

DESCRIPTION OF **Software Verification for TinyOS Applications**  
CURRENT (within EPSRC<sup>1</sup> UbiVal: *Fundamental Approaches to Validation of Ubiquitous Computing Appli-*  
PROJECT *cations and Infrastructures*)

Sensors and wireless sensor networks are increasingly playing a role within e.g. Pervasive Healthcare. Body Sensor Networks are being designed for prophylactic and follow-up monitoring of patients in e.g. their homes, during hospitalization, and in emergencies. In such cases, *software safety* and *reliability* are at their most important.

Given that sensors are embedded devices, their programming is done in inherently unsafe languages—usually flavours of C. With TinyOS the mainstream sensor operating system, and NesC its main programming language, standard programming errors are, unlike in the case of high-level programming, silent and extremely disruptive. *Memory violations* (e.g. writing a null pointer) in a TinyOS application may modify the content of registers, and even send commands to the peripheral pins. Furthermore, the *application's logic* may have been programmed incorrectly, leading to unexpected sensor behaviour after deployment in the field; for example, not accounting for erroneous packet lengths in incoming network packets may cause the application to block.

As such, measures must be taken to increase the safety of critical sensor applications to a certain *guaranteed standard*. We provide ([1-2]) the first method to allow the programmer to debug a TinyOS application without deploying it. To show that it adheres to an expected functionality, we instrument TinyOS code with *assertions* which should hold whenever they are reached, and input the resulting code into a fully-automated *verification toolchain*, which finally returns a program trace leading to an assertion violation, if any.

Future work aims at improving on the scalability issues of the verification scheme (e.g., by making the technique *compositional*, to allow component-local verification). Also, we are looking at allowing the programmers to state—in a language close to natural language—overall, high-level *specifications* for the behaviour of an application, and then automatically translate the specification into the required low-level assertions which then instrument the program.

[1] “Software Verification for TinyOS”. Doina Bucur and Marta Kwiatkowska. Submitted. Available on webpage.

[2] “Bug-Free Sensors: The Automatic Verification of Context-Aware TinyOS Applications”. Doina Bucur and Marta Kwiatkowska. Proceedings of the 3rd European Conference on Ambient Intelligence (AmI 2009), Springer Lecture Notes in Computer Science (*LNCS*), Volume 5859, Nov 2009.

KEYWORDS Wireless sensor networks, TinyOS, context awareness, reliability, debugging, software verification, pervasive healthcare.

---

<sup>1</sup>Engineering and Physical Sciences Research Council. The UK Government’s leading funding agency for research and training in engineering and the physical sciences.