PRAXE

8	SPŠ CHOMUTOV	Dengler Ondřej
1.2.2022	Trezor	V4

ZADÁNÍ:

Vytvořte program pro ovládání trezoru. Program bude v jazyce C v AVR.
Program musí:

- Umožnit zadávat kód
- Ukázat momentálně zadaný kód
- Otevřít trezor, pokud je zadaný správný kód
- Změnit heslo, když je otevřený trezor

TEORIE:

Nejdříve si propojíme trezor s naším mikrokontrolerem tak, že propojíme prvních 5 portů na PORTB pro ovládání znaku a na PORTD první 4 porty pro ovládání pozice a 5. pro ovládání dveří a nakonec na PORTF si připojíme na první port čtení klávesnice, kde můžeme použít pouze první 4 porty. Příkazem DDR(Data Direction Register) si nastavíme PORTB a PORTD jako výstupní porty (DDRB/D = 0xFF) a PORTF jako vstupní (DDRF = 0x00). Poté si připojíme mikrokontroler s PC přes programátor JTAG.

Kód se kompiluje pomocí programu Programmers Notepad. Dále je potřeba MakeFile, což je soubor s určitým formátováním, který si můžeme vytvořit pomocí MakeFile generátoru a musíme ho nastavit podle naších požadavků a ty jsou: MCU TYPE: atmega128, Programátor: jtagmkI, Port: sice usb, ale musíme ho nastavit jako sériovou komunikaci(com) a číslo zjistíme ve správci zařízení v portech, frekvence: 14745600Hz.

K samotnému trezoru musíme umět pracovat s porty. Nastavení hodnoty na portu je například PORTD = 0xFF (pošleme na všechny porty na PORTD jedničky) a čtení je jednoduše stav = PORTD. K pracování s porty používám bitové operace, kde vynulování bitu je clearBit(stav bitu, pozice bitu) pro nastavení 1 setBit(stav bitu, pozice bitu) a zjištění, zda je bit 0 bitIsClear(stav bitu, pozice bitu), který vrátí 1 pokud je na pozici bitu 0.

Poté musíme chápat, jak se pracuje s jednotlivými tlačítky a displejem. Ovládají se otevíráním tranzistorů. Vždy pouze jeden tranzistor může být otevřen pro správnou funkčnost. Vykreslování kódu potom funguje, tak že otevíráme a zavíráme všechny 4 tranzistory po sobě s 1 ms zpožděním. Pozici zmáčknutého tlačítka, zjistíme podle toho, v jaký cifře kódu zrovna byl. Když otevřeme dveře, musíme do nich trochu ťuknout, aby se opravdu otevřeli.

POPIS PROGRAMU:

Po spuštění programu se zapne rutina setup(), která nám nastaví porty dle potřeby tj. PORTB a PORTD jako výstupní a PORTF jako vstupní. Poté se nám vytvoří pole pro zadaná čísla ({1,1,1,1}) a pole pro heslo ({1,1,1,1}).

Potom se program dostane nekonečné smyčky while, ve které program kontroluje, zda je tlačítko zmáčknuté a pokud ano, tak na jaké cifře. Pokud na první zvýší naší momentální pozici o 1 (v případě, že je naše pozice 3 přepne se na 0), pokud na druhé zvýší se číslo na cifře na které momentálně jsme o 1(Pokud 9 přepne se na 0). U třetí cifry se číslo na cifře sníží o 1(Pokud 0 přepne se na 9). U poslední čtvrté cifry program zkontroluje pole zadaných čísel s polem hesla a pokud jsou tyto pole identická, tak otevře dveře od trezoru. Tlačítko taky znázorní na displeji „animaci“ kontroly hesla.

Tady by potom ještě byla změna hesla, která by měla podmínku otevřených dveří. Heslo by se naklikávalo stejně jako při zadávání kódu a 4. tlačítko by heslo potvrdilo.

Na konci smyčky je obstaráváno vykreslování kódu na displeji. Vždycky se vypíše znak a cifra se zvýší o 1 (U třetí cifry se přepíše na 0) a poté je krátký 1 ms zpoždění.

ROZBOR PROMĚNNÝCH A FUNKCÍ:

main.c
Proměnné:

Typ	Název	Popis
int	kratkyDelay	Zpoždění 1 ms
int	delsiDelay	Zpoždění 250 ms
int	nasePozice	Pozice cifry na které jsme
int	potvrzeni	Počet stejných cifer hesla
int[]	zadany	Námi zadané cifry
int[]	heslo	Pole s heslem

Funkce:

Typ	Název	Argumenty	Popis
int	main		Hlavní tělo programu
void	setup		Nastaví porty
void	znak	int cislo	Vykreslí na displej číslo
void	cifra	int pozice	Nastaví pozici na displeji
void	zmenaCifry	int momentalniPozice	Posune pozici o 1

bitFunkce.c
Funkce:

Typ	Název	Argumenty	Popis
int	setBit	value, bitno	Nastaví daný bit na 1
int	clearBit	value, bitno	Nastaví daný bit na 0
void	toggleBit	value, bitno	Změní stav daného bitu
int	bitIsSet	value, bitno	Vrátí true pokuď je daný bit v log. 1
int	bitIsClear	value, bitno	Vrátí true pokuď je daný bit v log. 0

VÝVOJOVÝ DIAGRAM:

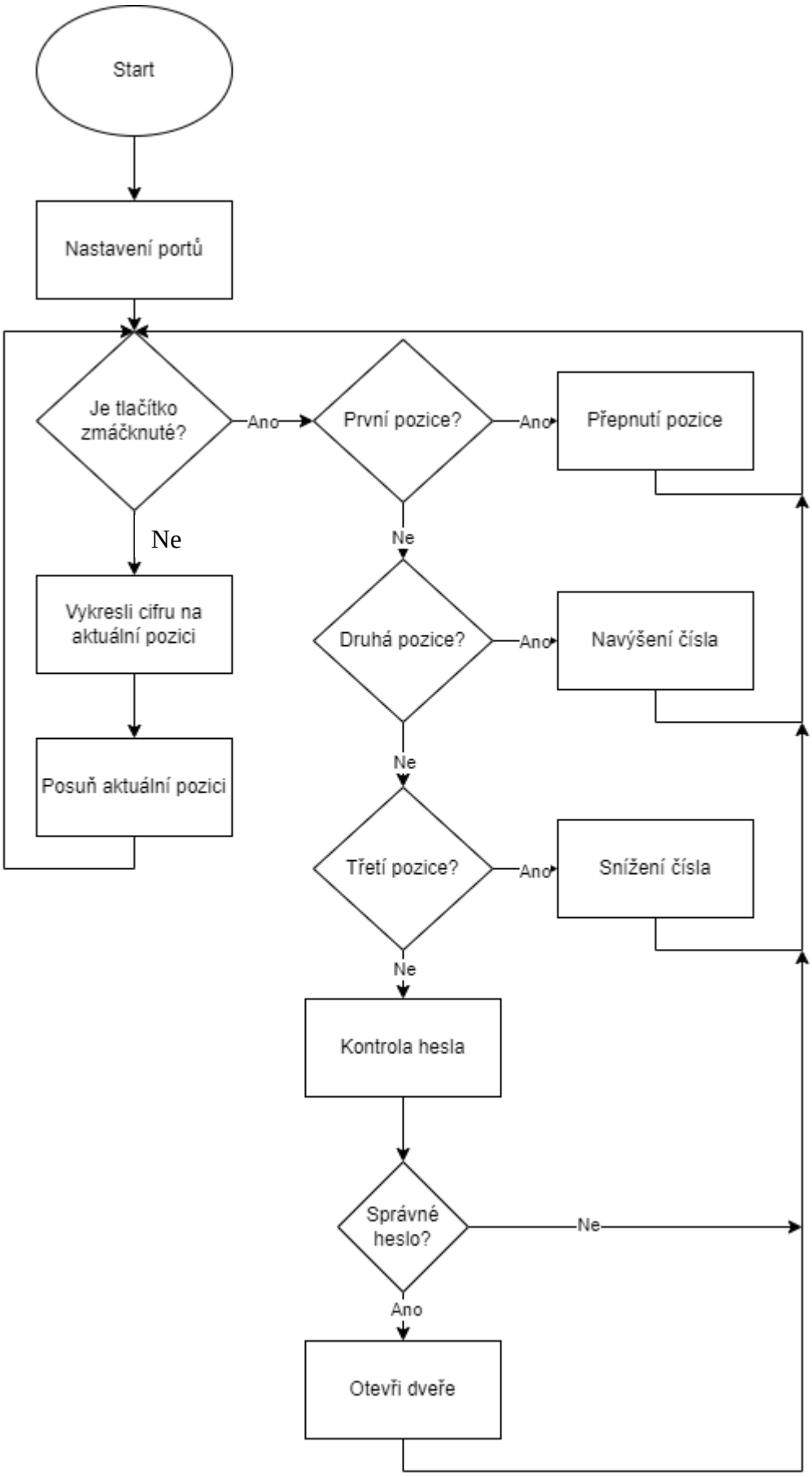
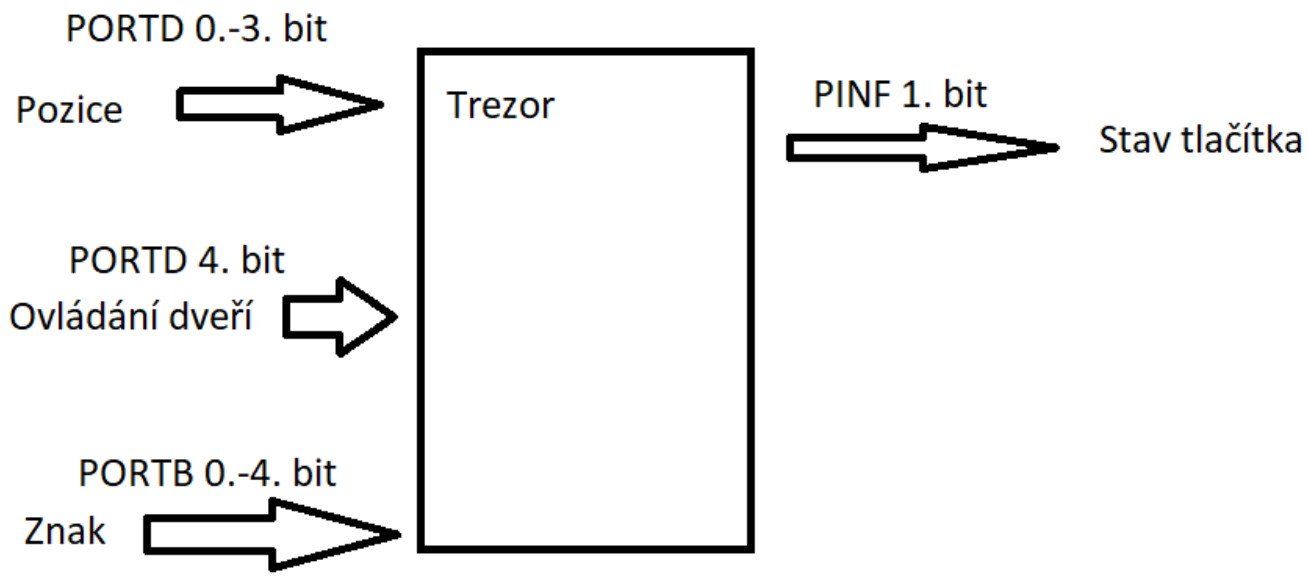


SCHÉMA ZAPOJENÍ:



ZDROJOVÝ KÓD:

main.c

```
#include <avr/io.h>
#include <stdio.h>
#include <util/delay.h>
#include "bitFunkce.h"

void setup(void){
	DDRB = 0xFF;
	PORTB = 0x00; //Ovládání znaku
	DDRD = 0xFF;
	PORTD = 0x00; //Ovládání pozice+Dvere
	DDRF = 0x00; //Klávesnice
	_delay_ms(50);
}

void znak(int cislo){
	PORTB = cislo;
}

void cifra(int pozice){
	PORTD = pozice;
}

void zmenaCifry(int momentalniPozice){
	int pozice = 0xFF;
	if(momentalniPozice == 4){
		pozice = clearBit(pozice,0);
		cifra(pozice);
	}
	else{
		pozice = clearBit(pozice,momentalniPozice);
		cifra(pozice);
	}

}
```

```c
int main(void){
	setup();
	int kratkyDelay = 1;
	int delsiDelay = 250;
	int nasePozice = 0;
	int potvrzeni = 0;
	int zadany[4] = {1,1,1,1};
	int heslo[4] = {1,1,1,1};


	while(1){
		for(int i = 0;i<4;i++){
			if(bitIsClear(PINF,1)){ //Zjištění zda je tlačítko zmáčknuto
				if(i == 1){ //První tlačítko
					_delay_ms(delsiDelay);
					if(nasePozice == 3){
						nasePozice = 0;
					}
					else{
						nasePozice++;
					}

				}
				if(i == 2){ //Druhé tlačítko
					_delay_ms(delsiDelay);
					if(zadany[nasePozice] == 9){ //Pokud je číslo 9 přepíše se na 0
						zadany[nasePozice] = 0;
					}
					else{
						zadany[nasePozice]++;
					}
				}
				if(i == 3){ //Třetí tlačítko
					_delay_ms(delsiDelay);
					if(zadany[nasePozice] == 0){ //Pokud je číslo 0 přepíše se na 9
						zadany[nasePozice] = 9;
					}
					else{
						zadany[nasePozice]--;
					}
				}
				else{ //Čtvrté tlačítko
					_delay_ms(delsiDelay);
					potvrzeni = 0;
					for(int y = 0;y<4;y++){
						if(zadany[nasePozice] == heslo[i]){ //Kontrola každé cifry
							potvrzeni++;
						}
					}
					if(potvrzeni == 4){ //Pokud jsou všechny OK, otevře dveře
						PORTD = clearBit(PORTD,4);
						_delay_ms(delsiDelay);
					}
					PORTD = setBit(PORTD,4);
				}
			}
			zmenaCifry(i); //Vypisovani cisel na displej
			znak(zadany[i]);
			_delay_ms(kratkyDelay);
		}
	}
}
```

bitFunkce.c

```c
#include "bitFunkce.h"

//Nastaví daný bit do log. 1
int setBit(int value, int bitno) {
    value |= (1 << bitno);
    return value;
}

//Vynuluje daný bit
int clearBit(int value, int bitno) {
    value &= ~(1 << bitno);
    return value;
}

//Změna stavu daného bitu
int toggleBit(int value, int bitno) {
    value ^= (1 << bitno);
    return value;
}

//Vrátí true pokud je daný bit log. 1
int bitIsSet(int value, int bitno) {
    return (value >> bitno) & 1;
}

//Vrátí true pokud je daný bit log. 0
int bitIsClear(int value, int bitno) {
    return !bitIsSet(value, bitno);
}
```

bitFunkce.h

```c
#ifndef BITFUNKCE_H
#define BITFUNKCE_H

int setBit(int value, int bitno);

int clearBit(int value, int bitno);

int toggleBit(int value, int bitno);

int bitIsSet(int value, int bitno);

int bitIsClear(int value, int bitno);
#endif
```

ODPOVĚDI NA OTÁZKY:

1) Pro zvýšení odolnosti proti odposlechu / odpozorování hesla uživatele se používá challenge response autentizace. Vysvětlete, jakým způsobem toho tato technologie dosahuje. Co uživatel potřebuje?

   Může se stát situace, kdy někdo uvidí naše heslo a poté by to heslo mohl použít pro vlastní účely. Tenhle problém můžeme vyřešit použitím více hesel, které budou označené jiným identifikátorem. Poté když se někdo zeptá na heslo k nějakému identifikátoru, tak je malá šance, že útočník bude znát právě tohle heslo. Tohle můžeme předpokládat pouze tehdy, když je hesel velké množství a heslo se vybírá náhodně. A tomu se říká challenge response autentizace(viz.[1]).

2) V reálném zařízení by procesor digitálním signálem ovládal nějaký typ elektromagnetického zámku. Pokuste se uvést požadavky na tento zámek abyste dosáhli bezpečnosti alespoň srovnatelné s klíčem.

   Ideálně bychom chtěli, aby byl „fail secure“ (viz.[2]). To znamená, že když se zámek odpojí od napájení, tak zámek zůstane zamknutý. Bezpečnost přirovnatelná ke klíči by mohla být dosažená spíše s nějakou kartou než heslem. Protože karty jako klíče bývají konečné, ale heslo může vědět spousta lidí a když máme kartu u sebe, můžeme si být jistý, že ji nikdo jiný nemá.

3) V rámci zabezpečení elektronických zařízení (např. platební terminály apod.) je často nutné mít jistotu, že zařízení nebylo neoprávněným způsobem pozměněno. Pokuste se vymyslet alespoň jeden způsob jako toho dosáhnout.

   Systém by si zapisoval, kdy bylo zařízení pozměněno a poté by ověřilo, zda je to validní úprava. Případně udělat systém, kde není možné zařízení upravit bez pomoci někoho. Například změnu by musel někdo potvrdit, ale zároveň by nikdo neměl práva změnit heslo bez potvrzení(Ani samotný správce). Tím by útočník už potřeboval zjisti alespoň dvě hesla.

ZÁVĚR:

Úlohu jsem stihl skoro celou a jsem si jistý, že kdybychom měli poslední cvičení celé, tak bych to stihl všechno. Na prvním cvičení jsem měl problém pochopit, jak vlastně vykreslovat všechny číslice na jednou a jak poznat, které tlačítko vlastně bylo zmáčknuto. To jsem do druhého cvičení zjistil a celý trezor až na pár drobností jsem stihl udělat. Také jsem později zjistil, jak bych mohl udělat lépe stisk tlačítka a to pomocí rising/falling edge(viz.[3]). To funguje tak, že kontrolujeme stisk tlačítka v momentálním tiku a v tiku předchozím. Pokud jsme v předchozím tlačítko měli zmáčknuté, tak neregistrujeme stisk tlačítka.

[1] https://en.wikipedia.org/wiki/Challenge%E2%80%93response_authentication – Challenge Response autentizace
[2] https://en.wikipedia.org/wiki/Electromagnetic_lock - Elektromagnetický zámek
[3] https://www.scilab.org/signal-edge-detection - Rising/Falling Edge