# DESIGN AND IMPLEMENTATION OF A CLOCK SYNCHRONIZATION ALGORITHM FOR STM32L MICROCONTROLLERS USING IEEE1588 PROTOCOL AND POWER CONSUMPTION MEASUREMENTS

***Luca Marturana, Sebastiano Milardo, Rosario Villari, Alessandro Sapienza***

Università degli Studi di Catania, Dipartimento di Ingegneria Elettrica Elettronica e Informatica
Catania, Italy

{lucamarturana,sebastiano.milardo,rsrvillari,alexsapienza}@gmail.com

**Abstract**

Clock synchronization is a very important feature in a wireless sensor network. This paper describes a system based on three STM32L microcontrollers performing clock synchronization using three XBee modules. This article also involves a power consumption measurement for the implemented system.

*Index Terms:* clock synchronization, STM32L, power consumption, XBee, ZigBee

## I. INTRODUCTION

The need for a common time has always existed in automated systems. This need is present in both wired and wireless systems. In fact the clocks trend, for various reasons, to drift from the reference clock. Small environmental changes, such as temperature, pressure and voltage of the battery may also increase this deviation. Implementing a clock synchronization algorithm is therefore an aspect that should not be underestimated. IEEE 1588, commonly known as Precision Time Protocol (PTP), offers a solution to the problem described above.

The goal of this project is the creation of a system composed of three STM32L microcontrollers, which will perform clock synchronization with the accuracy of the order of microseconds, communicating via the serial interface through three XBee modules. Our system will also measure the power absorbed by the XBee modules thanks to an appropriate circuit.

## II. SOFTWARE IMPLEMENTATION

*1) IEEE 1588*

Released in November 2002 (based on the work done by John Eidson [1] at Agilent Labs, IEEE 1588 specifics the hardware and the software needed to enable network devices to synchronize their clocks with the master.

The standard is applicable to LAN communications that support multicast. IEEE 1588 was initially based over Ethernet, but is not limited to it. In fact we implemented PTP on a wireless sensor network [2].

IEEE1588 protocol consist on a Master / Slave architecture, which is based on the exchange of a series of messages between a master clock and slave clocks. IEEE 1588 is able to synchronize systems with a clock that can vary in accuracy, resolution and stability, reaching an accuracy in the order of microseconds requiring a minimum exchange of messages and few computational resources [3, 4].

To achieve such a precision, it requires that the timestamps used must be generated by specific hardware as close as possible to the physical medium. It is important to note that this protocol is able to operate autonomously without maintenance. The entire network is structured according to a master/slave model. The master acts as a coordinator and distributes the clock.

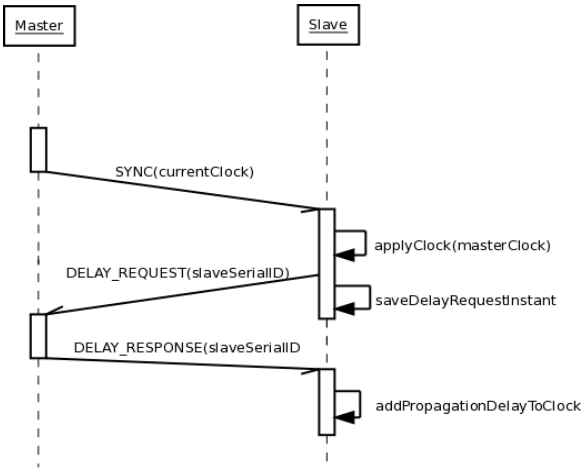



Fig. 1. IEEE 1588 protocol diagram

The master clock sends a message called "Sync messages", recorded at the instant of time in which such submission is made.

This value is transmitted in a second message called "follow-up message". The receiver uses its local clock to record the arrival time of the "Sync messages" and compares it with the time reference contained in the "followup message". The difference between these two values of time represent the propagation delay and the slave's offset.

The propagation delay is calculated by the receiver node by sending a message called "DelayRequest message" and recording the sending moment of this message. When the master clock receives a "DelayRequest" message, it records the arrival time and places it in a message called "DelayResponse message" and sends it to the slave clocks.

At this point, the slave clocks can change their clock in accordance with the master clock, having all the necessary information.

Because of the independent divergence of the clock involved in this procedure, it requires a periodic repetition to keep the clocks synchronized.

## III. HARDWARE IMPLEMENTATION

To implement the proposed system we used:

- 3 STM32L-Discovery boards;
- 3 XBee modules with their relative SimpleBoards;
- 1 PIC16F785 from MicroChip;
- 1 MAX232 adapting circuit;
- 4 breadboards;
- 1 USB adapter for XBee configuration;
- 1 32 segments DATAVISION display;
- 1 serial-USB adapter;
- various Wires, Resistors and Capacitors.

### *1) STM32L-DISCOVERY*

The STM32L-DISCOVERY is a board based on the STM32L152RBT6 microcontroller, including a STLINK/V2 link and integrating a debugging interface, a LCD, two LEDs, two push buttons, a linear touch sensor, and four touchkeys. The STM32L152RBT6 microcontroller is a low power 32bit MCU has 128 MB Flash, 16 MB RAM, 4 KB of EEPROM, RTC, LCD, timer, USART, I2C, SPI, ADC, DAC and comparators.

### *2) ZigBee*

ZigBee is a standard developed by the ZigBee Alliance, used in industrial environments and WSN to allow communications between low power consumption devices. It uses the 802.15.4 standard in the physical and MAC layer and defines a custom network and application layer.

The standard defines three different working frequencies: 868/915 MHz, using a BPSK modulation, and 2.4 GHz band, which can reach a data rate of 250 Kbits/s using a OQPSK modulation.

In the 2.4 GHz band, the standard 802.15.4 defines 16 channels of 2 MHz each. The modulation used at 2.4 GHz is an Offset Quadrature Phase Shift Keying (OQPSK).

### *3) PIC16F785*

The PIC16F785 from Microchip is a 8-bit flash memory Pic. The main feature of this microcontroller is the presence of two operational amplifiers used to estimate the voltage and current on the Xbee. Among the other characteristics we also have the ability to work at 2.0 or 5.5V. This micro also offers the nanoWatt technology to minimize consumption, A/D converters with 10 bit resolution and 2 high speed analog comparators [5].

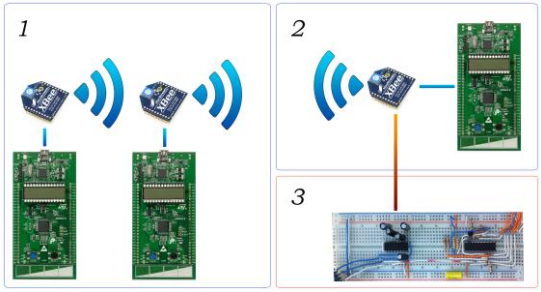



Fig. 2. Overview of the implemented system.
1 - Slaves, 2 - Master, 3 - Power measurement circuit

Connecting the development board to the XBee module is straightforward. In the EXT 5V pin we can find the voltage coming from the USB port, the GND pin is connected to the ground lines of the breadboard instead. The XBee module is then powered by the VIN pin and communicates with the STM32 using a USART port. The USART3 interface has been used. It transmits on PC10 (Transmission) and PC11 (Reception) pins, which are connected respectively to DIN and DOUT pins of the XBee module. To connect the XBee module with the breadboard an intermediate card having the same pin-spacing is needed. The intermediate card also converts the input voltage of 5V to 3.3V which are used by the XBee module. The power measurement circuit was built on a separate breadboard. It uses a PIC16F785 integrated controller and a MAX232 adapting circuit to convert a UART output to a RS232 output. This circuit implements a voltamperometer that allows to show the values of voltage and current. Four connectors link the voltamperometer to the remaining circuit. The first two should be placed in parallel to the voltage to be measured, the latter two should be placed in parallel to a resistance called "shunt" which is in series with the circuit from which the current is measured. The shunt resistor must be properly sized. In this particular case we used a 1 Ohm resistor.

| | |
|---|---|
| Max Input Voltage | 14.4 V |
| Min Input Voltage | 14.06 mV |
| Max Input Current | 1.09 A |

| | |
|---|---|
| Min Input Current | 1.06 mA |

The XBee module can work in the so called “transparent mode”. The STM controller simply sends data to the USART port and the Xbee module will route it to the proper destinations. Communications over USART port are really simple to use, except two cases:

- the serial port needs to transmit a byte and wait until it was sent broadcast before moving to the next byte;
- data can be received at any time.

To solve these two problems you can use the "active waiting" mode or the interrupt mode. In the first mode, the microcontroller keeps asking to the serial port if it has got something to work on. In the second mode the microcontroller is notified via interrupt when transmitting events occur and executes the code required to manage them. In our case we used the second mode.

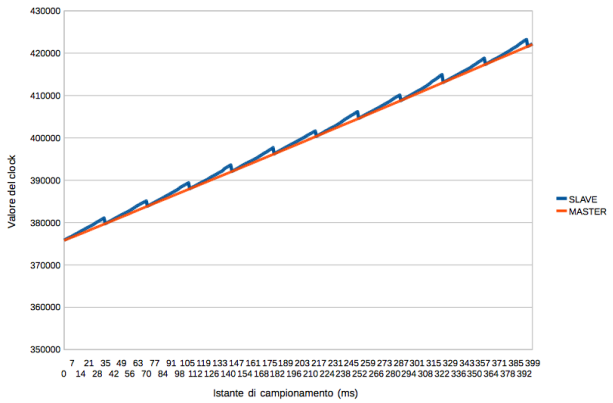
We will now see the code for PIC16HV785. The implemented firmware initializes various registers, set the clock speed and initialize the display. It enables the internal operational amplifiers, then enters an infinite loop that reads the voltage and current through AN0 and AN2. Finally, the microcontroller converts the values and displays it on the LCD. A USART port has been implemented on pin 3 using a technique called “bit banging”. This function is used to write a single character on the serial port, and using a for loop it sends all data to the MAX232 integrate.

A major issue concerned how to verify the correctness of system operations. It was initially thought to show the master clock time on the LCD devices. This approach, however, was not valid because it introduced too many delays. Subsequently it was thought to generate a square wave using the clock signal. In this way, the synchronization accuracy was shown by the phase difference between these two waves. This method showed a better accuracy but it was not possible to extrapolate data for a statistic use. Finally we sent a cable interrupt on pin PA0 using a device that periodically sent this signal.

## IV. DATA ACQUISITION

The data to be acquired are the master and slave clock and the power consumed by a XBee module [6].

As regards the clock synchronization, it was decided to use a further STM32 board as a reference node. In other words, this node is responsible for periodically send a signal on pin PA0 both on master and slaves, telling to sample the value of their clocks. It was decided to send such signal with a period of 250 ms. There were performed four runs of measurement, corresponding to different periods of synchronization between master and slaves (2, 4, 8 and 16 seconds). For each run 400 samples were taken.



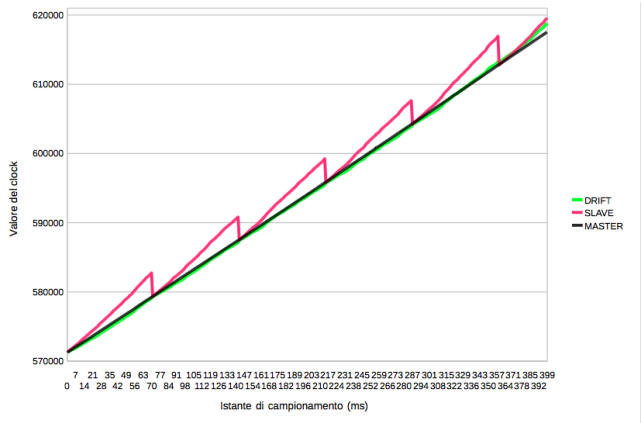

The purpose of these measurements is to estimate the skew between master and slaves and the drift rate.

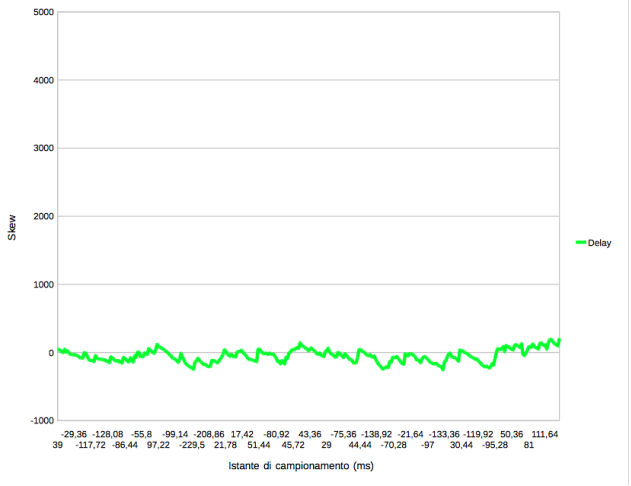
To measure the power consumption, it was chosen to sample the values of voltage and current every 20 ms. Then their product is transferred to a PC through a serial interface.

In this case it was decided to make two measurements. The graphs obtained are shown in the figures below.

### *A. Calculation of the drift*

Because of the excessive difference between clock frequencies the drift correction was performed offline [7].



From the data collected we represented two curves that show the evolution of the master and slave clocks. The trend of the slave, because of synchronizations, has a saw-tooth behavior. The calculation of the drift has been carried out using this formula:

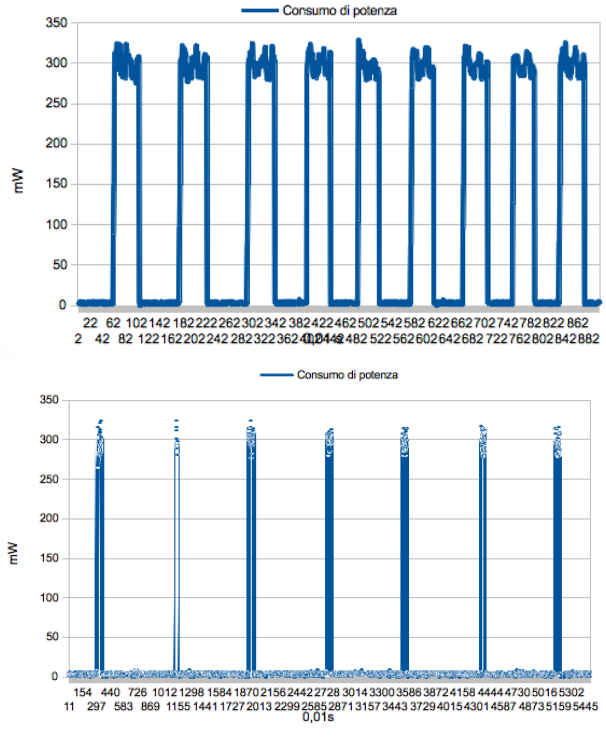
$$\alpha = \frac{C_{master}(t(i)) - C_{master}(t(i-1))}{C_{slave}(t(i)) - C_{slave}(t(i-1))}$$

To create the graph that shows the corrected value of the clock we used the following formula:

$$Clock_{corretto} = C_{slave} - (Fattore_{correttivo} \cdot i)$$

### B. *Power Consumption*

The following graphs regard power consumption [8].




## V. CONCLUSION

This paper has presented the implementation of a synchronization system. We used the IEEE1588 protocol with good results. The obtained accuracy is in the order of 1,025 ms every second. While power consumption measurement returned an average value of about 54,24 mW, approximately 300 mW during transmission periods and 4 mW in idle. The results obtained through measurements have shown that this system is ready for real applications.

## REFERENCES


[1] Mirabella, O.; Brischetto, M.; Raucea, A.; Sindoni, P.; , "Dynamic Continuous clock Synchronization for IEEE 802.15.4 based sensor networks," Industrial Electronics, 2008. IECON 2008. 34th Annual Conference of IEEE , vol., no., pp.2438-2444, 10-13 Nov. 2008 doi: 10.1109/IECON.2008.4758339

[2] Mirabella, O.; Brischetto, M.; Raucea, A.; , "Evaluation of clock synchronization protocols for Wireless Sensor Networks," *Wireless Days (WD), 2009 2nd IFIP* , vol., no., pp.1-5, 15-17 Dec. 2009 doi: 10.1109/WD.2009.5449682

[3] Mirabella, O.; Brischetto, M.; Raucea, A.; Bannò, F.; Caruso, N.; , "Improving the Dynamic Continuous Clock Synchronization for WSNs," *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society* , vol., no., pp.2126-2133, 7-10 Nov. 2010 doi: 10.1109/IECON.2010.5675292

[4] Eidson, J.; Kang Lee; , "IEEE 1588 standard for a precision clock synchronization protocol for networked measurement and control systems," Sensors for Industry Conference, 2002. 2nd ISA/IEEE , vol., no., pp. 98- 105, 19-21 Nov. 2002 doi: 10.1109/SFICON.2002.1159815

[5] Yussoff, Y.; Abidin, H.Z.; Rahman, R.A.; Yahaya, F.H.; , "Development of a PIC-based wireless sensor node utilizing XBee technology," Information Management and Engineering (ICIME), 2010 The 2nd IEEE International Conference on , vol., no., pp.116-120, 16-18 April 2010 doi: 10.1109/ICIME.2010.5477666

[6] Hyuntae Cho; Hyunsung Jang; Yunju Baek; , "Time synchronization via clock skew correction on ZigBee networks," Information and Communication Technology Convergence (ICTC), 2010 International Conference on , vol., no., pp.137-138, 17-19 Nov. 2010 doi: 10.1109/ICTC.2010.5674702

[7] Cox, D.; Jovanov, E.; Milenkovic, A.; , "Time synchronization for ZigBee networks," System Theory, 2005. SSST '05. Proceedings of the Thirty-Seventh Southeastern Symposium on , vol., no., pp. 135- 138, 20-22 March 2005 doi: 10.1109/SSST.2005.1460892

[8] Ascariz, J.M.R.; Boquete, L.; , "System for Measuring Power Supply Parameters with ZigBee Connectivity," Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE , vol., no., pp.1-5, 1-3 May 2007 doi: 10.1109/IMTC.2007.379344