MSE | MASTER OF SCIENCE IN ENGINEERING

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz
University of Applied Sciences and Arts Western Switzerland

Master of Science HES-SO in Engineering
Av. de Provence 6
CH-1007 Lausanne

Master of Science HES-SO in Engineering

Orientation: Information and Communication Technologies (ICT)

APDL : Acquire Process Description Language

Author:
Sylvain Julmy

Under the direction of:
Pierre-André Mudry
Institut Systèmes Industriels

External expert:
??

Lausanne, HES-SO//Master, March 21, 2017

Info page

Part I

Introduction

Chapter 1

Introduction

The Internet of Things Internet of Things (IoT) is one of the most prolific domains in computer science nowadays. There is connected object everywhere and the amount of data that we receive from them is growing day after day. Everybody wants to measure some environment variable using a simple and cheap device like an Arduino or a Raspberry Pi.

In another hand, the people who want to use those devices are often not familiar with the hardware and/or the software parts of an IoT project. For example, a chemistry engineer could be interested to measure some value in the air, but he is absolutely not in the IoT domain.

1.1 Context

Imagine an electrical engineer who needs to create a system which is able to recover data from multiple sensors, like external temperature and barometric pressure. With that data, the engineer need to compute a new pressure value with the temperature and compare it to the one give by the barometric pressure sensor using graph plot.

The problem for this electrical engineer is the software part. He doesn't know how to send data through the network, recover them and store them into a database for a future plot. In another hand, if it's a software engineer to do this, he would not know the hardware part.

IoT programming is at some point a merge of some specific domain like Software, Hardware, Telecommunication and Design. The figures 1.1 illustrate this concept of multi-domain project. Are involve in this concept:

- Hardware
- Software
- Design
- Telecommunication

1.2 Problem Statement

According to the previous example we could say that not everybody could develop a project like this from scratch. Maybe some people have the whole knowledge to do this by themselves but

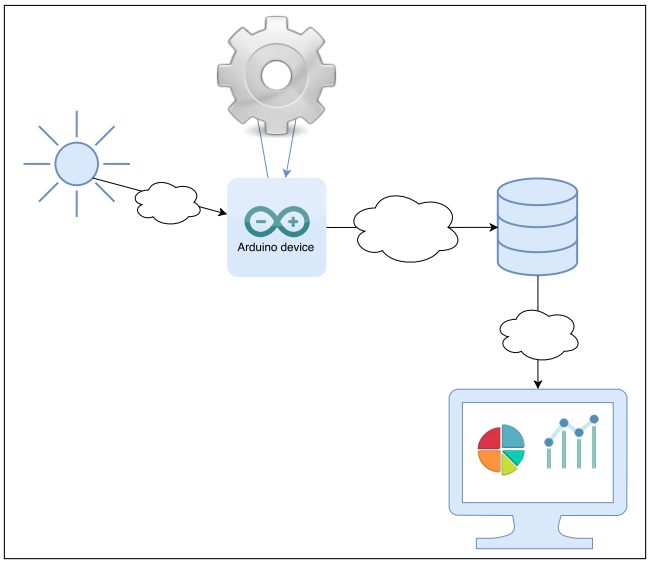

Figure 1.1: Visualisation of all the domain involve in the APDL ecosystem. There are hardware, software, design and telecommunication. And sometimes, a person doesn't know any part of this.

not a lambda person.

The big deal is that there is no way to simply create that without involving multiple people from different domains and bring it together, and that creates another difficulty: team management, different programming knowledge, the same understanding of the project, and many more.

1.3 Solution Statement

The Acquire and Process Description Language Acquire and Process Description Language (APDL) ecosystem goal is to provide a simple way to describe such a pipeline. From the sampling of the data from a sensor and his transformation to the display of the result of charts by storing them into a database or send them to a specific server.

1.4 Roadmap

TODO

Part II

A Domain specific language for the Internet of Things

Chapter 2

State of the art

IoT is a huge area of research and development over the few past year, developer and researcher in this field face some serious challenge [1][2][3]:

- Device diversity (Arduino, Raspberry PI, physical sensors, biometric sensors)
- Software implementation (Linux, iOS, Windows, Android)
- Interactive modes (publisher/subscriber pattern, request/response pattern, command pattern, pull/push pattern)
- Engineering unit (degree Farenheit/degree Celsius, foot/metre, and many more)
- Power management and optimisation
- Connectivity, the device is connected with various protocols like ZigBee or MQTT
- Reliability, the device could be instantly checked and have some error handling application level
- Data description, we need to annotate the produced data (raw data are a problem)
- Various security problems:
  - Physical security
  - Data exchange security
  - Cloud storage security
  - Update and patch

In this part, a state of the art and some various approaches for the IOT programming, development and design are presented. We will see some programming language which one could be the target of the DSL we are creating and some existing domain specific language. In the end, we fully describe the challenge of the IOT in order to offer a language which is able to answer all the developers wishes.

2.1 Programming Language for the Internet of Things

Using a programming language to create IOT Applications offer full control to the developer, but it comes to the handling of the challenge too. By the way, some language needs to embed

interpreter and/or virtual machine in order to work. This adds some additional memory, power and processor use.

2.1.1 C/C++

C and C++ are available for compilation on almost every platform, it's a hardware focused language which is not too complex and offers great speed and memory management. The very big advantages is that C and C++ are compiled, they don't need a runtime management and compiled binaries are very small for limited memory devices. From another point of view, the programmer needs to do almost everything by itself: memory management, pointer arithmetic error handling and, if we use C, there is no standard library for data structure and algorithm.

Notice that the Arduino language is a subset of C and owns the same advantages and problems with memory management.

Therefore, C and C++ are really good when it comes to a small device with very little memory and when no operating system are available.

2.1.2 Python

Python is a programming language created in 1990 by Guido van Rossum, it's an object-oriented, imperative and interpreted one. Because Python is interpreted, the language needs to run with a virtual machine.

By the way, a tool named Cython[4] is able to compile, after some work, a Python code to a C executable, which removes the interpreter and virtual machine needed at the origin.

We can go further, some embedded system for IOT does not run an operating system (the Arduino for example) and Cython need to have access to some operating system features. But there is some work about the possibility to run Python without any OS[5].

On the other hand, Python owns a great community with tons of tutorials and library for the IOT programming and especially with the Raspberry PI.

2.1.3 Java

C and C++ are great programming language for the IOT, it becomes easier to manage and control the hardware with that language, but they are hardware specific too. We can't write a program and run it into every possible hardware. Maybe a device is running on Linux and another one is running on Windows. The probably best answer to this problem is Java[6], which supports multiple hardware design through his virtual machine if the IOT project run needs to run on different platforms.

2.1.4 Another Programming Languages

There are many programming languages available for IOT development, we don't want to describe them all but we want a small overview of the field for the development of the DSL. We could talk a lot about language like Go, Rust and Javascript, which could be very good for a specific part of an IOT project.

We don't need to know all the usable programming language for the IOT, but we need a target for our DSL and those languages advantages and disadvantages would be considering for further work.

2.2 Domain specific language for the Internet of Things

Research and work has been done during the past year on the domain specific language for the Internet of Things. Programming for the IOT could be really anoying for the software developer who are not familliar with the hardware oriented concept or for the electrical enginner which are not software oriented. We need to know both soft and hard part of the project in order to realise it.

We present an overview of some research and technologies around the DSL created for the IOT. Notice that when the DSL is a visual one, we call it a VDSML : Visual Domain Specific Modeling Language.

2.2.1 Node-red

Node-red[7] is a tool, develop with Node.js and browser focused, for wiring the internet of things together. It is a purely VDSML without, practicly, no code to write. Node-red has compatibility with a lot of popular platforms[7] :

- Raspberry Pi
- BeagleBone Black
- Android
- Arduino
- Docker
- Amazon AWS
- IBM Bluemix
- Microsoft Azure

But we can go further, Node-red provides an editor, builtin function, a full compatibility with Javascript and the Node.js packages and finally a way to save and share graph information of the project with JSON.

Node-red is capable of simply create a complete pipeline from our diagram with just a mouse click. Using block for programming and connect some basic features do not require a high knowledge level.

Example

2.2.2 DSL-4-IoT

2.2.3 DiaSuite

2.2.4 PervML

2.2.5 OpenHab

2.2.6 LogicIOT

2.2.7 ArduinoML

2.3 Internet of Things programming challenge

Power supply, power-optimized algorithm, network protocols

Dynamic power management (DPM), to shud down device when they don't need to be online

DPM –> problem because of the latency and latency is an overall problem

Reliability: device could be constantely checked and Error handling procedure in the application level

Data curation and data brokering: device could produce a huge ammount of data

Data projection

Data description: metadata to describe raw-data

Declarative programming for the IOT !!!

Chapter 3

A Domain specific language

A Domain specific language Domain specific language (Dsl) is a programming language whose the goal is to provide a way to solve specific domain problem. For example, Matlab is a domain specific language because, in the first place, it was for matrix manipulation and mathematic.

What we call “domain” is a set of problem related together in the same scope. For Matlab, those domain could be a matrix multiplication or the resolution of a equantion system. Those to operation are related to numerical mathematic and that is the “domain” of Matlab.

This chapter will introduce and discuss the question “Why use a DSL ?” by exposing some advantages that APDL could bring for non-IoT-developers. Then we present the diffenrence between an internal and an external DSL and the advantages and disadvantages for both of them. Then we will conclude this chapter by exposing the development phase of a DSL describe by Van Deursen and Klint [8].

3.1 Why use a Dsl ?

3.2 Internal Dsl vs. External Dsl

3.3 Advantages and Disadvantages of a Dsl

3.4 Dsl Development phases

Chapter 4

Design of the DSL

As mentionned by Van Deursen and Klint [8], one point is to cluster the knowledge we acquire into a set of semantic notions and opertions on them. This chapter will present the whole design of the APDL DSL. Firstly, we analyse all the concept we need to have for a complete DSL design for our case-study. Those concept include the data source, the transmission, the storage and finally the visualisation of the data.

Then, we will present the design of the library which is implementing the clustered information.

4.1 Data source

The concept of data source include some different information :

- The source device
- The type of data
- The sampling

4.1.1 Source device

Firstly, we have to know that there is a lot of different existing device for the IoT programming, some of them are just connected ones like bluethoot tracker or smart device like fridge or door. For the purpose of the project we are just going the take device that are sensor oriented. All we want is to gather information for the environmment and communicate it to whatever we want.

TODO : write about arduino and raspeberry

Chapter 5

Implementation

Chapter 6

Test

Bibliography

[1] P. Biljanovic, Z. Butkovic, K. Skala, B. Mikac, M. Cicin-Sain, V. Sruk, S. Ribaric, S. Gros, B. Vrdoljak, M. Mauher, A. Sokolic, Eds., 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015, Opatija, Croatia, May 25-29, 2015, IEEE, **2015**.

[2] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, *Future Generation Computer Systems* **2013**, *29*, Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services ; Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond, 1645–1660, DOI http://dx.doi.org/10.1016/j.future.2013.01.010.

[3] B. Dickson, 4 Challenges to Consider Before Creating an IoT Device, **2016**, https://www.sitepoint.com/4-challenges-to-consider-before-creating-an-iot-device/.

[4] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. Seljebotn, K. Smith, *Computing in Science Engineering* **2011**, *13*, 31–39, DOI 10.1109/MCSE.2010.118.

[5] J. Edge, Python without an operating system, **2015**, https://lwn.net/Articles/641244/.

[6] P. Waher, *Learning Internet of Things*, Packt Publishing, **2015**.

[7] D. C.-J. IBM Emerging Technology, Nick O'Leary, Node-RED, **2017**, http://nodered.org/.

[8] A. van Deursen, P. Klint, *Journal of Software Maintenance* **1998**, *10*, 75–92, DOI 10.1002/(SICI)1096-908X(199803/04)10:2<75::AID-SMR168>3.0.CO;2-5.