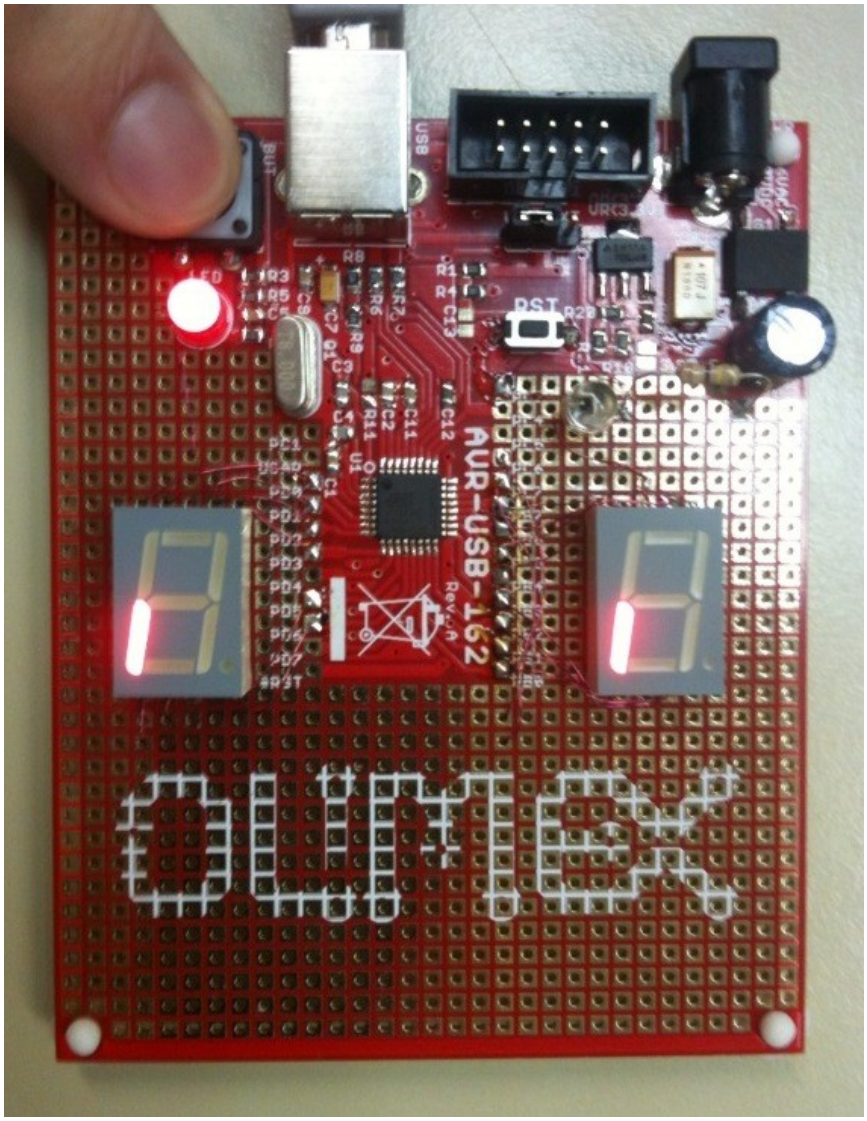
Projektarbeit in DVT

Reaktionsspiel

OLIMEX AVR-USB-162 Entwicklerboard



Matthias Kirschner - 21.07.2010

Inhaltsverzeichnis

1. Aufgabenstellung

Entwicklung eines kleinen Spieles auf einem Microcontroller Entwicklerboard. Der Spieler soll möglichst tief ins Spielgeschehen verwickelt werden. Eine weitere Anforderung sind aussagekräftige Rückmeldungen des Systems an den Spieler. Die Projektarbeit wurde während des DVT-Unterrichts durchgeführt.

2. Entwicklungsumgebung

Hardware:

- OLIMEX AVR-USB-162 Entwicklerboard
- ATmega AT90USB162 (auf Entwicklerboard)
- zwei 7-Segment Anzeigen (auf Entwicklerboard)
- eine LED (auf Entwicklerboard)
- Taster (auf Entwicklerboard)
- USB-Kabel
- Schul-PC

Software:

- AVR Studio 4.18
- Flip 3.4.1

3. Spielidee

Das System zeigt als Vorgabe ein Segment der 7-Segment Anzeige, dieses Segment muss vom Spieler getroffen werden. Um einen Treffer zu erzielen, muss die Reaktionszeit des Spielers ausreichend schnell sein. Am Schluss wird als Ergebnis die Anzahl getroffener Segmente angezeigt.

4. Spielverlauf

Spiel starten:

Das Spiel startet mit dem „Reset“-Taster. Mit dem „Reset“-Taster kann auch während oder am Ende eines Spiels ein neues Spiel gestartet werden.

Segment treffen:

Auf der linken 7-Segment Anzeige wird vom System die Vorgabe geschalten. Auf der rechten Anzeige laufen alle Segmente der Reihe nach durch. Der Spieler muss nun bei übereinstimmenden Segmenten den „Eingabe“-Taster drücken, um einen Treffer zu erzielen.

Nächstes Segment schalten:

Beim Loslassen des „Eingabe“-Tasters wird bei der linken Anzeige das nächste Segment geschalten, egal ob das vorherige Segment getroffen wurde oder nicht. Nun muss der Spieler wieder das übereinstimmende Segment treffen.

Ergebnis anzeigen:

Wenn alle Segmente als Vorgabe durchlaufen sind, wird auf der rechten Anzeige das Ergebnis des Spiels angezeigt. Alle Treffer werden gezählt und angezeigt, somit sind ein Maximalwert von 7 und ein Minimalwert von 0 möglich.

5. Implementierung

Definitionen

```
#include <avr/io.h>
#define F_CPU 1000000
#include <util/delay.h>



void setStaticLed(int position);
void setRunLed(int position);
int checkChoise(int* ledPosition, int position, int* oks);
void showResults(int oks);



enum segment1
{
	LED1B_A = 0x20,
	LED1B_B = 0x10,
	LED1B_C = 0x04,
	LED1B_D = 0x02,
	LED1B_E = 0x01,
	LED1B_F = 0x40,
	LED1B_G = 0x80,
	LED1B_P = 0x08
};



enum segment2
{
	LED2D_A = 0x04,
	LED2D_B = 0x08,
	LED2D_C = 0x40,
	LED2C_D = 0x80,
	LED2C_E = 0x40,
	LED2D_F = 0x02,
	LED2D_G = 0x01,
	LED2D_P = 0x20
};



enum others
{
	BUTD = 0x80,
	RSTC = 0x02,
	LEDD = 0x10
};
```

Schalten des Vorgabe-Segments

```
void setStaticLed(int position)
{
	switch (position)
	{
		case 0:
		{
			PORTC = 0x00;
			PORTD = LED2D_A;
		}
		break;

		case 1:
		{
			PORTC = 0x00;
			PORTD = LED2D_B;
		}
		break;

		case 2:
		{
			PORTC = 0x00;
			PORTD = LED2D_C;
		}
		break;

		case 3:
		{
			PORTC = LED2C_D;
			PORTD = 0x00;
		}
		break;

		case 4:
		{
			PORTC = LED2C_E;
			PORTD = 0x00;
		}
		break;

		case 5:
		{
			PORTC = 0x00;
			PORTD = LED2D_F;
		}
		break;

		case 6:
		{
			PORTC = 0x00;
			PORTD = LED2D_G;
		}
		break;
	}
}
```

Schalten des durchlaufenden Segments

```
void setRunLed(int position)
{
	switch (position)
	{
		case 0:
		{
			PORTB = LED1B_A;
		}
		break;

		case 1:
		{
			PORTB = LED1B_B;
		}
		break;

		case 2:
		{
			PORTB = LED1B_C;
		}
		break;

		case 3:
		{
			PORTB = LED1B_D;
		}
		break;

		case 4:
		{
			PORTB = LED1B_E;
		}
		break;

		case 5:
		{
			PORTB = LED1B_F;
		}
		break;

		case 6:
		{
			PORTB = LED1B_G;
		}
		break;
	}
}
```

Schalten der Ergebnisanzeige

```
void showResults(int oks)
{
	switch (oks)
	{
		case 0:
		{
			PORTB = 0x77;
		}
		break;

		case 1:
		{
			PORTB = 0x14;
		}
		break;

		case 2:
		{
			PORTB = 0xB3;
		}
		break;

		case 3:
		{
			PORTB = 0xB6;
		}
		break;

		case 4:
		{
			PORTB = 0xD4;
		}
		break;

		case 5:
		{
			PORTB = 0xE6;
		}
		break;

		case 6:
		{
			PORTB = 0xE7;
		}
		break;

		case 7:
		{
			PORTB = 0x34;
		}
		break;
	}

	PORTC = 0x00;
	PORTD = 0x00;
}
```

Vergleich der Segmente

```
int checkChoise(int* ledPosition, int position, int* oks)
{
	int i = 0;
	int currentState = 0;
	int lastState = 0;

	if (PIND & BUTD)
	{
		return 0;
	}

	for (i = 0; i < 1000; i++)
	{
		currentState = (PIND & BUTD);

		if (currentState != lastState)
		{
			i = 0;

			lastState = currentState;
		}
	}

	if ((*ledPosition) == position)
	{
		PORTD |= LEDD;

		(*oks)++;
	}

	while (!(PIND & BUTD));

	if ((*ledPosition) == 6)
	{
		showResults(*oks);

		return 1;
	}
	else
	{
		(*ledPosition)++;
	}

	return 0;
}
```

Programmablauf (main)

```
int main(void)
{
	int staticPosition = 0;
	int runPosition = 0;
	int oks = 0;

	DDRB  = 0xff;
	DDRC  = 0xc0;
	DDRD  = 0x7f;
	DDRD &= ~BUTD;


	while (1)
	{
		setStaticLed(staticPosition);

		if (checkChoise(&staticPosition, runPosition, &oks))
		{
			return 0;
		}

		if (runPosition == 6)
		{
			runPosition = 0;
		}
		else
		{
			runPosition++;
		}

		setRunLed(runPosition);

		_delay_ms(100);
	}

	return 0;
}
```