SITREP #0

Im Folgenden der erste Statusbericht. Der SITREP *(situation report)* stellt dabei eine Möglichkeit dar zu verschiedenen Zeiten einen Überblick über den Stand des Projekts zu geben. Zudem zeigt er Schwierigkeiten bei der Umsetzung auf und dokumentiert die Arbeit, sowie das Vorankommen.

Problemstellung

Entspricht Anmeldung des Projekts.

Ziel der Studienarbeit ist eine Positionsregelung einer Drohne für die horizontale Ebene. Optional kann die Regelung auf alle 3 Raumrichtungen erweitert werden. Bei der Drohne handelt es sich um das Model Clover (COEX). Diese nutzt ROS als Betriebssystem.

Die Regelung kann in unterschiedlichen Verfahren entwickelt werden:

- A: Halten der Position nach einer manuellen Positionsänderung
- B: Anfliegen von vorgegebenen Positionen. Hierbei ist ein Überschwingen möglichst zu vermeiden.

Zu entwickeln ist eine Positionsregelung auf Basis der verfügbaren Beschleunigungswerte der Drohne. Optional kann die Regelung um ein bildgestütztes System erweitert werden. Gewünscht ist eine Versuchsbeschreibung für das RaHM-Lab zur Durchführung durch Studentengruppen. Diese Versuchsbeschreibung entspricht nicht der Bewertungsgrundlage der Studienarbeit. Die Ansprüche an das Projekt werden durch Mitarbeitende des RaHM-Labs in einem Lastenheft formuliert und dem Studierenden unterbreitet

Aktuelle Situation

- Drohne wurde aus Bausatz aufgebaut.
- Konfiguration der Sensoren via QGroundControl wurde durchgeführt.
- Raspberry Pie 4 wurde eingerichtet; Netzwerkverbindung ist möglich.
- Lastenheft bzw. Aufgabenliste wurde mit Herr Müller abgestimmt.
- Konzept für Regelungsarchitektur wurde erarbeitet.
- Auslesen von Sensordaten möglich
    - /mavros/state (allgemeiner Zustand; „armed“)
    - /rangefinder/range (Lasermessung Bodenabstand)
    - /mavros/imu/data (Beschleunigungs- und Lagedaten)
- Dokumentation
    - LaTeX Gerüst erstellen

Probleme

Probleme sollen als *Issue* (Label *Bug*) in git geführt werden.

- Spannungversorgung für LED-Streifen verpolt
- Fehlende Kommunikation von Funkfernsteuerung.
- Netzwerk-Einstellung der genutzten VM umständlich.
    - Host OS = Windows 10, betriebliche Vorgabe
    - Verbindung zu RosMaster benötigt NAT, Kommunikation zu RosTopics benötigt Bridge

Anstehende Aufgaben

Aufgaben sollen als *Issue* in git geführt werden.

- Git Issues entsprechend den hier genannten Aufgaben erstellen.
- Manuelle Flugversuche
    - Kommunikation zwischen Fernsteuerung und PX4-Controller herstellen
- ROS-gesteuerte Flugversuche
    - RosNodes (Publisher & Subscriber) implementieren

        - Keyboard-Subscriber (50%)
        - MavLink-Subscriber
            - Rangefinder Reader
            - Acceleration Reader
        - MavLink-Publisher
        - Bridge zur Application-Schicht
- Validieren der entworfenen Architektur mit Herr Strand und/oder Herr Müller
- Implementieren der Architektur
    - Domain- Schicht (80%)
    - Application- Schicht
    - Adapter- Schicht
    - Plugin- Schicht
    - Einbindung externer .cpp und .h Dateien in catkin_make bzw. CMakeLists.txt ermitteln
    - Einlesen in Verfahren zur Signalglättung (ggf. ist ein Ringspeicher für Daten ausreichend)
- Implementierung von Software-Tests
- Dokumentation
    - Problemstellung aus Projektanmeldung übernehmen und ggf. ausformulieren
    - Coex Clover beschreiben
    - ROS beschreiben
    - Architektur beschreiben
- Kommunikation zwischen PX4-FlightController und Raspberry Pie 4 via UART (optional)

Entwurf Architektur

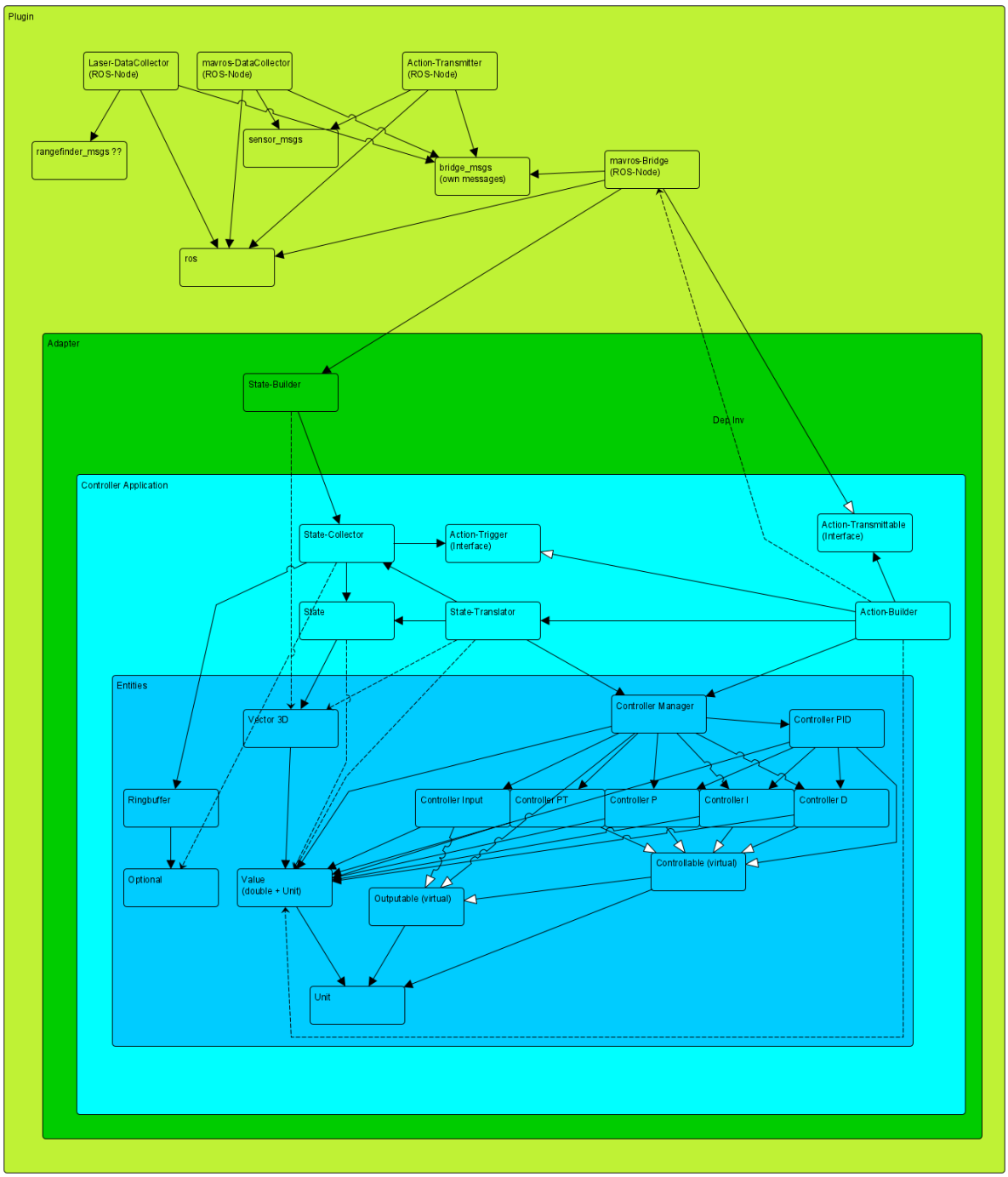
Die Architektur entspricht den Vorgaben der *clean architecture* entsprechend der Vorlesung Software Engineering 2. Entsprechend der vorangegangenen Absprache wird der Code dieser Studienarbeit als Grundlage für das Team-Projekt SWE2 genutzt.

In Absprache mit dem Dozenten der SWE2-Vorlesung soll die entstehende Regelung in der Application-Schicht und ROS und der Plugin-Schicht wiederzufinden sein. Für diese Realisierung wird eine Bridge zwischen ROS und der Positionsregelung (in c++ ohne Ros Nodes implementiert) benötigt.

Die Architektur wird derzeit noch verfeinert. Manche Abhängigkeiten innerhalb der Application-Schicht sind noch nicht abschließend geklärt. Vermutlich werden im Laufe des Projekts weitere Klassen hinzukommen.

Die Pfeile stehen für Abhängigkeiten (durchgezogene Linie => compile time dependency; gestrichelte Linie => runtime dependency). Ein Klassendiagramm wird daraus folgen.

Die Anordnung der Klassen und Pfeile wird in der abschließenden Ausarbeitung ordentlich aussehen ;)


Plugin
Laser-DataCollector (ROS-Node)
mavros-DataCollector (ROS-Node)
Action-Transmitter (ROS-Node)
rangefinder_msgs ??
sensor_msgs
bridge_msgs (own messages)
mavros-Bridge (ROS-Node)
ros
Adapter
State-Builder
Dep Inv
Controller Application
State-Collector
Action-Trigger (Interface)
Action-Transmittable (Interface)
State
State-Translator
Action-Builder
Entities
Vector 3D
Controller Manager
Controller PID
Ringbuffer
Controller Input
Controller PT
Controller P
Controller I
Controller D
Optional
Value (double + Unit)
Controllable (virtual)
Outputable (virtual)
Unit