.output(led, GPIO.HIGH)
    elif int(msg_str)==0:
        GPIO.output(led, GPIO.LOW)
def on_publish(client1,userdata,result):
    print("Veri yayınlandı. \n")
```

Bir mqtt.Client nesnesi türetilir ve gerekli ayarlar yapılır.

```
client1= paho.Client()
client1.on_publish = on_publish
client1.on_connect=on_connect
client1.on_message = on_message
client1.connect(broker, port, 60)
```

```
client1.loop_start()
while True:
    client1.subscribe("Ev/buton")
    if GPIO.input(buton)==1:
        client1.publish("Ev/buton",1)
    elif GPIO.input(buton)==0:
        client1.publish("Ev/buton",0)
    sleep(0.1)
```

Uygulama-4'e ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/xnuv6>) bağlantısı kullanabilir.

Uygulamanın test edilmesi

Uygulama çalıştırıldığında butona basıldığı sürece led ışık verecektir. Aynı zamanda butona basılma durumu ekrana yazdırılmaktadır.

Veri yayınlandı.

Bağlandı kod: 0

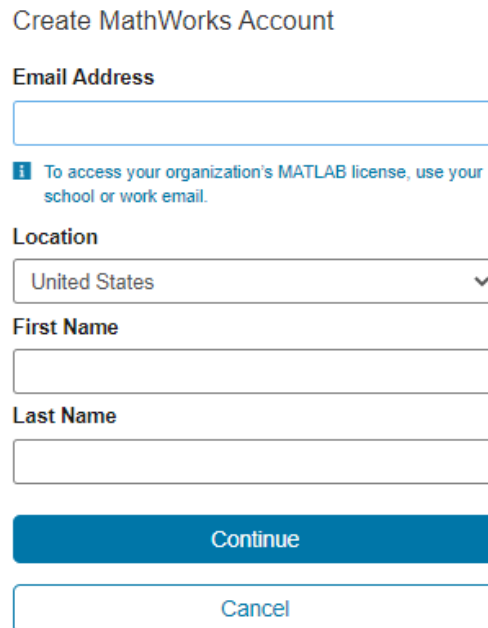
Veri yayınlandı.

Ev/buton 1

Veri yayınlandı.

Raspberry Pi ile Thing Speak üzerinden IoT uygulamaları

Thing Speak IoT analitik platform hizmetidir. Sensorler ve cihazlar tarafından gönderilen verinin anında görselleştirilmesini sağlar. Bu uygulama için **Thing Spek'de** bir hesap açılarak sensor verisini yayınlanacaktır. <https://thingspeak.com/> adresine girilerek daha önce hesap açılmamışsa **Create one!** bağlantısına tıklanarak gelen ekranda (Şekil 9.16) hesap bilgileri girilerek yeni bir hesap oluşturulur.

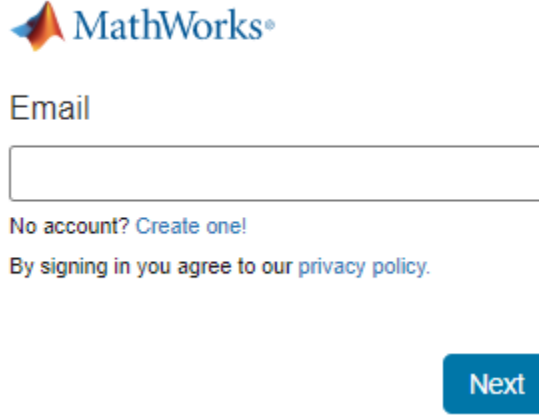


The image shows a web form titled "Create MathWorks Account". It contains the following fields and elements:

- Email Address:** A text input field.
- Location:** A dropdown menu with "United States" selected.
- First Name:** A text input field.
- Last Name:** A text input field.
- Buttons:** Two buttons at the bottom, "Continue" (blue) and "Cancel" (white with blue border).
- Help Text:** A small blue icon followed by the text: "To access your organization's MATLAB license, use your school or work email."

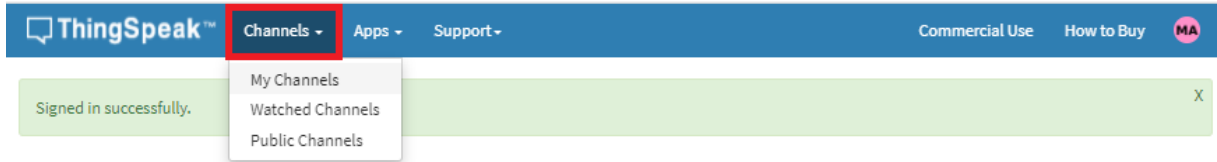
Şekil 9.16. Thing Speak hesap oluşturma

Hesap oluşturulurken e-posta adresine bir doğrulama epostası gelecektir. Doğrulama işlemi gerçekleştirildikten sonra e-posta adresi girilerek **Next** butonuna (Şekil 9.17) tıklanır ve sonra parola girilerek servis kullanılabilir.

The image shows the MathWorks login page. At the top is the MathWorks logo. Below it is a text input field labeled "Email". Under the input field, there is a link "No account? Create one!" and a statement "By signing in you agree to our privacy policy." At the bottom right is a blue button labeled "Next".

Şekil 9.17. Oturum açma ekranı

Hesap açıldıktan sonra giriş yapılır. Kanallar (**Channels**) menüsü altında (Şekil 9.18) **My Channels**, **Watched Channels** ve **Public Channels** menüleri (kanallarım, izlenen kanallar ve genel kanallar) bulunur.



Şekil 9.18. Servis işlemleri

My Channels altında yayın yapmak için **New Channel** seçilir ve yeni bir kanal eklenebilir. Kanalın altında 8 adet alan (Şekil 9.19) eklenebilmektedir.

Name	<input type="text" value="Sensor"/>	
Description	<input type="text"/>	
Field 1	<input type="text" value="Sicaklik"/>	<input checked="" type="checkbox"/>
Field 2	<input type="text" value="Nem"/>	<input checked="" type="checkbox"/>
Field 3	<input type="text" value="PIR"/>	<input checked="" type="checkbox"/>
Field 4	<input type="text" value="Field Label 4"/>	<input checked="" type="checkbox"/>
Field 5	<input type="text" value="Field Label 5"/>	<input checked="" type="checkbox"/>
Field 6	<input type="text"/>	<input type="checkbox"/>
Field 7	<input type="text"/>	<input type="checkbox"/>
Field 8	<input type="text"/>	<input type="checkbox"/>
Metadata	<input type="text"/>	
Tags	<input type="text"/>	
	<small>(Tags are comma separated)</small>	

Şekil 9.19. Kanal işlemleri

Name (kanalın adı) ve **Field 1 -Field 8** kanal altındaki konular içindir. Bu bölümlerden her biri bir sensöre, butona, düğmeye vb. karşılık kullanılabilir. Formdaki alanlar (Şekil 9.19) doldurulur ve **Save Channel** butonuna tıklanarak bir kanal oluşturulur.

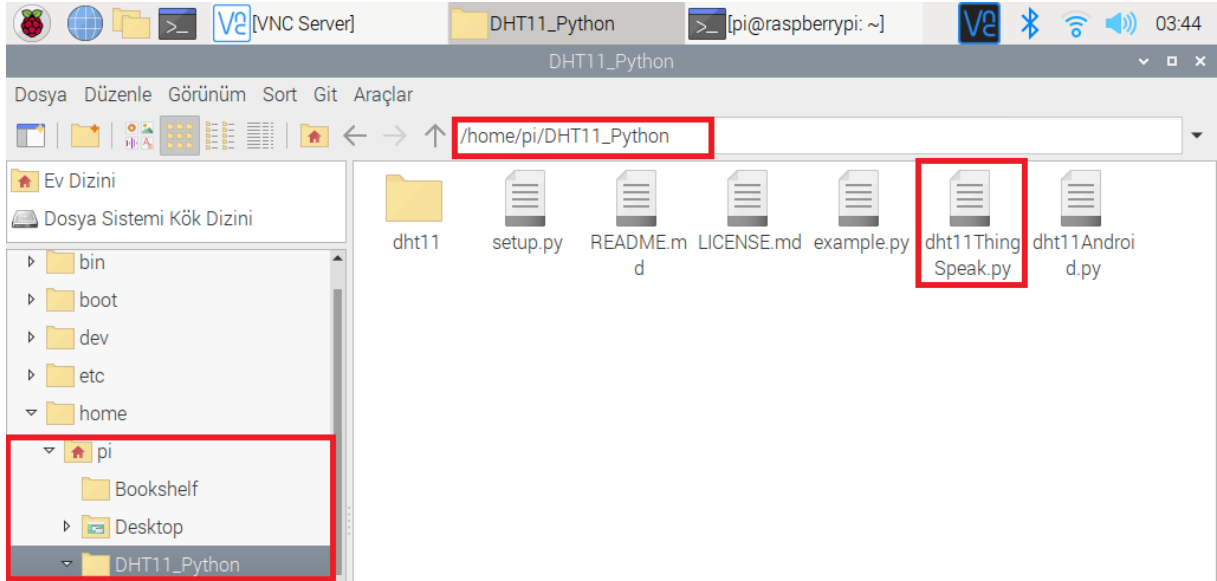
Uygulama -5

Sensör Değerlerini Web Üzerinden Yayınlamak (Thing Speak)

Bu uygulamada **DHT11** sıcaklık ve nem sensöründen alınan veri Thing Speak platformu üzerinde yayınlanacaktır. Daha önceki uygulamalarda yüklenmediyse (pi dizini içinde bulunmuyorsa) terminal açılarak aşağıdaki kod yazılır.

```
git clone https://github.com/szazo/DHT11_Python
```

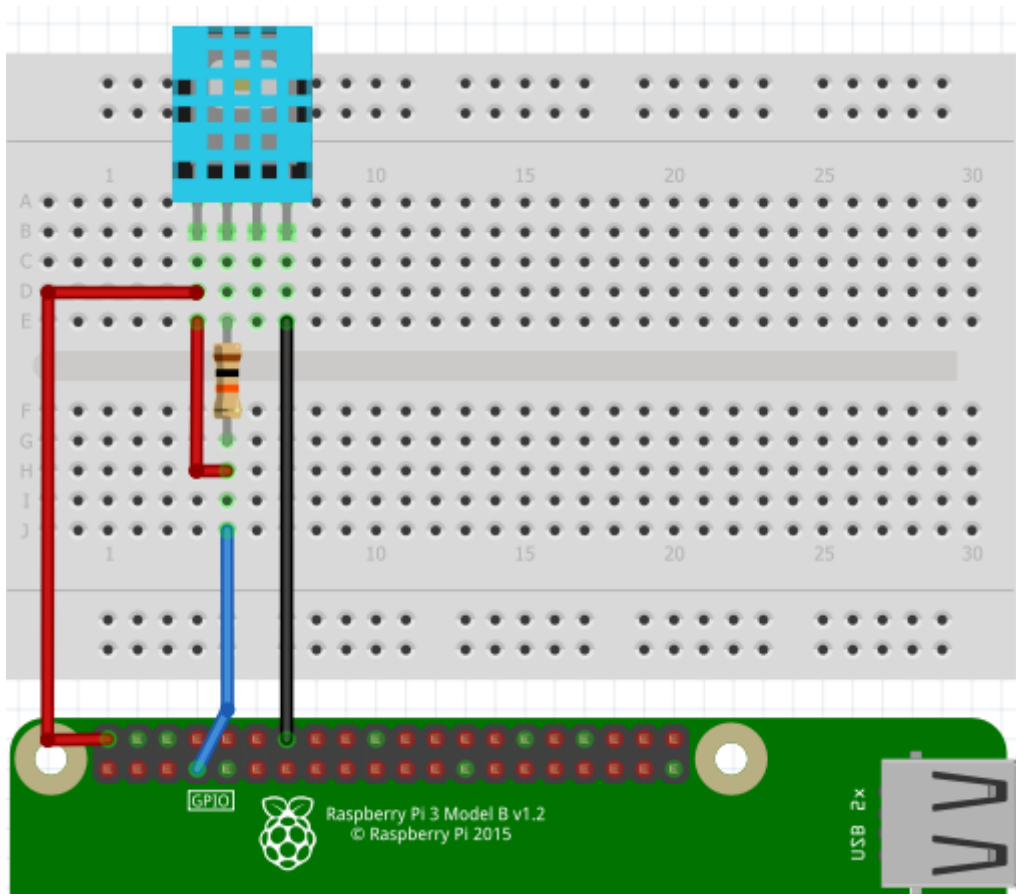
Komut çalıştırıldıktan sonra pi klasörü içerisine **DHT11_Python** isimli klasör (Şekil 9.20) oluşacaktır. DHT uygulamaları için oluşturulacak Python dosyalarının bu klasör içerisine kaydedilmesi gerekmektedir.



Şekil 9.20. DHT11_Python kitaplığının kopyalanması

Devrenin Kurulması

Uygulamanın devre şeması Şekil 9.21’de gösterilmiştir.



Şekil 9.21. DHT11 devresi

Kullanılan malzemeler
Raspberry Pi
Bread Board
1 Adet DHT11
Direnç 10 K Ω
Bağlantı kabloları

Devrede **DHT11**, **GND** (siyah kablo), **VCC** (kırmızı kablo) bağlantıları yapılmıştır. **DHT11** sinyal çıkışı GPIO 4 pinine bağlanmıştır. Sinyal piniyle **VCC** pini arasında **pull up** direnci (10 K Ω) bağlanmıştır.

Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır. Kodda ********* şifrelenen bölüme Thing Speak platformu üzerinde kanala ait api key girilmelidir. Bu api key kullanıcıya özeldir ve başkası ile paylaşılmamalıdır. Thing Speak üzerinde api key (Şekil 9.21) menüsü açılarak api key kopyalanarak kodda ilgili bölüme eklenir. Programda veri gönderildiği için **Write Api Key** kullanılmıştır.



ThingSpeak™ Channels Apps Support Commercial Use How to Buy MA

Sensor

Channel ID: 1306992
Author: mwa0000020012155
Access: Private



Private View Public View Channel Settings Sharing **API Keys** Data Import / Export

Write API Key

Key  

Generate New Write API Key

Read API Keys

Key  

Note

Save Note Delete API Key

Add New Read API Key

Help

API keys enable you to write data to a channel or read data from a private channel. API keys are auto-generated when you create a new channel.

API Keys Settings

- Write API Key:** Use this key to write data to a channel. If you feel your key has been compromised, click **Generate New Write API Key**.
- Read API Keys:** Use this key to allow other people to view your private channel feeds and charts. Click **Generate New Read API Key** to generate an additional read key for the channel.
- Note:** Use this field to enter information about channel read keys. For example, add notes to keep track of users with access to your channel.

API Requests

Write a Channel Feed

GET `https://api.thingspeak.com/update?api_key=TRFZ4T5BRMH74MNH&field=`

Read a Channel Feed

GET `https://api.thingspeak.com/channels/1306992/feeds.json?api_key=`

Read a Channel Field

GET `https://api.thingspeak.com/channels/1306992/fields/1.json?api_key=`

Şekil 9.21. Thing Speak api key

```
import RPi.GPIO as GPIO
from time import sleep
import dht11
```

```
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
sicaklikNem=dht11.DHT11(pin=4)
import urllib.request
URL='https://api.thingspeak.com/update?api_key=*****&field1=0'
KEY= '*****'
```

Sıcaklık ve nem değerlerini Thing Speak'e gönderen fonksiyon tanımlanır.

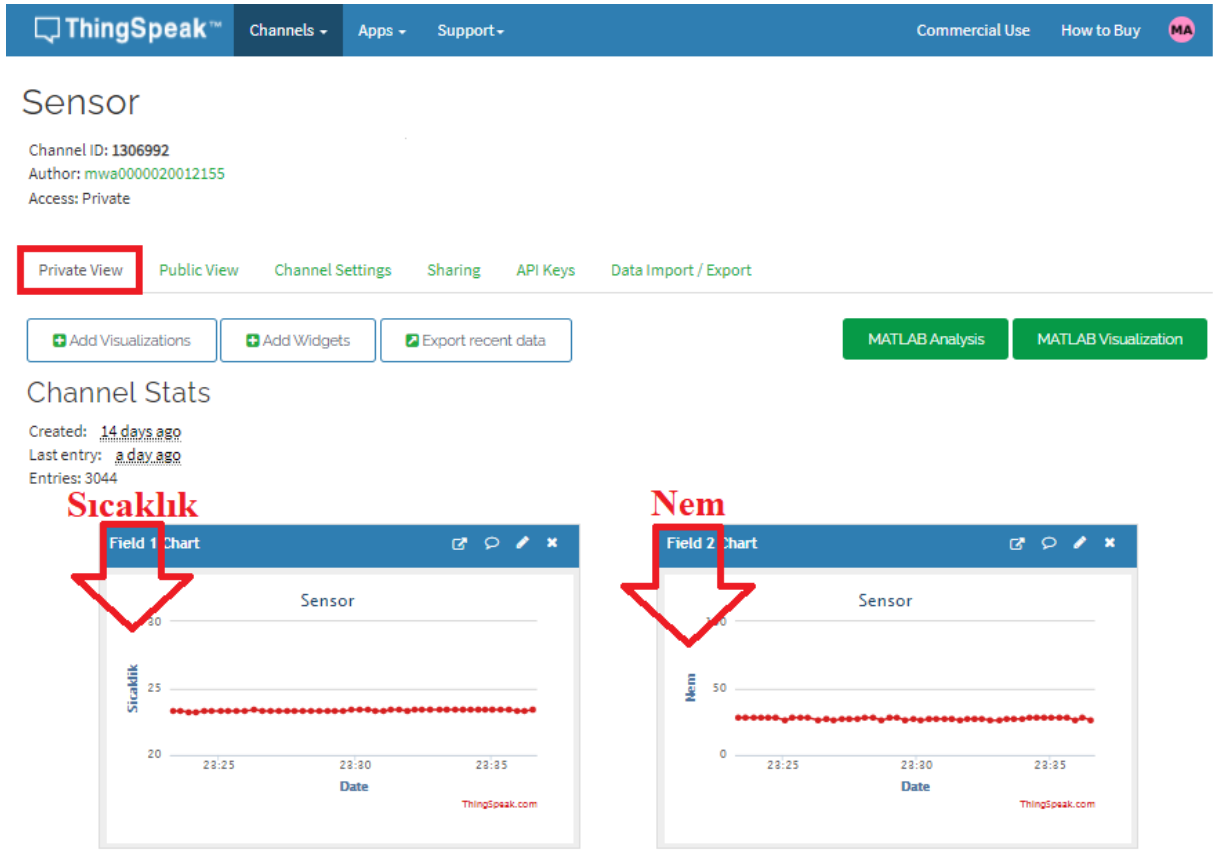
```
def thingspeak_post(sicaklik, nem):
    HEADER='&field1={}&field2={}&field3'.format(sicaklik,nem)
    NEW_URL = URL+KEY+HEADER
    data=urllib.request.urlopen(NEW_URL)
```

DHT11 sensöründen okunan değerler 15 saniye aralıklarla Thing Speak'e gönderilir. Thing Speak ücretsiz versiyonu en az 15 saniye aralıklarla veri güncelleme ve görselleştirmeye olanak vermektedir.

```
While True:
    deger=sicaklikNem.read()
    if deger.is_valid():
        print (deger.temperature,deger.humidity )
        thingspeak_post(deger.temperature,deger.humidity )
        sleep(15)
```

Uygulama-5'e ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/UUhxlq>) bağlantısı kullanılabilir.

Dosya editörde veya terminalde çalıştırıldığında ThingSpeak üzerinde **Channels** altında **My Channels** menüsünden **Private View** sekmesinde sıcaklık ve nem değerlerine ait grafiklere (Şekil 9.22) ulaşılabilir.



Şekil 9.22. Thing Speak veri görselleştirme

Uygulama -6

Web üzerinden Led Kontrolü (Thing Speak)

Bu uygulamada Thing Speak üzerinden led kontrolü yapılmaktadır. Thing Speak üzerindeki kanalın altında **field4** alanı tanımlanarak led kontrolü için kullanılmaktadır. Tarayıcıya aşağıdaki adres yazıldığında (yıldızlı bölüme **Write Api Key** girilerek) **field4** için **0** verisi yayınlanır.

https://api.thingspeak.com/update?api_key=*****&field4=0

field4 üzerindeki veri devre üzerindeki ledi kontrol etmek için kullanılmaktadır. Thing Speak üzerinde son yayınlanan veri okunur **1** ise led yanacaktır. Son veri **1** ise led sönecektir. Ledi yakmak için aşağıdaki adres satırı kullanılır.

https://api.thingspeak.com/update?api_key=*****&field4=1

Devrenin Kurulması

Devrede 18 nolu pine bir led (uygulama-2 ile aynı) bağlanmıştır.

Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır. Programa öncelikle gerekli kütüphaneler dahil edilir. Programda Thing Speak üzerinden sadece okuma yapıldığı için **Read Api Key** kullanılmıştır. Kodlardaki **0** ve **1** değerleri ledi kontrol

etmek için kullanılan değerlerdir. Bu değerlerin yerine farklı değerler de kullanılabilir. Farklı değerler kullanılırsa Python kodlarında da değiştirilmelidir.

```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import urllib.request
import json
from time import sleep
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
led=18
GPIO.setup(led, GPIO.OUT)
#Thing Speak read api key ve kanal id değerleri değişkenlere atanır.
READ_API_KEY='*****'
CHANNEL_ID=123456
```

Bağlantı sürekli olarak tekrarlanır ve Thing Speak üzerinde kanaldaki **field4**'deki son değer alınarak kontrol edilir. Alınan değer **0** ise led söner değer **1** ise led yanacaktır.

```
while True:
    conn = urllib.request.urlopen("http://api.thingspeak.com/channels/%s/feeds/last.json?api_key=%s" \
    % (CHANNEL_ID,READ_API_KEY))
    response = conn.read()
    print ("http durum kodu=%s" % (conn.getcode()))
    data=json.loads(response)
    print ("değer:", data['field4'], "oluşturulma:", data['created_at'])
    if int(data['field4'])==1:
        GPIO.output(led, GPIO.HIGH)
    elif int(data['field4'])==0:
        GPIO.output(led, GPIO.LOW)
    conn.close()
```

Uygulama-6'ya ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/2vrbg>) bağlantısı kullanılabilir.

Uygulamanın test edilmesi

Uygulamayı test etmek için tarayıcı açılır ve aşağıdaki adres kullanılır. Aşağıdaki adreste yıldızlar yerine **Write Api Key** yazılmalıdır. **field4=1** olduğunda Python koduna uygun olarak led yanacaktır.

https://api.thingspeak.com/update?api_key=*****&field4=1

Ledi söndürmek için ise aynı adreste **field4=0** olmalıdır. Aşağıdaki adres kullanıldığında led sönecektir.

https://api.thingspeak.com/update?api_key=*****&field4=0

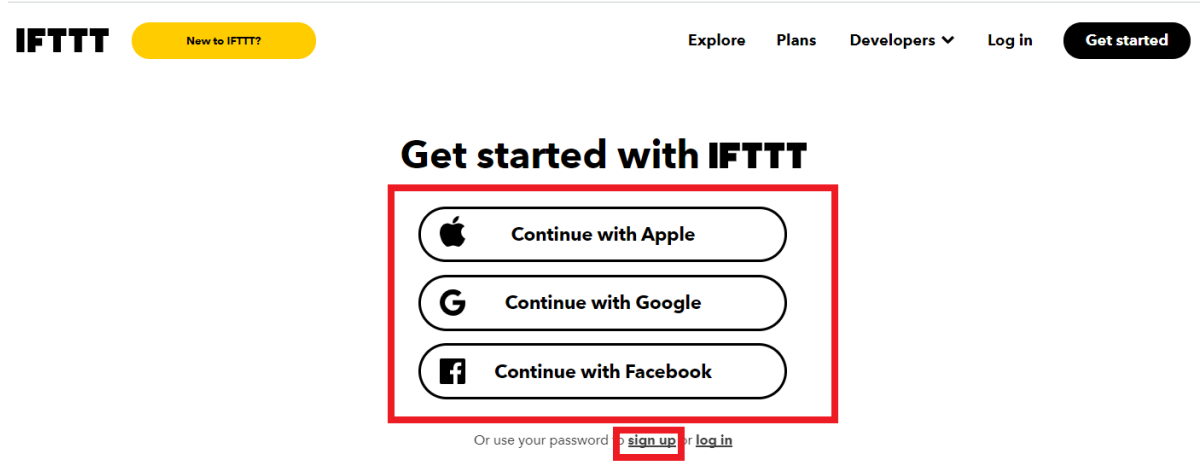
Thing Speak'te ücretsiz sürümden dolayı komutlar (field4'e değer verme) arasında 15 saniye olmalıdır. **Thing Speak** led kontrolü (field4) sonraki uygulamalarda da kullanılacaktır.

Uygulama -7

Raspberry Pi ile IFTTT üzerinden IoT uygulaması (Google Asistan ile led kontrolü)

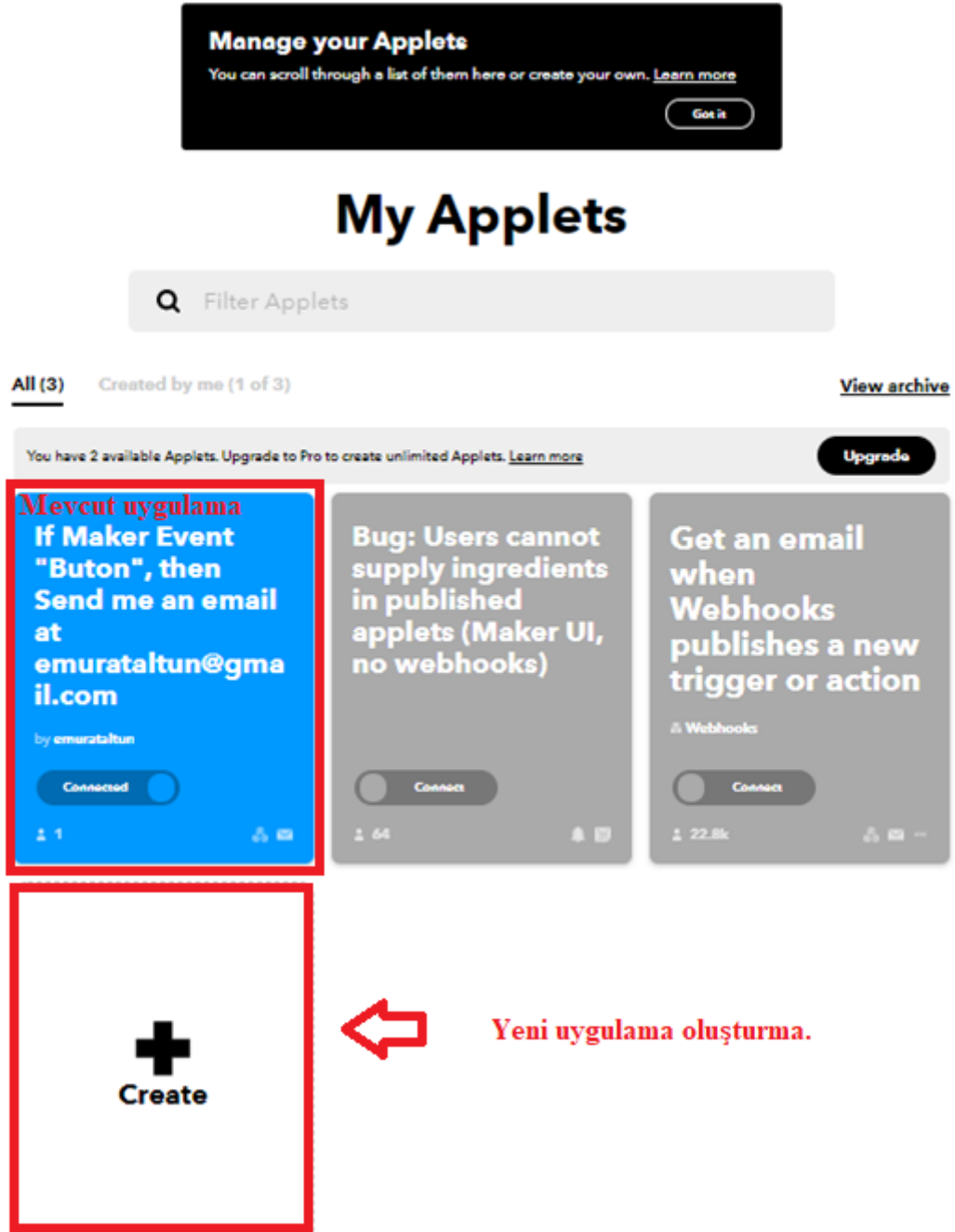
Bu uygulamada IFTT ile Google Asistan üzerinden Thing Speak aracılığıyla Raspberry Pi üzerindeki led kontrol edilmektedir (<https://www.mathworks.com/help/thingspeak/google-ifttt-thingspeak-lamp.html>).

IFTTT, farklı uygulamaları ve farklı cihazları birbirine bağlamak için ücretsiz servis sağlayan bir platformdur. Google, Apple vb. servislerin hesap bilgileri ile veya yeni bir hesap oluşturarak IFTTT (Şekil 9.23) kullanılabilir. Kullanıcılar durumlarına uygun seçeneği kullanarak oturum açarak hizmetlere erişebilirler.



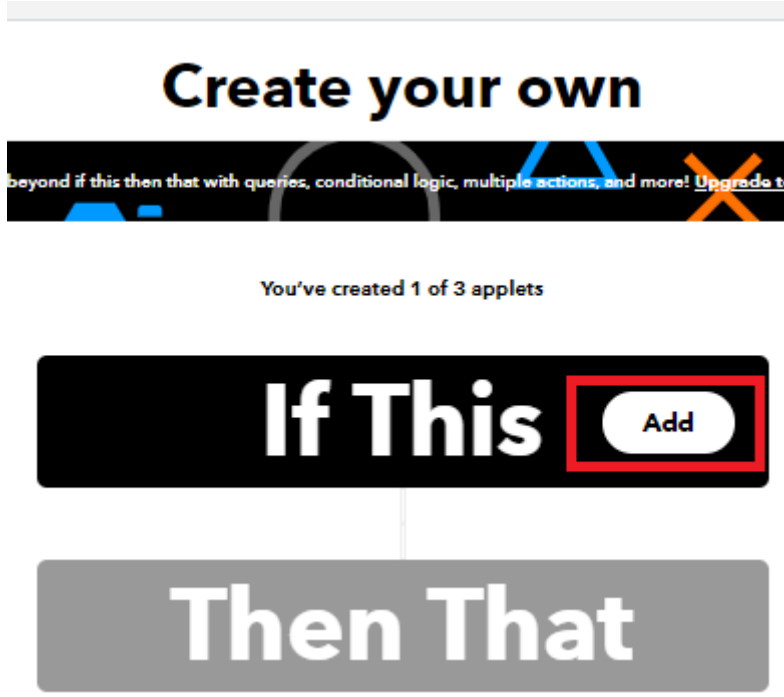
Şekil 9.23. IFTTT hesap işlemleri

Uygulamada Google hesabı ile devam edilmiştir. Google hesabı ile devam edildiğinde IFTTT, geçerli bir hesap seçilmesini istenmektedir. Hesap seçildikten sonra Google hesabıyla IFTTT servislerine erişilebilir. Uygulamaların kısmından (Şekil 9.24) **Create** (+) kullanılarak yeni bir uygulama oluşturmaya başlanabilir.



řekil 9.24. IFTTT yeni uygulama oluřturma

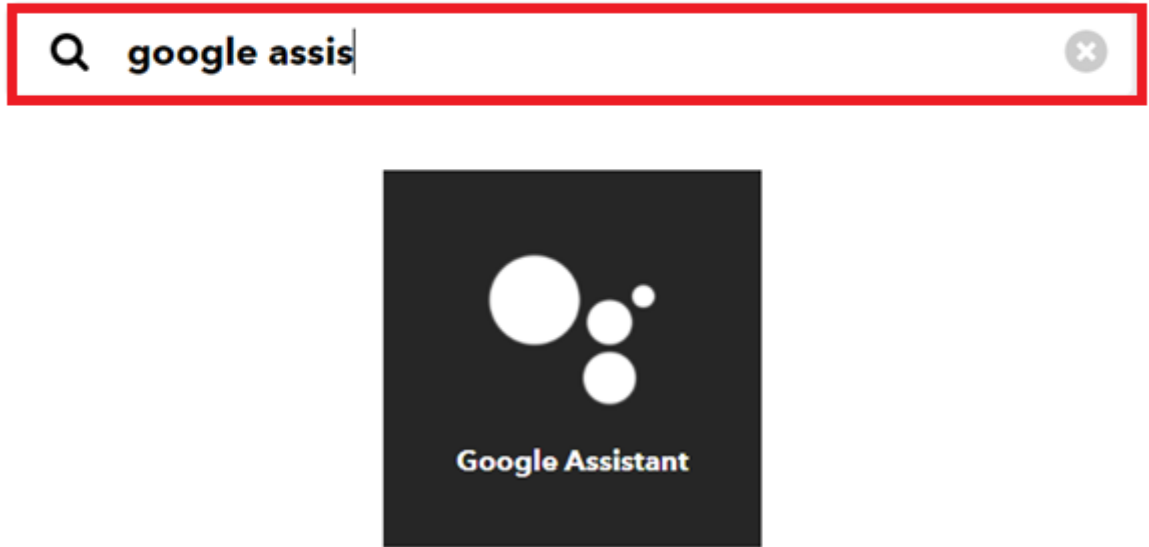
Create butonuna tıklandıktan sonra kullanıcının, tetikleyici olayları ve tetiklendikten sonra çalışması istenen uygulamaları ayarlaması (řekil 9.25) gerekir.



Şekil 9.25. Tetikleyicinin ve uygulamanın ayarlanması

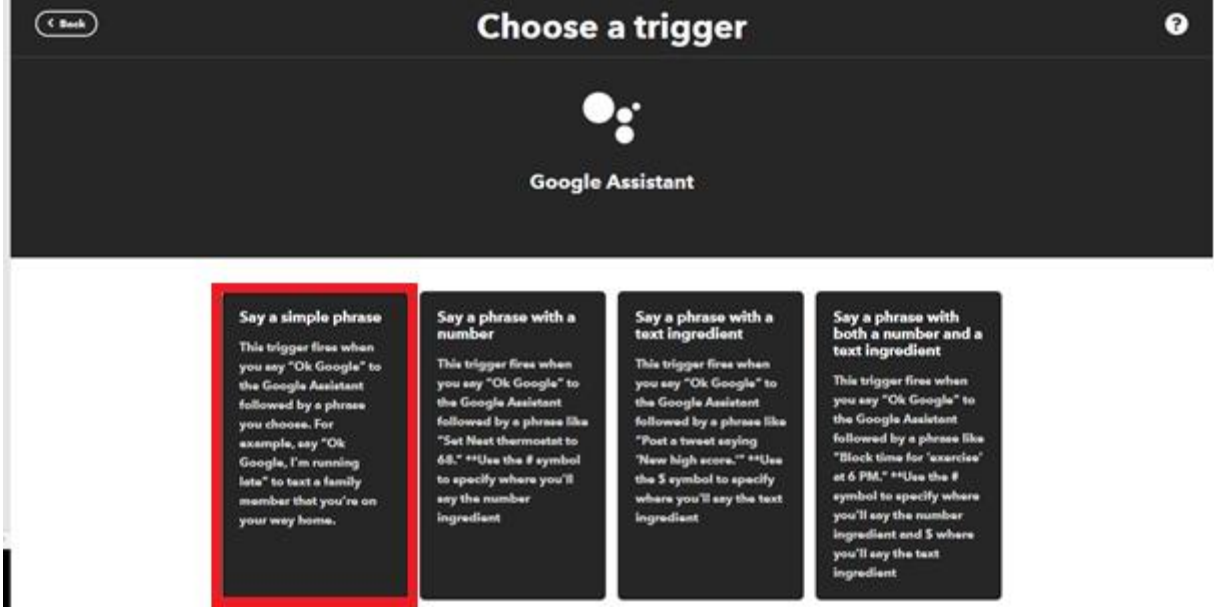
Create your own (kendi uygulamayı yarat) penceresinde **If This** (tetikleyici olay) belirlenmesi için **choose a service** bölümü açılacaktır. Uygulamada Google Asistan kullanılacağı için arama kutucuğuna “Google assis...” yazılarak (Şekil 9.26) servis bulunur.

Choose a service



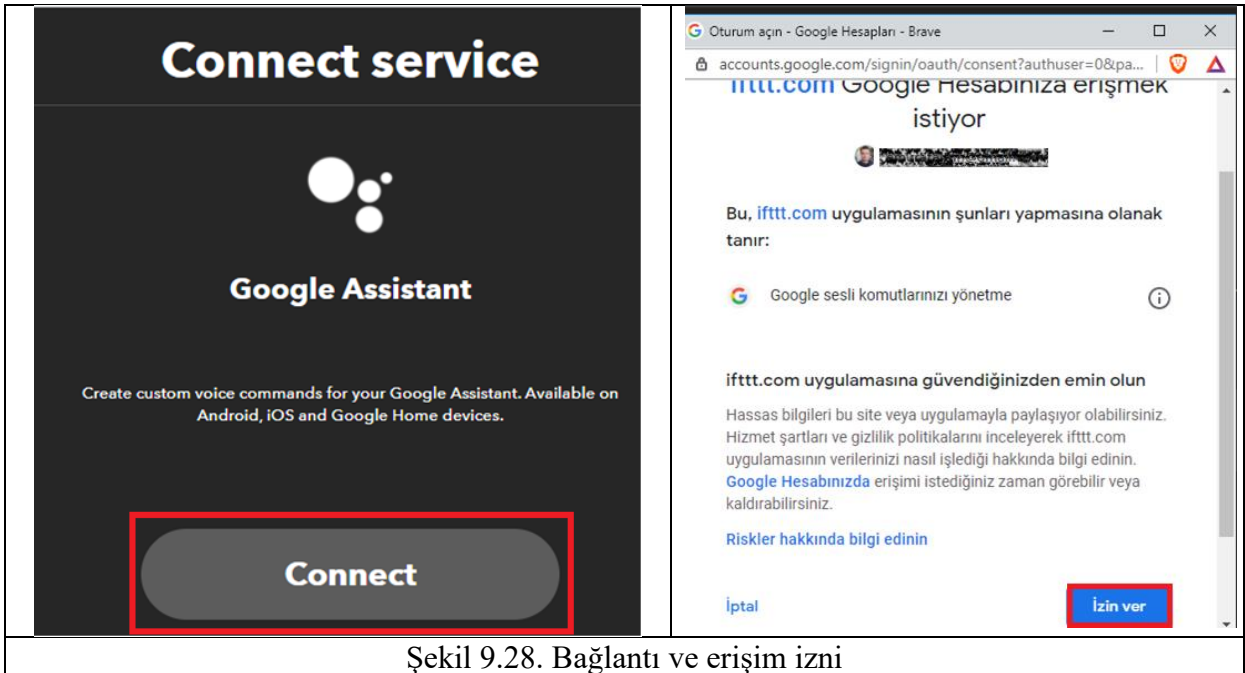
Şekil 9.26. Tetikleyici servis eklemek

Servis seçildikten sonra servisi tetiklemek için kullanılacak tetikleyiciler seçilmelidir. Bu uygulamada led kontrolü Google Asistan kullanılarak sesli olarak “lamp on, lamp off” gibi komutlarla yapılacaktır. Bunun için **Choose a trigger** (bir tetikleyici seç, şekil 9.27) bölümü kullanılır. Bu amaçla **Say a simple phrase** seçilir.



Şekil 9.27. Tetikleyici eklemek

IFTTT tarafından Google Asistan özelliğini kullanmak için bağlantı kurulması ve gerekli izinlerin verilmesi (Şekil 9.28) gerekmektedir.



Şekil 9.28. Bağlantı ve erişim izni

Ledi açmak için kullanılacak sesli komutlar yazılır. Serviste Türkçe seçeneği olmadığı için basit İngilizce cümleler kullanılmıştır. Google Asistana “Lamp On, Light lamp veya led on” olarak (Şekil 9.29) sesli komut verildiğinde veya yazıldığında tetikleyici çalışacaktır. **What do you**

want the Assistant to say in response? İşlem gerçekleştiğinde Google Asistan'ın vereceği yanıttır.

This trigger fires when you say "Ok Google" to the Google Assistant followed by a phrase you choose. For example, say "Ok Google, I'm running late" to text a family member that you're on your way home.

What do you want to say?

Lamp on

What's another way to say it? (optional)

Light lamp

And another way? (optional)

Led on

What do you want the Assistant to say in response?

lamp on

Language

English

Create trigger

Şekil 9.29. Komutların tanımlanması

Create Trigger butonuna tıklanır ve çalıştırılacak görev (eylem) seçilir. Bu eylem için **Choose a service** (bir servis seç) bölümünde arama kutucuğuna (Şekil 9.30) **webhooke** yazılarak bu servis seçilir.

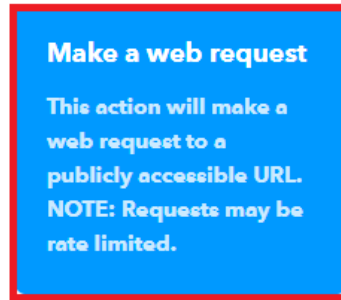
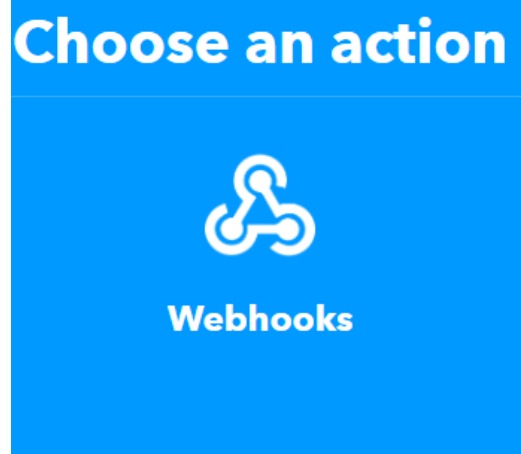
Choose a service

Q webhook

Webhook

Şekil 9.30. Webhooke servisinin seçilmesi


Webhooke seçildiğinde bir eylem seçilmesi istenmekte ve altında **make a web request** (bir web isteği oluştur) seçeneği (Şekil 9.31) çıkmaktadır.



Şekil 9.31. Bir web isteği oluşturma

Make a web request seçildikten sonra URL kısmına **Write Api Key** eklenmiş olan Thing Speak veri yayınlama adresi yazılır. Uygulama-5’te detaylı olarak anlatıldığı gibi ledi açmak için (Şekil 9.32) https://api.thingspeak.com/update?api_key=*****&field4=1 kullanılır. Method GET, content type text/plain olarak seçilmelidir.

Complete action fields



Make a web request

This action will make a web request to a publicly accessible URL. NOTE: Requests may be rate limited.

URL

`https://api.thingspeak.com/update
?api_key=*****
&field4=1`

Surround any text with <<< and >>> to escape the content

Add ingredient

Method

GET

The method of the request e.g. GET, POST, DELETE

Content Type

Please select

Optional

Body

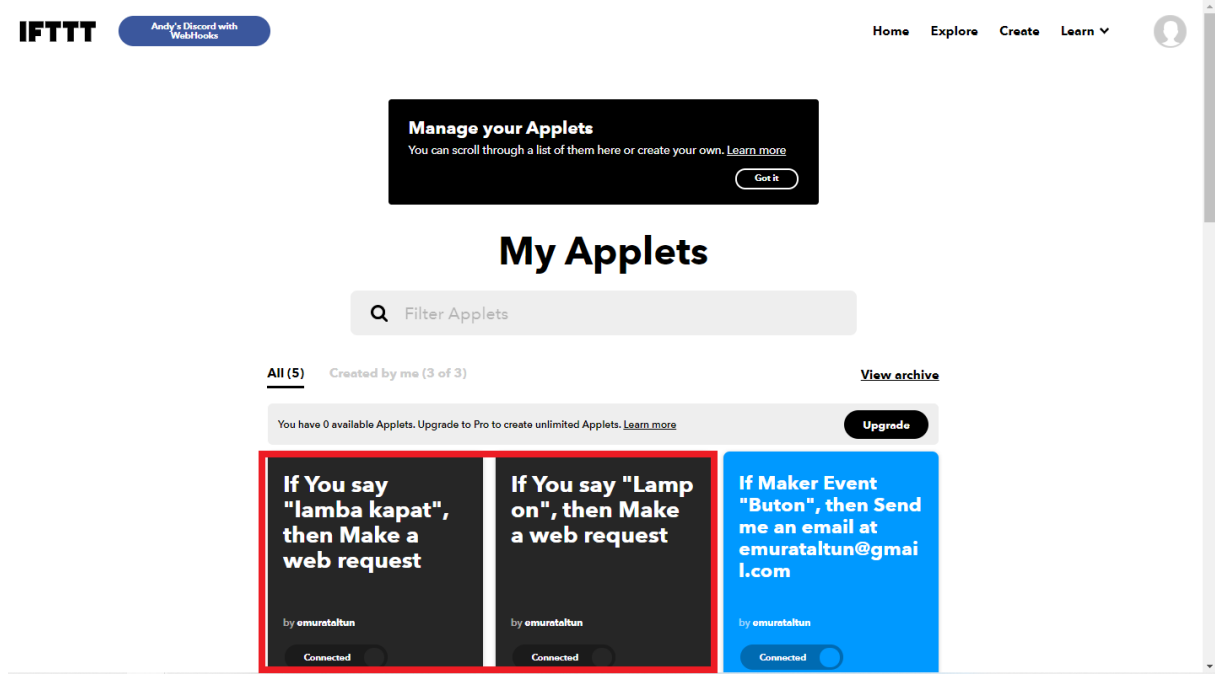
Surround any text with <<< and >>> to escape the content

Add ingredient

Create action

Şekil 9.32. Thing Speak api kullanımı

Ledi söndürmek için IFTTT üzerinde aynı işlem adımları **Create(+)** bölümünden itibaren tekrarlanır. **Say simple phrase** aşamasında ledi kapatmak için gerekli komutlar “Lamp Off, Light off veya led off” ve ledi söndürmek için webhooke kısmında adres olarak https://api.thingspeak.com/update?api_key=*****&field4=0 kullanılır. IFTTT üzerinde gerekli ayarlamalar yapıldığında kullanıcı sayfasında servisler (Şekil 9.33) aşağıdaki gibi 2 **web request** (Lamp on ve Lamp off) görünecektir.



Şekil 9.33. IFTTT Servisleri

Devrenin Kurulması

Devre Uygulama-2 ile aynıdır. Raspberry Pi, GPIO 18 nolu pine bağlı bir led bulunmaktadır.

Python Kodları

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır. Uygulama-2 'deki gibi **Thing Speak** üzerindeki kanal altındaki **field4** ile kontrol edilmektedir.

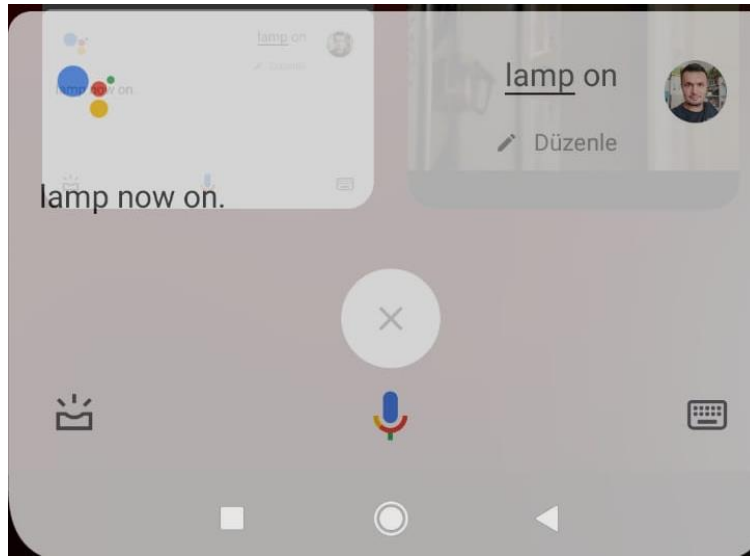
```
#!/usr/bin/env python
import RPi.GPIO as GPIO
import urllib.request
import json
from time import sleep
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
led=18
GPIO.setup(led, GPIO.OUT)
READ_API_KEY='*****'
CHANNEL_ID=123456
while True:
    conn = urllib.request.urlopen('http://api.thingspeak.com/channels/%s/feeds/last.json?api_key=%s' \
        % (CHANNEL_ID,READ_API_KEY))
    response = conn.read()
    print ('http durum kodu=%s' % (conn.getcode()))
    data=json.loads(response)
    print ('değer:', data['field4'], 'oluşturulma:', data['created_at'])
    if int(data['field4'])==1:
        GPIO.output(led, GPIO.HIGH)
    elif int(data['field4'])==0:
```

```
GPIO.output(led, GPIO.LOW)  
conn.close()
```

Uygulama-7'ye ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/2vrbg>) bağlantısı kullanabilir.

Uygulamanın test edilmesi

IFTTT üzerindeki **web request** servisleri aktif (connected) olmalıdır. Yukarıdaki Python kodu çalıştırılmalıdır. Telefon üzerinden Google Asistan'a sesli olarak "Lamp On, Light lamp veya led on" komutları verildiğinde led yanmalıdır. "Lamp Off, Light off veya led off" sesli komutları verildiğinde ise led sönmelidir. Led yandığında ve söndüğünde Google Asistan durumu bir mesajla (Şekil 9.34) belirtecektir.



Şekil 9.34. Google Asistan ile uygulamanın test edilmesi

Uygulama -8

Raspi Telegram Api Kullanımı

Bu uygulamada bir Telegram botu aracılığıyla Raspberry Pi üzerinde DHT11 sensörü ve bir ledten oluşan devre kontrol edilecektir. Bu uygulamayı yapabilmek için öncelikle bir Telegram botu oluşturmak gerekir. Telegram botu oluşturmak için gerekli işlem aşağıda sunulmuştur.

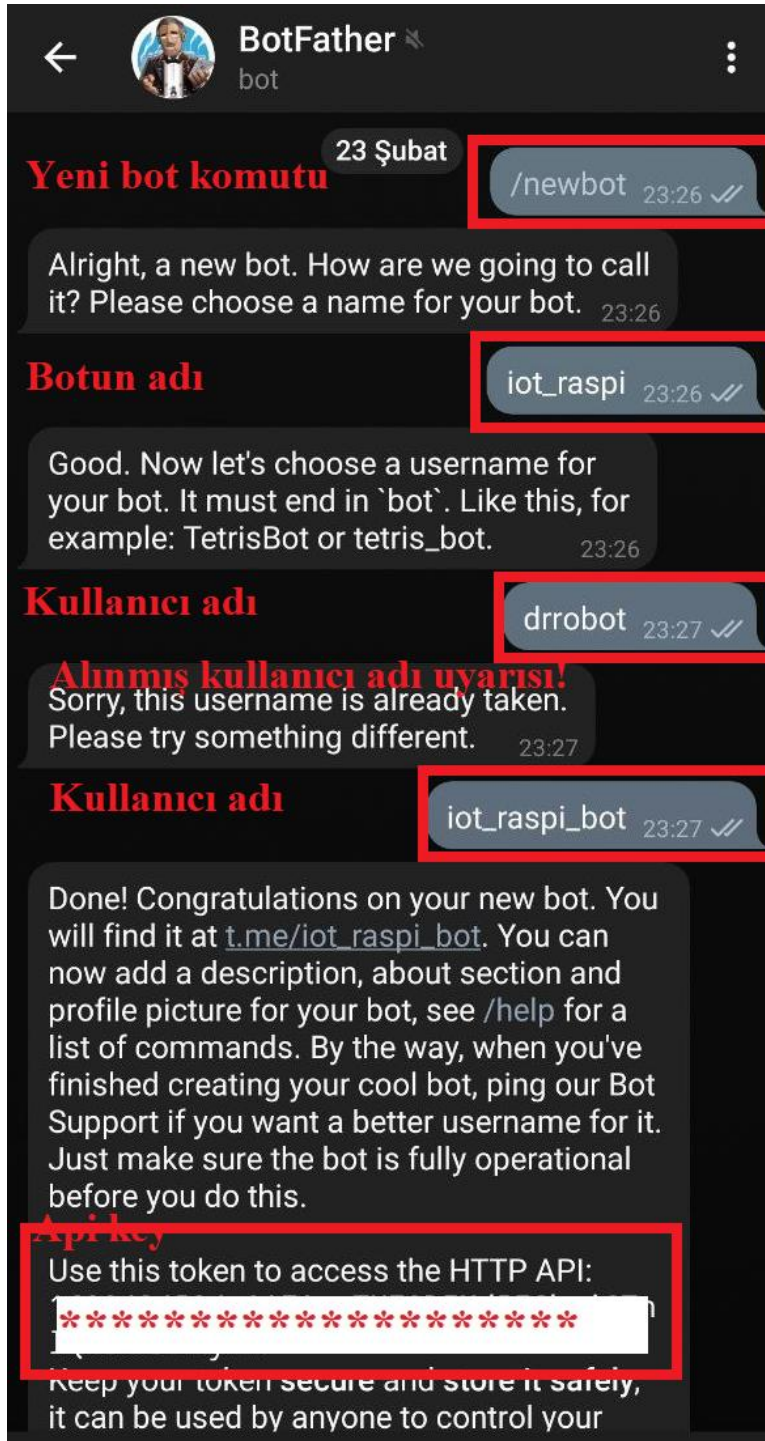
1. Akıllı telefona Telegram uygulamasının kurulması.
2. Bir bot oluşturmak için arama (Şekil 9.37) kısmına **botfather** yazılır. **BotFather** adlı bot açılarak yeni bir bot oluşturmak için işlemlere başlanır.



Şekil 9.35. Bot Father kullanımı

3. **BotFather** ile konuşma penceresinde komutlar gönderilerek (mesaj gönderir gibi) yeni bot oluşturulur.

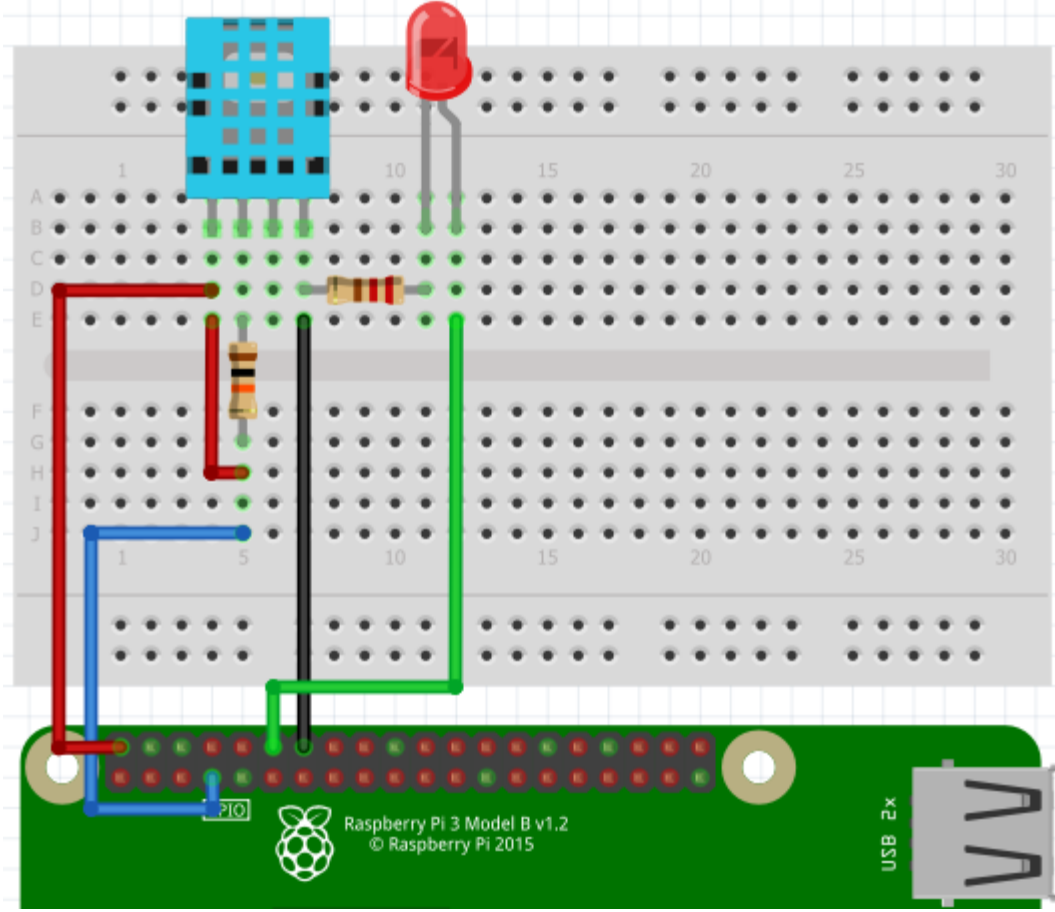
/newbot : Yeni bot oluştur. Bu komut gönderildikten sonra **BotFather** oluşturulacak bot için bir isim verilmesini isteyecektir. Bot için bir isim girilir ve gönderilir. **Bot Father** sonraki aşamada bir kullanıcı adı isteyecektir. Bot ismi ve kullanıcı adı daha önce alındıysa **Bot Father** kullanıcıya uyarı mesajı (bu isim daha önce alındı başka bir şey deneyin) verir. Kullanıcı adı da girildikten sonra **Bot Father** bot oluşturulduğuna dair bir mesaj gönderecektir. Mesajda, bot bağlantısı (web üzerinden de açabilir) ve programlarda kullanmak için HTTP Api bilgileri verilmektedir. Bu bilgiler daha sonra kullanılacaktır.



Şekil 9.36. Yeni bot oluşturma

Devre Kurulumu

Devre Şekil 9.37'deki gibi kurulur. Devrede bir DHT11 (GPIO 4) sensörü ve bir led (GPIO 18) bulunmaktadır.



Şekil 9.37. DHT11 ve led devresi

Kullanılan malzemeler
Raspberry Pi
Bread board
1 Adet DHT11
Direnç 10 K Ω
1 Adet Led
Direnç 220 Ω
Bağlantı kabloları

Python Kodları

Telegram Api kullanmak için **telepot** paketinin kurulması gerekir.

```
sudo pip3 install telepot
```

Mu editöründe yeni bir dosya açılarak aşağıdaki kodlar yazılır, dosya kaydedilir ve çalıştırılır. DHT11 kütüphanesinin kullanılabilmesi için python dosyası **/home/pi/DHT11_Python** dizini içine kaydedilmelidir. Programa öncelikle gerekli kütüphaneler dahil edilir.

```
import datetime # datetime kütüphanesi dahil ediliyor.  
import telepot # telepot kütüphanesi dahil ediliyor.  
# Telegram botuyla iletişimi sağlayacak kütüphane fonksiyonu.  
from telepot.loop import MessageLoop
```

```
# GPIO pinlerini kullanmak için GPIO kütüphanesi dahil ediliyor.
import RPi.GPIO as GPIO
import dht11          #DHT11 için gerekli kütüphane dahil ediliyor.
led_pin = 18          # ledin bağlı olduğu pin belirleniyor.
GPIO.cleanup()        #GPIO pinleri sıfırlanıyor.
GPIO.setmode(GPIO.BCM) # BCM pin dizilişi seçiliyor.
GPIO.setup(led_pin, GPIO.OUT) # led_pin çıkış olarak ayarlanıyor.
sicaklikNem=dht11.DHT11(pin=4) #DHT11 sinyal pini GPIO 4
now = datetime.datetime.now() # Tarih ve saat alınıyor.
```

Bot'a gelen mesajlar alınarak mesaja göre işlemler yapılmakta ve yanıtlar türetilmektedir.

```
def kontrol(msg):
    chat_id = msg['chat']['id'] #Alınan mesaj
    komut = msg['text'] # Mesajdaki metin komut olarak alınıyor.
    print ('Alındı:', komut)
    if komut == '/mrbr':
        bot.sendMessage(chat_id, str("Merhaba ben IOT Bot/Yazar: Dr. Murat Altun"))
    elif komut== '/saat':
        bot.sendMessage(chat_id, str("Time: ") + str(now.hour) + str(":") + str(now.minute) +
str(":") + str(now.second))
    elif komut == '/trhr':
        bot.sendMessage(chat_id, str("Date: ") + str(now.day) + str("/") + str(now.month) +
str("/") + str(now.year))
    elif komut == '/Ac':
        GPIO.output(led_pin, True)
        bot.sendMessage(chat_id, str("Led yanıyor."))
    elif komut == '/Kapa':
        GPIO.output(led_pin, False)
        bot.sendMessage(chat_id, str("Led söndü."))
    elif komut == '/Scklk':
        deger=sicaklikNem.read()
        bot.sendMessage(chat_id, deger.temperature)
    elif command == '/Nem':
        deger=sicaklikNem.read()
        bot.sendMessage(chat_id, deger.humidity)
```

HTTP Api key ile bot nesnesi türetilir ve sürekli olarak mesaj kontrolü yapılır.

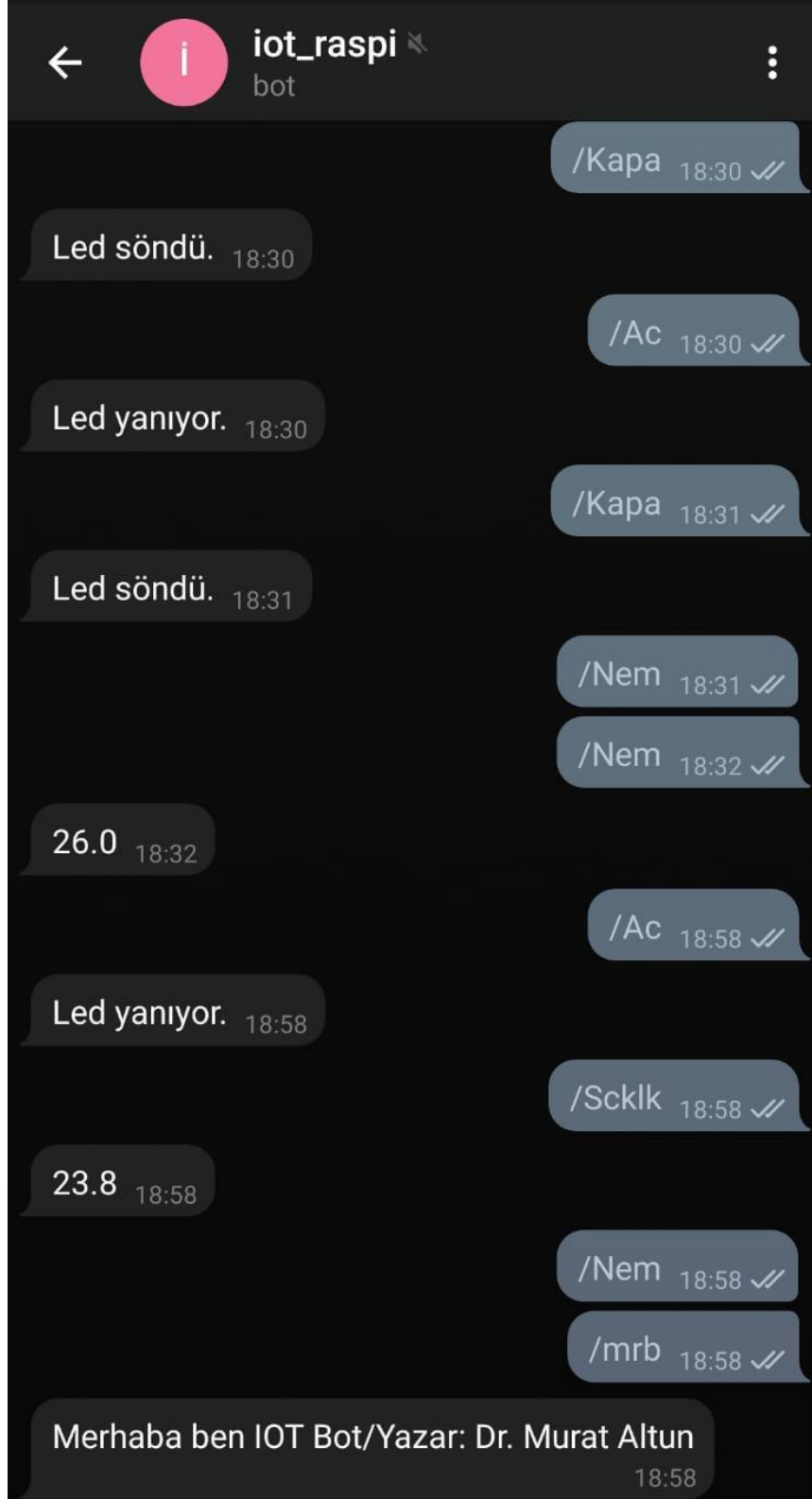
```
bot = telepot.Bot('*****') #HTTP API key girilmeli.
print (bot.getMe()) #Bot'un özellikleri yazdırılıyor.
MessageLoop(bot, kontrol).run_as_thread()
#Sürekli olarak bot mesaj durumu kontrol edilmektedir.
```

Uygulama-8'e ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/49WAt>) bağlantısı kullanılabilir.

Uygulamanın Test Edilmesi

Uygulamanın Python kodu çalıştırılır. Telegram uygulamasına girilir ve uygulama için geliştirilen **iot_raspi** botu açılır. Bot yazılan mesajlara uygun olarak işlemler yapıyorsa uygulama doğru çalışıyordur. **/Ac** yazıldığında devredeki ledin yanması gerekir. **/Kapa** komutu kullanıldığında ise led söner. **/Scklk** yazıldığında **DHT11** sensöründen okunan sıcaklık değeri

bot tarafından yazılır. **/Scklk** komutu ise nem deęerini verecektir. Bot ekran grnts Őekil 9.38’de verilmiřtir.



Őekil 9.38. Telegram bot uygulaması

Programda farklı komutlar eklenerek bunlar için farklı işlemler atanabilir. Böylece daha gelişmiş özelliklere sahip bir bot oluşturulur.

Uygulama -9

Twitter Uygulaması

Bu uygulamada DHT11 sıcaklık ve nem sensöründeki değerler, Twitter api aracılığıyla tweet olarak gönderilecektir. Twitter Api'nin kullanılabilmesi için developer olarak başvuru yapmak (<https://developer.twitter.com/en>) gerekir. Bu konuda ayrıntılı bilgi almak için <https://burakgarci.net/twitter-api-key-ve-token-alma/> adresi veya Twitter Developer sayfaları ziyaret edilebilir. Başvuru yapıldıktan sonra developer sayfasındaki Api Key kullanılarak program üzerlerinden Twitter fonksiyonlarına erişilebilir.

Devre Kurulumu

Devre Uygulama-5'deki gibi kurulmuştur. Devrede sadece DHT11 (GPIO 4) sensörü bulunmaktadır.

Python Kodları

Öncelikle Twitter Api kullanmak için Twython paketi kurulmalıdır. Paketi kurmak için terminale aşağıdaki komut yazılır.

```
sudo pip3 install Twython
```

DHT11 kütüphanesinin kullanabilmesi için Python dosyası **/home/pi/DHT11_Python** dizini içine kaydedilmelidir.

```
#Gerekli kütüphaneler
from twython import Twython
import RPi.GPIO as GPIO
import dht11
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
sicaklikNem=dht11.DHT11(pin=4)
APP_KEY = r'*****'
APP_SECRET=r'*****'
ACCESS_TOKEN =r'*****'
ACCESS_SECRET=r'*****'
deger=sicaklikNem.read()
if deger.is_valid():
    yeni_tweet = "Sensor sıcaklık: " + str(deger.temperature)+ " Sensor nem: "
    +str(deger.humidity )
    twitter = Twython(APP_KEY, APP_SECRET, ACCESS_TOKEN, ACCESS_SECRET)
    twitter.update_status(status=yeni_tweet)
```

Uygulama-9'a ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/rsxBO>) bağlantısı kullanılabilir.

Uygulamanın Test Edilmesi

Uygulama çalıştırıldığında Şekil 9.39'daki gibi tweet gönderecektir. Uygulama her çalıştırıldığında sadece bir tweet gönderir. Programdaki tweet gönderme işlemi belirli şartlara bağlanarak farklı programlar geliştirilebilir.



Şekil 9.39. Gönderilen tweetler

Uygulama -10

Buton e-posta uygulaması

Bu uygulamada Raspberry Pi'a bağlı bir butona basıldığında program e-posta gönderecektir.

Devre Kurulumu

Uygulama-1'deki devre ile aynı şekilde kurulmuştur. Devrede GPIO 23 pinine bağlı bir buton bulunmaktadır.

GMAIL Servis Ayarları

E-posta işlemleri için Gmail SMTP ayarlarının yapılması gerekir. Google hesaba girilerek ayarlardan SMTP ayarlarının değiştirilmesi gerekir. Daha az güvenilir kaynaklara erişimi açmak gerekir (<https://www.google.com/settings/u/2/security/lesssecureapps>). Bunu yapmak için hesap ayarlarında iki aşamalı doğrulamayı devre dışı bırakmak gerekir. Aşağıdaki işlemlerin yapılması gerekir.

1. Google hesaba girilir.
2. Profil resmine tıklayarak Google hesabınızı yönetin seçeneği seçilir.
3. Güvenlik bölümü altında **Oturum Açma/İki Adımlı Doğrulama** seçeneği pasif hale getirilir.
4. Güvenlik bölümü altında **daha az güvenli uygulama** erişimi aktif hale getirilir.
5. Gmail bölümüne geçilerek profil resminin yanındaki ayarlar simgesine tıklanır tüm ayarları görüntüleyin seçeneğine tıklanır. Açılan ayarlar penceresinden Yönlendirme ve POP/IMAP sekmesine gelinerek aşağıdaki seçeneklerden uygun olan işaretlenir.

Tüm postalar için POP'u etkinleştir (daha önce indirilmiş olan postaları da dahil et)

Şu andan itibaren gelen postalar için POP'u etkinleştir.

Aynı bölümde IMAP'ı etkinleştir seçeneği de işaretlenir.

Python Kodları

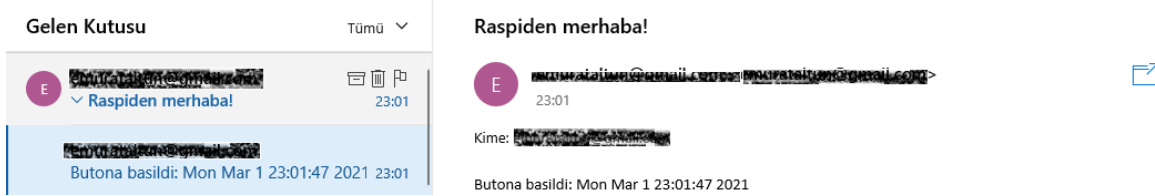
Mu editör açılarak yeni bir dosyada aşağıdaki kodlar oluşturulur. Python kodlarında e-posta adresi, e-posta gönderilecek adres, e-posta parolası değiştirilmelidir. Gmail e-posta servisi için SMP server ve SMTP port değerleri değiştirilmemelidir.

```
import smtplib
from time import sleep, ctime
import RPi.GPIO as GPIO
GPIO.cleanup()
buton=23
GPIO.setmode(GPIO.BCM)
GPIO.setup(buton,GPIO.IN,pull_up_down=GPIO.PUD_DOWN)
GPIO.setwarnings(False)
#Butona basılma olayını tarayan bir GPIO fonksiyonu eklenir.
GPIO.add_event_detect(buton, GPIO.RISING)
#E-posta değişkenleri.
SMTP_SERVER = 'smtp.gmail.com' #E-posta servisi değiştirmeyiniz!
SMTP_PORT = 587 #Sunucu portunu değiştirmeyiniz!
GMAIL_USERNAME = '*****@gmail.com' #kullanılacak e-posta adresi.
GMAIL_PASSWORD = '*****' #kullanılacak e-posta adresi parolası.
def sendmail(recipient, subject, content):
    #Create Headers
    headers = ["From: " + GMAIL_USERNAME, "Subject: " + subject, "To: " +
recipient,
    "MIME-Version: 1.0", "Content-Type: text/html"]
    headers = "\r\n".join(headers)
    #Gmail sunucuya bağlanılır.
    session = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    session.starttls()
    session.ehlo()
    #Gmail'de oturum açılır.
    session.login(GMAIL_USERNAME, GMAIL_PASSWORD)
    #E-posta gönderilir ve oturumdan çıkılır.
    session.sendmail(GMAIL_USERNAME, recipient, headers + "\r\n\r\n" + content)
    session.quit
while True:
    if GPIO.event_detected(buton):
        sendTo = 'emurataltun@gmail.com' #e-posta gönderilecek e-posta adresi.
        emailSubject = "Raspiden merhaba!"
        emailContent = "Butona basıldı: " + ctime()
        #Türkçe karakter sorunu yaşandığı için basıldı yazılmıştır.
        sendmail(sendTo, emailSubject, emailContent)
        print("E posta gönderildi")
        sleep(1)
```

Uygulama-10'a ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/baPas>) bağlantısı kullanabilir.

Uygulamanın Test Edilmesi

Butona basıldığında Raspiden merhaba! konulu bir e-posta gönderilecektir. Gönderilen e-posta içeriğinde “Butona basildi: Tarih ve saat” iletisi (Şekil 9.40) bulunmaktadır.



Şekil 9.40. Uygulamadan gönderilmiş e-posta

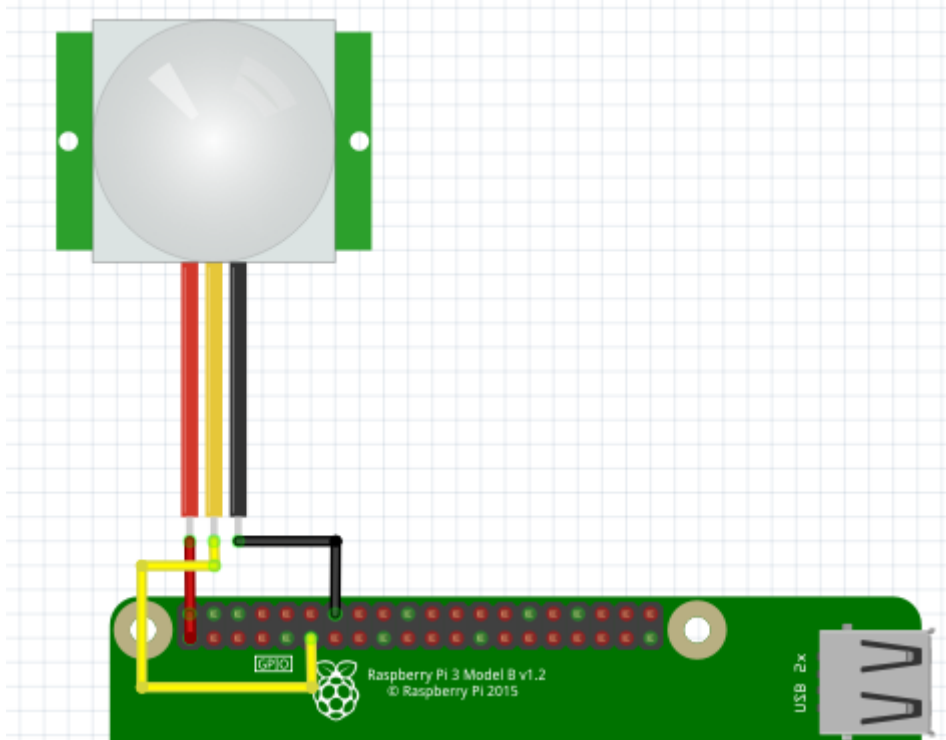
Uygulama -11

Güvenlik sistemi uygulaması

Bu uygulamada hareket sensörü ve bir kamerayla bir güvenlik sistemi oluşturulmaktadır. Bir hareket algılandığında fotoğraf çekerek e-posta ile göndermektedir.

Devre Kurulumu

Devre, Şekil 9.41'deki gibi kurulmuştur. PIR hareket sensörü sinyal çıkışı (sarı) GPIO 17 nolu pine bağlanmıştır. Raspberry Pi üzerinde bir adet usb web cam bulunmaktadır.



Şekil 9.41. Güvenlik sistemi

Python Kodları

Usb web kameradan görüntü alabilmek için Opencv paketi kurulmuştur. Paketi kurmak için terminalde aşağıdaki komut yazılır.

```
sudo apt-get install python3-opencv
```

Mu editör açılarak yeni bir dosyada aşağıdaki kodlar oluşturulur. Python kodlarında e-posta adresi, e-posta gönderilecek adres, e-posta parolası değiştirilmelidir. Gmail e-posta servisi için SMP server ve SMTP port değerleri değiştirilmemelidir.

```
#Hareket algılandığında e-posta gönderen uygulama
import datetime
import cv2
import smtplib
import time
from time import sleep, ctime
import RPi.GPIO as GPIO
GPIO.setwarnings(False)
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)
pir=17
GPIO.setup(pir,GPIO.IN)
SMTP_SERVER = 'smtp.gmail.com'
SMTP_PORT = 587
GMAIL_USERNAME = '*****@gmail.com'
GMAIL_PASSWORD = '*****'
def sendmail(recipient, subject, content,image):
    #Headers oluşturuluyor.
    emailData = MIMEMultipart()
    #konu
    emailData['Subject'] = subject
    #alıcı
    emailData['To'] = recipient
    emailData['From'] = GMAIL_USERNAME
    #mesaj içeriği
    emailData.attach(MIMEText(content))
    #resim dosyası tanımlama
    imageData = MIMEImage(open(image, 'rb').read(), 'jpg')
    imageData.add_header('Content-Disposition', 'attachment; filename="image.jpg"')
    #foto ekleniyor.
    emailData.attach(imageData)
    session = smtplib.SMTP(SMTP_SERVER, SMTP_PORT)
    session.starttls()
    session.ehlo()
    #Gmail'de oturum açılır.
    session.login(GMAIL_USERNAME, GMAIL_PASSWORD)
    #E-posta gönderilir ve oturumdan çıkılır.
    session.sendmail(GMAIL_USERNAME, recipient, emailData.as_string())
    session.quit
#Fotoğraf çekme fonksiyonu tanımlanıyor.
def foto_cek(camera_port = 0):
    camera = cv2.VideoCapture(camera_port)
```

```
time.sleep(0.1) # Bekleme eklenmezse foto karanlık olabilir.
return_value, image = camera.read()
#Otomatik dosya adı ekleniyor.
file_name = datetime.datetime.now().strftime("%Y-%m-%d_%H%M")+ ".jpg"
#Foto dosya olarak kaydedilir.
cv2.imwrite(file_name, image)
#Kaydeilen fotoğrafın adı döndürülüyor.
return file_name
#kamera meşgul kalmasın diye nesne siliniyor
del(camera)
#e-posta kütüphaneleri import ediliyor.
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
while True:
    print ("Sistem devrede.")
    #Hareket algılandığında
    if GPIO.input(pir)==1:
        print("Hareket algılandı!")
        sendTo = '*****@gmail.com '
        emailSubject = "Güvenlik sistemi uyarı!"
        emailContent = "Hareket algılandı." + ctime()
        sendmail(sendTo, emailSubject, emailContent,foto_cek())
        print("E posta gönderildi")
        sleep(2) #Çok sık gönderme olmaması için süre uzatılabilir.
```

Uygulama-11'e ait Python kodlarını dosya halinde indirmek için (<http://meb.ai/1vvS8>) bağlantısı kullanabilir.

Uygulamanın Test Edilmesi

Python kodları çalıştırıldığında ortamda (PIR sensörü ile) bir hareket algılandığı zaman web kamera tarafından çekilen fotoğraf ayarlanan e-posta adresine dosya olarak eklenerek otomatik olarak (Şekil 9.42) gönderilmektedir.

Guvenlik sistemi uyarı! ⚠

Gelen Kutusu x



Alıcı: ben ▼

Hareket algilandi. Tue Mar 2 17:49:45 2021



↩ Yanıtla

➡ Yönlendir

Şekil 9.42. Otomatik uyarı gelen e-posta

Bu bölümde Raspberry Pi ile Python programlama dili kullanılarak IoT uygulamaları geliştirilmiştir. Raspberry Pi ile farklı programlama dilleri veya farklı platformlar kullanılarak çeşitli IoT uygulamaları geliştirilebilir. Raspberry Pi ve Python ile geliştirilen IoT uygulamaları için ekonomik çözümler sunmaktadır. Raspberry Pi ile akıllı ev sistemleri, yüz tanıma sistemleri, akıllı tarım uygulamaları ve hava ölçüm istasyonu gibi uygulamalar yapılabilir.

Kaynaklar (Düzenlenecek)

<https://pypi.org/project/paho-mqtt/>
<https://www.abelectronics.co.uk/kb/article/1085/io-pi-tutorial---mqtt-control>
<https://www.mathworks.com/help/thingspeak/google-ifttt-thingspeak-lamp.html>
<https://www.ilhanakilli.com.tr/2017/10/mqtt-broker-kurulumu-mosquitto-ve-deneme.html>
<https://www.hivemq.com/blog/mqtt-essentials-part-5-mqtt-topics-best-practices/>
<https://maker.robotistan.com/kamerali-wifi-robot/>
<https://crosser.io/use-cases/use-cases-project-examples/>
<https://iot.eclipse.org/>
<https://mosquitto.org/>
<https://www.hivemq.com/public-mqtt-broker/>
<https://nevonprojects.com/raspberry-pi-projects/>
<http://www.steves-internet-guide.com/mqtt-username-password-example/>
<http://www.hivemq.com/demos/websocket-client/>
<https://community.blynk.cc/t/raspberry-pi-zero-w-as-blynk-local-server/13875>
<https://www.hivemq.com/blog/mqtt-client-library-paho-python/>
https://www.mathworks.com/help/thingspeak/examples.html?category=index&s_tid=CRUX_index
<https://www.element14.com/community/community/raspberry-pi/blog/2016/08/16/control-raspberry-pi-gpios-with-websockets>
<https://devnot.com/2017/mqtt-nedir-nasil-bir-mimaride-calisir/>
<http://mqtt.org/> (Resmi sayfası)
<https://dzone.com/refcardz/getting-started-with-mqtt>
<http://www.iot.gen.tr/2016/01/15/mqtt-message-queuing-telemetry-transport-nedir/>
<https://gu.ray.kim/devfest-3/> (Sunum)
<https://medium.com/@iotmqtt/mqtt-nedir-f50a91d372b1>
<https://selimyalnkaya.medium.com/message-queuing-telemetry-transport-mqtt-mesaj-kuyru%C4%9Fu-telemetri-aktar%C4%B1m%C4%B1-protokol%C3%BC-1e66ebdd9fbb>
<https://www.programcreek.com/python/example/98876/RPi.GPIO.setwarnings>
<https://maker.pro/raspberry-pi/projects/how-to-create-a-telegram-bot-with-a-raspberry-pi>