Tallinna Tehnikaülikool

Mehhatroonika instituut

Mikrokontrollerid ja praktiline robootika

Aruanne

Oliver Mets
(MAHB-41)

Tallinn 2011

Sissejuhatus

„Teeme ise 2011“ raames õppisime riistvarapõhilist mikrokontrolleri programmeerimist Pololu 3π roboti põhjal.

Robot on oma olemuselt ja eesmärgilt kaherattaline joonejärgija, millel on kolmandaks tugipunktiks kuul ja anduriteks viis infrapuna sensorit. Mikrokontrolleriks on ATmega328p, mis töötab maksimaalselt 20 MHz juures ja on 8-bitiste andmeedastussiinidega. Robotil on ka ekraan ja ka heligeneraator/sumisti, mida kursuse raames me ei jõudnud kasutada. Roboti mõlemat ratast veab eraldi mootor läbi reduktori ja neid juhib kontroller.

LED’id ja nupud

Esimese asjana tutvusime LED’ide ja nuppude tööga. Selleks oli tarvis kirjutada lühikene koodijupp:

```
#define LEDPIN 7
#define BTNPIN1 1
#define BTNPIN2 5

int main(void)
{		DDRB &=0xDD;
		DDRD |=1 <<LEDPIN;
		PORTD = 0x00;

		while(1)
		{
			if((PINB & 0x02) == 0)
				PORTD |= 0x80;
			if((PINB & 0x20) == 0)
				PORTD &= 0x7F;
		}
return 0;
}
```

Koodi oluliste osana on protsessori sisendite ja väljundite määramine ning pinnide oleku määramine, millele järgneb vastavalt nupuvajutuste fikseerimisele LED’i sisse ja välja lülitamine.

PWM ja mootorid

Järgnevalt sai tutvutud mootorite ja nende juhtimisega PWM'i abil. PWM e. Pulse-Width-Modulation on funktsioon, mis põhimõtteliselt teeb digitaalsest signaalist analoogsignaali. Andes tihedalt kõrget pikka signaali – töötavad mootorid kiirelt; andes kõrget lühikest signaali – töötavad mootorid aeglaselt. Eeltoodud näites oli välja toodud kaks varianti, mille vahele jääb suur arv lisavariante ja kombinatsioone. PWM'i abil saab diskreetset signaali kasutades juhtida mootoreid suvaliste kiiruste ja nende muutustega.

Allpool PWM'i käivitamise kood koos kommentaaridega:

```
        // Seame DDRD's PD5 ja PD6 väljundiks
DDRD |= (1<<MOT1A) | (1<<MOT1B);
        // Seame PORTB's PD5 ja PD6 madalaks
PORTD &= ~((1<<MOT1A) | (1<< MOT1B));
        // Seame DDRD's PB3 ja PD3 väljundiks
DDRB |= (1<<MOT2B);
DDRD |= (1<<MOT2A);
        // Seame PORTB's PB3 ja PD3 madalaks
PORTD &= ~(1<<MOT2A);
PORTB &= ~(1<<MOT2B);
        // Seame, et tahame pinid madalaks saada kui Timer0 väärtus TCNT0 saab võrdseks OCRx'ga
TCCR0A |= (1<<COM0A1) | (1<<COM0B1);
        // Seame, et kasutada fast PWM mode'i
TCCR0A |= (1<<WGM00) | (1<<WGM01);
        // Seame TCCR0B registris, et Timer0 kellasignaal on F_CPU/8
TCCR0B |= (1<<CS01);
        // Seame, et tahame pinid madalaks saada kui Timer0 väärtus TCNT2 saab võrdseks OCRx'ga
TCCR2A |= (1<<COM2A1) | (1<<COM2B1);
        // Seame, et kasutada fast PWM mode'i
TCCR2A |= (1<<WGM20) | (1<<WGM21);
        // Seame TCCR2B registris et Timer0 kellasignaal on F_CPU/8
TCCR2B |= (1<<CS21);
```

Roboti liikumist sai juhtida järgnevalt käsklusega:

```
mootoriseaded(va, sa , vb, sb)
```

kus

va - on vasaku mootori kiirus

sa - vasaku mootori suund

vb - on parema mootori kiirus

sb - parema mootori suund

Selle käskluse taga töötas lühike koodijupp:

```
void mootoriseaded(uint8_t vkiirus, uint8_t vsuund,uint8_t pkiirus, uint8_t psuund)
{
	if (vsuund==1)
	{
		OCR0A = 255;
		OCR0B = 255-vkiirus;
	}
	else
	{
		OCR0A = 255-vkiirus;
		OCR0B = 255;
	}
		if (psuund==1)
	{
		OCR2A = 255;
		OCR2B = 255-pkiirus;
	}
	else
	{
		OCR2A = 255-pkiirus;
		OCR2B = 255;
	}
```

Jooneandurid

Jooneandurid töötavad IR sensoritel, mille kaks peamist osa on fototransistor ja IR LED. Anduri tööpõhimõte seisneb skeemi ühendatud kondensaatori tühjenemiseks kulunud aja pikkusel. Mida tumedam on pind seda aeglasemalt tühjeneb kondensaator; mida heledam, seda kiiremalt. Lähtuvalt kulunud ajast saab hinnata, kas pind on tume või hele. Selle tarvis tuleb vastavalt keskkonnale või oma soovile määrata ära aja, mis on piiriks tumeda ja heleda pinna vahel.

Andurilt signaali välja lugemine käib järgneva koodijupi abil:

```
	uint16_t PA()
	{
		uint8_t andur;
		DDRC |= 1<<PC2; // Sensori pin väljundiks
		PORTC |= 1<<PC2; // Sensori pin kõrgeks
		_delay_us(10);
		DDRC &= ~(1<<PC2); // Sensori pin sisendiks
		PORTC &= ~(1<<PC2); // Internal pullup välja lülitamine
		TCNT1=0; // Taimeri reset

		while((PINC & (1<<PC2)) && (TCNT1<4000)) //ootab senikaua, kui PC2 läheb nulliks
		{
		if(TCNT1>500)
		andur=1;
		else
		andur=0;
	}
```

Kaugusandur

Robotitele lisatud SHARP'i kauguseandur töötab sarnaselt jooneanduritele koosnedes IR LED'ist ja fototransistorist. Antud juhul mõõdetakse aga aega, mis kulub IR kiire tagasipeegeldumiseks fototransistorile ja väljastatakse signaal analoogpinge abil. Saadus signaal läheb läbi ADC (AnalogDigitalCoverter), mis muudab saadud signaali diskreetseks. Arvesse tuli ka võtta seda, et saadud signaal ei ole lineaarses seoses mõõdetud kaugusega ning selleks tuleb teha vastavalt, võrrand, mis võimaldaks saada lineaarseid väärtuseid sõltuvalt kaugusest.

Lisatud lühike koodijupp, mis demonstreerib anduri kasutamist.

```
int main(void){
		//initialize ADC7 pin4 adc value stored in ADCH range 0-255
	initializeADC();
			sei();
		//	initializeInterrupts();

		while(1){
			//local variable
			int adcValue = 0;

			//check for calculated ADC values, for not doing it every cycle
			if(voltageCounter > (arraySize - 15)){

				//forbid interrupts
				cli();

				adcValue = average(voltageAVG, voltageCounter);
			sei();
				{
				mootoriseaded(adcValue,1,adcValue,1); //mida lähemal on objekt, seda kiiremini liigub vastassuunas)
				}

			}
```

PID

Järgnevalt sai tutvutud PID'iga, mis on meie puhul tagasisidestatud matemaatiline diskreetaja kontroller, mis võtab väljundsignaali arvutamisel arvesse sisendite väärtuseid ja ka iga eelnevat väljundi väärtust. PID kontrolleri eesmärk on kiirelt ja lihtsalt vea korrigeerimine, mis antud juhul on liikuva roboti joonest kõrvalekalde eemaldamine. PID kontrolleril on kolm peamist komponenti:

- proportion
- integraal
- derivetive

Neid valisime tunnis vastavalt kiiruse muutumisele katse-eksituse meetodil. Kõike optimaalsem oleks luua matemaatiline tagasisidestatud mudel näiteks MATLAB'is ning saades kõikide komponentide sõltuvuse kiirusest arvesse võttes kõiki sisend- ja roboti parameetreid.

PID kontrolleri kood:

```
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);
}
```

Laburünt

Laburündi läbimiseks tuli eelnevad teadmised kasutusele võtta ja kombineerides eelnevalt välja toodud koodijuppe ühendada ja täiendada. Põhimõte raja läbimisel on lihtne:

- läbida rada esimesest korda vasak-käsi-seinal meetodil ja meelde jättes kõik pööramised, mis sai tehtud
- saadud pööramistest koostatud massiiv optimeerida, eemaldades ebavajalikud pööramised (tupikud)
- läbida rada läbi kõige otsemat teed, põhinedes optimeeritud tulemustel

Vasak-käsi seinal meetod seisneb sellel, et igal ristmikul valib robot võimalusel vasaku pöörde. Juhul kui vasakule pöörata pole võimalik, siis liigub otse. Kui ka otse ei ole võimalik

liikuda, siis võimalusel paremale või ka selle variandi puudumisel, pöörab otsa ringi ja sõidab tagasi. Kõik need valikud jätab robot meelde ja sisestab massiivi
Seda iseloomustab koodijupp:

```
char suunavalik(unsigned char vasakandur, unsigned char paremandur, unsigned char keskandur)
{
	if(vasakandur)
	return 'L';
	else if(keskandur)
	return 'S';
	else if(paremandur)
	return 'R';
	else
	return 'B';
}
```

Optimeerimist iseloomustab järgnev koodilõik, mis kutsutakse välja peale igat pööret/ristmikku. Optimeerimise käsku täidetakse vaid juhul kui massiiv „pooretearv“ on 3 või rohkem pööret ning eelviimane liige on „B“. Sellega on massiivist välja optimeeritud kõik üleliigsed pöörded tupikutesse.

```
void optimeerimine()
{
	if(pooretearv < 3 || poorded[pooretearv-2] != 'B')
	return;
	int nurk = 0;
	int i;
	for(i=1;i<=3;i++)
	{
		switch(poorded[pooretearv-i])
		{
			case 'R':
			nurk += 90;
			break;
			case 'L':
			nurk += 270;
			break;
			case 'B':
			nurk += 180;
			break;
		 }
	 }

	 nurk = nurk % 360;

	 switch(nurk)
		{
			case 0:
			poorded[pooretearv - 3] = 'S';
			break;
			case 90:
			poorded [pooretearv - 3] = 'R';
			break;
			case 180:
			poorded [pooretearv - 3] = 'B';
			break;
			case 270:
			poorded [pooretearv - 3] = 'L';
			break;
		}

	pooretearv -= 2;
}
```

Optimeeritud teekonna järgi läbib robot laburündi kõige otsemat teed pidi lugedes andmed massiivist:

```
while(1)
{
	int i;
	for(i=0;i<pooretearv;i++)
	{
		straight();
		mootoriseaded(50,1,50,1);
		delay_ms(100);
		turn(poorded[i]);
	}
}
```

Kokkuvõte

Antud aines õpitu andis võimaluse kombineerida mitmete eelnevalt õpitud õppeainete teadmisi ja andis võimaluse näha reaalseid rakendusi. Õppeaine kestel sai ka omale sama robot tellitud, et omistatud teadmisi ka aine lõppedes saaks rakendada ja täiendada. Järgmise eesmärgina on soov katsetada ka ekraani ja heligeneraatori juhtimist.