Systemdokumentasjon

Unity integrasjon med SensMax TAC-B

IDATT2900-022

Erik Borgeteien Hansen,

Oda Alida Fønstelien Hjelljord

NTNU

Revisjonslogg

Dato	Versjon	Beskrivelse	Forfatter
02.05.2023	0.1	Begynt på kapittel 1	Erik
05.05.2023	0.2	Lagt inn diagrammer i kapittel 2, 3, 4	Erik
08.05.2023	0.3	Begynt på 3, 7, 9 og 10, gjennomgått kapittel 1 og 4	Alida, Erik
09.05.2023	0.4	Skrevet ferdig kapittel 5	Erik
21.05.2023	1.0	Fullført	Erik

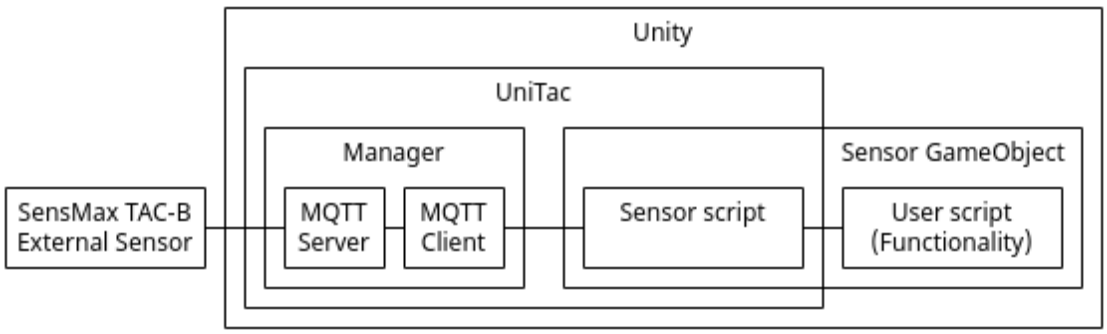
Innhold

1 Introduksjon

Dette dokumentet er skrevet i sammenheng med bacheloroppgaven til Oda Alida Fønstelien Hjelljord og Erik Borgeteien Hansen. Dokumentet beskriver og dokumenterer systemet som har blitt utviklet i prosjektperioden, og inkluderer beskrivelse av arkitekturen, strukturen, klassene, og den kontinuerlige integrasjonen og testingen til systemet. Det blir også skrevet om hvordan man går fram for å installere systemet, og hvilken avhengigheter dette krever.

2 Arkitektur

I figuren under ser vi de viktigste komponentene i systemet, og i hvilket domene de ligger. Kommunikasjonsflyten mellom de ulike komponentene er representert ved en svart strek.



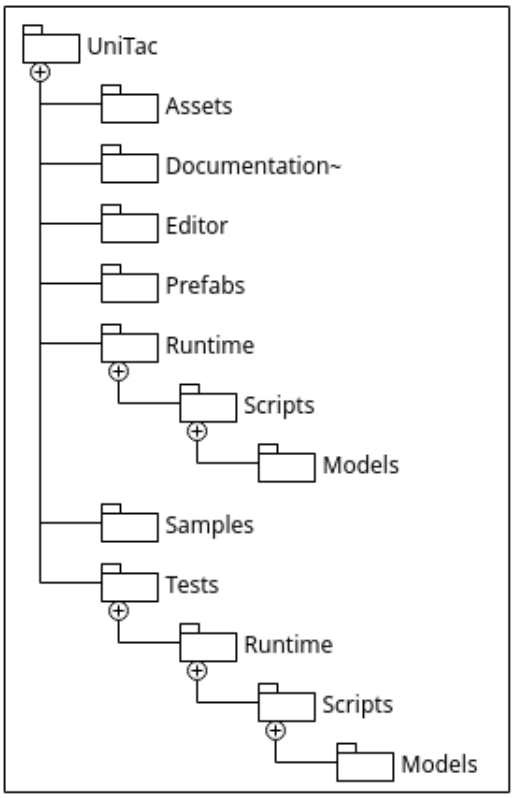
Figur 1: Den overordnede arkitekturen til systemet.

Den eksterne sensoren oppdager mennesker i synsfeltet sitt, og sender informasjon om dette som MQTT-pakker over wifi til MQTT-serveren. MQTT-serveren sender pakken videre til MQTT-klienten, som er konfigurert til å abonnere på alle pakker av denne typen. Både klienten og serveren er opprettet, konfigurert og kontrollert av "Manager.cs", som er festet til et GameObject (spillobjekt) i Unity for å kunne kjøre.

Klienten har ansvar for å deserialisere dataen i pakkene fra JSON til et C#-objekt. Dette objektet blir sendt videre til "Sensor.cs", som også er festet til et spillobjekt for å kunne eksistere.

3 Prosjektstruktur

Under er en oversikt over mappestrukturen i prosjektet. Mapper merket med ∼ er ikke synlig i "Unity Editor"-vinduet.

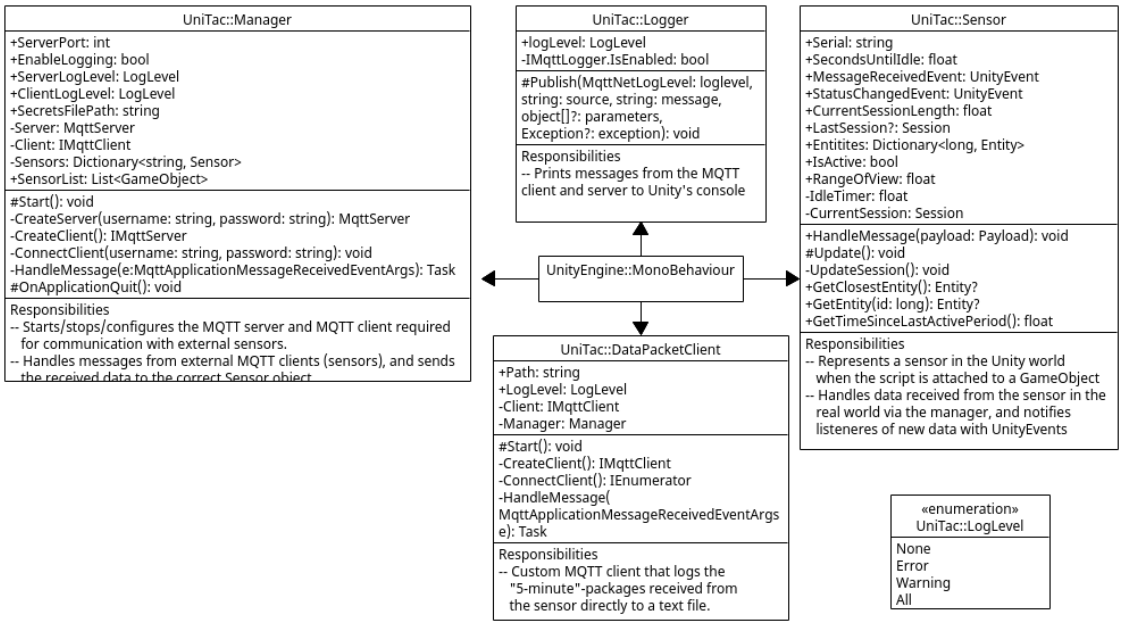


Figur 2: Oversikt over mappestrukturen i prosjektet

Editor-mappen har alle skript som endrer Unity sitt brukergrensesnitt. **Prefabs**-mappen har ferdilagde spillobjekter. **Runtime** inneholder alle script som brukes i kjøring av prosjektet. Disse skriptene er delt i to kategorier: modeller og monobehaviors. Modellene er datastrukturer som payload(desrialiserte fra MQTT-pakkene og MonoBehaviours er Unity-skript som kan sammenlignes med klasser med en main metode. **Sampels**-mappen inneholder all eksempelkoden.

4 Klassediagram

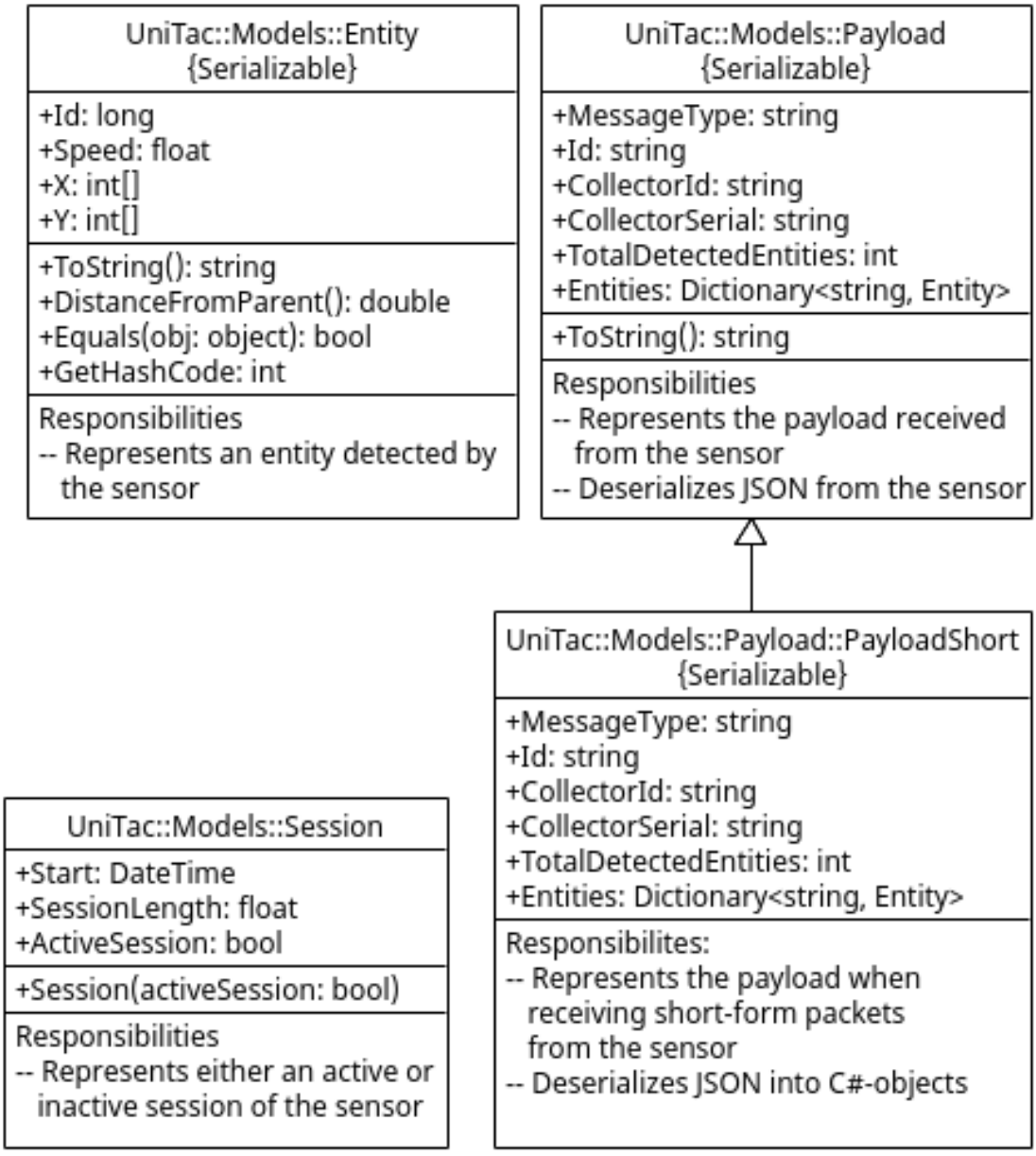
Som et resultat av konteksten systemet har blitt utviklet i, er det lav koblingsgrad i prosjektet. Integrasjonen mellom brukerkode og Unity sine systemer foregår via instanser av "GameObject"-klassen til Unity, Disse spillobjektene kan holde på komponenter som skript-filer, og gjør det mulig for koden vår å eksistere i Unity. For at et skript skal kunne legges på et spillobjekt må det arve fra "MonoBehaviour"-klassen, som medfører enkelte krav.



Figur 3: Klassediagram over koden i "Runtime"-mappen

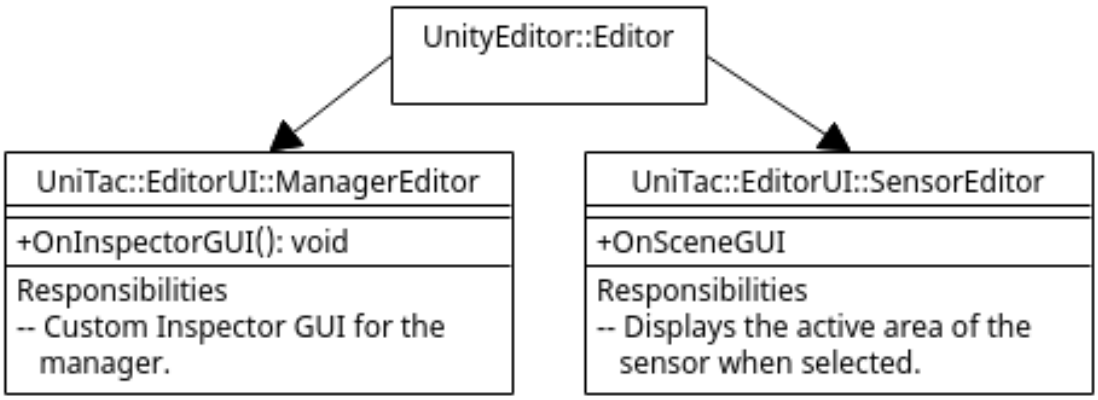
Manager administrer MQTT-kommunikasjonen for et helt prosjekt; både klienten som snakker med sensorspillobjektene i Unity og serveren som kommuniserer med den fysiske sensoren eksisterer her. Den tilbyr også UnityEvents som gir beskjed når ny data blir mottatt fra sensoren.

Sensor-klassen fungerer som en kontinuerlig oppdatert øyeblikksbilde av den virkelige sensoren den representerer. For eksempel vil Entities-variablen være en oppdatert oversikt over alle personene sensoren ser til en hver tid. Det er dette skriptet som tilbyr funksjonaliteten for pakken vår til endebrukeren. De kan enten kalle på metoder i skriptet for å bruke dataen, eller abonnere på UnityEvent-ene som blir sendt ut, og bygge sin egen funksjonalitet over dette.



Figur 4: Klassediagram for modellklassene

Payload- og Entity-klassene er basert på formatet som mottas fra sensoren. Session er en klasse laget for å organisere data fra sensorenes økter, blant annet lengde på økten og om det var en aktiv eller inaktiv økt. En økt starter hver gang sensoren skifter status mellom aktiv og inaktiv.



Figur 5: Klassediagram for editor klassene

Disse klassene er laget for å utvide Unity sitt brukergrensesnitt, for å gjøre pakken mer brukervennlig.

5 Installasjon og kjøring

5.1 Eksterne kodeavhengigheter

- **MQTTNet**: Brukes for å implementere MQTT-kommunikasjon. Tilbyr en klient og server. En kompilert versjon av denne kodeavhengigheten blir distribuert sammen med pakken vår. Dette er mulig da den benytter en ettergivende lisens kalt MIT-lisensen.
- **Unity.Newtonsoft.Json**: Brukes for å serialisere og deserialisere JSON.

5.2 Installasjonsveiledning

5.2.1 Forutsetninger

5.2.1.1 SensMax TAC-B sensor

Systemets funksjonalitet er helt avhenging av å motta data fra en SensMax TAC-B sensor. Uten en slik sensor vil ikke systemet ha noen funksjon. Det er derfor kritisk at man installerer og konfigurerer en sensor.

Mer detaljer rundt oppsett av sensoren, kan du lese i sensorens manual.

1. Montér sensoren på 2-3 meters høyde, eventuelt med en helningsgrad på 5-10°. Koble sensoren til strøm, slik at den slår seg på.
2. Koble til "wifien" som sensoren aktiverer. Deretter kan du åpne en nettleser og skrive inn "192.168.10.1" i adresse-feltet for å komme fram til sensorens webgrensesnitt.
3. Du kan logge inn i webgrensesnittet ved å skrive inn serienummeret til sensoren som passord.
4. Gå inn i innstillingene til sensoren øverst til høyre.
5. Under wifi-kategorien, skann etter nye nettverk. Når den er ferdig å skanne, koble til ett av nettverkene ved å velge det fra nedtrekksmenyen og å skrive inn passordet. Pass på at du kobler til samme wifi på enheten du planlegger å kjøre systemet på.

6. Trykk deg deretter videre på MQTT-kategorien i innstillingene. I serveradresse-feltet, skriv inn IPv4-adressen til enheten du skal kjøre systemet på senere. Fyll også inn ønsket port; 1883 er standard for MQTT.

7. Dersom du skal bruke passord og brukernavn, fyll også ut de feltene med ønsket passord og brukernavn.

8. Dersom du ønsker å motta pakker med posisjonshistorikk, velg den lange versjonen av pakken nederst på siden.

5.2.1.2 Unity

Systemet er en utvidelse til spillmotoren Unity, og krever dermed at du har installert denne.

1. Installer Unity Hub.

   For Windows og MacOS er dette så enkelt som å laste ned og kjøre installasjonsfilen herfra. Følg deretter instruksjonene på skjermen din.

   For Linux er prosessen noe mer innfløkt, se detaljerte instruksjon her.

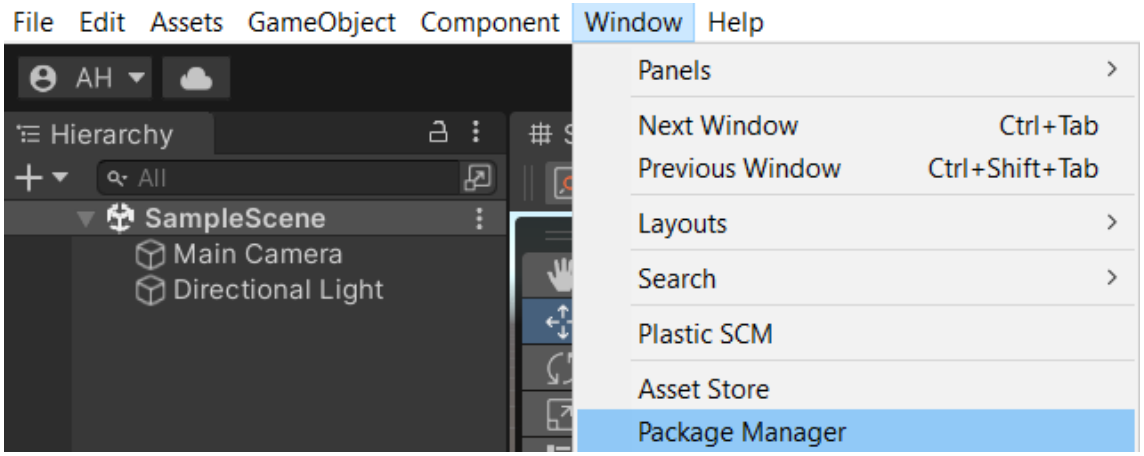
2. Åpne Unity Hub applikasjonen.

3. Logg inn.

4. Du vil du bli bedt om å installere en versjon av Unity Editor; dette er også påkrevd. Den nyeste langtidssupport-versjonen (LTS) er anbefalt, men systemet har blitt utviklet på versjon 2021.3.21f1 og å kjøre pakken på denne versjonen vil sikre at endringer i nyeste versjon påvirker kjøringen av pakken.

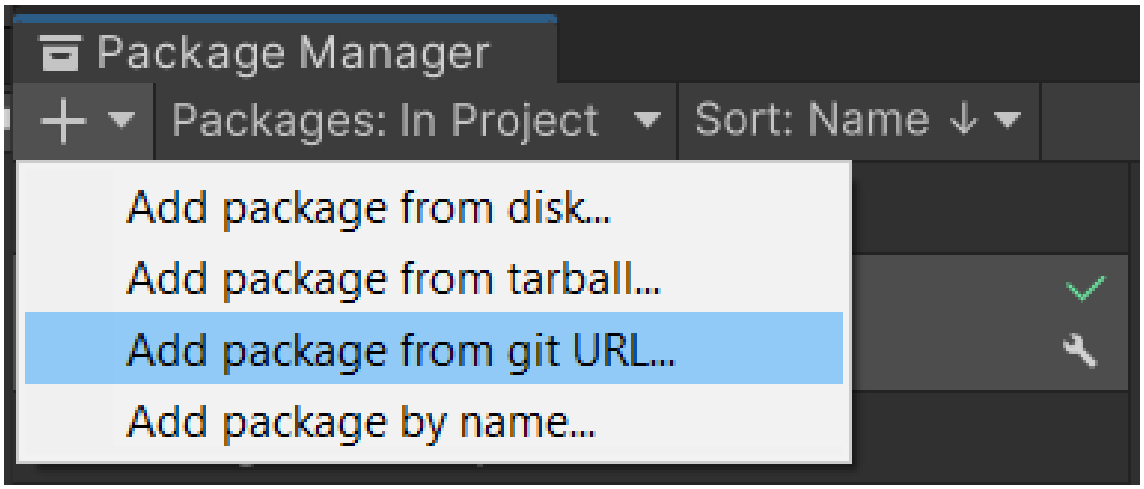
5.2.2 Installasjon

1. Åpne et Unity prosjekt fra Unity Hub, eller lag et nytt.
2. Åpne pakkebehandlervinduet ved å trykke på "Window", og deretter "Package Manager", som vist i bildet under.



Figur 6: Hvordan man åpner pakkebehandleren til Unity

3. Installer pakken via git; dette kan du gjøre ved å trykke på pluss-tegnet øverst til venstre i pakkebehandlervinduet, og deretter på "Add package from git URL". Du kan bruke både SSH og HTTPS for å klone prosjektet.



Figur 7: Kontekstmenyen hvor du kan legge til pakke fra git i pakkebehandlervinduet

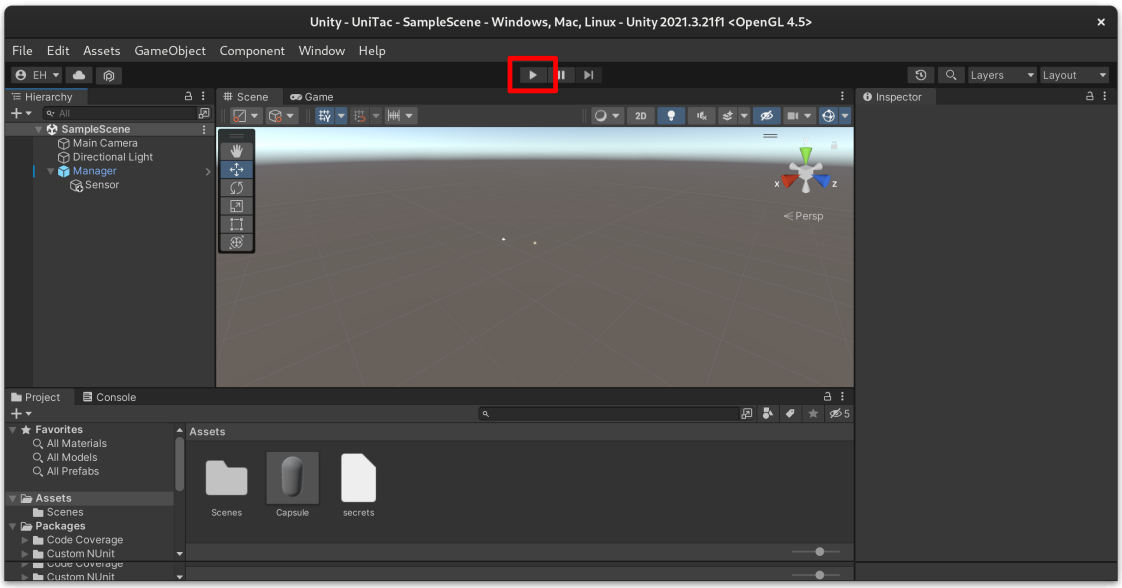
4. For å ha muligheten til å videreutvikle pakken er du nødt til å klone pakken via git. Trykk deretter på "Add package from disk..." og velg lokasjonen til den klonede pakken.

5.2.3 Kjøring

Da systemet ble utviklet som et utvidelse til Unity, var det ikke mulig å kjøre koden i en annen kontekst en et Unity-prosjekt. Derfor var det nødvendig å lage et Unity-prosjekt, og å ta i bruk pakken i prosjektet for å kjøre koden.

Etter å ha starter et prosjekt og installert pakken, kan du legge inn en manager-prefab og legge til en sensor. Deretter er du nødt til å konfigurere denne opp i mot den fysiske sensoren. Mer om dette kan leses i README-filen i dokumentasjonen eller på GitHub.

Du kan deretter bruke Unity sine standard metoder for å kjøre kode og tester.



Figur 8: Knappen som starter Unity-prosjektet er uthevet i rødt

6 Dokumentasjon av kildekode

Dokumentasjonen til prosjektet ble automatisk generert med verktøyet Doxygen ut i fra spesielle dokumentasjonskommentarer (spesielle kodekommentarer med enkelte XML-elementer) i kildekoden. Doxygen genererte en enkel interaktiv nettside i HTML, som ble lastet opp på GitHub Pages. Dokumentasjonen kan leses her.

Dokumentasjonskommentarene ble også vist i det integrerte utviklingsmiljøet som kodetips og automatisk kodefullføring.

Da prosjektet har benyttet seg av tredjepartskode ble det laget et dokument som beskriver lisensene til tredjepartskoden blitt skrevet. Dokumentet ble kalt "Third Party Notices.md", og ligger i mappen "Documentation∼". Det var mulig å lese dokumentet på dokumentasjonsnettsiden.

7 Kontinuerlig integrasjon og testing

7.1 Kontinuerlig integrasjon

Dokumentasjonen til prosjektet ble automatisk generert og rullet ut via GitHub sin plattform for kontinuerlig integrasjon og kontinuerlig utrulling (CICD); GitHub Actions. Arbeidsflyten til prosjektets CICD ble definert i filen ".github/workflows/gh-pages.yml", og hadde tre skritt; sjekk ut koden fra hovedgrenen, generer dokumentasjon fra kodekommentarene med Doxygen, og last opp den genererte dokumentasjonen til GitHub Pages. Arbeidsflyten kjørte på en virtuell maskin med Ubuntu hver gang ny kode ble sammenflettet med hovedgrenen, og sørget for at dokumentasjonen alltid stemte overrens med koden som lå i hovedgrenen.

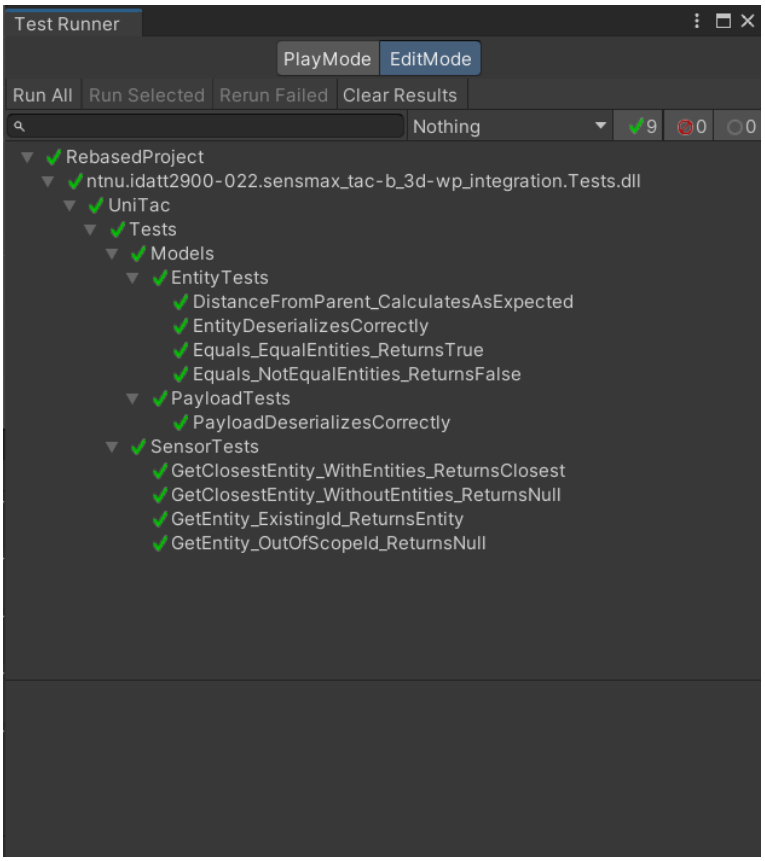
7.2 Testing

Alle metoder som kunne testes uten simulering av sensorinndata ble skrevet enhettester for. Alle modellklassene har en 100% testdekning. Av de øvrige klassene var bare sensor klassen uavhengig av simulering av sensorinndata og denne har 80% testdekning. Dette gir en total testdekning av 60%. Eksempelkode ble ikke skrevet tester for.

For å kjøre testene må pakken legges til et Unity-prosjekt og settes som “testable” i instillingene til prosjektet. Dette gjøres ved å redigere filen “manifest.json” i “Packages”-mappen i det lokale Unity-prosjektet. Legg til “testables”-feltet:

```
"testables": [ "edu.ntnu.idatt2900-022.unitac" ]
```

Deretter vil testene bli synlig i Unity Test Runner som kan åpnes gjennom "Window" → "General" → "Test Runner" i Unity editor. I dette vinduet kjøres testene ved å trykke på "Run All".



Figur 9: Unity Test Runner