Lokalisatie met Arduino‘s op basis van vier bakens

Peter van Dijk & Elizabeth Schermerhorn

8 november 2014

Inhoudsopgave

1 Inleiding

Dit is het verslag voor de opdracht 'Lokalisatie' van het vak Networked Smart Systems. Voor deze opdracht moest er een lokalisatiealgoritme worden ontwikkeld waarmee binnen een vak van 5 bij 5 meter de positie zo nauwkeurig mogelijk proberen te bepalen. Hierbij wordt gebruik gemaakt van ultrasone signalen vanuit vaste referentiepunten op de hoeken van dit vak. De locatie van een meetinstrument kan van belang zijn bij het verrichtingen van metingen, bijvoorbeeld bij een branddetector in een bos.

In dit paper zal het door ons geïmplementeerde lokalisatiealgoritme besproken worden. Er zal eerst een afweging worden gemaakt tussen bestaande algoritmen. Daarna zal de gebruikte hardware en software voor de lokalisatie besproken worden. Vervolgens wordt er uit deze algoritmen het algoritme gekozen dat het beste aansluit op deze hardware en software en de implementatie hiervan uitgelegd worden. Daaropvolgend zullen de testopstelling en de resultaten worden uitgelicht. Ten slotte zal hieruit een conclusie worden getrokken en zullen er aanbevelingen voor vervolgonderzoek gedaan worden.

2 Probleemstelling

Aan de hand van vier bakens die in een 5 bij 5 meter veld geplaatst worden, moet de positie van de ontvanger bepaald worden. De onderzoeksvraag die hier centraal staat: *"Is het mogelijk om met een afwijking kleiner dan 5 procent de positie van de ontvanger in een 5 bij 5 meter veld te bepalen?"* De verwachte uitkomst van dit onderzoek is dat het eindproduct in staat is om met een nauwkeurigheid van 25 centimeter (5 procent) de positie te bepalen binnen het 5 bij 5 meter veld.

Een limitatie bij deze positiebepaling is onder andere de gebruikte hardware. Er wordt gebruik gemaakt van een Arduino Uno, een computer met beperkte rekenkracht en beperkt werkgeheugen. Verwachte obstakels hierdoor zijn bit overflows en het beperkte aantal variabelen in de code (in verband met het beperkte werkgeheugen). De overflow wordt als probleem verwacht omdat de Arduino geen grote getallen kan representeren en er waarschijnlijk met grote getallen gerekend moet gaan worden. Een voorbeeld hiervan is bijvoorbeeld de halvering van het bereik van getallen. Zo is een `long` geen 64, maar 32 bits.[1] Een ander verwacht obstakel is de snelheid van het geluidsignaal dat de bakens zullen gaan uitzenden, omdat de geluidssnelheid varieert naarmate de temperatuur en luchtvochtigheid veranderen. Dit zou gevolgen kunnen hebben voor de te bepalen positie.

3 Gerelateerd werk

Op het gebied van lokalisatie-algoritmen is er al veel onderzoek verricht. Hierdoor is er veel werk wat gebruikt kan worden bij het onderzoek. Er zullen twee papers worden bekeken waarin globale positie systemen (GPS) beschreven worden. In de twee papers worden er trilateratie-algoritmen gepresenteerd waarmee de exacte locatie bepaald kan worden op twee verschillende manieren. In *A New Location Technique for the Active Office* worden twee manieren beschreven [4], namelijk:

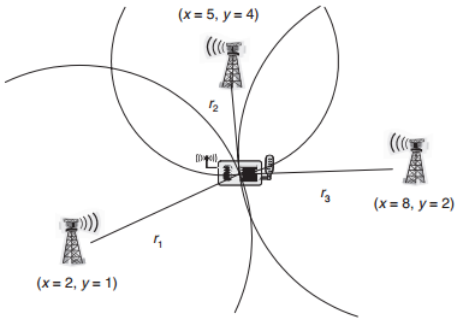
- Active badges
- ParcTab

Active Badges is gebaseerd op het periodiek verzenden van informatie naar iedereen. Door bijvoorbeeld een gebouw heen zijn er overal sensoren geplaatst die luisteren naar berichten die worden verstuurd door badges. Door vast te stellen welke sensoren het signaal van welke badge hebben ontvangen is het mogelijk een grove schatting te geven van de locatie van de badge.

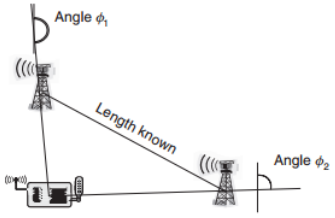
ParcTab is een zakcomputer die gebruik maakt van een netwerk dat communiceert via infraroodsignalen. Wat betreft de lokalisatie lijkt deze heel veel op de badges die hiervoor zijn beschreven. Er worden berichten verstuurd en afhankelijk van welke sensor deze ontvangen wordt de locatie bepaald. Een groot verschil is dat de ParcTab is bedoeld voor intensiever gebruik dan alleen het berekenen van de locatie.

In het algoritme in het hoofdstuk *Localization and positioning* van *Protocols and Architectures for Wireless Sensor Networks* wordt er gebruik gemaakt van bakens die een signaal uitzenden [3]. Met behulp van deze bakens kan op twee manieren een positie worden berekend: met *triangulatie* (Figuur 2) en *trilateratie* (Figuur 1). Triangulatie maakt gebruik van de invalshoek om de positie te berekenen. Hierbij moeten de hoeken tot tenminste twee bakens en de afstand tussen de bakens bekend zijn. Trilateratie gebruikt de afstand tot de bakens om de positie te bepalen. Voor dit algoritme is het van belang dat er minimaal tot drie bakens de afstand bekend is. Wanneer de afstand tot drie bakens bekend is kan met een simpele matrixvermenigvuldiging de positie berekend worden. Dit is het algoritme dat ook is gebruikt in de in dit paper besproken oplossing.

De opstelling die hier gebruikt wordt zal als basisprincipe worden gebruikt voor de implementatie beschreven in deze paper. Aangezien in dit experiment ook gebruik wordt gemaakt van bakens die een signaal versturen en een ontvanger die dit ontvangt en op basis van die gegevens de positie bepaald.



Figuur 1: Trilateratie [3]



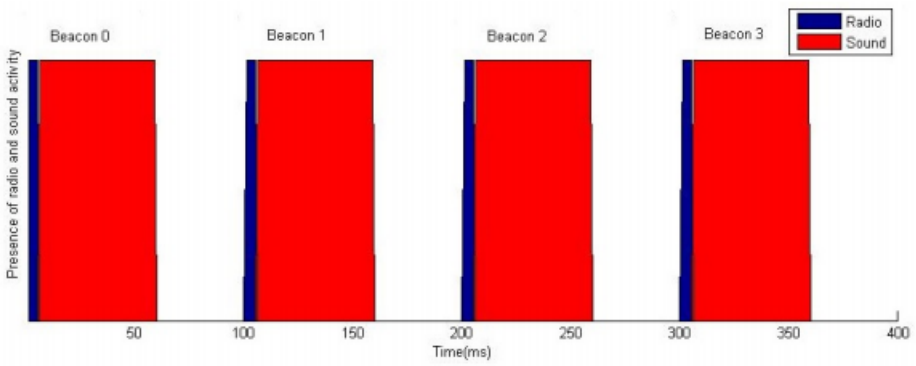
Figuur 2: Triangulatie [3]

4 Het algoritme

Het algoritme dat geïmplementeerd is, bestaat uit twee verschillende onderdelen. Het eerste deel bestaat uit het berekenen van de afstand tot de bakens en het tweede deel uit de berekening van de positie. Deze twee onderdelen zullen apart besproken worden voor de overzichtelijkheid.

4.1 Afstand bepalen tot aan de bakens

Zoals te zien is in Figuur 3 wordt er eerst een radiosignaal verstuurd, gevolgd door een ultrasoon signaal. De bakens gebruiken een master-slave opstelling, waarbij de master aangeeft wanneer elke slave mag sturen. Als eerste wordt er door de master-baken een radio uitzending uitgevoerd met als bericht het ID van de baken die vervolgens mag zenden. Omdat dit een radiosignaal is gaat dit met de snelheid van het licht, hierdoor is het tijdsverschil tussen de verschillende nodes die het op verschillende momenten ontvangen verwaarloosbaar. Op het moment dat het radiosignaal ontvangen wordt, begint de desbetreffende baken direct met het zenden van een ultrasoon geluid. Er is bekend dat dit meteen gebeurt, dus door bij te houden hoeveel tijd verstrijkt tussen het ontvangen van het radiosignaal en het ontvangen van het ultrasoon geluid kan er een schatting



Figuur 3: Het signaal van de bakens

gemaakt worden van de afstand tussen baken en ontvanger door de geluidssnelheid te gebruiken. Deze berekening ziet er als volgt uit:

$$d_b = (T_u - T_r) * g$$

Met:

d_b: afstand tot baken
T_u: tijd ontvangen ultrasoon geluid
T_r: tijd ontvangen radio
g: geluidssnelheid

Het is van belang om een juiste geluidssnelheid te kiezen bij de omgeving waarin het systeem staat, de afwijkingen kunnen groot zijn. Dit proces van luisteren en berekenen herhaalt zich oneindig lang zodat wanneer de node in beweging is er nog steeds een correcte locatie berekend kan worden.

4.2 Trilateratie

Wanneer van alle vier de bakens bekend is wat hun afstand is tot de node, kan door middel van matrixberekeningen de positie worden bepaald. Hierbij is er gebruik gemaakt van het hoofdstuk *Localization and positioning* van *Protocols and Architectures for Wireless Sensor Networks*[3]. Alle afstanden en berekende uitkomsten worden weergegeven in centimeters om de nauwkeurigheid te kunnen bepalen.
De volgende berekening is gebruikt bij het berekenen van de locatie:

$$2 * \begin{bmatrix} x_3 - x_1 & y_3 - y_1 \\ x_3 - x_2 & y_3 - y_2 \end{bmatrix} \begin{bmatrix} x_u \\ y_u \end{bmatrix} = \begin{bmatrix} (r_1^2 - r_3^2) - (x_1^2 - x_3^2) - (y_1^2 - y_3^2) \\ (r_2^2 - r_3^2) - (x_2^2 - x_3^2) - (y_2^2 - y_3^2) \end{bmatrix}$$

Met:

x_u: de te berekenen x-coördinaat van de Arduino.
y_u: de te berekenen y-coördinaat van de Arduino.
x_i: het x-coördinaat van baken i.
y_i: het y-coördinaat van baken i.
r_i: de afstand van de Arduino tot aan baken i.

4.3 Implementatie

De matrixvermenigvuldiging voor de trilateratie is om te schrijven naar de volgende formules, waarmee de x- en y-coördinaten van de ontvanger te berekenen zijn:

$$c_1 = (((r_1 * r_1) - (r_3 * r_3)) - ((x_1 * x_1) - (x_3 * x_3)) - ((y_1 * y_1) - (y_3 * y_3)))/2$$

$$c_2 = (((r_2 * r_2) - (r_3 * r_3)) - ((x_2 * x_2) - (x_3 * x_3)) - ((y_2 * y_2) - (y_3 * y_3)))/2$$

$$n = (-((y_3 - y_1) * 10000)/(x_3 - x_1))$$

$$m = ((c_1 * 10000)/(x_3 - x_1))$$

$$y_u = ((c_2 * 10000) - m * (x_3 - x_2))/(n * (x_3 - x_2) + (10000 * (y_3 - y_2)))$$

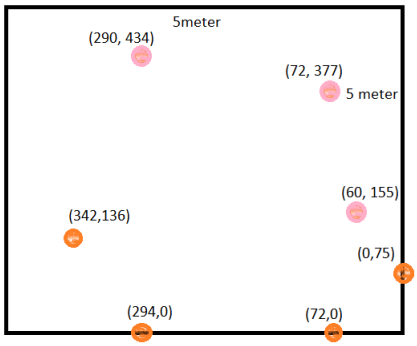
$$x_u = ((n * y) + m)/10000$$

Wat opvalt aan deze berekening is dat er een aantal maal 10.000 in staat. Deze factor is geïntroduceerd om te voorkomen dat afkapping van getallen zorgt voor foute resultaten. Als deze factor niet gebruikt zou worden dan zouden tussenberekeningen een getal kleiner dan 1 kunnen opleveren, wat afgekapt wordt tot 0. Hiermee doorrekenen levert incorrecte resultaten op. De factor wordt aan het einde weer weggedeeld, om het gewenste resultaat te verkrijgen. Een alternatief voor deze oplossing is het gebruik van decimale getallen, maar het rekenen hiermee is erg langzaam, daarom is daar niet voor gekozen.

Doordat de afstand tot aan de bakens elke 400 milliseconden opnieuw wordt berekend geldt dit ook voor de x- en y-coördinaten. Voor elke vier nieuwe afstandsmetingen worden de x- en y-coördinaten opnieuw berekend. Het algoritme maakt gebruik van alle vier de bakens en niet slechts drie voor de positiebepaling. Dit wordt gedaan als volgt. Voor elke combinatie van drie bakens wordt met behulp van de afstanden tot deze bakens met behulp van het hiervoor genoemde algoritme een set van coördinaten berekend. Vervolgens wordt het gemiddelde genomen van de berekende sets coördinaten. Dit zorgt er voor dat er voor kleine afwijkingen in afstandsberekening tot bakens gecompenseerd kan worden. In Appendix A staat de code die de berekeningen implementeert.

5 Testopstelling en metingen

Om de implementatie te testen is er gebruik gemaakt van een testopstelling zoals die in Figuur 4 staat weergegeven. De gebruikte hardware bestaat uit een Arduino UNO in combinatie met een ultrasoonontvanger. Daarnaast zal de software kort toegelicht worden en de daarbij horende constanten die zijn gebruikt.



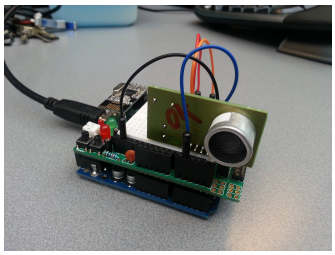
Figuur 4: Meetopstelling

5.1 De hardware

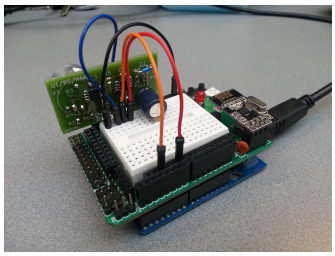
De hardware bestaat uit twee delen, er is een ontvanger die de geluidsgolven opvangt. Deze ontvanger bevindt zich op de Arduino. Er zijn tevens vier zenders, deze staan voor de vier bakens die worden gebruikt om het ultrasoongeluid uit te zenden.

5.1.1 Ontvanger

De Hardware die gebruikt is voor het ontvangen van de signalen is een Arduino UNO en een Nordic nRF24L01 draadloze transceiver(kan ontvangen en zenden) in combinatie met een ultrasoonontvanger. In Figuur 5 en 6 staat de gebruikte ontvanger vanuit twee posities bekeken.



Figuur 5: De ontvanger (voorkant)



Figuur 6: De ontvanger (achterkant)

5.1.2 Zender

Als zender is er gebruik gemaakt van vier bakens die een ultrasoon geluid uitzenden. Het signaal van deze bakens wordt verstuurd in de richting waarin ze wijzen. Deze bakens bevatten hardware die vergelijkbaar is met die van de ontvanger, met als grote verschil dat er in plaats van een ultrasoonontvanger een ultrasoonzender is gebruikt.

5.2 De software

Om de RF24 radio aan te sturen wordt gebruik gemaakt van de opensource-softwarebibliotheek voor de Arduino [2]. Met deze software kan de radio aangestuurd worden en kunnen pakketjes worden verzonden en ontvangen. De radio luistert en verstuurt pakketten over een bepaald frequentiekanaal. Het is van belang dat de radio's hetzelfde frequentiekanaal gebruiken, zodat ze met elkaar kunnen communiceren. De kanalen hebben een nummer van 0 tot en met 125. De gebruikte instellingen zijn als volgt:

- Frequentiekanaal: 76
- Automatisch herzenden: uit
- Transmissiesnelheid: 2 Mbps
- Adres pipe: 0xdeadbeefa1
- Payload-grootte: 1 byte

De ultrasoonontvanger gebruikt de volgende instellingen:

- 40 kHz signaal: uit
- Versterking: 50x

Om de tests te kunnen uitvoeren is er gebruik gemaakt van vier bakens die met een bepaald patroon informatie verzenden. In Figuur 3 is schematisch weergegeven hoe de signaalverzending werkt. In het schema zijn vier dezelfde

iteraties te zien. Omdat er gebruik wordt gemaakt van vier bakens staan er vier berichten in het schema weergegeven.

De ultrasoonontvangers waarmee de signalen van de bakens worden ontvangen werden op een afstand groter dan vijf meter zeer onnauwkeurig dus is ervoor gekozen om binnen een veld van vijf bij vijf meter te werken. In Figuur 4 staan de vier bakens aangegeven met de oranje stippen samen met hun coördinaten. Met de roze stippen worden de testlocaties bedoeld waarop is getest. De bakens die zijn gebruikt versturen alleen signalen in de richting waarin ze wijzen. Hierdoor is het niet mogelijk correcte gegevens te verkrijgen wanneer er vanachter de bakens wordt gemeten.

Tijdens het testen wordt de ontvanger op een bepaald coördinaat geplaatst. Het is belangrijk dat deze locatie precies is, anders ontstaat er een afwijking tussen de meetresultaten en de verwachte waarde. Het is van belang bij het testen dat er gebruik wordt gemaakt van een "schoon"grid. Dit houdt in, geen obstakels die de metingen zouden kunnen verstoren. Alle tests zijn "schoon"bepaald. Op de volgende locaties zijn de volgende tests uitgevoerd:

Test 1: De Arduino wordt op het coördinaat (290, 434) geplaatst.
Test 2: De Arduino wordt op het coördinaat (60, 155) geplaatst.
Test 3: De Arduino wordt op het coördinaat (72, 377) geplaatst.

5.3 Meetresultaten

In Appendix B staan alle meetresultaten van de drie tests verwerkt. In de tabellen staan 25 waarden die zijn gemeten op het aangegeven coördinaat. De gemiddelden hiervan zijn eronder aangegeven. De afwijkingen van deze gemiddelden zijn te vinden in Tabel 1, 2 en 3.

Tabel 1: Metingswaarden op coördinaat (290, 434) in cm

gemiddelde waarde x-coördinaat	gemiddelde waarde y-coördinaat
292,9	439,1
coördinaat	afwijking van coördinaat in cm
x	2,9
y	5,1
(x,y)	5,87

5.4 Analyse

Te zien is dat de metingen niet hetzelfde zijn, ze fluctueren binnen een interval. Een paar redenen hiervoor zijn:

Tabel 2: Metingswaarden op coördinaat (60, 155) in cm

gemiddelde waarde x-coördinaat	gemiddelde waarde y-coördinaat
60,4	151,8
coördinaat	afwijking van coördinaat in cm
x	0,4
y	3,2
(x,y)	3,22

Tabel 3: Metingswaarden op coördinaat (72,377) in cm

gemiddelde waarde x-coördinaat	gemiddelde waarde y-coördinaat
75,6	379,5
coördinaat	afwijking van coördinaat in cm
x	3,6
y	2,5
(x,y)	4,38

- Voor het algoritme is de geluidssnelheid erg belangrijk, omdat deze één keer gekozen is en niet elke keer wordt aangepast aan de temperatuur en luchtvochtigheid wijkt de berekende positie af van de echte positie.
- De ultrasoonontvanger ontvangt echo's die weerkaatsen op objecten. Deze foutieve metingen worden meegenomen in de berekening waardoor de waarde niet meer klopt. Er bleken echo's te zijn doordat er in het algoritme de afstand tot de verschillende bakens eerst wordt getoond voordat er een positie wordt berekend. Hieruit bleek dat de bakens soms elf meter van de ontvanger af stonden en dit kan alleen verklaard worden doordat er echo's zijn.

6 Conclusie

De onderzoeksvraag die aan het begin gesteld werd: *"Een algoritme opstellen waarmee de positie bepaald kan worden met een afwijking van minder dan tien centimeter."* Met dit experiment is het gelukt een lokalisatiealgoritme te ontwerpen en te implementeren met een afwijking kleiner dan tien centimeter. Uit de onderzoeksresultaten die weergegeven staan in Appendix B kan dit geconcludeerd worden. In Appendix B staat in de onderste tabel van elke grafiek de gemiddelde afwijking in centimeters. Voor bovenstaande hypothese geldt wel dat de ontvanger zich vóór de bakens moet bevinden en binnen een veld van vijf bij vijf meter. Anders kan er niet gegarandeerd worden dat de afwijking kleiner dan tien centimeter is.

6.1 Aanbevelingen

Voor een vervolgonderzoek zou er naar de volgend punten gekeken kunnen worden.

- De arduino die tijdens dit onderzoek gebruikt is heeft veel last van bit overflow. Om de nauwkeurigheid te vergroten zou er onderzoek gedaan kunnen worden naar het gebruik van Arduino's met een grotere rekencapaciteit. Zodat op deze manier ook berekeningen gedaan kunnen worden op een afstand groter dan zes meter.

- De bakens en ontvangers die tijdens dit onderzoek gebruikt zijn zijn maar nauwkeurig tot ongeveer 5 meter. Voor een vervolg onderzoek kan er gekeken worden naar de haalbaarheid met sterkere bakens en ontvangers zodat de positiebepaling nauwkeurig blijft wanneer de afstand groter dan 5 meter bedraagt.

- Het ontworpen algoritme werkt alleen als er geen verstoringen in het veld zijn, dit wil zeggen dat er geen obstakels staan tussen het uitgezonden geluid door de bakens en de ultrasoonontvanger. Een vervolgonderzoek kan zich richten op het nauwkeurig berekenen van de positie met obstakels in het veld.

Referenties

[1] Arduino Reference: long. `http://arduino.cc/en/Reference/long`, Nov. 2014.

[2] RF24 Class Reference. `http://maniacbug.github.io/RF24/classRF24.html`, May 2014.

[3] H. Karl and A. Willig. *Protocols and Architectures for Wireless Sensor Networks*. John Wiley & Sons, 2005.

[4] A. Ward, A. Jones, and A. Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, 1997.

A Code Trilateration

```
#include <SPI.h>
#include "nRF24L01.h"
#include "RF24.h"
#include "printf.h"

RF24 radio(3, 9);
const int audioPin = 1;
const uint64_t pipes[1] = { 0xdeadbeefa1LL };

int32_t x[4] = {100,200,300,400};
int32_t y[4] = {20,0,0,20};

uint8_t radio_received = 0;    //Received index of beacon
unsigned long radio_stopped;   //t0
unsigned long audio_started;   //t1
uint32_t distance;             //(t1-t0)*speed_of_sound
unsigned long speed_of_sound = 34421;//344.21 m/s = 34421 / 1000000 cm/us
int offset = 7;                //7 cm

uint32_t distances[4];         //calculated distances
int32_t x_crds[8];          //calculated coords
int32_t y_crds[8];          //calculated coords

int counter = 0;
int count   = 0;
boolean bereken = false;

void setup(void)
{
  Serial.begin(57600);
  printf_begin();
  radio.begin();
  radio.setChannel(76);
  radio.setRetries(0,0);
  radio.setPayloadSize(1);
  radio.setDataRate(RF24_2MBPS);
  radio.openReadingPipe(1,pipes[0]);
  radio.startListening();
  radio.setAutoAck(false);
  radio.printDetails();
}

void loop(void)
{
  if ( radio.available() )
  {
    bool done = false;
    while (!done)
    {
      done = radio.read( &radio_received, sizeof(uint8_t) );
    }
    bool received = false;
    bool waiting = true;
    radio_stopped = micros();
    while(waiting){
      if(analogRead(audioPin)>0){//radio signal received
        audio_started = micros();
        received = true;
        waiting = false;
      }
      else if(micros()-radio_stopped > 75000){//nothing received after radio
        waiting = false;
      }
    }

    if(received){
```

```
      distance = (audio_started-radio_stopped)*speed_of_sound/1000000;
      distance = distance -offset;
      if(distance< 700){//No false readings
        if(distance >= 0) {
          distances[radio_received] = distance;
        }
        else {
          distances[radio_received] = 0;
        }
      }
    }
    if(count == 3){
     for(int i=0;i<4;i++){
       printf("%i : %ld cm ", i, distances[i]);
     }
     printf("\n\r");
     counter = 0;
     while(counter < 4){
       calculateXY((counter%4) , ((counter+1)%4), ((counter+2)%4));
       counter++;
     }
     if(bereken){
       printGemiddelde();
     }
     bereken = !bereken;
    }
    count = (count +1) % 4;
  }

}

void calculateXY(int i, int j, int k){
  long r1 = distances[i];
  long r2 = distances[j];
  long r3 = distances[k];
  long x1 = x[i];
  long x2 = x[j];
  long x3 = x[k];
  long y1 = y[i];
  long y2 = y[j];
  long y3 = y[k];

  long c1 = (((r1*r1)-(r3*r3))-((x1*x1)-(x3*x3))-((y1*y1)-(y3*y3)))/2;
  long c2 = (((r2*r2)-(r3*r3))-((x2*x2)-(x3*x3))-((y2*y2)-(y3*y3)))/2;

  long n =(-((y3-y1)*10000)/(x3-x1));
  long m = ((c1*10000)/(x3-x1));

  long y = ((c2*10000) - m*(x3-x2))/(n*(x3-x2) + (10000*(y3-y2)));
  long x = ((n*y)+m)/10000;

  if(!bereken){
    x_crds[counter] = x;
    y_crds[counter] = y;
  }else{
    x_crds[counter+4] = x;
    y_crds[counter+4] = y;
  }
}

void printGemiddelde(){
   int x_gemiddelde =  (x_crds[0]+x_crds[1]+x_crds[2]+x_crds[3]+x_crds[4]+x_crds[5]+x_crds[6]+x_crds[7])/8;
   int y_gemiddelde =  (y_crds[0]+y_crds[1]+y_crds[2]+y_crds[3]+y_crds[4]+y_crds[5]+y_crds[6]+y_crds[7])/8;
   printf("De gemiddelde x:%i  en y: %i waarden \n\r", x_gemiddelde, y_gemiddelde);
}
```

B Meetresultaten

(290, 434)	
x-coördinaat	y-coördinaat
294	434
296	445
296	441
293	434
295	436
293	434
297	439
294	432
293	449
292	449
292	449
293	449
292	439
294	439
286	433
286	433
292	436
291	435
289	429
295	436
293	444
293	435
294	451
293	433
293	433
288	440
gemiddelde	
292,9	439,1
afwijking(290, 434) in cm	
x	2,9
y	5,1
(x,y)	5,87

(60, 155)	
x-coördinaat	y-coördinaat
59	137
57	147
58	154
58	151
60	157
60	148
60	154
62	147
63	151
62	147
61	137
60	154
60	150
60	152
59	152
60	154
61	151
60	152
62	155
61	152
61	155
61	149
61	155
61	154
60	152
63	156
gemiddelde	
60,4	151,8
afwijking (60, 155) in cm	
x	0,4
y	3,2
(x,y)	3,22

(72, 377)	
x-coördinaat	y-coördinaat
75	378
77	383
76	382
76	381
75	378
74	374
74	377
75	378
76	382
77	387
76	388
78	382
74	387
76	378
74	382
76	381
77	377
75	382
77	377
75	380
76	374
73	382
75	377
74	376
77	378
78	380
gemiddelde	
75,6	379,5
afwijking(72,377) in cm	
x	3,6
y	2,5
(x,y)	4,38