Demo Abstract: Software Factory for Wireless Sensor Networks

T. Naumowicz, B. Schröter, and J. Schiller

Freie Universität Berlin, Institute of Computer Science
{naumowic, schroete, schiller}@inf.fu-berlin.de

***Abstract*—Wireless Sensor Networks (WSNs) are not widely used in field research because of poor tool support and high complexity. To address this issue, we designed and developed a Software Factory for WSNs that hides the complexity of software development for embedded systems and exposes a visual domain-specific modeling language. Our solution makes WSNs more attractive as a tool for researchers from outside the computer science field and enables a wider adoption of WSNs in field research.**

We will demonstrate our Software Factory in a real-world scenario replicating our latest WSN deployment.

***Index Terms*— Wireless Sensor Network, Software Factory, Domain-Specific Languages**

I. Introduction

The vast majority of publications in the field of WSNs refer to environmental or habitat monitoring as the typical application for WSNs. You could expect that WSNs are already widely used in field research because they are often advertised with high sensing accuracy, long runtimes, and easy deployment. However, this is still not the case.

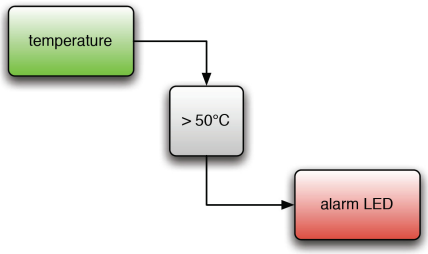

Figure 1: A conceptual example of a simple data flow

Research in the area of WSNs has previously focused on hardware design, self-organization, various routing algorithms, or energy saving patterns. This trend is already changing [1, 2] but the available tools typically target experienced software developers and not researchers from outside the computer science field. As of today, domain experts such as environmental scientists need extensive support from hardware and software engineers during planning, deploying and management of WSNs. This makes a wide adoption of WSNs in real-world scenarios difficult and, in combination with poor tool support, makes such adoption slow and error prone.

We have designed and developed a Software Factory to address this gap. Software Factories are model-driven development environments. Our solution is based on a data-centric programming model where data flows are used to describe how data should be processed (e.g. Figure 1). We will refer to our Software Factory as *Flow*.

We prototyped *Flow* for the resource constrained ScatterWeb WSN platform MSB-430 [3] with 55kB flash memory and 5kB RAM. The prototype is available for download and evaluation [4].

II. The Software Factory in Brief

Flow provides a visual editor for modeling of applications for WSNs and a native code generator. *Flow* focuses on the visual representation and execution of data flows at a very high abstraction level (e.g. Figure 2). It is not a visual programming language for describing program control flow – in such scenarios the textual representation is very often easier to be written and understood.

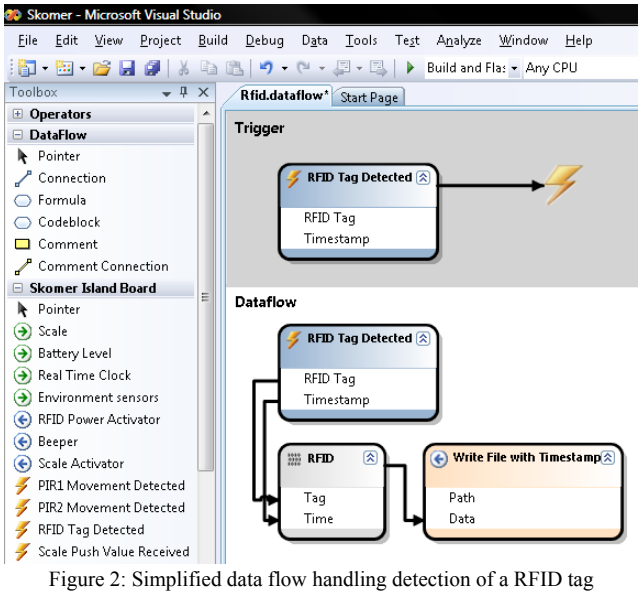

Figure 2: Simplified data flow handling detection of a RFID tag

Our Software Factory does not leverage a Virtual Machine. With this approach, we expect to achieve higher execution performance. The use of native C application code has an important advantage: it makes the

implementation of drivers for *Flow* less time-consuming and allows easy integration of specialized C code blocks to handle advanced cases.

Domain experts use the provided visual Domain-Specific Language (DSL) to define data flows and thus specify the behavior of nodes in the network. The modeled data flows are validated at design time and the user is provided with visual feedback. After successful validation, application code is generated. With *Flow,* domain experts no longer need to develop or maintain native C application code.

III. Real-World Scenario: Skomer Island

We deployed a WSN for environmental monitoring on Skomer Island in the United Kingdom in March 2007 [5]. The WSN was used to monitor the behavior of Manx Shearwaters, a burrow nesting seabird. We recorded birds' activity around entrances to the burrows as well as the identity of tagged individuals. In addition, we collected high resolution environmental data about the temperature and humidity inside and outside the burrows. In March 2008 an updated version of the system was deployed.

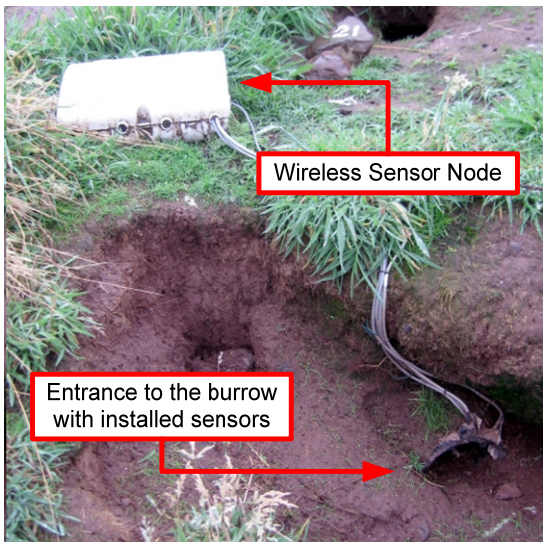

Figure 3: Sensor Node deployed on Skomer Island in 2008

During the deployments we observed that our expertise in software development for embedded systems was required in every phase of the project. The research goals shifted often and we had to update the deployed applications each time. High complexity of required changes prevented domain experts from modifying the application code themselves. The dependency on us was a big disadvantage – it resulted not only in delays but also distracted scholars in their research.

We use the Skomer Island scenario in our research to investigate how high levels of programming abstraction can be used in order to make WSNs more attractive as a tool in areas outside the computer science field.

IV. Demonstration Setup

The demonstration is built from a subset of our real-world WSN deployment to the Skomer Island in 2008. The functionality of the WSN from the deployment is now modeled using *Flow*. The WSN is enabled, i.e. conference attendees can interact with the installed demonstration. Visitors are encouraged to modify data flows using *Flow*, deploy the generated code, and evaluate the results.

The demonstration consists of the following components:

1) Wireless sensor node used during the expedition to the Skomer Island. The node is in its original packaging from the deployment (see Figure 3) and it's equipped with environmental sensors, movement detectors, and a RFID reader.
2) Wireless data logger node used as a data sink on the network. The node collects data from the network and stores it on a SD card.
3) Wireless gateway node attached to a notebook. The node provides access to the network for .NET based applications executing on the PC.
4) Notebook with installed *Flow* and .NET development environment. The notebook is used to model data flows and to present the capability of easy integration of WSNs with end user applications.

Interested visitors use the provided notebook to modify existing data flows and design their own ones. This is how they are able to modify the behavior of the network. *Flow* generates native code for the sensor nodes based on the designed data flows and compiles it for the selected hardware platform. Proxy libraries for the PC are generated as well. This enables easy access to the data in the WSN from the PC.

The generated code handles network discovery, gateway resolution, data communication and marshaling between the WSN and the client application. Visitors can access data and activate data flows using easy to handle .NET objects that represent wireless sensors in the network.

V. Conclusions

The proposed demonstration illustrates the achieved level of programming abstraction. Data flows can be easily created, read and modified by researchers with no experience in development for embedded systems.

In our opinion, the proposed solution has the potential to shorten the development cycles dramatically during WSNs deployments in field sciences, to reduce the dependency on hardware and software engineers, and to lead to a wider adoption of WSNs outside the computer science field.

References

[1] Martinez, K., Padhy, P., Riddoch, A., Ong, R., Hart, J.: Glacial Environment Monitoring using Sensor Networks. In: Proceedings of the Workshop on Real- World Wireless Sensor Networks (REALWSN'05), Stockholm, Sweden (2005).

[2] Talzi, I., Hasler, A., Gruber, S., Tschudin, C.: PermaSense: Investigating Permafrost with a WSN in the Swiss Alps. In EmNets 2007, Cork (2007).

[3] ScatterWeb, http://cst.mi.fu-berlin.de/projects/ScatterWeb

[4] Flow, http://cst.mi.fu-berlin.de/projects/flow

[5] Naumowicz, T., Freeman, R., Heil, A., Calsyn, M., Hellmich, E., Braendle, A., Guilford, T., Schiller, J.: Autonomous Monitoring of Vulnerable Habitats using a Wireless Sensor Network. In REALWSN'08, Glassgow, UK (2008).