MediaTek Labs Tutorial Template

IA:

Dev Tools and Resources
    (Platform section)
        Documentation and Training
            (other item)
            This tutorial
            (other item)

Platform Documentation and Training page update:

Training

- ...
- (other tutorial)
  Other tutorial blurb
- This tutorial
  This tutorial blurb
- (other tutorial)
  Other tutorial blurb
- ...

Side bar menu:

Tutorial content:

Title: Connecting LinkIt Smart 7688 to MediaTek Cloud Sandbox with Python

Introduction

MediaTek Cloud Sandbox is an IoT device prototyping service. In this guide you'll learn the steps to create a simple remote switch that allows you to turn on and off the on-board Wi-Fi LED from the web console of MCS using Python.

Before you start

If you haven't built a LinkIt Smart 7688 project before using MediaTek Cloud Sandbox, this section describes the steps you need to follow before commencing this project.

Setup your development environment

Full details on setting up the necessary LinkIt Smart 7688 development environment can be found in Get Started. Complete this before you continue, if you haven't already set up your development environment.

MediaTek Cloud Sandbox

This tutorial used the MediaTek Cloud Sandbox (MCS) to control the Wi-Fi LED of the LinkIt Smart 7688 from the web console. To use MCS register for a Labs account, if you haven't done so already, and activate your MCS account. You will then be able to define prototypes for your own devices and applications. By registering on Labs you also gain access to the hardware reference designs, the ability to submit and respond to items in the forums, and more.

Python modules

This tutorial uses `requests` module. It is an easy-to-use HTTP request library. You can install it using `pip`.The steps are:

1) Make sure your LinkIt Smart 7688 is connected to the Internet (station mode).
2) Open a system console using SSH
3) Use pip to install module, for example:

```
pip install requests
```

Building the LinkIt Smart 7688 MCS Hardware

This section describes the hardware needed to build this tutorial provides details on how to put them together.

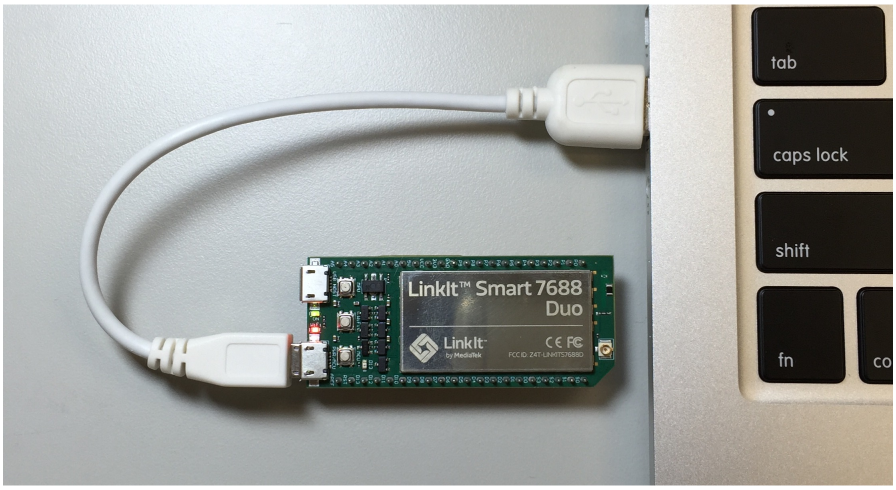
What you need

To build the (tutorial) hardware, in addition to a LinkIt (version) development board, you need the following components:

- LinkIt Smart 7688 development board
- Micro USB cable
- Host computer

Putting the components together

This section provides step-by-step instructions on putting the (tutorial) hardware together.



Hardware components

1) Connect the micro USB cable to the power connector of LinkIt Smart 7688 and a host computer.
2) Change LinkIt Smart 7688 to Station mode and connect to Internet. Please check LinkIt Smart 7688 Get Started Guide on using the web-based configuration tool to change to Station mode.

Setup LinkIt Smart 7688 in MCS

In this section you'll create a prototype device in MCS and connect it to LinkIt Smart 7688.

Step 1 : Registration

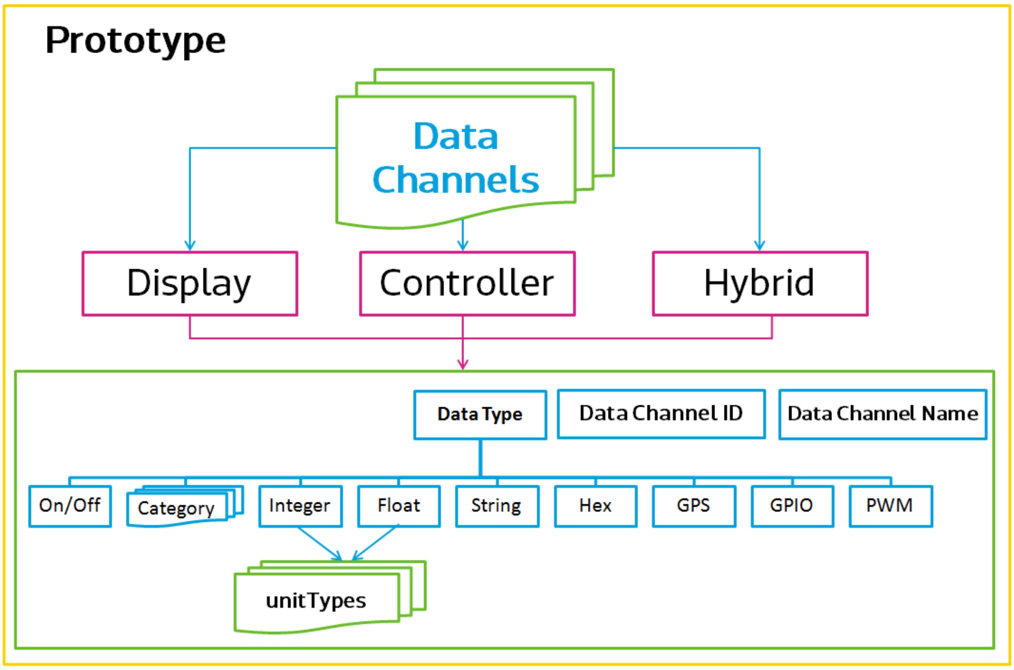
Click here to register on MCS. It's free.

Step 2: Setup

Activate a MCS account to prototype your own devices and applications. The connecting tutorial follows the general steps of application development on the MCS.

Step 3: Creating a new prototype for LinkIt Smart 7688

A prototype serves as a blueprint for the actual hardware setup. The prototype consists of one or more data channels of type display, controller and hybrid. The data channels are defined with a Data channel name, Data channel id and data type as required parameters. The variety of data types and overall structure of the prototype in general is shown in the figure below.

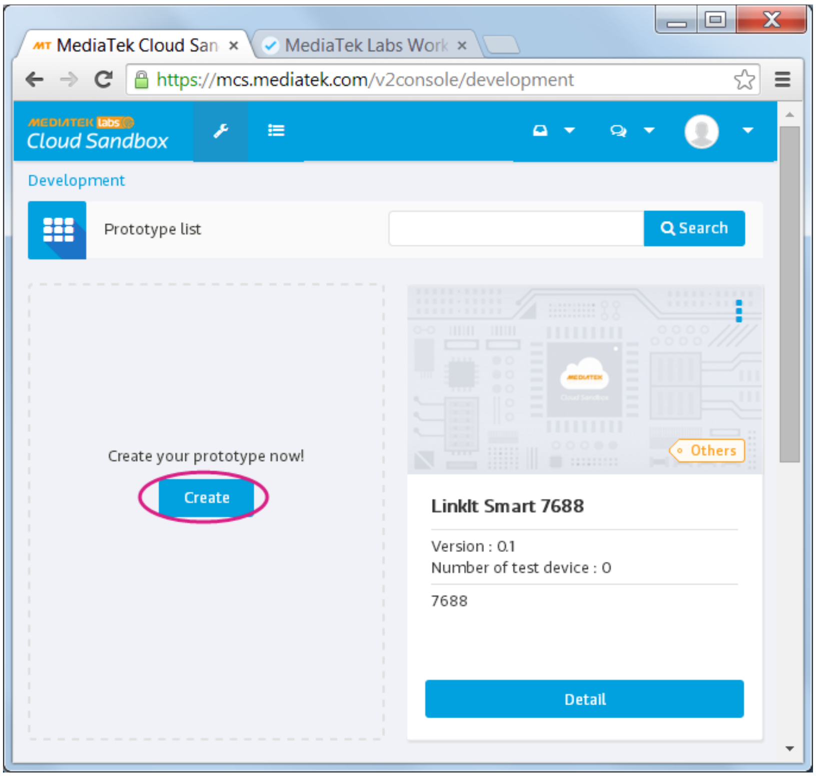


General prototype content in MCS

The LinkIt Smart 7688 prototype in this tutorial has one data channel to control Wi-Fi LED.

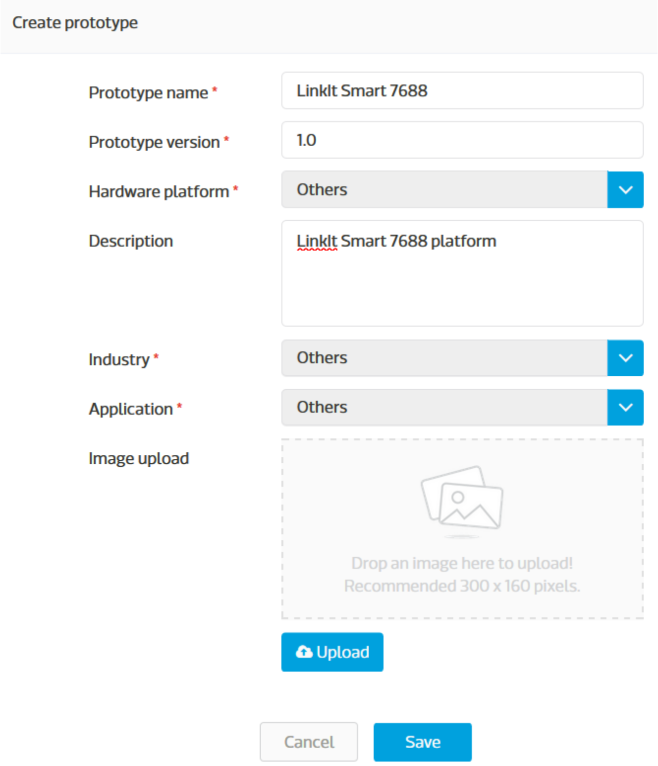
This section describes the details on how to create and configure a LinkIt Smart 7688 with corresponding data channel.

1) Click **Development** from the navigation toolbar, and then under Prototype list click **Create** to create a new prototype, as shown below.



Create a new prototype

2) In **Create prototype** define a basic profile of this prototype.



Create prototype

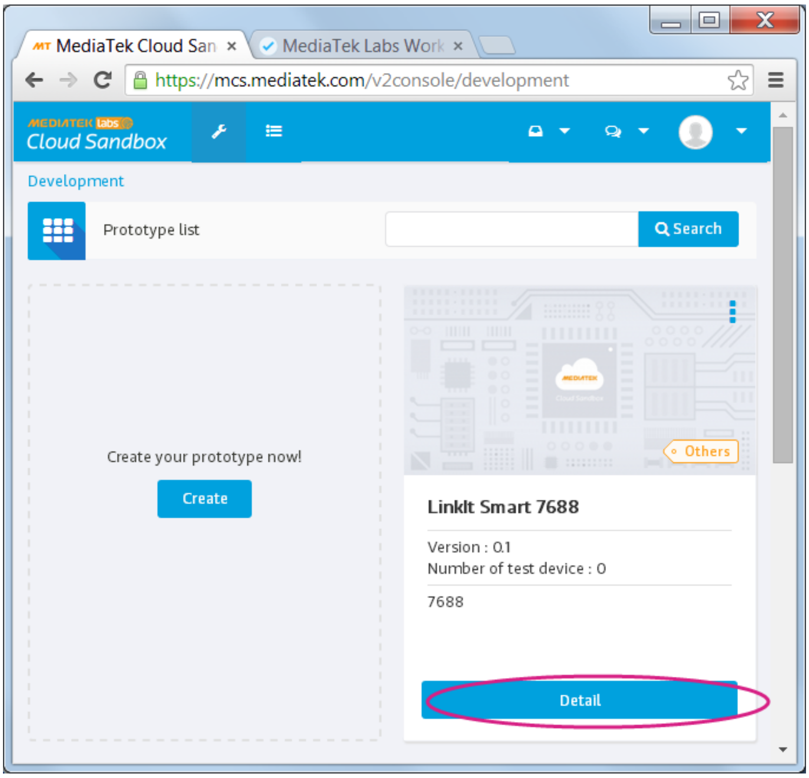
Prototype name *	LinkIt Smart 7688
Prototype version *	1.0
Hardware platform *	Others
Description	LinkIt Smart 7688 platform
Industry *	Others
Application *	Others
Image upload	Drop an image here to upload! Recommended 300 x 160 pixels.

Create a prototype profile

Save the information and proceed to the next step.

Note: Fields marked by a red asterisk (*) are required fields.

3) Click **Detail** to view the prototype information, as shown below.



MediaTek Cloud San ×
MediaTek Labs Work ×
https://mcs.mediatek.com/v2console/development

MEDIATEK labs Cloud Sandbox

Development

Prototype list

Search

Create your prototype now!

Create

Others

LinkIt Smart 7688

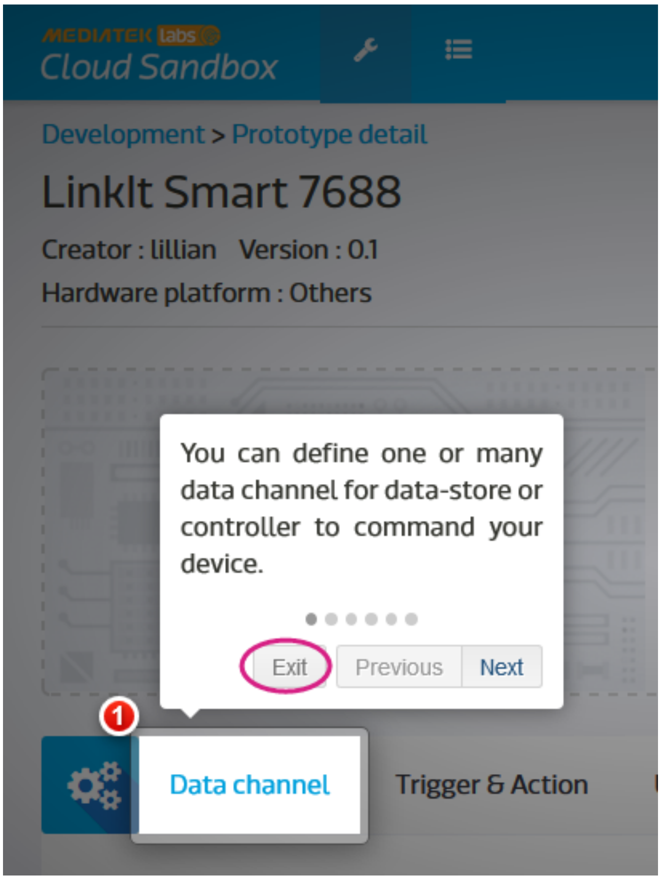
Version : 0.1
Number of test device : 0

7688

Detail

Accessing the LinkIt Smart 7688 prototype details

4) Click **Exit** to bypass initiation instructions, as shown below.



MEDIATEK labs Cloud Sandbox

Development > Prototype detail

LinkIt Smart 7688

Creator : lillian Version : 0.1

Hardware platform : Others

You can define one or many data channel for data-store or controller to command your device.

Exit Previous Next

1

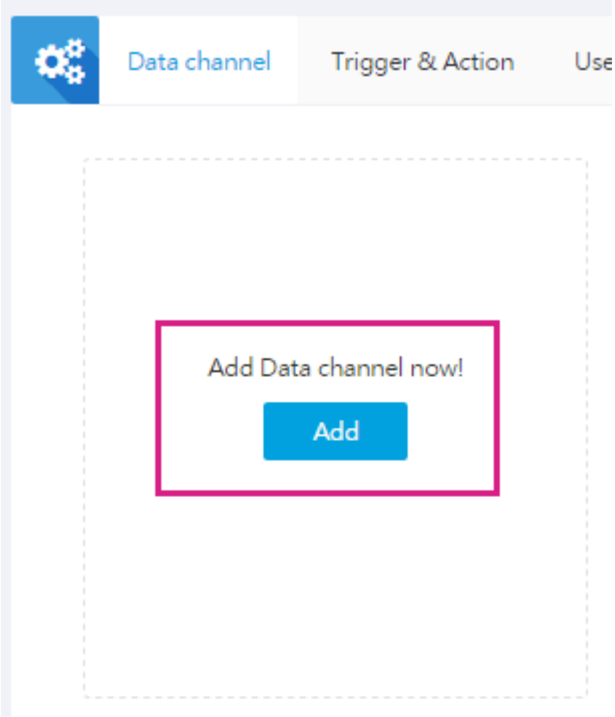
Data channel

Trigger & Action

Bypassing the data channel initiation instructions

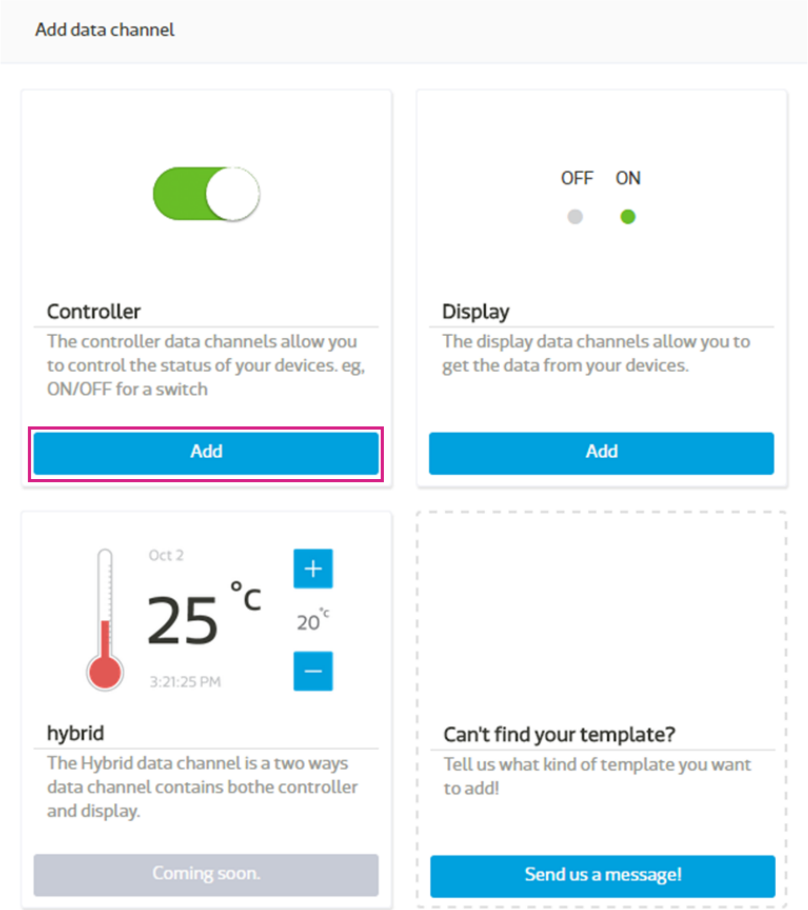
Next, add Data Channels for Data Control.

5) Click **Add** under **Data channel** toolbar, as shown below to provide data channels for the LinkIt Smart 7688 prototype.



Add a data channel to the LinkIt Smart 7688 prototype

6) Create a **Controller** data channel for Red color LED control on MCS by clicking **Add**.



Add data channel

Controller

The controller data channels allow you to control the status of your devices. eg, ON/OFF for a switch

Display

The display data channels allow you to get the data from your devices.

hybrid

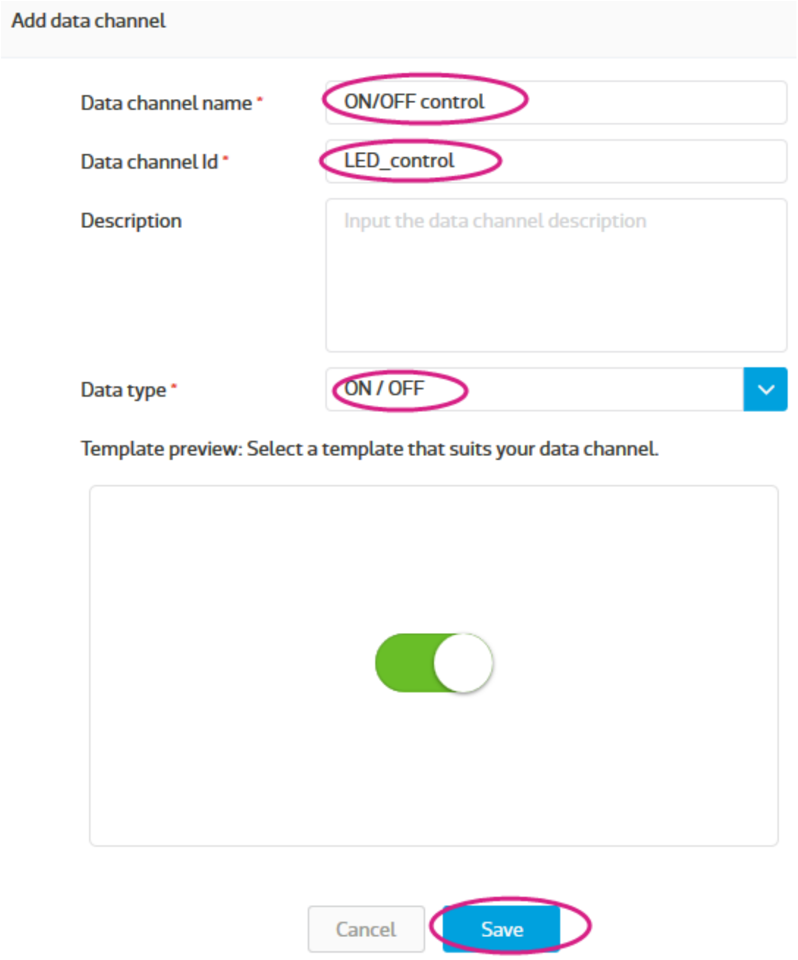
The Hybrid data channel is a two ways data channel contains bothe controller and display.

Can't find your template?

Tell us what kind of template you want to add!

Add a data channel type to the LinkIt Smart 7688 interface

7) Enter the information for the controller as shown and click **Save**. This defines the data channel and will be used in the Python program to communicate the data from the boards to MCS.



Add data channel

Data channel name * ON/OFF control

Data channel Id * LED_control

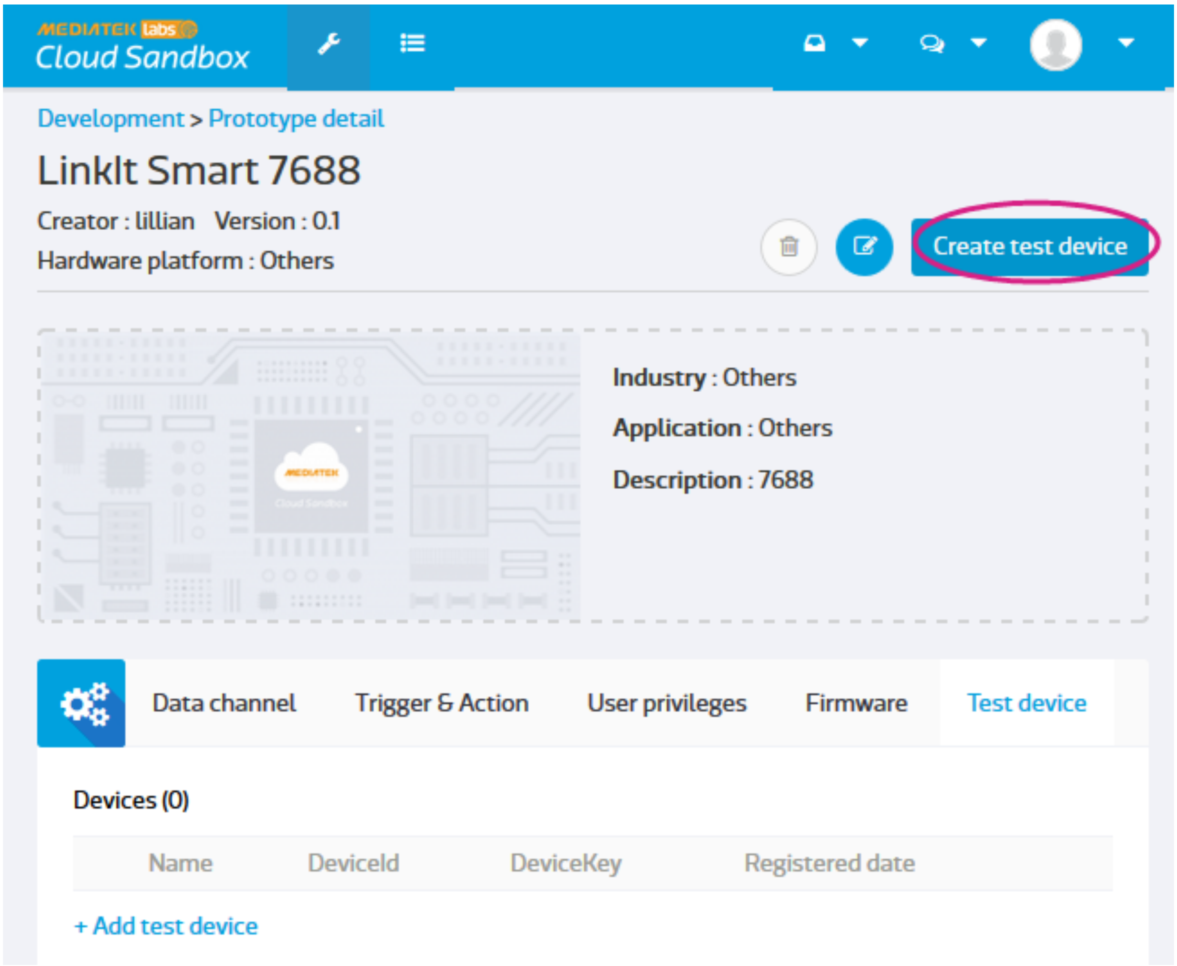
Description Input the data channel description

Data type * ON / OFF

Template preview: Select a template that suits your data channel.

Data channel details

8) Create a test device based on the LS7688 prototype as shown below. Each prototype on MCS is defined on a specific hardware platform. The hardware platform assigned in this tutorial is the LinkIt Smart 7688 development board. The LS7688 prototype is mapped to the actual device. Click **Create test device** as shown below.



Development > Prototype detail

LinkIt Smart 7688

Creator : lillian Version : 0.1

Hardware platform : Others

Industry : Others

Application : Others

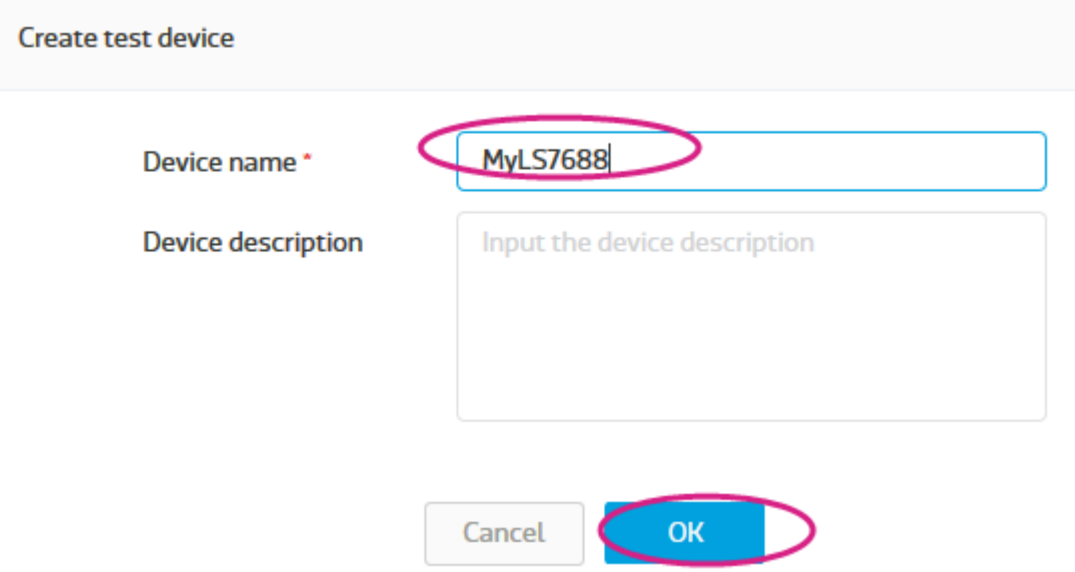
Description : 7688

Devices (0)

Name	DeviceId	DeviceKey	Registered date

Creating test device for LinkIt Smart 7688 prototype

9) Provide Device name, as an example **device name** shown in the figure below, click **OK** to continue.



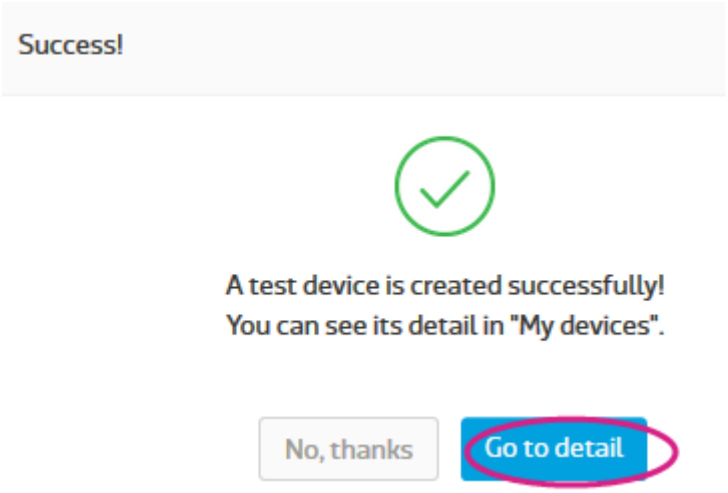
Create test device

Device name * MyLS7688

Device description: Input the device description

Test device configuration

10) Click **Go to detail**, as shown in the figure below to view the device information.

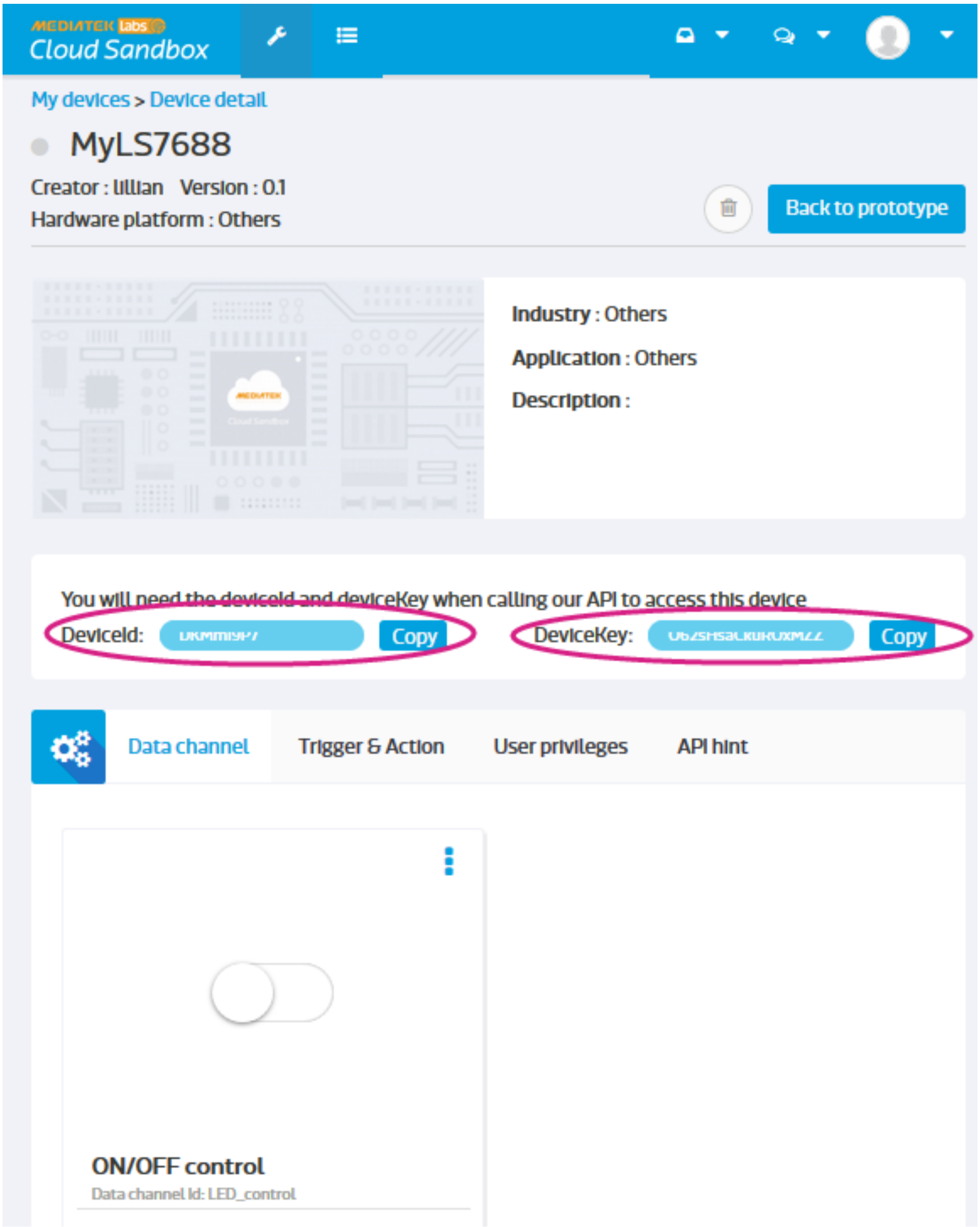


Success!

A test device is created successfully!
You can see its detail in "My devices".

1 Confirmation on creating a device for 7688 prototype

11) It’s essential to store the **DeviceID** and **DeviceKey** values in a file because you’ll need to replace them in the Python program in the next step to enable API calls and device connectivity.



MyLS7688

Creator : lillian Version : 0.1

Hardware platform : Others

Industry : Others

Application : Others

Description :

You will need the deviceId and deviceKey when calling our API to access this device

DeviceId: DKmmI9P7

DeviceKey: U6zSHsaLKuKUXMZZ

ON/OFF control

Data channel Id: LED_control

Device detail for the LinkIt Smart 7688 prototype

You have now created a new device in MCS matching the LinkIt Smart 7688 prototype with a data channel type Control (ON/OFF) as a switch to control the LED. Next section describes the software implementation.

Create a Python program to connect to MCS

This section describes the Python code example that listens for commands from MCS web console. The example is explained in steps and the complete example code is presented in the last step.

- Establish command pipe to MCS

- Send heart beat to TCP socket
- Parse the command
- Complete example code

Step1: Establish command pipe to MCS

To establish a command pipe to MCS, you need to create a TCP socket that connects to the command server. To connect to command server, you need to query the IP address and port of the command server by calling a RESTful API from MCS.

```
DEVICE_INFO = {
    'device_id' : 'YOUR_DEVICE_ID',
         'device_key' : 'YOUR_DEVICE_KEY'
}

# change 'INFO' to 'WARNING' to filter info messages
logging.basicConfig(level='INFO')

def establishCommandChannel():
    # Query command server's IP & port
    connectionAPI =

'https://api.mediatek.com/mcs/v2/devices/%(device_id)s/connections.csv'
   r = requests.get(connectionAPI % DEVICE_INFO,
           headers = {'deviceKey' : DEVICE_INFO['device_key'],
                     'Content-Type' : 'text/csv'})
   logging.info("Command Channel IP,port=" + r.text)
   (ip, port) = r.text.split(',')

   # Connect to command server
   s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
   s.connect((ip, int(port)))
   s.settimeout(None)
```

After tvihe TCP socket is connected to the command server, the server will send commands that reflects the status of the web console, such as the status of the ON/OFF switch. However, MCS requires you to send a heart beat to the command server every minute in order to keep the TCP socket active. You'll learn to do that in the next step.

Step 2: Send heart beat to TCP command

Create a Python program that sends heart beat to TCP command every 40 seconds using threading module.

Step 3: Parse the command

The server sends commands in the following format: `deviceId, deviceKey, timestamp, dataChannelId,` and `commandValue`. You can use comma "," to parse these commands. You also need to check the command type by their length because the server echoes heart beat command back to the device.

```
while True:
command = commandChannel.recv(1024)
logging.info("recv:" + command)
# command can be a response of heart beat or an update of the LED_control,
# so we split by ',' and drop device id and device key and check length
fields = command.split(',')[2:]

if len(fields) > 1:
   timeStamp, dataChannelId, commandString = fields
   if dataChannelId == 'LED_control':
      # check the value - it's either 0 or 1
      commandValue = int(commandString)
      logging.info("led :%d" % commandValue)
```

Please refer to MCS API for more information.

Step 4: Complete example code

The complete example code is presented below. Create a file mcs_blink.py in the system console and copy/paste the example code.

The following is used in the example:

- Device ID:ABC123
- Device Key:XYZ123

Replace the above with your device ID and device key.

```
import requests
import socket
import threading
import logging
import mraa

# change this to the values from MCS web console
DEVICE_INFO = {

    'device_id' : 'ABC123',
    'device_key' : 'XYZ123'

}
# change 'INFO' to 'WARNING' to filter info messages
logging.basicConfig(level='INFO')
heartBeatTask = None
def establishCommandChannel():
    # Query command server's IP & port
    connectionAPI =
'https://api.mediatek.com/mcs/v2/devices/ABC123/connections.csv'
    r = requests.get(connectionAPI % DEVICE_INFO,
               headers = {'deviceKey' : DEVICE_INFO['device_key'],
                           'Content-Type' : 'text/csv'})
    logging.info("Command Channel IP,port=" + r.text)
    (ip, port) = r.text.split(',')

    # Connect to command server
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((ip, int(port)))
    s.settimeout(None)

    # Heartbeat for command server to keep the channel alive
    def sendHeartBeat(commandChannel):
        keepAliveMessage = 'ABC123,XYZ123,0' % DEVICE_INFO
        commandChannel.sendall(keepAliveMessage)
        logging.info("beat:%s" % keepAliveMessage)
    def heartBeat(commandChannel):
        sendHeartBeat(commandChannel)
        # Re-start the timer periodically
        global heartBeatTask
        heartBeatTask = threading.Timer(40, heartBeat, [commandChannel]).start()

    heartBeat(s)
    return s
def waitAndExecuteCommand(commandChannel):
    while True:

        command = commandChannel.recv(1024)

        logging.info("recv:" + command)
        # command can be a response of heart beat or an update of the LED_control,
        # it’s split by ',' and drop device ID and device key and check length
        fields = command.split(',')[2:]

        if len(fields) > 1:
           timeStamp, dataChannelId, commandString = fields
           if dataChannelId == 'LED_control':
                 # check the value - it's either 0 or 1
                commandValue = int(commandString)
                logging.info("led :%d" % commandValue)
                setLED(commandValue)
    pin = None
    def setupLED():
        global pin
        # on LinkIt Smart 7688, pin 44 is the Wi-Fi LED.
        pin = mraa.Gpio(44)
        pin.dir(mraa.DIR_OUT)
    def setLED(state):
        # Note the LED is "reversed" to the pin's GPIO status.
        # So you need to reverse it here.

       if state:
          pin.write(0)
       else:
          pin.write(1)
    if __name__ == '__main__':
       setupLED()
       channel = establishCommandChannel()
       waitAndExecuteCommand(channel)
```

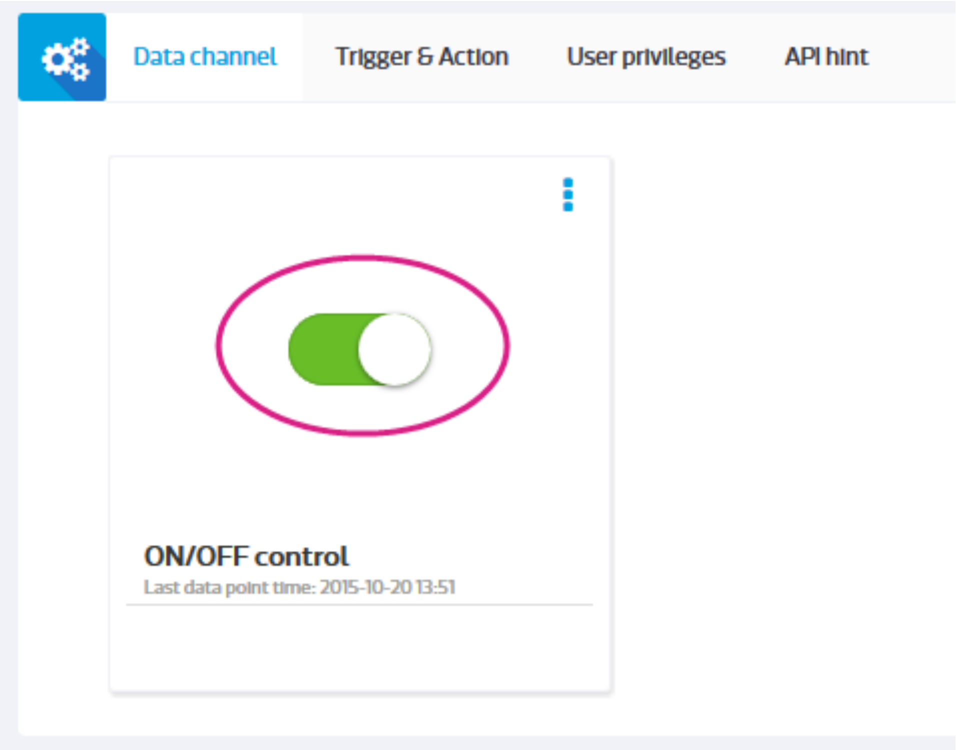
The LED value is set based on the commands value. Please refer to Chapter 5, “Peripheral programming on LinkIt Smart 7688" for more information on how to control the LEDs using mraa. Note that the Wi-Fi LED is turned ON when the GPIO 44 is set to LOW.

Run your application

You are now ready to execute the Python program. In the system console, type the following command:# is command prompt and is not part of command.

```
# python blink.py
```

Go to MediaTek Cloud Sandbox and use the controller panel to flip the button on and off and watch the Wi-Fi LED on LinkIt Smart 7688



ON/OFF control

Last data point time: 2015-10-20 13:51

Using MCS control switch to control LED

Conclusion

In this tutorial you've implemented a remote controlled LED switch application using LinkIt Smart 7688 development board, MediaTek Cloud Sandbox and Python programming language.

For more information on MediaTek LinkIt Smart 7688 development and prototyping board and cloud services refer to LinkIt Smart 7688 Developer's Guide and MediaTek Cloud Sandbox.