

# EZURiO

## Application Note

AN0013

*The WISM – A complete, embedded wireless processor*

## Overview

Adding wireless LAN capability to a product can be a surprisingly difficult and complex task. EZURiO has developed its Wireless Intelligent Serial Module (WISM) range of wireless processor modules to make it simple. This application note explains the differences between using a WISM and the traditional approach of a processor based design and illustrates the saving in cost and time-to-market.

## 1. Introduction

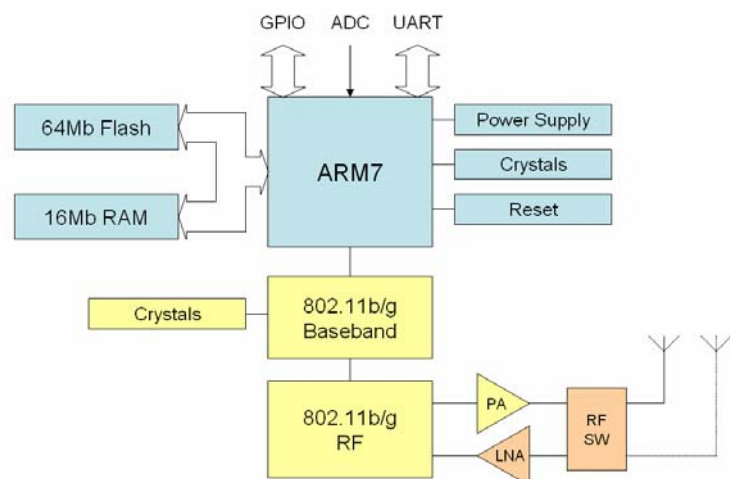
The simplicity of adding Wireless LAN capabilities to a laptop – essentially just purchasing a PCMCIA card or USB adaptor results in many people thinking that adding wireless LAN to an embedded product should be just as simple. What they forget is that the PC and the Windows operating system is performing a large part of the task. When you look at adding wireless LAN to an embedded design, the overhead that the PC performs needs to be duplicated in embedded hardware. However, most embedded designs do not have the memory or processing power to address this. The EZURiO WISM wireless LAN modules were designed specifically to address this marketplace, where the embedded processing capability is at a premium or non-existent.

## 2. Fundamentals of Wireless LAN design

In order to use 802.11 to connect your product to the internet you need a number of components.

The first is the Wireless LAN functionality. This is what you get in a USB adaptor or a simple 802.11 module. It consists of the radio transceiver, along with a baseband processor. It may include a Power Amplifier to boost the output power. The component parts are shown in yellow on the WISM block diagram.

In order to reduce costs for PC peripherals, such as USB adaptors, silicon vendors have cost reduced their chipsets by implementing the baseband as a processor core, rather than as a hardwired Media Access Controller (MAC). The one downside to this is that every time the wireless LAN silicon is powered up, the baseband firmware needs to be



Block diagram of the WISM wireless processor module

downloaded from a host processor. The average size of this code is around 100 kBytes, which starts to build up the memory footprint for the embedded processor system.

Once the baseband firmware has been deployed, the processor needs a driver that can control the baseband and provide the bottom layer to interface to a TCP/IP or UDP stack. These drivers are specific to each silicon chip, which means that if an embedded designer needs to change their 802.11 chips supplier, they will need to rewrite the driver. It's normally necessary to rewrite the driver to move between successive chips from a single vendor. As new silicon is released every eighteen months and the lifetime of any chips is now only about 3 years, this can be an expensive recurring task for a company.

Silicon vendors do offer drivers for common operating systems, such as Windows or Linux, but rarely support embedded operating systems. Even when a skeleton driver is supplied, experience shows that it takes around 2 man-months to integrate it into the RTOS and another 6 man-months to optimise it with the higher level stack to give robust, low power performance. The driver will consume another 150 kBytes of memory.

The driver just provides the foundations for the protocol stack that sits above it. In most cases this will be a TCP/IP stack. There are plenty of available stacks to license for Real Time Operating Systems, which vary in their complexity, size and robustness. Our experience with M2M applications is that it is important to use a fully featured, well tested stack. The reason is that you never know which Access Point your product will connect to. Choosing one of the industry standard stacks can remove months of field support.

If your wirelessly enabled device is going to be accessed from the internet, then you will also need to add a web server, which needs to be integrated on top of the stack and also to the I/O of the module, so that it can report external conditions. A solid TCP/IP implementation with an HTML server will take up about another 300 kBytes of your flash memory.

That provides your 802.11 implementation. But to use it you'll need to develop software that can search for access points, associate with them and control the flow of data from I/O or external sensors or processors. To do that on top of a "raw" 802.11 implementation will take considerable time and a thorough knowledge of the wireless standards. To simplify the process we have developed our UWScript language.

UWScript is a scripting language that we have evolved from the BASIC language. To make it suitable for embedded wireless communications we have enhanced the BASIC functionality with a range of wireless specific tokens that control the wireless LAN functionality. This isolates the user from the complexity of the wireless connection by providing high level abstract commands. Complete functions, such as finding and associating with access points, or sending email are provided as script libraries, which can be cut and pasted to produce complete working programs in a matter of hours.

Because the WISM modules also contain an integrated interpreter, user addressable Flash memory with a filing system and ample RAM, these programs can be stored and run natively within the WISM module. It allows complete embedded applications to be developed with no additional external hardware. Moreover development times decrease from months for a traditional embedded system to days.

## **The WISM vs. an external Processor implementation**

The WISM is not just a Wireless LAN module. Although it is no bigger than the average Wireless LAN module it is a complete wireless enabled processor system that is capable of stand-alone operation. It has been designed to contain all of the features that are required for 802.11 connectivity:

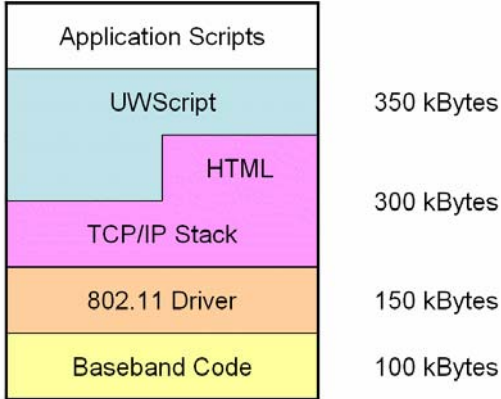
- ARM7 Processor
- 16M RAM
- 64M Flash Memory
- Integrated Voltage Regulators
- 9 GPIO & 2 ADC
- Integrated UWScript language
- Scripting Interpreter
- 802.11b/g Wireless LAN
- Power Amplifier
- Low Noise Amplifier
- Antenna Diversity
- High Performance Ceramic Antenna
- Full FCC and CE certification

To compare the design process and costs of using a WISM to the traditional approach, it's useful to look at a simple implementation, such as an internet enabled thermometer. To design this with a WISM, all you would need to do is to add a small motherboard containing the thermometer and any associated components, plus a battery pack and a 40 pin connector for the WISM.

For an embedded design, you would need same thermometer and components at one end and a wireless LAN module at the other. But in between you'd need to design a complete processor system to handle following tasks:

- Download the 100k baseband code for the WLAN module
- Run the chip specific 802.11 drivers
- Run a TCP/IP stack
- Run software to attach to an access point and manage the wireless connection
- Run your temperature measurement application

You'll also need to choose and implement a real-time operating system to bind it together. You'll either need to design this processor board from scratch, or else purchase a suitably specified commercial processor module.



In choosing a commercially available processor module it is important to ensure that there is sufficient memory available. In order to support the baseband code, driver and TCP/IP stack around 500kBytes will be required. That is before you consider your application needs. This takes processor modules from the cheap and cheerful into more exotic specifications and pricing.

If a Windows CE or Linux approach is taken, there will be an even larger memory and processor requirement. Although the market for these processor modules is sufficiently large to help contain prices, they are generally significantly larger and more power hungry.

**Hardware Cost - Wi-Fi Module plus Processor Board**

Wireless LAN Module	\$30.00	
Processor Module (ARM7)	\$89.00	Increased by Memory specification
Antenna	\$2.50	
4 layer pcb	\$3.00	
Power Supply	\$2.00	
	<b>\$126.50</b>	
Manufacturing & Test	\$5.00	
TOTAL	<b>\$131.50</b>	
Software Licences	\$4.00	Average – Win CE, etc
<b>Typical Cost</b>	<b>\$135.50</b>	<i>(based on 1,000 off pricing)</i>

Obviously another route is to design your own processor board. That will probably decrease the materials cost, but will add further development costs.

Having made the hardware comparisons, it is also important to look at the development time involved in different approaches. Whilst silicon manufacturers, RTOS and stack providers like to suggest that driver integration is straightforward, experience across many embedded developers suggests otherwise. Whilst it may not take too much time to integrate the various components in a real time environment, the experience is that the initial integration is often insignificant compared with the time to optimise the firmware and fine tune it for robustness.

## Typical Development Times & Costs

	CE	Linux	RTOS	
WLAN Driver	1	1	6	months
TCP/IP Stack	0	0	3	months
Application	0.5	0.5	0.5	month
Production Test & Approvals	1.5	1.5	1.5	months
	<b>3</b>	<b>3</b>	<b>11</b>	<b>months</b>
Internal Development Cost	\$48,000	\$48,000	\$176,000	
Stack License	\$0	\$0	\$50,000	
Wireless Product Approvals	\$30,000	\$30,000	\$30,000	
<b>Total Design Costs</b>	<b>\$78,000</b>	<b>\$78,000</b>	<b>\$256,000</b>	
Amortised per unit (1,000)	\$78.00	\$78.00	\$256.00	

In comparison, all of this is done for you with the WISM. You need to write your temperature monitoring application and wireless access, but by using the sample UWScripts supplied with the WISM, this can be developed within a few days. And if you have a long-lived product, you'll repeat these costs every time you change the 802.11 chipset.

The comparison cost above don't include the cost of development tools. For the WISM all you will need is the EZURiO Wireless Development Kit and your existing PC. An outlay of around \$150.

Finally, the product will need to be qualified and manufactured. Few wireless modules are pre-qualified, so your product will need to be taken through CE and FCC approval. You will also need to develop production test equipment to ensure that the RF performance of the complete product stays within the regulatory requirements.

Putting this together, the advantage of using the WISM for an embedded wireless LAN design becomes evident:

### Summary Costs (1,000 off)

	Processor Module			WISM	
	CE	Linux	RTOS		
<b>Total Hardware Cost</b>	\$135.00	\$135.00	\$135.00	\$102.00	
<b>Amortised Design Cost</b> (1,000 off)	\$78.00	\$78.00	\$256.00	\$4.00	
	<b>\$213.00</b>	<b>\$213.00</b>	<b>\$391.00</b>	<b>\$106.00</b>	
<b>Development Time</b> (hardware prices are indicative and for comparison purposes only)	<b>12</b>	<b>12</b>	<b>44</b>	<b>1</b>	<b>Weeks</b>

If you are planning to add Wireless LAN functionality to your products, there's no question – you can do it faster and cheaper using EZURiO's WISM. It's not just a wireless module, but a complete, embedded wireless processor, offering fastest time to market.

EZURiO Ltd  
 Saturn House, Mercury Park  
 Wycombe Lane, Wooburn Green, HP10 0HH  
 United Kingdom  
 Tel: +44 1628 858 940  
 Fax: +44 1628 528 382

[www.ezurio.com](http://www.ezurio.com)

Copyright ©2007 EZURiO Ltd. All rights reserved.

All other trademarks are the property of their respective owners.

The information contained within this Application Note is provided as a guide and is subject to change.