

An Intruder Alarm System

Bc. Václav Mach

Master's thesis
2016



Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2015/2016

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Václav Mach**
Osobní číslo: **A14434**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Bezpečnostní technologie, systémy a management**
Forma studia: **prezenční**

Téma práce: **Integrovaný poplachový systém**

Téma anglicky: **An Integrated Alarm System**

Zásady pro vypracování:

1. Prostudujte a popište požadavky na integrovaný poplachový systém a jeho technické součásti.
2. Navrhněte vlastní poplachový systém s běžně požadovanými funkcemi.
3. Realizujte navržený systém s využitím zvolených mikropočítačů.
4. Vytvořte programové vybavení pro mikropočítače použité v jednotlivých součástech systému včetně jejich komunikace.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **CATSULIS, John.** Designing embedded hardware. 2nd ed. Sebastopol, CA: O'Reilly, 2005, xvi, 377 p. ISBN 0596007558.
2. **MARGOLIS, Michael.** Arduino cookbook. 2nd ed. Sebastopol, Calif.: O'Reilly, 2012, xx, 699 p. ISBN 1449313876.
3. **MASSIMO BANZI.** Getting started with Arduino. 2nd ed. Farnham: O'Reilly, 2011. ISBN 9781449309879.
4. **MAZIDI, Muhammad Ali, Sarmad NAIMI and Sepehr NAIMI.** The AVR microcontroller and embedded systems: using Assembly and C. Upper Saddle River, N.J.: Prentice Hall, 2011, xiv, 776 p. ISBN 01-380-0331-9.
5. **PINKER, Jiří.** Mikroprocesory a mikropočítače. 1. vyd. Praha: BEN – technická literatura, 2004, 159 s. ISBN 80-7300-110-1.
6. **ARDUINO CORPORATION.** Arduino [online]. 2015 [cit. 2015-04-22]. Available online from: <http://www.arduino.cc/>

Vedoucí diplomové práce:

Ing. Jan Dolinay, Ph.D.

Ústav automatizace a řídicí techniky

Datum zadání diplomové práce:

5. února 2016

Termín odevzdání diplomové práce:

16. května 2016

Ve Zlíně dne 5. února 2016



doc. Mgr. Milan Adámek, Ph.D.
děkan



doc. RNDr. Vojtěch Kresálek, CSc.
ředitel ústavu

I hereby declare that:

- I understand that by submitting my Diploma thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Diploma Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Diploma/Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlin, and that a copy shall be deposited with my Supervisor.
- I am aware of the fact that my Diploma Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, TBU in Zlin has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work - Diploma Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlin with a view to the fact that Tomas Bata University in Zlin must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Diploma Thesis include the use of software provided by Tomas Bata University in Zlin or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Diploma Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Diploma Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- That the submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlin; dated: 8.5.2016


.....
Student's Signature

ABSTRAKT

Cílem této práce je vytvořit volně dostupný open-source Integrovaný Poplachový Systém, který je sestaven ze stejných součástí, ze kterých se skládají profesionální ústředny Poplachové Zabezpečovací a Tísňové Systémy (PZTS). Výsledné zařízení je schopné komunikovat se všemi běžnými poplachovými detektory. Jednotlivé moduly jsou na sobě nezávislé díky modulárnímu systému. Uživatelské prostředí je realizováno pomocí ethernetového serveru, ale je také přístupné pomocí klávesnice.

Klíčová slova:

Integrovaný Poplachový Systém, Atmel, Arduino, Open source

ABSTRACT

The aim of this thesis is to create an open source Integrated Alarm System. This system is composed of identical components as a real system which is done by commercial systems. Final device is able to communicate with all common alarm sensors. The system is modular, with the individual components independent of each other. Users interface is realized mainly by the Ethernet server, but it is also accessible using the keypad.

Keywords:

Integrated Alarm System, Atmel, Arduino, Open source

ACKNOWLEDGEMENTS

I would like to thank to my supervisor Ing. Jan Dolinay, Ph.D. for his valuable advice. Also i would like to thank to Printed s.r.o. for model creation.

I hereby declare that the print version of my Bachelor's/Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

CONTENTS

INTRODUCTION	9
I. THEORY.....	10
1 INTRUDER ALARM SYSTEM.....	11
1.1 CONTROL AND INDICATING EQUIPMENT.....	11
1.1.1 BUS CONNECTION	12
1.1.2 LOOP CONNECTION	13
1.2 ALARM TRANSMISSION SYSTEM	18
1.2.1 TELEPHONE COMMUNICATOR	18
1.2.2 CONNECTIONS USING ETHERNET	18
1.2.3 PROTOCOLS FOR CONNECTION TO ALARM RECEIVING CENTRE	19
2 INTEGRATED ALARM SYSTEMS.....	20
2.1 CONFIGURATION OF INTEGRATED ALARM SYSTEMS	21
2.1.1 TYPE 1.....	21
2.1.2 TYPE 2.....	22
2.2 SOFTWARE INTEGRATION	23
2.3 HARDWARE INTEGRATION	24
3 LEGISLATION AND STANDARDS.....	25
3.1 SYSTEM REQUIREMENTS	26
3.1.1 DEGREE OF SECURITY OF THE PROTECTED OBJECT.....	26
3.1.2 CLASSIFICATION OF ENVIRONMENT	26
3.1.3 CLASSIFICATION OF FAILURES	27
3.2 CONTROL AND INDICATING EQUIPMENT.....	27
3.2.1 DETECTION OF INVALID ATTEMPTS TO AUTHORIZATION	27
3.2.2 METHOD OF CONNECTION MONITORING.....	28
3.2.3 EVENT LOG AND MEMORY CAPACITY.....	28
3.3 POWER SUPPLY UNITS	29
4 ATMEL MICROCONTROLLERS	30
4.1 ATMEL AVR 8-BITAND 32-BIT.....	30
4.2 SMART ARM 32-BIT	31
4.3 MICROCONTROLLER ATMEL MEGA 2560.....	32
4.4 INTERRUPTS.....	33
4.4.1 INTERRUPT SERVICE ROUTINE	34
4.5 TIMERS	35
4.5.1 TIMER GENERATES ROLLOVER INTERRUPT	35
4.5.2 TIMER GENERATES INTERRUPT IN COMPARE MODE.....	36
4.6 SERIAL COMMUNICATION	36
4.6.1 UNIVERSAL ASYNCHRONOUS RECEIVER AND TRANSMITTER	37
4.6.2 RS-232.....	37
4.6.3 SERIAL PERIPHERAL INTERFACE	38
II. ANALYSIS	39

5 REQUIREMENTS FOR THE SYSTEM	40
5.1 CONTROL AND INDICATING EQUIPMENT	40
5.2 KEYPAD.....	41
6 CONTROL AND INDICATING EQUIPMENT	42
6.1 UNINTERRUPTIBLE POWER SUPPLY	43
6.2 MAIN BOARD WITH ETHERNET INTERFACE	44
6.2.1 MAIN MICROCONTROLLER.....	45
6.2.2 FT232RL UNIVERSAL SERIAL BUS INTERFACE	46
6.2.3 ETHERNET MICROCONTROLLER WITH W5100.....	47
6.2.4 UNIVERSAL SERIAL BUS HUB	50
6.2.5 POWER MANAGEMENT	51
6.3 GLOBAL SYSTEM FOR MOBILE COMMUNICATION MODULE.....	52
6.4 ZONE BOARD	54
6.4.1 END OF LINE ZONE WIRING.....	56
6.4.2 ADVANCED TECHNOLOGY ZONE WIRING	57
6.5 OUTPUT BOARD	59
7 KEYPAD.....	60
7.1 LIQUID CRYSTAL DISPLAY	61
8 SOFTWARE.....	63
8.1 MAIN PROGRAM.....	63
8.1.1 COMMUNICATION.....	64
8.1.2 GSM PROGRAM	66
8.2 ETHERNET PROGRAM.....	67
8.2.1 CHECK LOGIN	70
8.2.2 ADD USER.....	72
8.2.3 SET ZONE.....	73
8.3 ZONE PROGRAM.....	75
CONCLUSION	77
BIBLIOGRAPHY	78
LIST OF ABBREVIATIONS	80
LIST OF FIGURES	81
LIST OF TABLES	84
APPENDICES.....	85

INTRODUCTION

Every Control and Indicating Equipment (CIE) comes to the market with proprietary hardware and software and own customization is limited. The main goal of this thesis is to design CIE which can be fully customized by the user. The created device uses the same equipment which are commonly used in commercially available CIEs. These components are the Main board with Zone facilities, external and internal communication and also power management. All the designs of individual devices are made available under an open source licence.

The final system can be used as teaching equipment for students of security technology. Students can apply their knowledge to create a fully working system. Real detectors or other devices can be connected to the system, just like to other commercially available CIEs. The system needs software, which can be created by the students. Communication to the Alarm Receiving Centre (ARC) can be simulated.

Every CIE must conform to very strict requirements. The theoretical part of this thesis describes these requirements and legislation which the designer must comply to. It also contains basic information about Integrated Alarm systems and Intruder Alarm systems. Moreover, it provides basic information about microcontrollers, which are the core components of these systems.

The analytical part explains the individual components which were created: the Main board, Zone board, Output board and Keypad board. Each of these components has its own program and can work independently. A sample program was written for each created component, which can be used as a foundation for a more complex program created by the programmer. The web administration interface for the end user and communication software between each components were completely finished.

I. THEORY

1 INTRUDER ALARM SYSTEM

Intruder Alarm Systems (IAS) does not prevent the entry of unauthorized persons into the protected place or building. These systems are there only for the early warning of the intrusion. It is mainly focused on life, health and property protection of its owner. IAS can be categorized as technical protection. It can be further divided into perimeter, casing, area and object protection.

In general, it is a combination of these main elements:

- Control and Indicating Equipment (CIE)
- Detectors
- Warning Device (WD)
- Alarm Transmission System (ATS)
- Power Supply (PS)
- Ancillary Control Equipment (ACE)

The elements relevant for this work will be described in more detail.

1.1 Control and Indicating Equipment

Control and Indicating Equipment is the most important element in the IAS. It collects data from all other devices, which are connected to it. CIE is usually controlled by a microcontroller, which is able to control the whole system. The Keypad is a very important device, which provides the possibility to manage the system simply by prepared commands. The program and user interface can vary widely depending on the manufacturer. [8]

Every CIE has basic functions like:

- receiving and evaluating signals from the detectors
- controlling connected devices
- provides power supply to the detectors
- switching between armed and un-armed mode

Every CIE must contain a power supply, which transforms the input high voltage to the low working voltage. There must also be a place for connecting the alarm loops or some internal connector for digital communication with detectors.

CIE can be divided into several parts, depending on the point of view. Following divisions is from the perspective of alarm loop connection:

- wired connection
- wireless connection
- hybrid connection

The most important connection for this thesis is the wired connection. The wireless connection is more complicated in terms of communication and technical equipment and is not covered in this thesis. Wired connection can be divided into following variants:

- bus connection
- loop connection
- mixed connection

1.1.1 Bus Connection

All used detectors in this case are connected to a common bus. The physical bus consists of several cables. These cables are usually distinguished by colors. Two cables are used for the power voltage and another two for the communication. In this schema, every connected detector must have its own communication module. Such module consists of a microcontroller which is able to communicate with the command microcontroller in the CIE. Every detector also needs its own unique address number.

CIE periodically checks the addresses of the individual detectors and receives an immediate response. When a detector stops responding, CIE generates the alarm signal. A unique address of each detector must be set first, then bus communication can be used. The service worker must manually assign the address to the corresponding detector on the bus.

The maximum number of connected detectors depends on the selected addressing scheme. Most commonly, the maximum number of devices is set to 128. This number comes from the seven bits used, where the maximum number of devices is equal to $2^7 = 128$.

1.1.2 Loop Connection

Alarm loops are the main element of every CIE. Depending on the version used, one or more detectors can be connected to each loop. There are four variants of alarm loops and each can be further modified:

- Normally closed (NC)
- End of line (EOL)
- Double end of line (DEOL)
- Advanced technology zone (ATZ)

State of each loop is determined by measuring the resistance of the wire itself. Adding more resistors to the circuit can affect the final resistance. These differences are the main information for the evaluating device. Real devices can not measure the input resistance. It is the voltage that is measured at the input of each loop.

Nowadays, almost every electrical device works digitally, but the signal from the loop is in analog form. That means that every input must have an Analog to Digital Converter (ADC) which can convert analog signals to digital form. Each mentioned variant has its own thresholds, which are described in the following chapters.

Every loop should be able to distinguish between basic states. These states for every variant are as follows:

- Serenity
- Alarm
- Failure
- Sabotage

Serenity and Alarm state are very easy to identify. The tricky part comes with the detection and subsequent distinction between Failure and Sabotage. In the NC variant it is almost impossible to distinguish them. In such case, Failure must be detected as Alarm.

The connected detectors can have some additional outputs like tamper or anti-masking. These connections must be wired in the correct way according to the device manual and also an appropriate version must be used.

Normally Closed

Normally Closed loop is one of the simplest versions. On the other hand it has many disadvantages. When more than one detector is connected to the loop it is impossible to distinguish the Alarm state between two or more detectors. Also Sabotage by short-circuiting has the same conditions as Serenity.

Multiple detectors must be connected serially. The maximum number of connected detectors depends on the manufacturer, but in general, the recommended number is up to five detectors. An example wiring scheme of the loop is shown in the following figure.

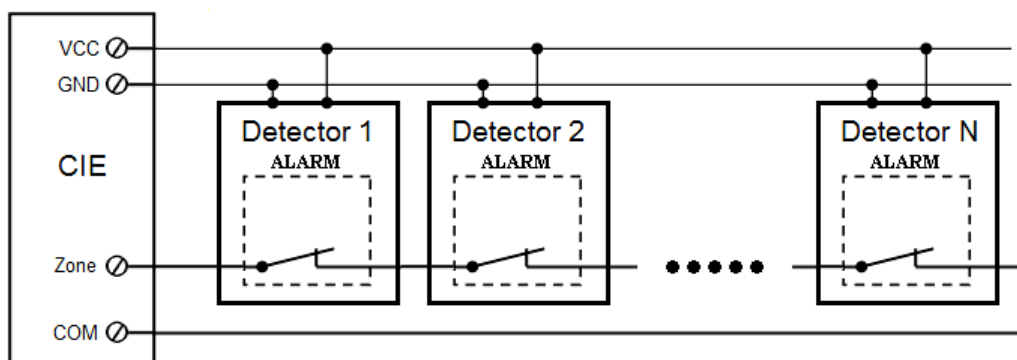


Figure 1: Normally closed loop[7]

The loop has only two states Serenity and Alarm. When the total resistance of the loop approaches zero, the state of the loop is evaluated as Serenity. In real operation, resistance is not precisely zero, but the resistance of wire itself is neglected. The Alarm state can represent multiple individual states. These states are Failure, Sabotage and Wire break. In these states, total resistance approaches infinity.

$R = 0\Omega$	$R = \text{inf. } \Omega$
Serenity	Alarm (Failure, Sabotage, Wire break)
$R = R_{\text{WIRE}}$	

Figure 2: Graphical states of the normally closed loop

Due to the disadvantages mentioned above, using this wiring in the IAS is strictly forbidden. Nevertheless, it's useful to describe it as all the following versions are based on NC wiring.

End of Line

This variant is based from the previously described NC wiring. The only change is an additional resistor, called end of line resistor. This small change is enough to add another state of the loop and distinguish the Sabotage between Short circuit and Wire break. However, this variant still cannot distinguish among individual detectors.

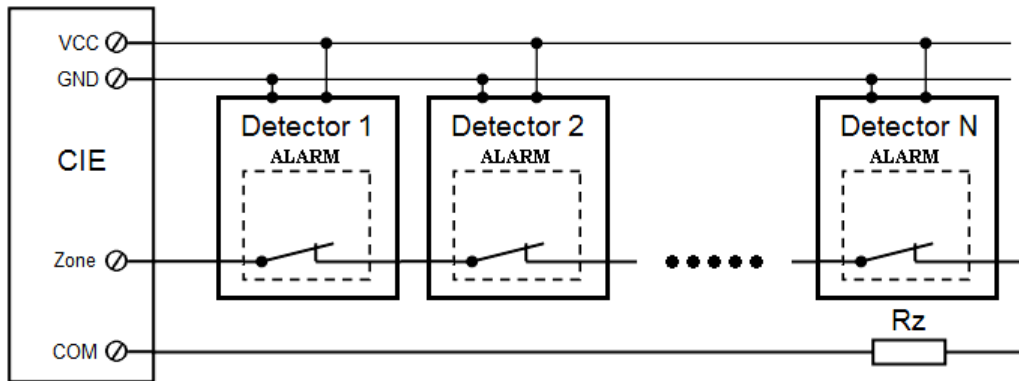


Figure 3: End of line loop [7]

In the Serenity state, total resistance is equal to R_z . The value of the used resistor is usually 2,2 k Ω , but the value can vary depending on the manufacturer. When resistance drops to zero, CIE evaluates this state as Sabotage by Short circuit. On the other hand, when resistance exceeds the value of R_z , CIE evaluates this state as Sabotage by Wire break. All states are graphically displayed in the following figure.

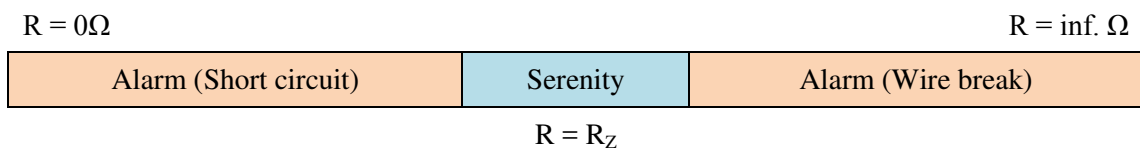


Figure 4: Graphical states of the End of line loop

Double End of Line

By adding another resistor in parallel to the alarm contact a new state can be created, which can be monitored. More detectors can be added serially and all values of resistors must be equal. In this case, it is possible to distinguish between Sabotage and real Alarm contact, but it is still impossible to distinguish between individual detectors in the loop.

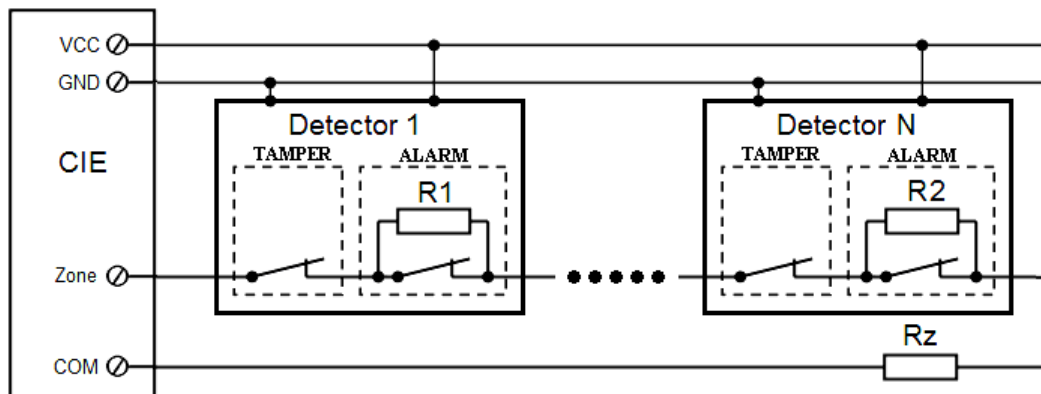


Figure 5: Double end of line loop[7]

In the Serenity state, total resistance of the loop is equal to R_z . The Alarm contact of the detector can be evaluated when the resistance is equal to $R_z + R_1$. This range can be shifted up to $R_z + (2xR_1)$ to cover more detectors. Greater resistance can be evaluated as Sabotage by the Wire break. When resistance drops under R_z , CIE identifies Sabotage by Short circuit.

$R = 0\Omega$

$R = \text{inf. } \Omega$

Alarm (Short circuit)	Serenity	Alarm	Alarm (Wire break)
$R = R_{\text{WIRE}}$	$R = R_z$	$R = R_z + R_1$	

Figure 6: Graphical states of Double end of line loop

End of line resistor must be much greater than other resistors, because voltage curve is logarithmic. Same values of resistors can cause too small a difference between Alarm and Sabotage by the Wire break.

Advanced Technology Zone

This variant uses different values of resistors. Different values can very precisely distinguish between individual detectors. Each detector has a resistor with a unique value. In real operation, multiple detectors are connected into several groups. Each group uses the same value of resistors. The real groups are shown in following figure.

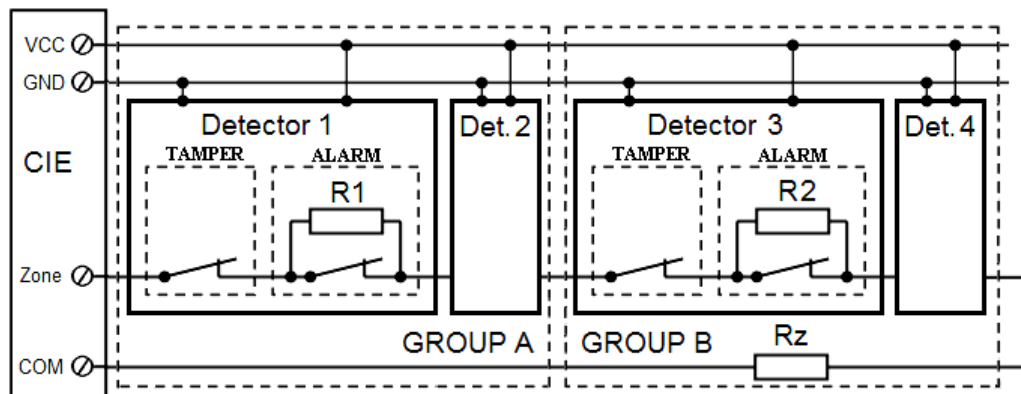


Figure 7: Advanced technology zone [7]

In general, the ATZ variant has four variants like the previous EOL variant. But each individual group can be distinguishable. It can be done by using different values of resistors. This variant is limited by the final numbers of used groups. Usual ranges of values are $R_1 = 10\text{ k}\Omega$, $R_2 = 30\text{ k}\Omega$ and $R_3 = 100\text{ k}\Omega$. Adding more resistors can result in difficulties in distinguishing between groups. Each group can contain up to three detectors.

$R = 0\Omega$		$R = \text{inf. } \Omega$		
Alarm (Short circuit)	Serenity	Alarm A	Alarm B	Alarm (Wire break)
$R = R_{\text{WIRE}}$	$R = R_Z$	$R = R_Z + R_1$	$R = R_Z + R_2$	

Figure 8: Graphical states of the advanced technology zone

The Serenity state is given by the R_Z . Alarm caused by the detector is given by the value of used group and R_Z . The previous figure illustrates all states.

1.2 Alarm Transmission System

Every modern CIE has a special connection, which is able to communicate with an Alarm Receiving Centre (ARC). This center can receive the alarm signal, and according to the received message can ensure the protection of property by armed intervention. This system can also transmit other data like photos or videos.

There are several ways to transmit the message to the ARC. Following bullets show the main transmission media:

- telephone communicator
- GSM communicator
- Ethernet communicator
- private radio transmission line

But only a few transmission media can guarantee faultless transmission to the ARC. GSM service using SMS is the most problematic. When the phone operator is busy, the alarm message cannot be delivered to the ARC in time. The most reliable way to send the alarm message is via a private radio transmission line. But on the other hand, this solution is also the most expensive.

This communication can be also used for the end customer, who can obtain the needed information from the system this way.

1.2.1 Telephone Communicator

This communication method is quite outdated nowadays. The house has to have its own home plug. With a connected phone line, the user needs to have a special device which automatically unplugs the phone line and releases its use to the alarm signal. The structure used by the protocol is described in next chapter.

1.2.2 Connections Using Ethernet

Ethernet is a modern communication medium with the ARC. The total baud rate is high enough for live broadcasting. Like in case of telephone communication, even Ethernet connection doesn't guarantee a faultless transmission to the ARC.

1.2.3 Protocols for Connection to Alarm Receiving Centre

There must be used special protocols which is understandable for both sides to ensure faultless transmission of the alarm signal to the ARC. Following text shows some of the most used protocols.

Protocol Ademco Contact ID (CID)

This protocol is used for the communication via the telephone line. It is based on tone dialing, which is called Dual Tone Multi Frequency (DTMF). Using the DTFM protocol, a message contains of 16 numbers. Features of individual numbers are following: [9]

- first two digits indicate version of the protocol
- four digits indicates the identification of the CIE
- three digits indicate message itself
- five digits transfers zones and detectors which causes the alarm
- last digit is checksum

The transmitting element in the CIE stores telephone numbers of one or more receivers in the ARC. In this protocol, CIE always initiates the alarm message. Communication in the opposite direction is not possible. [9]

Protocol IP Contact ID

This protocol is only an adaptation of CID to the TCP/IP protocol. It has the same features, but it has at least feedback of transmitted message from CIE to the final ARC. [9]

Protocol SIA IP Events Reporting

Security Industry Association (SIA) is a standardized universal protocol used in big corporations like Honeywell. Transmitted data can be encrypted using Advanced Encryption Standard (AES). Communication is also done by the TCP/IP protocol and it can transmit following messages:

- event message
- supervision message
- standardized message

2 INTEGRATED ALARM SYSTEMS

Integrated Alarm Systems are a modern form of connection between alarm applications which are complemented by other services. This connection creates a better and easier structure, which can affect the final customer. System can be used as commercial device but also in industrial environment. The system must follow the ČSN CLC/TS 50398 standard. The following sentence sums up the main idea of an IAS: **IAS is any system having one or more common devices where at least one of them is an alarm application.** [6]

The main element of IAS is always the alarm application. Common alarm applications are as follows:

- Intruder and Hold up Alarm Systems
- Social Alarm Systems
- Closed Circuit Television
- Access control systems
- Fire alarm systems

The opposite of the mentioned applications are non-alarm applications, which can be any other application. Usually, these applications can help to manage the customer's house. These include applications like:

- Lighting
- Heating
- Air conditioning
- Irrigation
- Energy management systems

The mentioned standard defines priorities which every system must follow. These priorities are sorted in ascending order:

- Alarm signals (protection of life, fire etc.)
- Alarm signals (asset protection, intrusion into the building)
- Alarm signals (other applications)
- Failure signals (protection of life and property)
- Failure Alarm (other applications)
- Information from non-alarm applications [6]

2.1 Configuration of Integrated Alarm Systems

According to ČSN CLC/TS 50398, configuration of IAS can be divided into several types. Each type has its own specification:

- Type 1
- Type 2

2.1.1 Type 1

Type 1 represents a combination of two or more single-purpose systems. Dedicated applications (alarm or non-alarm), are connected to a common optional device that is not required by the standard. The connection is realized using an additional transmission line. [6]

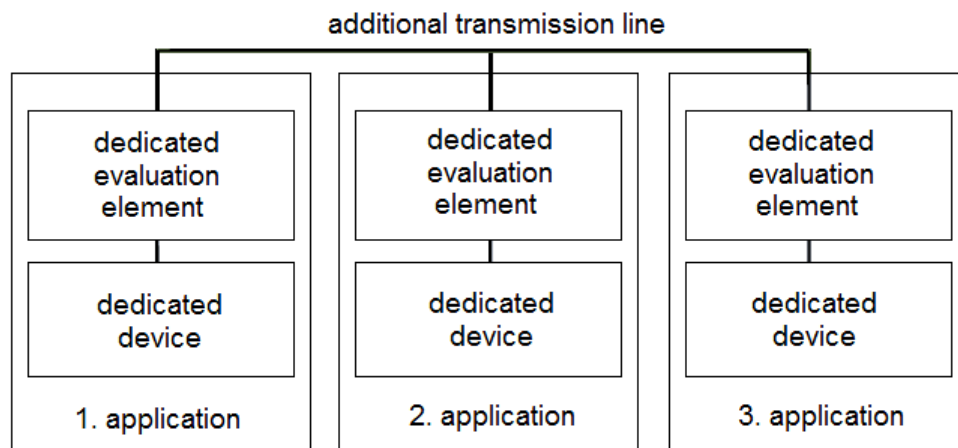


Figure 9: Type 1 configuration [6]

The previous figure shows how Type 1 can be created. The simplest way was picked for easy explanation. Dedicated devices are connected to a common complementary device via an additional transmission line. The device which is required by the standard in the alarm application must not be affected by any other dedicated or optional system in any operating condition.

This type should consist of a common evaluating device. According to the standard, this system must be in permanent surveillance by physical service. System also must be placed in the same room. [6]

2.1.2 Type 2

This type represents a combination of two dedicated systems which has a common device at least for one of the applications. The used device must follow the standard. According to the effect of the non-alarm application, this type is divided into two parts:

- Type 2A
- Type 2B

Type 2A

Type 2A represents a combination of multiple alarm or non-alarm systems. These systems use common devices and transmission lines. Failure of any application in the system has no impact on the integrity of the required device. According to needed requirements, there must be placed additional elements.

Type 2B

Type 2B has the same features as Type 2A except for possible failures. Failure of any application in the system can impact the integrity of the device required by the standard. It means that the integrity of the required device can be affected.

The following figure shows one possibility of connection between each element for both types A and B.

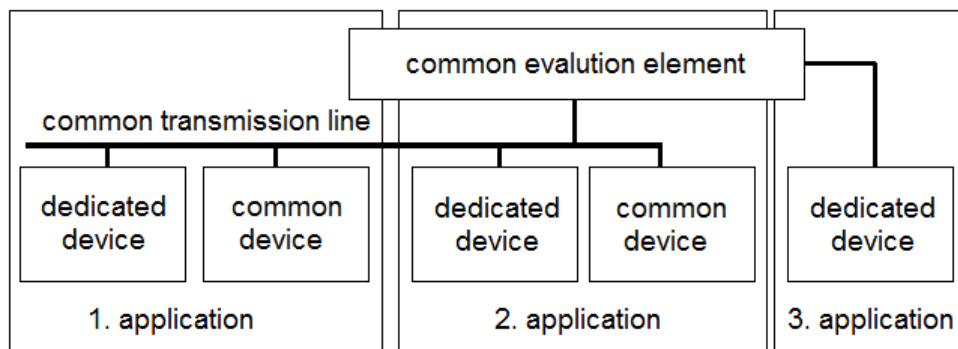


Figure 10: Type 2 configuration [6]

2.2 Software Integration

Software integration is based on the connection between individual applications via a common communication bus. The software product itself provides the integration. These products are mainly servers or computers. Each alarm or non-alarm application can be connected to the server via an Ethernet or USB port. [6]

Common access to the system is done by the mentioned computer or it can be accessible via mobile phone. System may have several functions:

- software of Intruder Alarm systems
- software for users
- software for visualization
- software for building systems

Software of Intruder Alarm systems

These systems provide a connection between CIE and the computer. The program running on the CIE can be easily changed via this software. It's mainly used for services, recovering or archiving of data. [6]

Software for Users

It can provide the user interface for accessing the system. The user can easily manage user data, create zones, sub-systems and also time tables. Changes are automatically saved and can be used in real operation. [6]

Software for Visualization

Visualization is very important for the customer. How the system works is an essential aspect, but the user interface to represent how it works is equally important. A graphical representation is much better for the customer. Software can easily display the real time operation. This information can be reached from any place on the planet using the internet connection.

2.3 Hardware Integration

Hardware and Software integration overlaps, because each hardware connection must also have some software reaction to the entering signal. Hardware integration is based on the physical connection among several components. The form of the connection can be one of two variants: [6]

- input and output connection
- bus connection

Both variants are based on the same connection form. Two or more elements of alarm or non-alarm type can be connected using data wires. Hardware integration uses interconnection between the outputs and inputs of several elements. Both sides must be able to transmit and receive the signal. In case of the bus connection, both sides must use the same protocol for communication.

Hardware integration can be applied on every element which has at least one programmable output. These basic elements are:

- alarm inputs
- programmable outputs
- output modules in general

By using hardware integration, individual elements can not affect each other. It means when one element is switched to failure state, other elements are not affected.

3 LEGISLATION AND STANDARDS

Every system intended for an end customer must follow the European standards. The most important standard is *EN 50131 Alarm systems - Electrical security systems*. It prescribes all information and parameters which every product must fulfill. The following table shows an overview of the standard's content.

Number of standard	Name of standard
EN 50131-1	System requirements
EN 50131-2	Intrusion detectors
EN 50131-3	The control panel
EN 50131-4	Alert devices
EN 50131-5	Requirements for devices that use radio connection
EN 50131-6	Power supply units
EN 50131-7	Guidelines for applications
EN 50131-8	Fogging security equipment

Table 1: Standard is EN 50131 [13]

Each part of the standard is focused on a specific device. Only those parts needed for this thesis were picked. These are the following parts:

- System requirements
- The control panel
- Power supply units

The needed information is listed on the following pages. Most information is in form of a number which specifies the threshold value of the current element. This information is used in chapter 8 **Software** (page 63). Each part is related to one of the listed parts of the standard.

3.1 System Requirements

This part contains the main requirements of alarm systems. The most important section of this part is Degree of Security of the Protected Object and Classification of Environment.

3.1.1 Degree of Security of the Protected Object

Every system must have a Degree of security assigned. The degree depends on the element with the lowest level of security. Each degree specifies the equipment of the expected intruder. The names are listed in the following table.

Degree of security	Name of Degree of security
1	low risk
2	low to moderate risk
3	moderate to high risk
4	high risk

Table 2: Degree of security[13]

3.1.2 Classification of Environment

Used component must be able to work correctly in a given environment. The expected environments are listed in following table. Each environment has its own description of possible conditions.

Class	Name of environment	Description	Temperature range
I	indoor	heated residential or business place	+5°C to +40°C
II	indoor general	intermittently heated or unheated place	-10°C to +40°C
III	outdoor protected	environment outside the building where the components are not permanently exposed to weather	-25°C to +50°C
IV	outdoor general	environment outside the building where the components are permanently exposed to the weather	-25°C to +60°C

Table 3: Classification of environment [13]

3.1.3 Classification of Failures

The system must be able to distinguish among possible failures. If the duration of the failure is longer than the time listed in the standard, the system must generate the failure signal. The following table shows all possible failures. Every failure can be Optional (O) or Mandatory (M) according to the degree of security.

Failures	Degree 1	Degree 2	Degree 3	Degree 4
Detectors	M	M	M	M
Emergency devices	M	M	M	M
Main power source	M	M	M	M
Alternative power supply	M	M	M	M
Interconnection	M	M	M	M
Alarm transmission system	M	M	M	M
Alert devices	M	M	M	M
Power supply failure	O	O	M	M
Monitoring function	O	O	M	M

Table 4: Classification of failures [13]

3.2 Control and Indicating Equipment

The following information is related mainly to the CIE. It means that every mentioned information must be followed according to the current Degree of security.

3.2.1 Detection of Invalid Attempts to Authorization

Every system must be protected against password cracking by bruteforce. The system must count every wrong attempt and generate alarm signal when the limit is exceeded.

Requirement	Degree of Security			
	1	2	3	4
Blocking the input of the user device. Maximum number of attempts before blocking.	O 10	O 10	M 10	M 3
Tamper signal or message. Maximum number of attempts before sabotage signal.	O 21	O 21	O 21	M 7

Table 5: Detection of invalid attempts to authorization [13]

3.2.2 Method of Connection Monitoring

Connection between all components which is mentioned in **Table 4: Classification of failures** [13] (page 27) should be constantly monitored. When connection is not available for an interval specified in the following table, the system must generate an alarm signal. Necessity and timing is also listed in following table.

Requirement	Degree of Security			
	1	2	3	4
System is armed. Timing.	M 400 ms	M 400 ms	M 400 ms	M 400 ms
Component substitution. Timing.	O O	O O	O 100 s	M 10 s
Maximum allowable Duration of unavailability of connections.	100 s	100 s	100 s	10 s
Maximum allowable interval of authentication.	240 min	120 min	100 s	10 s

Table 6: Method of connection monitoring [13]

3.2.3 Event Log and Memory Capacity

According to Degree of Security, every event should be saved in the system memory. The necessity, minimum number of events and term of duration are listed in the following table. The event log should be in understandable form.

Requirement	Degree of Security			
	1	2	3	4
Minimum number of events.	O	250	500	1000
Minimum durability of memory after power failure.	O	30 days	30 days	30 days

Table 7: Event log and memory capacity [13]

3.3 Power supply units

The power supply unit provides power for the whole system. In case of power failure, the unit has its own battery which can provide the needed power. Specific features exist which must be followed for the battery and the whole system. There are three types of power supply units:

- Type A
- Type B
- Type C

Type A: The power is supplied from an external source and in case of power failure the power is taken from an alternative source (battery). This source is automatically charged from an external source. [13]

Type B: The power is supplied from an external source and in case of power failure the power is taken from an alternative source (battery). This source is not automatically charged from an external source. [13]

Type C: The power is supplied only from an alternative source (battery). [13]

In case of power failure, the alternative power source must be able to power the whole device for the time specified. The time depends on the Degree of Security used. Limit values for each degree are shown in following table.

	Degree 1	Degree 2	Degree 3	Degree 4
Type A	12 h	12 h	60 h	60 h
Type B	24 h	24 h	120 h	120 h

Table 8: Minimum duration of alternative supply [13]

After the recovery of the external power supply, the battery must be charged again up to 80%. Also this time is strictly specified according to the used Degree of Security.

	Degree 1	Degree 2	Degree 3	Degree 4
Duration of charging	72 h	72 h	24 h	24 h

Table 9: Maximum duration of battery charging [13]

4 ATMEL MICROCONTROLLERS

Atmel corporation provides a wide selection of microcontrollers to their customers. Some of the microcontrollers are designed directly for the automotive industry or for health care. To satisfy the maximum number of customers, the assortment of microcontrollers is really huge. Production is divided into two general classes:

- AtmelAVR 8-bit and 32-bit
- SMART ARM 32-bit

4.1 Atmel AVR 8-bit and 32-bit

These microcontrollers are characterized by low power consumption and a high level of integration. Low consumption also results in low computing power. According to the power and bus width, AVR microcontrollers can be divided into the following groups:

- 32-bit AVR UC3
- AVR XMEGA MCU
- megaAVR MCU
- 8-bit tinyAVR MCU

Every microcontroller comes with a specific code name, which indicates basic information about the microcontroller. The number after general letters, which refers to the used group, shows the amount of flash memory. The following group of letters shows the features. Description of these letters can be found in following table.

Letter	Description
A	Second Revision - usually only a change of the internal structures without any impact to the user, some with an internal temperature sensor.
B	Third revision - some improvements: Improved ADC, integrated serial number (9 byte) and UART wake-up in the "power-down" state.
L/V	Low Voltage - Especially for lower clock speeds (8 and 10 MHz) and lower input voltages (1.8 and 2.7 V) of selected blocks.
P/PA	Piko Power - Reduced power consumption, especially in deep sleep mode (<1 μ A); Some modules (e.g. B. ATmega48) is available as P and PA.
HV/HVA	High Voltage - Special models with peripheral units for controlling a battery charger can be operated with up to 18 V.
RF	Radio Frequency - models with integrated transceiver for the 2.4 GHz ISM band.

Table 10: Description of used codenames [2]

4.2 SMART ARM 32-bit

These microcontrollers are based on a 32-bit bus and they are constructed for more sophisticated operation which needs more computing power. This concept is based on “system on chip”, which means that microcontroller has many extensions built-in for uses such as in industrial, automotive or health care systems. Microcontrollers ARM are named *Cortex-X*. The following table lists the main division of ARM microcontrollers. [14]

ARM Cortex-M	Hardware	ARM architecture
Cortex-M0	32-bit	ARMv6-M
Cortex-M0+	32-bit	ARMv6-M
Cortex-M1	32-bit	ARMv6-M
Cortex-M3	32-bit or 64-bit	ARMv7-M
Cortex-M4	32-bit or 64-bit	ARMv7E-M
Cortex-M7	32-bit or 64-bit	ARMv7E-M

Table 11: ARM Cortex-M variations [14]

The name of these microcontrollers comes from the abbreviation of “SMART Atmel microcontroller” (SAM). The development of these microcontrollers is shown in the following figure.

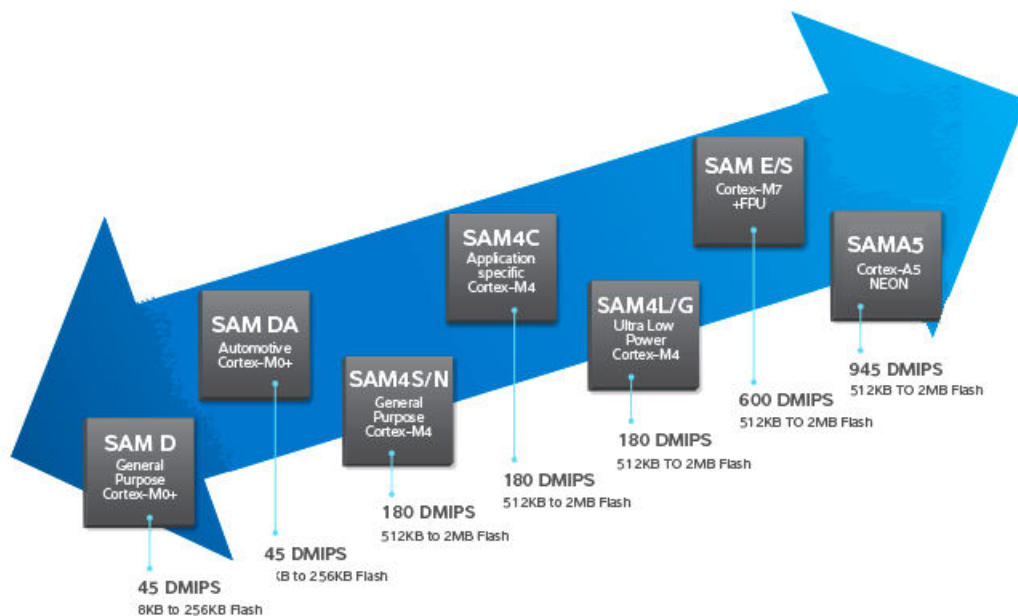


Figure 11: 32-bit ARM microcontrollers [14]

4.4 Interrupts

Generally, the interrupt is a signal to the processor which is initiated by external hardware or software. The processor has to react to this signal and execute the respective subroutine or function according to the Interrupt Service Routine (ISR) table. The following figure shows an example of actual operation.

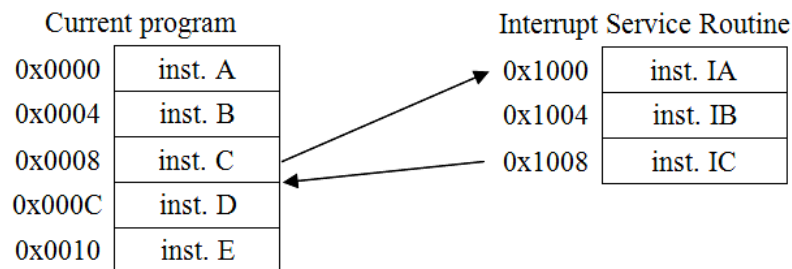


Figure 13: Interrupt Service Routine [4]

When an interrupt occurs in the *inst. C* program, the Program Counter automatically jumps into the ISR and executes the respective code. After that, the program continues back in the main program where it was interrupted. In general, interrupts can be divided into two sources:

- software
- hardware

Software Interrupt

A software interrupt can be initiated by the program itself. It is mostly used in software timing, when a predetermined time is reached and the program automatically generates the interrupt. More information about timing using interrupts are listed below in the following chapter **4.4.1 Interrupt Service Routine** (page 34).

Hardware Interrupt

A hardware interrupt is mostly used by external devices which are connected to the processor. By using interrupts, the processor doesn't need to check all devices repeatedly, but the device can ask for attention by using the interrupt. The interrupt wire must be connected to the processor's input, which can handle the interrupt signal. Atmel provides a complete list of ports in the datasheet. Some interrupt inputs particular to the *ATmega2560* can be found in the following table.

Vector No.	Program Address	Source	Interrupt Definition
1	\$0000	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset
2	\$0002	INT0	External Interrupt Request 0
3	\$0004	INT1	External Interrupt Request 1
4	\$0006	INT2	External Interrupt Request 2
5	\$0008	INT3	External Interrupt Request 3
6	\$000A	INT4	External Interrupt Request 4
7	\$000C	INT5	External Interrupt Request 5
8	\$000E	INT6	External Interrupt Request 6
9	\$0010	INT7	External Interrupt Request 7
10	\$0012	PCINT0	Pin Change Interrupt Request 0

Table 12: Reset and Interrupt Vectors [4]

Sometimes a problem may occur that two or more external devices ask for the interrupt at the same time. Every interrupt has its own priority which is shown in the previous table. The number of each interrupt is also the priority value for the interrupt. When two or more interrupts occur, the interrupt with the greater priority will be executed first. [4]

4.4.1 Interrupt Service Routine

ISR is a space in the program memory which stores the code which will be executed when an interrupt occurs. For every interrupt there must be one ISR. When an interrupt is detected, the processor must run the ISR. There are several steps which must be done when an interrupt occurs:

- current instruction must be completed
- address of the next instruction must be saved on the stack
- Program Counter is initialized with the address of the corresponding ISR
- ISR is executed until its last instruction
- program continues from where it has been interrupted

4.5 Timers

Timing is very important while using some time-based functions or operations. Not all processors have a built-in Real Time Clock (RTC), which generates actual time. So it can be done externally, outside of the processor by another element, or it can be done inside the processor using mentioned interrupts. Timers based on interrupts can be divided into two parts:

- timer generates rollover interrupt
- timer generates interrupt in compare mode

4.5.1 Timer generates rollover interrupt

Using the rollover interrupt the interrupt itself generates the signal when the *TCNTx* counter overflows. The example below explains this method.

Example: CLK = 4MHz, prescaler 8, 8 bit timer.

$$f = \text{CLK}/\text{prescaler} = 4\text{MHz}/8 = 500\text{kHz}$$

$$\text{Time} = 1/f = 1/500\text{kHz} = 2\mu\text{s} \text{ (one clock pulse)}$$

$$256 * 2\mu\text{s} = 512\mu\text{s} \text{ (counter overflows)}$$

Figure 14: Example rollover interrupt [4]

Each clock pulse increments the *TCNTx* register by one. To achieve counter overflow in a 100 clock pulse, counting must start from 156 (256-100). When the interrupt occurs, the *TCNTx* register always reinitializes.

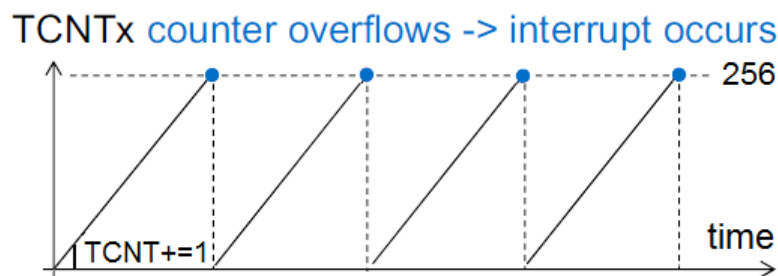


Figure 15: Timer using a rollover interrupt [4]

4.5.2 Timer generates interrupt in compare mode

In compare mode, interrupt is generated when registers $TCNTx$ and $OCRx$ are equal. The example in the following figure is also provided for current method with the same conditions as in the previous example.

Example: CLK = 4MHz, prescaler 8, 8 bit timer.

$f = \text{CLK}/\text{prescaler} = 4\text{MHz}/8 = 500\text{kHz}$

Time = $1/f = 1/500\text{kHz} = 2\mu\text{s}$ (one clock pulse)

Figure 16: Example compare mode [4,1]

The $TCNTx$ register in this example starts from zero, and the needed time value is loaded to the $OCRx$ register. Also in this case, the value is equal to 100. When the counter reaches the value 99, the condition $TCNTx == OCRx$ becomes true and an interrupt is generated.

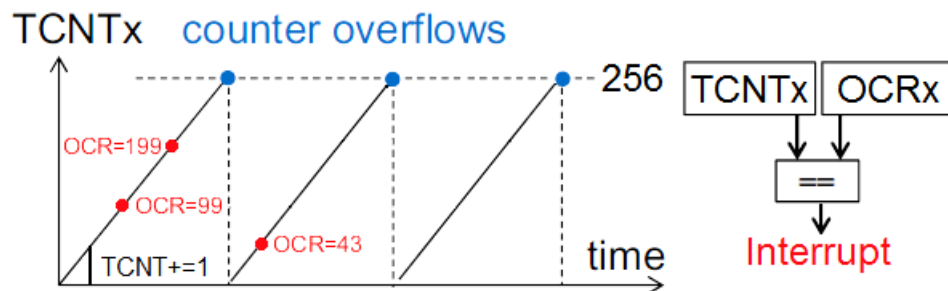


Figure 17: Timer using compare mode [4,1]

The figure shows a more complicated system of shifting the number which causes the interrupt. This is all done by the function of the used interrupt.

4.6 Serial Communication

This chapter is about the external communication standards which can be used with Atmel microcontrollers. There are many standards but only a few, which will be used in the final device, were picked. These protocols are following:

- Universal Asynchronous Receiver and Transmitter
- RS-232
- Serial Peripheral Interface

4.6.1 Universal Asynchronous Receiver and Transmitter

In general, a Universal Asynchronous Receiver and Transmitter (UART) is a module which provides serial communication between two devices. A device usually has a transmitting pin (TX) and a receiving pin (RX). The signal can have an adjustable voltage value, which must be positive (usually 3.3 V or 5.0 V). The voltage must be equal on both sides. The following figure shows the transmission of the value 0x33, which is char "3" in ASCII.

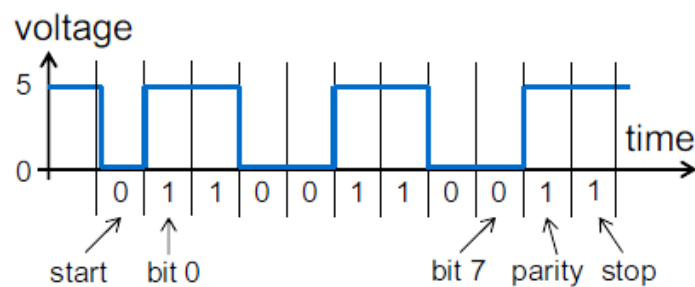


Figure 18: Universal Asynchronous Receiver and Transmitter [4]

Data are sent in serial order bit by bit. The idle state of the link is represented by the logical 1. The transmission itself begins with logical 0. Following bits are data and transmission stops with logical 1, same as the start bit. The transmission can be held in several speeds, most common speeds are following:

- 9 600 bps
- 19 200 bps

With increasing speed the noise on the line also increases. This can limit the maximum length of the cable.

4.6.2 RS-232

RS-232 is based on UART communication as described in the previous section. The main difference is in operating voltage. Representation of logical 0 and logical 1 is based on opposite polarity. Logical 1 can take up to +15 V and logical 0 can take down to -15 V. Communication can be complemented by additional command signals like *RTS*, *CTS*, *DTR*, *DSR*. Usage of these signals is optional. An actual representation of RS-232 is shown in the following figure.

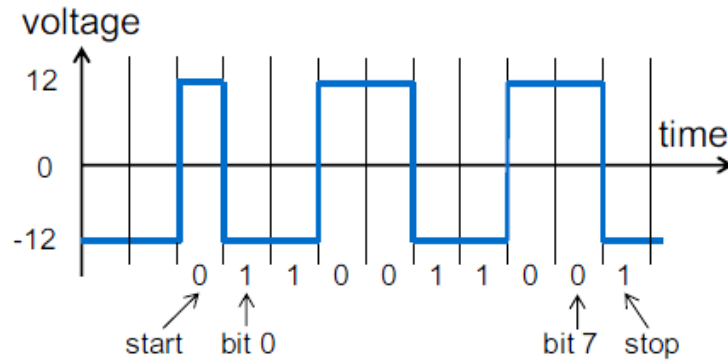


Figure 19: RS-232 [4]

Like in the previous UART example the transmission of an actual value 0x33 is shown here. Voltage values must be exactly opposite to comply with the standard.

4.6.3 Serial Peripheral Interface

Serial Peripheral Interface (SPI) is mainly used for the communication between microcontroller and other integrated circuits. It uses a common communication bus shared by the commands and data signals. Voltage level depends on the operating voltage of the microcontroller, but it is typically 5.0 V. Sometimes the operating voltage of microcontroller and the connected IC maybe different, in which case a voltage level shifter must be used.

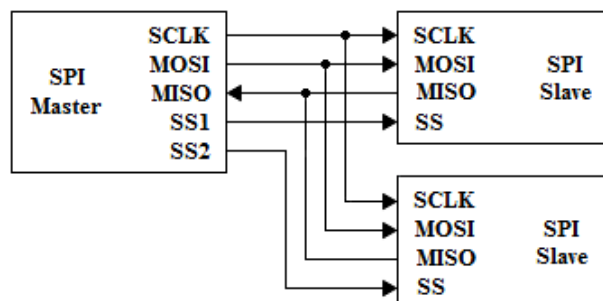


Figure 20: Principle of SPI communication

In general, the interface has four signals *SCK*, *MOSI*, *MISO* and *SS*. *MOSI* and *MISO* are the signals for data itself. *SCK* is the clock signal which is generated by the master microcontroller. One of the connected slave microcontrollers is selected by the *CS*. The number of connected devices is not limited, but every device must have its own *CS* signal from the master microcontroller. Sometimes the *CS* pin is also called *SS*.

II. ANALYSIS

5 REQUIREMENTS FOR THE SYSTEM

The main goal of this thesis is to create a functional prototype of Control and Indicating Equipment (CIE). System should be built from the same components and built from techniques as certificated CIE. These elementary components and features are following:

- final system placed in a solid box
- uninterruptible power supply
- programming via USB
- user interface via web pages
- at least eight alarm loops with adjustable modes (NC, EOL, DEOL and ATZ)
- graphical keypad

All these requirements should be achieved. Each previous bullet is described in this chapter. They are divided into requirements for the CIE and Keypad.

5.1 Control and Indicating Equipment

Requirements for the CIE are listed in this chapter. All devices should be built according to the general standards of IAS which are mentioned in chapter **3.2 Control and Indicating Equipment** (page 27).

Final system placed in a solidbox

Just like the regular CIE, also the created prototype should be placed into a solid box to protect it against bad conditions, dust or potential intruder. Connectors like power supply, USB, Ethernet, RS-232 and a hole for loop cables should be on the sides of the box. The box should include an openable small door.

Uninterruptible power supply

An uninterruptible power supply should be a part of the system. Actual requirements of the UPS depend on the connected devices. In design part, the final devices are not known yet. But the power supply should be able to provide at least 25 W. The battery, which is the main component of UPS should have a capacity at least 1000 mAh.

Programming via USB

The final system should be able to communicate with the connected computer via USB. The created channel should be also used for programming the devices inside. The system should consist of many programmable devices so for better comfort only one USB output channel is allowed.

User interface via web pages

Setting of all possible features should be controlled mainly via the created web interface. A web server must be created inside of the created system to allow for controlling the system without any internet connection.

The web client should offer secure login into the system using password. The main user should be able to manage other users and also modify loops, created zones and other features.

At least eight alarm loops with adjustable modes (NC, EOL, DEOL and ATZ)

The system should include also a part which controls the connected detectors. Loops should be in physical form and the system should be able to control all possible modes such as NC, EOL, DEOL and ATZ. Individual settings of mentioned loops should be available via the web interface.

5.2 Keypad

The keypad will be located outside of the CIE. Like all communication channels also the keypad should be connected using metallic cable. The keypad should be connected via RS-232. It also should be placed in a box to protect it against bad conditions and potential intruders.

Graphical keypad

The keypad should be the main element which allows the user to arm or unarm the system. The keypad should also provide basic applications of created function via the web pages. For comfortable control, a touch screen display should be used.

6 CONTROL AND INDICATING EQUIPMENT

All the used components are embedded into a special aluminum box which makes it look like box. The box has a small door without any lock. Unauthorized opening can cause an alarm signal according to actual program, which is listed in chapter **8.1 Main Program** (page 63).

The box has several openings for the output connectors such as USB, Ethernet and RS-232 on the front side. The back side opening is reserved to Zone cables. This opening is quite big, because in case that all zones are occupied it must hold up to 30 cables. It also has AC voltage input.

In the following figure the block diagram of the created CIE is shown. All components are placed into the aluminum box. Final device can be found in **Appendix VIII: Created CIE** (page 92).

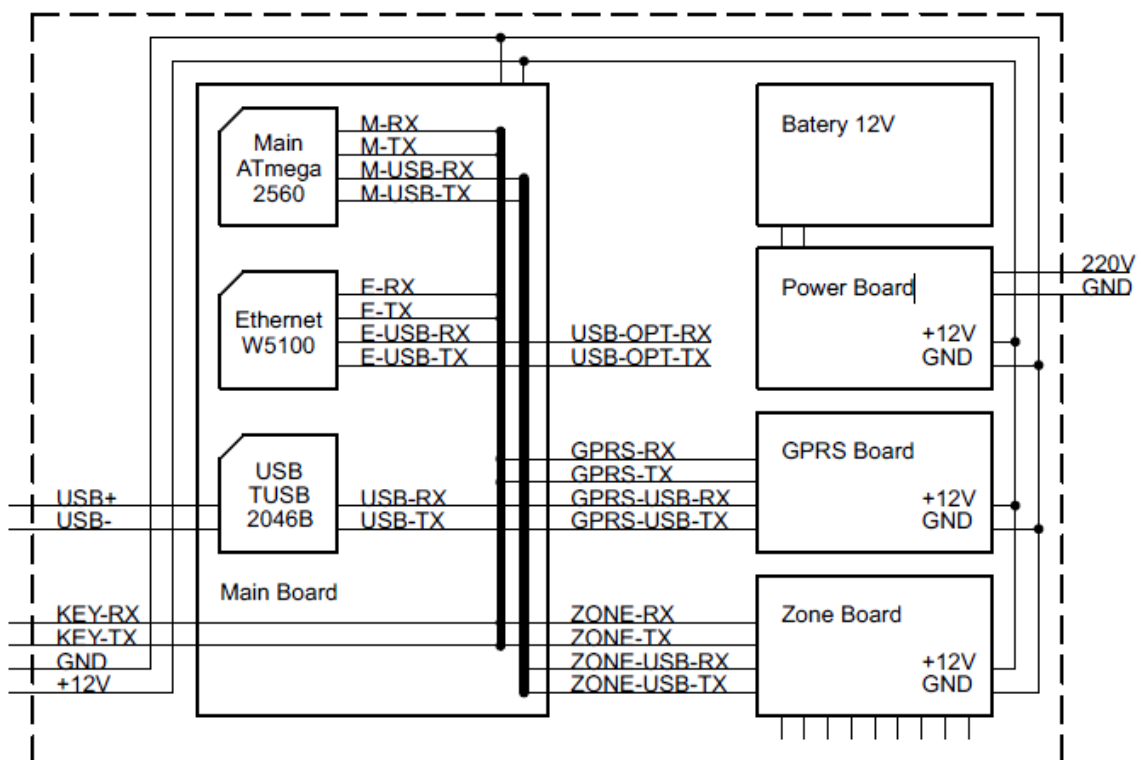


Figure 21: Created block diagram of the CIE

6.1 Uninterruptible Power Supply

According to chapter 1.1 **Control and Indicating Equipment** (page 11), one of the used components must be a part that provides power supply in case of power failure. Due to the final cost, it was decided to use final product of UPS. The used component must be able to accept 230 V input voltage, store backup voltage in a lead battery and provide stable 12 V output voltage.



Figure 22: Uninterruptible power supply [12]

The UPS device provides two channels: *CH1* and *CH2*. *CH1* provides output voltage for the connected device and *CH2* provides output for the connected lead battery. It has built-in charging module which keeps the battery charged. It has also many extra features like overcurrent, short-circuit and overheating protection. The UPS performs all these functions automatically. [12]

The actual state of charge can be observed by LEDs added at the factory. For better usage, all indicators can be connected to the Main Board, which can react to the current situation.

These situations are:

- DC output indicator
- AC input indicator
- battery charge indicator

6.2 Main Board with Ethernet Interface

Originally, there were supposed to be two different boards. One as a Main Board only for the main program and the second Ethernet Board was supposed to be responsible for communication via Ethernet. But due to the final cost and overall complexity, it was decided to unite these boards into one.

The Main Board contains several parts, integrated circuits and microcontrollers. These elements are described in detail on the following pages. The main board consists of these parts:

- Main microcontroller
- Ethernet microcontroller with W5100
- USB Hub

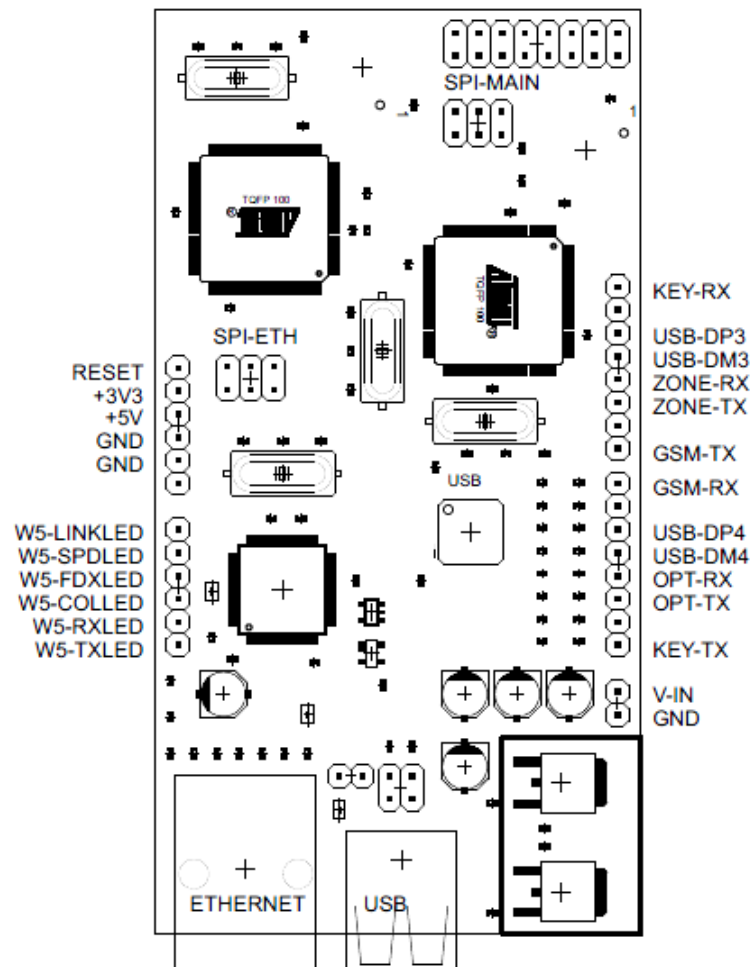


Figure 23: Main Board

6.2.1 Main Microcontroller

The main microcontroller is on the top of the created hierarchy. It can receive data from other devices and it is also able to transmit data to other microcontrollers. The main microcontroller is model *Atmega2560*, which is described in chapter **4.1 Atmel Microcontrollers** (page 30). The microcontroller communicates with other microcontrollers via serial UART interface, which is described in chapter **4.6.1 Universal Asynchronous Receiver and Transmitter** (page 37).

The *Atmega2560* microcontroller has embedded channels for the mentioned UART interface using so-called Hardware Serial. Moreover the first channel is reserved only for communication via USB with the computer. Due to this limitation was, it was decided to use so-called Software Serial, which can be implemented on any pin.

Software Serial is implemented using two pins - one for the Transmitter (TX) and second for the Receiver (RX). The transmitter pin can be connected to any output pin. The Receiver pin must be connected to internal interrupt (PCINT). Using this function it is able to interrupt the main program and receive the incoming data. In this project, four channels are needed to communicate with the connected devices:

Device	Receiver pin number	Transmitter pin number
Keypad	DP13 (PCINT7)	DP06
Zone Board	DP11 (PCINT5)	DP07
GSM Module	DP10 (PCINT4)	DP08
Ethernet	A08 (PCINT16)	DP03
Optional device	DP12 (PCINT6)	DP09

Table 13: Software Serial interface for the main board

The last Optional channel has no direct connected device. It can be connected to a device with an UART interface. The number of optional devices depends on the total number of pins which have internal interrupt. The number of these pins is 23. So there can be up to 18 devices connected in total. But for these pins it is necessary to physically add more connections in the schematics, because not all of pins are wired out of the board. [3]

The final schematics and the design of the board can be found in the **Appendix VIII: CD with thesis, source code and EAGLE files** (page 92).

6.2.2 FT232RL Universal Serial Bus Interface

Every *Atmega2560* is able to communicate with the computer via UART. But modern computers and notebooks typically do not have this interface built-in. For easy and comfortable communication a special integrated circuit is used which can convert UART communication to USB. This device is called *FT232RL*.

This circuit does not require any other special settings. The data cables from the USB connector are connected to the USB-DM and USB-DP pins. Converted UART interface is able on *RXD* and *TXD* pins. The *_DTR* reset pin must be connected to the master reset circuit and all of the power lines according to the official datasheet.

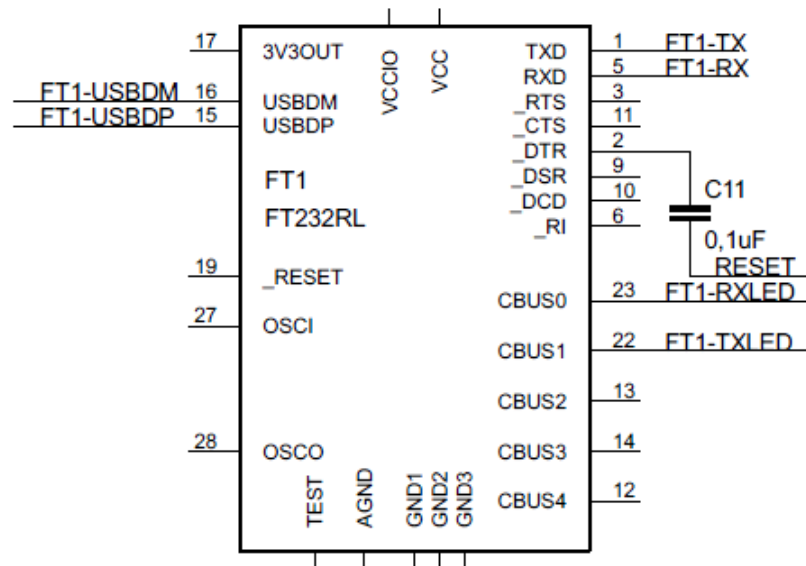


Figure 24: FT232RL connection

This integrated circuit is used in all the created boards. It is recommended to add a 1 k Ω resistor in between the circuit and the *Atmega2560*, which is also added in the schematics. One of the outputs of the IC is also the voltage 3V3OUT as a power voltage to the USB electronics. But the board already contains a voltage regulator to 3.3 V. So this pin is not used.

In the original boards made by the Arduino company a smaller microcontroller *ATmega16U2/8U2* is used instead of *FT232RL*. The *FT232RL* has pre-installed firmware for the mentioned communication. That is the reason why it is used instead of the other microcontroller.

6.2.3 Ethernet Microcontroller with W5100

One of the main goals of this thesis is to create easy and suitable interface for the end user and also for the technical support. Using another microcontroller which provides the Ethernet communication makes it very easy to add user interface via a website. It can be reached from all devices connected to the Ethernet network. At the same time the connection can be used as an interface to send the alarm signal to the Alarm Monitoring Center (AMC). More information about this can be found in chapter **1.2 Alarm Transmission System** (page 18).

Ethernet communication itself is performed by the **W5100** microcontroller made by the WIZnet company. It is a single-chip Ethernet controller and no separate operating system is required. The main features of the chip are: [5]

- supports TCP/IP protocols
- supports 10BaseT/100BaseTX Ethernet PHY
- supports Serial Peripheral Interface

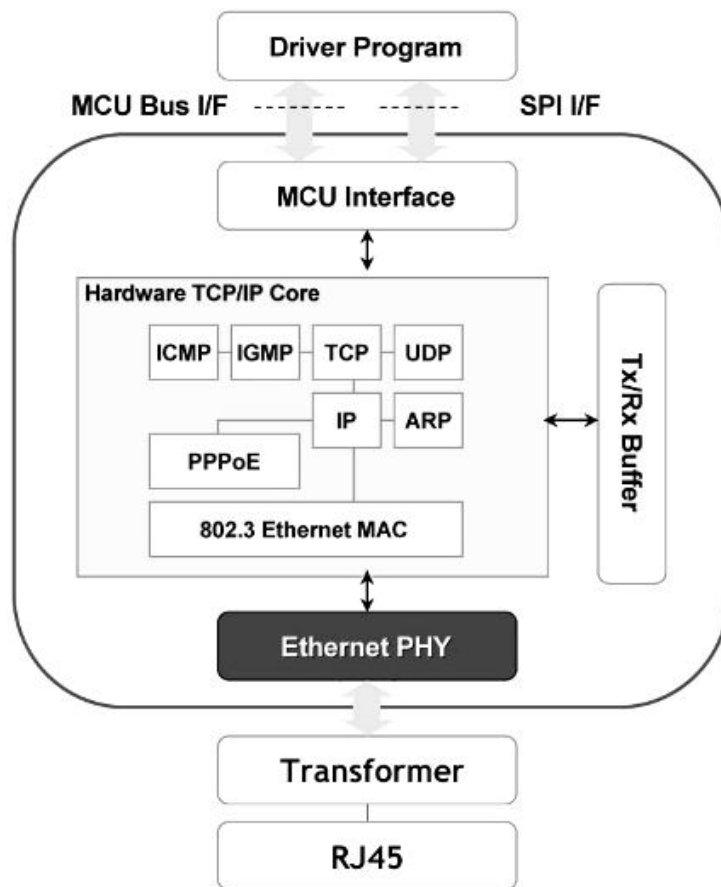


Figure 25: Block diagram of the W5100 [5]

The previous figure shows the block diagram of *W5100*. The communication with other microcontrollers can be done via so-called Bus Interface mode, using command pins (*/CS*, */RD*, */WR*, */INT*), address pins (*ADDR[14:0]*) and data pins (*DATA[7:0]*). But it can also communicate using SPI, which is described in chapter 4.6.3 **Serial Peripheral Interface** (page 38).

The *ATmega2560*, which is connected as the command microcontroller, is able to communicate via the SPI. In order to simplify SPI wiring, it was decided to use this method of communication. While using SPI communication, the microcontrollers must be wired according to the following rules. All unused address pins (*ADDR[14:0]*) and data pins (*DATA[7:0]*) must be wired to GND. Command pins (*/CS*, */RD*, */WR*) must be connected to the VCC. [5]

The SPI connection itself is shown in the following figure.

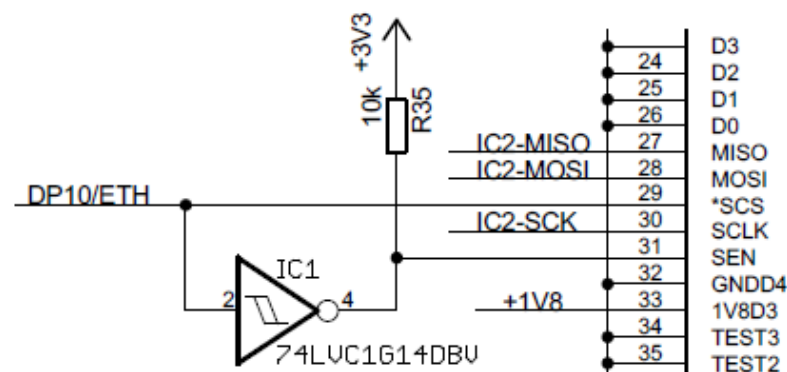


Figure 26: SPI connection of the W5100

A wire called *DP10/ETH* leads from the command microcontroller. This pin is used for the selection of the Ethernet chip. *SCS* pin is active low, so the signal from the command microcontroller must be logical zero to select the Ethernet chip. At the same time another pin of the Ethernet chip called *SEN* must be set to high. This is done by the *IC1* logic inverter, which flips the logical signal.

While using the SPI interface for communication, all data are sent as a text string. The final program for the Ethernet microcontroller is listed in chapter 8.2 **Ethernet Program** (page 67). Data sending itself is done by the Ethernet library, which is supplied by Arduino.

The **W5100** has also a verification of all possible states. These states are represented by the output pins which can be connected to the LEDs. In order to ensure minimal power consumption, no LEDs are connected. These pins are wired to the pinheads on the board. Arrangement of these pins can be found in the **Figure 23: Main Board** (page 44). Description of each pin is listed in following table.

Label	Description
W5-TXLED	presence of receiving activity
W5-RXLED	presence of transmitting activity.
W5-COLLED	presence of collision activity.
W5-FDXLED	status of full-duplex mode
W5-SPDLED	indicates 100Mbps link speed
W5-LINKLED	always ON when the link is OK

Table 14: LED status of W5100

The trickiest part of this board is connection to the Ethernet connector. Physical pins for the Ethernet connection are **RXIP**, **RXIN** for the receiving and **TXOP**, **TXON** for the transmitting part. These pins are directly connected to the special Ethernet connector which contains the isolating transformers. A special filter is added between the parts to protect against electro-magnetic interference. Wiring is shown in the following figure.

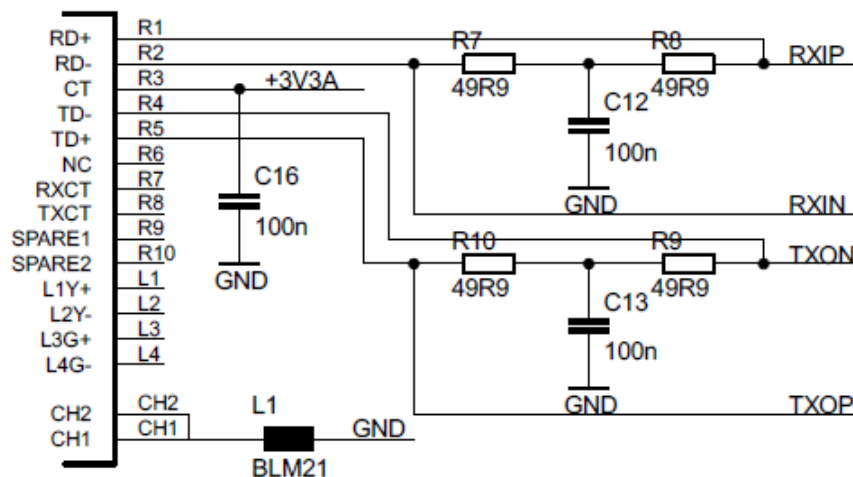


Figure 27: Wiring of Ethernet connector

In this case, the **UDE BS-RB10202** connector is used. The same component is used by the Arduino in their Ethernet shield.

6.2.4 Universal Serial Bus Hub

Every microcontroller must be connected via *FT232RL* to the USB interface for uploading and modification of its program. It means that each of four microcontrollers requires its own channel. It is not convenient to have four output USB ports on each single board. Due to this problem, an USB hub was added. To minimize the number of additional boards, it was decided to put the chip itself on the final board.

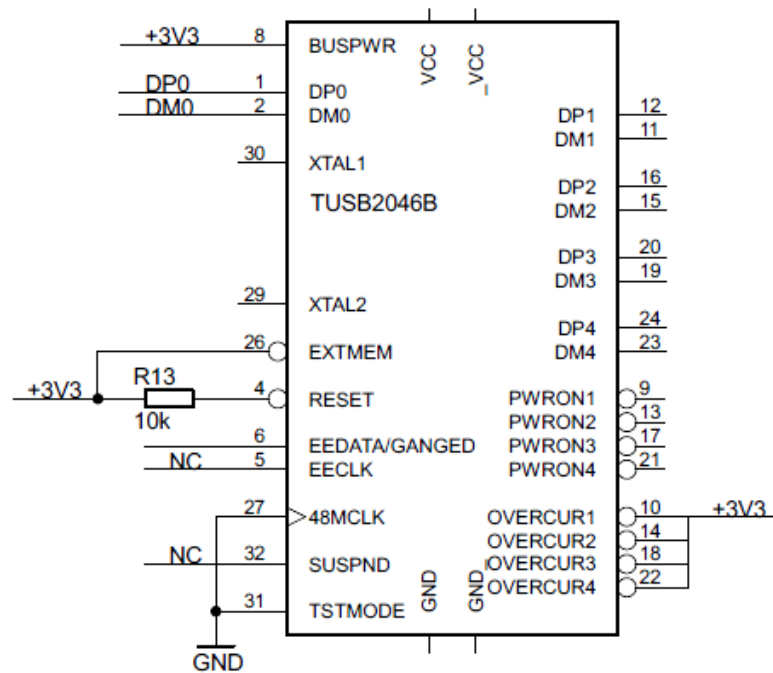


Figure 28: USB hub TUSB2046B

The used chip is called *TUSB2046B* and it can provide up to four channels for connected devices. This chip obviates the need for redundant USB connectors and saves a lot of space on the board. Correct wiring is very important for proper function. The chip can be used to access an external memory, but in this case this function is unnecessary and must be turned off. When the *EXTMEM* pin is high, the function is disabled.

TUSB2046B supports two ways how to power downstream channels. The first is bus-powered and the second is self-powered mode. The board has its own power management, so in this case the self-powered mode is used. It means that the *BUSPWR* pin must be low. The chip also has overcurrent detection on the input. Because all the connected devices have its own power management, this function must be turned off. Every channel has its own detection and all pins are active low, so all overcurrent inputs are high and activation never occurs.

6.2.5 Power Management

All the boards have the same 12 V input voltage. Each board has a voltage regulator to obtain suitable voltage for the connected microcontroller and chips. The main voltages which are needed are +5V (5 V) and +3V3 (3.3 V). The Zone Board only uses 12 V as power supply for connected detectors.

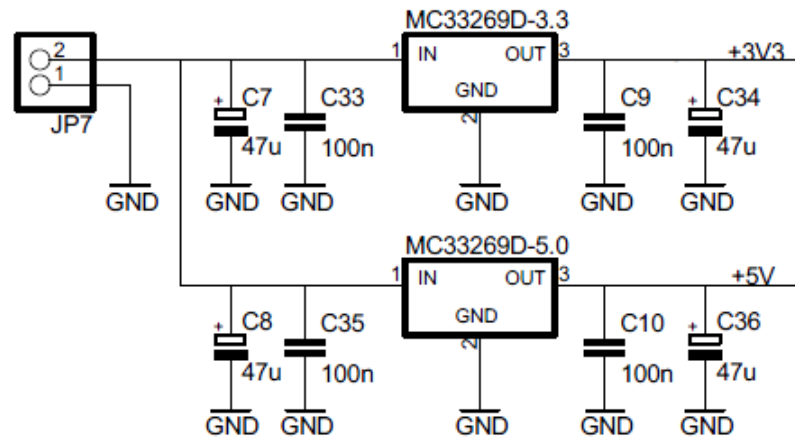


Figure 29: Power management

The SMD chip *LFXCDDT* is used as voltage regulator with one regulator for each voltage. Maximum output current for both chips is 500 mA. The voltage regulator produces some heat depending on the current load. A heat sink is placed on each regulator, which helps to dissipate the heat. Also the layer of copper under the chip is conducted to the bottom side where the heat can also be drained.

Each chip also has a couple of input and output capacitors. A ceramic capacitor reduces random peaks which can occur and an electrolytic capacitor reacts to any potential voltage drop.

6.3 Global System for Mobile Communication Module

Factory-made GSM module was used to decrease the final cost of device. The module is made by the GeeTech company. It is much cheaper than the official model from the Arduino company, it has the same features and it is smaller. This shield provides three methods of remote communication with the device. These methods are following:

- Short Message Service
- Audio Voice Call
- GPRS Service

The shield itself is very compact and has all the needed components. The main component for the communication is the SIM card holder placed on the bottom side of the shield. The holder uses the standard mini-SIM format (2FF). Next, it has an antenna interface, which must be connected to the external Athena to achieve sufficient signal strength. In this case, the built-in antenna interface is not used. An external connector for the antenna is placed on the side of the final box.



Figure 30 GSM Shield by GeeTech [10]

Communication with connected microcontroller *ATmega2560* is done via the UART interface. It allows communication using so-called AT commands. It means that text strings are used for sending commands. No other library is necessary. To achieve maximum flexibility, *GeeTech* decided not to connect communication directly to some pins. A pinhead with UART communication with the microcontroller *SIM900* is placed on the shield. By changing the jumpers the communication mode can be switched between software and hardware serial. This is shown in the following figure.



Figure 31 UART of the SIM900 [11]

In this case, the Software Serial pins are used to communicate with the Main Board. Physical pins of the communication are the following:

- GPRS-TX (Digital Pin 08)
- GPRS-RX (Digital Pin 10)

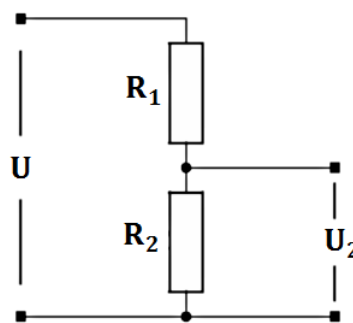
These pins do not comply with testing codes found on Geeetech's website. Pin for **GPRS-TX** can be any free pin for sending AT commands, but the **GPRS-RX** must be connected to the pin which can handle the internal interrupt. Digital Pin 10 is connected to the input of PCINT4, which means that this pin is able to generate the interrupt.

The main usage for this shield is to allow sending SMS and email as a notifications to the customer. It can be also used as an input to control temperature, lights or any other devices. Examples of communication are described in the chapter **8.1.2 GSM Program** (page 66).

6.4 Zone Board

The Zone Board contains 16 identical zones. The microcontroller, where each zone is connected, operates up to 5 V. Higher voltage can destroy it. On the other hand, all commercial detectors are designed to operate at 12 V. So every loop contains a voltage divider, which delivers safe voltage to the input of the ADC and the microcontroller can process the input voltage.

The actual voltage divider consists of two resistors. The input voltage is divided in the specified ratio according to the formula shown in the following figure.



$$U_2 = U \cdot \frac{R_2}{R_1 + R_2}$$

Figure 32: Voltage divider

The input voltage from power supply is reduced by the voltage drop, which is consumed by the protective diode. The voltage drop is precisely 1.1 V. So correct input voltage to the divider is 10.9 V. The main goal is to achieve maximum voltage at U_2 under 5 V. In this case resistors with values $R_1=12.4 \text{ k}\Omega$ and $R_2 = 10 \text{ k}\Omega$ are used. By using these values, even direct short circuit does not destroy the connected ADC. The calculation of maximum voltage is shown in following formula.

$$U_2 = U \cdot \frac{R_2}{R_1 + R_2}$$

$$U_2 = 10,9 \cdot \frac{10k}{12,4k + 10k}$$

$$U_2 = 4,87 \text{ V}$$

The main effect depends only on the ratio of connected resistors, but for insignificant value of the current, values in the order tens of k Ω are used. Resistance is inversely proportional to the current and for low power consumptions low current level is needed.

Every zone is also protected against overvoltage, which can be applied outside of the main box by the potential intruder. This protection is done by the component which is called transil and resettable fuse. Transil acts as a normal diode. In this case, every transil has the value of threshold voltage set to exactly 12 V. When the threshold voltage is exceeded, transil switches to short circuit mode and voltage is passed to GND. At the same time short circuit current through the fuse turns it off. To protect the power supply for the same reason a diode in forward direction is added to the beginning of each loop. The diode is dimensioned up to 10 kV, so it is protected very well against high voltage.

In order to maximize safety, zones are able to transmit and then receive random signal or a specific code in certain time. It is another way how to improve safety level. This task is done by the microprocessor. Output of each loop is connected to the optocoupler which is used for voltage shifting from 5 V to 12 V and also for microcontroller protection against overvoltage.

All of the common modes can be applied to these loops. Each of them needs special composition of used resistors. In practice, the most used modes are EOL and ATZ. These wirings are more described in following pages. These modes are:

- EOL
- DEOL
- ATZ

6.4.1 End of Line zone Wiring

Identification of the connected detector which has been activated is not required in this mode. That is the reason why parallel resistors, which are connected to contact of the detector, have the same values.

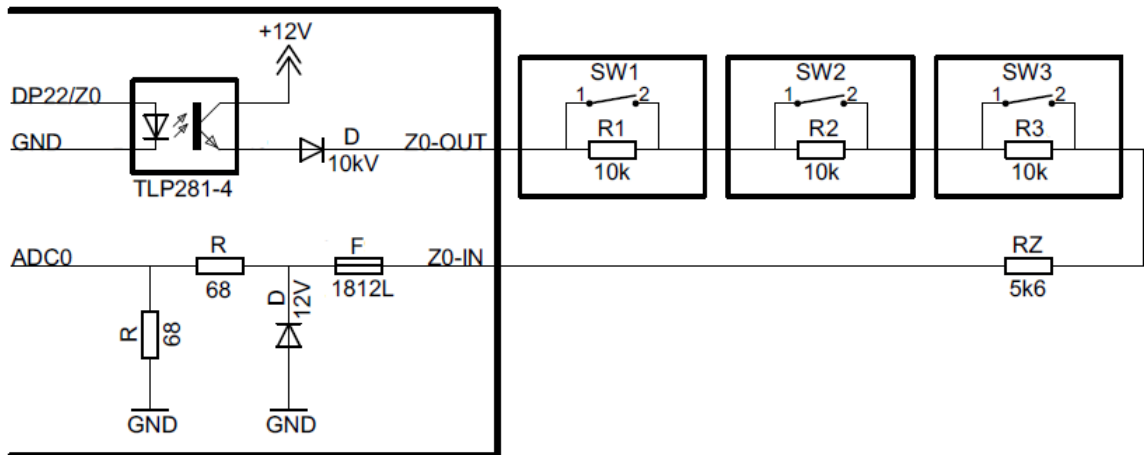


Figure 33: End of Line zone wiring

Only three states of the zone exist. In serenity state the value of all connected resistors is precisely 5.6 kΩ. This resistance causes the measured voltage U to be 3.89 V. When a short circuit occurs, no resistors are connected to the loop, and input voltage reaches the maximum up to 4.87 V. When one of the connected resistors becomes active, input voltage falls under 2.87 V. In case of sabotage by Wire Break no voltage can be measured on the input wires.

State	U [V]	R [kΩ]
Short circuit	4.87	0
Serenity	3.89	5.6
Active	<2.87	> 10
Sabotage	0	inf.

Table 15: End of Line loop values

6.4.2 Advanced Technology Zone Wiring

This mode allows distinguishing which detector in the zone has been triggered. It is based only on the different values of used resistors. Only the value of the end of line resistor is the same as in the previously described mode. Others resistors are different in order to better differentiate each voltage level. All levels are shifted approximately by 1 V.

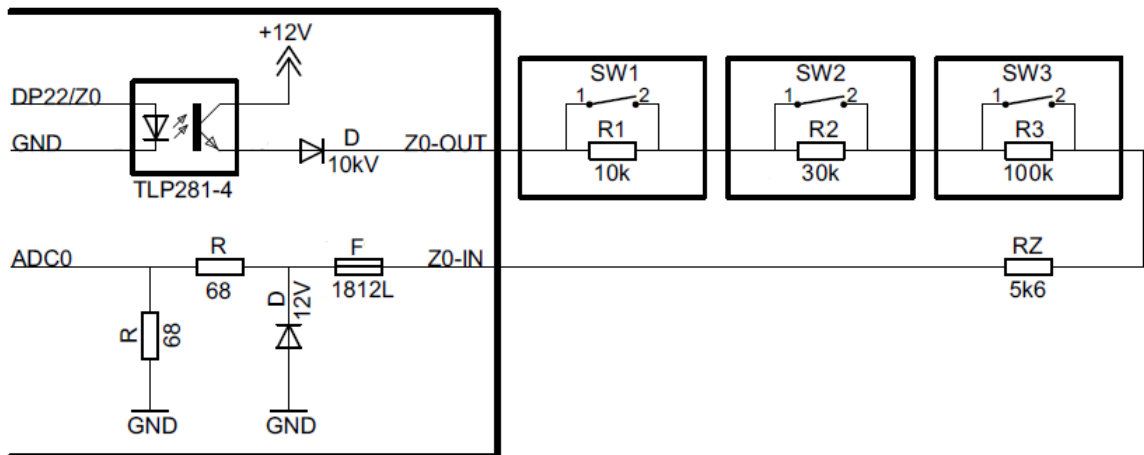


Figure 34: Advanced Technology Zone wiring

The loop can be divided only into three independent zones according to voltage level shifting. Different resistor values affect the input voltage. For proper function, these resistors are 10 kΩ, 30 kΩ and 100 kΩ. Evaluation can be done by program and it is listed in chapter **8.3 Zone Program** (page 75).

State	U [V]	R [kΩ]
Short circuit	4.87	0
Serenity	3.89	5.6
Active A	2.87	10
Active B	1.88	30
Active C	0.85	100
Sabotage	0	inf.

Table 16: Advanced Technology Zone values

This mode has the most difficult detection of possible states. Each entering voltage is converted using ADC to numbers between 0 - 1023. All these conversions are evaluated using the table listed in **Appendix I: Limit Values for Zones** (page 86). The evaluating program is listed in chapter **8.3 Zone Program** (page 75).

Power for the connected detector can be delivered directly from the Uninterrupted Power Supply, which is described in chapter **6.1 Uninterruptible Power Supply** (page 43). All loop's input are also prepared not only for the detectors, but there can be also connected continuous signal. But this kind of entering signal is not implemented into final program, because signal can be received in many kinds.

The final Zone Board has very compact dimensions - 108 x 60 mm. The photo of created Zone Board can be found in **Appendix VII: Zone Board** (page 91). The following figure shows labels for the Zone Board.

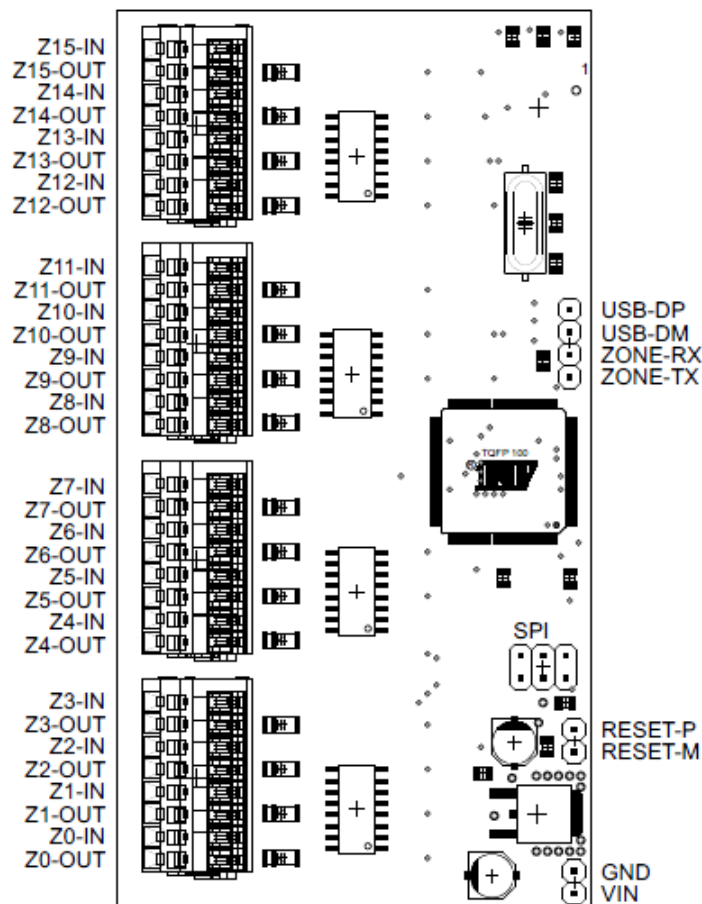


Figure 35: Zone Board

6.5 Output Board

Each CIE may optionally have some outputs called Programmable (PGM). These outputs are mostly used for turning on lights or other devices. It means that there must be a device which allows to control high voltage (230 V) by low voltage (5 V). Usually relay is used.

Like in other boards, also here it was decided to use *ATmega2560* as the command microcontroller. The pins of the microcontroller are connected directly into the relays. Each relay has two possibilities, how it can be connected. In this case, both possibilities are used. Four relays are connected as “closing” and two are connected as “switching”.

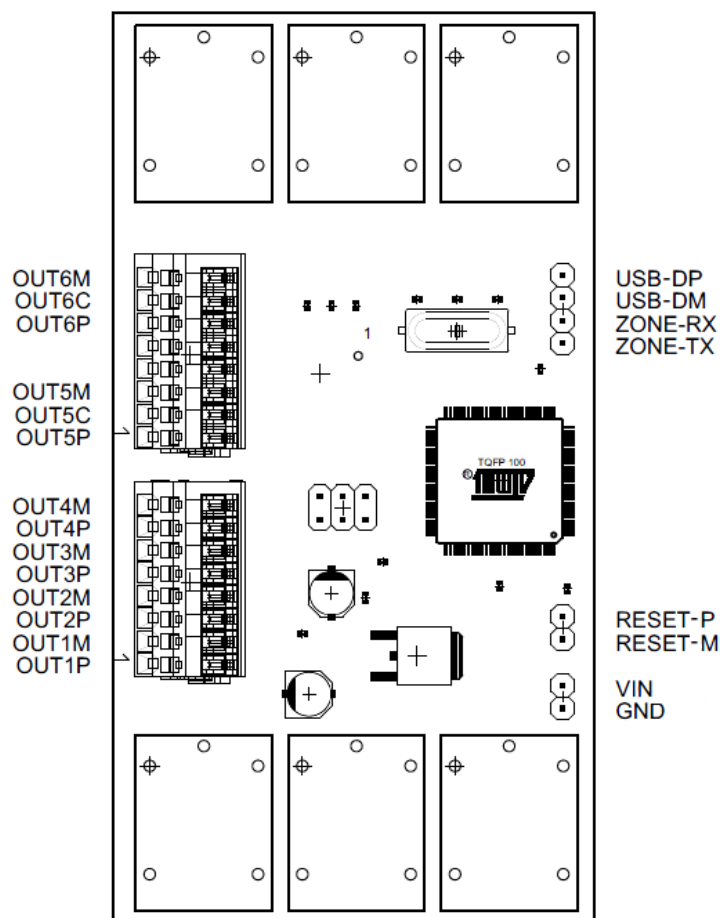


Figure 36: Output board

The previous figure shows the outputs of the board. Each closing port has Plus (P) and Minus (M) polarity. Switching ports has added Common (C) pin. Connection between microcontroller and relays can be found in **Appendix V: Output Board pins** (page 90).

7 KEYPAD

The keypad is the main device which can be used to arm or unarm the alarm system. The components should be embedded into a small case, which can be placed, for example on the wall. System uses metallic wiring and it communicates with the Main Board via the RS-232 protocol. UART communication guarantees reliable function only up to a few meters but the connected Keypad can be on the other side of the house. RS-232 can provide maximum length of the cable up to 100 m. More information about this protocol can be found in chapter **4.6.2 RS-232** (page 37).

The Keypad uses the following main components:

- Thin Film Transistor Liquid Crystal Display
- MAX232

The command microcontroller for the used TFT LCD is Atmel *SAM3X8E ARM Cortex-M3*. This microcontroller has been selected for several reasons. First is the operating voltage. The LCD model *JKL-TPM3221* operates at 3.3 V and the mentioned microcontroller operates at this voltage, too. So no other voltage converters are needed between other microcontroller.

The second reason is computing power. The *ATmega2560* was also tested with the voltage convertor, but the computing demands for the LCD were much bigger than the power of the tested microcontroller. This resulted in very slow rendering of content on the screen. This was caused by the clock speed of the microcontroller. *SAM3X8E ARM Cortex-M3* has clock speed of 84 MHz.

The last reason is that Atmel *SAM3X8E ARM Cortex-M3* is also used by Atmel in the *Atmel Due* board. All settings are already implemented in the Arduino IDE.

7.1 Liquid Crystal Display

The used LCD is called *TFT01_3.2WD*. It is made by the ElecFreaks company. The screen itself is controlled by *HX8352-A*. This model has a touch screen which is more comfortable than a basic keyboard with buttons. The touch screen controller is called *XPT2046*. LCD communicates with *SAM3X8E ARM Cortex-M3* via an 18-bit bus. The LCD module has one pinhead of 40 pins. Description of pins is listed in the following figure.

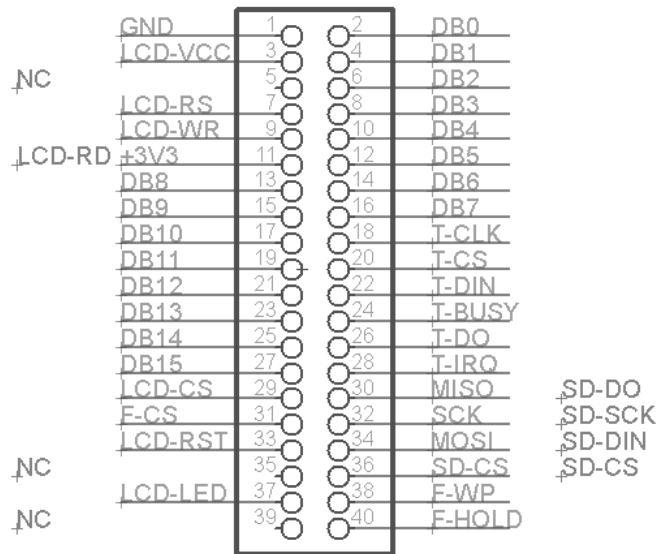


Figure 37: LCD pin description

All commands and communication is handled by libraries called *UTFT.lib* and *Utouch.lib*. There is also a card slot on the back side of the module, which can be used as storage for larger images.

The *Keypad Board* as a command module for the LCD has the same dimensions as the LCD module. Any other LCD from the ElecFreaks company up to model *TFT01_7.0* can be connected to the board. The LCD models differ only in their operating voltage, which can be changed using the built-in pinhead.

This device cannot be programmed remotely. The USB connector for connecting to the computer is placed on the board. Another connector called *DE-9* for serial communication is placed next to the USB connector. The connector ensures communication with the main board. The design of the keypad is shown in the following figure.

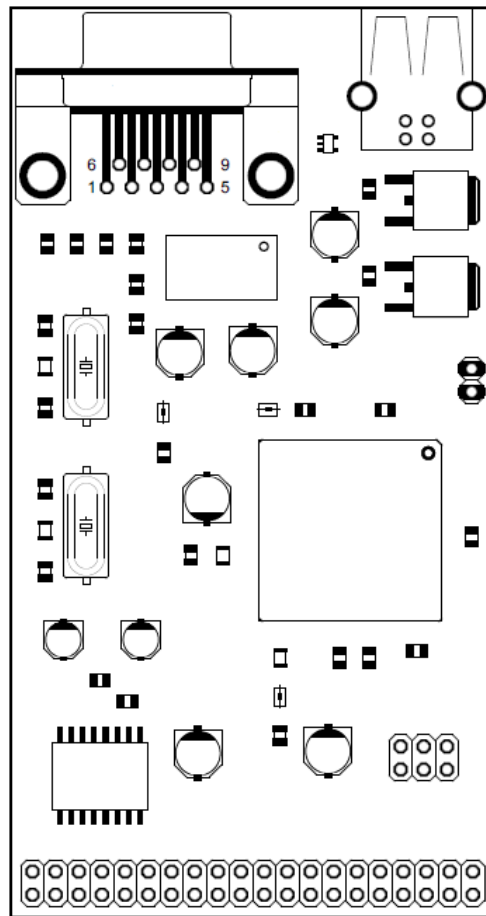


Figure 38: Keypad Board

The designed Keypad board was not created in physical form. But the design can be found in **Appendix VIII: CD with thesis, source code and EAGLE files** (page 92). The final element should be placed in a box to protect the Keypad against bad conditions.

8 SOFTWARE

The created source code is listed in this chapter, which can be used by the programmer as foundation. Every part of the created source code is described in detail. Complete programs can be found in **Appendix VI: CD with thesis, source code and EAGLE files** (page 92).

8.1 Main Program

The main program which runs on an independent microcontroller only provides common storage of needed information. According to created functions, program can evaluate information, which can lead to alarm initiation.

The main program should be able to receive data, which are sent from the device and save them to the relevant variable. All used variables are in form of text or integer array. The main variables which should be transferred between devices are listed in following figure.

```
char *users[] = {"admin", "Pavel", NULL, NULL, NULL};
char *passwords[] = {"admin", "54321", NULL, NULL, NULL};
char *ZoneName[] = { "Kitchen", "Bedroom", "Garage", "", "", "" };
char *ZoneMode[] = { "NC", "EOL", "ATZ", "NC", "NC", "NC" };
int ZoneStatus[16][6];
bool ZoneActive[16];
```

Figure 39: Main variables

SoftwareSerial must be set to use the proper pins for each device. The numbers of the pins used are mentioned in **Table 13: Software Serial interface for the main board** (page 45). The *SoftwareSerial* interface can be found in the following figure. Each device must create the mentioned interface correctly. Addresses of individual devices are fitting only with Main Board.

```
SoftwareSerial KeypadSerial(13, 6); // RX, TX
SoftwareSerial ZoneSerial(11, 7); // RX, TX
SoftwareSerial GSMSerial(10, 8); // RX, TX
SoftwareSerial ETHSerial(A8, 3); // RX, TX
```

Figure 40: Created SoftwareSerial interface

8.1.1 Communication

The created communication can be used in every connected device. Communication consists of a transmitting and receiving part. This concept is created for transmitting and receiving array with length of 16 elements. Every array consists of information, which can be evaluated.

Transmitting part

To send data, the program needs only the current array of data and the terminating text. Transmitting itself is controlled by the *SerialSoftware* function. The function automatically sends the current data. To send the whole array, it must be transmitted element by element. This is ensured by a loop, which transmits all the elements of the chosen array.

To make it easier, on the receiving part, a special character ‘%’ is placed between characters to separate them. When the array is sent, the terminating sequence must also be transmitted. The sequence consists of the starting and ending special character. The name of the transmitted array is placed between the starting character ‘/’ and the ending character ‘#’.

```
for (i = 0; i < 16; i++) {  
    mySerial.print( ZoneName[i]);  
    mySerial.print("%");  
}  
mySerial.println("/ZoneName#");
```

Figure 41: Transmitting function

The previous figure shows the basic elements, but a more sophisticated function can also be used.

Receiving part

The receiving part is more complicated than the transmitting part. The function must be able to accept the received array, split the text to data and name of the array, and then put the results into the relevant array. Every receiving pin can handle interrupts. The program does not have to continuously check the inputs. This is all done by the Arduino library.

The following figure shows the code which splits the incoming array into data and name of the array. The code is abbreviated for better understanding.

Incoming text consists of special characters. These characters are used for separation. Text is written into a temporary variable called *data[]* until the “/” character occurs. Afterwards the text is written into another temporary array called *dest[]*. The terminating character is then removed from each array.

```

c = mySerial.read();
if (c == '/') {
    data[i - 1] = '\0';
}
if (c == '#') {
    dest[i - 1] = '\0';
}

```

Figure 42: Splitting received text

Terminating characters should not occur in the data or in names of arrays. Using terminating characters in data could cause a crash of the function.

Using the *strcmp* function the program can obtain the name of the received array. The program can then separate individual elements of data using the *strtok* function and the special character ‘%’. After that, every element is placed in proper position in the current array. The described code is listed in following figure.

```

if (strcmp (dest, "/ZoneMode") == 0) {
    ZoneMode[0] = '\0';
    int z = 0;
    char * pch;
    pch = strtok (data, "%");
    while (pch != NULL)
    {
        //Serial.print(pch);
        ZoneMode[z] = pch;
        pch = strtok (NULL, "%");
        z++;
    }
}

```

Figure 43: Extracting data into the proper array

Before saving new values to the proper array, the current array must first be erased. Otherwise the function could crash.

8.1.2 GSM Program

The GSM program is used mainly by the Main program to send or receive messages or voice calls. The current program is only for basic explanation of communication with the *SIM900* module. The *SoftwareSerial* must be declared to communicate with the module. After declaration, baud rate must be set.

```
SoftwareSerial gprsSerial(10, 8);
//SoftwareSerial mySerial (RX,TX);
gprsSerial.begin(19200); // GPRS shield baud rate
```

Figure 44: Initialization of GSM program

When the program wants to send a text message, it can use a function called *SendTextMessage*. This function consists of the needed request name, phone number, the text itself and the terminating character. All these requirements are in the following figure.

```
void SendTextMessage() {
  gprsSerial.print("AT+CMGF=1\r"); // Set the shield to SMS mode
  gprsSerial.println("AT+CMGS = \"+420702119294\""); // the phone number
  gprsSerial.println("How are you today?"); //the content of the message
  gprsSerial.print((char)26); //the ASCII code of the ctrl+z is 26
  gprsSerial.println();
}
```

Figure 45: Function SendTextMessage

According to the datasheet, there must be a 100 ms delay after each command to ensure flawless processing. Content of the message can contain placeholders which can consist of dynamic information. Making a voice call is very similar. Example of voice call is in the following figure.

```
void DialVoiceCall() {
  gprsSerial.println("ATD+420702119294;");//number must include country code
  gprsSerial.println();
}
```

Figure 46: Function DialVoiceCall

Communication is based only on AT commands, so no special library is needed. It is very easy to add email communication to the program.

8.2 Ethernet Program

This program provides a user interface via web pages. Device *W5100* from chapter **6.2.3 Ethernet Microcontroller with W5100** (page 47) needs a special library which is called *Ethernet.h* for communication. This library was made by the Arduino company. The following code represents only the main block of the program. The whole code can be found in **Appendix VIII: CD with thesis, source code and EAGLE files** (page 92).

The *IP* address and *MAC* address must be set to establish communication with the connected router. User must set the IP address correctly depending on the used network. After that two functions are called.

```
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
IPAddress ip(192, 168, 10, 200);
Ethernet.begin(mac, ip);
server.begin();
```

Figure 47: Setup of Ethernet program

In the Main loop, the program waits for a client to connect. When a client connects, the program automatically reads the incoming content and stores it. The incoming content is only a standard header from the web browser, which consists of requirements for the program. Different browsers use different headers. Content processing is described using Mozilla Firefox 44.0.2 header. Accomodating other browsers is very easy. Headers from Mozilla Firefox are shown in the following figure.

```
GET / HTTP/1.1
Host: 192.168.10.200
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:44.0) Gecko/20100101 Firefox/44.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: cs,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
```

Figure 48: Header using Mozilla Firefox

The actual content can be very variable, so the processing must be dynamic. The library itself cannot identify the end of headers. This must be done by the programmer. When it is forgotten, the program will always crash.

Processing is based on counting and recognizing a blank line. The header ends with a blank line. A blank line is represented by two sequences of special characters `\r` and `\n`. Detection of a blank line is shown in the following figure.

```
if (c == '\n')
    currentLineIsBlank = true;
else if (c != '\r')
    currentLineIsBlank = false;
```

Figure 49: Detection of a blank line

All requests from the client are sent using the *POST* method. That means that the header only contains information about the requested page and the body can contain form parameters. Only the first request from the client called *index* is always sent using the *GET* method. So there must be different processing for each method.

Special request for the *index* page has always the same form. It is shown in the previous **Figure 48: Header using Mozilla Firefox** (page 67). The header has precisely eight lines. The following figure shows processing of the *index* page.

```
if (c == '\n' && currentLineIsBlank && x == 8) { // for index (login)
    client.println(login);
    break;
}
```

Figure 50: Request for the index page

When the previous condition is met, the program can send the *login* page to the client. Current character must be `\n` and current line must be blank and current line must be the eighth. Further processing of the main page is described in chapter **8.2.1 Check Login** (page 70).

A more complicated situation comes with the *POST* request. This request could contain form parameters below the header. This information is found on the twelfth line. This line is placed in a separate array. In this case, the array is called *char data[100]*.

```
if (x == 12) { // on last (12th) line read all
    data[i] = c;
    i++;
}
if (x == 12 && c == 'X') { // when X occurs add to last place terminating character
    data[i - 1] = '\0';
```

Figure 51: Data array

Only the characters from the twelfth line are written to the data array. But this string does not have any terminating character. So a special terminating character must be added to all requests. In this case, character *X* was chosen. This character can be found also in the condition in previous figure.

The header also consists of the requested page name. This information is located on the first line. The name of the requested page is very important. Saving the name of requested page is listed in the following figure. The name itself is located between characters “/” and “space”.

```

if (x == 0) { // condition for page array
  if (writetopage) {
    page[i] = c;
    i++;
  }
  if (c == '/' && z == 0) {
    writetopage = true;
    z = 1;
  }
  if (c == ' ')
    writetopage = false;
}

```

Figure 52: Page array

The library *strings.h*, which is used to compare of two strings, is already implemented in the Arduino main library. The page array is compared with all possible page names which can be requested by the client. Some of the created pages are listed below.

```

if (strcmp (page, "checklogin") == 0) // from index action checklogin
  client.println(login);
if (strcmp (page, "adduser") == 0) { // from index action checklogin
if (strcmp (page, "init") == 0) // from index action checklogin
  client.println(initialization); //temp
if (strcmp (page, "users") == 0) { // from index action checklogin
  sprintf(buffer, userspage, users[0], users[1] , users[2] , users[3] );
  client.println(buffer);
if (strcmp (page, "state") == 0) // from index action checklogin
  client.println(state); // temp
break;

```

Figure 53: Some of the created pages

8.2.1 Check Login

When the client types the IP address of the web server, the server sends the login page. This page consists only of a form to enter the user name and password. The code of login page is written in HTML code, which is loaded as a *char* array in the program. The server response containing the login page is in the following figure.

```
"HTTP/1.1 200 OK\n"
"Content-Type: text/html\n"
"Connection: close\n"
"\n"
"<!DOCTYPE HTML>\n"
"<html>\n"
"<title>Log in</title>\n"
"<h3>LOGIN</h3>\n"
"<form method='post' action='checklogin'>\n"
"User name: <br><input type='text' name='username' value='admin' /><br>\n"
"Password: <br><input type='password' name='password' value='admin' /><br>\n"
"<input type='hidden' name='X' value='' />\n"
"<input type='submit' value='Log In' />\n"
"</form>\n"
"</html>\n";
```

Figure 54: Login code in html

Every client must log in to the system using their user name and the corresponding password. The *POST* method sends user name and password in the following form. This string is stored in the data array.

```
username=*username*&password=*password*&X
```

Figure 55: Login data array

The username and password are placed into a string with separating characters. These characters are “=” and “&”. Before any comparison, the string must be split into arrays without any separating characters. This is done by the string function called *strtok*. The received query string is divided into several tokens, which contain the needed characters. Separation using *strtok* is listed below.

```
pch = strtok (data, "=");
while (pch != NULL) {
    tempbuffer[i] = pch;
    i++;
    pch = strtok (NULL, "&");
}
```

Figure 56: Strtok function

The *strtok* function splits the query string into its individual parts according to the separating character. Each fragment is then placed into a temporary array called *tempbuffer[]*. After that, the entered user name and password can be compared with the database of users. A very basic database is created as an array.

```
char *users[] = {"admin", NULL, NULL, NULL, NULL};
char *passwords[] = {"admin", NULL, NULL, NULL, NULL};
```

Figure 57: Database of users

By default, only the user name “admin” with the password “admin” can enter. The admin’s password can be changed. The actual comparison is listed in the following figure.

```
for (int i = 0; i < 5; i++)
    if ((strcmp(tempbuffer[1],users[i])==0) && (strcmp(tempbuffer[3],passwords[i])==0))
        currentuser = users[i];
return (currentuser);
```

Figure 58: User comparison with database

User name is placed in the *tempbuffer[1]* variable and password in *tempbuffer[3]*. The program checks all items in the database. When it finds with a matching user name and password, the function returns the name of the current user. When the information does not have a match in the database, the program returns *NULL*.

```
if (strcmp (page, "checklogin") == 0) // from index action checklogin
    if (strcmp (checklogin(data, len), NULL) == 0)
        client.println(login);
    else {
        sprintf(buffer, mainpage, currentuser);
        client.println(buffer);
    }
```

Figure 59: Check login function

The previous figure shows the authorization process. When the client is authorized, it means that the user name and password were found in the database and the program sends the client to *mainpage*. When the login information is not found, the program sends the login page again to repeat the login.

8.2.2 Add User

The next function is called *adduser*. An authorized client can manage the list of users. The users bookmark lists all users who can enter the system. This page called users comes with raw text in HTML where names of users from the database can be placed. The raw code is following.

```
<!DOCTYPE HTML>\n"
<html>\n"
<title>Users</title>\n"
<h3>Users</h3><br>\n"
1. %s <br>\n"
2. %s <br>\n"
3. %s <br>\n"
4. %s <br><br>\n"
Add New User:<br>\n"
<form method='post' action='adduser'>\n"
User name: <input type='text' name='username' value='' />\n"
Password: <input type='password' name='password' value='' />\n"
<input type='hidden' name='X' value='' />\n"
<input type='submit' value=' Add User '/>\n"
</form>\n"
</html>\n";
```

Figure 60: Users page code in html

The raw code contains placeholders *%s*, where actual text will be put later. In this case it will be user names. Before sending this html code to the client, it must first be filled with the needed names. The *sprintf* function is used to connect two strings into one. After that, the final string is sent to the client. The function is listed in the following figure.

```
if (strcmp (page, "users") == 0) { // from index action checklogin
    sprintf(buffer, userspage, users[0], users[1] , users[2] , users[3] );
    client.println(buffer);
}
```

Figure 61: Connecting two strings

A new user can be added very easily. On the page, there are two text boxes, where the user can type a new user with a password. This information will be automatically saved in the database. But the program must first check whether the new user doesn't already exist. The check is listed in the following figure.

```

for (int i = 0; i < 5; i++)
    if ((strcmp(tempbuffer[1], users[i]) == 0 ))
        return ("Exist");

```

Figure 62: Check existing user

If the new user already exists, the program returns the value “Exist”, and the function is terminated. If the new user does not exist, the program continues further. It must find an empty place in the database using the following code.

```

int pos = 0;
for (int i = 0; i < 5; i++)
    if (users[pos] != NULL)
        pos++;

```

Figure 63: Finding empty place in database

A new variable *pos* contains the position of the new user in the database. When the previous loop loads *NULL*, it automatically goes to the next position and also increments the *pos* value. After that, the new user with his password is saved into the database using the *strcpy* function, which is listed in the following figure.

```

strcpy(users[pos], tempbuffer[1]);
strcpy(passwords[pos], tempbuffer[3]);
return ("Saved");

```

Figure 64: Saving new user and password

8.2.3 Set Zone

User can manage options of loops via the Set Zone page. The page contains the individual loops. Via this page the name of the current loop and its mode can be set. Each loop has a button “Set Zone” which sends data to the Main Board. For better understanding, options were created for two loops. HTML code is listed in the following figure.

More loops can be added to the page very easily. Code from `<form>` to `</form>` can be copied. Only the loop number must be corrected to the current loop. Place for the loop name can be created as using a text field. Mode of the loop can be picked from a listbox using the select element in HTML.

```

"HTTP/1.1 200 OK\n"
"Content-Type: text/html\n"
"Connection: close\n"
"\n"
"<!DOCTYPE HTML>\n"
"<html>\n"
"<title>Initialization</title>\n"
"<h3>Initialization</h3>\n"
"<form method='post' action='demo' id='zone' >\n"
"Zone 1. Choose mane <input type='text' name='name' value='Kitchen'>\n"
"Select Mode <select name='mode' form='zone'>\n"
"<option value='NC'>NC</option>\n"
"<option value='EOL'>EOL</option>\n"
"<option value='DEOL'>DEOL</option>\n"
"<option value='ATZ'>ATZ</option>\n"
"</select>\n"
"<input type='hidden' name='X' value=''>\n"
"<input type='submit' value='Set zone'>\n"
"</form>\n"
"</html>\n"

```

Figure 65: Initialization page in html

The following figure shows the processing of initialization. When the “Set Zone” button is triggered, the *InitZone* function is executed. *InitZone* contains the same code as the previous *AddUser* function. The function is only adjusted for current initialization. When *InitZone* is done, the server sends the same page to the user again with the saved data.

```

if (strcmp (page, "init") == 0) // from main to init
    client.println(initialization); //temp
if (strcmp (page, "demo") == 0) { // from init action
    InitZone(data, len);
    client.println(initialization);
}

```

Figure 66: Processing of Initialization

The *InitZone* function expects only two arguments (*ZoneName* and *ZoneMode*). More arguments can cause unexpected results. While sending more than two arguments, the *InitZone* function must be adjusted to more arguments.

8.3 Zone Program

The Zone program controls all the connected devices. There are several arrays to store all the needed information. The first array called *ZoneNamePin[16]* contains the physical pin on the board. *ZoneVoltageValue[16]* stores the value of the measured voltage. *ZoneMode[16]* contains the actual state, and *ZoneActive[16]* only contains information if the current zone is active. Declaration of these variables is in following figure.

```
int ZoneNamePin[] = {A0, A1}; // name of physical pin on board
int ZoneVoltageValue[16]; // voltage level after ADC (0-1023)
int ZoneMode[16]; // NCL EOL ATZ
bool ZoneActive[16]; // status 1=active 0=off
```

Figure 67: Declaration of main variables

The main function of this program is to measure the value of input voltage. Measuring is done using a special function called *analogRead()*. Argument of this function is the number of the physical pin stored in *ZoneNamePin[]*. Measuring must be done for each zone. The cycle is in the following code.

```
for (x = 0; x < 15; x++) {
  ZoneVoltageValue[x] = analogRead(ZoneNamePin[x]); // read voltage value
  if (ZoneActive[x] != 1) // skip inactive zones
    continue;
```

Figure 68: Measuring entering voltage

Inactive zones are skipped. Each zone can have a different mode, so the next step is to set threshold values according to needed states. Only four modes can occur as described in chapter **6.4 Zone Board** (page 54). A switch keyword was picked to ensure that different modes are used.

```
switch (ZoneMode[x]) { // switch between zone modes
  case NCL:
    if (ZoneVoltageValue[x] > 972) // Serenity
      UpdateStatus(x, SER);
    else // Active, ShortCircuit, Sabotage
      UpdateStatus(x, AC1);
```

Figure 69: Switching between modes

Measured voltage is a value from 0 to 1023 in binary form. Threshold values for each mode are listed in **Appendix I: Limit Values for Zones** (page 86). A status is assigned to each threshold. When the program assigns a status, another function *UpdateStatus()* with the obtained status is called. The following figure shows the structure of EOL thresholds.

```

case EOL:
    if (ZoneVoltageValue[x] < 852 && ZoneVoltageValue[x] > 768) // Serenity
        UpdateStatus(x, SER);
    else if (ZoneVoltageValue[x] < 639 && ZoneVoltageValue[x] > 556) // Active-1
        UpdateStatus(x, AC1);
    else if (ZoneVoltageValue[x] > 972) // ShortCircuit
        UpdateStatus(x, SHC);
    else // Sabotage
        UpdateStatus(x, SAB);

```

Figure 70: Threshold values for EOL

The *UpdateStatus()* function manages the status of all zones. The function first checks if the current status was not already sent. When the status was already sent, the function ends. If not, the function clears the *ZoneStatusSend[16][6]*, which contains the zone number and status. Then the array of the current zone is filled with the obtained status.

```

void UpdateStatus(int ZoneNumber, int State) {
    if (ZoneStatusSend[ZoneNumber][State] == 1); // is status already send
    else {
        for (int z = 0; z < 6; z++)
            ZoneStatusSend[ZoneNumber][z] = 0; // reset all data in ZoneStatusSend
        ZoneStatusSend[ZoneNumber][State] = 1; // set current status
        UpdateMainBoard();
    }
}

```

Figure 71: Update status function

Finally, the updated array is transmitted to the main board. This program can be modified depending on the requirements of the programmer.

CONCLUSION

The main goal of this thesis was to create a fully working Control and Indicating Equipment (CIE) using the same components as other commercial CIEs available on the market. The theoretical part describes the requirements, legislation and the main function and features of Intruder Alarm Systems. The analytical part describes all the designed components. For every component, a program was created which is described in detail in the Software chapter.

The components which were completely finished are the Main board with an Ethernet interface and the Zone board. Other components were only designed, but not constructed. The final system consists of the Main board with an Ethernet interface, the Zone board, the GSM module and power management unit with UPS function and battery. The system is placed into a solid aluminium box. All components can be programmed via USB connection.

The Zone board provides connection to 16 zones. Each zone can handle common modes like NO, EOL, DEOL or ATZ. Any real detector can be connected to the Zone board. The Zone board consists of a microcontroller which handles the connected detectors and communication with the Main board.

The Main board is the core component of the system. It is responsible for internal and external communication. Used Ethernet microcontroller provides access to the system via the created web administration interface. The GSM module is also connected to the main board.

The final system can be used as teaching equipment for students of security technology. It cannot be used as a real Intruder Alarm System because it is not certified.

All data and programs for the created components are available on the enclosed CD.

BIBLIOGRAPHY

- [1] MAZIDI, Muhammad Ali, Sarmad NAIMI and Sepehr NAIMI. The AVR microcontroller and embedded systems: using Assembly and C. Upper Saddle River, N.J.: Prentice Hall, 2011, xiv, 776 p. ISBN 01-380-0331-9.
- [2] ATMEL CORPORATION. ATmega640/1280/1281/2560/2561: Atmel 8-bit AVR MCU FLASH Microcontroller, ATmega32A [pdf]. 2014, 435 p. [cit. 2015-04-22]. Available online from: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf
- [3] ARDUINO CORPORATION. Arduino [online]. 2015 [cit. 2015-04-22]. Available online from: <http://www.arduino.cc/>
- [4] GLOCKER, Tobias. Computer Architectures: Lecture material [pdf]. University of Vaasa, 2014 [cit. 2015-17-04]. Available online from: <http://teg.uwasa.fi/courses/tlte2100/index.php?content=lectures>
- [5] WIZNET CO. W5100 Datasheet [online]. 2011 [cit. 2016-03-03]. Available online from: http://www.wiznet.co.kr/wp-content/uploads/wiznethome/Chip/W5100/Document/W5100_Datasheet_v1.2.6.pdf
- [6] VALOUCH, Jan. Projektování integrovaných systémů. První vydání. Zlín: Univerzita Tomáše Bati ve Zlíně, 2013. ISBN 978-80-7454-296-1.
- [7] HANÁČEK, Adam. Security methods of wired central ESS against sabotage. Zlín, 2010. Bakalářská práce. Univerzita Tomáše Bati ve Zlíně.
- [8] KRAHULÍK, Lukáš. Alarm Security and Emergency Systems and Designing of their Functionality. Zlín, 2012. Diplomová práce. Univerzita Tomáše Bati ve Zlíně.
- [9] STRAŠIL, Ivo. Module Of Electronic Security Central With Ethernet Communicator. Brno, 2010. Diplomová práce. Vysoké Učení Technické v Brně.
- [10] Geeetech SIMCOM SIM900 Module. AliExpress [online]. 2015 [cit. 2016-03-01]. Available online from: http://www.aliexpress.com/store/product/Details-about-SIM900-Quad-band-Wireless-GSM-GPRS-Shield-Development-Board-for-Iduino-arduino/1149129_1664058541.html
- [11] Arduino GPRS Shield. GeeTech Wiki [online]. 2014 [cit. 2016-03-05]. Available online from: http://www.geeetech.com/wiki/index.php/Arduino_GPRS_Shield

- [12] UPS zdroj 12V 35W s nabíjením. LEDTech.cz [online].2013 [cit. 2016-03-14]. Available online from: <http://www.led-tech.cz/-napajeci-zdroje-12vinterierove/50781-ups-zdroj-12v-35w-s-nabijenim.html>
- [13] ČSN EN 50131-1 - Poplachové systémy - Poplachové zabezpečovací a tísňové systémy. ed. 2. Praha: Český normalizační institut, 2007.
- [14] Atmel Corporation - Microcontrollers [online]. California, United States: Atmel corp., 2016 [cit. 2016-04-16]. Available online from: <http://www.atmel.com/default.aspx>

LIST OF ABBREVIATIONS

AC	Alternating Current
ADC	Analog to Digital Converter
AMC	Alarm Monitoring Center
ARC	Alarm Receiving Center
ARM	Advanced RISC Machine
ATZ	Advanced Technology Zone
CIE	Control and Indicating Equipment
COM	Common
DC	Direct Current
DEOL	Double End Of Line
EOL	End Of Line
GND	Ground
GPRS	General Packet Radio Service
GSM	Global System for Mobile
HTML	HyperText Markup Language
IC	Integrated Circuit
IC	Integrated Circuit
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
LED	Light Emitting Diode
NC	Normally Closed
NC	Not Connected
NO	Normal Open
NO	Normally Open
RST	Reset
SAM	SMART Atmel Microcontroller
SCS	SPI Cable Select
SEN	SPI Enable
SIM	Subscriber Identity Module
SMD	Surface Mount Device
SMS	Short Message Service
TFT	Thin Film Transistor
UART	Universal Asynchronous Receiver and Transmitter
UPS	Uninterruptible Power Supply
USB	Universal Serial Bus
USB-DM	Universal Serial Bus-Data Plus
USB-DP	Universal Serial Bus-Data Minus

LIST OF FIGURES

Figure 1: Normally closed loop[7]	14
Figure 2: Graphical states of the normally closed loop	14
Figure 3: End of line loop [7]	15
Figure 4: Graphical states of the End of line loop	15
Figure 5: Double end of line loop[7]	16
Figure 6: Graphical states of Double end of line loop	16
Figure 7: Advanced technology zone [7]	17
Figure 8: Graphical states of the advanced technology zone	17
Figure 9: Type 1 configuration [6]	21
Figure 10: Type 2 configuration [6]	22
Figure 11: 32-bit ARM microcontrollers [14]	31
Figure 12: Memory map of Atmel AVR processor [2]	32
Figure 13: Interrupt Service Routine [4]	33
Figure 14: Example rollover interrupt [4]	35
Figure 15: Timer using a rollover interrupt [4]	35
Figure 16: Example compare mode [4]	36
Figure 17: Timer using compare mode [4]	36
Figure 18: Universal Asynchronous Receiver and Transmitter [4]	37
Figure 19: RS-232 [4]	38
Figure 20: Principle of SPI communication	38
Figure 21: Created block diagram of the CIE	42
Figure 22: Uninterruptible power supply [12]	43
Figure 23: Main Board	44
Figure 24: FT232RL connection	46
Figure 25: Block diagram of the W5100 [5]	47
Figure 26: SPI connection of the W5100	48
Figure 27: Wiring of Ethernet connector	49
Figure 28: USB hub TUSB2046B	50
Figure 29: Power management	51
Figure 30 GSM Shield by GeeTech [10]	52
Figure 31 UART of the SIM900 [11]	53
Figure 32: Voltage divider	54

Figure 33: End of Line zone wiring.....	56
Figure 34: Advanced Technology Zone wiring	57
Figure 35: Zone Board	58
Figure 36: Output board	59
Figure 37: LCD pin description.....	61
Figure 38: Keypad Board.....	62
Figure 39: Main variables	63
Figure 40: Created SoftwareSerial interface.....	63
Figure 41: Transmitting function	64
Figure 42: Splitting received text	65
Figure 43: Extracting data into the proper array.....	65
Figure 44: Initialization of GSM program	66
Figure 45: Function SendMessage.....	66
Figure 46: Function DialVoiceCall.....	66
Figure 47: Setup of Ethernet program	67
Figure 48: Header using Mozilla Firefox.....	67
Figure 49: Detection of a blank line	68
Figure 50: Request for the index page	68
Figure 51: Data array	68
Figure 52: Page array	69
Figure 53: Some of the created pages.....	69
Figure 54: Login code in html.....	70
Figure 55: Login data array.....	70
Figure 56: Strtok function.....	70
Figure 57: Database of users.....	71
Figure 58: User comparison with database	71
Figure 59: Check login function	71
Figure 60: Users page code in html	72
Figure 61: Connecting two strings	72
Figure 62: Check existing user	73
Figure 63: Finding empty place in database	73
Figure 64: Saving new user and password	73
Figure 65: Initialization page in html	74

Figure 66: Processing of Initialization	74
Figure 67: Declaration of main variables	75
Figure 68: Measuring entering voltage	75
Figure 69: Switching between modes	75
Figure 70: Threshold values for EOL	76
Figure 71: Update status function	76

LIST OF TABLES

Table 1: Standard is EN 50131 [13]	25
Table 2: Degree of security[13]	26
Table 3: Classification of environment [13]	26
Table 4: Classification of failures [13]	27
Table 5: Detection of invalid attempts to authorization [13]	27
Table 6: Method of connection monitoring [13]	28
Table 7: Event log and memory capacity [13]	28
Table 8: Minimum duration of alternative supply [13]	29
Table 9: Maximumduration of battery charging [13]	29
Table 10: Description of used codenames [2]	30
Table 11: ARM Cortex-M variations [14]	31
Table 12: Reset and Interrupt Vectors [4]	34
Table 13: Software Serial interface for the main board	45
Table 14: LED status of W5100	49
Table 15: End of Line loop values	56
Table 16: Advanced Technology Zone values	57

APPENDICES

Appendix I: Limit Values for Zones	86
Appendix II: Used Components for the Main Board	87
Appendix III: Used Components for the Zone Board	88
Appendix IV: Zone Pins	89
Appendix V: Output Board pins addressing	90
Appendix VI: Main Board	91
Appendix VII: Zone Board	91
Appendix VIII: Created CIE	92
Appendix IX: CD with thesis, source code and EAGLE files	92

Appendix I: Limit Values for Zones

U	Value	State
[V]	-	-
5,00	1023	-
4,87	1014	Shortircuit
4,67	972	-
4,09	852	-
3,89	810	Serenity
3,69	768	-
3,07	639	-
2,87	598	Active-1
2,68	556	-
1,08	433	-
1,88	391	Active-2
1,68	350	-
1,05	218	-
0,85	177	Active-3
0,65	135	-
0,2	41	-
0	0	Sabotage

Appendix II: Used Components for the Main Board

Component / value	additional parameters	package	quantity	manufacturer
Capacitor 100nF	10% / 16V / X7R	0402	21	
Capacitor 22pF	5% / 50V / NPO	0402	8	
Capacitor 47uF	20% / 35V / RM5,5	C	4	
Capacitor 10uF	20% / 35V / RM5,5	C	1	
Resistor 15k	1 % / 62,5 mW / 100 ppm	0402	8	
Resistor 27R	1 % / 62,5 mW / 100 ppm	0402	10	
Resistor 10k	1 % / 62,5 mW / 100 ppm	0402	4	
Resistor 1k	1 % / 62,5 mW / 100 ppm	0402	6	
Resistor 1k5	1 % / 62,5 mW / 100 ppm	0402	2	
Resistor 1M	1 % / 62,5 mW / 100 ppm	0402	3	
Resistor 49R9	1 % / 62,5 mW / 100 ppm	0402	4	
Resistor 12k	1% / 62,5 mW / 100 ppm	0402	1	
Resistor 300R	1% / 62,5 mW / 100 ppm	0402	1	
ATMEGA2560-16AU		TQFP-100	2	Atmel
FT232RL		SSOP-28	2	FTDI
W5100		LQFP-80	1	WIZnet
TUSB2046B		LQFP-32	1	Texas Instruments
HR911105A	PoE-RJ45		1	HanRun
Crystal 6MHz	30 ppm	HC49USSMD	1	
Crystal 16MHz	30 ppm	HC49USSMD	2	
Crystal 25MHz	30 ppm	HC49USSMD	1	
74LVC1G14DBVR		SOT-23	1	Texas Instruments
LF33CDT	3,3V / 0,5A	DPAK	1	Thomson
LF50CDT	5,0V / 0,5A	DPAK	1	Thomson
STM812LW16		SOT-143	1	STMicroelectronics
USB-B-H	USB B / 90°		1	
BLM21AG121SN1D	-	0805	3	Murata
S1G2			2	
S2G4			1	
S2G6			2	
S2G16			1	
BL06			2	
BL08			2	

Appendix III: Used Components for the Zone Board

Component / value	additional parameters	package	quantity	manufacturer
Capacitor 100nF	10% / 16V / X7R	0402	6	-
Capacitor 22pF	5% / 50V / NPO	0402	2	-
Capacitor 47uF	20% / 35V / RM5,5	C	2	
Resistor 1k	1 % / 62,5 mW / 100 ppm	0402	3	
Resistor 1M	1 % / 62,5 mW / 100 ppm	0402	1	
Resistor 10k	1 % / 125 mW / 100 ppm	0805	16	
Resistor 12k4	1 % / 125 mW / 100 ppm	0805	16	
TLP281-4		SOP-16	4	TOSHIBA
ATMEGA2560-16AU		TQFP-100	1	Atmel
FT232RL		SSOP-28	1	FTDIChip
Crystal 16MHz	30 ppm	HC49USSMD	1	
LF50CDT	5,0V / 0,5A	DPAK	1	Thomson
1N4007	1000V / 1A	SMA	16	
SM6T12A	12V/600W	SMB	16	Vishay
PPTC1812SMD010	30V/ 0,1A	1812	16	HITANO
WAGO233-508			4	WAGO
S1G2			2	
S2G6			1	
S1G4			1	

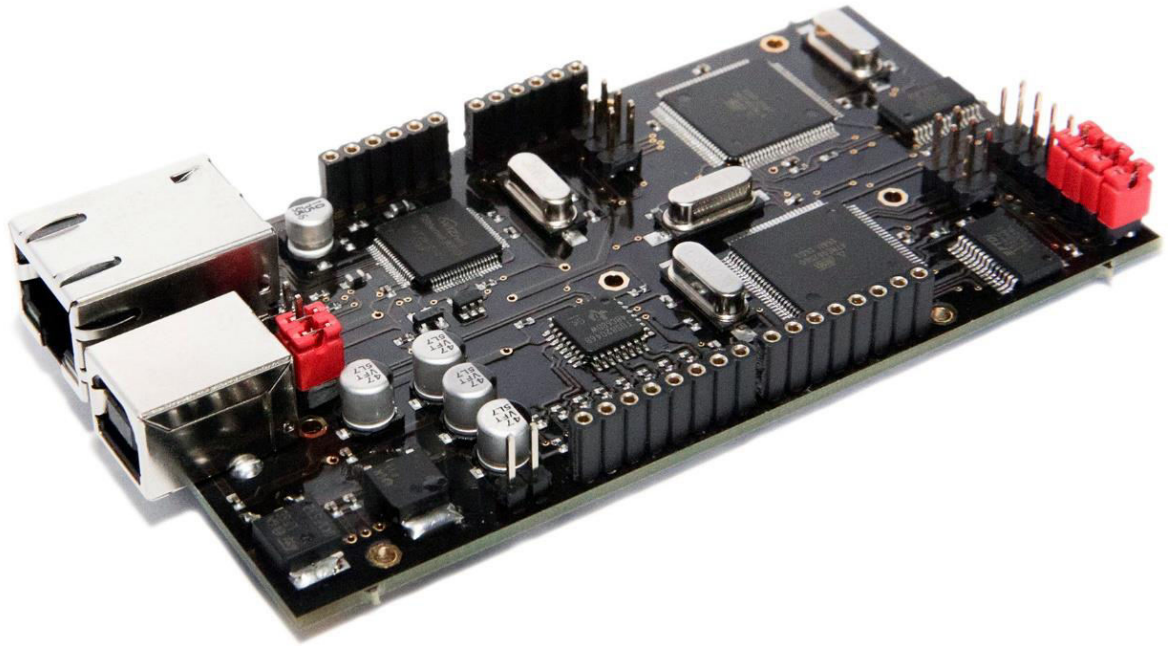
Appendix IV: Zone Pins

Z0-OUT	Digital Pin 22
Z0-IN	Analog Pin 0
Z1-OUT	Digital Pin 23
Z1-IN	Analog Pin 1
Z2-OUT	Digital Pin 24
Z2-IN	Analog Pin 2
Z3-OUT	Digital Pin 25
Z3-IN	Analog Pin 3
Z4-OUT	Digital Pin 26
Z4-IN	Analog Pin 4
Z5-OUT	Digital Pin 27
Z5-IN	Analog Pin 5
Z6-OUT	Digital Pin 28
Z6-IN	Analog Pin 6
Z7-OUT	Digital Pin 29
Z7-IN	Analog Pin 7
Z8-OUT	Digital Pin 30
Z8-IN	Analog Pin 8
Z9-OUT	Digital Pin 31
Z9-IN	Analog Pin 9
Z10-OUT	Digital Pin 32
Z10-IN	Analog Pin 10
Z11-OUT	Digital Pin 33
Z11-IN	Analog Pin 11
Z12-OUT	Digital Pin 34
Z12-IN	Analog Pin 12
Z13-OUT	Digital Pin 35
Z13-IN	Analog Pin 13
Z14-OUT	Digital Pin 36
Z14-IN	Analog Pin 14
Z15-OUT	Digital Pin 37
Z15-IN	Analog Pin 15

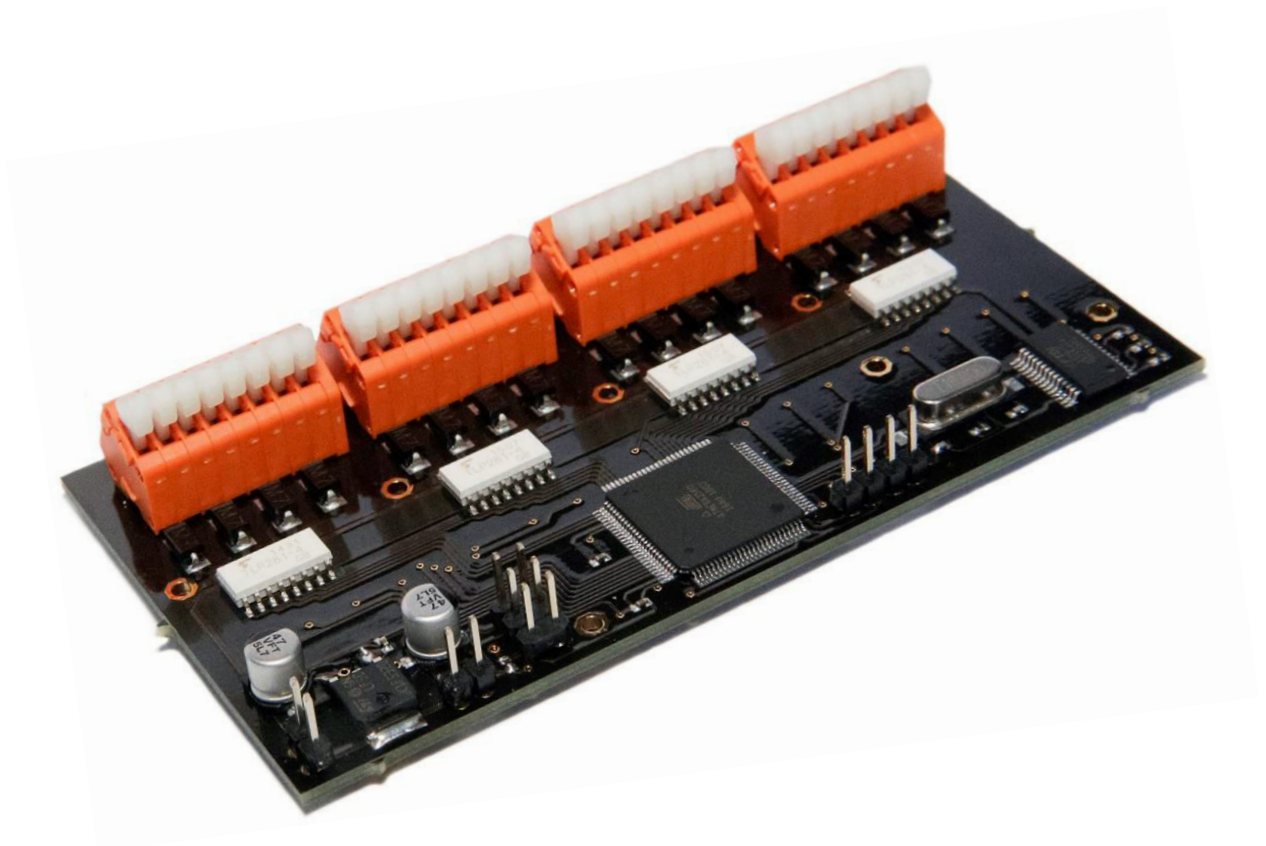
Appendix V: Output Board pins addressing

Pin Name	Control Pin	Function
OUT1P	DP22	Closing (NO)
OUT1M		
OUT2P	DP23	Closing (NO)
OUT2M		
OUT3P	DP24	Closing (NO)
OUT3M		
OUT4P	DP25	Closing (NO)
OUT4M		
OUT5P	DP26	Switching Minus (NC)
OUT5C		
OUT5M		
OUT6P	DP27	Switching Minus (NC)
OUT6C		
OUT6M		

Appendix VI: Main Board



Appendix VII: Zone Board



Appendix VIII: Created CIE

