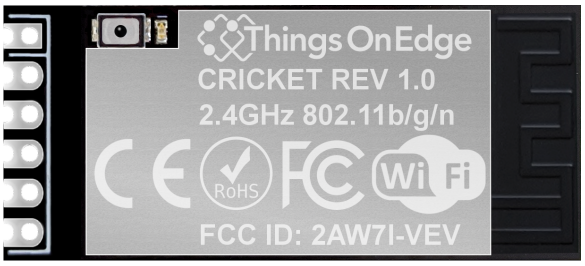
Things On Edge

thingsonedge.com

IOT Developer Guide

Technical documentation
Version 1.9



Cricket 1.0a

Cambridge, UK

TABLE OF CONTENTS:

Data privacy statement

Things On Edge keeps it very simple. **We DO NOT collect nor track any user data from anyone!** We love developing technology which serves people. We live from selling this technology.

We use personal data for purchases and shipments if you decide to shop on our Things On Edge - Store

About this guide

This guide endeavours to provide an ultimate technical background of what you as a developer need to know to use IOT Cricket Wi-Fi module, what components (buttons, sensors, etc.) you can attach to it, how Cricket works, how it connects to the Wi-Fi and internet, what services you can use, what communication channels are available such as MQTT and HTTP REST api and many more.

The **Platform** section provides a high level overview, connectivity options and possible integrations which come with a Cricket module.

The **Hardware** section contains technical information about the module such as power supply requirements, type of compatible batteries and many more. It is to give you an idea of what components you can attach to it.

The **Configuring Cricket** section provides technical information on how you configure devices either locally or remotely.

The **Communication APIs** section provides essential information on how you integrate your devices with the ecosystem software and services available on the internet. As it is not possible to explain all available services we picked up a few examples to give you an idea of how you can do it.

Platform

Overview

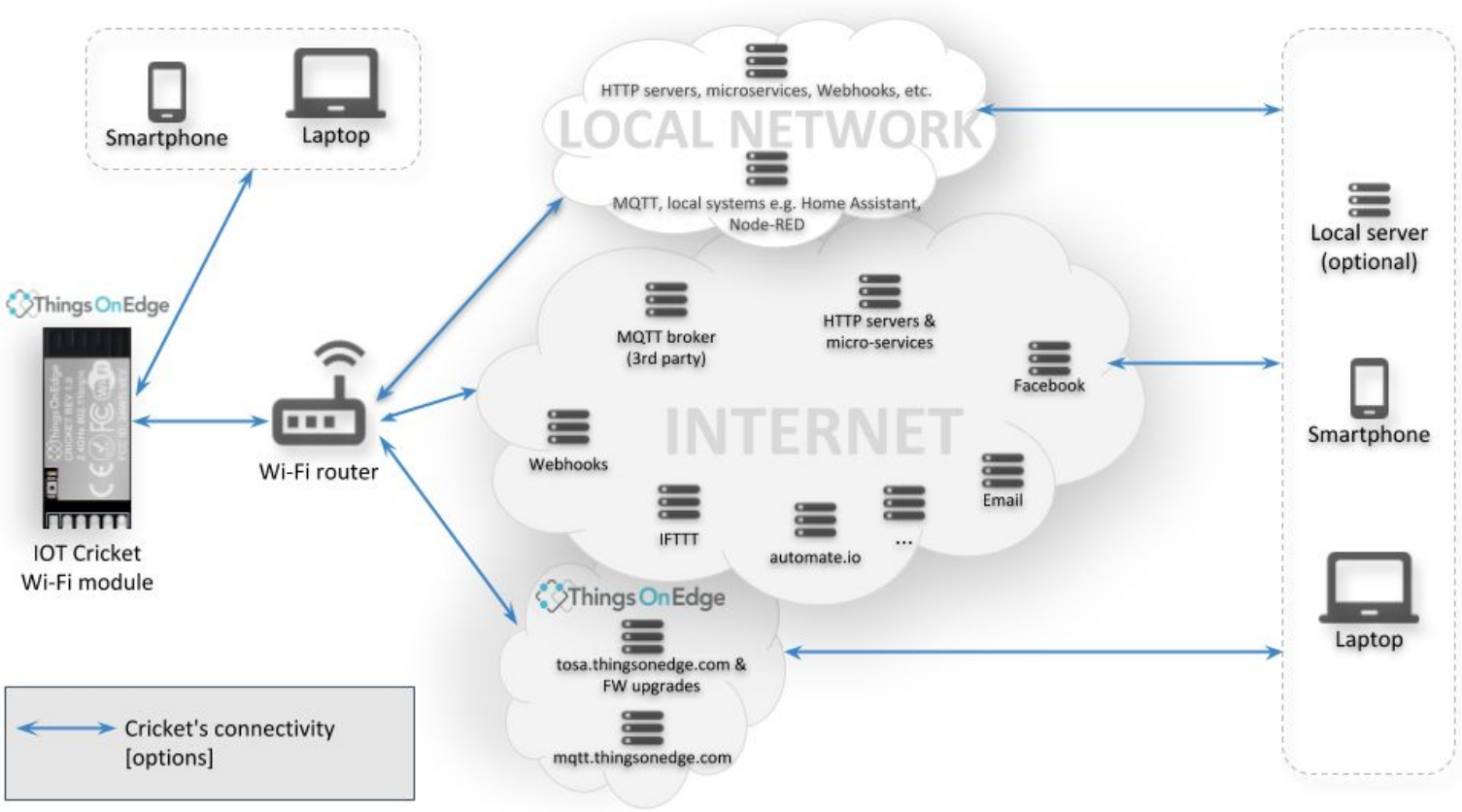
Things On Edge provides you an easy to use, ultra-low power Wi-Fi connectivity platform with the Cricket module. It is designed for developers, tinkerers, scientists, hobbyists, electronics device vendors, ... It allows you to make easy and fast IOT devices, power them on batteries for a very long time and integrate them to a huge ecosystem of systems and internet services. All this without writing a single line of code.

Key highlights of the platform:

1. A physical **IOT Cricket Wi-Fi module** - needs to be physically integrated into your device
2. An easy software integration to a huge internet ecosystems (over HTTP & MQTT)
3. OTA configuration of your devices either **Locally** or **Remotely**

It has been designed to sense & transmit data/events instantaneously from remote sensors, buttons, switches and other peripherals at low latency (~3 seconds). It doesn't require IOT hubs to connect your devices to the internet. It comes with a pre-installed software, which is fully configurable over the air from any web browser (smartphone, laptop, ...). You can manage device(s) either locally or remotely and integrate them to other systems using either MQTT or HTTP protocols. All this is explained in more details further in this guide.

The diagram below outlines the high level connectivity options, you have, that come with Cricket.

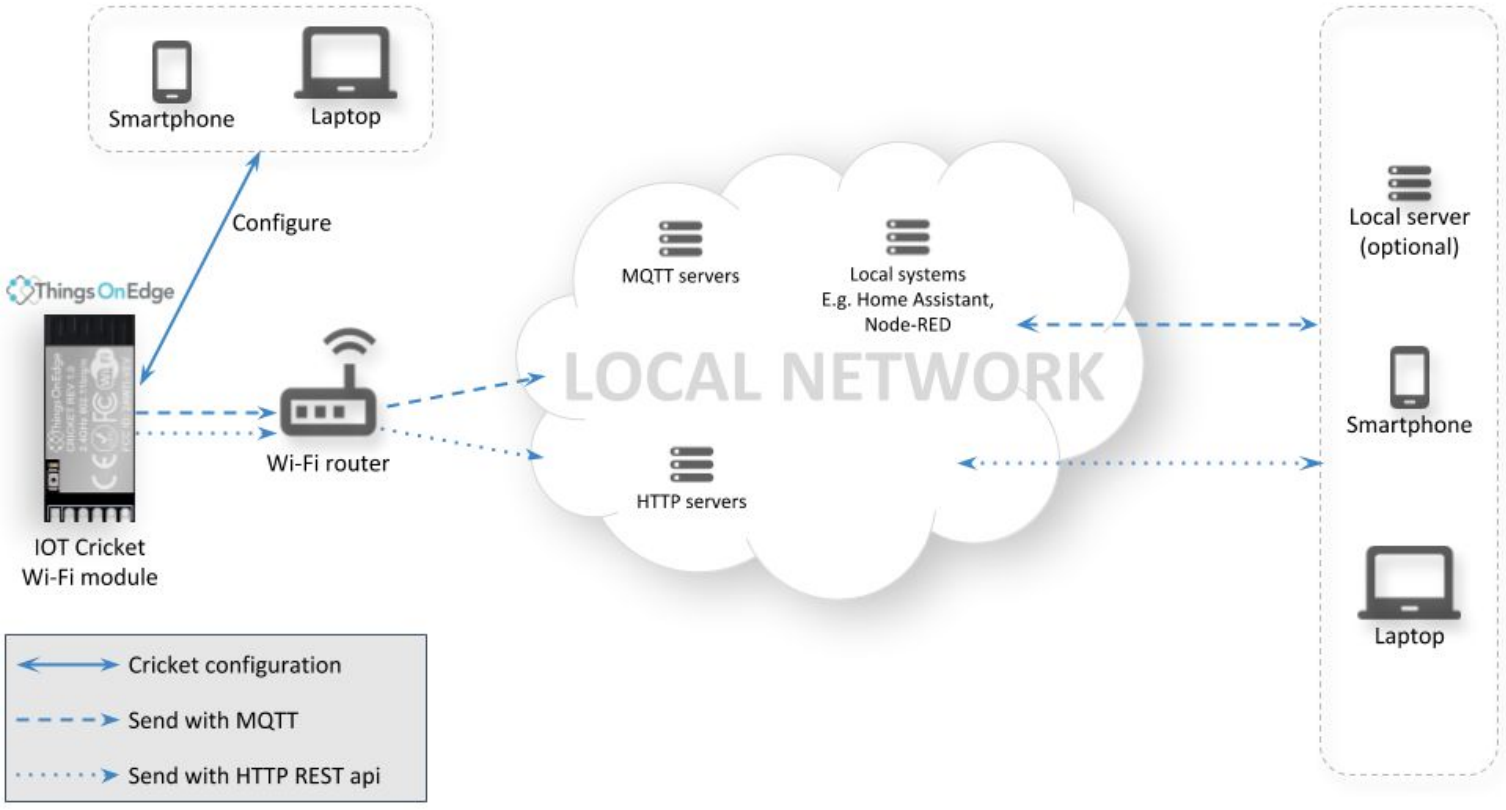


Connectivity

You can configure Cricket entirely either within your local network or remotely from the http://cota.thingsonedge.com microservice. Below you can find more information on each connectivity option.

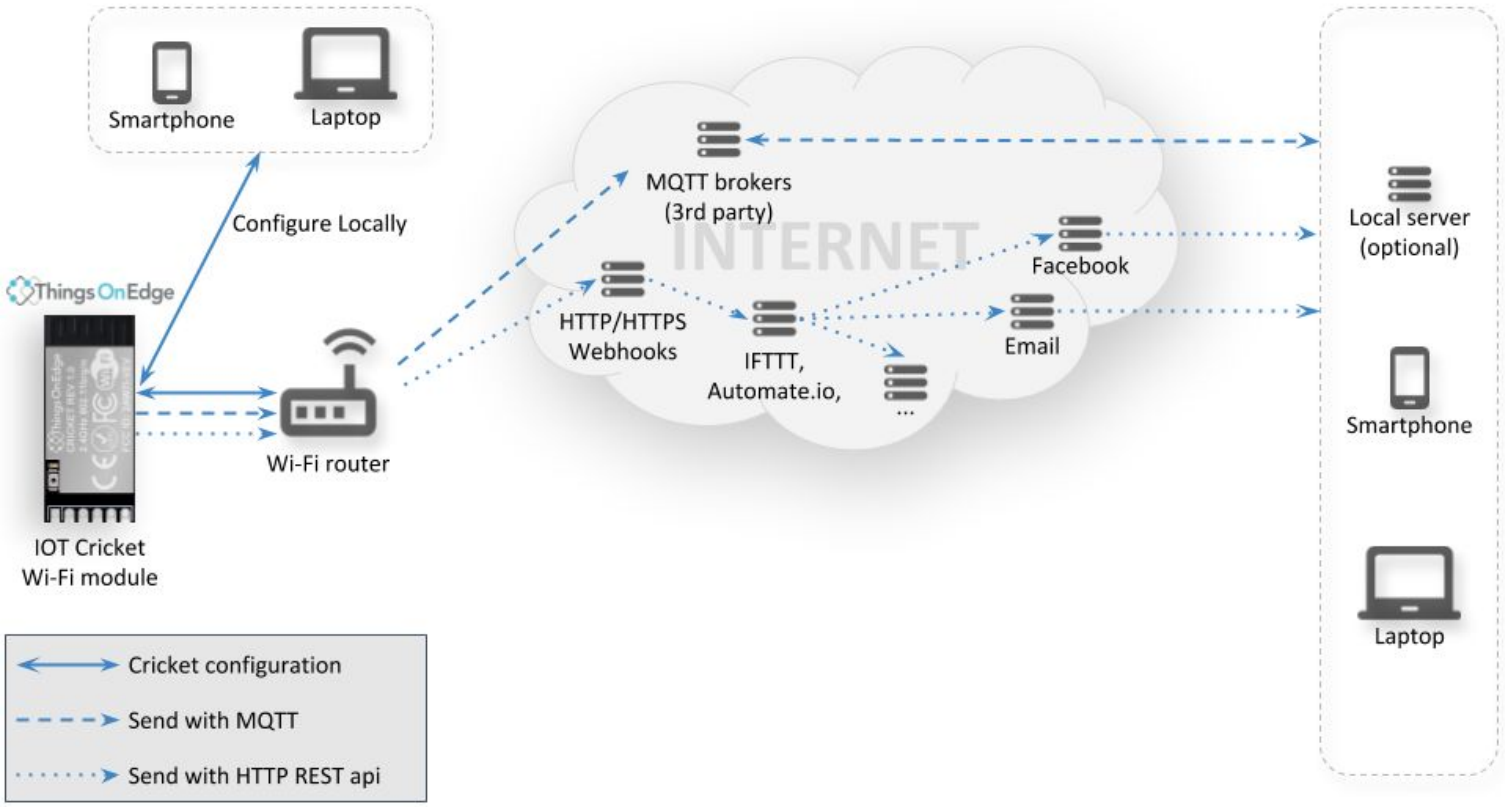
Local

Cricket can be entirely configured and used in your local network. You can configure Cricket directly on either a smartphone or a laptop via Cricket's Wi-Fi hotspot (toe_device). Then you can use either MQTT or HTTP communication channels to integrate Cricket to various systems in your local network e.g. Home Assistant



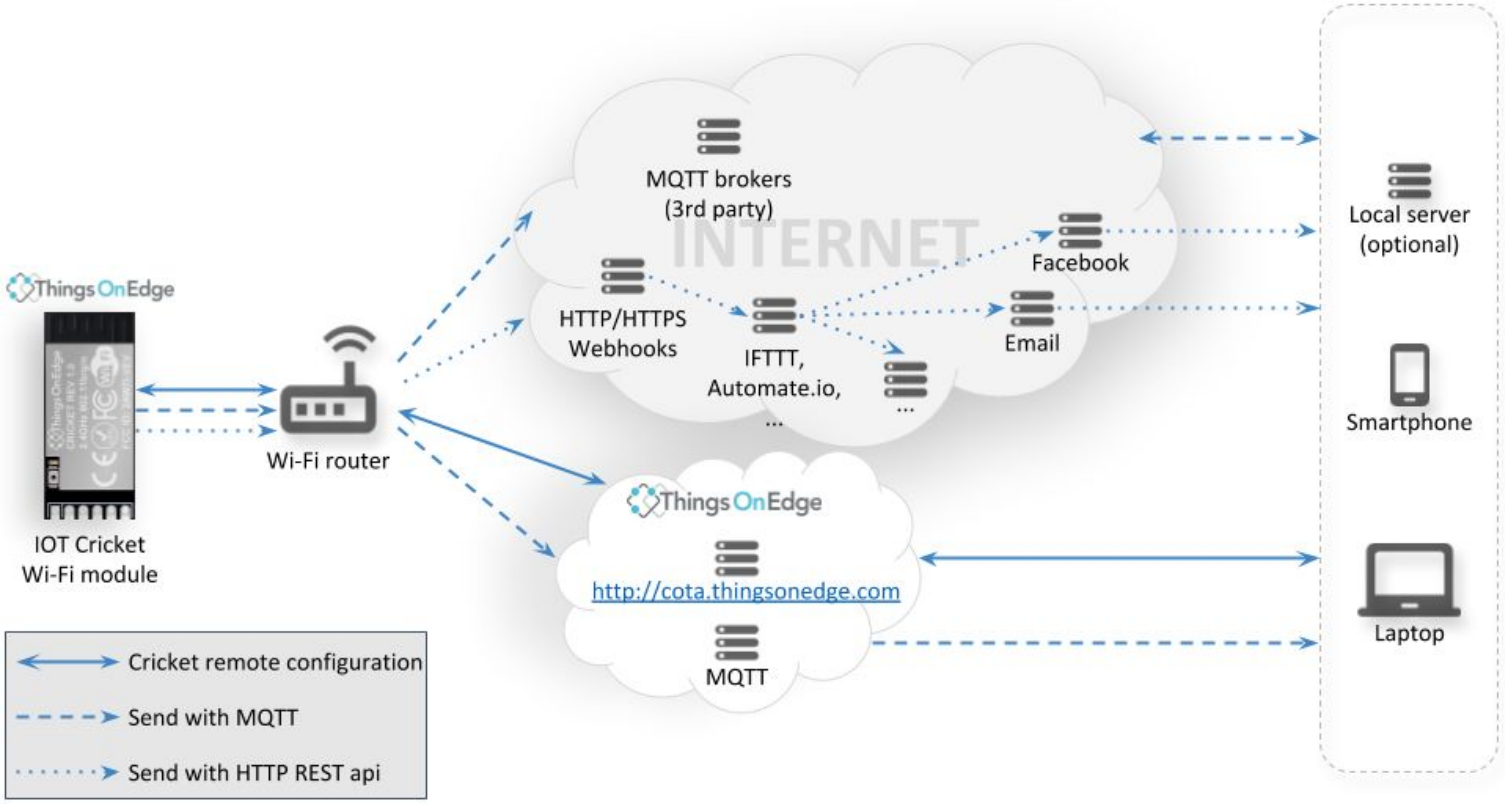
Configure locally & connect globally

Local configuration of Cricket is cloud/internet agnostic - you do not need them. As it is shown on the diagram below you can configure Cricket entirely locally (directly from Smartphone or Laptop) and send data globally over the internet to your chosen services.



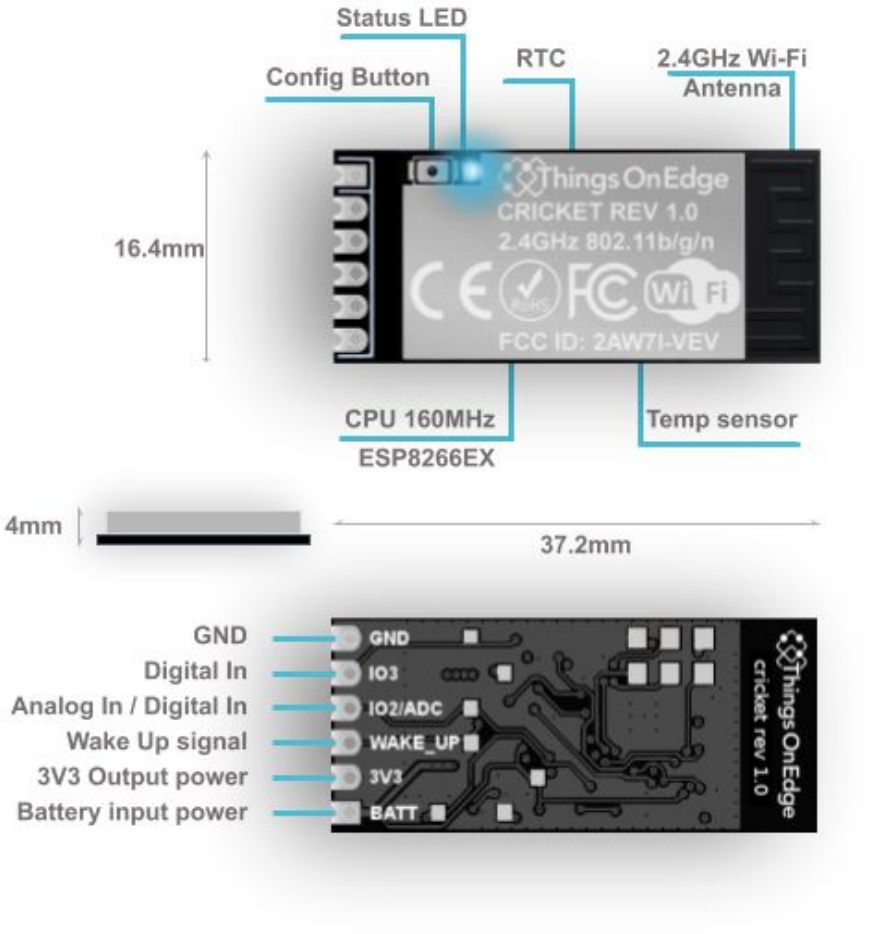
Configure remotely

Sometimes you might want to configure Cricket remotely from the internet. As shown on the diagram below Cricket comes with an option to be configured from the FREE of charge http://cota.thingsonedge.com microservice.



Hardware

Cricket Wi-Fi module



Key features

1. Ultra-low power, **true 0A** current when not in operation
2. Operates **directly on batteries below 3.5V** (AA, AAA, AAAA, ...)
3. Configurable **Analog** or **Digital inputs** for sensors, buttons, switches, ...
4. **Local configuration** (directly on a Cricket Wi-Fi hotspot)
5. **Remote configuration** (from TOE microservice)
6. Configurable **MQTT** (use either FREE Things On Edge or any 3rdparty MQTT broker)
7. Configurable secure / non-secure **HTTP** POST/GET requests
8. Configurable **battery monitor**
9. Built-in configurable Real-Time Clock (**RTC**) for regular wake ups with specified time intervals
10. Built-in configurable **temperature sensor**
11. Firmware updates

Applications

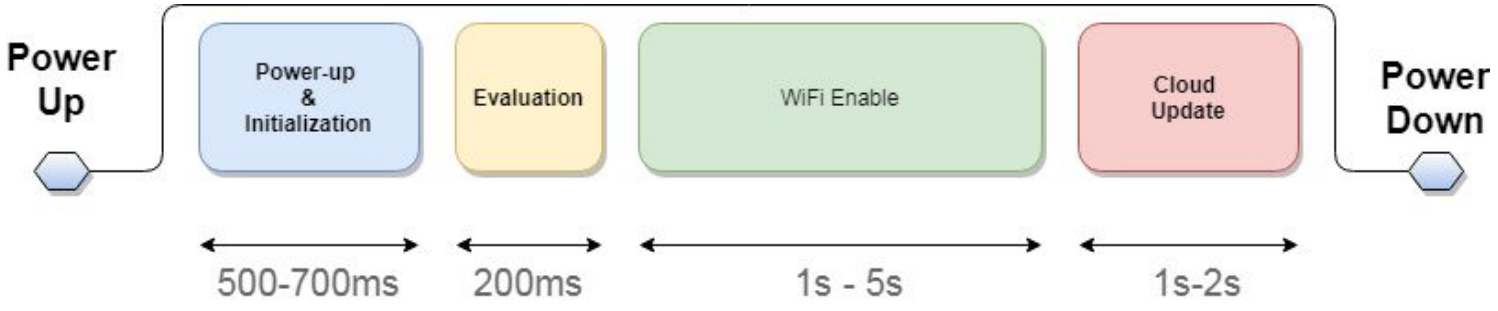
Below are a few example scenarios of what you can do with Cricket:
> Send notifications when someone is at your doorstep
> Ring your phone when someone presses a doorbell button
> Report moisture level in a flower
> Raise alarm on your phone when the moisture level goes below or above certain thresholds
> Report a local temperature information (garden / home / room)

> Report information about windows / doors are closed / open
> Report information when alarm goes off
> Report a detected movement either via email or ringing your phone
> Report noise detection
… there are literally endless applications for reporting various data from any remote sensory devices.

Functional description

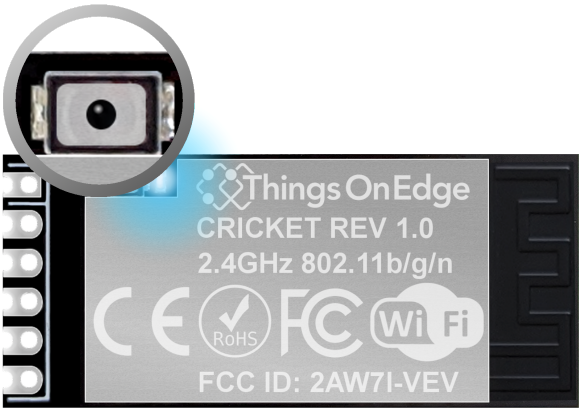
The Cricket module has 2 operation modes: **Normal** & **Configuration**

Normal mode
Normal mode is triggered either by an internal RTC or a signal raised on the WAKE_UP pin by a connected external component.



Power-up & Initialization	When the module wakes up it performs the following firmware initialisations: > Load FW from non-volatile memory > Read configuration from non-volatile memory > Initialize platform modules such as RTC, temperature sensor and GPIO's. Once the board is initialized it performs sensors evaluation. Only sensors which were configured are evaluated for example if the temperature sensor is not enabled the module will not read current temperature from the sensor.
Evaluation	There are a number of steps applied for each sensor evaluation process such as: > Read the current value > Store the current value in non-volatile memory > Only if the current and stored values are different or "Force updates" is on then the current value is sent to the internet. It is sent via either HTTP or MQTT which you configured.
Wi-Fi Enable	The Wi-Fi connectivity process is conditionally triggered based on the evaluation process. IF at least one value needs to be sent to the internet OR "force-updates" is On THEN the WiFi is enabled ELSE the board will power off and wait for another trigger Cricket connects to Wi-Fi, which was set from the Pairing / Binding process described further in this guide.
Communication	Once a WiFi connection is established, Cricket sends data & events out to either a local network or the internet (see Communication APIs section).

Configuration mode
The configuration mode is activated when you press an on-board button for **~5 seconds**.

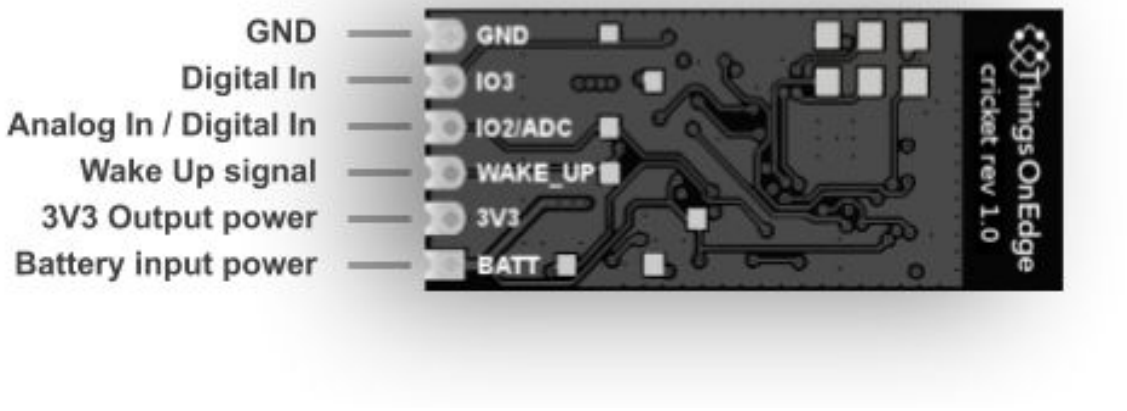


You know it is active when the LED is flashing fast with regular intervals. The board opens **toe_device** Wi-Fi hotspot for **~2 minutes**. You can connect to it either from a laptop or a smartphone to pair Cricket to your Wi-Fi network and configure it entirely locally. It is explained in more detail in the Configuring Cricket section.

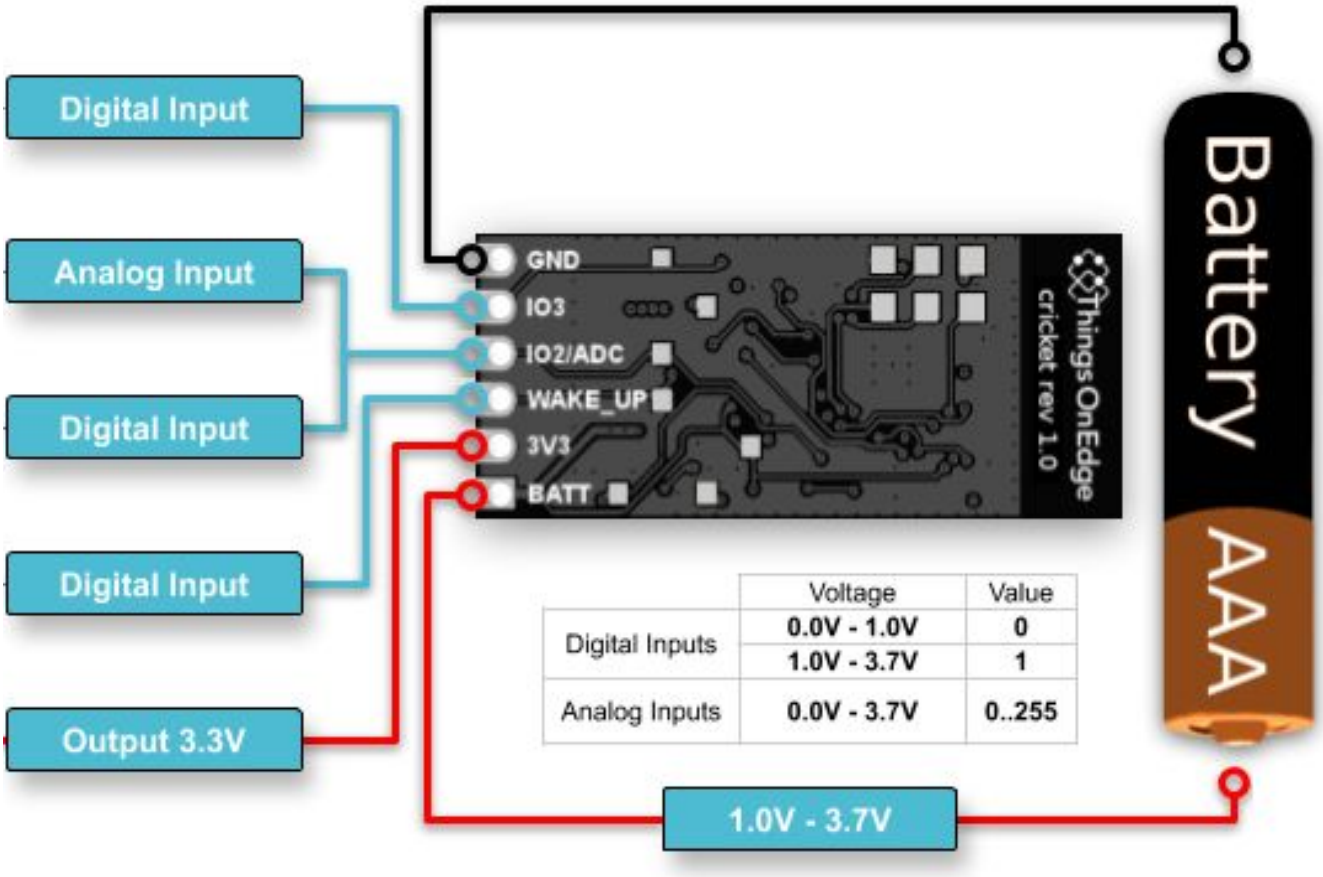
Specification

Physical dimensions	Length: 37.2 mm Width: 16.4 mm Height: 4 mm
Operating wireless range	Up to 100 meters
Wi-Fi	2.4GHz 802.11 b/g/n WPA / WPA2
Operating Voltage	1V~3.7V
Operating Temperature Range:	-20°C~80°C
Chipset	ESP8266EX CPU 32bits @160MHz

Pinouts



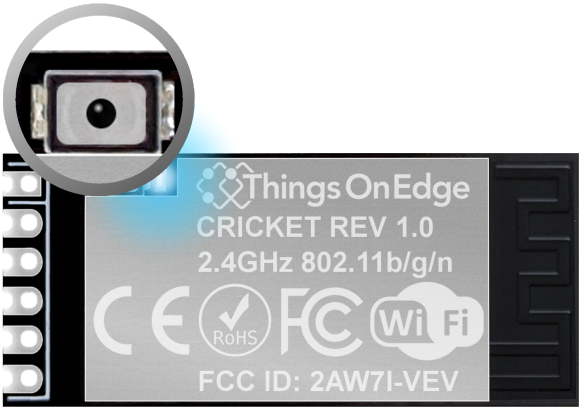
Pin	Description
GND	Ground
IO3	Digital input
IO2	Digital or Analog input signal
WAKE_UP	HI level on that pin will turn board on
3V3	Output power from internal regulator. This is always 3.3V regardless on BATT voltage level
BATT	Power supply VDD to the board, this can be connected directly to a battery (see compatible batteries section)



	Voltage	Value
Digital Inputs	0.0V - 1.0V	0
	1.0V - 3.7V	1
Analog Inputs	0.0V - 3.7V	0..255

Built-in button

Cricket comes with the built-in button, which serves the following functions:



Function	Keep pressed for	Description
Wake Up	**less than 1 second**	Wakes up (same as WAKE_UP pin)
Fetch Remote Config	**> 1 second**	Fetches a configuration from http://cota.thingsonedge.com (optional for remote configuration only see Remote configuration)

Open Local Configuration	**> 5 seconds**	Enters into the full local configuration mode and opens **toe_device** Wi-Fi hotspot (see the Configuring Cricket section)

Power supply

Power supply is designed to operate directly on the batteries and provide low power consumption for both when operating and not operating in power-off states.

In order to power-up the module the power source must provide at least 100mA@3.3V continuously and 0.5A@3.3V for ~1ms peak which usually occurs once per wake up.

Compatible batteries

Cricket's operating voltage is 1V - 3.5V and can be powered almost on any battery. However you need to take into account the battery voltage as you might need to use a DC-DC step down regulator. All batteries below 3.5V can power Cricket directly and all higher than 3.5V must use the step-down regulator.

Recommended operating conditions

The table summarises general operating conditions to be met:

Operating Condition	Symbol	Min	Max	Typical	Unit
Supply voltage	BATT	1	3.6	3	V
Wake up voltage	WAKE_UP	1	3.6	3	V
Operating temperature		-20	80	20	℃

Temperature sensor

Cricket has a built-in configurable temperature sensor TMP1075DSG. It has the following parameters shown in the table below.

Operating Condition	Value	Unit
Temperature range	−40°C to +75°C	°C
Resolution	0.5	°C
Accuracy	±1	°C

The temperature sensor is evaluated only if it is enabled. It is important to note that the temperature value, such as other sensors, are not always sent out by Cricket. Cricket is optimised to prevent sending data

when values do not change. However you also have an option to enforce Cricket to send data regardless of value changes. The table below illustrates scenarios when the value is sent on wake up events.

Temp value (initial value 20)	Sending out temp data	
	Force updates: OFF	Force updates: ON
20.0	NO	YES
20.5	YES *	YES
20.5	NO **	YES
20.0	YES *	YES

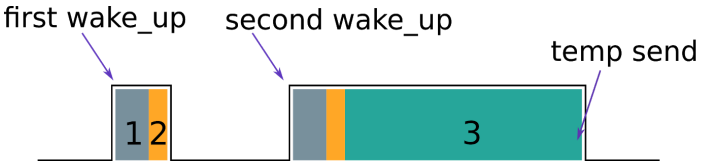
* temperature value is sent because is different compared to previously cached value by at least 0.5°C
** temperature is not sent because the difference is less than 0.5°C

When "**Force Updates"** is on then Cricket sends the temperature value on every wake up event, regardless of values cached previously.

You can extend the battery life of your device by tweaking a few parameters. For the temperature measurements you can think of the two key aspects to reduce the power:
> Reducing power consumption for a single event
> Reducing the number of events

Reducing power consumption for a single event

One of the Cricket's features is that it caches evaluated values from sensors by default. They are compared to the currently evaluated values from sensors and only if the values are different they are sent out. Below there are two wake ups shown with sending and not sending data.



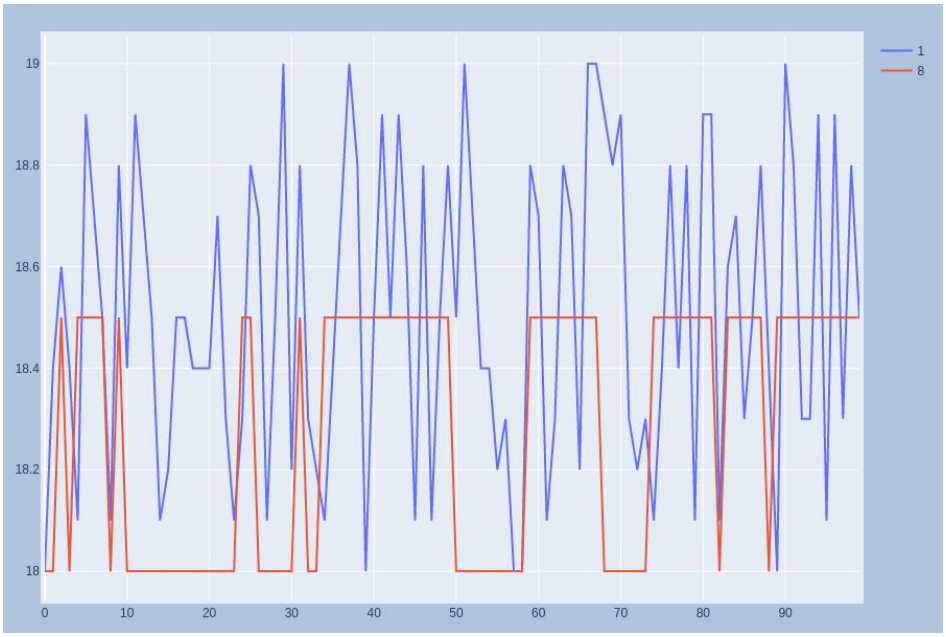
The *first wake_up* event does only FW initialization and sensor evaluation. Cricket determines that cached and current values are the same and there is no need to send data out.
The *second wake_up* event does FW initialization, sensor evaluation and sending data. Cricket determines that cached and current values are different and sends data out.

On average you can reduce 6x power consumption of the batteries when Cricket doesn't have to send data after sensor evaluation. You can save even more power when the Wi-Fi router and the internet connectivity happens to be slow.

It is recommended to keep "**Force Updates**" disabled to reduce connectivity to minimum and extend the battery life.

Reducing the number of events

Reducing the number of events with **averaging temperature** is another aspect of reducing power consumption. By decreasing the temperature sensor resolution, which also reduces the ambient temperature noise, you can reduce the number of connections respectively. Cricket performs less connections which results in more power saving. An example below shows the difference between the temperature read from the sensor and the average temperature of 8 samples.



The blue line in the graph represents a real temperature measurement in a home environment. It shows thermal noise usually not bigger than 0.5°C. However this may produce a significant number of unnecessary events for Cricket to send data and unnecessary power consumption. This can be reduced with the averaging temperature function which Cricket already has for the built-in temperature sensor. The red line shows a calculated average by Cricket from 8 recent samples. The graph shows an effective way of reducing either noise or spikes. It not only results in producing accurate results but also significantly improves the battery life.

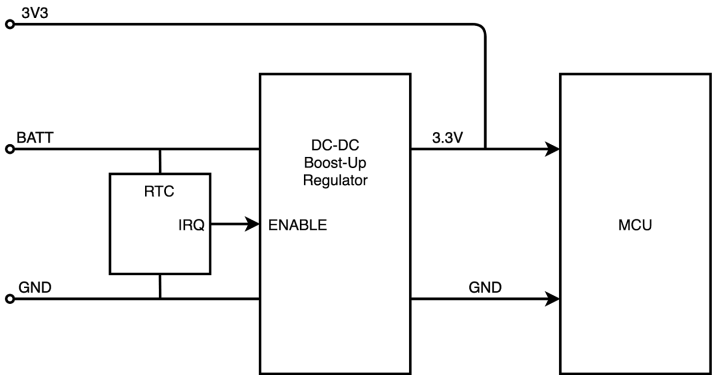
The averaging function takes N recent values stored in cache and calculates the average value, which then is stored in cache.

You can configure it. The averaging samples option appears only when the temperature sensor is enabled.

Both the “avg num points” parameter and wake up frequency can be tuned to increase battery life time. The wake up frequency can be further tuned with the RTC, which is explained below.

RTC (Real-Time Clock)

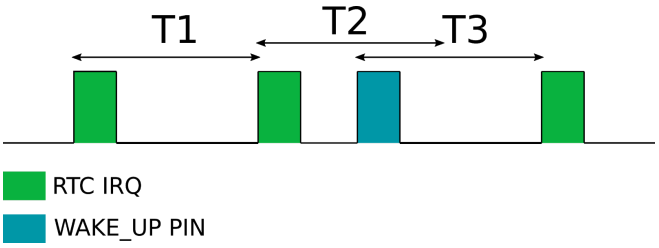
Cricket has a built-in RTC (Real-Time-Clock) module for performing regular wake ups. You can configure it with a selected time interval with the following units: Seconds, Minutes or Hours. They cannot be mixed. For example you can set a 3 hour time interval but not 3 hour and 30 min. Below you can find more details on power requirements and how it works.



The RTC requires ~0.3uA. It is connected directly on the battery power domain. Thus it remains powered on even when Cricket is off as long as the battery provides continuous voltage.

When RTC reaches the end of the configured period of time, it wakes up Cricket by providing a power supply to MCU. Then MCU boots FW which makes Cricket to enter into “Normal” mode operation described in the Functional description section. When RTC interrupt occurs FW reloads the alarm interval from the current configuration.

The RTC works with other wake up external sources connected to the WAKE_UP port.

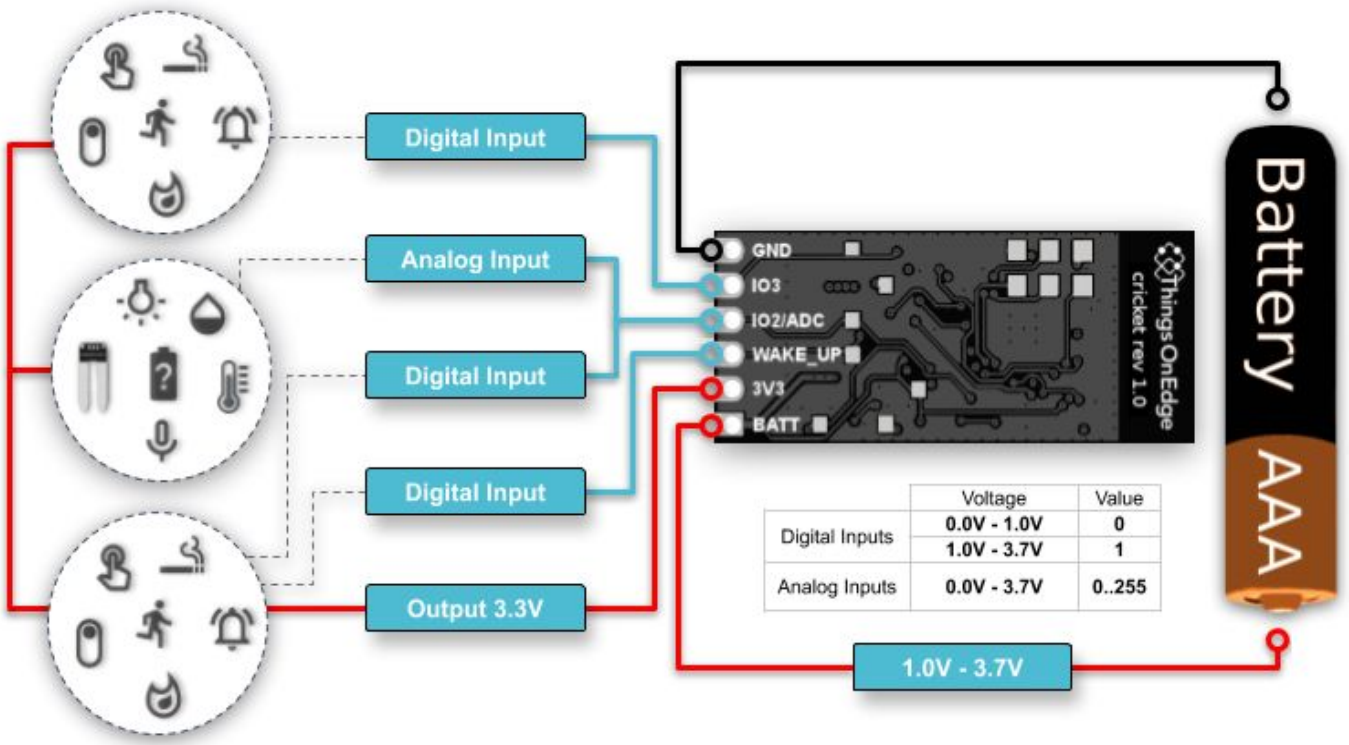


T1, T2, T3, … RTC time interval set 30 min

The diagram above shows mixed WAKE_UP signals. The first and second (green) wake ups come from the pre-configured RTC with a specified time interval (30 min). The next signal (blue) comes from the WAKE_UP pin and occurs in the middle of the RTC interval time. This (blue) signal wakes up Cricket, resets RTC to the pre-configured time (30 min) and restarts counting from 0 min. The last wake up (green) comes from RTC 30 min after the blue wake up.

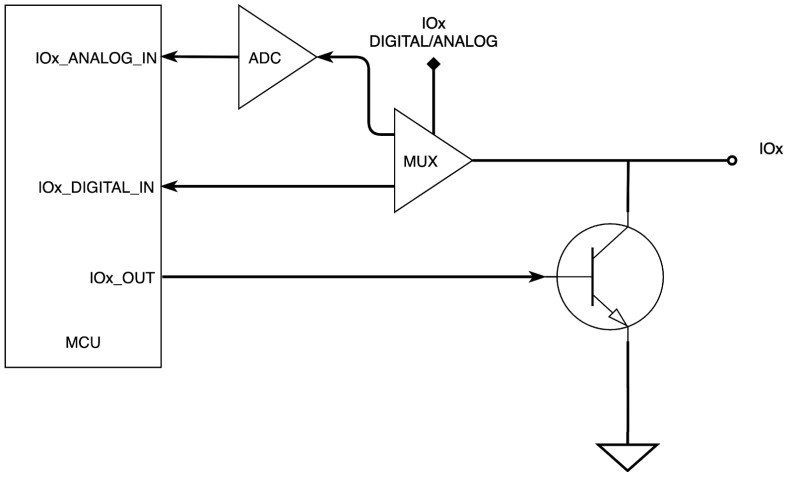
By default Cricket does not send data out on every RTC wake up. Unless values have changed. If you want Cricket to send data regardless of values being changed, set “force-update” to “On”.

IO ports



	Voltage	Value
Digital Inputs	0.0V - 1.0V	0
	1.0V - 3.7V	1
Analog Inputs	0.0V - 3.7V	0..255

Cricket has a general purpose IO port marked as IO2 / ADC on the board. You can configure this port as either Analog or Digital input from Cricket's configuration panel (see Configuring Cricket section).



Digital input port characteristics				
Operating Condition	Min	Max	Typical	Unit
VIL	0	0.7	0	V
VIH	1	3.5	3.3	V

Analog input port characteristics			
Operating Condition	Min	Max	Unit
Resolution	1	8	bits

Voltage range	0	3.5	V

Cricket sets the analog input resolution to 8 bits by default. However if you do not require such high resolution as you might want to reduce noise; reduce the number of events sent to the internet; or any other reasons you can change it. Go to Configuring Cricket and set the “resolution(bits)” parameter with a desired value. Note the parameter appears only when you select “ANALOG_IN” for IO2.

Analog value is an integer, which is sent to the cloud. It can be converted back to the input voltage with following formula:

$$\mathbf{Vin = (Vref / 2^{resolution}) * IO2}$$

Where

IO2:	*Value reported by Cricket to the cloud (see MQTT and HTTP POST Requests). The value range depends on the “resolution(bits)” parameter (see the table below).*
Vref:	3.5
“resolution(bits)”:	*Number of bits (max. 8) configured (see Configuring Cricket)*

resolution(bits)	Max IO2 value
8	255
7	127
6	63
5	31
4	15
3	7
2	3
1	1

Compatible external peripherals - a few examples

As it is impossible to list all possible sensors, buttons and switches that you can attach to Cricket we list just several to give you a flavour and inspiration on what devices you can attach to Cricket.

Temperature sensor [external]	TMP235 - Plug-and-Play STEMMA Analog
Light sensor	Light, Adafruit GA1A12S202 Log-scale Analog Light Sensor
Accelerometer	Level sensor, ADXL335 - 5V ready triple-axis accelerometer
Microphone	MEMS zero-amp mic sensor for noise detection

Motion sensor	HC-SR501 Human Body Sensor with MP1584EN DC-DC Buck Converter
Current Sensor Breakout	Adafruit INA169 Analog DC Current Sensor Breakout
Magnetic switch	Door Window Magnetic Switch
Moisture sensor	Soil Moisture Sensor
Button	Big Dome Push Button
...	...

Configuring Cricket

Configuring a.k.a. programming Cricket is done entirely OTA (Over The Air) over Wi-Fi. It doesn't require any programming and can be done from any device with Wi-Fi and web browser capabilities e.g. smartphone, tablet, laptop, smart TV, PC with Wi-Fi connector, etc.

You can choose to do it either **locally** over Cricket's toe_device Wi-Fi hotspot or **remotely** from the http://cota.thingsonedge.com microservice.

Local configuration

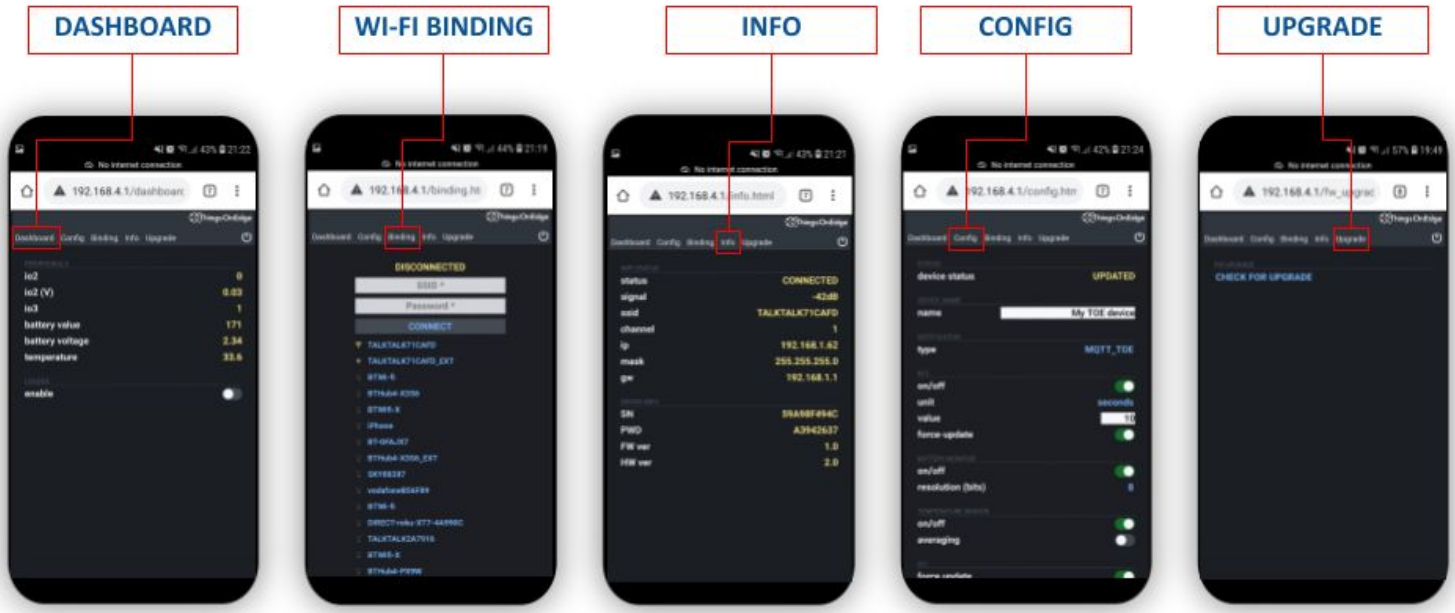
You can configure Cricket entirely within your local network which doesn't have the internet access.
The latest version of Cricket works in its full capabilities within isolated local WiFi networks.

In order to start configuring Cricket locally go to the Cricket Configuration Panel section.

Cricket Configuration Panel

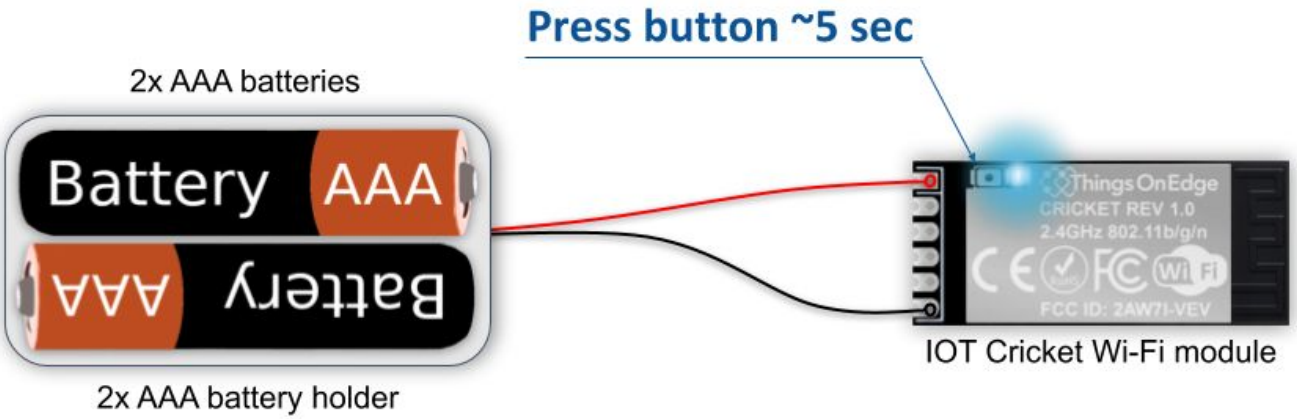
Cricket comes with its local Configuration Panel which allows you to:
> Pair Cricket to your Wi-Fi network from the BINDING panel
> Monitor your Wi-Fi network connectivity status from the INFO panel
> Monitor the actual data read from sensors and other peripherals from the DASHBOARD panel

> Configure Cricket locally from the CONFIG panel
> Upgrade firmware from the UPGRADE panel

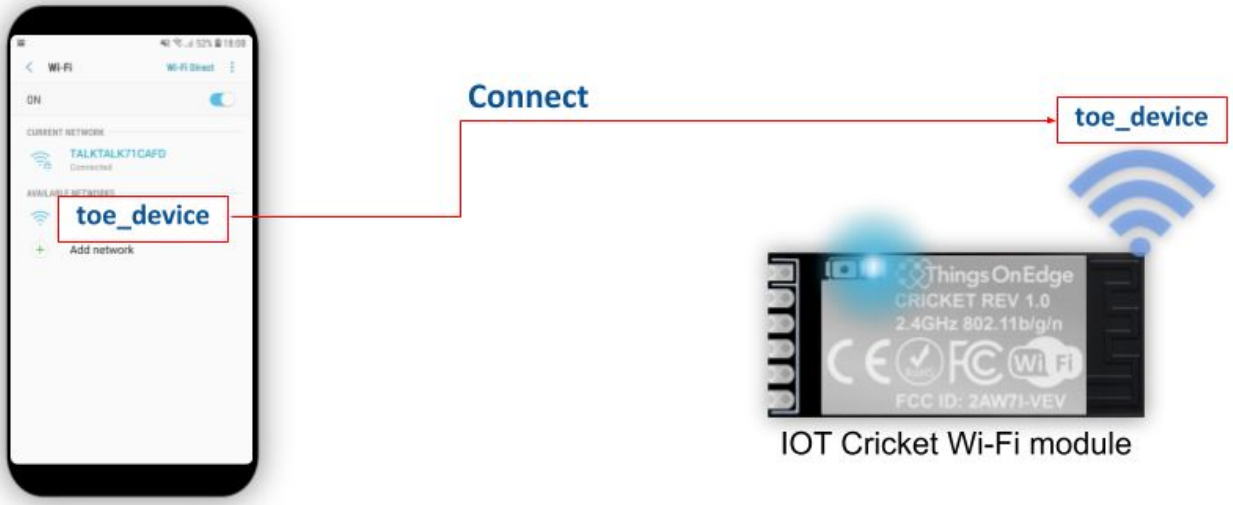


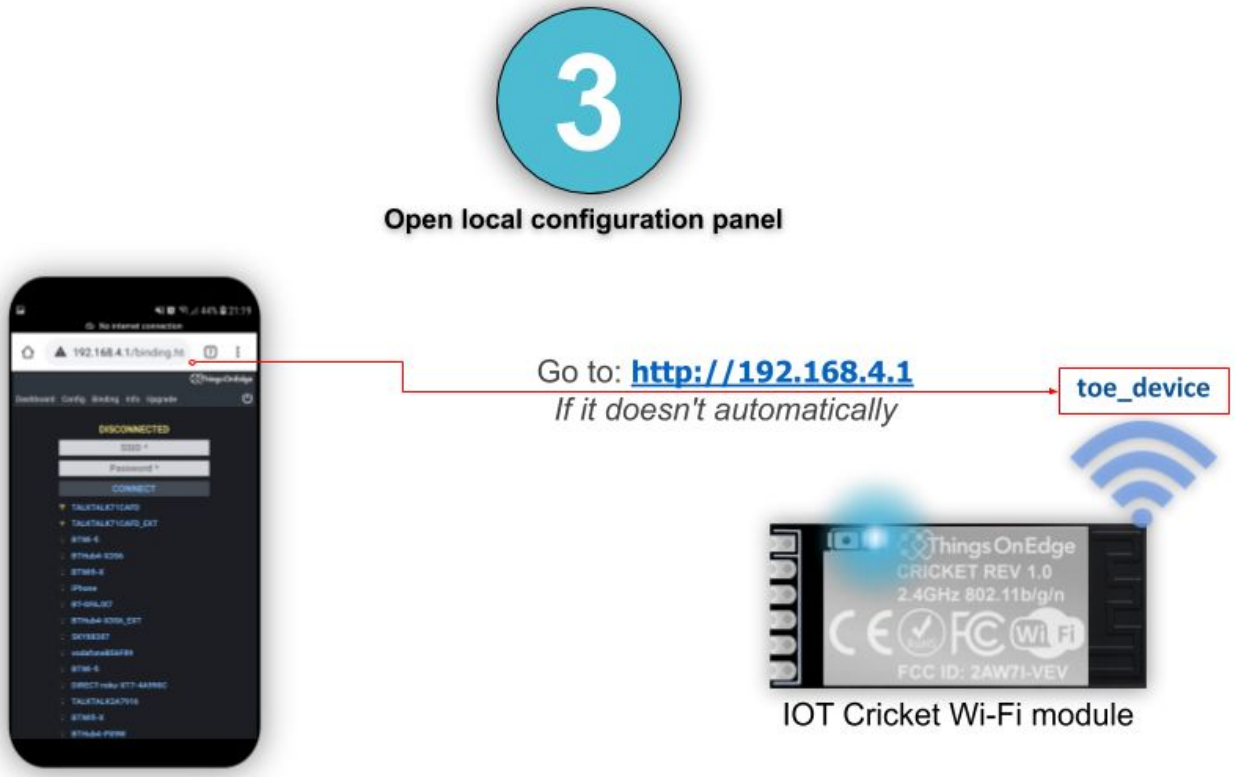
In order to open the configuration panel please follow the steps below:

Open private “toe_device” Wi-Fi hotspot



Connect from a phone or laptop to the “toe_device” Wi-Fi hotspot

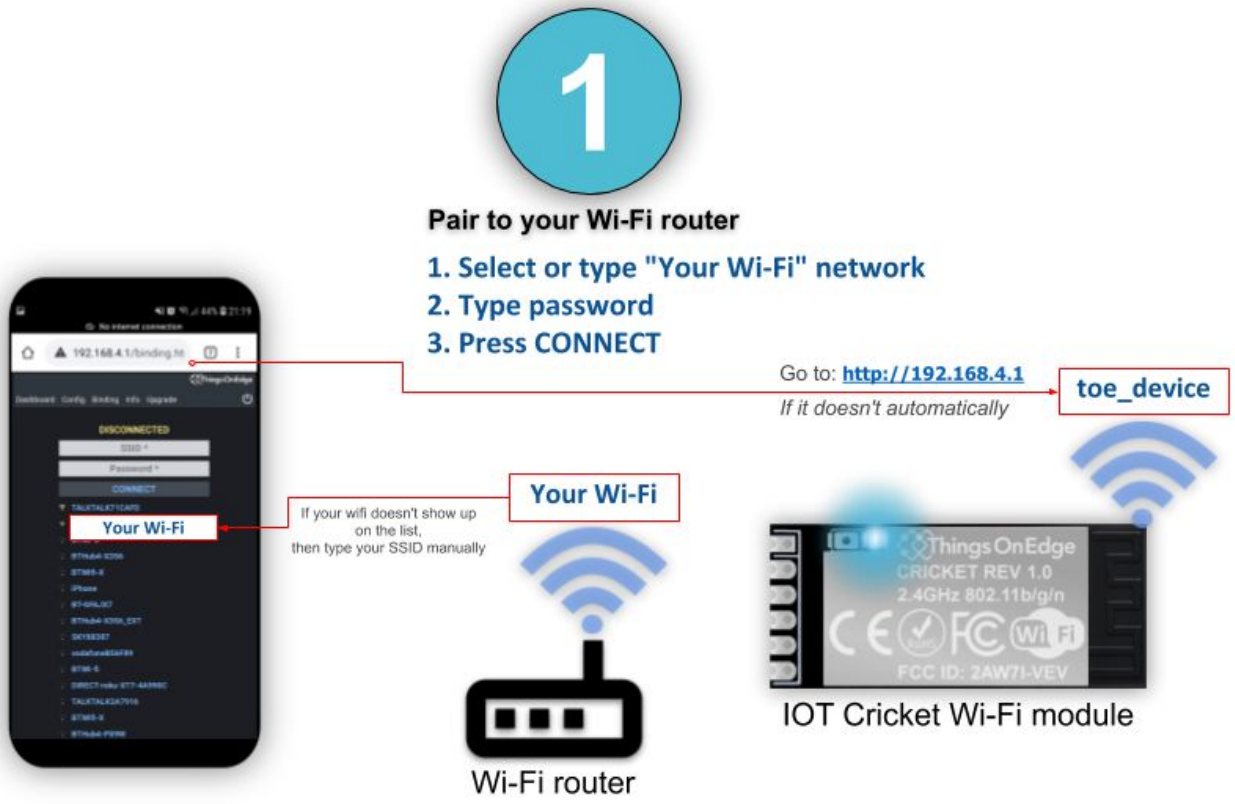


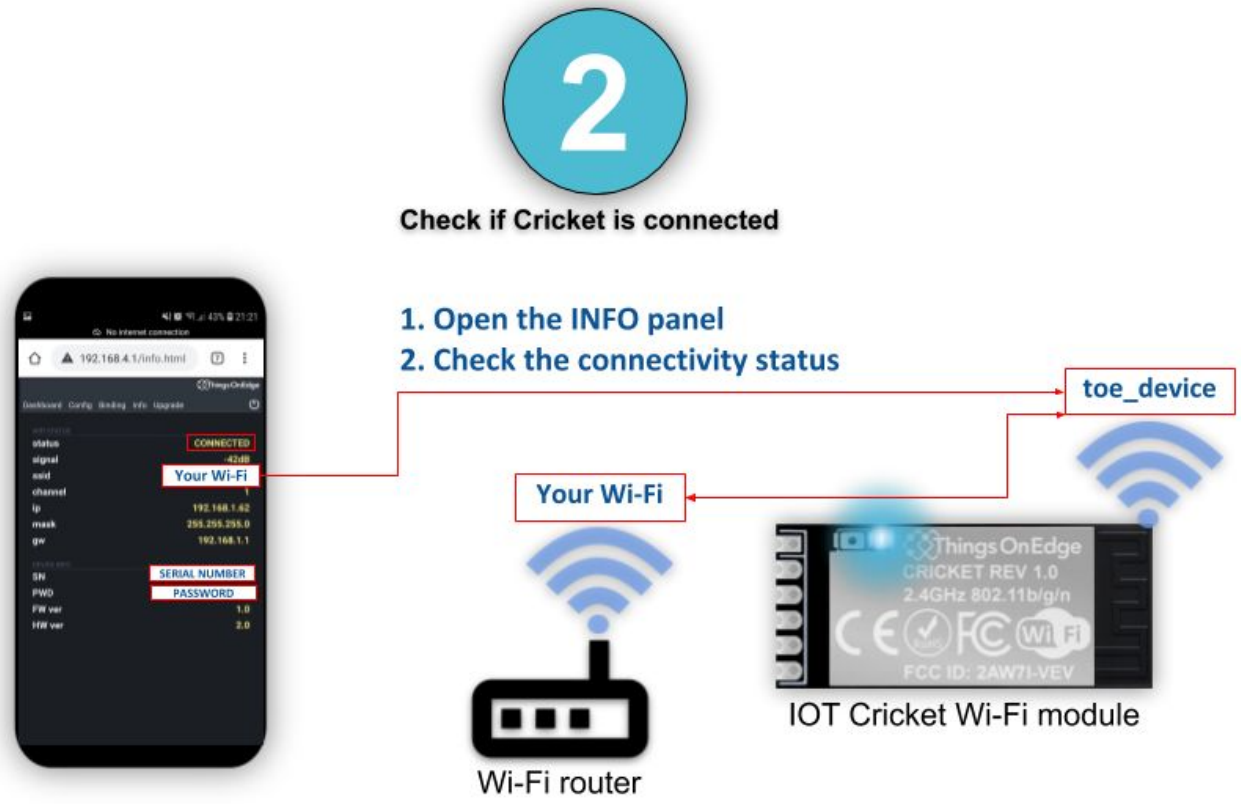


Now you can configure Cricket entirely from this local configuration panel. It is recommended to pair Cricket to your Wi-Fi network with the first step (see BINDING (Wi-Fi pairing) section) to unveil full Cricket's capabilities.

BINDING (Wi-Fi pairing)

In a few steps you can connect Cricket to your Wi-Fi network as shown below from the local Configuration Panel (see Cricket Configuration Panel section how to open the configuration panel)





Now you can start configuring Cricket (see the CONFIG section) and set it to send data out to client devices: smartphone, laptop, PC, … To find out more how to receive data from Cricket on client devices go to the Communication APIs section.

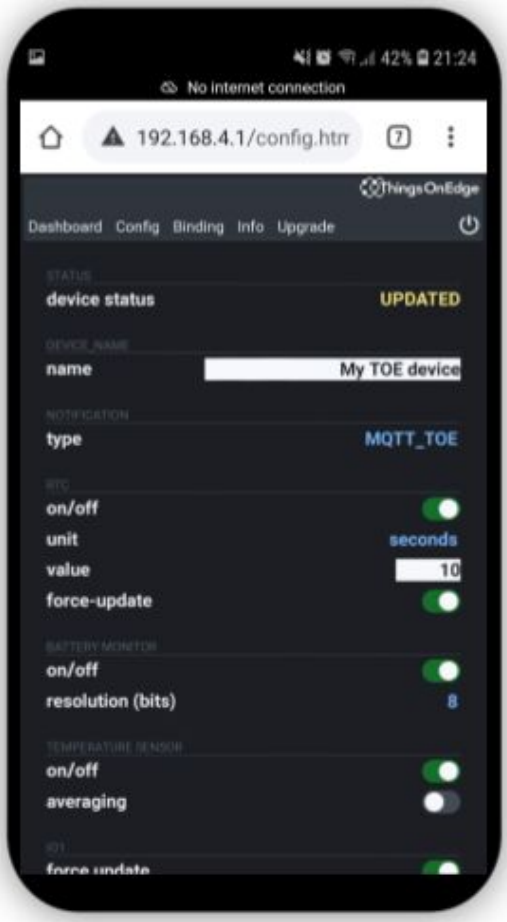
DASHBOARD

Cricket comes with a DASHBOARD panel where you can monitor actual values read from sensors on IO pins, battery status, built-in temperature sensor and the like. It may be useful for you to see what values are read by Cricket before you start sending them out. Though please notice that the only values which are being updated on the panel are the one you enabled in the CONFIG panel.

You can also enable / disable logging to get more information on what is actually happening on the device. The logging comes with a very limited print out buffer and they do not rotate. If they reach the buffer limit you need to clear it manually. In order to do that you need to enable / disable the switch button and the logger buffer will be cleared.
NOTICE: for the performance reasons it is strongly recommended to **keep the logger disabled**!

CONFIG

Regardless if you are configuring locally (see Cricket Configuration Panel section) or remotely (see Remote configuration section) there are the same parameters which you need to set such as RTC, physical ports, temperature sensor, MQTT & HTTP connectivity, etc. The configuration must reflect how you assembled your device and to which physical ports you connected sensors / peripherals. In the table below you can find a detailed description of each parameter and the meaning of its value(s).



Section	Parameter	Value	Description
DEVICE NAME	name	name	It is your arbitrary name of a device
CONNECTIVITY	type *[MQTT_TOE]*	MQTT_TOE	Use “Things On Edge” MQTT broker. Once you set this you do not have to configure MQTT it is already pre-configured for you. See MQTT Client configuration section how to receive data and events on client devices.
	type *[MQTT CUSTOM]*	MQTT CUSTOM	Configure Cricket to send data to your custom MQTT broker.
		url	Address of your MQTT broker. If you want to configure a port number use the following format: address:port e.g. m13.cloudmqtt.com:17605
		user	User name of your custom MQTT broker
		password	Password to your custom MQTT broker
	type *[HTTP_POST]*	HTTP_POST	Configure Cricket to send data over HTTP POST requests triggered when module is woken-up
		url	Defines URL address to which Cricket sends data via HTTP POST request

		payload	Define your custom payload and add #tags interpreted by Cricket. Find the list of tags at HTTP dynamic tags section of this guide.
	content-type *[HTTP_POST]*	auto	Cricket automatically detects format of the payload if it is either a plain text or JSON
		text/plain	Set explicitly payload format as plain text
		application/json	Set explicitly payload format as JSON
	type *[HTTP_GET]*	HTTP_GET	Configure Cricket to send data over HTTP GET request method, triggered on the module wake up
		url	Defines URL address to which Cricket sends data via HTTP GET requests
RTC (Real-Time Clock)	on/off	Off	Disable the built-in Real-Time Clock. It disables periodical wake ups of your device. However you can still wake it up from WAKE_UP port signal.
		On	Enable the built-in Real-Time Clock. It enables Cricket to wake up automatically with your configured time interval specified by both “unit” and “value” parameters. For example, you can configure Cricket to wake up every 1 hour to evaluate a temperature sensor and send it to a network.
	unit	hours	Select interval time units
		minutes	
		seconds	
	value	<0..59>	Specify time interval value expressed with a selected unit (above). The following range of values is allowed:: Hours: <0..23> Minutes & Seconds: <0..59>
	force-update	Off	Every time Cricket wakes up from the RTC it evaluates if property values have changed since the last evaluation. Only if they changed Cricket connects and sends data. Otherwise it powers off immediately.
		On	Every time Cricket wakes up from the RTC all properties are sent regardless if values changed or not.
BATTERY MONITOR	on/off	Off	Disable the built-in battery voltage monitor. If you are not going to report a battery level for users, it is recommended to keep this option Off.
		On	Enable the built-in battery voltage monitor. Cricket reports the battery voltage level every time it connects. It makes sense to keep this option On only if it is essential for users to monitor the battery level.

	resolution (bits)	<1..8>	Resolution of the battery analog monitor which ranges from 1 to 8 number of bits which results in reporting values in range <0..255> respectively
TEMPERATURE SENSOR	on/off	Off	Disable the built-in temperature sensor. Keep this option Off if you either do not require to report a temperature to device users or you attached an external temperature sensor to Cricket.
		On	Enable the built-in temperature sensor on the Cricket module. Keep this option on if you want to send users an estimated temperature information. Reporting temperature can co-exist with other device functionalities. However if you require accurate temperature measurements you may need to use a dedicated external sensor, which you can attach to Cricket.
	averaging	Off	Switch on / off averaging number of points
		On	
	avg num of points	2, 4, 8, 16	Defines moving average for built-in temperature sensor. The value defines a number of samples.
IO1 (WAKE_UP)	force_update	on	Every time Cricket wakes up from the WAKE_UP pin it evaluates if property values have changed since the last evaluation. Only if they changed Cricket connects to the internet and sends data. Otherwise it powers off immediately.
		off	Every time Cricket wakes up from the WAKE_UP pin all properties are sent to the internet regardless if values changed or not.
IO2	on/off	Off	Disable IO2 port. No value is read during sensory evaluation.
	type	ANALOG_IN	Configure IO2 as analog input
		resolution (bits)	Defines analog input resolution value between 0 and 7
	type	DIGITAL_IN	Configures IO2 port as digital input. Provide only 0 and 1 states
IO3	on/off	Off	Disable IO3 port. No value is read during sensory evaluation.
		On	Configures IO3 port as digital input. Provide only 0 and 1 states
WIFI	enable caching	On	Connectivity optimisations, which allow Cricket to connect to the WiFi faster and send data with much lower latency. It is strongly recommended to keep this option On.
		Off	Connectivity optimisations are off
CONFIG OTA	on/off	Off	Cricket is entirely configured locally and doesn't fetch any configuration from a remote server.

		On	Cricket is configured from a remote server (http://cota.thingsonedge.com) Press the built-in button on the Cricket board for ~1 second and it will fetch the config from the server. Please make sure your Cricket is connected to WiFi with internet access.
	interval	0 .. 10000	Set how often Cricket retrieves configuration from the remote server. The counter is based on the number of wake ups of Cricket. Regardless if it was connected to the internet or not. 0 - Never 1 - Every time when Cricket wakes up 2 - Every second wake up 3 - Every third wake up and so on In every **N**th wake up Cricket fetches config from http://cota.thingsonedge.com microservice

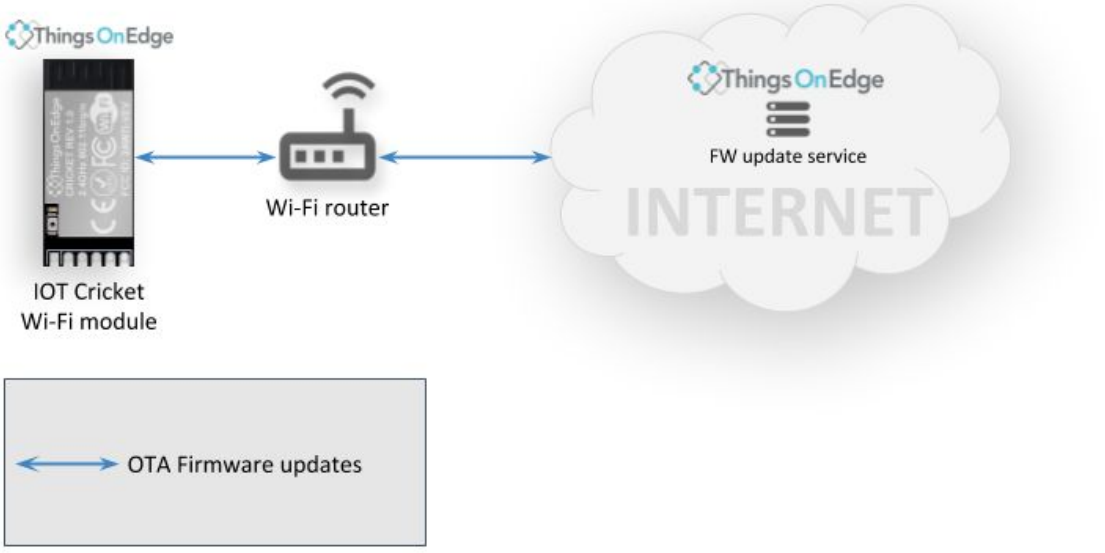
INFO

The INFO panel shows the current Wi-Fi connectivity status among other important information such as the **serial number** and **password** of the device. Please do not share them with anyone. These credentials are used for the following purposes:

- Access to the remote configuration http://cota.thingsonedge.com microservice
- Serial number for MQTT topics

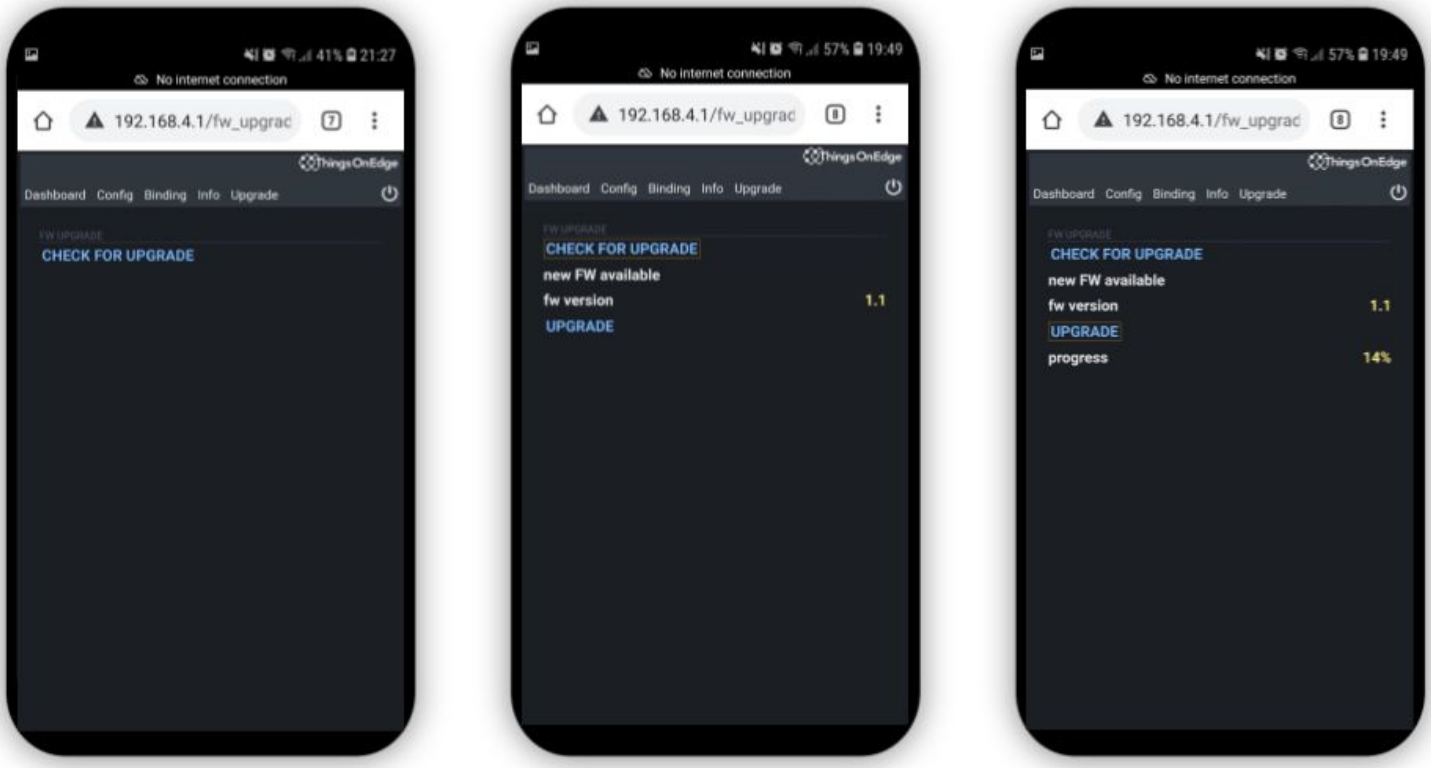
UPGRADE

Cricket also comes with OTA firmware upgrades.



In order to upgrade a firmware, open Cricket Configuration Panel, then go to the Upgrade tab and press "**CHECK FOR UPGRADE**".

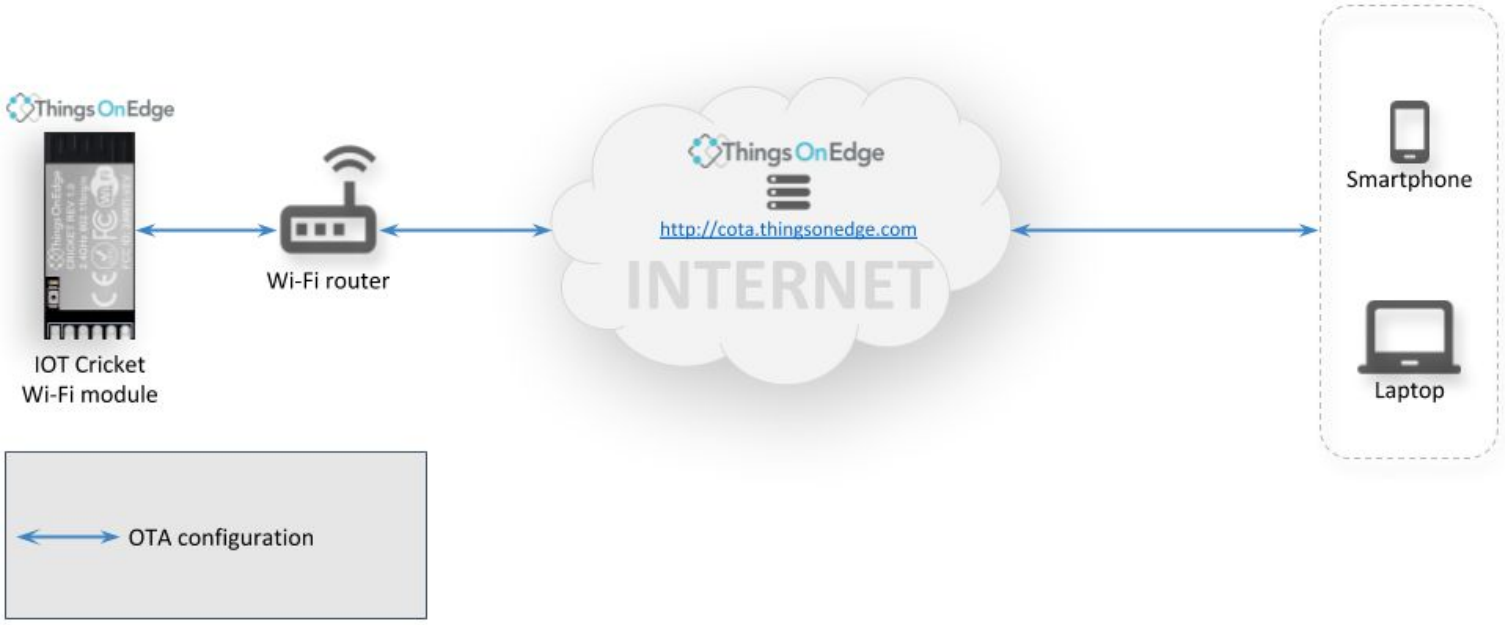
If a new firmware is available and you wish to continue, press "**UPGRADE**". It will start downloading the firmware and then install it automatically. Please make sure Cricket is connected to WiFi which has the internet access. It needs to connect to Things On Edge microservice to download the firmware.



After the upgrade process is finished the Cricket is powered off. If you want to check if it was successful you need to press the Cricket's button for ~5 sec and then open Cricket Configuration Panel again. Then go to the INFO tab and check the FW version.

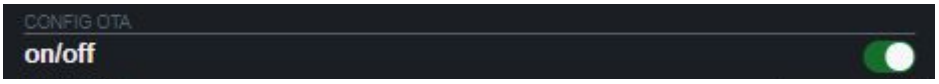
Remote configuration

You can also configure Cricket from a remote http://cota.thingsonedge.com microservice. It is useful especially when you have many devices and you would like to configure all of them remotely instead of doing it one by one for each device locally.



NOTE: in order to make Cricket fetching a remote configuration it MUST:

1. Be connected to Wi-Fi network with the internet access (see the BINDING section)
2. Be configured to fetch configuration from a remote server (see the CONFIG section)



The UI config panel is exactly the same for both remote and local configurations (see the CONFIG section). Below we show in a few steps how you can configure Cricket from a remote service:

Open remote configuration micro-service: http://cota.thingsonedge.com

Cricket's Configuration Panel

Type SN & PWD from your Cricket

Set parameters (configuration is autosaved)

When the configuration is saved
&
Cricket hasn't fetched it yet

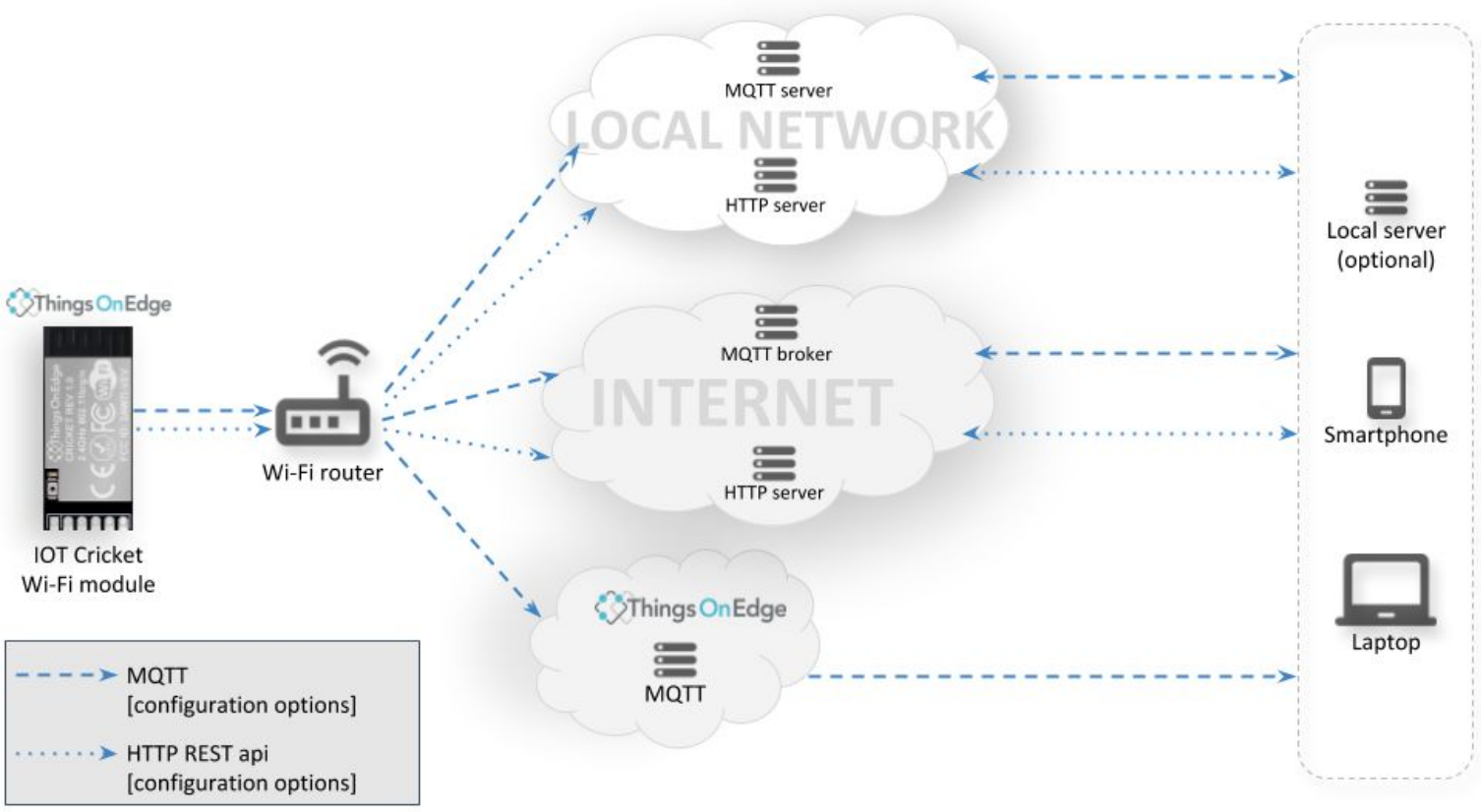
Communication APIs

This section provides details on how you can receive data and events from Cricket on client devices such as phones, tablets, PCs, or any other services.

You can choose one of the following communication channels:

> **MQTT** - one to many low latency communication channel(s)
> **HTTP POST/GET request** (a.k.a. Webhooks)

A high level overview of how Cricket integrates and sends data is shown below on the diagram.



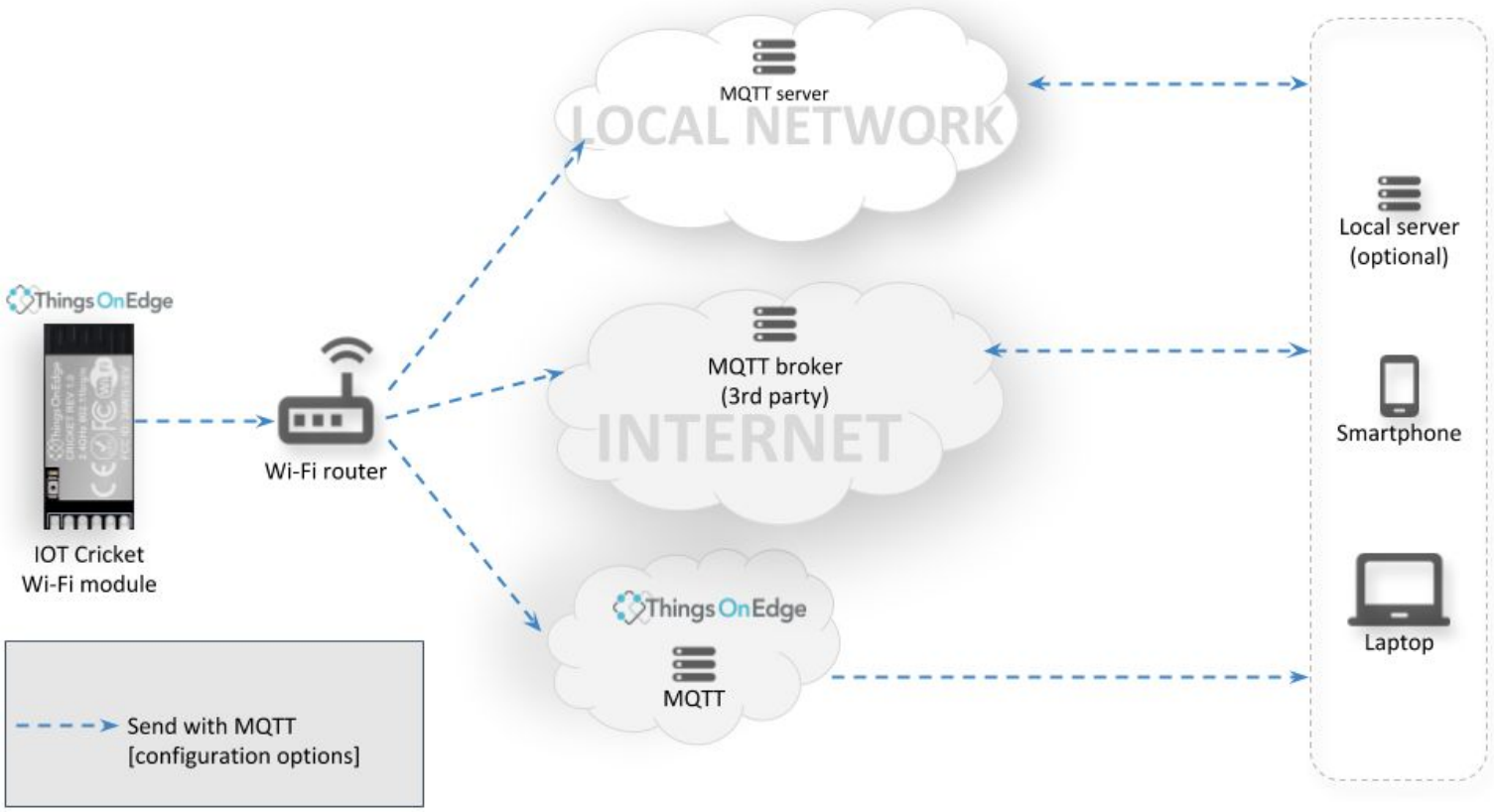
MQTT

“MQTT is an open OASIS and ISO standard (ISO/IEC PRF 20922) lightweight, publish-subscribe network protocol that transports messages between devices” (source Wikipedia)

The Cricket module has a built-in native MQTT support. It provides a direct, reliable and low latency connectivity which results in ultra-low power consumption. You have the following options to configure MQTT:

Local MQTT server - many hobbyists and DIYer prefer to manage all the data locally including MQTT servers (e.g. RabbitMQ). You can configure Cricket to use MQTT channel directly to your local MQTT server.

Internet MQTT broker - you can configure other 3rd party MQTT brokers to which Cricket directly sends data and events. Things On Edge also provides an optional free TOE MQTT broker to make it easier for you to set it up. Especially when you just start playing with MQTT and Cricket.



MQTT allows you to use Cricket to propagate data to many end-users directly on their devices (Smartphone, Laptops, …) by using off the shelf wide range of mobile apps and internet services.

Cricket sends data via MQTT broker, which you configured. You can connect and subscribe from any client device (smartphone, laptop, PC, …) directly to this service. Every Cricket comes with a unique serial number. Each topic also includes the serial number which makes all topics unique. In other words the data is sent on separate MQTT topics for each Cricket. This means that you do not require to use advanced filters to extract data from messages. They can be simply read as raw values. Below you can find more information and examples on what MQTT topics you can subscribe from client devices.

If you choose to use the TOE MQTT broker the unique serial number of Cricket is also used for authentication credentials i.e. username & password so you do not need to configure it explicitly.

MQTT topics

Cricket publishes (sends out) data on topics listed below. In order to receive data on your client devices you need to subscribe to these topics by using the following topic format:

/ SERIAL / PARAM

Where:

SERIAL: Cricket's unique serial number

PARAM: is a module property/sensor type (see the table below)

PARAM	Format	Unit	Range	eg.	Description

temp	dd.d	Celsuis	−40°C to +80°C	12.5	Temperature value from built-in Cricket's temperature sensor
batt	ddd	Decimal	0 to 255	124	Current battery voltage level represented as 8bits decimal value
io2	ddd	Decimal	0 to 255	100	Current level on the IO2 port. If you choose IO2 as digital than the value is either 0 or 1
io3	d	Decimal	0 or 1	0	Current value either 0 or 1 on the IO3 port
io1_wake_up	d	Decimal	0 or 1	1	The payload value of that topic is 1 if the module was woken up by a WAKE_UP pin otherwise is 0. This message is only published if either WAKE_UP is triggered or "Force Updates" > “IO1 Wake Up” is enabled in the module configuration.
rtc_wake_up	d	Decimal	0 or 1	1	The payload value of that topic is 1 if the module was woken up by internal RTC otherwise is 0. This message is only published if either RTC wake-up is triggered or "Force Updates” on “RTC Wake Up” is enabled in the module configuration.
hwc_wake_up	ddd	Decimal	0 to 2147483647	11	This is an internal counter representing a number of wake ups of Cricket
hwc_wifi_enabled	ddd	Decimal	0 to 2147483647	12	This is an internal counter representing a number of times Cricket connected to WiFi
device_sn	string	Text	ASCII	"foo"	Cricket's serial number, alphanumeric string of 10 characters
device_name	string	Text	ASCII	"TOE"	Your custom name of the device

d - a single decimal digit

Example of subscribe topics for Cricket with 1234566789 serial number:

/1234566789/temp the subscribe topic to monitor a temperature from the “1234566789” device

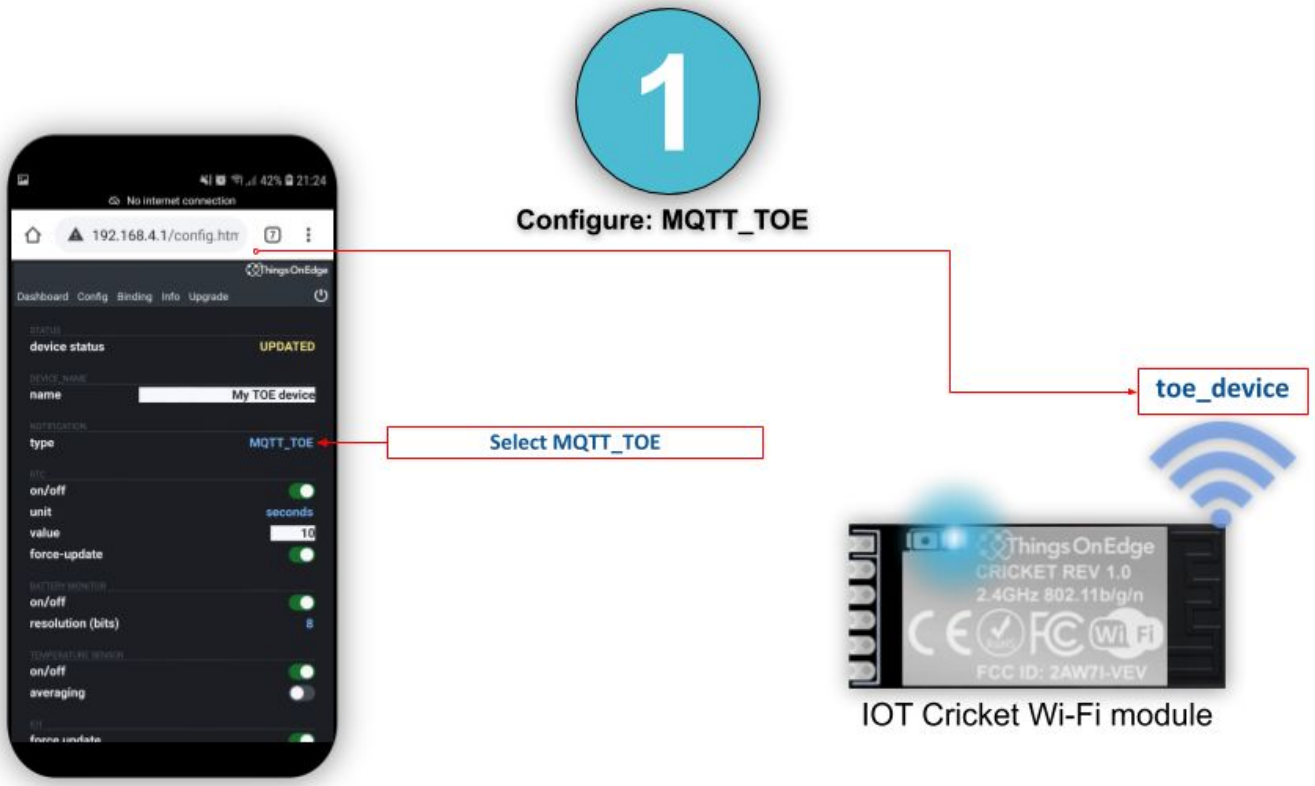
/1234566789/batt the subscribe topic to monitor a battery level of the “1234566789” device

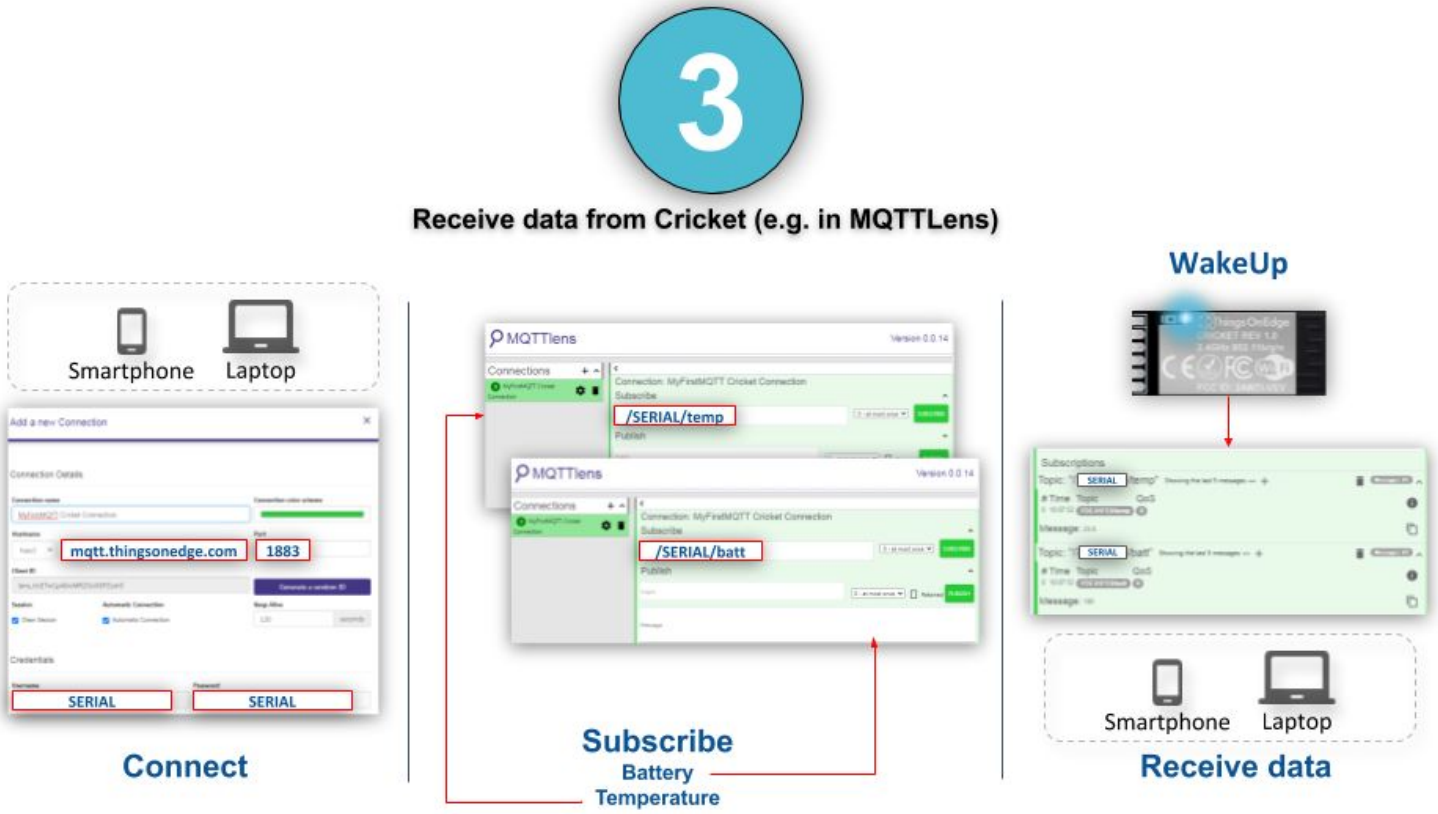
MQTT client configuration

In order to receive data on your client devices (phone, PC, laptop, server, …) you need to connect to the MQTT broker. It must be consistent with the broker you also configure Cricket. If you want to use a custom MQTT broker, see the Custom MQTT broker section. If you want to use the TOE MQTT broker, which doesn't require any extra account and works out of the box, go to the TOE MQTT section.

TOE MQTT

Things On Edge provides an optional free of charge MQTT broker. It is to help you get started as in a few steps you can receive data from Cricket on your client devices (phone, laptop, PC, server, …)

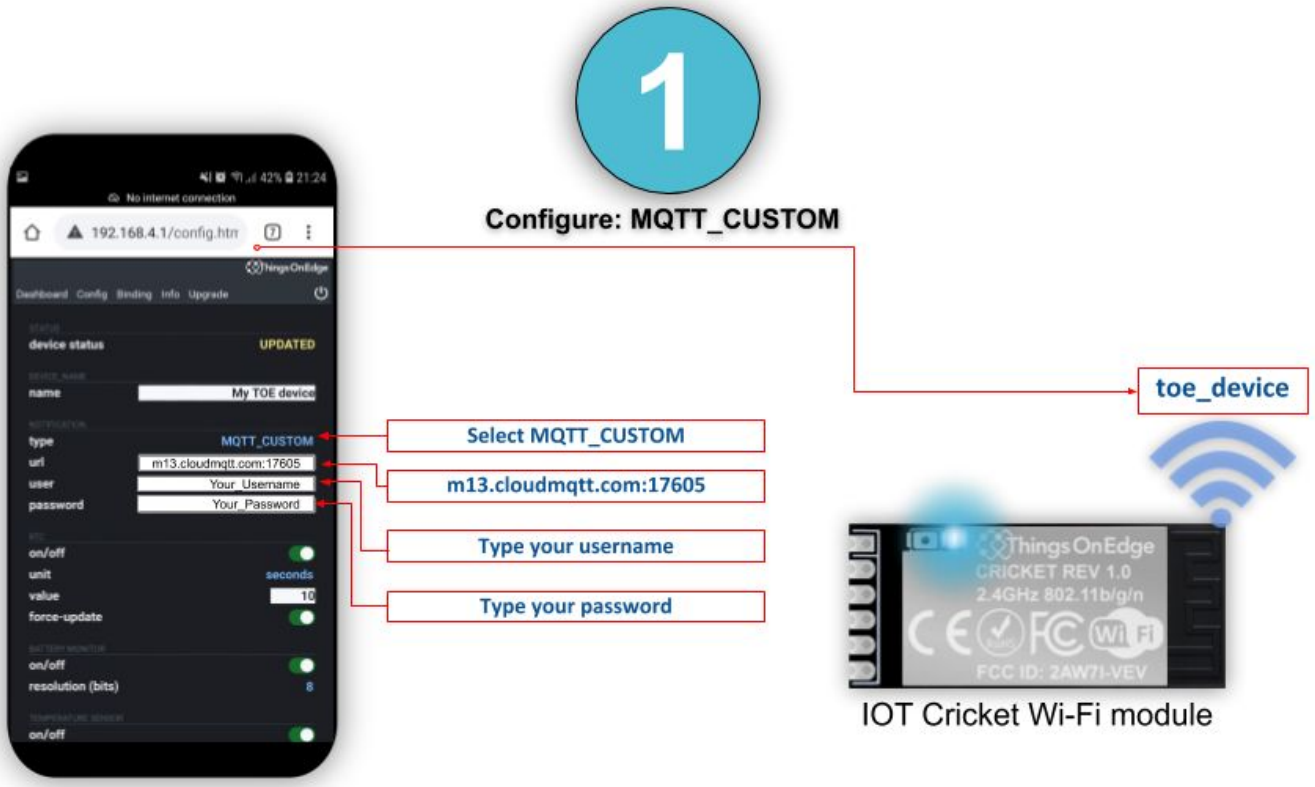


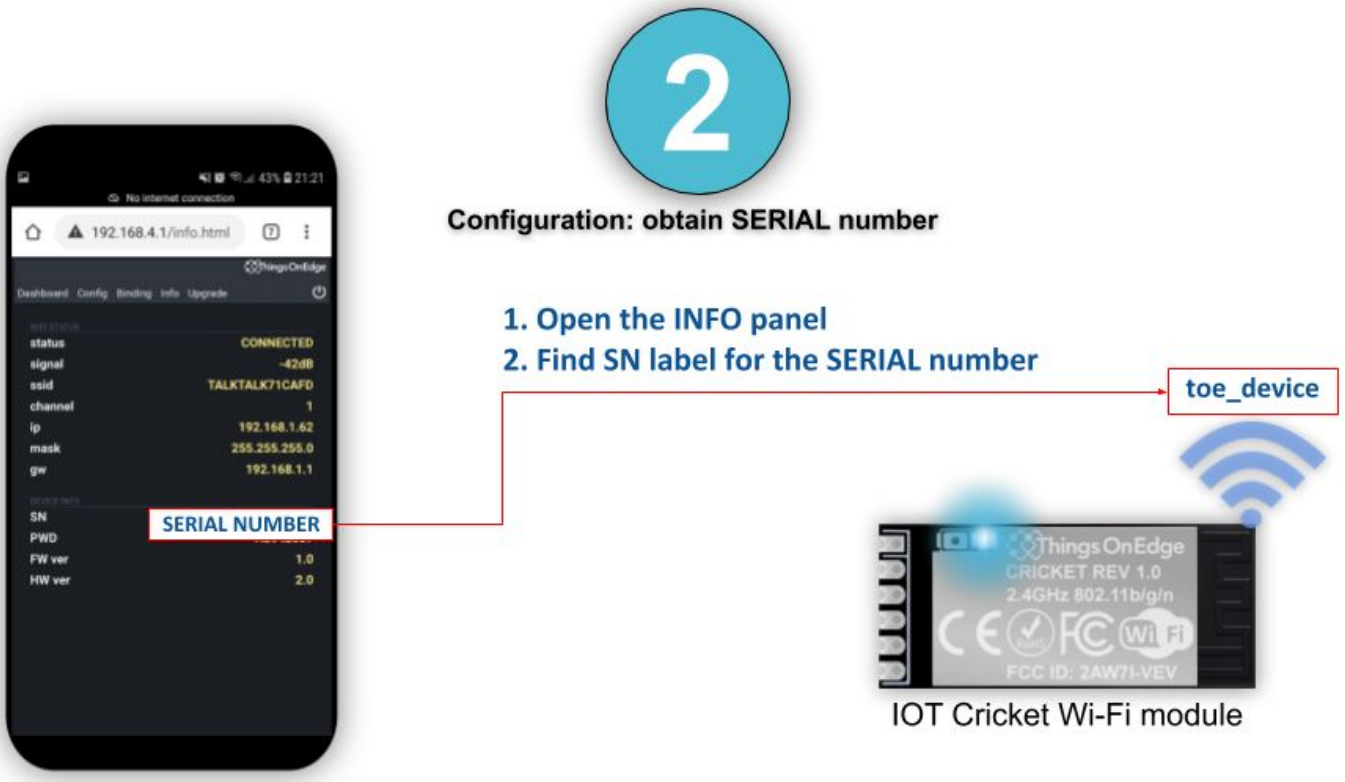


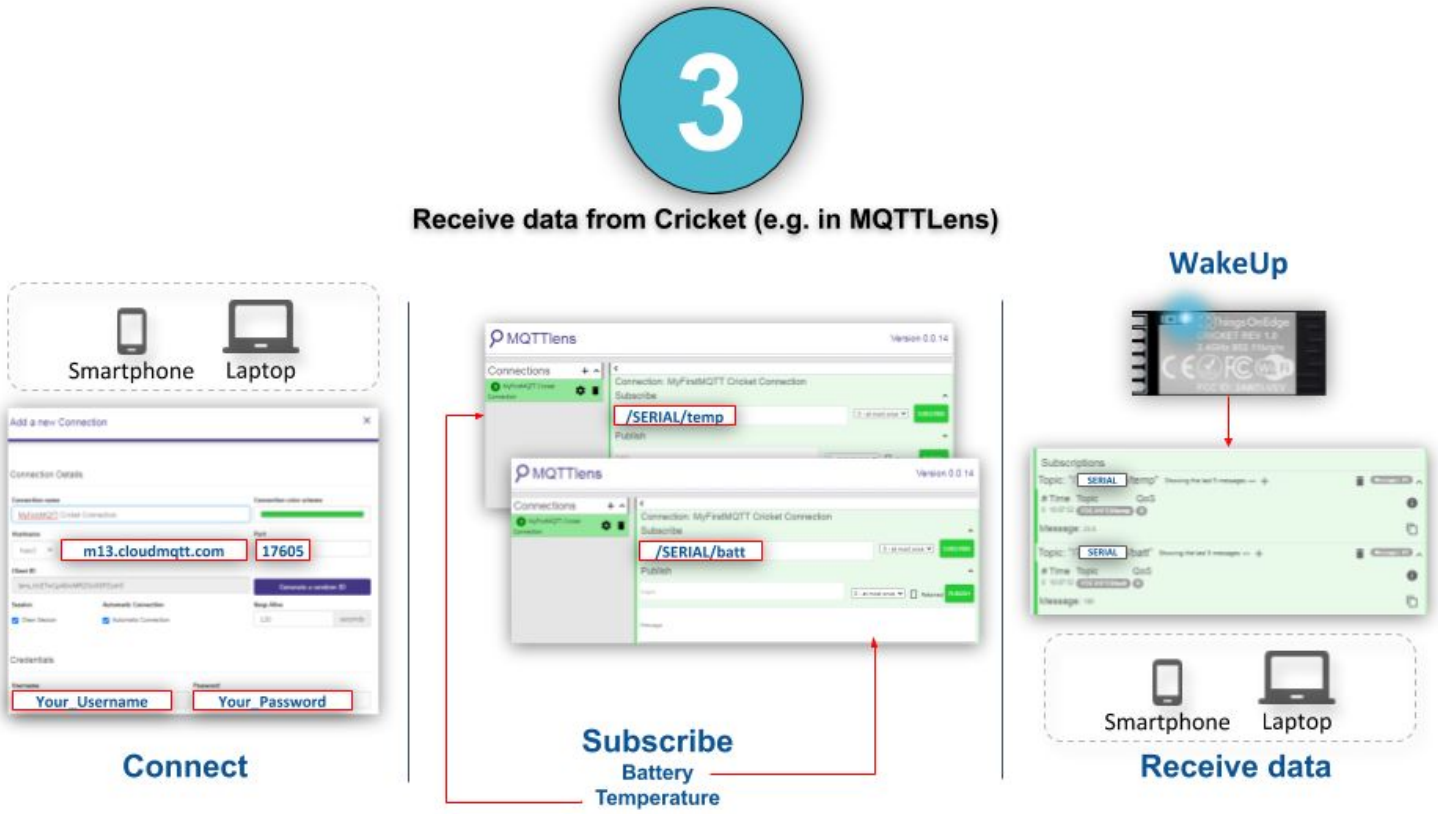
Now can subscribe to more topics than just a temperature or a battery (see the MQTT topics section).

Custom MQTT broker

You can configure Cricket to use other 3rd party MQTT brokers. All you need is a server address, port and user credentials (user & password). You can do this in the Cricket's configuration (see Configuration section). Then your client devices must be configured with the associated credentials. Here is an example of Cricket sending messages directly to i.e. **m13.cloudmqtt.com** address on **17605** port with a custom username and password; and MQTTLens as a client receiving the data.







Now can subscribe to more topics than just a temperature and a battery. See the MQTT topics section for more topics that you can subscribe to.

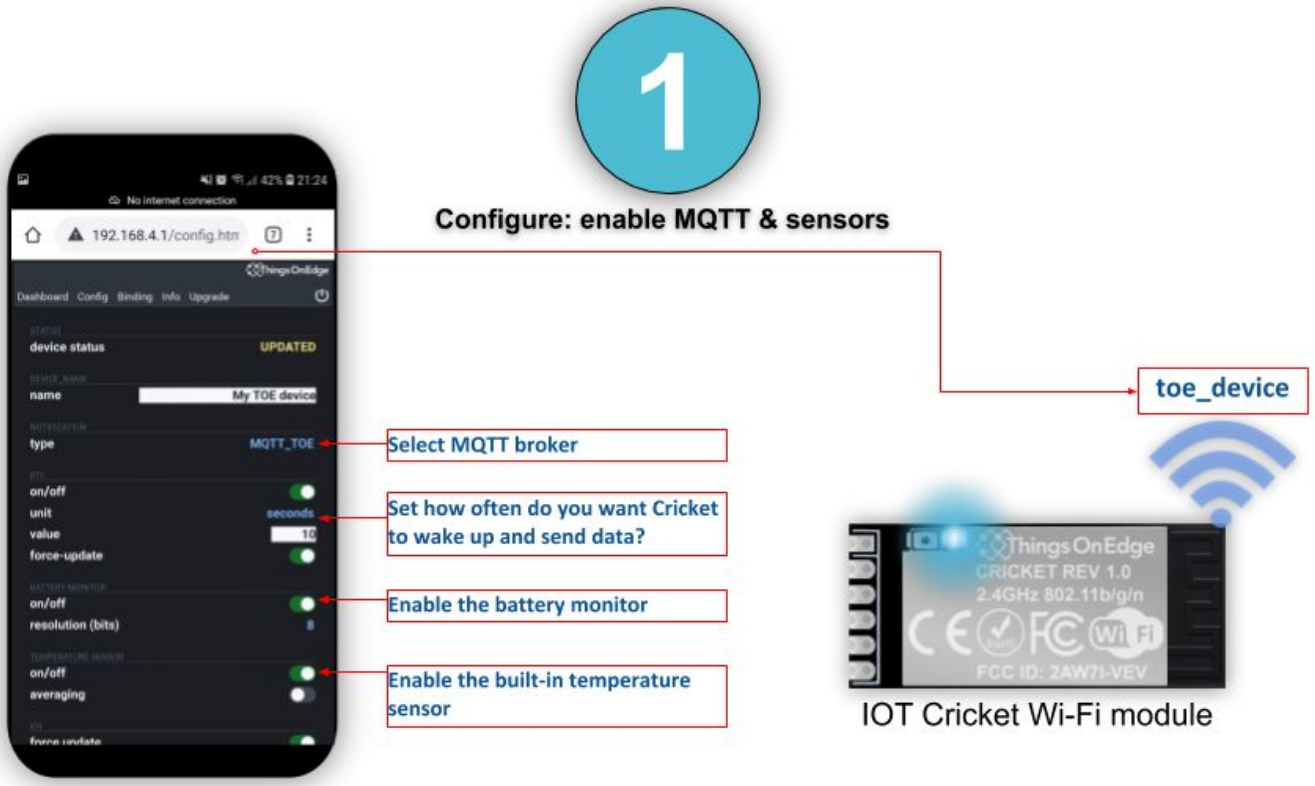
MQTT client examples

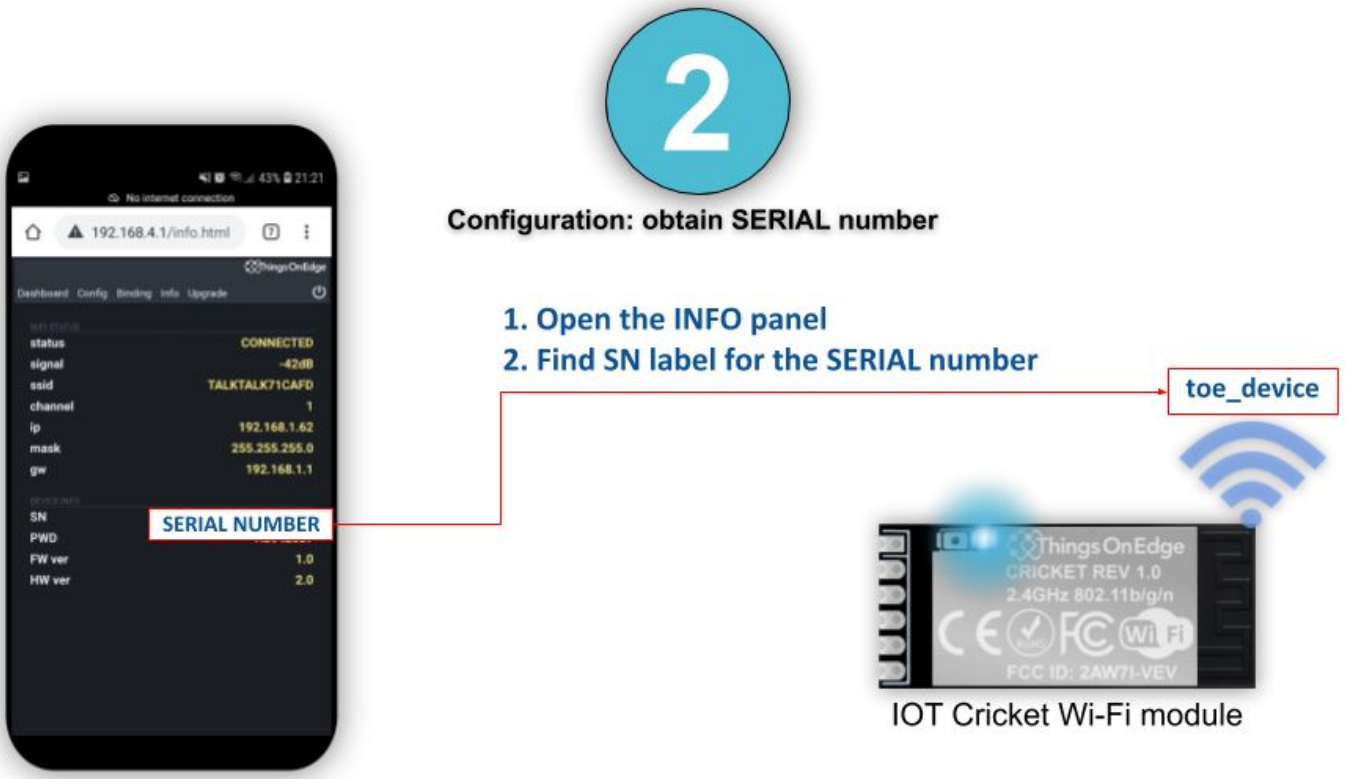
In this section you can find a few examples on how to receive data from Cricket on your client devices (PC, laptop, smartphone, …).

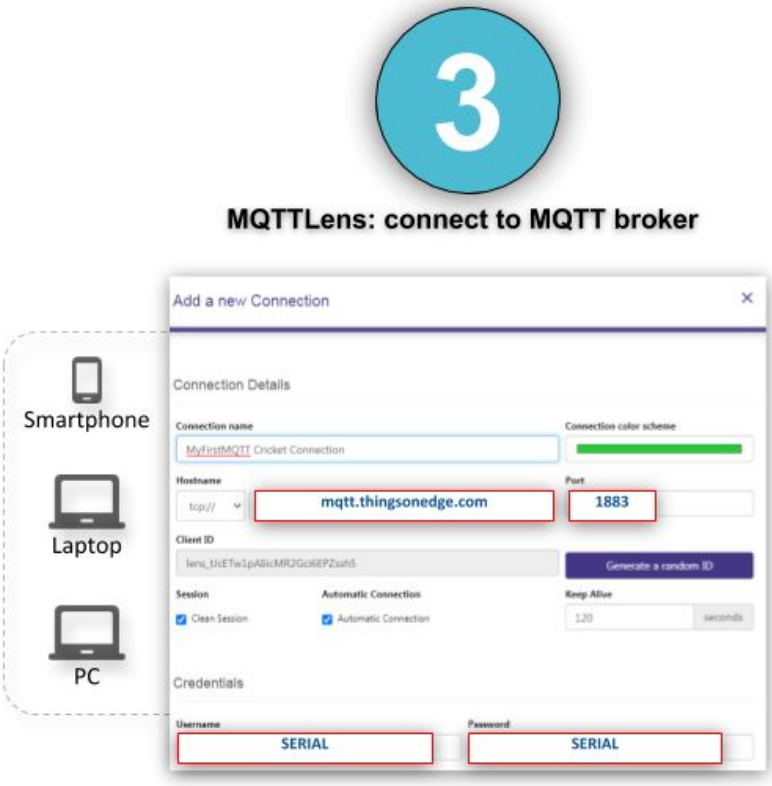
To keep it simple let's configure Cricket (see CONFIG) to use MQTT_TOE. Also make sure you enabled Cricket to send data which you want to receive such as sensors, battery, temperature, etc. Please note Cricket doesn't always send data if values remain unchanged. You can enable "force updates" to force Cricket to send data every time when it wakes up. This is just to make sure your setup works. Later you can switch off the force updates as it saves a battery.

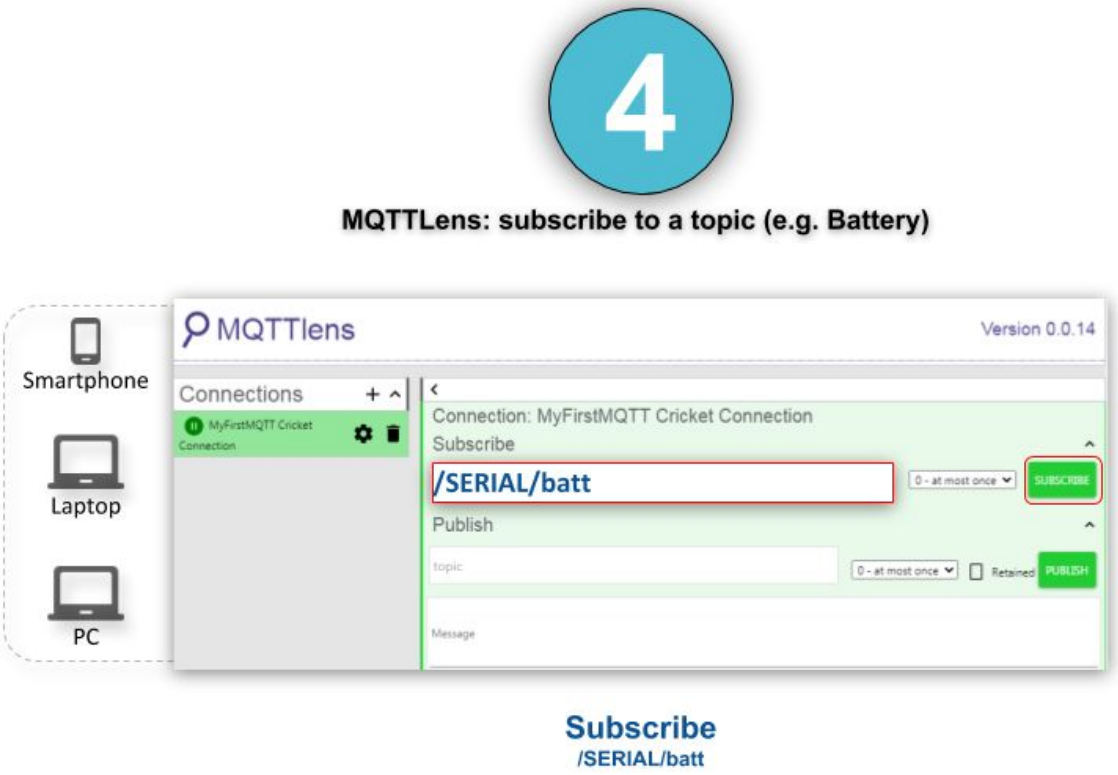
MQTTLens

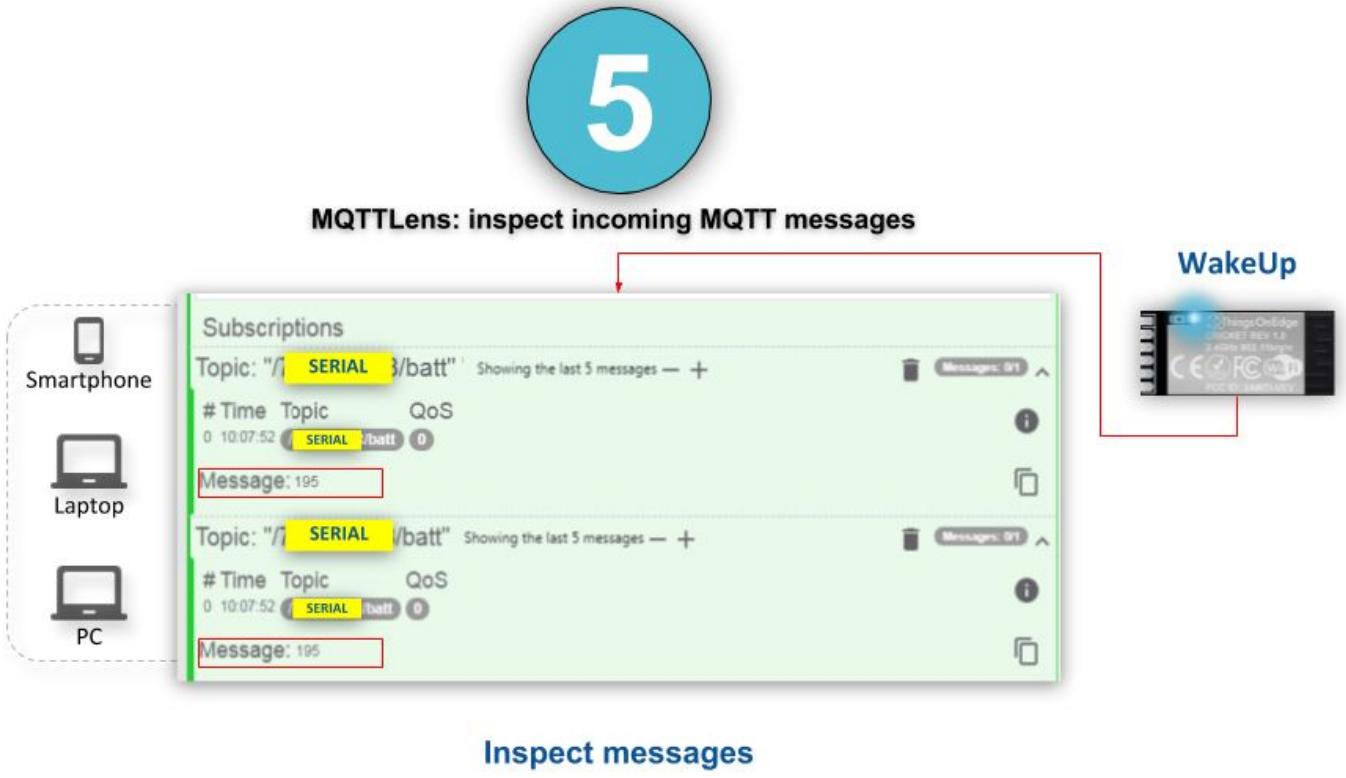
For inspecting MQTT messages one of the tools you might want to use is MQTTLens (it's an extension for a Google Chrome browser). This is a very easy to use tool and you can start receiving data from your Cricket in a few simple steps shown below.




2
Configuration: obtain SERIAL number
1. Open the INFO panel
2. Find SN label for the SERIAL number
192.168.4.1/info.html
status CONNECTED
signal -42dB
ssid TALKTALK71CAFD
channel 1
ip 192.168.1.62
mask 255.255.255.0
gw 192.168.1.1
SN
PWD
FW ver 1.0
HW ver 2.0
SERIAL NUMBER
toe_device
ThingsOnEdge
CRICKET REV 1.0
2.4GHz 802.11b/g/n
FCC ID: 2AW7I-VEV
IOT Cricket Wi-Fi module

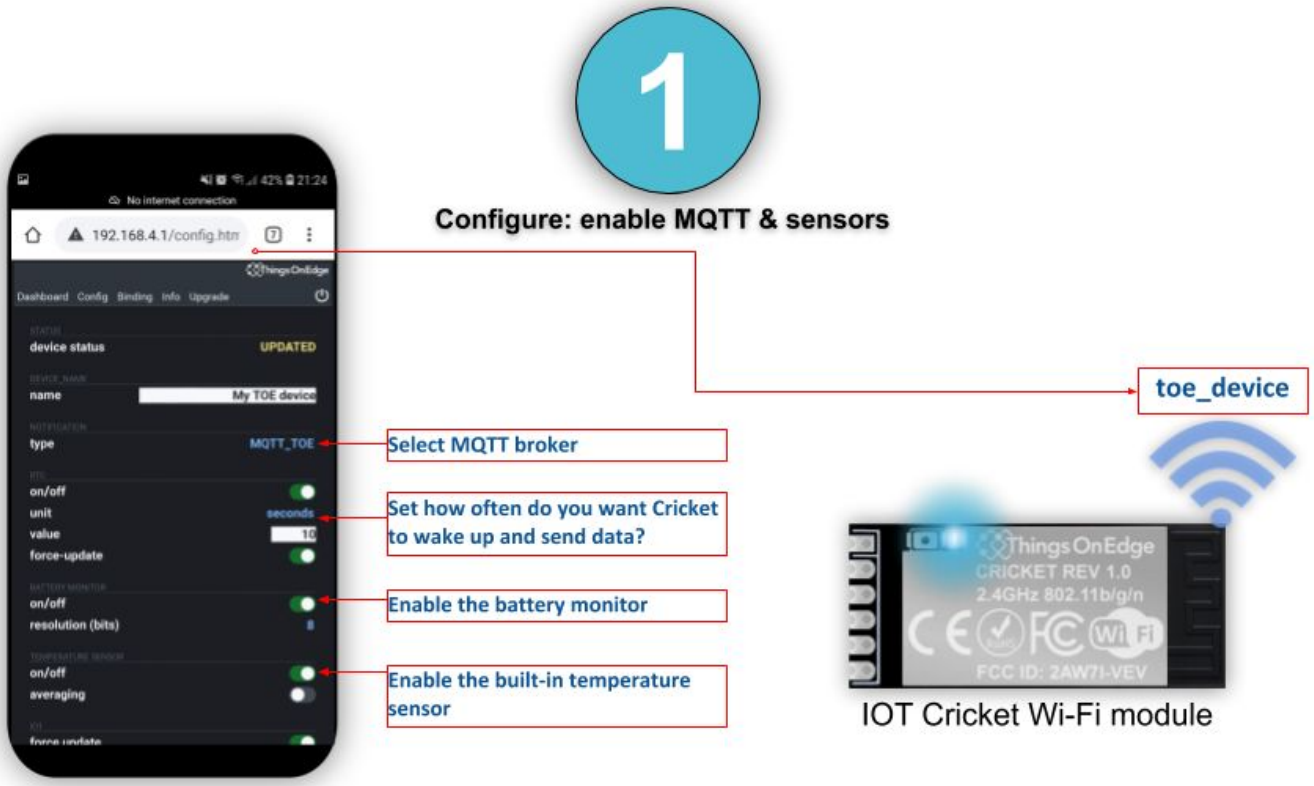

3
MQTTLens: connect to MQTT broker
Smartphone
Laptop
PC
Add a new Connection
Connection Details
Connection name
MyFirstMQTT Cricket Connection
Connection color scheme
Hostname
tcp://
mqtt.thingsonedge.com
Port
1883
Client ID
Generate a random ID
Session
Clean Session
Automatic Connection
Automatic Connection
Keep Alive
120
seconds
Credentials
Username
SERIAL
Password
SERIAL

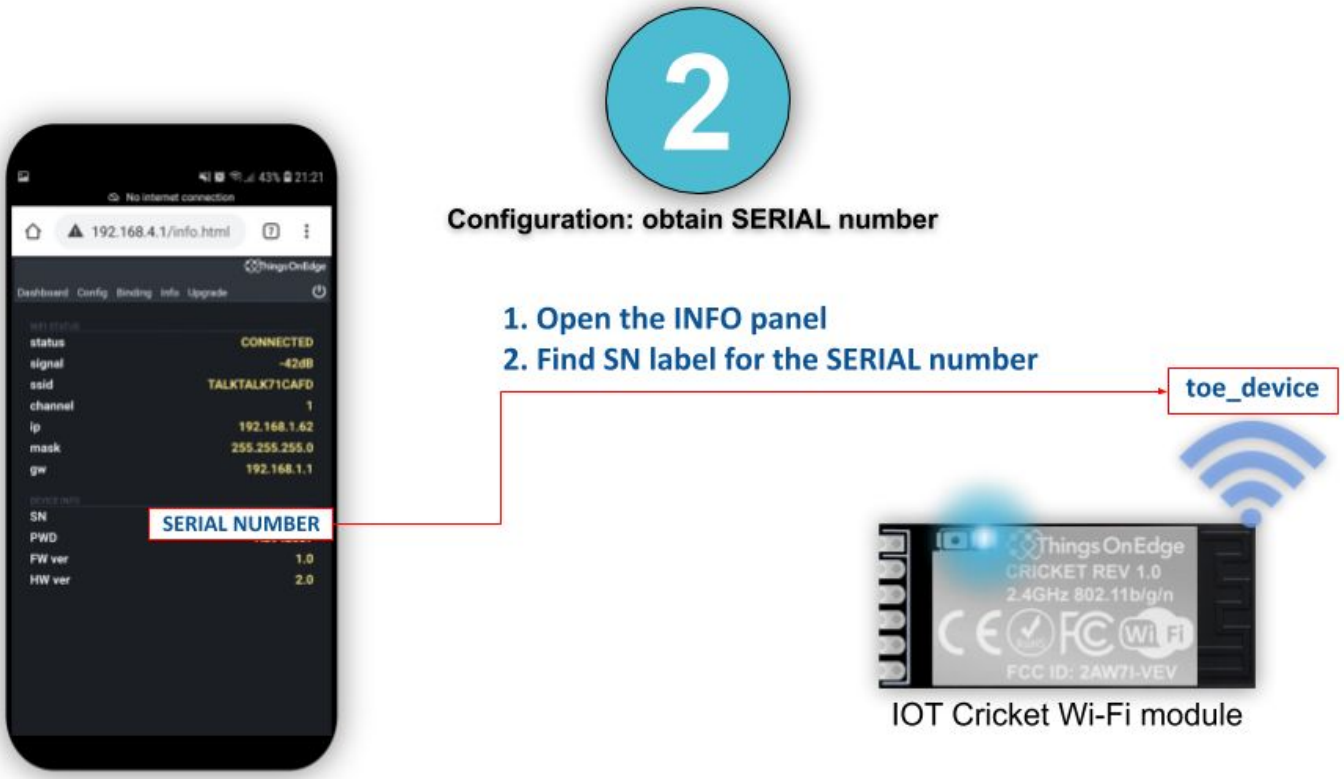




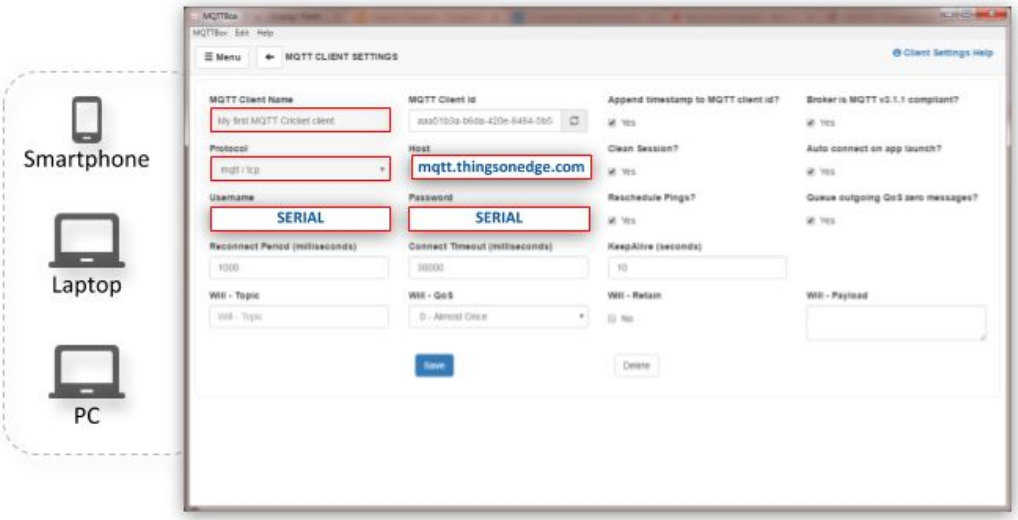
MQTT Box (PC, Windows, Mac, Linux)

MQTT Box is a multi-platform MQTT client to receive and inspect messages. See the steps below how to configure this app to work with Cricket. You can use any MQTT broker and in this example we use TOE MQTT broker to keep it simple.

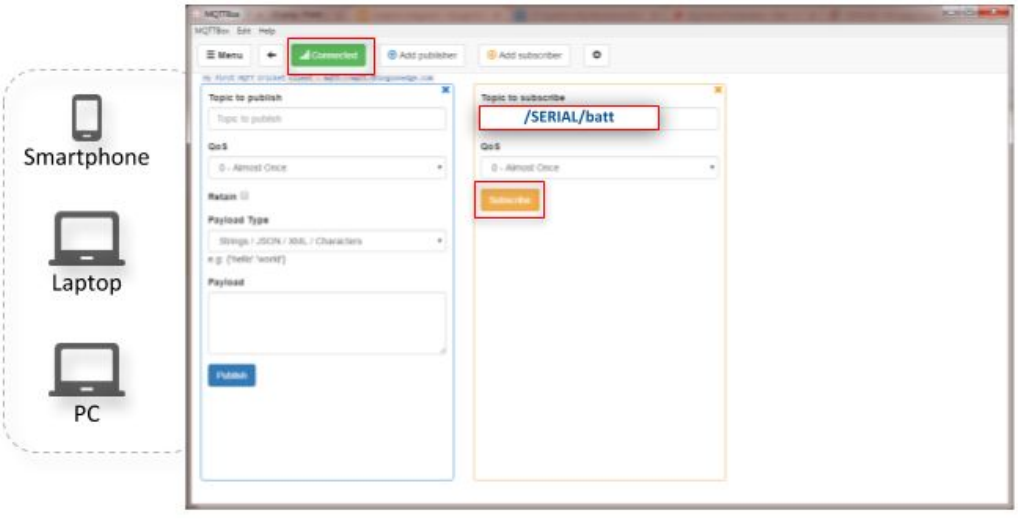

1
Configure: enable MQTT & sensors
192.168.4.1/config.htm
Dashboard Config Binding Info Upgrade
device status
UPDATED
name
My TOE device
type
MQTT_TOE
on/off
unit
seconds
value
10
force-update
on/off
resolution (bits)
on/off
averaging
Select MQTT broker
Set how often do you want Cricket to wake up and send data?
Enable the battery monitor
Enable the built-in temperature sensor
toe_device
ThingsOnEdge
CRICKET REV 1.0
2.4GHz 802.11b/g/n
FCC ID: 2AW7I-VEV
IOT Cricket Wi-Fi module

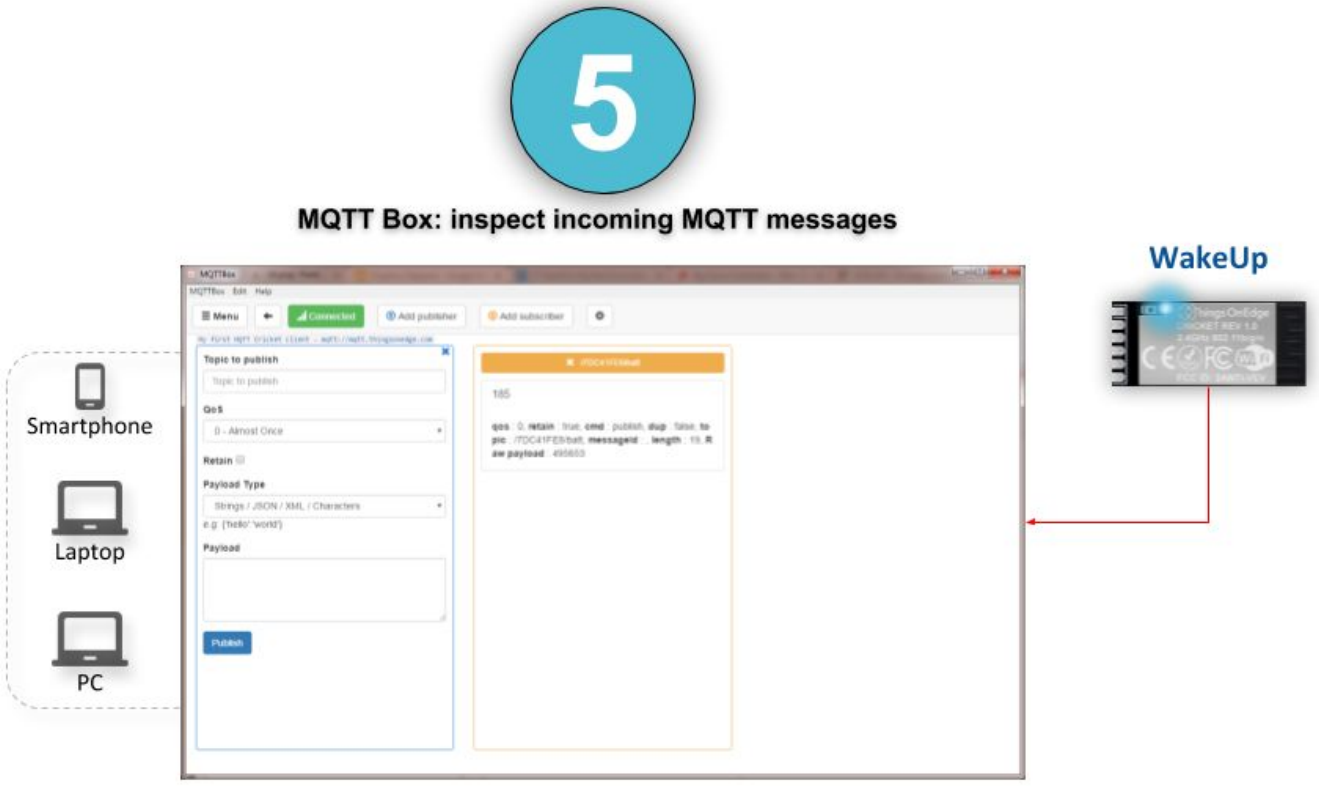

2
Configuration: obtain SERIAL number
1. Open the INFO panel
2. Find SN label for the SERIAL number
192.168.4.1/info.html
Dashboard Config Binding Info Upgrade
status
CONNECTED
signal
-42dB
ssid
TALKTALK71CAFD
channel
1
ip
192.168.1.62
mask
255.255.255.0
gw
192.168.1.1
SN
PWD
FW ver
1.0
HW ver
2.0
SERIAL NUMBER
toe_device
ThingsOnEdge
CRICKET REV 1.0
2.4GHz 802.11b/g/n
FCC ID: 2AW7I-VEV
IOT Cricket Wi-Fi module

MQTT Box: connect to MQTT broker



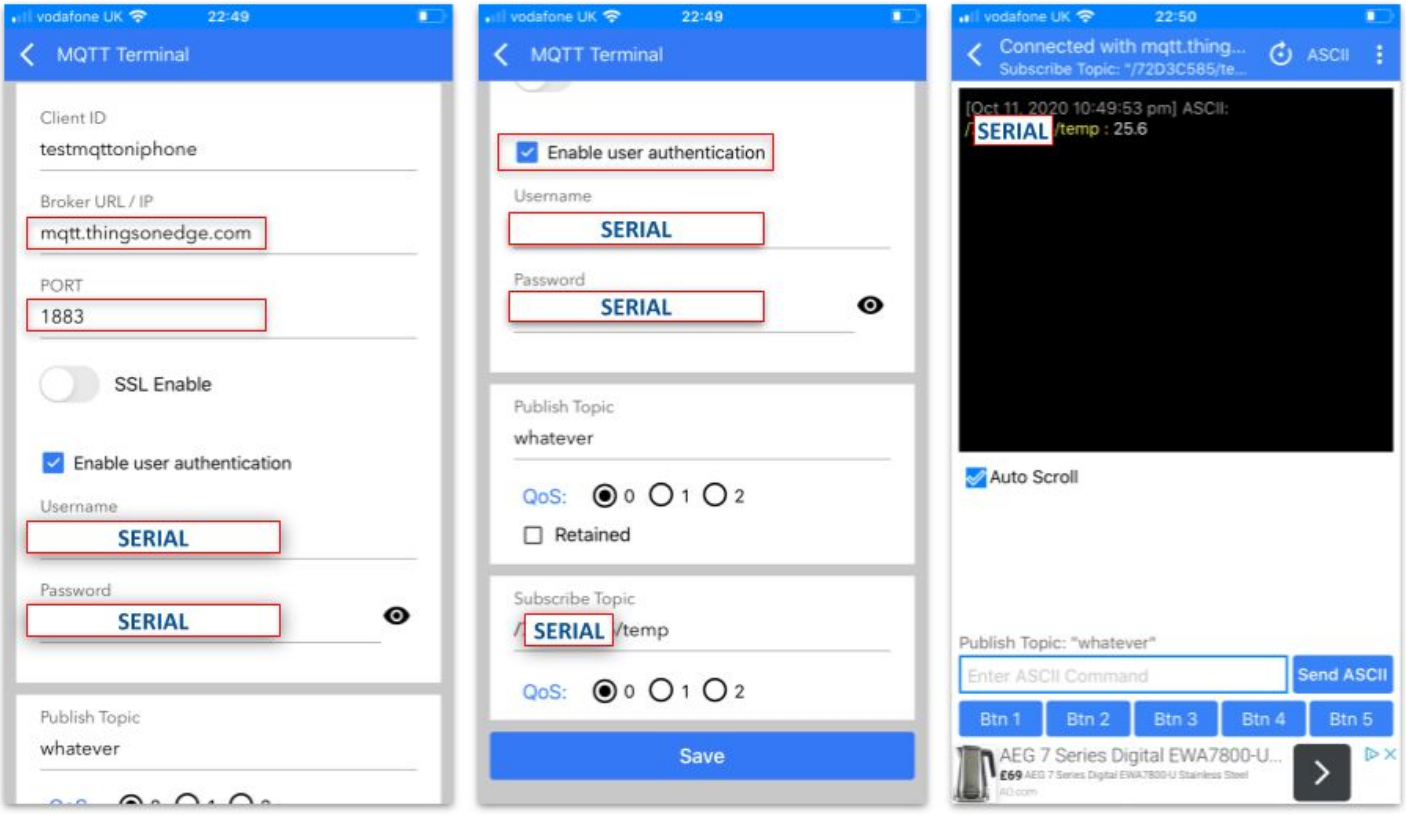
MQTT Box: subscribe to a topic (e.g. Battery)





MQTTTerminal (iPhone)

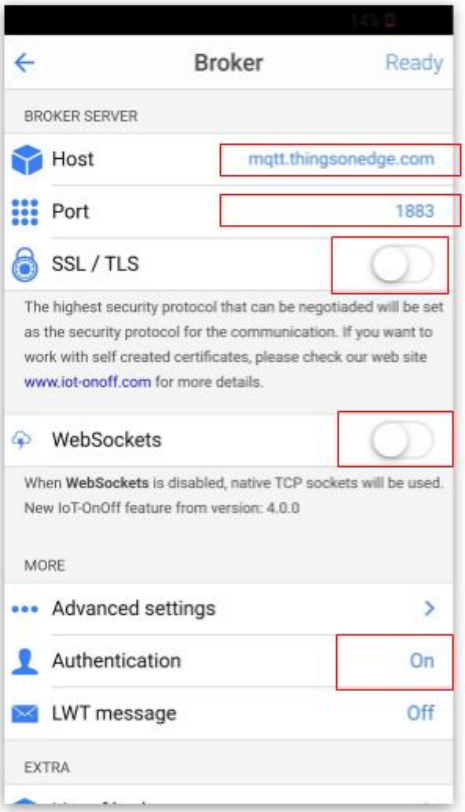
MQTTTerminal is an iPhone app which is fairly easy to use for inspecting MQTT messages. Please see below a few essential screenshots showing how to configure the app to receive data from your Cricket.

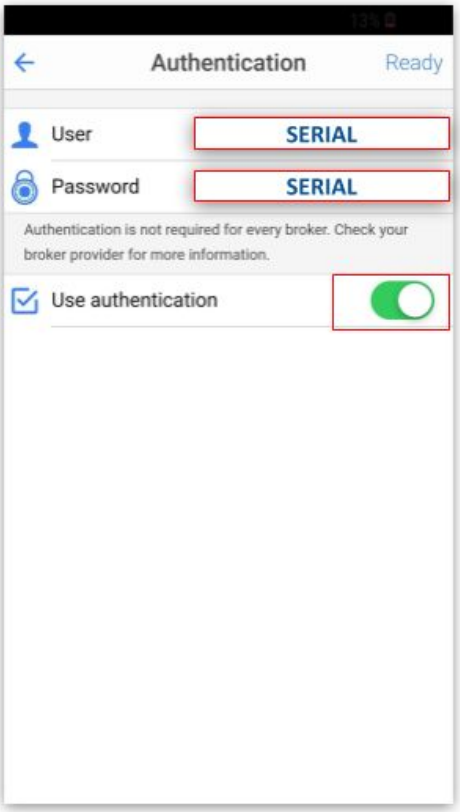


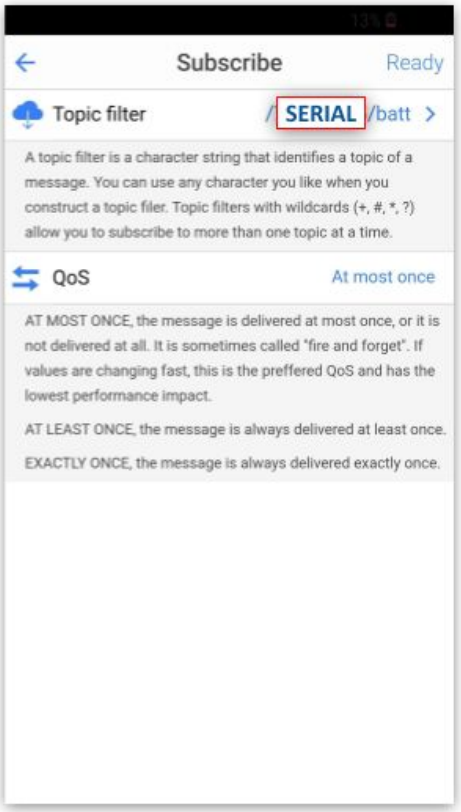
IoT OnOff (iPhone & Android app)

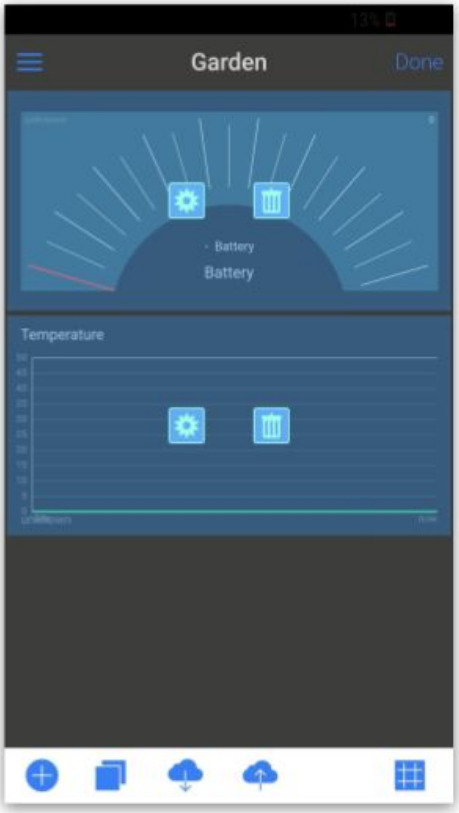
You can also try using IoTOnOff as the app provides pretty graphs and widgets so you can build a very nice dashboard for your Cricket. However please be patient as it may take a while to do what you want but

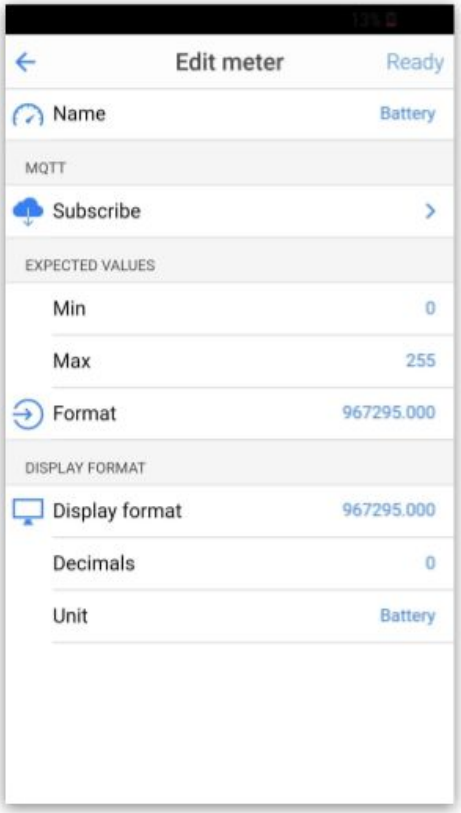
eventually you will get there and it might be worth it. The app has some issues and you may need to try to workaround it. For example you may need to switch on/off the app every time when you change settings. In particular settings related to the connectivity and topics subscriptions. For example when you get a connection error just KILL the app and launch it again. If it still cannot connect then you might want to revisit the settings.
The best way to start playing with the app is to clean all dashboards and start with the empty one. It will likely reduce connectivity errors.
Below you can find some key app screenshots which should help you set the right parameters.

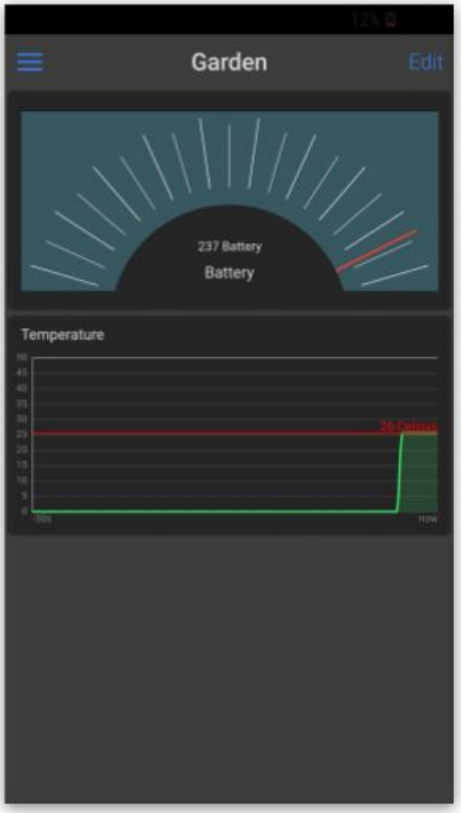












IoT MQTT Panel (Android only)

The IoT MQTT Panel app is another example to demonstrate how to receive and visualise data from Cricket. You can use any MQTT broker, in this example we use TOE MQTT broker to make it a bit easier for you.

1. Download and install IoT MQTT Panel
2. Configure a server connection with the following details:
   Server / Broker IP address: **mqtt.thingsonedge.com**
   Port number: **1883**
   Network protocol: **TCP**

3. Add device e.g. "**Temperature**"

4. Goto advanced options:
   Username: ***your_cricket_serial_number***
   Password: ***your_cricket_serial_number***
   Connect automatically: **YES**

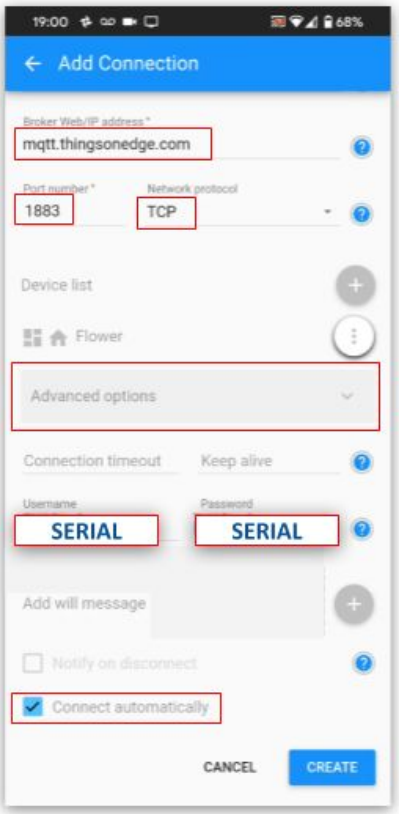
5. Press the **Create** button

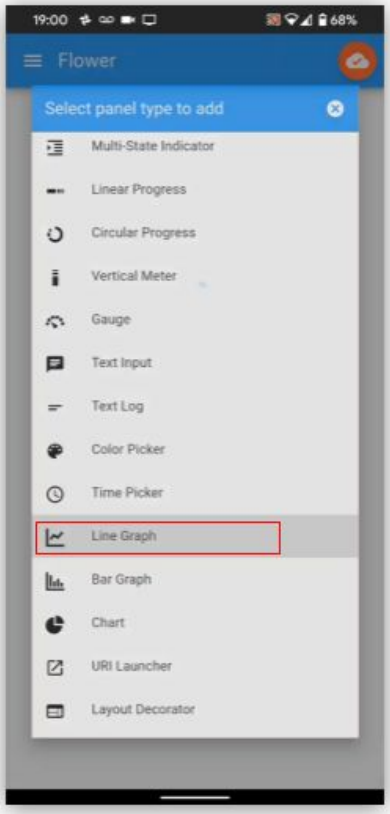
6. Press **ADD PANEL**

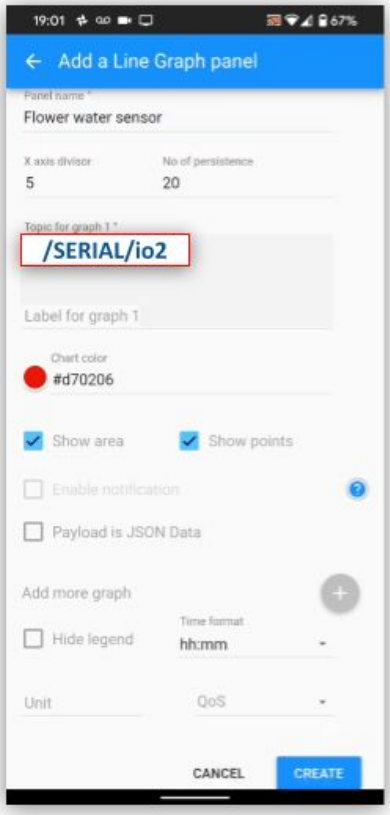
7. Select: **Line Graph**

8. Set the details for graph 1 to read data from sensor (from the Cricket IO2 port)
   Panel name: e.g. **Temperature**
   Topic for graph 1: ***/your_cricket_serial_number/*temp**
   Show area: **YES**
   Show points: **YES**

For more information please refer to the attached screenshots from IoT MQTT Panel below.





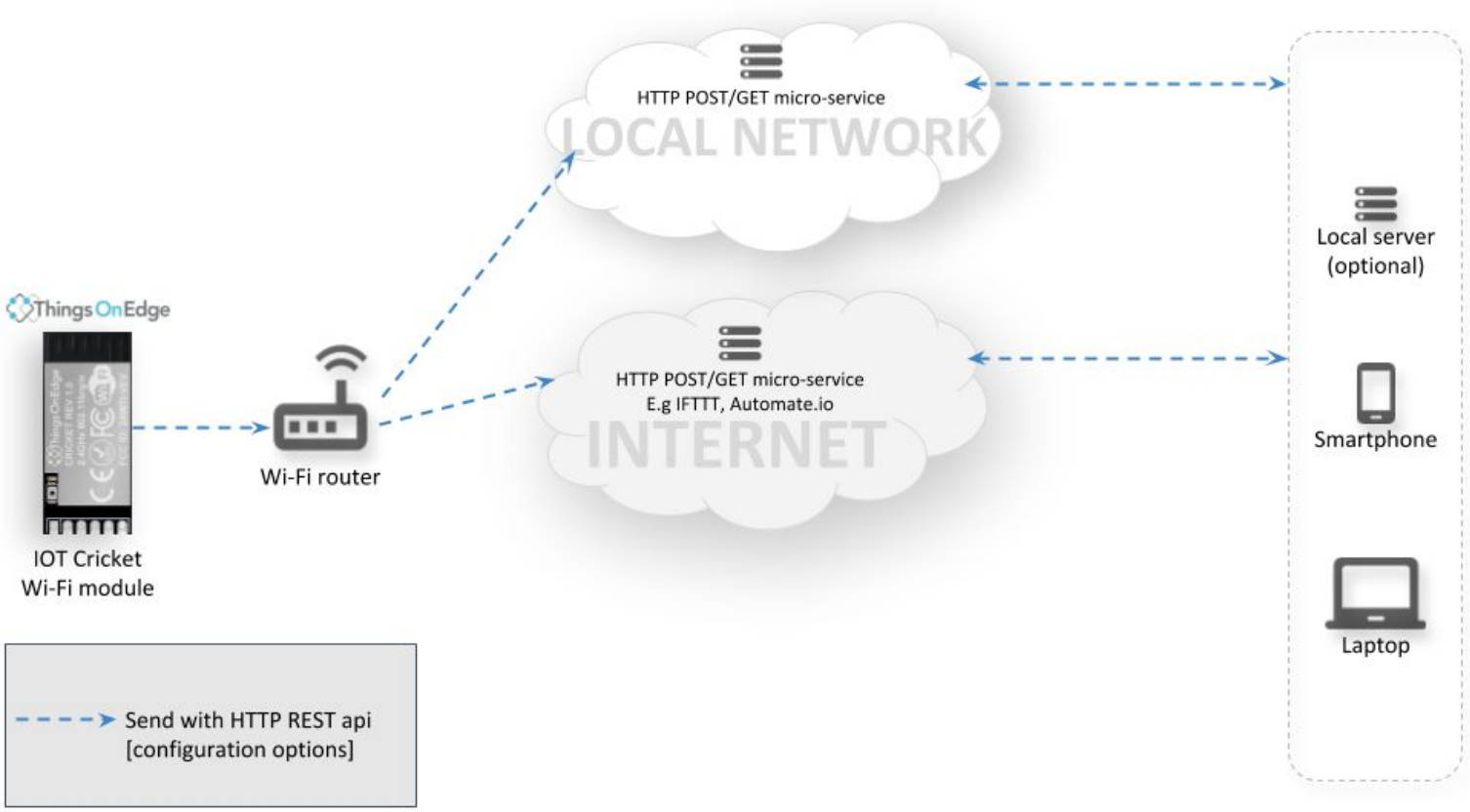


HTTP

"The Hypertext Transfer Protocol (HTTP) is an application layer protocol for distributed, collaborative, hypermedia information systems" (Wikipedia).

Cricket comes with a flexible integration to 3rd party microservices by using HTTP POST & HTTP GET requests (also known as Webhooks). Cricket sends data as parameters directly to given HTTP link

addresses. It opens up endless Cricket configuration options with other systems and internet services such as IFTTT, Automate.io, and the like. You can even make your own custom HTTP service and integrate with more sophisticated systems.



Cricket supports both secure (HTTPS) and open (HTTP) request protocols with POST and GET methods which are described below.

HTTP dynamic tags

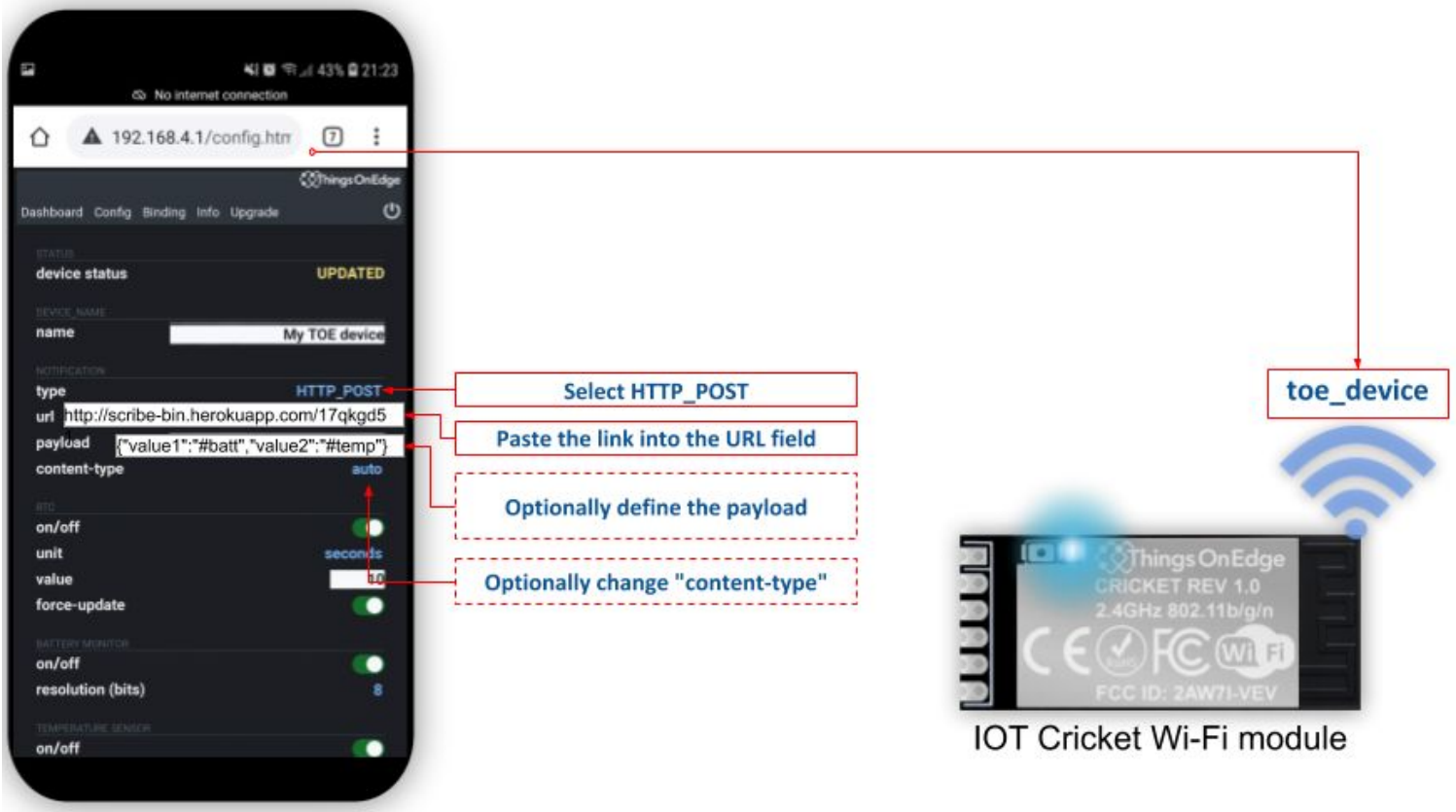
You can set Cricket to send data over either **payload** (in HTTP_POST) or **parameters** (in HTTP_GET) e.g. current IO values, battery level, temperature, etc.. In order to do that you need to specify tags with a preceding # character. If Cricket recognises a tag it is replaced with a corresponding value. The list of available tags is shown in the table below:

Tag	Description
#io1_wake_up	This tag is replaced by value 0 or 1. If the module was woken-up via external WAKE_UP pin this value is 1 otherwise 0
#rtc_wake_up	This tag is replaced by value 0 or 1. If the module was woken-up via internal RTC interrupt this value is 1 otherwise 0
#io2	This tag is replaced by the current value on IO2 port
#io3	This tag is replaced by the current value on IO3 port
#temp	This tag is replaced by the current tempere after averaging if enabled
#batt	This tag is replaced by current battery value after applying a divider if enabled

#hwc_wake_up	This tag is replaced by a counter representing number of wake ups
#hwc_wifi_enabled	This tag is replaced by a number of times Cricket connected to WiFi
#device_sn	This tag is replaced by Cricket's serial number
#device_name	This tag is replaced by the device name set in the configuration. Please note if you want to pass this parameter via HTTP GET and it includes some special characters such as:*space* !*'();:@&=+$,/?#[] it must be encoded with the percent-encoding method e.g. "my device" -> "my%20device"

HTTP POST requests

You can configure Cricket to send data (payload) to web servers via POST request method. It is often used to request a web server to process the payload which is attached to the request.



The type of content (payload format) is automatically detected by default (plain text or JSON) or you can explicitly set a desired format (in Cricket's configuration). Some web services are more relaxed than others in specifying the right payload format (content type). For example IFTTT requires JSON.

You can also instruct Cricket to use dynamic tags with # preceding character. A list of tags available you can find in the HTTP dynamic tags section. They are dynamically replaced with their corresponding actual values e.g. current temperature or IO values. Below is an example of payload definition with dynamic tags (see HTTP dynamic tags section for a complete list of tags available).

Example of your payload with tags: Payload processed and sent by Cricket

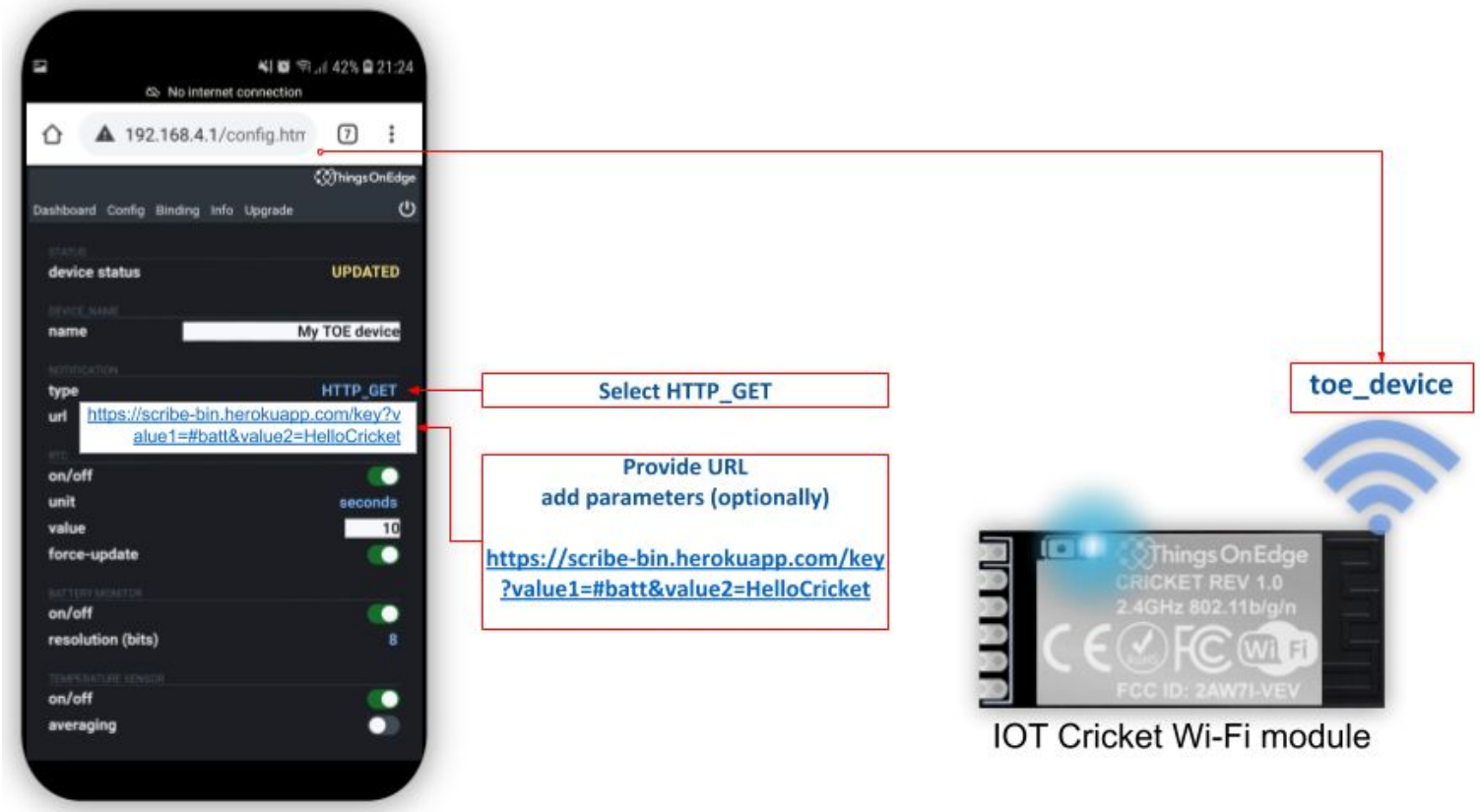
{"value1":"**#batt**","value2":"**#temp**"}	>>>	{"value1":"**120**","value2":"**22.0**"}

There is no limit of how many tags you can define in your payload. You can also use the same tag more than once in a payload.

HTTP GET requests

HTTP GET is used to request data from a URL web server. As part of the request it is possible to pass data as parameters within the URL's query string. So, you can set Cricket to send data via parameters with these methods.

For example:
URL: https://scribe-bin.herokuapp.com/key



You can also add parameters for example:
value1=#batt
value2=HelloCricket
in the following way: https://scribe-bin.herokuapp.com/key?value1=#batt&value2=HelloCricket

Cricket will replace the #batt tag onto the actual battery level value e.g. 120. See HTTP dynamic tags section for a complete list of tags available.

Example of your URL with tags		URL processed and sent by Cricket
https://scribe-bin.herokuapp.com/key?value1=**#batt**&value2=HelloCricket	>	

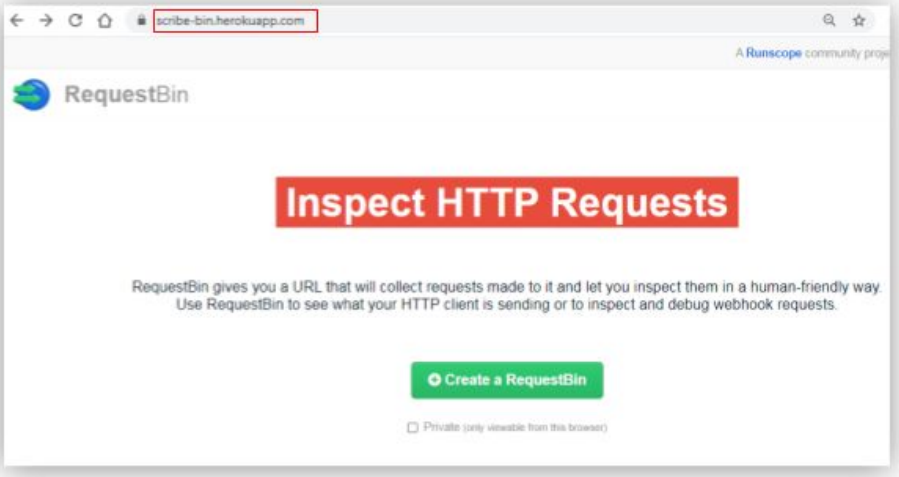
	https://scribe-bin.herokuapp.com/key?value1=**120**&value2=HelloCricket

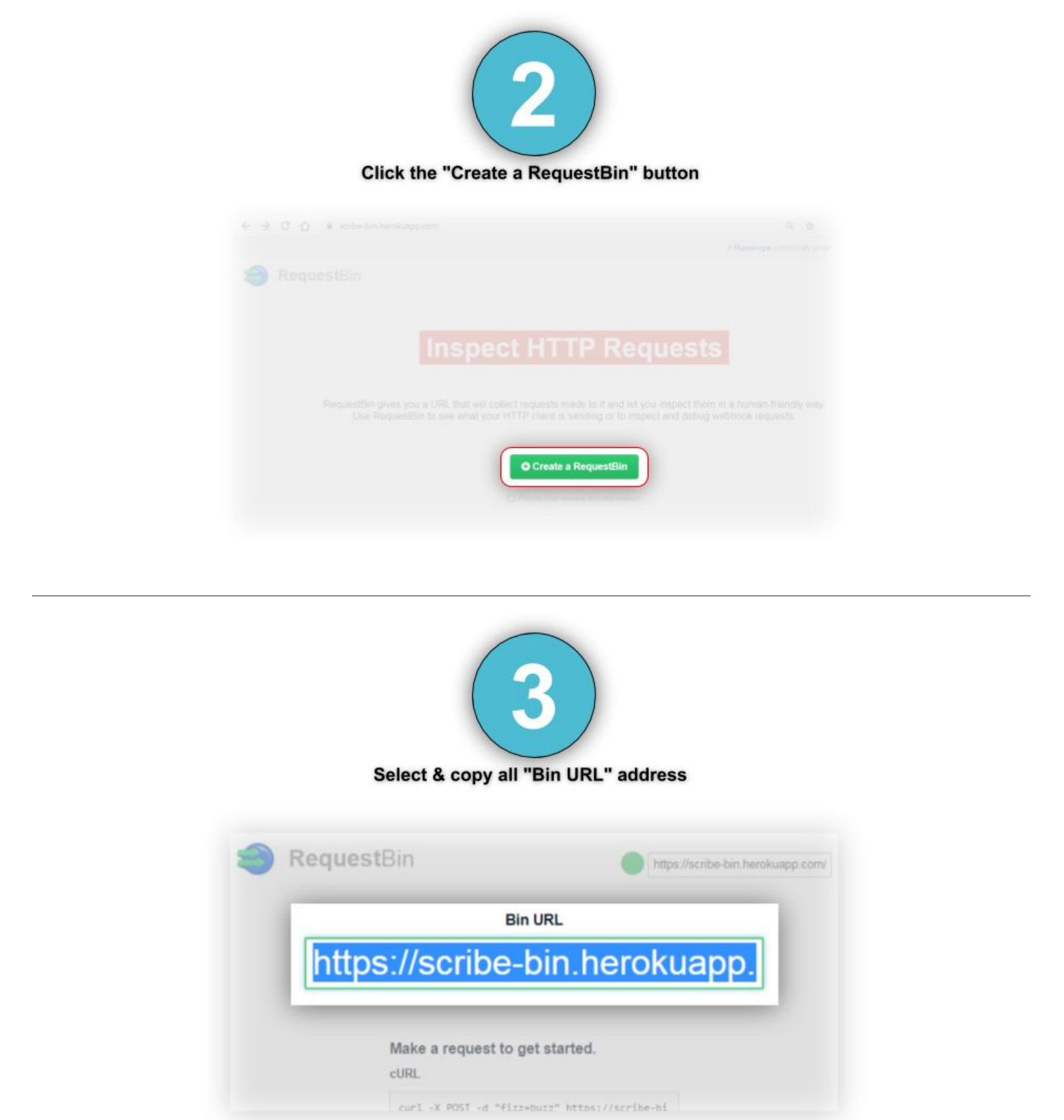
Examples

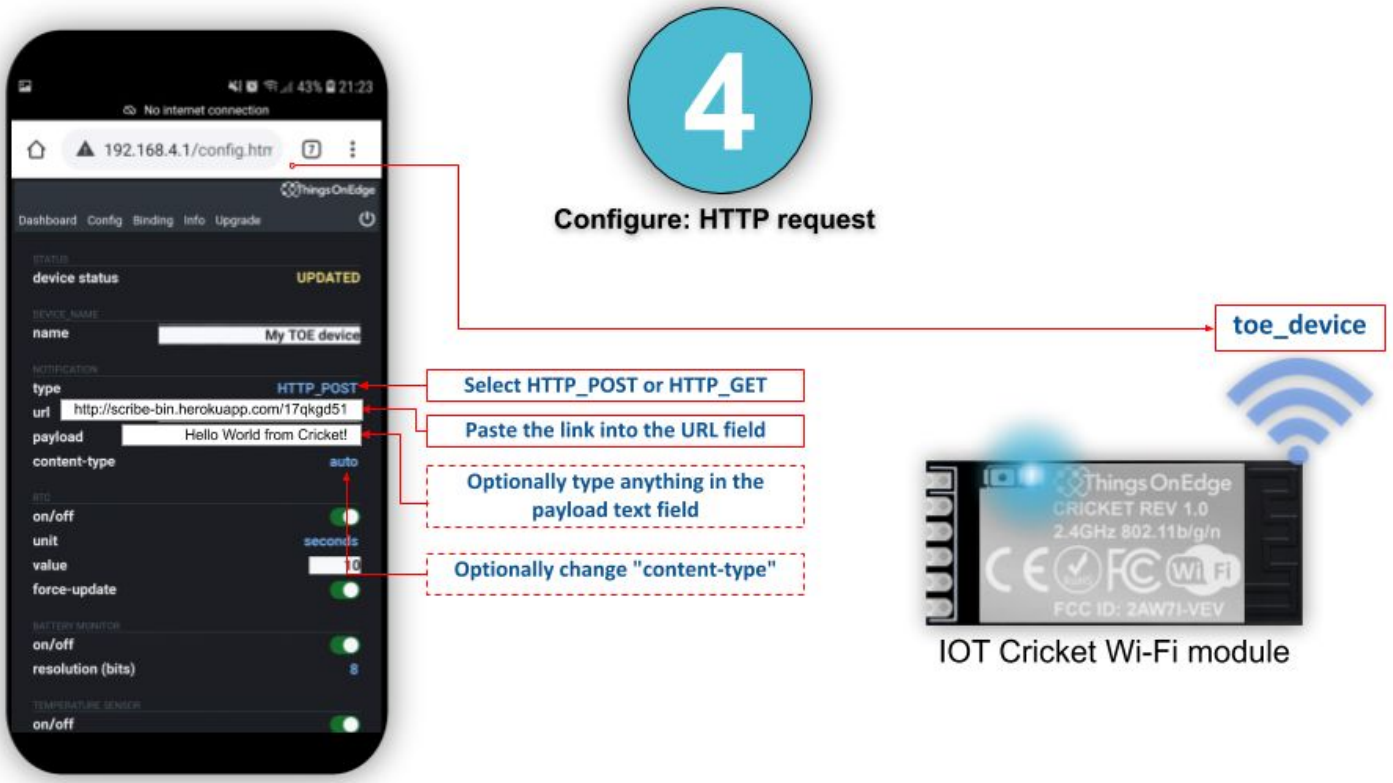
RequestBin

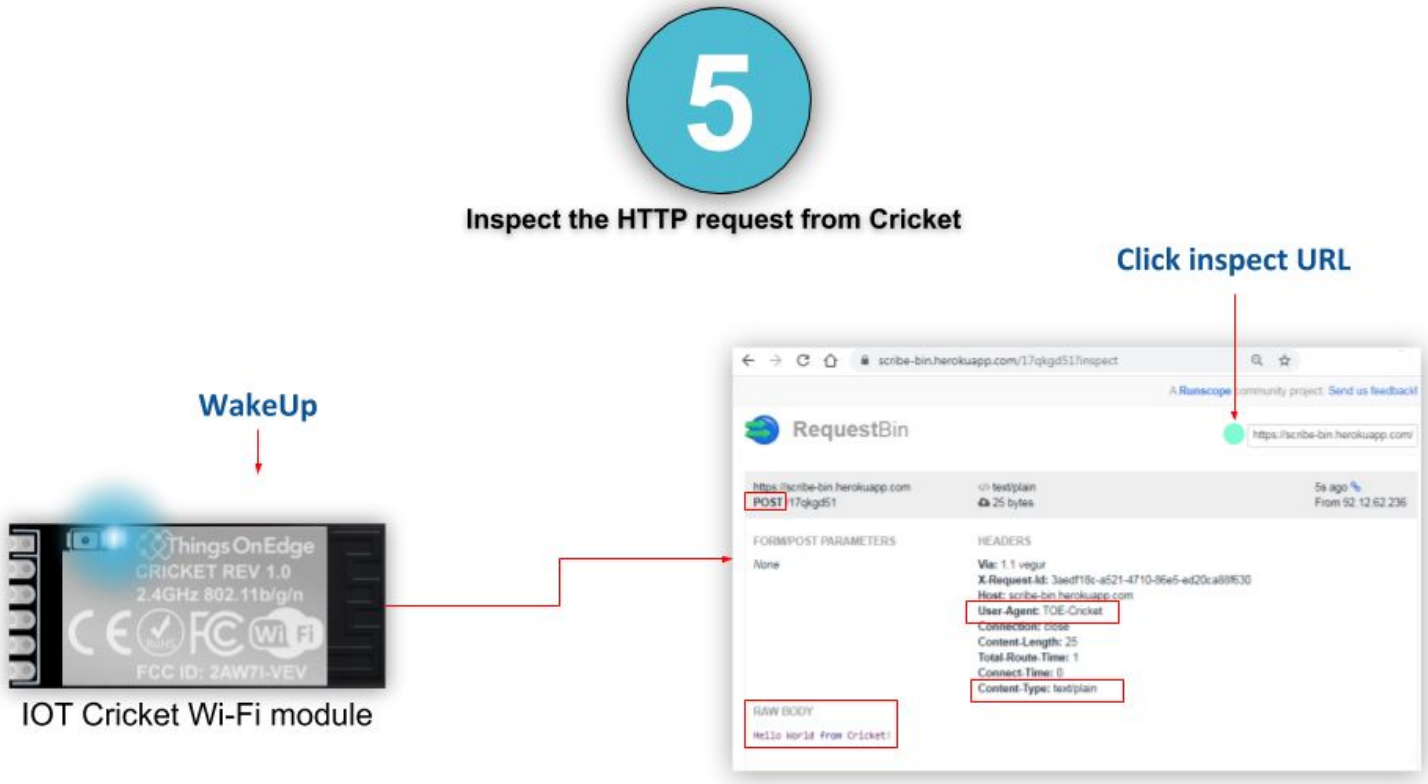
RequestBin is a lightweight webhook service which you can use to inspect what Cricket sends out over either HTTP POST or HTTP GET methods.

Goto https://scribe-bin.herokuapp.com/




2
Click the "Create a RequestBin" button
RequestBin
Inspect HTTP Requests
RequestBin gives you a URL that will collect requests made to it and let you inspect them in a human-friendly way.
Use RequestBin to see what your HTTP client is sending or to inspect and debug webhook requests.
Create a RequestBin
3
Select & copy all "Bin URL" address
RequestBin
https://scribe-bin.herokuapp.com/
Bin URL
https://scribe-bin.herokuapp.
Make a request to get started.
cURL





HTTP POST request: a button for sending emails over IFTTT example

In this example we show how you can configure Cricket to send email notifications with IFTTT when voltage is raised on the WAKE_UP port by using a button.

First you need to configure IFTTT service with Webhook

1. Go to: https://ifttt.com/
2. Login or register
3. Click **Create** from User / Account menu (top right corner)
4. Click **+** to create new source event
5. Select **Webhooks** service
6. Click **Continue**
7. Click **Receive a web request** (on the left hand side)
8. Create event name e.g. **email_button**
9. The source event should be set-up now, click **+** after Then event
10. Search **email** service
11. Change subject and body of the email accordingly
12. Click **Finish**

Now you need to get a HTTP address to which you can post events from Cricket. Search for **Webhooks** service and click on the document in the right up corner.

Copy web links under "**Make a POST or GET web request to:**"

Now you need to configure Cicket either locally or remotely (see Configuring Cricket) with the following parameters in the CONFIG panel:

Parameter	Value
type	*HTTP_POST*
url	*Copy / paste the link from Webhooks and do not forget replace **event** to **email_button** The link should look similar to the one shown below:* **https**://maker.ifttt.com/trigger/**email_button**/with/key/hfNIx8SKn_..._YW3xx5yFw5MGD
payload	*Leave it empty*

Do not forget to press the button on the Cricket module for 1 second to fetch this configuration.
Now whenever you press the attached button to Cricket's WAKE_UP pin you receive an email.

HTTP POST request: a payload with tags example

This example shows how you can configure Cricket to report a temperature every minute to your IFTTT service (for example log temperature into a Google spreadsheet).

Now you need to configure Cicket either locally or remotely (see Configuring Cricket) with the following parameters in the CONFIG panel:

Parameter	Value
type	HTTP_POST
url	http://maker.ifttt.com/trigge…. *[you must copy paste your url here]*
payload	{"value1":"#batt", "value2":"#temp"}
rtc	On
unit	minutes
value	1
BATTERY MONITOR	On
TEMPERATURE SENSOR	On

Once you set the configuration and restart Cricket, it will wake up every minute and send both the temperature and battery level to the IFTTT service.

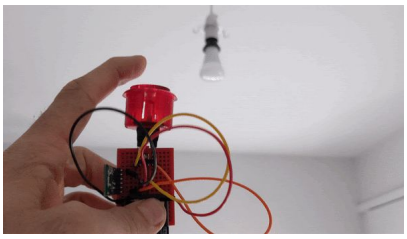
The payload message sent by Cricket, for this example, may look like this:
{"value1" : "120", "value2" : "22.0"}

If you wish you can calculate back the battery voltage on the client side with the following formula:
Batt_vcc = (3.5 / 256) * 120 = 1.64V

Complete examples

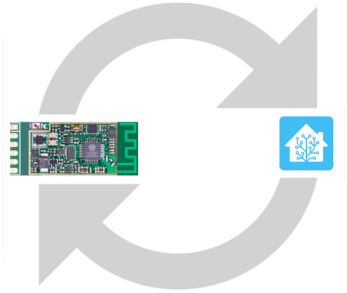
Below you can find a few examples with a full step by step guidance on how Things On Edge technology is applied.

[IoT Moisture Sensor - based on MQTT]



[IoT Button with IFTTT integration]

[IoT Motion Sensor with Email alerts]



[Home Assistant integration]

Things On Edge

thingsonedge.com