

Rapport

62410 - CDIO-project, DTU-Diplom
Gruppe 14

20. Juni 2019



Christian Alexander Sauer Mark
s164833@student.dtu.dk



Oliver Storm Køppen
s175108@student.dtu.dk



Mads Astrup Andresen
s174874@student.dtu.dk



Ali Harb El-Haj Moussa
s175119@student.dtu.dk



William Wulff
s174880@student.dtu.dk



Magnus Hansen
s175198@student.dtu.dk



Jonas Tofte Thomsen
s174867@student.dtu.dk



Sebastian Bilde
s175116@student.dtu.dk

Demo af robotten kan findes på: <https://www.youtube.com/watch?v=cduAYe1SKuA>
Github: https://github.com/unizippro/CDIO_Robot

Indhold

1 Indledning (William)	4
1.1 Introduktion	4
1.2 Problemstilling	4
2 Status over projektet og produktet (Christian & Jonas)	4
2.1 Projektplan	4
2.1.1 Projektledere	4
2.2 Endelig Status	4
2.2.1 Opgaver der blev ikke nåede at blive lavet færdige	5
2.2.2 Konklusion på statusrapport	5
3 Proces (Christian & Jonas)	5
3.1 Metoder og værktøjer	5
3.2 Løbende projektændringer	7
3.3 Projektstyring i projektet	7
4 Analyse og design	8
4.1 Krav	8
4.2 Afgrænsninger	8
4.3 Billedebehandling (Ali & Sebastian)	8
4.3.1 Farvespektrum	8
4.3.2 Bolde samt robottens placering	8
4.3.3 Banens hjørnepunkter samt kryds	9
4.4 Hardware (Mads & William)	9
4.5 Software (Magnus, Jonas, Christian & Oliver)	11
4.5.1 Robottens applikation	11
4.5.2 Ruteplanlægning	12
5 Implementation	14
5.1 Vakuumboldopsamlingsaggregat (Mads, Jonas & William)	14
5.2 Billedebehandling (Ali & Sebastian)	15
5.2.1 Farvespektrum	15
5.2.2 Bolde samt robottens placering	15
5.2.3 Banens fire hjørnepunkter samt kryds/midterforhindring	16
5.2.4 Projektering	17
5.3 Software (Magnus, Jonas, Christian & Oliver)	18
5.3.1 Pathfinding	18
5.3.2 GUI	18
5.4 Hardware (Mads & William)	19
5.5 Test	21
5.6 Delkonklusion	21
5.6.1 Billedebehandling	21
5.6.2 GUI	21
5.6.3 Opsamling/aflevering	21
5.6.4 Roadplanner	21
6 Follow-up og konklusion	21
7 Arbejdsfordeling	22

<i>INDHOLD</i>	3
8 Bibliografi	23
A Risikoanalyse	24
B Krav	25
C Test cases	27
D Projektplan	41

1 Indledning (William)

1.1 Introduktion

En fiktiv virksomhed har brug for en autonom robot, der kan opsamle goldbolde fra en goldbane, og har udskrevet en konkurrence hvor en række virksomheder - grupper - kan demonstrere deres prototyper.

Som prototype skal der udarbejdes en autonom robot, som kan opsamle bordtennisbolde.

Gruppen har igennem kurset designet og konstrueret en robot, der opfylder de givende krav og denne, samt arbejdsprocessen, dokumenteres i rapporten. Som følge af målstætningerne for kurset er der lagt særligt meget fokus på projektstyrelse og værktøjerne anvendt dertil.

I rapporten dokumenteres ligeledes analyse og design ud fra de specificeret krav. Dertil demonstreres det i et implementeringsafsnit, hvor der er lagt fokus på den faktiske implementering i dette projekt. Afslutningsvis konkluderes problemstillingen med perspektivering til processen og projektstyringen.

1.2 Problemstilling

I løbet af kurset skal gruppen designe og konstruere en autonom robot der er i stand til at opsamle bordtennisbolde. Systemet skal kunne identificere bolde og forhindringer, navigere igennem en bane hvor boldenes og én forhindrings position er ukendt, opsamle boldene, og aflevere dem gennem en af to udgange, hvor at det ene mål er mindre end det andet. Målet er at have en robot der opfylder alle krav i den endelige konkurrence og på kortest mulig tid - maksimum otte minutter - kan samle alle ti bolde uden nogen indgreb fra gruppens side.

Ydermere skal projektet planlægges og forløbe på en hensigtsmæssig måde, hvor gruppen anvender medlemmernes evner erfarede fra studiet samt de projektstyringsværktøjer gruppen finder passende fra CDIO-projektet, med henblik på at forbedre gruppens evne til at gennemføre ingeniørmæssige problemer efter CDIO-konceptet.

2 Status over projektet og produktet (Christian & Jonas)

2.1 Projektplan

Den endelige projektplan er udarbejdet løbende i Microsoft Project, og denne kan findes i appendiks [D](#) som et skærmbillede af et Gantt-diagram.

2.1.1 Projektledere

Projektlederne har igennem projektet været Jonas Thomsen (s174867) og Christian Mark (s164833). Det overordnede projekt er planlagt i programmet Microsoft Project.

2.2 Endelig Status

Projektnavn	Golfboldopsamler
Dato	20/06/2019
Udført af	Christian Mark (S164833)
Modtager	Stakeholders og læseren af rapporten

Den endelige helbredstilstand for projektet endte med risikabel, enkelte elementer stadig mangler (se afsnit [2.2.1](#)).

Det endelige projekt er endt ud med en robot, der kan køre rundt inde på banen og samle bolde op med en støvsuger-lignende opsamlingsmekanisme. Robotten vil tage bolde der ligger på banen, oppe

Generel helbredstilstand for projektet

	On target
*	Risikabel
	Fare!

af bander og i hjørnerne. Robotten vil forsøge at tage bolde, der ligger opad banderne og i hjørnerne på en "sikker måde", hvorimod at de andre bolde, der ligger mellem krydset og banderne vil blive taget "direkte". Når alle boldene, der IKKE ligger i krydset, er taget ville robotten køre til den lille "udgang", derefter forsøge at vende så udgangen på støvsugeren er ud mod målet og afgive boldene. Kalibrering og indstilling af parametre kan før kørslen opsættes via det udviklede GUI, og ændringerne kan ses "live" på et video-feed.

2.2.1 Opgaver der blev ikke nåede at blive lavet færdige

Gruppen har hængepartier i den færdige implementering med speciel henblik på, at få robotten til at tage bolde inde i krydset (uden at røre/skubbe krydset), og i at gøre præcisionen for at aflevere boldene bedre. Dette skyldtes, at der var for mange afhængigheder som "roadplaneren" ventede på var færdig implementeret og den endelige "GUI og integration" blev derfor forsinket. Derudover havde gruppen store problemer med, at der i den endelige integration opstod mange problemstillinger, som typisk resulterede i, at den originale arbejdsgruppe skulle lave ændringer til noget, der ellers var erklæret "færdigt". I værste tilfælde skulle en helt ny version/strategi findes for en given problemstilling.

2.2.2 Konklusion på statusrapport

Gruppen har fået lavet en "minimums-implementering" af det færdige produkt, da robotten ikke helt formår at kunne tage bolde inde i krydset på en "sikker" måde (uden at ramme ind i det). Robotten kører dog banen igennem konsekvent og henter alle boldene, ved optimal placering af krydset (i midten). Så det er gruppens overbevisning, at robotten når at tage alle boldene på konkurrence dagen og aflevere dem indenfor de 8 min. givet ved konkurrencen.

3 Proces (Christian & Jonas)

3.1 Metoder og værktøjer

Der er blevet anvendt følgende værktøjer

- Versionering - GitHub med Git-GUI'erne Tower og GitKraken
- Build automation - Gradle til at bygge projektet på både robotten og computeren
- Kommunikation - Der blev anvendt Facebooks Messenger til at kommunikere
- Dokumentdeling - Google Drive er blevet brugt til at dele mindre filer. Firefox Send er blevet brugt til større filer
- Projektstyring - YouTrack opsat på egen server til kanban board, Microsoft Project til at holde styr på tidsplan
- Skriveredskaber - Rapport er skrevet i LaTeX i Overleaf, Javakode er skrevet i JetBrains IntelliJ.

I gruppens første aflevering blev der lavet et "Project Scope Statement", hvor projektets formål, beskrivelse, gruppe kontrakt, ønskede resultat, acceptkriterium, begrænsninger og en kommunikationsplan, som kortlagde den måde gruppen internt holdte møder på lavet. Kommunikationensplanen indeholder 4 forskellige slags møder, hvor et af dem er styregruppemøderne, som blev holdt med forelæserne: Bjørn Klint Christensen og Christian Budtz udfra en opdateret projektplan og statusrapport. Herunder ses kommunikationsplanen:

Hvad	Ansvar	Publikum	Hvordan	Hvornår
Styregruppemøder	Team	Styregruppe	Møde	27/2 27/3 1/5 12/6
Morgenmøder	Projektleder	Team	Møde	Begyndelse på hver arbejdsdag
Statusmøde	Team	Team	Møde	Afslutning på hver arbejdsdag
Projektledermøde	Projektleder	Undergrupper i projektteam	Møde	Afslutning/begyndelse på hver periode

Tabel 1: Kommunikationsplan

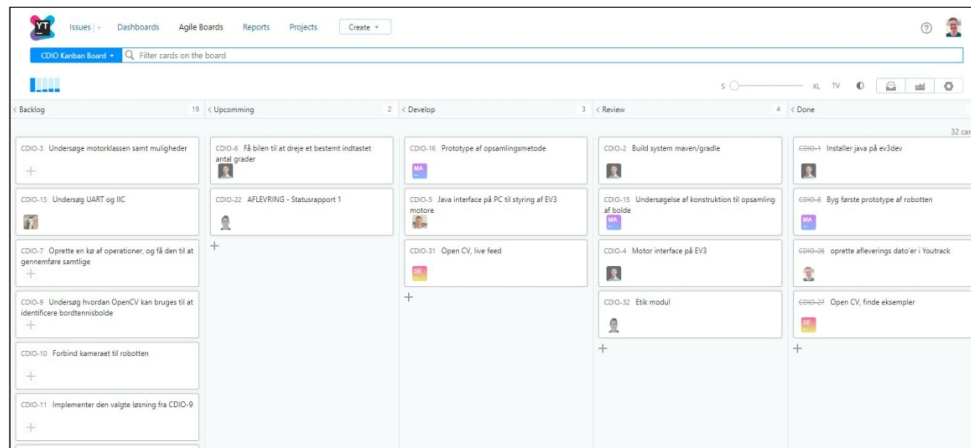
De sidste 3 møder blev afholdt internt i gruppen og undergrupperne, som der så blev skrevet referat af, så projektlederne kunne ligge timer ind i et Gantt-diagram og holde øje med fremskridt/tidsplanen. Derudover blev en kageliste holdt opdateret ved morgenmøderne så folk, der kom for sent, blev skrevet på og skulle give kage.

Til projektplanen blev der brugt et Gantt-diagram til tidsestimater, tidsforbrug, milestones og afhængigheder. Gantt-diagrammet blev lavet i Microsoft Project og løbende opdateret en gang om ugen. Grundet programmets mange funktionaliteter og størrelse var der en stejl læringskurve og det tog derfor noget tid, at få op og køre og vise noget relevant. Det viste sig dog at være tiden værd, da det var et rigtigt godt diagram, at kunne vise til statusmøderne og til at skabe et overblik over projektet med.

Til at holde styr på hvor langt delgrupperne var i deres arbejdsopgaver brugte gruppen et værktøj kaldet "Youtrack" [\[4\]](#) som supplement til Gantt-diagrammet. Det er et management værktøj til, at få et overblik over opgaver og evt. datoer (for afleveringer og deadlines). Det kræver dog, at folk aktivt går ind og opdatere deres fremskridt og nogle af opgaverne var meget svære, at dele op i mindre del-opgaver. Så der kom f.eks. meget store opgaver der hed "Prototype af opsamlingsmetode" eller lignende, som sagtens kunne have været delt op i del-tasks. Så værktøjet blev primært brugt til de software opgaver, der var i de 3 hoved-software grupper/opgaver:

- OpenCV
- EV3 opsætning, GUI og build system
- Road planning

På nedenstående billede ses et screenshot af gruppens Youtrack "Agile Board", hvor opgaver kan lægges i 5 forskellige "kasser- dette antal kan dog ændres og redigeres.



Figur 1: Screenshot af YouTrack

For at sikre at der var taget hånd om alle de opgaver, der kunne gå galt, er der løbende blevet fortaget en risiko analyse, hvor at der er blevet overvejet, hvilke risikoer der har været, og hvor meget det ville have betydet for projektet og tidsplanen, som det så ud i daværende øjeblik. Til dette har gruppen brugt "Google Docs" således at alle gruppe medlemmer havde mulighed for at tilføje ting som de tænkte på kunne komme til at påvirke projektet.

Se alle vores registrerede risici i appendix [A](#)

3.2 Løbende projektændringer

Gruppen havde i starten planlagt, at kameraet skulle sidde på robotten og OpenCV skulle køre på en Raspberry Pi som så sendte kommandoer til EV3 hjernen over en kablet kommunikations forbindelse. Dette blev dog ændret, da der blev skaffet nogle kamerastativer som gjorde, at kameraet kunne hænge over banen og OpenCV delen kunne derfor køres på en stærk pc, og så sende kommandoerne trådløst til EV3 hjernen via RMI over WiFi.

En anden projektændring var strategien over, hvordan robotten skulle køre rundt på banen. Strategien og koden har været igennem mange iterationer og har skulle tilpasses ved integrationen med OpenCV. Dette hang også sammen med at der prototypefasen af opsamlingsmekanismen blev foreslået to forskellige designs, som havde hver deres præcision og derfor hver deres strategi.

3.3 Projektstyring i projektet

Gruppen valgte at afholde to møder om dagen. Et morgenmøde hvor arbejdsopgaver blev fordelt og et opfølgingsmøde om eftermiddagen hvor der blev samlet op på dagens arbejde. Desuden blev møderne også anvendt til at diskutere diverse problemstillinger og til at skabe overblik over de forskellige delopgaver og projektet som helhed. Gruppen blev dog hurtigt enige om, at det med at diskutere problemstillinger til møderne hurtigt kunne blive til en diskussion mellem 2-3 gruppemedlemmer, mens de andre bare sad passivt og ikke kunne deltage. Så det styrede projektlederne, ved at tale med delgrupperne hver for sig og tage beslutning om diskussionen bare skulle være mellem delgrupperne eller skulle være med alle som "publikum".

Derudover har det været vigtigt med projekt- og tidsstyring da gruppen har bestået af medlemmer fra 3 studieretninger, med hvert deres skema. Dette har medført, at gruppen har skulle omstrukturere timer, der ellers har skulle ligge onsdag eftermiddag, og lægge dem ind andre steder. Gruppen startede derfor 3-ugers perioden d. 3/6 og har planlagt en arbejdsdag i alle weekenderne i 3-ugers perioden.

4 Analyse og design

(William indtil 4.3) Den stillede opgave er både omfattende og teknisk kompleks. Det er derfor nødvendigt at opdele projektet i mindre dele, så det er muligt at definere de opgaver, der skal løses. Denne tilgang gør det også muligt at lave et bedre estimat af hvor lang tid projektet vil tage, og hvordan de forskellige dele skal takles.

De vigtigste problemstillinger er:

- Det skal være muligt at genkende og lokalisere boldene og forhindringerne, og behandle denne data på en måde så systemet kender disses position.
- Ud fra boldenes og forhindringernes position skal en rute kunne findes, der sikrer at robotten på mindst mulig tid når alle bolde uden at ramme banens kanter og afleverer boldene i det ene af banens to mål.
- Robotten skal autonomt kunne opsamle, opbevare og afgive bordtennisbolde inden for konkurrencens regler.

4.1 Krav

Der er blevet udarbejdet en kravliste i begyndelsen af projektetforløbet. Efterhånden som projektet skred frem blev flere ting dog tilføjet til denne liste i takt med at nye problemer opstod eller blev identificeret og evt. nye krav/delopgaver skulle laves, som det gør i de fleste agile udviklingsprocesser.

Kravene kan findes i appendiks [B](#)

4.2 Afgrænsninger

Der skal så vidt som muligt anvendes de udleverede komponenter for at undgå at for mange komponenter og elementer skal udvikles, hvilket begrænser mulighederne i forhold til robottens konstruktion. Dette medfører blandt andet at der ikke kan bruges mere end fire motorer og fire sensorer, som var udleveret med legosættet. Dette omfatter dog ikke strukturelle elementer, da disse kan laves forholdsvis let.

De udleverede sensorer anvendes ikke; robotten skal være i stand til at navigere udelukkende ved brug af det udleverede kamera og softwaren skrevet hertil. Dette er som følge af, at det blev vurderet at opgaven kunne løses uden LEGO-sensorer, og at det derfor ikke ville give mening at bruge tid på at designe og implementere disse.

4.3 Billedebehandling (Ali & Sebastian)

4.3.1 Farvespektrum

Generelt er der to veje at vælge imellem med hensyn til farvespektrum, værende HSV (Hue, Saturation, Value), eller BGR (Blue, Green, Red), hvor hver har sine fordele og ulemper.

I bund og grund kan der ikke på forhånd siges hvorvidt et farvespektrum kan vælges fremfor noget andet, men i stedet afhænger det af den anvendte metode, samt en række andre faktorer. Dette er ensbetydende med at samtlige metoder bør afprøves i de omgivelser man måtte være i, før en endelig beslutning tages.

4.3.2 Bolde samt robottens placering

Tilgange til detektering af bolde samt robottens placering kan hovedsageligt inddeles i to kategorier, værende kantdetektion samt at tage udgangspunkt i cirkelsegenskaber. Kantdetektion kan udføres

ved hjælp af 'Edge detection' og 'Canny Algorithm' [6]. 'Edge detection' har til formål at detektere de kanter et objekt måtte have, ved at kigge efter markante farveændringer i et billede. Sidstnævnte udføres ved at beregne billede graduering. Dog er det problematisk at anvende edge detection på et graduering billede, da det kan indeholde for meget støj. Her kommer 'Canny Algorithm' ind i billedet. Sidstnævnte er en algoritme, der udfører samtlige operationer af gangen, hvor en canny process består af blandt andet støjreducering (gaussian filter), beregning af billede graduering og kant-forfølgning. Sammenlignet med at blot anvende kant-detektering, så er Canny algoritmen en mere lovende process, som kan resultere i en bedre detektion af eksempelvis cirkel formede objekter.

I den anden boldgade, findes en anden tilgang til detektering af cirkulært formet objekter, nemlig ved at kigge efter cirkelns egenskaber. Sidstnævnte kan udføres ved hjælp af Circle Hough Transform (CHT). Her determineres, hvorvidt et objekt er en cirkel på baggrund af henholdsvis centrum (a, b) samt radius r . Parametre så som minimum radius og maksimum kan gives med i metoden, således cirkler med en radius indenfor et bestemt interval bliver detekteret, mens resten bliver kasseret.

Bolde

CHT kan anvendes, således kun bolde beliggende indenfor radiusintervallet med dens minimum og maksimum mål detekteres. Derudover kan minimumdistancen mellem cirkulære objekter indstilles, således der ikke detekteres cirkler inde i cirkler. Dette kan til fordel anvendes, således bolde som ligger alt for tæt på hinanden kun bliver detekteret som enkelte bolde, og ikke bolde i bolde.

Robottens placering

Alt det nævnt tidligere er også gældende for detektion af robottens placering, da samme metode tænkes at anvendes, værende Circle Hough Transformation. Der kunne tænkes at placere to cirkler med forskellige farver på toppen af robotten, således robottens retning også kan genkendes.

4.3.3 Banens hjørnepunkter samt kryds

Der anvendes to tilgange til detektering af banens fire hjørnepunkter samt forhindringen; hjørnedetektion og konturdetektion. Hjørnedetektion kan udføres ved hjælp af 'Harris Corner Detection (HCD)'. Algoritmen har til opgave at identificere de interne hjørner af et billede. Billedets hjørner bliver dermed opfattet som værende områder hvor der er tale om markante intensiteter i gradueringen i alle mulige dimensioner samt directioner. HCD er dermed en metode til at ekstrahere hjørner samt træk fra et input billede.

Konturdetektion kan udføres ved hjælp af 'Contours'. En kontur linje indikerer en buet linje repræsenterende en grænse af samme værdier eller samme intensiteter. Kontur ser på overordnede skel/skillelinje af en figur, og kan defineres som værende nærliggende kurver, som fås fra kanter og afbildning af den overordnede skel/skillelinje af figurer. Dermed kan man se hvor mange kanter der er forbundet i hver kontur, og få isoleret dem man er interesseret i. Dette er især nyttigt, da vi så kan isolere firkanter med 4 sammenhængende linjer, og krydset med 12 sammenhængende linjer.

Sættes det tidligere nævnte i perspektiv, så kunne det være en idé at anvende enten HCD eller contours til lokalisering af banens hjørnepunkter samt midterforhindringen. Umiddelbart er der ikke en metode, som fra førstehåndsindtrykket ser mere egnet ud end den anden, og dermed anses det for at være et spørgsmål som bliver besvaret senere i processen baseret på afprøvning af begge.

4.4 Hardware (Mads & William)

Fra projektets start var det usikkert hvilken opsamlingsmetode der ville egne sig bedst. En af de største udfordringer identificeret er at bolde der ligger op af banens og krydses kanter - eller hjørner - skal

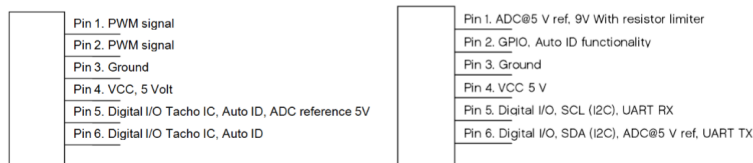
kunne fanges uden at robotten berører forhindringerne. Dette er svært at undgå da de fleste mekaniske løsninger skal røre boldene og, ofte, er mekanismen både bredere og dybere end boldene for at kunne opsamle disse, hvilke skaber problemer for en sådan bold.

En løsning på nogle af problemerne var at bolde i stedet kunne blive opsamlet ved brug af en støvsuger-agtig mekanisme. Der var dog tvivl om, om denne løsning ville fungere i praksis så det blev besluttet at to prototyper skulle udvikles, hvorfra den anden var et mere konventionelt aggregat, sideløbende så gruppen havde et alternativ hvis den præferable løsning skulle vise sig at være uladsiggørlig.

Den endelige løsning blev først endeligt fastlagt senere i projektet da støvsugeren viste sig at opfylde dets mål. Dette medførte at der måtte laves ændringer i den fysiske konstruktion af robotten senere i projektet og at en række nye problemer skulle løses.

Efter de første prototyper havde demonstreret at princippet virkede opstod der to hovedproblematikker. De to problematikker var at blæseren (senere blev en ekstra blæser tilføjet) skulle have strøm og skulle kunne tænde og slukkes. Desuden skulle den endelige udformningen af kammeret og opsamlings- og afleveringsmekanismen findes da de eksisterende løsninger ikke opfyldte de krav, der var blevet stillet. Grunden til ændringen af formen var, at boldene kunne lægge sig i spænd inde i kammeret, og at det derfor i nogle tilfælde ikke var muligt at aflevere alle bolde igen. Dette blev løst ved at designe et smalt kammer med en sliske i bunden, således boldene lå på en række i tre lag.

Et problem der hurtigt blev opdaget var at EV3en ikke er i stand til at levere mere end 750 mA (1 A peak) ved 9 V på motorportene. Dette var ikke nok til at drive motorerne på de blæserne gruppen havde til rådighed. Det var en mulighed at bruge en boost-konverter så blæserne kunne få den nødvendige spænding, men dette ville forsage et fald i strømstyrke. Og i alle tilfælde var det usikkert, hvorvidt at EV3ens batteri ville kunne holde i de krævede otte minutter ved denne belastning. [1]



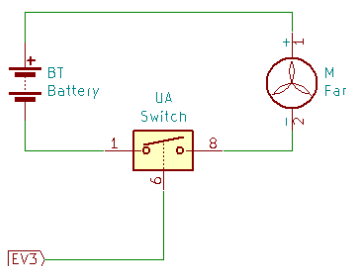
Figur 2: Pinout for 6P6C motor- og sensorportene.

EV3-computeren har en lang række af I/O porte, herunder kombinerede porte til kommunikationsprotokollerne UART og I²C. Størstedelen af I/O portene kan tilgås via EV3-blokkens 6P6C-kontakter, hvoraf der er to varianter: en for motorer og en for sensorer. Pinout for disse ses i ovenstående figur. Af de nævnte porte er UART/I²C langt den mest versatile, eftersom det er muligt at sende beskeder og skrive til registre over denne forbindelse. Det er derfor muligt at give komplekse kommandoer til den tilsluttede komponent. [1]

Til gengæld er disse mere komplekse end analoge eller simple digitale signaler og kræver at der anvendes en eller flere komponenter før at signalet kan tydes, hvilket øger projektets kompleksitet. For at UART eller I²C kan anvendes skal der enten bruges en microcontroller, der understøtter mindst en af disse protokoller, eller også skal et sender/modtager kredsløb designs - sandsynligvis i form af en finite state machine. Af de to muligheder er det langt lettere at anvende en microcontroller, da at man så ikke skal designe et digitalt system.

Såfremt det eneste krav er, at der skal kunne tændes eller slukkes for en kontakt, der styrer strømtilførslen til en eller flere motorer (se ovenstående figur), giver det ikke nødvendigvis mening at anvende de mere avancerede protokoller. Her vil det være mere hensigtsmæssigt at bruge en enklere løsning - ideelt et binært signal.

Det tætteste EV3'en har på et on/off signal, uden at det er nødvendigt at modificere firmware, er motorportenes PWM signaler. Motorne fungerer essentielt ved at én af PWM signalerne sættes til



Figur 3: Minimumsfunktion krævet af kredsløbet.

lav hvilket får motoren til at dreje i strømmens retning. Ved at sætte PWM signalets duty cycle kan hastigheden justeres. Disse signaler kan sådan set bruges til at styre en CMOS transistor, da de er lette at sætte fra software og resulterer i et gate-source potentiale, der er tilstrækkelig stor til at åbne for nogle MOSFETs given en source spænding på 12 V.

4.5 Software (Magnus, Jonas, Christian & Oliver)

I det følgende gennemgås software på henholdsvis robotten og ruteplanlægningsmodulet, der eksekveres på en ekstern computer. Det er nødvendig at lade en anden computer udføre de relativt tunge beregninger, da robotten er begrænset af en mindre regnekraft (ARM-processor på 300MHz). Det var også tiltænkt at billedbehandlingen kunne accelereres ved at bruge computerens GPU, hvilket OpenCV understøtter i form af CUDA APIen, og derfor gav det også mening at udvikle på computeren fra start af - også selvom at dette i sidste ende ikke blev implementeret. [2](#) [3](#) [7](#)

4.5.1 Robottens applikation

Det er væsentligt for vores program at lade den eksterne computer udføre tunge beregninger, således at robottens eneste ansvar er at styre dens komponenter, som består af støvsuger, vibrator og to hjul - alle styret gennem EV3ens motorporte. Robottens styresystem er LeJos, som har et bibliotek til at styre motorporte, sensorporte, og sågar robotnavigationen. Vi har dog blot valgt at anvende at styre motorportene med LeJos. Gruppen har valgt at anvende Java, således at vi kan kommunikere med robottens program via RMI (Remote Method Invocation), der er en ældre, metodeorienteret Java-teknologi til at kommunikere mellem en og flere applikationer. [8](#) [5](#)

Da der foregår netværkskommunikation, fandt vi det nødvendigt, at robotten får hele instruktions-sæt fra computeren. Indledningsvis kaldte vi fra computeren kommandoer som *leftMotor.forward()* og dernæst *rightMotor.forward()*. Dette gjorde, at højre motor startede umiddelbart efter venstre motor, således at den drejer mod højre først.

Robotten er udstyret med en forward- og turnmetode, som begge gør brug af LeJos' RegulatedMotor.rotate. Ved at kende hjulets omkreds, kan man beregne, hvor langt robotten bevæger sig ved at rotere hjulet 360 grader. Dette kan derfor nemt anvendes til at bevæge robotten frem og beregne, hvor langt hjulene skal dreje ved rotation af robotten. For at robotten kan roteres om egen akse, hvilket er en nødvendighed for en robot af denne størrelse, skal det ene hjul køres bagud, mens det andet hjul kører fremad.

4.5.2 Ruteplanlægning

Til at planlægge robottens kørsel (roadplanner) er det forventet, at følgende input modtages fra kameraet i form af koordinater for følgende elementer:

- Banens hjørner
- Krydsets hjørner
- Robottens front- og bag punkt
- Bolde

Roadplanneren skal modtage information og give en instruktion til robotten, hvorefter den afventer at skulle beregne en ny instruktion. En instruktion er defineret til at indeholde en vinkel i grader eller en afstand.

I programmets controller, der modtager elementernes koordinater, bliver der efterspurgt en instruktion fra roadplanneren, der sendes til robottens applikation via RMI. Robotten aflæser instruktionen og udfører den pågældende instruktion.

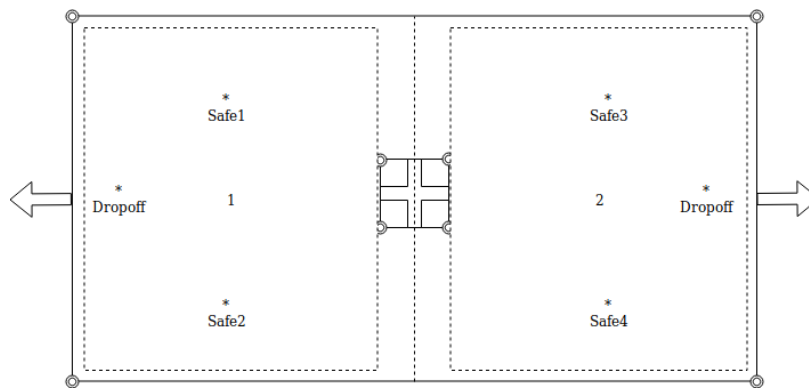
Roadplannerens algoritme fungerer således, at den inddeler banen i to kvadranter, skåret ved forhindringens x-værdi. Da instruktionerne typisk går til den nærmeste bold i samme kvadrant som robotten, kan der opstå problemer med at tage en bold, der er farligt tæt på forhindringer. Dette er løst ved at danne et safety-area i hver kvadrant, således at bolde kan være sikker eller usikre. Safety-area er en konstant/safety-margin mindre end kvadrants dimensioner. Mens en sikker bold bliver kørt direkte til, bliver en usikker bold håndteret ved at projekte et punkt på det tætteste punkt inden for safety-area. Robotten vil derfor rejse direkte til det projekteret punkt, og vil dernæst dreje, så den står vinkelret på banen, køre direkte ind til bolden og bakke tilbage. Det er vigtigt at bakke tilbage, da robotten, hvis den er uden for safety-area og vil vende, kan komme til at berøre en forhindring.

Bolde i hjørner og nær krydset uden for safety-area bliver håndteret ved at finde vinklen mellem centrum af kryds/hjørnet og bolden, og projekte et punkt på det nærmeste sted inden for safety-area.

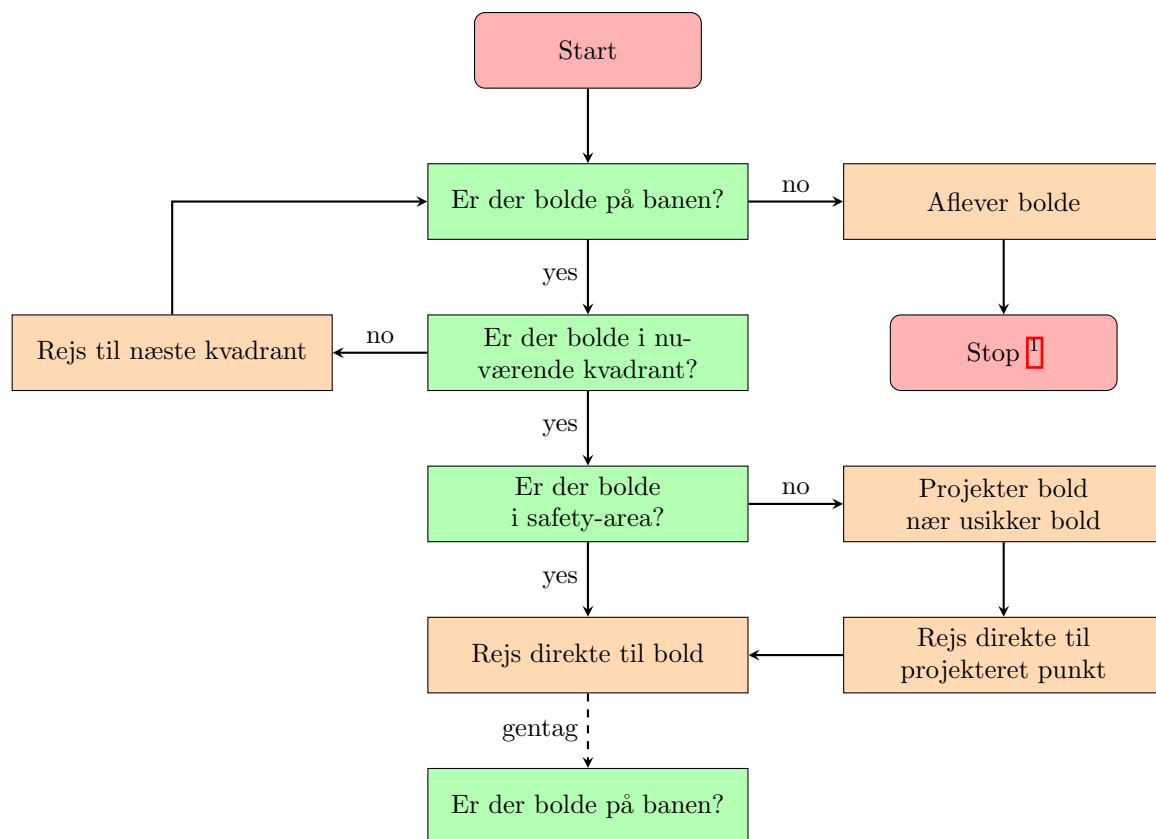
Når der ingen bolde er i en kvadrat, rejser robotten, såfremt der stadig er bolde på banen, til næste kvadrant. Hver kvadrant består af to safe-points, som skal ses som punkter, robotten kører til, når næste kvadrant besøges. Robotten starter med at køre til det nærmeste safe-point i dens nuværende kvadrant, hvorefter den rejser til det nærmeste safe-point i næste kvadrant. På denne måde undgår vi, at robotten rammer forhindringer under denne kørsel.

Når der ingen bolde er, laver robotten et tjek for, om der skulle gemme sig bolde bag robotten. Hvis det ikke er tilfældet, køres der til det mindste mål ved at projekte et punkt ud fra dette, vende sig om og aflevere bolde.

Herunder ses et banen visualiseret og et flowchart over ruteplanlæggerens algoritme.



Figur 4: Bane opdeling i kvadranter og krydset

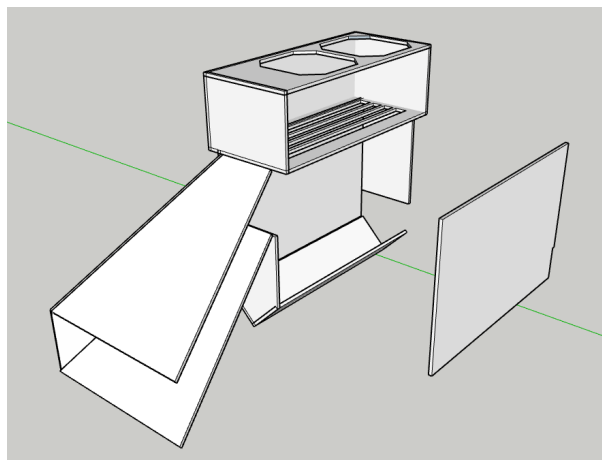


¹inkluderer check for, om der skulle gemme sig bolde bag robot og give lyd, når robotten er færdig.

5 Implementation

5.1 Vakuumboldopsamlingsaggregat (Mads, Jonas & William)

Efter flere prototyper blev det fastlagt, at der skulle anvendes en tredelt konstruktion bestående af en munding, et sugekammer og et opsamlingskammer (Se nedenstående figur). Dette design havde den bedste samlede kombination af egenskaber af de design, der var blevet testet.



Figur 5: 3D-billede af robotens konstruktion.

Aggregatet består af to 12 V servere-kølere. Disse skaber undertryk i sugekammeret, som er monteret på opsamlingskammeret. Sugekammeret og opsamlingskammeret er opdelt af et gitter. Dette agerer filter og sørger for, at boldene ikke kan blokere blæserne og fordeler trykket ud i kammeret.

Opsamlingskammeret er lidt bredere end en bold. Dette gør, at der kun kan ligge bolde i en række i kammeret, således de ikke sætter sig i spænd og blokerer for hinanden. Kammeret har en trekantet bund, som også hjælper med at holde boldene på en række.

På enden af opsamlingskammeret er munden monteret. Munden er en smal tragt formet således, den har en god balance mellem sugeevne og bredde.

I den bagerste ende af opsamlingskammeret er der monteret en låge. Denne låge holdes lukket af undertrykket i kammeret, når blæserne er tændt, således at luftens eneste indgang er igennem munden. For at sikre at den lukker tæt er der sat tape udover lågen hvilket agerer som et segl. Når blæserne slukkes er det muligt ved hjælp af en mellemstørrelse LEGO-motor at åbne denne klap. Derefter triller boldene via en skinne ud af kammeret med en vinkel, således at der kommer nok fart til, at boldene bliver præcist projekteret ud af robotten.

Hvis ikke alle bolde kommer ud af opsamlingskammeret bruges den vibrator, som er monteret udenpå kammeret. Denne ryster kammeret således boldene løsnes og ikke længere blokerer for hinanden, og derfor løber frit ud i en lige linje.

Opsamlingen har en rækkevidde omkring 5-10cm afhængigt af hvor bolden befinder sig, da luften vil blive suget hen over bolden og derfor ikke trække i bolden. Denne effekt er også til stede ved krydset, men den er minsket da bolden er højere end krydset, og luften der kommer fra over krydset stadig skubber til bolden.

På trods af reduktioner af kammerets størrelser, er der stadig nogle problematikker med hensyn til robotens dimensioner. Mest mærkbart er robotens højde, hvilket har konsekvenser i forhold til billedebehandlingen. En stor robot vil skygge for dele af banen og har vist sig at skabe problemer med genkendelse af forhindringer og bolde. Det er derfor muligt for robotten at dække for bolde og derved

tage en ikke optimal rute, da de nærmeste bolde ikke kan genkendes i software, hvilket vil medføre at robotten under visse omstændigheder skal køre længere og bruge mere tid end nødvendigt.

De to kamre er laserskåret i akryl, og sammensat med limpistol. Dette giver et robust og lufttæt kammer. Som kan bredde rystelserne fra vibratoren bedst muligt, grundet sin stivhed.

5.2 Billedebehandling (Ali & Sebastian)

Billedebehandling udgør en stor og essentiel del af projektet. For at få en fungerende robot, som med et kamera skal kunne lokalisere bolde, forhindring og robotten, er det nødvendigt at anvende en del billedebehandling. Til billedebehandling anvendes biblioteket OpenCV, implementeret i programmeringsproget Java. Det er denne del der giver data til vores navigering.

5.2.1 Farvespektrum

I starten blev farvespektrummet BGR anvendt til henholdsvis detektering af boldene, robotten, banens hjørnepunkter samt kryds. Hvad angår boldene og robotten, så blev der afprøvet med forskellige BGR samt HSV farverkombinationer, både lyse og mørke, og ikke mindst forskellige papirtyper og størrelser. Hvad angår banens hjørnepunkter samt kryds, så er der blevet afprøvet forskellige farverkombinationer/indstillinger. Grundet skiftende lysforhold, som drastisk reflekterede sig over hvorvidt henholdsvis boldene, robotten, banens hjørnepunkter samt kryds blev detekteret eller ej (BGR farven vil blive anset for at være lysere eller mørkere end de valgte parametre), blev HSV betragtet som en fin afløser. Der er blevet lavet en række sliders, som tillader manuel manipulation af HSV indstillingerne, således justering på HSV parametrene til nye omgivelser herunder konkurrencedagen var muligt. Det kan konkluderes på baggrund af de udførte test med HSV, viste resultaterne at HSV var mere lovende end oprindeligt valgt løsning værende BGR.

5.2.2 Bolde samt robotens placering

Til detektering af bolde, indenfor et forudbestemt farvespektrum, gøres der brug af `inRange` funktionen i OpenCV. Implementeringsmæssigt, så kan en farve defineres ved hjælp af `inRange` metoden samt `scalar` (vektor, bruges til passering af pixel værdier). Metoden tager imod en `Mat`, en `scalar` forhenholdsvis minimum- samt maksimumsværdier i HSV, og en `output Mat`.

```
Core.inRange(src, new Scalar(185, 185, 185), new Scalar(255, 255, 255), thresh);
```

Til detektering af bolde, gøres der brug af `HoughCircles`. `HoughCircles` er en funktion, som først transformerer billedet, og derefter, med udgangspunkt i nogle givne parametre, analyserer billedet. Disse parametre er metoden som ønskes anvendt, angivelse af grænseværdier til detektering af cirkelens kanter (Canny Edge hvis `HoughGradient` anvendes) samt grænseværdi til detektering af cirkelens centrum (jo lavere værdi, desto flere falske cirkler/falsk positiv), minimum distancen mellem centrum (x, y) koordinater, samt minimum- og maksimumradius på de cirkulære objekter som ønskes fundet. Følgende er et eksempel på en `HoughCircles` metode:

```
Imgproc.HoughCircles(threshold, image, method, dp, minDist, Param1, Param2, MinRadius, MaxRadius)
```

Fejlkilder

Somme tider har der været problemer med at en bold som er gemt i skyggen ej bliver fundet. Derudover er der blevet oplevet at robotten 'kører' over en bold, i og med at der er et mellemrum mellem karrosiet og papdækket. Lige nu sorteres der i BGR hvad angår boldene, dog er der blevet lavet en

metode til sortering i HSV.

Robottens placering

Lokalisering af robotten på banen finder sted ved brug af tidligere anvendte teknikker, mere specifikt HoughCircles. Idéen er at ved hjælp af to cirkler farvet forskelligt og placeret på toppen af robotten, er det muligt at genkende robottens placering på banen. Implementeringsmæssigt så er der tale om et stykke kode, der køres to gange. Den kører en gang for hver farve, og afleverer til sidst to sæt koordinater for hver cirkel. Koden er magen til det stykke kode som anvendes til at finde boldene på banen, i det at i begge tilfælde er det cirkelformet objekter der ønskes detekteret. Ved at have en cirkel prædefineret som værende fremadretningen, kan cirklen med sidstnævnte retning placeres forrest i et array. Arrayets førsteplads vil altid være forbeholdt fremadretningen, hvilket vil gøre afkodningen samt determinering af robottens retning nemmere. De to farver som er placeret på robotten, er blå og grøn, grundet at disse virkede bedst under afprøvning.

Fejlkilder

Robot detektering er ret præget af lysforhold, forstået således at lys kan spille en stor rolle i forhold til detektering af robottens placering. En fejlkilde kunne være at beregningerne baseres på start/default værdierne, uden at der justeres i henhold de daværende omgivelser. Dette gælder henholdsvis start parametrene som er hardcoded i koden. Sidst men ikke mindst kan en forkert indstilling af start parametre i GUI'en også have indflydelse på detekteringen.

5.2.3 Banens fire hjørnepunkter samt kryds/midterforhindring

Til lokalisering af banen samt forhindringen er samme metode blevet benyttet, værende findContours. Sidstnævnte metode har bragt den største succes, eftersom at denne virker til at finde banen rimelig nøjagtigt, samt rimeligt konsistent. Den tidligere anvendte metode, værende cornerHarris, havde problemer med at konsistent lokalisere hjørnerne, og vil dermed hoppe fra og på hele tiden, hvilket ej er nogen holdbar løsning. Ulempen ved brugen af findContours er, at når robotten kommer ud til kanten, så vil konturen brydes, og dermed vil banen ej blive fundet længere. Det samme gøres gældende for cornerHarris, i og med at robotten dækker for hjørnerne, så vil banen ikke kunne lokaliseres i visse tilfælde, og i andre registreres forkerte hjørner og forkert data returneres. Inden findContours benyttes, så transformeres billedet til HSV, og thresholds sættes således, at der kun er banen samt krydset som kan ses. Årsagen bag brugen af HSV fremfor BGR skyldes oplevelsen af, at BGR kan få kanterne til at blive rundere og dermed ikke så skarpe, hvorimod HSV får gjort kanterne forholdsvis skarpe. Sidstnævnte er et must for at konturerne kan detekteres på en passende vis. Desuden benyttes dialate funktionen, som forstører banen samt krydset, således at kanterne er endnu skarpere.

```
Imgproc.findContours(Mat image, List<MatOfPoint> contours, Mat
    hierachy, Imgproc.RETR_TREE, Imgproc.CHAIN_APPROX_SIMPLE)
```

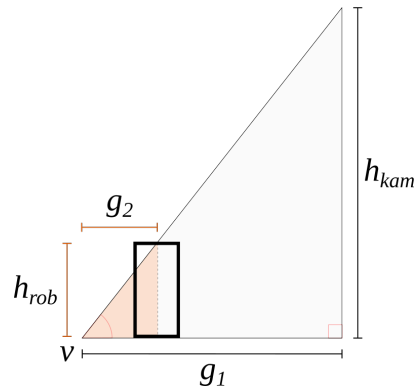
Ovenfor ses findContours metoden, som tager i mod et billede, en liste af MatOfPoint til lagring af fundne contours, en Mat hierachy, et mode som i det her tilfælde er RETR.TREE der konstruere et hierarki af de fundne contours, og sidst men ikke mindst ønskes en metode, her bruges CHAIN_APPROX_SIMPLE, hvilket sørger for at kun returnere hjørne punkterne fremfor en masse punkter langs linjerne. Grundet tidspres var det ikke muligt at gøre brug af hierachy, dog tænkes der at et hierarki dannes ud fra om en contour ligger inden i en anden contour, og dermed sørge for at på trods af at firkantede contours bliver fundet inde i banen, så ville det kunne ses i sin Mat hierachy. Ud fra de fundne contours kan der afgøres hvorvidt der er tale om et kryds eller en firkant, nemlig ved tjek af hver eneste contours længde, i og med at et kryds har 12 sammenhængende linjer, og en firkant har 4 sammenhængende linjer. Dernæst kan et minimum areal for contouren deklareres, hvilket hjælper med at undgå problemer i tilfælde af, at små firkanter bliver fundet.

Fejlkilder

Dækkes noget af banen til eller bliver der kun fundet tre ud af fire punkter, så vil detektionerne fejle og banen vil ikke kunne blive fundet. Dog formodes der, at banen eller krydset ikke bliver rykket, og dermed er det muligt at nøjes med et enkelt billede af banen ved start, og bygge de efterfølgende billede af boldene, robotten m.m. på det. Et scenarie som er muligt at komme ud i er at krydset eller banen bliver rykket i af robotten. Her kunne det tænkes at der tages billeder af banen hver gang robotten er i et "safe zone", således at det er muligt at fange ændringer i forbindelse med at banen eller krydset rykkes.

5.2.4 Projektering

Projektering var et problem vi stødte på i takt med at vores robot blev højere. Problemet ligger i at, når robotten kommer ud i kanten af billedet, så vil det fremstå, som at cirklerne på toppen af robotten, ligger udenfor banen. Vi er derfor interesserede i at projicere punktet vinkelret ned således, at vi får den reelle position. Problemet er skitseret på nedenstående billede:



Figur 6: Projektering af robot

Vi gør brug af de trigonometriske funktioner til at udlede, præcis hvor langt et stykke, vi skal forskyde vores punkt mod centrum, dette ses fra billedet. G_2 er det stykke, vi er interesserede i at vektorforskyde mod centrum, dette er gjort således: Højden på kameraet er givet på forhånd, samt højden på robotten. Vi starter med at beregne g_1 ud fra vores givne punkter, værende centrum af billedet samt det punkt vi ønsker at projicere. Derfra kan vi beregne hypotenusen vha. pythagoras, derefter bruger vi så cosinus relationen til at beregne vinkelen til g_1 :

```
var A = Math.toDegrees(Math.acos((b*b + c*c - a*a)/(2*b*c)));
```

Da det er en retvinklet trekant har vi nu alle 3 vinkler, og vi kan dermed beregne længden af g_1 - g_2 vha. sinus relation:

```
var realDistToPoint = (Math.sin(Math.toRadians(A)) * (cameraHeight -
    objectHeight)) / Math.sin(Math.toRadians(B));
```

Hvor A er vinkel til g_1 og B er vinkel til h_{kam} . Vi har dermed fundet frem til den projicerede afstand til punktet, og slutter af med at finde forskellen på startafstanden og den projicerede afstand, og derefter vektorforskyder punktet mod centrum.

5.3 Software (Magnus, Jonas, Christian & Oliver)

5.3.1 Pathfinding

Der vil i det følgende afsnit kort redegøres for væsentlige metoder i roadplanneren, som også er beskrevet i designafsnittet.

En af de væsentligste punkter, vi skal bruge til beregningen af ruter er rotationspunktet. Rotationspunktet befinder sig mellem hjulene, hvis position ikke er den samme som de to cirkler, der er monteret på robotten. Robottens rotationpunkt beregnes i roadplanneren vha. vektorregning. Da vi kender to punkter på robotten, kan vi finde vinklen og ved at kende afstanden fra bagpunkt til roterende punkt, kan vi projicere et nyt punkt. Dette punkt bliver udregnet med følgende metode, som anvender LeJos' Point2D-klasse.

```
public static Point2D getVectorEndPoint(Point2D startPoint, double
    angle, double magnitude) {
    var dx = magnitude * Math.cos(Math.toRadians(angle));
    var dy = magnitude * Math.sin(Math.toRadians(angle));

    return new Point2D(startPoint.x + dx, startPoint.y + dy);
}
```

En af de mest brugte metoder er direkte kørsel til en bold, typisk en bold inden for sikre områder. Herunder ses en del af koden til dette:

```
private void goToBallInSafeArea(InstructionSet instructionSet,
    Point2D robotPosition, Point2D ball) {
    instructionSet.setData(this.robot.getRotatingPoint(), ball,
        "Navigator: Safe ball");

    this.addTurnIfNeeded(instructionSet, robotPosition, ball);
    instructionSet.add(Instruction
        .forward(this.robot.getDistanceTo(ball)));

    this.state.resetState();
}
```

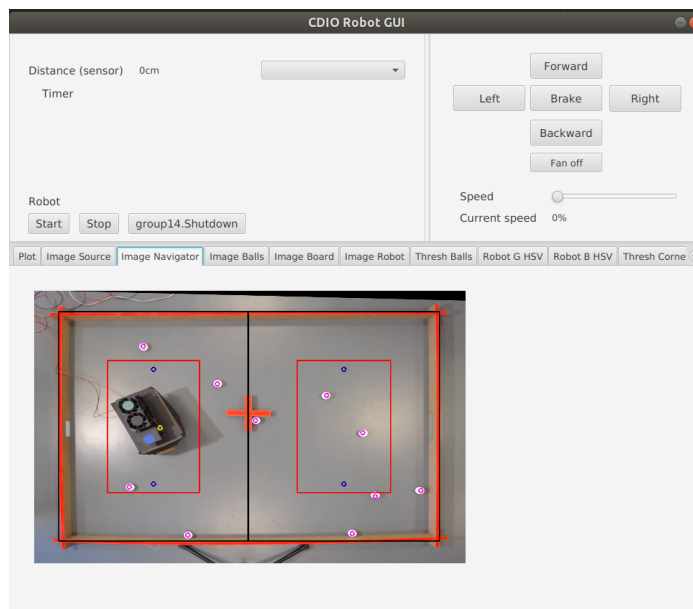
Koden til ruteplanlæggeren er i sagens natur omfattende, hvorfor der henvises til projektets repository på [GitHub](#).

Roadplannerens logik og algoritme kan findes i en controller på stien `CDIORobot/src/main/java/group14/navigator/Navigator.java`

5.3.2 GUI

Der er udarbejdet en GUI i JavaFX, så vi kan se det, programmet kan se. Det er et eminent værktøj til at debugge og rent faktisk få visuelt feedback for det, programmet ser som robotten, bolde og banepunkter.

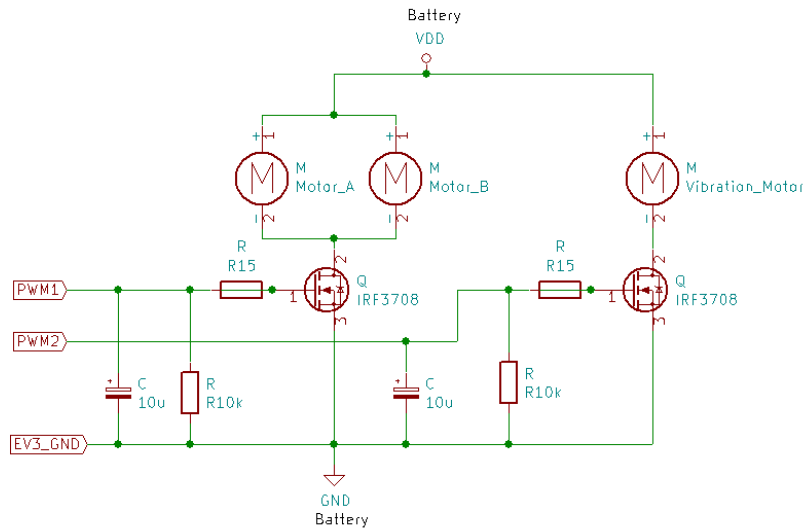
I GUI'en er det muligt at justere og kalibrere kameraet med homeografi og andre billedbehandlingsteknikker - on the go, således at programmet kan tilpasses forskellige scenarier. Der er derfor implementeret forskellige faner i GUI'en, herunder et view af robot (to views til forskellige farver på robot), bolde, forhindringen og bane. I en af fanerne vises punkter for safety-areas, safe-points, banens hjørner, robotten selv og forhindringen - denne kan anvendes til at kalibrere og justere billedet, så programmet kan genkende alle elementer.



Figur 7: Billede af GUI.

5.4 Hardware (Mads & William)

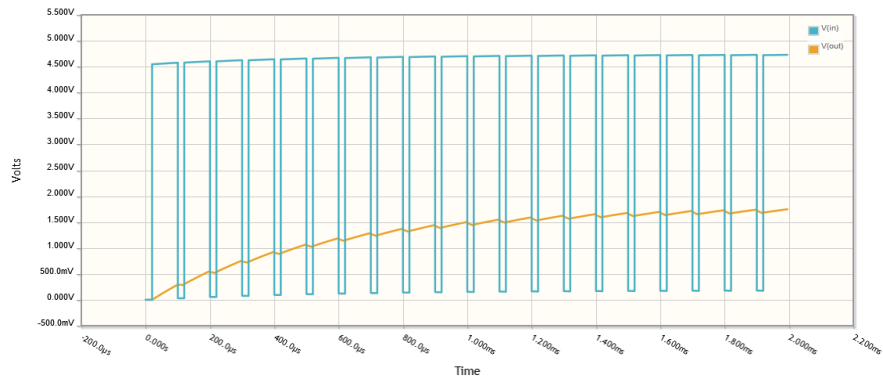
Robottens hardware består af 2 store LEGO-motorer, en mellem LEGO-motor, to blæsere, en vibrator og et 12V batteri. LEGO-motorene drives og kontrolleres fra LEGO EV3-computeren. De to blæsere og vibratoren drives af batteriet, men kontrolleres fra EV3-computeren. For at kontrollere de enheder der drives af det eksterne batteri, bygges der et kredsløb, som kan tændes og slukkes for motorene ved hjælp to PWM-signaler fra EV3-computeren. Kredsløbet styrer både blæsere og vibrationsmotor og er lavet på perfboard, da der ikke vil være meget at vinde i forhold til størrelse ved at lave en printplade. Der bruges et indkøbt kabel for at få direkte adgang til ledningerne fra EV3-computerens motorstyringsstik.



Figur 8: Kredsløbsdiagram for motorkontrols kredsene.

Kredsløbet kan ses på figur 2. Det består af to MOSFETs, to kondensatorer og fire modstande. IRF3708 MOSFETne er valgt som følge af deres lave Gate Threshold Voltage og induktans, hvilket gør, at de kan anvendes til applikationer med lav gate-source potential. En anden interessant egenskab ved MOSFETs er, at der afsættes en del mindre strøm i transistoren end i andre teknologier, hvilket er en fordel, da de drives af EV3-computeren, som har et begrænset potential. **91**

Kondensatorene bruges til at udglatte det PWM-signal, der kommer fra EV3-computeren. Dette gør, at signalet hele tiden holder sig over den spænding, der skal til for at aktivere transistoren, hvilket forhindrer at MOSFETen står og skifter mellem at være åben og lukket. Da PWM-signalet har en duty cycle på maximum 80% - signalet er altså højt 80% af tiden, vil den gennemsnitlige spænding i dette tilfælde kun være 80% af spændingen over batteriet (cirka 9,6 V), hvilket mindsker blæsernes effektivitet, og dermed sugekraften, mærkbart. Denne effekt ses i nedenstående figur.



Figur 9: Simulation af PWM og dæmpet PWM signal.

Simulationen er udført med større modstande end kredsløbet i diagrammet som følge af begrænsninger i simuleringsværktøjet, hvilket resulterer i en lavere spænding end den, der blev målt i kredsløbet. Som følge af dette er der også en markant længere - mere end faktor ti - setupetid i forhold til det faktiske kredsløb.

5.5 Test

Til hver delopgave blev der opstillet nogle krav, som skulle opfyldes, før opgaven var løst. Kravene er beskrevet i afsnit [4.1](#), og test cases er opsat efter dem. Roadplannerens metoder er, fordi de er meget beregningsmæssige, testet med JUnit 4. Disse JUnit-tests kan findes på følgende [link](#)

Alle test cases er beskrevet i appendiks [C](#)

5.6 Delkonklusion

5.6.1 Billedebehandling

Med udgangspunkt i de vedhæftede test cases, kan der konkluderes, at der umiddelbart ikke er noget problem i detektering af de forskellige objekter ved brug af billedebehandling. Dog bemærkede vi hurtigt udenfor testcasesne, at genkendelsen ikke var konsistent, forstået således at værdier taget på et tidspunkt af dagen ikke var de samme et par timer efter, hvilket skyldes forandrede lysforhold. Alt i alt kan vi konkludere, at vi har lavet en nogenlunde konsekvent billedebehandling, og grunden til at den ikke er fuldstændig konsekvent er, at den bliver så påvirket af forskellige lysforhold.

5.6.2 GUI

Vi kan konkludere, at vi har fået lavet en velfungerende GUI, som gør det lettere for os at indstille vores billeder, således at de passer til det lysforhold, der er gældende. Vi kan desuden få visualiseret, hvilke bolde vi finder, samt banen og robotten. Dette gør, at vi kan få en forståelse for, hvordan robotten agerer, da vi nu kan følge med i det data, vi sender til den.

5.6.3 Opsamling/aflevering

Opsamling af bolde opfylder de stillede krav, og den dokumenterede virkningsgrad oversteg forventningerne. Præcisionen af robotten samt sugeevnen var tilpas god nok til, at der ikke var nogle problemer i den henseende. Til gengæld var robotten ikke altid i stand til at aflevere boldene som følge af problemer med billedgenkendelsen. Præcisionen var i orden, men robotten opfattede ikke altid at den havde fået samlet alle bolde.

5.6.4 Roadplanner

Det kan konkluderes, at vores roadplanner kan føre robotten rundt på banen på en måde, så den så meget som muligt holder sig indenfor de sikre områder, som er defineret ud fra de punkter, der kommer fra OpenCV. Det er muligt for os at samle bolde, som er placeret i hjørner, krydset og andre tilfældige steder på banen. Det sker en gang imellem, at vi rammer banden eller krydset, ofte ved flytning mindre end 1 cm.

6 Follow-up og konklusion

Ved projektets konklusion var de fleste delmål nået, og robotten havde demonstreret, at den kan køre autonomt og opsamle bolde inden for konkurrencens rammer og regler. Gruppen har gennem dette projekt arbejdet med både tekniske udfordringer og projektstyring, og har dermed gennemført

et CDIO-forløb. Der er dog også flere ting, som gruppen ikke kunne nå eller ikke var i stand til at implementere, og der er en række af ting som set i perspektiv kunne være gjort bedre - eller være undgået.

Projektstyring har vist sig at være et godt værktøj, og projektet ville formodenligt ikke havde lykkedes, hvis ikke at der var blevet lagt så meget fokus på det. Gruppen har især haft megen gavn af dets interne møder, Microsoft Project til gantt diagram og Youtrack, der begge har bidraget til at skabe et overblik over projektets forløb og individuelle delopgaver. Det har været lettere at arbejde i gruppe, når alle havde et indblik i, hvad de andre arbejdede på, samt hvordan opgaverne stod i et større kontekst.

På trods af dette har det vist sig at være en udfordring at danne sig et fuldstændigt overblik over projektet. Nogle gange blev opgaver overset eller glemt, delopgaver var pludseligt ikke relevante og andre problemer opstod i takt med at projektet gled frem. Mange af disse kunne have været undgået med bedre planlægning og mere erfaring.

Et af de problemer der først blev synligt sidst i projektet var, at ikke alting blev undersøgt til en tilstrækkelig grad, hvilket resulterede i at udviklingstiden for flere delelementer blev forlænget signifikant. Dette inkluderer blandt andet nogle af de indbyggede funktioner i LeJOS og OpenCV hvor, i stedet for at anvende de givne klasser, blev brugt tid på at designe nye klasser til det samme formål.

Billedebehandling: OpenCV har generelt vist sig at være et godt værktøj til billedbehandling. Vi har dog stødt ind i nogle problematikker, såsom at lysforholdet spiller en ekstrem stor faktor, i forhold til hvilke BGR eller HSV værdier farverne har. Dette viste sig ved, at vi den ene dag kunne detektere en grøn farve til perfektion, hvorimod næste dag kunne vi slet ikke detektere den. Derudover har vi haft nogle problemer med HoughCircles, da vores cirkler på billedet bliver set som elipser. Dette gør, at vi skal sænke vores grænse for, hvor cirkelformet et objekt skal være, før det bliver godkendt. Vi kiggede på at lave blob-detection, hvor man blot søger efter en gruppe af pixels, der har f.eks. samme farvевærdi, men dette blev hurtigt droppet grundet tidspres, samt at dokumentationen til Java var manglende. Detektion af banen samt krydset virker konsekvent, og vi har fundet stor brugbarhed i OpenCV metoden findContours. Den eneste problematik der er med findContours er, at når noget bryder konturen, så vil den ikke længere registrere nogle punkter. Alt i alt kan vi konkludere, at vi har lavet en nogenlunde konsekvent billedebehandling, og grunden til at den ikke er fuldstændig konsekvent er, at den bliver så påvirket af forskellige lysforhold.

Ruteplanlægning: Ruteplanlæggeren afhang til dels af andre komponenter, herunder OpenCV og robotkonstruktion, hvorfor denne først blev udarbejdet til sidst i projektet. Selv om der er blevet brugt relativt kort tid på denne væsentlig del af projektet, bemærkes det, at ruteplanlæggeren betragtes som succesfuld, og dets fejl til dels skyldes OpenCV's præcision. Planlæggeren kan, som førnævnt, planlægge en rute udarbejdet med geo- og trigonometri, selvom det i retroperspektiv nok ville være mest effektiv med en implementation af grafteori.

Afslutningvis konkluderes det, at gruppen er glad for det endelige produkt og ser frem til, at skulle deltage i konkurrencen med en chance for at opnå det originale "project scopes desired result: Vores robot skal indsamle flest bolde på kortest tid og derved opnå flest point og vinde konkurrencen."

7 Arbejdsfordeling

Vi har valgt at inddele projektet i følgende grupper; projektledere, robotkonstruktion, ruteplanlægning, billedgenkendelse og GUI - da alle naturligvis ikke kan være ind over det hele. Ved at inddele projektet på denne måde, kan enkelte personer specialisere sig inden for det nævnte område. Der sørges dog for, at alle bliver introduceret til alle emner, samt at man kan følge status til hvert møde.

Arbejdsfordelingen er approksimativ og afspejler de primære fokusområder for følgende personer (se tabel). Det er dog nødvendigt at nævne, at selvom en person står på et emne, så er emnets designovervejelser blevet vendt og drejet i gruppen for at få flest mulige inputs. Da vi har arbejdet iterativt, især med ruteplanlægningen og billedgendelsen, har flere personer fået mulighed for at arbejde med dette.

De enkelte arbejdsgrupper har stået for deres respektive del i rapporten for at give det bedste indblik i analyse, design og implementering af disse problemstillinger.

	Projekt- ledelse	Robot- konstruktion	Robot- kørselsplanlægning	Billed- genkendelse	GUI
Ali				X	
Christian	X		X		
Jonas	X			X	
Mads		X			
Magnus			X	X	X
Oliver			X		X
Sebastian				X	
William		X			

8 Bibliografi

Referencer

- [1] The LEGO Group. *LEGO MINDSTORMS EV3 Hardware Developer Kit*. URL: <https://education.lego.com/en-us/support/mindstorms-ev3/developer-kits> (sidst set 19.06.2019).
- [2] The LEGO Group. *LEGO MINDSTORMS EV3 Hardware Developer Kit, Appendix 1 - LEGO MINDSTORMS EV3 programmable brick main hardware schematics*. URL: <https://education.lego.com/en-us/support/mindstorms-ev3/developer-kits> (sidst set 20.06.2019).
- [3] Texas Instruments. *AM1808 ARM Microprocessor datasheet*. URL: https://media.digikey.com/pdf/Data%20Sheets/Texas%20Instruments%20PDFs/AM1808,_Dec.22,2016.pdf (sidst set 20.06.2019).
- [4] JetBrains. *Youtrack website*. URL: <https://www.jetbrains.com/youtrack/> (sidst set 20.06.2019).
- [5] LeJOS. *LeJOS homepage*. URL: <http://www.lejos.org/> (sidst set 20.06.2019).
- [6] Maxime. *How to Detect Circles in Images*. URL: <https://www.codingame.com/playgrounds/38470/how-to-detect-circles-in-images1> (sidst set 18.06.2019).
- [7] OpenCV. *CUDA*. URL: <https://opencv.org/cuda/> (sidst set 20.06.2019).
- [8] Oracle. *Remote Method Invocation Home*. URL: <https://www.oracle.com/technetwork/java/javase/tech/index-jsp-136424.html> (sidst set 20.06.2019).
- [9] International Rectifier. *IRF3708 datasheet*. URL: <http://www.irf.com/product-info/datasheets/data/irf3708.pdf> (sidst set 20.06.2019).

A Risikoanalyse

ID	Risk	Impact (0-5)	Probability(0-5)	Score
R0	Gruppemedlemmer kommer ikke til planlagt arbejdsdag	4	3	12
R1	Aftaler/gruppekontrakt bliver ikke overholdt gentagende gange	5	2	10
R2	Robot/HW går i stykker	4	2	8
R3	Dellevering/mål opnås ikke med estimeret tid	4	3	12
R4	Gruppemedlem forlader gruppen	5	1	5
R5	Data/SW går tabt	5	1	5
R6	HW eller SW specifikationer ændres af stakeholders/gruppen, hvilket resultere i om strukturering/planlægning	2	4	8
R7	Misforståelser i kravspecificering/interfacet mellem delgrupper	4	1	4
R8	Stakeholder eller brugere godkender/udvælger ikke vores produkt som det endelige produkt	1	4	4
R9	Projektering i OpenCV kan påvirke præcisionen af robotstyring	3	4	12
R10	Limen som skal holde støvsugeren samlet ikke kan holde til vibrationerne fra vores vibrator.	2	1	2
R11	Fejl i implementering og samling af delelementer	5	2-3	10-15
R12	Vi er for langsomme til at tage et billede og vælge aktion der skal ske evt. nødstop	3	2	6

Figur 10: Identifikation af risici. Samlet fra statusrapporter.

ID	Risk	Score	Strategi	Ansvar
R0	Gruppemedlemmer kommer ikke til planlagt arbejdsdag (syg eller pjækker)	12	Det er enten noget der skal accepteres eller vi skal mindske risikoen, for at det påvirker arbejdsdagen for de andre gruppemedlemmer. F.eks. at folk ikke kun har tingene liggende lokalt på deres pc eller har taget ting med hjem.	Alle
R1	Aftaler/gruppekontrakt bliver ikke overholdt gentagende gange	10	Reducere sandsynligheden og impact ved, at opretholde moralen, følge op på straffen ved kontraktbrud (kage) og snakke i gruppen om det.	Projektlederne
R2	Robot/HW går i stykker	8	Problemet kan muligvis overføres til hjælpe lærerne der kan skaffe nyt ellers må det accepteres som risiko	HW delgruppen
R3	Dellevering/mål opnås ikke med estimeret tid	12	Risikoen skal fjernes ved, at tidsforbruget monitoreres og interne møder følger op	Projektlederne
R6	HW eller SW specifikationer ændres af stakeholders/gruppen, hvilket resulterer i omstrukturering/planlægning	8	En risiko som ikke kan andet end accepteres.	Alle
R9	Projektering i OpenCV kan påvirke præcisionen af robotstyring	12	Forsøg, at implementer projektering i det endelige produkt	OpenCV
R10	Limen som skal holde støvsugeren samlet ikke kan holde til vibrationerne fra vores vibrator.	2	Teste med tape også, og hvis det ikke holder,	Konstruktions holdet
R11	Fejl i implementering og samling af delelementer	10-15	Implementering sker med samtidig testing under ansvar af den pågældende	OpenCV, Planner og interface holdet
R12	Vi er for langsomme til at tage et billede og vælge aktion der skal ske evt. nødstop	6	Optimer OpenCV og evt. gentænk, hvordan instruktioner sendes og udføres på robotten	OpenCV

Figur 11: Handleplan for risici. Samlet fra statusrapporter.

B Krav

1. Billedbehandling

- Positionen af robotten skal tage højde for, at positionen bliver forskudt når robotten kommer væk fra kameraet.
- Positionen for robotten skal findes konsekvent hver gang korrekt.
- Robotten kaster skygger hvilket kan gemme bolde, dette skal der tages højde for i logikken.
- Bolde kan gemme sig bag robotten og ikke detekteres, derfor skal robotten være så lav som mulig.
- Lysforhold kan ændre sig og parametrene der bruges til detektering skal derfor kunne ændres

- Algoritmen for, at finde krydset og give et antal punkter skal kunne tage højde for, at krydset kan ligge skævt og detektering skal være konsekvent ved hvert billede taget.
- Banderne behøves kun, at kunne findes ved kalibrering

2. Opsamling-/afleveringsmekanisme

- En tilpas stor præcision skal opnås i forhold til billedgenkendelse, positionering og boldopsamling således at robotten er i stand til at opsamle bolde under størstedelen af dens forsøg - minimum 50% af tiden.
- Boldene skal kunne afleveres igennem det store mål og robotten skal kunne vinkles præcist nok til at gøre dette konsekvent 90% af tiden.
- Robotten skal kunne køre gennem begge gennemkørsler på konkurrencedagen uden at denne løber tør for strøm eller går i stykker.
- Robotten skal være så lille så muligt, så det er muligt at dreje om sig selv uden at ramme noget, og så det er lettere at undgå krydset - gerne under minimumsafstanden på 30 cm.
- Opsamlingen skal min. kunne samle bolde op på 5 cm. afstand

3. Roadplanneren

- Krydset skal angribes på en bestemt måde og tage højde for, at det ligger skævt.
- Krydset kan ligge andre steder end midten og kan rykkes af robotten, så krydsets position skal findes hver gang.
- Bolde tæt på banderne skal hentes vinkelret ind på banden og bakke ud, ikke bare dreje til ny bold.
- Bolde der ligger i hjørnerne skal hentes med 45° ind på hjørnet og skal bakke ud.
- Bolde kan til enhver tid "forsvinde" og dukke op på næste billede. Dette skal der tages højde for i logikken.
- Robottens brede og længde skal der tages højde for når der køres rundt på banen.
- Vinkler fra robot til bolde skal kunne udregnes med højeste 5 ° fejl.
- Robotten skal kunne planlægge en rute forbi krydset og mellem krydset og en bande, for at komme sikkert til en bold.

4. GUI og integrering

- Det samlede program skal være flertrådet og tage højde for de gængse flertråds-programmerings problemer.
- Parameter og kalibrering af kamera og robot skal ske via GUI'en
- Til testing af robotkørsel og OpenCV skal der laves forskellige "run configuration", så OpenCV f.eks. kan køre uden, at robotten er tilsluttet.
- Robotten og opsamling/skal kunne styres fra GUI'en
- GUI'en skal vise data fra OpenCV og kunne justere på parameterne der bruges i hver enkel algoritme til detektering af de forskellige dele (robot, bolde, bane og kryds).
- Robotten skal have intern tidstagning og signaler når den er færdig

C Test cases

Billedebehandling

Unique testcase ID	TC1
Testcase name	Konsistent genkendelse af bolde
Brief description	Finde bordtennisbolde konsekvent ved hvert billede taget, med mindre robotten skygger for dem
Preconditions	Et frame modtages
Postconditions	Boldene bliver fundet
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Hovedmetoden køres og der sendes et frame som parameter3. Det modtagne frame behandles vha. HSV, <u>Gaussianblur</u>, <u>dilate</u> samt <u>erode</u>4. Der udføres <u>HoughCircles</u> på det behandlede frame5. Der returneres, samt tegnes visuelt, de fundne bolde
Test data	10 bolde
Expected result	10 bolde detekteres
Actual result	10 bolde detekteret
Status	Godkendt
Tested by	Ali Moussa
Date	20/6-19

Unique testcase ID	TC2
Testcase name	Konsistent genkendelse af kryds
Brief description	Detektering af krydset skal findes konsekvent ved hvert billede
Preconditions	Et frame modtages
Postconditions	Krydset bliver fundet
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Hovedmetoden køres og der sendes et frame som parameter3. Det modtagne frame behandles vha. HSV, <u>dilate</u>, Canny4. Der udføres <u>findContours</u> på det behandlede frame5. Der returneres, samt tegnes visuelt, det fundne kryds
Test data	Et kryds
Expected result	Et kryds detekteres
Actual result	Et kryds detekteret
Status	Godkendt
Tested by	Ali Moussa
Date	20/6-19

Unique testcase ID	TC3
Testcase name	Konsistent genkendelse af banderne
Brief description	Banderne skal kunne findes og være afbilledet præcist ved kalibrering.
Preconditions	Et frame modtages
Postconditions	Banderne bliver fundet
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Hovedmetoden køres og der sendes et frame som parameter3. Det modtagne frame behandles vha. HSV, <u>dilate</u>, Canny4. Der udføres <u>findContours</u> på det behandlede frame5. Der returneres, samt tegnes visuelt, de 4 fundne punkter til banderne
Test data	4 punkter fra banderne
Expected result	4 punkter fra banderne detekteres
Actual result	4 punkter fra banderne detekteret
Status	Godkendt
Tested by	Ali Moussa
Date	20/6-19

Unique testcase ID	TC4
Testcase name	Konsistent genkendelse af robottens cirkler
Brief description	Robottens position skal findes konsekvent ved hvert billede
Preconditions	Et frame modtages
Postconditions	Cirklerne bliver fundet
Procedure	<ol style="list-style-type: none"> 1. Hovedprogrammet startes via GUI 2. Hovedmetoden køres og der sendes et frame som parameter 3. Det modtagne frame behandles vha. HSV, <u>Gaussianblur</u>, <u>dilate</u> samt <u>erode</u> 4. Der udføres <u>HoughCircles</u> på det behandlede frame 5. Der returneres samt tegnes visuelt de fundne cirkler
Test data	2 cirkler
Expected result	2 cirkler detekteres
Actual result	2 cirkler detekteret
Status	Godkendt
Tested by	Ali Moussa
Date	20/6-19

Opsamling-/afleveringsmekanisme

Unique testcase ID	TC1
Testcase name	Overordnet præcision af systemet
Brief description	I løbet af 10 forsøg skal robotten vise, at den kan succesfuldt opsamle bolde i 50% af tilfældene.
Preconditions	Bane og robot opsættes under de endelige forudsætninger
Postconditions	Robotten melder at den er færdig
Procedure	<ol style="list-style-type: none">1. Opsæt bane og robot2. Kør run3. Noter resultater4. Gentag 1-3 ti gange
Test data	
Expected result	For hvert run bliver hver anden bold opsamlet i første forsøg
Actual result	Det var ikke muligt at gennemføre mere end fem runs som følge af tidsbegrænsninger. Dog viste robotten sig at være i stand til at <u>opsamle</u> bolde første gang 7/10 gange.
Status	Delvist gennemført
Tested by	William
Date	20/6-19

Unique testcase ID	TC2
Testcase name	<u>Afleveringspræcision</u>
Brief description	Robotten skal i løbet af 10 forsøg vise, at den kan aflevere boldene ved det store mål 90% af tiden.
Preconditions	Robotten er fuld af bolde og er placeret et tilfældigt sted på banen
Postconditions	Alle bolde er afleveret
Procedure	<ol style="list-style-type: none">1. Banen og robotten klargøres2. Robotten fodres med ti bolde3. Robotten sættes til at køre4. 1-3 gentages <u>ti</u> gange
Test data	
Expected result	I gennemsnit afleveres ni eller flere bolde igennem det store mål
Actual result	Det var ikke muligt at gennemføre alle runs som følge af tidsbegrænsninger. Der var problemer med billedgenkendelsen så robotten var i nogle tilfælde ikke i stand til at aflevere bolde da den sad fast i en uendelig løkke. De gange hvor den kunne estimere dens egen position afleverede den dog alle bolde første gang
Status	Delvist gennemført med signifikante mangler
Tested by	William
Date	20/6-19


Unique testcase ID	TC3
Testcase name	Batterilevetid
Brief description	Robotten skal kunne køre 4 forsøg uden, at batteriet (EV3 og støvsugerbatteriet) løber tør.
Preconditions	Batteriet er fuldt opladt
Postconditions	Motorer og blæsere fungerer ved minimum 90% effektivitet
Procedure	Stresstest robot i <u>4x8</u> minutter
Test data	
Expected result	Robotten er stadig i stand til at køre videre
Actual result	Der var ikke nogle problemer i forhold til batterilevetid
Status	Afventer godkendelse
Tested by	Mads
Date	Førhen i projektet

Unique testcase ID	TC4
Testcase name	<u>Sugevidde</u>
Brief description	Opsamlingen af bolde skal måles til, at den som min. skal kunne samle bolde op på 5 cm. afstand
Preconditions	
Postconditions	
Procedure	<ol style="list-style-type: none">1. En bold placeres for enden af en lineal2. Robotten bevæges langsomt tættere på bolden indtil bolden begynder at bevæge sig3. Afstanden noteres
Test data	
Expected result	Den effektive rækkevidde er på 5 cm
Actual result	Den effektive rækkevidde er fra 5 cm helt op til 10 cm
Status	Afventer godkendelse
Tested by	Jonas
Date	Førhen i projektet

GUI og integrering

Unique testcase ID	TC1
Testcase name	Programkørsel uden tilsluttet robot
Brief description	Programmet skal kunne køre med og uden robotten tilsluttet systemet (demo mode)
Preconditions	Programmet startes uden at robotten er sluttet til
Postconditions	Billederne vises stadig, og programmet crasher ikke
Procedure	<ol style="list-style-type: none">1. GUI startes2. Der skiftes mellem de forskellige images
Test data	Webcam <u>livefeed</u>
Expected result	GUI kan starte op, samt vise de forskellige images uden at crashe
Actual result	GUI startede op, og man kan se samtlige images uden at den crasher
Status	Godkendt
Tested by	Sebastian Bilde
Date	20-06-19

Unique testcase ID	TC2
Testcase name	GUI-robotinteraktion
Brief description	Robotten skal kunne styres (dreje, bakke og køre frem) og opsamling-/afleveringsmekanisme skal kunne aktiveres/deaktiveres.
Preconditions	Robotten er tilsluttet, og GUI er startet op
Postconditions	
Procedure	<ol style="list-style-type: none">1. GUI startes op, og robotten connectes2. Der trykkes på de forskellige knapper i GUI'en, som rykker robotten
Test data	En robot
Expected result	Robotten rykker sig i forbindelse med hvilken knap, der er blevet trykket på.
Actual result	Robotten rykkede tilsvarende de knapper der blev trykket på
Status	Godkendt
Tested by	Sebastian Bilde
Date	20/6-19

Unique testcase ID	TC3 
Testcase name	Kalibrering af thresholds på billeder
Brief description	Robotten og kameraet skal kunne kalibreres og <u>parameterne</u> til OpenCV skal kunne sættes så det passer til lysforholdene.
Preconditions	GUI er startet op, og der er modtaget et frame
Postconditions	frame er blevet opdateret med de nye parametre.
Procedure	<ol style="list-style-type: none"> 1. GUI startes op 2. Der ses at frame er modtaget 3. Der justeres på sliders, som ændrer parametrene i thresholden
Test data	webcam <u>livefeed</u>
Expected result	Der ses på imaget i GUI'en at parametrene bliver korrekt opdateret, og der kan isoleres enkelte farver.
Actual result	Parametrene blev opdateret korrekt, og vi kunne isolere de ønskede farver.
Status	Godkendt
Tested by	Sebastian Bilde
Date	20-06-19

Roadplanner

Unique testcase ID	TC1
Testcase name	Fang bold i kryds
Brief description	Robotten skal kunne tage en bold der ligger inde i krydset
Preconditions	Et frame modtages
Postconditions	Framet analyseres for bander, kryds baner og robot, og rute planlægges
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Banen, krydset samt boldene analyseres3. En rute beregnes og sendes til robotten4. Robotten køre instruktionerne og fanger bolden
Test data	En bold placeres inde i krydset
Expected result	Robotten fanger bolden i krydset
Actual result	Robotten kan hvis krydset er placeret optimalt fange bolden uden at berøre ved krydset, hvis krydset ikke er optimalt placeret, kan det resultere i af krydset bliver skubbet over 1 cm væk.
Status	Godkendt med mangler
Tested by	Jonas
Date	20/6-19

Unique testcase ID	TC2
Testcase name	Manøvre uden om et kryds placeret og vinklet vilkårligt
Brief description	Robotten skal kunne manøvrere uden om krydset på dens vej rundt på banen, uden at røre det. Krydset skal ligge ved en vilkårlig vinkling og position.
Preconditions	Et frame modtages
Postconditions	Framet analyseres for bander, kryds baner og robot, og rute planlægges
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Banen, krydset samt boldene analyseres3. En rute beregnes og sendes til robotten4. Robotten køre instruktionerne og navigere rundt om krydset
Test data	En rute planlægges gennem et kryds
Expected result	Robotten navigere/manøvre sig væk fra krydset
Actual result	Robotten kan navigere udenom kryds og rejse fra venstre side til højre side selvom krydset ligger i vejen.
Status	Godkendt
Tested by	Jonas
Date	20/6-19

Unique testcase ID	TC3
Testcase name	Fang bold placeret i et hjørne eller op af banden
Brief description	Robotten skal kunne tage bolde der ligger i hjørnerne eller op af banderne uden, at røre banderne.
Preconditions	Et frame modtages
Postconditions	Framet analyseres for bander, kryds baner og robot, og rute planlægges
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Banen, krydset samt boldene analyseres3. En rute beregnes og sendes til robotten4. Robotten køre instruktionerne og fanger bolden
Test data	En rute planlægges, målrettet en bold i et hjørne eller op af banden
Expected result	Robotten navigere og fanger bold i et hjørne eller op af banden
Actual result	Robotten kan navigere til den nærmeste bold og opsamle bolde selv hvis de befinder sig op ad væge eller i hjørner, det hænder dog specielt ved hjørner at vægge rammes men bliver ikke flyttet over 1 cm.
Status	Godkendt, med få mangler
Tested by	Jonas
Date	20/6-19

Unique testcase ID	TC4
Testcase name	Placering af robot i et hjørne, vinklet parallelt
Brief description	Robotten skal kunne sættes på banen i hjørnet og vinklet parallelt med den korte side af banden på den måde, at den skal kunne køre fra den ene ende til den anden hen til en bold i en enkelt (kør ligeud) instruktion.
Preconditions	Et frame modtages
Postconditions	Framet analyseres for bander, kryds baner og robot, og rute planlægges
Procedure	<ol style="list-style-type: none">1. Hovedprogrammet startes via GUI2. Banen, krydset samt boldene analyseres3. En rute beregnes og sendes til robotten4. Robotten køre instruktionerne og bevæger sig hen til bolden
Test data	En rute planlægges, og der sendes en enkelt instruktion
Expected result	Robotten kører fra den ene ende til den anden hen til en bold via et enkelt instruktion (kør ligeud)
Actual result	Robotten havde et lille svaj i kørslen, men ikke noget der skal behandles.
Status	Godkendt
Tested by	Jonas
Date	20/6-19

D Projektplan

Kan ses på næste side

