REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre :…… /M2/2017

Mémoire

présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Génie Logiciel et Systèmes Distribués**

Transforming IoT Applications to RESTful Services:Toward a Cloud-Based Exploitation of Cyber Physical Systems.

Par :

TAHRI HAROUNE

Soutenu le …\…\….. , devant le jury composé de :

		Président
TIBERMACINE Okba	MCA	Rapporteur
		Examinateur

Acknowledgements

In the Name of Allah, the Most Beneficent, the Most Merciful.

All the praises and thanks be to Allah for giving me the strength to complete this project.

I would like to express my gratitude to my supervisor Dr.Tibermacine Okba for his immense knowledge, motivation, understanding and support that contributed to the success of this project.

My deepest gratitude to my parents for their love, kindness and support. Special thanks to my brothers for their love and moral support during my studies.

Last but not least, I'd like to thank my professors and colleagues for the great learning experience we had through the past few years.

Abstract

the world is witnessing an explosion in the computing and communication technology, such Internet of Things (IoT) which is a world of interconnected Things which are capable of sensing, actuating and communicating among themselves and with the environment.

Internet of Things devices are prevalent in all aspects of our lives, e.g., thermostat and smart lights.., those devices can be remotely controlled by the end user. their application must be a service which is RESTful (for its lightweight when sending data, ..) , so our approach it's a migration tool which migrate the Internet of Things systems to RESTful-based web service in the cloud, and that's allows to the end-user the remotely controlling and accessing to the service data.

Résumé

Le monde assiste à une explosion des technologies de l'informatique et de la communication, comme l'Internet des objets (IoT) qui est un monde de choses interconnectées capables de détecter, d'actionner et de communiquer entre elles et avec l'environnement.

Les appareils de l'Internet des objets sont répandus dans tous les aspects de nos vies, par exemple, le thermostat et les lumières intelligentes.., ces appareils peuvent être contrôlés à distance par l'utilisateur final. leur application doit être un service qui est RESTful (pour sa légèreté lors de l'envoi de données, ..), donc notre approche est un outil de migration qui migre les systèmes de l'Internet des objets vers un service Web basé sur RESTful dans le cloud, et cela permet end-user le contrôle à distance et l'accès aux données de service.

Contents

List of Figures

List of Abbreviations

IoT: Internet of Things.

CPS: Cyber Physical Systems.

REST: Representational State transfer.

SOAP: Simple Object Access Protocol.

SOA: Service Oriented Architecture.

QoS: Quality of Service.

XML: Extensible Markup Language.

HTTP: Hypertext Transfer Protocol.

URI: Uniform Resource Identifier.

URL: Uniform Resource Locator.

JSON: JavaScript Object Notation.

API: Application Programming Interface.

HTML: Hyper Text Markup Language.

IT: Information Technology.

ICT: Information and Communications Technology

VPN: Virtual Private Network.

LED: light-Emitting Diode.

IDE : Integrated Development Environment.

GUI: Graphical User Interface.

Introduction

In view of the widespread to the Internet of Things which works on the connection of anything with the Internet , and Cyber Physical Systems that's considered as systems of collaborating computational entities which are in connection and interaction with the the environment, they ideally use networks and services (web services).

Web services is any piece of software that can be available over the internet and perform a certain set of services, uses a standardized XML messaging system. REST is set of constraints to be used for creating Web services, REST based on lightweight HTTP methods.

This work aims to present a solution to reuse stand-alone Cyber physical systems on the cloud through migration to REST-based web services. This solution promotes reusability and scalability of such application by using cloud resources for storage, processing and scheduling. Hence, this manuscript is structured as follows:

- **Web Services and The Cloud Computing :**
  The first chapter is devoted to define the web service and cloud computing, describing general operations in these contexts as their principles, advantages and disadvantages.

- **Internet Of Things (IoT) and Cyber Physical Systems(CPS) :**
  The second chapter discusses the Internet Of Things and Cyber Physical Systems, their features, architecture, challenges and compare between them.

- **Conception :**
  The third chapter presents the architecture and the design our Approach and prototype.

- **Implementation :**
  This forth chapter outlines the implementation of the proposed solution and discusses the obtained result after testing it.

At the end of this thesis, a general conclusion is presented as a recapitulation of the released work.

Chapter 1

Web Services and Cloud Computing

1.1 Web Services

1.1.1 Introduction

With the widespread use of the web, researchers have developed software libraries to ensure and simplify communication between machines and applications connected via the network, this software is called "web services in the next.

1.1.2 Definition

"A service is a program unit which can be called by standardized procedures, and which can independently execute assigned function. Each service is executed on hetero-environment, namely different hardware, OS, programming language. Therefore, the service can be easily added or replaced or re-used. The granularity of the service size varies from function to function. Some services include a business process".[11]

So the web service is an electronic service, offered by universal technologies developed according to global networking protocols (mainly the Internet) which acts as a communication infrastructure.

- **The business process :** Is a chain of activities carried out by an actor, triggered by an event and which results in a precise deliverable.
- **characteristics of web services**
  - Service contract is exposed in an independent interface to all platforms. it contains the information necessary to describe the service and how to use it by the client by mention : the target and function of its operations, a set of conditions under which the operations are provided, the messages that need to be exchanged in order to engage the operation, the structure of the messages and it's provided data types ..
  - A service can be dynamically located and invoked by processing the description file of the service (service contract) which contains where is the service and how to invoke it.
  - The service is autonomous, it means that the service has the ability to carry out its logic independently of outside influences.

1.1.3 Service Oriented Architecture

The objective of a service-oriented Architecture (SOA) is to decompose functionality in a set of basic services , provided by components and to describe in detail the diagram of interaction between these services. In a few years, SOA has become a major theme for the

business information system. It represents a new approach to the design of applications that increase flexibility, agility and responsiveness at the information system level based on the principle of weak coupling and while guaranteeing interoperability between these applications.

1.1.3.1 SOA Definition

"Service Oriented Architecture (SOA) is defined as "an enabling framework for integrating business processes and supporting information technology infrastructure as loosely coupled and secure, standardized components services that can be reused and combined to address changing business priorities."[12]

SOA provides a vision where applications (internal and external) are exposed as services and can communicate with each other in a simple way. In this vision, each application provides some (or all) of its functionality in the form of invokable services by other applications. This concept of architecture therefore uses services as components principles for building distributed applications in an easy and inexpensive way.

1.1.3.2 SOA Characteristics

Each system's software architecture reects the different principles and set of trade offs used by the designers. Service oriented software architecture has the following characteristics :

1. **Discoverable and Dynamically Bound**

   SOA supports the concept of service discovery, which means that the services is easy to find and its name has a meaning of the service that it offers. A service consumer that needs a service can discovers service to use based on a set of criteria at runtime. The service consumer asks a registry for a service that fullls its need. services are bound to service requests at runtime and the choice of service is determined with minimal user intervention.

2. **Services are reusable**

   They are designed so that they can be reused later, and it's the goal of the SOA to make from the development easy task.

3. **Self-Contained**

   The tasks performed by a service have limits. The service has control over this limit and does not depend on other services to accomplish its task. This is true in the case of atomic services, composed services are in need to its atomic services. However, in general, we can say that elementary web service are self-contained and they do not need other parties to finish their execution.

4. **Modular**

   One of the most important aspects of SOA is the concept of modularity. A service supports a set of interfaces. These interfaces should be cohesive, meaning that they should all relate to each other in the context of a module. so that services can easily be aggregated into an application with a few well-known dependencies. The modularity has many types : Modular Decomposability , Modular Composability, Modular Understandability, Modular Continuity, Modular Continuity, Modular Protection.

5. **Interoperability**

   Service-oriented architecture is capable of coordinating number of services in different environments and operates them in a reliable and understandable manner, Each service provides an interface that can be invoked through a connector type, such that information from the various environments transmits the data between those organization mutually understood by both even though there are differences in nature of language, programming interface and in any other entities.

6. **Loose Coupling**

   The services are designed to interact with the minimum of interdependencies. Service-oriented architecture promotes loose coupling between service consumers and service providers and the idea of a few well-known dependencies between consumers and providers. And this is what qualifies services as self contained.

7. **Location Transparency**

   Location transparency is an important characteristic of SOA. when Consumers locate the service that they need, they will know a service's location. The lookup and dynamic binding to a service at runtime allows the service implementation to move from location to location without the client's knowledge. The ability to move services improves service availability and performance.

8. **Composability**

   Composability is Collections of services that can be coordinated and assembled to form a composition of services. One of the benefits of SOA is the ability to build new systems from existing services. the composition allows the reuse of web services, after discovery, you must be able to compose them by exploiting the technologies offered by the Internet and using a set of standards for composition. Composition has three types : An application, Service federations and Service orchestration, this classification depends on the business process.

1.1.3.3 SOA Collaborations

Figure 1 shows the collaborations in a service-oriented architecture. The collaborations between the parts of SOA (service provider, service registry and service consumer) by following the "publish, find, bind and invoke" paradigm.[5]

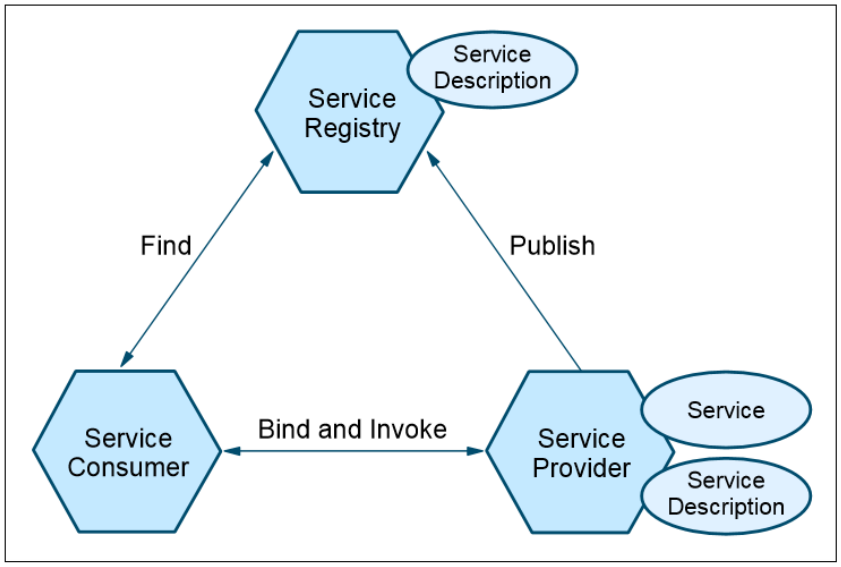

Figure 1: Collaborations in service-oriented architecture

- **Service description :**

  A service description specifies the format of the request and response from the service. This description may specify a set of preconditions, post conditions and/or quality of services.

The roles in a service-oriented architecture:

- **Service consumer** Service consumer or requesting client is who requests a specific web service to meet their needs. It initiates the enquiry of the service in the registry, binds to the service over a transport, and executes the service function. The service consumer executes the service according to the interface contract.

- **Service provider** The service provider implements the web service and makes it available to everyone on the internet. It accepts and executes requests from consumers.

- **Service registry** A service registry provides a place where the service consumer can find a web services and the service provider can publish a description of the new web services.

The operations in a service-oriented architecture :

- **Find** Service consumer can search for a service by querying the service registry for a service that it needs with a set of keywords and QoS values.
- **Publish** A service provider must publish its service description in a Registry to be discovered and invoked by service consumer.
- **Bind and Invoke** After finding the service on the service registry by the service consumer, it must retrieve the service description, the service consumer proceeds to invoke the service according to the information in the service description.

1.1.3.4 Advantages of SOA

- **Service reusability** In SOA, applications are made from existing services. Thus, services can be reused to make many applications.
- **Easy maintenance** As services are independent of each other So they can be updated and modified easily without affecting other services.
- **Platform independent** This platform independence allows the integration of different requested services from different sources to run harmoniously, independent of the platform.
- **Availability** SOA facilities are easily available to anyone on request.
- **Reliability** SOA services are complete and self-contained programs. This makes it easy for testing, debugging or any form of maintenance.
- **Scalability** Different organizations have different levels of complexity in their architecture. So, the ability of services to be successfully run on different servers within an environment increases the scalability of the service.

1.1.4 Simple Object Access Protocol (SOAP)

1.1.4.1 Definition

“ Simple Object Access Protocol is a standard for an XML-based exchange of information between distributed applications, transferring data over such standard transport protocols as HTTP. SOAP is a lightweight, platform-independent protocol ”.[29]

- Standardized by the W3C which is the proprietary of it (W3C is an international community where Member organizations, a full-time staff, and the public work together to develop Web standards).
- SOAP allows the remotely call to methods thanks to the use of HTTP protocols.

1.1.4.2 Essential Parts of SOAP

SOAP consisting of the following parts :

- An envelope that defines a framework for describing what is in the message and how to process it.
- A set of encoding rules for expressing instances of application-defined data types.
- A convention for representing remote procedure call and responses.
- A binding convention for exchanging messages using an underlying protocol.

1.1.4.3 SOAP Messages Blocks

The "SOAP message" [26] which is what is exchanging between to the web service and the client application has several blocks as in Figure 2 :

- An envelope that defines the structure of the message.
- A header (optional) which contains the header information (authorizations and transactions for example).
- A body containing the information on the call and the answer an error management which identifies the error condition of attachments or attachments (optional).

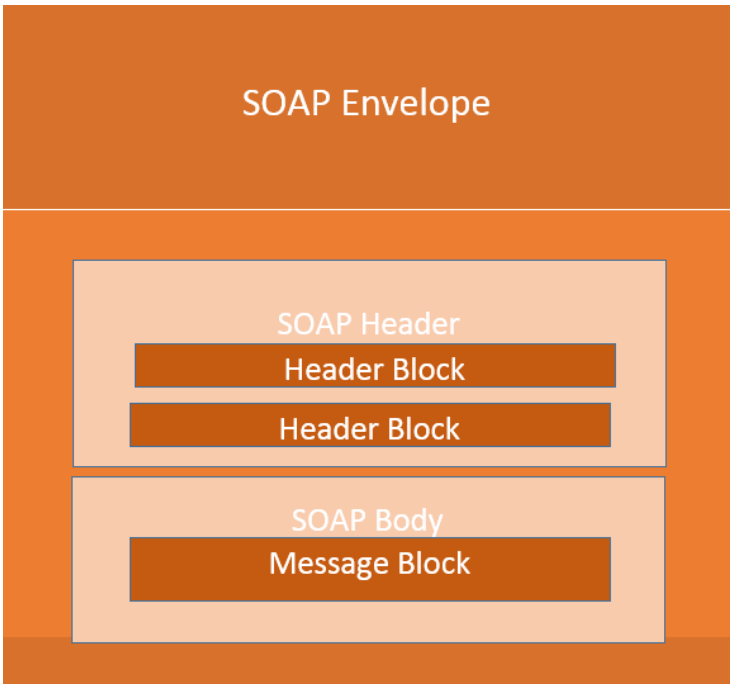

Figure 2: various building blocks of a SOAP Message .

1.1.4.4 Advantages of SOAP

SOAP is the protocol used for data interchange between applications. Below are some of the reasons as to why SOAP is used [26] .

- In the web there is lot of services are developed with different programing languages, so it must find a web standards. SOAP is the perfect medium which was developed in order to achieve this purpose.
- Since SOAP is based on the XML language is a light-weight protocol that is used for data interchange between applications. which itself is a light weight data interchange language.
- SOAP is designed to be independent from the both platform and operating system. So the SOAP protocol can work any programming language based applications on both Windows and Linux platform.
- It works on the HTTP protocol –SOAP works on the HTTP protocol, which is the default protocol used by all web applications. So , there is no sort of requirements or specifications which is required to run the web services built on the SOAP protocol to work on the World Wide Web.

1.1.5 Representational State Transfer (REST)

REST was proposed by Roy Thomas Fielding in his thesis Architectural Styles and the Design of Network-based Software Architectures [16].

1.1.5.1 Definition

"REST is a style of software architecture defining a set of constraints to be used to create web services. REST architecture style web services, also called RESTful web services, establish interoperability between computers on the Internet. REST web services allow requesting systems to manipulate web resources via their text representations through a set of stateless, predefined uniform operations".[24]

REST focuses on defining resources identified by URIs Uniform Resource Identifier, and uses HTTP protocol messages to define the semantics of client / server communication.

1.1.5.2 Resources in Vision of REST

The key abstraction of information in REST is "Resource". Any information that can be named can be a resource: a document or image, a temporal service, a collection of other resources ... , and it is through :

- **Resource identifier** A resource is identified by a resource identifier. It allows the components of the architecture to identify the resources they are handling. On the web these identifiers are the URI (Uniform Resource Identifier).
  Example of URI :
  "http://api.example.com/REST/"

- **Resource representation** The components of the architecture manipulate these resources by transferring representations of these resources. On the web, we most often find representations in HTML, JSON or XML format.

- **Operation on the resource** Resources are manipulated by the transfer of representations through a uniform interface addressed by the resource identifier.

1.1.5.3 Principles of REST Architecture

The design principles and constraints of the REST architectural style and introduce the notion of composite RESTful Web service.[18]

- **Resource addressing through URI :** Resources should be uniquely identifiable through a single URL.

- **Uniform interface :** Once a resource has been identified, the set of operations that can be applied to it is fixed by design to the same four HTTP methods like as shown in Figure 3 (PUT, GET, POST, and DELETE) :

  - **GET :** It instructs the server to transmit the data identified by the URL to the client. Data should never be modified on the server side as a result of a GET request.

  - **PUT :** A request is used when you wish to create or update the resource identified by the URL.

  - **Delete :** It should be used when you want to delete the resource identified by the URL of the request.

  - **Post :** Is used when the processing you wish to happen on the server should be repeated.

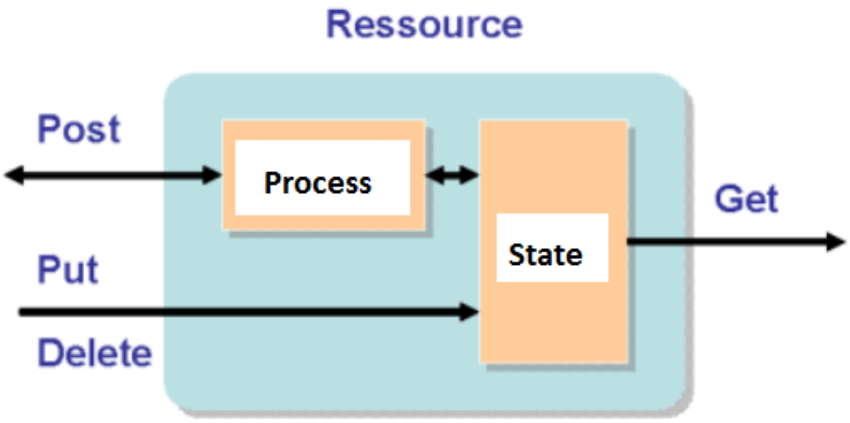

Figure 3: The use of HTTP verbs.

- **Client-server based :** The client-server environment defines a connection mode between several Programs:

- **Stateless request :** This means that the server does not process a request by referencing elements from a previous request. At the client level, everything necessary to process the request must be included in it.

1.1.5.4 Advantages of REST

- **Scalability**

  This protocol stands out due to its scalability. Thanks to the separation between client and server, the product may be scaled by a development team without much difficulty.

- **Flexibility and Portability** With the indispensable requirement for data from one of the requests to be properly sent, it is possible to perform a migration from one server to another or carry out changes on the database at any time. Front and back can therefore be hosted on different servers, which is a significant management advantage.

- **Independence** Due to the separation between client and server, the protocol makes it easy for developments across the various areas of a project to take place independently. In addition, the REST API adapts at all times to the working syntax and platform. This offers the Opportunity to try several environments while developing

1.1.6 Difference between SOAP and REST

The most important difference points are :

- **SOAP**
  - SOAP is a protocol.
  - SOAP permits XML data format only.
  - SOAP defines standards to be strictly followed.
- **REST**
  - REST is an architectural style.
  - REST permits different data format such as Plain text, HTML, XML, JSON etc.
  - REST does not define too much standards.

1.2 The Cloud Computing

1.2.1 Definition

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".[13]

Many people use the Cloud daily without knowing it. We find it by example in all versions of mails, Web mail, Hotmail, or Gmail, in offices like Word and Microsoft. it means that the users of these IT access, through the Internet, to resources that they cannot physically locate.

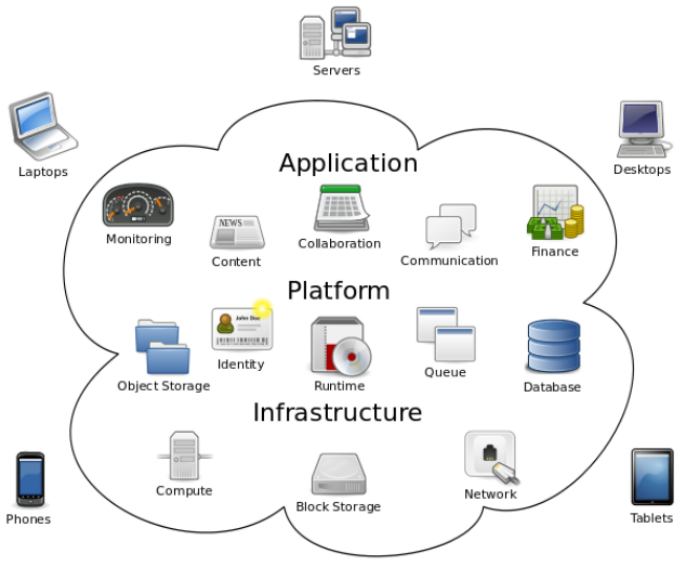

Figure 4: Cloud Computing.

Figure 4 presents the environment, infrastructures, platforms and applications available in the Cloud Computing .

1.2.2 Essential Characteristics of the Cloud Computing

There are five characteristics of the cloud computing :

1.2.2.1 On-Demand Self-Service

The concept of on-demand self-service is paramount for cloud service users. On-demand self-service allows the user to be able to provision, but also free up remote resources in real time as needed, and without requiring human intervention.

1.2.2.2 Broad Network Access

All resources must be accessible and available to the user universally and simply across the network, regardless of the clients used (server, PC, mobile client, etc.).

1.2.2.3 Resource Pooling

The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand.

1.2.2.4 Rapid Elasticity

Cloud computing offers a way to provide the IT resources necessary for evolution through the rapid elasticity. This feature allows users to quickly provision new resources so that they are able to respond to a sudden rise or fall in load. It is never easy to predict the resources that will be necessary to set up any IT service, especially when this need is constantly changing.

1.2.2.5 Measured Service

Cloud systems must be able to self-control and manage themselves to allow internal optimization of the system. To do this, they rely on baseline measurements obtained through various monitoring mechanisms. These precise measures allow fair billing of users; they will only pay for the resources they have used and only for the time they have used them.

1.2.3 Cloud Services Models

Conceptually, users acquire computing platforms or IT infrastructures from computing Clouds and then run their applications inside. Therefore, computing Clouds render users with services to access hardware, software and data resources, thereafter an integrated computing platform as a service, in a transparent way :

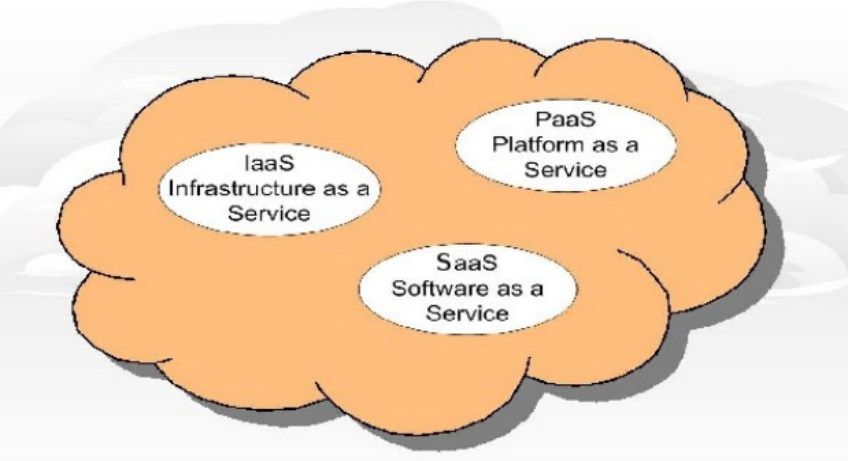

Figure 5: Cloud models .

Figure 5 shows the three existing models of the Cloud Computing .

1.2.3.1 Infrastructure as a Service (Iaas)

Infrastructure as a Service is a main service in the cloud, this service provides the company with different IT components such as storage spaces, network equipment,.. Users can access these services on demand via the Internet without restriction, as if they are working on local hardware.

- Advantage: great flexibility, total system control (remote administration by SSH or Remote Desktop, RDP), which allows you to install any type of business software.
- Disadvantage: need for system administrators as for conventional on-site server solutions.

1.2.3.2 Platform as a Service (Paas)

Platform as a Service is a model composed of all the elements and services necessary to facilitate the development of applications where Paas prepares specialized environment to help users in the construction, delivery and extension of their projects. the user hires a platform on which he can develop, test and execute his applications. because Paas avoids buying and installing software, it does not manage or control the underlying infrastructure,

but controls the applications deployed, also is an execution platform hosted by an operator connected to the internet.

- Advantage: the deployment is automated, no additional software to buy or install.
- Disadvantage: limitation to one or two technologies (eg Python or Java). No control over the underlying virtual machines. Only suitable for web applications.

1.2.3.3 Software as a Service (Saas)

Software as a Service is a cloud computing offering that gives users access to a provider's cloud software. Users do not install applications on their devices. They reside on a remote cloud network which they access via the web or via an API. With apps, users can store and analyze data and collaborate on projects.

- Advantage : more installation, more updating (they are continuous at the supplier), more data migration etc. Test new software with ease.
- Disadvantage: limitation by definition to the proposed software. No control over data storage and security associated with the software. Responsiveness of web applications not always ideal.

1.2.4 Types of the Cloud

Clouds can be classified in terms of who owns and manages the cloud infrastructure; a common distinction is Public Clouds, Private Clouds, Hybrid Clouds and Community Clouds.[6]

1.2.4.1 Public Cloud

A public cloud, or external cloud, is the most common form of cloud computing the company's IT resources are stored on a shared server, in other words shared between several clients, and accessible via the Internet. These servers are partitioned to prevent data leakage.

1.2.4.2 Private Cloud

As its name suggests, it is dedicated to a single user. The advantage of the private cloud is its high level of security, serves customers within the business fire-wall, reinforced by a VPN connection. The private cloud is administered by the company itself or a service provider. Most of the private clouds are large company or government departments who prefer to keep their data in a more controlled and secure environment.

1.2.4.3 Hybrid Cloud

A composition of the two types (private and public) is called a Hybrid Cloud ,the goal of implementing certain activities. For example, the public cloud is used by employees for operational tasks, while the private cloud is used to host the company's e-commerce website or its financial data, to reduce the risk of hacking.

1.2.4.4 Community Cloud

The idea of a Community Cloud is derived from the Grid Computing and Volunteer Computing paradigms, it is more rarely used, it consists of sharing a given space between several companies with the same security and confidentiality requirements with sharing the cost. So it's like a shared private cloud.

1.3 Web Services in the Cloud

Web Services and the Cloud are complementary activities, both play important roles in IT. Web Services encapsulates Cloud Computing because Cloud Computing uses Web Services for connections.

And because web service is a software designing model and method so it can be integrate well-defined services to a new solution in the cloud.

1.4 Conclusion

In this chapter, we presented the main concepts and definitions related to the fields of web services and cloud computing. The next chapter is devoted to the presentation of the Internet of Things (IoT) and Cyber Physical Systems (CPS), and their principles and characteristics.

Chapter 2

Internet of Things (IoT) and cyber Physical systems(CPS)

2.1 Internet of Things (IoT) :

2.1.1 Introduction

In this Chapter we talk about two Important concepts We'll discuss their features, principals and architecture, and a comparison between them.

2.1.2 Definition

"IoT is a world of interconnected *Things* which are capable of sensing, actuating and communicating among themselves and with the environment (i.e., smart things or smart objects) while providing the ability to share information and act in parts autonomously to real/physical world events and by triggering processes and creating services with or without direct human intervention."[8]

The objects around us (such as lamps, machines, clothing, etc.), whether physical or virtual, now have the ability to communicate with each other in real time. Connected objects offer the possibility of defining the precise needs of an individual, so it has to offer him a services.

Figure 1: Things and Internet of Things.[17]

Internet of Things is refer to the general idea of things, so every things from our environment can be a part of the IoT technology as shown in Figure 1.

2.1.3 IoT Architecture

A layered architecture of an IoT is depicted in Figure 2

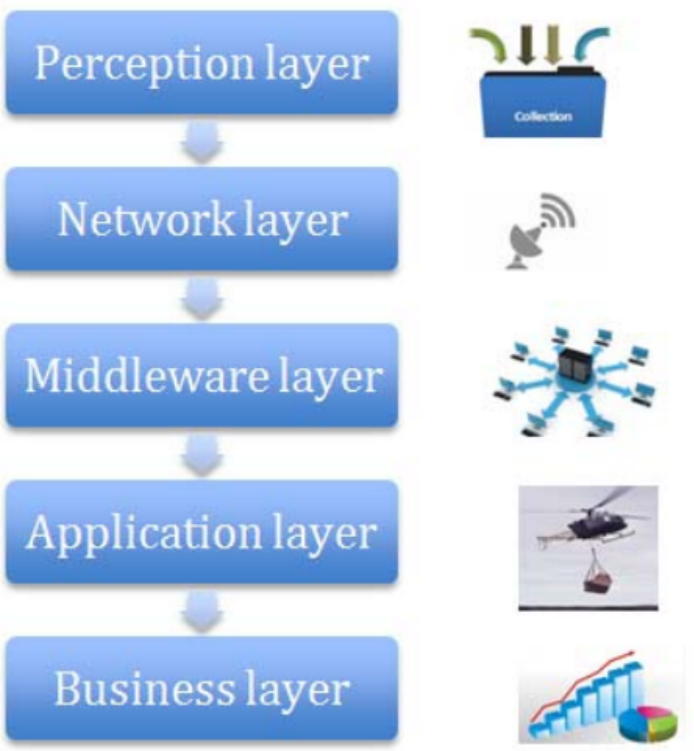

Figure 2: IoT's five layers.[7]

2.1.3.1 Perception Layer

It is also known as a sensor layer, which gives a physical meaning to each object. It works like people's eyes, ears and nose. It has the responsibility to identify things and collect the information from them. There are many types of sensors attached to objects to collect information such as RFID, 2-D barcode and sensors.. The information that is collected by these sensors can be about location, changes in the air, environment, motion, vibration, etc.

2.1.3.2 Network Layer

Network layer is also known as transmission layer. It transmits the information collected from the physical objects through sensors to the processing systems in the Middleware Layer through the transmission mediums who's can be wireless or wire based, with protocols like IPv4, IPv6,.. , It also takes the responsibility for connecting the smart things, network devices and networks to each other. Therefore, it is highly sensitive to attacks from the side of attackers.

2.1.3.3 Middleware Layer

middleware layer is also known as a The processing layer. It collects the information that is sent from a Network layer. It performs processing onto the collected information. It includes the technologies like Cloud computing, Ubiquitous computing which ensures a direct access to the database to store all the necessary information unit ,It has the responsibility to eliminate extra information that has no meaning and extracts the useful information.

2.1.3.4 Application Layer

Application layer denes all applications that use the IoT technology or in which IoT has deployed. It has the responsibility to provide the services to the applications. The services may be varying for each application because services depend on the information that is collected by sensors. So this layer is very helpful in the large scale development of IoT network.

2.1.3.5 Business Layer

The business layer refers to an intended behavior of an application and acts like a manager of the applications and services of IoT. It has responsibilities to manage and control applications, business and prots models of IoT. The user's privacy is also managed by this layer. It also has the ability to determine how information can be created, stored and changed.

2.1.4 Essential IoT Technologies

Five IoT technologies are widely for the deployment of successful IoT-based products and services [30] :

- Radio frequency identification (RFID).
- Wireless sensor networks (WSN).
- Middleware.
- Cloud computing.
- IoT application software.

2.1.4.1 Radio Frequency Identification

The term RFID includes all technologies that use radio waves to automatically identify objects or people. It's a technology that remembers and retrieves information remotely

thanks to a label that emits waves radio. This is a method used to transfer data from labels to objects, or to identify objects remotely.

2.1.4.2 Wireless Sensor Networks (WSN)

It is a set of nodes that communicate wirelessly and that are organized in a cooperative network. Each node has a processing capacity and can contain different types of memories, an RF transceiver and a power source, as it can also accommodate various sensors and actuators. As its name suggests, the WSN then constitutes a wireless sensor network that may be a technology necessary for the operation of the IoT.

2.1.4.3 Middleware

Middleware is software that serves as an interface between components of the IoT, making communication possible among elements that would not otherwise be capable. Often described as "software glue," middleware makes it easier for software developers to implement communication and input/output so that they can shift their focus to the specific purpose of their application.

2.1.4.4 Cloud Computing

Cloud computing is the delivery of different services through the Internet. These resources include tools and applications like data storage, servers, databases, networking, and software. Rather than keeping files on a proprietary hard drive or local storage device, cloud-based storage makes it possible to save them to a remote database. As long as an electronic device has access to the web, it has access to the data and the software programs to run it. Cloud computing is a popular option for people and businesses for a number of reasons including cost savings, increased productivity, speed and efficiency, performance, and security.

2.1.4.5 IoT Application Software

The IoT software is the applications that manages the system , addresses its key areas of networking and action via platforms, embedded systems, partner systems and middleware. Its tasks are data collection, device integration, real-time analysis, and the extension of applications and processes within the IoT network.

2.1.5 Application Area of Internet of Things (IoT) :

We see that the concept of the Internet of Things (IoT) is exploding as there is an increasing need in everyday life for smart objects which are able to make it easier to reach peoples' goals.

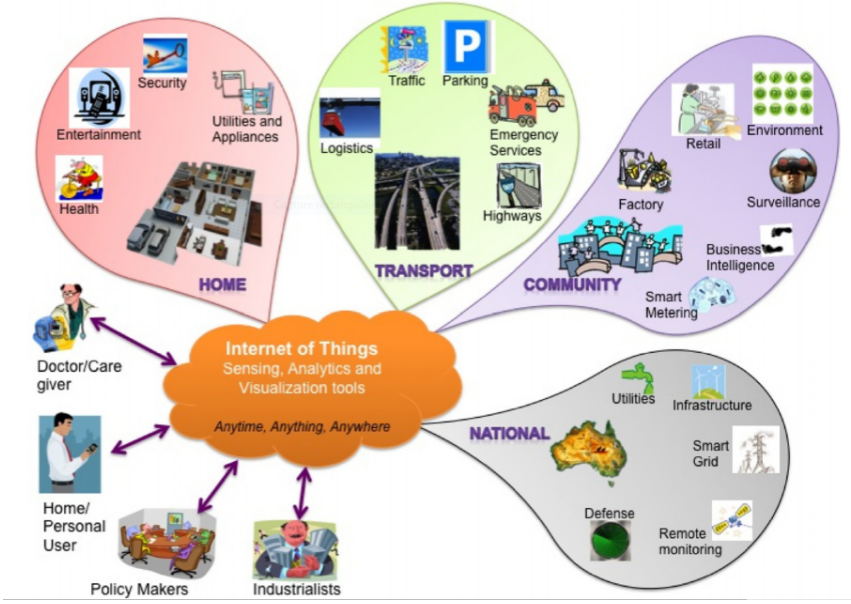

Figure 3: Application Area of IoT.[14]

IoT has multiple fields of application, as examples, industry, health, education and research like what's in Figure 3.

2.1.6 The Advantages and Disadvantages of IoT :

2.1.6.1 Advantages :

- **Communication :** Communication : Since IoT has communication between devices, in which physical devices are able to stay connected and hence the total transparency is available with lesser inefficiencies and greater quality.
- **Automation and Control :** Without human involvement, machines are automating and controlling vast amount of information, which leads faster and timely output.
- **Monitoring saves money and time :** Since IoT uses smart sensors to monitor various aspects in our daily life for various applications which saves money and time.
- **Better Quality of Life :** IoT based applications increases comfort and better management in our daily life; thereby improving the quality of life.

2.1.6.2 Disadvantages :

- **Compatibly :** As devices from different manufacturers will be interconnected in IoT, presently , there is no international standard of compatibility for the tagging and monitoring equipment.

- **Privacy/Security :** IoT has involvement of multiple devices and technologies and multiple companies will be monitoring it. Since lot of data related to the context will be transmitted by the smart sensors, there is a high risk of losing private data.

- **Lesser employment of menial staff:** With the advent of technology, daily activities are getting automated by using IoT with less human intervention, which in turn causes fewer requirements of human resources. This causes unemployment issue in the society.

- **Technology Takes Control of Life :** Our lives will be increasingly controlled by technology, and will be dependent on it. The younger generation is already addicted to technology for every little thing. With IoT, this dependency will spread amongst generations and in daily routines of users. We have to decide how much of our daily lives are we willing to mechanize and be controlled by technology.

2.2 Cyber Physical Systems (CPS) :

2.2.1 Introduction

In our time the world is witnessing an explosion in the computing and communication technology. Computing power and facilities are becoming more and more integrated into every aspect of our private and public lives. Computing devices can be found everywhere such workstations, laptops, smart phones and, smart home, etc.

2.2.2 Definition

Horizon 2020 (H2020) refers to CPS as "the next generation embedded ICT systems that are interconnected and collaborating providing citizens and businesses with a wide range of innovative applications and services.[15]"

The Commission's own Advisory Group - ISTAG1- considered CPS as "the evolution of embedded systems into smart objects that will be joined together to create highly distributed systems, bringing a wealth of opportunities and innovations in technology, applications and business models.[15]"

Finally, the ECSEL Joint Undertaking defines CPS as "embedded intelligent ICT systems that are interconnected, interdependent, collaborative, and autonomous. They provide computing and communication, monitoring/control of physical components/processes in various applications.[15]"

So Cyber-physical systems (CPS) are smart systems that include engineered interacting networks of physical and computational components.

2.2.3 Architectural of Cyber-Physical Systems :

This Figure 4 shows the architectural of a typical cyber-physical systems [9]:

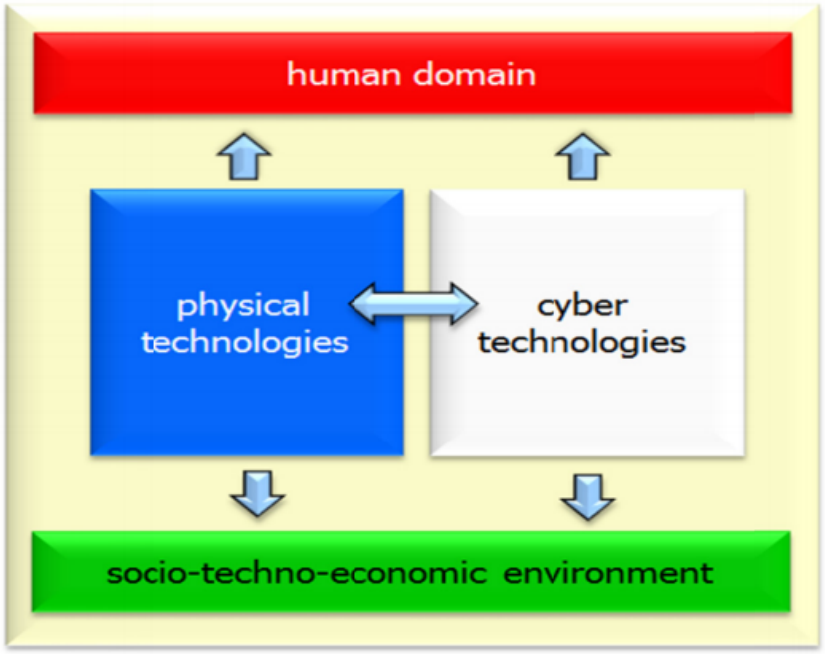

Figure 4: the standard architectural model of cyber-physical systems.

- **Physical technologies :** the physical technologies refers to all aspects of the hardware and its components, they operate in a continues time , and they are responsible for changing materials and energy flows.
- **Cyber technologies :** the Cyber technologies refers to all aspects of the software and its technologies of developments. They are responsible for computation , communication and control .
- **Socio-techno-economic environment :** Socio, techno and economic are the environments where the cyber physical systems in interaction with.
- **Human Domain :** the aim of the development of the cyber physical systems is the responding to the human needs in any domain.Socio, techno and economic are the environments where the cyber physical systems in interaction with.

2.2.4 Cyber Physical Systems Devices :

Cyber Physical Systems devices are the nonstandard computing devices that connect wirelessly to a network and have the ability to transmit data.

Cyber physical systems can be built by the Arduino , raspberry pi and the different sensors.

2.2.4.1 Arduino

Arduino is an electronics platform open-source, based on light-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, etc. And turn it into an output - activating a motor, turning on an LED, etc. You can control from the outputs by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language.[1]

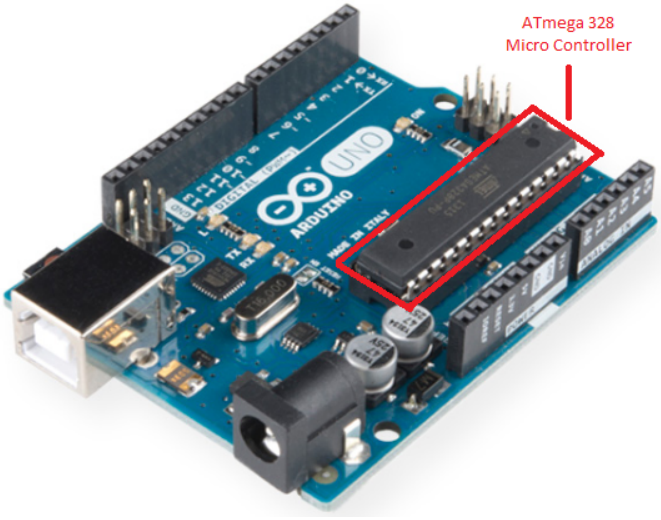

Figure 5: Labelled Arduino board.

The ATMega328 (Arduino's brain) who's in red in Figure 5 is an Integrated Circuit (IC) microcontroller. The flash memory of the chip contains the program necessary for the operation of the card.

2.2.4.2 Raspberry Pi

The Raspberry pi is more developed than Arduino , it is considered as a small computer that can be connected to a screen and used as a standard computer and uses a keyboard and mouse. Its cost is low and It is a capable little device that enables people of all ages to explore computing. It's capable of doing everything you'd expect a desktop computer to do.[23]

Figure 6: labelled raspberry pi board.

Raspberry Pi connects to a network via its Ethernet port or via Wi-Fi or USB adapters. The card is powered by a micro USB port as what is present in Figure 6. There is no hard drive: a microSD card is used as the storage medium. It contains the operating system, as well as all of your documents and programs.

2.2.4.3 Sensors

Sensor is a subsystem , module, whose purpose is to get information by detect events or changes in its environment and the sending to those information to other electronics, frequently a computer processor. A sensor can't work alone it's always used with other electronics.[25]

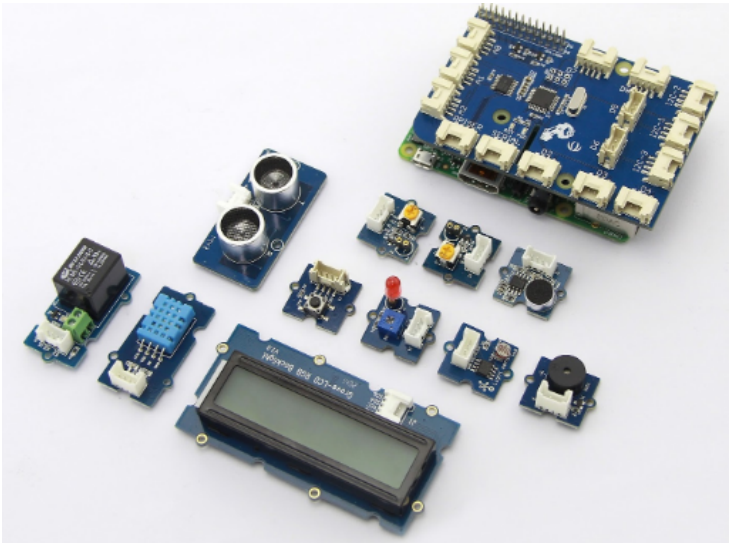

Figure 7: CPS's sensors.

Figure 7 shows the different types of sensors , and this difference is on the type of the captured information.

2.2.5 Application Area of Cyber Physical Systems (CPS):

Due to our needs to The cyber physical systems in our life because they use to help us and make our life more easy. So it spreads in all fields of our life, and those are some examples :

- Automobile Systems.
- Robotics Systems.
- Smart city and smart house
- Medical systems
- Transportation systems

2.3 Difference between Cyber-Physical Systems and Internet of Things

The differences between Cyber-Physical Systems and Internet of Things is that IoT makes more emphasis on connecting “things” towards connecting “everything” while that CPS put more attention on integrating computation, networking and physical systems[28].

All IoT devices are Cyber-Physical Systems, but CPSs are not necessarily connected to the Internet and thus, not necessarily IoT devices[4].

2.4 Conclusion

In this chapter, we have presented the different definitions of Internet of Things (IoT) and Cyber Physical Systems, their architecture, its characteristics and their areas of applications. The following chapter we will describe and write the approach proposed in this work which returns to the studied Problem Statement.

Chapter 3

Conception

3.1 Introduction :

After having presented the theoretical background of Internet of Things, Cloud Computing and Web services, we focus on this chapter to present the general and detailed architecture of a migrating solution to reuse cyber-physical system on the clouds using REST-based web services.

We will describe in this chapter the internal structure of the architecture of the proposed approach, and the overall and detailed conception.

3.2 Motivation

Nowadays, IoT is the technology of the moment, in view of lack of a standard the communication among various IoT devices. and because IoT devices are controlled by various end-user applications, the functionalities of IoT devices may be published as IoT services. IoT services are RESTful services that connect to IoT devices. The uniform interface of IoT services allows them to be integrated with existing applications. In this context, we propose to migrate IoT "stand-alone" application to a Service-based IoT. This solution allows :

- To leverage existing cyber physical systems in cloud-based systemeco-systems.
- The remotely access to the application and its data.
- The use of cloud resources for processing data.

3.3 General Structure of IoT-Application Source-Code

In general, we cam distinguish two type of methods in IoT-application source Code; The first type is Internet Methods and the second is External methods.

- **Internal Methods:**
  - An internal method is specialized in the set up of IoT devices, such as initialization, triggering,.. etc.
  - Their uses is within the IoT device (e.g. : init, setup, etc.)
  - Figure 1 depicts an example of an internal method.

```
def __init__(self, pin):

    GPIO.setup(pin, GPIO.OUT)
    self.pwm = GPIO.PWM(pin, 50)
    self.pwm.start(6)
```

Figure 1: Example of internal method.

- **External Methods:**
  - An external method considerate like an IoT device interface that can make exchanges with the cloud.
  - Some returned variables may represent the sensed data of sensor (e.g. : the light sensed, obstacle sensed, etc.)
  - It is possible to transform external method to an IoT service because the external method gives the possibility to end-users to control an IoT device or to obtain information from an IoT device
  - Figure 2 and Figure 3 depicts an example of an internal method.

```
def Move_On_To(Informtion):

    if information == Right:
        Turn_Left()
    elif information == Left:
        Turn_right()
```

Figure 2: Example of external method (1).

```
def Get_Light_State():
    Value = Get_Sensed_Light()
    return Value
```

Figure 3: Example of external method (2).

3.4 Approach for Migrating IoT Application to REST-Based API

3.4.1 General Architecture

In this approach, the transforming of IoT application to RESTful API is done automatically, there is a many tasks to do for getting our objective.

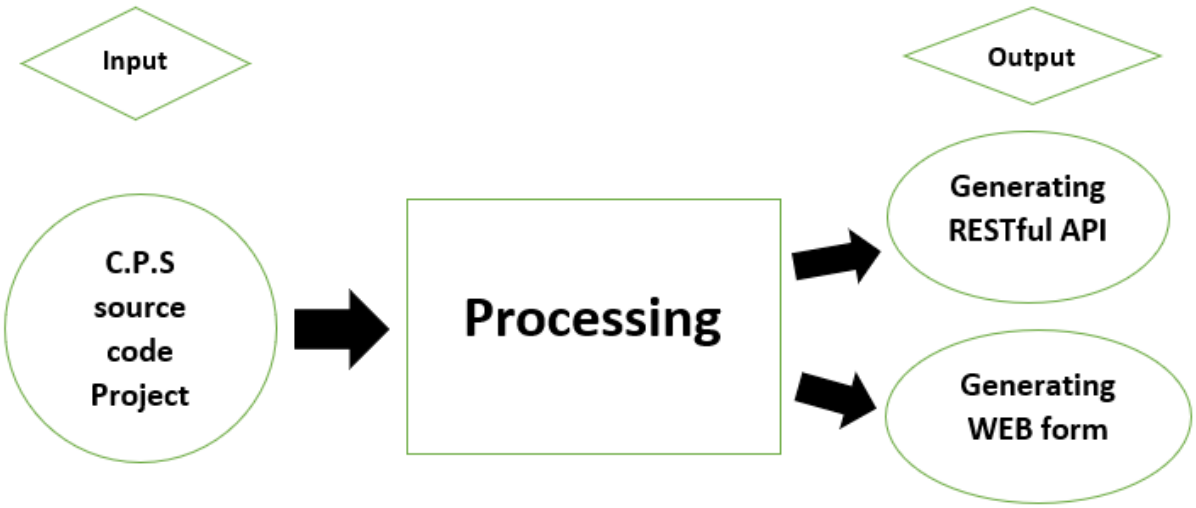

Figure 4: Input and Output of our approach.

Figure 4 shows the Input and the results of our approach after many tasks of processing :

- **Input :** We Start by a cyber-Physical System source code like an input to migrate it to REST API.
- **Processing:** We conduct many processing tasks on the input that we will demonstrate in the following sections.
- **Output:** There is two outputs; the first is the generation of the corresponding RESTful API, the second output is the generation of a web form to acquire data.

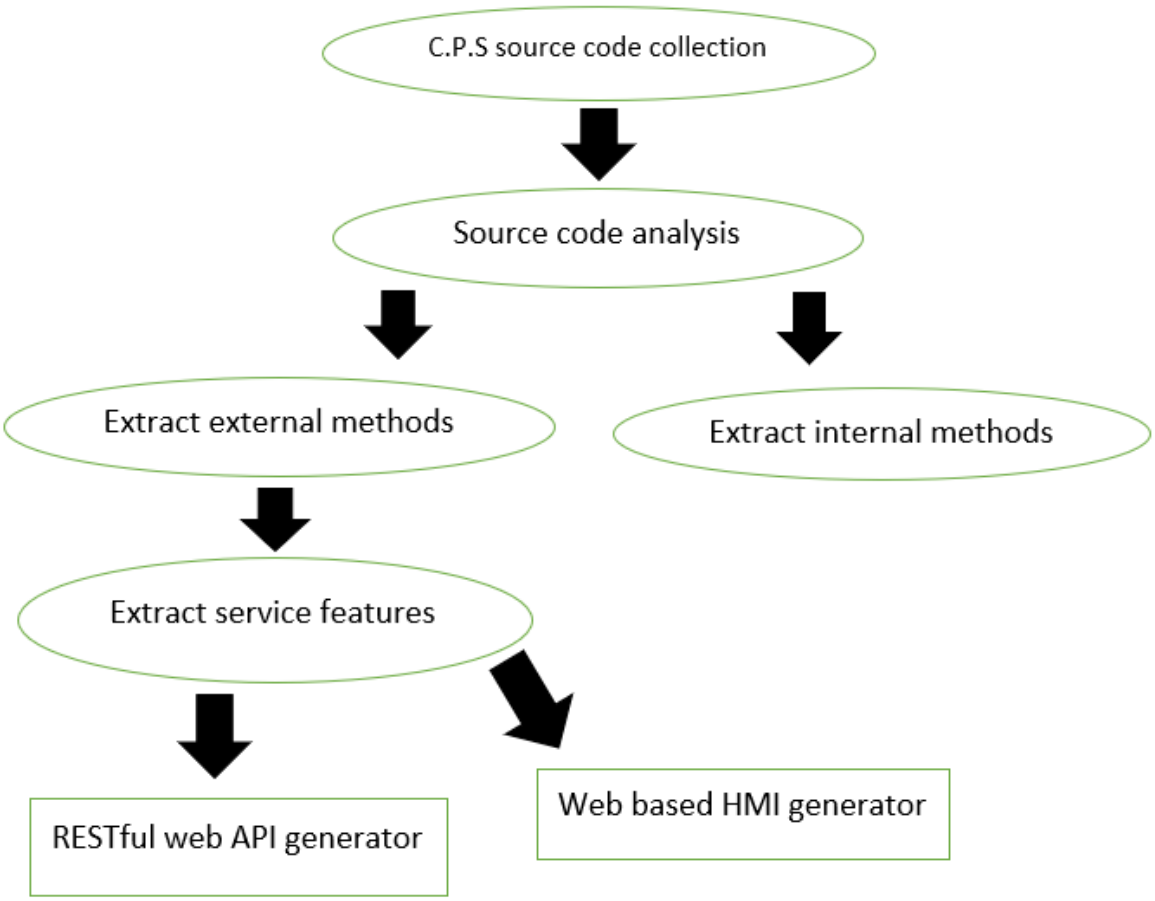

Figure 5: General Schema

Figure 5 presents the general architecture of our approach.

3.4.2 Collection of IoT Applications

Our approach is applied on Cyber.Physical.Systems applications programmed by python, since Python is considered as the smart and developed programming language especially for artificial intelligence and IoT systems, and easy to learn syntax [22].
So we will collect the Cyber.Physical.Systems source code applications to test for, we find those applications in many websites specialize of the IoT development [3].

3.4.3 IoT Source Code Analysis

- Get the Abstract syntax tree (AST) [2] of the Cyber-Physical-System source code.

" The AST module helps applications to process trees of the used programming language abstract syntax grammar. this module helps to find out programmatically what the current grammar looks like."

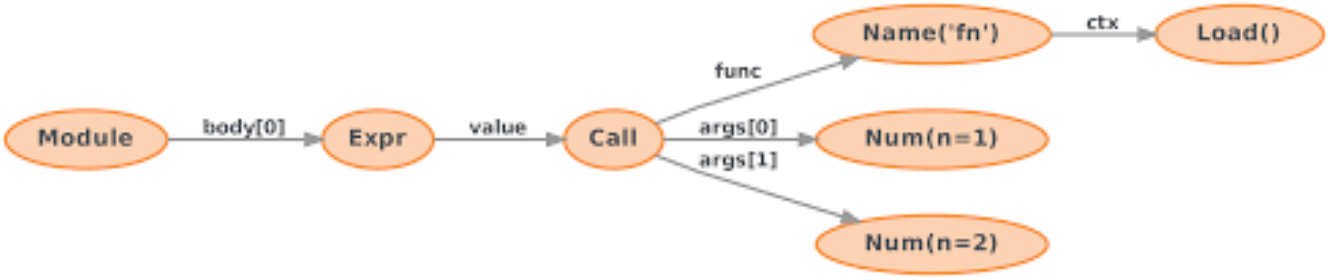

Figure 6: AST example

- Every node in the AST describes a construct from the constructs that are within the source code as in Figure 6.

- Analyze this AST to identify all nodes in methods to work on :

- **Method name :**Name of the method as it is in source code.
- **Input variables :**All the input variables of method.
- **Returned variable :**The returned variable if there is.
- **Method calls :** Methods that are used within the method.
- **If-else statement :** Tests within the methods.

3.4.4 Processing Method Names

We can distinguish between external and internal methods from words in their names, so we have to process all the methods names by :

- **We split Camel-Case words :** Example (GetData - > will be divided into Get and Data).
- **We remove punctuation :** Example (Get-Data - > Get and Data).
- **We remove numbers :** Example (GetData1 - > Get and Data)
- **We remove stop words :** Example (`Get_the_Data` - > Get and Data).

3.4.5 Extract Internal Methods

After the processing of method name we can easily extract internal method.
If the method name contains one of this key word “init, setup, debug, test ”,we can considered that it’s an internal method, because internal method specialize on the initialization, setup and test of IoT devices e.g.Figure1.

3.4.6 Extract External Methods

the extraction of external method is based on the testing about some features in methods whose obtained from IoT source code analysis and method name processing :

- **Condition 1 :** If the method name contains one word related to send, action or IoT service name (e.g. push, post, sense, publish, send, notify, subscribe, get, sense, set, receive, control, temperature, led, wireless..) , so this method considered as an external methods, because the method name denotes its objective

- **Condition 2 :** If method contains ”if-else statements” it means that it has a process task, and this is one of the external method features.

- **Condition 3 :** Methods calls within method body, if it has meaning of sending or action related words (e.g. post, publish, send..), and this is an external method feature.

In our approach if condition 1 is verified or the both of the condition 2 and condition 3 are verified, the method considered as an external method.

3.4.7 Extract Service Features

After the extraction of external methods that we will work with, we will try to get the services features corresponding to them :

- **Service name :** the service name is the method name (e.g.Figure 2 the service name is `Move_On_To`).

- **URL for access to the service :** since it’s RESTful services, the URL has to respect its syntax like the following (Figure 7) :

```
IoTdevice/<Device Id>/<Service Name>/<operation name>
```

Figure 7: URL schema of the service.

- **HTTP functions (Post or GET):**
  the choosing between HTTP functions it depending on methods feature that we've seen in (3.4.3):

  A – HTTP GET : It means that we will retrieve data (the service sends data), so if the method name and the method calls within, has send related words(e.g. "send, notify, Get, .."),and it has Return statement(e.g.Figure3).

  B – HTTP POST : Method name has receive related words (e.g. "set, receive,..") and the existing of If-else statements (e.g.Figure 2).

3.4.8 Generating Web Form based HTML

After extracting all the service features, and because the IoT device can be controlled by the end-user, the form that allows the end-user to control it, is generated automatically, each web form is depending on the code of the external methods corresponding.

Figure 8 is the HTML code of the web form that is used to control "the move side" by the end-user, that will sending POST request corresponds to the service extracted from the method in Figure 2.

```
<form action="http://localhost:5000/IoTdevice/b2d100/Services/
                        Move_On_To/information" method=POST>

   <label for="Move_On_To">Move_On_To : </label>
   <input type=radio name=information value=Left>Left
   <input type=radio name=information value=Right>Right
   <input type="submit" value="Submit"/>
</form>
```

Figure 8: Generated HTML code(1)

- **URL for access to service:** "http://localhost:5000/IoTdevice /b2d100/Services/`Move_On_To`/information".

- **HTTP method :**"POST".

- **Input type :**"Radio" because there is an if-else statements.

- **Parameter name :**"information" is the parameter of the method in Figure2.

- **Parameter value :** "Right" or "Left" is the value who will be sending.

Figure 9 is the HTML code of the web form that is used to control "the move side" by the end-user, that will sending POST request corresponds to the service extracted from the method in Figure 3.

```
<form action="http://localhost:5000/IoTdevice/b2d100/Services/
                          Get_Light_State/information" method=GET>

   <label for="Get_Light_State">Get_Light_State : </label>
   <input type=radio name=information value=Value>Value
   <input type="submit" value="Submit"/>
</form>
```

Figure 9: Generated HTML code(2)

- **URL for access to service:** "http://localhost:5000/IoTdevice /b2d100/Services/`Get_Light_State`/information".

- **HTTP method :**"GET".

- **Parameter value :** "Value" is the value which we need from the service

3.5 Generating the Corresponding RESTful API

The generation of service source code is automatically for the instantiating of the service after that.

The Figure 10 define the RESTful service source code corresponding to method in Figure 2.

```
@app.route('/iotdevice/b2d100/services/Move_On_To/information', methods=['POST'])
def add_statue():

    recieved_value = request.form['information']
    new_statue = ( recieved_value)
    Status.append( new_statue )
    return jsonify({'the value sending is : '})
```

Figure 10: RESTful API generator(1).

The data that is extracted from the method in Figure 2 :

Operation URL : "/iotdevice/b2d100/services/`Move_On_To`/information".

HTTP method : "POST".

Value received : Is the value of parameter "information" in HTML form.

- Type of data : the data transmitted follow JSON standard "jsonify".

The Figure 11 is the RESTful service source code corresponding to method in Figure 3.

```
Value = 'Must be initialized by the developer in first'

@app.route('/iotdevice/b2d100/services/Get_Light_State/information', methods=['GET'])
def Get_Light_State():


    return jsonify({'Return value' : Value})
```

Figure 11: RESTful API generator(2).

The data that is extracted from the method in Figure 3 :

Operation URL : "/iotdevice/b2d100/services/`Get_Light_State`/information".

HTTP method : "GET".

The getting value : Is "Value" which must be initialized by he developer in first.

- Type of data : the data transmitted follow JSON standard "jsonify".

3.6 Conclusion

In this chapter, we have presented an approach for the automatic transformation of IoT application to RESTful API, we started with the global architecture, then we described the different phases of the approach.

In the next chapter, we will present an implemented in Python which embodies our approach.

Chapter 4

Implementation

4.1 Introduction

The current chapter is dedicated to the presentation of the implemented prototype to migrate from legacy Cyber physical system to REST-Based system.

First we will present the languages and the development tools used during the implementation of the prototype. Then, we will describe the prototype itself, and we will end this chapter with a conclusion.

4.2 Software Tools Used

In this part we have listed the different software tools used throughout the programming phase

4.2.1 Development Environment

- **PyCharm**

  PyCharm is an Integrated Development Environment (IDE) used for programming in Python. PyCharm is developed by the Czech company JetBrains. It provides code analysis, a graphical debugger, an integrated unit tester, integration with version control systems (VCSes), and supports web development with Django as well as Data Science.

  It is cross-platform working on Windows, Mac OS X and Linux. PyCharm has a Professional Edition and a Community Edition.[20]

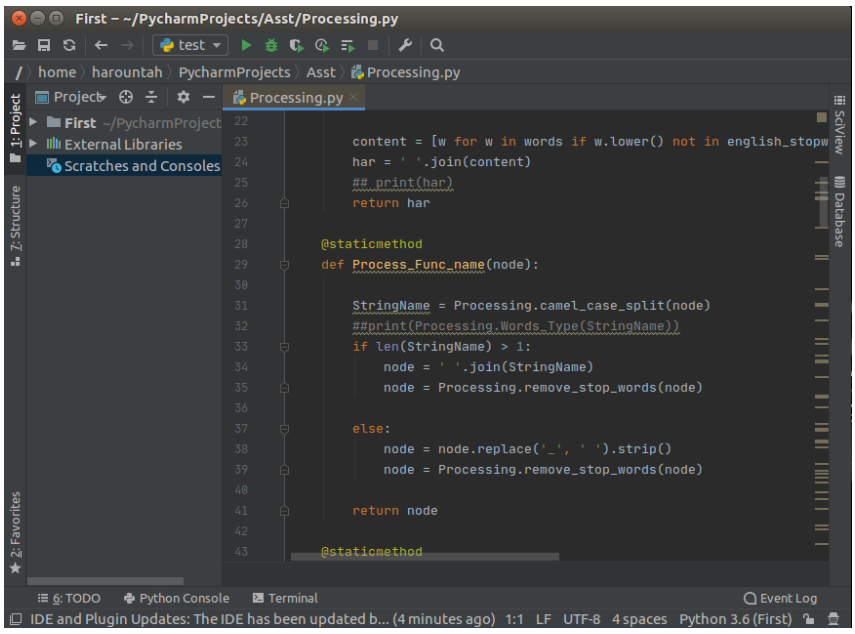

Figure 1: PyCharm IDE

Figure 1 shows a view of PyCharm IDE.

- **Sublime Text**

  Sublime Text is a shareware cross-platform source code editor with a Python application programming interface (API). This source code editor natively supports various programming and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses. [27]

- **LaTex**

  LaTeX is a software system for document editing and preparation. When writing, the writer uses plain text with commands. It is widely used in academia for the communication and publication of scientific documents in many fields, including mathematics, statistics, computer science .. etc

4.2.2 Programming Languages Used

- **Python**

  Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It is extremely attractive in the field of Rapid Application Development because it offers dynamic typing and dynamic binding options. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance and development because it allows teams to work collaboratively without significant language and experience barriers. Python supports modules and packages, which encourages program modularity and code reuse.
  The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.[19]

- **HTML**

  HTML is short for Hypertext Markup Language. HTML is used to create electronic documents (called pages) that are displayed on the World Wide Web. Each page contains a series of connections to other pages called hyperlinks. Every web page you see on the Internet is written using one version of HTML code or another. When multiple web pages are published with a common theme or within a common domain name, the collection is usually called a web site.

  HTML code ensures the proper formatting of text and images for your Internet browser. Without HTML, a browser would not know how to display text as elements or load images or other elements. HTML also provides a basic structure of the page, upon which Cascading Style Sheets are overlaid to change its appearance.[10]

4.2.3 Used Frameworks and APIs

- **Flask**

  Flask is an open-source Python Web framework, it provides tools, libraries and technologies that allow to build a web application.

- **PyQt5**

  PyQt is a GUI widgets toolkit. It is a Python interface for Qt, one of the most powerful, and popular cross-platform GUI library. PyQt is a blend of Python programming language and the Qt library [21].

- **Abstract Syntax Tree (AST)**

  AST is a way of representing the syntax of a programming language as a hierarchical tree-like structure where every node describes a construct from the constructs that are within the source code.

4.3 About the Migration Tool

The ultimate goal of the project is to build a tool who parse codes in python language and then generate a REST API (i.e. a set of invokable resources that corresponds to the external methods found in the original source code) and Web form that allows to end-user the control and the accessing to the service and its data. It is expected that this API do the same functionality as the original one with some deployment procedures.

4.4 Presentation of the Migration Tool

In this part we will present the main interfaces of our application :

4.4.1 Main Interface

The first window that will shown is Figure 2 :

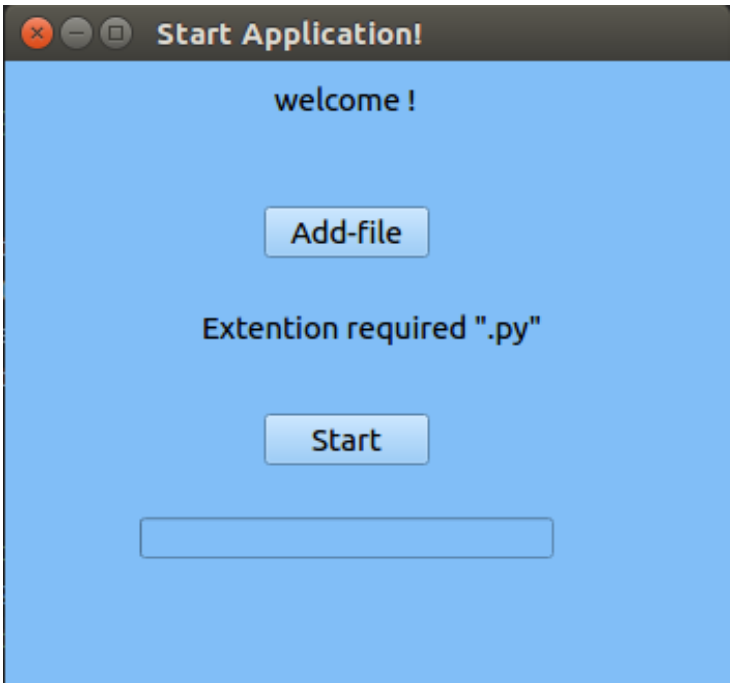

Figure 2: Main Interface

- By clicking on the start button without adding “file.py”, an error window will appear(See Figure 3 for illustration):

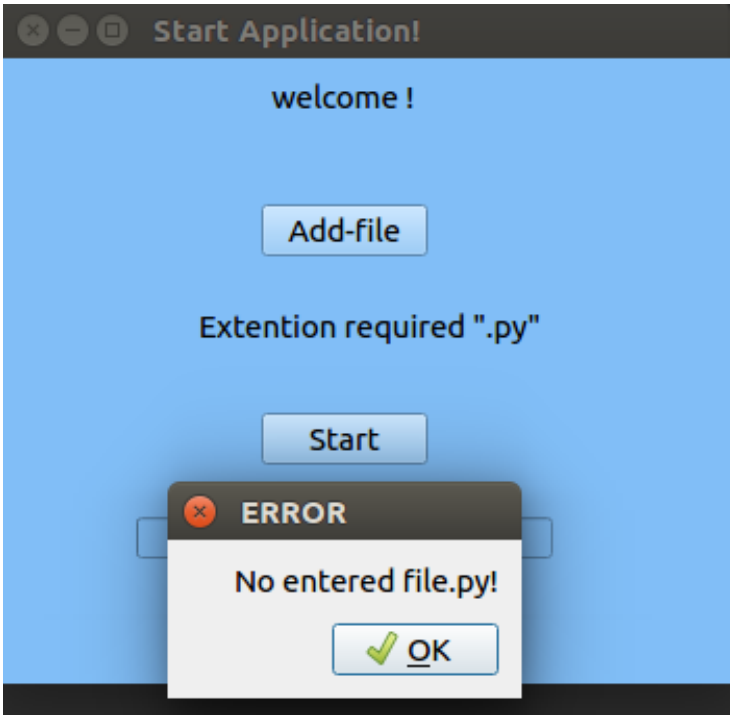

Figure 3: Error window.

- The clicking on add-file button allows to choose one file in our machine as below in Figure 4).

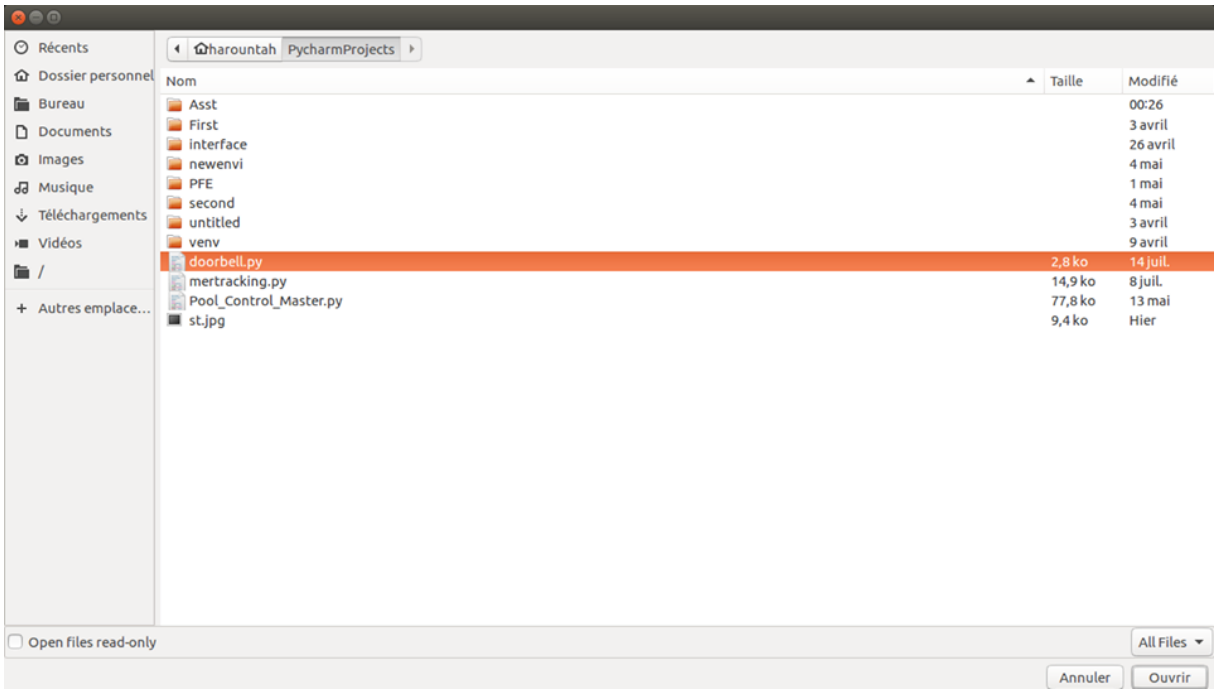

Figure 4: choosing “file.py” from machine..

- After choosing a file.py (source code of an CPS) than click on start bottom, the progressing bar appears Figure 5:

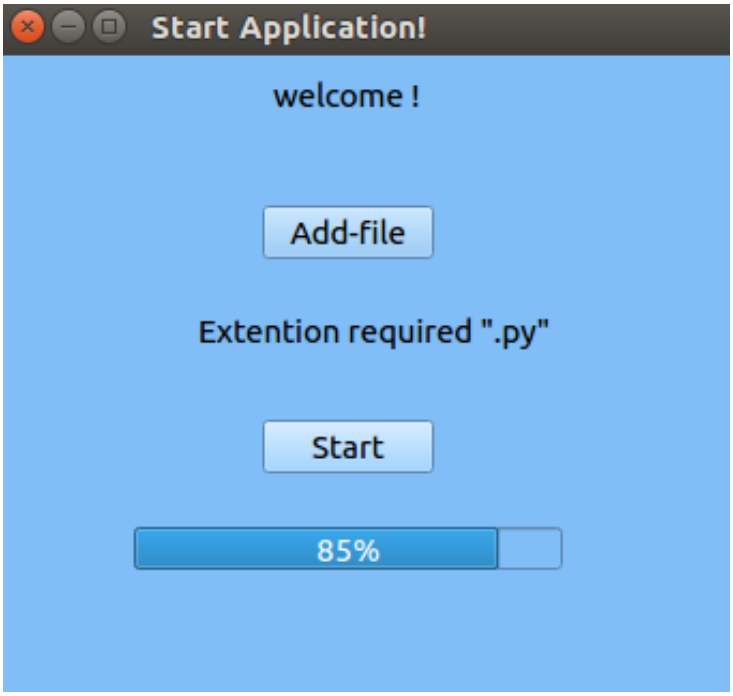

Figure 5: Progressing bar.

the progress bar will be progressing until the preparing of the results.

4.4.2 Results Interface

Figure 6 presents the first window result which holds the following components:

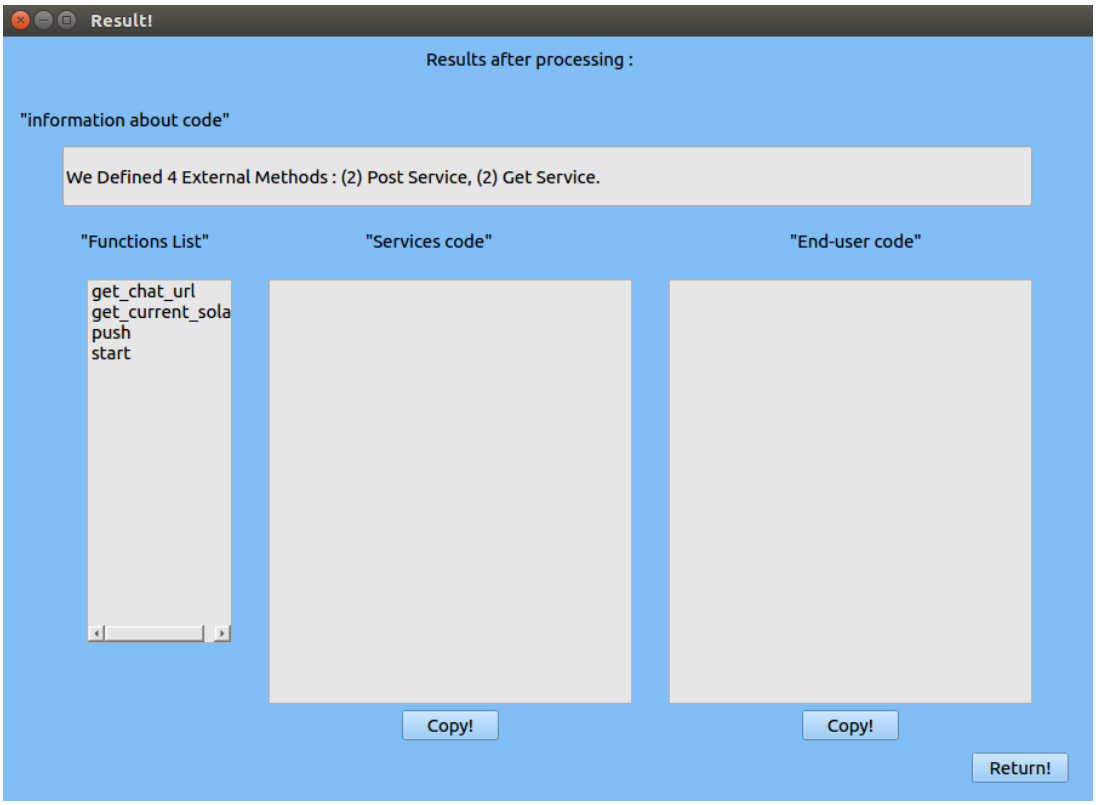

Figure 6: Result window.

- **Information about code :** In this area we will give information about the number of external methods extracting from the file entered and also the number of services types.

- **Function List :** This area contains the list of the external methods name, and we have to select one of the them to get its service code and end-user code.

- **Services code :** This area will contain the service code “PYTHON” of the function name selected from the function list.

- **End-User code :** This area contain the end-user code ”HTML” of the function name selected from the function list.

- **Return Button :** Clicking on “Return!” will take us to the start application window shown in Figure 2.

- Selecting of function name from function list gives this result (Figure 7 and Figure 8):

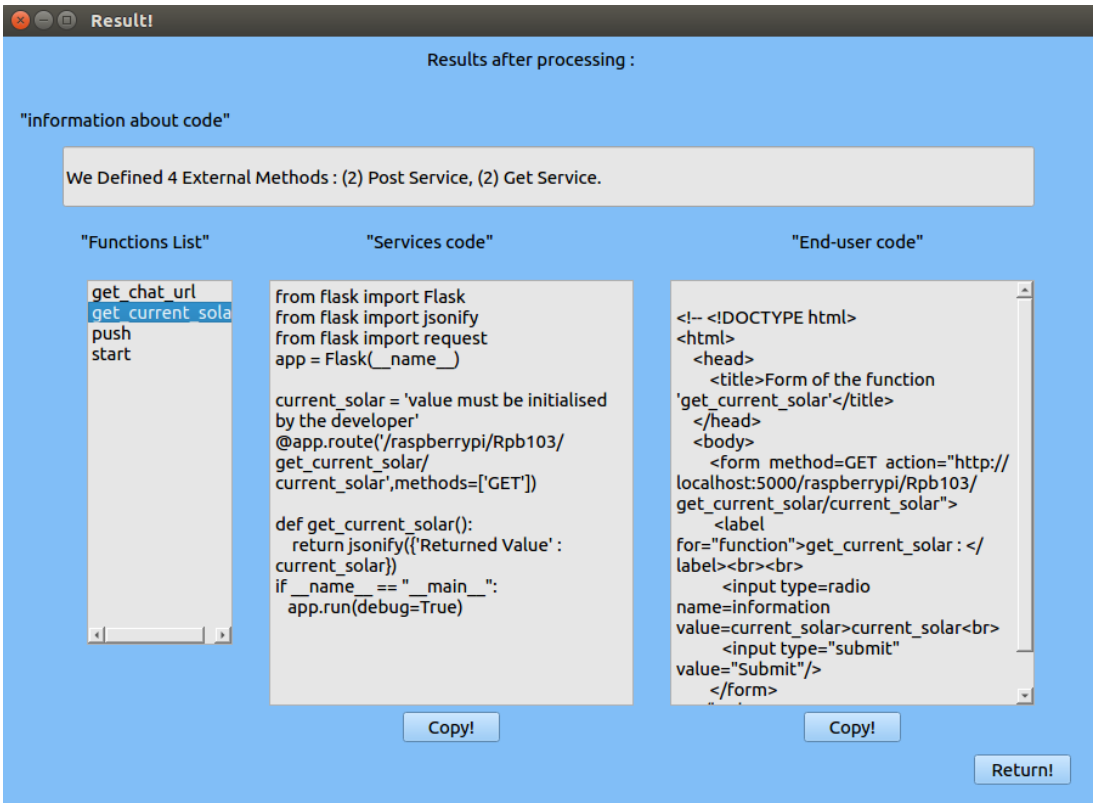

Figure 7: "get_current_solar"considered as "GET" service.

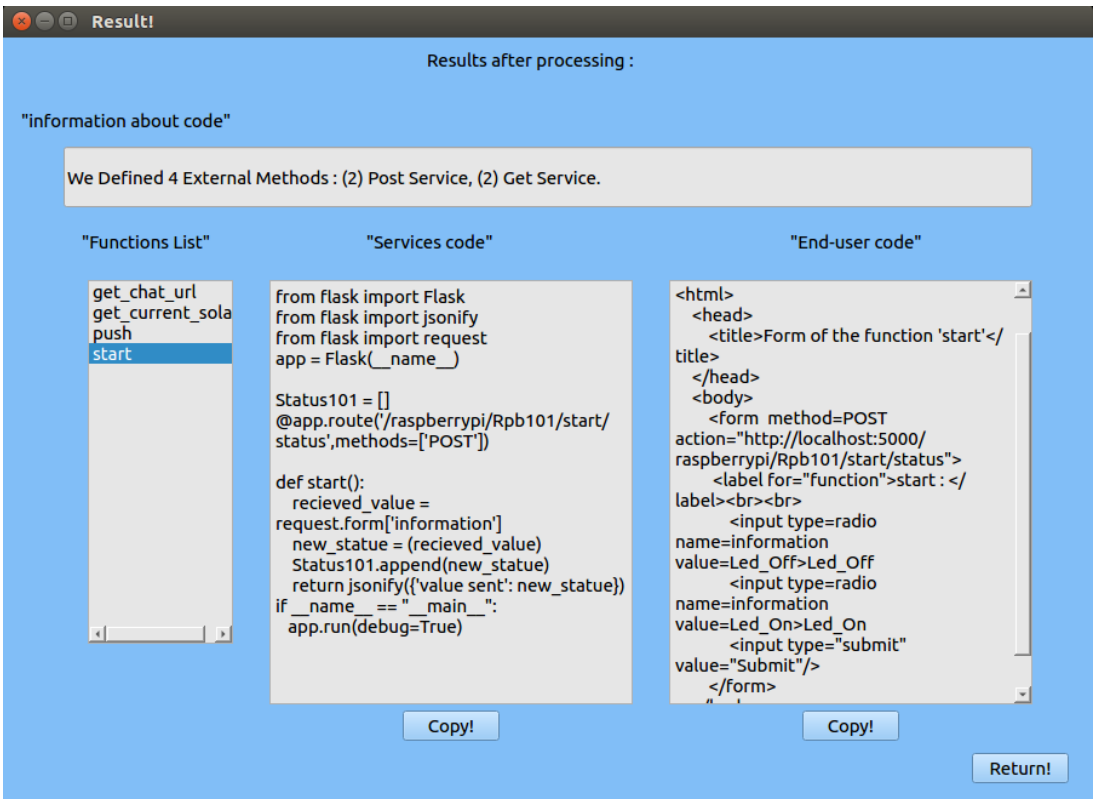

Figure 8: "star" considered as "POST" service.

4.4.3 Test of the Obtained Results

- **GET Request**

  First, we get a copy of services code by clicking on “Copy!” Button that situated down of the area (Figure 6) . next, creating new file ”service.py”, after that paste service code in “service.py”. And the same thing with End-user code but we paste it in new file who’s name “end-user.html” as in Figure 9 Figure 10.

Service.py EndUser.html

```
1  from flask import Flask
2  from flask import jsonify
3  from flask import request
4  app = Flask(__name__)
5
6  current_solar = 'initialized by developer'
7  @app.route('/raspberrypi/Rpb101/get_current_solar/current_solar',methods=['GET'])
8
9  def get_current_solar():
10     return jsonify({'Returned Value' : current_solar})
11 if __name__ == "__main__":
12     app.run(debug=True)
```

Figure 9: “GET” service code.

Service.py EndUser.html

```
1  <!DOCTYPE html>
2  <html>
3      <head>
4          <title>Form of the function 'get_current_solar'</title>
5      </head>
6      <body>
7          <form  method=GET  action="http://localhost:5000/raspberrypi/Rpb101/get_
8           <label for="function">get_current_solar : </label><br><br>
9             <input type=radio name=information value=current_solar>current_solar
10            <input type="submit" value="Submit"/><br>
11         </form>
12     </body>
13 </html>
```

Figure 10: EndUser code for “GET” service.

- Now we execute the “service.py” to run the RESTful service, Figure 11 presents that :

Figure 11: Run the REST service(1).

- By clicking on “EndUser.html” file, this form will appear (Figure 12) on the browser:

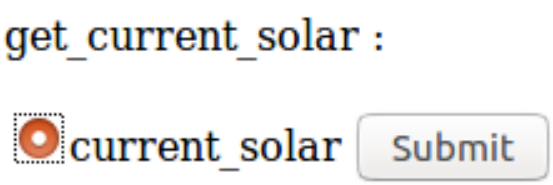

Figure 12: ”GET” End-user form.

- After choosing a checkbox and clicking on ”submit” button, A ”GET” request will sending to the ”REST API” :

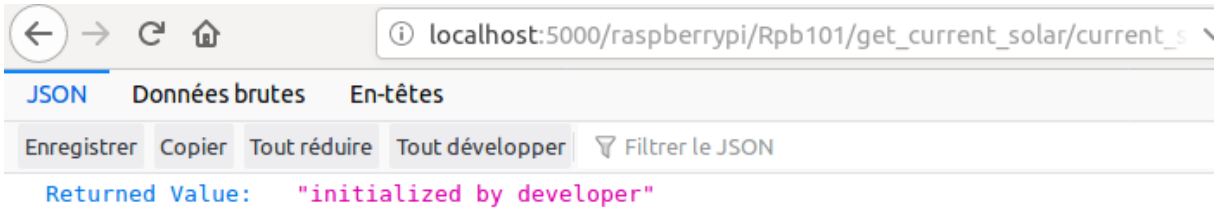

Figure 13: REST Result of “GET” request.

Figure 13 presents the result of GET request, for the returned value must be initialized by the developer in first use of the service.

- **POST Request**

  First, we get a copy of services code by clicking on “Copy!” Button that situated down of the area (Figure 6) . next, creating new file ”service.py” after that paste service code in “service.py”. And the same thing with End-user code but we paste it in new file who’s name “end-user.html”as in Figure 14 Figure 15.

Service.py EndUser.html

```python
from flask import Flask
from flask import jsonify
from flask import request
app = Flask(__name__)

Status101 = []
@app.route('/raspberrypi/Rpb101/start/status',methods=['POST'])

def start():
    recieved_value = request.form['information']
    new_statue = (recieved_value)
    Status101.append(new_statue)
    return jsonify({'value sent': new_statue})
if __name__ == "__main__":
    app.run(debug=True)
```

Figure 14: “POST” service code.

Service.py EndUser.html

```html
<!DOCTYPE html>
<html>
    <head>
        <title>Form of the function 'start'</title>
    </head>
    <body>
        <form  method=POST  action="http://localhost:5000/raspberrypi/Rpb101/start/status">
         <label for="function">start : </label><br><br>
            <input type=radio name=information value=Led_Off>Led_Off
            <input type=radio name=information value=Led_On>Led_On
            <input type="submit" value="Submit"/>
        </form>
    </body>
</html>
```

Figure 15: EndUser code for “POST” service.

- Now we execute the “service.py” to run the REST service, Figure 16 presents that :

Figure 16: Run the REST service(2).

- Get the end-user form by clicking on “EndUser.html”, this form will appear (Figure 17) on the browser:

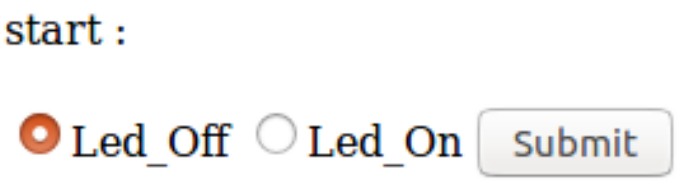

Figure 17: ”POST” End-user form.

- After choosing a checkbox and clicking on ”submit” button, ”POST” request will sending to the ”REST API” :

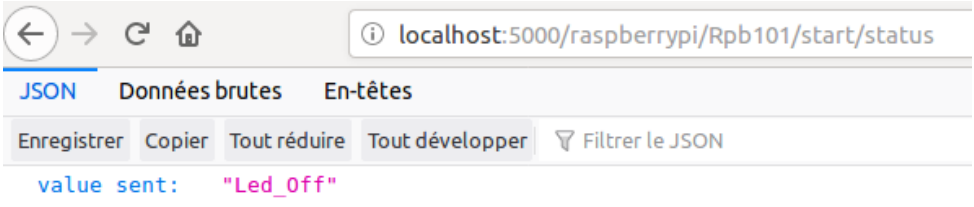

Figure 18: REST Result of “POST” request.

Figure 18 presents the result of POST request, after storing the data sending, the service will show you the sending data.

4.5 Conclusion

In this chapter we have presented all the steps of the implementation of our prototype, with all the tools and programming languages used.

The next section is an overall conclusion that encompasses everything we have done to achieve our goal.

Conclusion

So far, we have presented an approach for migrating local physical system applications to RESTful-based web service on the cloud. The manuscript was organized in four chapters; Where we presented the two research areas, Web services and Cloud computing in the first chapter. The objective was to cover all the concepts that are related to web services and the cloud. Then, in the second chapter, we made an overview of the two technologies Internet of Things (IoT) and Cyber Physical Systems (CPS) to get an idea about their principles and characteristics. In the third chapter we explained our approach that is a migration tool of Cyber Physical Systems to REST-based web service on the cloud. Our approach allows the end user to control and access to the IoT device through a REST API. The last chapter in this thesis was devoted for the presentation of the elaborated tool.

Bibliography

[1] *Arduino*. `https://www.arduino.cc/`. Accessed: 2020-01-10.

[2] *AST-Python*. `https://docs.python.org/2/library/ast.html`. Accessed: 2019-12-25.

[3] *CPS projects*. `https://www.hackster.io/`. Accessed: 2020-03-10.

[4] Ivan De Oliveira Nunes. *What is the difference between Internet of Things (IoT) and Cyber Physical Systems (CPS)?* June 2015.

[5] Mark Endrei et al. *Patterns: service-oriented architecture and web services*. IBM Corporation, International Technical Support Organization New York, NY, 2004.

[6] Borivoje Furht and Armando Escalante. *Handbook of cloud computing*. Vol. 3. Springer, 2010.

[7] Jayavardhana Gubbi et al. "Internet of Things (IoT): A vision, architectural elements, and future directions". In: *Future generation computer systems* 29.7 (2013), pp. 1645–1660.

[8] Qusay F Hassan. *Internet of things A to Z: technologies and applications*. John Wiley & Sons, 2018.

[9] Imre Horvath and Bart HM Gerritsen. "Cyber-physical systems: Concepts, technologies and implementation principles". In: *Proceedings of TMCE*. Vol. 1. 2. 2012, pp. 7–11.

[10] *HTML*. `https://www.computerhope.com/jargon/h/html.htm`. Accessed: 2020-07-03.

[11] Norihisa Komoda. "Service oriented architecture (SOA) in industrial systems". In: *2006 4th IEEE international conference on industrial informatics*. IEEE. 2006, pp. 1–5.

[12] James P Lawler and Hortense Howell-Barber. *Service-oriented architecture: SOA strategy, methodology, and technology*. CRC Press, 2007.

[13] Peter Mell, Tim Grance, et al. "The NIST definition of cloud computing". In: (2011).

[14] Andreas Metzger et al. “Cyber physical systems: Opportunities and challenges for software, services, cloud and data”. In: *NESSI White Paper* (2015).

[15] Andreas Metzger et al. “Cyber physical systems: Opportunities and challenges for software, services, cloud and data”. In: *NESSI White Paper* (2015).

[16] Andy Neumann, Nuno Laranjeiro, and Jorge Bernardino. “An analysis of public REST web service APIs”. In: *IEEE Transactions on Services Computing* (2018).

[17] Keyur K Patel, Sunil M Patel, et al. “Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges”. In: *International journal of engineering science and computing* 6.5 (2016).

[18] Cesare Pautasso. “RESTful Web service composition with BPEL for REST”. In: *Data & Knowledge Engineering* 68.9 (2009), pp. 851–866.

[19] *Programing language: Python*. `https://www.python.org/doc/essays/blurb/`. Accessed: 2020-06-20.

[20] *pycharm*. `https://www.jetbrains.com/help/pycharm/`. Accessed: 2019-12-25.

[21] *PyQt5*. `https://www.tutorialspoint.com/pyqt/index.htm`. Accessed: 2019-12-25.

[22] *python code*. `https://docs.python.org/3.8/tutorial/`. Accessed: 2020-02-20.

[23] *Raspberrypi*. `https://www.raspberrypi.org/`. Accessed: 2020-01-10.

[24] *REST*. `https://fr.wikipedia.org/wiki/Representational_state_transfer`. Accessed: 2020-02-15.

[25] *Sensors*. `https://en.wikipedia.org/wiki/Sensor`. Accessed: 2020-02-15.

[26] *SOAP*. `:https://www.guru99.com/soap-simple-object-access-protocol.html`. Accessed: 2020-01-02.

[27] *sublimetext*. `https://www.sublimetext.com/`. Accessed: 2020-05-15.

[28] Lihui Wang and Xi Wang. “Latest Advancement in CPS and IoT Applications”. In: Jan. 2018, pp. 33–61. ISBN: 978-3-319-67692-0. DOI: `10.1007/978-3-319-67693-7_2`.

[29] *Web services*. `https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/bpm-web/web-services-standards.html`. Accessed: 2020-01-02.

[30] Feng Xia et al. “Internet of things”. In: *International journal of communication systems* 25.9 (2012), p. 1101.