

Střední průmyslová škola Třebíč

Maturitní práce

Elektronický zámek s RFID

Profilová část maturitní zkoušky

Studijní obor: Informační technologie

Třída: ITB4

Školní rok: 2024/2025 Michal Paulas

Zadání práce

Obor studia: 18-20-M/01 Informační technologie

|  |  |  |
| --- | --- | --- |
| Celé jméno studenta: | **Michal Paulas** |  |
| Třída: | **ITB4** Školní rok: | **2024/2025** |
| Číslo tématu: | **7** |  |
| Název tématu: | **Elektronický zámek s RFID** |  |
| Rozsah práce: | **15 - 25 stránek textu** |  |

Specifické úkoly, které tato práce řeší:

Navrhněte a zrealizujte elektricky ovládaný dveřní zámek pomocí čipu RFID a klávesnice. Zámek umožní vstup buď po načtení autorizovaného čipu nebo po zadání PIN na klávesnici. Po úspěšné autorizaci se sepne relé ovládající zámek a zazní zvukový signál. Navrhněte způsob autorizace čipů (max. 10 čipů). ID čipu a PIN bude uložen tak, aby se neztratil vypnutím napájení. Pro realizaci prostředí AtmelStudio a použijte školní stavebnici.



|  |  |
| --- | --- |
| Termín odevzdání: | **28. března 2025, 23.00** |
| Vedoucí projektu: | **Ing. Ladislav Havlát** |
| Oponent: | **Ing. Jana Veselá** |
| Schválil: | **Ing. Petra Hrbáčková, ředitelka školy** |

ABSTRAKT

Tento projekt představuje návrh a realizaci elektronicky ovládaného dveřního zámku pomocí RFID čipu a klávesnice. Systém uděluje přístup buď naskenováním autorizovaného RFID chipu, nebo zadáním platného PIN kódu, který se nastaví při prvním spuštění zámku. Po úspěšné autentizaci se aktivují ledky indikující otevření zámku. Projekt také zahrnuje metodu pro správu až deseti autorizovaných RFID chipů a zajištění uložení dat pomocí EEPROM, zabraňující ztrátě dat po výpadku proudu. Systém zámku je vyvinut pomocí prostředí Atmel Studio a školní hardwarové sady založené na mikrokontroléru ATmega 644 A. Projekt se zaměřuje na bezpečnost, spolehlivost a snadnost použití, demonstruje základní principy vestavěných systémů a programování mikrokontrolerů.

KLÍČOVÁ SLOVA

RFID zámek, autentizační systém, ATmega 644 A, programování mikrokontrolerů, paměť EEPROM, řízení přístupu

PODĚKOVÁNÍ

Děkuji Ing. Ladislavu Havlátovi za cenné připomínky a rady, které mi poskytl při vypracování maturitní práce.

V Třebíči dne podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval/a samostatně a uvedl/a v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil/a.

V Třebíči dne

podpis autora

Obsah

[Úvod 7](#_Toc192747584)

[1 Použité vývojové prostředí a nástroje 8](#_Toc192747585)

[1.1 Atmel Studio 8](#_Toc192747586)

[1.2 SenderAVR 8](#_Toc192747587)

[1.3 GitHub 8](#_Toc192747588)

[2 Hardware 9](#_Toc192747589)

[2.1 ATmega 644 A 10](#_Toc192747590)

[2.1.1 Klíčové vlastnosti: 10](#_Toc192747591)

[2.2 RFID čtečka a čipy 12](#_Toc192747592)

[2.2.1 Specifikace čtečky 13](#_Toc192747593)

[2.3 Klávesnice 13](#_Toc192747594)

[2.4 LCD display 14](#_Toc192747595)

[2.4.1 Specifikace LCD displeje 14](#_Toc192747596)

[2.5 LED světla 15](#_Toc192747597)

[2.5.1 Specifikace M74HCT245B1 16](#_Toc192747598)

[3 Software 17](#_Toc192747599)

[3.1 Popis kódu 17](#_Toc192747600)

[3.1.1 Definice 17](#_Toc192747601)

[3.1.2 Include 17](#_Toc192747602)

[3.1.3 Definice vlastní 19](#_Toc192747603)

[3.1.4 Přerušení od sériové komunikace 20](#_Toc192747604)

[3.1.5 Přerušení od čítače/časovače 21](#_Toc192747605)

[3.1.6 Funkce na čtení a zápis do EEPROM 21](#_Toc192747606)

[3.1.7 Funkce pro zapnutí vypršení času 21](#_Toc192747607)

[3.1.8 Funkce pro zjištění vypršení času 21](#_Toc192747608)

[3.1.9 Funkce pro zobrazení základního textu na displej 21](#_Toc192747609)

[3.1.10 Funkce pro smazání displeje 21](#_Toc192747610)

[3.1.11 Funkce pro uložení pinu do paměti EEPROM 21](#_Toc192747611)

[3.1.12 Funkce pro první zapnutí zámku 21](#_Toc192747612)

[3.1.13 Funkce pro porovnání pinu 21](#_Toc192747613)

[3.1.14 Funkce pro prvotní nastavení veškerých funkcí 21](#_Toc192747614)

[3.1.15 Funkce pro otevření zámku 21](#_Toc192747615)

[3.1.16 Funkce pro zamítnutí přístupu 21](#_Toc192747616)

[3.1.17 Funkce pro nalezení volného místa v paměti 21](#_Toc192747617)

[3.1.18 Funkce pro uložení čipu 21](#_Toc192747618)

[3.1.19 Funkce pro přečtení čipu pomocí indexu 21](#_Toc192747619)

[3.1.20 Funkce pro zjištění, zda je čip uložený 21](#_Toc192747620)

[3.1.21 Funkce pro přidání čipu 21](#_Toc192747621)

[3.1.22 Funkce pro odstranění čipu 21](#_Toc192747622)

[3.1.23 Funkce pro zjištění, zda byl zámek otevřen pomocí RFID čipu 21](#_Toc192747623)

[3.1.24 Funkce pro ověření uživatele 21](#_Toc192747624)

[3.1.25 Funkce pro restart 21](#_Toc192747625)

[3.1.26 Vstupní bod programu 21](#_Toc192747626)

[3.2 Styl uložení do EEPROM 21](#_Toc192747627)

[3.3 Jak pracovat s RFID zámkem 21](#_Toc192747628)

[Závěr 22](#_Toc192747629)

Úvod

Tento projekt se zaměřuje na návrh a vývoj elektronického dveřního zámku, který využívá technologii RFID spolu s rozhraním klávesnice pro zvýšení bezpečnosti i uživatelského pohodlí. Systém umožňuje přístup dvěma způsoby: buď naskenováním autorizovaného RFID štítku nebo zadáním správného PIN kódu na klávesnici. Jakmile je uživatel úspěšně ověřen, aktivují se ledky, které indikují že byl přístup povolen.

Jádro projektu je postaveno na mikrokontroléru ATmega 644 A, který řídí činnost různých komponent včetně RFID čtečky, klávesnice, LCD displeje a Ledek. Kromě toho se EEPROM používá k ukládání důležitých dat, jako jsou ID štítků RFID a kódy PIN, což zajišťuje uchování dat i v případě výpadku napájení. Vývojovým prostředím použitým pro tento projekt je Atmel Studio se všemi hardwarovými komponenty poskytnutými školou.

Tato práce podrobně popisuje návrh a integraci hardwarových i softwarových prvků a řeší problémy, které se vyskytly během implementace.

# Použité vývojové prostředí a nástroje

## Atmel Studio

Atmel Studio je integrované vývojové prostředí (IDE) určené pro vývoj a ladění aplikací na mikrokontrolérech AVR® a SAM. Poskytuje většinou bezproblémové a uživatelsky přívětivé prostředí pro psaní, vytváření a ladění kódu v C/C++ nebo v assembleru. Atmel Studio také umožňuje import náčrtů Arduino® jako projekty C++, což usnadňuje přechod z prototypování k samotné výrobě a programování reálného produktu.

Je však důležité říct, že Atmel Studio se nedoporučuje pro nové návrhy a nepodporuje některé novější produkty od Microchip. Pro nejnovější funkce a podporu Microchip doporučuje používat MPLAB® X IDE.

## SenderAVR

SenderAVR je program od SPŠT který se využívá pro nahrávání zkompilovaného programu na samotný mikroprocesor. Využívá k tomu USB, to stačí z mikroprocesoru zapojit do počítače, nastavit v programu správný COM a pak už jen nahrát samotný program.

## GitHub

GitHub je široce používaná platforma pro správu verzí, spolupráci a správu kódu, postavená na systému správy verzí Git. Poskytuje cloudové prostředí, kde mohou vývojáři ukládat, sdílet a spravovat své projekty, ať už pracují samostatně nebo v týmech. Díky funkcím, jako jsou úložiště, větvení, sledování problémů a nepřetržitá integrace, GitHub zjednodušuje vývoj softwaru tím, že zajišťuje strukturovaný pracovní postup a bezproblémovou spolupráci. Platforma podporuje veřejná i soukromá úložiště, takže je vhodná pro open-source projekty i pro soukromý vývoj.

GitHub jsem hojně využíval pro verzování, sdílení a ukládání všech souborů a dokumentace souvisejících s projektem. Byl to základní nástroj pro organizaci mé práce, sledování změn v průběhu času a zajištění toho, že všechny projektové zdroje jsou bezpečně uloženy a dostupné odkudkoli. Tímto jsem se nemusel bát jakékoliv ztráty mých dat a samotné práce.

# Hardware

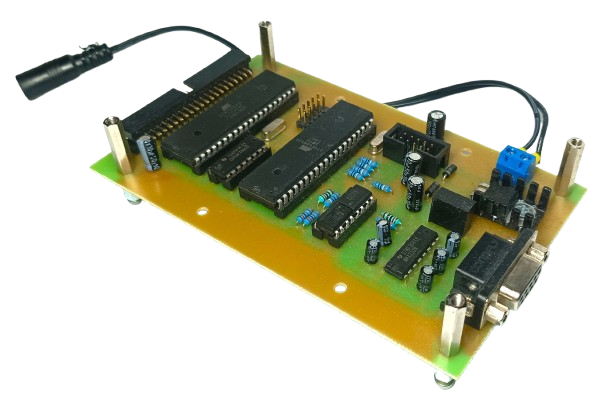
Veškeré HW prostředky mi byly poskytnuty školou. Během prvotního testování jsem narazil na problém s mikroprocesorem ATmega 644 A, na kterém byl zprvu pouze jeden nefunkční pin na PORTC, ale postupem času na něj bylo obtížnější až nakonec nemožné nahrávat jakýkoliv program. Nakonec mi byl mikroprocesor dodán funkční a já mohl bez problému pokračovat ve vývoji RFID Zámku.

HW poskytnutý školou dohromady funguje jako taková stavebnice. Každá z komponent je přidělána na PCB destičce a na ní deseti pinový konektor. Ten je u všech modulů stejný. Tím se pak pomocí rozbočovače, co jde přímo z mikroprocesoru, dají zapojit moduly na potřebné porty. Porty se jmenují PORTA, PORTB, PORTC a PORTD a v projektu jsem využil porty všechny. Na PORTA je připojený LCD display, na PORTB jsou připojené Ledky, na PORTC je připojený KeyPad a na PORTD samotná RFID čtečka.

## ATmega 644 A

ATmega644A je 8bitový CMOS mikrokontroler založený na architektuře AVR® RISC, určený pro vysoce výkonné a nízkoenergetické aplikace. V tomto případě je k mikroprocesoru připojený externí krystal pro přesnější časování mikroprocesoru. Použitý krystal kmitá ve frekvenci 11059200 Hz.

V projektu byl mikroprocesor použit jako mozek celého projektu. Využíval jsem na něm funkce jako I2C pro komunikaci s LCD displejem, USART ke komunikaci s RFID čtečkou a čítač/časovač pro umožnění programu mít funkci time-out která přeruší různé akce po určitém čase.



Obrázek Hlavní modul s ATmega 644A

### Klíčové vlastnosti:

Paměť a úložiště:

* 64KB programové paměti Flash
* 2KB EEPROM
* 4KB SRAM

Zpracování a výkon:

* Pokročilá architektura RISC se 131 výkonnými instrukcemi
* Propustnost až 20 MIPS při 20 MHz
* 32×8 univerzálních pracovních registrů

Periferní rozhraní:

* Dva 8bitové časovače/čítače a jeden 16bitový časovač/čítač
* Šest PWM kanálů
* 8kanálový 10bitový ADC s volitelným zesílením
* Dvě rozhraní USART, SPI a I2C
* Podpora JTAG pro ladění a programování

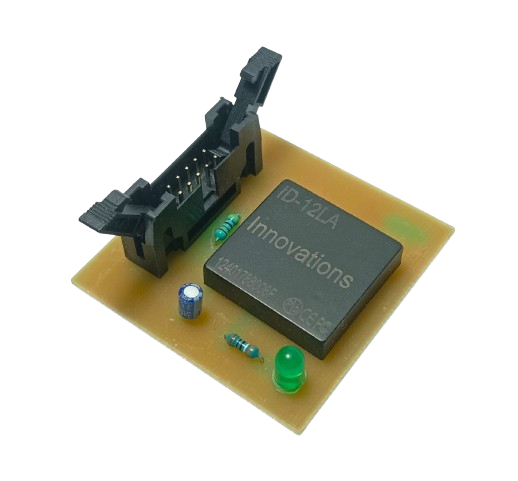
Energetická účinnost:

* Pracuje při napětí mezi 2,7V a 5,5V
* Má několik režimů úspory energie jako power-down a standby režimu

## RFID čtečka a čipy

Jako RFID čtečku byl modul již zmíněné stavebnice, na tomto modulu se nachází čtečka ID-12LA od společností SparkFun Electronics. Jedná se o snadno použitelnou čtečku, která umožnuje přijímat a přenášet na master zařízení ID naskenovaného chipu. Čip komunikuje skrze rozhraní USART, tudíž má výstupy RX a TX.

Jako chipy pro RFID jsem použil školní chip, jelikož pracuje na frekvenci, jakou RFID čtečka dokázala přečíst. Dále se také dal použít průkaz ISIC, jelikož disponuje tímto typem čtečky.



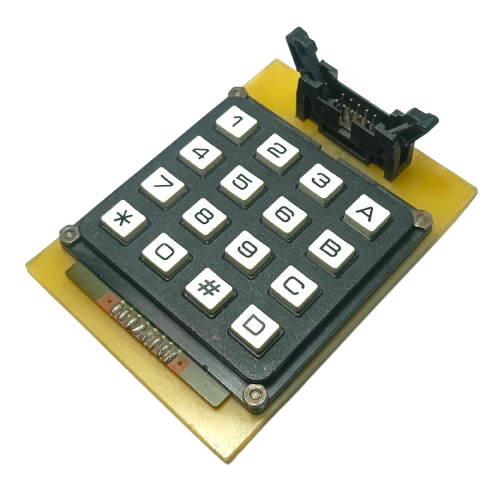
Obrázek Modul s RFID čtečkou

### Specifikace čtečky

* Napájecí napětí: 2,8 V - 5 V
* Nosná frekvence: 125 kHz
* Rozsah čtení: až 120 mm
* Standard: EM4001 ISO RFID IC
* Komunikace: sériové rozhraní TTL a RS232 - 9600 bps
* Vestavěná anténa
* Rozteč pinů: 2 mm
* Rozměry: 25 x 26 x 7 mm

## Klávesnice

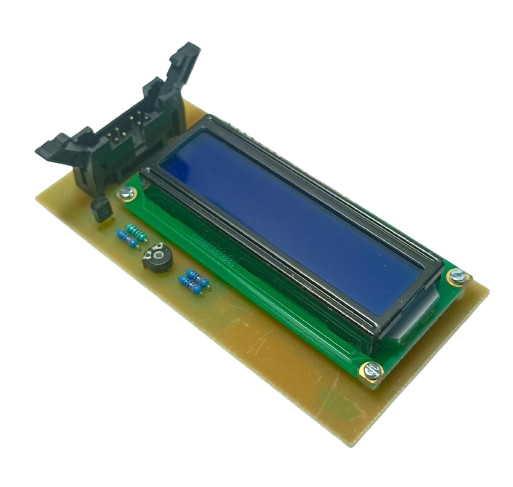
Pro klávesnici byla použita 4x4 klávesnice. Ta funguje na principu čtyř a čtyř vodičů položených na sebe pod úhlem devadesátí stupňů a tím jak se dva vodiče spojí tak to mikroprocesor zaregistruje. Na klávesnici se nachází celkem 16 tlačítek, tlačítka na sobě mají buďto čísla, písmena a znaky. Čísla od 0 až 9, písmena od A do D a dva znaky \* a #.



Obrázek Modul klávesnice 4x4

## LCD display

Jako display byl použit standardní LCD display 16x2 s dvěma řádky a šestnácti poli pro znaky. Display disponuje modrým podsvícením. Na modulu je také obvod který umožnuje regulovat intenzitu podsvícení pomocí potenciometru. Modul komunikuje skrze I2C komunikaci, ta je u modulu dána na piny SDA – B7, SCL – B6 a PWR – B0.



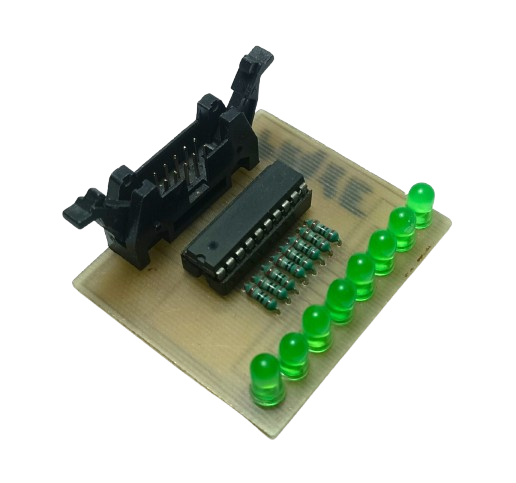
Obrázek Modul s LCD Displejem

### Specifikace LCD displeje

* Rozměry modulu: 80 x 35 x 11 mm
* Velikost samotné informační plochy: 64,5 x 14 mm
* LCD displej: s modrým podsvícením
* Široký pozorovací úhel a vysoký kontrast
* Průmyslový standard: HD44780
* Napájení: 5 V DC
* Pod světlení: A, K je na 5 V, stabilizátor je již na DPS displeje

## LED světla

Pro indikaci, zda je zámek otevřený či zavřený byl použit modul s ledkami. Ten disponuje osmi ledkami, každá na 5 V a budič sběrnice M74HCT245B1. Ten využívá pouze tří pinů z portu, na který se připojí a dokáže z toho různě rozsvěcovat ledky na modulu.



Obrázek Modul s ledkami

### Specifikace M74HCT245B1

* Budič
* Má 8 výstupních kanálů
* V pouzdře DIP20
* Montáž THT

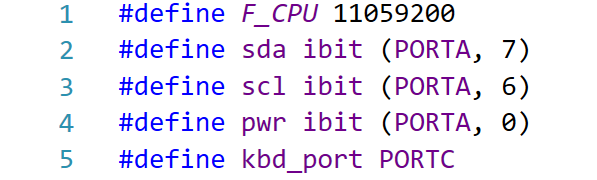
# Software

Program pro RFID čtečku je psaný v programovacím jazyce C++. Psaný byl ve vývojovém prostředí Atmel Studio, které bylo nastavené pro práci s mikroprocesorem ATmega 644 A. V programu jsem využil Přerušení, jak od časovače, tak od sériové komunikace, která byla využita pro komunikaci s RFID čtečkou. Dále byla využita EEPROM paměť pro ukládání čipů, pinu a stavu v jakém se čtečka nachází. Také jsme využil Watchdog Reset Flagu pro restartování mikroprocesoru, to bylo využito ve chvíli kdy uživatel restartuje pin.

## Popis kódu

### Definice

Na začátku programu se nachází definice. Nastavuji zde frekvenci mikroprocesoru, který má externí krystal. Pro upřesnění je to frekvence 11059200 Hz. Dále zde definuji SDA (Serial Data), SCL (Serial Clock) a PWR (Power) pin. Tyto piny využívá použitý LCD display pro komunikaci skrze I2C a napájení. Nastavují se na PORTA, a to přesně na jeho piny 7, 6 a 0. Na další definici nastavuji kbd\_port na PORTC. Kbd\_port je proměnná z knihovny která je zde využita modulem Klávesnice.



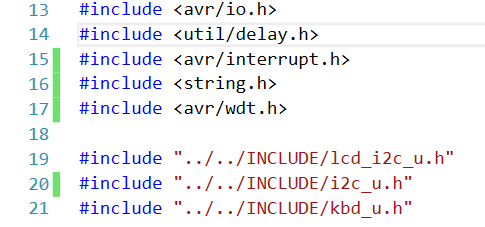
Obrázek Ukázka definicí v kódu

### Include

Využívám zde dva typy include. První, který se pozná použitím těchto závorek: <>, využívá knihovny, které jsou přístupné od samotného Atmel Studia. Poté druhý typ, který používá uvozovky. Tento se využívá pro využití knihoven, které jsou uloženy na disku v počítači. Do těchto uvozovek se pak napíše cesta, kde je knihovna uložena.

Jako první se zaměříme na include, které jsou přístupné od samotného Atmel Studia. Automaticky je zde přidaný include: #include <avr/io.h>, toto je základní knihovna, která se zde používá pro práci se samotným mikroprocesorem, jeho registry atd... Dále je zde použita knihovna: #include <util/delay.h>. Tato knihovna je zde využita pro funkci: „\_delay\_ms()“. Tato funkce způsobí že se na požadovaný čas v milisekundách mikroprocesor zastaví. Poté je zde využita knihovna: #include <avr/interrupt.h>. Jelikož je v kódu použito přerušení od čítače/časovače a od sériové komunikace, tak zde tato knihovna musí být přidána pro zajištění funkčnosti přerušení. Další include je: #include <string.h>. Tato knihovna je zde použita pro práci s textovými řetězci. A nakonec je tu include: #include <avr/wdt.h>, ten přidá knihovnu, který nám umožní využívat WatchDogTimer pro reset mikroprocesoru.

Další tu jsou include, které využívají externí soubory jako knihovny. Všechny zde použité knihovny jsou dodané od školy, autorem většiny těchto knihoven je Ladislav Havlát. Aby byl kód funkční na každém zařízení s Atmel Studiem bude potřeba upravit cestu k těmto souborům. Prvním z include je: #include "G:\\MIR\\INCLUDE\\lcd\_i2c\_u.h". Tato knihovna přidá funkce pro práci s LCD displejem, který komunikuje pomocí I2C. Další include: #include "G:\\MIR\\INCLUDE\\i2c\_u.h" umožní samotnou komunikaci skrze I2C. Poslední include: #include "G:\\MIR\\INCLUDE\\kbd\_u.h" umožňuje správnou práci s připojenou klávesnicí.

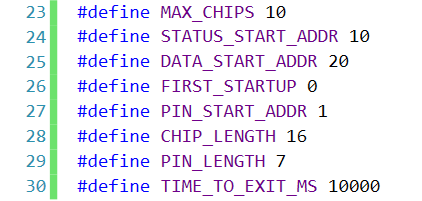


Obrázek Ukázka include z kódu

### Definice vlastní

Pro ulehčení následné modularity kódu jsme veškeré důležité hodnoty, které jsou limitované zadáním také vložil do definice. Pro příklad by tento mikroprocesor dokázal uložit přes 30 různých RFID čipů.

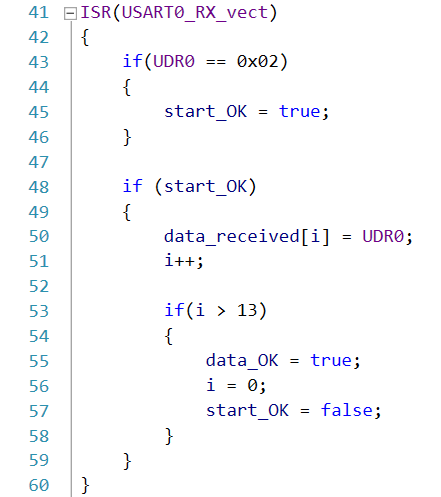
* MAX\_CHIPS – Maximální počet čipů, které mohou být přidány.
* STATUS\_START\_ADDR – Adresa, na které se začíná ukládat status každého čipu, zatím se využívá pouze že je slot zabraný nebo volny, ale pokud by se na zámku dále pokračovalo, bylo by možné implementovat funkce které by mohli nějaký čip pozastavit ve funkčnosti a nemazat, nebo cokoliv jiného.
* DATA\_START\_ADDR – Adresa, na které začíná ukládání samotných čipů do EEPROM paměti.
* FIRST\_STARTUP – Hodnota, co nabývá buďto 0 nebo 255 indikující, jestli je zámek zapnut poprvé. Pokud ano, tak zámek požádá o zadání pinu.
* PIN\_START\_ADDR – Adresa, kam se uloží pin, jeho délka je 6 znaků a jeden ukončovací znak.
* CHIP\_LENGTH – Určuje délku samotného čipu, u většiny je 16, ale pro jistotu se to stále dá pozměnit.
* PIN\_LENGTH – Délka pinu, samotný pin je dlouhý na 6 a je zde přidaný ukončovací znak, takže je celková délka 7.
* TIME\_TO\_EXIT\_MS – Čas v milisekundách který určí jak dlouho má uživatel pro dokončení akce. Po tomto čase se ukáže že byla akce přerušena a zámek se vrátí do základního stavu.



Obrázek Ukázka vlastních definicí

### Přerušení od sériové komunikace

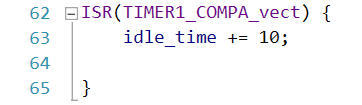
Toto přerušení se zavolá kdykoliv přijde nový bajt. Kód v přerušení slouží k uložení veškerých dat do proměnné: data\_recived, jedná se o pole proměnných typu char, což jsou znaky. Jako první se v kódu ptáme na registr UDR0, jestli má hodnotu 0x02, což značí že se začal číst čip. Pokud ano, tak se proměnná start\_OK typu bool nastaví na true. Dále je zde podmínka, která se ptá, jestli je již zmiňovaná proměnná start\_OK nastavena na true. Pokud to platí tak postupně další přečtené bajty ukládáme do pole data\_recived. V této podmínce je také ještě podmínka, která hlídá, jestli už byl čip kompletně přečtený a pokud ano tak nastaví proměnnou data\_OK na false a vynuluje celočíselnou proměnou i, která počítala počet kolik bajtů už bylo přečteno.



Obrázek Ukázka přerušení od sériové komunikace

### Přerušení od čítače/časovače

Vyvolá se pokaždé když má čítač/časovač stejnou hodnotu jako registr OCR1A. Je to nastaveno, aby se to stalo každých 10ms. Pokaždé co se přerušení vyvolá tak se do celočíselné proměnné idle\_time přičte 10. To slouží k počítání času kdy nastane vypršení již dříve nastaveného času v milisekundách.

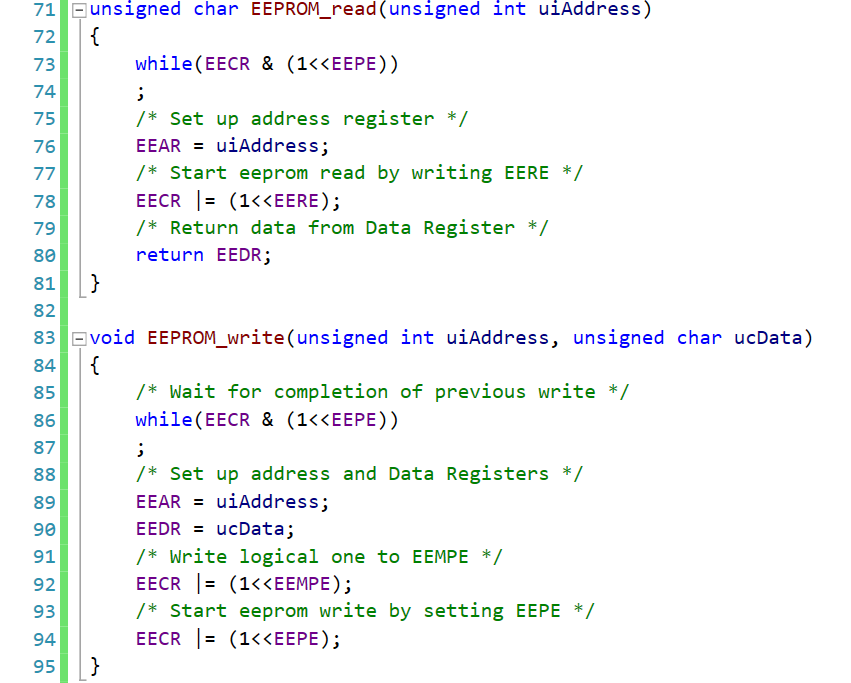


Obrázek Ukázka přerušení od čítače/časovače

### Funkce na čtení a zápis do EEPROM

Tyto funkce byly celé převzaty z dokumentace mikroprocesoru ATmega 644 A. V oběma funkcích prvně počkáme, dokud se nedokončí předchozí zápis, to zjistíme tím, že je v registru EECR namaskovaná logická 1 na místě EEPE. Dokud je to pravda, tak je program zaseklý v cyklu. V samotné funkci pro čtení poté uložíme do registru EEAR adresu, ze které chceme číst, ta je celočíselná, nezáporná proměnná. Poté pro začátek čtení z EEPROM namaskujeme v registru EECR logickou 1 na místo EERE. Nakonec vrátíme proměnnou typu unsigned char, v niž je hodnota registru EEDR, ve kterém je osmi bitová hodnota.

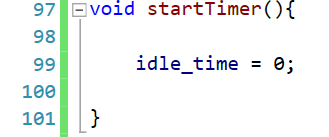
Funkce pro zápis po vyčkání na dokončení předchozího zápisu uloží adresu na kterou chceme psát do registru EEAR. Dále uloží data která chceme zapsat do registru EEDR, ukládá se zde proměnná typu unsigned char. Dále namaskuje logickou 1 do registru EECR na místo EEMPE. Nakonec namaskuje logickou 1 na místo EEPE což započne zápis do EEPROM.



Obrázek Ukázka funkcí pro čtení a zápis do EEPROM

### Funkce pro zapnutí vypršení času

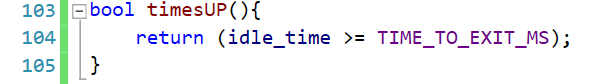
Logika pro celkový systém vypršení času spočívá v proměnné idle\_time, ta je typu unsigned int. Jak bylo již zmíněno v části o přerušení od čítače/časovače, přidáváme k ní každých 10ms číslo 10. V této funkci její hodnotu nastavíme na nulu a bude tím tak počítat čas od zavolání této funkce. Používá se kdykoliv je potřeba použít funkci pro vypršení času, ta je popsaná v následující části.



Obrázek Ukázka funkce pro zapnutí vypršení času

### Funkce pro zjištění vypršení času

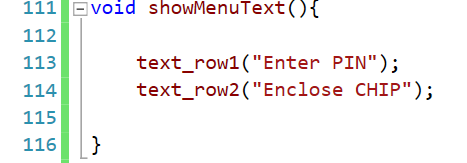
Tato funkce vrací proměnnou typu bool. Pokud se již uplynulý čas do zapnutí vypršení času rovná, nebo je větší než ve vlastních definicích TIME\_TO\_EXIT\_MS tak vrátí true.



Obrázek Ukázka funkce pro zjištění vypršení času

### Funkce pro zobrazení základního textu na displej

Aby bylo v hlavní části jednoduší přepisovat displej tak vznikla tato funkce. Kdykoliv se zámek zapne, když už má uživatel nastavený pin tak se na displeji musí zobrazit instrukce pro uživatele co má vlastně dělat a tato funkce to zařídí. Používá funkce z knihovny pro LCD displej které zapíší na display požadovanou posloupnost znaků.



Obrázek Ukázka funkce pro zobrazení základního textu na displej

### Funkce pro smazání displeje

Pro jednoduší mazání displeje. Používá stejné funkce z knihovny pro LCD displej, ale místo textu do něj vkládá prázdno.

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro smazání displeje

### Funkce pro uložení pinu do paměti EEPROM

Na vstupu je konstantní pole znaků což je pole typu char. Konstantní znamená že se ve funkci daná proměnná nemůže upravovat. Tím zajistíme že by se v budoucnu nemělo stát, aby se nijak neupravoval pin, co se musí uložit. Ve funkci se nachází cyklus for, který bude probíhat, dokud bude celočíselná proměnná i menší než již dříve zapsaná definice PIN\_LENGTH. Pokaždé co se projede cyklem se proměnná i zvětší o jedničku a tím se pak také samotný kód v cyklu řídí. Ten využívá této proměnné pro zapsání celého pole postupně do EEPROM paměti pomocí funkce EEPROM\_write.

Obsah obrázku text, snímek obrazovky, Písmo, řada/pruh

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka kódu pro uložení pinu do EEPROM paměti

### Funkce pro první zapnutí zámku

Na začátku funkce se vytvoří proměnná typu pole charů o velikosti 7 pro uložení budoucího pinu co se zadá. Poté nezáporná celočíselná osmibitová proměnná „index“ která bude určovat na jaké místo v poli se budou zadané znaky zapisovat, a nakonec proměnnou typu char „key“ do které se vždy uloží jaký znak byl zadán.

Zavolá se funkce „clearDisplay“ pro smazání obsahu displeje a poté se zapíše na vrchní řádek displeje výzva pro zadání pinu.

Následně je cyklus, který se ukončí až poté co uživatel zadá a potvrdí pin. V cyklu zavoláme „kb\_on\_timer1()“, je to funkce potřebná pro správné fungování klávesnice. Následuje podmínka, ve které se ptáme, jestli byla stisknuta klávesa. Pokud ano tak se uloží tento znak do proměnné „key“.

V další podmínce se poté ptáme, o jaký se jedná znak a podle toho pak program reaguje. Když se jedná o číslo a zároveň ještě nebyl zapsán požadovaný počet znaků, tak se uloží na pozici určenou proměnnou „index“ do pole. Proměnná index se poté zvětší o jedno a vypíše se celý obsah pole na displej. Pokud už se jedná o poslední znak, co se zadává, tak se na konec do pole vloží ukončovací znak.

Při další podmínce se ptáme, jestli není znak \*, což smaže jeden zadaný znak z pole, zmenší proměnnou „index“ o jedno a opět vypíše na displej.

Nakonec se pak v podmínce ptáme, jestli je zadaný znak D a zároveň je už zadaný dostatečný počet znaků. V tomto případě pak uložíme zadaný pin do pole „pristupovyPin“ což bude následně použito pro odemčení zámku pomocí pinu. Dále se uloží pomocí funkce „savePin“ pin. Nakonec už se pouze do EEPROM paměti na adresu FIRST\_STARTUP, která byla na začátku kódu definována, uloží že byl zámek už jednou zapnut, tudíž má uložený pin a vypíše se zpráva pro uživatele na displej, že byl pin uložen.

Obsah obrázku text, snímek obrazovky, dokument, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce FirstStartUp

### Funkce pro porovnání pinu

Jedná se o funkci vracející bool hodnotu a na vstupu má textový řetězec ve kterém je pin co uživatel zadal pro přístup do zámku. Uvnitř je podmínka, která pomocí funkce „strcmp“ z knihovny string.h porovnává dva textové řetězce, jestli jsou stejné. Pokud ano tak funkce vrátí true, naopak false.

Obsah obrázku text, Písmo, řada/pruh, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro porovnání pinů

### Funkce pro prvotní nastavení veškerých funkcí

Nastavují se zde veškeré registry, porty, inicializace komponent atd... Jako první se nastavují registry pro nastavení přerušení od příjmu na serial komunikaci. To je důležité pro funkčnost komunikace s RFID čtečkou. Také se nastaví registry pro čítač/časovač, aby každých 10 milisekund nastalo přerušení, že se hodnota v registru OCR1A rovná hodnotě čítače/časovače.

Následně se nastavují porty A a B na 255 což znamená logická 1, DDRA na logickou nulu a DDRB na logickou 1. To znamená že je DDRA nastavený pro vstup a DDRB pro výstup. Tímto se zajistí správná funkčnost ledek a LCD displeje.

Poté jsou inicializace veškerých komponent. Inicializuje se klávesnice a i2c komunikace pomocí funkcí „kb\_init“ a „i2c\_init“. Tyto funkce jsou z vložených knihoven pro i2c komunikaci a klávesnici. Nastaví se zde také pro displej pwr pin. Po tomto všem se zavolá funkce „sei“ která globálně povolí přerušení, takže veškeré nastavení registrů na přerušení bude funkční.

Nakonec se v podmínce zavolá funkce „i2c\_open\_wr“ která vezme hardwarovou adresu LCD displeje a zkusí s ním komunikovat. Pokud je to úspěšné tak se display vyčistí a problikne.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro prvotní nastavení

### Funkce pro otevření zámku

Jelikož se pouze jedná o ukázku zámku tak zde není reálný zámek, ale pouze ledky, které ukazují, jestli je zámek otevřen nebo ne. Funkce prvně využije funkci „clearDisplay“ ta displej smaže. Poté nastaví PORTB na logickou nulu, takže se ledky rozsvítí. Dále se na displej vypíše zpráva pro uživatele, že byl zámek otevřen. Program počká jednu vteřinu a poté ledky znovu zhasne nastavením logické 1 na PORTB. Nakonec se zavolá funkce „showMenuText“ a program se vrátí na hlavni obrazovku.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro otevření zámku

### Funkce pro zamítnutí přístupu

Jelikož je zámek normálně vždy zamčený tak se ve funkci pouze vyčistí displej funkcí „clearDisplay“, vypíše se na displej zpráva, že byl přístup zamítnut, program počká vteřinu a vrátí se na hlavní obrazovku.

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro zamítnutí přístupu

### Funkce pro nalezení volného místa v paměti

Ve funkci je pouze jeden cyklus, který projede všechny možné pozice pro uložení čipu. Cyklus jede tak dlouho dokud je proměnná typu „int“ menší než dříve nadefinovaná hodnota MAX\_CHIPS. V cyklu se poté nachází podmínka, ve které se ptáme, jestli přečtená hodnota z EEPROM z definované adresy STATUS\_START\_ADDR a k ní přidaná dříve zmírněná proměnná typu „int“ je rovna 255 (0xFF je zapsáno hexadecimálně). To znamená že je místo volné pro zapsání čipu a funkce vrátí hodnotu v dříve zmíněné proměnné „int“. Pokud nebylo žádné volné místo nalezeno tak se navrátí hodnota -1.

Obsah obrázku text, snímek obrazovky, Písmo, řada/pruh

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro nalezení volného místa v paměti

### Funkce pro uložení čipu

Funkce má dva parametry. Pole charů, ve kterém je uložený RFID čip, který je potřeba uložit a proměnou slot typu „int“ která říká do jakého slotu se má uložit RFID čip. Jako první přepíšeme adresu statusu daného slotu na 0, což bude indikovat že byl slot zabrán. Poté se vypočítá z proměnné slot adresa, na kterou máme začít ukládat RFID čip. Samotný čip se poté uloží následujícím cyklem, který z pomocí vypočítané adresy postupně zapíše všech šestnáct charů z pole. Pro ukázku se na konci smaže displej a na chvíli se na něj vypíše celé pole obsahující RFID čip.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro uložení čipu

### Funkce pro přečtení čipu pomocí indexu

Funkce má opět dva parametry, proměnnou slot typu int a pole charů, ale to zde slouží jako buffer pro čip, který se přečte. Dále funkce vrací proměnnou int, která bude indikovat co se ve funkci stalo. Jako první se podmínkou zjistí, zda náhodou není zadaný slot mimo hranici. To by se nemělo stát, ale pokud by byla omylem zadaná jiná hodnota, tak aby to nedělalo problémy. Když to nastane tak se vrátí 2.

Dále se podmínkou ptáme, jestli je na daném slotu vůbec nějaký čip uložený, pokud není tak funkce vrátí 1.

Z proměnné slot se dopočítá adresa, ze které budeme v následujícím cyklu číst čip. V cyklu opět projedeme všech šestnáct pozic a postupně je uložíme do buffer. Nakonec ještě na poslední pozici buffer vložíme ukončovací znak a vrátíme 0, což indikuje že vše proběhlo v pořádku.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro přečtení čipu

### Funkce pro zjištění, zda je čip uložený

Funkce vrací int značící, jestli je čip uložený, nepoužívá se zde bool, kdyby byly náhodou do funkce přidány další ověření a vracela by poté další hodnoty. Jako parametr je pole charů, ve kterém je uložený RFID čip. V samotné funkci se založí další pole charů délkou určenou definicí CHIP\_LENGTH. Následuje cyklus, který projede veškeré sloty kde mohou být uloženy čipy. V cyklu se nachází podmínka, která volá funkci „readChip“ která vrátí 0 pouze když daný čip nalezla a do buffer uloží přečtený čip. Zároveň ale musí funkce z knihovny „string.h“ „strcmp“ vrátit nulu, to se stane, když se buffer rovná čipu, který jsme zadali. Pokud vše sedí tak se vrátí 1. Pokud ne tak se vrátí 0.

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek Ukázka funkce pro zjištění, zda je čip uložen

### Funkce pro přidání čipu

### Funkce pro odstranění čipu

### Funkce pro zjištění, zda byl zámek otevřen pomocí RFID čipu

### Funkce pro ověření uživatele

### Funkce pro restart

### Vstupní bod programu

## Styl uložení do EEPROM

## Jak pracovat s RFID zámkem

Závěr