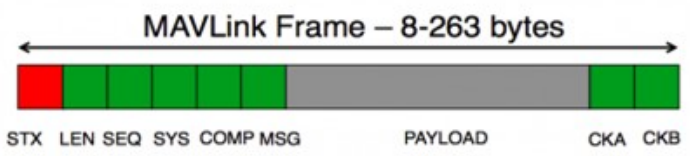
Les bases de MAVLink

MAVLink est un protocole série le plus couramment utilisé pour envoyer des données et des commandes entre véhicules et stations au sol.

Le protocole définit un grand ensemble de messages pouvant être trouvés dans les fichiers common.xml et ardupilot.xml
Les messages MAVLink peuvent être envoyés sur presque toutes les connexions séries et ne dépendent pas de la technologie sous-jacente (wifi, radio 900 MHz, etc.)
La livraison des messages n'est pas garantie, ce qui signifie que les stations au sol ou les ordinateurs doivent souvent vérifier l'état du véhicule pour déterminer si une commande a été exécutée.

I) Format d'un message MAVLINK



Byte Index	Content	Value	Explanation
0	Packet start sign	v1.0: 0xFE (v0.9: 0x55)	Indicates the start of a new packet.
1	Payload length	0 - 255	Indicates length of the following payload.
2	Packet sequence	0 - 255	Each component counts up his send sequence. Allows to detect packet loss
3	System ID	1 - 255	ID of the SENDING system. Allows to differentiate different MAVs on the same network.
4	Component ID	0 - 255	ID of the SENDING component. Allows to differentiate different components of the same system, e.g. the IMU and the autopilot.
5	Message ID	0 - 255	ID of the message - the id defines what the payload "means" and how it should be correctly decoded.
6 to (n+6)	Data	(0 - 255) bytes	Data of the message, depends on the message id.
(n+7) to (n+8)	Checksum (low byte, high byte)	ITU X.25/SAE AS-4 hash, **excluding packet start sign, so bytes 1..(n+6)** Note: The checksum also includes MAVLINK_CRC_EXTRA (Number computed from message fields. Protects the packet from decoding a different version of the same packet but with different variables).	

```
MAVPACKED(
typedef struct __mavlink_message {
    uint16_t checksum;      ///< sent at end of packet
    uint8_t magic;          ///< protocol magic marker
    uint8_t len;            ///< Length of payload
    uint8_t incompat_flags; ///< flags that must be understood
    uint8_t compat_flags;   ///< flags that can be ignored if not understood
    uint8_t seq;            ///< Sequence of packet
    uint8_t sysid;          ///< ID of message sender system/aircraft
    uint8_t compid;         ///< ID of the message sender component
    uint32_t msgid:24;      ///< ID of message in payload
    uint64_t payload64[(MAVLINK_MAX_PAYLOAD_LEN+MAVLINK_NUM_CHECKSUM_BYTES+7)/8];
    uint8_t ck[2];          ///< incoming checksum bytes
    uint8_t signature[MAVLINK_SIGNATURE_BLOCK_LEN];
}) mavlink_message_t;
```

II) Caractéristiques des messages :

- début de paquet:
  MAVLink 1: 0xFE
  MAVLink 2: 0xFD

-Les messages ne font pas plus de 263 octets

-L'expéditeur remplit toujours les champs ID système et ID composant afin que le destinataire sache d'où vient le paquet. L'ID système est un identifiant unique pour chaque véhicule ou station au sol. Les stations au sol utilisent normalement un identifiant système élevé tel que «255» et les véhicules utilisent par défaut «1» (cette valeur peut être modifiée en définissant le paramètre SYSID_THISMAV). Le numéro d'identification de composant de la station sol ou du contrôleur de vol est normalement “1”. Les autres périphériques compatibles MAVLink présents sur le véhicule (ordinateur compagnon, cardan, par exemple) doivent utiliser le même identifiant système que le contrôleur de vol mais utiliser un identifiant de composant différent.

-Le champ ID de message est visible dans les fichiers common.xml et ardupilot.xml en regard du nom du message. Par exemple, l'identifiant du message HEARTBEAT est "0",

-La partie Data du message contient les valeurs de champs individuelles envoyées .

III) Dialogue:

Une fois la connexion établie, chaque périphérique (ou «système») envoie le message HEARTBEAT à 1hz.

La station au sol ou l'ordinateur compagnon demande les données qu'il veut (et le débit) en envoyant des messages des types suivants :

-REQUEST_DATA_STREAM prend en charge la définition du taux de groupes de messages

-COMMAND_LONG contenant une commande SET_MESSAGE_INTERVAL permet de contrôler avec précision quels messages sont envoyés (et leur débit), mais n'est pris en charge que sur ArduPilot 4.0 et versions ultérieures.

Une station au sol ou un ordinateur envoie des commandes au véhicule.

IV) MAVLINK v1 vs MAVLINK v2 :

MAVLink2 étend MAVLink1 en permettant l'ajout de nouveaux champs aux messages MAVLink1 existants, prend en charge les nouveaux messages avec un ID de message supérieur à «255» et ajoute un support pour la signature de messages

-MAVLink2 est rétro-compatible avec MAVLink1, ce qui signifie que si un périphérique comprend les messages MAVlink2, il comprend certainement les messages MAVLink1.

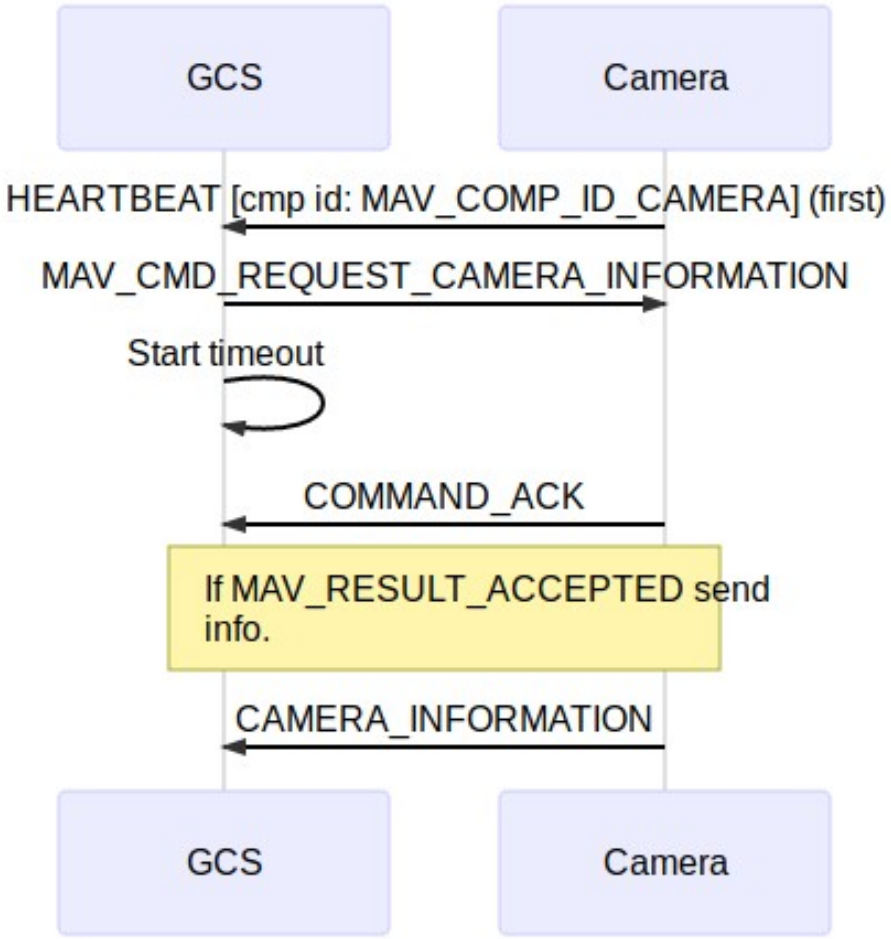
-Si un périphérique uniquement capable de comprendre MAVLink1 reçoit un message contenant des champs supplémentaires (ajoutés sous MAVLink2), il ne verra que les champs d'origine. C'est à dire. l'appareil sera capable de lire le message mais ne "verra" pas les champs supplémentaires

-Le port série d'un contrôleur de vol (probablement connecté à une radio de télémétrie) peut être configuré pour utiliser MAVLink2 en réglant le paramètre SERIALx_PROTOCOL sur «2» (où «x» correspond au numéro de port série du contrôleur de vol).

V) Protocole de communication entre une station de contrôle et une caméra :

1) Identification d'une caméra

La première étape pour établir une communication consiste à identifier une caméra. L'identification se fera en suivant les étapes indiquées sur le protocole ci-dessous :



Notes :

GCS = Ground Station Control

HEARTBEAT → C'est le premier signal nécessaire. Il permet d'indiquer à la station de contrôle que la caméra est disponible. Le protocole Heartbeat est utilisé pour annoncer l'existence d'un système sur le réseau MAVLink, ainsi que son ID système et composant, le type de véhicule, le type de composant et le mode de vol.

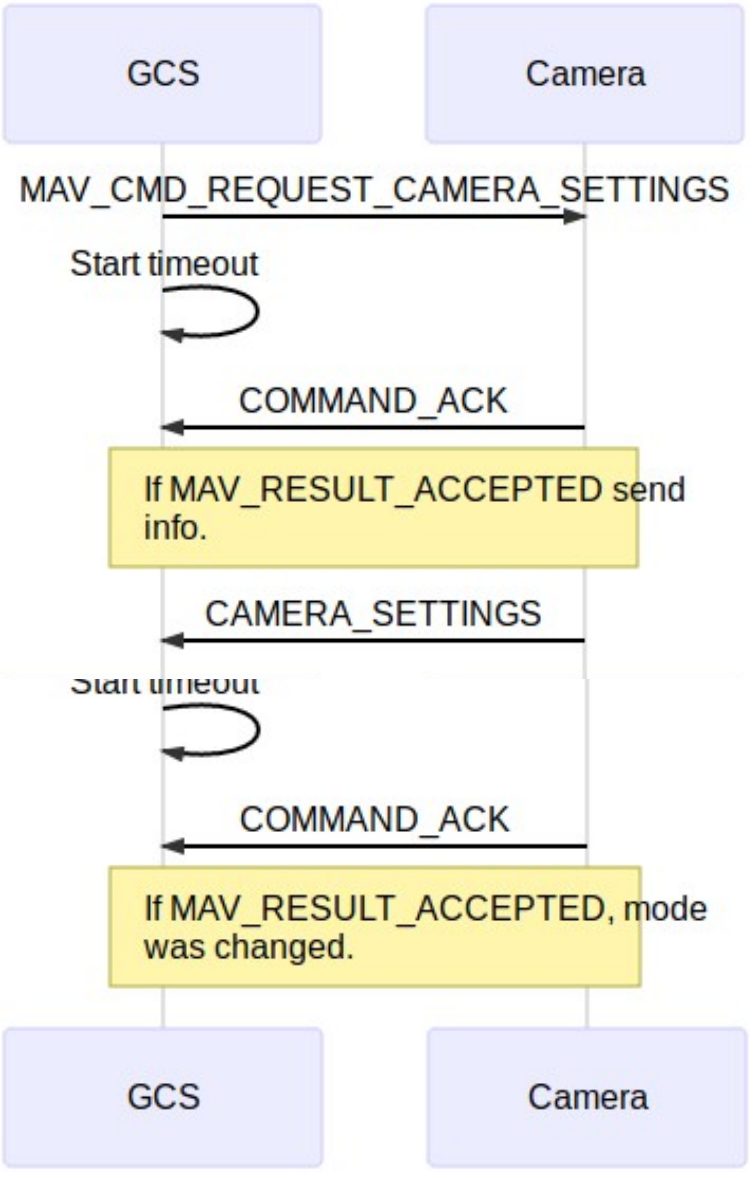
MAV_CMD_REQUEST_CAMERA_INFORMATION (521) → La station de contrôle envoie une requête à la caméra pour obtenir toutes ses informations techniques (marque, firmware, resolutions, définition etc …).

COMMAND_ACK → La caméra ayant reçu la requête y répond par le biais d'un acquittement dont le résultat est «MAV_RESULT_ACCEPTED ».

Enfin La caméra envoie ses informations techniques.

2) Mode caméra

Certaines caméras doivent être dans un certain mode pour capturer des images et / ou des vidéos.
La station de contrôle doit s'assurer que la caméra est dans le bon mode avant d'envoyer une commande de capture de début (image ou vidéo) en vérifiant si le bit CAMERA_CAP_FLAGS_HAS_MODES est défini sur « true » dans CAMERA_INFORMATION.flags.



3) Enregistrement vidéo

Une caméra prend en charge la capture vidéo si le bit CAMERA_CAP_FLAGS_CAPTURE_VIDEO est défini dans CAMERA_INFORMATION.flags.

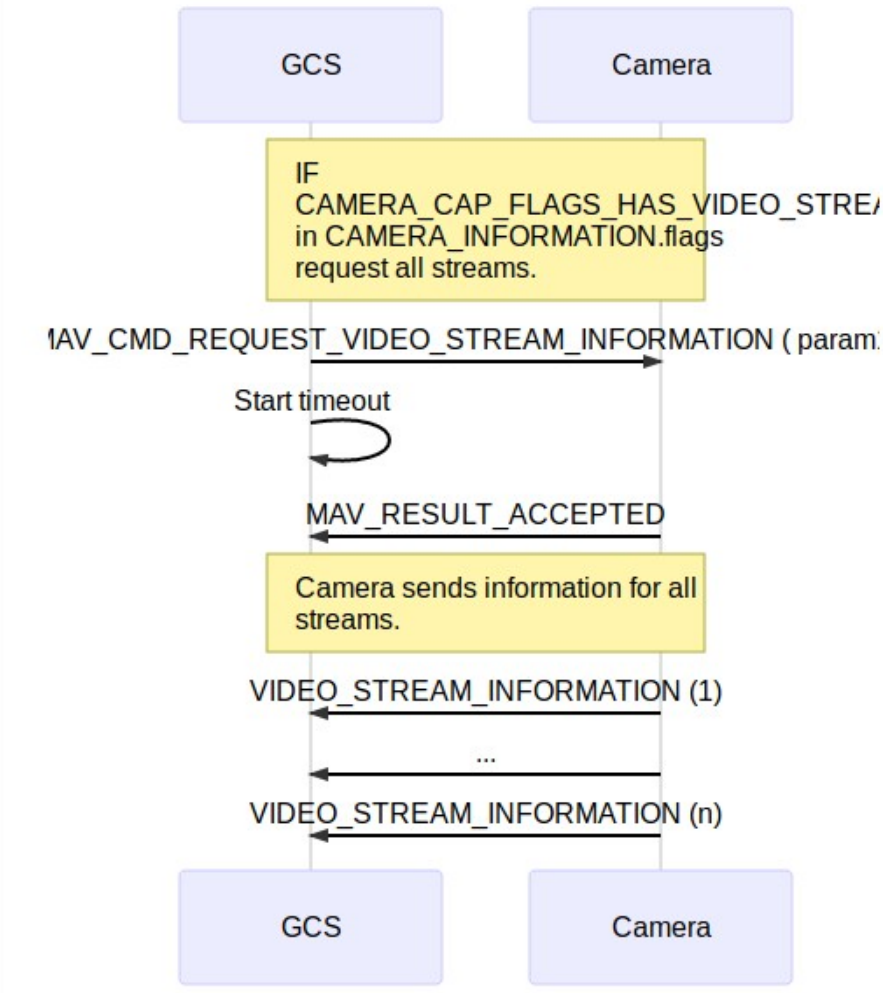
Pour commencer à enregistrer des vidéos, GCS utilise la commande MAV_CMD_VIDEO_START_CAPTURE. Si nécessaire, le message CAMERA_CAPTURE_STATUS est envoyé au système de contrôle mondial à un intervalle défini.

Pour arrêter l'enregistrement, le GCS utilise la commande MAV_CMD_VIDEO_STOP_CAPTURE.

3) Enregistrement vidéo (streaming)

Une caméra est capable de diffuser de la vidéo en continu si elle active le bit CAMERA_CAP_FLAGS_HAS_VIDEO_STREAM défini dans CAMERA_INFORMATION.flags.

La séquence de demande de tous les flux vidéo d'une caméra est indiquée ci-dessous:



VI) Mise en œuvre du protocole MAVLINK

1) Messages vs Commandes

Il existe deux méthodes pour envoyer des informations entre les systèmes MAVLink (y compris les commandes, les informations et les accusés de réception):

Les messages sont codés à l'aide d'éléments de message. La structure / les champs du message et leur traitement ne sont généralement pas soumis à des contraintes (c'est-à-dire au créateur).
Les commandes MAVLink sont définies comme des entrées dans l'énumération MAV_CMD **et sont codées en messages** réels envoyés à l'aide du protocole de commande. Leur structure est définie (ils ont 7 paramètres float et 2 int32_t) .

On utilise un message si:

Les informations requises ne rentrent pas dans une commande (c'est-à-dire qu'elles ne peuvent pas s'intégrer dans les 7 champs numériques disponibles).

Le message fait partie d'un autre protocole.

Le message doit être diffusé ou diffusé en continu (c.-à-d. Aucun ACK requis)

On utilise une commande si:

Le message doit être exécuté dans le cadre d'une mission.

Il existe un commandement de mission que vous souhaitez utiliser en dehors des missions. Selon le pilote automatique, vous pourrez peut-être traiter le message en utilisant le même code pour les deux modes.

Vous travaillez avec MAVLink 1 et il n'y a pas d'identifiant libre pour le nouveau message (MAVLink 1 a un pool d'identifiants libres beaucoup plus grand pour les commandes MAVLink que pour les identifiants de message).

Il est important que votre message de commande ne soit pas oublié. Un ACK / NACK est donc requis. L'utilisation des accusés de réception de protocole existants peut être plus rapide / facile que de définir un autre message pour les accusés de réception.

2) Envoie d'une commande

```
//  On renseigne les paramtères de la commande
mavlink_command_long_t camera_information; // création d'une structure contenant les paramètres de la commande
camera_information.command = MAV_CMD_REQUEST_CAMERA_INFORMATION ; // on rensigne l'ID de la commande souhaité,
// ici : MAV_CMD_REQUEST_CAMERA_INFORMATION
camera_information.target_system = system_id;// on rensigne l'ID du système
camera_information.target_component = component_id; // on rensigne l'ID du composant
camera_information.param1 = 1;// on rensigne les paramètres (ici param1 uniquement)
camera_information.confirmation = 0;

// On encode notre commande en message mavlink
mavlink_message_t message;
mavlink_msg_command_long_encode(1, 2, &message,&camera_information );

// On écrit le message sur le port série
int len = write_message(message);
```

exemple de mise en œuvre et de l'envoie d'une commande en C++

3) Envoie d'une message

```
// Configuration d'un message de type SET_POSITION
mavlink_set_position_target_local_ned_t sp; // on crée une structure pour
// renseigner les paramètres du message
sp.type_mask = MAVLINK_MSG_SET_POSITION_TARGET_LOCAL_NED_VELOCITY &
               MAVLINK_MSG_SET_POSITION_TARGET_LOCAL_NED_YAW_RATE;
sp.coordinate_frame = MAV_FRAME_LOCAL_NED;
sp.vx       = 0.0;
sp.vy       = 0.0;              // on renseigne les paramètres
sp.vz       = 0.0;
sp.yaw_rate = 0.0;
```

```
// ------------------------------------------------------------------------
//   ENCODE
// ------------------------------------------------------------------------
mavlink_message_t message;
mavlink_msg_set_position_target_local_ned_encode(system_id, companion_id, &message, &sp);

// ------------------------------------------------------------------------
//   WRITE
// ------------------------------------------------------------------------

// do the write
int len = write_message(message);
```

exemple de mise en œuvre et de l'envoie d'un message en C++

LIENS UTILES

Description des messages et commandes MAVLINK :

https://mavlink.io/en/messages/common.html#MAV_CMD_REQUEST_CAMERA_INFORMATION

Signature des messages :

https://mavlink.io/en/guide/message_signing.html

Protocoles caméras :

https://mavlink.io/en/services/camera.html

Installations des librairies :

https://mavlink.io/en/getting_started/

SOURCES

Site internet de MAVLINK : https://mavlink.io/en

Site internet de ARDUPILOT : http://ardupilot.org/