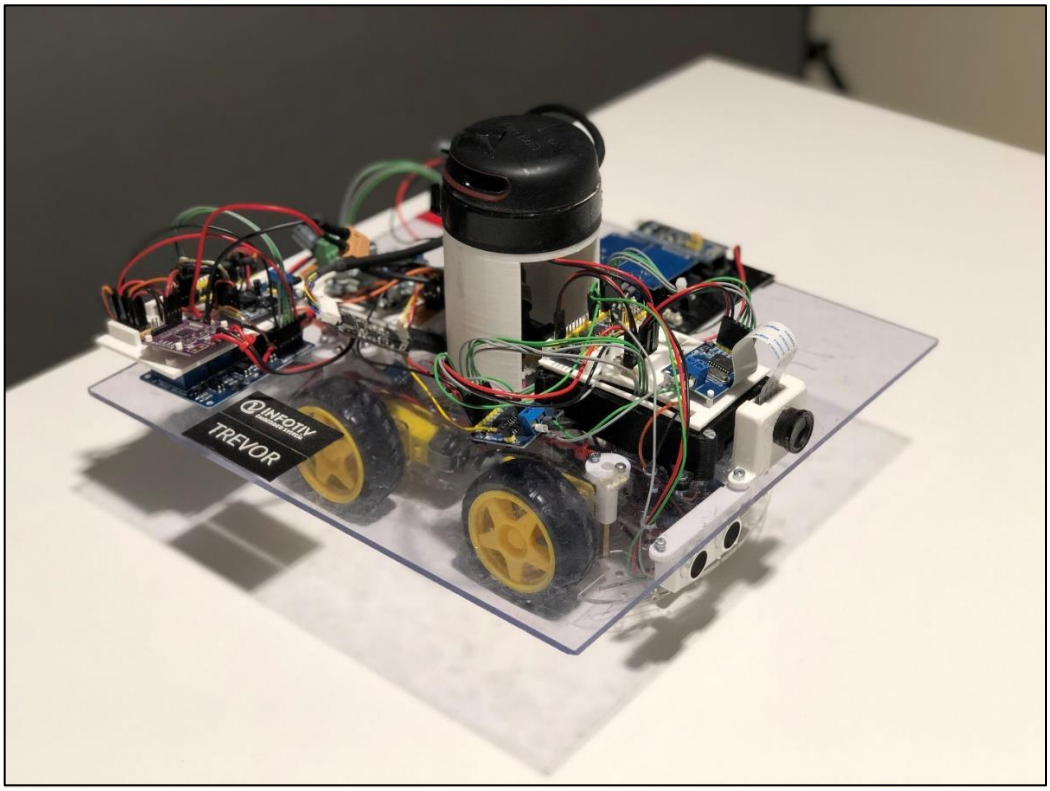
CHALMERS



Implementering av autonoma funktioner för en småskalig fordonsplattform med begränsad hårdvara

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

ADAM ALAMRO
VICTOR AXÉN

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022
www.chalmers.se

Implementing Autonomous Features Using a Small-scale Vehicle Platform with Constrained Hardware

ADAM ALAMRO & VICTOR AXÉN

Department of Computer Science and Engineering
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Sammanfattning

I dagens samhälle går utvecklingen i snabb takt inom många samhällsviktiga branscher, inte minst inom fordonsbranschen. Självkörande bilar blir mer aktuella och efterfrågade, men de är inte tillräckligt trafiksäkra och kan orsaka trafikolyckor. För att göra utveckling av självkörningsfunktionen i autonoma fordon effektivare undersöks i denna rapport implementering av självkörningsfunktioner i mindre prototypfordon som sedan utvärderas. Projektets mål är att implementera en självkörningsfunktion på ett småskaligt fordon med begränsad hårdvara. I arbetet har flera SLAM-paket testats och utvärderats. För att kartlägga omgivningen och lokalisera fordonet i kartan så används en LIDAR-enhet för att samla in data. Denna karta används i sin tur för implementering av självkörningen. Projektets resultat är en fullt fungerande självkörningsfunktion som planerar sin körväg utefter omgivningen.
Slutsatsen av detta är att det går att implementera självkörningsfunktion med relativt enkel hårdvara samt att det finns många framtida möjligheter att undersöka för optimering och utveckling.

Abstract

In today’s society, development is at a rapid pace in many socially important industries, especially in the automotive industry. Self-driving cars are more current and requested, but they are not sufficiently reliable. To make the development of self-driving functions in autonomous vehicles more efficient, in this paper investigations about implementation of self-driving functions in smaller prototype vehicles is explored and then evaluated. The project’s goal is to implement a self-driving function on a small-scale vehicle with limited hardware. In the work, several SLAM-packages have been tested and evaluated. To map the surroundings and locate the vehicle in the map, a LIDAR is used to collect data. This map is then used for the implementation of self-driving. The result of the project is a fully functional self-driving function that plans its route along the surroundings. The conclusion is that it is possible to implement self-driving function with relatively simple hardware and that there are many further opportunities to be investigated for optimization and development.

Innehåll

1 Inledning

Automatisering är ett effektivt sätt att öka kvaliteten och kostnadseffektiviteten i många industrier. Automatisering inom fordonsbranschen handlar om utveckling av bland annat självkörande fordon. Denna teknik har många applikationsområden såsom självkörande lastbilar för transport av varor, självkörande truckar i ett större lager och självkörande traktorer för effektivare jordbruk. Några fördelar med självkörande fordon är mer tidseffektiva transporter, mindre utsläpp av växthusgaser samt säkrare arbetsmiljö i industrier.

1.1 Bakgrund

I takt med samhällets utveckling blir automatisering mer och mer i fokus i många branscher, inte minst inom produktionsanläggningar och i fordonsindustrin. De flesta av dagens nyproducerade fordon har autonoma funktioner implementerade såsom adaptiv farthållare och automatisk inbromsning vid kollisionsrisk. Nuförtiden blir självkörande bilar mer aktuella och efterfrågade, men de är inte tillräckligt pålitliga och vidare utveckling behövs i områden såsom säkerhet och tillförlitlighet. Autonoma funktioner är i allmänhet ekonomisk kostsam samt resurskrävande att implementera direkt i en fordonsproduktion. För att minimera klyftan mellan vad industrin kan utnyttja och vad forskare utvecklar, måste en mer tids-, kostnads- och resurseffektiv lösning implementeras. Ett steg i detta är att undersöka implementering av självkörningsfunktioner i mindre prototypfordon och utvärdera olika lösningar.

Infotiv har utvecklat en småskalig autonom fordonsplattform som kan användas för att testa olika funktioner utan behov av en fullskalig utvecklingsplattform. Fordonet har flera elektroniska kontrollenheter (ECU, Electronic Control Unit) och sensorer, inklusive sonar, LIDAR (Light Detection And Ranging) och kamera. Hårdvarukomponenterna som används i plattformen är dock begränsade beträffande beräkningskraft och sensordataupplösning. I projektet undersöks möjligheterna att implementera en självkörningsfunktion med den befintliga hårdvaran.

1.2 Syfte

Syftet med projektet är att undersöka hur självkörningsfunktioner kan implementeras med hjälp av den befintliga hårdvaran. Detta görs för att se hur bra dessa funktioner kan implementeras i ett fordon på ett kostnads-, tids- och resurseffektivt sätt utan att behöva storskaliga tester. De autonoma funktioner som ska implementeras och testas är kartläggning av testmiljön, lokalisering av fordonet i den framtagna kartan samt självkörning till ett bestämt mål i kartan.

1.3 Mål

Projektets mål är att implementera en självkörningsfunktion med hjälp av data från tillgängliga givare. De implementerade givarna i form av olika sensorer innefattar sonar, LIDAR, och kamera. Fordonet ska kunna köra från en startposition till ett valbart mål och kunna navigera förbi objekt och hinder som är i fordonets körbana. Testmiljön kommer att variera så att olika hinder och objekt placeras slumpvis inom körbanan. Fordonet kommer då att behöva fatta beslut för att välja kortaste körvägen mellan startposition och målposition. Fordonet ska stanna vid risk för kollision och när fordonet har nått sitt mål. Ytterligare mål är att undersöka frågeställningarna:

- Vilken är den bästa metoden för implementering av en självkörningsfunktion med den tillgängliga hårdvaran?
- Går det att optimera beräkningarna så att de blir lika effektiva som i riktiga autonoma fordon?
- Går det att använda olika typer av givare för implementering av samma funktion, men med mindre hårdvara?

I mån av tid ska fordonet även kunna identifiera och reagera på olika trafikskyltar, exempelvis att stanna vid stoppskylt (”skyltidentifieringsfunktion”).

1.4 Avgränsningar

De avgränsningar som gäller för arbetet är följande:

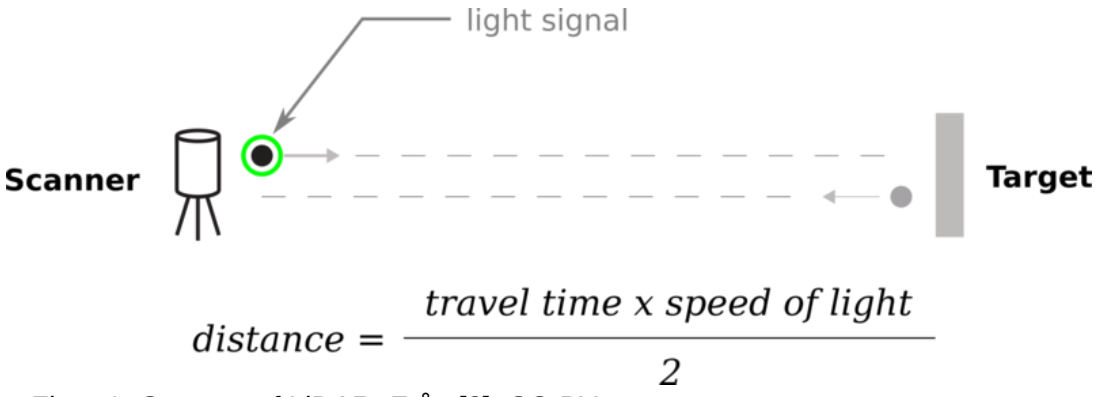
- Fordonets testmiljö är avgränsad till en enkel väg i inomhusmiljö.
- Det ingår inte att ta hänsyn till rörliga objekt eller hinder.
- All hårdvara som behövs för projektet är redan implementerad och är därför inte en del av projektet.
- Projektstart är 2022-01-17.
- Projektslut är 2022-06-05.

2 Teoretisk bakgrund

I projektet ingår flera olika hård- och mjukvaror som LIDAR, Enkortsdator, SLAM (Simultaneous localization and mapping) och ROS (Robot Operating System). Dessa beskrivs mer i detalj i följande avsnitt.

2.1 LIDAR

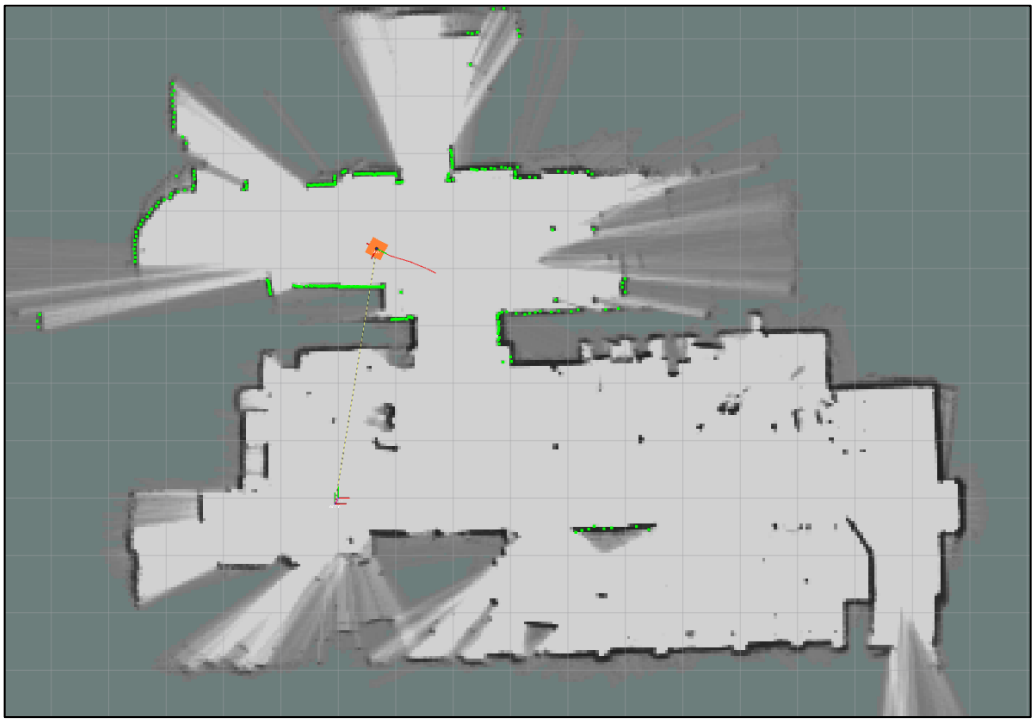
LIDAR är en avståndsmätningsmetod som i första hand använder en laser och lasersensor. Ett LIDAR-system använder principen att mäta ljussignalens restid för att bestämma avstånd, genom att rikta en ljussignal mot ett objekt eller en yta och mäta tiden för det reflekterade ljuset att återvända till mottagaren; detta illustreras i Figur 1. Den kan också användas för att göra digitala 2D- och 3D-representationer av olika miljöer eller ytor genom att variera ljusets våglängd. LIDAR används också i autonoma fordon, där den kombineras med en kamera för att upptäcka och identifiera objekt och föremål runt fordonet. Den är även pålitlig i miljöer där sikten är dålig [1].



Figur 1. Concept of LiDAR. Från [2], CC-BY

2.2 Simultaneous Localization And Mapping

SLAM-metoden används för att bygga en karta av omgivningen och för att lokalisera fordonet i kartan. SLAM-algoritmer tillåter fordonet att kartlägga okända miljöer, exempel visas i Figur 2. Informationen från kartan används i sin tur för att utföra uppgifter som körvägsplanering och undvikande av hinder [3]. SLAM är ett användbart verktyg i många applikationer såsom att navigera robotar för att ordna hyllor i ett lager, att parkera en självkörande bil på en parkeringsplats eller att leverera ett paket genom att navigera med en drönare i en okänd miljö. För att SLAM ska fungera behövs information om miljön kring fordonet. Informationen kan fås från olika typer av sensorer, där en av de mest användbara sensortyperna är LIDAR. Data som kommer in från LIDAR är information i form av koordinater antingen 2D (x, y) eller 3D (x, y, z) och vinklar relativt robotens position. Med en snabb skanningssekvens jämförs de gamla och de nya koordinaterna och kartan uppdateras, och samtidigt lokaliseras kontinuerligt fordonets position i kartan [4].



Figur 2. Karta genererad av Cartographer

2.3 Robot Operating System

ROS är ett open-source ramverk för att skriva programvara till olika typer av robotar. De vanligaste funktioner som ROS används för är meddelandeöverföring av data mellan olika processer och sensorer samt hantering av färdiga algoritmpaket som kan implementeras och användas direkt av roboten [5]. Meddelandeöverföringen görs med ett system som består av noder, meddelanden och topics. En nod är en process som utför någon form av en beräkning; det kan till exempel vara en LIDAR-enhet som beräknar avstånd. Ett system kan bestå av många noder som behöver kommunicera med varandra för att systemet ska fungera. Denna kommunikation kan inte ske mellan noder direkt utan måste ske genom så kallade topics. En topic är en samlingspunkt för meddelanden likt en kommunikationskanal där en nod publicerar data till en topic och de andra noderna som behöver data prenumererar på samma topic. Detta gör att kommunikationen blir effektiv eftersom informationen bara går dit den behövs. För att andra noder ska kunna läsa och förstå meddelandet som skickats behöver det följa en strikt typad datastruktur. Detta protokoll ger ROS en enorm flexibilitet där samma färdiga paket kan användas för många olika typer av robotar [5].

2.4 Enkortsdator i form av en Raspberry Pi

En enkortsdator är en komplett dator uppbyggd av ett enda kretskort. Enkortsdatorn har processor, minne, in-och utgångar samt möjlighet för uppkoppling till internet vilket gör det till en väldigt liten komponent med stor funktionalitet [6]. En Raspberry Pi har många tillämpningar inom många olika användningsområden som allt från inbyggda system i stora industrier till mindre hobbyprojekt.

3 Metod

Projektet grundar sig på traditionell projektmetodik där man arbetar i fasta faser med konkreta fasövergångar. Arbetet börjar med en förstudiefas där det bestäms vad som ska ingå i projektet och vilka delar som behöver göras. Det görs även en grov uppskattning av resursbehov samt kompetensbehov. I denna fas ses programvaror och tillgängliga resurser över för att få en uppfattning om vad som finns tillgängligt. Projektets mål tas fram och diskuteras med alla parter så att alla är eniga om projektets slutmål. Detta bryts sedan ner i en tidsplan i form av ett Ganttschema med tydliga milstolpar och deadlines. Tidsplanen beskriver vad som ska göras i detalj och när det ska göras. Denna tidsplan kommer sedan användas som en referens genom hela arbetet för att stämma av hur projektet ligger till tidsmässigt. Den första tiden av arbetet kommer främst att ägnas åt att undersöka hur SLAM och ROS fungerar samt vilka olika metoder som är etablerade inom dessa områden. Det kommer sedan tas ett beslut om hur dessa delar ska implementeras och hur själva utförandet ska gå till.

Nästa fas är genomförandefasen. Där kommer programmeringen att brytas ner i delar och självkörningen ska implementeras och testas. I denna fas börjar genomförandet av själva projektet där arbetet bygger på förberedelser från förstudiefasen. Det är viktigt att följa tidsplanen som tagits fram och att fokusera på projektets deadlines och milstolpar.

Under avslutningsfasen ägnas tiden åt att sammanställa all nödvändig dokumentation som gjorts under projektets arbetsgång, så att det finns väldokumenterade instruktioner och guider för hur implementering och testning fungerar. Detta är viktigt för fortsatt utveckling av projektet.

4 Genomförande och resultat

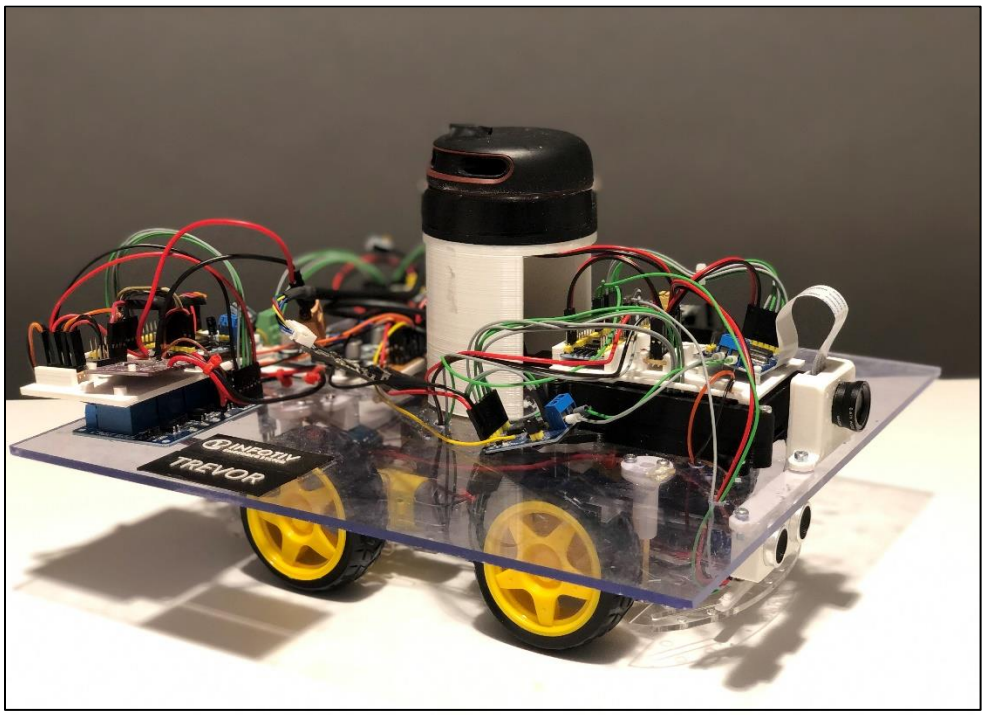
I detta avsnitt beskrivs steg för steg hur genomförandet av projektet har gått tillväga samt det åstadkomna resultatet.

4.1 Förstudiefas

Infotiv har en färdig grund för en fungerande småskalig fordonsplattform där all hårdvara på fordonet redan är implementerad och testad.

Fordonet består huvudsakligen av följande hårdvara som kan ses i Figur 3:

1. Fyra stycken hjul med var sin motor
2. En 360-graders LIDAR monterad i mitten av fordonet. LIDAR-enheten sitter på 150 mm i höjd från golvet
3. Två stycken sonarsensorer, en på baksida respektive framsida av fordonet
4. En kamera på framsida av fordonet
5. En Raspberry Pi som huvuddator



Figur 3. Bild på fordonet

Den implementerade mjukvaran har följande funktioner:

1. Etablera kommunikation med fordonet via Wi-Fi
2. Styra fordonets motorer genom att sätta fordonets linjära hjulhastighet och svänghastighet som representerar att köra framåt respektive att svänga. Detta görs med två Pythonfunktioner: set_wheelspeed och set_turnrate.
3. Få ut data från LIDAR, sonar samt kameran

I projektet används ROS för datakommunikation eftersom detta är ett väletablerat ramverk för mindre plattformar. Det ingår att använda SLAM-teknik för att skapa en karta över omgivningen. Det finns i nuläget en enkel implementering som testar detta men som inte är tillräckligt utvecklad för att kunna användas till någon form av navigation. Någon form av navigation eller självkörningsfunktion var inte påbörjad så det behöver undersökas hur detta kan implementeras.

En tidsplanering har skapats med dessa förutsättningar i åtanke. Den första delen av projekttiden är planerad för att sätta sig in i ROS, SLAM, självkörning och pythonprogrammering innan arbetet med implementering kan påbörjas.

Självkörningsfunktion i detta projekt innebär att fordonet kan navigera till ett givet mål utan interaktion av någon människa. Undersökning av tidigare arbeten inom området självkörning med SLAM och ROS gav en inblick i hur detta kan implementeras [7][8]. I denna undersökning användes ROS för implementering av självkörning för en lastpallsrobot som automatiskt ska kunna flytta lastpallar i ett lager. SLAM användes för kartläggning samt för att lokalisera roboten och detta gjordes i form av ett program kallat Cartographer. Navigation av roboten gjordes med ett programpaket kallat ROS Navigation som skapar en körplan åt roboten. Resultatet var att roboten kunde navigera autonomt till ett bestämt mål [7]. Roboten använde LIDAR samt rotationsgivare vilket skiljer sig mot hårdvaran för fordonet i detta projektet där endast LIDAR är tillgänglig.

Vad det gäller SLAM-tekniken så finns det olika paket som fungerar lite olika beroende på projekt, typ av robot och testmiljö. Noggrannheten hos lokaliseringsfunktionen varierar också beroende på vilken typ av sensor som används för datainsamling. Det finns en studie där en grupp studenter jämför och testar olika SLAM-paket för att avgöra vilket paket som har bäst lokaliseringsprecition. Gruppen jämför ett antal olika SLAM-paket bland andra Cartographer och Gmapping. Alla SLAM-paket testas i samma testmiljö och med samma hårdvara. Resultatet visar att Cartographer och Gmapping har en robust och noggrann lokaliseringsfunktion i en inomhustestmiljö där Cartographer är mindre resurskrävande vad det gäller utrymme i minnet och belastning av processorn i roboten [8].

4.2 Genomförandefas

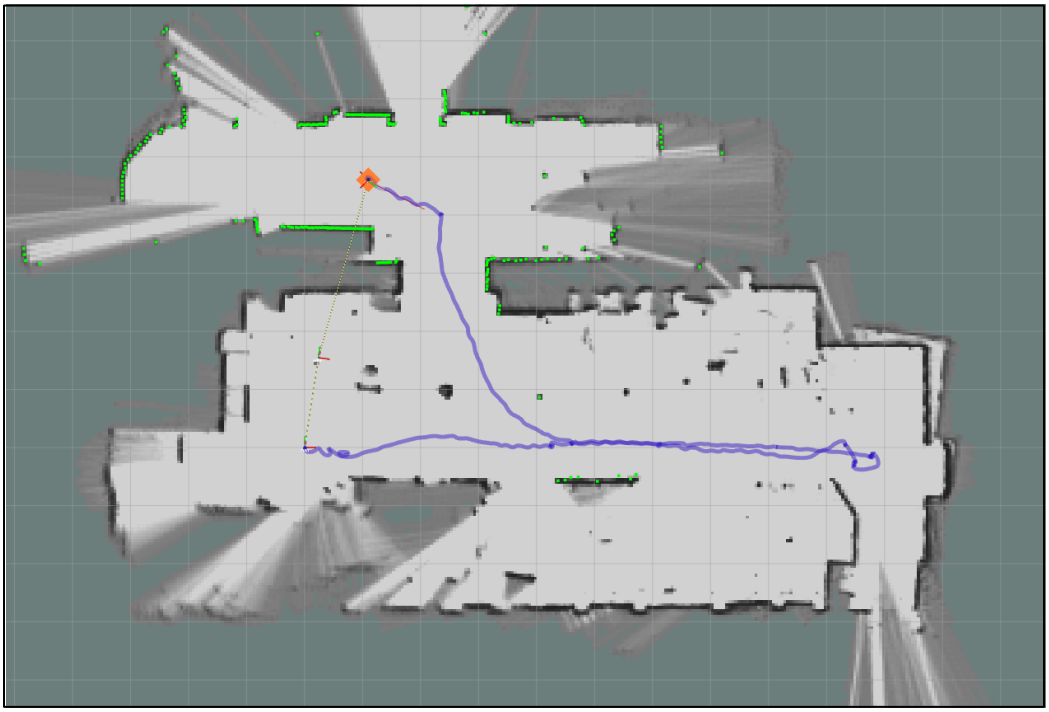
I följande avsnitt presenteras genomförandet av arbetet.

4.2.1 Implementering

I början av detta projekt saknades kunskaper och erfarenheter kring användning av SLAM och ROS. Projektet började därför med att undersöka teorin bakom SLAM och hur tekniken fungerar samt hur den kan implementeras med den tillgängliga hårdvaran. De funktioner som redan implementerats i form av manuellkörning, uppstart av kameran samt kommunikationen mellan fordon och dator testkördes och fungerade väl. Vad det gäller SLAM-tekniken så finns det flera olika SLAM-paket att välja mellan men vissa av dessa lösningar passar inte med den tillgängliga hårdvaran. Detta gjorde att flera olika lösningar har testats och utvärderats för att hitta en fungerande lösning. De SLAM-paket som testats är BreezySLAM [9], HectorSLAM [10] och Cartographer [11]. BreezySLAM fungerar med den befintliga hårdvaran och kan skapa en karta med data från LIDAR men kartan har inte tillräckligt bra upplösning och är inte tillräckligt detaljerad för att kunna användas för navigation av fordonet. HectorSLAM fungerade inte med den befintliga hårdvaran. Cartographer fungerar bra med hårdvaran, kartan som skapas har högre detaljeradnivå än den från BreezySLAM. Catographer har även bra stöd för att implementera navigation eftersom det finns goda möjligheter att anpassa variabler till ett specifikt fordon. Efter testning och utvärdering av de olika paketen bedömdes Cartographer vara lämpligast att användas för kartläggning av omgivningen och lokalisering av fordonet. Eftersom det gick att få bra resultat med Cartographer och på grund av tidsbegränsningen så genomfördes inte vidare testning med Gmapping eller andra SLAM-paket.

Cartographer använder information om fordonets specifikationer och utseende i separata filer. I dessa filer finns information om vilka givare som ska användas för kartläggningen och lokalisering i kartan. Dessa filer innehåller även information om fordonets olika delar samt var de sitter i förhållande till varandra. Fordonet i detta projekt använder bara en LIDAR-enhet som sitter i origo på fordonets basplatta. Med hjälp av detta kan Cartographer ge ut information om fordonets position och riktning i den skapade kartan.

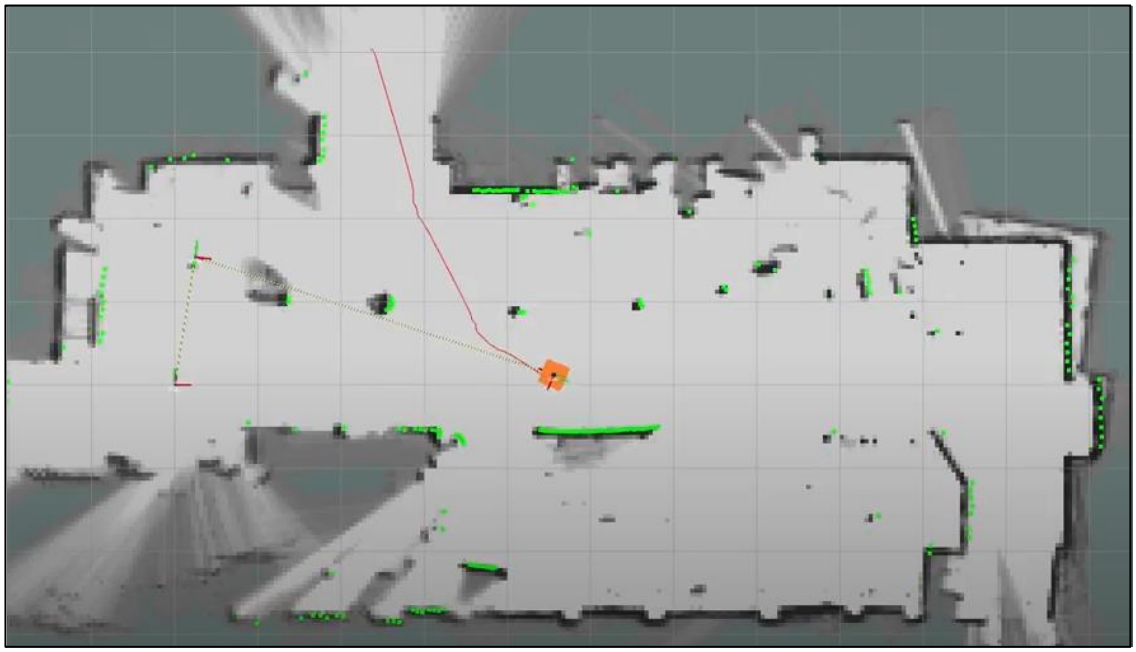
De funktioner som tidigare var implementerade för LIDAR fungerade inte att använda tillsammans med Cartographer. LIDAR och Cartographer är beroende av olika versioner av Ceres Solver som är ett mjukvarupaket för minsta kvadratmetoden och som används till lokalisering av fordonet i kartan [12]. Detta gjorde att en annan lösning var nödvändig. En ny implementering av funktioner för LIDAR behövdes och detta gjordes i form av ett paket som heter rplidar, som gör att det går att starta sensorn i en separat nod och sedan skicka LIDAR data till Cartographer [13]. Ett nytt problem uppstod i form av att rplidar inte fungerade med den uppkopplingen som används mellan fordonet och datorn. Lösningen blev att köra programmen rplidar och cartographer direkt på fordonets Raspberry Pi istället för på datorn. För detta krävdes en ominstallation av operativsystemet på fordonets Raspberry Pi vilket orsakade problem med CAN-bussen på fordonet då den behövde omkonfigureras. Detta gjorde att motorerna inte kunde få någon styrsignal och därmed inte fungerade. Det blev en flaskhals i projektet då omkonfigurationen av CAN-bussen krävde speciell kompetens och det dröjde några veckor innan problemet kunde lösas av en kollega på Infotiv. Den nya implementeringen av rplidar och Cartographer fungerade bra och den skapade kartan som uppdateras i realtid kan illustreras i ett visualiseringsprogram kallat Rviz, vilket presenteras i Figur 4.



Figur 4. Karta uppbyggd i Rviz med positionsuppdatering av fordonet.

Rviz är ett visualiseringsverktyg som används för att visualisera robotar, sensordata och robotens arbetsmiljö [14]. Verktyget är mycket flexibelt och kan anpassas till specifika projekt. I detta fall visualiseras sensordata från fordonets LIDAR, kartan som byggs upp av Cartographer samt fordonets estimerade position i kartan. Programmet kan även användas som simuleringsverktyg.

Med hjälp av den skapade kartan ska nu en självkörningsfunktion implementeras. ROS har ett välutvecklat paket för detta, kallat ROS Navigation [15]. Navigation-paketet tar in information om fordonet och dess omgivning samt ett destinationsmål för att sedan skapa en körplan åt fordonet som presenteras av den röda linjen i Figur 5. Den information som Navigation-paketet behöver är den skapade kartan, data från LIDAR, information om fordonets storlek och utseende, fordonets position, riktning och hastighet samt ett destinationsmål som kan sättas i Rviz. Den körplan som skapas är i form av en global körplan och en lokal körplan där den globala körplanen visar den kortaste vägen från fordonet till målet utan att kollidera med upptäckta hinder. Den lokala körplanen utgår från fordonets position och hastighet och uppdateras kontinuerligt för att följa den globala körplanen så bra som möjligt. Konfigurationsfiler behövde skapas för att dessa körplaner ska bli så exakta som möjligt där information om fordonets storlek, maximala svänghastigheter samt hur stort säkerhetsavstånd till objekt som ska finnas.



Figur 5. Körplan skapada av ROS Navigation

Information som behövs av Navigation-paketet (från Cartographer) samt från fordonets LIDAR och ett destinationsmål kan sättas på valfritt ställe inom den uppbyggda kartan. Med informationen skapas en global körplan och en lokal körplan för fordonet där den lokala körplanen uppdateras konstant utifrån ändringar i omgivningen för att kunna följa den globala planen utan kollision med andra objekt. Navigation-paketet skickar sedan ut körkommandon i form av hjulhastighet och svänghastighet som fordonets ska ha för att kunna följa körplanen. Dessa körkommandon skickas sedan vidare till fordonets motorer, vilket gör fordonet självkörande.

4.2.2 Testfas enkel miljö

Testfasen inleddes med ett enkelt test i en enkel testmiljö inomhus utan hinder på körbanan. Fordonet ska köra från given startposition mot givet mål i en rak sträcka på tre meter. Fordonet klarade testet utan några problem. Nästa steg blev att få fordonet att köra från start mot ett mål utan hinder där fordonet behöver svänga för att komma fram till målet. Fordonet klarar testet och kommer fram till målet men det behövdes små justeringar av säkerhetsmarginalen till objekt för att optimera självkörningen.

4.2.3 Testfas avancerad miljö

Efter att fordonet klarade de enkla testerna blev det dags för att testa mer komplicerad miljö där fordonet ska köra från start mot ett mål och undvika ett hinder som ligger inom körbanan. På det testet så dök det upp några problem, fordonet kunde upptäcka hindret men klarade inte av att köra runt det. Problemet var att fordonet fick körkommando att endast svänga medans hårdvaran kräver ett körkommando att köra framåt samtidigt som den svänger.

Problemet löstes genom att hårdkoda så att fordonet får ett körkommando att köra framåt även när fordonet endast får signal om att svänga. Hårdkodningen i sin tur orsakade nya problem, fordonet kan inte stanna när den har nått målet på grund av att körkommandot fås även när fordonet har nått målet. Detta på grund av att fordonet vill nå sitt mål både gällande rätt position och rätt riktning. Lösningen för detta blev att öka toleransen för accepterad målvinkel så att fordonet accepterar en större målvinkel och därav stannar vid målet.

Ett annat problem är att fordonet ibland kör för nära objekt och hinder trots de inställda säkerhetsmarginalerna, detta beror på att fordonet har svårt att svänga vänster. Det problemet beror på en hårdvarudefekt som innebär att motorerna på högersidan är svagare än motorerna på vänstersidan, detta leder även till att fordonet behöver utföra vänstersväng på en större bana för klara av det. För att lösa detta på ett tidseffektivt sätt så skickas större kommandovärden till motorerna när fordonet ska svänga vänster.

Eftersom det används data från endast en LIDAR-enhet i projektet, för att skanna omgivningen och upptäcka hinder, så kan fordonet inte upptäcka små objekt som ligger på golvet. Alla objekt som är lägre än LIDAR:n, det vill säga alla objekt som är lägre än 150 mm syns inte av fordonet.

Efter mycket testning och optimeringsförsök och med tanke på den hårdvaran som användes i det projektet så är självkörningen fullt fungerande.

4.3 Resultat

Arbetet har resulterat i att fordonet kan producera en karta över sin omgivning med hjälp av sensordata från fordonets LIDAR. Uppbyggnaden av kartan görs med Cartographer-paketet som även lokaliserar fordonet i den uppbyggda kartan. Kartan och fordonets position uppdateras i realtid med hjälp av kontinuerlig data från fordonets LIDAR. Navigation-paketet gör sedan att en målposition kan sättas i den uppbyggda kartan och en planerad körväg skapas mellan fordonet och målpositionen. Den hjul- och svänghastighet som krävs för att fordonet ska kunna följa körvägen skickas ut till en ROS-topic och omvandlas sedan till körkommandon för fordonet. Resultatet av detta är att fordonet är helt självkörande och planerar sin körväg runt de hinder och objekt som är inom fordonets körbana. Fordonet felbedömer ibland sin position och åker närmre objekt än vad den inställda säkerhetsmarginalen tillåter. Detta beror på att fordonet inte är helt stabilt vid körning vilket leder till brus i LIDAR-data som i sin tur stör lokalisering av fordonet. Se resonemang om detta i avsnitt 5.

5 Diskussion

I följande avsnitt diskuteras resultatet av arbetet.

5.1.1 Diskussion resultat

I starten av projekt var förväntningarna att någon enklare form av självkörning med endast undvikande av hinder skulle kunna gå att implementera. Det fanns inga tankar på att implementera någon form av vägplanering eller att sätta ut en målposition grafiskt. Planen var istället att använda data från fordonets LIDAR för att koda hur fordonet skulle bete sig när det närmar sig ett hinder. I förhållande till detta är resultatet mycket bra då fordonet nu självt planerar hur det ska åka för att undvika hinder samt att det automatiskt uppdaterar körvägen ifall fordonet skulle avvika från den först planerade vägen.

5.1.2 Felbedömning av fordonets position

Andra liknande projekt har använt sig av både rotationsgivare och IMU (Inertial Measurement Unit) vilket inte användes i detta projekt [7]. Konstruktionen av fordonet är lite skakig eftersom LIDAR-enheten sitter relativt högt upp och roterar vid skanning, fordonet har en relativt liten massa vilket leder till obalans i fordonet. På grund av detta felbedömer fordonet ibland sin position och åker närmre objekt än vad den inställda säkerhetsmarginalen tillåter. Felbedömningen av fordonets position beror på att data från LIDAR blir brusigt när fordonet är i rörelse. Inga åtgärder har gjorts för att försöka lösa problemet för att en omkonstruktion av fordonet hade behövts, detta gjordes ej på grund av tidsbrist. Upplösningen på kartan är tillräcklig hög och objekt i kartan representerar verkligheten med god precision.

5.1.3 Problem vid måldestination

Måldestinationen består både av en position och en vinkel som fordonet strävar efter att nå. När fordonet har nått sitt mål i position skickas det ut en svänghastighet för att nå rätt målvinkel. Ett av de större problem som identifierats är att hårdvaran gör att fordonet behöver både en linjär hastighet och svänghastighet för att kunna svänga, på samma sätt som att en verklig bil inte kan stå stilla och svänga. Navigation-paketet skickar ofta endast ut en svänghastighet vilket fordonet inte kan utföra. Lösningen att hårdkoda så att fordonet alltid får en hastighet framåt när det ska svänga leder till problem vid måldestinationen. Hårdkodningen gör då att fordonet även åker framåt och passerar målets position. Fordonet kommer då försöka vända och nå målpositionen på nytt och köra runt i cirklar. Det tog tid att hitta en lösning för detta men till slut hittades lösningen med toleransökningen av målvinkeln.

5.2 Frågeställningar för projektet

I följande avsnitt diskuteras frågeställningarna för arbetet.

5.2.1 Implementeringsmetod

Vilken är den bästa metoden för implementering av en självkörningsfunktion med den tillgängliga hårdvaran?

I projektet har endast fordonets LIDAR används för att samla data om omgivningen och implementering av en självkörningsfunktion. En stor del av projektet var att testa olika lösningar för SLAM. Detta gjordes genom att testa att implementera olika SLAM-paket och utvärdera resultatet. De paket som testats är BreezySlam, HectorSLAM och Cartographer. I kombination med ROS-ramverket som används i projektet var Cartographer den bästa lösningen då den har mest detaljerad karta och noggrannast lokalisering av fordonets position. ROS Navigation-paketet som används för körplanering är det mest utvecklade system för ROS och det fanns inga andra likvärdiga alternativ.

5.2.2 Optimering av beräkningar

Går det att optimera beräkningarna så att de blir lika effektiva som i riktiga autonoma fordon?

Utifrån den hårdvaran och mjukvaran som används så blir inte självkörningen optimal eller felfri. Det finns en liten risk att fordonet misslyckas att undvika hinder och avviker ibland från den planerade körbanan eftersom att fordonets positions estimering inte är helt optimal. Missbedömningen i positioneringen beror på brus i data från fordonets LIDAR vid körning. Bruset kan ses genom att olika avstånd skickas från LIDAR-enheten även fast fordonet står stilla. Detta gör att det inte är lika effektivt som ett riktigt autonomt fordon.

5.2.3 Alternativ hårdvara

Kan man använda olika typer av givare för implementering av samma funktion, men med mindre hårdvara?

Det går att kombinera ROS-ramverket med olika SLAM-paket och implementera en självkörningsfunktion på fordonet. I detta projekt har endast LIDAR använts men det går också att använda olika typer av givare eller sensorer för att implementera självkörningen, exempelvis sonar och kamera, men detta kräver en del kodmodifiering och hårdvarukonfigurering. På grund av tidsbrist undersöktes inte skyltidentifieringsfuntion med kameran men detta skulle kunna implementers i framtiden.

5.3 Vidareutveckling och förbättringsförslag

Nästa steg i projektet kan vara att implementera sonar och kamera med den befintliga självkörningen för att optimera resultatet och få bättre precision i körningen. Det kan även undersökas om en rotationsgivare som kan hjälpa till att förbättra självkörningen.

Vid implementering av sonar och kamera kan andra SLAM-paket testas och undersöka om resultatet blir bättre än med det befintliga SLAM-paketet.

För att få ett bättre resultat gällande lokalisering skulle en omkonstruktion av fordonet kunna göras. Fordonet behöver en stabilare konstruktion och mer tillförlitliga och välstyrda motorer. Detta för att få en mjukare körning och undvika brusiga LIDAR-data.

5.4 Samhälleliga, etiska och ekologiska aspekter

En småskalig plattform gör det möjligt att utföra mer testning och förbättringar än vad som skulle vara möjligt med fullskaliga fordon. Detta gör att autonoma fordon kan bli effektivare, säkrare och billigare. Projektet har utförts inomhus i en kontrollerad miljö på ett småskaligt prototypfordon och därför läggs inte så stort fokus på samhälleliga, etiska och ekologiska aspekter. Vid övergång till fullskaliga fordon så behöver man däremot ta hänsyn till dessa aspekter.

5.4.1 Samhälleliga aspekten

När det gäller den samhälleliga aspekten måste man ta hänsyn till cybersäkerhet och driftsäkerhet så att fordonet inte kan bli utsatt för bland annat cyberattacker [16]. Detta skulle kunna få stora konsekvenser, som att någon kan ta över styrningen av fordonet. En positiv effekt på samhället är att med en ökad användning av självkörande fordon så ökar också viljan att köra längre sträckor till jobb eller annat eftersom man inte behöver köra själv. Medan bilen kör till jobbet så kan man sitta och jobba eller bara vila. Självkörande fordon i trafiken minskar restiden genom att bilen eller bussen väljer kortast väg till destinationen och slipper onödiga inbromsningar och stopp och därmed bidrar till ett bättre flyt i trafiken [17].

5.4.2 Etiska aspekten

En annan viktig och intressant aspekt är den etiska, där frågan handlar om vem som bär ansvaret ifall det självkörande fordonet blir inblandat i en olycka. I ett helt autonomt fordon är det mycket som kan gå fel både när det gäller hårdvara och mjukvara. Detta kan leda till olyckor som är utom förarens kontroll. Allt detta måste tas med i en analys när man designar ett autonomt fordon. Det är en komplex fråga att besvara – är det föraren som sitter i fordonet, ingenjören som har programmerat fordonet eller företaget som har tillverkat fordonet som bär ansvaret vid en olycka [18]? En annan fråga är hur fordonet ska programmeras att prioritera vid en olycka, är det passageraren som ska prioriteras eller exempelvis cyklisten som dyker upp framför fordonet. Ska fordonet köra på cyklisten för att skydda passageraren eller köra på ett hinder och riskera att skada passageraren för att undvika att skada cyklisten?

Det är väldigt fascinerande med självkörande fordon och alla dess fördelar men vad det gäller den etiska aspekten så är det väldigt komplicerat och ett svårt dilemma för alla intressenter. Med de frågorna i åtanke så är det ett svårt val för en kund att välja att äga en självkörande bil med tanke på riskerna. Det är svårt för försäkringsbolag att hantera sådana frågor på grund av osäkerheten kring vem som bär ansvaret vid en trafikolycka [18]. Samhället har inte heller en enig uppfattning om hur självkörande fordon ska integreras i trafiken på ett säkert sätt. Utvecklingen av självkörande fordon är väldigt viktig och arbetet pågår med full fart, särskilt vad det gäller säkerhetssystemen. Fordonen ska vara säkra för kunden och andra medtrafikanter så att samhället kan gynnas av de signifikanta fördelarna med självkörande fordon på ett samhällssäkert sätt.

5.4.3 Ekologiska aspekten

Fordonsbranschen har en stor påverkan på den ökade globala uppvärmningen. Personbilar står för ca 60% av växthusgasutsläppen inom transportsektorn vilket gör att autonoma fordon kan bidra till en stor förbättring gällande minskat utsläpp [17]. I teorin kan autonoma fordon minska utsläppen genom att planera sin körning mycket bättre än en människa och därigenom minska onödiga inbromsningar och stopp. I praktiken fungerar inte detta lika väl för att många okända variabler tillkommer. Enligt den forskning som har gjorts så visar det sig att om det bara handlar om några enstaka autonoma fordon i trafiken så minskar inte utsläppen alls och de kan till och med öka. Detta beror på att mänskliga förare har svårt att anpassa sig till de autonoma fordonen och kör därför mindre effektivt. Om man däremot ökar andelen autonoma fordon i trafiken så blir resultaten bättre och utsläppen börjar att minska. Bäst resultat fås vid 100% autonoma fordon eftersom att de kan anpassa sig till varandra på ett bättre sätt och därigenom fås ett bra flyt i trafiken [17]. I nuläget är självkörningstekniken relativt ny och utvecklingen av autonoma fordon har bara startat men ju längre forskningen kommer desto miljövänligare kommer autonoma fordon att bli.

6 Slutsatser

Målet med projektet var att undersöka möjligheten att implementera en självkörningsfunktion med en begränsad hårdvara inom en enkel testmiljö. Projektets slutsats är att det definitivt är möjligt att implementera självkörningsfunktionalitet med mycket enkel hårdvara och enstaka sensorer. Det resultat som åstadkommits är en självkörningsfunktion som kan navigera fordonet från en startposition till ett mål samt navigera förbi hinder och objekt i fordonets körbana. Det finns många utvecklingsmöjligheter att undersöka samt alternativa lösningar som kan vidarearbetas för ett bättre resultat.

7 Källförteckning

[1] J. D. Choi and M. Y. Kim, “A Sensor Fusion System with Thermal Infrared Camera and LiDAR for Autonomous Vehicles: Its Calibration and Application,” *International Conference on Ubiquitous and Future Networks, ICUFN*, vol. 2021-August, pp. 361–365, Aug. 2021, doi: 10.1109/ICUFN49451.2021.9528609.

[2] “File:Concept of LiDAR.png - Wikimedia Commons.” https://commons.wikimedia.org/wiki/File:Concept_of_LiDAR.png (accessed May 11, 2022).

[3] “What Is SLAM (Simultaneous Localization and Mapping) – MATLAB & Simulink - MATLAB & Simulink.” https://se.mathworks.com/discovery/slam.html (accessed Mar. 21, 2022).

[4] “A Comparative Analysis of LiDAR SLAM-Based Indoor Navigation for Autonomous Vehicles | IEEE Journals & Magazine | IEEE Xplore.” https://ieeexplore.ieee.org/document/9381521 (accessed May 03, 2022).

[5] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” *lars.mec.ua.pt*, Accessed: May 05, 2022. [Online]. Available: http://lars.mec.ua.pt/public/LAR%20Projects/BinPicking/2016_RodrigoSalgueiro/LIB/ROS/icraoss09-ROS.pdf

[6] “Raspberry Pi.” https://www.raspberrypi.com/ (accessed Apr. 05, 2022).

[7] U. G. Lee, K. J. Choi, and S. Y. Park, “The Design and Implementation of Autonomous Driving Pallet Robot System using ROS,” *International Conference on Ubiquitous and Future Networks, ICUFN*, vol. 2021-August, pp. 372–374, Aug. 2021, doi: 10.1109/ICUFN49451.2021.9528735.

[8] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, “A Comparative Analysis of LiDAR SLAM-Based Indoor Navigation for Autonomous Vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, 2021, doi: 10.1109/TITS.2021.3063477.

[9] “GitHub - simondlevy/BreezySLAM: Simple, efficient, open-source package for Simultaneous Localization and Mapping.” https://github.com/simondlevy/BreezySLAM (accessed May 19, 2022).

[10] “hector_slam - ROS Wiki.” http://wiki.ros.org/hector_slam (accessed May 19, 2022).

[11] “Cartographer ROS Integration — Cartographer ROS documentation.” https://google-cartographer-ros.readthedocs.io/en/latest/ (accessed May 19, 2022).

[12] “Installation — Ceres Solver.” http://ceres-solver.org/installation.html (accessed May 19, 2022).

[13] “rplidar - ROS Wiki.” http://wiki.ros.org/rplidar (accessed May 19, 2022).

[14] “rviz - ROS Wiki.” http://wiki.ros.org/rviz (accessed May 19, 2022).

[15] “navigation - ROS Wiki.” http://wiki.ros.org/navigation (accessed May 19, 2022).

[16] “Rapport visar hur uppkopplade bilar är sårbara för cyberangrepp | Trend Micro Sweden.” https://www.mynewsdesk.com/se/trend-micro-sweden/pressreleases/rapport-visar-hur-uppkopplade-bilar-aer-saarbara-foer-cyberangrepp-3073793 (accessed May 19, 2022).

[17] Ó. Silva, R. Cordera, E. González-González, and S. Nogués, “Environmental impacts of autonomous vehicles: A review of the scientific literature,” *Science of The Total Environment*, vol. 830, p. 154615, Jul. 2022, doi: 10.1016/J.SCITOTENV.2022.154615.

[18] “Vem ansvarar för en olycka med en självkörande bil?” https://lawline.se/answers/vem-ansvarar-for-en-olycka-med-en-sjalvkorande-bil (accessed May 11, 2022).

INSTITUTIONEN FÖR DATA- OCH INFORMATIONSTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg, Sverige 2022
www.chalmers.se