LINKÖPINGS UNIVERSITET

TSEA29

Konstruktion med mikrodatorer

Designspecifikation

Autonom taxibil

2018-11-06

Redaktör: Jakob Arvidsson

Grupp 2

*V*ersion 0.1

Status

Granskad	JC, DD, EH	2018-11-06
Godkänd	Peter Johansson	

Projektidentitet

Grupp 2, 2018-11-06, Autonom taxibil

Namn	**Ansvarsområde**	**Epost**
Jakob Arvidsson	Projektledare och kundkontakt	jakar180@student.liu.se
Yousef Hashem	Testansvarig	youha847@student.liu.se
Noah Hellman	Arkitekt	noahe116@student.liu.se
Juan Basaez	Dokumentansvarig	juaba731@student.liu.se
Johan Can	Gränssnitt	johca907@student.liu.se
Dennis Derecichei	Hårdvara	dende301@student.liu.se
Emir Hadzisalihovic	Elektronik	emiha868@student.liu.se

E-postadress till gruppen
Grupphemsida: Git-repo för projekt

Kund: ISY
Kontaktperson hos kund: Mattias Krysander

Kursansvarig: Anders Nilsson
Handledare: Peter Johansson

Innehåll

Dokumenthistorik

Version	**Datum**	**Utförda förändringar**	**Utförda av**	**Granskad**
0.1	2018-11-06	Första utkast.	Grupp 2	JC, DD, EH

1 Inledning

Det här dokumentet går in i detalj på hur produkten beskriven i projektplanen skall konstrueras samt med vilka produkter. Utöver text kommer dokumentet innehålla fullständiga kretsscheman på de olika modulerna som bygger upp systemet. Designen sträcker sig över både mjukvara och hårdvara.

1.1 Bakgrund

Ett projekt utförs som moment i kursen TSEA29 på Linköping Universitet. Designspecifikationen är en del av LIPS-modellen som används för att genomföra projektet.

1.2 Syfte

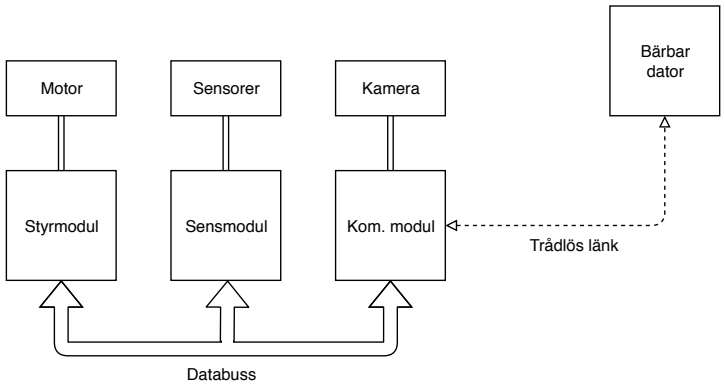
Syftet med designspecifikationen är att designa systemet i detalj innan påbörjandet av implementationen. På detta vis går det att följa designspecen som en mall under implementationsfasen.

2 Översikt

I denna del förklaras designen av systemet översiktligt.

2.1 Framträdande egenskaper

Produkten består av tre moduler: Kommunikation-, sensor- och styrmodul. Sensormodulen kommer implenteras på ett virkort varav de andra modulerna ska vara på ett separat virkort. Produkten har två avståndsmätare och en kamera till hjälp när produkten ska samla in information om sin omgivning. Vardera mikrokontroller klockas med en kristalloscillator. Produkten är ansluten till en fjärrklient via WLAN.



Figur 1: Övergripande bild över systemet och dess moduler. SPI-buss används

Kommunikationsmodulen består av en Raspberry Pi och kommer vara en central kommunikations- och beslutsenhet som ansvarar för att ta beslut om taxins beteende samt hantera kommunikationen mellan modulerna. Styrmodulen består av motorer och en mikrokontroller och står för drift samt reglering av taxins hastighet och svängradie. Sensormodulen består av en mikrokontroller samt avståndsmätare och halleffektsensorer monterade på bilens hjul.

2.2 Sensorer

Kameran ska fästas en bit ovan alla korten och titta framåt med en vinkel nedåt. Avståndsmätaren som ska mäta avstånd till hinder ska fästas på framsidan av bilens chassi. Om det behövs så ska lämpliga fotfästen skrivas ut med en 3D-skrivare. Den andra avståndsmätaren ska användas för att avgöra när ett hinder har passerats vid omkörning och kommer att placeras på bakre höger sida av bilens chassi. Odometern som används till att mäta kördistans är redan färdigmonterad på chassit.

2.3 Ställdon

Hjulen, motorerna och ställdonen är redan färdigmonterade. Motorerna används för att få bilen att åka framåt och bakåt samt för att svänga.

3 Kommunikationsmodul

Kommunikationsmodulen ska agera som taxins hjärna. Den styr bilen under autonom körning och kommunicerar med fjärrklienten. Sensorvärden från sensormodulen tolkas av kommunikationsmodulen som därefter skickar felvärden till styrmodulen.

3.1 Funktion

Kommunikationsmodulen har tre huvudsakliga uppgifter; bildbehandling, kommunicera med fjärrklienten och kontrollera taxin autonomt.

Bildbehandling skall utföras av kommunikationsmodulen för att avgöra bilens position i vägfilen och upptäcka stopplinjer. Med hjälp av kamerans bilder på vägen skapas ett felvärde som kan användas för att justera taxins riktning.

Kommunikation med fjärrklienten sköts av kommunikationsmodulen för att skicka sensorvärden och annan relevant information. Modulen skall även ta emot ett uppdrag som har skapats utifrån en karta och destination som användaren har matat in via fjärrklienten.

Autonomitet utförs av kommunikationsmodulen för att utföra uppdraget. Kommunikationsmodulen hämtar sensordata från sensormodulen och bildbehandlar kamerabilderna för att utföra beslut i realtid. Besluten kommer därefter att översättas till felvärden som skickas till styrmodulen.

3.2 Hårdvaruimplementation

Kommunikationsmodulen kommer att implementeras med hjälp av en Raspberry Pi 3, där en WLAN-komponent redan finns integrerat. Kameran kommer att kopplas direkt till kameraporten som finns på Raspberry Pi-kortet. GPIO-pinnarna kommer att användas för att ansluta till övriga moduler. Eftersom SPI-protokollet ska användas, kommer GPIO-portar för de signalerna som protokollet kräver användas. Det kommer att finnas nivåskiftare som skiftar spänningen på signalerna som går mellan kommunikationsmodulen och mikrokontrollerna. Ett kretsschema för modulen finns i bilaga A.1.

3.2.1 Budget

Nedan komponenter i kommunikationsmodulen ingår inte i baschassit och behöver beställas.

- **Raspberry Pi Camera V2 Video Module** Kamera till Raspberry Pi.
- **2 × TXB0104 Level Shifter** Nivåskiftare för 4 signaler vardera.

3.2.2 Kontroll och bedömning

För att kunna köra systemet med SPI-protokollet kommer ett antal pinnar att behövas. De signaler som kräver pinnar är:

- **MOSI** Datasignal från master till slav.
- **MISO** Datasignal från slav till master.
- **SCLK** Synkron klocka från mastern till mikrokontrollerna.
- **SS1** Slave select för sensormodulen.
- **SS2** Slave select för styrmodulen. slav.

Raspberry Pi:n har stöd för 2 bussar med SPI-protokollet varav 10 pinnar totalt. Eftersom endast 5 pinnar kommer användas räcker antalet pinnar. Två nivåskiftare kommer att användas vilket medför stöd för 8-signaler, men endast 5-signaler kommer att användas.

Det mest krävande för kommunikationsmodulen är bildbehandlingen. Raspberry Pi 3 model B har en fyrakärnig BCM2837 klockad på 1.2GHz och 1GB internt minne. Detta bör vara tillräckligt för att uppnå 20 bilder per sekund med en effektiv implementation i C eller C++.

3.3 Mjukvaruimplementation

Raspberry:n kommer köra operativsystemet Raspbian och program som exempelvis wpa_supplicant för att sätta upp anslutningen för WLAN. Ett program för kommunikationsmodulen kommer skrivas i C eller eventuellt C++ och ska bestå av två trådar; en huvudtråd som hanterar uppdrag, bildbehandlar och svarar på kommandon från fjärrklienten samt en tråd som sköter kommunikation med andra moduler.

Huvudtråden kommer att starta en TCP/IP-server för att lyssna på kommandon från fjärrklienten. Tråden kommer köra en while loop där den först accepterar inkommande anslutningar och agerar utefter protokollet som är specificerat i sektion 7.1.1. Därefter kommer den hantera uppdraget och eventuellt bildbehandla. Därefter uppdatera felvärde som ska skickas till styrmodulen.

SPI-bussen kommer hanteras av en dedikerad tråd. Den skall kontinuerligt hämta värden från sensormodulen oberoende av andra trådar. När bildbehandlingen är färdig med nästa bild skall felvärdet skickas till styrmodulen. Detta sker i en separat tråd för att inte fördröja bildbehandlingen i väntan på bussen.

3.3.1 Uppdrag

För att utföra uppdraget kommer en kö av kommandon tas emot från fjärrklienten. När uppdraget börjar kör taxin framåt och följer filen med hjälp av bildbehandling och reglering. Inför varje stopplinje tas ett kommando från kön och utförs. Om ett hinder upptäcks under uppdraget kommer bilen att stanna tills hindret har försvunnit eller eventuellt passera hindret. När ett visst antal stopplinjer har passerats och kön är tom är uppdraget avklarat. Nedan är en lista av alla möjliga kommandon hur de utförs.

`ignr` Kör förbi stopplinjen utan att stanna.

`entr` Sväng höger in i rondellen och hitta rondellens körfil.

`exit` Sväng höger ut ur rondellen och hitta körfilen.

`stop` Stanna framför stopplinjen, fortsätt efter några sekunder om kön inte är tom.

`park` Parkera i fickan till höger efter stopplinjen, lämna parkeringen och fortsätt efter några sekunder om kön inte är tom.

4 Styrmodul

Styrmodulen är en modul som har i uppgift att ta emot felvärden från kommunikationsmodulen och reglera bilens drivmotor och svängmotor utifrån dessa.

4.1 Funktion

Styrmodulens syfte är att hantera den direkta kontrollen av styrreglagen för taxins motorer. Styrmodulen ska reglera drivmotorn och svängmotorn utifrån felvärden som ges av kommunikationsmodulen.

4.2 Hårdvaruimplementation

Styrmodulen består av en mikrokontroller, taxins motorer samt en LCD för felsökning. Ett detaljerat kretsschema finns i bilaga A.2.

4.2.1 Budget

Utöver motorn och servon som redan sitter monterade på chassit behöver styrmodulen följande komponenter.

- **ATMega1284** Modulens microprocessor.
- **JTAG ICE mkII** Debugger för programmering och felsökning med microprocessorn.
- **LCD JM162A** LCD-display för att visa parametrar under körning i felsökningssyfte.
- **IQEXO3** Kristalloscillator på 20Mhz, skall använda som systemklocka till AVR.
- **Tryckknapp** Knapp används till RESET.

4.2.2 Kontroll och bedömning

Nedan är följande pinnar som används på mikrokontrollern.

- **PA0-PA7** Datasignaler till LCD.
- **PB0** RS-signal till LCD.
- **PB3** PWM-signal till drivmotor.
- **PB4-PB7** SS, MOSI, MISO och SCLK för SPI-bussen.
- **PC2-PC5** TCK, TMS, TDO, TDI för JTAG.
- **PD5** PWM-signal till svängmotor.
- **RESET** Resetsignal.

- **XTAL1** Klocka från kristalloscillatorn.

Inga fler signaler behövs så pinnarna på mikrokontrollen är tillräckliga.

Mikrokontrollen behöver endast kommunicera via SPI, utföra enkel reglering och därefter skapa en PWM-signal för motorerna. Inget av detta är särskilt krävande och bör klaras utan problem av ATMega1284. Kontrollern har ett flash-minne på 128kB, vilket bör vara mer än tillräckligt för att lagra programmet.

4.3 Mjukvaruimplementation

Styrmodulens program kommer bestå av en main loop som reglerar bilens hastighet efter ett felvärde. Ett avbrott som aktiveras när kommunikationsmodulen vill kommunicera kommer att ta emot nya felvärden med jämna mellanrum.

4.3.1 Huvud-loop

I huvudloopen kommer värdena från kommunikationsmodulen göras om till motsvarande tidsintervall i PWM. Beroende på hur snabbt det går så kommer huvudloopen vänta på avbrott.

4.3.2 Avbrott

När SPI-kontrollern har tagit emot en byte av data sätter den SPIF i SPSR (SPI Status Register) och aktiverar ett avbrott. Avbrottsrutinen kan flaggan kontrollera för att avgöra att kommunikationsmodulen vill skicka felvärden och börja ta emot värden.

5 Sensormodul

Sensormodulen ska agera som taxins känselspröt. Den kommer att mäta av olika värden från de olika sensorerna som finns för att sedan skicka mätdata till kommunikationsmodulen.

5.1 Funktion

Sensormodulen ska hämta eventuellt filtrerade värden från sensorerna, omvandla till linjära enheter och skicka vidare till kommunikationsmodulen.

Sensorer Taxin kommer att ha en optisk avståndsmätare som är riktad framåt för att upptäcka hinder framför bilen samt en optisk avståndsmätare på sidan för att detektera hinder höger om bilen, t.ex. vid omkörning.

Filtrering Eftersom sensorerna kan bli påverkade av eventuella störningar kommer det eventuellt att finnas brusfilter som filtrerar bort dessa störningar för att skapa så noggranna sensorvärden som möjligt.

5.2 Hårdvaruimplementation

Sensormodulen består huvudsakligen av en mikrokontroller, och fyra sensorer; en avståndsmätare på framsidan, en avståndsmätare på höger sida samt två halleffektsensorer placerade på två av hjulen. Mikrokontrollern är klockad av en kristalloscillator. Kontrollern är även kopplad till en LCD-display för att kunna visa värden vid felsökning.

På LCD:n har följande två pinnar valts till att ha ett konstant värde: R/W och E. Eftersom inget krav finns på att kunna släcka LCD-panelen alternativt läsa från displayen så är enable och write-signalen jordas.

Avståndsmätarna skickar en spänning direkt till mikrokontrollens AD-omvandlare. Spänningens värde motsvarar ett avstånd. Utspänningen kan eventuellt behöva filtreras från brus.

Önskad klockfrekvens från oscillatorn är 24Mhz.

Ett kretsschema över modulen finns i bilaga A.3.

5.2.1 Budget

Nedan är produkter som ej medföljer baschassit och behöver beställas.

- **ATMega1284** Modulens microprocessor.
- **GP2Y0A02YK** Avståndsmätare för hinder framför bilen.
- **GP2D120** Avståndsmätare för hinder höger om bilen.
- **LCD JM162A** LCD-display för att visa parametrar under körning i felsökningssyfte.

- **IQEXO3** Kristalloscillator på 20Mhz, skall använda som systemklocka till AVR.

Modulen använder även en JTAG ICE mkII och tryckknapp som delas med styrmodulen.

5.2.2 Kontroll och bedömning

Nedan är följande pinnar som används på mikrokontrollern.

- **PA0 & PA2** Vardera port får insignal från en avståndsmätare.
- **PB2-PB3** Vardera port får en avbrottsignal från en odometer.
- **PB4-PB7** SS, MOSI, MISO och SCLK för SPI-bussen.
- **PC2-PC5** TCK, TMS, TDO, TDI för JTAG.
- **PC7** Reset signal till LCD-display.
- **PD0-PD7** Databuss till LCD-display.
- **RESET(9)** Knapp till reset för MCU.
- **XTAL1** Klocka från Kristalloscillatorn.

Mikrokontrollen behöver endast kommunicera med kommunikationsmodulen via SPI samt ta emot värden från sensorerna och göra enkla beräkningar vid omvandling. Mikrokontrollen bör inte ha något problem att utföra dessa uppgifter.

5.3 Mjukvaruimplementation

Sensormodulen ska bestå av en tom while loop och ett avbrott som aktiveras när kommunikationsmodulen efterfrågar värden. Sensormodulen skickar då de senaste sensorvärdena och samplar därefter nästa grupp av värden. Avbrottet aktiveras som beskrivet i sektion 4.3.2

Halleffektsensorerna kommer skicka en avbrottssignal varje gång en magnet på hjulet passerar en sensor. Avbrottsrutinen ser då till att öka den körda distansen.

6 Fjärrklient

Fjärrklienten fungerar som ett användargränssnitt och har även en viktig roll i utförandet av uppdrag.

6.1 Funktion

Fjärrklientens uppgift är att bidra med ett gränssnitt till användaren för att kontrollera och övervaka taxin, samt skapa och skicka instruktioner till bilen utefter användarens uppdrag.

Gränssnitt Gränssnittet ska tillåta användaren att välja antingen autonom eller manuell körning. Vid manuell körning skickas styrkommandon till taxin när användaren trycker på piltangenterna. Vid autonom körning ska gränssnittet tillåta användaren att mata in en karta samt ange destination.

Gränssnittet ska visa nuvarande och historiska mätvärden från taxin med hjälp av diagram. Gränssnittet skall även visa taxins position under aktiva uppdrag.

Skapa kommandon för uppdrag Fjärrklienten ska utifrån den inmatade kartan och destination; räkna ut den kortaste vägen dit och utifrån vägen skapa en kö av kommandon som taxin kan följa för att nå destinationen.

6.2 Mjukvaruimplementation

Fjärrklienten kommer implementeras i Python och bestå av två olika trådar; en för det grafiska gränssnittet samt en huvudtråd för allt annat. Ett flödesdiagram för fjärrklienten kan ses i bilaga B.1.

6.2.1 Trådar

Programmet skall köras parallellt med två trådar för att förhindra att gränssnittet fryser för användaren. Programmet kommer behöva vänta på svar från taxin samt utföra längre beräkningar vilket innebär att programmet kanske inte hinner uppdatera gränssnittet tillräckligt ofta om det endast använder en tråd.

6.2.2 Karta och uppdrag

Kartan representeras med en riktad graf där varje nod motsvarar en stopplinje av tejp i vägfältet. Rondeller representeras av endast en nod istället för en nod per stopplinje. En båge representerar en enkelriktad körfil från en stopplinje eller rondell till en annan stopplinje eller rondell. Den här grafen representeras i sin tur av nedan datastrukturer.

Karta Kartan utgörs av en lista med noder.

Nod En nod är en klass som består av en nodtyp och en sorterad lista av utgående bågar. Det finns tre olika nodtyper; stopplinje, parkeringsficka och rondell.

Båge En båge består av ett avstånd och en destination. Den representeras av en tuple där första värdet är ett avstånd och det andra värdet är en pekare till en nod.

Ovanstående datastrukurer är valda för att göra det smidigt att räkna ut den närmaste vägen med Dijkstras alogritm. Med algoritmen utgår man från en nod och jämför alla dess grannar. Med ovanstående struktur kan man utgå från startnoden och därefter rekursivt gå igenom nodens alla grannar och grannar av grannar. Alternativt, med en *adjacency list* kräver detta en sökning efter alla bågar som noden utgår ifrån. Med en *adjacency matrix* måste man kolla igenom varje element i nodens rad i matrisen och eftersom graferna inte alls är särskilt täta är det inte särskilt effektiv användning av cykler eller utrymme.

När en närmaste väg har bestämts ska en kö av kommandon skapas. Detta kan göras genom att iterera varje nod i vägen och lägga till nedan kommandon för varje nodtyp. Hur taxin agerar vid varje kommando är specificerat i sektion 3.3.1.

stopplinje `stop` om markerad som destination, annars `ignr`.

parkeringsficka `park` om markerad som destination, annars `ignr`

rondell En `entr`, därefter en `ignr` för varje avfart den ska passera och slutligen en `exit`.

För att kunna avgöra vilken avfart taxin ska ta i en rondell är det viktigt att de utgående bågarna i rondellens nod är sorterade i samma ordning som avfarterna är placerade i den riktiga banan.

7 Kommunikation

Kommunikation kommer att ske mellan fjärrklienten och kommunikationsmodulen samt mellan kommunikationsmodulen och vardera mikrokontroller i styr- och sensormodulen. Denna sektion specificerar hur kommunikationen ska uppnås samt hur informationen som växlas skall uttryckas av sändaren och tolkas av mottagaren.

7.1 Trådlös länk

Taxin och fjärrklienten kommer att kommunicera via TCP/IP. Kommunikationsmodulen och fjärrklienten ansluter till ett gemensamt WLAN. Därefter binder kommunikationsmodulen sin IP-address och en port till en sockel som fjärrklienten kan ansluta till.

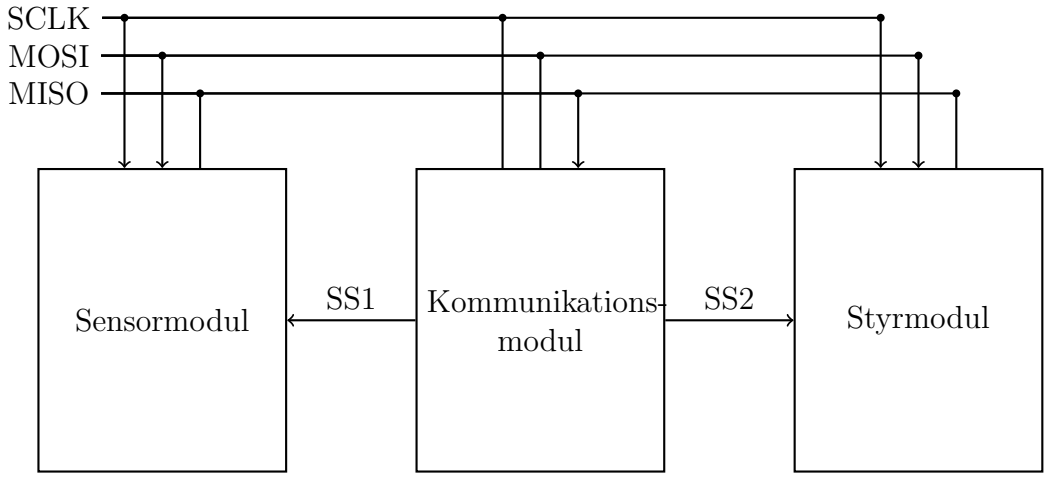
7.1.1 Protokoll

Kommunikationsmodulen kommer att vänta på inkommande instruktioner via sockeln och exekvera dem när de tas emot. Instruktioner skickas i form av strängar och kan inkludera argument. För instruktioner utan argument skickas endast en sträng med instruktionens namn; till exempel `get_sensor_data`. För instruktioner med argument består strängen av instruktionens namn följt av ett kolon och ett eller flera argument. Flera argument separeras med kommatecken; till exempel `set_mission:ignr,ignr,park`.

Instruktion	**Argument**	**Handling**
`get_sensor_data`	inga	En sträng med mätvärden för varje sensor skickas tillbaka.
`get_mission_status`	inga	En sträng med information om uppdraget skickas tillbaka, till exempel nuvarande position.
`set_mission`	variabelt antal, kommandon	Sätt en kö av kommandon som ska utföras vid stopplinjer. Kommandona är specificerade i sektion 3.3.1.
`execute_mission`	inga	Utför uppdraget enligt kommandona i det nuvarande satta uppdraget.
`cancel_mission`	inga	Avbryt utförandet av uppdrag. Uppdraget kan återupptas igen med `execute_mission`.
`set_speed_delta`	felvärde	Sätt felvärde för hastigheten, ignoreras om uppdrag är aktivt.
`set_turn_delta`	felvärde	Sätt felvärde för svängradien, ignoreras om uppdrag är aktivt.

7.2 Mellan taxins moduler

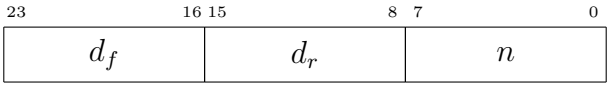
Mellan sensormodulen, kommunikationsmodulen och styrmodulen används gränssnittet SPI. Kommunikationsmodulen agerar som *master* medan sensormodulen och styrmodulen agerar som *slaves*. Varje modul är kopplad till en gemensam SCLK-signal för synkronisering, en MISO-signal för data till master:n och MOSI för data från master:n som visat i figur 2. Sensormodulen skickar sensorvärden till kommunikationsmodulen och kommunikationsmodulen skickar styrkommandon till styrmodulen. Eventuellt kan data överföras till sensormodulen eller från styrmodulen vid felsökning.



Figur 2: Den gemensamma databussen mellan modulerna.

7.2.1 Från sensormodul till kommunikationsmodul

Sensormodulen ska fortlöpande skicka värden för varje sensor till kommunikationsmodulen. Figur 3 visar bitsekvensen som sensormodulen skickar.



Figur 3: Bitsekvensen för datan som skickas fortlöpande till kommunikationsmodulen till sensormodulen.

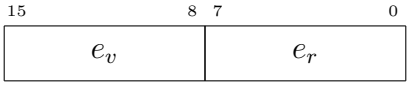
Frontavståndet d_f är ett 8-bitars osignerat heltal som representerar avståndet från sensorn på taxins framsida. Heltalet representerar avståndet som frontsensorn läser i centimeter.

Sidoavståndet d_r är ett 8-bitars osignerat heltal som representerar avståndet från sensorn på taxins högersida. Heltalet representerar avståndet som sidosensorn läser i centimeter.

Varvtalet n är ett 8-bitars osignerat heltal som specificerar antalet varv som taxins hjul har rullat sen förra sändelsen.

7.2.2 Från kommunikationsmodul till styrmodul

Kommunikationsmodulen ska skicka ett felvärde e_v för hastigheten och ett felvärde e_r för styrradien till styrmodulen med jämna mellanrum. Styrenheten ska försöka justera hastigheten och radien som specificerat av det senaste mottagna värdet. Figur 4 visar bitsekvensen för datan som sänds.



Figur 4: Bitsekvensen för datan som skickas fortlöpande från kommunikationsmodulen till styrmodulen.

Hastighetsfelet e_v består av ett signerat 8-bitars heltal i tvåkomplementsform. Värdet är ett felvärde för hastigheten som beräknas $e_v = v_{önskad} - v_{nuvarande}$. Negativa värden betyder att bilen kör för snabbt och positiva värden betyder att farten ska öka.

Styrfelet e_r består av ett signerat 8-bitars heltal i tvåkomplementsform. Värdet är ett felvärde för styrradien som beräknas av kommunikationsmodulen utifrån bilens förhållande till väglinjerna. Ett nollvärde betyder att hjulen står rätt, negativt att hjulen bör svänga mer åt vänster och positivt att hjulen bör svänga mer åt höger.

8 Implementationsstrategi

För att implementationen ska ske smidigt så ska en implementationsstrategi bestämmas. Det inkluderar tillvägagångssätt, återkoppling och timing. Tillvägagångssättet beskriver strategin för det praktiska, exempelvis i vilken ordning modulerna ska implementeras. Återkoppling avser hur implementationen kontinuerligt under arbetets ska testas, t.ex. genom att koppla en display till en modul för att verifiera indata samt utdata. Slutligen anger timing hur snabbt utdata respektive indatan ska uppdateras i varje modul. Detta kan avse hur många bilder i sekunden sensormodulen ska skicka till kommunikationsmodulen.

8.1 Tillvägagångssätt

Varje modul kommer att utvecklas parallellt för att sedan integreras tillsammans. Eftersom varje modul kopplas till kommunikationsmodulen kan en modul delvis integreras så fort den är klar om kommunikationsmodulen är redo. För att i så hög grad som möjligt undvika stora problem i slutet ska funktioner som beror av varandra och som finns i olika moduler implementeras samtidigt för att sedan testas. Ett exempel är avståndsigenkänningen i kommunikationsmodulen och stoppfunktionen i styrmodulen, där båda kan testas tillsammans för att verifiera att bilen kan stanna för ett hinder.

8.2 Återkoppling

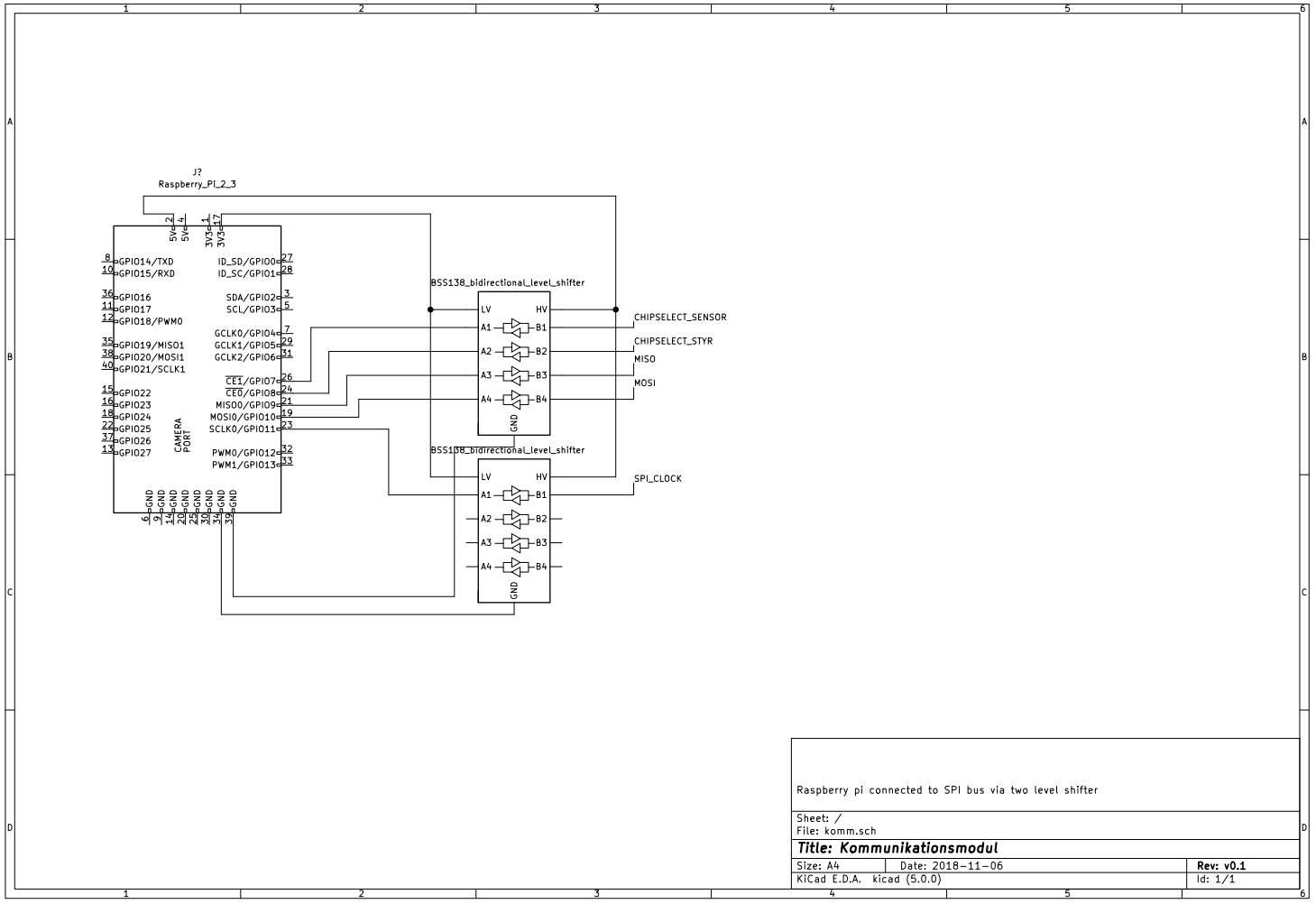
För att kunna felsöka problem under implementationen kommer olika sätt att få återkoppling att användas. Sensor- och styrmodulen ska kopplas till en LCD-display för att kunna visa värden. Värden kan också skickas via bussen till kommunikationsmodulen när den är implementerad. Kommunikationsmodulen kommer köra en SSH-server så att det går att komma åt värden på modulen. Värden kommer kunna skrivas till `stdout` eller filer. Programmet kommer också kunna köras via GDB så att minnet direkt kan läsas. På liknande sätt kan fjärrklienten på datorn felsökas.

8.3 Timings

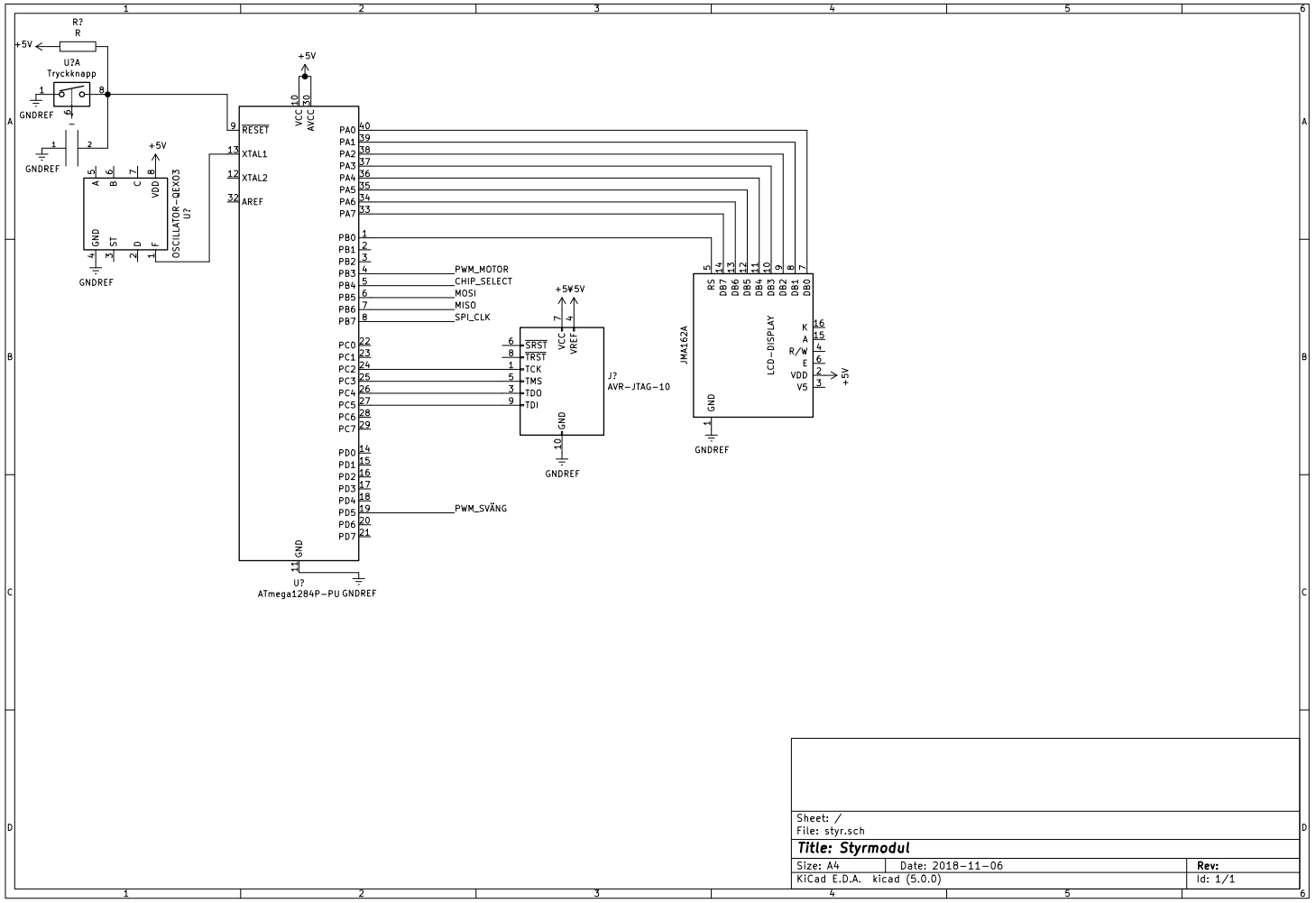
Det som begränsar frekvensen av beslut från kommunikationsmodulen är bildbehandlingen. Det vore önskvärt att nå åtminstone 20 bilder sekund. Utifrån vilken bildbehandlingsfrekvens som uppnås kommer frekvensen av sampling från sensorerna och styrkommandon till styrmodulen att anpassas.

A Bilaga. Kretsscheman

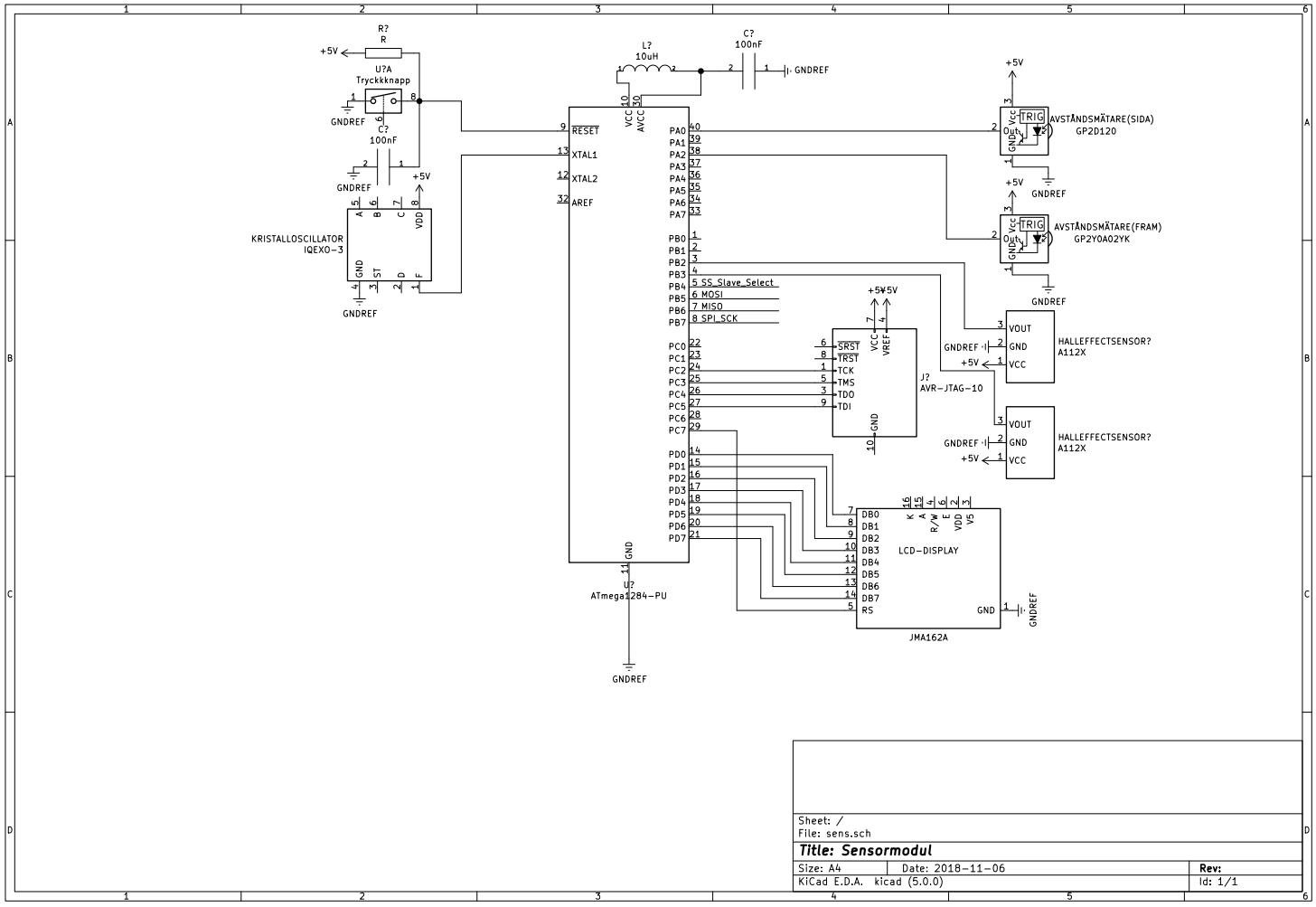
A.1 Kommunikationsmodul



A.2 Styrmodul



A.3 Sensormodul



B Bilaga. Flödesdiagram

B.1 Fjärrklient