

درس مبانی اینترنت اشياء

گزارش تمرین ۳ و ۴

فهرست

فهرست ۲

بخش ۱ ۳

همزادهای دیجیتال ۵

قراردادهای هوشمند ۶

IoT BMI Tool ۶

بخش دوم ۹

الف) ۹

ب) ۱۰

ج) ۱۰

د) ۱۲

ه) ۱۲

بخش سوم ۱۳

الف) ۱۳

ب) ۱۴

ج) ۱۵

د) ۱۵

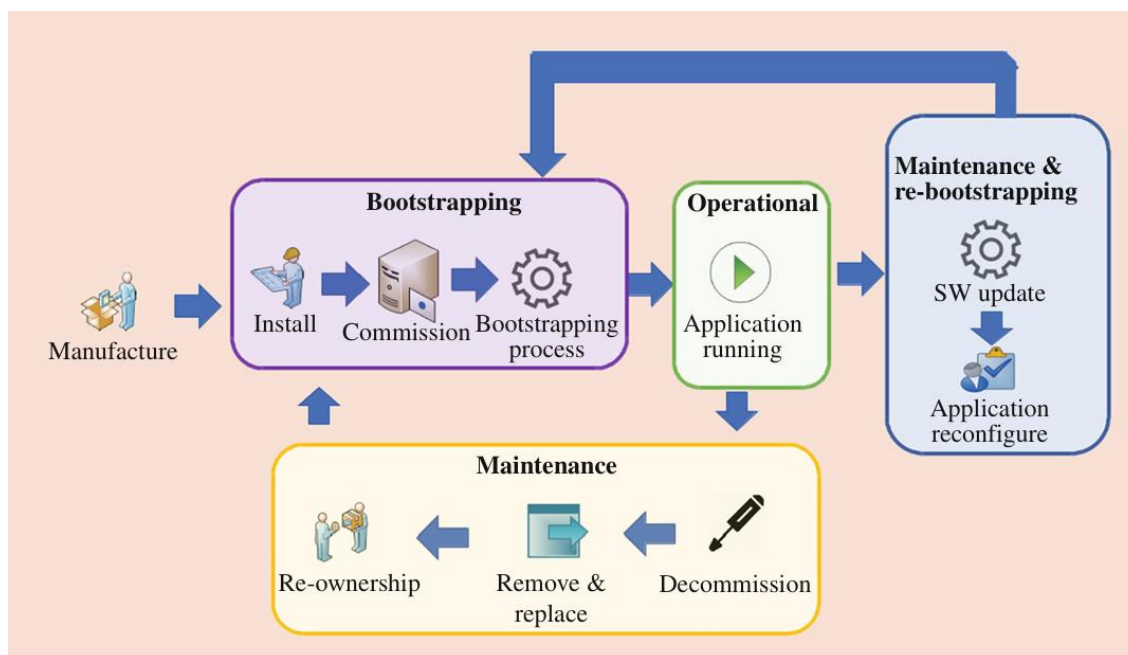
ه) ۱۶

بخش ۱

بوت‌استریپینگ

به فرآیند پیوستن امن یک دستگاه به شبکه و ایجاد اعتماد اولیه بین دستگاه و زیرساخت شبکه bootstrap کردن می‌گویند. امروزه، bootstrapping به فرآیندی اطلاق می‌شود که طی آن یک دستگاه (smart object) اطلاعات مورد نیاز برای پیوستن به یک شبکه و انجام عملیات را به دست می‌آورد، شامل:

- آدرس IP
- پارامترهای امنیتی (cipher suites، کلیدهای مشترک، گواهی‌نامه‌ها)
- اطلاعات خدماتی دیگر



چرخه‌ی عمر دستگاه هوشمند در IoT معمولاً شامل سه مرحله است:

تولید (Manufacture):

- تولید فیزیکی دستگاه و احتمالاً درج اولیه‌ی ریشه اعتماد (root of trust)

بوت‌استریپینگ (Bootstrapping):

- نصب و راه‌اندازی اولیه
- بارگذاری کلیدها و گواهی‌نامه‌ها
- پیوستن امن به دامنه امنیتی تحت نظارت Controller

مرحله عملیاتی (operational):

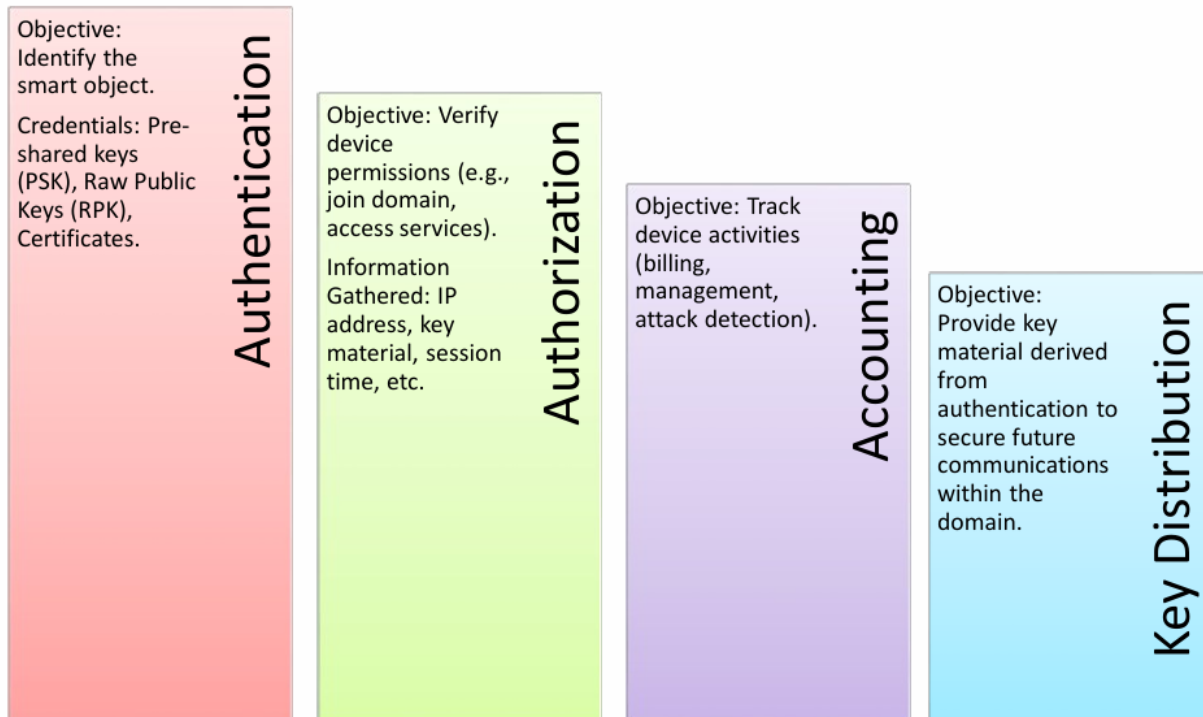
- آغاز عملیات اصلی (مثلاً جمع‌آوری داده از حسگرها)
- برقراری پروتکل‌های امنیتی جدید در صورت نیاز

نگهداری و Re-Bootstrapping:

- به‌روزرسانی نرم‌افزار/سخت‌افزار
- احراز هویت و مجوزدهی مجدد
- انتقال مالکیت در صورت تعویض مالک

Decommissioning:

- حذف ایمن اطلاعات و گواهی‌نامه‌ها



Bootstrapping در IoT بر چهار رکن امنیتی اصلی استوار است:

:Authentication

- شناسایی دستگاه هوشمند
- استفاده از کلیدهای pre-shared, کلیدهای عمومی خام، یا گواهی‌نامه‌ها

:Authorization

- تأیید دسترسی‌های دستگاه
- دریافت اطلاعات مانند IP، زمان جلسه، کلیدها و غیره

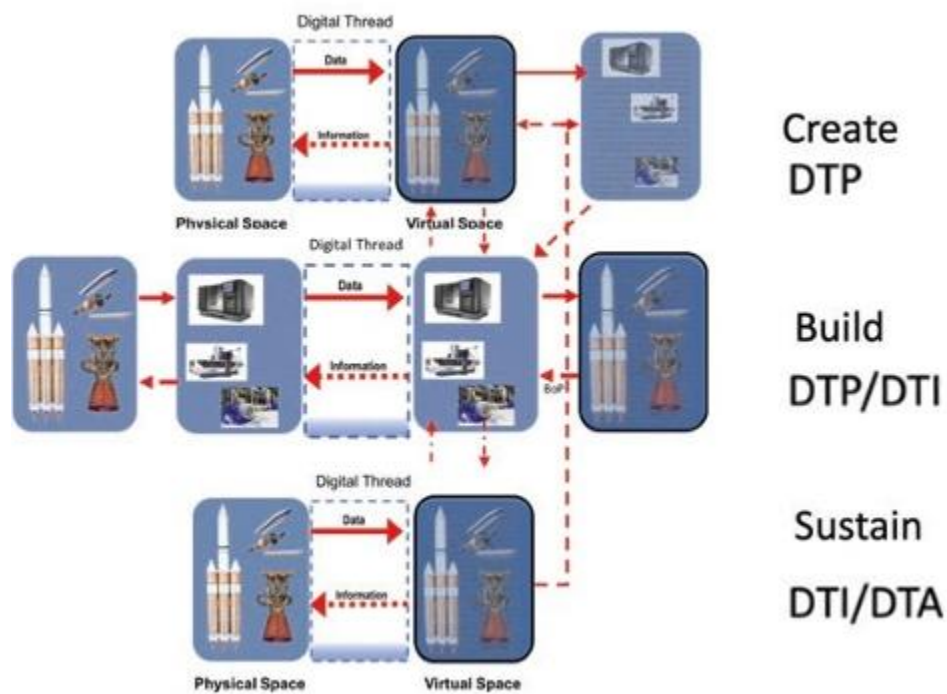
:Accounting

- ردیابی فعالیت‌های دستگاه
- اهداف مدیریتی، صورتحساب، و شناسایی حملات احتمالی

:Key Distribution

- استخراج کلیدهای امنیتی برای ایجاد ارتباطات امن با موجودیت‌های دیگر در دامنه امنیتی

همزادهای دیجیتال



Digital Twin Prototype (DTP)

- مرحله: Create
- کاربرد: برای طراحی، شبیه‌سازی، آزمون مجازی و برنامه‌ریزی ساخت پیش از تولید فیزیکی استفاده می‌شود.
- هدف: بهینه‌سازی و کامل‌سازی و رفع خطاها و نواقص احتمالی سیستم در فضای مجازی قبل از ساخت واقعی.

Digital Twin Instance (DTI)

- **مرحله:** Build و Sustain
- **کاربرد:** برای هر محصول یا سیستم واقعی ساخته شده با استفاده از داده‌های DTP یک نسخه دیجیتال اختصاصی ایجاد می‌کند. شامل داده‌های As-Built، شماره سریال، تاریخچه عملیاتی و داده live performance می‌باشد.
- **هدف:** وضعیت هر سیستم خاص به صورت زنده مانیتور شود، عملکرد واقعی آن ثبت شود و امکان عیب‌یابی و نگهداری هوشمند فراهم شود.

Digital Twin Aggregate (DTA)

- **مرحله:** Sustain
- **کاربرد:** تجمیع داده‌ها از چندین DTI در سطح یک مجموعه. امکان تحلیل جمعی، نگهداری پیش‌بینانه و بهینه‌سازی عملکرد را فراهم می‌کند.
- **هدف:** این داده‌ها برای پیش‌بینی خرابی‌ها، و بهینه‌سازی عملکرد کلی سیستم‌ها به کار گرفته می‌شوند. هدف DTA این است که سازمان بتواند با استفاده از بینش جمعی، از نگهداری زمان‌بندی شده به سمت نگهداری پیش‌بینانه حرکت کند و تجربه‌های گذشته را برای بهبود طراحی و عملکرد آینده به کار گیرد.

قراردادهای هوشمند

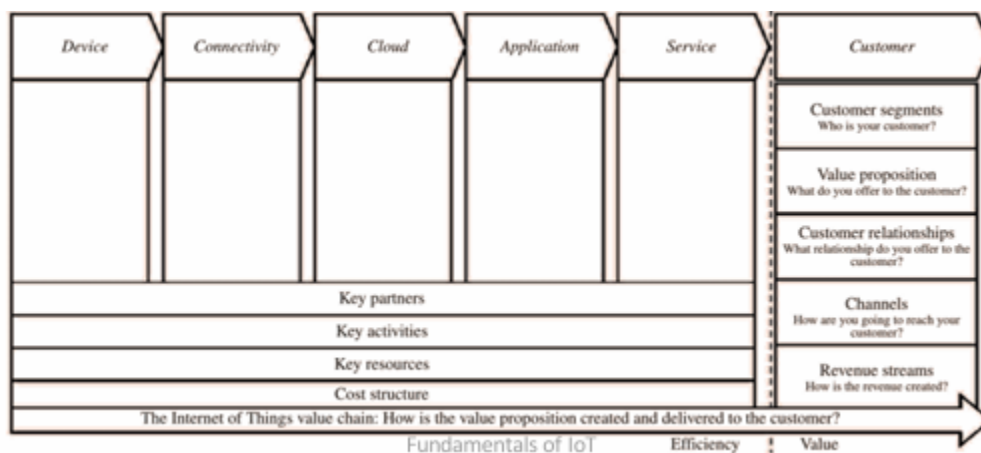
یک Smart Contract یک برنامه‌ی کامپیوتری Self-Executing است که روی یک Blockchain ذخیره می‌شود و به صورت خودکار شرایط یک توافق را زمانی که شرایط از پیش تعیین شده برقرار شوند، اعمال و اجرا می‌کند. برخلاف قراردادهای سنتی که به واسطه‌ها (مانند وکلا یا بانک‌ها) نیاز دارند، Smart Contract ها روی شبکه‌های Decentralized (مانند Ethereum) اجرا می‌شوند و آن‌ها را Transparent، Immutable و Trustless می‌سازند—به این معنا که هیچ طرفی نمی‌تواند نتیجه را پس از Deployed شدن تغییر دهد.

کاربردهای آن در IoT:

- Automate device interactions (مثال: اگر مقدار $Sensor X > Threshold$ شد، Actuator Y فعال شود)
- Manage access control rules
- Facilitate secure data exchange and micro-transactions

IoT BMI Tool

ابزار IoT Business Model Innovation (BMI) به این دلیل طراحی شد که چارچوب‌های سنتی (مثل St. Galler Magic و Triangle و Business Model Canvas) قادر به پوشش کامل پیچیدگی‌های خاص اکوسیستم‌های IoT نیستند.



این ابزار دو دیدگاه را با هم ترکیب می کند:

IoT Architecture Layers شامل Device, Connectivity, Cloud, Application, و Service (که تمرکز آن بر خلق ارزش پس از پیاده سازی است).

BMC Building Blocks شامل Value, Customer Segments, Cost Structure, Resources, Activities, Partners, Revenue Streams, و Channels, Relationships, Proposition.

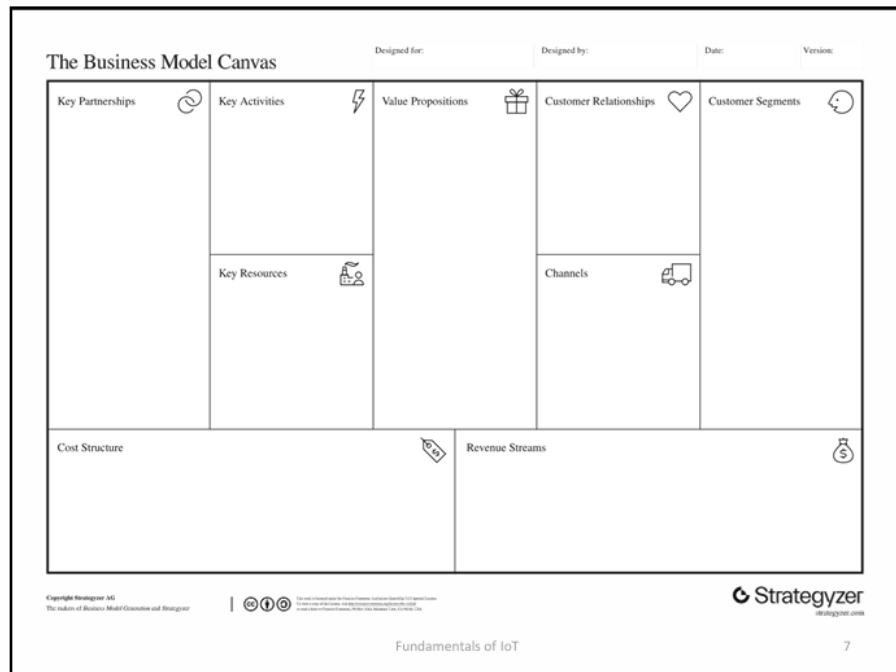
نواقص Magic Triangle



St. Galler Magic Triangle (شامل WHO، WHAT، HOW، و WHY) اگرچه چارچوبی زیباست، اما:

- بیش از حد سطح بالا و انتزاعی است و برای اکوسیستم های پیچیده IoT جزئیات کافی ندارد.
- لایه های فنی مثل Connectivity, Devices, یا Cloud Services را در نظر نمی گیرد.
- لایه ی Service بعد از استقرار (مثل Update, Maintenance, Analytics) غایب است.

نواقص Business Model Canvas (BMC)



اگرچه BMC جزئیات بیشتری دارد، اما:

- بیشتر روی ساختارهای عمومی کسب‌وکار تمرکز دارد، نه IoT.
- روی Data Monetization و Continuous Services که برای IoT حیاتی‌اند، تمرکز نمی‌کند.
- جنبه‌های بعد از پیاده‌سازی (مثل Software Updates، Predictive Maintenance، ارزش‌آفرینی مداوم) را پوشش نمی‌دهد.

چگونه IoT BMI Tool این شکاف‌ها را پر می‌کند؟

- لایه‌های IoT Architecture را به‌طور صریح وارد مدل کسب‌وکار می‌کند.
- لایه Service را اضافه می‌کند تا ارزش‌آفرینی پس از عرضه محصول نیز پوشش داده شود.

بخش دوم





(الف)

تینگزبرد را با استفاده از داکر و با استفاده از این تنظیمات بالا آوردیم:

```
1  services:
2    tb:
3      image: thingsboard/tb-postgres
4      restart: always
5      ports:
6        - "8080:9090"
7        - "1883:1883"
8        - "5683:5683/udp"
9      environment:
10       TB_QUEUE_TYPE: in-memory
11       SPRING_DATASOURCE_URL: jdbc:postgresql://postgres:5432/thingsboard
12       SPRING_DATASOURCE_USERNAME: postgres
13       SPRING_DATASOURCE_PASSWORD: postgres
14       SECURITY_CLAIM_ALLOW_CLAIMING_BY_DEFAULT: "true"
15       SECURITY_OAUTH2_ENABLED: "false"
16       TB_SKIP_INSTALL: "true"
17       JAVA_OPTS: "-Xms256M -Xmx512M"
18     depends_on:
19       - postgres
20
21   postgres:
22     image: postgres:12
23     restart: always
24     environment:
25       POSTGRES_DB: thingsboard
26       POSTGRES_PASSWORD: postgres
27       POSTGRES_USER: postgres
28   volumes:|
29     - pg_data:/var/lib/postgresql/data
30
31 volumes:
32   pg_data:
```

(ب)

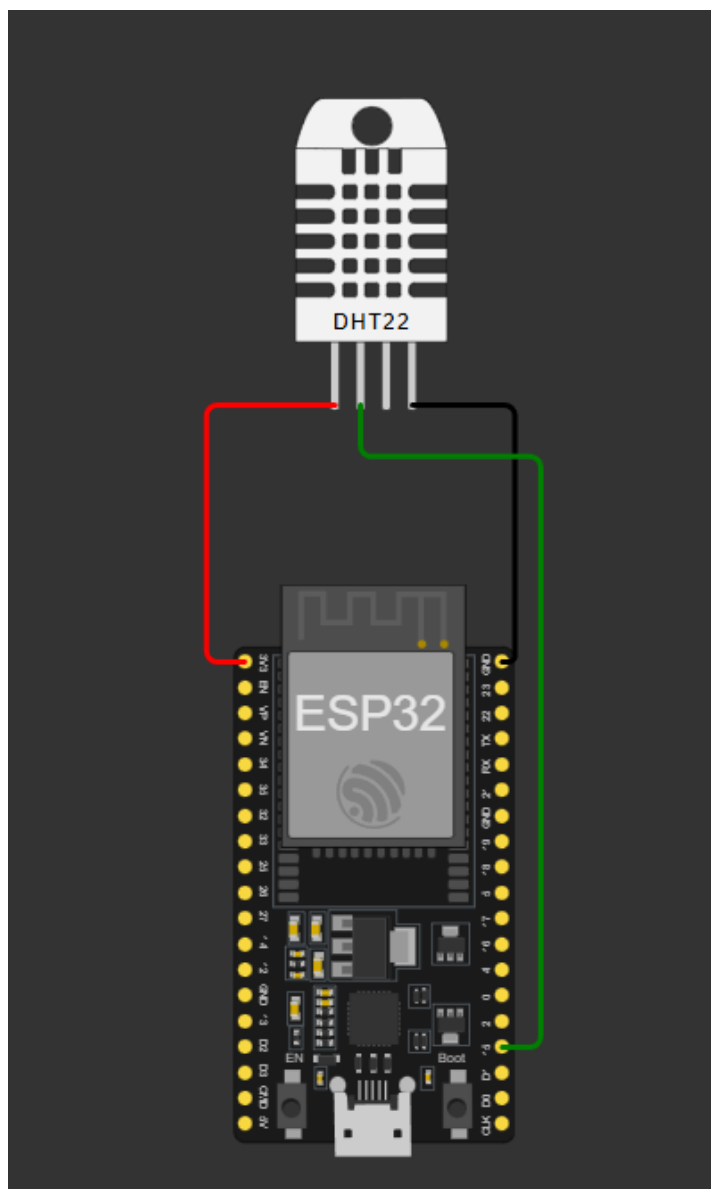
در بخش devices یک دستگاه جدید ایجاد کردیم:

<input type="checkbox"/>	Created time	Name	Device profile ↑	Label	State	Customer	Public	Is gateway	
<input type="checkbox"/>	2025-08-19 14:33:17	ESP32_DHT22_Q2	default		Inactive		<input type="checkbox"/>	<input type="checkbox"/>	   

access token: P6uskj7vhu55ZOEHsyTx

(ج)

در wokwi مدار خواسته شده را درست کردیم:



کد استفاده شده برای ESP32 برای فرستادن داده به تینگزپورد:

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <DHT.h>

#define DHTPIN 15
#define DHTTYPE DHT22

const char* ssid      = "Wokwi-GUEST";
const char* password   = "";
const char* mqttServer = "host.wokwi.internal";
const int   mqttPort   = 1883;
const char* token      = "P6uskj7vhu55Z0EHsyTx";

WiFiClient espClient;
PubSubClient mqtt(espClient);
DHT dht(DHTPIN, DHTTYPE);

void reconnect() {
  while (!mqtt.connected()) {
    Serial.print("Connecting to MQTT...");
    if (mqtt.connect("ESP32Client", token, nullptr)) {
      Serial.println("connected!");
    } else {
      Serial.print("failed, rc=");
      Serial.print(mqtt.state());
      Serial.println(" try again in 3 seconds");
      delay(3000);
    }
  }
}
```

```
void setup() {
  Serial.begin(115200);
  dht.begin();

  WiFi.begin(ssid, password, 6);
  while (WiFi.status() != WL_CONNECTED) {
    delay(100);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");

  mqtt.setServer(mqttServer, mqttPort);
}

void loop() {
  if (!mqtt.connected()) reconnect();
  mqtt.loop();

  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (!isnan(temperature) && !isnan(humidity)) {
    String payload = "{\"temperature\": " + String(temperature) +
      ", \"humidity\": " + String(humidity) + "}";
    mqtt.publish("v1/devices/me/telemetry", payload.c_str());
    Serial.println("Published: " + payload);
  } else {
    Serial.println("Failed to read from DHT");
  }

  delay(5000);
}
```

از توکن بدست آمده در قسمت قبل استفاده کردیم که به device ساخته شده دسترسی داشته باشیم.

(۵)

ESP32_DHT22_Q2

Device details

?

×

Details

Attributes

Latest telemetry

Calculated fields

Alarms

Events

Relations

Audit logs

Version control

Telemetry

+

🔍

☐

Last update time

Key ↑

Value

☐

2025-08-19 14:58:58

humidity

40.0

🗑

☐

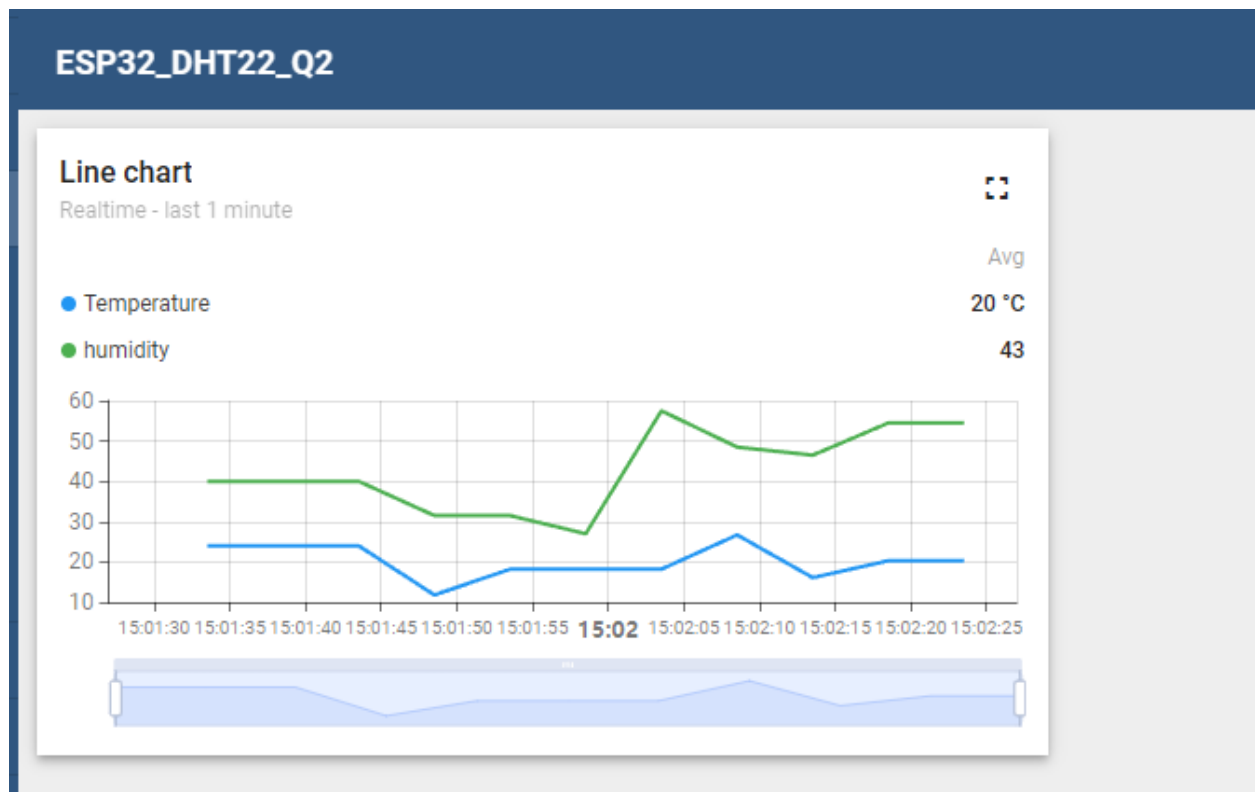
2025-08-19 14:58:58

temperature

24.0

🗑

(۶)



بخش سوم

(الف)

دستگاه جدید ساخته شده:

Device profiles						
<input type="checkbox"/>	Created time ↓	Name	Profile type	Transport type	Description	Default
<input type="checkbox"/>	2025-08-19 15:05:22	ESP32_DHT22_Provisioning_Q3	Default	Default		<input type="checkbox"/>
	2025-08-19 13:51:53	default	Default	Default	Default device profile	<input checked="" type="checkbox"/>

قسمت provisioning دستگاه:

ESP32_DHT22_Provisioning_Q3

Device profile details

Details

Transport configuration

Calculated fields

Alarm rules

Device provisioning

Audit logs

Version control

Provision strategy*

Allow to create new devices

Provision device key*

jpe1o0wus2pxfutdbcra

Provision device secret*

yk2hn65xf6tio0of9jwv

(ب)

تابع اضافه شده برای اجرای provisioning دستگاه جدید:

```
52 bool runProvisioning() {
53     provisionMqtt.setServer(THINGSBOARD_SERVER, THINGSBOARD_PORT);
54     provisionMqtt.setCallback(onProvisionMessage);
55     if (!provisionMqtt.connect("esp32-provision", "provision", "")) {
56         Serial.println("Provision MQTT connect failed");
57         return false;
58     }
59     if (!provisionMqtt.subscribe("/provision/response")) {
60         Serial.println("Subscribe to /provision/response failed");
61         return false;
62     }
63
64     StaticJsonDocument<256> req;
65     req["deviceName"] = DEVICE_NAME;
66     req["provisionDeviceKey"] = PROV_KEY;
67     req["provisionDeviceSecret"] = PROV_SECRET;
68     char buf[256];
69     size_t n = serializeJson(req, buf, sizeof(buf));
70
71     if (!provisionMqtt.publish("/provision/request", buf, n)) {
72         Serial.println("Publish /provision/request failed");
73         return false;
74     }
75
76     uint32_t start = millis();
77     while (!provResponseArrived && millis() - start < 8000) {
78         provisionMqtt.loop();
79         delay(10);
80     }
81     provisionMqtt.disconnect();
82
83     const char* status = provResponseDoc["status"] | provResponseDoc["provisionDeviceStatus"];
84     if (!status || strcmp(status, "SUCCESS") != 0) {
85         Serial.print("Provision failed, status=");
86         Serial.println(status ? status : "null");
87         return false;
88     }
89
90     const char* type = provResponseDoc["credentialsType"] | "";
91     const char* value = provResponseDoc["credentialsValue"] | provResponseDoc["accessToken"] | "";
92     if (strcmp(type, "ACCESS_TOKEN") != 0 || strlen(value) == 0) {
93         Serial.println("Unexpected credentials payload");
94         return false;
95     }
96
97     strncpy(DYNAMIC_TOKEN, value, sizeof(DYNAMIC_TOKEN) - 1);
98     return true;
99 }
```



<input type="checkbox"/>	Created time ↓	Name	Device profile	Label	State	Customer
<input type="checkbox"/>	2025-08-19 15:35:16	ESP32-DHT22-001	ESP32_DHT22_Provisioning_Q3		Active	

ESP32-DHT22-001

Device details



DetailsAttributesLatest telemetryCalculated fieldsAlarmsEventsRelationsAudit logsVersion control

Telemetry



<input type="checkbox"/>	Last update time	Key ↑	Value	
<input type="checkbox"/>	2025-08-19 15:36:03	humidity	40	
<input type="checkbox"/>	2025-08-19 15:36:03	temperature	24	



Customers



<input type="checkbox"/>	Created time ↓	Title	Email	Country	City	
<input type="checkbox"/>	2025-08-19 15:36:43	Customer Q3				

بعد از لاگین کردن توسط مشتری:

Device claiming widget

Device name*

ESP32-DHT22-001

Secret key*

123

Claim device

Line chart

Realtime - last 1 minute

No data to display on widget

ESP32_DHT22_Q3

Device claiming widget

Device name*

Secret key*

Claim device

Line chart

Realtime - last 1 minute

No data to display on widget

پس از وارد کردن اطلاعات:

