RADIOSTYRD BIL

Projektrapport i Digitala projekt EITF11

Institutionen för Elektro- och informationsteknik, Lunds tekniska högskola

Victor Girardin (I-10), Rickard Olsson (I-10) & Staffan Theander (I-09)
Handledare: Bertil Lindvall

20 maj 2013

Abstract

This project was made in the digital projects course, EITF11, at LTH during the spring of 2013.

The purpose of the course is to give students a hands-on introduction to electronic and digital systems development, through the means of building a electronic device of some kind. Our choice was to build a simple car controlled by a remote control.

The requirements were that it would be able to drive forwards, backwards, right and left after receiving commands from the remote control.

This report will give a summary of the construction and programming period, and in it we discuss some of the problems we encountered and how we solved those. In hindsight it has been a great experience which we all agree has been very rewarding.

Innehåll

1 Inledning

Med det här projektet avsågs att öka förståelsen för ellära, elektronik och programmering i C, men framförallt i drivandet av ett digitalt projekt. Projektet gick ut på att bygga och programmera en enkel radiostyrd bil enligt nedanstående kravspecifikation och under processen utveckla den problemlösande förmågan. De största svårigheterna bestod i att förstå det omgivande bruset av radiosignaler och att hitta ett sätt att anpassa radiokommunikationen därefter. Detta löstes genom att kommunicera med signalstyrka som urskiljande attribut.

På grund av tidskrävande arbete med radiokommunikation begränsades bilen till att uppfylla de mest basala funktionerna som att köra framåt, bakåt och svänga.

Rapporten syftar till att redovisa projektets utförande och resultat samt diskutera lärdomar och bilens förbättringsmöjligheter. För att underlätta läsning och förståelse finns även en teoridel som beskriver hårdvara och mjukvara som använts och skapats under projektet, samt bilagor för noggrannare studier av resultat och utförande. Författarna hoppas bjuda på intressant läsning för parallellt läsande kursdeltagare och underlätta arbetet för kommande kursdeltagare.

2 Teori

2.1 Kravspecifikation

De ursprungliga kraven såg ut enligt följande:

1. Bilen ska kunna köra framåt, bakåt och kunna svänga vänster och höger.
2. Bilen ska kunna bromsa.
3. Bilen ska styras från en extern styrplatta via radiosignaler.
4. Bilen ska ha en tuta eller en högtalare av något slag.
5. Bilen ska utrustas med blinkers. Respektive blinker ska aktiveras när bilen svänger.

På grund av tidsbrist, till följd av stora problem med att få radiosändaren att fungera tillförlitligt, utgick krav 4.

2.2 Hårdvara

2 st. processorer, ATmega16:
8 bitars mikrokontroll med 16 Kbyte programmerbart flashminne. Processorn har 40 pinnar, varav pinne 3 och 4 (AIN0 resp. AIN1) visade sig vara speciellt intressanta för projektet.

2 st. radiosändare/mottagare (433 MHz RF Tranceiver(Parallax)):
Fungerar både som mottagare och sändare av radiosignaler på frekvensen 433 MHz. Tranceivern har sex pinnar, bland annat en för att mäta signalstyrkan (RSSI).

1 st. Spänningsregulator (LP3852ET-5,0):
Reglerar spänningen från 6V in i komponenten till 5V ut från komponenten. Komponenten är essentiell då processorn kräver 5V.

1 st. Dubbel H-brygga (L298n):
Består av två bryggor som styr varsin motor. H-bryggan styrs av logiska signaler från processorn men får samtidigt en högre spänning direkt från batteriet(6V) som sedan skickas vidare till motorerna. Varje brygga har tre pinnar anslutna till processorn, en enable och två input. Input styr vilket håll som hjulen ska snurra medan en aktiverad enable-pinne medför att motorn får ström.

2 st. 6V motorer:
Två motorer sammankopplade genom en plastplattform. Till motorerna är larvfötter kopplade som medför bilens framfart.

1 st. 6V batteri:
Fem stycken 1,2 V laddningsbara seriekopplade batterier.

1 st. Debugger (AVR JTAGICE mkII):
Används för att ladda upp exekverbar kod på processorerna, med hjälp av AVR studio 4.

Dessutom användes fyra knappar, tre lysdioder samt talrika resistorer och kondensatorer med olika stora motstånd respektive kapacitanser.

2.3 Mjukvara - hjälpmedel

AVR Studio 4:
Programmet möjliggör kompilering och överföring av kod, skrivet i C i projektet, till processorerna. Dessutom underlättas testning av processorernas kopplingar till andra hårdvarukomponenter genom att det manuellt går att aktivera pinnar på processorn.

Power Logic
Program för att rita kretsscheman.

2.4 Mjukvara - skriven

2.4.1 Bilenhet

main:
Main-metoden innehåller en oändlig loop som ständigt tar emot och behandlar signaler via metoden readBits (se nedan) och sedermera motorControl (se nedan).

readBits:
I metoden readBits jämförs och matchas antalet ettor per tusen klockcyklar mot olika kommandon i motorControl. Dessutom styrs lamporna(vänster och höger) i metoden genom att tillhörande pinnar aktiveras i samband med att vänster- och högerkommandon skickas till motorControl.

motorControl:
Här utförs kommandona framåt, vänster, höger och backa genom att pinnar kopplade till H-bryggan aktiveras i olika kombinationer.

I bilaga A.1 ett presenteras hela koden för bilenheten.

2.4.2 Kontrollenhet

main:
Här ligger en oändlig loop som känner av knapptryckningar på kontrollen och beroende på vilken knapp som är nedtryckt skickar vidare olika signaler till sendBits.

sendBits:
Här skickas olika antal ettor eller nollor beroende på insignaler från mainmetoden.

I bilaga A2 presenteras hela koden för kontrollenheten.

3 Metod

3.1 Förberedelser

Under den första läsperioden under VT13, besöktes ett antal föreläsningar och laborationer som introduktion till elektronik, ellära och programmering i C, då projektgruppens medlemmar saknade erfarenhet av detta.

3.2 Idégenerering och framtagning av kravspecifikation

Projektgruppen var tidigt överens om att det var en radiostyrd bil som skulle byggas, och frågan blev således hur avancerad den skulle vara och vilka funktioner den skulle ha. I slutändan bestämdes det, att en ganska simpel bil skulle konstrueras och att extra funktioner skulle läggas till i mån av tid. Planen visade sig vara klok, då vissa moment tog längre tid än förväntat.

3.3 Kretsschema

Första steget i uppfyllandet av kravspecifikationen utgjorde skapandet av två kretsscheman. För tre ingenjörsstudenter med begränsade, för att inte säga näst intill obefintliga kunskaper inom elektronik och ellära blev detta en utmaning. Datorprogrammet, Power Logic som var avsett att underlätta arbetet, var inte så lättbegriplig och utgjorde till att börja med egentligen snarare ett hinder än ett hjälpmedel. Men med ständigt ökad förståelse för både program och ellära, uppdaterades och förbättrades kretsscheman kontinuerligt till att i slutet av projektet gestalta modeller av en fungerande radiostyrd bil med tillhörande styrenhet.

Det ska tilläggas att figurerna i kretsscheman (se bilagorna B1 och B2) inte alltid motsvarar den enhet som det faktiskt är, detta beroende på bristande databas i Power Logic.

3.4 Byggande

Precis som med kretsschemat så ändrades och uppdaterades den fysiska designen kontinuerligt. Den första versionen av grundstrukturen stod färdig tre veckor efter projektstart. Den fungerade dessvärre inte alls. Bland annat insåg gruppen att knapparna inte fungerade som trott och att radiokommunikationen inte fungerade. Det senare berodde dels på störningar i luften i e-huset och dels på den inte allt för högt upplevda kvaliteten på sändaren/mottagaren. Efter att ha löst problemet med knapparna och testat en tre - fyra sätt att lösa radiokommunikationen på, stod den slutgiltiga designen klar i början av vecka sex. Den stora skillnaden gentemot ickefungerande konstruktioner, var att RSSI-pinnen på radiomottagaren kopplades till en av de spänningsjämförande pinnarna (AIN0 och AIN1) på processorn.

3.5 Kodning

I och med problemen med radiokommunikationen, skrevs ett stort antal versioner av koden till både bilen och sändaren. Till en början fanns tron att det skulle gå att skicka bokstäver (i form av bytes) med hjälp av USART, men det visade sig vara en falsk förhoppning. Nästa försök handlade om att bara räkna ettor och nollor från den skickade data, men då uppstod det stora problem i form av störande brus. För att hantera bruset skrevs diverse kalibreringsmetoder och därpå beroende metoder för att hantera procentandelar ettor. Detta visade sig fungera till den grad att det gick att skicka två skilda kommandon, dock osäkert och med varierande resultat.

Till slut löstes problemet med hjälp av RSSI-pinnen på radiomottagaren och processorns ACSR-register som möjliggjorde det att jämföra två pinnars spänning.

Registrets ACO-bit sattes till high om första pinnen hade högre spänning än den andra, och till low om det var omvänt förhållande. En av jämförelsepinnarna kopplades därför till radiomottagarens RSSI-pinne och den andra kopplades med hjälp av ett par resistorer till en ganska konstant spänning på runt 1 V. Eftersom spänningen på RSSI-pinnen var på mellan 0,6 och 0,8 V då kontrollenheten inte skickade några signaler, men ökade till ca 1,5 V då den faktiskt skickade, gick det att avgöra om kontrollenheten försökte kommunicera eller inte, genom att avläsa ACO-biten. Som kommunikationsprotokoll användes en enkel metod som

gick ut på att skicka ett bestämt antal ettor (Kontrollenheten kommunicerar) per 1000 klockcyklar. Eftersom bilen endast skulle ha 6 (fick 5 till slut) kommandon gick det bra att låta bilens processor utföra samma operationer inom stora intervall av antal mottagna ettor. Sändare skickade därför lika många ettor som motsvarade intervallets mitt, vilket medförde en ganska säker och störningsfri kommunikation. Tester visade att 1000 klockcyklar var mer än tillräckligt, men då 1000 klockcyklar motsvarade en i sammanhanget försvinnande liten tid, beslöts det att kommunikationen fick jobba med de ursprungliga 1000 klockcyklarna.

4 Resultat

Eftersom radiokommunikationsbiten krävde mer tid än väntat, togs en funktion bort, nämligen tutan, då detta inte hanns med. Annars uppfylldes samtliga krav och en färdig prototyp för en radiostyrd bil skapades. Bilen styrs med hjälp av fyra knappar på en stationär kontrollenhet och fungerar på ett par meters avstånd från kontrollenhetens radiosändare. Signalen når bilens radiomottagare även om de skickas genom föremål av inte allt för tjocka material, men fungerar som bäst då sändaren riktas direkt mot bilens mottagare utan att något är i vägen. Tack vare bilens larvfötter klarar den av lättare terrängkörning.

Den för projektet kritiska biten, radiokommunikationen, löstes med signalstyrkevariationer istället för att avläsa bit-sekvenser från mottagarens datapinne.

Koden är kort och koncis, dels då bilen inte har så många funktioner och dels för att koden är skriven med avseende på signalstyrka och dessutom inte hanterar bokstäver eller andra bit-sekvenser. Kommandon skickas från kontrollenheten som andelar ”ettor” per tusen klockcyklar. Med den koden finns det möjlighet att öka antalet olika kommandon och därmed funktioner om man skulle vilja.

5 Diskussion

Efter genomförandet av kursen Digitala Projekts praktiska del, kan det konstateras att tre inom området okunniga studenter har lyckats med uppgiften att konstruera en radiostyrd bil. Vägen dit var allt annat än rak, men medförde utmaningar och frågeställningar som löstes med varierande framgång. Först och främst har förståelsen för elektronik och ellära gått från näst intill obefintlig till duglig gällande utförande av enklare projekt inom ämnena. Av större vikt är de nya erfarenheterna och upplysningarna kring hur information tas fram och sållas vid arbetet med ett digitalt projekt samt hur ett projekt, fullständigt utan administrativ handledning, startas, drivs och avslutas.

Tack vare handledarens ovilja att svara på icke-konkreta frågor, har även förmågan att ställa just konkreta frågor förbättrats. Detta medförde föga överraskande att de flesta problemen gick att lösa utan handledarens hjälp och gav även insikten att konkretisering och formulering av problem är otroligt viktigt.

Gällande projektets innehåll, fanns det egentligen bara en större flaskhals. Den utgjordes av problemen kring radiokommunikationsenheterna. Respekten och förståelsen för störningar i signaler har ökat men även tron på den egna problemlösande förmågan. Hade möjligheten och tiden funnits till att utveckla bilen vidare, finns det många förbättringar som kan göras. Bland annat kan ett snävare protokoll för radiokommunikationen skrivas för att dels öka hastigheten på kommandons utförande och dels för att lägga till funktioner. Funktionerna som ligger närmast till hands, är möjligheten att reglera hastigheten och att det ska gå att trycka på två knappar samtidigt så att det t.ex. går att backa och svänga samtidigt.

Vidare hade blinkerfunktionen kunnat förbättras så att oönskade blinkningar tas bort och kontrollenheten hade kunnat göras batteridriven.

Sammanfattningsvis, är det tre nöjda studenter med blodad tand, som ser fram emot påföljande kurser inom samma område. Det har varit en lärorik period, med nya lärdomar som upplevs som mer värdefulla inför arbetslivet, än de som fås av mer teoretiska kurser.

6 Referenslista

ATmega16 (processor), datablad:
`http://www.atmel.com/Images/doc2466.pdf`
L298N (H-brygga), databad:
`https://www.sparkfun.com/datasheets/Robotics/L298_H_Bridge.pdf`
Transceiver, datablad:
`http://www.wulfden.org/downloads/manuals/27982-433MHzRFTransceiver-v1.1.pdf`
Spänningsregulator, datablad:
`http://www.eit.lth.se/fileadmin/eit/courses/edi021/datablad/Analog/voltage/lp3855.pdf`

A Kod

A.1 Bilenhet

```
#include<avr/io.h>
#include<util/delay.h>
#define F_CPU 1000000UL
int speed;
void readBits(int counter_1)
{
    if (counter_1 > 200 && counter_1 < 400) {
        PORTC = (1 << PC0);
        motorControl(2);
    } else if (counter_1 > 400 && counter_1 < 600) {
        motorControl(1);
    } else if (counter_1 > 600 && counter_1 < 800) {
        PORTC = (1 << PC1);
        motorControl(3);
    } else if (counter_1 > 800 && counter_1 < 1000) {
        motorControl(4);
    } else {
        PORTC = 0x00;
        motorControl(0);
    }
}

void motorControl(int cmd)
{
    if (cmd == 1) {
        PORTD = 0 b10110100;
        if (speed == 1) {
            PORTD = 0 b10110100;
            speed = 0;
        } else {
            PORTD = 0x00;
            speed = 1;
        }
    } else if (cmd == 2) {
        PORTD = 0x30;
        if (speed == 1) {
            PORTD = 0x30;
            speed = 0;
        } else {
```

```
            PORTD = 0x00;
41          speed = 1;
        }
43  } else if (cmd == 3) {
        PORTD = 0x84;
45      11i f(speed == 1) {
            PORTD = 0x84;
47          speed = 0;
        }
49      else {
            PORTD = 0x00;
51          speed = 1;
        }
53  } else if (cmd == 4) {
        PORTD = 0 b11011000;
55      if (speed == 1) {
            PORTD = 0 b11011000;
57          speed = 0;
        } else {
59          PORTD = 0x00;
            speed = 1;
61      }
    } else {
63      PORTD = 0x00;
    }
65 }

67 void main(void)
  {
69    DDRC = 0x03;
    ACSR = 0x00;
71    int counter_1 = 0;
    int totalsum = 0;
73    DDRD = 0xFF;
    speed = 0;
75    while (1) {
        if (ACSR & (1 << ACO)) {
77          counter_1++;
        }
79      totalsum++;
        if (totalsum > 999) {
81          readBits(counter_1);
            totalsum = 0;
```

```
83                counter_1 = 0;
            }
85      }
    }
```

A.2 Kontrollenhet

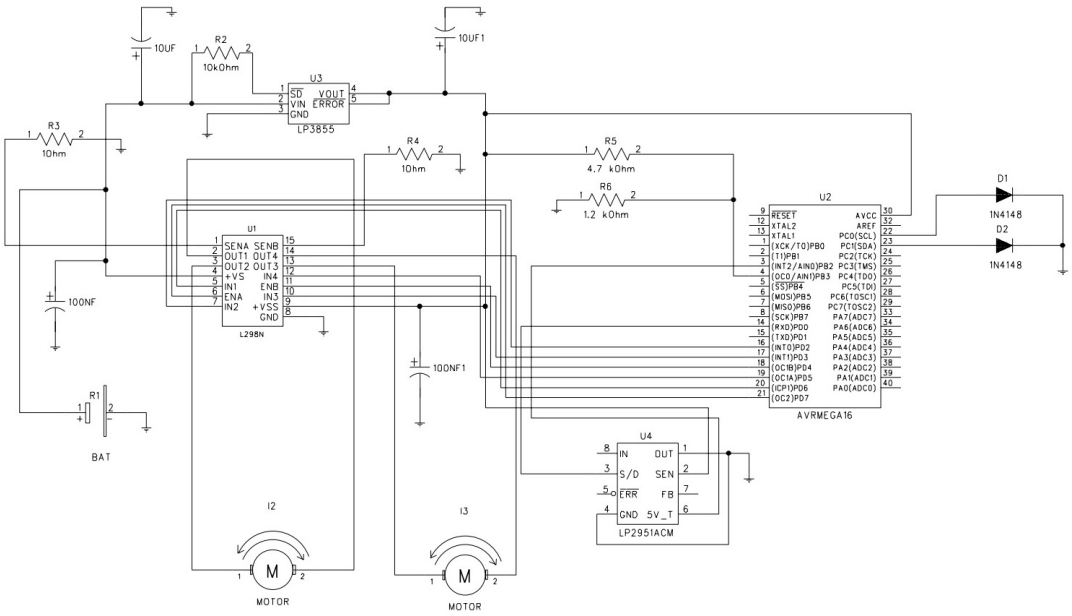
```
#define F_CPU 1000000UL
#include<avr/io.h>
#include <util/delay.h>
void sendBits(int type, int times)
{
    int counter = 0;
    while (counter++ < times) {
        if (type == 1) {
            PORTD = 0xFF;
        } else {
            PORTD = 0x00;
        }
    }
}

void main(void)
{
    DDRB = 0x00;
    DDRD = 0xff;
    PORTD = 0x00;
    int pressed;
    while (1) {
        pressed = 0;
        if (PINB & 0x01) {
            sendBits(1, 500);
            sendBits(0, 500);
            pressed = 1;
        }
        if (PINB & 0x02) {  //PB1 hi
            sendBits(1, 300);
            sendBits(0, 700);
            pressed = 1;
        }
        if (PINB & 0x04) {  //PB2 hi
            sendBits(1, 700);
            sendBits(0, 300);
            pressed = 1;
        }
        if (PINB & 0x08) {  //PB3 hi
            sendBits(1, 900);
            sendBits(0, 100);
```

```
42                  pressed = 1;
            }
44          if (pressed == 0) { //No button press
                PORTD = 0x00;
46          }
        }
48  }
```

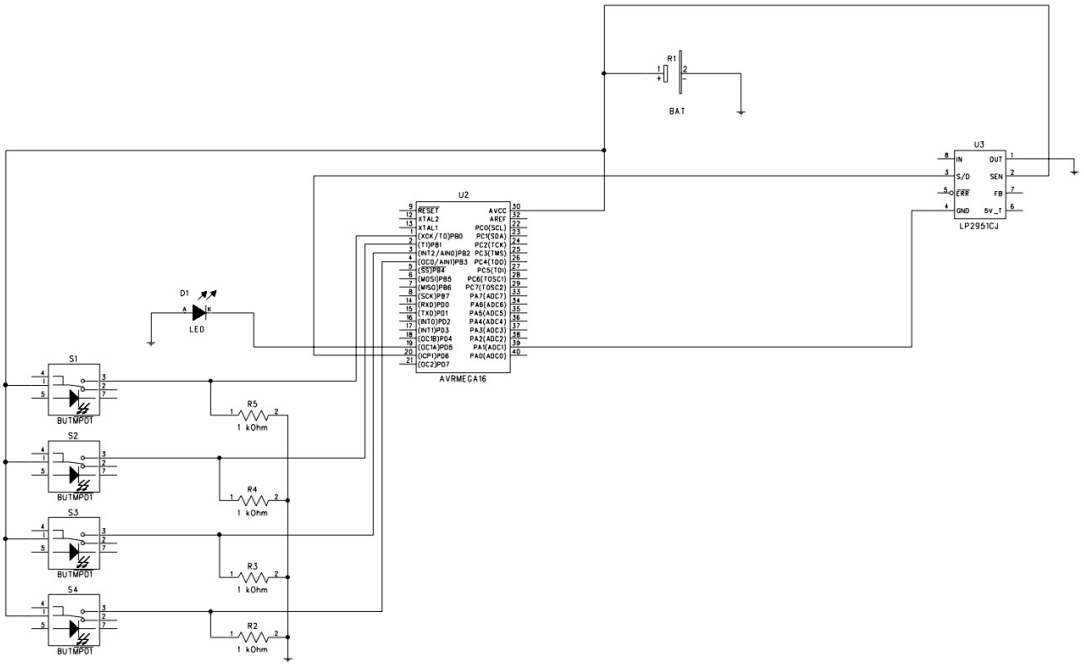
B Kretsscheman

B.1 Bilenhet



Figur 1: Kretsschema till bilenheten i färdigbyggt skick.

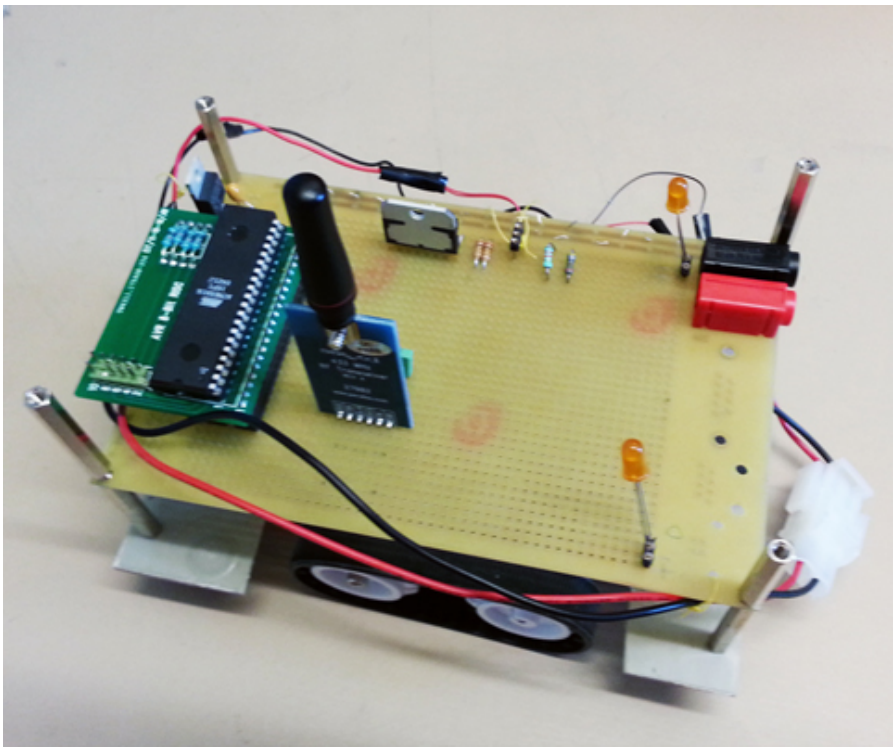
B.2 Kontrollenhet



Figur 2: Kretsschema till kontrollenheten i färdigbyggt skick.

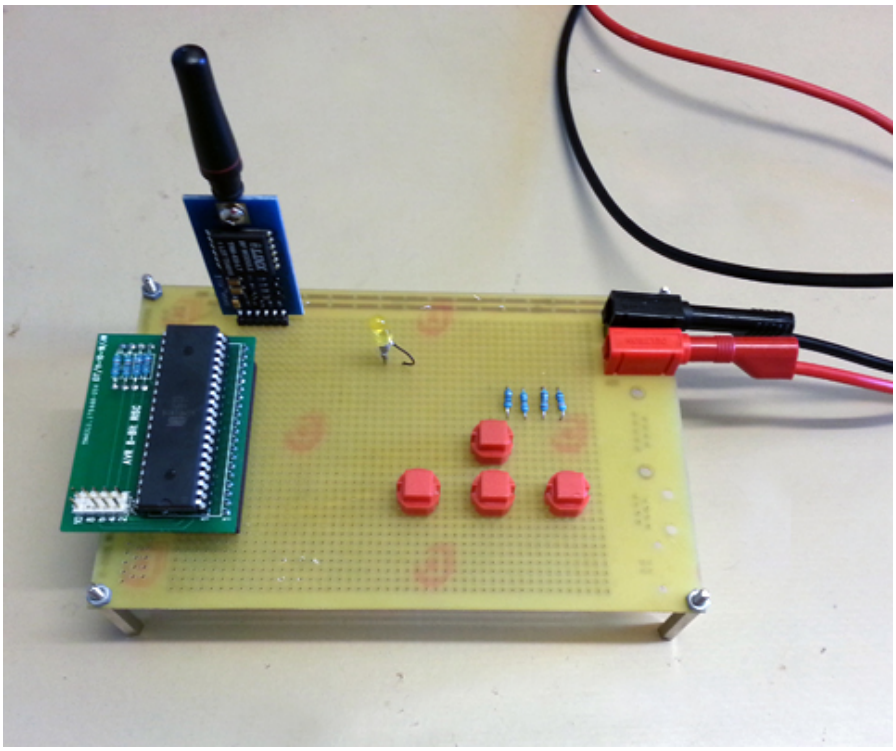
C Bilder

C.1 Bilenhet



Figur 3: Bild på bilenheten i färdigbyggt skick.

C.2 Kontrollenhet



Figur 4: Bild på kontrollenheten i färdigbyggt skick.