

Elektronisk Russisk Roulette

3. Semesterprojekt
Projektrapport

30. maj 2024



AARHUS
UNIVERSITET

3. Semester, F24

Gruppe 8
ASE - AU

Studienr.	Navn	Studieretning
202209869	Mads Villadsen	SW
202208480	Sammi	SW
201805038	Sofus	SW
202103258	Amin	SW
202209294	Asmahane Bismir	E

Vejleder: Michael Loft

Indhold

1 Resumé	4
2 Abstract	4
3 Forord	5
4 Projektformulering	6
4.1 Projektdeltagernes ansvarsområder	8
5 Metode og proces	9
5.1 Projektstyring	9
5.2 Agil Udvikling	9
5.3 Anvendelsen af Tidsplaner	10
5.4 Metode anvendelse	10
6 Kravspecifikation	12
6.1 Aktør og Use Case Diagrammer	12
6.2 Brief Use Case Beskrivelser	15
6.3 FDUC:	16
6.4 Ikke Funktionelle Krav:	17
7 Risikoanalyse:	18
7.1 Første Risikoanalyse	18
7.1.1 Mitigation/Contingency Plan	19
7.2 Udførelse af Contingency Plan	19
7.3 Efterfølgende Risikoanalyser	19
8 Teknologianalyse:	21
8.1 1. Overblik over Test	21
8.2 2. Krav til Komponenter	21
8.3 3. Mulige Kandidater	21
8.4 4. Resultater af Test	22
8.5 Reflektion over teknologianalysen	23
9 Systemarkitektur	24
9.1 Hardware Arkitektur	25
9.1.1 Block Diagrammer	26
9.2 Software Arkitektur	29
9.2.1 Sekvensdiagrammer	31
10 Design	33
10.1 HW Design	33
10.1.1 Funktionalitet og kredsløbsdesign	33
10.1.2 Kredsløbs forsyning	34
10.1.3 Sensore	34
10.1.4 IC	35

10.1.5 LED	37
10.1.6 Transistor	37
10.1.7 PCB Design	38
10.2 SW Design	41
10.2.1 PSoC Design	41
10.2.2 STM PSoC:	43
10.2.3 RPI Design	45
11 Realisering	47
11.1 Realisering og Design af klap aktivering	47
11.2 Database og Backend Server Implementering	47
11.2.1 Overblik og Funktionalitet	48
11.2.2 Kommunikation med UI	49
11.2.3 Datahåndtering	49
11.3 GUI Implementering	50
11.3.1 Sekvensdiagram for flow	50
11.3.2 Opbygning med React	50
11.3.3 Sideopbygning og Navigation	50
11.3.4 Interaktion med Backend	50
11.3.5 Styling og Responsivitet	51
11.4 Sikkerhed og Datahåndtering	51
12 Test af Elektronisk Russik Roulette	53
12.1 Modultest	55
12.1.1 Test af Klapaktivering	55
12.1.2 Modultest af Gui og Backend	56
12.2 Integrationstest	59
12.2.1 Samlet PSoC Test	59
12.3 Accepttest	61
12.4 Opsumering af resultater for udført accepttest:	61
13 Diskussion	63
13.1 Fejlkilder	63
14 Konklusion og Refleksion	64
14.1 Konklusion	64
14.1.1 Retrospektiv	64
14.1.2 Fremtidigt Arbejde	65
15 Bilagsoversigt	67
16 Ordliste	68

1 Resumé

Dette projekt fokuserer på udviklingen af et roulettespil, navngivet ”Elektronisk Russisk Roulette,” og en applikation, med hovedvægt på en effektiv brugergrænseflade bygget i React og en pålidelig backend-infrastruktur. Ved anvendelse af moderne sikkerhedspraksis såsom Cross-Origin Resource Sharing (CORS) og HTTP-anmodninger, har projektet sikret en robust sikkerhedsarkitektur og effektiv datahåndtering.

Gennem en omhyggelig testproces, der omfattede modul-, integrations- og accepttest, har gruppen verificeret systemets funktionalitet og kvalitet. Selvom visse krav, som ’Party Mode’ og præcis LED-afvikling, ikke blev fuldt opfyldt, har testen identificeret områder til forbedring og styrket tilliden til systemets pålidelighed.

Brugen af sekvens-, klasse- og andre diagrammer har sikret en struktureret forståelse af GUI-opbygningen og backend-interaktionerne. Disse diagrammer har været integreret i udviklingsprocessen og har bidraget til en sammenhængende og veldefineret arkitektur.

Undervejs i projektet stødte gruppen på forskellige udfordringer, herunder implementering af SCRUM-metoden til projektstyring og realistisk tidsplanlægning. Refleksion over disse udfordringer har fremhævet behovet for forbedret risikostyring, kommunikation og ressourceallokering mellem gruppens medlemmer henover hele projektperioden.

Samlet set repræsenterer projektet en værdifuld læringsoplevelse, der har styrket gruppens forståelse af både tekniske og procesmæssige aspekter inden for hardware- og softwareudvikling. Gruppen ser frem til at anvende disse erfaringer i fremtidige projekter og kontinuerligt forbedre deres kompetencer og praksis.

2 Abstract

This project focuses on the development of an advanced roulette game, nicknamed ”Electronic Russian Roulette”, and an implementation of an application using React to create an efficient and user-friendly interface for the game. The application is hosted on a Raspberry Pi, making it accessible through a standard web browser over LAN/Ethernet by different types of devices. The frontend architecture comprises four main components: HomePage, Game, ActiveBetsPage, and StatisticsPage, managed via React Router to ensure seamless navigation.

The system’s backend communication is achieved through asynchronous HTTP-requests, ensuring robust and efficient data transfer. Modern security measures, including Cross-Origin Resource Sharing (CORS), have been implemented to safeguard the data in the system against unauthorized access. An example of data is a bet by a player. An extensive testing process, including module, integration, and acceptance tests, has verified the system’s functionality and reliability. Although certain requirements, such as ’Party Mode’ and precise LED sequencing, were not fully implemented, the testing phase identified areas for improvement and reinforced confidence in the system’s reliability.

Utilizing sequence-, class-, and other diagrams has ensured a clear and structured understanding of both the GUI architecture and backend interactions. The project was executed using the SCRUM-methodology, providing valuable insights into project management and collaboration. The challenges encountered

underscored the importance of effective communication, realistic scheduling, and risk management.

Overall, this project represents a valuable learning experience, enhancing the group's technical and procedural skills, and these insights will be applied in future projects.

3 Forord

Rapporten er fremstillet i forbindelse med et 3. semesterprojekt på Aarhus Universitet. Projektet er udarbejdet af 1 Elektroteknologistuderende, og 4 Software Teknologistuderende. Rapporten er lavet med udgangspunkt i at illustrere læringsmål i forhold til projektstyrelse og gruppemedlemmernes faglige kompetencer.

Projektet har taget udgangspunkt i en fremstilling af en elektronisk roulettespil, hvor den normale rotation er erstattet med lysdioder, der blinker. Til dette er der fremstillet en brugergrænseflade, som gør det muligt at placere bets.

Gruppen har delvist fokus på en ”sjov” fremstilling af produktet, der primært bliver anvendt i f.eks. en fredagsbar som et ”hyggespil”. Med fokus på sjov, endte projektet med navnet ”Elektronisk Russisk Roulette” for at fange blikke, men den eneste association med en ’russisk roulette’ ligger kun i navnet.



Figur 1: Elektronisk Russik Roulette

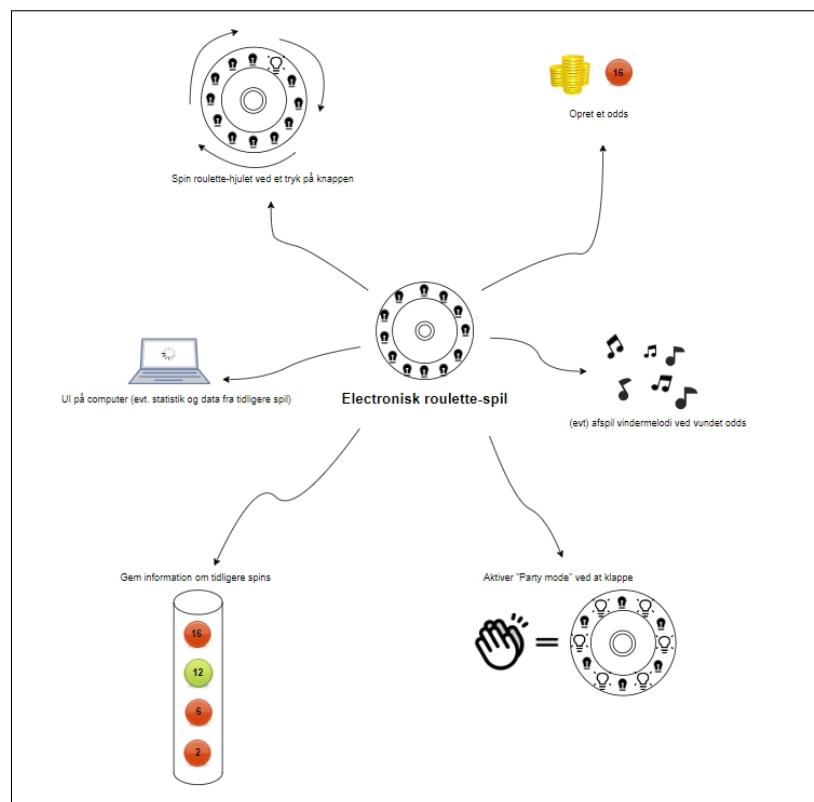
Rapporten er afleveret 30. maj 2024 og indeholder antal tegn: 55980 .

4 Projektformulering

Elektroniske spil og interaktive enheder er blevet en stigende popularitet iblandt de unges miljø, og behovet for mere innovative, brugervenlige og engagerende systemer er voksende. Ved at integrere embedded Linux og PSoC, kan gruppen udvikle et system, der kan styre mere komplekse spilmekanismer og styrke de sociale arrangementer og oplevelser blandt unge såsom i fredagsbar.

I dette projekt vil der udvikles en elektronisk roulette, der er fuldt elektronisk og simulerer det fysiske spin med en række lysdioder, som repræsenterer roulettens felter. En brugergrænseflade vil tillade spillere at vælge en af roulettens mange felter og placere bets. Desuden vil en server holde styr på bets, beløb og vindere. Den elektroniske roulette skal skabe underholdning og interaktion samt være brugervenlig for enhver person. Dette elektroniske roulettehjul aktiveres ved et tryk, og spillernes bets og statistikker vil være baseret på virtuelle karakterer, så der ikke er nogen økonomisk risiko involveret.

Figuren nedenfor giver en oversigt over produktets funktionalitet og illustrerer nogle af roulettehjulets væsentlige egenskaber.



Figur 2: Rigt billede af projektet

Systemet er designet som en prototype og består af flere nøglekomponenter: Roulettehjulet består af en PSoC, der styrer spillets gang, et lysdiodekredsløb bestående 37 lysdioder i tre udvalgte farver, lysensorer til aktivering ved klap og en højttaler, der afspiller lyd ved aktivering. Serveren omfatter en Raspberry Pi, der fungerer som den primære brugergrænseflade for den elektroniske roulette, begrænset til LAN, samt en PC udstyret med en Unix-VM til at tilgå serveren.

Roulettehjulet anvender integrerede lysdioder til at simulere hjulets rotation og stoppe ved et tilfældigt tal. Lysdiodernes lysstyrke og mønstre kontrolleres for at skabe en visuel repræsentation af roulettehjulets bevægelse. En indbygget sensor registrerer, når brugeren dobbeltklapper, hvilket aktiverer ”Party Mode” med blinkende lysdioder og musikafspilning.

Den overordnede styring af systemet er baseret på embedded Linux, som håndterer de forskellige softwarekomponenter, herunder spillokikken og brugerinteraktionen. PSoC fungerer som broen mellem software og hardware, håndterer brugerinput, beregner odds og styrer roulettehjulet. Kombineret med PSoC’s mikrocontroller og integrerede FPGA.

Kommunikationen mellem PSoC og roulettehjulet sker gennem et interface, der overfører kommandoer og data mellem de to enheder. Data fra hvert spin indsamlies og lagres til statistisk analyse, og en brugergrænseflade på en computer præsenterer disse data gennem et brugervenligt UI.

4.1 Projektdeltagernes ansvarsområder

Hver gruppemedlem har hver især ansvaret for forskellige områder af projektet.

Tabel over Ansvarsområder					
Ansvarshavende:	Mads	Asma	Sammi	Sofus	Amin
Rapport:					
Indledning:		P			
Arbejdsprocess og Metode:		P			
Kravspecifikation:					P
Risiko/Teknologi Analyse:				P	
Systemarkitektur:					
HW Arkitektur:			P		
SW Arkitektur:	P				
HW Design:		P			
SW Design:			P		
Implementering:				P	P
Test:				P	P
Diskussion:					
Konklusion:	P				
Bilag:					
Processbeskrivelse					
Teknologianalyse					
HW Arkitektur					
SW Arkitektur					
HW Design					
Implementering og Test				P	P
Implementering:					
PSoC HW:		P		S	
PSoC SW:		S		P	
GUI	S			S	P
Backend			S	S	P

Tabel 1: Ansvarsområder

Tabellen er opdelt efter ”P” for primær på og ”S” som sekundær på en opgave. Opgaver kan godt have flere primære, hvis gruppemedlemmer har arbejdet tæt sammen.

Derudover er der en række bilag som gruppen har udviklet i fællesskab, som derfor ikke er inkluderet i tabellen. Disse er: *Projektformulering*, *Kravspecifikation*, *Accepttest Specifikation* og *Systemarkitektur*.

5 Metode og proces

I dette afsnit kommer gruppen ind på, hvordan der er blevet gjort brug af SCRUM-metoden, hvordan arbejdsprocessen er blevet præget, og hvilke redskaber der er blevet brugt til opgaveuddeling blandt gruppens medlemmer og hvordan disse er blevet håndteret tidsmæssigt.

5.1 Projektstyring

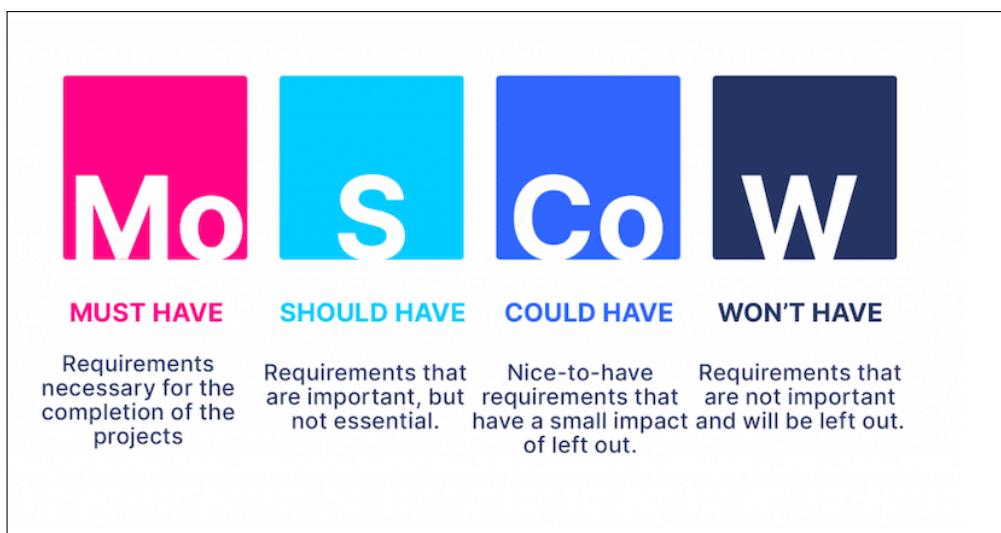
Til projektet er der anvendt SCRUM som gruppens primære projektstyringsmetode, understøttet af Redmine til opgavestyring og MindMeister til planlægning af opgaver og trin. Gruppen fungerede selvstyrende uden formel projektleder, med Sofus som Scrum Master og i starten Sammi som Product Owner, hvilket ændrede sig undervejs i projektet.

De første sprints blev udført i fællesskab, hvor gruppen mødtes ugentligt og fordelede ansvarsområderne. Der gruppen begyndte på design af hardware og software samt kommunikationen mellem Raspberry Pi og platformene, deltes gruppen op i disse kategorier. Stand-up møder hver mandag og fredag sikrede løbende opdateringer på fremdriften, mens vejledningsmøderne hver tirsdag gav feedback, sikrede løbende forståelse af projektets delmål og krav.

5.2 Agil Udvikling

Arbejdsprocesen var præget af en agil tilgang, der tillod gruppen at tilpasse sig og reagere hurtigt på ændringer og udfordringer undervejs. Selvom der blev fundet værdi i SCRUM-metoden, stod gruppen også over for udfordringer, især da flere gruppemedlemmer trak sig. Dette nødvendiggjorde en omstrukturering af arbejdsfordelingen og en tilpasning af gruppens tilgang til projektledelse.

For at imødekomme disse udfordringer valgte gruppen at prioritere projektets ”must-have”krav. Gruppen har forsøgt at opretholde en klar kommunikation og fokuserede på indsats for at opnå disse centrale mål først.

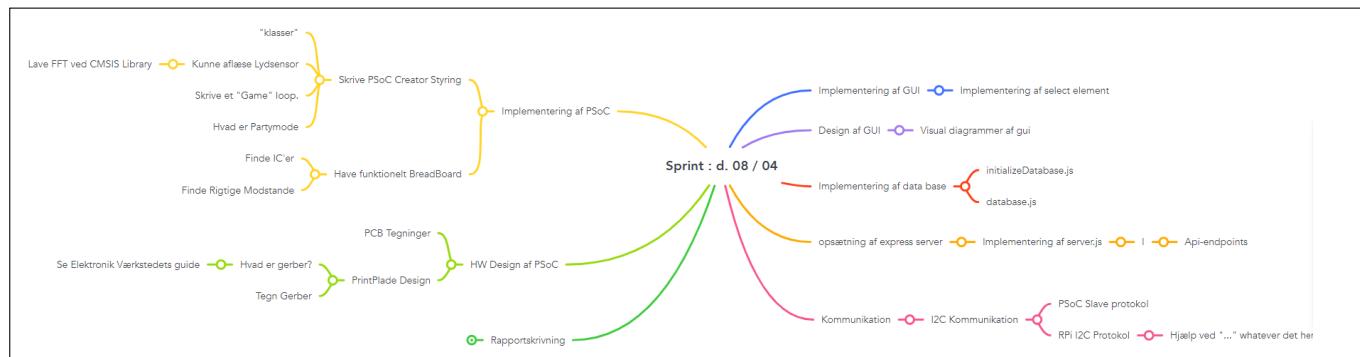


Figur 3: MoSCoW krav model, som blev anvendt til prioritering under opgavestyring [6]

Den agile tilgang gjorde det muligt for gruppen at være fleksible og tilpasse sig ændrede omstændigheder og krav. Selv når der opstod udfordringer, tillod den agile udvikling at opretholde fremdriften i projektet, sikre gruppensamarbejdet og levere et vellykket projekt inden for de givne rammer og begrænsninger.

5.3 Anvendelsen af Tidsplaner

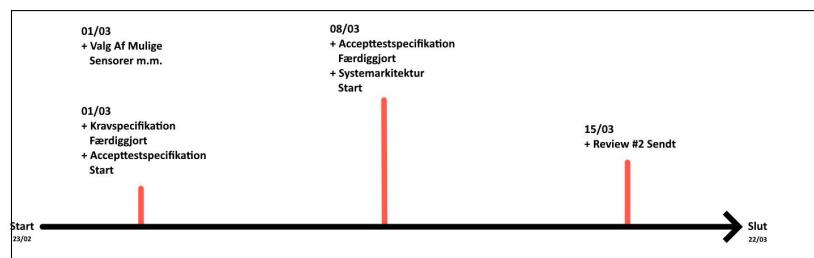
Der er brugt Redmine Gantt-diagrammer til at overvåge opgavestatus og tildele opgaver. MindMeister hjalp gruppen med at planlægge konkrete trin og øge fleksibiliteten i projektforløbet. Gruppen begyndte sent med kortsigtede tidsplaner/sprints, men justerede dem løbende for at sikre effektivitet. Der er løbende blevet fulgt op på arbejdsproces og tidsplan med vejleder, hvilket gav mulighed for at justere og forbedre gruppens tilgang.



Figur 4: Mindmap over Arbejdsopgaver

Under implementeringsfasen blev der stødt på forsinkelser med printplader og softwaretestningen, hvilket udskød accepttest med halvanden uge. Dette var en vigtig læring på både bedre kommunikation samt realistisk tidssætning på arbejdsopgaverne til næste semesterprojekt.

Gruppen oprettede en langsigtet tidsplan og planlagde derefter 2-ugers sprints. Når slutningen i processen, ændrede gruppen til ugentlige sprints, grundet de kortere sprint var med henblik at skabe bedre overblik, og få mere udarbejdet på kortere tid.



Figur 5: Tidsplan

5.4 Metode anvendelse

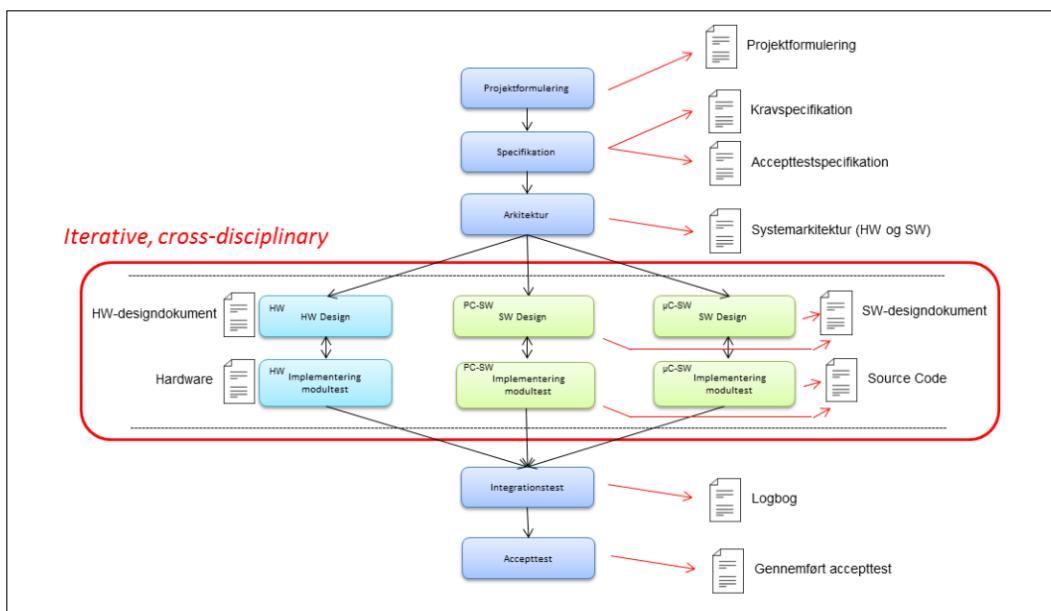
Projektet er uarbejdet efter den iterative ASE-model og udviklingsmodeller som UML og SysML. Dette bidrog til en struktureret og effektiv arbejdsproces. Gruppen kunne have brugt RUP-modellens struktur for en

mere detaljeret tilgang.

Hardwarearkitektur blev visualiseret gennem BDD'er og IBD'er. Disse diagrammer blev brugt til at beskrive roulettens gameboard og gameserver.

Softwarearkitekturen blev beskrevet gennem domænemodeller, klassediagrammer og sekvensdiagrammer. Domænemodellen skabte et overblik over systemet, mens klassediagrammerne specificerede de forskellige klasser og deres funktioner for både PSoC og Raspberry Pi. Sekvensdiagrammerne viste interaktionen mellem systemets komponenter under forskellige scenarier, såsom placering af bets og spilstart.

Ved at anvende ASE-modellen og de nævnte udviklingsmodeller arbejdede gruppen iterativt og inkrementelt, hvilket hjalp med at tilpasse ændringer og forbedre gruppens løsning løbende.



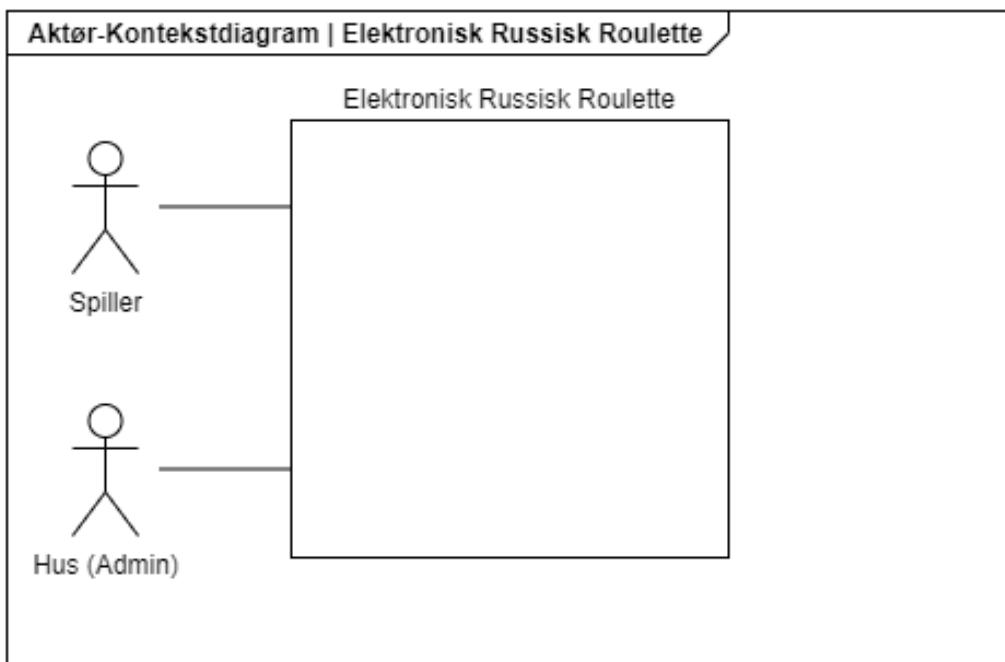
Figur 6: ASE-modellen blev anvendt som en arbejdsmetode igennem hele projektet
[11]

6 Kravspecifikation

Kravspecifikationen er udarbejdet for at lægge et klar fundament for, hvad der skal til for udføre projektet, samt fremlægge en klar prioritering af produktets funktioner. Flere opdelte use cases gjorde det nemmere både at teste og opstille konkrete testbare krav. Senere i rapporten vil der diskuteres indflydelsen dette har haft på projektet.

6.1 Aktør og Use Case Diagrammer

På figur 7 ses aktør-kontekstdiagrammet for Elektronisk Russisk Roulette.

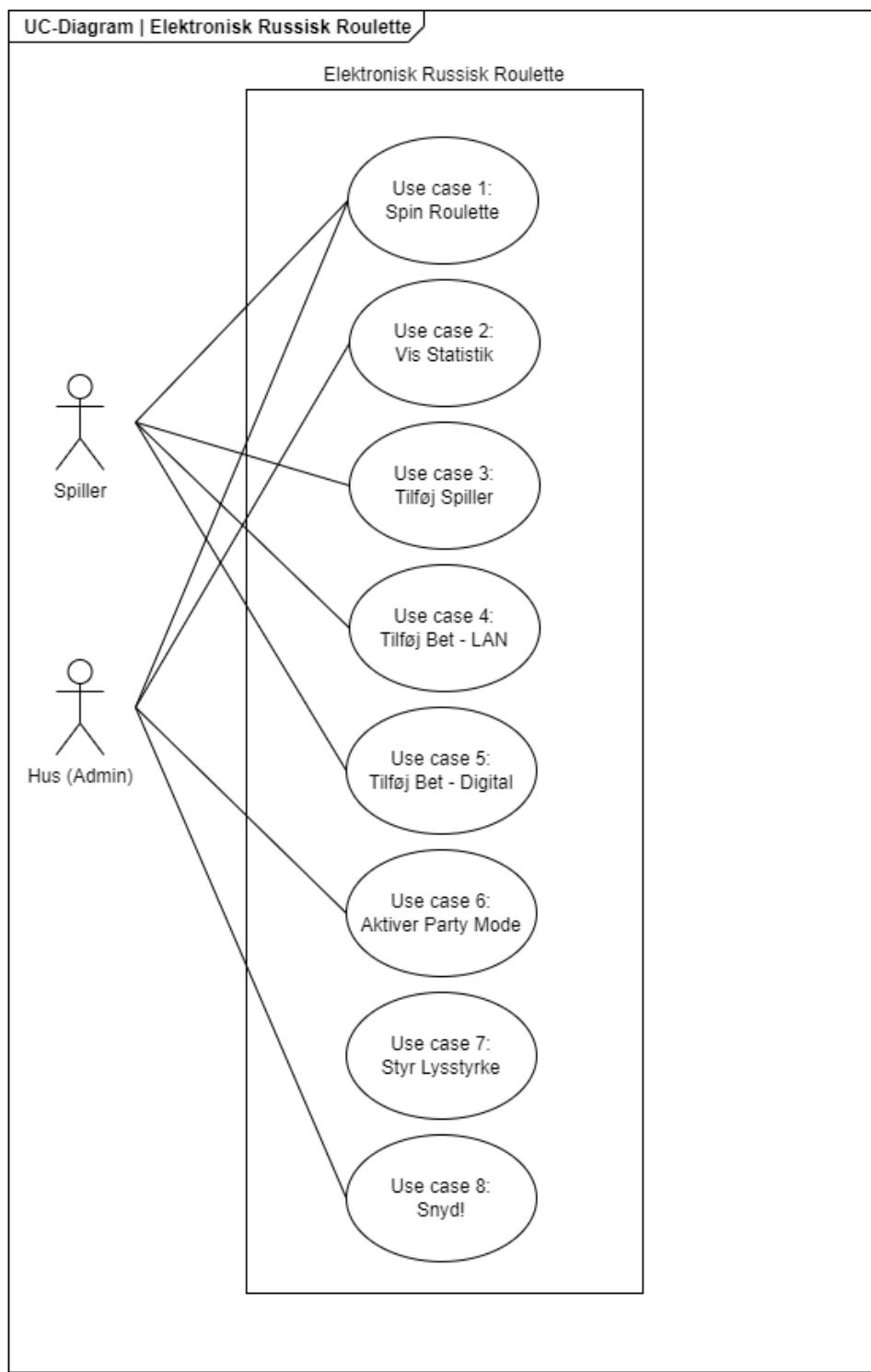


Figur 7: Aktør-kontekstdiagram for Elektronisk Russisk Roulette

Der er valgt at arbejde ud fra to primære aktører: Spilleren som deltager i roulette spillet, og huset der styrer spillets gang. Med visionen om at spillet bruges mest som sjovt, er det muligt tage rollen som både **Hus** og **Spiller**.

Aktør	Type	Beskrivelse
Spiller	Primær	Spilleren deltager aktivt i spillet ved at placere bets, spinne roulette og modtage resultaterne af spillets udfald.
Hus (Admin)	Primær	Huset styrer og overvåger spillets forløb, herunder administration af bets, udbetaling af gevinst, og vedligeholdelse af systemet.

Tabel 2: Beskrivelse af aktører i systemet



Figur 8: Use Case Diagram for Elektronisk Russisk Roulette

Ud fra aktør-kontekst diagrammet er der udviklet et UC-Diagram set på fig 8. Hvor der illustreres hvordan aktører tænkes at interagere med systemet.

MoSCoW

Grundet valget om at arbejde ud fra en relativt stor mænge use cases, er der lavet en klar prioritering af, hvilke har størst betydning for projektet. Til dette anvendes MoSCoW modellen.

MUST:

- **Use Case 1:** *Spin Roulette*
- **Use Case 2:** *Vis Statistik*
- **Use Case 3:** *Tilføj Spiller*

SHOULD:

- **Use Case 4:** *Tilføj Bet - LAN*
- **Use Case 5:** *Tilføj Bet - Digital*
- **Use Case 6:** *Aktiver Party Mode*
- **Use Case 7:** *Styr Lysstyrke*

COULD:

- **Use Case 8:** *Snyd!*

Must:

Must omfatter helt fundamentalle operationer for at den Elektroniske Roulette kan fungere, som UC 1 **Spin Roulette** og UC 3 **Tilføj Spiller**.

Vis Statistik er placeret under MUST, ikke fordi den er essentiel for produktet som helhed, men fordi den er vurderet som illustrativ for projektets læringsmål.

Hvor de er placeret under SHOULD er implementeringen af **ENTEN** UC4 eller UC5 også antaget som et MUST da det er essentiel for spillet.

6.2 Brief Use Case Beskrivelser

De enkelte Use Cases fra **Figur 8** beskrives her for at give en kort forståelse af deres funktion og formål indenfor systemet.

Use Case 1: Spin Roulette

Hus aktiverer spillet ved at trykke på knappen **Spin Roulette**. De 37 lysdioder blinks i en cirkulær sekvens og stopper således, at et tilfældigt tal lyser op.

Use Case 2: Vis Statistik

Hus trykker på *Statistik*-knappen for at vise resultater fra seneste spil samt overordnede spilstatistikker, som kan inkludere information om brugen af *Snyd*.

Use Case 3: Tilføj Spiller

Spiller tilgår systemet og indtaster et unikt brugernavn for at deltage i spillet, enten via en lokal displayenhed eller over et LAN-netværk.

Use Case 4: Tilføj Bet - LAN

Spiller har forbindelse til database-server via LAN eller Bridged Ethernet. Efter et bet er valgt af *Spiller*, kan *Spiller* indtaste beløb via GUI. Bet og beløb sendes derefter til database-server, der opdateres overfor andre eventuelle *spillere*, som kan se beløbet, som er blevet placeret på nuværende spil. *Spiller* kan ikke tilføje nye bets indtil **Use Case 1** er blevet udført.

Use Case 5: Tilføj Bet - Display

Spiller vælger bet ved hjælp af display, og herefter kan *Spiller* indtaste beløb via GUI på displayet. Bet og beløb gemmes lokalt og vises på GUI overfor andre *spillere*, som også kan tilføje deres bet og beløb på samme skærm. *Spiller* kan ikke tilføje nye bets indtil **Use Case 1** er blevet udført.

Use Case 6: Aktiver Party Mode

For at aktivere ”party mode” klapper *Spiller* eller *Hus* 2 gange. De 2 klap vil blive opfanget af roulettespillets lysensor. Herefter vil roulettens afspille en ”party-melodi”, mens de 18 røde lysdiode og 18 grønne lysdiode blinker på tværs af hinanden.

Use Case 7: Styr Lysstyrke

En lysensor på roulettespillet vil detektere om der er mørkt eller lyst i de omgivelser, hvor roulettespillet befinner sig. Hvis der er lyst vil lysstyrken for de 37 lysdiode blive hævet, hvorimod hvis der er mørkt, så vil lysstyrken for de 37 lysdiode blive sænket.

6.3 FDUC:

Til projektet er der lavet FDUC

Navn:	Spin roulette
Mål:	At udføre et roulette spil
Initiering:	Tryk på start-knap
Aktører:	Primære: Hus, Spiller Sekundære:
Antal Samtidige Forekomster:	1
Prækondition:	Systemet er funktionelt og tilkoblet en strømkilde
Postkondition:	Et spil er færdiggjort og resultatet er logged.
Hovedscenarie:	
1. Hus initierer et spil ved at trykke på startknappen 1.1 Extension 1: Intet Hus 2. Systemets led'er blinker rundt i en cirkel-sekvens 3. Et tilfældigt tal mellem 1 og 37 bliver genereret 4. Systemets led'er begynder efter 10 sekunder at blinke langsommere 5. Alle led'er slukker undtagen en enkelt led, som lyser ud for det tilfældigt genereret tal 6. Resultatet bliver logget i systemet	
Undtagelser/udvidelser:	
1. Ingen Extensions til denne use case	

Tabel 3: FDUC: UC1

Navn:	Tilføj Spiller
Mål:	Spiller tilkobler systemet via display eller LAN forbindelse
Initiering:	Spiller trykker på ”Join Game”
Aktører:	Primære: Spiller
Antal Samtidige Forekomster:	Kun én instans af use casen kan udføres af en spiller ad gangen
Prækondition:	<ol style="list-style-type: none"> 1. Spiller er ikke allerede tilkoblet til systemet 2. Under 8 andre spillere er tilkoblet systemet
Postkondition:	1 Spiller er tilkoblet systemet og kan se serverens startside
Hovedscenarie:	
<ol style="list-style-type: none"> 1. Systemet viser ”Welcome Screen” og beder spiller indtaste brugernavn. 2. Spiller indtaster ’unik’ brugernavn på 1 til 20 tegn. <ol style="list-style-type: none"> 2.1 Exception 1: Brugernavnet er for langt. 2.2 Exception 2: Anden spiller har allerede det indtastede brugernavn. 3. Systemet viser spilleren spilles startskærm og andre tilkoblede spillere. 4. Systemet opdaterer andre spillere. 	
Undtagelser/udvidelser:	
<ol style="list-style-type: none"> 1. Brugernavnet er for langt. <ol style="list-style-type: none"> 1.1 Systemet beder spiller indtaste nyt gyldigt brugernavn. 2. Anden spiller har allerede det indtastede brugernavn. <ol style="list-style-type: none"> 2.1 Systemet beder spiller indtaste nyt gyldigt brugernavn. 	

Tabel 4: FDUC: UC3

6.4 Ikke Funktionelle Krav:

Til projektets IFK anvendes **FURPS**-modellen, samt MoSCoW til prioritering, MUST kravene er inkluderet i rapporten, og resterende kan ses i [2]. Ingen af projektets Supportability krav er must, og er derfor ikke inkluderet

Usability

1. Systemet **skal** kunne ”supporte” op til 8 spillere
2. Spillere **skal** kunne vælge imellem præcis tal, farve, lige/ulige og høj/lav.

Reliability

1. Fordelingen af udfald i UC1 **skal** være uniform.

Performance

1. Systemet **skal** indeholde min. 21 lysdioder.
2. Systemets lysdioder **skal** være i min. 3 forskellige farver.

7 Risikoanalyse:

Gruppen har løbende foretaget en risiko analyse for at identificere mulige risici der kunne forekomme i løbet af projektet. Analysen er lavet iterativt og opdateret løbende igennem projektet med en planlagt ca. 2 ugers mellemrum, hvor den første blev gennemført d. 01/03/2024.

Metoden gruppen har foretaget analyserne på, var at brainstorme alle tænkelige problemer projektet kunne støde på, og derefter rangere dem efter konsekvenser og sandsynlighed med en score fra 1 til 5. Alle risici der blev vurderet for små blev frasorteret. Efter de irrelevante var frasorterede, diskuterede gruppen kort om de tidligere scorer var nogenlunde præcise.

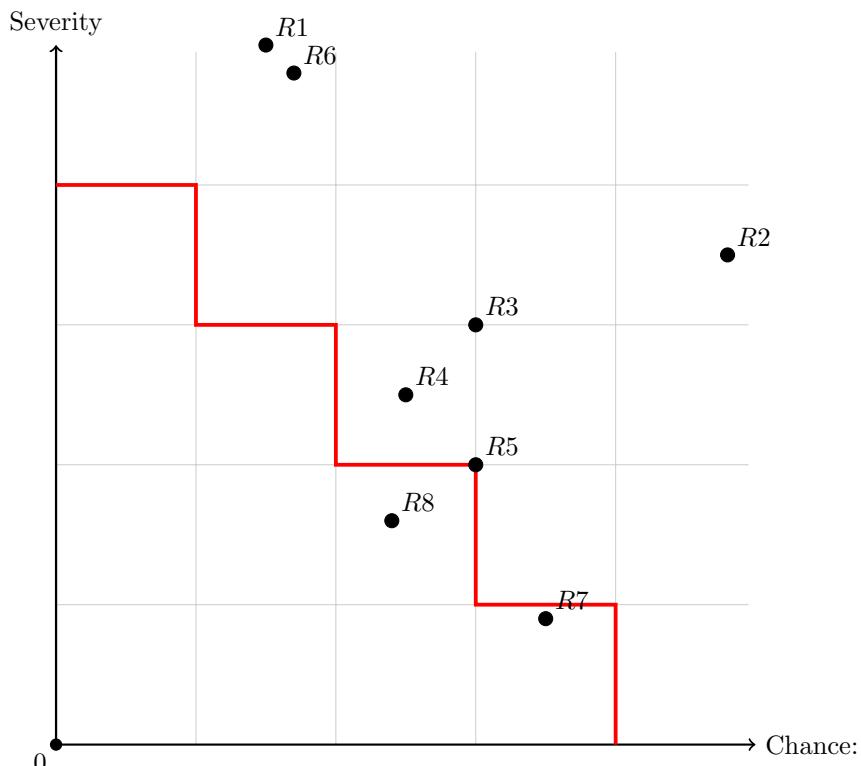
Risikoerne blev opdelt efter ”Tekniske” og ”Generele” for nemmere at kunne lægge mitigation og contingency planer.

7.1 Første Risikoanalyse

Hele risikoanalysen kan ses i bilag: Risikoanalyse[6], derudover kan udvalgte ses herunder:

- **R1:** Et medlem af gruppen dropper ud af studiet.
- **R8:** Finde passende sensorer + Læse output.

Alle de relevante risici er blevet plottet på en graf, se fig 9, hvor alle risici over den røde linje har fået lavet mitigation og contingency plan. Grafen er fra første risikoanalyse[6].



Figur 9: Risikoanalyse: 01/03/2024

7.1.1 Mitigation/Contingency Plan

I rapporten er der udvalgt planer for **R1**, dette skyldes at det var et risikoen blev en realitet, da et gruppemedlem droppede ud kort efter risikoanalysen blev lavet, og gruppen fik erfaring med at udføre contingency planer.

Mitigation Plan (R1): Gruppen støtter hinanden op, og sørger for, at hvergang gruppen mødes, laves der en hurtig runde, hvor alle gruppemedlemmer kommer med deres nuværende følelse af, hvordan studiet går for dem. Hvis et gruppemedlem føler sig utilpas ift. studiet (for mange afleveringer, mistrives, el.lign) sørger de andre gruppemedlemmer om at støtte gruppemedlemmet med kampgejst og eventuelle overvejelser til, hvordan medlemmets beslutning ift. at droppe ud kan mindskes.

Contingency Plan (R1):

Projektet revurderes og nedskaleres til en passende mængde for 4 medlemmer. Nedskaleringen gøres med udgangspunkt i at oprettehold MUST use cases ifht MoSCoW. Ny plan lægges til hvis flere medlemmer frafalder.

7.2 Udførelse af Contingency Plan

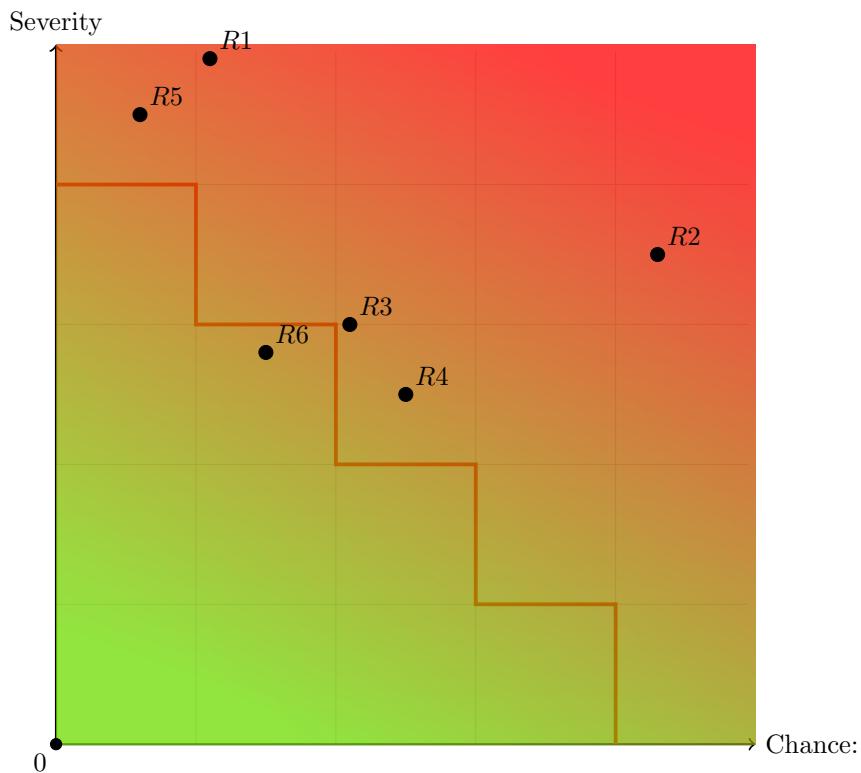
I udførelsen af planen tog gruppen disse skridt

- Gruppen har nedprioriteret ”UC8: Snyd!”. Den fremstår stadig i opsætningen, så hvis gruppen senere har mulighed for at implementere
- Gruppen genovervejede prioriteringen af projektets Use Case: Hvis nyt gruppemedlem falder fra, vil enten UC4(Tilføj Bet: LAN) eller UC5 (Tilføj Bet: Display) bortskaffes. Hvilken er afhængigt af studieretningen på den frafaldne.
- Risiciens ”Severity” er hævet, da $5 \Rightarrow 4$ tænkes at have større konsekvenser end $6 \Rightarrow 5$. Chance er uændret.

7.3 Efterfølgende Risikoanalyser

Efter første risikoanalyse har gruppen før sprintplanlægning gennemgået tidligere risici, og kort diskuteret om tidligere risici stadig havde samme indflydelse på projektet, og om det var mere eller mindre det sandsynligt om det forekom.

Udover at risikoerne blev opdateret i løbet af projektet, blev modellen også justeret en smule, da gruppen mente at en farveskala fra grøn → rød skabte synlighed i hvor stor indflydelse risici'en har på projektet. Dette kan ses på fig 10.



Figur 10: Risikoanalyse: 08/04/2024

Med retrospektiv kunne grafen opdateres til bedre at reflektere forholdet mellem konsekvenser og sandsynlighed. Hvor linjen i de ovenstående grafer har et proportionel forhold mellem chance/severity, kunne den ændres til at vægte konsekvenserne til en højere grad.

En anden ting gruppen har overvejet, er at anvende en mere diskret farve skala, og begrænse farverne til grøn/gul/rød. Dette skyldes feedback fra vejleder og review, da forskellige nuancer på den kontinuere farveskala delvist er arbitrere.

De planlagte 2 ugers mellemrum endte med ikke helt at blive en realitet, eksempelvis blev anden analyse blev foretaget d. 15/03/2024 hvor tredje først blev afholdt d. 08/04/2024. Dette skyldes at gruppen vurderet der ikke var sket nogle signifikante ændringer ifht til projektets risici, men derudover også delvist at gruppemedlemmer havde fokus på at overholde andre deadlines.

8 Teknologianalyse:

Til projektet er der foretaget en teknologianalyse, i rapporten vil der tages udgangspunkt i Lyssensorer, for analysen af resterende komponenter se [3]. Analysen har primært haft fokus på HW-komponenter til projektet, frem for software og andre teknologiløsninger (Communication Busses, design patterns, etc), dette bliver omtalt i senere i rapportien i afsnit 8.5.

Gruppen har valgt at lave teknologianalysen i 4 skridt:

1. Hvad skal projektet indeholde for at opfylde kravspecifikationen.
2. Hvilke krav stilles der til de enkelte komponenttyper.
3. Hvilke mulige kandidater kunne projektet bruge.
4. Hvilken kandidat passer bedst til kravene.

8.1 1. Overblik over Test

Ud fra projektets kravspecifikation lavede gruppen her blev det vurderet at relevante komponenter til projektet var:

- Lyssensor.
- Lydsensor / Mikrofon.
- Højtalere.
- Forstærker til højtalere.
- Gyroskop

8.2 2. Krav til Komponenter

Kravene til komponenterne lavede gruppen i fællesskab ud fra kravspec. Kravene blev opstillet efter MoSCoW. Til lyssensoren fokuserede gruppen på disse krav:

1. **Skal:** kan aflæses af PSoC.
2. **Skal:** afgive signifikant forskel på ”Mørke”, ”Normal Belysning” og ”Høj belysning fra lomme lygte”.
3. **Burde:** være så lille som muligt
4. **Burde:** kunne forsynes på 5V.

Krav til resterende komponenter kan ses i bilag [3]

8.3 3. Mulige Kandidater

Til hver sensorstype blev en række mulige kandidater blev fundet i EmbeddedStock, hvis alle kandidater var blevet erklæret uegnede til projektet ville gruppen søge bredere efter sensorer der forhåbenligt være bedre egnet til projektet. Mulige lyssensorer:

- TSL2591 high dynamic range light sensor
- TEMT6000 Ambient Light Sensor - SparkFun

8.4 4. Resultater af Test

I understående tabel 5 bliver krav og resultater beskrevet for 2 forskellige lyssensorer, som har været med til at give gruppen indblik i, hvilken lyssensor der bedst har kunne betale sig at vælge. Der er udarbejdet lignende tabeller for resterende komponenter, se [3].

Teknologi Analyse: Lyssensor	
Sensor: Lyssensor TSL 2591	
Sensortype: Lyssensor	
Krav:	Resultater:
<ul style="list-style-type: none"> 1. Skal: kan aflæses af PSoC. 2. Skal: af givesignifikant forskel på "Mørke", "Normal Belysning" og "Høj belysning fra lomme lygte". 3. Burde: være så lille som mulig 4. Burde: kunne forsynes på 5V. 	<ul style="list-style-type: none"> 1. Test af I₂C protokol aflæses fint af PSoC. 2. Sat til "Medium Gain" gav sensoren udmærkede resultater, dog med flere uventede udsving. Testen var ikke kontrolleret og fejlene kunne skyldes andre parametre 3. Sensor er en passende størrelse til projektet. 4. Sensoren er begrænset til 3.6V maks. Test er lavet med 3.3V forsyning fra en Arduino ATMega2560.
Overordnet: Hvor sensoren alt i alt fine resultater, er maks forsyningen på 3.6V et problem da PSoC har et output på 5V.	
Link: https://stock.ece.au.dk/components/details/12082	
Sensor: TEMT6000 Ambient Light Sensor - SparkFun	
Sensortype: Lyssensor	
Krav:	Resultater:
<ul style="list-style-type: none"> 1. Skal: kunne aflæses af PSoC. 2. Skal: af givesignifikant forskel på "Mørke", "Normal Belysning" og "Høj belysning fra lomme lygte". 3. Burde: være så lille som mulig 4. Burde: kunne forsynes på 5V. 	<ul style="list-style-type: none"> 1. Output signal aflæses fint af PSoC ADC. 2. Gav ADC værdier (18-35) tildækket/mørke, (129-138) normal belysning og (657-675) ved "Flash" fra telefon. 3. Sensor er en passende størrelse til projektet. 4. Sensoren kan forsynes på maks 6V. Test er udført med 5V forsyning fra PSoC.
Overordnet: Sensoren overholder projektets krav. Ifht. punkt 2, er testen lavet uden justering af sensorens påmonterede OP-Amp, og ved yderligere test kunne sensoren potentielt give bedre resultater.	
Link: https://stock.ece.au.dk/components/details/264	
Konklusion:	
Alt i alt har <i>TSL 2591</i> stort potentiale og rig mulighed for videre test. Flere funktioner er ikke omfattet af testen, og burde per datablad have en større precision end <i>TEMT6000'en</i> . Med dette sagt, føler gruppen det ville være unødvendig tid at lave en spændingsregulator til en enkelt sensor da <i>TEMT6000</i> også overholder alle krav.	

Tabel 5: Sensortest af Lyssensor

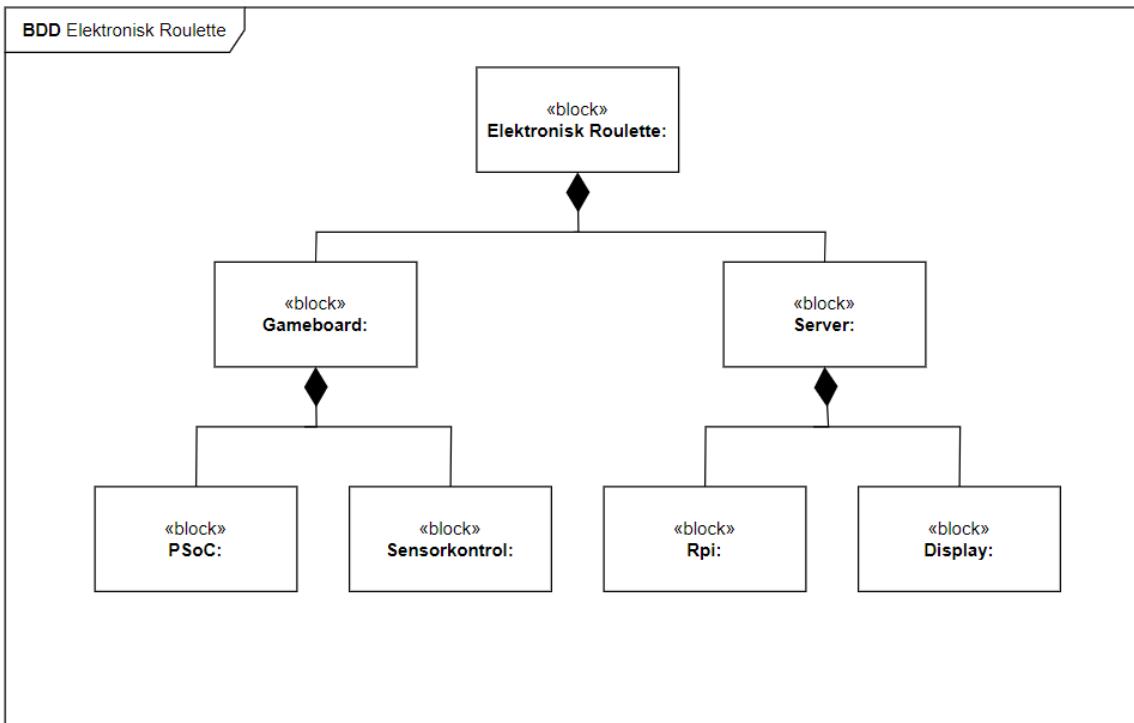
8.5 Reflektion over teknologianalysen

Hvor gruppen alt i alt er tilfreds med hvordan analyse af hardware komponenter er foretaget, skulle der havde været mere fokus på software løsninger i teknologianalysen. Et eksempel på dette var at kommunikationen mellem RPi og PSoC blev valgt til I2C uden større analyse. Dette endte med at skabe nogle problemer og kunne været sparet, hvis der var diskuteret andre muligheder tidligere i forløbet, da gruppen endte med at bruge SPI frem for I2C. Derudover kunne andre valg, som React til GUI, også være overvejet, som er ting der skulle undersøges nærmere hvis analysen skulle gøres om.

Et anden erfaring gruppen har fået er at gruppen skal se teknologi analysen som en kontinuert arbejde, frem for en ”one and done” håndtering som gruppen endte med at bruge i projektet.

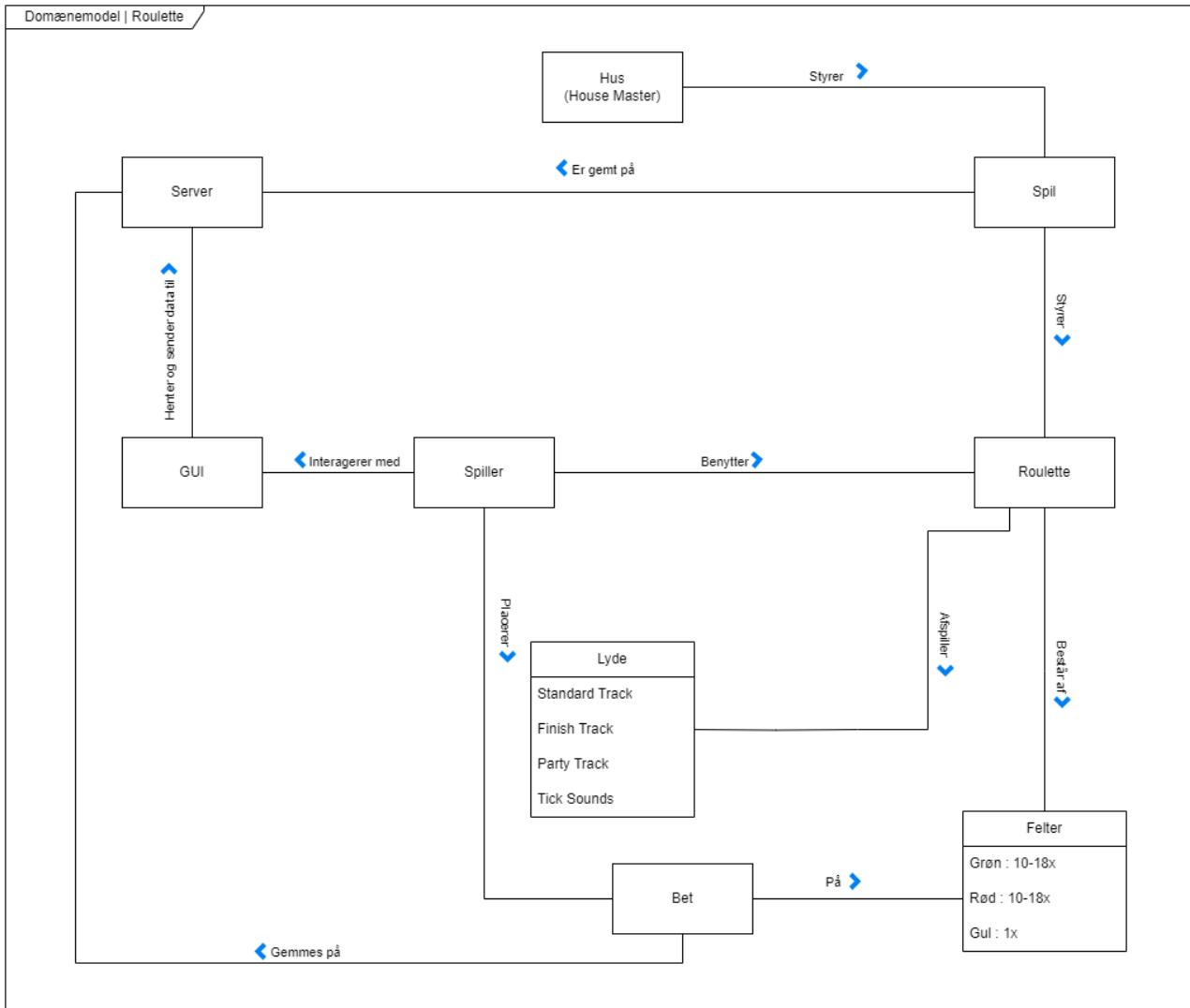
9 Systemarkitektur

Projektets systemarkitektur er opbygget ud fra BDD'en set på fig 11. Figuren er lavet med udgangspunkt i at skabe overblik, hvor enkelte komponenter i de enkelte blokke beskrives dybere i HW-arkitekturen.



Figur 11: Overordnet BDD for den Elektroniske Russiske Roulette

På Figur 12 ses domænemodellen, som har til formål at skabe overblik over systemet.



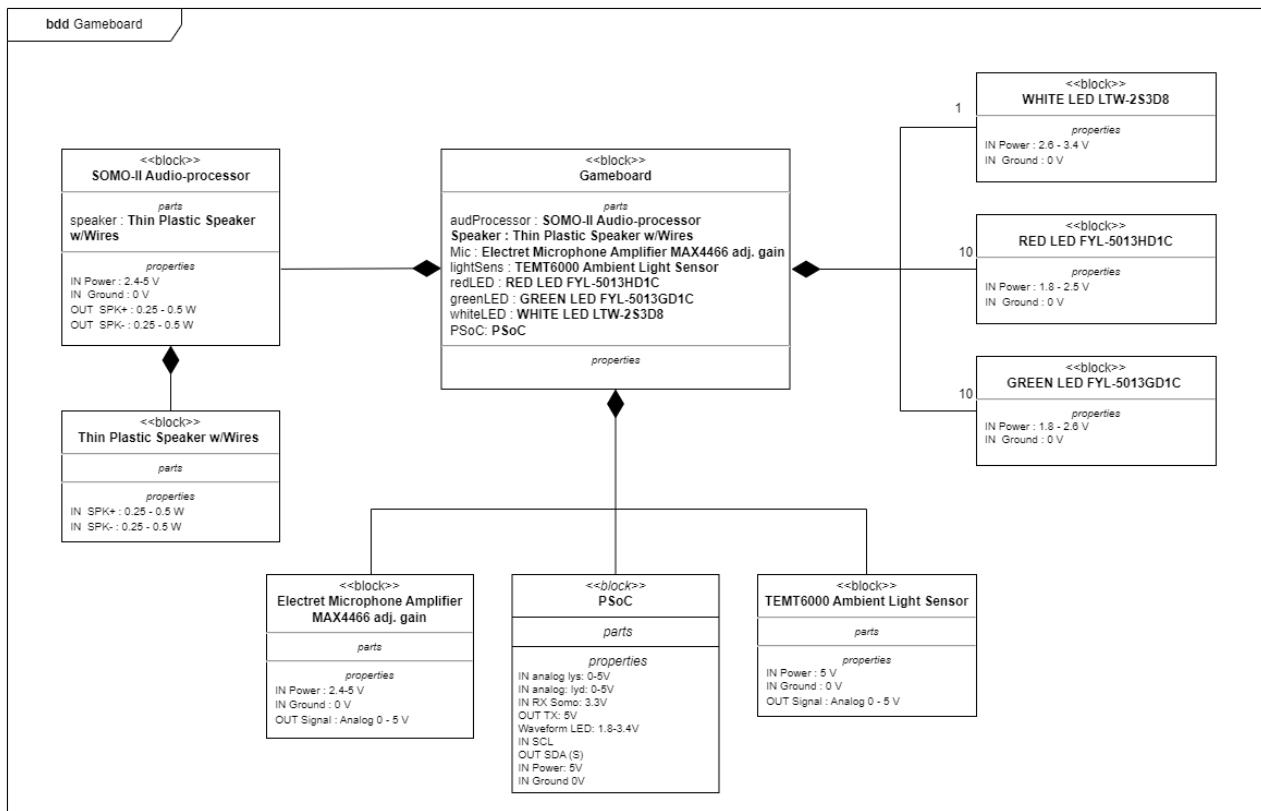
Figur 12: Domænemodel over systemet

På figuren ses det, hvordan systemet fungerer udefra, var man en person, som ikke havde kendskab til det. Der ses f.eks. hvordan rouletten består af både felter, og også afspiller 4 forskellige lyde. Formålet med domænemodellen er ikke, at navngive/oplyse forskellige komponenter, som systemet består af, men i stedet give et generelt overblik over det.

9.1 Hardware Arkitektur

Ud fra den generelle BDD, set på Figur 11, uddyber projektets HW-arkitektur blokkene GameBoard og UserInterface. Rapporten tager udgangspunkt i opbygningen af BDD og IBD for GameBoard, hvor UserInterface vil kunne findes i bilag [4].

9.1.1 Block Diagrammer



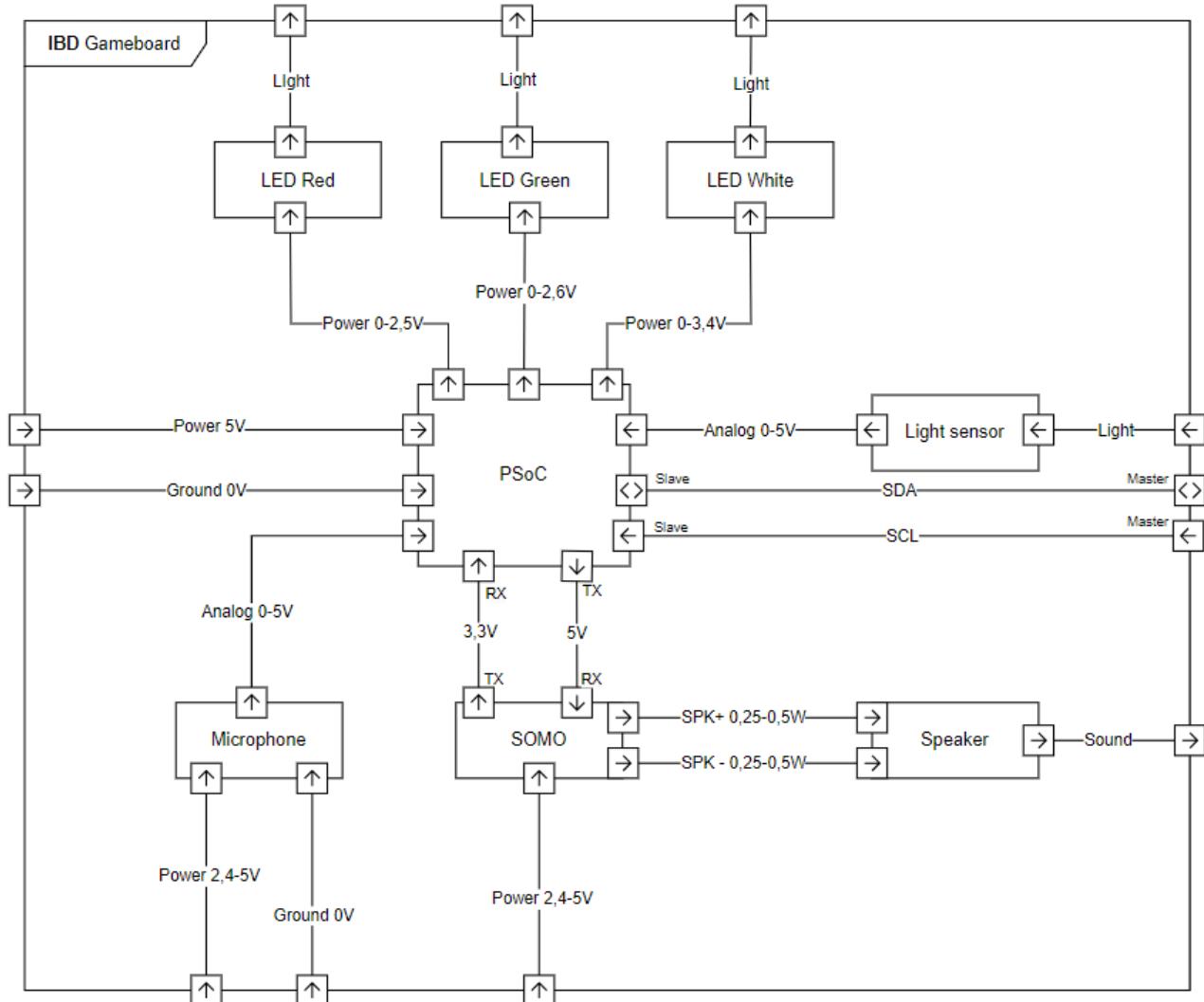
Figur 13: BDD Gameboard

Blokken GameBoard er en samlet blok for den fysiske roulette. På Figur 13 illustreres, hvilke komponenter tænkes, systemet opbygges af, hvor PSoC er et primært komponent, der står for aflæsningen af sensorer og styring af aktuatorer. De enkelte signaler beskrives dybere i tabellen 6

Signalbeskrivelse af BDD Gameboard			
Blok:	Funktionsbeskrivelse:	Signaler:	Kommentar:
PSoC	PSoC styring af Roulettekredsløb og sensorer.	in Analog 0-5V	Analog input fra Lyssensor til ADC
		in Analog 0-5V	Analog input fra Mikrofon til ADC
		in Rx: 3.3V	Serial receive fra SOMO-II
		out tx: 5V	Serial transmit til SOMO-II ¹
		out Waveform: 0-5V	Square Waveform til styring af LED kredsløb
		in/out SDA (S):	I2C slave til/fra RPI
		in scl: clock	I2C Clock fra RPI
		in POWER 5V	Power til Spændingskilde
TEMT6000 - Lyssensor	Måler lysstyrke og sender 0-5V analog signal til PSoC ADC	in GND 0V	Ground til Spændingskilde
		in POWER 5V	Power til Spændingskilde
		out analog 0-5V	Output til PSoC ADC
Electret Microphone Amplifier	Måler lydstyrke og sender 0-5V analog signal til PSoC ADC	in POWER 5V	Power til Spændingskilde
LED - Diverse	Afgive lys og ”simulere” roulettens rotation.	in GND 0V	Ground til Spændingskilde
		out analog 0-5V	Output til PSoC ADC
		in POWER 2.6 - 3.4V	Power til LED - HVID
SOMO - II	Styring af højtalere	in POWER 1.5 - 2.5V	Power til LED - GRØN + RØD
		in GND 0V	Ground til Spændingskilde
		in RX 3.3V	Serial input fra PSoC
		out TX 3.3V	Serial output til PSoC
		out SPK+: 0 - 0.5W	Audio kanal Left til Speaker
		out SPK-: 0 - 0.5W	Audio kanal Right til Speaker

Tabel 6: Signalbeskrivelse af BDD PSoC

Forbindelsen af komponenterne på figur 13 illustreres på IBD'en set på Figur 14, med en tilhørende signalbeskrivelse set på tabel 7



Figur 14: IBD af Gameboard

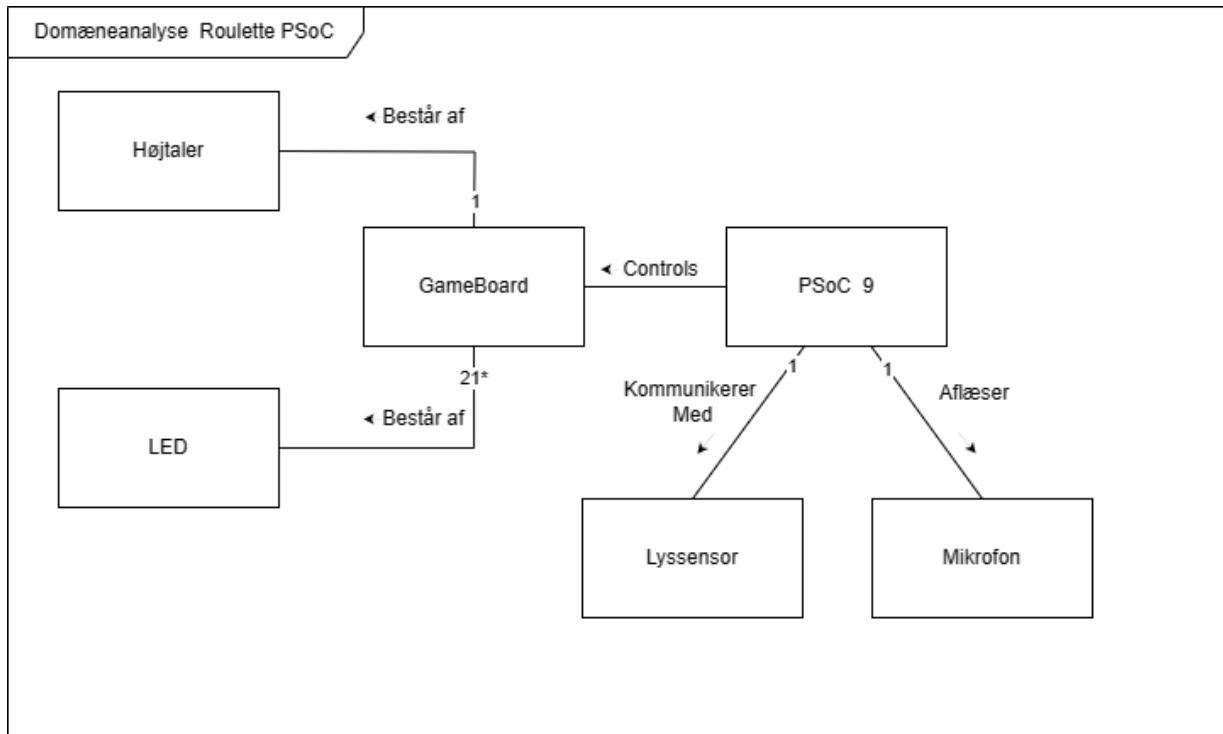
Signalbeskrivelse af IBD Gameboard:				
Signal-Navn	Funktion	Område	Fysisk Port	Kommentar
Power: 0 - 5V	Strømforsyning af PSoC, SOMO, sensorer mm.	Spændingskilde til Diverse	VCC	Generel strømforsyning/VCC
GND: 0V	Ground til PSoC, SOMO, sensorer mm.	Spændingskilde til Diverse	GND	Generel ground/GND
SPK \pm : 0 - 0.5W	Audiosignal fra SOMO-II	SOMO-II til Speaker	SOMO: SPK+, SPK-	Selve lyden, som kommer ud af højtaleren
Analog: 0-5V	Sensor output fra Lyd/Lyssensor	Lyd/Lyssensor til PSoC	PSoC: P0.3 & P0.5	Sensor som detekterer lysniveauet
rx: 3.3V	Serial forbindelse Receive	Mellem SOMO og PSoC	PSoC: P12.7 SOMO: rx	Formodstand til SOMO rx
tx: 3.3V	Serial forbindelse Transmit.	Mellem SOMO og PSoC	PSoC: 12.6 SOMO: tx	5V fra PSoC, formodstand nedsætter til 3.3V
SDA (M): I2C	I2C kommunikation mellem PSoC	RPI og PSoC	PSoC: Pin 5	Simpel I2C kommunikation
SDA (S): I2C	I2C Respons mellem PSoC	RPI og PSoC	PSoC: PIN: 12.1	I2C Respons 1
SCL: clock	I2C Respons fra PSoC	RPI til PSoC	PSoC: PIN: 12.2 RPI Pin 5	I2C Respons 2
Sound	Fysisk lyd opfanget af Lydsensor	Lydsensor	NULL	Lyd opfanget af lydsensor fra omgivelserne
Light	Fysisk lys opfanget af Lyssensor og emitteres af LED	Lyssensor og LED	NULL	Lys opfanget af lyssensor fra omgivelserne

Tabel 7: Signalbeskrivelse af IBD GameBoard

9.2 Software Arkitektur

I de følgende afsnit vil software arkitekturen for systemet blive beskrevet. Der indledes med, en underordnet domæneanalyse af PSoC for Rouletten, hvorefter forskellige sekvensdiagrammer vises.

Gruppen har udarbejdet en domæneanalyse over PSoC, som står for at styre de dele af rouletten, som kan ses med det blotte øje. Sammenlignet med den tidligere domænemodel over systemet på Figur 12, er denne figur/model mere dybdegående med specifikation af forskellige komponenter. På Figur 15 ses denne domæneanalyse.



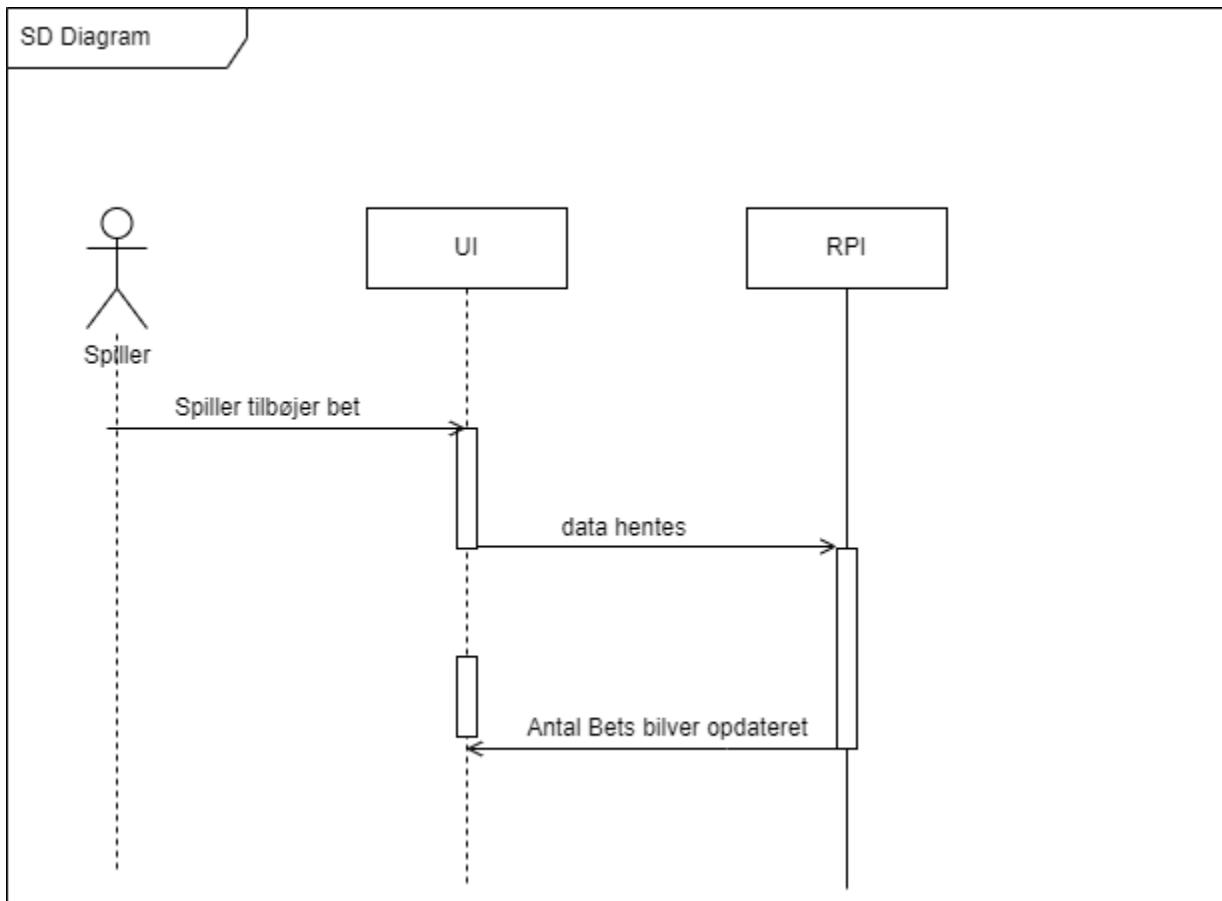
Figur 15: Domæneanalyse af PSoC

PSoC, som er hovedkomponentet i denne model, står for at kommunikere med lyssensoren, samt aflæse det som mikrofonen opfanger. Den står således også for at kontrollere gameboardet, som er spillepladen bestående af minimum 21 lysdioder (LED) og en højtalér, som afspiller forskellige lyde gennem spillet.

Til klassediagrammet tænkes der at inkludere Højtalér og LED i en GameBoard boundary klasse. De to sensorer tænkes også at sættes sammen til en samlet boundary klasse. Derudover tænkes der at dannes en control klasse der står for kommunikation mellem de to. Domænet menes ikke at indeholde nogle rene domain klasser.

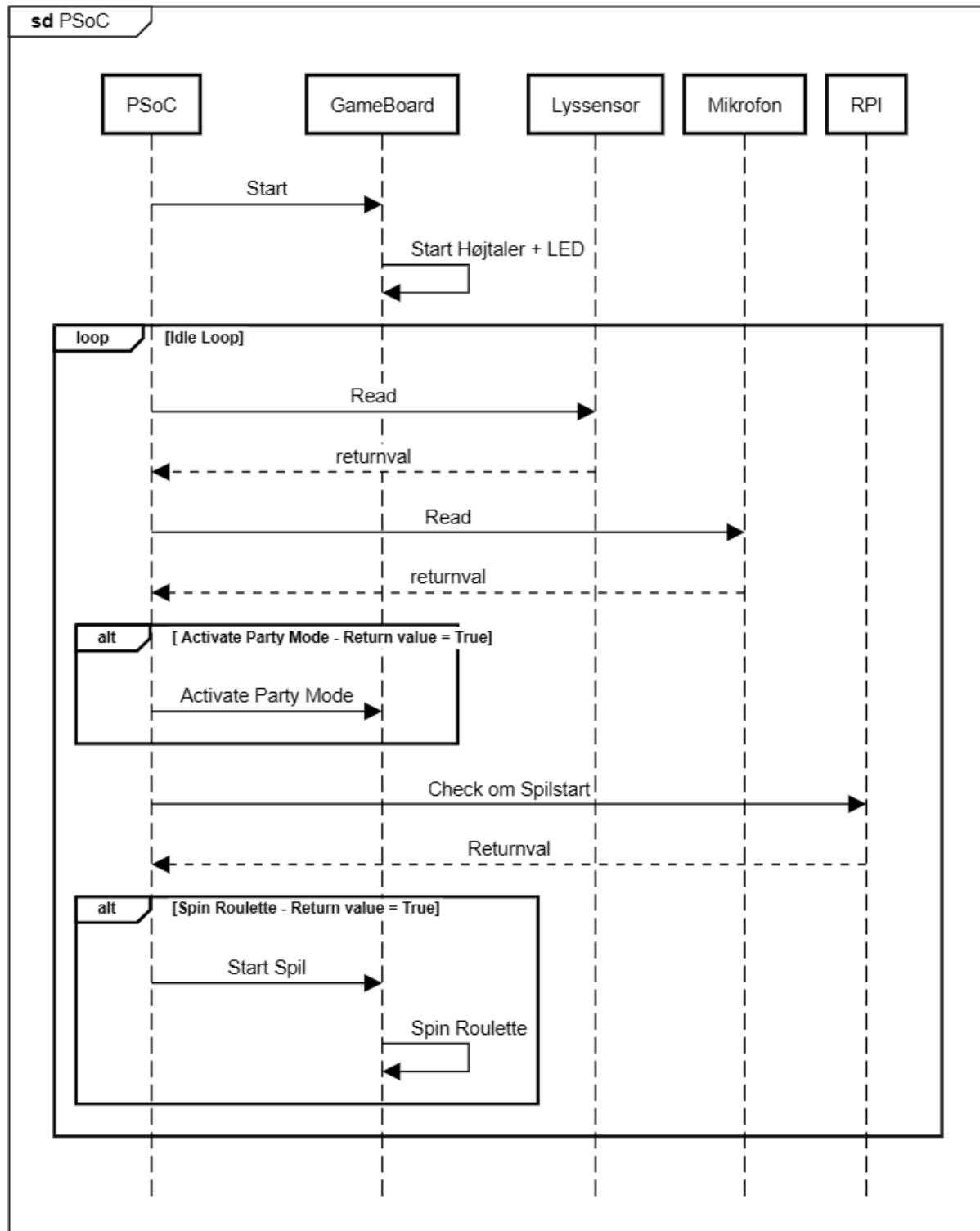
9.2.1 Sekvensdiagrammer

På nedenstående figur 16 ses et simpelt sekvensdiagrammet over placering af bet. På denne figur ses der, hvordan spilleren tilføjer et bet ved hjælp af UI/GUI, hvorefter data bliver forarbejdet af Raspberry Pi, og til sidst antallet af bets bliver opdateret overfor brugeren.



Figur 16: Sekvensdiagram for placering af bet

Sekvensdiagrammet set på fig 17 illustrerer det tiltænkte loop PSoC Styringen foretager for at køre rouletten. I looptet aflæses sensorer og der tjekkes om RPi systemet har initieret kontakt, og reageres på det fornævnte. Sekvensdiagrammet uddybbes i Design og Realiserings afsnittet.



Figur 17: Sekvensdiagram for PSoC

10 Design

I dette afsnit kommer gruppen ind omkring de forskellige designs for både hardware og software. der bliver beskrevet en del omkring elektronikken blandt hardwarekomponenterne i projektet, og hvordan de fysisk arbejder sammen, og der bliver beskrevet, hvordan softwaren for både PSoC og Raspberry Pi er blevet implementeret.

10.1 HW Design

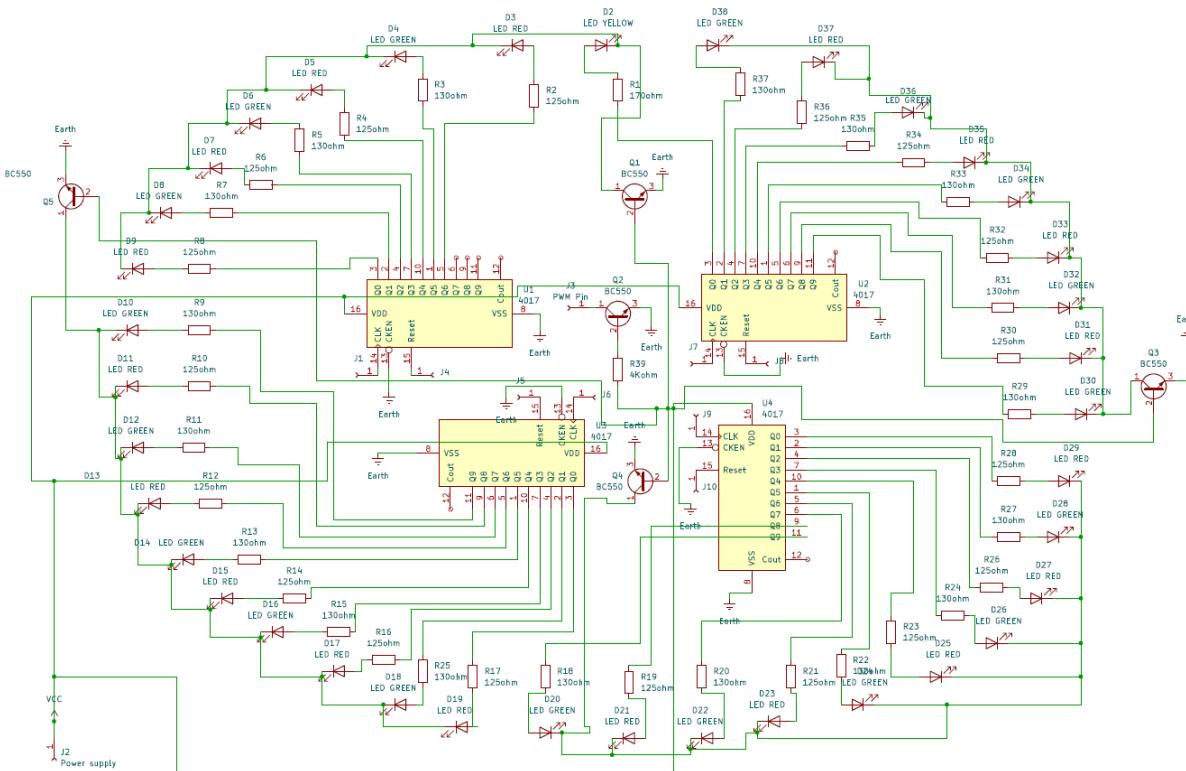
Denne sektion introducerer og beskriver hardwarekomponenterne og designovervejelserne, der ligger til grund for gruppens elektroniske rouletteprojekt. Dette afsnit dækker designprocessen fra initial konceptudvikling til implementering og test af det færdige PCB-layout, samt en detaljeret forklaring af de tekniske beslutninger, der blev truffet undervejs.

10.1.1 Funktionalitet og kredsløbsdesign

Den grundlæggende arkitektur for roulettens kredsløb, består af 4 IC 4017-dekade-tællere, der fungerer som sekventielle logikker, hver forbundet til en separat lysdiode-række og i alt 37 lysdioder. Disse IC'er, der fungerer som tællere og dividers, sender pulseret output til de tilknyttede lysdioder.

Komponent	Antal
4017B IC	4
Transistor BC550	5
LED rød	18
LED grøn	18
LED gul	1
Modstand 130 ohm	18
Modstand 125 ohm	18
Modstand 170 ohm	1
Modstand 1 Kohm	5

Tabel 8: Komponentliste til elektroniske roulette kredsløb



Figur 18: Kredsløbs design af den elektroniske roulette

For at styre strømmen til LED'erne anvendes transistorer, der fungerer som switcher. Hver transistor er tilknyttet en IC 4017 og regulerer strømmen til den tilknyttede lysdiode-række baseret på IC'ens output. Dette giver muligheden for at kunne styre lysdiode-aktiviteten og give systemet en individuel tænding og slukning af hver lysdiode.

10.1.2 Kredsløbs forsyning

Systemet forsynes med strøm fra en 5V strømforsyning via computeren. PSoC'en modtager direkte 5V DC fra hovedstrømkilden og konverterer internt denne 5V til 3.3V, hvilket er nødvendigt for driften af de logiske kredsløb. lysdiode-kredsløbet drives af PSoC'en, som genererer en firkantbølgeform til styring af lysdiodernes tænd/sluk-cyklus. Seriemodstande sikrer, at lysdioderne modtager den rette mængde strøm.

Sensorer, herunder lyssensoren TEMT6000 og lydsensoren Electret Microphone Amplifier, modtager også strøm fra hovedstrømkilden ved 5V DC og sender analoge signaler til PSoC's ADC for præcise målinger af lys- og lydniveauer.

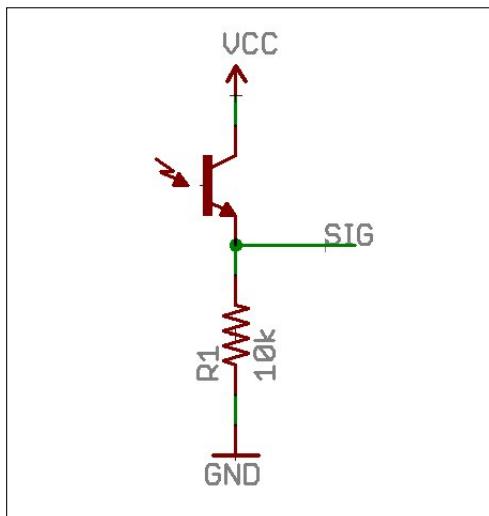
Raspberry Pi modtager ligeledes 5V DC fra hovedstrømkilden. Den kommunikerer med PSoC'en via I2C protokollen. For at sikre stabil drift har Raspberry Pi en intern regulator, som stabiliserer forsyningen til dens komponenter.

10.1.3 Sensore

Der anvendes to hovedtyper af sensorer: en lyssensor og en lydsensor.

Lyssensor: TEMT6000

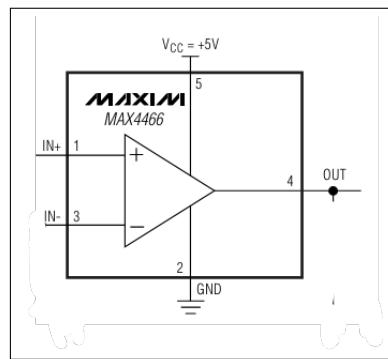
Lyssensoren TEMT6000 bruges til at måle lysintensiteten i omgivelserne. Den konverterer lysniveauet til en tilsvarende analog spænding, som læses af PSoC'ens ADC.



Figur 19: Kredsløbs design af lyssensor
[1]

Lydsensor: Electret Microphone Amplifier

Lydsensoren består af en elektretmikrofon med en forstærker, som vi anvender til at måle klaplydende. Denne sensor omdanner klaptrykket til en tilsvarende analog spænding.



Figur 20: Kredsløbs design af lyssensor
[4]

10.1.4 IC

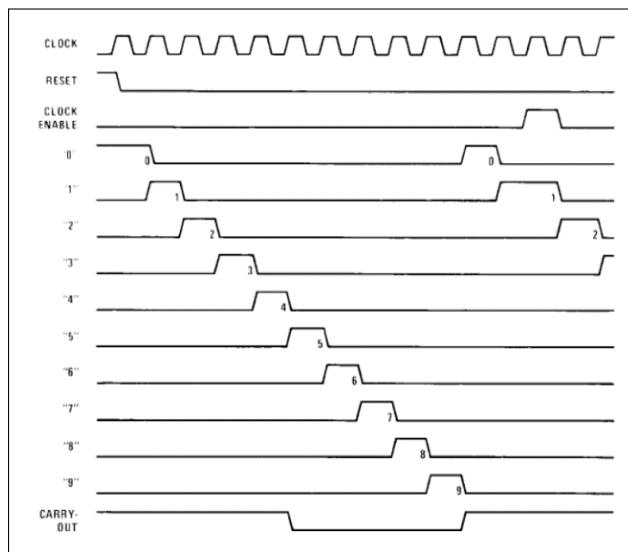
I designet af lysdiode-kredsløbet overvejede gruppen at anvende en IC af klassen MAX7219 frem for den anvendte IC 4017. Gruppen valgte IC 4017 primært på grund af dens funktion som dekade-tæller/divider, hvilket passede til gruppens behov for grundlæggende tælling og sekventiel kontrol af lysdioderne.

Ulempene ved IC 4017 inkluderer begrænset tællekapacitet og manglende indbyggede lysstyringsmuligheder, som findes i mere avancerede IC'er som MAX7219. Da dette projekt ikke krævede disse ekstra funktioner, blev der valgt at anvende transistorer til styring af lysdioderne, hvilket gav den nødvendige kontrol og var mindre komplekst at implementere end en IC som MAX7219.

IC Output pin	Forbindelse
Pin 1 (Decoded Output)	LED
Pin 2 (Decoded Output)	LED
Pin 3 (Decoded Output)	LED
Pin 4 (Decoded Output)	LED
Pin 5 (Decoded Output)	LED
Pin 6 (Decoded Output)	LED
Pin 7 (Decoded Output)	LED
Pin 8 (VSS)	Ground
Pin 9 (Decoded Output)	LED
Pin 10 (Decoded Output)	LED
Pin 11 (Decoded Output)	LED
Pin 13 (CLOCK ENABLE)	Ground
Pin 14 (CLOCK)	
Pin 15 (Reset)	
Pin 16 (VDD)	VDD

Figur 21: Pin tabel for IC 4017

Timingdiagrammet for IC 4017 viser sekvensen af LED-aktivering baseret på inputsignalet og tælletrinnet. Når et pulssignal sendes til IC'en, starter tælleprocessen, hvor hver tælling svarer til en bestemt lysdiode-tilstand. Dette skaber en sekventiel tænding og slukning af lysdioderne i rækken af lysdioder i kredsløbet.

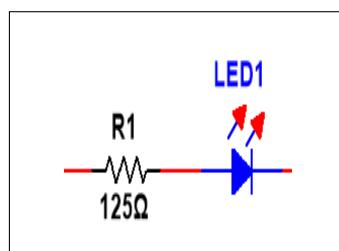


Figur 22: Timing diagram af IC 4017[9]

For yderligere detaljer og beregninger henvises til bilag (Hardware Design).

10.1.5 LED

For at sikre en korrekt strømregulering og beskyttelse af lysdioderne, anvendes modstandene i serie med hver lysdiode-række. Værdierne er blevet udregnet ude fra Ohm's lov, som henvises til i bilag [4].



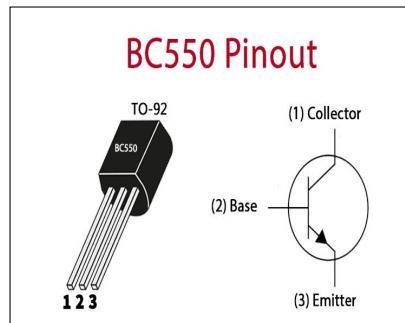
Figur 23: Rød LED

10.1.6 Transistor

I Gruppens design af lysdiode-kredsløbet begyndte gruppen med at vælge BC550 som transistor. Dog blev gruppen nødt til at skifte til BC547 på grund af lagerproblemer. BC547 er en NPN-type bipolær transistor med næsten identiske specifikationer som BC550, hvilket gjorde skiftet problemfrit.

BC547-transistoren blev valgt på grund af dens gode forstærkning, lave cutoff-strøm og lave lejespænding, hvilket gør den ideel til styring af lysdiodernes lysstyrke. Beregninger blev udført for at bestemme den nødvendige basestrøm, der sikrer tænding af lysdioderne uden at overbelaste transistoren. Basemodstande og PWM-signaler fra PSoC'en blev brugt til at styre transistorens åbning og lukning, hvilket muliggør præcis kontrol af lysstyrken.

Detaljerede beregninger og yderligere overvejelser kan findes i bilaget.[5]

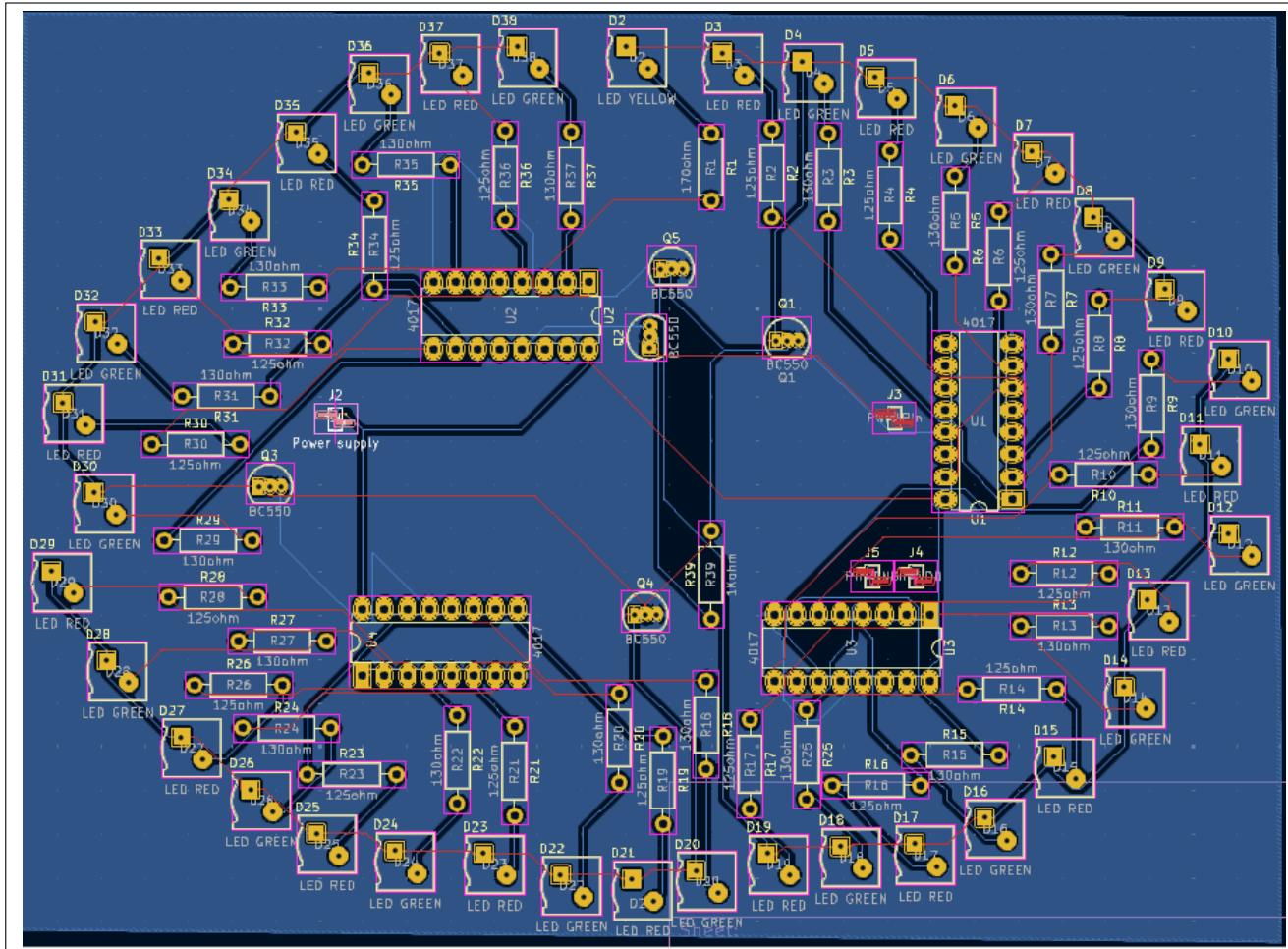


Figur 24: BC550 transistor pinout
[2]

10.1.7 PCB Design

Efter at have udarbejdet og testet en mindre version af det endelige kredsløbsdiagram, er der udviklet et PCD diagram ved brug af tegneværktøjet KiCad.

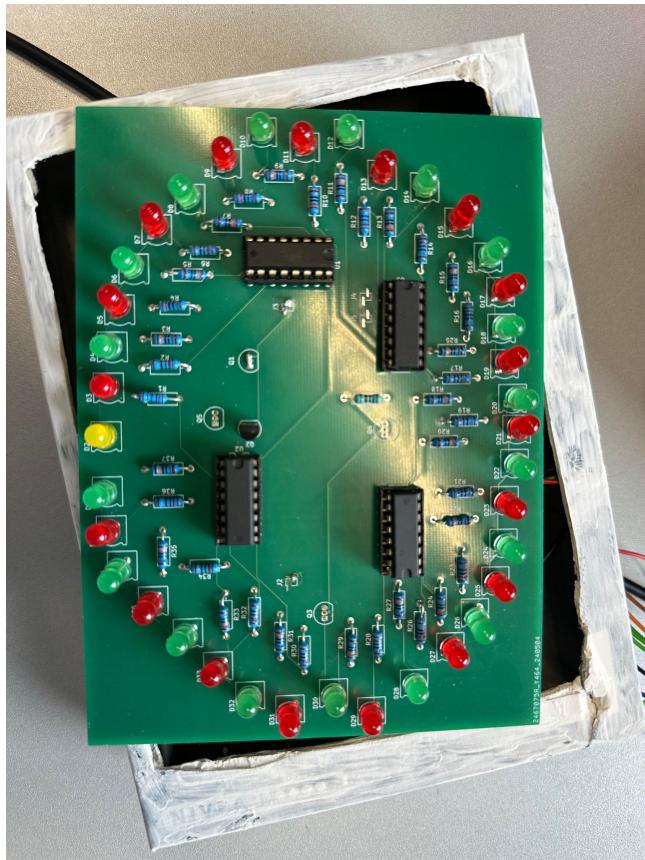
En af de vigtigste overvejelser foretaget i denne fase, var valg af et to-lags PCB. Dette er med til at håndtere de mange forbindelser og komponenter i systemets design uden at øge risikoen for støj, og kunne separere signal- og strømførende spor i designet.



Figur 25: PCB tegning af den elektroniske roulette

Derudover besluttedes der at implementeres en ground plan på hele bundlaget af PCB'et. For at minimere støj og forbedre signalintegriteten i kredsløbet, vil ground planen fungere som et effektivt skjold mod disse forstyrrelser og hjælpe med at reducere potentielle problemer med signalforringelsen.

Den endelige PCB-tegning blev derefter sendt til produktion, hvor det endelige resultat kan ses på fig 26.



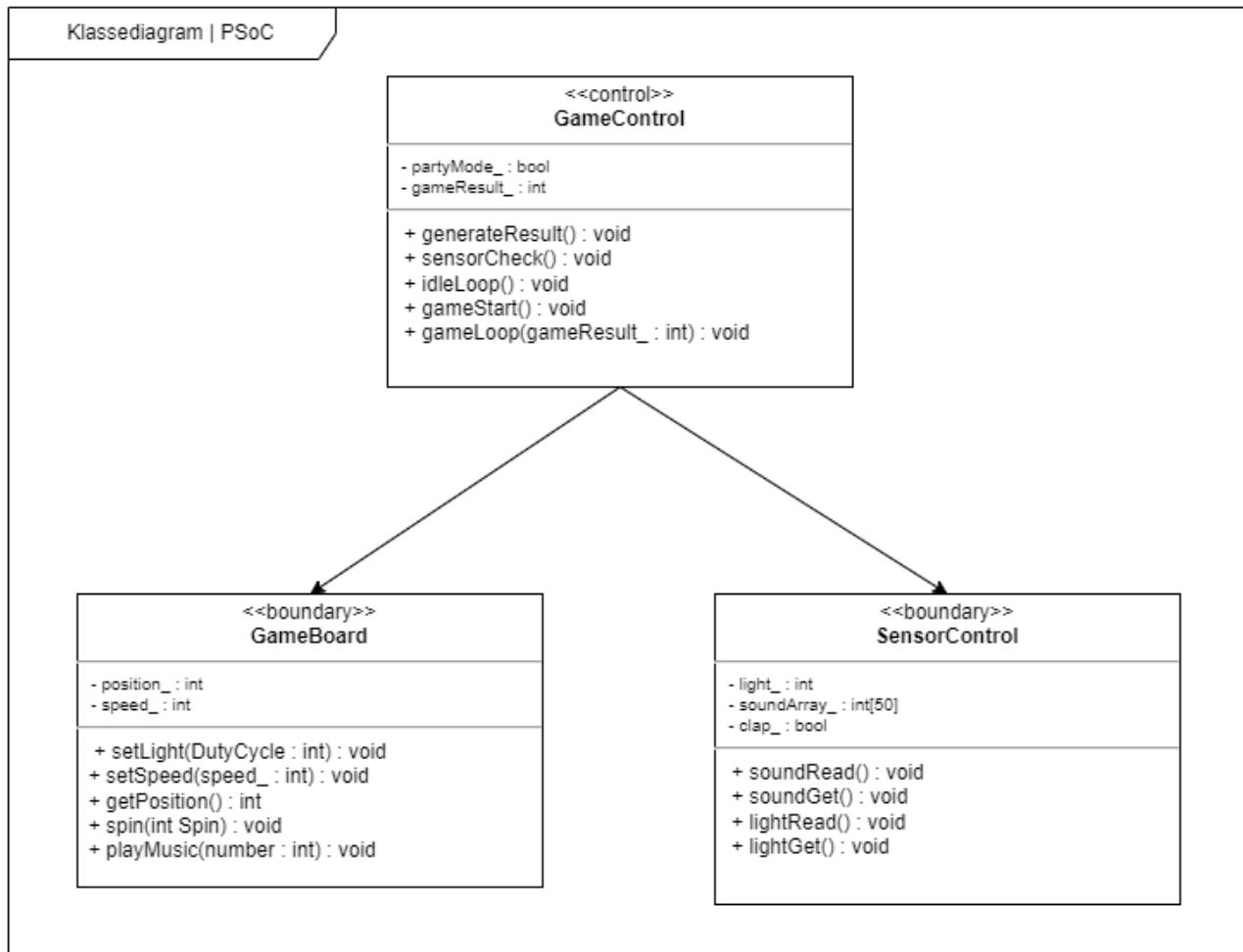
Figur 26: PCB tegning af den elektroniske roulette

10.2 SW Design

Softwaredesignet bygger videre på strukturen fastlagt i arkitektur-fasen, og indeholder STM's (State Machine Diagrams) for henholdsvis PSoC styring og RPI serverhåndtering. Har derudover også lavet nogle udkast til GUI design, for at danne et overblik over, hvordan gruppen vil have serverens brugergrænseflade til at se ud.

10.2.1 PSoC Design

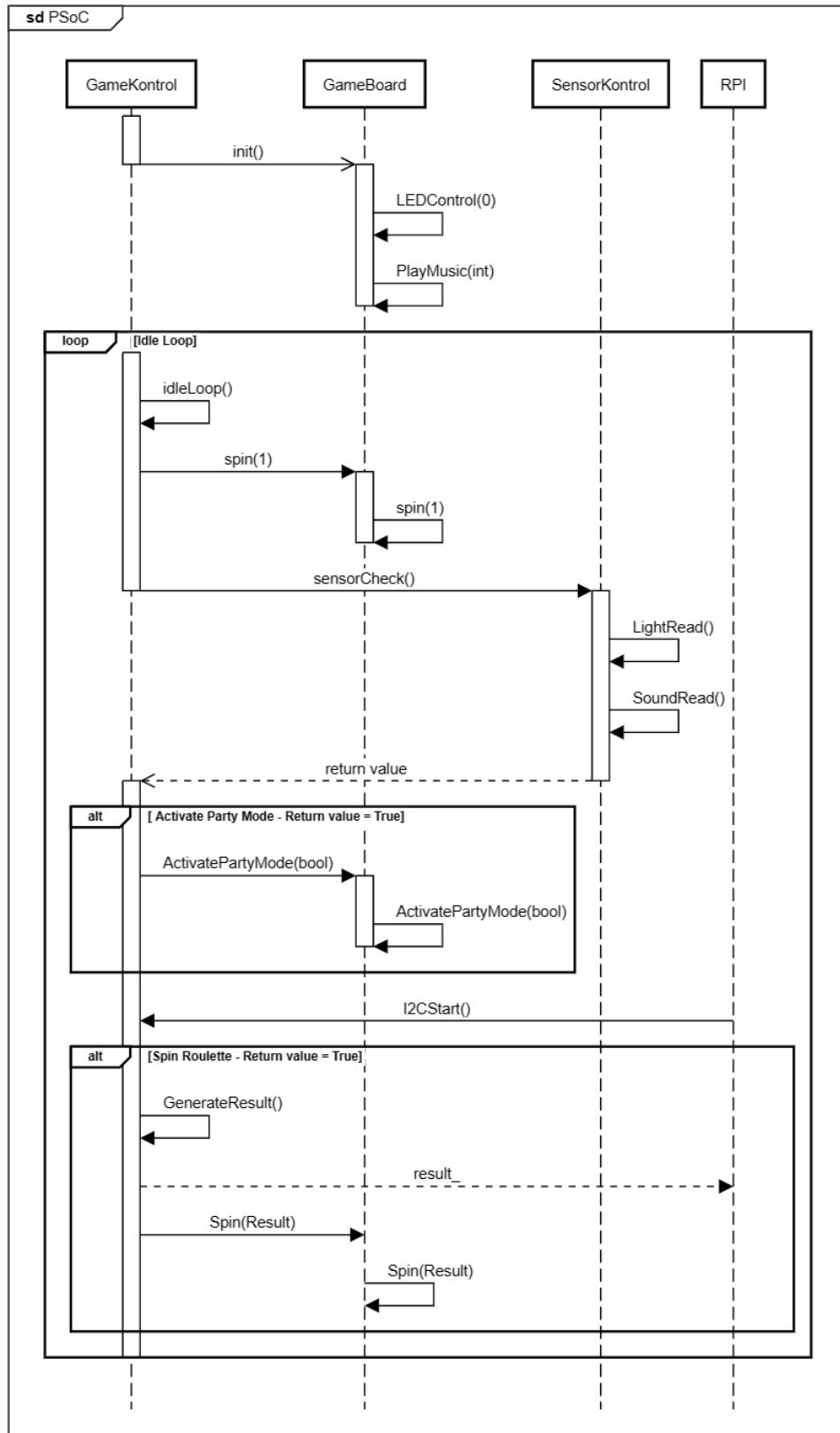
PSoC Klassediagram



Figur 27: Klassediagram for PSoC

Hvor PSoC Styringen er implementeret i C, og derfor ikke understøtter egentlige klasser, er det alligevel valgt at beskrive designet med et klassediagram. I stedet for klasser, er det valgt at arbejde med et tilsvarende struct, med attributterne beskrevet i klassediagrammet. Metoderne er erstattet med en række static funktioner. Klassediagrammet ses på fig 27. Der er desuden lavet tilhørende funktionsbeskrivelser, som kan findes i bilag [5].

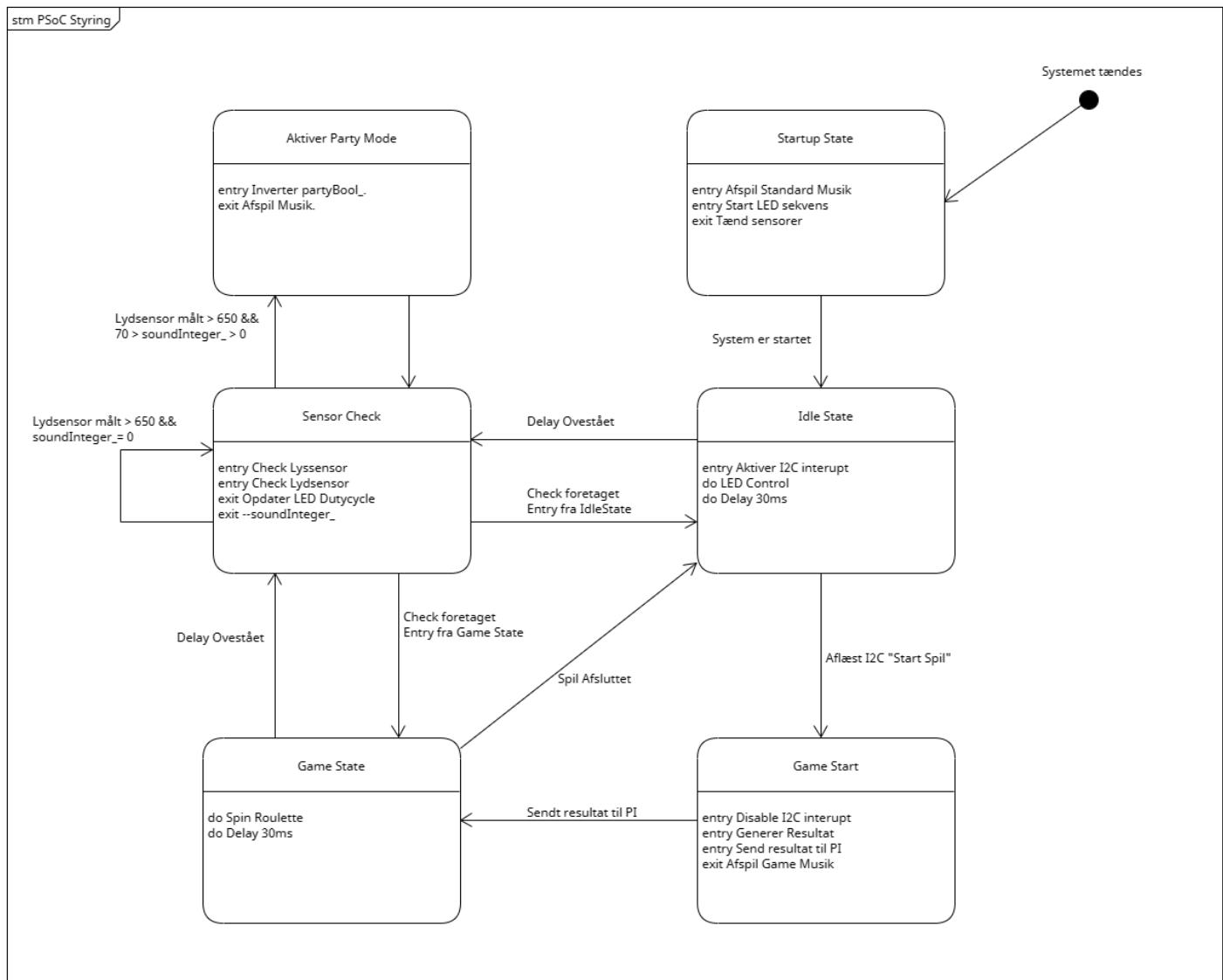
PSoC Sekvensdiagram



Figur 28: UML Sekvensdiagram for PSoC

Figur 28 viser et sekvensdiagram over PSoC styring. Her ses hvordan de forskellige dele af systemet kommunikerer med hinanden ved hjælp af funktionskald. Diagrammet tager udgangspunkt i et ”Idle Loop” styret af klassen GameControl. Hvordan disse interagerer er beskrevet som states på Figur 29

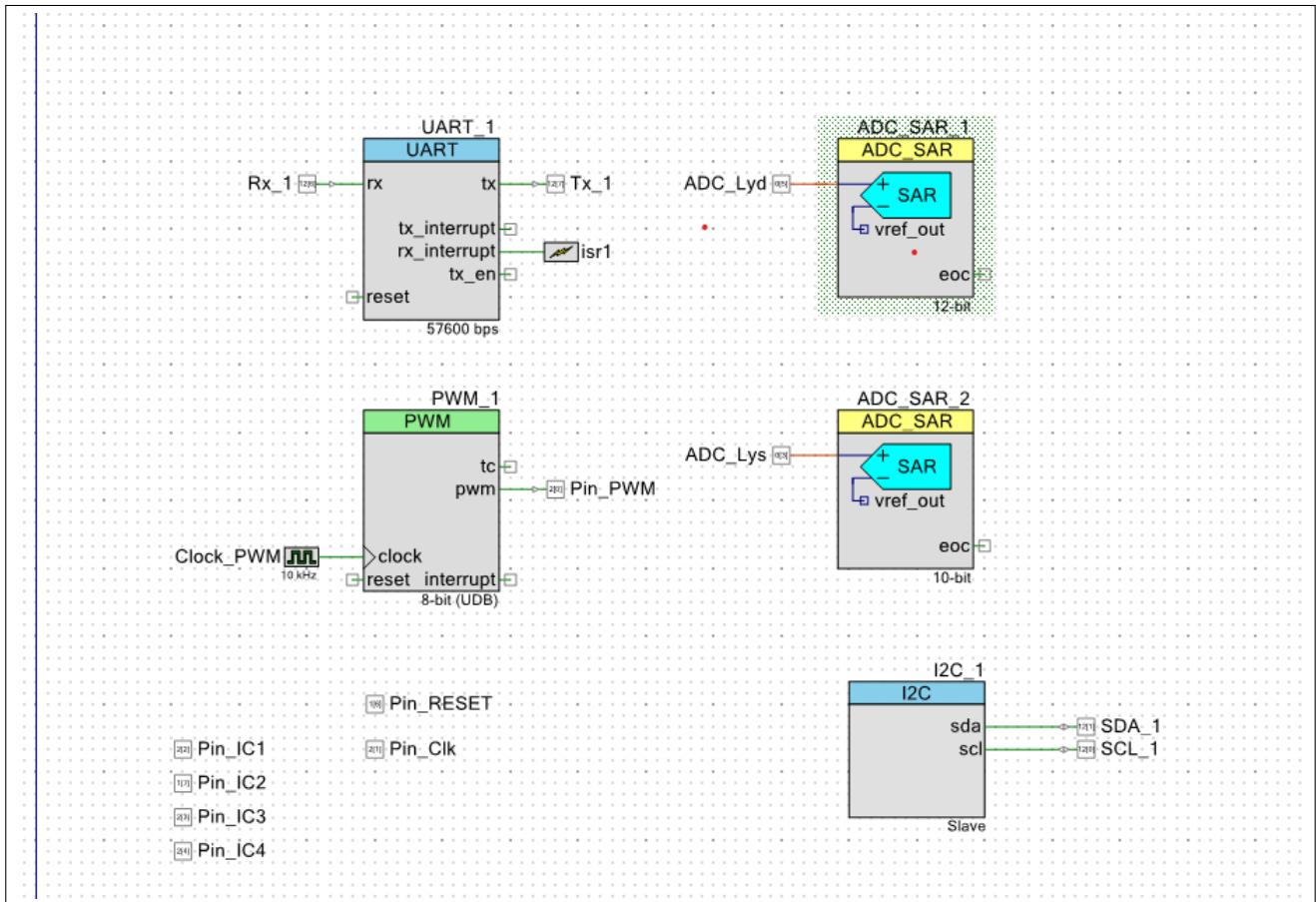
10.2.2 STM PSoC:



Figur 29: STM Diagram af PSoC Styring

Gruppens STM for PSoC styring giver et overblik over PSoC-systemets forskellige tilstande, samt betingelser for overgang mellem tilstandene. PSoC'en kommunikerer sammen med Gruppens Raspberry Pi ved hjælp af I2C, og venter i en ”idle state” indtil den modtager en kommando om at starte spillet. PSoC'en vender tilbage til ”idle state” efter at færdiggjort spil, med mindre at systemets lydsensor opfanger et signal på mere end 650 målt på vores 10.bit AD-konverter, hvorefter ”Party Mode” vil blive aktiveret.

Top Design

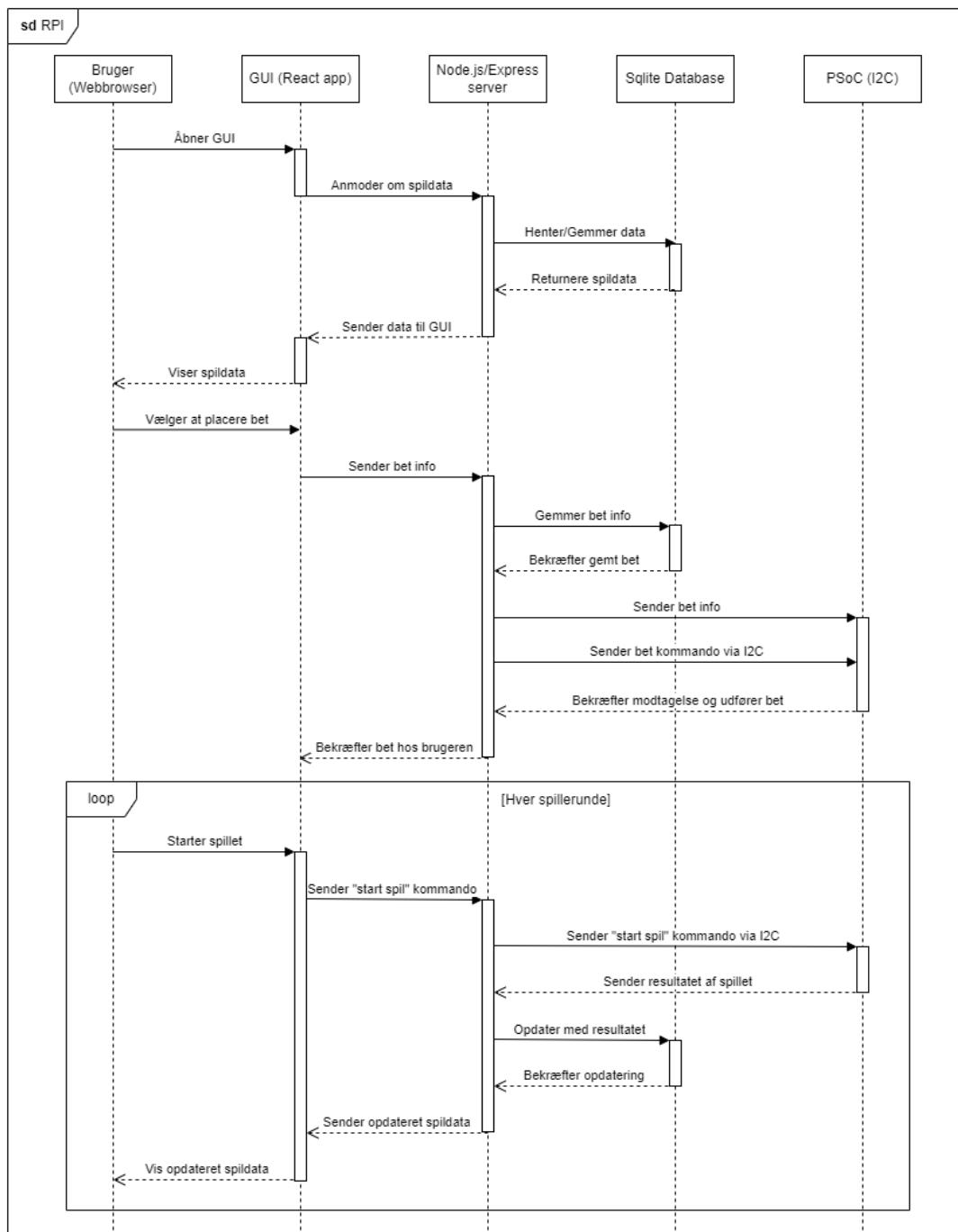


Figur 30: PSoC TopDesign

Grundet der anvendes PSoC creator som editor til styringen er der lavet et topdesign, fig 30. Top designet til PSoC inkluderer UART til styring af SOMO-II, samt test styring. 2 ADC til aflæsning af lys og lydsensorer, med hhv 8 og 12bit. Resolution er valgt lav, da hastighed er vurderet vigtigere end præcision for lyssensor, og høj til lydsensor for at øge præcision til Klap Aktivering af UC 6. 1 PWM styring til lysdiодernes lysstyrke. Samt 1 I2C styring til kommunikation med RPi.

10.2.3 RPI Design

RPI Sekvensdiagram



Figur 31: Sekvensdiagram for Raspberry Pi

Figur 31, som ses ovenfor, vises gruppens sekvensdiagram over Raspberry Pi delen af roulettespillet. Her er kommunikationen dog ikke repræsenteret som funktionskald, men derimod som de handlinger der bliver foretaget mellem de forskellige Raspberry Pi blokke. Diagrammet tager udgangspunkt i et normalt spil, hvor en spiller placerer et bet.

GUI Design

For at roulettespillet kan fungere, så skal der, som tidligere beskrevet, være en brugergrænseflade (GUI) tilsluttet, som både huset og spillere tilgår. Denne brugergrænseflade er styresystemet bag hele produktet, og uden denne, kan man ikke styre spillene eller skabe overblik over spillere, som er en del af spillet, samt deres bets.

Nedenstående figur viser et tidligt udkast af Brugergrænsefladens velkomstside. Resten af GUI-designs findes i bilag [5].



Figur 32: GUI: Welcome

11 Realisering

Realiseringen beskriver, hvordan gruppen har gjort forskellige overvejelser. Realiseringen fra PSoC siden tager primært udgangspunkt i *UC6: Aktiver Party Mode*, da det var et af de punkter, der var den største udfordring i implementering af PSoC delen.

11.1 Realisering og Design af klap aktivering

UC6 Aktiver Party Mode har påkrævet at rouletten er påført en lydsensor², der først og fremmest har kunnet registrere lyde via PSoC'ens ADC. Med dette blev der overvejet forskellige måder at løse klap aktiveringens på, fra "Høj Lyd" til kortlægning af frekvensspektret.

Høj lyd ville være det nemmeste at implementere, da det Problemer ved denne løsning var dog: **1.** Falske Positiver. **2.** Enten pladsmæssigt eller tidsmæssigt uelegant.

Det første punkt ville resultere i at enhver lyd over et givet threshold blive taget i stedet for klap, som ville resultere i mere Party Mode en forventet. Og hvis thresholdet blev sat for højt, ville klap under dette ikke registreres.

Andet punkt ville enten give en usikkerhed i tid, hvis et stort antal samples skulle foretages samt behandles kunne det have en lille effekt på tiden. Dette kunne modvirkes ved at gemme værdier i et array, men med en ADC samplingsfrekvens nær 100kHz ville dette være pladsmæssigt problematisk.

Derfor endte gruppen på at foretage en FFT til at kortlægge frekvensspektret af lydene registreret af lydsensoren.

Implementering tager udgangspunkt i at frekvenserne i et klap ligger mellem 2000-2800hz [5]. Et CMSIS bibliotek bliver anvendt og der er taget inspiration i eksempel [7] til PSoC koden.

Derefter kom der to problematikker ved : Samplefrekvens, og antallet af samples. Begge dele har indflydelse på hvor præcis funktionen ville ende med at blive. Fra Digital Signalbehandling vides det, at frekvens bins fra en FFT udregnes ved $\frac{\text{SamplingsFrekvens}}{\text{Samples}}$. Samplings frekvensen har været en lille smule begrænset ved 12bit ADC på PSoC, med internal clock på min 75kHz.

Det skal dog også nævnes at gruppen havde et mindre bias mod FFT løsningen, for at kunne indkorporere elementer fra kurset DSB i projektet.

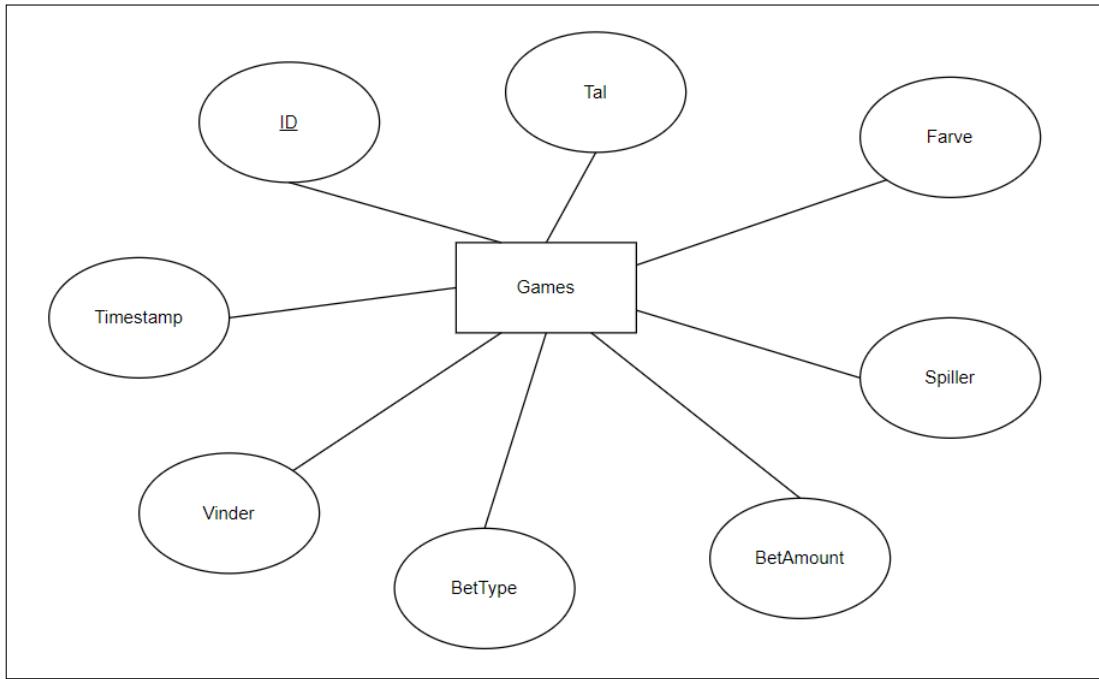
11.2 Database og Backend Server Implementering

I dette afsnit beskrives der, hvordan database og backend server er blevet implementeret ved hjælp af Node.js, IC2-kommunikationsprotokollen, og SQLite. Gruppen har gjort brug af Express.js til at opsætte HTTP-ruter, der håndterer anmodninger fra frontend. Forskellige tabeller for funktioner beskrives ligeledes også.

E/R Diagram for databasen

På følgende Figur 33 kan et E/R Diagram for databses ses. Et E/R diagram er en grafisk repræsentation af forholdet mellem enhederne i en database. E/R står for Entity-Relationship og refererer til et konceptuelt datamodel, der bruges til at beskrive de forskellige enheder og deres relationer i databasen.

²Se Bilag: Teknologianalyse



Figur 33: ER Diagram for Database

11.2.1 Overblik og Funktionalitet

Backend-serveren, udviklet i Node.js og koblet til Raspberry Pi via I2C-protokollen, er afgørende for gruppens realtids spilapplikation. Den sikrer hurtig og stabil kommunikation mellem softwaregrænsefladen og hardwarekomponenterne. Serveren administrerer også spildata effektivt gennem en SQLite-database[10], hvilket understøtter en kontinuerlig og pålidelig brugeroplevelse ved at bevare dataintegritet og sikre let adgang til information.

I2C-kommunikationshandlingen håndteres af moduler, som er defineret i `i2ccommunication.js`, som inkluderer funktioner til at sende og modtage beskeder (`sendMessage` og `receiveMessage`). Fuld opbygning af dette kan ses i bilag [12] Accepttest1.

Funktion	Parametre	Beskrivelse
startServer	Ingen	Initialiserer serveren, konfigurerer middleware og ruter, og starter lytningen på den angivne port. Sikrer, at serveren er klar til at modtage og håndtere HTTP-anmodninger.
handleStartGame	req (HTTP Request), res (HTTP Response)	Behandler anmodninger om at starte et nyt spil ved at interagere med I2C-hardware for at generere tilfældige resultater, evaluerer bets, og gemmer resultater i databasen.
handleSaveGameData	req (HTTP Request), res (HTTP Response)	Modtager og gemmer spildata sendt fra frontend i databasen. Omfatter information som spillerens navn, indsatsbeløb, og resultat.
fetchAllGameData	req (HTTP Request), res (HTTP Response)	Henter og sender alle spildata fra databasen til klienten. Anvendes til at vise historiske data eller til statistisk analyse.
fetchGameDataWithWinners	req (HTTP Request), res (HTTP Response)	Henter data for spil med vindere fra databasen og returnerer disse til klienten. Nyttig for analyse af succesfulde bets.
clearGameData	req (HTTP Request), res (HTTP Response)	Sletter alle data fra databasen. Anvendes til at nulstille databasen mellem spilsessioner eller til testning.
shutdownServer	Ingen	Lukker serveren sikkert, afbryder databasetslutninger og stopper portlytning. Vigtig for at frigøre ressourcer og undgå datakorruption ved nedlukning.

Tabel 9: metode beskrivelse af server

11.2.2 Kommunikation med UI

Serveren anvender Express.js til at opsætte HTTP-ruter, der håndterer anmodninger fra frontend. CORS er konfigureret til at sikre og begrænse ressourcer til GUI og sikrer, at gruppens server kun reagerer på anmodninger fra specificerede oprindelser. Denne konfiguration er afgørende for at opretholde integriteten og sikkerheden for alle datainteraktioner. [8]

Kommunikationsflowet involverer modtagelse af JSON-formaterede anmodninger fra frontend, bearbejdning af disse anmodninger til interaktion med I2C-hardwaren via det brugerdefinerede `i2ccommunication.js` modul, og afsendelse af kommandoer til at starte eller afslutte spilsessioner. Resultaterne transmitteres derefter tilbage til frontend i realtid.

11.2.3 Datahåndtering

Datahåndteringen på backenden er struktureret omkring en central SQLite-database, som illustreret i ER-diagrammet.[3] `Games`-tabellen fanger detaljerede oplysninger om hver spilsession, herunder spillernes bets, spilresultater og tidsstemplere.

Funktion	Parametre	Beskrivelse
saveResult	tal, farve, spiller, betAmount, betType, vinder	Indsætter et nyt spilresultat i databasen. Gemmer detaljer som tal, farve, spillerens navn, indsatsbeløb, bet type, og vinder.
getAllGameData	Ingen	Henter alle spildata fra databasen, sorteret efter tidsstempel i faldende rækkefølge. Nyttigt til historisk analyse eller visning.
getGameData	Ingen	Henter de seneste spildata baseret på tidsstempel. Anvendes til at hente data for de seneste spil.
getGamesWithWinners	Ingen	Henter kun spildata, hvor der er registreret en vinder. Nyttig for statistisk analyse af vindende spil.
clearResults	Ingen	Sletter alle data fra 'games'-tabellen. Anvendes ofte til at nulstille databasen under testning eller mellem spilsessioner.

Tabel 10: Funktions beskrivelse over database

11.3 GUI Implementering

I dette afsnit beskrives der, hvordan GUI er blevet implementeret. Der gives en forklaring af, hvordan det er opbygget med React, som er et JavaScript-framework. Hovedkomponenterne for GUI bliver nævnt, og hvordan de interagerer med backend nævnes også.

11.3.1 Sekvensdiagram for flow

Nedenstående Figur 34 viser et sekvensdiagram over flow for systemet. Dette diagram er med til at vise, hvordan spilleren interagerer med systemet, og hvordan systemet så håndterer disse interaktioner både på serveren og i databasen.

11.3.2 Opbygning med React

Systemet er udstyret med en brugergrænseflade, som er implementeret gennem React, et JavaScript-framework der tillader opbygning af dynamiske og interaktive brugergrænseflader i webbrowsere. Applikationen er hostet som en webserver på Raspberry Pi'en, hvilket gør det muligt for spillerne at interagere med systemet via en standard webbrowser.[13d]

11.3.3 Sideopbygning og Navigation

UI'en består primært af fire hovedkomponenter: HomePage, Game, ActiveBetsPage, og StatisticsPage. Disse sider styres via React Router, som håndterer rutingen i klientens webbrowser og gør det muligt at navigere mellem de forskellige sider uden at genindlæse hele applikationen. [13a]

11.3.4 Interaktion med Backend

Frontenden kommunikerer med backenden ved hjælp af HTTP-Anmodninger for at sende og modtage data. Dette inkluderer at hente seneste spilresultater, starte nye spil, og administrere spillerinformation og bets.

Brugen af asynkrone forespørgsler via fetch API'et gør det muligt for frontend at håndtere data i realtid uden at forstyrre brugerens interaktion med applikationen.[13c]

11.3.5 Styling og Responsivitet

Styling af komponenterne er håndteret ved hjælp af CSS, som er integreret direkte sammen med React-komponenterne eller gennem eksterne stylesheets. Dette sikrer en konsistent og tilstrækkende visuel præsentation, der også er responsiv og tilpasser sig forskellige skærmstørrelser og enheder.

GUI Funktions Tabel

Komponent	Funktion	Beskrivelse
HomePage	Navigation	Starter siden for applikationen hvor brugere kan navigere til spil eller statistik siden.
Game	Spilloogik	Håndterer tilføjelse af spillere, indsats, og interagerer med backend for at starte spillet og opdatere spillerens balance og aktive bets.
ActiveBetsPage	Vis aktive bets	Viser de aktuelle bets og spilresultater. Opdateres dynamisk efter hver spilrunde ved at hente data fra backend.
StatisticsPage	Vis statistik	Viser spilstatistikker og historiske data, hentet fra backend. Giver også mulighed for at rydde spildata via en DELETE request til backend.
MoreStatisticsPage	Udvidet statistik	Viser yderligere statistiske data som f.eks. antal af trukne numre, hentet fra backenden.

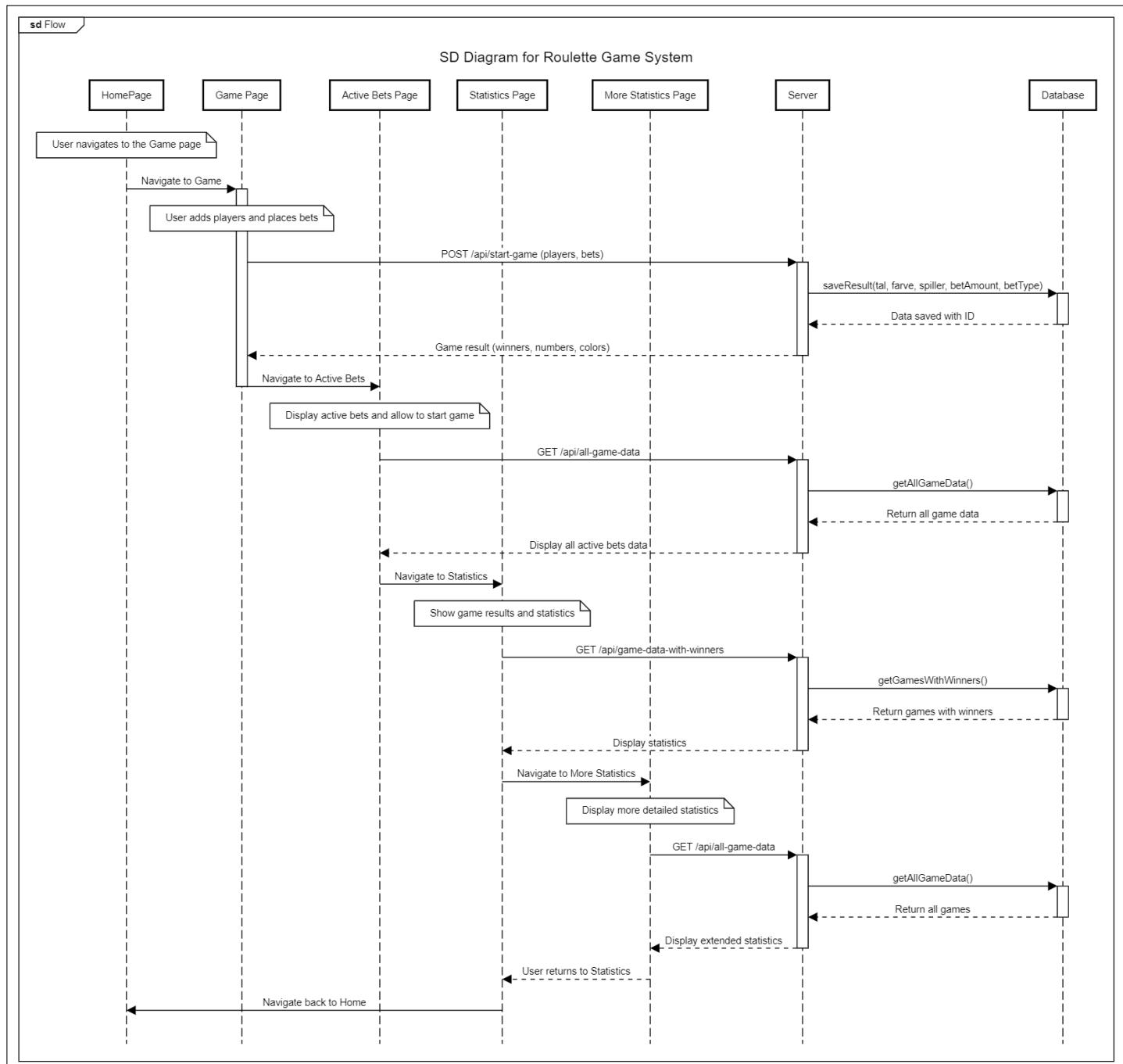
Tabel 11: Oversigt over GUI Pages



Figur 35: Component-Diagram over gui opbygning
[13b]

11.4 Sikkerhed og Datahåndtering

HTTP-requests sikres via moderne sikkerhedspraksis, som CORS (Cross-Origin Resource Sharing), der er konfigureret i backend for at acceptere requests kun fra de godkendte kilder. Dette beskytter systemet mod uautoriseret adgang og datalekage.



Figur 34: Sekvensdiagram over flow



12 Test af Elektronisk Russik Roulette

Gruppen har foretaget en række forskellige tests, som kan opdeles i 3 typer: Modultest, Integrationstest og Acceptttest. En oversigt over alle tests lavet til projeket kan ses på fig 12.

Oversigt af Test

Nr.	Navn	Beskrivelse
Modultest:		
1.	Sensor test	Test af lyd / lyssensor med SAR ADC. Fokus på at tilrettelægge værdier af sensoroutput, eksempelvis ADC værdier af lyssensor i mørk til fuld belysning.
2.	Test af Idle/Gameloop	Test af 13-LED kredsløb, med lyd og aflæsning af lyssensor. Bruger UART til styring.
3.	Test af Klapaktivering	Test af klapaktivering af Bruger UART til udskrivning af ADC aflæsning samt FFT.
4.	PSoC I2C-test	Master/slave test af I2C mellem 2 PSoC, med fokus på slave. Sætter adresse og sender forskellig respons efter input
5.	Tilføj Spille test	Der testes om vi er i stand til at tilføje spillere til roulettespillet, ved at udfyld fejlerne i gui med navn og saldo
6.	Fejlscenarier-test	Test 1: Ugyldig Tilføj Spiller test: Der indtastes ikke et spillernavn, samt en start saldo. Derefter trykkes der på "tilføj spiller" knappen. Test 2: Ugyldig Tilføj Spiller(forlangt navn): er indtastes et spillernavn med mere end 10 bogstaver, samt en start saldo. Derefter trykkes der på "tilføj spiller" knappen. er indtastes et spillernavn med mere end 10 bogstaver, samt en start saldo. Derefter trykkes der på "tilføj spiller" knappen.
7.	Tilføj BET-test	Der testes om vi er i stand til at tilføje et bet til roulette spillet, Ved at der indtastes et bet (100.00) og der vælges bet type(Rød), samt at saldoen ændres (0). Derefter trykkes der på "Place Bet" knappen
8.	Pages-test	Der bliver her testet, om man kan navigere mellem siderne og bruge alle funktionerne på de forskellige sider, såsom at starte et spil og om siden korrekt henter/sender og viser data.
10.	Clear data-test	Vi tester om vi kan clear game data og derved fjerne de seneste resultater fra StatisticsPage og MoreStatisticsPage ved at trykke på knappen Clear Game Data.
Integrationstest:		
12	Samlet PSoC Test	Overordnet test af PSoC styring uden I2C. Fokus på nøjagtighed af sensor-output imens rouletten er aktiv. Bruger UART til styring af spilstart.
13	RPi-PSoC I2C protokol	I2C test. Fokus på at kunne styre PSoC ved brug af RPi og modtage tilfældigt genereret tal i respons.
14	Fuld Integration	Test af styring af PSoC via RPi Server. Fokus på at få systemet klar til accepttest og finde eventuelle fejl.
Accepttest:		
Accepttest 1		Første Accepttest af original foretaget d. 20/05/2024
Accepttest 2		Andet forsøg på accepttest, udført ud fra opdateret Accepttest specifikation tilpasset ændret vision efter design og realisering. Ikke udført per afleveringsdato 31/05/2024

Tabel 12: Oversigt over Test af "Elektronisk Russisk Roullette"

12.1 Modultest

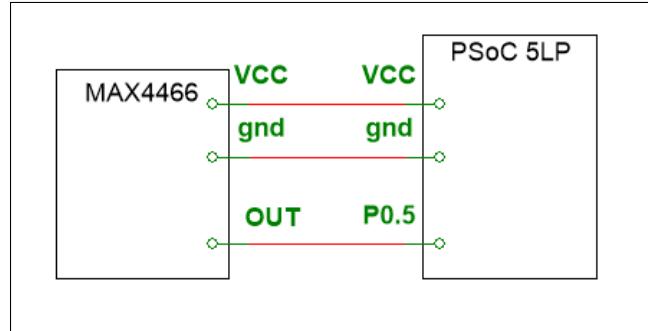
I afsnittet vil der gennemgåes de modultest gruppen har vurderet var vigtigst. Testene i afsnittet er: 3, fra tabel 12. Resterende modultest kan ses i [8]!

12.1.1 Test af Klapaktivering

Med udgangspunkt i problematikkerne
omtalt i afsnit 11.1 Realisering af Klap Aktivering.
Hvor der i afsnittet vil testes "Høj Lyd" samt
løsninger med FFT array længder på 128 og
256. Koden til test kan findes i TestCode/KlapTest.

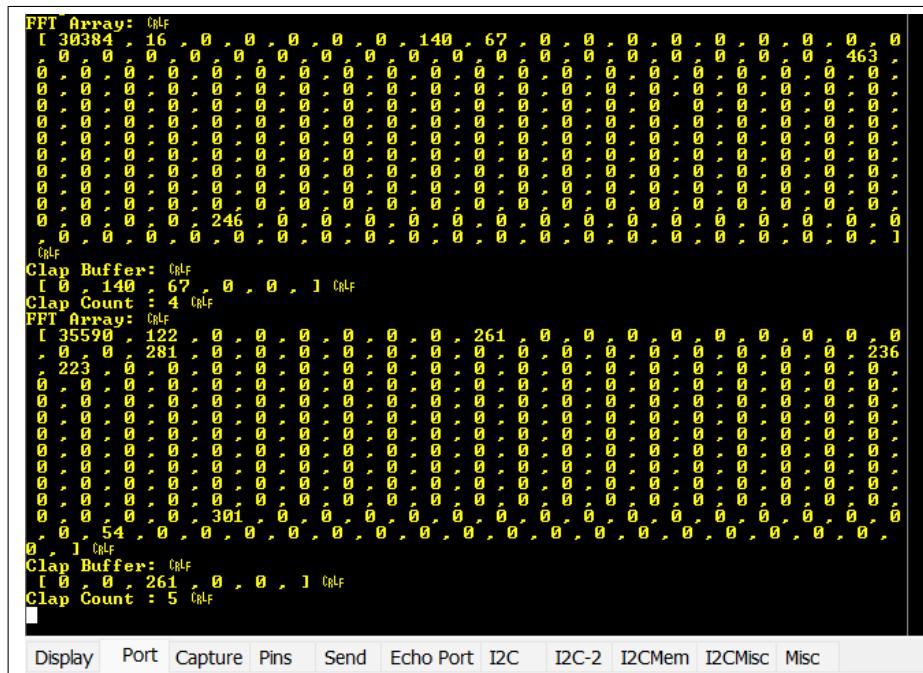
Testen foretages ved kontinuert
ADC aflæsning, hvor der bliver klappet 10
gange med en afstand på ca. 10cm fra Lydsensoren.

Alle testene er foretaget i en relativt støjfri lejlighed, og resultater opnået er muligvis bedre, dette vil også omtales kort i Accepttest Afsnit.



Figur 36: MultiSim Diagram af Lydsensor

Testopsætningen kan ses i på fig 36.



Figur 37: Realterm Output fra test af 256 array

På figur 37 ses der realterm output fra testen med FFT lavet over 256 samples, hvor hvert index svarer til ca. 300hz. Index 6- 10^3 highlightes i Clap Bins.

³Ca. 1800-3000hz

Resultater for Klap Aktivering:

FFTLength = 126: For 126samples blev 4/10 klap registreret

FFTLength = 256: For 256samples blev 6/10 klap registreret

Højlyd: For ren ADC blev 12/10 klap registreret

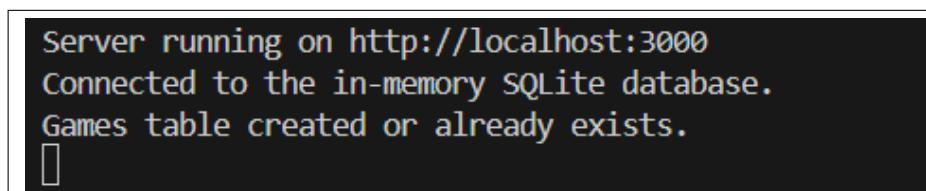
FFT løsningen bliver brugt frem for de andre, grundet at "Høj Lyd" løsningen opfangede falske positiver, og hvor dette muligvis kunne modarbejdes ved et højere threshold, blev testen lavet i et lokale med relativt lidt støj. Videre kode anvender array længde på 256, men hvis tiden tilader kunne der udføres videre test med længde 512 eller 1024.

12.1.2 Modultest af Gui og Backend

For at sikre, at backenden korrekt modtager og behandler data, har gruppen implementeret en række tests ved hjælp af en dummy server. Denne server simulerer backendens funktioner uden at påvirke den faktiske produktionsdatabase. Denne tilgang giver gruppen mulighed for at validere funktionaliteter som dataindsamling, query-håndtering, og fejlhåndtering under kontrollerede forhold.

Funktionaliteter testet omfatter:

- Modtagelse af spildata (POST /save-data) og validering af at data er korrekt gemt i databasen.



```
Server running on http://localhost:3000
Connected to the in-memory SQLite database.
Games table created or already exists.
```

Figur 38: Server og database er køre og er forbundt

- Hentning af spildata (GET /api/all-game-data og GET /api/game-data), herunder sikring af, at alle spildata er tilgængelige og præcist repræsenteret.
- Håndtering af spilresultater med vindere (GET /api/game-data-with-winners), hvilket tester systemets evne til at filtrere og returnere specifikke datasæt.
- Rydning af spildata (DELETE /api/clear-game-data), som tester systemets kapacitet til at nulstille databasen, en kritisk funktion for vedligeholdelse og fejlretning.

```
A row has been inserted with rowid 1
Fetched 1 games with winners.
Fetched 1 games with winners.
Fetched 1 games.
Fetched 1 games.
Fetched 1 games with winners.
Fetched 1 games with winners.
Cleared all game results, 1 rows affected.
```

Figur 39: indhente data og cleared data

Disse tests udføres ved at interagere med vores GUI på localhost, hvorfra der sendes forespørgsler til serveren. Responsen fra disse forespørgsler sammenlignes med de forventede resultater for at sikre nøjagtighed.

Frontend-testene fokuserer på brugerinteraktioner og den visuelle præsentation af data hentet fra backenden. Der anvendes manuelle teststrategier for at sikre, at brugergrænsefladen er intuitiv, responsiv og korrekt viser opdateringer baseret på brugerhandlinger og backend-responser.

Kerneområder testet inkluderer:

- Navigation mellem forskellige komponenter i applikationen (HomePage, Game, ActiveBetsPage, StatisticsPage), hvilket sikrer, at ruter fungerer korrekt og uden fejl.
- Interaktivitet i Game-komponenten, hvor brugernes evne til at tilføje spillere, placere bets og se resultater testes grundigt.
- Funktioner i ActiveBetsPage og StatisticsPage, som skal korrekt vise de seneste spilresultater og statistikker hentet fra backenden.

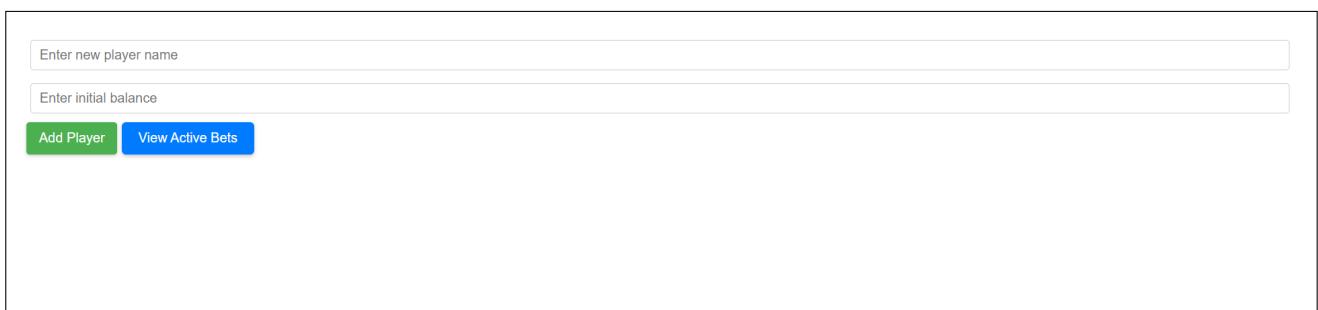
Fremfor I2C-kommunikation er der implementeret en funktion der genererer og sender et tilfældigt tal, for at simulere hvordan simulere spillets gang.

Tilføj Spiller

Der testes om gruppen er i stand til at tilføje spillere til roulettespillet

Test: Der indtastes et spillernavn (Sammi), samt en start saldo (100). Derefter trykkes der på ”Add player”knappen.

Resultat: En spiller er blevet tilføjet en spiller med navn Sammi og en start saldo på 100



Figur 40: GUI: Tilføj spiller



Figur 41: GUI: Spiller Tilføjet

ActiveBetsPage

Der testes om gruppen er i stand til at starte et roulettespil.

Test: Der trykkes på knappen ”start spil”.

Resultat: Der startes et spil, og gruppen kan se om, der er en vinder samt, hvilket tal rouletten er landet på.



Spiller	Bet
Sammi	Red - 100

Latest Game Result:

Drawn Number: 12
 Color: Red
 Winner(s): Sammi

Figur 42: GUI: Start af spil

Yderligere test kan findes i bilag[8]

Fejlhåndtering og Brugerfeedback

- Test af korrekt fejlhåndtering og brugervisning af fejlmeddelelser, hvis der opstår netværks- eller serverfejl.
- Validering af feedback mekanismer som alerts og statusmeddelelser, der informerer brugeren om handlingernes udfald.

Disse komponenter af testplanen sikrer, at både frontend og backend fungerer som forventet, og at applikationen som helhed tilbyder en robust og brugervenlig oplevelse. Disse tests er afgørende for løbende vedligeholdelse og forbedring af systemet, og hjælper med at sikre, at nye ændringer ikke introducerer fejl eller nedbryder eksisterende funktionaliteter.

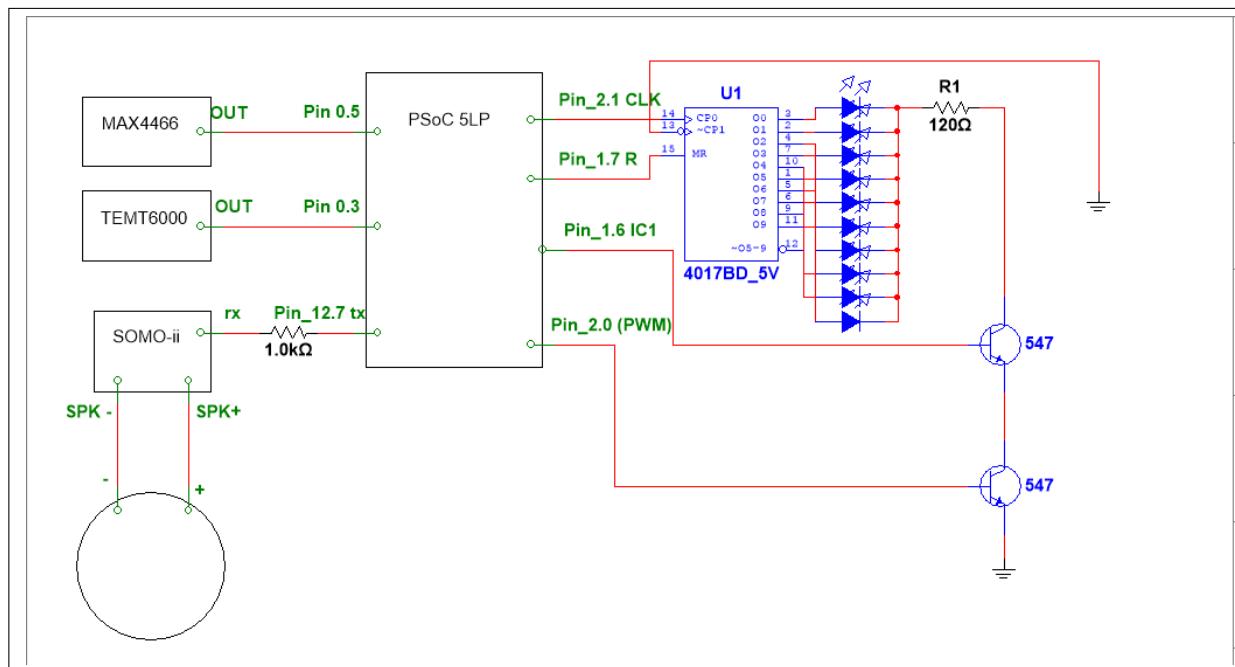
12.2 Integrationstest

Som modultestene har været udført og enkeltdede begyndte at fungere som ønsket, har gruppen foretaget en række integrationstest for at sikre delene fungerer som helhed. Af integrationstestene set i tabel 12 er testen 14, yy og zz inkluderet i rapporten, resterende kan ses i bilag[8]:

12.2.1 Samlet PSoC Test

Den samlede PSoC test laves efter alle modultest, og har fokus på at fange problemer kunne opstå når de forskellige komponenter sammensættes. Andet fokus er at se hvordan nøjagtigheden på Klap Aktiveringens ændres når systemet inkorporerer flere delays og andre opgaver. anvender opsætningen set i fig 43. Testen anvender UART styring til start af game, og anvender topdesignet set tilbage på figur 30.

Til test af lydsensor foretages der som i afsnit 12.1.1 10 klap og der noteres antal registrerede klap. Da klap aktivering deaktiveres under en runde, sikres der at alle klap er udført mens rouletten er idle.



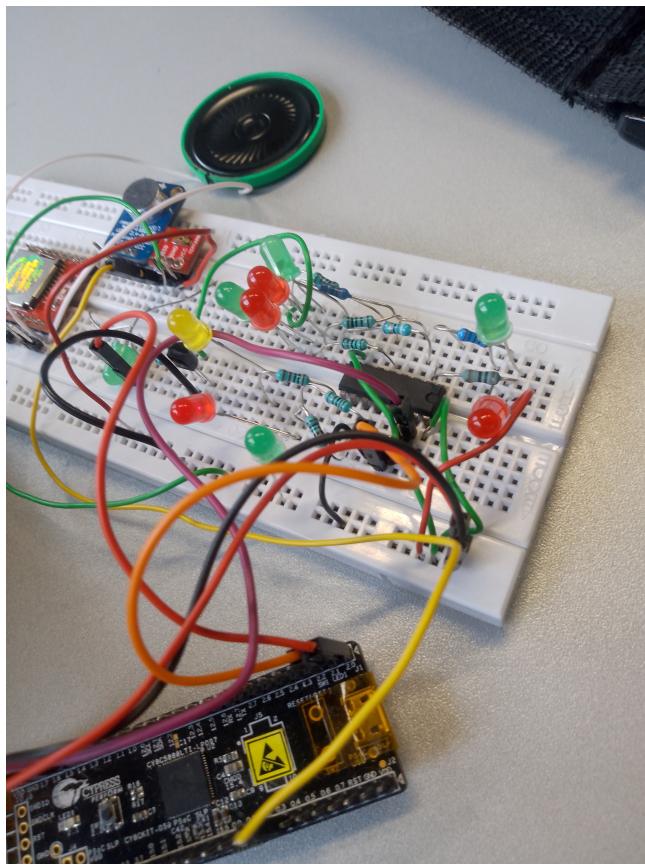
Figur 43: Multisim Diagram af Test Opsætning

Den fysiske opsætning af kredsløbet kan ses kan ses på fig 44.

Resultater for Samlet PSoC Test

Testen endte med at have mixed results, da langt de meste fungerede som ønsket. Dog har integrationen endt med at skabe nok støj, og resulteret i at i at rouletten rammer forkerte lysdioder. Dette formodes at blive fikset med pull-up på clock signalet.

Et andet problem fundet i integrationstesten er, at det kortere interval der bliver sampled på ADC'en til lydsensoren har resulteret klap aktiveringens er mindre nøjagtig. Hvor tidligere test af registreredes 6/10 opsamlede det fulde system kun 4/10.



Figur 44: PSoC Fuld Test Breadboard

12.3 Accepttest

Der er udført en enkelt accepttest i forbindelse med projektet per afleveringsdatoen, d. 31 / 05 / 2024, men pga. resultaterne fra denne ville gruppen forsøge at udføre en yderligere accepttest. Den fulde accepttest kan ses i bilag [9]. Der er opført en revurderet accepttestspezifikation der er vedlagt som bilag [10]. Den revurderet accepttestspezifikation nåede aldrig at blive testet.

Use case under test		UC 1: "Spin roulette"		
Scenarie		Hovedscenarie		
Prækondition		System er funktionelt og minimum 1 spiller tilkoblet		
Step	Handling	Forventet Observation/Resultat	Faktisk Observation/Resultat	Vurdering (OK/FAIL)
1	Tryk på startknap, og vent 5 sekunder.	Et spil startes på PSoC (Lysdioder blinker i en cirkelsekvens).	Et spil startes efter tryk på startknap.	OK
2	Et stopur startes samtidig med tryk på "startknappen". Verificer at systemets lysdioder blinker langsommere efter 10 sekunder (+/- 2 sekunder).	Det observeres at lysdioderne blinker langsommere efter 10 sekunder (+/- 2 sekunder).	LED'erne blinker langsommere efter 10 sekunder.	OK
3	Generation af et tilfældigt tal mellem 1-37 ⁴ .	Observation af at et tilfældigt tal mellem 1-37 genereres på brugergrænsefladen, når spillet startes.	Der genereres et tal mellem 1-37, og dette vises på brugergrænsefladen.	OK
4	Korrekt lysdiode forbliver tændt, mens resten slukkes.	Visuel test: Observation at den lysdiode, som forbliver tændt, er den som repræsenterer det tilfældigt genereret tal. De resterende lysdioder slukkes.	Den forkerte lysdiode forbliver tændt.	FAIL
5	Spillets resultat er aflæst korrekt i systemet.	Visuel test: Tjek af brugergrænsefladen at resultatet af spillet er aflæst korrekt i systemet.	Resultatet blev ikke aflæst korrekt.	FAIL

Tabel 13: Accepttest Specifikation af UC1

12.4 Opsumering af resultater for udført accepttest:

Overordnet set har resultaterne været mindre gode på accepttesten. Mange af stepsne blev til "**FAIL**" fremfor "**OK**". Gruppen ville have ønsket flere succesfulde steps ud fra de gennemførte tests. Gruppen havde udarbejdet en revurderet accepttest, som skulle have været udført, men som ikke nåede at blive gjort inden afleveringsfristen.

UC:	Resultat:	Kommentar:
UC1:	Delvist OK	Langt det meste fungerede, men pga printplade var der småfejl.
UC2:	FAIL	Systemet viste statistiker, men siden stemte ikke overens med implementeringen
UC3:	Delvist OK	Det var muligt at oprette spiller korrekt, men problemer med server gjorde det ikke var muligt at se spillere på tværs af platforme
UC4:	OK	Det muligt både at oprette bet og beløb som ønsket, dog skal formuleringen af testen revideres.
UC5:	FAIL	Var ikke muligt at starte spil fra Display, resterende funktioner virkede
UC6:	FAIL	Defekt SOMO gjorde at det ikke var muligt at teste på dagen
UC7:	FAIL	Problemer med PrintPladen gjorde at det ikke er implementeret i denne version.
IFK		
U:	Delvist OK	Alle MUST krav lykkedes, men problemer med SHOULD
R:	FAIL	Kravene til systemet realialiblty blev ikke overholdt
P	OK	Alle systemets performance krav blev overholdt
S:	FAIL	Supportability kravene var nedprioriteret og ikke implementeret

Tabel 14: Opsumering af Accepttest

13 Diskussion

Det er svært at pakke resultaterne fra Accepttest 1 på et ind. Testen levede ikke op til kravende fremstillet tidligere i rapporten.

En stor del af dette skyldes dog at gruppens vision om det endelige produkt udviklede sig i løbet af design og implementerings fasen. Denne ændring i vision, sammen med en idé om at stå fast ved den oprindelige accepttestspezifikation resulterede i en uoverensstemmelse mellem accepttesten og det endelige produkt, og en lang række af fejlende stammer fra dette.

Dog er der også en værre type fejl, ex. *UC7 Styr Lysstyrke*, der mislykkedes pga. mangler i printpladen. Hvis gruppen havde håndteret tiden bedre, var dette problemet formentlig fundet tidligere.

13.1 Fejlkilder

Ud fra de krav der ikke helt er blevet fuldendt er der opstillet en række mulige fejlkilder til projektet:

1. Mindre en nødvendig brug af SCRUM og projektmetoder
2. Fejlkommunikation / Enkeltpersoner på opgaver
3. Mindre brug af Risikoanalyse i sidste del af projektet

Det første punkt føler gruppen især var relevant de sidste uger på projektet, da målet føltes nær prioriterede gruppemedlemmer implementeringen frem frem for processen. Dette resulterede i at det var sværere at reagere på nye problemer der fremtrådte, grundet der ikke var en veldefineret contingency plan at følge.

Som fornævnt er det ikke en fuldtallig gruppe, samt en skæv fordeling af SW/E studerende. Dette medførte at nogle opgaver blev udført af enkeltpersoner Fremover tænker gruppen evt i større grad at vurdere hvilke elementer der er relevante for projektets funktion, og forløbets læringsmål, og skære unødvendigheder fra, så der er mere tid til kommunikation mellem medlemmerne.

Da en anden årsag til dette kunne også ligge og måden den femte risikoanalyse blev lavet, hvor de fire første var formuleret af gruppen i fællesskab, blev den sidste skrevet af et enkelt medlem og kun kort læst op og godkendt.

14 Konklusion og Refleksion

I dette afsnit bliver der beskrevet, hvordan gruppen har arbejdet ud fra SCRUM-metoden, som gruppens medlemmer ikke havde erfaringer med før projektsstart. Hvilke fordele og ulemper dette havde for gruppens arbejde med projektet. Afsnittet runder ud med en overordnet konklusion for projektet, hvorefter et afsnit med beskrivelser for retrospektiv følger.

14.1 Konklusion

Gruppen kan ud fra projektet som helhed konkludere, at have udviklet en elektronisk ”russisk” roulette-applikation med fokus på en effektiv brugergrænseflade opbygget i React og en pålidelig backend-infrastruktur. Ved at implementere moderne sikkerhedspraksis som CORS (Cross-Origin Resource Sharing) og benytte HTTP-anmodninger har gruppen sikret en robust sikkerhedsarkitektur og effektiv håndtering af dataoverførsel.

Gennem en omhyggelig testproces, der omfattede modul-, integration- og accepttest, har gruppen verificeret funktionalitet og kvalitet af systemet. Selvom visse krav, såsom ’Party Mode’ og nøjagtig lysdiodeafvikling, ikke blev fuldt opfyldt i accepttesten, har testfasen alligevel bidraget til at identificere områder til forbedring og styrket tilliden til systemets pålidelighed.

Gruppens anvendelse af sekvens-, klasse- og andre komponentdiagrammer har skabt en struktureret og klar forståelse af GUI-opbygningen og backend-interaktionerne. Ved at integrere og gøre brug af disse diagrammer i gruppens udviklingsproces har gruppen sikret en sammenhængende og veldefineret arkitektur.

I løbet af projektet stødte gruppen på udfordringer, herunder effektiv implementering af SCRUM-metoden til projektstyring og realistisk tidsplanlægning. Ved at reflektere over disse udfordringer har gruppen identificeret områder til forbedring, såsom bedre risikostyring, bedre og mere kommunikation, og mere nøjagtig ressourceallokering blandt gruppens medlemmer.

Samlet set repræsenterer dette projekt en værdifuld læringsoplevelse, der har styrket gruppemedlemmers forståelse for både tekniske og procesmæssige aspekter af hardware- og softwareudvikling. Projektet har mest været drevet af softwareudvikling grundet flertallet af softwarefolk i gruppen, hvor hardware har spillet en mindre rolle.

Gruppemedlemmerne ser frem til at anvende disse erfaringer i fremtidige projekter og kontinuerligt forbedre deres evner og praksis.

14.1.1 Retrospektiv

Når gruppen ser tilbage på projektet er der en række ting gruppen ønsker at have gjort bedre. Dette ligger især i forhold til metode og projektstyrelse. Tidligere introduktion til SCRUM havde givet gruppen længere tid til at få metoderne under huden. Hvis gruppen var kommet hurtigere i gang med at bruge Agile Boards ville have skabt et større overblik over projektet.

Som nævnt i afsnit 8.5 har der også været en række ting gruppen har lært i forhold til teknologianalysen. Hvor fremover vil gruppemedlemmerne mere se Teknologi Analysen som en kontinuert proces, frem for den ”One

and Done” mentalitet, som det endte med i projektet. Samt i større grad overveje hvilke elementer der foretages analyse på.

Med det sagt er der også en række mere positive ting gruppen har opnået: Især Risiko Analysen har været et brugbart værktøj, og selv hvor gruppen kunne have lagt større vægt på at fjerne største risici endnu hurtigere.

14.1.2 Fremtidigt Arbejde

Med de mere ambivalente resultater fra Accepttest 1, deles det fremtidige arbejde op i, hvad gruppen tænker, der skal til for at nå i mål med projektet, og derefter mulig viderebygning.

I mål med Accepttesten

Et stort skridt mod accepttesten ville være at få fremstillet et opdateret print til GameBoardet, som i følge gruppen, ville kunne løse langt de fleste af de fejlede accepttestspezifikationer, som blev testet i accepttesten.

Udvidelse af Projektet

Projektet havde primært fokus på at få roulettespillet til at virke med alt det, som et normalt roulettespil fra virkeligheden kan. Skulle gruppen udvide projektet, kunne gruppen eventuelt implementere en form for bluetooth-forbindelse, hvorved spillere ville kunne tilslutte til spillet ved hjælp af deres forskellige devices. Her kunne man logge deres devicenummer ud fra de forskellige bluetooth-forbindelser, så hvert enkelt device ville fungere som en spiller.

Litteratur

- [1] *Datasheet TEMT6000.* URL: <https://www.vishay.com/docs/81579/temt6000.pdf>.
- [2] Circuits diy. *Datasheet BC550.* URL:
<https://www.circuits-diy.com/bc550-npn-low-noise-transistor-datasheet/>.
- [3] Daniel Lockyer. *Documentation of SQLite API.* URL:
<https://github.com/TryGhost/node-sqlite3/wiki/API>.
- [4] MAXIM. *Datasheet Max4466.* URL:
<https://www.sgbotic.com/products/datasheets/sensors/MAX4465-MAX4469.pdf>.
- [5] nae.org.uk/new. *MAPPING CLAPPING. FIVE MONTHS OF CLAPS.* URL:
<https://www.nae.org.uk/news/blog/mapping-clapping-five-months-of-claps/#:~:text=The%20sound%20frequency%20of%20an,can%20activate%20the%20process..>
- [6] ON. *MoSCoW.* URL: <https://storiesonboard.com/blog/moscow-prioritization-model>.
- [7] Dennis Paul. *FFT with CMSIS-DSP Library.* URL:
<https://klangstrom.dennispaul.de/2021/08/30/FFT-with-CMSIS-DSP-Library.html>.
- [8] Tim Smith. *Creating a Basic Server with Express.js.* URL:
<https://www.iamtimsmith.com/blog/creating-a-basic-server-with-express-js>.
- [9] ti. *Datasheet HC4017.* URL: https://www.ti.com/lit/ds/symlink/cd4017b.pdf?ts=1717029541794&ref_url=https%253A%252F%252Fwww.google.com%252F.
- [10] James Turner. *Getting Started with Node.js SQLite.* URL:
<https://www.linode.com/docs/guides/getting-started-with-nodejs-sqlite/>.
- [11] Torben Gregersen - Ingeniørhøjskolen Aarhus Universitet. *ASE.*

15 Bilagsoversigt

Projekt:

1. **Bilag 1:** Projektformulering
2. **Bilag 2:** Kravspecifikation
3. **Bilag 3:** Teknologianalyse
4. **Bilag 4:** Systemarkitektur
5. **Bilag 5:** Design
6. **Bilag 6:** RisikoAnalyse
7. **Bilag 7:** Realisering
8. **Bilag 8:** Modul og Integrationstest
9. **Bilag 9:** Accepttest 1
10. **Bilag 10:** Revurderet Accepttest Specification

11. Process:

- 11.1 Procesbeskrivelse
- 11.2 Gantt-Diagram
- 11.3 Samarbejds aftale
- 11.4 Vejledermøder
 - 11.4.1 Mødeindkaldelser
 - 11.4.2 Referater

12. SourceCode:

- 12.1 Accepttest 1:
 - 12.1.1 PSoC Styring
 - 12.1.2 React-app

13. GUI Kilder:

- 13.1 React Router.pdf
- 13.2 Functional components.pdf
- 13.3 React useState Hook.pdf
- 13.4 React Overview_E22.pdf

16 Ordliste

- **UC:** Use Case
- **FDUC:** Fully-dressed Use Case
- **IFK:** Ikke-Funktionelle Krav
- **FURPS** Functionality, Usability, Reliability, Performance and Supportability
- **GUI:** Graphic User Interface
- **PSoC:** Programmable System on Chip. Oftest PSoC 5LP
- **RPi:** Raspberry Pi
- **SD:** Sequence Diagram
- **STM:** State Machine
- **ADC:** Analog-Digital Converter
- **FFT:** Fast-Fourier Transformation
- **ER:** Entity-Relation
- **BDD:** Block Definition Diagram
- **IBD:** Internal Block Diagram
- **IC:** Integrated Circuit
- **SCRUM:**
- **ASE:** Aarhus School of Engineering