БЕСПРОВОДНОЕ ОБНОВЛЕНИЕ ПРОГРАММИРУЕМОГО КОНТРОЛЛЕРА НА БАЗЕ МИКРОПРОЦЕССОРА ESP32

Е. Д. Золотухо, В. О. Басацкий

staiband2116@gmail.com, forester7251@gmail.com;
Научный руководитель — Н. В. Левкович, старший преподаватель

Предложены программные решения для беспроводного обновления прошивки микроконтроллера ESP32. Разработана программа нижнего уровня, в которой реализован протокол передачи данных между ESP32 и ПК, а также программа верхнего уровня для передачи команд на микроконтроллер. Реализована защита канала связи путём внедренияAES шифрования в программы обоих уровней.

Ключевые слова: обновление прошивки ESP32; протокол связи; таблица разделов.

ВВЕДЕНИЕ

Микроконтроллеры ESP32 могут использоваться на крупных предприятиях или системах умного дома, где они могут быть разнесены по различным труднодоступным местам, причём обеспечение безопасной работы и обновления прошивки может быть весьма важным аспектом в таких случаях, где нарушение производственных процессов может привести к серьёзным последствиям. Разработчик микроконтроллеров ESP32 предоставляет библиотеку для беспроводного обновления программного обеспечения. Недостатками данной библиотеки являются:

- прошивка передается в открытом виде;
- пароль для доступа к режиму обновления передается в открытом виде;
- не поддерживается дозагрузка, в случае обрыва соединения;
- прошивка происходит только через WiFi HTTP соединение через браузер.

Целью работы было расширение возможностей обновления микропрограммы на микроконтроллере ESP32, путем добавления поддержки других интерфейсов связи с использованием шифрования протокола.

Для выполнения работы использовалась отладочная платаLILYGO® TTGO ESP32-Paxcounter LoRa32, оснащенная микроконтроллером ESP32-PICO-D4, со встроенными модулями WiFi, Bluetooth. На плате установлены модуль LoRa, USB-разъем, предназначенный для подачи питания на микроконтроллер и его программирования, а также OLED дисплей [1].

Особенности ESP32-PICO-D4:

- WiFi и Bluetooth функционал в миниатюрном SiP корпусе 7×7мм;
- встроенная память 4MB SPI FLASH;
- широкаяпериферия: ADC, LNA, DAC, SD/SDIO/MMC Host Controller, SPI, SDIO/SPI Slave Controller, EMAC, PWM, LED PWM, UART, I2C, I2S, IR, GPIO;
- аппаратно-реализованные криптографические функции: AES, RSA, SHA-256, RNG [2].

ПРОТОКОЛ ОБМЕНА СООБЩЕНИЯМИ

Для передачи данных на микроконтроллер был разработан протокол, структура которого приведена в таблице 1.

Таблица 1

Структура протокола обмена сообщениями

№	Описание поля	Длина поля
1	номер команды	1 байт
2	длина дополнительных данных	2 байта
3	дополнительные данные	0..1024 байта
4	контрольная сумма	2 байта
5	дополнение пакета до размера кратного 16 байтам	0..15 байт

В протоколе используются команды:

- *Код команды 1 – GetID* – запрос названия и версии текущей исполняемой микроконтроллером прошивки в текстовом виде.
- *Код команды 2 – Update* – команда загрузки обновления, разбитого на блоки произвольного размера не более 1024 байт. В пакете запроса передаётся размер прошивки (uint32), смещение текущего записываемого блока относительно начала прошивки (uint32), очередной блок передаваемой прошивки. Микроконтроллер присылает ответ об успешности или неуспешности записи этого блока.
- *Код команды 3 – Restart* перезагружает микроконтроллер.
- *Код команды 4 – GetPartitionTable*– запрос дампа таблицы разделов в текстовом виде. Дополнительных данных в запросе нет.
- *Код команды 5 – GetPartitionData* – запрос бинарного дампа flash-памяти микроконтроллера. В запросе передаётся базовое смещение (uint32) во flash-памяти и размер читаемых данных (uint16). В ответе возвращает запрошенные данные.

В микроконтроллере ESP32 flash-память представлена в виде разделов, которые задаются таблицей. В первом разделе хранится загрузчик, во втором – таблица разделов, остальные разделы могут

использоваться либо для данных, либо для исполняемых прошивок. При загрузке микропроцессора запускается загрузчик, который выбирает какая из прошивок будет выполняться. Перезаписывать можно только тот раздел, который в данный момент не является исполняемым. Команды GetPartitionTable и GetPartitionData использовались для изучения работы таблицы разделов, а также для проверки на правильность при загрузке обновления.

ШИФРОВАНИЕ ПРОТОКОЛА СВЯЗИ

Данные в протоколе связи шифруются алгоритмом AES в режиме ECB (режим электронной кодовой книги) блоками по 16 байт. В микроконтроллере аппаратно поддерживается алгоритм шифрования AES с длинами ключа в 128 бит, 192 бит и 256 бит. Поэтому было необходимо определить с каким из ключей шифрования достигается наилучший баланс производительности и защищенности.

Были реализованы и протестированы алгоритмы с разной длиной ключа. Результаты измерения времени представлены в таблице 2.Как видно из таблицы разница во времени работы алгоритма при различных длинах ключа небольшая и составляет десятые от микросекунды, что позволяет нам сделать вывод, что, выбрав наиболее надёжный ключ в 256 бит мы не потеряем в быстродействии нашей программы, ведь разница во времени шифрования и дешифрования одного блока 16 байт для различных длин ключа составляет лишь 0,5–0,6 мкс.

Таблица 2

Среднее время шифрованияблока 16 байт для ключа разной длины

Направление / Размер ключа	Шифрование	Дешифрование
128 бит	10.6 мкс	10.6 мкс
192 бит	11.1 мкс	11.1 мкс
256 бит	11.5 мкс	11.5 мкс

На основании выбранного ключа и соответствующего ему времени шифрования и дешифрования можно вывести формулу для подсчёта времени, которое уйдёт на шифрование или дешифрование N байт:

$$t = \left\lceil \frac{11{,}5 \text{ мкс} \cdot N}{16} \right\rceil, \quad (1)$$

где t – время, которое уйдёт на шифрование или дешифрование N байт, N – размер шифруемого или дешифруемого пакета в байтах.

Можно подсчитать, что время шифрования одного мегабайта данных будет равно 736 мс, что на фоне времени прошивки микроконтроллера в 11 с является некритичным.

ЗАКЛЮЧЕНИЕ

Разработана программа нижнего уровня для микроконтроллера ESP32, консольная программа верхнего уровня для ПК. Реализовано шифрование протокола связи, обеспечивающее сокрытие передаваемых данных. На текущий момент передача обновления прошивки поддерживается через последовательный порт и через TCP соединение по WiFi. Разработанный протокол связи может использоваться и для UDP соединения, и для соединения через LoRa.

Библиографические ссылки

1. LILYGO® TTGOESP32-PaxcounterLoRa32 V2.1 1.6 Version 433/868/915 MHZLoRaESP-32 OLED 0.96 InchSDCardBluetoothWIFIModule [Электронный ресурс] : [сайт]. URL: http://www.lilygo.cn/prod_view.aspx?TypeId=50003&Id=1271 &FId=t3:50003:3(датаобращения: 23.05.2023).
2. ESP32-PICO-D4 Datasheet [Электронный ресурс]: [сайт]. URL: https://www.espres-sif.com/sites/default/files/documentation/esp32-pico-d4_datasheet_en.pdf(дата обращения: 23.05.2023).