

# IoT Software Foundation for Hackwil

The Mean Squares – PolyHACK 2020

ASUS Robotics & AI Center Challenge

# Our mission

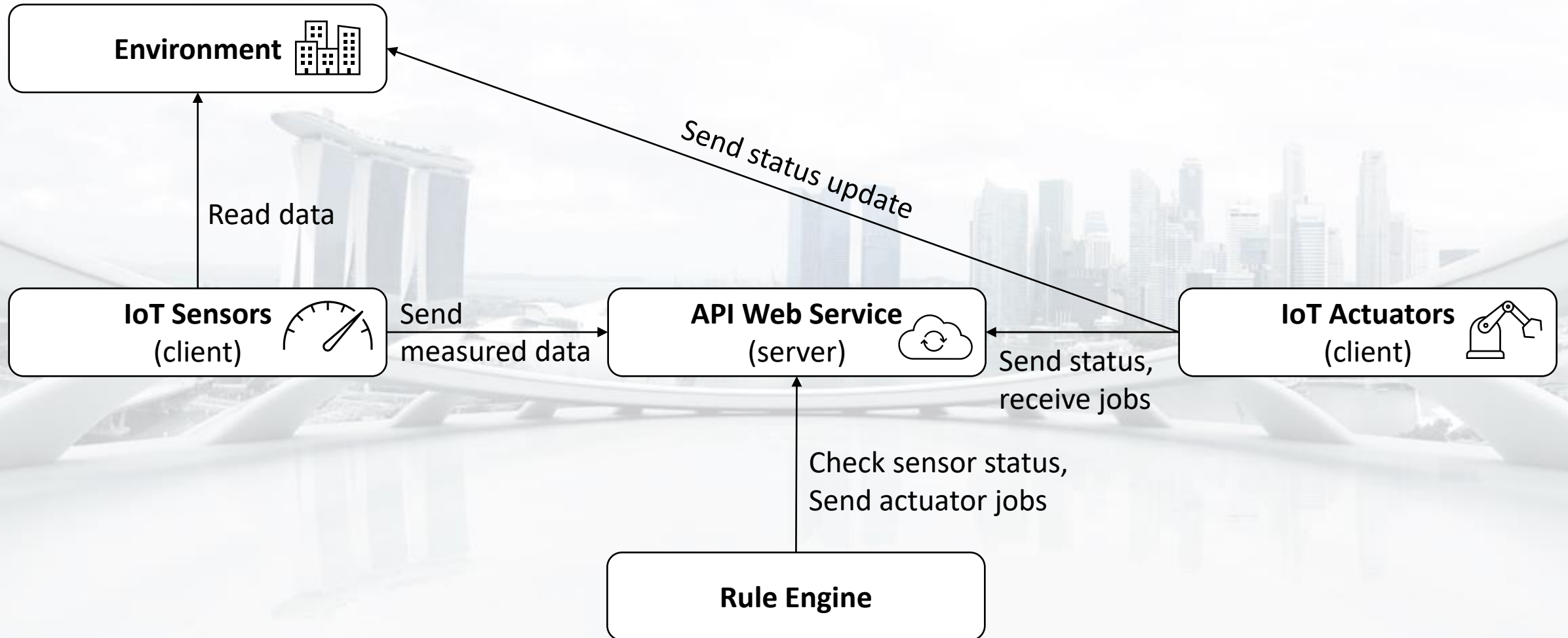
*We, The Mean Squares*, are representatives of the company Polycraft.

Our goal is to provide the software to automate IoT devices throughout Hackwil.

To proof our technical competence we have build an MVP with various types of sensors and actuators communicating via a server that act in a simulated environment.

# System Overview

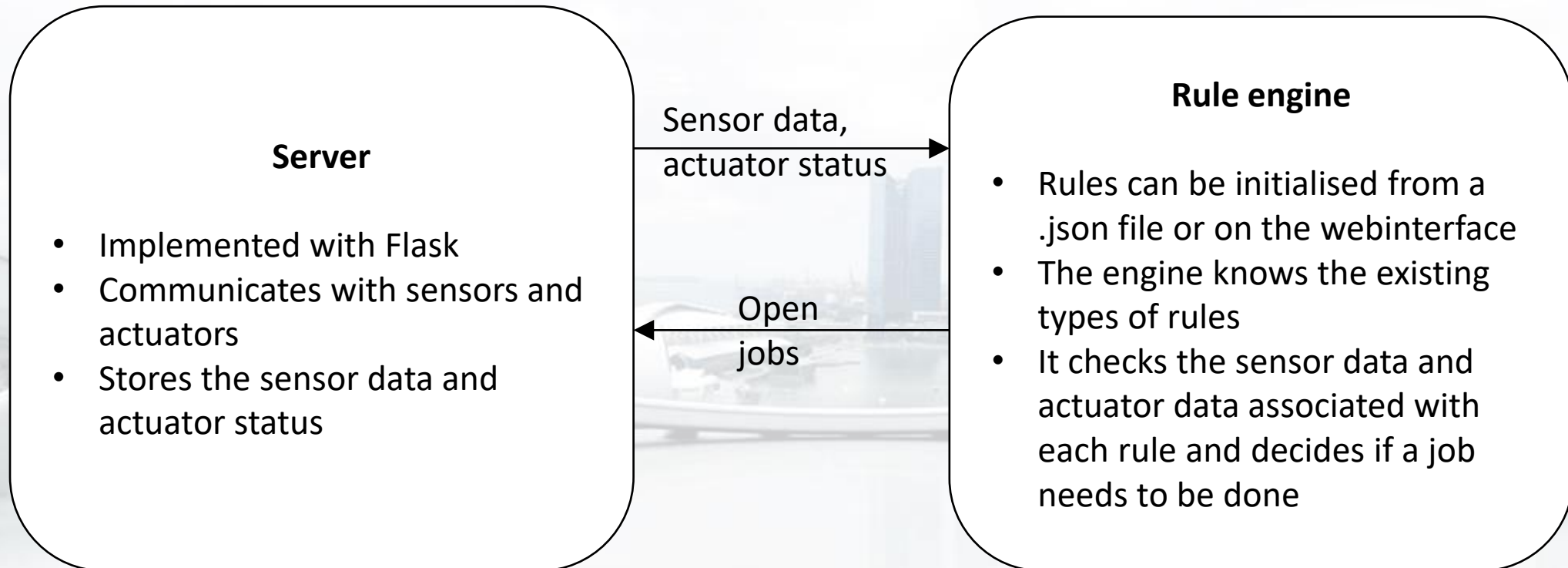
Fat server with thin clients implemented using Flask





# API Web Service & Rule engine

Fat server containing most of the application's logic





# Client-Server Communication Protocols

## Client side communication

All clients communication must contain the following fields:

- 'id': a unique identifier for the device
- 'ancestors': a list of inheritance of the device in question, in order from youngest to oldest (e.g. ['TemperatureSensor', 'Sensor', 'IOT\_Device'])

Clients are Sensor or Actuator.:

- A Sensor has a data entry, which contains a dictionary of measured values
- An actuator instance has a status entry, containing a dictionary describing the state of the device.

## Server side communication

The server receives messages from the client, and stores the received data locally.

Sensor:

- The sensor only sends data

Actuator:

- If an actuator needs actuating, the server will wait until the next time the actuator contacts the server, and will return the job to do be executed by the actuator.
- The response from the server is a list of jobs from the rules engine as a dictionary .
- If there is nothing to do, the dictionary is empty. If there is something to do, the dictionary will list what values to change. E.g. if a door should be unlocked, the response will be: {'DoorLocked': False}.





# IoT Sensors

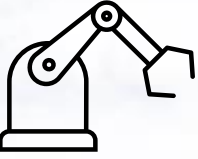
Sensors are Thin clients that read from the environment and send their data to the server

## Implemented sensor types:

- Temperature sensor
- Humidity sensor
- Brightness sensor
- Proximity sensor
- Noise sensor
- Motion sensor
- Distance sensor
- Airquality sensor

## Initialisation:

- Each sensor has an ID, a type and a position
- The number and type of sensors in Hackwil can be initialised in a .json file or via the webinterface



# IoT Actuators

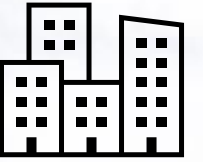
Actuators are thin clients that sent their status to the server and receive jobs to execute

## **Implemented actuator types:**

- Smart lamp
- Smart door lock
- Motor position
- Smart heating
- Smart sprinkler

## **Initialisation:**

- Each actuator has an ID, a type, a position and a status
- The number and type of sensors in Hackwil can be configured in a .json file or in the webinterface



# Simulated Environment

- The environment is a class simulating the input data for each sensor
- Each sensor can read a feed of random variables that are scaled according to its type
- The actuators can influence the environment (e.g. a heater increases the temperature in its surrounding)
- Due to the modular implementation, the environment can easily be replaced by real input data



# Try our MVP

Our sourcecode: <https://github.com/Karko93/Polyhack2020-Project.git>

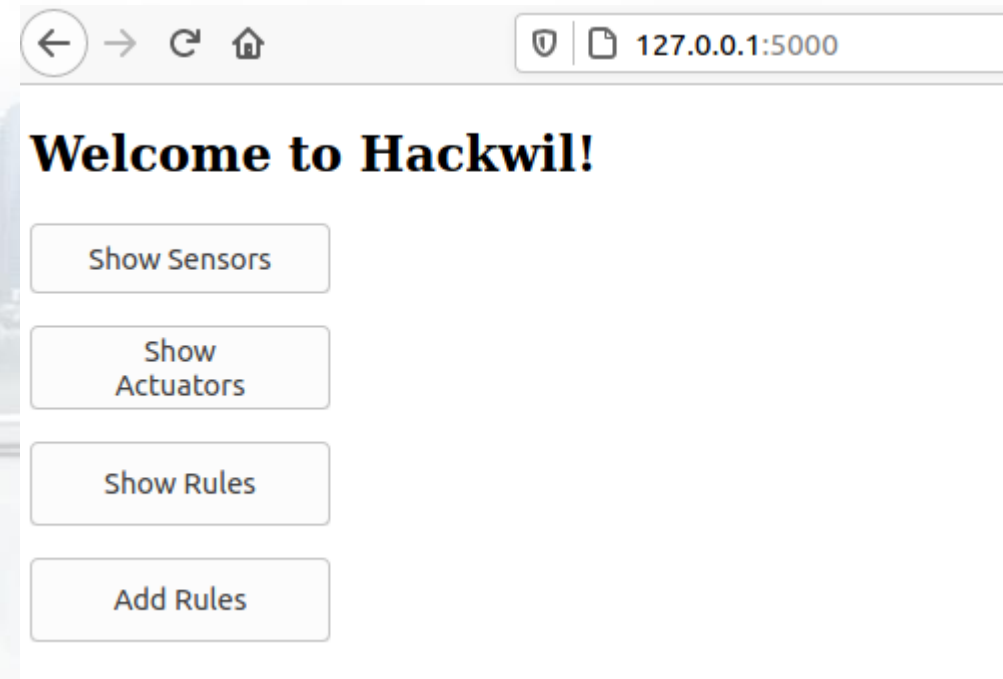
Run the code according to the readme.md

Go to the webpage <http://127.0.0.1:5000/> to see the simulations of Hackwil



# The Webinterface

With our interactive webinterface the state of our IoT system can be monitored. New Rules for interaction between actuators and sensors can be added on the fly.





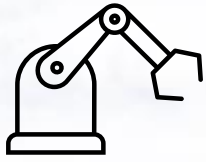
# The Sensor

|   | ID  | Type              | Timestamps                 |
|---|-----|-------------------|----------------------------|
| 0 | 001 | BrightnessSensor  | 2020-11-08 09:21:23.097778 |
| 5 | 002 | TemperatureSensor | 2020-11-08 09:21:23.166324 |
| 1 | 003 | ProximitySensor   | 2020-11-08 09:21:23.111367 |
| 8 | 004 | NoiseSensor       | 2020-11-08 09:21:23.207680 |
| 2 | 005 | DistanceSensor    | 2020-11-08 09:21:23.125170 |
| 6 | 006 | MotionSensor      | 2020-11-08 09:21:23.180323 |
| 3 | 007 | HumiditySensor    | 2020-11-08 09:21:23.139115 |
| 9 | 008 | HumiditySensor    | 2020-11-08 09:21:23.220761 |
| 4 | 009 | AirQualitySensor  | 2020-11-08 09:21:23.152817 |
| 7 | 010 | TemperatureSensor | 2020-11-08 09:21:23.194027 |

Overview over all connected sensor with last interaction

|    | brightness | timestamp                  |
|----|------------|----------------------------|
| 0  | 63.404223  | 2020-11-08 09:20:59.391306 |
| 1  | 46.725644  | 2020-11-08 09:21:00.571086 |
| 2  | 43.405267  | 2020-11-08 09:21:01.708654 |
| 3  | 44.516281  | 2020-11-08 09:21:02.918431 |
| 4  | 42.976452  | 2020-11-08 09:21:04.052620 |
| 5  | 41.632564  | 2020-11-08 09:21:05.164620 |
| 6  | 75.103981  | 2020-11-08 09:21:06.223578 |
| 7  | 38.688601  | 2020-11-08 09:21:07.304736 |
| 8  | 32.736825  | 2020-11-08 09:21:08.348638 |
| 9  | 46.672818  | 2020-11-08 09:21:09.510994 |
| 10 | 44.745092  | 2020-11-08 09:21:10.648388 |
| 11 | 54.738702  | 2020-11-08 09:21:11.740789 |
| 12 | 32.605155  | 2020-11-08 09:21:12.949979 |

The measurement data for each sensor given an ID



# The Actuator

|   | ID  | Type          | jobs |
|---|-----|---------------|------|
| 0 | 100 | SmartLamp     | {}   |
| 1 | 200 | SmartDoorLock | {}   |
| 2 | 300 | MotorPosition | {}   |
| 3 | 400 | Heating       | {}   |
| 4 | 500 | Sprinkler     | {}   |

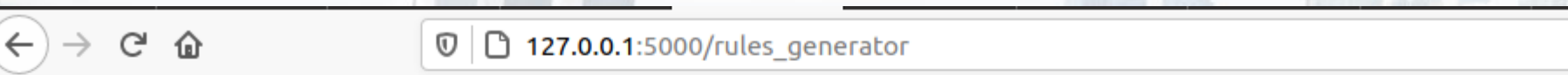
Overview over all connected actuators with assigned jobs

|    | intensity | timestamp                  |
|----|-----------|----------------------------|
| 0  | 0         | 2020-11-08 09:20:59.513107 |
| 1  | 0         | 2020-11-08 09:21:00.689211 |
| 2  | 0         | 2020-11-08 09:21:01.847277 |
| 3  | 0         | 2020-11-08 09:21:03.035147 |
| 4  | 0         | 2020-11-08 09:21:04.125799 |
| 5  | 0         | 2020-11-08 09:21:05.206190 |
| 6  | 0         | 2020-11-08 09:21:06.290141 |
| 7  | 0         | 2020-11-08 09:21:07.334318 |
| 8  | 0         | 2020-11-08 09:21:08.471834 |
| 9  | 0         | 2020-11-08 09:21:09.619535 |
| 10 | 0         | 2020-11-08 09:21:10.724443 |
| 11 | 0         | 2020-11-08 09:21:11.879709 |

The record of the latest state of each actuator by ID

|   | uniq_id | actuator_ids     | actuator_output | actuator_value_False | actuator_value_True | comparisons | requirement | sensor_ids       | sensor_reading           | thresholds   |
|---|---------|------------------|-----------------|----------------------|---------------------|-------------|-------------|------------------|--------------------------|--------------|
| 0 | 000000  | [000030, 000032] | [intensity]     | [0]                  | [1]                 | [=, =]      | all         | [000015, 000010] | [motion, noise_detector] | [True, True] |

Overview over all assigned rules with the relevant actuator and sensor IDs. Furthermore the type of sensor and rule is shown.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5000/rules\_generator'. The browser's navigation bar includes back, forward, refresh, and home icons. The main content area of the browser shows a web application interface for generating rules. This interface consists of two rows of controls. The first row contains three dropdown menus labeled 'Choose sensor', 'Choose reading', and 'Choose condition', followed by a 'threshold:' label and a numeric input field with up and down arrow buttons. The second row contains two dropdown menus labeled 'Choose actuator' and 'Choose action', followed by the text 'Actuator value: if true:', a numeric input field with up and down arrow buttons, the text 'if false:', another numeric input field with up and down arrow buttons, and a 'Submit Query' button.

Interface to live add rules to the system. All available sensors and actuators can be selected and a variety of conditions can be applied.



# The Mean Squares



Dominik Windey  
Eric Bonvin  
Marco Ketzel  
Andrei Militaru  
Rebecca Westphal