Shared Internet-of-Things Infrastructure Platform (SIoTIP)

Part 1: domain description

This project involves the development of a shared Cyber Physical System (CPS) infrastructure. The resulting system is an Internet of Things (IoT) platform used and maintained by various stakeholders. The system consists of gateways communicating with plug-and-play sensors and actuators spread around one or more buildings, and an online back-end that synchronises with these gateways.

This document describes the project, the problem space, and the existing technological and business constraints. Section 1 provides some background about the domain and introduces the envisioned platform. Subsequently, Section 2 lists the main stakeholders involved in our system. This is followed by Section 3, which zooms in on the details of the system under design. Section 4 describes some example scenarios of how the stakeholders could interact with the system. Finally, Section 5 presents the concrete instructions for this assignment.

1 Context

Building control

Building automation technologies can provide automation of the management and control of building facilities, in order to provide increased business value, convenience, comfort, energy efficiency, and security. This may include centralised control of lighting, HVAC (heating, ventilation and air conditioning), appliances, security locks of gates and doors, and other systems.

The popularity of such automation systems has, both in industrial and domestic contexts, increased significantly in recent years due to much higher affordability of hardware, as well as increased interconnectivity and simplicity via mobile devices such as smartphones and tablets.

Figure 1 provides a schematic overview of the set-up of a building automation system. It illustrates a number of **sensors** and **actuators** physically plugged into so-called **motes**, as installed in two distinct buildings. These devices connect to **gateways**, which in turn exchange information with an online back-end. Customer organisations can interact with the available sensors and actuators via specialised applications. Such applications enable monitoring (part of) a building, e.g. the temperature in a server room, or automating aspects of a building, e.g. heating or light management. Furthermore, certain alarming events, e.g. fire or break-ins, can be automatically reported to the appropriate entities, such as the fire department or security agents.

A shared IoT infrastructure

The **Internet of Things** paradigm allows for complex system compositions involving multiple stakeholders. Yet, the true potential of the IoT is unlocked if devices can be shared between multiple applications. Hence, the interests of all involved parties, e.g. those installing the IoT devices and those developing applications on top of the platform, have to be considered and integrated as much as possible. Obviously, the owners of the IoT platform also have their own interests.

The building control system described above can be deployed on top of such a shared IoT platform. The advantage of this infrastructure is the integration of technologies which are commonly thought of as separate systems, such as fire control and heating systems.

Sensor devices such as detectors for light conditions and motion, and actuators for locking/unlocking doors and controlling light bulbs can be expected to be integrated into this system. Applications can manipulate the same infrastructure elements in different ways. For example, one application could control

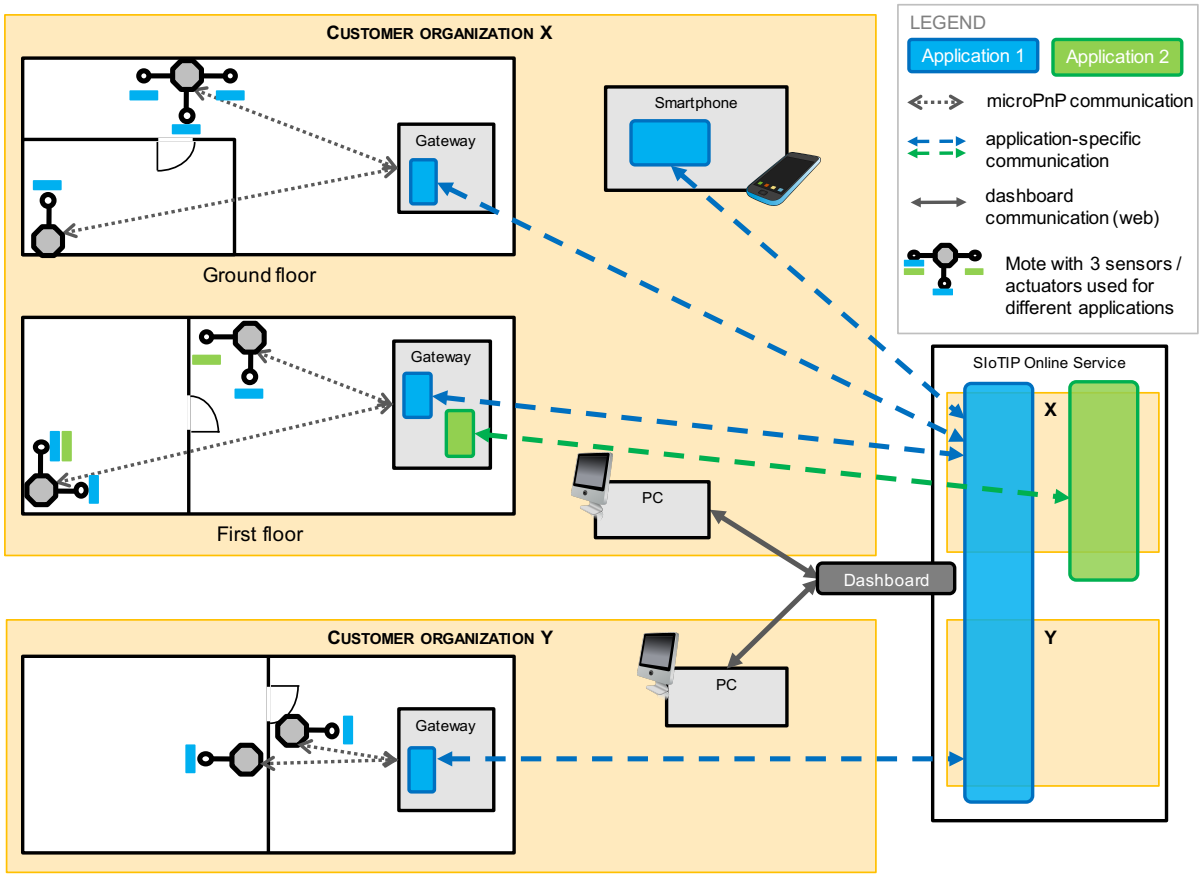

Figure 1: High-level overview of a building control system installation.

the door lock based on access cards. Another application may decide to open that same door automatically when fire is detected, in order to evacuate the building smoothly. Similarly, a passive infrared presence sensor can be used by a light management application to turn on the lights in a room when someone enters, but also by a burglary detection application to detect break-in attempts at night.

Availability and reliability of the shared platform offered to applications is of paramount importance. Therefore, potential redundancies can be leveraged. For example, a room could have multiple independent temperature sensors, such that applications need not solely rely on only one such device. Furthermore, several sensors or actuators can be used for the same goal. For example, both an infrared sensor and microphone sensor can be used to detect presence of people.

Positioning

In this assignment, we assume the point of view of a company that develops the software for a shared IoT platform, more specifically the platform software for the gateways and the online back-end. We primarily target industrial settings, such as airports or a company with geographically distributed plants, in which business value can be increased by more closely monitoring and controlling business assets or (further) automating their business processes. For example, the temperature and humidity in server rooms can be automatically monitored and adjusted, or physical access to protected areas can be controlled. In the future, other business applications are likely. For example, companies could automate their production process, thereby avoiding the need for dedicated, more expensive Industrial Control Systems (ICS). Furthermore, the domestic market, i.e. home automation, is continuously growing thereby providing extra revenue opportunities.

Note that a CPS does not necessarily has to be connected to the internet, but it could be a secluded system (e.g. confined to an airport building). In practice, however, the connection between gateways and

the online back-end will commonly use regular internet links, and the IoT devices will use standardised Internet technologies such as IPv6.

Business model. The shared CPS platform that is developed will be named "**SIoTIP**" (Shared Internet of Things Infrastructure Platform), and our company "the SIoTIP Corporation". Our goal is to sell SIoTIP gateways and additional hardware devices, such as sensors, enabling the deployment of a (shared) IoT infrastructure. Furthermore, third-party developers can build applications on top of the available infrastructure.

We will operate a service model, where the main revenue is expected to be generated by the services provided via our online back-end (called "**Online Service**" in the remainder of this description). The actual services provided are twofold. On the one hand, companies want to increase their business value by automating or simplifying building management using IoT devices. We sell them all the hardware required (i.e. gateways, motes, sensors, and actuators) to easily set up their own IoT infrastructure. Furthermore, we provide them with an inexpensive, flexible and reliable platform via which they can utilize their installed infrastructure via applications. On the other hand, application providers want to be able to develop and sell applications leveraging the available infrastructure to these companies. Since such applications are often domain-specific and highly specialised, they can best be developed by domain experts. Therefore, we offer such third-party developers a reliable, robust platform on top of which a wide range of different applications can be built, as well as a platform where application providers and potential buyers can meet at low entry cost.

Over time our platform will collect significant amounts of data: raw data from sensors as well as application (usage) information. This data trove can be utilised to improve our provided services, but also provides big data analysis opportunities we can offer to other (new) parties (research, government).

Initially SIoTIP is oriented towards the Belgian market, aiming to provide 50 different applications and install our hardware at 200 organisational customers by the end of the first year. In the longer term, the ambition is to provide our services across the globe.

The SIoTIP platform

The SIoTIP platform consists of two distinct but complementing parts: the gateways and the Online Service. Several gateways can be used within a single environment, each of them associated with their own set of sensors and actuators. For example, different floors or local sites of a large plant can each have their own gateway. Currently, a single Online Service is maintained by the SIoTIP cooperation.

In this model, sensors and actuators connect to a gateway, and all gateways connect to the Online Service to synchronise data. Most synchronisation activities are initiated by the gateway, e.g. to push their data and pull data and configuration updates from the Online Service. However, the Online Service must also be able to push application commands to applications running on the gateways.

Application support is one of the primary functionalities that SIoTIP should provide. The gateways are embedded the devices that have the ability only to run light-weight programs that are not too demanding in terms of resources (CPU, memory, storage). The gateway has direct access to the attached devices and their data, while the Online Service can only access these devices indirectly via the gateways. The Online Service has no practical limitations in terms of processing power and data storage.

Technology stack

The SIoTIP system could be made compatible with a wealth of IoT technologies. Initially (and for this assignment) we will use MicroPnP devices which operate via a mesh network. The MicroPnP technology was originally developed at imec-DistriNet, and is now marketed by the KU Leuven spin-off VersaSense[1].

The following building blocks will be used:

MicroPnP mote (or just mote) is a device that can host sensors and actuators, and which connects these to the mesh network. As illustrated in Figure 2, motes are powered either by a battery or they can be plugged into an electrical or light socket. Each MicroPnP mote has three USB-like peripheral connectors into which sensors and actuators can be hot-plugged.

[1] `https://www.versasense.com`

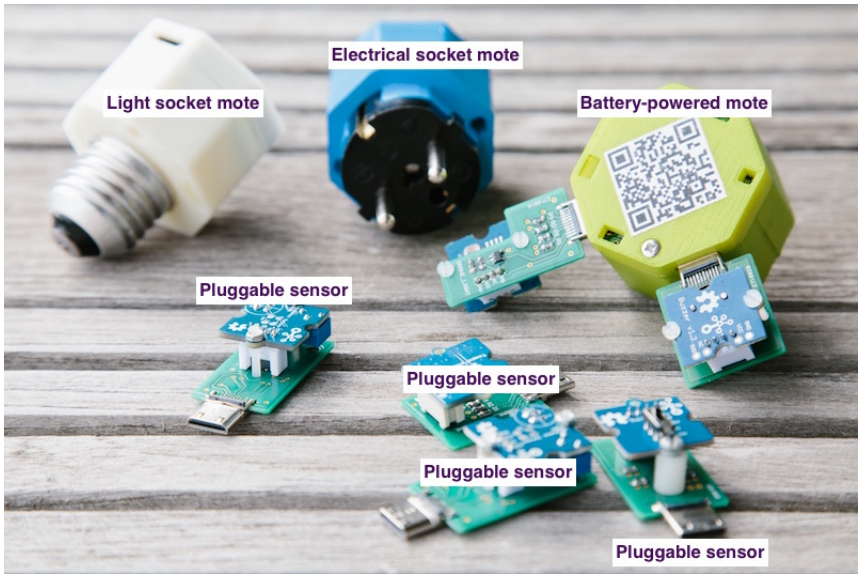

Figure 2: MicroPnP motes and sensors

MicroPnP network manager is a device which offers connectivity to the mesh network used by the motes and which can also be connected to a secondary network (e.g. a wired network). It runs an embedded operating system and can be used to run custom programs (e.g., written in Java). However, computing resources are fairly limited. By default, it contains software to access the motes and reliably send and receive messages via the mesh network. Hence, via the software on this device, the motes, sensors, and actuators can be accessed. For connectivity, the 6LoWPAN[2] technology is used. Pairing motes with network managers is done out-of-band, by using a cable to exchange network information.

SIoTIP gateway is a component of the SIoTIP system with access to the sensors and actuators, and which is connected to the Online Service. The gateway relays the data from connected devices to the Online Service, manages the sensors and actuators, and runs some application logic (i.e., relatively simple, rule-based condition-response logic). **The MicroPnP network manager is also used as SIoTIP gateway.** This device provides the necessary capabilities to allow running SIoTIP gateway software run on top of the network manager's embedded operating system (and is in line with how the hardware components are depicted in Figure 1).

Sensor is a hardware device that produces measurements and sends them to the gateway via a MicroPnP mote. The sensors are physically plugged into a mote (see Figure 2).

Actuator is a hardware device that has one or more actions associated with it. For example, a switch can "turn on" and "turn off". The actuators are also plugged into the MicroPnP motes.

SIoTIP Online Service is an online system which collects data collected by the sensors at each gateway, runs applications, manages gateway and application configurations, and provides a web portal to which users can connect.

2 Main stakeholders

This section provides an overview of the main stakeholders in our system and their points of interest.

Customer organisations. The customer organisations' main interest is that they want to subscribe to the applications to increase their business value. Therefore, they require a hassle-free experience in subscribing to applications available to them. For example, if new hardware, e.g. extra sensors, is required,

[2] `https://datatracker.ietf.org/wg/6lowpan/charter/`

the purchase and installation of this hardware should happen as transparently as possible from the point of view of the customer organisation. Moreover, applications should be activated if all their requirements are satisfied without requiring any further interactions from the customer organisation. Similarly they want to be able to unsubscribe from applications they no longer need or use. Finally, customer organisations should be able to get an overview of the invoices for their application subscriptions. Examples of customer organisations are the fire department and security departments of an airport, or an operator controlling a plant or subsystem in an industrial context.

Infrastructure owner. This stakeholder owns and manages the physical infrastructure, e.g. the building(s), as well as the gateways, motes, sensors and actuators installed in it. They want to be able to buy new hardware, e.g. to add new sensors or replace broken ones. Note that installing new sensors or actuators can be triggered by customer organisations that want to subscribe to a new application or as a result of changes to an already-deployed application. Owners also want to be capable of monitoring whether all their connected sensors and actuators function correctly. Thus, they want to be notified in case their installed hardware fails, for example if one or more sensors stop sending data.

Furthermore, owners are responsible for maintaining a topology of their infrastructure, this is a virtual view on where devices are installed throughout a building and documents any relevant relations between them. Note that motes and the sensors/actuators attached to it can generally be assumed to be physically in the same location. However, in the future it may be possible to hook up some motes to nearby devices via a wire (e.g. for door/window sensors). A possible relation captured in the topology is that two sensors are redundant with respect to each other. For example, an infrared sensor and microphone sensor installed in the same room can both be used to detect presence of people in this room. Note also that the possible relations are often application-specific and thus are likely to change over time. The topology is essential for the correct functioning of many applications. For example, in an access control application the correct doors must be unlocked or a fire detection application must activate the correct sprinklers in case of fire. The information in the topology also allows SIoTIP to provide some reliability. Knowledge of the position of the sensors and actuators can be used to automatically compare a sensor reading to values of similar devices that are nearby and this way, anomalies and/or faulty hardware may be detected and the appropriate stakeholder notified. Furthermore, applications can switch to a nearby and similar device, based on the relations in the topology, if the one they are listening to fails.

Furthermore, owners want to allocate their installed sensors and actuators to specific customer organisations. For example, an infrastructure owner renting office space in a building to several companies can decide that each company (i.e. customer organisation from the point of view of SIoTIP) can only use sensors and actuators installed in their offices and those installed in common areas such as hallways. Note that sensors and actuators can be shared between multiple customer organisations, while each piece of hardware has exactly one owner.

End-users. These stakeholders actively interact with the deployed applications. Such interactions can, depending on the application, be performed via a plethora of devices, e.g. a mobile app on a smartphone or a web browser on a personal computer. For example, a security guard can provide intermediate reports, via a mobile application, at certain checkpoints on his or her round. For all end-users, the use of applications should simplify or automate (parts of) their daily tasks, while the presence of sensors and actuators should not hinder them. People who simply interact with SIoTIP by influencing sensors only passively or unknowingly use the applications and are not considered to be end-users.

MicroPnP mote manufacturer. This stakeholder manufactures and sells the MicroPnP motes which serve as hubs for plugging in sensors and actuators.

Sensor and actuator manufacturer. This stakeholder produces and sells the sensors and actuators compatible with the MicroPnP technology, which can be installed in combination with SIoTIP.

Application providers. The main goal of this stakeholder is to sell (subscriptions to) applications running on top of SIoTIP to customer organisations. These providers often have expertise in a specific domain, e.g. building-level physical security, and thus develop specialised applications for their domain. Therefore, they require a development environment providing access to APIs provided by SIoTIP (e.g.

to facilitate communication between the Online Service and a gateway) as well as the possibility to debug applications. This allows them to focus solely on the development and maintenance of their applications and not on the underlying hardware such as gateways or specific sensors, or the exact communication protocols.

Furthermore, it must be possible to update already-provided applications in a smooth and user-friendly manner, for example, to add new features to an application or to fix discovered bugs. From the point of view of the application provider, this should not be more complex than adding a new application. Thus, SIoTIP has to support updating running applications without requiring intervention from the application provider.

SIoTIP system administrators. The system administrators are responsible for monitoring and maintaining SIoTIP for its correct working. This includes, among others, the performance and scalability of the Online Service. They are also responsible for assisting other parties (such as infrastructure owners) in case they have problems with SIoTIP. They are notified if alarming events threaten the platform, e.g. failure of internal components or a failure to communicate with one or more gateways.

The SIoTIP Corporation. This stakeholder want SIoTIP to be a successful platform for deploying IoT devices. Hence, they want a large number of infrastructure owners to make their infrastructure available for customer organisations. The majority of revenue will be created by these customer organisations that use applications on their platform. Therefore, they also want to attract many application providers to develop high-quality applications for the platform.

Telecom operators. Telecom operators provide means for the gateways to communicate with the Online Service and vice versa. A service level agreement between our company and the telecom operators determines the capabilities of these communication channels.

Server providers. This stakeholder provides physical or virtual servers that will be used to deploy and operate the Online Service. A service level agreement between our company and the server providers determines the server up-time constraints and the reaction time in case of hardware problems.

3 Details of the problem domain

This section zooms in on the most important aspects and requirements of the envisioned platform.

3.1 Overall system goals and requirements

The main goal of the system is to provide a platform which gives customer organisations the ability to control the shared infrastructure through applications. To achieve a useful and reliable platform, the goals described in this section should be realised.

Applications. The primary goal of SIoTIP is to support the development and deployment of applications which leverage the IoT infrastructure accessible through the platform. As described, these applications are developed and maintained by third-party application providers and are often specialised to a specific domain. The main added value of SIoTIP is that it supports these providers in developing and deploying complex, distributed applications by taking care of the distribution of and communication between different parts of an application. For example, a smart heating application can be split into several parts. The local part on a gateway can monitor and, if needed, adjust the temperature in a room by interacting with the connected sensors and actuators. The algorithms used to learn the behaviour and daily routine of the people in a building typically requires large amounts of historic data, and are thus executed on the Online Service. Finally, some persons, e.g. the employees of the customer organisation, can be allowed to manually adjust the temperature via a mobile app (provided as part of the application as a whole) on their smartphone.

Therefore, SIoTIP must provide services or APIs to accommodate such a distributed application, and deal with all encompassing communication between different parts of an application.

Application providers expect a reliable platform to deploy their applications on, and for instance a single broken sensor should not necessarily stop the application entirely or even hamper it correct functioning. SIoTIP has to provide APIs both for code running on the gateway and on the Online Service. They must also allow code running inside SIoTIP to interact with external applications and systems (e.g. mobile apps). Therefore, SIoTIP provides a library which can be incorporated in mobile apps to simplify their development.

For customer organisations, applications are the primary way to interact with the system, and thus to leverage the installed sensors and actuators. Depending on the nature of the application, some people can be allowed to issue commands to the application via a provided front-end, e.g. a mobile app or web browser. These people are thus assigned the role of end-user from the point of view of SIoTIP.

SIoTIP gives application providers the possibility to update their applications to newer versions. They have the freedom to choose for updates where all existing instances of the applications are automatically updated to the new version (e.g. a minor bug fix update) or to require customer organisations to explicitly update to the new version (e.g. a major feature update which requires a new subscription). In both cases, SIoTIP should automatically update all parts of the affected application instances. Note that in the latter case, different versions of the same application can co-exist.

For application providers, a development and debugging environment is provided. This environment provides easy access to all APIs provided by SIoTIP. Furthermore, it allows to emulate a more realistic deployment environment, e.g. mimicking incoming sensor data, to test and debug the application under development.

The system has to check new or updated applications to ensure that they, for example, cannot cause a system crash or contain memory leaks. Therefore, when uploaded, each application is automatically tested in a sandbox environment. If all tests are successful, the application is automatically made available, and otherwise a SIoTIP system administrator performs a secondary review and decides whether to accept or reject the application. In the latter case, the application provider is also notified of the reason for rejection. Application providers are given the choice between automatically updating existing instances of the application, or requiring customer organisations to subscribe to them explicitly.

The system should monitor application execution. This yields information regarding the performance of applications, such as the amount of CPU and memory used by each application. This can be complemented with application-specific information, such as when an application is executing a learning algorithm and hence can be expected to utilise more resources. In case of problems (e.g. an application misbehaving due to a bug), the system must notify a system administrator, who can (temporarily) shut down affected applications.

IoT devices and their data As the system is a platform for a shared CPS, it should be possible to interact with the connected IoT devices. Ease of installation is an important concern in this context: new sensors, actuators, and motes should be registered to the gateway automatically, and made available to applications. The plug-and-play functionality of the devices helps in achieving this. Each device is initialised with a functional, default configuration, which can then be adjusted by the platform, application, or user (e.g. setting the sampling rate of a sensor). Infrastructure owners must be able to configure the topology of the hardware they are responsible for. Such a topology describes where devices gateways, motes, sensors and actuators are installed in a building and documents any relevant relations between them.

Availability of these devices should be monitored by SIoTIP. Therefore, each mote periodically sends a heartbeat message to its gateway listing the sensors and actuators plugged in to it along with a timestamp. In case of unavailability, applications and infrastructure owners can be notified.

Data from sensors and actuators needs to be stored on both the gateway and Online Service side. Gateways contain limited historical data from the connected devices and possibly even have to remove the oldest information when they are short on storage space. At the Online Service, historical information sent by each sensor and actuator is collected. This historical data allows applications to, for example, learn the behaviour of people working in a building or give a long-term historical overview of monitored conditions. Finally, this data presents interesting opportunities with respect to big data analysis for the SIoTIP Corporation as well as interested third parties.

Notifications and alarms. In case of events that are interesting to customer organisations or specific end-users, applications can trigger notifications and alarms. For example, a burglary detection application

could trigger an alarm when motion is detected at night. These can be sent to the recipient via a mobile app, e-mail, or SMS, depending on the capabilities of the application at hand. The customer organisation is able to configure the method of delivery. Furthermore, recipients can request an overview of previously-received notifications and alarms.

Failures in the infrastructure, e.g. a failing sensor, should be reported to the infrastructure owner. System components can trigger notifications for system administrators to inform them of system events that require their attention, such as a misbehaving application.

As alarms indicate critical events that may need immediate attention, their delivery should be as fast as possible.

Gateway - Online Service connection. Gateways establish a connection with the Online Service which remains open. Gateway installation involves connecting it to the internet via a secondary network interface (i.e., a wired or wifi network). The Gateway will then register itself with the Online Service. The Gateway identifies itself, and the Gateway identifier is used by the system to link it to the infrastructure owner that has originally purchased it (information which was registered on purchase). Afterwards, communication in both directions is possible, as described below.

A gateway should periodically synchronise data, e.g. sensor measurements or sensor configurations, with the Online Service. The synchronisation interval depends on the applications the respective customer organisations are subscribed to. It is essential that the synchronisation protocol works correctly in the presence of non-reliable network communication, i.e. that there is no loss of data or inconsistencies. Besides periodic synchronisations, important data such as alarms and notifications must be communicated to the Online Service as soon as possible.

The Online Service also has to communicate with the gateway to, for example, update sensor configurations or activate actuators. This is done as soon as the corresponding command is generated, which can be triggered by end-user input via an application front-end or the occurrence of certain events. For example, when fire is detected, all relevant buzzers (possibly spread over the entire building) should be activated to alert those present to evacuate the premises. If the command cannot be pushed immediately, it must be ensured that the command reaches its destination when the gateway becomes available again.

The Online Service must detect gateway failure and notify the system administrators and the infrastructure owner when such problems persist.

A reliable and robust platform. An important goal of SIoTIP is providing a reliable, adaptable platform to its users. This implies that unavoidable hardware failures should, where possible, be dealt with by leveraging redundancies in the deployed infrastructure. For example, when a passive infrared sensor fails, a burglary detection application could switch to a nearby microphone sensor to detect break-in attempts.

Security. Since the infrastructure is shared among multiple users, it is important to authenticate them, verify whether they are authorised to perform the requested actions and prevent users from (actively) influencing applications from others. More specifically, application providers, infrastructure owners and system administrators need to be authenticated. This could, for example, be achieved by a username and password combination or by a public-private key pair. The system also needs to ensure that customer organisations are only given access to the sensors and actuators that are exposed to them. Note that while applications themselves can employ a variety of security measures, these can not be counted on to provide security at the platform level.

3.2 Sensors and actuators

As mentioned, the sensors and actuators are provided by a third-party hardware provider, and initially we only collaborate with VersaSense hardware. Their sensors and actuators are plugged into MicroPnP motes which are organised in a mesh network, thus motes can use other motes as intermediaries in order for the data to reach the destination.

For resource access, the devices use interfaces compliant with the Constrained Application Protocol[3] (CoAP), which is a RESTful protocol. The motes offer their attached devices (sensors and actuators) as

[3] `https://tools.ietf.org/html/rfc7252`

CoAP resources that are available via GET/PUT. The content (e.g. the value provided by a temperature sensor) is encapsulated into a JSON message. The motes can emit the following types of messages:

1. *Data*. Motes use this to send information from one of the connected devices to the gateway (e.g. a temperature reading, buzzer is on, motion is detected). Apart from the value itself, the mote also adds the mote identifier and a timestamp. The message also contains an explicit "sensor/actuator type" indicating how the contained value must be interpreted. For example, a data message could be indicated to come from a temperature sensor measuring in degrees Celsius, allowing to correctly use the contained value and, for instance, not as degrees Fahrenheit.

2. *Mote heartbeat*. Every mote periodically sends a heartbeat to indicate it is alive. This includes a list of the sensors and actuators connected to it and a timestamp indicating when the heartbeat was generated.

3. *Device connected/disconnected*. Whenever a new sensor or actuator is inserted in the mote or removed from it, the mote immediately sends a notification to the gateway. This message specifies the sensor/actuator type of the inserted/removed device and a timestamp.

Relaying information to the motes by the gateway is done by GET and PUT requests. The former can be used to retrieve a list of attached devices or specific information about a specific sensor, actuator, or mote (e.g. a configuration parameter such as the sampling frequency). The latter is used to update specifications (e.g. set the sampling rate of a sensor) and to send actuation commands.

All sensors will relay new information to the gateway via their mote. This can be done periodically, e.g. for temperature readings, or triggered by state changes, e.g. detection of motion. In case of periodic communication, the sampling rate of the sensor can be configured and this rate will depend heavility on the nature of applications that depend on that sensor. Request-response based retrieval of information is also possible but is, with the currently supported devices, only used for configuration. For example, requesting the current sampling rate of a sensor.

The remainder of this section outlines the sensors and actuators currently supported by VersaSense as well as the accompanying MicroPnP support material. Since IoT is regularly applied to new domains and due to the constant technological evolution, it is very likely that over time new (types of) sensors and actuators have to be added. Also, integration with specialised sensors/actuators for domain-specific applications is possible. An example are sensors to detect water leaks and actuators to control valves in a water leak-detection application. In these cases, a new sensor/actuator type will be created and SIoTIP will need information on how to interact with them (e.g. writing value "1" to open a valve). Furthermore, it would be possible to integrate technologies other than MicroPnP provided by VersaSense in the future.

Table 1 gives an overview of the currently-available MicroPnP sensors and actuators which could be deployed on the SIoTIP and which would be useful for monitoring a building or other business contexts.

Different sensors can be used to achieve the same goals. For example, presence of people can be detected either using a passive infrared sensor or a microphone sensor. Similarly, fire can be detected using heat flux sensors and/or temperature and humidity sensors. For actuation, there currently are only two actuators to choose from. A buzzer actuator can be turned on and off to, for example, function as a warning signal. A power socket actuator can be used to control a plethora of devices plugged into the controlled socket. For example, this could control a light switch or activate sprinklers of a fire alarm system.

Table 2 lists other MicroPnP materials which can be used to construct richer infrastructures. It includes the MicroPnP mote and network manager which were mentioned in Section 1. SIoTIP will need to monitor the availability of motes, however, other materials are managed by MicroPnP and transparent from the SIoTIP and application point-of-view. As mentioned, the network manager is part of the gateway.

Note that MicroPnP motes are constrained by a trade-off between latency, energy, and bandwidth. Motes can be battery-operated, which means that they sleep most of the time to save energy. The lower the latency of the motes, the higher the energy consumption. Lowering the latency is also tied to a lowered bandwidth. Furthermore, bandwidth in the mesh network is currently limited to roughly 26 data packets per second. This makes it currently impossible to support features such as live video streams using the MicroPnP technology.

Table 1: Summary of currently available MicroPnP sensors and actuators.

Name	Description and important details	Configuration
Passive Infrared (Presence) Sensor	Allows detection of motion, and hence presence of people. This is done by regularly taking a low-resolution infrared picture and comparing it to the previous picture taken. The sensor sends a "1" upon detecting motion.	None
Accelerometer	Measures acceleration in 3 dimensions as a tuple of floating point numbers: "(*x-axis,y-axis,z-axis*)". Each number represents the acceleration component in that dimension, is measured in m/s^2. All-weather proof.	Measurement interval (default 0.5s)
Air Temperature and Humidity Sensor	Measures air temperature and air humidity. The temperature is expressed in °C (range -35 up to 95°C with a precision of 0.01°C). Air humidity is expressed as relative humidity in % (precision 0.1, a comfort level is usually between 45 and 55%).	Measurement interval (default 10s)
Microphone Sensor	Allows detection of sound, and hence presence of people. Returns a "0"" or "1"" depending on whether or not the amount of detected noise/sound is above a given threshold (in dB). Detection range is between 20 dB and 120 dB with a precision of 0.1 dB.	Measurement interval (default 5s), noise threshold (default 40 dB).
Air Pressure Sensor	Measures air pressure in bar. Range is between 0.10 and 1.15 bar. The precision is 0.00001 bar. (A normal level is around 1.01325 bar)	Measurement interval (default 10s)
Light Sensor	Measures the amount of light in LUX. Range 20 up to 20000 LUX. The precision is 1 LUX (A normally lit room is usually between 300 and 600 LUX).	Measurement interval. (default 10s)
Distance Sensor	Measures a distance in cm. The sensor detects distances between 2 and 350 cm with a precision of 0.1cm. The measurement is done by sending an infrared light beam and measuring the time it takes for the light beam to be reflected back to the sensor.	Measurement interval (default 5s)
O_2 Gas Sensor	Measures the amount of O_2 gas in the air, in % with a precision of 0.01. All-weather proof. Requires external power supply.	Measurement interval (default 5s)
Buzzer Actuator	Sounds a buzzer. Put a "0" or "1" for turning off or on, respectively.	None
Power Socket Actuator	Turns a power socket on/off (by putting "0"/"1"), hence controlling the power supply of some other device.	None

3.3 Sample applications

This section describes some applications for building control. It is outside the scope of this document to provide a detailed, algorithmic description of each application. Rather, they are meant as illustrative examples of how the SIoTIP platform can be leveraged.

Fire control application

The fire control application uses the sensors to detect fire and take suitable action, e.g. sounding buzzers as warning signal and issuing notifications.

Fire can be detected by (a combination of) air temperature and humidity sensors, surface temperature

Table 2: Summary of MicroPnP support material.

Name	Description and important details
SmartMesh IP Indoor Board (the *MicroPnP mote*)	A board with peripherals for connecting up to 3 sensors/actuators.
SmartMesh IP Network Manager	Network manager allowing connectivity with up to 100 motes. The interconnection between the gateway and the motes.
SmartMesh IP Range Extender	Range extender for indoor or outdoor use.
Weather-proof armour	Durable, weather-proof armour for protecting motes.

sensors, O_2 gas sensors and heat flux sensors. For the application to be activated, each room should be equipped with at least two of these sensors. Furthermore, fire alarm buttons can also be integrated using, for example, power socket actuators. In case a fire is detected, a first task of the system is to alert anyone present, for example by activating the available buzzers or activating alarm bells using power socket actuators. Based on the topology, doors nearby the affected area can be unlocked to simplify evacuation and any available sprinklers can be activated. Note that when unlocking doors, the fire control application possibly overrides any active access control system. Furthermore, the local fire department or emergency response team can be automatically notified.

Building access control application

This application allows to control access to rooms and areas of a building by controlling door locks, for example if some areas are only accessible for authorised personnel. Thus access can be granted or refused after someone authenticates him- or herself using, for example, a badge reader or by entering a code on a key pad. Therefore, such a (custom) sensor must be installed near each controlled door and each such door has to be equipped with a mechanism to lock and unlock it, e.g. an electromagnet that can be turned on and off via a power socket actuator.

Furthermore, different end-users can be involved. For example, a security guard must check whether each door on his or her route is indeed locked and intermediately sends status reports via a mobile app. Furthermore, it is possible to differentiate the actions performed by the application based on who authenticates and/or where they authenticate. For example, a notification can be sent to the security guard when an unauthorised person tries to enter a specific room. Depending on the authentication method used and data stored, this application can also provide a log of who entered specific areas and when.

Burglary detection application

The burglary detection application detects break-in attempts in the monitored area. End-users could arm and disarm the system via a mobile app or via an attached IoT device such as a console near an entrance and exit.

Burglary attempts can be detected by monitoring presence and by sensors installed on doors and windows. Thus at least one sensor that can detect presence of a person should be installed in each monitored room. Furthermore, all windows and doors providing access to a monitored area should be equipped with accelerometers to detect when they are opened.

Upon detecting a break-in, a notification is sent to the relevant parties, e.g. the security guard(s) on duty or the local police department. Furthermore, the application can activate actuators to sound an alarm or activate cameras.

Smart heating application

The heating application allows customer organisations to control the heating (and cooling) system in (a part of) the building in a (semi-)automated way. Customer organisations can specify general policies about, for example, the minimum and maximum temperature. The heating application will control the heating system based on received temperature sensor readings, detected motion of people, weather

predictions and the policy of the customer organisation for that area of the building. Therefore, sufficient sensors should be available in each room to at least sense the temperature and presence of people.

Scenarios such as the following are supported:

- The temperature within a room can be kept within an interval defined by the customer organisation.
- In empty rooms, a minimum temperature is maintained at all times, but the temperature is increased when motion is detected in order to accommodate the comfort of persons in the room.
- The application could include learning behaviour, learning about the daily routine of employees in an office. This allows the system to pre-heat the room prior to the employees their arrival.
- End-users can manually adjust, within the bounds of the enforced policy, the settings via, for example, a mobile app or a web portal.

Light management application

The light management application provides control of the lighting conditions in areas of the building. This is based on the time of the day, daylight levels, presence of people and the preferences of the customer organisation for that area of the building. Furthermore, based on the available topology, lights can be switched on before the room itself is reached, e.g. when detecting motion in an adjacent hallway.

Scenarios such as the following are supported:

- If the light sensor detects that it is getting dark outside and people are present, the lights are switched on.
- The customer organisation can modify the daylight illumination level sensitivity via their online dashboard.
- The customer organisation can indicate that certain light bulbs (e.g. toilet lights, corridor lights) should only be switched on when motion is detected.

All lights managed by this application have to be equipped with a power socket actuator. Furthermore, each relevant room should be equipped with sensors for motion detection or there should be such sensors nearby, e.g. in the adjacent hallway.

3.4 Services offered to stakeholders

This section describes the services offered by SIoTIP to the main stakeholders, which is done through interfaces of the Online Service.

Customer organisation dashboard

Customer organisations interact with the system primarily via the SIoTIP Online Service's customer organisation dashboard accessible via a web portal. This dashboard allows customer organisations to (un)subscribe to and configure applications (and other tasks such as consulting their invoices). Furthermore, customer organisations can manage the roles of the end-users of their application. For example, they can assign certain employees the role of evacuation manager or allow them to control the smart heating via a mobile app.

Infrastructure owner dashboard

Infrastructure owners can use this dashboard to monitor and configure their installed hardware and customer organisations. For instance, they can assign (parts of) the installed gateways, sensors and actuators to customer organisations, add new customer organisations or remove an existing one. It also provides access to a hardware store, allowing infrastructure owners to order gateways, motes, sensors, and actuators. Furthermore, infrastructure owners can configure their topology, essential for many applications, via this dashboard. Finally, the infrastructure owner can consult any received notifications and alarms, for example when a sensor or actuator has failed.

System administrator dashboard

As SIoTIP system administrators must monitor and manage the entire platform, their dashboard should provide them with the necessary tools. This includes showing information (e.g. resource usage statistics) about running applications, offering the ability to shut applications down, receiving and viewing notifications about SIoTIP system components as well as external component failures.

Application provider dashboard

Finally, application providers have access to their own dashboard. It enables them to get an overview of their applications and provides the ability to upload new and update existing applications. As the uploaded applications are tested online/automatically, the dashboard also presents to them the status of this application check and points out potential issues that may impede deployment.

3.5 Service Level Agreements and legal constraints

The SIoTIP organisation offers a Service Level Agreement (SLA) enclosed as a part of the provided support towards our customer organisations. This agreement stipulates certain quality requirements (in terms of performance, availability, security and privacy).

A number of contracts will have to be negotiated with other stakeholders. Some of these will be with application providers, stating some constraints their applications have to conform to. In return we provide certain availability and reliability guarantees to application providers. This includes, among others, that the SIoTIP Online Service must be almost permanently available and that sensor data is never lost or inconsistent. Similarly, a contract with the telecom operator should ensure the desired degree of internet connectivity.

Some legal constraints should also be taken into account. The system will be used within corporations and will collect, process and store business data. Hence, the way in which SIoTIP deals with this data must be in line with security and privacy regulations.

4 Example scenarios

This section presents a set of concrete scenarios concerning the typical usage of SIoTIP.

4.1 Application providers

The following scenarios describe typical interactions between an application provider and SIoTIP. These scenarios feature ACMEapps, a company successfully developing home automation software that wants to enter the industrial building management automation market.

4.1.1 Enrol as application provider

A representative of ACMEapps, Bob, contacts the SIoTIP corporation in order to initiate contract negotiation. This negotiation covers, among other, the minimum service level ACMEapps expects from SIoTIP and payment details. Furthermore, ACMEapps receives documentation on the API available for their applications and rules of conduct (e.g. not purposely overloading the platform). After the negotiation is concluded, application provider dashboard accounts for the individual developers employed by ACMEapps are set up.

4.1.2 Develop application

ACMEapps employee Alice is tasked to manage the development team that will migrate the existing ACMEapps smart heating application for SIoTIP. To achieve this, the application is redesigned and split into three distinct parts: (i) a behaviour learning part to be executed on the Online Service, (ii) a part to be run on a gateway and (iii) a set of front ends such as a mobile app. The developers use the SIoTIP development environment providing easy access to all necessary SIoTIP APIs and the possibility to emulate a realistic deployment environment to debug the application.

4.1.3 Add new application

After a number of iterations, the redesigned ACMEapps smart heating application is ready for release to the SIoTIP's customer organisations. Therefore, Alice logs in into the application provider dashboard and uploads the new application. Once uploaded, SIoTIP initiates a number of automated tests, e.g. to check for possible memory leaks. Alice can follow the progress and results of these tests via her application provider dashboard. The ACMEapps smart heating application successfully passes all tests and is automatically made available to customer organisations for subscription. A notification is sent to Alice informing her of this event.

4.1.4 Update existing application

Over time, bugs are discovered in the ACMEapps smart heating application. Alice's development team fixes these bugs and prepares a new version of the application to be released. Similarly as for new applications, Alice uploads the updated application to SIoTIP via her dashboard. This time the automated tests cannot be completed successfully due to the discovery of a potential memory leak. The current tests are halted and Alice as well as the SIoTIP system administrators are notified of the encountered problem. The SIoTIP system administrator on duty reviews the error report and concludes that the updated application has to be rejected until this issue is resolved. After resolving the issue, Alice uploads a new version, which does pass the tests and the new version of the application is made available. Since ACMEapps provides their subscribers with updates free of charge, the current instances of the smart heating application must be updated to the new version.

4.2 Customer organisations and infrastructure owners

The following scenarios describe typical interactions between SIoTIP and customer organisations and infrastructure owners. These scenarios feature OfficeSpace as owner of multiple buildings in which companies can rent office space and one of their customers Northwind.

4.2.1 Register to SIoTIP

Following renovations in one of their buildings, OfficeSpace wants to upgrade the current fire alarms to a more modern and flexible system. After considering the options, they decide the fire detection application available via SIoTIP best suits their needs. A representative of OfficeSpace, Tracy, contacts the SIoTIP corporation to start a contract negotiation. In this meeting, Tracy provides all necessary information, including that their clients are only allowed to deploy applications involving sensors and actuators located in the offices they rent. An infrastructure dashboard owner account for Tracy is also set up, i.e. her user name and password is registered with SIoTIP. Tracy also provides the names of the currently renting companies, and SIoTIP contacts them for registration. In the registration process, each renting company provides billing and contact information. A registration with SIoTIP has to be carried out each time a new company rents space with OfficeSpace.

4.2.2 Purchase SIoTIP hardware

Back at her desk, Tracy logs in into the infrastructure owner dashboard using her user name and password and subscribes OfficeSpace to the fire detection application. Note that this implies that OfficeSpace has also the role of customer organisation in the context of the fire detection application. Since there is no SIoTIP hardware installed in OfficeSpace's building, new hardware must be purchased and installed before the application is activated. To determine the required hardware, Tracy must provide some extra information, such as the number of floors, the number of rooms on each floor and the total surface. Based on this information, SIoTIP adds sufficient gateways, e.g. one per floor, to the shopping basket. Furthermore, sufficient air temperature and humidity sensors, heat flux sensors and buzzer actuators, as well as the required motes to cover each room are added. Tracy finalises the purchase by accepting the basket and specifying the payment information and delivery address. She also contacts the hardware manufacturer to request integration with the existing alarm buttons. The hardware manufacturer satisfies this request by connecting the alarm buttons' output to generic input sensor devices.

4.2.3 Install new hardware

When the purchased hardware is delivered, Tracy tasks her colleague Ronny to install everything. Later that day, Ronny installs a gateway on each floor and configures these gateways to connect to the local WiFi network. Once online, the gateway immediately connects to the Online Service to register itself. Then, Ronny spreads the motes throughout each floor, and plugs in the sensors and actuators. Note, that each mote was paired to a specific gateway up front and can thus immediately send heartbeats and data to its gateway. Finally, Ronny logs in to the infrastructure owner dashboard and provides the necessary topology information, indicating for example which sensors and actuators are in the same room. He also assign access rights to the customer organisation on each floor to use sensors and actuators installed in their offices.

4.2.4 Configure end-users

Before the fire detection application can be activated, Tracy must assign the role of evacuation manager to at least three persons per floor. More specifically, each floor must have one primary evacuation manager and two secondary evacuation managers. Since OfficeSpace already enforced a similar policy before, Tracy assigns these end-user roles to those who filled the similar role before. Therefore, Tracy logs in into her customer organisation dashboard and provides contact details (i.e. name, e-mail address and phone number) of the selected persons. The new evacuation managers who have a smartphone are e-mailed instructions on how to install the fire detection mobile app on their smartphone. Through this mobile app they can indicate, in case of fire, when the evacuation of their floor is complete. Furthermore, they can indicate when they will be absent, allowing the fire detection application to promote a secondary evacuation manager to the role of primary if necessary.

4.2.5 Hardware failure

A few days later, Tracy receives an SMS notification informing her that one of the installed air temperature and humidity sensors is no longer sending data. The fire detection application remains fully operational because it can still rely on the heat flux sensor that was installed in the same room as the failed sensor. Tracy logs in to the infrastructure owner dashboard and orders a replacement sensor. A few days later, this sensor is delivered and Ronny is tasked to replace the broken sensor with the new one.

4.2.6 Subscribe to application

Jon, a manager of Northwind, became curious about the newly-installed fire detection system in his office and did some research on SIoTIP. While exploring his customer organisation dashboard, he notices that there is a smart heating application available as well. Jon figures that this application could alleviate the common complaint of his employees that the offices are too cold in the morning. Jon subscribes to this application through his dashboard and is informed that the application will be activated once the required extra sensors and actuators, such as passive infrared sensors, are installed. Ronny is notified of this subscription via the infrastructure owner dashboard and he approves the purchase. When the new hardware arrives, Ronny tasks one of his employees to install these and update the topology and access rights via the infrastructure owner dashboard accordingly. Once this is done, the smart heating application for Northwind is activated and Jon is notified via e-mail of this event. Upon receiving this e-mail, Jon logs in into his customer organisation dashboard and navigates to his active application subscriptions to configure the smart heating application, e.g. set the minimum and maximum temperature.

4.2.7 Data transmission

When detecting motion, the infrared sensors in Northwind's offices will send a message to the corresponding gateway. The local smart heating application on the gateway will take the necessary actions such as turn up the heating depending on the last-known temperature.

The gateway locally stores the sensor data (along with a timestamp) and the actuation command it issued, if any. This data will be sent to the Online Service as part of the next periodic synchronisation between the gateway and the Online service. The behaviour learning algorithm running on the Online Service as part of the Northwind's subscription can then take this new data into consideration.

5 Instructions

The project involves the development of the SIoTIP platform as described above. It consists of three parts: part 1 (2 lab sessions) involves both analysis of the problem domain and requirements elicitation. Part 2a (2 lab sessions) involves the initial design and inception of a software architecture for SIoTIP. Finally, in part 2b (3 lab sessions) the initial architecture is extended and completed.

This document presents the specific assignment for part 1. In this part of the assignment, you are asked to provide (i) an analysis of the problem domain, (ii) a set of functional requirements (use cases) and (iii) a set of non-functional requirements (quality attribute scenarios).

5.1 Domain analysis

In this part, you eliminate gaps and ambiguities from the problem description by unambiguously describing the problem domain. The results should consist of:

1. **Conceptual model(s) of the problem domain.** This includes the relations or associations between the different domain concepts and their relevant characteristics. Create a UML class diagram[4].
2. Relevant additional **domain constraints** (informal text suffices, but OCL is allowed).
3. A **glossary**, which presents an overview of the different concepts and terms, together with a short description for those concepts that need further clarification.

5.2 Functional requirements

Describe the main functional requirements as use cases, and present the use case diagram(s)[5]. Provide a list of all use cases (title and summary) and write at least **five** of the **most important and relevant** use cases in full detail.

5.3 Non-functional requirements

Describe the main non-functional requirements as quality attribute scenarios[6]. Create **two** scenarios for each of the following scenario types: Availability, Performance, Modifiability and Usability. Focus on the most important and relevant scenarios and document them in a high level of detail.

5.4 Format of the report

Integrate the documents and models that are generated in the analysis phase in a coherent document. Adhere to the following structure:

1. Domain Analysis
   (a) Domain model(s)
   (b) Domain constraints
   (c) Glossary
2. Functional requirements
   (a) Use case diagram
   (b) List of use cases
   (c) Detailed use cases

[4]For more information, please refer to the Larman book "Applying UML and Patterns, 3rd ed.", chapter 9 (Domain Models).

[5]For more information about modeling use case diagrams, refer to the Larman book "Applying UML and Patterns, 3rd ed.", Chapter 6 (Use Cases).

[6]For more information about the format in which quality attribute scenarios are documented, refer to the book of Bass, Clements and Kazman "Software Architecture in Practice, 3rd ed.", Part 2 (Quality Attributes).

3. Non-functional requirements
    - Availability scenarios
    - Performance scenarios
    - Modifiability scenarios
    - Usability scenarios

A LaTeX template, containing the above structure, is made available via Toledo. Hand in the report in the dedicated project post boxes (main floor of the department of computer science, in the student printer room A00.03) and submit the digital version via Toledo. The deadline for this is **Friday March 10 at noon**.

Good luck,
The SA team.