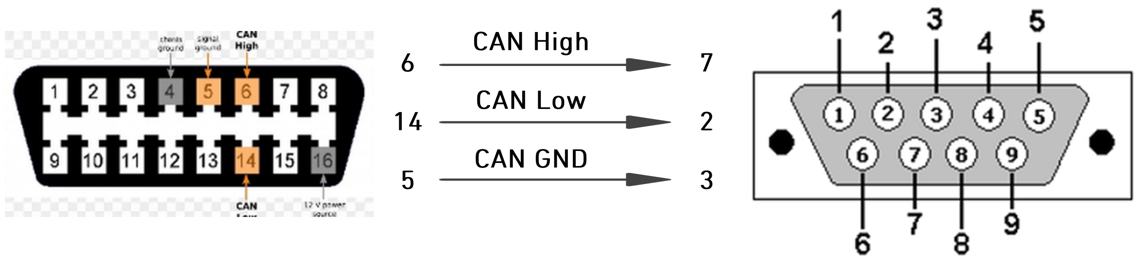
Connecting a Vehicle ECU Simulator to an IoT backend platform with the FATBOX G3 Gateway

Cloud monitoring vehicle data via CAN Bus ISO15765-4 and 4G/LTE/3G

- In this tutorial, we will set up a OZEN Elektronik OE91C1610 CAN BUS ECU simulator that is compatible with IS015765-4 (part of OBD2). Interfaces to our FATBOX G3 gateway's CAN bus interface and connect to AWS IoT.
- Our G3 gateway is based on the iMX6 CPU with built-in CAN port functionality. The CAN bus is driven by an industrial proven high speed CAN transceiver that provides a robust & reliable interface with high noise immunity.
- We will also look at **connecting the G3 IOT Gateway to a designated Cloud Service**. Supported clients are AWS IoT, Azure IoT Hub and Ubidots.
- Users have the option of backhaul via either cellular (4G/3G), WIFI or wired Ethernet, depending on version

Hardware Wiring

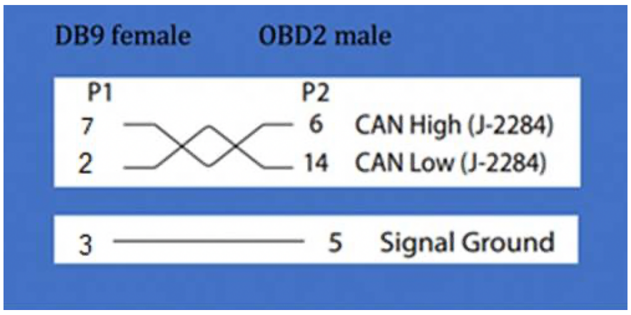


OZEN Elektronik OE91C1610 (J1962 female OBD2)

G3 CAN BUS port (DB9 male)

Above shows the CAN BUS signal connection usually implemented using shielded twisted pair cable.

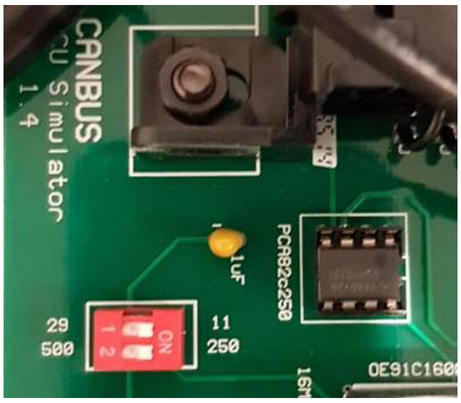
You can use a standard off-the-shelf 'OBD2 male to DB9 female cable' with pin connection as below:



DB9 female	OBD2 male	
P1	P2	
7	6	CAN High (J-2284)
2	14	CAN Low (J-2284)
3	5	Signal Ground

Ozen ECU simulator Configuration

Use the DIP switch on the board to select CAN frame ID 11-bit type.
For the CAN BUS baud rate, select 250 kbps.

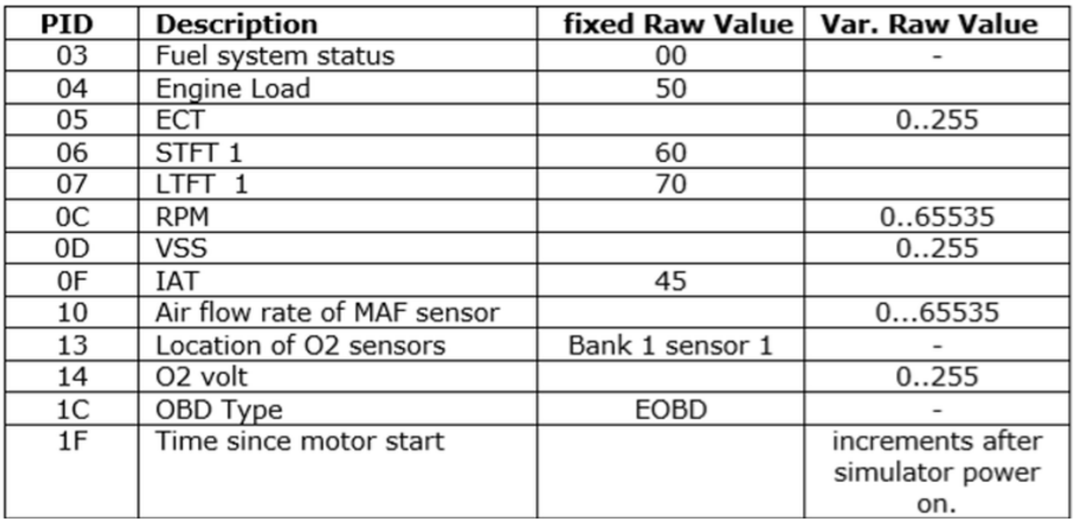


The baud rate set on the G3 gateway CAN BUS must be same.

Connect the ECU Simulator to Gateway

Our gateway uses a configuration file (\user\iotasset.txt) to map the required Parameter IDs (PIDs) to be read during each polling cycle. This flexibility allows wide compatibility with different CAN Bus devices from different manufacturers.

To write this iotasset.txt file we will be referencing the PIDs table from the product user manual (see sample below):



PID	Description	fixed Raw Value	Var. Raw Value
03	Fuel system status	00	-
04	Engine Load	50	
05	ECT		0..255
06	STFT 1	60	
07	LTFT 1	70	
0C	RPM		0..65535
0D	VSS		0..255
0F	IAT	45	
10	Air flow rate of MAF sensor		0...65535
13	Location of O2 sensors	Bank 1 sensor 1	-
14	O2 volt		0..255
1C	OBD Type	EOBD	-
1F	Time since motor start		increments after simulator power on.

And here is a sample of the iotasset.txt.

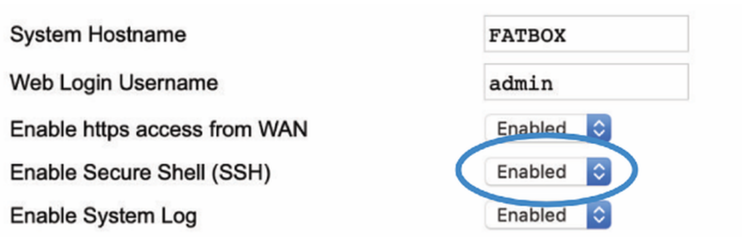
In the top example, we are configuring to read ECT (Engine Coolant Temperature) PID.

- "7DF" is the functional address of ECU which based on ISO-15765-4 11-bit protocol.
- "7E8" is the response message ID used to filter the response message from the query.
- "0105" is the CAN request ECM mode 01 & PID 05.
- Response data is located at offset +3 from first byte and length of 1 byte.
- Name data field name, as "EngineCoolantTemp".
- The gateway is able to apply a customer multiplier (x1 in this case) and offset (-40 in this case) to match the requirements of their cloud service.

```
CAN_START

TYPE,OBD
CANID,7DF,7E8
CANREQ,0105
CANDATA,3,1,UINT8,1,-40
Key,EngineCoolantTemp
Unit,degF

TYPE,OBD
CANID,7DF,7E8
CANREQ,010C
CANDATA,3,2,UINT16HL,0.25,0
Key,EngineRPM
Unit,rpm

TYPE,OBD
CANID,7DF,7E8
CANREQ,010D
CANDATA,3,1,UINT8
Key,VehicleSpeed
Unit,mph

CAN_STOP
```

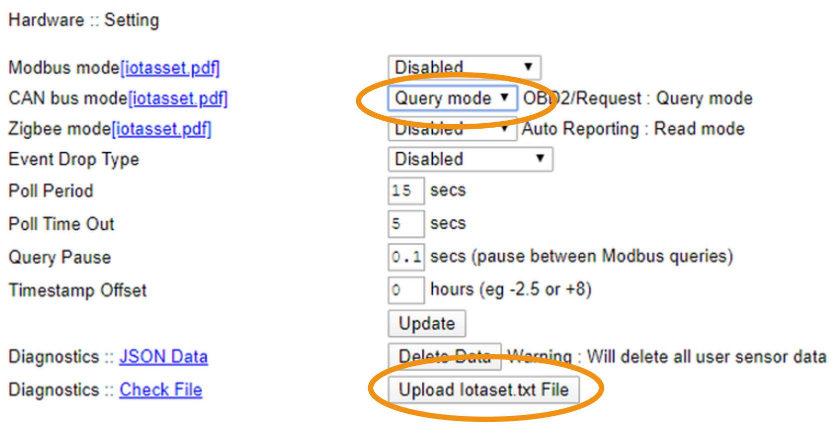
Once you have configured your device configuration file, we can update to your FATBOX gateway securely over the air. The gateway HTTPS connected web console must be accessible, if you do not have an Internet connection, you can follow the alternative steps here).

Log into the G3 (the default address is 192.168.1.1) and enter in the user name and password. Go to the <Management> tab in the G3 web configuration menu. Enable the SSH option (see below) and click the 'UPDATE' button to save your settings. Then reboot the gateway.



System Hostname	FATBOX
Web Login Username	admin
Enable https access from WAN	Enabled
Enable Secure Shell (SSH)	Enabled
Enable System Log	Enabled

Then log in again, go to the <IOT Hardware> menu to set the CAN bus mode to "Query mode" and register the polling configuration as required by the user. Click 'UPDATE' to save your settings.
(Note that all other interfaces like Modbus and Zigbee can also run concurrently).

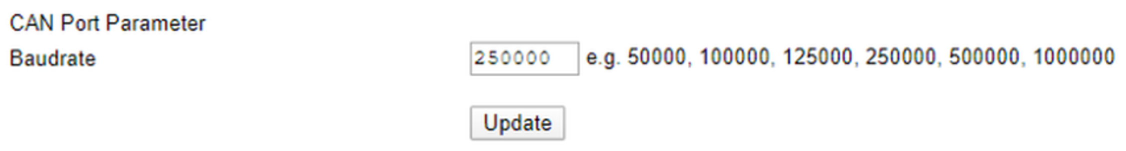


Hardware :: Setting

Modbus mode[iotasset.pdf]	Disabled
CAN bus mode[iotasset.pdf]	Query mode OBD2/Request : Query mode
Zigbee mode[iotasset.pdf]	Disabled Auto Reporting : Read mode
Event Drop Type	Disabled
Poll Period	15 secs
Poll Time Out	5 secs
Query Pause	0.1 secs (pause between Modbus queries)
Timestamp Offset	0 hours (eg -2.5 or +8)
	Update
Diagnostics :: JSON Data	Delete Data Warning : Will delete all user sensor data
Diagnostics :: Check File	Upload Iotaset.txt File

Next, click "Upload iotasset.txt" button and in the new window, use the 'CHOOSE FILE' tab to select the above prepared file from your local folder to to send over to G3. After the upload is successful you will need to close the page and log in again for security purpose. The user can also confirm the status of iotasset.txt file upload by clicking on Diagnostics::Check File.

We will now also configure the gateway's CAN BUS baud rate. Go to the <Port Settings> tab and enter the following settings below. Once you are done click the 'UPDATE' button to save your settings.



CAN Port Parameter

Baudrate	250000 e.g. 50000, 100000, 125000, 250000, 500000, 1000000
	Update

After these settings are updated, **REBOOT** the gateway. The user can then check the CAN BUS vehicle data collected (Diagnostics::JSON Data) or delete for testing.

Connect to AWS IoT Cloud Service

Now that the connectivity between the ECU Simulator and the G3 gateway has been set, we will then look at getting the data onto a designated cloud service. We support an open customer software and 3rd party API (e.g. REST API for HTTP, HTTPS and MQTT) integration to connect to a chosen cloud data or dashboard service (or to use for on board data processing).

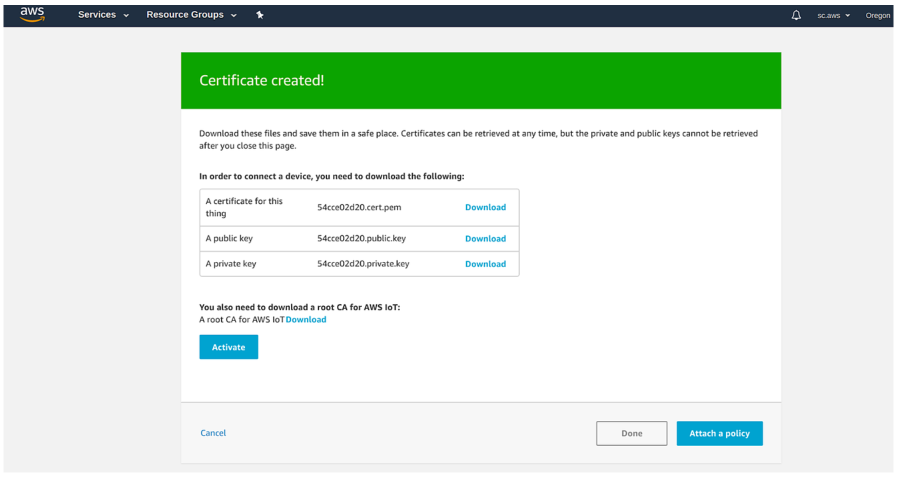
Direct deployment on Azure, AWS IoT or Ubidots can also be done as these IoT clients have been integrated on the G3 Gateway. For this project, we are going to showcase using AWS IoT (note that an AWS account is required to continue this tutorial).

1. Create a new AWS IoT Thing

First log into your AWS IoT Management Console and create a Thing:
AWS IoT > Manage > Thing > Create

Next go to:
Secure > Certificates
Then download the 'Certificate' & 'Private Key File'.

Ensure the Certificate is "Active", otherwise activate it under ACTIONS in
Things > Your Certificate > Security



Certificate created!

Download these files and save them in a safe place. Certificates can be retrieved at any time, but the private and public keys cannot be retrieved after you close this page.

In order to connect a device, you need to download the following:

A certificate for this thing	54cce02d20.cert.pem	Download
A public key	54cce02d20.public.key	Download
A private key	54cce02d20.private.key	Download

You also need to download a root CA for AWS IoT:
A root CA for AWS IoT Download

2. Update your G3 IoT Gateway

Create a new local folder and name it "AWS".
Save the downloaded certificate files into this folder and rename them as the following:

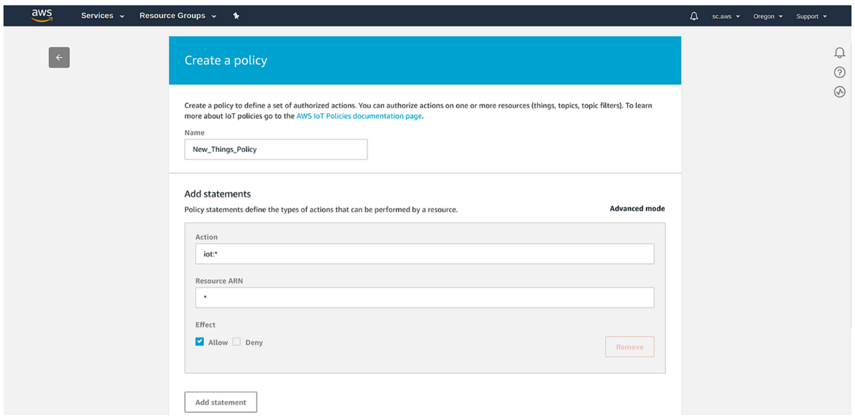
"**certificate.pem.crt**"
"**private.pem.key**"

Next zip the folder (ensure that you zip the entire folder and not just the files inside).

Then log into your FATBOX G3 web console and go to the <Management> tab.
Patch the zipped folder to the gateway using the UPDATE FIRMWARE button.

3. Create a Security Policy for your Thing

In your AWS IoT Management Console go to:
Secure > Policies > Create



Create a policy

Create a policy to define a set of authorized actions. You can authorize actions on one or more resources (things, topics, topic filters). To learn more about IoT policies go to the AWS IoT Policies documentation page.

Name: New_Things_Policy

Add statements

Policy statements define the types of actions that can be performed by a resource.

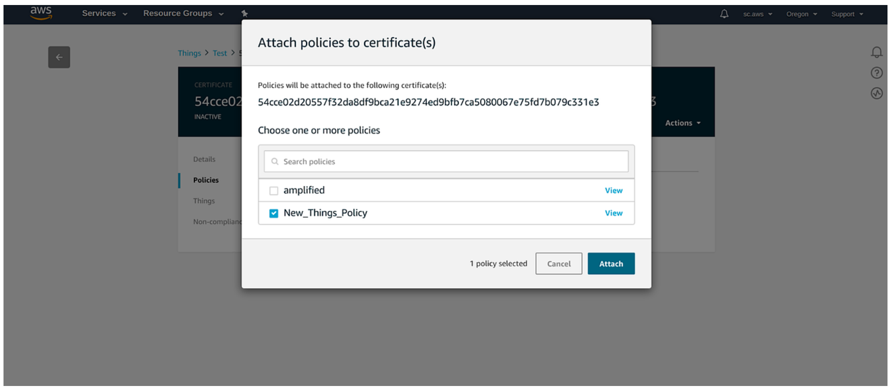
Action: iot:*

Resource ARN: *

Effect: [✓ Allow] Deny

At <Things>, select your new Thing then click on
Security > Certificates > Policies

Under <Actions>, choose to Attach a Policy:



Attach policies to certificate(s)

Policies will be attached to the following certificate(s):
54cce02d20557f32da8df9bca21e9274ed9bfb7ca5080067e75fd7b079c331e3

Choose one or more policies

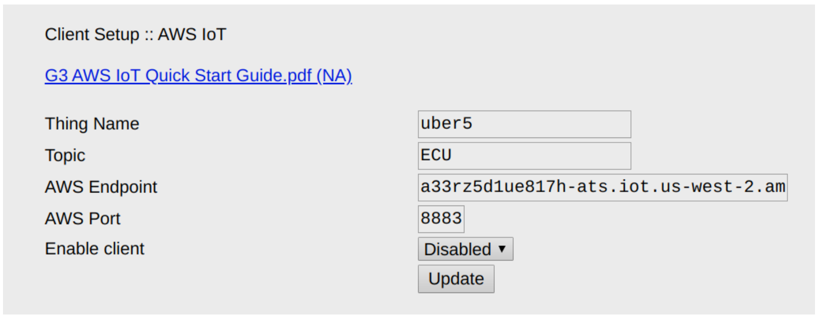
Search policies

- [ ] amplified — View
- [x] New_Things_Policy — View

1 policy selected — Cancel — Attach

4. Configure your G3 IoT Gateway to send data to your AWS Account

Now, you are ready to configure your gateway to send AWS IoT endpoint to feed data to your AWS applications. In the FATBOX G3 web configuration menu, go to the <IoT Client> tab and configure your AWS client settings according as per your AWS end point and Thing settings. Then REBOOT your gateway.

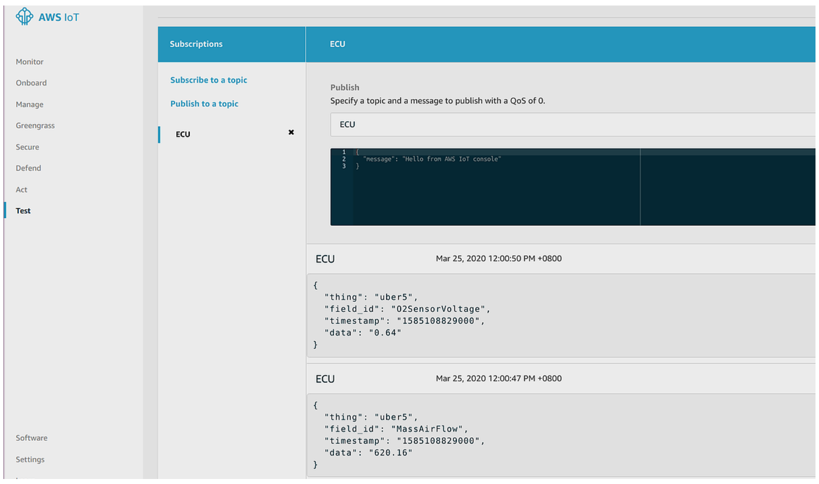


Client Setup :: AWS IoT

G3 AWS IoT Quick Start Guide.pdf (NA)

Thing Name	uber5
Topic	ECU
AWS Endpoint	a33rz5d1ue817h-ats.iot.us-west-2.am
AWS Port	8883
Enable client	Disabled ▾

Next go back to go your AWS IoT console and subscribe to the Topic to "Test" that data is being received.



ECU

Publish
Specify a topic and a message to publish with a QoS of 0.

ECU

```
{
  "message": "Hello from AWS IoT console"
}
```

ECU Mar 25, 2020 12:00:50 PM +0800

```
{
  "thing": "uber5",
  "field_id": "O2SensorVoltage",
  "timestamp": "1585108829000",
  "data": "0.64"
}
```

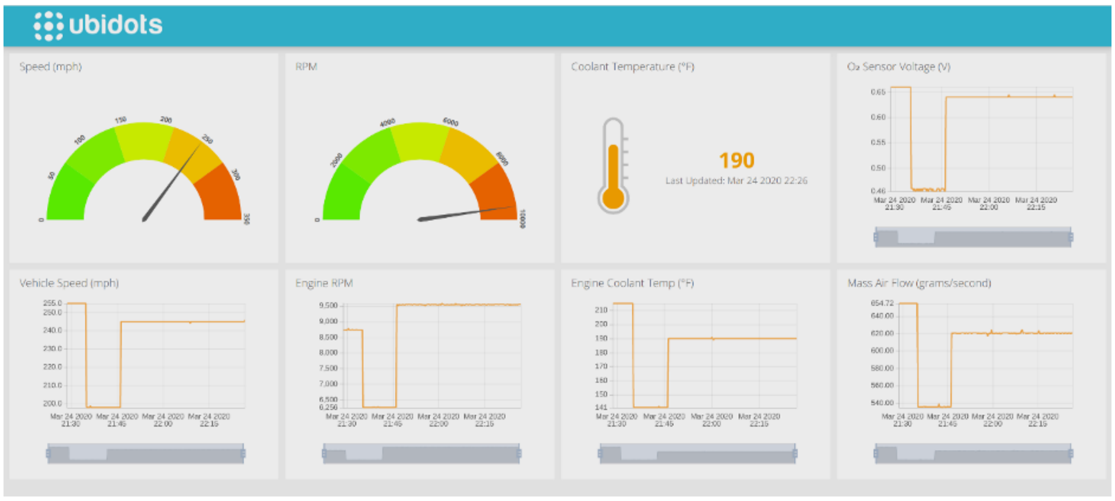
ECU Mar 25, 2020 12:00:47 PM +0800

```
{
  "thing": "uber5",
  "field_id": "MassAirFlow",
  "timestamp": "1585108829000",
  "data": "620.16"
}
```

Congratulations! You have succesfully sent your Modbus or CAN bus data to your AWS IoT endpoint and ready to, for example, push the data to a S3 bucket using a Rule in 'Act > Rules'.

The FATBOX G3 AWS IoT client side is built using AWS IoT Device SDK for Python and users are free to install, modify our device client codes for enhanced edge capabilities or other required functionalities.

See Also:: Our guides below for connecting to other supported IoT platforms includes Azure IoT and Ubidots. Here is a sample dashboard from Ubidots.


ubidots
Speed (mph)
RPM
Coolant Temperature (°F)
190
Last Updated: Mar 24 2020 22:26
O₂ Sensor Voltage (V)
Vehicle Speed (mph)
Engine RPM
Engine Coolant Temp (°F)
Mass Air Flow (grams/second)