Micro Helicopter API: analysis

Cedric Pradalier

March 30, 2008

1 Introduction

This document provides a first analysis of an API for an easy to use micro-helicopter. It is intended as a discussion document. No conclusions are definitive, everything is arguable.

Most important, this has been written from a user point of view: it shows what I think a user would expect, without taking into account the reality of the controller implementation.

2 Modes

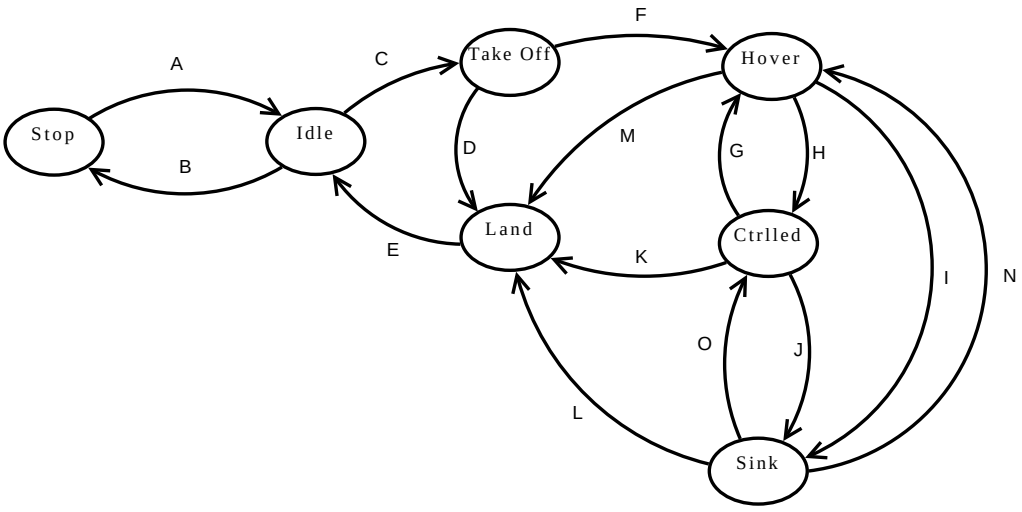

Figure 1: MAV API: Envisaged control modes

Figure 1 is a state automaton showing the various behaviour modes expected from a MAV. It is important to notice that these modes may be virtual modes: they don't have to correspond to what is really implemented in the controller, they describes various behaviours, semantically consistent and with the same set of constraints. The remaining of this section details the expected behaviour in each mode, the transition conditions and the accepted commands.

For the safety of the MAV, it is assumed that a control watchdog is implemented. This watchdog is triggered when the connection between the MAV and the host computer is broken, or if the host controller stop sending control commands. See section 3 for details.

2.1 Stop

2.1.1 Description

- System is checked up

- All the motors are powered off
- The pressure sensor reference is initialised

2.1.2 Transition conditions

Transition A : on user request.

2.1.3 Accepted commands

- Set Mode to Idle

2.2 Idle

2.2.1 Description

- All systems powered on
- All motors rotating slowly

2.2.2 Transition conditions

Transition B : on user request or on watchdog timeout

Transition C : on user request: mode set to Take Off or Hover

2.2.3 Accepted commands

- Set Mode to Stop, Take Off or Hover

2.3 Take Off

2.3.1 Description

- The MAV stabilise itself to hovering at 10cm above ground (TBC), based on pressure sensor. The resulting displacement is relative.

2.3.2 Transition conditions

Transition F : the system is stable at the required height.

Transition D : on user request, watchdog timeout, action timeout or exception.

2.3.3 Accepted commands

- Set Mode to Land

2.4 Land

2.4.1 Description

- Bring the MAV to touch down. Downward range sensor is used to estimate touch down. The maximum velocity downward is XX (TBD).

2.4.2 Transition conditions

Transition E : Contact has been detected.

2.4.3 Accepted commands

- None

2.5 Hover

2.5.1 Description

- Yaw and altitude are kept constant, roll and pitch kept to zero. The MAV will certainly be drifting in this mode.
- If available, the horizontal obstacle avoidance is active.
- Downward obstacle avoidance is active
- Data fusion for altitude estimation is active.

2.5.2 Transition conditions

Transition H : on user request

Transition I : on watchdog timeout

Transition M : on user request

2.5.3 Accepted commands

- Set Mode to Land
- Set Mode to Controlled

2.6 Controlled – Ctrlled

2.6.1 Description

- Altitude, yaw and position are controlled as requested
- Downward obstacle avoidance may be activated
- If available, horizontal obstacle avoidance may be activated

2.6.2 Transition conditions

Transition G : on user request or on control timeout (i.e. the watchdog has not been triggered, but no remote commands have been received).

Transition J : on watchdog timeout

Transition K : on user request

2.6.3 Accepted commands

- Set Mode to Hover
- Set Mode to Land
- Desired altitude, yaw and horizontal controls. In position or rate as available.

2.7 Sink

2.7.1 Description

- Altitude is slowly controlled down (for instance 5cm/s) until touch down
- Any desired control is ignored until the connection is reset (see section
- If available, the horizontal obstacle avoidance is active. 4).

2.7.2 Transition conditions

Transition N : on user request after connection reset.

Transition O : on user request after connection reset.

Transition L : on user request after connection reset, or when the distance to ground becomes smaller than 10cm (TBC).

2.7.3 Accepted commands

- Set Mode to Hover
- Set Mode to Land
- Set Mode to Controlled

3 Use Cases: from the programmer side

Several type of user programming can be foreseen. This section try to list them all. Not all of them may be implementable.

3.1 Considering only one helicopter

3.1.1 Polling, single thread

Set Communication Mode to Polling
Forever:

Get MAV State	*Polling*
Set Control Mode	*Reset WatchDog*
Set Control	*Z, Yaw, ...*

3.1.2 Polling, multi thread

Set Communication Mode to Polling

Thread 1	Thread 2
Forever: Get MAV State	Forever: Set Control Mode Set Control

3.1.3 Continuous, single thread

Set Communication Mode to Continuous
Forever:

Receive MAV State	*Server push*
Set Control Mode	*Reset WatchDog*
Set Control	*Z, Yaw, ...*

3.1.4 Continuous, multi thread

Set Communication Mode to Continuous

Thread 1	Thread 2
Forever: Receive MAV State	Forever: Set Control Mode Set Control

3.1.5 The video

In addition to the above options, the video stream can also be polled or received continuously, and it could (or should) be acquired in an independent thread.

3.2 Considering several helicopter

The type of MAV developed by FlyBotics are likely to be very suitable for multi-UAV control. This should be taken into account when designing the API, so that multiple UAV can be controlled from a single host computer.

4 Identified functions

Based on the above use cases, the following functions have been identified. They should all be implemented in a thread-safe manner. If not possible, it is always possible to specify that the user has to protect them appropriately.

4.1 System

4.1.1 Connect(MAV_id, Observer:bool)

- Connects to the MAV "MAV_id" as a controller or as a simple observer.
- Returns a handle describing the connection.

The assumption behind this function is that the MAV can only have one controller. On the other hand, we can design so has to allow multiple observer. Furthermore, after a timeout or an error in the communication, the communication must be reset before accepting new control. This can prevent chattering in case of a process too slow to control the MAV properly.

Each time a connection request is received, the connection id is increased by 1. As a result, a connection id can be

- Active: it is the id of the host controller currently controlling the MAV
- Inactive: it is not yet a valid id
- Revoked: it was a valid id, but has been revoked due to a disconnection or a timeout
- Observer: it is the id of an observer, probably 0 (TBC)

The resulting handle is composed on the MAV_id and the connection id.
The communication channel is currently expected to be Bluetooth.

4.1.2 Disconnect(handle)

- Close the connection and revoke the connection id
- No return

4.1.3 HasVideo(handle)

- Test if a video stream is installed on this MAV
- Bool

4.1.4 SetObstacleAvoidance(handle,vertical,horizontal)

- Enable or disable the obstacle avoidance control in vertical or horizontal direction if available.
- No return

4.1.5 SetCommunicationMode(handle,mode)

- Set the communication mode to polling or continuous
- No return

In polling mode, the MAV state is sent to the host controller only on request. In continuous mode, the MAV controller keeps pushing data to the host controller. The latter is responsible for collecting these data fast enough to keep the channel clear. Failure to do so should provoke a timeout and revoke the connection id.

The expected data rate is 50Hz, in polling or continuous mode.
See also GetState and ReceiveState functions

4.2 Video System

4.2.1 ConnectVideo(handle, ip)

- Open a connection to the MAV embedded computer so as to stream video data.
- Returns a handle on the video connection

The assumption here is that the embedded computer is accessible via a wireless lan on the given ip. The TCP/IP layer of the embedded computer will be used to stream the video frame.

4.2.2 DisconnectVideo(handle)

- Close the connection
- No return

4.2.3 SetVideoCommunicationMode(handle,mode)

- Set the video streaming mode to polling or continuous
- No return

In polling mode, the video frames are sent only on request. In continuous mode, the MAV embedded computer keeps pushing data to the host controller. The latter is responsible for collecting these data fast enough to keep the channel clear. Failure to do so should provoke a timeout and close the connection.

4.3 Data Collection

4.3.1 GetState(handle)

- Request the state of the MAV, only valid in polling mode
- Returns a structure describing the state as follows:
  - timeStamp:
  - handleState: Active, Inactive, Revoked, Timedout
  - commMode: Polling, Continuous
  - mavMode: id of an automaton state from section 2.
  - obstacleAvoidanceMode [vertical horizontal]
  - axisMode: control mode of the axis, see SetAxisMode.
  - errorFlags: TBD
  - Roll, Pitch, Yaw: from the IMU
  - Roll rate, Pitch rate, Yaw rate: from the IMU if available
  - Z, (Zrate if available): fused altitude estimation
  - Pressure
  - Battery
  - Proximeter [Down Left, Right, Front, Back]
  - RPM if available

4.3.2 ReceiveState(handle)

- Wait to receive the state of the MAV, only valid in continuous mode
- Returns a structure describing the state as GetState

4.3.3 GetVideo(vhandle)

- Request a video frame, only valid in video polling mode
- Returns the video frame, in a format to be specified.

4.3.4 ReceiveVideo(vhandle)

- Wait to receive the next video frame, only valid in video continuous mode
- Returns the video frame, in a format to be specified.

4.4 Control

4.4.1 SetMode(handle,mode)

- **Reset the WatchDog timer**
- Change the control mode as specified in the automaton in section 2. If the requested mode change is invalid in this context, it is ignored.
- Returns a flag indicating if the mode was accepted

The idea behind this function is that any user program should continuously call it to confirm its intention to take charge of the MAV. If not called at a high enough frequency, the MAV will slowly touch down, land and stop the rotors.

The watchdog function could also be an independent function, similar to the Pulse function on pioneers. TBD.

4.4.2 SetAxisMode(handle,yaw,altitude,horizontal)

- Defines the control mode of for each independent axis. Each control mode can be PositionControl or RateControl. Some of them may not be applicable currently and shall be ignored.
- Returns an acknowledgement

The expected default for the MAV control are:

- Yaw: position control, rate available
- Altitude: position control, rate available
- Horizontal: open loop rate control

4.4.3 SetControl(handle, yaw, altitude, horizontal

- In Controlled mode, set the desired yaw, altitude and horizontal control. The resulting control is affected by the AxisMode
- Returns an acknowledgement