CHALMERS
UNIVERSITY OF TECHNOLOGY

Aktivt säkerhetsystem för cykel

Framtagning av prototyp för aktivt säkerhetssystem för applicering på cykel

Examensarbete inom Mekatronik

JONAS LARSON
JOSEFIN VILHJALMSSON

Institutionen för Elektroteknik

Examensarbete inom Mekatronik 2019:01

Aktiva säkerhetssystem för cykel

Framtagning av prototyp för aktivt säkerhetssystem för applicering på cykel

JONAS LARSON

JOSEFIN VILHJALMSSON

CHALMERS

Institutionen för Elektroteknik
Chalmers tekniska högskola
Göteborg, Sverige 2018

Aktivt säkerhetsystem för Cykel
Framtagning av prototyp för aktivt säkerhetssystem för applicering på cykel
JOSEFIN VILHJALMSSON
JONAS LARSON

Handledare: Fredrik Hansson, Johan Svensson, Conmore Ingenjörsbyrå AB
Examinator: Göran Hult, Institutionen för elektroteknik

Institutionen för elektroteknik
Chalmers tekniska högskola
412 96 Göteborg
Telefon +46 31 772 1000

Förstasida: En tidig visualisering av projektets önskade slutresultat.

Göteborg, Sverige 2018

Sammanfattning

Enligt Statens väg- och transportforskningsinstitut ökade cykeltrafiken i Malmö mellan 2003 och 2015 med 53% och liknande trender kan ses i många större städer. Med fler cyklister ökar också risken för olyckor i vilka cyklister oftare skadas allvarligt. I detta arbete har en prototyp på ett aktivt säkerhetssystem för att varna cyklister för hinder och andra fordon tagits fram. Dessa system finns sedan tidigare i andra fordon men har ännu inte släppts som en kommersiell produkt för cyklar.

För att få kunskap om hur dessa system fungerar undersöktes liknande system på bilar och vi träffade Claes Broberg, Manager på Collision Avoidance Functions hos Volvo Cars. Efter mötet bestämdes det att ett färdigt system borde använda en radar men för den första prototypen skulle ultraljudssensorer tillsammans med en Arduino användas.

För att få så god täckning som möjligt används fem ultraljudssensorer. Sensorerna mäter distansen till omkringliggande objekt och använder denna information för att beräkna hastigheten på objekten för att i sin tur varna cyklisten. Cyklisten varnas genom lysdioder placerade på styret och vibrationsmotorer placerade i handtagen.

En fungerande prototyp togs fram. Den är inte provad på en cykel utan enbart i laboratoriemiljö. Funktionaliteten på varningssystemet uppfyller de ställda kraven.

I ett framtida system kommer prototypens ultraljudssensorer att ersättas med en short range radar. En ARM-processor kommer att ersätta Arduinon som används för att göra beräkningarna. Även monteringsdetaljer kommer att vidareutvecklas.

Keywords: cykel, aktivt säkerhetssystem, kollisionsundvikande, sonar.

Abstract

According to Swedish Statens väg- och transportforskningsinstitut, bicycle traffic in Malmö increased between 2003 and 2015 by 53% and similar trends can be seen in many major cities. With more cyclists the risk of accidents, in which cyclists are more often injured, are increasing. In this thesis work, a prototype of an active safety system has been developed for warning cyclists for obstacles and other vehicles. These systems have previously been offered in other vehicles but has yet to be released as a commercial product for bicycles.

To get knowledge about how these systems work, similar systems in cars were investigated and we met Claes Broberg, Manager at Collision Avoidance Functions at Volvo Cars. After the meeting, it was decided that a finalized system should use a radar but for the first prototype would use ultrasonic sensors together with an Arduino.

In order to obtain the best possible coverage, five ultrasonic sensors are used. The sensors measure the distance to the surrounding objects and use this data to calculate the speed of the objects to warn the cyclist in turn. The cyclist is warned through light emitting diodes placed on the handlebar and vibration motors placed in the handles.

A working prototype was developed but it has not been tested on a bicycle but only in a laboratory environment. The functionality of the warning system meets the set requirements.

In a further developed system, the prototype ultrasonic sensors will be replaced with a single shorts range radar. An ARM processor would replace the Arduino used to make the calculations. Mounting details will also be further developed

Keywords: bicycle, active safety system, collision avoidance, sonar.

Förord

I denna rapport presenteras hur aktiva säkerhetssystem kan anpassas för applicering på en cykel. Arbetet är ett examensarbete på Chalmers tekniska högskola hos institutionen för Elektroteknik. Arbetet gjordes på Conmore ingenjörsbyrå AB, som är ett konsultbolag riktat mot produktutveckling, processindustri samt elektronik och mjukvara.

Vi vill först tacka våra handledare Fredrik Hansson och Johan Svensson på Conmore och Göran Hult på Chalmers. Er stöttning och ert engagemang hjälpte oss igenom arbetet. Vi vill också tacka Claes Broberg på VCC som tog sig tid att diskutera med oss hur aktiva säkerhetssystem fungerar och gav oss många bra tips på tillvägagångssätt. Vi vill också tacka Joachim von Hacht på Chalmers för att han hjälpte oss med datatyper och att få en fungerande och snabbt exekverande kod.

Jonas Larson och Josefin Vilhjalmsson, Göteborg, 2018

Innehåll

Förkortningar

- Arduino shield — Ett kretskort som koppplas in ovanpå Arduinon
- CPU — Central Processing Unit
- LED — Light Emitting Diode
- IDE — Integrated Development Environment
- PLA – Polylaktid är en naturligt nedbrytningsbar termoplast som ofta används för 3D-utskrift
- VTI – Statens väg- och transportforskningsinstitut

1
Introduktion

1.1 Bakgrund

Ur ett hållbart perspektiv är cykeln ett bra alternativ vid transport. Enligt Statens väg- och transportforskningsinstitut ökade cykeltrafiken i Malmö mellan 2003 och 2015 med 53% och liknande trender kan ses i många större städer [1]. Med fler cyklister ökar också risken för olyckor i vilka cyklister oftare skadas allvarligt. Samtidigt väljer många att använda eldrivna cyklar vilket skapar en större hastighetsskillnad mellan cyklister och kan upplevas obehagligt vid exempelvis omkörningar. Statens väg- och transportforskningsinstitut har i undersökningar kommit fram till att de flesta olyckor sker med motorfordon eller cyklister emellan och en stor anledning är troligen dålig uppsikt hos cyklisten [1]. På marknaden finns idag inga system som aktivt kontrollerar omgivningen på liknande sätt som hos bilar.

Automatisering kan höja säkerheten i trafiken då information från omgivningen tas in och behandlas säkrare och därmed kan fordonsföraren fatta snabbare och bättre beslut. SAE international, som är en organisation som arbetar inom teknikbranchen med fokus på fordonsindustrin, har utvecklat ett klassificieringssystem för självkörande bilar. Systemet har sex nivåer där noll är ingen automatisering och fem är fullt självkörande fordon [2]. En av de första grundläggande principerna tillämpar aktiva säkerhetssystem som behandlar information om sin omgivning för att observera hinder och objekt.

1.2 Syfte

Syftet med projektet är att undersöka hur man kan applicera ett aktivt säkerhetssystem på cyklar genom att använda avståndssensorer samt att ta fram en prototyp som ska testas på en cykel. Detta projektet ska ligga till grund och därmed vara första steget mot att skapa en fungerande självkörande cykel.

1.3 Mål

Systemet ska med hjälp av sensorer upptäcka hinder som befinner sig i närheten av cykeln. Genom att kontinuerligt mäta avståndet till objekt samt beräkna dess hastighet ska systemet varna föraren för faror utan att påverka körförmågan.

Föraren ska enkelt kunna se på styret om något befinner sig bakom cykeln samt genom vibrationer i handtagen veta om något närmar sig i hög hastighet.

Följande mål ska också behandlas:

- Ta fram lämpliga komponenter för att mäta avstånd längre än 4 meter och mäta hastigheter
- Konstruera prototyphöljen och fästen med hjälp av CAD
- Skriva ut samtliga delar genom 3D-printing
- Konstruera kretschema i KiCad

1.4 Problemformulering

För att uppfylla målen kommer följande frågor att behandlas.

- Vad finns på marknaden idag?
- Hur kan man applicera aktiva säkerhetssystem på cyklar?
- Vilka sensorer är lämpligast att använda?

1.5 Avgränsningar

Följande punkter beskriver projektets avgränsningar.

- Projektet avgränsas till att ta fram ett aktivt säkerhetssystem till en cykel (ej fullt automatiserat)
- Systemet kommer att varna cyklisten med hjälp av ljus och vibrationer och inte genom styrning eller inbromsning för att inte påverka körförmågan.
- Programmet kommer att programmeras i Arduinos egna IDE.
- Projektet kommer ej ta hänsyn till en budget, dock ska kostnader hållas så låga som möjligt.
- Val av komponenter kommer delvis göras beroende på kostnad men mestadels på leveranstid, då projektet kräver snabb tillgång till dessa.
- Prototypen kommer testas på en kontorsstol och därmed kommer fästet anpassas för detta.

2
Teoretisk referensram

Nedan presenteras teknisk bakgrund till de begrepp och komponenter som används. Informationen som tas fram kommer ligga till grund för projektets genomförande.

2.1 Avståndsmätning

De vanligaste metoderna för avståndsmätning är med sonar-, radar- och lidarteknik. Samtliga metoder har det gemensamt att de utifrån en pulsvåg mäter tiden det tar för vågen att färdas fram och tillbaks mellan sensorn och objektet.

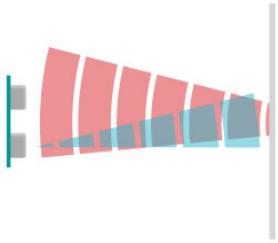
2.1.1 Sonar

Sonar står för Sound Navigation And Ranging och är en teknik som skickar ut en ljudvåg med en viss frekvens, se figur 2.1, och med hjälp av ljudets hastighet beräknas avståndet s mellan sensor och objekt enligt formeln:

$$s = \frac{t * v}{2}$$

Där t är tiden för ljudet att färdas fram och tillbaka och v är ljudets hastighet. Ljudets hastighet och vald frekvensen varierar beroende på användningsområde.

Sonarsensorer används under vattnet med ljudvågor av en låg frekvens. Medan högre frekvenser, runt 40kHz, används för sensorerna i luften. Dessa sensorer kallas ultraljudsensorer. Ultraljudssensorer används som back- och parkeringssensorer på bilar där de varnar om man kör nära något eller för att automatiskt hitta och placera bilen i en ledig parkeringsplats. Nackdelen är att ultraljudsensorer är känsliga för störningar samt att de inte kan mäta längre distanser, jämfört med sonarsensorer med lägre frekvens[3].



Figur 2.1: Sonarsensor

2.1.2 Radar

Radar, Radio Detection And Ranging, fungerar på ett liknande sätt som sonar men istället för ljudvågor används radiovågor. Radar har under en längre tid används i bilar för att identifiera faror och hinder framför bilar. Det används också ofta för adaptiv farthållare som ställer in hastigheten efter bilen framför. Nackdelen med radar är att den kan mäta avståndet till ett objekt men det kan vara svårt att veta var det objektet befinner sig i rummet. Därför används oftast en kamera som komplement för att på så sätt kunna kombinera datan och få en mycket precis modell av omgivningen[4]. Fördelen med radar är att de har bred synvinkel och är inte lika känslig för yttre störningar så som Sonar.
Tidiga radarsystem hade problem att mäta avstånd i fuktiga miljöer men tekniken har utvecklats för att kringgå dessa svagheter. Moderna radarsystem kan mäta avstånd i dimma, regn och snö. Dessutom behöver radarsystem inte kunna se objektet för att mäta distansen till det. Detta kan till exempel göras genom att studsa radiovågorna under framförvarande bilar. En radar kan mäta avstånd upp till 50 kilometer men under bra förhållanden kan de mäta upp till flera 100 kilometer.

2.1.3 Lidar

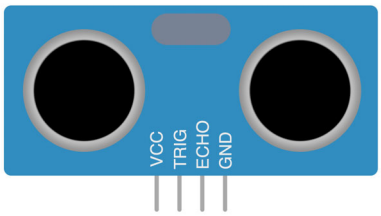
Namnet Lidar är en förkortning av Light Detection And Ranging och en sammanslagning av Light och Radar. Den mäter avstånd genom att skicka ut pulser av laserljus istället för en radio- eller ljudvåg. Pulserna reflekterats tillbaka till sensorn och tiden som det tog från att ljuset skickades till att det reflekterades tillbaka används för att beräkna distansen till objektet.[5] Då ljusets hastighet är snabbare än ljudets kan en lidarsensor göra fler mätningar på en given tid.

En fördel med Lidar är dess långa räckvidd och höga mätsäkerhet. Den kan mäta på många olika material och kan även mäta oregelbundna objekt. Däremot är synvinkeln snäv vilket innebär att lidarsensorn behöver svepas eller snurra för att kunna få in information från en större yta. Lidartekniken används ofta vid kartläggning både monterade på bilars tak men också monterade på helikoptrar och flygplan.

Ljuset som används i Lidar har en kortare våglängd än radiovågorna hos en radar vilket gör att Lidar kan mäta med en högre upplösning. Tyvärr medför den kortare våglängden att Lidar lättare kan störas av partiklar vilket gör att den fungerar bäst i torrt och klart väder.

2.2 Ultraljudsensor HC-SR04

Ultraljudsensorn HC-SR04 är en distanssensor som använder sig av ultraljudvågor på 40kHz. Den är en liten, strömsnål och mycket billig sensor. Sensorn har fyra ben, Vcc(matningspänning), Gnd(jord), Trig och Echo där Trig skickar ut en puls och Echo tar emot pulsen. Den kan mäta avstånd från två centimeter upp till fyra meter med en nogrannhet på tre millimeter [6]. Den har dock nackdelarna att den under vissa förhållanden kan ge mätfel och spikvärden vilket medför att den kan kräva en del signalbehandling.



Figur 2.2: HC-SR04

2.3 Arduino Uno

Arduino UNO är en mikroprocessor baserad på öppen källkod med en Atmel ATmega328P [7] mikrokontroller. Fördelen med Arduino är att den är snabb att komma igång med, mycket anpassningsbar samt billig och lättillgänglig. Det är enkelt att koppla upp sensorer och andra komponenter till de 20 in/ut-portarna[8]. Detta gör det lämpligt att bygga prototyper med Arduino och den färdiga koden kan enkelt laddas in i andra mikroprocessorer för att användas i den slutgiltiga produkten.

Programmering sker via Arduinos egna IDE eller med andra IDEs via tilläggsprogram. Språket som används är en förenklad version av C. Eftersom Arduino bygger på öppen källkod finns det många färdiga kodexempel och bibliotek att hämta på internet. Dock kan inte färdiga bibliotek användas om en annan IDE används. Programmet som skrivs laddas över direkt till Arduinon och data från till exempel sensorer kan visas i realtid på datorns skärm.

2.3.1 Data-typer

För optimering av processortiden behöver man ta hänsyn till datatyperna då de tar olika långt tid vid beräkningar. I ett realtidssystem är det mycket viktigt att exekveringstiden hålls låg för att få ett system som reagerar snabbt på yttre händelser. Detta betyder att lämpliga datatyper måste användas för att både exekveras snabbt och ge möjlighet att lagra den nödvändiga datan. Användaren DBgit på Arduinos officiella forum har tagit fram ett program som beräknar genomsnittstiden och antalet CPU-cykler det tar att exekvera kod. Detta program modifierades och användes för att testa hur lång tid olika operationer tog med olika datatyper. Resultatet visade att Long är den datatyp som har en kombination av relativa korta exekveringstider och möjlighet att lagra mycket stora tal, såvida om division används till ett minimum.

Testkod av DBgit på arduinoforum[9], se bilaga 1. De olika datatypernas storlek hämtades från Arduinos hemsida[10].

Tabell 2.1: Beräkningstid för olika datatyper

Datatyp	int 16bit		long 32bit		float 32bit		double 32bit	
Operation	µs	cykler	µs	cykler	µs	cykler	µs	cykler
$a = b + c$	0,870	14	1,751	28	7,853	126	7,791	125
$a = b - c$	0,870	14	1,751	28	8,120	130	8,139	130
$a = b \cdot c$	1,374	22	6,090	97	9,704	155	9,696	155
$a = b \div c$	14,340	229	38,895	622	30,920	495	30,719	492

3
Metod

Följande kapitel presenterar de metoder och verktyg som användes under projektets gång. Metoderna ligger till underlag för framtagning och resultatet av en färdig prototyp.

3.1 Förstudier

Under projektets inledande fas görs en förstudie i syfte att få en bra överblick inom området. Tillförlitlig information samlas in genom att studera litteratur, databaser samt genom intervjuer och marknadsundersökningar. Den information som sammanställs kommer vara den huvudsakliga underlaget till projektet.

3.2 Kravspecifikation

Vid framställning av en produkt tillkommer förväntningar från uppdragsgivare, användare och andra intressenter. Genom att ställa upp en kravspecifikation sammanställs de krav och önskemål som förväntas på ett organiserat och ordnat sätt. Kravspecifikationen används sedan för att ta fram och ta bort olika idéer och koncept.

3.3 Flödesdiagram

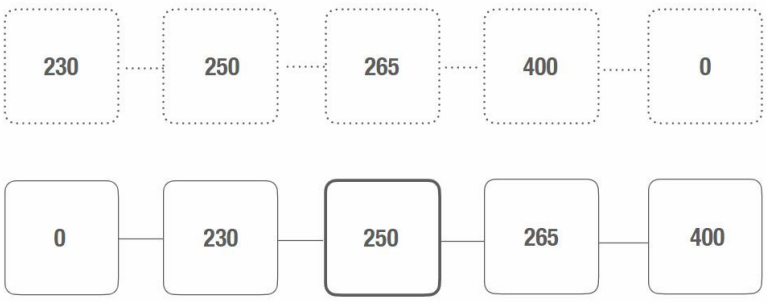
Ett system består oftast av flera delfunktioner som systematiskt kopplas samman genom villkor. För att få en bättre förståelse av processen görs ett flödesdiagram som är en grafisk illustration över hur processen är uppbyggd.

3.4 Filtrering av spikvärden

Störningar i olika former sker kontinuerligt vilket kan resultera i spikande värden. Orsaken till detta kan bero på dåliga sensorer eller yttre faktorer där sensorn inte är avsedda för den miljön de utsätts för. Avvikelser medför försämrad exakthet och beslut baseras på felaktiga värden. För att få en mer tillförlitlig distansmätning behandlas mätdatan på olika sätt för att få bort de avvikande värdena.

3.4.1 Median

För att stabilisera det uppmätta värdet och filtrera bort värden som sticker ut, tas ett medianvärde för varje sensor fram. Detta görs genom att ett angivet antal mätvärden från varje sensor sparas i ett fält. Fältet sorteras i storleksordning och medianvärdet tas ut. Se figur nedan.



Figur 3.1: Fält med osorterade och sorterade värden

3.4.2 Extrapolering

I det färdiga programmet är det tänkt att om en lång sträng av ogiltiga mätvärden kommer in skall approximativa värden extrapolering från de senast giltigt uppmätta värdena. Detta görs genom att distansskillnaden mellan de sista giltiga värdena beräknas för att få en distansskillnad över en tid. Det används för att sedan beräkna nya värden.

3.5 Prototypframtagning

För att få en uppfattning av hur produkten kommer att fungera på en cykel tas en prototyp fram. En användbar metod är att konstruera produkten i CAD och skriva ut den med 3D-printer.

4
Analys

Projektet har delats in i fyra projektsteg där analys är första steget. I detta kapitel utformas projektets struktur och uppbyggnad genom att samla relevant information och studera de krav som ställs.

4.1 Förstudier

Att upptäcka och undvika hinder är en funktion som redan används i andra system i bilar, robotar och drönare. En förstudie görs i syfte att undersöka i vilken situation olyckan sker samt vilka lösningar som tillämpas för att undvika dessa situationer.

4.1.1 Olyckor

Enligt VTI sker cykelolycker oftast till och från jobbet och 90 procent av fallen är i tätort. De flesta olyckor där cyklisten blivit allvarligt skadad, sker med motorfordon och andra cyklar. För att få en överblick på hur olyckorna sker skissades olika situationer som kan uppkomma. Dessa situationer kan delas in i tre typer: korsande möten, i samband med svängning eller med bakifrån kommande trafik, se bilaga 2. Gemensamt för samtliga situationer är att det är svårast för cyklisten att ha full kontroll på vad som sker bakom.

4.1.2 Tillämpade lösningar

Litteraturstudier visade att vid avståndsmätning används sonar-, radar- eller lidarsesnorer. Val av sensor beror på budget och användningsområde då de har olika egenskaper. Det är också fördelaktigt att kombinera olika sensorer för att minimera risken för störningar.

Ett möte med Claes Broberg, chef på Collision Avoidance Functions hos Volvo Cars, bokades i syfte till att få kunskap om hur existerande system fungerar samt diskutera sensorernas egenskaper. Claes berättade om hur de arbetar med aktiva säkerhetssystem på Volvo samt vilka sensorer som används och dess användningsområden. För att få snabba och precisa mätvärden rekommenderade han att använda objektsigenkänning med hjälp av kameror kompletterat med radar eller lidar.

4.2 Kravspecifikation

De krav och önskemål som presenteras nedan är baserade på vad som ansågs relevant vid projektets start.

Tabell 4.1: Kraven som sattes vid projektets start

Funktion:	Krav	Önskemål
Observera objekt bakåt	x	
Varna föraren för faror inom 4m	x	
Varna föraren för faror med hög hastighet	x	
Komponenter med lågt pris och snabb leveranstid	x	
Tydlig kommunikation mellan systemet och föraren	x	
Ska skilja på faror		x
Programmet ska vara så snabbt som möjligt		x

4.3 Flödesschema

I bilaga 3 visas flödesdiagrammet för programmets generella struktur. Programmet läser och gör beräkningarna för en sensor åt gången.

4.4 Val av sensor

Tabell 4.2: Grundläggande data för de olika sensorerna

	Sonar	Radar	Lidar
Uppfyller funktionskrav:	Ja	Ja	Ja
Räckvidd:	4 m	25 m	40 m
Cirka storlek:	13,5 cm^3	30 cm^3	40 cm^3
Pris i kr:	60	1 300	800
Tillgänglighet:	Direkt	ca 2 mån	ca 2 mån

Då arbetet innefattar att ta fram en första prototyp ligger fokuset och utmaningen i att få ett program att fungera. Därför är pris och snabb leverans de två faktorer som väger tyngst. Sonarsensorer, HC-SR04, används i detta arbete och eftersom utvecklingen sker i labbmiljö räcker sonarsensorns korta räckvidd.

5
Mjukvara

Projektsteg två innefattar bearbetning av information och data från sensorerna. Programmet delas upp i olika funktioner, där varje funktion uppfyller kraven.

5.1 Avståndsmätning

För att få en snabb och precis distansmätning från sensorn används biblioteket NewPing [11]. Detta gör det enklare att hämta data från flera sensorer. NewPing kan också i många situationer räkna hastigheten snabbare än det inbyggda biblioteket. Data lagras i en kö för att sedan bearbetas.

5.2 Filtrering av spikar

En ultraljudssensor är relativt känslig för yttre störningar. Störningarna kan bero på ultraljudsvågor från andra sensorer, maskiner i omgivningen, materialet på objektet som vågorna studsar emot eller att objektet inte är tillräckligt parallellt med sensorn. Detta medför ofta att sensorn mäter korta spikar av mycket höga eller låga mätvärden. Därför används en medianfunktion som sparar de tolv senast uppmätta värdena i ett fält och sorterar sedan dessa värden efter storlek för att hitta ett medianvärde. Detta medianvärde är sedan det värdet som används i resten av programmet. Detta gör att spikvärden och annat brus blir bortsorterat.

Om ett objekt rör sig i en mycket hög hastighet mot sensorn händer det i vissa fall att det inte registreras korrekt. Det kan orsakas av att den uppmätta distansen blir noll eller maxvärdet för vad sensorn kan mäta uppnås. Detta beror på att NewPing sätter mätvärdet då inget giltigt mätvärde uppmätts inom en given tid. Om detta sker kan sensorn i vissa fall till en början ge bra mätvärden för att efter någon sekund ge den felaktiga distansen. För att approximera var objektet är och ändå kunna varna cyklisten används en extrapoleringsfunktion. Den använder de sista giltiga uppmätta värdena för att beräkna en hastighet och använder den för att ta fram nya värden tills sensorerna ger korrekta värden igen.

5.3 Hastighetsberäkning

Hastigheten är tidsderivatan av sträckan och fås approximativt enligt formeln nedan. För att göra detta i Arduino behövs två uppmätta sträckor och tiden mellan de två distansmätningarna.

$$v = \frac{\Delta s}{\Delta t}$$

Eftersom beräkningarna behöver vara snabba vill man undvika att låsa programmet vid en hastighetsberäkning genom exempelvis delays eller for-loopar. Därför används inga fasta tider utan istället sparas tiden från Arduinons interna klocka varje gång en avståndsmätning görs.

5.4 Varningssystem

Varningssystemet aktiveras beroende på olika villkor enligt figur 5.1. Stjärnorna i figuren representerar antalet lysdioder som tänds och beror på objektets avstånd. Exempelvis tänds två lysdioder när ett objekt befinner sig inom fyra meter. Varningssystemet tar även hänsyn till objektets placering dvs att ett objekt som befinner sig på höger sida tänder lysdioder på höger handtag och vice versa. När objektet närmar sig i hög hastighet startas vibrationsmotorerna i båda handtagen.

Tabell 5.1: Varningar

Villkor:	Händelser
Inom 4m:	**
Inom 3m:	** **
Inom 2m:	** ** **
Inom 1m:	** ** ** **
Hög hastighet:	Vibrationer i handtagen

6
Hårdvara

6.1 Ultraljudssensor

Ultraljudssensorn HC-SR04 används för att mäta avstånd. Det är en billig och kompakt sensor som använder ljudets hastighet för att mäta avstånd. En fördel med ultraljudssensorer är att de fungerar i alla typer av ljus. Nackdelar är att de kan bli störda av andra ultraljud och har svårt att detektera material som absorberar ljudvågor. Eftersom säkerhetssystemet till största del ska detektera hårda material ska detta inte vara något problem för prototypen.

Tabell 6.1: Data för ultraljudssensorn HC-SR04

HC-SR04	
Drivspänning	5V DC
Strömförbrukning i standby	2 mA
Strömförbrukning vid användning	15 mA
Mätvinkel	$<15°$
Längsta mätavstånd	4 m
Kortaste mätavstånd	2 cm
Mätfel	± 3 mm

6.2 Varningssystem LED

För att informera cyklisten om riskabla objekt sitter lysdioder på cykelns styre. Det är åtta på vardera sida om styret som visar cyklisten vilken sida eller hur nära objektet är genom antalet tända lysdioder. För att spara på antalet portar som används på microprocessorn används en ledramp med en integrerad controller, istället för enskilda lysdioder som kräver en utport per lysdiod. Detta gör det möjligt att bara använda 2 portar för tända eller släcka 16 lysdioder enskilt.

Tabell 6.2: Data för LED-stripen

LED Strip 5050 RGB	
Drivspänning	4-7V DC
Ljusstyrka	275 cd/m^2
Uppdateringsfrekvens	400 Hz
Temperatur vid användning	-25°c – 50°C

6.3 Varningssystem vibrator

Om ett objekt rör sig mot cyklisten i mycket hög hastighet kommer också handtaget på den sidan att börja vibrera och på så sätt informera cyklisten om faran. För att göra detta används två DC-motorer med obalanserade vikter på dess axlar. Dessa motorer kräver en högre ut-effekt än vad portarna på Arduino kan ge. Därför används en transistor för varje motor.

Tabell 6.3: Data för vibrationsmotorerna

El-motor, PPN13LB11B	
Drivspänning	5V – 12V
Varvtal	16000 RPM
Vikt	39g

6.4 CAD

I projektet togs flera prototyphöljen och fästen fram för att hålla sensorer och Arduino. Dessa höljen konstruerades i Catia V5 och skrevs ut i röd PLA, Polylaktid, med en 3D-skrivare.

7

Prototypframtagning

För att testa systemet i verkligheten togs höljen och fästen fram för att montera systemet på en kontorsstol då det ej finns tillgång till en cykel.

7.1 Höljen

Flera CAD modeller togs fram för att kunna montera sensorer i och fästen för att montera dessa höljen på en kontorsstol. Därefter skapades en verklig prototyp av modellen genom 3D-printing.

Figur 7.1: Renderingar av systemet monterat på cykel

7.2 Kretskort

Ett prototypkretskort för att montera de nödvändiga komponenterna tillverkades. Detta togs fram med hjälp av ett experimentkort det vill säga en glasfiberplatta med hål i och kopparbanor som sammanbinder hålen. Kortet designades för att anslutas till Arduinon som en shield, se bilaga 7.

7.3 Montering i laboratoriemiljö

De första proven gjordes i innomhusmiljö. Planen att använda en kontorsstol fick överges. Istället monterades alla ingående komponenter på ett bord försett med hjul. Detta bord rullades mot olika föremål för att studera systemets funktion. Dessutom användes en skärm för att simulera ett fordon som rör sig mot cykeln.

Resultaten från dessa inledande prov användes för att optimera program och sensorplaceringar. Den slutliga lösningen dokumenterades med en video som lades upp på YouTube [12].

8

Resultat

Syftet med det här projektet var att undersöka och ta fram ett aktivt säkerhetssystem som kan monteras på en cykel. Vi insåg att det inte var möjligt att ta fram ett system på den nivån vi först tänkt inom den givna tidsramen och tog därför enbart fram en fungerande prototyp som visar att iden fungerar i verkligheten och att systemet troligen skulle kunna höja säkerheten för cyklister. Prototypen till systemet är enbart tänkt att monteras bak på cykeln. Prototypen som togs fram använder fem ultraljudssensorer för mätning av data och en Arduino Uno för databehandling.

Systemet kommunicerar informationen till föraren genom att tända lysdioder om något befinner sig inom fyra meter eller vibrationer i handtagen när något närmar sig i hög hastighet. Antalet lysdioder som tänds ökar när objektet närmar sig samt ändras från gul till röd färg när den är inom två meter, en cykellängd. Systemet tar även hänsyn till vilken sida faran befinner sig eller om den är rakt bakifrån.

Figur 8.1: Illustration på hur varningssystemet fungerar

Tiden för att köra hela programmet en gång varierar. De faktorer som påverkar tiden mest är distansmätning med sonarsensorerna, hastighetsberäkningar och att uppdatera lysdiodrampen. Anledningen till att distansmätningen ökar tiden är på grund av att ljudvågorna måste färdas fram och tillbaka mellan sensorn och objektet innan en mätning kan fastställas. Detta gör att om det sker snabba hastighetsförändringar långt från sensorn kommer programmet att ta mycket längre tid att köras jämfört med små förändringar nära sensorn. Lysdiodrampen kan

också ta lång tid att uppdatera då varje lysdiod måste adresseras var för sig. Detta medför att uppdatering av många lysdioder på en gång tar lång tid.

I tabellen nedan presenteras ungefärliga uppdateringstider för olika mätningar. De två första mätningarna visar maximala tiden då en hastighetsförändring sker och när ingen hastighetsförändring sker. Tredje mätningen visar minsta tiden programmet kan uppdateras.

Tabell 8.1: Tiden mellan mätningar

Max med hastighetsförändring	90ms
Max utan hastighetsförändring	41ms
Min	5ms

Den givna tidsramen gav ingen möjlighet att ta fram ett fullt fungerade system. Arduinoprototypen visade att idéen var genomförbar och att det finns en god möjlighet att förbättra säkerheten för cyklister.

Vi har tagit fram ett system som fungerar inomhus i kontrollerade miljöer. För att ta fram ett robustare och bättre system behövs mer arbete för att hitta lämpliga sensorer som kan användas i de miljöer och förhållanden som en cykel utsätts för.

I våra tester har vi sett att placeringen av sensorerna fungerar bra för den valda typen av sensorer. I en framtida utveckling av systemet kan en annan placering av sensorerna dock vara nödvändig för att uppnå lika bra eller bättre mätdata. Det skulle också vara föredelaktigt att kombinera datan från olika typer av sensorer för att förhindra att systemet blir lättstört. Nackdelen med detta är att strömförbrukningen för systemet skulle riskera att öka mycket.

Från vad vi har sett finns det få lämpliga sensorer på konsumentmarknaden men flera passande sensorer används i personbilar. Eftersom dessa sensorer sitter inuti bilen är de troligen inte anpassade för att tåla de väderpåfrestningar som en cykel utsätts för. Om inte lämpliga sensorer kan hittas som redan tillverkas i stor skala kan styckpriset per sensor bli högt då volymerna eventuellt blir relativt små.

Systemet skulle mycket fördelaktigt kunna integreras i en elcykel. Det stora batterikapaciteten skulle möjliggöra bättre processor, bättre sensorer och ge möjlighet att visa mer detaljerad data för cyklisten. Många elcyklar har en stor display vilket skulle göra visning av denna data enklare.

För att ett system som det här skulle vara möjligt att sälja till konsumenter i en större utsträckning måste den ha ett relativt lågt försäljningspris. Ett försäljningspris till slutkund har inte beräknats.

Den mest kostnadseffektiva lösningen är att sensorerna integreras i en komponent som redan finns på cykeln exempelvis baklampan.

9

Diskussion och slutsats

De flesta cykelolyckor med svåra eller dödliga skador sker med andra fordon eller cyklister. Det framtagna systemet skulle hjälpa och varna cyklisten. Arbetet att ta fram ett system som det här var mycket mer omfattande än som först uppskattades.

9.0.1 Kretskort

Tjuvström försvårade tillverkningen av prototypkretskortet. Det var vid en kontroll av lödningen som det hittades tjuvström mellan flera lödningar när kretskortet var monterat till Arduinon.

Vi skrapade rent så noggrant vi kunde men det var fortfarande tjuvström någonstans. Tillslut tog vi beslutet att börja om. Den nya kretskortet visade samma tjuvström. Det visade sig att Arduino sluter kretsen och ger ett falskt utslag att det är kontakt mellan banorna.

En lärdom av detta är att i kommande projekt skall inga yttre komponenter vara kopplade till kretsen under felsökning.

9.0.2 Program

När vi började projektet trodde vi att programmeringen skulle vara den enkla delen. Vi insåg när vi skulle lägga till en till sensor att flera datakällor försvårade både hämtning och lagring av data.

Koden skrevs därför om för att enklare kunna skalas beroende på hur många sensorer som används. Den ökade komplexiteten tillsammans med att Arduinos IDE inte har någon funktion för felsökning gjorde kodskrivning mycket mer tidskrävande än vi först trott.

För att kunna felsöka vår kod togs det därför fram program i Xcode som möjliggjorde att vi kunde testa enskilda funktioner var för sig.

9.1 Förslag på vidareutveckling

Om det här systemet skulle göras om till en kommersiell produkt finns det några saker som skulle behöva ändras. Detta för att få ett system som kan tåla att vara utomhus, är pålitligt utan falska varningar och går att massproducera.

9.1.1 Val av processor

En bättre processor rekommenderas då Arduinon är långsam vid större beräkningar. En av typen ARM lik de i mobiltelefoner är billig och kraftfull.

9.1.2 Val av sensor

Ultraludssensorn HC-SR04 som används i det här projektet är inte lämplig att ha i det här användningsområdet. Detta då den är känslig för vatten, har kort mäträckvidd och är ganska långsam. Den är däremot en mycket bra sensor att använda i en prototyp som denna. I en kommersiell produkt bör en radar användas. Detta då den kan se i alla väderförhållanden och ger mycket bra mätdata.

OPS241-A Short Range Radar Sensor kan vara ett bra alternativ i en vidareutvecklad produkt.

Litteraturförteckning

[1] J. Eriksson. Säkerhetseffekten av ökat cyklande. *vti*, 2017. http://vti.diva-portal.org/smash/get/diva2:1150985/FULLTEXT02.pdf,Hämtad2019-01-15, [Läst: 15 Januari 2019].

[2] PA WARRENDALE. Sae international releases updated visual chart for its "levels of driving automation" standard for self-driving vehicles. *sae international*, 2018. https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-"levels-of-driving [Läst: 15 Januari 2019].

[3] S. J. Ramold. Sonar technologies. *Salem Press Encyclopedia of Science*, 2013. http://proxy.lib.chalmers.se/login?url=http://search.ebscohost.com.proxy.lib.chalmers.se/login.aspx?direct=true&db=ers&AN=89250583&lang=sv&site=eds-live&scope=site, [Läst: 20 Oktober 2018].

[4] M. S. Ameigh. Radar. *Salem Press Encyclopedia of Science*, 2013. http://proxy.lib.chalmers.se/login?url=http://search.ebscohost.com.proxy.lib.chalmers.se/login.aspx?direct=true&db=ers&AN=89317181&lang=sv&site=eds-live&scope=site, [Läst: 21 Oktober 2018].

[5] T. Vanessa. Lidar (remote sensing technology). *Salem Press Encyclopedia of Science*, 2016. http://proxy.lib.chalmers.se/login?url=http://search.ebscohost.com.proxy.lib.chalmers.se/login.aspx?direct=true&db=ers&AN=87323881&lang=sv&site=eds-live&scope=site, [Läst: 20 Oktober 2018].

[6] Datablad for ultrasonic ranging module hc - sr04. . https://www.electroschematics.com/wp-content/uploads/2013/07/HCSR04-datasheet-version-1.pdf, [Hämtad: 25 Januari 2019].

[7] Datablad for 8-bit avr microcontroller with 32k bytes in-system programmable flash. 2015. http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf, [Läst: 16 Januari 2019].

[8] Produktsida för arduino uno. . https://store.arduino.cc/arduino-uno-rev3, [Hämtad: 16 Januari 2019].

[9] DBgit. Calculation speed tables. http://forum.arduino.cc/index.php?topic=200585.0, [Hämtad: 16 Oktober 2018].

[10] AlphaBeta. Datatype practices for arduino. https://playground.arduino.cc/Code/DatatypePractices, [Hämtad: 16 Oktober 2018].

[11] Bitbucket sida för biblioteket newping. . https://bitbucket.org/teckel12/arduino-new-ping/wiki/Home, [Hämtad: 10 Oktober 2018].

[12] Aktivt säkerhetssystem för cyklar, video. . https://youtu.be/WBfuPNYJFZo, [Hämtad: 30 januari 2019].

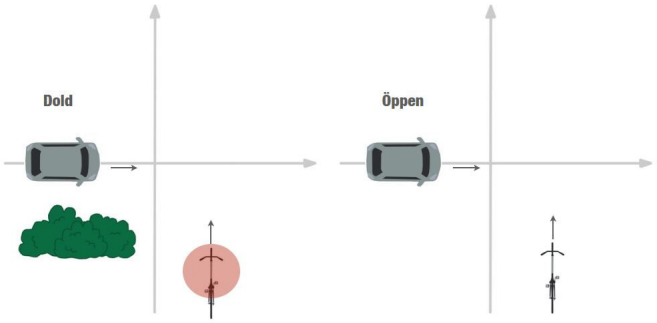
Bilagor

Bilaga 1, Beräkning av exekveringstid

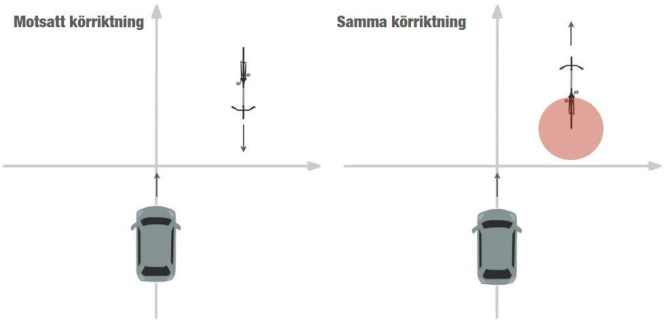
```
const int nb_of_interation_per_pass = 30000;
const int nb_of_pass = 100;

unsigned long initial_time = 0;
unsigned long final_time = 0;

float duration_dummy_loop = 0;
float duration = 0;
float duration_sum = 0;


// All variables used for calculations are declared globally and volatile to minimize
// any possible compiler optimisation when performing the same operation multiple times.

volatile int dummy = 0;

volatile float float_1 = 0;
volatile float float_2 = 0;
volatile float float_3 = 0;

volatile double double_1 = 0;
volatile double double_2 = 0;
volatile double double_3 = 0;

volatile long long_1 = 0;
volatile long long_2 = 0;
volatile long long_3 = 0;

volatile unsigned long ulong_1 = 0;
volatile unsigned long ulong_2 = 0;
volatile unsigned long ulong_3 = 0;

volatile int int_1 = 0;
volatile int int_2 = 0;
volatile int int_3 = 0;

volatile byte byte_1 = 0;
volatile byte byte_2 = 0;
volatile byte byte_3 = 0;

volatile boolean bool_1 = 0;
volatile boolean bool_2 = 0;
volatile boolean bool_3 = 0;

void setup()
{
  Serial.begin (115200);
}
```

```
void loop() {

  for (int j = 0; j < nb_of_pass ; j++) {

      // STEP 1: We first calculate the time taken to run a dummy FOR loop to measure the overhead cause by the execution of the loop.
      initial_time = micros();
      for (int i = 0; i < nb_of_interation_per_pass ; i++)
      {
        // A dummy instruction is introduced here. If not, the compiler is smart enough to just skip the loop entirely...
        dummy++;
      }
      final_time = micros();
      // The average duration of a dummy loop is calculated
      duration_dummy_loop = float(final_time - initial_time)/nb_of_interation_per_pass;
      dummy = 0;


      // STEP 2 (optional): Pick some relevant random numbers to test the command under random conditions.
      //Make sure to pick numbers appropriate for your command (e.g. no negative number for the command "sqrt()")
      randomSeed(micros()*analogRead(0));
      ulong_1 = random(0,256);
      ulong_2 = random(0,256);


      // STEP 3: Calculation of the time taken to run the dummy FOR loop and the command to test.
      initial_time = micros();
      for (int i = 0; i < nb_of_interation_per_pass ; i++)
      {
         // The dummy instruction is also performed here so that we can remove the effect of the dummy FOR loop accurately.
         dummy++;
         // *************** PUT YOUR COMMAND TO TEST HERE ******************
         //double_3 = double_1+double_2; // Target command example

         //long_3 = abs(long_2-long_1);

         ulong_3 = (ulong_2-ulong_1);
         if(ulong_2<ulong_1)
         {
            ulong_3 = -ulong_3;
         }

         //long_3 = (long_1*36)>>11;
         //long_3 = long_1/57;


         // *************** PUT YOUR COMMAND TO TEST HERE ******************
      }
      final_time = micros();

      // STEP 4: Calculation of the time taken to run only the target command.
      duration = float(final_time - initial_time)/nb_of_interation_per_pass - duration_dummy_loop;
      duration_sum += duration;
      dummy = 0;

      Serial.print(j);
      Serial.print(". ");
      print_result(duration);
      }

  Serial.println();
  Serial.println("********* FINAL AVERAGED VALUE ********** ");
  print_result(duration_sum/nb_of_pass);
  Serial.println("****************************************** ");
  Serial.println();
  duration_sum = 0;
  delay(2000);
}

void print_result(float value_to_print)
{
  Serial.print("Time to execute command: ");
  Serial.print("\t");
  Serial.print(value_to_print,3);
  Serial.print(" us");
  Serial.print("\t");
  Serial.print(round(value_to_print*16));
  Serial.println(" cycles");
}
```

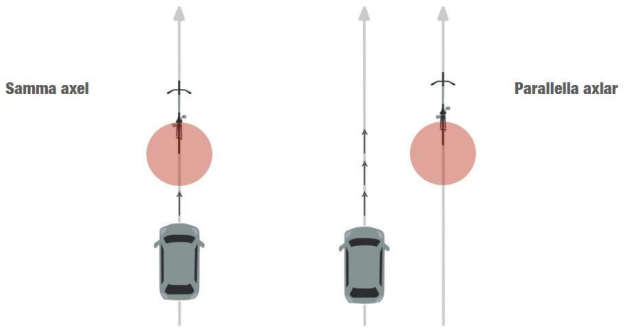
Bilaga 2, Trafiksituationer



Figur 9.1: Korsande möte

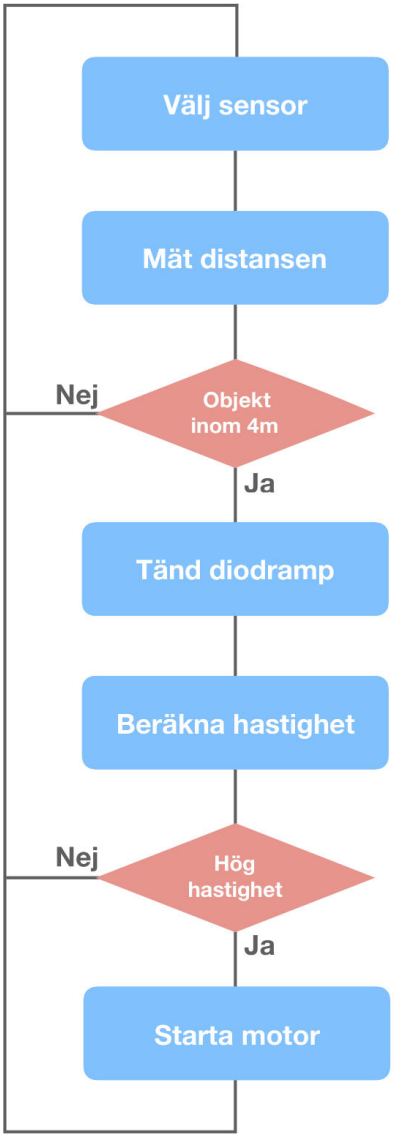


Figur 9.2: Svängande trafik



Figur 9.3: Bakomliggande trafik

Bilaga 3, Flödesschema



Bilaga 4, Kod

```
    breaklines
#include <NewPing.h>
#include <Adafruit_NeoPixel.h>

//DEBUGING
//#define DEBUGTIME
//#define DEBUGPRINT
#define FASTCALC
#define VIDEO
#define SPEEDDEBUG

//SONAR SENSORS
#define SONAR_NUM 3
#define MAX_DISTANCE 4000
#define RIGHTSENSOR 0
#define MIDDLESENSOR 1
#define LEFTSENSOR 2
#define RIGHTWALLSENSOR 3
#define LEFTWALLSENSOR 4

//SAVING DATA
#define QUELENGTH 5

//MOTORS
#define VELOCITYWARNING 6
#define MOTORSPEED 40
#define MOTORTIME 500
#define RIGHTMOTOR 9
#define LEFTMOTOR 10

//LED
#define LEDPIN 11
#define N_LEDS 16
#define BRIGHTNESS 10

//CALCULATIONS
#define TANTOLERANCE 20


//Array for att spara QUELENGTH antal varden fran sensor
unsigned long queArray[SONAR_NUM][QUELENGTH];

//Array med 2 platser, anvands for att berakna delta tiden
unsigned long timeArray[SONAR_NUM][2];

//Array med 2 platser, anvands for att berakna delta distansen
unsigned long medianArray[SONAR_NUM][2];

//Array for att spara och rita ut varden pa LED
unsigned long ledArray[SONAR_NUM][1];


// Sensor object array.
```

```
// Each sensor's trigger pin, echo pin, and max distance to ping.
NewPing sonar[SONAR_NUM] ={
  NewPing(4, 4, MAX_DISTANCE),
  NewPing(5, 5, MAX_DISTANCE),
  NewPing(6, 6, MAX_DISTANCE)
};

Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, LEDPIN, NEO_GRB + NEO_KHZ800);

void setup()
{
   //Staller in LED
   strip.setBrightness(BRIGHTNESS);
   strip.begin();

   //Motorer och LED som outputs
   pinMode(LEFTMOTOR, OUTPUT);
   pinMode(RIGHTMOTOR, OUTPUT);
   pinMode(LEDPIN, OUTPUT);

   #ifdef DEBUGPRINT
      Serial.begin(115200);
   #endif
}

void loop()
{
   unsigned long dist;
   unsigned long median;
   unsigned long deltaDist;
   unsigned long deltaTime;
   unsigned long velocity;
   int negVelocity = 0;
   int toggleSave = 0;

   while(1)
   {
      // put your main code here, to run repeatedly:
      for(int sensor = 0; sensor < SONAR_NUM; sensor++)
      {
          //Mater distans
          dist = measureDist(sensor);
          deltaTime = getDeltaTime(sensor, &toggleSave);
          addToQueue(sensor, dist, &toggleSave);
          median = getMedian(sensor);
          deltaDist = getDeltaDist(sensor, median, &negVelocity, &toggleSave);

          velocity = clacVelocity(deltaDist, deltaTime);
          velWarning(sensor, velocity, &negVelocity);

          //Sparar medianvarden
          ledArray[sensor][0] = median;


          #ifdef DEBUGPRINT
            //Skriver ut alla matta och beraknade varden
```

```
            printFunc(dist, deltaTime, &negVelocity, deltaDist,
                                    median, velocity, &toggleSave);
          #endif
      }

      //uppdaterar led
      updateLed();

      //Inverterar "toggleSave"
      toggleSave = toggleSave^1;
      #ifdef DEBUGPRINT
         Serial.println();
      #endif
   }

}


//Funktion for att skriva ut matta och beraknade varden fran loop()
void printFunc(unsigned long dist, unsigned long deltaTime, int *negVelocity,
                    unsigned long deltaDist, unsigned long median, unsigned long velocity,
                    int *toggleSave)
{
  Serial.print("dist ");
  Serial.print(dist);
  Serial.print("\t");

  #ifdef DEBUGTIME
    Serial.print(deltaTime);
    Serial.print("\t");
  #endif //DEBUGTIME

  #ifndef SPEEDDEBUG
  Serial.print(*negVelocity);
  Serial.print(" ");
  Serial.print(deltaDist);
  Serial.print("\t");
  Serial.print(median);
  Serial.print("\t");
  #endif //SPEEDDEBUG

  Serial.print("Velocity ");
  Serial.print(velocity);
  Serial.print("\t");

  #ifndef SPEEDDEBUG
  Serial.print(*toggleSave);
  Serial.print("\t");
  #endif //SPEEDDEBUG
}


//Mater distansen med hjalp av biblioteket NewPing
unsigned long measureDist(int index)
{
   unsigned long timeSonar = sonar[index].ping();
```

```
   //Division i ATmega328P med longs ar langsamt och darfor anvands en "shift"
   //for att reducera exikveringstiden.
   //Den forkortas fran 660 cykler till 160. Om FASTCALC ar definierat anvands shift,
   //annars anvands den vanliga divisionen.
   #ifndef FASTCALC
      //gor om fran centimeter till millimeter
      timeSonar = timeSonar*10;

      //Returnerar uppmatt langd i mm, delar p 58 for att ljudets hastighet ar 340m/s
      // eller 29 microsekunder per centimeter. Eftersom ljudet fardas fram och tillbaka
      // maste vi dela den uppmata tiden med 2*29 = 58
      return(timeSonar/58);
   #endif //FASTCALC

   //Returnerar uppmatt stracka i millimeter, *360>>11 r en approximation av /58
   // som anvnds da division ar lngsamt med longs i Arduino.
   return ((timeSonar*360)>>11);
}


//Mater tiden och sparar den i Arrayen "timeArray".
//Beraknar sedan skillnaden i tid sen forra kallelsen med hjalp av timeArray
unsigned long getDeltaTime(int index, int *toggleSave)
{
   unsigned long deltaTime;

   //Micros ger tiden i millisekunder sedan prosessorn startades
   unsigned long currentTime = micros()/100;


   //if-funktion for att spara den senaste tiden over den aldsta
   if(*toggleSave == 0)
   {
      //Sparar det nya vardet
      timeArray[index][1] = currentTime;
      deltaTime = currentTime - timeArray[index][0];
   }
   else //(*toggleSave == 1)
   {
      timeArray[index][0] = currentTime;
      deltaTime = currentTime - timeArray[index][1];
   }

   //Returnerar tidsskillnaden
   return(deltaTime);
}


//Lagger till senaste matvardet i array "queArray"
void addToQueue(int sensNumb, unsigned long newVal, int *toggleSave)
{
    static int i;

    //Senate "toggleSave"-vardet
    static int previousToggleSave;
```

```
    //"toggleSave" ar en 1 eller 0 varaibel som skiftar varde nar alla sensorer
    //lasts av i loop.
    //For att spara vardena i ratt kolumn for varje sensor kollar
    //vi ifall "toggleSave" har andrat varde.
    //Om sa ar fallet borjar vi spara de nya vardena i nasta kolumn av queArray"
    if(*toggleSave != previousToggleSave)
    {
        i++;
        previousToggleSave = *toggleSave;
    }

    //Ifall "i" far ett varde storre an "queArray" langs satts den till 0.
    if(i>=QUELENGTH)
    {
        i = 0;
    }

    //nya vardet sparas i "queArray"
    queArray[sensNumb][i] = newVal;
}


//Funktion for att hitta medianvarde av "queArray" for vald sensor
unsigned long getMedian(int sensorIndex)
{
    unsigned long sortedArray[QUELENGTH];


    //Sorterar en kopia av den valda sensorns ko
    for(int i = 0; i < (QUELENGTH); i++)
    {
        sortedArray[i] = queArray[sensorIndex][i];
    }

    for(int i = 0; i < (QUELENGTH-1); i++)
    {
        //Gissar pa att i ar minst
        int mini = i;

        //Kollar om i ar storre an nasta tal j
        for(int j = i+1; j < QUELENGTH; j++)
        {
            if(sortedArray[j] < sortedArray[mini])
            mini = j;
        }

        // byter plats pa i och j
        double temp = sortedArray[i];
        sortedArray[i] = sortedArray[mini];
        sortedArray[mini] = temp;
    }

    //Tar fram och returnerar medianvardet
    unsigned long median = sortedArray[QUELENGTH/2];
    return median;
```

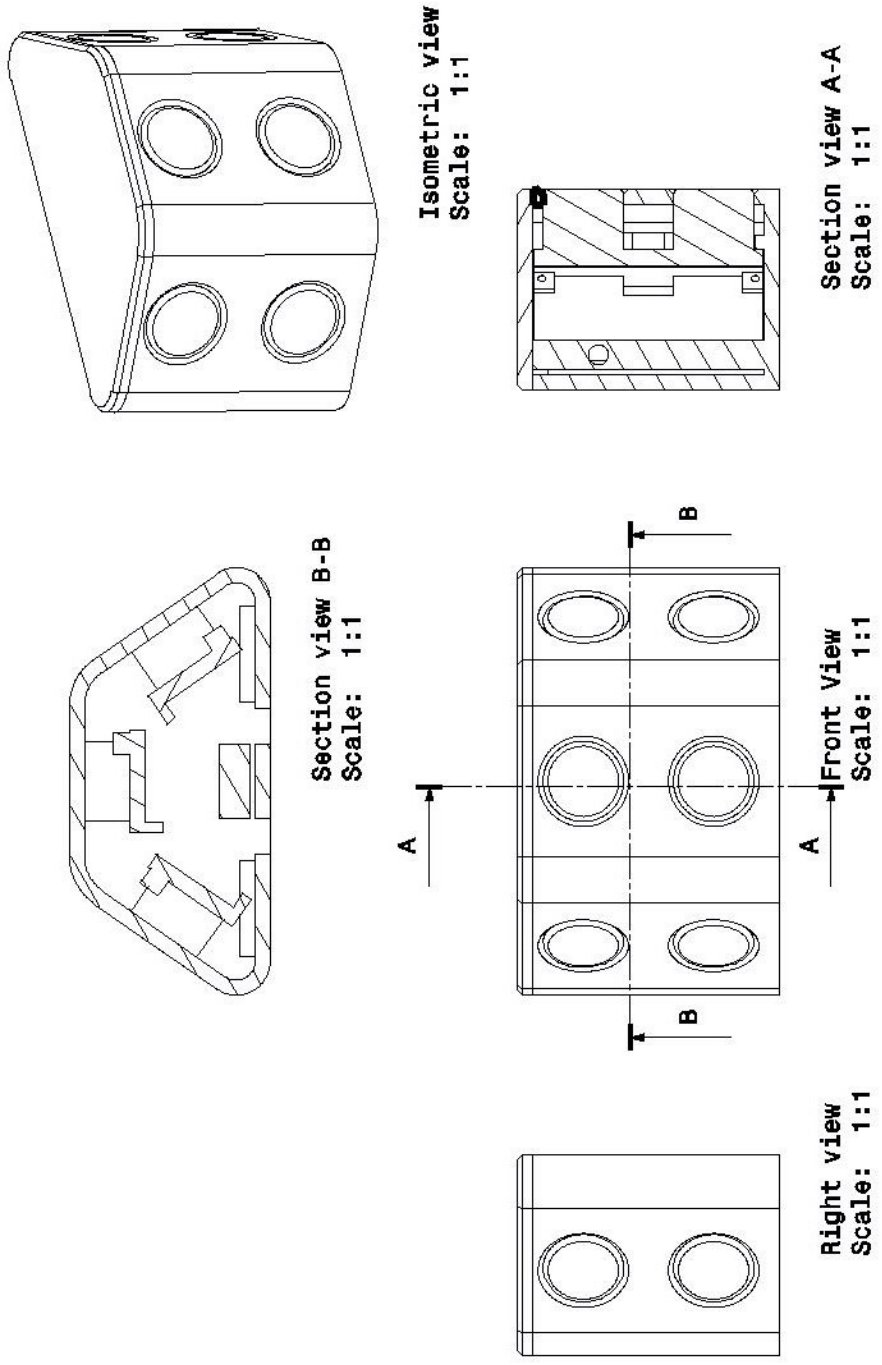
```
}


//Beraknar hastigheten
unsigned long clacVelocity(unsigned long deltaDist, unsigned long deltaTime)
{
    unsigned long velocity;
    deltaDist = deltaDist*10;
    velocity = deltaDist/deltaTime;
    return(velocity);
}


//Beraknar distansskillnaden
unsigned long getDeltaDist(int sensor, unsigned long median, int
                                    *negVelocity, int *toggleSave)
{
    unsigned long deltaDist;


    //if-funktion for att spara den senaste distansen over den aldsta
   if(*toggleSave == 0)
    {
        medianArray[sensor][1] = median;

        //Kontrollerar ifall delta distansen ar negativ eller positiv, 1 = negativ
        if(median>medianArray[sensor][0])
        {
            *negVelocity = 1;
            deltaDist = median - medianArray[sensor][0];
        }
        else
        {
            *negVelocity = 0;
            deltaDist = medianArray[sensor][0] - median;
        }
    }
    else// if(toggleSave == true)
    {
        medianArray[sensor][0] = median;

        if(median>medianArray[sensor][1])
        {
            *negVelocity = 1;
            deltaDist = median - medianArray[sensor][1];
        }
        else
        {
            *negVelocity = 0;
            deltaDist = medianArray[sensor][1] - median;
        }
    }

    //Returnerar delta distansen
    return(deltaDist);
}
```

```
//Kor vibrator
void velWarning(int sensor, unsigned long velocity, int *negVelocity)
{
  static unsigned long time1;
  static unsigned long time2;
  unsigned long currentTime = millis();

  //Om hastigheten ar tillrackligt hog och aker emot oss
  if(velocity > VELOCITYWARNING)
  {
    //Starta motor 0
    if((sensor == RIGHTSENSOR) && (*negVelocity == 0))
    {
      analogWrite(RIGHTMOTOR, MOTORSPEED);

      //Spara tid 3
      time1 = currentTime;
    }

    //Starta bada motorer
    else if(sensor == MIDDLESENSOR)
    {
      analogWrite(LEFTMOTOR, MOTORSPEED);
      analogWrite(RIGHTMOTOR, MOTORSPEED);

      //Spara tid 1
      time1 = currentTime;
      time2 = currentTime;
    }

    //Starta motor 2
    else if((sensor == LEFTSENSOR) && (*negVelocity == 1))
    {
      analogWrite(LEFTMOTOR, MOTORSPEED);

      //Spara tider
      time2 = currentTime;
    }
  }

  //Om tid1-nutid>godtycklig tid
  if((currentTime - time1) > MOTORTIME)
  {
    //stoppa hoger motor
    analogWrite(RIGHTMOTOR, 0);
  }

  //Om tid2-nutid>godtycklig tid
  if((currentTime - time2) > MOTORTIME)
  {
    //stoppa vanster motor
    analogWrite(LEFTMOTOR, 0);
  }
}
```

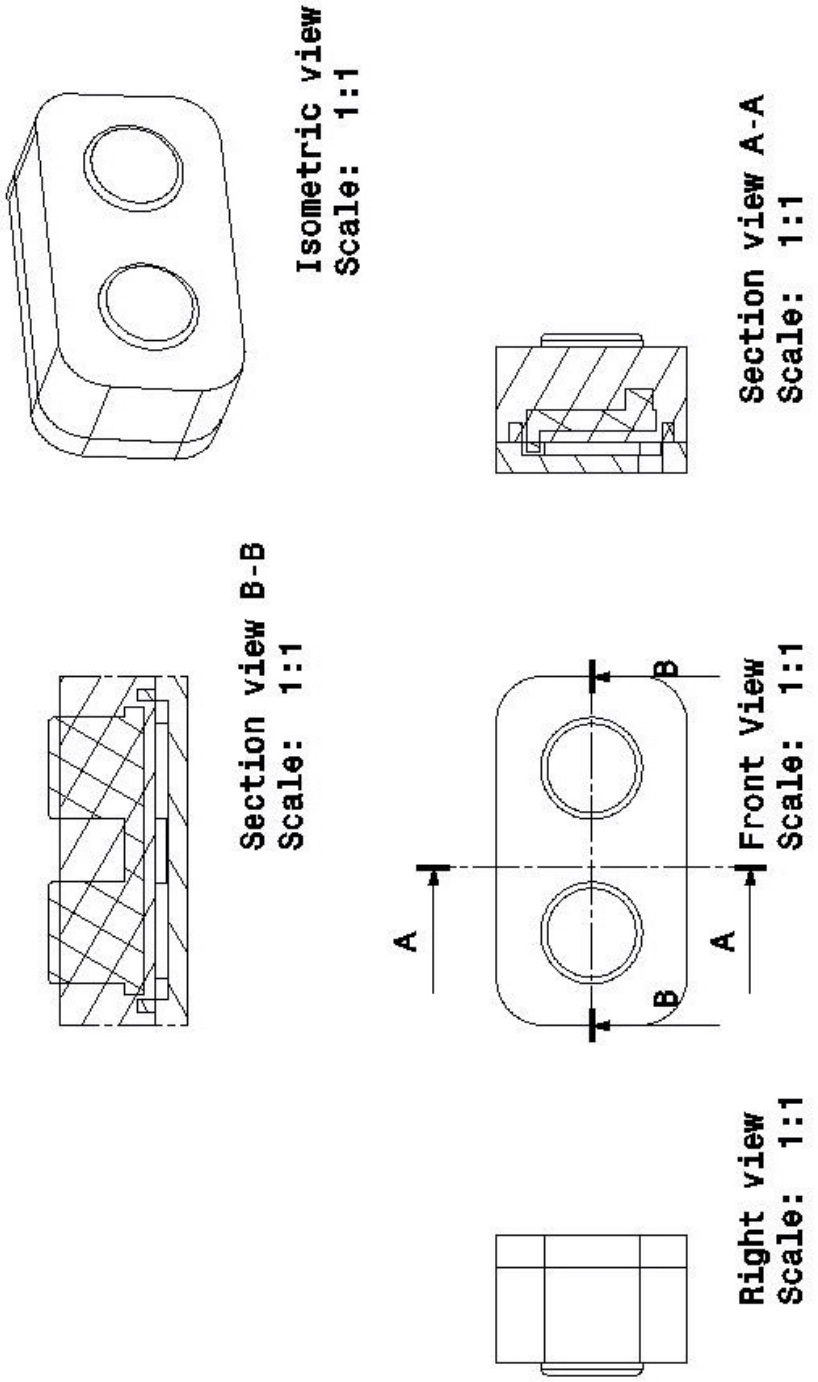
```
//Uppdaterar led
static void updateLed()
{
  strip.clear();

  //Hoger sensor
  warningLed(ledArray[RIGHTSENSOR][0], RIGHTSENSOR);

  //Vanster sensor
  warningLed(ledArray[LEFTSENSOR][0], LEFTSENSOR);

  //Mittensenson
  warningLed(ledArray[MIDDLESENSOR][0], RIGHTSENSOR);
  warningLed(ledArray[MIDDLESENSOR][0], LEFTSENSOR);
}


// Tander Led om en varning har skett
static void warningLed(unsigned long dist, int sensor)
{
    int viewDist = MAX_DISTANCE - dist;
    int j;

    if(sensor < MIDDLESENSOR)
    {
      j = 0;
    }else
    {
      j = 8;
    }

    if(dist>20)
    {
      for(int i=0; i<viewDist; i=i+500)
      {
        if(dist > 100)
        {
          //Rod
          strip.setPixelColor(j , strip.Color(255, 0, 0));
        }else
        {
          //Gul
          strip.setPixelColor(j , strip.Color(255, 110, 1));
        }

        j++;

        if(j>N_LEDS)
        {
          j=0;
        }
      }
    }
   strip.show();
```

```
}


//Funktion for att extrapolation varden
unsigned long extrapolation(unsigned long prevVal, unsigned long *prevTime, float prevSpeed)
{
  unsigned long currentTime  = millis();

  //Tidsskillnaden sedan forra extrapolationen
  unsigned long deltaT = currentTime - *prevTime;
  *prevTime = currentTime;

  unsigned long extrapolationVal = prevVal+(prevSpeed*deltaT);

  return extrapolationVal;

}

unsigned int checkForWall(unsigned long sensor)
{
  int valid = 0;
  //distans mellan sensor och centrum pa mittensensorn
  unsigned long sensorOffset = 0;
  unsigned int sensorAngle = 45;
  unsigned long calculatedValue;
  unsigned long meassuredValue = ledArray[sensor][1];

  //
  if((calculatedValue+TANTOLERANCE>meassuredValue) &&
  (calculatedValue+TANTOLERANCE<meassuredValue))
  {
    valid = 1;
  }

  return valid;
}
```

Bilaga 5, Ritningar av höljen



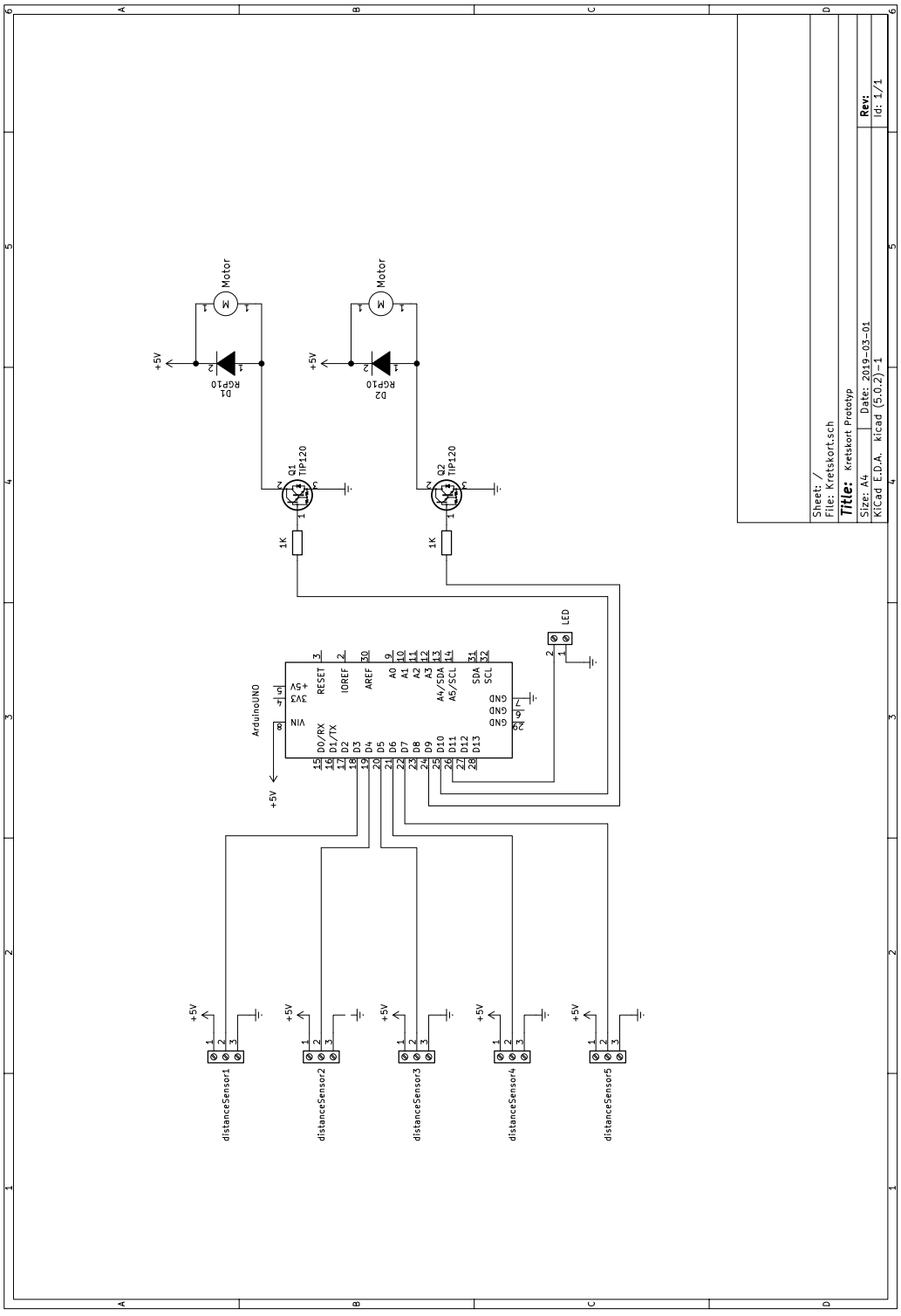
Figur 9.4: Bakre hölje för sonarsensorer


Isometric view
Scale: 1:1
Section view A-A
Scale: 1:1
Section view B-B
Scale: 1:1
Front View
Scale: 1:1
Right view
Scale: 1:1
A
A
B
B

Bilaga 6, Video

https://www.youtube.com/watch?v=WBfuPNYJFZo

Bilaga 7, Kretschema



Bilaga 8, Kretskort