THINK INDIA (Quarterly Journal)

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Review on MAV Link for Unmanned Air Vehicle to Ground Control Station Communication

Pratik PS
Department of Computer Science and Engineering
Bangalore Institute of Technology Bangalore, India
sankkipratik@gmail.com

Abhishek
Department of Electronics and Communication Engineering
Bangalore Institute of Technology Bangalore, India
vishek97@gmail.com

PrakharAgarwal
Department of Computer Science and Engineering
Bangalore Institute of Technology Bangalore, India
prakhar9837@gmail.com

Suneetha k r
Department of Computer Science
and Engineering
Bangalore Institute of Technology
Bangalore, India
suneetha.bit@gmail.com

***Abstract*—This paper briefs on MAVlink(Micro Air Vehicle Communication Protocol), a protocol used for communication between UAV (unmanned air vehicle), drone to GCS (ground Control station) and inter-communication of subsystems of the vehicle. It can be used to transmit the orientation of the vehicle, GPS location and speed. The protocol follows modern hybrid publish/subscribe and point-to-point design pattern. Data streams are sent as topics while configuration subprotocols are point-to-point. It implements multiple subprotocol for operator's ease to control. Messages are defined within a XML file. it is designed as a header-only message marshalling library. It has Support for many programming languages, running on numerous micro-controllers/operating systems (including ARM7, ATMega, dsPic, STM32 and Windows, Linux, MacOS, Android and iOS. Allows up to 255 concurrent systems on the network (vehicles, ground stations, etc.).it has been a very reliable protocol. MAVLink has been used since 2009 to communicate between many different vehicles, ground stations (and other nodes) over varied and challenging communication channels (high latency/noise). It provides methods for detecting packet drops, corruption, and for packet authentication.**

Keywords—Micro Air Vehicle (MAVLink); Global positioning System GPS , Publish/Subscribe; Point-to-Point(PPP); XML; Unmanned Air Vehicle(UAV)

I. INTRODUCTION

MAVlink or Micro air vehicle Link is a protocol for Communicating with small unmanned vehicle. MAVlink was first released early 2009 by Lorenz Meier under LGPL license.It specifies a set of messages that are exchanged between a small unmanned vehicle and a ground station.

II. GUIDE

A. Versions of MAVLink

MAVLink has 2 versions:

- **MAVLink v2.0** current version Adapted by many user from early 2017.
- **MAVLink v1.0** was adopted around 2013. Still used by a number of legacy peripheral.

MAVLink v2.0 has support for **C/C++** and **python** libraries which are also back word compatible with MAVLink v1.0. **Version Handshakin**g and **Negotiating Version** tell us which version is used.

Determining Protocol

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

- Autopilot version- .capabilities checked with MAV_PROTOCOL_CAPABILITY_MAVLINK2 flag to verify MAVlLink 2 support
- Protocol version- version contains version number multiplied by 100 eg: v2.3 is 230
- HeartBeat- .mavlink_version contains minor version number. This <version> field defined in Message Definitions.
- The major version can be determined from the packet start maker byte: MAVLink v1.0: 0xFE , MAVLink v2.0: 0xF

III. MAVLINK2

MAVLink 2 is a backward compatible update to the existing MAVLink protocol for security and flexibility to communication. It has been developed for C, C++ and Python.

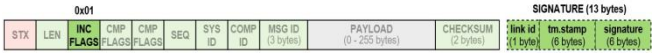
- 24 bit message ID - Allows over 16 million unique message definitions in a dialect (MAVLink 1 was limited to 256)
- Packet signing - Authenticates that messages were sent by trusted systems.
- Message extensions - Add new fields to existing MAVLink message definitions without breaking binary compatibility for receivers that have not updated.
- Empty-byte payload truncation - Empty (zero-filled) bytes at the end of the serialised payload are removed before sending (All bytes were sent in *MAVLink 1*, regardless of content).
- Compatibility Flags / Incompatibility Flags - Allow for backwards compatible evolution of the protocol by indicating frames that must be handled in a special/non-standard way (packets with compatibility flags can still be handled in the standard way, while packets with incompatibility flags must be dropped if the flag is not supported).

A. Message Singing

It allows the system to verify message originated from trusted source. The support for this was added by MAVLink2.

B. File Format

For a signed packet the **0x01** bit of the incompatibility flag field is set true and an additional 13 bytes of "signature" data appended.



The 13 bytes of the signature are:

Data	Description
linkID(8 bits)	ID of link on which packet is sent. Normally this is the same as the *channel*.
timestamp(48 bits)	Timestamp in 10 microsecond units since 1st January 2015 GMT time. This *must* monotonically increase for every message on a particular link. Note that means the timestamp may get ahead of the actual time if the packet rate averages more than 100,000 packets per second.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

signature(48 bits)	A 48 bit signature for the packet, based on the complete packet, timestamp, and secret key.

C. MAVLink2 Packet Format

Below is the over-the-wire format for MAVLink 2 packet.

Byte Index	C version	Content	Value	Explanation
0	`uint8_t magic`	Packet start marker	0xFD	Protocol-specific start-of-text (STX) marker used to indicate the beginning of a new packet. Any system that does not understand protocol version will skip the packet.
1	`uint8_t len`	Payload length	0 - 255	Indicates length of the following `payload` section. This may be affected by payload truncation.
2	`uint8_t incompat_flags`	Incompatibility Flags		Flags that must be understood for MAVLink compatibility (implementation discards packet if it does not understand flag).

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

3	`uint8_t compat_flags`	Compatibility Flags		Flags that can be ignored if not understood (implementation can still handle packet even if it does not understand flag).
4	`uint8_t seq`	Packet sequence number	0 - 255	Used to detect packet loss. Components increment value for each message sent.
5	`uint8_t sysid`	System ID (sender)	1 - 255	ID of *system* (vehicle) sending the message. Used to differentiate systems on network.
6	`uint8_t compid`	Component ID (sender)	0 - 255	ID of *component*sending the message. Used to differentiate components in a *system* (e.g. autopilot and a camera).
7 to 9	`uint32_t msgid:24`	Message ID (low, middle, high bytes)	0 - 16777215	ID of *message type* in payload. Used to decode data back into message object.
10 to (n+10)	`uint8_t payload[max 255]`	Payload		Message data. Depends on message type (i.e. Message ID) and contents.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

(n+11) to (n+12)	`uint16_t checksum`	Checksum(low byte, high byte)		X.25 CRC for message (excluding `magic` byte). Includes CRC_EXTRAbyte.
(n+12) to (n+26)	`uint8_t signature[13]`	Signature		(Optional) Signature to ensure the link is tamper-proof.

The minimum packet length is 8 bytes for acknowledgment packets without payload and maximum packet length is 263 bytes for full payload.

MAVLink does not include information about the message structure in the payload itself (in order to reduce overhead)! Instead the sender and receiver must share a common understanding of the meaning, order and size of message fields in the over-the-wire format.

Checksum

Define abbreviations and acronyms the first time they are used in the text, even after they have been defined in the abstract. Abbreviations such as IEEE, SI, MKS, CGS, sc, dc, and rms do not have to be defined. Do not use abbreviations in the title or heads unless they are unavoidable.

CRC_EXTRA Calculation

This is used to verify that the sender and receiver have an understanding of a particular message. The addition of CRC_BYTE to the checksum calculated by the user to ensure common understanding of the message definition.

Routing Detail

Systems/components process a message locally if any of these conditions hold:

- It is a broadcast message (target_system field omitted or zero).
- The target_system matches its system id and target_component is broadcast (target_component omitted or zero).
- The target_system matches its system id and has the component's target_component
- The target_system matches its system id and the component is unknown (i.e. this component has not seen any messages on any link that have the message's target_system/ target_component.

Systems forward messages to another link if any of these conditions hold:

- It is a broadcast message (target_system field omitted or zero).
- The target_system does not match the system id *and* the system knows the link of the target system (i.e. it has previously seen a message from target_system on the link).
- The target_system matches its system id and has a target_component field, and the system has seen a message from the target_system/target_component combination on the link.

General telemetry

QGroundControl and many other GCS support an older plain-text format for missions. This is not

Mavlink was designed to support continuous telemetry streams including position altitude and similar keys states of a drone.

Off-board control interface

It allows to send low-level and high level altitude, velocity or position commands to the vehicle.

File format

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

The mavlink systems store restore exchange MAVLink information from mission plans, geofencing definitions, rally point, parameter, logsetc.The standard used in GCS systems and developers APIs for storing information: plain-text file format.

IV. MICROSERVICES

The MAVLinkmicroservices define higher-level protocols that MAVLink systems can adopt in order to better inter-operate.

The microservices are used to exchange various types of data, including: parameters, missions, trajectories, images. Etc. If the data is far larger than that which can be fit into a single message, services will define how the data is split and re-assembled, and how to ensure that any lost data is re-transmitted. Other services provide command acknowledgment and / or error reporting.

Most of the services use the client-server pattern, such that the ground control system (client) initiates a request and the vehicle (server) responds with data.

The most common microservices are:

- Heartbeat/Connection Protocol
- Mission Protocol
- Parameter Protocol
- Command Protocol
- Camera Protocol
- Gimbal Protocol
- Arm Authorisation Protocol
- Image Transmission Protocol
- File Transfer Protocol (FTP)
- Landing Target Protocol
- Ping Protocol
- Path Planning Protocol (Trajectory Interface)

A. Heartbeat / Connection Protocol

The heartbeat protocol is used to determine whether a particular system is connected and also to detect when it has been disconnected.

MAVLink component considers itself connected to another system if it regularly receives the heartbeat message and disconnected if the expected messages are not received within the defined time period.

Therefore, to set up a connection every component must regularly broadcast its Heartbeat and monitor for heartbeats from other systems.

The rate at which the Heartbeat message must be broadcast and how many messages may be missed before the connection is considered to be broken depends on the channel alone.

B. Command Protocol

The MAVLink command protocol allows guaranteed delivery of commands.
It consists of the original command message and the matching acknowledgement (ACK).

C. Parameter Protocol

The Parameter microservice is used to exchange configuration settings between MAVLink systems.

The protocol guarantees delivery/reliable synchronisation of parameters for systems where the parameter set does not change during normal operation.

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

Each parameter is represented as a key/value pair. The key is usually the human-readable name of the parameter (maximum of 16 characters) and a value - which can be one of a number of types.

This key/value pair has a number of important properties:

- The human-readable name is small but useful.
- Unknown autopilots that implement the protocol can be supported-out of the box.
- A GCS does not have to know in advance what parameters exist on a remote system.
- Adding a parameter only requires changes to the system with parameters. A GCS that loads the parameters, and the MAVLink communication libraries, should not require any changes.

D. Mission Protocol

The mission b-protocol allows a GCS or developer API to manage mission, geofence and safe point information on a drone.

The protocol covers:

- Operations to upload, download and clear missions, set/get the current mission item number, and get notification when the current mission item has changed.
- Message type(s) for exchanging mission items.
- MAVLink commands that are common to most autopilots
  The protocol supports re-request of messages that have not arrived, allowing missions to be reliably transferred over a lossy link. The protocol follows the client/server pattern.
  MAVLink 2 supports three types of missions, namely:
- Flightplans:

1. NAV commands (MAV_CMD_NAV_*)for navigation/movement (e.g. MAV _CMD _NAV_WAYPOINT, MAV_CMD_NAV_LAND)
2. DO commands (MAV_CMD_DO_*) for immediate actions like changing speed or activating a server (e.g. MAV _CMD _DO _CHANGE_SPEED).
1. CONDITION commands (MAV_CMD _CONDITION_*) for changing the execution of the mission based on a condition - e.g. pausing the mission for a time before executing next command (MAV_CMD_CONDITION_DELAY).

- Geofence mission items: Prefixed with MAV_CMD _NAV_FENCE_ (e.g.MAV_CMD_NAV_FENCE _RETURN_POINT).
  1. NAV commands (MAV _CMD _NAV _*) for navigation/movement (e.g. MAV _CMD _NAV _ WAYPOINT,

- Rally point mission items:
  1. There is just one rally point MAV _CMD : MAV _CMD _NAV_RALLY_POINT.

E. Mission Camera Protocol

The camera protocol is used to configure camera payloads and request their status. It supports photo capture, video capture and streaming. It also includes messages to query and configure the onboard camera storage. Camera components are expected to follow the Heartbeat Protocol and send a constant flow of heartbeats (nominally at 1Hz). Each camera must use a different pre-defined camera component ID: MAV_COMP_ID_CAMERA to MAV_COMP_ID_CAMERA6.

The first time a heartbeat is detected from a new camera, a GCS (or other receiving system) should start the Camera Identification process.

->BASIC CAMERA OPERATIONS

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

The Camera information.lags provides information about camera capabilities. It contains a bitmap of CAMERA_CAP_FLAGS values that tell the GCS if the camera supports still image capture, video capture, or video streaming, and if it needs to be in a certain mode for capture, etc.

F. Gimbal Configuration Protocol

The gimbal configuration protocol is used to configure a payload mount and this is done by using a number of command and special-purpose messages. By default the gimbal should be communicating with the component ID MAV_COMP_ID_GIMBAL.

• COMMANDS TO CONFIGURE AND CONTROL GIMBAL
The two main commands to use mavlink gimbals are MAV _CMD_DO_MOUNT_CONFIGURE and

MA V_CMD_DO_MOUNT_CONTROL.

Another possible control command would be MAV _CMD _DO_MOUNT_CONTROL_QUAT. This command does not seem to be implemented by any systems at time of writing.

• COMMAND TO REBOOT OR SHUTDOWN GIMBAL
To reboot or shut down a gimbal send the command MAV_CMD_PREFLIGHT_REBOOT_SHUTDOWN.

The options to be set for the gimbal are found in param4.

G. Arm Authorisation

When enabled by setting a parameter on flight stack, the drone will only arm the motors if authorised by an external entity. This external entity is responsible for requesting any information that it needs from the drone and from other sources to authorise the arming procedure.

H. Image Transmission Protocol

The image transmission protocol uses MAVLink as the communication channel to transport any kind of image (raw images, Kinect data, etc.) from one MAVLink node to another. It basically takes a live camera image, splits it into small chunks and sends it over MAVLink.

The image streaming component uses two MAVLink messages:

- A handshake message, DATA _TRANSMISSION _HANDSHAKE, to initiate, control and stop the image streaming
- A data container message, ENCAPSULATED_DATA, to transport the image data.

I. Path Planning Protocol (Trajectory Interface)

The path planning protocol (a.k.a. trajectory interface) is a general-purpose protocol for a system to request dynamic path planning from another system (i.e. for an autopilot to request a path from a companion computer). The protocol is primarily intended for cases where constraints on the path to a destination are unknown or may change dynamically, but it can also be used for any other path management activities. Examples include: obstacle avoidance when following a pre-planned mission, determining paths for healing swarms, offloading geofence management to a companion computer, etc.

Field Name	Type	Units	Values	Description
angle_x	float	rad		X-axis angular offset of the target from the center of the image
angle_y	float	rad		Y-axis angular offset of the target from the center of the image
distance	float	m		Distance to the target from the vehicle
size_x	float	rad		Size of target along x-axis
size_y	float	rad		Size of target along y-axis

IMPLEMENTATIONS *EXAMPLE*

Obstacle Avoidance in PX4 Mission Mode

The protocol has been used to provide obstacle avoidance in PX4's mission mode. PX4 sends the current position, current waypoint, and next waypoint in a TRAJECTORY _REPRESENTATION_WAYPOINTS message (at 5Hz). The path planning software (a ROS node) sends set points for the duration of the mission. These navigate the vehicle in a straight line

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

to each waypoint, navigating around obstacles as needed.

J. File Transfer Protocol(FTP)

The FTP Protocol enables an FTP-like file transfer protocol over MAVLink. It supports common FTP operations like: reading, truncating, writing, removing and creating files, listing and removing directories. The protocol follows a client-server pattern, where all commands are sent by the GCS (client), and the Drone (server) responds either with an ACK containing the requested information, or a NAK containing an error. The GCS sets a timeout after most commands, and may resend the command if it is triggered. The drone must re-send its response if a request with the same sequence number is received.
All messages (commands, ACK, NAK) are exchanged inside FILE_TRANSFER_PROTOCOL messages. This message type definition is minimal, with fields for specifying the target network, system and component, and for an arbitrary variable-length payload.
The different commands and other information required to implement the protocol are encoded within the File Transfer Protocol payload. The payload format explains the encoding, packing format, commands and errors, and the order in which the commands are sent to implement the core FTP functionality.

K. Ping Protocol

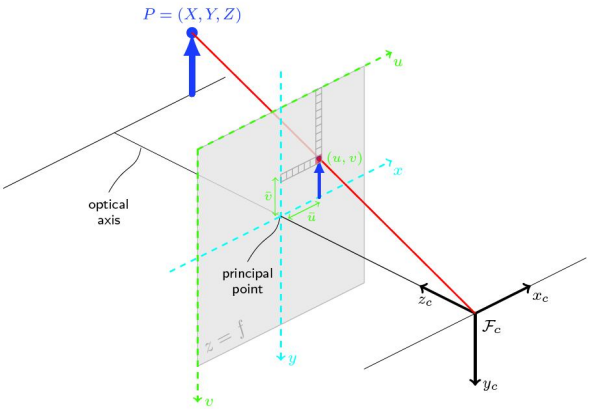
The Ping Protocol enables the system to measure system latencies on any connection: serial port, radio modem, UDP, etc.The PING protocol is implemented with just the Ping message. The message is send with a time stand and a sequence number that are returned by recipients, and can hence be used to determine the round trip time.

L. Landing Target Protocol

The landing target services/message communicates the position of one or more targets from MAVLink positioning system to an autopilot. A multicopter can use the message to land with far greater positional accuracy than provided by conventional GPS (GPS provides position within several meters while a landing- target system might reasonably provide centimetre-level precision landing).

A positioning system might typically consist of an onboard companion computer with a vision system that can detect a light beacon or target image. Radio beacons and different types of visual markers and tags are also supported.

TARGET RELATIVE TO CAPTURED IMAGE



The sizes in x and y direction are analogous (size_x/size_y). They describe the angle between the smallest and biggest pixel in x/y direction respectively of the target as seen in the image.

V. MAVLINK XML FILE SCHEMA / FORMAT

A. File Structure

```
<?xml version="1.0"?>
<mavlink>
```

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

```
<include>common.xml</include>
<include>other_dialect.xml</include>

<!-- NOTE: If the included file already contains a version
tag, remove the version tag here, else uncomment to
enable. -->
<!--<version>6</version> -->

<dialect>8</dialect>

<enums>
<!--Enums are defined here (optional) -->
</enums>

<messages>
<!-- Messages are defined here (optional) -->
</messages>

</mavlink>
```

The main tags for the above XML Document are listed below:

- include: This tag is used to specify any other XML files included in your dialect.
    - Typically dialect files will include *common.xml* as shown above.
    - You can include multiple files using separate tags.
    - The path to included files can be relative to your dialect file. Note however that the project tests only cover the case where dialects are in the same folder.
    - Nested include of files is not supported (only files specified in the top level include are imported).
    - When building, generator toolchains will merge/append enums in all files, and report duplicate enum entries and messages.
- version: The minor version number for the release, as included in the HEARTBEATmavlink_version field.
    - For dialects that include**common.xml** the tag should be removed so that the version from **common.xml** is used (version from top level file will be used if specified).
    - For private dialects you can use whatever version you like.
- dialect: This number is unique for your dialect. You should use: TBD
- enums: Dialect-specific enums can be defined in this block (if none are defined in the file, the block is optional/can be removed).
- messages: Dialect-specific messages can be defined in this block (if none are defined in the file, the block is optional/can be removed).

B. Enum Definition (enums)

```
<enum name="LANDING_TARGET_TYPE">
<description>Type of landing target</description>
<entry value="0" name="LANDING_TARGET_TYPE_LIGHT_BEACON">
<description>Landing target signaled by light beacon (ex: IR-LOCK)</description>
</entry>
<entry value="1" name="LANDING_TARGET_TYPE_RADIO_BEACON">
<description>Landing target signaled by radio beacon (ex: ILS, NDB)</description>
</entry>
<entry value="2" name="LANDING_TARGET_TYPE_VISION_FIDUCIAL">
<description>Landing target represented by a fiducial marker (ex: ARTag)</description>
```

ISSN:0971-1260
Vol-22-Issue-4-October-December-2019

```
</entry>
<entry value="3" name="LANDING_TARGET_TYPE_VISION_OTHER">
<description>Landing target represented by a pre-defined visual shape/feature (ex: X-marker, H-marker, square)</description>
</entry>
</enum>
```

The main enum tags/fields are:

- name: The name of the enum (mandatory). This is a string of capitalised, underscore-separated words.
- description (optional): A string describing the purpose of the enum
- entry (optional): An entry (zero or more entries can be specified for each enum)
- deprecated(optional): A tag indicating that the enum is deprecated.

REFRENCE

- MAVLink Developer Guide- https://mavlink.io/en/
- Data Analysis of MAVLink Protocol- http://ieeex- plore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8188563&isnumber=8188558
- Designing Purpose Built Drones For Ardupilot By Ty Audronis