Damian MAZUR [1], Piotr MACIĄG [2], Paweł KIELAN [3]
[1] RZESZOW UNIVERSITY OF TECHNOLOGY, DEPARTMENT OF ELECTRICAL AND COMPUTER FUNDAMENTALS, 2 Wincentego Pola St., 35-959 Rzeszów, Poland
[2] WARSAW UNIVERSITY OF TECHNOLOGY, INSTITUTE OF COMPUTER SCIENCE, 15/19 Nowowiejska St., 00-665 Warszawa, Poland
[3] SILESIAN UNIVERSITY OF TECHNOLOGY, FACULTY OF ELECTRICAL ENGINEERING, 2 Krzywoustego St., 44-100 Gliwice, Poland

Building a wireless sensor network with use of RFM 12 modules - data exchange between nodes

Abstract

IEEE 802.15.4 standard defines Physical (PHY) and Wireless Medium Access Control (MAC) layers for wireless sensor networks (WSN). In recent years, many types of wireless sensor networks have been developed based on this standard. Currently, probably the most popular implementation of the mentioned standard are wireless ZigBee networks. Moreover, ZigBee provides functions normally attributed to the higher layers of network protocol stack. Among those the most important functions are: routing of data, data encryption and providing redundancy connections between devices in the network. We present a slightly different approach to the construction of wireless sensor networks. This is the beginning of series of papers about implementation of the IEEE 802.15.4 standard and the previously mentioned expansion functions in the network, built with RFM wireless communication modules and ATmega processors. This paper describes the basics of compiling wireless connection and data transmission with using RFM modules. In addition, blocking and non-blocking types of connections between radio modules and ways to connect RFM modules with ATmega processors are described.

Keywords: IEEE 802.15.4 standard, wireless sensor networks, RFM modules, ATmega processors.

1. Introduction

Sensor systems have many applications. Fig. 1. presents the architecture of a sensor system. Devices in the system can operate in both hardware and software layer. In the hardware layer, the devices collect the measurement data and send the data to the gateway. The collected data may be related to current temperature, humidity, weather status or actually monitored patient health. Typically, sensors communicate directly with the network gateway, which transmits the received data to the server. Frequently, communication is one-way, that is, from sensors to gateway, but this is not the rule. The gateway, e.g., can transmit user commands from the server to the sensors. The software layer consists of a program installed on the server, which is responsible for receiving data from the network gateway, recording them in database and visualizing the status of the whole system [8].

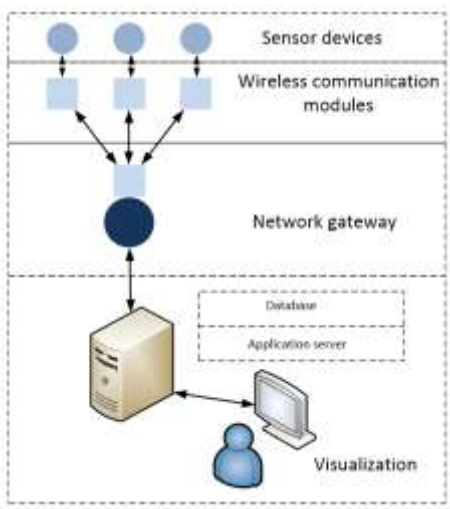

Fig. 1. Architecture of a wireless sensor network [8]

A wireless sensor network is built on the basis of the protocol stack. In sensor networks, we can distinguish the following layers of the protocol stack: physical, data link (media access control), network and application. Fig. 2. Shows the protocol stack used in a ZigBee network.

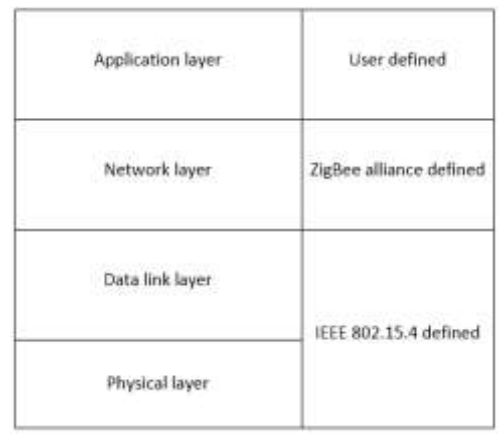

Fig. 2. ZigBee protocol stack [3]

The application layer is generally defined by the API, by which the user can control radio communication modules. Data encryption is an example. The network layer is responsible for transmitting data in the network. The data link layer manages the order of access to media and the resolving of transmission conflicts. In the physical layer, we have a function for selecting an appropriate radio frequency and modulation rate. For RFM modules, only commands to configure the physical layer have been provided by the manufacturer. Higher layer functions should be defined by a programmer [3].

2. Connection between RFM modules and ATmega processor. Sensor design

Each sensor consists of three parts: a microprocessor, a radio communication module and other peripherals, like: thermometers, optical sensors, relays and many others. The microcontroller ATmega8 is a processor based on the RISC architecture, with a 2-cycle multiplier. Most instructions are executed within a single clock cycle. The processor is equipped with RAM, FLASH and EEPROM memories. FLASH memory stores the user application and defined constants, while EEPROM is available to the programmer and can store a variable state even after power is turned off. RAM stores variables currently used by the program. Fig. 3. shows the ATmega8 microprocessor in the PDIP housing [2].

Several peripherals devices are built in the microprocessor: an analog to digital converter, an interrupts system, two 8-bit and one 16-bit Timers/Counters, thirty-two 8-bit general purpose registers. The ATmega 8 CPU clock speed is up to 16 MHz, but in the sensors described in this paper the clock speed is default – 2 MHz.

RFM radio modules can act as a transmitter or a receiver. There are versions that operate at 433, 868 and 915 MHz. The bit rate may reach 115.2 kbps in the digital mode and 256 kbps in the analog mode. These modules can operate with 3 V or 5 V supply voltage. The microprocessor is connected to the RFM radio

module via an SPI interface, but on the processor side the SPI interface is emulated in software. Fig. 4. shows the RFM12 module, while Table 1. Presents description of the pins [4].

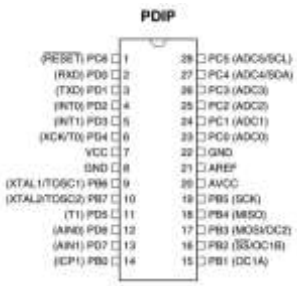

Fig. 3. ATmega8 in the PDIP housing [2]

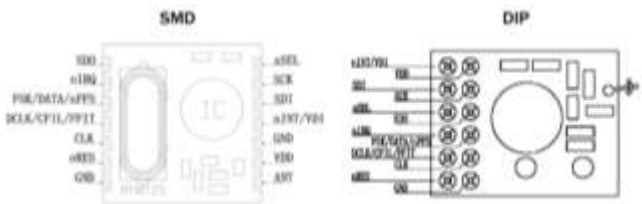

Fig. 4. RFM12 module in the housing [4]

Tab. 1. Description of the RFM12 pins (D=digital, A=analog, S=supply, I=input, O=output, IO=input/output) [4]

Definition	Type	Function
nINT/VDI	DI/DO	Interrupt input (active low)/Valid data indicator
VDD	S	Positive power supply
SDI	DI	SPI data input
SCK	DI	SPI clock input
nSEL	DI	Chip select (active low)
SDO	DO	Serial data output with bus hold
nIRQ	DO	Interrupts request output (active low)
FSK/DATA/ nFFS	DI/DO/ DI	Transmit FSK data input/Received data output (FIFO not used)/ FIFO select
DCLK/CFIL/ FFIT	DO/AIO/ DO	Clock output/ external filter capacitor/FIFO interrupts
CLK	DO	Clock output for external microcontroller
nRES	DIO	Reset output (active low)
GND	S	Power ground

As already mentioned earlier, the manufacturer of RFM modules provides commands only for the physical layer configuration. All data exchanged between the microcontroller and the module is transmitted serially. The bits placed on the SDI line are passed to the module at the time of rising edge on the SCK line, at the same time the nSEL pin should be in a low state. The state high at the nSEL pin initializes the SPI interface. Commands consists of two parts: a number of the command and parameters, eventually the data. The data is transmitted from the most significant bit. Fig. 5. shows the connection scheme between the radio module and the microprocessor [9].

Not all the outputs of the module were used in this project of a sensor node. These which were used had the following functions: VDD and GND – power input, nSEL – all data is transferred to the module at the low state of this output, SDI – data input, SDO – data output, SCK – clock input, nIRQ – output which can generate interrupt in the case of a specific event. The module can be put into the sleep mode, which significantly reduces the energy consumption. In the case of a predefined event, the module wakes up and generates a signal on the nIRQ line. The signal on the nIRQ line can be generated, among others, by the following events: RGIT – TX register is empty, occurs only in the transmitter mode, FIFT – a number of bits in the receiver FIFO buffer reach a certain level (generated only in the receiver mode). POR – power-on reset interrupt – interrupt generated when the supply voltage level is changed or the RESET command is issued, FFOV – FIFO buffer overflow, works only in the receiver mode, LBD - interrupt is generated when the supply level falls below a certain threshold.

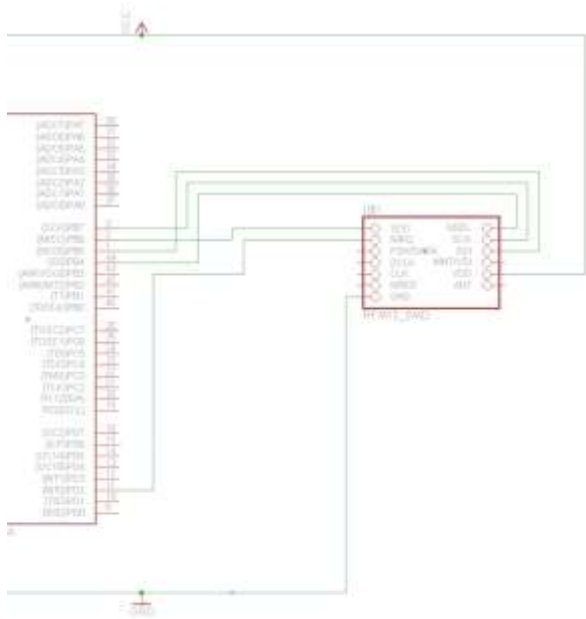

Fig. 5. Connections between the ATmega and the RFM12 module [1]

3. Data exchange between two sensors. Initialization commands and operation modes

The transmission and reception of data is carried out according to the strict rules. Let us start with sending the data. The transmitter operates according to the scheme shown in Fig. 6.

Fig. 6. Sending data and transmitter initialization [7]

The first step is to set appropriate states at the outputs of the microcontroller, connected to the radio module. The procedure is shown on Listing 1.

```
#define SCK 7
#define SDO 5
#define SDI 6
#define CS 4
#define NIRQ 2

#define HI(x) PORTB |= (1<<(x))
#define LO(x) PORTB &= ~(1<<(x))
#define WAIT_IF_NIRQ_LOW() while (PIND&(1<<NIRQ))

#define LED 6
#define LED_OFF() PORTD &= ~(1<<LED)
#define LED_ON() PORTD |= (1<<LED)

void portInit()
{
    HI(CS);
    HI(SDI);
    LO(SCK);
    DDRB = (1<<CS) | (1<<SDI) | (1<<SCK);
    DDRD = (1<<LED)
}
```

Listing 1. Initialization of the microprocessor ports connected to the radio module [6]

Initialization of the input and output ports corresponds to the procedure `portInit()`, which first sets the high state on the pins CS (chip selection) and SDI (data output) and the low state on the clock line (SCK). Then, all these lines are set as the outputs. In addition, a LED diode that indicates the beginning and the end of the transmission, is connected to pin 6 of the port D . Macro `WAIT_IF_NIRQ_LOW()` is responsible for detection if the line NIRQ transits to the low state. This means that the receiver detects the data. After the initialization procedure, the FIFO queue should be reset.

```
void FIFOReset()
{
    writeCMD(0xCA81);
    writeCMD(0xCA83);
}
```

Listing 2. Procedure of FIFO queue reset [6]

The procedure sends commands to the radio module to set the FIFO queue and reset the FIFO. All the commands sent by the microcontroller to the radio module consists of two bytes. For most commands, the first byte is the number of the command and the second byte consists of the command parameters. The above listing shows the command with the hexadecimal number `0xCA`. The exact description of the commands can be found in the datasheets of the RMF12 module.

The next step after resetting the FIFO is to initialize the state of the transmitter. Initialization is done by sending a series of commands to the chip. The most important of them are: setting the overall frequency of the transmitter (command number `0x80`), setting the module to the transmitter mode (command number `0x82`), definition of the transmission channel (command `0xA6`) and the choice of the bit rate (command `0xC6`). Listing 3. shows the transmitter initialization procedure. In the main loop of the program, the data is sent periodically by the radio module, and other tasks of the microcontroller are executed. The beginning of the data transmission is indicated by the LED diode. First, three bytes of preamble are sent, and the value of each byte is 0xAA. After them two synchronization bytes are sent. The values of these bytes can be changed by using appropriate commands. The default value of the first synchronization byte is 0x2D and the second is 0xD4. Next, the user data can be sent. The end of the transmission is signalled by three bytes of the footer. The value of each byte is 0xAA [5].

The initialization procedure of the receiver is similar to that of the transmitter. The initialization includes a series of commands issued to the radio module by the controller. The initialization procedure is shown in Listing 3 [6].

```
void rfInitRec() {
    writeCmd(0xA640); //select Channel
    writeCmd(0xC647); //set transmission speed (4.8kbps)
    writeCmd(0xCED4); //Sent preamble and Synchro bytes (2DD4)
    FIFOReset();
}
```

Listing 3. Initialization procedure of the receiver [6]

Fig. 7. shows the initialization process in the state of the receiver and data receive process.

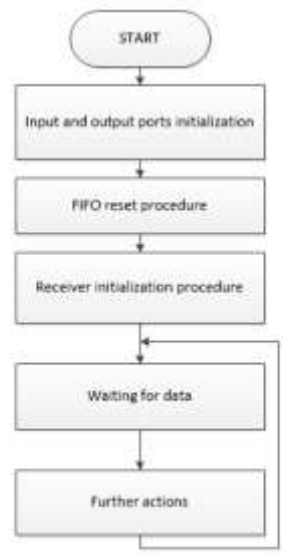

Fig. 7. Process of initializing modules in the receiver state [7]

In the state of the receiver, the module can operate in either blocking or non-blocking state. In the blocking mode, the receiver waits for the data over an unlimited period of time. In the non-blocking mode, the waiting for data is limited to a specific segment of time. The counter which specifies how long the controller should wait for the data is decremented in the interrupt handling procedure. Listing 4. shows the procedure of initializing the microcontroller timer. Listing 5. shows the counter decrementing procedure (interrupt handling procedure).

```
void timInit( )
{
    TCCR1B |= (1 << CS12);
    TCCR1B |= (1 << WGM12);
    OCR1A=1000;
    SREG=(1 << 7);
    TIMSK=(1 << OCIE1A);
}
```

Listing 4. Timer 1 initialization procedure [6]

In the first line of the Timer 1 initialization procedure, the value of the prescaler is set to 256, which means that the timer counting will be at $clk_{i/o} / 256$.

Line 4 in Listing 4. sets the timer in the CTC mode. This means that count will be followed from 0 to the value entered in the register OCR1A. The bit OCIE1A in the TIMSK register is set. This means permission to interrupt when the Timer1 counts to the value entered in the OCR1A. Setting the seventh bit in the SREG register causes global permission for interrupts.

```
1  ISR(TIMER1_COMPA_vect)
2  {
3      if(tim)
4          --tim;
5  }
```

Listing 5. Timer 1 interrupt handling procedure [6]

Listing 5. shows the interrupt service routine of the Timer 1. In this procedure, the variable `tim` is decremented. At the time when the value of `tim` is equal to zero, the waiting time for the data is interrupted. The `Tim` variable is always set at the entrance to the state of waiting for the data. The value of this variable should be adjusted to the sending data frequency.

4. Transmission range for modules RFM12 – 433 MHz

A wireless transmission node in addition to the wireless module, has several other components. The most important are: an LCD display, a temperature sensor DS18B20 and buttons to simulate gas and carbon monoxide. In addition, the node is equipped with a LED diode, which indicates the data transmission. Fig. 8. shows the wireless node.

Fig. 8. The designed wireless node [2]

The gateway, as a measuring node, is equipped with the RFM module. In addition, the gateway connects to the server via the USB port. Figure 9. shows the designed gateway.

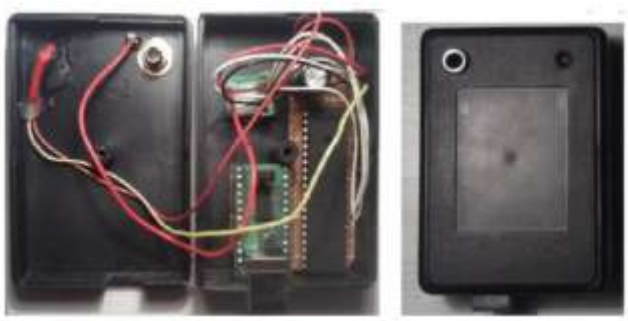

Fig. 9. The designed gateway [2]

The most interesting parameter from the network designer point of view is the transmission range. Fig. 10. shows the dependence of the number of discarded packets as a function of the distance for RFM modules. It was measured for both open and closed spaces.

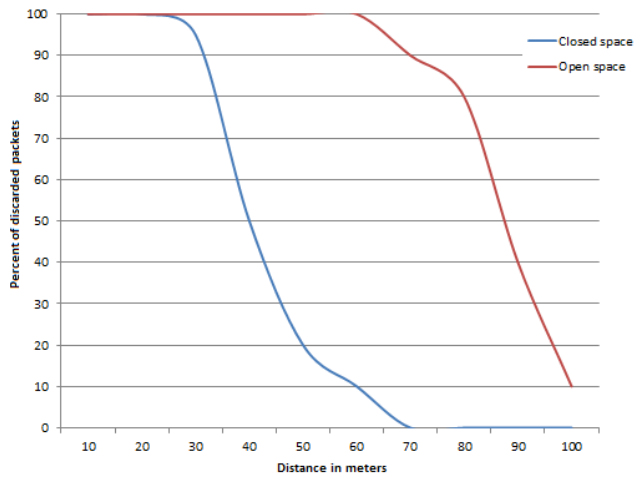

Fig. 10. The number of discarded packets as a function of the distance [2, 4]

5. Conclusions

Sensor networks are a rapidly growing branch of technology. The most important problems facing the designer of the network are: the development of protocols for data transmission between nodes, the development of access to the media and addressing devices on the network, as well as the design of data acquisition and storage of data on the server.

The network security and data encryption are other important issues.

Sensor networks are used in the design of intelligent buildings, industrial automation systems and prompting threats.

6. References

[1] Abbasi Q.H, Rehman M.U, Liaqat S., Alomainy A.: Multiple input multiple output radio channel characterisation for ultra wideband body centric wireless communication. RF and Microwave Conference (RFM), IEEE International 2013.

[2] Datasheet for ATmega8 - http://www.atmel.com/images/atmel-2486-8-bit-avr-microcontroller-atmega8_l_datasheet.pdf.

[3] Dymora P., Mazurek M., Maciąg P.: System monitorujący pracę bezprzewodowej sieci sensorowej oparty na przestrzennych typach danych. PAK 2014 nr 10, p. 849-853, 2014

[4] Dymora P., Mazurek M., Nieroda S.: Sensor network infrastructure for intelligent building monitoring and management system. Annales UMCS Informatica , Vol.12 (2), 2012, pp. 59-71.

[5] Chlumsky P., Vodrazka J.: Innovative Two-path Data Transmission Scheme Proposal. Elektronika Ir Elektrotechnika, Vol 20, No 10, 2014.

[6] Lim J.C., Wong K.D.: More Power-Efficient Data Gathering Trees for Wireless Sensor Networks. RF and Microwave Conference (RFM), IEEE International, 2006.

[7] Logan S., Venugopal M.: Analysis of wireless system design for aircraft indoor wireless sensor communication. RF and Microwave Conference (RFM), IEEE International, 2013.

[8] Faludi R.: Building Wireless Sensor Networks. Published by O'Reilly Media, Inc., 2011.

[9] Martusevičius V., Kazanavičius E.: Self-localization System for Wireless Sensor Network. Elektronika Ir Elektrotechnika, Vol 103, No 7, 2010.

[10] Marsic I.: Wireless Networks - Local and Ad Hoc Networks. Department of Electrical and Computer Engineering and the CAIP Center, Rutgers University.
[11] Severdaks A., Supols G., Greitans M., Selavo L.: Wireless Sensor Network for Distributed Measurement of Electrical Field", Vol 107, No 1, Elektronika Ir Elektrotechnika, 2011.

Received: 16.06.2015 *Paper reviewed* *Accepted: 03.08.2015*

Damian MAZUR, DSc, PhD

Damian Mazur works at The Faculty of Electrical and Computer Engineering in Rzeszow University of Technology as a lecturer. At his didactic work he is dealing with diagnostics of the electromechanical devices (calculations and measurements of the electric machines), numerical methods (finite element method, boundary element method), object programming and databases.

e-mail: mazur@prz.edu.pl

Piotr S. MACIĄG, MSc

Piotr Maciąg works at the Institute of Computer Science, Warsaw University of Technology. He is currently PhD Student at Warsaw University of Technology. He graduated from Rzeszow University of Technology as Master of Science.

e-mail: piotr.maciag32@gmail.com

Paweł KIELAN, PhD

Paweł Kielan works at Faculty of Electrical Engineering at the Silesian University of Technology. His area of research interest covers mechatronics, mechatronic control systems via the Internet, smart grids, rapid prototyping, robotics.

e-mail: Pawel.Kielan@polsl.pl