CANopen IMPLEMANTATION TO WIRELESS SENSOR NETWORK

Markus Madetoja and Reino Virrankoski

Communications and Systems Engineering Group
Department of Computer Science
University of Vaasa, Finland

ABSTRACT

Field buses are widely applied in the control of mobile machines. They enable us to build embedded control systems, where the sensors and actuators are connected to each other by the bus. ISOBUS is the most commonly used bus standard for Control Area Network (CAN) between tractors and implements in agriculture and forestry. Once the number of sensors and actuators increases in the implement side, a combination of ISOBUS and CANopen can be applied. CANopen is a communication protocol and device profile specification for embedded systems used in automation. In addition to agricultural applications, it is used in mobile machinery, in mines and forestry, in power systems such as substation automation and wind turbines, and in many other kinds of industrial systems, which require embedded automation to be able to operate. However, there are always places which are not reachable by cables. These parts can be mobile, or the mechanical conditions can otherwise be too harsh for the cables. A wireless extension of CANopen enables us to connect these parts to the CAN-based control system. It will also enable us to connect several mobile machines to the same control network.

In this paper we present the implementation of CANopen protocol to wireless sensor platform called the UWASA Node. The UWASA Node is a modular and stackable wireless sensor node, which is designed to fill the requirements of wireless automation. There are enough memory and computation power to run some computation required by the control applications in the node. The modular hardware architecture and the protocol software architecture enable a relatively easy integration of different kinds of industrial sensors depending on the measurement needs. On the other hand, the UWASA Node can operate just as a low energy consuming router in a mesh type of network, if such architectural solution is preferred. The CANopen implementation to UWASA Node is done by

using the open source stack CanFestival. It is tested by a set of standard CANopen functionality tests in cooperation with TK-Engineering Ltd. and CANopen competence center C3 Vaasa. Certain errors and operational shortages are detected and fixed based on the test results. Finally, the hybrid control system consisting of CANopen fieldbus and the wireless sensor network is tested and its performance in terms of capability and reliability is evaluated based on the test results. In the conclusions we also evaluate the feasibility and possible limitations of CanFestival compared to some other CANopen protocol stacks.

Keywords: Wireless CAN, CANopen, Wireless Sensor Network

INTRODUCTION

As a part of the increased efficiency, also the degree of automation is increasing in the work machines. In agriculture, tractors are becoming equipped with embedded computers and electronic devices. Advanced automation solutions require real-time control, which utilizes real-time measurements. To be able to do so, a fast and reliable communication is required. If there are several sensors and actuators in different locations in one implement machine, an embedded control system is needed to control the machine operation.

Field buses were developed to enable fast and reliable communication between different entities in the machine. The CAN bus (Control Area Network) was introduced in 1986 by Robert Bosh GmbH. It was first widely applied in the car industry from which is spread to other industrial areas. ISOBUS (ISO 11783) and CANopen are both higher layer communication protocols, which are using CAN bus as a physical medium. Currently ISOBUS is the most common way to connect the implement and tractor devices, but once the implements are becoming more complicated, a hybrid system where CANopen is used on the implement side and then interfaced to tractor over ISOBUS will be one option for the control and communication architecture (Hensler, 2012).

When thinking one step further, there are always such places in the machines, which are not reachable by cables. These can be, for example, rotating parts or locations with such a level of vibrations, stretching forces, or dust, that it is not reasonable to build cable connections there. Moreover, if we want a huge amount of measurements from several locations for some purpose like condition monitoring, the price of the cabling may exceed the reasonable limits. These challenges can be solved by wireless communication.

During the latest decade, wireless sensor and actuator networks have been developing rapidly, and many new WSN applications have entered to the markets quite recently. The most common communication protocols the wireless sensor networks (WSNs) are currently using are IEEE 802.15.4 and IEEE 802.15.4a. They operate on the license-free industrial, scientific and medical (ISM) bands, which enable relatively short-range communication with limited data transmission capacity. As a consequence, the WSNs which are using IEEE 802.15.4 or IEEE 802.15.4a communication protocols cannot fill all requirements of the complete

CAN standard. Thus, instead of going completely wireless one must use a hybrid system, where the backbone network is cabled but there can be wireless extensions to such parts, which are not otherwise reachable. To be able to make such wireless extensions, a gateway between CAN bus and WSN is needed. In this work we implemented CANopen protocol stack to wireless sensor platform called the UWASA Node. The implementation was done with open source CANopen stack CanFestival and its functionality was evaluated by using standard test tools.

The rest of the paper is organized as follows. In Chapter 2 we present an overview of the UWASA Node and in Chapter 3 about CANopen. The implementation of CanFestival stack and the test tools which are used to test the implementation are explained in Chapter 4. System performance is evaluated in Chapter 5, and finally we present the conclusions and point some directions to future work in Chapter 6.

THE UWASA NODE

The UWASA Node is a modular and stackable wireless sensor node developed for wireless automation (Yigitler, 2010), (Virrankoski, 2012). The software and hardware architecture of the node support the modular structure so that depending on the needs of some particular application, the node can easily be extended to powerful actuator. Alternatively, it can be used as a low power consuming relay node in the mesh type of network, or it can be something between these two examples. The hardware architecture of the node is presented in Figure 1. There are five types of modules in the hardware: main module, power module, active slave module, passive slave module and extension slave module. The main module is always needed, since it contains the main controller, RF controller and wireless transceiver. Power module is usually used because it carries the battery, but if energy harvesting or some other external power source is used to power up the node, the power module is not necessarily needed. The number and type of the slave modules used in each node depends on the particular needs; there can be from zero to many of them (Yigitler, 2010).

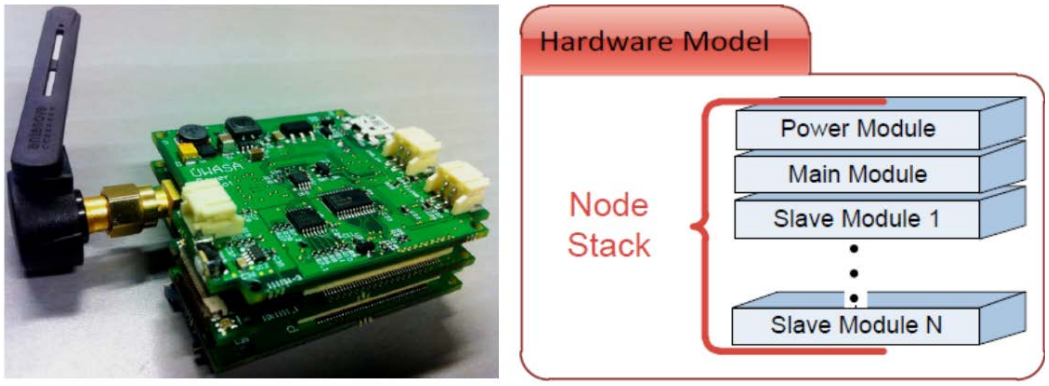

Figure 1. The UWASA Node and its hardware architecture.

Active slave module contains its own slave controller. One or several sensors can be mounted into it via sensor/actuator interface. Sensor relater data processing and driver issues can be handled by the slave microcontroller. Passive slave module has only sensor/actuator interface but not its own microcontroller. Extension slave module contains extension hardware or connector to the extension hardware. In addition to these node hardware modules, there is also a development board which simplifies the development phase by providing the software development interfaces for UWASA Node Main Controller and RF Controller (Yigitler, 2010).

We have been using UWASA Node in several applications related to industrial automation, agriculture and tactical communication systems (Virrankoski, 2012), (Björkbom et al., 2013), (Virrankoski, 2013). The processing power of the node enables us to run computation in a distributed or locally centralized manner in the network. It is also possible to develop advanced algorithms for network management, such as power saving and time synchronization. Reliable communication range depends a lot on the surrounding radio environment and follows the specifications of IEEE 802.15.4. Roughly speaking, the communication range it is at least tens of meters, in good line of sight conditions even more. It is also possible to change the radio from IEEE 802.15.4 to something else without changing the complete node design, but then the power consumption and the license legislation must be considered.

CANopen

The CANopen is a CAN-based higher layer protocol (CiA, 2013e). Its specifications are maintained by a nonprofit organization CAN in Automation (CiA), which is headquartered in Germany (CiA, 2013c). CANopen specification states following: “A CANopen device can be logically structured in three parts. One part provides the CAN interface and another part provides the device's application, which controls e.g. the Input/Output (I/O) lines of the device in case of an I/O module. The interface between the application and the CAN-interface is implemented in the object dictionary. Object dictionary is unique for any CANopen device.” (CiA, 2013b).

Each CANopen device operates under CANopen protocol stack, which is used for gaining access to object dictionary from the CAN bus. The object dictionary consists of different objects which can be accessed via the network as specified in the CiA301 specification. Each CANopen stack has its own type of object dictionary implementation and different implementations are usually not interchangeable. The CANopen object dictionary is divided into different sections which have different objects. According to CiA specification, each object within the object dictionary is addressed using a 16-bit index and an 8-bit sub-index (CiA, 2013b).

IMPLEMENTATION

CanFestival CANopen stack

Each CANopen device operates under CANopen protocol stack which provides access to the CANopen object dictionary (CiA, 2013b). There are many different CANopen object dictionary implementations, closed- and open source. There are at least two open source CANopen protocol stack implementations, CanFestival (CanFestival, 2013a) and CANopenNode (CANopenNode, 2013). Both of them provide tools to make CANopen object dictionary. We decided to use CanFestival stack in our implementation. CanFestival is an open source CANopen protocol stack, which development has been started on 2001. It has been licensed under LGPL and GPL (CanFestival, 2013b).

By choosing an open source stack for our CANopen implementation to UWASA Node, the implementation remains also open source without a need to pay licensing fees. It was also part of our interests in this work to test the feasibility of open source implementation.

Test Tools

The CANopen Electronic Data Sheet (EDS) file in the UWASA Node describes every CANopen object dictionary object, which exists in the UWASA Node CANopen object dictionary. The implementation test was based on EDS file and on CiA301, CiA401 and CiA305 specifications. Also a series of performance tests were performed (TK Engineering, 2013a). CAN peripheral drivers were tested with CAN message sending and receiving test and more testing was done when the CANopen implementation was tested.

CiA301, CiA305, CiA401 and the EDS files conformance were tested with the CANopen conformance test tool (TK Engineering 2013a, 2013b). CANopen conformance test tool is the official tool which is used for testing the conformance for CANopen certification (CiA, 2013a). There are different levels of CANopen conformance defined by three different test types: lower-, dynamic -and upper test (CiA, 2009).

Lower test consists of two different parts: EDS test and device test. On the EDS test correctness, completeness and consistency of the EDS file is checked. The EDS test is divided to two different parts: reading values of the EDS file and checking if those values are valid (CiA, 2011).

The device test consist of protocol verification, object dictionary, emergency, error control protocols, synchronization producer, other object dictionary entries, CANopen network management (NMT) states and NMT state transitions tests (CiA, 2009).

Passing of the lower tests is needed for CiA device certification (CiA, 2013a). The dynamic and the upper tests are additional (CiA, 2009).

SYSTEM PERFORMANCE

Most of the problems found on CANopen conformance test are in the CanFestival CANopen stack. There were five errors in Service Data Objects (SDO), five in Process Data Objects (PDO), five in object dictionary, one in state test and two in error control test errors as described in the first full report provided by TK Engineering. On the second report there were 10 errors in SDO, five in PDO, two in object dictionary, one in emergency and two in error control. (TK

Engineering, 2013a) This increasing error amount was caused by fixing low limit error in SDO Test 11. It was writing on object 1800h/00h which is COB-ID of the Transmit PDO Communication Parameter 0.

Problems were fixed by doing as little modifications as possible. This enables to easily add new features to UWASA Node CanFestival implementation, if such features would be added to CanFestival stack by the stack developers. One must also notice that these tests were made when there were no other tasks running on UWASA Node's operating system, which is the best case when testing response times and other time related activities.

SDO test 10 CANopen conformance test tool tried to write value 253 to entry 1A00h/00h when the high limit in EDS file was 8. The device should return an error code instead of accepting the value. (TK Engineering, 2013a) This was fixed by adding a check for high limits to object dictionary based on EDS file. We also added a property to send an error code 0609 0031h when the high limit is exceeded.

In SDO test 11, the UWASA Node accepted a lower value than what was defined in the EDS file. Test tool was trying to write value 0 to entry 1801h/0h, devices communication object identifier (COB-ID), when low limit was 1. The device accepted this value when it was supposed to discard it and return the error code 0504 0002h Invalid block size or 0800 0000h General error (TK Engineering, 2013a). This test was causing a failure in PDO Test 2, which was reading the value of entry 1800h/00h to EDS file value. Because the value accepted in SDO Test 2 was lower than the one accepted in PDO Test 2, the value was not the same as in EDS file.

PDO Test (3)

In the PDO Test 3, the test tool changed the state of the device from pre-operational to operational. After changing the state, the device should send its transmit PDO values to CAN bus. When UWASA Node state was changed to operational, it was receiving transmit PDO 0 value but not transmitting PDO 1 value. (TK Engineering, 2013a) This was caused by the CAN driver and CanFestival co-operation problem. The problem was fixed by enabling the transmit buffers of LPC2378 microcontroller.

PDO Tests (8, 9 and 16)

There are three different ways of mapping PDOs: static, variable and dynamic. On static mapping variables that are going to be send on PDO, are predefined and they cannot be changed via CANopen interface. Variable PDO mapping can be changed in NMT pre-operational state. Dynamic PDO mapping differs from variable PDO mapping by providing a possibility to change the mapping in NMT operational state (CiA, 2013f).

PDO Tests 8, 9 and 16 are used to test the correctness of the PDO mapping. Wrong mapping procedure was tested in PDO Test 8. An attempt to map too many objects to any supported PDO was tested in PDO Test 9. An attempt to map an existing object that cannot be mapped to any supported transmit PDO was

tested in PDO Test 16. The performance of UWASA Node CanFestival implementation failed in these tests (TK Engineering, 2013a).

The fact that CanFestival does not support dynamic or variable PDO mapping was causing errors on SDO tests. This was fixed by adding more PDO entries to object dictionary and checks for handling enabling and disabling of PDO entries.

PDO Tests (25 and 26)

PDO Test 25 was trying SDO block transfer with a wrong block size. This should generate an abort message 0504 0002h Invalid block size or 0800 0000h General error. (TK Engineering, 2013a) The problem was fixed by adding a block size check to CanFestival codes and the property that if the block size check fails, then the device should send the error code 0504 0002h.

Other CANopen Tests (1, 3 and 4)

Other CANopen tests (1, 3 and 4) were used to test device's capability to store and restore parameters to their default values. (TK Engineering, 2013a) Errors existed because these parts were not implemented to UWASA Node when those tests were carried out.

On CANopen device there are parameters which the user can change on runtime. CANopen provides a way of storing these parameters and restoring parameter default values. CANopen specifications do not take into account where the data should be stored. The specification tells only that the data should be stored to non-volatile memory (CiA, 2011). In UWASA Node that non-volatile memory is only in the flash memory of the microcontrollers.

On LPC2378, there are specific in-application programming (IAP) functions which are used for writing data from microcontrollers RAM to microcontrollers flash memory. These functions copy certain size of blocks from RAM to flash. LPC2378 flash memory is divided to 27 different sectors and each of them has different size parts. This part size limits the size of data that can be copied from RAM to flash.

Each of CanFestival object dictionary entries have their own structure which is stored to microcontrollers RAM. Because different entries are located in different structures, the object dictionary entries are not stored to one continuous location in RAM. This made it impossible to directly use IAP functions for copying data from RAM to flash memory. When store parameters is triggered, each parameter which should be stored is copied to one array and after copying the data that array is transferred from RAM to flash memory using IAP functions. This happens when signature save is written to entry 1010h (CiA, 2011).

When signature load is written to entry 1011h, the device should restore its default parameters next time it is powered up (CiA, 2011). The first byte of the parameter array, which is used to check whether the device restore its parameters or not, is written to one. Once the device powers up, it checks whether the first byte is one. If it is, it restores the default values and writes them also to flash memory.

Error Control Tests (7 and 10)

Error Control Tests 7 and 10 tests are used to test the sending of heartbeat messages (TK Engineering, 2013a). Heartbeat messaging is a mechanism to check if a certain node is alive or not. Heartbeat messages can be sent on pre-defined sequence. The transmitting device is called a heartbeat producer. A device that listens (receives) the heartbeat messages transmitted by other devices is called heartbeat consumer. On CiA 301 specification the meaning of the heartbeat messages is described as follows "If the heartbeat cycle fails for the heartbeat producer the local application on the heartbeat consumer will be informed about that event" (CiA, 2013d). This is to get information about the network and network devices.

Error Control Test 7 was testing the accuracy of the produced heartbeat messages. In this test the heartbeat messages were received either in random interval or not at all (TK Engineering, 2013a). These problems were related to the timer settings of the microcontroller. CanFestival was not able to set the timer the way it wanted and that caused the error in this test. After the problem was fixed the deviation of the heartbeat messages time interval was small enough to be feasible. It was less than 1 ms between each message.

CONCLUSIONS AND FUTURE WORK

This paper describes a CANopen implementation to wireless sensor network. Such an implementation enables us to build wireless extensions into CAN based control system so that we can reach such parts of the machine, which are not reachable by cables. We used an open source CANopen stack CanFestival and a wireless sensor platform the UWASA Node in our implementation.

The implementation was tested by using CANopen conformance test tool, which is an official tool for CANopen testing. Most of the problems and shortages, which were found in the testing, were rising from the CanFestival CANopen protocol stack, not from mistakes or other issues in the implementation itself. Based on the test results certain basic tasks can be performed by using the current implementation. On the other hand, it was also indicated that the CanFestival stack is more suitable for smaller and less complex devices, for example CANopen IO-boards. For more complex devices it would be better to use other commercial- or open source CANopen protocol stacks.

In the nearby future we will test our CANopen implementation further by some pilot implementations with real machines. We will also implement some commercial CANopen stack and compare its performance with CanFestival.

REFERENCES

Björkbom, M., Timonen, J., Yigitler, H., Kaltiokallio, O., Vallet, J., Myrsky, M., Saarinen, J., Korkalainen, M., Cuhac, C., Koivo, H., Jäntti, R., Virrankoski, R. and Vankka, J. 2013. Localization Services for Online Common Operational Picture and Situation Awareness, IEEE Access, Vol. 1, no. 1, pp.742-757.

CanFestival 2013a, CanFestival website, http://www.canfestival.org, cited 2013.

CanFestival 2013b, The CanFestival CANopen stack manual, http://dev.automforge.net/CanFestival-3/raw-file/tip/objdictgen/doc/manual_en.pdf, cited 2013.

CANopenNode 2013, CANopenNode website, http://sourceforge.net/projects/canopennode/, cited 2013.

CiA 2009, CiA 310 Draft Standard Proposal, Conformance test plan.

CiA 2011, CiA301, CANopen application layer and communication profile.

CiA 2013a, CANopen Conformance test tool, CiA website, http://www.can-cia.org/index.php?id=conformance, cited 2013.

CiA 2013b, CANopen internal device structure, CiA website, http://www.can-cia.org/index.php?id=506, cited 2013.

CiA 2013c, About CAN in Automation (CiA), CiA website, http://www.can-cia.org/index.php?id=aboutcia, cited 2013.

CiA 2013e, CANopen introduction, CiA website, http://www.can-cia.org/index.php?id=17, cited 2013.

CiA 2013f, Process data object (PDO), CiA website, http://www.can-cia.org/index.php?id=153, cited 2013.

Hensler, T. 2012. CANopen and J1939 sensors for agricultural technology. CAN newsletter 4/2012.

TK Engineering 2013a. UWASA Node CiA-301 Test Report. TK Engineering 2013.

TK Engineering 2013 b. UWASA Node CiA-401 Test Report. TK Engineering 2013.

Virrankoski, R. (Ed.), Generic Sensor Network Architecture for Wireless Automation (GENSEN), Proceedings of the University of Vaasa, Reports 174, Vaasa 2012.

Virrankoski, R. (Ed.). 2013. Wireless Sensor Systems in Indoor Situation Modeling II (WISM II). Proceedings of the University of Vaasa, Reports 188, Vaasa 2013.

Yigitler, H., Virrankoski, R. and Elmusrati, M. S. 2010. Stackable Wirless Sensor and Actuator Network Platform for Wireless Automation: the UWASA Node. Aalto University Workshop on Wireless Sensor Systems, November 19th, 2010, Espoo, Finland.