# Porting Low-power Wireless protocols on the µ111 Real-time Operating System

R. Berguerand, L. Bergamini, E. Franzi

*With the rapid proliferation of the Internet of Things, the need for low power wireless protocols is expected to grow in the coming years. At the same time, the possibility of running such protocols concurrently with other applications, as made possible by an RTOS (Real-Time Operating System) like µ111, will pave the way to new possibilities in terms of network analysis, machine learning and multitasking, not to mention the ease of porting the protocol from one hardware platform to the other. This work presents a first effort to run two wireless protocols on the µ111 operating system.*

Low power wireless protocols have been one of the key enablers for the success of the Internet of Things. Standard solutions, such as Bluetooth Low Energy, are among the market leaders. One of the major reasons for their success is the fact that, apart from being available as standalone solutions running on a custom hardware and firmware, they are well integrated with commercial operating systems (Android OS or OSx, Windows and so on). This is not the case for many existing wireless sensor network protocols, such as the TDMA (Time Division Multiple Access) solutions or CSMA (Carrier Sense Multiple Access)-based WiseMAC [1] protocol, which has been available for many years, but has so far only been available running on dedicated hardware in an OS-less fashion (or with a very trivial OS structure). Recently, CSEM has ported WiseMAC and a novel TDMA protocol also developed by CSEM, to our µ111 RTOS [2]. This solution offers several advantages with respect to the previous OS-less solution, namely:

- Multiple applications can run in parallel on the same processor, enabling both communication and data analysis/decision processes to run at the same time.
- Encapsulation of applications in the OS reduces the risk of fatal errors and improves management.
- Porting from one hardware platform to the other is quicker and safer than in an OS-less environment.

The main challenge with respect to properly porting the WiseMAC and TDMA protocols, was to guarantee that time constraints would be respected, while limiting the usage of radio and CPU for low power operation.

The first step in the porting was to adapt the driver for the radio and the wireless protocols source code to the architecture of the OS, namely the respect of *mutex* (semaphores) to forbid concurrent access to shared resources. The concept of mutex is proper to operating systems: in case a resource is busy, the process is suspended until the resource is released, or if a certain amount of time has passed. The second step was to replace the OS-less scheduler that was formerly used to run the protocols with a scheduler managed by the central OS. To do this, inter-process communication was implemented as depicted in Figure 1. In this case, the process timer updates the software timers periodically (1 ms) using a precise signal (a signal is sent from one process to the other to provide notification of an event, i.e., the passing of 1ms in this case). The process timer informs the protocol of the timeout so that related actions, such as radio driver commands or protocol functions, can be executed with high time precision.

The first implementation was affected by two issues:

- in the timer process implementation, the CPU could almost never switch to the low power mode, resulting in high power consumption.
- timer resolution was 1ms, so nodes could experience up to a 1ms time shift, which is unacceptable for the TDMA protocol.

Both problems were solved by implementing a new solution using an internal CPU clock that can also run while the processor is asleep and which provides a lower timer resolution of 31 µs, which is acceptable also for the TDMA protocol.

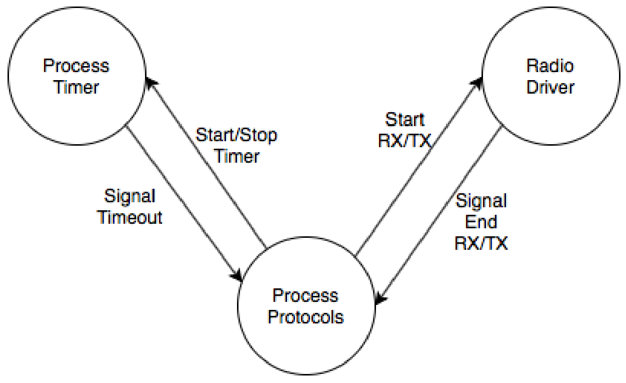

*Figure 1: Communication between processes.*

To validate the porting of the protocols, preliminary experiments were conducted on a small star-topology network composed of 7 CSEM custom boards featuring an NRF52840 processor. The performance of the protocols, in terms of packet delivery and latency, were found to be comparable to the OS-less version, suggesting that the use of an OS has little to no impact. Ongoing research is focused on measuring the impact on the overall power consumption running the OS vs. the OS-less version of the protocols. The ability to switch from one protocol to the other in real time has also been successfully validated, opening the way to future activities such as the possibility of automatically selecting the more suitable wireless protocol according to traffic conditions among the ones available on the OS. This idea was initially presented in WiseTOP [3] , but the use of an OS will allow the implementation of a more advanced traffic analyzer module that will be able to analyze the traffic patterns thanks to machine learning techniques, and automatically switch to the most suitable protocol in real time.

[1] A. El-Hoiydi Amre, J.-D. Decotignie (2004), "WiseMAC: An ultra low power MAC protocol for the downlink of infrastructure", Wireless Sensor networks, International Symposium on Computers and Communications.

[2] E. Franzi (2019), UKOS-III an RTOS for embedded system.

[3] L. Bergamini, J.-D. Decotignie, P. Dallemagne (2018), "WiseTOP: a multimode MAC protoocol for wireless implanted devices", 41-50. 10.1145/3273905.3273919.