Måns Oskarsson mnsosk-7@student.ltu.se

SELF-SUFFICIENT SMART HOUSE

INTRODUCTION

As a part of the university course D7042E IoT-based industrial automation and digitalization the student is given the project to design their own use case scenario which should represent and simulate how a real credible application system would be integrated into the arrowhead framework.

The project must have a realistic approach on how the dataflow work. Meaning it must have some realism and coherence in the scenario.

The project must follow certain criteria's, it must be integrated into at least one arrowhead local cloud.

This arrowhead local cloud must include the mandatory core systems built in arrowhead, which is the service registry, orchestration and authorization system.

It also must consist of four different application systems. Which means that at least four systems must provide or/and consume functionality inside the arrowhead local cloud.

A working demo is important. Which will be showcased at the end of the course.

SCENARIO

The scenario in my case that I have created will be energy consumption/production for an energy self-sufficient smart house.

The simulation and scenario will be in real time. Meaning one second of execution time will represent one day of scenario time.

The scenario would consist of four different application systems:

- Solar panels that will be generating electricity.

  The simulation for this will be real-time which means that the kw/h generated will change depending on an efficiency factor of how efficient the solar panel was during the day (night/day and cloudiness).

  In this case the static variable for how much one solar panel generates per day on average is 2 kW/h.

  For the terms of simplicity, the efficiency factor will be randomized between a value of 0.7 to 1.2. Meaning on average the solar panels will have an efficiency rate of 0.95 each day.

  The total amount of solar panels can be determined during execution of simulation. The total amount of solar panels will be chosen so it sometimes will undercut and overcut the house daily consumption. Not so that every day will result in an overproduction of electricity.

$$Daily\ kw/h\ = 2\ kw/h * Total\ amount\ of\ solar\ panels * Efficiency\ factor$$

The solar panel system will update the efficiency factor after the house has sent a GET request to the solar panel, meaning a day has passed.

- A smart house that will be receiving electricity from the solar panels which provides electricity.

The smart house will have an energy storage, which is the total kw/h accumulated from the solar panel.

$$Current\ electricity\ stored = Current\ electricity\ stored + Daily\ kw/h$$

The smart house will have a wallet with money balance, a number which represent the dollars gained from selling excess energy to the electricity grid.

The smart house will also have a variable which decides how much of excessive energy it will sell. Ranging between 0-100%. In this case a value of 40% will suffice.

$$Shipped\ kW/h = (Current\ electricity\ stored - Total\ electrical\ consumption) * Sell\%$$

If the house is consuming more than it's producing it will lose balance since the value of shipped kW/h will be negative. Hence the household will lose money and possibly be in debt if money balance is negative. Being in debt in this case means that the household will own the electricity grid money (the public).

- An electricity grid, which is an application system that makes it possible to sell electricity to the public electricity grid from the smart house.

Which rewards the household with some kind of money balance depending on price per kW/h and the current demand. The average price per kW/h in my simulation will be 0.11$, a value chosen from a site where electricity prices ranges in different U.S States.

$$Money\ gained = 0.11 * (Shipped\ kW/h) * demand$$

- Outlets/consumption, outlets for the smart house which makes it possible to charge and use electrical devices in the house.

This is the current electricity consumption. On average household consumes around 26-33 kW/h per day.

The sources which values are taken from can be taken by a grain of salt but since this is a simulation, the requirement for the simulation values is that they are consistent and somewhat realistic.

LOCAL CLOUD

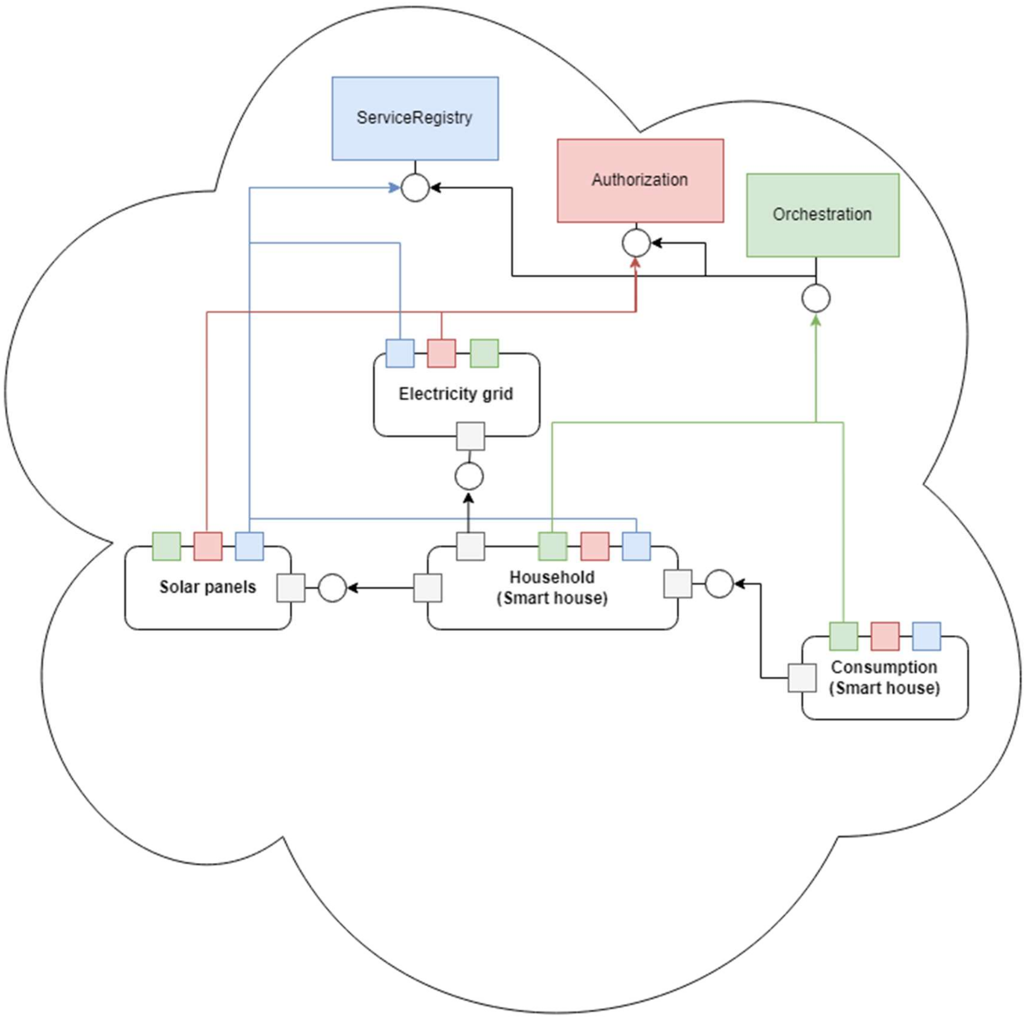

FIGURE 1: LOCAL CLOUD

DATAFLOW

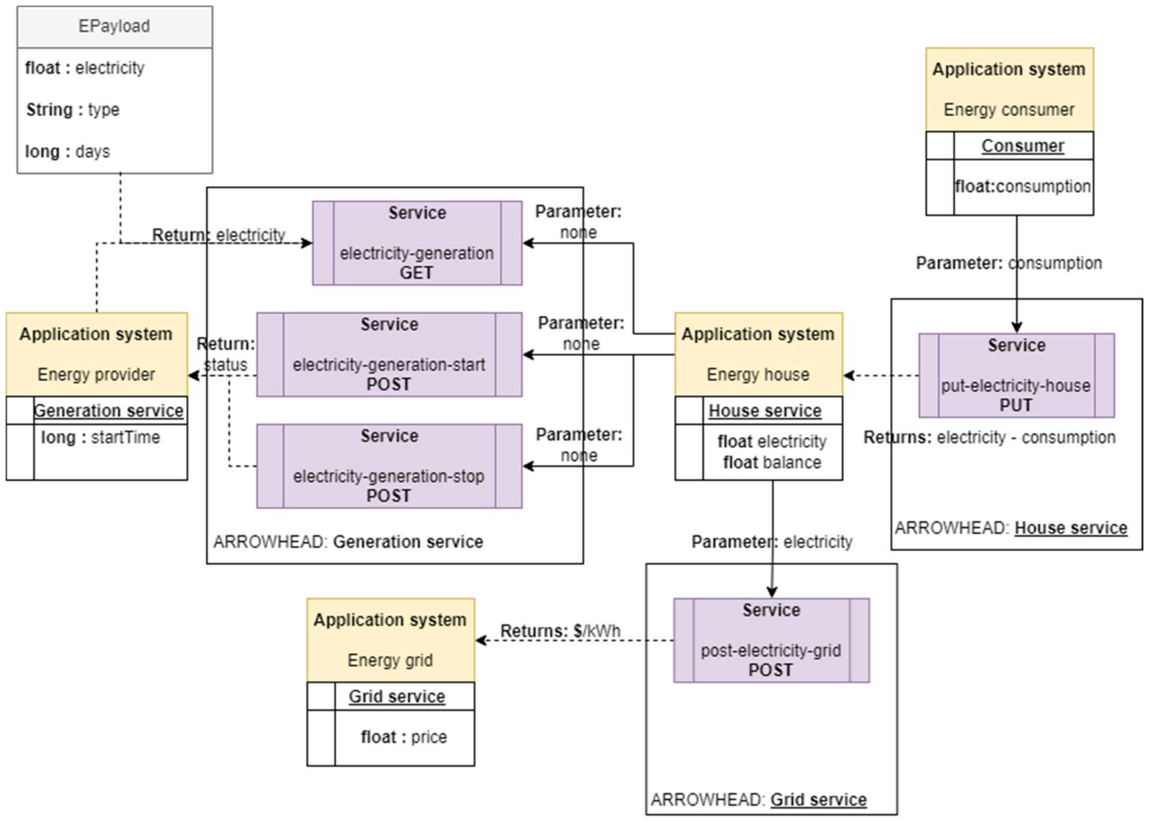

FIGURE 2: DATAFLOW OF LOCALCLOUD

SoSD

USE CASES:

Name of the Use-case
ID: 1
Brief description: Sell to the electricity grid to earn money.
Primary actors: Household with excessive energy.
Secondary actors: Grid, Energy market. Fortum, E.ON, Skellefteå Kraft.
Preconditions: Household having internal storage of energy.
Main flow:

Present in a sequence of steps the interactions among the actors 1- Household has internal energy storage. 2- Household sell to their assigned electricity grid (electricity company). 3- Household receives money. ………
Postconditions: Have wallet which can store money,
Alternative flows: -

Name of the Use-case
ID: 2
Brief description: Buy electricity from the market.
Primary actors: Household consumer with no excessive energy or internal battery storage.
Secondary actors: Grid, Energy market. Fortum, E.ON, Skellefteå Kraft.
Preconditions: Household consuming more that its producing. Or no production at all.
Main flow: Present in a sequence of steps the interactions among the actors 1- Household consuming electricity. 2- Household buy electricity from the electricity grid (market). 3- Household receives debt for buying electricity. ………
Postconditions: Have money balance that loses money based on amount bought.
Alternative flows: -

Name of the Use-case
ID: 3
Brief description: Electricity generation source which can provide generation service.

Primary actors: Wind turbine, solar panels.
Secondary actors: Household consumer/s which accumulates from electricity source.
Preconditions: Electricity generation source with internal accumulation.
Main flow: Present in a sequence of steps the interactions among the actors 1- Electricity generation source which accumulates electricity. 2- Household send request to generation source. 3- Household receives electricity from generation source. ………
Postconditions: Household having internal storage of energy.
Alternative flows: -

Name of the Use-case
ID: 4
Brief description: Collection of consumer devices which can consume from household.
Primary actors: Consumption devices. Telephones, charging devices, kitchen.
Secondary actors: Households.
Preconditions: Electricity generation source with internal accumulation
Main flow: Present in a sequence of steps the interactions among the actors 1- Electricity generation source which accumulates electricity. 2- Household send request to generation source. 3- Household receives electricity from generation source. ………
Postconditions: Household having internal storage of energy or access to electricity grid (market).
Alternative flows:

-

SYSD

PRODUCED SERVICES:

Service	IDD
electricity-generation	Energy provider **Service 1**
electricity-generation-start	Energy provider **Service 2**
electricity-generation-stop	Energy provider **Service 3**
electricity-generation-time	
post-electricity-grid	Energy grid **Service 1**
put-electricity-house	Energy house **Service 1**

CONSUMED SERVICES:

Service	IDD
electricity-generation	Energy provider **Service 1**
electricity-generation-start	Energy provider **Service 2**
electricity-generation-stop	Energy provider **Service 3**
post-electricity-grid	Energy grid **Service 1**
put-electricity-house	Energy house **Service 1**

ENERGY PROVIDER (SD/IDD)

- Protocol: HTTP(S).
- Encoding: JSON.
- Compression: None.
- Security: Optionally using TLS and X.509 certificates (server/client).
- Accessed at http(s)://127.0.0.1:8880/electricity-generation/<extension>.

SERVICE 1: ELECTRICITY-GENERATION

- Data model is JSON.
- No payload encryption.

Function	Service	Extension	Method	Input	Output
getGeneratedEletricity()	electricity-generation	**-**	**GET**	-	EPayload

DESCRIPTION:

Returns generated electricity made from the application system "Energy provider" which consists of solar panels producing electricity depending on time passed.

PARAMETERS:

This interface doesn't take any parameters.

RESPONSE CODE:

Code	Meaning	Comment
200	Successful request	Success
401	Unauthorized	Access denied
400	Bad request	If electricity production hasn't started.

ERROR HANDLING

All errors are handled using HTTP response code. Error message is added in the response payload.

When electricity production hasn't started, status code 400 is returned with a message stating that electricity production hasn't started.

OUTPUT

Returns payload consisting of generated electricity.

Example: {EPayload : ["electricity": 34.61, "days": 4:, "type": "kW/h"] }

INTERACTION WITH CONSUMERS

When a consumer requests total generated electricity. After successful request all accumulated electricity will go back to zero.

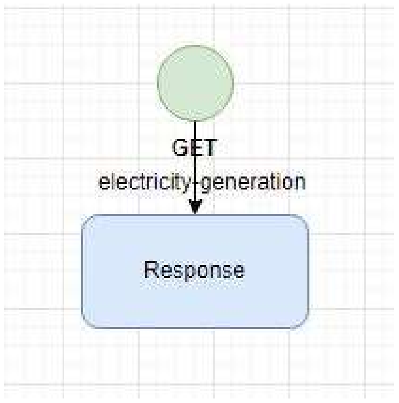

FIGURE 3: ELECTRICITY GENERATION INTERFACE

SERVICE 2: ELECTRICITY-GENERATION-START

- Data model is plain text.
- No payload encryption.

Function	Service	Extension	Method	Input	Output
startGeneratedEletricity()	electricity-generation-start	start	**POST**	-	String

DESCRIPTION:

Starts electricity production of solar panels. Returns String stating that production has started.

PARAMETERS:

This interface doesn't take any parameters.

Response code:

Code	Meaning	Comment
200	Successful request	Success
401	Unauthorized	Access denied
400	Bad request	If production already have been started.

Error handling

All errors are handled using HTTP response code. Error message is added in the response payload.

Output

Returns String with status code of electricity production.

Example: { " Solar panels started " }.

Interaction with consumers

When a consumer starts sends a start request. Application system starts generating electricity.

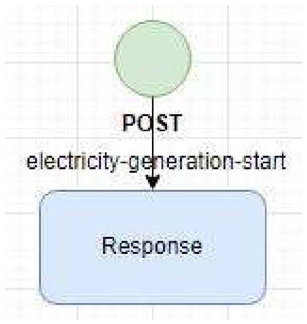

FIGURE 4: ELECTRICITY GENERATION START INTERFACE

Service 3: electricity-generation-stop

- Data model is plain text.
- No payload encryption.

Function	Service	Extension	Method	Input	Output
stopGeneratedEletricity()	eletricity-generation-stop	stop	**POST**	-	String

Description:

Stops internal electricity production of solar panels. Returns String stating that production has stopped.

Parameters:

This interface doesn't take any parameters.

Response code:

Code	Meaning	Comment
200	Successful request	Success
401	Unauthorized	Access denied

400	Bad request	If production already have been stoppped.

Error handling

All errors are handled using HTTP response code. Error message is added in the response payload.

Output

Returns String with status code of electricity production.

Example: { “ Solar panels stopped “ }.

Interaction with consumers

When a consumer starts sends a stop request. Application system stops generating electricity and wipes all generated electricity.

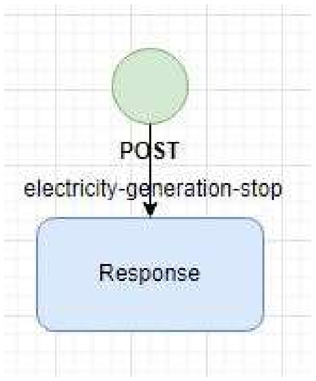

FIGURE 5: ELECTRICITY GENERATION STOP INTERFACE

Energy grid (SD/IDD)

- Protocol: HTTP(S).
- Encoding: JSON.
- Compression: None.
- Security: Optionally using TLS and X.509 certificates (server/client).
- Accessed at http(s)://127.0.0.1:8882/electricity-grid/

Service 1: POST-ELECTRICITY-GRID

- Data model is plain text.
- No payload encryption.

Function	**Service**	**Method**	**Input**	**Output**
money()	post-electricity-grid	**POST**	String	String

Description:

Takes a number as plain text. Returns total sold amount of electricity based on parameter and what current price of electricity is in application system.

Parameters:

This interface takes the following query parameters.

Parameter	**Usage**	**Example**
electricity	The amount of electricity that should be sold in kW/h	Electricity=23.61 will return 23.61 multiplied by the price of current cost of kW/h.

RESPONSE CODE:

Code	**Meaning**	**Comment**
200	Successful request	Success
401	Unauthorized	Access denied
400	Bad request	If parameter isn't a number.

ERROR HANDLING

All errors are handled using HTTP response code. Error message is added in the response payload.

OUTPUT

Returns String with total amount money earned from selling.

Example: { "5.223"}.

INTERACTION WITH CONSUMERS

When a consumer starts sends a sell request to the service. The application system will calculate the money the consumer will receive by multiplying the modelled price with the electricity from the consumer. Many consumers can use this service as a way as selling their excessive electricity.

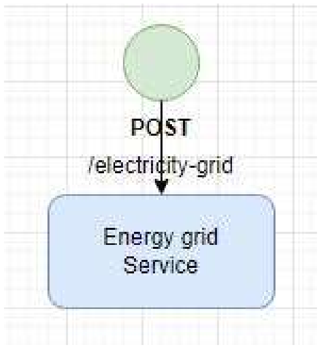

FIGURE 5: ELECTRICITY GENERATION STOP INTERFACE

ENERGY HOUSE (SD/IDD)

- Protocol: HTTP(S).
- Encoding: JSON.
- Compression: None.
- Security: Optionally using TLS and X.509 certificates (server/client).
- Accessed at http(s)://127.0.0.1:8881/electricity-house/

SERVICE 1: PUT-ELECTRICITY-HOUSE

- Data model is plain text.
- No payload encryption.

Function	**Service**	**Method**	**Input**	**Output**
updateBattery()	put-electricity-house	**PUT**	String	String

DESCRIPTION:

Takes a number as plain text. Updates the household internal electricity variable with the consumption from the parameter.

PARAMETERS:

This interface takes the following query parameters.

Parameter	**Usage**	**Example**
electricity	Adds electricity to the household.	Electricity=-20.6 will add this value to the application system internal battery.

RESPONSE CODE:

Code	**Meaning**	**Comment**
200	Successful request	Success
401	Unauthorized	Access denied
400	Bad request	If parameter isn't a number.

ERROR HANDLING

All errors are handled using HTTP response code. Error message is added in the response payload.

OUTPUT

Returns String with net change after adding consumption to electricity of house.

Example: { "25.7"}.

INTERACTION WITH CONSUMERS

When a consumer starts sends a consumption request to the service. In this case the household service the battery will be updated with the consumption parameter. After this the household will also send a request to the electricity provider service which will give a full update to the battery in the household application system.

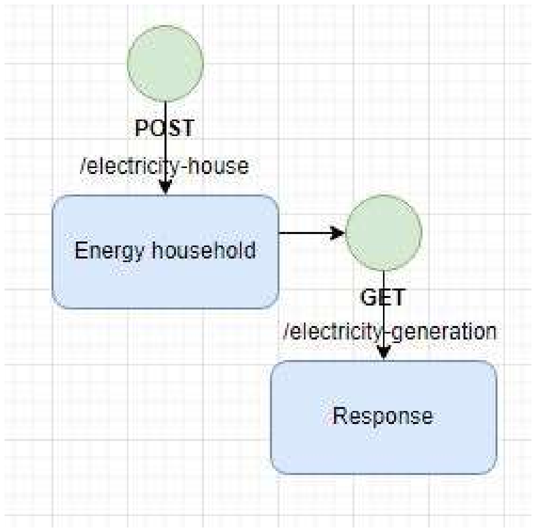

FIGURE 5: ELECTRICITY HOUSEHOLD INTERFACE

ERRORS OCCURRED DURING DEVELOPMENT

- Bad windows support, errors when running in windows sub system (Linux in windows). MySQL and arrowhead initialize script didn't run properly which resulted in the need of manually editing the MySQL database when creating tables and rules for the different core and support system. Running MySQL through WSL as arrowhead database is highly impractical and caused many conflicts with database. However usually a pure UNIX development environment is probably used instead which will result in no errors.
- Tracking small errors, which is avoided of course by having good tests. But because of the nature of Java and arrowhead being such a large framework. When errors occurred, they could be sometimes hard to track.

SOURCES

Project: https://github.com/MansOsk/D7042E_IoT

Arrowhead framework: https://github.com/arrowhead-f/