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 wifi 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 funktioerne 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 kommunikere 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 konstueret 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'en 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 joystickkene 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 kommunikere 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 fungere 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 wifi 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 wifi mode op, som gør det muligt at sende frem og tilbage.

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 inde, n.

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 fungere 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 restrictions 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.