

Применение MicroPython для разработки встраиваемых устройств

Болтовский Гавриил Александрович

Приамурский государственный университет им. Шолом-Алейхема

Студент

Аннотация

Цель статьи – описание технологии MicroPython и демонстрация её возможностей для разработки встраиваемых устройств. Представлен процесс создания скетча, включая основные этапы настройки и программирования. В результате продемонстрирована работа устройства, разработанного с использованием MicroPython, с пояснением его функциональности.

Ключевые слова: MicroPython, разработка встраиваемых систем, Arduino

Using MicroPython to develop embedded devices

Boltovsky Gavriil Alexandrovich

Sholom-Aleichem Priamursky State University

Student

Abstract

The purpose of the article is to describe MicroPython technology and demonstrate its capabilities for embedded device development. The process of sketch creation is presented, including the main stages of customization and programming. As a result, the operation of the device developed using MicroPython is demonstrated with explanation of its functionality.

Keywords: MicroPython, embedded systems development, Arduino

1 Введение

1.1 Актуальность исследования

Встраиваемые системы занимают ключевое место в современных технологиях благодаря их широкому применению в промышленности, медицине и повседневной жизни. MicroPython предоставляет лёгкий и эффективный инструмент для разработки таких устройств, позволяя реализовывать сложные алгоритмы даже на аппаратных платформах с ограниченными ресурсами. Изучение и популяризация MicroPython способствует снижению барьеров для входа в сферу разработки встраиваемых систем.

1.2 Обзор исследований

Исследование Н. Н. Беспалова посвящено программной реализации управляющего устройства на базе ПЛИС, предназначенного для обмена данными по интерфейсу SPI [1]. В работе подробно описаны алгоритмы

управления передачей данных, а также рассмотрены особенности реализации на программируемых логических интегральных схемах.

Г. Т. Катукия провёл сравнительный анализ различных интерфейсов передачи данных, используемых в микроконтроллерах [2]. В своей работе автор рассматривает характеристики, преимущества и недостатки наиболее популярных интерфейсов, таких как UART, SPI и I2C.

История создания и эволюции архитектуры файловой системы FAT изложена в работе В. Мешкова [3]. Автор подробно описывает ключевые этапы разработки FAT, её основные версии, а также анализирует влияние архитектурных особенностей на производительность и применимость системы. Работа представляет интерес как для исторического понимания, так и для практического использования файловой системы в современных приложениях.

А. В. Деткова разработала модуль для проверки плат семейства Arduino, который позволяет проводить автоматизированное тестирование функциональности аппаратного обеспечения [4]. В статье подробно описаны структура и алгоритмы работы модуля, а также приведены результаты его применения на практике.

1.3 Цель исследования

Целью статьи является описание технологии MicroPython и демонстрация её возможностей для разработки встраиваемых устройств.

1.4 Постановка задачи

Для достижения поставленной цели следует особенности работы с MicroPython, включая описание процесса её применения для создания скетчей и демонстрация работы готового устройства.

2 Методы исследования

MicroPython – облегчённая версия языка Python [5], предназначенная для работы с микроконтроллерами. Созданный в 2014 году Дэмианом Джорджем, он быстро завоевал популярность благодаря простоте и поддержке платформ, таких как ESP32, STM32 и Raspberry Pi Pico.

Основная цель MicroPython – упростить разработку программного обеспечения для встраиваемых систем. Он позволяет писать лаконичный и понятный код, снижая вероятность ошибок. Встроенные библиотеки облегчают работу с периферийными устройствами/

Традиционно разработка микроконтроллеров требовала сложного низкоуровневого программирования. MicroPython устраняет эту проблему, делая процесс доступным для школьников, студентов, и энтузиастов. Использование популярного Python ускоряет обучение и прототипирование.

Работа с микроконтроллером на MicroPython принципиально отличается от классического подхода, основанного на языках программирования C или C++. В основе использования MicroPython лежит идея загрузки универсальной прошивки, которая превращает микроконтроллер в устройство, способное

интерпретировать код на Python. После прошивки можно работать с микроконтроллером через интерактивную консоль REPL, выполняя команды в реальном времени. Это упрощает тестирование и позволяет быстро отлаживать программы. Написанные на Python скрипты загружаются на устройство без необходимости компиляции, а их запуск происходит мгновенно. Такой подход обеспечивает высокую скорость разработки и удобство при работе с периферийными устройствами.

3 Результаты и обсуждения

Среда разработки была настроена в интегрированной среде Visual Studio Code (VS Code) [6]. Для эмуляции работы с микроконтроллером использовался инструмент Wokwi [7], который позволяет моделировать поведение встраиваемых систем (рис. 1).

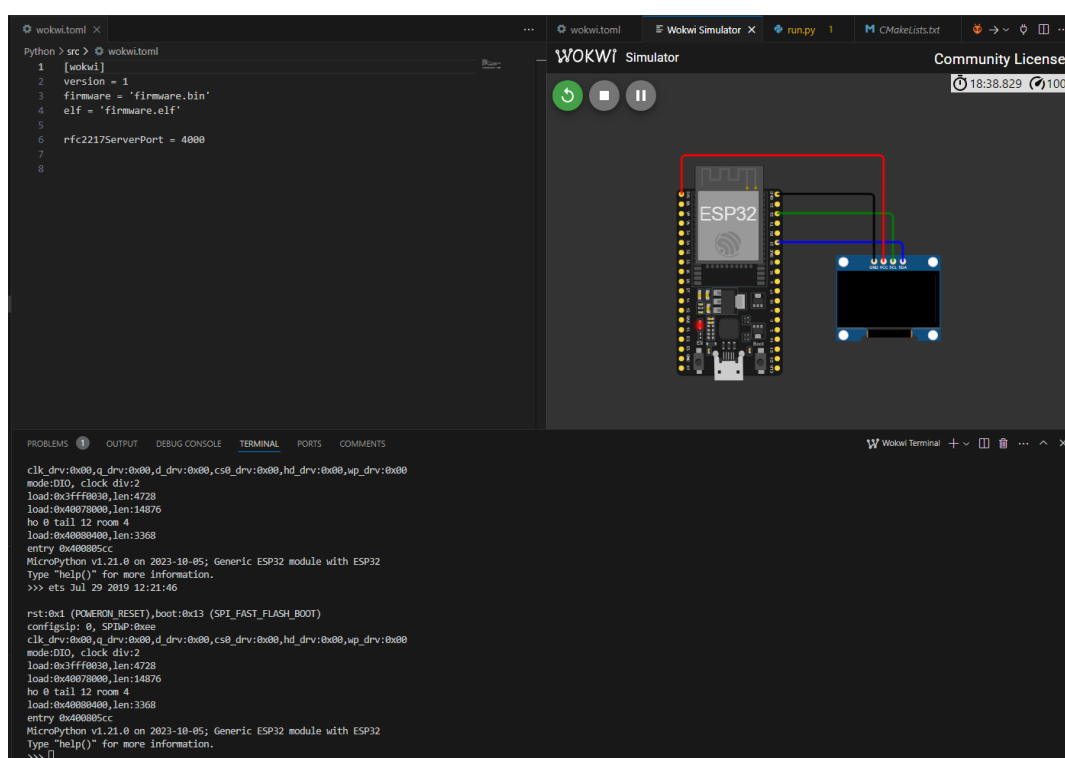


Рисунок 1 – Настроенная виртуальная среда для отладки микроконтроллера

Для корректного запуска эмулятора требуется предварительная настройка, включающая создание двух конфигурационных файлов. Первый файл, «wokwi.toml», определяет путь к прошивке микроконтроллера, а второй, «diagram.json», описывает подключенные периферийные модули и их конфигурацию. После успешного запуска эмулятора в терминале становится доступен интерпретатор MicroPython, что позволяет взаимодействовать с микроконтроллером через интерпретатор.

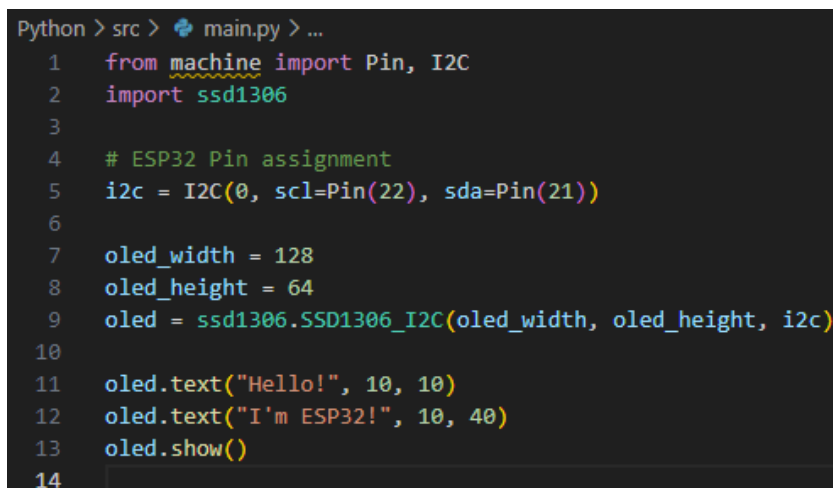
Разработчик может работать в двух основных режимах. В интерактивном режиме интерпретатора MicroPython удобно разрабатывать и тестировать небольшие программы, которые взаимодействуют с микроконтроллером для выполнения задач, таких как вычисления, измерения,

обработка сигналов или управление простыми устройствами. Второй режим предполагает использование прошивки, при которой микроконтроллер автоматически исполняет код, записанный в файле «main.py», при каждом запуске. Это позволяет создавать автономные встраиваемые системы, не требующие постоянного взаимодействия с компьютером.

Для загрузки скриптов на микроконтроллер может быть использована утилита ampy, которая обеспечивает передачу файлов через последовательный интерфейс. После установки утилиты синтаксис команд для загрузки файлов выглядит следующим образом:

```
ampy --port COMX put main.py
```

Где X — номер COM-порта, к которому подключен микроконтроллер. После успешной загрузки файла «main.py» устройство автоматически начнет выполнение загруженного скрипта. Пример скрипта на MicroPython представлен на рисунке (рис. 2).



```
Python > src > main.py > ...
1  from machine import Pin, I2C
2  import ssd1306
3
4  # ESP32 Pin assignment
5  i2c = I2C(0, scl=Pin(22), sda=Pin(21))
6
7  oled_width = 128
8  oled_height = 64
9  oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
10
11 oled.text("Hello!", 10, 10)
12 oled.text("I'm ESP32!", 10, 40)
13 oled.show()
14
```

Рисунок 2 – Код скрипта «main.py»

Этот код на Python предназначен для работы с OLED-дисплеем. Сначала импортируются необходимые библиотеки: Pin и I2C из модуля machine, а также библиотека ssd1306 для взаимодействия с дисплеем. Затем создается объект I2C, которому назначаются выводы SCL и SDA. После этого определяются размеры дисплея: ширина 128 пикселей и высота 64 пикселя. Создается объект ssd1306.SSD1306_I2C, который инициализирует дисплей с заданными размерами и интерфейсом I2C. В коде также используется метод text для отображения двух строк текста: "Hello!" и "I'm ESP32!", каждая из которых размещается в определенной позиции на экране. Наконец, метод show обновляет дисплей, чтобы отобразить добавленный текст.

Результат работы скрипта после его загрузки видна на рисунке (рис. 3).

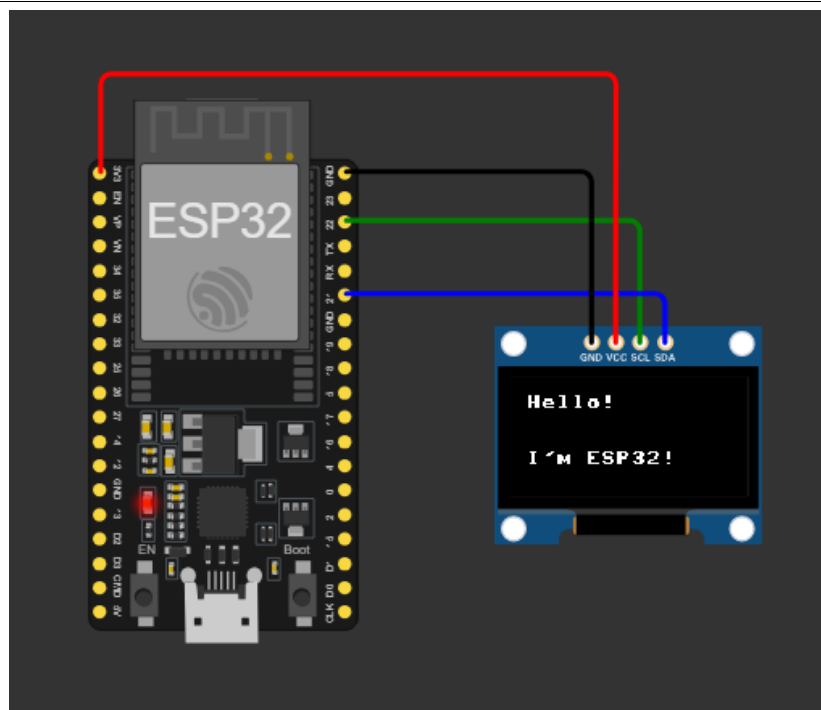


Рисунок 1 – Состояние эмулятора

Использование MicroPython для разработки встраиваемых устройств предоставляет множество преимуществ, однако наряду с этим сопровождается и определенными ограничениями. К числу положительных сторон можно отнести его высокую доступность для начинающих разработчиков благодаря простоте синтаксиса и низкому порогу вхождения. Поддержка множества популярных микроконтроллеров и широкая экосистема библиотек позволяют создавать функциональные прототипы за минимальное время. Встроенная интерактивная среда разработки, возможность быстрой отладки с помощью REPL и гибкость в работе с периферийными устройствами делают MicroPython удобным инструментом для задач, где требуется оперативная разработка и тестирование.

Однако использование MicroPython имеет и ограничения, которые необходимо учитывать при выборе этой технологии. Производительность интерпретируемого кода ниже по сравнению с решениями на языках, компилируемых в машинный код, таких как C или C++. Это может стать существенным недостатком для задач, требующих высокой скорости выполнения или оптимального использования ресурсов.

3. Выводы

Использование MicroPython значительно упрощает разработку встраиваемых устройств благодаря удобству языка Python и доступности готовых библиотек. Технология позволяет эффективно решать задачи программирования микроконтроллеров, даже при ограниченных аппаратных ресурсах.

Библиографический список

1. Беспалов Н. Н. Программная реализация управляющего устройства на ПЛИС для обмена данными по интерфейсу SPI // Научно-технический вестник Поволжья. 2016. № 3. С. 75-78.
2. Катукия Г. Т. Анализ и сравнение интерфейсов передачи данных на микроконтроллерах // Политехнический молодежный журнал. 2020. № 7(48). С. 2
3. Мешков В. Архитектура файловой системы FAT // Системный администратор. 2004. № 2(15). С. 42-54.
4. Деткова А. В. Модуль для проверки плат семейства Arduino // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2023. № 1. С. 43-49.
5. MicroPython URL: <https://micropython.org/> (дата обращения: 29.01.2025)
6. Visual Studio Code. URL: <https://code.visualstudio.com/> (дата обращения: 29.01.2025).
7. Wokwi. URL: <https://wokwi.com/> (дата обращения: 29.01.2025).