Przemysław ZIELONY[1], Emil MICHTA[2], Krzysztof PIOTROWSKI[1]
[1]IHP – Leibniz-Institut für innovative Mikroelektronik, Im Technologiepark 25
[2]Uniwersytet Zielonogórski, Instytut Metrologii Elektroniki i Informatyki

TINYDSM: DATA-ORIENTED MIDDLEWARE FOR LOW-POWER IOT DEVICES

Developing applications for distributed measurement and control systems based on wireless sensor networks or, general, on the Internet of Things (IoT) devices requires knowledge and experience. This task can be simplified by approaching it in a modular way. The approach for application development based on the tinyDSM middleware, proposed in this paper, is data-centric and supports reuse of code, but it also simplifies the program debugging. It will be applied in the SmartRiver project.

TINYDSM: ZORIENTOWANA NA DANE WARSTWA POŚREDNIA DLA ENERGOOSZCZEDNYCH URZĄDZEŃ INTERNETU RZECZY

Tworzenie aplikacji dla rozproszonych systemów pomiarowych i sterujących opartych na bezprzewodowych sieciach czujników lub ogólnie na urządzeniach Internetu rzeczy (IoT) wymaga wiedzy i doświadczenia. Zadanie to można uprościć, podchodząc do niego w sposób modułowy. Tworzenie aplikacji wykorzystujących warstwę pośrednią tinyDSM, zaproponowane w niniejszej pracy, jest skoncentrowane na danych i wspiera ponowne wykorzystanie kodu, ale także upraszcza uruchamianie programu. Zostanie ono zastosowane w projekcie SmartRiver.

1. INTRODUCTION

Modular application development simplifies the process of programming wireless sensor network applications. Simple modules with limited functionality allow for better testing and code reusability. Limiting the complexity of individual blocks enables keeping the code simple and concise. It also simplifies the program debugging. Then, the combinations of such simple functional blocks can be used to address specific and complex application scenarios.

The approach proposed in this paper focuses on the tinyDSM middleware [1] that will be used as a base for the distributed application used in the wireless sensor network (WSN) to be developed in the SmartRiver project. However, in order to fit better to scenarios relevant for the project, the original middleware was modified and optimized. In its original form it was meant for dense mesh networks with intense data replication [2] and the typical environment monitoring scenarios work with more sparse networks. These networks are typically based on tree topologies with limited data redundancy within the network to reduce the energy consumption.

The following sections present the current concept of tinyDSM and introduce other modules that make up the target application. The article concludes with a perspective on future developments.

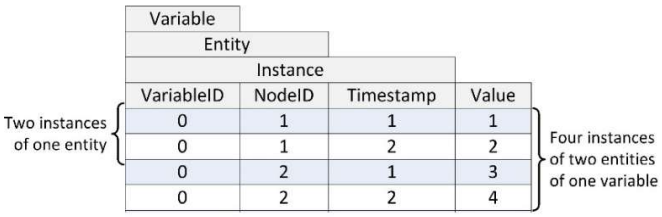
2. THE TINYDSM MIDDLEWARE

The tinyDSM middleware allows defining local variables that can store application data in a structured way and enables building the application code around the middleware based on events related to that data. The data is stored in tuples, each representing an instance of a given variable and containing the value together with the metadata that describes it (see Fig. 1).

Rys.1 Struktura, w której przechowywane są zmienne tinyDSM
Fig.6 Structure used to store tinyDSM variables

The variables are defined as static (created at compilation time). Due to that, the middleware implementation is tailor-cut and optimized to the application needs, but the flexibility is limited. An

option to handle this lack of flexibility is to define some of the variables in a general way so that they are usable for different purposes. The metadata contained in the tuples, allows addressing the data according to its meaning (*VariableID*), its origin (*NodeID*) and the time it was created (*Timestamp*) (see Fig. 2).



Rys. 2 Adresowanie danych w tinyDSM
Fig. 7Data addressing in tinyDSM

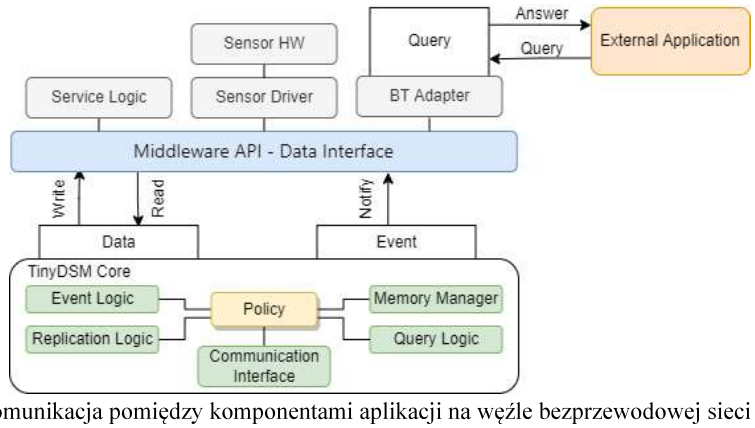
The definitions cover the variables and events to be detected on these variables, see (1) and (2), respectively. In each case, a variable is created to store the written values or the results of expression evaluation (always Boolean type). For these values, the definitions specify the desired behavior within the application – policy.

distributed [*policy parameters*] **type** name[=init_value or expression]; (1)

event [*policy parameters*] name = expression; (2)

All the variable definitions for a given application are located in the definition file. This definition file is precompiled to create the tinyDSM C-code, to be finally compiled into the executable image for the wireless sensor nodes.

The tinyDSM middleware provides an API for sensor drivers and service (or application) logic – the Data Interface. Applications based on tinyDSM middleware can be realized as a single component or as multiple components accessing the data in the middleware. In the latter case, each component is accessing the middleware independent from the others, although they all might be notified about the events detected on the defined variables. By that, the application components share the data over the middleware. An example architecture of an application is shown in Fig. 3.



Rys.3 Komunikacja pomiędzy komponentami aplikacji na węźle bezprzewodowej sieci sensorów
Fig.8 Communication between application components on wireless sensor network node

The middleware is agnostic to the network topology and in the original; the nodes could replicate the data of their neighbors in a defined manner. Compared to the original approach, the current tinyDSM middleware is simplified with respect to the data replication. In the SmartRiver project application and in many Smart City and environmental monitoring scenarios the network topology is usually a tree and replicating data is thus translated to sending it to the root of the tree – the gateway. The data is then

stored in the middleware layer above the tinyDSM – the smartDSM (ranging between the Edge and Cloud levels). The networking layer within the wireless sensor network below the tinyDSM middleware is the Kangaroo multi-hop network protocol [3].

In a typical sensing and control application implemented in a flexible way, we can identify two types of variables: measurement variables and configuration variables. Measurement variables are those storing the values collected from the sensors connected to the node, but also those that monitor the operation of the node such as current flows, processor temperature or time. Configuration variables are storing values related to the operation of the node, such as the frequency of taking the measurements or sending these to the gateway. In the tinyDSM middleware, we can specify the way in which each variable is handled by the system, e.g., if the specific measurement variable will have a history of values and how long it should be, or if they should be stored only until sent away to the gateway. We can also use filters that decide which values are to be stored and which should be sent. For configuration variables, it makes also sense to set them from outside to influence the node operation.

The event variables enables defining expression-based conditions that can help detecting specific and multi-parameter events that can trigger additional actions. For example, if a sensor measurement goes out of a predefined range, an alarm notification is generated, or, depending on the energy level of the node and the user's needs, it is possible to adapt the sampling rate to trade-off between the high reading frequency and the long battery life.

3. SENSOR DRIVERS, SERVICES AND ADAPTERS

Sensors drivers, services and adapters are the application components. They access the data in the middleware and realize the tasks of the application on the node.

Sensor drivers access the sensors interacting with the environment and push the read values into the middleware that can further process the data and triggers events, if necessary. The tinyDSM middleware notifies the sensor driver if the storing was successful so that the driver can retry the access or implement custom logic. By that, each sensor driver might have its functionality limited to reading the sensor and storing the value. This improves the code reusability and modularity of the applications. That is because the individual drivers can be implemented to work independent from each other and different driver combinations can compose an application. Further, the sensor driver can be controlled by configuration variables. Notified about the changes of the variable controlling the sampling frequency, the driver can adapt it as the value changes.

Services are like algorithm implementations that use the data stored in the middleware to generate new data. Thus, they provide entirely data processing functionality.

Adapters are drivers that translate the data between the tinyDSM Data Interface form and any other. The Bluetooth Adapter is an example here, as it translates the data accesses to an external interface for a mobile application.

4. BLUETOOTH ADAPTER

This adapter makes use of the variables and provides a direct interface to the data stored at each node. This approach implements the concept we proposed for sensor nodes equipped with Bluetooth module [4], but in the modular way and based on the tinyDSM middleware.

Variables are marked in the definitions as those being configuration variables and those being measurement variables. Accessing these with the smartphone application allows configuring the nodes as well are exporting the measurement data directly in the field or presenting these in a user-friendly way.

The smartphone application communicates with the node via the Bluetooth 5 standard. The application connects to the selected node and mutual authentication takes place. This is to prove that a network node is truly part of the network and that the user belongs to a particular group and has appropriate access rights.

The adapter on the node provides a service for data exchange in JSON format. The smartphone sends predefined commands to the node, which are interpreted by the adapter. The user can read all available variables with the LIST command. After sending the command, the available configuration and measurement variables with information about each variable are displayed in the dashboard tab on the smartphone. At this point, the user can select the available operations on variables. For example, display single value or a range of values of the selected variable (GET command), set the value of the configuration variable (SET command) or delete historical values (DELETE command).

The mobile application also provides a remote software upload service, so there is no need to connect an external programmer by wire in order to reprogram the node. This is a very useful feature, considering that the nodes are usually placed in enclosures that are located in hard-to-reach places.

5. CONCLUSIONS

The tinyDSM middleware is a tool to split the data plane from the logic plane in the wireless sensor network applications. It also supports modularity and code reuse, i.e., using the Data Interface and defining a set of variables allows implementing reusable sensor drivers and logic blocks. These components can then be connected on top of the middleware to create more complex applications.

The presented set of components, together with the Bluetooth adapter accelerates the application development, but also simplifies the deployment and maintenance of the sensor network.

The middleware platform core is implemented in the C programming language, so that most of the microcontrollers can embed it without additional effort. Given the simplicity of the basic C language, it is easy to embed the code of the middleware within any programming framework (operating system) on the sensor nodes.

The current scenario in the SmartRiver project includes 4 types of measuring stations. Multiple sensors are available at these stations to measure the parameters of air, water and soil. We plan to apply intensive tests in this scenario.

ACKNOWLEDGMENTS

This work was supported by the European Regional Development Fund within the BB-PL INTERREG V A 2014-2020 Programme, “reducing barriers – using the common strengths”, project SmartRiver, grant number 85029892 and by the German government, under grant 01LC1903M (AMMOD project). The funding institutions had no role in the design of the study, the collection, analyses, or interpretation of data, the writing of the manuscript, or the decision to publish the results.

REFERENCES

1. Piotrowski, K., Langendörfer, P., Peter St., *tinyDSM: A Highly Reliable Cooperative Data Storage for Wireless Sensor Networks,* in Proc. of the IEEE International Symposium on Collaborative Technologies and Systems (CTS), 225 (2009).
2. Piotrowski, K., *Distributed Shared Memory and Data Consistency (for Wireless Sensor Networks - Assessment of the Feasibility),* ISBN 978-3838131597,SüdwestdeutscherVerlagfürHochschulschriften (2012).
3. Maj, J., Piotrowski, K., Michta, E., *Kangaroo: Multi-Hop Protocol Stack for Smart City Sensor Networks*, in Proc. 14th Conference on Measurement Systems in Research and in Industry (MS 2022).
4. Piotrowski, K., Koropiecki, I., Zielony, P., Powroznik, P., Szulim R., *Mobile Application to Support Maintenance of Wireless Sensor Networks*, in Proc. 13th Conference on Measurement Systems in Research and in Industry (MS 2020), 81 (2020).