D6.1

Drone use-case - Initial

Grant agreement no.	780785
Project acronym	OFERA (micro-ROS)
Project full title	Open Framework for Embedded Robot Applications
Deliverable number	D6.1
Deliverable name	Drone use-case - Initial
Date	June 2019
Dissemination level	public
Workpackage and task	WP6-Task 6.1
Author	Julian Bermudez Ortega (eProsima)
Contributors	Borja Outerelo Gamarra (eProsima)
Keywords	Robots, use-case, requirements, ROS, ROS2
Abstract	Report with detailed description for drone use-case.

Contents

1 Summary

This document provides a detailed description of the Autopilot Drone use-case sketched in D1.7 Reference Scenarios and Technical System Requirement Definition.

2 Acronyms and keywords

Term	Definition
UAV	Unmanned Aerial Vehicle
GCS	Ground Control Station
ROS	Robotics Operating System
MAV	Micro Aerial Vehicles
BLE	Bluetooth Low Energy
DDS	Data Distribution Service
DDS-XRCE	DDS for eXtremely Resource Constrained Environments
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ToF	Time of Flight
I2C	Inter-Integrated Circuit
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks

3 Introduction

Unmanned aerial vehicles (UAVs), known as drones, are one of the most complex environments in robotics due to their extremely restricted onboard computation resources. This fact is more remarkable in micro aerial vehicles (MAVs), vehicles smaller than 15 cm in any dimension [1], being their maximum payload just a few grams. The major constraint in MAVs is the power because the energy efficiency of batteries is an order of magnitude lower than the gasoline used in propulsion systems [2]. It involves the use of very low power processor (< 100 mW) in these vehicles.

Communication system, together with the propulsion system, is one of the most power demanding systems is a UAV. For that reason, the choice of this system in a key aspect in the UAV design. This issue becomes more important on MAVs due to the limited payload of these vehicles. In general, this kind of vehicles demands a communication system with low power consumption and weight at the cost of low bandwidth and range. Fortunately, there are several communication technologies which try to address this issue. Some examples of this kind of communication technologies are: Bluetooth Low Energy (BLE) [3], ZigBee [4] or 6LoWPAN [5]. In addition to the communication technology selection, the communication protocol is a decisive factor in the communication system design. The use of an appropriate communication protocol reduces bandwidth consumption, translating to greater efficiency.

Another important factor in power consumption is the software architecture of the UAV. Obviously, the more optimized the software is, the less power consumption, but there are other relevant aspects in the software architecture design. The software architecture should find an equilibrium point between efficiency and flexibility, that is, good software architecture should be easy to expand. A way of providing

flexibility is through the use of message-oriented middleware, being the publish-subscribe pattern the most common pattern in this kind of middleware [6]. Some examples of publish-subscribe pattern in UAV are [7] and [8]. In [7] a publish-subscribe middleware for interprocess communication in the onboard flight control software, called μORB, is present, while [8] describes a general publish-subscribe architecture which involves also communication with the GCS.

Taking into account the aforementioned, it is easy to realize that micro-ROS fits perfectly to *MAV* environments. Firstly, micro-ROS is targeted on low resource devices, such as the *MAVs* onboard one. Secondly, micro-ROS supports multiple transport protocol (UDP, TCP, Serial, 6LoWPAN, etc.) thanks to its communication middleware *eProsima Micro XRCE-DDS*. Finally, micro-ROS follows a publish-subscribe pattern inherited from ROS2.

4 Goals

This use-case tries to show the benefits of micro-ROS regarding its low resource consumption and its extensible and modular communication system. In particular, it is focused on the micro-ROS' middleware implementation *eProsima Micro XRCE-DDS*. This software, based on the DDS-XRCE (DDS for eXtremely Resource Constrained Environments) [9] wire protocol, offers to micro-ROS a client-server communication protocol with the following characteristics:

1. multi-transport protocol support (UDP, TCP and Serial),
2. a pluggable transport layer,
3. peer-to-peer communication,
4. server discovery,
5. reliable and best-effort communication,
6. message fragmentation.

Each one of the aforementioned characteristics will be used around this use-case.

5 Scenario

An *MAV* overflies a given area commanded by a *Flight Operator* through a *GCS*. The area contains *Remote Sensors*, which take environmental measures (temperature, pressure and humidity), distributed over it. The *Flight Operator* shall command the *MAV* toward the *Remote Sensors* and once positioned over them, the *MAV* shall establish a connection with the *Remote Sensor* in order to gather its data. Finally, the *Flight Operator* shall command the *MAV* toward the home position.

This use-case is based on the data gathering and processing scenario, commonly used in the commercial and industrial UAV markets. This scenario appears in different applications such as: crop management [10], [11]; environment measure and observation [12], [13]; and search and rescue [14]. Specifically, this use-case tries to emulate the environment measure and observation scenario, but due to flight restriction, it is reproduced in an indoor environment.

In this use-case, there are three different micro-ROS users, *MAV*, *GCS* and *Remote Sensors*, which interact in two different ways.

On the one hand, the link between the *GCS* and the *MAV* follows a client-server communication pattern. The *GCS* works as a server using a **micro-ROS Agent** application, while the *MAV* works as a client through a **micro-ROS Client** application. The connection between both is continuous during the entire mission and it is done over the proprietary crazyflie real-time protocol.

On the other hand, the link between the *MAV* and the *Remote Sensors* follows a peer-to-peer pattern. Both actors work as clients communicating through a lightweight micro-ROS Agent application, called **micro-ROS Agent lite**, running on the *Remote Sensor* side. This micro-ROS Agent lite application works as a centralized broker where the *MAV* and the *Remote Sensors* exchange its topics without ROS2 output and allows the *MAV* to discovery the *Remote Sensors* dynamically, In this case the communication is done over the BLE protocol.

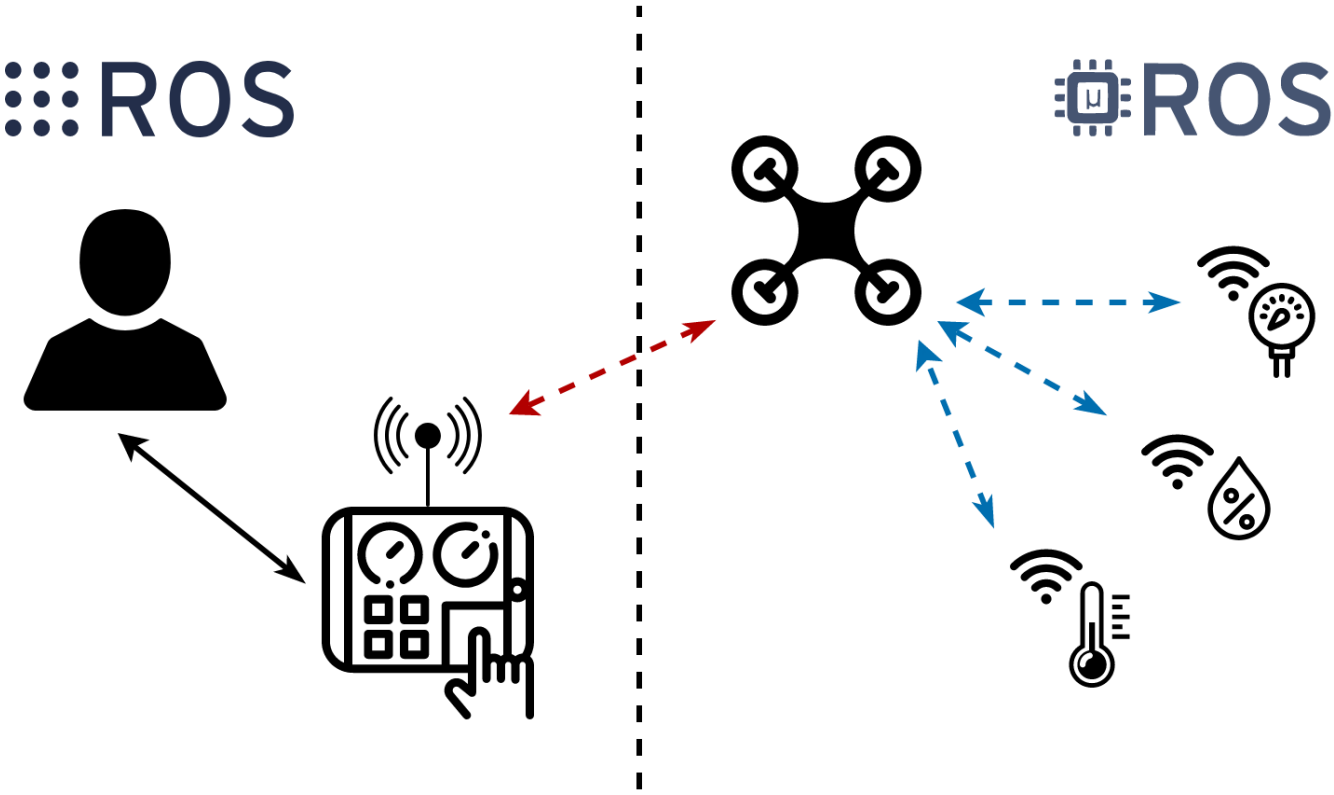

Figure 1: Use-case overview

The interaction between micro-ROS actors is performed through a set of topics:

- *CmdTopic* is published by the *GCS* in order to send flight command to the *MAV*. It is an enumeration type topic which contains a set of command inspired by the MAVLink communication protocol [15].
- *MAVStateTopic* is published by the *MAV* in order to send its state information to the *GCS*. It is an enumeration type topic also inspired by the MAVLink communication protocol.
- *MAVSensorTopics* is a set of topics published by the *MAV* and subscribed by the *GCS*. They are used to display the *MAV*'s state to the *Flight Operator*.
- *RemoteSensorTopics* is a set of topics published by the *Remote Sensors*. The *MAV* is in charge of gathering these topics from the *Remote Sensors* and publishing them in order to be received by the *GCS*.

```
enum CmdTopic
{
    takeoff,
    nav_to,
    start_gathering,
    stop_gathering,
    land
}

enum MAVStateTopic
{
    in_air,
    in_position,
    landed
}

/*
 * MAVSensorTopics
 */
struct Position
{
    float x,
    float y,
    float z
}

struct Attitude
{
    float roll,
    float pitch,
    float yaw,
}

/*
 * RemoteSensorTopics
 */
struct Temperature
{
    float temperature,
}

struct Pressure
{
    float pressure,
}

struct Humidity
```

```
{
    float humidity
}
```

The operative of this use-case can be summarized in the following steps:

1. The *Flight Operator* starts the mission.
2. The *GCS* publishes a `takeoff` *CmdTopic*.
3. The *MAV* receives the *CmdTopic*, takes off and publishes an `in_air` *MAVStateTopic*.
4. The *GCS* receives the *MAVStateTopic* and published a `nav_to` *CmdTopic* with the position of the *Remote Sensor*.
5. The *MAV* receives the *CmdTopic*, flies toward the *Remote Sensor* and once over it publishes an `in_position` *MAVStateTopic*.
6. The *GCS* receives the *MAVStateTopic* and publishes a `start_gathering` *CmdTopic*.
7. The *MAV* receives the *CmdTopic*, connects to the *Remote Sensors* using the discovery mechanisms, subscribes to the *SensorTopic* and publish a `gathering` *MAVStateTopic*.
8. The *Remote Sensor* publishes its *RemoteSensorTopics* data.
9. The *MAV* receives the *RemoteSensorTopics* and publish the data.
10. The *GCS* receives the *RemoteSensorTopics* published by the *MAV*.
11. The *GCS* publishes a `stop_gathering` *CmdTopic*.
12. The *MAV* receives the *CmdTopic* and publishes an `in_air` *MAVStateTopic*.
13. Steps 3-11 are repeated for each *Remote Sensor*.
14. The *GCS* publishes a `land` *CmdTopic*.
15. The *MAV* receives the *CmdTopic*, flies to home position, lands and publishes a `landed` *StateCmd*.
16. The *Flight Operator* finishes the mission.

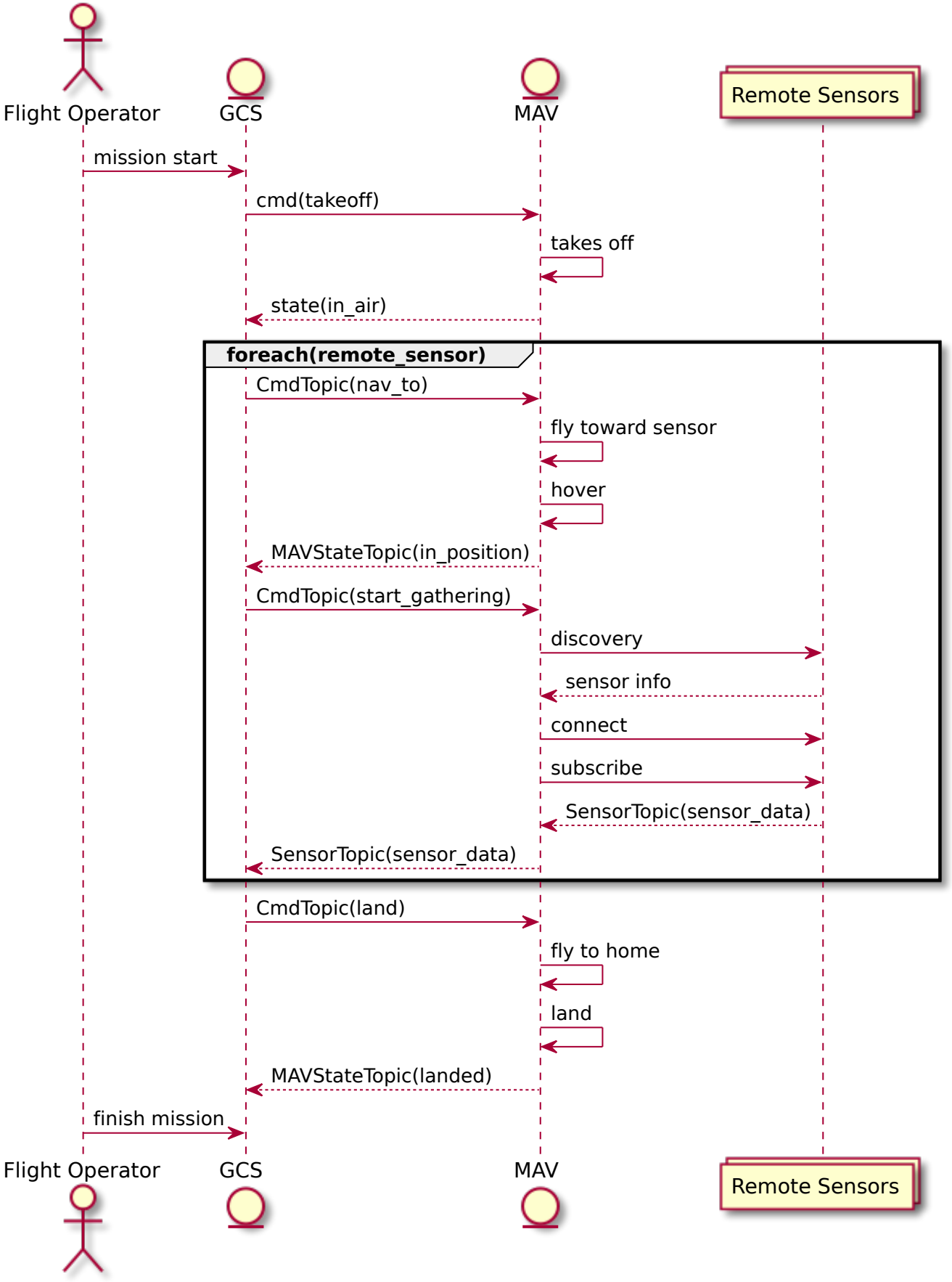

Figure 2: Use-case as sequence diagram

6 Actors

This section describes in detail the *Actors* involved in this use-case.

6.0.1 Flight Operator

The *Flight Operator* is mainly in charge of starting, canceling, finishing and supervising the mission, that is, he/she shall verify that the *MAV* is not experiencing any problems. This supervision is carried on thanks to the sensor's data that the *Flight Operator* received from the *MAV* (battery level, position, attitude, altitude, etc). In case of repetitive mission under well-known condition, the *Flight Operator* could be replaced by an artificial intelligence system.

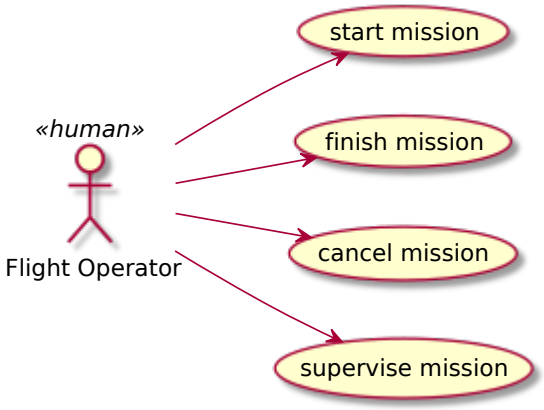

Figure 3: Use-case diagram of the Flight Operato

6.0.2 GCS

The *GCS* is composed by a general purpose computer together with a communication system and data link necessary to access the *MAV*. This actor is in charge of publishing *CmdTopic* data to the *MAV*, subscribing *MAVStateTopic*, *MAVTopics* and *RemoteSensorTopics* data from the *MAV*, as well as, showing this information to the *Flight Operator*. For it, the *GCS* shall run a **micro-ROS Agent** enabled application which provides communication with the *MAV*.

Regarding the communication system, the *GCS* uses the Crazyradio PA, a long-range open USB radio dongle based on the nRF24LU1+ chip with up to 2 km of range.

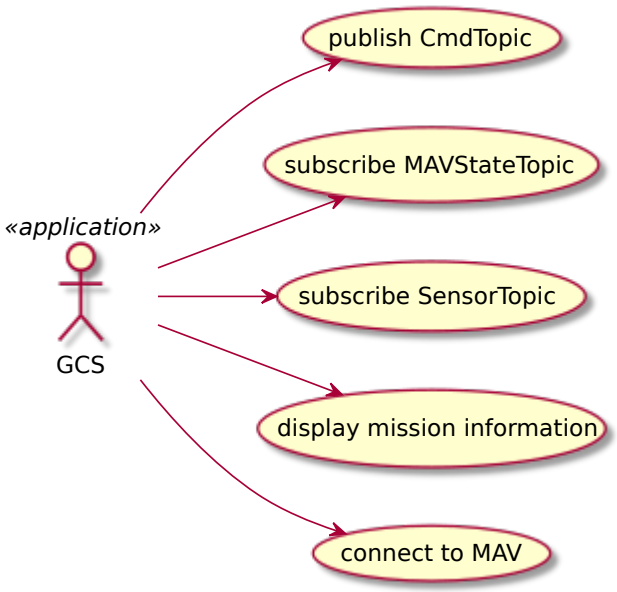

Figure 4: Use-case diagram of the GCS

GCS specifications:

- **Hardware**:
  - General purpose computer
- **Software**:
  - ROS2 application
  - Crazyradio PA

Figure 5: Crazyradio PA

6.0.3 MAV

The *MAV* is the main actor of this use-case. It is in charge of:

1. flying over the recognition area according to the mission planned by the *Flight Operator*;
2. connecting to the *GCS* subscribing to *CmdTopic* in order to receive flight command from the *GCS*;
3. publishing *MAVStateTopic* data of its sensors;
4. discovering the *Remote Sensors*;
5. connecting to the *Remote Sensors*;
6. subscribing to the *RemoteSensorTopics*;
7. and publishing the *RemoteSensorTopics* data received from the *Remote Sensors*.

For this use-case, the Crazyflie 2.1 has been selected as the reference platform. The Crazyflie 2.1 is a versatile open source flying development platform with only 27 g of weigh. It is equipped with low-latency/long-range radio as well as Bluetooth LE. Furthermore, the Crazyflie counts with multiple expansion boards, known as decks. Among these decks, the Multi-range and Flow v2 must be pointed out. These desks will be used for position estimation and collision avoidance.

The *MAV* has a flight control software which takes data from the *MAV* sensors and generates the control signal for the propulsion system. The software architecture of the *MAV* will be extended integrating a **micro-ROS Client** applications which will be in charge of connecting to *GCS* and *Remote Sensors* as well as publishing/subscribing *CmdTopics*, *MAVStateTopic*, *MAVSensorTopics* and *RemoteSensorTopics*.

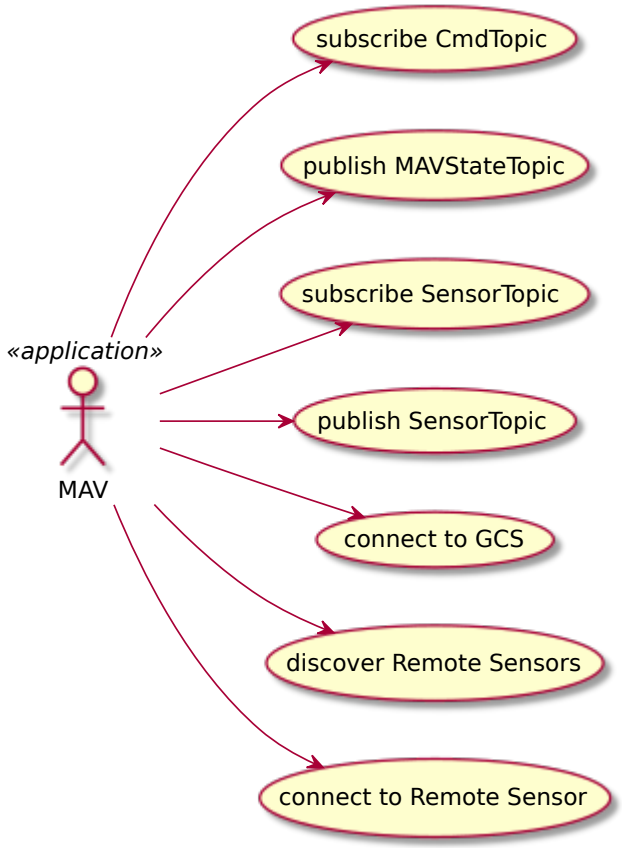

Figure 6: Use-case diagram of the MAV

MAV specifications:

- **Hardware**:
  - STM32F405 main application MCU (Cortex-M4, 168 MHz, 192 KB SRAM, 1 MB flash)
  - nRF51822 radio and power management MCU (Cortex-M0, 32 MHz, 16 KB SRAM 128 KB flash)
  - 3 axis accelerometer / gyroscope (BMI088)
  - High precision pressure sensor (BMP388)
  - 6 x ToF sensor (VL53L1x)
  - Take-off weight: 27 g
  - Size (WxHxD): 92x92x29 mm
- **Software**:
  - micro-ROS Client
  - PX4 autopilot

Figure 7: Crazyflie 2.1

6.0.4 Remote Sensors

Remote Sensors take environmental measures following a short sleep cycle. In particular, *Remote Sensors* measure temperature, pressure and humidity using the SparkFun Weather Shield. The board is connected to

a Raspberry Pi 3 Model A+ board over I2C serial protocol. The Raspberry Pi has a software application in charge of reading the sensors signals and connecting with the *MAV*. This application uses **micro-ROS Agent lite**, a reduced micro-ROS Agent version without ROS2 output, which shall establish a **peer-to-peer** communication with the *MAV*. The aforementioned peer-to-peer connection is done through the BLE protocol.

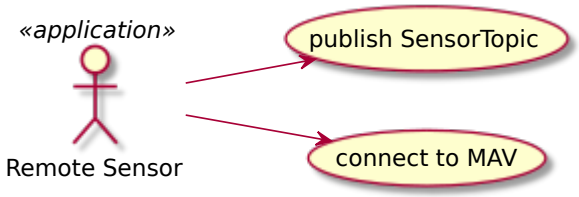

Figure 8: Use-case diagram of the Remote Sensors

Remote Sensors specifications:

- **Hardware**:
  - Broadcom BCM2837B0 (Cortex-A53 (ARMv8) 64-bit SoC @ 1.4 GHz)
  - 512 MB LPDDR2 SDRAM
  - 2.4 GHz and 5 GHz IEEE/802.11.b/g/n/ac wireless LAN, Bluethooth 4.2/BLE
  - Humidity/temperature sensor SI7021
  - Barometric pressure sensor MPL3115A2
  - Light sensor ALS-PT19
- **Software**:
  - micro-ROS Agent (lite)
  - Raspbian OS

Figure 9: Raspberry Pi 3 Model A+

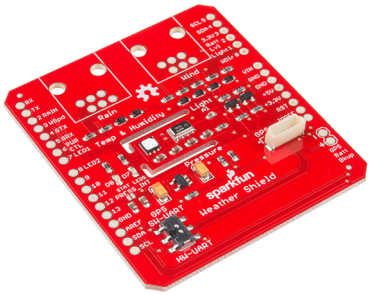

Figure 10: Sparkfun weather station

References

[1] R. C. Michelson, "Overview of micro air vehicle system design and integration issues," *Encyclopedia of Aerospace Engineering*, 2010.

[2] C. Michelson, "Issues surrounding communications with micro air vehicles," *Handbook of Unmanned Aerial Vehicles*, pp. 1415–1439, 2015.

[3] C. Gomez, J. Oller, and J. Paradells, "Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology," *Sensors*, vol. 12, no. 9, pp. 11734–11753, 2012.

[4] S. Farahani, *ZigBee wireless networks and transceivers*. Newnes, 2011.

[5] Z. Shelby and C. Bormann, *6LoWPAN: The wireless embedded internet*, vol. 43. John Wiley & Sons, 2011.

[6] C. Pippin, "Integrated hardware/software architectures to enable uavs for autonomous flight," *Handbook of Unmanned Aerial Vehicles*, pp. 1725–1747, 2015.

[7] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 ieee international conference on robotics and automation (icra)*, 2015, pp. 6235–6240.

[8] J. López, P. Royo, E. Pastor, C. Barrado, and E. Santamaria, "A middleware architecture for unmanned aircraft avionics," in *Proceedings of the 2007 acm/ifip/usenix international conference on middleware companion*, 2007, p. 24.

[9] "DDS for extremely resource constrained environments," Object Management Group, 2019.

[10] D. Anurag, S. Roy, and S. Bandyopadhyay, "Agro-sense: Precision agriculture using sensor-based wireless mesh networks," in *2008 first itu-t kaleidoscope academic conference-innovations in ngn: Future network and services*, 2008, pp. 383–388.

[11] C. Attene, "Sensors and drones: Hi-tech sentinels for crops." 2015 [Online]. Available: http://www.youris.com/Bioeconomy/Agriculture/Sensors-And-Drones-Hi-Tech-Sentinels-For-Crops.kl. [Accessed: 19-Jun-2019]

[12] J. Roldán, G. Joossen, D. Sanz, J. del Cerro, and A. Barrientos, "Mini-uav based sensory system for measuring environmental variables in greenhouses," *Sensors*, vol. 15, no. 2, pp. 3334–3350, 2015.

[13] B. A. White, A. Tsourdos, I. Ashokaraj, S. Subchan, and R. Zbikowski, "Contaminant cloud boundary monitoring using network of uav sensors," *IEEE Sensors Journal*, vol. 8, no. 10, pp. 1681–1692, 2008.

[14] T. Tomic *et al.*, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue," *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[15] L. Meier *et al.*, "Mavlink: Micro air vehicle communication protocol," *Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22]*, 2013.