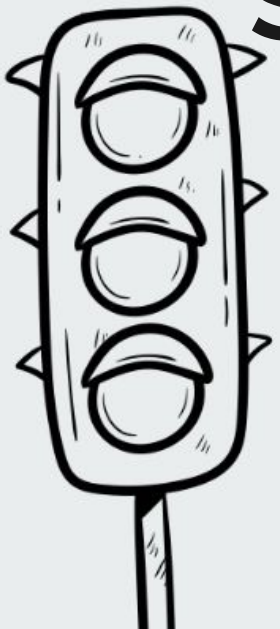
Smart Traffic Lights

Traffic for a better world!



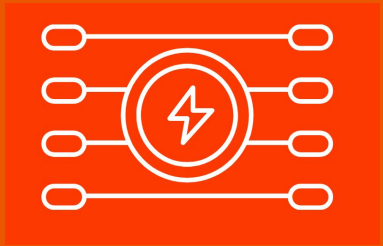
Team:
Francesca Fanelli – s313432
Aaron Segers – s306765
Cristiano Vittori – s316801
Maureen Zwart – s307021

Course:
01QWRBH: Programming for IoT Applications
Prof. Edoardo Patti
Prof. Matteo Orlando

Agenda

1. Problem Statement and Objectives
2. Use Case Diagram
3. Communication Protocols
4. Service catalog and resource catalog
5. Sensors
6. Traffic Lights protocols and LEDs management
7. User Awareness

Problem Statement



Energy Abuse due to Traffic Lights Always Switched ON

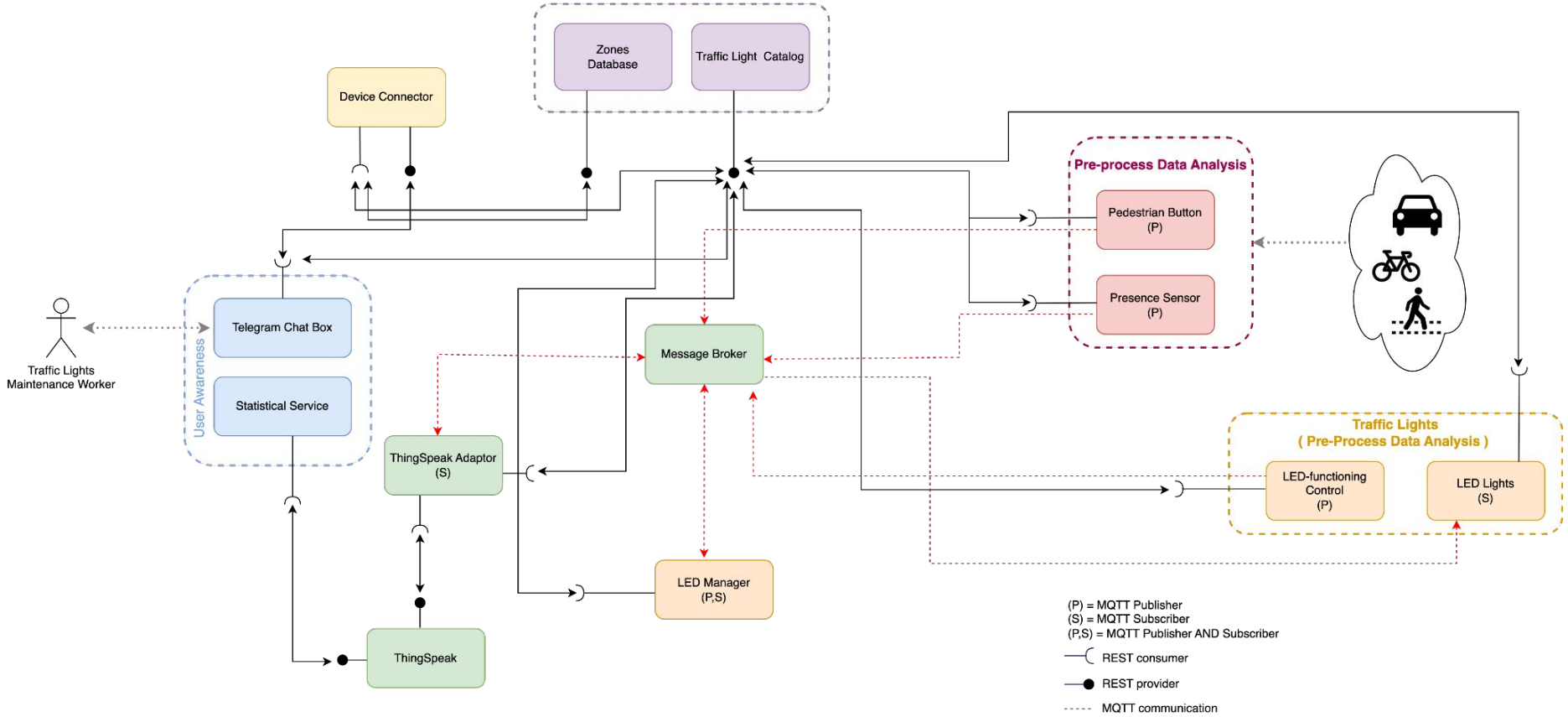
Insufficiently Planned working days for Traffic Light Manufacturers

Objectives

Decrease Energy Abuse for a More Sustainable Future

Improve Efficiency of Traffic Light Manufacturers

Use Case Diagram



Communication Protocols

MQTT

- Asynchronous communication
- Lightweight/Flexible
- Publish/subscribe paradigm
- Hierarchical message **topics**

REST

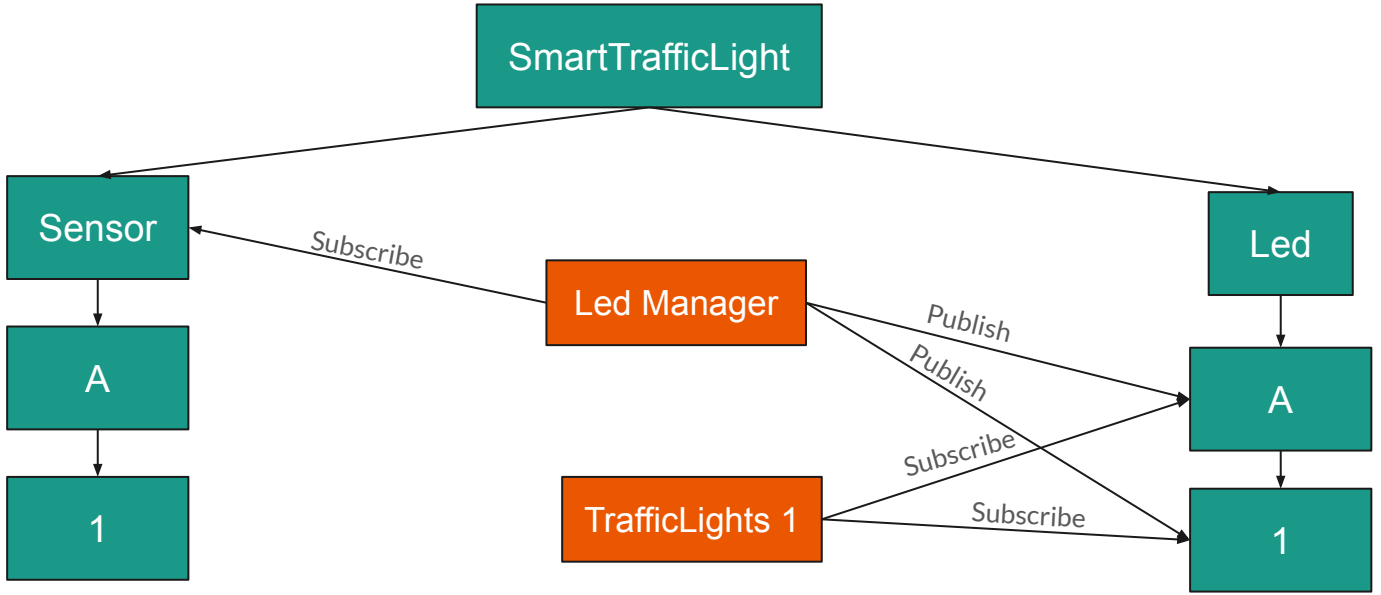
- Synchronous Communication
- Request/Response paradigm
- On demand access to the system information via **Catalogs**

MQTT Topics

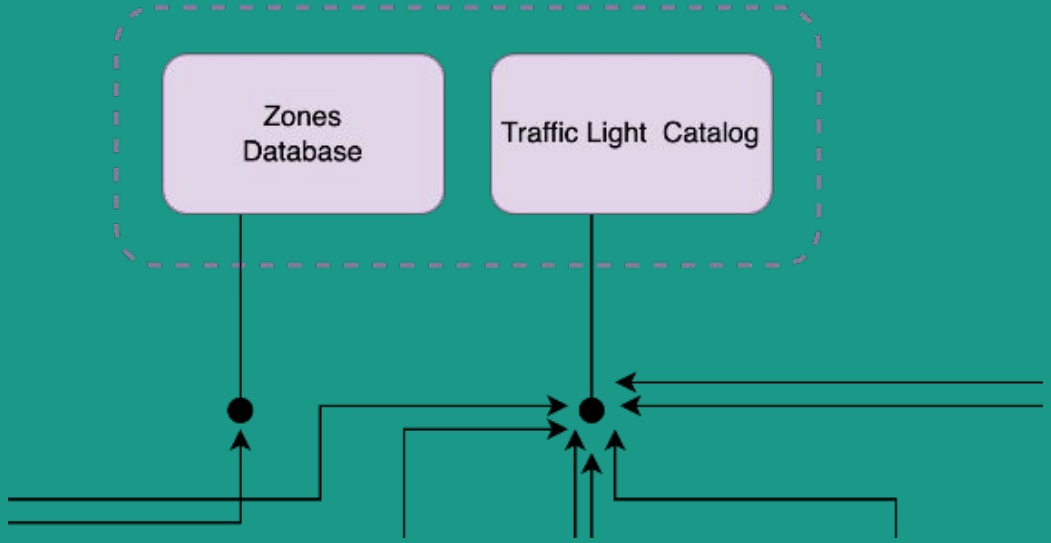
<base topic>/<...>/<zone>/<crossroad>

Led Sensor

example:



Service and resource catalogs



Service and Resource Catalogs

- Service and Resource catalogs are implemented.
- Communication with all other devices/resources via REST.
- Periodical registration of actors on Resource catalog.
- Periodical registration of Resource catalog on Service catalog

Server catalog info available to all actors.

```
{
    "ip_address_service":"192.168.1.8",
    "ip_port_service":"8080",
    "base_topic": "SmartTrafficLight"
}
```

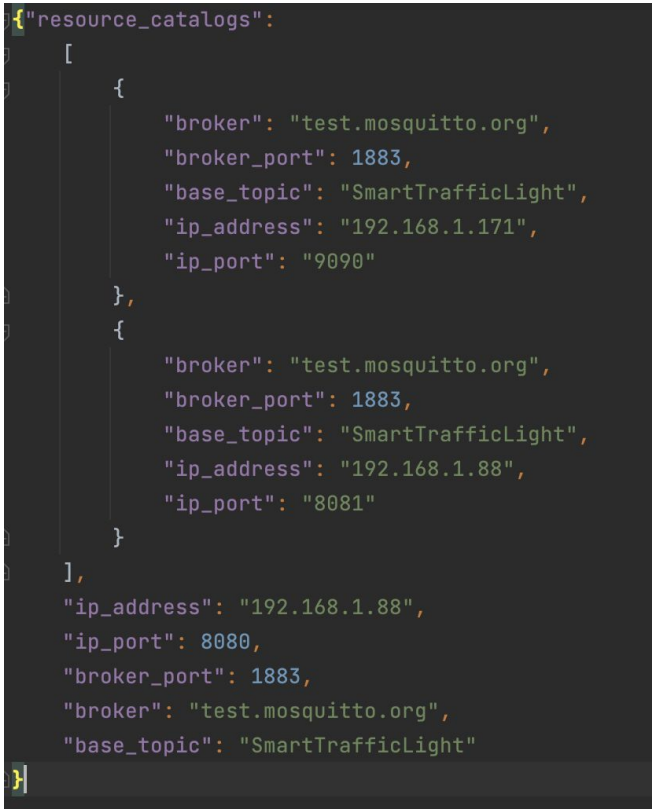
Resource catalog info provided by service catalog to all actors.

```
{
    "broker": "test.mosquitto.org",
    "broker_port": 1883,
    "ip_address": "192.168.1.171",
    "ip_port": "9090",
    "base_topic": "SmartTrafficLight"
}
```

Service Catalog

GET: provides information about **resource catalog** (latest registered, full list) or **broker**

PUT: register updated version of **resource catalog**



Resource Catalog

Periodical registration to Service catalog (PUT requests)

PUT: register **resources** to catalog

GET: provides information about broker, full resources list, **Zone Database**

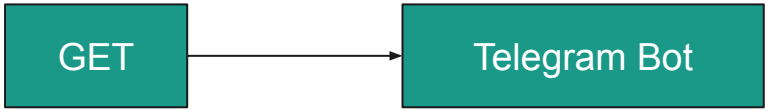
Resource Catalog

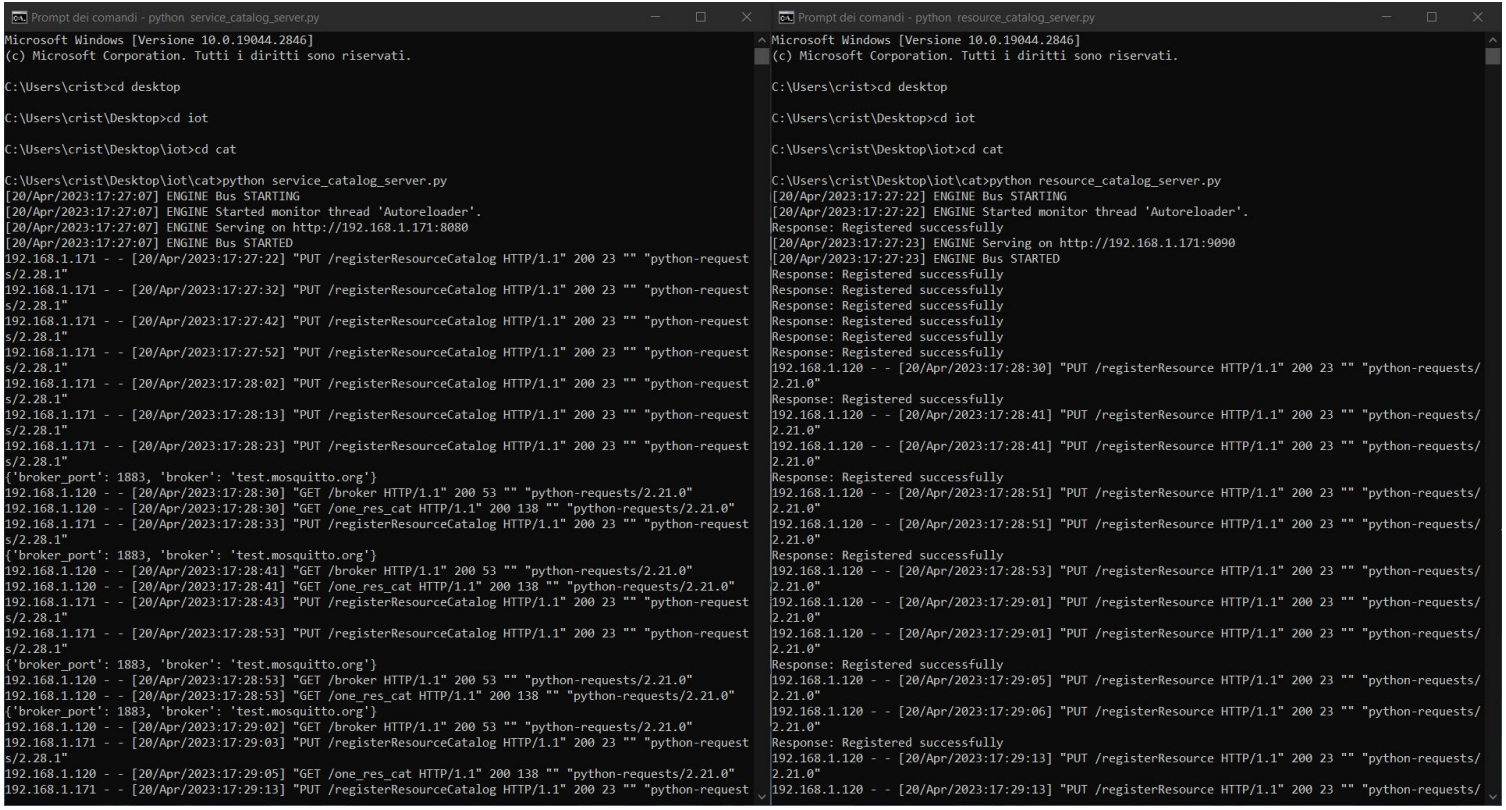
Zones Database

Zone Database includes all resources which are **malfunctioning** or whose registration has **expired**

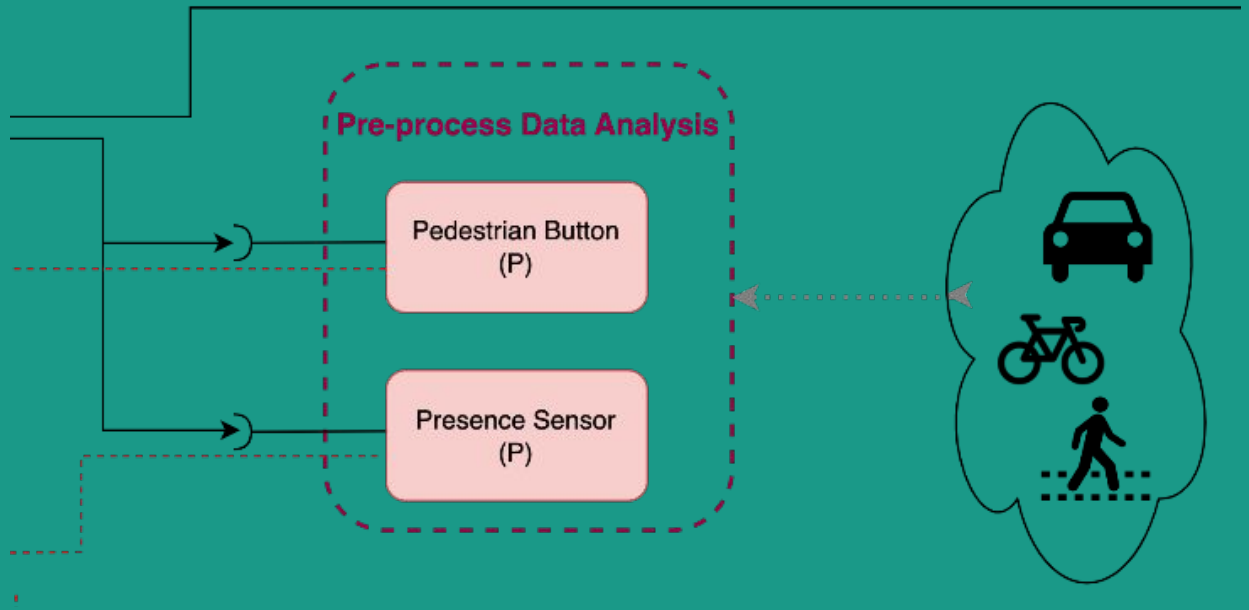
Zone Database is forwarded to manufacturers via **Telegram Bot** (REST)

```
if uri[0] == 'ZoneDatabase':
    # Retrieve the information Zone Database (all failures among devices)
    zone = uri[1]
    listZone = []
    for item in self.cat['resourcesList']:
        if item['zone'] == zone:
            # Check if resource's status is ok
            if item['status'] != 'OK':
                listZone.append(item)
            # Check if resource's registration has expired
            elif (time.time() - item['lastUpdate']) > EXPIRATION_TIMEOUT:
                item['status'] = 'EXPIRED'
                listZone.append(item)
    ZoneDatabase = {"zone": listZone}
    output = json.dumps(ZoneDatabase)
    return output
```




Prompt dei comandi - python service_catalog_server.py
Microsoft Windows [Versione 10.0.19044.2846]
(c) Microsoft Corporation. Tutti i diritti sono riservati.
C:\Users\crist>cd desktop
C:\Users\crist\Desktop>cd iot
C:\Users\crist\Desktop\iot>cd cat
C:\Users\crist\Desktop\iot\cat>python service_catalog_server.py
[20/Apr/2023:17:27:07] ENGINE Bus STARTING
[20/Apr/2023:17:27:07] ENGINE Started monitor thread 'Autoreloader'.
[20/Apr/2023:17:27:07] ENGINE Serving on http://192.168.1.171:8080
[20/Apr/2023:17:27:07] ENGINE Bus STARTED
192.168.1.171 - - [20/Apr/2023:17:27:22] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:27:32] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:27:42] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:27:52] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:28:02] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:28:13] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:28:23] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
{'broker_port': 1883, 'broker': 'test.mosquitto.org'}
192.168.1.120 - - [20/Apr/2023:17:28:30] "GET /broker HTTP/1.1" 200 53 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:28:30] "GET /one_res_cat HTTP/1.1" 200 138 "" "python-requests/2.21.0"
192.168.1.171 - - [20/Apr/2023:17:28:33] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
{'broker_port': 1883, 'broker': 'test.mosquitto.org'}
192.168.1.120 - - [20/Apr/2023:17:28:41] "GET /broker HTTP/1.1" 200 53 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:28:41] "GET /one_res_cat HTTP/1.1" 200 138 "" "python-requests/2.21.0"
192.168.1.171 - - [20/Apr/2023:17:28:43] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.171 - - [20/Apr/2023:17:28:53] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
{'broker_port': 1883, 'broker': 'test.mosquitto.org'}
192.168.1.120 - - [20/Apr/2023:17:28:53] "GET /broker HTTP/1.1" 200 53 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:28:53] "GET /one_res_cat HTTP/1.1" 200 138 "" "python-requests/2.21.0"
{'broker_port': 1883, 'broker': 'test.mosquitto.org'}
192.168.1.120 - - [20/Apr/2023:17:29:02] "GET /broker HTTP/1.1" 200 53 "" "python-requests/2.21.0"
192.168.1.171 - - [20/Apr/2023:17:29:03] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-requests/2.28.1"
192.168.1.120 - - [20/Apr/2023:17:29:05] "GET /one_res_cat HTTP/1.1" 200 138 "" "python-requests/2.21.0"
192.168.1.171 - - [20/Apr/2023:17:29:13] "PUT /registerResourceCatalog HTTP/1.1" 200 23 "" "python-request
Prompt dei comandi - python resource_catalog_server.py
Microsoft Windows [Versione 10.0.19044.2846]
(c) Microsoft Corporation. Tutti i diritti sono riservati.
C:\Users\crist>cd desktop
C:\Users\crist\Desktop>cd iot
C:\Users\crist\Desktop\iot>cd cat
C:\Users\crist\Desktop\iot\cat>python resource_catalog_server.py
[20/Apr/2023:17:27:22] ENGINE Bus STARTING
[20/Apr/2023:17:27:22] ENGINE Started monitor thread 'Autoreloader'.
Response: Registered successfully
[20/Apr/2023:17:27:23] ENGINE Serving on http://192.168.1.171:9090
[20/Apr/2023:17:27:23] ENGINE Bus STARTED
Response: Registered successfully
Response: Registered successfully
Response: Registered successfully
Response: Registered successfully
Response: Registered successfully
Response: Registered successfully
192.168.1.120 - - [20/Apr/2023:17:28:30] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
Response: Registered successfully
192.168.1.120 - - [20/Apr/2023:17:28:41] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:28:41] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
Response: Registered successfully
192.168.1.120 - - [20/Apr/2023:17:28:51] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:28:51] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
Response: Registered successfully
192.168.1.120 - - [20/Apr/2023:17:28:53] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:29:01] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:29:01] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
Response: Registered successfully
192.168.1.120 - - [20/Apr/2023:17:29:05] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:29:06] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
Response: Registered successfully
192.168.1.120 - - [20/Apr/2023:17:29:13] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/2.21.0"
192.168.1.120 - - [20/Apr/2023:17:29:13] "PUT /registerResource HTTP/1.1" 200 23 "" "python-requests/

Sensors


Pre-process Data Analysis
Pedestrian Button
(P)
Presence Sensor
(P)

Pedestrian Button

Push the button

Triggers the traffic light cycle (if turned off)
→ Pedestrian protocol

Publish (MQTT) on topic **SmartTrafficLight/Sensor/A/1**

REST connection to Traffic light catalog

Scalability ensured through **JSON settings files**: same code for more than one sensor

Presence Sensor

Detection of cars approaching crossroad

Triggers the traffic light cycle (if turned off)
→ Cars protocol

Publish (MQTT) on topic **SmartTrafficLight/Sensor/A/1**

REST connection to Traffic Light Catalog

Scalability ensured through **JSON settings files**: same code for more than one sensor

TrafficLight Temperature Sensor

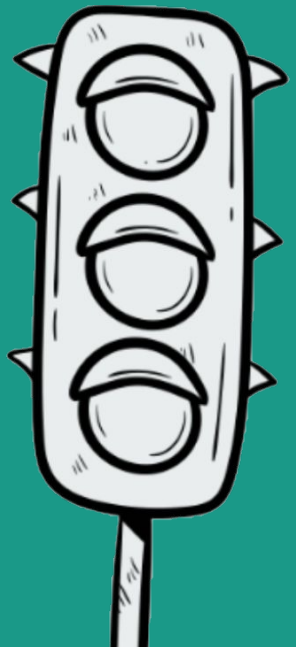
It controls functioning of Traffic Lights

Implemented in LED actuator code

If it detects **too high** temperatures
→ change "**status**" of Traffic Light

```
def register(self):
    # Check temperature is under emergency threshold
    humidity, temperature = Adafruit_DHT.read(self.led_ctrl, self.led_ctrl_pin)
    if humidity is not None and temperature is not None:
        print(f'Temperature of the traffic light = {temperature}')
        if temperature > 80:
            # Overheated traffic light: malfunctioning detected
            data = json.load(open(self.led_info))
            data["status"] = "Malfunctioning"
            json.dump(data, open(self.led_info, "w"))
            print("Traffic light overheated, malfunctioning detected.")
        else:
            # Temperature under control: traffic light correctly functioning
            data = json.load(open(self.led_info))
            data["status"] = "OK"
            json.dump(data, open(self.led_info, "w"))
```

Traffic light protocols



STOP!!!

WAIT!!!

GO!!

Car Protocol

Presence sensor positioned 100m before the traffic light

On car detection:

Turn on Traffic Lights of the **whole zone**

Start: **green** for cars

Traffic cycle: 20s before turning off

5s **Green** -> 5s **Red**

Timer and duty cycle can be modified through a practical **JSON configuration file**

Pedestrian protocol

On pedestrian request (button push)

Turn on **only** Traffic Lights of that **crossroad**

Start: **green** for pedestrians

Traffic cycle: 20s before turning off

5s **Green** -> 5s **Red**

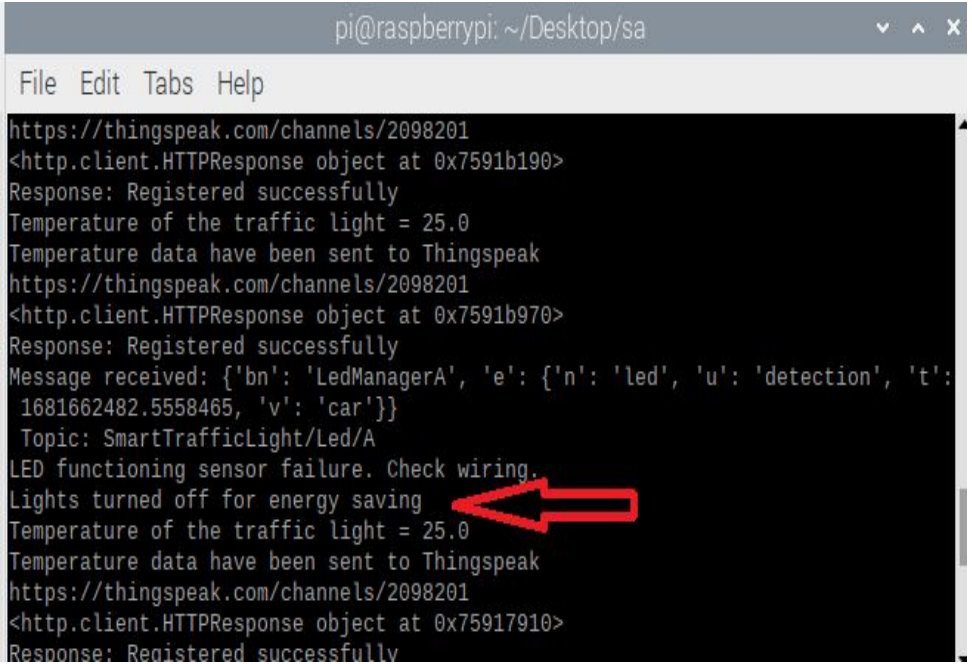
Timer and duty cycle can be anyway modified through a practical **JSON configuration file**

Energy saving protocol

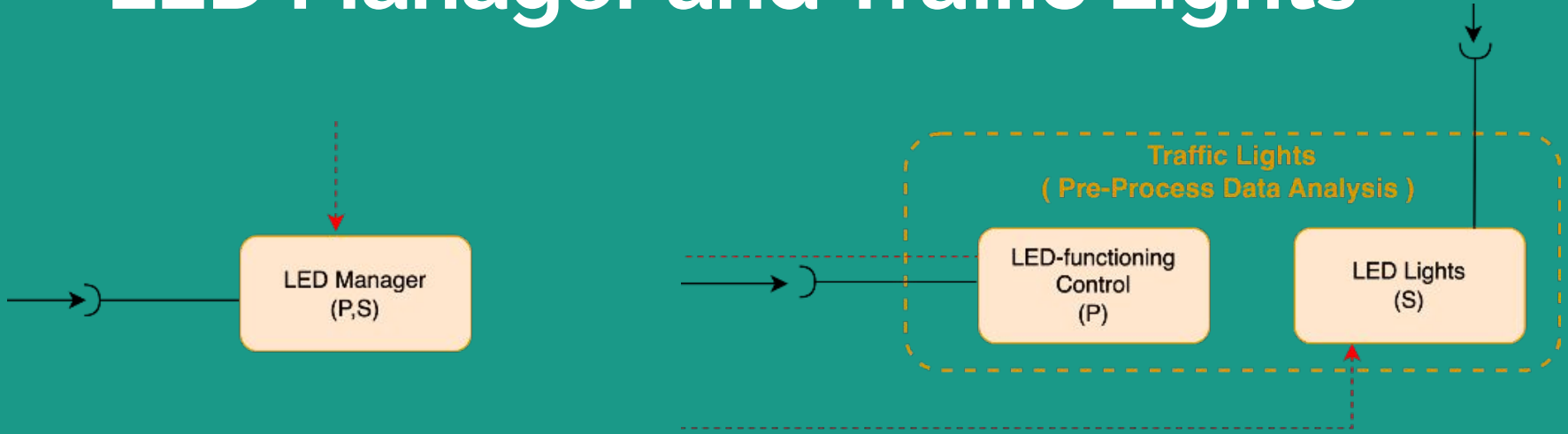
After **timer expiration:**

Turn **off** traffic light for **energy saving**

New detection → Start correct protocol



LED Manager and Traffic Lights



Traffic Lights

The led actuator → Turns on/off the led of crossroad

Subscribes to topics:

- **SmartTrafficLight/Led/A** (common to all zone A)
- **SmartTrafficLight/Led/A/1**

LED Manager

Zone-specific

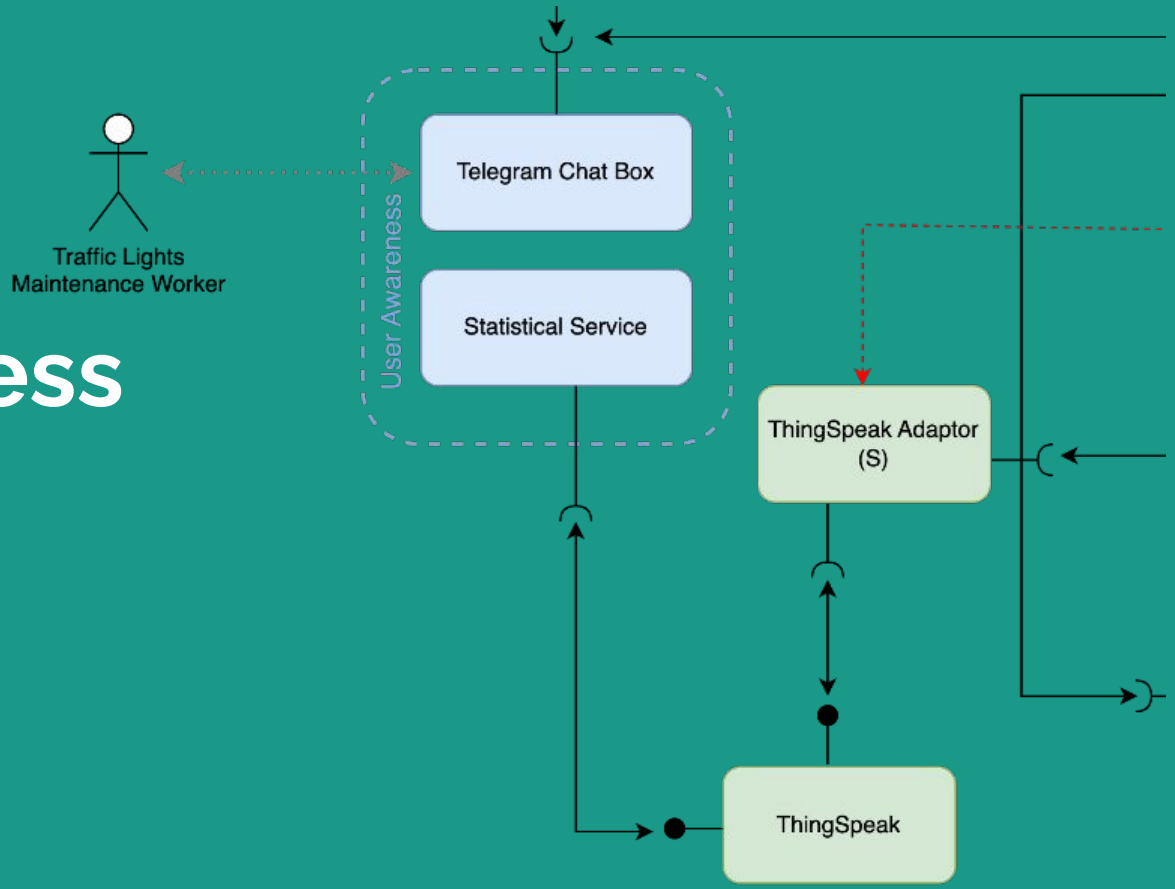
Communicates to all LED actuators

Subscribes to topic **SmartTrafficLight/Sensor/A/#**

Publishes to topic (if car detection):
SmartTrafficLight/Led/A (common to all zone A)

Publishes to topic (if pedestrian on crossroad *c*):
SmartTrafficLight/Led/A/*c*

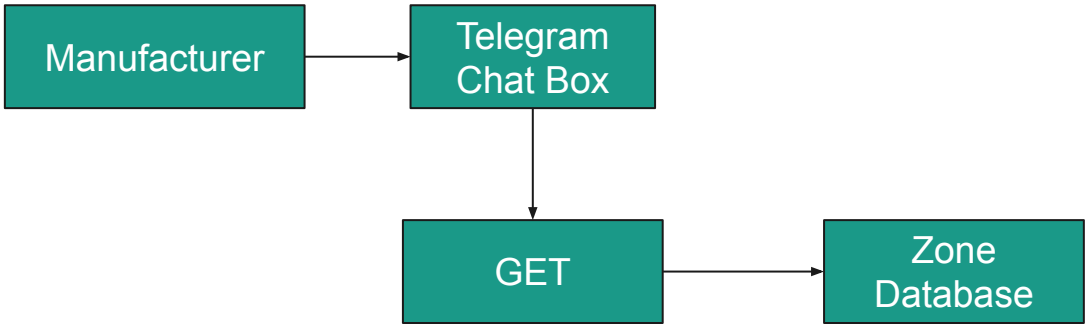
User Awareness

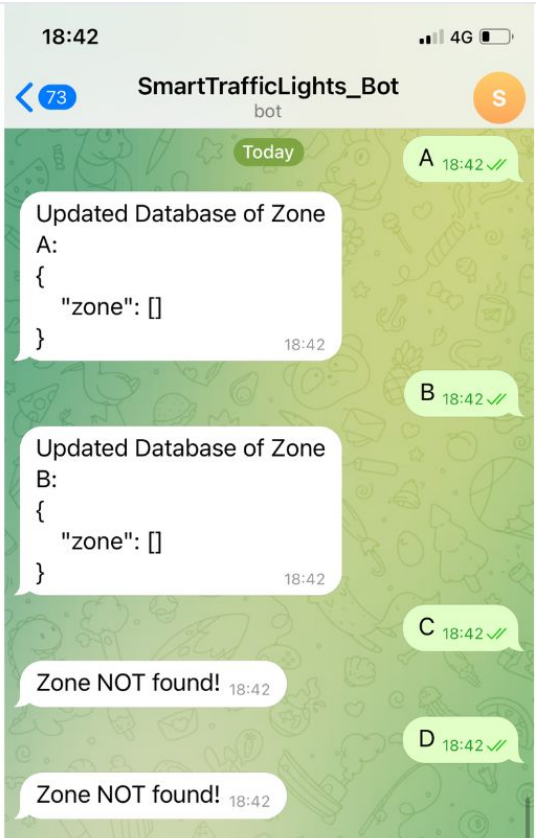


Telegram Bot

Allows fast and efficient maintenance

On demand it provides **Zone Database** containing a list of all the **malfunctioning traffic lights** of a **specific zone**





Thingspeak

- Distributed Thingspeak Adaptor:

  thingspeak_post method for every sensor and actuator

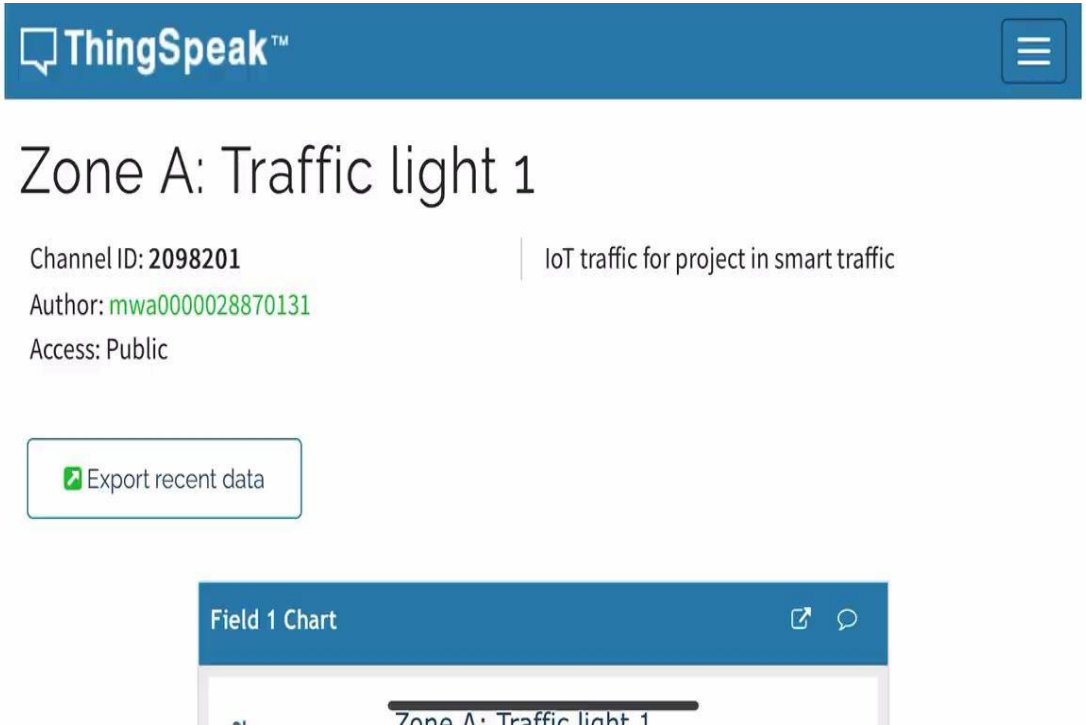
- ThingSpeak provides:
  - Data analysis over time
  - Pedestrians and cars detection rate
  - Traffic peaks analysis
  - Temperature over time

```
def thingspeak_post(self):
    val = 2  # Value 2 for pedestrians
    URL = self.base_url  #This is unchangeble
    KEY = self.key  #This is the write key API of your channels

    #field one corresponds to the first graph, field2 to the second ...
    HEADER = '&field1={}'.format(val)

    NEW_URL = URL+KEY+HEADER
    URL_read = self.url_read
    print('A pedestrian has been detected. Thingspeak link: \n' + URL_read)
    data = urllib.request.urlopen(NEW_URL)
    print(data)
```

ThingSpeak demonstration



Smart Traffic Lights

Traffic for a better world!

1) Energy Saving
2) Traffic statistics
3) Traffic control
4) Clear interface

Thank You!

Francesca Fanelli – s313432

Aaron Segers – s306765

Cristiano Vittori – s316801

Maureen Zwart – s307021