**Manus – ROS og Gazebo**

**Slide 1: Introduksjon**

I denne presentasjonen skal vi diskutere de viktigste aspektene med ROS (Robot Operating System) og Gazebo.

**Slide 2: Hva er ROS?**

ROS er ikke et operativsystem som navnet tilsier, men er et åpent kildekode-rammeverk for å bygge robotprogramvare. Det gir et felles operativsystemlignende lag for robotapplikasjoner. Det muliggjør maskinvareabstraksjon, enhetsdrivere, kommunikasjon mellom prosesser og mer. Viktigst av alt, det er åpen kildekode, noe som betyr at fellesskapet kontinuerlig forbedrer og utvider det.

**Slide 3: Hovedfunksjoner i ROS**

ROS har flere viktige funksjoner, noen av disse er:

* **Meldingsformidling:** Noder i ROS kommuniserer ved å sende «messages», som er strukturerte dataenheter. Noder kan kommunisere på to måter: enten via «topics» eller via «services».
* **Støtte for flere språk:** ROS-moduler kan skrives på alle språk som det finnes et klientbibliotek for.
* **Pakkehåndtering:** ROS bruker et pakkehåndteringssystem for å organisere og dele kode.
* **Modularitet:** ROS oppmuntrer til en modulær tilnærming til programvareutvikling, noe som gjør det enklere å arbeide med store og komplekse robotssystemer.
* **Støtte fra fellesskap:** Et stort og aktivt fellesskap sørger for at ROS stadig utvikler seg og støttes.

**Slide 4: ROS-arkitektur**

La oss se på arkitekturen til ROS:

* **Nodes:** Dette er «prosesser» som utfører beregninger. De kommuniserer gjennom «topics» eller «services».
* **Topics:** Representerer en måte for noder å kommunisere.
  + En node kan sende ut informasjon ved å «publishe» den til et gitt «topic».
  + En annen node kan hente informasjon av interesse ved å «subscribe» på en gitt «topic».
  + Vi kan ha:
    - Flere «publishers» og «subscribers» på et gitt «topic».
    - En enkelt node som publiserer/abonnerer på et «topic».
* **Service:** Brukes for synkrone meldingstransaksjoner.
  + En service er definert av
    - et strengnavn, og
    - Et par «messages» – en for forespørsel og en for svar.
  + I motsetning til «topics», brukes services til en-til-en kommunikasjon.
* **Messages:** Kan sendes via «topics» eller «services». De er språknøytrale (f.eks. node implementert i C++ sender «messages» til en node implementert i Python).
  + Svært korte tekstfiler beskriver feltene til hver beskjed
* **ROS Master:** Er en oppslagsmekanisme som lar prosesser (noder) finne hverandre under kjøring.

**Slide 5: Jobbe med ROS**

Hvordan jobber du med ROS?

* Start med å opprette en ROS-workspace for å organisere prosjektene dine.
* Utvikle ROS-pakker for å dele opp funksjonalitet.
* Innenfor pakkene oppretter du noder som utfører spesifikke oppgaver og kommuniserer med hverandre ved hjelp av «topics» og/eller «services».

**Slide 6: Gazebo-introduksjon**

Nå over til Gazebo, det er en åpen kildekode 3D-robot-simulator med flere bruksområder. Gazebo gir et virtuelt miljø der du f.eks. kan teste og utvikle robotens evner før du implementerer dem i den virkelige verden.

**Slide 7: Hovedfunksjoner i Gazebo**

Gazebo har imponerende funksjoner:

* **Fysikkmotor:** Gazebo simulerer realistisk fysikk, slik at roboter kan samhandle med miljøet som de ville gjort i virkeligheten.
* **Sensor-simulering:** Den replikerer ulike sensorer som kameraer, lidar og GPS, noe som gjør den til et verdifullt verktøy for testing av sensorbaserte algoritmer.
* **Flere robotmiljøer:** Gazebo støtter simulering av flere roboter i samme miljø, noe som gjør den ideell for utvikling av flerrobotssystemer.
* **Åpen kildekode og plugin-system:** Som ROS, er Gazebo åpen kildekode, og den tilbyr et omfattende plugin-system for å utvide funksjonaliteten.

**Slide 8: Integrering av ROS og Gazebo**

En av styrkene til både ROS og Gazebo er deres integrasjon. ROS kan styre og samhandle med roboter som simuleres i Gazebo, noe som muliggjør omfattende testing og utvikling.

**Slide 9: Simulering av roboter i Gazebo med ROS**

For å simulere roboter med ROS og Gazebo:

* Opprett robotmodeller innenfor Gazebo, definer deres fysiske egenskaper og sensorer.
* ROS samhandler deretter med Gazebo, slik at du kan kontrollere og overvåke disse robotene innenfor det simulerte miljøet.

**Slide 10: Bruksområder**

Disse verktøyene har varierte bruksområder:

* De brukes innen autonom kjøretøyutvikling, industriell automatisering og forskning i akademia.
* ROS og Gazebo er avgjørende for trening og testing av AI-algoritmer for roboter.

**Slide 11: Utfordringer og Begrensninger**

Selv om de er utrolig verdifulle, har de utfordringer:

* **Resurskrevende simuleringer:** Å simulere komplekse miljøer kan være ressurskrevende.
* **Læringskurve:** Å mestre disse verktøyene kan kreve tid og innsats, spesielt for nybegynnere.

**Slide 12: Konklusjon**

Oppsummert er ROS og Gazebo viktige verktøy for robotikkmiljøet. ROS gir et rammeverk for å bygge robotprogramvare, mens Gazebo tilbyr et rikt 3D-simuleringsmiljø. Deres integrasjon gir et nyttig og kraftig verktøy innen robotutvikling og forskning.