

Andrea Finardi

IoT Simulations with Cisco Packet Tracer



Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

4th June 2018

Preface

Time has passed since I started my studies, many things happened and many new places have been visited. It is time however to finally close this chapter.

Thanks to Ville Jääskeläinen, my thesis lecturer, for giving me the opportunity to work on this thesis. Thanks to Veera for being patient and support me in this long journey. Thanks to my parents for keep asking me the thesis progress updates every fifteen minutes.

At last, but not least, thanks to our dog, Pablo, to be the best distraction when working on this thesis.

Now I have to go, I need to get married tomorrow.

Ne è passato di tempo da quando ho iniziato il mio corso di studi, tante cose sono accadute e tanti posti sono stati visitati. E' venuta l'ora di chiudere questo capitolo.

Un grazie va a Ville Jääskeläinen, il mio relatore, per avermi dato la possibilità di poter lavorare su questa tesi. Grazie a Veera per essere stata paziente ed avermi supportato in questa lunga esperienza.

Grazie ai miei genitori per aver continuato a chiedere aggiornamenti sullo stato della tesi ogni quindici minuti.

Infine, grazie a Pablo, il nostro cane, per essere stato la miglior distrazione mentre lavoravo su questa tesi.

Adesso devo andare, devo sposarmi domani.

Espoo, 1st of June 2018

Andrea Finardi

Author	Andrea Finardi
Title	IoT Simulations with Cisco Packet Tracer
Number of Pages	89 pages + 3 appendices
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Networking and Services
Instructor	Ville Jääskeläinen, Lecturer
<p>This thesis work was aiming to deliver practical IoT simulations using the Cisco Packet Tracer tool to support the Internet of Things course in Metropolia University of Applied Science.</p> <p>The work was conducted as a project where requirements were first gathered, then simulations were developed and finally introduced to the students during two practical classes. Students and IoT lecturer feedbacks are also listed in the conclusions.</p> <p>Four Cisco Packet Tracer IoT simulations were designed. They consisted of pre-configured IoT scenarios, simulating home and industrial applications, where a network layout and IoT devices were already set along with an IoT simulation backend intelligence and an example of microcontroller programming.</p> <p>Full explanations of the simulations are included in the thesis.</p> <p>The thesis work, backed up by the student feedback, was proven to be successful both from contents, methodology and support point of view. Conclusions were, however, highlighting that additional practical classes should be added in future implementations of the IoT course.</p> <p>Future studies should also be conducted in order to explore new IoT simulator tools and possibility to utilize IoT real hardware technologies such as Raspberry Pi and Arduino</p>	
Keywords	IoT, Internet of Things, Cisco Packet Tracer, Simulations

Contents

Preface	
Abstract	
Table of contents	
Abbreviations and Acronyms	
1 Introduction	1
2 Internet of Things (IoT)	5
2.1 History and Evolution of IoT	5
2.2 Definition of IoT	7
2.2.1 Cloud Essential Characteristics	8
2.2.2 Cloud Service Models	9
2.2.3 Cloud Deployment Models	11
2.3 IoT Networking Overview	12
2.3.1 LoRaWAN Overview	13
2.3.2 SigFox Overview	15
2.3.3 Narrowband-IoT Overview	16
2.4. Cisco Packet Tracer Overview	17
3 Methods and Materials	19
4 Cisco Packet Tracer Simulations	25
4.1 IoT Exercises Introduction	25
4.2 Cisco Packet Tracer Technology Introduction	27
4.3 IoT Simulations	37
4.3.1 Smart-Home 1	37
4.3.2 Smart-Home 2 (SaaS)	46
4.3.3 Smart-Campus	57
4.3.4 Smart-Industrial	68
4.4 Limits and Expansions of the IoT Simulations	79
5 Feedback and Recommendations	82
5.1 Students Feedbacks	83
5.2 Feedbacks and Suggestions for Future IoT Courses	84
6 Conclusions	88
References	
Appendices	

Appendix 1. Blockly custom software for IoT simulations

Appendix 2. Network details utilized in the IoT simulations

Appendix 3. Students feedback form

Abbreviations and Acronyms

3GPP	3 rd Generation Partnership Project
ALOHA	Additive Links On-line Hawaii Area
API	Application Programming Interface
APN	Access Point Name
ARPANET	Advanced Research Project Agency Network
AWS	Amazon Web Service
CLI	Command Line Interface
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EC2	Elastic Computer Cloud
GUI	Graphical User Interface
I/O	Input/Output
IaaS	Infrastructure as a Service
IoE	Internet of Everything
IoT	Internet of Things
ISM	Industrial, Scientific and Medical
ISP	Internet Service Provider
LAN	Local Area Network
LCD	Liquid Crystal Display
LPWAN	Low-Powered Wide Area Networking
MCU	Multi-Chip Unit
NetAcad	Cisco Networking Academy
NB-IOT	Narrowband IoT
NIC	Network Interface Card
NIST	National Institute of Standards and Technology
PaaS	Platform as a Service
RFID	Radio Frequency Identification
RIP	Routing Information Protocol
SaaS	Software as a Service
SBC	Single-Board Computer
SNO	SigFox Network Operators
SSID	Service Set Identifier
UNB	Ultra Narrow Band

URL	Uniform Resource Locator
VPN	Virtual Private Network
WLAN	Wireless Local Area Network

1 Introduction

Internet of Things and Internet of Everything are two words that commonly refers to the new trend to have small, cheap and always-connected devices used to send data to a backend cloud based applications. This opens up a new set of possibilities and products that companies are developing and selling in both industrial and consumer markets.

In 2018 Metropolia University of Applied Science started a new study course, called Internet of Things. The study course includes first an overall Introduction of IoT, followed by a development of an IoT business case and finally over a practical IoT simulation.

This thesis work was aiming to build practical cases where students could experience, through an IoT simulator, the various IoT sensor-based components, network landscapes where all the devices are connected and backend intelligence where logic and analysis of sensor-based data can be gathered and analyzed.

The tool chosen for the simulations is Cisco Packet Tracer, this tool has been used for many years to train students on Cisco networking. Main strength of the tool is the offering of a variety of network components that simulate a real network, devices would then need to be interconnected and configured in order to create a network. In the last version of the tool Cisco introduced IoT functionalities, and now it is possible to add to the network smart devices, components, sensors, actuators and also devices that simulate microcontrollers such as Arudino or Raspberry Pi. All the IoT devices can be run on standard programs or can be customized by programming them with Java, Phytion or Blockly. This makes Cisco Packet Tracer an ideal tool for building IoT practical simulations and class exercises.

The scope of this study was to focus on preparing four different pre-defined Cisco Packet Tracer scenarios that would help students to quickly understand the IoT functionalities of the tool. An introduction of the tool, explanation of the IoT functionalities of it and support the students during the group work exercises was also part of the thesis work.

The need of the pre-configured exercise comes to the fact that only two classes were destined for the IoT practical simulations within the study course. These exercises represent a solid foundation for the students to expand the simulations closely to the own business case developed in the previous part of the course of study.

The four simulations environments provide a fully working network utilizing various Cisco components such as: router, wireless router, switch, internet connectivity cloud and backend IoT servers. Additionally, in all four simulations, there are examples of IoT smart devices already connected to the local network. Also backend logic is provided and programming of these sensors have been created in order to give examples to the students of how setup further and more complicated cases.

For more advanced users and, in order to build more realistic cases, Cisco Packet Tracer offers also the possibility to a more low-level IoT simulation using microcontroller, sensors and actuators. These scenarios are not utilizing smart devices always connected to an IoT network, but they replicate cases where Arduino or Raspberry Pi microcontrollers are used, including cabling and creation of custom made programs.

In each of the four simulations there is one example of sensor-to-actuator cases using basic Blockly programming of the microcontroller devices.

The methodology used in the thesis has been the similar utilized in a business typical project: demand, development, delivery, feedback and closure.

The starting point of the thesis work was to interview and gather requirements from the course lecturer on the needs and contents for the IoT course. Even if need to have practical exercises was clear, the tool, methodology and simulation structure was open at this stages, especially as the Internet of Things course was never been part of the degree program before. The other limitations that were kept in mind in the planning phase was to be able to structure the exercises in order to meet different skillset within the students group to balance networking and programming knowledge. The other constraint that emerged during the interviews was that practical slots were limited to two session in computer class. Needs to have pre-packaged simulations was clear.

Once demand part of the project had been clarified the next part was the development of the exercises with the Cisco Packet Tracer tool.

The Cisco Packet Tracer learning material was not fully accessible or even available, especially for the IoT section. In order to gather initial knowledge of the tool, and develop them by building the simulations, part of the thesis was to follow three online Cisco NetAcad course: Intro to IoT, Packet Tracer 1o1 (2016) and Packet Tracer 1o1 (2017). These three courses helped to get a solid overview of the tool and the IoT capabilities of it.

The next core activity of the thesis development was to prepare and document the simulations, building them started with the creation of specifications and setup of basic networks and then adding IoT smart devices, creation of backend intelligence and then the addition of a small microcontroller examples.

The four IoT cases are simulating Smart-Homes, in two variants, Smart-Campus and Smart-Industrial. Network layers were built using a combination between router, wireless router, switches, backbone connection, 3G antennas and internet connection clouds.

Smart-Home cases simulate a domotic experience where IoT smart devices are connected to a local network in order to give automation within the house. Examples of home automations include climate control, alarm and security events, electricity storing and intelligent lights.

Smart-Campus simulates a university campus with different network zones, where electricity is produced and utilized by smart devices and, security sensors. Smart building access control is also in place.

Smart-Industrial is a simulation of a power plant that produces and stores electricity via solar panels and wind turbines. All the electricity is produced by smart devices, then stored and utilized to power a production chain filled with smart sensor and actuators. IoT security features are also introduced in the simulations.

The other fundamental part of the thesis work was to deliver the exercise and introduce the simulations to the students of the Metropolia Internet of Things course. Two sessions were organized in order to first give to the student a brief introduction of the tool and its capabilities, in addition to that a small networking exercise was also given to students in order to experience the setup of a basic interconnected network using basic components

such: router, switch and simulated PC. In the first practical class also an introduction of Cisco Packet Tracer IoT components was given.

The second practical session was for the students to purely practice with the IoT component offered by the Cisco Packet Tracer. The groups used the four pre-defined exercise as foundation to build a simulation close to the own IoT business case developed in the early stages of the IoT study course. During the practical session support, knowledge sharing and tips were given to the groups in order to create the own network and IoT simulation. For students where the own business case was not practically achievable using Cisco Packet Tracer it was asked to modify one of the four pre-build simulations.

Last part of the project was to gather feedback from the students at the end of the two practical session. Feedbacks and suggestions were both regarding the four simulation cases but also on the eligibility of the Cisco Packet Tracer tool itself. These inputs have been used to integrate the conclusion section of the thesis work along with experiences gathered while building the examples.

Conclusions are also commenting the possible future study course structure and also how the future simulation should be linked deeper to the students business case, possibly including a real practice with microcontrollers.

The thesis report is written in four main sections, chapter two give an introduction of IoT and the Cisco Packet Tracer tool, chapter three describes the procedure and steps of the project, chapter four give the technical explanation of the four simulation cases and chapters five and six gather the student's feedback and conclusion of the study.

2 Internet of Things (IoT)

This chapter briefly introduces the concept of Internet of Things (IoT) illustrating the basic concepts of cloud, its definition, the various type of implementations and the network aspect of IoT.

Second part of the chapter also briefly introduces the Cisco Packet Tracer tool.

2.1 History and evolution of IoT

According to Gartner studies [1], amount of connected IoT connected devices, excluding computers, smartphones and tablets, will reach more than 20 billion, largely overpassing the human world population.

The origin of “cloud” term is not clear, the early concept of cloud and shares services dates back in the sixties [2]. The first concept was referring to a vague and yet distant future in which the computing would occur in few and remote locations without much human intervention and where the services would be equally distributed among the public users.

One example of early concepts can be traced back on 1961, when computer scientist John McCharty proposed the first concept of “computer public utility”:

“If computers of the kind I have advocated become the computer of the futures, then computing may someday be organized as a public utility just as the telephone system is a public utility. ... the computer utility could become the basis of a new and important industry” [2].

Another important statement was published by Leonard Kleinrock in 1969 (Chief scientist of the Advanced Research Project Agency Network or ARPANET) strengthening the concept of public utility:

“As of now, computer networks are still in the infancy, but as they grew up and they become sophisticated, we will probably see the spread of “computer utilities” [3].

In the same year J.C.R. Licklider (Responsible for enabling development in ARPANET) also introduced the idea of “intergalactic computer network” visioning for everyone in the globe to be interconnected and accessing programs and data from anywhere.” [4]

The term “cloud” itself, over the modern years, has been also commonly used to describe and refer to different technologies. For example in the early 1990s the term was, and still is, used by the network industry to refer to an abstract layer to deliver data in heterogeneous public and semi-public networks.

The term “cloud” in computing was also used to describe platforms for distributed computing (Wired’s magazine April 1994). [2]

A big milestone in the cloud computing history was reached in the 1999 when Salesforce.com pioneered the concept of enterprise remote provisioned software via website. [2][3]

The next milestone was in 2002 when Amazon.com released the Amazon Web Service (AWS), which provided a suite of enterprise-oriented services that included, remote provisioned computing processing power, storage and other business functionalities. [2][3]

In 2006 another big step on the computing cloud history was marked. Amazon released its Elastic Compute Cloud (EC2) services, enabling organizations and private to “lease” computing power in order to run their own applications. Few years later, in 2009, the Google App Engine was also released. [2][3]

These two services forged the modern cloud computing concept.

Cloud computing success has been also enabled by several key factor such as maturity of virtualization technology, wide-spread of low latency high-speed networks, cost reduction of power processing and storage space.

Exploring these concepts is out of scope for this thesis work.

2.2 Definition of IoT

The cloud computing definition that received the industry-wide approval was published by the US National Institute of Standards and Technology (NIST) back in 2009, reviewed version was then published in September 2011:

“Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, server, storage, application and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models and four deployment models.” [5]

In other words this defines that access and provisioning of computing resources should be easy and possible from everywhere. Resources should be scalable, organized in pools, and based on requirements, they can be allocated by a minimum management efforts.

Essential IoT characteristics, by definitions, are: on-demand self-service, broad network access, resource pooling, rapid elasticity and measurable services.

Three service models are: IaaS (Infrastructure as a service), PaaS (Platform as a service) and SaaS (Software as a service).

Four deployment models are: public clouds, community clouds, private clouds and hybrid clouds.

2.2.1 Cloud Essential Characteristics

According to definition [5] on-demand characteristic gives the freedom to the cloud user to self-provisioning the IT cloud resources without a human intervention. Provisioning is mostly made by self-service portals where the user can chose computing power, storage capacity, network connection, eventual software etc. This characteristic enables the main concept of “service-based” and “usage-driven content” of a cloud environment.

Broad access defines that cloud services must be largely accessible by the users using heterogeneous access devices [6]. Users in fact need to be able to connect to the service using different types of terminal (PC, tablet, mobile phones), transport protocols and security technologies. Due to the broad access, the service might be tailored fit to suit the requirements, additional Application Programming Interface (API) will be required.

Resource pooling, and multitenancy, refers to the dynamical allocation of IT resources in order to meet the customer demand. The resource allocation should be totally transparent to the end user and not related to the location where the cloud service is hosted. Resource pooling mostly uses virtualization technologies and allows the cloud provider to serve multiple could customers using the same infrastructure, this is called multitenancy. Different tenants are not aware of each other’s presence, due to resource isolation, and might dynamically reserve and release IT resources.

Rapid elasticity is the ability of the cloud service to automatically, and transparently, allocate IT resources in order to satisfy the cloud users need. Users have the illusion of infinite resources. Scaling is usually done using probes and scaling agents that can detect the needs and immediately allocate more IT resources such as network, memory, storage, processing power, VM. This characteristic is a core reason of the cloud service existence itself.

As per the NIST definition measurability is a key cloud element defining the characteristics that all the cloud services need to have measurable features for billing, monitoring and reporting purpose. This is a fundamental requirement for both not-charged usage and for more common pay-per-use cloud services.

Resilience is a characteristic that originally was not included in the NIST cloud definition, however over the years this aspect gained a significant importance in the cloud solutions justifying the usage of the cloud itself against the on-premises systems.

In cloud computing the resiliency refers to the capability of the cloud service to failover and distributes the service over redundant pool of IT resources across physical locations or within the same cloud. Usually the failover mechanism is fully automatic and relies on probes that detect the failures and react according to a pre-defined set of instructions [7].

2.2.2 Cloud Service Models

Cloud service models, also called cloud delivery models, are a set of pre-packaged combination of IT resources offered by the cloud providers. Those models are specialized following the needs of the users and grant a certain degrees of configuration freedom. Three models included in the NIST cloud definitions are: IaaS, PaaS and SaaS.

Infrastructure as a Service or IaaS is a cloud model where the provider offers to the users a self-contained IT environment that user can maintain and administer via administration tools accessed by a cloud service portal. This IT environment usually refers to hardware, processing capacity, storage, networks, virtualized servers, Operating systems etc. In contrast to other service models, the responsibility to administer the cloud service is on the cloud consumers. Provider might offer bundle of pre-set virtual server in order to ease the cloud consumer administration activities. Cloud providers could also offer IaaS to other cloud providers that will then create own services on this cloud infrastructure. The benefits of this delivery model is that a customer has full control of the infrastructure itself; drawback is that customer would need to have internal IT resources to administer the cloud infrastructure.

Examples of IaaS are: Amazon EC2, Windows Azure, Rackspace and Google Compute Engine.

Platform as a service, or PaaS, usually refers to a “ready to use” platform where cloud customers can start developing their own applications. In this delivery model all the IT resources must be fully deployed, configured and “ready to be use”. Platform comes also with a comprehensive suite of application development toolkit (i.e. Google App Engine

offers some Java and Python based environments) to follow the entire life-cycle of application development.

This model usually ease the cloud customer from IT administration tasks as the underlying infrastructure is not manageable, however cloud consumer has the control over the application deployment and the configuration settings of the IT resources for the application hosting.

Examples of PaaS are: AWS Elastic Beanstalk, IBM Watson IoT, Windows Azure, Heroku, Force.com, Google App Engine and Apache Stratos.

SaaS, or Software as a Service model, usually refers to a fully-available and pre-packaged environment that cloud customers can use over cloud services. This solution allows the customers to access to a service that is really easy and quick to setup, allowing also the cloud provider to re-use the same cloud product for several customers. Cloud users, in this model, do not have any administrative access and control over the IT resources, only minimal settings changes on the software itself can be done.

Multitenancy technologies are used to distribute load on several resources, making the SaaS a reliable and distributed service. SaaS can be both a “pay-per-use” or a “free-of-charge” service for the users. In the second models the provider would get revenues from commercial advertisements or re-selling statistical information of the service users.

Examples of SaaS are Google Apps, Microsoft Office 365 and many other commercial webmail platform.

Over the recent year a multitude of more specialized service models was released mostly focusing on a specific services. Examples are: Storage as a Service, Database as a Service, Security as a Service, Process as a Service, Testing as a Service, Integration as a Service etc. Additionally also combination of cloud delivery models can be offered to customers, for example IaaS plus PaaS can give the cloud user a software development kit also granting a major degree of administering resources compared to only a PaaS scenario.

2.2.3 Cloud Deployment Models

Deployment models are foundation of the NIST definition and describes ownership, size and who can access the cloud infrastructure. Four models are included in the definition: private cloud, public cloud, community cloud and hybrid cloud.

Private cloud is an infrastructure owned and at disposal of one single organization. Cloud itself can be hosted either on-premises or hosted on third party facility. Private companies might use private clouds to centralize IT environments or to extend the on-premises service to cloud solutions running on third party companies. In this scenario the cloud consumer is also cloud provider, in-house IT department can assume a specific role of providers. Additionally, as long as IT resources are remotely accessible, they can be consider as cloud resources. Private clouds have a significant physical footprint and usually require capital investments.

Public cloud is a widely accessible cloud environment owned and hosted by a third party company, that assumes the role of cloud provider. This is usually at disposal of general public and is most likely free of charge.

Security concerns are raised when utilizing public clouds as data are hosted “outside the premises” and, as service is provided for a broad audience, is most like accessed over non-trusted networks.

Community cloud deployment model is similar to the public cloud concept, however access is limited to a specific set of organization that have common needs. Underneath infrastructure can be either owner by a third party company or co-owned by the member of the community. Usually access for parties outside the community is denied.

Hybrid cloud is the last deployment model on the NIST definition and usually refers to a combination of other deployment models. Due to criticality of the data some organization, for example, could outsource some services over public cloud and maintain others within the own private cloud. Hybrid clouds present the unique challenge to avoid disparity in the cloud environments if managed by different providers.

2.3 IoT Networking Overview

As discussed in the previous chapters the NIST definition of cloud is quite accurately listing characteristics, service models and deployment models, however it does not refer to networks. Networks in IoT are not in fact a characteristic but they are enablers.

One key-contributor factor for the success widespread of IoT technology is in fact due to the raise of modern, fast, reliable, low-latency and low-cost networks.

Specifically for IoT the most common network types range between Bluetooth, traditional Wireless Local Area Network (WLAN), cellular and a new generation of Lower-Power Wide Area Network (LPWAN).

At the moment in the IoT industry there are no standards for networking [8], however few technologies have a clear advantage compared to others.

WLAN and Bluetooth technology are without any doubt the most common type of consumer network in the market at the moment. They both work in a license-free radio frequency band, they both ensure a good bandwidth transfer rate and they both require fairly inexpensive receivers. Limitation comes however from the fact that they have evident range limitation that precludes them to be the main choice for being used in extensive IoT applications. Range in fact is limited to few tens of meters in WLAN and few meters for Bluetooth connection.

As IoT industrial applications are intended to work mostly with devices distributed in a wide area, often with bad cellular coverage, and that would require a strict power management to extend the battery lifetime, a new technology of Lower-Power Wide Area Networking (LPWAN) is raising in IoT. [9]

LPWAN are networks that combine technologies in order to achieve long-distance, robust and low-bitrate communications with battery operated sensors geographically located in a wide area.

The three most important LPWAN technologies are LoRaWAN, SigFox, and Narrowband IoT. These three technologies are briefly covered in the next chapters.

2.3.1 LoRaWAN Overview

LoRaWAN is a technology created by the non-profit, multi-vendor LoRa Alliance™ [10], and relies in a star-of-star infrastructure where many devices are connected with a single hop to a receiver gateway, that then relays the messages via traditional IP networks to the central backend servers.

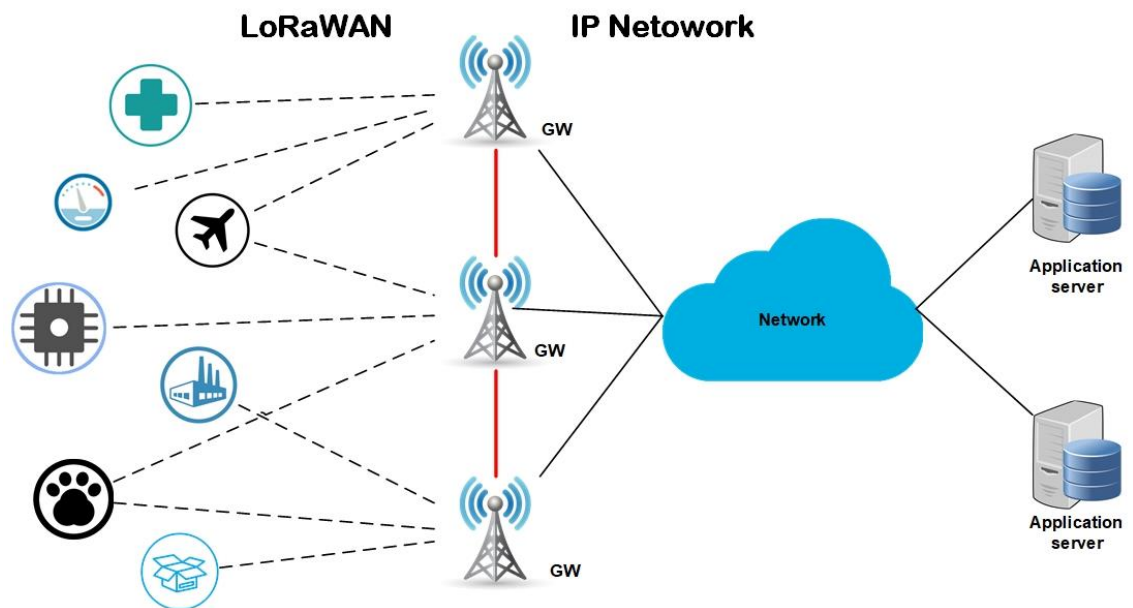


Figure 1 - LoRaWAN Infrastructure

LoRaWAN offers bi-directional connectivity over multi-kilometres range and with bitrate between 0.3 kbps to 50 kbps. Even if communication is bi-directional the uplink traffic, from device to network, is preferred. [11]

LoRaWAN allows three classes (A, B and C) of transmitting devices based on throughput and power needs of the IoT devices [12] [13] [14].

A class devices are low-power devices that require bi-directional communication and rely on an ALOHA (Additive Links On-line Hawaii Area) type of protocol. Uplink and downlink communications are in fact asynchronous and communication is triggered by the IoT device sending a frame to the uplink channel and then listening for an answer, for few seconds, in the next two downlink windows. Once uplink communication is acknowledged by the gateway the downlink traffic can start. As the communication is triggered

by the device and does not require any other periodical transmission, this allows the device to be in a constant sleep mode, saving batteries, and communicating only when necessary.

B class devices are designed for applications where additional downlink is required. A periodical beacon message is in fact sent by the gateway to the IoT device to schedule additional downlink windows without the need to previous successful uplink communications. Device battery lifetime is impacted due to the receiving of the additional synchronization messages.

Class C communications ensure a low-latency communication as, as opposite of class A devices, class C devices are always listening for downlink traffic. Gateway is able to know the status of the device and start the sending data to the device at any time. Battery optimization in such cases is achieved by switching the device from class A to class C when necessary.

Apart from the downlink limitation for the lower class devices the LoRaWAN biggest performance issue is due to the duty-cycle limitation imposed in the ISM bands regulation. [15] Typically the regulations, for instance in Europe [16], dictates that only 1% of the time the device is allowed to transmit in each sub-band. This limits the traffic for each device but also add extra complexity to the network when big amount of device are connected to the same network.

2.3.2 SigFox Overview

SigFox is a French company, founded in Toulouse in 2009, that created their own IoT LPWAN proprietary solution on cellular-style networks.

SigFox architecture is based on single-hop connection between device and gateway where network coverage is achieved by utilizing SigFox Network Operator (SNO) telecommunication infrastructure. Receiving gateways are often hosted in the SNO cellular towers. [14] [17]

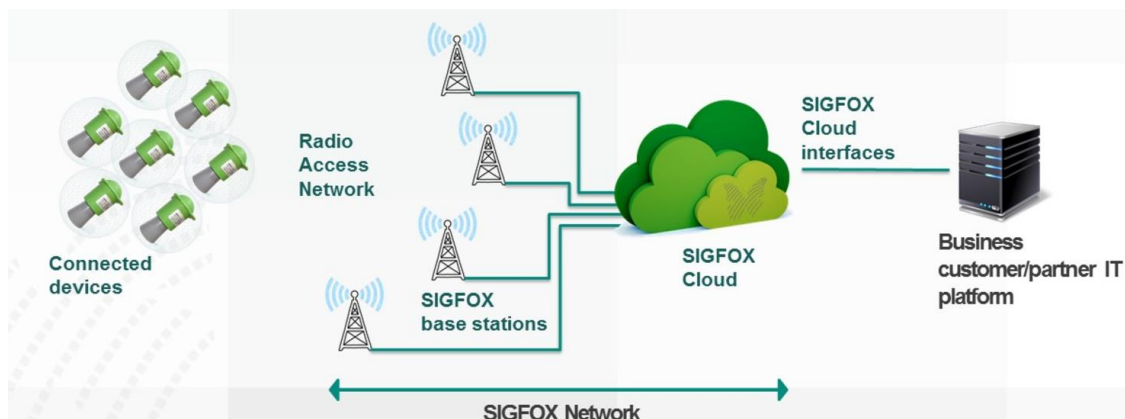


Figure 2 - SigFox infrastructure [18]

SigFox operates at 869 MHz (Europe) and 915 MHz (North America) Industrial, Scientific and Medical (ISM) radio bands and it utilizes Ultra Narrow Band (UNB) 100 Hz bandwidth [19] in order to pass the signal through solid object. This allows the signal to be propagated for many kilometres, also underground, allowing also power saving to the transmitting devices.

There are no standard for LPWAN device power consumption however, 10 years while utilizing two AA batteries, is the referred battery lifetime that IoT LPWAN devices should achieve. [20]

Devices connected to SigFox are able to achieve maximum sending of 140 packets per day with a payload of 12 Bytes per package with a maximum wireless throughput of 100 bps in the uplink channel [14]. Coverage of the network is directly based on the environment and can go from 1km to several tens of km in rural or open space areas.

The limitation of SigFox are coming from the throughput data restriction and the fact that solution is a SigFox proprietary product, and not an open product like LoRaWAN.

2.3.3. Narrowband-IoT Overview

Narrowband IoT (NB-IoT) it is a LPWAN solution developed by the 3rd Generation Partnership Project (3GPP) that is aiming to provide a bi-directional, cost-effective, simple to use IoT network for connecting large amount of device while optimizing their power consumption.

NB-IoT utilizes cellular networks frequencies and also aims to reutilize the building blocks of the LTE connectivity layer while simplify the requirements for device hardware complexity [21].

The NB-IoT specifications are currently frozen to release 13 [22].

The release 13 specifications have been agreed to provide an uplink and downlink bandwidth of 250 kbps while utilizing a 180 KHz ultra narrowband signal. With an acceptable latency between 1.6 and 10 seconds [23].

Advantages of Release 13 is to lower down the device transmission power to 20 dBm, allowing a battery lifetime of more than 10 years with two standard AA batteries.

As the solution is coming from an extensive partnership of major telecom providers, one advantage of this technology is built by making sure that NB-IoT can co-exist with existing GSM, UMTS and LTE networks.

2.4 Cisco Packet Tracer Overview

Cisco Packet Tracer is a Cisco proprietary multi-platform tool that enables possibility for students to create networking and IoT simulations without need of a hardware or pre-existing network.

The tool is free of charge, runs on the major operating systems and it is downloadable from Cisco NetAcad page for all students and teachers having a valid NetAcad account.

The tool has been available through the years for all the students participating in Cisco courses and, originally, was designed to support practical exercises for students attending the Certified Cisco Network Associated (CCNA) Academy courses.

At the time when this thesis work was written the latest release available was the 7.1.1.

According to 2017 Corporate Social Responsibility report [24] Cisco Networking Academy, also referred as NetAcad, has educated so far over the years more than 7.8 million people in more than 170 countries. It forms more than twenty two thousand educators worldwide and can count on more than ten thousand partner institutions. Many of these students have used Cisco Packet Tracer in their Cisco education.

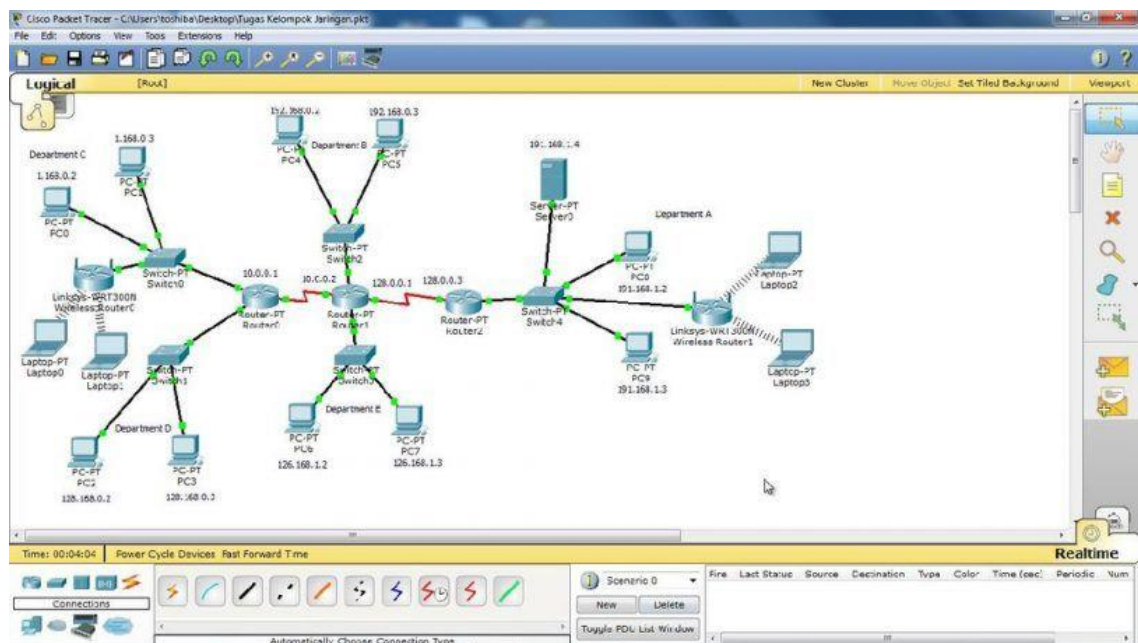


Figure 3 - Cisco Packet Trace user interface

The tool was built in order to allow students to experiment with networking without having a need for costly network infrastructure and lengthy hardware setup procedures. The tool in fact offers an extensive set of hardware and cabling that allows students to setup from a basic to very a complex network, enabling them to learn how to program Cisco appliances via Command Line (CLI) interface. It also educates how to troubleshoot network related problems, as the tool also includes realistic features for debugging.

From version 7.0 Cisco also introduced IoT functionalities in the tool, allowing students to practice by setting up IoT devices and IoT automations. Also a possibility for a lower level IoT simulation using single board computer (SBC) and sensor was offered in the same release.

More complete explanation of the Cisco Packet Tracer feature can be found in the Chapter 4.2.

This thesis work was only focusing on delivering IoT simulations utilizing Cisco Packet Tracer. It was not in the scope of this thesis to evaluate or compare other IoT simulator available.

It is also expected that Cisco will release more IoT functionalities in the future due to the growing amount of IoT courses offered in NetAcad. Functionalities, devices and tools might differ from what is explained in the chapter 4.2.

3 Methods and Materials

The purpose of this chapter is to describe the method on how this thesis work was conducted, explaining the process, the methodologies and the practical steps achieved.

The second part of the chapter is focusing on how the IoT simulations were built and how they were explained to the students. Technical explanations of the exercises are included in the chapter 4.

As earlier mentioned the original necessity for this thesis works came from the need to build a practical section for the Internet of Things course taught in Helsinki Metropolia University of Applied Science starting from January 2018.

The course lecturer had in fact already the structure for the theoretical classes, however practical sessions were also required in order to give the students a possibility to familiarize with the IoT components.

Due to the extra complexity in having real hardware such as microcontrollers, sensors and actuators, it was decided to utilize an IoT simulator. A choice was made to use Cisco Packet Tracer simulation tool.

Once the needs and the tool were clarified and agreed, the next step was to decide how to structure the practical classes that, due to time limitation in the study course, were agreed to be diluted in two sessions.

From this step onwards the thesis work was conducted following typical IT project methodologies.

The work was in fact divided in five main sub categories and periodical check-up sessions were organized in order to steer the contents of the deliverables.

As one can see in the below Figure 4 below the four main categories were: requirement gathering, analysis of the tool, development of the simulation environment, roll-out of the simulation during the classes and last the feedback collection.

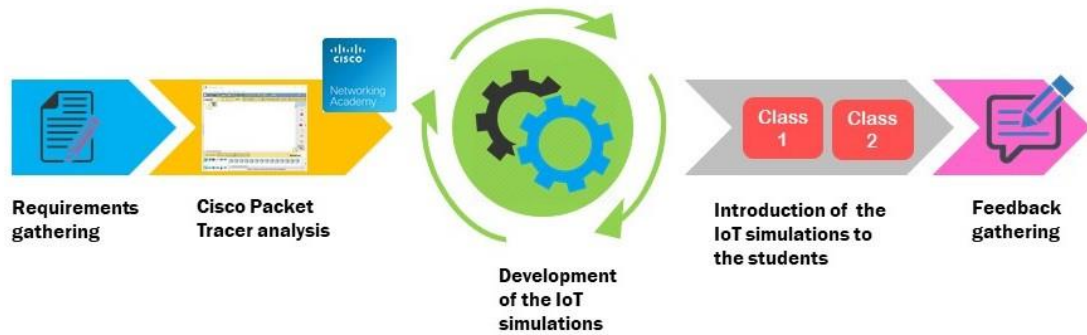


Figure 4 - Thesis work process

Gathering of the requirements was done by having few meetings with the IoT course lecturer and details shared were quite simple. The need was to build pre-defined simulations to be used in the practical class sessions. Exercises needed to be simple enough for students to understand the basic concepts of the simulations, yet challenging them for future developments.

The cases also required to have most of the basic configurations done, so students could effectively concentrate on the IoT aspect and not spend too much time on the networking side. Simulations also needed to be flexible enough in order to be expanded by students in the future implementations of the course.

Contents and themes of the simulations were left open, however ideas were suggested to represent both home and industrial IoT applications.

Due to the number of the student it was decided to build a maximum of four exercises. A deadline for the readiness of the practical cases was also clear as classes needed to fit in a particular timeline within the study course.

At this stage of the project, even though the outcomes and needs were clear, quite many other issues were left open regarding the technical aspects of the simulations. The biggest concern was in fact if Cisco Packet Tracer was the correct IoT simulator to use, but also what could be achieved with it. Historically the tool had been in fact utilized only for networking exercises purpose, but never for IoT cases.

Reasons however why Cisco Packet Tracer was believed to be the correct choice were mostly because tool was also used in few Cisco NetAcad IoT courses but also because

it had the big advantage to have already proven networking features utilized by many students worldwide.

Once the requirements were fixed the real development of the exercise started. This work was mostly divided in two phases: first to familiarize with the tool and understand what were its IoT capabilities and second to prepare the IoT automations.

As briefly introduced in the chapter 2, Cisco Packet Tracer is a Cisco proprietary tool utilized in many Cisco NetAcad courses. Even though the tool is quite popular there were not extensive guides or instructions publicly shared in the internet. There were however few blogs and Youtube videos that helped for some specific aspects.

When accessing to the Cisco NetAcad portal, however, there were many excellent online courses that explained the functioning of the tool. Along with specific Cisco Packet Tracer classes, the tool was also utilized in many other networking courses, helping students to gain knowledge in steps.

Specifically for this thesis work, the knowledge of the simulator was built by following: Introduction of Cisco Packet Tracer (0118), Introduction of Cisco Packet Tracer (1217), Packet Tracer 1o1 (2016-11) and Intro to IoT – English – 2016.

Online classes were usually structured by both viewing some theoretical material and also via more step-by-step videos on how to use the tool. The biggest sections of the classes were however the practical exercises. For each session in fact detailed exercises were required and only when the setup was correct the exercise was passed.

The last part of the courses was usually regarding a quiz exam both including theoretical and practical questions on the exercises.

While attending the online studies one could observe, however, that most of contents were related to the basic function of the tool and on the networking part. Only basic IoT components were introduced by building simple IoT automations.

When the four IoT simulations for the students were build a lot of time was spent in order to understand the logic on how the more complex devices functioned.

Also microcontroller programming part was not included in the Cisco classes when the above courses were attended in early 2018.

By spending time working with the tool it came clear the fact that most of the requirements for the Internet of Thing course could be met by creating simulations with Cisco Packet Tracer.

The second sub-part of this project phase continued by creating the four simulations. For all the cases the methodology was similar: firstly a basic connectivity layer was build, making sure that connectivity between all the networking components were established, then IoT devices were added and last the simulations were created.

Originally the microcontroller examples were not part of the requirements, however, one example of interaction between an SBC and a sensors was added to each IoT simulation in order to give a more comprehensive IoT experience to the students.

Before completing to the final version of the IoT automations few basic simulation were shared with the IoT course lecturer in order to make sure that they were following the requirements. The only open point at this stage was how complex and complete these simulations needed to be.

Temporary versions were more simple compared the finished product however they already included most of the network parts and some of the IoT simulations. The major difference was regarding the Smart-Industrial case. This was not in fact a part of the original example bundle but it had been added in later phases in order to deliver an extra, more complex, simulation.

After few iterations and adjustments the IoT exercises were ready and approved by the course lecturer. A small guide was also built for each case to be used as a reference document for the students. The guide included a generic overview of the network and the IoT layout, information about used IP addresses and IoT credentials and also few suggestions how to expand the simulation further. The main content of this document is described in the chapter 4 in this thesis work.

Once the development of the exercises was completed the next step of the project was to introduce them to the students. Arrangements done by the IoT course lecturer consisted two practical classes planned a week apart in the Helsinki Metropolia premises.

The first class was split into three sections: a brief overview of Cisco Packet Tracer, a small practical networking exercise for the students to familiarize with the tool and for

last the introduction of the IoT components within the tool. Apart from the first section, the rest of the classes were held in the PC laboratory of Metropolia.

As discussed further in the chapter 5 few delays happened before starting with the networking exercise, mostly due to the fact that not all the students had a valid NetAcad account open, necessary when logging into Cisco Packet Tracer.

Also some comments on the tight schedule are shared in the chapter 5.

The second class, organized a week apart, started by introducing the four IoT simulations to the students. While giving an overview of the exercise, mostly explaining the IoT scenarios and not focusing on the networking part, also tips and deeper aspects of the tool were shared with the students.

Once introductions were completed the students gathered in the original groups, defined at the beginning of the IoT course, in order to adapt the own IoT business case to the IoT simulations.

Idea was that they could utilize the pre-configured four IoT simulations to adapt them to follow their own cases. Unfortunately few groups could not achieve that as Cisco Packet Tracer did not have the IoT sensors required by the business cases, mostly medical, automotive and wearable sensors.

These groups were asked to expand one of the four simulations and practice by adding more IoT components and the backend simulation intelligence.

During the class constant support to the students was provided by sharing tips on how to configure the devices, helping them in setting up network connections and also building microcontrollers programming logic.

Classes were very effective and at the end of the second session most of the groups achieved a very good level of IoT simulations utilizing basic networking components. More advance groups also experimented with basic microcontroller programming.

As discussed deeper in the chapter 5 the time spent for students practicing on the simulations was unfortunately not enough in order to gain a full understanding of complex IoT simulations using automations, sensor variables or microcontroller programming.

Last part of the project was to gather feedback from students on the two classes and on the four IoT simulations. A feedback form was distributed in paper form and by mail to the students, unfortunately only seven forms were returned.

Feedback document can be found in the Appendix 3 and conclusions are formulated in the chapter 5.

4 IoT environment Simulations

Below chapters are focusing on the core exercise shared with the students, giving first a general introduction of the simulations and an overview of Cisco Packet Tracer tool.

In the second part of the chapter the four IoT automations are analyzed in more details giving more technical explanation on the network and IoT aspect of the exercise, also giving few suggestion for future expansion.

Purpose of the explanations are to guide the reader to fully understand the background of the IoT simulations and also use this document as quick reference guide when practically working on the Cisco Packet Tracer exercises.

4.1 Exercises Introduction

This chapter includes and analysis of the four IoT simulations discussed with the students in the practical classes. These were used as a starting point to learn how to utilize IoT functionalities in Cisco Packet Tracer. For the groups where their own business case was not directly achievable with the tool it was requested to modify one of the four simulations in order to practice with IoT components.

Every exercise consisted of a pre-defined physical layer separation, a fully connected and configured network where IoT components can be connected, few examples of IoT Smart devices and a backend logic to show how these devices could interact automatically. For each case there was also one example of sensor-to-actuator logic using a microcontroller board and custom Blockly program.

All the examples came with a dedicated documentation showing the network layout, network details (Appendix 2), overview of the infrastructure, Blockly program logic (Appendix 1) and some future ideas how to expand the simulations. Documentation was distributed to the students along with the Cisco Packet Tracer exercises.

The four examples were called: Smart-Home 1, Smart-Home 2 (SaaS), Smart-Campus and Smart-Industrial. The first two simulated an IoT automated house where the IoT components were connected to a local WLAN network and where smart devices were interacting with each other. The difference between the two home examples was that in the first case the IoT backend functionalities were hosted within the home LAN. In the

second case the devices were only having a connectivity layer inside the house LAN, but the IoT server components were hosted in a third-party network, simulating a provider cloud Software-as-a-Service scenario.

Both of the cases included a remote network, in the form of a corporate office network or a mobile 4G network, where home owner could connect to in order to monitor the IoT activities in the house.

The third simulation was about an interconnected university campus network where IoT smart devices were connected. Along with the two basic networks, classrooms and apartments, a third network was connecting all the IoT devices and the backend IoT servers. Through a user authorization mechanism from any point of the network it was possible to access the IoT devices and monitor their status.

The last case represented a more complex industrial environment where five networks were connected. Two of networks were connecting the IoT devices, via a switch or a dedicated 3G network, for an electricity production. The electricity was then consumed by a third network of IoT devices used in a production line. The remaining two networks were utilized to simulate a corporate office remote area and a more important control room, where backend IoT server was located and where users could connect to control the IoT devices.

In order to be able to interact with the four simulations some basic knowledge of Cisco Packet Tracer were required, due to the fact that IoT simulations were not only exploring the basic network components the students needed to be familiar with concepts such as physical containers, environmental variables, IoT backend servers and IoT component programming. The following chapter covers the basic non-networking components concepts, a deeper practical explanation was given to the students during the two practical classes.

4.2 Cisco Packet Tracer IoT Technology Introduction

Along with the standard networking functionalities, in the version 7.0.0, Cisco Packet Tracer received an important upgrade with IoT components [25].

This chapter is mainly focusing on the IoT functionalities of the tool, only giving a brief overview of the network components. Purpose of the study course was on Internet of Things and not networking, hence all the four IoT simulations had a pre-configured network to allowing IoT students to concentrate more on the IoT aspects.

In the four IoT simulation the network devices were the backbone elements for connecting the IoT devices allowing them to interact between each other.

Every simulation had a dedicated network layout, however basic components were similar among the examples. Routers, switches and wireless router were in fact commonly used to create the network foundation.

In the figure below it is possible to see, as an example, the different routers offered by Cisco Packet Tracer, main difference to consider when placing the device in the simulations are the possibly hardware limitations that are coming with the devices, in terms of number of ports available, options to change the network interfaces, number of expansion slot etc. An extensive list of switch, server, PC and laptop is also available in the tool.



Figure 5 - List of routers in Cisco Packet Tracer

In special cases, such as the two Smart-Home examples, also a simulation of an internet connection was used. This connection was aiming to simulate a normal Internet Service Provider (ISP) connectivity, giving the possibility to the home owner to remotely connect to the own home network from an external network, such a corporate office or a cellular network. In the second Smart-Home simulation the ISP was also used to link the home devices to the backend IoT intelligence as IoT functionalities were provided as-a-service.

Additionally, in the Smart-industrial and one of the Smart-home cases, a 3G networks were also implemented. Additional component such cell-tower and backend servers were necessary to ensure the functioning of the network. Utilizing cellular network gave more flexibility while connecting IoT devices to the network. As setup was very easy and no limitation, such range and number of connected devices, are present as it would be in real WLANs.

Once a network devices was placed into the simulation, the next step was to configure it. Cisco Packet Tracer offered two options: configuration of the device via a user interface or via a Command Line Interface (CLI). While using CLI method specific Cisco commands need to be used and real device logic applies. Configuring via user interface was more intuitive, did not require knowledge of Cisco commands but limited the amount of parameters that could be set.

Depending on different setups different type of cabling were used to interconnect the network devices such as copper straight cables, copper crossover cables and optic fast-Ethernet cables. Also IoT custom cables were utilized in the microcontroller example.

As one can see in the below figure, Cisco Packet Tracer, offered several cabling options, however an important feature was the auto-cabling option (lightning icon in the Figure 6). When selecting it, the tool would automatically chose the correct cable to connect two network interface. For learning reason this option can also be disabled.



Figure 6 - List of available cables in Cisco Packet Tracer

Another important feature in the Cisco Packet Tracer tool is the possibility to separate the network at physical level using sub-environments such as city, building, containers and wiring cabinets. From the main view of the tool it is possible to switch very quickly between logical and physical layer as illustrated in the Figures 7 and 8.

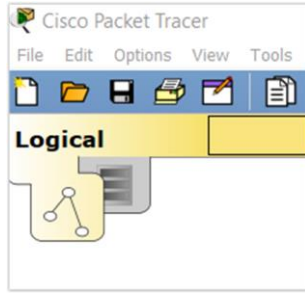


Figure 8 - Logical view

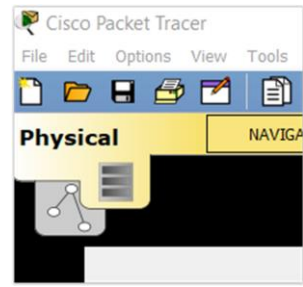


Figure 7 - Physical view

When creating network simulations, in the logical view, by default all the components are placed in the same physical space. For basic simulation this is not probably a detail that should be taken care, however for IoT simulations it was advisable to utilize different physical layers in order to be able to adjust the environment variable to influence the IoT devices behavior. Below figure shows the physical separation of the Smart-industrial exercise, where five different containers were utilized to physically split the various networks.

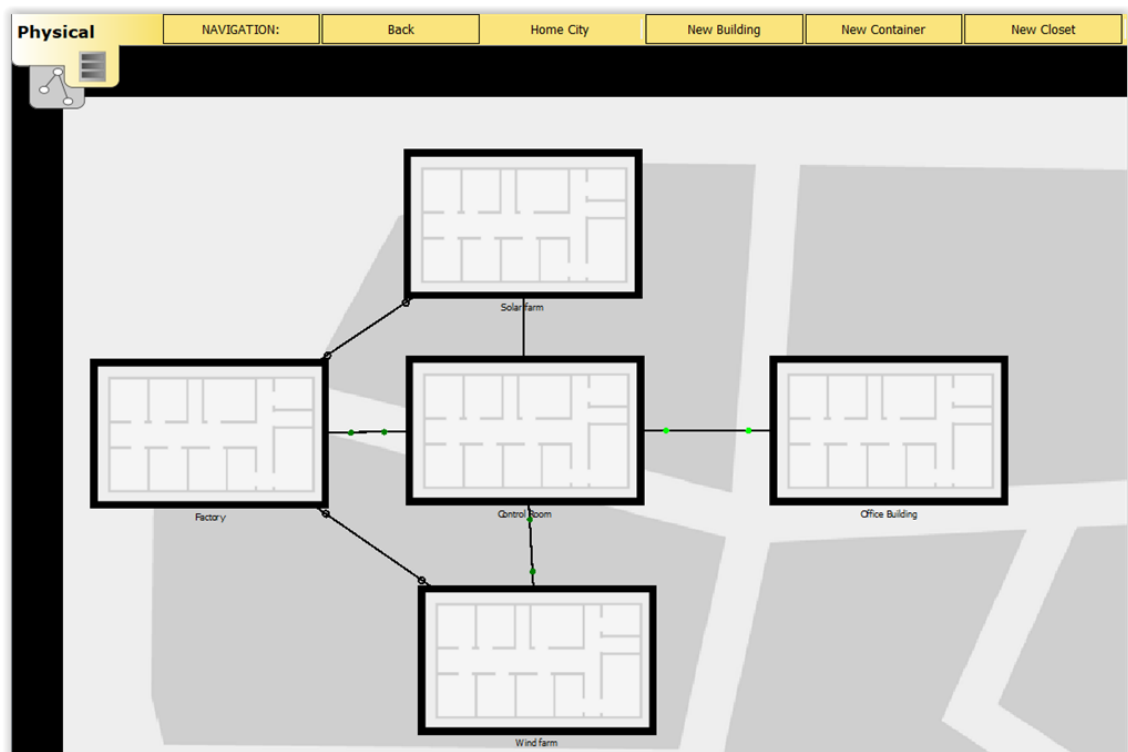


Figure 9 - Example of physical view from Smart-Industrial simulation

Physical separation helps to divide physically the network in several sub-areas introducing a more realistic aspect of it such as a need of backbone connections, a physical wiring length limitation and WLAN coverage limitations and a need of custom environmental variables.

Every physical sub-layer, except for the wiring cabinets, are coming with a large set of fully customizable environmental variables,.

Variables are tunable parameters for representing real life environments such as: amount of sunlight, air carbon dioxide concentration, gravity, winds speed and many more. In Cisco Packet Tracer there are more than fifty different variables that students can adjust accordingly based on a 24 hours time range.

In the Figure 10 below one can see the amount of sunlight and rain setup in the Smart-campus exercise.

Variables are necessary in order to influence the sensor behavior in the IoT simulations. Variables are in fact detected by the sensor and, as a consequence, actions are then triggered. Adjusting variables was also helping students in order to validate immediately if the IoT logic setup was done correctly.



Figure 10 - Example of environmental variable setup

In the provided four simulations the variables utilized were: sunlight and wind speed level, in order to boost electricity production done by solar panel and wind turbine. Also amount of raining manipulations were used during a random time of the day in order to activate the water sensors, sensors were then placed in the school sport field, to detect extra water level and stop the water irrigation sprinklers. In the Smart-Home cases also humidity and temperature levels variable were changed in order to fire up the home AC unit.

Creation of a smaller sub-container helped also to boost the environmental changes in the variable parameter. As an example, if an old car was placed in the city level and it was turned on, the level of carbon dioxide build up in the city level would have raised at a slower pace compared if the car was turned on in a sub-container. In the second case, being in a smaller space, like a car garage, the carbon dioxide sensor picked up carbon dioxide variation faster than in the city level.

Physical layers and network components are present in the Cisco Packet Tracer since many year, the real addition that make the tool capable to simulate IoT environments was the introduction of IoT devices.

The main categories are: smart-devices, sensors, actuators and microcontrollers.

In the below figure there is an example of a list of home smart devices that can be added in the IoT simulations.



Figure 11 - List of smart device

Smart-devices are devices that are fully capable to be connected to a wired or wireless network and where the behavior and interaction logic can be quickly set up by utilizing pre-loaded Python programs. These sensors include smart lights, AC units, coffee maker, alarm sirens, RFID readers and a long list of other sensors, such as carbon dioxide, humidity, temperature, water level etc.

These devices usually are plug-and-play type of elements and they just need to be connected to a local LAN. In some cases, if a wireless LAN was needed, a physical network configuration required to be done mostly by swapping the LAN network interface for a WLAN card.

Once connected to a network the devices needed to be remotely connected to the IoT backend server. Creating the IoT connection enabled the possibility for the users to check the status of the IoT devices from an IoT browser homepage. As illustrated in the Figure 12 below from a home tablet, also connected to the local WLAN, it was possible via browser to connect to a dedicated IoT homepage to monitor the list of all connected IoT devices.

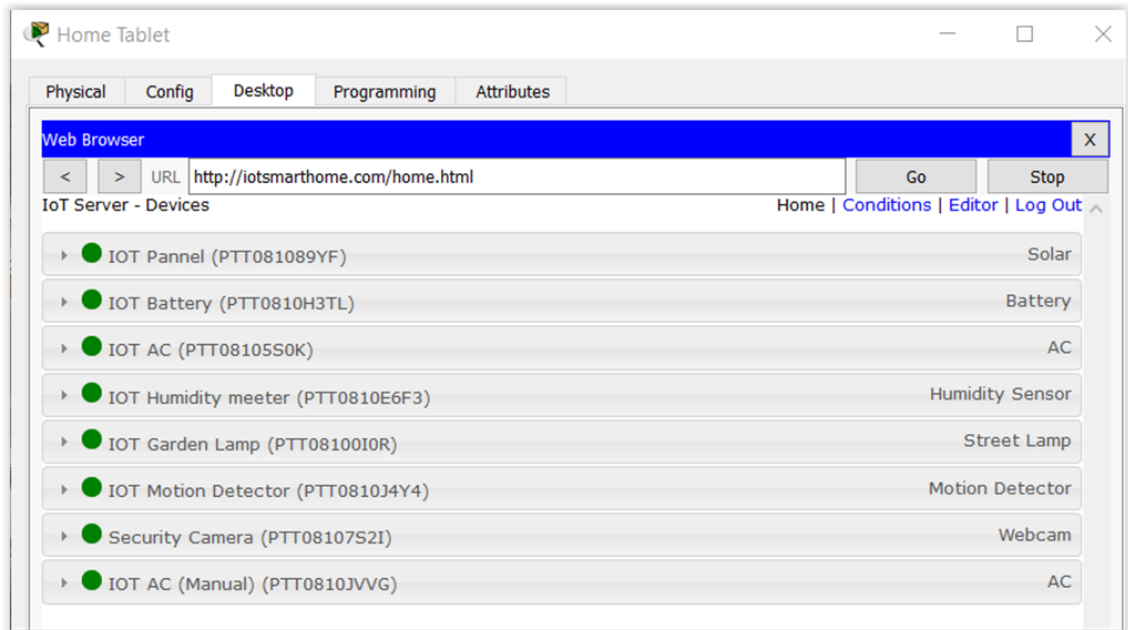


Figure 12 - Example of IoT homepage from Smart-Home 2 exercise

When accessing the list of the devices it was also possible to visualize their status but also interact with the device remotely. As an example one can turn on the garden light or check the remaining power of the IoT battery.

Furthermore, while connected to the IoT main home page, it was also possible to setup a basic IoT logic for creating an interaction between the devices. As demonstrated in the below Figure 13, the humidity sensor was connected to the home AC unit, starting AC in case humidity would pass the fifty percent threshold. More detailed explanation of the pre-set IoT logic in the four IoT simulations will be covered in the chapter 4.3.

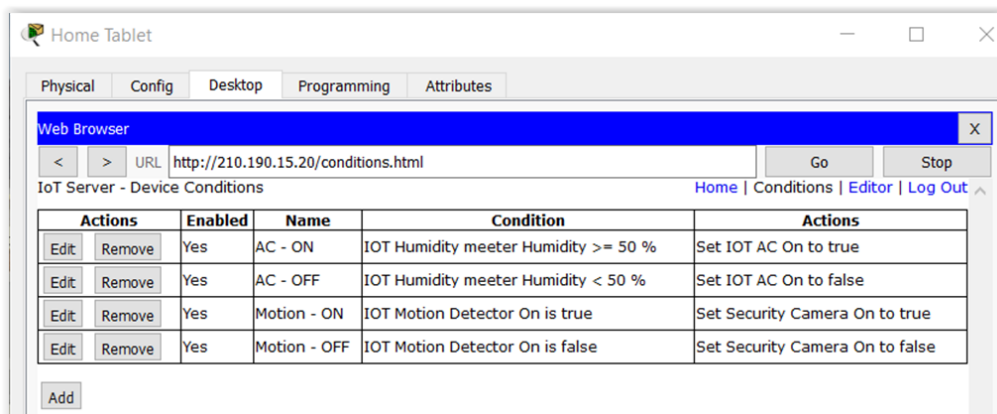


Figure 13 - Example of pre-set conditions from Smart-Home 2 exercise

Cisco Packet Tracer also offered the possibility to use other non-smart devices such as actuators. These type of components required additional actions in order to be connected and operated but, despite the extra complexity, offered a more realistic degree of simulation and flexibility when setting up the exercises.

Non-smart components were usually not network-capable, unless configured otherwise, and needed to be connected to a microcontroller board via some special IoT cabling. In Cisco Packet Tracer the microcontroller simulated an Arduino or Raspberry Pi devices that, based on a custom software, took the input from some device, mostly input sensors, and produced output, mostly operating actuators.

In order to create these simulations basic programming skills were required in Java, Python or Blockly. Creating Java or Python scripts enabled the possibility to fully command and control all the sensor and actuators. The use of Blockly language helps less skilled programmers to visually create the programming logic. All programming done for the four IoT exercises was done with Blockly, enabling all students to understand the logic behind.

Before programming a sensor or an actuator, students must understand the device specifications. A brief explanation of how the device behaves can be found in the specification tab when placing a sensor in a logical window of the tool. Specification included all the necessary information on how the device operates, what are the different states, what type of input the device would expect or output that the component would generate. For more complex devices also an example of API command was suggested.

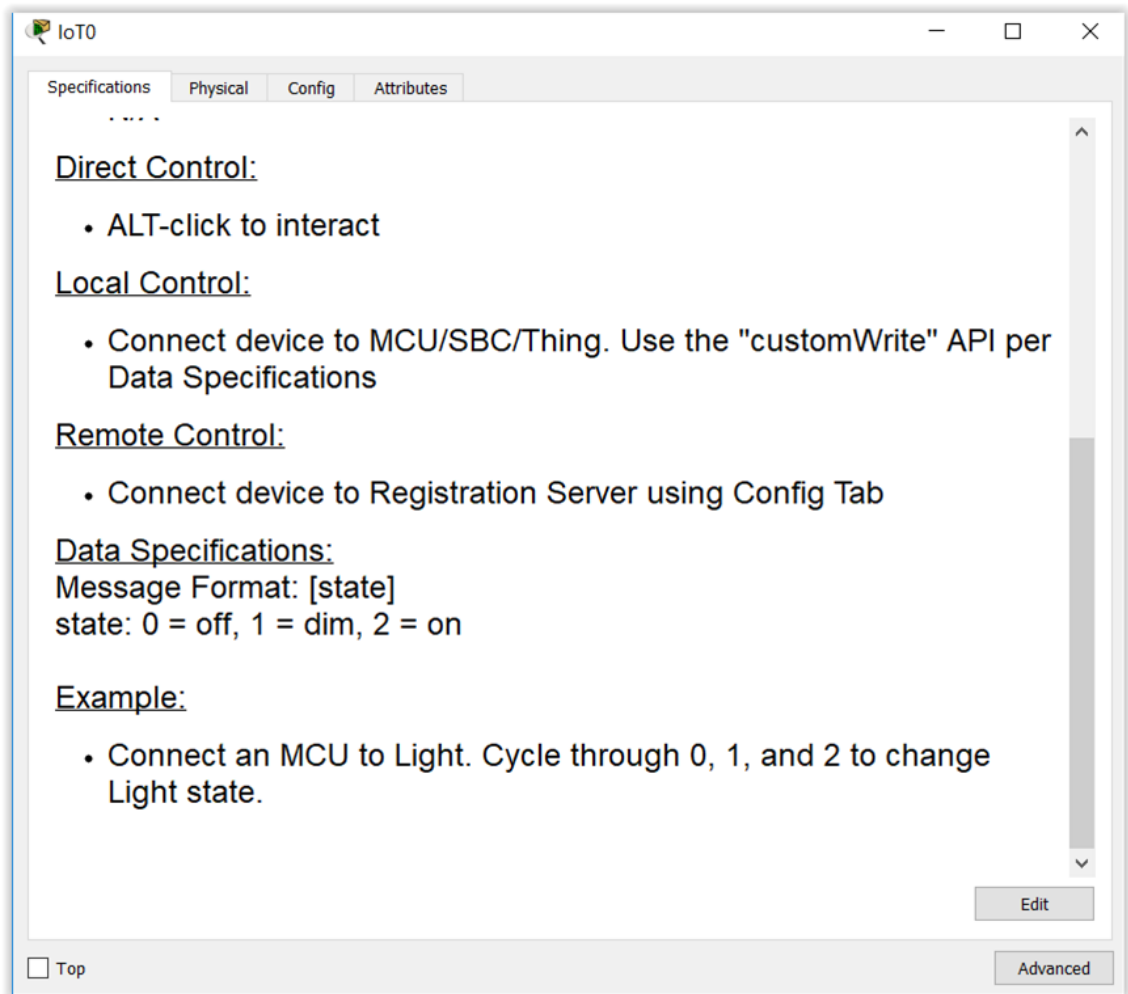


Figure 14 - Example of a smart-lamp specifications

Figure 14 above illustrates, as an example, the behavior of a simple smart-lamp. The lamp can be directly controller by pressing ALT on the keyboard and clicking on it. Lamp could also be connected to a microcontroller and a customWrite command would need to be used in the output pin to operate it. Lamp is also having three states: off, dim and on. So the previous customWrite command should also carry a variable to specify the state.

Cisco Packet Tracer offered for each device a very helpful and complete specification tab.

Another very important Cisco Packet Tracer feature worth to mention was the possibility to switch from a real time to a simulation mode. The first mode enabled the possibility to

create the underneath network, connect IoT devices and define IoT backend logic. However, only in the simulation mode, it was really possible to validate that the network communication layer really happened between the devices.

In the simulation mode it was in fact possible to simulate a packet traffic between nodes and devices in order to check the connectivity, routing protocols and other network logic. This mode helped to physically visualize and troubleshoot any kind of network, for example setting up pings, or more complex packages, between nodes. The simulation mode also listed in real time the various packages broadcasted in the network, allowing to deep dive in the payload of the network package.

Vis.	Time(sec)	Last Device	At Device	Type	Info
	0.424	Panel 1	Solar 1	IoT	■
	0.723	--	Solar 1	IoT	■
	0.723	Solar 1	IOT Motor	IoT	■
	0.762	--	Wind 1	IoT	■
	0.762	Wind 1	IOT light	IoT	■
	1.036	--	IOT light	UDP	■
	1.037	IOT light	IOT Ind...	UDP	■
	1.043	IOT Ind...	IOT Ind...	UDP	■

Reset Simulation Constant Delay Captured to: 1.054 s

Figure 15 - Example of packages captured in the simulation mode

4.3 IoT Simulations

This chapter is a guide for the four IoT simulations presented to the student during the practical classes.

Purpose is to analyze deeper the cases separating the network and IoT layout, giving a deeper explanation of the purpose of the simulations, presenting all the information needed to utilize the exercises but also giving suggestion how to expand the exercises further.

4.3.1 Smart-Home 1

Smart-home 1 was the first of two simulations that covered the domotic area of IoT. Smart devices were in fact connected to IoT in order to simulate full components interaction and capability to remote control the devices. Home owner in fact, after connecting via browser and pass the authentication, was able to command garage door or the house ventilation but also check the current status of the alarm system or the level of carbon dioxide in the garage.

This case also offered the possibility to the students to expand the simulation utilizing the remote corporate office network in order to build a remote access to the home LAN.

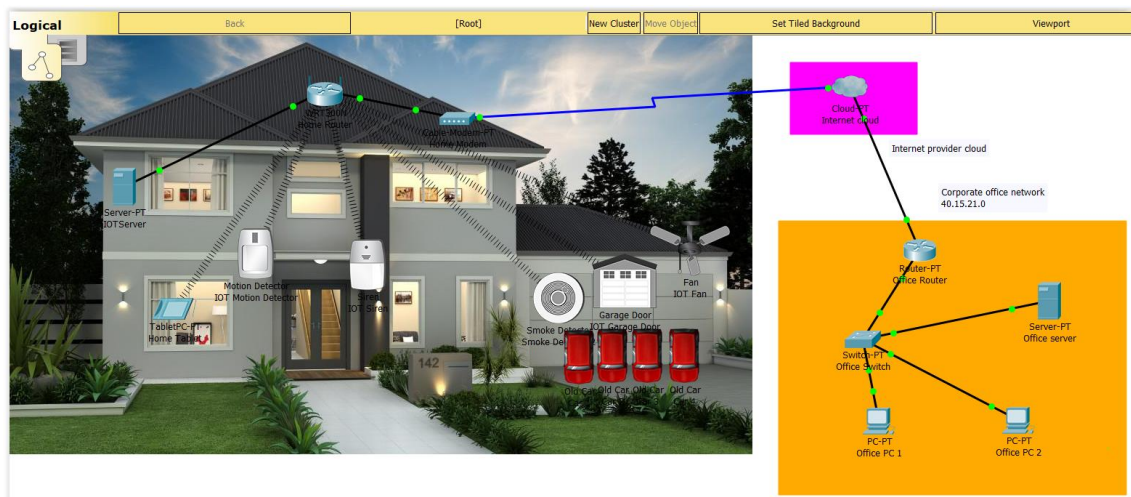


Figure 16 - Cisco Packet Tracer layout of Smart-Home 1 simulation

Network Layout

This exercise had the simplest setup compared to the other simulations.

Network was logically separated in three areas: home network, ISP Cloud and corporate office network.

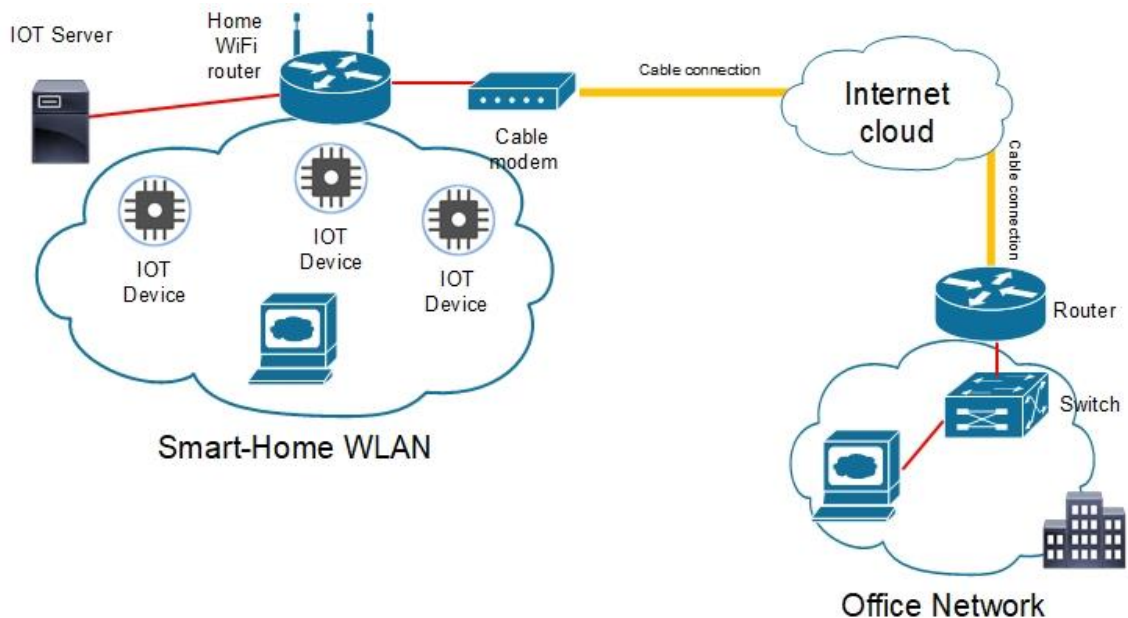


Figure 17 – Smart-Home 1 network topology

As one can see in the Figure 17 above the heart of this simulations was the home network, all IoT devices, home tablet and IoT backend servers were connected to the home WLAN. Network was simulating a normal home wireless network where every appliance was connected to a wireless router, router was then connected to a dedicated cable modem. In the exercise the modem was required as wireless router was only equipped with Ethernet ports and ISP connectivity was provided via coaxial cable.

Internet settings of the router were set on default DHCP ISP values, internal WLAN DHCP was however disabled via the Graphical User Interface (GUI), as local server connected in the WLAN was functioning as DHCP. Only other settings in the WLAN router were the home wireless SSID and password.

All wireless devices needed to use same SSID, password and DHCP default settings, except the local server that used 10-class static IP.

Static IPs ensured that, even if WLAN router was rebooted, the server IP remained the same, without having the need to re-configure the devices with a new IoT server IP.

The server, in addition to DHCP services as early mentioned, also provided IoT and DNS functionalities. IoT features were needed in order to offer backend intelligence to the IoT simulation and for being able to host the IoT homepage where home user could connect. DNS service was also required in order to translate the IoT homepage URL into the own IoT server IP.

The second network provided in the exercise was the ISP cloud. This was an artificial simulation of relay server that created the communication between two separate interfaces, server was setup to connect the coaxial cable from the home modem network to the Ethernet cable coming from the office router. No further configuration parameters were allowed in this device.

The corporate office network was a basic setup with a router connected both to the internet provider and also to a local switch where also two PCs and one server were connected. Router NICs were set with ISP IP in one interface and the LAN IP in the other, basic RIP protocol was also utilized in the setup to enable connectivity between the two networks.

The office switch and the office PCs utilized the standard default configuration. Local server was configured to utilize static IPs and to provide DHCP functionalities within the office LAN.

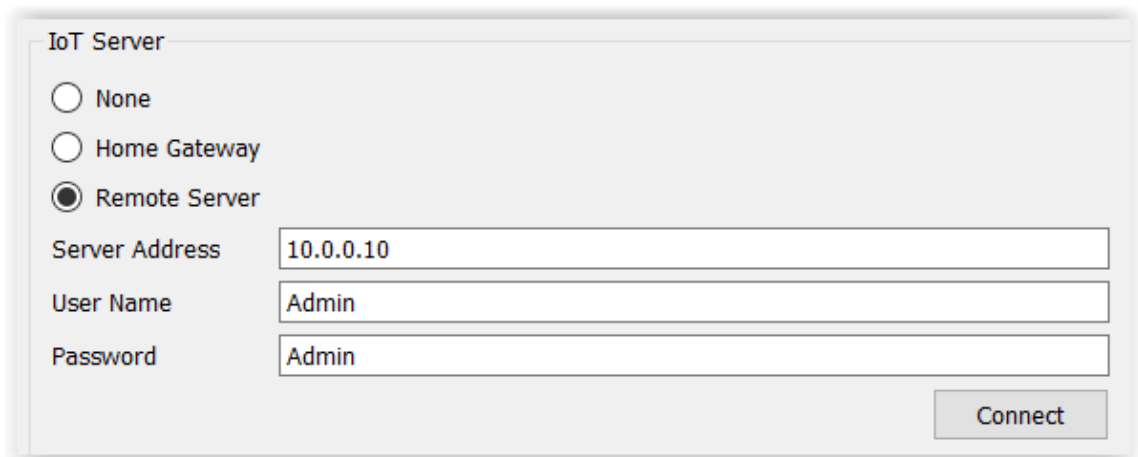
IoT Layout

For this simulation, as mentioned earlier, all IoT devices and IoT server were connected to the same WLAN network.

IoT logic connectivity was established on the top of the network connectivity.

As shown in the Figure 18 all IoT devices must be set to be connected remotely to the IoT server using previously created username, password and the server IP.

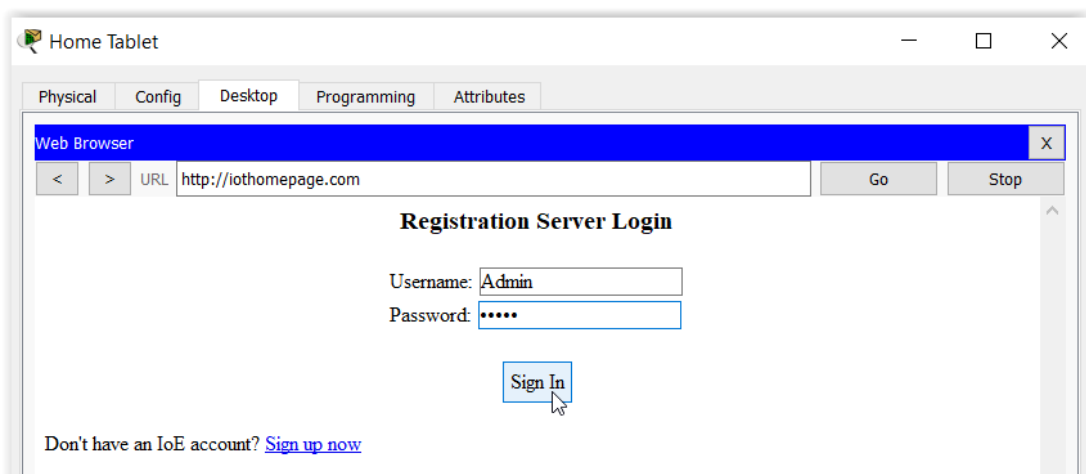
Successful connectivity was established when “Connect” button changed into “Refresh”.



The screenshot shows a configuration window titled "IoT Server". It contains three radio buttons: "None", "Home Gateway", and "Remote Server". The "Remote Server" option is selected. Below the radio buttons are three text input fields: "Server Address" with the value "10.0.0.10", "User Name" with the value "Admin", and "Password" with the value "Admin". A "Connect" button is located at the bottom right of the window.

Figure 18 - Example of IoT setup of motion detector

All device must use the same IoT credentials, same credentials were also used by the home owner for passing the authentication when connecting via browser to the main IoT monitoring homepage as shown in the Figure 19.



The screenshot shows a web browser window titled "Home Tablet". The browser's address bar shows the URL "http://iothomepage.com". The page content is titled "Registration Server Login". It features two input fields: "Username:" with the value "Admin" and "Password:" with masked characters "*****". Below these fields is a "Sign In" button. At the bottom of the page, there is a link that says "Don't have an IoE account? [Sign up now](#)".

Figure 19 - Login to the IoT homepage

As the IoT server was setup also for DNS services, the iothomepage.com was translated with the IoT server static IP.

Once user was connected to IoTHomepage.com it was possible to visualize the status of the IoT devices but it is also possible to review the interaction logic between them.

As visible in the Figure 20, in this exercise five IoT smart-devices were used: motion detector, siren, garage door, fan and smoke detector. Also cars were utilized, but these were only for influence the environmental variable of the simulation.

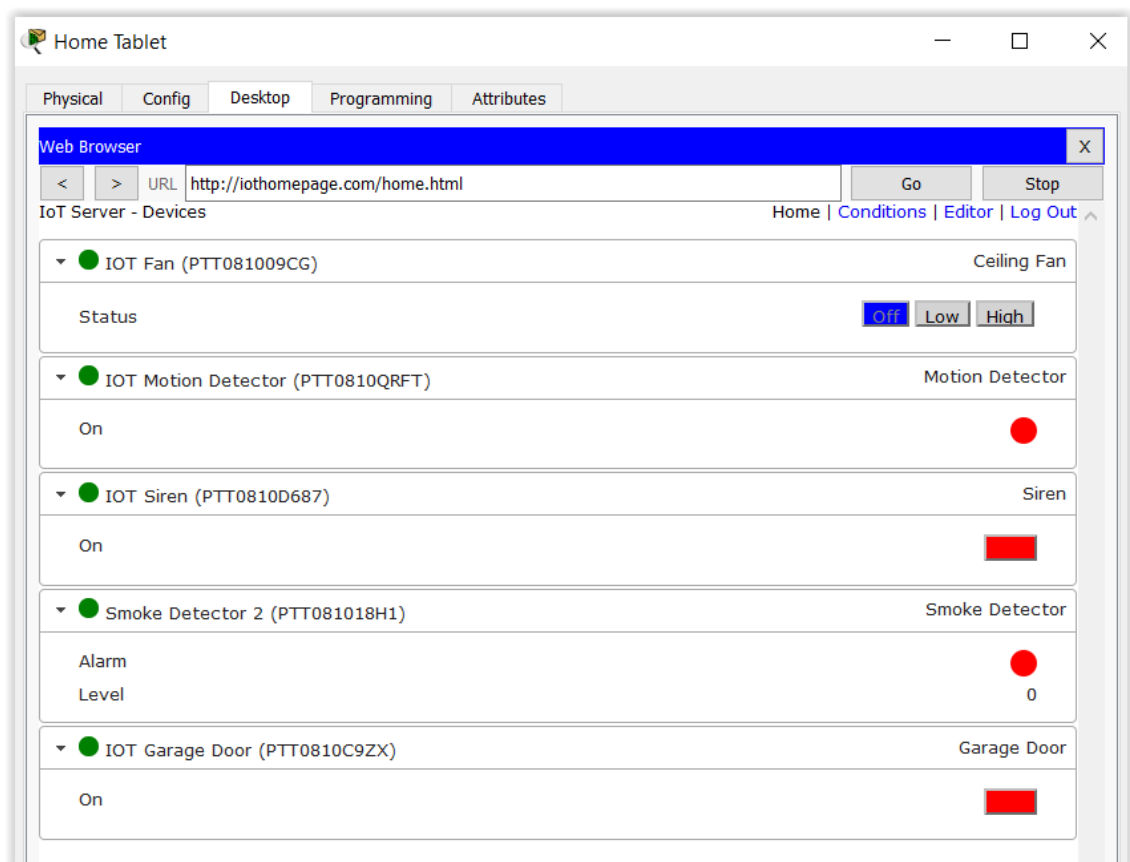


Figure 20 - Status of the IoT devices connected

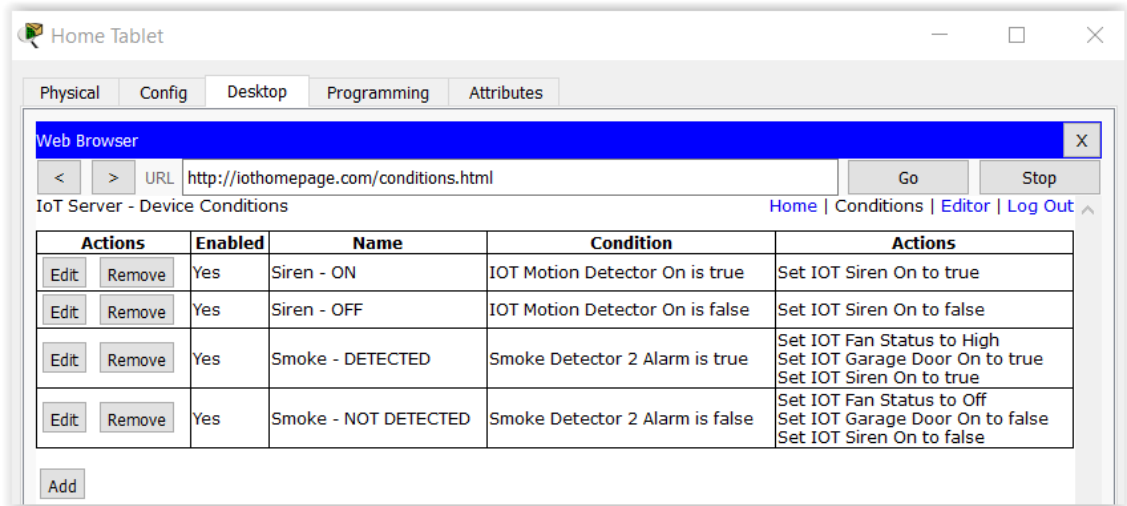


Figure 21 - Pre-set conditions

As one can see in the Figure 21 while being connected to the IoT homepage it was also possible to browse to the conditions page and visualize the two main IoT simulation examples.

The first case utilized a motion sensor to temporary activate the alarm siren as sort of basic home alarm system. Logic is visible in the previous Figure 21, simply, when sensor was triggered, the siren was set on. When sensor was not detecting any movement, and after a pre-set timeout, the siren was set to be off. This logic was easily validated by pressing ALT on the keyboard and hovering the mouse cursor on the motion detector, siren was then activated immediately as shown in the Figure 22 below.



Figure 22 - Active IoT siren

The second simulation logic was regarding the home garage, in this case an additional separate physical container was utilized in order to influence the environmental variables.

As visible in the Figure 21, the sensor was set to turn on the fan, to open the garage door and to sound the alarm siren in case the carbon dioxide level within the container reached a pre-set threshold. No action were triggered if level of carbon dioxide was under the threshold.

In order to raise the carbon dioxide level few cars were utilized. By pressing ALT on the keyboard and clicking on the car with the mouse, the car turned on. Once all cars were on the level of carbon dioxide was rapidly raising, till the moment that alarm rang, garage door opened and ceiling fan started venting out the carbon dioxide. Simulations statuses are visible in the below Figure 23.



Figure 23 - Simulation of carbon dioxide building up in the garage

IoT Microcontroller Example

One example utilizing an SBC was also included in the Cisco Packet Tracer simulation. In this scenario none of the sensor or devices were connected to the home WLAN or any other sort of network. All components were connected via custom IOT cables to the SBC board.

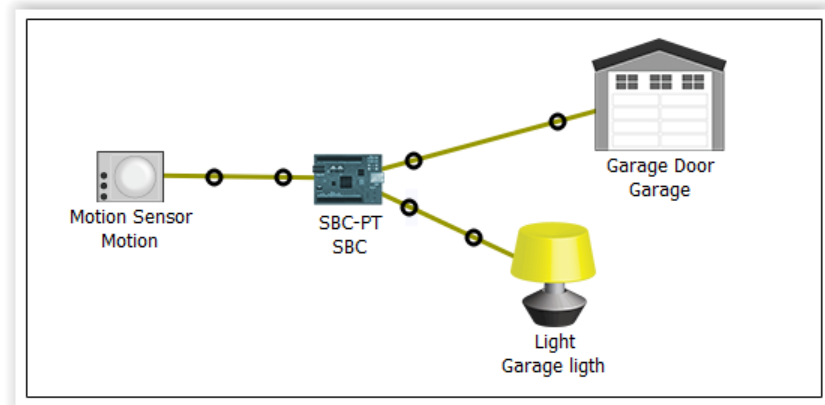


Figure 24 - Smart-Home 1 microcontroller example

The example simulated the garage door opening, along with light turning on, when the motion sensor detected some movement. In this case the intelligence of the simulation was not provided by the backend IoT server but by a custom software running on the SBC board.

The logic of the software was very simple, sensor was directly connected in an SBC input pin. When objects were detected, the sensor fed a value to the SBC input pin, the microcontroller was then comparing it to a pre-set threshold value and if the logic clause was met, a command to the output pins was sent. Command was a custom API command that opened the garage door and turned the lights on.

As previously discussed in the chapter 4.2 the sensor, light and door specifications were visible in the device specification tab.

The logic of the Blockly program running on the SBC is visible in the Appendix 1.

Future Expansion of the Simulation

Being the exercise very simple students could expand the simulation very quickly.

On the network prospective all the logic was located in the home network, students could use the corporate office network to build the possibility, via VPN, to the home owner to remotely connect to the iothomepage.com.

On IoT level more complex simulation scenario could also be added to build a more resilient alarm system, using cameras, RFID readers, tripping sensors but also to include room climate control or electricity production, via solar panels, on the roof.

The microcontroller example could be expanded by connecting the device to the network or to add more sensors, actuators and more complex programming.

4.3.2 Smart-home 2 (SaaS)

Smart-home 2 was a variation from the previous example of smart house, the main difference was on the backend IoT intelligence, now provided by a remote provider as-a-service.

In this case the home owner needed to only setup the home WLAN and then backend logic was provided by remotely connecting to a cloud server hosted by a third party provider. In this scenario the home owner had also the possibility to remotely connect to the main IoT control page using own mobile phone via a 3G network.

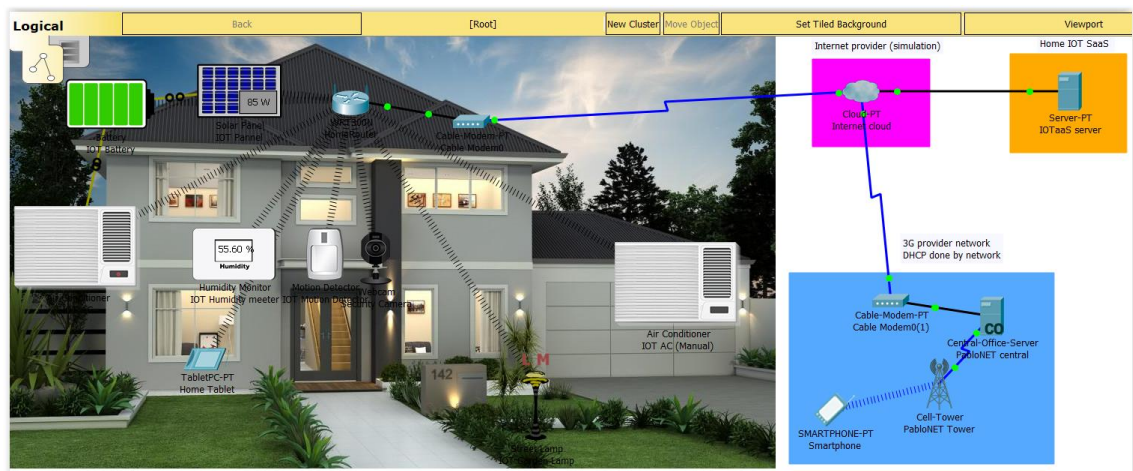


Figure 25 - Cisco Packet Tracer layout of Smart-Home 2 (SaaS) simulation

Network Layout

Smart-Home 2 had a more complex network setup compared to the other domotic example. In this case the network layout was divided in four areas: home network, ISP provider, 3G mobile network and the backend IoT.

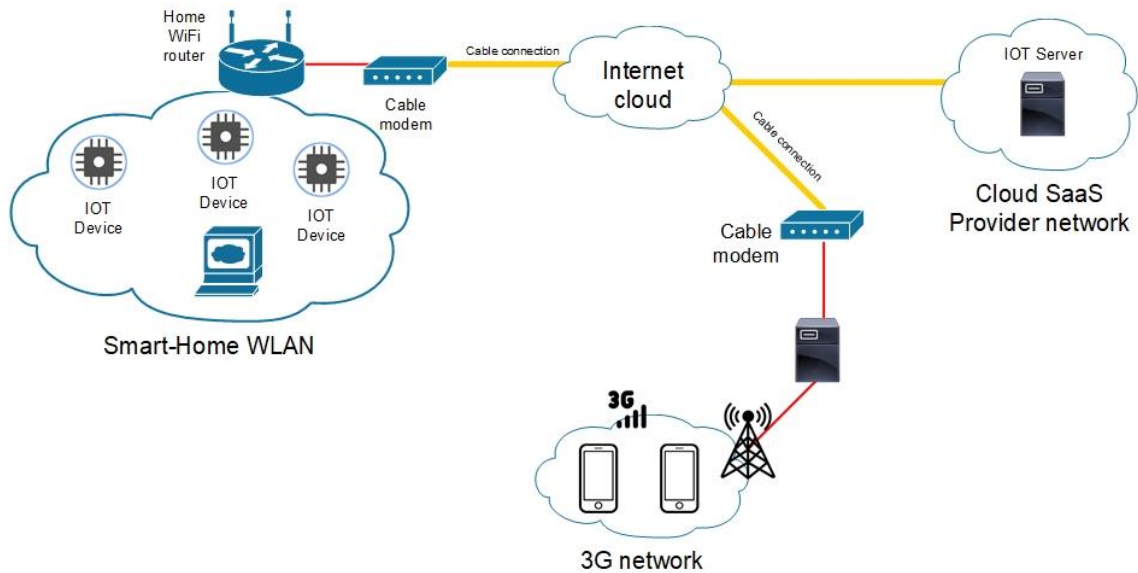


Figure 26 - Smart-Home 2 (SaaS) network topology

Despite the need of four network, the setup was simpler compared to previous scenario as the IoT, DNS and DHCP server was not connected to the WLAN. Wireless network was provided by a wireless router that, in this case, also provided DHCP functionalities.

Being a simple home network the device setup was done only by using specific SSID and password. Home cable modem was also utilized only to connect the router to internet, no configuration could be done on this component.

The ISP network, similarly as in the other smart home example, was artificially connecting different interfaces to each other in order to simulate internet connectivity. Coaxial cables coming from home modem and 3G network provider were linked to the Ethernet cable where the cloud IoT server was connected.

In order to simulate a remote network, a mobile 3G provider infrastructure was utilized in this example. This allowed the home owner to connect to the IoT monitoring page using its own mobile outside the home WLAN. Cellular network setup was very straight-forward

and only required a cell tower and a central office center server. Server consolidated all the signals coming for the tower coaxial cables into a Ethernet backbone connection that, via cable modem, was then connected to the ISP network.

Both cell tower and central office server had only a limited amount of parameters that could be configured. A maximum of six cell tower can be connected to the central office server.

The handheld devices were then connected to the 3G cell tower network by setting up the correct APN name.

In order to simplify the cloud provider network, the remote IoT server was directly connected to the ISP without the usage of any router. Server utilized static IP in order to make sure that IoT device could always connect to it. DNS services were also running on this server resolving the IOTSmarthome.com URL address into its own IP.

The setup of the SaaS cloud services utilized in the exercise was artificial and would not be realistic in a real SaaS scenario.

IoT Layout

As in the other smart home example, all IoT devices were connected in the same network, the only difference was in the IoT server being not connected to the home LAN. This required that local WLAN router acted as DHCP server but, for IoT functions, device were connected to the remote IoT server.

As previously all the devices were also sharing the same IoT username and password, same credentials needed also to be used by the home owner in order to remotely connect, via browser, to the IoT homepage.

As one can see in the below Figure 27, this simulation had already eight pre-configured IoT Smart-devices: solar panel, battery, two AC units, humidity sensor, motion sensor, webcam and a garden light.

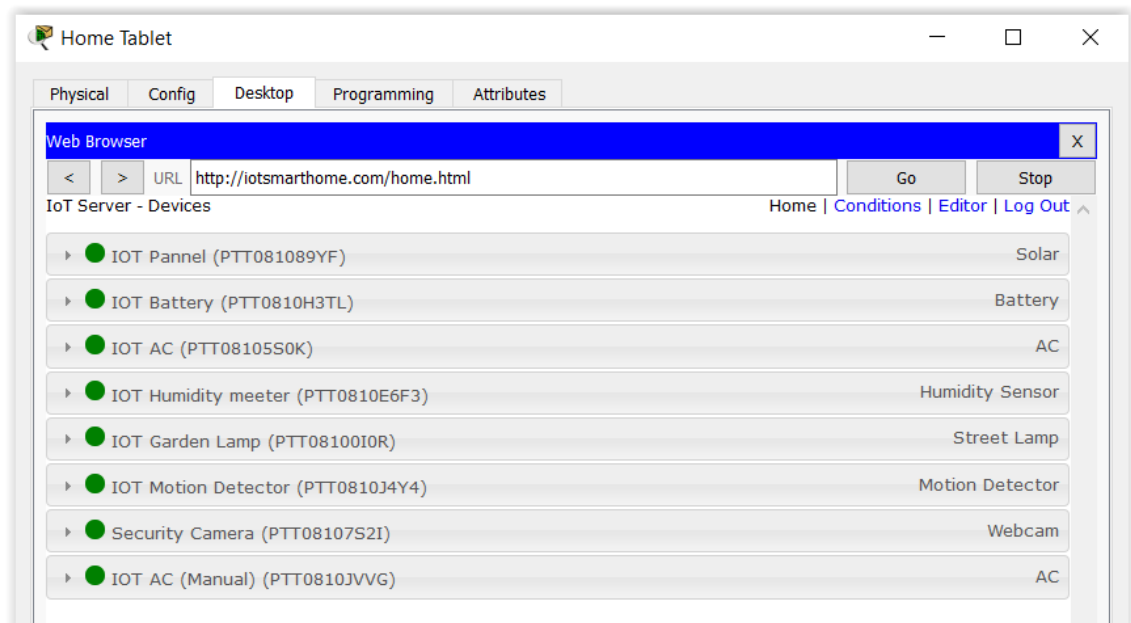


Figure 27 - List of IoT device connected

Also two IoT simulations were pre-configured and visible from the main IoT homepage, as shown in the Figure 28.

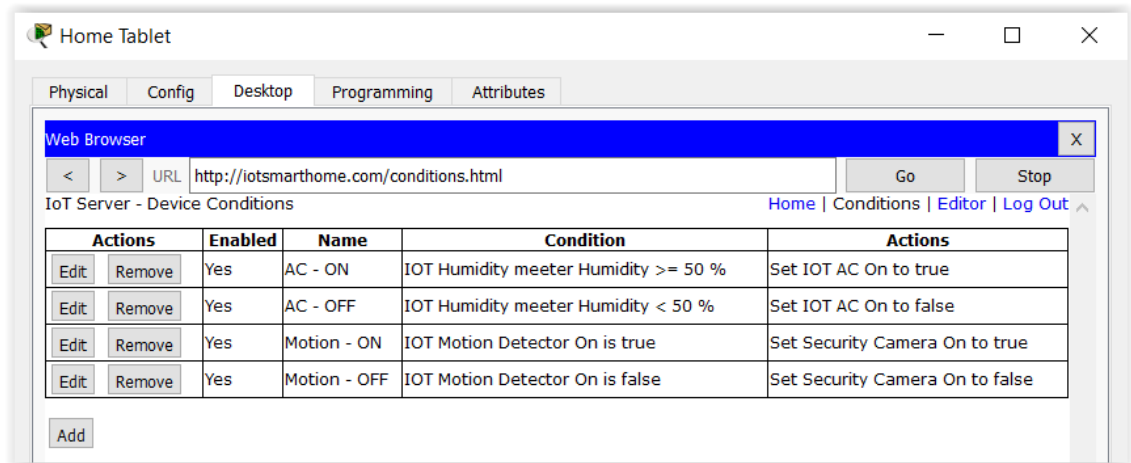


Figure 28 - Pre-set IoT simulations

First automation replicated a scenario where an IoT humidity sensor was installed in the house in order to monitor the ambient humidity. In case the detected humidity exceed the pre-set threshold value the AC unit was automatically power on, dropping the level of humidity. Ambient level of humidity could be monitored from both the sensor or by selecting if from the main IOT homepage.

To increase the complexity of the simulation, a solar panned and a smart battery were also added and connected to the AC unit.

The concept was that, during the daytime when sun shone, the solar panel produced electricity, charging the battery. At the same time however the sun increased the humidity in the house causing the AC unit to start. Being the AC unit connected to the IoT battery it was possible to observe that battery stored the electricity when AC unit was off, but drained fast when AC was on.

It is worth to mentioned however that, due to Cisco Packet Tracer logic, the AC did not stop in case battery stored power reached zero and also that battery did naturally loose the charge even if AC was not on.

Both amount of electricity produced by the solar panel and remaining power in the battery could be seen in real time from the main IoT homepage as illustrated in the Figure 29.

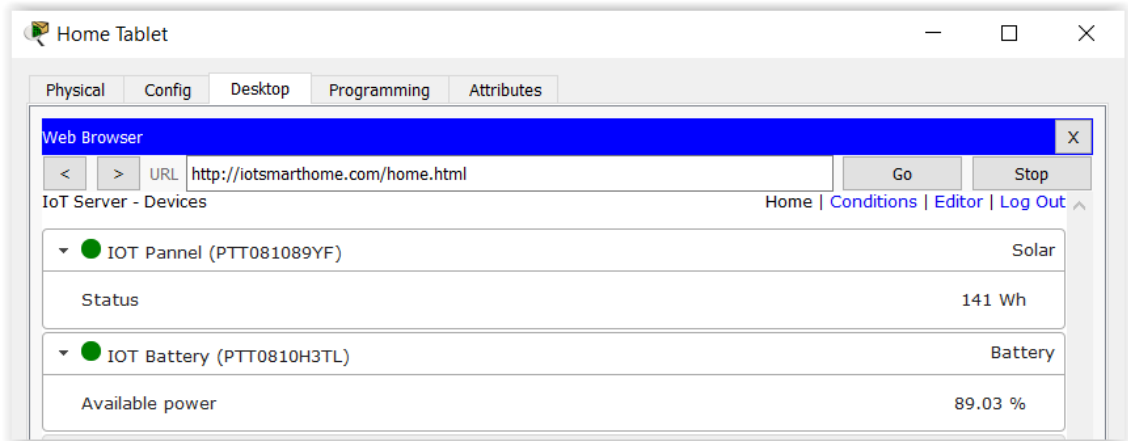


Figure 29 - Monitoring of power produced and stored by IoT devices

The second IoT simulation included in the exercise was similar to the previous Smart-home alarm system. Main difference in this case was regarding the motion detector triggering not a siren but, for a few second, a webcam broadcast if something was detected.

Simulation could be validated by pressing ALT on the keyboard and hovering on the motion detector sensor with the mouse, from the IoT home page the webcam then broadcasted some images.

Figure 30 shows the case when no objects were detected and webcam was off, in Figure 31 is possible to see that sensor was triggered and webcam was showing images.

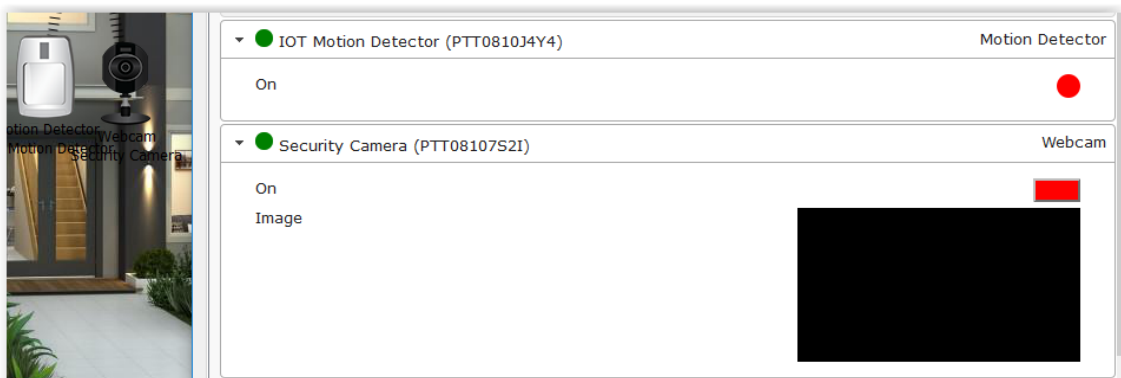


Figure 30 - Webcam off as no movements are detected by the motion sensor

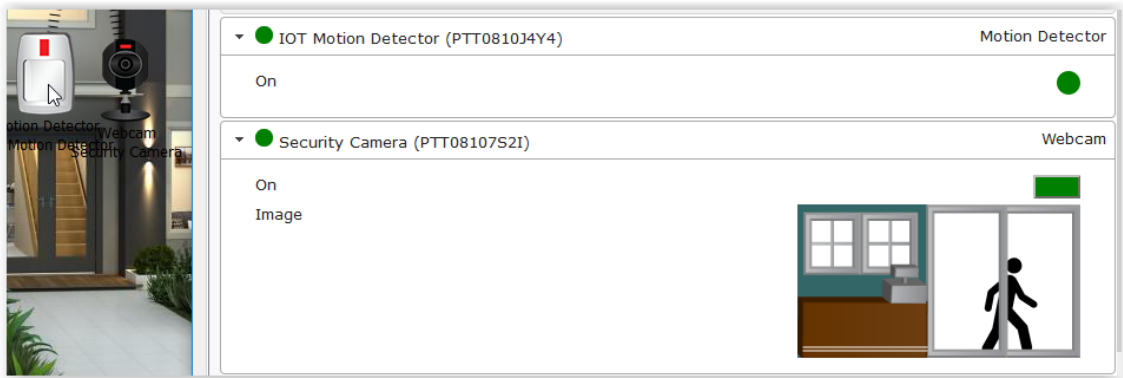


Figure 31 - Webcam on as movements are detected by the motion sensor (ALT + mouse)

As mentioned earlier in the chapter, the smart garden light was also included in the connected devices. The IoT light was an example of a smart device that incorporates own sensors, and due to the complexity of its software, could function even without backend pre-set conditions. The street lamp unit had in fact embedded a motion sensor and a light sensor; these allowed the lamp turn itself on automatically if nearby object or a low level of visible light were detected.

Along with the “SaaS” concept the second big difference, compared to the previous smart-home simulation, was the use of specific environmental variable.

As briefly explained in the chapter 4.2, Cisco Packet Tracer offers variables in order to create more realistic simulations, these in fact can help to influence the behavior of the sensor but also validates the accuracy of the IoT device simulation logic.

In this exercise variables were used in order to influence the production of electricity, the home humidity build up and the sunlight level for triggering the garden lamp.

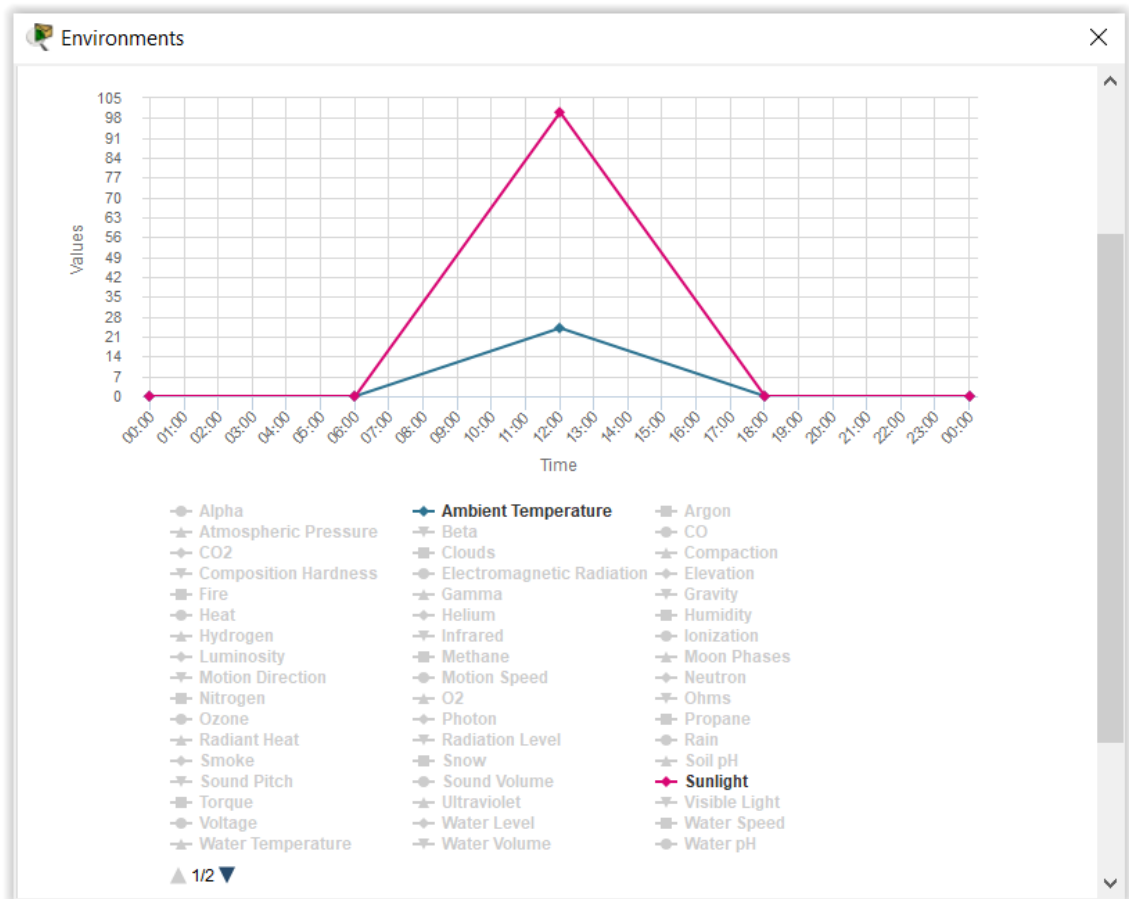


Figure 32 - Environmental variable graph

As illustrated in the above Figure 32 the amount of sunlight, humidity and temperature were tuned for representing a realistic hot sunny day. Behavior of the IoT rules could be influence by adding or modifying these variables.

In Cisco Packet Tracer all the variables were set to follow a 24 hours pattern.

When following the simulation time, reported in the right corner of the main simulator page, it was possible to monitor the different level of produced electricity between night and day, or the level of the humidity, or even the automatic triggering of the garden lamp.

The below Figure 33 represents a day time scenario. It is clearly shown that solar panels produced electricity and charged the battery, the AC unit was on, because of the high humidity, and garden light was off. Figure 34 then represents the night scenario, with AC unit off, no electricity produced and garden light turned on.

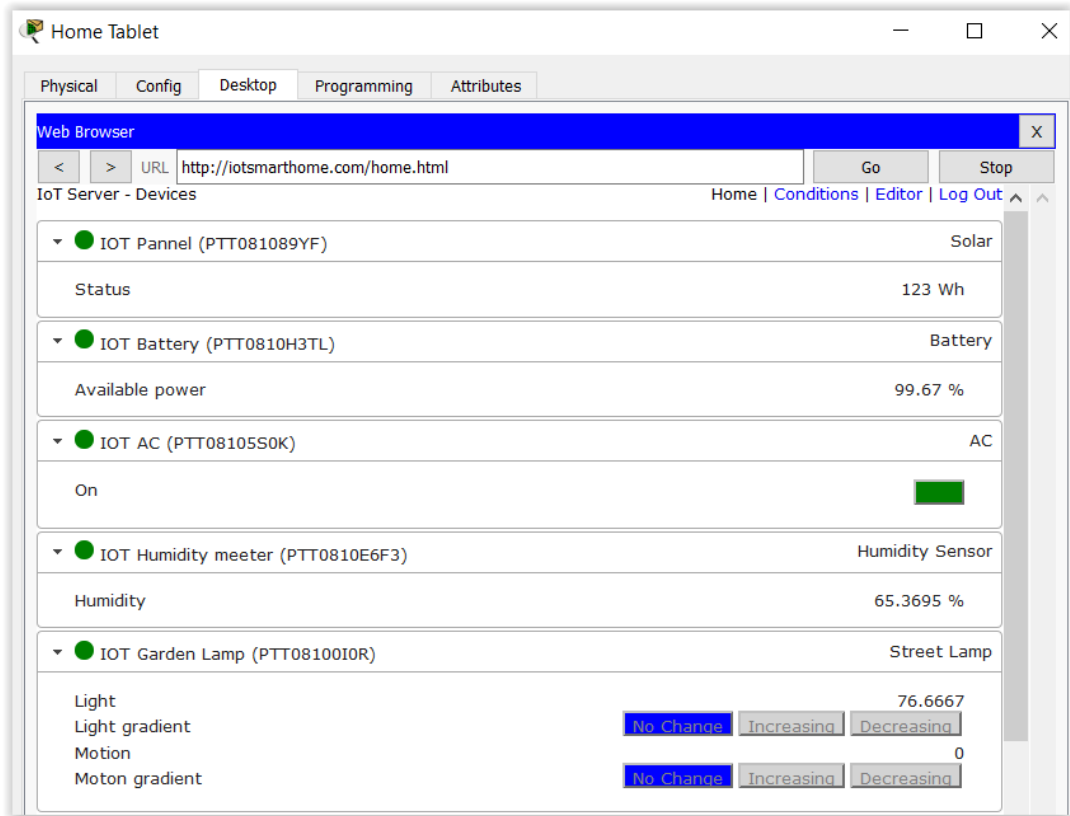


Figure 33 - Example of equipment behavior during daytime

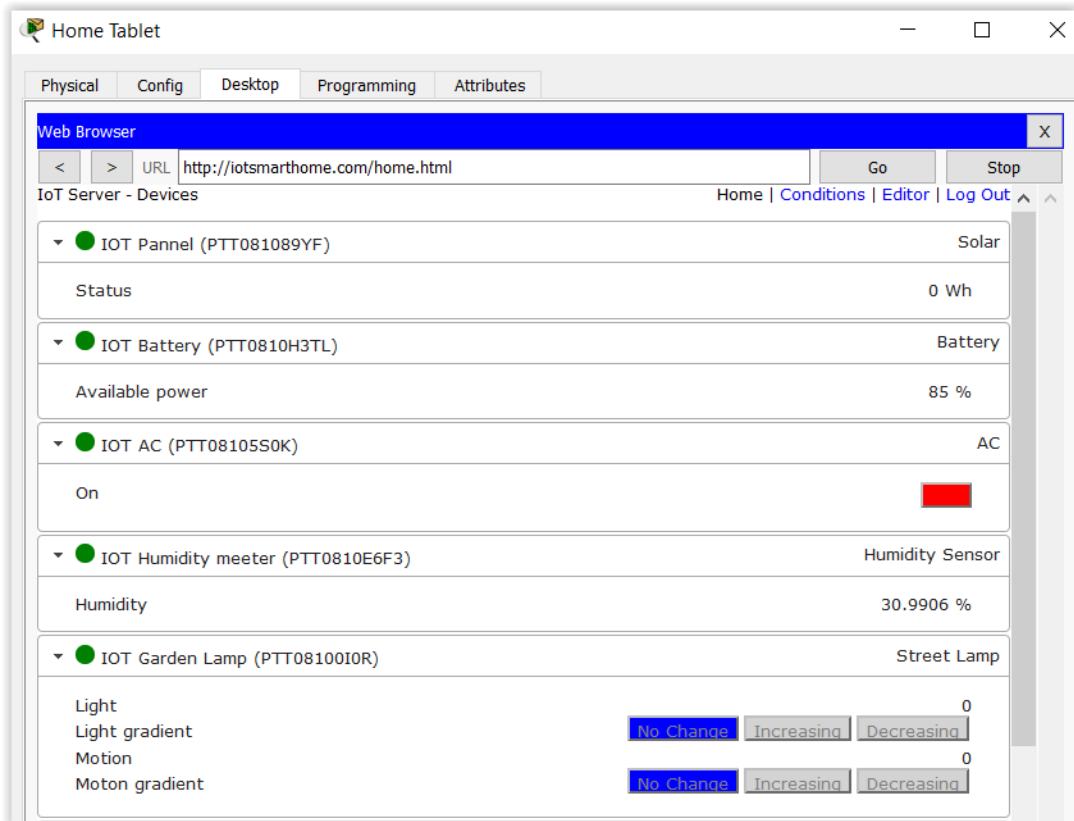


Figure 34 - Example of equipment behavior during night time

IoT Microcontroller Example

One example of IoT sensor-to-actuator was also included in this Cisco Packet Tracer simulation. As in the previous case, an SBC board was used in order to connect a sensor to an actuator without the need of having a network connectivity.

As the Figure 35 illustrates, in this scenario the sensor checked the temperature in the physical container, turning on the ventilation fan if heat exceed a pre-defined value. For this example a non-smart IoT sensor was used, the sensor was then connected to the input pins of the SBC board. An absolute value was send by the sensor from in a scale from 0 to 1023, SBC custom program then detected and mapped the value to a standard Celsius reading comparing it at the same time to the pre-set condition. In case conditions were met the SBC signaled, via output pins, the fan to turn on. An LCD screen was also connected to the SBC output pins to display the temperature level.

Blockly program can be found in the Appendix 1.

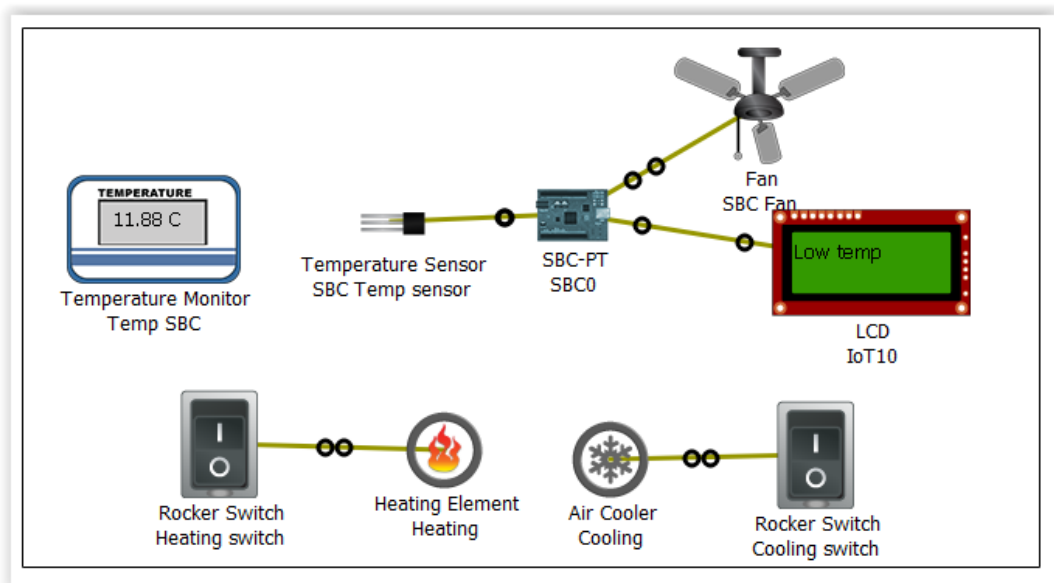


Figure 35 - Setup of the microcontroller example

As explained in the previous chapter 4.2, sensor behavior is directly linked to environment variables, for proving the logic of this simulation heating and cooling elements were added to quickly influence the container ambient temperature.

These elements could be quickly turned on by pressing ALT on the keyboard and clicking on the connected rocker switch.

As demonstrated in the Figure 26 when the heating element was turned on, the temperature raised fast, turning on the ventilation fan.

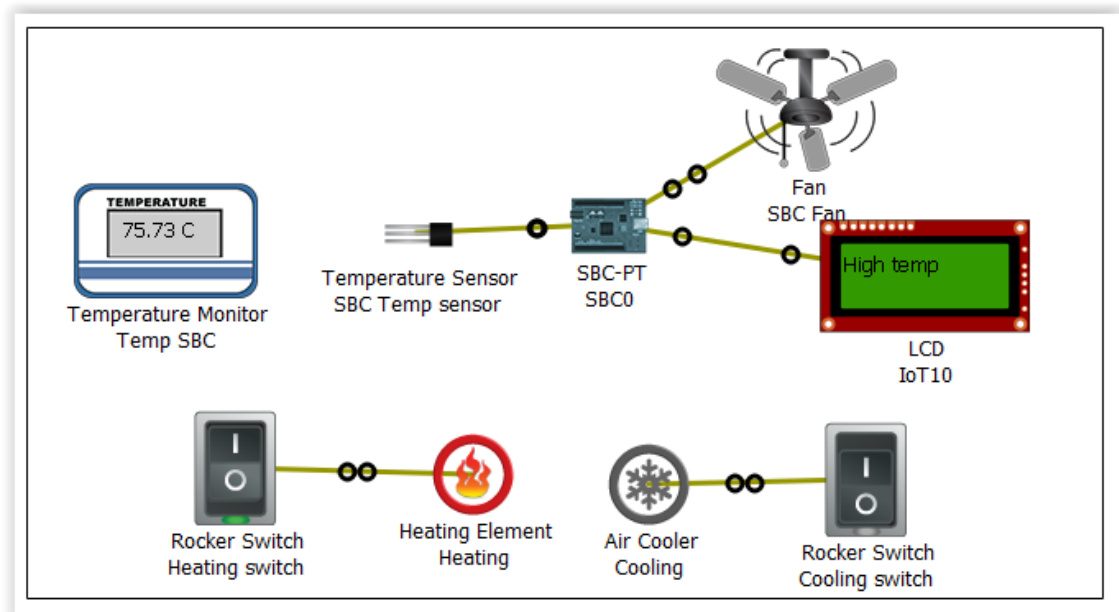


Figure 36 - Example with heating element switched on

Future Expansion of the Simulation

Compared to the previous Smart-home simulation, the network layout in this case was more complex, however it offered many possibility to the students to work on.

On the network prospective the remote cloud IoT network should be strengthened by adding proper connectivity layers such as a router, a set of cluster IOT servers and a security layer etc. This would reflect a more realistic SaaS provider network, redundant and shared between several customers.

On the IOT side the expansion could be done by enhancing the security surveilling system, upgrading it with more complex detectors or more intricate triggering actions.

Home could also be equipped with a network of safety detectors such: smoke detector, carbon monoxide detector, fire detector etc. Actuating extra ventilation, fire sprinklers, or door and windows opening in case hazards would be detected.

For microcontroller simulations, as in the Smart-home 1, there are no boundaries of what can be achieve when programming SBCs.

4.3.3 Smart-Campus

Smart campus was a more comprehensive IoT simulation compared to the previous two smart home exercises, both network and IoT layout, were in fact more complex in order to show a deeper interaction between the IoT devices but also to give more options to the students for future exercise expansion. Smart-Campus simulated an university campus where, along with traditional school and apartment networks, an IoT network allowed to connect different IoT devices spread across the campus premises. Examples of RFID access control management and intelligent sport field watering solution were included in the simulation.

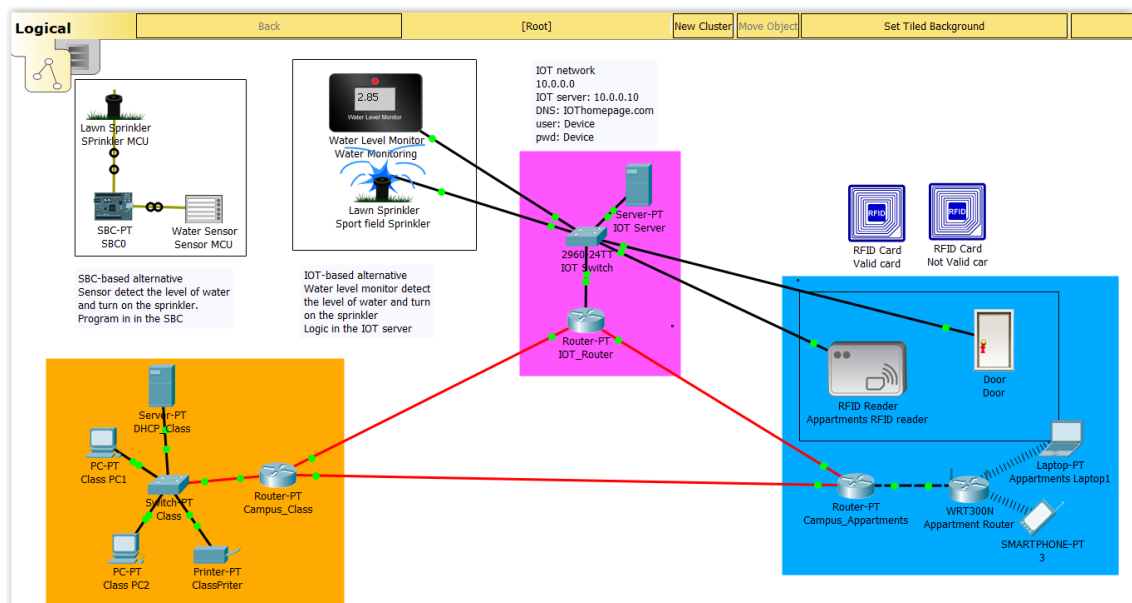


Figure 37 - Cisco Packet Tracer layout of Smart-Campus simulation

Network Layout

As briefly mentioned, the network layout in this exercise was more complex compared to previous simulations and include: a backbone router network, a traditional switch-based classroom wired network, a wireless LAN for the apartment buildings and a dedicated IoT network based also on switch.

The backbone network was created utilizing three interconnected routers. Every router had a connection to the other two in order to build a redundant infrastructure that could withstand failures of trunks between the routers.

In order to represent a realistic network where routers were physically located in different campus building, optical Fast-Ethernet cabling was used instead of traditional straight copper cables.

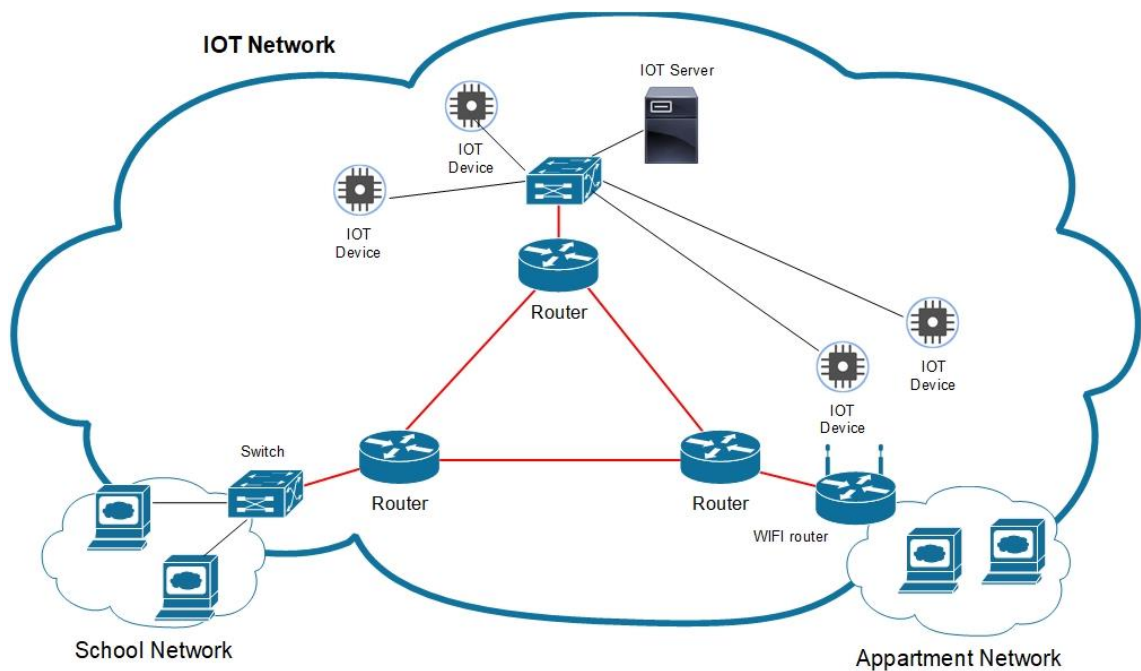


Figure 38 – Smart-Campus network topology

For keeping the routing between the backbone devices simple yet, to allow full connectivity between the three networks, a basic Routing Information Protocol (RIP) was used in the router configuration. RIP is a very simple, and old, routing protocol that periodically shares routing table between devices. In real life complex scenarios the protocol is usually not utilized due to its scalability limitation, as protocol in fact allows only a maximum of fifteen network hops .

However, due to its setup simplicity, RIP was a perfect candidate for routing protocol in Cisco Packet Tracer exercise.

In each router the setup was done adding the directly connected networks IPs in the RIP configuration as shown in the Figure 39, routing table exchange will then take care of spread the routing logic between the device, as illustrated in the Figures 40 and 41.

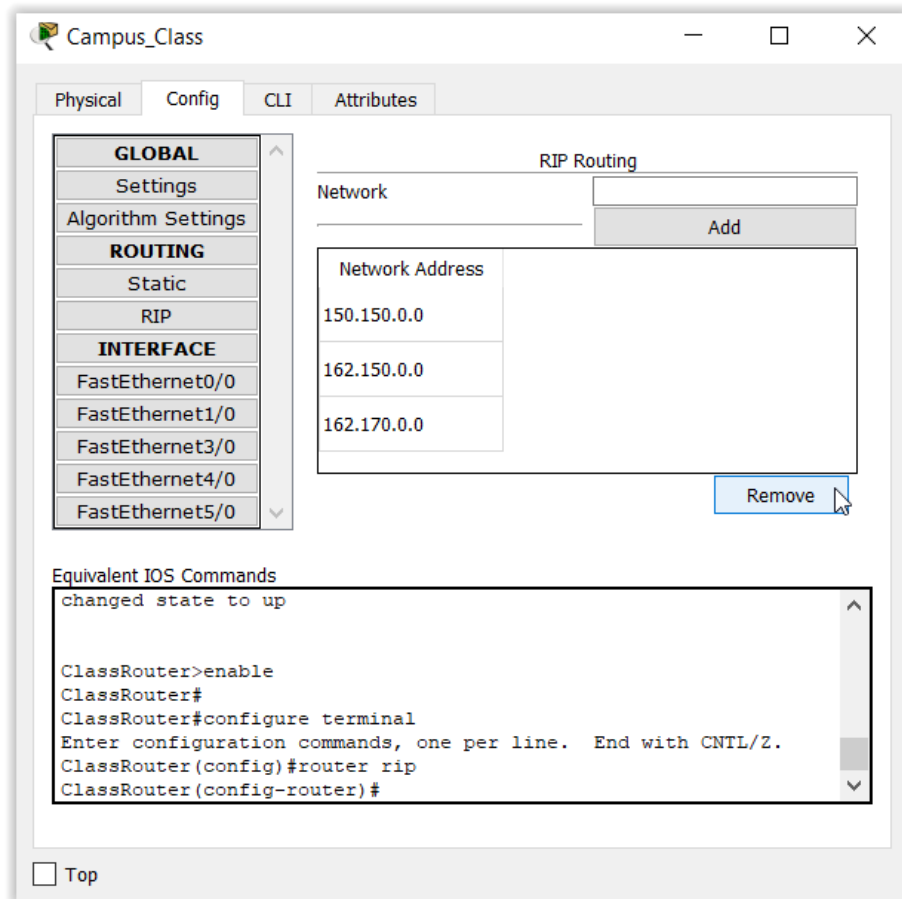


Figure 39 - Example of simple RIP setup of the Classroom router

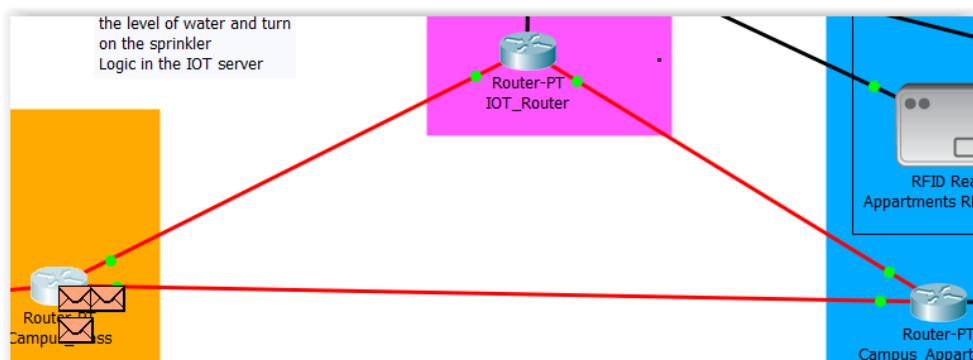


Figure 40 - Example of RIP messages broadcasting by the Classroom router

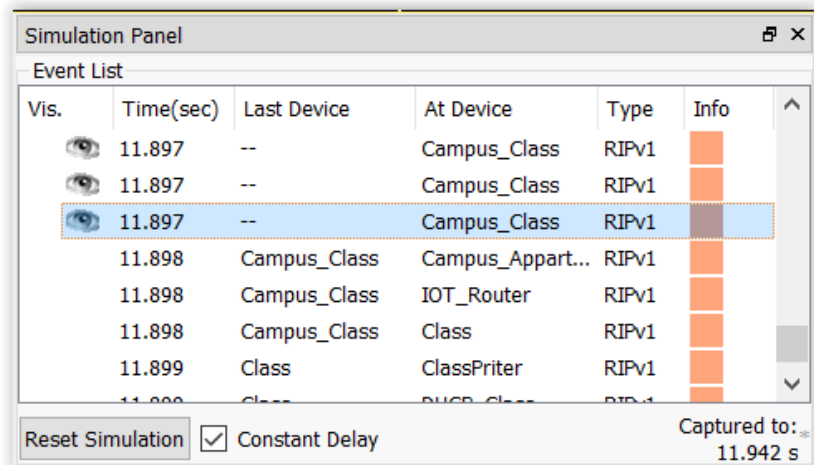


Figure 41 - Packet captured during RIP routing message broadcasting

Along with the backbone connection each router was also connected to one of the three sub-networks: class building network, apartment building network and IoT network. All the three networks were physically separated by placing them in an own dedicated physical container.

The first network was a simple network for emulate a PC classroom where, two PCs, one network printer and a server were connected by Ethernet cables to the classroom switch. Switch was then connected to one of the Ethernet port of the router. DHCP functions were carried on by the local servers.

The apartment building network was also a simple WLAN network that simulated a wireless connectivity in the students apartment buildings. In this case a WLAN router was utilized in order to create the local wireless network, router was then connected with one of the backbone routers. DHCP functions in this network were also carried on by the WLAN router. One laptop and a smartphone were connected to the wireless network.

Last, but most important network, was the IoT network. This was a switch-based network connected to the third backbone router. IoT devices and IoT server were all connected to the same switch. In the original specification of the IoT simulation a WLAN router was supposed to be utilized to connect all the IoT device, making the simulation closer to the reality. However, due to the coverage range of WLAN signal, and the absence of wireless repeaters in Cisco Packet Tracer, the furthest IoT devices had very bad coverage and sometimes connection to the IoT server would timeout and fail. Usage of switch and

copper cables was perhaps not fully realistic and not applicable in real life applications, but suited better for this Cisco Packet Tracer exercise.

The IoT server, in addition to IoT backend intelligence, was also utilized as DNS server and DHCP server allocating IP address to the connected IoT devices.

IoT Layout

As in the previous IoT automations all the smart devices were remotely connected to the IoT server sharing the same username and password credentials. Connection also was established by using the static IP of the IoT server hosted in the same IoT network.

As per design and, due to network layout and utilization of RIP protocol, every PC, laptop and smartphone connected in any of the campus network could remotely access to the IoT homepage URL. The IoT server, being also DNS server, translated the homepage URL into its own static IP.

When connected to the IoT homepage via browser, and after successfully passing the authentication, IoT users were able to see the list of connected devices and the interaction logic between them.

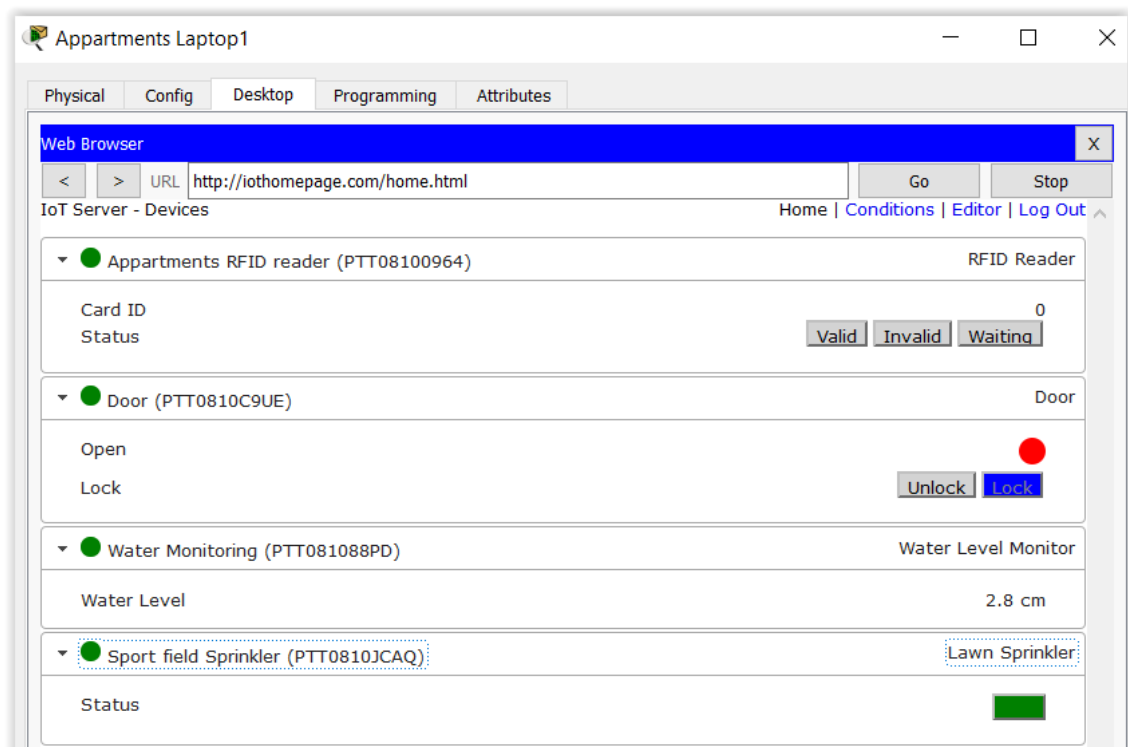


Figure 42 - IoT connected devices in Smart-Campus simulation

As displayed in the above Figure 42, in this Cisco Packet Tracer exercise four IoT devices were connected: RFID reader, apartments door, water monitoring sensor and sport field sprinkler.

Also visible in the Figure 43 is the logic assigned to these four devices

Actions	Enabled	Name	Condition	Actions
Edit Remove	Yes	RFID waiting	Appartments RFID reader Card ID = 0	Set Appartments RFID reader Status to Waiting Set Door Lock to Lock
Edit Remove	Yes	Door unlock	Appartments RFID reader Card ID = 3535	Set Appartments RFID reader Status to Valid Set Door Lock to Unlock
Edit Remove	Yes	Door lock	Match all: <ul style="list-style-type: none"> Appartments RFID reader Card ID != 3535 Appartments RFID reader Card ID != 0 	Set Appartments RFID reader Status to Invalid Set Door Lock to Lock
Edit Remove	Yes	Sprinkler OFF	Water Monitoring Water Level >= 3.0 cm	Set Sport field Sprinkler Status to false
Edit Remove	Yes	Sprinkler ON	Water Monitoring Water Level < 3.0 cm	Set Sport field Sprinkler Status to true

Figure 43 - Pre-set IoT conditions

The first automation was aiming to create an IoT RFID access control solution for managing the access in the students apartments using a RFID reader, couple of RFID cards and a smart door.

The concept behind was very simple, when an authorized RFID card was swiped onto the RFID reader the door opened, but if an unauthorized RFID card was utilized the door remained locked. In order to achieve such scenario the RFID card parameters were modified by editing the card attribute tab as shown in the Figure 44 below.

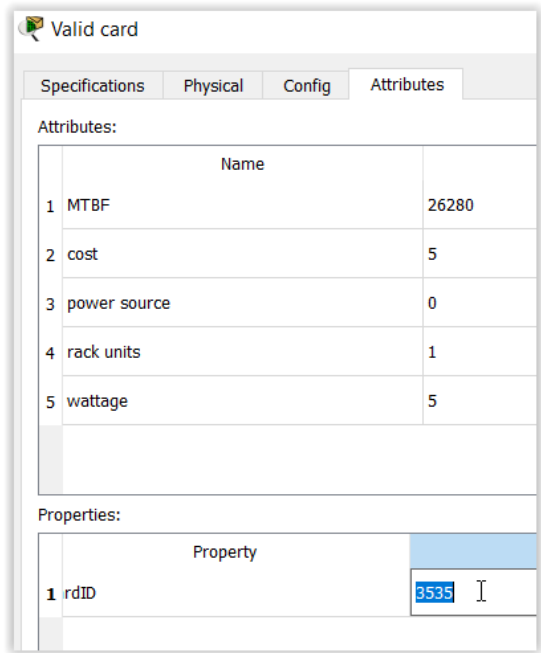


Figure 44 - RFID card attribute modification

In this cases two cards were utilized, one with value 3535 and the other one with 1212. As illustrated in the Figure 43 the authorization logic was done in the IoT backend server, as the condition was set to unlock the door when the 3535 was swapped. Swapping the card was done simply by dragging the card on the RFID reader using the mouse. Green icon appearing in the reader meant that card was authorized, at the same time also door icon turned from red to green. If wrong card, with value 1212 in the example, was utilized, the reader icon stayed red along with the locked door icon as visible in the Figure 46.

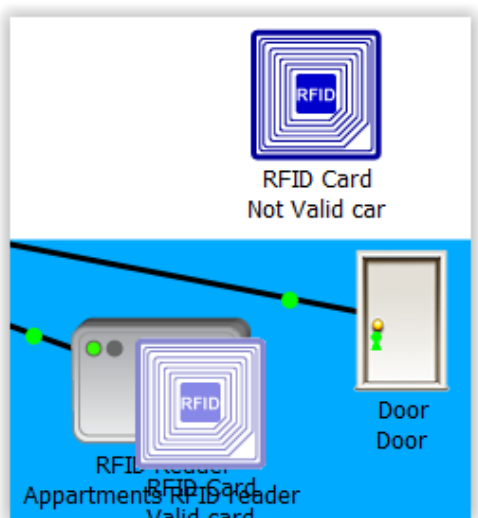


Figure 45 - Valid card swapped on the reader

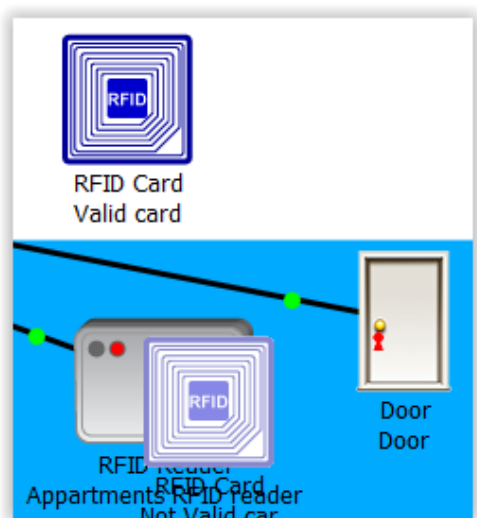


Figure 46 - Invalid card swapped on the reader

For this specific case, and only because of the logic how the reader worked, a third waiting condition was set. Reader in fact stayed in a waiting loop, ready to accept new card, till any card was placed on it. While reader was in this loop the smart-door remained locked.

The specifications of how the reader worked and all the expected status were clearly explained in the device specification tab.

In this scenario the smart-door was utilized as device that reacted to conditions however, due to its versatility, the device could be used in more complex simulations. Direct interaction of the door was also possible by pressing ALT in the keyboard and clicking the door icon.

The RFID simulation was only created with the main purpose to show different IoT scenarios to the students. It was however clear that, in more complex applications, utilizing the IoT backend logic was not the best option, as far different combinations of cards and access level were impossible to achieve with simple conditions.

Another consideration to mention is that, in Cisco Packet Tracer, the RFID reader did not always perform properly. When launching the simulation for the first time, the reader did not accept any card. Students were advised to always stop and restart the reader internal program when launching the exercise for the first time. This was done by clicking on the device, then on the advance button and last on the Programming tab, once there it was possible to stop and start again the program as shown in the Figure 47 below.

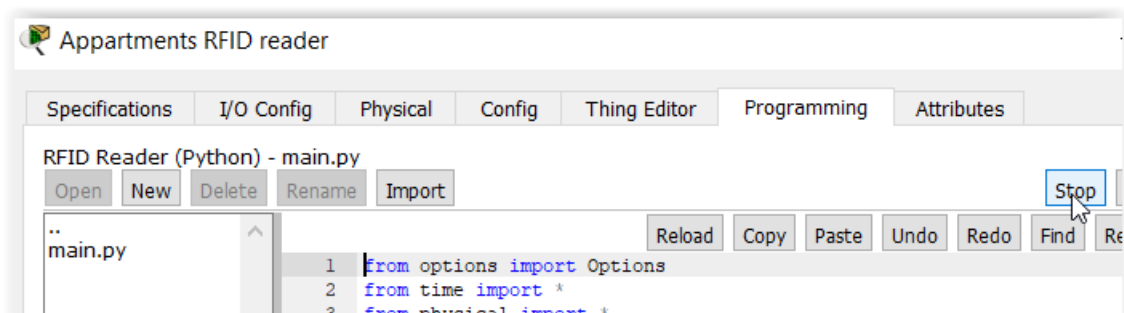


Figure 47 - Stopping a device program

The second IoT case in this exercise simulated a smart sport field irrigation system, where a sensor detected the level of water and, in case level was low, started the water sprinklers. Logic can be seen in Figure 43.

In order to make the automation more realistic the environmental variable of the sport field containers were changed to simulate raining through some hours during the day.

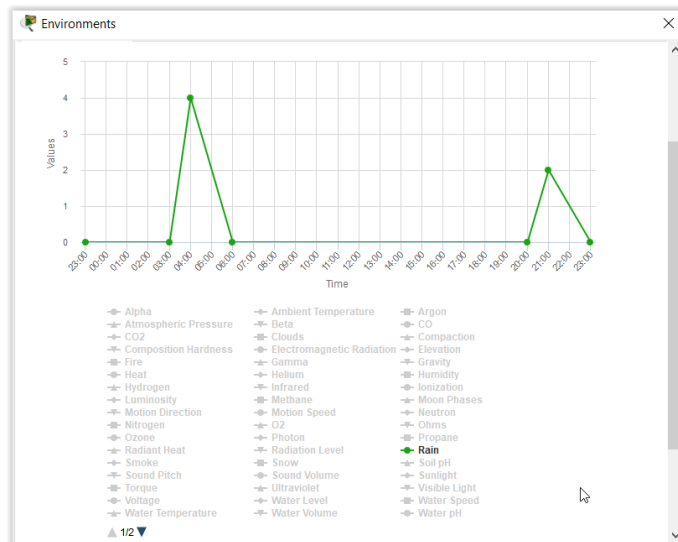


Figure 48 – Environmental variables for Smart-Campus simulation

As visible in the above Figure 48 rainfall were set to be happening between 03.00 and 06.00 and between 20.00 and 23.00 every day. Rainfalls increased the water level in the sport field container, causing the water sensor to detect a higher amount of water and not starting the sprinklers.

The below Figure 49 illustrates the sprinkler action when level of water was below or above three centimeters.

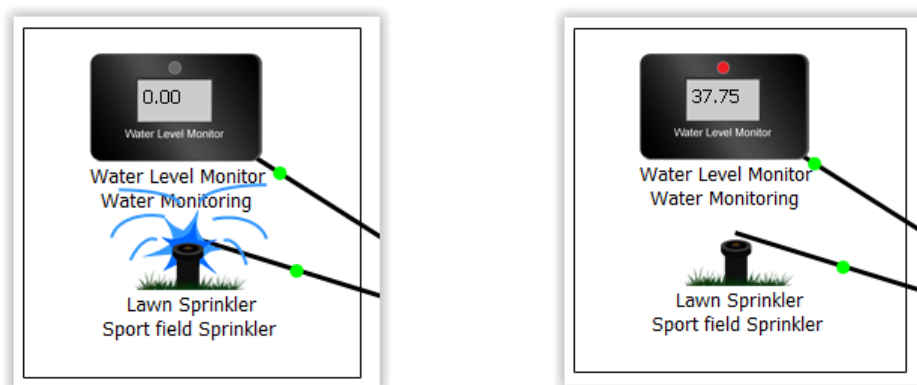


Figure 49 - Smart-sprinkler running and not running

The field water absorption speed could also be boosted by adjusting sunlight and temperature in the environmental variables.

IoT Microcontroller Example

As in the previous two smart-home simulations also a microcontroller example was added in the smart-campus Cisco Packet Tracer exercise. The simulation replicated the same IoT sprinkler case while using non smart devices.

SBC module input pins were connected, via dedicated IoT cable, to a non-smart water sensor. SBC output was then connected to a lawn sprinkler as illustrated in the Figure 50 below.

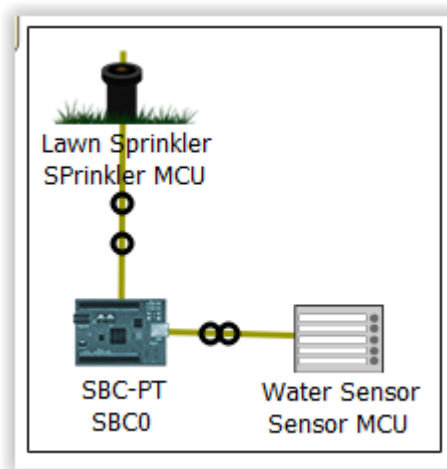


Figure 50 - Setup of the microcontroller example

As in the IoT version of the simulation, the water sensor detected the water level and fire up the sprinkler if a value below the threshold was detected.

Difference compared to the IoT case was that logic was defined not in the IoT backend server but in a custom Blockly program stored in the SBC itself.

The other small difference was that the water sensor utilized returned a number from 0 to 255, this value was then mapped in the SBC program to a range between zero and twenty centimeters.

Also in this case the rain environmental variables have been manipulated in order to rainfall through the day.

Logic of the Blockly program running on the SBC is visible in the Appendix 1.

Future Expansion of the Simulation

Due to the possibilities opened by exercise complexity, students were encouraged to widely modify the simulation adding new network and IoT features.

On network level, by design, all the device were able to access to the remote IoT server. Access restrictions could be apply to make sure that only authorized IPs could connect to it.

Due to wireless range limitations all IoT device were connected via switch, as this would be probably not realistically in real applications, students could re-engineer the IoT network setup in order to create a wider WLAN network or a cluster of wired IoT subnetwork.

On IoT prospective, many more simulations could be added in the campus scenario: electricity generated to power street lights, network of security sensors, evacuation plans to open all doors and windows etc.

Also microcontroller examples could be integrated in the simulation in order to cover cases where the WLAN coverage of the IoT network was not sufficient.

4.3.4 Smart-Industrial

Smart-industrial was the most complex IoT simulation created for the IoT course. Complexity of the case was mostly because the network layout and also due to logical connections between the IoT devices.

The exercise simulated an industrial application where electricity was generated via solar panel and wind turbines, temporarily stored in batteries, and then spent by an industrial production line made by actuators and component. Same electricity was also utilized for power accessory items like cooling units and lights.

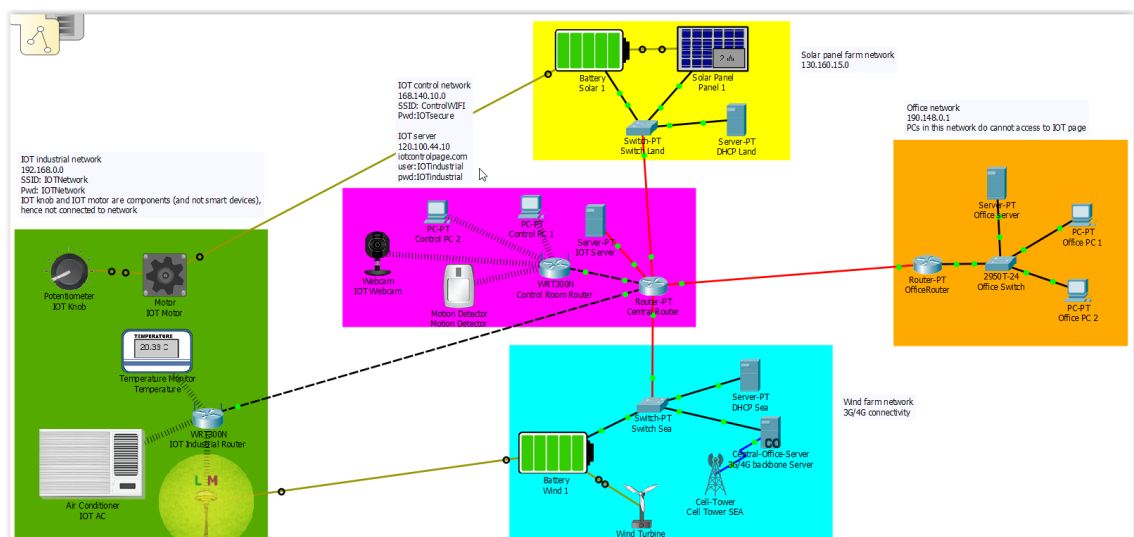


Figure 51 - Cisco Packet Tracer layout of Smart-Industrial simulation

Network Layout

Network layout was more complex compared to the previously analyzed examples, physical separation of the networks, achieved by creating separate physical containers, was required due to the logic of the exercise.

Network topology was divided in five main sub-networks: two where IoT devices producing and storing electricity were connected, one for the corporate office building, one where IoT device that utilized electricity were located and last one for overall IoT control. All these networks were interconnected to each other by a central core router located in the IoT control network. This non-redundant layout might not be suitable for real life industrial applications, however it helped to simplify the Cisco Packet Tracer exercise.

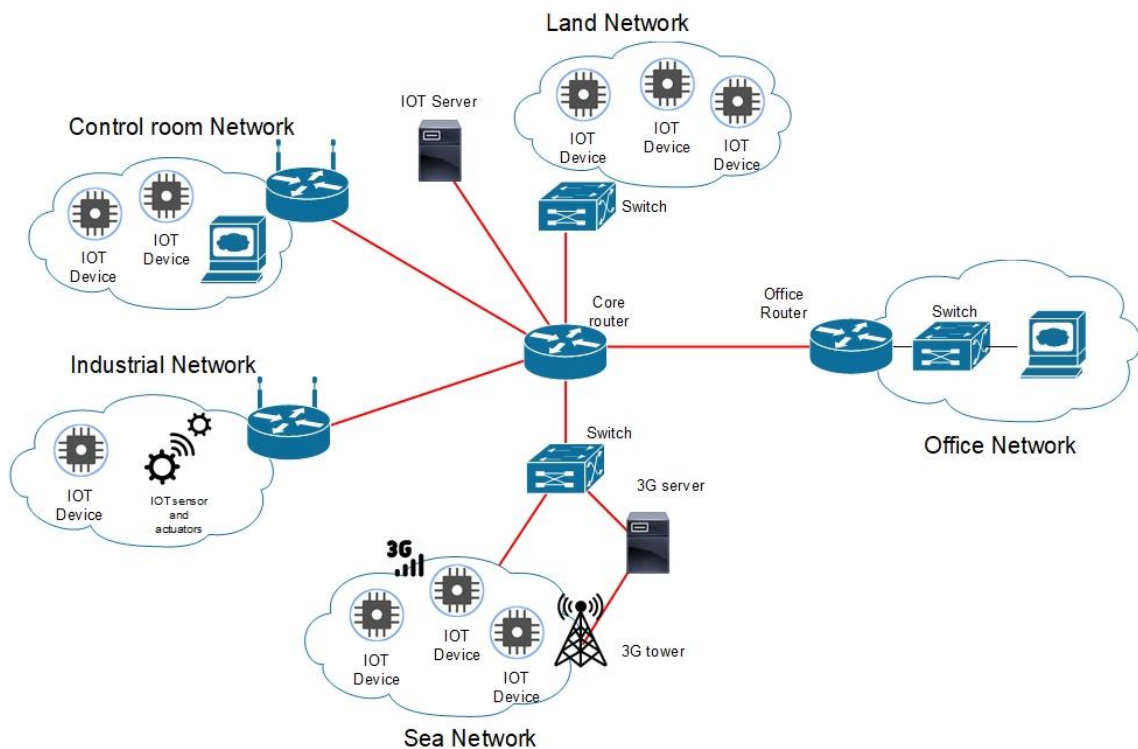


Figure 52 – Smart-Industrial network topology

The simplest network was the corporate office building LAN. Network consisted in a main router connected to the central router and a local office switch. PCs and office DHCP server were also connected to the local switch.

By network design, none of the office PCs were able to reach the IoT homepage or any of the IoT devices. Intention in fact was to isolate and restrict the access to control IoT device for only authorized user physically connected to the IoT control network.

Sea and Land were the two networks where IoT device producing electricity were connected, both of these LANs were connected to the central core router.

Land was a switch-based network utilized to simulate the functioning of a solar panel farm, where electricity was produced and stored in smart IoT batteries.

In order to simplify the design only straight copper cables have been used to connect the IoT devices to the switch. In a real life case optic or wireless technology might be preferred to overcome the communication distance limit dictated by the copper cabling.

Within the same network a DHCP server was installed in order to distribute local IPs to device connected to the land network.

The Sea network was conceptually similar to the land network, in fact, also in this case, IoT wind turbines were producing electricity then stored in batteries. As in the previous network also a traditional switch, connected to the main core router, was utilized, however main connectivity to IoT wind turbines was provided by 3G.

The 3G cellular network was used to differentiate the setup, giving the possibility to students to familiarize with different types of network, but also to give a more realistic aspect of a network over a sea.

As explained in the previous Smart-Home 2 SaaS example, due to the 3G network, additional equipment must be installed. An antenna was required in order to provide, via a predefined APN, connectivity to the turbines. A central-office server component was also necessary in order to consolidate the signal from the cell towers, coming via coaxial cable, to Ethernet. Central-office server was then connected to the land network switch. As in the previous setup also in the sea network a DHCP server was installed in order to propagate IPs to the IoT devices.

The fourth, and less complex, network was the IoT industrial WLAN. This simple network was providing wireless connectivity to the IoT devices that were draining power from the IoT batteries located in the sea and land network. Wireless signal was created by a local WLAN router connected to the central core router.

The last, but most important network, was the IoT control LAN.

Purpose of the networks was to be the main connection point between other WLANs and LAN, but also to conceptually work as main control room for the IoT devices.

All IoT devices were in fact remotely connected to the IoT server hosted in the IoT control LAN. Heart of this grid was the core routers. As device was the central connection point additional NIC cards were needed in order to be able to connect all the other sub-networks.

RIP routing protocol also played a key role in the simulation, enabling the possibility for the IoT remote devices to connect to the IoT Server. As previously explained RIP is a very old but simple routing protocol that, because of its setup simplicity, was chosen to be used in the IoT automations.

Other than the IoT server also a WLAN router was connected to the core router, providing WLAN connection to the control room PCs and to two smart-devices that simulated an intrusion detection system.

IoT Layout

IoT setup was similar to all other three cases, even if devices were in fact connected to different LANs, all of them were logically connected to the IoT server hosted in the control room LAN.

As before, all IoT devices shared the same username and password credentials. Due to network design in this Cisco Packet Tracer exercise, only the PCs connected to the control room LAN could access to the IoT homepage.

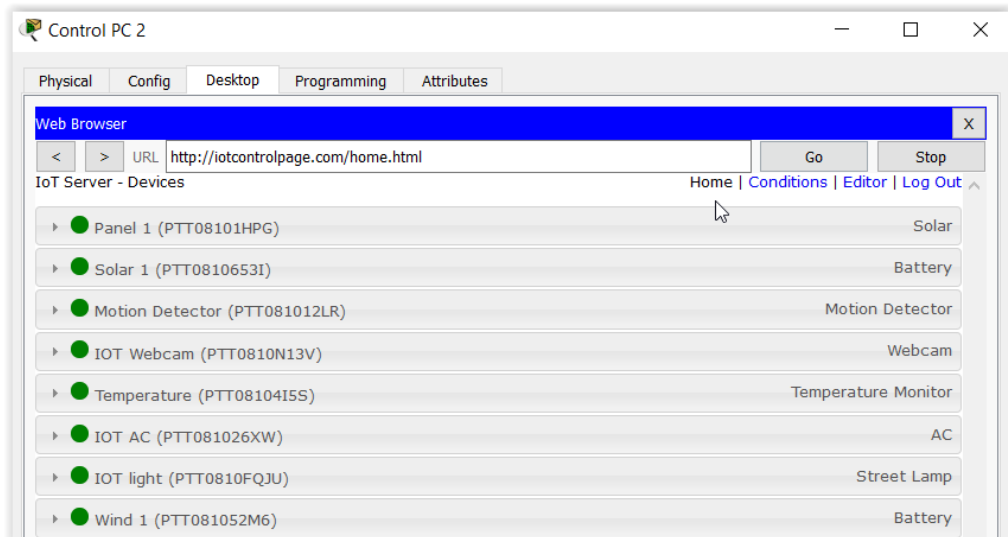


Figure 53 - IoT connected device homepage

As illustrated in the Figure 53, Smart-industrial exercise had several IoT devices already configured such as: solar panel, wind turbines, batteries, motion detector, webcam, AC unit, temperature sensor and an IoT smart-light.

For the solar panels, wind turbines and batteries, located in the land and sea network, no backend intelligence was required as only an IoT cable was needed in order to connect the power generators to the batteries.

Clear instructions how to connect IoT device input and output slots were listed in the device specification tab. As visible in the Figure 54 the output port of the solar panel is D0 and the input of the battery is D0, cable was then connected accordingly.

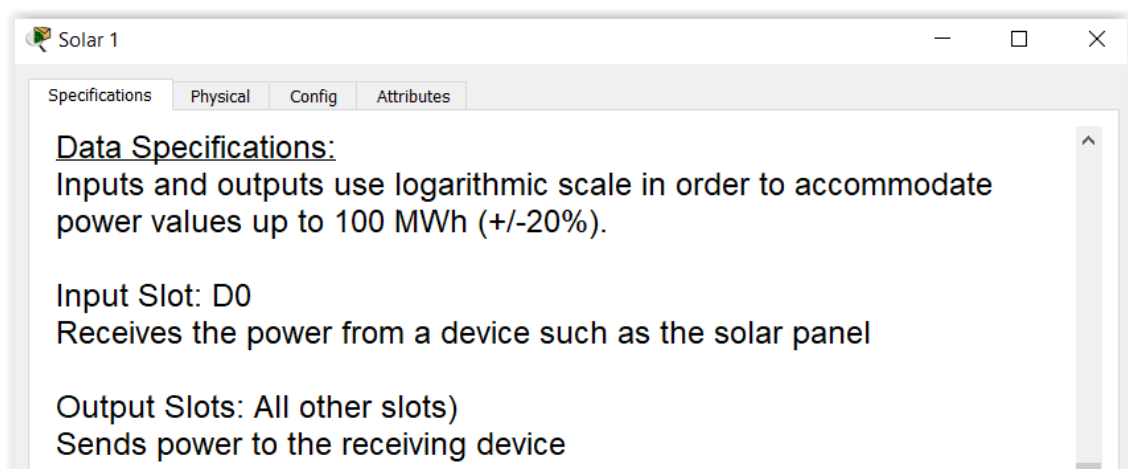


Figure 54 - Example of battery specifications

As quickly mentioned, for this scenario, no backend logic is implemented, however from the main IoT page was still possible to monitor in real time the amount of power generated by the units and the amount of electricity stored by the batteries.

For future expansion additional logic conditions can be added in order to raise alarms if the battery stored power would drop below a certain threshold. Also a power meter could be installed between the turbine and the batteries in order to visually show the amount of electricity produced.

As solar and wind electricity generation was directly proportioned to the solar rays and wind gusts, the relative environmental variables were modified.

As visible in the Figure 55 below, two separate containers have been created and variables have been manipulated in order to generate wind and sun during the day. Sunlight variable does not necessary need to be changed as by default over the daytime hours the simulator will produce sunlight.

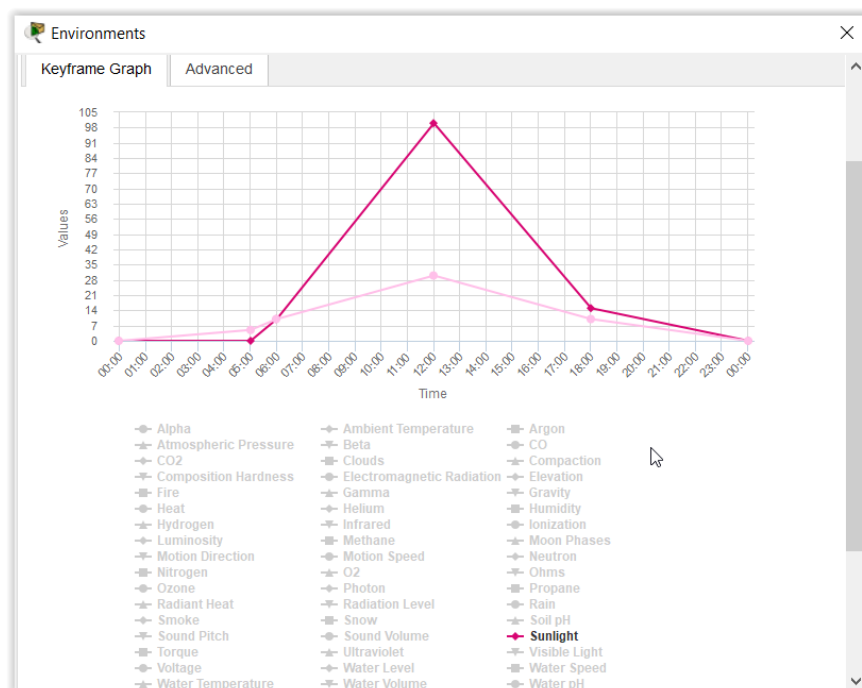


Figure 55 - Environmental variables

As demonstrated in the Figure 56 below, while observing the environment clock, it was very evident to see the peak time of solar electricity production.

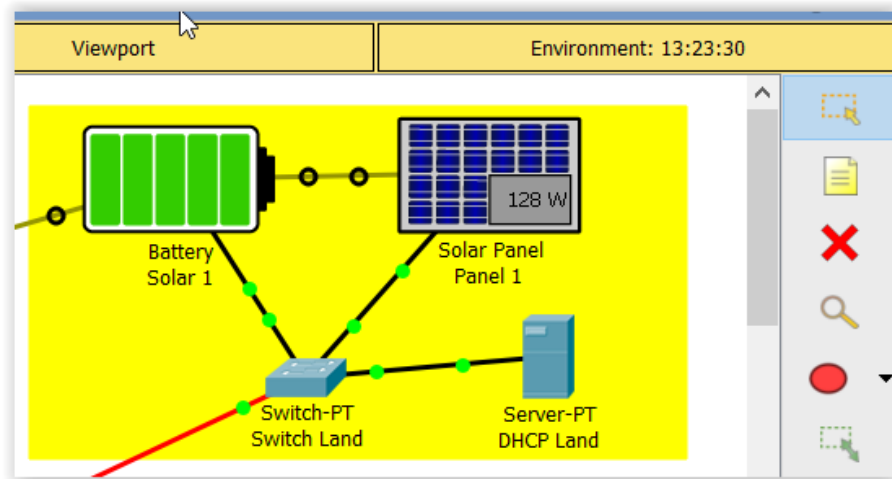


Figure 56 - Electricity production peaking during the daytime

It was also observed that, even without any device connected to the battery draining its power, there was a constant battery discharge happening by default.

As previously mentioned the IoT server was hosted in the IoT control network. Server, as in the other IoT simulations, used static IPs also acted as DNS server translating the `iotcontrolpage.com` address in its own IP address.

Along with the server also an IoT surveillance solution example was connected to the IoT control network. This case simulated a security system, running in the power plant control room that, when motion was detected, automatically started a webcam broadcast for few second.

This IoT automation could be triggered via pressing ALT on the keyboard and hovering the mouse on the motion detector. While connected to `iotcontrolpage.com` via browser it was also possible to see the changing of the webcam icon showing some images frame. Once no movements were detected the webcam automatically stopped.

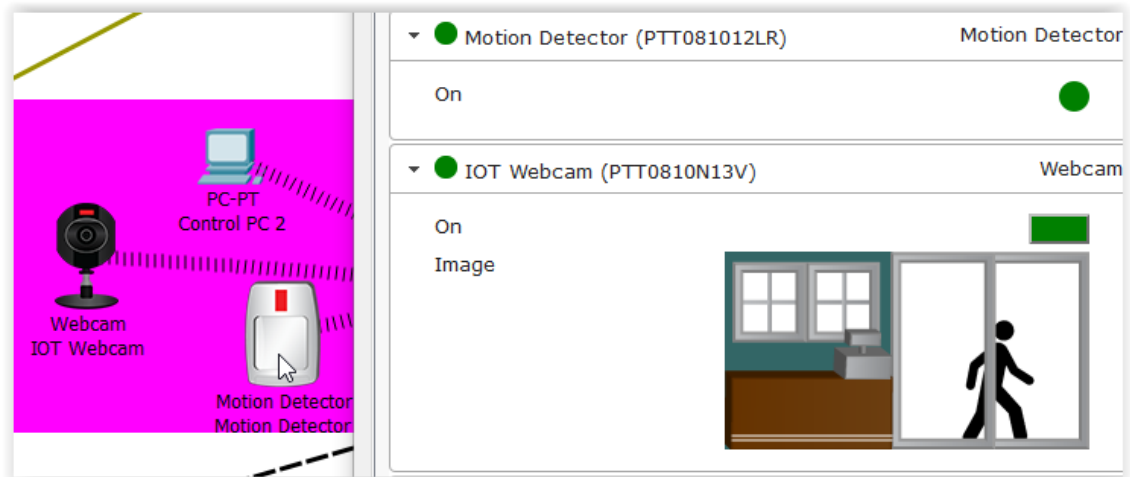


Figure 57 - Example of webcam broadcasting when motion detector is triggered

The other IoT simulation samples were for devices connected to the IoT industrial network such as: an intelligent lamp, and automatic AC unit with a thermometer, a motor and potentiometer.

As explained earlier the intelligent lamp did not required any backend intelligent being a very smart device on its own, unit had in fact embedded a motion and a light sensor, making the lamp itself able to decide to lid in case of object are detected or if it is late in the day and no sunlight is visible. The lamp was included in the Cisco Packet Tracer example in order to show also the possibility to connect devices to an IoT battery in order to drain its power. As illustrate in the Figure 58 below a special IoT cable was used to connect the two units, port specifications were available in the device specification tab.

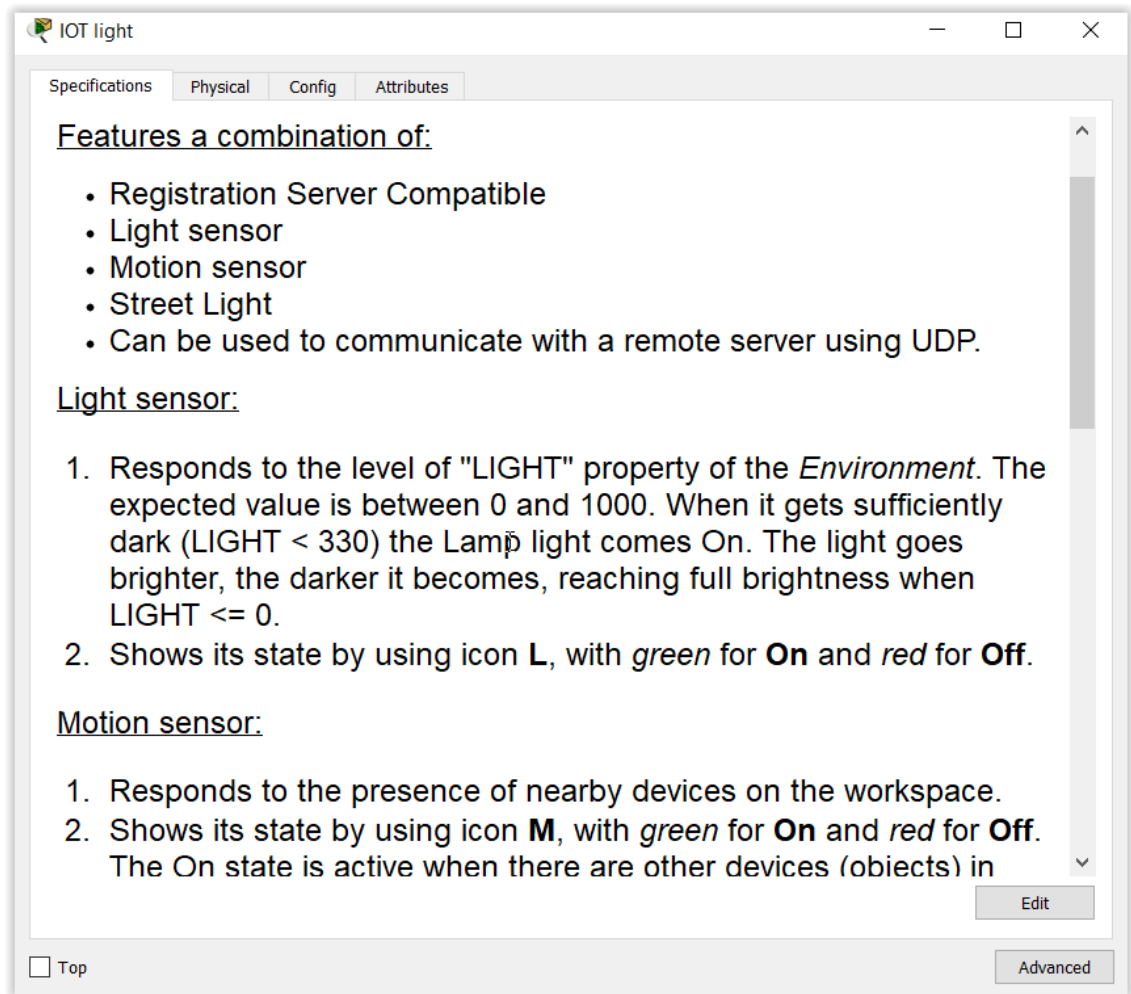


Figure 58 - Example of complex specifications for the smart-lamp

The other standard IoT simulation in this example, that required backend logic, automated a machine room temperature monitoring system. For this purpose a smart temperature meter and an AC unit were connected to the IoT industrial WLAN. Additional backend logic was set to automatically start the AC unit when a temperature below 35 degrees was detected. By design, and in order to make students practice with them, no environmental variables of the engine room physical container were modified, this resulted in the temperature being constantly under the 35 degrees threshold, hence the AC did not start. It is advisable that students will manipulate the variables in order to validate the logic of the rule.

Temperature and AC status could be monitored by connecting to the IoT homepage.

▼ ● Temperature (PTT08104I5S)	Temperature Monitor
Temperature	14.3 °C
▼ ● IOT AC (PTT081026XW)	AC
On	

Figure 59 - AC and temperature monitoring from IoT homepage

The last IoT simulation in the exercise was the only non-smart device example, without using microcontrollers, provided in any of the four exercise.

This case was utilized to show to the students that even without microcontrollers, or IoT backend connection, it is possible to utilize IoT actuators and sensors, however with very limited capabilities.

In this case a knob potentiometer was connected to an industrial motor, motor was then draining power of one IoT battery from the land network. As in previous applications a special IoT cable was required between the units.

Interaction with the motor was also possible by pressing ALT on the keyboard and moving the knob clockwise with the mouse. Motor rotations increased by rotating the knob. However, as far as the knob and the motor were not connected to the backend IoT server, it was not possible to observe or control them from the iotcontrolpage.com page.

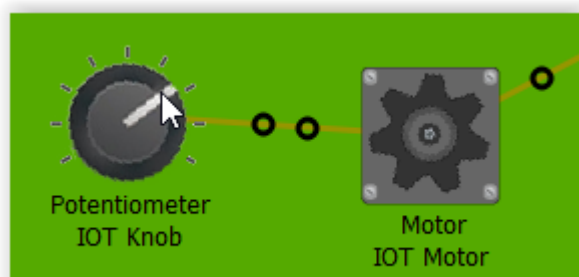


Figure 60 - Knob rotation impact the motor speed

Cisco Packet Tracer offered also the possibility to transform sensors and actuators in smart devices by double-clicking on the device, selecting the advance button, adding then a wireless card in the I/O config tab and then setting up SSID and password in the config tab.

Once done the devices were visible in the main IoT page, however, as the control APIs were missing there was no possibility to check the status or setup further functioning conditions.

IoT Microcontroller Example

As in all other three Cisco Packet Tracer exercises, also in Smart-industrial case an example of non-IoT connected devices using a microcontroller was provided.

As illustrated in the Figure 61 the case reproduced the similar IoT scenario, hosted in the land network, where a solar panel produced electricity and a battery stored the power. Difference was that devices were directly connected to a SBC unit feeding data to the board.

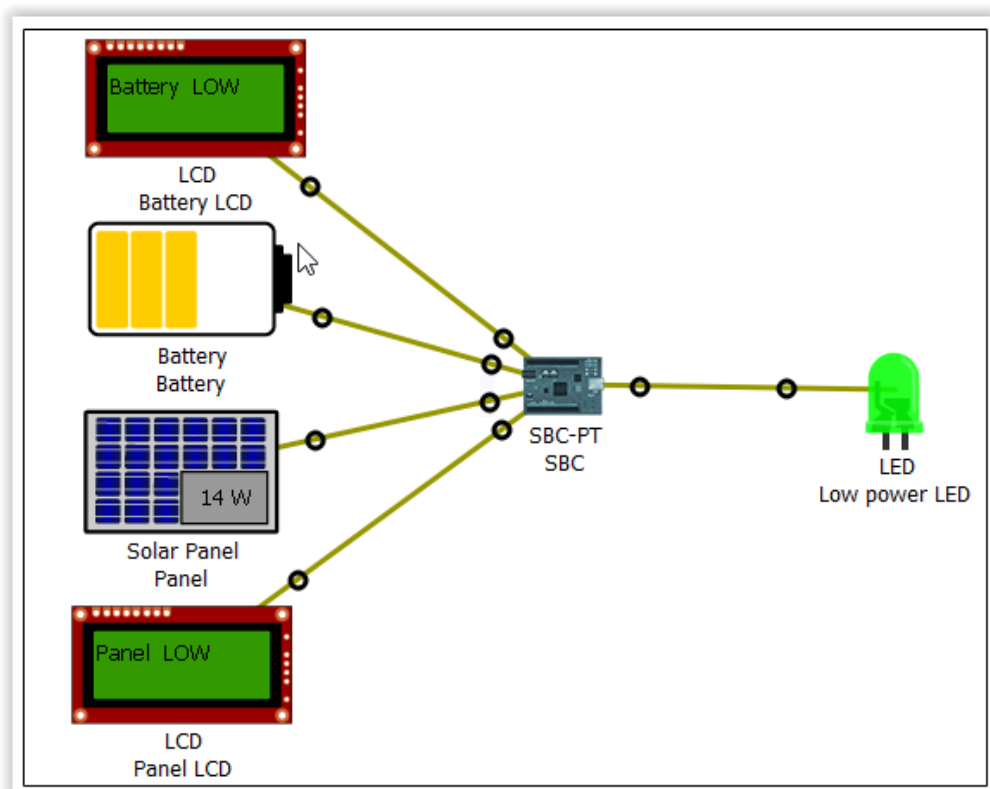


Figure 61 - Setup of the microcontroller example

Via a Blockly custom program the SBC constantly read the input coming from the panel and the battery prompting, in a two separate LCD screens, a small text about the status of the energy produced and stored. In case both values were low an alarm led was lid,

along with prompting to the LCDs the message battery low and panel low, as shown in the above Figure 61.

Complete logic of the Blockly program can be found in the Appendix 1.

Future Expansion of the Simulation

Smart-industrial was the most complete IoT simulation of the four, yet the exercise that offered more possibilities to for students to expand it, both on network and IoT level.

On the network side different levels of access control could be setup, allowing also some of the PC in the office network to access the IoT webpage, but also creating cellular networks for IoT devices in the sea and land networks.

More scenarios could be also added to the IoT simulations, for instance a more complex alarm system might be needed in such industrial cases. This could be achieved using sirens, tripping wires, motion detector and other IoT components.

On the industrial network also a more complex iteration between smart and non-smart devices could be engineered to simulate a proper production line chain. Also a more complex monitoring solution for the electricity production could be a good example for future student's exercises.

4.4 Limitations and Expansions of the IoT Simulations

The four IoT simulations offered a solid foundation for the students to experience the Internet of Things without having the access to physical IoT devices or IoT network. Exercises also offered the possibility to expand the simulation even further. However, due to limitation of the tool and due to time-constraint, some aspect of the exercise were simplified or artificially created.

In all the Packet Tracer examples where an ISP was required, there was no real ISP component in the tool, hence the network was made up using routers and static IPs. In the second smart home exercise, where backend capabilities were provided “as-a-service”, the home router was also artificially setup to use default gateway IP of the IoT cloud server. This would not have been possible in a real case.

The other examples of artificial setups were in the Smart-Campus and Smart-Industrial, where some IoT device were in fact connected via a switch using normal straight copper cables. This setup would not be practical in real scenarios due to the distance limitation of the LAN connectivity over copper.

The other limitations, or one could call them as bugs, are due to general behavior of Cisco Packet Tracer tool itself. Even though the tool seemed to be more stable compared to the earlier version, it tended to crash or not functioning properly every now and then. When making the simulations, but also during the class, the network components were not always correctly communicating between each other even if the setup was correctly made.

Other problematic cases happened when the IoT Smart-device configuration was changed few times, in some of these cases the IoT device did not connect to the backend IoT server anymore.

In all of these examples the suggestion given to the students was to erase the components and restart the configuration by adding new device. This in most of the cases solved the issues.

Other known issues were related to the malfunctioning of the custom program running on the IoT smart devices. The program seems to run, but logic did not work, as happened with the RFID reader program.

The students were advised to stop and restart manually the application in the device programming tab, as shown in the below figure, every time the Cisco Packet Tracer was launched the first time.

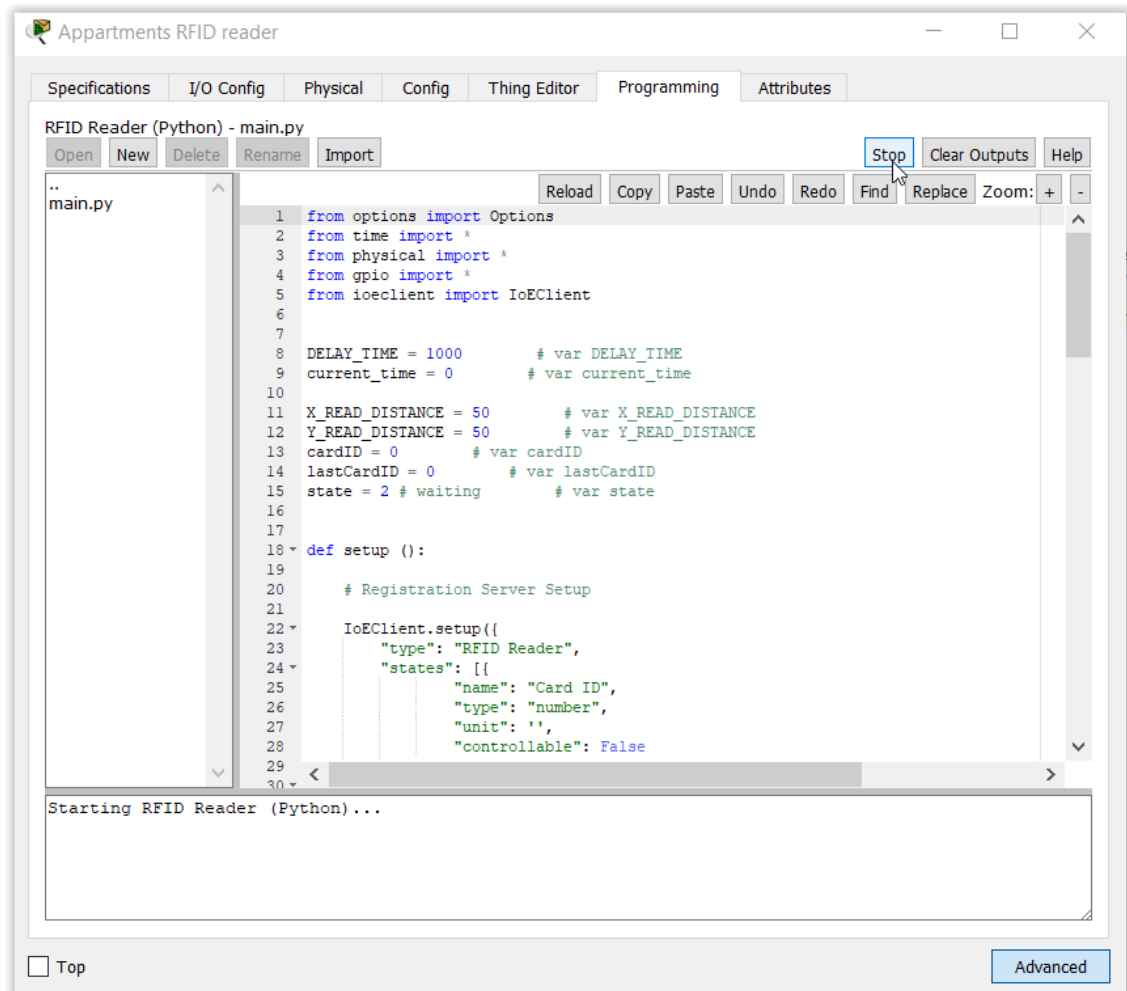


Figure 62 - Example of custom Python program running on RFID reader device

Another limitation observed during the course, not related to Cisco Packet Tracer but more on the structure of the classes, was related to the limited amount of time reserved for the IoT practical part in the study course.

Being the first time this course had been taught, and the fact that the structure of the classes had been decided before a deeper analysis of Cisco Packet Tracer complexity was completed, it resulted in underestimating the time necessary to be reserved for the practical classes.

Other smaller limitations came from the fact that all students should have created their own Cisco NetAcad account beforehand and also have the Cisco Packet Tracer installed in their own laptop.

As mentioned earlier, despite the limitations caused by the tool or the time, the simulations had been built beforehand allowing students, with different skillset, to adapt their own IoT business case or even further expand them, both on network and IoT level.

As previously listed in the chapter 4.3 future expansion of the simulations are possible.

Regarding the network area students could, in the two Smart-home cases, upgrade the VPN connectivity in order to be able to remotely access the home LAN when connecting from an external network. Another example is to add a firewall in order to limit the access to the IoT network from the remote office LAN in the Smart-industrial case. In Smart-Campus the IoT network could be re-organized by using a cellular infrastructure.

IoT smart-devices expansion should be also very simple for the student, purpose would be to add more devices within the IoT network and create more intelligent rules via the IoT backend server. In some of the case this would require also that IoT network would be expanded in order to be capable in running more devices.

Last and more technical expansion could be the usage of microcontroller. Multi-Chip Units (MCU) or Single-Board Computer (SBC) can be widely used in the simulations in order to fully control the IoT device functions. IoT server backend intelligence, programmable via browser, it is only possible for smart-devices and only allows simple IF-THEN-ELSE rules. Programming of microcontroller instead offers full visibility on sensor parameters, actuator functions, data logging and also allows to use a wider range of IoT devices.

Microcontrollers should also be connected to the local IoT network.

Students with stronger programming skills, were encouraged not only to program MCU/SBC board but also modify or create the pre-set program of the sensor and actuators, giving full freedom and control of the IoT device.

5 Feedback and Recommendations

After the completion of the two IoT practical classes a feedback form was distributed to the students in order to gather comments, advices and suggestion for improvements regarding the practical Cisco Packet Tracer exercises.

The feedback form included a small section regarding overall IoT topics, a dedicated part on the Cisco Packet Tracer tool and one chapter regarding the exercise structures and suggestion on how to utilize them in the next IoT courses.

A complete template of the feedback from is available in the Appendix 3.

A complete feedback on the IoT course, classes and exam was not part of this study as it was shared, as a standard methodology, by the IoT course lecturer.

The overall response by the students was limited as only seven forms were returned. However, all of them, were quite aligned showing a great interest in the IoT topics, giving a good feedback on the exercises methodology and to the Cisco Packet Tracer tool.

The below sections describes the students feedback, followed by comments from the IoT course lecturer the thesis writer.

5.1. Students Feedbacks

On the generic questions regarding overall IoT capabilities, maturity levels and area of interests, all the students were quite aligned by feeling that IoT is definitely a technology that will revolutionize the IT industry, both for commercial and private market.

The students commented that migration of IoT is also inevitable as most of the current networks will be eventually connected to some sort of IoT network, opening up multiple possibilities for industries. Another good point raised was that the overall IoT technology requires a different skillset in the development and support workforce, challenging the more rigid traditional IT structures.

Connecting many IoT devices will also increase the amount of data, as commented by one student, this will introduce new possibilities and needs for industries to be able to manipulate and create monetary values from it.

On the technology maturity question, few comments were related to the general feeling that the security in IoT is a big gap not yet standardized or even clarified, opening up also a new set of threats for the IT industry.

One student also manifested concerns on the IoT wireless technology, not yet fully available, but also the fact that IoT devices still haven't reach an optimal maturity in power consumption management.

While answering the question on IoT area of interest, students manifested a very diverse list of topics that they would like to focus on: the security of the devices, home monitoring systems, medical applications and automotive automations. These areas could also be taken into considerations for the next course implementations.

Second part of the feedback form was focusing on the Cisco Packet Tracer tool and on the IoT exercises. Students were quite satisfied with the tool as five over seven graded the tool from good to excellent, two students graded it ok.

Advantages identified by students, while utilizing the tool, were the ease of how simulation can be setup, the fact that the tool is free and easy to download, and the fact that Cisco Packet Tracer offers a big range of IoT devices and functionalities to work with.

Students also seemed to like the fact that the tool provides the possibility to configure the network devices via command line as it would be in real life applications, but also that it offered functions to ease the network setup, such as the automatic selection of best suitable cable or a simpler user interface for the device configuration.

One student openly commented that Cisco Packet Tracer is a very good tool for learning IoT.

On the tool limitations students highlighted the fact that the tool was often unstable, requiring the need to frequently save the project in order to recover from application crashes. One student, working closely to the networking industry, also commented that Cisco network appliances provided in the tool are also old and not updated.

Last limitation, mentioned by two students, was the fact that some scenarios were impossible to do while using Cisco Packet Tracer.

Last part of the form was related to the IoT simulations and the practical classes.

All students expressed very good feedbacks on the exercises, mentioning that it was a very “compact” way to explain practically how IoT works and giving good comments on the methodology of introducing pre-configured IoT simulation.

Positive comments were also received for the support given to them during the practical classes.

Highlights mentioned were the clarity of the examples on how devices worked, the visualization process of the exercises and the fact that simulations were easy yet challenging to use.

Only one negative feedback was raised by one student regarding the limited time and the deliverables that were achieved within the two classes.

Same feeling was also shared in couple open comments regarding possible future enhancement for the IoT course. It seemed clear that more time is required and a more step by step guide on how to setup the exercises is needed.

5.2. Feedbacks and Suggestions for Future IoT Courses

This section collects a series of thesis writer's feedbacks and observations, integrated with comments from the IoT course lectured, on the Cisco Packet Tracer exercises. Also recommendations for future study course implementation are shared.

More generic conclusion on the IoT thesis work are reported in the conclusion chapter 6.

Adding practical classes in the Internet of Things study course was undeniably necessary and a great opportunity for students to familiarize with the topic. After acquainting with the tool, creating the exercises and seeing the students utilizing it, one can conclude that Cisco Packet Tracer is a good simulator tool for such practical cases.

One strength of the tool is the fact that the network aspect is very well designed, proved to be reliable and easy to use. All students worldwide attending Cisco courses are utilizing Cisco Packet Tracer for practical network exercises, hence amount of users and various release of the tool has made Cisco Packet Tracer a perfect networking simulator for learners.

IoT functionalities, as explained in chapter 2, have been only recently introduced and future development are needed, also considering that Cisco will offer more and more IoT courses over the years.

Even if Cisco Packet Tracer offers a good overview of IoT devices, enables a quick connection and setup of them and also allows to intuitively troubleshoot it evident that tool still have issues.

The tool itself seems to be quite unstable, often crashing and without any possibility of autosaving or recovery of the exercises. A tip shared since the beginning with the students was to save often the projects.

Some bug are also visible in the tool regarding IoT devices. More complex devices that are utilizing pre-defined applications seems not to work well when the Cisco Packet Tracer simulation is launched first. One example is the RFID reader in the Smart-Campus exercise. It happened in fact that, when the simulation was first open, the reader was not accepting properly any correct or incorrect card. Stopping and restarting the custom program fixed the issue.

Another bug, happening both with IoT and normal devices, was clearly visible when a configuration was changed multiple time within a short time. This causes the device to not respond properly, not to get connected to the network or not to act as it was programmed.

Two examples emerged during the practical classes when an IoT server did not connect logically to the router even if configuration was correct, or when some students struggled for long time to connect some IoT devices to the WLAN even if the setup was correct.

The tip shared to the students was that, if troubleshooting of the issue would take more than few minutes, it would have been better to erase the device and re-add it to the simulation from scratch, this solved almost every time the issue. Other option was also to close and restart the entire Cisco Packet Tracer.

Another limitation identified when creating the exercise was the obvious fact that not all that is possible in real life is possible in Cisco Packet Tracer. There were in fact multiple limitation for example when setting up ISPs, or when expanding network device ports but also more in general on what the IoT smart-device could do.

Limitations were usually overcome by creating workaround or clearly explain them to the students during the classes.

Regarding the methodology of the exercise the main problem was related to the time spent for the practical classes. As explained in the chapter 3, during the initial conversation with the IoT course lecturer, as this was the first time that course was taught, the time to be allocated for the practical classes was not clear.

It was then decided to allocate two classes, first one for a generic introduction of the tool and the second one for a group practical exercise.

Feedback shared with the course lecturer afterward was that more time was required for the practical classes. Idea for future classes could be to allocate three slots for the topic, dedicating the first class for the introduction of the tool and diluting the group work in two sessions, allowing the students to spend more time building the own business case.

Even if three classes would be required, it was observed that some other small aspects needs to be taken care before the next course.

Most of the students, even if asked, did not create beforehand the Cisco NetAcad account. This caused a bit of delays in start the class. For future course the account should be created by default during the starting of the course or perhaps be requested by other non IoT courses, so students will have it already when IoT practical classes will start.

Another problem, emerged during the lab sessions, was that some of the IoT business cases, developed by the students in previous classes of the course, were not achievable in Cisco Packet Tracer. This was mostly due to fact that Cisco simulator did not have sensors required in the student's cases. For future iteration it would be better to introduce Cisco Packet Tracer to the students earlier so they could familiarize with the options offered by the tool and create their own IoT business cases on the top of it.

In order to enhance the IoT practical experience of the students another possibility for the future could be to limit the time spend in simulating the IoT cases with Cisco Packet Tracer and have one class where students could practice with real microcontroller scenarios.

This would however require a more complex setup as hardware and sensors would be necessary, a dedicate IoT network would be needed and also students groups should be built in such a way that programming skills are available in the group.

To conclude, despite the limitations and bugs in Cisco Packet Tracer, the simulation tool was a very good choice for the Internet of Thing study course. Regarding the contents and the methodology a positive feedback was shared by students and the course lecturer. Everyone also seemed to agree that additional time is required in future courses in order to have a more complete IoT experience for the students.

6 Conclusions

As discussed in the previous chapter 5 the overall conclusion on the thesis work was positive, as the IoT simulations were delivered and students of the Internet of Thing course were able to work on them familiarizing with the IoT simulations.

Conclusions below cover both the deliverables and the methodology of the thesis work.

Regarding purely the IoT exercises, the research question of the thesis was answered by analyzing the Cisco Packet Tracer, finding if suitable for the simulations, and by building the IoT automations with it.

As also requested by the IoT course lecturer, the IoT cases were built to offer the students both a pre-configured environment and also a showcase of IoT simulations for them to immediately understand how Cisco Packet Tracer works, avoiding them to spend time on setting up a network and basic simulations.

As mentioned multiple time in the thesis document, the IoT simulations were the first step for practical classes in the Internet of Things course. The Cisco Packet Tracer exercises should be treated as a starting point for future and more complex IoT simulations in Metropolia University of Applied Science IoT courses.

Future studies could in fact cover the possibility to integrate Cisco Packet Tracer cases with more complex IoT automations by utilizing real microcontroller and sensors hardware but also to expand the four delivered exercises with more complex automations. One could assume that, as Cisco will release more and more IoT courses over the coming years and, Cisco Packet Tracer tool will be updated in the future with more IoT functionalities.

Future suitability of the tool and also a comparison with other available IoT simulators could be also ground for future studies, as in fact, in this thesis work no comparison research was made with other available IoT simulators such: IBM, NetSim or NodeRed.

Conclusions on the methodology should be separated between the process to conduct the thesis works and the actual implementation of it.

As explained in the chapter 3 this thesis work was conducted as a project, first gathering the requirements, then developing the automations and last introducing them to the students and gathering the feedback.

The entire process was successful as it was proven to be reliable, it had a strict timeline to follow and in multiple instances the feedback from the IoT course lecturer helped to steer it. Dividing the thesis work into phases also helped to track down properly the progress while making sure to respect the needed deadline.

It is advisable, for future studies with a similar research questions, to utilize the same methodology.

Regarding the implementation phase, when exercises were introduced to the students, a more careful planning should be done in future implementations of the IoT courses. As largely explained in the previous chapter 5 the conclusion was that not enough time in the course was reserved for the practical section, making it hard for the students to really spent time on familiarizing with the IoT components.

To be mentioned also is that a direct contact with the students was beneficial for them in order to have a person to help them during the exercises. It was also beneficial for the thesis prospective as a direct work in the class and the feedback gathering was very helpful for drawing the conclusion of the thesis work.

Future studies could also help to re-structure the IoT course, synchronizing exercise perhaps in different phases of the course and having them going hand-to-hand with the theoretical part and not only in a concentrated manner.

Concluding, as also reflected in the students feedback, the thesis work was successful by supporting the practical classes of the Internet of Thing course. For future implementations of the course changes in the agenda are advised, along with studies to explore different IoT simulator technologies and feasibility to utilize real IoT hardware.

References

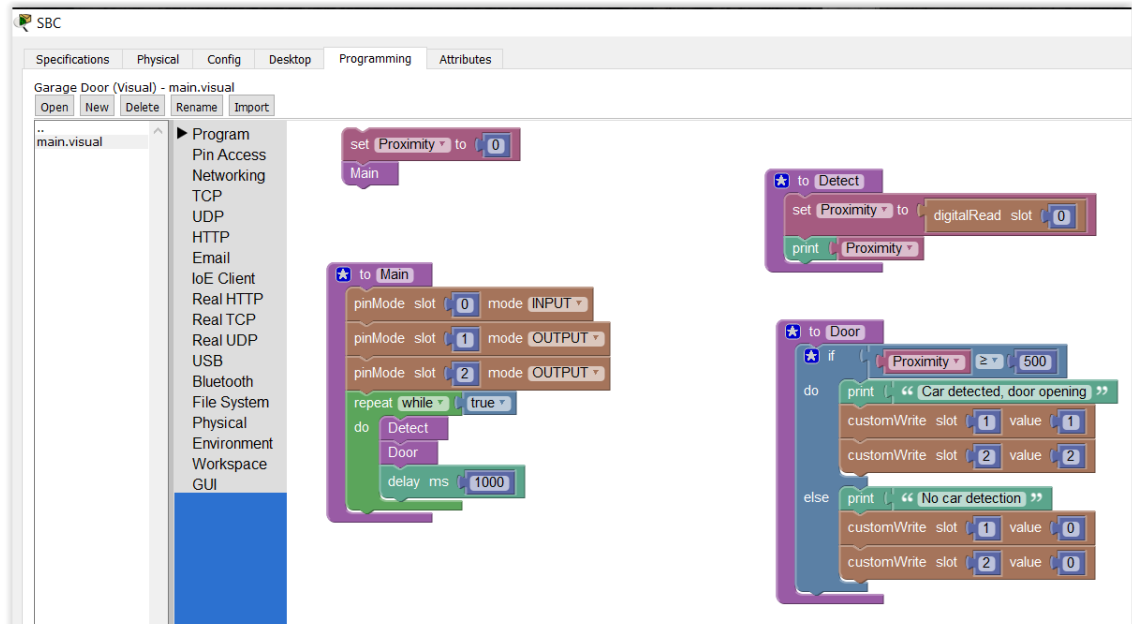
- [1] Egham. (2017, February 7). Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. Retrieved from <https://www.gartner.com/newsroom/id/3598917>
- [2] Thomas Erl, Z. M. (2013). Cloud Computing Concepts, Technology & Architecture. Prentice Hall.
- [3] A Brief History. (n.d.). Retrieved from What is cloud: http://whatiscloud.com/origins_and_influences/a_brief_history
- [4] Pelkey, J. (n.d.). Entrepreneurial Capitalism and Innovation. A History of Computer Communications 1968-1988. Retrieved from History of computer communication: http://www.historyofcomputercommunications.info/Book/2/2.1-IntergalacticNetwork_1962-1964.html
- [5] Peter Mell (NIST), T. G. (2011, September). The NIST Definition of Cloud Computing - SP 800-145. Retrieved from <https://csrc.nist.gov/publications/detail/sp/800-145/final>
- [6] Ovidiu Vermesan, P. F. (2014). Internet of Things - From Research and Innovation to Market Deployment. River Publishers.
- [7] Arpan Pal, B. P. (2017). IOT Technical Challenges and Solutions Artech house
- [8] Wentao Shang, Y. Y. (2016, February 10). Challenges in IoT Networking via TCP/IP Architecture. NDN Technical Report NDN-0038, 2016.
- [9] Pethuru Raj, A. C. (2017). The Internet of Things: Enabling Technologies, Platforms, and Use Cases. CRC Press.
- [10] About LoRa Alliance™. (2018). Retrieved from LoRa Alliance: <https://loralliance.org/about-lora-alliance>
- [11] What is the LoRaWAN™ Specification? Data Rates (2018). Retrieved from LoRa Alliance: <https://loralliance.org/about-lorawan>

- [12] Ferran Adelantado, X. V.-P.-S. (2017). Understanding the Limits of LoRaWAN. IEEE Communication Magazine.
- [13] Michael Freitag, H. K., J.P (2018) Dynamics in Logistics. 6th International conference, LDIC 2018. Bremen, Germany: Springer
- [14] Khaldoun Al Agha, G. P. (2016). Advanced Network Set - Volume 2 Mobile and Wireless Networks. ISTE LTD, John Wiley & Sons, Inc.
- [15] DIRECTIVE 2014/53/EU OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL. (2014, April 16). harmonisation of the laws of the Member States relating to the making available on the market of radio equipment and repealing Directive 1999/5/EC.
- [16] (2017, February). ETSI EN 300 220-2 V3.1.1 (2017-02). European Telecommunications Standards Institute (ETSI).
- [17] Dominique Paret, J-P. H (2017). Secure Connected Objects. ISTE LTD, John Wiley & Sons, Inc.
- [18] (2016). Retrieved from <https://ask.sigfox.com/questions/606/is-there-a-general-network-architecture-descriptio.html>
- [19] Giedre Dregvaite, R. S. (2016). Information and Software Technologies. 22nd International Conference, ICIST 2016. Druskininkai, Lithuania: Springer.
- [20] Atkinson, J. (2015, December 2). In search of the low-power wide area network standard for IoT. Retrieved from ITproportal: <https://www.itproportal.com/news/in-search-of-the-low-power-wide-area-network-standard-for-iot/>
- [21] GSMA. (2016, September). 3GPP Low Power Wide Area Technologies - GSMA white paper.
- [22] 3GPP. (2016, February). RAN approved REL-13 NB_IOT CRs (RAN#72):. Retrieved from http://www.3gpp.org/ages/PDF/R13_IOT_rev3.pdf

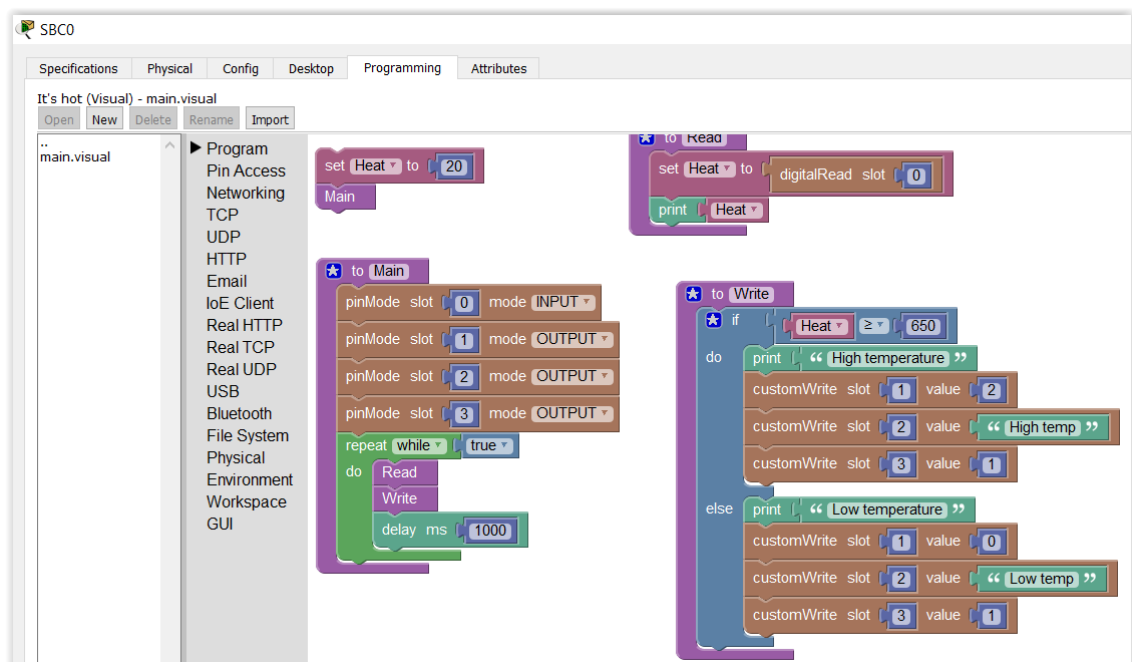
- [23] 3GPP, Luo Chao. (2017, March 20). 3GPP TS 45.001 V14.1.0 (2017-03) technical specifications. 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; GSM/EDGE Physical layer on the radio path; General description (Release 14). 3GPP.
- [24] Corporate Social Responsibility. (2018). Retrieved from Cisco:
<https://www.cisco.com/c/en/us/about/csr.html>
- [25] What's new in Cisco Packet Tracer 7.0. (2018, January). Retrieved from Packet Tracer Network: <http://www.packettracernetwork.com/features/packettracer-7-new-features.html>

Blockly custom software for IoT Simulations

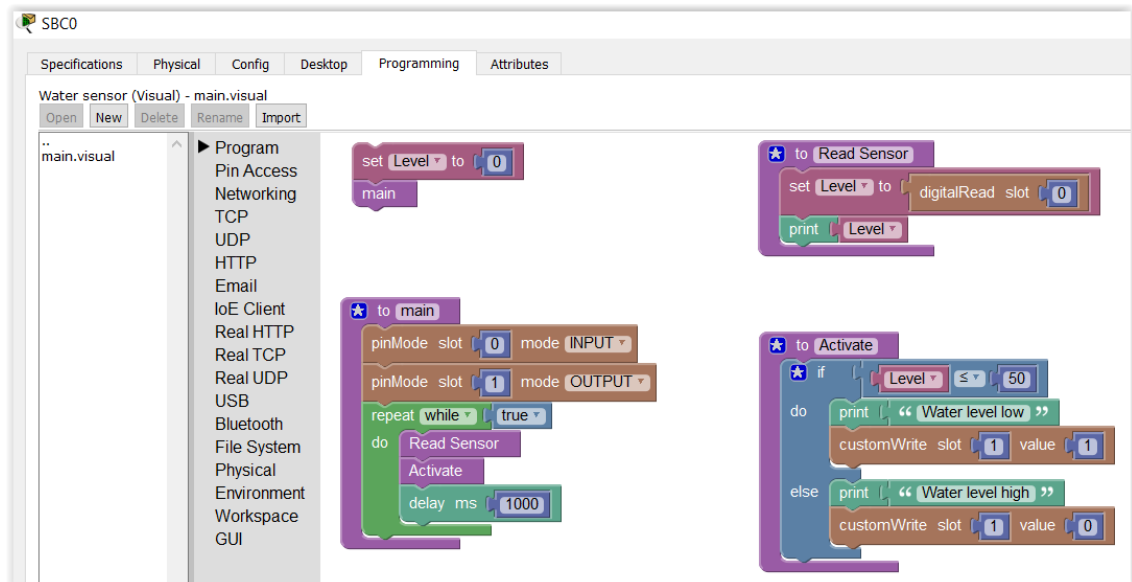
Smart-Home 1



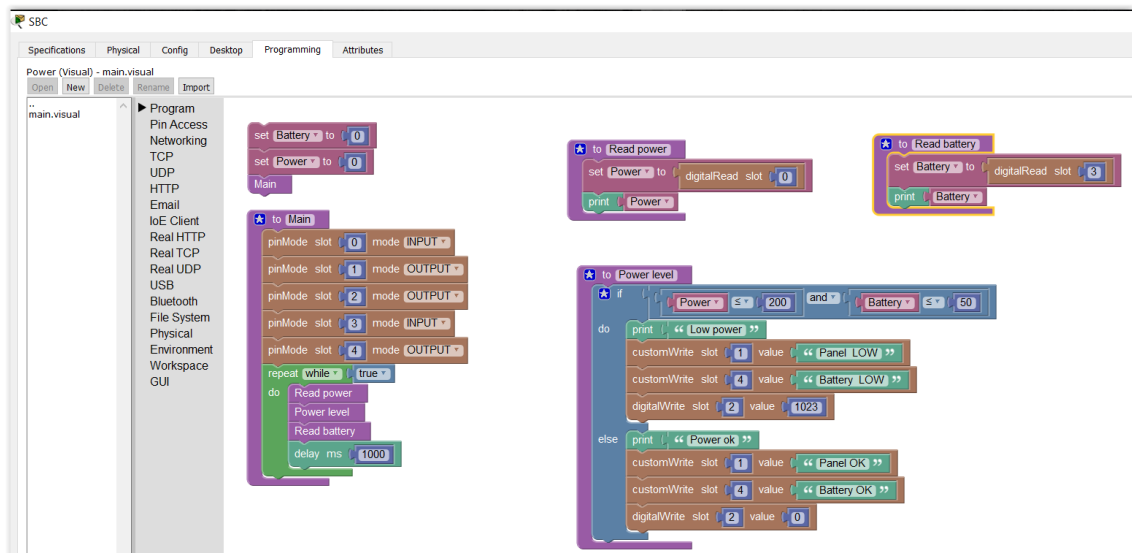
Smart-Home 2 (SaaS)



Smart-Campus



Smart-Industrial



Network details utilized in the IoT simulations

Smart-Home 1

Home network:

SSID: HomeWIFI

Pwd: HomeWIFI

IP: 10.0.0.0

Subnet: 255.0.0.0

IOT server:

IP:10.0.0.10

DNS: iothomepage.com

User: Admin

Passwordd: Admin

Corporate office network:

IP: 40.15.25.0

Subnet: 255.0.0.0

DHCP server: 40.15.25.10

Smart-Home 2 (SaaS)

Home network

SSID: HomeWIFI

Password: HomeWIFI

IP: 192.168.0.0

Subnet: 255.255.255.0

IOT Server (remote)

IP: 210.190.15.20

DNS: IOTSmarthome.com

User: HomeIOT

Password: HomeIOT

3G network:

Provider: PabloNET

Smart-Campus

Class building network

IP: 150.150.0.0

Subnet: 255.255.0.0

DHCP server: 150.150.0.1

Apartments building network

SSID: ApartmentWIFI

Password: HomeWIFI

IP: 210.140.0.0

Subnet: 255.255.0.0

IOT network

IP: 10.0.0.0

Subnet: 255.0.0.0

DHCP server: 10.0.0.10

IOT Server (remote)

IP: 10.0.0.10

DNS: IOThomepage.com

User: Device

Password: Device

Smart-Industrial

Office network

IP: 190.148.0.0

Subnet: 255.255.0.0

DHCP server: 190.148.0.2

Solar panel network

IP: 130.160.15.0

Subnet: 255.255.0.0

DHCP server: 130.160.15.0

Wind farm network (4G)

IP: 140.185.13.0

Subnet: 255.255.0.0

DHCP server: 140.185.13.10

IOT industrial network

SSID: IOTNetwork

Password: IOTNetwork

IP: 192.168.0.0

Subnet: 255.255.255.0

IOT control network

SSID: ControlWIFI

Password: IOTSecure

IP: 168.140.10.0

Subnet: 255.255.255.0

IOT Server (remote)

IP: 120.100.44.10

DNS: iotcontrolpage.com

User: IOTindustrial

Password: IOTindustrial

Students feedback form

Internet of things (IOT)

1 – In your opinion, how much IOT is going to revolutionize the IT industry and what are the key elements?

2 – What is in your opinion the maturity of IOT technologies in the today private and industrial market?

3 – What are the IOT areas that most interest you and why?

Tool (Cisco packet tracer)

1 – What is your opinion on Cisco packet tracer to be used as IOT simulator tool?

Please drag in the smiley face on the correct grade below



1 – Excellent tool	2 – Good tool	3 – Ok, but...	4 – Not the tool for the job
--------------------	---------------	----------------	------------------------------

2 – Please list advantages and limitations of the tool (based on your user experience)

Good (max 3 points)

1) 2) 3)

Bad (max 3 points)

1) 2) 3)

IOT simulation exercises

1 – Please share some open feedback on the class exercises

--

2 – Please list some highlights and lowlights on the simulations exercise

Highlights (max 3 points)

1) 2) 3)

Lowlights (max 3 points)

1) 2) 3)

3 – Please share some feedback (if any) of how the exercise could be improved (e.g. contents, tool, method)

Extra feedback