milepæl 1

Lexar Kjøgx - Gruppe 4 - 6 medlemmer - (Daniel, Magnus, Sebastian, Lexar, André og jacob)

Opgave 1 - Base - Milepælsopgave

hvad lærte jeg i fagene?

programmering

i programmering der har vi lært en masse om hvordan c++ generelt virker og lært det grundlæggende af hvad man kan gøre i c++. VI har lært hvordan vi opsætter et miljø i visual studio code vi kan arbejde i med alle mulige projekter og især med vores ESP-32. vi har lært en mase om hvordan man kommunikere med en ESP-32 og hvordan vi definere de fysiske pins på Espen i vores kode. Vi kan nu give dem forskellige definitioner og bestemme hvilke pins der styrer de forskellige motorer i vores kode.

Vi kan styre mængden af strøm der bliver sendt ud igennem de pins og se hvordan det påvirker DC-motorerne. vi kan i koden nu styre 4 individuelle DC-motorer og vi har lært hvordan vi kan få roveren til køre langsomt, hurtigt, til højre og til venstre.

Der er blevet undersøgt hvordan vi får sensorerne sat op, som der gav udfordringer til at starte med da der er 2 typer pins sensorerne skal bruge. Time-of-flight sensorer skal bruge en SDA pin og en SCL pin, og dem er der kun 2 af på vores ESP-32. Der er derfor blevet lagt meget tid i at få flere sensorer sat op ved at bruge XSHUT pins, som der gør at vi kan give vores sensorer unikke adresser så Espen kan differentiere mellem sensorerne som har gjort at vi kan bruge de 3 sensorer individuelt af hinanden i forskellige funktioner. derfor har vi nu funktioner der bruger vores sensorer til at måle hvad afstanden, og så efter de afstandsparametre vi har sat så starter den forskellige funktioner alt efter hvilke parametre der bliver opfyldt.

nu vil jeg gerne undersøge hvordan vi kan få vores kode til roveren kogt ned, gjort pænt og om der er nogen bedre funktioner i c++ som vi kan bruge, istedet for dem som vi bruger ligenu.

Elektronik og indlejrede systemer

vi har arbejdet med de forskellige komponenter som f.eks. modstande, kondensatorerne, kapacitatorer og H-broer. H-broer har vi lært hvordan fungere ved først at bygge et kredsløb med knapper der bruger det samme koncept som en H-bro er bygget efter, der har vi bagefter koblet kredsløbet til vores DC-motorerne og testet det. Den information har vi derefter brugt til at koble vores H-broer op til vores motorer og til vores ESP-32 som kan styre dem lidt ligesom vi gjorde med knapperne, så vores base nu kunne køre frem og tilbage.

Efter vi havde gjort det, prøvede vi at forbinde en sensor, som var problematisk, men vi fik den ene sensor til at virke, så prøvede vi at forbinde 2 mere til, som tog ekstremt lang tid at få at til at virke efter vi havde skrevet en kode vi vidste ville virke da den var blevet testet uden roveren kørte. Vi fejlsøgte i koden i rigtig lang tid. vi fandt derefter frem til at koden slet ikke var problemet, men at vi efterhånden havde så mange ting på roveren, at der bare ikke var strøm nok til de kunne virke imens at motorerne også var tændt, som vi løste ved at tilføje en ekstern strømforsyning.

Det har medført at elektronikken på basen nu virker og at det er blevet muligt at koble lygter på foran og bagved.

projekt

i projekt har vi arbejdet med de 4 forskellige faser inden for projektarbejde, idé og målsætning, analyse og planlægning, gennemførelse med overgang til drift, samt evaluering og læring.

vi har brugt mindmaps og brainstorming til at opbygge en generel ide om hvad det er vi gerne vil opnå i milepæl 1, vi har efter de ideer sat os nogen mål vi rigtig gerne ville have opfyldt, derefter har vi haft nogen forskellige arbejdsopgaver som alle sammen er blevet udfyldt som har gjort vi har opnået de mål vi har sat for os selv. Bagefter har vi sat flere ting på for at udfordre os selv og komme lidt på forkant i næste milepæl.

Komponentliste

- base
- 2x H-bro
- 1x Fumlebræt med 2 røde LED'er(bak lys)
- 1x Fumlebræt med 2 hvide LED'er(for lys)
- 1x Fumlebræt med styring
- 1x ESP-32
- 4x DC-motorer
- 3x Time-of-flight sensor
- 39x jumpers

Beskrivelse af basen

Basen er opbygget sådan at der er to H-broer som styrer DC-motorerne, altså hjulene, de er sat op sådan at motorerne i venstre side sidder i en H-bro og at motorerne i højre side sidder i den anden H-bro. Vi har valgt at gøre sådan fordi at det mener vi giver den bedste styring til når roveren skal dreje og generelt er mere overskueligt at styre. foran der har vi en time-of-flight sensor sådan at roveren "kan se" når den er ved at køre ind i noget, det styrer vi gennem vores ESP-32. Det samme gør vi med vores to andre time-of-flight sensorer, hvor der en sidder en på venstre side og en på højre side. På basen der har vi også 3 fumlebræt med, et til bag lygter, et til for lygter og en til generel styring med ESP-32. vi har så skrevet en kode der gør at ESP'en tænder og slukker motorerne alt efter hvad sensorerne læser, sådan at hvis den er tæt på noget i højre side drejer den til venstre, og det modsatte i den anden side, hvis den ser noget tæt på foran, så bakker den og drejer en lille smule indtil den er fri fra det der er foran den.

Det har været udfordrende at få startet op på at bygge basen og sammensætte delene korrekt i samarbejde med koden. Der er blevet brugt meget tid på fejlfinding. Som jeg helt klart finder mest udfordrende da det er helt nye emner vi arbejder med så derfor skal man lære mens man fejlfinder. Hvis man starter sin fejlfinding det forkerte sted er det allerede der svært at finde fejlen hurtigt. vi har haft en udfordring med at vi ikke har haft strøm til at trække hele roveren med en forsyning, så det tog lidt ekstra tid at regne ud hvorfor roveren ikke gjorde som vi havde bedt den om i koden.

Vores rover fungerer nu ret godt med at den kan køre selv uden at kører ind i noget, jeg er stolt af at på en måned der har vi lært hvordan programmeringen skal sættes op men også at vi kan bygge basen op fra bunden med motorer, H-broer og styring, og at vi har fået det hele til at spille sammen med vores ESP-32, især sensorerne har været en udfordring så det er jeg meget stolt over der virker ikke bare med én sensor, men med tre sensorer der kan bruges.

milepæl 2

Lexar Kjøgx - Gruppe 4 - 5 medlemmer - (Daniel, Magnus, Sebastian, Lexar og André)

Hvad lærte jeg i fagene?

Programmering

I programmering der har vi haft om structs som gør det muligt at strukturere koden og bygge objekter i den. Vi har også haft om at lave classes i koden, der gør at vi kan gøre vores programmering mere objektorienteret da vi her kan sætte et objekt op som vi kan tilskrive egenskaber og funktioner, det har vi oprettet i en header. Det gør at vi med en class som f.eks. "Class arm" kan lave en kommando der tilskriver arm en forward funktion som vi selv har skrevet og besluttet hvad gør.

Elektronik - indlejrede systemer

buck converters

en buck converter er et meget praktisk værktøj til at kontrollere hvor meget spænding der kommer igennem vores kredsløb men også til at sikre vi ikke brænder noget af, vi bruger en step down converter fra vores serie forbundede batterier til at gå fra 11.1v til 6v. Buck converteren består af en masse små komponenter, hovedsageligt en diode, switch, mosfet, variabel modstand og kapacitatorer som i samarbejde gør at vi kan styre vores udgangsspænding.

PWM

PWM er forkortelsen for pulse width modulation som er noget vi bruger til at styre de forskellige slags motorer, det giver en fordel i forhold til strøm forbrug og varmen som de producere da de ikke vil være tændt konstant. Vi får også muligheden for at styre hastigheden på vores DC-motorer, som gør vi kan lave mere avancerede køre funktioner. På servomotorerne bruger vi det til at styre positionen de holder sig på, et kort signal får os tættere på 0 og et langt signal tætttere på 180 grader.

servo motorer

en servo motor minder lidt om DC-motor, men i stedet for en hastighed der ændre sig efter PWM, så er det ændring af servo motorens position PWM'en bestemmer, sådan at hvis der er ingen strøm er servo motoren i 0, og hvis PWM'en er sat til 255 vil servo motoren sædde i 180 som er dens maks position. servo motorerne bruger vi på armen til at styre bevægelserne på armen.

kirchhoffs lov

kirchhoffs lov kogt helt ned i det simple siger egentlig bare at hvis jeg står ved et knudepunkt med 3 komponenter, så på den anden side af komponenterne hvor de 3 mødes vil min samlede strømstyrke på den anden side være det samme som min samlede strømstyrke da jeg gik ind i knudepunktet.

Projekt

I projekt har vi haft om læserskæring og 3D-tegning samt at printe 3D objekter, vi har lært at bruge prusaslicer, inkscape og fusion 360. inkscape bruger man til at tegne 2D tegninger man efterfølgende kan skære ud på MDF i 3 eller 6 milimeter, eller i plexiglas. Det har vi brugt til at lave f.eks. 2 plader til vores rover. Fusion 360 har vi brugt til at lave f.eks. en batteriholder til vores rover som kan tages af og på ved brug af velkro tape, det gør at vi kan flytte vores batterier frem og tilbage fra rover til rover. Prusaslicer skal man bruge til at få lavet koden som 3D-printeren printer efter, her kan man også lave små ændringer som farveskift igennem printet.

netværk

det eneste vi har haft om i netværk der kan anbringes på roveren er at vi har lært hvordan man sætter en web-server op på roveren der kan kontrollere GPIO pins trådløst.

Opgave 2 - Arm - Milepælsopgave

Komponentliste

rover fra milepæls opgave 1 4x servomotor 3D-printet arm 12x Jumpers

Beskrivelse af armen

Armen er bygget ud af plastik og printet nede i FABlab, det gør at vi selv har kunne bestemme hvor meget fyld der skal være inde i delene til armen sådan at de dele der er belastede kan være mere holdbare end de dele der ikke bliver tungt belastede. Derefter har vi samlet armen og påsæt 4 servo motorer, til at styre armen frem og tilbage. En til at dreje hele armen rundt og en til at åbne og lukke den griber der taget fat om objekter, alle motorerne er begrænsede til kun at kan dreje 180 grader.

Vi har haft nogen udfordringer angående vores arm da vi har 3 forskellige rovers og vi har skiftet i mellem dem som gør at der altid er noget der skal ændres og skiftes over. De første par 3D-print vi lavede havde også nogen små fejl som satte os en anelse bagud. Der har også været issues angående vores servomotorere, vi har generelt brugt meget tid på at få kontrol på styringen til dem hvilket vi ikke skulle have gjort da det automatisk styring på dem er irrelevant da vi kun skal styre dem via Joystick.

Er personligt ret stolt over at vi har nået så meget på armen, vi startede med at tro vi var bagud men har nu fået samlet ret godt op igen som gør at vi faktisk ikke længere er bagud.

milepæl 3

Opgave 3 - Joystick og fjernstyring - Milepælsopgave

Hvad lærte jeg i fagene?

Programmering

i programmering har vi lært hvordan vi får 2 esp'er til at kommunikere trådløst gennem koden og hvordan vi skriver kode der tager brug af alle ESP'ens processor kræfter og ikke begrænser det til at bruge halvdelen. vi bruger wi-fi til den trådløse kommunikation og tasks til at optimere vores kode.

elektronik

i Elektronik der har vi haft om hvordan man SMD-lodder sådan at vi har kunne sætte de små dele fast på vores controller og gøre brug af alle funktionerne PCB-boardet tilbyder i form af knapper og joysticks. vi har også lært yderligere om hvordan kommunikation i mellem de to esp'er fungere og grundprincippet bagved master-slave funktioner der kommunikerer igennem i2C-protokoller.

projekt

I projekt der har vi har haft om Moscow, som egentlig er en metode vi har brugt til at opstille egenskaber vi skal have, burde have, kunne have, og ikke kommer til at have, det hjalp os med at holde styr på de krav vi vil opfylde. Vi har også haft om risikoidentifikation som vi har brugt til at skabe overblik over hvad vi skal holde øje med at muligheder der kan ødelægge eller forlænge projektet. Der har også været om feedback og om hvordan man bruger forskellige slags feedback til at opnå forskellige former for læring.

Komponentliste

- 2x joystick
- 1x pcb board
- 2x SMD capacitor
- 2x SMD resistor
- 1x liligo T-display
- 6x jumper wires
- 2x Knapper

Beskrivelse af joystick samt fjernstyring

Basen af joysticket er et PCB som er konstrueret sådan at der kan være et joystick i venstre side og et joystick i venstre side samt et styk SMD kapacitor og et styk SMD-resistor for at fjerne elektrisk støj. Der er på PCB’et lavet sådan at vi kan lodde vores liligo fast og bruge den til at sende koden. Koden er opbygget med en reciever kode som er den ESP-32 der sidder på vores rover, den er sat til at modtage data som vi bruger til at styre de forskellige funktioner vores rover har. Koden er opbygget i tasks for at kan maximere processor kraften på vores ESP-32 og så gør det at koden er mere responsive og at vi kan styre forskellige funktioner ved at skifte task, og ikke vente på koden er kommet til det punkt. Den anden del af koden er vores sender kode som vi bruger til at sende info fra vores joystick til vores ESP-32. Infoen der bliver sendt, er aflæsninger fra joystickene som vi bruger til at kontrollere både køre funktioner og arm funktioner, der bliver også sendt information om de 4 knapper som der sidder på PCB'en, der er 2 integrerede knapper i joysticket og 2 knapper vi selv har monteret som alle 4 bliver brugt til at skifte mellem forskellige funktioner på roveren.

Kort gennemgang

Den controller vi har nu, kan styre roveren trådløst, både skifte mellem automatisk og manuel kørsel, samt skifte over til at styre armen trådløst. Den kommunikerer med roveren gennem et wifi signal der kun er rettet fra vores controller til vores rover. koden er delt op i flere bider og beskrevet herunder.

Kode

Reciever

i koden har vi flere forskellige headers hvor vi har oprettet structs og funktioner vi kan hente ind i vores main kode sådan at vi ikke skal skrive flere linjer kode hver gang, i reciever koden er det opbygget sådan at de forskellige egenskaber roveren har, altså armen, moterer og sensorer er opdelt i headers. Koden der er ansvarlig for at vores rover kan modtage dataen er skrevet i vores main kode. Det fungerer sådan at vi aller først har en struct der hedder "struct_message" som egentlig bare er en struct med data der modtages direkte fra senderen som har en struct der er helt ens, da det er et krav for at det fungere, derfra kan vi så hente de forskellige informationer fra joysticket. efter det bruger vi egentlig forskellige funktioner, der er en funktion til når der bliver modtaget data så kan vi se hvor meget data der bliver modtaget, det bliver vist i bytes, det er en funktion vi henter fra header biblioteket ved navn "esp-NOW". Det samme gør vi med wifi-funktionen som initalisere esp-NOW sådan at vi kan modtage data, den fortæller om der sker en fejl med initialiseringen. Den sidste funktionen angående wi-fi er den vi bruger til at holde koden i gang, som egentlig registrere hver gang at vi modtager noget og så smider det tilbage til "ondata receieve".

Dataen vi gerne vil hente ud fra den struct der bliver modtaget tager vi ud af structen ved at lave en "struct_message Mydata;" som gør vi kan kalde information fra structen ved f.eks. at sige "Mydata.joystick2" som så holder værdierne der bliver sendt fra joystick 2, det kan gøres med alt informationen vi modtager fra senderen.

Sender

sender koden er opbygget i noget af samme dur udover at her har vi skrevet alle funktionerne vi skal bruge i main koden og derefter kaldt på dem. Senderen har samme struct som recieveren, igen fordi at det er et krav de er ens. I senderen bruger vi det samme bibliotek altså "esp-NOW" biblioteket som allerede har de funktioner vi skal bruge for at sende data, her skal vi registrere MAC-adressen på den esp der sidder på roveren sådan at dataen kun bliver sendt til vores ESP og ikke bliver broadcastet til alle ESP'er i nærheden. vi har også her en sendata funktion som der er hentet fra ovennævnte bibliotek og egentlig har ansvaret for at sende dataen, funktionen fortæller os både hvis det lykkes at sende data men også hvis det fejler. Til sidst bruger vi igen en funktion til at lave et callback efter dataen er sent sådan at den bliver ved med at sende, altså den smider koden tilbage til send funktionen sådan at den sender igen.

Libraries

-- vi bruger librariet "esp-NOW" til de fleste af funktionerne vi skal bruge til at sende og modtage med, biblioteket "wifi.h" bruger vi kun til at sætte wi-fi mode op, som gør det muligt at sende frem og tilbage. #include <ESP-NOW.h> bruger vi til at opsætte den struct vi sender frem og tilbage og til de funktioner der er krævet for at sende det derhen og modtage det.

include <wifi.h> bruger vi til at opsætte en standard for hvordan wifi'et skal være sat op.

Metoder fra undervisningen

Vi har hovedsageligt fokuseret på hvordan at vi for koden simplificeret og gjort overskuelig, dermed har vi brugt funktioner der gør at vi kan nøjes med at gøre noget en gang og derfra kun skal kalde funktionen istedet for at skrive det flere gange. Vi har brugt structs hovedsageligt til at tilskrive værdier men også til at modtage og sende data trådløst, da det er et krav at vores struct vi sender er ens med structen på modtager siden, altså at data typerne er samme sted i structen, navnet på en bestemt integer eller string er ligegyldig, bare datatyperne er ens.

Udfordringer

Udfordringer vi har mødt

Vi har haft nogen udfordringer med at få vores rover til at styre armen smooth sådan at den ikke hakker rundt i de positioner vi gerne vil have den skal køre over i, samt udfordringer med at få integreret knapperne da vores styring af knap værdierne ikke har været som det skulle så at i starten skiftede de kun fra true til false når vi holdte knappen inde.

Udfordringer - hvorfor?

problemet med knapperne opstod helt klart fordi at vi har set forkert på switch-funktionen der er på de 2-joysticks, vi har set det som at man klikker for at ændre værdi, når man reelt kun skifter værdien mens knappen er holdt inden.

Løsninger

problemet med knapperne har vi først løst ved at bruge de 2 ekstra knapper der er på joysticket, og derefter har vi skrevet en funktion der holder styr på knap værdierne sådan at det fungerer optimalt til hvad vi gerne vil have.

problemet med at armen hakker rundt er vi ved at løse, første step til at gøre den bevæger sig pænt har været at lave restriktioner der gør værdierne holder sig indenfor 0-180 som er der armen kan bevæge sig, da det er 180-graders servoer vi bruger, derefter har vi kigget på vores pulsbreddetid og rettet den til. Nu kan vi bevæge 2 servoer pænt men slet ikke bevæge de 2 sidste, vi har tænkt os at prøve at oprette en ny task til de 2 sidste servoer, og se om det løser problemet.