Universal Robot 10

# 2016/9/8

* Robotten flyttes fra robolab til bachelorkontore. Selve armen er meget tung. Robot-armens mekaniske funktioner virker, hvis man bruger dens indbyggede software.

# 2016/9/9

* Ethernet-kabel skaffes (lånt fra Heidi) som forbindelse til UR10. Da ingen i gruppen har en computer med en ethernet indgang, mangler der en convertor. *Mathias har en convertor liggende hjemme, som tages med næste gang.*

# 2016/9/12

* Ethernet til USB converter bringes til kontoret.
* Software fra tidligere bachelorprojekt-gruppe (Karsten, Andreas, Rasmus, beskrives som ’KAR’ fremadrettet) kigges igennem. KAR’s små test-projekter virker ikke, da der er referencer til biblioteker (.dll-filer) der ikke eksisterer. Fikser man referencerne til de nyeste biblioteker udleveret af KAR, vil man opdage at de små test-projekter refererer til funktioner og klasser der ikke eksisterer i de biblioteker. Der er altså en uoverensstemmelse ift. hvad programmet forventer og hvad vi har. *Der blev taget kontakt til Karsten fra KAR.*

# 2016/09/16

* Der er lånt et access-point fra skolen. UR10 forbindes med ethernet-kabel til denne, og vi andre kan nu forbinde til UR10 trådløst.
* Der kan pinges til UR10s ip, og der er hul igennem.
* Der blev fundet en .dll fil der hed ’Util[Konflikt]’. Det var åbenbart det gamle bibliotek som KAR-gruppen brugte til deres småprojekter. Med denne virker en af deres programmer (URPortListener)
* Dog er der stadig huller i softwaren på de interessante programmer, fx FullRotationProgramWPF.
* *Karsten fra KAR har sagt han kunne samle et lille projekt sammen der kunne være interessant for gruppen.*

# 2016/09/20

* I dag blev brugt på at hacke rundt i FullRotationProgramWPF (FRP)
* Programmet blev lappet sammen med alt det den havde brug for, for at virke.
* Programmet endte ud med at blive funktionelt. Der er stadig nogle få ting der mangler at blive justeret, men det ville være nye tiltag ift. programmet originalt var lavet til.
* Undervejs var der forvirring omkring hvilke enheder XYZ var, og hvor XYZ var på robotten, og hvad der egentligt blev rykket rundt med.
* Fx vil vi gerne kunne justere hastigheden på robotten, da den flytter sig meget hurtigt pt.
* Ting der blandt andet blev lært undervejs:
  + MODBUS er en protokol til seriel kommunikation. Klassen der findes i FRP er hentet ned fra nettet. Klassen er kaldt *modbusTCP*
  + Til at bruge MODBUS har KAR-gruppen lavet et simpelt interface. Denne klasse hedder *ModBus*
  + Når der læses fra robotten sker det gennem *URSocketListener* på port 30002 med en frekvens på 10Hz. Dette bruges til at vise den nuværende position og rotation af TCP på UR10.
  + En URPose (som er en klasse KAR-gruppen har lavet) er designet til at rykke robottens TCP i en position og rotation man ønsker. XYZ er i meter, rotation (RX,RY,RZ) er i radianer.
  + UR10’s koordinat-system er som følger, hvis man står bag den, altså modsat hvor touch-skærmen normalt sidder:
    - X er ’bredden’, dvs venstre og højre side. Venstre er negativ, højre er positiv.
    - Y er ’længden’, dvs frem og tilbage. Frem er positiv, tilbage er negativ.
    - Z er ’højden’, dvs op og ned. Op er positiv, ned er negativ.

# 2016/09/21

* Der blev oprettet et nyt WPF projekt der skulle forsøge at være en renere version af *FullRotationProgramWPF*, der ikke indeholder de unødvendige biblioteker.
* Dette program kaldes *UR10TCPController*
* I dette program er der mulighed for at ’steppe’ robottens position og rotation, i stedet for at man manuelt skal indtaste det.
* Programmet fetcher selv den nuværende UR10 pose.
* Programmet har snart mulighed for at sende konfiguration til UR10.

# 2016/09/22

* *UR10TCPController* kan sende konfiguration med ’Speed’ og ’Acceleration’ til robotten.
* Selv om at denne konfiguration kan hentes fra robotten igen, er det som om at de ikke bliver brugt.
* Robotten kører på sine egne ’programmer’; URP-filer
  + Her har KAR-gruppen lavet et program der læser registrene på modbussen.
  + X, Y, Z, rX, rY og rZ puttes i en vector
  + Der kaldes ’move(vector, acceleration=0.4, speed=1.0)’.
  + Acceleration og hastighed er altså hardcoded til 0.4 og 1.0.

# 2016/09/23

* Ændret GUI i *UR10TCPController* så den ikke viser en double som et kommatal (0,04) men som 0.04.
* I KAR-gruppens kode havde de sendt et ’konfigurations’-objekt som indeholdte en række parametre der blev skrevet i modbussens registre.
* De eneste vi er interesserede i pt er ’acceleration’ og ’speed’.
* Der formodes at ’speed’ så er maksimum hastighed.
* De blev skrevet på henholdsvis register 151 og 152.
* På UR10’s interface er det muligt at omskrive det program den kører (URP-fil), så den den læser register 151 og 152 og giver dem med i ’move’ metoden i stedet for de hardcodede værdier.
* Det resulterer i at vi nu kan styre position, rotation, acceleration og hastighed på UR10, og kan dermed gå videre til næste fase.

# 2016/09/26

* En specifik position kan sendes
* Man kan sende en ’path’ til robotten. Det er muligt at sende en ’cirkel’ til robotten den skal bevæge sig i.

# 2016/09/27

* Robotten går i en cirkel samt drejer hovedet mod midten af cirklen.
* Mellem TCP og det punkt den skal rette sig imod skal der findes ud af hvordan man drejer TCP i radianer i RX, RY og RZ.
* Robotten kan ikke blive ved med at dreje dens led for evigt. Den kan maks gå mellem -360 og +360 grader.
* Ikke en ny opdagelse, men der skal også tages forbehold for at robotten kan komme til at gå ind i sig selv. Det er ikke en triviel løsning der skal til der, tænkes der.