Autopilot and companion computer for unmanned aerial vehicle: Survey

L. Delvaux[1], L. Di Naro[1]

[1]Haute Ecole de la province de Liège (HEPL), electronic and embedded systems, Liège Belgium

January 2023

***Abstract*— The popularity of drones has grown rapidly in recent years. Recently, more and more projects have been conducted by amateurs and academics. In this research, one element allows the drone to be able to stabilise itself and fly according to predefined trajectories: the autopilot. Of course, there are many different autopilots. This article is a study of the different existing autopilots. In this article, a comparison of the different autopilots that make up the Pixhawk autopilot range was conducted. In order to increase the capabilities of these autopilots, a companion computer can be added to the system. This companion computer allows a specific mission to be carried out. In the following, communication between the Pixhawk and the companion computer will be implemented via the MAVLink protocol.**

I. INTRODUCTION

Nowadays, public interest in UAVs (unmanned aerial vehicles) is growing steadily. A large number of projects are being developed from UAVs all around the world in many different fields. These range from UAVs delivery [1] projects to industrial safety [2], but also for military applications [3]. This increasing in the number of projects is mainly due to the democratization of UAVs.

There are many of off-the-shelf drones available on the market. These drones are often easy to use, even for someone without any experience. These UAVs are used by many photographers or are used to visualize an area. However, not all UAVs are capable of flying autonomously.

Whether it is a simple remote-controlled UAV or a fully autonomous UAV, electronic boards have been developed to make these machines more intelligent and easier to control. These boards, which are called "autopilot" are becoming more and more complex and incorporating more and more features. These systems are able to help or even completely replace the human pilot.

The autopilot [4] is a system consisting of software and hardware to control the UAVs (or any other vehicles that can be remoted). To do this, the autopilot uses different sensors such as an IMU (inertial measure unit), a barometer and sometimes a GPS. In addition, it can generate motor control signals and receive RF instructions.

The computational capabilities of the autopilot alone limit the possible application of the drone. Therefore, it is interesting to add a companion computer [5] that will make the UAV more flexible. This means that the drone can be programmed accordingly to suit the requirements of the user.

The communication with the drone is done via a protocol called MavLink (Micro air vehicle communication protocol). This protocol follows a modern hybrid publish-subscribe and point-to-point design pattern. It is also used for communication between the companion computer and the autopilot.

This article is a pre-study of an indoor drone flight project using an autopilot. The aim of this project is to provide a drone that can be programmed by students to perform mission. To achieve this, it is necessary to find an autopilot and make it communicate with a companion computer. In addition, one of the challenges of flying a drone indoors is the detection of obstacles and the path to follow.

This article also aims to provide references for choosing a suitable autopilot controller board. Next, the paper will show how communication between the autopilot and the companion computer is possible via the MavLink protocol. Therefore, this paper will also provide a brief introduction to the Mavlink protocol.

II. AUTOPILOT

A. Role of the autopilot

An autopilot is an on-board system that allows an aircraft to fly and navigate autonomously, without human intervention. It uses flight control algorithms and sensors to maintain the aircraft's altitude, speed and flight path, following predefined flight instructions or reacting to changing flight conditions. The autopilot can be used to perform routine flight tasks in an automated manner, allowing the pilot to concentrate on other tasks, such as mission management or communication with air traffic control. Autopilots are commonly used in airliners and military aircraft [6], but they are also used in other types of autonomous vehicles, such as drones and spacecraft.

The autopilot is a closed-loop control system that has two parts, the state observer and the controller. The state observer part allows to read the data coming from the sensors and the GPS and to determine the altitude, the speed, the position and the inclination of the drone [7]. Then, these data will pass through a PID regulation to provide the commands for the motors. Figure 1 shows the functional structure of an autopilot.

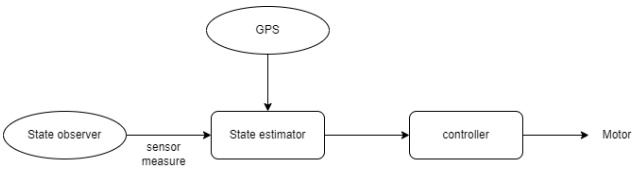

Figure 1: Function structure of an autopilot (reproduced from [7]).

In general, the autopilot includes a microcontroller, an IMU, a barometer and various communication ports. The microcontroller allows the autopilot to perform all the calculation part and interface the input and output. The IMU is composed of a three-axis accelerometer and a three-axis

gyroscope and a magnetometer. The IMU allows to obtain all the acceleration and position data of the drone.

B. Existing projects

There is a very large number of autopilots available on the market. In this section, several off-the-shelf autopilots will be presented and compared.

- NAVIO 2 autopilot [10]: The navio 2 project is an integration of a GNSS receiver, power modules and sensor modules [9]. This project allows to transform the raspberry into an autopilot. It has been developed by the company EMLID and costs about 200$. The shield of Navio 2 has the following sensors and interfaces:
    - 2 redundant IMU
    - 1 high resolution barometer
    - Glonass/GPS/Beidou receiver
    - UART, I2C andA ADC for extensions
    - PPM/SBus input
    - 14 PWM servo output.

  In fact, the NAVIO 2 project uses the processor of the raspberry Pi for the calculation part. In the case of a raspberry Pi 3, CPU will be a 64 bits quad-core ARMv8 whit a clock frequency of 1.2Ghz. This raspberry has 1Gb of RAM. Moreover, the shield has an IO/co-processor for RC command and the activation of PWM output.

Figure 2: Shield Navio 2.

  Navio 2 is a project that allows the implementation of high-level algorithms such as obstacle avoidance, image processing, ...

- Snapdragon Flight Autopilot [11] [12]: Developed by Qualcomm, the snapdragon flight autopilot project offers an autopilot with a very powerful CPU and two onboard cameras. Indeed, the system is based on a Snapdragon 801 on chip system which has a quad core Krait CPU clocked at a frequency of 2.26GHZ, a DSP, and a Qualcomm Adreno GPU. The CPU has a RAM of 2GB. The card has the following sensors and interfaces:
    - 1 IMU
    - 1 Barometer
    - 2 camera videos
    - 1 GNSS receiver
    - I2C
    - 1USB 3.0
    - Wi-Fi 4 & Wi-Fi 5
    - Bluetooth
    - Slot SD card
    - Many others interface …

Figure 3: Snapdragon flight autopilot.

Snapdragon flight autopilot allows the implementation of high-level algorithms such as obstacle avoidance and visual odometry [9]. In addition, this autopilot is equipped with cameras with a resolution of up to 4K, which can be streamed via the numerous connections that the autopilot has.

- VOXL 2 autopilot [13]: developed by modalIA and costs about 1000$, this autopilot is designed to integrate artificial intelligence directly into the drone. The VOXL2 combines a companion computer and an autopilot. This board has the following sensors and interfaces:
    - 2 redundant IMUs
    - 2 redundant barometers
    - 5G/4G

  The main argument of this card is the capacity of the neural network that it embeds. Indeed, this card has 1 NPU of 15 TOPS which is allied with a GPU and a CPU 8 cores clocked at more than 3.091GHz.

Figure 4:VOXL 2.

- OcPoC (Octogonal Pilot on Chip system) Autopilot [9][20] : Developed by Xilinx, OcPoC is an autopilot that uses the power of the FPGA to perform tasks in parallel, allowing it to have a higher computational and input/output capacity than others. This board has the following sensors and interfaces:
    - 2 redundant IMU
    - 1 barometer
    - GNSS receiver (GPS, Glonass and Beidou)
    - Ethernet
    - Many others interface …

  The OcPoC autopilot is based on a system on chip which consists of a microcontroller and an FPGA part. The FPGA part of the OcPoC is an Artix 7 with 28k logic elements. The CPU is a dual core ARM A9 clocked at 667MHz.

Figure 5: OcPoC autopilot.

  The advantage of combining an FPGA with a microcontroller is that the FPGA can quickly and in parallel perform the IO and image processing, while the microcontroller can perform the calculation and software.

Even if all these autopilots are interesting and have undeniable advantages. We decided to work for the project on a type of autopilot based on a well-known STM32, the "PixHawk". This brand is the most used in the literature and the most readily available. Therefore, we will make a comparison of the PixHawk range.[19]

- The "Radiolink Pixhawk" is a Chinese branded "Pixhawk" that is quite diverse in its specifications. [14][20] It has a main processor that makes the main regulation calculations and a co-processor that allows it to manage external communications. These specifications include:
    - 2 telemetry ports,
    - 1 accelerometer
    - 1 magnetometer
    - 1 gyroscope
    - 1 barometer
    - Clock frequency : 168 MHz
    - Flash memory : 2 MB
    - RAM memory : 256 KB
    - Dimensions: 82.2 x 51.8 x 16.5 mm
    - Many other interfaces …

Figure 6: Radiolink Pixhawk.

- The “Holybro Pixhawk” 4 is an American-made autopilot with a more powerful processor than the “Radiolink” and a co-processor that also handles external communications. [15] It is interesting to know that it has an additional GPS with an integrated compass. The Holybro Pixhawk 4 includes:
    - 2 telemetry ports,
    - 1 GPS + integrated magnetometer
    - 2 accelerometers
    - 1 magnetometer
    - 2 gyroscopes
    - 1 barometer
    - Clock frequency : 216 MHz
    - Flash memory : 2 MB
    - RAM memory : 512 KB
    - Dimensions: 44 x 84 x 12 mm
    - Many other interfaces …

Figure 7: Holybro Pixhawk 4.

- The "Holybro Pixhawk 4 Mini" is a lighter version of the previous autopilot as here, there is no coprocessor and there are some peripherals missing. [16] Much used in the field of drone racing, it is appreciated for its minimalist size. Here are its main features:
    - 1 telemetry ports,
    - 1 GPS + integrated magnetometer
    - 2 accelerometers
    - 1 magnetometer
    - 2 gyroscopes
    - 1 barometer
    - Clock frequency: 216 MHz
    - Flash memory: 2 MB
    - RAM memory: 512 KB
    - Dimensions: 38 x 55 x 15.5 mm
    - Many other interfaces …

Figure 8: Holybro Pixhawk 4 Mini.

- The "Black Cube" from "Cubepilot" (previously known has “Pixhawk 2.1”) is an autopilots from the wide "Cube" family. [17] Their distinctive feature is to separate on one hand the connectors to the outside (telemetry, various interfaces, power port,...) and on the other hand the cubic box containing all the embedded electronics contained in any standard autopilot. The cubic container is designed in order that the vibrations caused by the drone's motors are strongly attenuated. In addition, the 'Cube' is equipped with triple redundancy for the IMUs (distributed in different places in the housing) in such a way that the inertial situation of the drone is reflected as accurately as possible. They also all have 2 barometers. Here are the different characteristics of the "Black Cube":
    - 2 telemetry ports,
    - 3 accelerometers
    - 3 magnetometers
    - 3 gyroscopes
    - 2 barometers
    - Clock frequency: 180 MHz
    - Flash memory: 2 MB
    - RAM memory: 256 KB
    - Dimensions :
        - Cube : 38.25mm x 38.25mm x 22.3 mm
        - Carrier : 94.5mm x 44.3mm x 17.3 mm
    - Many other interfaces …

Figure 9: Hex Cube Black.

This range of "Cube" is differentiated first of all by the colour of the "Cube", but above all by the performances they can have. In fact, the most powerful of them, the "Orange Cube", can reach frequencies of 400 MHz while having a RAM memory of 1 MB!

- The Holybro Pixhawk 6C is one of Holybro's most powerful autopilots. With its 3 telemetry ports and high clock rate, it is equipped with a high-performance processor. [18] It is one of the latest autopilots to be released. This autopilot includes :
    - 3 telemetry ports,

- 2 accelerometers
- 1 magnetometer
- 2 gyroscopes
- 1 barometer
- Clock frequency : 480 MHz
- Flash memory : 2 MB
- SRAM memory : 1 MB
- Dimensions: 84.8 x 44 x 12.4 mm
- Many other interfaces …

Figure 10: Holybro Pixhawk 6C.

C. Comparison of Pixhawk

The objective of this research was also to choose the autopilot according to various parameters that fit with our project, such as:
- Its price,
- Its interfacing,
- Its weight/dimensions,
- Its capacity to receive orders from the outside via an external processor (preferably on the STM32 platform).

A comparison table has been done with all of the presented Pixhawk regarding to their main specifications (see Table 1).

In regard to these different considerations, and in the light of the research done on the subject, the choice has been made for the "Radiolink Pixhawk".
Indeed, the mandatory requirements for the development part were the following:
- Use of an external computer driver (companion computer),
- 2 telemetry ports (simultaneous acquisition of altitude, speed, angular position, etc., on the companion computer and on GCS (Ground Control Station) via an external antenna),
- Use of the "MAVLink" communication protocol,
- Library to be used open source.

For the last point, it should be noted that all the aforementioned Pixhawk autopilots were all licensed under the BSD (Berkeley Software Distribution) licence and were hence all open sources.

Board and version	Telemetry inputs	Main Processor	Co-Processor	Sensors	GPS	Weight/dimension	Mean price
Pixhawk 1 -Radiolink Pixhawk (v2)	2	32 Bit Arm® Cortex®-M4, 168 MHz, 2MB memory, 256 k RAM + Co-processeur (failsafe) STM32F1	Co-processeur (failsafe) STM32STMF100	Accel/Gyro:MPU6000 , Magnetometer:IST8310, Barometer: MS5611	/	Carrier: 94.5mm x 44.3mm x 17.3mm, 38 g	207.8$
Pixhawk 4 (v5)	2	32 Bit Arm® Cortex®-M7, 216MHz, 2MB memory, 512KB RAM	32 Bit Arm® Cortex®-M3, 24MHz, 8KB SRAM	Accel/Gyro: ICM-20689, Accel/Gyro: BMI055 or ICM20602, Magnetometer : IST8310, Barometer: MS5611	blox Neo-M8N GPS/GLONASS receiver + Magnetometer : IST8310	44 x 84 x 12 mm, Plastic case: 33.3g; Aluminium: 49g	190$
Pixhawk 4 mini (v4)	1	32 Bit Arm® Cortex®-M7, 216MHz, 2MB memory, 512KB RAM	/	Accel/Gyro: ICM-20689, Accel/Gyro: BMI055 or ICM20602, Magnetometer: IST8310, Barometer: MS5611	blox Neo-M8N GPS/GLONASS receiver + Magnetometer : IST8310	38 x 55 x 15.5 mm, 37.2g	159$
Pixhawk 2 - Hex Cube Black (v3)	2	32 Bit Arm® Cortex®-M4, 168 MHz, 2MB memory, 256KB RAM	Co-processeur (failsafe) STM32F103	Accel/Gyro/Magnetometer: MPU9250, Accel/Gyro/Magnetometer: ICM20602, Accel/Gyro/Magnetometer: ICM20948, Barometer (x2): MS5611	/	Cube: 38.25mm x 38.25mm x 22.3mm, Carrier: 94.5mm x 44.3mm x 17.3mm, 32.8 g	354.47$
Pixhawk 6C (v6)	3	32 Bit Arm® Cortex®-M7, 480MHz, 2MB memory, 1MB SRAM	32 Bit Arm® Cortex®-M3, 72MHz, 64KB SRAM	Accel/Gyro: ICM-42688-P Accel/Gyro: BMI055 Mag: IST8310 Barometer: MS5611	/	84.8 x 44 x 12.4 mm, 59.3g, 59.3 g	222,99$

Table 1: Comparative table of autopilots.

For our application, the Pixhawk 4 has some interesting capabilities. It has an integrated GPS and a new generation FMU, all this for a reasonable cost. In addition, it has 2 redundant IMUs which allows for greater accuracy.

The Pixhawk 4 mini is not suitable for our application as it has only one telemetry port. Finally, the Pixhawk 2 and the 6C are more expensive. Even though their FMUs are more recent, they do not have any particular advantage over the PixHawk 4.

III. DEVELOPMENT

The development includes the handling of the communication protocol "MAVLink" in order to be able to interpret various data coming from the autopilot. This data must be understood by the companion computer in order for it to act according to the pre-programmed points of passage.

A. *MAVLink*

1) *Transport and communication protocol*

The MAVLink protocol provides communication between the UAVs and other systems such as the GCS or the Integrated Driver. MAVLink serialises system status messages and commands in a dedicated binary format. As a result, the MAVLink protocol is a very lightweight protocol.

Furthermore, the companion computer must also be able to react accordingly to obstacles such as those found inside a building. Indeed, the drone must be able to fly indoors in the best conditions of safety for itself, and especially for the people.

The MAVLink protocol [21], by its construction in rather light structures, is advantageous because it makes it compatible with various types of physical layer (OSI model). These include Wi-Fi, Ethernet, but also and in particular serial communication. The latter allows the highest transmission speed of over 250 kbp/s. Tests performed during development have shown that the telemetry port (serial port) can be used at a speed of 921.6 kbit/s!

There are 2 versions of MAVLink, this article will introduce version 2.0 of this protocol. As shown in Figure 11, MAVLink 2.0 has 11 fields.

Figure 11: Frame of MAVLink v2.

As shown in table 2, a MAVLink message starts with a start byte which is 0xFD. Then it contains the size of the message. It also contains a compatibility flag and an incompatibility flag. The incompatibility flag affects the structure of the message and indicates whether certain functionality should be considered when analysing the packet. Then, the compatibility flag does not affect the structure of the message and can refer, for example, to a flag indicating the priority of the message. Then in the frame, we find the sequence number, the system ID, the component ID. The frame also contains an MSGID field which defines the type of message. This is coded on 24 bits which offers 16777215 possible types. Then, the frame contains the Payload. Finally, there is a checksum for the integrity of the message and an optional signature field to guarantee the inviolability of the message.

2) *MAVLink requests and commands*

Different types of messages are defined by MAVLink and are identified by their "Message ID". MAVLink version 1 is distinguishable by the length of its "Message ID" of maximum 8 bits, whereas version 2 is 24 bits. There are therefore 2 categories of messages categorised by MAVLink :

- Status messages : These are used to send altitude, speed, position, etc. data to the GCS or companion computer.
- The command messages: They make it possible to control the behaviour of the UAV such as, for example :
    - Take-off,
    - Landing,
    - Passing through waypoints at a certain height, at a certain speed,
    - Arming or disarming the engines,
    - Setting the take-off home position,
    - Return to the take-off home position
    - ...

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).
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. Note that the broadcast address 0 may not be used in this field as it is an invalid *source* address.
6	`uint8_t compid`	Component ID (sender)	1 - 255	ID of *component* sending the message. Used to differentiate *components* in a *system* (e.g. autopilot and a camera). Use appropriate values in MAV_COMPONENT. Note that the broadcast address `MAV_COMP_ID_ALL` may not be used in this field as it is an invalid *source* address.
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.
For *n*-byte payload: `n=0` : NA, `n=1` : 10, `n>=2` : 10 to (9+n)	`uint8_t payload[max 255]`	Payload		Message data. Depends on message type (i.e. Message ID) and contents.
(n+10) to (n+11)	`uint16_t checksum`	Checksum (low byte, high byte)		CRC-16/MCRF4XX for message (excluding `magic` byte). Includes CRC_EXTRA byte.
(n+12) to (n+25)	`uint8_t signature[13]`	Signature		(Optional) Signature to ensure the link is tamper-proof.

Table 2: MAVLink V2 Frame.

The range of possible controls and data requests being very large, the documentation describing them can be found in the references. [21] [22]

a) The most used commands

There are different types of commands supported, but the most commonly used are the "long" type with a "Message ID" of "76". Coded on 16 bits, the command type specifies precisely what action the UAV should perform. Next come the various useful fields, such as the target system (on 8 bits), the target component (on 8 bits) which will execute the given command. The following parameters (float) are not mandatory but depend on the command to be performed.

Target system	Target component	Command	Confirmation	Param 1	Param 2	Param 3	Param 4	Param 5	Param 6	Param 7
8 bits	8 bits	16 bits	8 bits	float	float	float	float	float	float	float

Table 3 : command long structure

To illustrate a command, the takeoff command that has a Message ID equal to 22, can take as a parameter just the target altitude that the drone has to reach (param7). On the other hand, the command to arm the engines requires only 2 parameters; a boolean to arm/disarm the engines and an integer to force arm/disarm. These two parameters are set to "param1" and "param2" respectively, while "param3" to "param7" are unused. (See Table 4)

There are many different types of commands, and depending on the type, the parameters are used or not. Not all the parameters of a command are necessarily critical to its proper working.

b) The most used status

The MAVLink protocol allows for status messages to be displayed to show system information. In this section, the article will introduce some of the most important and widely used status messages.

The heartbeat message is a very important message in the MAVLink protocol and has the message id 0. It indicates that the vehicle's system (Autopilot) is connected and active. This message is generated once every second. This message is a mandatory message in the MAVLink protocol.

Type	Autopilot	Base_mode	Custom_mode	System_status	mavlink_version
8 bits	8 bits	8 bits	32 bits	8 bits	8 bits

Table 5: Heartbeat Structure.

Table 5 shows the structure of a heartbeat message. The message has a type field which defines the type of device that is used, in our case a quadrotor (MAV_TYPE_QUADROTOR = 2). Then the frame defines the autopilot. The frame has 2 fields to define the mode, this means if the device is armed, if the joystick is connected and the active mode. There is also a status field which allows you to give information such as boot problems or an undefined system. Finally, the packet defines the version of MAVLink used.

The SYS_STATUS command provides a range of information about the battery, the communication error rate, or specific errors related to the autopilot. This command also allows to know which sensors are present and which are active. The "message ID" of this command is 1. The structure of this command is explained in Table 6.

The companion computer can obtain other data using commands such as request_data_stream or request_message. In these commands, the system requests data that will be sent to the companion computer. This data can be the position of the UAV, information from the IMU or the GPS.

Command	Command ID	Parameter	Description
Takeoff	22	1: Pitch	Minimum pitch (if airspeed sensor present), desired pitch without sensor
		2	Empty
		3	Empty
		4: Yaw	Yaw angle (if magnetometer present), ignored without magnetometer. NaN to use the current system yaw heading mode (e.g. yaw towards next waypoint, yaw to home, etc.).
		5: Latitude	Latitude
		6: Longitude	Longitude
		7: Altitude	Altitude
Command	**Command ID**	**Parameter**	**Description**
Arm/Disarm	400	1: Arm	0: disarm, 1: arm
		2: Force	0: arm-disarm unless prevented by safety checks (i.e. when landed), 21196: force arming/disarming (e.g. allow arming to override preflight checks and disarming in flight)
		3	Empty
		4	Empty
		5	Empty
		6	Empty
		7	Empty
Command	**Command ID**	**Parameter**	**Description**
Land	51	1: Abort alt	Minimum target altitude if landing is aborted (0 = undefined/use system default)
		2: Land mode	Precision land mode
		3	Empty
		4: Yaw angle	Desired yaw angle. NaN to use the current system yaw heading mode (e.g. yaw towards next waypoint, yaw to home, etc.)
		5: Latitude	Latitude
		6: Longitude	Longitude
		7: Altitude	Landing altitude (ground level in current frame)

Table 4 : MAVLink command.

Field Name	Type	Units	Values	Description
onboard_control_sensors_present	uint32_t		MAV_SYS_STATUS_SENSOR	Bitmap showing which onboard controllers and sensors are present. Value of 0: not present. Value of 1: present.
onboard_control_sensors_enabled	uint32_t		MAV_SYS_STATUS_SENSOR	Bitmap showing which onboard controllers and sensors are enabled: Value of 0: not enabled. Value of 1: enabled.
onboard_control_sensors_health	uint32_t		MAV_SYS_STATUS_SENSOR	Bitmap showing which onboard controllers and sensors have an error (or are operational). Value of 0: error. Value of 1: healthy.
load	uint16_t	d%		Maximum usage in percent of the mainloop time. Values: [0-1000] - should always be below 1000
voltage_battery	uint16_t	mV		Battery voltage, UINT16_MAX: Voltage not sent by autopilot
current_battery	int16_t	cA		Battery current, -1: Current not sent by autopilot
battery_remaining	int8_t	%		Battery energy remaining, -1: Battery remaining energy not sent by autopilot
drop_rate_comm	uint16_t	c%		Communication drop rate, (UART, I2C, SPI, CAN), dropped packets on all links (packets that were corrupted on reception on the MAV)
errors_comm	uint16_t			Communication errors (UART, I2C, SPI, CAN), dropped packets on all links (packets that were corrupted on reception on the MAV)
errors_count1	uint16_t			Autopilot-specific errors
errors_count2	uint16_t			Autopilot-specific errors
errors_count3	uint16_t			Autopilot-specific errors
errors_count4	uint16_t			Autopilot-specific errors
onboard_control_sensors_present_extended **	uint32_t		MAV_SYS_STATUS_SENSOR_EXTENDED	Bitmap showing which onboard controllers and sensors are present. Value of 0: not present. Value of 1: present.
onboard_control_sensors_enabled_extended **	uint32_t		MAV_SYS_STATUS_SENSOR_EXTENDED	Bitmap showing which onboard controllers and sensors are enabled: Value of 0: not enabled. Value of 1: enabled.
onboard_control_sensors_health_extended **	uint32_t		MAV_SYS_STATUS_SENSOR_EXTENDED	Bitmap showing which onboard controllers and sensors have an error (or are operational). Value of 0: error. Value of 1: healthy.

Table 5 : SYS_STATUS structure.

B. *Hardware part*

This part develope the hardware of the drone in more detail. Indeed, the autopilot will be connected to a companion computer and to various elements so that the drone can fly. Therefore, it is necessary to understand these elements in order to use them in the best way.

As shown in figure 12, a typical autopilot assembly consists of a battery, a regulator with an ampmeter and a voltmeter, a card for sharing the power to the different ESCs, a horn, a safety switch, 4 ESCs, 4 motors and a radio control (RC) module.

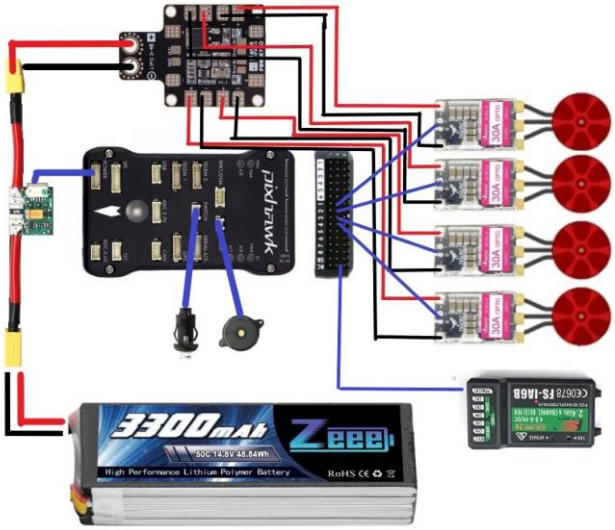

Figure 12 : Basic hardware structure of an autopilot.

Of course, other components such as GPS, various sensors or telemetry can be added to this assembly to improve the system.

1) *RC Module*

The RC module is present to receive manual commands from the joystick. There are several possible transfer modes for the commands. In our case, we need a PPM (Pulse position modulation) command. This modulation mode allows the combination of various PWM signals on a single input.

This type of receiver is mandatory for our RadioLink. However, some have SBUS or PWM receivers.

As shown in Figure 13, each PWM channel generates a pulse on the PPM channel. This mode of programming information allows different motor commands to be communicated on a single channel.

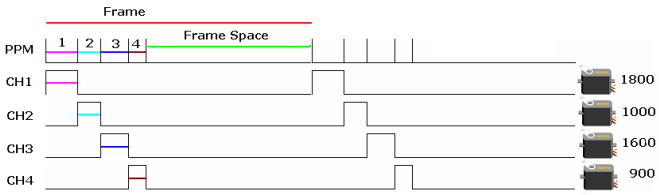

Figure 13 : PPM message.

The RC module is mandatory in our case because in the event of a failsafe, we are obliged to be able to take over. Indeed, we have kept the parameter that in case of a failsafe makes it switch to the RC command. However, the reaction of the Pixhawk to the failsafe can be set if a GPS is present.

2) *Power splitter*

The power splitter is used to distribute the energy from the battery to the various components of the UAV in an equitable manner. This type of device often includes a voltage regulator to supply auxiliary circuits (like a pilot for example).

3) *Power module*

The power module guarantees the correct distribution of the battery's energy to the various components of the drone while monitoring the output power. It is very common for the currents measured by these modules in flying drones to exceed 100A at peak !

4) ESC

The ESC is used to manage the speed of the motors. It transforms the PWM signal into a three-phase signal that generates the rotating field and makes the motor turn at the desired speed.

5) Companion computer

The companion computer is a controller on the drone that allows it to carry out a specific mission. The goal of the project is to create a drone capable of flying indoors autonomously. The purpose of a companion computer is to increase the capabilities of the autopilot and to provide it with additional interfaces such as Wi-Fi or 4G depending on the needs. Hundreds of projects exist that link the autopilot and a companion computer, whether it is a Jetson Nano for artificial intelligence or a Raspberry Pi to be able to hack the drone via Wi-Fi or to do image recognition thanks to the OpenCV library of python.

In the majority of projects seen in the literature, the link between the companion computer and the autopilot is made via the MAVLink prototype and allows the autopilot to be given commands. Therefore, in order to control our drone, we will implement a companion computer.

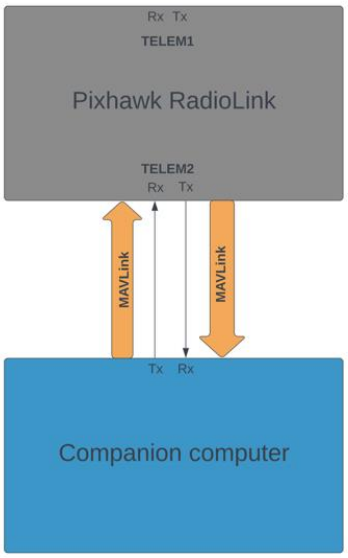

Figure 14: Mavlink set up.

For indoor flying, it is absolutely necessary to install various sensors for obstacle management and avoidance. It is therefore important to consider whether the sensors should be connected to the autopilot or to the companion computer. In fact, there are no precise rules. It depends on the case, but if the sensors are necessary for navigation, it is better to place them on the autopilot. This allows the autopilot to have access to all the information needed for the decision-making process and relieves the companion computer of this burden. However, the autopilot cannot do everything and if it requires artificial intelligence or image processing, the companion computer will have to do it. Moreover, it is interesting to connect the necessary sensors to the companion computer. Indeed, if it is the companion computer that must control and dictate the position orders to the autopilot, it must be able to access all the necessary information to perform this task. In short, it will depend on the architecture and the desired performance.

IV. TESTS PERFORMED

During the development process, several actions were executed with the "MAVLink" protocol in order to :

- Establish communication between the autopilot and the pilot (acquisition of the "Heartbeat" status),
- Acquire data from the autopilot sensors to the pilot,
- Send basic commands from the pilot to the autopilot.

Through the tests made, it emerged that a great potential for pilot control of the autopilot is conceivable. The number of available commands on MAVLink 2 is very large, as is the number of parameters that can be collected from the autopilot's sensors by the pilot. As a result, the number of features that can be implemented is accordingly large! Furthermore, the pilot can also be equipped with AI (with or without image processing if a camera is installed) This expands the possibilities of control/decision making by the pilot even more!

V. CONCLUSION

During this project, a comparison of autopilots showed that there are many different autopilots with different qualities. The article develops the logic and implementation of the communication between the autopilot and the companion computer. For this purpose, a study of the MAVLink protocol was necessary. In the future, it will be necessary to improve and finalise the communication and to define a precise mission for the UAV. In addition, it will be necessary to carry out a more in-depth study of the sensors and details of the UAVs for indoor use.

REFERENCES

[1] Sunghun J, Hyunsu K, “Analysis of Amazon Prime Air UAV Delivery Service”. Journal of Knowledge Information Technology and Systems

[2] Pari P, “FINAL SWARM OF DRONES DEMONSTRATION”, CPSwarm Consortium

[3] Philippe C, “DRONES”. Universalis.fr, https://www.universalis.fr/encyclopedie/drones/3-les-systemes-de-drones-militaires/, accessed the 29/12/2022

[4] Safran, “UAV &ROV Autopilot” https://www.colibrys.com/mems-application/autopilot-uav-rov/, accessed the 29/12/2022

[5] McGrath, R. V., Regalado, M., Aniceto Jr, S. B., Ochengco, C. J., & Chua, A. Design of Modular Casing for Flight Controller and Companion Computer Integration for a Quadcopter UAV.

[6] Chao, H., Cao, Y., & Chen, Y. (2010). Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, *8*(1), 36-44.

[7] Feriel M. (2019). « Conception d’un autopilote pour un drone léger type quar-rotor », Mémoire de master, Université Saad Dahlab

[8] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," in *IEEE Access*, vol. 7, pp. 87658-87680, 2019, doi: 10.1109/ACCESS.2019.2924410.

[9] Zhaolin Yang, Feng Lin and B. M. Chen, "Survey of autopilot for multi-rotor unmanned aerial vehicles," *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, 2016, pp. 6122-6127, doi: 10.1109/IECON.2016.7793820.

[10] Navio 2 documentation, Turn your raspberry pi into a drone controller https://navio2.emlid.com/https://navio2.emlid.com/, accessed the 31/12/2022

[11] Ardupilot, Archived: Qualcomm snapdragon flight kit, https://ardupilot.org/copter/docs/common-qualcomm-snapdragon-flight-kit.html, accessed the 31/12/2022

[12] Documentation PX4: Snapdragon Flight autopilot, https://docs.px4.io/v1.9.0/en/flight_controller/snapdragon_flight.html , accessed the 31/12/2022

[13] VOXL 2, projet ModalAI, https://www.modalai.com/products/voxl-2?variant=39914779836467 accessed the 31/12/2022

[14] Documentation Robu.in: Pixhawk Radiolink, https://robu.in/product/pixhawk-radiolink/, consulté le 31/12/2022

[15] Documentation PX4 : Pixhawk 4, https://docs.px4.io/main/en/flight_controller/pixhawk4.html, accessed the 31/12/2022

[16] Documentation PX4 : Pixhawk 4 Mini, https://docs.px4.io/main/en/flight_controller/pixhawk4_mini.html, accessed the 31/12/2022

[17] Documentation PX4, Hex Cube Black, https://docs.px4.io/main/en/flight_controller/pixhawk-2.html, consulté le 31/12/2022

[18] Documentation PX4, Holybro Pixhawk 6C, https://docs.px4.io/main/en/flight_controller/pixhawk6c.html, accessed the 1/01/2023

[19] Emad Ebeid, Martin Skriver, Kristian Husum Terkildsen, Kjeld Jensen, Ulrik Pagh Schultz, A survey of Open-Source UAV flight controllers and flight simulators, Microprocessors and Microsystems, Volume 61, 2018 , Pages 11-20

[20] E. Ebeid, M. Skriver and J. Jin, "A Survey on Open-Source Flight Control Platforms of Unmanned Aerial Vehicle," *2017 Euromicro Conference on Digital System Design (DSD)*, 2017, pp. 396-402, doi: 10.1109/DSD.2017.30.

[21] Koubaa, Anis & Allouch, Azza & Alajlan, Maram & Javed, Yasir & Belghith, Abdelfettah & Khalgui, Mohamed. (2019). Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2924410.

[22] Documentation MAVLink, Commands/Status, https://mavlink.io/en/messages/common.html#enums, accessed the 3/01/2023