

Designspecifikation

Projektgrupp 13

1 november 2022

Version 1.0



Status

Granskad	Hannes Nörager	2022-11-01
Godkänd		

Projektidentitet

Grupp E-post: TSEA29_2022HT_E7-Grupp13@groups.liu.se

Hemsida: <https://gitlab.liu.se/da-proj/microcomputer-project-laboratory-d/2022/g13>

Beställare: Anders Nilsson, ISY, Linköpings universitet
Tfn: 013-28 26 35
E-post: anders.p.nilsson@liu.se

Kund: Anders Nilsson, ISY, Linköpings universitet
Tfn: 013-28 26 35
E-post: anders.p.nilsson@liu.se

Handledare: Peter Johansson
Tfn: 013-28 1345
E-post: peter.a.johansson@liu.se

Kursansvarig: Anders Nilsson, ISY, Linköpings universitet
Tfn: 013-28 26 35
E-post: anders.p.nilsson@liu.se

Projektdeltagare

Namn	Ansvar	Telefon	E-post
Linus Thorsell	Projektledare	0765612171	linth181@student.liu.se
Oscar Sandell	Testansvarig	0709416866	oscsa604@student.liu.se
Hannes Nörager	Utvecklare	0733118779	hanno696@student.liu.se
Johan Klasén	Dokumentansvarig	0730982555	johkl473@student.liu.se
Zackarias Wadströmer	Utvecklare	0706142029	zacwa923@student.liu.se
Thomas Pilotti Wiger	Konstruktionsansvarig	0761708593	thopi836@student.liu.se

INNEHÅLL

1	Inledning	1
2	Översikt av systemet	1
3	Delsystem 1: Kommunikationsmodul	2
3.1	Sensor, Kamera	4
3.2	Navigatör	4
3.3	Datorseende	4
4	Delsystem 2: Styrmodul	6
5	Delsystem 3: Sensormodul	7
5.1	Ultraljudssensor	7
5.2	Halleffektssensor	7
5.3	Data via UART	8
6	Delsystem 4: Extern Applikation	9
7	Kommunikation mellan modulerna	9
7.1	Modulerna på taxin	9
7.2	Den Externa applikationen	10
8	Komponentlista	10
9	Implementeringsstrategi	10
9.1	Konstruktionsmetodik	10
9.2	Programmeringsspråk	10
9.3	Kontinuerlig Testning	10

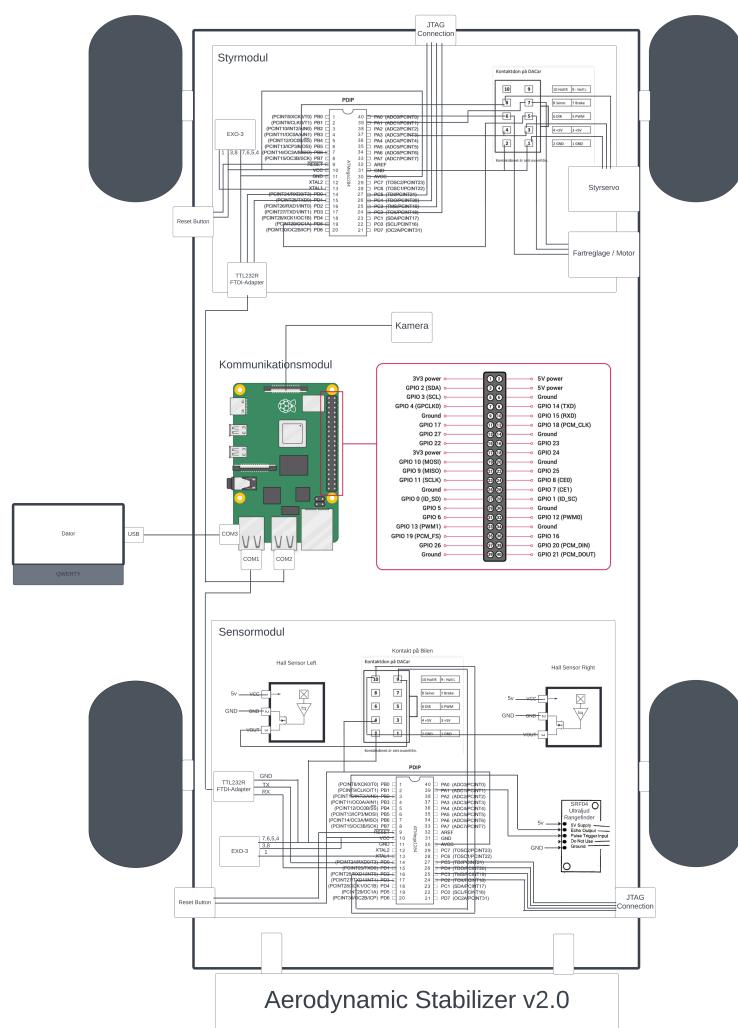
DOKUMENTHISTORIK

Version	Datum	Utförda förändringar	Utförda av	Granskad
0.1	2022-10-05	Första utkast	Gruppen	Zackarias Wadströmer
0.2	2022-10-13	Andra utkast	Gruppen	Johan Klasén
1.0	2022-11-01	Första Versionen	Gruppen	Hannes Nörager

1 INLEDNING

Det system som beskrivs i detta dokument har i uppdrag att agera autonom taxibil. Taxibilen har i uppdrag att hämta upp en passagerare vid en viss punkt för att sedan åka till en annan punkt där passageraren ska avlämnas. Systemet i fråga består av fyra olika delsystem; Kommunikationsmodul, Styrmodul, Sensormodul och en även en Extern Applikation.

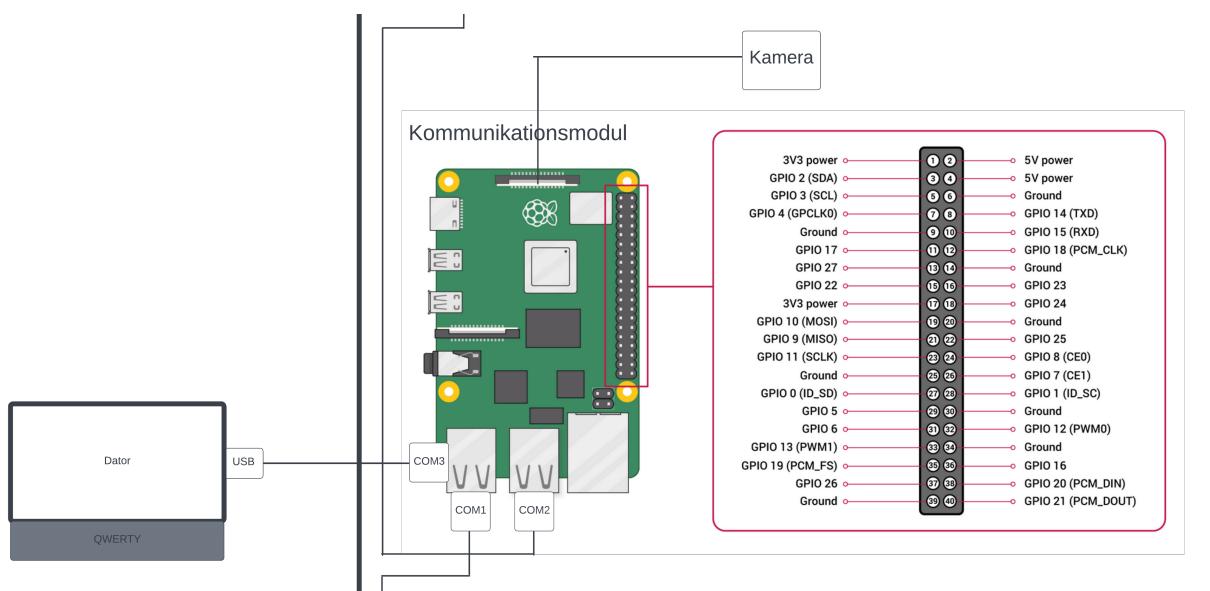
2 ÖVERSIKT AV SYSTEMET



Figur 1: Övergripande blockschema över systemet som ska konstrueras.
<https://www.overleaf.com/project/6328331fd6cd43a87f835da1>

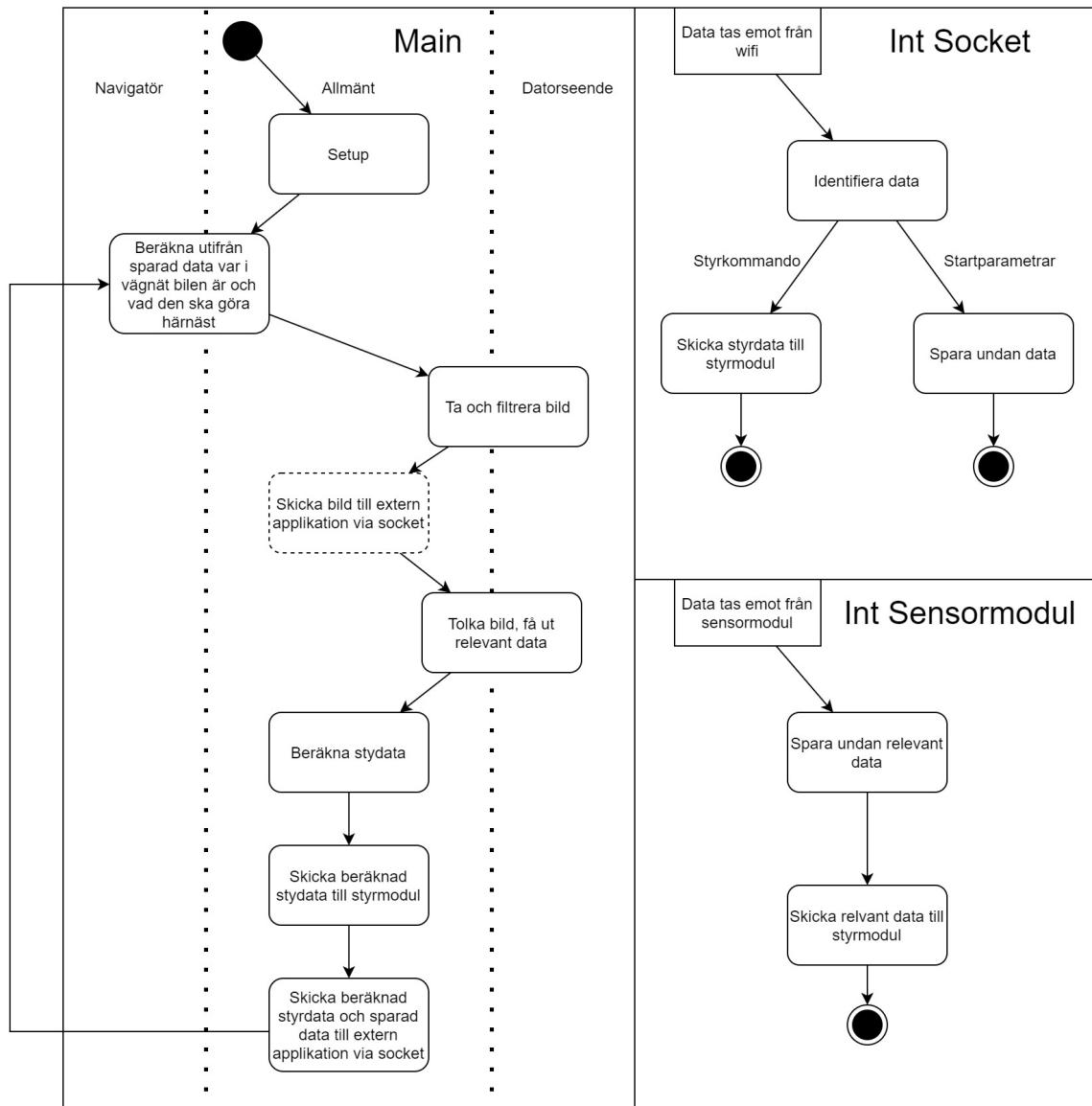
I figur 1 syns ett övergripande blockschema över systemet som ska konstrueras. Den visar de fyra delsystemen *kommunikations-, styr- och sensormodulerna* samt den *externa applikationen*, i bilden benämnd “Webapp“. I centrum står kommunikationsmodulen som är det enda delsystemet som är anslutet direkt till dem andra delsystemen. All data som behöver nå en annan modul går alltid via kommunikationsmodulen först om datan inte redan kommer därifrån. Kommunikationsmodulen är ansluten till både sensor- och styr-modulen via *UART*-protokoll och till webappen via WiFi. De tre modulerna ska alla kunna bytas ut mot en annan modul som följer samma implementationsspecifikation, även om projektet endast innehåller utvecklingen av en av varje modul.

3 DELSYSTEM 1: KOMMUNIKATIONSMODUL



Figur 2: Övergripande blockschema över kommunikationsmodulen.

Kommunikationsmodulen består av en Raspberry-pi 3B+ och en Raspberry-pi-kamera. Denna modul ansvarar för ett antal saker, för det första är modulen ansvarig för att bearbeta data som tas emot från kameran och skicka vidare till styrmodulen. Kommunikationsmodulen fungerar också som en brygga mellan sensormodulen och styrmodulen, den tar emot data från sensormodulen och skickar vidare till styrmodulen. Utöver de ovan nämnda funktionerna skickar kommunikationsmodulen all tillgänglig data till en extern dator. Kommunikationsmodulen kan också ta emot kommandon från en extern enhet och vidarebefordra dessa till styrmodulen.



Figur 3: Skiss på idé på hur processen i kommunikationsmodulen kan se ut. Streckad funktionalitet är önskat men inte krävt. Int är förkortning för interrupt och motsvarar subrutiner eller parallella processer som körs vid sidan av huvud processen.

3.1 Sensor, Kamera

En Raspberry-pi-kameran kommer att vara inkopplat direkt till Raspberry-pi för den att hantera indata internt i. Kameran kan producera olika video format med bildfrekvenser för just dem formaten som finns i tabell 1. Exakt vilket videoformat som används beslutas först när behovet av bildkvalité och frekvens blir uppenbart.

Tabell 1: Olika videoformat för Raspberry-pi kamera modul v2

Video format	Frekvens
1080	30
720	60
640x480	60/90

3.2 Navigatör

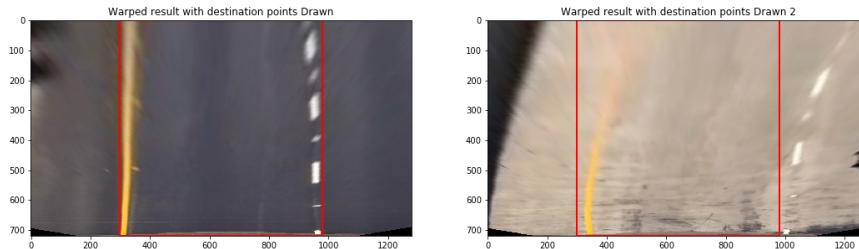
Navigatören är funktionaliteten i kommunikationsmodulen som ansvarar för att minnas var bilen är på banan och vad för beslut som den måste ta för att nå sitt mål snabbast, samt om den ska stanna vid nästa stopp. Navigatören tar emot data från den externa applikationen för att konstruera en datastruktur som beskriver banan innan start och använder denna genom körningen.

3.3 Datorseende

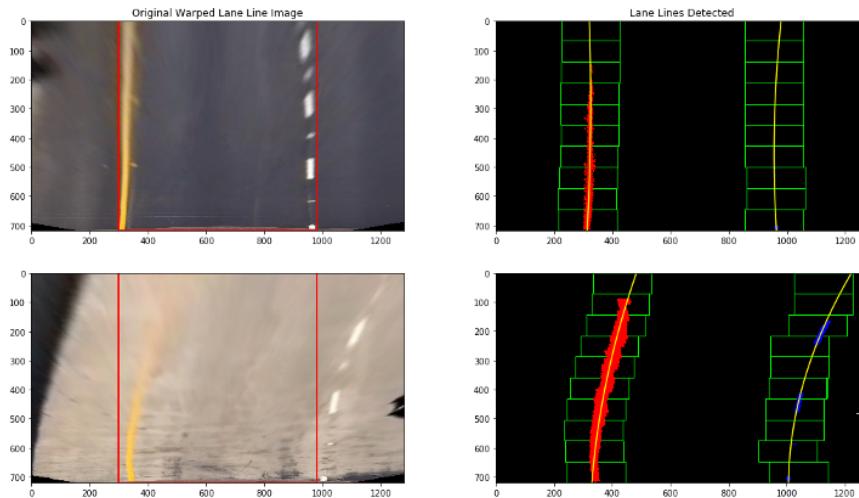
För att taxibilen ska kunna navigera sig genom banan kommer den använda sig av kamerasensorn i kommunikationsmodulen. I den modulen behandlas bilden för att producera styrdata till styrmodulen. Datorseendet kommer att hanteras med hjälp av biblioteket "*OpenCV*" vilket kommer att installeras på Raspberry-Pi. Bilden från kameran kommer att skalas och filtreras för att bli hanterbart och sedan tolkas med hjälp av "*OpenCV*" för att identifiera vägbana, stopp och korsningar. Den filtrerade bilden kommer att skickas till den externa applikationen för att visas upp. Tolkningen om vad som finns framför bilen kommer att beaktas i samband med informationen från navigatören för att producera korrekt styrdata. Från bilden ska även felet från centrum av vägen räknas ut samt graderna mellan bilen och vägen. Den producerade styrdatan kommer att bestå av avstånd till stopplinje, avstånd från vägens centrum och vinkelfel vilket alla kommer att skickas till styrmodulen. Avstånden kommer att vara i centimeter och vinkelfel i grader (negativa grader för vänster, positiva för höger, 0 för mitten).



Figur 4: En exempel bild på en väg med två linjer att följa. Anpassad från [1]



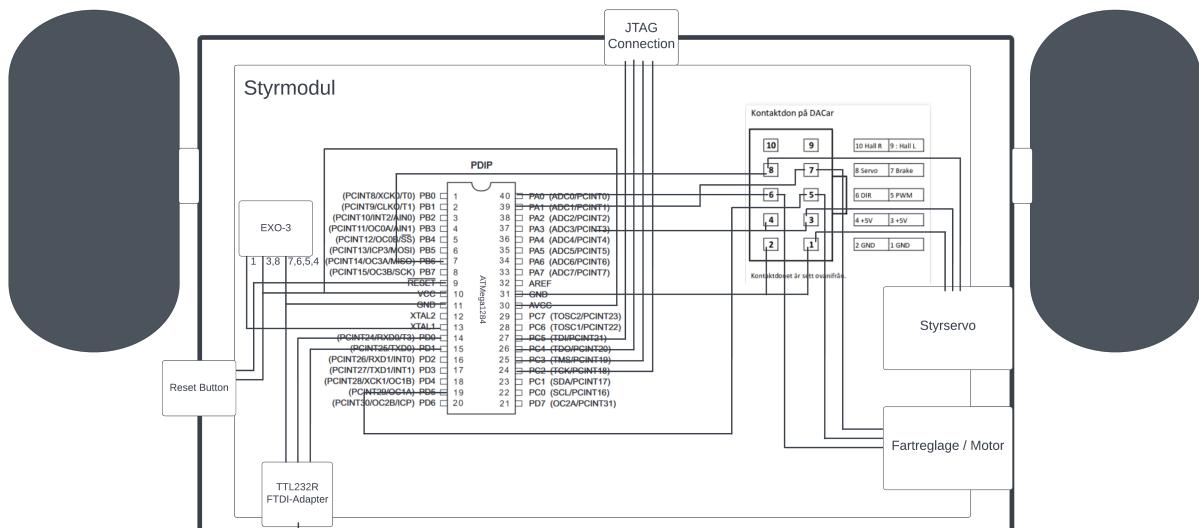
Figur 5: Detta är ett exempel på filtrerade bilder av dem från figur 4. Den har plattats ut så att perspektiv och linsböjning är borttaget och endast det område av intresse, vägen, är på bilden. Hämtad från [1]



Figur 6: Exempel på hur uträkningen kan se ut på bilderna från figur 5. Uträkningarna kommer att hitta linjerna på bilden och markera dem för att sedan lista ut böjning och placering av dessa i förhållande med bilen. Hämtad från [1]

4 DELSYSTEM 2: STYRMODUL

Styrmodulen består av en ATmega1284-processor. Denna processor har i uppdrag att kontrollera systemets motorer och servon, via koppling till ett kontaktdon i bilens skelett. Motorstyrningen sker via pulsmodulering där pulsbredden avgör hastighet respektive styrvinkel.



Figur 7: Övergripande blockschema över styrmodulen.

Modulen tar emot indata från kommunikationsmodulen i form av manuella kommandon, styrinformation för autonom körning i form av ett vinkelfel och avstånd till stopplinjer, samt tolkad sensordata i form av avstånd till fysiska hinder som kommer från ultraljudssensorn (se sektion 5.1 om sensorn). De manuella kommandona är förprogrammerade rutiner för att åka framåt, bakåt, höger, vänster eller stanna. Styrinformation används för att räkna ut hur bilen ska gasa, bromsa eller svänga för att hålla sig på banan, detta via en hastighetsloop och en PD-loop. Hastighetsloopen kontrollerar bilens hastighet och är beroende både av det avståndet till stopplinjer och vinkelfelet som ges i styrinformationen. Avståndet ska hjälpa till att avgöra när bilen nått sitt mål och ska stanna och vinkelfelet ska minimera risken för en s.k overshoot genom att få bilen att köra längsammare när vinkelfelet är väldigt stort.

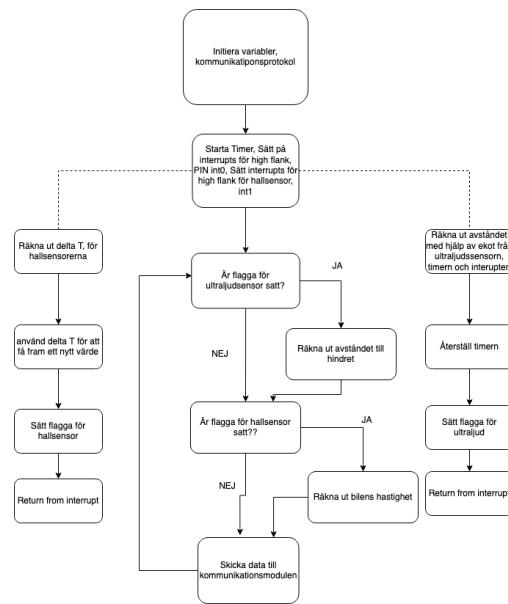
$$u[n] = K_p * e[n] + K_d * (e[n] - e[n-1])$$

Figur 8: Ursprunglig formel för PD-reglering

PD-loopen använder sig av nuvarande samt tidigare vinkelfel för att beräkna hur mycket bilen ska svänga för att nå ideal position (mitt emellan linjerna). Beräkningen görs enligt formeln som visas i figur 8 där e är ett invärde vid tid n , u utvärde och K konstanter för termerna. Utvärdet omvandlas sedan till en PWM-signal som skickas till styrservot och ger ett styrutslag. Styrutslaget kommer påverka bilens position i vägbanan och ge upphop till ett nytt vinkelfel i bildbehandlingen. När det nya vinkelfelet har tagits emot från kommunikationsmodulen används det i nästa iteration av loopen. Daten från ultraljudssensorn används för att avgöra när inbromsning för hinder ska ske och åsidosätter den autonoma körningen.

5 DELSYSTEM 3: SENSORMODUL

Sensormodulen består av en ATmega-1284 processor, en ultraljudssensor och 2 halleffektssensorer. ATmega-1284 processorn används för att samla data från ultraljudssensorn och Halleffektsensorn, båda via interrupts. Tolka datan och skicka vidare den till kommunikationsmodulen via UART.



Figur 9: Väldigt simpelt flowchart för koden i sensormodulen.

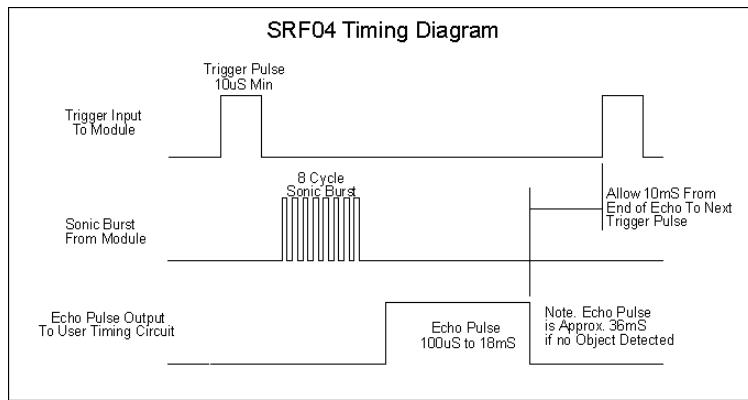
5.1 Ultraljudssensor

Ultraljudssensorn skickar ut en ultraljud-signal i en 45 grader bred kon och ger en hög signal för att visa sig redo att ta emot ett eko. Ekosignalen är hög tills ett eko uppfattats eller 36 ms förflutit (timeout). Längden till det objekt som signalen studsat mot kan då beräknas från bredden på ekosignalen i förhållande till ljudets hastighet (med en förmad korrigering). Sensormodulen tittar på eko-signaler på en bredd som motsvarar 0.1-1 m.

Ultraljudssenorns uppgift i systemet blir att fungera som en kollisionsdetektor och kommer sitta i framändan på bilen riktad i bilens körriktning. Ger sensorn utslag indikerar det att det finns något oväntat föremål på körbanan och man skickar då vidare avståndet till föremålet till styrmodulen så att den kan avgöra att den måste stanna,

5.2 Halleffektssensor

Vi använder en så kallad switch-halleffektssensor, vilket betyder att den slår på när ett stort nog magnetfält påverkar den, och slår av när magnetfältet försvinner. Detta ger en 5 V signal när en magnet passerar och hastigheten tolkas från frekvensen av 5 V pulser.

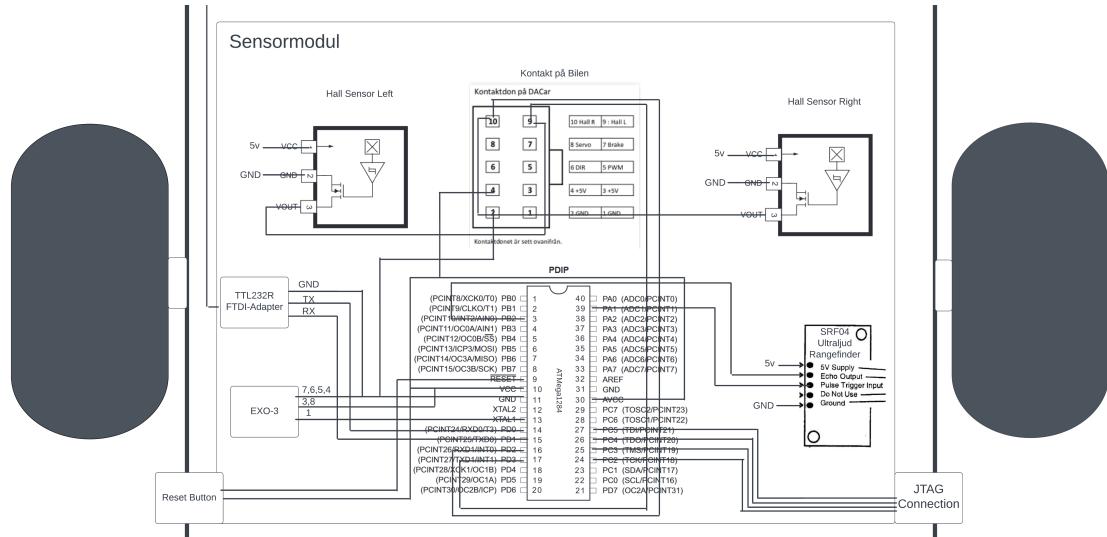


Figur 10: Tidsdiagram för ultraljudsensorns signaler. *Trigger Input* innebär starten av en mätning där sensorn svarar med att dra *Echo Pulse Output* hög så fort den är redo att lyssna på ett eko och låg igen efter 36 ms (timeout) eller så fort ekot har tagits emot, vilket som kommer först.

Halleffektssenorerna sitter i anslutning till bilens båda bakhjul och på hjulen sitter tio magneter. Dessa magneter kommer få hallsensorerna att ge utslag när bilen är i rörelse och genom att räkna takten på utslagen och omkretsen på hjulen kan man räkna ut bilens hastigheten.

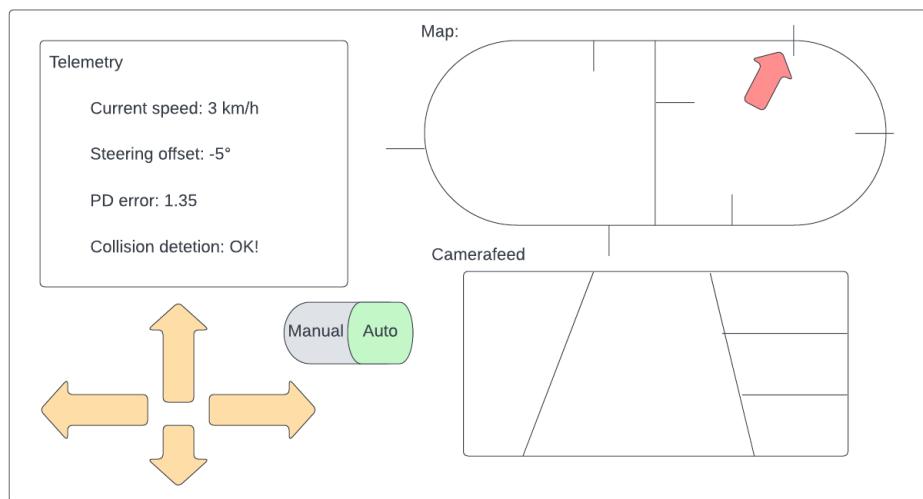
5.3 Data via UART

Datan som skickas från sensormodulen ska vara genomsnittshastigheten (enhet kan angis senare när man vet ungefär hur snabbt den kör) och även avståndet till eventuella hinder i cm.



Figur 11: Väldigt enkel skiss på sensormodulen och de komponenter som är kopplade till den.

6 DELSYSTEM 4: EXTERN APPLIKATION



Figur 12: Väldigt enkel skiss på hur layouten på den externa applikationen skulle kunna se ut.

Den Externa Applikationen består av en webserver som körs på en dator och kommunicerar telemetri och eventuellt en videoström mellan kommunikationsenheten och applikationen med hjälp av sockets i ett P2P förhållande, där servern är datorn och Raspberry-Pi'n är klienten. Telemetrin består av styrparametrar, som då kan ändra hur bilen beter sig på banan.

Figur 12 visar ett exempel på hur layouten på applikationen skulle kunna se ut. Det finns fält med information om diverse styrparametrar och sensordata. Det finns möjlighet att växla mellan autonomt och manuellt styrläge och knappar eller på något annat vis möjlighet att skicka manuella styrkommandon till taxibilen. Vid sidan om kan man se en visualisering av banan med bilens position och en separat ruta som visar livebilder från kameran. Lämpligtvis finns det även möjlighet att ange och starta ett autonomt köruppdrag.

7 KOMMUNIKATION MELLAN MODULERNA

7.1 Modulerna på taxin

Både sensor- och styrmodulen kommunicerar med kommunikationsmodulen via UART protokollet och kommunikationsmodulen är i sin tur ansvarig för att föra vidare relevant information från sensormodulen till styrmodulen.

När sensor- och styrmodulen kommunicerar med kommunikationsmodulen så sker det med en simpel sträng. Strängen skulle kunna se ut så här "`{D:45,V:5}`", där alla tecken är kodade med exempelvis ASCII. Där D är en etikett för avstånd i cm, V för velocity i km/h. Det gör att man troligen behöver plocka ut delsträngar och göra en omvandling när man vill lägga in värdena i nummerära variabler, men gör att det väldigt lätt att utöka sändningen om man i senare skede inser att man behöver skicka mer data.

7.2 Den Externa applikationen

Kommunikationsmoudulen kommunicerar med den externa applikationen med hjälp av Wi-Fi då den externa applikationen är en webapp som Raspberry-Pi'n kopplar upp sig mot. Kommunikationen sker med hjälp av ett *JSON*-format för informationsutbytet. Förhållandet mellan den externa applikationen och Raspberry-Pi'n är ett P2P förhållande.

8 KOMPONENTLISTA

Utöver bilen och dator för extern applikation krävs:

- Raspberry-pi 3B.
- Raspberry-pi Kamera v2 (160 graders lins).
- Ultraljudssensor, SRF04.
- Halleffektssensor, A1120.
- 2x Mikroprocessorer, ATmega1284P.
- 2x FTDI adapter, TTL232R.
- 2x Kristalloscillatörer, EXO-3 16MHz.
- 2x Reset-knappar.
- 2x Avstudsade knappar för testning av halleffektsensorer.

9 IMPLEMENTERINGSSTRATEGI

9.1 Konstruktionsmetodik

När bilen utvecklas så ska det ske utifrån ”inifrån ut”, dvs att modulerna utvecklas i isolation för att sedan integreras till ett system.

9.2 Programmeringsspråk

Mjukvaran i Raspberry-Pi i kommunikationsmodulen kommer att programmeras genom ”*Python*”. Mikrodatorerna i sensormodulen och styrmodulen kommer att programmeras med ”*C/C++*”.

9.3 Kontinuerlig Testning

När modulerna utvecklas så kan det ske med kontinuerlig testning. Denna testning kan ske med en logikanalysator kopplade till modulerna. Man kan då använda sig utav en *JTAG*-anslutning för att få en korrekt bild av vad som händer i processorn.

Ett annat sätt är att datan som skickas till kommunikationsmoudlen sparas i loggfiler så att man kan titta i loggfilen om något behöver debuggas.

REFERENSER

- [1] M. Elmasry, “Computer vision for lane finding,” <https://towardsdatascience.com/computer-vision-for-lane-finding-24ea77f25209>, 2018, [Online; hämtad Oktober 11, 2022].