PiFrame: A Framework for Home Automation Platform on the Full Feature OS

Pasd Putthapipat
Department of Computer Engineering,
Vincent Mary School of Engineering, AU
Bangkok, Thailand 10540
PasdPtt@au.edu

Kittiphan Techakittiroj
Department of Mechatronics Engineering
Vincent Mary School of Engineering, AU
Bangkok, Thailand 10540
Ktech@talent.in.th

Abstract

This work provides the framework for implementing the embedded systems on the embedded computer that running full-feature OS to shorten the development cycle and reduces the embedded system limitation. The Raspberry Pi, running the full-feature Linux, is used as the based system of the framework with the additional development of software environment and general purpose hardware extension. The framework can be used for both education and research.

Keywords: Raspberry Pi, Embedded System, Framework, Home Automation.

I. INTRODUCTION

The objective of this work is to develop the general purpose framework for embedded computing education and IoT in house research base on Raspberry Pi [1] at Vincent Mary School of Engineering, Assumption University. The Raspberry Pi itself has been widely used a lot in application research such as home automation [2-3], robotic [4], wireless sensor network [5]. With the affordable price and fully equipped interfaces, it was also been used as a based hardware for multiple framework development [6-9]. Our work relies on using the standard modern tools especially the full feature OS which support wide range of applications.

The purposes of this framework are three main issues; reducing the development-cycle, reducing the cost of the development environment, and reuse capability.

In the process of computer engineering education, we knew that the hardware development process is time consuming and cost ineffective. In many case, the majority of course was invested to the PCB design and fabrication, which might not relevance to the key content of the course. Also the tailor-made design hardware is difficult to redesign and rebuild, also the software which was coded for those hardware is also difficult to reuse with the different design. To overcome those hardware development issues, the general purposed framework is desired. This framework must be simple however still flexible enough due to the fact that it must support wide range of users, undergraduate students to the field research engineers. The hardware and software for this framework also have to be easy to develop and reuse over and over to speed-up development process and encourage team project.

The remaining part of this paper consists of the following sections: section II Smart Devices; section III Framework Architecture; section IV Advantages and Disadvantage of the Framework; section V conclusion.

II. SMART DEVICES FRAMEWORKS

The world is moving toward the IoT (Internet of Things). All “smart” devices are supposed to be connected to the Internet. There is an increasing demand of the control and acquisition feature in the embedded systems for these devices. The network feature is an unavoidable feature of the embedded systems for “smart” devices. Also the internet access will give the limitless function to the device itself. However the conservative embedded system is based on the microcontroller which is difficult to deploy the network capability. However with the current technology, the embedded computer, such as Raspberry Pi or Beagleboard [10], can be installed on the full-feature Linux. The full-feature Linux allows the high flexibility in expanding the systems to advanced OS features and Internet.

III. FRAMEWORK ARCHITECTURE

A. Framework Architecture Diagram

Fig. 1 demonstrates the framework structure which consists of five layers. The software layers, applications and in house processes would communicate with the hardware through the OS capability and standard API. An operating system itself would take care the resource management, hardware request and communications. By doing this, the troublesome issues, i.e. TCP/IP stack handling or process scheduling, in embedded system development were minimized.