

EMBEDDED UDVIKLING MED FYSISKE GRÆNSEFLADER



AARHUS UNIVERSITET

Periode: Efterår 2023

Afleveringsdato: 05-01-2024

Vejleder: Gustav Kastrup Nielsen

E2PRJ2-03 SEMESTERPROJEKT 2

PROJEKTGRUPPE 05

Mads Dittmann Villadsen - 202209869

Altaf Rezai - 202209871

August Gravsen - 202210047

Hafsa M M Yousuf - 202104677

Louise Andersen - 202104844

Ruben Gullborg - 202208849



Executive summary (Mads)

Today, characterized by the rapid integration of technology into our daily lives, the demand for innovative solutions to address security concerns has never been more pronounced. That's why, as part of our semester project in software engineering, we embarked on a journey to conceptualize, design, and implement a cutting-edge home security service in hopes of addressing those security concerns.

Throughout this project, we delved into the intricacies of software design by using different programming languages like Python and C++, and by designing our product based off different system architecture models like block definition and internal definition diagrams. We've also thought deeply about the user experience to create a solution that not only meets the highest standards of security but also seamlessly integrates into the daily lives of the users of our product.

At the core of our home security service is a seamlessly integrated system centered around a Raspberry Pi, transforming it into a dynamic hub of security and monitoring capabilities. Leveraging the versatility of this compact yet powerful device, we have meticulously designed a solution that encompasses video surveillance, motion detection, and environmental monitoring, all which can be accessed from anywhere in the world because of it working as a webserver provided by Ngrok free hosting service (Ngrok, 2024).

Our home security system is made up of 3 different key features:

- Live Camera Feed
 - The Raspberry Pi serves as a webserver, offering users a live camera feed accessible through a web interface. This camera feed comes from a camera component connected directly to the Raspberry Pi. The feed gets recorded for 10 seconds whenever motion is detected by the motion sensor, and these files are saved and are accessible through the web interface as well.
- Motion Detection
 - An integrated motion sensor actively monitors the surroundings. Upon detecting motion, a prominent "Motion detected! - Video recording" text overlay is displayed on the website below the live camera feed, alerting users to potential activity. The motion sensor plays a role in the functionality of the camera.
- Temperature Monitoring
 - A connected temperature sensor continuously measures the ambient temperature in the environment. The web interface displays the current temperature below the live camera feed as well. An intelligent alert system triggers a visual change in the display when the temperature surpasses a predefined threshold of 28 degrees Celsius. Users are immediately notified of elevated temperatures with a clear "HIGH TEMP ALERT" message besides the current temperature.

This summarizes our second semester home security project, which is not just about Raspberry Pis and cameras, but about making homes as secure as they can be. In Denmark, homes face a yearly risk of about 7.870 break-ins (Chabert, 2023). We as a group are not saying we've got a magic fix, but we're hopeful that our product can become a game-changer. Maybe not turn that big number into zero overnight but at least making a dent in it.

Imagine a future where your home is your fortress, no matter where you are...

Indhold

Executive summary (Mads)	
Projektoplæg	1
Inputs til jeres projekt:	4
Hardware:.....	4
Software:	4
System:.....	4
Alternativt SW-PRJ2 Embedded projekttema	6
Gruppens projektkoncept	6
Krav	7
Problem- & projektformulering (Mads).....	8
Kravspecifikation.....	11
Systembeskrivelse (Mads)	11
Funktionelle krav (Mads).....	13
Beskrivelse af systemets aktører (Mads)	13
UC-diagram (Mads)	14
Fully dressed UC-beskrivelser (Altaf og Mads)	15
MoSCoW (Louise, Hafsa Mads, August, Altaf og Ruben).....	19
Ikke-funktionelle krav (Louise, Hafsa Mads, August, Altaf og Ruben)	22
Accepttestspecifikation for funktionelle krav	25
Use Case 1: Manuelt tjek af systemets komponenter (Mads og August).....	25
Use case 2: Forhøjet temperatur opfanges (Louise og Hafsa)	26
Use case 3: Bevægelse opfanges (Altaf og Ruben)	28
Test af ikke funktionelle krav	30
Temperatursensor (Hafsa og Louise).....	30
Kamera (Mads og August)	31
Bevægelsessensor (Altaf og Ruben)	32
Systemarkitektur (Mads og August)	34
Grænseflader (Mads og August)	35
HW-arkitektur	36
Raspberry Pi (Mads og August).....	36
Block Definition Diagram (BDD) (Mads og August).....	36
Internal Block Diagram (IBD) (Mads)	37
Arduino (Ruben og Altaf)	38
Block Definition Diagram (Ruben og Altaf)	38

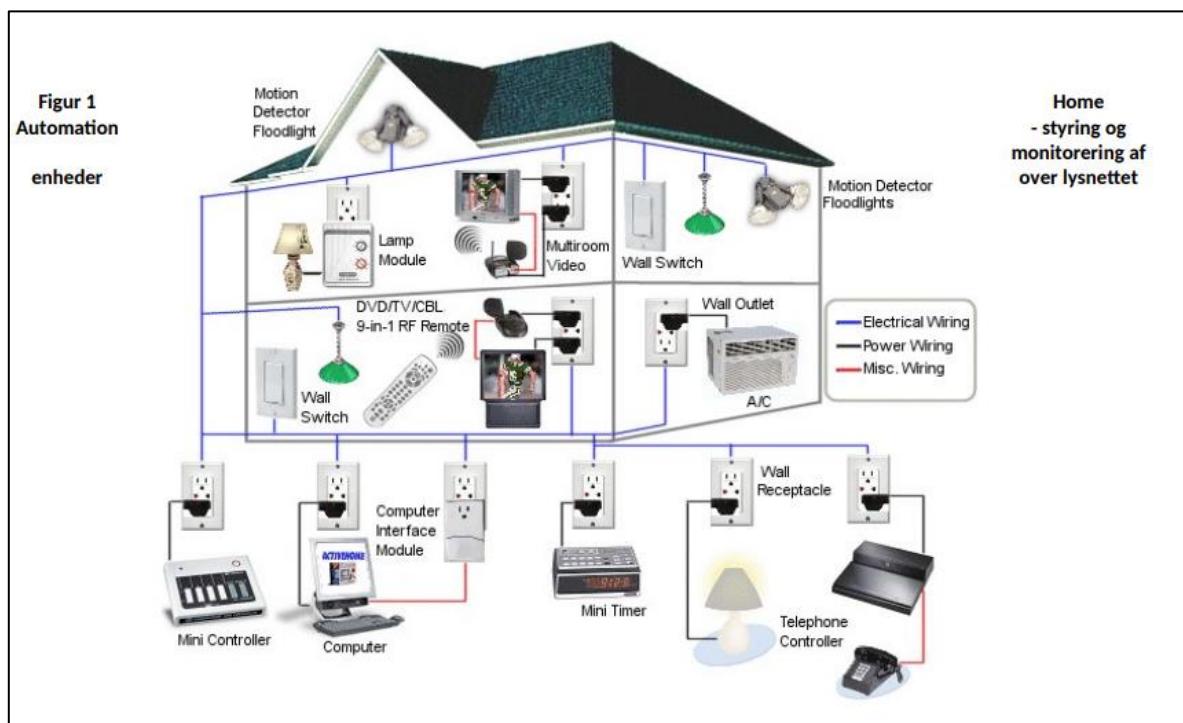
Internal Block Diagram (IBD) (Altaf og Ruben)	39
Sekvensdiagrammer.....	40
Use case 1: Manuel ttek af systemets komponenter (Altaf).....	40
Use case 2: Temperatursensoren (Hafsa)	41
Use case 3: BevægelsesSensoren (Altaf og Ruben)	42
Signalbeskrivelser (Ruben, Altaf, Hafsa)	43
SW-arkitektur.....	45
En domænemodel for hele systemet (Mads og Louise)	45
Modul- og klassebeskrivelse	46
KAMERA (August):.....	46
BEVÆGELSESSENSOR: (Altaf, Ruben)	48
TEMPERATURSENSOR (Hafsa):.....	49
Klassediagram for hele projektets software (Ruben)	51
SW Design	52
Implementering	52
Bevægelsessensor (Altaf):.....	52
Temperatursensor (Hafsa og Louise):.....	56
Kamera (August):	58
GUI (August):.....	62
HW Design.....	68
Temperatursensor (Hafsa og Louise)	68
Kamera (August)	70
Bevægelsessensor: (August)	72
Udførelse af accepttest.....	74
Accepttestudførelse for de funktionelle krav	74
Use Case 1: Manuelt ttek af systemets komponenter (Mads og August).....	74
Use case 2: Forhøjet temperatur opfanges (Hafsa)	76
Use case 3: Bevægelse opfanges (Altaf og Ruben)	77
Accepttestudførelse for de ikke-funktionelle krav	80
Temperatursensor (Hafsa).....	80
Kamerasensor (Mads og August).....	80
Bevægelsessensor (Altaf og Ruben)	82
Modultest (Mads).....	84
Integrationstest (Ruben).....	86
Fejlkilder (August).....	90
Tidsplan (Mads og Louise).....	92

Konklusion (Alle gruppemedlemmer)	93
Konklusion for hardwaredelen	93
Konklusion for softwaredelen	93
Overordnede konklusion for projektet	94
Individuelle konklusioner	95
Mads' individuelle konklusion	95
Hafsa's individuelle konklusion	96
Altafs individuelle konklusion	97
Louises individuelle konklusion	98
Rubens individuelle konklusion	99
Augsts individuelle konklusion	100
Bilag	101
1. E2PRJ2_Projektoplæg.pdf	101
2. Vejledning_for_gennemfoerelse_af_projekt_2_V1_00.pdf	101
3. Arbejdet_i_projektgruppen_2._semester.pdf	101
4. tidsplan_gruppe_5_fardig.pdf	101
5. Samarbejdsaftale_for_Semesterprojekt_2_Gruppe_5.pdf	101
6. 1._vejledermode__140923.pdf	101
7. 2._vejledermode__200923.pdf	101
8. 3._vejledermode__270923.pdf	101
9. 4._vejledermode__041023.pdf	101
10. 5._vejledermode__251023.pdf	101
11. 6._vejledermode__011123.pdf	101
12. 7._vejledermode__081123.pdf	101
13. 8._vejledermode__151123.pdf	101
14. 9._vejledermode__221123.pdf	101
15. PIRMotionSensorDONE.zip	101
16. Video_server_temp_update.py	101
17. Temperaturesensor.py	101
Bibliografi	101

Projektoplæg

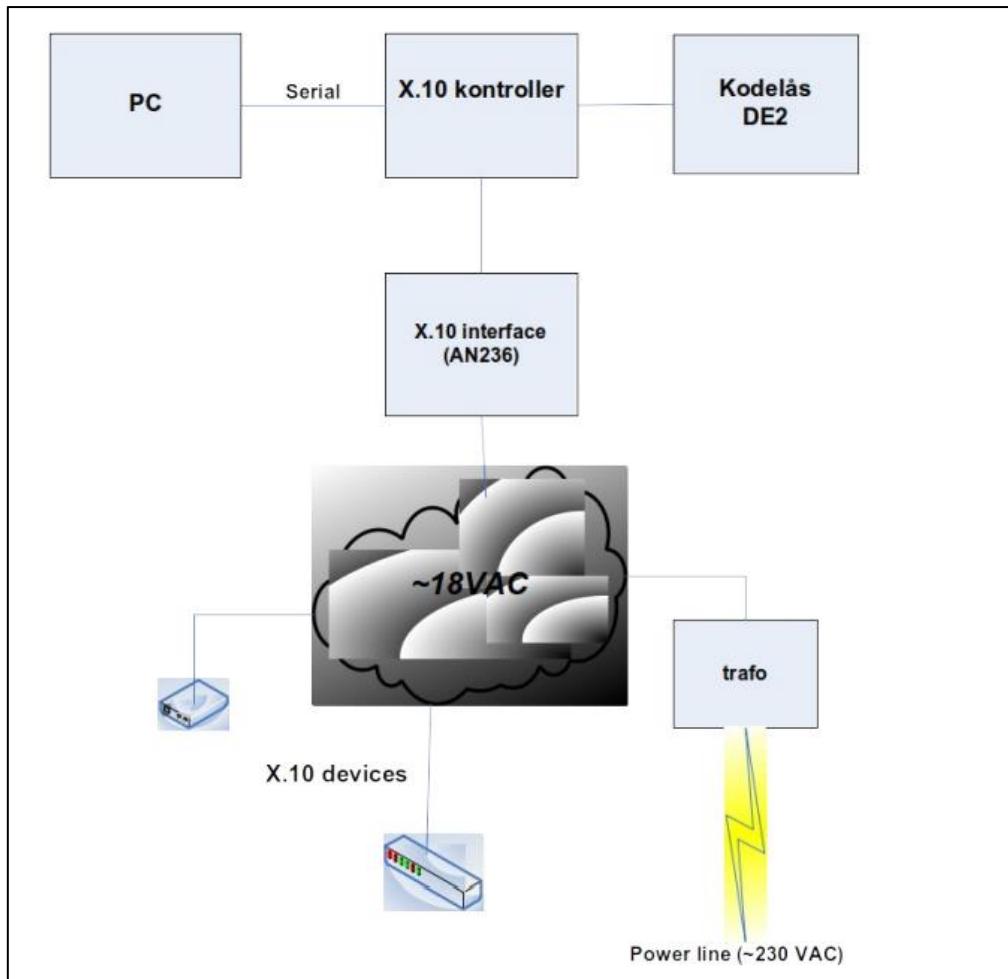
Har du oplevet situationen, hvor du kommer hjem fra ferie eller jul, der har været indbrud i lejligheden og det hele er rodet igennem? Din computer er væk og din elskede samling af flaskekapsler, som du har samlet på siden du var lille. Samlingen er uerstattelig for dig.

Oplægget til dette projekt er at opfinde et system til tyveriforebyggelse. Ikke et traditionelt system kun med bevægelsessensorer og alarmer, men et system der kan simulere, at der er aktivitet og nogen hjemme. Udgangspunktet er at benytte principperne fra "Home Automation" med styring af enheder over lysnettet (X.10), som illustreret i nedenstående Figur 1.



Figur 1: Automationenheder

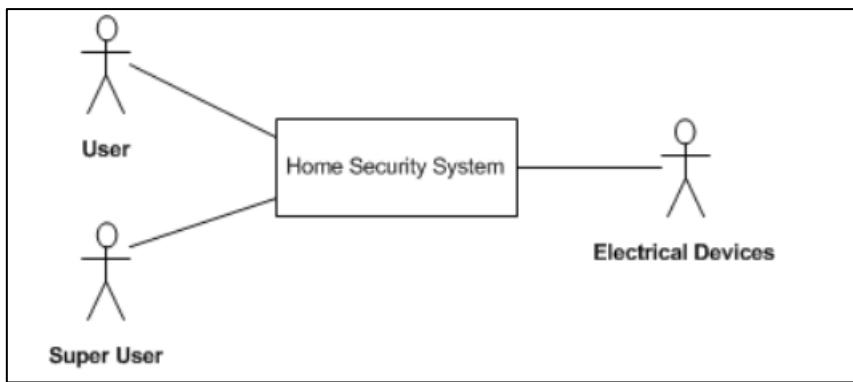
Jeres løsning skal som minimum indeholde en computer (PC), der via en seriel forbindelse kommunikere med en kontroller (arduino). Kontrolleren får signal fra en kodelås (DE2 board), men benytter i øvrigt power line kommunikation til styring af én eller flere X.10 enheder. Illustreret i nedenstående figur:



Figur 2: Home Security System – Hardware setup

Formålet med ”Home Security” systemet kunne være tyveriforebyggelse, og skal kunne programmeres til simulering af, at der er ”nogen hjemme”. Det kunne være automatisk tænd/sluk af lys, fjernsyn o.l. aktiviteter i hjemmet. Systemet skal kunne styres og konfigureres i forskellige scenarier som gemmes og vælges fra en PC. Når PC’en er slukket, skal systemet stadig kunne simulere tyveriforebyggelsen, så længe den centrale X.10 kontroller er tændt. Kodelåsen DE2 sikrer at kun personer med korrekt kode kan skifte scenario og konfigurer sytemet.

I definerer selv scenarier og funktionalitet i samarbejde med vejlederen. Det er vigtigt at disse scenarier er specificeret i den kravspecifikation, der godkendes efter første review. Den udvidede funktionalitet prioriteres og vurderes i samarbejde med projektvejlederen for den efterfølgende design og implementeringsfase.



Figur 3: Home Security Context Diagram

Konceptet hedder ”power line communication”, og findes i mange varianter beregnet til data-transmission over et eksisterende 230Volt vekselspændingsnet. Se http://en.wikipedia.org/wiki/Home_automation. Vi benytter kommunikationsteknikken X.10, der både definerer hvordan digitale signaler overlejres nettet og den protokol der benyttes. Af sikkerheds grunde udvikler vi systemet på et lukket 18 volts AC-net, der fås fra en transformator i kan hente på værkstedet.

Home Security Systemet består af PC'en, der via en USB seriellkonverter sender kommandoer til eksterne enheder tilkoblet det lukkede 18 volts AC-net. Mellemleddet er X.10 kontrolleren, der består af et udviklingsboard (Arduino Mega2560) med en mikrocontroller efterfulgt af et elektrisk kredsløb (X.10 interface), der kobler direkte på 18 volts AC-nettet. Modtageren afhænger af den ønskede funktionalitet, men indeholder altid en X.10 modtager, der afkoder det digitale signal sendt over nettet.

Applikationsnoten AN2361¹ beskriver et eksempel med brug af en PIC-processer og interface kredsløb til 120 V AC 60 Hz nettet (USA). Denne note kan bruges som inspiration, men I designer jeres egne kredsløb, der passer til den prototype I udvikler og som I forstår. Undlad at udvikle strømforsyninger, brug laboratoriets. Der kan kun trækkes 500 mA fra den 18 VAC-forsyning som udleveres fra værkstedet brug den kun til X10 kommunikation.

Systemet skal indeholde et DE2 board, f.eks. implementation af en kodelås programmeret i VHDL, der via udviklingsboardet kan låse op for systemet. Kodelåsen udvikles i faget ”Digital Systemdesign” signalere med et digitalt signal om ”Home Security” systemet skal aktiveres eller deaktiveres. DE2 board kan også bruges til andre funktionaliteter i jeres løsning (fortrækkes).

¹ Application note: AN236_ApplicationNote.pdf findes på Brightspace

Der er også mulighed for at dele egenudviklede X.10 enheder med andre gruppens enheder til demonstration af det samlede system. Hermed menes at I fra jeres kontroller kan styre en X.10 enhed udviklet af en anden gruppe. F.eks. en triac-styret lysdæmper eller digital kontakt.

Inputs til jeres projekt:

Start tidligt i jeres projekt med tekniske forundersøgelser og forsøg. Søg information om X.10 og de emner som I ikke kender til. Så snart kerne-teknikkerne er identificeret kan principperne afprøves. **Hvis I går istå, søg hjælp hos vejleder eller underviser i det fagområde problemet er relateret til!**

Hardware:

- Udviklingsboards: Arduino Mega2560 og DE2
- 18 VAC/500 mA strømforsyning
- Egen udviklet X.10 sender og modtager enheder

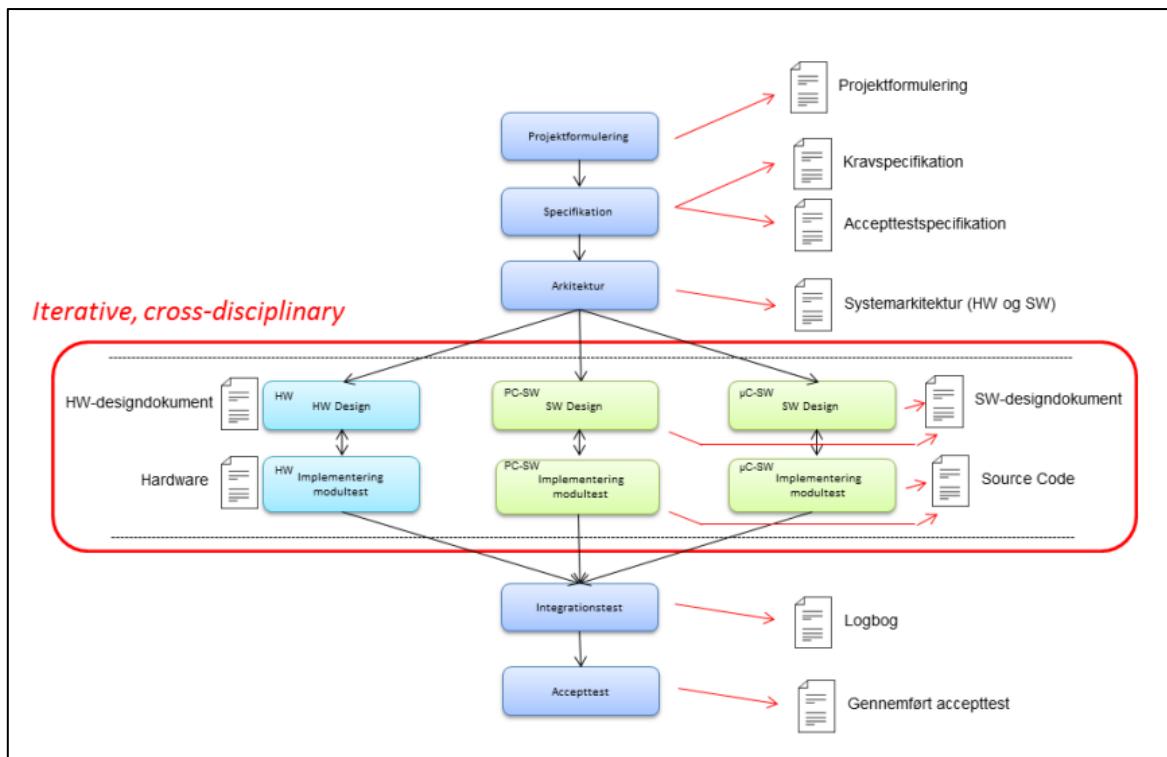
Software:

- Microsoft Visual C++ compiler til PC
- C/C++ compiler og udviklingskit som benyttes for Arduino Mega2560
- Det frarådes at gå i gang med GUI udvikling i .NET og C++. Det er helt fint med en simpel tekst baseret terminal. Alternativt kan vi anbefale at kigge på Qt², som også bruger C++ til GUI udvikling.

System:

- UML/SysML diagram tegningsværktøj (Visio med stencil) <http://softwarestencils.com/sysml/>
- Alternativt benyttes Enterprise Architect (EA), universitet har license

² <https://www.qt.io/>



Figur 4: Semesterprojekt modellen illustreret med de faser som projektet gennemløber

Kravspecifikationen og accepttesten må gerne indeholde flere krav og scenarier (Use Cases), end der er medtage i jeres endelige prototype. Det er dog vigtigt, at det klart fremgår af dokumentationen, hvilke scenarier og dele af systemet som er med i systemarkitekturen, designet, implementeret og testet i jeres prototype.

For hver fase i projektet, skal gruppen holde review med en anden gruppe. Review gruppens projektvejleder skal deltage, indkaldes:

(For dato-deadlines: se Brightspace)

- **Projektformulering godkendt af vejleder.**

- **1. Review**

Omfatter dokumenterne: Projektformulering, tidsplan, kravspecifikation og accepttestspezifikation

- **2. Review**

Omfatter dokumenterne: Systemarkitektur-dokument og evt. første version af HW/SWdesigndokument, hvis det foreligger. Udgangspunktet er den nu reviderede Kravspecifikation, der også sendes til reviewgruppen.

- **Demonstration og accepttest**

Projektet funktionalitet fremvises for vejleder, iflg. accepttestspezifikationen.

- Projekt aflevering

Projektet afsluttes med aflevering af projektrapport og -dokumentation.

- ### • Eksamens

Mundtlig gruppeeksamen på baggrund af projektrapport og -dokumentation

Alternativt SW-PRJ2 Embedded projekttema

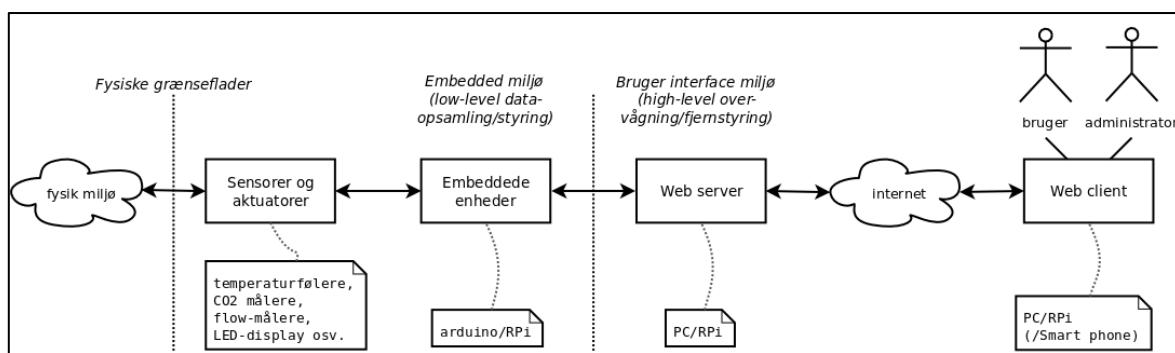
Der er herudover mulighed for, at 'rene' SW-PRJ2 grupper kan erstatte X10-kommunikationen med et alternativt embedded-software tema.

Grupper, der har E-studerende, kan også vælge dette alternative spor, men de har som udgangspunkt et bedre grundlag for at lave det traditionelle X10-projekt.

Gruppens projektkoncept

Dvs. at gruppen kan udtænke et projektkoncept, og herfra lave en alternativ problemformulering til godkendelse hos vejleder.

Jeres projektkoncept kan godkendes, så længe projektet dækker temaet 'embedded udvikling med fysiske grænseflader' og er baseret på diverse hardware sensorer tilkoblet til Arduino (ala temperaturfølere, CO₂ målere, flow-målere, LED-displays, etc.), se Figur 5.



Figur 5: koncept diagram over det alternative projekt, der inkluderer grænseflader til et fysisk miljø samt, et embedded miljø til dataopsamling og styring samt et brugerinterface miljø til overvågning og fjernstyring, her opsat som en client-server model. Int

Projektkonceptet skal således indeholde et embedded system, der mÅler og styrer noget i et fysisk miljø via et sæt af embedde enheder. Disse enheder opsamler løbende data, der så kan tilgås og præsenteres i et brugerinterface miljø, som f.eks. kan tilgås over internettet via en web-browser.

Fokus på projektkonceptet kunne f.eks. være energi-optimering, mens det tekniske fokus på softwaren bør ligge på modul-opdeling og interface-beskrivelser.

Krav

Krav til et embedded projekttema er som for "X-10" projektoplæget, på nær de tre punkter

- Kommunikation over X10/elnetværk via et eget udviklet system,
- Brug af minimum en Arduino X10 sender og en Arduino X10 modtager,
- Brug af et (eller flere) DE2-boards.

som bortfalder. Dvs. at alt X10 og DE2-board HW og SW udgår i det alternative projekt, og disse krav erstattes så med (**MUST**):

- En eller flere arduinoer til et low-level dataopsamling/styring (**MUST**),
- Tre eller flere sensorer og/eller aktuatorer opkoblet til en eller flere arduinoer (**MUST**),
- En eller flere PCere/RPi (Raspberry Pi) til et high-level bruger interface miljø (f.eks. via en client/server model) til monitorering og/eller fjernstyring (**MUST**),
- En UI (grafisk/GUI eller tekst/konsolbaseret) i brugerinterface miljøet (**MUST**),
- Brug af standard kommunikationsprotokoller mellem (**MUST**):
 - Enhederne i det embedded miljø (f.eks. UART, SPI, eller GPIO),
 - Enhederne i brugerinterface miljøet (f.eks. WiFi, Bluetooth eller Ethernet),
 - Det embeddede miljø og bruger interfacet (f.eks. UART eller Ethernet).

Herudover følgende krav (**SHOULD**, **COULD** og **WONT**)

- Gerne brug af Raspberry PI enheder som alternativ til PC og arduinos (**SHOULD**, **COULD**),
 - Dog skal der stadig være mindst en arduino i systemet (**MUST**).
- Kommunikation fra web server til client kan også være over internet (**COULD**),
- Ingen eller meget lidt fokus på mobile-apps (**WONT**).

Problem- & projektformulering (Mads)

Vi lever i dag i en verden fyldt med elektroniske komponenter, som er med til at bidrage til vores hverdag. Dette er fra små simple ting som bevægelsessensorer, der tænder lyset foran hoveddøren, når nogle går forbi, til store samarbejdende systemer af adskillige elektroniske komponenter, som hver især har sin brik i det store puslespil. Mange ting som allerede er opfundet, bliver langsomt med tiden fornyet og implementeret med den nyere teknologi, og før man ved af det, så er hele hjemmet blevet en stor samling af elektroniske komponenter. Dette er hvor, at begrebet 'smart homes' opstår.

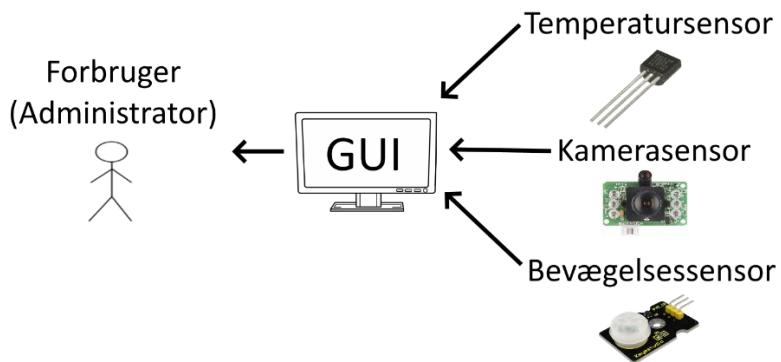
Smart homes, også kendt som intelligente hjem, er blevet mere udbredt i verden i dag, og der findes mange forskellige måder, hvorpå hverdagen kan gøres nemmere for forbrugerne. I Danmark meddelte hver femte familie i år 2021, at de havde elektronisk styring af hjemmets varme og el, såsom en elpære med indbygget Bluetooth eller en vejrstation, der styrer temperaturen i rummet. Det samme var gældende for styring af hjemmets sikkerhed eller overvågningsprodukter f.eks. via smartphone (Bosanac, 2021). Disse former for elektronik kan være med til at hjælpe med at øge sikkerhedsniveauet i forbrugernes hjem. Der findes mange forskellige sikkerhedsrelaterende smart home produkter som bl.a. smarte alarmer, røgalarmer, temperaturmålere eller overvågningskameraer. I 2020 undersøgte Danmarks Statistik, om disse produkter var med til at skabe tryghed i hjemmet, og kom frem til, at de hos 15 procent ud af 1.5 millioner danske borgere resulterede i en øget tryghed i hjemmet (Tassy, 2020).

I Danmark blev der i andet kvartal 2023 anmeldt 7.870 indbrud i alt, og ud af disse var 3.607 indbrud i beboelser. Selvom dette antal er væsentlig mindre end de 6.109 indbrud i beboelser målt i andet kvartal 2019, så kan statistikkerne ikke sammenlignes på lige fod, da der under coronaviruspandemien, som medførte til lockdown-perioden, blev begået langt mindre indbrud grundet, at næsten alle danskere arbejdede hjemmefra og befandt sig i deres hjem næsten 24/7 (Chabert, 2023). Man kan ikke med sikkerhed vide, om det var det, som var skyld i, at antallet af indbrud begået i beboelser i Danmark faldt drastisk hen over de næste par år. Dog har man gjort sig den erfaring, at indbrudstyve helst undgår at bryde ind imens der er nogle hjemme, da risikoen for at blive opdaget er væsentlig højere.

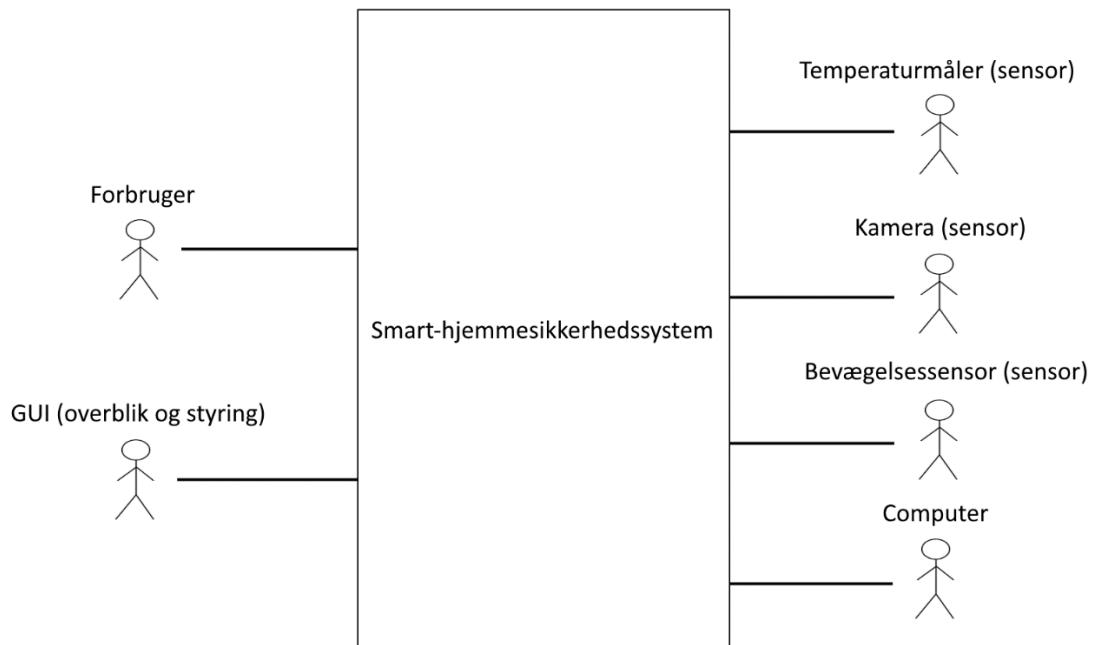
Dette projekt går ud på at udarbejde et smart-hjemmesikkerhedssystem, som kan opdage indbrud ved hjælp af sensorer, overvåge aktivitet ved hjælp af kameraer og endda give dig besked om potentielle nødsituationer såsom hvis der skulle opstå brand, som ville kunne opfanges ved hjælp af benyttelse af en temperatursensor. Dette smart-hjemmesikkerhedssystem skal være med til at mindske antallet af indbrud i Danmark. Både i private ejendomme, men også virksomhedsejendomme.

Projektgruppen overordnede fokus vil være på:

Hvordan kan man ved hjælp af en temperatursensor, en kamerasensor og en bevægelsessensor udarbejde et smart-hjemmesikkerhedssystem med et højt sikkerhedsniveau for forbrugernes hjem, hvor de forskellige mængder data som bliver målt, bliver konverteret til læsbare data, og derefter sendes fra de forskellige komponenter til en udviklet GUI, som skaber et overblik, når dette tilgås af forbrugerne via en computer?



Figur 6: Projektgruppens idé illustreret (eksempel 1)



Figur 7: Projektgruppens idé illustreret (eksempel 2)

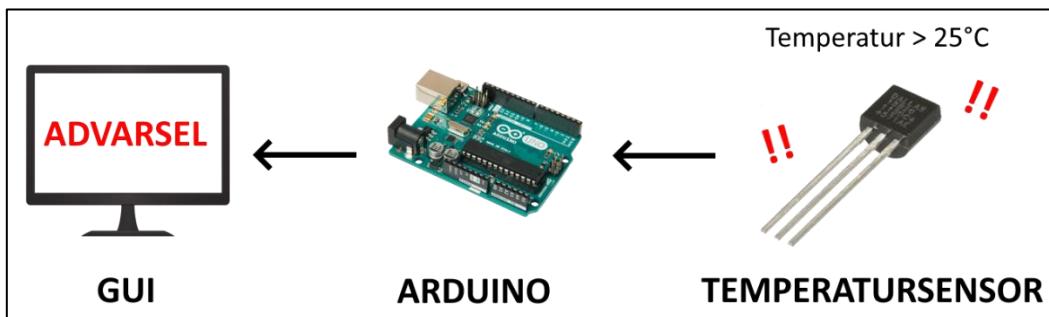
I dette projekt har gruppen valgt at prioritere en temperaturmåler, et kamera og en bevægelsessensor, som alle sammen skal kunne tilgås via en udviklet GUI på en computer, højest. GUI skal kunne give et overblik over alle de forskellige sensorer. En senere opgradering af systemet kan være at implementere flere elektroniske komponenter til smart-hjemmesikkerhedssystemet. F.eks. indbygget alarmer, som kan afspille høje toner for at skræmme indbrudstystene, eller mikrofoner til opfangning af lyd, som kan tilgås og afspilles via den udviklet GUI på en computer.

Kravspecifikation

Systembeskrivelse (Mads)

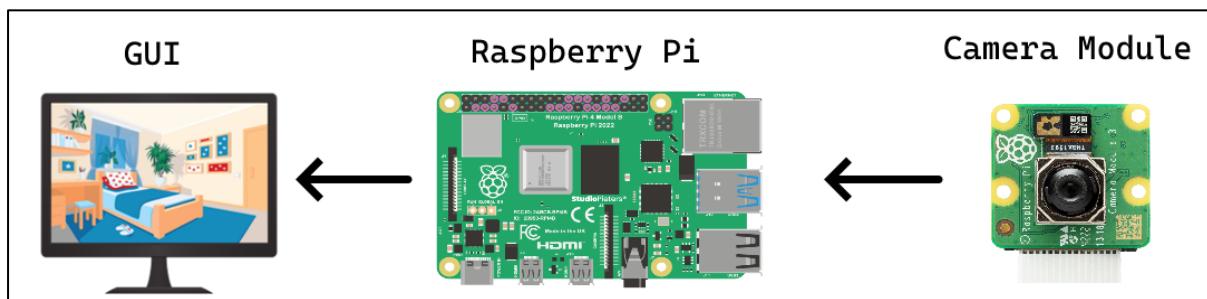
Projektet tager udgangspunkt i følgende systembeskrivelse af et smart-hjemmesikkerhedssystem. Et smart-hjemmesikkerhedssystem bestående af en temperatursensor, en kamerasensor og en bevægelsessensor kan måle forskellige mængder data. Disse data kan opsamles og konverteres til læsbare værdier ved hjælp af en Arduino, som derefter sendes til en brugergrænseflade (GUI), som brugeren kan tilgå ved hjælp af en computer.

Temperaturssensoren i smart-hjemmesikkerhedssystemet skal konstant måle temperaturen. Når temperaturen når over en bestemt værdi, f.eks. 25 grader, skal den sende data til GUI, som brugeren dernæst kan tilgå og aflæse. Her vil brugeren kunne se, at temperaturen er forhøjet, hvilket kan være et tegn på, at der er opstået brand i huset.



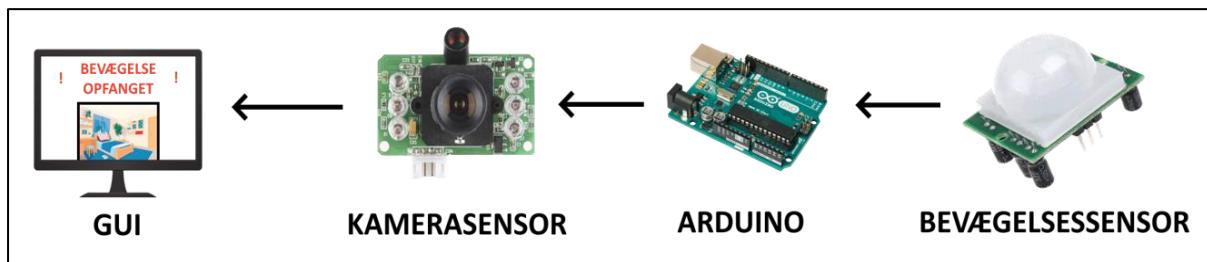
Figur 8: Illustration af eksempel på temperaturssensorens funktionalitet

For at bekære om der er brand i huset eller ej kan brugeren tilgå sig kamerasensoren ved hjælp af GUI. Brugeren skal kunne se realtidsvideoovervågning af huset igennem GUI. Når brugeren har tjekket realtidsvideoovervågningen, så skal brugeren også kunne gemme videoovervågningsoptagelser ned på sit lokale drev, som er blevet optaget efter at bevægelse har været opfanget af bevægelsessensoren.



Figur 9: Illustration af eksempel på kamerasensorens funktionalitet for sig selv

Bevægelsessensoren skal altid være aktiv. Når bevægelsessensoren opfanger bevægelse, skal den sende data til GUI, som f.eks. kan være en streng tekst, som beskriver, at der har været opfangelse af bevægelse. Da bevægelsessensoren er koblet op til kameraet, bliver kameraet også aktiveret til at optage dens feed i 10 sekunder. Når timeren render ud, gemmes optagelsen af feedet, som kameraet opfanger, på GUI, og besked'en om at der er opfanget bevægelse fjernes. Denne videooptagelse kan så tilgås sammen med alle de andre foretaget tidligere videooptagelser på GUI, så brugeren af systemet kan se, om der har været tale om f.eks. et indbrud.



Figur 10: Illustration af eksempel på bevægelsessensorens funktionalitet sammenkoblet med kamerasensoren

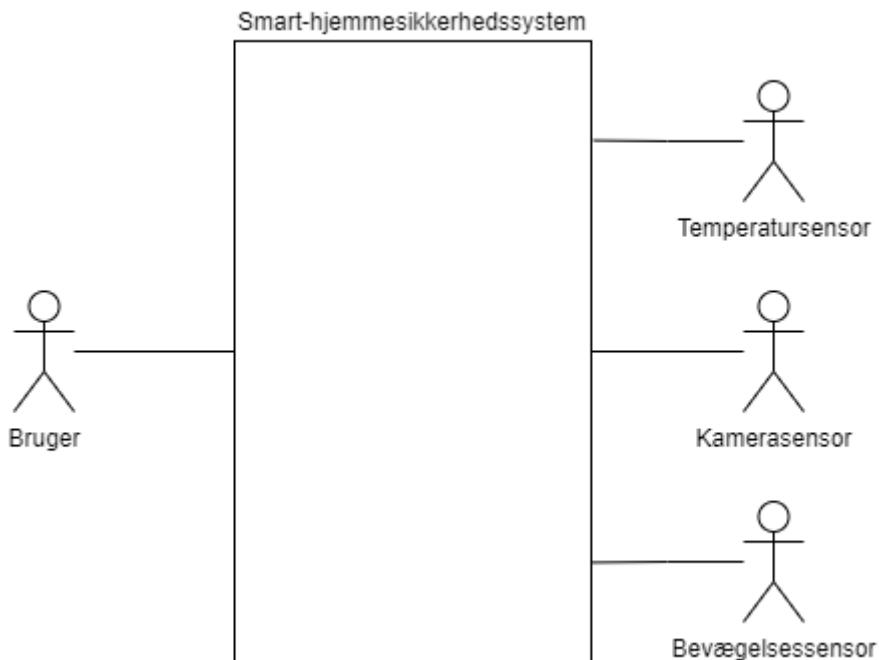
Nedenstående gælder om smart-hjemmesikkerhedssystemet:

- Smart-hjemmesikkerhedssystemet består af 1 temperaturssensor, 1 bevægelsessensor, 1 relæ, 1 kamerakomponent, 1 Arduino og 1 computer (**Raspberry Pi 3 Model B+**).
- Temperatursensoren kan måle temperaturen og sende data til brugeren, hvis den opfanger forhøjet temperaturer.
- Bevægelsessensoren kan opfange bevægelse og sende data til brugeren samt aktivere kamasensoren, hvis den opfanger bevægelse.
- Kamasensoren kan opfange realtid video af det, som den ser, og blive aktiveret af bevægelsessensoren, hvis bevægelsessensoren opfanger bevægelse.
- Dataene fra sensorerne bliver styret via low-level dataopsamling/styring fra 1 Arduino
- Alle 3 sensorers data kan blive tilgået af brugeren via en brugergrænseflade (GUI) på en computer.
- Hvis en komponent i systemet bliver defekt, bør det nemt kunne identificeres og udskiftes af brugeren.

Funktionelle krav (Mads)

Beskrivelse af systemets aktører (Mads)

Til at beskrive systemet, er der blevet opstillet følgende aktør-kontekst diagram. På Figur 11 vises den ene aktør der interagerer med smart-hjemmesikkerhedssystemet, brugeren. På dette niveau vises smart-hjemmesikkerhedssystemet som en boks, som brugeren kan interagere med.



Figur 11: Aktør-kontekst diagram for "Smart-hjemmesikkerhedssystem"

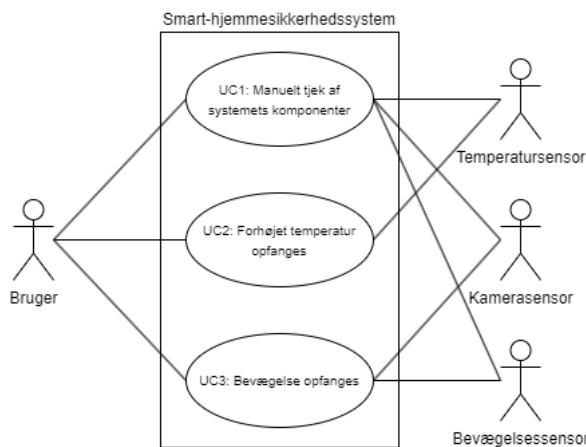
Herunder følger en beskrivelse af brugeren.

Navn på aktør	Bruger
Alternativ reference	Ejer af smart
Aktørtype	Primær
Beskrivelse	Brugeren er ejeren af smart-hjemmesikkerheds-systemet

Tabel 1: Aktørbeskrivelse for "Bruger"

UC-diagram (Mads)

På Figur 12 ses UC-diagrammet for smart-hjemmesikkerhedssystemet. Der er i alt 3 use cases, som hver især beskriver en bestemt funktionalitet smart-sikkerhedssystemet tilbyder brugeren. Her vises smart-sikkerhedssystemet som en boks, da det her er muligt at se funktionen inde i blokken.



Figur 12: UC-diagram for "Smart-hjemmesikkerhedssystem"

UC-beskrivelse (Mads)

UC1: Manuelt tjek af systemets komponenter

Et manuelt tjek af systemets komponenter går ud på, at brugeren manuelt tilgår GUI på en computer, og derefter gennemtjekker, at alle komponenterne i smart-hjemmesikkerhedssystemet fungerer korrekt og kontinuerligt giver et live-feed på hver af deres måle parametre.

UC2: Forhøjet temperatur måles

Hvis temperatursensoren mäter en forhøjet temperatur i det område, hvor den er placeret, skal den kunne alarmere brugeren gennem GUI. Dette skal gøres ved at temperaturen aflæses og fremvises for brugeren på GUI sammen med noget givent tekst, som beskriver, at der er en forhøjelse af temperaturen ift. den temperatur, der burde være i området, hvor den er placeret.

UC3: Bevægelse opfanges

Hvis bevægelsessensoren opfanger en form for bevægelse, skal brugeren have besked. Bevægelsessensoren alarmerer brugeren gennem GUI ved hjælp af en tekst, som beskriver at der er opfanget bevægelse. Herefter optages og lagres video fra kamerasensoren i et bestemt stykke tid, samtidig med at brugeren bliver vist live videoovervågning, af det som kameraet vender mod.

Fully dressed UC-beskrivelser (Altaf og Mads)

I dette afsnit bliver der gennemgået *fully dressed UC-beskrivelser* for alle use cases på Figur 12.

USE CASE 1 MANUEL TJEK AF SYSTE- METS KOMPONENTER	
MÅL	Alle komponenter kan manuelt tjekket, virker gennem brugergrænsefladen
INITIERING	Brugeren åbner for brugergrænsefladen gennem computer
AKTØRER	Bruger, Temperatursensor, Kamerasensor og Bevægelses-sensor
ANTAL SAMTIDIGE FOREKOMSTER	1
PRÆKONDITION:	Alle komponenter i smart-hjemmesikkerhedssystemet er tilsluttet strøm, og har forbindelse til brugergrænsefladen.
POSTKONDITION	Alle komponenter i smart-hjemmesikkerhedssystemet fungerer, og har forbindelse til brugergrænsefladen.
HOVEDSCENARIE	<ol style="list-style-type: none">1. Brugeren åbner for brugergrænsefladen (GUI) ved hjælp af computer.2. Kamerasensoren sender konstant en live-feed til GUI.3. Bruger tester herefter, at bevægelse opfanges af bevægelsessensoren, og at en streng bliver ændret fra "No motion!" til "Motion detected!" på brugergrænsefladen.4. Når bevægelse opfanges, starter kamerasensoren en videooptagelse på 10 sekunder, som kan tilgås af brugeren umiddelbart efter de 10 sekunder, er gået.5. Brugeren kan nu vælge den sidst optagede video-fil fra selve GUI, som viser brugeren de først 10 sekunder efter opfangelse af bevægelsen.6. Temperatursensoren afmåler - samtidigt med al ovenstående - konstant temperaturen, og returnerer temperaturmålingen til brugeren gennem en streng.

	<p>7. [EXTENSION 1]: Hvis temperaturen når over 28 grader.</p>
UDVIDELSER/UNDTAGELSER	<p>[EXTENSION 1]:</p> <ol style="list-style-type: none"> 1. Temperaturen når over 28 grader. 2. Temperatursensoren afmåler stadig konstant temperaturen, og returnerer - samtidig med temperaturmålingen - en alarmerende besked til brugeren.

Tabel 2: UCI - Manuelt tjek af systemets komponenter

USE CASE 2 FORHØJET TEMPERATUR OPFANGES	
MÅL	Temperatursensor har oplyst temperatur for brugeren gennem brugergrænsefladen.
INITIERING	Brugeren åbner for brugergrænsefladen gennem computer.
AKTØRER	Bruger og Temperatursensor.
ANTAL SAMTIDIGE FOREKOMSTER	1.
PRÆKONDITION	Temperaturssensoren er tilsluttet strøm, og er aktiv. har forbindelse til brugergrænsefladen.
POSTKONDITION	Temperaturssensoren er tilsluttet strøm, og har stadig forbindelse til brugergrænsefladen.
HOVEDSCENARIE	<ol style="list-style-type: none"> 1. Temperatursensoren begynder i sin standardtilstand, hvor den måler stuetemperaturen. 2. Brugeren holder et varmeforhøjende element foran temperatursensoren. 3. Når temperaturssensoren når over 28 grader Celsius afmåler den temperaturen på daværende tidspunkt, og returnerer temperaturmålingen til brugeren. gennem en streng med en alarmerende beskrivelse. 4. Temperatursensoren bliver ved med at aflæse og returnere temperaturmålingen til brugeren i 1 sekunds interval, så længe temperaturen er over 28 grader Celsius. 5. Brugeren aflæser temperaturmålingerne på brugergrænsefladen.

	6. Brugeren afventer til temperaturen falder under 28 grader Celsius. 7. Brugeren observerer strengen med temperaturmålingen forsvinde fra brugergrænsefladen.
UDVIDELSER/UNDTAGELSER	

Tabel 3: UC2 - Forhøjet temperatur opfanges

USE CASE 3 BEVÆGELSE OPFANGES	
MÅL	Bevægelsessensor har oplyst, at der har været bevægelse samtidigt med, at kameraet er blevet vist overfor brugeren gennem brugergrænsefladen.
INITIERING	Brugeren åbner for brugergrænsefladen gennem computer.
AKTØRER	Bruger, Kamerasensor og Bevægelsessensor.
ANTAL SAMTIDIGE FOREKOMSTER	1
PRÆKONDITION	Kamerasensoren og bevægelsessensoren er tilsluttet strøm, har forbindelse mellem hinanden og er tilkoblet gennem brugergrænsefladen.
POSTKONDITION	Kamerasensoren og bevægelsessensoren er tilsluttet strøm, og har stadig forbindelse til hinanden og brugergrænsefladen.
HOVEDSCENARIE	<ol style="list-style-type: none"> Bevægelsessensoren starter i sin default-stadie med strengen "No motion!", hvor den er klar til at detektere bevægelse. Der bliver registreret en bevægelse af bevægelsessensoren. Der bliver nu ændret på default-strengen fra "No motion!" til "Motion detected!" på brugergrænsefladen. Brugeren kan nu observere gennem brugergrænsefladen - via en streng, live video og senere en videooptagelse - at der har været opfanget bevægelse på sensoren, og hvad der er foregået. Efter et delay (forsinkelse) på 5 sekunder, vil bevægelsessensoren og kamerasensoren vende tilbage til sit default-stadie.

	6. Systemet er klar til at detektere bevægelse igen.
UDVIDELSER/UNDTAGELSER	

Tabel 4: UC3 - Bevægelse opfanges

MoSCoW (Louise, Hafsa Mads, August, Altaf og Ruben)

I dette afsnit er MoSCoW-metoden blevet brugt til at udarbejde nogle krav, som de forskellige komponenter/sensorer **skal have**, **bør have**, **kunne have** og **skal ikke have**.

Temperatursenor (Louise og Hafsa)

LM75

Must have:

- Brugerens **skal** hele tiden kunne tilgå systemet og aflæse temperaturmåleren via GUI.
- GUI **skal** sende data hvis temperaturen når over 28 grader.

Should have:

- Sensoren **bør** kunne justeres i dens følsomhed af temperaturmåling, så den kan tilpasses bestemte miljøer.
- Gui **bør** kunne benyttes af en ikke instrueret person.
- Systemets spændingsforsyning **bør** have en udformning så al forsyning gemmes væk bag et stik, der uden videre kan sættes i stikkontakten.

Could have:

- **Kunne** være i stand til at måle i både celsius og fahrenheit.
- **Kunne** måle både + og minus grader.
- **Kunne** have både analogt og digitalt display.
- Sensoren **kunne** have evnen til at blive koblet sammen med andre temperatursensorer, så man kan dække flere dele af et hus ind.

Won't have (this time):

- Systemet **vil ikke** kunne alarmere hvis temperaturen er under 25 grader.

2.2 Kamera (*Mads og August*)

Raspberry Pi Camera Board v1.3 (5MP, 1080p)

Must have:

- Kameraet **skal** kunne sende live-feed til en hjemmeside med en grænseflade.
- Kameraet **skal** kunne køre konstant.
- Kameraet **skal** kunne have en direkte forbindelse til en Raspberry Pi 3 Model B+, som står for håndtering af signal.

Should have:

- Kameraet **bør** kunne lagre gemte optagelser, som senere kan genafspilles.
- Kameraet **bør** kunne aktiveres, når der opfanges bevægelse på bevægelsessensoren
- Kameraet **bør** kunne streame uden store forsinkelser
- Kameraets kvalitet **bør** være uden stort kompromis

Could have:

- Kameraet **kunne** optage lyd
- Kameraet **kunne** have en indikator, der viser at det er tændt
- Kameraet **kunne** have en fysisk tænd og sluk knap.

Won't have (this time):

- Kameraet **vil ikke** være infrarødt. (*det vil kun kunne virke om dagen*)
- Kameraet vil være statisk, og **vil ikke** kunne bevæges med motor efter montering.

2.3 Bevægelsessensor (Altaf og Ruben)

Must have:

- Bevægelsessensoren **skal** kunne detektere bevægelse i det område den overvåger (det spænd - i grader - den er sat til).
- Sensoren **skal** have en timer med delay på 30 sekunder.
- Sensoren **skal** interagere med kameraet.

Should have:

- Sensoren **bør** kunne gemmes væk så den ikke er nemt synlig.
- Sensoren **bør** kunne justeres i dens følsomhed af bevægelsesdetektion, så den kan tilpasses bestemte miljøer.
- Brugeren **bør** kunne aktivere og deaktivere bevægelsessensoren via. GUI.

Could have:

- Sensoren **kunne** i bestemte tidsintervaller sættes til at være hhv. aktiv eller inaktiv.
- Brugeren **kunne** have muligheden for at tilvælge sig notifikation via E-mail på GUI
- Sensoren **kunne** have evnen til at blive koblet sammen med andre bevægelsessensorer, så man kan dække flere dele af et hus ind.

Won't have (this time):

- Sensoren **vil ikke** have indbygget ansigtsgenkendelse til at have forudbestemt godkendte personer i husstanden.
- Sensoren **vil ikke** have indbygget lydoptagelse, så man kan høre hvad der sker når den detekterer bevægelse.

Ikke-funktionelle krav (Louise, Hafsa Mads, August, Altaf og Ruben)

I dette afsnit vil de ikke-funktionelle krav til systemet være beskrevet. De ikke-funktionelle krav er delt op i kategorierne Fysiske dimensioner, GUI og Andet.

Temperatursensor (Louise & Hafsa)

Fysiske dimensioner:

- 1.1 Temperaturmåleren skal være: 18 x 8 x 2 (± 2) cm.
- 1.2 Temperatursensoren må maksimalt veje 20 gram.

GUI:

- 2.1 Temperaturmålingerne skal kunne aflæses på GUI.
- 2.2 GUI skal opdatere temperaturmåling hvert sekund, så længe temperaturen er over 28 grader og hvert 2. sekund så længe temperaturen er under 28 grader

Andet:

- 3.1 Temperaturmåleren kan klare varme op til 45 grader og kulde ned til -5 grader.

Kamera (Mads og August)

Raspberry Pi Camera Board v1.3 (5MP, 1080p)

Fysiske dimensioner:3.2 Raspberry Pi Zero skal være: $10.0 \text{ cm} \times 5.5 \text{ cm} \times 3.0 \text{ cm}^3 \pm 2 \text{ cm}$ (Reichelt, 2023).3.3 Raspberry Pi Camera Board v1.3 (5mp, 1080p) skal være: $0.5 \text{ cm} \times 0.4 \text{ cm} \times 0.1 \text{ cm} \pm 0.25 \text{ cm}$ (Components, 2023).3.4 Kameraets optagelsesopløsning målt i pixels skal være $320\text{px} \times 240\text{px} \pm 100\text{px}$.**GUI:**2.1 Kameraets videovisningsopløsning på brugergrænsefladen målt i pixels skal være $320\text{px} \times 240\text{px} \pm 100\text{px}$.2.2 Kameraets opdateringshastighed skal være $15\text{fps} \pm 5\text{fps}$ ⁴.

2.3 Kameraets livevideooptagelse skal kunne slås til og fra manuelt af brugeren.

2.4 Reaktionstiden fra når man manuelt trykker ”slå til” til at kameraet livevideooptagelse fremvises overfor brugeren skal være 5 sekunder ± 2 sekunder.2.5 Reaktionstiden fra når man manuelt trykker ”slå fra” til at kameraet livevideooptagelse ikke længere fremvises overfor brugeren skal være 5 sekunder ± 2 sekunder.2.6 Reaktionstiden fra når kameraet får signal fra bevægelsessensoren til at kameraet livevideooptagelse fremvises overfor brugeren skal være 5 sekunder ± 2 sekunder.**Andet:**

3.1 Raspberry Pi Zero må maksimalt veje 75.0 gram.

³ Længde x Bredde x Højde⁴ FPS: Opdateringshastigheden af billeder målt per sekund

*Bevægelsessensor (Altaf og Ruben)***Fysiske dimensioner:**

- 1.1 Bevægelsessensoren skal være: 5 (bredde) x 3 (højde) x 8 (længde) (± 2) cm.
- 1.2 Vægt og udseende rettes til ved hardware-design.

GUI:

- 2.1 Når bevægelsessensoren detekterer bevægelse, skal den sende et signal til GUI'en.
- 2.2 På GUI'en bliver der printet en string-tekst, som en advarsel på detektering af bevægelse.
- 2.3 Da bevægelsessensoren og kameraet er koblet sammen, sendes der også et signal til kameraet (dette uddybes i sektion omkring kameraet).

Andet:

- 3.1 Når bevægelsessensoren detekterer bevægelse, ryger den i ON-tilstand.
- 3.2 Samtidigt startes en timer (delay) på 30 sekunder, der beholder bevægelsessensoren i sin ON-tilstand, indtil timeren rammer 0 sekunder, hvor den slukker (ryger i OFF-tilstand).
- 3.3 Hvis bevægelsessensoren detekterer bevægelse inden timeren har ramt 0 (altså under nedtælling), sættes timeren tilbage til dens startværdi (30 sekunder), og begynder nedtællingen forfra samtidigt med, at kamerasensoren bliver ved med at være aktiv.

Accepttestspezifikation for funktionelle krav

Use Case 1: Manuelt tjek af systemets komponenter (Mads og August)

Use case under test		Manuelt tjek af systemets komponenter	
Scenarie		Hovedscenariet	
Prækondition		Alle komponenter i smart-hjemmesikkerhedssystemet er tilsluttet strøm, og har forbindelse til brugergrænsefladen. Computeren er tilsluttet internettet.	
Step	Handling	Forventet observasjon/resultat	Faktisk observation/resultat
1	Brugeren åbner for brugergrænsefladen (GUI) ved hjælp af computer.	Brugergrænsefladen (GUI) indlæses på skærmen og vises	
2	Temperatursensoren afmåler temperaturen på daværende tidspunkt, og returnerer temperaturmålingen til GUI'en via en streng.	Hvis temperaturen er over 28 grader Celsius, så kommer en "HIGH TEMP ALERT" streng på GUI'en.	
3	Brugeren aflæser temperaturmålingen og en "HIGH TEMP ALERT" streng	Temperaturmålingen og en "HIGH TEMP ALERT" streng vises klart og tydeligt på GUI.	
4	Bevægelsessensoren frakobles, og brugeren tjekker ved at manuelt foretage en bevægelse foran sensoren, at intet bliver opfanges af sensoren.	En "NO motion" streng vises på brugergrænsefladen (GUI)	
5	Bevægelsessensoren tilkobles, og brugeren tjekker ved at foretage en bevægelse foran sensoren, at der opfanges bevægelse.	Brugergrænsefladen (GUI) viser "Motion detected" streng.	

6	Når bevægelse opfanges af sensoren, bliver en streng returneret til brugergrænsefladen, som fortæller brugeren at bevægelse er blevet opfanget, og kamera-sensoren aktiveres i 10 sekunder.	Efter at ”Motion detected” strengen vises på GUI’en, aktiverer kameraet automatisk og påbegynder stream til GUI.	
7	Brugeren aflæser strengen, som fortæller ham/hende, at bevægelse er blevet opfanget, og observerer live videooptagelse, som kamerasensoren fremviser. Alt sammen gennem brugergrænsefladen.	brugergrænsefladen (GUI) viser video fra kamera og strengen.	
8	Efter 10 sekunder fjernes strengen fra brugergrænsefladen, som fortæller brugeren, at bevægelse er blevet opfanget, og kamerasensorens live videooptagelse stoppes også med at blive fremvist på brugergrænsefladen.	”Motion detected” strengen og kameravideoen vises kun i løbet af 10 sekunder på brugergrænsefladen (GUI).	

Tabel 5: Use case 1: Manuelt tjek af systemets komponenter

Use case 2: Forhøjet temperatur opfanges (Louise og Hafsa)

Use case under test		Forhøjet temperatur opfanges	
Scenarie		Hovedscenariet	
Prækondition		Temperaturssensoren er tilsluttet strøm, og har forbindelse til brugergrænsefladen. Temperatursensoren mäter temperaturen.	
Step	Handling	Forventet observasjon/resultat	Faktisk observation/resultat

1	Temperatursensoren begynder i sin standardtilstand, hvor den mäter stuetemperaturen.	Brugerfladen er aktiv og mäter den aktuelle stuetemperaturen.	Vi ser at stuetemperaturen bliver vist på GUI'en, med 2 sekunders interval.
2	Brugeren holder et varmeforhøjende element foran temperatursensoren.	Temperaturen stiger til 28 + grader og vi modtager en 'High temperature alert' besked på GUI'en	Vi holder en varm finger på temperatursensoren, og ser på GUI'en at temperaturmålingerne stiger over 28 grader og vi modtager en 'High temperature alert' besked
3	Temperatursensoren mäter temperaturen over 28 grader Celsius og returnerer temperaturmålingen en tekst med en alarmerende besked til GUI'en.	Brugergrænsefladen (GUI) viser den opdateret temperaturmåling i 1 sekunders interval samtidigt med en 'high temperature alert' besked.	Vi ser at at GUI'en viser den opdateret temperaturmåling i 1 sekunders interval og vi ser også en 'high temperature alert' besked på GUI'en
4	Brugeren aflæser både de opdateret temperaturmålingerne og den alarmerende besked på brugergrænsefladen.	Temperaturen vises og kan aflæses i brugergrænsefladen (GUI).	Brugeren aflæser på GUI'en både de opdateret temperaturmålingerne og den alarmerende besked som siger "high temperature alert" på brugergrænsefladen
6	Brugeren afventer, indtil temperaturen falder under 28 grader Celsius.	Brugergrænsefladen (GUI) opdaterer målingen hvert sekund, så længe temperaturen er over 28 grader.	Vi fjerner vores finger fra temperatursensoren, og observere at temperaturmålingen falder på GUI'en
7	Brugeren observerer, at beskeden med temperaturmålingen forsvinder fra brugergrænsefladen.	Brugergrænsefladen (GUI) vender tilbage til normalen og opdatere nu hvert 2. sekund	Vi observerer at den alarmerende beskeds som siger 'high temperature alert' forsvinder

			når temperaturmålingerne bliver under 28 grader. Nu vises opdatering af temperaturmåling i 1 2 sekunders intervaller på GUI'en
--	--	--	--

Tabel 6: Use case 2: Forhøjet temperatur opfanges

Use case 3: Bevægelse opfanges (Altaf og Ruben)

Use case under test		Bevægelse opfanges	
Scenarie		Hovedscenariet	
Prækondition		Kamerasensoren og bevægelsessensoren er tilsluttet strøm, har forbindelse mellem hinanden, og er tilkoblet gennem brugergrænsefladen.	
Step	Handling	Forventet observation/resultat	Faktisk observa-tion/resultat
1	Bevægelsessensoren starter i sin default-stadie, hvor den er klar til at detektere bevægelse.	Brugergrænsefladen er aktivt og i standby tilstand, klar til at reagere på eventuel bevægelse.	
2	Der bliver registreret en bevægelse af bevægelsessensoren.	Brugergrænsefladen viser at en bevægelse er blevet detekteret, med en besked eller en indikator for at signalere dette.	
3	Bevægelsessensoren opfanger bevægelsen, og starter en timer på 30 sekunder.	Sensoren har startet en timer i brugergrænsefladen som tæller ned fra 30 sekunder.	
4	Bevægelsessensoren returnerer en streng til brugergrænsefladen samtidigt	Kamerasensoren aktiveres og begynder at optage i 30 sekunder.	

	med, at den sender et signal til kamera-sensoren om at den skal aktiveres.		
5	Brugeren kan nu observere gennem brugergrænsefladen - via en streng - at der har været opfanget bevægelse på sensoren.	Brugergrænsefladen viser en besked eller en streng, der informerer brugeren om, at der er blevet opfanget bevægelse på sensoren.	
6	Efter timeren når 0, fjernes strengen, og bevægelsessensoren returnerer til sin default-stadie.	Brugergrænsefladen (GUI) vender tilbage til normalen og systemet er nu i en ventetilstand og venter på at ny bevægelse bliver detekteret.	.

Tabel 7: Use case 3: Bevægelse opfanges.

Test af ikke funktionelle krav

I dette afsnit er de ikke-funktionelle krav, specificeret i afsnit 4 i Kravspecifikation, blevet testet.

Til udførelsen af testen anvendes følgende udstyr.

Udstyr til test af krav:

- Målebånd (bruges til alle test med en længde)
- Køkkenvægt (bruges til vejning af de forskellige ting)
- Stopur (bruges til tidtagning af alle tests med et tidsinterval)
- Varmeforhøjende element (Til påvirkning af temperaturmåler)

Temperatursensor (Hafsa og Louise)

Accepttestspezifikation for ikke-funktionelle krav (Temperatursensor)

Krav nr.	Krav	Faktisk Test/resultat	God-kendt (JA/NEJ)
Fysiske dimensio- ner 1.1	Temperatursensoren skal have dimensioner på max 8 x 8 (± 2) cm.	Måletest: Temperatursensoren måles, og dimensionerne viser til at være på 18 x 8 x 2 (± 2) cm.	
GUI: 2.1	Temperaturmålingerne skal kunne aflæses på GUI'en	Visuel test: Temperaturmålingerne vises og aflæses på GUI'en	
Andet: 3.1	Temperatursensoren må maksimalt veje 20 gram.	Måletest: Temperatursensoren vejer maks. 20 gram.	
Andet: 3.2	Temperatursensoren kan klare varme op til 45 grader og kulde ned til -5 grader.	Visueltest: Temperatursensoren måler temperaturen mellem -05 grader og 45 grader	

Tabel 8: Accepttestspezifikation for ikke-funktionelle krav for temperatursensor

Kamera (Mads og August)

Raspberry Pi Camera Board v1.3 (5mp, 1080p)

Accepttestspezifikation for ikke-funktionelle krav

Krav nr.	Krav	Faktisk Test/resultat	Godkendt (JA/NEJ)
Fysiske dimensioner 1.1	Arduinoen skal være: 10.0 cm x 5.5 cm x 3.0 cm ¹ ±2 cm (MiniElektro, 2023).	Måletest: Arduinoen måles med målebånd, og dimensionerne viser til at være 10.0 cm x 5.5 cm x 3.0 cm ¹ ±2 cm	
Fysiske dimensioner 1.2	Raspberry Pi Camera Board v1.3 (5mp, 1080p) skal være: 20 mm x 25 mm x 9 mm ±25 mm (Lavpris, 2023)	Måletest: Kamera måles med målebånd, og dimensionerne viser til at være 20 mm x 25 mm x 9 mm ±25 mm	
Fysiske dimensioner 1.3	Kameraets optagelsesopløsning målt i pixels skal være 320px x 240px ±100px.	Måletest: Kameraets video bliver gemt som en fil, hvori oplosning aflæses.	
GUI: 2.1	Kameraets videovisningsopløsning på brugergrænsefladen målt i pixels skal være 320px x 240px ±100px.	Måletest: Kameraets videoopløsning aflæses på hjemmesiden for at sikre at oplosningen ikke er forringet ved overførsel.	
GUI: 2.2	Kameraets opdateringshastighed skal være 15fps ±5fps ² .	Måletest: Kameraets opdateringshastighed måles igennem hjemmesiden for at sikre at alle frames er inkluderet.	
GUI: 2.3	Reaktionstiden fra når kameraet får signal fra bevægelsessensoren til at kameraet livevideooptagelse fremvises overfor brugeren skal være 5 sekunder ±2 sekunder.	Måletest: Stopur startes ved bevægelse og stoppes ved video vises på GUI	
Andet:	Arduinoen må maksimalt veje 75.0 gram	Måletest: Arduinoen måles på en vægt	

3.1			
-----	--	--	--

Tabel 9: Accepttestspezifikation for ikke-funktionelle krav for kamera

Bevægelsessensor (Altaf og Ruben)
Accepttestspezifikation for ikke-funktionelle krav

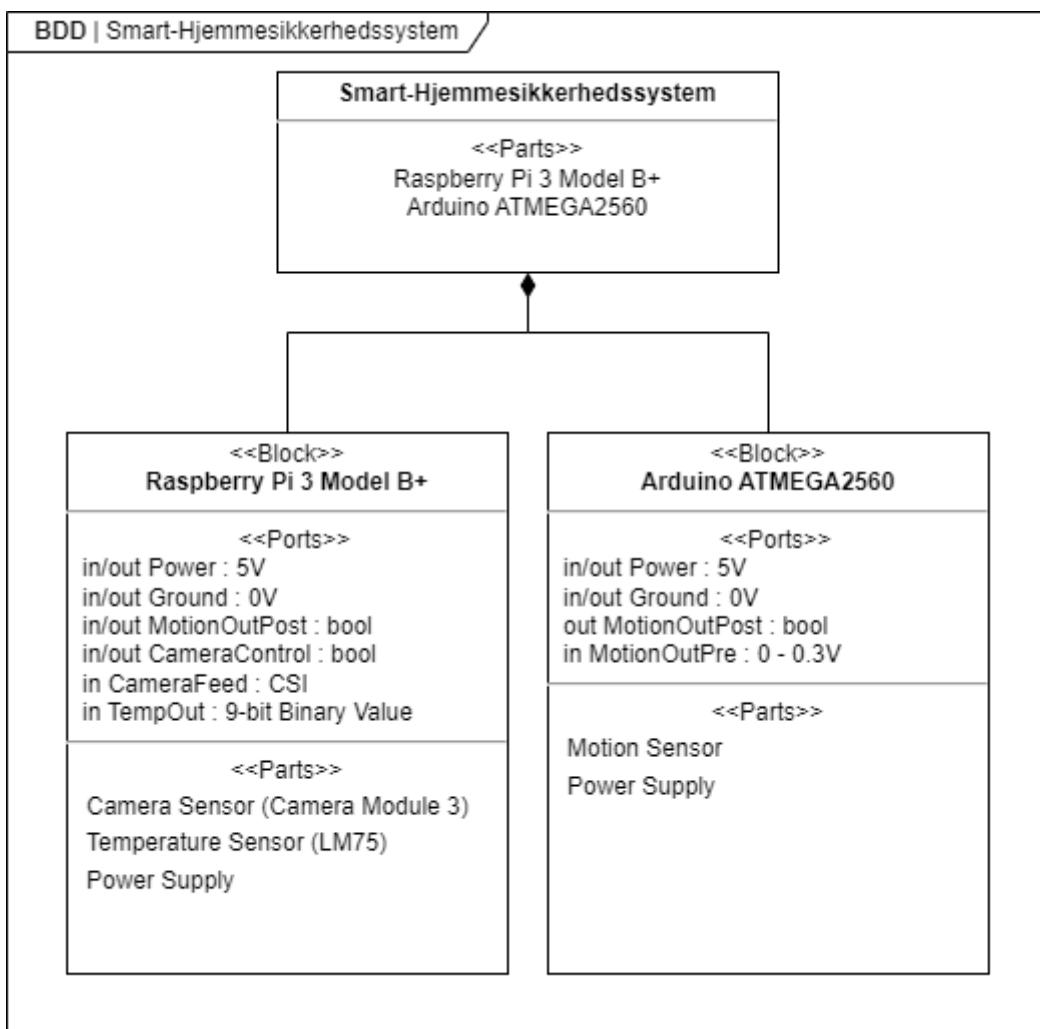
Krav nr.	Krav	Faktisk Test/resultat	Godkendt (JA/NEJ)
Punkt 1	Sensoren skal maksimalt måle 8 (længde) x 5 (bredde) x 3 (højde) ±2 cm	Måle test: Vi bruger en lineal til at måle hver dimension og verificere at de er under den maksimale grænse.	
Punkt 2	Sensoren vejer maksimalt 15 gram	Måle test: Sensoren vejes på en vægt med en nøjagtighed, der er bedre end 10 gram. Vægten skal vise 15 gram eller derunder.	
Punkt 3	Sensoren skal kunne detektere bevægelse i en 60 graders vinkel	Visuel test: Vi tegner en 60 graders vinkel og sensoren placeres i nulpunktet. Herefter laver vi bevægelse hhv. Indenfor 60-graders-arealet, og udenfor 60-graders-arealet. Hertil verificerer vi, at sensoren registrerer bevægelsen igennem GUI.	
Punkt 4	Sensoren sender en string ud på GUI	Visuel test: Når der foretages en bevægelse foran sensoren, ser vi visuelt en tekststreng blive vist på GUI.	

Punkt 5	Sensoren sender et signal til kameraet om at det skal tænde.	Måle test: Når der detekteres bevægelse, tændes kamerastream på GUI'en.	
Punkt 6	Efter bevægelse er detekteret, startes en 30 sekunders timer.	Måle test: Bevægelse foretages foran sensoren og vi verificerer at et signal sendes; hvis dette er tilfældet, så starter vi et stopur på 30 sekunder og venter til det rammer 0 sekunder. Herefter verificerer vi, at sensoren er gået tilbage til 'default tilstand'.	

Tabel 10: Accepttestspezifikation for ikke-funktionelle krav for bevægelsessensor

Systemarkitektur (Mads og August)

Systemet kan overordnet inndeles i 2 delsystemer, som er vist på Figur 13. Et delsystem defineres som et system med egen ”kode”. Det vil sige, at der skal udvikles software til hver af de 2 blokke. Alle de forskellige softwarekoder for de forskellige komponenter samles til en stor samlet softwarekode, som gør de forskellige data fra de forskellige komponenter kan tilgås af brugeren via en brugergrænseflade. Disse to delsystemer sammen er vores system i helhed, den blok kaldet ”Smart-Hjemmesikkerhedssystem”. De to delsystemer arbejder i fællesskab om udførelsen af det overordnede systems ønskede funktioner. Her spiller Arduinoen rollen som ”translatør” af bevægelsessensoren med en simpel ADC. Denne kan tilsluttes Raspberry Pien, og de kan i helhed udføre systemets funktioner.



Figur 13: Block Definitions Diagram (BDD) over det overordnede system (Smart-Hjemmesikkerhedssystem)

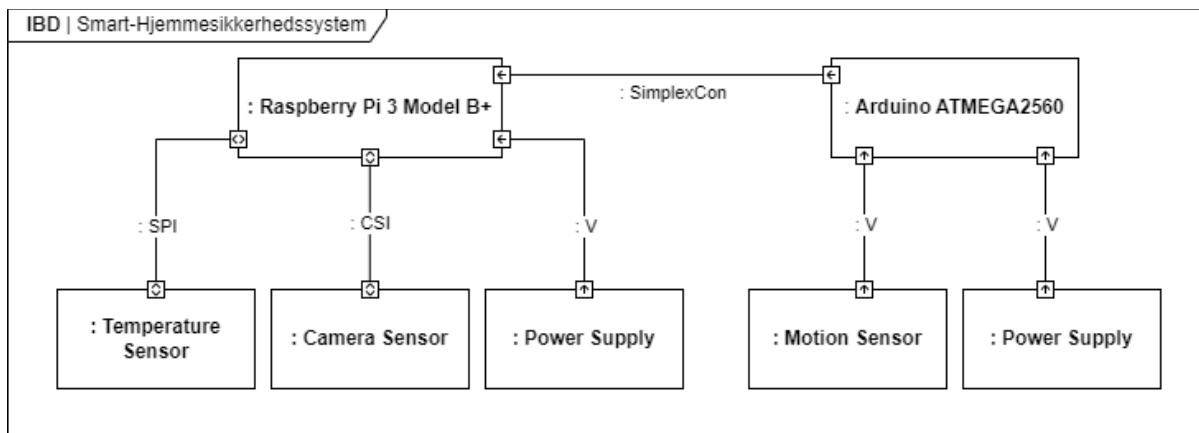
I Tabel 11 følger en kort beskrivelse af hver bloks formål og funktion.

Raspberry Pi 3 Model B+	Står for host af web-server som indehaver GUI til brug af bruger. Står derudover også for at kunne opsamle data fra temperaturmåler og kamera samt bevægelsessensoren der er tilsluttet Arduino.
Arduino ATMEGA2560	Skal udnytte en ADC til at opsamle info om bevægelse fra bevægelsessensoren, og kommunikere med Raspberry pien sådan at den kan reagere i overensstemmelse med ønsket funktioner.

Tabel 11: Beskrivelse af Block Definitions Diagram (BDD) over det overordnede system (Smart-Hjemmesikkerhedssystem)

Grænseflader (Mads og August)

De overordnede forbindelser og det logiske flow mellem delsystemerne er beskrevet i detalje på Figur 14. Her vises de protokollerne Serial Peripheral Interface (SPI), Camera Serial Interface (CSI), Simplex seriel kommunikation (SimplexCon), og Elektrisk Spænding (V) som delsystemerne benytter sig af.

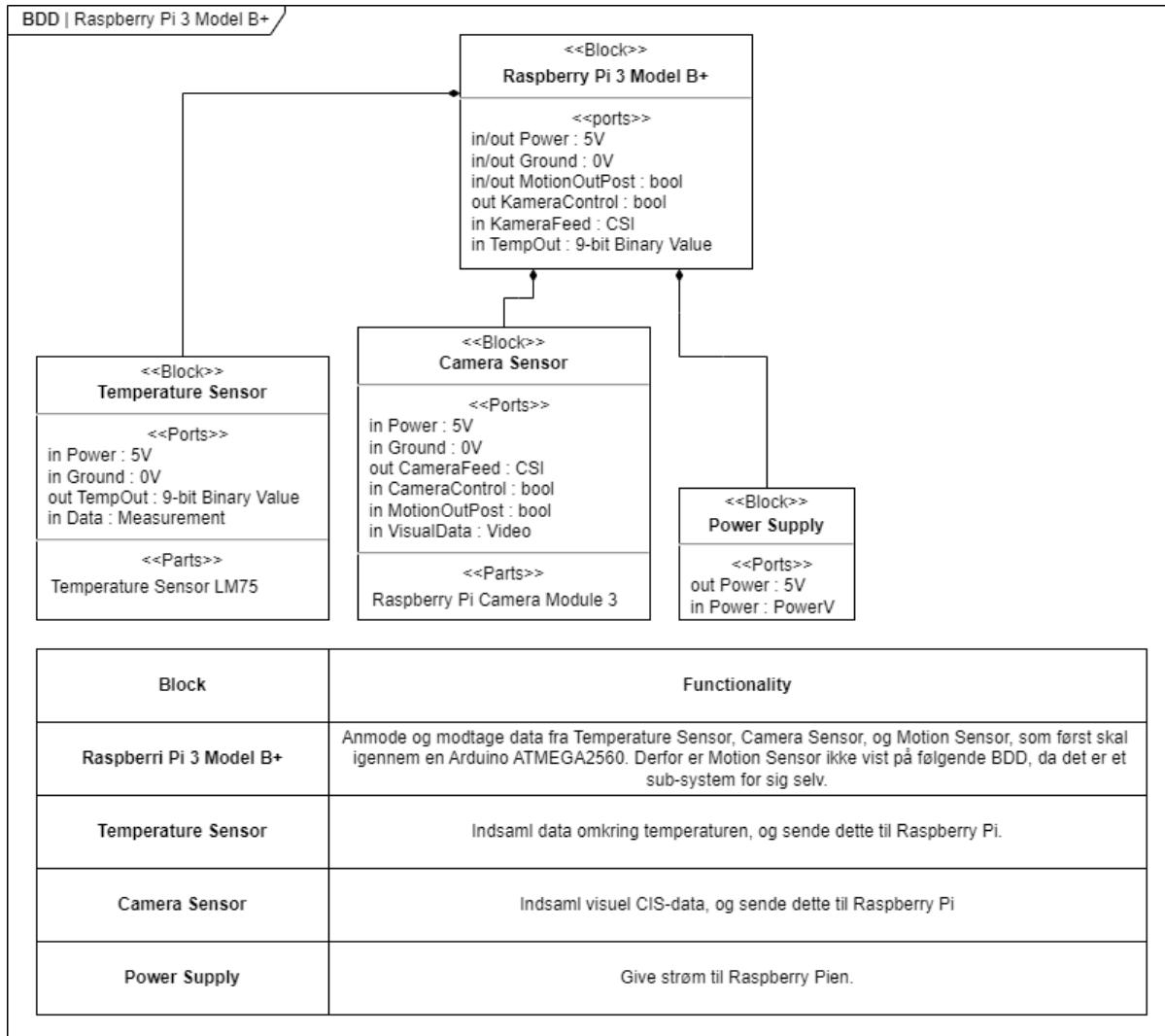


Figur 14: Internal Block Diagram (IBD) over det overordnede system (Smart-Hjemmesikkerhedssystem)

HW-arkitektur

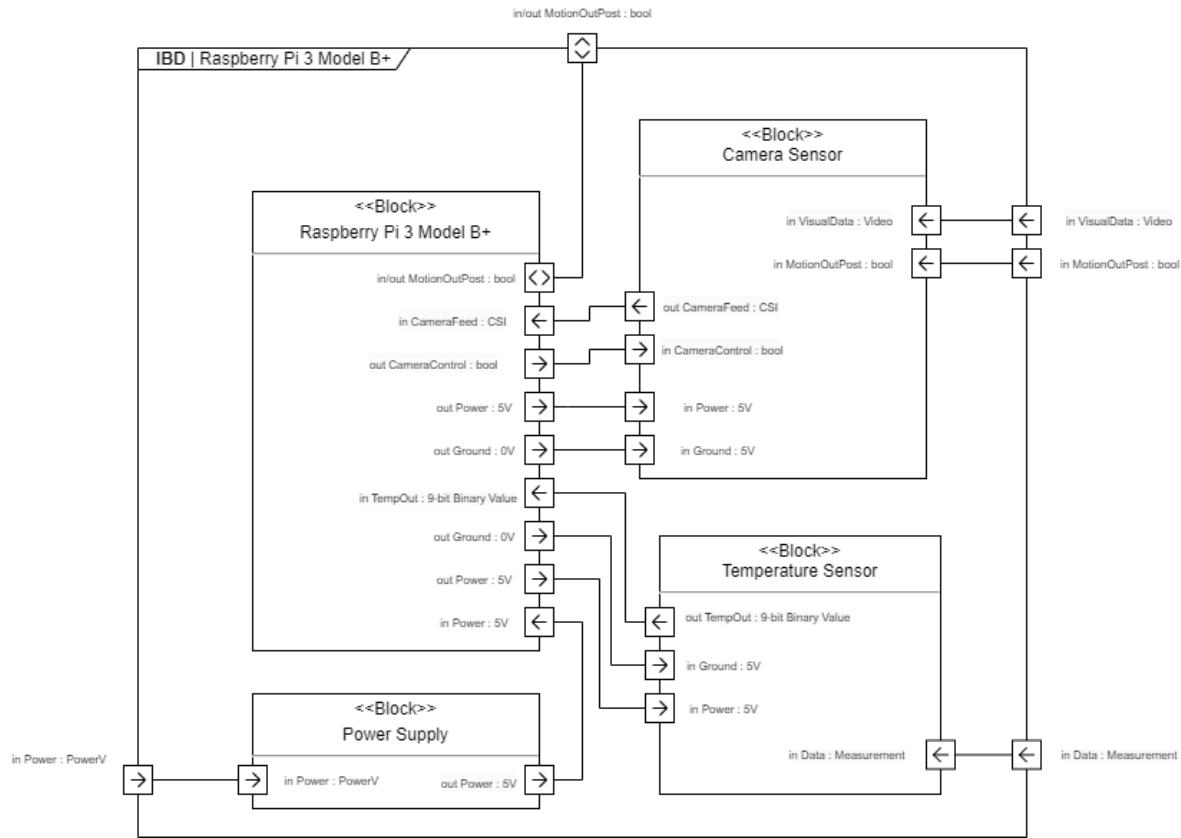
Raspberry Pi (Mads og August)

Block Definition Diagram (BDD) (Mads og August)



Figur 15: Blok Definition Diagram (BDD) for Raspberry Pi 3 Model B+

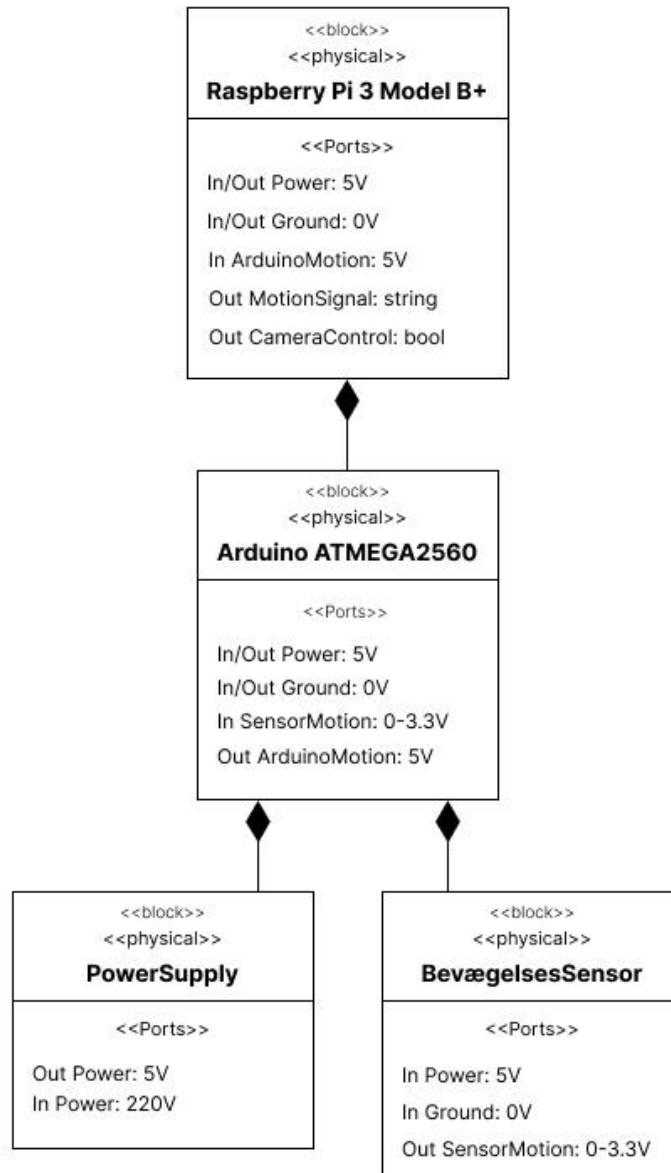
Internal Block Diagram (IBD) (Mads)



Figur 16: Internal Block Diagram (IBD) for Raspberry Pi 3 Model B+

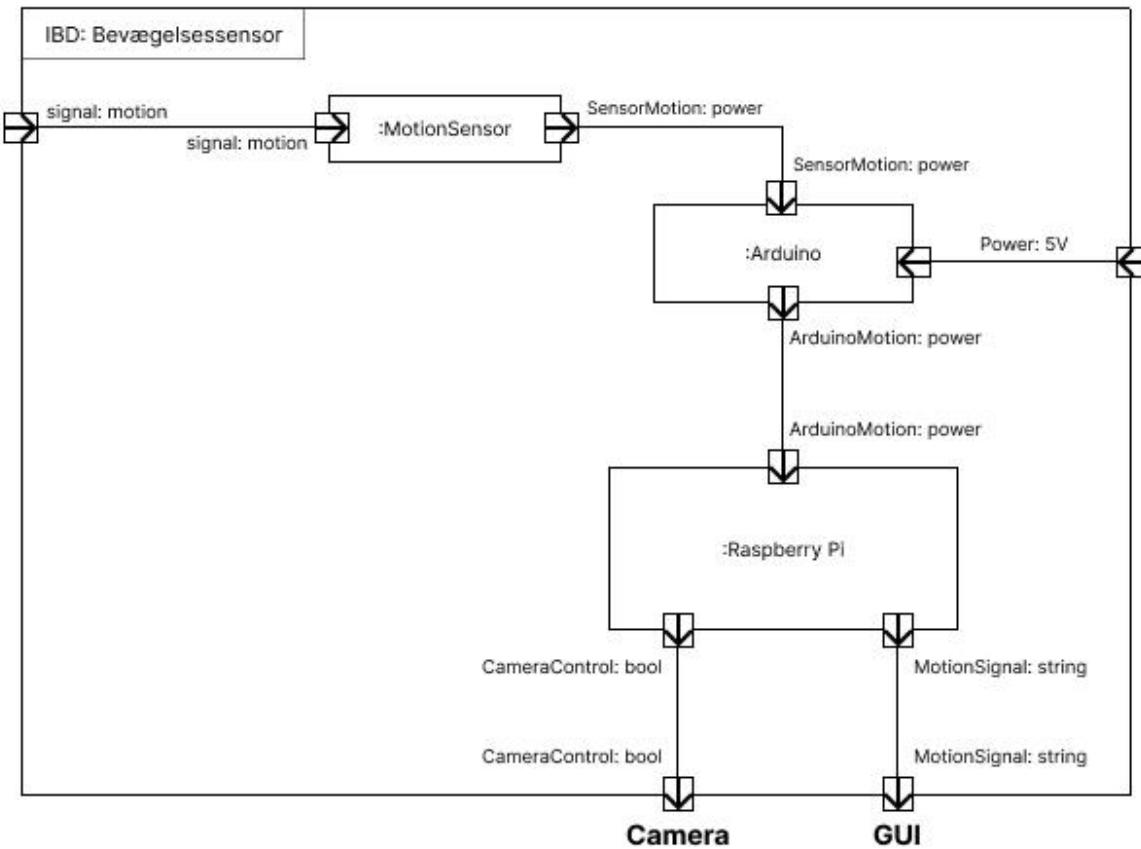
Arduino (Ruben og Altaf)

Block Definition Diagram (Ruben og Altaf)



Figur 17: Bevægelsessense Blok Definition Diagram (BDD)

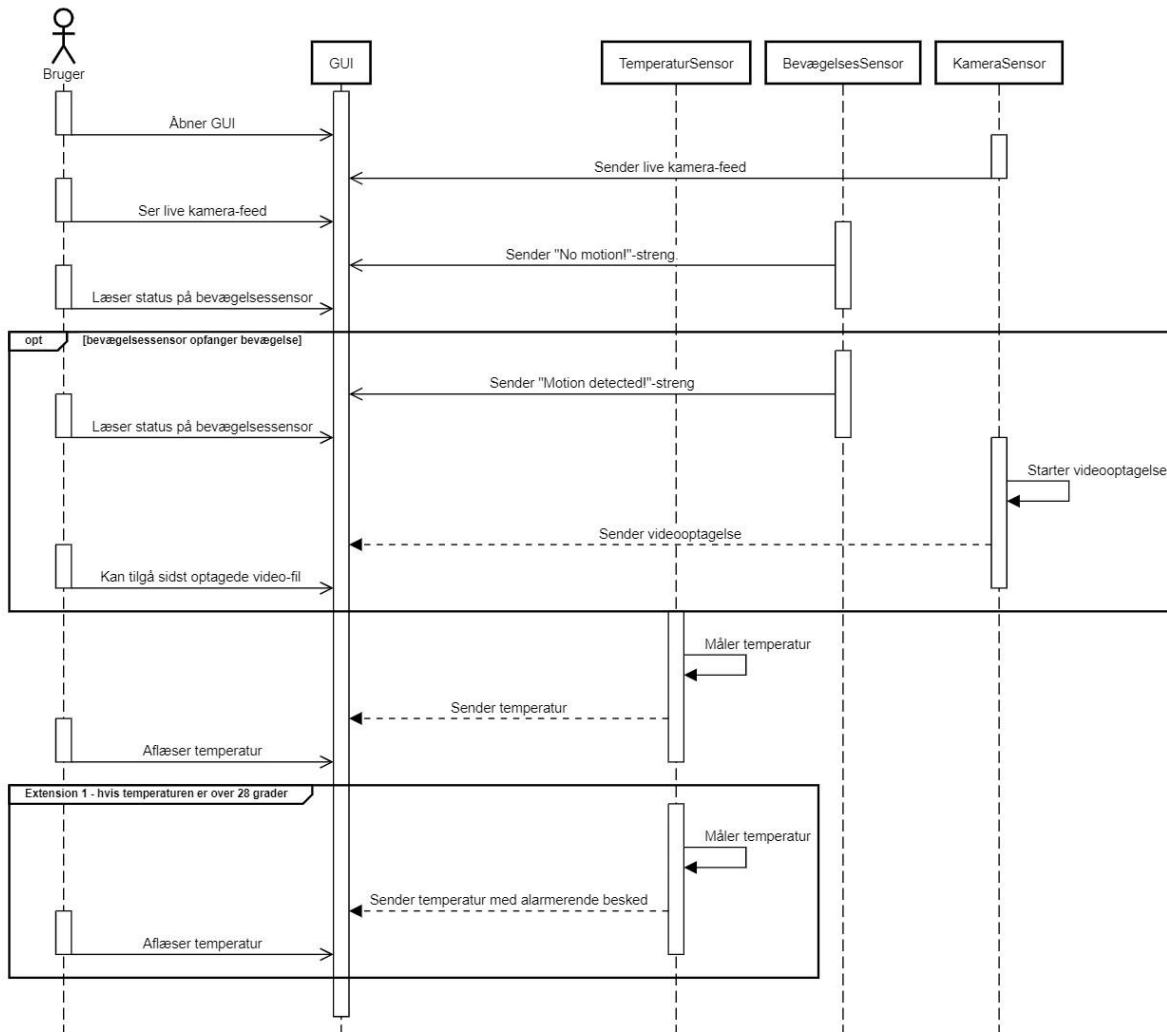
Internal Block Diagram (IBD) (Altaf og Ruben)



Figur 18: Bevægelsessensor Internt Blok Diagram (IBD)

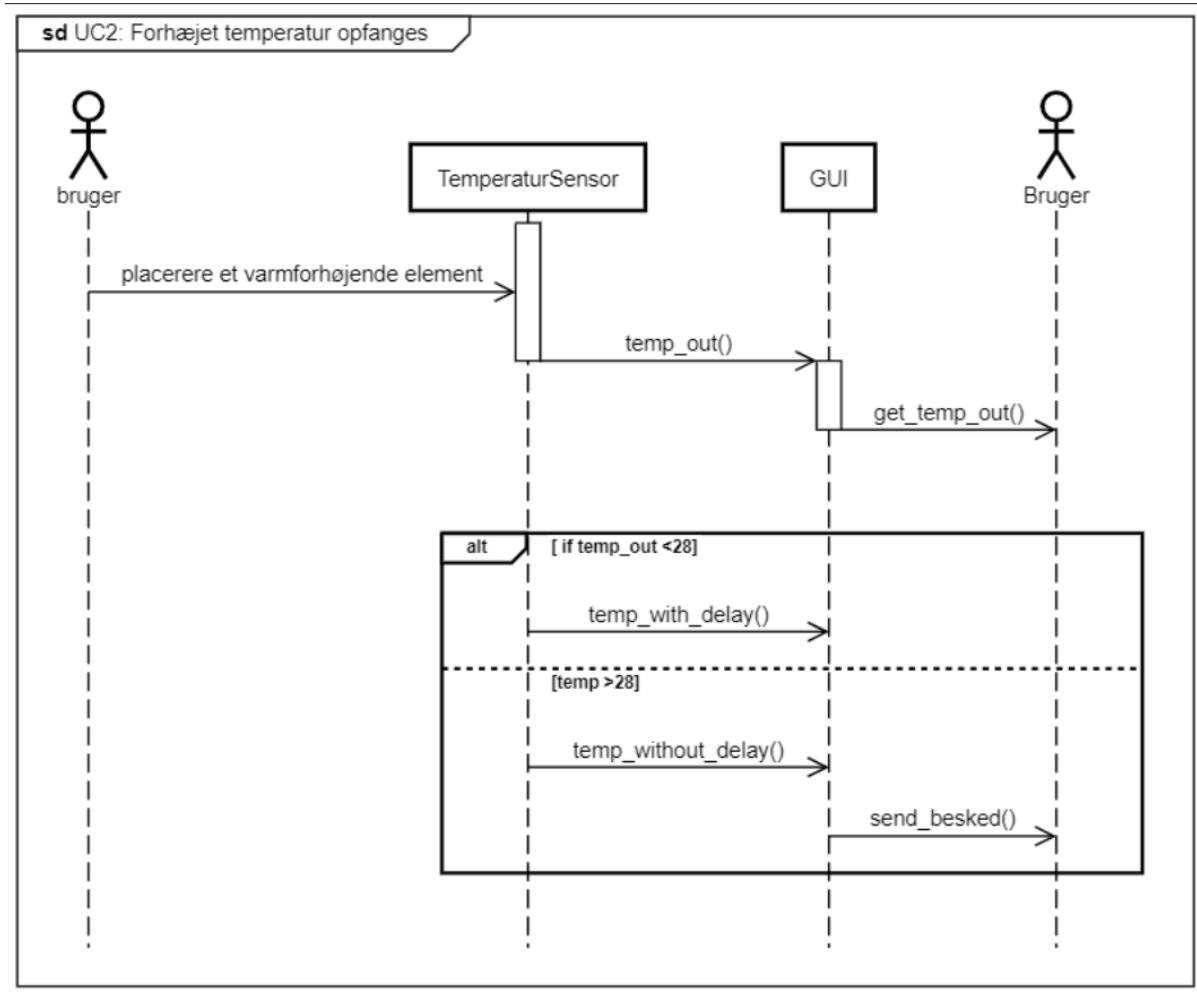
Sekvensdiagrammer

Use case 1: Manuel tjek af systemets komponenter (Altaf)



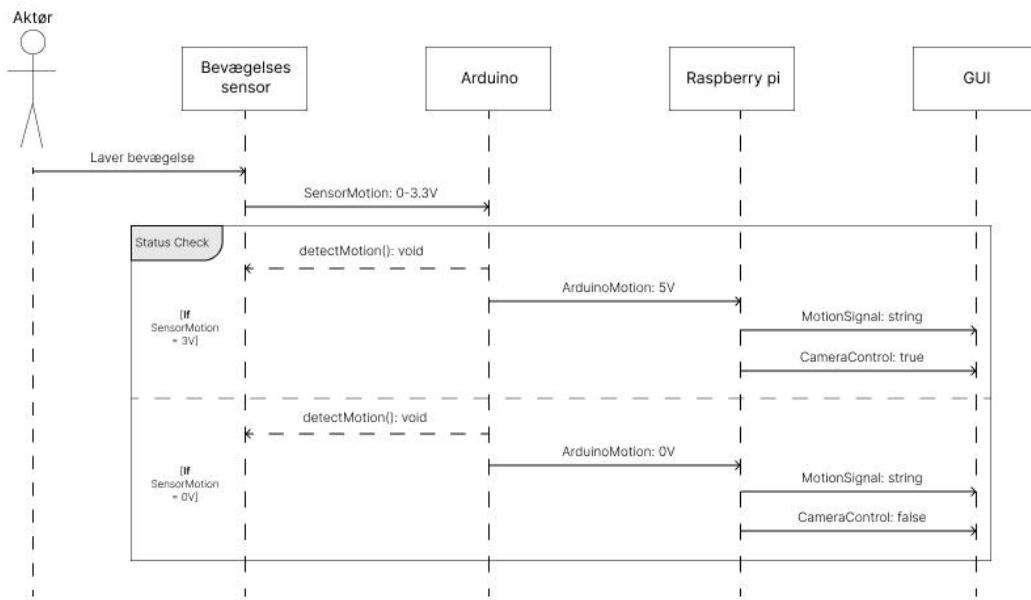
Figur 19: Use case 1: Manuel tjek af systemets komponenter

Use case 2: Temperatursensoren (Hafsa)



Figur 20: Use case 2: Temperatursensoren

Use case 3: BevægelsesSensoren (Altaf og Ruben)



Figur 21: Use case 3: BevægelsesSensoren

Signalbeskrivelser (Ruben, Altaf, Hafsa)

På nedenstående Tabel 12 kan projektets signalbeskrivelser ses:

Signal-navn	Funktion	Område	Port 1 (source)	Port 2 (destination)	Kommentarer
Ground	Reference til analog spænding	0,0V	Power Supply	GUI, ground Bevægelses-Sensor, ground CameraSensor, ground Temperatur-Sensor, ground	GUI indeholder vores Raspberry Pi 0. Dermed refererer vi automatisk til Raspberry Pi 0, når vi skriver GUI.
Power (VCC)	Spænding	5,0V	Power Supply	GUI, power Bevægelses-Sensor, power CameraSensor, power Temperatur-Sensor, power	
Sensor-Motion	Bevægelsesdetektoring	0,0V - 3,3 V (analog signal)	Sensor	ARDUINO 2560	Fra sensoren til Arduino 2560 sendes der et analog signal mellem 0,0V - 3,0V.
ArduinoMotion	Kamerastyring	5,0V	Arduino 2560	Raspberry Pi 3 B+	Her sendes et signal fra Arduino'en til vores relæ, og derefter vores 3,3 V signal til Raspberry Pi'en.
Temp-Out	Temperaturdetektoring	9-bit digital temperatur (-25C til 100C)	Sensor	GUI	Her bruges en LM75A-sensor, som har en indbygget ADC, dermed kan signalet gå direkte gennem GUI'en.
KameraFeed	Kamera live video feed	CSI	Kamera	GUI	Her er det feed som kameraet sender tilbage

					til vores GUI (user inter- face)
Kamera- Control	Kamerastyring (Slukket/Tændt)	Boolean	GUI	Kamera	<p>Her sendes in- formationen om hvorvidt kameraet skal være optage eller ej. Be- stemmes af ArduinoMo- tion signalet.</p> <p>Her indgår både sluk- ket/tændt sta- die for kame- raet.</p>

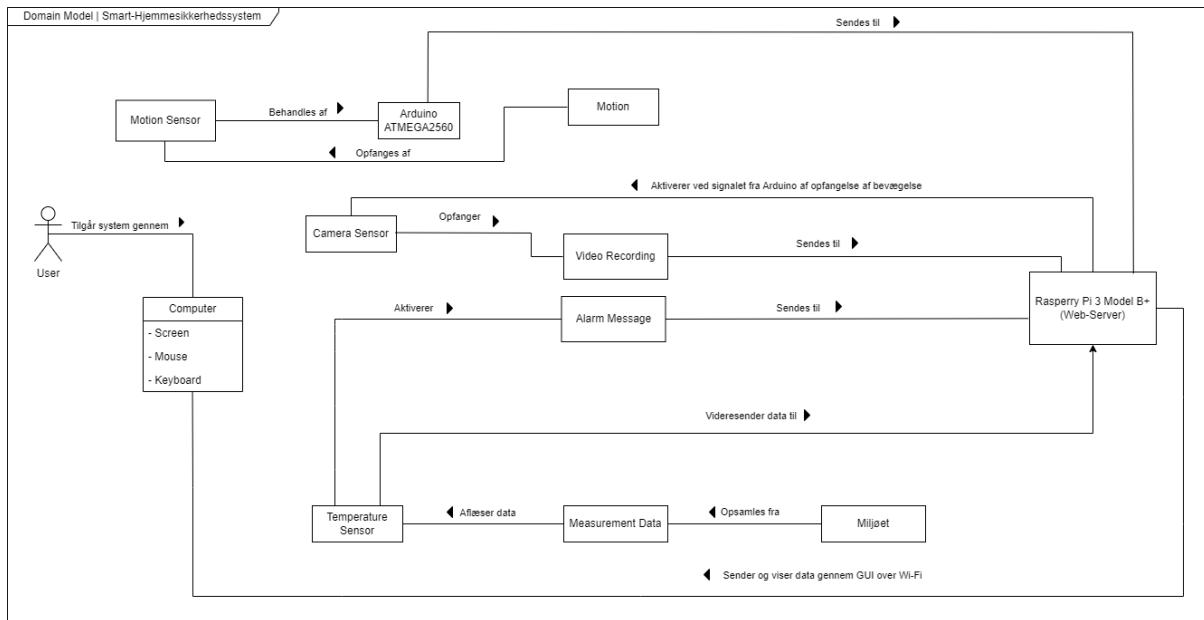
Tabel 12: Signalbeskrivelser for produktet (Smart-Hjemmesikkerhedssystemet)

SW-arkitektur

En domænemodel for hele systemet (Mads og Louise)

Gruppen har i dette afsnit udarbejdet en domænemodel for hele Smart-Hjemmesikkerhedssystemet for at give et bedre overblik over, hvordan hele systemet fungerer overfor brugeren.

Der beskrives let, hvordan de forskellige dele af systemet snakker sammen, og hvordan brugeren "user" tilgår systemet gennem en computer, hvor alt data fra Raspberry Pien bliver vist på en GUI gennem lokalt wi-fi/Ethernet. Alt dette kan observeres på Figur 22.



Figur 22: Domænemodel for hele systemet

Modul- og klassebeskrivelse

Main():

- **Ansvar:** Her kaldes de andre vitale funktioner der skal til for at styre de forskellige dele af vores sikkerhedssystem.
- **Metoder:** ingen
- **Parametre:** ingen

KAMERA (August):

StreamingOutput()

- **Ansvar:** Denne klasse håndterer frames til streaming af video
- **Parametre:** ingen

Metoder:

void __init__(camera):

- **Returværdi:** ingen
- **Beskrivelse:** Initialisering af Klassen, sørger for at frames opdateres med de nyeste billeder til video.

Int write(buf):

- **Returværdi:** int buffer.write(buf)
- **Parametre:** ingen
- **Beskrivelse:** Skriver video-data til bufferen

Attributter:

Byte frame:

- **Default værdi:** "none"
- **Beskrivelse:** Her gemmes den nyeste frame fra kameraet

Byte buffer:

- **Default værdi:** ingen
- **Beskrivelse:** Her gemmes en buffer af frames til kameraet

condition condition:

- **Default værdi:** ingen
- **Beskrivelse:** Denne type attribut kan sikre at en handling har taget sted for at opnå synkronisering

camera camera:

- **Default værdi:** ingen
- **Beskrivelse:** Et objekt af typen "camera" som kan sættes til forskellige parametre og metoder

CameraManager()

- **Ansvar:** Denne klasse laver et kamera-objekt med specifikke parametre vi-deo og har flere metoder til kamera-funktioner
- **Parametre:** ingen

Metoder:

void __init__(camera):

- **Returværdi:** ingen
- **Parametre:** camera
- **Beskrivelse:** Initialiserer kamera objektet

Void start_streaming():

- **Returværdi:** ingen
- **Parametre:** Ingen
- **Beskrivelse:** Starter streamen fra kamera

Void record_video(filename, duration):

- **Returværdi:** ingen
- **Parametre:** filename, duration
- **Beskrivelse:** Optager en video som gemmes

Attributter:

Bool recording:

- **Default værdi:** "false"
- **Parametre:** ingen
- **Beskrivelse:** Opdateres til "true" når video optages

Lock lock:

- **Default værdi:** "none"
- **Parametre:** ingen
- **Beskrivelse:** Holder styr på deling af info til andre classes

BEVÆGELSESSENSOR: (Altaf, Ruben)

BevægelsesSensor(int, int)

- **Ansvar:** Denne klasse styrer bevægelsessensoren
- **Parametre:** int, int

Metoder:

Void initialize():

- **Parametre:** ingen
- **Returværdi:** ingen
- **Beskrivelse:** Her etableres seriel kommunikation.

Void detectMotion():

- **Parametre:** ingen
- **Returværdi:** ingen
- **Beskrivelse:** Når denne metode kaldes, vil vores sensor konstant udføre målinger af vores bevægelsessensor.

Attributter:

Int bevaegelsesValue:

- **Default værdi:** ingen
- **Parametre:** ingen
- **Beskrivelse:** Her gemmes værdien for pirPin_-attributten.

TEMPERATURSENSOR (Hafsa):

TemperaturSensor()

- **Ansvar:** Denne klasse styrer temperatursensoren.
- **Metoder:** Ingen
- **Parametre:** Ingen

Attributter:

Double temp_out

- **Default værdi:** 0.0 (temperaturmålinger i grader)
- **Beskrivelse:** Her gemmes en output-værdi fra temperatursensoren.

Bool temp_status

- **Default værdi:** true
- **Beskrivelse:** Her gemmes status fra temperatursensoren.

Int i2c_address:

- **Default værdi:** 0x4F (LM75-adresse)
- **Beskrivelse:** Her gemmes I2C-adresse for temperatursensoren.

Metoder:

Double get_temp_out()

- **Parametre:** Ingen
- **Returværdi:** Double
- **Beskrivelse:** Denne metode returnerer den seneste temperaturmåling som er gemt i temp_out.

Void print_temperature()

- **Parametre:** Ingen
- **Returværdi:** Ingen
- **Beskrivelse:** Denne metode udskriver den aktuelle temperatur.

String send_besked()

- **Parametre:** Ingen
- **Returværdi:** String
- **Beskrivelse:** Denne metode sender en advarsel hvis temperaturen overstiger 28 grader.

Double temp_without_delay()

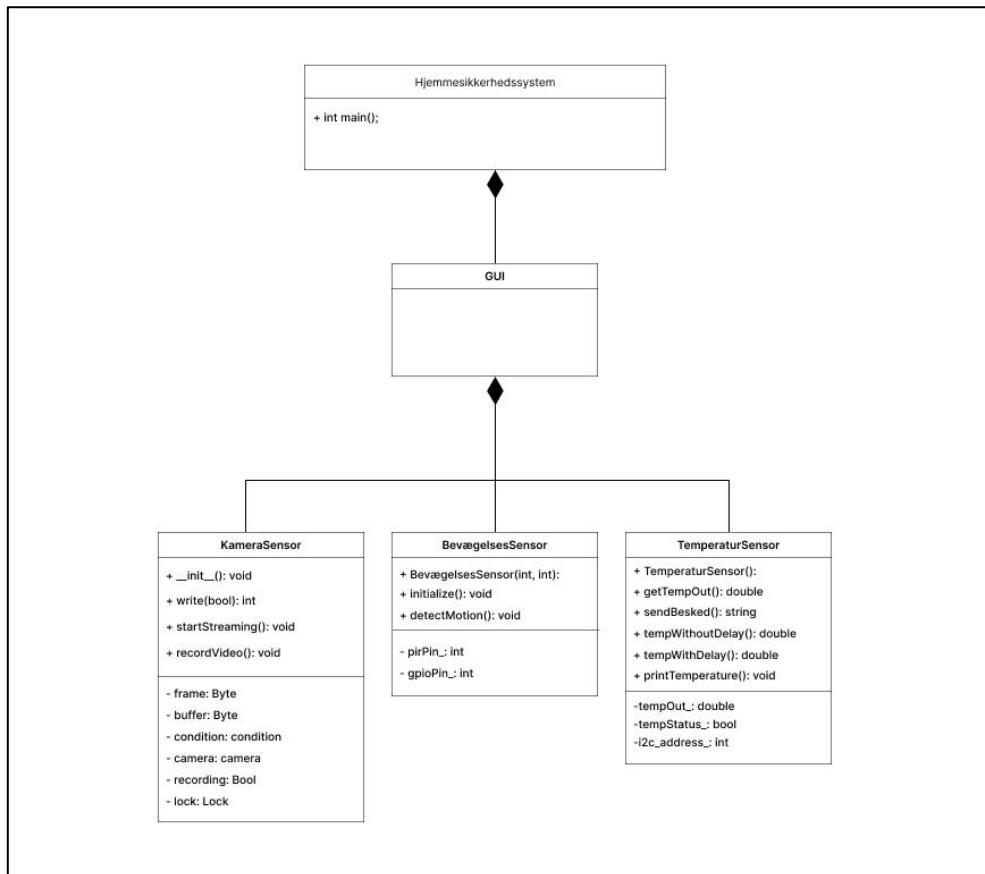
- **Parametre:** Ingen
- **Returværdi:** Double
- **Beskrivelse:** Denne metode aflæser temperaturen fra sensoren uden forsinkelse.

Double temp_with_delay()

- **Parametre:** Ingen
- **Returværdi:** Double
- **Beskrivelse:** Denne metode aflæser med forsinkelse, baseret på den aktuelle temperatur.

Klassediagram for hele projektets software (Ruben)

På nedenstående Figur 23 kan klassediagrammet for de forskellige softwareimplementationer for projektet ses. Dette er både for kamerasensoren, bevægelsessensoren og temperaturssoren samt GUI.



Figur 23: Klassediagram for hele projektets software

SW Design

Denne del uddyber vores software-design som omfatter vores home-security system, hvori der indgår en bevægelsessensor, en temperatursensor, en kamerasensor samt en GUI. Vi har i projektet gjort brug af kodesprogene: C++ og Python.

Implementering

Bevægelsessensor (Altaf):

Denne del fokuserer specifikt på implementeringen af vores bevægelsessensor (MotionSensor), der registrerer bevægelse og dertil muliggør passende handlinger som en del af sikkerhedssystemet.

Funktionalitet af sensoren:

Sensoren leverer et digitalt output, der skifter til "HIGH" når bevægelse registreres, og vendet tilbage til "LOW", når bevægelse ikke registreres. Dette gør det muligt for systemet at reagere på bevægelse og dermed sende besked videre til vores Arduino, som sensoren er koblet til via de passende pins. Nedenfor beskrives initialiseringen af vores sensor, og dertil funktionaliteten af vores pins yderligere.

Initialisering af bevægelsessensoren:

Ved opstart af softwaren foretages der en konfiguration af de involverede pins på Arduino-boardet. Denne konfiguration er afgørende for, hvordan Arduino kommunikerer med de tilsluttede enheder, som i dette tilfælde er bevægelsessensoren og Raspberry Pi'en. Dermed er denne konfiguration altafgørende for en funktionel kommunikation mellem de respektive dele af systemet.

Pin-definitioner:

På nedenstående code-snippet kan ses de pins vi benytter for vores sensor, som hhv. er pirPin (pin 2), og gpioPin (pin 13):

```
BevaegelsesSensor(int pirPin, int gpioPin);  
  
BevaegelsesSensor BevaegelsesSensor(2, 13); //pirPin = 2, gpioPin = 13
```

- **pirPin (Pin 2):** Denne pin går direkte fra vores bevægelsessensor til vores Arduino, og er ansvarlig for at modtage output-signaler fra bevægelsessensoren. Disse signaler kan variere mellem "HIGH" og "LOW", og indikerer henholdsvis tilstedeværelsen eller fraværet af bevægelse.
- **gpioPin (Pin 13):** Denne pin går fra Arduino'en til vores Raspberry Pi, og fungerer som en kommunikationskanal mellem de to enheder - deraf navnet GPIO-pin. Når bevægelse detekteres af bevægelsessensoren, udsender Arduino-boardet et signal gennem denne pin til Raspberry Pi'en, og dette signal viderefører informationen om detekteret bevægelse.

Initialisering af pirPin (Input):

Denne del forklarer hvordan vores input-pin initialiseres. pirPin, som er ansvarlig for at modtage output fra bevægelsessensoren, bliver i denne del deklareret som en input-pin. Dette fortæller Arduino, at den skal forvente at modtage information fra bevægelsessensoren via.

denne pirPin (input-pin):

```
pinMode(pirPin, INPUT);  
pirPin_ = pirPin;
```

Initialisering af gpioPin (Output):

Denne del forklarer hvordan vores output-pin initialiseres. gpioPin, der sender signaler til Raspberry Pi'en, bliver i denne del deklareret som en output-pin. Dette fortæller Arduino at den via. denne gpioPin (output-pin), skal sende enten "HIGH"- eller "LOW"-output til Raspberry Pi'en, som som Raspberry Pi'en kan aflæse og reagere på:

```
pinMode(gpioPin, OUTPUT);  
gpioPin_ = gpioPin;
```

Derudover etableres seriell kommunikation, hvilket opnås ved at specificere en baudrate på 9600:

```
void BevaegelsesSensor::initialize()
{
    Serial.begin(9600);
}
```

Bevægelsesdetektion:

I selve koden for bevægelsesdetektion, starter vi med at definere funktionen loop(), da denne er nødvendig i Arduino-sketch:

```
void loop() {
    BevaegelsesSensor.detectMotion();
}
```

Hertil kan vi se, at vores detectMotion()-metode kaldes. Denne metode tilhører vores objekt ”BevægelsesSensor”, som samtidig er en instans af selve klassen ”BevægelsesSensor”.

Selv funktionen fungerer på den måde, at vores loop-funktion konstant udfører målinger af vores bevægelsessensor, ved at kalde detectMotion()-metoden på vores ”BevægelsesSensor”-objekt inde i loopen.

Nedenfor ses vores implementering af vores loop-funktion i cpp-filen:

```
void BevaegelsesSensor::detectMotion()
{
    int bevægelseValue = digitalRead(pirPin_);

    if (bevægelseValue == HIGH)
    {
        digitalWrite(gpioPin_, HIGH);
        Serial.println("Motion detected!");
        delay(0);
    }
    else
    {
        digitalWrite(gpioPin_, LOW);
    }
}
```

Implementeringen omfatter de linjer kode som benytter sig af vores detectMotion-metode, og dertil de tilføjede funktioner. Disse linjer kode omfatter:

1. ***"void BevaegelsesSensor::detectMotion()"***: Denne del definerer detectMotion()-metoden i klassen "BevaegelsesSensor".
2. ***int bevaegelseValue = digitalRead(pirPin_)***: Denne del læser output-værdien fra vores bevægelsessensor via. pirPin_-benet, og gemmes i variablen "bevaegelsesValue".
3. ***"if (bevaegelseValue == HIGH)"***: Denne del tjekker om bevægelsessensoren har registreret bevægelse. Hvis dette er tilfældet, udføres:
 - ➔ ***"digitalWrite(gpioPin_, HIGH)"***: som sætter gpioPin_-benet til logisk høj (HIGH), og sender dermed et signal til din Raspberry Pi om, at bevægelse er blevet registreret.
 - ➔ ***"delay(5000)"***: Dette delay er baseret på hvor længe vores sensor forbliver i sit høje stadie. Dog er dette baseret på, om sensoren er sat til retrigger- eller single-trigger-mode.

Temperatursensor (Hafsa og Louise):

Temperatursensoren spiller en central rolle i vores hjemmesikkerhedssystem. Den er ansvarlig for at fortage temperaturmålinger og at tage hensigtsmæssige beslutninger baseret på disse målinger. I denne sektion fokuserer vi nærmere på temperatursensorens software-design, udforsker de funktioner, der er implementeret, og undersøger, hvordan de samlet set bidrager til systemets overordnede funktionalitet

Kommunikation og I2C:

Temperatursensoren bruger I2C-protokollen til at kommunikere med Raspberry Pi'en. Viden om I2C-protokollen er opnået gennem et af semestrets kurser kaldet ”Grænseflader Til Den Fysiske Verden”. I2C-protokollen er tildelt en bestemt I2C-adresse, nemlig 0x4F. Gennem denne adresse læser sensoren data og konverterer dem til en temperatur i grader Celsius. Temperaturen opdateres i realtid i variablen `temp_out`. Dette kommunikationssystem muliggør ikke kun effektiv dataoverførsel, men garanterer også præcis temperaturmålinger

Initialisering af sensoren:

Ved opstart konfigurerer vi de nødvendige indstillinger for at etablere en stabil forbindelse mellem temperatursensoren og Raspberry Pi'en. Dette inkluderer at specificere I2C-adressen, initialisere temperaturopoutput (`temp_out`), og oprette forbindelse til I2C-bussen.

```
class TemperatureSensor:  
    def __init__(self, i2c_address=0x4F):
```

Læsning af temperatur:

Læsning af temperatur spiller en central rolle i sensorens funktion. Metoden ”temp_without_delay” udfører læsning af temperaturdata fra sensoren ved at bruge I2C-kommunikation. Den bearbejder dataene for at beregne temperaturen og opdaterer `temp_out` med den aktuelle måling. Denne løbende opdatering, sikrer at sensoren mäter de seneste temperaturmålinger. Ved læsning af temperatur, bruger vi MSB (Most Significant Bit) og LSB (Least Significant Bit) til at tolke de digitale data fra sensoren.

```
def temp_without_delay(self):  
    temperature_data = self.bus.read_i2c_block_data(self.i2c_address, 0, 2)
```

```
    temperature = ((msb << 8) | lsb) / 256.0  
    self.temp_out = temperature
```

Derudover, laver metoden ”temp_with_delay” en forsinkelse, før den læser temperaturen.

Forsinkelsen afhænger på den læste temperatur. Hvis temperaturen er over 28 grader Celsius, er der en forsinkelse på 30 sekunder; ellers er der en forsinkelse på 1 sekund. Samlet set sikrer læsningsfasen, at temperatursensoren leverer pålidelige og aktuelle data.

```
def temp_with_delay(self):
    if self.temp_out > 28.0:
        delay = 30 # 30 sekunder forsinkelse hvis temperaturen er under 28 grader
    else:
        delay = 1 # hvis ikke, så er der 1 sekund forsinkelse
```

Visning af temperatur og advarsler:

Der bruges ”print_temperature” metoden for at vise den aktuelle temperatur, så brugeren kan let følge med temperaturen. Samtidig bruger vi ”send_besked” metoden tjekker, om temperaturen overstiger 28 grader Celsius, og returnerer en advarsel, hvis dette er tilfældet. Denne kombination af temperaturvisning og advarsel sikrer, at brugeren er opmærksom på eventuelle uønskede temperaturforhold i systemet.

```
def send_besked(self):
    if self.temp_out > 28.0:
        return "High temperature alert!"
    else:
        return ""
```

Hovedløkkken:

I hovedløkkken samles alle dele af temperatursensoren. Den kører konstant og indeholder læsning af temperatur, visning af temperaturen, og tjek for advarsler. Hovedløkkken sikrer, at temperaturen opdateres løbende, og brugeren bliver informeret om ændringer i temperaturen. Dette softwaredesign bidrager til pålidelige overvågnings af temperaturen i hjemmesikkerhedssystemet.

Kamera (August):

Kameraet har to hovedfunktioner i koden. Nemlig optagelse af video og livestreaming. De to skal fungere individuelt. Video-funktionen skal fungere i samarbejde med bevægelsessensor, sådan at en 10-sekunders video gemmes på Raspberry Pi'en når bevægelse opfanges.

Streaming-funktionen skal give in konstant strøm af en MJPG-fil til GUI 'en som herefter står for at streame den til hjemmesiden.

Imports og initialisering:

```
import io
import picamera
import os
import threading
camera_manager = CameraManager()
```

Her importeres de vigtigste biblioteker til både streaming og videooptagelse med kamera.

”picamera” sørger for alt kamerafunktion direkte. Normalt kan kameraet kun bruges med en linux bash-kommando. Men med dette bibliotek kan man direkte tilgå kameraet fra python-kode.

”io” sørger for input og output funktioner. I forhold til kamera-koden bruges den kun som en buffer til at indeholde det streamede billede.

”os” sørger for at Python koden direkte kan interagere med styresystemet på Raspberry pi'en. Dette bruges til optagelse af video, sådan at videoerne kan gemmes på enheden.

”threading” bruges til at informere andre klasser i koden om at en ny frame er tilgængelig. Dette bruges til video streaming.

Et objekt af klassen ”CameraManager” laves som initialisering af kamera.

Streamingoutput class:

```
class StreamingOutput(object):
    def __init__(self, camera):
        self.frame = None
        self.buffer = io.BytesIO()
        self.condition = Condition()
        self.camera = camera

    def write(self, buf):
        if buf.startswith(b'\xff\xd8'):
            self.buffer.truncate()
            with self.condition:
                self.frame = self.buffer.getvalue()
                self.condition.notify_all()
            self.buffer.seek(0)
        return self.buffer.write(buf)
```

Denne class er designet til at håndtere video frames til streaming. Hvis en ny frame detekteres med typen .jpg, trunkerer den bufferen (Her io.BytesIO() bufferen), opdaterer ”frame” variablen med indholdet af bufferen og giver besked til andre classes ved hjælp af condition.notify_all() metoden.

Denne klasse bruges sammen med kameraets videooptagelses- og streamingfunktionalitet for at give en kontinuerlig strøm af video-frames. Betingelsesvariablen hjælper med at synkronisere adgang til rammevariablen mellem forskellige tråde.

CameraManager class:

```
class CameraManager:  
    def __init__(self):  
        self.camera = picamera.PiCamera(resolution='640x480', framerate=24)  
        self.camera.rotation = 180  
        self.output = StreamingOutput(self.camera)  
        self.recording = False  
        self.lock = threading.Lock()  
  
    def start_streaming(self):  
        with self.lock:  
            if not self.recording:  
                self.camera.start_recording(self.output, format='mjpeg')  
  
    def stop_streaming(self):  
        with self.lock:  
            if not self.recording:  
                self.camera.stop_recording()  
  
    def record_video(self, filename, duration):  
        with self.lock:  
            self.recording = True  
            self.camera.stop_recording()  
            self.camera.start_recording(filename)  
            time.sleep(duration)  
            self.camera.stop_recording()  
            self.camera.start_recording(self.output, format='mjpeg')  
            self.recording = False
```

Denne class sørger for initialisering af et kamera-objekt. I klassen sættes oplosning, framerate, rotation på kamera. Et objekt af klassen ”StreamingOutput” laves også inden i denne class. Herefter sættes et boolean-flag som hedder ”recording” til false, for at kunne holde styr på om der optages. Threading.Lock() bruges til at holde styr på på instanser sådan at der ikke opstår konflikt imellem dem. Herefter defineres 3 metoder. ”start_streaming”, ”stop_streaming” og ”record_video”.

”start_streaming” er en metode som starter streamen af mjpeg fra kameraet. For at kunne gøre dette bruges objektet af klassen ”StreamingOutput”. Denne funktion kan kun bruges når der ikke optages video, ellers gør den intet.

”stop_streaming” er en metode som også kun kan bruges ved at der ikke optages video. Her stoppes blot med at optage med kameraet, hvilket stopper streamen. Denne funktion bruges kun som oprydning når programmet terminernes.

”record_video” er en metode som optager en video og gemmer den til raspberry pien. Metoden tager 2 argumenter, ”duration” som sætter længden på den optagte video og ”filename” som giver videon sit navn. Metoden starter med at sætte flaget ”recording” til true. Herefter stopper den streamen ved at stoppe for kameraets optagelse.

En kommando til optagelse af en ny video sættes i gang, med en sleep-timer på 10-sekunder indtil optagelsen stopper igen. Streamen startes nu igen ligesom i klassen "start_streaming" og flaget sættes til false igen.

GUI (August):

GUI'en er en stor del af koden, da den består af en webserver, brugergrænseflade samt funktioner til integration af klasserne fra de forskellige komponenter. Ved brug af ngrok (Ngrok, 2024) kan vi hoste en HTML/JSON-hjemmeside til internettet som kan tilgås fra alle steder i verden, hvor man har adgang til nettet.

Imports og initialisering:

```
import time
import logging
import socketserver
import subprocess
import json
from http import server
from pyngrok import ngrok
from datetime import datetime
```

”Time og datetime” giver forskellige tidsrelaterede funktioner. Det bruges til at tilføje forsinkelser og arbejde med tidsstempler som tilføjes til de optagte videoer.

”logging” giver mulighed for udsendelse af logmeddelelser fra Python-programmer. Det giver også mulighed for at konfigurere forskellige logniveauer og destinationer. Dette blev mest brugt til at fejlfinde under udvikling.

”Socketserver” bruges til oprettelse af en netværksserver. Det forenkler processen med at skabe servere i Python-dokumenter. Det bruges her til at skabe den server som hostes med ngrok (Ngrok, 2024).

”Subprocess” kan køre eksterne kommandoer fra et Python-script. Det bruges til at ændre video-formatet fra .h264 til .mp4 på den video som gemmes ved bevægelse registreret.

”json” bruges til at kunne skrive json i koden til hjemmesiden. Dette giver mere funktionalitet end blot HTML.

”http.server” inkluderer moduler til at arbejde med HTTP (Hypertext Transfer Protocol). Det giver grundlæggende HTTP-serverfunktionalitet og bruges til at kunne hoste serveren.

”pyngrok” er en tjeneste, der skaber sikre tunneler til localhost. Det bruges til at åbne lokale servere for internettet. I koden bruges ngrok til at oprette en offentlig URL til den lokale webserver.

```
port = 8000
public_url = ngrok.connect(port)
print("* ngrok tunnel \"{}\" -> \"http://127.0.0.1:{}\"".format(public_url, port))
```

Her vælges en port som åbnes via ngrok til det offentlige internet. I koden bruges der port 8000. Herefter skabes en url med ngrok som printes til terminalen.

HTML/JSON:

```
PAGE = """
<html>
<head>
    <title>Raspberry Pi - Surveillance Camera</title>
    <script>
        function refreshMotionStatus() {
            fetch('/motion-status')
                .then(response => response.text())
                .then(data => {
                    document.getElementById('motionStatus').innerHTML =
data;
                });
        }

        function refreshTemperature() {
            fetch('/temperature')
                .then(response => response.text())
                .then(data => {
                    document.getElementById('temperature').innerHTML =
data;
                });
        }

        function refreshVideoList() {
            fetch('/video-list')
                .then(response => response.json())
                .then(data => {
                    const videoSelect = document.getElementById('videoSelect');
                    videoSelect.innerHTML = '<option value="" disabled selected>Select Video</option>';
                    data.forEach(video => {
                        const option = document.createElement('option');
                        option.value = video;
                        option.text = video;
                        videoSelect.add(option);
                    });
                });
        }

        function playSelectedVideo() {
            const selectedVideo = document.getElementById('videoSelect').value;
            const videoPlayer = document.getElementById('videoPlayer');
            if (selectedVideo) {
                // Pause the video before changing the source
                videoPlayer.pause();

                // Provide the correct path or URL to the selected video
                videoPlayer.src = selectedVideo;
            }
        }
    </script>
</head>
<body>
    <div id='motionStatus'></div>
    <div id='temperature'></div>
    <div id='videoSelect'></div>
    <div id='videoPlayer'></div>
</body>
</html>
```

```
        videoPlayer.load();
        videoPlayer.play();
    }

}

setInterval(refreshMotionStatus, 2000);
setInterval(refreshTemperature, 5000);
setInterval(refreshVideoList, 5000);
</script>
</head>
<body>
    <center><h1>Raspberry Pi - Home Security - Group 5 Project</h1></center>
    <center></center>
    <center><h2 id="motionStatus">Checking motion...</h2></center>
    <center><h2 id="temperature">Checking temperature...</h2></center>

    <!-- Display a dropdown menu of available .mp4 files -->
    <center><select id="videoSelect" onchange="playSelectedVideo()">
        <option value="" disabled selected>Select Video</option>
        <!-- Options will be dynamically populated using JavaScript -->
    </select>

    <!-- Add a video player element -->
    <center><video id="videoPlayer" controls width="640" height="480">
        Your browser does not support the video tag.
    </video>
</body>
</html>
"""
```

Denne kode skrevet I HTML/JSON giver alt funktionalitet og udseende til hjemmesiden. Vi har et par funktioner som sørger for at opdatere både temperatur-status (refreshTemperature()), bevægelses-status (refreshMotionStatus()) samt listen af videoer (refreshVideoList()). Disse opdateres med et interval defineret med ”setInterval”.

Funktionen ”playSelectedVideo” sørger for at den valgte video fra drop-down menuen bliver valgt som den video der spilles is video-playeren.

Resten er ”body” hvilket blot er alle de elementer som hjemmesiden består af. Dette er bl.a. Videoafspiller, en Box til stream, en drop-down menu samt info om temperatur og bevægelse.

StreamingHandler class:

```
class StreamingHandler(server.BaseHTTPRequestHandler):
    def do_GET(self):
        global motion_detected
        if self.path == '/motion-status':
            self.send_response(200)
            self.send_header('Content-Type', 'text/plain')
            self.end_headers()
            message = "Motion detected! - Video recording" if motion_detected else "No motion"
            self.wfile.write(message.encode('utf-8'))
        elif self.path == '/temperature':
            self.send_response(200)
            self.send_header('Content-Type', 'text/plain')
            self.end_headers()
            temperature = temperature_sensor.temp_without_delay()
            if temperature>28:
                self.wfile.write(f"temperature:{.2f}°C HIGH TEMP ALERT".encode('utf-8'))
            else:
                self.wfile.write(f"temperature:{.2f}°C".encode('utf-8'))
        elif self.path == '/':
            self.send_response(301)
            self.send_header('Location', '/index.html')
            self.end_headers()
        elif self.path == '/index.html':
            content = PAGE.encode('utf-8')
            self.send_response(200)
            self.send_header('Content-Type', 'text/html')
            self.send_header('Content-Length', len(content))
            self.end_headers()
            self.wfile.write(content)
        elif self.path.endswith('.mp4'):
            video_path = self.path[1:] # Remove the leading slash
            with open(video_path, 'rb') as video_file:
                self.send_response(200)
                self.send_header('Content-Type', 'video/mp4')
                self.end_headers()
                self.wfile.write(video_file.read())
        elif self.path == '/video-list':
            self.send_response(200)
            self.send_header('Content-Type', 'application/json')
            self.end_headers()

            # List all .mp4 files in the project folder
            video_files = [f for f in os.listdir() if f.endswith(".mp4")]
            video_files_json = json.dumps(video_files)
            self.wfile.write(video_files_json.encode('utf-8'))
        elif self.path == '/stream.mjpg':
            self.send_response(200)
            self.send_header('Age', 0)
            self.send_header('Cache-Control', 'no-cache, private')
            self.send_header('Pragma', 'no-cache')
            self.send_header('Content-Type', 'multipart/x-mixed-replace;
boundary=FRAME')
            self.end_headers()
            camera_manager.start_streaming()
            try:
                while True:
                    with camera_manager.output.condition:
```

```
        camera_manager.output.condition.wait()
        frame = camera_manager.output.frame
        self.wfile.write(b"--FRAME\r\n")
        self.send_header('Content-Type', 'image/jpeg')
        self.send_header('Content-Length', len(frame))
        self.end_headers()
        self.wfile.write(frame)
        self.wfile.write(b'\r\n')
    except Exception as e:
        logging.warning(
            'Removed streaming client %s: %s',
            self.client_address, str(e))
    finally:
        camera_manager.stop_streaming()
else:
    self.send_error(404)
    self.end_headers()
```

Denne klasse star overordnet for at håndtere alt data der opstår/opdateres imens programmet kører. Den håndterer så den data sådan at den reageres eller opdateres på hjemmesiden.

- **/motion-status:**

Hvis den anmodede data er /motion-status, sender den et svar med en statuskode på 200 (OK).

Den sender en almindelig tekstbesked, der angiver, om der registreres bevægelse eller ej.

- **/temperatur:**

Hvis den anmodede data er /temperatur, sender den et svar med en statuskode på 200 (OK).

Den sender den aktuelle temperatur i svaret, og hvis temperaturen er større end 28, inkluderer den en højtemperaturalarm.

- **/ :**

Hvis den anmodede sti er /, sender den et omdirigeringsvar (statuskode 301) til /index.html.

- **/index.html:**

Hvis den anmodede sti er /index.html, sender den et svar med en statuskode på 200 (OK).

Den sender HTML-indholdet af PAGE-variablen.

- **.mp4:**

Hvis den anmodede data ender med .mp4, antages det, at det er en videofil.

Den læser videofilens indhold og sender det som et svar med en indholdstype video.mp4.

- **/video-liste:**

Hvis den anmodede sti er /video-list, sender den et svar med en statuskode på 200 (OK).

Den sender en JSON-kodet liste over alle .mp4-filer i projektmappen.

- **/stream.mjpg:**

Hvis den anmodede sti er /stream.mjpg, sender den et svar med statuskode 200 (OK) og de headere, der er nødvendige for at levere MJPEG-indhold.

Den starter streaming af billeder fra kameraet ved hjælp af camera_manager.start_streaming() metoden.

Den går ind i en loop, hvor den konstant venter på nye frames og sender dem som MJPEG-indhold til klienten.

- **Andet:**

Hvis den anmodede sti ikke matcher nogen af de foruddefinerede stier, sender den et svar med statuskode 404 (Ikke fundet).

StreamingServer class:

```
class StreamingServer(socketserver.ThreadingMixIn, server.HTTPServer):  
    allow_reuse_address = True  
    daemon_threads = True
```

StreamingServer-klassen er konfigureret til at bruge threading til håndtering af anmodninger, så den kan håndtere flere anmodninger samtidigt. Derudover er det sat op til at tillade genbrug af serveradressen og til at afslutte når hovedprogrammet er færdigt, selvom der er aktive tråde. Disse konfigurationer er nyttige til at køre serveren i baggrunden, håndtere anmodninger effektivt og lette nem genstart af serveren.

HW Design

Temperatursensor (Hafsa og Louise)

I denne del udforsker vi hardware-designet af temperatursensoren for at forstå, hvordan den er integreret i vores system.

Valg af Temperatursensor:

Vi har valgt at anvende en LM75 temperatursensor til at måle temperaturen i vores system.

LM75 er kendt for sin nøjagtighed og pålidelighed og tilbyder en digital udgang, hvilket gør den ideel til vores system.

Tilslutning til Raspberry Pi:

For at oprette forbindelse mellem temperatursensoren og vores Raspberry Pi bruger vi I2C (Inter-Integrated Circuit) -kommunikationsprotokollen. Dette giver os mulighed for hurtig og pålidelig dataoverførsel mellem sensoren og kontrolenheden.

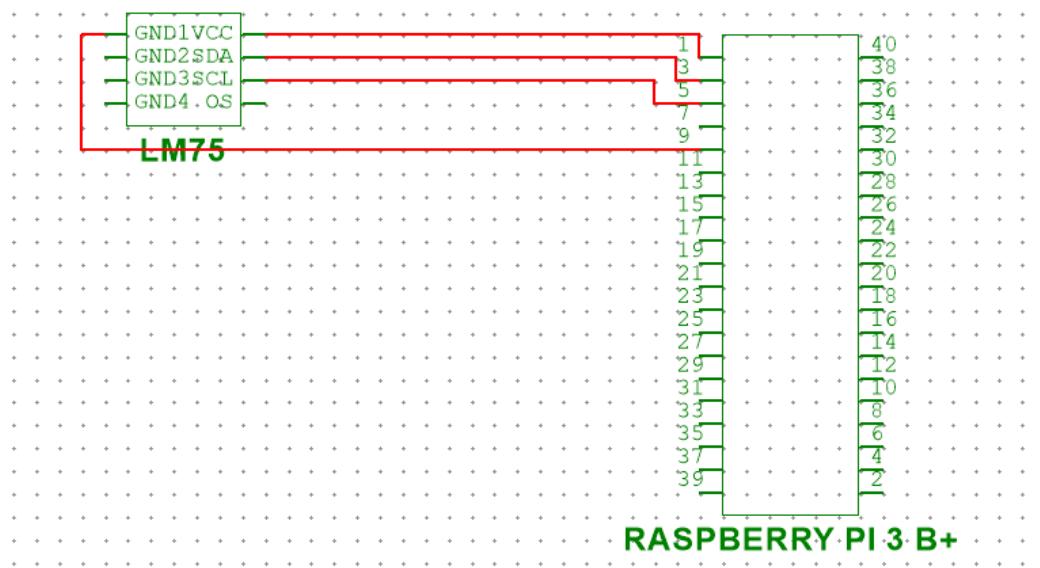
Pin-definitioner:

- SDA (Serial Data): Dette er dataforbindelsen mellem sensoren og Raspberry Pi. Den bruges til at sende og modtage temperaturdata.
- SCL (Serial Clock): Dette er clock-forbindelsen, der synkroniserer dataoverførslen mellem sensoren og Raspberry Pi.
- GND (Ground): Der er ground på Sensoren og Raspberry Pi'en.
- VCC: Strøm fra Raspberry Pi som sensoren bruger (3.3)

Når vi tilslutter sensoren til Raspberry Pi'en, følger vi den almindelige tilslutning for I2C-kommunikation mellem en sensor og en Raspberry Pi. Raspberry Pi modellen som vi bruger, er Raspberry Pi 3 B+.

VCC på temperatursensor forbindes med VCC (3.3V) på Raspberry Pi som er på pin 1.

Ground på Temperatursensoren forbindes til Ground på Raspberry Pi, som er på pin 9. SDA (Serial Data) på temperatursensoren forbindes til SDA-pin på Raspberry Pi, på pin 3. Til sidst forbinder vi SCL (Serial Clock) på temperatursensoren til SCL-pin på Raspberry Pi, som er på pin 5. Udover det, vi har også sørget for at vores Raspberry Pi have de nødvendige biblioteker og indstillinger konfigureret for at kunne kommunikere med vores temperatursensor.



Figur 24 multisim diagram af vores opsætning

Digital udgang:

En fordel ved LM75-sensoren er dens digitale udgang, der eliminerer behovet for analoge konverteringer. LM75-sensoren er i stand til at måle temperaturer med en nøjagtighed på op til ± 2 grader Celsius. Dette gør det muligt for vores software at læse præcise temperaturdataene direkte fra sensoren.

Effektiv strømstyring:

LM75-sensoren har et lavt strømforbrug, hvilket er vigtigt for vores system. Det kræver en stabil strømforsyning for pålidelig drift.

Konklusion:

LM75 temperatursensoren er en central del af vores hjemmesikkerhedssystem og sikrer, at vi kan måle temperaturen på en nøjagtig måde. Dens integration med Raspberry Pi og det softwaremæssige design giver os mulighed for effektivt at overvåge og reagere på temperaturændringer, hvilket er vigtigt for systemets succes.

Kamera (August)

Valg af kamera:

Raspberry pi cam 2 kameraet er et godt valg til vores projekt da den er nem at tilslutte, har god understøttelse i både C++ og Python, det giver god kvalitets video og bruger ikke alt for meget strøm. Alt sammen i en kompakt komponent.

Tilslutning til Raspberry Pi:

Et Raspberry pi kamera er heldigvis lavet til vores hardware, og derfor bruges der kun 1 kabel til både strømforsyning samt dataoverførsel imellem kameraet og Raspberry pi'en. Denne forbindelse kaldes CSI (Camera Serial Interface). Kablet tilsluttes direkte imellem kameraet og raspberry pi'en i dens CSI-connector.

Pin-definitioner:

- CSI (Camera Serial Interface): Både data til og fra kameraet løber her. Udover det gives der også strøm fra Raspberry pi'en til kameraet i dette kabel, sådan at det kan bruges.

Video:

Raspberry pi camera 2 understøtter en video-opløsning på op til 1080x1920p med en FPS på 47. Altså kan det optage video der er 1080 pixels gange 1920 pixels som opdateres med nye billede 47 gange i sekundet. Dette er en vigtig detalje, da et normalt kamera som blot kan tage billeder, ikke er nok til dette projekt. Den høje opløsning samt billedhastighed giver også sine fordele i form af klarhed og detaljer. Flere billeder med flere oplysninger er altid et plus, så længe at det kan lade sig gøre og giver mening.

Software:

Kameraet har en masse indbygget kommandoer som kan bruges direkte med bash-kommandoer fra Raspberry pi'ens terminal. Dog har vi et mere avanceret mål i tankerne, så det er også vigtigt med understøttelse af hardwaren i softwaren. Her er kameraet også rigtig godt, da det har en stor brugerbase, og derfor også stor understøttelse i software. Både i Python og C++. Dette giver god fleksibilitet samt nem udnyttelse af kameraets funktioner.

Konklusion:

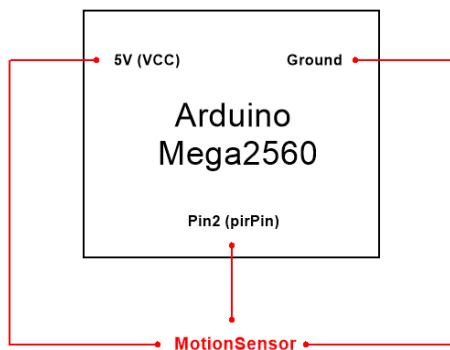
Raspberry pi kameraet er en perfekt løsning til vores projekt, med en nem forbindelse, mange funktioner og god understøttelse.

Bevægelsessensor: (August)

Valg af sensor:

PIR Infrared Sensor er et oplagt valg til et securitysystem, da det registrer infrarødt lys som udskilles fra bl.a. mennesker. Den kan få alt nødvendig strøm fra en Arduino(5V) og outputter nok spænding til registrering med Arduino(3,3V).

Tilslutning til Arduino:



Figur 25 - forbindelse mellem bevægelsessensor og Arduino

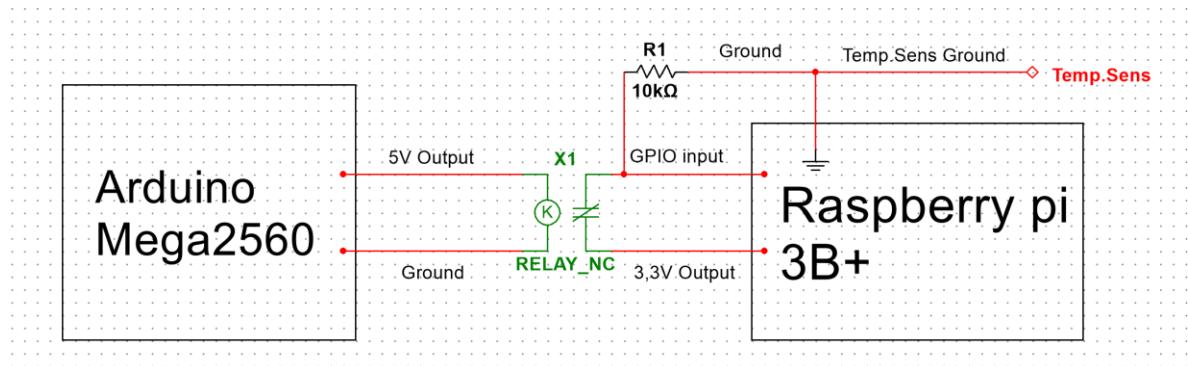
Sensoren har 3 pins: VCC, GND som giver sensoren strøm, og outputtet som går høj når bevægelse/infrarødt lys opfanges. Arduinoen giver os mulighed for, igennem kode, at indstille på hvordan sensoren skal opføre sig før signalet sendes til Raspberry pi'en.

Pin-definitioner:

- VCC: Strøm fra Arduinoen som sensoren bruger. 5V
- GND: Ground fra Arduinoen. 0V
- GPIO: Outputtet fra sensoren går høj (3,3V) når bevægelse/infrarødt lys opfanges, ellers er den lav (0V). Dette registreres af en af Arduinoens GPIO pins.

Tilslutning til Raspberry pi:

Arduinoen giver et output som styres med kode. Dette output åbner eller lukker for et relæ, som forbinder Raspberry pi'en med sin egen spænding. Dette system sørger for at de 5V som outputtes fra Arduinoen ikke går direkte på Raspberry pi'en som opererer på 3,3V. Hertil er der tilføjet en pull-down modstand og en ekstra ground til tempuratursensoren. Dette kører på sin egen printplade.



Figur 26: Diagram af relæ-opsætning

Pin-definitioner:

- 5V Output: Output fra en af Arduinoens GPIO pins. Outputter 5V når bevægelse opfanges
- Ground (Arduino): Ground fra Arduinoen.
- 3,3V Ouput: Raspberry pi'ens 3,3V pin
- GPIO-input: En af Raspberry pi'ens GPIO pins som registrerer de 3,3V
- Ground (Raspberry pi)/Temp.Sens Ground: Ground fra Raspberry pi. Bruges både med pull-down modstanden for at sikre en constant LOW-state indtil andet opfanges samt en ekstra ground pin til brug af temperatursensoren.

Manuel indstilling:

En vigtig funktion på PIR-sensoren er de to knapper der kan drejes på selve hardwaremodulet. Disse bruges til at indstille følsomhed og tid. Specielt er følsomhedsindstillingen vigtig, da dette ikke kan ændres i software. Den skal gerne indstilles sådan at den passer med den situation/det rum den sættes i. Sensoren burde gerne KUN aktiveres når et menneske går forbi.

Konklusion:

PIR-sensoren tilsluttes altså først en Arduino som igennem et relæ sender info til Raspberry pi'en. Dette er en god løsning for vores projekt, da vi har gode muligheder for at indstille på en meget simpel sensor, sådan at den opfører sig perfekt i forhold til vores behov.

Udførelse af accepttest

I dette afsnit udføres der accepttest af både accepttestspezifikationerne for de funktionelle og ikke-funktionelle krav. Disse tests kan ses at være udfyldte i de forskellige tabeller på de kommende sider.

Accepttestudførelse for de funktionelle krav

Use Case 1: Manuelt tjek af systemets komponenter (Mads og August)

Use case under test		Manuelt tjek af systemets komponenter		
Scenarie		Hovedscenariet		
Prækondition		Alle komponenter i smart-hjemmesikkerhedssystemet er tilsluttet strøm, og har forbindelse til brugergrænsefladen. Computeren er tilsluttet internettet.		
Step	Handling	Forventet obser-vation/resultat	Faktisk obser-vation/resultat	Vurdering (OK/FAIL)
1	Brugeren åbner for brugergrænsefladen (GUI) ved hjælp af computer.	Brugergrænsefladen (GUI) indlæses på skærmen og vises	Vi kan frit tilgå Url'en til GUI og den indlæses korrekt	Godkendt.
2	Temperatursensoren afmåler temperaturen på daværende tidspunkt, og returnerer temperaturmålingen til brugeren gennem en streng.	Er der under 28 grader celsius vises det, og over 28 grader celsius sendes der en ''high temp alert''.	Vi ser en tekststreng	Godkendt.
3	Brugeren aflæser temperaturmålingen på brugergrænsefladen.	Temperaturmålingen vises klart og tydeligt på GUI og aflæses nemt at brugeren.		Godkendt.
4	Bevægelsessensoren frakobles, og brugeren tjekker ved at manuelt foretage en bevægelse foran	Ingen resultater vises på brugergrænsefladen (GUI)		Godkendt.

	sensoren, at intet bliver opfanget af sensoren.		
5	Bevægelsessensoren tilkobles, og brugeren tjekker ved at foretage en bevægelse foran sensoren, at der opfanges bevægelse.	Brugergrænsefladen (GUI) viser en besked med <i>''Motion detected''</i>	Godkendt.
6	Når bevægelse opfanges af sensoren, bliver en streng returneret til brugergrænsefladen, som fortæller brugeren at bevægelse er blevet opfanget, og kamerasensoren aktiveres i 10 sekunder.	Kameraet begynder automatisk at optage det der streames, og en <i>''No motion''</i> string printes til GUI.	Godkendt.
7	Brugeren aflæser strenge, som fortæller ham/hende, at bevægelse er blevet opfanget, og observerer live videooptagelse, som kamerasensen fremviser. Igennem brugergrænsefladen kan brugeren vælge de optagede videoer.	Brugergrænsefladen (GUI) viser video fra kamera og brugeren har mulighed for at vælge at afspille tidligere optagede videoer i en dropdown menu.	Godkendt.
8	Efter 10 sekunder fjernes strengen fra brugergrænsefladen, som fortæller brugeren, at bevægelse er blevet opfanget.	Brugergrænsefladen (GUI) viser efter 10 sekunder ikke længere <i>''Motion detected''</i> men returnere i stedet til <i>''No motion''</i>	Godkendt.

Tabel 13: Accepttestudførelse på Use case 1: Manuelt tjek af systemets komponenter

Use case 2: Forhøjet temperatur opfanges (Hafsa)

Use case under test		Forhøjet temperatur opfanges		
Scenarie		Hovedscenariet		
Prækondition		Temperaturssensoren er tilsluttet strøm, og har forbindelse til brugergrænsefladen. Målingerne er grøn.		
Step	Handling	Forventet observa-tion/resultat	Faktisk observa-tion/resultat	Vurdering (OK/FAIL)
1	Brugeren tænder temperatursensoren og modtager en ny måling hver 30 sekund. (antaget stuetemperatur)	Temperaturen bliver printet på terminalen hver 30 sekunder, så længe den er under 28 grader.	Vi ser at stuetemperaturen bliver printet på terminalen, med et 30 sekunds interval.	Godkendt.
2	Brugeren holder et varmeforhøjende element foran temperatursensoren.	Temperaturen stiger til 28 + grader og vi indgår 'High alert mode'	Vi holdt en finger på temperaturesensenoren, og modtog en 'high temperature alert' besked på terminalen. Temperaturen opdateres hver sekund.	Godkendt.
3	Temperatursensoren mäter temperaturen over 28 grader Celsius og returnerer temperaturmålingen til brugeren gennem en tekst med en alarmerende besked.	Brugergrænsefladen (GUI) sender en alarmerende tekst ud til brugeren.	Vi ser at der printes den reelle temperatur samt ''High temp alert''	Godkendt.
4	Temperatursensoren fortsætter med at aflæse og returnere temperaturmålingen til brugeren i 1 sekunds intervaller, så	Brugergrænsefladen (GUI) opdaterer temperaturen hvert sekund	Vi ser at der printes en ny opdateret temperatur med 1	Godkendt.

	længe temperaturen er over 28 grader Celsius.	så længe temperaturen er over 28 grader.	sekunds intervaler, fremfor 2 sekunders intervaller da temperaturen var under 28 grader.	
5	Brugeren aflæser beskedten om temperaturmålingen på brugergrænsefladen.	Temperaturen vises og kan aflæses i brugergrænsefladen (GUI).	Vi kan til enhver tid gå ind og se hvad temperaturen er og se om der er en alarmerende mængde varme.	Godkendt.
6	Brugeren afventer, indtil temperaturen falder under 28 grader Celsius.	Brugergrænsefladen (GUI) opdaterer målingen hvert sekund, så længe temperaturen er over 28 grader, og hvert andet når den er under.	Brugeren ser at alarmen forsvinder når temperaturen går under 28 grader og der bliver opdateret med længere intervaller.	Godkendt.
7	Brugeren observerer, at beskedten om temperaturmålingen forsvinder fra brugergrænsefladen.	Brugergrænsefladen (GUI) vender tilbage til normalen og opdaterer nu hvert 2. sekund.	Vi ser en konstant opdatering af temperaturen hver 2. sekund og ingen alarmgivende beskeder længere.	Godkendt.

Tabel 14: Accepttestudførelse på Use case 2: Forhøjet temperatur opfanges

Use case 3: Bevægelse opfanges (Altaf og Ruben)

Use case under test	Bevægelse opfanges
Scenarie	Hovedscenariet
Prækondition	Kamerasensoren og bevægelsessensoren er tilsluttet strøm, har forbindelse mellem hinanden, og er tilkoblet gennem brugergrænsefladen.

Step	Handling	Forventet observation/resultat	Faktisk observation/resultat	Vurdering (OK/FAIL)
1	Bevægelsessensoren starter i sin default-stadie, hvor den er klar til at detektere bevægelse.	Brugergrænsefladen viser, at systemet er aktivt og i standby tilstand, klar til at reagere på eventuel bevægelse.	Start med "No motion!"	Godkendt.
2	Der bliver registreret en bevægelse af bevægelsessensoren.	Brugergrænsefladen viser at en bevægelse er blevet detekteret, med en besked eller en indikator for at signalere dette.	Der er observeret at sensoren detekterer bevægelse, da tekstrængten på GUI ændrer sig til " <i>Motion detected</i> "	Godkendt.
3	Bevægelsessensoren opfanger bevægelsen, og starter en timer på 10 sekunder.	Sensoren har startet en timer internt, der gør den sender et højt output i 10 sekunder.	Vi tjekker efter med en timer. I tilfælde af upræcision eller drift, kan sensorens potentiometer justeres, så dens high-output varer i 10 sekunder +/-	Godkendt.
4	Bevægelsessensoren returnerer en streng til brugergrænsefladen samtidigt med, at den sender et signal til kamerasensoren om at den skal aktiveres.	Kamerasensoren aktiveres og begynder at optage i 10 sekunder.	Vi placerer en hånd foran sensoren, hører et klik fra relæet og observerer teksten på GUI ændre sig fra " <i>No motion</i> " til " <i>Motion detected</i> "	Godkendt.

5	Brugeren kan nu observere gennem brugergrænsefladen - via en streng - at der har været opfanget bevægelse på sensoren.	Brugergrænsefladen viser en besked eller en streng, der informerer brugeren om, at der er blevet opfanget bevægelse på sensoren.	Observeret, som beskrevet i punkt 4.	Godkendt.
6	Efter 10 sekunder returnerer den nu et low-output, strengen fjernes, og bevægelsessensoren returnerer til sin default-stadie.	Brugergrænsefladen (GUI) vender tilbage til normalen og systemet er nu i en ventetilstand og afventer at ny bevægelse bliver detekteret.	Vi observerer teksten på GUI ændre sig fra ''Motion detected'' til ''No motion''	Godkendt.
7	Der gemmes en optagelse fra da bevægelse blev opfanget, og 10 sekunder frem.	Brugergrænsefladen har en menu hvori samtlige video-filer kan tilgås, brugeren kan nu vælge at afspille den nyeste video og se hvilken bevægelse sensoren opfangede.	Vi trykker på dropdown menuen og vælger den øverste video. Herefter afspilles videoen i GUI'en.	Godkendt.

Tabel 15: Accepttestudførelse på Use case 3: Bevægelse opfanges.

Accepttestudførelse for de ikke-funktionelle krav

Temperatursensor (Hafsa)

Krav nr.	Krav	Test	Faktisk observa-tion/resultat	God-kendt (JA/NEJ)
Fysiske di-mensioner 1.1	Temperatursensoren skal have dimensioner på max $8 \times 8 (\pm 2)$ cm.	Måletest: Temperatursensoren måles, og dimensionerne viser til at være på $18 \times 8 \times 2 (\pm 2)$ cm.	2,29 cm bredde & 4,40 cm længde	Godkendt
GUI: 2.1	Temperaturmålingerne skal kunne aflæses på GUI'en	Visuel test: Temperaturmålingerne vises og aflæses på GUI'en	Der bliver regelmæssigt, med enten 2 eller 1 sekunds intervaller, opdateret en temperaturstatus på GUI'en.	God-kendt.
Andet: 3.1	Temperatursensoren må maksimalt veje 20 gram.	Måletest: Temperatursensoren vejer maks. 20 gram.	Temperatursenso-ren vejer 5g	God-kendt.
Andet: 3.2	Temperatursensoren kan klare varme op til 45 grader og kulde ned til -5 grader.	Visueltest: Temperatursensoren måler temperaturen mellem -5 grader og 45 grader	Temperaturmåling måler op til 45 grader og ned til -5 grader	God-kendt.

Tabel 16: Ikke-funktionelle krav for temperatursensor

Kamerasensor (Mads og August)

Raspberry Pi Camera Board v1.3 (5mp, 1080p)

Krav nr.	Krav	Test	Faktisk observa-tion/resultat	God-kendt (JA/NEJ)
Fysiske di-mensioner 1.1	Arduinoen skal være: $10.0 \text{ cm} \times 5.5 \text{ cm} \times 3.0 \text{ cm}^1 \pm 2 \text{ cm}$ (MiniElektro, 2023).	Måletest: Arduinoen måles med målebånd, og dimensionerne viser til at være $10.0 \text{ cm} \times 5.5 \text{ cm} \times 3.0 \text{ cm}^1 \pm 2 \text{ cm}$	5,13 cm bredde & 9,87 cm længde	God-kendt.

Fysiske dimensioner 1.2	Raspberry Pi Camera Board v1.3 (5mp, 1080p) skal være: 20 mm x 25 mm x 9 mm ±25 mm (Lavpris, 2023)	Måletest: Kamera måles med målebånd, og dimensionerne viser til at være 20 mm x 25 mm x 9 mm ±25 mm	2,38 cm bredde & 2,49 cm længde	God-kendt.
Fysiske dimensioner 1.3	Kameraets optagelsesopløsning målt i pixels skal være 320px x 240px ±100px.	Måletest: Kameraets video bliver gemt som en fil, hvori oplosning aflæses.	3280x2464px	God-kendt.
GUI: 2.1	Kameraets videovisningsopløsning på brugergrænsefladen målt i pixels skal være 320px x 240px ±100px.	Måletest: Kameraets video-opløsning aflæses på hjemmesiden for at sikre at oplosningen ikke er forringet ved overførsel.	3280x2464px	God-kendt.
GUI: 2.2	Kameraets opdateringshastighed skal være 15fps ±5fps ² .	Måletest: Kameraets opdateringshastighed måles igennem hjemmesiden for at sikre at alle frames er inkluderet.	Observeret omkring 15fps.	God-kendt.
GUI: 2.3	Reaktionstiden fra når kameraet får signal fra bevægelsessensoren til at kameraet livevideo-optagelse fremvises overfor brugeren skal være 5 sekunder ±2 sekunder.	Måletest: Stopur startes ved bevægelse og stoppes ved video vises på GUI	Ved stopur test så vi at det tog 4,2 sekunder fra vi trykkede på start, til det kunne ses på kameraet.	God-kendt.
Andet: 3.1	Arduinoen må maksimalt veje 75.0 gram	Måletest: Arduinoen måles på en vægt	36 gram	God-kendt.

Tabel 17: Ikke-funktionelle krav for kameraasensor

Bevægelsessensor (Altaf og Ruben)

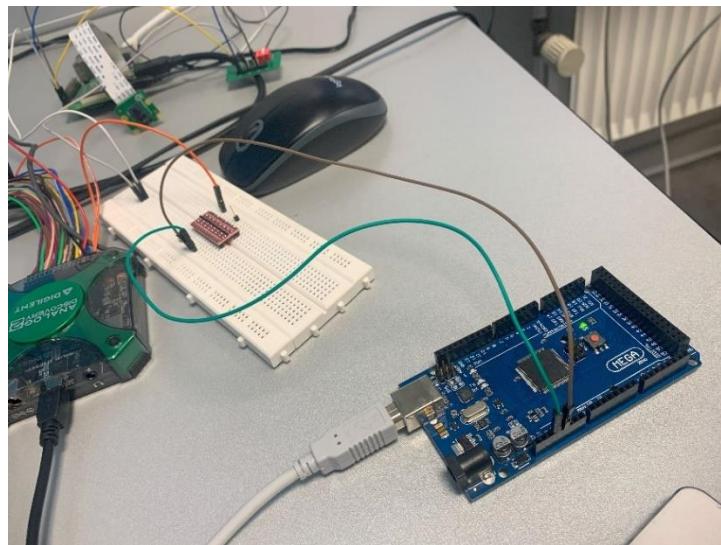
Krav nr.	Krav	Test	Faktisk observasjon/resultat	Godkendt (JA/NEJ)
Punkt 1	Sensoren skal maksimalt måle 8 (længde) x 5 (bredde) x 3 (højde) ±2 cm	Måle test: Vi bruger en lineal til at måle hver dimension og verificere at de er under den maksimale grænse.	3,20 cm længde & 2,41 cm bredde & 2,41 cm højde	Godkendt.
Punkt 2	Sensoren vejer maksimalt 15 gram	Måle test: Sensoren vejes på en vægt med en nøjagtighed, der er bedre end 10 gram. Vægten skal vise 15 gram eller derunder.	6 gram	Godkendt.
Punkt 3	Sensoren skal kunne detekttere bevægelse i en 60 graders vinkel	Visuel test: Vi tegner en 60 graders vinkel og sensoren placeres i nulpunktet. Herefter laver vi bevægelse hhv. Indenfor 60-graders-arealet, og udenfor 60-graders-arealet. Her til verificerer vi, at sensoren registrerer bevægelsen igennem GUI.	Fra et bestemt punkt tegnede vi to linjer med 60 graders mellemrum. Vi placerede herefter sensoren ved mødepunktet for de to linjer, og testede herefter om sensoren reagerede på bevægelse både indenfor 60-graders-arealet, og udenfor 60-graders-arealet.	Godkendt.
Punkt 4	Sensoren sender en string ud på GUI	Visuel test: Når der foretages en bevægelse foran sensoren, ser vi visuelt en tekststreng blive vist på GUI.	Vi ser en tekststreng der enten siger ''Motion detected'' eller ''No motion'' på GUI'en konstant, alt efter sensorens output.	Godkendt.

Punkt 5	Sensoren sender et signal til kameraet om at det skal optage.	Måle test: Når der detekteres bevægelse, begynder kameraet at optage.	Sensoren sender et Motion-high signal, ændrer output på GUI til ''Motion detected'' og i de resterende 10 sekunder fra dette, optages video-streamen fra kameraet, til senere afspilning.	God-kendt.
Punkt 6	10 sekunder efter bevægelse er detekteret, kan videooptagelse afspilles.	Måle test: Bevægelse foretages foran sensoren og vi verificerer at et signal sendes; hvis dette er tilfældet, så starter vi et stopur på 10 sekunder og venter til det rammer 0 sekunder. Herefter verificerer vi, at der er gemt en ny videofil lokal der kan afspilles fra GUI.	Dropdown menuen neders på GUI får endnu en videofil der kan afspilles. Videofilen er 10 sekunder lang og korresponderer med det kameraet kiggede på, da sensoren sendte et højt output.	God-kendt.

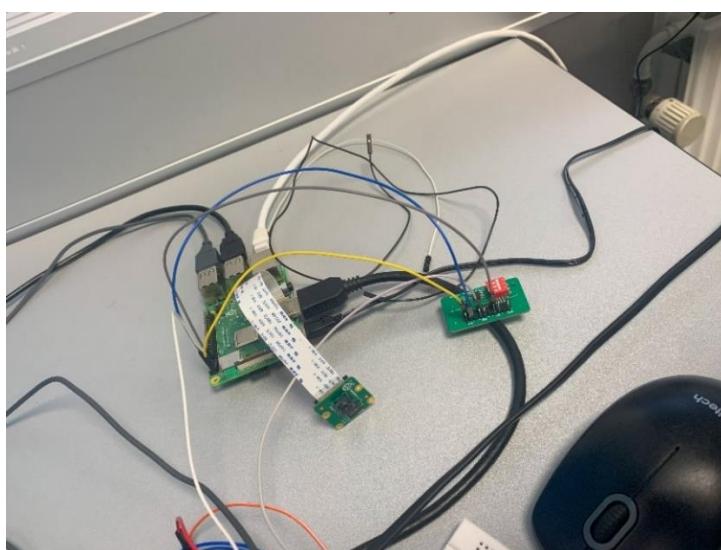
Tabel 18: Ikke-funktionelle krav for Bevægelsessensoren.

Modultest (Mads)

Når man som gruppe i et hvilket som helst projekt skal udarbejde et produkt, er det vigtigt løbende at foretage modultests af de forskellige typer hardware ved hjælp af sit software. Gøres dette ikke, kan man risikere, at fejlene bliver svære at finde frem til desto mere fuldendt produktet bliver. Især hvis produktet er meget komplekst, og består af mange forskellige små hardwarekomponenter, som alle sammen skal kunne snakke sammen for at produktet opnår fuld funktionalitet. Derfor valgte gruppen at udføre en masse forskellige tests ved hjælp af både fumlebræt, men også de rigtige komponenter valgt til det endelige produkt, for at se om der var nogle fejl, som optrådte og skulle løses inden gruppen fortsatte med konstruktionen frem imod det færdigfunktionelle produkt.

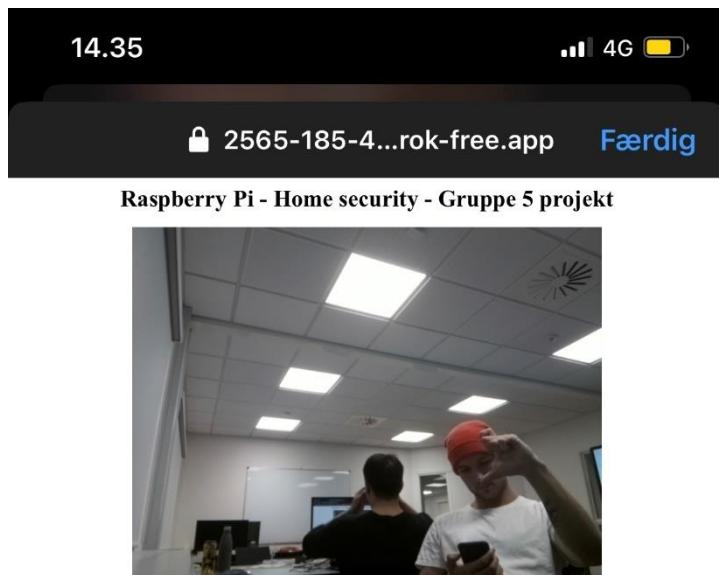


Figur 27: Oversigt over komponenter ved modultest (1)



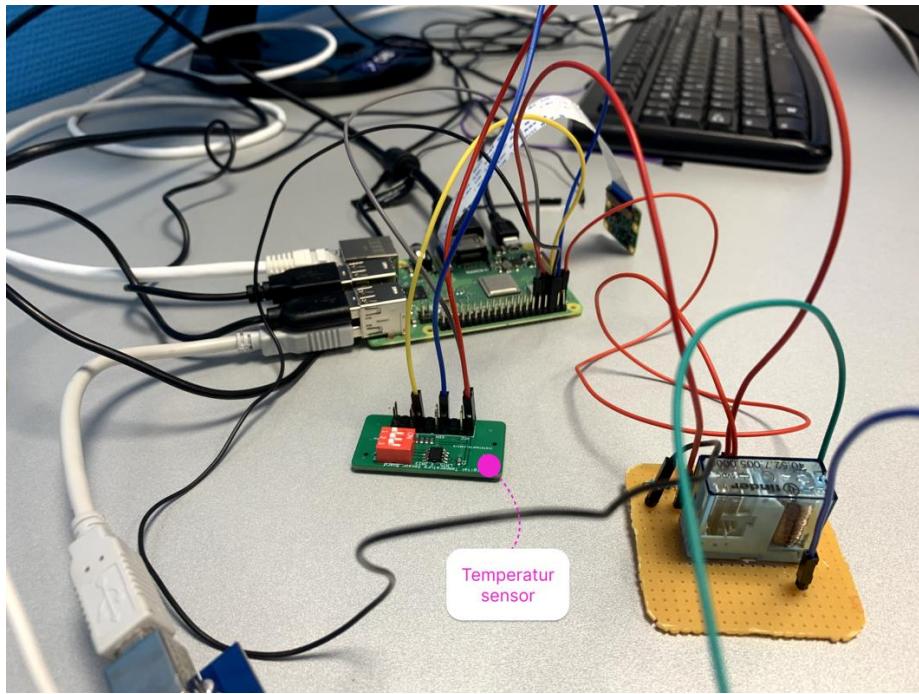
Figur 28: Oversigt over komponenter ved modultest (2)

Efter at gruppen havde foretaget en masse forskellige fejlfindinger ved hjælp af forskellige modultests, f.eks. det beskrevet i afsnittet 'Fejlkilder (August)', kunne gruppen konstruere produktet, som de havde planlagt det skulle være. Efter at dette var gjort, udførte gruppen igen lignende modultests for at se, om det færdigkonstruerede produkt var fuld funktionelt. Det blev testet at de forskellige sensorer udførte de korrekte handlinger gennem GUI, og gruppen fik hermed klargjort, at alt software og hardware for produktet fungerede.

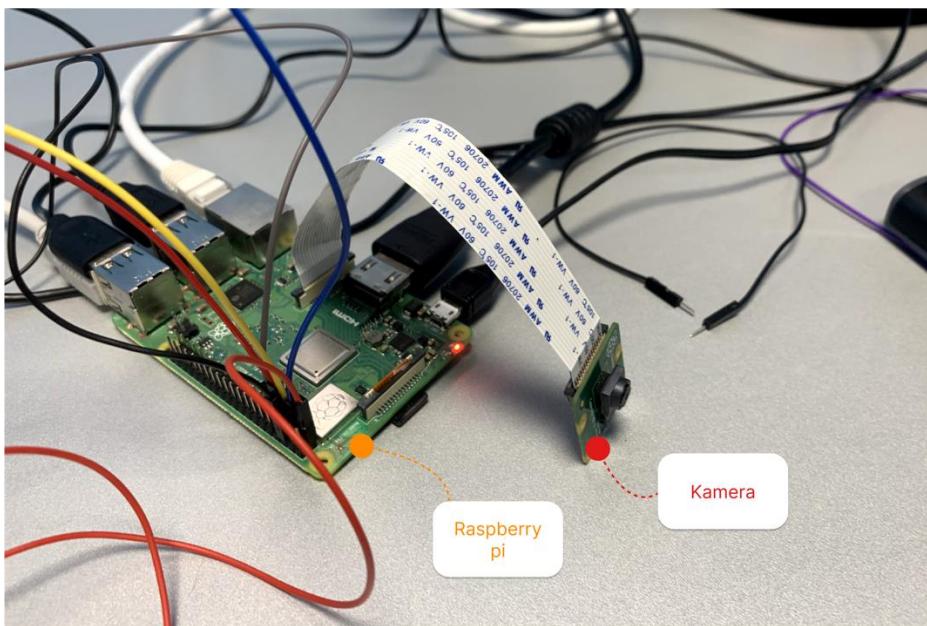


Figur 29: Tjek af fuldt funktionelt produkt gennem GUI

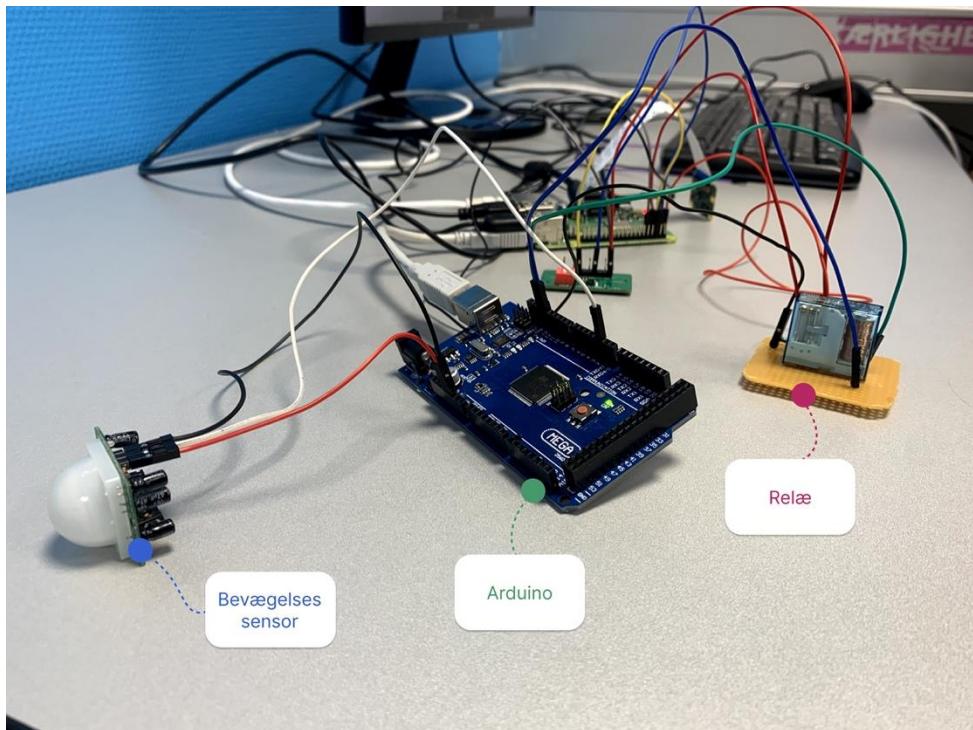
Integrationstest (Ruben)



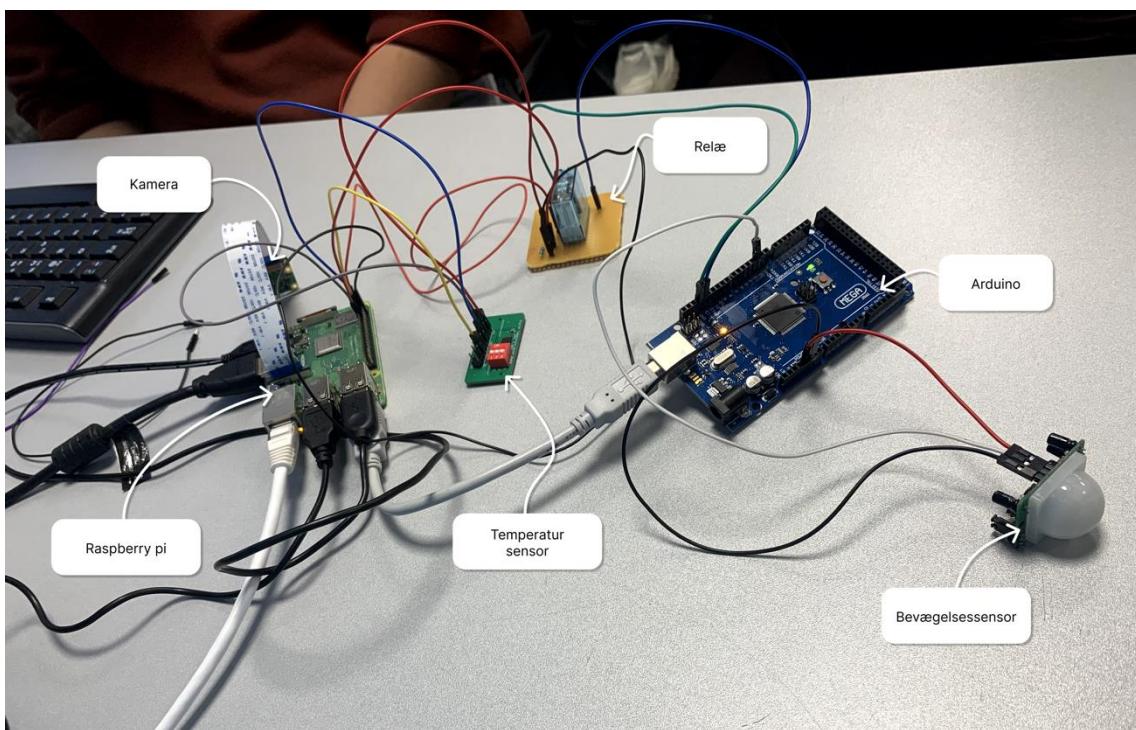
Figur 30: Billede af hvordan temperatur sensoren indgår i den færdige systemopsætning



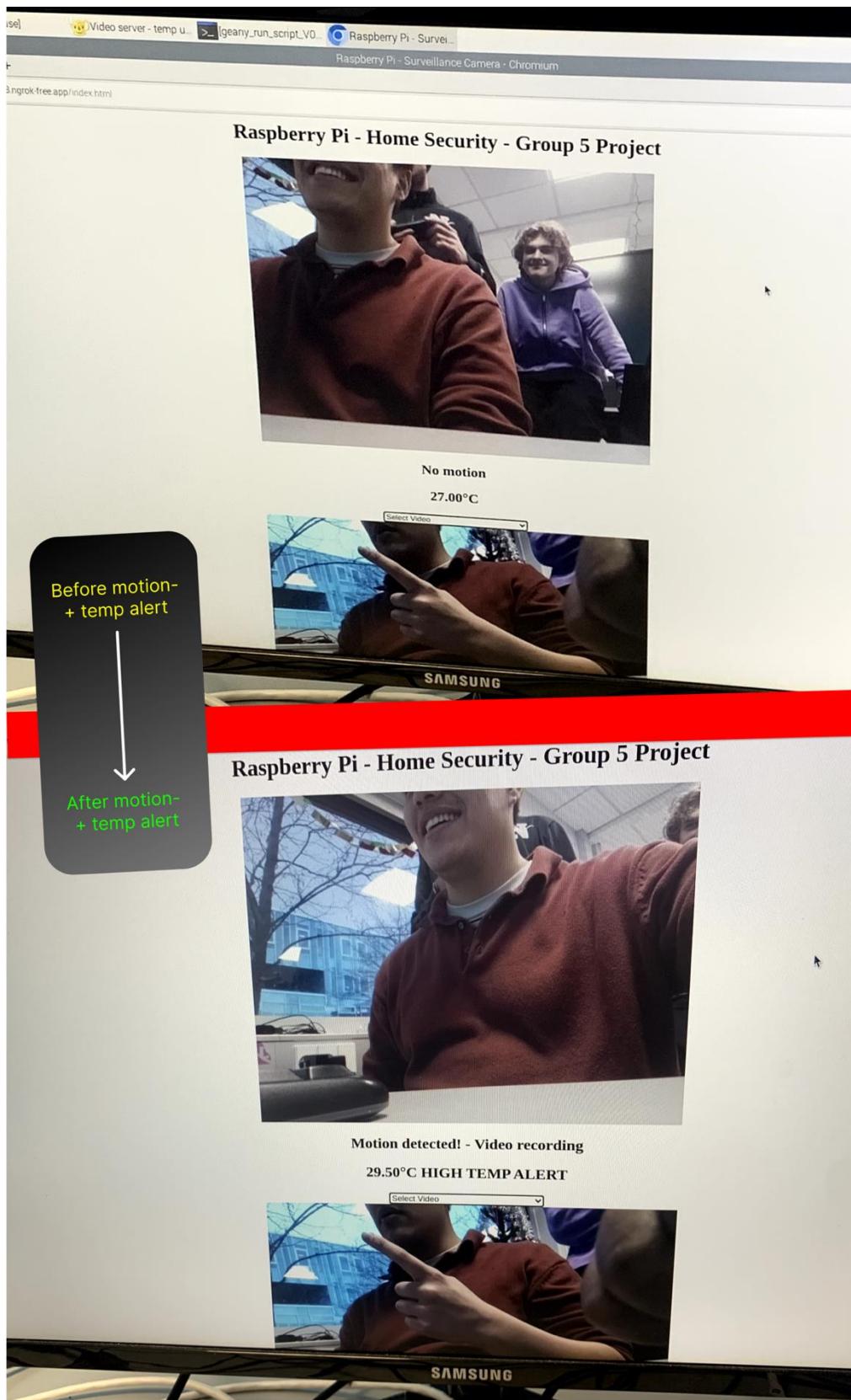
Figur 31: Billede af hvordan Raspberry pi og Kameraet kobles sammen, i den færdige systemopsætning.



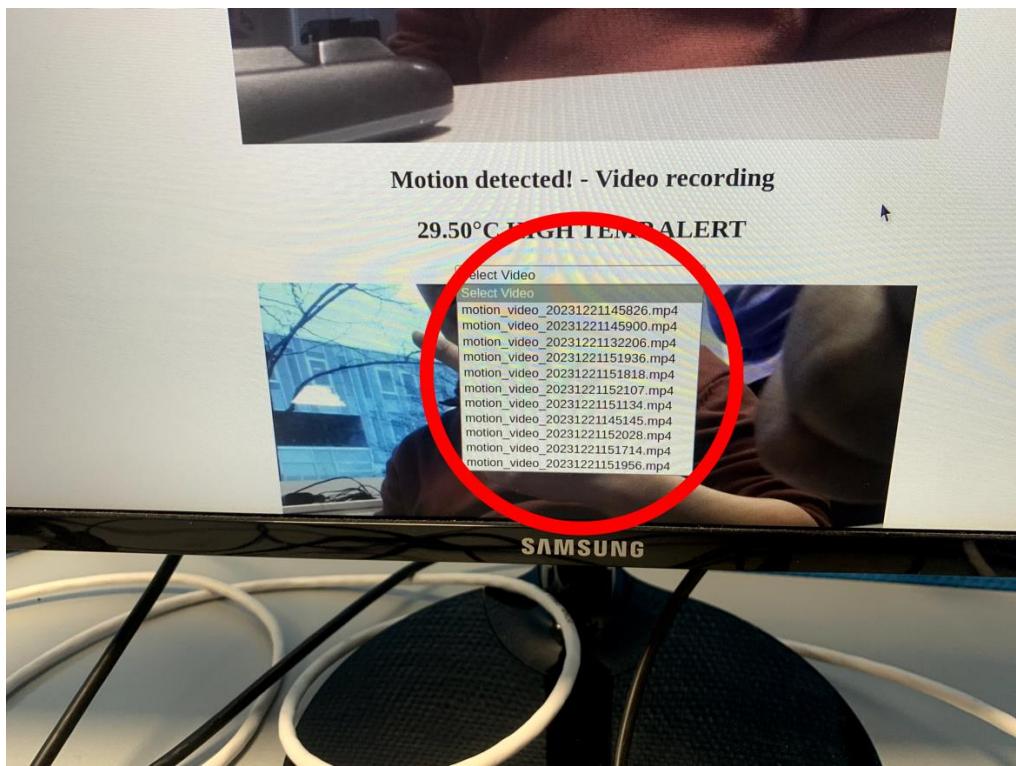
Figur 32: Billede af hvordan Bevægelsessensor, Arduino og Relæ indgår i den færdige systemopsætning



Figur 33: Billede af den færdige systemopsætning.



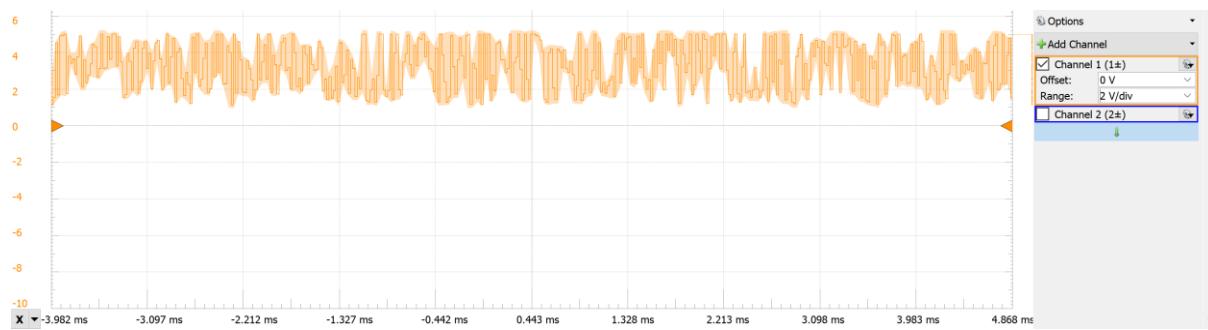
Figur 34: Sammenligning imellem GUI før og efter bevægelse og høj temperatur.



Figur 35: Visuelt overblik over drop-down menu, med alle de optagne videoer.

Fejlkilder (August)

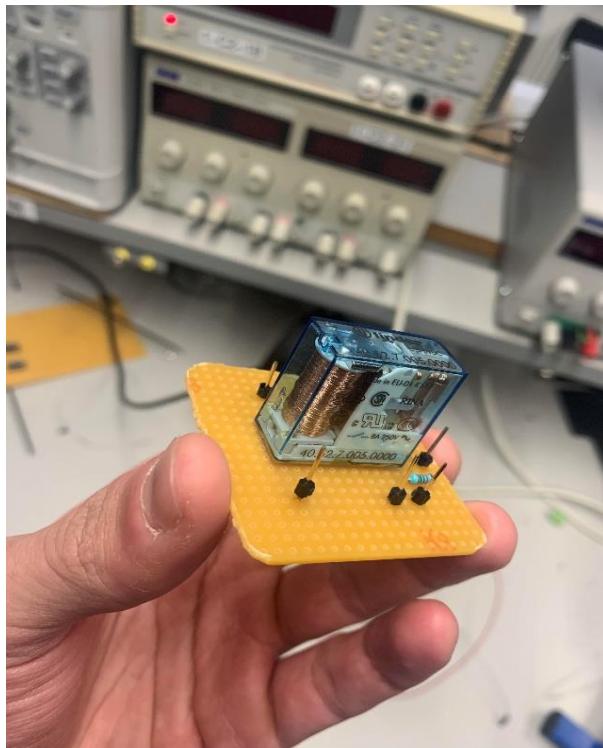
Den største fejkilde der er stødt på, har været den spændingsforskelse der er imellem Arduinoens 5V og Raspberry Piens 3,3V. For at kunne tilføje bevægelsesinputtet til GUI skal de nemlig forbindes imellem Arduinoens output og Raspberry Piens GPIO pin. Disse kan dog ikke forbindes direkte da Arduinoen kører på 5V og Raspberry pi'en kun kan tage 3,3V på sine GPIO pins. Derfor var første forsøg at skabe en simpel voltage divider med modstande, som kunne give de 3,3V. Dog opstod der problemer, da der var en markant forskel i Arduinoens ground og Raspberry Piens ground. Dette er i hvert fald den primære teori. Det blev målt med et oscilloskop og kan ses her:



Figur 36: Ground sat til Raspberry pi og input 1 sat til Arduinoens output for bevægelsessensor.

Derfor sås den bedste løsning som tilføjelse af et relæ, sådan at Arduinoen blot åbner og lukker for det, hvilket så kan skabe forbindelse imellem Raspberry piens egen 3,3v pin og GPIO-pinnen. Udover dette blev der tilføjet en pull-down resistor på 10k ohm, hvilket altid er en god ide når man har med digitale input at gøre. Denne sikrer sig, at spændingen ikke flukturede når der ingen input er. Udover dette, gives der også en ekstra ground til temperatursensoren.

Alt dette skaber selvfølgelig behov for en ekstra printplade, samt et forholdsvis strømkrævende relæ, hvilket ikke er optimalt. Nedenfor ses et billede af printpladen:



Figur 37: printplade med relæ

Et andet kompromis som måtte tages, var kameraets funktion. I koden, kan et kamera-objekt kun initialiseres en enkelt gang. Dette betyder derfor også, at hvis man både ønsker at kameraet skal kunne live-streame men også optage video, må den ene stoppes, før den anden kan startes. Dette er resultatet af at en kamerainstans kun kan enten optage video eller taget et billede. Da livestreamen egentlig blot er en lang streng af billeder (MJPG er en video form af billede typen JPG), er kameraet allerede fra starten initialiseret i billede-tilstand. For at dette kan overkommes, må livestreamen stoppes, før optagning af video kan starte. Dette betyder altså, at i de 10 sekunder efter at bevægelse er blevet opfanget, vil livestreamen være frosset. Dette er selvfølgelig heller ikke optimalt, men understøtter stadig den overordnede ønskede funktion.

Tidsplan (Mads og Louise)

Gennem projektets forløb er en tidsplan udarbejdet i starten af projektet blevet fulgt og er løbende blevet rettet til ud fra, om gruppen overholdt de forskellige satte deadlines. Disse deadlines formåede gruppen delvist at overholde ved hjælp af ugentlige vejledermøder med gruppens tildelte vejleder Gustav Kastrup Nielsen. Den endelige tidsplan kan ses på Figur 38 og Figur 39.

Faser\Ugenr.	36	37	38	39	40	41	42	43	44
Opstart							Efterårsferie		
Samarbejdsaftale									
Projektformulering									
Tidsplan									
Kravspecifikation									
Acceptestspecifikation									
Review 1						06.10.2023			
HW-arkitektur									
SW-arkitektur									
HW-design									
SW-design									
Review 2									
HW-modultest									
SW-modultest									
Integrationstest									
Accepttest									
Rapportskrivning									
Aflevering	04.09.2023	11.09.2023	18.11.2023	25.11.2023	02.10.2023	09.10.2023	16.10.2023	23.10.2023	30.10.2023

Figur 38: Tidsplan for projektets forløb uge 36 - 44

Faser\Ugenr.	45	46	47	48	49	50	51	52	1
Opstart									
Samarbejdsaftale									
Projektformulering									
Tidsplan									
Kravspecifikation									
Acceptestspecifikation									
Review 1									
HW-arkitektur									
SW-arkitektur									
HW-design									
SW-design									
Review 2	10.11.2023								
HW-modultest									
SW-modultest									
Integrationstest									
Accepttest									
Rapportskrivning									
Aflevering	06.11.2023	13.11.2023	20.11.2023	27.11.2023	04.12.2023	11.12.2023	18.12.2023	25.12.2023	01.01.2024

Figur 39: Tidsplan for projektets forløb uge 45 - 1

Konklusion (Alle gruppemedlemmer)

Konklusion for hardwaredelen

På baggrund af hardwaredelen for projektet kan gruppen konkludere, at alt hardwaren for gruppens produkt for anden semesterprojektet opfylder kravene og fungerer. I gruppen er de forskellige stykker hardware blevet fordelt ud på de forskellige gruppemedlemmer, hvor Louise og Hafsa har stået for temperatursensoren og dens funktionalitet, Ruben og Altaf har stået for bevægelsessensoren og dens funktionalitet, August og Mads har stået for kamerasensoren og dens funktionalitet, og August har stået for GUI samt web-serveren på Raspberry Pi og deres funktionalitet. Hvert gruppemedlem har haft til ansvar, at deres hardware har overholdt kravene, og har været fuldt funktionel for, at smart-hjemmesikkerhedssystemet i sidste ende har kunne fungere og gennemføre de forskellige udarbejdede accepttestspezifikationskrav.

Konklusion for softwaredelen

På baggrund af softwaredelen for projektet kan gruppen konkludere, at alt softwaren, som skal styre hardwaren i smart-hjemmesikkerhedssystemet, fungerer. Gruppen har i dette projekt gjort brug af programmeringssprogene Python og C++ til at udarbejde en brugergrænseflade (GUI), hvorfra forskellige sensorers opsamlede data fra miljøet har kunne overvåges. Denne brugergrænseflade har kunne tilgås fra enhver lokation via en specifik linkadresse grundet hele systemet blev opsat som en web-server gennem web-servertjenesten Ngrok (Ngrok, 2024).

Bevægelsessensoren leverer et digitalt output, der skifter til "HIGH" når bevægelse registreres, og vender tilbage til "LOW", når bevægelse ikke registreres. Dette gør det muligt for systemet at reagere på bevægelse og dermed sende et signal videre til gruppens Arduino, som sensoren er koblet til via de dertil definerede pins. Temperatursensoren bruger I2C-protokollen til at kommunikere med gruppens Raspberry Pi. Gennem den valgte adresse læser sensoren data og konverterer dem til en temperatur i grader Celsius. Kamerasensoren har både kode for optagelse af video og livestreaming, og består også af flere klasser, som styrer de forskellige metoder. Bl.a. klassen CameraManager, som har funktionerne start_streaming, stop_streaming, og record_video.

Alle disse nævnte forskellige sensors kode er samlet i gruppens brugergrænseflade (GUI). Den indeholder en stor del kode, da den består af både en web-server, brugergrænseflade og funktioner til integration af klasserne fra de forskellige sensorer.

Gruppen hoster web-serveren ved hjælp servicetjenesten Ngrok (Ngrok, 2024) en HTML/JSON-hjemmeside til internettet, som kan tilgås fra alle steder i verden gennem en specifik linkadresse.

Overordnede konklusion for projektet

På baggrund af hele projektet kan gruppen konkludere, at der med viden fra de forskellige kurser/fag på 2. semester, samt allerede opnået viden fra 1. semesters kurser, kan udvikles hardware og software, som sammen ender ud i et velfungerende produkt, som i dette tilfælde er smart-hjemmesikkerhedssystemet.

Gruppen har ved hjælp af kurset, Indledende System Engineering, kunne udarbejde forskellige ingeniørfaglige diagrammer for projektets produkt, bl.a. Block Definition Diagram, Internal Block Diagram og Domainmodel, som ved støttende hjælp fra vejleder har resulteret i en mere overskuelig og klar arbejdsproces frem imod det færdigkonstrueret produkt.

Softwareen for projektet er blevet udarbejdet af gruppens forskellige medlemmer ud fra allerede opnået viden fra 1. semester samt kildekritisk søgen gennem internettet. Temperatursensoren bruger I2C-protokollen til at kommunikere med Raspberry Pi'en, hvilket er noget som gruppen ved hjælp af kurset, Grænseflader Til Den Fysiske Verden, og dets laboratorieøvelser, har kunne udarbejde sig fra.

Alle stillede krav i projektet opfyldes for både hardware og software. Det færdigkonstrueret produkt fungerer efter systembeskrivelse fremstående i rapporten. Alle gruppemedlemmer har udarbejdet og fuldendt deres tildelt del af projektet, og tidsplanen er nogenlunde blevet overholdt hele vejen igennem projektets forløb i takt med de forskellige ugentlige opfølgningsvejledermøder (se bilag).

Individuelle konklusioner

Mads' individuelle konklusion

Jeg kan på baggrund af 2. semesterprojekt individuelt konkludere, at jeg har fået et stort udbytte fra alle de forskellige dele fra de forskellige fag på 2. semester, som projektet har været bygget op omkring.

Jeg har i løbet af projektprocessen fået videreudviklet mine samarbejdsevner fra 1. semesterprojekt. Jeg har oplevet, hvordan man håndtere nye forskellige gruppemedlemmers personligheder og arbejdsmoraler ift. at nå forskellige deadlines, og hvor vigtigt endnu engang det er, at have en klar aftale om, hvornår noget specifikt arbejde skal være færdiggjort. Hvis ikke klare aftaler har været sat, vil deadlines og tidsplan eventuelt ikke overholdes, og mere pres vil blive lagt på alle medlemmer i gruppen for at kunne nå de forskellige deadlines.

Både hardware- og softwaremæssigt har jeg lært yderligere at kunne benytte mig af forskellige programmeringssprog til forskellige komponenter og mål. Heriblandt Python, C og C++. jeg har også videreudviklet mine kompetencer for, hvordan man håndterer, når noget ikke virker, og hvordan man ved hjælp af fejlfindingstests såsom debugging kan komme frem til endelig at få noget til at virke, både hardware- og softwaremæssigt.

Semesterprojektet på 2. semester har ligesom med 1. semesterprojektet været bygget op omkring alle kurserne/fagene på semesteret, hvor man har fået lov til at benytte sig af den viden, som man har fået fra de forskellige individuelle kurser/fag. Dog til forandring har vi som gruppe bestående af ren softwarestuderende personer fået lov til at have et lidt mere frit valg ift. hvad vi som gruppe ville udarbejde. Personlig synes jeg, at det har været en god læringsmetode, da jeg altid har været sådan en person, som lærer mere ved at få det i hænderne sammenlignet med at have det skrevet på papir. At selv at få lov til at vælge lidt mere bredt har også været med til at vi selv som gruppe har kunne sætte fokusområdet, da vi selv har måtte udarbejde produktet.

Hafssas individuelle konklusion

Efter at have arbejdet på Semesterprojekt 2, kan jeg se tilbage på et forløb, hvor samarbejde og fælles indsats virkelig har gjort en forskel. Jeg har tidligere fået erfaring fra Semesterprojekt 1 i forhold til, hvordan man arbejder i en gruppe. Trods min erfaring fra det tidligere semester, var dette semesterprojekt meget udfordrende. Nogle af udfordringerne skyldtes vores gruppodynamik, da der var mangel på kommunikation mellem gruppen og et af medlemmerne. Forskellige arbejdsstile førte til konflikter og misforståelser, men dette har også lært mig værdien af kompromis og fleksibilitet. Udover det, har været en fornøjelse at arbejde med denne gruppe. Det var så fedt at se, hvordan vores forskellige evner komplementerede hinanden. Hver gang vi stødte på et problem, kunne vi sammen finde en løsning, og jeg tror, at det er sådan vi fik det bedste resultat ud af vores arbejde.

Dette semesterprojekt var bygget op omkring alle fagene fra dette semester. Det har hjulpet mig med at forstå fagene bedre. I forhold til Semesterprojekt 1 var dette projekt meget mere komplekst. Vi var meget mere på egne ben, og vi skulle starte hele projektet fra bunden af. I det første semesterprojekt havde vi mange flere hjælpemidler. Vi er en gruppe bestående kun af softwarestuderende, hvilket gjorde, at vi kunne frit vælge vores emne. Jeg personligt var meget taknemmelig for dette, da det gjorde, at vi kunne være kreative med vores valg.

Jeg kunne bedst lide at arbejde med software-delen, da jeg interesserer mig for det, men dette semesterprojekt har styrket min viden om hardware. Jeg føler, vi havde en struktureret tilgang til vores projekt, hvad angår tidsplan og arbejdsfordeling. Dette har gjort, at vi kunne blive færdige med vores projekt uden at være stressede på grund af eksamener i de andre fag.

Selvom det ikke altid var let, er jeg taknemmelig for hver eneste udfordring, vi stod overfor, fordi de har formet mig og gjort mig bedre rustet til fremtidige projekter. Og selvom projektforløbet måske ikke var perfekt, er processen og de færdigheder, jeg har opnået, noget, jeg værdsætter højt.

Altafs individuelle konklusion

Gennem dette semesterprojekt har jeg - og vi som gruppe - stået overfor forskellige udfordringer og dilemmaer. Selvom vi har haft det svært med manglende bidrag og tilstedeværelse af en i gruppen, har vi andre drøftet tingene og løftet projektet i samlet flok.

Dertil har vi andre i gruppen selvfølgelig også haft kommunikative problemer her og der, men dette har vi løst, hvilket også kan ses på det færdige produkt. Selvom disse ting ikke har været let for mig - da jeg arbejder bedst med klare retningslinjer og protokoller -, har dette styrket min evne til at håndtere og samarbejde konstruktivt gennem projektet.

Personligt synes jeg, at de fleste i gruppen har skabt et positivt og støttende læringsmiljø, hvor vi har hjulpet hinanden og sammen holdt overblikket. Selv når ikke alle har bidraget lige meget, har vi som individer taget initiativ til at fremme gruppedydynamikken og dertil støtte hinanden gennem vanskelige situationer.

Til sidste vil jeg tilføje, at jeg - på trods af udfordringer i gruppen -, er stolt af den samlede præstation og det produkt, vi har skabt.

Louises individuelle konklusion

Dette semesterprojekt har været en udfordrende, men lærerig rejse for mig. Vi har desværre stået over for kommunikationsvanskeligheder, hvilket har resulteret i manglende klare retningslinjer og forventninger for tidsstyring fra starten af projektet. Dette har givet os problemer med at nå deadlines. Gruppen har desværre ikke prioriteret at indgå kompromiser i løbet af projektet, men har i stedet ladet flertallet bestemme og der er derfor ikke blevet taget højde for vores forskellige udfordringer i hverdagen. Forsøg på at nå fælles beslutninger eller finde midtpunkter har ikke været en central del af vores tilgang, hvilket har resulteret i en mangel på afbalancering og en udfordrende dynamik. Jeg erkender, at min egen udfordring med at kommunikere effektivt til de andre gruppemedlemmer har været med til at bidrage, til denne problematik. Den manglende klarhed blandt gruppemedlemmer, bidrog til et suboptimalt arbejdsmiljø, hvilket påvirkede vores evne til effektivt at gennemføre opgaverne. Vi er dog kommet igennem projektet, med et meget flot produkt.

De samarbejdsvanskeligheder, vi stod over for, har dog ikke overskygget de positive aspekter af projektet. På trods af disse udfordringer har jeg dog opnået en betydelig mængde viden inden for både software og hardware, hvilket har bidraget til min personlige og faglige udvikling. Projektet har tvunget mig til at fordybe mig i komplekse tekniske emner og har styrket mine kompetencer på dette område. Jeg har lært at navigere gennem tekniske udfordringer og jeg har udviklet en dybere forståelse for både software- og hardwareaspekterne af projektarbejde.

Selvom samarbejdsprocessen var udfordrende, har jeg draget værdifulde erfaringer om vigtigheden af effektiv kommunikation og teamkoordination. Min egen manglende evne til at formidle information effektivt har været en værdifuld erkendelse, som jeg vil arbejde på at forbedre i fremtiden. Jeg indser nu, at et effektivt samarbejde ikke kun kræver forståelse for tekniske aspekter, men også evnen til at kommunikere klart og åbent med alle gruppemedlemmer.

På trods af de hårde tider under projektet er jeg glad for de færdigheder og den viden, jeg har opnået i løbet af projektet. Dette semester har ikke kun styrket mine tekniske kompetencer, men har også udstyret mig med værdifulde indsiger i gruppedynamik og samarbejde, som jeg vil tage med mig videre i min akademiske og professionelle rejse. Jeg ser frem til at anvende disse erfaringer i fremtidige samarbejdsprojekter og fortsætte min personlige og faglige udvikling.

Rubens individuelle konklusion

Da 2. semesterprojektet nu er ved at nærme sig sin ende, er det tid til at kigge tilbage og reflektere over hvilke erfaringer jeg går herfra med, både på godt og ondt.

Det mest prominente ved dette projekt for mig, har været at det på ingen måde har føltes lige så nemt og ligetil, som i det første semesterprojekt. I modsætning til tidligere har vi været meget mere på egne ben i forhold til opgavens rammer, den kreative frihed, mængden af vejledning og gruppens overordnede dynamik. Alle disse faktorer har føltes markant anderledes, fra hvad jeg oplevede sidste semester. Dette er dog ikke nødvendigvis en dårlig ting, da det dybest set fra start af har tvunget både mig, og også mine gruppemedlemmer, til virkelig at tage os sammen hvis vi ville gøre os nogle forhåbninger om at nå i mål med projektet.

Internt er der til tider opstået problematikker i forhold til kommunikationen og det har også virket til at motivationen iblandt de forskellige gruppemedlemmer har været lidt svingende. Imidlertid har der dog stadig været en fælles kampgejst og vi har været gode til at opildne hinanden og fokusere på det endelige mål, fremfor de, til tider, trælse situationer vi har været i. Når problemer er opstået, har vi været gode til at løse dem på tværs af de forskellige ansvarsområder i gruppen. Det har aldrig føltes som om os som stod for bevægelsessensoren, ikke har kunne hjælpe dem som stod for kameraet, og vice versa. Jeg syntes selv at jeg har bevaret fokus og holdt mig til den mentalitet at *''Vi færdiggør en ting ad gangen, og så skal vi nok nå i mål''*, hvilket nu også har vist sig succesfuldt.

Jeg står helt klart tilbage med en følelse af succes, da vi er endt med et fuldt funktionelt produkt, der lever op til de krav vi stillede fra starten af. Dog har det været en noget snævret og ulige vej dertil, med mange problematikker som, til tider, virkede fuldstændig uoverskelige. Men når vi så er endt med at løse et svært problem, har det også være meget belønnende og helt sikkert resulteret i en øget selvtillid, gående ind i 3. semester.

Augusts individuelle konklusion

Der har, igennem projektet, været mange udfordringer samt succeser som alle har udvidet min horisont for gruppearbejde og produktudvikling. Udfordringerne har været alt fra strøm-spænding til dårlig koordination. Når noget så stort som dette projekt skal laves af forskellige mennesker, som alle kontribuer i forskellige mængder og på forskellige måder, kan det kun forventes at der vil opstå lidt komplikationer. Især det med at møde op og svare på beskeder har været en udfordring for nogle. Dog, igennem alt, har vi formået at kunne overkomme disse problemer og skabe et sammenhængende forløb for udviklingen.

Jeg har selv lært meget om mine egne styrker og svagheder igennem dette semester, takket være projektet. Uden en klar opgave, er jeg meget hovedløs og har svært ved at kontribuere med meget, hvilket selvfølgelig ikke går i et gruppe-sammenhæng. Men med klar forståelse for dette har jeg kunne bede ind om opgaver, som jeg har kunnet løse selv. Dette har f.eks. været GUI 'en og funktionaliteten af Raspberry pi'en, som jeg med nogenlunde frihed har kunne fordybe mig i. Dette har givet mig en masse nyttig viden om hvad der er muligt og hvad der ikke er, hvilket jeg har kunnet give videre til gruppens forbedring af overblik.

Projektet har også hjulpet stort med forståelse af UML-diagrammer i et større sammenhæng, sådan at alt går op i en højere enhed. Forståelse for UML-diagrammer har også hjulpet med en af mine personlige svagheder, som nemlig er mangel på overblik. Herunder har det også givet rigtig god oplæring i brug af sensorer til løsning af opgaver, hvilket helt klart er en stor fordel til fremtidige projekter.

Jeg er personligt meget stolt af vores færdige projekt, og syntes rent faktisk at det er super brugbart som produkt. For at teste det, brugte jeg det nemlig til at overvåge min lejlighed imens jeg var på Sjælland. Her kunne jeg se en live-feed fra min lejlighed hvornår end jeg ville, tjekke temperaturen og se om der er nogen der har været inde i lejligheden på video.

Jeg kan derfor kun se frem til de udfordringer og succeser som 3. semesters projekt vil bringe.

Bilag

1. E2PRJ2_Projektoplæg.pdf
2. Vejledning_for_gennemfoerelse_af_projekt_2_V1_00.pdf
3. Arbejdet i projektgruppen 2. semester.pdf
4. tidsplan_gruppe_5_fardig.pdf
5. Samarbejdsaftale_for_Semesterprojekt_2_Gruppe_5.pdf
6. 1._vejledermode__140923.pdf
7. 2._vejledermode__200923.pdf
8. 3._vejledermode__270923.pdf
9. 4._vejledermode__041023.pdf
10. 5._vejledermode__251023.pdf
11. 6._vejledermode__011123.pdf
12. 7._vejledermode__081123.pdf
13. 8._vejledermode__151123.pdf
14. 9._vejledermode__221123.pdf
15. PIRMotionSensorDONE.zip
16. Video_server_temp_update.py
17. Temperaturesensor.py

Bibliografi

- Bosanac, D. J. (5. Maj 2021). *Smart-home vinder frem i danske hjem*. Hentet fra Danmarks Statistik:
<https://www.dst.dk/da/Statistik/nyheder-analyser-publ/nyt/NytHtml?cid=32813>
- Chabert, I. P. (4. August 2023). *Mindre stigning i anmeldte butikstyverier*. Hentet fra Danmarks Statistik: <https://www.dst.dk/da/Statistik/nyheder-analyser-publ/nyt/NytHtml?cid=45731>
- Components, R. (01. November 2023). *Raspberry Pi Camera Module*. Hentet fra Raspberry Pi:
<https://docs.rs-online.com/2888/0900766b8127db0a.pdf>
- Lavpris, E. (21. December 2023). *Raspberry Pi kamera modul - 5Mpix. v1.3*. Hentet fra Elektronik Lavpris: <https://elektronik-lavpris.dk/p145103/rpi-camera-board-raspberry-pi-kamera-modul-5mpix-v13/>
- Ngrok. (4. Januar 2024). *Ngrok*. Hentet fra Ngrok: <https://ngrok.com/>
- Reichelt. (01. November 2023). *Raspberry Pi Zero W v.1.1*. Hentet fra Reichelt:
[https://www.reichelt.com/de/en/raspberry-pi-zero-w-v-1-1-1-ghz-512-mb-ram-wi-fi-bt-rasp-pizero-w-p256438.html](https://www.reichelt.com/de/en/raspberry-pi-zero-w-v-1-1-1-ghz-512-mb-ram-wi-fi-bt-rasp-pi-zero-w-p256438.html)
- Tassy, A. (8. Oktober 2020). *Danske hjem bliver mere intelligente*. Hentet fra Danmarks Statistik:
<https://www.dst.dk/da/Statistik/nyheder-analyser-publ/nyt/NytHtml?cid=41576>