```c
#include <avr/io.h>
#define F_CPU 8000000UL
#include <util/delay.h>
#include <avr/interrupt.h>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>


char nextRow = 0x40;            //Displayens adress till första tecknet på andra raden
char p1Points = 0b00110000;
char p2Points = 0b00110000;
int nbrOfGames = 0;             //Antal spelade omgångar
int selection = 0;              //Valt antal spelomgångar
int t = 0;

char diode1 = 0b00000001;
char diode2 = 0b00000010;
char diode3 = 0b00000100;
char green1 = 0b00001000;
char red1 = 0b00010000;
char green2 = 0b00100000;
char red2 = 0b01000000;

char p1k1 = 0b00000010;         //Spelare 1, knapp 1
char p1k2 = 0b00000100;
char p1k3 = 0b00001000;
char p2k1 = 0b00010000;
char p2k2 = 0b00100000;
char p2k3 = 0b01000000;

void setRSHigh()
{
     PORTD = PORTD | 0b10000000;
}

void setRSLow()
{
     PORTD = PORTD & 0b01111111;
}

void setRWHigh()
{
     PORTC = PORTC | 0b00000001;
}

void setRWLow()
{
     PORTC = PORTC & 0b11111110;
}

void setEHigh()
{
     PORTC = PORTC | 0b00000010;
}

void setELow()
{
```

```
	PORTC = PORTC & 0b11111101;
}

/*Skickar det som just nu står i PORTB till displayen */
void write()
{
	setELow();
	_delay_ms(2);
	setEHigh();
}

/*Flyttar positionen på displayen*/
void setAddress(char c)
{
	setRSLow();
	setRWLow();
	setEHigh();
	_delay_ms(2);
	PORTB = 0b10000000 + c;
	write();
}

/*Programmerar displayen och skriver ut "Click reset to start"*/
void setUpDisplay()
{
	setRSLow();
	setRWLow();
	setEHigh();
	PORTB = 0b00111000;
	write();
	PORTB = 0b00001100;
	write();

	setRSHigh();
	PORTB = 0b01000011; //C
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01101001; //i
	write();
	PORTB = 0b01100011; //c
	write();
	PORTB = 0b01101011; //k
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110100; //t
	write();

	setAddress(nextRow);
```

```
	setRSHigh();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01101111; //o
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01100001; //a
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01110100; //t
	write();
}

/*Suddar allt som står på displayen*/
void clearDisplay()
{
	setRSLow();
	setRWLow();
	setEHigh();
	_delay_ms(2);
	PORTB = 0b00000001;
	write();
}

/*Visar vilken spelare som är vilken*/
void introduction()
{
	clearDisplay();
	setAddress(0x00);
	setRSHigh();
	PORTB = 0b00111100; //<
	write();
	PORTB = 0b00101101; //-
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01100001; //a
	write();
	PORTB = 0b01111001; //y
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110001; //1
	write();
```

```
	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01100001; //a
	write();
	PORTB = 0b01111001; //y
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110010; //2
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00101101; //-
	write();
	PORTB = 0b00111110; //>
	write();
}

/*Nollställer spelarnas poäng och antalet spelade omgångar och skriver ut
	P1 0pts
	P2 0pts
 */
void newGame()
{
	clearDisplay();
	setRSHigh();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b00110001; //1
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b00110000; //0
	write();
	PORTB = 0b01110000; // p
	write();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01110011; //s
	write();

	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01010000; //P
	write();
	PORTB = 0b00110010; //2
	write();
	PORTB = 0b10000000; //tomt
	write();
```

```
	PORTB = 0b00110000; //0
	write();
	PORTB = 0b01110000; //p
	write();
	PORTB = 0b01110100; //t
	write();
	PORTB = 0b01110011; //s
	write();
	nbrOfGames = 0;
	p1Points = 0b00110000;
	p2Points = 0b00110000;
}

/*Skriver ut att spelare 1 vann*/
void playerWins(int k)
{
	clearDisplay();
	setRSHigh();
	PORTB = 0b01010000; //P
	write();

	if (k == 1)
	{
		PORTB = 0b00110001; //1
		write();
	}
	if (k == 2)
	{
		PORTB = 0b00110010; //2
		write();
	}
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110111; //w
	write();
	PORTB = 0b01101001; //i
	write();
	PORTB = 0b01101110; //n
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b00100001; //!
	write();

	setAddress(nextRow);
	setRSHigh();
	PORTB = 0b01001110; //N
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110111; //w
	write();
	PORTB = 0b00111010; //:
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01000011; //C
	write();
```

```
	PORTB = 0b01101100; //l
	write();
	PORTB = 0b01101001; //i
	write();
	PORTB = 0b01100011; //c
	write();
	PORTB = 0b01101011; //k
	write();
	PORTB = 0b10000000; //tomt
	write();
	PORTB = 0b01110010; //r
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110011; //s
	write();
	PORTB = 0b01100101; //e
	write();
	PORTB = 0b01110100; //t
	write();
}

/*Ger spelare 1 en poäng och ökar poängen på displayen*/
void p1Point()
{
	char place = 0x03;
	setAddress(place);
	p1Points++;
	PORTB = p1Points;
	setRSHigh();
	write();
}

/*Ger spelare 2 en poäng och ökar poängen på displayen*/
void p2Point()
{
	char place = 0x43;
	setAddress(place);
	p2Points++;
	PORTB = p2Points;
	setRSHigh();
	write();
}

/*Hjälpmetod för att skriva ut en siffra på displayen*/
void printNbr(int k)
{
	PORTB = 0b00110000 + k;
	write();
}

/*Skriver ut reaktionstiden för spelare 1*/
void playerTime(int ms, int player)
{
	char place;
	if (player == 1)
	{
		place = 0x0A;
```

```
	}
	if (player == 2)
	{
		place = 0x4A;
	}
	setAddress(place);
	setRSHigh();
	int k1 = ms / 1000;
	printNbr(k1);
	int k2 = (ms / 100) % 10;
	printNbr(k2);
	int k3 = (ms / 10) % 10;
	printNbr(k3);
	int k4 = ms % 10;
	printNbr(k4);
	PORTB = 0b01101101; //m
	write();
	PORTB = 0b01110011; //s
	write();
}

/*Slumpar fram vilken diod som ska tändas*/
int randDiode()
{
	#define RAND_MAX = 2;
	return (rand() + 1);
}

/*Slumpar fram hur lång tid det ska ta innan dioden tänds*/
int randTime()
{
	#define RAND_MAX = 4000;
	return rand();
}

/*Tar bort reaktionstiderna från displayen*/
void clearTime()
{
	char place = 0x0A;
	setAddress(place);
	setRSHigh();
	for(int i = 0; i < 6; i++){
		PORTB = 0b10000000;
		write();
	}
	place = 0x4A;
	setAddress(place);
	setRSHigh();
	for(int i = 0; i < 6; i++){
		PORTB = 0b10000000;
		write();
	}
}

/*En spelomgång där en diod tänds*/
void gameRound()
{
	int b = randTime() / 10;
```

```
_delay_ms(b + 3000);
int a = randDiode() / 10000;
if (a == 0)
{
	PORTD = 0b00000001;
}
if (a == 1)
{
	PORTD = 0b00000010;
}
if (a == 2)
{
	PORTD = 0b00000100;
}
clearTime();
t = 0;

int p1Pressed = 0;
int p2Pressed = 0;
while(1)
{
	if (p1Pressed == 0 && (PINA & p1k1) == p1k1)
	{
		p1Pressed++;
		int t1 = t;
		playerTime(t1, 1);
		if (a == 0)
		{
			PORTD = PORTD | green1;
			if(p1Pressed > p2Pressed)
			{
				p1Point();
			}
		}
		else
		{
			PORTD = PORTD | red1;
			if(p1Pressed > p2Pressed)
			{
				p2Point();
			}
		}
	}
	if (p1Pressed == 0 && (PINA & p1k2) == p1k2)
	{
		p1Pressed++;
		int t1 = t;
		playerTime(t1, 1);
		if (a == 1)
		{
			PORTD = PORTD | green1;
			if(p1Pressed > p2Pressed)
			{
				p1Point();
			}
		}
		else
		{
```

```
            PORTD = PORTD | red1;
            if(p1Pressed > p2Pressed)
            {
                  p2Point();
            }
      }
}
if (p1Pressed == 0 && (PINA & p1k3) == p1k3)
{
      p1Pressed++;
      int t1 = t;
      playerTime(t1, 1);
      if (a == 2)
      {
            PORTD = PORTD | green1;
            if(p1Pressed > p2Pressed)
            {
                  p1Point();
            }
      }
      else
      {
            PORTD = PORTD | red1;
            if(p1Pressed > p2Pressed)
            {
                  p2Point();
            }
      }
}
if (p2Pressed == 0 && (PINA & p2k1) == p2k1)
{
      p2Pressed++;
      int t2 = t;
      playerTime(t2, 2);
      if (a == 0)
      {
            PORTD = PORTD | green2;
            if(p1Pressed < p2Pressed)
            {
                  p2Point();
            }
      }
      else
      {
            PORTD = PORTD | red2;
            if(p1Pressed < p2Pressed)
            {
                  p1Point();
            }
      }
}
if (p2Pressed == 0 && (PINA & p2k2) == p2k2)
{
      p2Pressed++;
      int t2 = t;
      playerTime(t2, 2);
      if (a == 1)
      {
```

```
                        PORTD = PORTD | green2;
                        if(p1Pressed < p2Pressed)
                        {
                              p2Point();
                        }
                  }
                  else
                  {
                        PORTD = PORTD | red2;
                        if(p1Pressed < p2Pressed)
                        {
                              p1Point();
                        }
                  }
            }
            if (p2Pressed == 0 && (PINA & p2k3) == p2k3)
            {
                  p2Pressed++;
                  int t2 = t;
                  playerTime(t2, 2);
                  if (a == 2)
                  {
                        PORTD = PORTD | green2;
                        if(p1Pressed < p2Pressed)
                        {
                              p2Point();
                        }
                  }
                  else
                  {
                        PORTD = PORTD | red2;
                        if(p1Pressed < p2Pressed)
                        {
                              p1Point();
                        }
                  }
            }
            if (p1Pressed == 1 && p2Pressed == 1)
            {
                  _delay_ms(2500);
                  PORTD = 0x00;
                  break;
            }
            if (t > 3000)
            {
                  PORTD = 0x00;
                  nbrOfGames--;
                  break;
            }
      }
      nbrOfGames++;
}

/*Spelarna får välja hur många omgångar spelet ska pågå i*/
void chooseRounds() {
      clearDisplay();
      setRSHigh();
      PORTB = 0b01000011; // C
```

```
write();
PORTB = 0b01101000; //h
write();
PORTB = 0b01101111; //o
write();
PORTB = 0b01101111; //o
write();
PORTB = 0b01110011; //s
write();
PORTB = 0b01100101; //e
write();
PORTB = 0b10000000; //tomt
write();
PORTB = 0b01101110; //n
write();
PORTB = 0b01100010; //b
write();
PORTB = 0b01110010; //r
write();
PORTB = 0b10000000; //tomt
write();
PORTB = 0b01110010; //r
write();
PORTB = 0b01101110; //n
write();
PORTB = 0b01100100; //d
write();
PORTB = 0b01110011; //s
write();
PORTB = 0b00111010; //:
write();

char place = 0x40;
setAddress(place);
setRSHigh();
PORTB = 0b00110101; //5
write();
place = 0x48;
setAddress(place);
setRSHigh();
PORTB = 0b00110111; //7
write();
place = 0x4F;
setAddress(place);
setRSHigh();
PORTB = 0b00111001; //9
write();

while(1)
{
	char temp = PINA & p1k3;
	if (temp == p1k3)
	{
		selection = 5;
		newGame();
		break;
	}
	temp = PINA & 0x01;
```

```
            if (temp == 0x01)
            {
                  selection = 7;
                  newGame();
                  break;
            }
            temp = PINA & p2k3;
            if (temp == p2k3)
            {
                  selection = 9;
                  newGame();
                  break;
            }
      }
}

/*Sätter timern till att göra avbrott varje millisekund*/
void setTimer()
{
      TCCR1A = 0;
      TCCR1B = 0;
      TCNT1 = 0;
      OCR1A = 999;
      TCCR1B |= (1 << WGM12);
      TCCR1B |= (1 << CS11);
      TIMSK |= (1 << OCIE1A);
      sei();
}

/*Ökar t med 1 vid varje avbrott*/
ISR(TIMER1_COMPA_vect)
{
      t++;
}

int main(void)
{
      DDRD = 0xFF;
      DDRA = 0x00;
      DDRB = 0xFF;
      DDRC = 0x03;

      setTimer();

      clearDisplay();
      setUpDisplay();
      srand(time(NULL));

      while(1)
      {
            while(1)
            {
                  char temp = PINA & 0x01;
                  if (temp == 0x01)
                  {
                        newGame();
                        break;
                  }
```

```
            }

            introduction();
            _delay_ms(3000);
            chooseRounds();

            while(nbrOfGames < selection)
            {
                  gameRound();
                  if (p1Points == 0b00110000 + selection / 2 + 1)
                  {
                        playerWins(1);
                        break;
                  }
                  if (p2Points == 0b00110000 + selection / 2 + 1)
                  {
                        playerWins(2);
                        break;
                  }
            }
      }
}
```