Stogu Pavel

Smart Lock System

Technology and Communication

2015

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Degree Program in Information Technology

ABSTRACT

Author	Pavel Stogu
Title	Smart Lock System
Year	2015
Language	English
Pages	46 + 1 Appendix
Name of Supervisor	Smail Menani

The thesis was initiated by an idea to control the student's access to the I2A area. The main idea consisted of letting the student to use their phones to unlock the doors. Later the idea was expanded- where added more functionalities (Not just to lock and unlock the door using the mobile phone) such as student monitoring and integration to student project management system.

The development of the thesis work involved a lot of research in the field of wireless communications, embedded systems and software design.

As a result, a Smart Lock System was developed for the thesis work to remotely control the access to critical area. The whole process of the work lasted six months and it was conducted in Technobothnia laboratory. The System is based on the Bluetooth modules, serial communication, embedded system and client/server GUI application. The two Bluetooth modules were used to pass wirelessly the data between the M.D. and PC. Embedded system was used to verify the data which was passed from the users and control the door. The GUI application was used as a monitoring system.

During the test phase, the Smart Lock System could successfully control the solenoid and BT-modules; the data between mobile device and monitoring system was successfully checked and passed. The objectives of the thesis were achieved and the processes run according to the plan.

CONTENTS

LIST OF FIGURES

ABBREVIATIONS

MCU	Microcontroller Unit
BPS	Bits per Second
BT	Bluetooth
MD	Mobile Device
MFC	Microsoft Foundation Classes
UART	Universal Asynchronous Receiver/Transmitter
SI	Serial Interface
BTS	Bluetooth Protocol Stack
RxD	Receive Data
TxD	Transmit Data
I/O	Input/Output
ISP	In-System Programming
LED	Light Emitting Diode
GND	Ground
VCC	Positive voltage supply
Vref	Reference Voltage
T	Transistor

1. INTRODUCTION

In order to increase the security level of critical area and data, a system, which is called “Smart Lock System”, was developed. The system was designed to remotely open the door using a mobile device and to give access only to the authorized personnel. It is based on a few key parts: Bluetooth-radio communication, serial communication, GUI monitoring system and embedded system. There are used two HC-06 Bluetooth-to-UART modules, one of which is connected to the embedded system which controls the lock, another one is connected to the serial port of the host PC, in order to receive the data and to show all the needed information about the personnel according to that data. The embedded system consists of the Atmega32L microcontroller, a few transistors and a solenoid, which opens the door.

The monitoring system is a GUI application written in MFC and it is used to monitor the activity of the personnel. It consists of two parts: client and server side. The client side is connected via a Bluetooth module to the embedded system. When the valid is received, it passes the data to the server side application, which is located in the security personnel PC. In order to store the data about the people, there was designed a database where all this data is stored. This database can be easily modified through the application. The client side of the application contains all the needed functions, including Serial Communication options, network options, processing in the idle state, personnel data options. All the history of personnel activity is written to a log file. The server side of the application will only receive the data from the client. It also contains the data options of the personnel.

The third part of the system is mobile application. Every authorized member of the Academy will have a special mobile application installed on his mobile device. Only this application will be able to communicate with the embedded system, because there was developed a special data format for this communication. If a person is not a part of the Academy, the application is deleted

from his device. In order to prevent the application copying and thus give access to unauthorized people, the application was developed mobile device dependent.

There are a few main benefits of the system. Firstly, it is very secure: unauthorized people will not have any possibility to gain access to the laboratory. Secondly, there is no need for the key – the only thing which will be needed is a registered mobile device with the needed software. All the students' activity is monitored: the chief of the laboratory will exactly know who came in, how much time the member was inside, when he left and so on. The application contains all the information about the members: mobile phone number, email address, what project he is currently running etc. In order to prevent the door locking in case of absence of power, the microcontroller will monitor the battery status and will inform when the battery will need to be changed or recharged. In addition, this application is very extendable: it can be used everywhere.

This thesis work consists of several parts:

In chapter 2, there is a brief explanation of the system work and how the data is processed.

Chapter 3 includes all the needed theory. It consists of several parts: Bluetooth protocol stack theory, Bluetooth module AT commands, MFC basics theory, microcontroller and solenoid.

In chapter 4 is described how the hardware and software parts are developed and combined to a whole system.

Chapter 5 states how the system was tested including all the needed analysis.

In chapter 6 are presented the conclusions regarding the thesis work and possibilities of future extensions.

2. SYSTEM STRUCTURE

This chapter will describe the system architecture and data processing.

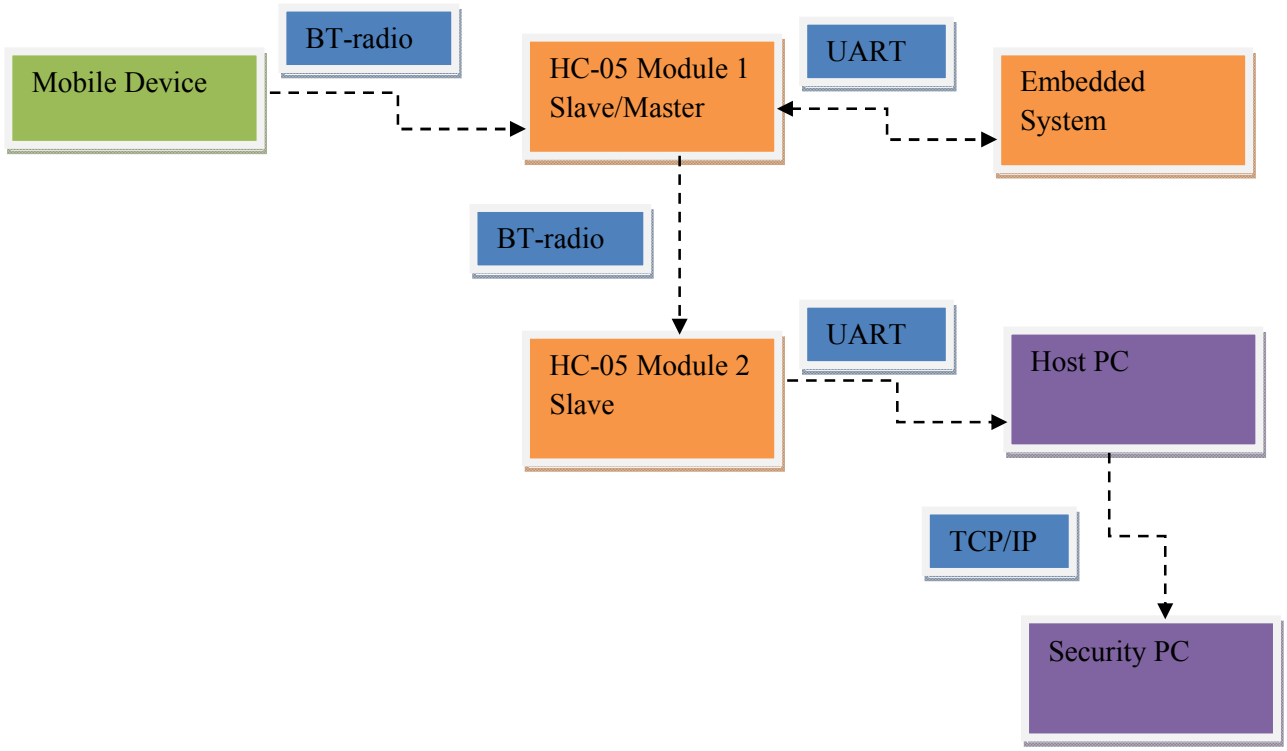

Fig.1. "Smart Lock System" block diagram

As shown in figure 1, the system consists of several parts. The mobile device can be any device: mobile phone, iPad etc. – in general, any device able to communicate via Bluetooth protocol. Two HC-05 Bluetooth modules are used to wirelessly pass the data to the system: from mobile device to embedded system and from Module 1 to Module 2, which is connected to the host PC. HC-05 module 1 passes the data using UART protocol to the microcontroller, which decides to allow access or not, depending on the data: if the data is valid – the microcontroller gives access to the restricted area. Then, this data is passed wirelessly to the second HC-05 module, which passes the data to the host PC and the monitoring application. As it was stated before, according to the data received the application will display who came in. If the received data is valid the information will be passed to the server PC. In order to inform that the person will

no longer stay in the room, he will have to pass the data to the system again, and in this case his status will change. The system has one more feature: it will not close the door while there is somebody present. The door will be locked only when the last person leaves the area. The most important reason for this decision is power saving, because the microcontroller will hold the transistor opened only if somebody is inside.

The application will not pass all the data from host PC to the server. Instead, there will be used special “codes” which will be passed, and depending on that code, the server will decide whose this code is. In case of power loss, there will be possibility to restore all the data from the separate files.

The mobile device application is machine dependent, which means that even if the application will be copied to another device, the person will not be able to get access with it.

3. THEORETICAL STUDY

3.1 Bluetooth Protocol Stack and HC-05 module

Bluetooth technology was originally invented by Ericsson in 1994 as a wireless alternative to RS-232 data cables. It is a standard for exchanging data over short distances using UHF radio waves in the band from 2.4 to 2.485 GHz from different devices and creating personal area networks and each channel having a bandwidth of 1MHz. The data is transmitted in packets and each packet is transmitted on one of 79 designated Bluetooth channels.

The word “Bluetooth” is a version of the Scandinavian Blatand/Blatann, the epithet of the tenth-century king Harald Bluetooth who united Danish tribes into a single kingdom. The Bluetooth logo is a bind rune merging the Younger Futhark runes Hagall and Bjarkan.

Gaussian frequency-shift keying was used as a modulation technique for Bluetooth communication. Also, other modulations can be used, such as 2.0+EDR, π/4-DQPSK (Differential Quadrature Phase Shift Keying) and 8DPSK. Devices which functions with GFSK are operating in Basic Rate (BR) with 1Mbit/s data rate. EDR term (Enhanced Data Rate) is used to describe 8DPSK and π/4-DQPSK schemes, which give 3 and 2Mbit/s data rate respectively.

Bluetooth is a packet-based protocol and has a master-slave structure. One master can communicate with up to 7 slave devices. All slaves share masters’ clock rate. Packet exchange is based on the basic clock, defined by the master, ticking at 312.5 us interval.

The IEEE standardized Bluetooth as IEEE 802.15.1 but it is no longer maintained. It is managed by Bluetooth Special Interest Group, or SIG, which includes more than 20000 member companies.

In order to support the wide range of Bluetooth applications, there are many Bluetooth software layers. Radio Baseband, Link Controller and Link Manager are the lower layers and are very similar to the over-air transmission. They can

provide voice connections and data pipes between Bluetooth devices. In order to ease the integration of the Bluetooth, there are some middle layers that hide some of the complexities of the wireless communication. These layers can take many familiar data formats and protocols, package them, multiplex and pass them so that it matches the lower layers' capabilities.

Radio Baseband, Link Manager, Logical Link Control and Adaptation Protocol (L2CAP) and Service Discovery Protocol (SDP) are the layers which are fundamental to Bluetooth technology. Different applications may require different selections from these and higher layers and each profile can call up different layers. It is important to notice that not all vendors support all the layers, that's why one must be sure that the stack which is used supports all the layers the application require. In Figure 2 are shown the layers defined by the Bluetooth specification.

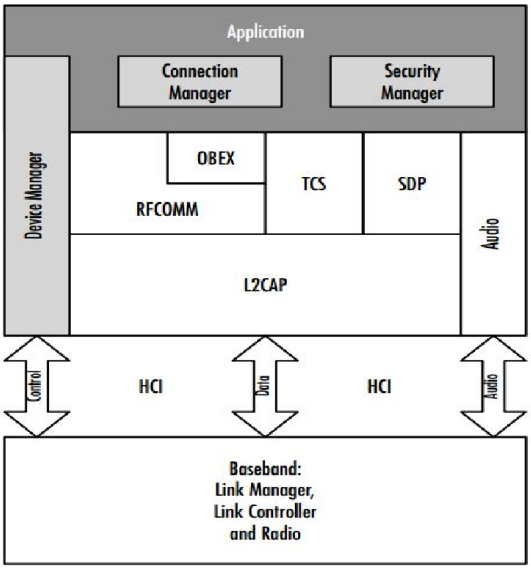

Fig.2 Bluetooth Protocol Stack

3.1.1 L2CAP

L2CAP – Logical Link Control and Adaptation Protocol multiplexes upper layer data into one single Asynchronous Connectionless (ACL) connection between two devices and if the device is master it directs data to the appropriate slave. It also split the data into pieces that fit the maximum HCI payload (HCI – Host Controller Interface – connects higher layers on a host to lower layers on a Bluetooth device). Each L2CAP channel has a unique CID – Channel Identifier. CIDs 0x0000 to 0x003F are reserved; 0x0000 is unused; 0x0001carry signaling information; 0x0002 identifies received broadcast data. PSM value – Protocol Service Multiplexor can identify the stack layers above the L2CAP. When device requests a connection to particular PSM, L2CAP allocates the CID. Each Bluetooth layer above L2CAP has its' own PSM:

- SDP – 0x0001;
- RFCOMM – 0x0003;
- TCS-BIN (Telephony Control Protocol Specification Binary) – 0x0005;
- TCS-BIN-CORDLESS – 0x0007;

L2CAP works only with data, not the voice and all channels (not broadcast) are considered reliable.

3.1.2 RFCOMM

RFCOMM – Radio Frequency-oriented emulation of the serial COM port on a PC – the protocol which is used to emulate RS-232 serial communication over an L2CAP channel. It is based on TS 07.10 standard for software emulation of the RS-232 interface. TS 07.10 can multiplex several emulated ports into one single data connection using different DLCI – Data Link Connection Identifier for each port. Each TS session can connect only over single L2CAP channel. A master device must have a separate RFCOMM session for each slave.

In Bluetooth version 1.1, TS 07.10 was added a flow control capability.

3.1.3 SDP

SDP – Service Discovery Protocol allows a device to discover the services which are provided by other devices. This layer acts like a service database. The local application must register available services in a database and keep this database up to date. Remote devices may query this database to find out what services are available and how to contact them. For example, a mobile phone with Bluetooth headset uses this protocol in order to find what Bluetooth profiles the headset can use (Headset Profile, Hands Free Profile, Advanced Audio Distribution Profile etc.) and the protocol multiplexer settings needed for the phone to connect to the headset. Each service is identified by Universally Unique Identifier (UUID), with official services (Bluetooth profiles) assigned a short form UUID (16 bits instead of full 128).

These 3 protocols are the most important for the thesis work. The rest of the protocols can be studied in /4/.

3.1.4 Device Connections

Inquiry – is the procedure which is used to find the devices.

Paging – is the procedure used to connect the devices.

In order to be able to connect to the device, first of all it must be in a discoverable mode and transmit the following information:

- The name of the device
- The class of the device
- The list of available services
- Technical information (i.e. features, manufacturer and so on).

Basically, these two procedures represent special sequences of frequencies which are known to all the devices. If a device in a listening mode receives a correct transmission it sends out a reply. If the device must be discovered, the application must place it to the listening mode. The discoverable mode or inquiry scanning is

the mode in which a device can be found. The listening mode in which a device can be connected is called connectable mode or page scanning.

In order for a communication to take place, the device must transmit data on the same frequency the other device is receiving. To do so, the transmitter changes its frequency very fast, 1600 times per second, while the receiver change its frequency very slowly – every 1.28 seconds. Frequency hopping is not synchronized, so the procedure must last long enough for the devices to collide on the frequency which is not interfered.

A Bluetooth device which transfers the voice cannot find or connect to other devices. This is due to the fact that voice links take priority over everything, while inquiry and page operations take precedence over data transfers. There is possibility to inquire and page in the gaps between the voice transmission, but because this transmission is of the highest priority, the responses very often will be lost, so finding and connecting the devices can be very slow and unreliable while the voice link is in use.

After a device was discovered, the information which was gathered can be used to create the connection. To create this connection, one device pages another device, which must be in Page Scan mode to respond. At the Radio level, a connection means that the devices are frequency-hopping together and synchronized to the master device's Bluetooth address and clock. In the upper protocols, it means that ACL link was established for the data to pass over. This allows using the L2CAP and other protocols including the service discovery. To allow an incoming connection, a device must be placed in Page Scan mode.

3.1.5 Bluetooth Module

There are a lot of different Bluetooth modules presented on the market. They differ by price, functionality, hardware and software presented. In order to achieve the goals of the thesis work, HC-05 Bluetooth SPP (Serial Port Protocol) module was used. This is a very cheap, powerful and easy to use module, which can be integrated very fast in any Bluetooth application. It is fully qualified Bluetooth

V2.0+EDR (Enhanced Data Rate) 3Mbp/s modulation with complete 2.40GHz radio transceiver and baseband. The module is based on Cassira CSR Bluecore 04-External Chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature).

3.2 Bluetooth module AT commands

It is possible to change the default Bluetooth setup using the AT commands. To do that, the module must be started in proper way. The PIO11 pin must be connected to VCC. When the module is started, it will enter in AT command mode. To communicate properly with the device, the Serial Monitor software(or another) must be configured to use 38400 baud rate, 8 data bits, 1 stop bit and to attach CR+LF to the end of the message.

Before sending the actual commands, there must be sent a test command in order to check the connection. If there are no errors, the module will answer “OK” or the result of the command. (note: all the letters are capital)

1. Test command:
   AT
2. Reset Command:
   AT+RESET
3. Get Firmware version
   AT+VERSION?
4. Restore defaults
   AT+ORGL
5. Get Module Address
   AT+ADDR?
6. Set/Check module name
   AT+NAME=<PARAM> - To set the name
   AT+NAME? – Command to check the name
7. Set Check module role

AT+ROLE? – To check the role. The answer will be ROLE:number, where number refers to: 0 – slave mode, 1 – master mode, 2 – slave loop mode

AT+ROLE=<PARAM> - command to set the mode of operation, where PARAM is 0, 1 or 2.

8. Set/Check PIN code

   AT+PSWD? – Check command

   AT+PSWD=<PARAM> - The command used to set the PIN code

9. Set/Check Serial parameters:

   AT+UART? – Check command

   AT+UART=<PARAM1>,<PARAM2>, <PARAM3>, where PARAM1: Baud rate, PARAM2: Stop bits, PARAM3: Parity

10. Initialize the SPP profile

    AT+INIT

These are the most commonly used AT commands.

3.3 Microcontroller

A microcontroller is a small computer located on a single integrated circuit. A typical microcontroller contains different peripheral components (which are external to processing unit) and the processing unit itself. The processing unit includes: registers, control clock and arithmetic-logic unit or ALU. There are presented different memories: ROM memory or Flash memory, which is used to store the program which must be executed by the microcontroller, SRAM – which is used to temporarily keep the needed data and EEPROM memory which is used to constantly save some needed data (this memory is electrical independent and it keeps the stored data even if the microcontroller is off). In addition to that, every microcontroller comes with different peripherals, which can be: ADC/DAC, which are used to communicate with the outside world, timers/counters which are used to work with timing operations, different communication modules, which can be UART, TWI, CAN, USB and so on. In general, it is very hard to mention all the peripherals which a microcontroller can include, because there are a lot of microcontroller manufacturers, microcontroller types and their power ranges (by

power in this case I mean the operating frequency, word length(8 bits, 16 bits or 32 bits) and so on). The main difference between a microcontroller and a microprocessor is that a microprocessor is designed to work with general purpose applications, such as PCs, but the microcontrollers are designed to work in embedded applications.

In this thesis work, the microcontroller acts as a main control unit. It is used to communicate with the user and to validate the information which is received from the Bluetooth module. The module is also controlled by it using AT commands. The commands are sent via UART interface. In order to switch HC-05 to the AT command mode, special microcontroller pins are used: one of them is used to reset the module, another one is used to connect the KEY pin to the power supply.

In addition, the microcontroller is used to drive the solenoid, which opens/close the door.

3.4 Solenoid

A solenoid is a coil wound into a tightly packed helix. The term was invented by French physicist André-Marie Ampère to designate a helical coil. In engineering, the term may also refer to a variety of transducer devices that convert energy into linear motion. The term is also often used to refer to a solenoid valve, which is an integrated device containing an electromechanical solenoid which actuates either a pneumatic or hydraulic valve, or a solenoid switch, which is a specific type of relay that internally uses an electromechanical solenoid to operate an electrical switch; for example, an automobile starter solenoid, or a linear solenoid, which is an electromechanical solenoid.

Electromechanical solenoids consist of an electromagnetically inductive coil, wound around a movable steel or iron slug (termed the armature). The coil is shaped such that the armature can be moved in and out of the center, altering the coil's inductance and thereby becoming an electromagnet. The armature is used to provide a mechanical force to some mechanism (such as controlling a pneumatic valve). Although typically weak over anything but very short distances, solenoids

may be controlled directly by a controller circuit, and thus have very quick reaction times.

The force applied to the armature is proportional to the change in inductance of the coil with respect to the change in position of the armature, and the current flowing through the coil. The force applied to the armature will always move the armature in a direction that increases the coil's inductance.

3.5 MFC

The Microsoft Foundation Classes Library or MFC is a library which gives possibility to a developer GUI applications for Windows in C++ language using a very rich library classes. It provides a lot of code which is necessary for managing windows, menus, dialog boxes, buttons; input/output functions; work with the databases; sockets, RTTI and so on.

It was firstly introduced in 1992 with Microsoft C/C++ 7.0 compiler for use with 16-bit versions of Windows as an object-oriented wrapper for Windows API. Initially, Windows API interface was developed as procedural. But, when the object-oriented extensions of C language got their next development in C++ language, the next natural step was to develop the object-oriented interface framework for Windows API. In 1992, Microsoft developed such an interface as Application Frameworks (AFX) product. This interface got its' next development in Microsoft Foundation Classes packet, which exists today in a little changed form.

The MFC library is object-oriented. But that doesn't mean that a developer cannot use Windows API functions in MFC program: all what is needed is to connect the needed API library and the functions can be used. Actually, MFC was developed as object-oriented interface-framework for Windows API. All what is needed is to add application-specific code into the framework. It is very simple to override the basic functionality that MFC framework provides due to the nature of C++ class programming.

MFC shortens development time; makes code more portable; provides tremendous support without reducing programming freedom and flexibility; and gives easy access to "hard to program" user-interface elements and technologies, like ActiveX technology, OLE, and Internet programming. Furthermore, MFC simplifies database programming through Data Access Objects (DAO) and Open Database Connectivity (ODBC), and network programming through Windows Sockets. MFC makes it easy to program features like property sheets ("tab dialogs"), print preview, and floating, customizable toolbars.

Generally speaking, practically all the MFC classes are derived from a single class – CObject. Its' main functions are: processing information about run-time type and object saving (or serialization, if MFC terminology is used), as well as diagnostic of output of the derived classes. These classes are able to use the functional possibilities of CObject class.

MFC, as all GUI programs is event driven. Application responds to user actions which generates events (such as button press, mouse movement etc.). These events are communicated to programs through messages which are sent by the operating system.MFC provides all the needed logic for handling messages in the object-oriented framework.

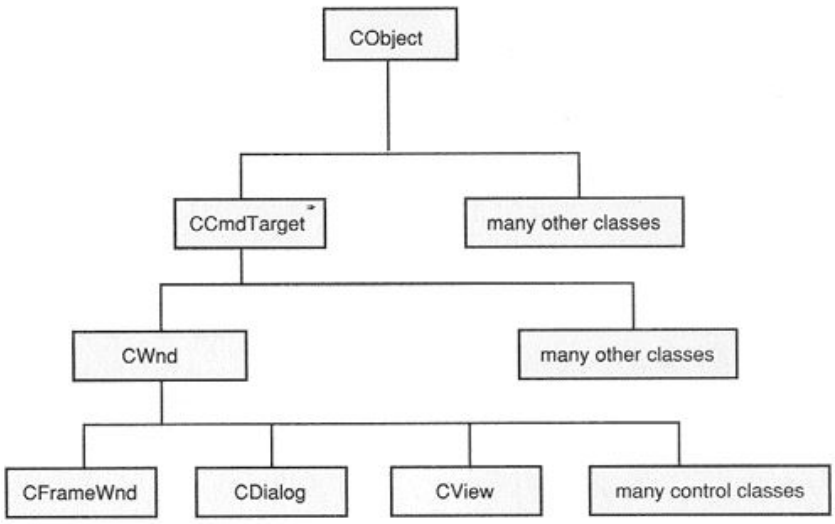

Fig.3 Basic MFC class hierarchy

CCmdTarget class is the base for MFC message-map architecture. Message-map routes commands or messages to the member functions which handles them.

CWnd class provides the base functionality of all window classes.

CDialog class is the base class used for displaying dialog boxes on the screen. There are two types of boxes: modal and modeless. A modal dialog box must be closed by the user before the application continues. A modeless dialog box allows the user to display the dialog box and return to another task without cancelling or removing the dialog box.

The CView class provides the basic functionality for user-defined view classes. A view is attached to a document and acts as an intermediary between the document and the user: the view renders an image of the document on the screen or printer and interprets user input as operations upon the document.

The CFrameWnd class provides the functionality of a Windows single document interface (SDI) overlapped or pop-up frame window, along with members for managing the window.

These are the basic classes which are needed in every MFC application. The rest of the classes can be used very easy; the only thing which is needed is to include the header files and to create the exemplar of the class.

There are several possibilities of starting an MFC application. The easiest one is to use MFC Application Wizard in Microsoft Visual Studio. This is the best choice, because it allows the SDK to perform all the needed initializations, and the developer will just need to add the required features to the application. The second possibility is to start from the plain Win32 application and to perform all the initialization, including header file connections and parental window creation manually.

4. IMPLEMENTATION OF THE PROJECT

This chapter describes how the hardware and software systems were designed and integrated.

4.1 Hardware Design

4.1.1 HC-05 Bluetooth module

There a lot of Bluetooth modules available on the market. In order to achieve the goals of the project, the HC-05 Bluetooth SPP module was chosen. This is a very small, cheap and easy-to-use module which can be integrated very fast in any Bluetooth application.

Some of the hardware features of the HC-05 module:

- -80dBm sensitivity
- Up to +4dBm RF transmit power
- Low power consumption, up to 5V and 50mA consumption
- PIO control
- UART interface with settings possibility
- Integrated antenna
- Edge connector

Some of the software features of the module are:

- Default Baud rate: 38400, 8 Data bits, No parity control, and 1 stop bit. Supported baud rates are: 9600, 19200, 38400, 57600, 115200, 230400, and 460800.
- Auto-connect to the last device on power
- Pairing device default connection possibility
- Auto-reconnect in 30 minutes if disconnected as result of beyond of connection range
- AT-command mode of operation with setting setup possibility

In order to drive the module into HC commands mode, there is a need to perform a few steps. First of all, the “KEY” pin, which is GPIO34 on the chip, must be connected to the power supply. After that, the chip must be restarted. When it is on, it will automatically be set to command mode. The default configuration of the chip in the command mode is: 34800 baud rate, 8 data bits, 1 spot bit and no parity checking. When the chip is configured, the “KEY” pin must be disconnected and the module restarted.

HC-05(as well as other modules of this line) has an interesting feature, which can be used to automate the run of the system. The feature is the possibility to reconnect to the slave device even after restart or change of settings. It is very useful, because as it was stated, one module will work only in the slave mode, while the one which is connected to the embedded system will change its roles. However, change of settings using the microcontroller was a bit complex, especially taking into consideration that the module must be restarted. The solution was to use two transistors . The transistors are connected to the microcontroller output pins and are operating as keys, switching the “KEY” pin and module power on and off. The Bluetooth SPP module HC-05 is presented in the Fig. 4.

All the data is passed automatically, however, there is possibility to develop applications for the HC modules using the IDE, BlueLab xIDE3. Developing applications for this module is problematic, because it requires their development board and programmer, which are expensive.

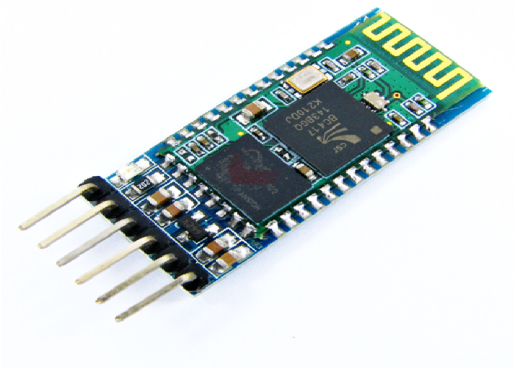

Fig.4 HC-05 Bluetooth Module

The HC-05 module has an advantage in respect to HC-04 in cense that is has all the needed pins connected to the connector and all what is required is to correctly connect them to the device. Those pins are: VCC and GND for power supply, TxD and RxD for serial communication and KEY pin to set the mode of operation.

4.1.2 Atmega32 microcontroller

The Atmel Atmega32 microcontroller is the very heart of the system. It is used in controlling the Bluetooth module and giving access to the laboratory, as well as performing the embedded system side data verification.

The main features of Atmega32 microcontroller, which are relevant for this thesis work, are:

- High-performance, Low-power Atmel AVR 8-bit microcontroller
- Up to 16 MIPS throughput at 16MHz
- 32Kbytes of In-System Flash program memory
- 1024Bytes of EEPROM
- 2Kbytes of internal SRAM
- Two 8-bit Timer/Counters
- One 16-bit Timer/Counter
- Programmable Serial USART
- 4.5V – 5.5V Operating voltages
- 32 Programmable I/O lines
- 40-Pin PDIP Package (used in this work)

To perform the tests and debugging of the software, AVR Universal Board was used. It contains the needed socket for microcontroller, as well as output ports where the peripheral can be connected, LED-s for status indication. The AVR Universal Board is presented in Fig.5.

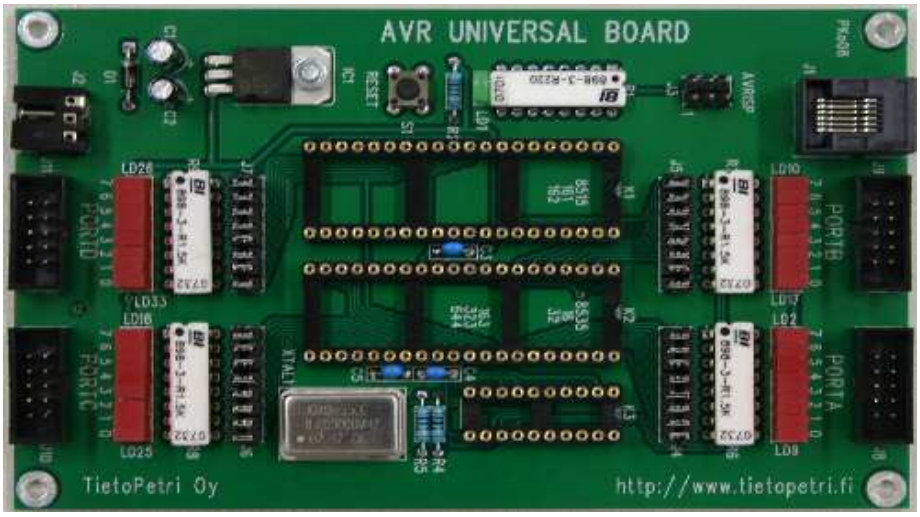

Fig.5 AVR Universal Board (source – tietopetry.fi)

In order to control the Bluetooth module, the microcontroller already contains the needed AT commands, which it passes to the module. To drive the module and "KEY" pin power, the microcontroller opens and closes the transistors. These are the following steppes which microcontroller performs if it receives data:

1. The microcontroller checks the data to find out who sent it and if it is valid.
2. If the data is valid, microcontroller starts the configuration of the module:
   2.1 To drive the module into the AT commands mode, the "KEY" pin is connected to VCC and the module is restarted. Then, the microcontroller switch on the solenoid to open the door.
   2.2 Microcontroller passes all the needed AT commands to the module: changes its role of operation; sets the address of the auto-paired device; initializes the SPP profile. The CR+LF (carrier return + line feed, "\r\n") must be present in the end of each AT command.
   2.3 Then, the module is restarted, and it will automatically connect to the slave device. During this time, microcontroller makes a little break in its operation to give the module time to successfully connect and after this break it sends the data about the person to the host PC.

2.4 When the data is successfully sent, the microcontroller again modifies the module settings, in this case changing its role from master to slave and waits for next data to arrive.

The microcontroller serial communication pins are connected directly to the HC-05 communication pins. To drive the KEY pin and the power of the module, the microcontrollers' pins drive the operation of transistors, which operate as switches, providing power to the KEY and VCC pins of the module. In this project, microcontroller drives the operation of 3 transistors: BD140 PNP bipolar transistor is used to provide the power to the HC-05 module, 2N3904 NPN bipolar transistors drives the operation of KEY pin and BD139 NPN bipolar transistor is used to control the solenoid. In Fig.6 is presented the schematic of the embedded system. In order to make the change of components easier, the HC-05 module is plugged-in in the SIL-6 connector, microcontroller is plugged into DIP-40 connector. The TBLOCK-2 connectors are used for connection of power supply and solenoid.

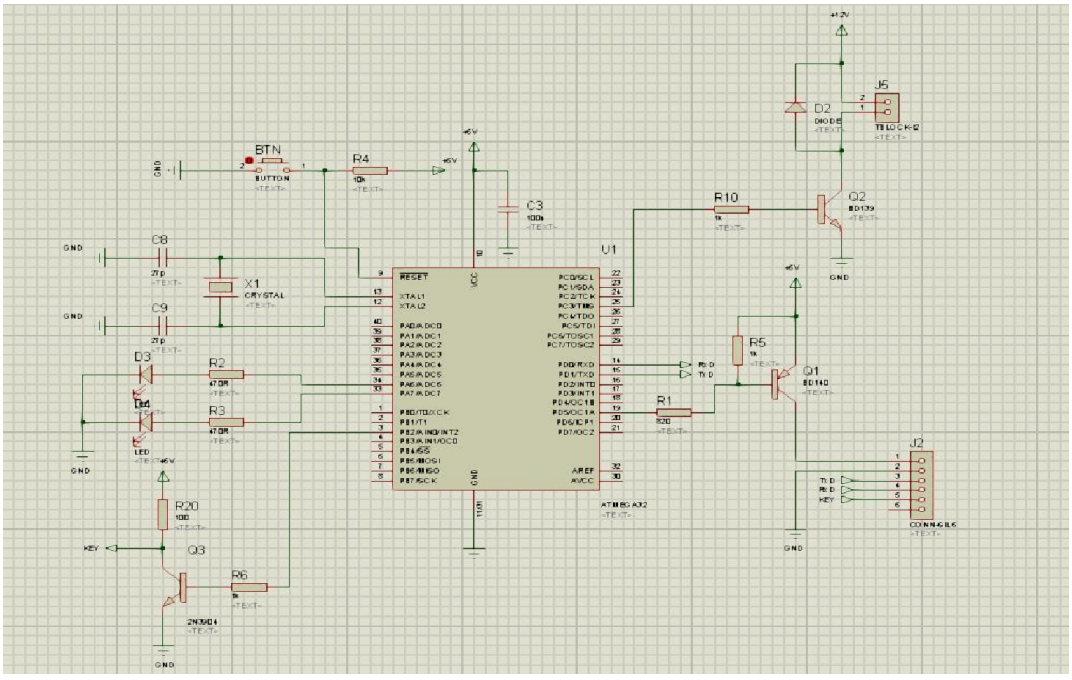

Fig.6 The schematic of the embedded system

Two LEDs presented in the schematic are used to show the status of the lock – if it is closed or opened. One more LED is presented in the power-supply part of the

circuit to show if the board is on. The CAD application which was used to create the schematic and the layout of the PCB was ISIS Proteus (for the schematic) and Proteus ARES (for the layout).

4.2 Software Design

4.2.1Microcontroller Application

The microcontroller application was written in C language, the Atmel AVR Studio IDE was used. To program the microcontroller, the AVR ISP mkII programmer was used. The application consists of several important parts, which are: serial communication interrupts, and the main function, in which is performed the initialization of the needed variables and some computation in the infinite-loop for the debugging purposes. The flow chart of the application is presented in Fig.8.

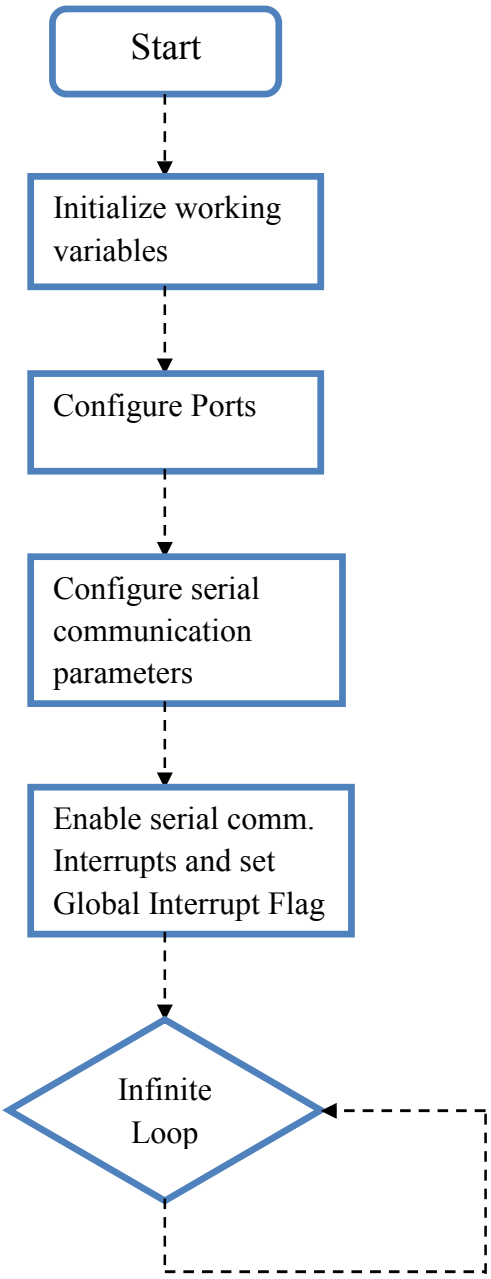

Fig.7 The main function flow chart

One of the most important parts of the main function is to initialize all the working variables, including AT-commands character arrays, ports configuration and the initialization and enabling of serial communication interrupts. When the system starts, the HC-05 module connected to the board is switched-off, the solenoid power is disconnected and therefore the door is closed. In order to decrease the level of computational tasks when microcontroller is working with

the module, the serial communication is initialized with the following settings: 38400 Baud rate, 8 data bits, one stop bit and no parity checking. To initialize the serial communication on Atmega32 microcontroller, there are needed to perform several steps. Firstly, there must be loaded a special value to the UBRRH/UBRRL registers. The value is dependent on the needed speed of operation and the frequency of the oscillator. The Atmel Company made a great deal when included in the datasheet the table with the needed values of the registers depending on the frequency and the needed baud rate. In our case, the microcontroller is operating on 8MHz crystal and we needed 38400 Baud rate. Thus, the value which must be set is 12(this table can be found in Atmega32 datasheet, listed in the Appendix). The next steps which must be done are the setting of the data bits number, the number of stop bits and parity control bits, as well as enabling of the interrupts. The UCSRB and UCSRC registers are used for these purposes. The RXEN and TXEN bits in UCSRB registers enable the UART receive and transmit interrupts if are set. The URSEL and UCSZ0…UCSZ2 are used for accessing the UCSRC register and the character size respectively. There are other bits presented in this register, but there was no need to initialize them with other values except of those set by default. These bits are USBS bit, which is used to set the number of stop bits, and it uses 1 stop bit by default (the bit is cleared), UPM0:1 which are used to select the parity mode (disabled by default) and UMSEL register which is used to select mode of operation: asynchronous or synchronous mode. The actual UART data can be achieved using the UDR register. It is used for both receiving and transmitting the data.

When all the needed variables and settings are done, the microcontroller enables all the interrupts. The serial communication interrupts play a very important role in this work, because they are used to communicate with the modules and host PC, as well as checking the validity of data and to accept or reject the data, thus giving or not the access to the laboratory. Initially, only the UART Receive Interrupt is enabled, and the Transmit Interrupt is enabled through it. The reason to do that was that the data must be sent to the host PC only if it is valid, otherwise the microcontroller has to skip it as being irrelevant. The flow chart of serial communication interrupts is presented in Fig.9. The embedded system

accepts only the data of the special frame and length. If the data which was received in the interrupt is valid, then the microcontroller enables the Transmit Interrupt, drives the module to the master mode of operation and passes all the needed data to the host PC, as well as giving access to the lab. After the end of operation, microcontroller drives the module back to the slave mode; otherwise there will be no possibility to connect to it again. When these steps are performed, the Transmit Interrupt is disabled and the microcontroller waits for the next data.

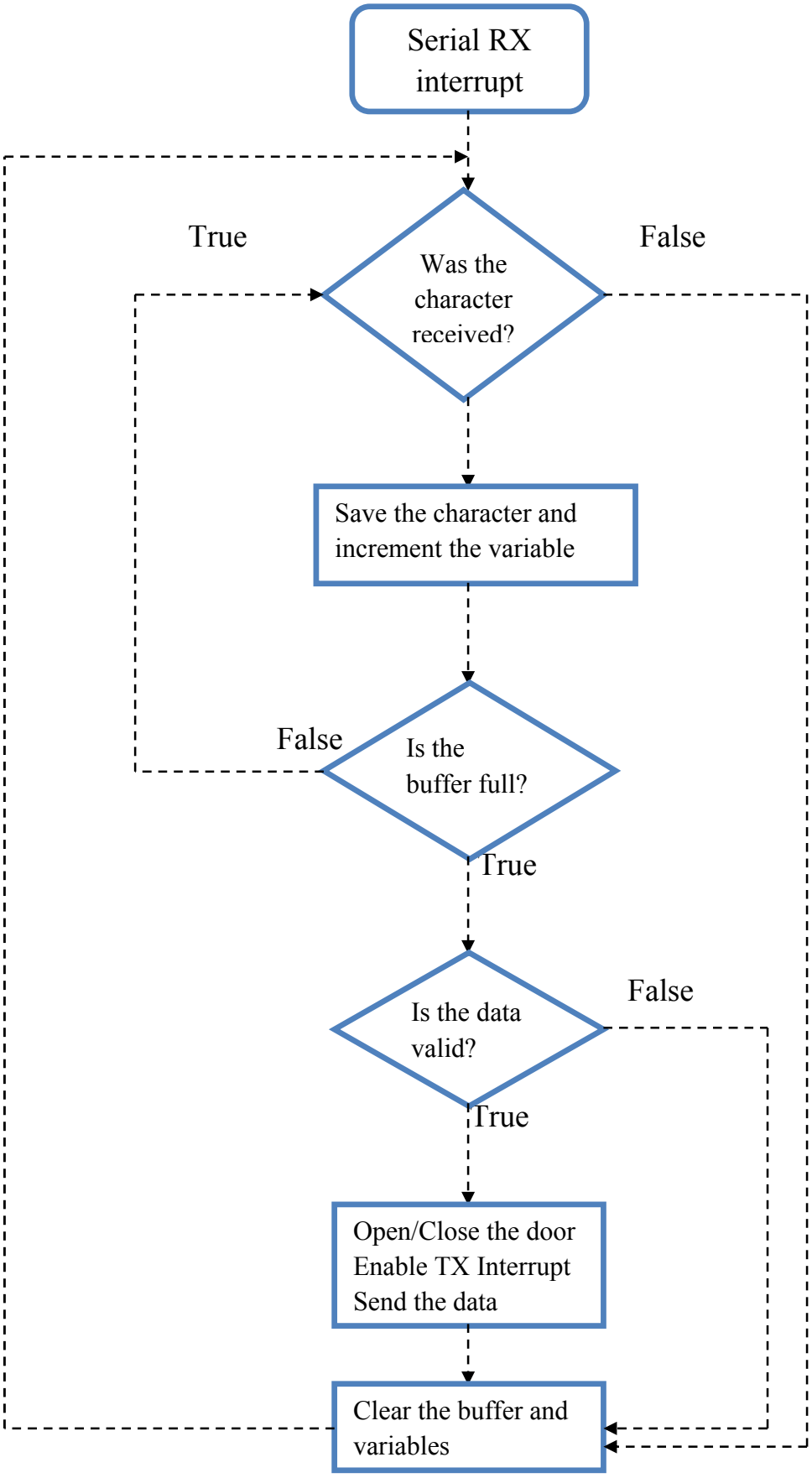
Serial RX interrupt
Was the character received?
True
False
Save the character and increment the variable
Is the buffer full?
False
True
Is the data valid?
False
True
Open/Close the door
Enable TX Interrupt
Send the data
Clear the buffer and variables

Fig.8a Serial Communication RX Interrupt

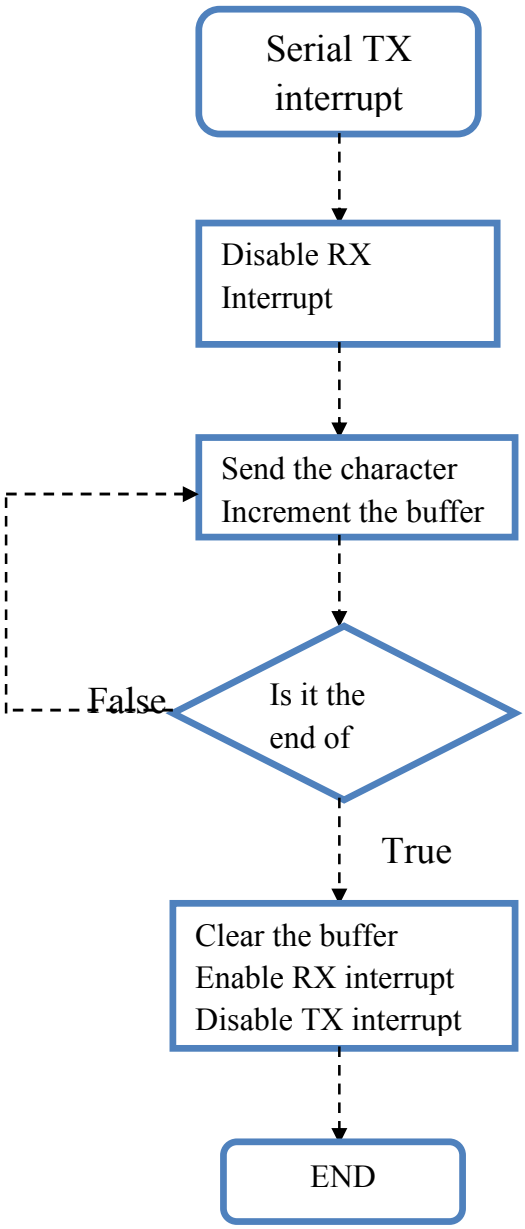

Fig.8b Serial Communication TX Interrupt

In order to close the door, the separate command is used. If the microcontroller receives it, it will lock the door disconnecting the solenoid from the power supply. No module configuration or data pass procedures will be performed in this case.

4.2.2 MFC Application

MFC application plays a significant role in the thesis work. It executes a lot of functions:

1. Database for storing the personal information.
2. Data receiver.
3. It gives possibility to add new users, change the user data, establish serial and network communication.
4. It gives possibility to monitor the user activity.
5. It possesses an intuitive graphical interface, which allows fats configuration on a new system.
6. The application is client/server – the server side of the application is supposed to be installed on the security personnel PC.

The main window of the application is presented in Fig.9

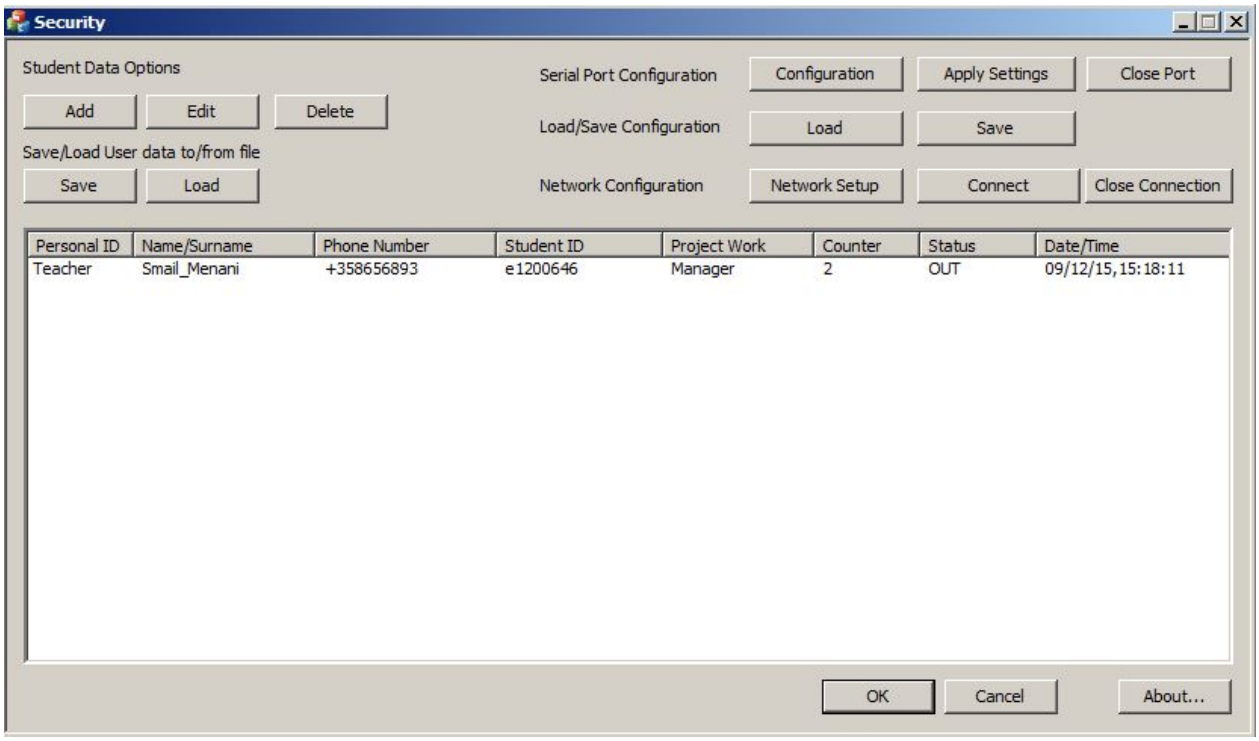

Fig.9 The applications' main window

The main window of the application contains a lot of options. It consists of two parts: communication setup and user data change/include options. Everything can be achieved using the buttons, which already contains all the needed help for ease of navigation. All the configurations and data can be stored into separate files, which helps to restore the data in case of fault. As it was stated, the application is client/server, which means that after the validation of data, the message which contains all the needed information will be passed to the server.

Because communication plays a key role in the thesis work, the application must then continuously listen for the incoming data. There exist a few possibilities to do that in MFC: using the OnKickIdle() or OnIdle() functions(depending on the application: in case of Dialog Based application is used OnKickIdle, for SDI(single-document interface), MDI(multiple-document interface) or Document/View application is used OnIdle() function) which was developed to compute the needed tasks when the message queue is empty, using the PeekMessage() and PostMessage() functions and multithreading. Any MFC event is driven by message. Any button press, any mouse press etc. is a message. That means that every control element which is present in the window is driven by separate event handler. The massages are mapped, or sent to the appropriate event handler automatically, the message mapping functionality is provided by MFC.

In Fig.10 is presented the message mapping of the application.

```
//Application Message Map
BEGIN_MESSAGE_MAP(CSecurityDlg, CDialogEx)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BTN_ABOUT, &CSecurityDlg::OnBnAbout)
    ON_BN_CLICKED(IDC_BTN_ADD, &CSecurityDlg::OnBtnAdd)
    ON_BN_CLICKED(IDC_BTN_EDIT, &CSecurityDlg::OnBnEdit)
    ON_BN_CLICKED(IDC_BTN_DELETE, &CSecurityDlg::OnBnDelete)
    ON_NOTIFY(NM_DBLCLK, IDC_LIST_CTRL, &CSecurityDlg::OnDblClkListCtrl)
    ON_MESSAGE(WM_KICKIDLE, &CSecurityDlg::OnKickIdle)
    ON_BN_CLICKED(IDC_BTN_CFGM, &CSecurityDlg::OnBtnCfgm)
    ON_BN_CLICKED(IDC_BTN_CNCLS, &CSecurityDlg::OnBnConnectionClose)
    ON_BN_CLICKED(IDC_BTN_CONNECT, &CSecurityDlg::OnBnConnect)
    ON_BN_CLICKED(IDC_BTNSCONFIG, &CSecurityDlg::OnBtnsconfig)
    ON_BN_CLICKED(IDC_BTN_APLY, &CSecurityDlg::OnBtnAply)
    ON_BN_CLICKED(IDC_BTN_SPCLS, &CSecurityDlg::OnBtnSpcls)
    ON_BN_CLICKED(IDC_BTN_SAVE, &CSecurityDlg::OnBtnSave)
    ON_BN_CLICKED(IDC_BTN_LOAD, &CSecurityDlg::OnBtnLoad)
    ON_BN_CLICKED(IDC_BTN_SPLOAD, &CSecurityDlg::OnBtnSpload)
    ON_BN_CLICKED(IDC_BTN_SVSP, &CSecurityDlg::OnBnSVSP)
END_MESSAGE_MAP()
```

Fig.10 Application Message Map

The MFC message map is enclosed between BEGIN_MESSAGE_MAP (application window class, base class) and END_MESSAGE_MAP (). Each message map starts with the message which is needed to be processed, for example ON_BN_CLICKED (button click message), and as parameters are specified the ID of the element and the handling function. All the events are driven automatically; a developer must only provide the mapping.

The execution of the idle function starts only after the initialization of the serial communication and networking; special flags are used for this purpose.The flow chart of the application is presented in Fig.8.

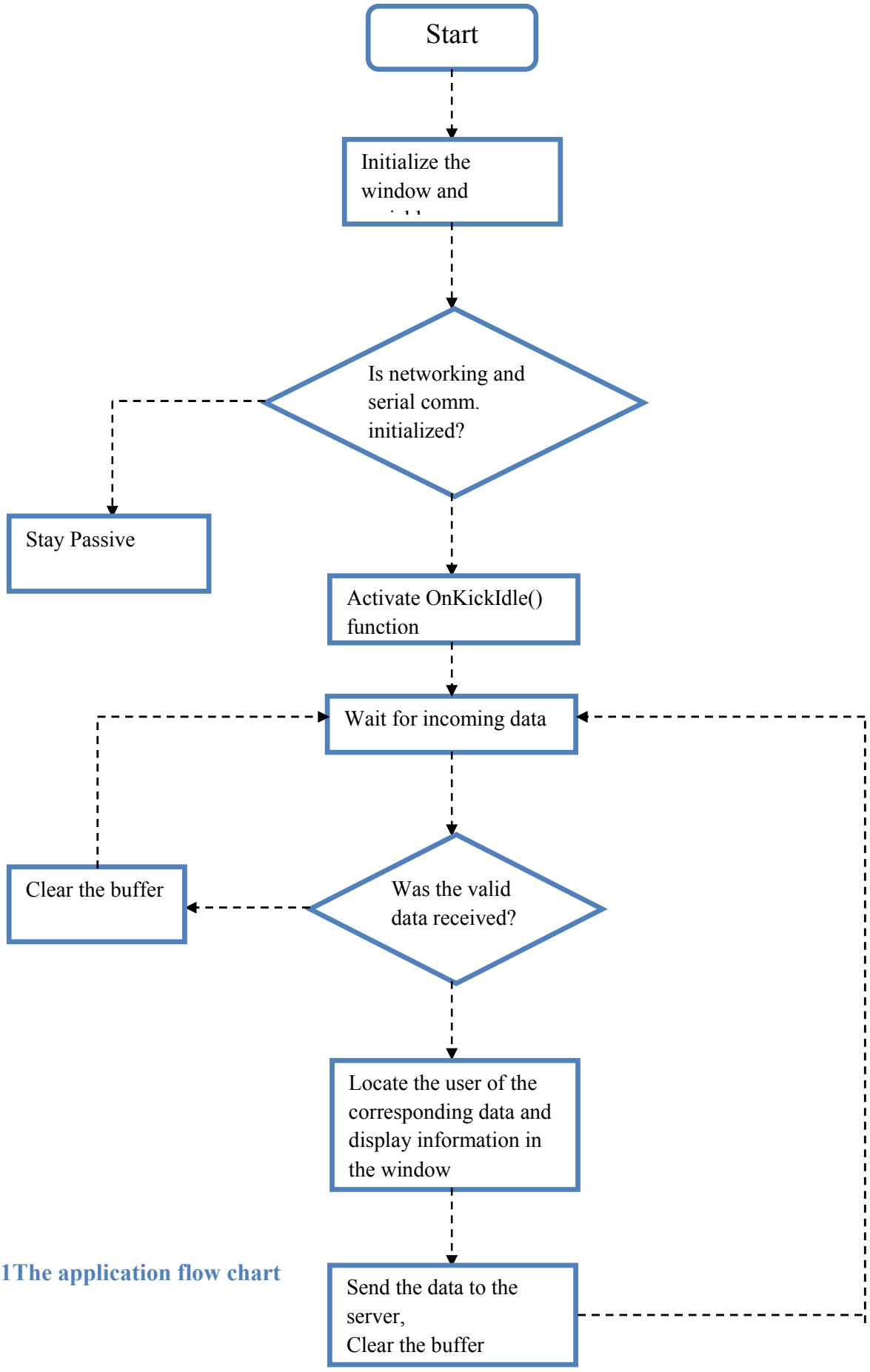

Fig.11The application flow chart

To display the needed information in the window, the CListCtrl class is used. It encapsulates the functionality of a "list view control," which displays a collection of items each consisting of an icon (from an image list) and a label. The developer can control it in a lot of ways – specify the number of columns, specify the names for columns, specify the style of items display and so on.

Each item in a list view control consists of an icon (from an image list), a label, a current state, and an application-defined value (referred to as "item data"). One or more subitems can also be associated with each item. A "subitem" is a string that, in report view, can be displayed in a column to the right of an item's icon and label. All items in a list view control must have the same number of subitems. Class CListCtrl provides several functions for inserting, deleting, finding, and modifying these items. Additional information about how to use this and other classes can be found on the reference page.

The initialization on a new platform or system is done only once. After the first initialization of settings, including serial communication and networking, the user will have possibility to save the settings to the files, which can be loaded on the next start. The same refers to the list of personnel – all the information about them, as well as their status or last access can be saved to the file and then loaded from it. The data is saved to the file as strings with delimiters between the different fields. When this data is loaded from the file, the Tokenize() function is used to split it into parts and then fill the option fields with this information. It is important to mention that MFC does not contain in-built functions or classes which allow working with serial communication. However, this environment is very mobile, in sense that all the additional functions can be taken from WinAPI. All what is needed is to include the needed header file. The work with serial communication is performed in the same way as with files – it is opened and closed with the needed parameters (to read or write) and so on. The additional steps which must be performed are the settings of the baud rate, parity, stop and data bits and timeouts. If the timeouts are not set – the program will stuck on waiting for incoming data. There is possibility to close the current port in order to initialize a new one with new settings. The same refers to the network setup.

The work with network communication in MFC can be achieved through two classes: CSocket and CAsyncSocket. The second one is derived from the CSocket, including all its function members and having its' own. The difference between these two classes is that the socket which will be created using the first class will be blocking, which means that the application will stuck on the waiting of the incoming connection or data – thus the program will stuck. The socket which will be created with the CAsyncSocket will not be blocking, and the application will function normally when listen for incoming connections or data. The application which will run on host PC will operate only as client, for this reason the data fields with the IP address and the port are presented in the menu. The initialization of server requires only the initialization of the port on which it will listen for the incoming connections.

4.2.3 Android Application

The last application in the software part of this thesis work is mobile application. The first version of it was written for the Android platform using Java language and Eclipse development environment. The Android SDK is required for developing application for this platform. The application is used for connection and communication with the module. This application has its' own security measures. To be able to send the data to the module, the user will have to provide the valid information required in its fields. The reason to implement this feature was that the mobile phone can be found or stolen by the person who is not authorized to enter the laboratory. If this person doesn't know what data is required for the application – it will be absolutely useless for him. In addition, the module itself requires a password for pairing and, as it was stated before, the special length and format of the data. The application contains the button controls for both sending the data and closing the door, as well as needed fields which must be filled before proceeding. The Android application flow chart is presented in Fig. 9.

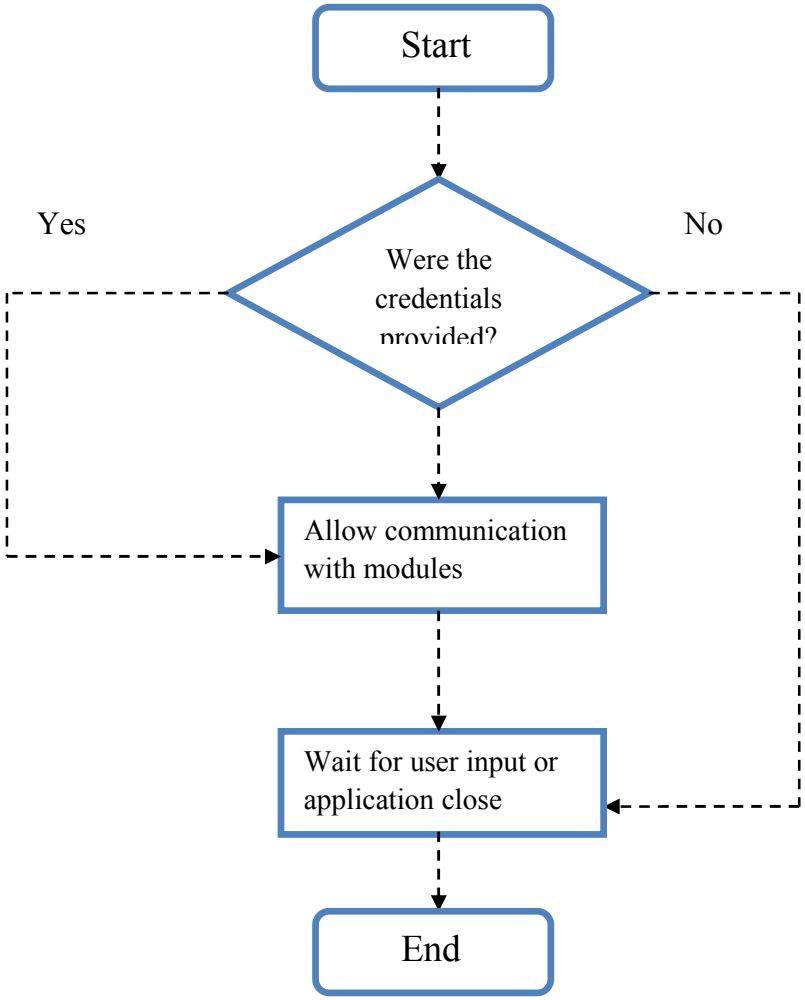

Fig.12 The mobile application flow chart

5.1 SYSTEM TEST

In this chapter, the test of the Smart Lock System is illustrated, including the list of equipment, tests arrangements, the analysis and results. The aim of this phase was to test the operation of both hardware and software parts of the system.

5.1 List of Equipment

In order to test the operation of the system, the following list of equipment was required:

- AVR Universal Board with Atmega32 microcontroller
- 12V DC Power Supply
- Two HC-05 Modules
- USB to UART module to connect the module to the host PC
- Android mobile device with the installed application
- Solenoid
- Three transistors mentioned in the previous part
- LCD Display for easy navigation and status indication
- Breadboard for modules placement and wiring

5.2 System Connection

The connection of the test system is presented in Fig. 10. The additional PCB presented in the Fig.10 is the actual PCB which will drive the work of the lock and modules.

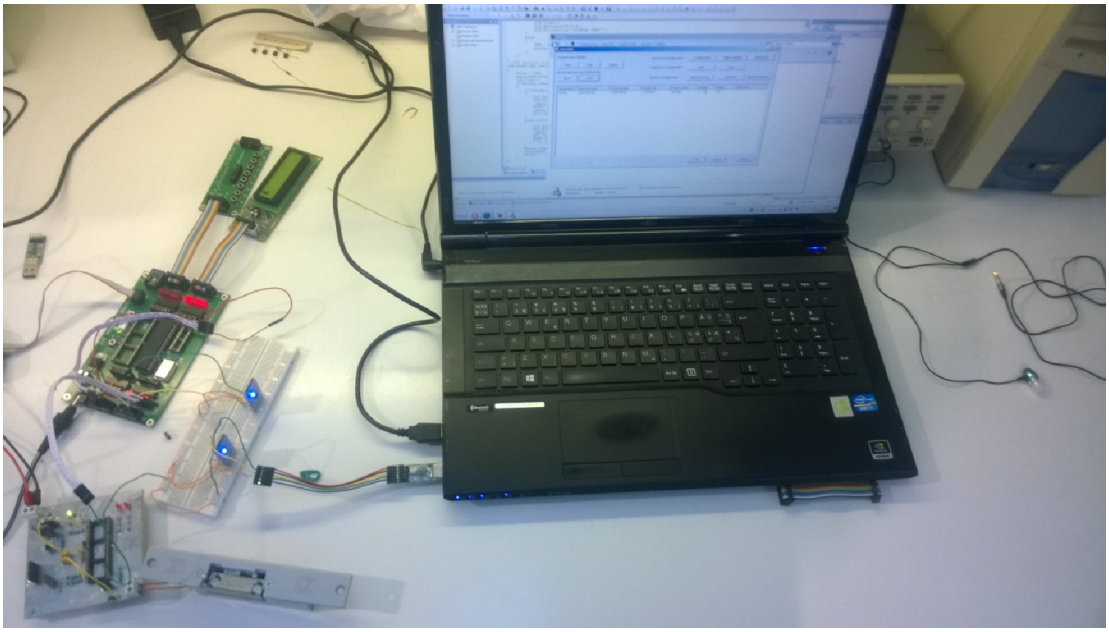

Fig.13 Test System Connection

During the test phase, the solenoid was replaced with an actual electric lock, which itself contains a solenoid and the needed mechanical part for every door. The power requirements for the separate solenoid and the electric lock are the same: 12V DC voltage and 400mA current. The microcontroller was not removed from the board; instead, the grounds of the board and PCB were connected to create a common ground. The Bluetooth module which was named as HC-S was connected to the PC. Another one was connected to the microcontroller communication pins. The power to the module was provided by the development board, the second module was powered by the PC with the ground connection to common one. To power on the PCB, an external 12V DC power supply was used. To display the status of the system and the needed information, 16x2 LCD was connected to the board. The microcontroller output pins were connected to the transistors in order to drive them.

After all connections were made, the system was started up. The LEDs presented on the Bluetooth modules are blinking fast if they are not paired; otherwise, the speed of blinking is decreased. The LCD will display the information only if the events occur, the most of the time it is cleared.

5.3 Test operation and Results

The system test was split into several steps. The first step was to test the data pass between the module and the microcontroller. For this step, the UART Transmit interrupt was disabled in order to not distract the MCU. In this step, both the data pass of the user data and the data used to close the door was tested. After the successful completion of this part, the next part was to test the data pass between the microcontroller and the host PC. For this purpose, the UART receive interrupt was disabled and only UART Transmit interrupt was enabled. The keyboard which was connected to the board was used to generate special events which served as flags for sending data. At this point, the operation of the solenoid was also tested. When the first two tests were completed, the final test was performed. At this point, all the needed features of the system were enabled and it was tested as a whole system. To successfully connect to the module, the user will have to provide the needed data to the Android application, as well as the password for the module. Then, the needed data was sent to test the overall functionality of the system. Firstly, the data needed to access the door was passed. This data was received by both the microcontroller and MFC application successfully and the corresponding data was displayed; solenoid was powered on to open the door. Then, the user data was sent once again to check if its' status was changed. The final step was to check the processing of the data needed to power off the solenoid.

When the user gave the valid credentials to the embedded system, the LCD informed that the Access to the laboratory was granted. When the close message was provided, it informed that the door is closed.

All the three steps of the test operation were completed successfully. The embedded system as well as the MFC application could successfully receive the needed data and to perform the required tasks. The system behaved as it was desired: it gave access to the user if the data which was sent was valid, otherwise, no actions were performed and the data was just skipped. All the valid data was successfully processed and passed as it was designed. The MFC application was tested several times, in order to test its' setup, including loading the specification

and needed data from the file. The networking tests of the application were performed several times before the system tests and they also completed successfully.

6. CONCLUSIONS

The Smart Lock System could run properly and stably. It could implement all the designed and needed features and functions. This could be achieved only by proper software and hardware development, as well as multiple test and debug procedures.

All the planed functions were implemented in the system: the valid data was processed as it had to, the solenoid was successfully driven; all the needed information was displayed in the main window of the application. The system could be easily initialized with just 3 clicks – to initialize the serial communication, networking and the users list. All the data could successfully pass from the mobile device to the server PC.

However, during the development phase some errors occurred. For example, one of them was that the server PC could not receive all the data. The reason for this error was that instead of passing the character array, as it was initially planned, there was done a type conversion from CHAR type to LPCTSTR (Long pointer to constant string), which is 2 times bigger than CHAR type(by bigger I mean the size in bytes). Because of that, after each symbol there was added a space.

In my opinion, Smart Lock System has great potential. It will allow users to forget about their traditional key and to use only their mobile device to get access to the needed are. To reduce the risks, all the possible security measures were taken, including the authorization to the mobile application, the requirement to introduce the password to connect to the BT module. The MFC application can be used to monitor the user activity, which will help authorized personnel to check if the users do what they do and when they do.

However, there is a very big weakness of the system – it has big power requirement. To deal with this issue, a door engineer can do the needed operation to provide the power supply from the normal socket and to not use external battery, which can run out of charge in any moment.

The system will be developed in future to provide more extensions and to be as mobile as possible. For example, in case of loss of the mobile device, there will be connected a keyboard which will be used to provide the valid password to the system, and thus giving access to the area.

A very important step will be to encrypt all the data which will be exchanged. But this step will require the change of microcontroller to a more powerful one, because the encryption/decryption algorithms are very heavy computational tasks, which of course can be done with the current microcontroller, but will be done very slowly.

There is a possibility to increase the speed of operation of the system. However, it requires the change of the microcontroller to one which contains more than one serial communication channel. In addition, it will require one more module, so there will be two paired device and one receiver/slave. In this case, the microcontroller will already react as data buffer and data check point.

Another possibility to increase the speed of operation of the system is to pass the needed data to two modules at the same time. In this case, the microcontroller will not need to forward the data to the application and will have to check only the validity of data.

REFERENCIES

Books

David White, Eugene Olafsen and Kenn Scribner (2000). MFC Programming with Visual C++ 6.0. Moscow, Russia. Diasoft

David J. Kruglinski (2004). Inside Visual C++. Saint-Petersburg, Russia. Russian Redaction and PITER.

Barnett Cox, Barnett O'Cull (2004). Embedded C Programming and the Atmel AVR. Clifton Park, USA. Delmar Cengage Learning

David Kammer, Gordon McNutt, Brian Senese, Jenifer Bray (2002). Bluetooth Application Developer's Guide: The short range interconnect solution. Rockland, USA. Syngress

Electronic publications

Atmel Atmega32 Datasheet

<URL:http://www.atmel.com/images/doc2503.pdf>

HC-06 Bluetooth Module Datasheet

<URL:ftp://imall.iteadstudio.com/Modules/IM120723009/DS_IM120723009.pdf>

Solenoid Information

<URL:https://en.wikipedia.org/wiki/Solenoid>

BD140 Datasheet

<URL:http://www.elektronik-kompendium.de/public/schaerer/FILES/bd140.pdf>

2N3904 Datasheet

<URL:https://www.sparkfun.com/datasheets/Components/2N3904.pdf>

BD139 Datasheet

<URL:http://www.st.com/web/en/resource/technical/document/datasheet/CD00001225.pdf>

AVR Universal Board Technical Information

<URL:http://www.tietopetri.fi>

Transistor as a switching device

<URL:http://www.rason.org/Projects/transwit/transwit.htm>