Lab 2 – IoT with MediaTek Cloud Sandbox (MCS)

Introduction to IoT
Fall 2020
Outline

• Lab Objective
• Basic MCS
  • Set the Environment
  • Data Channel Types
  • Data Types
• Using MCS
  • Create account
  • Create prototype
  • Create datachannel
• Storage in MCS
  • Monitor DHT11 from MCS
  • Turn On LED from MCS
• Computing in MCS
  • Automatically Turn On LED from MCS based on LDR Input
• Assignment 2 Specifications
Lab Objective

• Connect Raspberry Pi to MCS

• Capable to Store the sensor data in MCS
  • Use Cloud Services to collect data from sensors
    • Monitor DHT sensor from MCS
  • Use Cloud Services to trigger a reaction to actuators
    • Turn on LED from MCS

• Capable to make a computing program in MCS
  • Use Cloud Services to compute and make a decision
    • Automatically Turn On LED based on LDR from MCS
What is MCS?

• Cloud based data service platform for IoT devices.

Why MediaTek Cloud Sandbox?

- Remote control
- Reliable time series data storage
- Firmware Over-the-Air (OTA)
- Trigger and action
- User privilege access control
- Rich set of RESTful APIs
MCS Features (1/3)

Remote control
MCS allows you to directly control the devices from the web console directly anytime, anywhere. A special designed mobile application enables the developer to easily control the device or to view the uploaded data points.

 Reliable Time series data storage
MCS provides time series data storage service allows you to view the visualized data gathered by the connected devices. You can see the current data and historical data by use the time filter to extract the time period of data you are interested in.
Firmware Over-the-Air (OTA)

You can upload firmware and push firmware to device directly. The developer can manage all the firmware for a prototype and select which test device is taking the upgrade.

Trigger and action

MCS enable you to set the trigger and the action for the test device. Whenever the trigger is activated, the corresponding actions will be triggered. The action includes sending email notification and GCM service.
MCS Features (3/3)

User privilege access control

You can manage the access right for your prototype and test device to share the data to specific users to edit or to view only. The prototype and device can have separate user privilege access base on your needs.

- Owner
- Administrator
- Viewer

Rich set of RESTful APIs

MCS provides a rich set of RESTful APIs facilitating the developers to upload and retrieve the data points, get connection, and get device information.

https://mcs.mediatek.com/
Other Platforms

About 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 visualization of live data, and send alerts.

Collect
Send sensor data privately to the cloud.

Analyze
Analyze and visualize your data with MATLAB.

Act
Trigger a reaction.

AWS IoT

IoT services for industrial, consumer, and commercial solutions

Firebase helps mobile and web app teams succeed

Build better apps
Jump to develop products

Improve app quality
Jump to quality products
Cloud – Raspberry Architecture

[Diagram showing the architecture with steps:
- Push data points into clouds
- Send notification for specific event
- Remote switching and control
- Retrieve data generated from device]

How to Use the MCS?

• Register an account ([https://mcs.mediatek.com/oauth/en/login](https://mcs.mediatek.com/oauth/en/login)).
• Verify and complete the registration process
• Creating several things
  • Your own prototypes,
  • data channels,
  • Devices
• Start to manage via :
  • MCS web console
  • Mobile App or
  • RESTful APIs.
MCS – Set the Environment (1/2)

1. Create prototype
2. Add data channel to prototype
3. Create test device from prototype
MCS – Set the Environment (2/2)

• Prototype
  • The blueprint of the device you’re developing.
  • Each prototype has a unique prototype ID and prototype key.

• Data Channel
  • Data generated that will be stored in the MCS.
  • A command that can be sent from the MCS to the device.

• Test Devices
  • Each device is given an unique **Device ID** and **Device Key**
  • It use for testing the prototype
  • Maximum 10 test devices in one prototype
MCS - Data Channel Types

• Display
  • To store and display the data from a device.
  • For example, data from a temperature sensor

• Controller
  • Data generated in the MCS and sent to the device to control the setting
  • For example, a switch to turn a LED on or off.

• Hybrid
  • The combination of Display and Controller
  • For example, the data from a temperature sensor and the control settings for a fan.
MCS – Data Types

• **ON/OFF**
  A switch such as turning a light on or off.

• **Int**
  Represents an arbitrary integer, such as the number of steps a user has taken.

• **float**
  Represents an arbitrary floating point number, such as the temperature.

• **GPS**
  Represents a geo-location identified by longitude, latitude, elevation and related attributes.

• **Analog**
  An analog slider that allows you to fine tune the settings, such as music volume controls.

• **Gamepad**
  Represents a direction controller with four-way digital buttons.
  You can control your robot or remote control car easily with this data channel.

• **Image**
  Data channels for images. The images will be stored in PNG format after uploaded.

• **Video Stream**
  Data channels for video streaming application, such as a home surveillance camera.
MCS – Create an Account

Click the follow link to create an account and sign in:
https://mcs.mediatek.com/oauth/en/login
MCS – Create a Prototype

- Development -> Prototype
- Create a new prototype
- Follow the setting on next page
MCS – Create a Prototype

- You can type anything you want at here.
- For Hardware Platform, choose Raspberry Pi
- Finally, click the save button!
MCS – Create a Data Channel (1/4)

• On the prototype page, add a new data channel
MCS – Create a Data Channel (2/4)

Add data channel

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

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

Function
Function data channels process your device data and return value to other data channels.

Can't find your template?
Tell us what kind of template you want to add!

Add
Add
Add
Send us a message!
MCS – Create a Data Channel (3/4)

- Please remember **data channel id**, since we will use it in Restful API
MCS – Create a Data Channel (4/4)
MCS – Create a Test Device (1/2)

- Don’t click “Create as public device” if you don’t want to let everyone watch your data.
MCS – Create a Test Device (2/2)

- Development -> Test devices
- Click the Detail and play with Restful API!
Storage in MCS
Temperature Monitoring in MCS

21 degree Celsius

82 percent

Temperature

Humidity

Last data point time: 2020-11-05 14:41

Temperature

Humidity
import time
import http.client, urllib
import json
import Adafruit_DHT
import socket

# Set Pin No, MediaTek Cloud Sandbox (MCS) Key
sensor = Adafruit_DHT.DHT11
pin = 21

deviceId = "YOUR_DEVICE_ID"
deviceKey = "YOUR_DEVICE_KEY"

# Set MediaTek Cloud Sandbox (MCS) Connection
def post_to_mcs(payload):
    headers = {"Content-type": "application/json", "deviceKey": deviceKey}
    not_connected = 1
    while (not_connected):
        try:
            conn = http.client.HTTPConnection("api.mediatek.com:80")
            conn.connect()
            not_connected = 0
        except (http.client.HTTPException, socket.error) as ex:
            print("Error: %s" % ex)
            time.sleep(10)  # sleep 10 seconds
    conn.request("POST", "/mcs/v2/devices/" + deviceId + "/datapoints", json.dumps(payload), headers)
    response = conn.getresponse()
    print(response.status, response.reason, json.dumps(payload), time.strftime("%c"))
data = response.read()
    conn.close()

# Post MediaTek Cloud Sandbox (MCS)
while True:
    [humidity, temp] = Adafruit_DHT.read_retry(sensor, pin)
    print("temp = %.02f C humidity =%.02f%%"%(temp, humidity))
    payload = {"datapoints": [{"dataChnId":"humidity","values":{"value":humidity}},{"dataChnId":"temperature","values":{"value":temp}}]}
    post_to_mcs(payload)
    time.sleep(5)
Controlling LED with MCS

Temperature
Last data point time: 2020-11-05 16:46
Temperature

Humidity
Last data point time: 2020-11-05 16:46
Humidity

LED Switch
Last data point time: 2020-11-05 15:15
LED Switch

21 degree Celsius

82 percent

LED turning on.
LED turning on.
LED turning on.
LED turning on.
LED turning on.
# Set the LED PIN
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
ledPin = 36
GPIO.setup(ledPin,GPIO.OUT)

deviceId = "YOUR_DEVICE_ID"
deviceKey = "YOUR_DEVICE_KEY"

# Set MediaTek Cloud Sandbox (MCS) Connection

def get_to_mcs():
    host = "http://api.mediatek.com"
    endpoint = "/mcs/v2/devices/" + deviceId + "/datachannels/ledswitch/datapoints"
    url = host + endpoint
    headers = {"Content-type": "application/json", "deviceKey": deviceKey}
    r = requests.get(url,headers=headers)
    value = (r.json())["dataChannels"][0]["dataPoints"][0]["values"]["value"]
    return value

# Receive the Data from MCS
while(True):
    if(get_to_mcs()==1):
        print("LED turning on.")
        GPIO.output(ledPin,GPIO.HIGH)
        time.sleep(0.5)
    
    if(get_to_mcs()==0):
        print("LED turning off.")
        GPIO.output(ledPin,GPIO.LOW)
        time.sleep(0.5)
Computing in MCS
Automatically Turn On LED

```
1. // Use context.value to get uploaded data point.
2. // Following is an example of temperature conversion:
3. 4. // var fahrenheit = context.value;
5. // var celsius = (fahrenheit - 32) / 1.8;
6. // return (celsius);
7. // dataChnlId: fahrenheit,
8. // dataChnlId: celsius
9. // });
10. var ldrValue = context.value;
11. if (ldrValue>=3000) return (ledSwitch:1);
12. else
13. return (ledSwitch:0);
```
Automatically Turn On LED (Source Code)

import time
import http.client, urllib
import json
import RPi.GPIO as GPIO
import requests
import socket
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
ledPin = 40
ldrPin = 36
GPIO.setup(ledPin,GPIO.OUT)
deviceId = "Db0B5DJQ"
deviceKey = "nrG983uburMaldDz"

# Set MediaTek Cloud Sandbox (MCS) Connection

def post_to_mcs(payload):
    headers = {
        "Content-type": "application/json",
        "deviceKey": deviceKey
    }
    not_connected = 1
    while (not_connected):
        try:
            conn = http.client.HTTPConnection("api.mediatek.com:80")
            conn.connect()
            not_connected = 0
        except (http.client.HTTPException, socket.error) as ex:
            print ("Error: %s" % ex)
            time.sleep(10)  # sleep 10 seconds
    conn.request("POST", "/mcs/v2/devices/" + deviceId + "/datapoints",
    json.dumps(payload), headers)
    response = conn.getresponse()
    print("%s" % response.status, response.reason)
    time.sleep(5)
    data = response.read()
    conn.close()

    def get_to_mcs():
        host = "http://api.mediatek.com"
        endpoint = "/mcs/v2/devices/" + deviceId + "/datachannels/ledswitch/datapoints"
        url = host + endpoint
        headers = {
            "Content-type": "application/json",
            "deviceKey": deviceKey
        }
        r = requests.get(url, headers=headers)
        value = (r.json() ["dataChannels][0]["dataPoints][0]["values]["value"]
        return value

    def readLDR(PIN):
        reading=0
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.output(PIN, False)
        time.sleep(0.1)
        GPIO.setup(PIN, GPIO.IN)
        while (GPIO.input(PIN)==False):
            reading=reading+1
        return reading

    def readLDR(PIN):
        reading=0
        GPIO.setup(PIN, GPIO.OUT)
        GPIO.output(PIN, False)
        time.sleep(0.1)
        GPIO.setup(PIN, GPIO.IN)
        while (GPIO.input(PIN)==False):
            reading=reading+1
        return reading

    # Post MediaTek Cloud Sandbox (MCS)
    while True:
        ldr_reading = readLDR(ledPin)
        print(ldr_reading)
        payload = {
            "datapoints": ["ldr","values":str(ldr_reading)]
        }
        post_to_mcs(payload)
        if(get_to_mcs()==1):
            print("LED turning on.")
            GPIO.output(ledPin,GPIO.HIGH)
            time.sleep(0.5)
        if(get_to_mcs()==0):
            print("LED turning off.")
            GPIO.output(ledPin,GPIO.LOW)
            time.sleep(0.5)
        time.sleep(5)
Assignment 2 - Specification

• Objectives:
  • IoT with “Your” Choices of Sensors and Actuators
  • Python on Raspberry Pi and Javascript on MCS
  • Storage in MCS
  • Computing in MCS

• Upload to E3 before 11/30 at 23:59PM
  • Assignment 2 – deliverables
    • Report (2-4 pages)
      • Explain the objectives
      • Explain the specification of sensors and actuators used
      • Explain the system design
      • Explain your python and javascript source codes
        • The differences with example codes
        • The detail of how your scripts can post and get the data from MCS to your sensors and actuators
    • Source Codes
    • 3-minute demo video (just the URL of video)
    • Report can be written in Chinese, and for Video, it must be delivered in English or with English Caption
    • Zip the above files into one compressed file and upload
  • Q&A? Post on E3 discussion board
Frequent Issues in the 1-Page Proposals

Common issues in the 1-page proposals:
- defocused: stay in main theme
- too many/few sensors/actuators
- not related functions
- lab2: no MCS/cloud
- lab3: no 2-devices (Pi)
- too much: weeks/months ⇒ days
Thank You