DOI 10.37882/2223–2966.2023.04.05

АППАРАТНО-ПРОГРАММНЫЕ АСПЕКТЫ ОРГАНИЗАЦИИ ПЕРЕДАЧИ ТЕЛЕМЕТРИИ МОБИЛЬНЫХ ЦИФРОВЫХ АВТОНОМНЫХ УСТРОЙСТВ

HARDWARE AND SOFTWARE ASPECTS OF THE ORGANIZATION OF TELEMETRY TRANSMISSION OF MOBILE DIGITAL AUTONOMOUS DEVICES

E. Grach

Summary. Debugging algorithms for controlling mobile robots with a large number of analog sensors is complicated by the lack of the ability to reset telemetry information. The article suggests the simplest and cheapest way to organize wireless communication using the ESP8285 microcontroller, discusses the circuit aspects of installing and programming this controller within the previously ready-made device architecture, reveals the software concept of transmitting information from the main microcontroller to the computer screen.

Keywords: telemetry, robotics, electronic circuitry, ESP8266 (8285) microcontroller.

Грач Евгений Петрович
Кандидат физико-математических наук, ФГБОУ ВО «МИРЭА — Российский технологический университет»
vader701@mail.ru

Аннотация. Отладка алгоритмов управления мобильными роботами с большим количеством аналоговых датчиков затруднена отсутствием возможности сброса телеметрической информации. В статье предложен наиболее простой и дешевый способ организации беспроводной связи с использованием микроконтроллера ESP8285, рассмотрены схемотехнические аспекты установки и программирования данного контроллера в рамках ранее готовой архитектуры устройства, раскрывается программная концепция передачи информации с главного микроконтроллера на экран компьютера.

Ключевые слова: телеметрия, робототехника, схемотехника электронных средств, микроконтроллер ESP8266 (8285).

Введение

Понятие телеметрии очень широкое. Авторы [1] определяют термином «телеметрия» получение любой информации от удаленного объекта. В данной статье, как и в научно-космической области [2], под телеметрией подразумевается передача служебной (не основной) информации о техническом состоянии автономного объекта и его компонентов, с которым не имеется возможности организовать проводной канал связи.

Рассматриваемое устройство представляет из себя мобильного автономного робота класса сумо, имеющего на борту: оптопары (датчики линии) для позиционировании себя в разрешенной (черной) области соревновательного ринга; лазерные дальномеры VL53L0X для обнаружения робота-соперника, сближения с ним и атаки; стартовый модуль, необходимый по регламенту соревнований, представляющий из себя цифровой датчик, выдающий либо ноль, либо единицу, и принимающий сигнал от инфракрасного пульта; средства предстартового управления в виде 0.49`` экрана, нескольких кнопок и переключателей; микроконтроллер ATmega2560, имеющий достаточное количество входов-выходов для подключения не только указанных компонентов, но и сигнальных светодиодов, позволяющих отслеживать режимы работы.

Данная концепция робота возникла в результате последовательного усложнения от простого к сложному. На начальном этапе развития, когда робот собирается за два дня от эскиза до первых заездов, достаточно нескольких цифровых дальномеров (видит — не видит), микроконтроллера с 5–8 входами-выходами. Однако предусмотреть поведение такого робота без минимальной отладочной информации невозможно. Так, иногда оказывается, что цифровые датчики могут реагировать на посторонние объекты, либо на дефекты соревновательного ринга, либо на элементы конструкции самого робота. Для исключения подобных ситуаций нужны, как минимум, сигнальные светодиоды. На рассматриваемом роботе, оборудованном лазерными датчиками VL53L0X, сигнальные светодиоды остались как технический атавизм, т.к. лазерные датчики измеряют и выдают расстояние до препятствия перед собой в единицах, сопоставимых с миллиметрами. С одной стороны, это открывает режимы поведения робота, более сложные, чем таблица бинарных состояний, с другой стороны, для точной настройки этих режимов необходимо знать

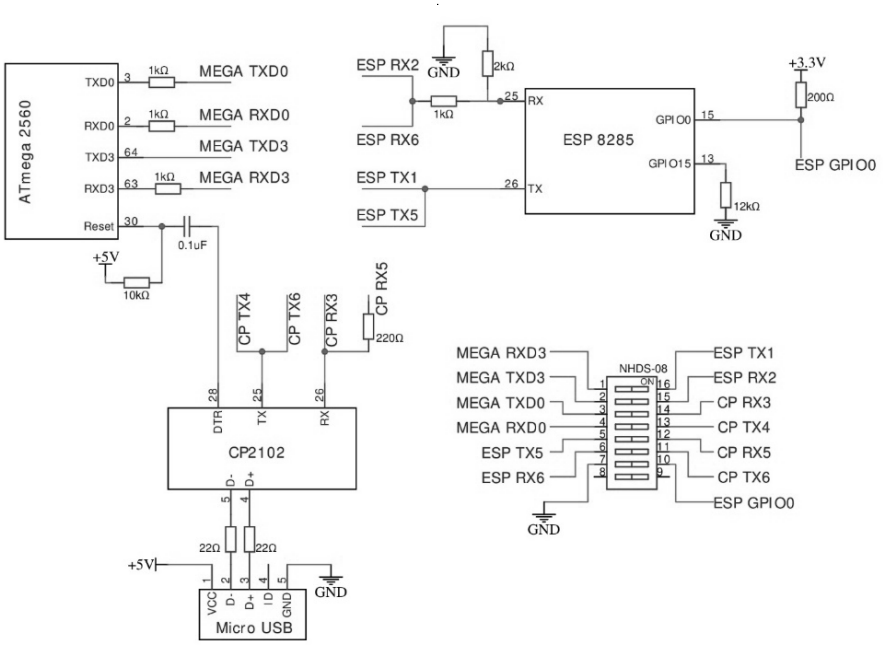

Рис. 1. Общая схема подключения микроконтроллеров ATmega2560, ESP8285 и преобразователя CP2103. Блок smd-перелючателей определяет активные подключения микроконтроллеров к преобразователю и друг к другу.

показания датчиков и состояния робота в каждый момент времени, что вынуждает передавать телеметрическую информацию беспроводным способом.

Схемотехнические аспекты метода решения

Наиболее очевидный и доступный способ — использовать микроконтроллер серии ESP, способный подключаться к беспроводной WiFi-сети и передавать большой объем текстовой информации. По сравнению с ATmega2560, микроконтроллер ESP8266 имеет больший объем памяти, более высокую тактовую частоту, однако меньшее количество цифровых входов-выходов, в том числе аналоговых, другой уровень питания. Решение обозначенных проблем потребовало бы капитальных изменении, в том числе схемотехнических, в конструкции робота, добавления сдвиговых регистров как входных, так и выходных, добавления расширителя аналоговых входов. Поставленная задача (передача текстовой телеметрии) была решена наиболее простым способом — подключением ESP8285 к ATmega2560 через последовательный порт. Это привело к существенным программным сложностям передачи телеметрии, но упростило схемотехническое решение задачи. Необходимый уровень питания (3.3 В) на борту рассматриваемого устройства уже был от датчиков VL53L0X, сопоставление логических уровней последовательного порта обеспечивается простейшим делителем. Интеграция данного микроконтроллера на печатную плату устройства проводилось по схеме R3 ATmega2560+ESP8266 от производителя RobotDyn [3]. Схема хороша тем, что позволяет использовать всего один преобразователь серийного интерфейса в USB на оба микроконтроллера. Более того, т.к. у контроллера ESP8285 (так же как и у ESP8266) всего один последовательный порт передачи данных, то он должен быть использован и для программирования самого контроллера ESP и для связи с основным контроллером ATmega2560. Изменения по сравнению со схемой R3 от RobotDyn — использование преобразователя интерфейса CP2103 вместо CH340G и использование контроллера ESP8285 вместо ESP8266.

Таким образом, общая схема взаимодействия представляет собой два микроконтроллера: ATmega2560 и ESP8285, которые связываются с преобразователем интерфейсов CP2103 либо друг с другом по последовательному порту. Каждая конкретная связь определяется парой smd-переключателей, отдельный седьмой переключатель необходим для ввода ESP8285 в режим программирования по последовательному порту.

Напрямую к 4-му цифровому выходу микроконтроллера ESP8285 подключен сигнальный светодиод, который поможет при отладке программы управления. Наличие LED-экрана, подключенного к ATmega2560

Таблица 1. Пример обмена сообщениями о пересылке ip-адреса сервера между двумя микроконтроллерами

Фрагмент исходного кода ATmega2560	Фрагмент исходного кода ESP8285

int ip1 = EEPROM.read (82);
int ip2 = EEPROM.read (83);
int ip3 = EEPROM.read (84);
int ip4 = EEPROM.read (85);
if (Serial3.available () > 0) {
 	
String res=»»;
while (Serial3.available ()) {
if (millis () -time>5) break;
 	
res+= (char) Serial3.read ();
}
 	
if (res.indexOf («give_IP»)!= –1) {
String ip = «server_ip:»+String (ip1) +».»+String (ip2)
+».»+String (ip3) +».»+String (ip4) +»@@@»;
Serial3.println (ip);
 	
}
if (res.indexOf («ip OK»)!= –1) {
WiFi_status = 2;
}
}
	unsigned long st=millis ();
	String mess=»»;
	String s=»server_ip:»;
	String f=»@@@»;
	while (mess.indexOf (s) == –1 || mess.indexOf (f) == –1) {
	if (millis () -st>1000) {
	Serial.println («give_IP»);
	st=millis ();
	mess=»»;
	}
	while (Serial.available ())
	mess += (char) Serial.read ();
	}
	int index1 = mess.indexOf (':', mess.indexOf (s));
	int index2 = mess.indexOf ('@', index1 +1);
	String ip = mess.substring (index1+1, index2);
	mess=»»;
	for (int i=0; i<18; i++) {
	if (i<ip.length ())
	servername [i] =ip [i];
	else
	servername [i] =0;
	}
	Serial.println («ip OK»);
	Serial.println (servername);

позволит контролировать стадии подключения по беспроводной связи и отображать уровень сигнала.

Как было сказано, используется схема R3 ATmega2560+ESP8266 от производителя RobotDyn, однако представленная производителем схема в [3] не работает. Проблема состоит в невозможности программирования микроконтроллера ESP8285. При этом не имеет значения тот факт, что используется ESP8285 вместо ESP8266 и CP2103 вместо CH340G. Доступно много разных схем подключения микроконтроллеров ESP8266 и ни одна из них не может обеспечить программирование контроллера преобразователем интерфейса через USB. Проблема решается добавлением стягивающего резистора на 12 кОм между шиной нулевого потенциала и 13-м выводом микроконтроллера (GPIO15). Данный резистор присутствует на оригинальных схемах производителей отладочных плат семейства ESP и отсутствует на всех производных. Как пишет производитель [4], данный резистор при подтягивании к логической единице активирует интерфейс SDIO, а при стягивании к нулю — интерфейс UART. Полное отсутствие данного резистора не предусмотрено. Часть схемы, имеющая отношение к микроконтроллеру ATmega2560 работает без замечаний и не потребовала доработки. Внедрение навесным монтажом резистора 12 кОм между шиной нулевого потенциала и 13-м выводом микроконтроллера ESP8285 (GPIO15) позволило завершить схемотехнический этап разработки и перейти к программированию.

Программные аспекты метода решения

Порядок передачи телеметрической информации прост: основной микроконтроллер ATmega2560 отправляет в свой третий серийный порт текстовую информацию, ESP8285 принимает эту информацию и отправляет на сервер по беспроводной сети. Сервер собирает эту информацию и сохраняет в файл.

При подаче питания основной микроконтроллер ATmega2560 работает по своей логике, не дожидаясь сигналов готовности от ESP8285, но проверяя содержимое своего RX-буффера. При поступлении в RX-буффер определенной команды выдает в TX-буффер необходимую информацию. Дело в том, что к основному микроконтроллеру подключен LED-экран с несколькими кнопками управления, что позволяет изменять некоторые простые настройки, регулируемые в бинарном (включено-выключено) или в числовом диапазоне (больше-меньше). Это позволяет вывести в регулируемые непосредственно на верхней части робота настройки четыре числа, задающие ip-адрес и сам факт

передачи телеметрии. При включении ESP8285 контроллер по серийному порту отправляет текстовую команду “send IP”, а ATmega2560 при наличии такого сочетания символов в своем RX-буфере пересылает ip сервера, к которому должен подключиться ESP8285.

ESP8285 при включении, не дожидаясь дополнительных команд и информации от ATmega2560, подключается к беспроводной сети, имя и пароль от которой жестко прописаны в программе управления и не изменяются настройками на верхней панели робота.

После успешного подключения к прописанной беспроводной сети ESP8285 отправляет в свой TX-буфер текстовую строку «WiFi OK», что интерпретируется основным контроллером и отображается на LED-экране. После прохождения данного этапа ESP8285 раз в секунду отправляет микроконтроллеру ATmega2560 текстовое сообщение «give_IP», делает это до тех пор, пока в входной RX-буфер не поступит информация. Повторение отправки сообщения необходимо для ликвидации возможных рассинхронизаций двух микроконтроллеров, связанных как с не мгновенным подключением к беспроводной сети со стороны ESP8285, так и с возможными неполадками на шине I2C или особыми режимами работы со стороны ATmega2560 (функционирование меню настроек реализовано через цикл, в котором программа управления не взаимодействует с RX-буффером).

При обнаружении тестовой строки «give_IP» в своем RX-буфере, микроконтроллер ATmega2560 из 4-х байт своей энергонезависимой памяти собирает текстовую строку вида «server_ip:192.168.0.102@@@» и отправляет микроконтроллеру ESP8285. Особый признак окончания сообщения потребовался для корректного выделения IP-адреса из строки. Пример данного обмена приведен в таблице 1.

На стороне микроконтроллера ATmega2560 используется принудительный выход из цикла считывания RX-буффера. ESP8285 работает на большей тактовой частоте, чем ATmega2560, и, нельзя исключать ситуацию, при которой ESP8285 пересылает информацию на сторону ATmega2560 быстрее, чем тот может обработать. В этом случае ATmega2560 будет наполнять объект res до переполнения доступной динамической памяти, с непредсказуемыми последствиями. Сбои в работе ATmega2560 недопустимы, и, поэтому, предусмотрен выход из режима чтения RX-буффера, если чтение происходит дольше 5 миллисекунд, с последующей очисткой объекта res на следующей итерации основной программы управления.

На стороне ESP8285 используется уже упомянутый повтор сообщения «give_IP» по временной метке, проверка наличия в RX-буффере признака начала и окончания ожидаемой строки. При выполнении этого условия, прекращается передача запросов «give_IP», выделяется подстрока, содержащая сам ip-адрес сервера, переписывается в глобальную строку, используемую в следующем фрагменте при подключении к серверу. На сторону ATmega2560 пересылается подтверждающее сообщение, ATmega2560 при получении подтверждения меняет свой внутренний флаг статуса беспроводного подключения и индикацию на LED-экране.

Следующий этап подключения — соединение ESP8285 с сервером по полученному ip. Выполняется командой

```
while (!client.connect (servername, 7770));
```

Микроконтроллер ATmega2560 в этом не участвует, работая по своему алгоритму, проверяя содержимое RX-буфера на каждой итерации своего глобального цикла. Если по каким-то причинам подключение установить не удастся, то это никак не повлияет на основной микроконтроллер ATmega2560, а ESP8285 формально зависнет в бесконечном цикле. Порт подключения прописан в программе управления ESP8285 и не изменяется. При успешном подключении к серверу ESP8285, как и в предыдущем примере, отправляет текстовую строку на сторону ATmega2560, на которую тот отреагирует изменением статуса подключения и индикацией на LED-дисплее.

Следующий этап — пересылка уникального однобайтового серийного номера робота. Этот номер записан в ячейке энергонезависимой памяти ATmega2560, пересылается в сторону ESP8285 один раз в финальный момент подключения. ESP8285 добавляет этот уникальный номер к каждому пересылаемому телеметрическому кадру. После получения микроконтроллером ESP8285 уникального номера робота, производится отправка первого сообщения на сервер вид

```
;;;4Raider;;;5
```

Данное сообщение интерпретируется сервером как начало передачи информации, отображается в консоли сервера и именно поэтому номер робота преобразован в текстовое имя.

Работа основного микроконтроллера ATmega2560 построена на показании датчиков VL53L0X, которые выдают новые данные через каждые 20мс. Сами датчики могут быть опрошены и чаще, но смысла это не имеет, т.к. более свежая информация поступает только с интервалом 20 мс. На основе показаний контроллер устанавливает ШИМ-скорости и направления вращения

моторов, отправляет информацию о показаниях датчиков и установленных скоростях, дополняет временем, прошедшим с момента включения микроконтроллера, уникальным порядковым номером 20-милисекундного интервала (кадра), дожидается окончания этого интервала, после чего снова переходит к опросу датчиков. Каждый фрагмент информации отправляется командой Serial3.print () без накопления в промежуточной строке и поступает по серийному порту во входной RX-буфер контроллера ESP8285. Скорость передачи данных по серийному порту между двумя микроконтроллерами установлена в 1 млн. бод/с. При этом, размер TX-буфера микроконтроллера ATmega2560 установлен в 64 байта (значение по умолчанию), а размер RX-буфера контроллера ESP8285 необходимо увеличить как минимум до 2048 байт для исключения его переполнения, потери информации, и, что наиболее критично, нарушения покадровой разбивки.

Как было сказано, каждый свой 20-милисекундный интервал работы микроконтроллер ATmega2560 начинает с анализа имеющейся в RX-буфере информации. Если в буфере присутствует сочетание символов «charge» (отправляется микроконтроллером ESP8285 каждые 5 секунд) — в TX-буфер сбрасывается информация о текущем заряде батарей робота в виде «1S [4.03] 2S [4.00] 3S [-3.64] ».

Каждая итерация продолжается опросом имеющихся на борту датчиков, вместе с получением показаний датчиков информация отравляется по Serial3 в ESP8285 в текстовом формате вида «a=xxx/600», где a — идентификатор датчика, xxx — снятое показание датчика, может быть, в пределе, как четырехзначным, так и однозначным, второе число — установленный предел чувствительности. Для аналогового датчика линии (оптопары), подключенной к 10-битовому АЦП микроконтроллера ATmega2560, фактически, задает границу — «белое — черное», переводя его в цифровой режим работы. Данное значение регулируемое и устанавливается для каждого датчика в каждом конкретном случае индивидуально и должно отслеживаться так же, как и текущее показание. Лазерные дальномеры VL53L0X имеют такое же граничное значение «видит — не видит», которое используется в некоторых примитивных режимах поиска, так же настраивается под конкретные условия, и так же должно быть известно. Единственный цифровой датчик — стартовый модуль — добавляется в виде «sm [x] », где x — текущее состояние.

Заканчивается 20-милисекундный интервал добавлением в телеметрический кадр информации о времени выполнения этого интервала (позволяет отследить неполадки в шине I2C), времени, прошедшим с момента включения микроконтроллера, уникального порядкового номера телеметрического кадра. Кадр заканчивается сочетанием символов, которое гарантированно не встречается в других частях кадра. Выбрано сочетание;;;5 — признак окончания кадра, который необходим на стороне сервера для разделения кадров друг от друга.

На стороне микроконтроллера в телеметрический кадр добавляется информация о уровне сигнала (RSSI). Стоит отметить, что, поскольку телеметрический функционал вообще и микроконтроллер ESP8285 в частности не являются критически необходимыми для выполнения главной функции робота, то место под микроконтроллер ESP8285, его внешний обвес и антенну на печатной плате выделялось по остаточному принципу (размер робота ограничен 10x10 см).

Результат

Однако на ранних стадиях отладки всей системы обнаружены серьезные (до 90%) потери информации. Пропускались целые серии телеметрических кадров (до 150 штук), корректно фиксировались сервером серия из 10–15 кадров, после чего снова существенный провал.

Потерь информации не происходит на стороне ATmega2560, любой объем данных просто уходит в последовательный порт, что возможно установить подключением к компьютеру через преобразователь интерфейса CP2103. Так же установлено, что потерь информации не происходит и при передаче по беспроводному каналу. Сервер фиксирует любой генерируемый микроконтроллером ESP8285 объем данных. Методом исключения делается вывод, что информация теряется при ее получении на стороне микроконтроллера ESP8285. Увеличение скорости передачи данных по последовательному порту до 1 млн. бод/с и размера RX-буфера до 1000 байт решило, на первый взгляд, данную проблему.

Содержимое RX-буффера микроконтроллера ATmega2560 полностью переписывается во внутренний объект класса string, обрабатывается покадрово (удаляются двойные пробелы, нуль-байты, добавляется имя робота, RSSI, признак начала кадра) и покадрово отправляется на сервер.

Сервер представляет собой консольное windows-совместимое сетевое приложение под управлением Windows Socket API (WSA), функционирующее в соответствии с [5] в каких-либо исключительных пояснениях не нуждается. Серверное приложение свою работу начинает с определения и вывода в консоль собственного ip устройства, на котором запущено приложение

```
2 b=962 c=1023 d=849 lt=9 Mm=29636 sc=2698;;;5
;;;4Giperon: s[-65]_ sm[0] a=1021/600 b=961/504 c=1023/562 d=838/788 lt=10 Mm=29645 sc=2699;;;5
;;;4Giperon: s[-65]_ sm[0] a=1022/600 b=961/504 c=1023/562 d=838/788 lt=10 Mm=29656 sc=2700;;;5
;;;4Giperon: s[-65]_ sm[0] a=1022/600 b=962/504 c=1023/562 d=838/788 lt=10 Mm=29665 sc=2701;;;5
;;;4Giperon: s[-65]_ sm[0] a=1022/600 b=961/504 c=1023/56
```

Рис. 2

сервера. Это необходимо для выставления ip-адреса сервера на панели настройки робота.

Соединение с каждым роботом представляет собой объект класса SOCKET, дополненный ip-адресом робота, его текстовым и числовым именем, контейнером snap, вмещающем 100 кадров телеметрии, необходимые поля для ориентации в этом контейнере. Назначение этого контейнера — сохранить 100 телеметрических кадров до момента активации стартового модуля. После установления соединения с новым подключенным клиентом его socket отправляется в общий пул подключенных клиентов, ему выделяется отдельная потоковая функция и символьный приемный буфер размером 10 тысяч байт. Потоковая функция каждые 75 миллисекунд принимает во входной буфер поступившую информацию, разбивает ее на кадры и покадрово заполняет контейнер snap. Микроконтроллер ESP8285 настроен таким образом, чтобы сброс информации серверу происходил покадрово, однако на деле, поскольку работа идет через какой-то дополнительный буфер беспроводного соединения, на краях приемного буфера, как правило, оказывается лишь часть телеметрического кадра (полукадр-начало в конце приемного буфера и полукадр-конец в начале буфера). Это вынудило, помимо признака окончания кадра (;;;5) на стороне ESP8285 дополнять начало каждого кадра признаком начала (;;;4).

Содержимое входного буфера сервера перед покадровой разбивкой выглядит примерно как на рисунке 2.

Добавление признака начала кадра позволило без потерь информации склеивать такие полукадры на соседних итерациях потоковой функции. Целые кадры выделяются из приемного буфера сервера и отправляются в контейнер snap.

От включения робота, соединения с сервером и начала сброса телеметрии до момента активации стартового модуля может пройти значительное время (до минуты), телеметрия за которое не нуждается в записи в файл. Состояние робота в этот момент может быть отслежено с помощью бортового LED-дисплея, либо с помощью приложения-клиента, обладающего, в отличие от сервера, графическим интерфейсом, и предназначенного фактически, для вывода мгновенной информации о параметрах робота. Однако для полноценного анализа решено записывать в файл не только телеметрию в момент разрешающего сигнала со стартового модуля, но и по 100 кадров до активации и после деактивации этого сигнала. При интервале сброса телеметрии в 20 миллисекунд это соответствует 2 секундам. Если с кадрами после деактивации стартового модуля проблем не возникает — просто от уникального номера последнего кадра отсчитывается и пишется еще 100 кадров, то с кадрами до активации все сложнее. В контейнер snap по кругу записываются вновь поступившие кадры телеметрии, номер самого свежего хранится в отдельном поле. Содержимое кадра анализируется, и, если произошло переключение стартового модуля с нуля на единицу, в папке, соответствующей конкретному роботу, создается файл, имя которого содержит некоторый текстовый ассоциативный префикс, дату и время активации стартового модуля. В этот файл сбрасывается содержимое контейнера, после чего файл дополняется поступающими с робота в реальном времени телеметрическими кадрами. При обратном переключении стартового модуля запускается счетчик на запись 100 последующих кадров, после чего файл закрывается.

Выводы

Таким образом, реализованная система выполняет свою функцию по фиксации телеметрической информации с автономного устройства. Как было отмечено в самом начале, передача телеметрии не является критически важной функцией робота, что и отразилось на общей реализации системы в виде дополнительного микроконтроллера. Наиболее рациональным (и, как отмечено выше, наиболее трудозатратным со схемотехнической точки зрения) представляется упомянутый перевод всего функционала робота на микроконтроллер ESP8285. Это позволит полностью избежать проблем с фиксацией данных на входе в микроконтроллер ESP8285. Как оказалось, при увеличении размера кадра наступают все те же описанные проблемы с потерей информации. В работах [2,5] данной проблемы не возникает, т.к. в первом случае сброс информации происходит на порядки реже, а во втором — в гораздо меньшем объеме. В данной схеме передачи телеметрии решить эту проблему можно кодированием информации. Например, совершенно не нужно четырехзнач-

ные показания оптопар передавать четырьмя байтами с именным префиксом и выставленной границей чувствительности. Границы достаточно передавать периодически, как это реализовано с передачей заряда батарей. Полезные показания датчиков, разделенные на 4, можно передать одним байтом (с допустимой потерей информации), а идентификаторы датчиков заменить идентификаторами групп датчиков. Тем не менее, реализованная система работает и выполняет свою функцию.

ЛИТЕРАТУРА

1. «Современная телеметрия в теории и на практике», Назаров В.А., Козырев Г.И., Шитов И.В. и др — СПб.: Наука и техника, 2007
2. «Разработка программного обеспечения для обработки и анализа данных телеметрии малого космического аппарата» Дубовик А.А. — Сборник работ 74-й научной конференции студентов и аспирантов Белорусского государственного университета: в 3 частях. 2017.
3. Mega WiFi R3 ATmga2560+SP8266 Schmatic — RobotDyn — https://robu.in/wp-content/uploads/2018/08/Schematic0G-00005806MEGAWiFi-R3-AT2560-ESP8266–32MB-CH340G.pdf — 2018
4. "ESP8266 Technical Reference" Espressif Inc. — https://www.espressif.com/ — 2020
5. «Client part of robot telemetry based on TRIK controller» Pikkio P.F., Zheleznjakov I.E. — Молодой ученый. 2022. № 33 (428).

© Грач Евгений Петрович (vader701@mail.ru).

Журнал «Современная наука: актуальные проблемы теории и практики»

МИРЭА — Российский технологический университет