Localisatie met Arduino‘s op basis van vier bakens

Peter van Dijk & Elizabeth Schermerhorn

13 juli 2014

Inhoudsopgave

1 Inleiding

In deze paper zal een localisatiealgoritme besproken worden. De localisatie zal gedaan worden op basis van vier bakens die een ultrasoon geluid uitzenden en een arduino met een ultrasoon ontvanger. Het doel van dit experiment is om binnen een 5m bij 5m veld de positie zo nauwkeurig mogelijk proberen te bepalen. Aan de hand van het ultrasone geluid in combinatie met een positiebepalings algoritme wordt nauwkeurig bepaald wat de coördinaten zijn.

2 Probleemstelling

De onderzoeksvraag die hier centraal staat: *"Een algoritme opstellen waarmee met trilateratie de positie bepaald kan worden met 5 % nauwkeurigheid."* De hypothese is dat dit een uitdaging is, er zijn veel factoren die van invloed zijn op de berekening van de trilateratie. Twee belangrijke factoren zullen de geluidssnelheid worden en de overflow(Getallen zijn te groot om te representeren) in arduino zijn.

3 Gerelateerd werk

Op dit gebied is er al veel onderzoek verricht. Hierdoor is er veel werk wat gebruikt kan worden bij het onderzoek. Aangezien dit een onderzoek is wat al vele malen is uitgevoerd is er veel gerelateerd werk wat is gebruikt in dit onderzoek. In deze twee papers worden er globale localisatie systemen beschreven en hoe deze gebruik maken van externe factoren. In de volgende twee papers worden er trilateratiealgoritmen gepresenteerd waarmee de exacte locatie bepaald kan worden. In de eerste twee papers worden twee manieren toegelicht waarop localisering mogelijk is. In [http://www.cl.cam.ac.uk/research/dtg/www/publications/public/files/tr.97.10.pdf] worden twee manieren beschreven, namelijk:

- Active badges
- ParcTab

Active Badges is gebasseerd 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 g even van de locatie van de badge.

ParcTab is een PDA die gebruik maakt van infrarood voor het netwerk.Wat betreft de localisatie bepaling lijkt deze heel veel op de badges die hierboven staan 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 veel meer en intensiever gebruik dan alleen de localatie berekenen.

In de laatste twee papers, [http://en.wikipedia.org/wiki/Trilateration] en [Localization and Positioning, CH. 9] worden er twee manieren besproken waarop de GPS positie bepaald kan worden.

In hoofdstuk 9 van Localization and Positioning wordt er ook gebruik gemaakt van bakens die een signaal uitzenden. Er moet eerst berekend worden wat de afstand is tot de verschillende bakens. Voor dit algoritme is het van belang dat er minimaal tot drie bakens de locatie bekend is. Wanneer de afstand tot drie bakens bekend is wordt er gebruik gemaakt van matrixberekeningen om de locatie te bepalen. Dit is de implementatie die ook is gebruikt in het experiment wat in deze paper is gemaakt.
Een tweede positiebepaling die gebruikt zou kunnen worden heet triangulation. Ook hier wordt gebruik gemaakt van de afstand tot de bakens die een signaal verzenden. Dit algoritme lijkt veel op het algoritme uit CH9 maar hier wordt gebruik gemaakt van de berekening van cirkels waardoor er geen matrixberekeningen nodig zijn.

4 Instellingen

Om deze hypothesen te toetsen moeten er metingen gedaan worden. Hier zijn verschillende zaken van belang.

- De gebruikte hardware
- De gebruikte software
- De instellingen/vastgestelde constanten
- De onderzoeksopstelling

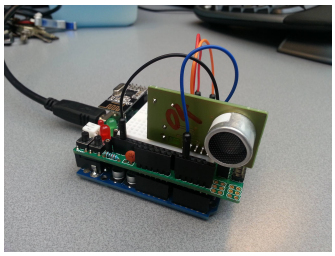
4.1 De hardware

4.1.1 Ontvanger

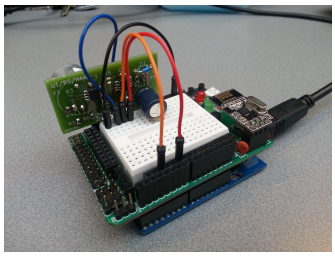
De Hardware die gebruikt is bij het ontvangen van de signalen is een Arduino UNO en een Nordic nRF24L01 draadloze transceiver in combinatie met een ultrasoonontvanger. In Fig. 1 en 2 staat de gebruikte arduino vanuit twee posities bekeken.

4.1.2 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. In Fig. 3 staat een schematische weergave van hoe een baken eruitziet. Er is duidelijk te zien dat het ultrasone geluid één kant uitgezonden wordt.



Figuur 1: De gebruikte arduino(voorkant)



Figuur 2: De gebruikte arduino(achterkant)

4.2 De software

Om de hardware te gebruiken wordt gebruik gemaakt van de opensourcesoftwarebibliotheek voor de Arduino, die te vinden is op:

`http://maniacbug.github.io/RF24/`

Met deze software kunnen de radio en ontvanger aangestuurd worden en kunnen pakketjes worden verzonden en ontvangen. De radio luistert en verstuurt pakketten over een bepaald kanaal. Het is van belang dat de radio's hetzelfde kanaal gebruiken, zodat ze met elkaar kunnen communiceren. De kanalen hebben een nummer van 0 tot en met 125. De radio heeft de volgende instellingen nodig:

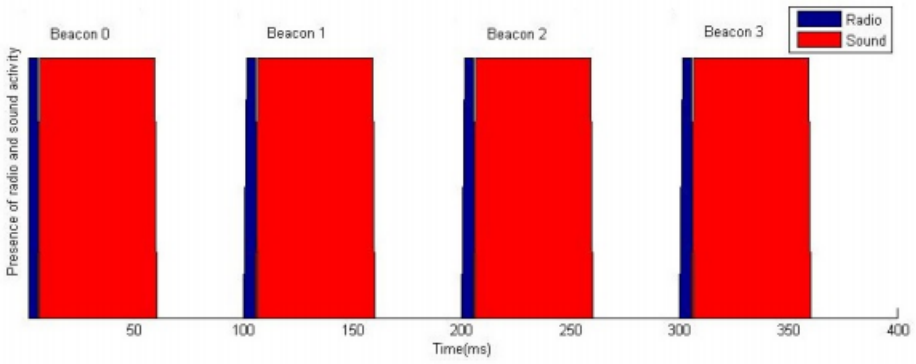
- Kanaal 76
- Automatisch herzenden Uit
- Transmissiesnelheid: 2 Mbps
- Adres verzendende pipe: 0xdeadbeefa1LL
- Payload-grootte 1 byte

De ultrasoon ontvanger moet op een bepaalde manier aangesloten worden zodat deze geluid kan ontvangen:

Figuur 3: schematische baken

afkorting	Wijze van aansluiten
E	Verbinden met GND van Arduino
GN	Verbinden met 5V van Arduino
++	Verbinden met 5V van Arduino
GND	Verbinden met Ground van Arduino
S	niet aansluiten

Om de tests te kunnen uitvoeren is er gebruik gemaakt van vier bakens die met een bepaald patroon informatie verzenden. In fig. 4 is schematisch weergegeven hoe de signaal verzending 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.



Figuur 4: Het signaal van de bakens

5 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 uiteindelijke positiebepaling. Deze twee onderdelen zullen apart besproken worden voor de overzichtelijkheid.

5.1 Afstand bepalen tot aan de bakens

Zoals te zien is in Fig.4 wordt er eerst een radio signaal verstuurd gevolgd door een ultrasoon signaal. Het versturen van een ultrasoon bericht wordt door een master-baken bepaald. Als eerste wordt er door de master-baken een radio bericht uitgezonden naar iedereen met het bericht welke baken mag sturen. Omdat dit een radiosignaal is gaat dit met de snelheid van het licht, hierdoor is het verschil tussen de verschillende nodes die het op verschillende momenten ontvangen verwaarloosbaar. Op het moment dat het radio signaal ontvangen wordt, begint de desbetreffende baken direct met een ultrasoon geluid versturen. Doordat bekent is dat dit meteen gebeurt wordt er bijgehouden hoelang het duurt tussen het ontvangen van het radiosignaal en het ontvangen van ultrasoon geluid. Op het moment dat het bericht wordt ontvangen wordt er met behulp van de geluidssnelheid berekend wat de afstand tot het baken is. Met de formule:
$d_b = (T_u - T_r) * g$

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.

5.2 Trilateration

Nu 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 [iets]. De volgende berekening is gebruikt bij het berekenen van de locatie:

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

De argumenten in bovenstaande berekening betekenen:
x_u = de te berekenen x-coördinaat van de arduino.
y_u = de te berekenen y-coördinaat van de arduino.
x_x = het x-coördinaat van baken x.
y_x = het y-coördinaat van baken x.

r_x = de afstand van de arduino tot aan baken x.

Doordat de te berekenen x en y coördinaten aan de linkerkant genest in de formule staan was het nodig om deze om te schrijven. De volgende hulpberekeningen volgden daaruit:
$c_1 = (((r_1 * r_1) - (r_3 * r_3)) - ((x_1 * x_1) - (x_3 * x_3)) - ((y_1 * y_1) - (y_3 * y_3)))/2$

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

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

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

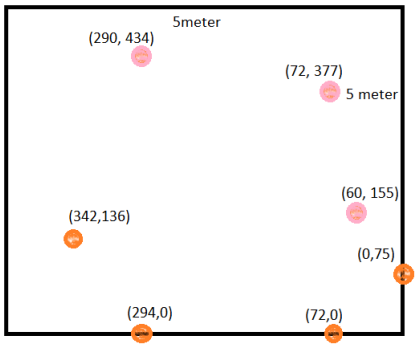
$y = ((c_2 * 10000) - m * (x_3 - x_2))/(n * (x_3 - x_2) + (10000 * (y_3 - y_2)))$

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

Er wordt eerst vermenigvuldigd met 10000 om later weer te delen door 10000. Dit is gedaan omdat arduino's niet goed rekenen met floats, de oplossing hiervoor was de getallen groot houden zodat er geen komma getallen ontstaan. Door de formules in deze volgorde uit te voeren worden de x en y uiteindelijk berekend. Doordat de afstand tot aan de bakens elke 400ms opnieuw wordt berekend geldt dit ook voor de x en y coördinaten. Elke keer als er een nieuwe afstand bekend is worden bovenstaande formules opnieuw berekend.

6 Testopstelling en metingen

Om de implementatie te testen is er gebruik gemaakt van een testopstelling zoals die in Fig. 5 staat weergegeven. De unisoonontvangers waarmee de signalen



Figuur 5: Meetopstelling

van de bakens worden ontvangen werden na vijf meter zeer onnauwkeurig dus is ervoor gekozen om binnen een veld van 5 bij 5 meter te werken. In het

schema 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 een arduino met radio en unisoon 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 zoals mensen die erdoorheen lopen. Alle tests zijn "schoonüitgevoerd. Op de volgende locaties zijn de volgende tests uitgevoerd.
Test1: De arduino wordt op het coördinaat (290, 434) geplaatst.
Test2: De arduino wordt op het coördinaat (60, 155) geplaatst.
Test3: De arduino wordt op het coördinaat (72, 377) geplaatst.
Verwacht wordt dat de metingen zeer nauwkeurig zullen zijn ¡ 5% afwijking. In Appendix Meetresultaten staan alle meetresultaten van de drie tests verwerkt. In de tabellen staat de laatste waarde voor het gemiddelde wat werd berekend door de arduino. de overige waarden zijn 25 gemeten waarden op die positie. Te zien is dat de metingen niet hetzelfde zijn, ze fluctueren binnen een interval. Een paar redenen hiervoor zijn:

- De node zit niet vast, iemand houdt de node vast en hier kan een paar centimeter verschil in zitten.
- De geluidssnelheid is erg belangrijk voor de berekening en het is mogelijk dat voor die omgeving op dat moment de verkeerde geluidssnelheid was ingevoerd.
- De ultrasoon ontvanger ontvangt ook echo's die weerkaatsen op objecten. Deze foutieve metingen worden meegenomen in de berekening waardoor de waarde niet meer klopt.
- Hoewel er op gelet is dat er niemand door het grid heen loopt tijdens de metingen is het moeilijk om alles tegelijk in de gaten te houden en zou het kunnen voorkomen dat iemand erdoorheen gelopen is.

7 Conclusie

De onderzoeksvraag die hier centraal staat: *"Een algoritme opstellen waarmee met trilateratie de positie bepaald kan worden met 5 % nauwkeurigheid."* Dit experiment is een succes, het is gelukt om een algoritme te ontwikkelen en implementeren waarmee de marge ¡=5% is. Hiermee is het experiment een succes.

7.1 Aanbevelingen

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

- de overflow die ontstaat met het rekenen met integers, op het moment dat de arduino verder staat dan 6 meter bij de bakens vandaan ontstaat er overflow.
- betere bakens en ontvangers, de signalen worden minder na 6 meter
- een algoritme ontwerpen wat beter tegen verstoringen in het veld kan.

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_coordinates[8];      //calculated coords
int32_t y_coordinates[8];      //calculated coords

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

void setup(void)
{
  Serial.begin(57600);
  printf_begin();
  //
  // Setup and configure rf radio
  //
  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 r1_r1 = r1*r1;
  long r2_r2 = r2*r2;
  long r3_r3 = r3*r3;
  long x1_x1 = x1*x1;
  long x2_x2 = x2*x2;
  long x3_x3 = x3*x3;
  long y1_y1 = y1*y1;
  long y2_y2 = y2*y2;
  long y3_y3 = y3*y3;

  //Beide c1 en c2 berekeningen gaan goed!
  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)); // 0.2551020
  long m = ((c1*10000)/(x3-x1));//32/6

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

```
  printf("[");
  Serial.print(x);
  printf(",");
  Serial.print(y);
  printf("]\n\r");

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

void printGemiddelde(){
   int x_gemiddelde =  (x_coordinates[0]+x_coordinates[1]+x_coordinates[2]+x_coordinates[3]+
   int y_gemiddelde =  (y_coordinates[0]+y_coordinates[1]+y_coordinates[2]+y_coordinates[3]+
   printf("De gemiddelde x:%i  en y: %i waarden \n\r", x_gemiddelde, y_gemiddelde);
}
```

B Meetresultaten

(290, 434)		(60, 155)		(72, 377)	
x-coordinaat	y-coordinaat	x-coordinaat	y-coordinaat	x-coordinaat	y-coordinaat
294	434	59	137	75	378
296	445	57	147	77	383
296	441	58	154	76	382
293	434	58	151	76	381
295	436	60	157	75	378
293	434	60	148	74	374
297	439	60	154	74	377
294	432	62	147	75	378
293	449	63	151	76	382
292	449	62	147	77	387
292	449	61	137	76	388
293	449	60	154	78	382
292	439	60	150	74	387
294	439	60	152	76	378
286	433	59	152	74	382
286	433	60	154	76	381
292	436	61	151	77	377
291	435	60	152	75	382
289	429	62	155	77	377
295	436	61	152	75	380
293	444	61	155	76	374
293	435	61	149	73	382
294	451	61	155	75	377
293	433	61	154	74	376
293	433	60	152	77	378
288	440	63	156	78	380
292,9	439,1	60,4	151,8	75,6	379,5

%- afwijking(290, 434)	
x	0,34
y	0,011

%- afwijking (60, 155)	
x	0,006
y	2,581

%- afwijking(72,377)	
x	5
y	0,531