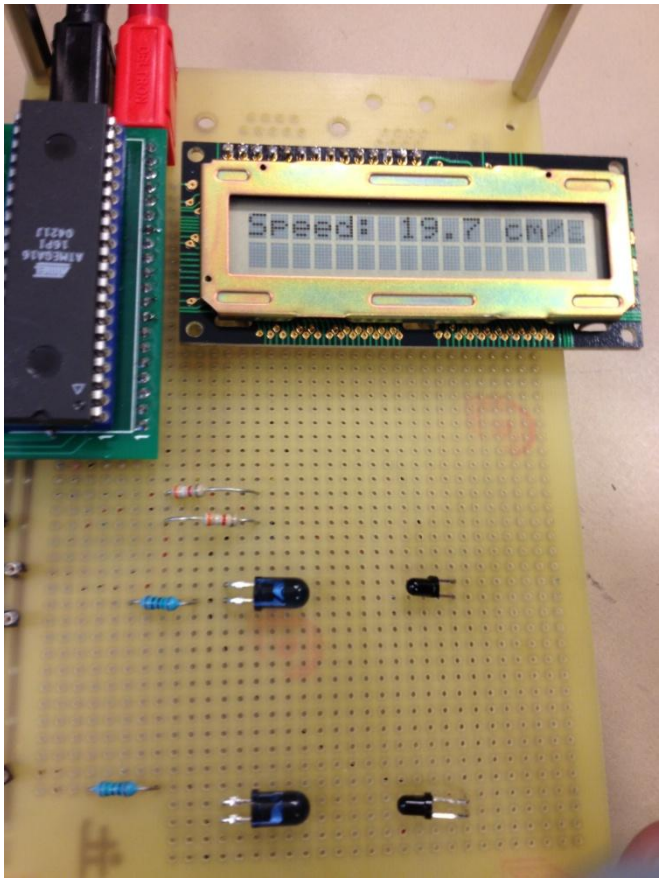
ELEKTRO- OCH INFORMATIONSTEKNIK

DIGITALA PROJEKT (EITF40) – Fartmätare

Handledare: Bertil Lindvall

Per Fernström, ie09pf7
Albin Nilsson, ie09an4

5/14/2012



Abstract

The purpose behind this project is to produce a speedometer. The speedometer is supposed to be able to measure the time with which an object travels between two sensors. The basic method is to measure the time that it takes in between the two sensors are activated. This time is then used together with the knowledge of the distance between the sensors to calculate the speed. In this report all the different components will be briefly described together with the way with which they have been connected to each other. The software code will also be briefly explained.

Innehållsförteckning

1. Introduktion

Hela projektet inleddes med att fundera ut vilka komponenter som skulle behövas. Därefter inleddes ytliga genomläsningar av varje komponents respektive datablad för att få någon slags introduktion till komponenten. Grundläggande saker så som vilka pinnar som gjorde vad utforskades. Därefter drogs en ritning/blockschema upp för att användas vid lödandet/virandet. Vidare testade vi ut en preliminär komponentplacering. Därefter började det spännande och stundom frustrerande arbetet att bygga ihop och programmera de olika komponenterna till en färdig fartmätare.

2. Kravspecifikation

Vårt projekt är att bygga en fartmätare. Den färdiga fartmätaren skall uppfylla samtliga nedanstående krav.

2.1 Hårdvarukrav

- Konstruktionen skall ha en display som kan visa siffror.
- Konstruktionen skall ha två sensorer som kan känna av när något rör sig framför dem.
- Konstruktionen skall ha en processor som kommer användas för att sköta tidtagning samt beräkning av hastighet.

2.2 Funktionella krav

- När fartmätaren är i startläge skall displayen visa ”Ready”
- När ett objekt far förbi sensorerna skall farten som objektet passerade sensorerna med visas på displayen med texten ”Speed: XX.X cm/s”.
- Meddelandet i displayen skall visas i tre sekunder. Under denna tid skall sensorerna inaktiveras.

3. Hårdvara

3.1 Processor

Som processor i vårt system har vi använt oss av en 8-bit AVR ATMega16. Det är till denna som alla andra komponenter kommer att kopplas och det är den som styr hela systemet. Processorn har ett 16 KB inbyggt flashminne. Vidare har den 40 pinnar varav 32 är programmerbara in-utlinjer fördelade på de fyra portarna A, B, C och D.

3.2 LCD-display

Fartmätaren skall kommunicera med användaren via en LCD-display. Displayen skall vara en alfa-numerisk display och kunna visa två rader med 16 tecken. Modellen vi valde var av sorten SHARP Dot-Matrix LCD.

3.3 IR-emitter

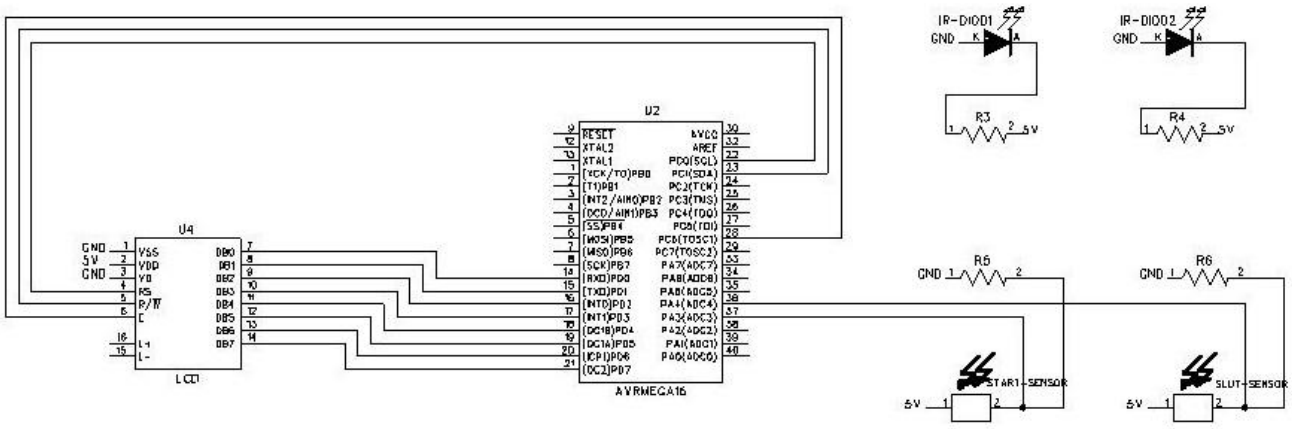
För att sända ut en IR-signal som skall brytas vid tidtagning används en IR-emitter. Den sänder en konstant IR-signal och är därför ej kopplad via processorn.

3.4 Fototransistor

För att ta emot IR-signaler har vi använt oss av en fototransistor. Den skickar en signal mellan noll volt och arbetsspänningen. Det innebär att när IR-signalen från emittern bryts så sjunker spänningen till noll för att sedan återgå till arbetsspänningen då signalen återvänder.

4. Blockschema

Nedan följer vårt blockschema efter vilket vi har kopplat ihop vår fartmätare, se figur 1.



Figur 1

5. Konstruktion

Fartmätaren byggdes med en central display med vilken all kommunikation med användaren görs. Här kan användaren få reda på om fartmätaren är på eller av, om den är redo att mäta en ny hastighet samt med vilken hastighet ett objekt har färdats förbi sensorerna.

Fototransistorerna kopplas till processorn på port A. Det är härifrån systemet hämtar information om det är tid att starta eller avsluta tidtagningen. Eftersom IR-emittrarna konstant skall vara påslagana kopplades dessa ej till processorn.

6. Mjukvara

Mjukvaran till systemet skrevs i programspråket C. Eftersom tid är väldigt centralt för vårt program har vi byggt upp det kring avbrott med en frekvens på 30 Hz. Det är med dessa avbrott som vi mäter tiden. Därför begränsas också tidtagningens precision till en tretiondels sekund.

Detta är något som eventuellt skulle kunna tänkas höjas för ännu högre precision. För att programmet skall kunna hålla koll på om man befinner sig i eller utanför en tidtagning används två globala variabler kallade ”ready” och ”running”. Dessa initieras med värde noll. Då processorn har skrivit ”ready” på displayen sätts också variabeln ready till ett. Detta är för att processorn inte skall skriva ”ready” vid fel tillfällen. Variabeln ”running” sätts till ett då en tidtagning startar.

7. Resultat

Fartmätaren har konstruerats enligt det blockschema som vi ritade i början av projektet. Viss förvirring uppstod innan vi insåg att processorns piggar i blockschemat inte stämde överens med den processor som vi hade i handen. Flera liknande missförstånd skulle följa och vid flera tillfällen under arbetets gång har vi insett att sladdar kopplats fel.

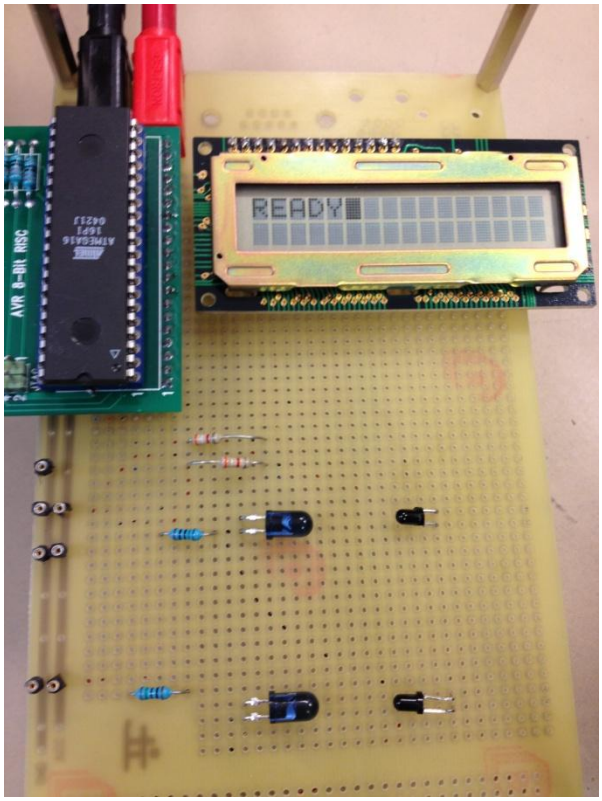
Att koppla in skärmen och få den till att fungera på ett önskvärt sätt var tämligen problematiskt. Förutom att vi insåg att vi virat sladdarna till processorn fel så insåg vi efter mycket möda att både displayen och processorn krävde två sladdar till jord. Vidare insåg vi att det ej räckte med en virning och var tvungna att löda sladdarna till spänningen. När vi väl lyckats koppla in displayen skulle det visa sig att den var trasig och tvungen att bytas.

Vid inkoppling av fototransistorerna visade det sig att vi initialt använt oss av för stora motstånd och var tvungna att byta ut dessa till mindre för att lyckas få en tillförlitlig signal till processorn.

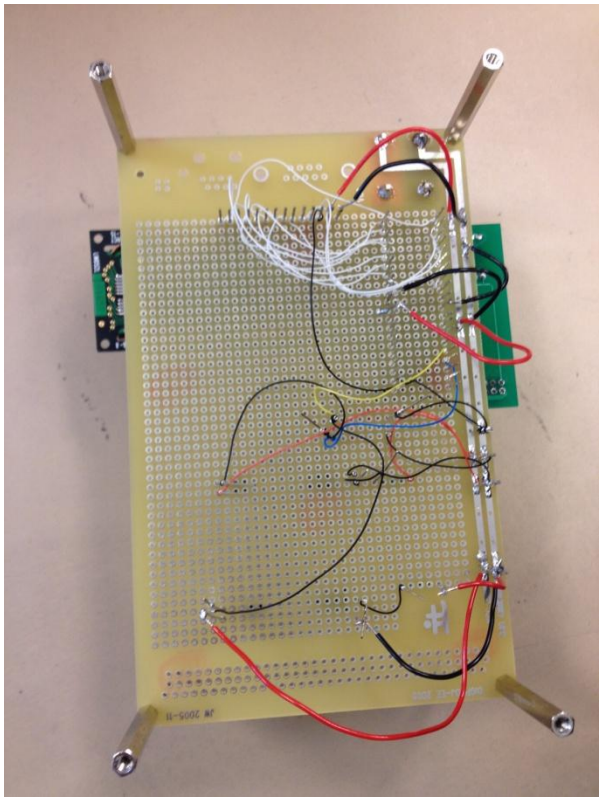
Jämfört med att hantera hårdvaran gick mjukvaruprogrammeringen tämligen smärtfritt. Vi hade initialt vissa problem med mjukvaran innan vi insåg att processorns kommunikation med AVR Studio är relativt instabil och kräver regelbundna omstarter av systemet. Ett annat problem som uppstod under vår programmeringsdel av projektet var att det ibland lös upp två pinnar då fototransistorerna aktiverades och ibland en. Detta gjorde att vår ursprungliga if-sats, som räknade med att endast en pinne skulle aktiverades, inte fungerade som önskat.

I övrigt var det främst att få igång lämpliga avbrott som ställde till vissa problem.

Allt som allt tycker vi dock att projektet gick bra. Vid vissa tillfällen kunde vi uppleva viss frustration men detta övergick snart i stor tillfredställelse då vi succesivt överkom samtliga problem. När fartmätaren var färdig och fungerade i enlighet med kravspecifikationen kände vi stor stolthet då vi båda påbörjade detta projekt utan några som helst förkunskaper. Den färdiga fartmätaren kan ses i figur 2.1 och 2.2.



Figur 2.1 Fartmätaren i ready-state


Figur 2.2 Fartmätaren med sladdarna exponerade

8. C-kod

Nedan följer källkoden som har använts i vår fartmätare.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000UL // 16 MHz
#include <util/delay.h>

volatile uint8_t count;
int running;
int ready;

void main()
{
	TCCR0|=(1<<CS02) | (CS01);  //Prescaler igång

	TIMSK|=(1<<TOIE0); //Overflow-interrupt igång

	sei(); // Sätter igång globala interrupts

	TCNT0 = 0;

	count = 0;

	//Initierar variabler
	running = 0;
	ready = 0;



	DDRC = 0x43; //Port C(0, 1 och 6) får riktning output
    DDRD = 0xFF; //Port D(alla) får riktning output



	display_init(); //Initierar LCD-skärmen



	//Evighetsloopen
	while(1){

		//Om ingen av sensorerna ger utslag skrivs "Ready" ut
		if ((PINA == 0xFF) && (running == 0)){
			if(ready == 0) {
				writeReady();
				ready = 1;
			}

		//Om startsensorn ger utslag startas klockan och displayen
rensas
		}else if(((PINA == 0xF7) && (running == 0)) || ((PINA ==
0xF3) && (running == 0)))  {
			startClock();
			display_clear();
			running = 1;
```

```
                //Om andra slutsensorn ger utslag och running = 1 så visas
hastigheten
                //och sedan återgår programmet till grundläge.
                }else if(((PINA == 0xEF) && (running == 1)) || ((PINA ==
0xCF) && (running == 1))){
                        showSpeed();
                        _delay_ms(1000);
                        display_clear();
                        running = 0;
                        ready = 0;
                }



        }


}

// 3.3 = avståndet mellan sensorerna, count är timerns tick, ger
hastigheten i cm/sek
void showSpeed(){
        double temp = count;
        double v = 3.3 / (temp/30);

        //Delar upp hastigheten i ensiffriga variabler
        int num1 = v / 10;
        int temp2 = (int) v;
        int num2 = (temp2 % 10);
        int dec = (v - (num1*10) - num2)*10;

        writeSpeed(num1, num2, dec);




}

//Skriver ut hastigheten som skickas in på displayen
void writeSpeed(int num1, int num2, int dec) {
        //Konverterar till char
        char cnum1 = (char) (((int) '0') +num1);
        char cnum2 = (char) (((int) '0') +num2);
        char cdec = (char) (((int) '0') +dec);

        display_writeCh('S');
        display_writeCh('p');
        display_writeCh('e');
        display_writeCh('e');
        display_writeCh('d');
        display_writeCh(':');
        display_writeCh(' ');
        display_writeCh(cnum1);
    display_writeCh(cnum2);
        display_writeCh('.');
    display_writeCh(cdec);
        display_writeCh(' ');
        display_writeCh('c');
        display_writeCh('m');
        display_writeCh('/');
        display_writeCh('s');
```

```
}


//Nollställer timern
void startClock(){
	count = 0;
}

//Detta utförs vid varje interupt
ISR(TIMER0_OVF_vect)
{
	count++;
}




void writeReady(){
	display_writeCh('R');
	display_writeCh('E');
	display_writeCh('A');
	display_writeCh('D');
	display_writeCh('Y');
}



//Används för att ge en pin(char pin) på porten (char port)
//ett speciellt värde(char state)
void set_pin(char port, char pin, char state){


	char set = 1 << pin;


	if(port == 'A'){
		set &= PORTA;
	if(set && !state){ // 1 -> 0
		PORTA ^= set;
	}
	if(set == 0 && state){ // 0 -> 1
		set = 1 << pin;
		PORTA ^= set;
	}
	}

	else if(port == 'B'){
		set &= PORTB;
	if(set && !state){ // 1 -> 0
		PORTB ^= set;
	}
	if(set == 0 && state){ // 0 -> 1
		set = 1 << pin;
		PORTB ^= set;
	}
	}
	else if(port == 'C'){
		set &= PORTC;
		if(set && !state){ // 1 -> 0
		PORTC ^= set;
	}
```

```
        if(set == 0 && state){ // 0 -> 1
                set = 1 << pin;
                PORTC ^= set;
        }
        }
        else if(port == 'D'){
                set &= PORTD;
        if(set && !state){ // 1 -> 0
                PORTD ^= set;
        }
        if(set == 0 && state){ // 0 -> 1
                set = 1 << pin;
                PORTD ^= set;
        }
        }
}

//Anropas när displayen skall ta emot ett kommando
void display_cmd(char val) {
    PORTD=val;
    _delay_ms(3);
    set_pin('C', PC1, 0); //RW -> 0
    set_pin('C', PC0, 0); //RS -> 0
    _delay_ms(3);
    set_pin('C', PC6, 1); //E -> 1
    _delay_ms(3);
    set_pin('C', PC6, 0); //E -> 0
}

//Anropas när displayen skall skriva ut ett tecken
void display_writeCh(char val) {
    PORTD=val;
    set_pin('C', PC1, 0); //RW -> 0
    set_pin('C', PC0, 1); //RS -> 1
    _delay_ms(3);
    set_pin('C', PC6, 1); //E -> 1
    _delay_ms(3);
    set_pin('C', PC6, 0); //E -> 0
}


void display_clear() {
    display_cmd(0x01); //Skickar in kommandot som rensar displayen
    display_cmd(0x38); //Functions set
}

void display_init() {
    display_clear();
    display_cmd(0x0F); //Skickar in kommandot som sätter på displayen
    display_cmd(0x06); //Entry mode set
}
```