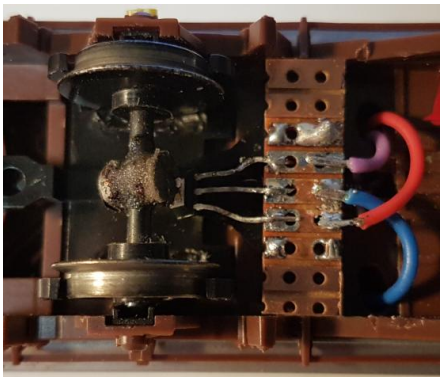
Målevogn

Ved anvendelse af WindigPet PC program er det muligt at have såkaldte intelligente stop ved fx stationer. For at få det, er det nødvendigt at angive de enkelte sektioners længde. Målevognen er fremstillet for at kunne udføre målingerne af sektionernes længde, særlig ved skjulte sektioner. Målevognen kan måle aktuel hastighed i skala H0 og strækninger i cm. Målevognen har ikke indbygget display, men sender alle data til en modtager med display. Derved er det mulig at få alle måleoplysningerne fra såvel synlige som skjulte spor. Fra modtageren kan strækningsmålingen resettes. Alle målinger bliver beregnet i målevognen og det er resultaterne af disse beregninger, der sendes til modtageren.

Vogn med sender

Selve senderen og måling er opbygget med en Arduino Nano og et sendemodul NRF 24L01. Til aftastning af hastighed og længde er der anvendt et Hall element der bliver påvirket af 2 magneter fastlimet på en aksel.



Figur 1 Hall element

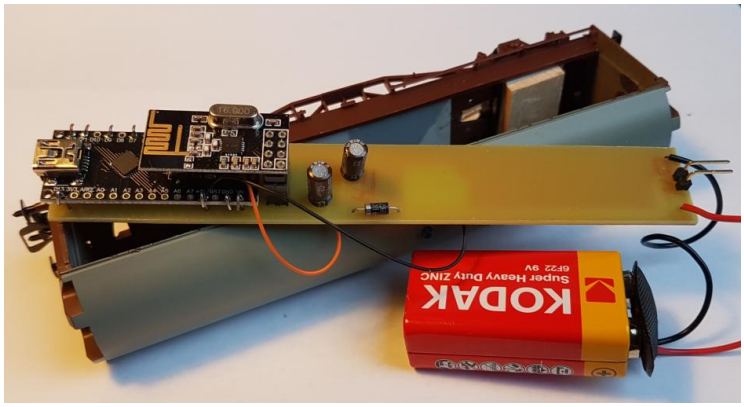
Forsyningen er fra et 9V batteri. Tænd og sluk foretages med en jumper der påsættes stikben, som er ført ud af den ene ende af vognen.

Figur 2 Stikben

Figur 3 Stikben med jumper

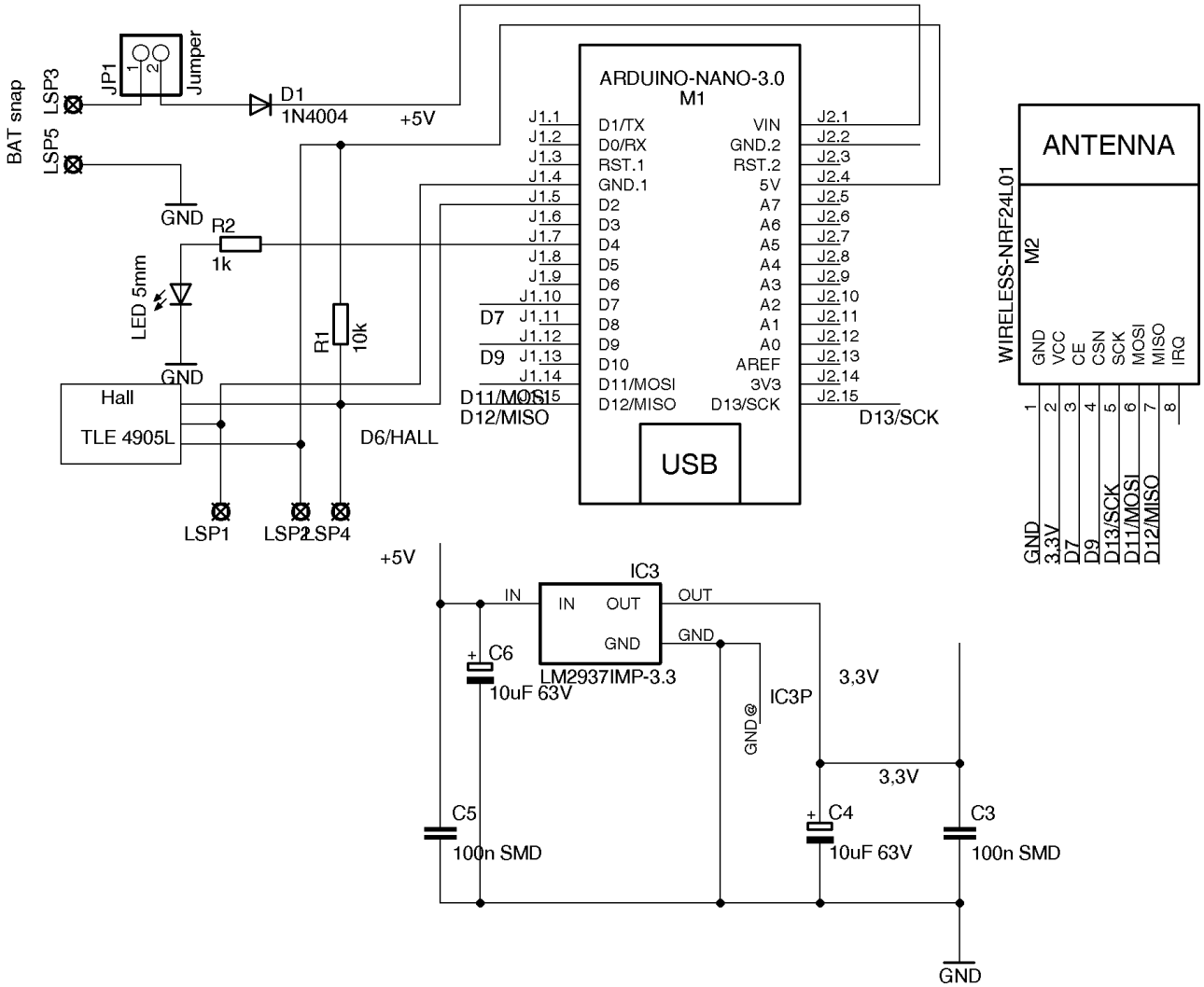
Figur 4 Set indvendigt

Her ses print med Arduino Nano og sender NRF 24L01. Stifterne til jumperen ses også på billedet. USB stikket til Arduinoen er ligeledes ført ud af vognens gavl.

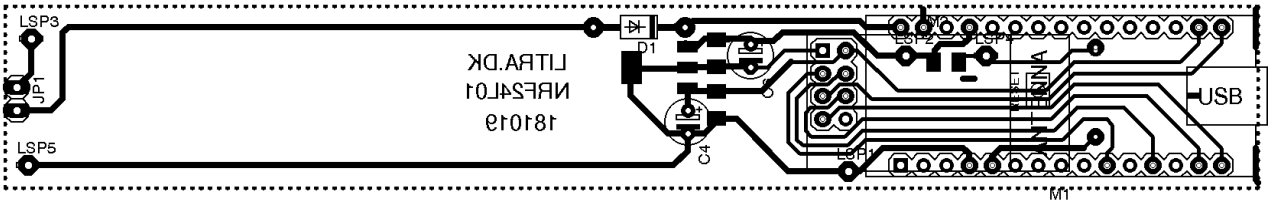


Figur 5 Print og batteri

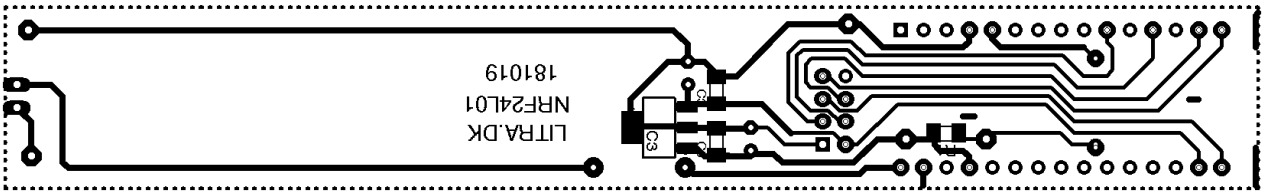
Diagram



Print



Komponentside



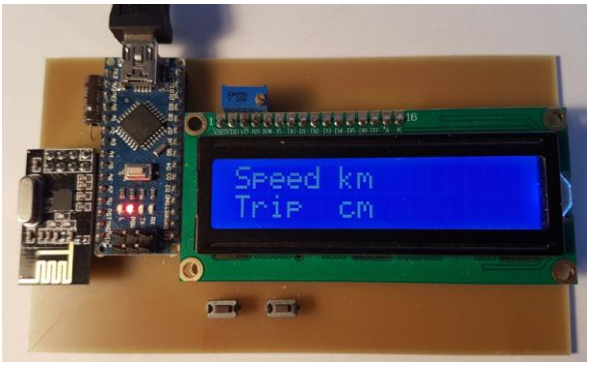
Printside

Stykliste

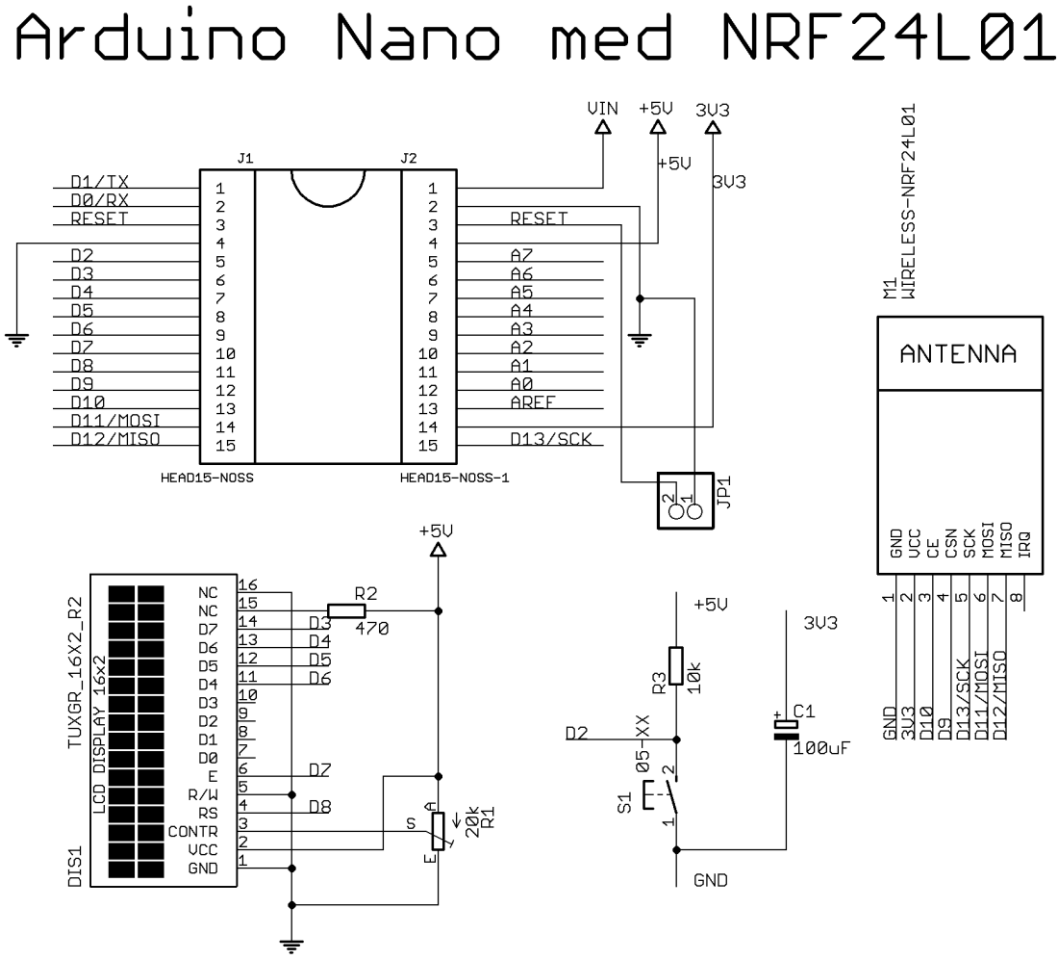
Qty	Value	Parts
2	100n SMD	C3, C5
1	10k SMD	R1
2	10uF 63V	C4, C6
1	1N4004	D1
1	1k	R2
1	ARDUINO-I	M1
1	Jumper	JP1
1	LED 5mm	LED1
1	LM2937IM	IC3
1	WIRELESS-	M2
1	2 pol stift 90gr	
1	8 pol sokkel spec.	

Modtager

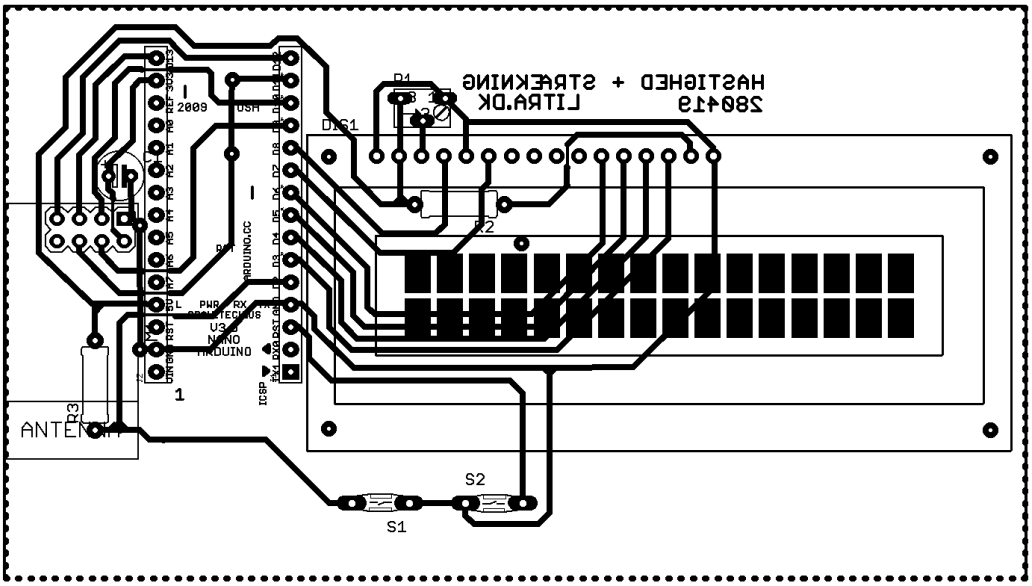
Modtageren består ligeledes af en Arduino Nano og NRF 24L01 samt et display. På modtageren er et tryk til reset af strækningen. Når trykket aktiveres sender modtageren signal til senderen og derved resættes strækningsmålingen. Modtageren forsynes med spænding gennem USB kabel tilsluttet en oplader fra fx en mobiltelefon.



Diagram



Print



Stykliste

Qty	Value	Parts
1	Reset tryk	JP1
1	Tryk	S1
1	100uF	C1
1	10k	R3
1	20k trimmer 10 turn	R1
1	470	R2
1	Arduino Nano	
1	WIRELESS-NRF24L01	M1
1	TUXGR_16X2_R2	DIS1

Arduino software

Sendevogn

```
// Sender 2 numeriske vÃ¦rdier 20-10-19 med Hall
// Med NRF24L01 sender og modtager
// Haakon Hansen for Litra.dk

#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>

const unsigned int hjul = (17.8756622*1.063);
unsigned long counter = 0;
const int ledPin =  4;
int ledState = 0;
unsigned long starttid, interval, tid, start, ftid ,tidnu, ventetid;
unsigned int data[2];
unsigned long hast, afstand;
const byte HallPin = 2 ;
boolean S1data = 0;
bool vtid = false;
bool nulstil = false;
bool beregnflag = false;
bool timer1 = false;

RF24 radio(7, 9); // Gl. vogn (10,9)
//const uint64_t pipe = 0xE8E8F0F0E1LL;
const byte addresses [][6] = {"00001", "00002"};


void setup(void) {
  start = millis();
  starttid = millis();
  interval = 3000;


Serial.begin(9600);
radio.begin();
radio.openWritingPipe(addresses[0]);     //Setting the address at which we will send the data
radio.openReadingPipe(1, addresses[1]);
pinMode (ledPin, OUTPUT);
pinMode ( HallPin , INPUT) ;
attachInterrupt ( digitalPinToInterrupt ( HallPin ), beregn, RISING ) ;
}

void loop(void)
{
delay(5);
  radio.startListening();               //This sets the module as receiver
  if (radio.available()) {
  radio.read(&S1data, sizeof(S1data)); //Reading the data

  if (S1data == LOW)
  {
    counter = 1;    //0 stiller triptÃ¦ller
    afstand = 0;
    hast = 0;
  }
  }

delay(5);
radio.stopListening();

if(vtid){
  tidnu = millis();
  if((tidnu-ventetid)>=500){
     ventetid = tidnu;
    senddata();
   // Serial.println("  sender  ");
    if (ledState == LOW) {
```

```
      ledState = HIGH;
    } else {
      ledState = LOW;
    }
     vtid = false;

  }
}

if(hast==0) ledState = HIGH;
 digitalWrite(ledPin, ledState);


// Nulstil ved stop ******************************
if(nulstil){
  if((millis()-starttid)>=interval){
    starttid = millis();
    }else{
      starttid = millis();
      }
  nulstil = false;
}else{

  if((millis()-starttid)>=3000){
  hast = 0;
  senddata();
  starttid = millis();
 }
}
}

void beregn()
{
  vtid = true;
  nulstil = true;
  if(beregnflag){
  tid=millis()-start;
  start = millis();
  hast =((608 *hjul)/tid);
  }
  beregnflag = !beregnflag;
// Serial.print("Beregner");
 // Afstandsberegning
  counter++;
 // counter = 50000;
afstand = (counter * hjul/10)*1.0;
  // afstand = afstand / 10;
}

void senddata()
{
data[0] = hast;
data[1] = afstand;
// Serial.print("    Sendt hast      ");
// Serial.println(hast);

radio.write(data, sizeof(data));

}
```

Modtager

```
// Modtager 2 numeriske værdier 02-7-19
// Med NRF24L01 sender og modtager
// Haakon Hansen for Litra.dk

#include <LiquidCrystal.h>
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>

const int rs = 8, en = 7, d4 = 6, d5 = 5, d6 = 4, d7 = 3;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

int printflag;
int data[2];
bool done = false;
const int S1 = 2;
int S1data = 0;

RF24 radio(10, 9);
//const uint64_t pipe = 0xE8E8F0F0E1LL;
const byte addresses [][6] = {"00001", "00002"};


void setup(void) {
 noInterrupts();           // disable all interrupts

  TCCR1A = 0;
  TCCR1B = 0;
  TCNT1  = 0;


  OCR1A = 124976;            // 115312 compare match register 16MHz/256/2Hz
  TCCR1B |= (1 << WGM12);   // CTC mode
  TCCR1B |= (1 << CS12);    // 256 prescaler
  TIMSK1 |= (1 << OCIE1A);  // enable timer compare interrupt
  interrupts();             // enable all interrupts

Serial.begin(9600);
pinMode(S1, INPUT);

radio.begin();

 radio.openWritingPipe(addresses[1]);
 radio.openReadingPipe(1, addresses[0]);

radio.startListening();
lcd.begin(16, 2);
//lcd.backlight();
lcd.clear();
lcd.print("Speed & trip");
delay(1000);
lcd.clear();
lcd.print("Starter.....");
delay(1000);

lcd.clear();
delay(500);
lcd.setCursor(0, 0);
lcd.print("Speed km ");
lcd.setCursor(0, 1);
lcd.print("Trip  cm ");
lcd.setCursor(9, 0);

delay(1000);
}
ISR(TIMER1_COMPA_vect)          // timer compare interrupt service routine
{
  printflag = 1;
 }
```

```
void loop(void)
{
 // S1data = digitalRead(S1);
//if(S1data==LOW){
 // Serial.println("S1 0n");
//}else{
  //Serial.println("S1 off");
//}
radio.startListening();
if ( radio.available() )
{
radio.read(data, sizeof(data));
//Serial.println(data[0]);
//Serial.println(data[1]);
if(printflag == 1){
     visspeed();
     printflag = 0;
}
vistrip();
//visspeed();
}
delay(5);
 radio.stopListening();
S1data = digitalRead(S1);
  radio.write(&S1data, sizeof(S1data));

}

void visspeed(){
//delay(50);
lcd.setCursor(9, 0);
lcd.print(data[0]);
lcd.print("   ");
delay(5);
return;
}

void vistrip(){
lcd.setCursor(9, 1);
lcd.print(data[1]);
lcd.print("   ");
delay(5);
return;
}
```