УДК 004.232

Кучеренков Александр Петрович, Конченков Владимир Игоревич

МОДУЛЬ СБОРА ДАННЫХ С LoRa КЛИЕНТОВ

Разработан аппаратно-программный комплекс на базе LoRa-модуля и микроконтроллера серии STM32. Система осуществляет приём пакетов данных со сторонних LoRa передатчиков и по разработанному протоколу обмена данных на базе интерфейса SPI отправляет данные главному контроллеру.

Технология LoRa, интерфейс SPI, микроконтроллер STM32, модульный принцип.

Kucherenkov Alexander Petrovich, Konchenkov Vladimir Igorevich

DATA COLLECTION MODULE FROM LORA CLIENTS

A hardware and software package based on the LoRa module and the STM32 series microcontroller has been developed. The system receives data packets from third-party LoRa transmitters and sends data to the main controller using the developed data exchange protocol based on the SPI interface.

LoRa technology, SPI interface, STM32 microcontroller, modular principle.

Введение

Стремительное развитие систем автоматического мониторинга и развитие радиоэлектроники позволяют разрабатывать энергоэффективные и автономные средства передачи малых пакетов данных.

Технология LoRa это технология модуляции маломощной сети передачи данных со скоростью 0,3-50 кб/с и дальностью от 1 до 15 км. создана для межмашинного (M2M) взаимодействия и используется для передачи данных в автономных датчиках экологического наблюдения и коммунальном хозяйстве.

В настоящей работе выполнено проектирование и изготовлен прототип комплекса сбора данных с LoRa клиентов, используемый для удалённого мониторинга температуры, скорости ветра, влажности и прочих параметров на объектах промышленности и домашней автоматизации. Так же для повышения эффективности системы выполнена разработка протокола обмена данными между модулями системы.

Цель статьи - продемонстрировать процесс разработки и тестирования приёмника пакетов данных с использованием технологии LoRa и протокола обмена между модулями системы.

Проектирование приёмника сообщений по сети LoRa

Модуль приёмника LoRa представляет собой систему из двух чипов - микроконтроллера серии STM32 (STM32F030F4P6) и LoRa-модуля (RFM96W).

Соединение с модулем LoRa происходит по интерфейсу SPI, где LoRa-модуль выступает ведомым устройством, а контроллер STM32F030F4P6 выступает в роли ведущего. Также у LoRa-модуля есть несколько программируемых пинов, из которых используется всего один, который сигнализирует о получении пакета данных с внешних LoRa передатчиков.

Соединение с внешним главным устройством сделано аналогичным способом, с добавлением ещё двух линий ввода-вывода общего назначения, отвечающих за выбор режима работы микроконтроллера. В данном случае уже микроконтроллер STM32F030F4P6 выступает в роли ведомого устройства.

На рис. 1 показана система подключения приёмника LoRa модуля с главным устройством. В качестве главного устройства выступает микроконтроллер серии STM32 (STM32F407VET6).

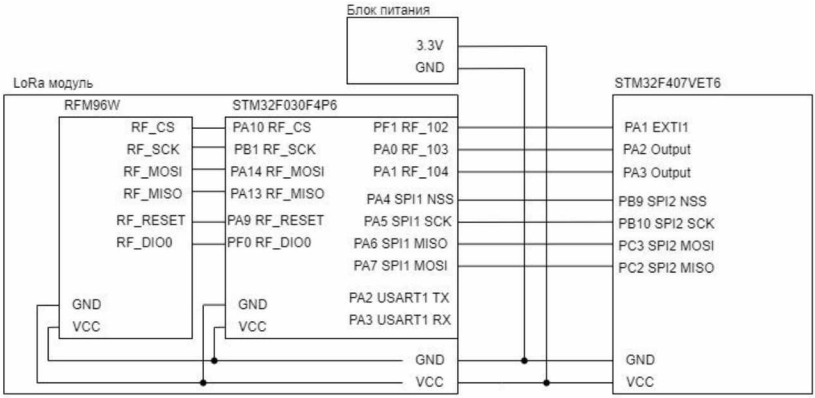

Рис. 1. Схема электрическая принципиальная макета базовой станции для объединения всех модулей

Микроконтроллер STM32F030F4P6 располагает только одним аппаратным интерфейсом SPI, в то время как необходимо использовать два таких интерфейса. В связи с этим с LoRa модулем RFM96W соединение реализовано с помощью программного SPI на пинах: PB1 (CKL), PA14 (MOSI), PA13 (MISO), PA10 (CS).

Аппаратный интерфейс SPI используется для соединения с главным модулем в режиме ведомого. Пин PF1 микроконтроллера STM32F030F4P6 является аналогом пина DIO0 на LoRa модуле и выполняет те же функции, а именно сигнализирует главному устройству что пакет был успешно получен, обработан и готов для пересылки главному модулю.

По сути, микроконтроллер STM32F030F4P6 повторяет работу самого модуля RFM96W, но он необходим для расширения его возможностей в перспективе и снятия с главного модуля части задач, таких как первичная инициализация и настройка модуля.

Таблица 1

Список команд для управления ведомым устройством

1	2	3	4	Описание
0x00	address	X	X	Операция чтения регистра LoRa модуля. Результат чтения будет записан в первый байт промежуточного буфера. Второй параметр - адрес регистра Lora модуля. Третий параметр - не используется и может быть любым.
0x01	address	var	X	Запись значения в регистр LoRa модуля. Второй параметр - адрес регистра LoRa модуля. Третий параметр - значение для записи.
0x02	num	X	X	Заполнение промежуточного буфера данными одного из буферов датчиков. Всего 8 буферов датчиков, которые заполняются последовательно сдвигом вправо, соответственно по адресу 0x00 всегда последний пришедший пакет. Второй параметр - номер буфера датчиков (0x00 – 0x07). Третий параметр - не используется и может быть любым.
0x03	var	X	X	Заполнение промежуточного буфера пришедшим байтом. Весь промежуточный буфер заполнится одним и тем-же значением. Второй параметр – значение для записи. Третий параметр - не используется и может быть любым.
0x04	index	var	X	Заполнение определённого байта промежуточного буфера. Промежуточный буфер размером в 65 байт. Второй параметр – индекс байта в промежуточном буфере. Третий параметр – значение для записи.
0x05	X	X	X	Операция установки в первый байт промежуточного буфера значение rssi Значение rssi последнего пришедшего пакета хранится в отдельной переменной и его таким образом можно считать. Второй параметр – не используется и может быть любым. Третий параметр - не используется и может быть любым.
0x06	var	X	X	Вкл/Выкл кода вечного цикла. В вечном цикле происходит постоянный опрос LoRa модуля, этой командой его можно отключить. (в последней ревизии данная команда отключена) Второй параметр - 0 – опрос выключен, 1 – опрос включен. Третий параметр - не используется и может быть любым.

0x07	var	X	X	Выбор TX/RX В теории в будущем можно будет перенастроить модуль с приёма на передачу. На данный момент передача не реализована и этой командой лучше не пользоваться, (в последней ревизии данная команда отключена) Второй параметр - 0 – приём, 1 – передача. Третий параметр - не используется и может быть любым.
0x08	var	var	X	Установка размера буфера Второй параметр - кол-во байт для чтения. Третий параметр - не используется и может быть любым.
0x09	var	X	X	Установить флаг, согласно которому в конце пакета будет приписываться RSSI Второй параметр: 0 – пакет без RSSI 1 – пакет с RSSI Третий параметр - не используется и может быть любым.
0x0A	var	var	X	Объединение двух предыдущих команд только флаг RSSI теперь передаётся вторым параметром, а размер буфера первым

Линии PA0 и PA1 управляются со стороны главного устройства и необходимы для выбора режимов работы ведомого.

При PA0 равном 0 и PA1 равном 0 ведомое устройство будет работать в режиме исполнения кода вечного цикла.

При PA0 равном 0 и PA1 равном 1 ведомое устройство один раз повторно инициализирует аппаратный SPI. Такая реализация позволяет в случае неисправности в любой момент повторно инициализировать SPI на ведомом устройстве.

При PA1 равном 0 и PA0 равном 1 микроконтроллер переходит в режим обработки команд. В буфер SPI подгружаются данные для обмена с ведущим, например ответ на предыдущую команду.

При PA1 равном 1 и PA0 равном 1 микроконтроллер переходит в режим передачи данных. В буфер SPI подгружается выбранный пакет данных, как правило — это последний принятый пакет от LoRa модуля.

Обмен данными с ведомым происходит согласно командам, представленным в табл. 1. Команда представляет собой слово из 4 байт. Некоторые команды представляют собой запросы на получение какой-либо

информации. Как правило, это получение всего буфера данных или получение одного байта информации сигнализирующего о успешности или неуспешности выполнения команды.

LoRa-модуль получает данные от сторонних LoRa передатчиков и может хранить до 8 пакетов данных размером до 65 байт.

Ниже приведен пример реализации настройки и управления ведомым устройством согласно табл. 1. Функция set_pins() настраивает линии, отвечающие за управление режимами работы ведомого устройства.

```
void set_pins(uint8_t mode)
{
    switch (mode)
    {
    case 0:  RF_103_LOW;  RF_104_LOW;                break;
    case 1:  RF_103_LOW;  RF_104_HIGH; delay_ms(1);  break;
    case 2:  RF_103_HIGH; RF_104_LOW;  delay_ms(1);  break;
    case 3:  RF_103_HIGH; RF_104_HIGH; delay_ms(1);  break;
    default: RF_103_LOW;  RF_104_LOW;                break;
    }
}
```

Функция SPI_TransmitReceive() организуется стандартным образом, рекомендуемым в описании библиотеки HAL.

```
void SPI_TransmitReceive(uint8_t* pTxData, uint8_t* pRxData,
uint8_t Size)
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
    HAL_SPI_TransmitReceive(&hspi2, (uint8_t*)(pTxData),
(uint8_t*)(pRxData), Size, 1000);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
}
```

Функция send_command() реализует отправку команду ведомому устройству.

```
uint8_t send_command(uint8_t p1, uint8_t p2, uint8_t p3,
uint8_t p4)
```

```
{
    EXTI->RTSR &= ~EXTI_RTSR_TR1;
// Блокируем прерывания пина RF_102
    uint8_t tx_tmp[4] = { 0 };
    uint8_t rx_tmp[4] = { 0 };
    set_pins(2);
    tx_tmp[0] = p1; tx_tmp[1] = p2;
    tx_tmp[2] = p3; tx_tmp[3] = p4;
    SPI_TransmitReceive(tx_tmp, rx_tmp, 4);
    set_pins(1);
    EXTI->RTSR |= EXTI_RTSR_TR1;
// Разрешаем прерывания пина RF_102
    return rx_tmp[0];
}
```

Функция command_responce() используется для отправки команды и получения ответа ведомого.

```
uint8_t command_response(uint8_t p1, uint8_t p2, uint8_t p3,
uint8_t p4)
{
    uint8_t tmp = 0;
    tmp = send_command(p1, p2, p3, p4);
    tmp = send_command(0xFF, 0xFF, 0xFF, 0xFF);
    return tmp;
}
```

Функция get_buffer() служит для считывания буфера.

```
void get_buffer(uint8_t* pTxData, uint8_t* pRxData, uint8_t
Size){
    EXTI->RTSR &= ~EXTI_RTSR_TR1;
    // Блокируем прерывания пина RF_102
    set_pins(3);
    SPI_TransmitReceive(pTxData, pRxData, Size);
    set_pins(1);
    EXTI->RTSR |= EXTI_RTSR_TR1;
```

```
    // Разрешаем прерывания пина RF_102
}
```

Функция get_buffer_sensor() служит для получения данных одного из буферов. Буфер может хранить до 8 последних принятых пакетов.

```
void get_buffer_sensor(uint8_t* pTxData, uint8_t* pRxData,
uint8_t Size, uint8_t rssi, uint8_t num_buf){
    // Установка размера буфера обмена
    send_command(0x0A, Size, rssi, 0xFF);
    // Установка одного из буферов датчиков для обмена
    send_command(0x02, num_buf, 0xFF, 0xFF);
    get_buffer(pTxData, pRxData, Size);
}
```

Выводы

Разработана система LoRa приёмника и протокол обмена данными с внешним контроллером по интерфейсу SPI, расширенному за счёт двух дополнительных управляющих линий, служащих для выбора режима работы модуля.

Устройство может служить для дополнения существующих устройств по передаче данных в условиях ограниченного покрытия других сетей или же их полного отсутствия, т.к. технология LoRa не нуждается в сторонней инфраструктуре.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. LoRa RFM95/96/97/98(W) – supply24 – [Электронный ресурс] – URL: https://supply24.online/doc/manual/Poluprovodniki/Moduli-svyazi/HOPE-MICROELECTRONICS/RFM95_96_97_98W.pdf (дата обращения 12.05.2024).

2. Serial Peripheral Interface – Wikipedia – [Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/Serial_Peripheral_Interface (дата обращения 15.05.2024).

3. LoRa-module (SX1278) with STM32 – GitHub – [Электронный ресурс] – URL: https://github.com/SMotlaq/LoRa (дата обращения 17.05.2024).

Кучеренков Александр Петрович, Магистрант Волгоградского государственного технического университета, Россия, город Волгоград, проспект им.

В.И. Ленина, д. 28, 400005, телефон: +7 (9610) 69-90-24, email: kucherenko-valexander@gmail.com.

Конченков Владимир Игоревич, кандидат физико-математических наук, доцент кафедры "Электронно-вычислительные машины и системы", Волгоградского государственного технического университета, Россия, город Волгоград, проспект им. В.И. Ленина, д. 28, 400005, телефон: +7 (9047) 56-86-41, email: kontchenkov@yandex.ru.

Kucherenkov Alexander Petrovich, Undergraduate student at Volgograd State Technical University, Volgograd, V.I. Lenin Avenue, 28, 400005, Russia, phone: +7 (9610) 69-90-24, email: kucherenkovalexander@gmail.com .

Konchenkov Vladimir Igorevich, Candidate of Physical and Mathematical Sciences, Associate Professor of the Department of Electronic Computing Machines and Systems, Volgograd State Technical University, Volgograd, V.I. Lenin Avenue, 28, 400005, phone: +7 (9047) 56-86-41, email: kontchenkov@yandex.ru .