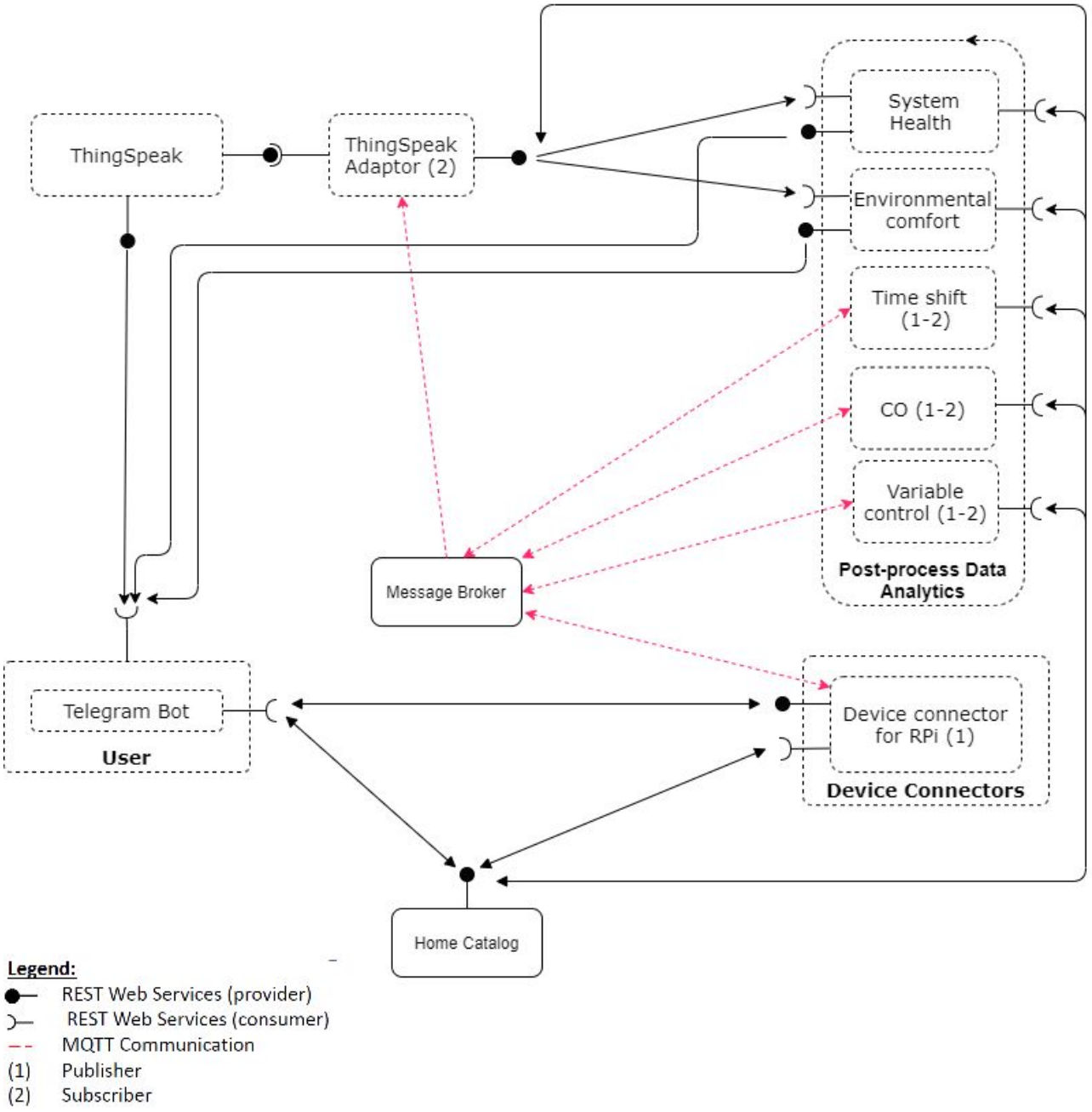
1 Name of Use Case

Name of the Use Case	**IoT platform for Smart Building**
Version No.	V0.1
Submission Date	08/12/2020
Team Members (with student ids)	Davide Delucchi (S286251), Manfredi Di Lorenzo (S288270), Martina Faggiano (S290352), Salvatore Failla (S288250)

2 Scope and Objectives of Function

Scope and Objectives of Use Case	
Scope	The proposed IoT platform aims at providing services for smart home management.
Objective(s)	The expected results consist on providing a smart control of the temperature of internal environments in a smart way to minimize the energy waste and maximize room's comfort.
Domain(s)	Smart Home, Smart Building.
Stakeholder(s)	Home inhabitants, Companies, Energy aggregators
Short description	The proposed IoT platform aims at making smart our buildings. It integrates different IoT devices for managing building's rooms from the point of view of cooling and heating. It provides air comfort's monitoring, detection of dangerous situations and energy saving. The overall platform provides unified interfaces (through both REST and MQTT) to integrate the home into Smart Building environments. Summarizing, the main features it offers are: ● remote control of cooling and heating; ● control strategies for heating systems; ● unified interfaces (i.e. REST Web Services and MQTT queues) available to enable Demand/Response; ● end-user applications for setting and scheduling the temperature.

3 Diagram of Use Case



4 Complete description of the system

The proposed IoT platform for Smart Home follows the microservices designing pattern. It also exploits two communication paradigms: i) publish/subscribe based on MQTT protocol and ii) request/response based on REST Web Services.

In this context, ten actors have been identified and introduced in the following:

- The **Message Broker** provides an asynchronous communication based on the publish/subscribe approach. It exploits the MQTT protocol.
- The **Home Catalog** works as a service and device registry system for all the actors in the system. It provides information about end-points (i.e. REST Web Services and MQTT topics) of all the devices, resources and services in the platform. It also provides configuration settings for applications and control strategies (e.g. timers, list of sensors and actuators). Each actor, during its start-up, must retrieve such information from the Home Catalog exploiting its REST Web Services.
- The **Raspberry Pi Connector** is a Device Connector that integrates into the platform raspberry pi boards. The raspberry is equipped with temperature, humidity and CO2 sensors to provide environmental information about the status of a room and three relays to control heating and cooling system and an air purification system. It provides Rest Web Services to retrieve environmental information (i.e. temperature, humidity, air quality). It also works as an MQTT publisher sending environmental data (every 5 minutes).
- The **Variable Control** is a system to manage appliances depending on environmental data like the temperature provided by the Home Catalog. For example if the temperature in the room is lower than the set one the system turns on the heating. It works as an MQTT publisher to send actuation commands to IoT Devices and as an MQTT subscriber that receives measurements on environmental measurements.
- The **CO** is a control strategy to manage appliances depending on CO level in the room. For example, if the level of CO in the room is too high the air purification system is powered on and a notification is sent to the user using telegram bot. It works as an MQTT publisher to send actuation commands to IoT Devices and as an MQTT subscriber to receive the CO level from the **Device Connector**. The algorithm has two threshold levels that are set to 1000 ppm and 2000 ppm.
- The **Time Shift** is a control strategy to manage appliances depending on temperature-schedules provided by the Home Catalog. For example, it allows users to set different temperatures during the different hours of the day. It uses REST Web Service to set the desired temperature of the room in the Home Catalog. It works as an MQTT publisher to send actuation commands to IoT Devices and as an MQTT subscriber that receives measurements on environmental measurements.
- The **System Health** is a microservice that analyzes the collected data and checks if there are some problems in our system. For example if the temperature in the room goes down for a certain amount of time and the heating system is powered on it means that there is a problem. If some issues occur it sends a notification to the user through the **Telegram Bot** using REST Web Service.
- The **Environmental comfort** is a service that analyzes the mean temperature, humidity and CO2 level of a certain period of time and gives a valuation of the room's comfort. For example if the humidity is too high we obtain the valuation: "too humid" and some tips to improve the comfort like "Mold risk! Open the windows ten minutes per day". To evaluate the comfort we consider the ideal values of 20° for the temperature and 50% for the humidity and we measure the deviation from these values. It works as a REST Web Service used by the **Telegram Bot.**
- The **Thingspeak Adaptor** is an MQTT subscriber that receives measurements on environmental measurements and upload them on **Thingspeak** through REST Web Services.

- **Thingspeak** is a third-party software (https://thingspeak.com/) that provides REST Web Services. It is an open-data platform for the Internet of Things to store, post-process and visualize data (through plots).
- **Telegram Bot** is a service to integrate the proposed infrastructure into Telegram platform, which is cloud-based instant messaging infrastructure. It retrieves measurements from IoT devices exploiting the REST Web Services provided by **Raspberry Pi** and **Arduino Connectors**. It also allows users to set the desired temperature or turn on/off the heating/cooling system again exploiting REST.