Graduate School and Research Center In communication Systems

EURECOM
Sophia Antipolis

Huynh-Dan Vo

December 23, 2015

Thesis Report

Secure integration of Internet of Things

Confidential

Company: SAP
Supervisor: Dr. **Laurent Gomez**
Academic supervisor: Prof. **Ludovic Apvrille**
Track: Communication System Security

SAP

Abstract

Français

L'Internet des Objects tend à atteindre 50 milliards d'objets connectes et une valeur sur le marché de plus de 14 trillions de dollars d'ici 2020. McKinsey Research [1] estime que l'IoT représentera un potentiel de 11 trillion de dollars par an d'ici 2025.

Dans ce contexte, SAP Lab France co-innove en partenariat avec des acteurs majeurs du secteur publique et de l'industrie petro chimique. Un exemple de ces initiatives est le projet de maintenance prédictive avec la ville d'Antibes Juan les Pins. Les villes intelligentes cherche à avoir une meilleure vue sur leurs infrastructures, connectant leurs opérations avec leurs processus financiers.

A ce titre, nous élaborons avec la ville une solution de maintenance prédictive de leur réseau d'eau. La ville instrumente son réseau à l'aide de compteurs intelligents de pression, débit, pH. Or la sécurité joue un rôle essentiel pour la viabilité de cette solution. SAP adresse les besoins en terme de sécurité pour la protection des données, leur intégrité et disponibilité, tout en s'assurant être en conformité avec la rglementation en vigueur. Dans cette perspective, nous profitons de cette collab-

English

The Internet of Things (IoT) is expected to grow to 50 billion connected devices and $14.4 trillion in value at stake until 2020. McKinsey Research [1] "estimates a potential economic of as much as $11.1 trillion per year in 2025 for IoT applications".

In that perspective, SAP Lab France co-innovates with public sector and oil gas industries. One example of those initiatives is a project conducted together with the City of Antibes. Smart cities call for high resolution management systems, connecting operational levels with financial and controlling business processes. We elaborate with the City of Antibes a secure predictive maintenance system for their water distribution network. City of Antibes has already instrumented its water distribution network with connected smart metering devices (e.g., pressure, debit, pH),. Our joint Proof of Concept demonstrates the connectivity between their IoT devices and the SAP Hana Cloud Platform.

In that context, security is also a key enabler for the adoption of IoT by our customers. SAP needs to address their security requirements for data protection and

oration pour mettre en œuvre des solution pour garantir la sécurisation des communication de bout en bout, la gestion des objets et des logiciels embarques.

Dans le cadre de mon stage, j'ai mis en place une solution de chiffrement de bout en bout depuis les objets IoT, et l'application de contrôle du reseau d'eau.

privacy, integrity and availability while being compliant with incoming IoT regulations. On that perspective, we take advantage of those collaborations to design and develop security solutions for end to end communication, device and software life-cycle management for IoT.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Secure Predictive Maintenance for Smart City

Internet of Things (IoT) [2] has an increasing impact on multiple industries, and Predictive Maintenance is not an exception. The goal of Predictive Maintenance [3] is to make the move from reactive to predictive maintenance with IoT. It aims at leveraging remote machine sensors that monitor equipment behavior 24/7; predicting machine and equipment malfunctions before they happen; and optimizing asset maintenance and servicing - and automate operations. Nevertheless the integration of IoT with business applications raises several security challenges. Confidentiality and integrity of IoT information, identity management and authentication of devices, or secure on boarding of devices. In addition, because smart metering provides personal information of customers, individual privacy needs to be addressed.

Figure 1.1: Secure Predictive Maintenance for Smart City

2 Architecture View

2.1 Components

The figure 1.2 depicts a general view of a structure of this work. The architecture is oragnised around the following components: PLCs [4], Raspberry Pi [5], SIGFOX [6] Server, REST API [7], Dashboard, Cockpit, IoT Service [8], HANA Database [9], Authentication Proxy Server [10].

It is clear to see from Internet of Things perspective, there are programmable logic controllers including Arduino Yun [11], and Arduino Uno [12]; Raspberry Pi, SIGFOX Server - SIGFOX is a network carrier.

SAP HANA Cloud Platform [13] has reponsibilities to receive data from IoT sources, and expose the data by integrated applications. In parcular, there are SAP HCP IoT Service which is in charged of collecting data from Internet of Things sources, and storing the data in SAP HANA Database. SAP HANA XS [14] supports various services such as REST API, Dashboard, and Cockpit. It connects all services together.

The last component in the architechture is Authentication Proxy Server which plays an important role for SAP HANA Trial accounts [15]. It collects data from the REST API, and stores the data as a local server by using NodeJS. Trial accounts derive the data from this local server through SAP Cloud Connector [16] and Destination setting.

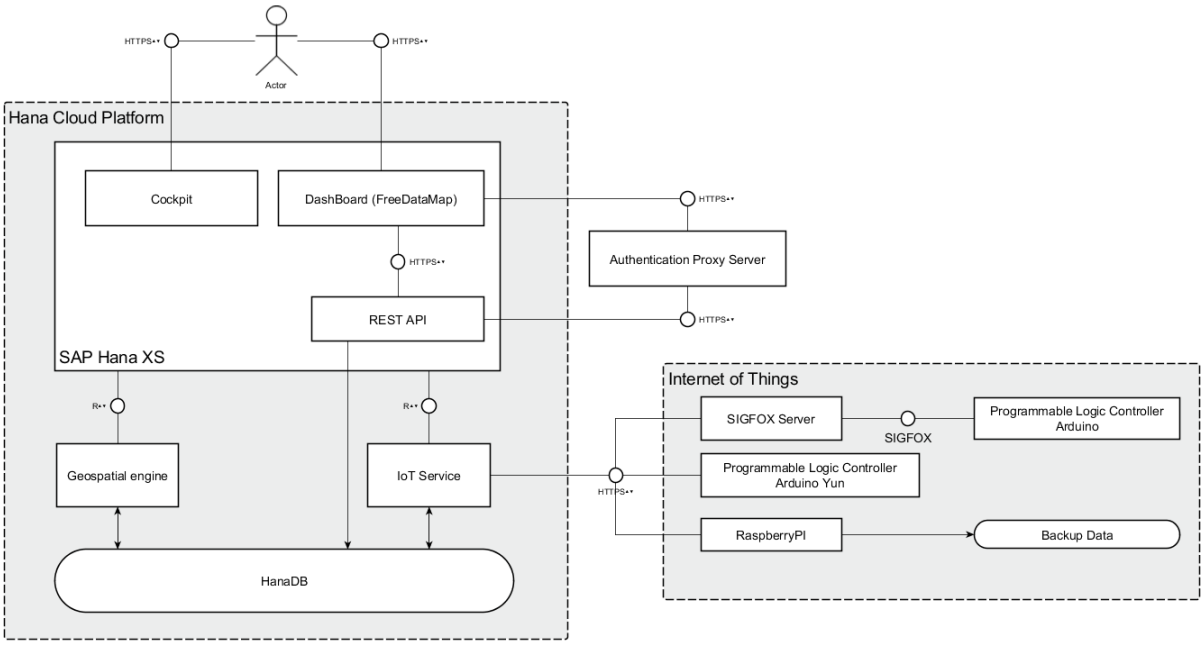

Figure 1.2: Overall Architecture

2.2 Message flows

- First of all, data from sensors is collected, and encrypted by PLCs. Backup data that is gathered and saved as files including id of devices, values, timestamps, and geolocations are extracted, encrypted by Raspberry Pi.
- The data is push to IoT Service on SAP HCP. It can be pushed directly to IoT Service via WIFI, or it can be pushed to SIGFOX, and SIGFOX pushes data to IoT Service afterwards.
- The data on SAP HCP is sent to the Dashboard. Here, it is decrypted and displayed.

3 Work Description

The objectives for this internship are listed below:

- An assessment of different scenarios, and identify associated security requirements. In this internship, we have selected a particular French Public Sector use case. It focuses on visualisation real time of water network, and predictive model for water pipeline failure. Therefore, secure integrated operations require an end-to-end information confidentiality, and water network integrity.

- An implementation of a Proof of Concept of this selected scenario. In specific, Efficient Access Control for Wireless Sensor Data [17] is implemented. An end-to-end encryption of sensor data is applied to ensure the security from devices to platforms, and the data is decrypted only when it is used.

EFFICIENT ACCESS CONTROL FOR WIRELESS SENSOR DATA

Because predictive analytics relies on customer information, this raises security requirements. The confidentiality of data is crucial. Furthermore, predictive analytics also needs to have the guarantee the integrity of data which means that the delivered data has not been altered, modified, and accurate. However, we only address confidentiality and authenticity of the message. Another difficulty is the limitation of connectivity requires a maximum message length of 12 bytes because of dependence on SIGFOX network.

1 Approaches

To begin with, Efficient Access Control for Wireless Sensor Data (EAC) [17] and Order-Preserving Symmetric Encryption (OPE) [18] are applied to encrypt data coming from Arduino. However, after being encrypted by those algorithms and base64 encoding. The final ciphertext is up to 16 bytes. This ciphertext is not suitable for the connectivity because the maximum size of a message should be 12 bytes. Therefore, an end-to-end encryption of sensor data from sensor nodes to SAP HANA Cloud Platform Internet of Things service [8] is performed with a new modification of EAC to adapt the message limitation.

2 Cryptographic Scheme

2.1 Process

HMAC SHA256 [30] is mainly used for creating keys following the EAC paper since this hash function is suitable with size limitation. Keys are generated for the en-

cryption and regenerated in order to decrypt data.

After being encrypted by EAC, the ciphertext will be added one more byte to store Sequence number as an input for decryption scheme. This means that sequence number has values from 0 to 255

In the next phase, it will be done by a base64 encoding to assure message consistency to obtain the final ciphertext.

The figure 2.3 depicts the structure of a ciphertext.

- The first 8 bytes are used to store encrypted data:
- 9th byte is used to store sequence number which has values from range (0,255).
- The 9-byte message will be applied with base64 encoding to become a 12-byte message in order to assure the consistency of data transfer.

Figure 2.3: Payload structure

2.2 Encryption

Regarding to the Encryption process 2.4, a 64-bit initial key, and Authorization Level (AL) values are used as inputs for the first HMAC SHA256 to create the first key (K1). Then this key, along with the combination between ID and Sequence No will become inputs for generating the second key (K2) by using HMAC SHA256 again. The second key is used to do the One Time Pad encryption with real sensor data, plaintext, that gives us encrypted data (E).

$$K_1 = HMAC\_SHA256(K_0, AL)$$
$$K_2 = HMAC\_SHA256(K1, ID|SEQ)$$
$$E = OneTimePad(P, K_2)$$

To confront with limitation of transmission, a 8-byte ciphertext will be obtained by applying EAC. In particular, the length of plaintext is 8 bytes. Therefore, I get 8 first bytes from the second key K2 to do the One Time Pad encryption, and obtain

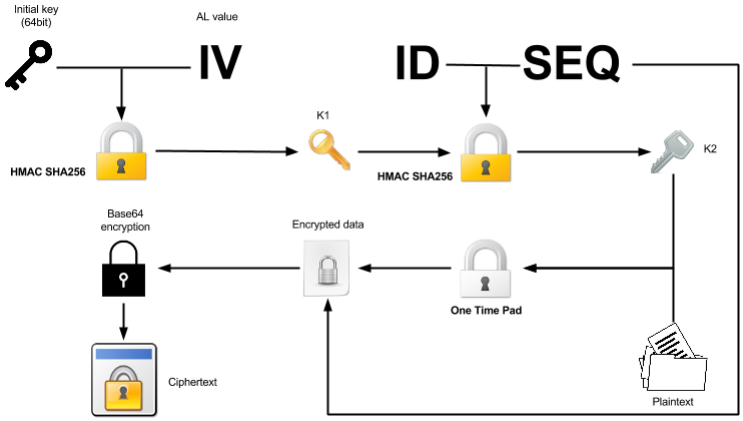

Figure 2.4: General encryption scheme

a 8-byte encrypted ciphertext. Next, one byte is saved to store Sequence Number value with a range of value from 0 to 254. Therefore, it's a 9-byte message. Finally, Base64 encoding is performed to get the final 12-byte ciphertext 2.3. Base64 encoding is used to prevent the loss of data in case of strange, unprintable, unreadable characters.

$$E^* = (E + SEQ)$$

$$C = Base64Encryption(E^*)$$

2.3 Decryption

From the decryption 2.5 perspective, ID, Seq No, and encrypted data are required to do the decryption. Firstly, base64 decoding of C is performed to get E^*. Then E, SEQ from E^*, are extracted from, and the key is regenerated by using HMAC SHA256. Eventually, One Time Pad encryption is performed once more to obtain the plaintext - real sensor value.

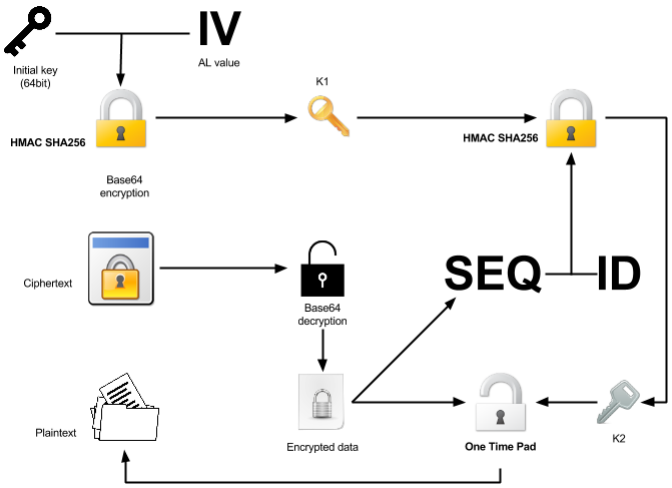

Figure 2.5: General decryption scheme

$$E^* = Base64Decryption(C)$$
$$(SEQ + E) = E^*$$
$$K_1 = HMAC\_SHA256(K_0, AL)$$
$$K_2 = HMAC\_SHA256(K1, ID|SEQ)$$
$$P = OneTimePad(E, K_2)$$

3 Evaluation

The encryption and decryption process mainly depend on HMAC_SHA256 function. In a general sense, the security provided by the HMAC-SHA-256 algorithms is based both upon the strength of the underlying hash algorithm, and upon the additional strength derived from the HMAC construct. However, as with any cryptographic algorithm, an important component of these algorithms' strength lies in the correctness of the algorithm implementation, the security of the key management mechanism, the strength of the associated secret key, and upon the correctness of the implementation in all of the participating systems. This specification contains test vectors to assist in verifying the correctness of the algorithm implementation, but these in no way verify the correctness (or security) of the surrounding security infrastructure.

4 Attack models

In this section, we establish an attack model for our cryptographic scheme. The figure 2.6 depicts the attack tree. We identify different assets: devices, back-end systems, and communications. We focus on three targets:

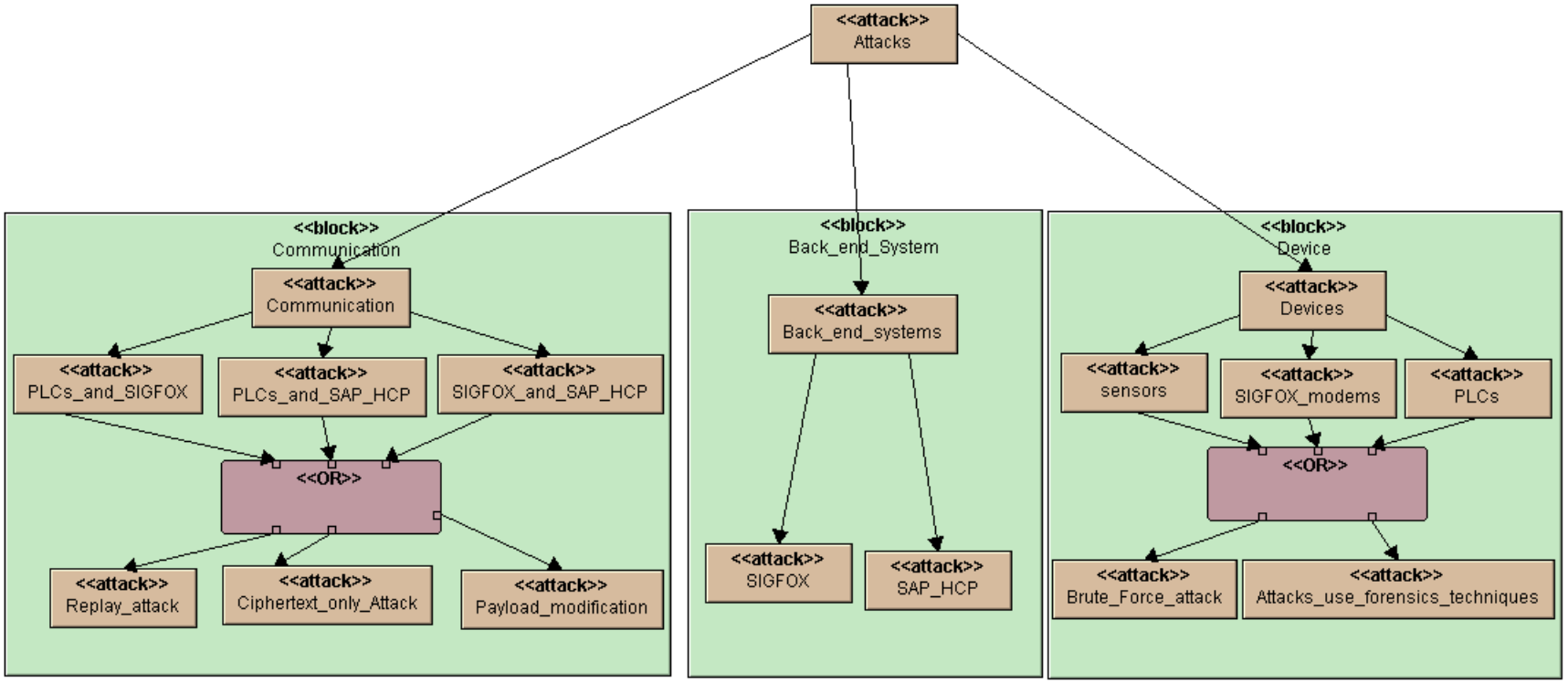

Figure 2.6: Attack tree

1. Communications between devices and systems
   - Between PLCs and SIGFOX.

- Between PLCs and SAP HANA Cloud Platform (SAP HCP).
- Between SIGFOX and SAP HCP.

2. Devices
   - Sensors
   - Programmable Logic Controllers (PLCs)
   - SIGFOX modems
3. Back-end systems
   - SIGFOX
   - SAP HCP

4.1 Communications

The attacker listens to communications between PLCs and SIGFOX, PLCs and SAP HCP, SIGFOX and SAP HCP. He captures messages in order to perform further attacks on the systems such as replay attack, or DDOS attack.

Replay attack				
Description 2.7	**Risks**	**Impacts**	**Solutions**	**Mitigations**
Continued on next page				

Description	Risks	Impacts	Solutions	Mitigations
The attacker analyzes captured messages to obtain destination. He keeps on resending those messages to the destination.	The attacker might send the same data to the system.	The predictive model computation is not accurate because of fake data. This has a serious effect on City of Antibes. They use predictive model to predict failure of pipelines for the water network. This leads to the wrong information in the water network such as wrong broken, leak pipelines.	SIGFOX communication technology: There is no negotiation between the device and a receiving station. The device simply emits in the available frequency band (+/- its own frequency shift). The signal is detected by the closest base stations, decoded, and forwarded to the network backend. Deduplication, and other protocol operations are handled by the network itself.	Keep track on sequence numbers from received messages to recognize its changes. The sequence number should increase by 1, and will reset to 0 if it is 256.

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
			Messages are then forwarded to your own application, and made accessible using SIGFOX's API. Each message is authenticated using a hash mechanism, and a private key specific to the device. This offers a great protection against replay attacks. The SIGFOX radio protocol also offers a great resistance to interferers.	Use session tokens. For example, if SIGFOX back-end wants to send data to SAP HCP, SAP HCP will send a token, which should be chosen by a random process (pseudorandom number), to SIGFOX, SIGFOX will calculate the hash value of message and token, and send it along with the message.

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
				Use timestamping. However this technique requires a secure protocol to synchronize clocks between systems. For example, SIGFOX back-end wants to send to a message to SAP HCP, it also sends its timestamp along with the message. SAP HCP only receives the message in a reasonable' limit.
	The attacker might perform DDOS attack with captured messages.	The DDOS attack might cause server overload, traffic congestion, and data loss		Set up the firewall to have some simple rules to deny all incoming traffic from the attackers based on the protocols, ports, or originating ID addresses

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
				Apply packet filtering on systems. Bogus information can be detected to some extent by filtering based on obvious inconsistencies in the packets. For example: A source address is not correct, or illegal sequence number and flags combinations, or from dark addresses

Table 2.1: Replay attack description table

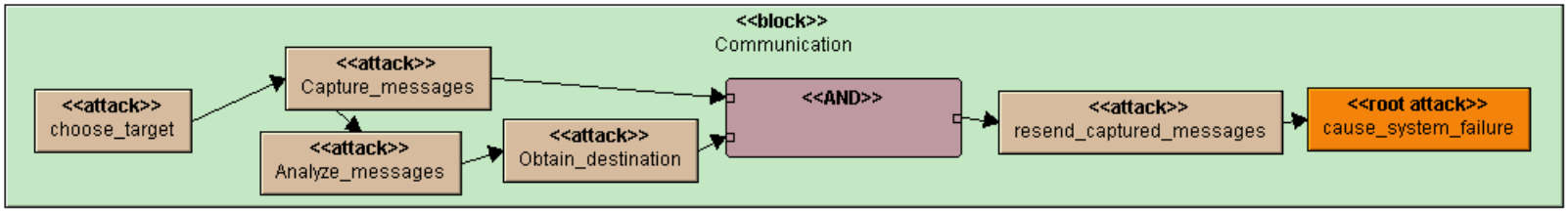

Figure 2.7: Replay attack model

Ciphertext-only Attack				
Description 2.8	**Risks**	**Impacts**	**Solutions**	**Mitigations**
The attacker uses captured messages as ciphertext to obtain cryptographic assets such as sequence number, and plaintext, and the root key.	He could use ciphertext as input of the brute-force attack 2.4 to obtain the root key.	Information disclosure because he obtains the root key which means that he is able to decrypt all captured ciphertext to obtain plaintext.		
	He might use ciphertext to perform Replay attack 2.1. He could resend those ciphertext to the system.	The predictive model computation is not accurate because of fake data. This has a serious effect on City of Antibes. They use predictive model to predict failure of pipelines for the water network. This leads to the wrong information in the water network such as wrong broken, leak pipelines.		

Table 2.2: Ciphertext-only attack description table

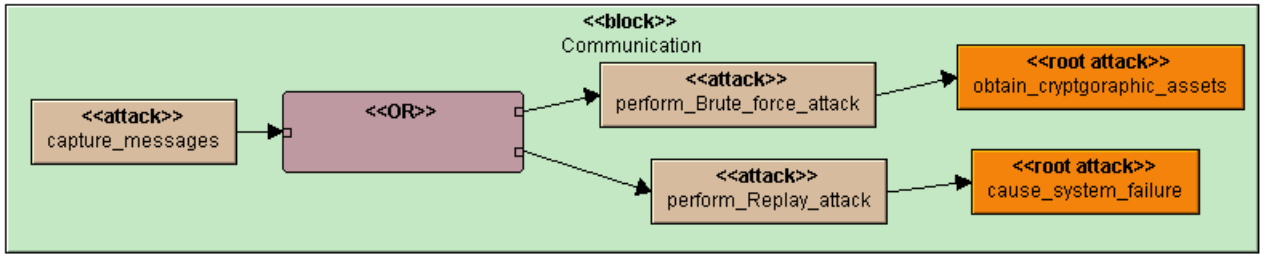

Figure 2.8: Ciphtertext-only attack model

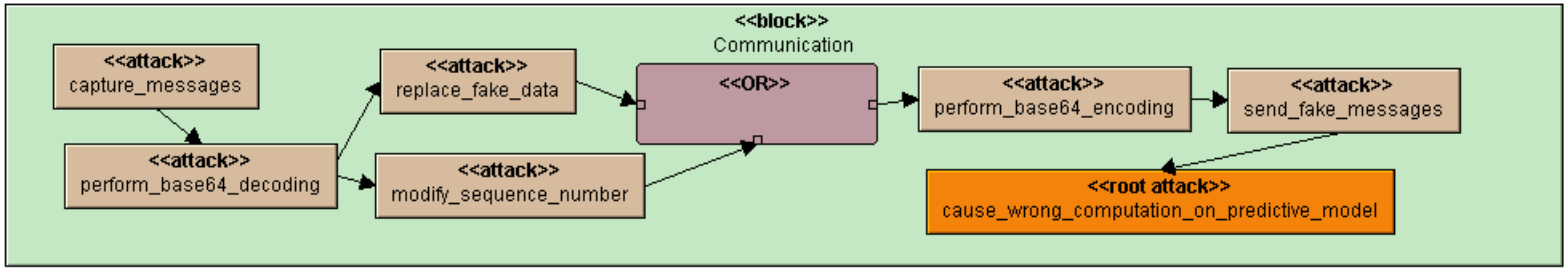

Figure 2.9: Payload modification model

Payload modification				
Description 2.9	**Risks**	**Impacts**	**Solutions**	**Mitigations**
Continued on next page				

Description	Risks	Impacts	Solutions	Mitigations
An attacker modifies payload of captured messages to create fake data. He does base64 decoding on the ciphertext. He modifies sequence number. He does not need to decrypt data, but simply replaces data with random value. He sends those messages to systems.	Fake data coming from the attacker might cause inaccurate computation of predictive model.	The predictive model computation is not accurate because of fake data. This has a serious effect on City of Antibes. They use predictive model to predict failure of pipelines for the water network. This leads to the wrong information in the water network such as wrong broken, leak pipelines.		Keep track on sequence numbers from received messages to recognize its changes. The sequence number should increase by 1, and will reset to 0 if it is 256.
				Keep track on plaintext. If the application cannot decrypt it properly. It could be faked data. The plaintext should be a positive number, not strange characters.

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
				Apply packet filtering on systems. Bogus information can be detected to some extent by filtering based on obvious inconsistencies in the packets. For example: The system will drop a packet with A source address is not correct, or illegal sequence number and flags combinations.

Table 2.3: Payload modification attack description table

4.2 Devices

When the attacker has physical access to devices. He replaces sensors, SIGFOX modems, and PLCs to send fake data. He also breaks sensors, SIGFOX modems, and PLCs to cause data loss. In addition, he injects malicious code, and sends fake data to systems. He uses computer forensics techniques to obtain cryptographic materials such as the root key, and OAuth token stored insecurely on PLCs.

Brute-force attack				
Description 2.10	**Risks**	**Impacts**	**Solutions**	**Mitigations**
The attacker tries to use HMAC SHA256 with a guess root key, and a guess AL value to create K1. He guesses ID of device, and obtains the sequence number from captured ciphertext. He uses K1 and guess ID and sequence number to create K2.	The attacker might access the root key	Information disclosure because he obtains the root key which means that he is able to decrypt all captured ciphertext to obtain plaintext.		A key management system to add, or revoke a key. The key is used by the attacker can be revoked.
Continued on next page				

Description	Risks	Impacts	Solutions	Mitigations
He uses One Time Pad to decrypt data with K2 to obtain plaintext.				Improve mechanism to choose sub keys for One-Time-Pad function. We choose a sub key (8bits) from key K2. The first position of a sub key depends on the sequence number. For example: first position = sequence number mod K2 length.
The plaintext should be meaningful such as a positive number, not strange characters. If it is not, the attacker does the process times. If he finds a root key which gives a good result, he will use this key to test on another ciphertext to verify the root key.				Add "salt" to the sequence number to make it difficult to predict. We choose a byte of a root key or key K1, then we could do XOR function to get a new sequence number before adding it to the payload.

Table 2.4: Brute-Force attack description table

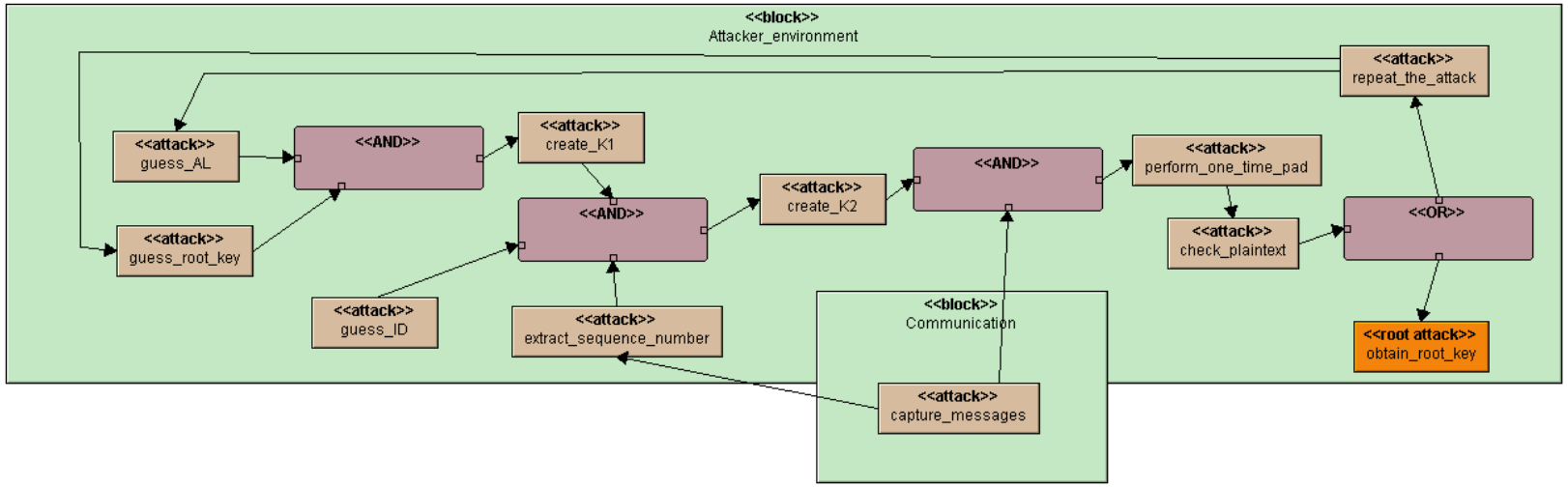

Figure 2.10: Brute-force attack model

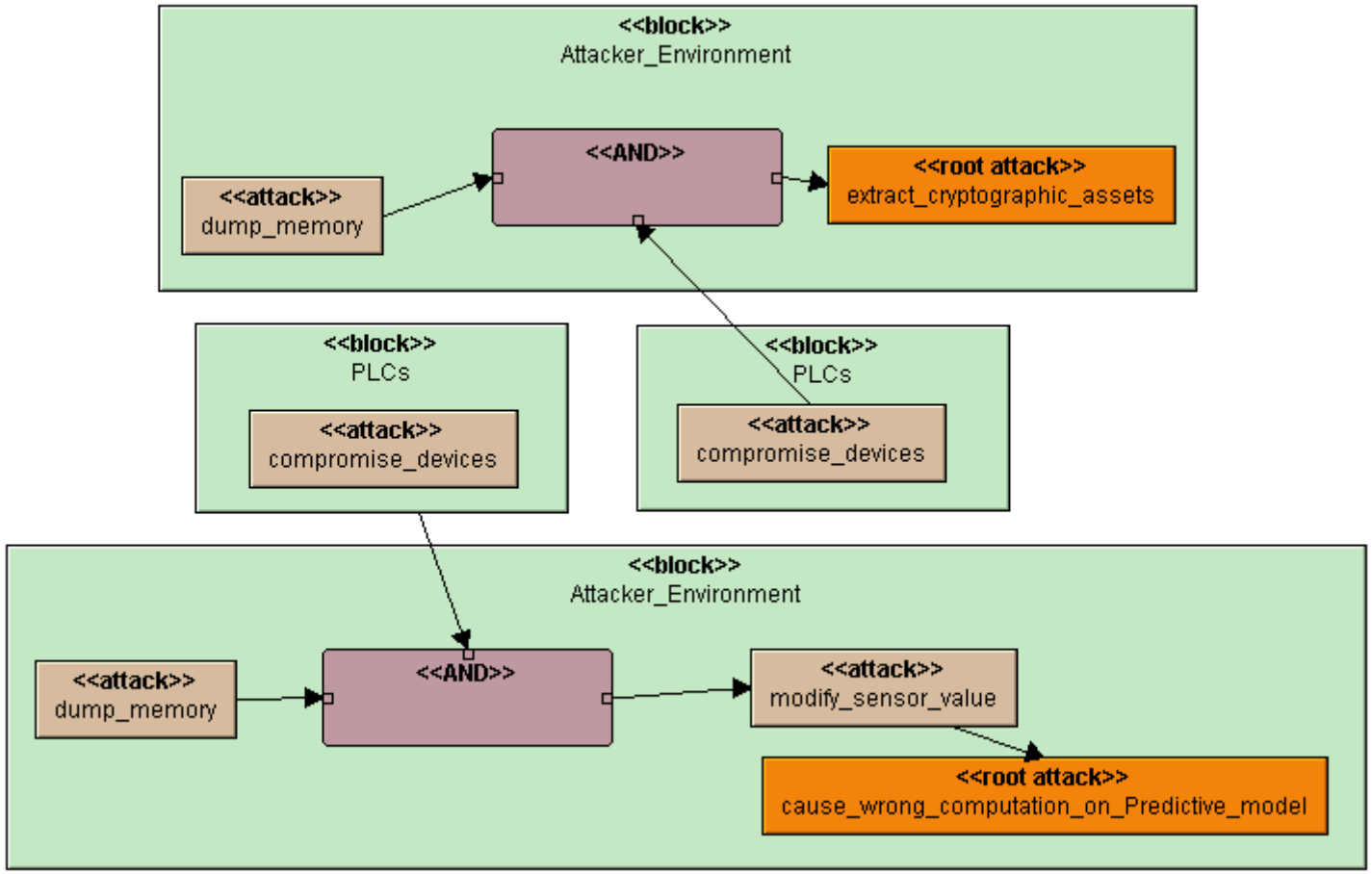

Figure 2.11: Attacks use forensics techniques model

Attacks use forensics techniques				
Description 2.11	**Risks**	**Impacts**	**Solutions**	**Mitigations**
The attacker uses forensics techniques to access memory of devices. He finds values of root key, OAuth token, sequence number, ID of devices here. In addition, he also dumps the memory to change sensor values into a fake values, and continues the encryption process to create legal ciphertext with fake data, send them to SAP HCP	He might have access to cryptographic assets such as root key, OAuth token, ID, and sequence number because they are not stored secretly on devices.	Information disclosure because he obtains the root key which means that he is able to decrypt all ciphertext to obtain plaintext.		Physical security improvements: Devices are human inaccessible by cases, locks. GPS on a device to manage its location change. If the location is changed, we could know that probably the device is compromised so that we inactivate it. Secure storage to store cryptographic assets.

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
	He could create legal ciphertext with fake data, and send them to SAP HCP.	The predictive model computation is not accurate because of fake data. This has a serious effect on City of Antibes. They use predictive model to predict failure of pipelines for the water network. This leads to the wrong information in the water network such as wrong broken, leak pipelines.		2.3

Table 2.5: Attacks use forensics techniques description table

4.3 Back-end systems

Attacks on SIGFOX backend				
Description 2.12	**Risks**	**Impacts**	**Solutions**	**Mitigations**
The attacker hacks SIGFOX backend to have the access. He logs in SIGFOX back-end to modify system's behaviors. He modifies SIGFOX backend features such as callbacks, device and device type managers to cause bugs in the system, or stops communications between SIGFOX and PLCs, between SIGFOX and SAP HCP.	The attacker might send fake data to the SAP HCP by modifying callbacks setting	The predictive model computation is not accurate because of fake data. This has a serious effect on City of Antibes. They use predictive model to predict failure of pipelines for the water network. This leads to the wrong information in the water network such as wrong broken, leak pipelines.		RSA token for log-in system. This is an application can be integrated on your laptop or mobile. Every time a user needs to log in to the back-end system. He must have a passcode which can be randomly generated by the app. The app also requires a pin code to give the user a correct token as passcode. The passcode is valid for a limited time.

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
	The attacker could illegally registers fake devices. He use them to send fake data to SAP HCP	The predictive model computation is not accurate because of fake data. This has a serious effect on City of Antibes. They use predictive model to predict failure of pipelines for the water network. This leads to the wrong information in the water network such as wrong broken, leak pipelines.		Support the principle of "Segregation of Duties" for critical combinations of authorizations. Following different usages should be supported at least: configuration, device registration, and sending messages.
	The attacker could observe encrypted data by message feature from SIGFOX. He uses those messages to perform ciphertext only Attack	The system is put in the higher level for serious attacks such as Replay attack, and DDOS attack		Support the principle of "Segregation of Duties" for critical combinations of authorizations. Following different usages should be supported at least: Configuration, device registration, and sending messages.

Table 2.6: Attacks on SIGFOX backend description table

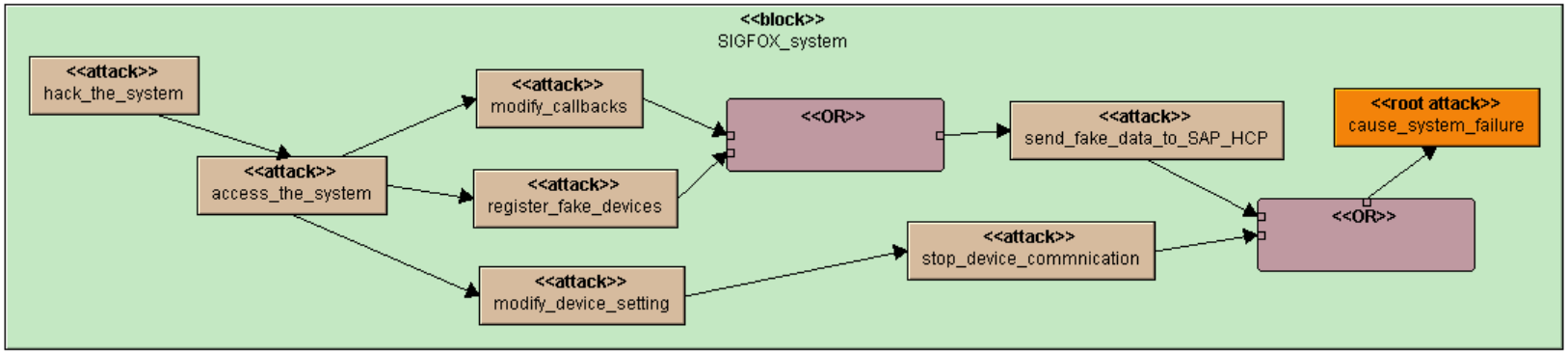

Figure 2.12: Attacks on SIGFOX backend model

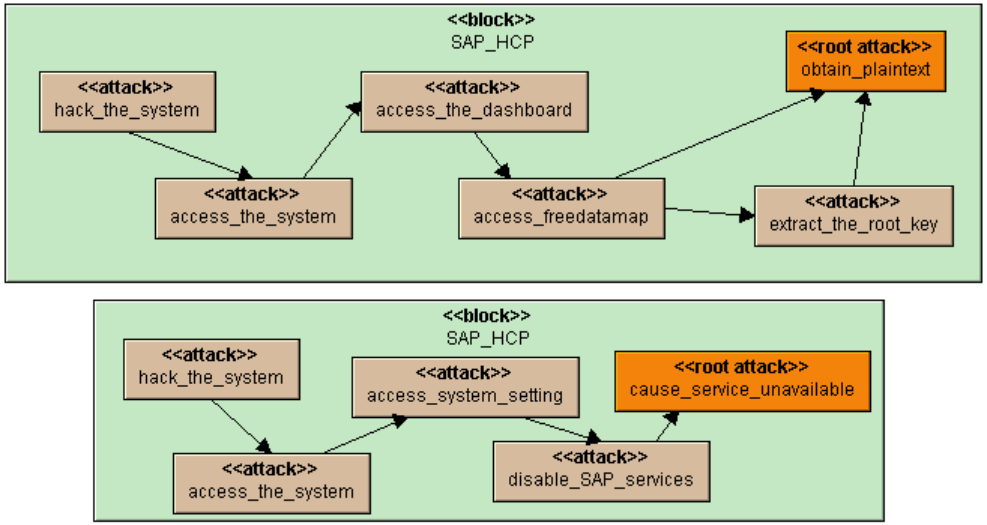

Figure 2.13: Attacks on SAP HCP backend model

Attacks on SAP HANA Cloud Platform				
Description 2.13	**Risks**	**Impacts**	**Solutions**	**Mitigations**
The attacker hacks SAP HCP to have the access. He logs in SAP HCP. On SAP HANA Cloud Platform, the attack has access to every service running there such as the dashboard, HANA DB, IoT services. He could also do serious actions. He modifies data in HANA DB, or disables IoT services.	The attacker might access integrated applications on SAP HCP and obtain plaintext.	Information disclosure because he obtains the root key which means that he is able to decrypt all ciphertext to obtain plaintext.		
	The attacker might access HANA Database. He could empty the DB, or inject fake data, or obtain sensitive data such as device coordinates, ID of devices.	The system does not work properly because of bug appearance. The application cannot decrypt fake data for example. This affects directly to the predictive model.	Restrict user permissions. Only allow a few users to access to HANA DB	Data back-up, and logging systems to recover data loss.

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
	The attacker might disable all services from SAP HANA Cloud Platform Cockpit such as IoT services, HANA DB, HANA XS, Java, and HTML applications.	SAP services are not available.		RSA token for log-in system. This is an application can be integrated on your laptop or mobile. Every time a user needs to log in to the back-end system. He has to have a passcode which can be randomly generated by the app. The app also requires a pin code to give the user the correct token as passcode.
	He could also change the OAuth token from Internet of Things Services Cockpit	Data loss because SAP IoT Service cannot receive any message from SIGFOX, and devices.		Support the principle of "Segregation of Duties" for critical combinations of authorizations. Following different usages should be supported at least: Pure consumption of data, Configuration, and key management, Device registration, Sending messages

Continued on next page

Description	Risks	Impacts	Solutions	Mitigations
	The attacker might obtain cryptographic assets such as the root key, ID of devices from applications running from the dashboard			Secure storage on SAP HCP to store cryptographic assets.

Table 2.7: Attacks on SAP HANA Cloud Platform description table

We identified both security requirements from French Public Sector - City of Antibes, and limitations from the network carrier. Therefore, we decided to apply the cryptographic scheme EAC with a modification to overcame the difficulty. In addition, we also have addressed the confidentiality and authenticity of messages.

INTERNET OF THINGS

1 Internet of Things Devices

In this section, we would like to go through all components of Internet of Things in the general architecture 1.2 to understand clearly their roles as well as how they are used, and deployed in detail.

- Arduino collects data from sensors first of all. Next, it encrypts the data that is collected and stored as files including id of devices, sensor values, and locations from the City of Antibes, and sends encrypted data to SIGFOX. Finally, SIGFOX sends the data to SAP HCP.
- Arduino Yun collects data from sensors. It encrypts the data, and sends directly to SAP HCP via WIFI
- Raspberry Pi encrypts backup data. It sends encrypted data , and geolocation data to SAP HCP via WIFI.

1.1 Arduino

Arduino [19] is a family of single-board microcontrollers. Open-source, the platform is based on a simple microcontroller board, and includes a development environment for writing software for the board. Arduino uses C and C++ programming languages, and offers a number of different boards with varying features and functionalities built into them.

1.1.1 Related Work

First off all, Arduino Uno 3.14 gathers water consumptions, or temperature data from sensors. The data then is encrypted follow the cryptographic scheme. In the

next phase, the data will be pushed directly to SIGFOX servers. Finally SIGFOX back-end sends the encrypted data to the final destination - SAP HCP IoT service.

The advantage of this data direction is the internet connection is not needed to transfer data from devices to SAP HCP comparing to Raspberry Pi or Intel Edison devices.

Figure 3.14: Arduino use in sending messages to SAP HCP

1.1.2 Configuration

(a) **Hardware**

– Two boards of **Arduino Uno** 3.15 are implemented. On top of those devices are **Akene board** [20]. One connects to a debit meter, another connects to a temperature sensor

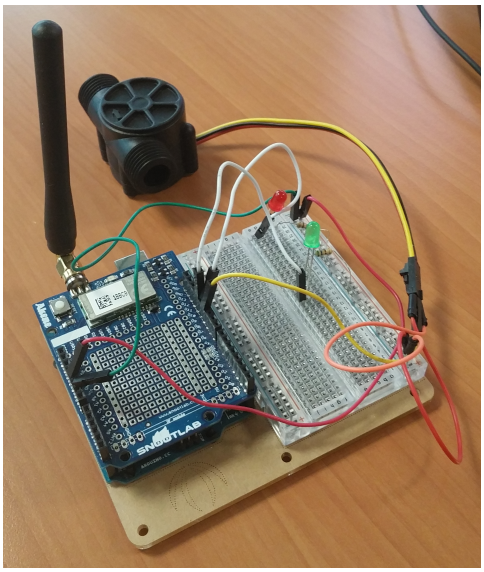

Figure 3.15: Arduino Uno with sensors, and Akene shield

– Arduino Yun [11] is also implemented the cryptographic scheme. It sends encrypted data to SAP HCP directly via WIFI. Therefore, SIGFOX is not used in this data direction. However, the internet connection is mandatory.

(b) **Software**

To setup the working environment, Arduino Software (IDE) is used, and installed on both Raspberry Pi, and Windows. This particular software is a platform which can be used to program Arduino in C/C++ language. It later uploads to the Arduino.

(i) Arduino Software Installation

- Install Arduino Software (IDE) [21]
  - *Windows:* Download this file, and install it.
- *Raspberry Pi:* Open a terminal and type

```
sudo apt-get install arduino
```

- Choose **Board**, and **Port** 3.16 settings correctly.

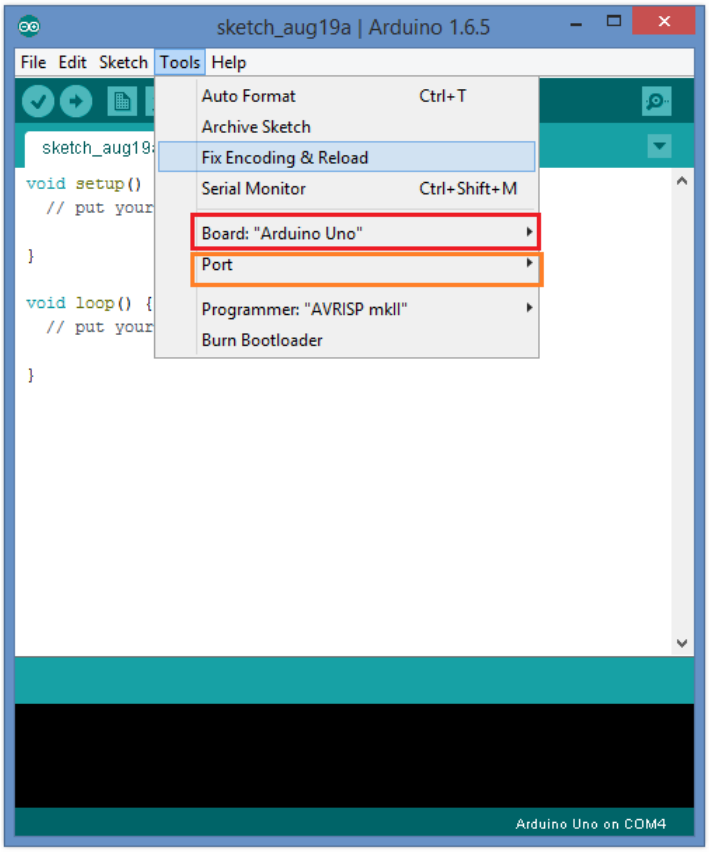

Figure 3.16: Board, and Port selections

- Copy all libraries to

* *Windows:*

```
<Your_Arduino_Installation_folder>/libraries/

   example:
   C:/Users/<yourname>/Documents/Arduino/libraries/

```

* *Linux:*

```
<Your_Arduino_Installation_folder>/libraries/

   example:
   /usr/share/arduino/libraries/
```

– Finally, restart Arduino IDE, and include it in the code with include statement:

```
#include <Akene.h>
#include "sha256.h"
#include "base64.h"

```

To setup the connection with Arduino on Windows for getting data. Pycharm Community Edition version 4.5.3 [31] is used. Install Pycharm, and follow the code for connection with Arduino Uno board on Windows:

```
serialport = "\.\COM4"
```

In addition, *Visual Micro*[32] is used as a plugin in *Visual Studio Community 2013*[33] to debug the code, and an extra library named MemoryFree[34] is used to check free memory in Arduino if the test is replicated. Furthermore, Pycharm is used to capture messages from Arduino in order to debug the code, and to assure messages before encrypted messages are sent via the wire to SIGFOX.

1.1.3 Code Usage

The code is implemented in C/C++ language as follows:

(a) Get the value from sensors
(b) Encrypt the value
(c) Send encrypted value to SIGFOX servers

In addition, all source code can be found in **Appendices**.

- Sending data from *Arduino* to *Raspberry Pi*
  (i) File Arduino_RasPi_HCP_Temperature.ino is used to send *temperature* data.
  (ii) File Arduino_RasPi_HCP_DebitMeter.ino is used to send directly *debit meter* data.
- Sending data from *Arduino* to *SIGFOX*
  (i) File Arduino_SIGFOX_Temperature.ino is used to send *temperature* data.
  (ii) File Arduino_SIGFOX_Debit.ino is used to send *debit meter* data.

1.1.4 Deployment

These code snippets on Arduino are for two different sensors. One is debit meter sensor with measure of water flow, and another is temperature sensor.

- Open Arduino IDE, and click **File**, then **Open** 3.17.

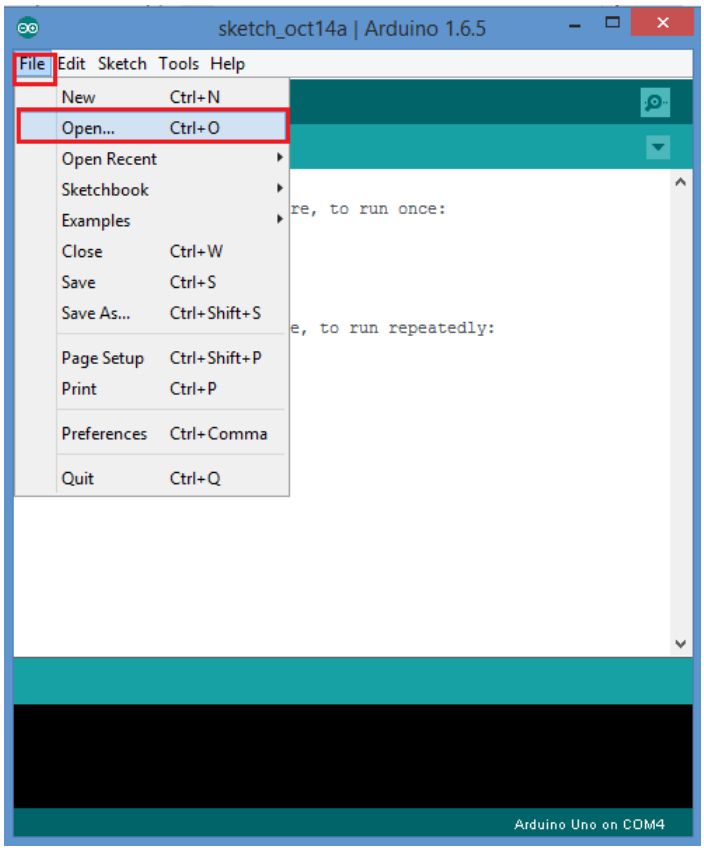

Figure 3.17: File opening

– Browse to the file to open it.
– Hit **Upload** button to verify, and upload the code to Arduino 3.18.

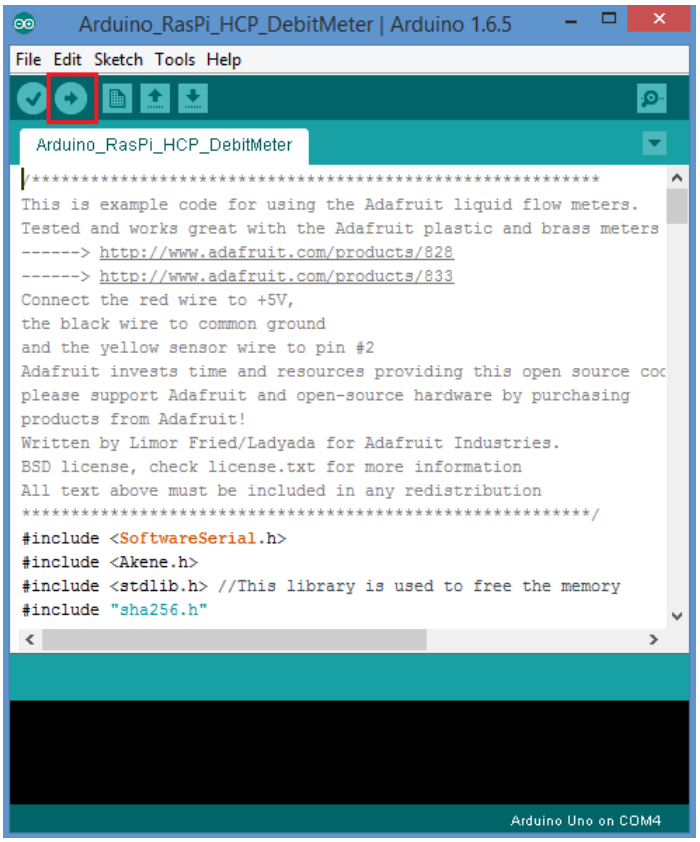

Figure 3.18: Code uploading

– Wait for a second to finish the uploading 3.19.

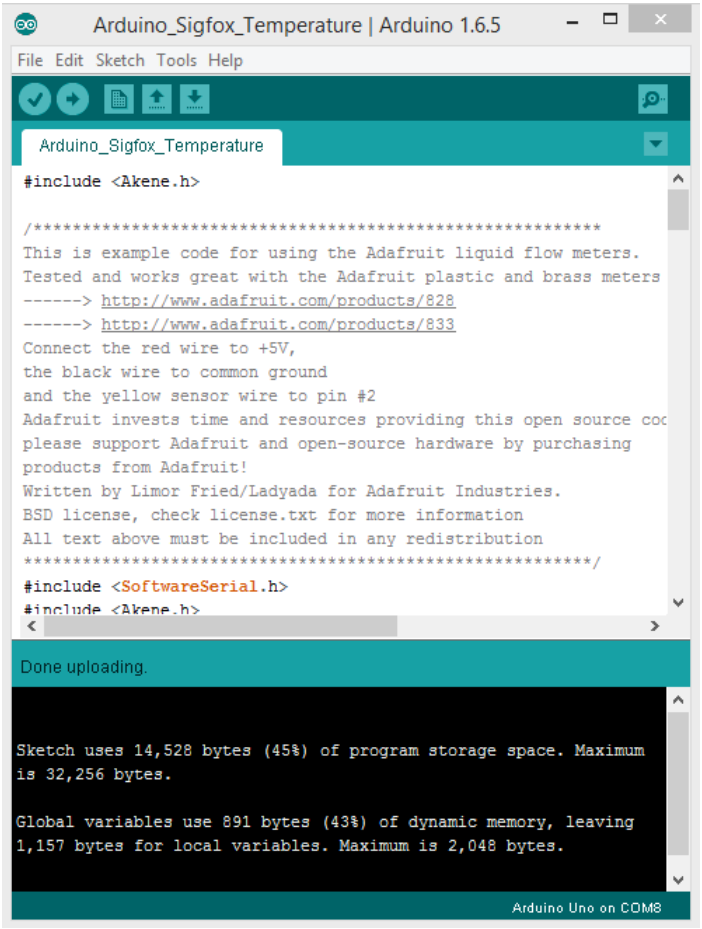

Figure 3.19: Compling result

– Check it by clicking **Tools**, then choose **Serial Monitor ??**.

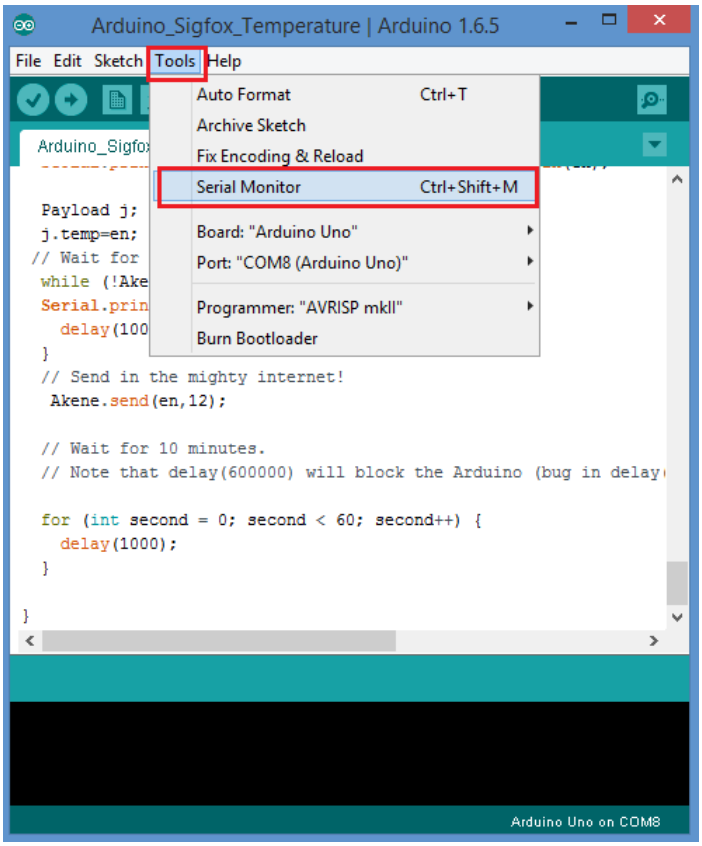

Figure 3.20: Serial Monitor opening

– The pop-up shows data coming to SIGFOX every 10 minutes according to the code 3.21.

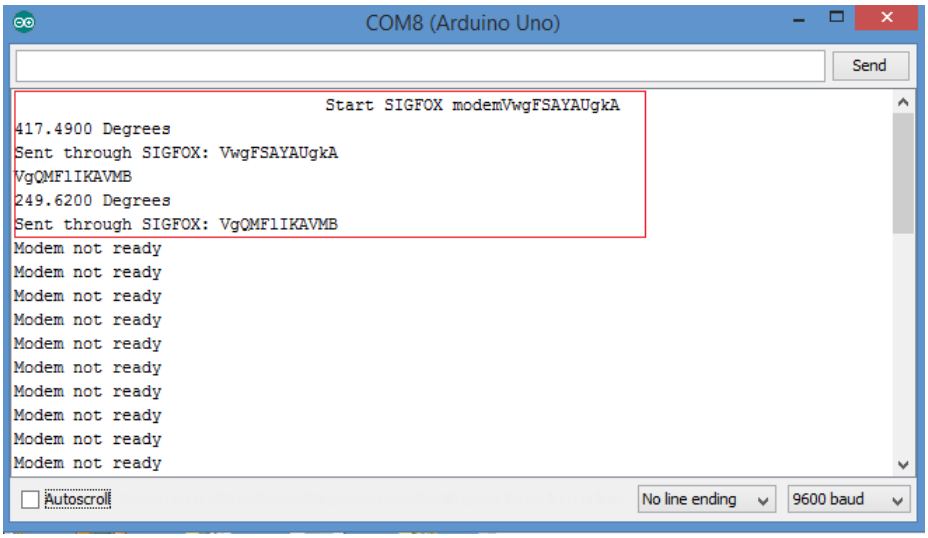

Figure 3.21: Message observation from Serial Monitor

1.2 Raspberry Pi

Raspberry Pi [5] is a single-board computer with a micro processor, and can run an OS. It can be used for electronics projects with its connectors, as well as operate as a standard desktop computer running on Linux. Modular, Pi makes it easy to write drivers for automating processes on the small electronic projects, and mainly uses the Python programming language.

1.2.1 Related Work

Raspberry Pi 3.22 is used to push data to HCP IoT Service. In specific, it will :

- Directly collect encrypted data from Arduino, and push it to SAP HCP IoT.
- Take offline data, encrypt it, and send it to SAP HCP IoT.
- Extract, and send geolocation data to SAP HCP IoT.

(c) Receive data follow the callbacks coming from SIGFOX back-end, and forward it to SAP HCP IoT. Furthermore, Raspberry Pi is also value in data collection from Arduino to analyze, and verify the correctness of encrypted messages before the code is applied on Arduino.

Figure 3.22: Raspberry Pi

1.2.2 Configuration

1. Download, and extract urllib3 on a Raspberry Pi.
2. Install **Python 2.7**, **pip** and by the following code:

```
sudo apt-get update
sudo apt-get install python2.7
sudo apt-get install python2.7-dev

sudo apt-get install python-pip
```

3. Install *urllib3*[22]
   (a) Open a terminal
   (b) *CD* to urllib3 extracted folder
   (c) Type

```
pip install urllib3
```

4. Install *Arduino Studio IDE* on Raspberry Pi.

```
sudo apt-get install arduino
```

Create and copy library files to the folder:

```
/usr/share/arduino/libraries/
```

5. Install *certifi* [23]

```
pip install certifi
```

To deploy Arduino codes, make sure that all necessary libraries are installed correctly. All libraries are stored in the libraries folder.

- Windows : Copy them to $C : /Users/i321696/Documents/Arduino$
- Linux : Copy them to $/usr/share/arduino/libraries/$

1.2.3 Code Usage

The code is implemented in python language as follows:

- To receive data from SIGFOX, a bash script is run to listen to a paricular port with netcat software. The message comes from SIGFOX is pharsed, and pushed to SAP HCP
  1. Use files **script**, and **pushtoHANA.py**
  2. Open a **terminal**
  3. Type this piece of code

```
chmod 700 script
```

  4. Continually input this code:

```
while : ; do ./script; done
```

5. Do not close the terminal

– Raspberry Pi receives encrypted data from Arduino. It pushes the data directly to SAP HCP.
  1. Use file Arduino_RasPi_HCP.py
  2. Open a *terminal*
  3. Type this piece of code:

```
python Arduino_RasPi_HCP.py
```

– Raspberry Pi pushes **offline data** to SAP HANA Cloud Platform IoT every 10 seconds.
  1. Use file LocalToHCP_FULL.py, and offline data
  2. Open a *terminal*
  3. Type

```
python LocalToHCP_FULL.py <Folder_offline_data>

#Example: python LocalToHCP_FULL.py offlinedata\201503\
```

– Raspberry Pi pushes **coordinate data** to SAP HANA Cloud Platform IoT
  1. Use files ExtractCoordinates.py, and COMPTEUR.geojson
  2. Open a **terminal**
  3. Type

```
 python ExtractCoordinates.py <geolocation_file>

#Example: python ExtractCoordinates.py COMPTEUR.geojson
```

– if it works properly, the data comming from **Nessage Management Service Cockpit**
  1. Click to **Display stored messages** in **Message Management Service Cockpit** 3.23

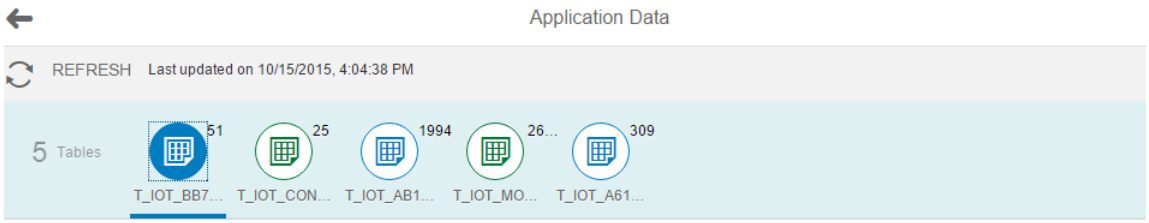

Table [T_IOT_BB7F2789479BF02C2799] contains [51] entries.

G_ID	G_DEVICE	G_CREATED	C_ID	C_COORDINATE
51	809fd3e6-1926-4981-a8ad-281f53a82792	2015-09-18 10:11:29.212	R_GAROUPE	43.563903, 7.132188
50	809fd3e6-1926-4981-a8ad-281f53a82792	2015-09-18 10:11:03.206	R_AMES	43.589926, 7.09654
49	809fd3e6-1926-4981-a8ad-281f53a82792	2015-09-18 10:10:27.199	R_SEMBOULES	43.601957, 7.068402
48	809fd3e6-1926-4981-a8ad-281f53a82792	2015-09-18 09:54:00.02	A52	43.608586, 7.111124
47	809fd3e6-1926-4981-a8ad-281f53a82792	2015-09-18 09:54:00.02	A47	43.608364, 7.110878
46	809fd3e6-1926-4981-a8ad-281f53a82792	2015-09-18 09:54:00.02	A54	43.608081, 7.111207

Figure 3.23: Message observation

1.3 Intel Edison

A promising device, which is as known as a mix between a PLC, and a single-board computer, could give a better performance as well as extra features. It supports many platforms. Developer have more opportunities to develop their programs without limitations

1.3.1 Related work

Intel Edison [24] is used along with Akene shield from SIGFOX to push encrypted data to SIGFOX backend. The system runs Yocto Linux [25]. Therefore the work in this section is divided into two parts:

- Set up Intel Edison 3.24 device
- Encrypt and push messages to SIGFOX back-end in Python language.

In addition, the device can use codes from Raspberry Pi as long as it is installed all suitable libraries.

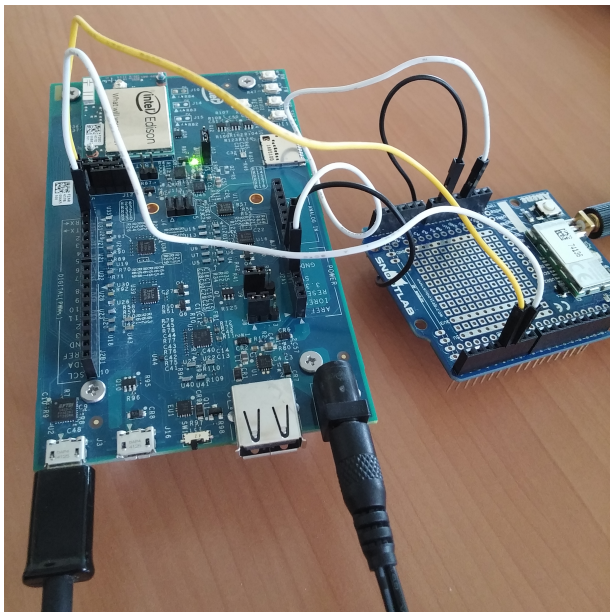

Figure 3.24: Intel Edison

1.3.2 Configuration

1. Intel Edison first set-up: Basic guide - Video
   - Install manually components together
   - Connect the device to your computer, and install drivers src - Windows, Linux
   - First flash Yocto Linux 2.1 as in the video above, or it can be done easily by **Flash Tool Lite** - Windows, Linux - guide
   - Use putty to login to the device 3.25.
     * Choose **Serial** for **Connection type**.
     * Type port number which can be checked in the **Device Manager**, and **115200** in **Speed**.
     * Hit **Open** finally to open the connection.

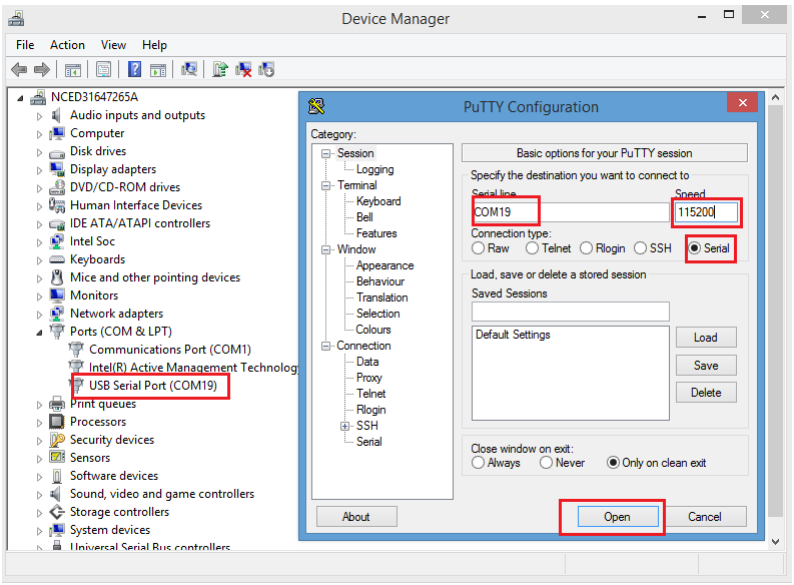

Figure 3.25: Putty setting

- Set new **password**, and set up Wifi connection. Therefore, the device can be accessed by both SSH or putty
- Install libraties to execute the code - src
- Install **pip**

```
opkg install python-pip
```

- Install **pyserial** [26]

```
pip install pyserial
```

2. Intel Edison with Akene shield - SIGFOX modem
   - Simply put the Akene shield above the device Intel Edison
   - Test the connection by the blink test. Both leds from Edison and Akene are going to blink.

```
#!/usr/bin/env python

import mraa
import time
```

```
import random

led = mraa.Gpio(13)
led.dir(mraa.DIR_OUT)

while True:
    i = random.uniform(0.1,6.5)
    led.write(1)
    time.sleep(i)
    led.write(0)
  time.sleep(i)
```

3. Sending the first message to SIGFOX
   - Reorganize jumps to connect Intel Edison and Akene shield

```
Ground to Ground
3.3v to 3.3v

serial Rx (pin 0) of Edison to Tx of Akene (pin D4)
serial Tx (pin 1) of Edison to Rx of Akene (pin D5)
```

   - Connect Intel Edison with computer with the lowest USB port for serial connection
   - Check connection between Intel Edison and Akene modem
     * From putty type

```
    python -m serial.tools.miniterm
```

     * Type **/dev/ttyMFD1** for **Port name**
     * Use **AT** which should reply OK (otherwise there is a problem), and **AT&V** which reply by the TD1208 identification.
     * Send the first message with **AT$SS=message** 3.26

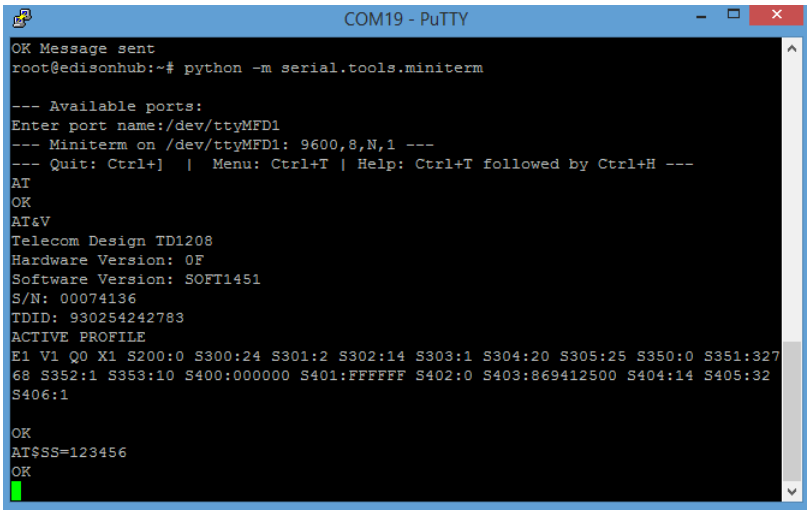

Figure 3.26: Sending first message manually

* Message can be sent via this python code 3.27 **python SIGFOX.py message** with message is a message in hex format, and not longer than 24 characters. The piece of code for **SIGFOX.py** as follows:

```
1     #!/usr/bin/python
  import time
3 import serial
  import sys
5 from time import sleep

7 #LINE ADDED
  import mraa
9
  SOH = chr(0x01)
11 STX = chr(0x02)
  EOT = chr(0x04)
13 ACK = chr(0x06)
  NAK = chr(0x15)
15 CAN = chr(0x18)
  CRC = chr(0x43)
17
  def getc(size, timeout=1):
19     return ser.read(size)

21 def putc(data, timeout=1):
```

```
    ser.write(data)
23  sleep(0.001) # give device time to prepare new buffer
    and start sending it

25 def WaitFor(ser, s, timeOut):
        nbMax = 0
27      ser.timeout = timeOut
        currentMsg = ''
29      while currentMsg.endswith(s) != True :
            # should add a try catch here
31          c=ser.read()
            if c != '' :
33              currentMsg += c
            else :
35              print 'timeout waiting for ' + s
                return False
37          nbMax = nbMax + 1
            if nbMax > 150:
39  print 'Timeout expired'
    return False
41      return True


43
print('Sending SIGFOX Message...')
45
#LINE ADDED: 0 ie '/dev/ttyMFD1'
47 uart = mraa.Uart(0)

49 modem = serial.Serial(
       uart.getDevicePath(),
51     baudrate=9600,
           parity=serial.PARITY_NONE,
53         stopbits=serial.STOPBITS_ONE,
           bytesize=serial.EIGHTBITS
55  )

57 #LINE ADDED: closing serial before opening it, as otherwise
    the Edison serial seems to be already open by calling
    serial.Serial
modem.close()
```

```
59
modem.open()
61
modem.write('AT\r')
63
if (WaitFor(modem, 'OK', 3)):
65    print('SIGFOX Modem OK')
else:
67    print('SIGFOX Modem Error')
      modem.close()
69    exit()

71 modem.write("AT$SS={0}\r".format(sys.argv[1]))

73 print('Sending ...')
if (WaitFor(modem, 'OK', 15)):
75    print('OK Message sent')
else:
77    print('Error Sending message')
      modem.close()
79    exit()

81 modem.close()


83
```

```
root@edisonhub:~# python sendtosigfox.py 123456
Sending SigFox Message...
SigFox Modem OK
Sending ...
OK Message sent
root@edisonhub:~#
```

Figure 3.27: Sending first message with python

- More information can be found from this project [27]

4. Cryptographic scheme
   - The code can be found in **Appendices**

- From this piece of code, the **value** is generated randomly from 0 to 99999999. It can be replaced by the real value collected from sensors.
- There functions encryption, decryption, and sendtoSIGFOX are distinct.
- An encrypted message is sent to SIGFOX servers every 10 minutes

1.3.3 Code usage

The code is implemented in Python language as follows:

- File *blink.py* is used to do blink the LED on device.
- File *SIGFOX.py* is used to send data to SIGFOX. The data should have maximum hexadecimal digits is 24 (12 bytes).
- File *Send_encrypted_data_to_SIGFOX* is used to encrypt data, and send it to SIGFOX.

2 SIGFOX - IoT Network Carrier

SIGFOX [6] is a company providing global cellular connectivity for the Internet of Things, fully dedicated to low-throughput communications. SIGFOX is re-inventing connectivity by radically lowering prices and energy consumption for connected devices.

2.1 Related Work

SIGFOX is used as a IoT device Network Provider. Encrypted data will be sent to SIGFOX's servers. Callback function is set up to send back the data to SAP HCP IoT.

In this section, SIGFOX callbacks is set up to transfer collected data from devices to SAP HCP.

2.2 Callbacks Configuration

- Register devices first.
- Sign in to the SIGFOX Back-end
- Set up a callback function for every device type 3.28.
- Click on **Device Types**. Next click on **Callback** on the left menu, and **New** 3.29

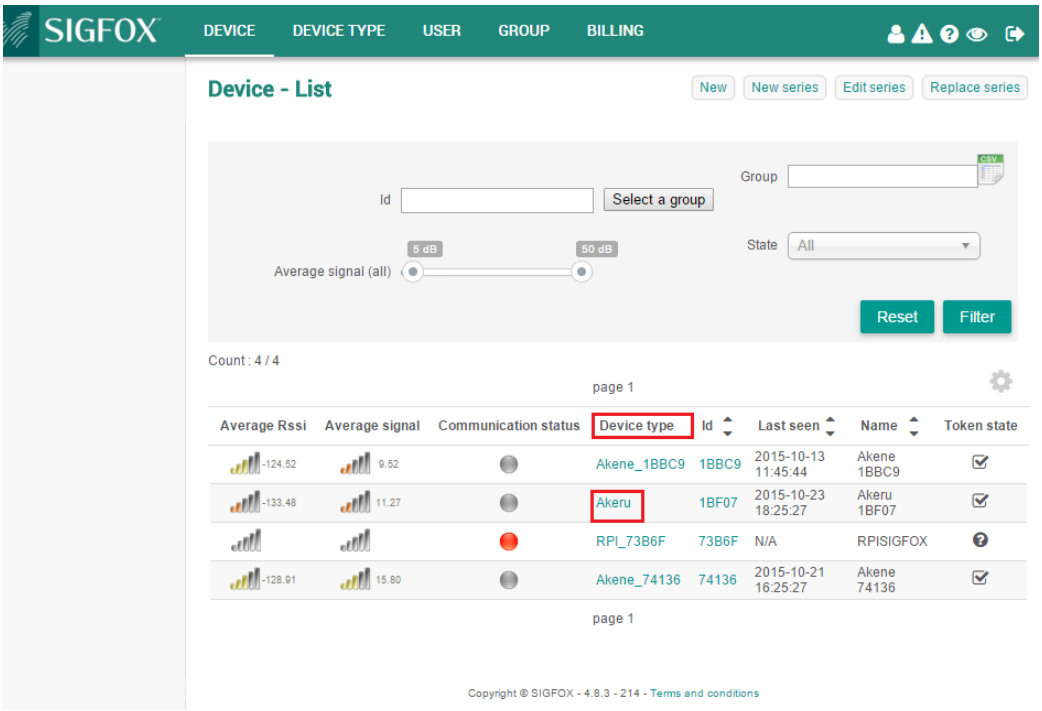

Figure 3.28: SIGFOX - Callback function guild

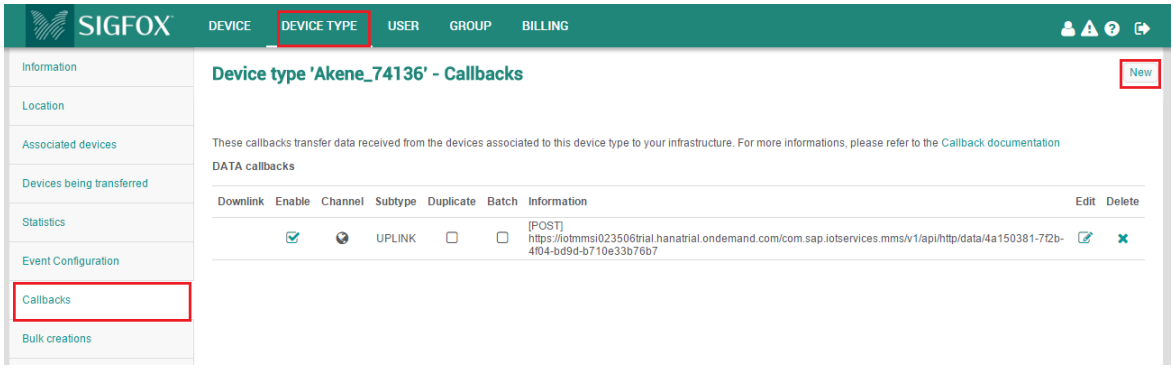

Figure 3.29: SIGFOX - Callback function guild 2

– A pop-up will be shown, choose **DATA** and **UPLINK** in Type, **URL** for channel 3.30. Fill the Url partern and click **OK**.

```
http://<ipaddress>:<portnumber>/id={device}&data={data}&time={time
    }&duplicate={duplicate}
```

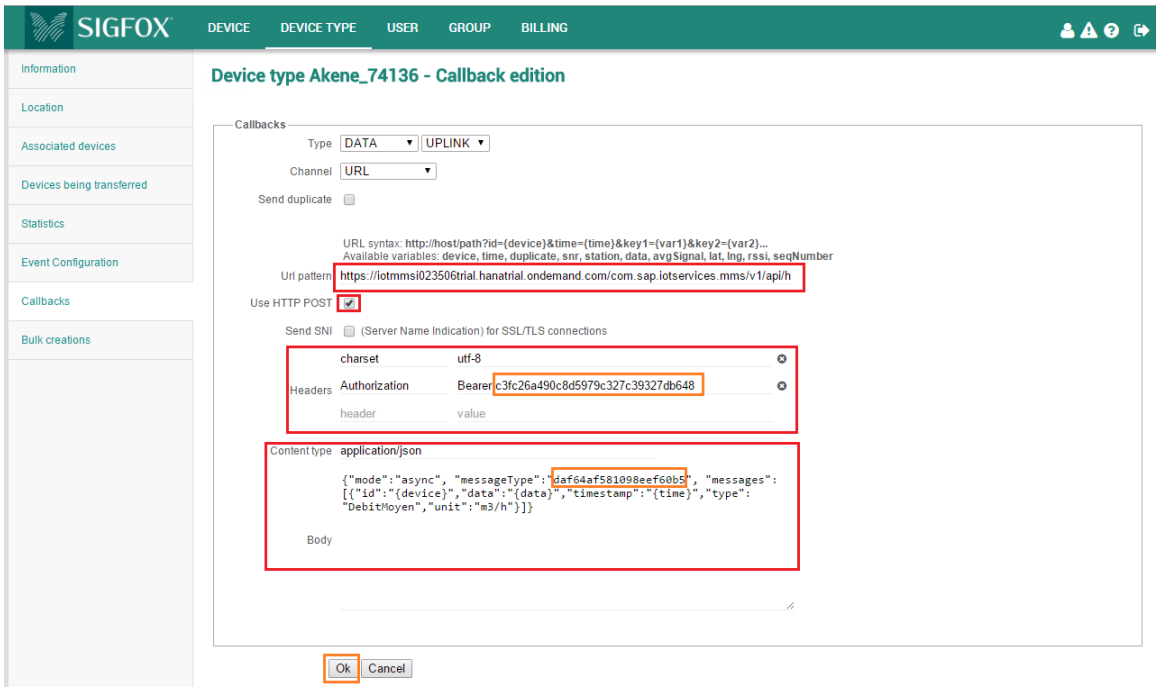

Figure 3.30: SIGFOX - Callback function guild 3

– Configure the Url pattern

```
https://iotmms<your_account>trial.hanatrial.ondemand.com/com.sap.
    iotservices.mms/v1/api/http/data/<your_device_id>

    For example:
    https://iotmmsd046598trial.hanatrial.ondemand.com/com.sap.
    iotservices.mms/v1/api/http/data/4be16905-0244-4bd0-8299-3
    bec61814458
```

– Check Use HTTP POST checkbox.

– Add Authorization and Bearer Your_Token in Headers. Adapt Your_Token

```
Bearer <Your_Token>

For example:

Bearer 78a1725b741f12fbadcd8bc7fa222a5
```

– In the **Body**, use this piece of code. However, adapt **message**

```
{"mode":"async", "messageType":"374383ef66c7d32afa8b", "messages"
    :[{"id":"{device}","data":"{data}","timestamp":"{time}","type":
     "Art","unit":"Einheit"}]}
```

– Adapt your_message_type (It's the data table id registered by SAP IoT Service)

```
"messageType":"<your_message_type>"

For example:
"messageType":"374383ef66c7d32afa8b"
```

– Modify **type** and **unit** which are suitable with your device.

```
"type": "Art","unit":"Einheit"

or can be "type": "Temperature","unit":"C"
```

– Hit **OK** to finish the setting

– The following values will be sent to Raspberry Pi : id, data, and timestamp.

– Go back to the Device list, choose **ID** 3.31

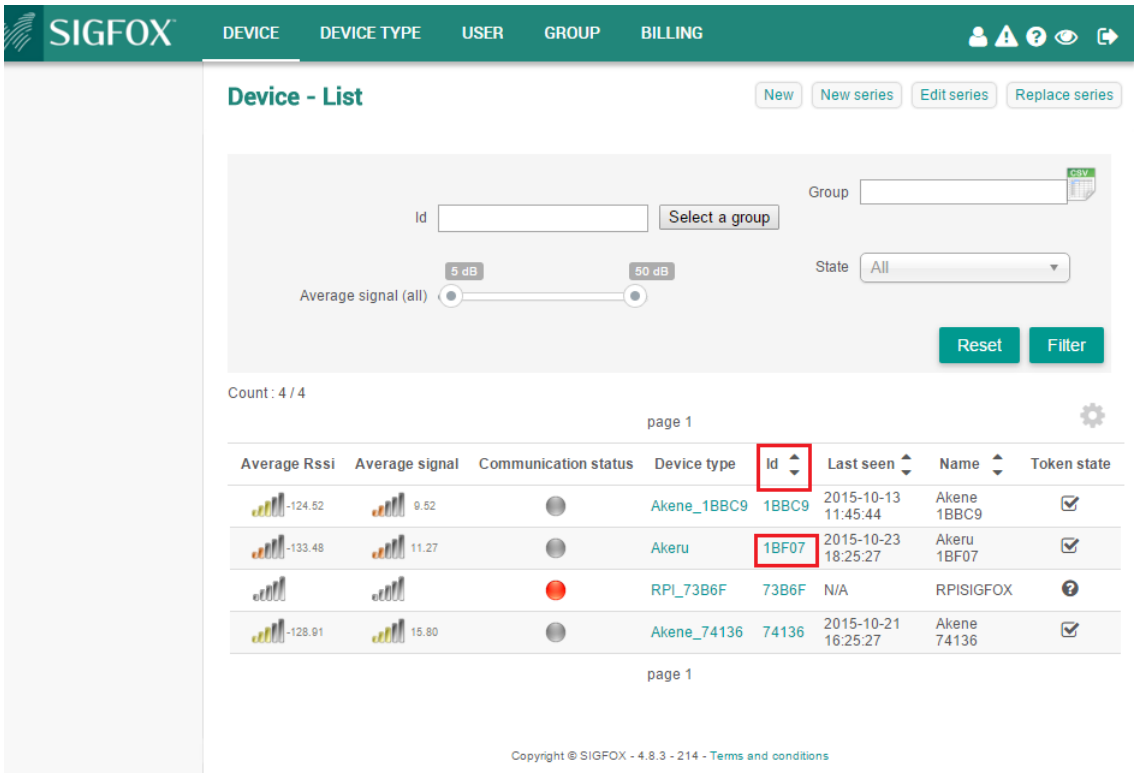

Figure 3.31: SIGFOX - Callback function guild 4

– Then click on Messages. Here 3.32 is list of received messages and they are pushed follow callbacks setting. They are pushed to SAP HANA Cloud Platform follow the callbacks setting.

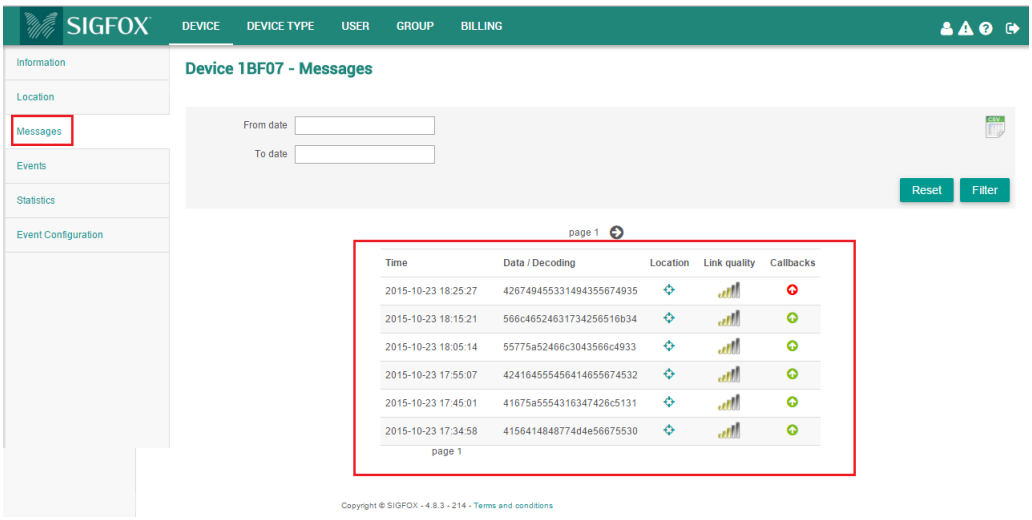

Figure 3.32: SIGFOX - Callback function guild 5

– In addition, there is also statistics 3.33 for messages to observe statistics on the left menu

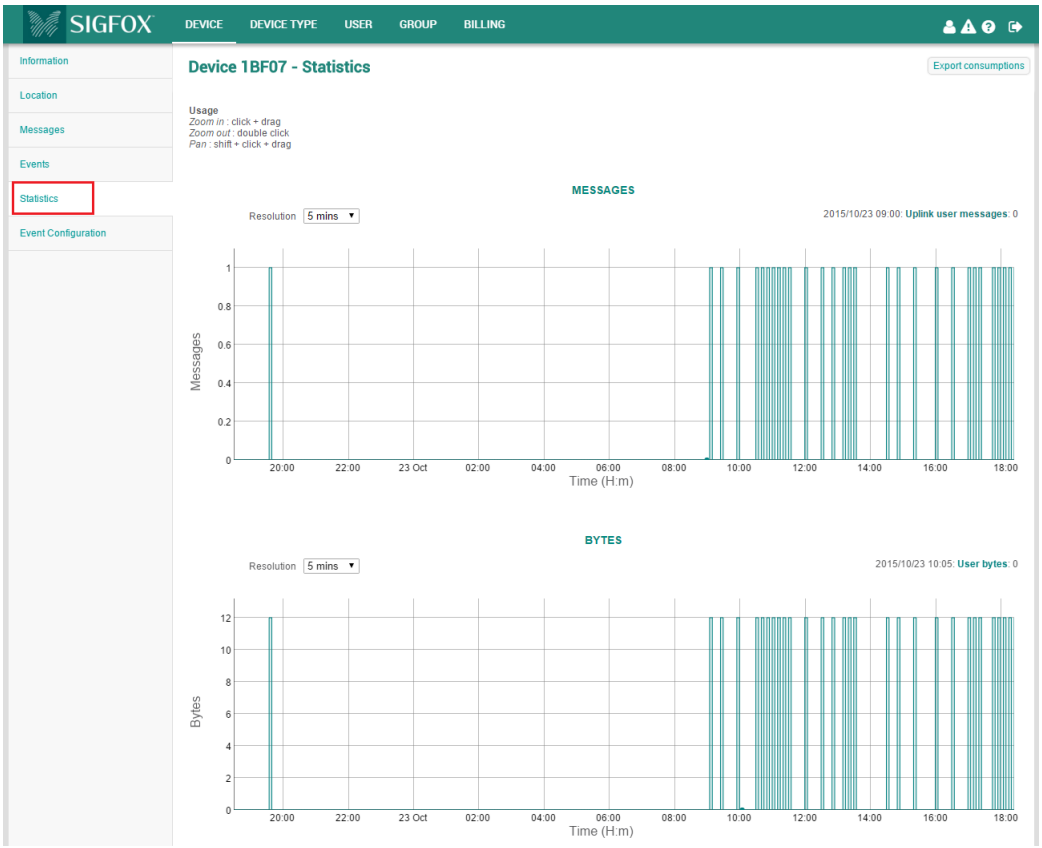

Figure 3.33: SIGFOX - Callback function guild 6

– From now on, everytime SIGFOX back-end server receives any data from Arduino devices, it will push the data according to callbacks setting.

2.3 Downlink Callbacks

This feature [28] will be used to implement the re-keying in future work. IoT devices, Arduino in particular can receive a message from a server via SIGFOX. Arduino sends a message to SIGFOX with an acknowledgement to clarify that it is waiting for a response from servers. SIGFOX back-end pushes the message to servers, and receives a response as a json format. SIGFOX sends back the response to Arduino. We used Raspberry Pi acting as a server in this implementation.

2.3.1 Configuration

– Setup the downlink callbacks on SIGFOX back-end

1. Login to SIGFOX back-end.

2. Access the setting for device type.
3. Right click 3.34, and choose Edit

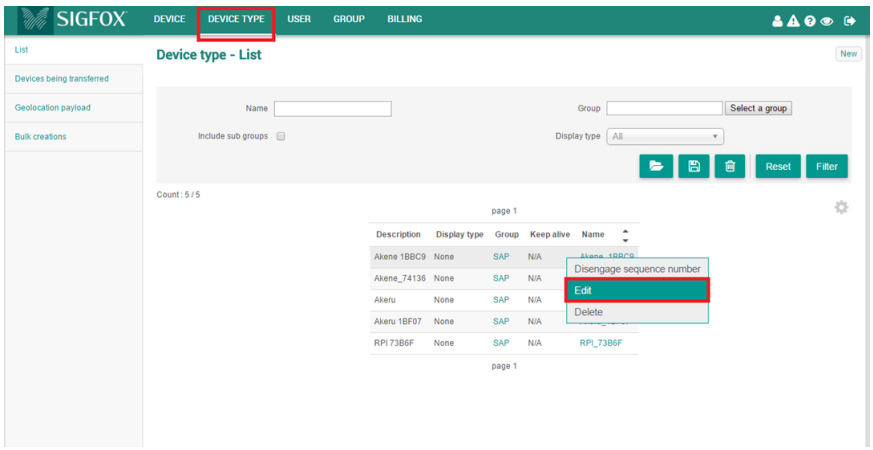

Figure 3.34: Device type access

4. Choose **CALLBACK** for Downlink mode, and hit **OK** 3.35

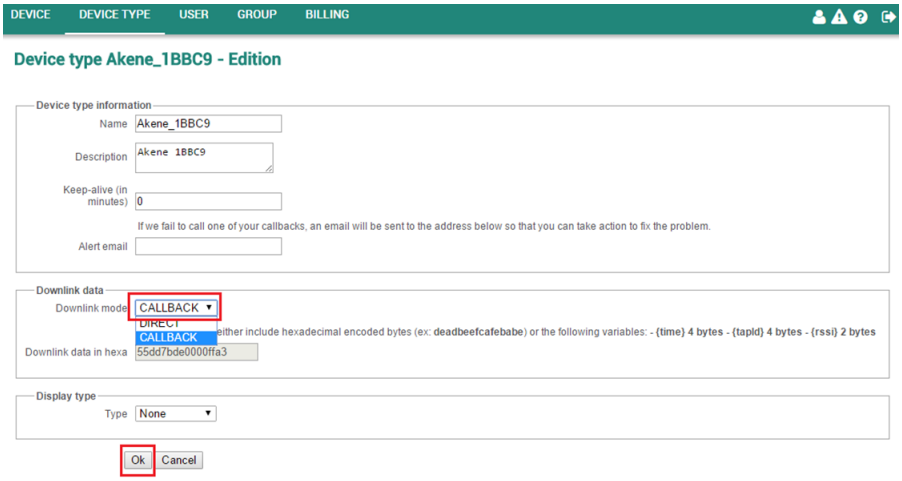

Figure 3.35: Downlink mode selection

5. Go to the callback setting, choose settings as figure 3.36

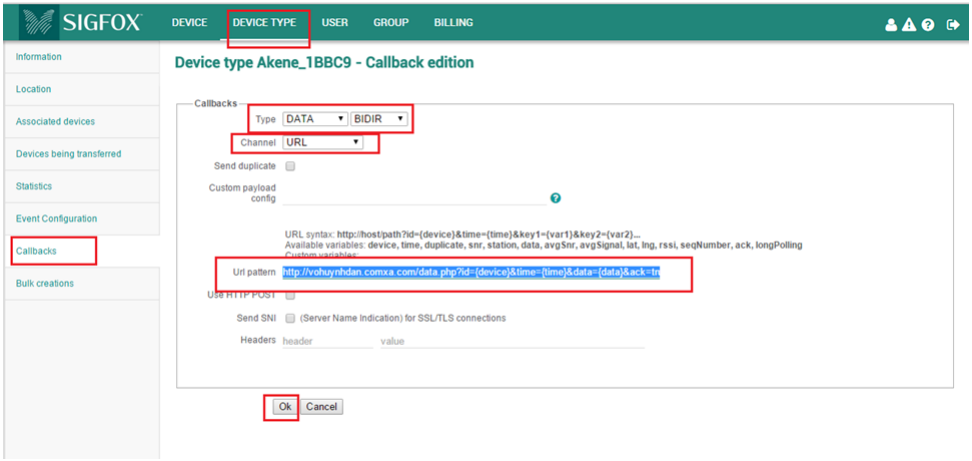

Figure 3.36: Callback configuration

6. **URL Pattern:**

```
http://vohuynhdan.comxa.com/data.php?id={device}&time={time}&
  data={data}&ack=true

http://vohuynhdan.comxa.com/data.php is a host running PHP
  code to send back the json file to SIGFOX. It could be a
  server running on RasPi.

For example : http://217.108.215.247:12080/index.php?id={
  device}&time={time}&ack=true

ack=true : This is important!
```

7. the **data.php**, and **index.php** have the same code as follows:

```
<?php
  $_id = $_GET["id"];
  $_time = $_GET["time"];
  $_ack = $_GET["ack"];
  $_data = $_GET["data"];

  if ( $_ack == "true" ) {
    echo "{";
    echo "\"". $_id ."\" : { \"downlinkData\" : \"
```

```
    deadbeefcafebabe\" }";
      echo "}";
11  }
    header("HTTP/1.0 200 OK");
13  header("Content-Type : application/json");
?>
15
```

deadbeefcafebabe is data server would like to send back to Arduino

8. Remember to choose those settings below in order to activate the feature 3.37

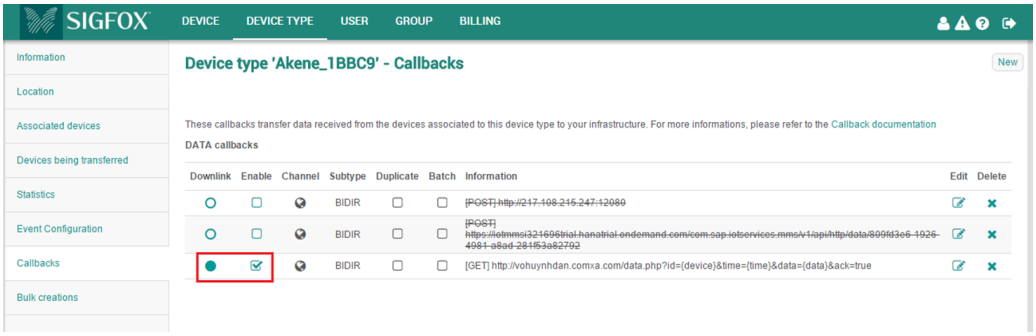

Figure 3.37: Activate downlink mode for callbacks

2.3.2 Deployment

- Open the Example.ino file
- Build, and upload it
- Open the Serial Monitor
- Observe *Serial Monitor*, it shows the message **deadbeefcafebabe** 3.38 from server.

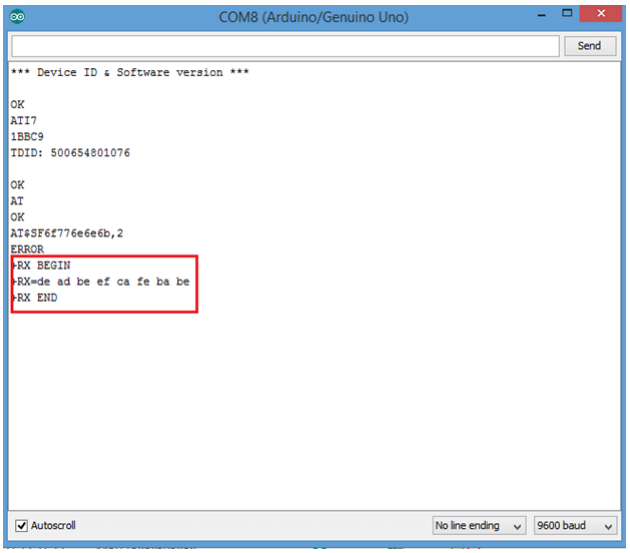

Figure 3.38: Serial Monitor for Downlink callbacks

– From SIGFOX back-end, all arrows should be green 3.39

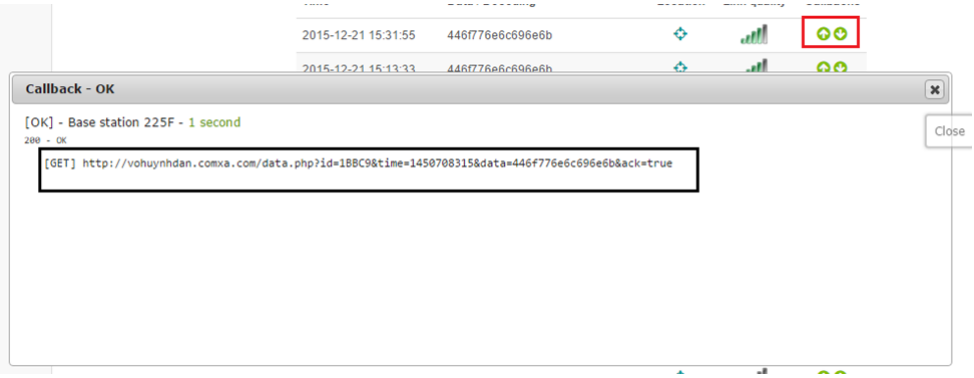

Figure 3.39: Callbacks status 1

– The information which is sent from server could also be checked in figure 3.40

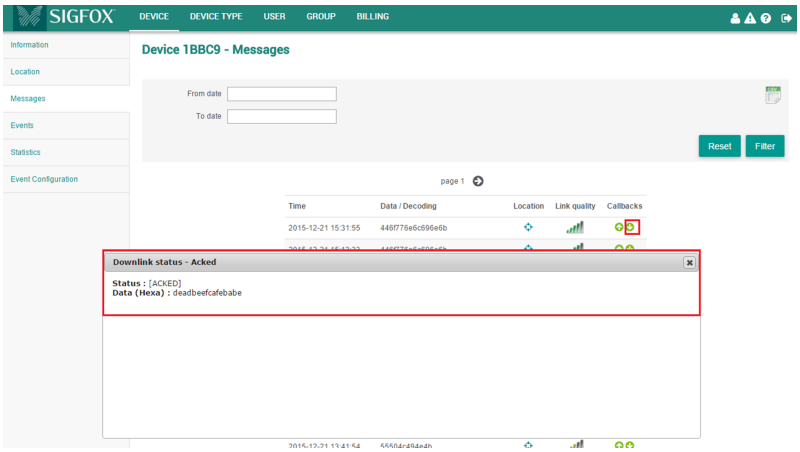

Figure 3.40: Callbacks status 2

We have implemented the cryptographic scheme on PLCs, Raspberry Pi, and Intel Edison as well as configured all components in Internet of Things.

1 Hana Cloud Platform Internet of Things Services

1.1 Introduction

SAP HANA Cloud Platform IoT Services [8] 4.41 are designed to facilitate and support the implementation of Internet of Things applications. The services provide interfaces for registering devices and their specific data types, sending data to a data base running in SAP HANA Cloud Platform (HCP) in a secure and efficient manner, storing the data in HCP as well as provide easy access to the data stored.

The respective services are distributed across two main components: Remote Device Management Service (RDMS) and Message Management Service (MMS). Moreover, there is a web-based interface called IoT Services Cockpit which provides easy access to the various services.

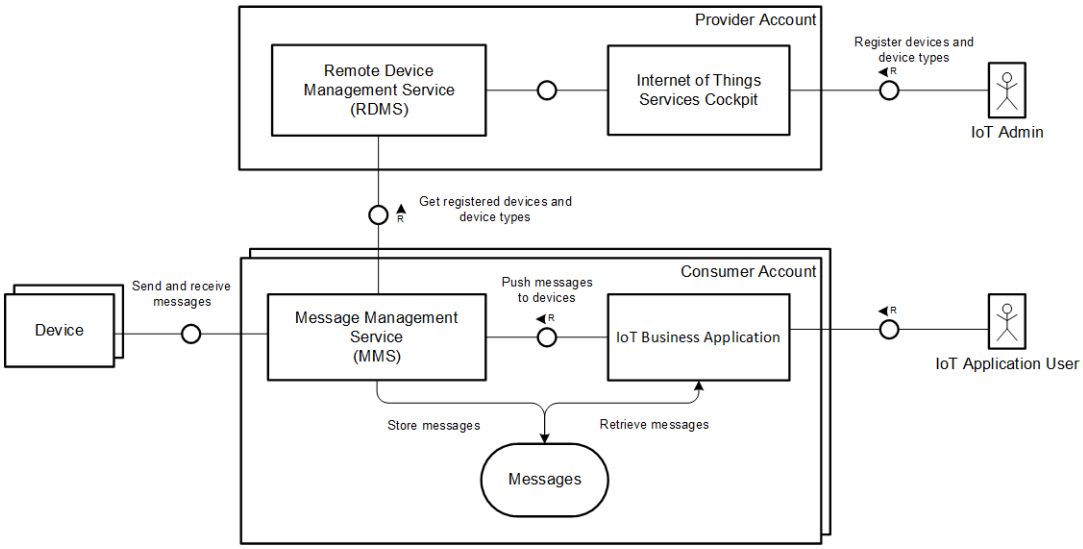

Figure 4.41: System Architecture

MMS provides various APIs that can be used by devices to send data to the SAP HANA Cloud Platform. It processes the data and persists the data in the attached databases. There may be other use cases, though, which require forwarding the data to other Message Brokers or Event Stream Processors.

The Internet of Things Services Cockpit is the main interface for users to interact with the Remote Device Management Service (RDMS). It can be used to register new devices, to define the schema of messages (devices types and message types) they can send and/or receive, as well as to establish the necessary trust relationship devices need to interact with MMS. The Internet of Things Services Cockpit and RDMS are provided as Cloud services and can be used through subscriptions.

1.2 Related Work

1. SAP HCP Services configuration is the last part of connectivity; to receive, and store messages from **SIGFOX**, and **devices**.
   - Enable, and configure SAP IoT Service
   - Enable, and configure HANA XSJS, and HANA DB services
   - Deploy DataGraph application on SAP HCP
2. REST API is built to expose the collected data.
   - Build REST API with OData
   - Build REST API with HANA XSJS
   - Grant user permissions to use REST API

   - Develop Data removal feature, and retrieve information by ID from the xsjs REST API

3. A Dashboard is developed with the integration of FreeDataMap with SAP UI5 to expose the data.
   - Build a dashboard by SAP HANA Web IDE service as a HTML application.
   - Do a workaround to retrieve information from the REST API built by HANA XS
   - Integrate FreeDataMap on the Dashboard by using SAP UI5
   - Build a java application named GraphData to decrypt data, and plot decrypted values as a line graph.

1.3 Configuration

1.3.1 HANA Cloud Platform for IoT

- Get HANA Cloud Platform Developer Account, and Enable Internet of Things Services
  1. Create a trial account.
  2. Log on the SAP HANA Cloud Platform Cockpit.
  3. Go to the **Services** tab.
  4. Select to the **Internet of Things Services** 4.42 entry

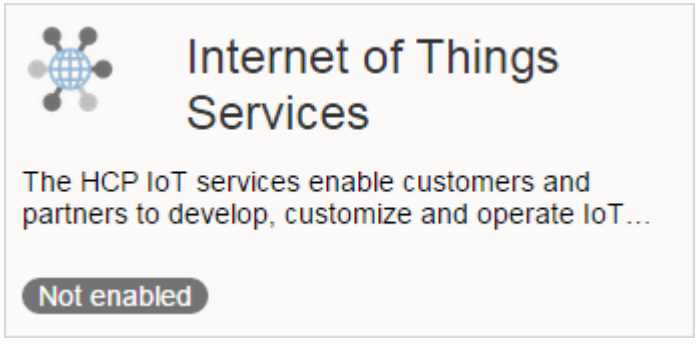

Figure 4.42: IoT service

  5. Press **Enable** 4.43.

- Accessing the Internet of Things Services Cockpit. The Internet of Things Cockpit allows to securely register devices. A device must be of a certain device type which is specified by its supported message types. Both types can be defined in the IoT Cockpit as well.

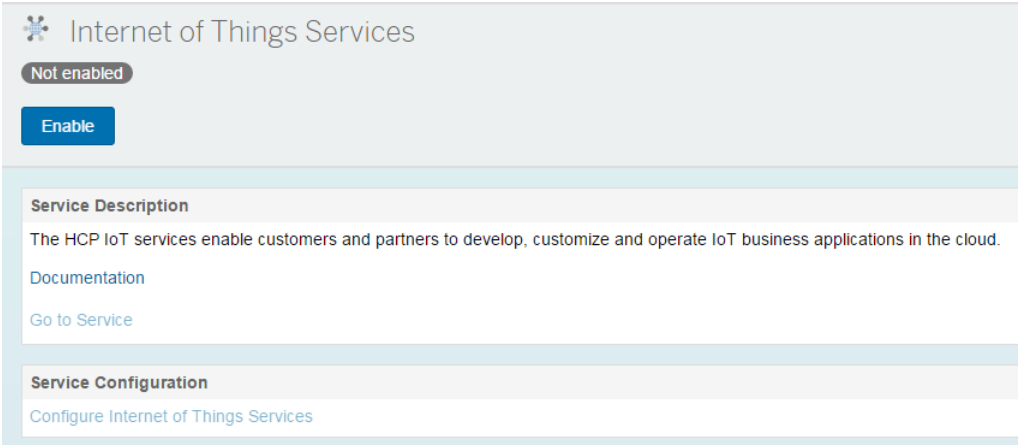

Figure 4.43: Enable IoT service

1. Choose **Go to Service** for the Internet of Things Services subscription.
2. **The Internet of Things Services Cockpit** 4.44 is displayed.

– Create Device Information in **Internet of Things Services Cockpit**[35]. The deployment of the Message Management Service (MMS) is necessary to be able to send data from a device to consumer account.

1. Press on the **Go to Service** icon of the Internet of Things Services subscription. The Internet of Things Services Cockpit will open.
2. Create Device Type:
   - Press on **Device Types** tile in the Internet of Things Services Cockpit.
   - Press on the + button to add a new device type. Enter a Name for the device type.
   - Press on **Create** to continue 4.45.
3. Create Message Type:
   - Press on**Message Types** tile in the Internet of Things Services Cockpit.
   - Press on the + button to add a new message type.
   - Enter a Name for the message type. **The Message Type Name should be exactly the same Device ID for later purpose**.
   - Select a **Device Type** from the dropdown list.
   - Select a Direction **"From Device"**.
   - Enter a Name and select a Type for the first row of the Fields table.
   - Add additional Fields by pressing on the + button on the top right corner of the **Fields** table.
   - There are 3 values including: **ID**, **Data**, and **Timestamp** 4.46 with **String** type.

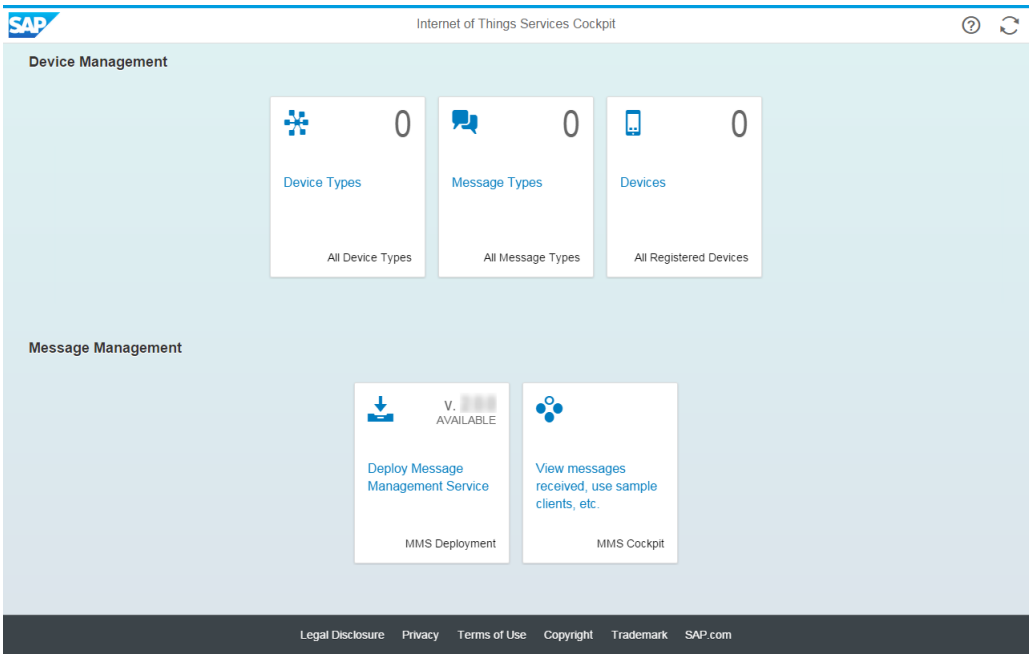

Figure 4.44: The IoT Services Cockpit

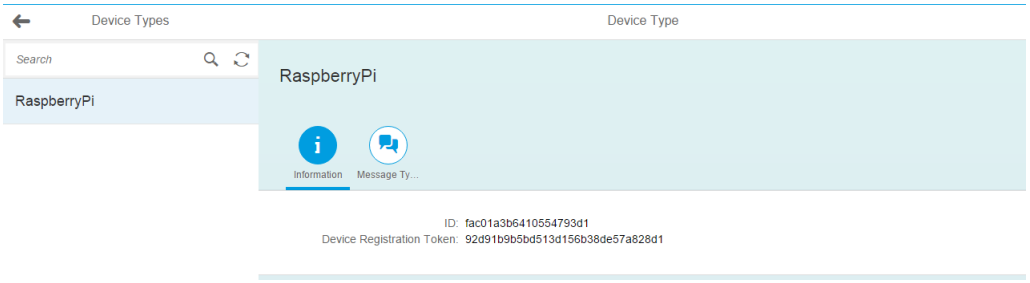

Figure 4.45: Create Device Type in the IoT Services Cockpit

- Press on **Create** to continue.

4. Create Device:
   - Press on **Devices** tile in your Internet of Things Services Cockpit.
   - Press on the + button to add a new device.
   - Enter a Name for the device.
   - Select the **Device Type** for the new device from the drop down menu.
   - Press on **Create** to continue 4.47.
   - A pop-up window Device Token Generated including the Token ID generated for the new device.
   - **Copy the generated Device Token** since it is needed on the device as OAuth credential for secure communication with the services.

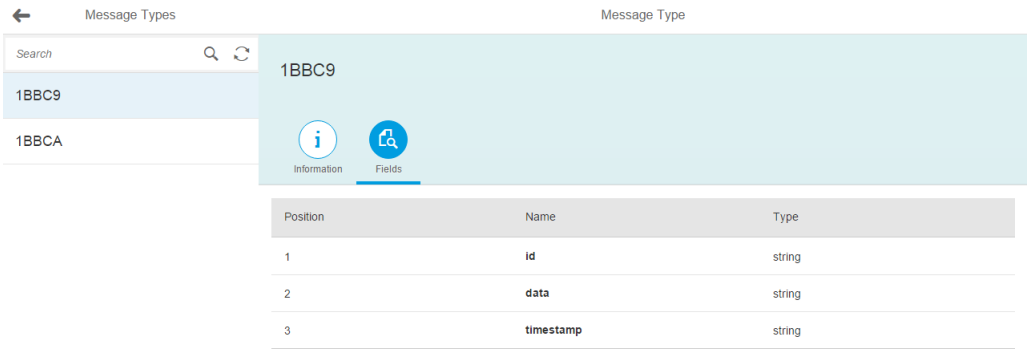

Figure 4.46: Create Message Type in the IoT Services Cockpit

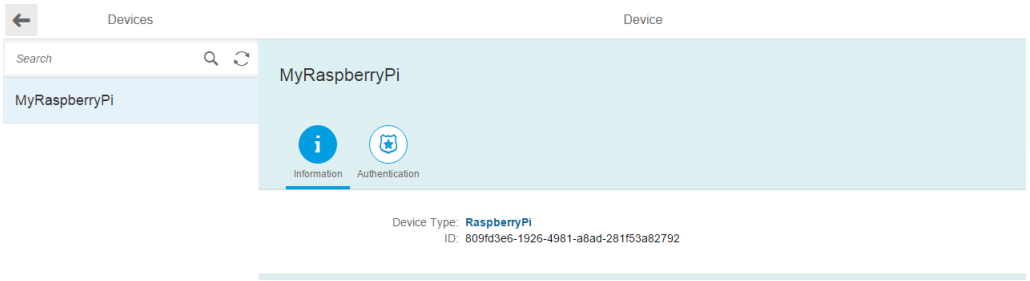

Figure 4.47: Create Device the IoT Services Cockpit

5. Communication with the HCP IoT Services is protected by 2 different authentication mechanisms:
   - IoT Devices use **OAuth authentication** with a respective OAuth token for an individual device provided via the IoT Cockpit.
   - Applications that trigger interactions with devices via IoT Services or consume data provided by these use **Basic Authentication** with the respective credentials for the individual user of the HCP account.
6. Deploy the Message Management Service (MMS):
   The deployment of the Message Management Service (MMS) step needs to be done from the Deploy Message Management Service 4.48 tile in the IoT Services Cockpit and deploys/starts the Message Management Service that takes care of receiving data from IoT Devices and sending to these.
   Then assign the Role IoT-MMS-User 4.49 for the newly deployed iotmms Java Application (otherwise you will be denied access to the iotmms Application URL with an HTTP Status 403 Error). To do so:

   - Go to the **Java Applications** tab in your SAP HANA Cloud Platform cockpit of the account.
   - Choose the iotmms application.
   - Choose the **Role** tab of the Application details.

Figure 4.48: Deploy Message Management Service

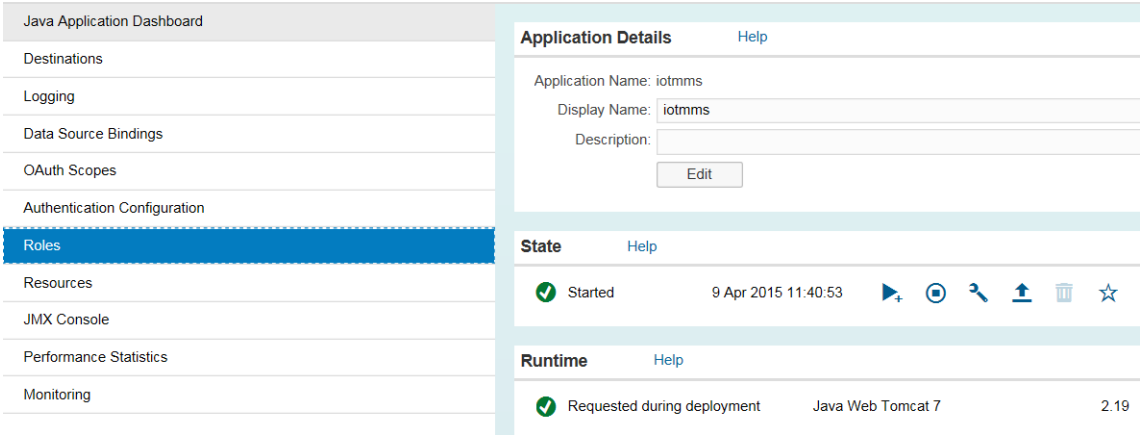

Figure 4.49: Role assignment of IoT-MMS-User

- Do the assignment of the role to your user 4.50.

The Push Service API is protected by means of Basic Authentication. The usage of this mechanism needs to be enabled explicitly in the SAP HANA Cloud Platform Cockpit (otherwise you will receive a Login Page instead of a success message as answer for a Push API request).

In order to enable Basic Authentication the following steps need to be executed:

- Go to the SAP HANA Cloud Platform Cockpit and select the iotmms application from the Java Application section.
- Go to the **Authentication Configuration** tab.
- Change type to **Custom**.
- Add **User name** and **password** to the selection in the Form section.
- Restart the iotmms application.

– A test to ensure data will be collected by the service. Once MMS is deployed and you have correctly done role assignment as well as the Authentication

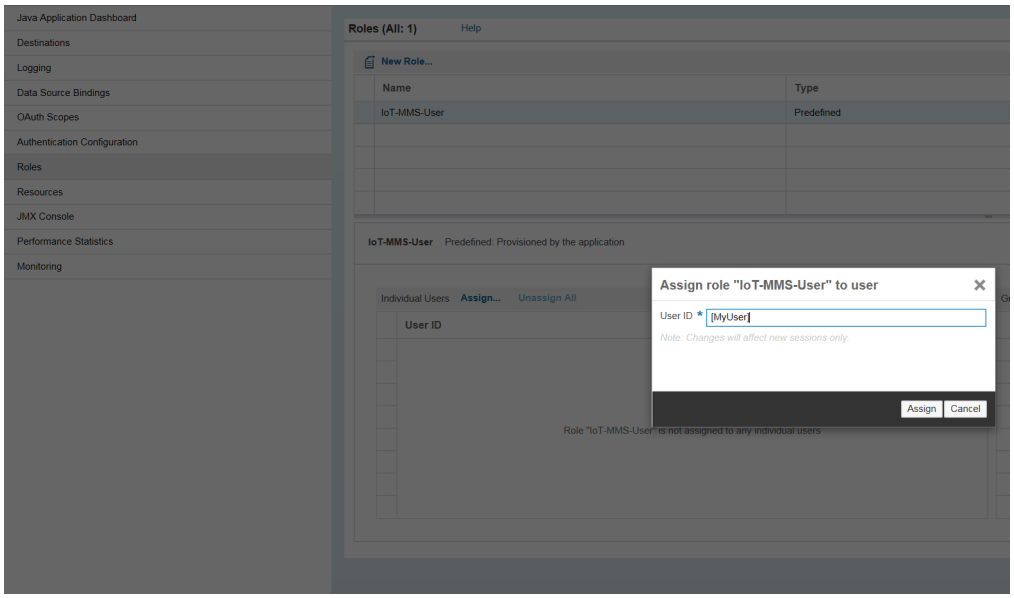

Figure 4.50: Role assignment of IoT-MMS-User 2

Configuration for the Push service you can click on the iotmms Java application URL in your HCP Cockpit and get to the MMS Cockpit as shown below. It provides access to the MMS API as well as a "Display stored messages" tile for the access to data received from IoT Devices.

1. Go to MMS Cockpit 4.51
2. Click on **Send and receive messages via HTTP** 4.52 tile.
3. Adapt the Device ID
4. Adapt the Message by changing **messageType**, variables of **messages**.
5. Click on **Post** button, if the message is collected, this text will be shown

```
[200] {"msg":"1 message(s) received from device [809fd3e6
    -1926-4981-a8ad-281f53a82792]"}
```

6. Go back to the **Display stored messages**, there will be collected data 4.53.

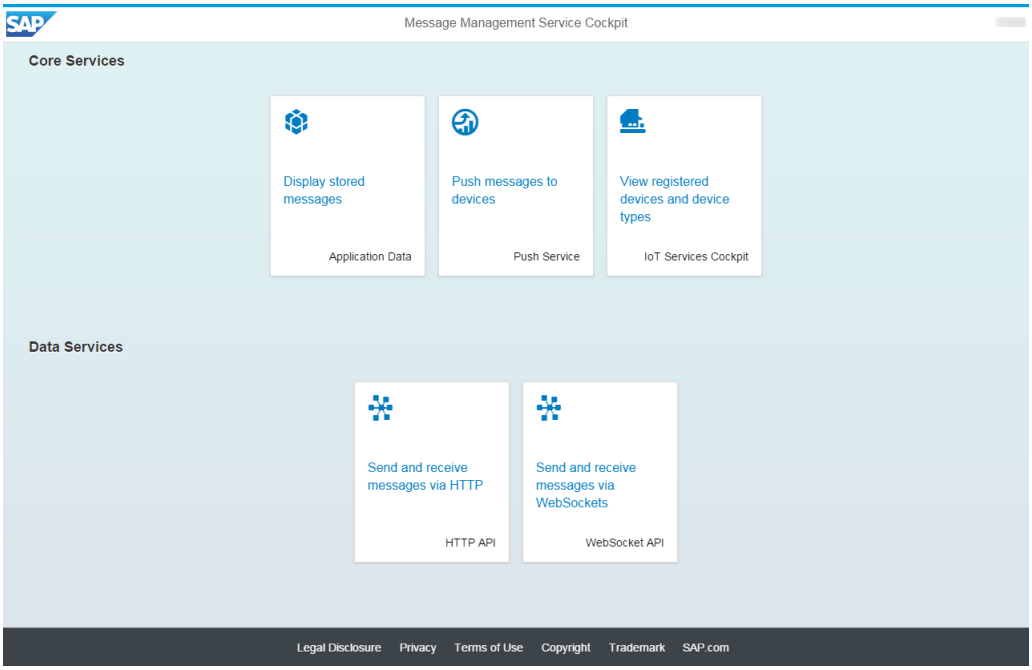

Figure 4.51: MMS Cockpit interface

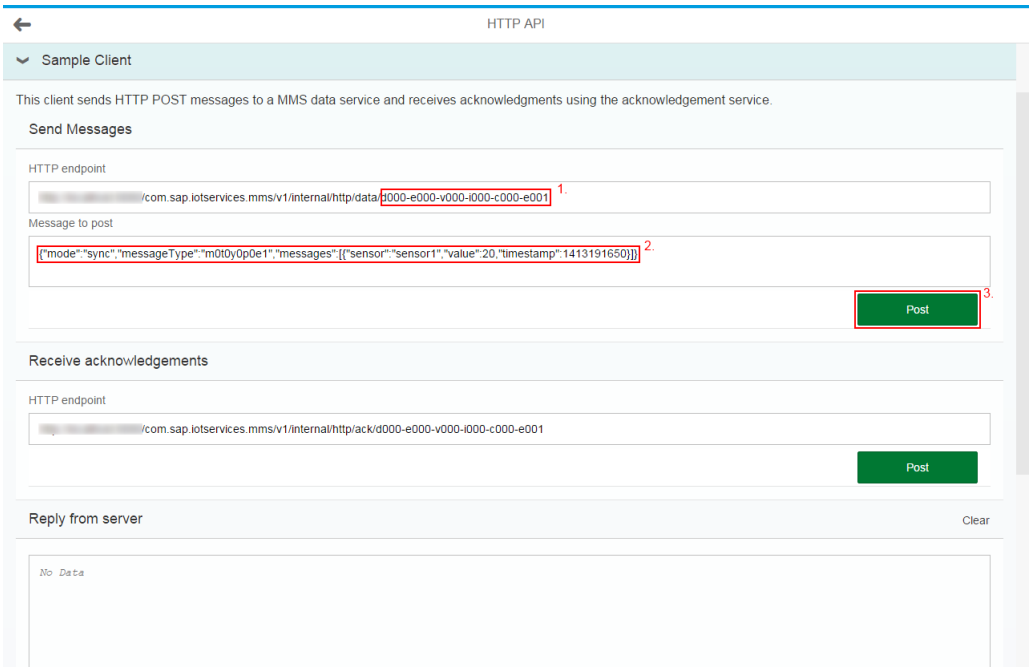

Figure 4.52: HTTP Test API

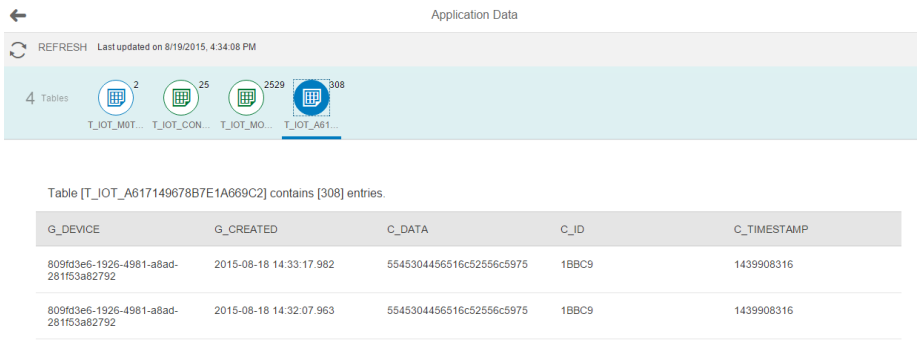

Figure 4.53: Display stored messages Dashboard

1.3.2 Maven Installation

To install Apache Maven on Windows, download the zip file of Maven, and then extract it into the folder to install, and set up environment variables. JDK 1.7 is necessary.

1. Set up environment variable JAVA_HOME
   - Right click on **windows**, choose **System** 4.54

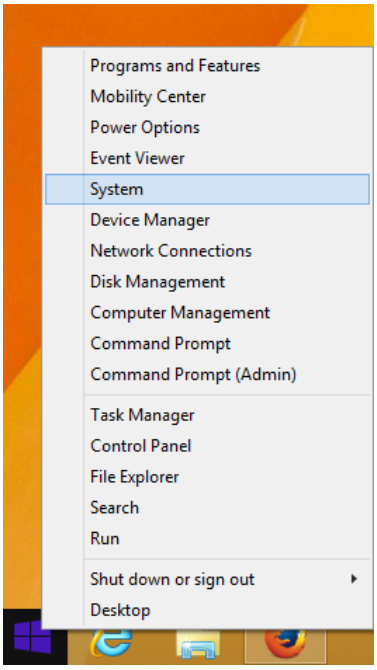

Figure 4.54: JDK Installation Guide 1

   - Click **Advanced System Settings** in a new pop-up windows
   - In **System properties**, choose **Advanced** tab. Then click on **Environment Variables ...**
   - Click **New** 4.55 to install JAVA_HOME

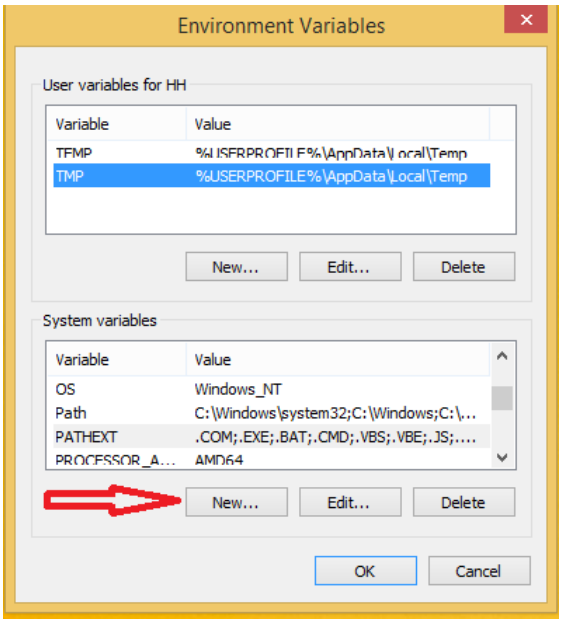

Figure 4.55: JDK Installation Guide 2

- In **Variable name:** JAVA_HOME. Type the link of JDK where you installed in your computer 4.56. Then click **OK**

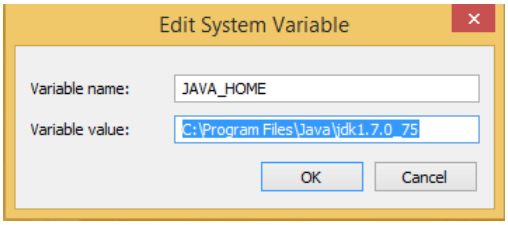

Figure 4.56: JDK Installation Guide 3

- Next, find variable **Path** 4.57 in **System variables**, click on **Edit** and type
  $\%JAVA\_HOME\%\backslash bin$;

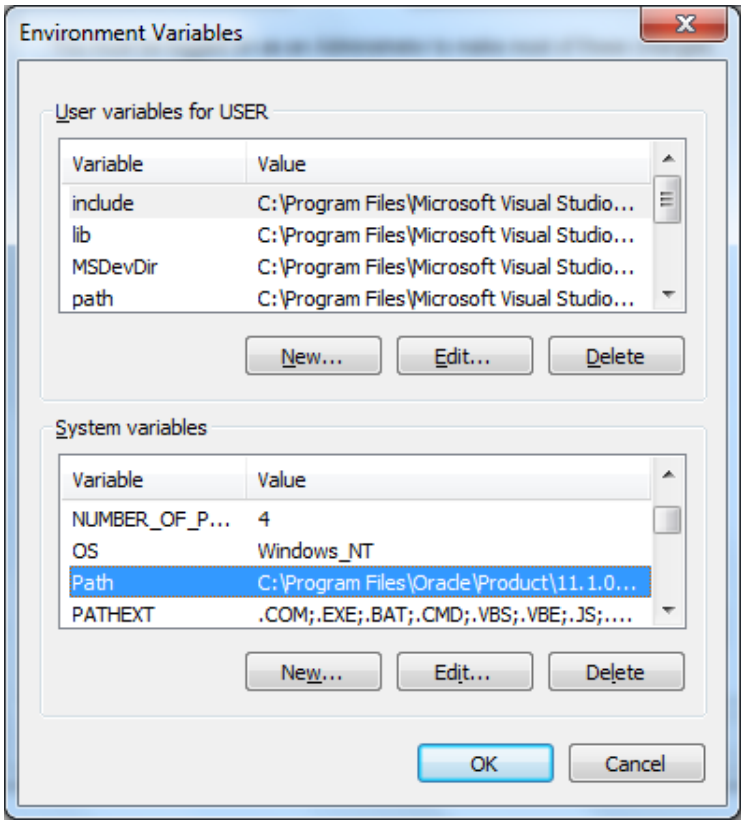

Figure 4.57: JDK Installation Guide 4

- Click **OK** 4.58 to finish the process.

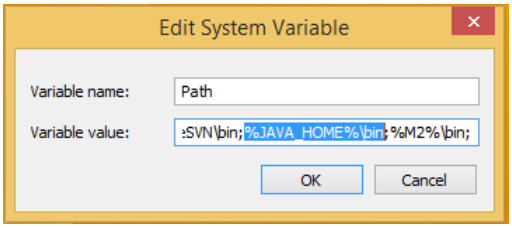

Figure 4.58: JDK Installation Guide 5

- To test whether it is installed successfully or not. Open **cmd**, then type **java -version** to check JDK version, and **javac -version** to check javac version. If it shows like in the figure 4.59, you succeeded in installing.

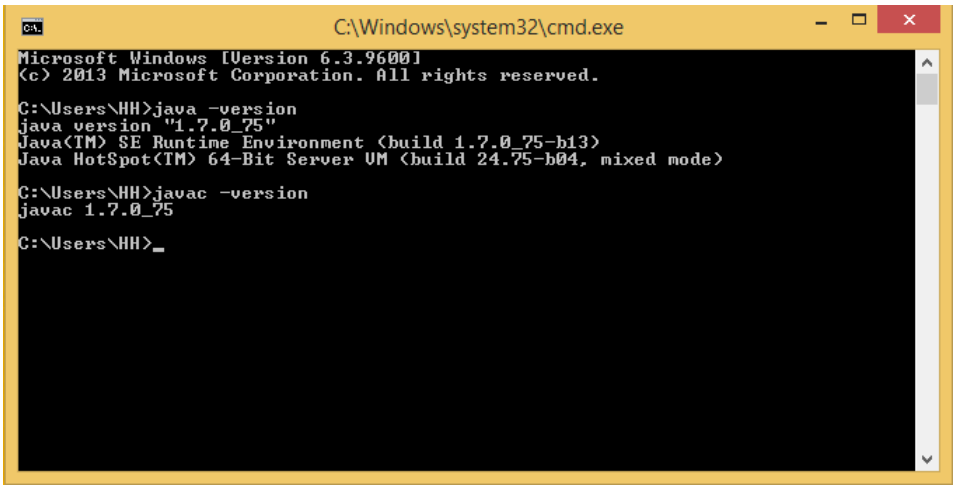

Figure 4.59: JDK Installation Guide 6

2. Install Apache Maven, and set up environment variable for Maven
   - Download Apache Maven, and extract to the place you want to store in your drive.
   - Set up Environment variable for Maven 4.60

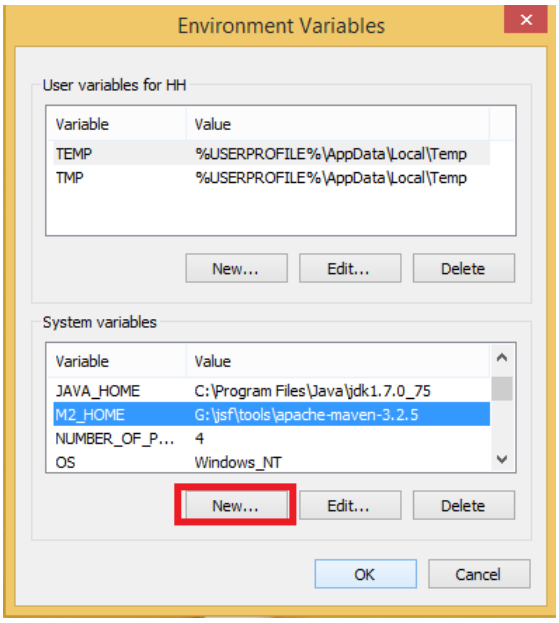

Figure 4.60: Maven Installation Guide 1

- M2_HOME in Variable name, and link to the directory stores Maven in Variable value 4.61

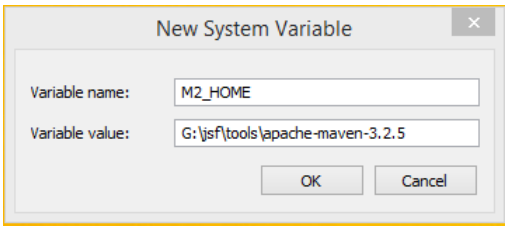

Figure 4.61: Maven Installation Guide 2

- include $\%M2\_HOME\%\backslash bin$; into variable **Path** 4.62 4.63

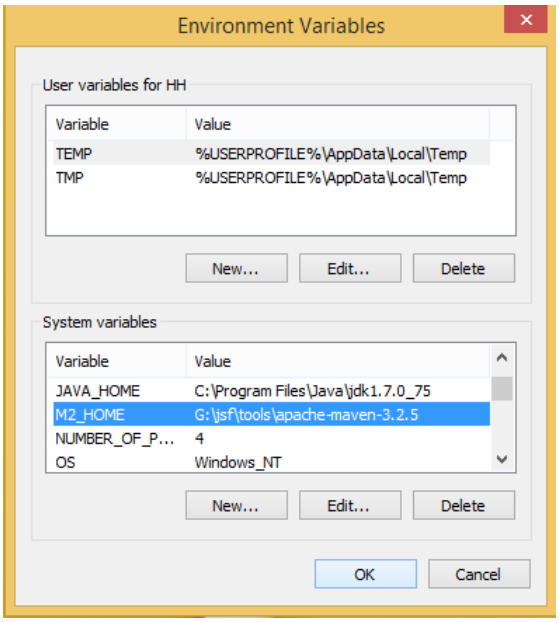

Figure 4.62: Maven Installation Guide 3

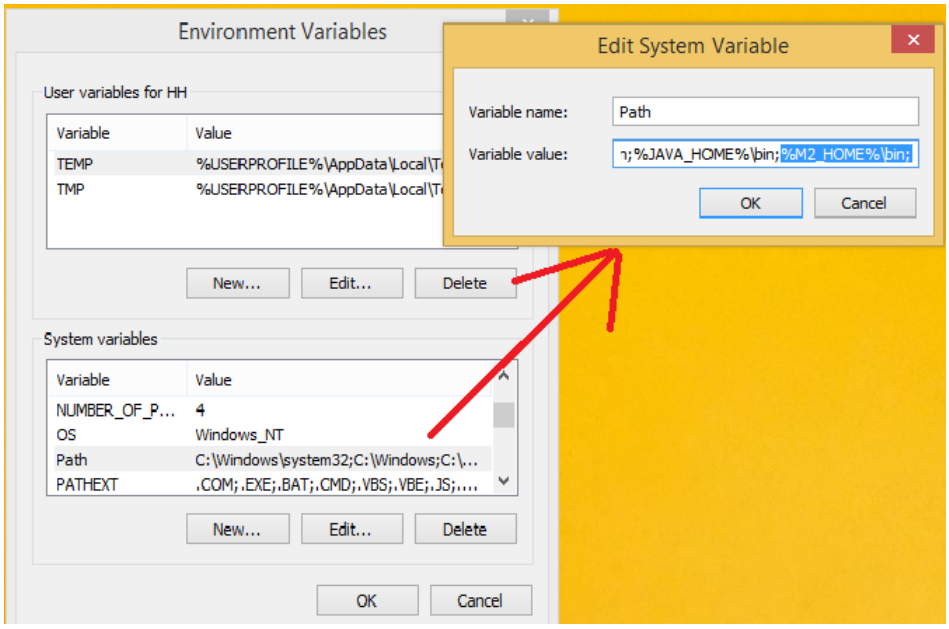

Figure 4.63: Maven Installation Guide 4

- To check whether it is successfully installed or not, open **cmd** and type **mvn -version** 4.64

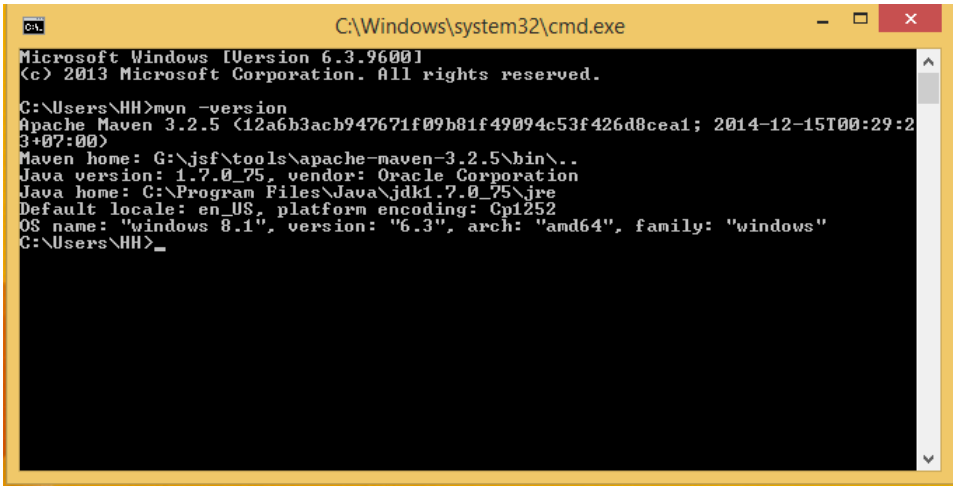

Figure 4.64: Maven Installation Guide 5

1.4 Implementation

1.4.1 Consume the messages with Web Application

To display encrypted data stored in database, thus a java application named data graph is built, and deployed. However, the data is encrypted. Therefore, it cannot be used for visualization. Therefore it must be decrypted in order to display the graph. The figure 4.65 illustrates for the type of visualization which is needed to achieve.

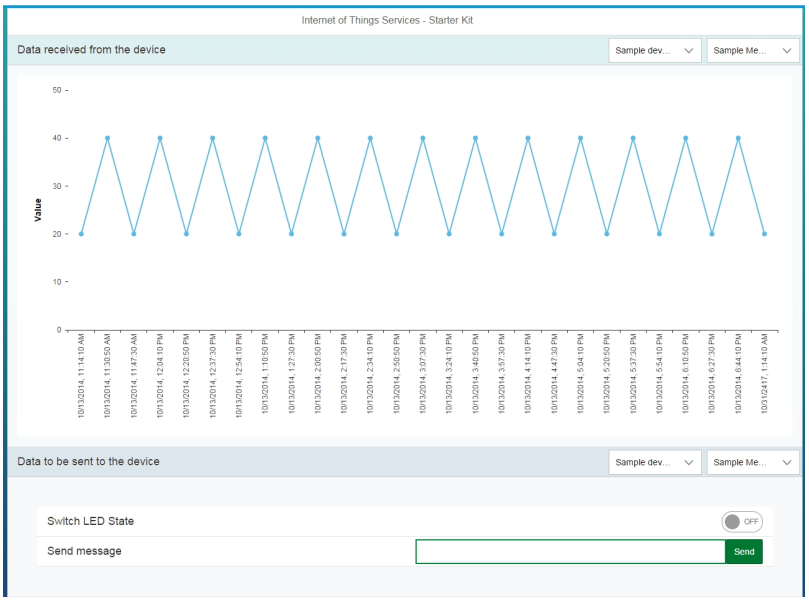

Figure 4.65: Visualization

In the next step, the applications is built by Maven:

1. Build the project with Maven using this piece of code, it will produce the .war file to be used in the target directory.

```
1 mvn clean install
```

2. Deploy the .war file using the **HCP Cockpit** > **Java Application** > **Deploy Application** 4.66, but don't start it yet.

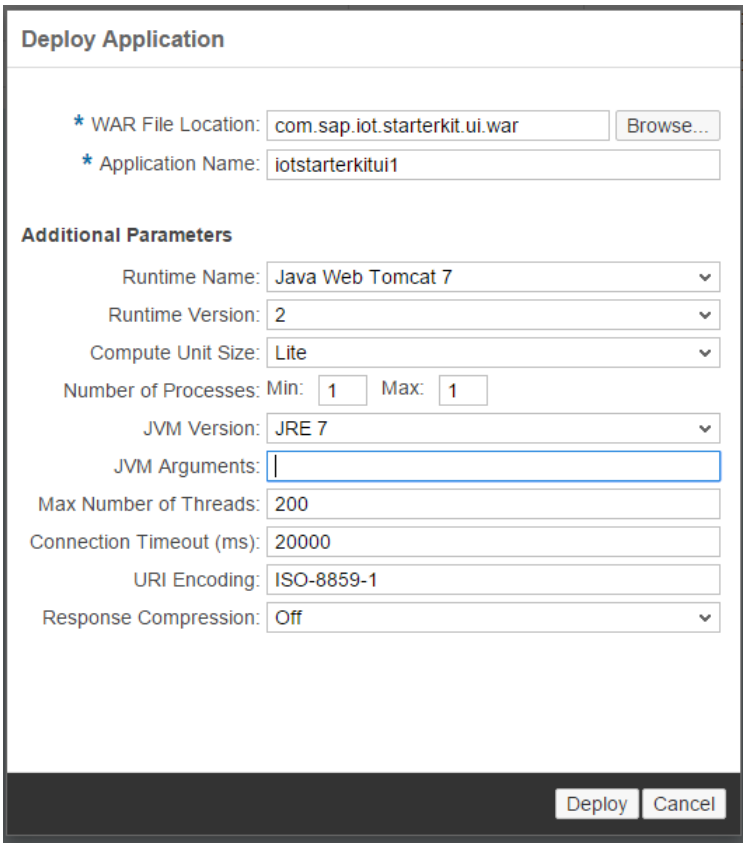

Figure 4.66: Deploy application

3. Select the application and go to **Destinations** > **Import from file** (use the provided file **iotmms** in the **com.sap.iot.starterkit.ui/destinations** folder) 4.67 to correctly route requests from your User Interface to the iotmms application providing the MMS Services.

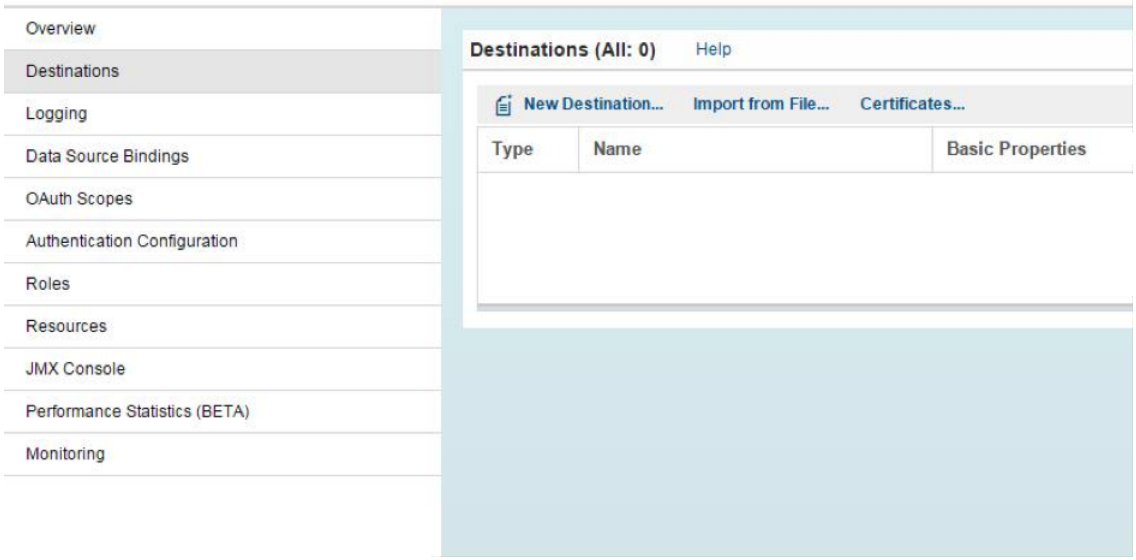

Figure 4.67: Destination

4. Within this process adapt **URL**, **User** and **Password** fields 4.68 with the information for your account.

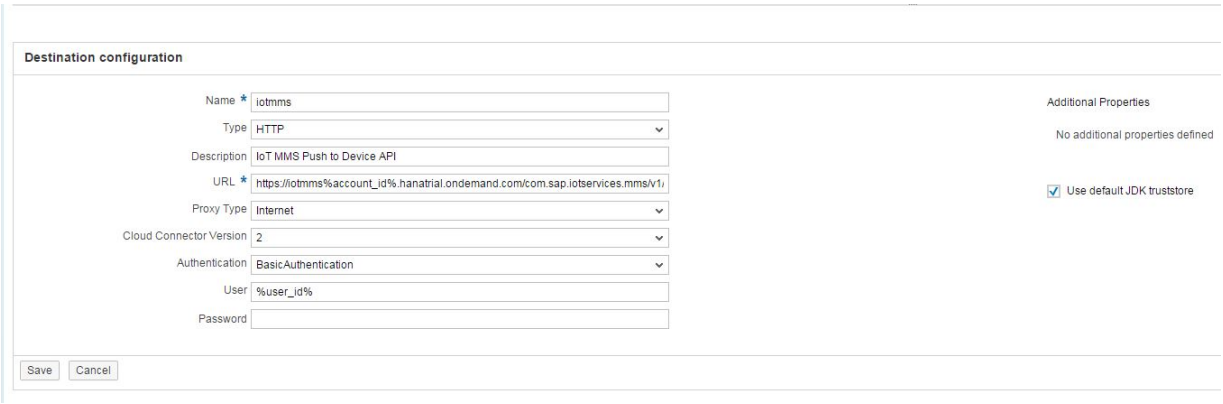

Figure 4.68: Destination configuration 1

5. Select the application and go to **Destinations** > **Import from file** (use the provided file iotrdms in the **com.sap.iot.starterkit.ui/destinations** folder) 4.69 to correctly route requests from your User Interface to the iotrdms application providing the RDMS Services.
6. Create a new **Data Source Binding** 4.70 4.71 to ensure that both the iotmms as well as your UI application use the same database schema.
7. Finally start your UI application by clicking at the URL for the running application and use it.

1.4.2 Implemented Code

Before the data is used, it is decrypted to get the real value. In the file

\com.sap.iot.starterkit.ui\src\main\webapp\index.html

The *createDataSet*() function is modify to adapt the situation, **measures** in specific to decrypt the data with the following code:

```
function createDataSet() {
    return new sap.viz.ui5.data.FlattenedDataset({
      dimensions : [ {
        name : "Timestamp",
        value : {
          path : "sensor>C_TIMESTAMP",
          formatter : function(oValue) {
            //can be a string primitive in JSON, but we need a number
            if ((typeof oValue) === "string") {
              oValue = parseInt(oValue);
            }
            //ensure that UNIX timestamps are converted to
   milliseconds
            var oDate = new Date(oValue * 1000);
            return oDate.toLocaleString();
          }
        }
      } ],
      measures : [ {
        name : "Value",
        value :{
          path : "sensor>C_DATA",
          formatter : function(oValue) {
            if ((typeof oValue) === "string")
            {
              //Decrypt the data here
              var data = decryption(oValue,id);
              return data;
            }
            return oValue;
          }
        }
      } ],
```

```
33          data : {
              path : "sensor>/",
35          }
        });
37    }
```

The problem was in the **measures**, There is no approaches to obtain two values *C_DATA*, and *C_ID* in the *path* because of the SAP.UI5 library. Therefore, ID value is obtained by the name of selected message which means that name of the message has to be exactly the same as ID of devices. This code below show how a name as ID of selected message is obtained:

```
1 var id;
function createConsumptionPanel(){
3 ...
    id = oSelectedItem.getText();
5 ...
}
```

Finally, the application visualize the data as a line graph 4.72

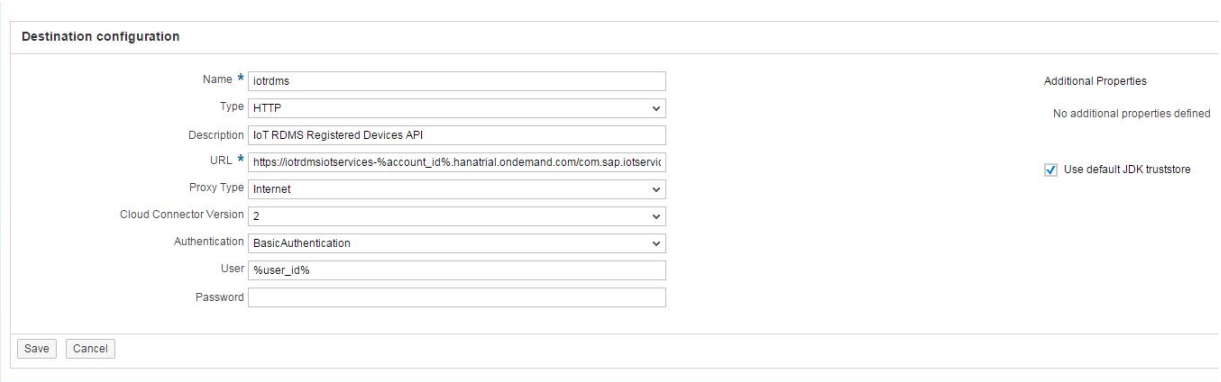

Figure 4.69: Destination configuration 2

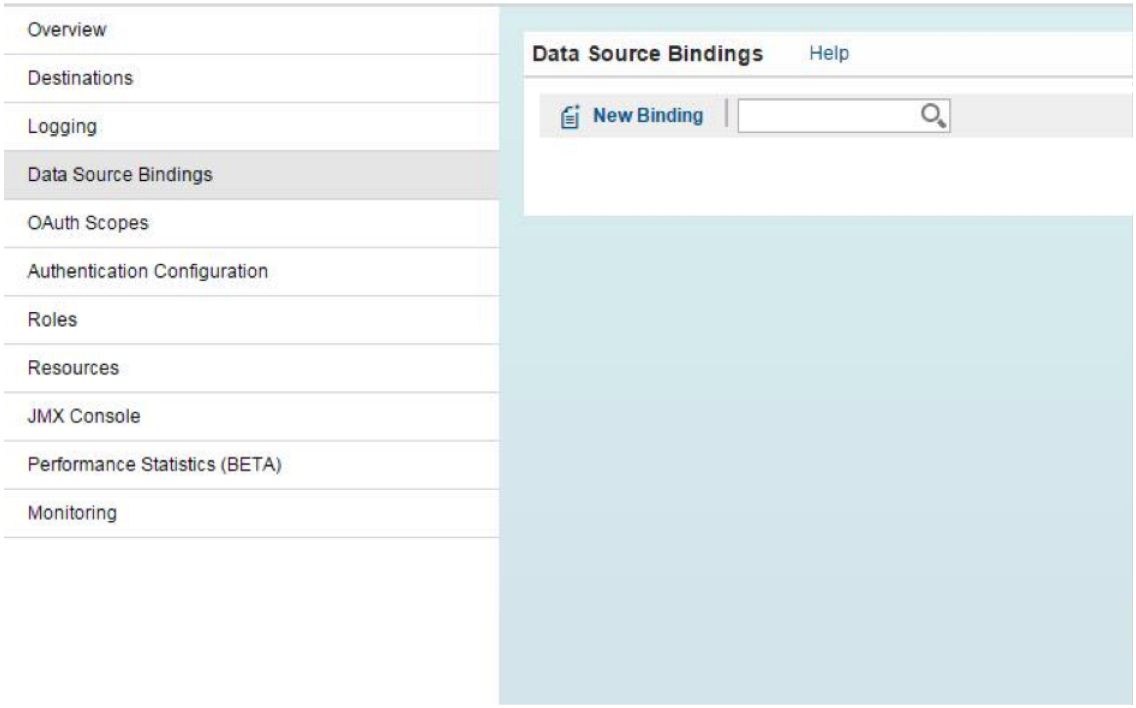

Figure 4.70: Create Data Source Binding 1

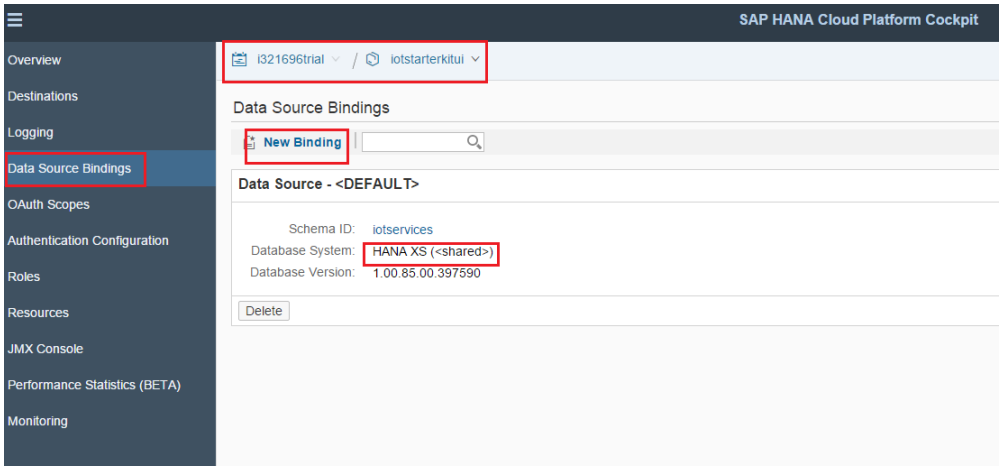

Figure 4.71: Create Data Source Binding 2

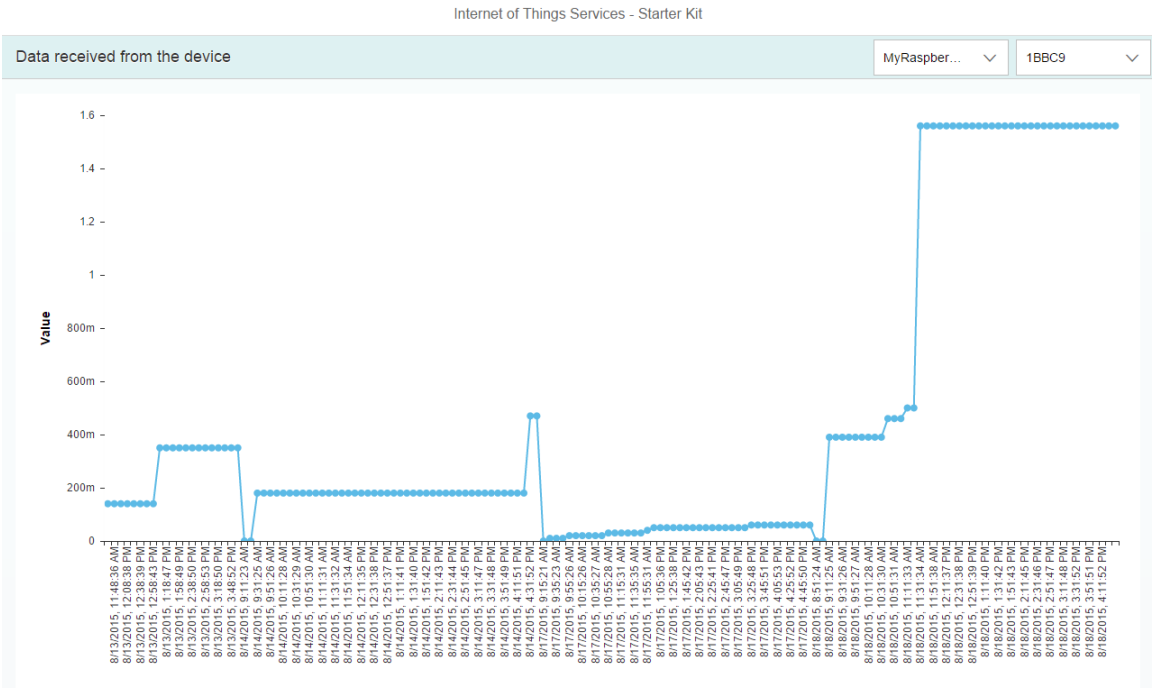

Figure 4.72: Visualization Application

2 REST API

2.1 Introduction

In computing, Representational State Transfer (REST) is a software architecture style for building scalable web services. REST gives a coordinated set of constraints to the design of components in a distributed hypermedia system that can lead to a higher performing and more maintainable architecture.

RESTful systems typically, but not always, communicate over the Hypertext Transfer Protocol with the same HTTP verbs (GET, POST, PUT, DELETE, etc.) which web browsers use to retrieve web pages and to send data to remote servers. REST interfaces usually involve collections of resources with identifiers, for example /people/paul, which can be operated upon using standard verbs, such as DELETE /people/paul.

2.2 Related Work

To expose the data for different purposes, REST API is built to retrieve records directly from the database or by using oData [38] for other bussiness applications. The work includes:

- Configure SAP HANA XSJS services
- Develop REST APIs.
  1. REST API with OData
  2. REST API with SAP HANA XSJS
- Develop data removal feature, and select data by ID feature for REST API XSJS

2.3 Implementation

2.3.1 Configuration

- Message Storage: Messages can be send to the HCP IoT Services using the Message Management Service (MMS) component deployed in the consumer account. By default MMS stores incoming messages into a relational database. The respective message tables are automatically created by MMS based on the device type and message type definitions. More details can be found in the IoT Services documentation. By default, MMS uses the automatically created database (schema) for storing data and the default data source binding for accessing this schema. The name of the underlying database schema is displayed in the "Databases and Schemas" 4.73 section of the HCP Cockpit. The default data source bindings for this schema are also displayed on this page. In the example below, MMS is bound to a shared HANA instance.

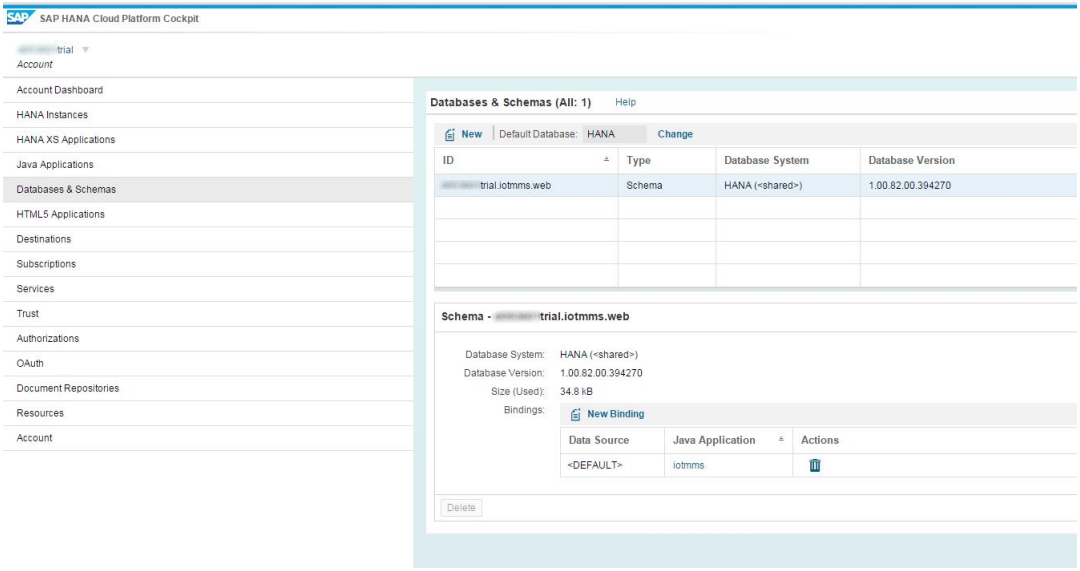

Figure 4.73: Database and Schemas

The default schema binding can be changed to point to a dedicated HANA instance or any other database available in HCP.

– Creating a HANA XS [39] instance:

1. For the purpose of HANA XS development, a HANA XS database instance will be provided on HCP account-level in HCP. The instance 4.74 can be created using the HCP cockpit.

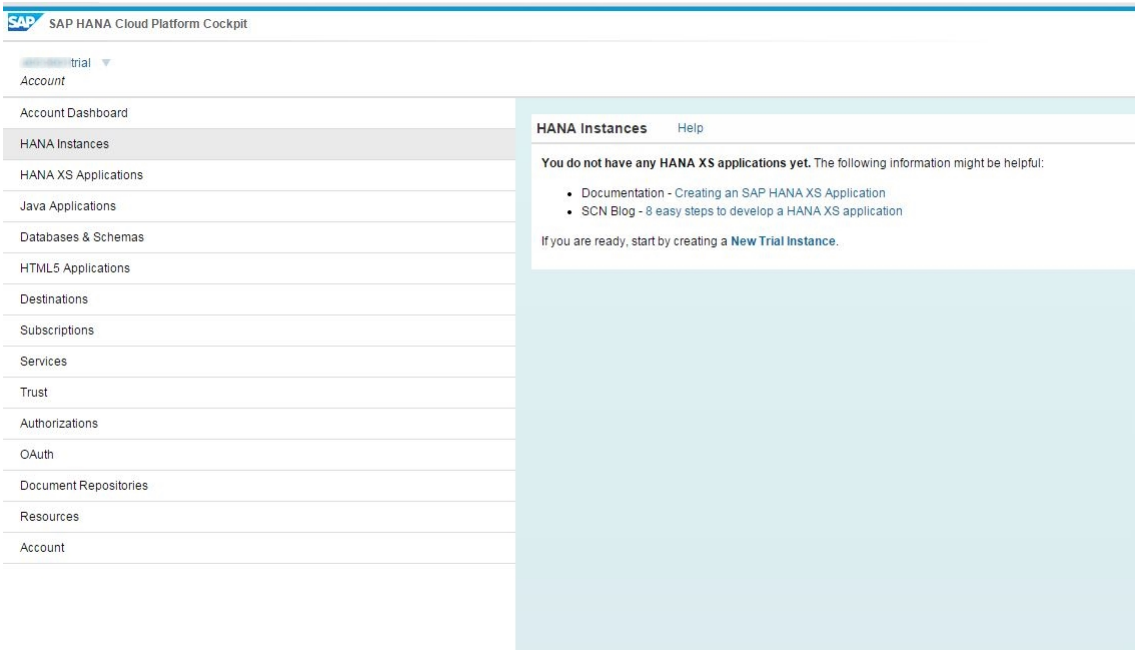

Figure 4.74: Database and Schemas

2. In the example below a HANA XS instance called iotmmsxs 4.75 is created.

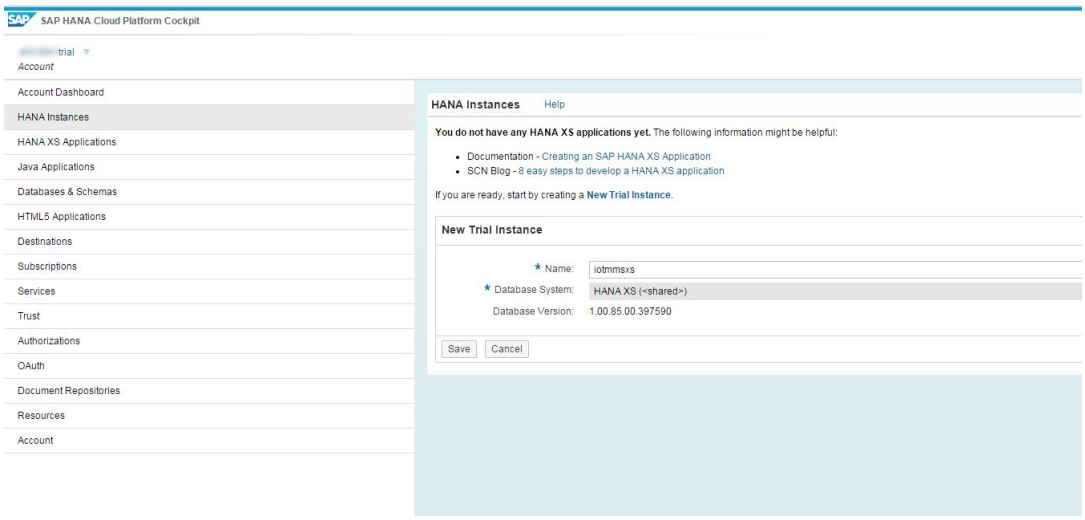

Figure 4.75: Creating iotmmsxs

3. The HANA XS instance will also show up as a new schema 4.76 in the Databases and Schemas section.

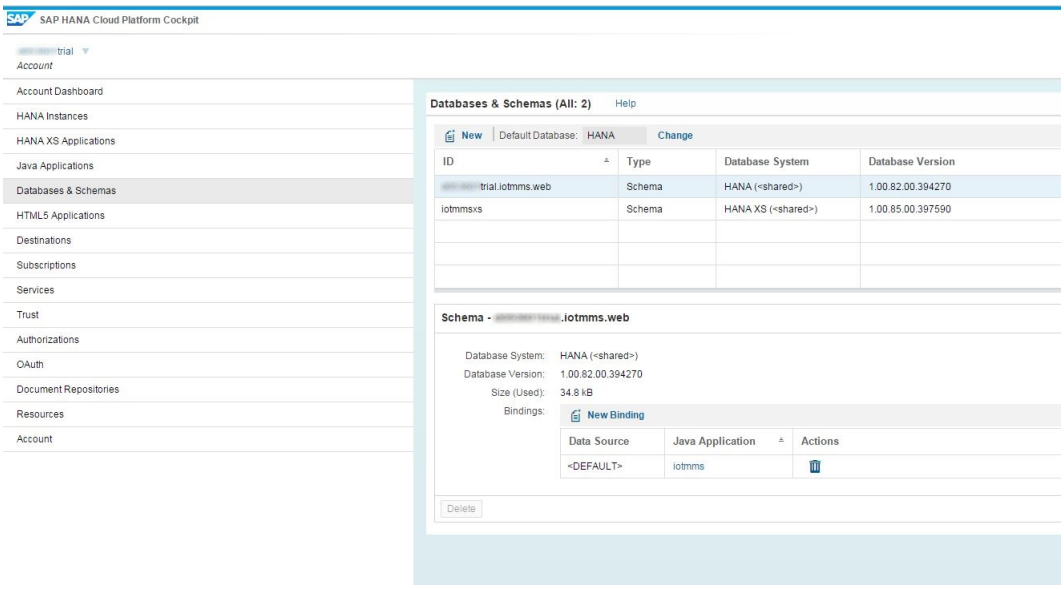

Figure 4.76: HANA XS Database

– Changing the database binding of MMS:

1. In order to switch the MMS database binding from HANA to HANA XS 4.77, select the HANA schema and remove the iotmms binding. After that, select the new iotmmsxs schema and add a new binding to iotmms.

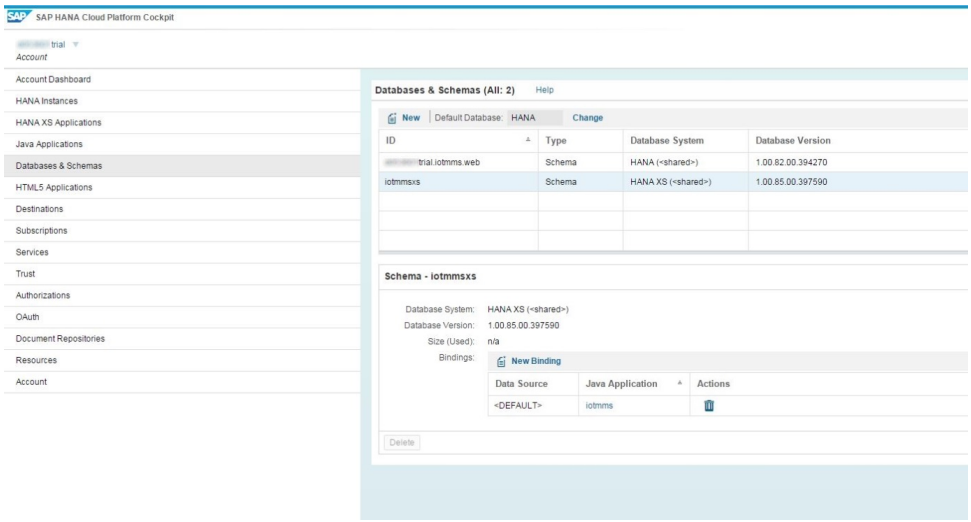

Figure 4.77: Binding IoT service with HANA XS Database

2. After doing that MMS needs to be restarted. As soon as MMS will receive data from devices it will automatically create all tables in the new HANA XS instance. Hence, all the data can be accessed via HANA XS application running in the same HANA instance.

2.3.2 Deployment

– HANA XS Development

1. Now we are ready for XS development. Only the browser based tools will be used for that. Click on Development Tools 4.78 link available for your XS instance.

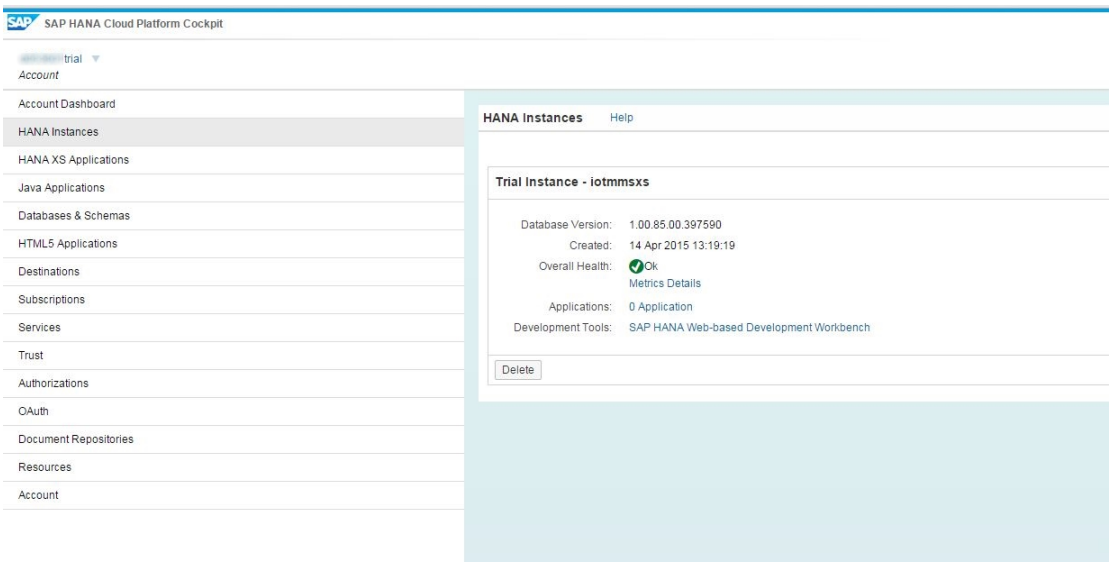

Figure 4.78: Open Development Tool

2. This will open an Editor tool 4.79 for you showing your account specific package.

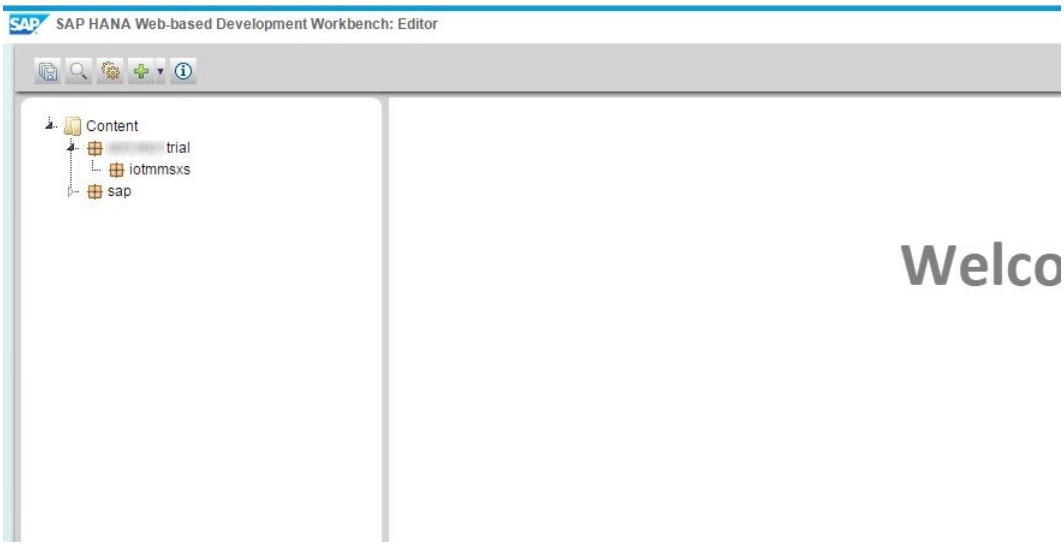

Figure 4.79: Development Tool interface

3. Right away open another web based tool which we will use in the next steps. Click on a black arrow right to green plus and select Catalog' 4.80

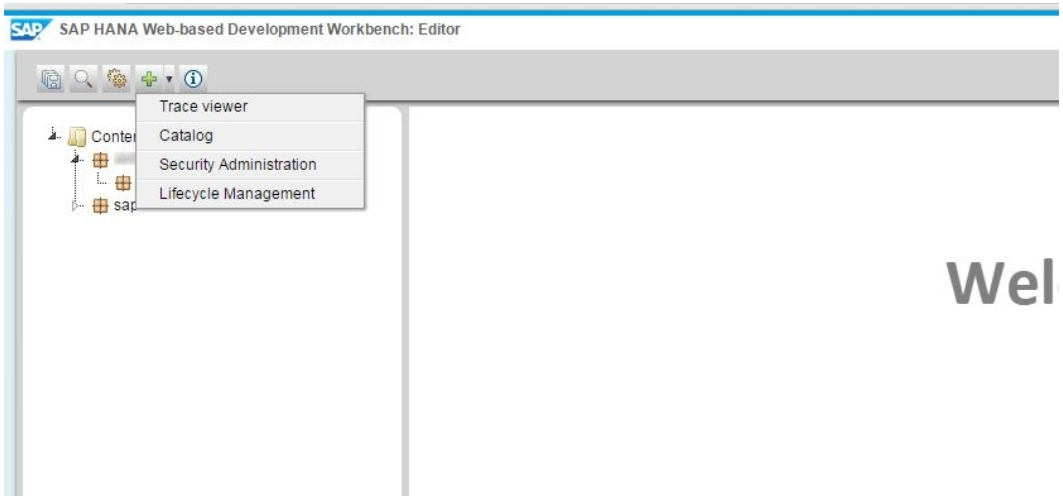

Figure 4.80: Development Tool interface - Catalog function

4. This will open a Catalog tool for you showing your DB schemas 4.81 with their content.

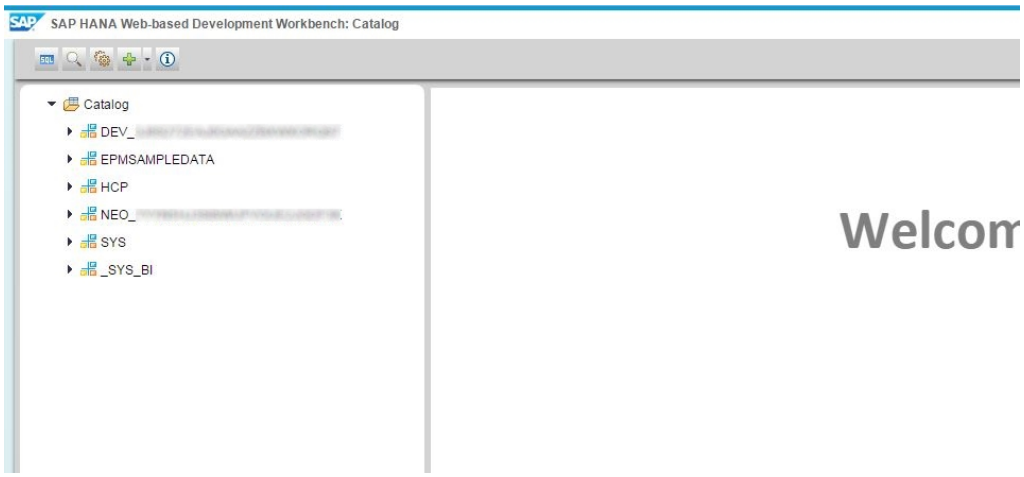

Figure 4.81: Database Schemas

5. Pay your attention that $NEO\_\%$ schema 4.82 is a real name of your (iotmmsxs) schema displayed in your HCP Cockpit. All $T\_IOT\_\%$ tables are landed there.

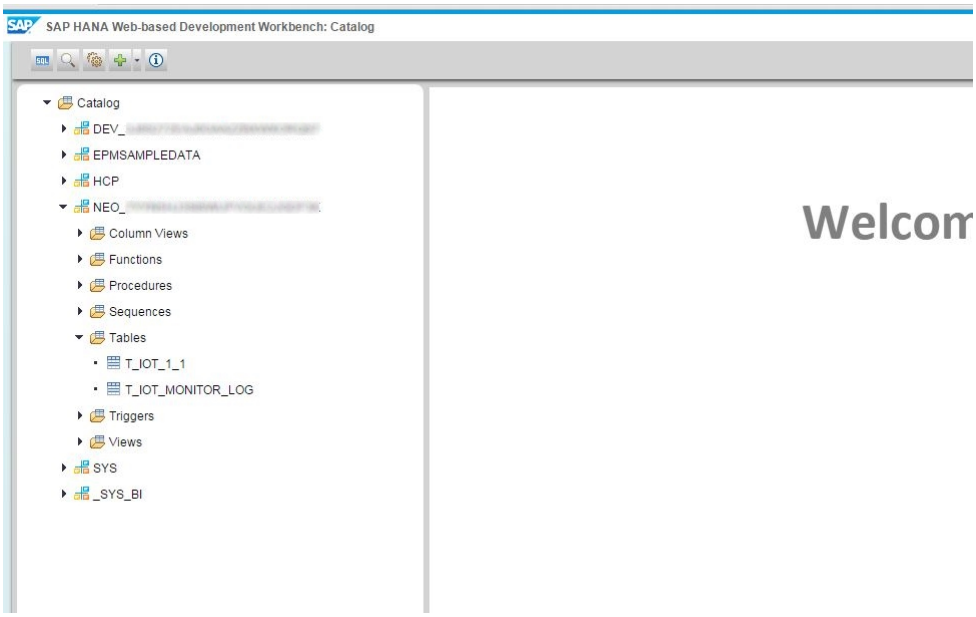

Figure 4.82: Database Schema tables

6. Return back to Editor, right click on (iotmmsxs) and select Create Application 4.83.

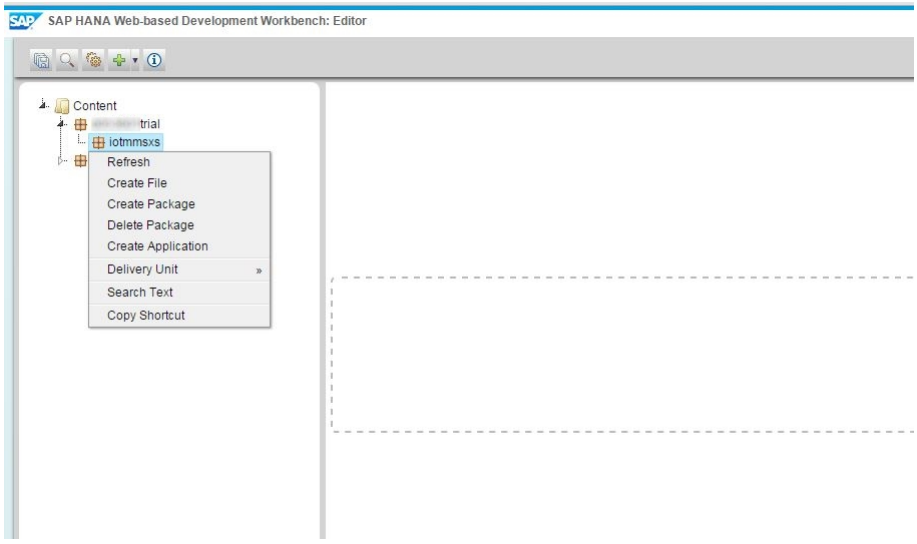

Figure 4.83: Creating an Application

7. Select Create in selected package' option (this is not mandatory) and choose Blank Application' template 4.84. This will create a blank XS application for you.

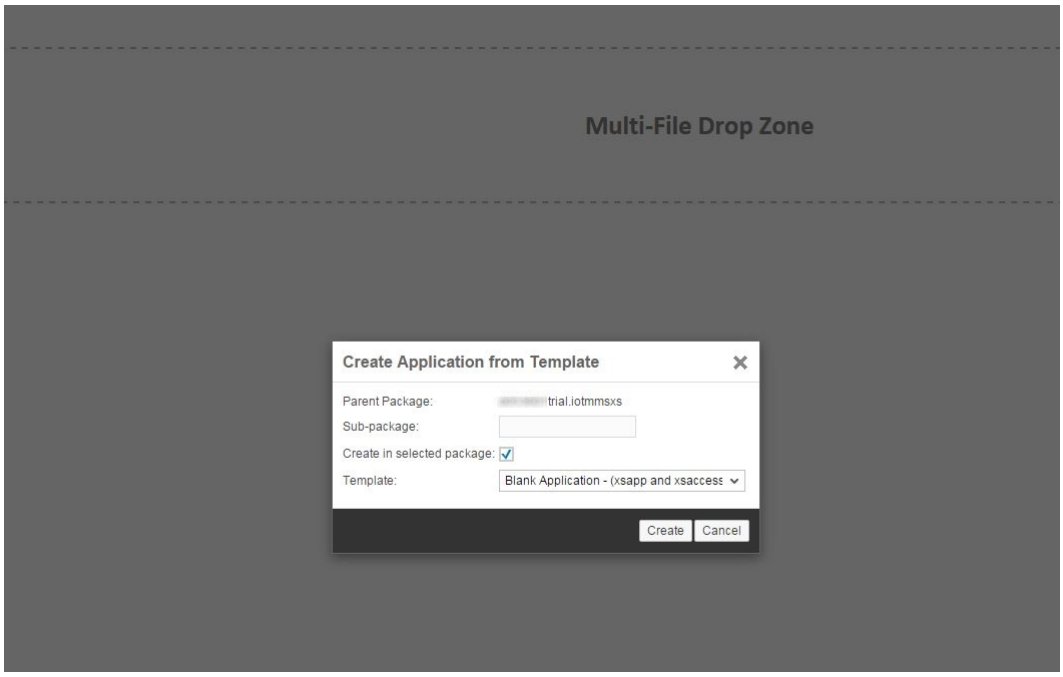

Figure 4.84: Creating Application from Template

8. Create those files. The content of your package should look as follows 4.85. Then, adapt all necessary files and activate them.
   - **.xsaccess**

```
{
        "exposed": true,
        "authentication" :
            [
                { "method" : "Basic" }
            ],
            "cors" :
            {
                "enabled" : true
            }
}

```

   - **.xsapp**

```
{
  {}
```

```
}
```

- **.xsprivileges**

```
{
  "privileges" : [ {
    "name" : "Basic",
    "description" : "Basic IoT MMS privilege"
  } ]
}
```

- **iotaccess.hdbrole**

```
role <user_id>trial.iotmmsxs::iotaccess {
  application privilege: <user_id>trial.iotmmsxs::Basic;
  catalog schema "NEO_<schema_id>": SELECT;
}
```

- **iotservice.xsodata**

```
service {
  "NEO_<schema_id>"."T_IOT_<message_type_id>" key
  generate local "GEN_ID";
}
```

- **iotserviceolddbapi.xsjs**

```
var oConnection = $.db.getConnection();

var sQuery = "SELECT * FROM NEO_<schema_id>.T_IOT_<
  message_type_id>";

var oStatement = oConnection.prepareStatement(sQuery);
```

```
7   var oResultSet = oStatement.executeQuery();

9   var sBody = "";

11  while (oResultSet.next()) {
      sBody += oResultSet.getNString(1) + "\t" + oResultSet.
    getNString(2) + "\t" + oResultSet.getNString(3) + "\t" +
     oResultSet.getNString(4) + "\n";
13  }

15  oResultSet.close();
    oStatement.close();
17
    $.response.status = $.net.http.OK;
19  $.response.setBody(sBody);
```

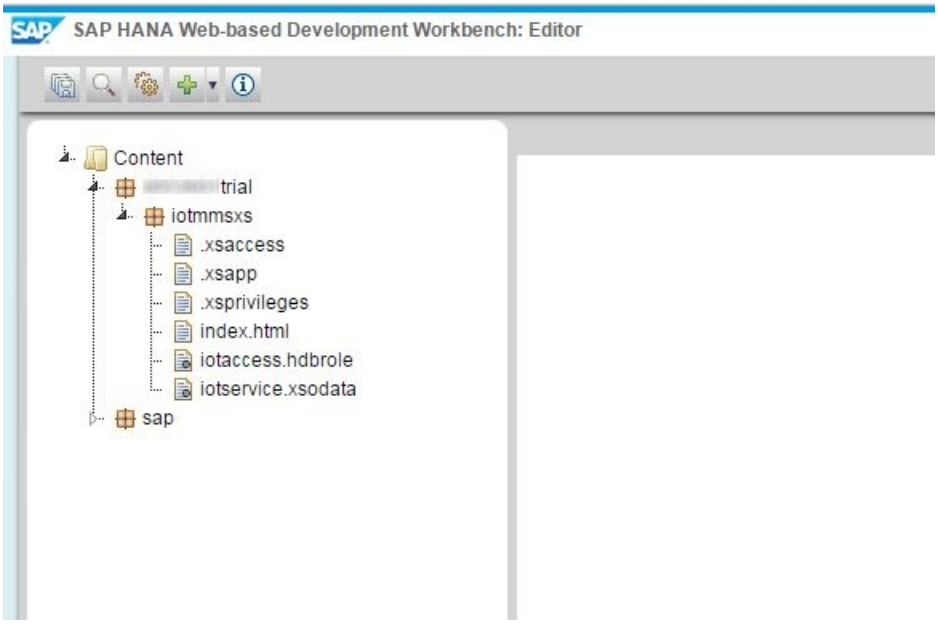

Figure 4.85: Application file list

– Grant Roles

1. Return back to Catalog tool and select a SQL button 4.86 on top.

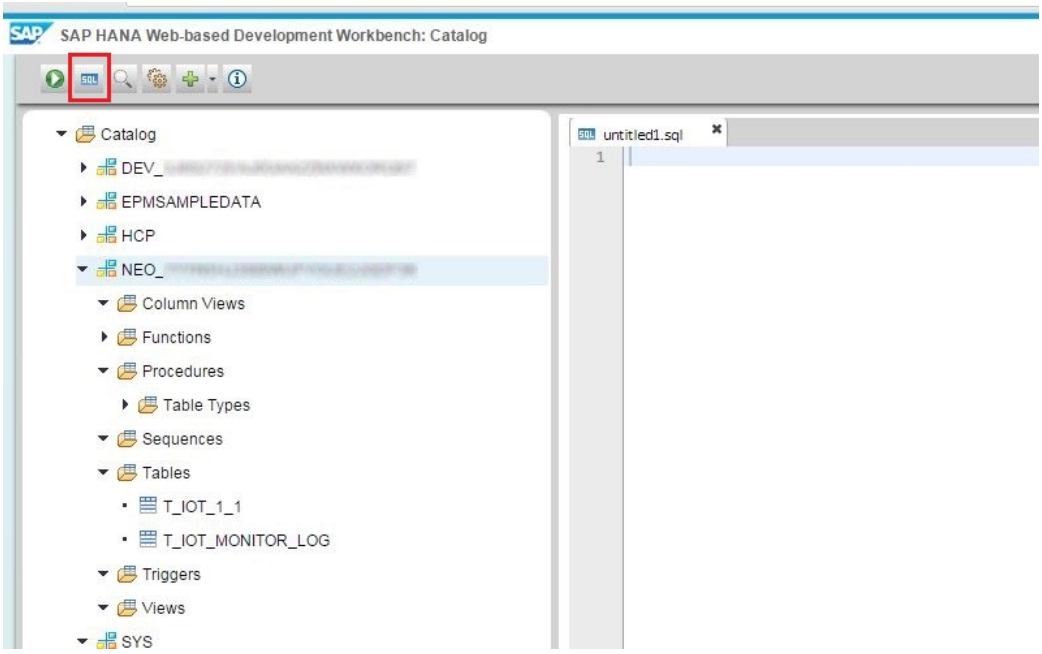

Figure 4.86: Application file list

2. Insert the following SQL script there, adapt it and press green execute button. This should result in success execution.

```
call "HCP"."HCP_GRANT_ROLE_TO_USER"('<user_id>trial.iotmmsxs
  ::iotaccess', '<user_id>');
```

– Start XS application

1. Return back to HCP Cockpit and navigate to HANA XS Application 4.87.

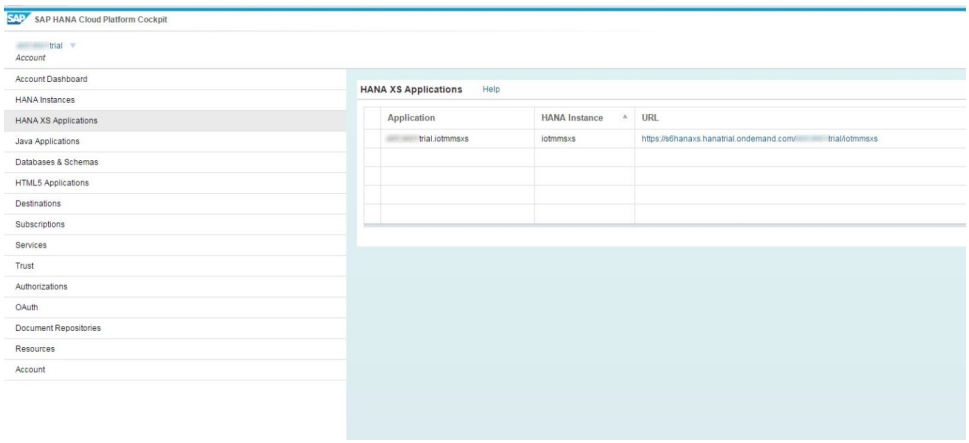

Figure 4.87: HANA XS application link

2. Click on application URL. This will open a default index.html file located in your XS application.
3. Add /iotservice.xsodata at the end of that URL. This will point you to OData service 4.89.
   $https://<system\_id>hanaxs.hanatrial.ondemand.com/$
   $<user\_id>trial/iotmmsxs/iotservice.xsodata$
4. The next link shows OData service metadata 4.91
   $https://<system\_id>hanaxs.hanatrial.ondemand.com/$
   $<user\_id>trial/iotmmsxs/iotservice.xsodata/\$metadata$
5. The next link shows entity content in XML format 4.88
   $https://<system\_id>hanaxs.hanatrial.ondemand.com/<user\_id>$
   $trial/iotmmsxs/iotservice.xsodata/T\_IOT\_<MESSAGE\_TYPE\_ID>$
6. The next link shows entity content in JSON format
   $https://<system\_id>hanaxs.hanatrial.ondemand.com/<user\_id>$
   $trial/iotmmsxs/iotservice.xsodata/T\_IOT\_<MESSAGE\_TYPE\_ID>$

   $?\$format = json$
7. XSJS Service based on old database $.db interface 4.90
   $https://<system\_id>hanaxs.hanatrial.ondemand.com/$
   $<user\_id>trial/iotmmsxs/iotserviceolddbapi.xsjs$

Figure 4.88: REST API OData

Figure 4.89: REST API OData Metadata


<?xml version="1.0" encoding="utf-8" standalone="yes"?><feed xml:base="https://s8hanaxs.hanatrial.ondemand.com:443/i321696trial/iotservices/iotservice.xsodata/" xmlns:d="http://schemas.microsoft.com/ado/2007/08/dataservices"
xmlns:m="http://schemas.microsoft.com/ado/2007/08/dataservices/metadata" xmlns="http://www.w3.org/2005/Atom"><title type="text">T_IOT_A617149678B7E1A669C2</title>
<id>https://s8hanaxs.hanatrial.ondemand.com:443/i321696trial/iotservices/iotservice.xsodata/T_IOT_A617149678B7E1A669C2</id><author><name /></author><link rel="self" title="T_IOT_A617149678B7E1A669C2" href="T_IOT_A617149678B7E1A669C2" /><entry>
<id>https://s8hanaxs.hanatrial.ondemand.com:443/i321696trial/iotservices/iotservice.xsodata/T_IOT_A617149678B7E1A669C2('353768791111036231')</id><title type="text"></title><author><name /></author><link rel="self" title="T_IOT_A617149678B7E1A669C2"
href="T_IOT_A617149678B7E1A669C2('353768791111036231')"/><category term="i321696trial.iotservices.iotservice.T_IOT_A617149678B7E1A669C2Type" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /><content type="application/xml"><m:properties>
<d:GEN_ID m:type="Edm.String">353768791111036231</d:GEN_ID><d:G_DEVICE m:type="Edm.String">809fd3e6-1926-4981-a8ad-281f53a82792</d:G_DEVICE><d:G_CREATED m:type="Edm.DateTime">2015-08-13T09:38:52.9910000</d:G_CREATED><d:C_DATA
m:type="Edm.String">42786f424146454555675152</d:C_DATA><d:C_ID m:type="Edm.String">1BBC9</d:C_ID><d:C_TIMESTAMP m:type="Edm.String">1439458716</d:C_TIMESTAMP></m:properties></content></entry><entry>
<id>https://s8hanaxs.hanatrial.ondemand.com:443/i321696trial/iotservices/iotservice.xsodata/T_IOT_A617149678B7E1A669C2('353768791111036232')</id><title type="text"></title><author><name /></author><link rel="self" title="T_IOT_A617149678B7E1A669C2"
href="T_IOT_A617149678B7E1A669C2('353768791111036232')"/><category term="i321696trial.iotservices.iotservice.T_IOT_A617149678B7E1A669C2Type" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /><content type="application/xml"><m:properties>
<d:GEN_ID m:type="Edm.String">353768791111036232</d:GEN_ID><d:G_DEVICE m:type="Edm.String">809fd3e6-1926-4981-a8ad-281f53a82792</d:G_DEVICE><d:G_CREATED m:type="Edm.DateTime">2015-08-13T09:39:59.9940000</d:G_CREATED><d:C_DATA
m:type="Edm.String">42786f424146454555675152</d:C_DATA><d:C_ID m:type="Edm.String">1BBC9</d:C_ID><d:C_TIMESTAMP m:type="Edm.String">1439458716</d:C_TIMESTAMP></m:properties></content></entry><entry>
<id>https://s8hanaxs.hanatrial.ondemand.com:443/i321696trial/iotservices/iotservice.xsodata/T_IOT_A617149678B7E1A669C2('353768791111036233')</id><title type="text"></title><author><name /></author><link rel="self" title="T_IOT_A617149678B7E1A669C2"
href="T_IOT_A617149678B7E1A669C2('353768791111036233')"/><category term="i321696trial.iotservices.iotservice.T_IOT_A617149678B7E1A669C2Type" scheme="http://schemas.microsoft.com/ado/2007/08/dataservices/scheme" /><content type="application/xml"><m:properties>
<d:GEN_ID m:type="Edm.String">353768791111036233</d:GEN_ID><d:G_DEVICE m:type="Edm.String">809fd3e6-1926-4981-a8ad-281f53a82792</d:G_DEVICE><d:G_CREATED m:type="Edm.DateTime">2015-08-13T09:48:49.1040000</d:G_CREATED><d:C_DATA
m:type="Edm.String">42456f4256776749555675553</d:C_DATA><d:C_ID m:type="Edm.String">1BBC9</d:C_ID><d:C_TIMESTAMP m:type="Edm.String">1439459316</d:C_TIMESTAMP></m:properties></content></entry><entry>

Figure 4.90: REST API OData Table

Figure 4.91: REST API OData Table XML format

The advantages are brought by OData service is undeniable. Other SAP applications can connect, and use directly the data here. With many ways to extract information from the database (reference here).

– **Data Removal:** To interact more with the data. Delete permission is also neccessary. To achieve this, I got an idea from watching this video. Therefore, I use OData again, and add an identify index to the database as *primary key*:

1. Click **Open in Web-based Development Workbench** from SAP HANA Cloud Platform Cockpit 4.92

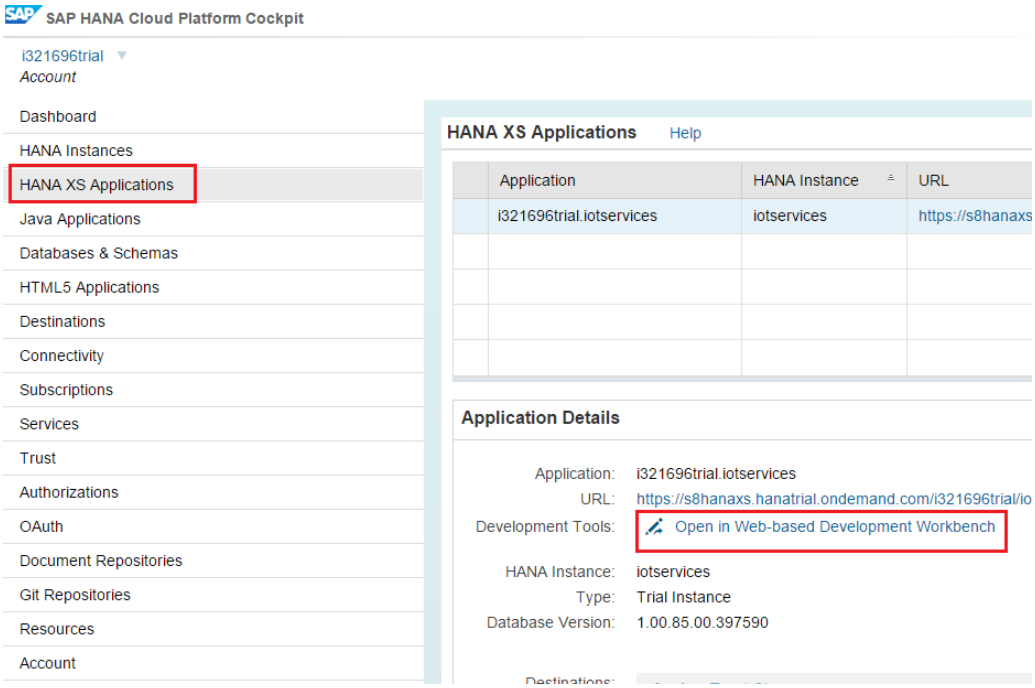

Figure 4.92: Accessing Web-based Development Workbench

2. Access catalog interface 4.80. Then click to SQL button4.86.
3. Drop the old database created by iotmms application:

```
DROP TABLE SCHEMA.TABLE
For example:
DROP TABLE NEO_0BGGKBHFWIAW0PYZAHCB0E89I.
    T_IOT_55BB5AF4ED3641AED580
```

4. Then create the new one with only adding a new Index. The **COLUMN TABLE** is important in this case because the HANA Database only support many features on this COLUMN TABLE.

```
CREATE COLUMN TABLE "NEO_0BGGKBHFWIAW0PYZAHCB0E89I."
    T_IOT_55BB5AF4ED3641AED580" ("G_ID" bigint primary key
    generated by default as identity, "G_DEVICE" VARCHAR(255)
    CS_STRING,
    "G_CREATED" LONGDATE CS_LONGDATE,
    "C_DATA" VARCHAR(255) CS_STRING,
    "C_ID" VARCHAR(255) CS_STRING,
    "C_TIMESTAMP" VARCHAR(255) CS_STRING,
    "C_TYPE" VARCHAR(255) CS_STRING)
```

5. Now everytime a new record is added into the database, the index is automatically created and increased by 1 4.93.

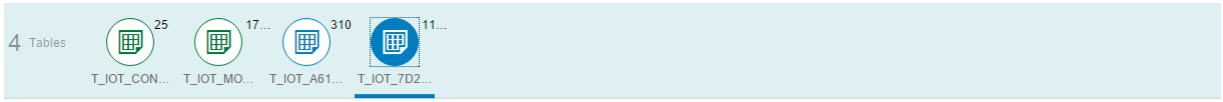

Table [T_IOT_7D277D02623030A23268] contains [116179] entries.

G_ID	G_DEVICE	G_CREATED	C_DATA	C_ID	C_TIMESTAMP	C_TYPE
116181	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:26:12.462	5530706555386b3d	s_soleau	1425281100	Pression_Admission_Soleau
116180	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:26:12.462	41556f444367453d	s_soleau	1425281100	Pression_Refoulement_Soleau
116179	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:26:12.462	565263484269303d	s_ermitage	1425281100	ErmitageDebitDN300
116178	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:26:12.462	4342634a4353593d	s_ermitage	1425281100	ErmitageD_bitDN100
116177	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:25:59.46	466b385354426c4f4668394754784d61515267554742303d	u_loubet	1425280800	CompteurLoubet
116176	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:25:59.46	426863504348303d	s_soleau	1425280800	Pression_Admission_Soleau
116175	809fd3e6-1926-4981-a8ad-281f53a82792	2015-08-27 12:25:59.46	554538435656593d	s_soleau	1425280800	Pression_Refoulement_Soleau

Figure 4.93: Display stored messages

6. Access my Database via OData, and pick a record. For example:

```
https://s8hanaxs.hanatrial.ondemand.com/i321696trial/
    iotservices/iotservice.xsodata/IoT(2)/?$format=xml
```

7. Postman Launcher is used to send to request 4.94. Put the link of record in **DELETE** request, and finally send it. This record will be removed.

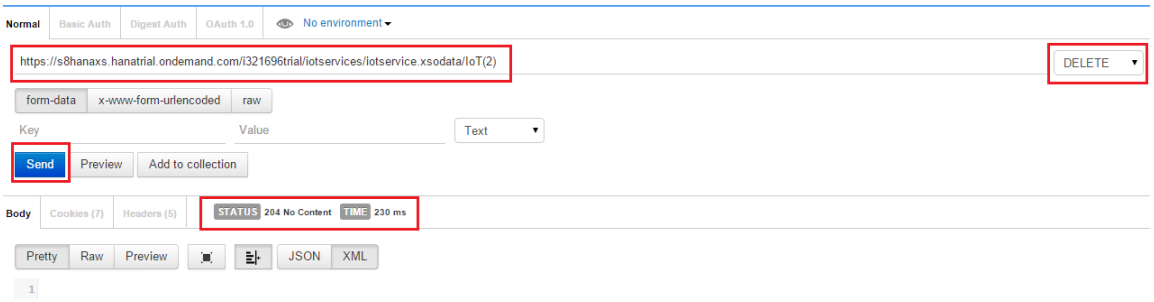

Figure 4.94: Postman Launcher Usage

However, the data is only removed one by one which this technique. Therefore, it takes time to delete the whole database with one million records for example. Another solution is to use HANA xsjs service to do this task. There are also two approaches as follows:

* Using **Procedure** in HANA Database to delete, and recreate the table, which literally removes data. There is a limitation that HANA Database somehow cannot create the same database by using procedure comparing to normal SQL command. If this technique is applied, the database needs to be manually created.

  1. Create the delete procedure from the catalog

```
CREATE PROCEDURE NEO_0BGGKBHFWIAW0PYZAHCB0E89I.DROPDATA()
LANGUAGE SQLSCRIPT
AS
 BEGIN
     DROP TABLE "Your_Schema"."Your_Table";
END

```

  2. Create a xsjs file for calling the procedure with this code

```
var oConnection = $.db.getConnection();
var sQuery = 'CALL Your_Schema.DROPDATA()';
var oStatement = oConnection.prepareCall(sQuery);
var oResultSet = oStatement.execute();
var body="";

oStatement.close();
$.response.status = $.net.http.OK;
$.response.setBody(body);
```

```
10
```

3. Create back your database in the catalog. Open SQL console and type:

```
1  CREATE COLUMN TABLE "Your_Schema"."Your_Table" ( "G_ID"
     bigint primary key generated by default as identity,"
     G_DEVICE" VARCHAR(255) CS_STRING, "G_CREATED" LONGDATE
     CS_LONGDATE, "C_DATA" VARCHAR(255) CS_STRING, "C_ID"
     VARCHAR(255) CS_STRING, "C_TIMESTAMP" VARCHAR(255)
     CS_STRING, "C_TYPE" VARCHAR(255) CS_STRING, "C_UNIT"
     VARCHAR(255) CS_STRING )
```

4. Pay attention with the table name. It should be the same table which is created by the IoT MMS. Otherwise, we cannot push the data to SAP HCP IoT.

* Using TRUNCATE query to empty the database. However, this technique will not reset the primary key as identity numbers.
  1. Create a .xsjs file with this code

```
1  var oConnection = $.db.getConnection();
   var oStatement = oConnection.prepareStatement("TRUNCATE
     TABLE \"Your_Skema\".\"Your_Table\"");
3
   var oResultSet = oStatement.execute();
5  var body="";

7  oStatement.close();
   $.response.status = $.net.http.OK;
9  $.response.setBody(body);
```

  2. Then run this file to clear the database.
  3. The result can be checked through REST API or Iot MMS Message Management Cockpit.

– **Permission Allowance** is important to allow particular users to access the database and have rights to perform on the database. Replace user_id1 by a suitable id, and user_id2 by an account id which would need to access the database

```
call "HCP"."HCP_GRANT_ROLE_TO_USER"('<user_id1>trial.iotmmsxs::
  iotaccess', '<user_id2>');
for example:

call "HCP"."HCP_GRANT_ROLE_TO_USER"('I321696trial.iotservices::
  iotaccess', 'P1941603003')

```

– **User Experience**

1. Change the table with new represented name for convenience

```
service {
  "NEO_<schema_id>"."T_IOT_<message_type_id>" as "IoT";
}

```

From now on we can access with new link

```
https://s8hanaxs.hanatrial.ondemand.com/i321696trial/
    iotservices/iotservice.xsodata/IoT/
```

2. We can also access a specific record we prefer too by using a specific link

```
https://s8hanaxs.hanatrial.ondemand.com/i321696trial/
    iotservices/iotservice.xsodata/IoT(1)
```

3. Or even access an attribute of the record

```
https://s8hanaxs.hanatrial.ondemand.com/i321696trial/
    iotservices/iotservice.xsodata/IoT(1)/C_DATA
```

– **Exposing Database:**

As can be seen, there are two ways to expose the database. The first approach is to use OData service, and the second one is to use XSJS Service based on old database $.db interface. With the second approach, format of the REST API could be modified to improve client requirements. This piece of code below illustrates the approach.

```
var oConnection = $.db.getConnection();

//Change your suitable Data tables
//Get data from the IoT Service
var sQuery = "SELECT * FROM NEO_0BGGKBHFWIAW0PYZAHCB0E89I.
    T_IOT_10BD82E37593AD36CD3D";
//Get geolocation
var sQueryC = "SELECT * FROM NEO_0BGGKBHFWIAW0PYZAHCB0E89I.
    T_IOT_BB7F2789479BF02C2799";

var oStatement = oConnection.prepareStatement(sQuery);
var oStatementC = oConnection.prepareStatement(sQueryC);

var oResultSet = oStatement.executeQuery();
var oResultSetC = oStatementC.executeQuery();

var sBodyC = [];

//Get the ID and its coordinate as 2D array
while (oResultSetC.next()) {
    var sComponent = [];

    sComponent.push(oResultSetC.getNString(4));
    sComponent.push(oResultSetC.getNString(5));

    sBodyC.push(sComponent);
}

var temp="";

var sBody = "{";

while (oResultSet.next()) {
```

```
    var id = oResultSet.getNString(5);
33  sBody += "\"message\":{\"id\":\""+ id + "\"," +"\"data\":\"" +
    oResultSet.getNString(4)+ "\"," + "\"type\":\"" + oResultSet.
    getNString(7) + "\"," + "\"unit\":\"" + oResultSet.getNString
    (8) + "\"," + "\"coordinate\":[";

35  var coordinate="43.612116, 7.017207";

37  for(var i =0; i < sBodyC.length ;i++)
    {
39
        if (id.toUpperCase().localeCompare(sBodyC[i][0].
    toUpperCase())===0 )
41      {
            coordinate = sBodyC[i][1];
43
        }
45  }

47  sBody += coordinate;

49  sBody += "]},";
}
51
//Delete the last ,
53 sBody = sBody.substring(0, sBody.length - 1);
sBody += "}"
55
oResultSet.close();
57 oStatement.close();
oResultSetC.close();
59 oStatementC.close();

61 $.response.status = $.net.http.OK;
$.response.setBody(sBody);
```

The process from this code could be understood:

1. Get data of two tables from the database.
2. Do the mapping between two tables to get the data.

3. Display the result as json format 4.95

{"id":"a36","data":"42314e564767525379673d3d","type":"IndexJournalier2","unit":"m3","coordinate":[7.084550, 43.602347]},"message":
{"id":"a36","data":"56786342564a493d","type":"DebitMoyenJourMin1","unit":"m3/h","coordinate":[7.084550, 43.602347]},"message":{"i
{"id":"a36","data":"55514144537767482b513d3d","type":"Compteur2","unit":"m3","coordinate":[7.084550, 43.602347]},"message":{"id":
{"id":"a36","data":"4142594356523O3d","type":"DebitMoyen2","unit":"m3/h","coordinate":[7.084550, 43.602347]},"message":{"id":"a36"
{"id":"a36","data":"4155785856416f3d","type":"DebitMoyen1","unit":"m3/h","coordinate":[7.084550, 43.602347]},"message":{"id":"a36"
{"id":"a36","data":"55523842352513d","type":"DebitMoyenJourMin2","unit":"m3/h","coordinate":[7.084550, 43.602347]},"message":{"
{"id":"a36","data":"566b77455572673d","type":"DebitMoyen1","unit":"m3/h","coordinate":[7.084550, 43.602347]},"message":{"id":"a05"
{"id":"a05","data":"424146564251565742683849426543d","type":"Compteur1","unit":"m3","coordinate":[7.117420, 43.580622]},"messa
{"id":"a05","data":"557830414343493d","type":"DebitMoyen2","unit":"m3/h","coordinate":[7.117420, 43.580622]},"message":{"id":"a05"
{"id":"a05","data":"55774d6642314658","type":"DebitMoyenJourMax2","unit":"m3/h","coordinate":[7.117420, 43.580622]},"message":{"
{"id":"a05","data":"43516c5157426c574247513d","type":"IndexJournalier2","unit":"m3","coordinate":[7.117420, 43.580622]},"message":
{"id":"a05","data":"434263445665453d","type":"DebitMoyenJourMin1","unit":"m3/h","coordinate":[7.117420, 43.580622]},"message":{"
{"id":"a05","data":"43316f6355564e7a","type":"NiveauDeReceptionGSM","unit":"unite","coordinate":[7.117420, 43.580622]},"message":
{"id":"a05","data":"437745414330315342614l3d","type":"CapaciteElectrique","unit":"unite","coordinate":[7.117420, 43.580622]},"messa
{"id":"a05","data":"42316f4f57457742564a673d","type":"NbTotalSMS_Emis","unit":"unite","coordinate":[7.117420, 43.580622]},"messa
{"id":"a05","data":"5577564e41416434","type":"DebitMoyenJourMax1","unit":"m3/h","coordinate":[7.117420, 43.580622]},"message":{"
{"id":"a26","data":"436756534777634746673d3d","type":"Compteur1","unit":"m3","coordinate":[7.091822, 43.602597]},"message":{"id":
{"id":"a26","data":"44466b505531384854517453547l3d3d","type":"Compteur2","unit":"m3","coordinate":[7.091822, 43.602597]},"messa
{"id":"a26","data":"426873424179413d","type":"VolumeJour1","unit":"m3","coordinate":[7.091822, 43.602597]},"message":{"id":"a26","
{"id":"a26","data":"424151484741494454673d3d","type":"IndexJournalier1","unit":"m3","coordinate":[7.091822, 43.602597]},"message":
{"id":"a26","data":"565567464135553d","type":"VolumeJour2","unit":"m3","coordinate":[7.091822, 43.602597]},"message":{"id":"a26","
{"id":"a26","data":"42527052434c383d","type":"Pression","unit":"bars","coordinate":[7.091822, 43.602597]},"message":{"id":"a26","data
{"id":"a26","data":"4277415741514971","type":"NiveauDeReceptionGSM","unit":"unite","coordinate":[7.091822, 43.602597]},"message"
{"id":"a26","data":"41773041554235554345d4d3d","type":"CapaciteElectrique","unit":"unite","coordinate":[7.091822, 43.602597]},"messa
{"id":"a26","data":"41674d4255687053434d513d","type":"NbTotalSMS_Emis","unit":"unite","coordinate":[7.091822, 43.602597]},"messa
{"id":"a26","data":"42426b434130453d","type":"DebitMoyenJourMax1","unit":"m3/h","coordinate":[7.091822, 43.602597]},"message":{"

Figure 4.95: XSJS Service based on old database

In addition, more features for REST API with xsjs is developed following this guide. In specific, select information by ID feature. To achieve this, arguments are caught, and pharsed to perform the request, and display the result.

```
//Change your suitable Data tables
var id = $.request.parameters.get('ID');
//If the ID, change the query to retreive information from the DB
if (id)
{
    var sQuery = "SELECT * FROM NEO_0BGGKBHFWIAW0PYZAHCB0E89I.
    T_IOT_AB1D68560D53A408FEDD WHERE C_ID = '"+id+"'";
}

//Access link
https://<Your_Link>?ID=<Id_Name>
```

3 Integration

3.1 Related work

The question is how to merge all services such as applications, REST APIs as a platform. The answer is Web IDE Platform [36]. Web IDE is used to:

- Build the main page as known as cockpit to connect, and manage all services with HTML and I5.
- Integrate FreeDataMap into SAP Hana Cloud Platform.

3.1.1 Problem

HANA trial consuming in SAP Web IDE via HCP destination service: **not yet**

- What cannot do is there is not permissions to access HANA XS as public service for trial accounts, but it could be done with productive accounts.
- Every time a request is sent to the REST API, the authentication would redirect the request to another authentication page.

3.1.2 Approaches

Accessing service via **Node JS** with **SAP Cloud Connector** and **Destinations** in **SAP Web IDE**: **yes**

- Use Node Js to get all data from REST API stored on a local machine (virtual machine).
- Use SAP Cloud Connector to setup the connect with HANA Trial account and the local machine via Destination configuration.
- Modify the resources in WebIDE.
- Use correct queries to retrieve REST API information.

3.2 Deployment

3.2.1 Preparation

- Web IDE subscription
  1. Login to your SAP HANA Cloud Platform Cockpit.
  2. Hit **Subscriptions**, and then choose **Web IDE**.

3. Click **Application URL** to starts SAP Web IDE.
4. You will have an interface which is similar in figure 4.96.

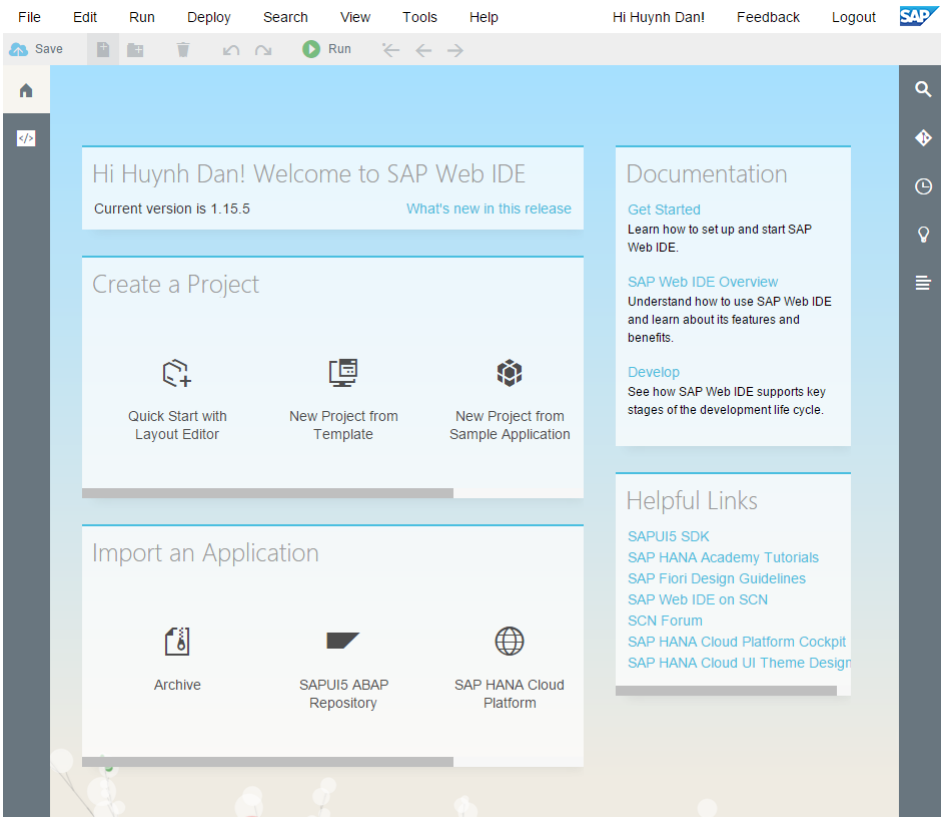

Figure 4.96: Web IDE interface

– Cloud Connector Installation
Guide Videos:
Set up SAP HANA Cloud Connector
SAP Cloud Connector - Resources
SAP Cloud Connector - Destinations

1. Install Java SE Development Kit 7, and set up environment variable.
2. Get SAP Cloud Connector - Developer version here
3. Extract the zip file, and simply click to the file **go.bat** 4.97 to run the program

Figure 4.97: Cloud Connector Installation guide 1

4. Open a web browser, type **http://localhost:8443** to reach the login page 4.98

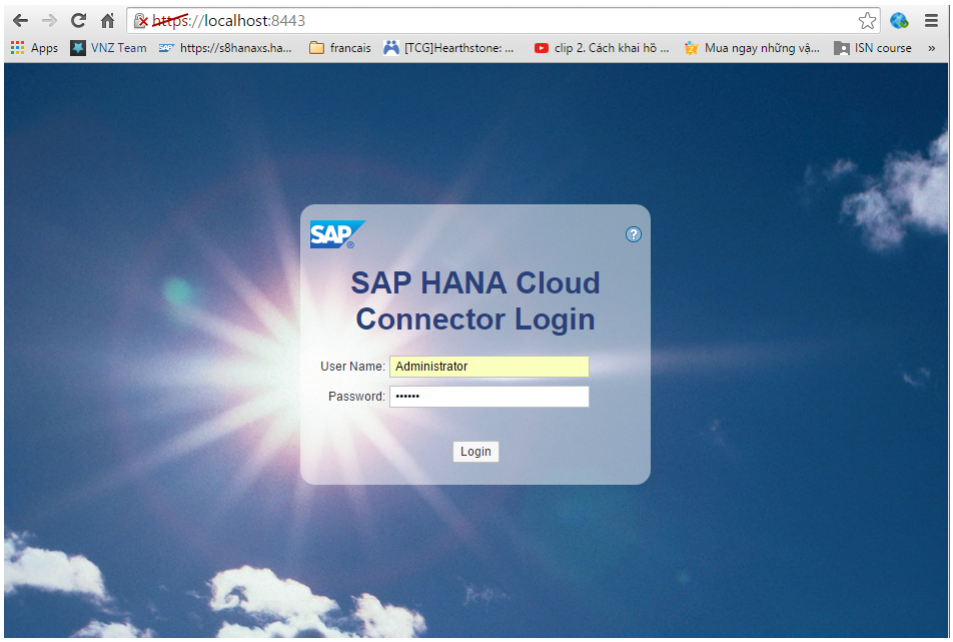

Figure 4.98: Cloud Connector Login interface

5. The ID is **Administrator** and Password is **Manage**. Hit login and change the password then.
6. Choose **Master**(Primary Installation) 4.99

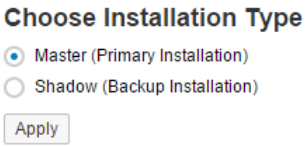

Figure 4.99: Cloud Connector installation type

7. Set-up initial configuration to connect with a trial account 4.100. Remember to change account id and add proxy if behind a company network.

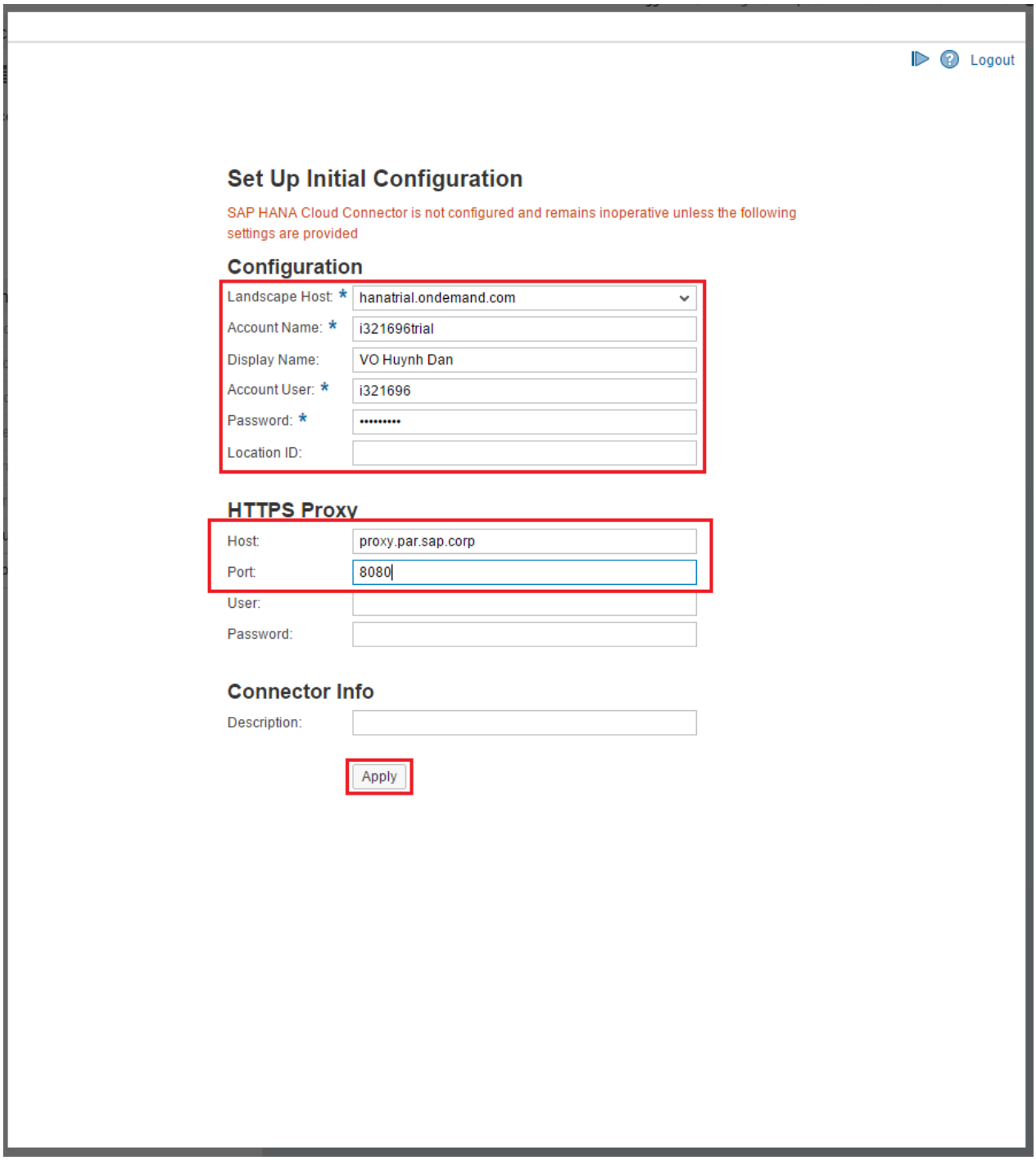

Figure 4.100: Cloud Connector initial configuration

8. Reach the interface in figure 4.101. The Connector is orange and Landscape Host should be green. However, do not worry because we need to finish some other minor configurations.

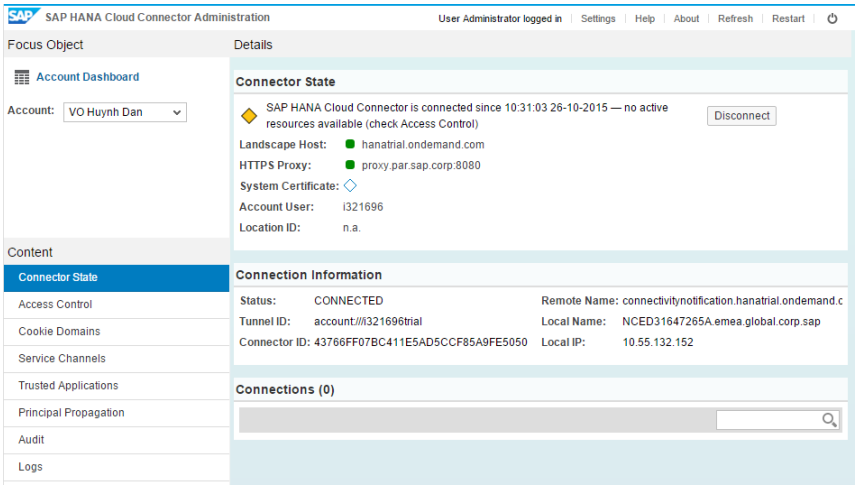

Figure 4.101: Cloud Connector interface

9. Hit **Access Control** on the left panel menu. Next, click **Add** from **Mapping Virtual To Internal System**. Fill information like in figure 4.102.

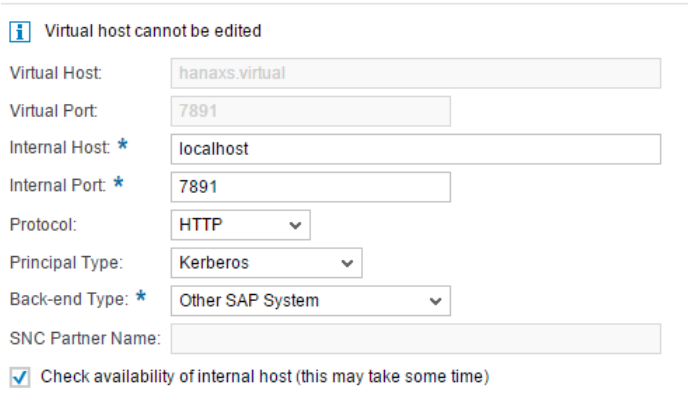

Figure 4.102: Cloud Connector Mapping Virtual To Internal System Set-up

10. Hit **Add** from **Resources Accessible On localhost:7891** Panel. and fill up information 4.103. Notice that what should be typed in **URL Path** is the link of your package containing REST API files 4.104

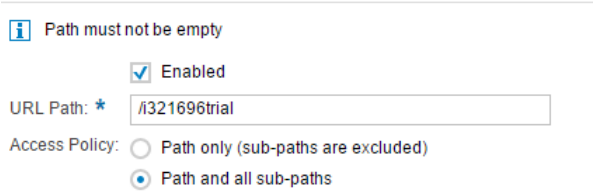

Figure 4.103: Cloud Connector Resources Accessible Set-up 1

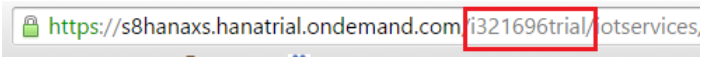

Figure 4.104: Cloud Connector Resources Accessible Set-up 2

11. Go back to **Connector State**, everything should be green now.
12. Setting up Destinations in **SAP HANA Cloud Platform Cockpit** to enable to connections between Web IDE and the local machine.
13. Hit **Destinations** in **SAP HANA Cloud Platform Cockpit**
14. Click **Add**, and fill information in figure 4.105. Then click save to establish the destination.

Figure 4.105: Cloud Connector Destination Set-up

– Node JS Installation
  1. Download a suitable version and install it.
  2. Setup proxy configuration if behind a proxy network, by opening a **cmd** console and type the follow code:

```
npm config set registry "http://registry.npmjs.org/"
npm config set proxy http://<proxy_page>:<port_number>
```

– Authentication Proxy Server Installation
  1. Download the application from this github .

2. Extract it and use **Node JS** to build it.
3. Modify the file **config.js** in *examples* folder.

```
host: 's2hanaxs.hanatrial.ondemand.com'

to host: 's8hanaxs.hanatrial.ondemand.com'
(The version your REST API is running on SAP HCP)
```

4. Change the proxy if behind a company network.
5. Open **cmd** and cd to root folder, then simply run those code lines described in the Github:

```
npm install

//To the run server
node examples\server-basic-auth.js
```

6. After installing, it should similar to what is shown in figure 4.106

```
C:\Users\i321696\Desktop\hanatrial-auth-proxy-master>npm install
npm WARN package.json hanatrial-auth-proxy@0.2.0 No license field.
request@2.64.0 node_modules\request
├── aws-sign2@0.5.0
├── forever-agent@0.6.1
├── stringstream@0.0.4
├── oauth-sign@0.8.0
├── caseless@0.11.0
├── tunnel-agent@0.4.1
├── isstream@0.1.2
├── json-stringify-safe@5.0.1
├── extend@3.0.0
├── qs@5.1.0
├── node-uuid@1.4.3
├── tough-cookie@2.0.0
├── mime-types@2.1.7 (mime-db@1.19.0)
├── form-data@1.0.0-rc3 (async@1.4.2)
├── http-signature@0.11.0 (assert-plus@0.1.5, ctype@0.5.3, asn1@0.1.11)
├── combined-stream@1.0.5 (delayed-stream@1.0.0)
├── bl@1.0.0 (readable-stream@2.0.2)
├── hawk@3.1.0 (cryptiles@2.0.5, boom@2.9.0, sntp@1.0.9, hoek@2.16.3)
└── har-validator@1.8.0 (bluebird@2.10.1, commander@2.8.1, chalk@1.1.1, is-my-js
on-valid@2.12.2)

cheerio@0.19.0 node_modules\cheerio
```

Figure 4.106: Authentication proxy server with Node js installation

7. After running program, it should similar to the result in figure 4.107

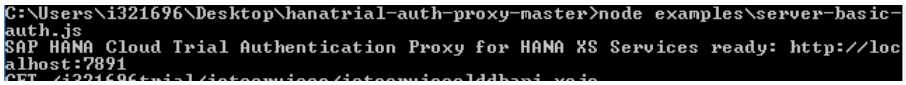

Figure 4.107: Authentication proxy server running interface

8. For the testing part, type this link in a web browser to get the data from the REST API.

```
http://localhost:7891/<your_link_to_the_REST_API>
<your_link_to_the_REST_API> : Should be what after <https://
  s8hanaxs.hanatrial.ondemand.com/>

//For example
http://localhost:7891/i321696trial/iotservices/
  iotserviceolddbapi.xsjs

```

3.2.2 Secure Predictive Maintenance @ Antibes Cockpit

The Cockpit is built on by using SAP UI5 in order to connect all the services, applications together in one interface. There will be FreeDataMap, a Java graph app , MMS Management, REST API pages, and Database Removal.

– Interface: Created by using UI5 API with HTML and Javascript mainly 4.108

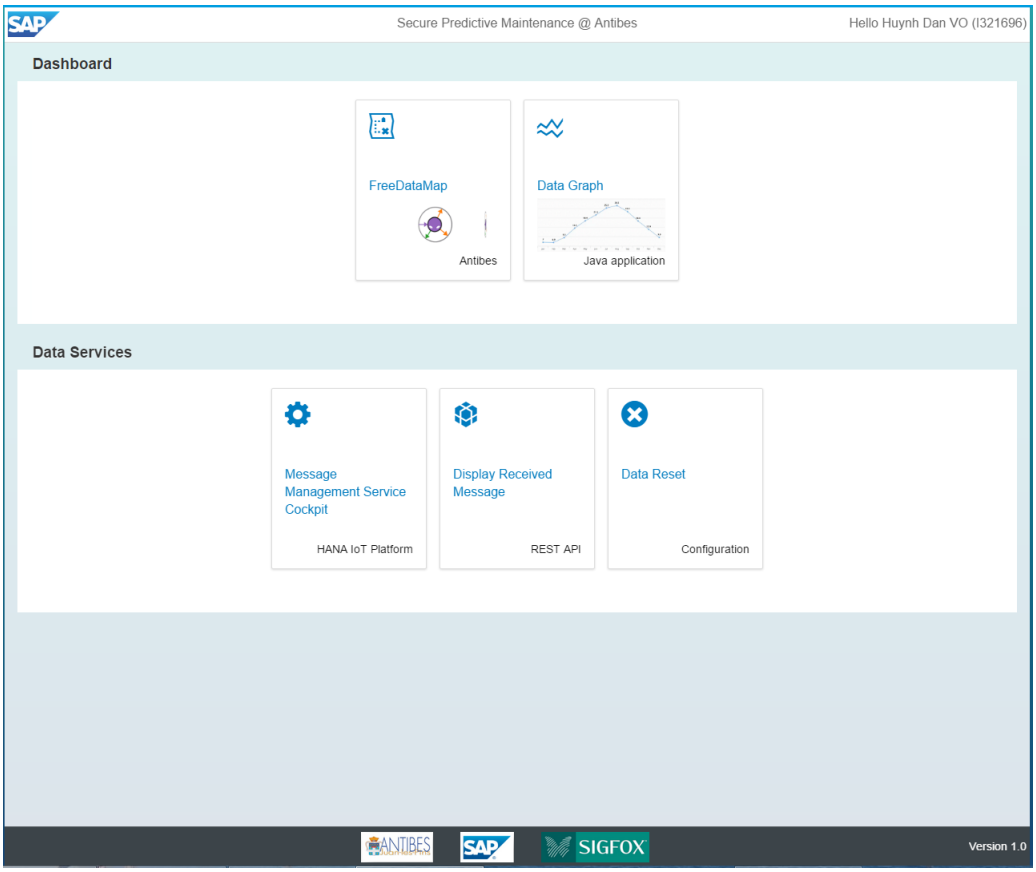

Figure 4.108: The cockpit page

1. FreeDataMap tile represents for FreeDataMap application integrated to SAP HCP.
2. Data Graph tile connects to an application plotting graph from collected data.
3. Manage Management Service Cockpit leads to the cockpit page for MMS IoT Application.
4. Display received Message tile will open the REST API pages.
5. Data Removal tile will empty the database.

– Integrate directly FreeDataMap on SAP Web IDE, and run it

– Configure system, and call correct queries to retrieve information from the REST API,

3.2.3 FreeDataMap

FreeDataMap [29] is integrated as an internal webpage along with the cockpit. Since the author of FreeDataMap runs, builds, and tests it on Apache HTTP Server 2.2.25,

SAP HCP is totally suitable to integrate FreeDataMap with Web IDE.

– Integration of FreeDataMap

1. Change name of the file **index.html** to **CopyOfindex.html** of FreeDataMap application, and zip them all as a Zip file 4.109.

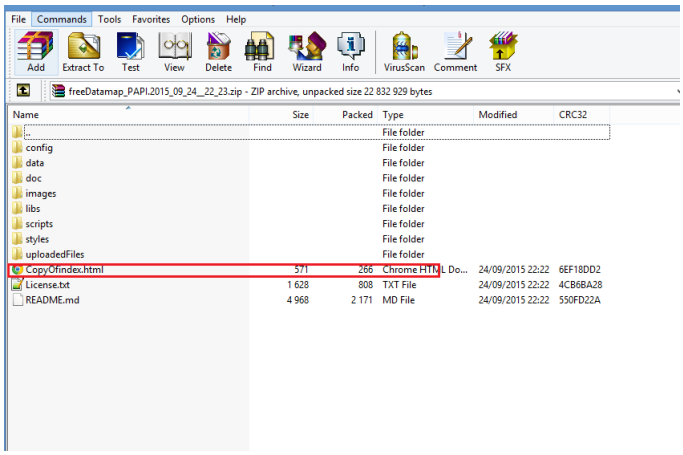

Figure 4.109: FreeDataMap zip file

2. Then we import this file to the folder containing the cockpit we built above by hitting right click to the folder, choose import, and file system then 4.110.

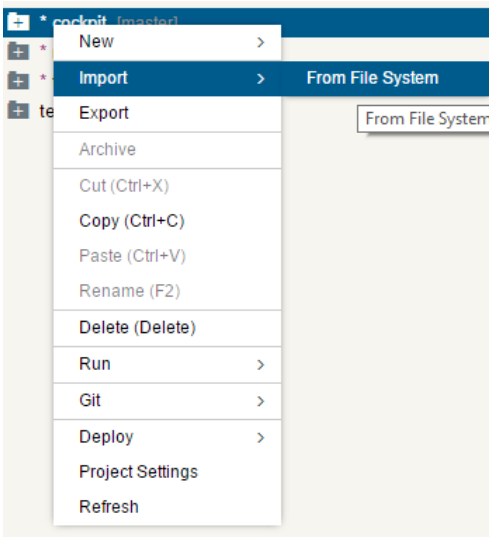

Figure 4.110: FreeDataMap file import

3. Add this piece of code to **routes** of **neo-app.json** file to activate REST API services from the authentication proxy server built locally.

```
{
    "path": "/destinations/hanaxs",
    "target": {
      "type": "destination",
      "name": "hanaxs"
    },
    "description": "hanaxs"
  }
```

4. To test whether it will be able to access the REST API or not, use the link to point directly to the REST API. It will show everything similarly to what you receive from the REST API.

```
https://<your_app_name><your_trial_account>.dispatcher.
    hanatrial.ondemand.com/destinations/hanaxs/<
    link_of_your_REST_API_service>

For example:
https://cockpit-i321696trial.dispatcher.hanatrial.ondemand.com/
    destinations/hanaxs/i321696trial/iotservices/
    iotserviceolddbapi.xsjs
```

3.2.4 Offline Cockpit and FreeDataMap importation

This is a complete package containing the cockpit and FreeDataMap integrated on SAP Web IDE.

1. Open **Web IDE**.
2. **Right click**, choose **Import**, then **From File System**.
3. Choose zip file **cockpit_nonIBM.zip** and extract this zip file.
4. Test, and delpoy the web application.

In this section, we finally have finished a configuration to ensure the connectivity throughout different platforms such as PLCs, Cloud Platforms. Configured SAP HCP services to guarantee data from PLCs follow a proper direction, and it is

received, exposed correctly on SAP HCP. The deployment of gathered security requirements, the Proof of Concept of end to end scheme from devices to platforms, and integration of a third party UI into SAP HCP.

1 Conclusion

To sum up, we have presented, and implemented the cryptographic scheme to protect the data. The scheme which overcomes the limitation of message transmission is applied in C/C++, Python, and Javascript languages. In addition, we also successfully implemented the Proof of Concept. We are able to collect the data, encrypt, store, and display it on SAP HCP Dashboard. Furthermore, we also created a Github to replicate the Proof of Concept.

As future work, we intend to focus on another requirement which is data integration. We have a very first step for key management system by using the cryptoshield, and re-keying feature via SIGFOX downlink callbacks. A set of keys could be used and changed frequently on devices to improve the security. Moreover, it is also very interesting if we integrate hardware security on IoT devices to use secure storage, and secure hash algorithms.

I truly believe I improved not only my programming skills such as Python, JavaScript, but also many other techniques incluing code debugging and analysis after this internship . Working in an professonal open space, and participarting in a real project gives me such a good experience, and it helps me a lot in developing my soft skills as well as future study, and career.

2 Schedule

Total: 26 weeks

- Ramp-up/Education: 3 weeks
- Modification of the EAC for microcontrollers: 2 weeks

- Implementation on microcontrollers: 3 weeks
- Integration between SIGFOX and SAP HCP: 3 weeks
- SAP HCP configuration, and application integration : 5 weeks
- Attack models : 3 weeks
- Building libraries, and implementation on Intel Edison : 1 week
- Cryptoshield usage, and SIGFOX downlink callbacks : 2 weeks
- Documentation: 4 weeks

APPENDICES

1 Arduino

1.1 Cryptographic library

– EAC.h

```
/*
This file is part of cryptographic scheme
*/

#ifndef _EAC_H
#define _EAC_H

class EAC{
  public:
    //Constructor - format EAC(ID,key,keylength,al)
    EAC(String, uint8_t*, int , char*);

    //Get the ID value
    String getID();

    //Set ID with a new ID
    void setID(String);

    //Encryption function - Need value and sequence number as
    inputs
    char* encrypt(float value, int seq);

```

```
    //rekey function - in progress
    boolean reKey(int rekeyIndex);

  private:
    //ID of the device
    String id;
    //a 64-root key
    uint8_t* eac_key;
    //Key length
    int eac_key_length;
    //Authorization level value
    char * eac_al;

    //One time pad encrypt function
    char* CXOR(const char* const ,const char* const );

    //This function belongs to HMAC_SHA256
    String getHashString(const uint8_t*);

    //This function belongs to HMAC_SHA256
    void printHash(uint8_t*);

    //Add padding to the string
    String addPadding(String);
};

#endif // _EAC_H
```

– EAC.cpp

```
#include <Arduino.h>
#include <SoftwareSerial.h>  //Built-in library
#include <stdlib.h> //C library
#include "sha256.h" //HMAC SHA256 library
#include "base64.h" //Base64 encryption/decryption library
#include "EAC.h"

//Message length is equal to 9
const int message_len = 9;
```

```
10
/***************
12 Constructor for EAC class - required variables:
     1. ID of the device
14   2. a 64 bit root key
     3. key length
16   4. authorization level

18 For example:

20      String id="1BBC9"; //ID of Device, you can change your ID
       device here, replace "74136"
        char * const eac_al = "0";
22      uint8_t eac_key[] ={
        0xAa,0x0A,0xa1,0xAa,0x2A,0xa3,0xAa,0x4A,0xa5,0xAa,0x6A,0xa7,0
       xAa,0x8A,0xa9,0xAb,0x0A,0xb1,0xAb,0x2A,
24      0xb3,0xAb,0x4A,0xb5,0xAb,0x6A,0xb7,0xAb,0x8A,0xb9,0xAc,0x0A,0
       xc1,0xAc,0x2A,0xc3,0xAc,0x4A,0xc5,0xAc,
        0x6A,0xc7,0xAc,0x8A,0xc9,0xAd,0x0A,0xd1,0xAd,0x2A,0xd3,0xAd,0
       x4A,0xd5,0xAd,0x6A,0xd7,0xAd,0x8A,0xd9,
26      0xAe,0x0A,0xe1,0xAe};
          int key_len = sizeof(eac_key);
28

    EAC C(id,eac_key,key_len, eac_al);
30 ***************/

32 EAC::EAC(String _id, uint8_t* _key,int _eac_key, char* _eac_al)
   {
34   this->id = _id;
     this->eac_key = _key;
36   this->eac_key_length = _eac_key;
     this->eac_al=_eac_al;
38 }

40 //Getter - Get the current ID value
   String EAC::getID()
42 {
     return this->id;
44 }
   //Setter - Replace value of ID by a new value _id
```

```
46
void EAC::setID(String _id)
48 {
  this->id = _id;
50
}
52
/***************
54 One time pad encryption function

56 Do the XOR function with two char* c1, and char* c2
Receive a new char* c as a result
58
***************/
60 char* EAC::CXOR(const char* const c1,const char* const c2)
{
62   char* c = strdup(c1);
  int i=0;
64   int len=strlen(c1);

66   for (i=0;i<len;i++)
  {
68     c[i] = c1[i] ^ c2[i];
  }
70   return c;
}
72
//Get hash value of a string
74 String EAC::getHashString(const uint8_t* hash)
{
76   int i;
  String str;
78   for (i=0; i<32; i++) {
    str += ("0123456789abcdef"[hash[i]>>4]);
80     str += ("0123456789abcdef"[hash[i]&0xf]);
  }
82   return str;
}
84
//Print hash value as readable characters after using
```

```
    getHashString function
void EAC::printHash(uint8_t* hash)
{
  int i;
  for (i=0; i<32; i++) {
    Serial.print("0123456789abcdef"[hash[i]>>4]);
    Serial.print("0123456789abcdef"[hash[i]&0xf]);
  }
  Serial.println();
}

/***************
//Add padding 0 to the 8-byte message
For example:
  S = 10.10
  X = 11111111

  After the function -> S = 10.10000
              X = 11111111
***************/

String EAC::addPadding(String s)
{
  String res= s;
  int s_len = s.length();
  while(s_len<=7)
  {
    res = res+"0";
    s_len++;
  }

  return res;
}

/***************
Encryption function - Do the encryption follow the cryptographic
    scheme

Inputs:
  1. value
```

```
124   2. sequence number

126 output is a encrypted string (char*) with base64 encoding

128 For example :
    EAC C;
130   C.encrypt(100.00,1)

132 ***************/
    char* EAC::encrypt(float value, int seq)
134 {
    //Create a variable to store the encryption value
136   char en[20];

138   //Add padding to the value
    String padded_litters = addPadding(String(value));
140
    //Convert the value to char*
142   const char* sensorvalue = padded_litters.c_str();

144   //Start the encryption from here
    //EAC init
146   String result;
    int key_len = this->eac_key_length;
148
    //Prepare to do the first HMAC SHA256
150   //Init sha256 hmac function with Key and keylength
    Sha256.initHmac(this->eac_key,key_len);
152
    //Init sha256 hmac function with al value
154   Sha256.print(this->eac_al);

156   //Get the result from hmac sha256 function
    uint8_t* hash = Sha256.resultHmac();
158
    //Get the K1 as a string
160   String eac = getHashString(hash);

162   //Prepare to do the second HMAC SHA256
    //Convert K1 to char*
```

```
164  const char *eac_c = eac.c_str();

166  //Concat ID and sequence number together
     String s="";
168  s = s + (String)this->id + (String)seq;

170  //Init sha256 hmac function with K1, and its keylength
     Sha256.initHmac(hash,32);
172
     //Init sha256 hmac function with ID|seq value
174  Sha256.print(s);

176  //Get the result from hmac sha256 function - K2
     hash = Sha256.resultHmac();
178
     //Get K2 as a string
180  String eac_onetimekeyString = getHashString(hash);

182  //Convert K2 to char*
     const char *eac_onetimekey_c = eac_onetimekeyString.c_str();
184
     //Execute the One-time-Pad function
186  char* const xor1 = CXOR(sensorvalue,eac_onetimekey_c);

188  //Create a variable - temp as an input for base64 encoding
     char temp[10] = "000000000";
190  int i;
     for(i=0; i < 8;i++)
192  {
       temp[i] = xor1[i];
194  }

196  //Declare a new variable equal to sequence number
     int p = seq;
198
     //Add sequence number to the last byte
200  temp[8] = (char)p;

202  //Perform base64 encoding with 'en' as an output
     base64_encode(en, temp, message_len);
```

```
204
  //Free the memory for the one-time-pad
206  free(xor1);

208  //Return the encrypted message - ciphertext
  return en;
210 }
```

– **toSIGFOXLib.ino** is an example to demonstrate the library.

```
1 /*************************************************************
  This is an example to show how to use the EAC library
3
  1. Get value as a random number less than 8 characters
5 2. Encrypt the data
  3. Send the data to SIGFOX
7
  *************************************************************/
9
  #include <SoftwareSerial.h>  //Built-in library
11 #include <Akene.h> //SIGFOX Library
  #include <stdlib.h> //C library
13 #include "EAC.h" //Include the cryptography library

15 #include <MemoryFree.h>

17 const int DELAY=1000;
  int seq=0;
19 //const int message_len = 9; //Do not change this
  String id="1BBC9"; //ID of Device, you can change your ID device
     here, replace "74136"
21 char * const eac_al = "0";

23 //a 64-bit root key
  uint8_t eac_key[] ={put your key here};
25
  float getVoltage(int pin){
27  return (analogRead(pin) * .004882814); //converting from a 0 to
     1023 digital range
```

```
                                                    // to 0 to 5 volts (each 1
    reading equals ~ 5 millivolts
}

void setup() {
Serial.begin(9600);
// Wait 3 second for the modem to warm
  delay(3000);
  Serial.println("Start SIGFOX modem");
  Akene.begin();
  delay(15000);
}

typedef struct {
  String temp;
} Payload;

void loop() // run over and over again
{
  //Get the random value - Can replace by the value of sensor
  //This value has to be maximum 8 numbers  including .00
  //For example 11111.00 = 8 characters
  float temperature = random(0,99999);

  //Get key length of the root key
  int key_len = sizeof(eac_key);

  //Constructor with id, eac key, eac al, and key length
  EAC C(id,eac_key,key_len, eac_al);

  char* en =  C.encrypt(temperature,seq);

  //Serial.println(eac);
  Serial.print("Sending to SIGFOX: ");
  Serial.println((String)en);

  seq++;
  //Reset seq to 0 if it's larger than 256 - One byte stores value
    from 0 -> 255
  if(seq==256)
```

```
    seq=0;

  //Prepare to use Akene library and send message to SIGFOX
  Payload j;
  j.temp=en;
 // Wait for availability of the modem
  while (!Akene.isReady()) {
  Serial.println("Modem not ready");
    delay(1000);
  }
  //Send in the mighty internet!
   Akene.send(en,12);
  //Wait for 10 minutes to send the next message to SIGFOX

  delay(600000);
}
```

1.2 Implementation

– Sending data from Arduino to Raspberry Pi

1. **Arduino_RasPi_HCP_Temperature.ino**

```
/*************************************************************
This is an example code for getting temperature sensor data
    from Arduino
and data is sent to Raspberry Pi

1. Arduino collects data
2. Encrypt the data
3. Send to Raspberry Pi

*************************************************************/

#include <SoftwareSerial.h>  //Built-in library
#include <Akene.h> //SIGFOX Library
#include <stdlib.h> //C library
#include "sha256.h" //HMAC SHA256 library
#include "base64.h" //Base64 encryption/decryption library
```

```
17 const int DELAY=1000;
   int seq=0;
19 const int message_len = 9; //Do not change this
   const char id[6]="74136"; //ID of Device, you can change your
      ID device here, replace "74136"
21 const char type[12]="Temperature"; //Replace your type
   const char unit[2]="C"; //Replace your unit
23 char * const eac_al = "0";

25 //a 64-bit root key
   uint8_t eac_key[] ={put your key here};
27
   //XOR function - One time pad
29 char* CXOR(const char* const c1,const char* const c2)
   {
31   char* c = strdup(c1);
     int i=0;
33   int len=strlen(c1);

35   for (i=0;i<len;i++)
     {
37     c[i] = c1[i] ^ c2[i];
     }
39   return c;
   }
41
   String getHashString(const uint8_t* hash) {
43   int i;
     String str;
45   for (i=0; i<32; i++) {
       str += ("0123456789abcdef"[hash[i]>>4]);
47     str += ("0123456789abcdef"[hash[i]&0xf]);
     }
49   return str;
   }
51
   void printHash(uint8_t* hash) {
53   int i;
     for (i=0; i<32; i++) {
```

```
55      Serial.print("0123456789abcdef"[hash[i]>>4]);
        Serial.print("0123456789abcdef"[hash[i]&0xf]);
57    }
      Serial.println();
59  }

61  //Add 0 numbers at the end of a string
    String addPadding(String s)
63  {
      String res= s;
65    int s_len = s.length();
      while(s_len<=7)
67    {
        res = res+"0";
69      s_len++;
      }
71
      return res;
73  }

75  //TMP36 Pin Variables
    int temperaturePin = 0; //the analog pin the TMP36's Vout (
        sense) pin is connected to
77                          //the resolution is 10 mV / degree
        centigrade
                            //(500 mV offset) to make negative
        temperatures an option
79


81  void setup() {
    Serial.begin(9600);
83  }

85  // int is 16 bits, float is 32 bits. All little endian
    /*
87   * getVoltage() - returns the voltage on the analog input
        defined by
     * pin
89   */
    float getVoltage(int pin){
```

```
91  return (analogRead(pin) * .004882814); //converting from a 0
      to 1023 digital range
                                          // to 0 to 5 volts (
      each 1 reading equals ~ 5 millivolts
93 }

95 //Encryption function
   void Encryption(float temperature, char en[])
97 {

99   String padded_litters = addPadding(String(temperature));
     const char* sensorvalue = padded_litters.c_str();
101
     //Start the encryption from here
103
     //EAC init
105  String result;
     int key_len = sizeof(eac_key);
107
     //Key and keylength
109  Sha256.initHmac(eac_key,key_len);
     Sha256.print(eac_al);
111  uint8_t* hash = Sha256.resultHmac();
     //The returned result is a key pointer hash - point to a a 32
       uint8_t array
113  String eac = getHashString(hash);

115  const char *eac_c = eac.c_str();

117  String s="";
     s = s + (String)id + (String)seq;
119
     Sha256.initHmac(hash,32);
121  Sha256.print(s);
     hash = Sha256.resultHmac();
123  String eac_onetimekeyString = getHashString(hash);

125  const char *eac_onetimekey_c = eac_onetimekeyString.c_str();

127  char* const xor1 = CXOR(sensorvalue,eac_onetimekey_c);
```

```
129   char temp[10] = "000000000";
      int i;
131   for(i=0; i < 8;i++)
      {
133     temp[i] = xor1[i];
      }
135   int p = seq;

137   temp[8] = (char)p;

139   base64_encode(en, temp, message_len);

141   free(xor1);

143 }

145 void loop() // run over and over again
    {
147  float temperature = getVoltage(temperaturePin);  //getting the
        voltage reading from the temperature sensor
     temperature = ((temperature - .5) * 100)*1.8+32;          //
       converting from 10 mv per degree wit 500 mV offset
149                                                    //to degrees
       ((volatge - 500mV) times 100)
      char en[20];
151   //Encrypted data - temperature value
      Encryption(temperature,en);
153
      //Start to send data to Ras Pi
155   Serial.print("\"id\":\"");
      Serial.print(id);
157   Serial.print("\",\"data\":\"");
      int j;
159    for(j=0; j < strlen(en);j++)
      {
161     Serial.print(en[j],HEX);
      }
163   Serial.print("\",\"type\":\"");
      Serial.print(type);
```

```
165   Serial.print("\",\"unit\":\"");
      Serial.print(unit);
167   Serial.println("\"");

169   //Reset seq to 0 if it's larger than 256 - One byte stores
        value from 0 -> 255
      seq++;
171   if(seq==256)
        seq=0;
173
      //Wait for 10 seconds to send the next message to RasPi
175   delay(10000);
}
```

2. **Arduino_RasPi_HCP_DebitMeter.ino**

```
/************************************************************
2 This is an example code for getting debit meter data from
      Arduino
  and data is sent to Raspberry Pi
4
  1. Arduino collects data
6 2. Encrypt the data
  3. Send to Raspberry Pi
8
  ************************************************************/
10
  #include <SoftwareSerial.h> //Built-in library
12 #include <Akene.h> //SIGFOX Library
  #include <stdlib.h> //C library
14 #include "sha256.h" //HMAC SHA256 library
  #include "base64.h" //Base64 encryption/decryption library
16
  const int DELAY=1000;
18 int seq=0;
  const int message_len = 9; //Do not change this
20 const char id[6]="1BBC9"; //ID of Device, you can change your
      ID device here, replace "1BBC9"
  const char type[11]="DebitMoyen"; //Replace your type
22 const char unit[5]="m3/h"; //Replace your unit
```

```
char * const eac_al = "0";

//a 64-bit root key
uint8_t eac_key[] ={put your key here};

//XOR function - One time pad
char* CXOR(const char* const c1,const char* const c2)
{
  char* c = strdup(c1);
  int i=0;
  int len=strlen(c1);

  for (i=0;i<len;i++)
  {
    c[i] = c1[i] ^ c2[i];
  }
  return c;
}

String getHashString(const uint8_t* hash) {
  int i;
  String str;
  for (i=0; i<32; i++) {
    str += ("0123456789abcdef"[hash[i]>>4]);
    str += ("0123456789abcdef"[hash[i]&0xf]);
  }
  return str;
}

void printHash(uint8_t* hash) {
  int i;
  for (i=0; i<32; i++) {
    Serial.print("0123456789abcdef"[hash[i]>>4]);
    Serial.print("0123456789abcdef"[hash[i]&0xf]);
  }
  Serial.println();
}

//Add 0 numbers at the end of a string
String addPadding(String s)
```

```
{
  String res= s;
  int s_len = s.length();
  while(s_len<=7)
  {
    res = res+"0";
    s_len++;
  }

  return res;
}

// which pin to use for reading the sensor? can use any pin!
#define FLOWSENSORPIN 10
// count how many pulses!
volatile uint16_t pulses = 0;
// track the state of the pulse pin
volatile uint8_t lastflowpinstate;
// you can try to keep time of how long it is between pulses
volatile uint32_t lastflowratetimer = 0;
// and use that to calculate a flow rate
volatile float flowrate;
// Interrupt is called once a millisecond, looks for any pulses
    from the sensor!
SIGNAL(TIMER0_COMPA_vect) {
uint8_t x = digitalRead(FLOWSENSORPIN);
if (x == lastflowpinstate) {
lastflowratetimer++;
return; // nothing changed!
}
if (x == HIGH) {
//low to high transition!
pulses++;
}
lastflowpinstate = x;
flowrate = 1000.0;
flowrate /= lastflowratetimer; // in hertz
lastflowratetimer = 0;
}
void useInterrupt(boolean v) {
```

```
102 if (v) {
// Timer0 is already used for millis() - we'll just interrupt
    somewhere
104 // in the middle and call the "Compare A" function above
OCR0A = 0xAF;
106 TIMSK0 |= _BV(OCIE0A);
} else {
108 // do not call the interrupt function COMPA anymore
TIMSK0 &= ~_BV(OCIE0A);
110 }
}
112
void setup() {
114 Serial.begin(9600);

116 //Serial.print("Flow sensor test!");
pinMode(FLOWSENSORPIN, INPUT);
118 digitalWrite(FLOWSENSORPIN, HIGH);
lastflowpinstate = digitalRead(FLOWSENSORPIN);
120 useInterrupt(true);

122 }

124 // int is 16 bits, float is 32 bits. All little endian

126 void loop() // run over and over again
{
128
float liters = pulses;
130 liters /= 7.5;
liters /= 60.0;
132
//liters contains value collected from debit meter sensor
134 String padded_litters = addPadding(String(liters));
const char* sensorvalue = padded_litters.c_str();
136
  //Start the encryption from here
138
  //EAC init
140  String result;
```

```
  int key_len = sizeof(eac_key);
142
  //Key and keylength
144 Sha256.initHmac(eac_key,key_len);
  Sha256.print(eac_al);
146 uint8_t* hash = Sha256.resultHmac();
  //The returned result is a key pointer hash - point to a a 32
    uint8_t array
148 String eac = getHashString(hash);

150 const char *eac_c = eac.c_str();

152 String s="";
  s = s + (String)id + (String)seq;
154
  Sha256.initHmac(hash,32);
156 Sha256.print(s);
  hash = Sha256.resultHmac();
158 String eac_onetimekeyString = getHashString(hash);

160 const char *eac_onetimekey_c = eac_onetimekeyString.c_str();

162 char* const xor1 = CXOR(sensorvalue,eac_onetimekey_c);

164 char temp[10] = "000000000";
  int i;
166 for(i=0; i < 8;i++)
  {
168   temp[i] = xor1[i];
  }
170 int p = seq;

172 temp[8] = (char)p;
  char en[20];
174 base64_encode(en, temp, message_len);
  //Start to send data to Ras Pi
176 Serial.print("\"id\":\"");
  Serial.print(id);
178 Serial.print("\",\"data\":\"");
  int j;
```

```
180     for(j=0; j < strlen(en);j++)
    {
182       Serial.print(en[j],HEX);
    }
184   Serial.print("\",\"type\":\"");
    Serial.print(type);
186   Serial.print("\",\"unit\":\"");
    Serial.print(unit);
188   Serial.println("\"");

190   free(xor1);
    //Reset seq to 0 if it's larger than 256 - One byte stores
      value from 0 -> 255
192   seq++;
    if(seq==256)
194     seq=0;

196   //End of the encryption
    //Wait for 10 seconds to send the next message
198   delay(10000);
}
```

– Sending data from Arduino to SIGFOX

1. **Arduino_Sigfox_Temperature.ino**

```
1 /*************************************************************
This is an example code for getting temperature data from
    Arduino
3 and data is sent to SIGFOX

5 1. Arduino collects data
2. Encrypt the data
7 3. Send to SIGFOX

9 *************************************************************/

11 #include <SoftwareSerial.h> //Arduino built-in library
#include <Akene.h> //Library for pushing data to SIGFOX -
    Change it if you are using Areku and other devices
```

```
13 #include <stdlib.h> //This library is used to free the memory
   #include "sha256.h" //sha256.h library
15 #include "base64.h" //base 64 en/de library

17 //HMAC SHA256 Implementation - https://github.com/Cathedrow/
       Cryptosuite

19 const int DELAY=5000;
   int seq=0;
21 const int message_len = 9; //Do not change this - Size of the
       message
   const char id[6]="74136"; //ID of Device, you can change your
       ID device here, replace "74136"
23 char * const eac_al = "0"; //Authorization level

25 //a 64-bit root key
   uint8_t eac_key[] ={put your key here};
27
   //XOR function - One time pad
29 char* CXOR(const char* const c1,const char* const c2)
   {
31   char* c = strdup(c1);
     int i=0;
33   int len=strlen(c1);

35   for (i=0;i<len;i++)
     {
37     c[i] = c1[i] ^ c2[i];
     }
39   return c;
   }
41
   //This function belongs to HMAC_SHA256
43 String getHashString(const uint8_t* hash) {
     int i;
45   String str;
     for (i=0; i<32; i++) {
47     str += ("0123456789abcdef"[hash[i]>>4]);
       str += ("0123456789abcdef"[hash[i]&0xf]);
49   }
```

```
  return str;
}

//This function belongs to HMAC_SHA256
void printHash(uint8_t* hash) {
  int i;
  for (i=0; i<32; i++) {
    Serial.print("0123456789abcdef"[hash[i]>>4]);
    Serial.print("0123456789abcdef"[hash[i]&0xf]);
  }
  Serial.println();
}

//Add padding 0 to the 8-byte message
String addPadding(String s)
{
  String res= s;
  int s_len = s.length();
  while(s_len<=7)
  {
    res = res+"0";
    s_len++;
  }

  return res;
}

//TMP36 Pin Variables
int temperaturePin = 0; //the analog pin the TMP36's Vout (
    sense) pin is connected to
                        //the resolution is 10 mV / degree
    centigrade
                        //(500 mV offset) to make negative
    temperatures an option
void setup() {
  Serial.begin(9600);
  // Wait 3 second for the modem to warm
  delay(3000);
  Serial.println("Start SIGFOX modem");
  Akene.begin();
```

```
87    delay(30000);
}
89
// int is 16 bits, float is 32 bits. All little endian
91 typedef struct {
    String temp;
93 } Payload;

95 // int is 16 bits, float is 32 bits. All little endian
/*
97  * getVoltage() - returns the voltage on the analog input
      defined by
  * pin
99  */
float getVoltage(int pin){
101  return (analogRead(pin) * .004882814); //converting from a 0
      to 1023 digital range
                                          // to 0 to 5 volts (
      each 1 reading equals ~ 5 millivolts
103 }

105 void Encryption(float temperature, char en[])
{
107
    String padded_litters = addPadding(String(temperature));
109   const char* sensorvalue = padded_litters.c_str();

111   //Start the encryption from here

113   //EAC init
    String result;
115   int key_len = sizeof(eac_key);

117   //Key and keylength
    Sha256.initHmac(eac_key,key_len);
119   Sha256.print(eac_al);
    uint8_t* hash = Sha256.resultHmac();
121   //The returned result is a key pointer hash - point to a a 32
       uint8_t array
    String eac = getHashString(hash);
```

```
123
   const char *eac_c = eac.c_str();
125
   String s="";
127 s = s + (String)id + (String)seq;

129 Sha256.initHmac(hash,32);
   Sha256.print(s);
131 hash = Sha256.resultHmac();
   String eac_onetimekeyString = getHashString(hash);
133
   const char *eac_onetimekey_c = eac_onetimekeyString.c_str();
135
   char* const xor1 = CXOR(sensorvalue,eac_onetimekey_c);
137
   char temp[10] = "000000000";
139 int i;
   for(i=0; i < 8;i++)
141 {
     temp[i] = xor1[i];
143 }
   int p = seq;
145
   temp[8] = (char)p;
147
   base64_encode(en, temp, message_len);
149
   free(xor1);
151
}
153
void loop() // run over and over again
155 {

157 float temperature = getVoltage(temperaturePin);  //getting the
      voltage reading from the temperature sensor
 temperature = ((temperature - .5) * 100)*1.8+32;          //
    converting from 10 mv per degree wit 500 mV offset
159                                                  //to degrees
    ((volatge - 500mV) times 100)
```

```
  char en[20];
161  //Encrypted data - temperature value
  Encryption(temperature,en);
163
  //Reset seq to 0 if it's larger than 256 - One byte stores
    value from 0 -> 255
165  seq++;
  if(seq==256)
167    seq=0;

169  Serial.print("Sent through SIGFOX: "); Serial.println(en);

171  Payload j;
  j.temp=en;
173 // Wait for availability of the modem
  while (!Akene.isReady()) {
175  Serial.println("Modem not ready");
    delay(1000);
177  }
  // Send in the mighty internet!
179   Akene.send(en,12);

181  // Wait for 10 minutes.
  // Note that delay(600000) will block the Arduino (bug in
    delay()?)
183  //Wait for 10 minutes to send new data
  for (int second = 0; second < 60; second++) {
185    delay(10000);
  }
187
}
```

2. Arduino_Sigfox_Debit.ino

```
/*************************************************************
2 This is an example code for getting debit meter data from
    Arduino
and data is sent to SIGFOX
4

1. Arduino collects data
```

```
2. Encrypt the data
3. Send to SIGFOX

**********************************************************/

#include <SoftwareSerial.h> //Built-in library
#include <Akene.h> //SIGFOX Library - Change it if you are using Areku and other devices
#include <stdlib.h> //C library
#include "sha256.h" //HMAC SHA256 library
#include "base64.h" //Base64 encryption/decryption library

const int DELAY=5000;
int seq=0;
const int message_len = 9; //Do not change this - Size of the message
const char id[6]="1BBC9"; //ID of Device, you can change your ID device here, replace "1BBC9"
char * const eac_al = "0"; //Authorization level


//a 64-bit root key
uint8_t eac_key[] ={put your key here};

//XOR function - One time pad
char* CXOR(const char* const c1,const char* const c2)
{
  char* c = strdup(c1);
  int i=0;
  int len=strlen(c1);

  for (i=0;i<len;i++)
  {
    c[i] = c1[i] ^ c2[i];
  }
  return c;
}

String getHashString(const uint8_t* hash) {
  int i;
```

```
  String str;
  for (i=0; i<32; i++) {
    str += ("0123456789abcdef"[hash[i]>>4]);
    str += ("0123456789abcdef"[hash[i]&0xf]);
  }
  return str;
}

void printHash(uint8_t* hash) {
  int i;
  for (i=0; i<32; i++) {
    Serial.print("0123456789abcdef"[hash[i]>>4]);
    Serial.print("0123456789abcdef"[hash[i]&0xf]);
  }
  Serial.println();
}

//Add 0 numbers at the end of a string
String addPadding(String s)
{
  String res= s;
  int s_len = s.length();
  while(s_len<=7)
  {
    res = res+"0";
    s_len++;
  }

  return res;
}

// which pin to use for reading the sensor? can use any pin!
#define FLOWSENSORPIN 10
// count how many pulses!
volatile uint16_t pulses = 0;
// track the state of the pulse pin
volatile uint8_t lastflowpinstate;
// you can try to keep time of how long it is between pulses
volatile uint32_t lastflowratetimer = 0;
// and use that to calculate a flow rate
```

```
volatile float flowrate;
// Interrupt is called once a millisecond, looks for any pulses
    from the sensor!
SIGNAL(TIMER0_COMPA_vect) {
uint8_t x = digitalRead(FLOWSENSORPIN);
if (x == lastflowpinstate) {
lastflowratetimer++;
return; // nothing changed!
}
if (x == HIGH) {
//low to high transition!
pulses++;
}
lastflowpinstate = x;
flowrate = 1000.0;
flowrate /= lastflowratetimer; // in hertz
lastflowratetimer = 0;
}
void useInterrupt(boolean v) {
if (v) {
// Timer0 is already used for millis() - we'll just interrupt
    somewhere
// in the middle and call the "Compare A" function above
OCR0A = 0xAF;
TIMSK0 |= _BV(OCIE0A);
} else {
// do not call the interrupt function COMPA anymore
TIMSK0 &= ~_BV(OCIE0A);
}
}

void setup() {
  Serial.begin(9600);
  pinMode(FLOWSENSORPIN, INPUT);
  digitalWrite(FLOWSENSORPIN, HIGH);
  lastflowpinstate = digitalRead(FLOWSENSORPIN);
  useInterrupt(true);

  // Wait 3 second for the modem to warm
  delay(3000);
```

```
122   Serial.print("Start SIGFOX modem");
      Akene.begin();
124
      delay(30000);
126 }

128 // int is 16 bits, float is 32 bits. All little endian
    typedef struct {
130   String debit;
    } Payload;
132


134 void loop() // run over and over again
    {
136 float liters = pulses;
    liters /= 7.5;
138 liters /= 60.0;

140 //liters contains value collected from debit meter sensor

142 String padded_litters = addPadding(String(liters));
    const char* sensorvalue = padded_litters.c_str();
144
      //Start the encryption from here
146
      //EAC init
148
      String result;
150   int key_len = sizeof(eac_key);

152   //Key and keylength
      Sha256.initHmac(eac_key,key_len);
154   Sha256.print(eac_al);
      uint8_t* hash = Sha256.resultHmac();
156   //The returned result is a key pointer hash - point to a a 32
        uint8_t array
      String eac = getHashString(hash);
158
      const char *eac_c = eac.c_str();
```

```
160
    String s="";
162 s = s + (String)id + (String)seq;

164 Sha256.initHmac(hash,32);
    Sha256.print(s);
166 hash = Sha256.resultHmac();
    String eac_onetimekeyString = getHashString(hash);
168
    const char *eac_onetimekey_c = eac_onetimekeyString.c_str();
170
    char* const xor1 = CXOR(sensorvalue,eac_onetimekey_c);
172
    char temp[10] = "000000000";
174 int i;
    for(i=0; i < 8;i++)
176 {
      temp[i] = xor1[i];
178 }
    int p = seq;
180
    temp[8] = (char)p;
182 char en[20];
    base64_encode(en, temp, message_len);
184 free(xor1);

186 //Reset seq to 0 if it's larger than 256 - One byte stores
     value from 0 -> 255
    seq++;
188 if(seq==256)
      seq=0;
190
    //End of the encryption
192
    Serial.println("-----------------------");
194 Serial.print(sensorvalue); Serial.println(" m/3");
    Serial.print("Sent through SIGFOX: "); Serial.println(en);
196 Serial.println("-----------------------");

198 Payload j;
```

```
    j.debit=en;
200 //Wait for availability of the modem
    while (!Akene.isReady()) {
202 Serial.println("Modem not ready");
      delay(1000);
204 }
    // Send in the mighty internet! - SIGFOX
206  Akene.send(en,12);

208 // Wait for 10 minutes.
    // Note that delay(600000) will block the Arduino (bug in
     delay()?)
210
    //Wait for 10 minutes to send new data
212 for (int second = 0; second < 60; second++) {
      delay(10000);
214 }

216 }
```

2 Python

2.1 Raspberry Pi

1. **pushtoHANA.py**

```
'''
2  This codes get callback function from SIGFOX, and push it to SAP
     HCP
'''
4

import urllib3
6 import certifi
import os
8 import sys
import time
```

```
# disable InsecureRequestWarning if your are working without
    certificate verification
# see https://urllib3.readthedocs.org/en/latest/security.html
# be sure to use a recent enough urllib3 version if this fails
try:
  urllib3.disable_warnings()
except:
  print('urllib3.disable_warnings() failed - get a recent enough
    urllib3 version to avoid potential InsecureRequestWarning
    warnings! Can and will continue though.')

# use with or without proxy
http = urllib3.PoolManager(
  cert_reqs='CERT_REQUIRED', # Force certificate check.
  ca_certs=certifi.where(),  # Path to the Certifi bundle.
)
# http = urllib3.proxy_from_url('http://proxy_host:proxy_port')

# interaction for a specific Device instance - replace 1 with your
    specific Device ID
url = 'https://iotmmsi321696trial.hanatrial.ondemand.com/com.sap.
    iotservices.mms/v1/api/http/data/809fd3e6-1926-4981-a8ad-281
    f53a82792'

headers = urllib3.util.make_headers()

# use with authentication
# please insert correct OAuth token
headers['Authorization'] = 'Bearer ' + '
    faeffcb16e6ba97b777d1e57e807daa'
headers['Content-Type'] = 'application/json;charset=utf-8'

#Get ID
id=sys.argv[1]
#Get Data
data=sys.argv[2]
#Get timestamp
timestamp =sys.argv[3]

# send message of Message Type 1 and the corresponding payload
```

```
    layout that you defined in the IoT Services Cockpit
body='{"mode":"async", "messageType":"a617149678b7e1a669c2", "
    messages":[{"id":'+id+',"data":'+data+',"timestamp":'+timestamp
    +'}]}'

#Push to SAP HCP
r = http.urlopen('POST', url, body=body, headers=headers)
```

2.

3. **Arduino_RasPi_HCP.py**

```
\item \textbf{Arduino\_RasPi\_HCP.py}
\begin{lstlisting}
#This one is used for getting data from Arduino, then push it to
    SAP HCP
#Check the output from Arduino first, then build the message, and
    send it
import os
import sys
import hashlib
from hmac import new as hmac
import serial
import base64
from binascii import unhexlify
import binascii
import time
import urllib3
import certifi
# disable InsecureRequestWarning if your are working without
    certificate verification
# see https://urllib3.readthedocs.org/en/latest/security.html
# be sure to use a recent enough urllib3 version if this fails

#Define a class bcolors with all parameters as follows
class bcolors:
    HEADER = '\033[95m'
    OKBLUE = '\033[94m'
    OKGREEN = '\033[92m'
    WARNING = '\033[93m'
```

```
    FAIL = '\033[91m'
    ENDC = '\033[0m'
    BOLD = '\033[1m'
    UNDERLINE = '\033[4m'

#################################################################
### JSON MANIPULATION FUNCTIONS

#################################################################

def onetimepad(s1, s2):
  return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s1,s2))

# Init Efficient Access Control
# Set al and primary key attributes
def find_between(s, first, last):
  try:
          start = s.index( first ) + len( first )
          end = s.index( last, start )
          return s[start:end]
  except ValueError:
          return ""

ser = serial.Serial('/dev/ttyACM0',9600)
while 1 :
  # Get JSON string from Arduino
  message = ser.readline()
  #Remove  newline, space from Arduino messages
  message = message.strip()
  message = "[{" + message
  date = str(int(time.time()))
  message = message + ",\"timestamp\":\"" + date  + "\"}]"
  print message


  # disable InsecureRequestWarning if your are working without
    certificate verification
    # see https://urllib3.readthedocs.org/en/latest/security.html
      # be sure to use a recent enough urllib3 version if this
    fails
```

```
64      try:
            urllib3.disable_warnings()
66      except:
        print('urllib3.disable_warnings() failed - get a recent enough
        urllib3 version to avoid potential InsecureRequestWarning
        warnings! Can and will continue though.')
68
        # use with or without proxy
70      http = urllib3.PoolManager(
        cert_reqs='CERT_REQUIRED', # Force certificate check.
72      ca_certs=certifi.where(),  # Path to the Certifi bundle.
    )
74        # http = urllib3.proxy_from_url('http://proxy_host:
        proxy_port')

76      # interaction for a specific Device instance - replace 1 with
        your specific Device ID
        url = 'https://iotmmsi321696trial.hanatrial.ondemand.com/com.
        sap.iotservices.mms/v1/api/http/data/809fd3e6-1926-4981-a8ad
        -281f53a82792'
78
          headers = urllib3.util.make_headers()
80
        # use with authentication
82        # please insert correct OAuth token
        headers['Authorization'] = 'Bearer ' + '
        faeffcb16e6ba97b777d1e57e807daa'
84      headers['Content-Type'] = 'application/json;charset=utf-8'

86        # send message of Message Type 1 and the corresponding
        payload layout that you defined in the IoT Services Cockpit
          body='{"mode":"async", "messageType":"ab1d68560d53a408fedd",
         "messages": '+message+'}'
88        r = http.urlopen('POST', url, body=body, headers=headers)
```

4. **LocalToHCP_FULL.py**

```
__author__ = 'I321696'
2
```

```
'''
This code takes a folder as a argument, and push data to SAP HCP
    IoT every 10 seconds
10 seconds pushing = 5 minutes in the offline data

'''
import urllib3
import certifi
import sys
import os
import time
from datetime import datetime, timedelta
import hashlib
from hmac import new as hmac
import base64
from binascii import unhexlify
import binascii
import random
import re

folder=sys.argv[1]
listFile = []
listID = []

#AL  value of EAC
eac_al="0"
# a 64-bit key
eac_key='put your key here'


def extractID(line):
  return 0

def extractSeq(line):
  start = 0
  end = line.index(':',start)
        return line[start:end]

def extractValue(line):
  start1 = line.index(':',0)
```

```
42    start = line.index(':',start1+1)
      end = line.index(':',start+1)
44          return line[start+1:end]

46  def extractType(line):
      start = line.index(':',0)
48    end = line.index(':',start+1)
          return line[start+1:end]
50
    def extractUnit(line):
52    start1 = line.index(':',0)
      start2 = line.index(':',start1+1)
54    start = line.index(':',start2+1)
          return line[start+1:-1]
56
    def onetimepad(s1, s2):
58    return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s1,s2))

60  def find_between(s, first, last):
      try:
62            start = s.index( first ) + len( first )
              end = s.index( last, start )
64            return s[start:end]
      except ValueError:
66            return ""
    #Take a list of ID from the folder
68  def list_files(dirt):
        #Check whether dir exists or not
70      if (os.path.isdir(dirt)):
            global listFile
72    global listFileName
          #Get absolute paths
74        for path, subdirs, files in os.walk(dirt):
              for name in files:
76                listFile.append(os.path.join(path, name))
        ID = path.split(os.path.sep)[-1]
78      listID.append(ID)


80
    list_files(folder)
```

```
82 def encryption(e_id, e_seq,e_value):
    # Generate va0 from the primary key, with al set to 0
84  idseq = e_id + str(e_seq)

86  #Return the hex value of encrypted message
    eac_va0 = hmac(eac_key,eac_al,hashlib.sha256).hexdigest()
88  #print eac_va0
    #eac_va0 remains all time
90
    #Change the encrypted message to hex format
92  eac_test = binascii.a2b_hex(eac_va0)
    eac_onetimekey_c = hmac(eac_test,idseq,hashlib.sha256).
    hexdigest()
94  # print eac_onetimekey_c
    #print eac_onetimekey_c
96  e = onetimepad(e_value,eac_onetimekey_c)

98  e=e+chr(e_seq)
   #print base64.b64encode(e)
100  en = base64.b64encode(e).encode("hex")

102  #print en.encode("hex")


104
    #print en
106  return en

108 #Beginning day
check_date = "20150301000000"
110 t_date = datetime.strptime(check_date,"%Y%m%d%H%M%S")
#Ending day
112 while check_date != "20150401000000":
#    check_time = "20150302000000"
114  message="["
    for i in range(len(listFile)):
116      file = listFile[i]
        ID = listID[i]
118   if ID[1:3].isdigit():
        ID = ID[0:3]
120
```

```
        with open(file) as f:
122      for line in f:
          if check_date in line:
124       #Get timestamp from file
              ts = extractSeq(line)
126           #convert timestamp to datetime format
              date = datetime.strptime(ts,"%Y%m%d%H%M%S")
128           #print date
              t_date = date + timedelta(minutes=5)
130           #convert timestamp to unix timestamp
              timestamp = str(int(time.mktime(date.timetuple())))
132           #print timestamp
              type = extractType(line)
134           type = type.replace("\"", "")
        data = extractValue(line)
136     data = data.replace("\"", "")
        unit = (extractUnit(line))
138     unit = unit.replace("\"", "")

140           unit = repr(unit)
        poe = "\\"
142     poe = poe +'x'
        if poe in unit:
144   unit = unit.replace('\\xb5','  ')
      unit = unit.replace('\\xb0','  ')
146     unit = unit.replace('\\xe9','  ')
          sequence = random.randint(0, 255)
148

          ciphertext= encryption(ID,sequence,data)
150       message = message + "{" + "\"id\":" + ID +',' + "\"
    data\":" + ciphertext +',' + "\"type\":" + type +',' +  "\"unit
    \":"  + unit + ','+ "\"timestamp\":" + timestamp+"}" + ';'

152   message = message[:-1]
      message = message + "]"
154

      print message
156

      try:
158       urllib3.disable_warnings()
```

```
    except:
160 print('urllib3.disable_warnings() failed - get a recent enough
    urllib3 version to avoid potential InsecureRequestWarning
    warnings! Can and will continue though.')

162     # use with or without proxy
        http = urllib3.PoolManager(
164 cert_reqs='CERT_REQUIRED', # Force certificate check.
    ca_certs=certifi.where(),  # Path to the Certifi bundle.
166     )
        # http = urllib3.proxy_from_url('http://proxy_host:proxy_port
        ')
168
        # interaction for a specific Device instance - replace 1 with
        your specific Device ID
170     url = 'https://iotmmsi321696trial.hanatrial.ondemand.com/com.
        sap.iotservices.mms/v1/api/http/data/809fd3e6-1926-4981-a8ad
        -281f53a82792'

172     headers = urllib3.util.make_headers()

174     # use with authentication
        # please insert correct OAuth token
176     headers['Authorization'] = 'Bearer ' + '
        faeffcb16e6ba97b777d1e57e807daa'
        headers['Content-Type'] = 'application/json;charset=utf-8'
178
        # send message of Message Type 1 and the corresponding payload
        layout that you defined in the IoT Services Cockpit
180     body='{"mode":"async", "messageType":"ab1d68560d53a408fedd", "
        messages": '+message+'}'
        r = http.urlopen('POST', url, body=body, headers=headers)
182
        time.sleep(10)
```

5. **ExtractCoordinates.py**

```
1 __author__ = 'I321696'
```

```
'''
  This example will read a geolocation file and push it to SAP HCP
'''
import os
import sys
import urllib3
import certifi

def find_between(s, first, last):
  try:
          start = s.index( first ) + len( first )
          end = s.index( last, start )
          return s[start:end]
  except ValueError:
          return ""
message="["
fname=sys.argv[1]
# get sensor ids
with open(fname) as f:
  for line in f:
    if "NUMERO" in line:
      id = find_between(line,'\"NUMERO\": "', '\",')
      if id != '0':
        coordinate = find_between(line,'\"coordinates\": [',']')
        #print id + " - " + coordinate
        #Reverse the coordinate, because they are not correct
        coordinate = coordinate.split(',')

        coordinate = coordinate[1].strip() + ", " + coordinate[0].
    strip()

        message = message + "{" + "\"ID\":" + id +',' + "\"
    Coordinate\":" + "\"" +coordinate +"\"" + "}" + ';'
message = message[:-1]
message = message + "]"
print message

# disable InsecureRequestWarning if your are working without
    certificate verification

```

```
# see https://urllib3.readthedocs.org/en/latest/security.html
# be sure to use a recent enough urllib3 version if this fails
try:
  urllib3.disable_warnings()
except:
  print('urllib3.disable_warnings() failed - get a recent enough
    urllib3 version to avoid potential InsecureRequestWarning
    warnings! Can and will continue though.')

# use with or without proxy
http = urllib3.PoolManager(
  cert_reqs='CERT_REQUIRED', # Force certificate check.
  ca_certs=certifi.where(),  # Path to the Certifi bundle.
)
# http = urllib3.proxy_from_url('http://proxy_host:proxy_port')

# interaction for a specific Device instance - replace 1 with your
    specific Device ID
url = 'https://iotmmsi321696trial.hanatrial.ondemand.com/com.sap.
    iotservices.mms/v1/api/http/data/809fd3e6-1926-4981-a8ad-281
    f53a82792'

headers = urllib3.util.make_headers()

# use with authentication
# please insert correct OAuth token
headers['Authorization'] = 'Bearer ' + '
    faeffcb16e6ba97b777d1e57e807daa'
headers['Content-Type'] = 'application/json;charset=utf-8'

# send message of Message Type 1 and the corresponding payload
    layout that you defined in the IoT Services Cockpit
body='{"mode":"async", "messageType":"bb7f2789479bf02c2799", "
    messages": '+message+'}'
r = http.urlopen('POST', url, body=body, headers=headers)
```

2.2 Intel Edison

- **sendtoSIGFOX.py**

```
# coding=utf-8
__author__ = 'I321696'

#!/usr/bin/python

import time
import serial
import sys
from time import sleep

#LINE ADDED
import mraa

import urllib3
import certifi
import sys
import os
import time
from datetime import datetime, timedelta
import hashlib
from hmac import new as hmac
import base64
from binascii import unhexlify
import binascii
import random
import re


SOH = chr(0x01)
STX = chr(0x02)
EOT = chr(0x04)
ACK = chr(0x06)
NAK = chr(0x15)
CAN = chr(0x18)
CRC = chr(0x43)
#ID of device - SIGFOX modem ID
id = '74136'
```

```
#AL  value of EAC
eac_al="0"
# a 64-bit key
eac_key='put your key here'

#XOR function
def onetimepad(s1, s2):
  return ''.join(chr(ord(a) ^ ord(b)) for a,b in zip(s1,s2))

#Encryption fuction
def encryption(e_id, e_seq,e_value):
    if str(e_value) > 8:
        # Generate va0 from the primary key, with al set to 0
        idseq = e_id + str(e_seq)

        #Return the hex value of encrypted message
        eac_va0 = hmac(eac_key,eac_al,hashlib.sha256).hexdigest()
        #print eac_va0
        #eac_va0 remains all time

        #Change the encrypted message to hex format
        eac_test = binascii.a2b_hex(eac_va0)
        eac_onetimekey_c = hmac(eac_test,idseq,hashlib.sha256).
    hexdigest()
       # print eac_onetimekey_c
        #print eac_onetimekey_c
        e = onetimepad(e_value,eac_onetimekey_c)

        e=e+chr(e_seq)
       #print base64.b64encode(e)
        en = base64.b64encode(e).encode("hex")

        #print en.encode("hex")
        #print en
        return en
    return 0

#Decryption function
def decryption(e_id, e_data):
    #Generate va0 from the primary key, with al set to 0
```

```
    #If ciphertext is hex data - Convert it to ASCII value
    data = e_data.decode("hex")
  #Base64 decoding
    data = base64.decodestring(data)

    #Get sequence number
    e_seq = ord(data[8])

    #Concat ID and Seq
    idseq = e_id + str(e_seq)

    #Return the hex value of encrypted message
    eac_va0 = hmac(eac_key,eac_al,hashlib.sha256).hexdigest()

    #Change the encrypted message to hex format
    eac_test = binascii.a2b_hex(eac_va0)
    eac_onetimekey_c = hmac(eac_test,idseq,hashlib.sha256).hexdigest()
    #Do one-time-pad to decrypt the data with key eac_onetimekey_c
    plaintext = onetimepad(data[0:7],eac_onetimekey_c)

    return plaintext


def getc(size, timeout=1):
    return ser.read(size)

def putc(data, timeout=1):
    ser.write(data)
    sleep(0.001) # give device time to prepare new buffer and start
   sending it

def WaitFor(ser, s, timeOut):
        nbMax = 0
        ser.timeout = timeOut
        currentMsg = ''
        while currentMsg.endswith(s) != True :
            # should add a try catch here
            c=ser.read()
            if c != '' :
                currentMsg += c
```

```
116             else :
                    print 'timeout waiting for ' + s
118                 return False
                nbMax = nbMax + 1
120             if nbMax > 150:
    print 'Timeout expired'
122     return False
        return True
124
def SendToSIGFOX(message):
126     print('Sending SIGFOX Message...')

128     #LINE ADDED: 0 ie '/dev/ttyMFD1'
        uart = mraa.Uart(0)
130
        modem = serial.Serial(
132                 uart.getDevicePath(),
                    baudrate=9600,
134                 parity=serial.PARITY_NONE,
                    stopbits=serial.STOPBITS_ONE,
136                 bytesize=serial.EIGHTBITS
                )
138
        #LINE ADDED: closing serial before opening it, as otherwise the
        Edison serial seems to be already open by calling serial.Serial
140     modem.close()

142     modem.open()

144     modem.write('AT\r')

146     if (WaitFor(modem, 'OK', 3)):
            print('SIGFOX Modem OK')
148     else:
            print('SIGFOX Modem Error')
150         modem.close()
            exit()
152
        modem.write("AT$SS={0}\r".format(message))
154
```

```
    print('Sending ...')
    if (WaitFor(modem, 'OK', 15)):
        print('OK Message sent')
    else:
        print('Error Sending message')
        modem.close()
        exit()

    modem.close()
#Sequence number is equal to 0 at the beginning
sequence = 0
while True:
    #Get random value from 500 to 1000 - Can replace the value of
    sensor here
    value = random.randint(0, 99999999)
    #Convert value to string
    value = str(value)
    #Get random value of sequence number

    #Get the ciphertext as a hex number
    ciphertext = encryption(id,sequence,value)

    #Send to SIGFOX this message
    SendToSIGFOX(ciphertext)

    #Print out screen the cyphertext
    print ciphertext
    print
    sequence = sequence + 1
    #if the sequence number = 256 -> reset it to 0
    if sequence == 256:
        sequence = 0
    sleep(600)
```

3 Javascript

1. **sha.js** is the library for HMAC SHA256 in Javascript

```
/*
 A JavaScript implementation of the SHA family of hashes, as
 defined in FIPS PUB 180-2 as well as the corresponding HMAC
    implementation
 as defined in FIPS PUB 198a

 Copyright Brian Turek 2008-2015
 Distributed under the BSD License
 See http://caligatio.github.com/jsSHA/ for more information

 Several functions taken from Paul Johnston
*/
'use strict';(function(T){function y(c,a,d){var b=0,f=[],k=0,g,e,n
    ,h,m,u,r,p=!1,q=!1,t=[],v=[],x,w=!1;d=d||{};g=d.encoding||"UTF8
    ";x=d.numRounds||1;n=J(a,g);if(x!==parseInt(x,10)||1>x)throw
    Error("numRounds must a integer >= 1");if("SHA-1"===c)m=512,u=K
    ,r=U,h=160;else if(u=function(a,d){return L(a,d,c)},r=function(
    a,d,b,f){var k,e;if("SHA-224"===c||"SHA-256"===c)k=(d
    +65>>>9<<4)+15,e=16;else if("SHA-384"===c||"SHA-512"===c)k=(d
    +129>>>10<<5)+31,e=32;else throw Error("Unexpected error in SHA
    -2 implementation");
for(;a.length<=k;)a.push(0);a[d>>>5]|=128<<24-d%32;a[k]=d+b;b=a.
    length;for(d=0;d<b;d+=e)f=L(a.slice(d,d+e),f,c);if("SHA-224"===
    c)a=[f[0],f[1],f[2],f[3],f[4],f[5],f[6]];else if("SHA-256"===c)
    a=f;else if("SHA-384"===c)a=[f[0].a,f[0].b,f[1].a,f[1].b,f[2].a
    ,f[2].b,f[3].a,f[3].b,f[4].a,f[4].b,f[5].a,f[5].b];else if("SHA
    -512"===c)a=[f[0].a,f[0].b,f[1].a,f[1].b,f[2].a,f[2].b,f[3].a,f
    [3].b,f[4].a,f[4].b,f[5].a,f[5].b,f[6].a,f[6].b,f[7].a,f[7].b];
    else throw Error("Unexpected error in SHA-2 implementation");
return a},"SHA-224"===c)m=512,h=224;else if("SHA-256"===c)m=512,h
    =256;else if("SHA-384"===c)m=1024,h=384;else if("SHA-512"===c)m
    =1024,h=512;else throw Error("Chosen SHA variant is not
    supported");e=z(c);this.setHMACKey=function(a,d,f){var k;if
    (!0===q)throw Error("HMAC key already set");if(!0===p)throw
    Error("Cannot set HMAC key after finalizing hash");if(!0===w)
    throw Error("Cannot set HMAC key after calling update");g=(f
    ||{}).encoding||"UTF8";d=J(d,g)(a);a=d.binLen;d=d.value;k=m
    >>>3;f=k/4-1;if(k<
a/8){for(d=r(d,a,0,z(c));d.length<=f;)d.push(0);d[f]&=4294967040}
```

```
    else if(k>a/8){for(;d.length<=f;)d.push(0);d[f]&=4294967040}for
    (a=0;a<=f;a+=1)t[a]=d[a]^909522486,v[a]=d[a]^1549556828;e=u(t,e
    );b=m;q=!0};this.update=function(a){var c,d,g,h=0,p=m>>>5;c=n(a
    ,f,k);a=c.binLen;d=c.value;c=a>>>5;for(g=0;g<c;g+=p)h+m<=a&&(e=
    u(d.slice(g,g+p),e),h+=m);b+=h;f=d.slice(h>>>5);k=a%m;w=!0};
    this.getHash=function(a,d){var g,m,n;if(!0===q)throw Error("
    Cannot call getHash after setting HMAC key");n=M(d);switch(a){
    case "HEX":g=
16  function(a){return N(a,n)};break;case "B64":g=function(a){return O
    (a,n)};break;case "BYTES":g=P;break;default:throw Error("format
     must be HEX, B64, or BYTES");}if(!1===p)for(e=r(f,k,b,e),m=1;m
    <x;m+=1)e=r(e,h,0,z(c));p=!0;return g(e)};this.getHMAC=function
    (a,d){var g,n,t;if(!1===q)throw Error("Cannot call getHMAC
    without first setting HMAC key");t=M(d);switch(a){case "HEX":g=
    function(a){return N(a,t)};break;case "B64":g=function(a){
    return O(a,t)};break;case "BYTES":g=P;break;default:throw Error
    ("outputFormat must be HEX, B64, or BYTES");
}!1===p&&(n=r(f,k,b,e),e=u(v,z(c)),e=r(n,h,m,e));p=!0;return g(e)
    }}function b(c,a){this.a=c;this.b=a}function V(c,a,d){var b=c.
    length,f,k,e,l,n;a=a||[0];d=d||0;n=d>>>3;if(0!==b%2)throw Error
    ("String of HEX type must be in byte increments");for(f=0;f<b;f
    +=2){k=parseInt(c.substr(f,2),16);if(isNaN(k))throw Error("
    String of HEX type contains invalid characters");l=(f>>>1)+n;
    for(e=l>>>2;a.length<=e;)a.push(0);a[e]|=k<<8*(3-l%4)}return{
    value:a,binLen:4*b+d}}function W(c,a,d){var b=[],f,k,e,l,b=a
    ||[0];d=
18  d||0;k=d>>>3;for(f=0;f<c.length;f+=1)a=c.charCodeAt(f),l=f+k,e=l
    >>>2,b.length<=e&&b.push(0),b[e]|=a<<8*(3-l%4);return{value:b,
    binLen:8*c.length+d}}function X(c,a,d){var b=[],f=0,e,g,l,n,h,m
    ,b=a||[0];d=d||0;a=d>>>3;if(-1===c.search(/^[a-zA-Z0-9=+\/]+$/)
    )throw Error("Invalid character in base-64 string");g=c.indexOf
    ("=");c=c.replace(/\=/g,"");if(-1!==g&&g<c.length)throw Error("
    Invalid '=' found in base-64 string");for(g=0;g<c.length;g+=4){
    h=c.substr(g,4);for(l=n=0;l<h.length;l+=1)e="
    ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
    +/".indexOf(h[l]),
n|=e<<18-6*l;for(l=0;l<h.length-1;l+=1){m=f+a;for(e=m>>>2;b.length
    <=e;)b.push(0);b[e]|=(n>>>16-8*l&255)<<8*(3-m%4);f+=1}}return{
    value:b,binLen:8*f+d}}function N(c,a){var d="",b=4*c.length,f,e
    ;for(f=0;f<b;f+=1)e=c[f>>>2]>>>8*(3-f%4),d+="0123456789abcdef".
```

```
    charAt(e>>>4&15)+"0123456789abcdef".charAt(e&15);return a.
    outputUpper?d.toUpperCase():d}function O(c,a){var d="",b=4*c.
    length,f,e,g;for(f=0;f<b;f+=3)for(g=f+1>>>2,e=c.length<=g?0:c[g
    ],g=f+2>>>2,g=c.length<=g?0:c[g],g=(c[f>>>2]>>>8*(3-f%4)&255)
    <<16|
(e>>>8*(3-(f+1)%4)&255)<<8|g>>>8*(3-(f+2)%4)&255,e=0;4>e;e+=1)8*f
    +6*e<=32*c.length?d+="
    ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789
    +/".charAt(g>>>6*(3-e)&63):d+=a.b64Pad;return d}function P(c){
    var a="",d=4*c.length,b,f;for(b=0;b<d;b+=1)f=c[b>>>2]>>>8*(3-b
    %4)&255,a+=String.fromCharCode(f);return a}function M(c){var a
    ={outputUpper:!1,b64Pad:"="};c=c||{};a.outputUpper=c.
    outputUpper||!1;a.b64Pad=c.b64Pad||"=";if("boolean"!==typeof a.
    outputUpper)throw Error("Invalid outputUpper formatting option
    ");
if("string"!==typeof a.b64Pad)throw Error("Invalid b64Pad
    formatting option");return a}function J(c,a){var d;switch(a){
    case "UTF8":case "UTF16BE":case "UTF16LE":break;default:throw
    Error("encoding must be UTF8, UTF16BE, or UTF16LE");}switch(c){
    case "HEX":d=V;break;case "TEXT":d=function(c,d,b){var e=[],l
    =[],n=0,h,m,u,r,p,e=d||[0];d=b||0;u=d>>>3;if("UTF8"===a)for(h
    =0;h<c.length;h+=1)for(b=c.charCodeAt(h),l=[],128>b?l.push(b)
    :2048>b?(l.push(192|b>>>6),l.push(128|b&63)):55296>b||57344<=b?
    l.push(224|
b>>>12,128|b>>>6&63,128|b&63):(h+=1,b=65536+((b&1023)<<10|c.
    charCodeAt(h)&1023),l.push(240|b>>>18,128|b>>>12&63,128|b
    >>>6&63,128|b&63)),m=0;m<l.length;m+=1){p=n+u;for(r=p>>>2;e.
    length<=r;)e.push(0);e[r]|=l[m]<<8*(3-p%4);n+=1}else if("
    UTF16BE"===a||"UTF16LE"===a)for(h=0;h<c.length;h+=1){b=c.
    charCodeAt(h);"UTF16LE"===a&&(m=b&255,b=m<<8|b>>>8);p=n+u;for(r
    =p>>>2;e.length<=r;)e.push(0);e[r]|=b<<8*(2-p%4);n+=2}return{
    value:e,binLen:8*n+d}};break;case "B64":d=X;break;case "BYTES":
    d=W;break;default:throw Error("format must be HEX, TEXT, B64,
    or BYTES");
}return d}function w(c,a){return c<<a|c>>>32-a}function q(c,a){
    return c>>>a|c<<32-a}function v(c,a){var d=null,d=new b(c.a,c.b
    );return d=32>=a?new b(d.a>>>a|d.b<<32-a&4294967295,d.b>>>a|d.a
    <<32-a&4294967295):new b(d.b>>>a-32|d.a<<64-a&4294967295,d.a>>>
    a-32|d.b<<64-a&4294967295)}function Q(c,a){var d=null;return d
    =32>=a?new b(c.a>>>a,c.b>>>a|c.a<<32-a&4294967295):new b(0,c.a
```

```
>>>a-32)}function Y(c,a,d){return c&a^~c&d}function Z(c,a,d){
return new b(c.a&a.a^~c.a&d.a,c.b&a.b^~c.b&d.b)}function R(c,a,
d){return c&
a^c&d^a&d}function aa(c,a,d){return new b(c.a&a.a^c.a&d.a^a.a&d.a,
c.b&a.b^c.b&d.b^a.b&d.b)}function ba(c){return q(c,2)^q(c,13)^q
(c,22)}function ca(c){var a=v(c,28),d=v(c,34);c=v(c,39);return
new b(a.a^d.a^c.a,a.b^d.b^c.b)}function da(c){return q(c,6)^q(c
,11)^q(c,25)}function ea(c){var a=v(c,14),d=v(c,18);c=v(c,41);
return new b(a.a^d.a^c.a,a.b^d.b^c.b)}function fa(c){return q(c
,7)^q(c,18)^c>>>3}function ga(c){var a=v(c,1),d=v(c,8);c=Q(c,7)
;return new b(a.a^d.a^c.a,a.b^d.b^c.b)}function ha(c){return q(
c,
17)^q(c,19)^c>>>10}function ia(c){var a=v(c,19),d=v(c,61);c=Q(c,6)
;return new b(a.a^d.a^c.a,a.b^d.b^c.b)}function B(c,a){var d=(c
&65535)+(a&65535);return((c>>>16)+(a>>>16)+(d>>>16)&65535)<<16|
d&65535}function ja(c,a,d,b){var f=(c&65535)+(a&65535)+(d
&65535)+(b&65535);return((c>>>16)+(a>>>16)+(d>>>16)+(b>>>16)+(f
>>>16)&65535)<<16|f&65535}function C(c,a,d,b,f){var e=(c&65535)
+(a&65535)+(d&65535)+(b&65535)+(f&65535);return((c>>>16)+(a
>>>16)+(d>>>16)+(b>>>16)+(f>>>16)+(e>>>16)&65535)<<16|e&65535}
function ka(c,
a){var d,e,f;d=(c.b&65535)+(a.b&65535);e=(c.b>>>16)+(a.b>>>16)+(d
>>>16);f=(e&65535)<<16|d&65535;d=(c.a&65535)+(a.a&65535)+(e
>>>16);e=(c.a>>>16)+(a.a>>>16)+(d>>>16);return new b((e&65535)
<<16|d&65535,f)}function la(c,a,d,e){var f,k,g;f=(c.b&65535)+(a
.b&65535)+(d.b&65535)+(e.b&65535);k=(c.b>>>16)+(a.b>>>16)+(d.b
>>>16)+(e.b>>>16)+(f>>>16);g=(k&65535)<<16|f&65535;f=(c.a
&65535)+(a.a&65535)+(d.a&65535)+(e.a&65535)+(k>>>16);k=(c.a
>>>16)+(a.a>>>16)+(d.a>>>16)+(e.a>>>16)+(f>>>16);return new b((
k&65535)<<16|
f&65535,g)}function ma(c,a,d,e,f){var k,g,l;k=(c.b&65535)+(a.b
&65535)+(d.b&65535)+(e.b&65535)+(f.b&65535);g=(c.b>>>16)+(a.b
>>>16)+(d.b>>>16)+(e.b>>>16)+(f.b>>>16)+(k>>>16);l=(g&65535)
<<16|k&65535;k=(c.a&65535)+(a.a&65535)+(d.a&65535)+(e.a&65535)
+(f.a&65535)+(g>>>16);g=(c.a>>>16)+(a.a>>>16)+(d.a>>>16)+(e.a
>>>16)+(f.a>>>16)+(k>>>16);return new b((g&65535)<<16|k&65535,l
)}function z(c){var a,d;if("SHA-1"===c)c
=[1732584193,4023233417,2562383102,271733878,3285377520];else
switch(a=[3238371032,914150663,
812702999,4144912697,4290775857,1750603025,1694076839,3204075428],
```

```
    d
    =[1779033703,3144134277,1013904242,2773480762,1359893119,2600822924,52873
    c){case "SHA-224":c=a;break;case "SHA-256":c=d;break;case "SHA
    -384":c=[new b(3418070365,a[0]),new b(1654270250,a[1]),new b
    (2438529370,a[2]),new b(355462360,a[3]),new b(1731405415,a[4]),
    new b(41048885895,a[5]),new b(3675008525,a[6]),new b
    (1203062813,a[7])];break;case "SHA-512":c=[new b(d
    [0],4089235720),new b(d[1],2227873595),new b(d[2],4271175723),
new b(d[3],1595750129),new b(d[4],2917565137),new b(d
    [5],725511199),new b(d[6],4215389547),new b(d[7],327033209)];
    break;default:throw Error("Unknown SHA variant");}return c}
    function K(c,a){var d=[],b,e,k,g,l,n,h;b=a[0];e=a[1];k=a[2];g=a
    [3];l=a[4];for(h=0;80>h;h+=1)d[h]=16>h?c[h]:w(d[h-3]^d[h-8]^d[h
    -14]^d[h-16],1),n=20>h?C(w(b,5),e&k^~e&g,l,1518500249,d[h]):40>
    h?C(w(b,5),e^k^g,l,1859775393,d[h]):60>h?C(w(b,5),R(e,k,g),l
    ,2400959708,d[h]):C(w(b,5),e^k^g,l,3395469782,d[h]),l=g,g=k,k=w
    (e,30),e=b,b=n;a[0]=
30 B(b,a[0]);a[1]=B(e,a[1]);a[2]=B(k,a[2]);a[3]=B(g,a[3]);a[4]=B(l,a
    [4]);return a}function U(c,a,b,e){var f;for(f=(a+65>>>9<<4)+15;
    c.length<=f;)c.push(0);c[a>>>5]|=128<<24-a%32;c[f]=a+b;b=c.
    length;for(a=0;a<b;a+=16)e=K(c.slice(a,a+16),e);return e}
    function L(c,a,d){var q,f,k,g,l,n,h,m,u,r,p,v,t,w,x,y,z,D,E,F,G
    ,H,A=[],I;if("SHA-224"===d||"SHA-256"===d)r=64,v=1,H=Number,t=B
    ,w=ja,x=C,y=fa,z=ha,D=ba,E=da,G=R,F=Y,I=e;else if("SHA-384"===d
    ||"SHA-512"===d)r=80,v=2,H=b,t=ka,w=la,x=ma,y=ga,z=ia,D=ca,E=ea
    ,G=aa,
F=Z,I=S;else throw Error("Unexpected error in SHA-2 implementation
    ");d=a[0];q=a[1];f=a[2];k=a[3];g=a[4];l=a[5];n=a[6];h=a[7];for(
    p=0;p<r;p+=1)16>p?(u=p*v,m=c.length<=u?0:c[u],u=c.length<=u
    +1?0:c[u+1],A[p]=new H(m,u)):A[p]=w(z(A[p-2]),A[p-7],y(A[p-15])
    ,A[p-16]),m=x(h,E(g),F(g,l,n),I[p],A[p]),u=t(D(d),G(d,q,f)),h=n
    ,n=l,l=g,g=t(k,m),k=f,f=q,q=d,d=t(m,u);a[0]=t(d,a[0]);a[1]=t(q,
    a[1]);a[2]=t(f,a[2]);a[3]=t(k,a[3]);a[4]=t(g,a[4]);a[5]=t(l,a
    [5]);a[6]=t(n,a[6]);a[7]=t(h,a[7]);return a}var e,S;e
    =[1116352408,
32 1899447441,3049323471,3921009573,961987163,1508970993,2453635748,2870763221,

    430227734,506948616,659060556,883997877,958139571,1322822218,1537002063,1747
    S=[new b(e[0],3609767458),new b(e[1],602891725),new b(e
    [2],3964484399),new b(e[3],2173295548),new b(e[4],4081628472),
```

```
    new b(e[5],3053834265),new b(e[6],2937671579),new b(e
    [7],3664609560),new b(e[8],2734883394),new b(e[9],1164996542),
    new b(e[10],1323610764),new b(e[11],3590304994),new b(e
    [12],4068182383),new b(e[13],991336113),new b(e[14],
633803317),new b(e[15],3479774868),new b(e[16],2666613458),new b(e
    [17],944711139),new b(e[18],2341262773),new b(e[19],2007800933)
    ,new b(e[20],1495990901),new b(e[21],1856431235),new b(e
    [22],3175218132),new b(e[23],2198950837),new b(e
    [24],3999719339),new b(e[25],766784016),new b(e[26],2566594879)
    ,new b(e[27],3203337956),new b(e[28],1034457026),new b(e
    [29],2466948901),new b(e[30],3758326383),new b(e[31],168717936)
    ,new b(e[32],1188179964),new b(e[33],1546045734),new b(e
    [34],1522805485),new b(e[35],2643833823),
new b(e[36],2343527390),new b(e[37],1014477480),new b(e
    [38],1206759142),new b(e[39],344077627),new b(e[40],1290863460)
    ,new b(e[41],3158454273),new b(e[42],3505952657),new b(e
    [43],106217008),new b(e[44],3606008344),new b(e[45],1432725776)
    ,new b(e[46],1467031594),new b(e[47],851169720),new b(e
    [48],3100823752),new b(e[49],1363258195),new b(e
    [50],3750685593),new b(e[51],3785050280),new b(e
    [52],3318307427),new b(e[53],3812723403),new b(e
    [54],2003034995),new b(e[55],3602036899),new b(e
    [56],1575990012),
new b(e[57],1125592928),new b(e[58],2716904306),new b(e
    [59],442776044),new b(e[60],593698344),new b(e[61],3733110249),
    new b(e[62],2999351573),new b(e[63],3815920427),new b
    (3391569614,3928383900),new b(3515267271,566280711),new b
    (3940187606,3454069534),new b(4118630271,4000239992),new b
    (116418474,1914138554),new b(174292421,2731055270),new b
    (289380356,3203993006),new b(460393269,320620315),new b
    (685471733,587496836),new b(852142971,1086792851),new b
    (1017036298,365543100),new b(1126000580,2618297676),
new b(1288033470,3409855158),new b(1501505948,4234509866),new b
    (1607167915,987167468),new b(1816402316,1246189591)];"function"
    ===typeof define&&define.amd?define(function(){return y}):"
    undefined"!==typeof exports?"undefined"!==typeof module&&module
    .exports?module.exports=exports=y:exports=y:T.jsSHA=y})(this);
```

2. **Decryption.html**

```
<!DOCTYPE html>
<html>
    <head>
        <title>Decryption</title>
        <meta charset="UTF-8">
    </head>
    <body>
    <form name ="decrypt">
    Encrypted message:
    <input type="text" id="data" size="25">
    ID:
    <input type="text" id="id2" size="10">
    <input id="decrypt" type="button" value="Decrypt" onclick="
    decryption()"/>
    <br />
    <br />
    <textarea rows="4" cols="50" id="result">
    </textarea>
    </form>
    <script type="text/javascript" src="sha.js"></script>
    <script type="text/javascript">

    function asciibin(convert, type)
    {

      //if this is binary...
      if (type == 'binary') {
        //strip any white space
        convert = convert.replace(/ /g,'');

        //remove any line breaks
        convert = convert.replace(/\n/g,'');

        var ascii_string = '';
        var current_byte;

        //if the binary does not consist of 8 digit chunks (bytes)
        if (convert.length % 8 != 0) {return 'Binary length is not
    divisible by eight.'}
```

```
39        //otherwise
          else {
41          //split it into bytes
            for (var i=0; i < convert.length/8; i++) {
43
              //get the current byte
45            current_byte = convert.substring(i*8, (i*8)+8);

47            //get the charcode from the current_byte and add it to
     the ascii_string
              ascii_string += String.fromCharCode(parseInt(
    current_byte, 2));
49          }
            return ascii_string;
51        }
      }
53
      //if this is an ASCII
55    if (type == 'ascii') {
        var binary_string = '';
57      var current_letter;

59      //separate the letters
        for (var i=0; i < convert.length; i++) {
61        //get the current letter, pull the char code, and
    convert to binary
          current_letter = convert.substring(i, i + 1).charCodeAt
    (0).toString(2);
63
          //if the 'byte' is less than eight values prepend some
    zeros
65        if (current_letter.length < 8) {
            while(current_letter.length < 8) {
67            current_letter = '0' + current_letter;
            }
69        }
          binary_string += current_letter;
71      }
        return binary_string;
```

```
73        }
      }
75
      function getChar(str,no)
77    {
        var res='';
79      var i=0;
        for(i;i<no; i++)
81      {
          res = res + str[i];
83      }
        return res;
85    }

87    function onetimepad(v,k)
      {
89      var res="";
        var length = v.length;
91      var i=0;

93      for(i;i<length;i++)
        {
95        res = res + (v[i]^k[i]);
        }
97
        return res;
99    }

101   function hex2a(hex)
      {
103     var str = '';
        for (var i = 0; i < hex.length; i += 2)
105       str += String.fromCharCode(parseInt(hex.substr(i, 2), 16))
      ;
        return str;
107   }

109   function decryption()
      {
111
```

```
        //Get the id from the text box
113     var id = document.getElementById('id2').value;

115     //Get the ciphertext from textbox ;
        var hexdata = document.getElementById('data').value;
117
        var data = hex2a(hexdata);
119
        //decode base 64 the string
121     var decodedString = atob(data);

123     //Get the position of sequence number from the encryption
    data
        len_decodedString = decodedString.length -1;
125
        //Get read value sensor from chars - the last one is
    sequence
127     var value = getChar(decodedString,len_decodedString);

129     //Get sequence number
        var seq = decodedString[len_decodedString].charCodeAt(0);
131
        //EAC Init
133     var hmacText = "0"; //AL Value
        var hmacTextType = "TEXT"; //Type : "TEXT", "Base-64", "HEX"
135     var hmacKeyInput = "put your key here";
        var hmacKeyInputType ="HEX"; //Type: "Base-64", "HEX"
137     var hmacVariant = "SHA-256"; // "SHA-1", "SHA-224", SHA
    -256", "SHA-384", "SHA-512"
        var hmacOutputType = "HEX"; //Type: "Base-64", "HEX"
139     var hmacObj = new jsSHA(
            hmacVariant,
141         hmacTextType
        );
143
        hmacObj.setHMACKey(
145         hmacKeyInput,
            hmacKeyInputType
147     );
```

```
149         hmacObj.update(hmacText);

151         //Get the first key from the first HMAC-SHA256 HASH
            var K1 = hmacObj.getHMAC(hmacOutputType);
153
            //2nd EAC Init
155         //Concat iq + seq
            var hmacText2 = id + seq;
157         var hmacTextType2 = "TEXT";
            var hmacKeyInput2 = K1;
159         var hmacKeyInputType2 ="HEX";
            var hmacVariant2 = "SHA-256";
161         var hmacOutputType2 = "HEX";
            var hmacObj2 = new jsSHA(
163             hmacVariant,
                hmacTextType
165         );

167         hmacObj2.setHMACKey(
                hmacKeyInput2,
169             hmacKeyInputType2
            );
171
            hmacObj2.update(hmacText2);
173
            //Obtain the second key from the second HMAC-SHA256
175         var K2 = hmacObj2.getHMAC(hmacOutputType2);

177         //Get the first length-byte of the key
            var key = getChar(K2,len_decodedString)
179
            //Transform key, value to binary for Xoring
181         var value_binary = asciibin(value,'ascii');
            var key_binary = asciibin(key,'ascii');
183
            //Get the binary value by xoring
185         var encrypted_value_binary = onetimepad(value_binary,
    key_binary);

187         //Transform back the binary value to ascii letters
```

```
    decrypted_value = asciibin(encrypted_value_binary,"binary");
189
    //Display the result in the textbox
191 result.value = decrypted_value;
  }
193
  /**********************
195 This function has inputs
  id : id of the device
197 hexdata : ciphertext

199 return : real value
  **********************/
201
  function decryption2(id,hexdata)
203 {

205   var data = hex2a(hexdata);

207   //decode base 64 the string
    var decodedString = atob(data);
209
    //Get the position of sequence number from the encryption
  data
211   len_decodedString = decodedString.length -1;

213   //Get read value sensor from chars - the last one is
  sequence
    var value = getChar(decodedString,len_decodedString);
215
    //Get sequence number
217 var seq = decodedString[len_decodedString].charCodeAt(0);

219 //EAC Init
    var hmacText = "0"; //AL Value
221 var hmacTextType = "TEXT"; //Type : "TEXT", "Base-64", "HEX"
    var hmacKeyInput = "put your key here";
223 var hmacKeyInputType ="HEX"; //Type: "Base-64", "HEX"
    var hmacVariant = "SHA-256"; // "SHA-1", "SHA-224", SHA
  -256", "SHA-384", "SHA-512"
```

```
225         var hmacOutputType = "HEX"; //Type: "Base-64", "HEX"
            var hmacObj = new jsSHA(
227             hmacVariant,
                hmacTextType
229         );

231         hmacObj.setHMACKey(
                hmacKeyInput,
233             hmacKeyInputType
            );
235
            hmacObj.update(hmacText);
237
            //Get the first key from the first HMAC-SHA256 HASH
239         var K1 = hmacObj.getHMAC(hmacOutputType);

241         //2nd EAC Init
            //Concat iq + seq
243         var hmacText2 = id + seq;
            var hmacTextType2 = "TEXT";
245         var hmacKeyInput2 = K1;
            var hmacKeyInputType2 ="HEX";
247         var hmacVariant2 = "SHA-256";
            var hmacOutputType2 = "HEX";
249         var hmacObj2 = new jsSHA(
                hmacVariant,
251             hmacTextType
            );
253
            hmacObj2.setHMACKey(
255             hmacKeyInput2,
                hmacKeyInputType2
257         );

259         hmacObj2.update(hmacText2);

261         //Obtain the second key from the second HMAC-SHA256
            var K2 = hmacObj2.getHMAC(hmacOutputType2);
263
            //Get the first length-byte of the key
```

```
265         var key = getChar(K2,len_decodedString)

267         //Transform key, value to binary for Xoring
            var value_binary = asciibin(value,'ascii');
269         var key_binary = asciibin(key,'ascii');

271         //Get the binary value by xoring
            var encrypted_value_binary = onetimepad(value_binary,
        key_binary);
273
            //Transform back the binary value to ascii letters
275         decrypted_value = asciibin(encrypted_value_binary,"binary");

277         //Display the result in the textbox
            return decrypted_value;
279      }

281         </script>


283
      </body>
285 </html>
```

BIBLIOGRAPHY

[1] McKinsey Research], *The Internet of Things: Mapping the Value Beyond the Hype*, available at http://www.mckinsey.com/insights/business_technology/the_internet_of_things_the_value_of_digitizing_the_physical_world.

[2] Wikipedia, *Internet of Things*, available at https://en.wikipedia.org/wiki/Internet_of_Things.

[3] SAP, *Predictive Maintenance*, available at http://www.sap.com/pc/tech/internet-of-things/software/predictive-maintenance/index.html.

[4] Wikipedia, *Programmable logic controller*, available at https://en.wikipedia.org/wiki/Programmable_logic_controller.

[5] Raspberry Pi, *Raspberry Pi*, available at https://www.raspberrypi.org/.

[6] SIGFOX, *SIGFOX*, available at http://www.sigfox.com/en/.

[7] Wikipedia, *Representational state transfer*, available at https://en.wikipedia.org/wiki/Representational_state_transfer.

[8] SAP, *SAP HANA Cloud Platform for the Internet of Things*, available at http://go.sap.com/product/technology-platform/iot-platform-cloud.html.

[9] ———, *SAP HANA Database*, available at https://www.sapappsdevelopmentpartnercenter.com/en/get-started/big-data/.

[10] gregorwolf, *SAP HANA Cloud Authentication Proxy for HANA XS*, available at https://github.com/gregorwolf/hanatrial-auth-proxy.

[11] Genuino, *Arduino Yun*, available at https://www.arduino.cc/en/Guide/ArduinoYun.

[12] ———, *Arduino Uno*, available at https://www.arduino.cc/en/Main/ArduinoBoardUno.

[13] SAP, *SAP HANA Cloud Platform*, available at http://hcp.sap.com/index.html.

[14] ———, *SAP HANA Extended Application Services (XS)*, available at http://scn.sap.com/docs/DOC-60322.

[15] _____, *SAP HANA Cloud Platform Cockpit*, available at https://account.hanatrial.ondemand.com/.

[16] _____, *SAP HANA Cloud Connector*, available at https://hcp.sap.com/developers/TutorialCatalog/con100_1_setting_up_cloud_connector.html.

[17] Refik Molva Alessandro Sorniotti Laurent Gomez, *Efficient Access Control for Wireless Sensor Data*, available at http://www.eurecom.fr/en/publication/2473/download/ce-publi-2473_3.pdf.

[18] Nathan Chenette Alexandra Boldyreva Younho Lee and Adam ONeill, *Order-Preserving Symmetric Encryption*, available at http://www.cc.gatech.edu/~aboldyre/papers/bclo.pdf.

[19] Arduino, *Arduino*, available at https://www.arduino.cc/.

[20] Snootlab, *Akene v1*, available at http://snootlab.fr/lang-en/snootlab-shields/889-akene-v1-en.html/.

[21] Arduino, *Arduino IDE website*, available at https://www.arduino.cc/en/Main/Software/.

[22] *urllib3 library*, available at https://urllib3.readthedocs.org/en/latest/.

[23] *certifi library*, available at https://pypi.python.org/pypi/certifi.

[24] *Intel Edison*, available at https://software.intel.com/en-us/iot/hardware/edison.

[25] *Yocto Linux*, available at https://www.yoctoproject.org/.

[26] *pyserial 2.7 library*, available at https://pypi.python.org/pypi/pyserial.

[27] exasens, *Simple as sending IoT sensor values through SigFox and Intel Edison, using Python*, available at http://www.instructables.com/id/Simple-as-sending-IoT-sensor-values-through-SIGFOX/?ALLSTEPS.

[28] SIGFOX, *Callback API*, available at https://backend.sigfox.com/apidocs/callback.

[29] FreeDataMap, *FreeDataMap Application*, available at http://www.freedatamap.com/.

[30] HMAC SHA256, *Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec*, available at https://tools.ietf.org/html/rfc4868.

[31] JetBrains, *Pycharm*, available at https://www.jetbrains.com/pycharm/download/.

[32] *Arduino debugging tool*, available at http://playground.arduino.cc/Code/VisualMicro.

[33] *Visual Studio Community 2013*, available at https://www.visualstudio.com/en-us/downloads/download-visual-studio-vs.aspx.

[34] McNeight, *Free Memory library for Arduino*, available at http://playground.arduino.cc/Code/AvailableMemory.

[35] *Create Device Type on HCP for IoT Service*, available at https://help.hana.ondemand.com/iot/frameset.htm.

[36] *SAP Web IDE*, available at `https://uxexplorer.hana.ondemand.com/_item.html?id=10667`.

[37] *urllib3 library*, available at `https://urllib3.readthedocs.org/en/latest/`.

[38] *oData, futher reading*, available at `https://en.wikipedia.org/wiki/Open_Data_Protocol`.

[39] *HANA XS*, available at `http://scn.sap.com/docs/DOC-60322`.