MangOH multi-protocol converged IoT End-device

- Prototype development

LIN Huijie
PICOU Léo
SELVARAJ Mohan Prabhu
XIONG Xuantang
QU Zhe

5th year students in Department Electronic and Computer Science,
specialized in Innovative Smart System

Tutors : THIERRY Monteil, TRICOT Clément

Abstract

Kinéis, a New Space IoT Network Operator wants to create a converged IoT End-device with transceiver capabilities of both terrestrial cellular/LPWAN and SATCOM protocols. We attempted to develop a prototype based on MangOH to validate the multiprotocol IoT device concept. The goals are to validate the automatic selection of network on the basis of availability of terrestrial or SATCOM network and to validate the sensor data collection, transmission and reporting to the user. At mangOH, we used Legato, a Linux based embedded application development framework for implementation of sensor data collection. For data transmission, we could validate the cellular (3G) network only. A Django server (coupled with Nginx, Gunicorn and Docker) was used to retrieve, merge and display the sensor data to the end-user. We succeeded in sensor data collection, transmission using cellular network and reporting to the user using charts & visuals. We did succeed in encoding the sensor data in Kinéis ARGOS format for SATCOM transmission. In our attempt to develop a strategy that delivers the longest battery life, we inferred that, with a standard 28.8 wh battery, approximately 313 days of battery life can be achieved. Once the dependencies/barriers related to key components are resolved by Kinéis team the automatic network selection feature can be developed & tested.

Keywords: satellite, terrestrial network, network switching, MangOH, open source hardware, legato

Table of Contents

Ⅰ. Introduction

Kinéis, a subsidiary of CLS Ltd. has the vision of becoming the New Space IoT Network operator. The company's mission is to establish a constellation of satellites in the LEO orbit of space and use it as a SATCOM communication network for IoT use-cases on earth such as telemetry, tracking, alarms. Conventionally the IoT End-devices used in IoT applications & services have support for only terrestrial network technologies such as Cellular (3G/4G/LTE-M) and LPWAN (LoRA, Sigfox, NB-IoT) for data transmission [15]. Because of which the IoT use-case deployments are constrained within the coverage scope of terrestrial network [14, 15]. This is a business pain area for IoT network operators and end customers who wants to use SATCOM network and deploy IoT use-cases in remote terrains & oceans where terrestrial network coverage is not available. Kinéis believes that by making a End-device that can switch between terrestrial network and their SATCOM network for data transmission it can address a huge business opportunity. Kinéis plans to develop a prototype of the end-device using the open source mangOH hardware and KIM-1, a SATCOM transmitter module for validation of the concept.

From INSA, our group joins the Kinéis endeavour in developing a prototype for 'multiprotocol mangOH converged IoT End-device'. The important project goals are,

- Access mangOH Sensors & data retrieval
- choose between Cellular & SATCOM transmission based on network availability
- transmit the sensor data to central Server via either cellular or SATCOM network
- process the data & plot as graphs & report it to the user.

1) Project Architecture

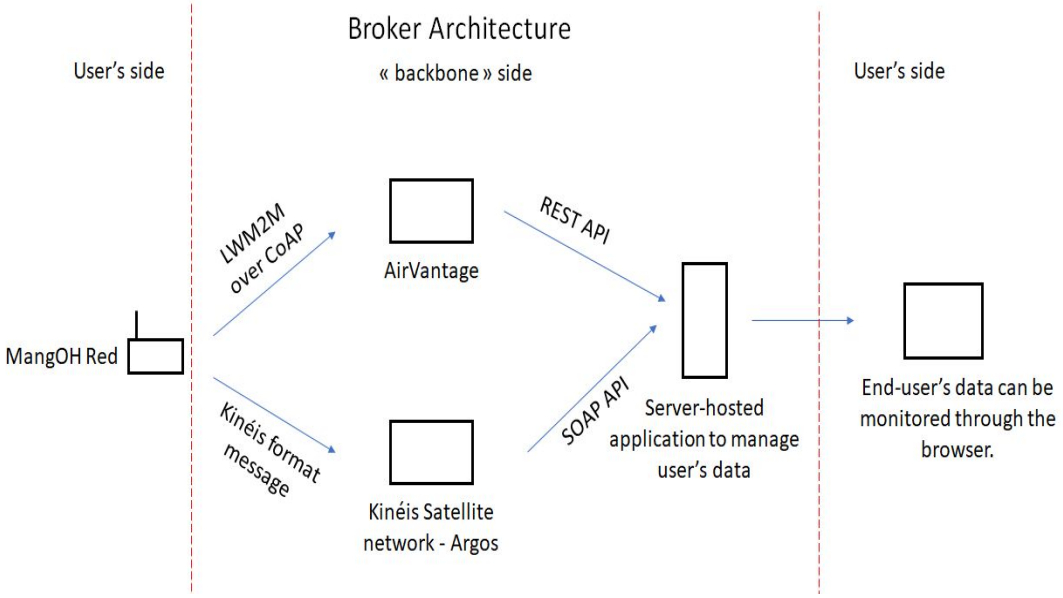

Fig 1. Architecture

As you can see in Fig 1, our main goal is to create a two-way transmission, which can switch from cellular network (Airvantage) to Kinéis Satellite network (Argos) whenever the terrestrial cellular network is not available.

2) Time Organization

We followed a time schedule as find in fig.2.

Fig.2 Timeline

3) Components Description

a) MangOH-Red

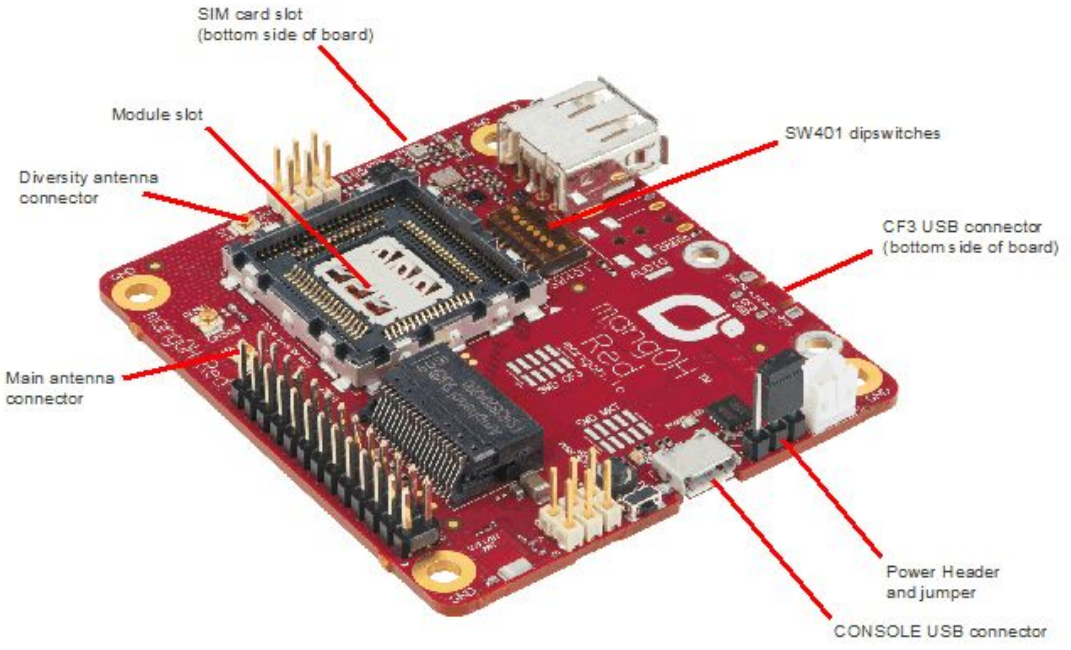

Fig.3 MangOH-Red

- Cortex-M4 with RTOS : The cortex processor runs real-time operating system whose application's data comes in minimum delay.
- Built-in sensors : Including sensors for temperature, light and air pressure.
- USB connectivity : Permitting us to write our software with the open-source Legato Linux platform.
- Raspberry Pi Shield Connector : Allowing us to plug in different HATs to our design[9].

b) Kim1 Module

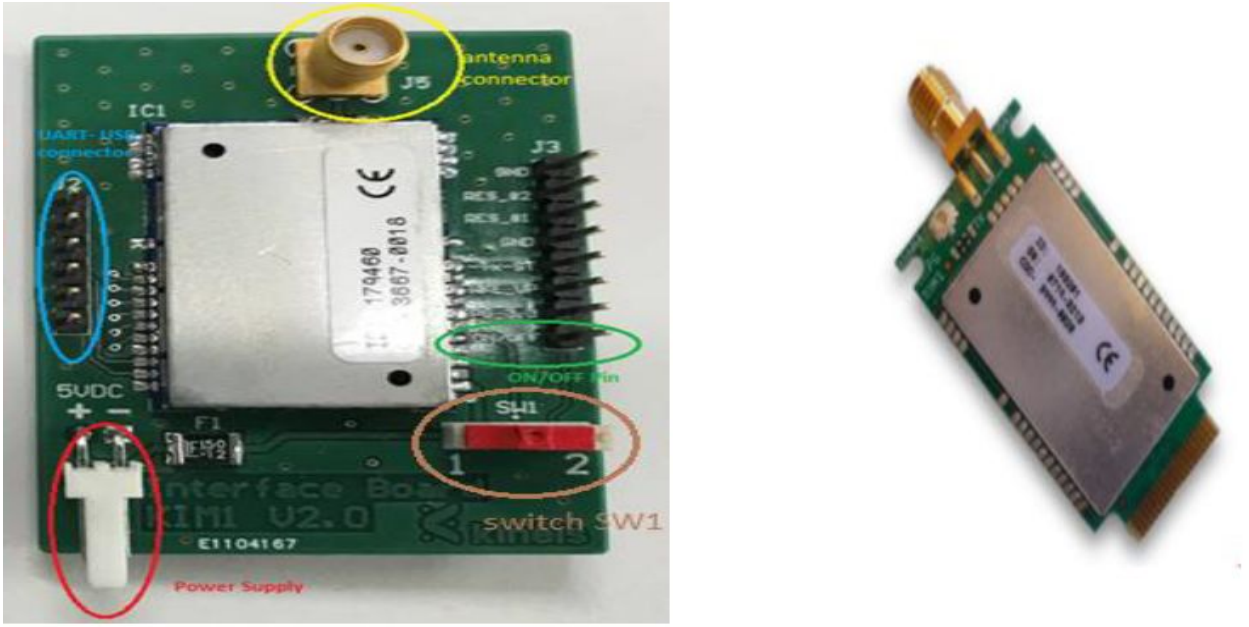

Fig.4 Kim1 Interface Board and Kim 1 module

The Kim1 interface board that connect with computer uses a basic TTL 3.3V level signals with UART protocol[8].

- Power supply connector (red)
- USB-UART connector (blue)
- Antenna connector (yellow)
- ON/OFF Pin (green)

We plug the Kim1 module in MangoH board with port UART1.

Ⅱ. Embedded System

In this part, we will describe the sensors we used in our project and the process of collecting, packaging and transmitting the sensor data as a specific message that can be analyzed by users. After that, we will explain the transmission process via satellites and via cellular networks separately. In the end, we will gather these two transmitting methods into our embedded application and show the results.

1) Sensors

a) Description

We used five different sensors in our project as referred in table-1, which are a temperature sensor, light sensor, pressure sensor, acceleration sensor, and location sensor. These sensors are based on MangOH open-source platform.

Light sensor: PNJ4K01F ambient light sensor, this sensor uses ADC3 to communicate with the microprocessor[18]. We can use interface 'le_adc_ReadValue' to retrieve the light intensity easily[7].

Location sensor: We connect it with mangoH board. We can use the interface 'le_pos_Get3DLocation' to retrieve the GPS location of our board.

Name	Type	Bus	Interface
Light Sensor	PNJ4K01F	ADC3	le_adc_ReadValue
Pressure sensor /Temperature Sensor	BMP280 barometric pressure sensor	I2C	
Acceleration /Gyroscope Sensor	integrated BMI160 sensor	I2C	
Location Sensor	external GPS module	U.FL connectors	le_pos_Get3DLocation

Table.1 Sensors Used

Pressure sensor and temperature sensor: They are integrated into one sensor, this sensor communicates with the microprocessor by using I2C[16]. In legato system(operating system on mangoH board), there are two files which stock the data of temperature and pressure value. We retrieve the temperature and value of pressure from these two files.

Accelerometer sensor: It communicates with microprocessor by using bus I2C, the data are stocked in a specific file, we should retrieve the acceleration of the board from this file[7].

b) Data Treating

Since the data we got from the sensors above are all in the form of integer it can not be transmitted as a message. So, we wrote a function to transfer them into the chain of char.

2) Cellular Network

We plugged a 3G sim card into MangOH board to send our messages by the cellular network. In the part 'Data Treating', we know that now we can retrieve data and transfer it into the form 'char'.

So based on it, we use an integrated application of MangOH board (Redsensortocloud) to send our messages by cellular network to the Airvantage Platform. The function of this application includes an interruption that can send our messages to the cloud, changing the sending frequency, and also retrieve the useful data from sensors.

3) Satellites Network

We used a Kim1 module of Kinéis to make MangOH-red board to connect with satellites.

Before starting to begin our transmission, we need to learn about the message format at first. Because all the messages sent to the satellites will be decoded by Kinéis' data center which expects the sensor data to be in a specific message format.

There are three standard **message formats**[20]**:**

1) Raw message without formatting
2) message with one GPS position and
3) message with one GPS position + raw message

For this prototype development, we attempt using the 3rd option. The message format is as referred in Fig. 4[20]:

Data Field	#bits allocated	Content
CRC	16	CRC16 CCITT X16 + X12 + X5 + 1 (from "PERIOD_ACQ_GPS» to "USER_DATA")
PERIOD_ACQ_GPS	3	"111": Service Loc Inst & Messages
DAT_DAY_1	5	[1-31] days
DAT_HOUR_1	5	[0-23] hours
DAT_MIN_1	6	[0-59] minutes
LON_1	22	22 bits: from 0 (0.0000°) to 3 600 000 (360.0000°) LON_1 [0°,360°] resol GPS LON: 0.0001° (~11m at the equator)
LAT_1	21	21 bits: from 0 (0.0000°) to 1 800 000 (180.0000°) (LAT_1 - 90.0000°) [-90°,+90°] resol GPS LAT: 0.0001° (~11m)
ALT_1	10	10 bits: from 0 (0m) to 1023 (10 230m) resol GPS ALT: 10m
USER_DATA	160	20 characters, 8bits per character (table 258 characters ASCII. Fixed length that needs to be filled with 0 if data size is inferior to 160bits

Fig.4 Message Format

As we can see in the figure above, we need to put all our sensor data into the last 'USER_DATA' part (160 bits). We organize the encoding as below:

Light Sensor: The range of the light intensity is between 0 and 3000 lumens, so we use 12 bits.

Pressure: the range of the air pressure sensor is from 950 to 1050 hPa +/- 0.12hPa = 833 values, so we use 10 bits.

Temperature: the range of the temperature sensor is from -40 to 85°C +/- 0.01°C = 12500 values, so we use 14 bits.

Acceleration: from the datasheet, we can know that we need to use 16 bits.

Gyro: 16 bits per axis.

The range of measurement for both acceleration and gyroscope sensor depends on the sensitivity selected.

We designed the message format as referred in fig. 5 to match the requirements imposed by the Kinéis maximum payload size :

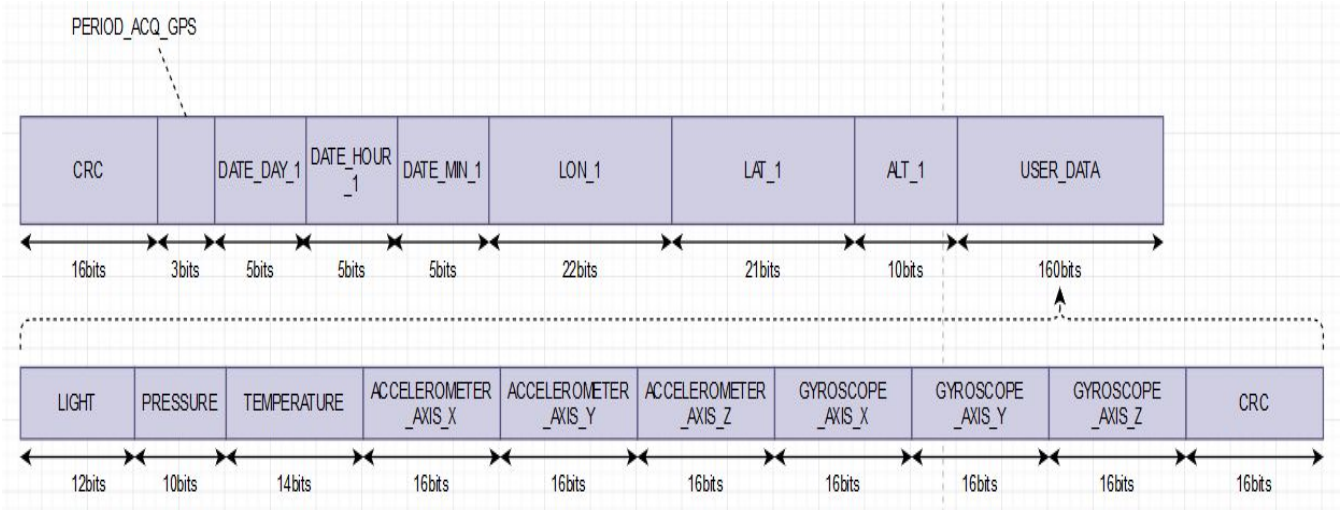

Fig.5 Message format precisement

Besides that, we will use a coding theory to ensure the accuracy of our message during transmission. There are two possibilities for us: 32 bits-BCH and 16 bits-CRC. Since our sensors data has already taken 148 bits of 160 bits, the theory BCH can not satisfy our requirement. So if we insist on using BCH theory, the precision of our sensors data will decrease.

In November, we got a test Kim1 module from Kinéis which can be used to simply send our messages to satellites with the satellite prediction and an integrated PC application as can be found in Fig.6.

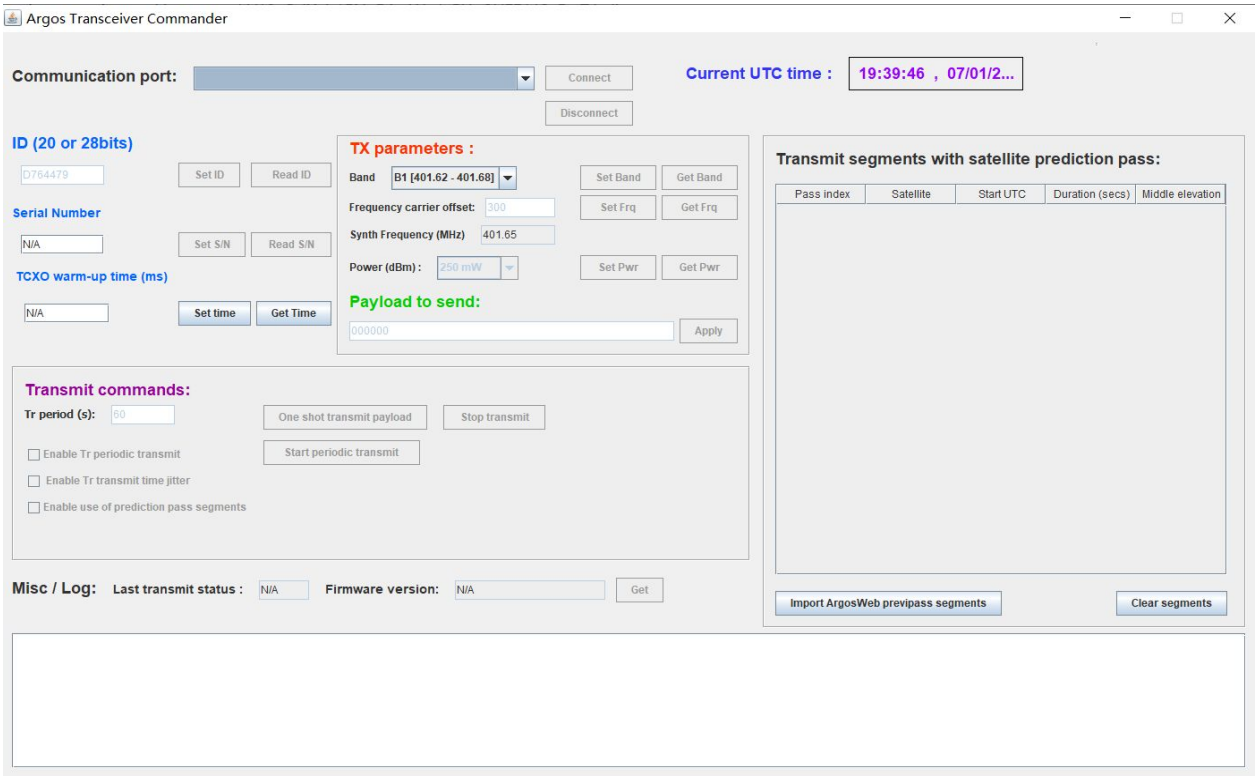

Fig.6 Argos Transceiver Commander

In this PC application, we can get the ID of the Kim1 module, and set several parameters as found in Fig.7 for the transmission, such as Band of frequency, power consumption, and the payload to send. In the section 'Transmit commands', we should choose that we send the message only once or periodically or with the pass prediction file of satellites. We can get the pass prediction file of satellites on ArgosWeb.

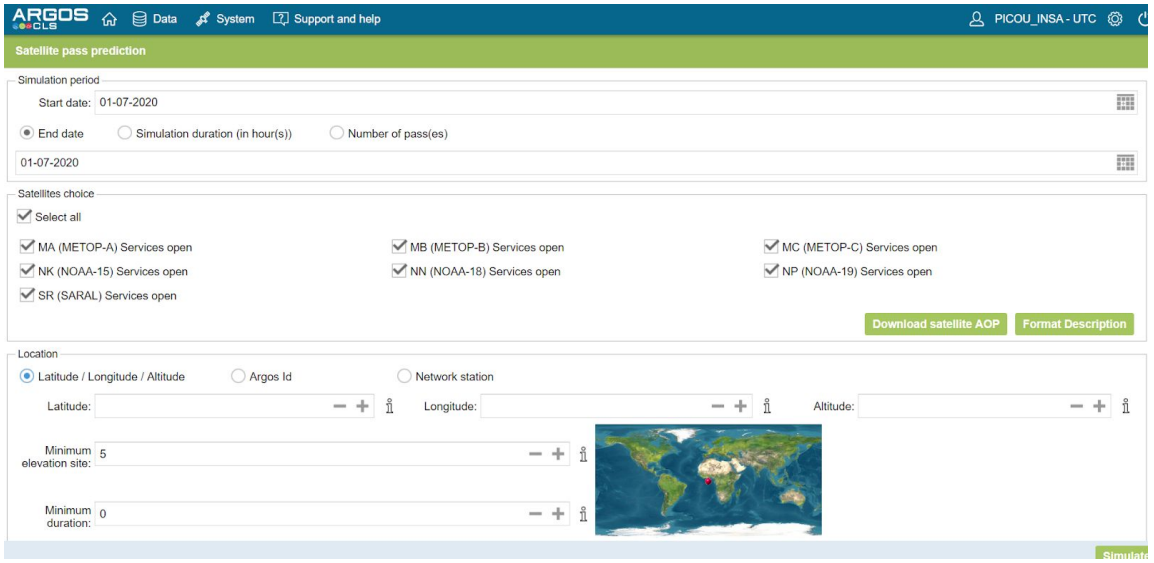

Fig.7 Argos Platform

As we can see, we should choose a period of prediction, the satellite that we want to use, and the latitude/longitude/Altitude of our device. And then, we got a CSV form pass prediction file that we can import in the software of Kinéis (Fig 6). And it will send the payload when a satellite passes over the device.

With this simple module, we can test our front-end application for the satellite part. But we can't combine the module with mangoH board.

Finally, we received the module on 15 December. We plugged the module on the board and try to execute the demo code. Unfortunately, the demo code from the company didn't work. So we should wait for them to solve the problem. In the datasheet, the module needs 2A to transmit the data to the satellite, but the USB interface can only supply 900mA. So, we used an external AC adapter.

4) Embedded Application

The main challenge of this project is to switch the network between Cellular (3G/NB-IoT) and satellite. At the beginning, we thought that with the final Kim1 module, there was an API (Application Programming Interface) which can detect if there is a satellite over the beacon. And with this API , we can develop our application easily. But we didn't receive the API until the middle January.

During the waiting time, we tried to find another solution. We found a defaut application of Legato (Operating system of MangOH-red) which can change the transmission network between cellular network and wifi. So we tried to replace Wifi with our satellite network. But unfortunately, the Legato application can not be used to manage the Kim1 module which is necessary for satellite communication.

On 14-Jan, we received the demo API of Kim1 module, and can execute it successfully. But for some unknown reason, we can not send our messages to satellites. We are still trying to figure out the problem. We suspect that the power from the USB port is not sufficient enough for KIM-1 module.

Ⅲ. mangOH sensors data processing & reporting

Referring to Fig 1, we can understand that the sensors data can follow two different paths :

- If the cellular network (3/4/5G..) isn't available, the MangOH will send the data via the satellite link to the *Argos* cloud, thanks to the Kinéis KIM-1 module installed on the card.
- If the cellular network is available, the MangOH will use the CF3 embedded module to send the data to the AirVantage cloud.

Since satellite communications are more expensive than terrestrial cellular communications, we understand the importance of this switching capability.
The 5G connectivity offered by the CF3 module provide a full-stack TCP/IP communication, open to the whole Internet. Therefore we could have sent the data to a private server instead of the AirVantage platform, but the current situation is preferable in two ways :

1. As Sierra Wireless wants to promote the openSource MangOH project, the integration of the cellular connectivity, the mangOH card and the AirVantage cloud was already perfect and worked really well. A Legato application's basic package was installed during the setup of the card. A redevelopment of this part was therefore not considered, because we decided to spend our time on the switching mechanism and the satellite transmission.
2. The AirVantage platform can store the data for a longer duration and that too from numerous concurrent devices. In the nearby future, reducing the cost of the data storage and ensuring the user's data availability can be good for Kinéis prototypes. The scalability of the platform is another key feature that is suitable for our project.

Concerning the satellite connectivity, we did not have the choice to send the data elsewhere than Argos, since the channel works in a different manner.

Once the data were stored in both clouds, our next task was to design an architecture to retrieve the data from both clouds, merge them, and display them to the user. We considered two of the following different architectures as can be referred from fig.8 and fig.9.

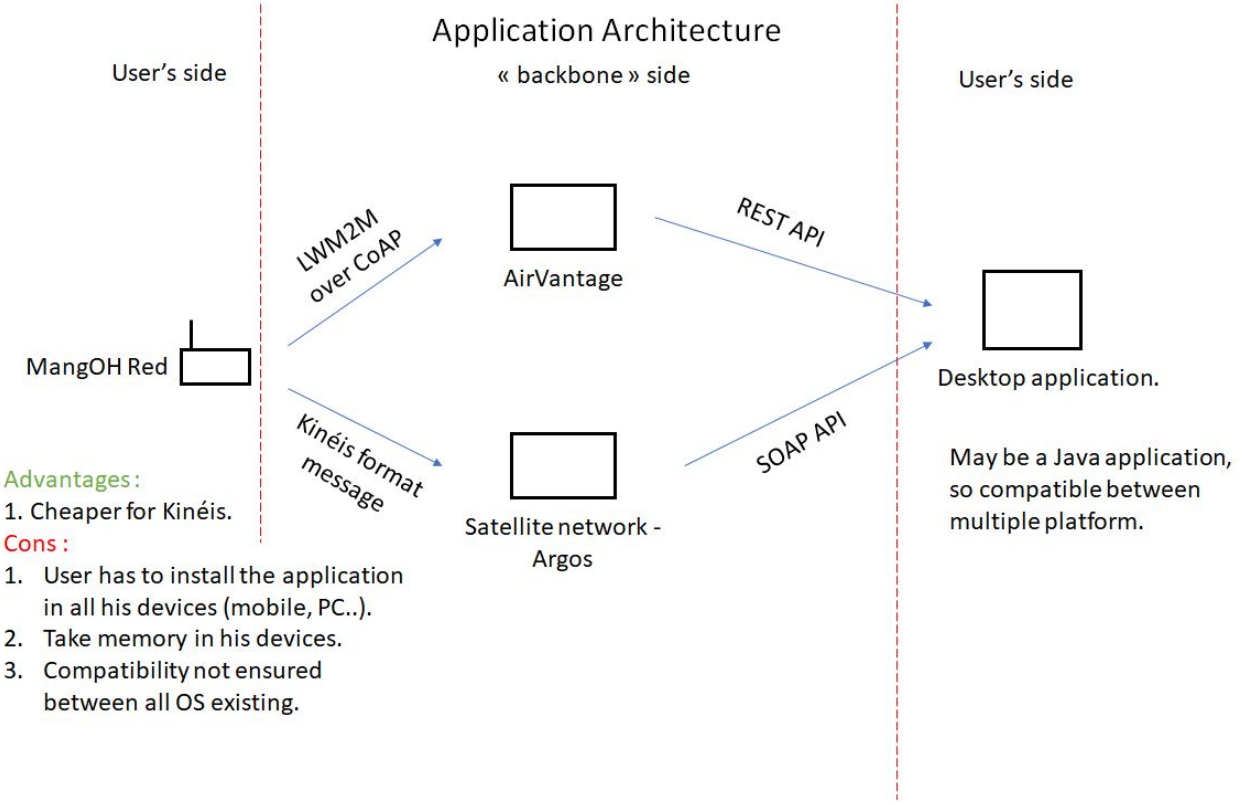

Fig.8 - Application Architecture

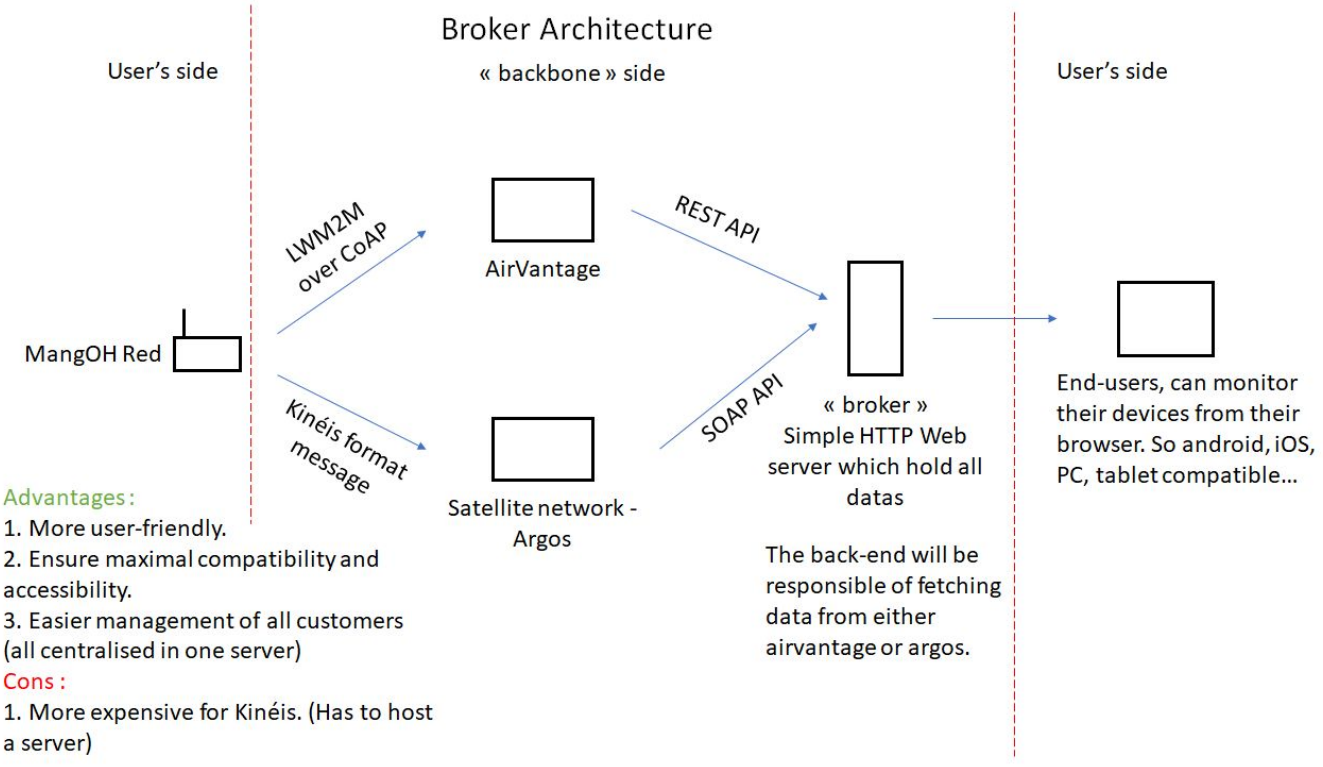

Fig.9 - Broker Architecture

Each architecture hold its set of advantages and drawbacks. After discussion and analysis with Kinéis, we decided to develop the Broker architecture in Fig.9 because the advantages are more significant than the drawbacks. We also found easier to develop a robust web server than a desktop application. We know today that there are really easy solution to deploy stable and secure web sites with a friendly responsive user-interface. Since the core of the project is the switching capability of the MangOH, we didn't want to spend too much time on a desktop application. Furthermore, a web application ensures a maximal compatibility among all users's devices since it runs on their browser. It also allows Kinéis to perform easy maintenance and management because they own the server.

The only drawback could be the cost, because Kinéis needs to host a server which can hold all the required traffic, and it could be huge. On the other hand, the cost of development and maintenance for a desktop application can also be significant.

We decided to use a Python framework called Django to develop the web server, because we wanted to try a different back-end environment than the traditional PHP. Python is easy to use, and everyone in the team has used it before, in contrary of Javascript (Node.JS) or PHP. We also could have developed a purely front-end website, but we decided to avoid doing this to lighten the client resource needed by the website. Indeed if all the processing of thousand of data sample is done in Javascript, it could have a heavy impact on the user experience.

A. User interface

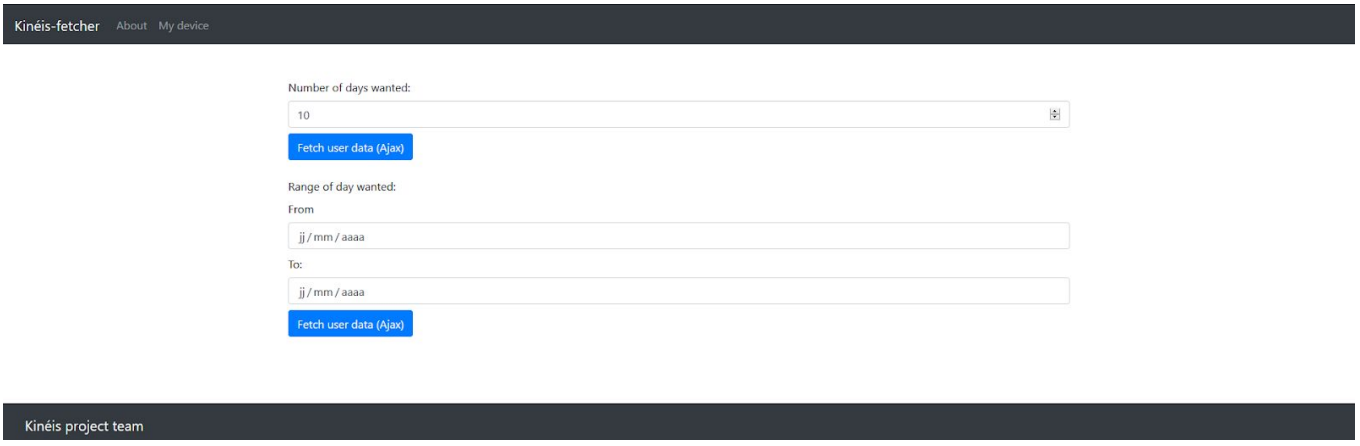

Fig.10 - Index page

This is the index web page which provides the users with two possibilities to recover the data - either by selecting a number of days by counting from today, or by selecting a range of date. Once the users selected their preferred option, their data is displayed in their browser. This interface is then scrollable, to show the different graphs available. The users can also click, zoom or download these plots as it is fully interactive. They are designed to take the full width of the screen in order to improve readability.

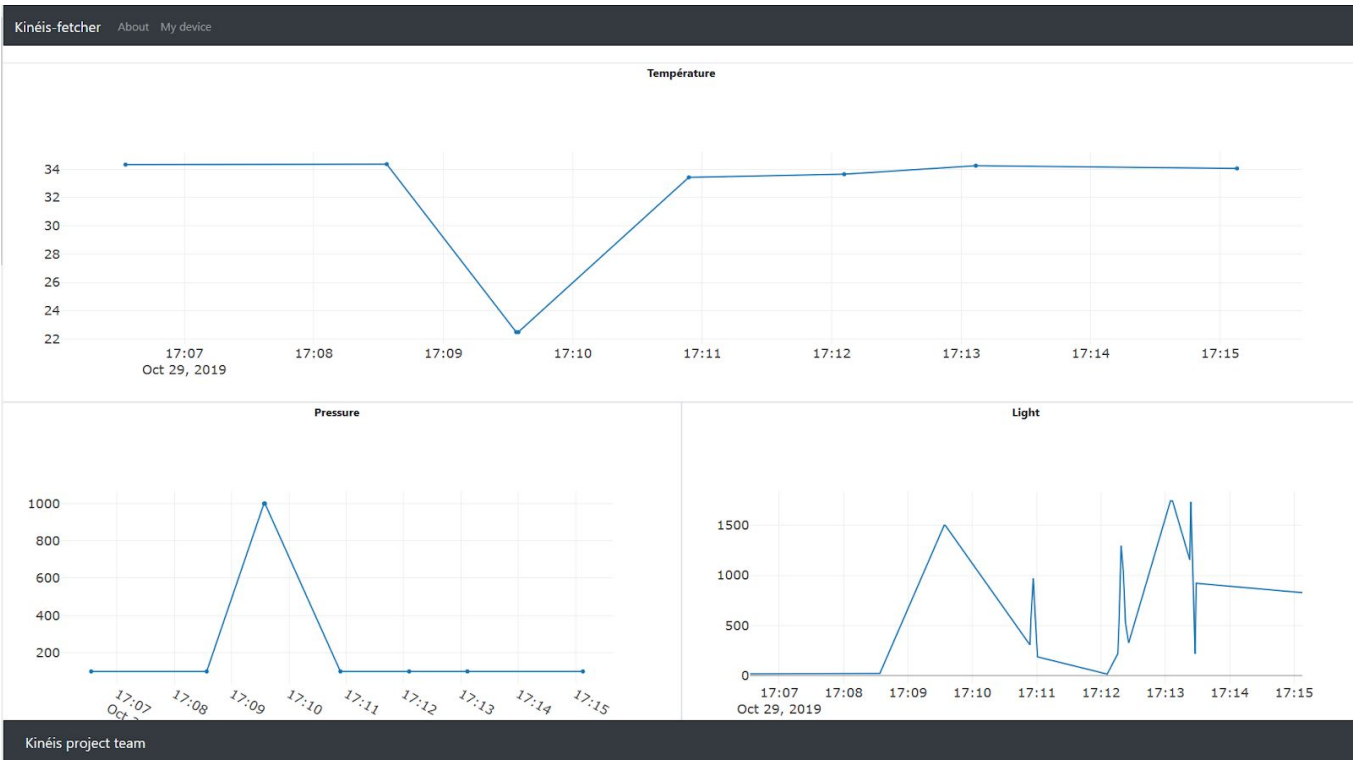

Fig.11 - Plots

We decided to focus on a responsive design, because we thought that it was really important that the user could access his data from any devices he wants.

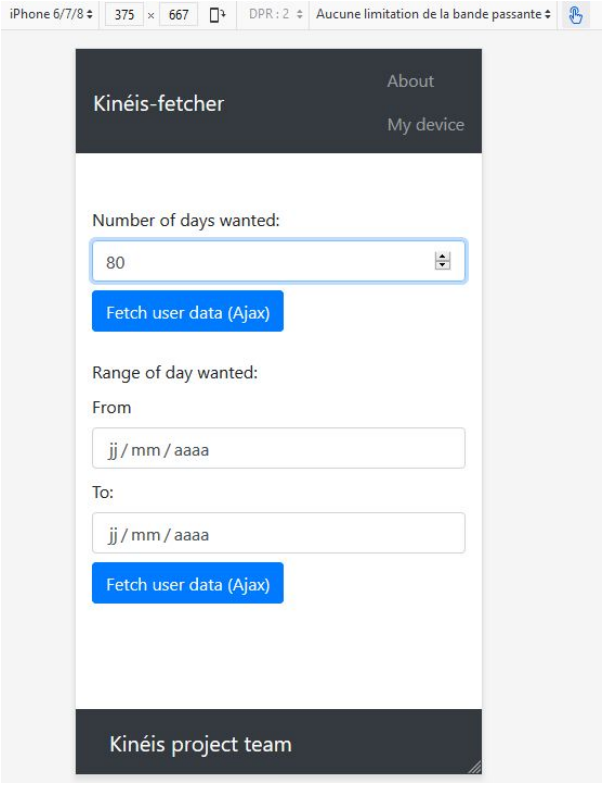

Fig.12 - Responsive design

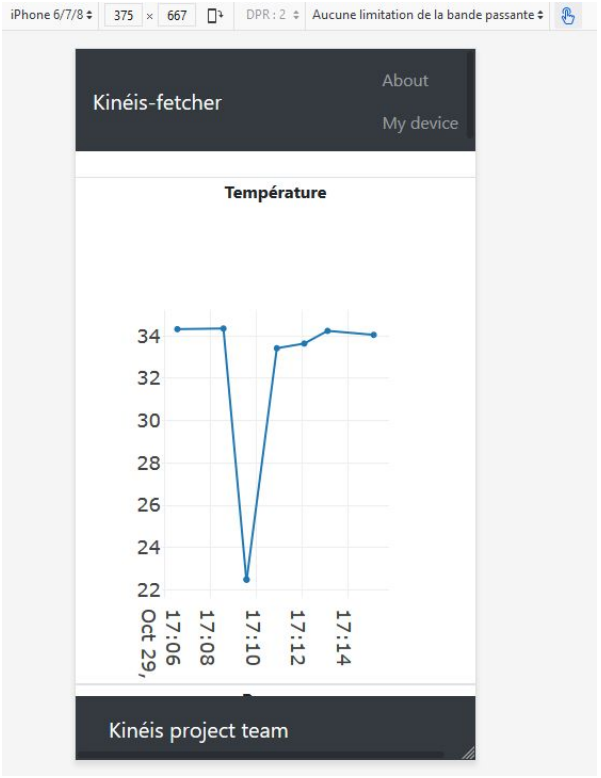

Fig.13 - Responsive plots

B. Data access

Each cloud exposes an API for fetching all desired data. When user clicks on one of the two blue buttons, it sends an asynchronous request (AJAX) to the front-end, asking for data for example via URL :
GET http://localhost:8000/?nbDays=80
GET http://localhost:8000/?dateFrom=2019-12-12&dateTo=2020-01-02

When Django receive this request, he calls a Python function which will use the exposed cloud's API.

AirVantage:
AirVantage provides a REST API, the documentation is available on [1].

A simple HTTP GET request on this URL allowed us to retrieve all the desired data.

```
sensor_data_raw_bulk = (
    "https://eu.airvantage.net/api/v1/systems/data/raw?targetIds=beb91e6f88984b27b8fca27506a85cc9&dataIds=MangOH.Sensors.Accelerometer.Acceleration.X,"
    "MangOH.Sensors.Accelerometer.Acceleration.Y,"
    "MangOH.Sensors.Accelerometer.Acceleration.Z,"
    "MangOH.Sensors.Accelerometer.Gyro.X,"
    "MangOH.Sensors.Accelerometer.Gyro.Y,"
    "MangOH.Sensors.Accelerometer.Gyro.Z,"
    "MangOH.Sensors.Pressure.Temperature,"
    "MangOH.Sensors.Pressure.Pressure,"
    "MangOH.Sensors.Light.Level&size=500")
```

An access token was needed to access the API, we didn't have trouble to set it up correctly. It is sent in the header of the request, like the traditional way of functioning of REST APIs.
We use the *requests* Python library to send HTTP requests easily.

Data are then sent back by AirVantage in JSON format. We reformat and sent them back to the front-end. The whole AirVantage processing is located inside the python module AVapi.py.

Argos:
Argos expose a SOAP API, the documentation is available on [2]. The WSDL file (it defines the set of service and message format used in a SOAP API) is publicly available on [3] . We used *zeep* Python module which is very efficient to manipulate SOAP requests.

As we saw in II.3., we designed our message format to match our needs. Therefore, we need to convert the raw 148 bits payload into real values and parse it in JSON in order to send it to the front-end.
We developed this function, which can take any type of raw data and convert it to a float value :

```
def convertToFloat(dataBinary,nbBits,startRange,endRange,name):
    if(len(dataBinary)!=nbBits):
        print("ERROR This data ("+name+") is encoded on " + str(nbBits) + " bits, received ", len(dataBinary))
        return -100
    intValue = int(dataBinary,2)# convert to base 10
    sensitivity = 2**nbBits / (endRange - startRange) # Sensitivity is defined as the number of bit divided by the amplitude of the input
    absScale = intValue / sensitivity  #convert to absolute scale, which go from 0 to 150 hPa for pressure
    relativeScale = absScale + startRange #convert to real scale, with the correct offset (e.g. 950 hPa to 1050 hPa for pressure)
    return relativeScale
```

It is located inside the Python module ArgosParser.py.

Unfortunately, we couldn't do the sensor data transmission via Argos SATCOM. So, some dummy data was crafted in order to test the integration of this function with the whole web site.

```
# Payload format : Light(12)|Pressure(10)|Temperature(14)|Accelerometer(16x3)|Gyroscope(16x3)|CRC(16)

# Light Example
# 3000 lumen = 2**12
# 1500 lumen = 2**15 (2**16 /2, to get the midrange)
Light = "100000000000" # 1500



# Pressure Example
# 1050 hPa = 2**10
# 950 hPa = 0
Pressure = "1000000000" # 1000 hPa

# Temperature Example
# 85°C = 2**14
# -45°C = 0
Temperature = "10000000000000" # 20°C 2**13

# Accelerometer Example (Same for the gyro)
# 2g = 2**16
# 0g = 2**15 (2**16 /2, to get the midrange)
# -2g = 0
Accelerometer = "1000000000000000" # 0g

# Example of decoding

payload = Light + Pressure + Temperature + Accelerometer*6 + "0"*16 # Plus 16 bits of CRC
msg_soap = {1578351600: payload, # Some timestamp
            1578361600: payload
            }
```

That's the reason you can see two points which are very strange in the front-end plots, because we took unusual values to distinguish the dummy data from the real one (from AirVantage).

C. Data merge

Once we retrieved the data from both AirVantage and Argos, we merged them. We placed the data from both the sources inside Python dictionaries and used the *update* method to merge them. As the *update* method doesn't work on nested dictionaries we developed a recursive function that can be found in views.py.

D. Data display

Once the data is merged, the merged data is sent to the AJAX function linked to the button. Further, the is given to the plotting library called *Plotly*. Documentation can be found here [4]. This library is supported on various languages, including Javascript. It allows fully interactive plots, where the user can zoom, highlight an area, download and edit the plot directly from his browser. jQuery and Bootstrap have been used to facilitate our front-end development.

E. Containerization of the website

In production, the HTTP Server created by Django is not suitable for real deployment. It is only a development server, not resistant against malicious intention and not really performance-oriented (it is slow) as stated in the documentation[5].

When the server part implementation got completed, we decided to improve the scalability of our solution and to make it more realistic & more suitable for a **real world deployment**.
To meet these needs, we decided to use Nginx, Gunicorn and Docker.

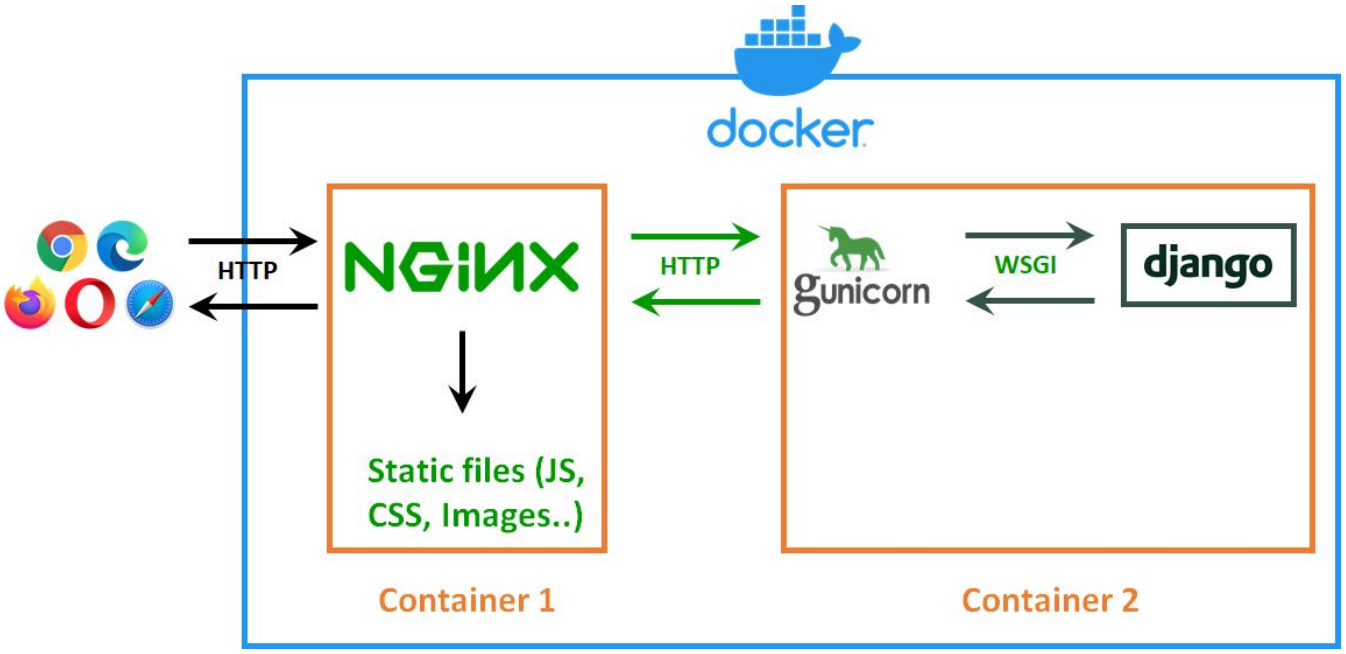

Fig.14 - Diagram of the installation

Gunicorn:
Gunicorn is a WSGI HTTP Server, responsible for handling HTTP requests and talking with our python application. Its role is not to serve static files (like images or css/js files). It is used only for translating the http requests to wsgi requests as an application server. It is designed to be fast and robust, and it works very well with Django. Since both of the software can run on Python, we placed them inside the same container as it uses the same docker image (Python3.8-Alpine).

Nginx:
Nginx is a reverse-proxy. Its role is to hide all the network behind him. It is exposed to the whole internet and is capable of handling a *lot* of requests. It is made to be robust and fast. Its role is also to serve static files to the user and to forward all other HTTP requests to Gunicorn. This setup means that no one can directly access the django application and the user has to go through Nginx first (but it is completely transparent for the user).

Docker:
Docker allows our whole website to run in a virtual environment, isolated from the host operating system. It is therefore more secure for the operating system. If an attacker succeeds to pierce through the web site defense, all the operating system's data will not be exposed. It also allows to package the web site into an image, and to share

it very easily to other people. We also could have used a virtual machine, but it is heavier, and much more resource intensive causing performance deterioration for the website.
Our website run in a multi-container environment which means that each container (Django/gunicorn and nginx) are isolated and built separately.

This whole setup can be seen as unnecessary for our project, **but we wanted to take a look at how things are done in the real world**. So we spent some hours to design this in order to improve the security, scalability and performance of our web server. Furthermore, sharing the website with Kinéis is made easier now because we just have to share the docker images of our project or to give them the docker recipe available on our Github [6].

IV. Energy Consumption Estimation

a) Background

To estimate the battery life of the Kineis IoT End Device, we used

- Assumptions on the transmission strategy, duration, no. of transmissions as found in table-2
- Power consumption and Energy Consumption data for SATCOM tests using the KIM-1 module. Energy consumption estimate in sleep mode and active transmission mode are found from table-3 and table-4 respectively. The values were referred from [8].
- Power consumption and Energy Consumption data for NB-IoT tests on the mangOH Embedded IoT card. Energy consumption estimate in sleep mode and active transmission mode are found from table-5 and table-6 respectively. The values were referred from [13].
- A sample battery specification is assumed and final battery life estimate was made as found in table-7.

b) Transmission strategy assumptions

Title	Value	Unit
Length of 3G emission	3	sec
Length of Argos emission	11	sec
Number of 3G transmission / day	2	
Number of Argos transmission / day	2	
Power Consumption while 3G transmitting	1.40	Watt
Power Consumption while Argos transmitting	0.02	Watt

In this table we have listed the assumptions that we have made to estimate the energy consumption and battery life.

Table.2 Energy Consumption Assumptions

c) ARGOS SATCOM Transmission

KIM-1 in SLEEP Mode

KIM-1 SLEEP MODE	Value	Unit
Current	0.001	Amp
Voltage	3.3	Vol
Power	0.0033	Watt
Duration/length of sleep in a day	86378	sec
Energy consumption in a day	0.07917983	Watt Hr/day

With reference to the KIM-1 SATCOM radio module datasheet, the current and voltage values are used to derive the power. Assuming on KIM-1 sleep duration energy consumption estimate is made.

Table.3 SATCOM Sleep Mode

KIM-1 in TRANSMISSION Mode

KIM-1 TRANSMISSION MODE	Value	Unit
Current	0.006	Amp
Voltage	3.3	Vol
Power	0.0198	Watt
Duration/length of transmission/day	22	sec
Energy consumption in a day	0.000121	Watt Hr/day

With reference to KIM-1 SATCOM radio module datasheet, the current & voltage consumed during transmission is used to derive the power. Based on table.1, transmission duration & energy consumption estimate is made.

Table.4 SATCOM Transmission Mode

d) Cellular NB-IoT Transmission

mangOH NB-IoT ULPM Sleep mode

mangOH SLEEP MODE	Value	Unit
Current	0.1	Amp
Voltage	3.7	Vol
Power	0.0005	Watt
Duration/length of sleep/day	86394	sec
Energy consumption in a day	0.01199917	Watt Hr/day

Assumption is made as mangOH card is in Ultra Low Power Mode (ULPM). Based on a academic research report on mangOH NB-IoT tests, average power value is referred and energy consumption estimation was made.

Table.5 mangOH NB-IoT ULPM Sleep mode

mangOH NB-IoT ULPM Transmission mode

mangOH 3G TRANSMISSION MODE	Value	Unit
Current	0.1	Amp
Voltage	3.7	Vol
Power	0.5	Watt
Duration/length of transmission/day	6	sec
Energy consumption in a day	0.00083333	Watt Hr/day

Based on power measurement tests documented in an academic research report, the average power value during NB-IoT transmission using mangOH card is found to be 0.5 watts. Accordingly, the energy consumption estimate is made.

Table.6 mangOH NB-IoT ULPM Transmission mode

e) Battery Life Estimation Results

Considering the above energy consumption estimations and assuming a standard battery and its energy of 28.86 wh, the battery life would last for 313 days.

BATTERY LIFE CALCULATION		
Total Energy consumption / day	0.09213333	Wh / day
Standard battery	28.86	Wh
Battery Life	**313.2417**	**day**

The total Energy consumption value is derived from energy values in table-2 to 5. Assuming a standard battery & it's energy, the battery life is estimated.

Table.7 Battery Life estimation results

Ⅴ. Network coverage

a) Cellular NB-IoT

URBAN Coverage

NB-IoT as a Low Power network fills the gap between the 'short-range' wireless network and the 3G/4G/LTE cellular network. The network coverage of NB-IoT totally depends on the availability of the 4G/LTE infrastructure. In the Urban areas, due to the widespread of 4G/LTE network infrastructure, the usage of NB-IoT is more suitable. The coverage ranges between 2 – 4 kms based on the number of transmission repetitions as found from [15].

Rural coverage

In case of the rural areas, the NB-IoT coverage can extend between 10 – 15 kms as found from [15]. But, since the 4G/LTE infrastructure is not wide spread across rural areas, for end-to-end connectivity, as a

gap filler, other LPWAN technologies such as LoRA & Dash7 are combined with NB-IOT for rural use-cases.

b) SATCOM Argos Coverage

Kineis uses the ARGOS polar orbiting satellites that are flying at the orbit of 850 km above the earth. The time taken by an ARGOS satellite to complete one revolution of earth is 100 minutes as found from [19]

At any given point, the visibility circle of a satellite where it can see all the ARGOS transmitters at earth will be approximately 5000 kms. The period during which the ARGOS satellite is available for an ARGOS transmitter in the visibility circle is approximately 10 minutes. The received Argos messages are stored on the onboard recorder and retransmitted to the ground each time the satellite passes over one of the three main receiving stations based on Wallops Island (Virginia, United States), Fairbanks (Alaska, United States), and Svalbard (Norway). Or, they are retransmitted to the regional reception stations in the satellite's field of view. There are nearly 70 such reception stations which forms the worldwide 'coverage network'.

Ⅵ. Applications/Use-Cases

Cellular NB-IoT

Sierra wireless WPxxxx module in the mangOH card has the support for 4G-LTE & NB-IoT.

NB-IoT suits better for the urban use-cases due to its support for low power, low latency and high dense IoT network as found from [15]. Some of the suitable use-cases are,

Pallet Tracking

Pallets as connected objects will be using the 'short-range wireless' technologies within the company premise. Once, the pallets are out of the company range, NB-IoT suits as a better technology for tracking.

Health-Care

Health-care/Elder care institutions and hospitals use NB-IoT to track their patients who are in remote places. Because, NB-IoT is deterministic in nature, it is preferred over other LPWAN technologies.

Smart Vehicles

In smart vehicles use-cases, mobility and downlink communication are critical requirements as the applications range from surveillance, passenger assistance services such as road navigation, weather maps. NB-IoT suits best as it is designed for both uplink and downlink communications.

Integrated SATCOM & NB-IOT

In this project, we have attempted to test the kim-1 SATCOM transmitter module in mangOH card as a external IoT module. The kim-1 SATCOM module uses the ARGOS satellites which is based on LEO SATCOM transmission technology. As we had discussed above an ARGOS satellite will have a very wide coverage of 5000 kms and provides a visibility period of 10 minutes for a kim-1 transmitter module. This enables to use the Kineis integrated IoT device (ARGOS + Sierra Cellular) to be deployed in a variety of use-cases that needs wide coverage and reliable communication at low power, low cost. Hereby, we would like to present some of the use-cases as found from [14].

Telemetry and Tracking Devices

- International Cargo/Asset Tracking & Geo Positioning
- Wild-life tracking & Monitoring
- Environmental monitoring sensors
- Energy and water management
- Integration with smartphones/tablets

Alarm Devices

- Emergency and distress signals
- Theft prevention
- Geofencing

VII. Issues

Our team members are coming from different countries with different cultures, which caused some issues with communication and group dynamics.

At the beginning, due to the different backgrounds, we met some obstacles when selecting the representative of our group. In some cultures, it's not a big deal because everyone is working on this project, while it's so important in other cultures with the reason that it means who will take the responsibility to charge the whole group.

But the point is that we understand that different cultures cause misunderstandings, which makes us treat these conflicts correctly and solve these problems successfully.

There were also some understanding issues, at the beginning of the project, we thought that the project should be divided into three parts, hardware, software, and 3d printing. And after our first meeting, the group was divided into hardware part and software part. It was not until the later part of the project that we found that the core of the project was the switching mechanism, and because of that, our time was wasted a lot. We met big troubles concerning the technical deployment of the SATCOM communication, we received the final KIM-1 module in mid December, therefore the time frame was really short to be able to start developing with the module's API. Once we received the module, important compilation errors prevented us from doing anything, it leads to long exchanges between the Kinéis's developer team and us.

VIII. Conclusion

In this final section, we will begin with revisiting the project tasks and our prototype demo status against each task. Then, we will present our key results in each subsection of the project namely – mangOH, NB-IOT & SATCOM transmission and broker server architecture.

S.No	Project tasks	Demo Status
1	Access mangOH Sensors & data retrieval	Demo Ready
2	Transmitting sensor data via cellular network	Demo Ready
3	Transmitting sensor data via SATCOM network	Pending
4	Switching between Cellular & SATCOM	Pending
5	Receiving data from Cellular (AirVantage) & Store in Server	Demo Ready
6	Simulation of receiving data from SATCOM (ArgosWeb) & Store in Server	Demo Ready
7	Process the data & plot as graphs & reports	Demo Ready
8	3D Print of Prototype	Pending

mangOH Part (Embedded hardware)

In our attempt on solving the challenge of accessing the sensor data and retrieving them, we succeeded in locating where in mangOH hardware the sensor data is stored and how to access them. Further, the next challenge we succeeded was encoding the sensor data as required by transmission message formats. For the transmission network switching between SATCOM and Cellular, how to develop a Legato application was figured out but implementation could not be possible due to the unavailability of certain key components of the project.

Communication technologies (LEO SATCOM/NB-IoT/3G)

The sensor data transmission from mangOH end-device to end-user reporting server was tested using 3G/NB-IoT communication technologies successfully. For SATCOM testing, though transmission via the Kineis KIM-1 module was not possible as of now, we made simulation tests using Kineis provided temporary transmission application and found the end-user reporting to be successful.

Energy Consumption Estimation

One of the main criteria for the mangOH end-device is that it has to be very efficient in energy consumption. With that goal, we made assumptions on transmission strategy (number of times, duration), data aggregation strategy, transmission modes (sleep duration, active duration) and accordingly estimated the power consumption and energy consumption. As per our estimation, we inferred that a standard 28.8 Wh battery could provide a life of around 313 days.

Server Part (Cloud/data analysis/Reporting)

We successfully retrieved the data from the cellular network data cloud and SATCOM network data cloud, merged them, and displayed them as reports for the user. We made a responsive and fast interface which provides a comfortable user's experience. Thanks to Docker, Nginx, Gunicorn and Django, we managed to develop a scalable, secure, and responsive environment which is suitable for real world deployment.

Through the work so far, we have built some of the main blocks in the Kineis mangOH multiprotocol prototype. It can serve as a base to accelerate the development of remaining blocks that are - communication network switching, performance optimization, 3D printing.

REFERENCES

[1] https://source.sierrawireless.com/airvantage/avc/reference/cloud/API

[2]http://www.argos-system.org/wp-content/uploads/2016/09/r1626_9_argos_webservices-1_7.pdf

[3] http://ws-argos.cls.fr/argosDws/services/DixService?wsdl

[4] https://plot.ly/graphing-libraries/

[5] https://docs.djangoproject.com/en/3.0/ref/django-admin/#runserver

[6] https://github.com/Xt-X/Project-of-Kineis

[7] mangOH Red Developer guide Version 4

[8] Kim-1 Expansion card User manual v0.2

[9] https://mangoh.io/

[10] https://legato.io/

[11] https://doc.airvantage.net/av/reference/cloud/API/

[12] https://source.sierrawireless.com/airvantage/avc/howto/cloud/gettingstarted_api/

[13] Energy Consumption of Low Power Wide Area Networks, Sebastian Nyberg,˚Abo Akademi University, Spring 2018

[14] On the Satellite Role in the Era of 5G Massive Machine Type Communications, European Space Agency, September 2018

[15] Internet of Mobile Things: Overview of LoRaWAN,DASH7, and NB-IoT in LPWANs standards and Supported Mobility, Jun 2018

[16] Pressure/Temperature BM280 : https://cdn-shop.adafruit.com/datasheets/BST-BMP280-DS001-11.pdf

[17]Gyro/Accelerometer

BMI160: https://www.mouser.com/datasheet/2/783/BST-BMI160-DS000-07-786474.pdf

[18] Light PNJ4K01F : https://industrial.panasonic.com/content/data/SC/ds/ds4/PNJ4K01F_E.pdf

[19] http://www.argos-system.org/applications-argos/

[20] KINEIS-MU-19-0060 - Message format and distribution manual rev 1-0

ABSTRACTS

Leo's abstract:
The emergence of low-power embedded systems like the MangOH project allowed to face the specific needs of Internet Of Things (IoT) networks. These systems also need low-power transmission techniques to maintain their battery life. We decided to use the new 5G LPWAN networks in conjunction with a LEO low power satellite link. The aim of this project is to develop a system capable of switching between the two technologies when needed, with the maximal battery life possible. To achieve this, it is needed to develop a Legato application on the embedded system with the right emission strategy to ensure an optimal power consumption. Space constraints are also present since this embedded system needs to be as little as possible, so the battery choice needs to be optimal. Furthermore, an interface for the final user is also needed to display the sensor data. Results show that the implementation of this kind of system is possible on a MangOH card, thanks to the KIM-1 expansion card brought by the Kinéis team and the built-in CF3 3G module. Unfortunately errors prevented us to finalize the satellite communication on the prototype. However, we successfully implemented a Django web application coupled with Nginx and Gunicorn to retrieve, merge, and display the data sent by the MangOH.

Huijie's abstract:
Nowadays, satellite is widely used in communication industry, but it's not always available. Our project aims to develop a system whose data transmission method can be switched between satellites and cellular network with MangOH-red board and Kim1 module. We will use the integrated Legato application within MangOH card to achieve the communication via cellular network and the use the Kim1 application to send messages via the satellites. When the things above are realized successfully, we will develop an application that combines the two transmission methods. And then, we will develop a website that can appear our retrieved data. For now, we succeed in transmitting our messages via satellite and cellular network separately and building our website. But unfortunately, due to some technical problems, we can not combine these two transmission way into one application and switch between them so far. During this project, all of us has practiced our technical skills and learnt a lot about group collaboration and time management. Although we don't finish what we've expected at first, we have tried our best and showed our teamwork when facing different problems.

Zhe's abstract
Satellite, as one of the most currently stable communication technologies have many advantages, such as long distance of communication. But satellite is not always available (over our device), this is why we do this project. This project aims to do an embedded application that collects the data of our sensors such as Temperature, location, pressure, etc and allows an automatically switching mechanism between NB-IOT and satellite, the goal is to provide the connection all along the day. Apart from this, we should develop a front-end application which show the data of our sensors by figures. For achieve the objective of our project, we divide our group into two sub-group. Zhe and Huijie take responsibility of the embedded application, Leo, Xuantang and Mohan take the server part and chose the battery. For each part of the project, we developed separately the application for NB-IOT and satellite. After 3 months of work, we finished the server part, the front-end application can retrieve the data from both platform of NB-IOT(AirVantage) and satellite (Argos). For the embedded application, we can send the data to AirVantage, but until now, the module of communication with the satellite from Kineis company still doesn't work. So, we can not send the data to Argos and the automatically switching doesn't work either. We are waiting for the final module of Kineis company and still working on it. We show the details of our work in the following network, and we describe all the issues that we met during these three months.

Mohan's Abstract

Kineis, a New Space IoT Network Operator wants to create a converged IoT End-device with transceiver capabilities of both terrestrial cellular/LPWAN and SATCOM protocols. We attempted to develop a prototype based on MangOH to validate the multiprotocol IoT device concept. The goals are to validate the automatic selection of network on the basis of availability of terrestrial or SATCOM network and to validate the sensor data collection, transmission and reporting to the user. At mangOH, we used Legato, a Linux based embedded application development framework for implementation of sensor data collection. For data transmission, we could validate the cellular (3G) network only. A Django/python based server implementation was used to store, process and report the data to the end-user. We succeeded in sensor data collection, transmission using cellular network and reporting to the user using charts & visuals. We did succeed in encoding the sensor data in Kineis ARGOS format for SATCOM transmission. In our attempt to develop a strategy that delivers the longest battery life, we inferred that, with a standard 28.8 wh battery, approximately 313 days of battery life can be achieved. Once the dependencies/barriers related to key components are resolved by Kinéis team the automatic network selection feature can be developed & tested.

Xuantang's abstract

Internet of Things will bring a lot of convenience to people's lives while MangOH is a development board based on it by NB-Iot network. For more stable data transmission and more precise location, we decide to apply the KIM-1 module to connect it to the network of satellite. The aim of our project is to provide an optimal connection to transfer the data from MangOH based on NB-Iot and satellite, then visualize the data on the client end. At the part of the MangOH, we use the Legato application to compile the system . As for transmission, AirVantage platform is used for NB-Iot network while Argos is used for satellite module KIM-1**.** FInally, we were able to successfully receive data from two different platforms, and we also completed the part of the web application by Django. But for transmitting data to satellites, we still have some technical problems, which is what we will solve next.