

amsmath



Innovasjonsprosjekt Fase I

Prosjekt: BIRDEX360

Skrevet av: Espen, Lars, Luyang, Jesper, Runa, Fredrik, Lasse

Gruppe nr: 11

Dato: 21.11.18

Innhold

1 Problemstilling	1
2 Konsept	1
3 Design	2
3.1 Det fysiske sensortoget	2
3.2 Nettside	3
3.3 Trådløs kommunikasjon	3
4 Implementering	4
4.1 Kretsen	4
4.1.1 Komponenter i kretsen	5
4.2 Beregning av posisjon rundt vindmølle	5
4.3 Overordnet system	6
5 Verifikasjon og test	9
6 Konklusjon	10
A Vedlegg - Nettside	11
B Vedlegg - Arduino	11
C Vedlegg - Node.js kode	12
D Vedlegg - Nettsidekode	13

1 Problemstilling

Trønderenergi er avhengig av et system som har muligheten til å detektere fugler på et område, for å få en oversikt over fuglelivet på steder der det kan være aktuelt å sette opp vindmøller, eller hvor det allerede befinner seg vindmøller. Vi har basert på dette kommet frem til en forenklet problemstilling.

Hvordan kan man på en effektiv måte detektere fugler i området rundt en vindmølle?

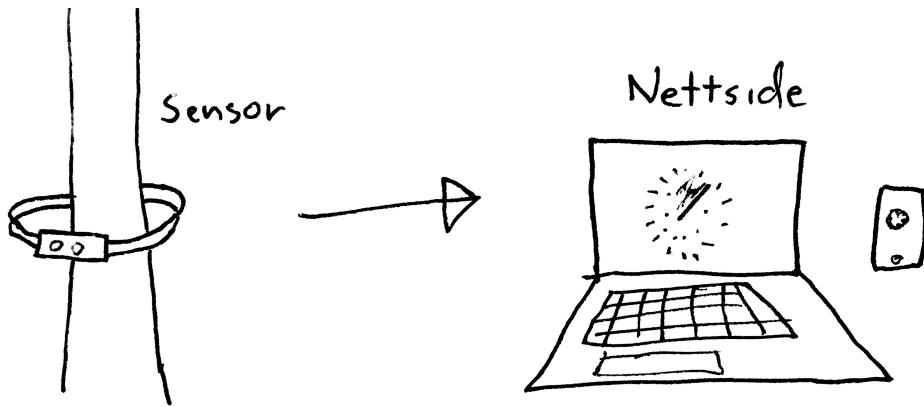
Dette er en problemstilling som ikke bare angår Trønderenergi spesielt, men de fleste vindkraftbedrifter som ønsker å ta hensyn til miljøet. FNs bærekraftsmål har som plan å stoppe klimaendringene innen 2030. Vindkraft har en sentral rolle i dette. Det vil derfor bli mer og mer aktuelt for bedrifter å investere i vindmøller. Trønderenergi ønsker å fokusere på vindkraft. Problemstillingen vil være med på å simplifisere prosessen Trønderenergi må gå gjennom for å bygge vindmøller. Problemstillingen er aktuell for å spare miljøet og fuglelivet for unødvendig skade forårsaket av at vindmøller er satt opp på steder der de vil ha en negativ innvirkning.

2 Konsept

I fase I av dette innovasjonsprosjektet har vi fokusert på å lage en prototype som enkelt og greit kan detektere fugler. Når en fugl kommer innenfor en viss rekkevidde av vindmøllen skal prototypen registrere fuglen. Data blir sendt fra prototypen til en nettside, som presenterer alt av data visuelt. (Se figur 1). Parameteren som blir målt er avstand. En høyttaler representerer et mulig virkemiddel for å skremme bort fugler.

I fase II av prosjektet skal vi ta i bruk maskinlæring, fotografering og lyd. Når fuglen blir avbildet skal produktet identifisere hvilken type fugl det er ved hjelp av maskinlæring. Etter maskinens vurdering, sendes det et varselsignal ut i fra en høyttaler plassert på vindmøllen. Dette signalet skal skremmefuglen til å tro at den er i fare, slik at den skifter retning. Dette skal gi trønderenergi muligheten til å kunne plasseres en vindmølle ved Froan, selv med et rikt fugleliv.

Dette konseptet er en effektiv løsning på problemstillingen vi har kommet frem til. Produktet gir brukeren konkret og oversiktlig informasjon om fuglelivet i et område der det skulle være hensiktsmessig å foreta slike målinger. På denne måten får brukeren et godt grunnlag til å kunne avgjøre om stedet er egnet for vindmøller eller ikke.

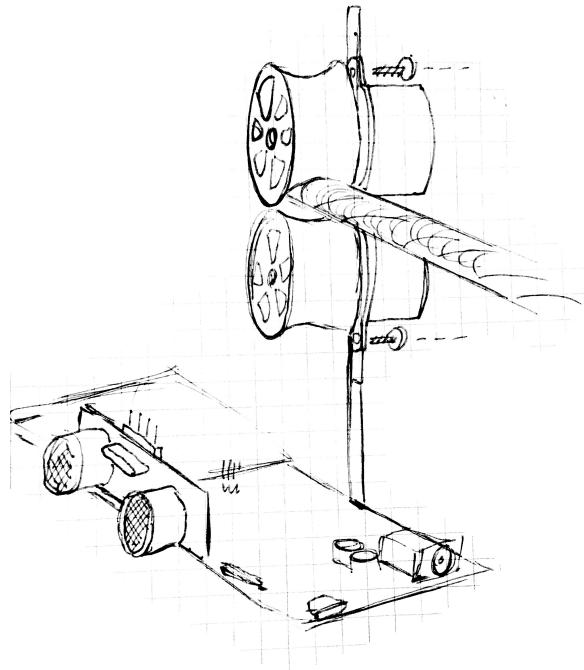


Figur 1: Sensor som gir realtime-data til en nettside.

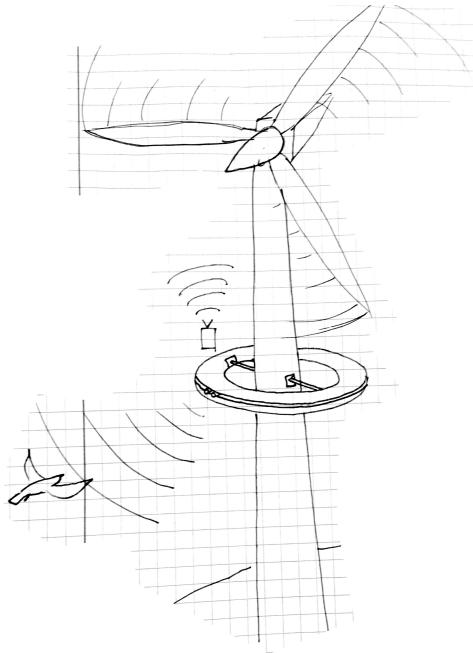
3 Design

3.1 Det fysiske sensortoget

Alt av sensorer, batteri og mikrokontrollerer brukt av sensoren vil være festet til en vogn. Denne sensorvognen skal da kjøre rundt på en skinne som går rundt vindmøllen, vist i figur 2. Tanken her er at det ferdige produktet utseendemessig skal stått i stil med de fleste vindmøller, altså et hvitt og simplistisk design. Dette er noe som kommer frem i figur 3. Sensoren som er brukt i denne prototypen er en avstandssensor av typen HC-SR04 Ultrasonic Distance.



Figur 2: Sensorvogn hektes på skinnen som går rundt vindmøllen.



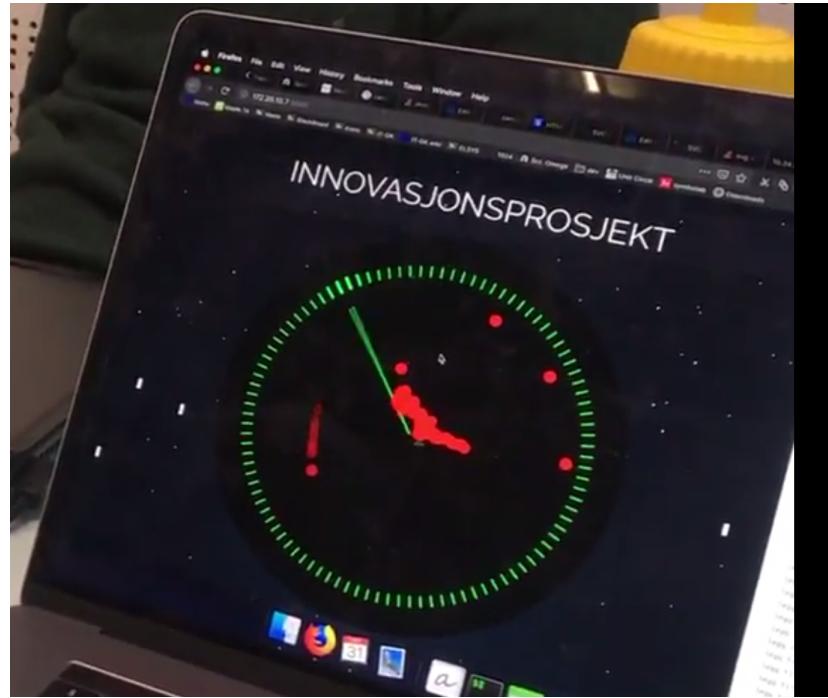
Figur 3: Skinnen går rundt vindmøllen for å gi oss 360 graders synsfelt.

3.2 Nettside

Nettsiden vil gi en sanntidsrepresentasjon av radaren og det den detekterer. I første omgang tenker vi å kun presentere data fra det som har skjedd i nær fortid, men i fremtiden ønsker vi å presentere historisk data og resultat fra statistikk og maskinlæring.

3.3 Trådløs kommunikasjon

For at data fra sensoren til slutt skal komme opp på en nettside til sluttbrukeren, må vi ta i bruk trådløs kommunikasjon. For å gjøre dette mulig tenker vi å ha en liten datamaskin (for eksempel en Raspberry Pi) ved siden av sensoren, som er koblet på internett. Ved å sende all data gjennom internett kan vi behandle data i en sky og sende det ned til en nettside

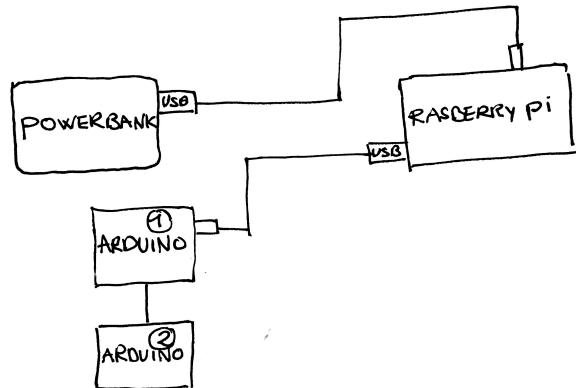


Figur 4: Nettsiden med sanntidsdata fra radaren.

4 Implementering

4.1 Kretsen

Dette systemet består av tre delsystemer, derav to arduinoer, og én raspberry pi. Disse delsystemene er koblet sammen som vist i figur 5.

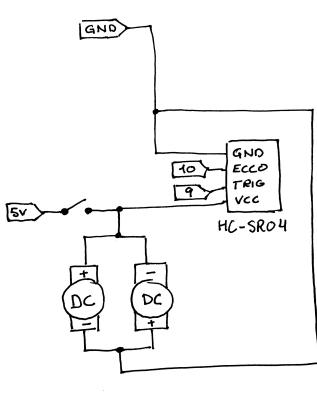


Figur 5: Sammenkobling av delsystemer

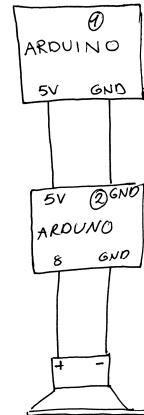
Hovedkretsen styrer DC-motorene og avstandssensoren. Komponenter som inngår i denne kretsen er to DC-motorer, en bryter samt en avstandssensor. Bryteren styrer spenningen til DC-motorene avstandsensoren. DC-motorene er koblet i parallel med motsatt polaritet

for å rotere motsatt vei. DC-kilden som leverer 5 volt til kretsen er en mikrokontroller av typen arduino. Denne tar også inn data fra avstandssensoren. Oppkobling mot Raspberry pi er beskrevet i avsnitt 4.3.

Den andre Arduinoen har en krets bestående av en høyttaler og en bryter som skur av og på høytaleren. Denne kretsen er illustrert i Figur 7. Bryteren står mellom pluss-polen på høytaleren og arduinoen.



Figur 6: Kretstegning hovedkrets



Figur 7: Sammenkobling av arduinoer og høytaler

4.1.1 Komponenter i kretsen

- 2 x DC-motorer - DC Worm Gear low speed motor, 5-12 RPM
- Avstandssensor - HC-SR04 Ultrasonic Sensor Distance Measuring Module
- 2 x Vippebrytere
- 2 x Arduino UNO (uoffisiell)
- Ledninger

4.2 Beregning av posisjon rundt vindmølle

For å synkronisere data fra radaren med nettsiden, er det ønskelig å kunne beskrive posisjonen til radaren rundt vindmøllen til en hver tid. Metoder som her kan implementeres er f.eks bruk av et kompass eller en GPS. I prototypen er posisjonen representert med en enkel tilnærming som vist nedenfor.

Posisjonsvektoren til radaren på nettsiden avhenger av rundetiden. Vi har gjort en måling med usikkerhet på rundetiden til radaren, T .

I Tabell 1 har vi målt rundetiden 7 ganger. Vi forkastet måling 1 fordi den avvok stort fra de andre målingene

Tabell 1

Måling nr	4	2	3	4	5	6	7
Rundetid i sek	40.9	34.9	34.6	34.8	34.1	34.3	34.0

Beregning av middelverdi for rundetiden:

$$\bar{T} = \frac{34,9+34,6+34,8+34,1+34,3+34,0}{6} = 34,5s$$

$$\Delta T = \frac{1}{2} \cdot (T_{maks} - T_{min}) = \frac{1}{2} \cdot (34,9s - 34,0s) = 0,5s$$

Rundetiden:

$$T = (34,5s \pm 0,5s)$$

Den relative usikkerheten:

$$\frac{\Delta T}{\bar{T}} = \frac{0,5s}{34,5s} = 1,4\%$$

4.3 Overordnet system

Den roterende avstandssensoren (1 i figur 8) har som oppgave å sende en kontinuerlig strøm av data over USB videre. Koden er en enkel arduino-kode som leser av fra en avstandssensor og printer resultatet. Motorene er styrt gjennom den elektriske kretsen og ikke software.

Den komplette koden for arduinoen kan leses i Appendix B. Den viktigste linjen er den siste, den som printer avstanden til Serial:

```
Serial.print(avstand)
```

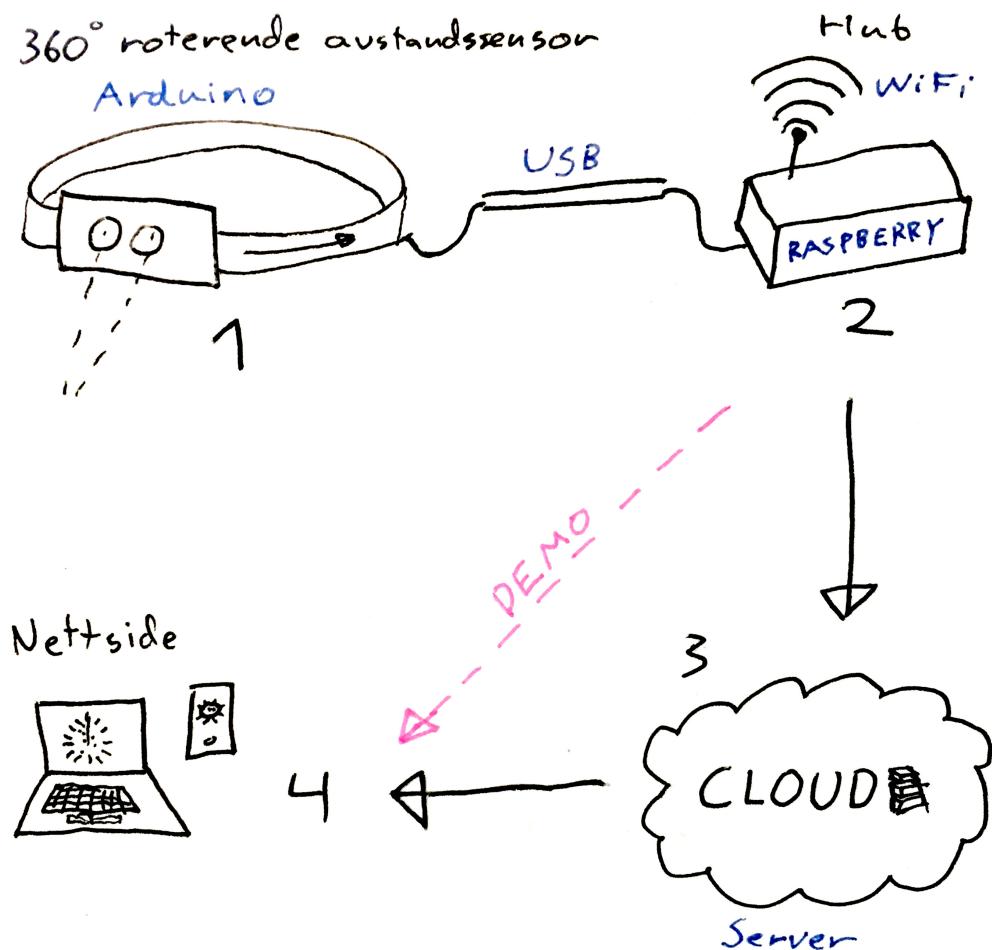
Videre fanges dataen opp i en Raspberry Pi (2 i figur 8) som gjør denne dataen tilgjengelig for omverdenen gjennom internett

Foreløpig fungerer Raspberry Pi'en som en fullverdig server, og cloud-steget vil derfor ikke bli brukt i demoen. I fremtiden vil det være mer hensiktmessig å sende dataen til clouden for videre prosesering og lagring.

Vi bruker Node.js som server på Raspberryen.

Vi bruker to hjelpeprogrammer som kan installeres vha. *npm*, *serialport*"og *socket.io*".

Ved hjelp av programmet SerialPort tar vi inn data fra Arduino gjennom USB-porten.



Figur 8: Vår signalløype, fra sensor til nettside

```

var port = new SerialPort('/dev/ttyUSB0');

// hver gang det kommer data på porten
port.on('readable', function () {

    // gjør noe
});

```

Vi bruker Socket.io til å streame data videre til tjenere.

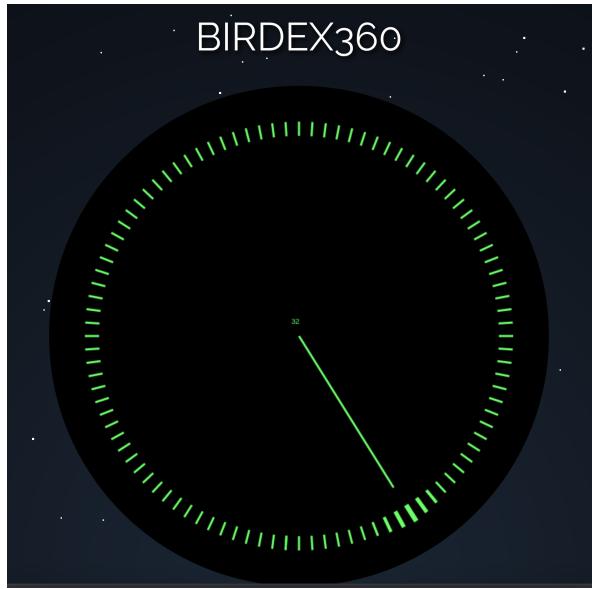
```

// hver gang det kommer data på porten
// emmitter en pakke 'a' med dataen
socket.emit('a', port.read().toString());

```

Resten av nodejs-koden finnes i Appendix C

Nettsiden (4 i figur 8) er en web-applikasjon skrevet i Javascript, HTML og CSS. Ved hjelp av javascript og HTML5/canvas kan vi tegne en grafisk radar i nettsiden.



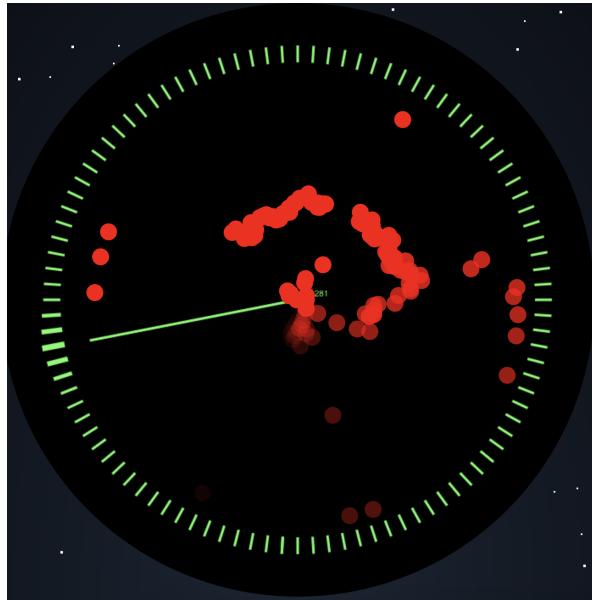
Figur 9: Radar tegnet i nettsiden.

Vi bruker Socket.io, og får inn data fra raspberryen. Vi bruker her IP-adressen til Raspberryen, i vårt tilfelle 172.20.10.10 og port 3000. Koden `socket.on('a')` vil kjøre hver gang applikasjonen mottar en melding med tittel 'a'. Vi bruker deretter funksjonen `registerHit` til å sende data videre i applikasjonen (se figur 10).

```
var socket = io('http://172.20.10.10:3000');

socket.on('a', function (data) {
    registerHit(data)
});
```

Komplett javascript-kode for nettsiden finnes i Appendix D



Figur 10: Radar tegnet på nettsiden. Hver røde prikk representerer når den har mottatt data fra sensoren. Avstanden fra senter representerer avstanden på ekte også.

5 Verifikasjon og test

Systemet vårt fungerer så godt som vi kan forvente på dette stadiet, med den kunnskapen og tiden vi har hatt til rådighet. Vognen går rundt på en skinne og sender data som den skal. På nettsiden får vi opp røde prikker når den mottar data.

En ting som fungerte dårlig er at motorene fikk litt for lite kraft. Dette kunne bli løst ved å gi motorene en separat spenning på 9V, istedet for å hente strømmen fra arduinoen. Det at skinna er laget av patentbånd gjør at bevegelsen rundt skinna blir ujevn, på grunn av hull i skinna og en sirkel som ikke er perfekt. Dette er noe som kunne blitt fikset ved å bruke en skinne uten ujevheter. En jevnere skinne ville også ført til at det er lettere å synkronisere farten til toget med radaren som blir vist på nettsiden.

6 Konklusjon

Ett av stegene for å kunne bygge en vindmølle er å forsikre at dens påvirkning på omstendighetene er minimal. Analyse av fuglelivet rundt en eksisterende, eller potensiell vindmølle er dermed viktig. Effektivisering av denne prosessen gjør det enklere for bedrifter som Trønderenergi å delta i FNs bærekraftsmål og bruke vindkraft som en miljøvennlig ressurs. Sensortoget med på å løse denne problemstillingen.

Vi fikk til en god demo som løser problemstillingen. Produktet vårt kan videreutvikles og få en reell virkning i den virkelige verden. Løsningen vår er enkel men smart, den ser 360 rundt seg og kan bli brukt både der det allerede er vindmøller og der det er planlagt utbygging av vindmøller. Usikkerheten av posisjonen til målingene våre er beregnet til å ligge på 1,4%, slik det kommer frem i avsnitt 4.2, noe vi er fornøyde med.

I en videreutvikling av dette prosjektet kan vi gå videre å behandle og bruke dataene vi får. Ved hjelp av flere sensorer og bedre innsamling av data vil vi kunne tilby innsikt og intelligente analyser som vil kunne presenteres for en kunde i et web-grensesnitt, og da vil vi ha et godt, ferdigutviklet system.

A Vedlegg - Nettside

Nettsiden er tilgjengelig her: <http://folk.ntnu.no/larskka/birdex360/>

B Vedlegg - Arduino

```
// konstanter
const int trigPin = 9;
const int echoPin = 10;

// variabler
long duration;
int distance;

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    pinMode(buzzer, OUTPUT);

    // set baud rate, viktig å ikke kørde med den her
    // fordi raspberryen må forstå serial
    Serial.begin(9600);
}

void loop() {
    // sett trigPin på lav og ha en liten delay
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);

    // Sett trigPinen på HIGH i 10 microsekund
    digitalWrite(trigPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    // Les echoPin, gir lydbølge sin reisetid i microsekunder
    duration = pulseIn(echoPin, HIGH);

    // kalkuler distansen
    distance = duration*0.034/2;

    // print distance til serial AKA send til raspberrry
    Serial.println(distance);
}
```

C Vedlegg - Node.js kode

```
var SerialPort = require("serialport");

var io = require('socket.io')();

var port = new SerialPort('/dev/ttyUSB0', { autoOpen: false });

port.open(function (err) {
  if (err) {
    return console.log('Error ved åpning av port: ', err.message);
  }
});

port.on('open', function () {
  console.log('Port opened successfully')
});

io.on('connection', function (socket) {
  socket.emit('event', {
    melding: 'Hei dette er en initiell melding fra raspberry pi om at ting fungerer'
  });

  // hver gang det kommer data på porten
  port.on('readable', function () {
    // emmitter en pakke 'a' med dataen
    socket.emit('a', port.read().toString());
  });
});

io.listen(3000);
```

D Vedlegg - Nettsidekode

Relevant HTML:

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/svg.js/2.6.6/svg.min.js"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/socket.io/2.1.1/socket.io.js"></script>
<canvas id="canvas" width="700" height="700"></canvas>
<script src="script.js"></script>
```

Javascript (script.js):

```
/* canvas init */
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
const w = canvas.width;
const h = canvas.height;

/* svg init */
const svgjs = SVG('svgcont')
var bird = svgjs.circle().radius(10)
bird.fill('red').move(-9999, -9999)

/* vars */
const durationAround = 10000;
let currentDegree;
let forrigeUtslag = 50
const hits = []
const beginningTime = new Date()

/*
    Canvas drawing functions:
*/

/* Draw background with all taps */

function drawBG(currentDegree) {
    ctx.fillStyle = "black"
    ctx.fillRect(0, 0, w, h)

    ctx.strokeStyle = "#6bff68"

    const taps = 100
```

```

for (let i = 0; i < taps; i++) {
    const degree = (i / taps) * Math.PI * 2

    let extraLength
    let color
    extraLength = 0
    color = '#6bff68'

    let close
    const diffDegree = (degree - currentDegree) % (Math.PI * 2)
    if (Math.abs(diffDegree) < 0.2) {
        close = 1 - Math.abs(diffDegree) / 0.2
    } else {
        close = false
    }

    if (close) {
        ctx.lineWidth = 3 + 4 * close
    } else {
        ctx.lineWidth = 3
    }

    ctx.strokeStyle = color
    ctx.beginPath();
    const from = degreesToXY(degree, 280 - extraLength)
    ctx.moveTo(from.x, from.y)
    const to = degreesToXY(degree, 300 + 7 * close)
    ctx.lineTo(to.x, to.y)
    ctx.stroke()
}
}

/* Draw radar */
function drawRadar(currentDegree) {
    ctx.lineWidth = 3
    const from = degreesToXY(currentDegree, 0)
    ctx.moveTo(from.x, from.y)
    const to = degreesToXY(currentDegree, 250)
    ctx.lineTo(to.x, to.y)
    ctx.stroke()
}

/* Draw text */
function drawText(currentDegree) {
    ctx.fillStyle = "#6bff68"
    const from = degreesToXY(currentDegree + Math.PI, 20)
    ctx.fillText(Math.round(currentDegree / Math.PI * 180), from.x, from.y)
}

```

```

}

/* Calls all draw functions continuously */

function renderLoop() {
    // clear canvas
    ctx.fillRect(0, 0, w, h);

    const nowTime = new Date();

    const durationTime = (nowTime - beginningTime) % durationAround
    const progress = durationTime / durationAround
    currentDegree = progress * Math.PI * 2

    drawBG(currentDegree);
    drawRadar(currentDegree)
    drawText(currentDegree)

    requestAnimationFrame(renderLoop);
}

renderLoop();

/* Trigonometric helper function */
function degreesToXY(degrees, distance) {
    const half = w / 2
    return {
        x: half + Math.sin(degrees) * distance,
        y: half + Math.cos(degrees) * distance
    }
}

/* Add (svg) bird */
function addBird(degree, distance) {
    if (distance > 50) {
        return
    }
    const from = degreesToXY(degree, Math.min(250 * (forrigeUtslag / 50), 250))

    bird.clone().move(from.x, from.y)
        .animate({
            delay: (durationAround / 2) - 1000,
            duration: 5000
        }).opacity(0)
}

```

```

function registerHit(data) {
    hits.push({
        degree: currentDegree,
        distance: 0.5,
        bird: addBird(currentDegree, data)
    })
    forrigeUtslag = data
}

/*
    Connectivity:
*/
var socket = io('http://172.20.10.11:3000');
socket.on('connect', function () {
    console.log('connected')
});

socket.on('a', function (data) {
    console.log('asdf', data)
    registerHit(data)
});

socket.on('disconnect', function () {
    console.log('disconnected')
});

```