

## การใช้งาน ThingsBoard IoTs Platform เพื่อสร้างและจัดการระบบอัจฉริยะ ThingsBoard IoTs Platform for smart system

### 2/4 – ThingsBoard IoTs Platform

- IoTs Platform
- Best IoT Platforms for building IoT projects
- การโปรแกรมเพื่อแสดงค่าจาก ESP32 ไปที่ ThingsBoard
- การโปรแกรมเพื่อควบคุมและแสดงค่าระหว่าง ESP32 และ ThingsBoard
- โครงการงาน “ตรวจวัดอุณหภูมิความชื้นของฟาร์มไก่ ให้มีการควบคุมอุปกรณ์และมีการแจ้งเตือน”
- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

### 1/6 -- IoTs Platform

<https://www.scimath.org/article-technology/item/9084-2018-10-18-07-45-35>

ประเทศไทย 4.0 โมเดลขับเคลื่อนประเทศไทยสู่ความมั่นคง มั่งคั่ง และยั่งยืนนั้น เป็นยุคเทคโนโลยี Creative และ Innovation เน้นการสร้างให้คนไทยสามารถคิดเองได้ การที่จะให้บุคลากรในประเทศได้ใช้เทคโนโลยีที่เหมาะสมในการบูรณาการกับความรู้ทางด้านวิทยาศาสตร์และคณิตศาสตร์ เพื่อแก้ปัญหาหรือพัฒนางานด้วยกระบวนการออกแบบเชิงวิศวกรรม ที่นำไปสู่ การคิดค้นสิ่งประดิษฐ์ หรือสร้างนวัตกรรมต่าง ๆ ที่เอื้อประโยชน์ต่อการดำรงชีวิตได้นั้น จำเป็นต้องมีระบบนิเวศน์ (Ecosystem) ด้าน IoT (Internet of Things) ซึ่งเป็นเทคโนโลยีหนึ่งที่เป็นตัวผลักดันขับเคลื่อน ประเทศไทย 4.0

#### 1.1 IoT Platform คืออะไร

IoT Platform เป็นองค์ประกอบที่สำคัญของระบบนิเวศของ IoT แต่หลายคนไม่ทราบว่า IoT Platform คืออะไร ความแตกต่างระหว่างแต่ละคืออะไร

ดังนั้น IoT Platform คืออะไร มาดูกันแต่ละข้อ

1. ระบบ IoT ที่สมบูรณ์ต้องใช้ฮาร์ดแวร์ เช่น เซ็นเซอร์หรืออุปกรณ์ เซ็นเซอร์และอุปกรณ์เหล่านี้รวบรวมข้อมูลจากสิ่งแวดล้อม (เช่น เซ็นเซอร์วัดความชื้น) หรือดำเนินการในสิ่งแวดล้อม (เช่น การรดน้ำพืช)
2. ระบบ IoT ที่สมบูรณ์แบบต้องการการเชื่อมต่อ ฮาร์ดแวร์ต้องมีวิธีการส่งข้อมูลทั้งหมดไปยังระบบคลาวด์ (เช่น การส่งข้อมูลความชื้น) หรือต้องการวิธีรับคำสั่งจากระบบคลาวด์ (เช่น ให้น้ำตอนนี้) สำหรับระบบ IoT บางระบบอาจมีขั้นตอนกลางระหว่างฮาร์ดแวร์และการเชื่อมต่อกับระบบคลาวด์เช่นเกตเวย์หรือเราเตอร์
3. ระบบ IoT ที่สมบูรณ์แบบต้องการซอฟต์แวร์ ซอฟต์แวร์นี้เป็นโฮสต์ในระบบคลาวด์ และมีหน้าที่ในการวิเคราะห์ข้อมูลที่รวบรวมจากเซ็นเซอร์และการตัดสินใจ (เช่น รู้จากข้อมูลความชื้นที่ฝนตกและบอกระบบชลประทานไม่ให้เปิดวันนี้)
4. สุดท้ายระบบ IoT ที่สมบูรณ์แบบจำเป็นต้องมีส่วนติดต่อผู้ใช้ เพื่อให้สิ่งนี้มีประโยชน์ต้องมีวิธีสำหรับผู้ใช้โต้ตอบกับระบบ IoT (เช่น แอปบนเว็บพร้อมแดชบอร์ดที่แสดงแนวโน้มความชื้นและช่วยให้ผู้ใช้สามารถเปิดหรือปิดระบบชลประทานด้วยตนเองได้)

IoT Platform เป็นซอฟต์แวร์สนับสนุนที่เชื่อมต่อทุกอย่างในระบบ IoT แพลตฟอร์ม IoT ช่วยให้การสื่อสารการไหลของข้อมูลการจัดการอุปกรณ์และการทำงานของแอปพลิเคชัน IoT Platform มีอยู่ใน 3 ส่วนแรก และมักเป็นส่วนที่ 4 ของสิ่งที่อธิบายข้างต้น กับทุกชนิดที่แตกต่างกันของฮาร์ดแวร์และตัวเลือกการเชื่อมต่อที่แตกต่างกันจะต้องมีวิธีการทำให้ทุกอย่างทำงานร่วมกัน แพลตฟอร์ม IoT ช่วยแก้ปัญหาดังกล่าว

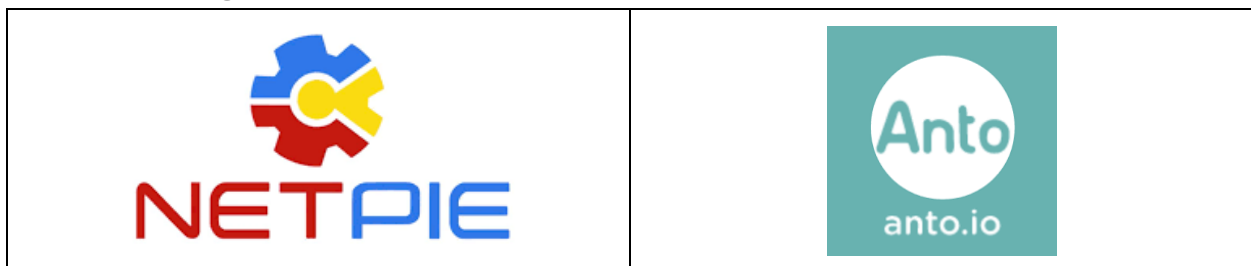
## 1.2 IoT Platform ช่วยให้

- เชื่อมต่อฮาร์ดแวร์เซ็นเซอร์และอุปกรณ์
- จัดการโปรโตคอลการสื่อสารซอฟต์แวร์และฮาร์ดแวร์ที่แตกต่างกัน
- ให้การรักษาความปลอดภัยและการตรวจสอบสิทธิ์สำหรับอุปกรณ์และผู้ใช้
- เก็บภาพและวิเคราะห์ข้อมูลที่เซ็นเซอร์และอุปกรณ์ต่าง ๆ รวบรวม
- ผสานรวมทั้งหมดข้างต้นกับบริการเว็บอื่น ๆ

## 1.3 IoT Platform ฝีมือคนไทย

ในปัจจุบันมี IoT Platform ที่พัฒนาโดยคนไทย ให้เลือกใช้งานมากมาย เช่น NETPIE, Anto.io (Anto IoT platform), ioTtweet, AIS NB-IoT และ True NB-IoT

- **NETPIE:** เป็นผลงานวิจัยของ ศูนย์เทคโนโลยีอิเล็กทรอนิกส์และคอมพิวเตอร์แห่งชาติ (NECTEC) ที่สนับสนุน Arduino Platform และ Raspberry Pi โดยใช้ชุดไลบรารีที่ชื่อว่า Microgear โดยมีหน้าจอตouchสำหรับสร้างเครื่องมือควบคุมได้
- **Anto.io (Anto IoT platform):** เป็น IoT Platform ที่ทำให้เราสามารถเรียนรู้และเข้าใจเกี่ยวกับความรู้พื้นฐานเบื้องต้นเกี่ยวกับ IoT ได้อย่างรวดเร็ว ด้วยความที่ Anto นั้นได้เรียงลำดับในการอธิบายได้ดี ซึ่งเริ่มจาก Introduction จะพูดถึง Anto มีหน้าที่ทำอะไร และรองรับอุปกรณ์ใดบ้าง พร้อมทั้งมี Prepare และ Quick Start ที่มีขั้นตอนการอธิบายที่เรียบง่าย ทำให้เราสามารถใช้เวลาทำตามได้อย่างรวดเร็ว แต่ Anto นี้สามารถรองรับได้เพียง ESP8266 และ Arduino เท่านั้น จุดเด่นหลักของ Anto ก็คือมี Pattern การสอนที่เรียบง่าย ทำให้ตามได้อย่างรวดเร็วจนรู้สึกว้าว!! เสร็จแล้วหรอ นอกจากนี้ Anto ยังมี Dashboard ในตัว ซึ่งมีความสนุกตรงที่ Dashboard นั้นมีความสวยงามและยืดหยุ่นมาก พร้อมทั้งยังสามารถดูค่าและปรับค่าของ Thing ได้ จากรูปลักษณ์แล้วทำให้ผู้ใช้งานเข้าถึงได้ง่ายมาก และหน้าตาออกมาใช้งาน พร้อมทั้งสามารถเปิดใช้ผ่านมือถือได้ นั่นคือเราสามารถเปิดเว็บ Anto แล้ว Login เข้ามาในหน้านี้ก็สามารถควบคุมอุปกรณ์ได้
- **ioTtweet:** เป็นผลงานของคุณ อิศราณ จันทร์ทอง สามารถใช้งาน สนับสนุน Arduino Platform และ Raspberry Pi โดยมีหน้าจอตouchใช้งานง่ายไม่เก๋ไก๋ที่ ก็สามารถควบคุมอุปกรณ์ในการรับ-ส่งข้อมูลได้แล้ว
- **AIS NB-IoT:** เป็นผลิตภัณฑ์ของบริษัท แอดวานซ์ อินโฟร์ เซอร์วิส จำกัด (มหาชน) หรือ AIS ที่มี platform ที่ใช้งานชื่อว่า Magellan
- **True NB-IoT:** เป็นผลิตภัณฑ์ของบริษัท ทูริ ดิจิตอล แอนด์ มีเดีย แพลตฟอร์ม จำกัด ที่มี platform ที่ใช้งานชื่อว่า Thingsboard



#### 1.4 นำ IoT Platform ไปใช้ในด้านใดบ้าง

##### NETPIE

- ชุด KidBright เชื่อมต่อInternet of things (IoT) ผ่านแอปพลิเคชันบนสมาร์ตโฟนด้วยKidBright App (Android)
- บุคคลธรรมดาหรือเอกชนที่นำไปใช้ในงานตนเองการเรียนการสอนการเชื่อมต่ออุปกรณ์ในรูปแบบ IoT (Internet of Things)

##### Anto.io (Anto IoT platform)

- ระบบงานต่างๆ ที่ใช้งานด้าน IoT (Internet of Things)
- บุคคลธรรมดาหรือเอกชนที่นำไปใช้ในงานตนเอง
- การเรียนการสอนการเชื่อมต่ออุปกรณ์ในรูปแบบ IoT (Internet of Things)

##### ioTtweet

- ระบบงานต่างๆ ที่ใช้งานด้าน IoT (Internet of Things)
- บุคคลธรรมดาหรือเอกชนที่นำไปใช้ในงานตนเอง
- การเรียนการสอนการเชื่อมต่ออุปกรณ์ในรูปแบบ IoT (Internet of Things)

##### AIS NB-IoT

- บมจ.ปตท. : นำ IoT ไปตรวจสอบท่อส่งก๊าซธรรมชาติ ผ่านเซ็นเซอร์ที่จะส่งข้อมูลให้ผู้ควบคุมได้ทันที
- บมจ.พร้อมเพอร์ดี เพอร์เฟค : นำ IoT ไปยกระดับใน 15 โครงการที่ทักอาศัยให้เป็นรูปแบบ Smart City
- โคตรรถกรู๊ป : ผู้ผลิต และจำหน่ายเครื่องหยอดเหรียญรูปแบบต่าง ๆ ประยุกต์ใช้ IoT กับเซ็นเซอร์ตรวจจับเหรียญภายในเครื่อง เมื่อเหรียญเต็มก็สามารถไปเก็บได้ทันเวลา
- มหาวิทยาลัยสงขลานครินทร์ นำเซ็นเซอร์ตรวจวัดสภาพอากาศ และบริหารจัดการน้ำ ไปใช้งานจริงที่เทศบาลนครหาดใหญ่ จังหวัดสงขลาและจังหวัดภูเก็ต
- มหาวิทยาลัยขอนแก่น นำ Smart Trash Bin แก้ไขปัญหาขยะล้นถัง ด้วยเซ็นเซอร์ที่ส่งสัญญาณภายใต้ IoT Platform
- MoBike ที่สามารถรายงานตำแหน่งของจักรยานจำนวนมากได้ว่าจอดอยู่ตำแหน่งใด ทำให้ศูนย์กลางสามารถติดตามจักรยานและจัดการวางจักรยานตามความต้องการของผู้ใช้ได้

##### True NB-IoT

- ระบบงานต่าง ๆ ที่ใช้งานด้าน IoT (Internet of Things)
- บุคคลธรรมดาหรือเอกชนที่นำไปใช้ในงานตนเอง

## 2/6 -- Best IoT Platforms for building IoT projects

<https://iotbyhvm.ooo/best-iot-platforms/>



In this post You can find out a list of the Best IoT platforms. Today, Internet of things (IoT) is one of the fastest growing industries. IoT enabled Devices are around us such as Smart watches and wearables, Smart homes and so on. An IoT platform offers several services that simplify the project development and a set of tools to remotely manage devices. Generally speaking, an IoT platform is a multi-layer technology that enables users to manage connected devices.

*Below there is a list of Best IoT platforms in a random order. Some of these platforms have a free account and some have a premium account that enables other interesting features.*

Basically, An IoT Platform provides these services:

- Data ingestion
- Dashboard creation
- Data transformation
- Device management
- Rule management
- Platform integration
- Security services

## 2.1 Best IoT Platforms

### ThingSpeak

ThingSpeak is an IoT analytics platform service that allows you to aggregate, visualize, and analyze live data streams in the cloud. You can send data to ThingSpeak™ from your devices, create instant visualizations of live data, and send alerts using web services like Twitter® and Twilio®. With MATLAB® analytics inside ThingSpeak, you can write and execute MATLAB code to perform preprocessing, visualizations, and analyses. ThingSpeak enables engineers and scientists to prototype and build IoT systems without setting up servers or developing web software.

#### Features:

- Realtime sensor data visualization
- Data aggregation from 3rd parties providers
- Schedule IoT analytics tasks to analyze data
- Event scheduling
- Run actions according to data acquired

### Kaa

Kaa is an enterprise-grade IoT platform built on a modern cloud-native architecture and a fully customizable feature set. Based on flexible microservices, Kaa easily adapts to almost any need and application. It scales from a tiny start-up to a massive corporation and supports advanced deployment models for multicloud IoT solutions. But you can also use it to put together a smart thermostat for your living room. As long as it's IoT, it's Kaa.

#### Features:

- Device connectivity
- Device management
- Data collection
- Data processing and analysis
- Data visualization
- Command execution

### Adafruit IO

Adafruit.io is a *cloud service* – that just means we run it for you and you don't have to manage it. You can connect to it over the Internet. It's meant primarily for storing and then retrieving data but it can do a lot more than just that!

#### Features:

- Display your data in real-time, online
- Make your project internet-connected: Control motors, read sensor data, and more!
- Connect projects to web services like Twitter, RSS feeds, weather services, etc.
- Connect your project to other internet-enabled devices
- The best part? All of the above is do-able for **free** with Adafruit IO

**AWS IoT**

AWS IoT (Amazon internet of things) is an Amazon Web Services platform that collects and analyzes data from internet-connected devices and sensors and connects that data to AWS cloud applications.

AWS offers a set of services:

- Amazon FreeRTOS: This is an OS for microcontrollers that provides some services as connectivity, security, easy programming
- AWS Greengrass: It is a software that enables to run local computation on edge devices
- AWS IoT Analytics: It enables us to run sophisticated analytics on gathered data
- AWS IoT device management: It simplifies the process of device management especially when there are thousands of connected devices
- AWS IoT Core: It is the core of the AWS IoT and enables connected devices to interact with cloud services easily and securely

**Google Cloud IoT**

Google Cloud IoT is one of the most interesting enterprise platforms. This IoT platform has a set of tools to manage connected devices and the data at the edge level or in the cloud. The connected device can use cloud Pub/sub to publish the data. Moreover, we can apply BigQuery analysis or we can apply Machine learning on this data. Google Cloud IoT has a reference architecture that describes the role of each component that builds this platform. There are countless possibilities to explore using professional services. This platform has all the services an IoT platform must have starting from the security aspects.

### Microsoft Azure IoT

Microsoft Azure IoT is another IoT platform. This is a professional platform with several services. It supports bi-directional communication between connected devices and the platform itself using IoT standard protocols. Moreover, it supports device authentication to address all the security aspects. Microsoft Azure IoT simplify the process of IoT project development addressing all the challenges we have to face during this process starting from the security aspects.

### Artik Cloud

Artik Cloud is an IoT platform developed by Samsung. This platform enables devices to connect each other and connect to cloud services. It has a set of services to rapidly connect devices to the cloud and start gathering data. Moreover, this IoT platform has a set of connectors that can be used to connect to third-party services. Like other platforms, it is possible to store data coming from connected devices and aggregate this information.

### IBM Watson IoT

IBM Watson IoT Platform (Connection Service and Analytics Service) is a ready-to-run, pre-integrated SaaS managed service IoT platform with capabilities in connectivity, data management and advanced analytics.

#### Features:

- Device management: Using this service, it is possible to act remotely on the device such as rebooting or firmware update
- Responsive, scalability, connectivity: The platform uses industry standard protocol MQTT to exchange data
- Secure communication: Secure data exchange using MQTT and TLS
- Data lifecycle management

### TheThings.io

thethings.io is an another IoT platform that lets any kind of companies to deploy scalable and flexible IoT solutions for their customers and connected products.

Features:

- CONNECTIVITY AND NORMALIZATION
- DEVICES & LICENSES MANAGEMENT
- CLOUD CODE PROCESSING & ACTION MANAGEMENT
- DATA MONITORING & VISUALIZATIONANALYTICS, AI, PREDICTIVE & OTHER TOOLS
- INTEROPERABILITY AND INTEGRATIONS

### myDevices Cayenne

Cayenne is the world's first drag and drop IoT project builder that empowers developers, designers and engineers to quickly prototype and share their connected device projects. Cayenne was designed to help users create Internet of Things prototypes and then bring them to production.

There are several major components in the platform:

- **Cayenne Mobile Apps** – Remotely monitor and control your IoT projects from the Android or iOS Apps.
- **Cayenne Online Dashboard** – Use customizable widgets to visualize data, set up rules, schedule events and more.

### Ubidots

Ubidots technology and engineering stack was developed to deliver a secure, white-glove experience for our users. Device friendly APIs (accessed over HTTP/MQTT/TCP/UDP protocols) provide a simple and secure connection for sending and retrieving data to and from our cloud service in real-time. Ubidots' time-series backend services are performance optimized for IoT data storage, computation, and retrieval. Our application enablement platform supports interactive, real-time data visualization (widgets), and an IoT App Builder that allows developers to extend the platform with their own HTML/JS code for private customization when desired. Ubidots exists to empower your data from device to visualization.

### Temboo

This platform uses **choreos** that are connectors toward external services, so that events in Arduino, like sensor signals, can be transformed into different kind of events. Moreover, it provides some logic like IF-THEN. Moreover, Temboo supports M2M applications using MQTT, CoAP, HTTP protocols. The main features provided by Temboo are:

- **Code generation:** This platform generates optimized code for several devices using many different languages such as Java, C/C++, Python and so on
- **Interoperability:** Temboo provides a set of services named choreos that simplify the integration process with other cloud services
- **Data:** Temboo stores and visualizes different kinds of data.



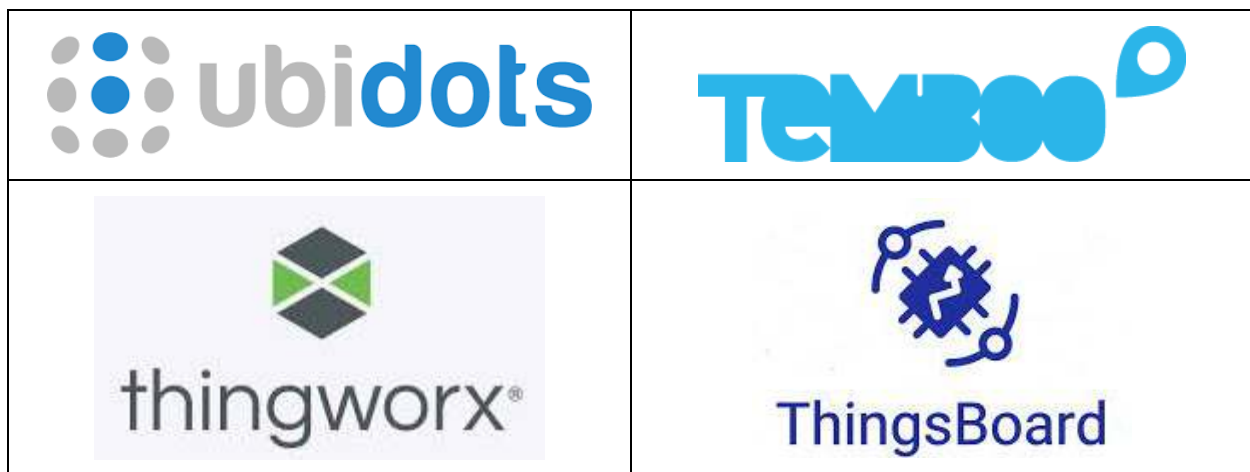
**ThingWorx**

The ThingWorx low-code IoT development environment provides you with the flexibility to rapidly connect, create, and deploy comprehensive industrial IoT applications. With pre-built extensions and widgets as well as a large ecosystem of partners, ThingWorx addresses the fundamental IoT development challenges, freeing you up to create solutions that accommodate constantly changing business needs.

**ThingsBoard**

ThingsBoard is an open-source IoT platform for data collection, processing, visualization, and device management

It enables device connectivity via industry standard IoT protocols - MQTT, CoAP and HTTP and supports both cloud and on-premises deployments. ThingsBoard combines scalability, fault-tolerance and performance so you will never lose your data.

**2.2 You may also like:**

- [Dynamic WLAN configuration for ESP32 Board / AutoConnect](#)
- [ESP32 BLE on Arduino IDE with UART Test](#)
- [ESP32 Bluetooth Low Energy \(BLE\) on Arduino IDE](#)
- [ArduinoOTA ESP32: Wi-Fi \(OTA\) Wireless Update from the Arduino IDE](#)
- [ESP32 with LoRa using Arduino IDE](#)
- [How To Use Grove-LCD RGB Backlight with NodeMCU](#)
- [NodeMcu to DHT Interface in Blynk app](#)
- [How To ON/OFF a bulb by Google voice assistant](#)
- [Arduino IDE / Arduino / Open Source Hardware/Softawre / Arduino Vs RPi](#)
- [WiFi LoRA 32 \(V2\) ESP32 / Overview / Introduction](#)
- [DHT11 sensor with ESP8266/NodeMCU using Arduino IDE](#)
- [Arduino Support for ESP8266 with simple test code](#)

## 2.3 The 7 thoughts on “Best IoT Platforms for building IoT projects”

- Pingback: [Best IoT Platforms for building IoT projects — IoTbyHVM – Bits & Bytes of IoT – hashstacks](#)
- Pingback: [W600-PICO | A W600-Based Board Running MicroPython for Only \\$2?](#)
- Pingback: [What is Azure Sphere ? - IoTbyHVM - Bits & Bytes of IoT](#)
- Pingback: [What is IoT ? | Internet of Things - IoTbyHVM - Bits & Bytes of IoT](#)
- Pingback: [Go Programming Language | Installation | Gobot Framework](#)
- Pingback: [How to setup a Mosquitto MQTT Server and receive data from OwnTracks](#)
- Pingback: [Best IoT Apps for Android - IoTbyHVM - Bits & Bytes of IoT](#)

## 2.4 Read More

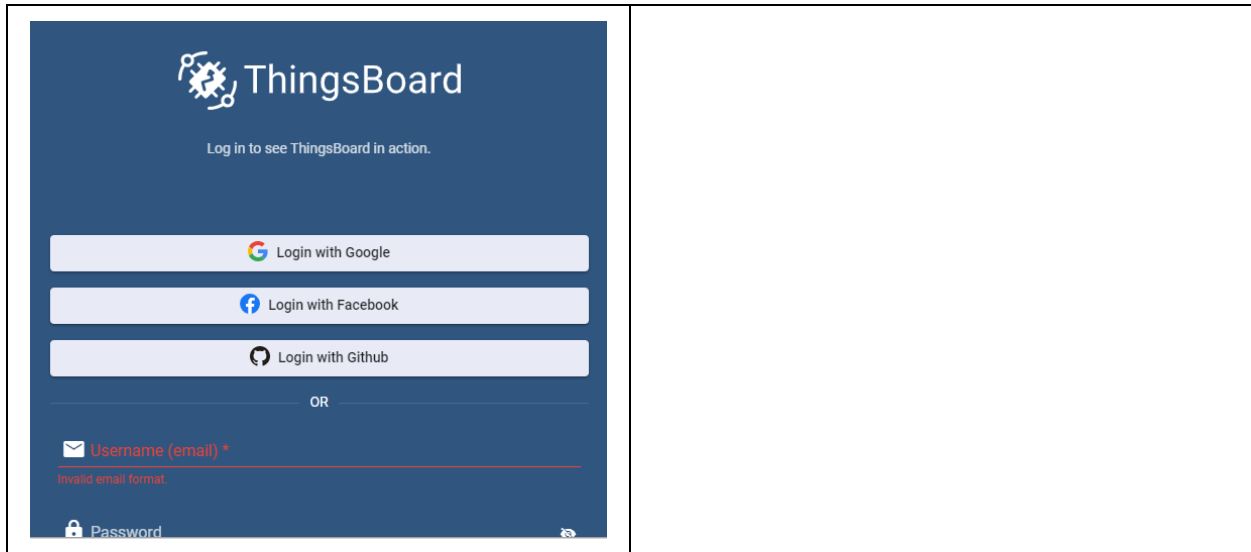
- <http://nilhcm.com/iot/cloud-iot-core-with-the-esp32-and-arduino>
- <https://ludovic-emo.medium.com/how-to-send-esp32-telemetry-to-google-cloud-iot-core-caf1a952020d>
- <https://cloud.google.com/community/tutorials/monitoring-iot-data-grafana>
- <https://www.survivingwithandroid.com/cloud-iot-core-esp32/>
- Cayenne <https://developers.mydevices.com/cayenne/docs/cayenne-mqtt-api/#cayenne-mqtt-api-using-arduino-mqtt>

### 3/6 -- การโปรแกรมเพื่อแสดงค่าจาก ESP32 ไปที่ ThingsBoard

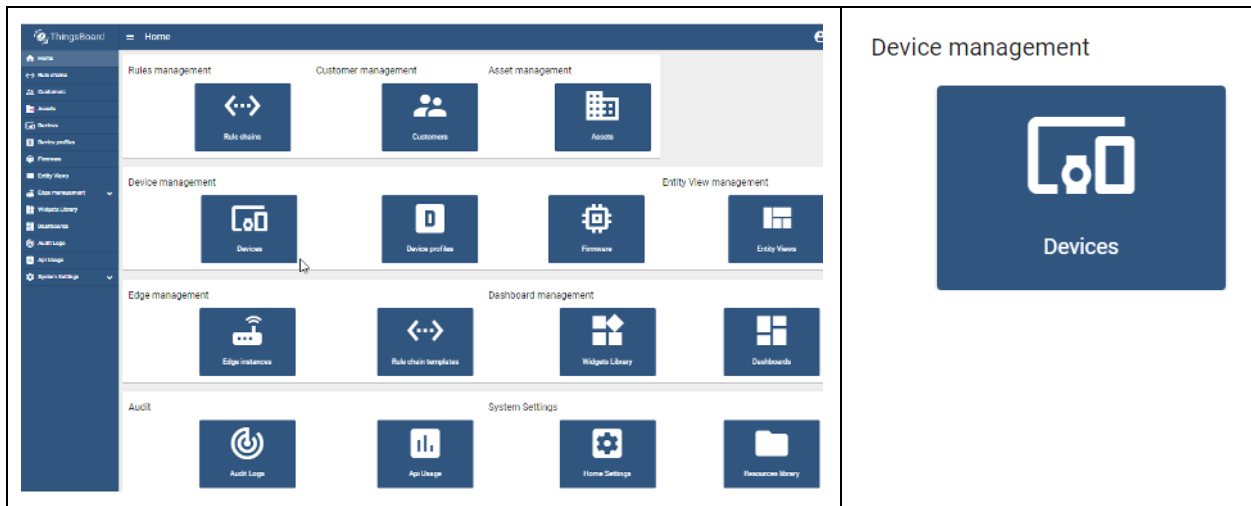
<https://thingsboard.io/docs/samples/esp8266/gpio/>  
<https://thingsboard.io/docs/samples/esp32/gpio-control-pico-kit-dht22-sensor/>

#### Lab201 – ThingsBoard data monitor

1. เข้าสู่ระบบ ที่ Web Site Live Demo ของ ThingsBoard <https://demo.thingsboard.io/>



2. เข้ามาหน้า Main ของ ThingsBoard → เลือก Device .=ใน Device Management เพื่อสร้าง Device



3. สร้าง Device ด้วยการกดเครื่องหมาย + แล้วกด Add New Device → ให้กรอกข้อมูลของ Device ดังตัวอย่าง คือ Name = PkTest1234, Type = Default, กด Add

4. จะปรากฏ Device PkTest1234 ที่หน้าหลัก → Click ที่ Device นี้ แล้วเลือก Copy Access Token Code

5. หรือสามารถเรียกดู Access Token ได้จากไอคอน Manage Credential ได้เช่นกัน

## 6. ติดตั้ง Arduino Library

- Add Library “ThingsBoard by ThingsBoard Team -- V 0.4.0”
- Add Library “ArduinoHttpClient by Arduino -- V 0.4.0”
- Add Library “ArduinoJson by Benoit Blanchon – V6.18.0”

The screenshot shows the Arduino IDE Library Manager interface. At the top, the search bar contains 'thingsboard'. Below the search bar, three libraries are listed, each with its name, author, version, and status (INSTALLED).

- ThingsBoard** by ThingsBoard Team Version 0.4.0 **INSTALLED**. Description: ThingsBoard library for Arduino. A library for connecting to the ThingsBoard IoT platform. Thin wrapper on PubSubClient. [More info](#)
- ArduinoHttpClient** by Arduino Version 0.4.0 **INSTALLED**. Description: [EXPERIMENTAL] Easily interact with web servers from Arduino, using HTTP and WebSocket's. This library can be used for HTTP (GET, POST, PUT, DELETE) requests to a web server. It also supports exchanging messages with WebSocket servers. Based on Adrian McEwen's HttpClient library. [More info](#)
- ArduinoJson** by Benoit Blanchon Version 6.18.0 **INSTALLED**. Description: A simple and efficient JSON library for embedded C++. ArduinoJson supports ✓ serialization, ✓ deserialization, ✓ MessagePack, ✓ fixed allocation, ✓ zero-copy, ✓ streams, ✓ filtering, and more. It is the most popular Arduino library on GitHub ♥♥♥♥♥. Check out arduinojson.org for a comprehensive documentation. [More info](#)

## 7. ทดสอบโปรแกรมต่อไปนี้ &lt; อย่าลืมแก้ไข Wifi, Pass, Token Key &gt; ดูการทำงานที่ Serial Monitor

```
#include "ThingsBoard.h"
#include <WiFi.h>
#define WIFI_AP           "Test1234"
#define WIFI_PASSWORD     "MaiMeeJingJing"
#define TOKEN             "fm6EwZ1L9tZrSV1f06pM"
#define THINGSBOARD_SERVER "demo.thingsboard.io"

// Baud rate for debug serial
#define SERIAL_DEBUG_BAUD 115200

// Initialize ThingsBoard client
WiFiClient espClient;
// Initialize ThingsBoard instance
ThingsBoard tb(espClient);
// the Wifi radio's status
int status = WL_IDLE_STATUS;

void setup() {
  // initialize serial for debugging
  Serial.begin(SERIAL_DEBUG_BAUD);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();
}

void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }

  if (tb.connected()) {
```

```

// Connect to the ThingsBoard
Serial.print("Connecting to: ");
Serial.print(THINGSBOARD_SERVER);
Serial.print(" with token ");
Serial.println(TOKEN);
if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
  Serial.println("Failed to connect");
  return;
}

Serial.print("Sending data...");

// Uploads new telemetry to ThingsBoard using MQTT.
// See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
// for more details
float xTemp = random(00, 50);
float xHmid = random(51, 99);
Serial.print(xTemp, 2);
Serial.print(",");
Serial.print(xHmid, 2);
Serial.println();

//tb.sendTelemetryInt("temperature", xTemp);
//tb.sendTelemetryInt("humidity", xTemp);
tb.sendTelemetryFloat("temperature", xTemp);
tb.sendTelemetryFloat("humidity", xHmid);

tb.loop();
delay(5000);
}

void InitWiFi()
{
  Serial.println("Connecting to AP ...");
  // attempt to connect to WiFi network

  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  // Loop until we're reconnected
  status = WiFi.status();
  if (status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to AP");
  }
}

```

```

Connecting to AP ...
.....Connected to AP
Connecting to: demo.thingsboard.io with token fm6EwZ1L9tZrSV1f06pM
Sending data...31.00,93.00
Sending data...47.00,69.00

```

☒ Autoscroll ☐ Show timestamp

Both NL & CR

115200 baud

Clear output

8. กลับมาที่ ThingsBoard หน้า Device ให้เลือก Latest Telemetry จะปรากฏข้อมูลที่ส่งเข้ามา

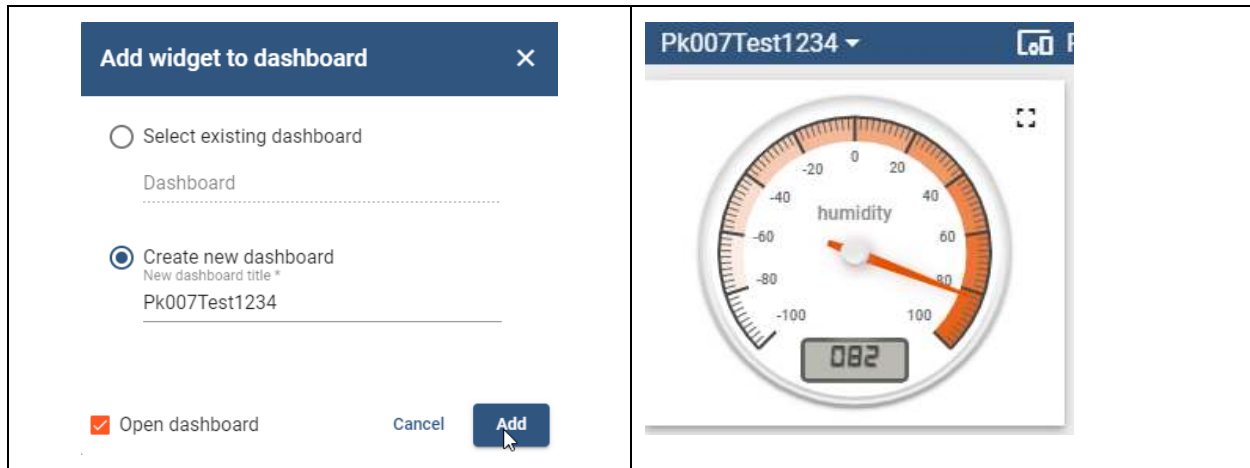
The screenshot shows the 'PkTest1234' device details page. The 'Latest telemetry' tab is selected, displaying a table of recent data points. The table has columns for 'Last update time', 'Key', and 'Value'.

Last update time	Key	Value
2021-05-21 23:53:10	humidity	70
2021-05-21 23:53:09	temperature	35

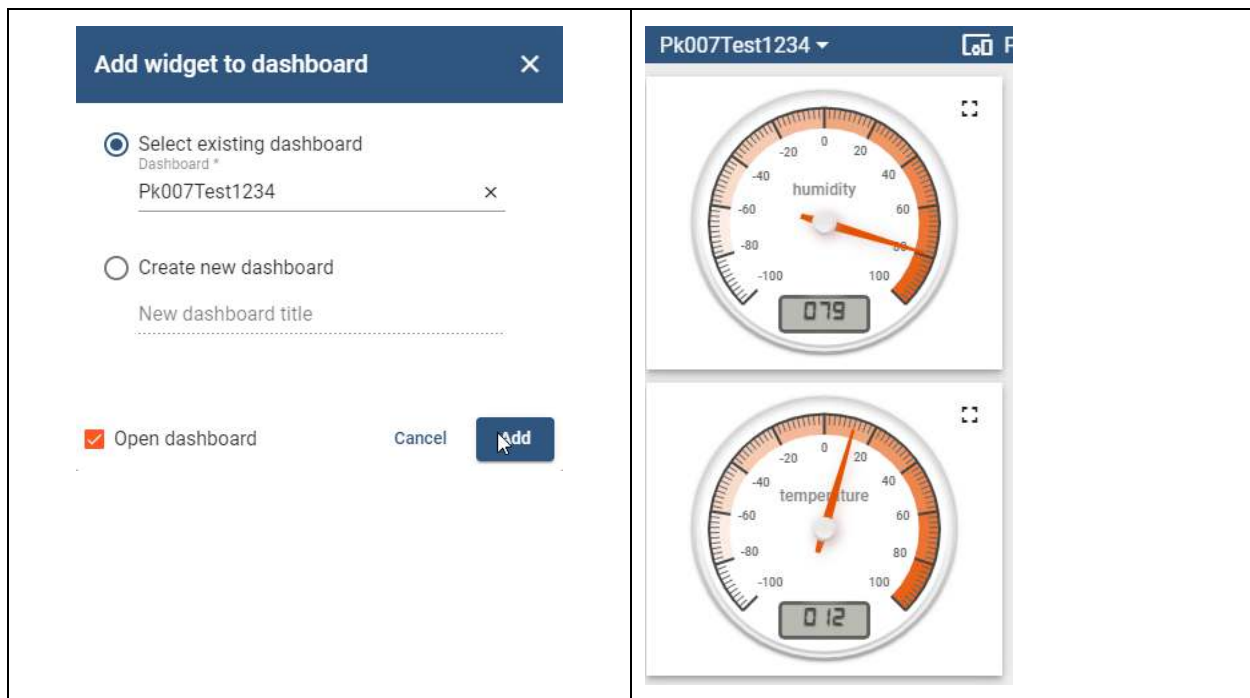
9. เลือกข้อมูล humidity แล้วเลือก Show on Widget → ให้ใช้ Analogue Gauges → Add to dashboard

The screenshot shows the 'Show on widget' button being clicked, which opens a modal for selecting a widget. The 'Analogue gauges' widget is selected, and the 'Add to dashboard' button is clicked. The resulting dashboard shows an analogue gauge for 'humidity' with a value of 75.

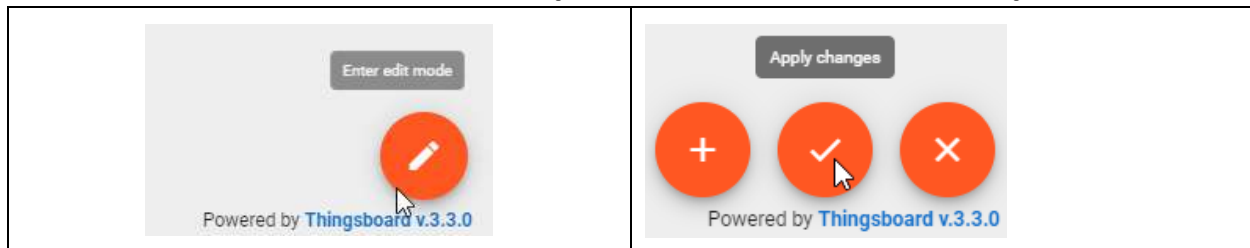
10. กรณี **dashboard** ใหม่ให้สร้างโดยการ Create new dashboard พร้อมทั้งตั้งชื่อและเลือก Open dashboard ตามด้วยการกด ADD



11. ทำซ้ำกับตัวแปร temperature

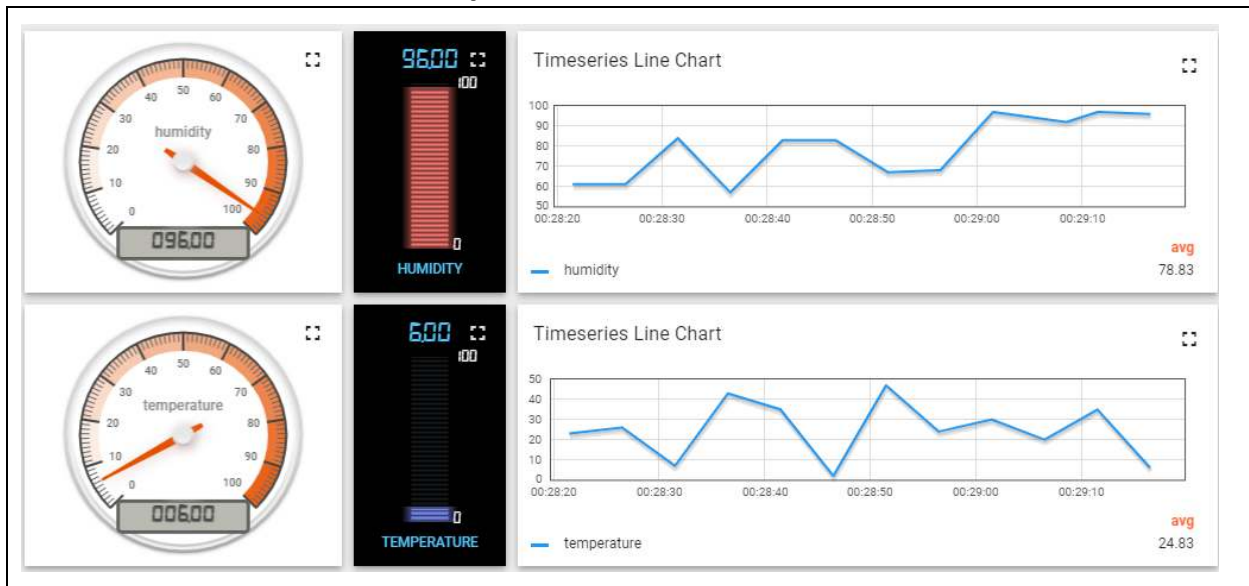


12. เข้าไปแก้ไข dashboard โดยการกดไอคอนรูป ดินสอ ด้านล่างขวามือ, กดเครื่องหมายถูกคือบันทึก





13. ปรับแก้ไข dashboard ให้เป็นตามรูป และมีข้อสังเกต ดังนี้



- ค่า Analog Gauge หน้าปัดต้องมีค่าระหว่าง 0-100
- ค่า Digital Gauge หน้าปัดต้องมีค่าระหว่าง 0-100

14. แชร์ Dashboard ให้ User โดยการแชร์ Device และ แชร์ Dashboard

**Devices** Device profile All

Created time ↓	Name	Device profile	Label	Customer	Public	Is gateway	
2021-05-21 23:23:26	PkTest1234	default		Public	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

Test Navina

---

**Dashboard is now public**

Your dashboard **Pk007Test1234** is now public and accessible via next public link:

<https://demo.thingsboard.io/dashboard/92acbd00-ba56-11eb-a311-cf80d7127bfd?publicId=40fa9860-ba10-11eb-a311-cf80d7127bfd>

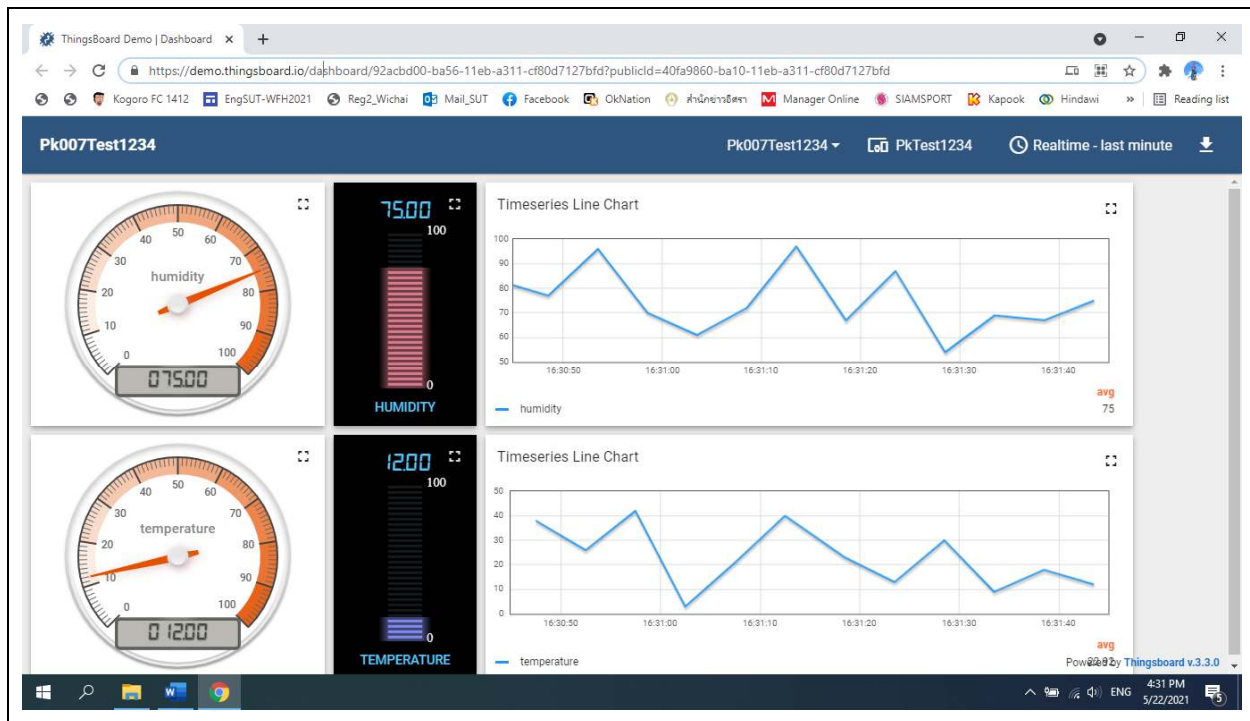
Note: Do not forget to make related devices public in order to access their data.

---

**Dashboards**

Created time ↓	Title	Assigned to customers	Public	
2021-05-22 00:04:15	Pk007Test1234		<input checked="" type="checkbox"/>	

15. ตัวอย่างผมทดสอบ <https://demo.thingsboard.io/dashboard/92acbd00-ba56-11eb-a311-cf80d7127bfd?publicId=40fa9860-ba10-11eb-a311-cf80d7127bfd>



**Mission 1/4:** ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง Dashboard นี้

## 4/6 -- การโปรแกรมเพื่อควบคุมและแสดงค่าระหว่าง ESP32 และ ThingsBoard

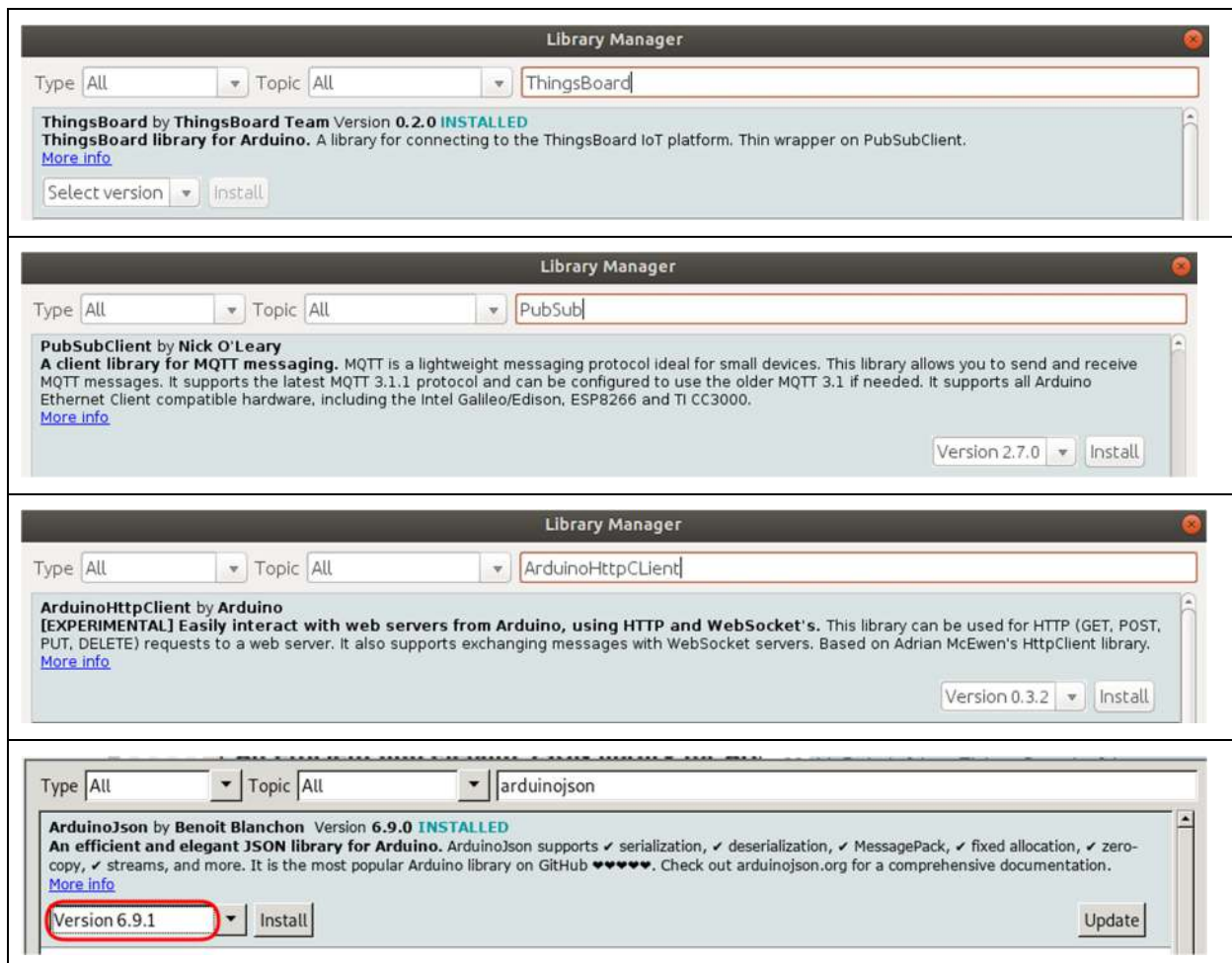
<https://thingsboard.io/docs/samples/esp8266/gpio/>

<https://thingsboard.io/docs/samples/esp32/gpio-control-pico-kit-dht22-sensor/>

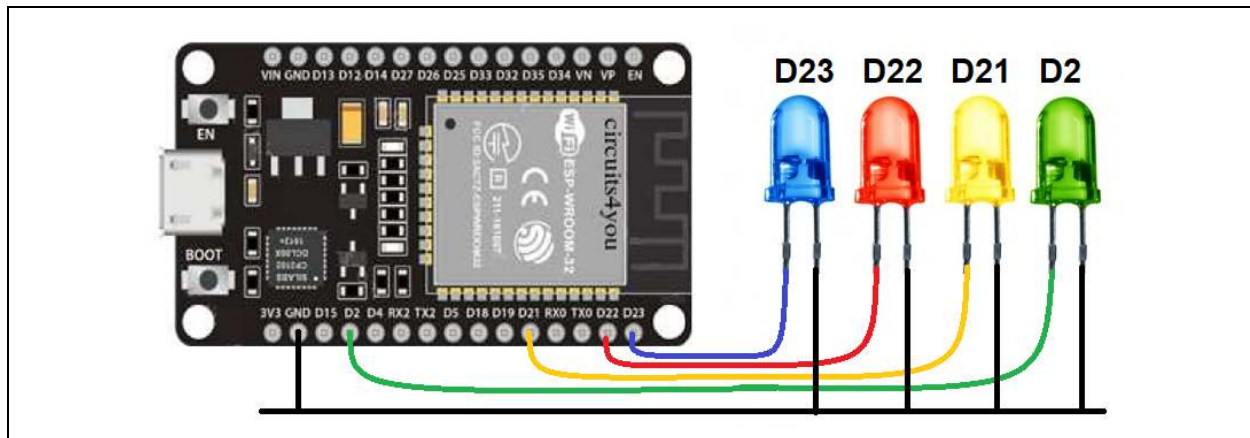
## Lab202 -- ThingsBoard data monitor and control

## 16. ปรับปรุง Arduino Library

- ThingsBoard Arduino SDK Ver 0.2.0
- PubSubClient by Nick O'Leary Ver 2.7.0
- ArduinoHttpClient libraries. Ver 0.3.2
- ArduinoJSON library Ver 6.9.1



## 17. ตัวอย่าง



## 18. โหลดโปรแกรมไปยัง ESP32 โดยโปรแกรมแบ่งออกเป็น 3 ไฟล์ และอย่าลืมแก้ Wifi-Name, Wifi-Pass, Token Key

 _ConnectWifi.h	8/6/2021 2:44 PM
 _ThingBoardRPC.h	8/6/2021 2:44 PM
 Lab202_Test002	8/6/2021 2:49 PM

```
// Part 1 of 3
// Helper macro to calculate array size
#define COUNT_OF(x) ((sizeof(x)/sizeof(0[x])) / ((size_t)(!(sizeof(x) % sizeof(0[x])))))

#include <WiFi.h>
#include <ThingsBoard.h>
#define WIFI_AP_NAME "Test1234"
#define WIFI_PASSWORD "passssssssss"
#define TOKEN "kkkkkkkkkkkkkkkkkkkk"
#define THINGSBOARD_SERVER "demo.thingsboard.io"
#define pinLEDBlink 2

WiFiClient espClient;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;

uint8_t leds_PinControl[] = {21, 22, 23};
int leds_Ststus[] = { 0, 0, 0 };
char StringEcho[] = "stsLED_1";

int loopDelay = 20; // Main loop delay(ms)
int sendDataDelay = 2000; // Period of Sending Tempp/Humid.
int BlinkLEDDelay = 500; // Initial period of LED cycling.
int Count_BlinkLEDDelay = 0; // Time Counter Blink peroid
int Count_sendDataDelay = 0; // Time Counter Sending Tempp/Humid

bool Subscribed_Status = false; // Subscribed_Status for the RPC messages.
int ststus_BlinkLED = 0; // LED number that is currently ON.

#include "_ThingBoardRPC.h"
#include "_ConnectWifi.h"

//=====
void setup() {
  // Initialize serial for debugging
  Serial.begin(115200);
  WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
  WiFi_init();

  // Pinconfig
  pinMode(pinLEDBlink, OUTPUT);
  for (size_t i = 0; i < COUNT_OF(leds_PinControl); ++i) {
    pinMode(leds_PinControl[i], OUTPUT);
  }
}
```

```

//=====
void loop() {
  // Step0/6 - Loop Delay
  delay(loopDelay);
  Count_BlinkLEDDelay += loopDelay;
  Count_sendDataDelay += loopDelay;

  // Step1/6 - Check if next LED Blink
  if (Count_BlinkLEDDelay > BlinkLEDDelay) {
    digitalWrite(pinLEDBlink, ststus_BlinkLED);
    ststus_BlinkLED = 1 - ststus_BlinkLED;
    Count_BlinkLEDDelay = 0;
  }

  // Step 2/6 - Reconnect to WiFi, if needed
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
    return;
  }

  // Step 3/6 - Reconnect to ThingsBoard, if needed
  if (!tb.connected()) {
    Subscribed_Status = false;
    // Connect to the ThingsBoard
    Serial.print("Connecting to: "); Serial.print(THINGSBOARD_SERVER);
    Serial.print(" with token "); Serial.println(TOKEN);
    if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
      Serial.println("Failed to connect");
      return;
    }
  }

  // Step 4/6 - Subscribe for RPC, if needed
  if (!Subscribed_Status) {
    Serial.println("Subscribing for RPC...");
    // Perform a subscription. All consequent data processing will happen in
    // callbacks as denoted by callbacks[] array.
    if (!tb.RPC_Subscribe(callbacks, COUNT_OF(callbacks))) {
      Serial.println("Failed to subscribe for RPC");
      return;
    }
    Serial.println("Subscribe done");
    Subscribed_Status = true;
  }

  // Step 5/6 - Check if it is a time to send Temp/Humid
  if (Count_sendDataDelay > sendDataDelay) {
    Serial.print("Sending data...");
    float temperature = random(20, 30);
    float humidity = random(40, 50);
    tb.sendTelemetryFloat("temperature", temperature);
    tb.sendTelemetryFloat("humidity", humidity);
    Serial.print("T=" + String(temperature, 2) + ", ");
    Serial.print("H=" + String(humidity, 2) + ", ");
    Serial.print("LED=");
    for (size_t i = 0; i < COUNT_OF(leds_PinControl); ++i) {
      StringEcho[7] = 0x30 + i; // Set 0 to "0"
      tb.sendTelemetryInt(StringEcho, leds_Ststus[i]);
      Serial.print(leds_Ststus[i]);
    }
    Serial.println();
    Count_sendDataDelay = 0;
  }

  // Step 6/6 - Process messages
  tb.loop();
}

```

```

// Part 2 of 3
// _ThingBoardRPC.h

//=====
// Processes function for RPC call "setValue"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscript/ for more details
//=====
RPC_Response processDelayChange(const RPC_Data &data)
{ Serial.println("Received the set delay RPC method");
  BlinkLEDDelay = data;
  Serial.print("Set new delay: ");
  Serial.println(BlinkLEDDelay);
  return RPC_Response(NULL, BlinkLEDDelay);
}

//=====
// Processes function for RPC call "getValue"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscript/ for more details
//=====
RPC_Response processGetDelay(const RPC_Data &data) {
  Serial.println("Received the get value method");
  return RPC_Response(NULL, BlinkLEDDelay);
}

//=====
// Processes function for RPC call "setGpioStatus"
// RPC_Data is a JSON variant, that can be queried using operator[]
// See https://arduinojson.org/v5/api/jsonvariant/subscript/ for more details
//=====
RPC_Response processSetGpioState(const RPC_Data &data) {
  Serial.println("Received the set GPIO RPC method");
  int pin = data["pin"];
  bool enabled = data["enabled"];
  if (pin < COUNT_OF(leds_PinControl)) {
    Serial.print("Setting LED "); Serial.print(pin);
    Serial.print(" to state "); Serial.println(leds_Ststus[pin]);
    leds_Ststus[pin] = 1 - leds_Ststus[pin];
    digitalWrite(leds_PinControl[pin], leds_Ststus[pin]);
  }
  return RPC_Response(data["pin"], (bool)data["enabled"]);
}

//=====
// RPC handlers
//=====
RPC_Callback callbacks[] = {
  { "setValue",    processDelayChange },
  { "getValue",    processGetDelay },
  { "setGpioStatus", processSetGpioState },
};

```

```
// Part 3 of 3
// _ConnectWifi.h

//=====
void WiFi_Init() {
  Serial.println("Connecting to AP ..."); // attempt to connect to WiFi network
  WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nConnected to AP");
  Serial.print("Local IP = ");
  Serial.println(WiFi.localIP());
}

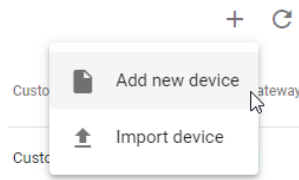
//=====
void reconnect() {
  status = WiFi.status(); // Loop until we're reconnected
  if (status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("\nConnected to AP");
    Serial.print("Local IP = ");
    Serial.println(WiFi.localIP());
  }
}
}
```

LED-2 > ESP32\_DHT22\_GPIO

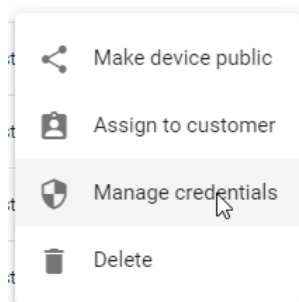
Search ESP32\_DHT22\_GPIO

Name	Date modified	Type	Size
_ConnectWifi.h	5/23/2021 11:41 PM	H File	1 KB
_ThingBoardRPC.h	5/24/2021 12:37 AM	H File	3 KB
ESP32_DHT22_GPIO	5/24/2021 12:36 AM	Arduino file	4 KB

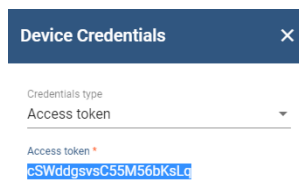
## 18.1 Add New Device



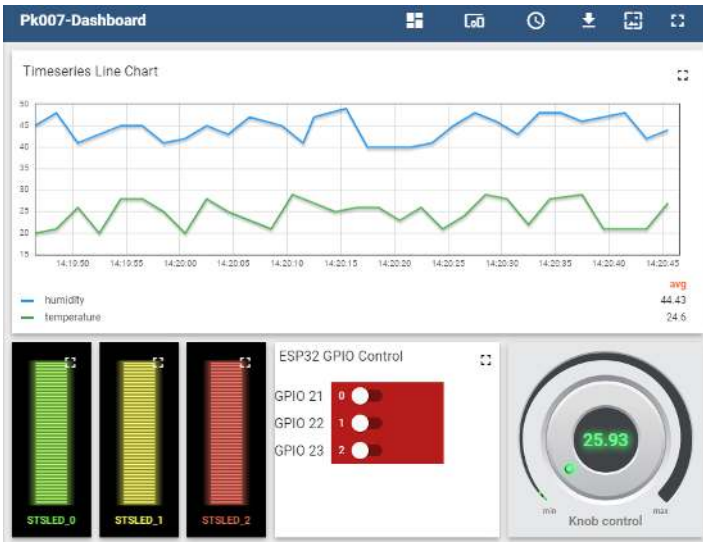
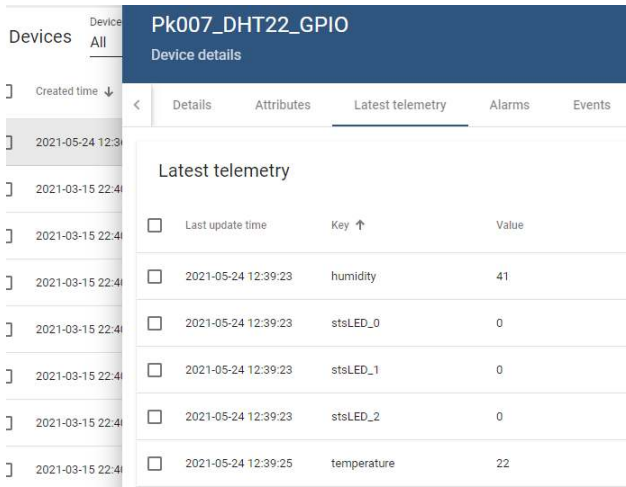
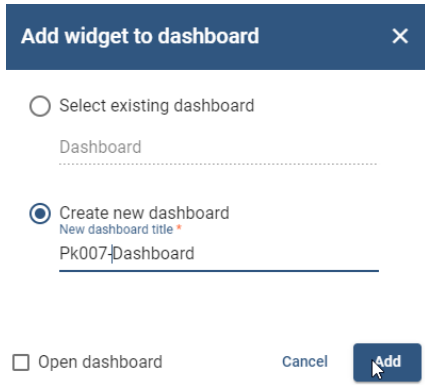

## 18.2 Get Token Key



## 18.3 Copy Token Key

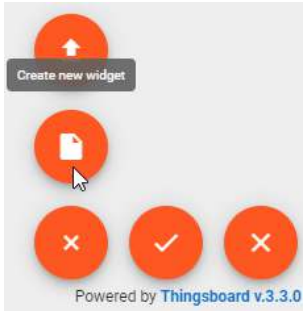


19. ปรับปรุงและตั้งค่า Dashboard ดังรูป ประกอบด้วย Widget 4 ชนิด จำนวน 6 ตัว

																			
<p>ตรวจสอบข้อมูลล่าสุด</p>	 <table border="1"> <thead> <tr> <th>Created time</th> <th>Key</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>2021-05-24 12:39:23</td> <td>humidity</td> <td>41</td> </tr> <tr> <td>2021-05-24 12:39:23</td> <td>stsLED_0</td> <td>0</td> </tr> <tr> <td>2021-05-24 12:39:23</td> <td>stsLED_1</td> <td>0</td> </tr> <tr> <td>2021-05-24 12:39:23</td> <td>stsLED_2</td> <td>0</td> </tr> <tr> <td>2021-05-24 12:39:25</td> <td>temperature</td> <td>22</td> </tr> </tbody> </table>	Created time	Key	Value	2021-05-24 12:39:23	humidity	41	2021-05-24 12:39:23	stsLED_0	0	2021-05-24 12:39:23	stsLED_1	0	2021-05-24 12:39:23	stsLED_2	0	2021-05-24 12:39:25	temperature	22
Created time	Key	Value																	
2021-05-24 12:39:23	humidity	41																	
2021-05-24 12:39:23	stsLED_0	0																	
2021-05-24 12:39:23	stsLED_1	0																	
2021-05-24 12:39:23	stsLED_2	0																	
2021-05-24 12:39:25	temperature	22																	
<p>1/4 Add Widget 1</p> <ul style="list-style-type: none"> <li>เลือกข้อมูล humidity และ temper.</li> <li>ชนิด Chart → timeseries Line Chart</li> <li>Add to dashboard</li> <li>New → Pk007-Dashbooard</li> </ul>																			
<ul style="list-style-type: none"> <li>กดปากกาเพื่อปรับตำแหน่งและขนาด</li> </ul>																			

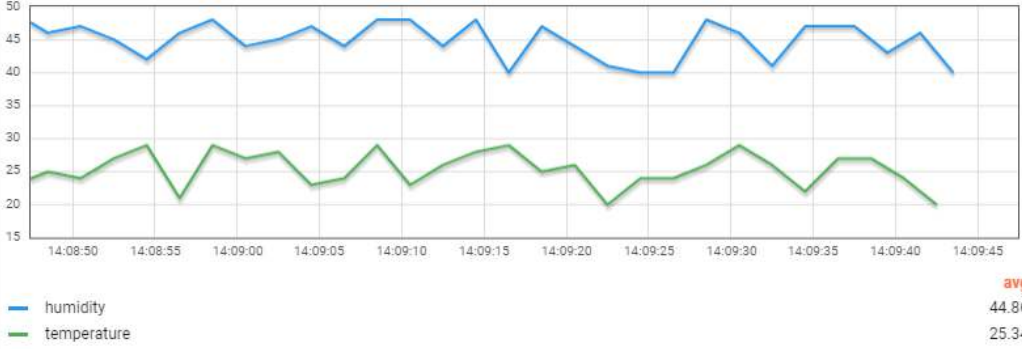


<p>2/4 Add Widget 2,3,4</p> <ul style="list-style-type: none"> <li>เลือกข้อมูล STSLED_0</li> <li>ชนิด Digital Gauge → digital vertical bar</li> <li>Add to dashboard</li> <li>Select → Pk007-Dashboard</li> </ul>	
<ul style="list-style-type: none"> <li>แก้ไขค่า Min,Max = 0, 1</li> <li>แก้ไขสี Default Color เป็น โทนมเขียว</li> </ul>	<div data-bbox="625 535 836 913"> <p><b>Digital vertical bar</b></p> <p>Digital vertical bar</p> <p>Data      Set</p> <p>Minimum value 0</p> <p>Maximum value 1</p> <p>Gauge type Vertical bar</p> </div> <div data-bbox="1112 556 1323 766"> <p>Default color</p> <p>Background color of t</p> <p>rgb(32, 205, 72)</p> <p><input type="checkbox"/> Use precise value</p> </div>
<ul style="list-style-type: none"> <li>Copy สำหรับ 3 LED ปรับ โทนมสีเป็น Grn, Ylw, Red</li> <li>Ylw-Data เป็น stsLED_1 และ Red-Data เป็น stsLED_2</li> </ul>	
	<div data-bbox="625 1176 1177 1344"> <p>Key *</p> <p>stsLED_1</p> <p>Label *</p> <p>stsLED_1</p> <p>Color * #f3f021</p> </div>
	

<p>3/4 Add Widget 5</p> <ul style="list-style-type: none"> <li>• ที่ Widget</li> <li>• เปิด Pk007-Dashboard</li> <li>• กดดินสอ</li> <li>• เลือก Create New Widget</li> <li>• เลือก Control Widget</li> <li>• เลือก Knob Control</li> </ul>		<p><b>New Knob Control</b></p> <p>Knob Control</p> <p>Data      Sett</p> <hr/> <p>Minimum value *</p> <p>0</p> <hr/> <p>Maximum value *</p> <p>1000</p> <hr/> <p>Initial value</p> <p>250</p> <hr/> <p>Knob title</p> <p>Knob control</p> <hr/> <p>Get value method *</p> <p>getValue</p> <hr/> <p>Set value method *</p> <p>setValue</p> <hr/> <p>RPC request timeout *</p> <p>500</p> <hr/>
<ul style="list-style-type: none"> <li>• กำหนด Data เป็น Device ที่สร้างขึ้นมา</li> <li>• กำหนดค่าต่างๆ ดังรูปขวาสุด</li> <li>• ปรับ Knob สัญเหตุการณ์ทำงานของ LED D2</li> </ul>	<p><b>Add Widget: Knob Control</b></p> <p>Data      Settings      Advanced</p> <hr/> <p>Target device</p> <p>Pk007_DHT22_GPIO</p> <hr/>	

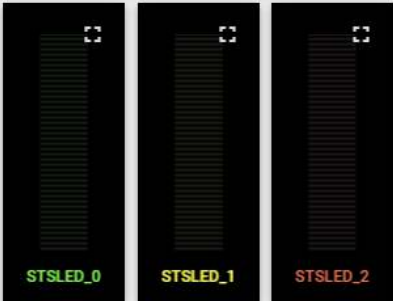
  

**Timeseries Line Chart**




avg  
44.86  
25.34

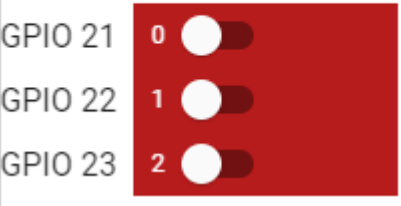
  




STSLED\_0      STSLED\_1      STSLED\_2



Knob control

<p>4/4 Add Widget 6</p> <ul style="list-style-type: none"> <li>• ที่ Widget</li> <li>• เปิด Pk007-Dashboard</li> <li>• กดติ๊กสอ</li> <li>• เลือก Create New Widget</li> <li>• เลือก GPIO Widget</li> <li>• เลือก GPIO Control</li> </ul>	<p>ESP32 GPIO Control</p> 	<p>Advance Setting</p> <p>Pin 0, GPIO 21,R 0, C 0</p> <p>Pin 1, GPIO 22,R 1, C 0</p> <p>Pin 2, GPIO 23,R 2, C 0</p>
--	--	---

The dashboard displays a Timeseries Line Chart with two data series: humidity (blue line) and temperature (green line). The x-axis represents time from 14:19:50 to 14:20:45. The y-axis represents values from 15 to 50. The average values are 44.43 for humidity and 24.6 for temperature.

The bottom section contains three LED status indicators: STSLED\_0 (green), STSLED\_1 (yellow), and STSLED\_2 (red). It also includes the ESP32 GPIO Control widget and a Knob control widget showing a value of 25.93.

20. สามารถ Export Dashboard เพื่อเก็บไว้ใช้งานใน Server อื่นๆ ได้



The screenshot shows the dashboard's toolbar with various icons. The 'Export dashboard' button is highlighted, indicating the option to export the dashboard for use on other servers.

**Mission 2/4:** ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง ThingsBoard พร้อมทั้งควบคุม On/Off - 4 LED และ Blink Speed สำหรับอีก 1 LED

## Lab203 – ThingsBoard Monitor and Control with MQTT Protocol

<https://thingsboard.io/docs/samples/esp8266/gpio/>

<https://blog.thingsboard.io/2017/01/esp8266-gpio-control-over-mqtt-using.html>

<https://thingsboard.io/docs/reference/mqtt-api/>

<https://thingsboard.io/docs/user-guide/integrations/mqtt/>

## 21. ปรับปรุง Arduino Library

- PubSubClient by Nick O'Leary. Ver 2.8.0
- ArduinoJSON library by Benoit Blanchon Ver 5.8.0

## 22. โหลดโปรแกรมไปยัง ESP32 (โปรแกรมแยกเป็น 3 ไฟล์) อย่าลืมใส่ข้อมูลเฉพาะบุคคล และต่อวงจรเพื่อใช้งาน

```
// File 1 of 3
// https://thingsboard.io/docs/samples/esp8266/gpio/
// https://blog.thingsboard.io/2017/01/esp8266-gpio-control-over-mqtt-using.html

#include <WiFi.h>
#include <ArduinoJson.h> // by Benoit Blanchon >> Ver 5.8.0
#include <PubSubClient.h> // by Nick O'Leary. >> Ver 2.8.0
// replace #ifdef ESP8266
// to #if defined (ESP8266) || defined(ESP32)

#define WIFI_AP_NAME "Test1234"
#define WIFI_PASSWORD "0816601929"
#define Device_Name "PkTest300"
#define Device_Token "cSWddgsvsC55M56bKsLq"
#define thingsboardServer "demo.thingsboard.io"

#define GPIO1_ESP32Pin 22
#define GPIO2_ESP32Pin 23
boolean gpioState[] = {false, false};
int status = WL_IDLE_STATUS;

WiFiClient wifiClient;
PubSubClient client(wifiClient);

#include "_HandOnMQTT.h"
#include "_WifiConnect.h"

void setup() {
  Serial.begin(115200);
  // Set output mode for all GPIO pins
  pinMode(GPIO1_ESP32Pin, OUTPUT);
  pinMode(GPIO2_ESP32Pin, OUTPUT);
  delay(10);
  InitialWifi();
  client.setServer( thingsboardServer, 1883 );
  client.setCallback(on_message);
}

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

```

// File 2 of 3
// _HandOnMQTT.h
//=====
//=====
String get_gpio_status() {
    // Prepare gpio's JSON payload string
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject & data = jsonBuffer.createObject();
    data[String(GPIO1_ESP32Pin)] = gpioState[0];
    data[String(GPIO2_ESP32Pin)] = gpioState[1];
    char payload[256];
    data.printTo(payload, sizeof(payload));
    String strPayload = String(payload);
    Serial.print("Get GPIO Status: ");
    Serial.println(strPayload);
    return strPayload;
}

//=====
//=====
void set_gpio_status(int pin, boolean enabled) {
    if (pin == GPIO1_ESP32Pin) {
        gpioState[0] = 1 - gpioState[0];
        digitalWrite(GPIO1_ESP32Pin, gpioState[0]);
    }
    if (pin == GPIO2_ESP32Pin) {
        gpioState[1] = 1 - gpioState[1];
        digitalWrite(GPIO2_ESP32Pin, gpioState[1]);
    }
}

//=====
//=====
// The callback for when a PUBLISH message is received from the server.
void on_message(const char* topic, byte* payload, unsigned int length) {
    Serial.println("\nOn message");
    char json[length + 1];
    strncpy(json, (char*)payload, length);
    json[length] = '\0';
    Serial.print("Topic: "); Serial.println(topic);
    Serial.print("Message: "); Serial.println(json);

    // Decode JSON request
    StaticJsonBuffer<200> jsonBuffer;
    JsonObject & data = jsonBuffer.parseObject((char*)json);

    if (!data.success()) {
        Serial.println("parseObject() failed");
        return;
    }

    // Check request method
    String methodName = String((const char*)data["method"]);


    // If Reply with GPIO status
    if (methodName.equals("getGpioStatus")) {
        String responseTopic = String(topic);
        responseTopic.replace("request", "response");
        client.publish(responseTopic.c_str(), get_gpio_status().c_str());
    }

    // If Update GPIO status and reply
    if (methodName.equals("setGpioStatus")) {
        set_gpio_status(data["params"]["pin"], data["params"]["enabled"]);
        String responseTopic = String(topic);
        responseTopic.replace("request", "response");
        client.publish(responseTopic.c_str(), get_gpio_status().c_str());
        client.publish("v1/devices/me/attributes", get_gpio_status().c_str());
    }
}


```

```
// File 3 of 3
// _WifiConnect.h
//=====
//=====
void InitialWifi() {
  Serial.println("Connecting to AP ...");
  WiFi.begin(WIFI_AP_NAME, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}


//=====
//=====
void reconnect() {
  // Loop until we're reconnected
  while (!client.connected()) {
    status = WiFi.status();
    if (status != WL_CONNECTED) {
      InitialWifi();
    }
    Serial.print("Connecting to ThingsBoard node ...");
    // Attempt to connect (clientId, username, password)
    if (client.connect(Device_Name, Device_Token, NULL)) {
      Serial.println(" [DONE]");
      // Subscribing to receive RPC requests
      client.subscribe("v1/devices/me/rpc/request/+");
      // Sending current GPIO status
      Serial.println("Sending current GPIO status ...");
      client.publish("v1/devices/me/attributes", get_gpio_status().c_str());
    } else {
      Serial.print(" [FAILED] [ rc = ");
      Serial.print(client.state());
      Serial.println(" : retrying in 5 seconds]");
      delay(5000); // Wait 5 seconds before retrying
    }
  }
}
}
```

 \_HandOnMQTT.h

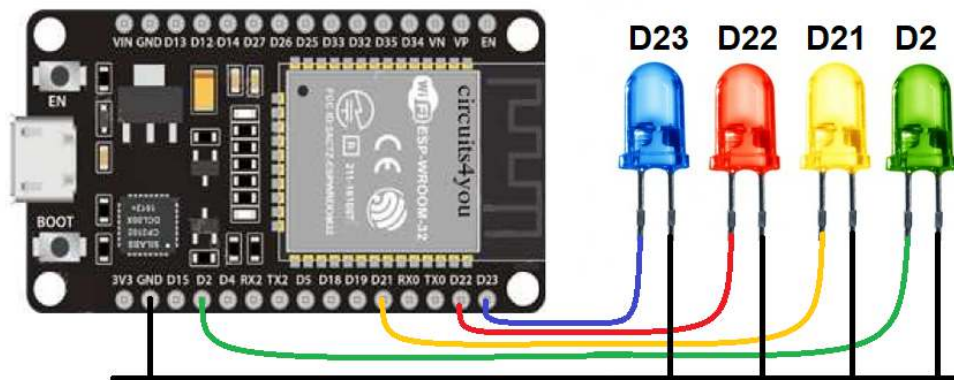
5/24/2021 11:17 PM

 \_WifiConnect.h

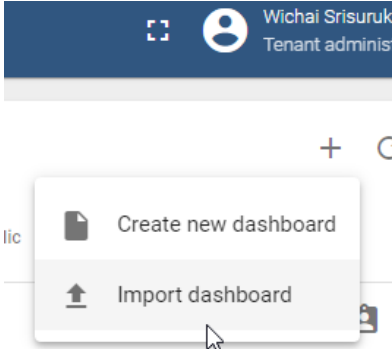
5/24/2021 10:58 PM

 ESP32\_Mission3\_GPIO




5/24/2021 11:17 PM



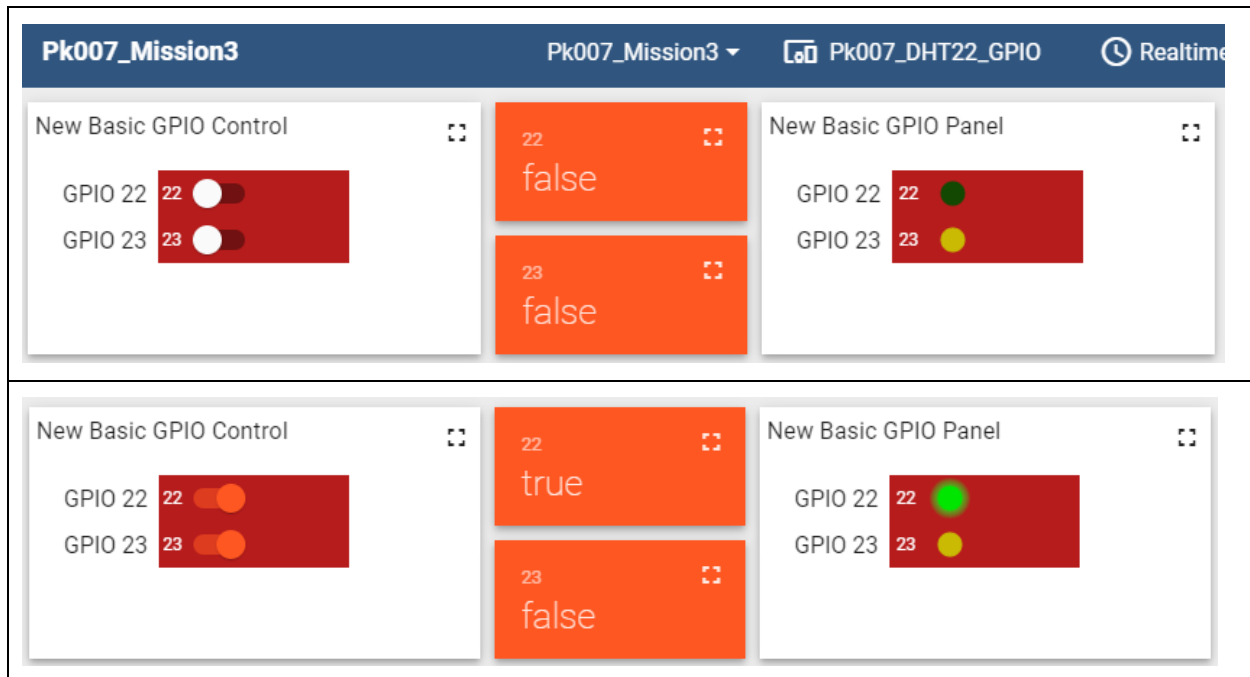
23. สร้าง Device ใหม่ให้มีชื่อเป็น “PkTest300” แล้ว Copy Token Key เพื่อใส่ใน Arduino Code  
 24. ตั้งค่า Dashboard ให้เป็นดังรูป ด้วยวิธีการ Import Dashboard

	<p>Dashboard</p> <p>Add → Import dashboard                  เลือกไฟล์ “pk007_mission3.json”</p>
---	---

25. Add Alias เพื่อตั้งค่าตัวแปรใน Device มาแสดง ด้วยการเข้า Edit Mode และ Setting

	<p>Edit Mode Setting</p>
	<p>PkTest300Alias                  Single entity                  Device                      PkTest300</p>
	<p>เปลี่ยน parameter ทั้ง 4                  Device                  Entity → Pk300Alias</p>

## 26. ผลการทำงาน



## 27. ทดสอบการทำงาน

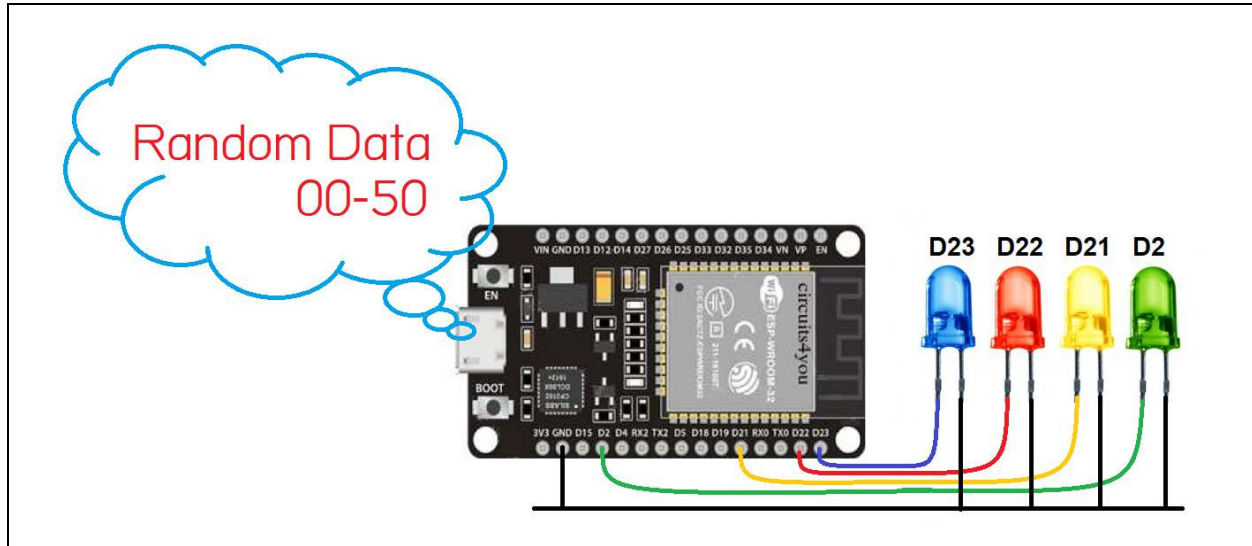
```
Connecting to AP ...
.....Connected to AP
Connecting to ThingsBoard node ...[DONE]
Sending current GPIO status ...
Get GPIO Status: {"22":false,"23":false}

On message
Topic: vl/devices/me/rpc/request/159
Message: {"method":"setGpioStatus","params":{"pin":22,"enabled":false}}
Get GPIO Status: {"22":true,"23":false}
Get GPIO Status: {"22":true,"23":false}
```



**Mission 3/4:** ให้ใช้ MQTT กับ ThingsBoard

1. ปรับปรุงเพื่อให้ทำงานควบคุมการ On/Off - 4 LED
2. เพิ่มเติม คือ ทดสอบส่งข้อมูล 1 ค่าแบบสุ่มระหว่าง 00 – 50 ไปแสดงที่ Dashboard ด้วย ได้หรือไม่ ทำอย่างไรบ้างให้อธิบาย {Read <https://thingsboard.io/docs/user-guide/device-profiles/> }



## 5/6 -- โครงการ “ตรวจวัดอุณหภูมิความชื้นของฟาร์มไก่ ให้มีการควบคุมอุปกรณ์และมีการแจ้งเตือน”

<https://www.smethailandclub.com/startups-6416-id.html>

<https://thingsboard.io/docs/getting-started-guides/helloworld/>

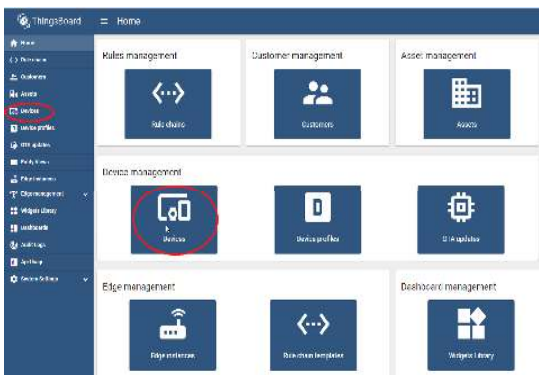
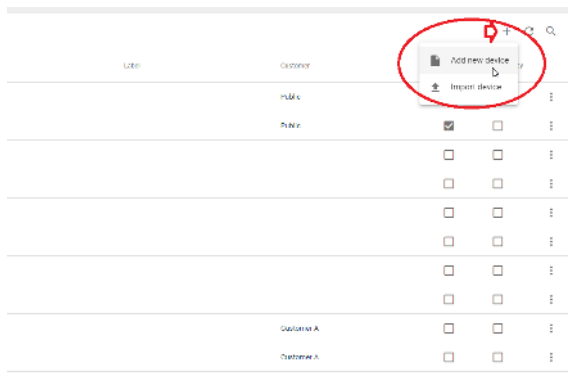
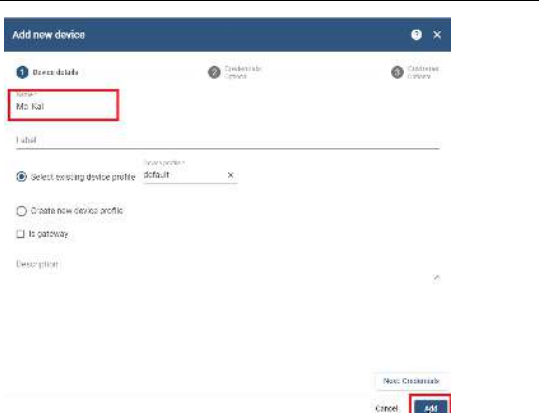
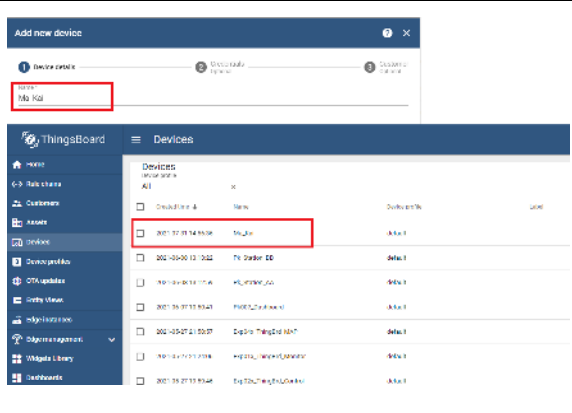
หัวข้อนี้จะเป็นการใช้งานเพื่อทดสอบการส่งข้อมูลจาก ESP-32 Device ไปยัง ThingsBoard.IO กำหนดให้เป็นโครงการเรื่อง “ตรวจวัดอุณหภูมิความชื้นของฟาร์มไก่ ให้มีการควบคุมอุปกรณ์และมีการแจ้งเตือน” โดยจะมีขั้นตอนการทำงาน ดังนี้

1. สร้าง Device
2. จัดการ Device ผ่าน Token
3. สร้าง Dashboard และส่งข้อมูลไปยัง Device
4. กำหนดเงื่อนไขการแจ้งเตือน
5. สร้างการแจ้งเตือน
6. กำหนดช่องทางการแจ้งเตือน
7. สร้างและแชร์ Dashboard เพื่อแสดงค่าที่ส่งจาก ESP32 Board

### Lab204 – Smart Poultry House via ThingsBoard

#### Step 1. Provision Device

1. Login <https://demo.thingsboard.io/login>
2. สร้าง Device Name = My\_Kai

Click Device	Add New Device
	
Fill Name, Add	New Device Name = My_Kai
	

## Step 2. Connect device

### 3. Create Token

The screenshot shows the ThingsBoard IoT Platform interface. On the left, a list of devices is displayed, with 'Ma\_Kai' highlighted. The main panel shows the 'Ma\_Kai' device details. The 'Details' tab is selected, and the 'Copy access token' button is highlighted with a red box. A green notification bar at the bottom right states 'Device access token has been copied to clipboard'.

### 4. Test with curl Online Command

```
curl -v -X POST -d '{"temperature": 25}' $HOST_NAME/api/v1/$ACCESS_TOKEN/telemetry --header "Content-Type:application/json"
```

```
curl -v -X POST -d '{"temperature": 25}' https://demo.thingsboard.io/api/v1/$ACCESS_TOKEN/telemetry --header "Content-Type:application/json"
```

### 5. Goto >> <https://reqbin.com/curl>

```
curl -v -X POST -d '{"temperature": 25}' https://demo.thingsboard.io/api/v1/EMCttqEvfFLvgTRfQLB1/telemetry --header "Content-Type:application/json"
```

The screenshot shows the reqbin.com/curl website. The 'Run Curl Commands Online' interface is displayed. The curl command is entered in the input field, and the 'Run' button is highlighted.

## 6. Result

The screenshot displays the ThingsBoard IoT Platform interface. On the left, a table lists various devices. The device 'My\_Kai' is highlighted with a red box. On the right, the 'My\_Kai' device details are shown, with the 'Latest telemetry' tab selected and highlighted with a red box. A specific telemetry entry for 'temperature' with a value of '29' is highlighted with a blue box.

Created time	Name	Device profile
2021-07-31 15:08:42	My_Kai	default
2021-09-28 18:19:22	PK_Station_B#	default
2021-06-28 13:12:59	PK_Station_AA	default
2021-06-27 10:50:41	PK067_Boardboard	default
2021-05-27 21:50:57	Exp004_ThingRnc_MAP	default
2021-05-27 21:24:06	Exp01x_ThingRnc_Monitor	default
2021-05-27 19:59:46	Exp02x_ThingRnc_Control	default
2021-05-25 14:48:37	PK007Exp03_MQTT	default
2021-05-25 13:28:06	PK007Exp01_Monitor	default
2021-03-15 22:40:58	Test Device 63	default

**My\_Kai**  
Device details

Details | Attributes | **Latest telemetry** | Alarms | Events | Relations | Audit Logs

Latest telemetry

Last update time	Key	Value
2021-07-31 15:17:28	temperature	29

## 7. ESP32 Coding – Random temperature generate

### Check Arduino Library

- **ThingsBoard Arduino SDK** Ver 0.2.0
- **PubSubClient by Nick O'Leary** Ver 2.7.0
- **ArduinoHttpClient libraries.** Ver 0.3.2
- **ArduinoJSON library** Ver 6.9.1

```
#include "ThingsBoard.h"
#include <WiFi.h>

#define WIFI_AP           "Test1234"
#define WIFI_PASSWORD     "0816601929"
#define TOKEN             "EMCttqEvfFLvgTRfQLB1"
#define THINGSBOARD_SERVER "demo.thingsboard.io"
```

```
WiFiClient espClient;
ThingsBoard tb(espClient);
int status = WL_IDLE_STATUS;
```

```
void setup() {
  Serial.begin(115200);
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  InitWiFi();
}
```

```
void loop() {
  delay(1000);
  if (WiFi.status() != WL_CONNECTED) {
    reconnect();
  }
}
```

```
if (!tb.connected()) {
  // Connect to the ThingsBoard
  Serial.print("Connecting to: ");
  Serial.print(THINGSBOARD_SERVER);
  Serial.print(" with token ");
  Serial.println(TOKEN);
  if (!tb.connect(THINGSBOARD_SERVER, TOKEN)) {
    Serial.println("Failed to connect");
    return;
  }
}
```

```

}

Serial.println("\nSending data...");
// Uploads new telemetry to ThingsBoard using MQTT.
// See https://thingsboard.io/docs/reference/mqtt-api/#telemetry-upload-api
// for more details
float Tempp = random(2000, 6000) / 100.0;
Serial.println("Tempp = " + String(Tempp, 2) + "°C");
tb.sendTelemetryInt("temperature", Tempp);
//tb.sendTelemetryFloat("humidity", 42.5);
tb.loop();
}

void InitWiFi() {
  Serial.println("Connecting to AP ...");
  WiFi.begin(WIFI_AP, WIFI_PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected to AP");
}

void reconnect() {
  status = WiFi.status();
  if (status != WL_CONNECTED) {
    WiFi.begin(WIFI_AP, WIFI_PASSWORD);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    Serial.println("Connected to AP");
  }
}

```

COM3

.....Connected to AP  
Connecting to: demo.thingsboard.io with token EMCttqEvfFLvgTRfQLB1

Sending data...  
Tempp = 54.07°C

Sending data...  
Tempp = 51.67°C

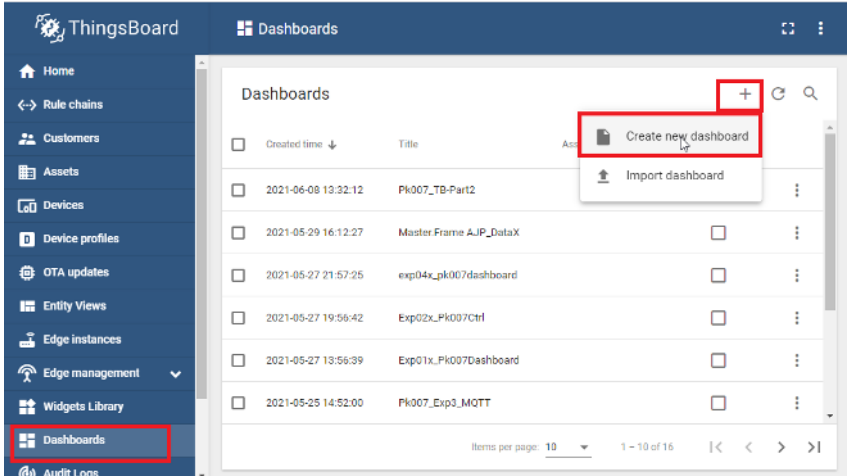
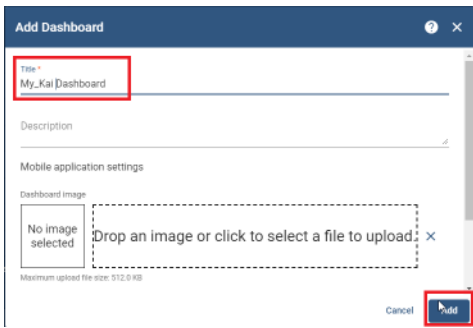
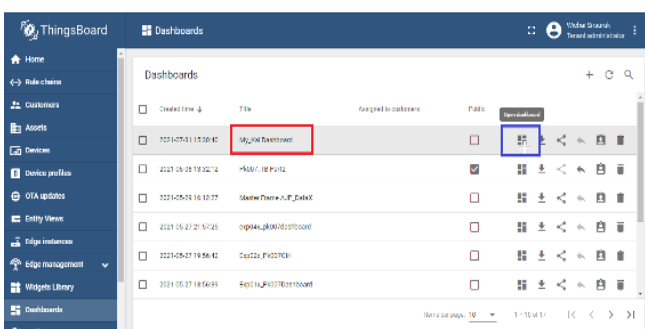
Sending data...  
Tempp = 40.54°C

Sending data...  
Tempp = 20.80°C

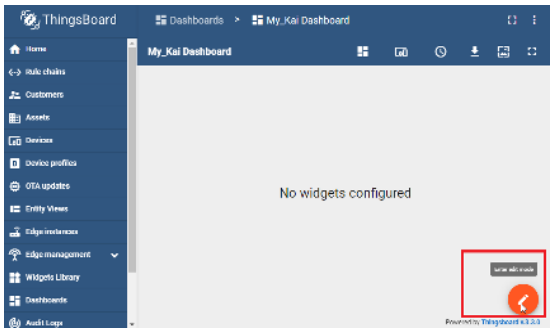
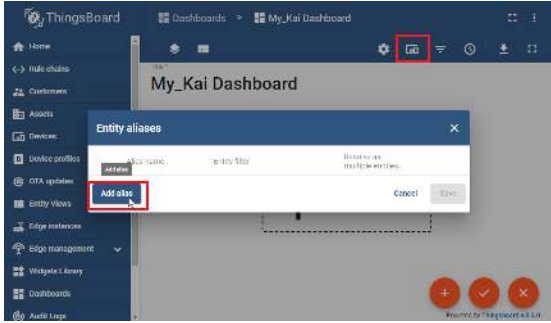

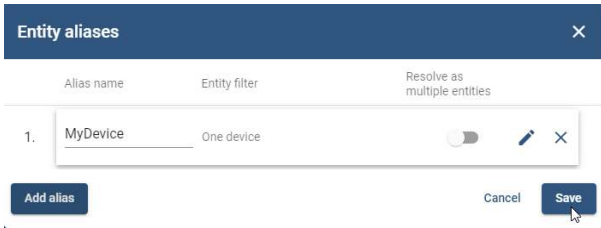
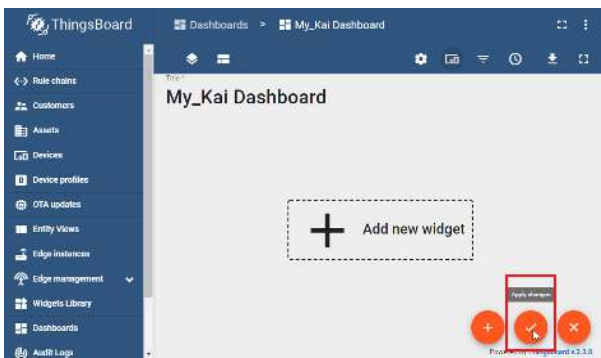
☐ Autoscroll ☐ Show timestamp Carriage return 115200 baud Clear output

## Step 3. Create Dashboard



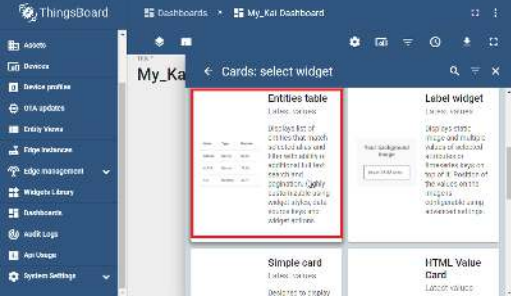
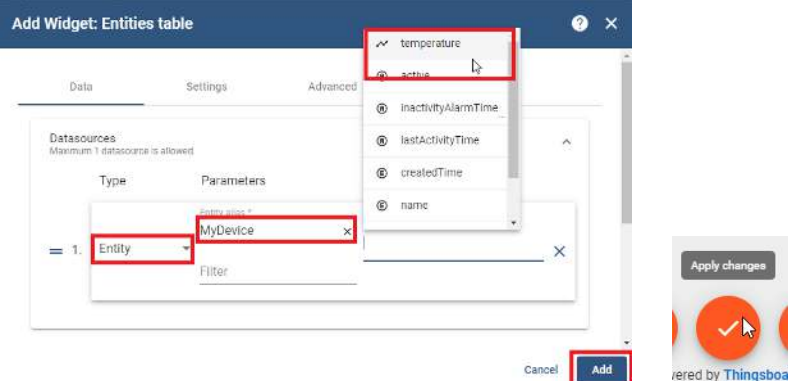
## Step 3.1 Create Empty Dashboard

	Create new dashboard
	Name = My_Kai Dashboard
	Goto Dashboard

Step 3.2 Add Entity Alias

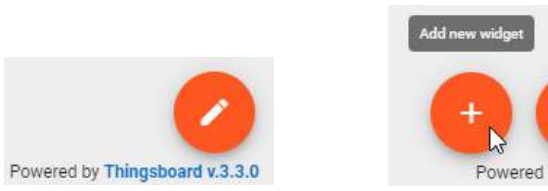
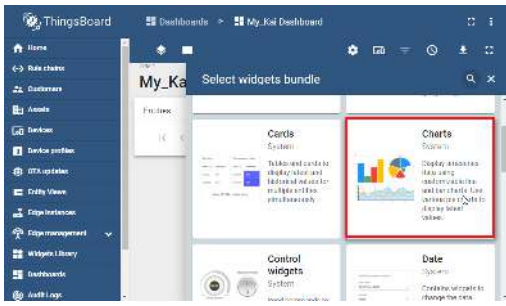
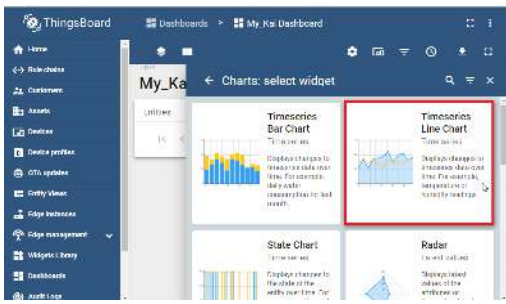
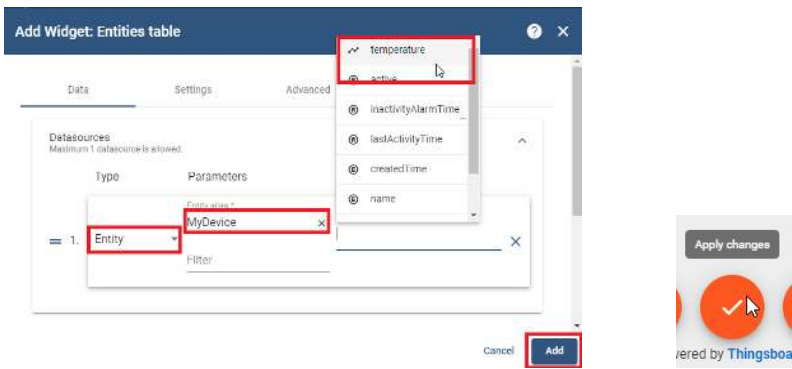
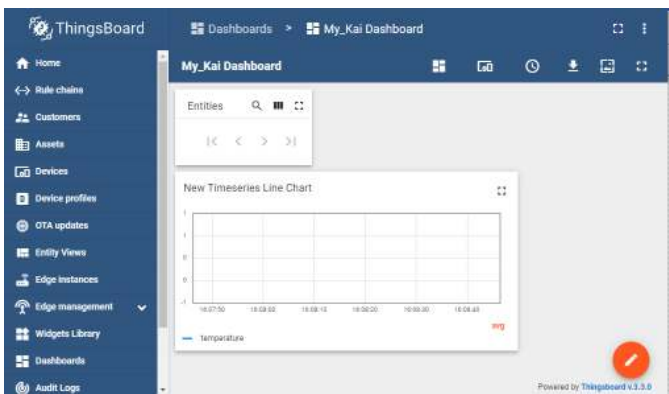
	Edit
	Add Alias
	MyDevice Single entity Device My_Kai
	Save
	Apply

## Step 3.3 Add Table Widget

 <p>Powered by Thingsboard v.3.3.0</p>	Edit Add New Widget
	Cards
	Entities Table
	+Add  Entity → MyDevice → temperature  Add  Apply



## Step 3.4 Add Chart Widget

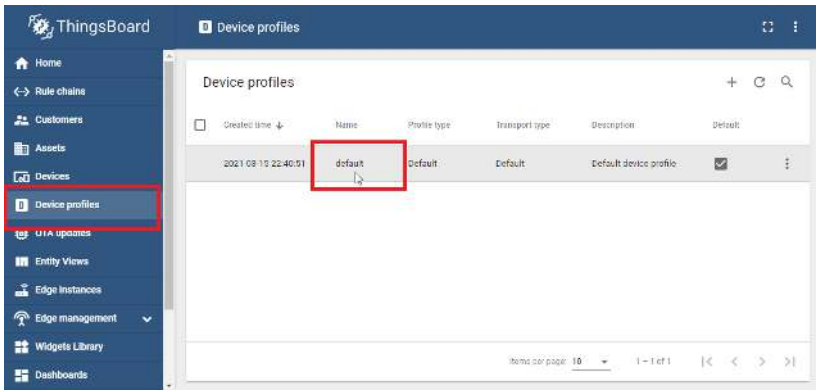
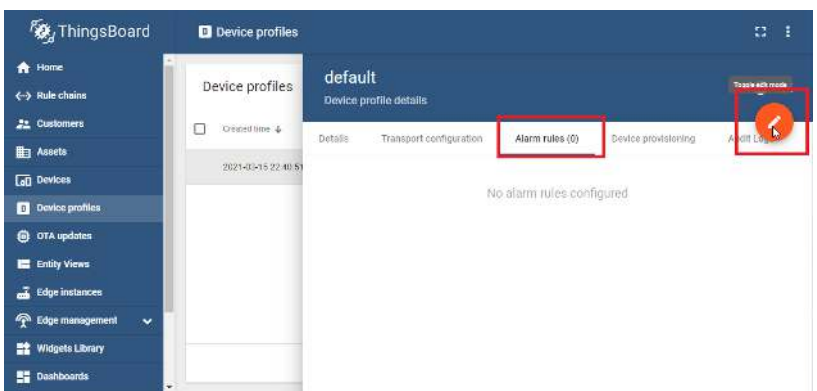

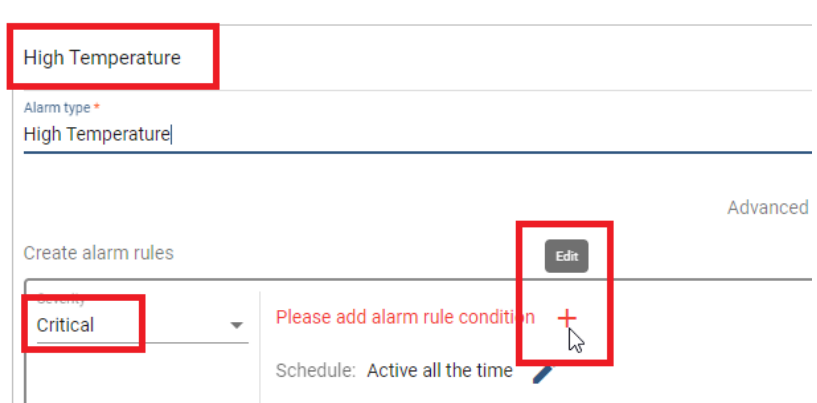
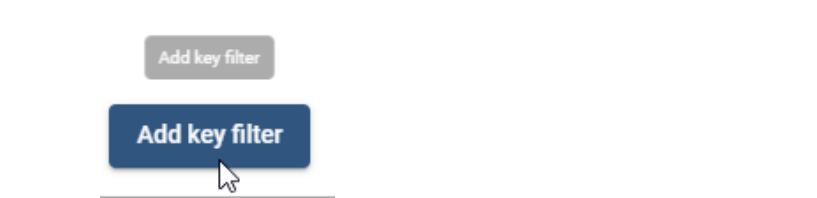
	Edit Add new widget Create new widget
	Charts
	Timeseries Line Chart
	+Add  Entity → MyDevice → temperature  Add  Apply
	

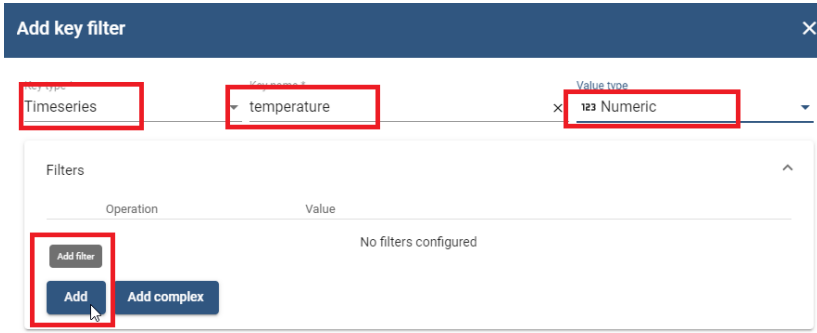
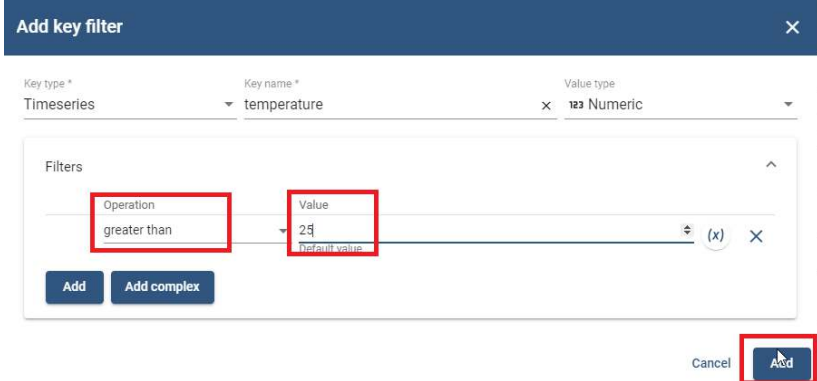

	<p>Realtime – last hour</p> <p>1-day</p> <p>None</p> <p>Max = 150</p> <p>Time(UTC+07:00)</p> <p>Update</p>
--	--

### Step 3.5 Add Alarm Widget

	<p>Add Alarm widget</p> <p>→ Alarm table</p>
	<p>Entity</p> <p>Device → MyDevice</p> <p>...</p> <p>Add</p>
	<p>Apply</p>

## Step 4. Configure Alarm Rules

	<p>Device profiles</p> <p>Default</p>
	<p>Alarm rules</p> <p>Edit</p>
	<p>Add alarm rule</p>
	<p>High Temperature</p> <p>Critical</p> <p>Edit</p>
	<p>Add key filter</p>

	<p>Timeseries</p> <p>Temperature</p> <p>Numeric</p> <p>Add</p>
	<p>Grater than</p> <p>25</p> <p>Add</p>
	<p>Save</p> <p>Apply</p>

### Step 5. Create Alarm

	
--	--

## Step 6. Alarm notifications

It is quite easy to configure email or sms notifications for alarms. We recommend reviewing alarm rule [examples](#) and documentation about [alarm notifications](#).

**Note:** At the moment ThingsBoard supports AWS SNS and Twilio to send SMS. Both services are non-free and require you to create an account. However, you may integrate with other SMS/EMAIL gateways using [REST API call](#) node.

## Step 7. Assign Device and Dashboard to Customer

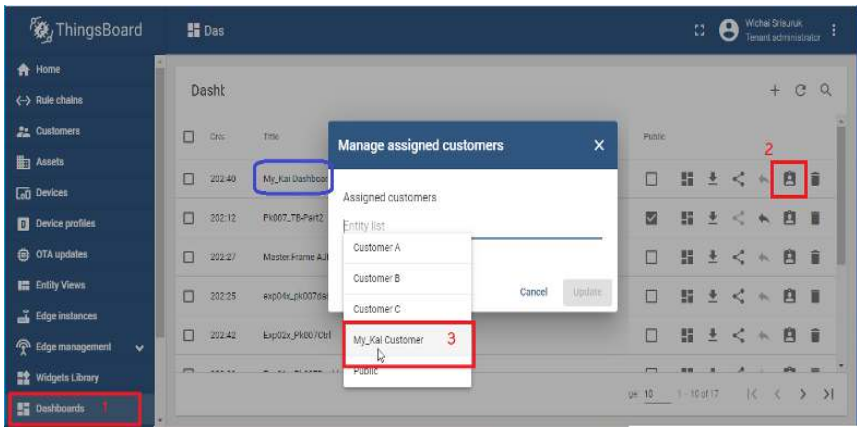
### Step 7.1 Create customer

	<ol style="list-style-type: none"> <li>1. Customers</li> <li>2. (+) Add Customers</li> <li>3. Name = MyKai Customer</li> <li>4. Add</li> </ol>
--	--

### Step 7.2 Assign device to Customer

	<ol style="list-style-type: none"> <li>1. Device Tab</li> <li>2. On row MyKai → Assign Device(s) to Customer</li> <li>3. Select MyKai Customer</li> </ol>
<p>Cancel Assign</p>	<ol style="list-style-type: none"> <li>4. Assign</li> </ol>

## Step 7.3 Assign dashboard to Customer



1. Dashboard Tab

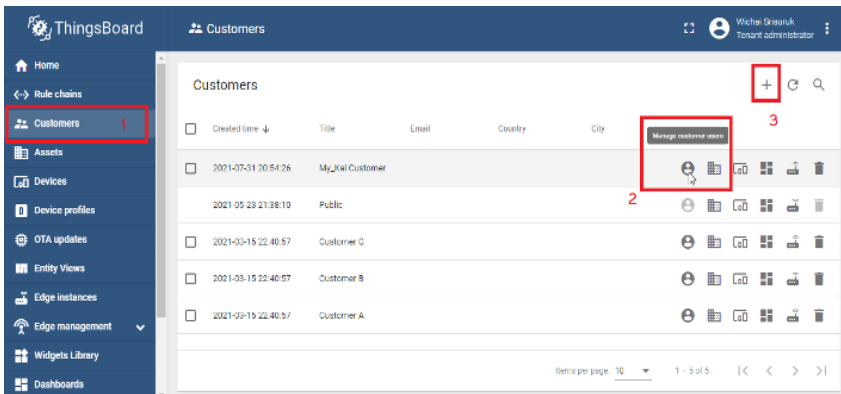
2. On row MyKai Dashboard → Assign Device(s) to Customer

3. Select MyKai Customer

4. Update

- Dashboard Tab
- On row MyKai Dashboard → Assign Device(s) to Customer
- Select MyKai Customer

## Step 7.4 Create customer user

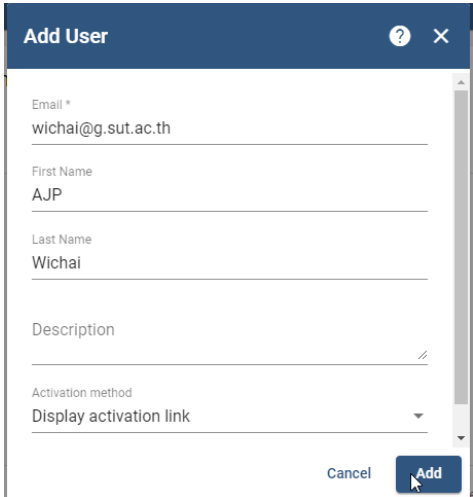


1. Customers Tab

2. Manage customer user

3. (+)Add

- Customers Tab
- Manage customer user
- (+)Add





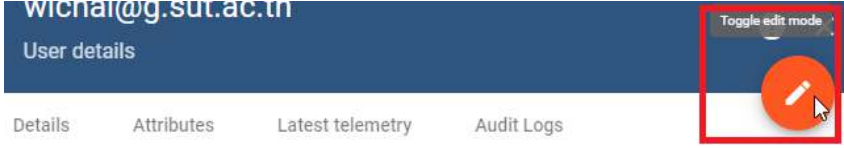

1. Fill Email

2. First Name

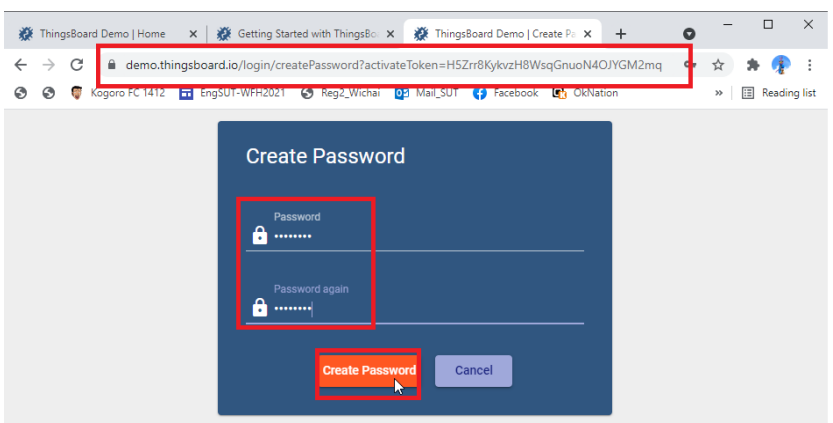
3. Last Name

4. Add

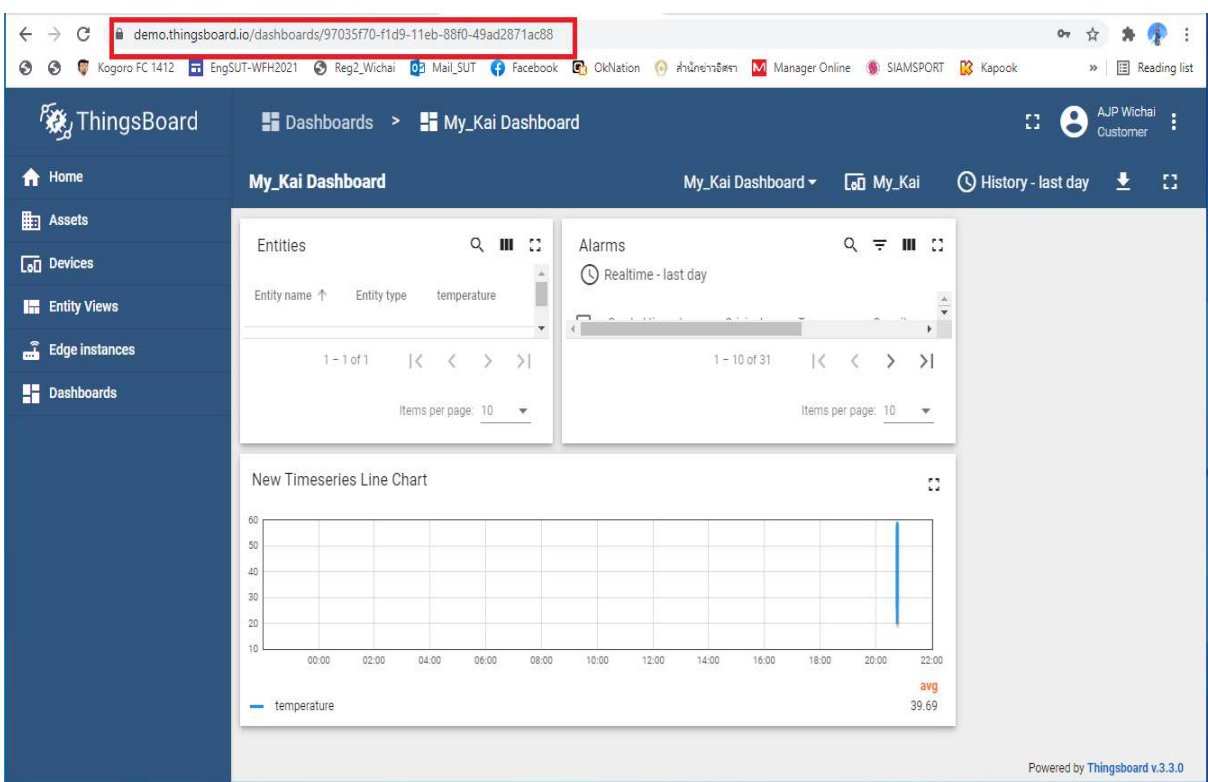
- Fill Email
- First Name
- Last Name
- Add

 <p><b>User activation link</b></p> <p>In order to activate user use the following <a href="#">activation link</a> :</p> <p><code>https://demo.thingsboard.io/api/noauth/activate?activateToken=H5Zrr8KykvzH8WsqGnuoN40JYGM2mq</code></p> <p>Copy activation link</p> <p>OK</p>	Copy activate link
<p><a href="https://demo.thingsboard.io/api/noauth/activate?activateToken=H5Zrr8KykvzH8WsqGnuoN40JYGM2mq">https://demo.thingsboard.io/api/noauth/activate?activateToken=H5Zrr8KykvzH8WsqGnuoN40JYGM2mq</a></p>	
 <p>My_Kai Customer: Customer Users</p> <p>Created time ↓ First Name Last Name Email</p> <p>2021-07-31 21:43:07 AJP Wichai wichai@g.sut.ac.th</p> <p>Click to open user detail</p>	Click to open user detail
 <p>wichai@g.sut.ac.th</p> <p>User details</p> <p>Details Attributes Latest telemetry Audit Logs</p> <p>Toggle edit mode</p>	Edit Mode
 <p>wichai@g.sut.ac.th</p> <p>User details</p> <p>Email * wichai@g.sut.ac.th</p> <p>First Name AJP</p> <p>Last Name Wichai</p> <p>Description</p> <p>Default dashboard 1 <input type="checkbox"/> Always fullscreen 2</p> <p>Home dashboard My_Kai Dashboard x <input checked="" type="checkbox"/> Hide home dashboard toolbar</p> <p>Apply changes</p>	<ol style="list-style-type: none"> <li>1. Allow access MyKai Dashboard</li> <li>2. Hide toolbar</li> <li>3. Apply Change</li> </ol>

## Step 7.5 Activate customer user



1. Open web browser
2. Insert activate link url
3. Fill password  
**{12345678}**
4. Create password



Dashboard URL = <https://demo.thingsboard.io/dashboards/97035f70-f1d9-11eb-88f0-49ad2871ac88>



การใช้งาน ThingsBoard IoTs Platform เพื่อสร้างและจัดการระบบอัจฉริยะ  
ThingsBoard IoTs Platform for smart system

ชื่อ-สกุล :

6/6 -- คำถามท้ายบทเพื่อทดสอบความเข้าใจ

Quiz\_101 – ThingsBoard Data Monitor

- Mission - 1/4: ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง Dashboard

โปรแกรมที่ใช้ทดสอบ
Capture Dashboard
รูปการทดสอบ 1
รูปการทดสอบ 2

Quiz\_102 – ThingsBoard Data Monitor and Control

- Mission 2/4: ให้ส่งข้อมูลค่า Humidity และ Temperatures จากเซ็นเซอร์ DHT-22 ไปยัง ThingsBoard พร้อมทั้งควบคุม On/Off - 4 LED และ Blink Speed สำหรับอีก 1 LED

โปรแกรมที่ใช้ทดสอบ
Capture Dashboard
รูปถ่ายหน้า Web Browser
รูปการทดสอบ 1
รูปการทดสอบ 2

**Quiz\_103 – ThingsBoard Data Monitor and control with MQTT Protocol**

- Mission 3/4: ให้ใช้ MQTT กับ ThingsBoard
  - ปรับปรุงเพื่อให้ทำงานควบคุมการ On/Off - 4 LED
  - เพิ่มเติม คือ ทดสอบส่งข้อมูล 1 ค่าแบบสุ่มระหว่าง 00 – 50 ไปแสดงที่ Dashboard ด้วย ได้หรือไม่  
 ทำอย่างไรบ้างให้อธิบาย {Read <https://thingsboard.io/docs/user-guide/device-profiles/> }

โปรแกรมที่ใช้ทดสอบ
Capture Dashboard
รูปถ่ายหน้า Web Browser
รูปการทดสอบ 1
รูปการทดสอบ 2

**Quiz\_104 – Web Control 4 LED and Monitor Humid/Temperature**

- Mission 4/4: การตรวจสอบและควบคุม อุณหภูมิ-ความชื้น ของโรงเรือนเลี้ยงไก่
  - ให้ใช้ ESP32 ส่งข้อมูลแบบสุ่มสองจำนวน คือ
    - Tempp\_A      สุ่มระหว่าง 20-40
    - Hudmid\_A    สุ่มระหว่าง 60-80
  - ข้อมูลทั้งสองค่าจะนำมาแสดงที่ Dashboard
  - สร้าง Alarm โดย หาก Tempp\_A > 35 หรือ Hudmid\_A > 70 ให้ Alarm
  - ศึกษาการตั้ง Alarm - <https://thingsboard.io/docs/user-guide/alarms/>
  - กำหนดรอบการตรวจสอบทุกๆ 20 วินาที
  - แชร Dashboard ไปให้ผู้ใช้งาน

โปรแกรมที่ใช้ทดสอบ
Capture Dashboard
รูปถ่ายหน้า Web Browser
รูปการทดสอบ 1
รูปการทดสอบ 2