Factory of the Future

2MAE501: Architecture and Programming project

Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO

Table of Contents

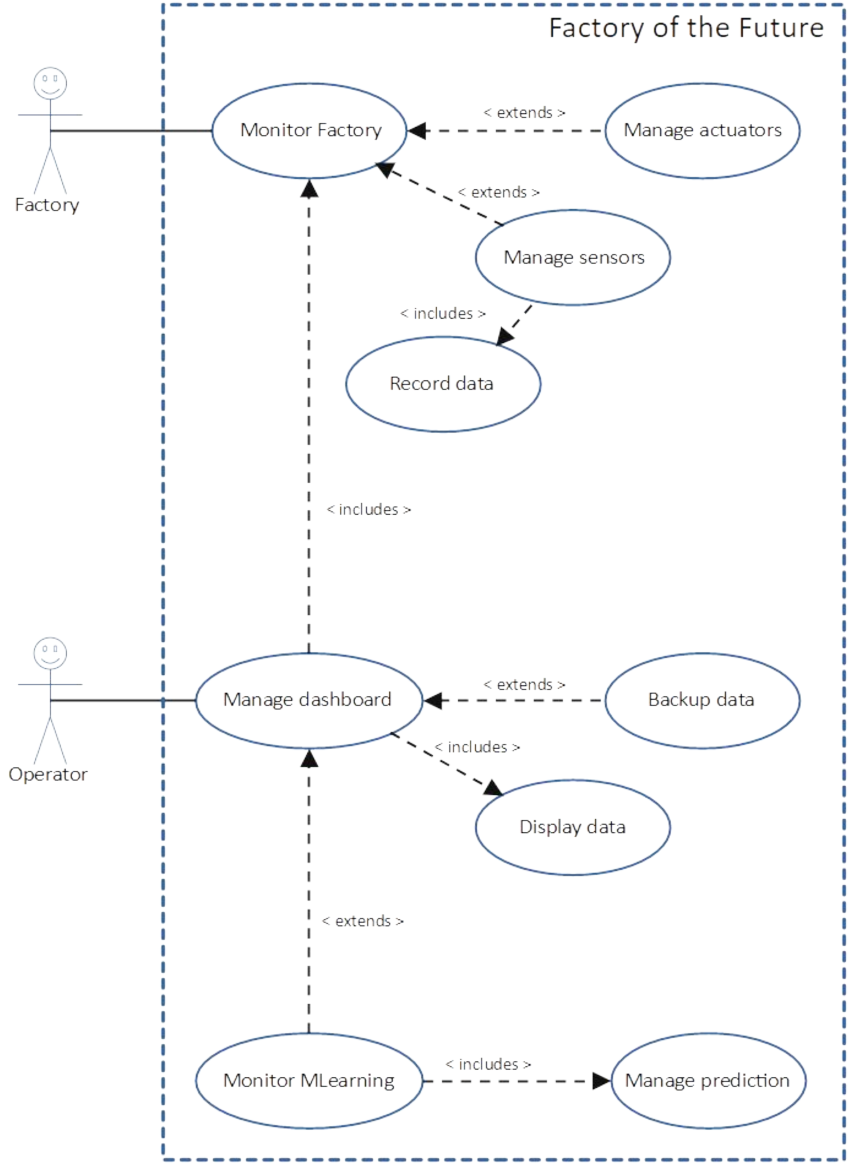
All code available at *github.com/guipenedo/factory-of-the-future/*

General Overview

Requirements

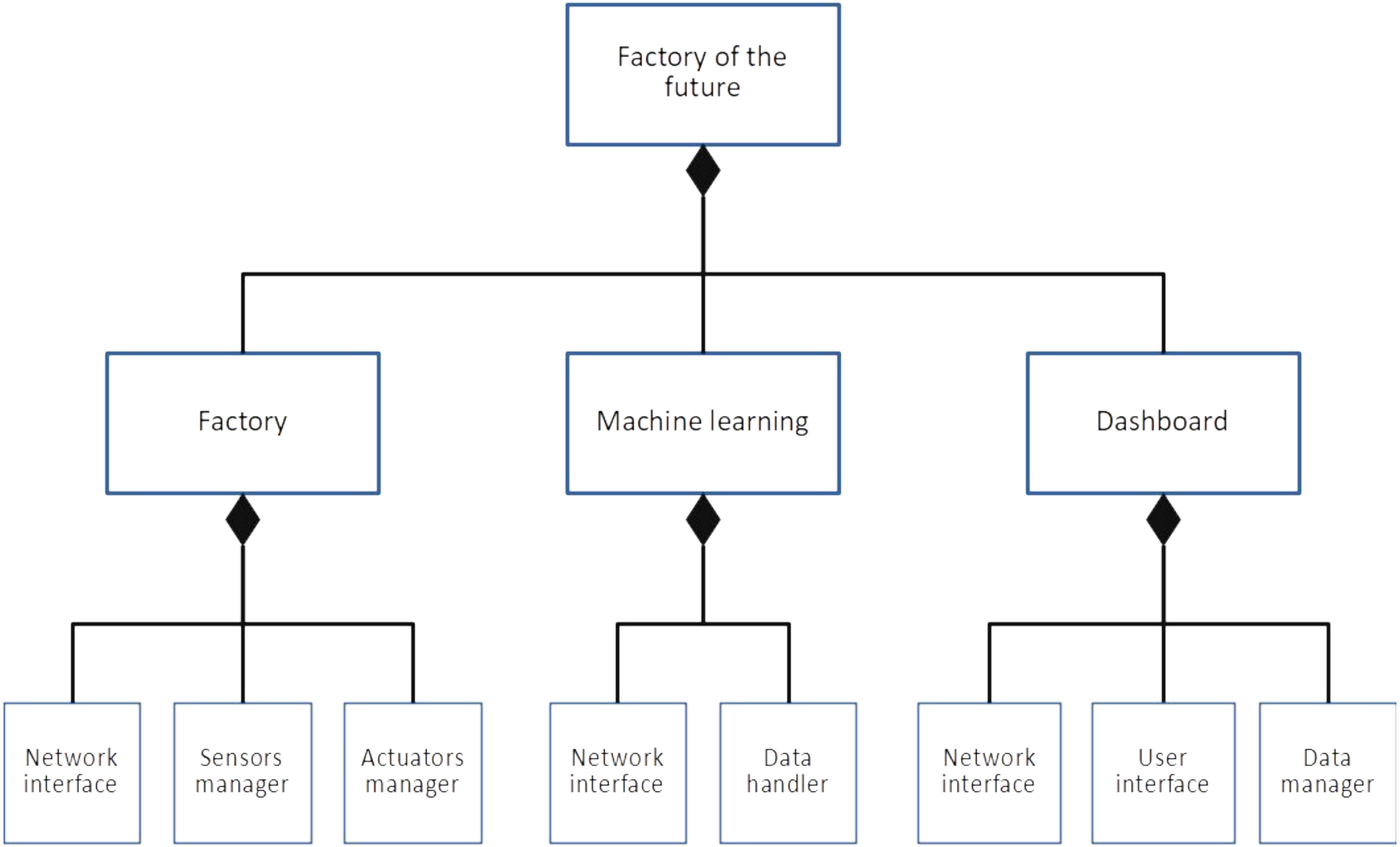
ID	Requirements	Subsystem
1	Sensors and actuators shall be driven remotely.	Network Infrastructure
2	Alarms shall be triggered in case of threshold violation and last for 60 seconds.	
3	Alarms shall be broadcasted over the network.	
4	Factories shall broadcast sensor data each 5 seconds.	
5	Factories shall implement at least a temperature/pressure/humidity sensor.	Factory
6	Factories shall be capable of giving their list of available sensors/actuators.	
7	Factories shall implement at least an LED and a relay.	
8	Dashboard shall store received data.	Dashboard
9	Dashboard shall display sensors data.	
10	Threshold on the measurement shall be adjusted.	
11	Dashboard shall display current factories status (actuators, sensors, alarm).	
12	Machine learning usage shall provide forecastings about factories sensors values.	Machine Learning

Use-Case Diagram



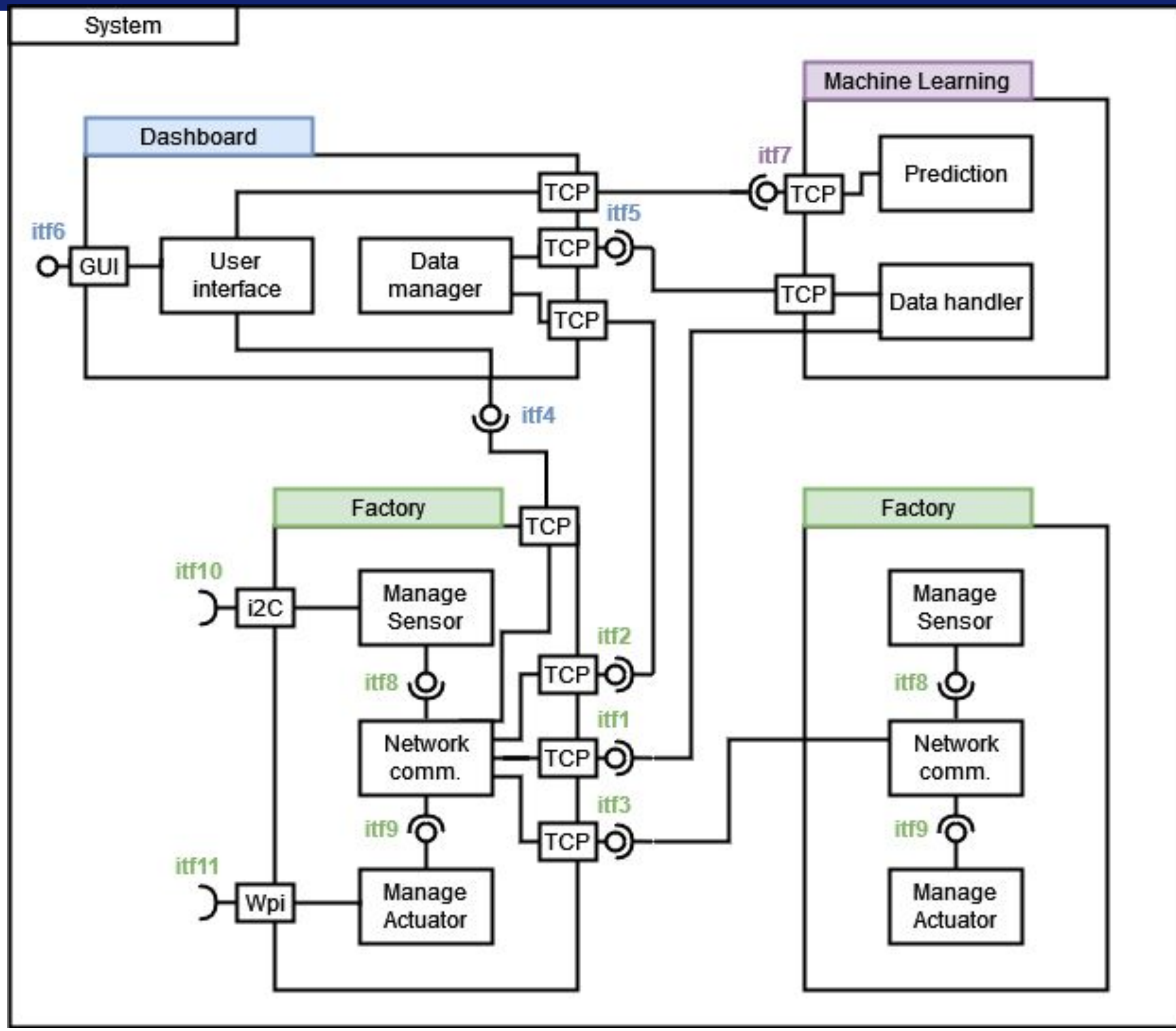
Use case diagram of the factory of the future

Block Diagram



Block diagram of a factory of the future

Internal Block Diagram



Internal block diagram of the project

Interfaces list

ID	Provided to	Purpose	Subsystem
N/A	Any	send close comm. command from a server to all its clients	All
itf1	ML	send sensors value history	Factories interfaces
itf2	Dashboard	send sensors/status datas	
itf3	Factory	send sensors data	
itf4	Factory	new host init; send commands (alarm, led, relay); get sensor and actuators list	Dashboard interfaces
itf5	ML	send commands (initialisation, ask for prediction)	
itf6	Operator	G.U.I. interactions	
itf7	Dashboard	send prediction for a specific factory	Machine Learning interfaces
itf8	Sensor man.	send request for sensor data	Network communications
itf9	Actuator man.	send order to drive the GPIO	

Network

Guilherme, Naël & Yannick

Network - Main Principles

The network provides a general and **modular interface** used by every actor
-> The factory of the future can be **easily scaled-up**

Every actors is connected by a **client/server** bridge

The **dashboard** is the "connection enabler"

Packets are strings formatted like so: "**command id**" + "**args**"

- **Args** size is variable
- **Spaces** are used as separators between each argument

Communication is always **synchronous** to ensure a deterministic behaviour

- Only asynchronous communication for alarm trigger

Network - TCP Server

- Listens on **all interfaces** (INADDR_ANY)
- **New thread** for each incoming connection

Serve client thread:

- Infinite loop
- Wait for msg from client (read)
- Call **command_handler** function
- Send back result of command_handler function

command_handler function:

void (int *commandId*, char * *args*, char * *response*, int *connfd*, char * *client_ip*)

- Unique command id
- Arguments for this command call
- Buffer to write response on
- Socket id
- IP of client that sent command

Network - TCP Client

- One thread for each server connection

Server connection thread:

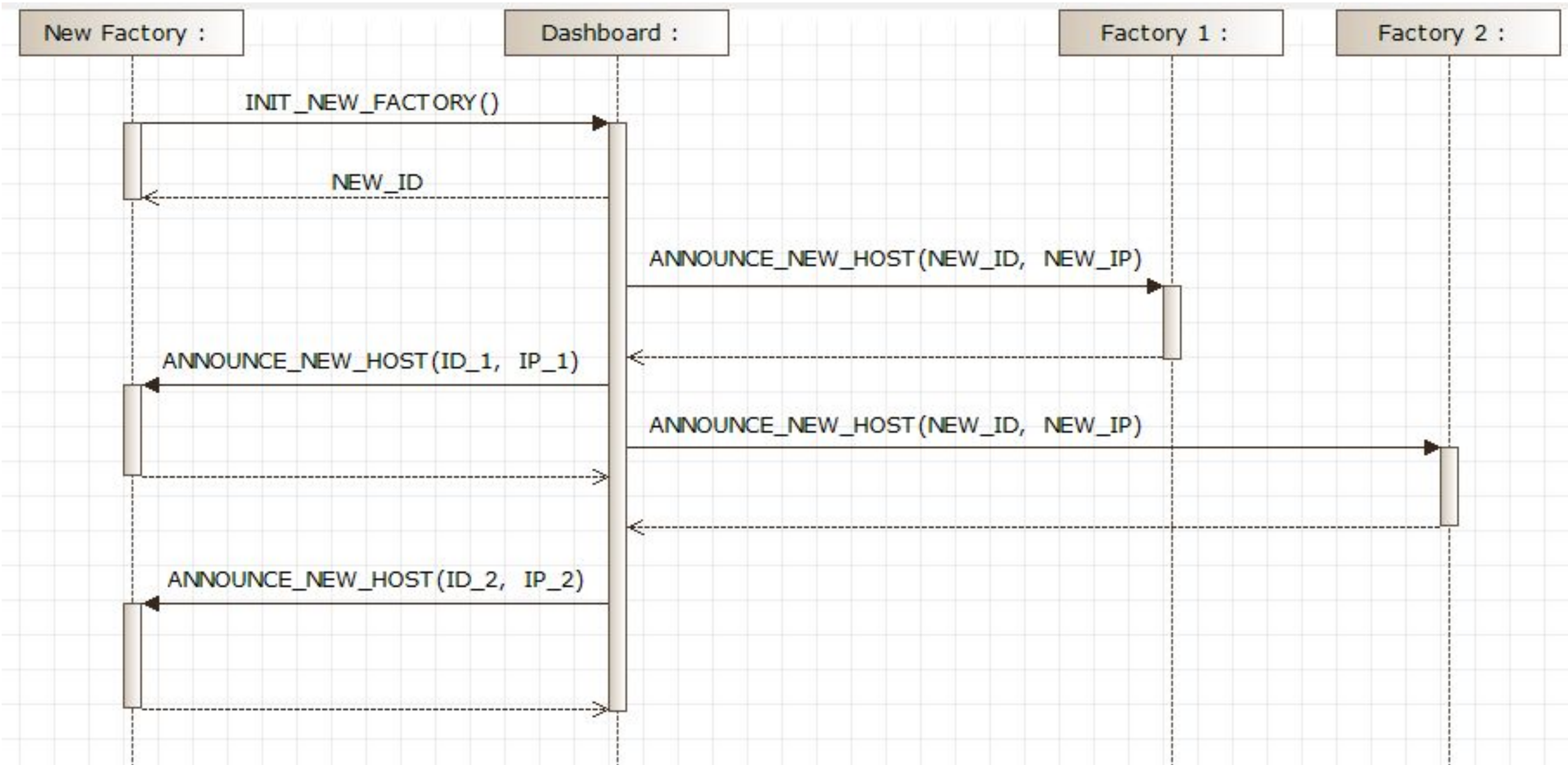
- Infinite loop
- Wait for command to send (*pthread_cond_t*)
- Send CommandId and Args to Server
- Wait for the response
- Signal response received and save it

Sending a command:

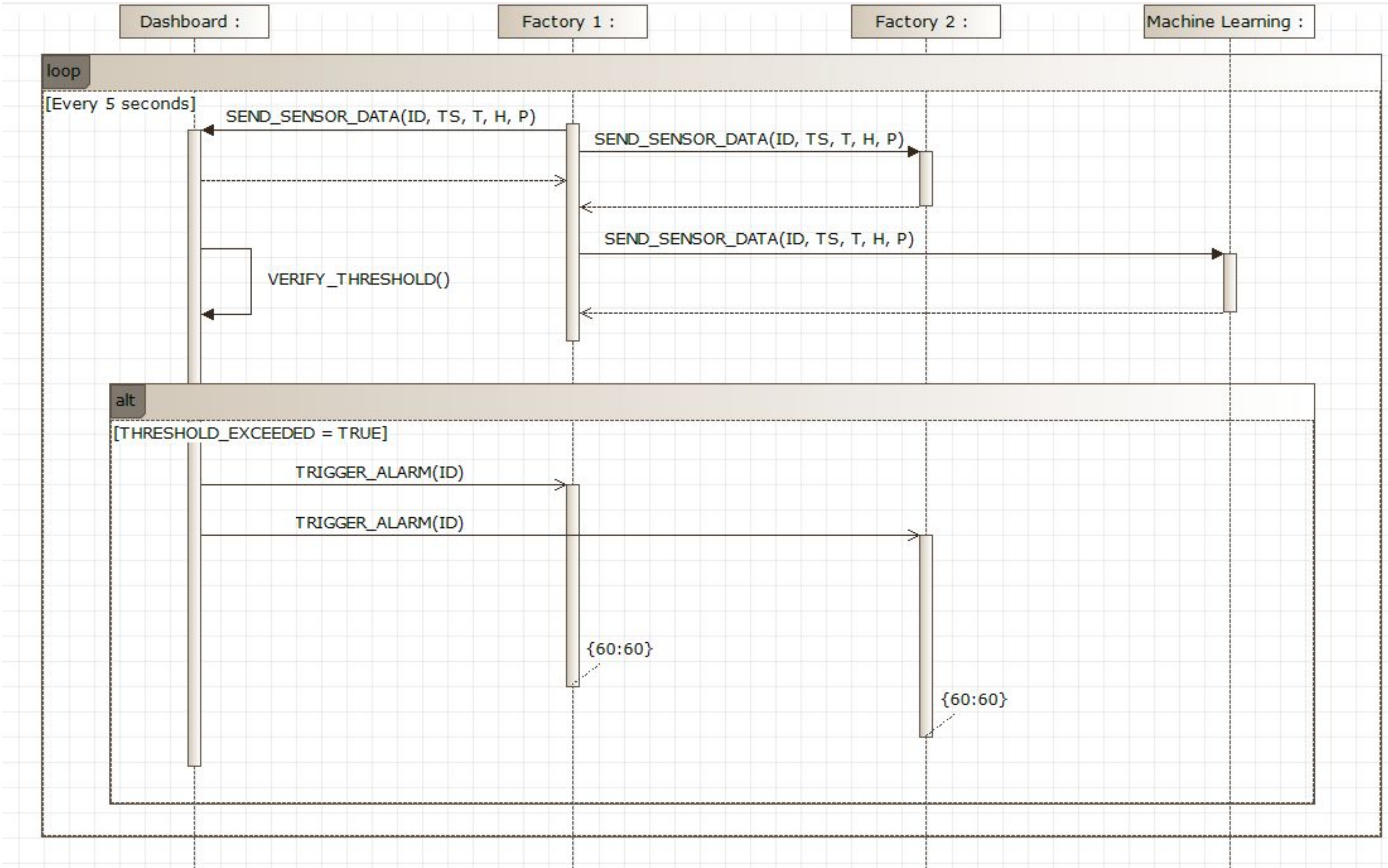
1. Lock mutex
2. Save command data to structure
3. Signal that there is a command
4. Wait for response signal (blocking call)
5. Unlock mutex

- Connection data stored in linkedlist.

Network - Factory Initialization Sequence



Network - Sensor Data Communication Sequence

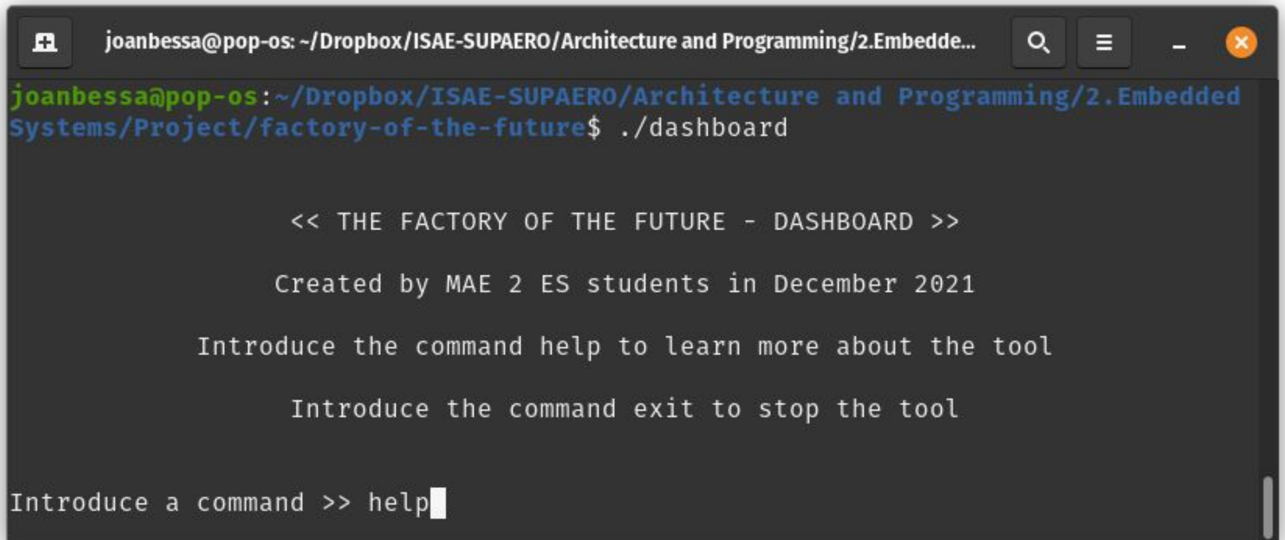


Dashboard

Alberto, Gonzalo, Joan & Nicola

Dashboard - Main Principles

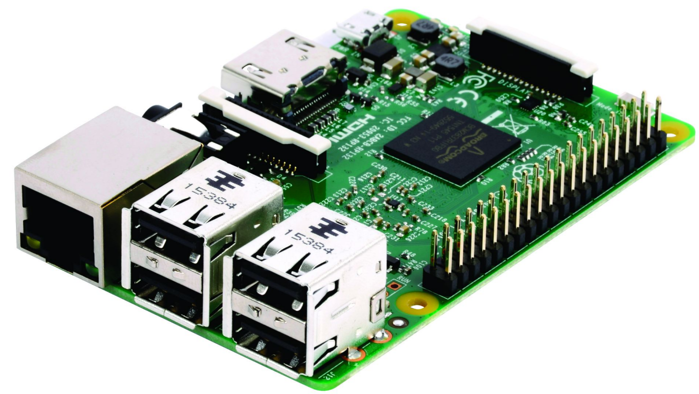
- Act as a Central Control Unit for the distributed factory
- Provide the user with a Command Line Interface
- Display information and data from the factories



Dashboard - Main Principles

What does the main function do?

- Assigns an ID to the recently connected factories.
- Accepts TCP connections.
- Runs the dashboard GUI.
- Closes connections.



Dashboard - commands

All available commands:

- show
- plot
- sendcom
- settrheshold
- downloadhistory
- listperipherals
- listalarms
- predict

Command functions include:

- *Detection of flags*
- *Automatic error messages*

and the most important one: - help

Dashboard - database

The next measures overwrite the latest ones when the database is full.

	Time	Temperature	Humidity	Pressure
1	HH:MM:SS	ºC	%	Pa
2	HH:MM:SS	ºC	%	Pa
3	HH:MM:SS	ºC	%	Pa
4	HH:MM:SS	ºC	%	Pa
⋮				
2000	HH:MM:SS	ºC	%	Pa

Dashboard - factory_comm

- Manages communication between Dashboard, Factories and Machine Learning

trigger_alarm
Instructs all factories to activate their alarm

sendcommand
Send command to a factory sensor

listPeripherals
For a given factory, lists the available sensors/actuators

getPrediction
Demands a prediction from the ML module and prints it

Dashboard - factory_data

- Manages communication between Dashboard, Factories and Machine Learning

downloadhistory

Download factory data into a text file

showhistory

Shows the downloaded text file in the interface

Dashboard - listing

- Lists the factories as well as their alarm state.

listFactories

Shows the current factories connected to the network.

listAlarms

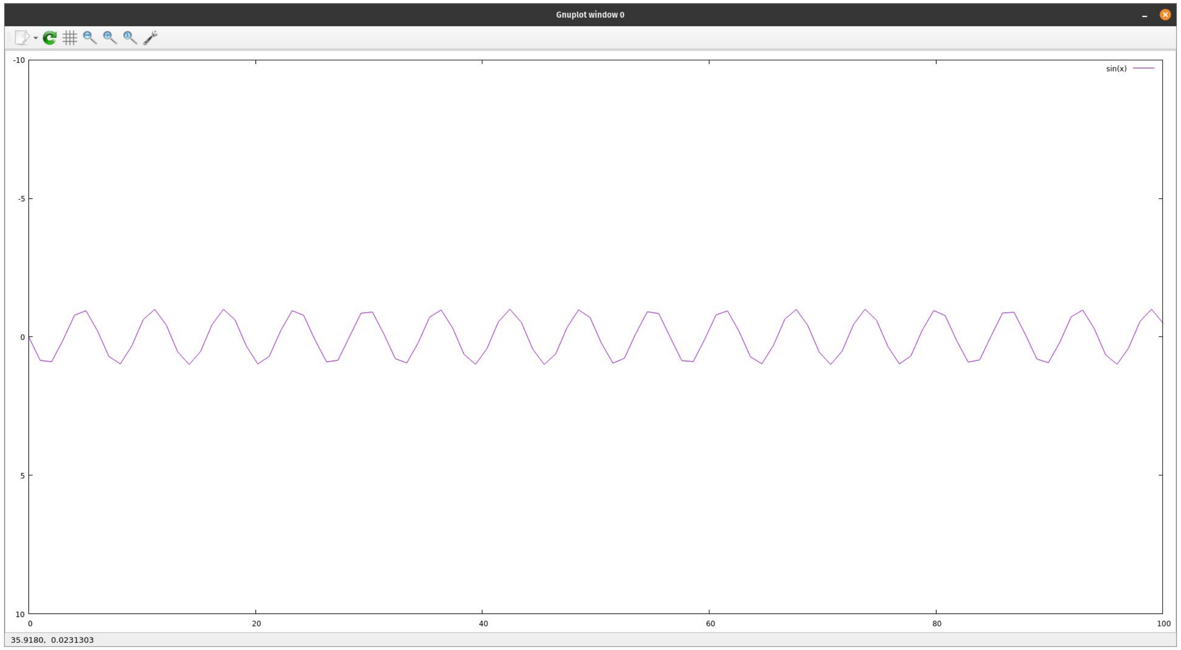
For all the current connected factories, display the current state of their alarms (ON, OFF).

Dashboard - plot

- Provides plotted information of the sensors

plot_sensors

Plots the latest information of all the sensors within a factory using **gnu_plot**



Sensors & Actuators

Fernando, Neel, Tatiana & Wendi

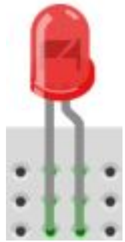
Description of the System

Each factory is equipped with its own set of sensors and actuators.

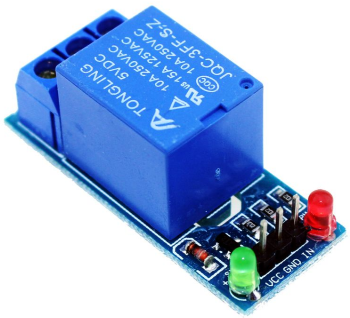
Every factory has a BME280 sensor which measures three separate parameters. The data taken from the sensor is transmitted periodically by each factory.

Each factory is equipped with three types of actuators -

- LED
- Relay
- Buzzer



LED



RELAY

Description of the System

The BME280 sensor used here measures the *relative humidity*, *barometric pressure* and *ambient temperature*.

This sensor can use the i2C bus or the SPI bus. All the sensors and actuators in the factories have been configured to use the i2C bus.

The sensor data from each factory is transmitted to the other factories and to the dashboard every five seconds.

BME280 Sensor

Description of the System

The LED and Buzzer are utilised when the factory alarm is triggered.

The function **run_alarm()** will trigger the alarm for a time period of sixty seconds. The LED and the Buzzer are alternatively switched on and off when the alarm is sounding.

Even if only one factory triggers the alarm, the alarm in all the factories will be triggered simultaneously.

The LED and the Relay also have separate functions to be triggered individually: **set_led_state(short state)** and **set_relay_state(short state)**. If *state* = 0, they are off, else they are on.

Sensors - Main Principles

Abstraction of the handling of the sensors and actuators

- Enable the raspberry I2C bus
- Enable the raspberry inputs and outputs
- Define the architecture of the Factory
- Initialization of the sensors of the factory
- Initialization of the actuators of the factory
- Read Data from sensors
- Offer the sensor data as a service.
- Receive Commands from the interfaces to drive actuators
- Enable and program the Alarm (LED and Buzzer)
- Check the presence of sensor and actuators

Functions & Interfaces

Rq	Use Case	Functions	Command (handle_command())
1	It is possible to drive the GPIO remotely.	set_led_state(); set_relay_state()	CMD_SET_RELAY_STATE
2	It is possible to control the actuators remotely.	set_led_state(); set_relay_state()	CMD_SET_LED_STATE
5	An alarm is sent to the system that makes the request to trigger the alarm.	trigger_factory_alarm()	CMD_TRIGGER_ALARM
6	Each factory has at least the following information: temperature, pressure, humidity.	read_sensor_data(); store_sensor_data()	-
7	Each factory has at least two actuators: led, relay.	has_led(); has_relay()	CMD_GET_PERIPHERALS
8	The factories communicate with each other every 5 seconds: temperature, humidity, pressure.	broadcast_sensor_data()	CMD_SEND_SENSOR_DATA; CMD_SEND_SENSOR_HISTORY_FILE
9	The factories have a service that communicate the list of available sensors.	has_sensors()	CMD_GET_PERIPHERALS

Initialization

- ➔ Start TCP server
  accept_tcp_connections_non_blocking(handle_command, &server_thread);

- ➔ Initialize the sensors and data buffer
  init_sensor();
  init_sensor_data_buffer(&sensor_history);

- ➔ Connect to the Dashboard & get factory ID
  connect_to_dashboard(dashboardAddr, &host_list, &fact_ID, 1);

- ➔ Check sensors & alarm state
- ➔ Broadcast and store sensor data

…

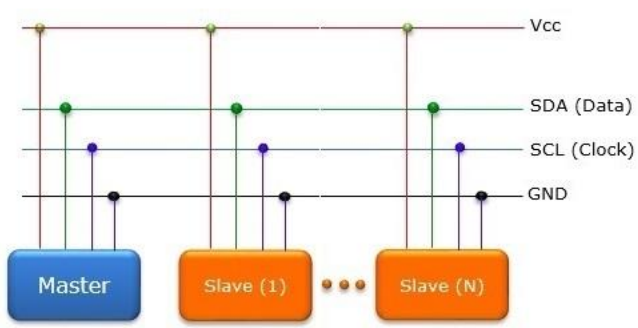
Data Structure

- ❖ Transmitted to the Dashboard and each factory every 5 seconds through broadcast
- ❖ Stored locally in sensor_history.dat
- ❖ File with all past data is transmitted to the Dashboard or the Machine Learning Manager upon request (network)

```
typedef struct SensorData {
    time_t time;
    double temperature, humidity, pressure;
} SensorData;
```

Enabling the raspberry I2C

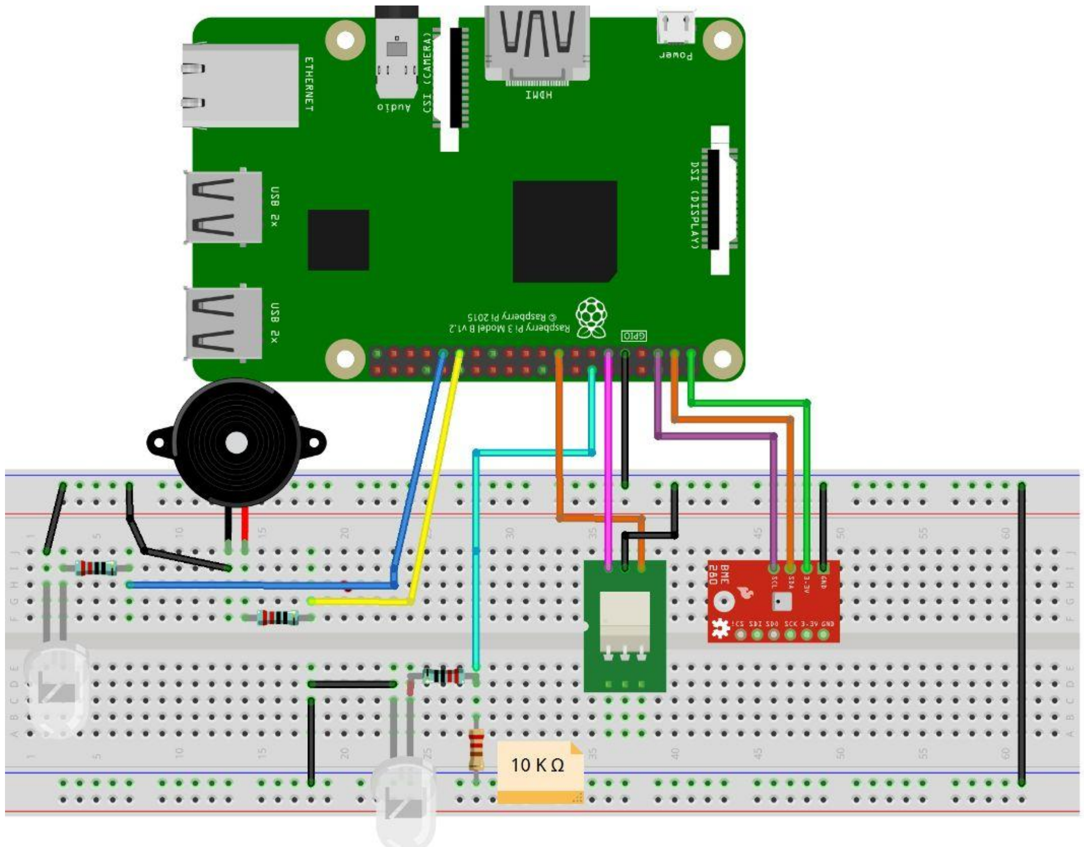
- Only uses two wires Serial Clock (or SCL) and Serial Data (or SDA)
  - The SCL wire is the clock signal that synchronizes the data transfer between devices on the I2C bus and is generated by the master device.
  - The other wire is the SDA line that carries the data.
- Supports multiple masters and multiple slaves
- ACK/NACK bit gives confirmation that each frame is transferred successfully



Set up

Connections diagram

-Raspberry PI3 B+
-1 breadboard
-2 LED
-wires
-3 resistors (220 Ω)
-1 resistor (10 K Ω)
-1 BME280 sensor
-1 relay
-1 buzzer



Set up

The connections are specified in the following figure

External pin		WiringPi	Name	Physical		Name	WiringPi	External pin	
BM280	Vin		3.3V	1	2	5.5V			
BM280	SDI	8	SDA.1	3	4	5.5V			
BM280	SCK	9	SCL.1	5	6	0V		GND	BM280
		7	1-Wire	7	8	TxD	15		
RELAY	GND		0V	9	10	RxD	16		
RELAY	+	0	GPIO.0	11	12	GPIO.1	1	+	LED
		2	GPIO.2	13	14	0V			
		3	GPIO.3	15	16	GPIO.4	4		
RELAY	Vcc		3.3V	17	18	GPIO.5	5		
		12	MOSI	19	20	0V			
		13	MISO	21	22	GPIO.6	6		
		14	SCLK	23	24	CE0	10		
			0V	25	26	CE1	11		
		30	SDA.0	27	28	SCL.0	31		
BUZZER	+	21	GPIO.21	29	30	0V			
LED_ALAR	+	22	GPIO.22	31	32	GPIO.26	26		
		23	GPIO.23	33	34	0V			
		24	GPIO.24	35	36	GPIO.27	27		
		25	GPIO.25	37	38	GPIO.28	28		
			0V	39	40	GPIO.29	29		

Device availability

The system requires that Factories shall be capable of giving their list of available sensors/actuators (*req 6*).

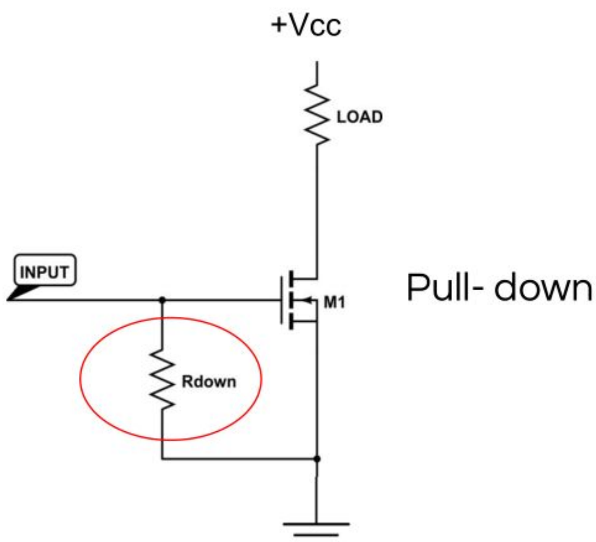
In order to update this list, a service is provided to check the availability of the sensors and actuators currently connected to the factory.

- **has_sensors()**
- **has_led()**
- **has_relay()**

Device availability - LED

Reading a disconnected pin will return unstable values that will oscillate between 0 and 1.

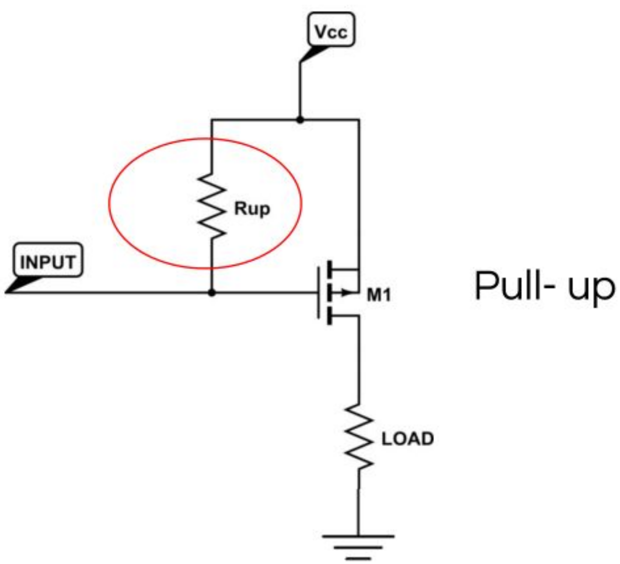
An external pull-down resistor is implemented in the LED actuator, to ensure a constant value of 0 in this pin when the LED is correctly connected.



Note that depending on the nature of the failure this system may not detect the disconnection.

Device availability - RELAY

A similar approach is used with the relay, however in this case there is a pull up resistor in the wiring of the relay module, so the checking is adjusted to consider a logical 1 as a connected device.



Note that depending on the nature of the failure this system may not detect the disconnection.

Device availability - BME280 sensor

For the temperature, pressure and humidity sensor, the initialization process offers a variable that indicates the status of the sensor after the initialization.
we take advantage of this to periodically check the sensor availability.

Note that the initialization process is time consuming and shall not be used every time a reading is required.

Machine Learning

Bárbara, Jerzy, Kaavya & Maanasa

Machine Learning - Main Principles

Requirement:

A service makes it possible to give estimates on the different physical values of the factories.

- To define in advance the behaviour of the system;
- To use linear regression on the evolution of temperature sensor;
- To give estimates on the different physical values of factories.

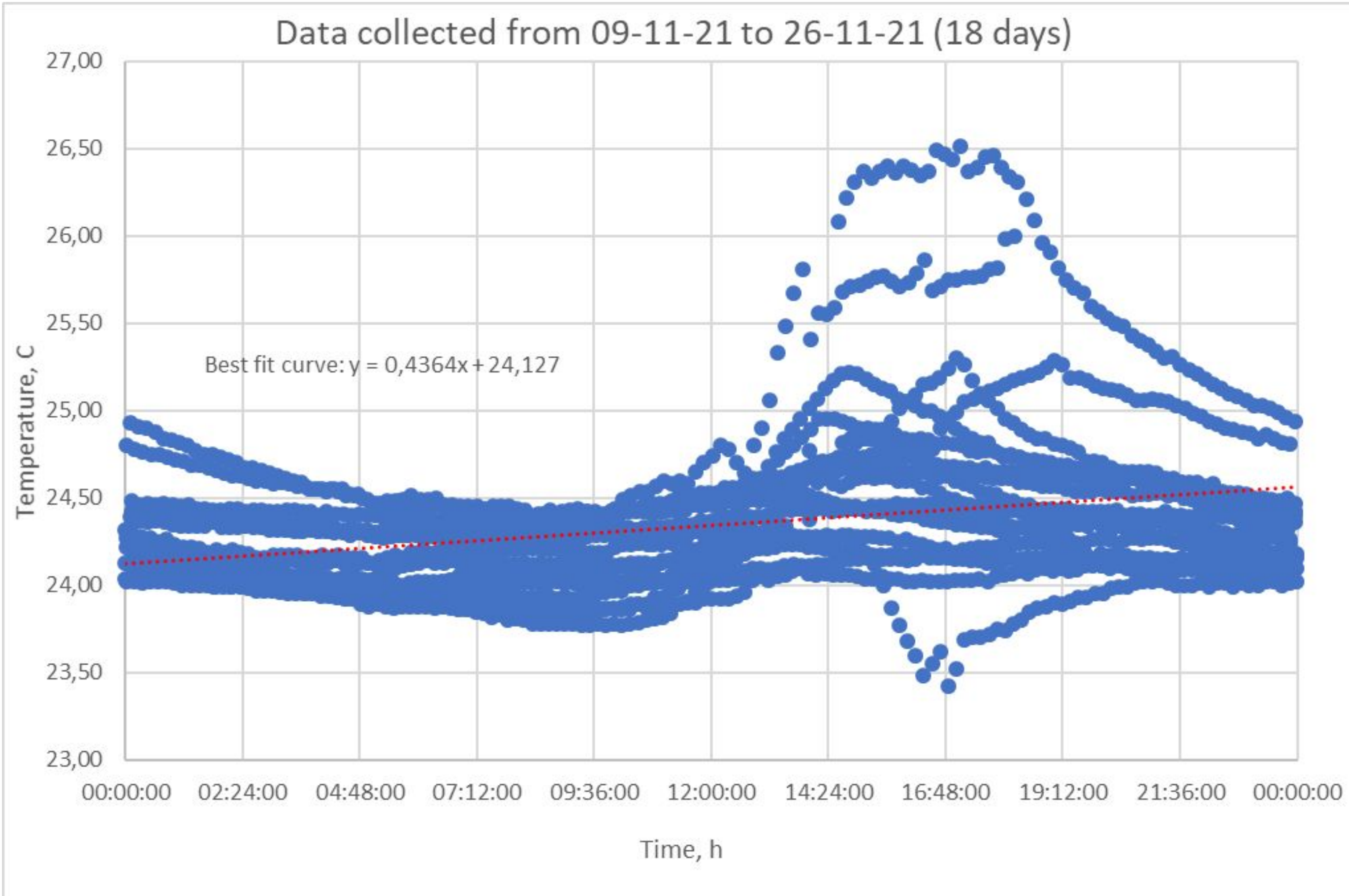
Machine Learning - Procedure

- Data Collection;
- Data Preparation;
- Choose a model;
- Train the model;
- Apply the model/Make predictions.

Machine Learning - Data Collection

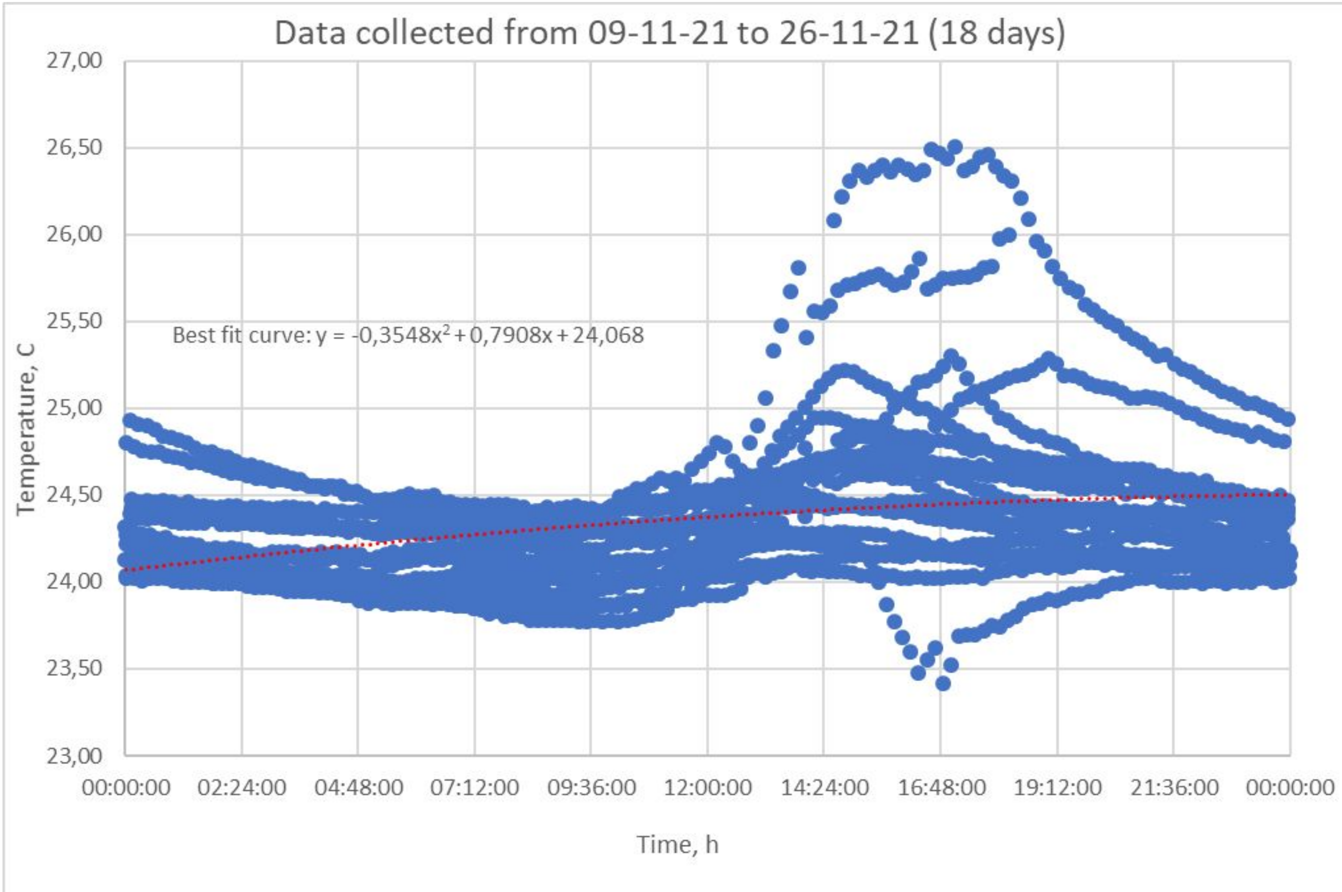
- Time and the corresponding temperature;
- Data Set - 18 days, collecting every 10 minutes;
- 1974 values for training (80% of the data);
- 494 values for prediction (20% of the data);

Machine Learning - Data Preparation



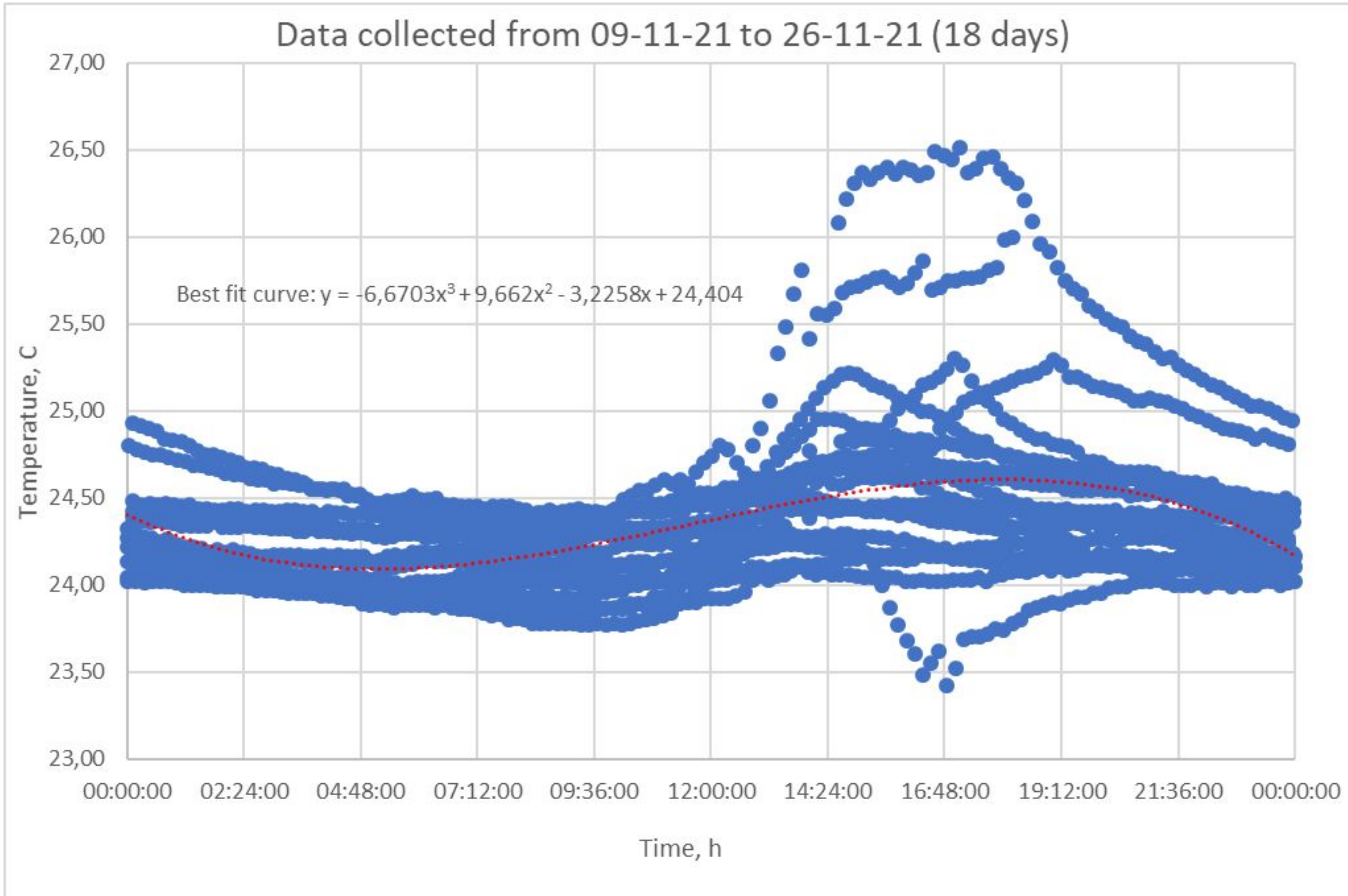
Linear (1st degree) best fit curve

Machine Learning - Data Preparation



2nd degree polynomial best fit curve

Machine Learning - Data Preparation



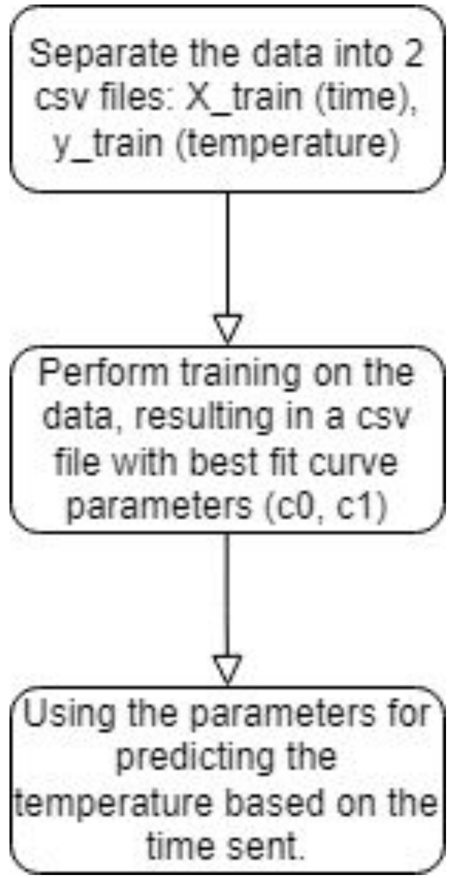
3rd degree polynomial best fit curve

Machine Learning - Model flow chart

Input data

Training

Prediction



Machine Learning - Choose a model

From **GSL - GNU Scientific Library:**

Linear regression with a constant term

$$Y(c,x) = c_0 + c_1 x$$

get_matrix_dims
load_matrix_from_csv
gsl_blas_dgemm
gsl_linalg_LU_decomp
gsl_matrix_get

gsl_matrix_alloc
gsl_matrix_transpose_memcpy
gsl_permutation_alloc
gsl_linalg_LU_invert gsl_matrix_sub
gsl_matrix_scale

Machine Learning - Choose a model

From **GSL - GNU Scientific Library:**

Linear regression with a constant term

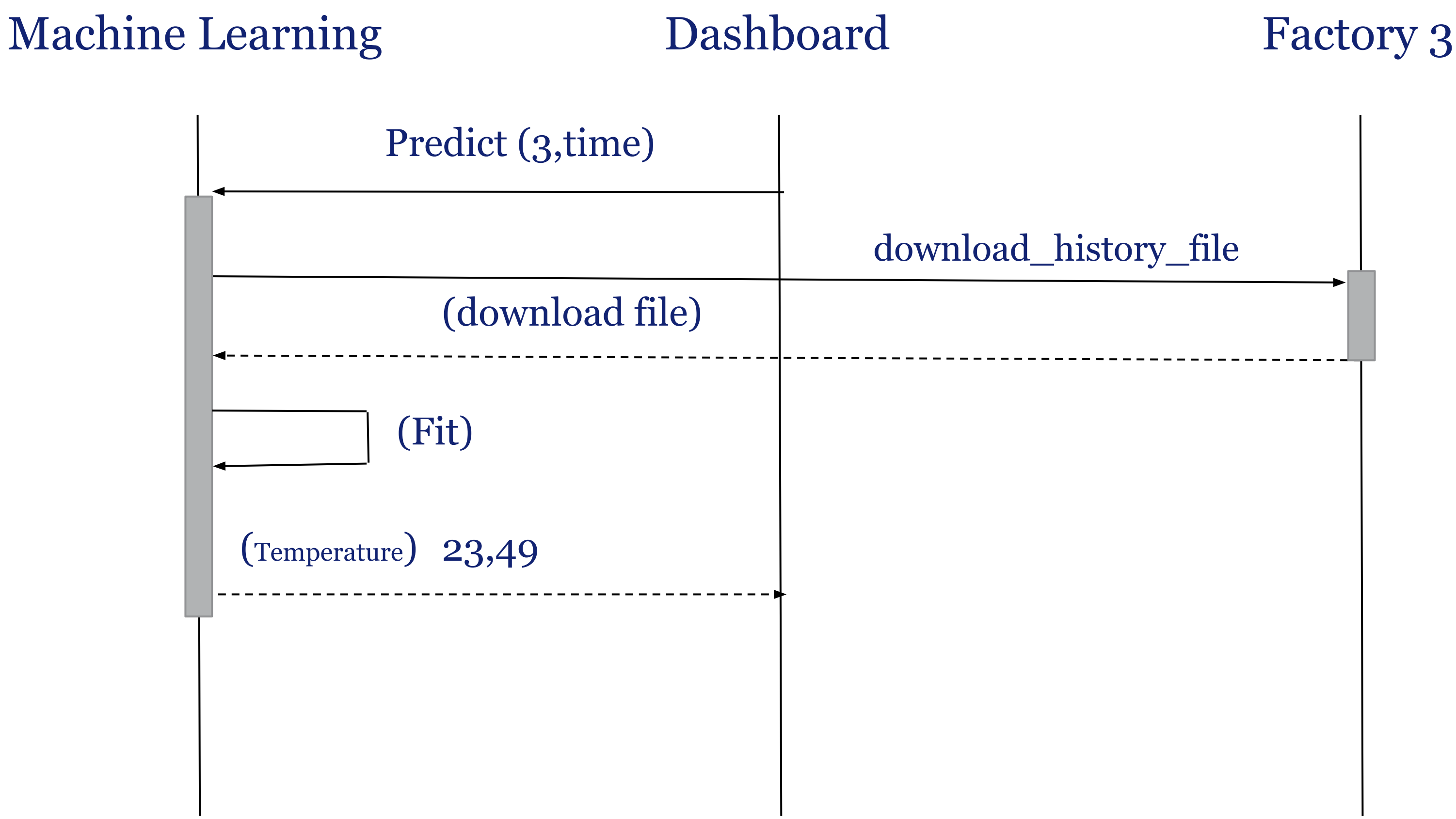
$$Y(c,x) = c_0 + c_1 x$$

$$\sigma^2 = \sum (y_i - Y(c,x_i))^2/(n-p)$$

$$\chi^2 = \sum_i w_i (y_i - Y(c,x_i))^2$$

$w = 1/\sigma^2$ Weights $C_{ab} = \langle \delta c_a \delta c_b \rangle$ Covariance Matrix

Sequence Diagram - Real time prediction



Machine Learning - Real time prediction

1. **predict 3 10 20 30** (predict factID hour minute second) -> Accepts predict command from dashboard (Network)
2. function receives_sensor_history_file -> Request and receive history file from factory
3. function append_factory_data -> Append data to existing factory sensor data
4. function fit -> Train the linear model
5. function predict_temp -> Predicts the value of temperature for a given factory at the requested time

```
temperature = gsl_matrix_get(beta, 0, 0) + date * gsl_matrix_get(beta, 1, 0);
```

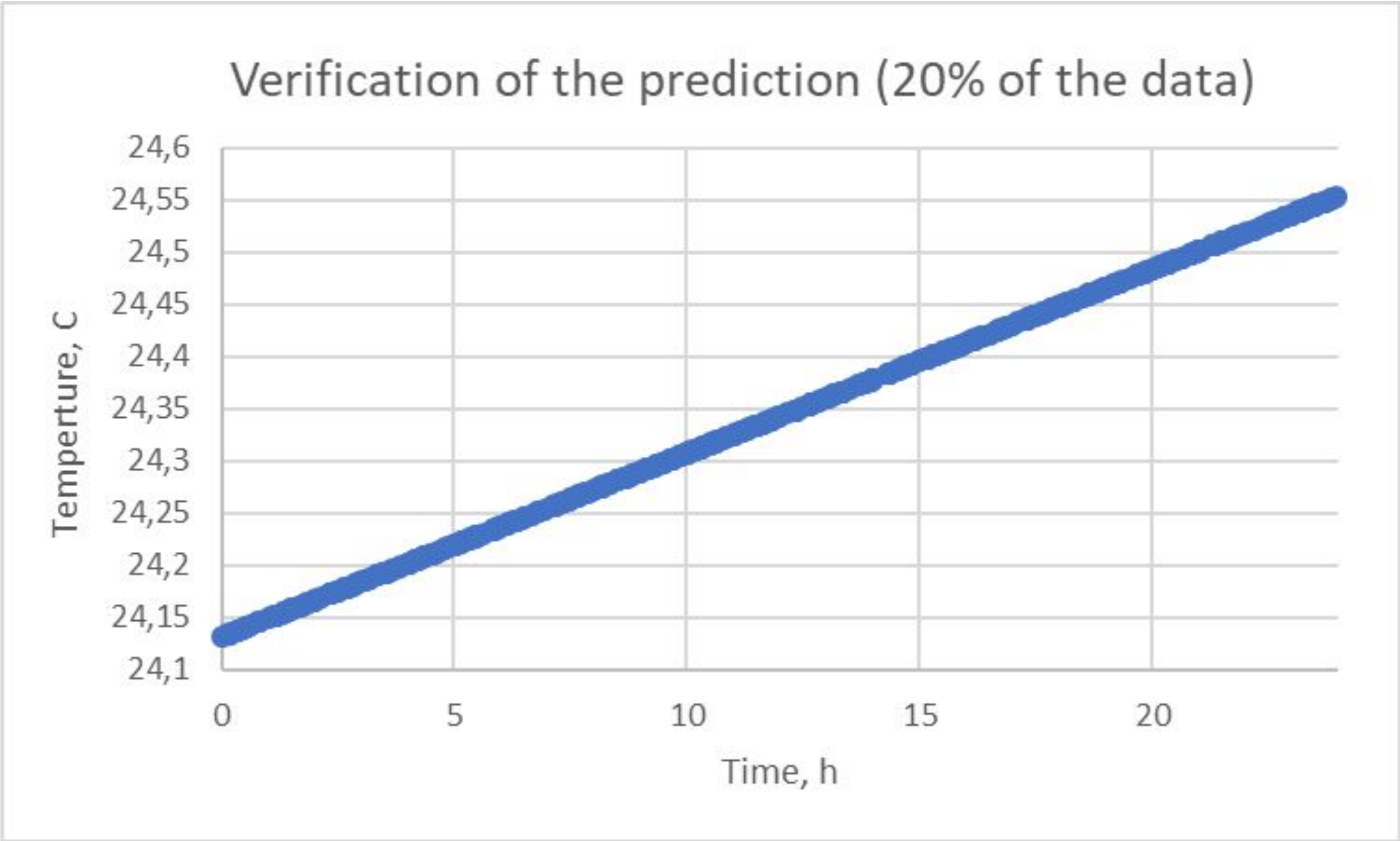
Machine Learning - Training and evaluation

Using the features available on **GSL - GNU Scientific Library,** the model was trained to predict the temperatures in 24h range using 80% of collected data.

```
X_train.csv contains 1 features and 1974 examples [15792 bytes]
Reading X_train.csv
Reading y_train.csv
s2 = 0.142503
sigma2 = 0.142359
beta estimates:
est beta[0] = 24.1 (0.00029)
est beta[1] = 0.0176 (1.5e-06)

errors (y - y_hat):
u[0] = 0.0597
u[1] = -0.00731
u[2] = 1.2
u[3] = 0.226
u[4] = -0.0593
u[5] = -0.167
u[6] = 0.0518
u[7] = 0.381
u[8] = 1.94
```

Machine Learning - Prediction verification



Demo

Entire Team

DEMO - Connecting all hosts

Instance	Lab ID	IP Address
Dashboard	16	192.168.47.60
Factory 1	8	192.168.47.44
Factory 2	10	192.168.47.48
Factory 3	11	192.168.47.50
Factory 4	14	192.168.47.56
ML Module	15	192.168.47.58

1. ./dashboard
2. ./factory [dashboard IP]
3. ./ml [dashboard IP]

Institut Supérieur de l'Aéronautique et de l'Espace
10, avenue Édouard-Belin – BP 54032
31055 Toulouse Cedex 4 – France
T +33 5 61 33 80 80

www.isae-supaero.fr

Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO