Tallinna Tehnikaülikool
Mehhatroonika instituut

Teeme Ise 2011
Aruanne

Harri Parker

093430 MAHB-41

Tallinn 2011

Sisukord

Sissejuhatus

Selle kursus võtsin vaba ainena, et õppida riistvaralist programmeerimist. Eelnevalt olin õppinud gümnaasiumis 3 aastat Pascalit siis tulles ülikooli oli meil aine Informaatika II kus õppetati VBA'd, mis minu arust oli aja raiskamine. Eelmine semester võtsin aine C++ programmeerisime, kus õpetati just tarkvaralist programmerimist. See pärast oli see kursus minu jaoks väga kasulik, sest minu arust peaks mehhatroonikas just riistvaralist programmeerimist õpetama. Me programmeerisime Pololu 3π robotit millel on 3 nuppu, 5 jooneandurit, kaks mootorit ja lisaks lisaime ka kaugus anduri. Roboti mikrokontrolleriks on ATmega 328p. Põhiline asi, millele rõhku panid oli andmelehe kasutamine ja sealt vajaliku leidmine.

Kursuse käigus õpitu

LED'ide ja nuppude kasutamine

Esimesena õpetati meile kuidas portide pinnidele väärtusi anda ja neid lugeda. Ja tegime koodi, kus nupp A pani LED'i põlema ja nupp C kustutas selle.

LED'ide ja nupude kasutamine

```
include <avr/io.h>
#define LEDPIN 7
#define BTNPINA 1
#define BTNPINC 5

int main(void)
{
	uint8_t seis = 0;
	DDRB &= ~(1 << BTNPINA);
	DDRB &= ~(1 << BTNPINC);
	//DDRB &= 0xDD;
	DDRD |= 1 << LEDPIN;
	PORTD = 0x00;
	while(1)
	{
		if((!(PINB & 1 << BTNPINA)) && (!seis))
		{
			PORTD |= 1 << LEDPIN;
			seis = 1;
		}
		if((!(PINB & 1 << BTNPINC)) && (seis))
		{
			PORTD &= ~(1 << LEDPIN);
			seis= 0;
		}
	}
return 0;
}
```

PWM ja mootorite kasutamine

Teiseks räägiti meile PWM olemusest ja kuidas seda rakendada meie robotitel. PWM e. Pulse-Width-Modulation. Sellega saab digitaalsest signaalist teha analoog signaal.

Põhimõtteliselt kui anda lühikesed sisenpulsid siis väljundiks on madal pidev pinge aga kui anda pikad sisend pulsid siis on väljundiks kõrge pidev pinge. Meie robotitel kasutasime PWM'i mootorite juhtimiseks kuna mootoritele tuleb anda analoog signaal kas siis vastavalt aeglasti või kiiresti liikuda. Järgnev kood on PWM initsialiseerimisest ja mootorite liigutamiseks kasutatavad funktsioonid, kus antakse taimeri võrdlus muutujatele väärtusi. Viimane funktsioon Coast() on selleks, et kui tavalistele mootoritele anda kiirus 0 siis ta pidurdab aga Coast() funktsiooniga jääb sujuvalt seisma.

PWM ja mootorite kasutamine

```
void initPWM()
{
	DDRD |= (1 << MOT1A) | (1 << MOT1B) | (1 << MOT2A);
	DDRB |= (1 << MOT2B);
	PORTD &= ~((1 << MOT1A) | (1 << MOT1B) | (1 << MOT2A));
	PORTB &= ~(1 << MOT2B);

	TCCR0A |= (1 << COM0A1) | (1 << COM0B1);
	TCCR0A |= (1 << WGM00) | (1 << WGM01);
	TCCR0B |= (1 << CS01);

	TCCR2A |= (1 << COM2A1) | (1 << COM2B1);
	TCCR2A |= (1 << WGM20) | (1 << WGM21);
	TCCR2B |= (1 << CS21);
}

void mootor_V(uint8_t suund, uint8_t kiirus)
{
	if(!suund)
	{
		OCR0A = 255;
		OCR0B = 255 - kiirus;
	}
	else
	{
		OCR0A = 255 - kiirus;
		OCR0B = 255;
	}
}

void mootor_P(uint8_t suund, uint8_t kiirus)
{
	if(!suund)
	{
		OCR2A = 255;
		OCR2B = 255 - kiirus;
	}
	else
	{
		OCR2A = 255 - kiirus;
		OCR2B = 255;
	}
}

void Coast()
{
		OCR0A = 0;
		OCR0B = 0;
		OCR2A = 0;
		OCR2B = 0;
}
```

Joone andurite kasutamine

Kolmandaks õpetati meile joone andurite kasutamist. Selleks olid robotil 5 IR sensorit koosnevad kahest põhilisest komponendist IR LED ja fototransistor, skeemis on veel ka kondensaator, mis algul täislaetakse ja siis mõõdetakse aega kui kaua läheb tühjaks laadimiseks. See aeg sõltub sellest kui palju voolu fototransistor läbi laseb ning see sõltub kui palju valgust selle pinnale langeb kui on heledam pind siis peegeldub rohkem valgust fototransistorile kui tumedalt pinnalt. Järelikult mida tumedam on pind seda pikem on tühjaks laadimise aeg. Järgnev koodijupp funktsioon, mis muudab globaalses massiivis buf[] väärtusi vastavalt nii, et kui on joon siis on väärtuseks 1 vastsel juhul 0.

Joone andurite kasutamine

```
void readSensor()
{
	uint8_t mas[5] = {0, 0, 0, 0, 0};

	DDRC |= (1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP);
	PORTC |= (1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP);

	_delay_us(10);

	DDRC &= ~((1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP));
	PORTC &= ~((1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP));

	TCNT1 = 0; // reset timer;

	while(TCNT1 < 500)
	{
		if((!(PINC & (1 << SENP))) && (!mas[0]))
		{
			mas[0] = 1;
		}
		if((!(PINC & (1 << SENPK))) && (!mas[1]))
		{
			mas[1] = 1;
		}
		if((!(PINC & (1 << SENK))) && (!mas[2]))
		{
			mas[2] = 1;
		}
		if((!(PINC & (1 << SENVK))) && (!mas[3]))
		{
			mas[3] = 1;
		}
		if((!(PINC & (1 << SENV))) && (!mas[4]))
		{
			mas[4] = 1;
		}
	}

	for(uint8_t i = 0; i < 5 ;i ++)
	{
		if(!mas[i])
		{
			buf[i] = 1; //on joon
		}
		else
		{
			buf[i] = 0; //ei ole joon
		}
	}

}
```

ADC ja SHARP sensori kasutamine

Me lisasime SHARP sensori et kaugust mõõta selleks kasutasime ADC'd ADC – analoog digita converter. Seda oli meil vaja sellepärast, et SHARP andur annab annab analoogväljundi ja selle kontrollerile arusaadavaks teha tuleb see digitaliseerida. Koodijupis on 3 funktsiooni: esimene on katkestus teine on ADC initsialiseerimiseks ja kolmas on funktsioon, mille sisendiks on SHARP andurist saadav pinge väärtus ja väljundiks on kaugus. Selle sõltuvuse kalkuleerisin excelis kasutades andmelehelt saadud väärtusi.

ADC ja SHARP sensori kasutamine

```
ISR(ADC_vect)
{
	if(voltageCounter > arraySize)
	{
		voltageCounter = 0;
	}
	voltageAVG[voltageCounter] = ADCH;
	voltageCounter++;
}

void initADC()
{
	ADCSRA |= ((1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0) | (1 << ADATE) | (1 << ADEN) | (1 << ADSC) | (1 << ADIE));

	ADMUX |= ((1 << REFS0) | (1 << ADLAR) | (1 << MUX1) | (1 << MUX2));
}

double RangeSharp(int sisend)
{
	double volt = sisend * 5 / 255;
	return 27.609 * pow(volt, -1.191);
}
```

PID kontrolleri kasutamine

Meile räägiti ka PID kontrollerist. Mis iseenesest on põhimõte kus väljund sõltub veast, meie puhul oli veaks kõrvale kalle joonelt. PID koosneb kolmest komponendist proportsionaalne, integraalne ja tuletis. Selle kasutamine muutis robotite liikumise joonel sujuvamaks. Me esialgu leidsime PID konstante katse eksitus meetodi abil. Aga seda saab optimeerida ja leida funktsioon, mis annab konstantide väärtusi vastavalt kiirusel ja soovitud käitumisele (stabiilsus, vea korrigeerimise kiirus, ülereageerimine ja stabiilsuse saavutamise kiirus). Järgnev koodijupp näitab kuidas leidsime vea ja kuidas sellest tulenevalt saame kiiruste erinevuse.

PID kontrolleri kasutamine

```
int error()
{
	uint8_t i;
	int angle[] = {45, 15, 0, -15, -45};
  uint8_t darkest_sensor = 0;
	for(i = 0; i < 5; i++)
	{
    if(linesensor[i] > linesensor[darkest_sensor])
		{
			darkest_sensor = i;
```

```
                }
        }
        return angle[darkest_sensor] - 2000;
}

#define K_P 4.5
#define K_I 0.00005
#define K_D 1.0

float integral = 0.0;
float proportional, derivative;
float last_proportional = 0.0;

int16_t pid()
{
        proportional = (float) error();
        derivative = proportional - last_proportional;
        integral += proportional;
        last_proportional = proportional;
        return(K_P * proportional + K_I * integral + K_D * derivative);
}
```

Labürindi läbimine

Labürindi läbimiseks kasutasin vask käsi seinal meetod. Ühesõnaga kui robot jõuab ristmikule siis ta vaatab kas saab vasakule pöörata kui ei siis vaatab kas saab otse minna ja kui otse ka minna ei saa siis vaatab kas saab paremale pöörata ja kui tuleb välja et on tupik siis tee u-pöörde. Iga kord kui robot teeb pöörde siis ta kirjutab ühemõõtmelisesse maatriksisse kas v, o, p või u tähe vastavalt sooritatud pöördele.

Iga kord kui toimub pööre siis pärast seda üritab kood kirjapandud teed optimeerida. Kood teeb seda sel juhul kui teepikkus on vähemalt 3 ehk on olnud kolm pööret ja eelviimane pööre oli u-pööre. Optimeerimine põhineb sellel et u-pöörded saab välja jätta sest need on liigsed. Kui robot jõuab lõppu siis ta välju esimesest tsüklist ning pärast nupu vajutust läbib robot optimaalseima tee lõpuni. Järgnev kood on optimeerimise kood.

Optimeerimise kood

```
void rada_korda()
{
        switch(teekond[tee_pikkus-3])
        {
                case 'v':
                        switch(teekond[tee_pikkus-1])
                        {
                                case 'v':
                                teekond[tee_pikkus-3] = 'o';
                                break;
                                case 'o':
                                teekond[tee_pikkus-3] = 'p';
                                break;
                                case 'p':
                                teekond[tee_pikkus-3] = 'u';
                                break;
                        }
                        break;
                case 'o':
                        switch(teekond[tee_pikkus-1])
                        {
```

```
                                        case 'v':
                                        teekond[tee_pikkus-3] = 'p';
                                        break;
                                        case 'o':
                                        teekond[tee_pikkus-3] = 'u';
                                        break;
                                }
                                break;
                        case 'p':
                                teekond[tee_pikkus-3] = 'u';
                                break;
        }
        tee_pikkus -= 2;
        return;
}
```

Kokkuvõte

Minu jaoks oli kursus väga tore ja sai ka uusi teadmisi. Sai ka endale ostetud robot millega oli hea lihtne kodus katsetada. Labürindi kood on lisas kui mina seda katsetasin siis igatahes töötas aga seda koodi saaks kindlasti parandada ja arendada. Kuna nüüd endal robot siis on aega seda teha ja leida veel uusi kasutus ideid robotile. Oli tore ka käia Leedus joonejärgmise võistlusel, kus sain oma robotiga 4’nda koha. Oli ka tore see, et kui küsisin abi teemadel, mida tundides ei kajastatud siis oldi alati sõbralik ja abivalmis.

Lisa

Programmi kood

```
#include <avr/io.h>
#include <inttypes.h>
#include <string.h>

#define F_CPU 20000000UL // 20 MHz
#include <util/delay.h>

#define MOT1A PD5
#define MOT1B PD6
#define MOT2A PD3
#define MOT2B PB3

#define BTNA PB1
#define BTNB PB4
#define BTNC PB5

#define LEDG PD7
#define LEDR PD1
#define IRLED PC5

#define SENP PC4
#define SENPK PC3
#define SENK PC2
#define SENVK PC1
#define SENV PC0

#define TUME 500

char teekond[100] = "";
unsigned char tee_pikkus = 0;
uint8_t buf[5];

void readSensor()
{
	uint8_t mas[5] = {0, 0, 0, 0, 0};

	DDRC |= (1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP);
	PORTC |= (1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP);

	_delay_us(10);

	DDRC &= ~((1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP));
	PORTC &= ~((1 << SENV) | (1 << SENVK) | (1 << SENK) | (1 << SENPK) | (1 << SENP));
```

```
		TCNT1 = 0; // reset timer;

		while(TCNT1 < 900)
		{
			if((!(PINC & (1 << SENP))) && (!mas[0]))
			{
				mas[0] = 1;
			}
			if((!(PINC & (1 << SENPK))) && (!mas[1]))
			{
				mas[1] = 1;
			}
			if((!(PINC & (1 << SENK))) && (!mas[2]))
			{
				mas[2] = 1;
			}
			if((!(PINC & (1 << SENVK))) && (!mas[3]))
			{
				mas[3] = 1;
			}
			if((!(PINC & (1 << SENV))) && (!mas[4]))
			{
				mas[4] = 1;
			}
		}

		int i;
		for(i = 0; i < 5 ;i ++)
		{
			if(!mas[i])
			{
				buf[i] = 1;
			}
			else
			{
				buf[i] = 0;
			}
		}

}


void initPWM()
{
		DDRD |= (1 << MOT1A) | (1 << MOT1B) | (1 << MOT2A);
		DDRB |= (1 << MOT2B);
		PORTD &= ~((1 << MOT1A) | (1 << MOT1B) | (1 << MOT2A));
		PORTB &= ~(1 << MOT2B);
```

```
	TCCR0A |= (1 << COM0A1) | (1 << COM0B1);
	TCCR0A |= (1 << WGM00) | (1 << WGM01);
	TCCR0B |= (1 << CS01);

	TCCR2A |= (1 << COM2A1) | (1 << COM2B1);
	TCCR2A |= (1 << WGM20) | (1 << WGM21);
	TCCR2B |= (1 << CS21);
}

void mootor_V(uint8_t suund, uint8_t kiirus)
{
	if(!suund)
	{
		OCR0A = 255;
		OCR0B = 255 - kiirus;
	}
	else
	{
		OCR0A = 255 - kiirus;
		OCR0B = 255;
	}
}

void mootor_P(uint8_t suund, uint8_t kiirus)
{
	if(!suund)
	{
		OCR2A = 255;
		OCR2B = 255 - kiirus;
	}
	else
	{
		OCR2A = 255 - kiirus;
		OCR2B = 255;
	}
}

void Coast()
{
		OCR0A = 0;
		OCR0B = 0;
		OCR2A = 0;
		OCR2B = 0;
}

void otse()
{
```

```
	while(1)
	{
		readSensor();
		mootor_V(0, 40);
		mootor_P(0, 40);

		if(!buf[1] && !buf[2] && !buf[3])
		{
			return;
		}
		else if(buf[0] || buf[4])
		{
			return;
		}

	}
}

void keera(char suund)
{
	switch(suund)
	{
	case 'v': // Keera vasakule
		mootor_V(1, 80);
		mootor_P(0, 80);
		_delay_ms(200);
		break;
	case 'p': // Keera paremale.
		mootor_V(0, 80);
		mootor_P(1, 80);
		_delay_ms(200);
		break;
	case 'u': // Pööra ümber.
		mootor_V(0, 80);
		mootor_P(1, 80);
		_delay_ms(385);
		break;
	case 'o': // Sõida otsa
		break;
	}
}

void rada_korda()
{
	switch(teekond[tee_pikkus-3])
	{
		case 'v':
			switch(teekond[tee_pikkus-1])
```

```
                                        {
                                                case 'v':
                                                teekond[tee_pikkus-3] = 'o';
                                                break;
                                                case 'o':
                                                teekond[tee_pikkus-3] = 'p';
                                                break;
                                                case 'p':
                                                teekond[tee_pikkus-3] = 'u';
                                                break;
                                        }
                                        break;
                        case 'o':
                                        switch(teekond[tee_pikkus-1])
                                        {
                                                case 'v':
                                                teekond[tee_pikkus-3] = 'p';
                                                break;
                                                case 'o':
                                                teekond[tee_pikkus-3] = 'u';
                                                break;
                                        }
                                        break;
                        case 'p':
                                        teekond[tee_pikkus-3] = 'u';
                                        break;
                }
                tee_pikkus -= 2;
                return;
}

int main(void)
{
                DDRB &= ~(1 << BTNA);
                DDRB &= ~(1 << BTNB);
                DDRB &= ~(1 << BTNC);

                DDRD |= (1 << LEDG);
                DDRD |= (1 << LEDR);
                DDRC |= (1 << IRLED);

                PORTC |= (1 << IRLED);
                TCCR1B |= 1 << CS11;

                uint8_t a = 0;
                uint8_t b = 0;
                uint8_t c = 0;
```

```
unsigned char suund;

initPWM();
while (1)
{
	if(!(PINB & 1 << BTNB))
	{
		b = 1;
		break;
	}
}
while(b)
{
	otse();
	mootor_V(0, 50);
	mootor_P(0, 50);
	_delay_ms(50);

	unsigned char on_V=0;
	unsigned char on_P=0;
	unsigned char on_O=0;

	readSensor();

	if(buf[0])
	{
		on_V = 1;
	}
	if(buf[4])
	{
		on_P = 1;
	}

	mootor_V(0, 40);
	mootor_P(0, 40);
	_delay_ms(200);

	readSensor();
	if(buf[1] || buf[2] || buf[3])
	{
		on_O = 1;
	}

	if(buf[1] && buf[2] && buf[3])
	{
		c = 1;
		Coast();
		break;
```

```
		}

		if(on_V)
		{
			suund = 'v';
		}
		else if(on_O)
		{
			suund = 'o';
		}
		else if(on_P)
		{
			suund = 'p';
		}
		else
		{
			suund = 'u';
		}

		keera(suund);

		teekond[tee_pikkus] = suund;
		tee_pikkus ++;

		if((tee_pikkus > 2) && (teekond[tee_pikkus-2] == 'u'))
		{
			rada_korda();
		}
}
while (c)
{
		if(!(PINB & 1 << BTNA))
		{
			a = 1;
			break;
		}
}

while(a)
{
		int i;
		for(i=0;i<tee_pikkus;i++)
		{
			otse();
			mootor_V(0, 50);
			mootor_P(0, 50);
			_delay_ms(50);
			mootor_V(0, 40);
```

```
                        mootor_P(0, 40);
                        _delay_ms(200);
                        keera(teekond[i]);
                }
                otse();
                Coast();
                break;
        }
        while(1)
        {
                Coast();
        }
        return 0;
}
```