Data Flow Webinars

Session 1
6.10
Data stream from Retail and Manufacturing


IoT Device


IoT Hub


Online data


Event Hub Kafka API


History data


Azure Storage

Session 2
8.10
Data transformation
Data engineering


Service Bus


Synapse Data Warehouse


Stream Analytics


CosmosDB


Data Factory

Session 3
13.10
Stream processing
Real time visualization


SendGrid


Azure Function


App Service


SignalR


App Insight

Session 4
15.10
Machine Learning and cognitive services


Azure Databricks


HDInsight


Synapse workspace


Azure Machine Learning


Azure Data Explorer


Cognitive Services

Session 5
20.10
Data visualisation and reporting

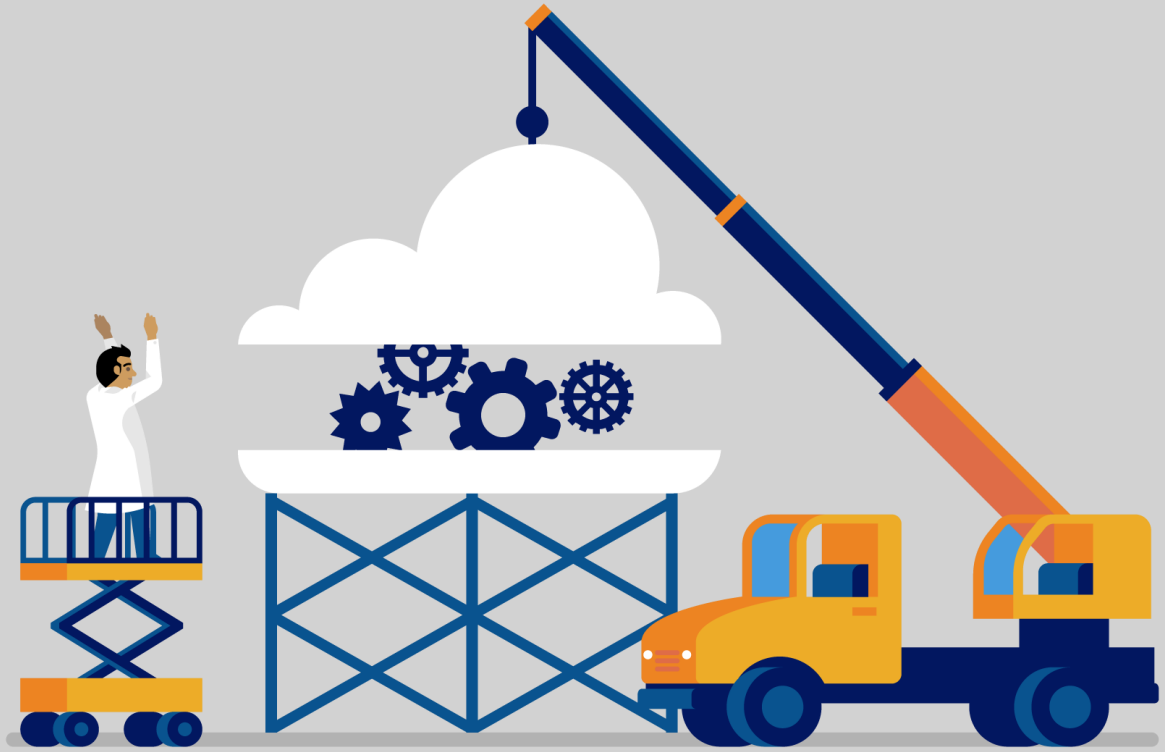
PoweBI


PowerBI Embedded

Microsoft

IoT Hardware and messages. Azure Included

Tomasz Kopacz



October 2020

Two main tools

IoT Hub

Device to cloud messages

Cloud to devices

Security per device

Provisioning

Device Twins

Price per performance (msgs/day)

Protocols:

- MQTT | HTTP | AMQP

Preview:

- Device Streams (incl. TCP/IP protocols)
- SSH Proxy
- RDP Proxy

Time Series Insights (Gen2), PAYG

Retention-based warm and near infinite BYO cold stores.

Azure Blob and Azure Data Lake Gen2 storage.

Apache Parquet file format on cold storage.

Time Series model to contextualize IoT data

Azure IoT Hub and Event Hubs

A first-class web experience.

Rich query APIs PowerBI connector.

For: Debug, fast analysis, dig

Price per stream of data (GB)

SDK - https://github.com/Azure/azure-iot-sdks

Azure IoT SDK for C

written in ANSI C (C99) for portability and broad platform compatibility. There are two device client libraries for C, the low-level iothub_ll_client and the iothub_client (threaded).

Azure IoT SDK for Python

Azure IoT SDK for Node.js

Azure IoT SDK for Java

Azure IoT SDK for .NET

But: HTTP Protocol (REST) + MQTT

And – Device Streams (any TCP/IP protocol!). Sample

"Demo"

C# + IoT Hub – simple, really!

```
clt = DeviceClient.CreateFromConnectionString(connectionStrings[arg.DeviceId], TransportType.Mqtt ); //MQTT or MQTT over WebSocket
await clt.SetDesiredPropertyUpdateCallbackAsync(desiredTwins, null);
await clt.SetMethodHandlerAsync("demo01", methodDemo01, null);
await clt.SetMethodHandlerAsync("demo02", methodDemo02, null);


string dataBuffer = $"{{\"deviceId\":\"netdevice\",\"dt\":\"{DateTime.UtcNow:yyyy-MM-
ddTHH:mm.sss}\",\"messageId\":{count},\"temperature\":{temperature},\"humidity\":{humidity}}}";

using var eventMessage = new Message(Encoding.UTF8.GetBytes(dataBuffer))
{
    ContentType = "application/json",
    ContentEncoding = Encoding.UTF8.ToString(),
};

eventMessage.Properties.Add("temperaturealert", tempAlert.ToString());
await clt.SendEventAsync(eventMessage);
```

Device Twin

Physical device mapped to digital (json) object

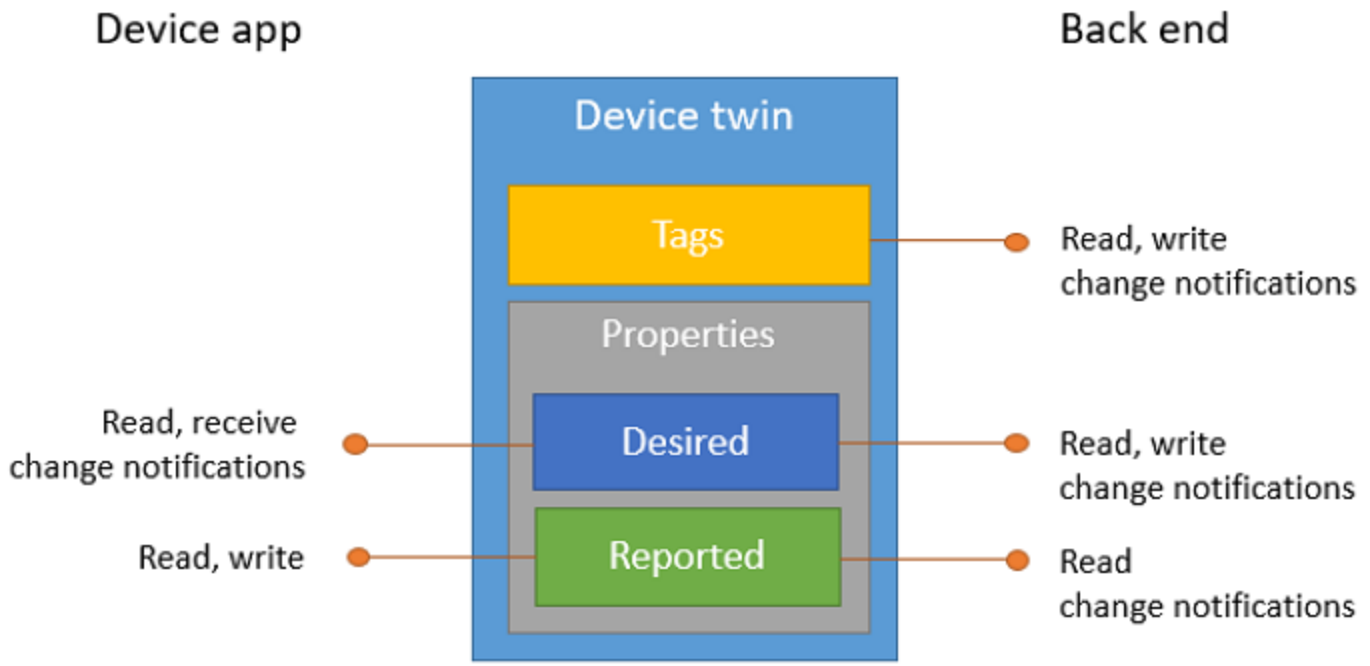
Can be used in query

Execute “that” on devices where tag.color = red

Can be synchronized with physical device

Cloud can “set” device twin

Device can “confirm” (or reject) changes. Or make own!



C# + IoT Hub – twins (simplify!)

```
private static async Task desiredTwins(TwinCollection desiredProperties, object userContext) {

if (desiredProperties.Contains("delayMS")) {
    delayMS = desiredProperties["delayMS"];
    if (delayMS < 1000) {
       //Device limitation!
       delayMS = 1000;
    } }
TwinCollection reportedProperties = new TwinCollection();
foreach (dynamic item in desiredProperties) reportedProperties[item.Key] = item.Value;
reportedProperties["delayMS"] = delayMS;
await clt.UpdateReportedPropertiesAsync(reportedProperties);
}
```

C# + IoT Hub – methods (simplified!)

```
private static async Task<MethodResponse> methodDemo01(MethodRequest methodRequest, object userContext) {
Console.WriteLine("methodDemo01");
return new MethodResponse(
    UTF8Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(
new { State = true, Msg = "OK, methodDemo01" }))
    , 0);
}

private static Task<MethodResponse> methodDemo02(MethodRequest methodRequest, object userContext) {
Console.WriteLine("methodDemo02");
Console.WriteLine(methodRequest.DataAsJson.ToString());
return Task.FromResult(new MethodResponse(
    UTF8Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(
new { State = true, Msg = "OK, methodDemo02" }))
    , 0));
}
```

Demo

IoT Hub, TSI, C# client, Twins

Menu | r4 - Microsoft Azure | TSI Explorer

ms.portal.azure.com/#@microsoft.onmicrosoft.com/resource/subscriptions/e96d15e4-fed1-4c6f-8395-03ebfde5631d/resourceGroups/IoT/providers/Microsoft.Devices/IotHubs/pltkdpepliot2016S1/Overview

C | AZ | Azure Cloud Shell | TSI Explorer | VSO | DR | tomaszkopacz.spac... | Notebooks | SF | M | D | Office 365 | OLD | V | AAD | ADX (demo12.west... | Databricks | Azure IoT Demo M... | Provider: Azure - Te... | JavaScript Window...

Microsoft Azure (Preview) | Report a bug | Search resources, services, and docs (G+/) | tkopacz@microsoft.com MICROSOFT

Dashboard >

pltkdpepliot2016S1

IoT Hub

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Events

Settings

- Shared access policies
- Identity
- Pricing and scale
- Networking
- Certificates
- Built-in endpoints
- Failover
- Properties
- Locks

Explorers

- Query explorer
- IoT devices

Automatic Device Management

- IoT Edge
- IoT device configuration

Messaging

- File upload

Move | Delete | Refresh

Azure IoT Hub and the Azure Device Provisioning Service are updating their TLS certificates starting October 5, 2020 with a new Microsoft Certificate Authority (CA) chained under the existing Baltimore root. If your devices pin certificates, you may need to take action to ensure your devices can continue to connect. Learn more

Essentials JSON View

Resource group (change)	: IoT	Hostname	: pltkdpepliot2016S1.azure-devices.net
Status	: Active	Pricing and scale tier	: S1 - Standard
Current location	: North Europe	Number of IoT Hub units	: 1
Subscription (change)	: DPEPL	Device streams (preview)	: https://db-002.northeurope-001.streams.azure-devices.net
Subscription ID	: e96d15e4-fed1-4c6f-8395-03ebfde5631d		Device streams documentation
Tags (change)	: Click here to add tags		

Need a way to provision millions of devices?

IoT Hub Device Provisioning Service enables zero-touch, just-in-time provisioning to the right IoT hub without requiring human intervention.

Need a way to monitor and secure your IoT solution?

Azure Security for IoT (ASC for IoT) is a unified security management service. It provides end-to-end threat analysis and protection across hybrid cloud workloads and your Azure IoT solution.

Want to learn more about IoT Hub?

Check out IoT Hub documentation. Learn how to use IoT Hub to connect, monitor, and control billions of Internet of Things assets.

We'd love your feedback!

Your valuable feedback will help us to better understand your requirements in order to improve IoT Hub.

Need to simulate IoT Devices?

IoT Device Simulation accelerates solution development using simulated devices to help build and test your project throughout the development lifecycle.

Need to validate device IoT Plug and Play models?

Use IoT Explorer app to view and to validate devices' implementation of IoT Plug and Play models stored locally or models published to the global repo.

Show data for last: 1 Hour | 6 Hours | 12 Hours | 1 Day | 7 Days | 30 Days

Menu | New - Microsoft Azure | TSI Explorer

ms.portal.azure.com/#create/hub

C | AZ | Azure Cloud Shell | TSI Explorer | VSO | DR | tomaszkopacz.spac... | Notebooks | SF | M | D | Office 365 | OLD | V | AAD | ADX (demo12.west... | Databricks | Azure IoT Demo M... | Provider: Azure - Te... | JavaScript Window...

Microsoft Azure (Preview) | Report a bug | Search resources, services, and docs (G+/) | tkopacz@microsoft.com MICROSOFT

Dashboard >

New

Search the Marketplace

Azure Marketplace See all

- Get started
- Recently created
- AI + Machine Learning
- Analytics
- Blockchain
- Compute
- Containers
- Databases
- Developer Tools
- DevOps
- Identity
- Integration
- Internet of Things
- IT & Management Tools
- Media
- Migration
- Mixed Reality
- Monitoring & Diagnostics
- Networking
- Security
- Software as a Service (SaaS)
- Storage
- Web

Featured See all

IoT Central application
Learn more

CrystalBall (preview)
PREVIEW
Learn more

IoT Hub Device Provisioning Service
Quickstarts + tutorials

Azure Digital Twins (Preview) (preview)
PREVIEW
Learn more

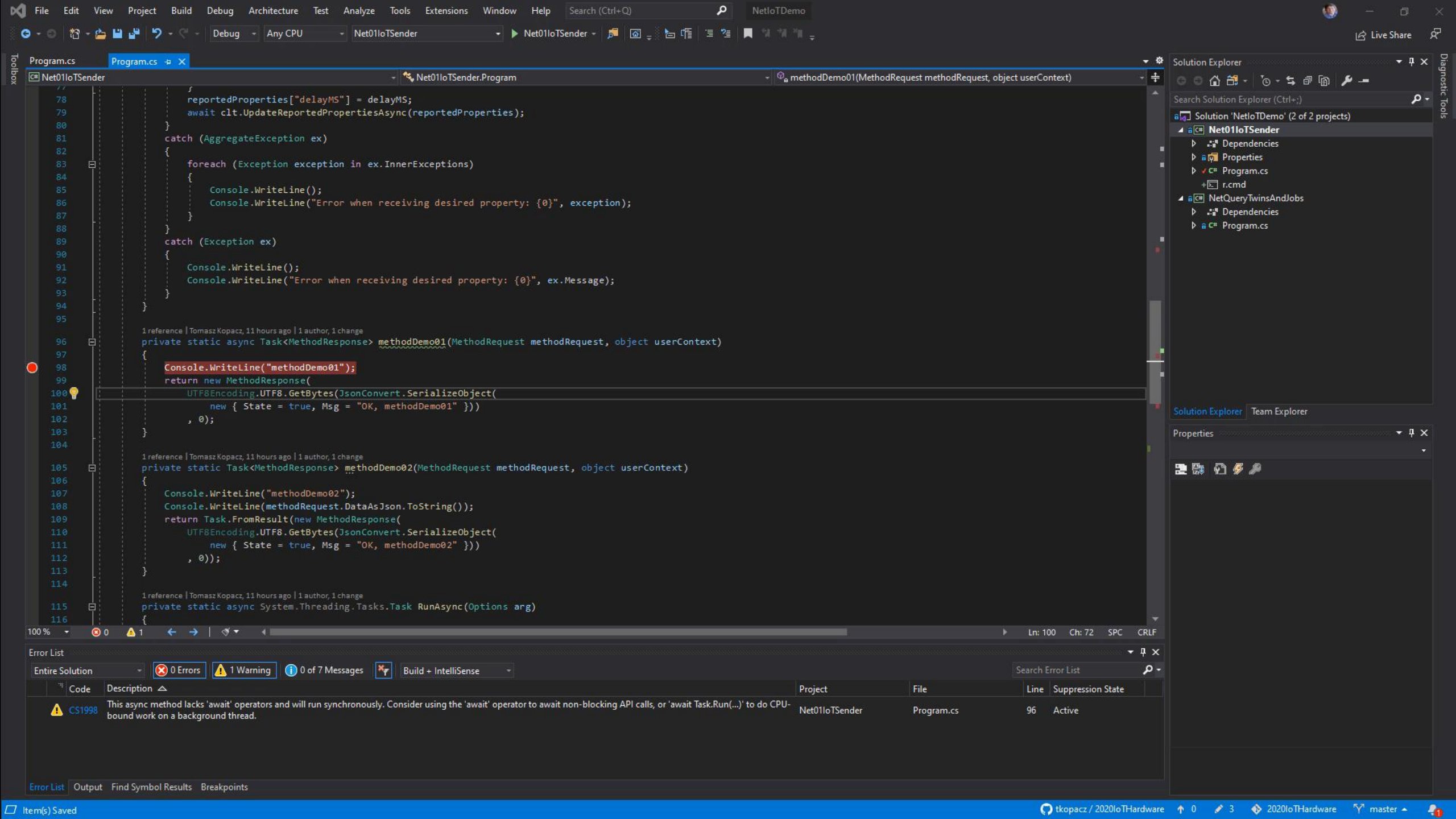
Time Series Insights
Quickstarts + tutorials

Machine Learning Studio (classic) Workspace
Learn more

Azure Stack Edge / Data Box Gateway
Learn more

Event Grid Topic
Learn more

Function App
Quickstarts + tutorials

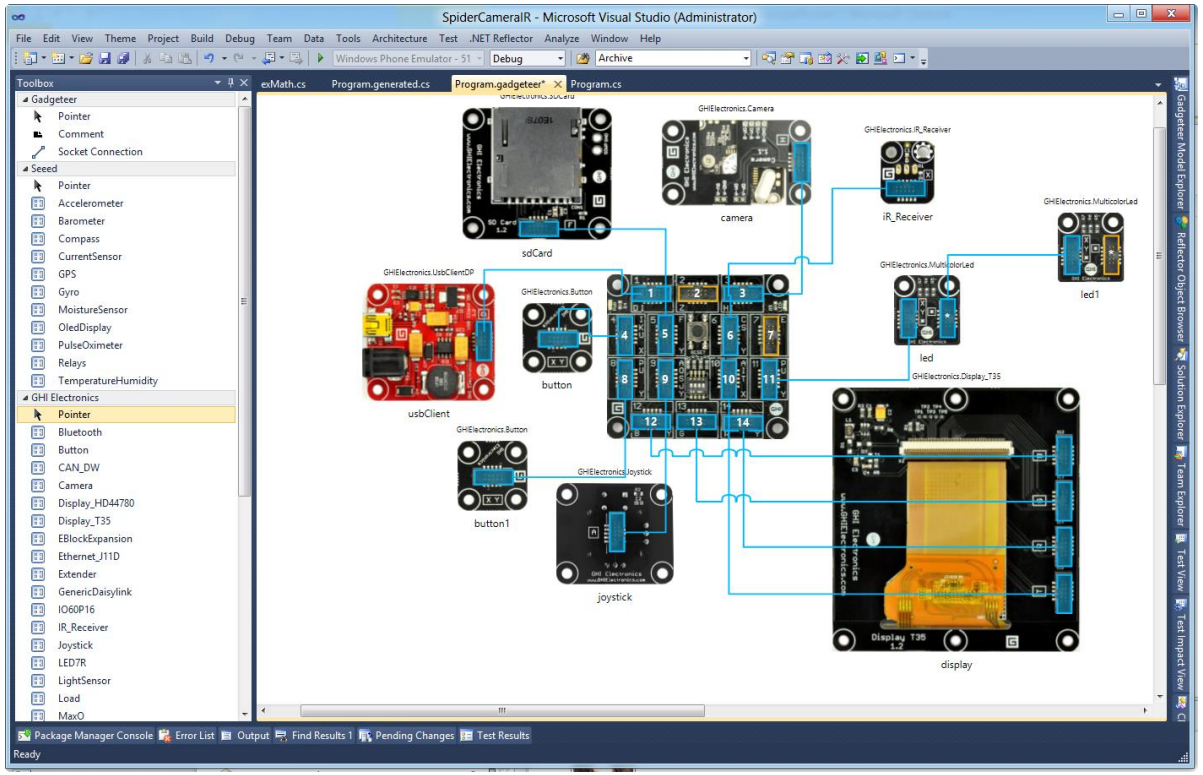

File Edit View Project Build Debug Architecture Test Analyze Tools Extensions Window Help
Search (Ctrl+Q)
NetIoTDemo
Debug
Any CPU
Net01IoTSender
Live Share
Program.cs
Net01IoTSender.Program
methodDemo01(MethodRequest methodRequest, object userContext)
reportedProperties["delayMS"] = delayMS;
await clt.UpdateReportedPropertiesAsync(reportedProperties);
catch (AggregateException ex)
foreach (Exception exception in ex.InnerExceptions)
Console.WriteLine("Error when receiving desired property: {0}", exception);
catch (Exception ex)
Console.WriteLine("Error when receiving desired property: {0}", ex.Message);
1 reference | Tomasz Kopacz, 11 hours ago | 1 author, 1 change
private static async Task<MethodResponse> methodDemo01(MethodRequest methodRequest, object userContext)
Console.WriteLine("methodDemo01");
return new MethodResponse(
UTF8Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(
new { State = true, Msg = "OK, methodDemo01" }))
, 0);
1 reference | Tomasz Kopacz, 11 hours ago | 1 author, 1 change
private static Task<MethodResponse> methodDemo02(MethodRequest methodRequest, object userContext)
Console.WriteLine("methodDemo02");
Console.WriteLine(methodRequest.DataAsJson.ToString());
return Task.FromResult(new MethodResponse(
UTF8Encoding.UTF8.GetBytes(JsonConvert.SerializeObject(
new { State = true, Msg = "OK, methodDemo02" }))
, 0));
1 reference | Tomasz Kopacz, 11 hours ago | 1 author, 1 change
private static async System.Threading.Tasks.Task RunAsync(Options arg)
Ln: 100 Ch: 72 SPC CRLF
Solution Explorer
Search Solution Explorer (Ctrl+;)
Solution 'NetIoTDemo' (2 of 2 projects)
Net01IoTSender
Dependencies
Properties
Program.cs
r.cmd
NetQueryTwinsAndJobs
Dependencies
Program.cs
Solution Explorer
Team Explorer
Properties
Error List
Entire Solution
0 Errors
1 Warning
0 of 7 Messages
Build + IntelliSense
Search Error List
Code
Description
Project
File
Line
Suppression State
CS1998
This async method lacks 'await' operators and will run synchronously. Consider using the 'await' operator to await non-blocking API calls, or 'await Task.Run(...)' to do CPU-bound work on a background thread.
Net01IoTSender
Program.cs
96
Active
Error List
Output
Find Symbol Results
Breakpoints
Item(s) Saved
tkopacz / 2020IoTHardware
master

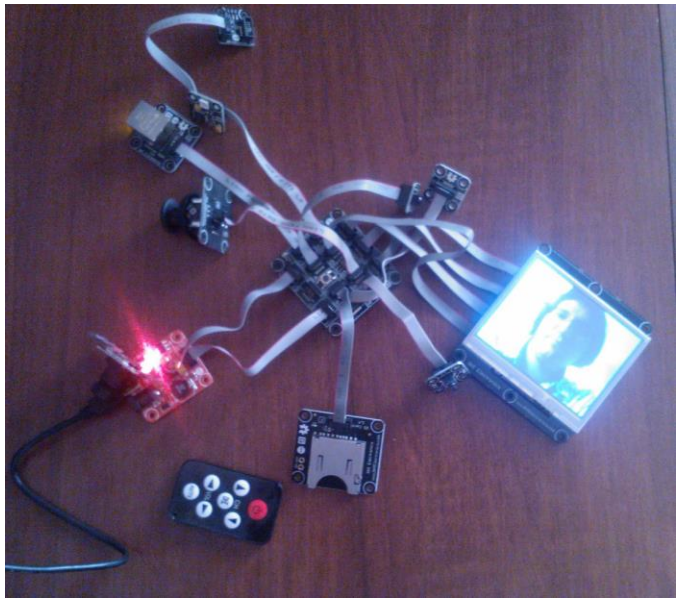
For .NET fans (1/3)

Microsoft SPOT -> .NET Micro Framework (.NET Gadgeteer – driver's model)

Smart Personal Objects Technology (2002)

DirectBand R.I.P 2008
MSN Direct - 2012





https://github.com/NETMF
Last update: 2018

For .NET fans (2/3)

GHI Electronics – TinyCLR OS = .NET MF *Next* - Gadgeteer

STM ST32++

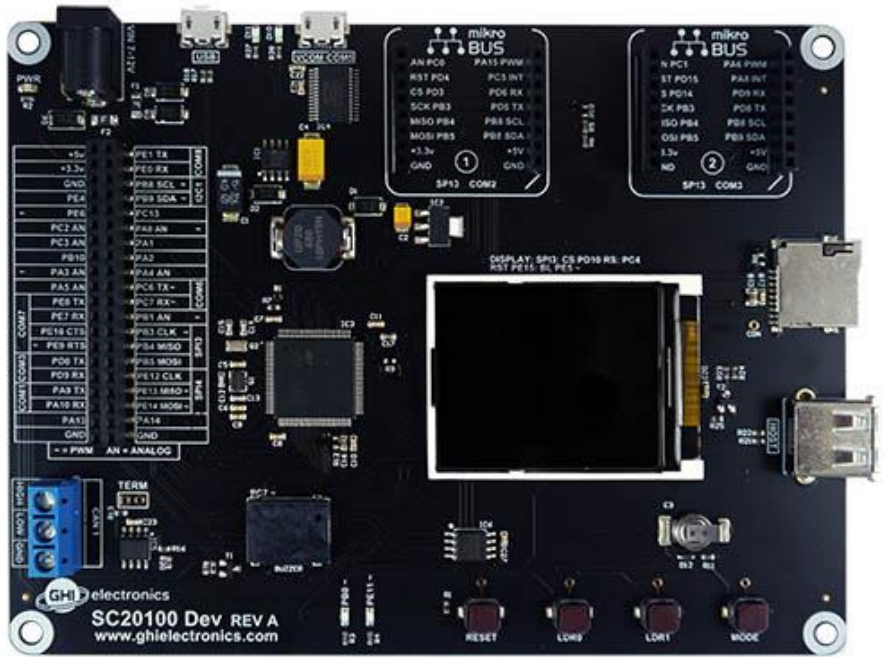
- TinyCLR OS 1.0

ARM Cortex M7

- TinyCLR OS 2.0
- (ex: 163 GPIO)

Short:

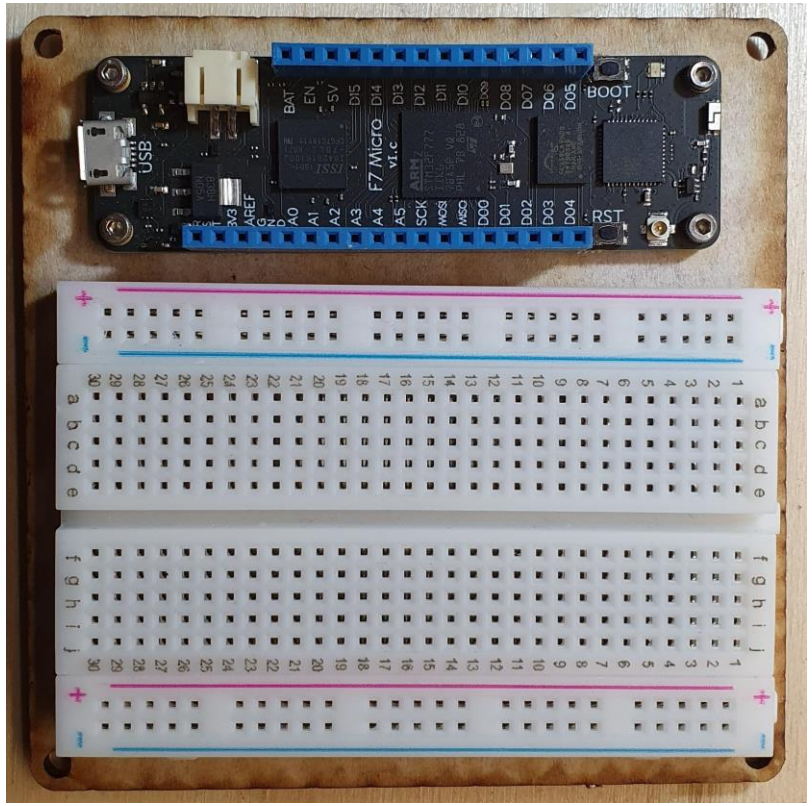
- C#
- Interrupt as event
- “driver model”

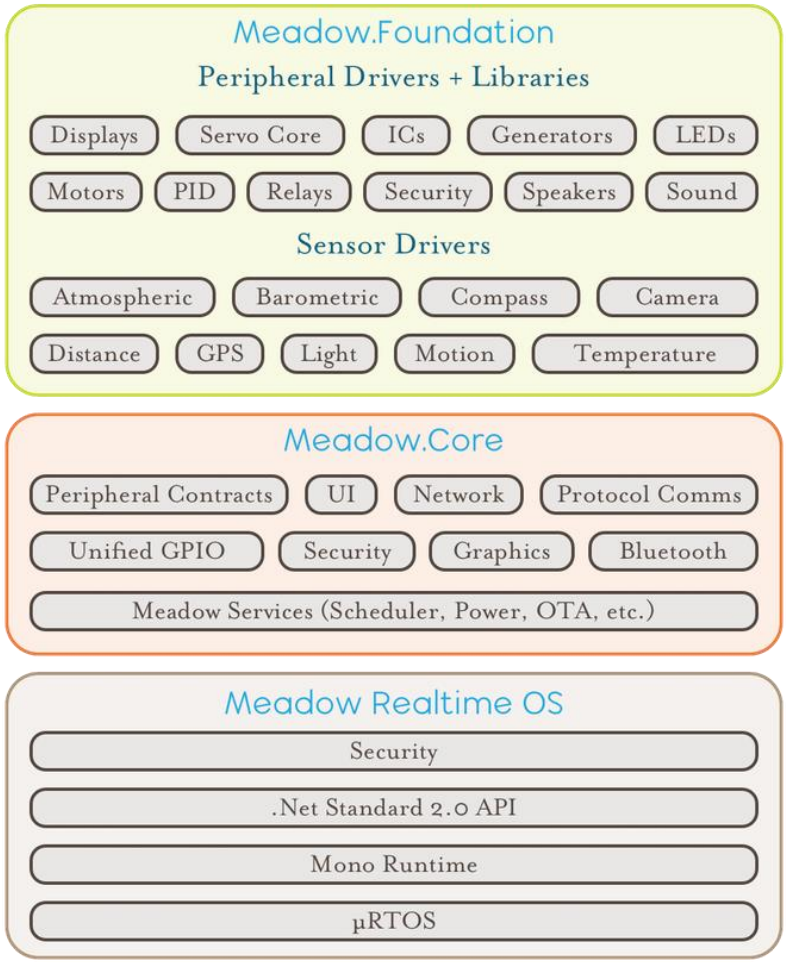


(ca 200$)
SoC $19.95 / 100

For .NET fans (3/3) - .NET Standard - Meadow

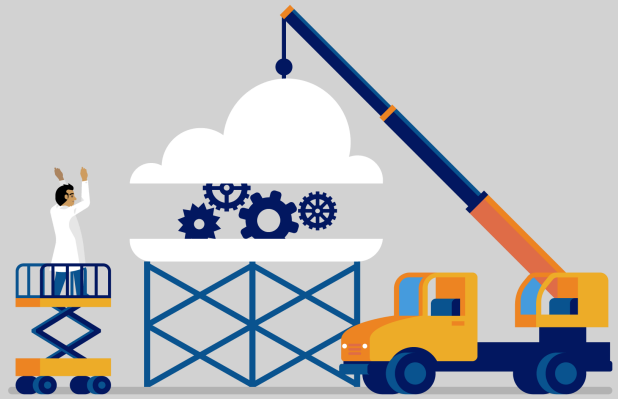
STM32 + ESP32 (~~very~~ beta)

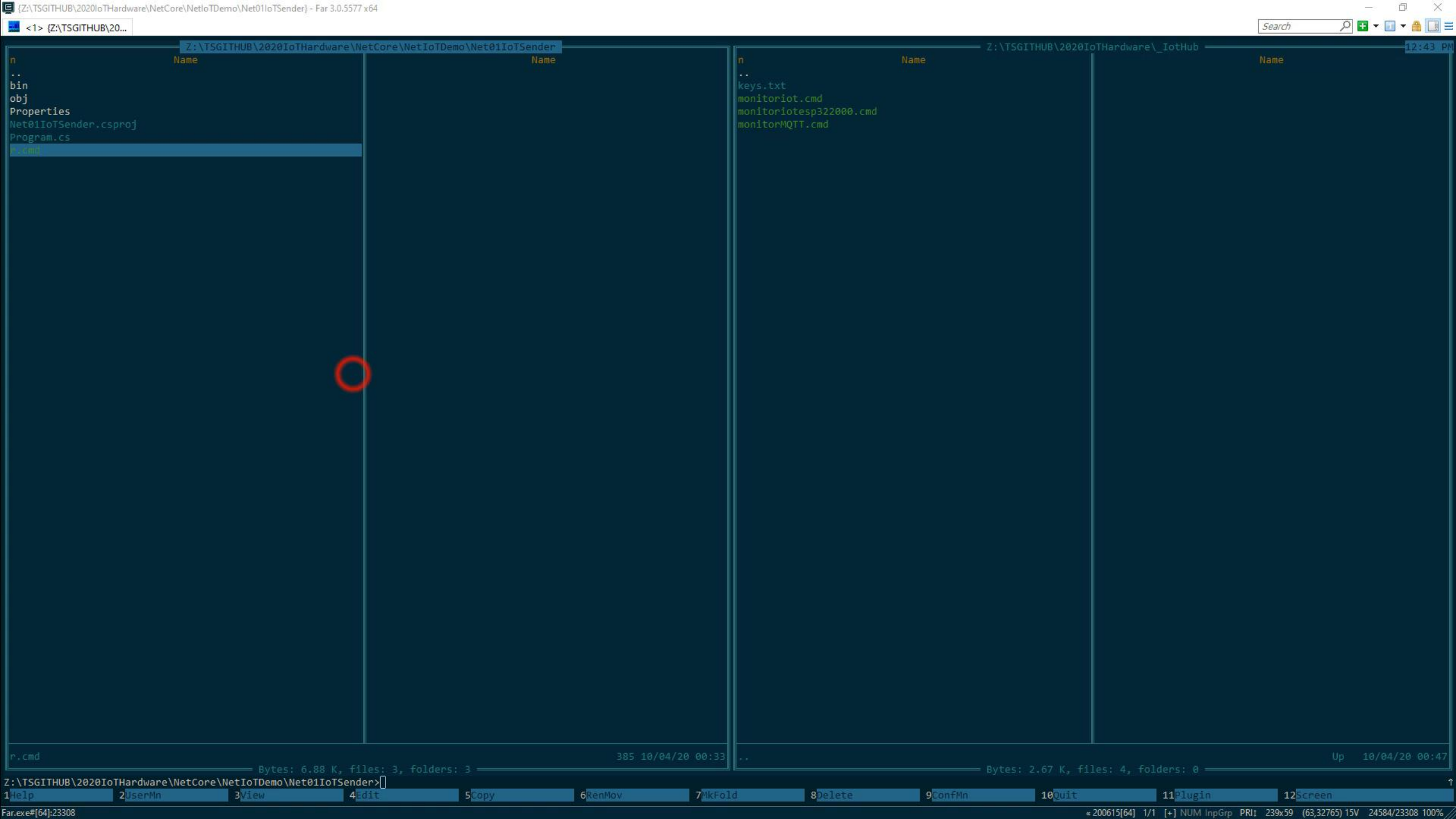




Demo

Jobs, Queries and C#
Managing devices from code




{Z:\TSGITHUB\2020IoTHardware\NetCore\NetIoTDemo\Net01IoTSender} - Far 3.0.5577 x64
<1> {Z:\TSGITHUB\20...
Search
Z:\TSGITHUB\2020IoTHardware\NetCore\NetIoTDemo\Net01IoTSender
n Name
Name
..
bin
obj
Properties
Net01IoTSender.csproj
Program.cs
r.cmd
r.cmd
385 10/04/20 00:33
Bytes: 6.88 K, files: 3, folders: 3
Z:\TSGITHUB\2020IoTHardware_IotHub
12:43 PM
n Name
Name
..
keys.txt
monitoriot.cmd
monitoriotesp322000.cmd
monitorMQTT.cmd
..
Up 10/04/20 00:47
Bytes: 2.67 K, files: 4, folders: 0
Z:\TSGITHUB\2020IoTHardware\NetCore\NetIoTDemo\Net01IoTSender>
1Help 2UserMn 3View 4Edit 5Copy 6RenMov 7MkFold 8Delete 9ConfMn 10Quit 11Plugin 12Screen
Far.exe#[64]:23308
« 200615[64] 1/1 [+] NUM InpGrp PRI↕ 239x59 (63,32765) 15V 24584/23308 100%

Message Routing + Custom Endpoints + Enrich messages

Routing based on properties! Not body

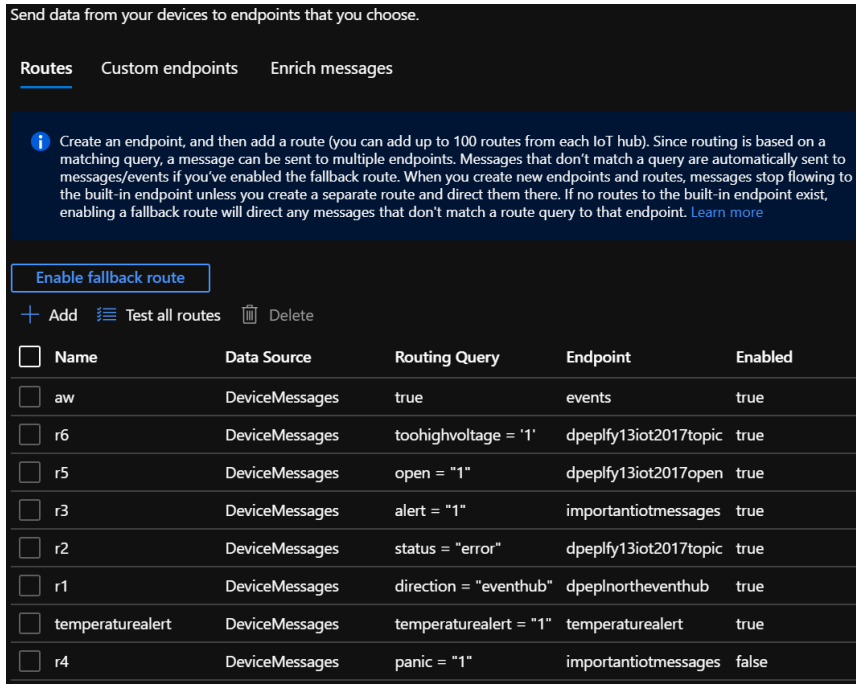
Custom endpoints

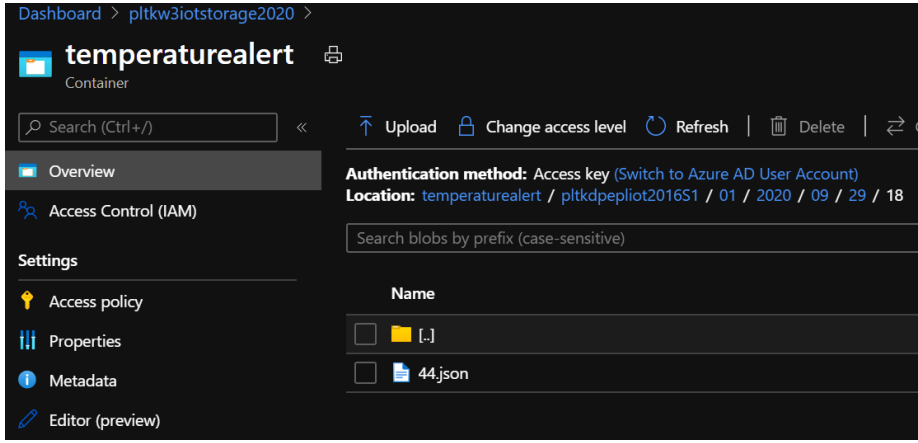
- Built-in endpoint
- Azure Storage
- Service Bus Queues and Service Bus Topics
- Event Hubs

Enrich messages sent to endpoint

Digression – routing based on message body

- Stream Analytics
- Azure Function (or code) attached to Endpoint





Reading from IoT Hub

Consumer Group – "cursor" for data stream, per app (max 20/IoT Hub)

Ordering guarantees with at least once delivery: [1,2,1,2,3,1,2,3,4] -> if you ever receive message [1], it would always be followed by [2,3,4].

How to process:

- Azure Functions. Processing data from IoT Hub with Azure Functions.
- Azure Stream Analytics. Stream data as input into Stream Analytics.
- Time Series Insights. Add an IoT hub event source to your Time Series Insights environment.
- Apache Storm spout. You can view the spout source on GitHub.
- Apache Spark integration.
- Azure Databricks.

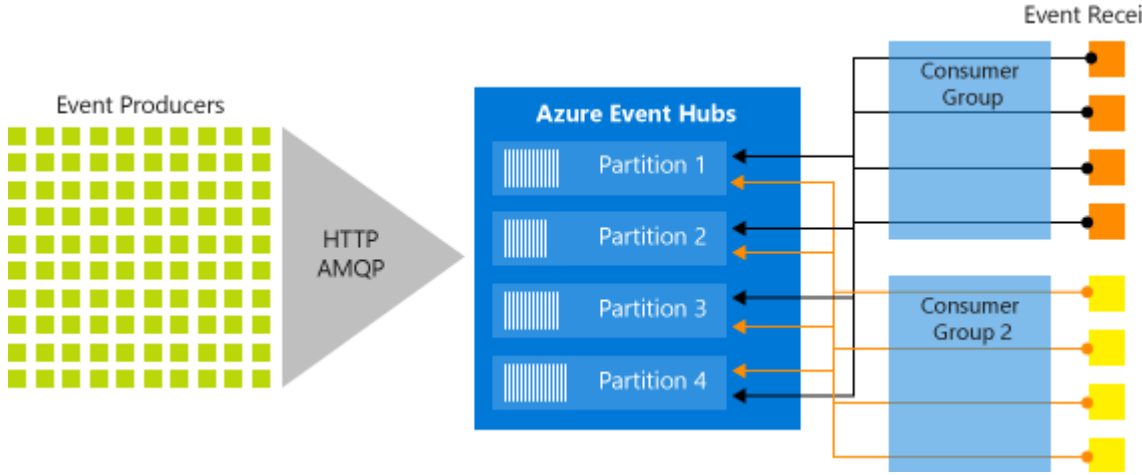
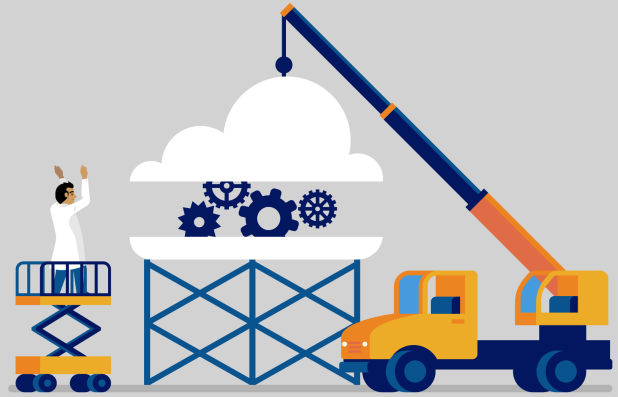


Image from Event Hub docs – IoT Hub use Event Hub, but partitions are derived from device id

```
[FunctionName("IoTReaderV3")]
public static async Task Run([EventHubTrigger("pltkdpepliot2016s1", Connection = "eh", ConsumerGroup = "fun1")]
    EventData[] events, ILogger log, Microsoft.Azure.WebJobs.ExecutionContext context)
{
    foreach (EventData eventData in events)
    {
        messageBody = Encoding.UTF8.GetString(eventData.Body.Array, eventData.Body.Offset, eventData.Body.Count) + "\n";
        sb.Append(messageBody);
    }
```

Hardware



Simplified view on hardware (processor + WiFi stack)

"Microcontrolers", MCU – RAM + CPU + FLASH + IO

- ATMEL ("Arduino")
- **ESP8266** (2014)
- ESP32-S2 (2020)
- **ESP32** (2016)
- STM32 (+something - ESP32 in case of Meadow board)
- Nordic nRF52840 (Cortex™-M4 + WiFi)
- Hybrid – Azure Sphere (RTOS (CortexM4) + ARM Cortex A7 ("normal" os)), Secure Smart device

"MiniPC"

- Raspberry PI (ARM, Cortex-A72, **UK**) | Orange PI (ARM, AllWinner H3, **CN**) | BeagleBone (ARM, Cortex-A8, **US**)
- Up-Board (x86, Asus)
- Lattepanda (x64 + Arduino)

Frameworks (+OS)

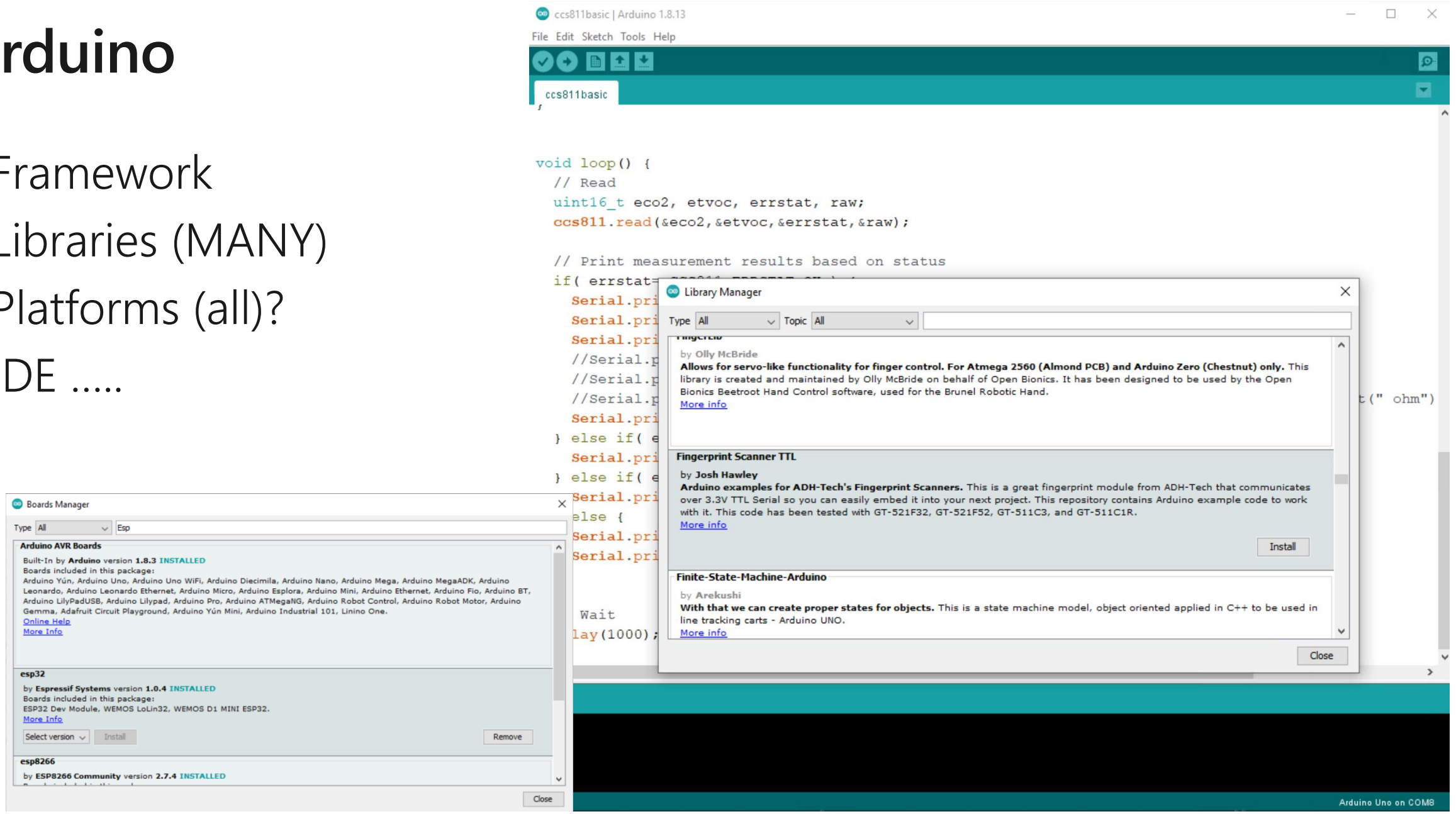
- FreeRTOS
- Rt-Thread
- Azure RTOS

- Mbed
- Zephyr RTOS (Freescale Kinetis, NXP)
- ESP-IDF

- Many “makers” platforms, tools have binding to “Arduino”
- “OS” – well… (miniPC)

Arduino

- Framework
- Libraries (MANY)
- Platforms (all)?
- IDE .....



Tools

Free / almost free

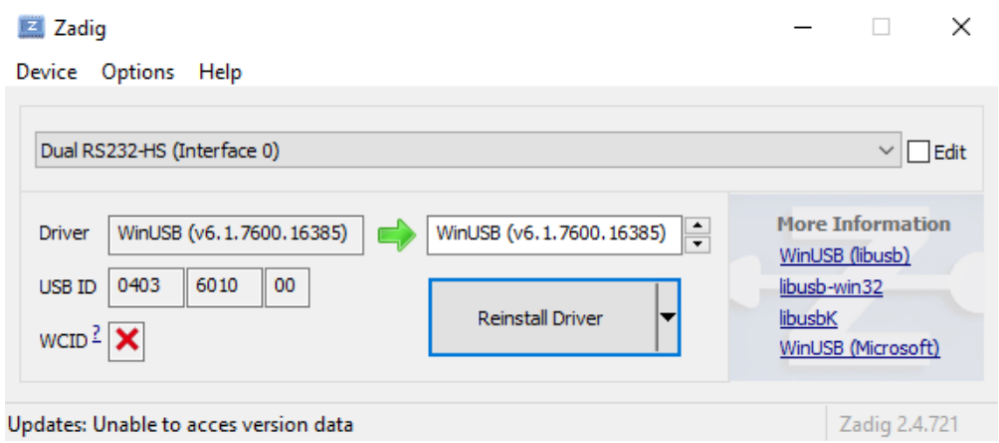
- **VS Core + Platform IO**
- Arduino
- (Eclipse)
- GNU Assembler (GAS)
- VisualMicro

Pro (costly!)

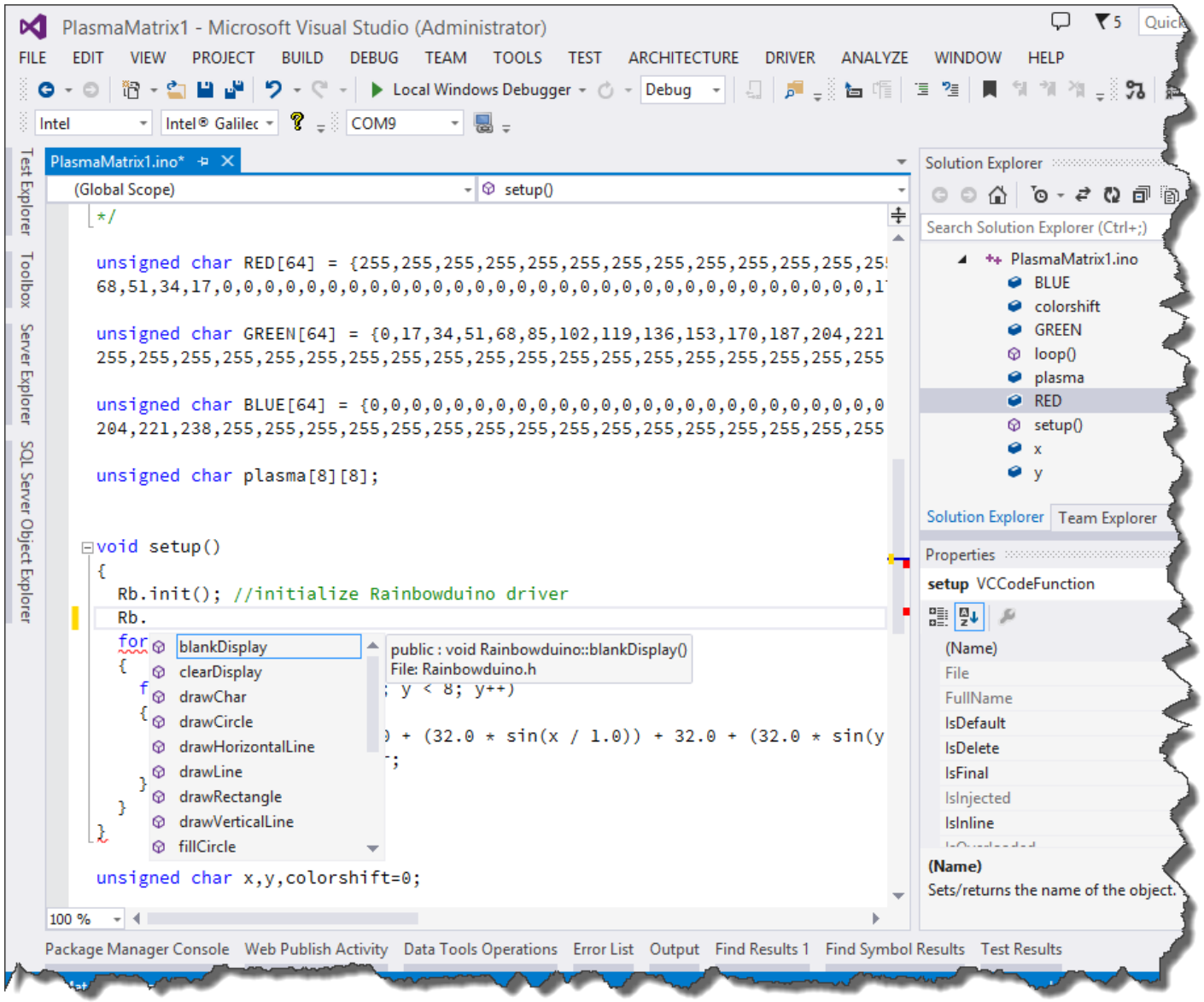
- ARM Keil (MDK Core – small)
  - Keil C51
- Segger JLink (Edu – cheap)
- VisualGDB

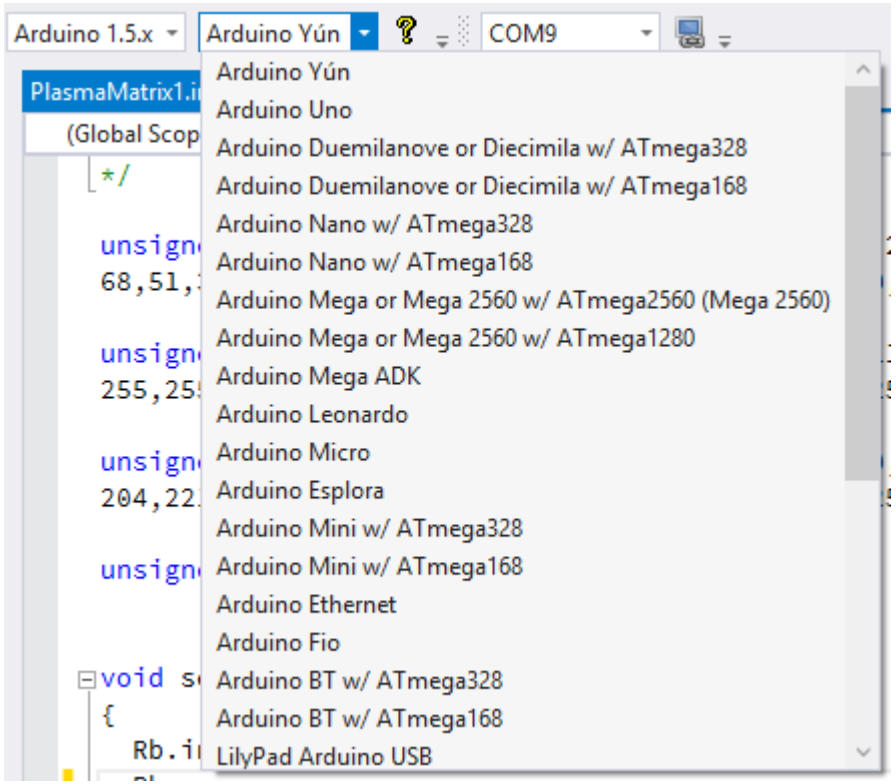
Tech Challenges

1. Debugging
2. Logging
3. Updating software
4. How to connect to “WiFi”
5. Power
6. USB ☺



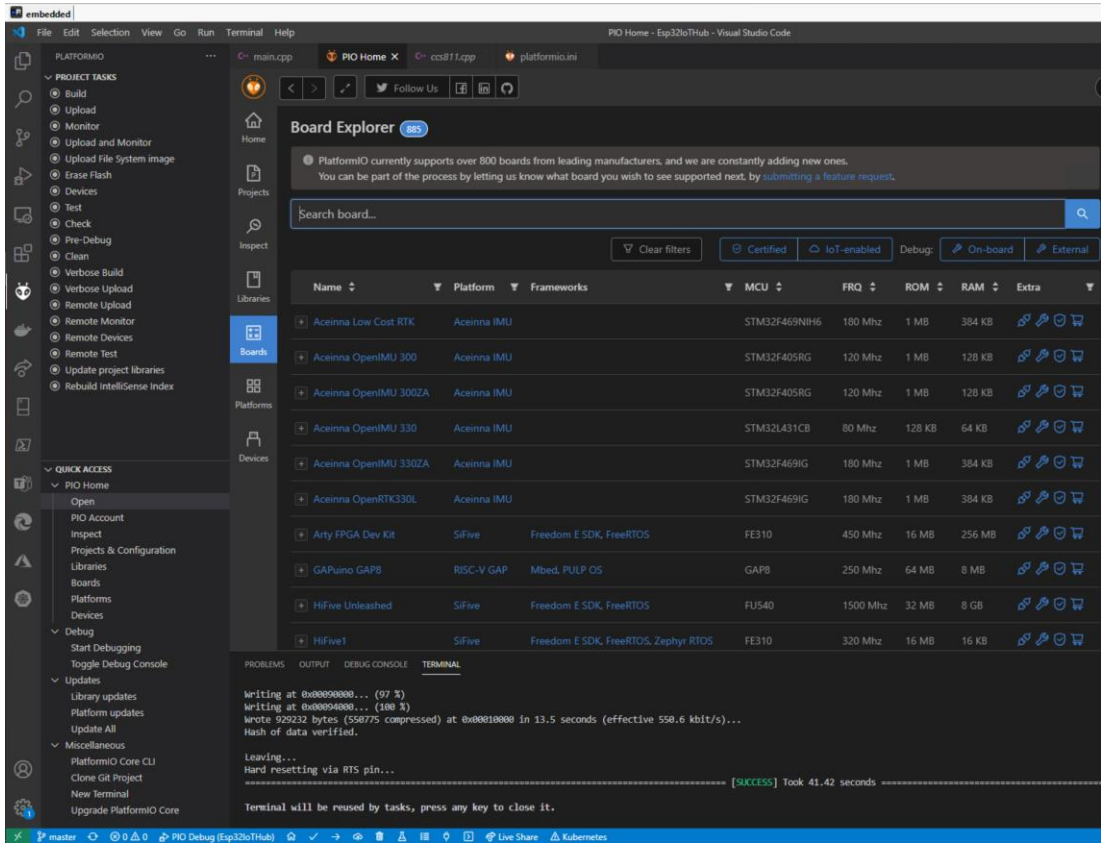
Visual Micro





Platform IO

Platforms + libs



Auto dependencies!

```
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 41 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <AzureIoTHub> 1.3.9
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |   |-- <src>
|   |   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <AzureIoTSocket_WiFi> 1.0.0
|   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |   |-- <src>
|   |   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <AzureIoTProtocol_MQTT> 1.3.9
|   |-- <AzureIoTHub> 1.3.9
|   |   |-- <AzureIoTUtility> 1.3.9
|   |   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |   |   |-- <src>
|   |   |   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |   |-- <src>
|   |   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <AzureIoTProtocol_HTTP> 1.3.9
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |   |-- <src>
|   |   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <AzureIoTUtility> 1.3.9
|   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|   |-- <src>
|   |   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <src>
|   |-- <src@src-22da4c01638ad979a3ed95b8542b4318>
|-- <ArduinoJson> 6.16.1
|-- <CCS811> 10.0.0
|   |-- <Wire> 1.0.1
|-- <ClosedCube HDC1080> 1.3.2
|   |-- <Wire> 1.0.1
|-- <Wire> 1.0.1
Building in release mode
```

Debugger!

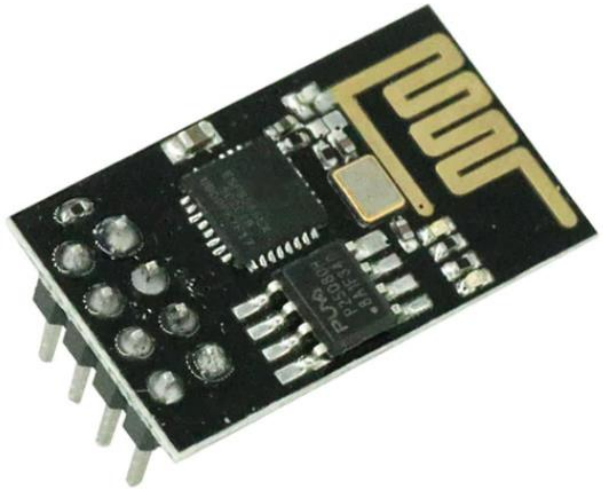
- Altera / Intel USB-Blaster
- Atmel-ICE
- Black Magic Probe
- CMSIS-DAP
- JTAG-HS1
- ESP-Prog
- FTDI Chip
- GD-LINK
- oddWires IOT-Bus JTAG
- J-LINK
- Mini-Module FT2232H
- MSP Debug
- Olimex ARM-USB-OCD-H
- Olimex ARM-USB-OCD
- Olimex ARM-USB-TINY-H
- Olimex ARM-USB-TINY
- QEMU
- Renode
- RV-LINK
- simavr
- Sipeed RV Debugger
- ST-LINK
- TI-ICDI
- TIAO USB Multi-Protocol Adapter (TUMPA)
- UM232H
- Verilator
- Custom

ESP8266 and ESP32

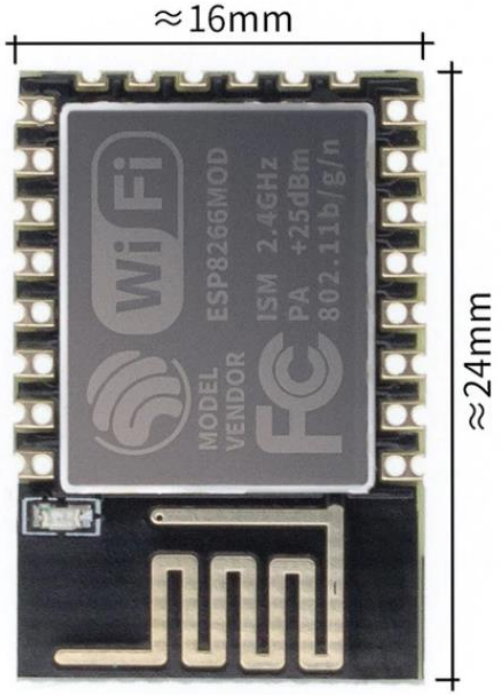
WiFi + BT + 1 or 2 cores

(major I/O protocols)

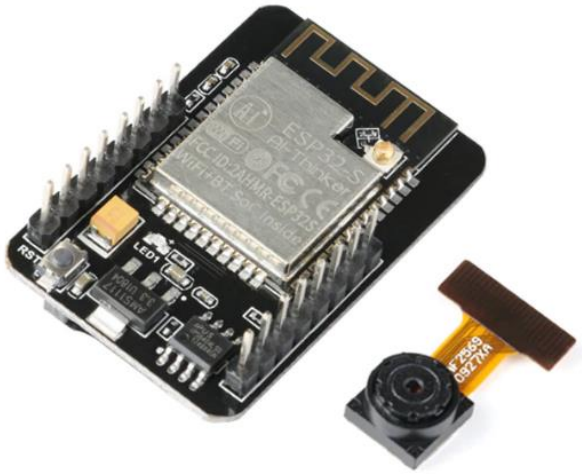
(AP mode, WiFi “modem”) | ESP-IDF | Arduino



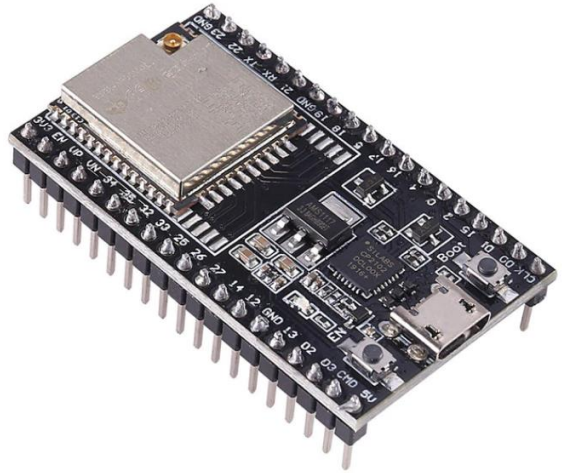
ESP-01, **$1.12**
($3 in 2015)
2 GPIO



ESP-12E, **$1.09**
11 GPIO + 1 ADC

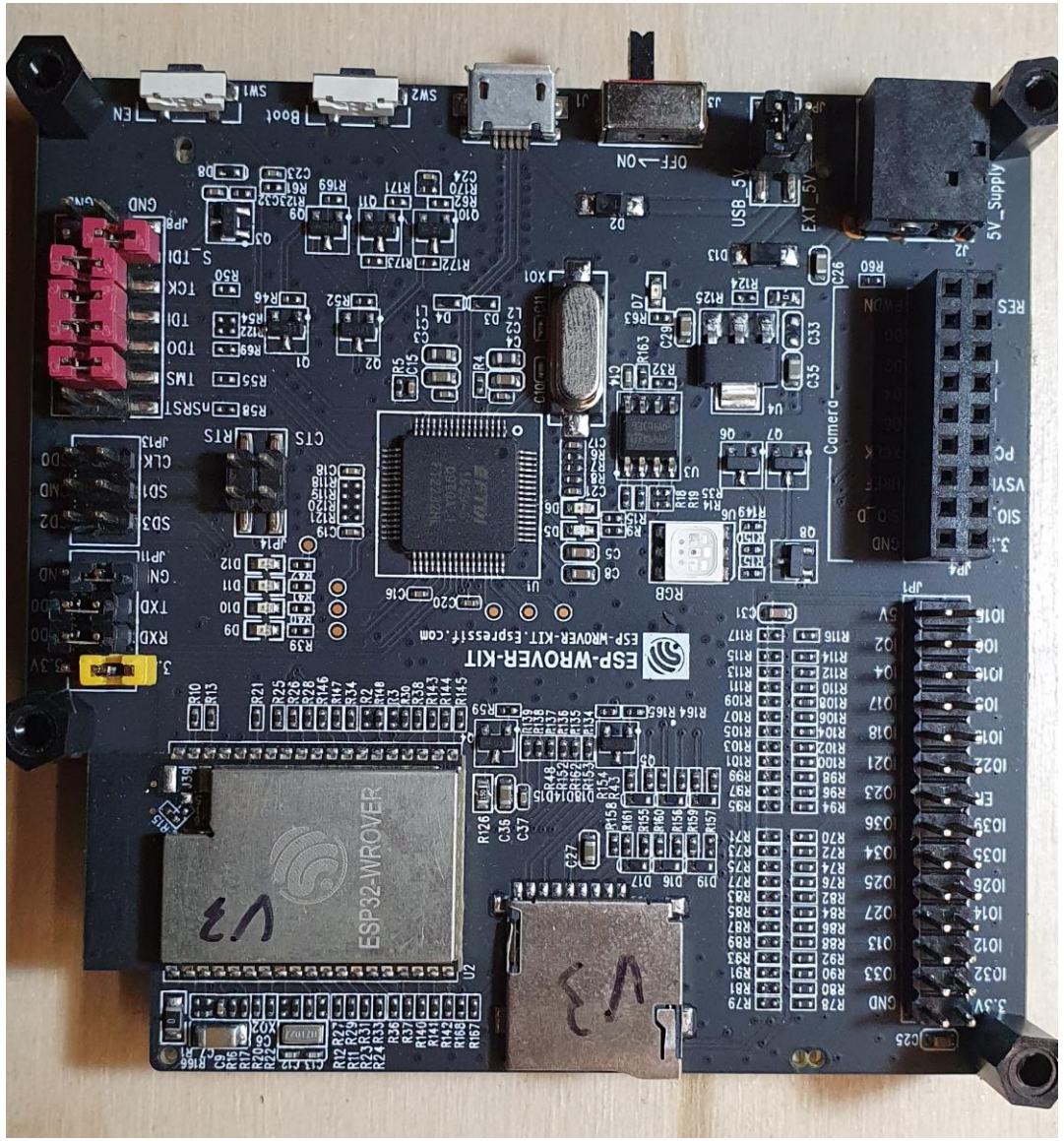


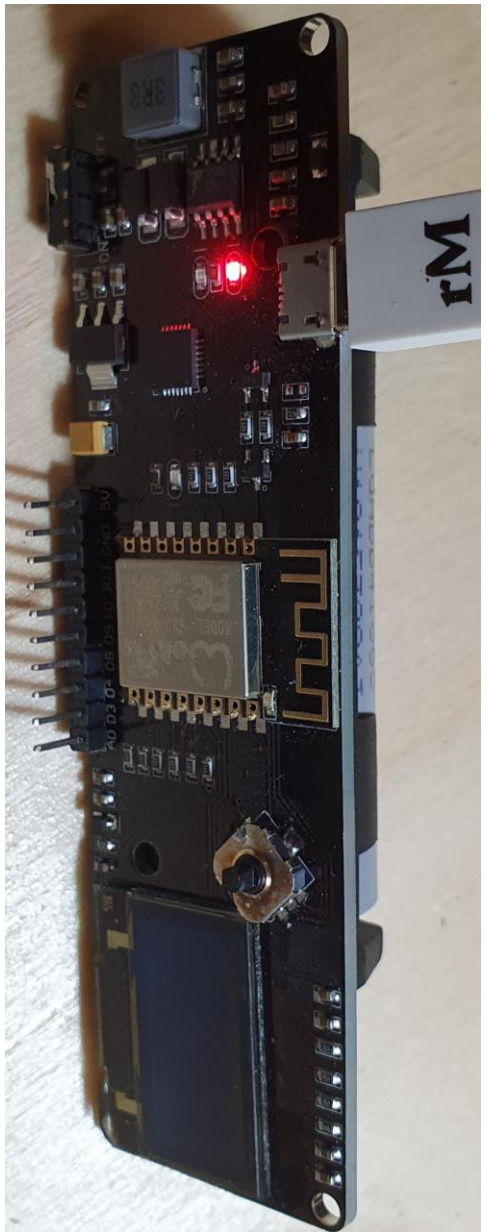
ESP32 + CAM **$5.75**
2 cores, 11 GPIO + 1 ADC



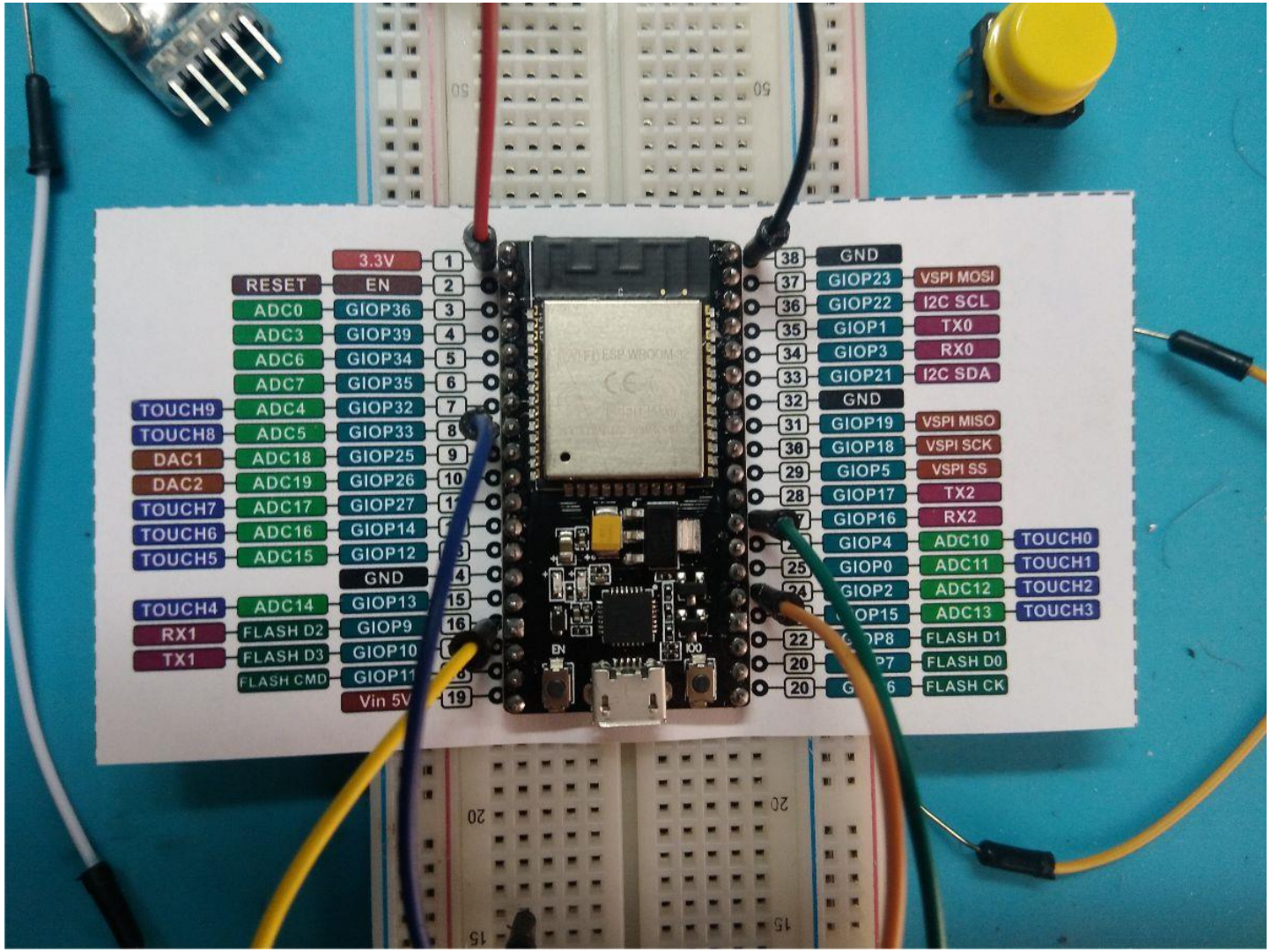
ESP32 **$3.89 ($1.3)**
2 cores, 11 GPIO + 1 ADC

My two “main” dev devices





PINS


3.3V
RESET
ADC0
ADC3
ADC6
ADC7
TOUCH9
TOUCH8
DAC1
DAC2
TOUCH7
TOUCH6
TOUCH5
TOUCH4
RX1
TX1
EN
GIOP36
GIOP39
GIOP34
GIOP35
ADC4
ADC5
ADC18
ADC19
ADC17
ADC16
ADC15
GND
ADC14
FLASH D2
FLASH D3
FLASH CMD
GIOP32
GIOP33
GIOP25
GIOP26
GIOP27
GIOP14
GIOP12
GIOP13
GIOP9
GIOP10
GIOP11
Vin 5V
GND
GIOP23
GIOP22
GIOP1
GIOP3
GIOP21
GND
GIOP19
GIOP18
GIOP5
GIOP17
GIOP16
GIOP4
GIOP0
GIOP2
GIOP15
GIOP8
GIOP7
GIOP6
VSPI MOSI
I2C SCL
TX0
RX0
I2C SDA
VSPI MISO
VSPI SCK
VSPI SS
TX2
RX2
ADC10
ADC11
ADC12
ADC13
FLASH D1
FLASH D0
FLASH CK
TOUCH0
TOUCH1
TOUCH2
TOUCH3

M5Stack – “gadgeteer like”

Connect peripherials?

- Wired (link layer, network buses !)
  - “Serial port”, UART, RS 232 (±5:±15(25)V, DB-25, 9 pin), RS 485 (±1.5V:±6V), up to 1.2 KM
  - I2C (1982), I3C (2006)
  - SPI
  - 1-Wire
  - SMBus, PMBus
  - ModBus (for PLCs, free, 1979!, still in use!)
  - CAN / CanBus (Bosh, 1983)
- LPWAN – **low-power wide-area network** (**LPWAN**) or **low-power wide-area** (**LPWA**) network or **low-power network** (**LPN**)
  - WiFi as extension
  - LORA

(What to do with connection)

PIR, digital temperature - GPIO

Analog temperature, light - (ADC) Analog or SPI for high performance measurements

Stepper (unipolar|bipolar) – driver like ULN2003A (+ GPIO)

DC Motor – PWM + L298 (or similar) H bridge (+ GPIO)

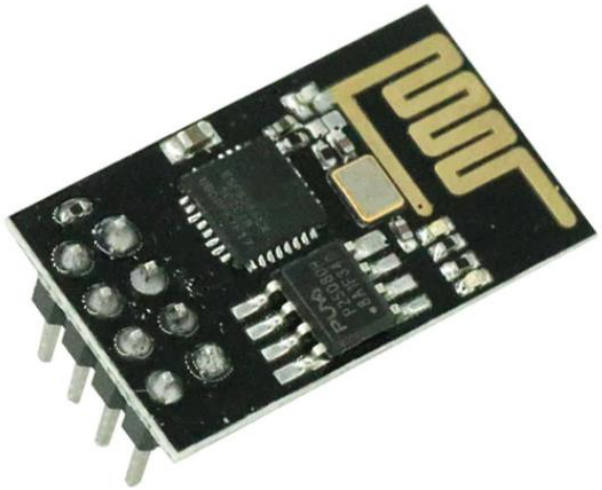
Servo motor – usually PWM (pulse-width modulation) (+ GPIO)

Complex motor drivers – I2C/SPI

Camera – Serial (slow, cheap) | custom 8bit

LCD – SPI/I2C

First „demo” – how to connect to WiFi?



WPS (vulnerabilities… 2012)
SmartConfig (*WiFi.beginSmartConfig();* ESP specific)
Texas Instument (**CC3200**) – another SmartConfig (different magic package!)
Touch&Connect (few routers)
(Bluetooth, BLE GATT based)
(good solution – SD Card with WiFi credentials)
(THE WORST SOLUTION – ~~hardcoded credentials in the code~~)

Universal Solution:

1. Local AP without password
2. Run mini WebServer on device
3. User connect to AP, use browser to enter connection details. Save them(?)
4. (device reset)
5. (device will connect to normal AP)

Technical challenges – save data

EEPROM – write cycles limited!

Or:

Use SD Card and read WiFi AP + password data from there

- We can prepare SD cards upfront!

```
bool ESPPreferences::writeAll() {
    m_lastByteIndex = 0;
    //Version
    EEPROM.begin(m_size);
    EEPROM.write(0,m_version);
    EEPROM.write(m_size-1,m_version);
    m_lastByteIndex = 4;
    EEPROM.write(m_lastByteIndex++,m_lastIndex);
    for (int8_t i = 0; i < m_lastIndex; i++)
    {
        if (m_lastByteIndex>=(m_size-2)) break;
        int l = m_keys[i].length();
        for (int j = 0; j < l; j++)
        {
            if (m_lastByteIndex>=(m_size-2)) break;
            char b = m_keys[i][j];
            EEPROM.write(m_lastByteIndex++,b);
        }
        EEPROM.write(m_lastByteIndex++,0);
        l = m_values[i].length();
        for (int j = 0; j < l; j++)
        {
            if (m_lastByteIndex>=(m_size-2)) break;
            char b = m_values[i][j];
            EEPROM.write(m_lastByteIndex++,b);
        }
        //Not mistake - last "zero"
        if (m_lastByteIndex>=(m_size-1))  break;
        EEPROM.write(m_lastByteIndex++,0);
    }
    EEPROM.end(); //commit, physical write
    return true;
}
```

```
bool ESPPreferences::readAll() {
    m_lastIndex = m_lastByteIndex = 0;
    //Check Version
    EEPROM.begin(m_size);
    int val1 = ((int)EEPROM.read(0));
    int val2 = ((int)EEPROM.read(m_size-1-0));
    if (val1==m_version && val2==m_version) {
        m_lastByteIndex = 4;
        int datacnt = ((int)EEPROM.read(m_lastByteIndex++));
        //Serial.println(datacnt);
        for (byte i = 0; i < datacnt; i++)
        {
            String key,value;
            //Reading key
            while(m_lastByteIndex < (m_size - 1)){
                char b = EEPROM.read(m_lastByteIndex++);
                if (b==0) break;
                key+=b;
            }
            //Reading value
            while(m_lastByteIndex < (m_size - 1)){
                char b = EEPROM.read(m_lastByteIndex++);
                if (b==0) break;
                value+=b;
            }
            m_values[m_lastIndex] = value;
            m_keys[m_lastIndex] = key;
            m_lastIndex++;
        }
    }
    EEPROM.end();
    return true;
}
```

“Web Server” and processing

```
#include <WiFiClient.h>
#include "WiFiClientSecure.h"

#ifdef ESP8266
    #include <ESP8266WiFi.h>
    #include <ESP8266mDNS.h>
    #include <ESP8266WebServer.h>
    #include "ESP8266Preferences.h"
#else
    #include <WiFi.h>
    #include "WebServer.h"
#endif
```

1) Includes, libraries

```
void WWWESPSetupServer::setupMode()
{
  WiFi.disconnect();
  delay(100);
  WiFi.mode(WIFI_STA);
  delay(100);
  WiFi.disconnect();
  delay(100);
  int n = WiFi.scanNetworks();
  delay(100);
  for (int i = 0; i < n; ++i)
  {
    ssidList += "<option value=\"";
    ssidList += WiFi.SSID(i);
    ssidList += "\">";
    ssidList += WiFi.SSID(i);
    ssidList += "</option>";
  }
  delay(100);
  WiFi.softAPConfig(apIP, apIP, IPAddress(255, 255, 255, 0));
  WiFi.softAP(apSSID);
  WiFi.mode(WIFI_AP);
  startWebServer();
}
```

2) Get SSID
3) Setup AP

```
webServer.on("/settings", [this]() {
  String s = "<h1>Settings</h1><p>Please select WiFi network,
  s += "<form method=\"GET\" action=\"set\"><br><b>WiFi connec
  s += ssidList;
  s += "</select>";
  s += "<br><label>Password:</label><input name=\"WIFI_PASSWD\
  s += "<br><input type=\"submit\"></form>";
  s += "<p>Or for advanced users, http://192.168.4.1/set?WIFI_
  //http://192.168.4.1/set?WIFI_SSID=5GBEMOWO&WIFI_PASSWD=MESH
  //http://192.168.4.1/set?WIFI_SSID=5GBEMOWO&WIFI_PASSWD=MESH
  webServer.send(200, "text/html", makePage("Wi-Fi Settings",
});
webServer.on("/set", [this]() {
  Config::WIFI_SSID = urlDecode(webServer.arg("WIFI_SSID"));
  Config::WIFI_PASSWD = urlDecode(webServer.arg("WIFI_PASSWD")
  Config::DEVICE_NAME = urlDecode(webServer.arg("DEVICE_NAME")
```

```
webServer.begin();
```

```
void loop() {
  if (www.IsSettingsMode())
  {
  }
  else
  {
    loopIoTHub();
  }
  www.HandleClient();
}
```

4) Setup WWW
5) Loop (no threads!)

```
void WebServer::handleClient() {
  if (_currentStatus == HC_NONE) {
    WiFiClient client = _server.available();
    if (!client) {
      return;
    }

    log_v("New client");

    _currentClient = client;
    _currentStatus = HC_WAIT_READ;
    _statusChange = millis();
  }

  bool keepCurrentClient = false;
  bool callYield = false;

  if (_currentClient.connected()) {
    switch (_currentStatus) {
    case HC_NONE:
      // No-op to avoid C++ compiler warning
      break;
    case HC_WAIT_READ:
      // Wait for data from client to become a
      if (_currentClient.available()) {
        if (_parseRequest(_currentClient)) {
          // because HTTP_MAX_SEND_WAIT is exp
          // it must be divided by 1000
          _currentClient.setTimeout(HTTP_MAX_S
          _contentLength = CONTENT_LENGTH_NOT_
          _handleRequest();

          if (_currentClient.connected()) {
```

5.1)Real loop on sockets

Wstring.h is you friend!

Actual time! Critical for TLS session, SAS generation - SNTP

```
#define MIN_EPOCH (40 * 365 * 24 * 3600)
void WWWESPSetupServer::initTime()
{
  time_t epochTime;

  configTime(0, 0, "pool.ntp.org", "time.nist.gov");

  while (true)
  {
    epochTime = time(NULL);

    if (epochTime < MIN_EPOCH)
    {
      Serial.println("Fetching NTP epoch time failed! Waiting 2 seconds to retry.");
      delay(2000);
    }
    else
    {
      Serial.printf("Fetched NTP epoch time is: %ld \r\n", epochTime);
      break;
    }
  }
}
```

```
void configTime(long gmtOffset_sec, int daylightOffset_sec, cons
{
    if(sntp_enabled()){
        sntp_stop();
    }
    sntp_setoperatingmode(SNTP_OPMODE_POLL);
    sntp_setservername(0, (char*)server1);
    sntp_setservername(1, (char*)server2);
    sntp_setservername(2, (char*)server3);
    sntp_init();
    setTimeZone(-gmtOffset_sec, daylightOffset_sec);
}
```

Some devices will have an RTC clock with memory. But – accuracy – varies!

“Secure” SSL on WiFi

```
WiFiClientSecure wifissl;
```

Client(s)

```
char certificates[1283] =
/* Baltimore */
"-----BEGIN CERTIFICATE-----\r\n"
"MIIDdzCCAl+gAwIBAgIEAgAAuTANBgkqhkiG9w0BAQUFADBaMQswCQYDVQQGEwJJ\r\n"
"RTESMBAGA1UEChMJQmFsdGltb3JlMRMwEQYDVQQLEwpDeWJlclRydXN0MSIwIAYD\r\n"
"VQQDExlCYWx0aW1vcmUgQ3liZXJUcnVzdCBSb290MB4XDTAwMDUxMjE4NDYwMFoX\r\n"
"DTI1MDUxMjIzNTkwMFowWjELMAkGA1UEBhMCSUUxEjAQBgNVBAoTCUJhbHRpbW9y\r\n"
"ZTETMBEGA1UECxMKQ3liZXJUcnVzdDEiMCAGA1UEAxMZQmFsdGltb3JlIEN5YmVy\r\n"
"VHJ1c3QgUm9vdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAKMEuyKr\r\n"
"mD1X6CZymrV51Cni4eiVgLGw41uOKymaZN+hXe2wCQVt2yguzmKiYv60iNoS6zjr\r\n"
"IZ3AQSsBUnuId9Mcj8e6uYi1agnnc+gRQKfRzMpijS3ljwumUNKoUMMo6vWrJYeK\r\n"
```

Hard-coded certificate….
Different for „normal” Azure, gov, cn, …

```
#ifdef ESP8266
    wifissl.setCACert((const uint8_t*)certificates,sizeof(certificates));
#else
    wifissl.setCACert((const char*)certificates);
#endif
```

Bind certificate per client

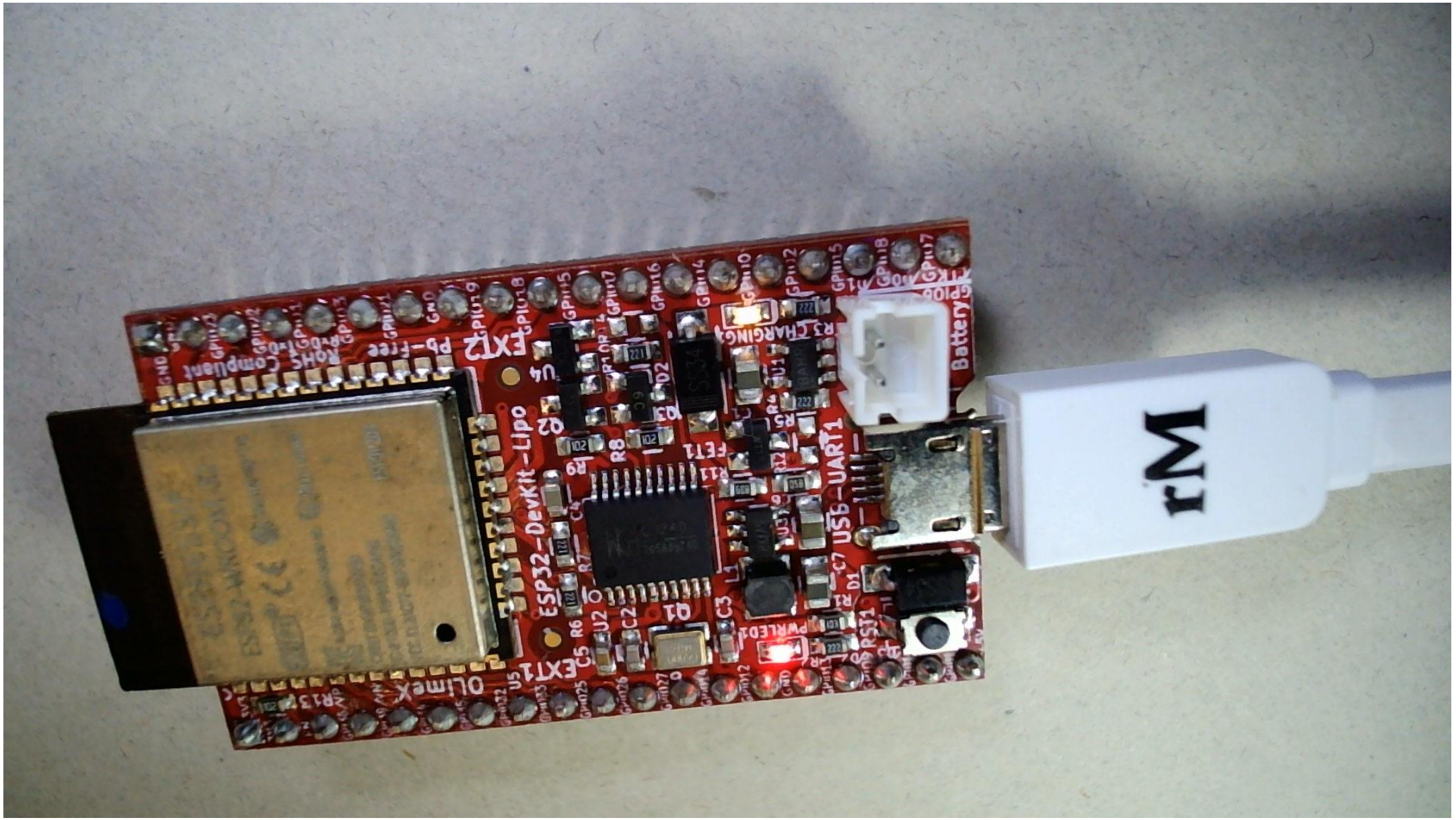
```
m_clientIoT.setCACert(certificates);
m_clientBlob.setCACert(certificates);
```

This is per domain (cached internally)

```
HTTPClient https;
https.begin(m_clientBlob, uploadURL.c_str());
https.addHeader("x-ms-blob-type", "BlockBlob");
```

Use with HTTPClient here

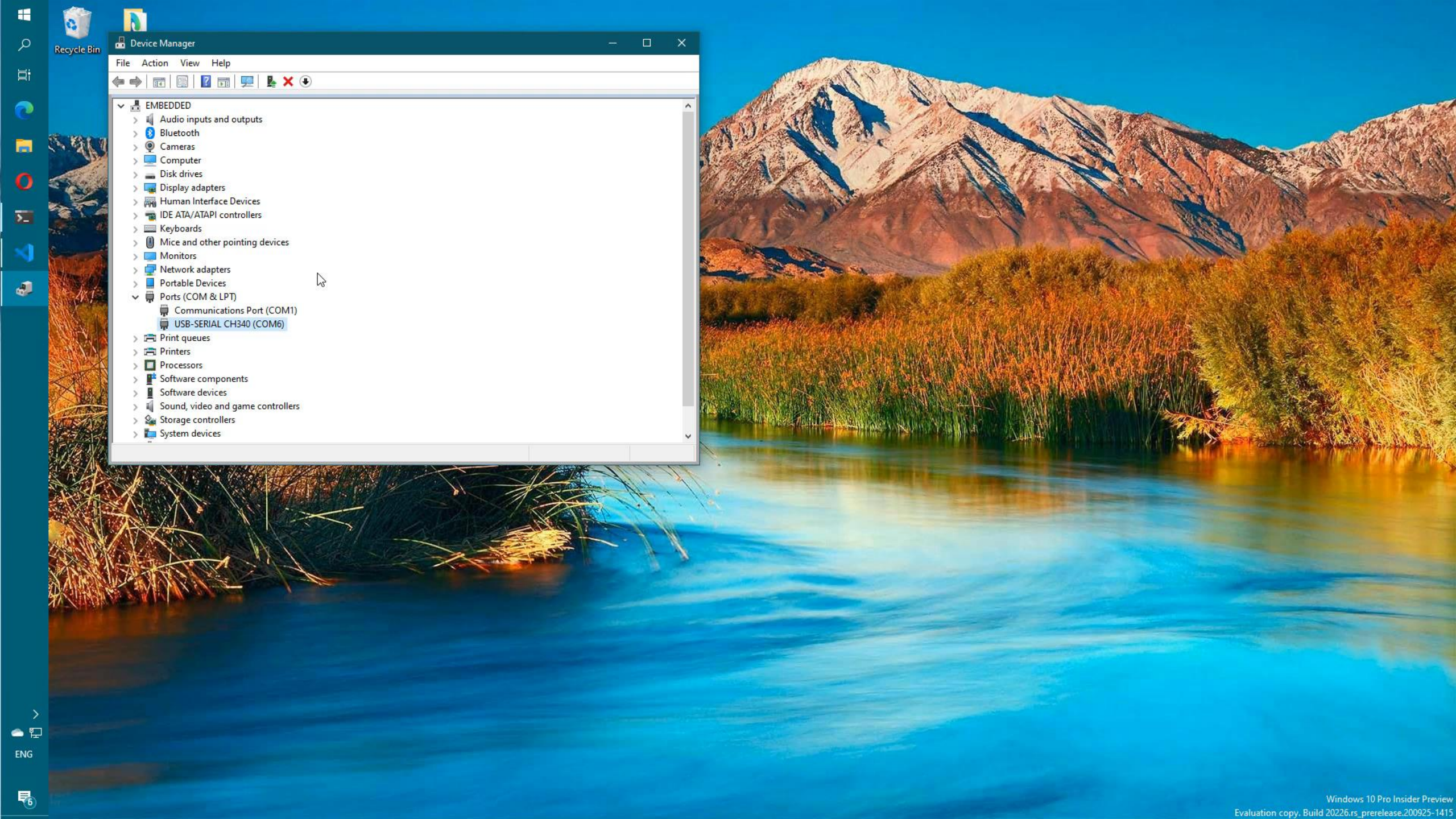
This is first device ESP32-DevKit-LiPo



Demo

ESP32 and WIFI

Z:\TSGITHUB\2020IotHardware\ESP32\Esp32WifiIoTBlob


Recycle Bin
Device Manager
File
Action
View
Help
EMBEDDED
Audio inputs and outputs
Bluetooth
Cameras
Computer
Disk drives
Display adapters
Human Interface Devices
IDE ATA/ATAPI controllers
Keyboards
Mice and other pointing devices
Monitors
Network adapters
Portable Devices
Ports (COM & LPT)
Communications Port (COM1)
USB-SERIAL CH340 (COM6)
Print queues
Printers
Processors
Software components
Software devices
Sound, video and game controllers
Storage controllers
System devices
ENG
Windows 10 Pro Insider Preview
Evaluation copy. Build 20226.rs_prerelease.200925-1415

IoT Hub – C99 library …

AzureIoTHub
AzureIoTProtocol_HTTP
AzureIoTProtocol_MQTT
AzureIoTSocket_WiFi
AzureIoTUtility

(WiFi)
(Certificates)
(Clock)
No Blobs (large messages) on small devices – but SAS … (next demo)

Demo

ESP32 and (WIFI) and Arduino IDE

master · 4 branches · 16 tags · Go to file · Code

jbobotek update to v1.3.9 — 3306566 on Jul 22 — 116 commits

examples	update to v1.3.9	2 months ago
src	update to v1.3.9	2 months ago
.travis.yml	Update to v1.3.8	8 months ago
LICENSE	Update AzureIoTHub for Arduino using the latest azure-iot-sdks version	4 years ago
README.md	Update to v1.3.8	8 months ago
keywords.txt	Sync Arduino libraries with latest Azure IoT SDK 1.0.46	2 years ago
library.properties	update to v1.3.9	2 months ago

README.md

This project has adopted the Microsoft Open Source Code of Conduct. For more information see the Code of Conduct FAQ or contact opencode@microsoft.com with any additional questions or comments.

Azure IoT Hub Library source files

AzureIoTHub - Azure IoT Hub library for Arduino

This is the location of the Arduino-specific source files for the

AzureIoTHub Arduino published library.

This library is a port of the Microsoft Azure IoT device SDK for C to Arduino. It allows you to use several Arduino compatible boards with Azure IoT Hub. Please submit any contribution directly to azure-iot-sdks.

Currently supported hardware:

- ESP8266 based boards with esp8266/arduino
  - SparkFun Thing

About

Azure IoT library for the Arduino

Readme

View license

Releases

16 tags

Packages

No packages published

Contributors 13

+ 2 contributors

Languages

C 98.6% C++ 1.4%

Platform IO - config

```
; https://docs.platformio.org/page/projectconf.html
[common]
pio_libdir = C:/Users/tkopa/.platformio/packages/framework-arduinoespressif32/libraries

[env:esp-wrover-kit]
platform = espressif32
board = esp-wrover-kit
framework = arduino
upload_port = COM[4]
build_flags =
    -DCORE_DEBUG_LEVEL=ARDUHAL_LOG_LEVEL_VERBOSE
    -DDONT_USE_UPLOADTOBLOB
    -DESP32
    -DARDUINO_ARCH_ESP32
    -DUSE_BALTIMORE_CERT
    -w
    -IInclude
    -I${common.pio_libdir}/WiFi/src
    -I${common.pio_libdir}/WiFiClientSecure/src
    -Ilib/AzureIoTSocket_WiFi/src
    -Ilib/AzureIoTHub/src
    -Ilib/AzureIoTProtocol_HTTP/src
    -Ilib/AzureIoTProtocol_MQTT/src
    -Ilib/AzureIoTUtility/src
;lib_ldf_mode = off
lib_deps =
    lib/AzureIoTHub
    lib/AzureIoTSocket_WiFi
    lib/AzureIoTProtocol_MQTT
    lib/AzureIoTProtocol_HTTP
    lib/AzureIoTUtility
    ${common.pio_libdir}/WiFi/src
    ${common.pio_libdir}/WiFiClientSecure/src
    ArduinoJson
    CCS811
    ClosedCube HDC1080
```

Platform

Build config

Folders

Libraries by folder

Libraries by PIO

```
> Executing task: C:\Users\tkopa\.platformio\penv\Scripts\platformio.exe run <

Processing esp-wrover-kit (platform: espressif32; board: esp-wrover-kit; framew
------------------------------------------------------------------------------
Verbose mode can be enabled via `-v, --verbose` option
CONFIGURATION: https://docs.platformio.org/page/boards/espressif32/esp-wrover-k
PLATFORM: Espressif 32 (2.0.0) > Espressif ESP-WROVER-KIT
HARDWARE: ESP32 240MHz, 320KB RAM, 4MB Flash
DEBUG: Current (ftdi) On-board (ftdi) External (esp-prog, iot-bus-jtag, jlink,
PACKAGES:
 - framework-arduinoespressif32 3.10004.200129 (1.0.4)
 - tool-esptoolpy 1.20600.0 (2.6.0)
 - toolchain-xtensa32 2.50200.80 (5.2.0)
LDF: Library Dependency Finder -> http://bit.ly/configure-pio-ldf
LDF Modes: Finder ~ chain, Compatibility ~ soft
Found 41 compatible libraries
Scanning dependencies...
Dependency Graph
|-- <AzureIoTHub> 1.3.9
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src> 0.0.0+20201001201656
|   |   |-- <src> 0.0.0+20201001201657
|   |   |   |-- <src> 0.0.0+20201001201656
|-- <AzureIoTSocket_WiFi> 1.0.0
|   |-- <src> 0.0.0+20201001201656
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src> 0.0.0+20201001201656
|   |   |-- <src> 0.0.0+20201001201657
|   |   |   |-- <src> 0.0.0+20201001201656
|-- <AzureIoTProtocol_MQTT> 1.3.9
|   |-- <AzureIoTHub> 1.3.9
|   |   |-- <AzureIoTUtility> 1.3.9
|   |   |   |-- <src> 0.0.0+20201001201656
|   |   |   |-- <src> 0.0.0+20201001201657
|   |   |   |   |-- <src> 0.0.0+20201001201656
|   |-- <AzureIoTUtility> 1.3.9
|   |   |-- <src> 0.0.0+20201001201656
```

IoT Hub, initialization – code (simplified)

```
(void)IoTHub_Init();
IOTHUB_CLIENT_TRANSPORT_PROVIDER protocol = MQTT_Protocol;
OR : IOTHUB_CLIENT_TRANSPORT_PROVIDER protocol = HTTP_Protocol;
iotHubClientHandle = IoTHubClient_LL_CreateFromConnectionString(connectionString, protocol)

// turn off diagnostic sampling
int diag_off = 0;
IoTHubDeviceClient_LL_SetOption(iotHubClientHandle, OPTION_DIAGNOSTIC_SAMPLING_PERCENTAGE, &diag_off);

#ifndef USE_HTTP
    // Example sdk status tracing for troubleshooting
    bool traceOn = true;
    IoTHubDeviceClient_LL_SetOption(iotHubClientHandle, OPTION_LOG_TRACE, &traceOn);
#endif //

// Setting the Trusted Certificate.
IoTHubDeviceClient_LL_SetOption(iotHubClientHandle, OPTION_TRUSTED_CERT, certificates);

#if defined USE_MQTT
IoTHubClient_LL_SetMessageCallback(iotHubClientHandle, ReceiveMessageCallback, &receiveContext);
IoTHubClient_LL_SetDeviceMethodCallback(iotHubClientHandle, DeviceMethodCallback, &receiveContext)
// Setting connection status callback to get indication of connection to iothub
(void)IoTHubDeviceClient_LL_SetConnectionStatusCallback(iotHubClientHandle, connection_status_callback, NULL);
```

IoT Hub – ReceiveMessageCallback (simplified!)

```
static IOTHUBMESSAGE_DISPOSITION_RESULT ReceiveMessageCallback(IOTHUB_MESSAGE_HANDLE message, void *userContextCallback)
{
  int *counter = (int *)userContextCallback;
  MAP_HANDLE mapProperties;
  // Message properties
  if ((messageId = IoTHubMessage_GetMessageId(message)) == NULL)
  if ((correlationId = IoTHubMessage_GetCorrelationId(message)) == NULL)
  if ((userDefinedContentType = IoTHubMessage_GetContentTypeSystemProperty(message)) == NULL)
  if ((userDefinedContentEncoding = IoTHubMessage_GetContentEncodingSystemProperty(message)) == NULL)
   // Message content
  if (IoTHubMessage_GetByteArray(message, (const unsigned char **)&buffer, &size) != IOTHUB_MESSAGE_OK)
   LogInfo("Received Message [%d]\r\n Message ID: %s\r\n Correlation ID: %s\r\n Content-Type: %s\r\n Content-
Encoding: %s\r\n Data: <<<%.*s>>> & Size=%d\r\n",
   (*counter), messageId, correlationId, userDefinedContentType, userDefinedContentEncoding, (int)size, buffer, (int)size);

  // Retrieve properties from the message
  mapProperties = IoTHubMessage_Properties(message);
  Map_GetInternals(mapProperties, &keys, &values, &propertyCount)
  /* Some device specific action code goes here... */
  (*counter)++;
  return IOTHUBMESSAGE_ACCEPTED;
}
```

IoT Hub – DeviceMethodCallback (simplified!)

```
static int DeviceMethodCallback(const char *method_name, const unsigned char *payload, size_t size, unsigned char **response, size_t *resp_size, void *userContextC
allback)
{
  (void)userContextCallback;
  LogInfo("\r\nDevice Method called\r\n");
  LogInfo("Device Method name:    %s\r\n", method_name);
  LogInfo("Device Method payload: %.*s\r\n", (int)size, (const char *)payload);
  if (strcmp(method_name, "start") == 0) messageSending = true;
  else if (strcmp(method_name, "delay") == 0) {
    StaticJsonDocument<200> doc;
    DeserializationError error = deserializeJson(doc, payload);
    if (error) { LogError("deserializeJson() failed: %s", error.c_str());  }
    else {
      IntervalMs = doc["ms"];
      LogInfo("IntervalMs:%d", IntervalMs);
    }
  }
  *resp_size = strlen(RESPONSE_STRING);
  if ((*response = (unsigned char *)malloc(*resp_size)) == NULL)
  {
    status = -1;
  }
  else
  {
    (void)memcpy(*response, RESPONSE_STRING, *resp_size);
  }
  return status;
}
```

IoT Hub – connections handling (simplified!)

```
static void SendConfirmationCallback(IOTHUB_CLIENT_CONFIRMATION_RESULT result, void *userContextCallback)
{
  //Setup watchdog
  //esp_task_wdt_reset();
  IOTHUB_MESSAGE_HANDLE *msg = (IOTHUB_MESSAGE_HANDLE *)userContextCallback;
  LogInfo("Confirmation %d result = %s\r\n", callbackCounter, MU_ENUM_TO_STRING(IOTHUB_CLIENT_CONFIRMATION_RESULT, result));
  /* Some device specific action code goes here... */
  callbackCounter++;
  if (result != IOTHUB_CLIENT_CONFIRMATION_OK)
    //esp_restart();
  //TK:Or caller
  IoTHubMessage_Destroy(*msg);
}

static void connection_status_callback(IOTHUB_CLIENT_CONNECTION_STATUS result, IOTHUB_CLIENT_CONNECTION_STATUS_REASON reason, void* user_
context)
{
    (void)reason;
    (void)user_context;
    if (result == IOTHUB_CLIENT_CONNECTION_AUTHENTICATED)
LogInfo("The device client is connected to iothub\r\n");
    else
LogInfo("The device client has been disconnected\r\n");
}
```

IoT Hub – message sending

```
ccs811.read(&eco2,&etvoc,&errstat,&raw);
if( errstat==CCS811_ERRSTAT_OK ) { t.SetValue("eco2",eco2); t.SetValue("etvoc",etvoc); }
t.SetValue("tempearture",hdc1080.readTemperature());

messagePayload = t.GetJson();
msg = IoTHubMessage_CreateFromByteArray((const unsigned char *)messagePayload, strlen(messagePayload)));
snprintf(buf,sizeof(buf),"%d",messageCount);
(void)IoTHubMessage_SetMessageId(msg, buf);
//(void)IoTHubMessage_SetCorrelationId(msg, "CORE_ID");
(void)IoTHubMessage_SetContentTypeSystemProperty(msg, "application%2Fjson");
(void)IoTHubMessage_SetContentEncodingSystemProperty(msg, "utf-8");

MAP_HANDLE propMap = IoTHubMessage_Properties(msg);
(void)sprintf_s(propText, sizeof(propText), (random(100)>50) ? "true" : "false");
if (Map_AddOrUpdate(propMap, "valAlert", propText) != MAP_OK)
(IoTHubClient_LL_SendEventAsync(iotHubClientHandle, msg, SendConfirmationCallback, &msg) != IOTHUB_CLIENT_OK)
{
  LogError("ERROR: IoTHubClient_SendEventAsync.....FAILED!\r\n");
}
messageCount++;
      }
```

IoT Hub – message – REMEMBER SINGLE THREAD

```
while ((IoTHubClient_LL_GetSendStatus(iotHubClientHandle, &status) == IOTHUB_CLIENT_OK) &&
(status == IOTHUB_CLIENT_SEND_STATUS_BUSY))
{
IoTHubClient_LL_DoWork(iotHubClientHandle);
ThreadAPI_Sleep(100);
}
```

Technically, IoTHubClient_LL_SendEventAsync add a message to the internal queue.

```
else
{
    /*Codes_SRS_IOTHUBCLIENT_LL_02_013: [IoTHubClientCore_LL_SendEventAsync shall a
    newEntry->callback = eventConfirmationCallback;
    newEntry->context = userContextCallback;
    DList_InsertTailList(&(iotHubClientHandle->waitingToSend), &(newEntry->entry));
    /*Codes_SRS_IOTHUBCLIENT_LL_02_015: [Otherwise IoTHubClientCore_LL_SendEventAsy
    result = IOTHUB_CLIENT_OK;
```

IoTHubClient_LL_DoWork "do work"

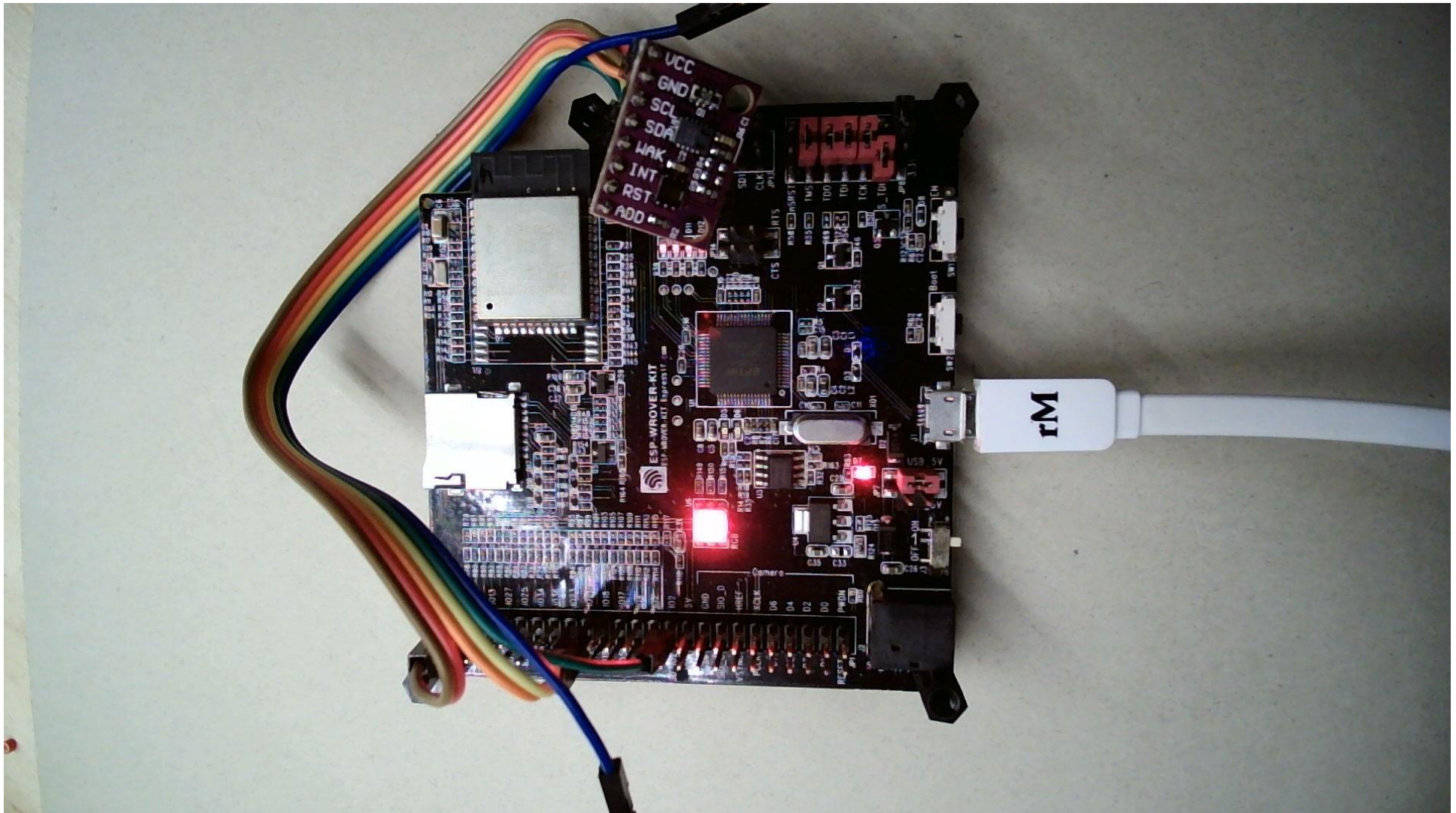
```
/*Codes_SRS_IOTHUBCLIENT_LL_07_008: [ IoTHubClientCore_LL_DoWork shall iterate the me
DLIST_ENTRY* client_item = handleData->iot_msg_queue.Flink;
while (client_item != &(handleData->iot_msg_queue)) /*while we are not at the end of
{
    PDLIST_ENTRY next_item = client_item->Flink;

    IOTHUB_DEVICE_TWIN* queue_data = containingRecord(client_item, IOTHUB_DEVICE_TWIN
    IOTHUB_IDENTITY_INFO identity_info;
    identity_info.device_twin = queue_data;
    IOTHUB_PROCESS_ITEM_RESULT process_results =  handleData->IoTHubTransport_Process
    if (process_results == IOTHUB_PROCESS_CONTINUE || process_results == IOTHUB_PROCE
```

There is also API without "LL" (LowLevel) where IoT Hub use threads. But – this requires device with normal OS (ARM, x86 etc)

Maybe interrupt?

```
void IRAM_ATTR onTimer() {
  portENTER_CRITICAL_ISR(&timerMux);
  ccs811.read(&eco2,&etvoc,&errstat,&raw);
  if( errstat==CCS811_ERRSTAT_OK ) {
    t.SetValue("eco2",eco2);
    t.SetValue("etvoc",etvoc);
  }
  portEXIT_CRITICAL_ISR(&timerMux);
}


// //Set Interrupt
timer = timerBegin(0, 80, true);
timerAttachInterrupt(timer, &onTimer, true);
///////1000000 = 1s
timerAlarmWrite(timer, 10 * 1000000, true);
timerAlarmEnable(timer);
```

ESP-WROVER-KIT + (I2C) + [CCS811 (gas sensor) + HDC1080 (humidity + temperature sensor)]



Demo

ESP32 and IoTHub

Z:\TSGITHUB\2020IotHardware\ESP32\Esp32IoTHub

EXPLORER

OPEN EDITORS

ESP32IOTHUB

- .pio
  - build
  - libdeps \ esp-wrover-kit
    - ArduinoJson
      - examples
      - src
      - .clang-format
      - .gitattributes
      - .gitignore
      - .mbedignore
      - .travis.yml
      - appveyor.yml
      - ArduinoJson.h
      - banner.svg
      - CHANGELOG.md
      - CMakeLists.txt
      - component.mk
      - CONTRIBUTING.md
      - keywords.txt
      - library.json
      - library.properties
      - LICENSE.md
      - README.md
      - SUPPORT.md
    - ClosedCube HDC1080
      - src
      - src@src-22da4c01638ad979a3ed...
- .vscode
- include
- lib
  - AzureIoTHub
  - AzureIoTProtocol_HTTP
  - AzureIoTProtocol_MQTT
  - AzureIoTSocket_WiFi
  - AzureIoTUtility
  - README
- src
- test
- .gitignore
- .travis.yml

OUTLINE

TIMELINE

AZURE IOT HUB

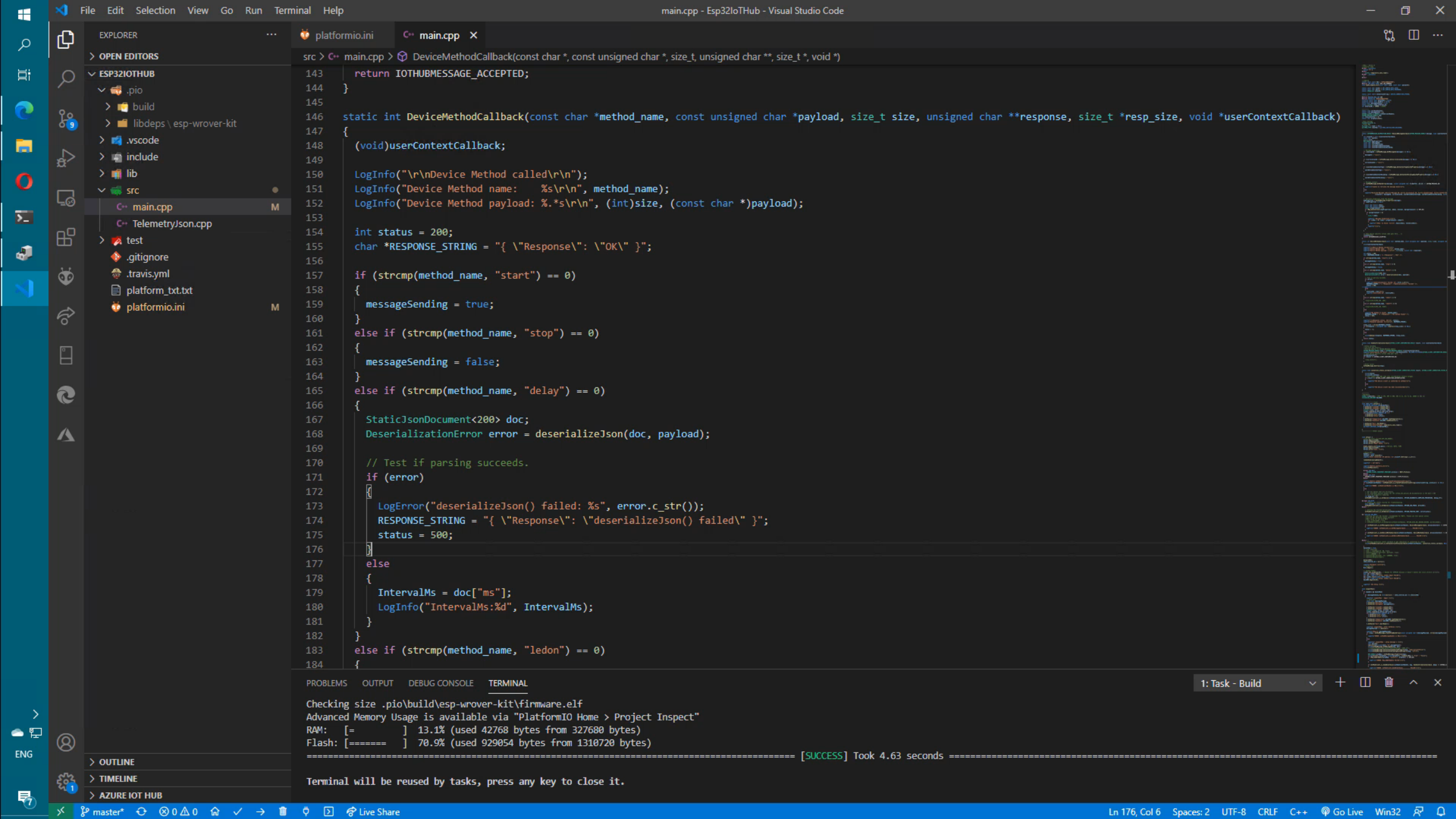
platformio.ini

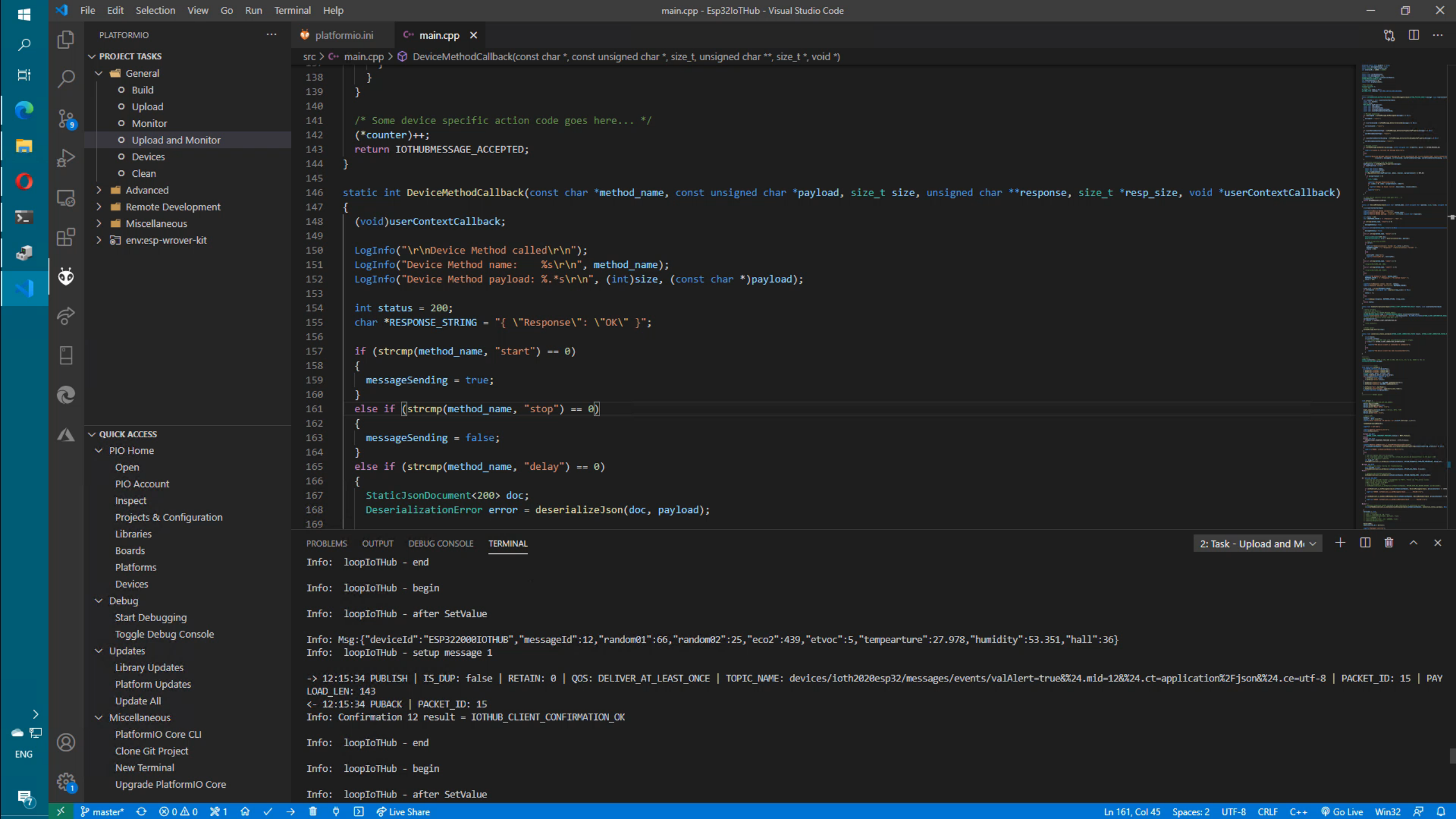
```
; https://docs.platformio.org/page/projectconf.html
[common]
pio_libdir = C:/Users/tkopa/.platformio/packages/framework-arduinoespressif32/libraries

[env:esp-wrover-kit]
platform = espressif32
board = esp-wrover-kit
framework = arduino
monitor_speed = 115200
upload_speed = 2000000
monitor_port = COM[5]
upload_port= COM[5]

build_flags =
    -DCORE_DEBUG_LEVEL=ARDUHAL_LOG_LEVEL_VERBOSE
    -DDONT_USE_UPLOADTOBLOB
    -DESP32
    -DARDUINO_ARCH_ESP32
    -DUSE_BALTIMORE_CERT
    -w
    -IInclude
    -I${common.pio_libdir}/WiFi/src
    -I${common.pio_libdir}/WiFiClientSecure/src
    -Ilib/AzureIoTSocket_WiFi/src
    -Ilib/AzureIoTHub/src
    -Ilib/AzureIoTProtocol_HTTP/src
    -Ilib/AzureIoTProtocol_MQTT/src
    -Ilib/AzureIoTUtility/src
lib_deps =
    lib/AzureIoTHub
    lib/AzureIoTSocket_WiFi
    lib/AzureIoTProtocol_MQTT
    lib/AzureIoTProtocol_HTTP
    lib/AzureIoTUtility
    ${common.pio_libdir}/WiFi/src
    ${common.pio_libdir}/WiFiClientSecure/src
    ArduinoJson
    CCS811
    ClosedCube HDC1080

```

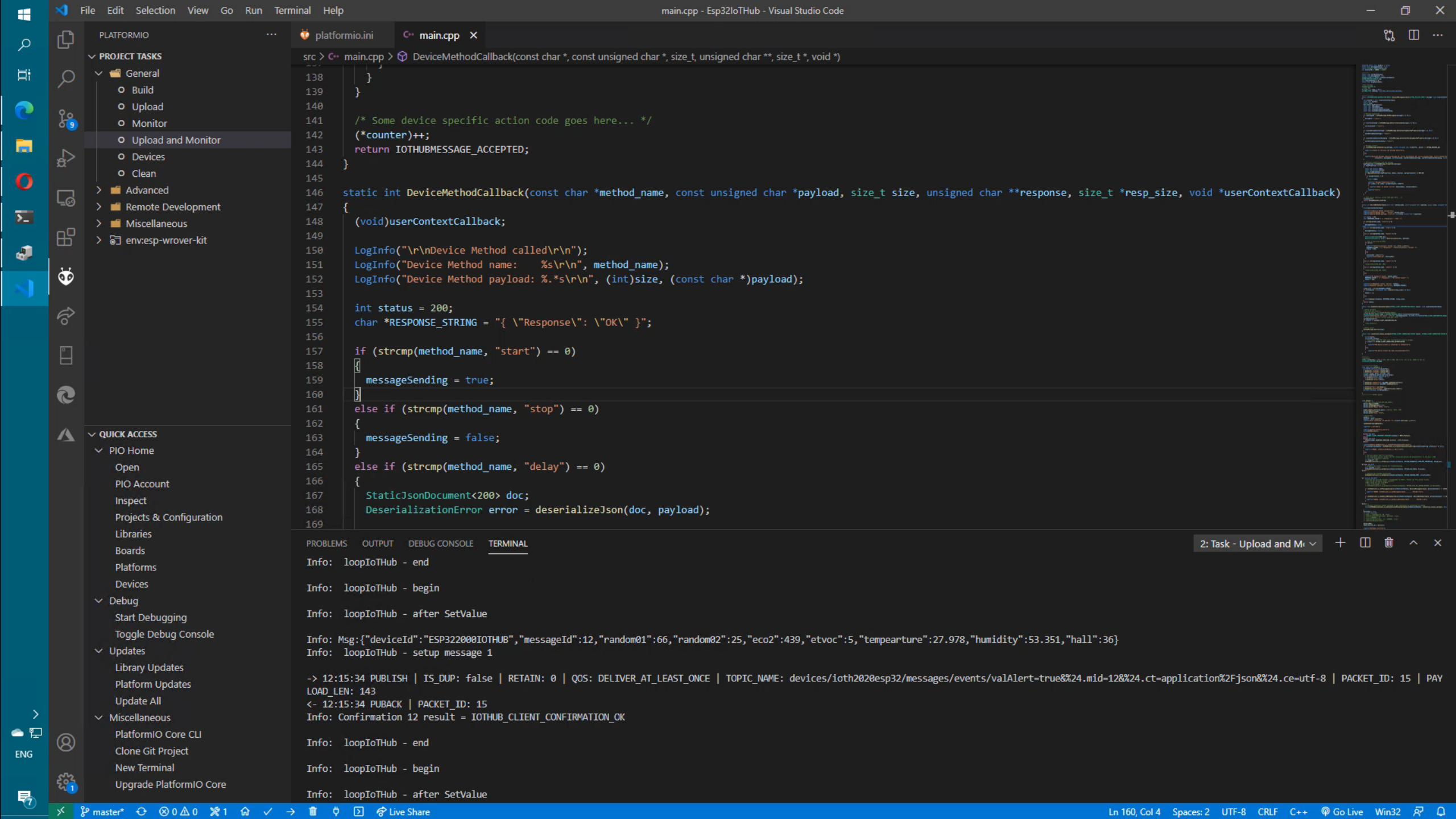
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: Task - Build

```
Checking size .pio\build\esp-wrover-kit\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM:   [=         ]  13.1% (used 42768 bytes from 327680 bytes)
Flash: [=======   ]  70.9% (used 929054 bytes from 1310720 bytes)
========================================================================================== [SUCCESS] Took 4.63 seconds ==========================================================================================

Terminal will be reused by tasks, press any key to close it.
```


File Edit Selection View Go Run Terminal Help
main.cpp - Esp32IoTHub - Visual Studio Code
EXPLORER
OPEN EDITORS
ESP32IOTHUB
.pio
build
libdeps \ esp-wrover-kit
.vscode
include
lib
src
main.cpp M
TelemetryJson.cpp
test
.gitignore
.travis.yml
platform_txt.txt
platformio.ini M
platformio.ini
main.cpp
src > main.cpp > DeviceMethodCallback(const char *, const unsigned char *, size_t, unsigned char **, size_t *, void *)
143 return IOTHUBMESSAGE_ACCEPTED;
144 }
145
146 static int DeviceMethodCallback(const char *method_name, const unsigned char *payload, size_t size, unsigned char **response, size_t *resp_size, void *userContextCallback)
147 {
148 (void)userContextCallback;
149
150 LogInfo("\r\nDevice Method called\r\n");
151 LogInfo("Device Method name: %s\r\n", method_name);
152 LogInfo("Device Method payload: %.*s\r\n", (int)size, (const char *)payload);
153
154 int status = 200;
155 char *RESPONSE_STRING = "{ \"Response\": \"OK\" }";
156
157 if (strcmp(method_name, "start") == 0)
158 {
159 messageSending = true;
160 }
161 else if (strcmp(method_name, "stop") == 0)
162 {
163 messageSending = false;
164 }
165 else if (strcmp(method_name, "delay") == 0)
166 {
167 StaticJsonDocument<200> doc;
168 DeserializationError error = deserializeJson(doc, payload);
169
170 // Test if parsing succeeds.
171 if (error)
172 {
173 LogError("deserializeJson() failed: %s", error.c_str());
174 RESPONSE_STRING = "{ \"Response\": \"deserializeJson() failed\" }";
175 status = 500;
176 }
177 else
178 {
179 IntervalMs = doc["ms"];
180 LogInfo("IntervalMs:%d", IntervalMs);
181 }
182 }
183 else if (strcmp(method_name, "ledon") == 0)
184 {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Task - Build
Checking size .pio\build\esp-wrover-kit\firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [=] 13.1% (used 42768 bytes from 327680 bytes)
Flash: [=======] 70.9% (used 929054 bytes from 1310720 bytes)
========== [SUCCESS] Took 4.63 seconds ==========
Terminal will be reused by tasks, press any key to close it.
OUTLINE
TIMELINE
AZURE IOT HUB
master*
Live Share
Ln 176, Col 6 Spaces: 2 UTF-8 CRLF C++ Go Live Win32


File Edit Selection View Go Run Terminal Help
main.cpp - Esp32IoTHub - Visual Studio Code
PLATFORMIO
PROJECT TASKS
General
Build
Upload
Monitor
Upload and Monitor
Devices
Clean
Advanced
Remote Development
Miscellaneous
env:esp-wrover-kit
QUICK ACCESS
PIO Home
Open
PIO Account
Inspect
Projects & Configuration
Libraries
Boards
Platforms
Devices
Debug
Start Debugging
Toggle Debug Console
Updates
Library Updates
Platform Updates
Update All
Miscellaneous
PlatformIO Core CLI
Clone Git Project
New Terminal
Upgrade PlatformIO Core
platformio.ini
main.cpp
src > main.cpp > DeviceMethodCallback(const char *, const unsigned char *, size_t, unsigned char **, size_t *, void *)
138 }
139 }
140
141 /* Some device specific action code goes here... */
142 (*counter)++;
143 return IOTHUBMESSAGE_ACCEPTED;
144 }
145
146 static int DeviceMethodCallback(const char *method_name, const unsigned char *payload, size_t size, unsigned char **response, size_t *resp_size, void *userContextCallback)
147 {
148 (void)userContextCallback;
149
150 LogInfo("\r\nDevice Method called\r\n");
151 LogInfo("Device Method name: %s\r\n", method_name);
152 LogInfo("Device Method payload: %.*s\r\n", (int)size, (const char *)payload);
153
154 int status = 200;
155 char *RESPONSE_STRING = "{ \"Response\": \"OK\" }";
156
157 if (strcmp(method_name, "start") == 0)
158 {
159 messageSending = true;
160 }
161 else if (strcmp(method_name, "stop") == 0)
162 {
163 messageSending = false;
164 }
165 else if (strcmp(method_name, "delay") == 0)
166 {
167 StaticJsonDocument<200> doc;
168 DeserializationError error = deserializeJson(doc, payload);
169
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Task - Upload and M
Info: loopIoTHub - end
Info: loopIoTHub - begin
Info: loopIoTHub - after SetValue
Info: Msg:{"deviceId":"ESP322000IOTHUB","messageId":12,"random01":66,"random02":25,"eco2":439,"etvoc":5,"tempearture":27.978,"humidity":53.351,"hall":36}
Info: loopIoTHub - setup message 1
-> 12:15:34 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONCE | TOPIC_NAME: devices/ioth2020esp32/messages/events/valAlert=true&%24.mid=12&%24.ct=application%2Fjson&%24.ce=utf-8 | PACKET_ID: 15 | PAYLOAD_LEN: 143
<- 12:15:34 PUBACK | PACKET_ID: 15
Info: Confirmation 12 result = IOTHUB_CLIENT_CONFIRMATION_OK
Info: loopIoTHub - end
Info: loopIoTHub - begin
Info: loopIoTHub - after SetValue
ENG
master*
Live Share
Ln 161, Col 45
Spaces: 2
UTF-8
CRLF
C++
Go Live
Win32


File Edit Selection View Go Run Terminal Help
main.cpp - Esp32IoTHub - Visual Studio Code
PLATFORMIO
PROJECT TASKS
General
Build
Upload
Monitor
Upload and Monitor
Devices
Clean
Advanced
Remote Development
Miscellaneous
env:esp-wrover-kit
QUICK ACCESS
PIO Home
Open
PIO Account
Inspect
Projects & Configuration
Libraries
Boards
Platforms
Devices
Debug
Start Debugging
Toggle Debug Console
Updates
Library Updates
Platform Updates
Update All
Miscellaneous
PlatformIO Core CLI
Clone Git Project
New Terminal
Upgrade PlatformIO Core
platformio.ini
main.cpp
src > main.cpp > DeviceMethodCallback(const char *, const unsigned char *, size_t, unsigned char **, size_t *, void *)
138 }
139 }
140
141 /* Some device specific action code goes here... */
142 (*counter)++;
143 return IOTHUBMESSAGE_ACCEPTED;
144 }
145
146 static int DeviceMethodCallback(const char *method_name, const unsigned char *payload, size_t size, unsigned char **response, size_t *resp_size, void *userContextCallback)
147 {
148 (void)userContextCallback;
149
150 LogInfo("\r\nDevice Method called\r\n");
151 LogInfo("Device Method name: %s\r\n", method_name);
152 LogInfo("Device Method payload: %.*s\r\n", (int)size, (const char *)payload);
153
154 int status = 200;
155 char *RESPONSE_STRING = "{ \"Response\": \"OK\" }";
156
157 if (strcmp(method_name, "start") == 0)
158 {
159 messageSending = true;
160 }
161 else if (strcmp(method_name, "stop") == 0)
162 {
163 messageSending = false;
164 }
165 else if (strcmp(method_name, "delay") == 0)
166 {
167 StaticJsonDocument<200> doc;
168 DeserializationError error = deserializeJson(doc, payload);
169
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Task - Upload and M
Info: loopIoTHub - end
Info: loopIoTHub - begin
Info: loopIoTHub - after SetValue
Info: Msg:{"deviceId":"ESP322000IOTHUB","messageId":12,"random01":66,"random02":25,"eco2":439,"etvoc":5,"tempearture":27.978,"humidity":53.351,"hall":36}
Info: loopIoTHub - setup message 1
-> 12:15:34 PUBLISH | IS_DUP: false | RETAIN: 0 | QOS: DELIVER_AT_LEAST_ONCE | TOPIC_NAME: devices/ioth2020esp32/messages/events/valAlert=true&%24.mid=12&%24.ct=application%2Fjson&%24.ce=utf-8 | PACKET_ID: 15 | PAYLOAD_LEN: 143
<- 12:15:34 PUBACK | PACKET_ID: 15
Info: Confirmation 12 result = IOTHUB_CLIENT_CONFIRMATION_OK
Info: loopIoTHub - end
Info: loopIoTHub - begin
Info: loopIoTHub - after SetValue
master*
Live Share
Ln 160, Col 4
Spaces: 2
UTF-8
CRLF
C++
Go Live
Win32
ENG

Menu | Direct method - Microsoft | TSI Explorer

insights.timeseries.azure.com

C | AZ | Azure Cloud Shell | TSI Explorer | VSO | DR | tomaszkopacz.spac... | Notebooks | SF | M | D | Office 365 | OLD | V | AAD | ADX (demo12.west... | Databricks | Azure IoT Demo M... | Provider: Azure - Te... | JavaScript Window...

Time Series Insights

New explorer On

Tomasz Kopacz
TKOPACZ@MICROSOFT.COM

Open | Save | Share | Refresh | Auto Refresh Off

10/02/2020 18:09 - 10/03/2020 18:09 (CEST)

pltkw3tsi

Warm store queries On

To organize time series instances into a hierarchy, create a Model

Unassigned Time Series Instances

Search Time Series Instances...

- null
- EE
- ESP322000IOTHUB
- esp32fy18olimexespevb
- esp32wemoslcd
- ioth2020ESP8266
- iothe32soil01
- m5stickc01
- netdevice
- sphere01

Select time series instances from the hierarchy
or restore previous session

ENG

IoT Hub + large object (large messages) + (MQTT)

Flow:

Device -> IoT Hub -> Azure Storage

Authorization: SAS

(ok – “large” IoT Hub Client Library - `clt.UploadToBlobAsync`)

Security IoT

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-security

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-file-upload

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-mqtt-support#using-the-mqtt-protocol-directly-as-a-device

Custom library, SaaS

(and) – intro to MQTT pure communication

URLs

```
Config::DEVICE_KEY = pref.GetValue("DEVICE_KEY"); // key like emABCK9SiWnIGHYIDBHVrpELlTw3W4F7PsySXeuFqyg=
Config::DEVICE_NAME = pref.GetValue("DEVICE_NAME"); // device name
Config::IOT_HUB_NAME = pref.GetValue("IOT_HUB_NAME"); // iot hub (name from Azure) device name

// DNS Name of MQTT Server = IoT Hub Name
MQTT_SERVER      // Config::IOT_HUB_NAME + ".azure-devices.net";
// User is combination of DNS Name and device name (password - SAS - see next slide)
USER_NAME// Config::IOT_HUB_NAME + ".azure-devices.net/" + Config::DEVICE_NAME;
// Topic name for receiving cloud to device messages (and - then methods etc)
MQTTReceiveTopic // "devices/" + Config::DEVICE_NAME + "/messages/#";
// Topic name for sending telemetry
MQTTSendTopic    // "devices/"+ Config::DEVICE_NAME + "/messages/events/";
FileUploadURI    // "https://" + Config::MQTT_SERVER + "/devices/" + Config::DEVICE_NAME + "/files?api-version=2018-06-30";
NotificationURI  // "https://" + Config::MQTT_SERVER + "/devices/" + Config::DEVICE_NAME + "/files/notifications?api-version=2018-06-30";
```

Steps

Generate SAS based on time, key, device name and IoT Hub DNS

Generate container, name etc (call **FileUploadURI** REST) :

- hostname, containerName, blobName ,sasToken, and cid (id of upload, **correlation id**)

Upload to BLOB (separate HttpClient and WiFiClientSecure)

Confirm upload (using notification API and **correlation id**)

Code – Generate SAS (simplified)

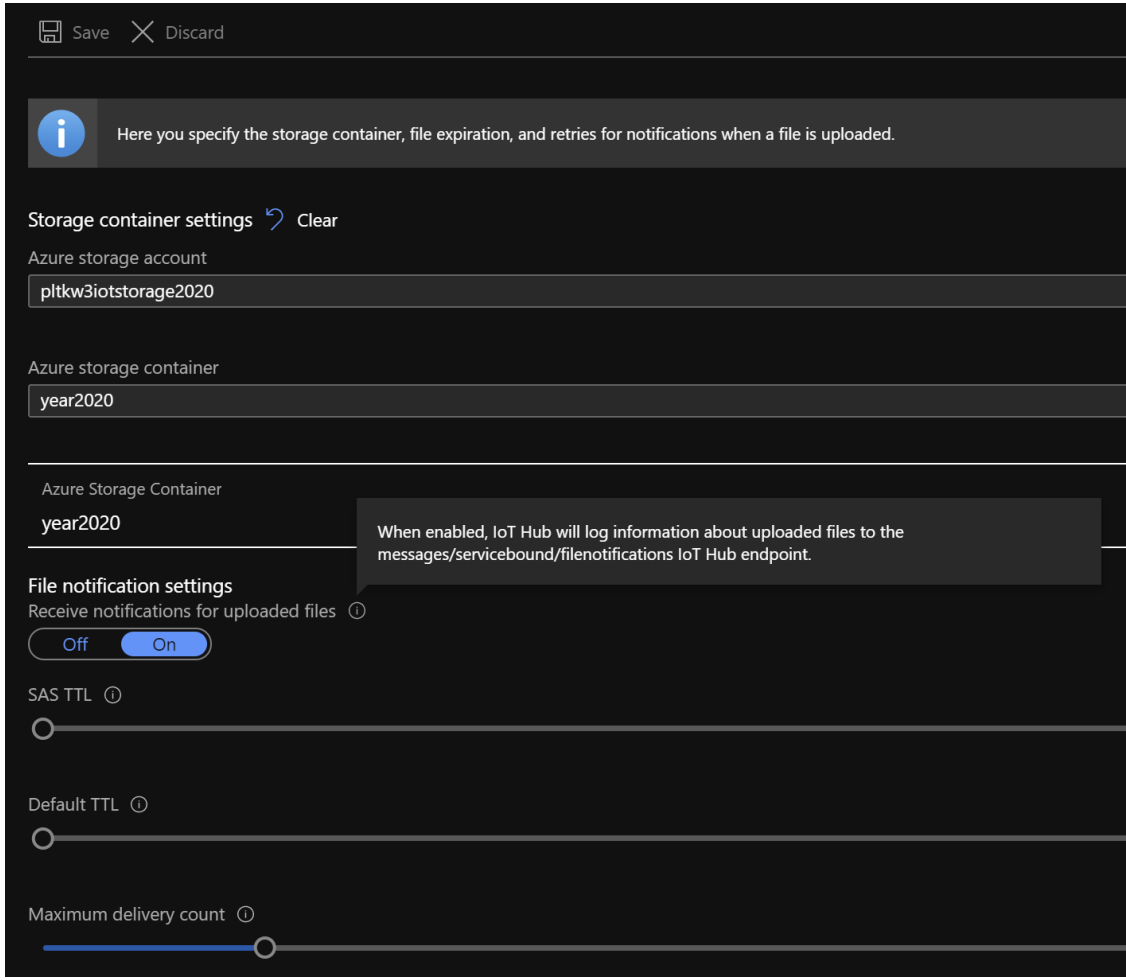
```
int32_t epoch = (int32_t)time(0);
int32_t tokenExpiry = epoch + IoTHub_tokenExpirationPeriod; //360
char buf[30];
itoa(tokenExpiry,buf,10);
String tokenExpiryString = buf;
String sign_key = urlEncode(String(Config::MQTT_SERVER)) + "\n" + tokenExpiryString;
int keyLen = decodeBase64(Config::DEVICE_KEY, NULL, 0);
uint8_t* keyBase64 = new uint8_t[keyLen];
keyLen = decodeBase64(Config::DEVICE_KEY, keyBase64, keyLen);
String signature = hashIt(sign_key, keyBase64, keyLen);
String result = "SharedAccessSignature sr=" + String(Config::MQTT_SERVER) + "&sig=" + signature + "&se=" + tokenExpiryString;
return result;

hashIt
    generateHash(signedOut, (uint8_t*)data.c_str(), (size_t)data.length(), key, keyLength); // SHA256 (look for sample implementation)
    work = encodeBase64(signedOut, sizeof(signedOut)); //required - for passing in headers
    return urlEncode(work);
```

Code – Prepare for upload

```
strSas = m_sas.get_iot_hub_sas_token();
log_v("SAS:\r\n%s\r\n",strSas.c_str());
HTTPClient https;
https.begin(m_clientIoT, Config::FileUploadURI);
https.addHeader("Authorization", strSas.c_str()); https.addHeader("Content-Type", "application/json");
String postBody = "{ \"blobName\": \"" + blobName + "\"}";
int httpCode = https.POST(postBody);
log_v("Body:\r\n%s\r\nRES:\r\n%d\r\n",postBody.c_str(),httpCode);
String resp = https.getString();
log_v("Resp:\r\n%s\r\n",resp.c_str());
https.end();
StaticJsonDocument<500> doc;
deserializeJson(doc, resp);
 String hn = doc["hostName"];
 String cn = doc["containerName"];
 String bn = doc["blobName"];
 String sas = doc["sasToken"];
 cid = doc["correlationId"].as<String>();
uploadURL = "https://" + hn + "/" + cn + "/" + bn + sas;
log_v("\r\n---------------\r\n%s\r\n-----------------\r\n",uploadURL.c_str());
```

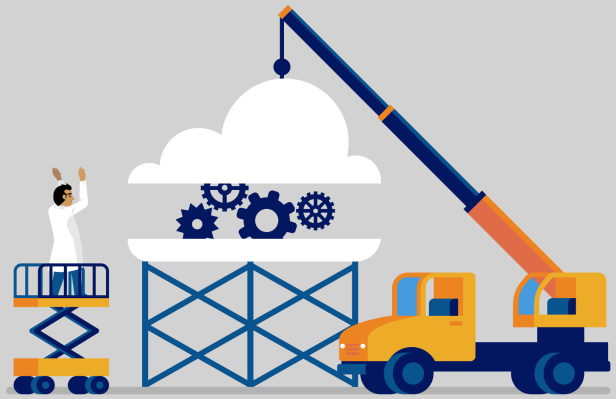
Upload (BLOB REST) & notification (IoT Hub REST)

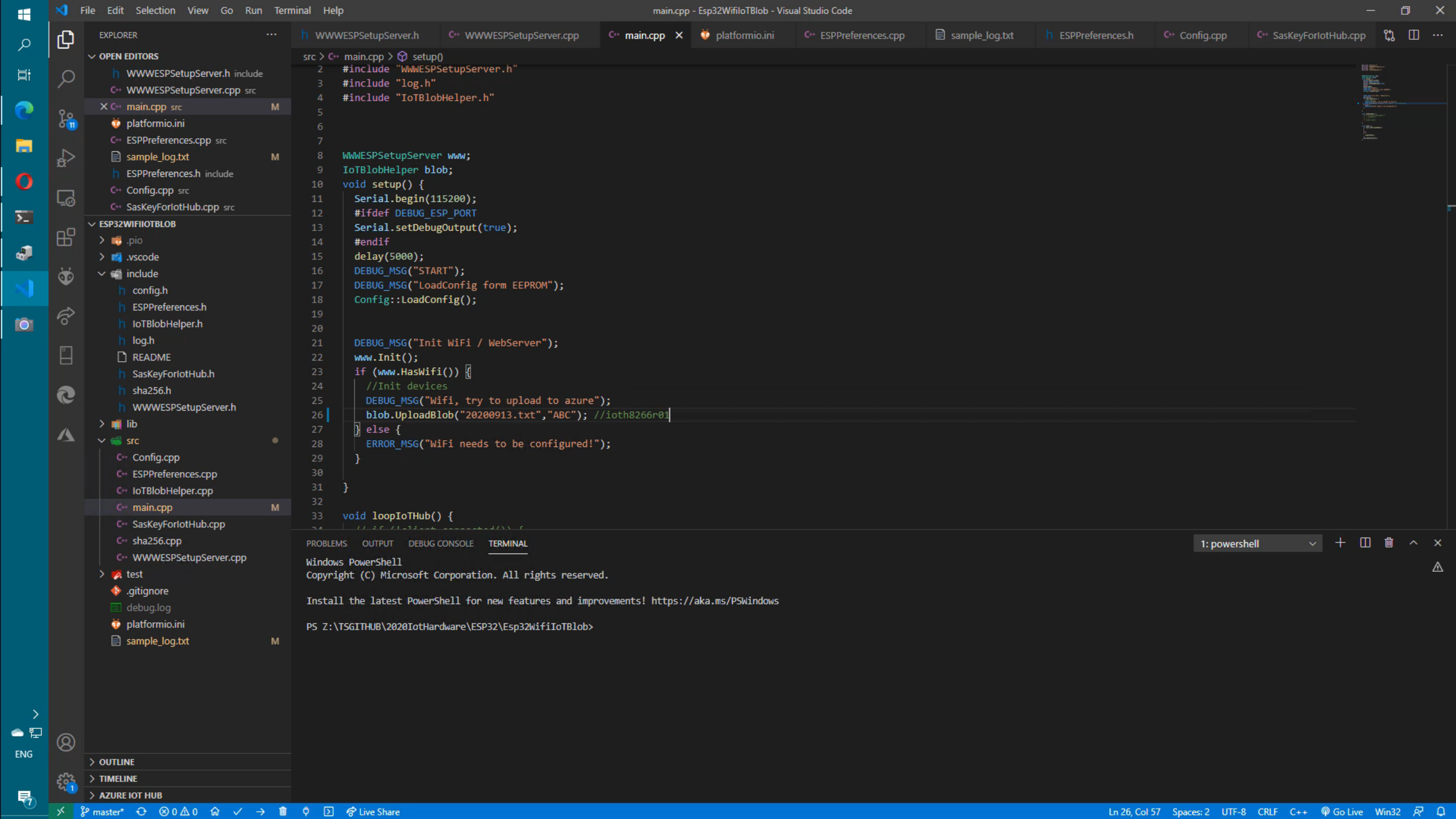
```
// Upload
HTTPClient https;
https.begin(m_clientBlob, uploadURL.c_str());
https.addHeader("x-ms-blob-type", "BlockBlob");
https.addHeader("Content-Type",
  contentType.c_str()); //"application/json"
int httpCode = https.PUT(data, size);
https.end();

// Notification
HTTPClient https1;
https1.begin(m_clientIoT, Config::NotificationURI);
https1.addHeader("Authorization", strSas.c_str());
https1.addHeader("Content-Type", "application/json");
https1.POST("{ \"correlationId\": \"" + cid + "\"}");
https1.end();
```

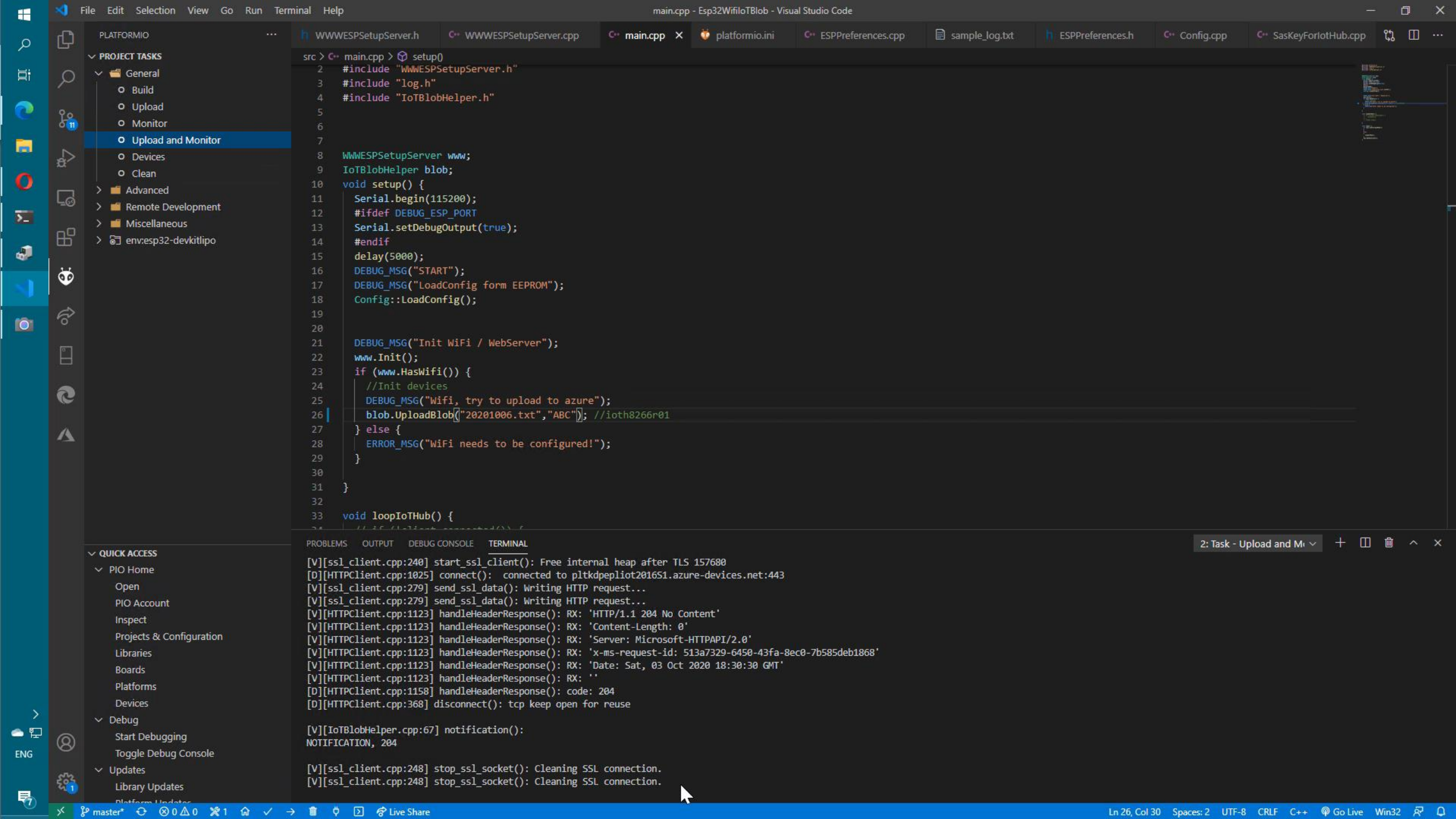


Demo

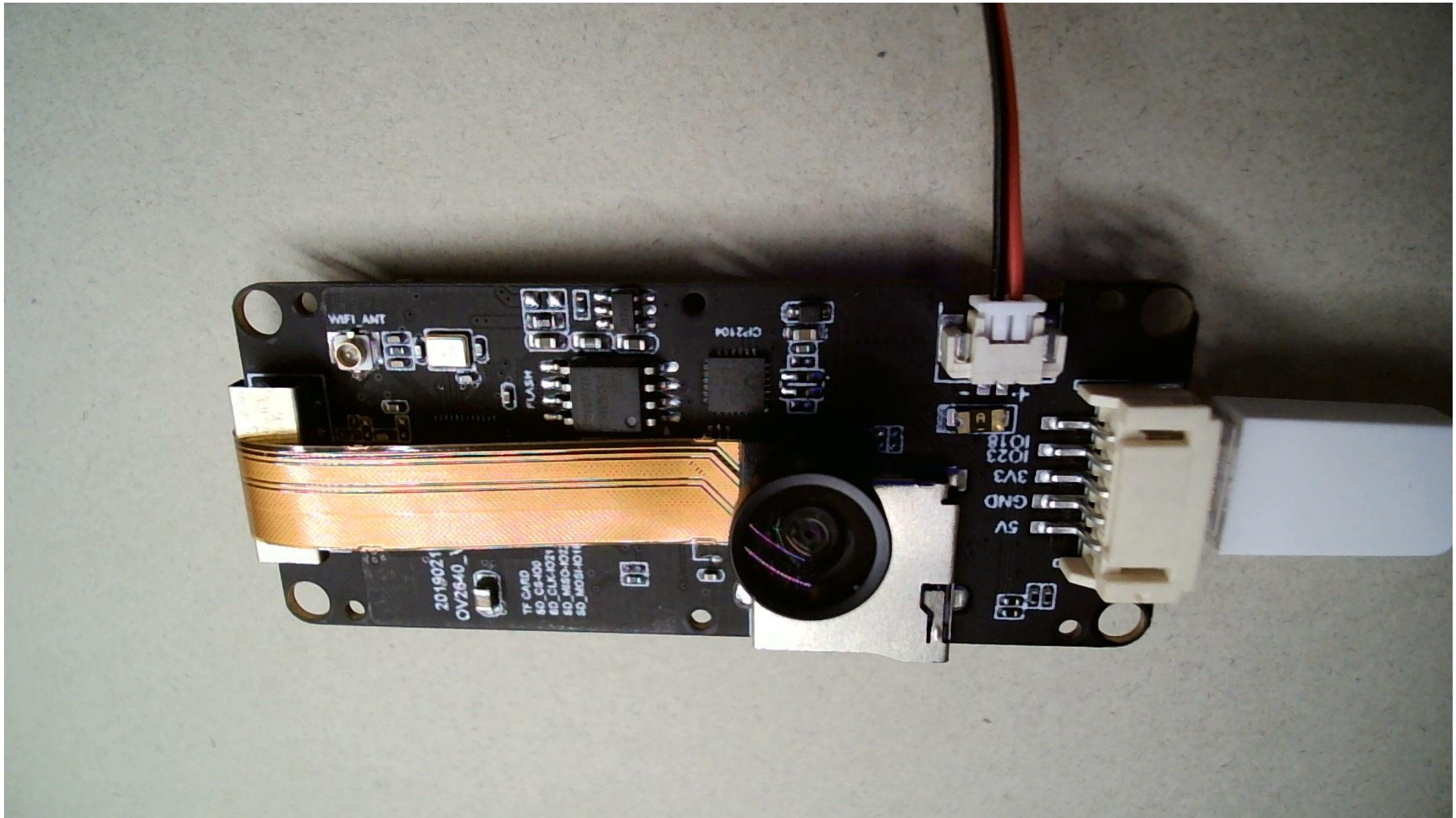
ESP32 and blob (large messages)
Also – High frequency / batch / package

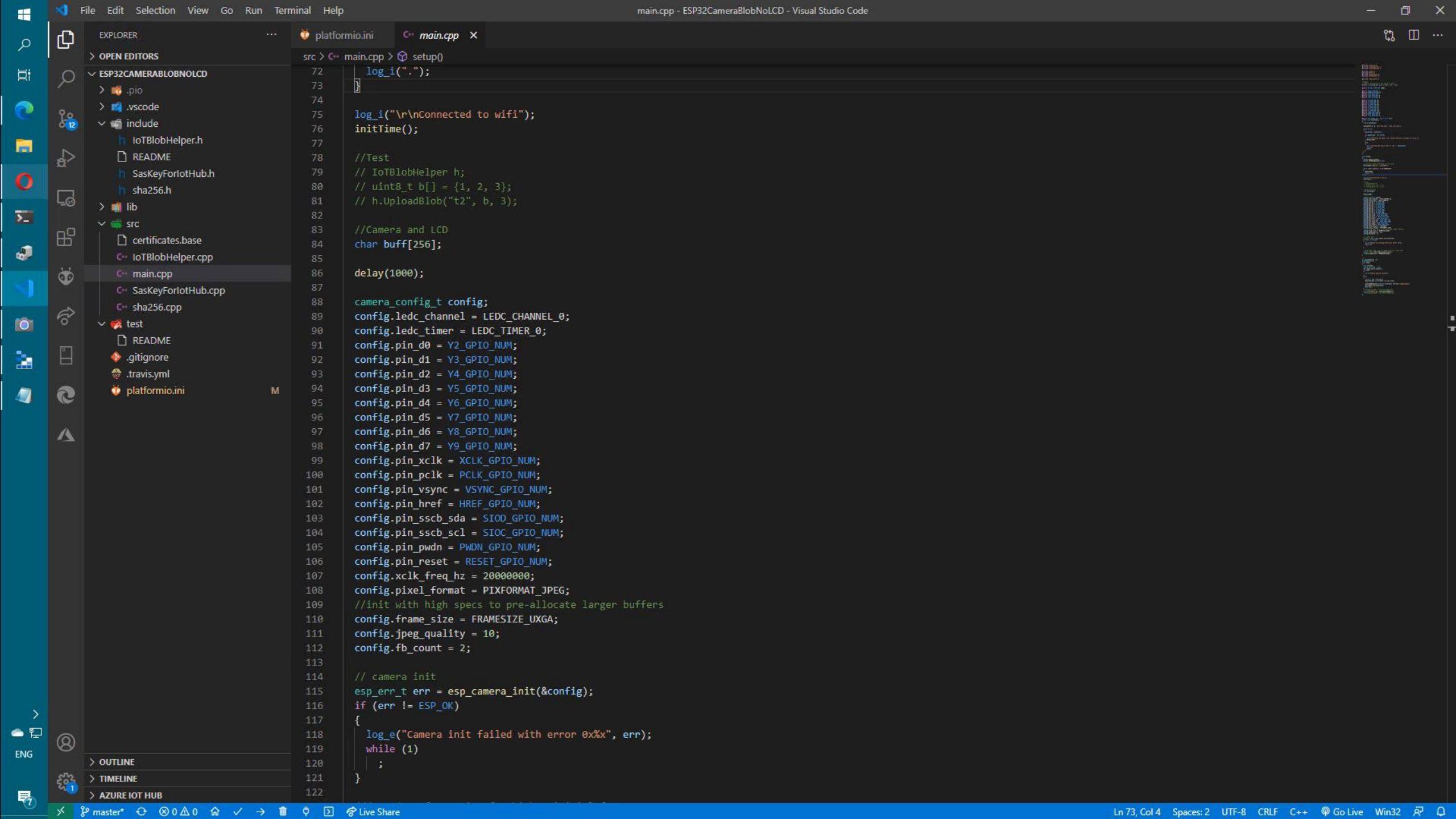



File Edit Selection View Go Run Terminal Help
main.cpp - Esp32WifiIoTBlob - Visual Studio Code
EXPLORER
OPEN EDITORS
WWWESPSetupServer.h include
WWWESPSetupServer.cpp src
main.cpp src M
platformio.ini
ESPPreferences.cpp src
sample_log.txt M
ESPPreferences.h include
Config.cpp src
SasKeyForIotHub.cpp src
ESP32WIFIIOTBLOB
.pio
.vscode
include
config.h
ESPPreferences.h
IoTBlobHelper.h
log.h
README
SasKeyForIotHub.h
sha256.h
WWWESPSetupServer.h
lib
src
Config.cpp
ESPPreferences.cpp
IoTBlobHelper.cpp
main.cpp M
SasKeyForIotHub.cpp
sha256.cpp
WWWESPSetupServer.cpp
test
.gitignore
debug.log
platformio.ini
sample_log.txt M
WWWESPSetupServer.h
WWWESPSetupServer.cpp
main.cpp
platformio.ini
ESPPreferences.cpp
sample_log.txt
ESPPreferences.h
Config.cpp
SasKeyForIotHub.cpp
src > main.cpp > setup()
#include "WWWESPSetupServer.h"
#include "log.h"
#include "IoTBlobHelper.h"
WWWESPSetupServer www;
IoTBlobHelper blob;
void setup() {
Serial.begin(115200);
#ifdef DEBUG_ESP_PORT
Serial.setDebugOutput(true);
#endif
delay(5000);
DEBUG_MSG("START");
DEBUG_MSG("LoadConfig form EEPROM");
Config::LoadConfig();
DEBUG_MSG("Init WiFi / WebServer");
www.Init();
if (www.HasWifi()) {
//Init devices
DEBUG_MSG("Wifi, try to upload to azure");
blob.UploadBlob("20200913.txt","ABC"); //ioth8266r01
} else {
ERROR_MSG("WiFi needs to be configured!");
}
}
void loopIoTHub() {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
PS Z:\TSGITHUB\2020IotHardware\ESP32\Esp32WifiIoTBlob>
OUTLINE
TIMELINE
AZURE IOT HUB
master*
Live Share
Ln 26, Col 57
Spaces: 2
UTF-8
CRLF
C++
Go Live
Win32
ENG


File Edit Selection View Go Run Terminal Help
main.cpp - Esp32WifiIoTBlob - Visual Studio Code
PLATFORMIO
PROJECT TASKS
General
Build
Upload
Monitor
Upload and Monitor
Devices
Clean
Advanced
Remote Development
Miscellaneous
env:esp32-devkitlipo
WWWESPSetupServer.h
WWWESPSetupServer.cpp
main.cpp
platformio.ini
ESPPreferences.cpp
sample_log.txt
ESPPreferences.h
Config.cpp
SasKeyForIotHub.cpp
src > main.cpp > setup()
#include "WWWESPSetupServer.h"
#include "log.h"
#include "IoTBlobHelper.h"
WWWESPSetupServer www;
IoTBlobHelper blob;
void setup() {
Serial.begin(115200);
#ifdef DEBUG_ESP_PORT
Serial.setDebugOutput(true);
#endif
delay(5000);
DEBUG_MSG("START");
DEBUG_MSG("LoadConfig form EEPROM");
Config::LoadConfig();
DEBUG_MSG("Init WiFi / WebServer");
www.Init();
if (www.HasWifi()) {
//Init devices
DEBUG_MSG("Wifi, try to upload to azure");
blob.UploadBlob("20201006.txt","ABC"); //ioth8266r01
} else {
ERROR_MSG("WiFi needs to be configured!");
}
}
void loopIoTHub() {
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Task - Upload and M
[V][ssl_client.cpp:240] start_ssl_client(): Free internal heap after TLS 157680
[D][HTTPClient.cpp:1025] connect(): connected to pltkdpepliot2016S1.azure-devices.net:443
[V][ssl_client.cpp:279] send_ssl_data(): Writing HTTP request...
[V][ssl_client.cpp:279] send_ssl_data(): Writing HTTP request...
[V][HTTPClient.cpp:1123] handleHeaderResponse(): RX: 'HTTP/1.1 204 No Content'
[V][HTTPClient.cpp:1123] handleHeaderResponse(): RX: 'Content-Length: 0'
[V][HTTPClient.cpp:1123] handleHeaderResponse(): RX: 'Server: Microsoft-HTTPAPI/2.0'
[V][HTTPClient.cpp:1123] handleHeaderResponse(): RX: 'x-ms-request-id: 513a7329-6450-43fa-8ec0-7b585deb1868'
[V][HTTPClient.cpp:1123] handleHeaderResponse(): RX: 'Date: Sat, 03 Oct 2020 18:30:30 GMT'
[V][HTTPClient.cpp:1123] handleHeaderResponse(): RX: ''
[D][HTTPClient.cpp:1158] handleHeaderResponse(): code: 204
[D][HTTPClient.cpp:368] disconnect(): tcp keep open for reuse
[V][IoTBlobHelper.cpp:67] notification():
NOTIFICATION, 204
[V][ssl_client.cpp:248] stop_ssl_socket(): Cleaning SSL connection.
[V][ssl_client.cpp:248] stop_ssl_socket(): Cleaning SSL connection.
QUICK ACCESS
PIO Home
Open
PIO Account
Inspect
Projects & Configuration
Libraries
Boards
Platforms
Devices
Debug
Start Debugging
Toggle Debug Console
Updates
Library Updates
master*
Live Share
Ln 26, Col 30
Spaces: 2
UTF-8
CRLF
C++
Go Live
Win32
ENG

ESP32 + CAMERA




File Edit Selection View Go Run Terminal Help
main.cpp - ESP32CameraBlobNoLCD - Visual Studio Code
EXPLORER
OPEN EDITORS
ESP32CAMERABLOBNOLCD
.pio
.vscode
include
IoTBlobHelper.h
README
SasKeyForIotHub.h
sha256.h
lib
src
certificates.base
IoTBlobHelper.cpp
main.cpp
SasKeyForIotHub.cpp
sha256.cpp
test
README
.gitignore
.travis.yml
platformio.ini M
platformio.ini
main.cpp
src > main.cpp > setup()
log_i(".");
}
log_i("\r\nConnected to wifi");
initTime();
//Test
// IoTBlobHelper h;
// uint8_t b[] = {1, 2, 3};
// h.UploadBlob("t2", b, 3);
//Camera and LCD
char buff[256];
delay(1000);
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sscb_sda = SIOD_GPIO_NUM;
config.pin_sscb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 20000000;
config.pixel_format = PIXFORMAT_JPEG;
//init with high specs to pre-allocate larger buffers
config.frame_size = FRAMESIZE_UXGA;
config.jpeg_quality = 10;
config.fb_count = 2;
// camera init
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK)
{
log_e("Camera init failed with error 0x%x", err);
while (1)
;
}
OUTLINE
TIMELINE
AZURE IOT HUB
ENG
master*
Live Share
Ln 73, Col 4
Spaces: 2
UTF-8
CRLF
C++
Go Live
Win32

MQTT – but without MS Libraries

Why? IoT SDK:

```
RAM:   [= ]  12.7% (used 41744 bytes from 327680 bytes)
Flash: [=======   ]  70.9% (used 928822 bytes from 1310720 bytes)
```

(and – it was almost MINIMAL program + IOT Library without BLOB)

[PubSubClient](#) – open source library, MQTT

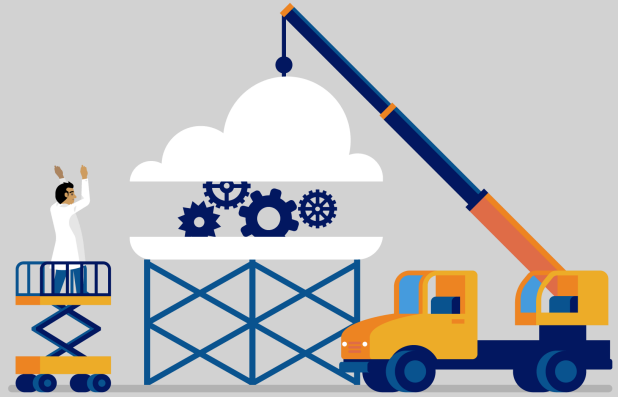
```
// DNS Name of MQTT Server = IoT Hub Name
MQTT_SERVER      // Config::IOT_HUB_NAME + ".azure-devices.net";
// User is combination of DNS Name and device name (password – SAS – see next slide)
USER_NAME// Config::IOT_HUB_NAME + ".azure-devices.net/" + Config::DEVICE_NAME;
// Topic name for receiving cloud to device messages (and – then methods etc)
MQTTReceiveTopic // "devices/" + Config::DEVICE_NAME + "/messages/#";
// Topic name for sending telemetry
MQTTSendTopic    // "devices/"+ Config::DEVICE_NAME + "/messages/events/";
```

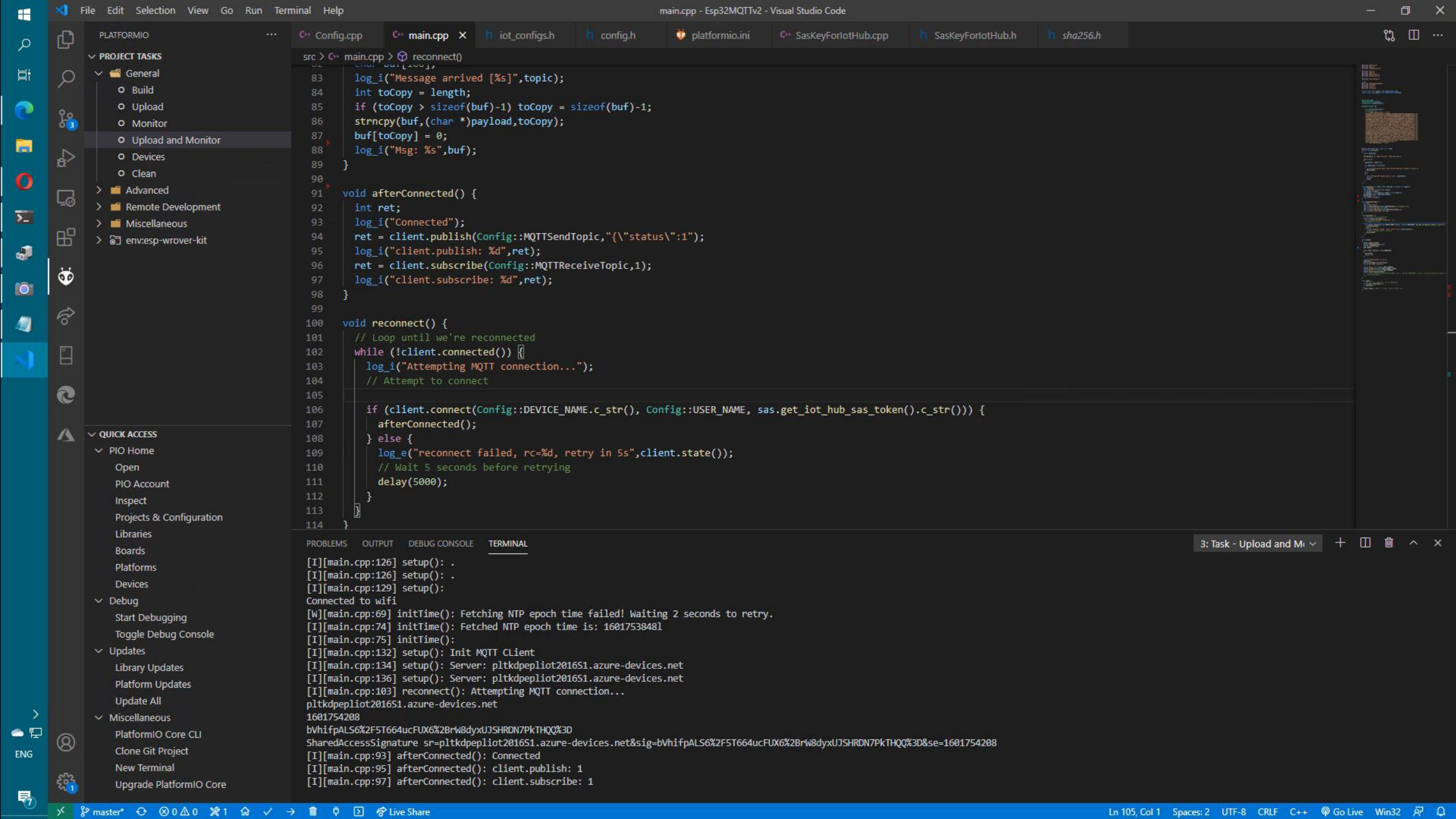
MQTT

```
WiFiClient wifi; WiFiClientSecure wifissl; PubSubClient client(wifissl); SasKeyForIotHub sas;

wifissl.setCACert(certificates);
client.setServer(Config::MQTT_SERVER, 8883);
client.setCallback(callback);

void callback(char* topic, byte* payload, unsigned int length) {
  char buf[100];
  log_i("Message arrived [%s]",topic);
  int toCopy = length; if (toCopy > sizeof(buf)-1) toCopy = sizeof(buf)-1; strncpy(buf,(char *)payload,toCopy); buf[toCopy] = 0;
  log_i("Msg: %s",buf);
}
void afterConnected() {
  client.publish(Config::MQTTSendTopic,"{\"status\":1"); client.subscribe(Config::MQTTReceiveTopic,1); }

void reconnect() {
  while (!client.connected()) {
    if (client.connect(Config::DEVICE_NAME.c_str(), Config::USER_NAME, sas.get_iot_hub_sas_token().c_str())) { afterConnected();
    } else {delay(5000); }  } }

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

Demo

ESP32 and pure MQTT (no MS library)




File Edit Selection View Go Run Terminal Help
main.cpp - Esp32MQTTv2 - Visual Studio Code
PLATFORMIO
PROJECT TASKS
General
Build
Upload
Monitor
Upload and Monitor
Devices
Clean
Advanced
Remote Development
Miscellaneous
env:esp-wrover-kit
QUICK ACCESS
PIO Home
Open
PIO Account
Inspect
Projects & Configuration
Libraries
Boards
Platforms
Devices
Debug
Start Debugging
Toggle Debug Console
Updates
Library Updates
Platform Updates
Update All
Miscellaneous
PlatformIO Core CLI
Clone Git Project
New Terminal
Upgrade PlatformIO Core
Config.cpp
main.cpp
iot_configs.h
config.h
platformio.ini
SasKeyForIotHub.cpp
SasKeyForIotHub.h
sha256.h
src > main.cpp > reconnect()
log_i("Message arrived [%s]",topic);
int toCopy = length;
if (toCopy > sizeof(buf)-1) toCopy = sizeof(buf)-1;
strncpy(buf,(char *)payload,toCopy);
buf[toCopy] = 0;
log_i("Msg: %s",buf);
}
void afterConnected() {
int ret;
log_i("Connected");
ret = client.publish(Config::MQTTSendTopic,"{\"status\":1");
log_i("client.publish: %d",ret);
ret = client.subscribe(Config::MQTTReceiveTopic,1);
log_i("client.subscribe: %d",ret);
}
void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
log_i("Attempting MQTT connection...");
// Attempt to connect
if (client.connect(Config::DEVICE_NAME.c_str(), Config::USER_NAME, sas.get_iot_hub_sas_token().c_str())) {
afterConnected();
} else {
log_e("reconnect failed, rc=%d, retry in 5s",client.state());
// Wait 5 seconds before retrying
delay(5000);
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
3: Task - Upload and M
[I][main.cpp:126] setup(): .
[I][main.cpp:126] setup(): .
[I][main.cpp:129] setup():
Connected to wifi
[W][main.cpp:69] initTime(): Fetching NTP epoch time failed! Waiting 2 seconds to retry.
[I][main.cpp:74] initTime(): Fetched NTP epoch time is: 1601753848l
[I][main.cpp:75] initTime():
[I][main.cpp:132] setup(): Init MQTT CLient
[I][main.cpp:134] setup(): Server: pltkdpepliot2016S1.azure-devices.net
[I][main.cpp:136] setup(): Server: pltkdpepliot2016S1.azure-devices.net
[I][main.cpp:103] reconnect(): Attempting MQTT connection...
pltkdpepliot2016S1.azure-devices.net
1601754208
bVhifpALS6%2F5T664ucFUX6%2BrW8dyxUJSHRDN7PkTHQQ%3D
SharedAccessSignature sr=pltkdpepliot2016S1.azure-devices.net&sig=bVhifpALS6%2F5T664ucFUX6%2BrW8dyxUJSHRDN7PkTHQQ%3D&se=1601754208
[I][main.cpp:93] afterConnected(): Connected
[I][main.cpp:95] afterConnected(): client.publish: 1
[I][main.cpp:97] afterConnected(): client.subscribe: 1
master*
Live Share
Ln 105, Col 1
Spaces: 2
UTF-8
CRLF
C++
Go Live
Win32
ENG

Smallest (ESP01, d1, etc)

No exceptions!

Code – almost the same

IoT Hub SDK Api working (LL)

Worth: Remove additional libraries

And – compile own library

(remove logging)

(remove exceptions)

(or use pure, simplest MQTT or even HTTP (for telemetry))

```
[env:wemosbat]
platform = espressif8266
board = d1
framework = arduino
;    -fexceptions - NO + 200KB per eh_table

build_flags =
    -DDONT_USE_UPLOADTOBLOB
    -DUSE_BALTIMORE_CERT
    -w
    -IInclude
    -Ilib/AzureIoTHub/src
    -Ilib/AzureIoTProtocol_HTTP/src
    -Ilib/AzureIoTProtocol_MQTT/src
    -Ilib/AzureIoTUtility/src
upload_port = COM[5]

;-Ilib/AzureIoTSocket_WiFi/src
; //edit round() in Arduino.h
;lib_ldf_mode = off
lib_deps =
    lib/AzureIoTHub
    lib/AzureIoTProtocol_MQTT
    lib/AzureIoTProtocol_HTTP
    lib/AzureIoTUtility
    ESP8266WiFi
    ArduinoJson
```

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-develop-for-constrained-devices

“Other” devices, language

ARM, PC -> better to use docker and maybe high level language

Azure IoT SDK for C written in ANSI C (C99) for portability and broad platform compatibility. There are two device client libraries for C, the low-level iothub_ll_client and the iothub_client (threaded).

Azure IoT SDK for Python

Azure IoT SDK for Node.js

Azure IoT SDK for Java

Azure IoT SDK for .NET

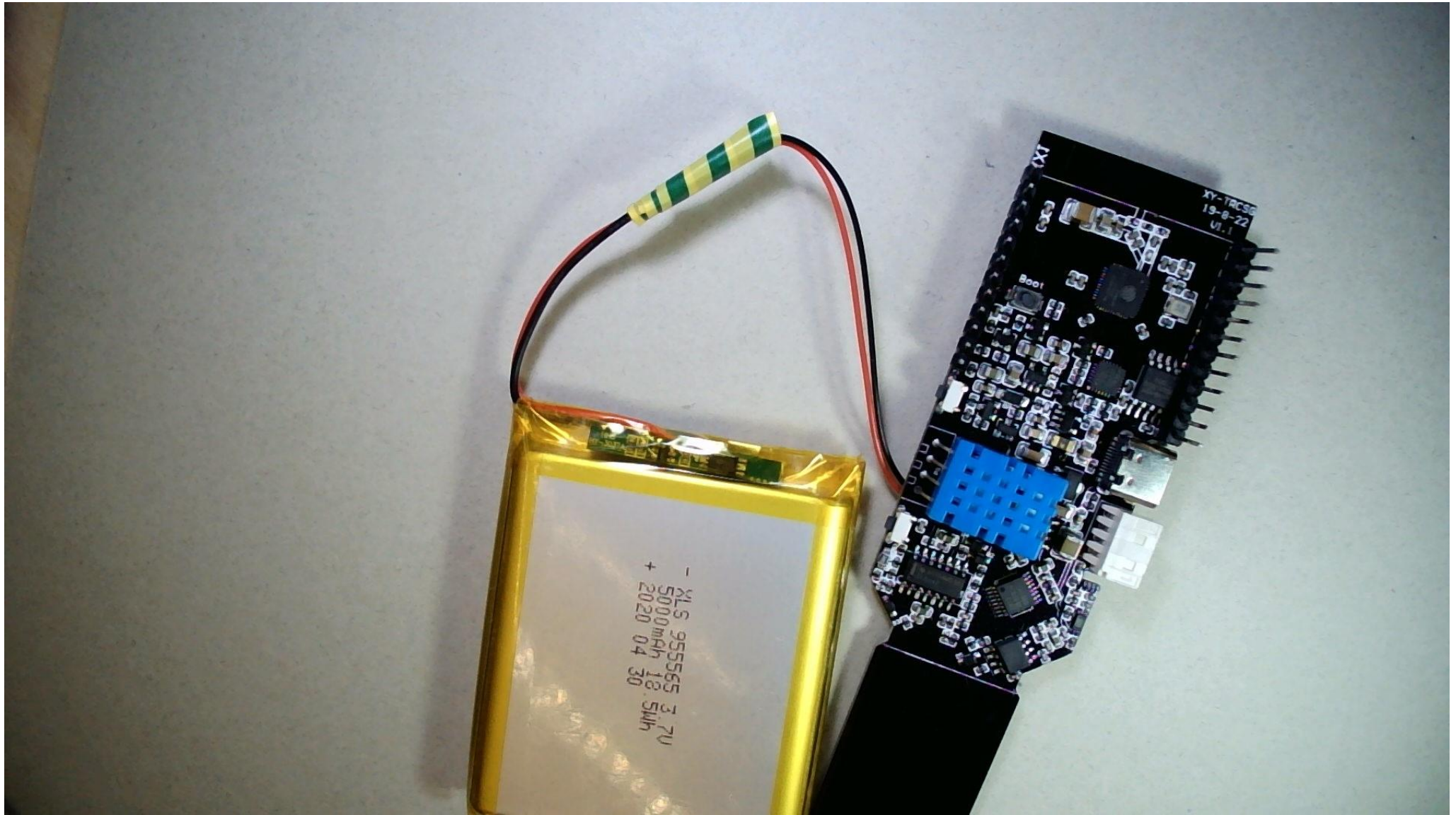
```
import iothub_client
# pylint: disable=E0611
from iothub_client import IoTHubClient, IoTHubClientError, IoTHubTransportProvider, IoTHubClientResult
from iothub_client import IoTHubMessage, IoTHubMessageDispositionResult, IoTHubError, DeviceMethodReturnValue

client = IoTHubClient(CONNECTION_STRING, PROTOCOL)
message = createMessage();

# Add a custom application property to the message.
prop_map = message.properties()
prop_map.add("ABC", ("%.2f" % random.random()))

# Send the message.
client.send_event_async(message, send_confirmation_callback, None)
```

Watchdog and power consumption (ESP32)

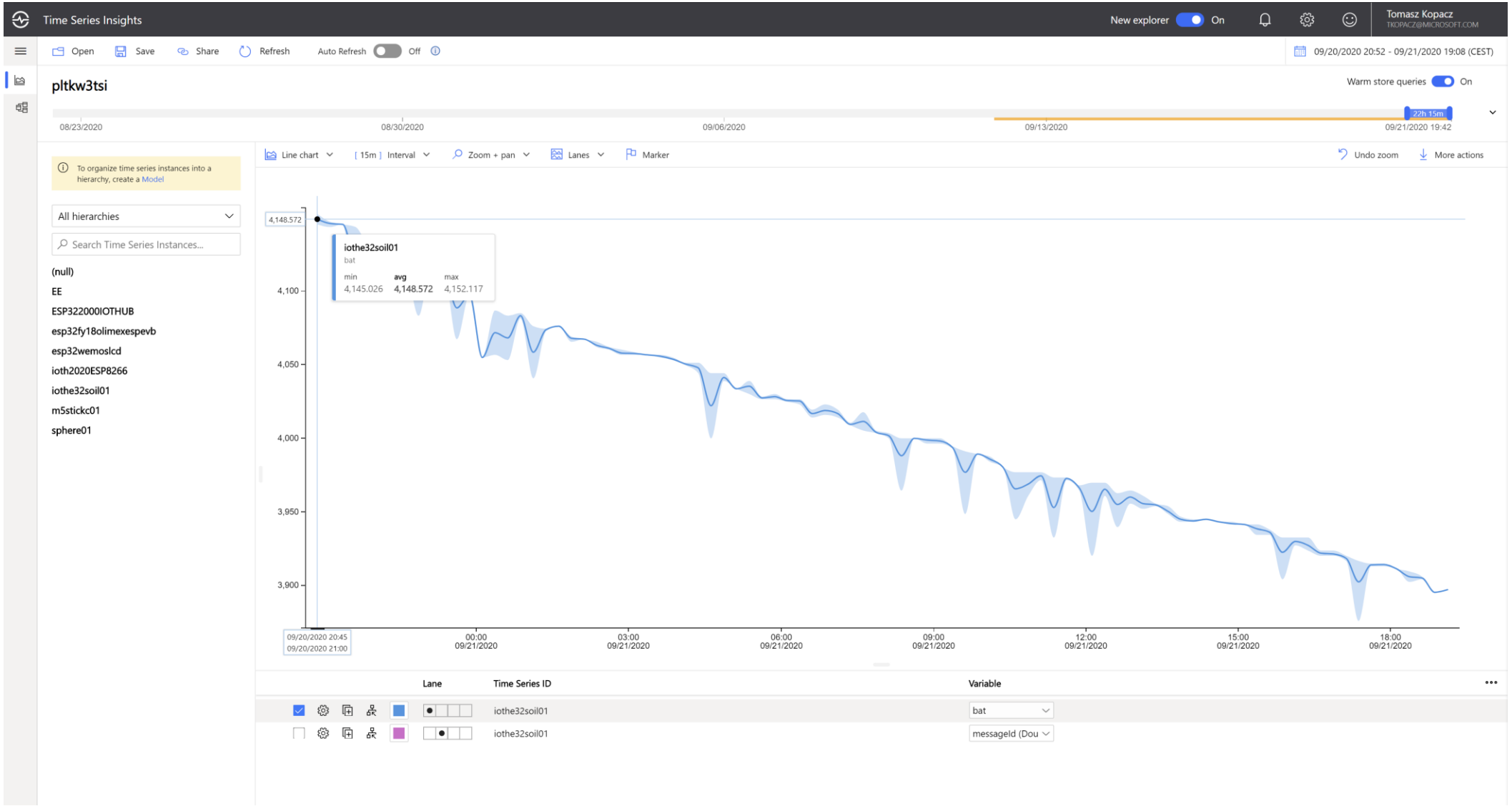
```
  esp_sleep_wakeup_cause_t wakeup_reason; wakeup_reason = esp_sleep_get_wakeup_cause();
  switch (wakeup_reason)  {
    case 4: DEBUG_MSG("ESP_SLEEP_WAKEUP_TIMER, Wakeup caused by timer"); break;
    case 6: DEBUG_MSG("ESP_SLEEP_WAKEUP_ULP, Wakeup caused by ULP program"); break;
    case 11: DEBUG_MSG("ESP_SLEEP_WAKEUP_COCPU_TRAP_TRIG, Wakeup caused by COCPU crash"); break;
  }
  esp_task_wdt_init(Config::WD_TIMEOUT, true); //enable panic so ESP32 restarts
  //Watchdogs in callbacks
  esp_task_wdt_add(NULL); //add current thread to WDT watch
  esp_sleep_enable_timer_wakeup(Config::INTERVAL_DEEP_SLEEP_S * uS_TO_S_FACTOR);
...
static void connection_status_callback(IOTHUB_CLIENT_CONNECTION_STATUS result, IOTHUB_CLIENT_CONNECTION_STATUS_REASON reason, void *user_
context)
{
  esp_task_wdt_add(NULL); if (result == IOTHUB_CLIENT_CONNECTION_AUTHENTICATED) esp_task_wdt_reset();

...
 if (result != IOTHUB_CLIENT_CONFIRMATION_OK)  esp_restart();
```

```
delay(Config::INTERVAL_MS);

OR

esp_deep_sleep_start();
```

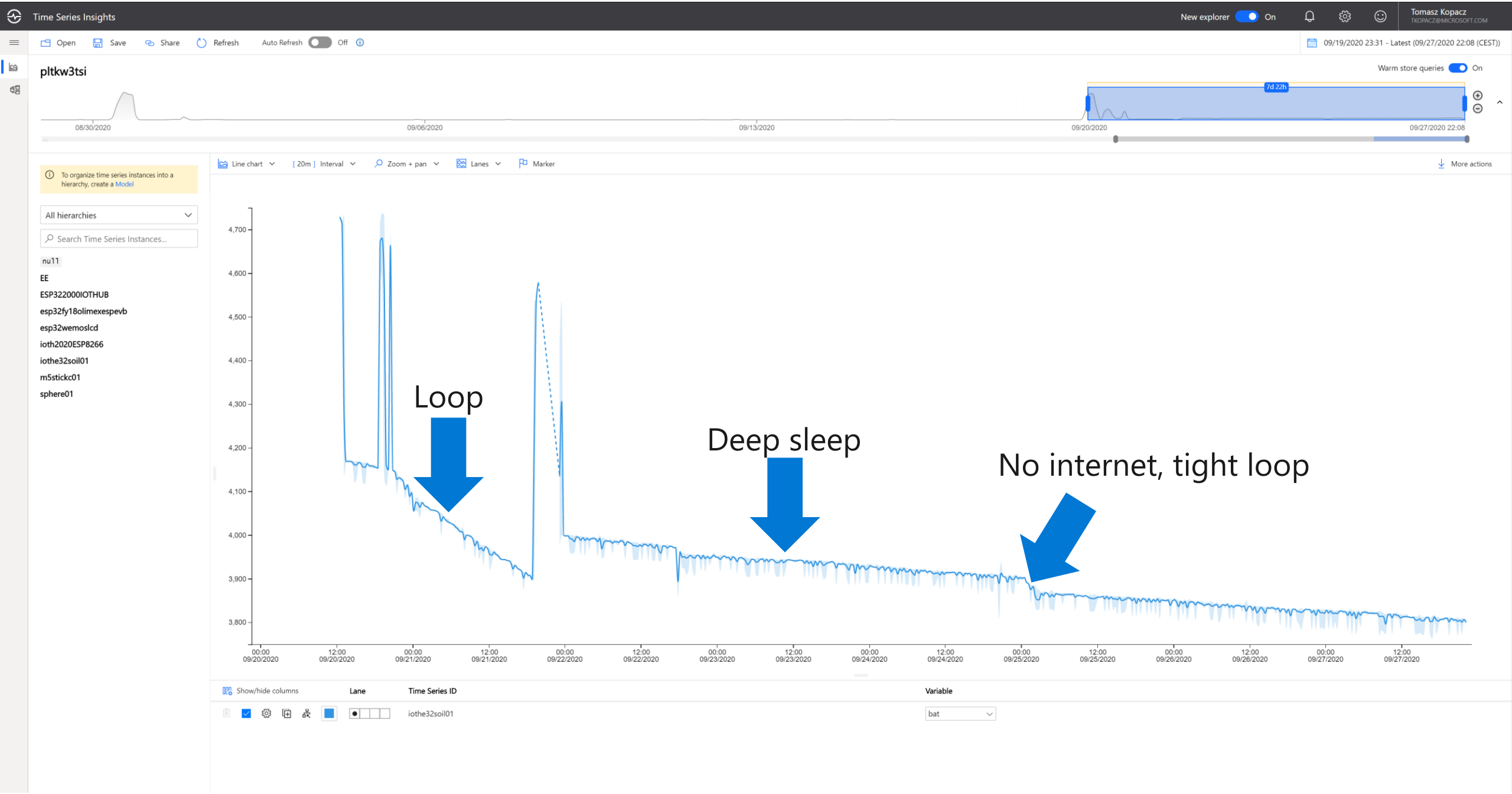
Device – soil monitoring + LiPo battery



Traditional “sleep” loop, 3min, 22H, usage 11.44mA/H

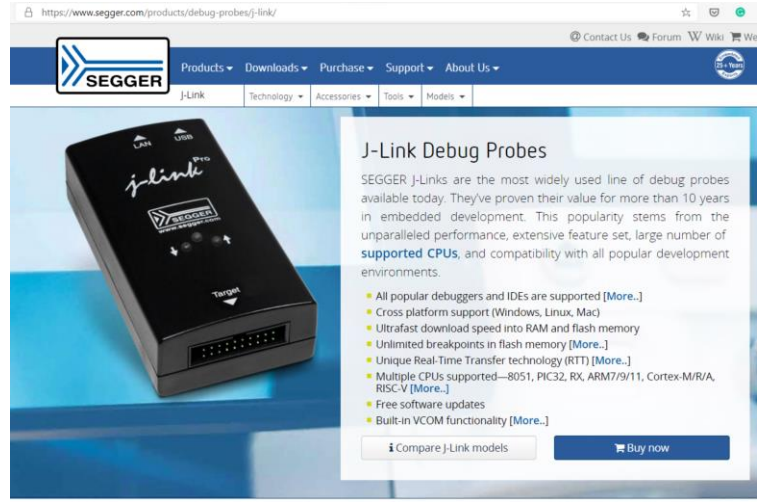


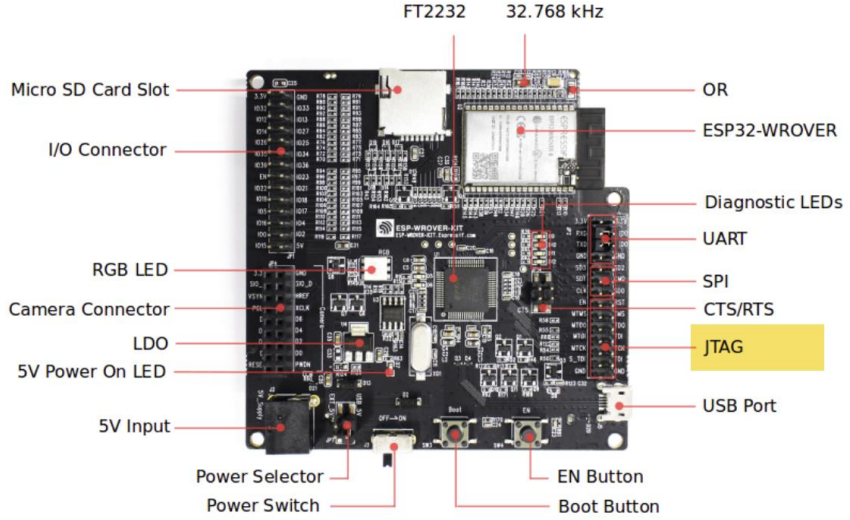
Deep “sleep” loop, 3min, 22H, usage 2.11mA/H, 1.68mA based on 285h

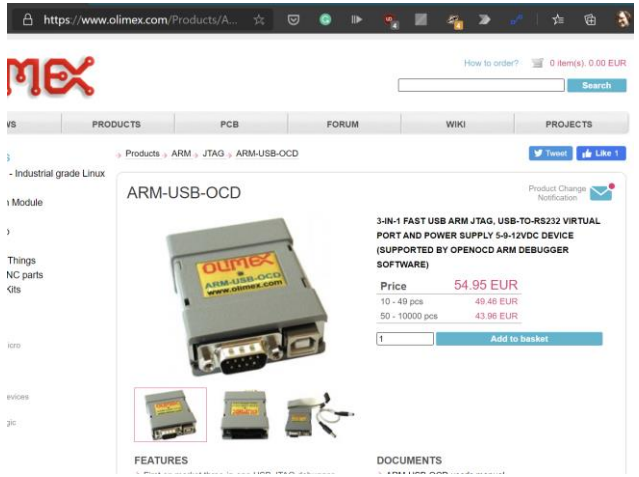

Time Series Insights
New explorer On
Tomasz Kopacz
TKOPACZ@MICROSOFT.COM
Open
Save
Share
Refresh
Auto Refresh Off
09/19/2020 23:31 - Latest (09/27/2020 22:08 (CEST))
pltkw3tsi
Warm store queries On
7d 22h
08/30/2020
09/06/2020
09/13/2020
09/20/2020
09/27/2020 22:08
To organize time series instances into a hierarchy, create a Model
All hierarchies
Search Time Series Instances...
null
EE
ESP322000IOTHUB
esp32fy18olimexespevb
esp32wemoslcd
ioth2020ESP8266
iothe32soil01
m5stickc01
sphere01
Line chart
[20m] Interval
Zoom + pan
Lanes
Marker
More actions
Loop
Deep sleep
No internet, tight loop
4,700
4,600
4,500
4,400
4,300
4,200
4,100
4,000
3,900
3,800
00:00 09/20/2020
12:00 09/20/2020
00:00 09/21/2020
12:00 09/21/2020
00:00 09/22/2020
12:00 09/22/2020
00:00 09/23/2020
12:00 09/23/2020
00:00 09/24/2020
12:00 09/24/2020
00:00 09/25/2020
12:00 09/25/2020
00:00 09/26/2020
12:00 09/26/2020
00:00 09/27/2020
12:00 09/27/2020
Show/hide columns
Lane
Time Series ID
Variable
iothe32soil01
bat

Dev notes

1. Debugging (JTAG)
2. Logging (memory!, Serial port(s))
3. Exceptions???
4. Power (consumption)
5. [IoT throttles and quotas]

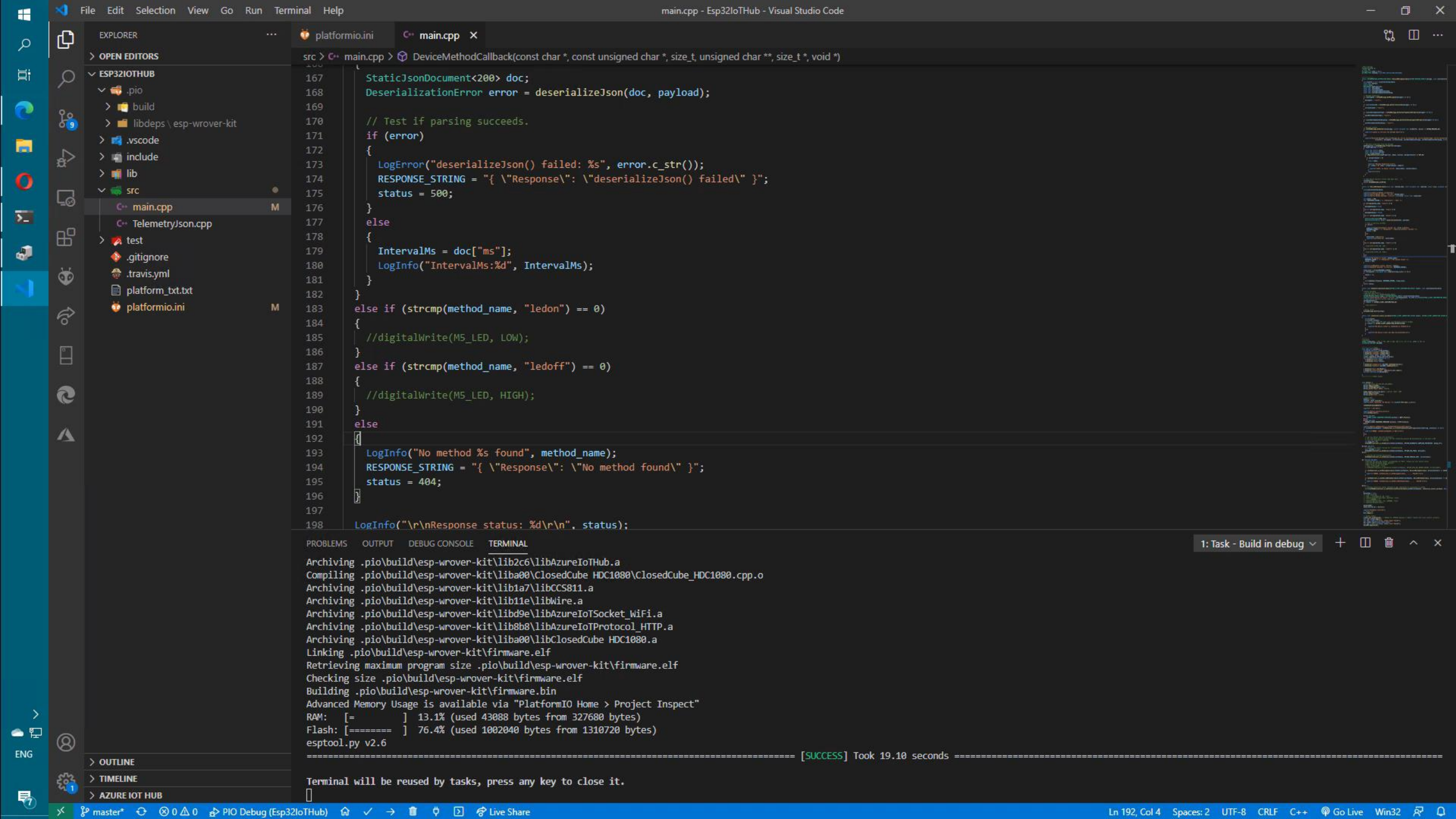






Demo

ESP-WROVER-KIT and debugger (PIO)


File Edit Selection View Go Run Terminal Help
main.cpp - Esp32IoTHub - Visual Studio Code
EXPLORER
OPEN EDITORS
ESP32IOTHUB
.pio
build
libdeps \ esp-wrover-kit
.vscode
include
lib
src
main.cpp M
TelemetryJson.cpp
test
.gitignore
.travis.yml
platform_txt.txt
platformio.ini M
platformio.ini
main.cpp
src > main.cpp > DeviceMethodCallback(const char *, const unsigned char *, size_t, unsigned char **, size_t *, void *)
167 StaticJsonDocument<200> doc;
168 DeserializationError error = deserializeJson(doc, payload);
169
170 // Test if parsing succeeds.
171 if (error)
172 {
173 LogError("deserializeJson() failed: %s", error.c_str());
174 RESPONSE_STRING = "{ \"Response\": \"deserializeJson() failed\" }";
175 status = 500;
176 }
177 else
178 {
179 IntervalMs = doc["ms"];
180 LogInfo("IntervalMs:%d", IntervalMs);
181 }
182 }
183 else if (strcmp(method_name, "ledon") == 0)
184 {
185 //digitalWrite(M5_LED, LOW);
186 }
187 else if (strcmp(method_name, "ledoff") == 0)
188 {
189 //digitalWrite(M5_LED, HIGH);
190 }
191 else
192 {
193 LogInfo("No method %s found", method_name);
194 RESPONSE_STRING = "{ \"Response\": \"No method found\" }";
195 status = 404;
196 }
197
198 LogInfo("\r\nResponse status: %d\r\n", status);
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: Task - Build in debug
Archiving .pio\build\esp-wrover-kit\lib2c6\libAzureIoTHub.a
Compiling .pio\build\esp-wrover-kit\liba00\ClosedCube HDC1080\ClosedCube_HDC1080.cpp.o
Archiving .pio\build\esp-wrover-kit\lib1a7\libCCS811.a
Archiving .pio\build\esp-wrover-kit\lib11e\libWire.a
Archiving .pio\build\esp-wrover-kit\libd9e\libAzureIoTSocket_WiFi.a
Archiving .pio\build\esp-wrover-kit\lib8b8\libAzureIoTProtocol_HTTP.a
Archiving .pio\build\esp-wrover-kit\liba00\libClosedCube HDC1080.a
Linking .pio\build\esp-wrover-kit\firmware.elf
Retrieving maximum program size .pio\build\esp-wrover-kit\firmware.elf
Checking size .pio\build\esp-wrover-kit\firmware.elf
Building .pio\build\esp-wrover-kit\firmware.bin
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [=] 13.1% (used 43088 bytes from 327680 bytes)
Flash: [========] 76.4% (used 1002040 bytes from 1310720 bytes)
esptool.py v2.6
[SUCCESS] Took 19.10 seconds
Terminal will be reused by tasks, press any key to close it.
OUTLINE
TIMELINE
AZURE IOT HUB
master* 0 0 PIO Debug (Esp32IoTHub) Live Share
Ln 192, Col 4 Spaces: 2 UTF-8 CRLF C++ Go Live Win32
ENG

“Summary”

Samples/scenarios:

1. IoT Hub SDK + C# - client
2. Time Series Insight (Gen2)
3. IoT Hub SDK + C# - service (managing devices) – jobs etc.
4. Small device – how to connect to WiFi
5. ESP32 + IoT Hub SDK
6. Large binary object, Camera, SAS
7. Pure MQTT + small device + IoT Hub
8. (almost demo) – power & deep sleep
9. Debugging on devices

What I skipped:

Concept of Modules (nested “devices” in Twin)

Device Twin/Module automatic configuration

Rotation of keys for devices (important – but easy to visualize)

IoT Edge modules (but – this will be in the next session)… But – short intro

Azure IoT Hub protocol Gateway | IoT Edge

Main reason - (re) package protocol.

Translate messages (both ways)

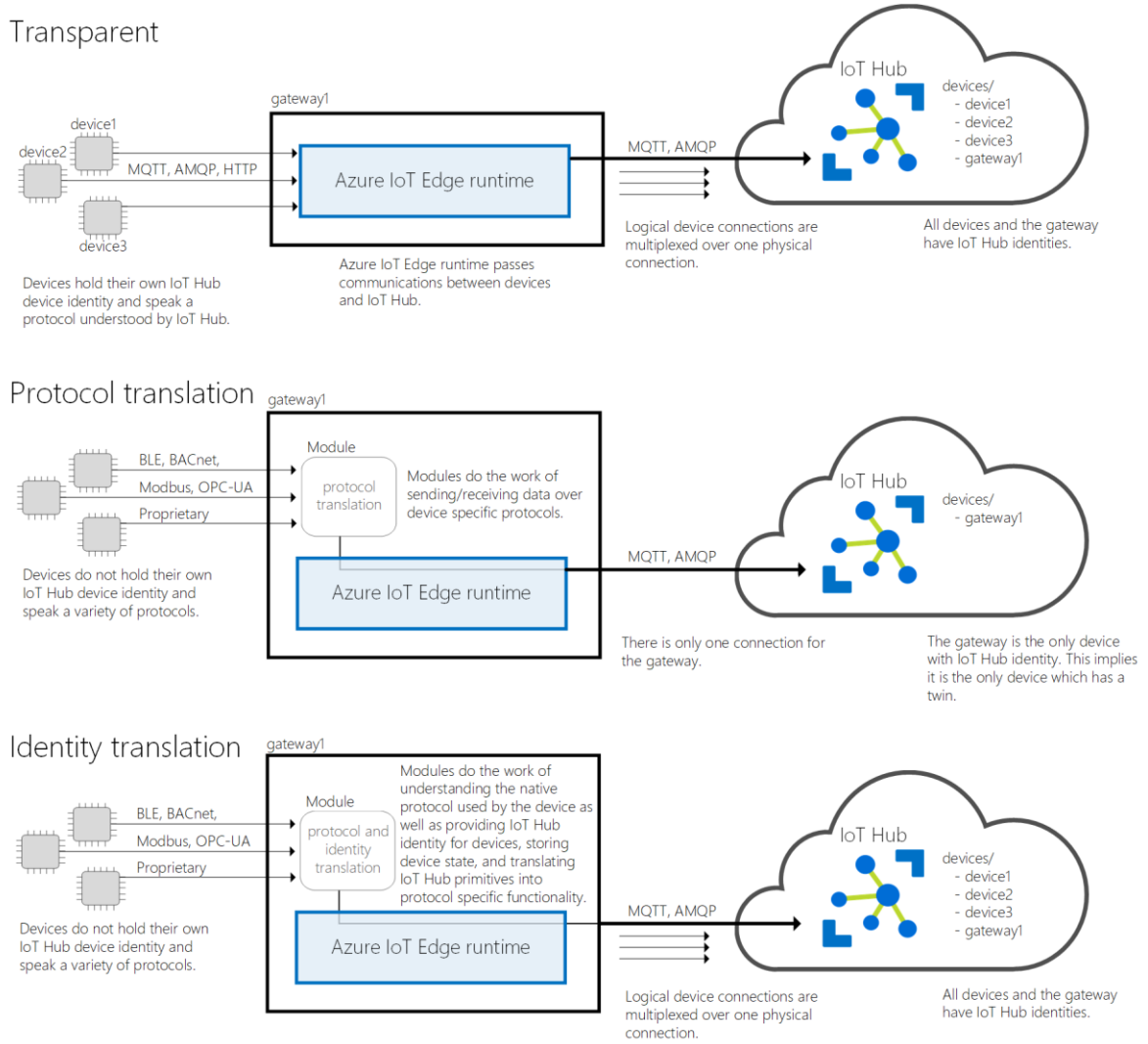
Two approaches:

Code: https://github.com/Azure/azure-iot-protocol-gateway

Docker & IoT Edge: https://docs.microsoft.com/en-us/azure/iot-edge/deploy-modbus-gateway

Or – custom “solution”

IoT Edge Device as gateway

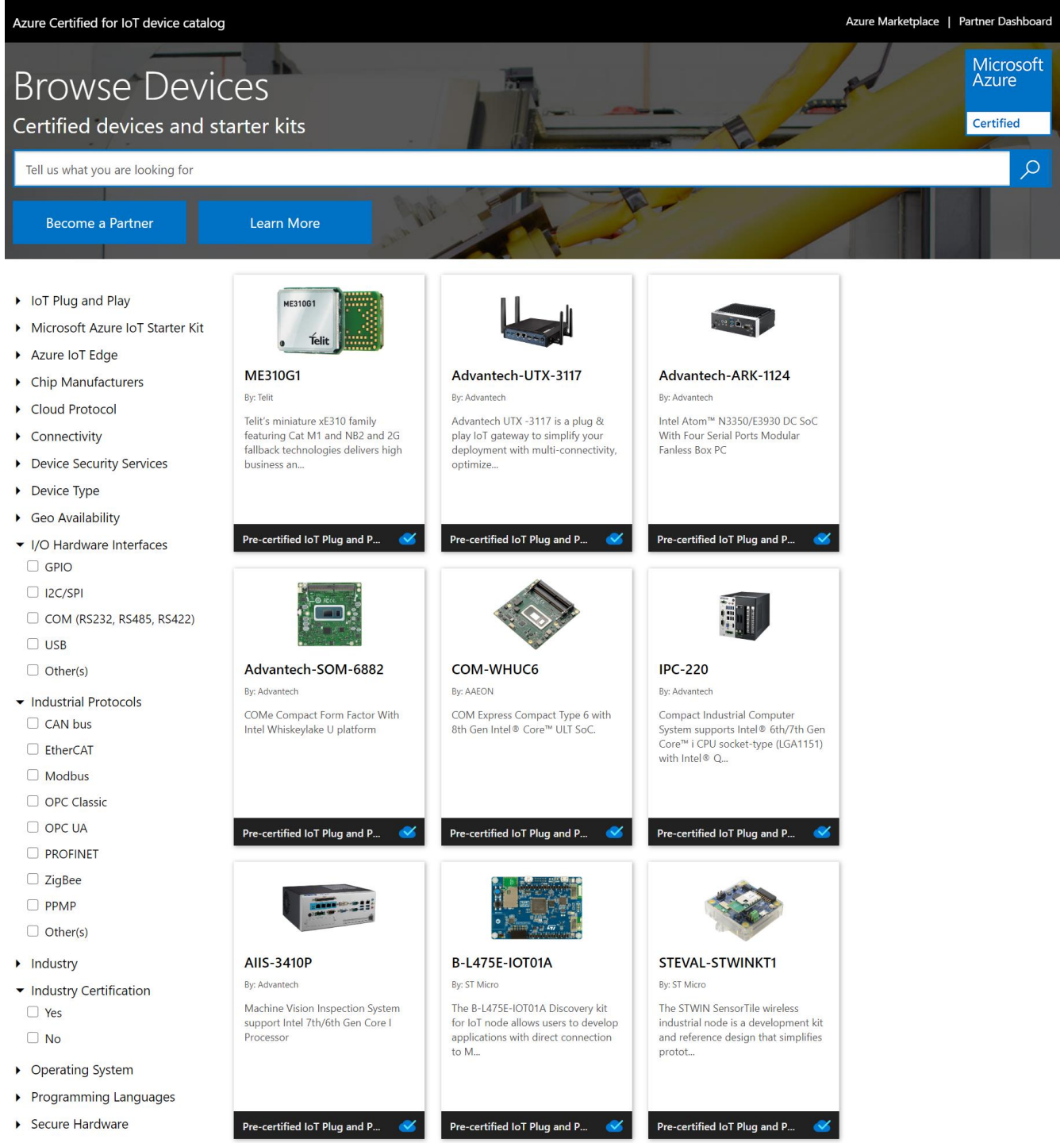


Primitive	Transparent gateway	Protocol translation	Identity translation
Identities stored in the IoT Hub identity registry	Identities of all connected devices	Only the identity of the gateway device	Identities of all connected devices
Device twin	Each connected device has its own device twin	Only the gateway has a device and module twins	Each connected device has its own device twin
Direct methods and cloud-to-device messages	The cloud can address each connected device individually	The cloud can only address the gateway device	The cloud can address each connected device individually
IoT Hub throttles and quotas	Apply to each device	Apply to the gateway device	Apply to each device

Encapsulate in IP (WiFi)

https://catalog.azureiotsolutions.com/alldevices

You don’t have to start from “scratch”



LAST SLIDE BEFOR DEMO – Szabolcs Baranyi (after break)

IoT Edge + Docker + ….