Power Analysis of Local Transmission Technologies

Darshana Thomas
University of Strathclyde
Department of Electronic
& Electrical Engineering
Glasgow, United Kingdom
Email: darshana.thomas@strath.ac.uk

Ross McPherson
University of Strathclyde
Department of Electronic
& Electrical Engineering
Glasgow, United Kingdom
Email: ekb12174@uni.strath.ac.uk

James Irvine
University of Strathclyde
Department of Electronic
& Electrical Engineering
Glasgow, United Kingdom
Email: j.m.irvine@strath.ac.uk

***Abstract*—With the number of Internet of Things (IoT) devices expected to explode to over 20 Billion devices by 2020, it is vital that efficient communication technologies are used. While ideally a single technology would emerge to simplify deployment, in practice the varying power and bandwidth requirements of different devices has led to an industry split over communication technologies, and while a number of new technologies have been designed with IoT in mind, commercial imperatives have meant that existing wireless protocols, in particular Wi-Fi and 433 MHz AM, remain the most prevelent. This article outlines the power usage of these two most common protocols, and considers power aspects of using each protocol in an IoT setting with experiments carried out with real world devices used in current products.**

I. Introduction

The ever decreasing cost of silicon has made it possible to have tiny computers monitor more and more aspects of our lives, this can be seen with such devices as Fitbit. A tiny computer positioned in a wrist band that has the sole purpose of helping us monitor our fitness. This could not have even been imagined 20 years ago, due to the cost and size of computers [1]. The opportunities offered by increasing access to data means that we are likely to have ever more sensors detecting our movement, vitals and habits. However, while the computing power of these sensors becomes easier and cheaper to provide, the issue remains of getting this sensor data back to a platform [2] that the user can easily access. This is the most energy demanding part of the system, an important consideration for portable devices which must rely on battery power.

This constraint has meant that a number of low power wireless technologies have been developed for wireless sensors and IoT. Bluetooth Low Energy (BLE) – Bluetooth Smart – is an enhanced version of Bluetooth which is designed for low power operation. Zigbee was designed from the start to be energy efficient. A low power extension of Wi-Fi, IEEE 802.11ah – Wi-Fi HaLow – is being standardised. However, this latter technology is not yet deployed. Zigbee is deployed, but has little installed base. BLE has been adopted by many smart phones, but has little installed base in the home, which is where many of these devices will be installed. This leaves companies which wish to deploy products now with a dilemma – while many promising technologies are on the horizon, the current choice of wireless interface is much more limited.

978-1-5090-0493-5/16/$31.00 2016 IEEE

The first wireless radio technology widely used for remote devices in the home is 433MHz. Named for the frequency band it is transmitted in, this is a very simple control protocol transmitting a command to a device by sending the serial number of the device followed by the command. 433 has extremely limited hardware requirements – an oscillator, sequence generator and antenna – and the resulting low cost has made it very popular. One of the most popular uses is for remote control power sockets, but a wide variety of other sensors, such as smoke alarms, door sensors, occupancy sensors, etc, are available with 433 transmission.

The other home wireless technology currently deployed is Wi-Fi. The method of choice for communicating with broadband routers, and supported by all tablets and smart phones, Wi-Fi is ubiquitous and capable of supporting IoT devices. Unfortunately, Wi-Fi is very power hungry, which holds it back in the IoT market.

From the consumer's point of view, two important areas of consideration for IoT devices are energy efficiency and cost. While Wi-Fi has until recently been significantly more expensive than 433 modules, new very low cost Wi-Fi modules are becoming available which makes Wi-Fi cost effective in an IoT scenario. This paper therefore examines the remaining factor – power consumption. The idea behind reducing power usage of IoT sensors and transmitters is, if the device can rely on its own batteries and be able to last a reasonable amount of time then the barrier of having to be near a power source would be removed, allowing greater flexibility and uptake. Previous research has looked into the power consumption aspect of wireless technologies designed for IoT [3],[4], but little is available on real world power consumption of 433, or of current Wi-Fi in an IoT scenario. This paper looks into this aspect with experiments carried out initially with the 433MHz AM protocol and with a developed prototype which has Wi-Fi capability.

A. 433 MHz AM

433 MHz is commonly used for local area communication, as frequencies within this band are internationally reserved for short range non-specific applications. This means many manufacturers produce and sell these components, and this, along with the inherent simplicity of the design, mean that a wireless interface can be added to an existing device very

cheaply. Examples of some of these products are remote controls, garage door fobs and smart LED lights.

Devices are available in AM and FM varieties, of which AM is the more common. Other frequencies, such as 868MHz, are also used. Transmission is extremely simple – on-off keying of a fixed pattern, and devices are usually designed so that they can go into a learning mode and pair by recognising a specific pattern from 433 protocol. This was first used for devices and later all the other technologies were evolved from this concept. An advantage of using 433 for low powered sensors is that the device can be configured so that it doesn't have to join a network before transmitting content, since the receiver is always listening. Having to connect to a network involves keeping the transceiver circuitry powered for longer. The disadvantage is that there is no confirmation of receipt (messages are usually repeated to increase reliability), although in many cases sensors have very limited storage and would not be able to do anything if a message was not acknowledged anyway. But systems like this rely heavily on having a base station or alternative receiver. 433 is short range, but again this is not a particularly significant disadvantage, as it limits interference with neighbouring properties.

It should be noted that there is very little security offered with 433. The simplistic nature of the protocol means that it is very easy to listen to communications and to spoof a transmitter, so anyone within the range of the 433 receivers would have the capability to manipulate them. However, such an attack requires physical proximity, and it could be argued that it is no greater threat than physical vandalism.

B. Wi-Fi

Wi-Fi has been a phenomenal success with the vast majority of households having a Wi-Fi network [5]. This prominence would make Wi-Fi the ideal candidate for a sensor device since all the infrastructure is already in place. System on chip (SoC) devices have been rapidly expanding in popularity, particularly within the areas of mobile electronics and embedded systems. Mainly due to their reduced power consumption achieved through tightly coupled integration between the CPU, RAM and other components, which would normally be separate chips. One of these SoC devices is the ESP8266 made by Expressif which integrates an Xtensa lx106 processor, RAM, standard interfaces with a Wi-Fi chipset. Providing a low cost device capable of running small programs with network capability.

Two devices which use the ESP8266 [6] chip are the ESP-01 [7], and the ESP-03. These share most of the same functionality, the ESP-03 offers more GPIO pins for interaction with other systems, it is also surface mount package. Where the ESP-01 has pin connectors to allow it to connect to a breadboard. Both packages are able to run both standard C code, Lua and most recently Python [8] [9] [10]. When transmitting data using a low power embedded system, it is best to transmit as little data as possible to save time and therefore power. This can be achieved by shortening the message, but a much easier change would be removing any unnecessary data such as a full HTTP packet [11]. For this reason, protocols which would use a very limited set of resources were designed. Two of these protocols are the Constrained Application Protocol (CoAP) [12] and Message Queuing Telemetry Transport (MQTT) [13]. CoAP is based on the popular Representational State Transfer (REST) model. It allows payloads of any data type, manages to only use a 4 Byte header, (about 13% the size of a comparable HTTP header.) MQTT is similarly lightweight, but is designed with a Machine to Machine network in mind: for example, communication between multiple temperature sensors and a central heating unit. CoAP is for single direct communication, usually between a user and a device.

Wi-Fi has the advantage that with having a full IP stack on the sensor and a Wi-Fi access point in the home, sensors can be designed to access a server outside the home over IP and therefore only the sensor has to be provided – the receiving infrastructure is already present. This contrasts with 433 where a receiver is necessary. Wi-Fi devices have to connect to the network before being able to transmit. This takes time and energy. One way to reduce this is to use a static IP address rather than relying on DHCP to allocate one. However, this comes at additional complexity for the user.

II. EXPERIMENT

A. 433 MHz AM

In order to get realistic results from a 433 device using an AM transmitter, it was required to create a device on a PCB board. An ATmega328P chip was programmed to output pulses on a digital GPIO pin. This pin was then connected to the data input on the AM transmitter, along with all appropriate power connections. Two 22 µF were also placed in series with a 16 MHz crystal to provide an external clock for the ATmega328P. Finally the reset pin was set high and connected to Vcc via a 10 kΩ resistor. The pulses generated by the ATMega were preconfigured to match a signal copied from another transmitter, which would toggle a wall power socket, with a 15 second delay between on and off. A PortaPow power monitor was inserted between the input power supply of the created system, and a power generator. The power monitor was therefore able to measure the current, voltage and power used by the system. The system was turned on, with the transmitter being in line of sight of the power socket. The current values used by the device were then recorded every third of a second as this was the limitation of the PortaPow resolution. These values are shown in Figure 1. During the experiment there were noticeable increases in the current draw moments before the wall socket would switch state, confirming the device was operating correctly and as expected.

Taking the average of the values, gives 21.1 mA while the device is transmitting, and an average of 15.9 mA while in idle mode. Since the device would spend most of its life in idle mode, the idle current had to be dramatically reduced. Turning off all unnecessary parts of the chip and putting it into a deep sleep mode, caused the power usage to drop to 360 µA.

B. Wi-Fi

To properly test the ESP-01's Wi-Fi capabilities it was decided to test both the C and Lua implementations of both lightweight protocols CoAP and MQTT. The experimental setup was consistent for each test, where the ESP-01 would be flashed and then powered via the PortaPow power monitor. The values were then recorded at third of a second intervals.

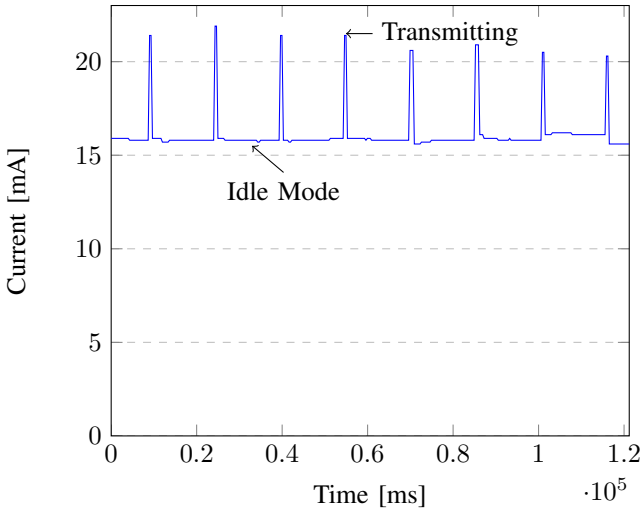

Fig. 1. Current Analysis of 433 MHz Transmitter Circuit

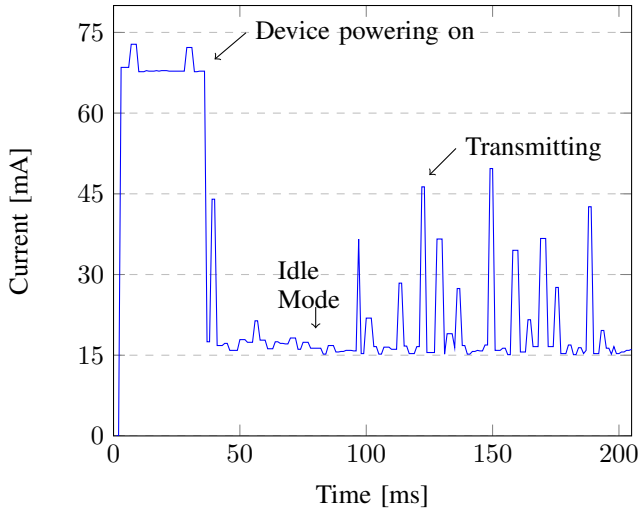

Fig. 2. Current Analysis of ESP-01 CoAP C Implementation

The ESP8266 supports 4 sleep modes: none-sleep/idle, light-sleep, modem-sleep and deep-sleep. When attempting to measure the current consumption in deep-sleep mode, the PortaPow recorded the result as zero. Instead it was measured with other means explained later. The values of current draw in the rest of these modes was recorded. It was considered to be useful to measure the current usage when transmitting. These values for each implementation are shown in Table I. All values are in milliamps.

TABLE I. CURRENT CONSUMPTION OF COMMON MODES IN EACH IMPLEMENTATION.

Mode	Lua		C	
	MQTT	CoAP	MQTT	CoAP
Transmitting	48.7	48.3	41.6	39.2
No Sleep	65.9	64.3	17.6	17.3
Light Sleep	17.4	17.5	5.3	5.4
Modem Sleep	17.9	18.1	6.1	5.5

This shows that during transmission all methods were roughly similar, but substantial changes can be seen between the Lua and C implementations for the rest of the operating modes. The slight changes in transmission power usage can be explained due to Lua running a virtual machine and using an interpreter rather than the precompiled machine code created by the C implementations. This will cause the code to run faster on the processor and therefore use less power. The dramatic changes between the power usage of the sleep modes is harder to explain, multiple tests resulting in the same values pointed to an issue. Since the sleep modes also use roughly the same amount of power as the normal none-sleep/idle mode in the C implementations, it is possible there is an issue within the framework. Between the C implementations the results were roughly the same, with the CoAP versions using slightly less power. The main cause of this is most likely to be that the MQTT messages use TCP which has to set up a connection before transmitting, where UDP is used in CoAP, which is able to instantly send. Since the C CoAP version used the least power, it was further developed. A graph of the life cycle of a series of transmissions is shown in Figure 2

To attempt to save further power the possibility of using an additional external low powered micro controller was introduced. This would give the functionality to toggle the state of the ESP8266 device using the CH_PD pin. This results in the ESP only using 0.5 µA when CH_PD is low. The device would then be powered on using an interrupt provided by an ATmega328P.

The ATmega was configured to output pulses consisting of high for 6000 ms - was found to be the smallest amount of time the device could successfully deliver a message to a server. Then low for 1000 ms to allow the device to power off. Simulating an environment where it would transmit and then go back to sleep.

The ESP-01 was flashed with C code to automatically connect to a saved network, then emit a CoAP PUT request to the server. The devices were both powered, and the PortaPow was configured to measure the energy use (mWh) over 100 cycles. The resulting value was then divided by 100 to give the power required for one transmission. This resulted in a value of 4.43383 mWh per transmission.

Knowing that using a static IP address would reduce the transmission time, the system was changed from using DHCP to allocate an IP address to using a static IP address. This meant the device was only powered for the length of the high pulse of 2500 ms. Running the experiment again using a static IP, resulted in a value of 4.38383 mWh per transmission. A comparison of these results with the DHCP is shown in *Figure* 3.

III. DISCUSSION AND CONCLUSION

Doing the previous tests showed the absolute values of each stage – transmission, idle, and sleep. However the results don't take into account the power used during transition between stages. To gain an understanding for the power used when waking the device up, to calculate the energy used per hour relative to the number of transmissions, graph was modified to show the differences between the ESP-01 and the 433 energy usage. This is shown in Figure 4.

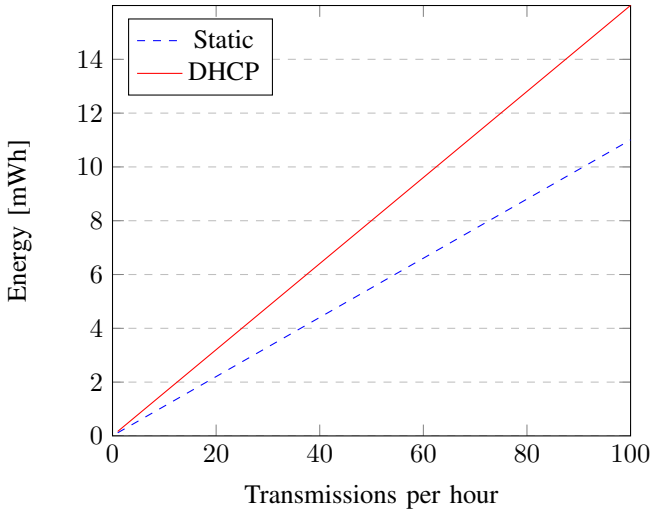

Fig. 3. Energy Analysis of ESP-01 Static Vs DHCP

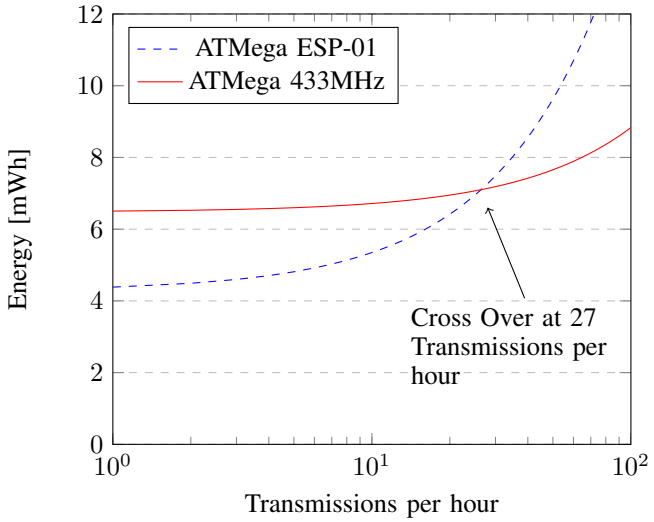

Fig. 4. Energy Comparison of the Number of Transmissions Per Hour

Based on the Figure 4 the results for both 433 and ESP-01 running with the ATmega processor, illustrates the fact that ESP-01 is capable of transmitting with less energy consumption. Up to 27 transmissions could be carried out by ESP-01 per hour. This has the benefit in a real world scenario, as the majority of sensing applications require less frequent transmission. This means that implementing ESP-01 as a IoT sensor device provides greater efficiency, with the additional benefits of reliability and security.

Although the ESP-01 uses more power than the 433 configuration while transmitting, because the Wi-Fi SOC device uses significantly less energy during idle mode, the overall energy usage for the ESP-01 is lower. This is clearly shown by the cross over point in the graph shown in Figure 4. This shows that for estimated use of less than 27 transmissions per hour, the ESP-01 would be more efficient, while for use greater than that the ESP-01 becomes exponentially less efficient. Additionally the ability to implement security onto a Wi-Fi platform is a huge bonus, since security is becoming a rapidly increasing issue. Even though the power usage of constant transmission with these Wi-Fi chips are greater, with the implementation of Wi-Fi disconnected and appropriate delay, power consumption is reduced. To this can be added the bonus of having an Internet compatible platform with the full IP stack of the ESP-01 (although security has to be considered), making even current Wi-Fi a practical choice for IoT applications.

The control microcontroller in our Wi-Fi sensor was the ATmega. While this is relatively efficient in sleep mode, it does not compare to the very low sleep currents offered by, for example, the Texas Instruments MSP430 range [14], which is capable of current drains of less than 500nA in standby mode (and less than 1000nA even when having a Real Time Clock active). While this difference in sleep current will have little effect on sensors which transmit several times a minute, for sensors which transmit only every few hours or even days it will make a significant difference, and so this lower cost design should be used in these applications. Our group is currently constructing such a sensor using an MSP430G2553 microcontroller instead of the ATmega.

REFERENCES

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Comput. Netw.*, 54(15):2787–2805, October 2010.

[2] C. Perera, P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos. Dynamic configuration of sensors using mobile sensor hub in internet of things paradigm. In *Intelligent Sensors, Sensor Networks and Information Processing, 2013 IEEE Eighth International Conference on*, pages 473–478, April 2013.

[3] A. Dementyev, S. Hodges, S. Taylor, and J. Smith. Power consumption analysis of bluetooth low energy, zigbee and ant sensor nodes in a cyclic sleep scenario. In *Wireless Symposium (IWS), 2013 IEEE International*, pages 1–4, April 2013.

[4] M. D. Prieto, B. Martinez, M. Montn, I. V. Guillen, X. V. Guillen, and J. A. Moreno. Balancing power consumption in iot devices by using variable packet size. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2014 Eighth International Conference on*, pages 170–176, July 2014.

[5] Ofcom. The Communications Market 2015.

[6] ESP8266. http://espressif.com/en/products/hardware/esp8266ex/overview.

[7] ESP-01. http://esp8266.co.uk/modules/esp-01/. [Online; accessed 10-March-2016].

[8] NodeMCU. https://github.com/nodemcu/nodemcu-firmware. [Online; accessed 10-March-2016].

[9] C Expressif API. http://bbs.espressif.com/viewtopic.php?f=46&t=1703. [Online; accessed 10-March-2016].

[10] micropython. https://github.com/micropython/micropython/tree/master/esp8266. [Online; accessed 10-March-2016].

[11] Tapio Levä, Oleksiy Mazhelis, and Henna Suomi. Comparing the cost-efficiency of coap and http in web of things applications.

[12] Constrained Application Protocol for Internet of Things. http://www.cse.wustl.edu/ jain/cse574-14/ftp/coap.pdf. [Online; accessed 17-March-2016].

[13] MQTT For Sensor Networks (MQTT-SN) Protocol Specification. [Online; accessed 17-March-2016].

[14] MSP430x1xx Family. http://www.ti.com/lit/ug/slau049f/slau049f.pdf. [Online; accessed 17-March-2016].