```c
/*
 * GccApplication4.c

  Simulazione tastierino numerico in modo che soltanto all'inserimento della password B3 B5 B4 B5 si abbia l'apertura della porta,
  simulata dall'accensione del led L5. l'inserimento delle cifre deve essere segnalato dall'accensione dei led L0 L1 L2 L3 in sequenza.
  Alla fine dell'inserimento delle cifre se la password è corretta dovranno lampeggiare 3 volte (500 ms acceso, 200ms spento, 500ms acceso, 200ms spento e 500ms acceso).
  Premendo B7 due volte si richiuderà la porta senza inserimento della password.
 *
 * Created: 27/05/2024 17:05:41
 * Author : miria
 */

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdbool.h>

#define L0 (1<<PORTC0)
#define L1 (1<<PORTC1)
#define L2 (1<<PORTC2)
#define L3 (1<<PORTC3)
#define L5 (1<<PORTC5)

#define B2 (1<<PORTD2)
#define B3 (1<<PORTD3)
#define B4 (1<<PORTD4)
#define B5 (1<<PORTD5)
#define B6 (1<<PORTD6)
#define B7 (1<<PORTD7)

#define LEDS (L0 | L1 | L2 | L3 | L5)
#define BUTTONS (B2 | B3 | B4 | B5 | B6 | B7)

volatile uint8_t press = 0;
volatile uint8_t oldval = 0xff;
volatile uint8_t password[] = {B3, B5, B4, B5};
volatile uint8_t guess[] = {0, 0, 0, 0};
volatile int currentPos = 0;
volatile int tick = 0;
volatile int loopcount = 0;
volatile bool exitpress = false;

typedef enum{
	PASSWORD,
	BLINK,
	WAIT,
	RESET
	}States;
States currentState = PASSWORD;

bool passCheck();
void displayLed();
void blink(int loops, int cycle, int period, States nextState);

int main(void)
{
	DDRC |= LEDS;
	PORTC &=~ LEDS;

	DDRD &=~ BUTTONS;
	PORTD |= BUTTONS;

	PCICR |= (1<<PCIE2);
```

```
	PCMSK2 |= BUTTONS;

	TCCR1B |= (1<<WGM12);
	TIMSK1 |= (1<<OCIE1A);
	OCR1A = 1564; //100ms

	sei();

    while(1){
		switch(currentState){
			case PASSWORD:
				if(press != 0){
					guess[currentPos] = press;
					displayLed();
					currentPos ++;
					press = 0;
				}
				if(currentPos == 4){
					if(passCheck()){
						currentState = BLINK;
					}
					else{
						currentState = RESET;
					}
				}
				break;

			case BLINK:
				TCCR1B |= ((1<<CS10) | (1<<CS12));
				blink(3, 5, 7, WAIT);
				break;

			case WAIT:
				TCCR1B &=~ ((1<<CS10) | (1<<CS12));
				PORTC |= L5;

				if(press == B7){
					if(exitpress){
						currentState = RESET;
					}

					exitpress = true;
					press = 0;
				}
				else if(press != 0){
					exitpress = false;
					press = 0;
				}

				break;

			case RESET:
				TCCR1B &=~ ((1<<CS10) | (1<<CS12));
				PORTC &=~ LEDS;
				exitpress = false;
				currentPos = 0;
				tick = 0;
				currentState = PASSWORD;
				loopcount = 0;
				break;
		}
    }
}

ISR(PCINT2_vect){
```

```
	uint8_t change = oldval ^ PIND;
	oldval = PIND;
	for(uint8_t i = PIND2; i<=PIND7; i++){
		if((change & (1<<i)) && !(PIND & (1<<i))){
			press = (1<<i);
		}
	}
}

ISR(TIMER1_COMPA_vect){
	tick++;
}

bool passCheck(){
	bool check = true;
	for(int i = 0; i<4; i++){
		if(guess[i] != password[i]){
			check = false;
		}
	}
	return check;
}

void displayLed(){
	switch(currentPos){
		case 3:
			PORTC |= L3;
			break;
		case 2:
			PORTC |= L2;
			break;
		case 1:
			PORTC |= L1;
			break;
		default:
			PORTC |= L0;
			break;
	}
}

void blink(int loops, int cycle, int period, States nextState){
	if(tick <= cycle){
		PORTC |= (L0 | L1 | L2 | L3);
	}
	else{
		PORTC &=~ (L0 | L1 | L2 | L3);
	}

	if(tick == period){
		loopcount++;
		if(loopcount == loops){
			currentState = nextState;
		}
		tick = 0;
	}
}
```