ОТЧЕТ
по состоянию разработки программных и аппаратных средств для реализации режима совместного полета (РСП) нескольких БПЛА от 2017.04.21

Часть 1. Введение

Содержание отчета описывает самые общие принципы РСП и его реализации. Некоторые из приведенных здесь данных на текущий момент могут частично или полностью не соответствовать действительности, т.к. проект находится в активной разработке. Более актуальную версию данного отчета можно получить в корне репозитория проекта по адресу:
https://github.com/MartinDlugozh/swarm_controller

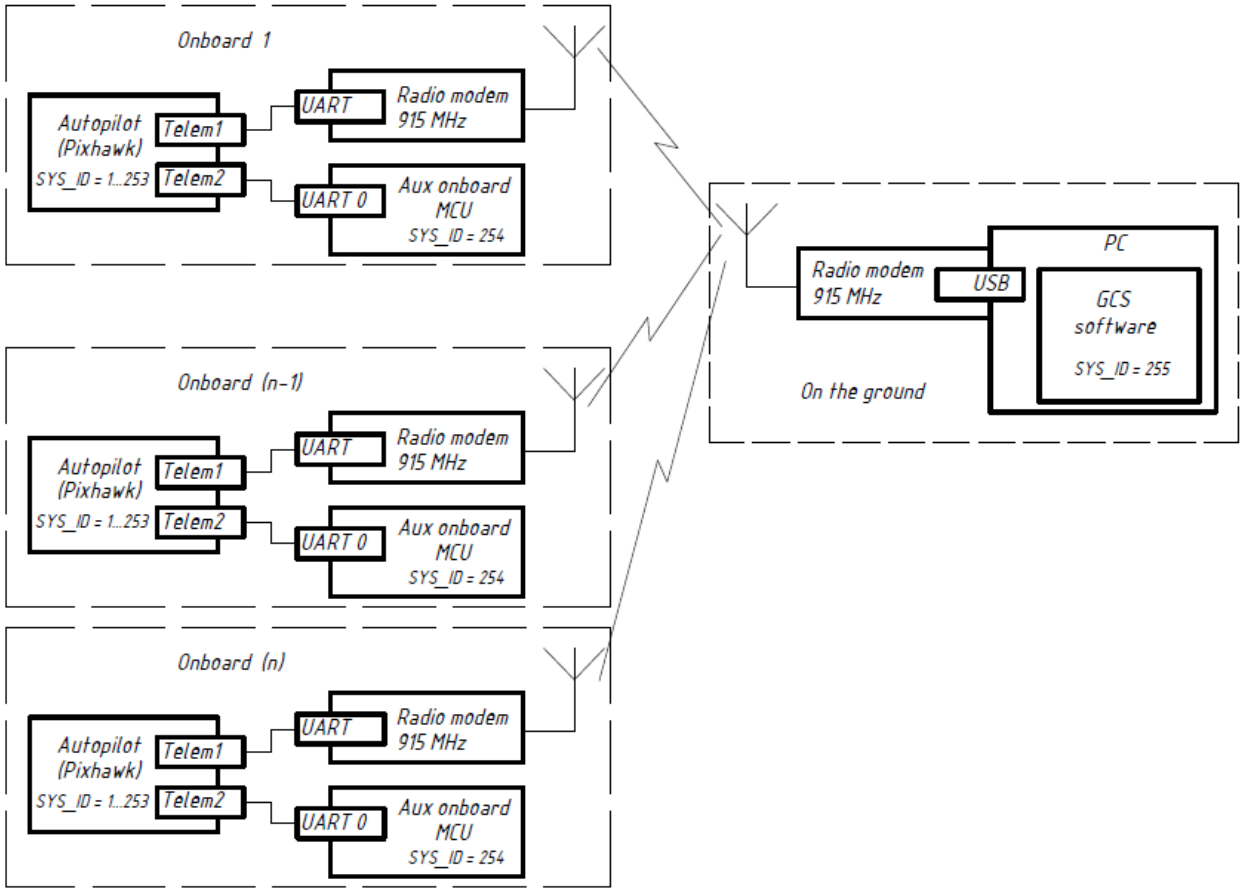
1.1. Основные компоненты обеспечивающие РСП:

1.1.1. Два или больше БПЛА, среди которых один является ведущим (**leader**), а все остальные – ведомыми (**followers**). БПЛА должны быть оборудованы автопилотами с поддержкой по **ArduPilot** (например, Pixhawk или PX4);

1.1.2. Наличие установленных на бортах БПЛА вспомогательных бортовых контроллеров (процессоров), ПО которых реализует выполнение и контроль РСП;

1.1.3. Наличие прямого соединения по радиоканалу между всеми БПЛА (multipoint-сеть).

1.2. Схема сети БПЛА



1.3. Особенности работы ПО

1.2.1 Взаимное расположение БПЛА в пространстве может быть задано тремя способами (определяется параметром **INIT_POS**):

0 – их расположением до начала миссии (до дачи команды ARM);

1 – с помощью предварительно установленных смещений (offsets) для ведомых БПЛА относительно ведущего БПЛА;

2 – с помощью предварительно установленных смещений (offsets) для ведомых БПЛА относительно БПЛА, SYS_ID которого на единицу меньше, чем данного;

1.2.2. Наземная станция управления (GCS или НСУ) не участвует в обеспечении РСП, т.е. ее наличие не является обязательным, если управление ведущим БПЛА осуществляется в ручном режиме;

1.2.3. Установка параметров РСП (в том мисле и смещений) осуществляется с помощью ПО наземной станции;

1.2.4. РСП активируется с помощью пульта ручного управления ведущего БПЛА по каналу 6 (**RC_CH6**).

1.4. Принцип работы программы бортового микроконторллера ведущего БПЛА

1.3.1. Изначально ведущий борт находится в любом режиме полета (например, ручном), ведомый в режиме стабилизации;

1.3.2. После взлета ведущего борта ведомому с помощью пульта ручного управления ведущего борта подается команда на взлет (соотв. тумблер "на себя");

1.3.3. Приняв по последовательному порту (с исп. радиоканала) от ведущего борта сигнал об соотв. изменении положения тумблера, бортовой микроконтроллер вырабатывает последовательность команд для ведомого борта:

а) Set mode - Guided;

б) ARM;

в) Takeoff;

1.3.4. После взлета ведомы борт принимает положение в пространстве относительно своей точки запуска (**MAV_2_HOME**), соответствующее положению ведущего борта относительно его точки запуска (**MAV_1_HOME**)

1.5. Физическая реализация

В качестве бортового контроллера предлагается использование устройства на базе микроконтроллера **ATMega328** (Arduino Nano) (или любой микроконтроллер с по крайней мере одним последовательным портом (UART), объемом оперативной памяти не менее 2Кб и тактовой частотой процессора не менее 16МГц).

МК, подключенный к порту Telem2 автопилота ведущего БПЛА, принимает на вход данные о его положении относительно точки HOME (MAV_1_HOME) – точки запуска. В момент перевода тумблера **канала 6** на пульте в положение «на себя» (pwm > 1500) бортовой МК дает ведомому БПЛА команду ARM, устанавливается MAV_2_HOME, производится взлет ведомого БПЛА (ведущий при этом должен уже находиться в воздухе) до высоты расположения ведущего, координаты ведомого коптера в горизонтальной плоскости при этом устанавливаются в соответствии с выбранным режимом и параметрами взаимного расположения бортов в воздухе в РСП.

При перемещении ведущего коптера в пространстве (относительно MAV_1_HOME), бортовой МК передает ведомому коптеру команды на изменение его положения относительно его HOME (MAV_2_HOME).

В момент перевода тумблера канала 6 на пульте в положение «от себя» (pwm < 1500) бортовой МК дает ведомому БПЛА команду LAND (в текущей точке).

Режим ведущего БПЛА при этом не меняется.

1.6. Ограничения

1.6.1. Количество ведомых БПЛА

Количество ведомых БПЛА программно ограничено до **253** (из-за особенностей программной реализации ArduPilot), однако в будущем с внесением необходимых правок это число может быть увеличено.

Также существует ограничение, устанавливаемое перед компиляцией исполняемого кода бортовых МК, регламентируемое параметром **FOLL_NUM_MAX** (по-умолчанию равно **3**, см. раздел 4.3. - "mc_config.h"). Оно связано с ограниченностью вычислительных ресурсов и оперативной памяти базового микроконтроллера. Число **FOLL_NUM_MAX** может быть увеличено при использовании более производительного микроконтроллера (например ATMega2560 или подобного из серии STM32).

Наиболее проблематичным является ограничение, связанное с пропускной способностью радиоканала обмена данными между БПЛА. Используемое оборудование (**RFD-900+ Multipoint**), максимальная пропускная способность которого составляет 64000 бод (работа возможна только в асинхронном режиме). При этом время активной работы (приема или передачи) при конфигурации сети с одной НСУ, одним ведущим БПЛА и одним ведомым БПЛА составляет около 24% в режиме ожидания и около 29% в РСП. Исходя из этих данных следует сделать вывод о том, что на текущем оборудовании возможно построение сети, содержащей **не более 4 (четырех)** ведомых БПЛА. Дополнительную информацию см. в разделе 5.3.

Часть 2. Средства связи и протоколы

2.1. Ardupilot: MAVLink как средство коммуникации

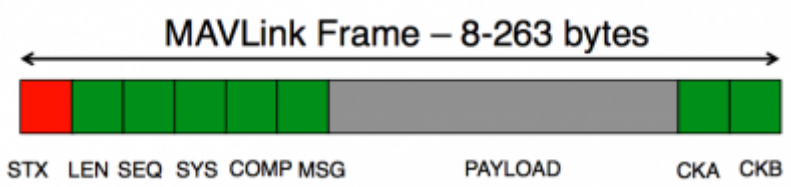
Основным каналом связи, посредством которого осуществляется передача данных между БПЛА, а также связь с наземной станцией управления во время полета, аппаратно являются установленные на каждом БПЛА и на наземной станции телеметрийные модемы, работающие в диапазонах частот 415-425МГц или 900-928МГц. Модемы бортов и НСУ имеют последовательный порт передачи данных (аппаратный протокол UART), на основании которого реализуется программный протокол передачи данных MAVLink.

2.2. MAVLink: характеристика протокола

MAVLink – Micro Air Vehicle Link – это протокол для обмена данными с малым беспилотным летательным аппаратом. Он был разработан как библиотека функций для упорядочивания сообщений. В коде представлен header-only C-библиотекой (это означает, что весь исходный код предоставляется в виде подключаемых *.h-заглавных файлов).

Протокол впервые был выпущен в 2009 году швейцарцем Лоренсом Майером с правами лицензии LGPL (Lesser General Public License) [1].

Диаграмма структуры MAVLink-frame:



Каждый фрейм в общем случае состоит из заголовка (header) фиксированной длины (байты 0 – 5), поля полезной нагрузки сообщения (длина которого меняется в зависимости от типа и количества передаваемых значений) и контрольной суммы (последние два байта сообщения).

Таблица 2.1.1. Структура MAVLink-frame

Имя	Номер байта	Назначение
STX	0	Обозначает начало передачи данных (v1.0: 0xFE)
LEN	1	Длина полезной нагрузки (n)
SEQ	2	Последовательность отправки. Нужно для идентификации потерь данных при передаче
SYS	3	Идентификатор ОТПРАВЛЯЮЩЕЙ системы. Позволяет определять различные системы в одной сети.
COMP	4	Идентификатор ОТПРАВЛЯЮЩЕГО компонента системы. Позволяет определить различные компоненты в одной системе (инерциальный модуль, GPS и т.п.).
MSG	5	Идентификатор сообщения. Указывает тип полезной нагрузки и, следовательно, способ ее декодирования.
Payload	6 to (n+5)	Полезная нагрузка сообщения (в зависимости от типа сообщения ее может и не быть (напр.: Heartbeat).
CRC	(n+6) to (n+7)	Контрольная сумма.

Рекомендованые скорости передачи данных следующие (при передаче всех MAVLink-стримов с частотой 1-5 Гц, IMU – 10 Гц):

1. Для проводного соединения с автопилотом по USB – 115200 baud;
2. Для беспроводного соединения с использованием радомодема типа 3DRradio или RFD900 на канальной скорости 64000 baud – 57600 baud;
3. Для беспроводного соединения с использованием радомодема типа RFD900+ или RFD900x в режиме AES128 шифрования «на лету» – 38400 baud (при передаче всех MAVLink-стримов с частотой 1 Гц).

2.3. Взаимное определение компонентов в системе

MAVLink является последовательным пакетным широковещательным протоколом. Это означает, что каждый компонент сети (борт, наземная станция управления, дополнительные системы) информационно автономный, т.е. постоянная поддержка соединения между ними не обязательна. Для того, чтобы одна система могла опознать другую, необходим прием хотя бы одного специального служебного сообщения **Heartbeat** [2]. При этом следует отметить, что системы, не принадлежащие типу **MAV_TYPE_GCS**, ретранслируют всем системам все принятые от других систем пакеты (пакеты принятые от систем типа MAV_TYPE_GCS **НЕ** ретранслируются).

Heartbeat-frame формируются всеми компонентами системы с заданной частотой (как правило 1 Гц, но опционально может регулироваться и задаваться в пределах от **1 Гц до 255 Гц**). В него входит стандартный заголовок MAVLink-frame: STX, LEN, SEQ, SYS_ID, COMP_ID и MSG_ID; последний принимает значение **MAVLINK_MSG_ID_HEARTBEAT = 0**.

Пример посылаемых MissionPlanner heartbeat-frame в hex представлении:

```
00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16
FE 09 4F FF BE 00 00 00 00 00 06 08 00 00 03 A3 9E
```

```
FE 09 50 FF BE 00 00 00 00 00 06 08 00 00 03 90 D7
FE 09 51 FF BE 00 00 00 00 00 06 08 00 00 03 7A A9
FE 09 52 FF BE 00 00 00 00 00 06 08 00 00 03 44 2A
FE 09 53 FF BE 00 00 00 00 00 06 08 00 00 03 AE 54
```

Как видно из листинга выше, MAVLink использует порядок байтов **Little-Endian****. Следовательно, принаписании программы для работы с протоколом MAVLink на микроконтроллере, убедитесь, что выполнено следующее объявление в **mavlink.h**:

```
#ifndef MAVLINK_ENDIAN
#define MAVLINK_ENDIAN MAVLINK_LITTLE_ENDIAN
#endif
```

Так же следует учесть, что стандартная настройка UART-буфера в проектах оболочки Arduino (если она спользуется) предполагает использование 128 байт оперативной памяти для предварительной выгрузки принятых данных. Это число следует увеличить до максимальной длины MAVLink-сообщения, т.е. до 256: ***#define SERIAL_RX_BUFFER_SIZE 256*** (HardwareSerial.h).

2.4. Прием данных: первоначальный фрейминг

Фрейминг принимаемых по UART данных осуществляется с помощью методов класса mavlink_helpers: **mavlink_parse_char()** или **mavlink_frame_char()**, на вход которых подается прямая последовательность байтов, принимаемых во входной буфер UART (каждый следующий байт пишется в буфер в памяти, длина которого определяется после приема второго байта фрейма - **LEN**). Отличие методов состоит в том, что первый возвращает единицу только если фрейм принят корректно и вычисленная контрольная сумма совпадает с принятой, а в остальных случаях – ноль; второй же кроме этого в случае, если фрейм принят корректно, но контрольные суммы не совпадают, возвращает значение 2. Это позволяет принимать нестандартные или содержащие ошибки фреймы. В остальном результат работы обеих фрейминг-методов одинаков. По завершению корректного приема данных заполняется глобальная структура данных, с отдельными составляющими заголовка фрейма, двоичным массивом, содержащим полезную нагрузку и контрольной суммой.

2.5. Прием данных: парсинг полезной нагрузки сообщений

Далее выполняется проверка **MSG_ID** принятого фрейма. В соответствии со значением MSG_ID выполняется парсинг* полезной нагрузки сообщения (напр.: ATTITUDE, GPS_RAW, RAW_IMU, RAW_PRESSURE и т.д), и, следовательно, заполняются глобальные структуры соответствующих переменных. Напр. для ATTITUDE:

```
typedef struct __mavlink_attitude_t
{
 uint32_t time_boot_ms;     /*< Timestamp (milliseconds since system boot)*/
 float roll;          /*< Roll angle (rad, -pi..+pi)*/
 float pitch;               /*< Pitch angle (rad, -pi..+pi)*/
 float yaw;                 /*< Yaw angle (rad, -pi..+pi)*/
 float rollspeed;           /*< Roll angular speed (rad/s)*/
 float pitchspeed;          /*< Pitch angular speed (rad/s)*/
 float yawspeed;            /*< Yaw angular speed (rad/s)*/
} mavlink_attitude_t;.
```

Процедура парсинга достаточно проста: для каждого MSG_ID определено соответствие структуре данных с типами переменных и офсетами от начала двоичного массива, что позволяет однозначно распознать и обработать принимаемые данные.

2.6. Отправка MAVLink-фреймов

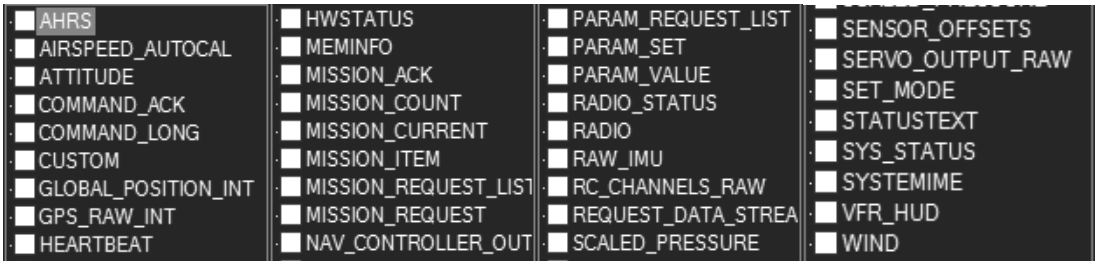
Для отправки сообщений в общем случае используются методы вида **mavlink_msg_*_pack().** Они определены для каждого типа сообщений в соответствующих классах.

Например: После установления соединения (устойчивого двустороннего обмена Heartbeat-frame) между APM и AASC возможен прием AASC данных от APM и наоборот. AASC после десятисекундной задержки для инициализации APM отправляет последнему запрос на передачу потока необходимых данных (далее запрос на передачу потока данных должен отправляться с частотой не ниже частоты отпавки Heartbeat, в противном случае будет передаваться только стандартный минимальный набор параметров борта). Запрос формируется следующей функцией:

```
    mavlink_msg_request_data_stream_pack(uint8_t system_id,                 /*
запр.*/
    uint8_t component_id,      /* запр.*/
    mavlink_message_t* msg,    /* сообщ.*/
    uint8_t target_system,     /* куда? */
    uint8_t target_component,  /* куда? */
    uint8_t req_stream_id,     /* что? */
    uint16_t req_message_rate, /* Гц */
    uint8_t start_stop);       /* 0/1 */
```

Далее сформированный запрос финализируется (составляется форматированное содержимое буфера) с помощью метода **mavlink_msg_to_send_buffer()** класса mavlink_helpers и полностью копируется в буфер, содержимое которого побайтово отправляется через последовательный порт UART.

Pixhawk или бортовой контроллер, приняв сообщение запроса, так же производит его парсинг и заполняет структуру, содержащую данные о параметрах передаваемых потоков. Начинается непрерывную передачу запрошенных данных (req_stream_id) с запрошенной частотой (req_message_rate). Поток запрошенных данных может быть прерван в случае запроса на перкращение передачи (установки в ноль частоты передачи).



Перечень групп параметров борта, записываемых в полетный лог

Для получения всех параметров, отображаемых в GraphLog, необходимо запрашивать req_stream_id = 1, 2, 3, 12. Однако, дополнительно можно в случае необходимости запрашивать и другие специфические параметры борта такие как ориентация гиростабилизированного подвеса, состояние камеры, данные посадочного дальномера (сонара или лидара) и т.п.

2.7. Использование протокола MAVLink в системе РСП

Таким образом ПО бортового МК прежде всего должно иметь в своем составе библиотеки для работы с протоколом MAVLink. В частности, в проекте используются

такие типы сообщений принимаемые от автопилота ведущего борта (телеметрические данные):

- MAVLINK_MSG_HEARTBEAT;
- MAVLINK_MSG_GLOBAL_POSITION_INT;
- MAVLINK_MSG_RC_CHANNELS_RAW.

Используются сообщения подпротокола обмена параметрами:

- MAVLINK_MSG_PARAM_REQUEST_LIST;
- MAVLINK_MSG_PARAM_REQUEST_READ;
- MAVLINK_MSG_PARAM_SET.

И сообщения управления, передаваемые ведомым БПЛА:

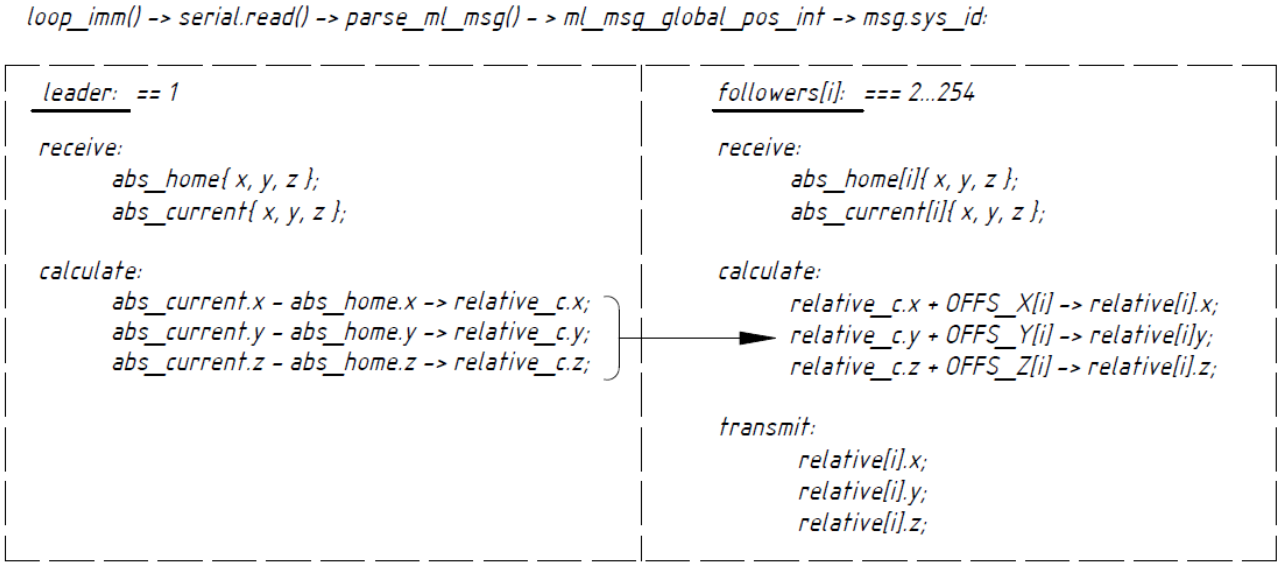
- MAVLINK_MSG_SET_POSITION_TARGET_LOCAL_NED;
- MAVLINK_MSG_SET_POSITION_TARGET_GLOBAL_INT;
- MAVLINK_MSG_SET_HOME_POSITION;
- MAVLINK_MSG_COMMAND_LONG;
- MAVLINK_MSG_ID_SET_MODE.

Часть 3. Описание работы ПО

Базовой задачей при выполнении режима совместного полета является координирование движения ведомых БПЛА и привязка их перемещений к перемещениям ведущего БПЛА. Изначально необходимо обеспечить возможность точного повторения ведомыми БПЛА перемещений ведущего. Использование программно-аппаратного комплекса ArduPilot обеспечивает поддержку автоматического выполнения всех необходимых преобразований координат.

3.1. Алгоритм безусловного позиционирования

Позиционирование ведомых БПЛА относительно ведущего в базовом варианте ПО осуществляется с помощью вычисления перемещения ведущего БПЛА относительно его точки запуска (MAV_1_HOME) и передачи команд ведомым БПЛА на осуществелние аналогичного перемещения относительно их точек запуска (MAV_n_HOME), где n – номер ведомого БПЛА.



3.2. Системы координат

3.2.1. Локальные СК

MAV_FRAME_LOCAL_NED - координаты являются относительными к точке HOME данного БПЛА в системе «Север, Восток, Низ» (North, East, Down (NED)). Может использоваться для указания положения в виде координат: x метров на север, y метров на восток и -z метров над точкой запуска (т.е. точкой HOME).

Направления векторов скоростей так же укаываются в системе NED.

MAV_FRAME_LOCAL_OFFSET_NED - координаты являются относительными к текущим координатам БПЛА в системе NED. Может использоваться для указания положения в виде координат: x метров на север, y метров на восток и -z метров над текущей точкой (т.е. координатой БПЛА в моммент принятия команды).

Направления векторов скоростей так же укаываются в системе NED.

MAV_FRAME_BODY_OFFSET_NED - координаты являются относительными к текущим координатам БПЛА в системе, направление оси OX которой совпадает с текущим курсом БПЛА. Может использоваться для указания положения в виде координат: x метров вперед по текущему курсу, y метров вправо и -z метров над текущей точкой (т.е. координатой БПЛА в моммент принятия команды).

Направления векторов скоростей принимаются относительными к текущему курсу. Может использоваться для указания скоросей в виде составляющих по осям «вперед», «вправо» и «вниз».

MAV_FRAME_BODY_NED - координаты являются относительными к точке HOME данного БПЛА в системе NED. Может использоваться для указания положения в виде координат: x метров на север, y метров на восток и -z метров над точкой запуска (т.е. точкой HOME).

Направления векторов скоростей принимаются относительными к текущему курсу. Может использоваться для указания скоросей в виде составляющих по осям «вперед», «вправо» и «вниз».

СК содержаще в названии **_OFFSET_** являются относительными к текущему положению БПЛА; **_LOCAL_** - относительно к точке запуска (точке HOME); **_BODY_** - указывает, что направления векторов скоростей являются относительными к текущему курсу.

3.2.2. Глобальные СК

MAV_FRAME_GLOBAL_INT – глобальная система координат (WGS84): x: широта в градусах(или метрах)*1.0e-7, y: долгота в градусах(или метрах)*1.0e-7, z: положительная высота над уровнем моря (MSL).

MAV_FRAME_GLOBAL_RELATIVE_ALT_INT - глобальная система координат (WGS84): x: широта в градусах(или метрах)*1.0e-7, y: долгота в градусах(или метрах)*1.0e-7, z: положительная высота над уровнем ландшафта в точке запуска (точке HOME).

3.3. Команды управления

Для управления ведомыми БПЛА используются команды (используется протокол MAVLink), доступные в полетном режиме **GUIDED**: [2]

SET_POSITION_TARGET_LOCAL_NED – установка целевого положения БПЛА в системе NED относительно координат точни запуска или относительно текущего положения (текущих координат) БПЛА.

Таблица 3.3.1. Параметры команды SET_POSITION_TARGET_LOCAL_NED

Поле команды	Описание
time_boot_ms	Время в миллисекундах с момента загрузки АП. Используется

	для компенсации задержки на передачу данных при установлении следующей целевой координаты перемещения.
target_system	System ID
target_component	Component ID
coordinate_frame	Используемая СК (описаны выше)
type_mask	Маска, указывающая, значения каких параметров должны быть проигнорированы (значение 0b0000000000000000 или 0b0000001000000000 означают, что ничего не игнорируется). Значения: бит 1: x, бит 2: y, бит 3: z, бит 4: vx, бит 5: vy, бит 6: vz, бит 7: ax, бит 8: ay, бит 9: az **Примечание:** Всегда должны быть акривными ТОЛЬКО поля пизиции (0b0000111111111000) или ТОЛЬКО поля скоростей (0b0000111111000111). Установка одновременно полей позиции и полей скоростей невозможна. Значения **acceleration**, **yaw**, **yaw_rate** не поддерживаются текущей версией ArduPilot.
x	Координата X в указанной СК в метрах
y	Координата y в указанной СК в метрах
z	Координата Z в указанной NED СК в метрах (т.е. относительная высота со знаком «минус»)
vx	Скорость по оси X в указанной СК в м/с
vy	Скорость по оси Y в указанной СК в м/с
vz	Скорость по оси Z в указанной СК в м/с
afx	Игнорируется
afy	Игнорируется
afz	Игнорируется
yaw	Игнорируется
yaw_rate	Игнорируется

SET_POSITION_TARGET_GLOBAL_INT - установка целевого положения БПЛА в глобальной СК WGS84.

Таблица 3.3.2. Параметры команды SET_POSITION_TARGET_GLOBAL_INT

Поле команды	Описание
time_boot_ms	Время в миллисекундах с момента загрузки АП. Используется для компенсации задержки на передачу данных при установлении следующей целевой координаты перемещения.
target_system	System ID
target_component	Component ID

coordinate_frame	Используемая СК (описаны выше): MAV_FRAME_GLOBAL_INT, MAV_FRAME_GLOBAL_RELATIVE_ALT_INT
type_mask	Маска, указывающая, значения каких параметров должны быть проигнорированы (значение 0b0000000000000000 или 0b0000001000000000 означают, что ничего не игнорируется). Значения: бит 1: x, бит 2: y, бит 3: z, бит 4: vx, бит 5: vy, бит 6: vz, бит 7: ax, бит 8: ay, бит 9: az **Примечание:** Всегда должны быть акривными ТОЛЬКО поля пизиции (0b0000111111111000) или ТОЛЬКО поля скоростей (0b0000111111000111). Установка одновременно полей позиции и полей скоростей невозможна. Значения **acceleration**, **yaw**, **yaw_rate** не поддерживаются текущей версией ArduPilot.
lat_int	Координата X в СК WGS84 в метрах * 1e7
lon_int	Координата Y в СК WGS84 в метрах * 1e7
alt	Абсолютная высота (AMSL altitude) если используется СК MAV_FRAME_GLOBAL_INT, или относительная высота над ландшафтом, если используется GLOBAL_TERRAIN_ALT_INT
vx	Скорость по оси X в СК MAV_FRAME_LOCAL_NED в м/с
vy	Скорость по оси Y в СК MAV_FRAME_LOCAL_NED в м/с
vz	Скорость по оси Z в СК MAV_FRAME_LOCAL_NED в м/с
afx	Игнорируется
afy	Игнорируется
afz	Игнорируется
yaw	Игнорируется
yaw_rate	Игнорируется

Часть 4. Посянения к исходному коду

Далее описаны некоторые фрагменты исходного кода, понимание которых необходимо. По мере доработки программы и документации этот раздел будет дополняться.

4.1. Подключаемые модули

Представлены в заголовке **"inc.h"**.

```
#include "HardwareSerial.h"     // Arduino HAL UART
#include <avr/pgmspace.h> // Средства работы с программной памятью

/**
 * Board and software sepcific headers
 */
#include "mc_config.h"    // Общие настройки платы
#include "mc_var.h" // Глобальные переменные
```

```
#include "buzz.h"         // Звуковой пьезогенератор (пищалка)

/**
 * MAVLink headers
 * заголовки билиотеки протокола MAVLink
 */
#include "./mavlink_avr/ardupilotmega/mavlink.h"
#include "./mavlink_avr/common/mavlink.h"
#include "./mavlink_avr/protocol.h"
#include "./mavlink_avr/mavlink_helpers.h"
#include "./mavlink_avr/checksum.h"
#include "./mavlink_avr/mavlink_types.h"
#include "mavlink_hlp.h"

/**
 * Parameters and common functions
 */
#include "../libraries/LltoUTM.h"
// преобразование координат из GPS WGS84  в UTM и обратно
#include "param_helper.h" // хранение параметров (PROGMEM & EEPROM)
#include "actions.h"      // команды, отправляемые МК ведущего БПЛА
#include "handlers.h"     // обработчики сообщений от GCS и АП ведущего БПЛА
```

4.2. Глобальные переменные

Представлены в заголовке **"mc_var.h"**.

```
/**
 * Main loop timers
 * coord_dt[2] - приращение времени
 * (интегрирование скорости при определении приращения координаты)
 * loop_50Hz - обновление вывода периферийных блоков
 * loop_5Hz - отправка команд ведомому
 * loop_1Hz - отладочные сообщения и heartbeat
 */
struct {
  volatile uint32_t coord_dt[2];
  volatile uint32_t guide;
  volatile uint32_t loop_50Hz;
  volatile uint32_t loop_5Hz;
  volatile uint32_t loop_1Hz;
}main_timer = { 0, 0, 0, 0, 0 };

/**
 * Connection timers
 * leader - состояние соединения с АП ведущего БПЛА
 * followers - состояние соединения с АП ведомого БПЛА
 * gcs - - состояние соединения с наземной станцией
 */
struct Connection{
  uint8_t leader;
  uint8_t followers;
  uint8_t gcs;
}connect = { 0, 0, 0, };
```

```
/**
 * Main loop flags
 * флаги состояния выполнения главного цикла
 */
uint8_t flag_guide = 0; // стадия выполнения программы (значения описаны в
                        // заголовке "mc_config.h"
uint8_t flag_guide_timer = 0;   // флаг таймера задержек при переходе
                                // к следующей стадии

/**
 * Значение задержки при переходе к следующей стадии
 */
uint16_t guide_timer_delay = 0;

/**
 * Координаты лидера в системе NED (D отрицателен)
 * leader_home_x и leader_home_y необходимы, т.к. переход от глобальных
 * координат GPS в NED выполняется на этом МК.
 * relative_c – координаты относительно точки HOME, полученные от ведущего
 * relative_p – координаты относительно точки HOME, вычисленные по
 * приращению времени и составляющим скорости
 */
struct
{
  float x;
  float y;
  float z;
}relative_c = { 0 }, relative_p = { 0 };

/**
 * AP parameters
 * Параметры подключенных к МК автопилотов
 * timer - время крайнего heartbeat от АП
 */
struct
{
  uint8_t sys_id;
  uint8_t status;
  volatile uint32_t timer;
  double home_x;
  double home_y;
  double home_z;
  double curr_x;
  double curr_y;
  double curr_z;
  uint16_t curr_hdg;
  int curr_zone;
  float vx;
  float vy;
  float vz;
}follower[FOLL_NUM_MAX] = {0}, leader = {0};
```

4.3. Предустановленные параметры и значения флагов

Представлены в заголовке **"mc_config.h"**.

```
#define UART_0 Serial     // Номера портов - основной
#define UART_1 Serial1
#define UART_2 Serial2
#define UART_3 Serial3
#define UART_DEF UART_0

#define BAUD_38 38400
#define BAUD_57 57600
#define BAUD_115 115200
#define BAUD_DEF BAUD_57 // Скорость порта для соединения с АП

#define SYS_ID_GCS 255

#define MY_SYS_ID   99    // SYSTEM_ID микроконтроллера
#define MY_COMP_ID   0    // COMPONNENT_ID микроконтроллера

#define SYS_ID_LEADER 1   // SYSTEM_ID ведущего коптера
#define FOLL_NUM_MAX 3

#define TYPE_POS_BITMASK 0b0000111111111000 // NED(x,y,-z)

#define GUIDED_B          89    // Режим Guided
#define GUIDED_C          4     // Режим Guided
#define STABILIZE_B       81    // Режим Stabilize
#define STABILIZE_C       0     // Режим Stabilize

#define GUIDE_NOT_GUIDED 0 // Флаг стадии полета - инициализация
#define GUIDE_GEN   1     // ФСП - разрешение режима GUIDED выполнено
#define GUIDE_ARM   2     // ФСП - ARM ведомых БПЛА выполнен
#define GUIDE_TAKEOFF 3   // ФСП - взлет ведомых БПЛА выполнен
#define GUIDE_GUIDED 4    // ФСП - положение ведомых контролируется ведущим
#define GUIDE_LANDING 5   // ФСП - посадка ведомого

#define TIMER_RELEASED                0
#define TIMER_SET                     1

#define GUIDE_DELAY_GEN               500
#define GUIDE_DELAY_ARM               1000
#define GUIDE_DELAY_TAKEOFF           1000

#define FOLL_STAT_UNKNOWN       0
#define FOLL_STAT_CONNECTED           1
#define FOLL_STAT_NOT_GUIDED          2
#define FOLL_STAT_GUIDED              3
#define FOLL_STAT_LANDING       4
#define FOLL_STAT_LOST                10

#define CONNECTION_LOSS_TIME    5000

#define POS_START   0 // Значение INIT_POS_p - оффсет по месту старта
```

```
#define POS_OFFS  1 // Значение INIT_POS_p - оффсет от MAV1_HOME
```

4.4. Главный исполняемый файл

```
/**
 * Прием сообщений MAVLink от ведущего БПЛА и от НСУ
 * Выполняестя всегда, как только есть свободное процессорное
 * время и проверяет наличие информации в приемном буфере UART;
 * если что-то пришло, передаем это парсеру
 */
void loop_imm(void)
{
  if(UART_DEF.available() > 0)
  {
        mavlink_message_t rd;
        mavlink_status_t status;

        do{
        uint8_t c = UART_DEF.read();

        if (mavlink_parse_char(MAVLINK_COMM_0, c, &rd, &status))
        {
              // обрабатываем телеметрию от ведомых
              handler_followers(rd); // определен в "handlers.h"

              if(rd.sysid == SYS_ID_LEADER)
              {                         // обрабатываем телеметрию от ведущего
                    handler_leader(rd); // определен в "handlers.h"
              }

              if (rd.sysid == SYS_ID_GCS)
              {     // обработка данных от НСУ (прием и отправка параметров)
                    handler_gcs(rd); // определен в "handlers.h"
              }
        }
        }while(UART_DEF.available() > 0);
  }
}

/**
 * Обновление вывода периферийных блоков и таймера
 * Вызывает обновление состояния вывода пьезоизлучателя а также
 * проверяет на переполнение таймер задержки при
 * переходе к следующей стадии РСП
 */
void loop_50Hz(void)
{
  buzz_update();    // определен в "buzz.h"

  // Проверка переполнения таймера
  if(((millis() - main_timer.guide) >= guide_timer_delay)
                                    && (flag_guide_timer == TIMER_SET))
  {
```

```
            flag_guide_timer = TIMER_RELEASED;  // освободили таймер
            if(flag_guide == GUIDE_LANDING)
            {           // после посадки переходим в состояние инициализации
                  flag_guide = GUIDE_NOT_GUIDED;
            }else{
                  flag_guide++;     // или переходим к следующей стадии РСП
            }
      }
   }

   /**
    * Отправка команд ведомым (позиционнирование)
    */
   void loop_5Hz(void)
   {
     for (uint8_t i = 0; i < FOLL_NUM_p; i++) // for each folower
     {
            if(connect.leader == 1)       // if leader is connected
            {
                  if(POS_TYPE_p == 0)           // update follower coordinates
                  {
                        follower[i].curr_x = relative_c.x;
                        follower[i].curr_y = relative_c.y;
                        follower[i].curr_z = relative_c.z;
                  }else if(POS_TYPE_p == 1){
                        follower[i].curr_x = relative_p.x;
                        follower[i].curr_y = relative_p.y;
                        follower[i].curr_z = relative_p.z;
                  }

                  if(leader.curr_hdg != UINT16_MAX)
                  {
                        follower[i].curr_hdg = leader.curr_hdg;
                  }

                  if(flag_guide == GUIDE_GUIDED)
                  {
                        // обертка do_goto(…) определена в "actions.h"
                        do_goto(follower[i].curr_x, follower[i].curr_y,
                              follower[i].curr_z, follower[i].curr_hdg,
                                    UART_DEF, follower[i].sys_id);
                  }
            }

            if(flag_guide == GUIDE_LANDING)
            {
                  // обертка do_land(…) определена в "actions.h"
                  do_land(UART_DEF, follower[i].sys_id);
            }
      }
   }

   /**
```

```
 * Проверка состояния соединиений и отправка сообщений, необходимых для
 * идентификации системы в сети
 */
void loop_1Hz(void)
{
  heartbeat_send(UART_DEF);
  systime_send(UART_DEF);

  if((millis() - leader.timer) > CONNECTION_LOSS_TIME)
  {
        if(connect.leader == 1)
        {
              connect.leader = 0;
        }
  }

  for (uint8_t i = 0; i < FOLL_NUM_p; i++)
  {
        if(((millis() - follower[i].timer) > CONNECTION_LOSS_TIME)
                        && (follower[i].status != FOLL_STAT_UNKNOWN))
        {
              follower[i].status = FOLL_STAT_LOST;
              buzz(200);
        }
  }
}
```

Часть 5. «The Грабли»

Первые попытки использования протокола MAVLink в встраиваемой системе (имеется в виду система бортового МК на базе процессора ATMega328) неминуемо приводят к возникновению ряда проблем и ошибок, решение которых может быть совсем таки неочевидным без детального анализа кода библиотек.

Далее приводится перечень «граблей», на которые наступил автор данной статьи в процессе разработки ПО для РСП и работы с ArduPilot.

5.1. Вывод в UART

Аппаратно зависимая часть нуждается в доработке. Если есть желание использовать **MAVLINK_USE_CONVENIENCE_FUNCTIONS**, то их надо переписать под собственный драйвер последовательного порта (mavlink_helpers.h: строки 672 – 685). Но можно просто не делать #define MAVLINK_USE_CONVENIENCE_FUNCTIONS, а написать свою функцию для вывода в порт. ***Например***:

```
void send_message(mavlink_message_t* msg) {
uint8_t buf[MAVLINK_MAX_PACKET_LEN];
uint16_t len = mavlink_msg_to_send_buffer(buf, msg);
for (uint16_t i = 0; i < len; i++) {
      Serial_ML.write(buf[i]); } }
```

5.2. Использование билиотекой MAVLink большого объема RAM

При первой компиляции почти пустого проекта под ATMega328 сталкиваемся с непомерным для такого контроллера использованием оперативной памяти – 1628 байт (а у нас их всего 2048). Если задача контоллера просто вычленить из потока телеметрии одно сообщение, распарсить в структуру и передать другому

контроллеру, то можно оставить как есть. Но лучше сделать несколько оптимизаций. *Примечание:* некоторые фрагменты кода и подходы могут устареть на данный момент.

Решение 1: mavlink_helpers.h: строки 262 – 267:

```
#if MAVLINK_CRC_EXTRA
#ifndef MAVLINK_MESSAGE_CRC
static const uint8_t mavlink_message_crcs[256] = MAVLINK_MESSAGE_CRCS;
#define MAVLINK_MESSAGE_CRC(msgid) mavlink_message_crcs[msgid]
#endif
#endif
```

заменить на:

```
#if MAVLINK_CRC_EXTRA
#ifndef MAVLINK_MESSAGE_CRC
static const uint8_t mavlink_message_crcs[256] PROGMEM = MAVLINK_MESSAGE_CRCS;
#define MAVLINK_MESSAGE_CRC(msgid) {pgm_read_byte(&mavlink_message_crcs[msgid])}
#endif
#endif
```

Перед этим (можно в main.cpp) сделать #include <avr/pgmspace.h>, если его еще нету. Этот прием позволит перебросить в программную память 256 байт контрольных сумм. То же самое можно сделать и с MAVLINK_CHECK_MESSAGE_LENGTH, если она используется.

Решение 2: mavlink_types.h: строки 185 – 191:

```
#ifndef MAVLINK_COMM_NUM_BUFFERS
#if (defined linux) | (defined __linux) | (defined  __MACH__) | (defined _WIN32)
# define MAVLINK_COMM_NUM_BUFFERS 16
#else
# define MAVLINK_COMM_NUM_BUFFERS 4
#endif
#endif
```

заменить на:

```
#ifndef MAVLINK_COMM_NUM_BUFFERS
#if (defined linux) | (defined __linux) | (defined  __MACH__) | (defined _WIN32)
# define MAVLINK_COMM_NUM_BUFFERS 1
#else
# define MAVLINK_COMM_NUM_BUFFERS 1
#endif
#endif
```

В проекте МК для РСП задействован только один порт, обмен данными идет одновременно только в одном направлении, соответсятвенно, создавать лишние буферы нет необходимости.

5.3. Перегрузка канала передачи данных

Из-за прямого соединения «ведущий – ведомый» все телеметрические данные ведущего БПЛА транслируются не только наземной станции, но и ведомому БПЛА, т.к. для передачи команд управления от бортового МК к ведомому БПЛА используется канал телеметрии (других вариантов пока нет, если не учитывать возможности использования иного радиоканала на другой частоте для общения МК ведущего коптера с ведомым); как следствие, могут возникнуть «тормоза» в работе автопилота ведомого БПЛА из-за необходимости обработки большого количества «мусора» (на текущий момент загруженность канала передачи данных 41% периферийного блока UART Pixhawk – не превышает 22%, а ЦП – 39% - по результатам тестов).

Ссылки:

1. Отчет от 2017.04.11;
2. Copter commands in Guided mode – Dev Documentation [http://ardupilot.org/dev/docs/copter-commands-in-guided-mode.html];