Extension to Middleware for IoT Devices, with Applications in Smart Cities

Christos Bouras[1,2](✉), Vaggelis Kapoulas[1,2], Vasileios Kokkinos[1,2], Dimitris Leonardos[3], Costas Pipilas[3], and Nikolaos Papachristos[3]

[1] Computer Technology Institute and Press "Diophantus", Patras, Greece
{bouras,kapoulas,kokkinos}@cti.gr
[2] Department of Computer Engineering and Informatics, University of Patras, Patras, Greece
[3] ECONAIS, Patras, Greece
{dleonardos,cpipilas,nikolas}@wubby.io

Abstract. This work proposes extensions to Wubby (a device-level software platform for IoT devices, a technology developed by Econais A.E.) to support wireless modules for mobile networks (4 G / LTE-A, and also supporting the forthcoming 5 G). The proposed extension leverages the use of such modules, as it allows easy programming and existing code re-use. It thus adds a compatibility layer across the different modules as it a common set of classes for the wireless modules. The system can be used to support the networking aspects of a variety of IoT applications, including applications for Smart Cities, using a variety of IoT devices. This work suggests such a case focusing on air quality monitoring.

Keywords: Wireless modules · Middleware · Python · Internet of Things · IoT

1 Introduction

The ever increasing interest in the Internet of Things (IoT) and its immense growth over the last years [1–3], has led to the implementation of various computing devices of very small size (intended to be incorporated into various 'smart' objects), as well as numerous modules intended to enhance the functionality of these devices.

An important type of these kind of modules is the one for wireless network connectivity modules. With the technological progress in wireless communication already be in its 4th generation (4 G / LTE-A) of cellular networks and directed towards the 5th generation (5 G) of wireless networking, respective wireless networking modules are implemented for objects of the IoT (in addition to these for Wi-Fi, etc.) [4]. An important feature of the wireless modules is their diversity, both in terms of the wireless technology used, and the way they are implemented (design, chipsets, etc.).

Programming of these modules is usually done at a very low level, and this is generally "tied" to the chipset used. So the programs, in general, are not transferable to other wireless modules. In addition, programming in low level requires considerable expertise, which the companies that manufacture devices for the IoT, do not have or do not want to acquire.

Companies using such modules to build devices for the IoT show a preference for higher-level programming languages; one of their most important preference being the Python programming language [5].

Currently, the IoT market is dominated by approaches where the devices are "built" around one or more modules. The role of these modules is to add intelligence and connectivity with previous-generation devices. As mentioned however, in these approaches software is "tied" to the hardware and they require the customer familiarity with each manufacturer's software and libraries, making the development of new products difficult. On the other hand, in cloud controlled approaches the cloud service providers offer an infrastructure for storing and managing information, together with software to link the data with a range of services. To allow different devices to connect and make use of these services, the cloud service providers give source code snippets or libraries to popular languages (such as PHP, Python, Java, etc.), which are incorporated in the software of the devices during the development process. Main drawback of the cloud controlled approaches is that there are serious risks of data security and privacy [6–8].

This work proposes an extension to Wubby, an existing Python-based middleware, to also support wireless modules for mobile networks, that are used in IoT devices. The middleware is upgraded to support wireless modules of various cellular network technologies (e.g. 3G, 4G, etc.) and is ready to integrate the forthcoming 5 G modules. The extension exposes a consistent well-defined set of common functions that capture the features and the use of the wireless modules, that are accessible through some common classes for networking. Thus the extension hides from the (higher-level) programmer both the wireless module's implementation details and the underlying networking technology used. The resulting system allows easy programming of these modules, leading to programs that are reusable with different wireless networking modules.

The remainder of the paper is organised as follows: Sect. 2 presents the proposed middleware, its architecture, and its interfaces/APIs; Sect. 3 presents the features and characteristics of the middleware; Sect. 4 discusses one use case / application; finally, Sect. 5 summarises the paper and outlines future work.

2 The Middleware and the Proposed Extension

When we talk about IoT devices, we usually mean embedded electronics, a microprocessor to provide intelligence used in conjunction with an RF chip or module providing connectivity. In that context, when it comes to IoT development, for the most part we are talking about Embedded Development. The products are designed in the Device Makers or Design Houses labs and their software development stays there, remains static and unable to change by someone else, other than the manufacturer himself. The contribution of the development community is minimum or zero, and this is because the products are closed, not following any standards, and the development of embedded applications remains challenging.

The proposed solution is literally changing this by drastically broadening the audience of developers that can contribute, extending an existing offering to address the emerging 5 G market, specifically targeting Smart Cities applications.

The idea behind Wubby is that future solutions should be based on a Virtual Machine that runs on a list of supported microcontrollers and provides a runtime environment for python code execution. Wubby will be used as the root infrastructure, but the software stack will be enhanced with APIs and libraries focused on the integration of 5 G solutions, targeting applications in Smart Cities.

2.1 Wubby

Wubby (pronounced Wha-bee) [9] is a software platform that simplifies the development of IoT devices by providing a programming environment that supports Python code execution directly in the devices microcontroller. This introduces several advantages: (a) It allows a broader set of developers to contribute, giving them the opportunity to design and develop new everyday objects based on a popular programming language like Python. (b) It speeds up the development process (c) It reduces development costs (d) It results in smarter, interoperable everyday objects Wubby separates hardware from software, abstracting the hardware complexity, while at the same time allowing developers to contribute by writing simple python scripts, rather than having to deploy the whole device image.

2.2 Architecture

The architecture makes use of the existing Wubby development environment, providing extensions in the Wubby VMs that focus on the use of mobile networking (i.e. 4 G / LTE-A, and the forthcoming 5 G) solutions. The high-level architecture is shown in Fig. 1.

The *Wubby VM* runs in the 'smart' object (actually in its microcontroller), and abstracts the hardware (i.e. provides a hardware agnostic environment).

In order to support application creation the environment includes:

Wubby Cloud: provides all the services for application deployment and backend device management

Wubby Client: allows a user to control and configure each device. This can be either a smart phone app or web service.

Wubby IDE: platform independent development environment that allows easy application development (debugging, code uploading, simulation, etc.) with Wubby.

Every Wubby enabled product is registered at the Wubby Cloud and is assigned to a default Wubby Application, a device-level application written in Python that can be uploaded on the Wubby Cloud and run in any Wubby enabled device. Wubby Cloud acts as a market place for Wubby Applications in

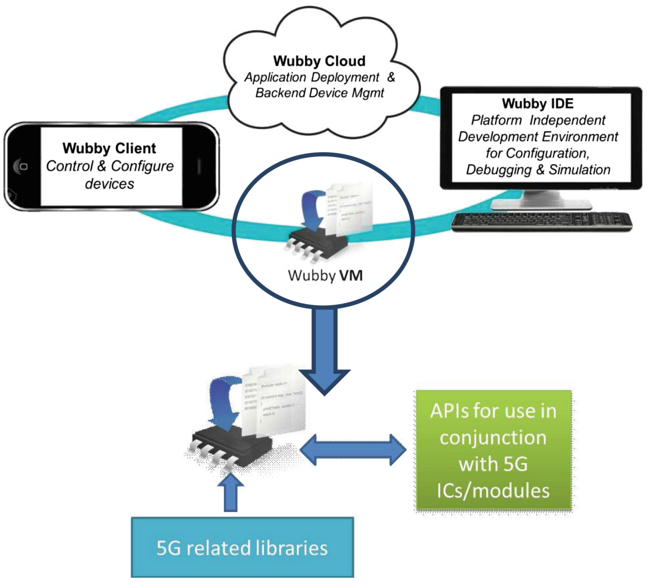
Wubby Cloud
Application Deployment & Backend Device Mgmt
Wubby IDE
Platform Independent Development Environment for Configuration, Debugging & Simulation
Wubby Client
Control & Configure devices
Wubby VM
APIs for use in conjunction with 5G ICs/modules
5G related libraries

Fig. 1. High-level architecture of the system

the same way as iOS Apple Store and Google Play and the access to it is possible from the Wubby Clients (Web or Android/iOS Apps). For this purpose, owners of Wubby devices (end users) can register them in the Wubby Cloud in which they can select one of the compatible applications to install in their devices.

2.3 The Proposed Extension

The proposed extension involved changes in the Wubby VM to support the mobile (cellular) network wireless modules. These changes regard:

- supporting the relevant modules by installing and activating the necessary drivers,
- implementing the python classes to communicate with the module / chip set, and manage it. this is done by providing an implementation of the corresponding class for the specific module,
- implementing the additions/changes to the wlan class to support inquiring link status for the mobile (cellular) network modules,
- installing the network stack for these mobile networking technologies, and
- implementing the additions/changes socket class in order to support networking over the newly supported modules.

3 Features of the Middleware

3.1 Features

The programming of the Smart Cities applications is based on scripts, which run on top of a stack providing:

- Lexical analyzer
- Parser
- Compiler
- Code emitter: emits byte code or machine code
- Virtual machine: interprets bytecode

Some characteristics (already provided by Wubby) are:

- support for Python 3.4
- almost entire Py3 standard library
- subset of the CPython object model
- runs on bare metal (no Operating System) or on RTOS (freeRTOS is already supported)
- supports multiple platforms (SoCs)
- RF independence
- small footprint (75-250KB flash, can run with 8KB RAM)
- supports optimizations to create native code for cortex and others (native bitwise operations, dynamic type handling etc.)
- Python has special commands that interface directly with assembly (specifically, the ARM Thumb-2 instruction set)
- Python in Wubby targets any environment with ANSI C99 support (works on 8-bit or even 1-bit microcontrollers, given enough code storage and RAM)
- Inline assembler
- is written in C99 ANSI C
- runtime helper functions, etc.

3.2 Benefits

Wubby already offers several benefits, as it:

- reduces the overall development time, offering a much simpler programming environment, language syntax and restrictions,
- provides a separation between software and hardware, thus making applications (scripts running on top of the middleware) re-usable among different hardware platforms,
- reduces the after-sales support needs,
- dramatically broadens the developer audience that is able to contribute in the development of such applications,
- adds intelligence at the device level, contributes in the efficiency of device-cloud communications, reducing the amount of data that needs to be transferred, as a pre-processing phase is executed at the lowest level, and

– supports networking using various WiFi and BTLE modules. With the proposed extension, support will be added for networking using mobile (cellular network wireless modules), which is of great importance for Smart Cities applications.

4 Example Application: Air Quality Monitoring

This case study aims to give citizens a comprehensive view of the air quality, using smart sensors and base stations established in different places (see, e.g. [10,11]). Wubby enabled devices, equipped with various air-quality embedded sensors, which are small in size and exploit the features offered by mobile networks, are used to monitor the collected air quality data in real time, and upload the data to remote servers for further analysis.

The first step in this scenario is the installation of a sensor network that provides real-time measurements of carbon dioxide, temperature, pressure and humidity. Each sensor can collect the data and transmit them over a low power wide area network, exploiting the high speeds of modern cellular networks (e.g. LTE, LTE-A, etc.) and further 5 G features.

The above study could be designed to cover entire cities or hundreds of square kilometers giving many capabilities for air quality monitoring for indoor and outdoor conditions, independent of the location. The application provides the end user with intelligence and better understanding of the environment that one lives in.

5 Conclusions and Future Work

This works proposes an extension to the Wubby Python-based middleware for IoT devices, to support wireless modules for mobile (cellular) networks. The extension actually concerns the Wubby Virtual Machine, which is enhanced to support 4 G / LTE-A wireless modules. The extension integrates seamlessly with the existing networking classes of Python, and allows existing applications to work with the enhanced Wubby VM. Thus it promotes code reuse and extends the scope and of existing applications to more networking domains.

Future work will focus on the support of specific 4 G / LTE-A wireless modules by the Wubby VM, as well as preliminary support for 5 G wireless modules (based of course on the availability of the expected development boards for the respective modules).

References

1. Castillo, A., Thierer, A.D.: Projecting the growth and economic impact of the Internet of Things. Economic perspectives, pp. 1–10 (2015)
2. Verizon: State of the market: Internet of Things 2016, pp. 1–24 (2016)
3. Popescu, G.H.: The economic value of the industrial Internet of Things. J. Self-Governance Manag. Econ. **3**, 86–91 (2015)

4. Wang, S., Hou, Y., Gao, F., Ji, X.: A novel IoT access architecture for vehicle monitoring system. In: 3rd IEEE World Forum on Internet of Things, pp. 639–642. IEEE, Reston (2016)
5. Python. http://www.python.org
6. Gubbi, J., Buyya, R., Marusic, S., Palaniswami, M.: Internet of Things (IoT): a vision, architectural elements, and future directions. Elsevier Future Gener. Comput. Syst. **29**, 1645–1660 (2013)
7. Tao, F., Cheng, Y., Xu, L.D., Zhang, L., Li, B.H.: CCIoT-CMfg: cloud computing and Internet of Things-based cloud manufacturing service system. IEEE Trans. Industr. Inf. **10**, 1435–1442 (2014)
8. Rao, B.B.P., Saluia, P., Sharma, N., Mittal, A., Sharma, S.V.: Cloud computing for Internet of Things & sensing based applications. In: 6th International Conference on Sensing Technology, pp. 374–380. IEEE, Kolkata (2012)
9. Wubby documentation. http://www.wubby.io/docs
10. Cho, H., Kyung, C.-M., Baek, Y.: Energy-efficient and fast collection method for smart sensor monitoring systems. In: International Conference on Advances in Computing, Communications and Informatics, pp. 1440–1445. IEEE, Mysore (2013)
11. Postolache, O.A., Dias Pereira, J.M., Silva Girao, P.M.B.: Smart sensors network for air quality monitoring applications. IEEE Trans. Instrum. Meas. **58**, 3253–3262 (2009)