Proof of Concept of a PIC Wireless Programmer Interface For Prototyping

Slim Chtourou[2],Mohamed Kharrat[1], Nader Ben Amor[2], Mohamed Jallouli[2], Mohamed Abid[2]

1.Aliyet, Tokyo, Japan (mohamed@aliyet.com)
2. Université de Sfax, Ecole Nationale d'Ingénieurs de Sfax
Computer & Embedded Systems CES laboratory, Sfax, Tunisia
(slim.chtourou@ieee.org, {nader.benamor, mohamed.jallouli, mohamed.abid}@enis.rnu.tn)

***Abstract*—In the near future, computers will be embedded everywhere and the Ubiquitous computing will be widely used. Ubiquitous computing requires multidisciplinary knowledge. To design such systems, a specific heterogeneous prototyping platform is required. Such platform has to be sufficiently versatile and in the same time easy to program even for beginners. Most ubiquitous systems are microcontrollers-based and are already too hard to program, limiting therefore their flexibility.**
In this paper we present a new concept of an easy to use wireless prototyping platform for Ubiquitous Computing suitable for people with limited computer skills.
The prototype realized in this paper consists of a PIC that interfaces a Bluetooth USB dongle to communicate with the computer and send the program file to the targeted PIC to program it in place and from a distance.

***Keywords*—Ubiquitous Computing, Prototype, PIC, USB, Bluetooth**

I. Introduction

In his speech at Computex in the 2nd of June 2011, Mr. Ian Drew, the Vice President of ARM Marketing department stated that it is expected that the number of connected devices will pass from 12.5 billion today to 1 trillion devices by 2025 [1]. The main reason behind such an expected exponential increase is that most objects surrounding us will embed a computer in the future. These objects will be able to communicate together intelligently and execute tasks without human intervention. Taking consideration of many aspects such energy issue, it is expected that most of these object will run on low speed microcontrollers. According to Ian Drew, it is expected that microcontroller market will have the highest market expansion growth by 2020, exceeding those of high performance processor used today mainly in communication devices, mobile phone and personal computers.

Ubiquitous computing, known also as pervasive computing and ambient intelligence consists of intelligent computer that can auto-adapt to the human environment to provide a new service or adapt it to the customer. These computers are expected to be integrated into everyday activities and objects such as furniture's tables, chairs, beds and even clothes, introducing therefore, the new era of internet of things. [2]

As an example of possible future scenario of Ubiquitous computing technology, a refrigerator will be able to detect automatically foods and this by reading RFID labels of each product and adjust internal temperature based on its content. A shortage of some food product can be detected automatically too. In such a case, a notification can be sent to the user asking him about agreeing the transaction of ordering these products.

Nowadays, the computers are embedded in many places. However, many objects are still without computers.

Ubiquitous computing requires multidisciplinary knowledge across ranging from systems design and engineering, systems modeling, user interface design, computer networking, wireless sensor communication, security and energy supply.

The first step to a new design is prototyping, this procedure is very important to validate an industrial concept and convince investor when seeking funds for mass production.

Prototyping is not an easy task, since embedded systems are getting more and more complicated and their design presents many difficulties such finding the reference design, installing software, debugging and modifying code. There are some open source developing platforms like Lego Mindstorms but they remain expensive. They support few components and are not suitable for prototyping.

The ultimate objective of Ubiquitous computing is to make technology fade in the background and made the humans' life comfortable without worrying about technical details. In this context, our objective is to create an easy to program prototyping platform based on microcontrollers that is suitable even for people with limited computer knowledge. It allows them to learn basic concepts of electronics and programming without entering complex details.

II. Overview of Existing PIC Programmers

A PIC programmer is a device that allows programming other PICs, there are several programmer types:

A. RS232 PIC programmer

This PIC programmer can be connected to a computer via a classic RS232 serial interface, this type of cables is getting outdated and not included anymore in recent computer models.

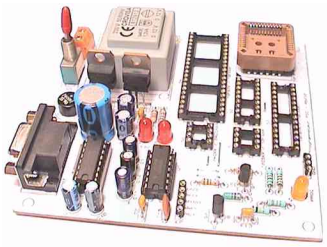

Fig. 1.RS232 PIC programmer

978-1-4799-1758-7/15/$31.00 ©2015 IEEE

B. USB PIC programmer

This device is similar to the RS232 PIC programmer, the only difference is that it connects to a computer through a USB interface

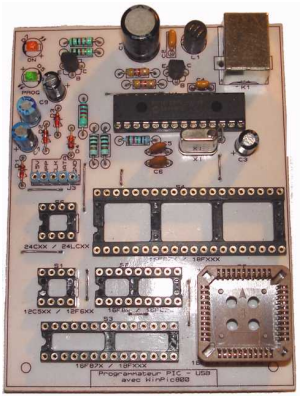

Fig. 2.USB PIC programmer

C. Bluetooth PIC programmer

Actually, it is not exactly a Bluetooth programmer, but rather a combination of a RS232 PIC programmer and an RS232-BT adapter as show in figure 3

Fig. 3.Bluetooth serial adapter

III. CONTEXT OF WORK

Even though there a lot of PIC programmer available, they present some limitations: the RS232 interface is available mainly on older computers, so in order to program PIC, a user must use an old computer or buy a RS232 to USB adapter, the USB PIC programmer is good but it has to be connected to the computer in order to transfer a program to the PIC, the Bluetooth serial adapter may seem a good solution but actually, the Programming PIC needs first to interface that device which is a complex tasks, not mentioning the additional costs of purchasing the device.

In this paper we will detail the process of creating a low cost microcontroller programmer that makes benefit of the wireless technology to program the device from a distance without the need to use cables, costly programmer, programming software and compiler. Therefore, the user will no longer need to attach the microcontroller to the computer and install the compiler, adding therefore a high degree of user flexibility.

In this paper we show a wireless programmer based on PIC microcontroller that can communicate with a host computer via a Bluetooth module. The programming PIC will receive the program file from the Bluetooth USB module and then transfer it to the programmable device using a serial transmission.

IV. WIRELESS PROGRAMMING PLATFORM OVERVIEW

A. System general Structure

The system includes two main components: the wireless programmer and the ubiquitous system. In this paper, we consider the elementary example of a microcontroller-based ubiquitous system.

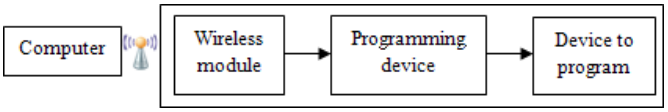

Fig. 4.Overview of the wireless device programmer

The program file (binary .hex file) will be transferred from the computer via wireless transmission to the wireless module. The programming device will communicate with the wireless module through an interface to acquire the .hex file. After receiving the program file, it will send it serially to the ubiquitous system.

B. Wireless programmer design

It is composed of two modules: a wireless module and a programmer microcontroller (MCU)

1) the programmer MCU

For the programming device, the choice has been set on Microchip PIC MCU. The reasons of this choice are the following:

- Among various microcontroller types such as AVR and STM, PIC microcontrollers are more optimized for USB communication.
- Even though the Arduino platform is getting widely known, the PIC microcontrollers are more popular and widely used in the industry.
- PIC devices are widely available with affordable prices.

For the PIC, we choose the reference 24FJ64GB002 for these reasons:

- Previous works has been done before using this reference.
- In USB communication there are two sides: a host and a device. The host has a full support of USB protocol, transaction management..., while the device has a limited support of USB protocol, just the minimum requirements for sending receiving data. The heavier workload is assured by the host.
  In the PIC there are various families: 12F, 16F, 148F 24F, 32F…, although the 18F has a support of USB but it can act only as a device, we need it to act as a host to be able to control the USB wireless module. That's why we can't use that reference for this case. [3,4]
  In the PIC24F families there are certain devices that are OTG device: these devices have a full support of USB functionalities and can act at times as USB targeted host and at times at USB peripheral. [5]
- Program memory: 64 Kilobytes (Kb): with a larger memory space, it is possible to create more developed

applications unlike some PIC families that have a rather little memory program space (16 Kb)

- Operating voltage range: 2 ~3.6 V: some PICs operate at 4 or 5.5V, when it comes to embedded battery operated systems, the PIC24FJ64JB002 would be an excellent choice.

This PIC reference is new comparing to other families like 12F or 16F, conventional programmers can't program this reference. So to program PIC 24FJ64GB002, we will use the Pickit3, a microchip's programmer & debugger kit. [6]

2) Wireless module

For the wireless module, we studied three wireless technologies: Infrared (IR), Bluetooth (BT) and Radio Frequency Identification (RFID). The choice is fixed to Bluetooth technology. The reasons for this choice are the following:

- Bluetooth is a very popular technology.
- Bluetooth modules can be found everywhere and at a good price.
- This technology is reliable: data transmission can be done even through obstacles and power consumption is low..
- The Bluetooth standard is official and detailed documentation are provided unlike some other wireless technologies such as RFID
  Since a PIC can handle Bluetooth directly, we will use a BT USB dongle.

V. WIRELESS PROGRAMMING SYSTEM'S CONCEPTION

A. Design of the system

Now that we have made a choice for the programmable device and wireless technology, we can set the structure of the targeted system as shown in figure 5

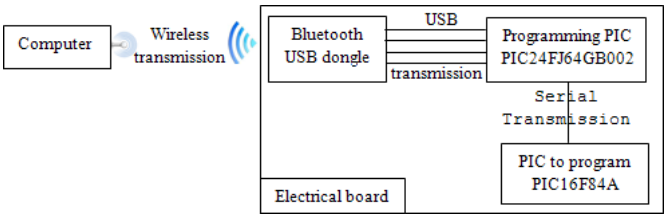

Fig. 5.Structure of the targeted System

B. System Principle

The programmer PIC will communicate with the Bluetooth USB dongle using the USB interface so it can receive the code.

The data received from the programming PIC is packed just like network packets .The received frame will be unpacked by the PIC and it will extract the useful information known also as payload. In our case, it is the binary hex file that has to be run by the Ubiquitous system. To be able to do that, PIC has to use Bluetooth commands related to the BT module.

The Bluetooth protocol has many layers. In order to retrieve the payload, unpacking function must be used for each layer.

After receiving the .hex file from the computer, the PIC24F will send the file to the PIC that we want to program using serial transmission.

C. Bluetooth Interface

1) Bluetooth layers

According to the Bluetooth specification V4.0 [7], the Bluetooth protocol stack provides several layers of functionality. These layers range from the low-level radio link to the profiles as illustrated in figure 6

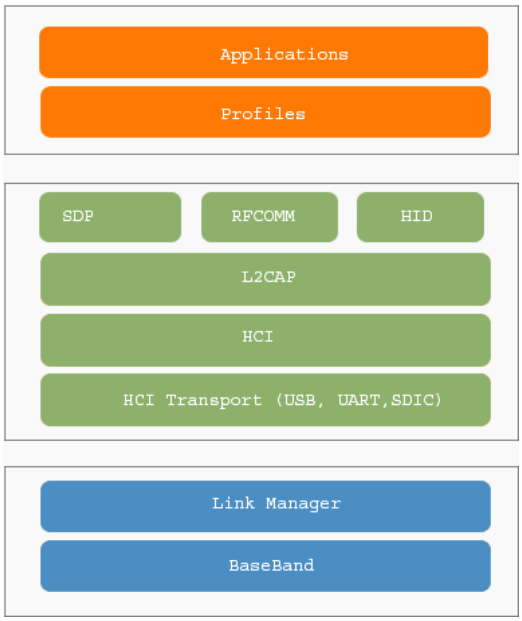

Fig. 6.The Bluetooth protocol stack

We want the programming PIC to send commands to the Bluetooth module and acquire the data sent from the computer.

"Radio", "Baseband/Link controller" and "link manager" are the lowest layers in the Bluetooth Protocol. Interfacing these layers is like trying to create an application using binary language only.

It is possible to communicate with these low layers with higher layers, the HCI layer designed to abstract and simplify physical communication between the Bluetooth stack and the controller.

We need to interface and define the following layers:

- HCI: (Host Controller Interface) provides a command interface to the baseband controller and link manager, allowing access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities. There are three HCI transport layer standards, they transfer the same command but each one uses a different hardware interface. These layers implement few functions designed to send Bluetooth commands and data packets, and receive data packets and events.

- L2CAP(Logical Link Control and Adaptation Protocol) provides data flow control and management. This layer is used to multiplex multiple logical connections using

different higher level protocols and provide packets segmentation and reassembly.

- RFCOMM:(Radio Frequency COMMunications) is a protocol that can create a virtual serial data stream. It is used to transport the user data, modem, control signals and configuration commands.

2) Bluetooth data transmission

A network that follows the OSI model is composed of seven layers. As data goes through these layers, a header will be added to the data so that it could be identified specifically by that layer on the receiver side.

The same principle applies to Bluetooth layers, when the data is transmitted from the computer via its Bluetooth module. The information will be packaged three times: the first one is while passing the RFCOMM layer, the second in the L2CAP layer and the last in the HCI layer.

When this package arrives to the Bluetooth module attached to the programmer PIC, it will undergo the reverse process: the data will be unpacked going up from HCI layer to the RFCOMM layer, the data is then transmitted to the PIC via USB interface.

The process of packing, unpacking is shown in Figure 7:

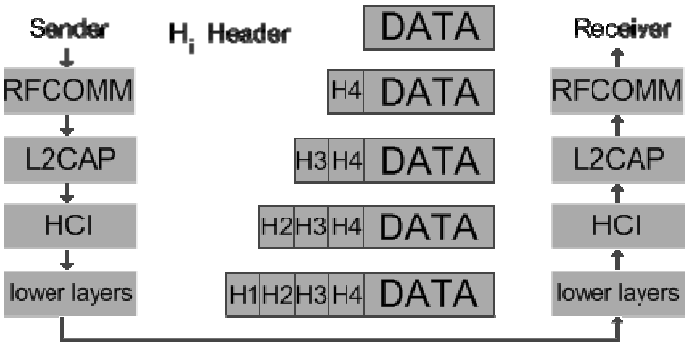

Fig. 7.Data packaging through Bluetooth layers

D. USB Interface

1) USB communication flow

In order to be able to send commands to the Bluetooth module and retrieve the incoming data, the PIC has to interface the Bluetooth module via the USB protocol. [8]

The USB data flow between the host and the device is illustrated in the figure 8:

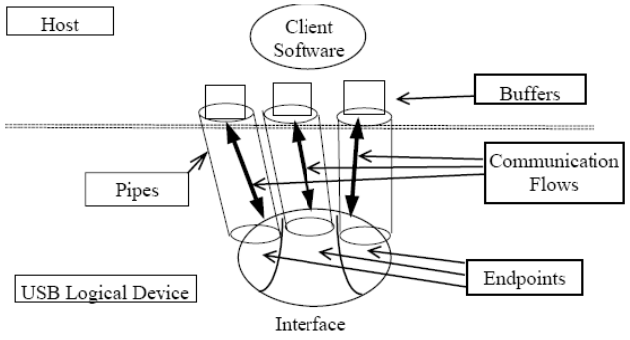

Fig. 8.USB communication flow

The client software is responsible for sending USB requests, it is embedded within the host which is the PIC 24FJ64GB002 in our case.

Buffers are storage space used to compensate for differences in data rate, time of occurrence of events when exchanging data between two USB devices.

An endpoint is the terminus of a communication flow between the host and the device. Each endpoint on a device has a unique identifier, a device-determined direction of data flow. A device may contain several endpoints and each one is uniquely referenced.

Pipes represent the ability to exchange data between the software on the host and an endpoint on a device via a memory buffer.

The interface guarantees the interoperability between the USB logical devices and endpoints.

2) USB descriptors

Each USB device has descriptors that contain different information:

- Device descriptor: this descriptor provides general information about the USB device. These information are global and apply to the device and all of its configurations.
- Device qualifier descriptor: it describes information about high speed capable devices that would change if the device were operating at other speed.
- Configuration descriptor: it contains information of the device's configuration, a configuration provides one or more interfaces
- String Descriptor: this is an optional descriptor, it contains information about the device that will be displayed in a graphical interface. For example when a device is plugged to the computer, a tooltip appears saying ''new hardware found'' and below, there is the device name which comes from the string descriptor.
- Interface descriptor: this descriptor contains information of a specific interface within a configuration. This descriptor is returned as a part of a configuration descriptor.
- Endpoint descriptor: this descriptor contains information needed by the host to determine the bandwidth requirements for each endpoint. Each endpoint used for an interface has its own descriptor.
  Like the interface descriptor, the endpoint descriptor is returned as a part of the configuration information.

3) Retrieve Bluetooth USB dongle characteristics

The programming PIC24 needs to know the USB details of the Bluetooth device for it to be able to communicate with it.

To identify those details we will use the software USB Trace.

This software allows a user to capture all USB frames exchanged between the computer and USB devices and furnish also a detailed description about the device's USB resources.

With this software, this Bluetooth USB device's characteristics were retrieved successfully.

E. Ubiquitous system

The ubiquitous system that will be programmed is based on PIC 16F84A MCU.
To program a PIC, specific connections have to be made:

- The reset PIC input named MCLR (V_{pp}) will be powered with 12V, indicating to the PIC the beginning of the programming procedure.
- The data will be transmitted to the PIC via its PGD (data) pin.
- The transmission is synchronous, to guarantee this, the clock is sent to the PIC via its PGC pin.

In order to transmit data to a PIC, a serial port can be used to transmit the .hex file, byte by byte using a RS232 connection.

VI. SIMULATION AND RESULTS

A. Program the PIC

In this simulation, we will program a PIC 16F84. The programming device for this simulation is the PIC 18F2550, a simulation couldn't be made with PIC 24FJ64GB002 since this reference is not available in ISIS libraries. However, in this simulation, we will focus on the data transmission aver USB protocol, the principle is the same if excluding the fact that the PIC will communicate with Bluetooth device.

To transfer the .hex file to the PIC, we will use the WinPic software. After opening the file and clicking on the "program" button we can observe USB transaction via ISIS VUSB Analyzer.

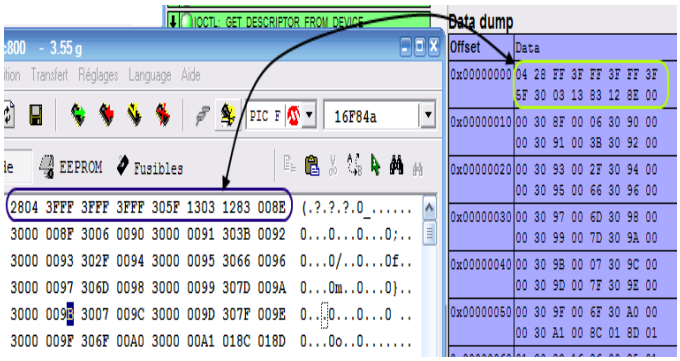

Fig. 9.Transmission of the .hex (right) file using WinPic (left)

The file transmission was successful over USB bus, the bytes display is inverted in WinPic that's why we have 28 04 in WinPic, while in fact, it is 04 28 (the first value displayed in this simulation in the figure 9).

B. Detect payload within Bluetooth packages

In this simulation, we will identify the data transmitted in packets from a Bluetooth device to the USB module.

To detect the incoming USB packets we will use the USB Trace software.

As a data sample, we will consider the data transmitted after pressing a keyboard key (scan code), we will use two different keyboards. We turn the USB buffer capture "on" and for each keyboard, we will press the B key then the C key,

The table1 illustrates scan code values of some keyboard characters:

TABLE I. A PORTION OF KEYBOARD SCAN CODE VALUES [9]

Key Name	HID Usage ID
No Event	00
Overrun Error	01
a A	04
b B	**05**
c C	**06**
d D	07
e E	08

The result is illustrated in figure 10:

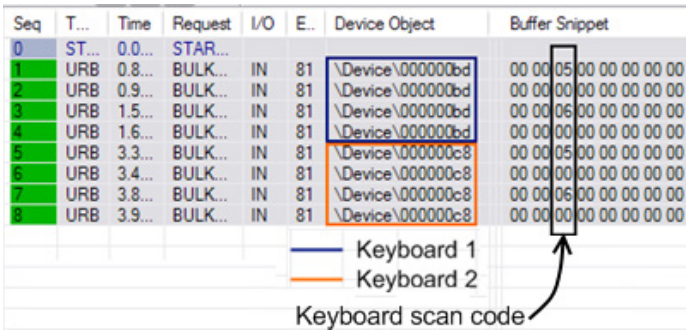

Fig. 10.captured USB packets from 2 different USB keyboards

"Périphérique USB Composite" in the USB Trace software's "Device Object" refers to the keyboard, in the "buffer Snippet" column, there is an 8 bytes data.

The difference between all the packets captured is the 3rd byte which indicates whether a key is pressed or not and which one it is.

As shown in figure 10, the packets from 1 to 4 are identical to the packets from 5 to 8, each set of packets comes from a different keyboard. We can identify them by their "Device Object" value. (right before the "Buffer Snippet" column).

The data captured here is a keyboard's key unique identifier called scan code. A list of different key values can be found at Microsoft MSDN site [10].

A scan code is always the same, whether we use AZERTY or QWERTY keyboard, a USB or a Bluetooth keyboard. So we will go and analyze data coming from a Bluetooth keyboard and detect the scan code data within the frame.

In figure11, we capture the data transmitted from both a USB and a Bluetooth Keyboard.

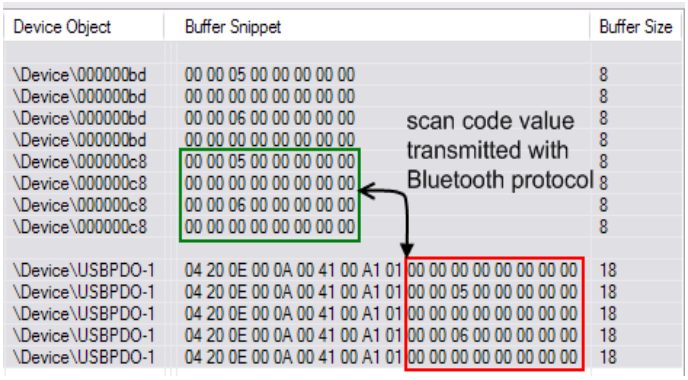

Fig. 11.captured USB packets from a Bluetooth keyboard

At the first look it may look different, the frame size for an USB keyboard is 8 bytes while for a Bluetooth keyboard, it is 18 bytes. However, after taking a closer look we observe that the last 8 bytes of this frame are identical to the frames captured earlier with USB keyboard.

This means that the code scan value has been transmitted successfully, the 10 first bytes are specific to Bluetooth protocol:
1st& 2nd byte: HCI CID.
3rd and 4th byte: length of the HCI packet data, the bytes read order is from left to right. It is written 0E 00 but the DATA size in this example is 00 0E (14 in decimal) and not 0E 00 (3584_{10}).
The HCI data bytes are in fact a frame of another Bluetooth layer which is L2CAP:
5th and 6th byte: L2CAP CID
7th and 8th byte: L2CAP data length, like in HCI frame, the read order is from left to right, so the data length in this example is 00 0A=10_{10}: that is the number of remaining bytes.
The remaining bytes represent a frame of a higher Bluetooth layer which is RFCOMM:
9th and 10th byte: RFCOMM CID

Now the remaining data represents our USB frame, the same value that we obtained using an USB keyboard.

The process of packaging can be resumed in figure 12, the USB frame contains the "B" scan code

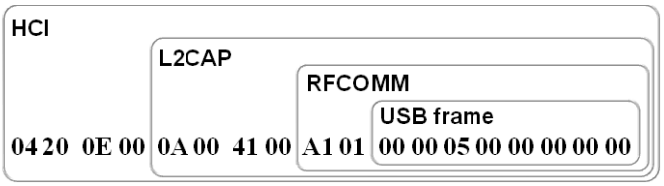

Fig. 12.USB frame inside the Bluetooth frame

VII. CONCLUSION

The ultimate objective of Ubiquitous Computing is to establish the best possible environment for humans and simplify complex tasks.

This paper aims to concept and design a prototyping card for PIC wireless interface, in order to facilitate the systems' prototyping and make the creation of an embedded system as easy as creating a robot using Lego.

There are several microcontrollers-based Ubiquitous systems available, however, their programmers are still not easy to use and quite expensive. The concept of the proposed prototype itself can be seen as a standalone system that can be embedded into objects such as doors, chair etc. It makes benefit of wireless technology to program various PIC references (such as 16F and 18F etc) in their place without the need to remove the microcontroller from its place and thus damaging its pins.

We showed in this paper how to create a low-cost wireless PIC programmer that can be as low as 10$ when entering the mass production stage. It solves the problem of cable congestion and makes the programming task easier. This programmer is the first step in creating a prototyping platform that enables people with limited electronics and computer skills to master these skills and build their own prototype.

REFERENCES

[1] Computex Taipei 2011, https://www.computextaipei.com.tw, May 2011 [21 Nov 2014]
[2] Gregory Adam Greenfield., *Everyware, The Dawning Age of Ubiquitous Computing*, 2006.
[3] PIC16F84A datasheet
[4] PIC18F2550 datasheet
[5] PIC 24FJ64GB002 datasheet
[6] PICkit_3_User_Guide
[7] *Specification of the Bluetooth System*, 2010.
[8] Universal Serial Bus Specification Revision 2.0, 2000.
[9] Microsoft USB HID scan code map for keyboards
[10] *The Unicode Standard Version 5* (2007)