

ОТКРЫТОЕ СООБЩЕСТВО РАЗРАБОТЧИКОВ, ПРОГРАММИСТОВ  
И ПОЛЬЗОВАТЕЛЕЙ МИКРОКОНТРОЛЛЕРА K1948BK018 MIK32 АМУР  
[http://vk.com/mik32\\_amur](http://vk.com/mik32_amur)

---

НАРОДНАЯ  
ИНТЕГРИРОВАННАЯ СРЕДА РАЗРАБОТКИ  
ДЛЯ МИКРОКОНТРОЛЛЕРА K1948BK018  
(MIK32 Амур)

КРАТКОЕ РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Версия 0.1 (2024-08-10)

## СОДЕРЖАНИЕ

Введение.....	3
1. Установка и настройка среды разработки .....	6
1.1. Установка VS Code и расширений для программирования на языке Си .....	6
1.2. Установка расширения PlatformIO для VS Code .....	8
1.3. Установка драйвера для программатора .....	10
1.4. Установка платформы для программирования МК Амур .....	13
2. Краткое описание интерфейса расширения PlatformIO для VS Code .....	17
3. Быстрый старт .....	19
3.1. Создание проекта на основе примера .....	19
3.2. Типовая структура проекта.....	22
3.3. Создание нового (пустого) проекта .....	24

## Введение

Настоящее руководство предназначено для ознакомления пользователей с назначением компонентов кроссплатформенной народной<sup>1</sup> интегрированной среды разработки (далее — ИСП) для микроконтроллера K1948BK018<sup>2</sup> (далее — MIK32, МК Амур) и порядком её установки.

ИСП для МК Амур разрабатывается и поддерживается сообществом разработчиков, программистов и пользователей микроконтроллера MIK32 Амур ([https://vk.com/mik32\\_amur](https://vk.com/mik32_amur)) инициативно, за счёт личных сил и средств, без привлечения коммерческого капитала.

В состав ИСП для МК Амур входят:

- текстовый редактор VS Code;
- расширение PlatformIO для VS Code;
- набор инструментов для программирования микроконтроллеров на основе архитектуры RISC-V (xPack toolchain);
- инструмент для отладки на кристалле, внутрисистемного программирования и тестирования методом граничного сканирования (OpenOCD);
- программная платформа разработки MIK32 Platform для PlatformIO;
- программная платформа разработки приложений для МК Амур, разрабатываемая сообществом (народная);
- программная платформа разработки приложений для МК Амур, поставляемая в составе ИСП от АО Микрон (официальная);
- инструмент для загрузки приложений во все виды памяти МК Амур;
- исходные коды базовых примеров приложений для МК Амур.

За основу разработки ИСП для МК Амур на базе PlatformIO был использован материал опубликованный в репозитории <https://github.com/MikronMIK32>. В настоящее время, в виду значительной переработки исходного кода и расширения функционала, а также изменения ряда функций платформы разработки данные сборки являются не совместимыми. Поэтому, для старых проектов (в том числе при их

---

<sup>1</sup> Под значением слова народная мы понимаем: общая, общедоступная, общественная, открытая, национальная. Также народным называют то, что имеет значение для всего населения страны (см. Толковый словарь Дмитриева).

<sup>2</sup> Микросхема K1948BK018 (коммерческое название MIK32 АМУР) — это 32-х битный микроконтроллер на основе процессорного ядра RISC-V. Разрабатывается и производится предприятием «Микрон» (г. Зеленоград).

миграции на новую платформу) можно использовать официальный комплект разработки, а для новых проектов, воспользоваться сборкой от сообщества.

Несмотря на внушительный объём входящего в состав ИСП программного обеспечения, в том числе сторонних разработчиков, мы приложили максимум усилий, чтобы процесс установки и настройки был простым и понятным пользователям, которые только начинают изучать программирование МК Амур.

Благодарим Вас за выбор нашей интегрированной среды разработки для МК Амур, желаем легкой учебы и продуктивной работы!

Если у вас возникнут вопросы или замечания по работе ИСП, в т.ч. предложения по её совершенствованию, пишите их в соответствующем разделе обсуждений на сайте группы ВКонтакте ([https://vk.com/mik32\\_amur](https://vk.com/mik32_amur)).

Загружая, устанавливая и/или используя данную инструментальную среду разработки приложений для МІК32 Амур, вы соглашаетесь, что

ДАННОЕ ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ПРЕДОСТАВЛЯЕТСЯ «КАК ЕСТЬ», БЕЗ КАКИХ-ЛИБО ГАРАНТИЙ, ЯВНО ВЫРАЖЕННЫХ ИЛИ ПОДРАЗУМЕВАЕМЫХ, ВКЛЮЧАЯ ГАРАНТИИ ТОВАРНОЙ ПРИГОДНОСТИ, СООТВЕТСТВИЯ ПО ЕГО КОНКРЕТНОМУ НАЗНАЧЕНИЮ И ОТСУТСТВИЯ НАРУШЕНИЙ, НО НЕ ОГРАНИЧИВАЯСЬ ИМИ. НИ В КАКОМ СЛУЧАЕ АВТОРЫ НЕ НЕСУТ ОТВЕТСТВЕННОСТИ ПО КАКИМ-ЛИБО ИСКАМ, ЗА УЩЕРБ ИЛИ ПО ИНЫМ ТРЕБОВАНИЯМ, В ТОМ ЧИСЛЕ, ПРИ ДЕЙСТВИИ КОНТРАКТА, ДЕЛИКТЕ ИЛИ ИНОЙ СИТУАЦИИ, ВОЗНИКШИМ ИЗ-ЗА ИСПОЛЬЗОВАНИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИЛИ ИНЫХ ДЕЙСТВИЙ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ.

НАСТОЯЩИЙ ДОКУМЕНТ ЯВЛЯЕТСЯ НЕОТЪЕМЛЕМОЙ ЧАСТЬЮ народной интегрированной среды разработки приложений для МІК32 Амур.

©Mikron (<https://mikron.ru>)

©МІК32 Амур (<https://mikron.ru/products/mikrokontrollery/mk32-amur>)

©VS Code (<https://code.visualstudio.com>)

©PlatformIO (<https://platformio.org>)

©Windows (<https://support.microsoft.com/ru-ru/windows>)



1. Обращаем внимание, что VS Code и PlatformIO собирают, накапливают и передают телеметрию<sup>3</sup>. Подробнее о составе и объеме собираемых данных, а также способах отключения телеметрии можно узнать на официальных сайтах данных программных продуктов<sup>4</sup>.

2. К большому нашему сожалению, разработчик PlatformIO отклонился от истинного пути, и использует своё открытое программное обеспечение в политических целях, чем нарушает главные принципы Open Source!

3. На сайте документации и репозитории GitHub PlatformIO автором программного обеспечения размещаются призывы к противоправной деятельности, поэтому рекомендуем вам воздержаться от посещений данных ресурсов, не поддаваться на провокации, при необходимости использовать плагины веб-браузера для блокировки всплывающей рекламы и нежелательных элементов на веб-страницах.

\* плакат «Враг коварен», источник — <https://darminaopel.ru/library/vrag-ne-dremlet-kartinki.html>

---

<sup>3</sup> Телеметрия — это совокупность способов сбора информации и измерение параметров, позволяющих получить сведения о пользователе. Дословно с греческого термин переводится как «удаленное измерение».

<sup>4</sup> Для отключения телеметрии пользователя в PlatformIO выполните команду (из терминала PlatformIO) — `pio settings set enable_telemetry no`

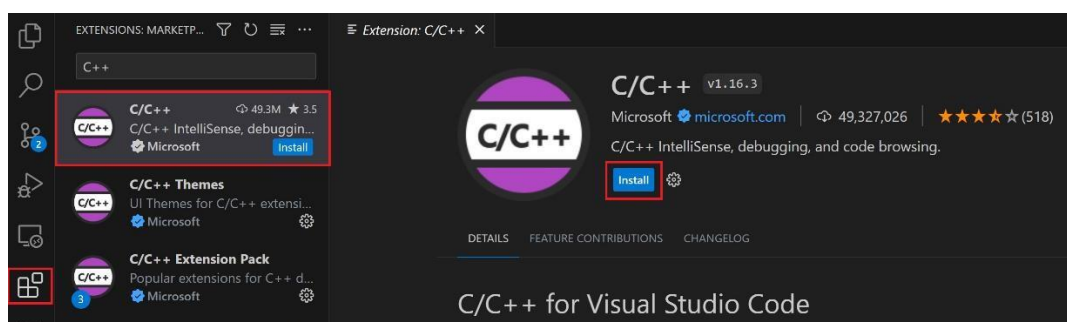
## 1. Установка и настройка среды разработки

### 1.1. Установка VS Code и расширений для программирования на языке Си

Текстовый редактор Visual Studio Code (VS Code), разрабатывается Microsoft для Windows, Linux и macOS. Позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб- и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, IntelliSense<sup>5</sup> и средства для рефакторинга<sup>6</sup>. Имеет широкие возможности для настройки интерфейса: пользовательские темы, сочетания клавиш и файлы конфигурации. Распространяется бесплатно, разрабатывается как программное обеспечение с открытым исходным кодом, но готовые сборки распространяются под проприетарной лицензией.

Для установки редактора и расширений выполните следующие шаги:

- 1) Загрузите дистрибутив с официального сайта VS Code <https://code.visualstudio.com/Download> для вашей операционной системы. Выполните установку программы, следуя инструкции, размещённой на официальном сайте <https://code.visualstudio.com/docs/setup/setup-overview>.
- 2) Запустите VS Code.
- 3) Откройте панель «Расширения» (Extensions) введите в поисковую строку значение «C++» (латинская раскладка)



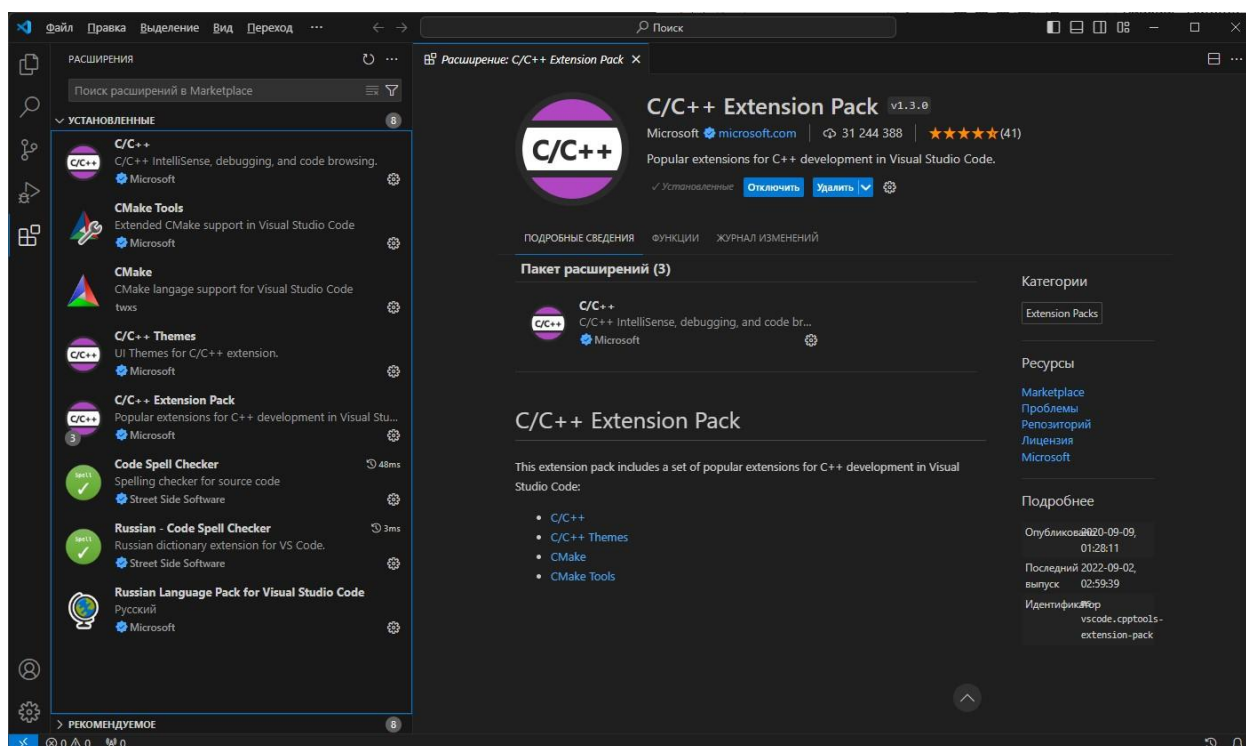
<sup>5</sup> IntelliSense — это технология автодополнения кода Microsoft. Дописывает название функции при вводе начальных букв. Кроме прямого назначения, IntelliSense используется для доступа к документации и для устранения неоднозначности в именах переменных, функций и методов.

<sup>6</sup> Рефакторинг (англ. refactoring) — это процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

Выберите пакет расширений C/C++ Extension Pack и нажмите кнопку «Установить» (Install), расположенную в верхней части карточки расширения, выведенной в основном окне приложения. Данный пакет включает в себя расширение поддержки языков программирования Си/Си++, а также инструменты для поддержки редактором системы сборки CMake.

4) *Дополнительно.* Для удобства работы в текстовом редакторе можно установить следующие расширения:

- Russian Language Pack for Visual Studio Code — языковой пакет для русского языка (переопределяет язык интерфейса VS Code);
- Russian - Code Spell Checker — расширение для проверки орфографии русского языка. При установке данного расширения также будет установлен инструмент для проверки орфографии Code Spell Checker.



На рисунке приведён результат выполнения установки VS Code и расширений для поддержки программирования на языках Си/Си++. Далее в руководстве будут приводиться снимки экрана и описание интерфейса с установленным в редактор языковым пакетом для русского языка.

**Примечание.** Установка расширения Russian - Code Spell Checker автоматически не включает проверку орфографии русского языка. Для активации этой функции необходимо нажать клавишу «F1»,



и в появившемся диалоговом окне ввести по очереди, следующие две команды:

- Enable Russian Spell Checker Dictionary
- Enable Russian Spell Checker Dictionary in Workspace

**Примечание.** Расширения для поддержки языков программирования для Си/Си++ в VS Code не содержат компиляторов C/C++, отладчиков GDB, линковщиков, систему сборки CMake и систему управления версиями Git. Данные расширения только добавляют функции поддержки инструментов разработки в текстовом редакторе.

5) *Дополнительно.* Если вы планируете использовать возможности системы управления версиями (СУВ) Git, то скачайте и установите соответствующий дистрибутив: для ОС Windows — <https://git-scm.com/downloads>, для ОС Linux — СУВ можно установить из репозитория сборки, например, командой (требуется права root):

```
apt-get install git
```

6) *Дополнительно.* Если у вас есть необходимость в использовании системы сборки CMake, то скачайте и установите дистрибутив: для ОС Windows — <https://cmake.org/download/>, для ОС Linux — установите приложение из репозитория, например, командой:

```
apt-get install cmake
```

## 1.2. Установка расширения PlatformIO для VS Code

PlatformIO — это кроссплатформенный набор инструментов разработки приложений для микроконтроллеров. Распространяется по лицензии Apache 2.0. Набор включает в себя утилиту командной строки, позволяющую автоматизировать достаточно сложные для начинающих разработчиков процессы компиляции, загрузки приложений в память микроконтроллера, отладки и модульного тестирования приложений непосредственно на самом устройстве. Набор инструментов позиционируется разработчиком, как универсальная среда разработки для IoT приложений (Next-Gen IDE). Изначально PlatformIO ориентирован на работу из командной строки, имеет встроенные генераторы проектов для: Atom, CLion, Eclipse, Emacs, NetBeans, Qt Creator, Sublime Text, Vim и Visual Studio, а также плагины для интеграции своих функций в перечисленные текстовые редакторы.

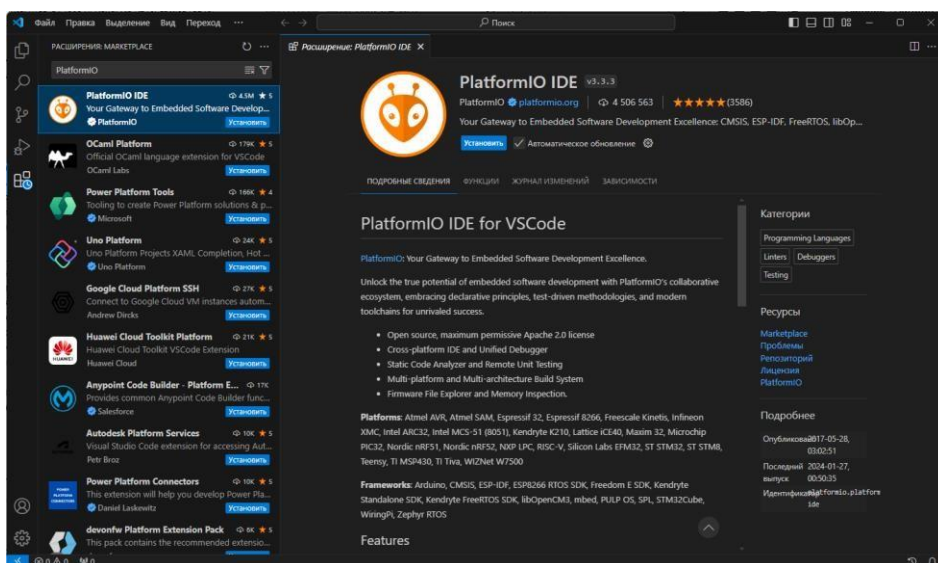


PlatformIO для VS Code — это расширение, которое добавляет поддержку функции PlatformIO в текстовый редактор VS Code, в виде дополнительных экранных форм и элементов интерфейса. Расширение встраивает в интерфейс редактора командные кнопки для запуска: компиляции, загрузки, отладки и модульного тестирования приложений для микроконтроллера. Также, расширение интегрирует в редактор собственный монитор последовательного порта для вывода информации в отдельную панель VS Code. Расширение PlatformIO для VS Code включает в себя встроенный установщик ядра PlatformIO, и устанавливает его автоматически.

**Примечание.** PlatformIO представляет собой набор инструментов запуск которых осуществляется с помощью скриптов, написанных на языке программирования Python. В PlatformIO все скрипты Python запускаются в изолированной среде (виртуальном окружении Python), но несмотря на это *в ОС Windows существует несовместимость PlatformIO с пакетом Python, распространяемым через магазин Windows (с официальным дистрибутивом Python такая проблема не наблюдалась).*

Для установки расширения PlatformIO для VS Code выполните следующие шаги:

- 1) Запустите приложение VS Code.
- 2) Откройте панель «Расширения», нажав на одноимённую кнопку на основной боковой панели.
- 3) Введите в строку поиска «PlatformIO».
- 4) Выберите в появившемся списке расширений «PlatformIO IDE».



- 5) Ознакомьтесь с информацией, отображаемой в карточке расширения.

- 6) Нажмите на кнопку «Установить».
- 7) Дождитесь окончания установки и перезапустите редактор VS Code.

### 1.3. Установка драйвера для программатора

Программатор для МК Амур — это аппаратно-программное устройство, предназначенное для записи/считывания информации в запоминающие устройства микроконтроллера (системное ОЗУ, встроенное ЭСППЗУ<sup>7</sup>, внешнюю память с последовательным интерфейсом QSPI).

В отладочных платах на основе МК Амур, как правило, для загрузки приложений (прошивок) в запоминающее устройство применяются бюджетные программаторы на основе чипов серии FT2232H. Они могут быть встроены непосредственно в отладочную плату (например — отладочная плата NUKE МІК32 v.0.3, производство компании Микрон) или поставляться в виде отдельного устройства (например — программатор ELJTAG, производство компании Элрон).

До начала использования программатора необходимо выполнить ряд настроек, причём содержание этих настроек будет отличаться в зависимости от используемой вами операционной системы.

#### 1.3.1. Настройка программатора в ОС Windows

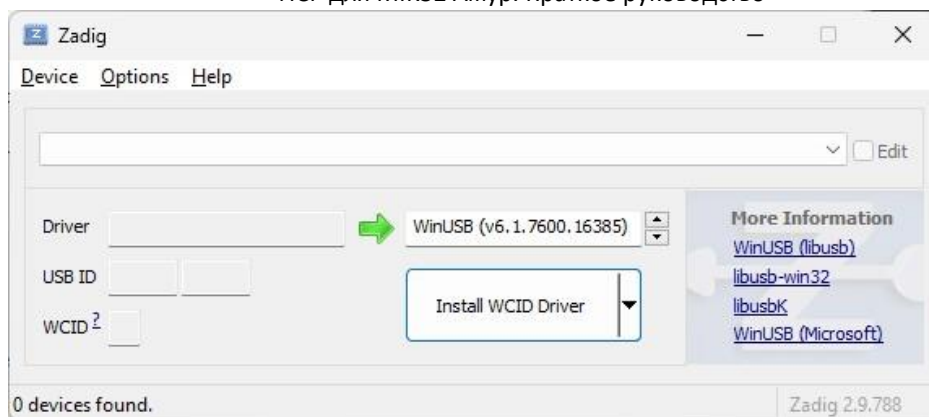
Для использования программатора в ОС Windows необходимо установить дополнительные драйверы USB-устройств. Упростить процесс их установки, можно воспользовавшись утилитой Zadig, которая возьмёт на себя все рутинные операции по установке специальных драйверов в ОС Windows.

Для настройки программатора выполните следующие шаги:

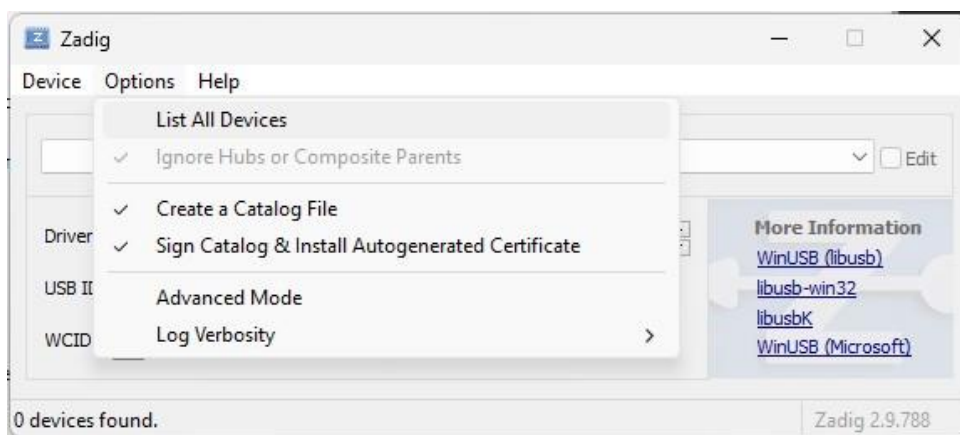
- 1) Загрузите утилиту Zadig, раздел Download официального сайта (<https://zadig.akeo.ie/>).
- 2) Подключите программатор к USB-порту ПЭВМ.
- 3) Запустите утилиту Zadig (требуется права администратора).

---

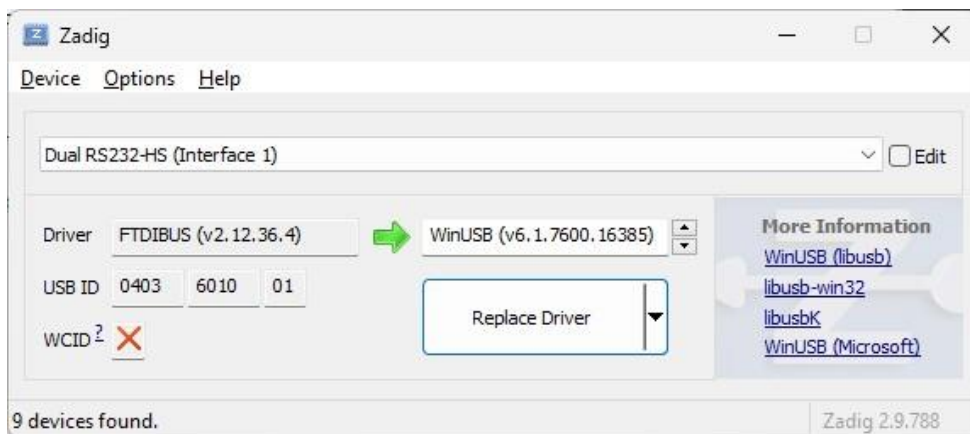
<sup>7</sup> ЭСППЗУ (англ. Electrically Erasable Programmable Read-Only Memory, EEPROM) — электрически стираемое перепрограммируемое ПЗУ.



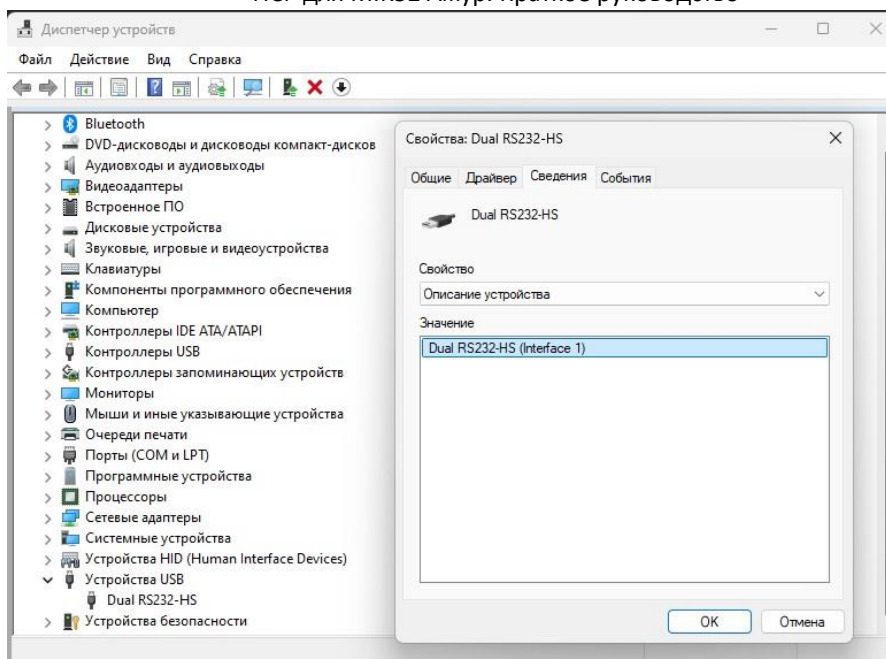
- 1) В меню «Options» выберите пункт «List All Devices».



- 2) В выпадающем списке выберите устройство «Dual RS232-HS (Interface 1)».



- 3) Убедитесь, что в выпадающем списке «Driver» установлено значение «WinUSB (v6.1.x.x)», если нет, то выберите его.
- 4) Нажмите на кнопку «Replace Driver».
- 5) После установки в разделе устройства USB у вас должно появиться устройство Dual RS232-HS.



**Важно!** В программаторах, поставляемых с МК Амур «Interface 1» используется для загрузки приложений (FTDI интерфейс), а «Interface 0» для строковой отладки приложений (виртуальный COM-порт).

### 1.3.2. Настройка программатора в ОС семейства Linux

В отличие от ОС Windows в ОС Linux установка дополнительных драйверов для программатора не требуется, но для работы с ним потребуется настроить права доступа к внешнему USB-устройству.

Для получения доступа к программатору необходимо создать конфигурационный файл для службы udev<sup>8</sup>.

Для конфигурации udev Выполните следующие шаги (требуется права пользователя root):

1) В терминале с помощью текстового редактора создайте файл `99-mik32-udev.rules` в каталоге `/etc/udev/rules.d/` со следующим содержанием:

```
SUBSYSTEM=="usb", ATTR{idVendor}=="0403", ATTR{idProduct}=="6010",
GROUP=="plugdev", MODE="0666"
```

<sup>8</sup> udev (userspace /dev) — управление устройствами для новых версий ядра Linux, являющийся преемником devfs, hotplug и HAL. Его основная задача — обслуживание файлов устройств (англ. device nodes) в каталоге /dev и обработка всех действий, выполняемых в пространстве пользователя при добавлении/отключении внешних устройств, включая загрузку firmware.

- 2) Сохраните файл на диск.
- 3) Перезагрузите службу udev используя команду

```
service udev restart
```

- 4) Добавьте вашу учетную запись в группы plugdev и dialout<sup>9</sup> с помощью следующей команды:

```
usermod -aG plugdev dialout
```

- 5) Отключите программатор и подключите его повторно.

**Примечание.** Если вы уже используете PlatformIO для программирования микроконтроллеров, то скорее всего у вас в каталоге `/etc/udev/rules.d/` имеется файл `99-platformio-udev.rules`. Соответственно, для настройки работы с программатором МК Амур вам можно выполнить вышеописанные шаги 1–4, либо добавить конфигурационные параметры из п. 1 в файл `99-platformio-udev.rules`.

#### 1.4. Установка платформы для программирования МК Амур

PlatformIO имеет децентрализованную архитектуру и предоставляет набор программных интерфейсов, которые должны быть реализованы в платформах разработки (далее — платформа) для конкретных целевых архитектур (классов) микроконтроллеров. Это позволяет унифицировать процедуры создания приложений для различных микроконтроллеров, упростить процесс разработки, и что немаловажно — снизить порог вхождения для новых пользователей.

Таким образом вся сложность настройки и использования инструментов сборки, загрузки и отладки приложений для конкретной архитектуры микроконтроллеров возлагается на разработчика платформы. При создании платформы для конкретной архитектуры (класса) микроконтроллеров разработчик реализует обязательный набор программных интерфейсов, требуемых для автоматизации выполнения компиляции, загрузки приложений в память микроконтроллера, настройки и запуска инструментов для отладки приложения с учётом специфики архитектуры. Также разработчиком

---

<sup>9</sup> dialout — это группа пользователей удаленного доступа, которым разрешен доступ к порту последовательного интерфейса (ttyX). По умолчанию только пользователь root является членом группы удаленного доступа.

определяются зависимости от сторонних средств разработки приложений. Совокупность правил создания приложений, файлов конфигурации платформы и описаний зависимостей от внешних инструментов позволяют автоматизировать, не только процесс разработки, но и установку и конфигурирование всех необходимых компонентов среды разработки.

Платформа разработки МК Амур для PlatformIO включает в себя следующие компоненты и зависимости:

*компоненты:*

- скрипты, реализующие интерфейсы сборки и загрузки, отладки приложений;

- Id-скрипты для линковки приложения, запускаемого из ОЗУ (RAM), ЭСППЗУ (EEPROM) или внешней памяти с последовательным интерфейсом QSPI;

- конфигурационные файлы для отладочных плат на основе МК Амур;

- SVD-файл с описанием регистров периферийных устройств МК Амур;

- программная платформа разработки приложений для МК Амур разрабатываемая сообществом (framework\_mik32\_folk);

- программная платформа разработки приложений для МК Амур поставляемая АО Микрон в составе ИСП для МК32 (framework\_mik32\_off);

- инструмент для загрузки приложений во все виды памяти МК Амур (tool\_uploader\_mik32);

- исходные коды базовых примеров приложений для МК Амур;

*зависимости:*

- набор инструментов для программирования микроконтроллеров на основе архитектуры RISC-V (xPack toolchain), версия 13.2.0;

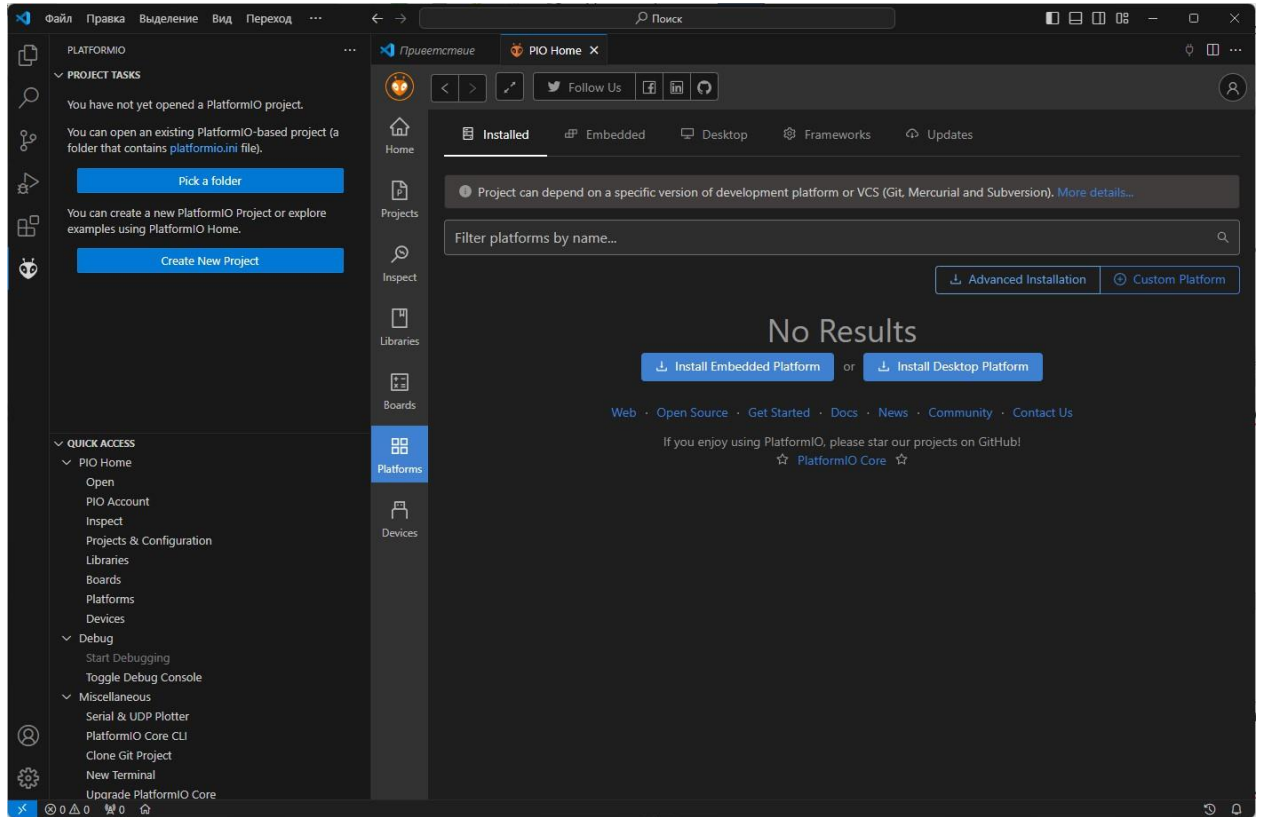
- инструмент для отладки на кристалле, внутрисистемного программирования и тестирования методом граничного сканирования (OpenOCD), версия 0.12 и новее.

Для установки платформы разработки МК Амур для PlatformIO выполните следующие шаги:

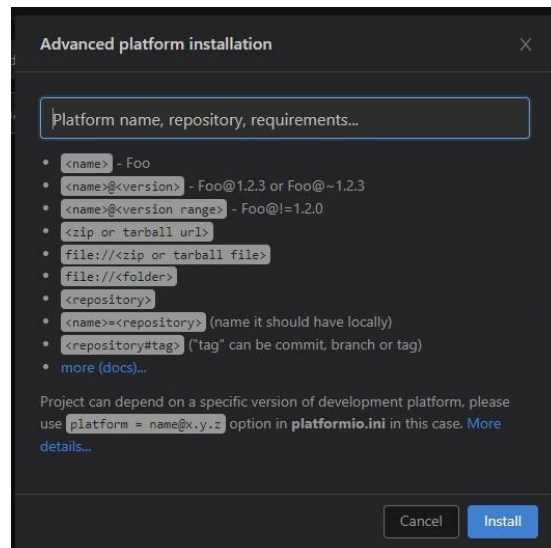
- 1) Запустите приложение VS Code.

- 2) Откройте домашнюю страницу PlatformIO с помощью команды «Open» в боковой панели или нажав на кнопку «PlatformIO:Home» на панели инструментов в строке состояния редактора.

3) Нажмите на кнопку «Platforms» в панели инструментов домашней страницы PlatformIO.



4) Во вкладке «Installed» нажмите на кнопку «Advanced Installation».



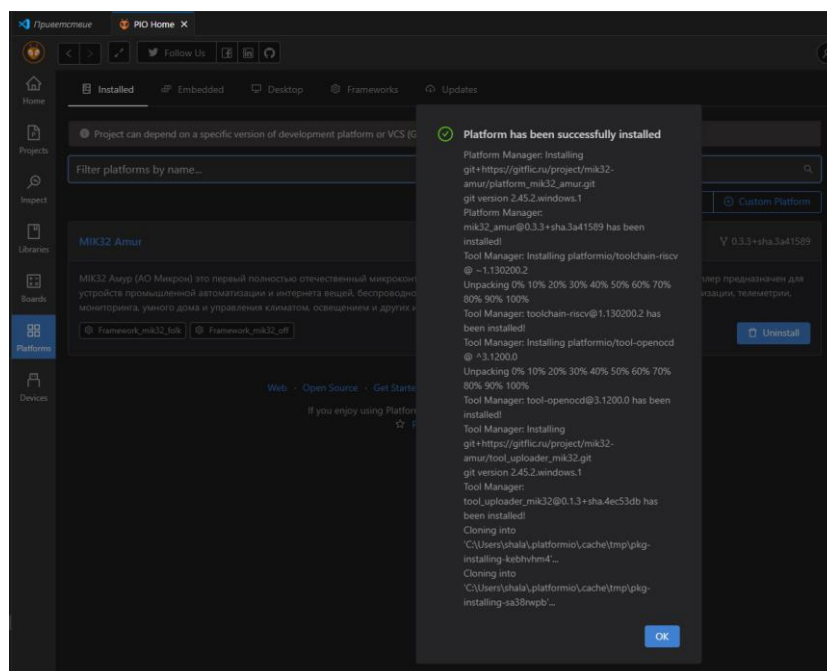
5) Скопируйте и вставьте в поле «Platform name, ...» ссылку на платформу разработки для МК Амур —

[https://gitflic.ru/project/mik32-amur/platform\\_mik32\\_amur.git](https://gitflic.ru/project/mik32-amur/platform_mik32_amur.git)

6) Нажмите на кнопку «Install».



7) Дождитесь окончания процесса установки платформы. По окончании установки будет выведено диалоговое окно следующего содержания

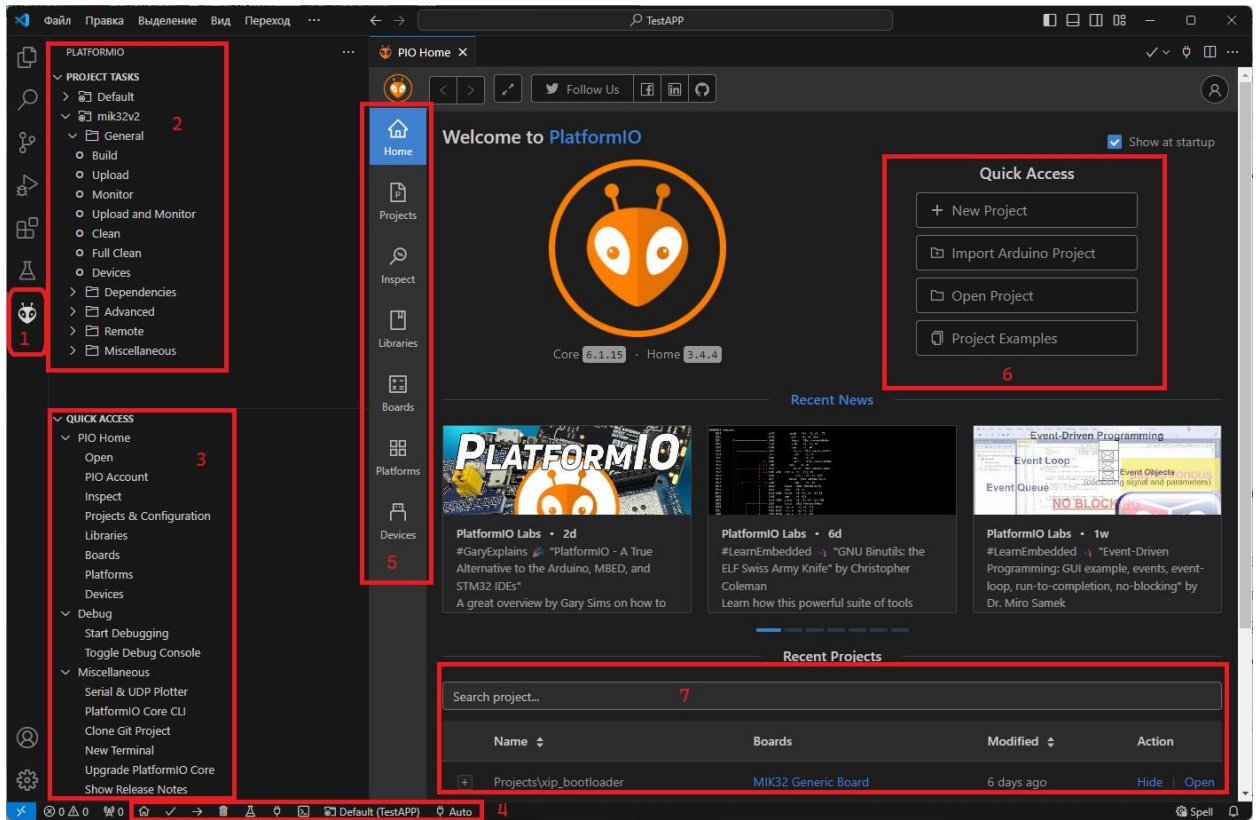


8) Нажмите на кнопку «ОК».

**ПОЗДРАВЛЯЕМ!!!** Вы только что установили основной набор компонентов ИСР для МК Амур. Теперь вы можете приступать к разработке своего первого приложения.

Как вы могли заметить, процесс установки ИСР достаточно прост и интуитивно понятен. Уделите часть своего времени на ознакомление с интерфейсом ИСР, это поможет вам быстрее освоиться и меньше отвлекаться от основного процесса разработки приложений.

## 2. Краткое описание интерфейса расширения PlatformIO для VS Code



1) Кнопка открывает/закрывает боковую панель расширения PlatformIO.

2) Вкладка «PLATFORMIO» содержит список задач управления текущим проектом.

«*Build*» — собрать<sup>10</sup> приложение;

«*Upload*» — загрузить приложение в память устройства;

«*Monitor*» — запустить встроенный монитор последовательного порта;

«*Upload and Monitor*» — загрузить приложение в устройство и открыть монитор последовательного порта;

«*Clean*» — очистить каталог сборки проекта;

«*Full clean*» — очистить каталог сборки проекта и удалить загруженные внешние библиотеки (libdeps);

«*Devices*» — вывести в терминал список доступных виртуальных COM-портов.

<sup>10</sup> Сборка (англ. build) — процесс получения информационного продукта из исходного кода. Включает компиляцию и компоновку, выполняется инструментами автоматизации.

3) Вкладка «QUICK ACCESS» предназначена для быстрого доступа к функциям PlatformIO.

«Open» — открыть домашнюю страницу PlatformIO в основном окне редактора;

«PIO Account» — открыть диалоговое окно для авторизации пользователя в системе PlatformIO Account;

«Inspect» — открыть управляющую форму статического анализатора кода;

«Project & Configuration» — открыть список проектов для доступа к форме настройки параметров проекта (экранные формы для редактирования содержания файла platformio.ini);

«Libraries» — открыть менеджер управления зависимостями от внешних библиотек;

«Boards» — открыть менеджер отладочных плат;

«Platforms» — открыть менеджер платформ разработки;

«Devices» — открыть форму отображения информации о подключенных устройствах (виртуальные COM-порты, логические разделы дисков и Multicast DNS);

«Start Debugging» — запустить отладку приложения на устройстве.

4) Панель инструментов PlatformIO для управления текущим проектом. Дублирует команды со вкладки «PLATFORMIO».

Первая кнопка открывает домашнюю страницу PlatformIO, остальные кнопки дублируют задачи управления текущим проектом вкладки «PLATFORMIO» (слева направо: собрать, загрузить, очистить, запустить модульные тесты, открыть монитор последовательного порта, открыть терминал, установить текущий проект, выбрать виртуальный COM-порт используемый по умолчанию).

5) Панель инструментов для запуска функций ядра и настройки параметров PlatformIO.

Панель инструментов дублирует команды доступа к функциям PlatformIO, описанных в п. 3.

6) Группа командных кнопок для управления проектами.

Командные кнопки для управления проектами позволяют (сверху вниз): создать новый проект, импортировать проект из среды Arduino, открыть существующий проект, открыть список примеров для отладочной платы.

7) Картотека созданных проектов с возможностью поиска и сортировки.

### 3. Быстрый старт

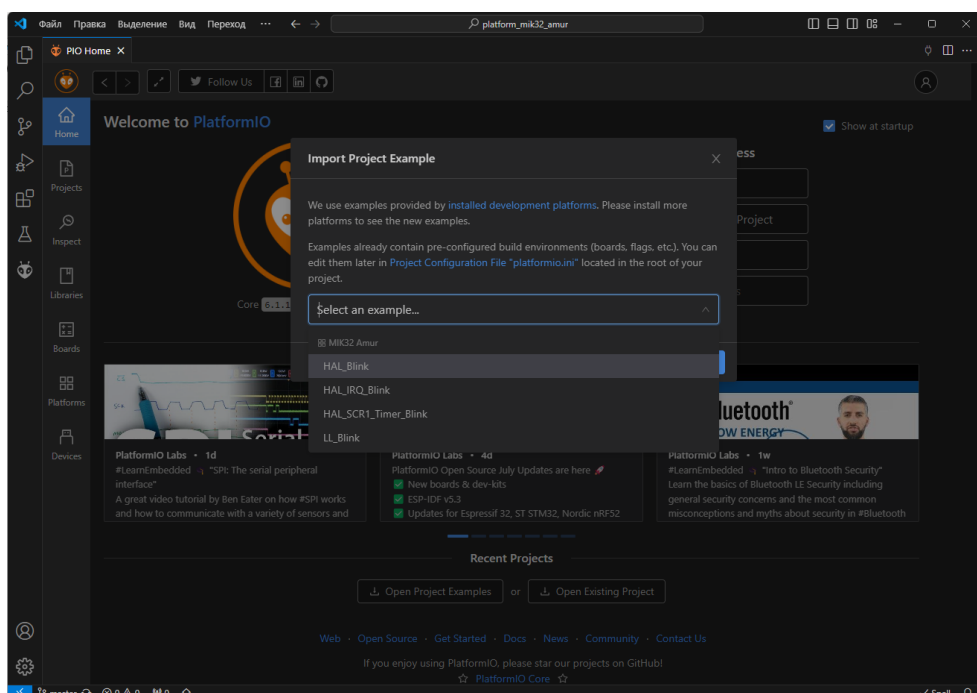
Расширение PlatformIO для VS Code позволяет создавать новые проекты двумя способами. Первый — на основе импорта примеров приложений, поставляемых в составе платформ разработки, и второй — пустой проект.

**Важно!** Проекты в PlatformIO имеют жестко регламентированную структуру, и ее изменение может привести к нарушению процесса сборки и/или работы приложения. Будьте внимательны! Описание структуры и назначения ее элементов приведено в подразделе 3.2.

#### 3.1. Создание проекта на основе примера

Для создания нового проекта на основе примера выполните следующие действия:

- 1) Запустите приложение VS Code.
- 2) Откройте боковую панель PlatformIO.
- 3) Откройте домашнюю страницу PlatformIO (команда «Open»).
- 4) В разделе «Quick Access» нажмите кнопку «Project Examples».
- 5) Выберите пример «HAL\_Blink» в выпадающем списке «Select an examples...».



Обратите внимание, что примеры сгруппированы по категориям, соответствующим платформам разработки, установленным в PlatformIO.

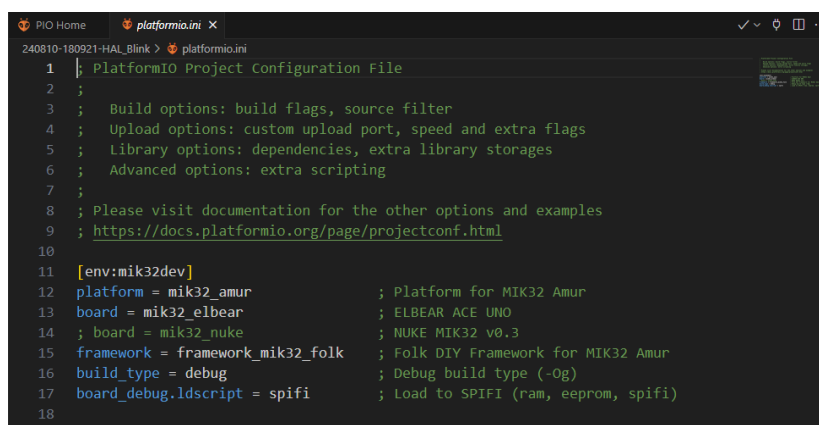
6) Нажмите на кнопку «Import». Среда разработки сохранит новый проект в папке проектов PlatformIO<sup>11</sup>.

После создания проект автоматически откроется в VS Code.

7) В появившемся диалоговом окне «Вы доверяете авторам файлов в этой папке?» нажмите кнопку «Да, я доверяю авторам».

8) Подключите отладочную плату к ПЭВМ через программатор.

9) После создания проекта среда разработки автоматически откроет в основном окне файл конфигурации проекта (platformio.ini). Для правильной работы примера необходимо указать используемую вами отладочную плату. Для этого необходимо убрать/установить символ комментария «;» перед названием платы (свойство board). На рисунке ниже приведен пример выбора отладочной платы ELBEAR ACE-UNO.



```
1 PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:mik32dev]
12 platform = mik32_amur           ; Platform for MIK32 Amur
13 board = mik32_elbear           ; ELBEAR ACE UNO
14 ; board = mik32_nuke           ; NUKE MIK32 v0.3
15 framework = framework_mik32_folk ; Folk DIY Framework for MIK32 Amur
16 build_type = debug             ; Debug build type (-Og)
17 board_debug.ldscript = spifi   ; Load to SPIFI (ram, eeprom, spifi)
18
```

**Важно!** Выбор одновременно двух отладочных плат, как и их одновременное отключение с помощью комментария приведет к ошибке.

**Примечание.** Для управления установкой/снятием комментария можно воспользоваться комбинацией клавиш «Ctrl+/**».**

10) С помощью параметра board\_debug.ldscript установите вид памяти, в которую будет загружаться проект. Доступные варианты: ram (ОЗУ), eeprom (встроенная флэш-память) и spifi (внешняя флэш-память). По умолчанию, платы запускают приложение из встроенной флэш памяти

---

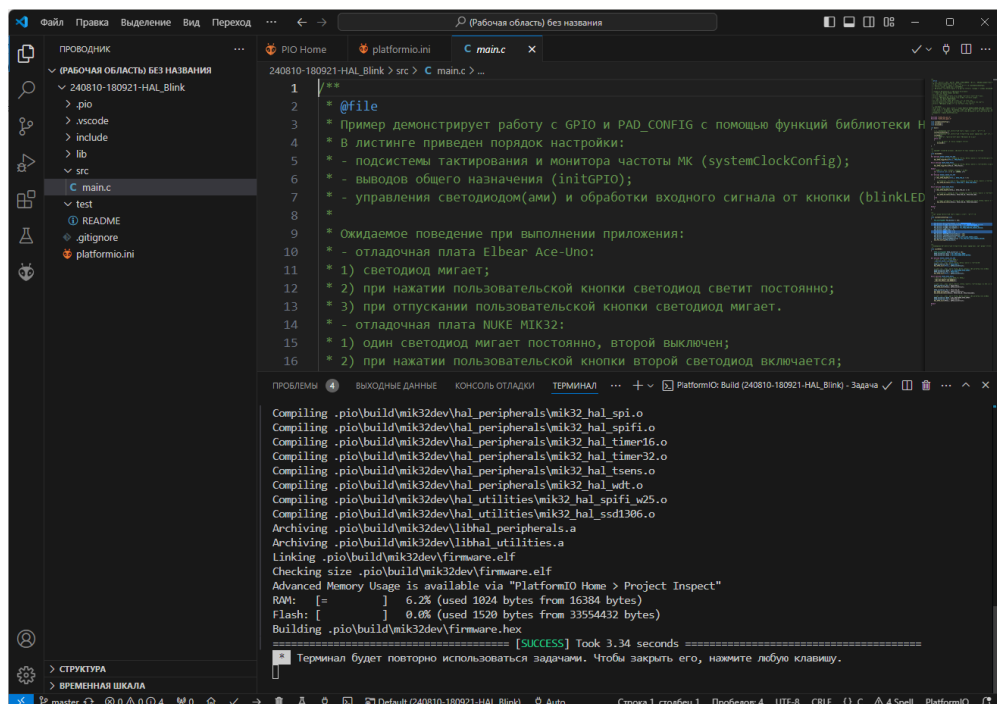
<sup>11</sup> По умолчанию все проекты сохраняются в папке PlatformIO/Projects в документах пользователя. К сожалению, PlatformIO некорректно работает с проектами если путь до проекта содержит символы кириллицы. Для изменения папки проектов по умолчанию, используйте команду (в терминале PlatformIO) — pio settings set projects\_dir путь\_до\_папки\_проектов

(EEPROM). Для выбора способа загрузки устройства смотрите руководство, поставляемое с вашей отладочной платой.

11) Сохраните файл (Файл->Сохранить, Ctrl+S). После сохранения файла настроек проекта среда разработки применит внесенные изменения в проект.

12) Откройте вкладку «Проводник» в VS Code, выберите файл main.c из каталога src.

13) Соберите проект командой «Build» на вкладке PlatformIO или нажмите соответствующую кнопку в строке состояния.



Во время сборки проекта в окно терминала будут выводиться шаги сборки и предупреждения компилятора. Если вы получили сообщение [SUCCESS] в конце, значит сборка проекта прошла успешно.

14) Выполните загрузку приложения в память контроллера, выберите команду «Upload» на боковой панели PlatformIO, или нажмите соответствующую кнопку в строке состояния.

## ИСП для МІК32 Амур. Краткое руководство

```

ПРОБЛЕМЫ 4 ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ
Начальный адрес линейной записи: 0x80000000
Настройка тактовой генерации контроллера интерфейса SPIFI
Отправляем команду 'Reset' через SPIFI в режиме 'QPI'
Отправляем команду 'Reset' через SPIFI в режиме 'SPI'
JEDEC ID = EFh 7018h
Режим 'Quad SPI' флеш-памяти: РАЗРЕШЕН
Выбран рабочий режим 'Single SPI'
Стираем сектор 0x00000000...
Записываем страницу 0x00000000... 0%
Записываем страницу 0x00000100... 16%
Записываем страницу 0x00000200... 33%
Записываем страницу 0x00000300... 50%
Записываем страницу 0x00000400... 66%
Записываем страницу 0x00000500... 83%
Запись страниц во внешнюю память через SPIFI успешно завершена
[18:23:35] Записано 1536 байт за 1,48 секунд (в среднем 1,0 кбайт/с)
===== [SUCCESS] Took 3.65 seconds =====
* Терминал будет повторно использоваться задачами. Чтобы закрыть его, нажмите любую к
лавишу.

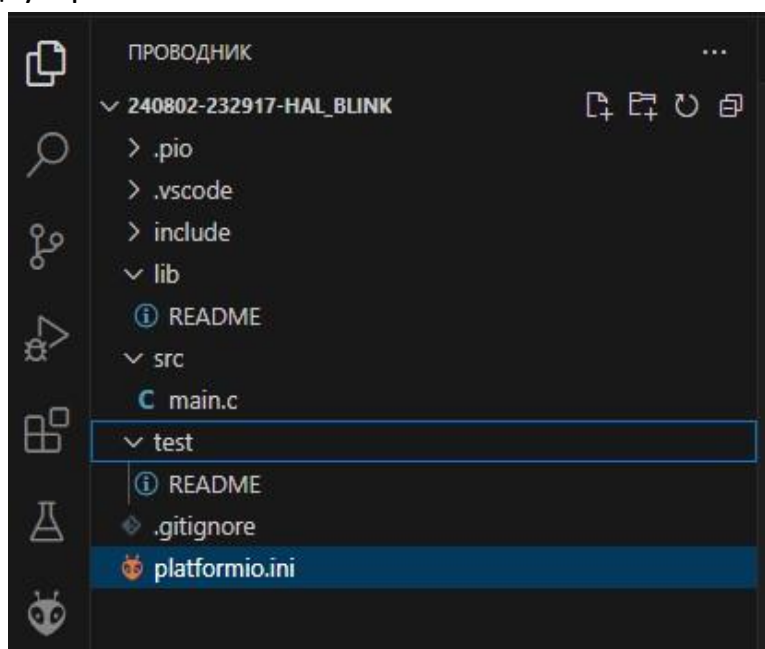
```

Во время загрузки приложения в память МК Амур в окно терминала будут выводиться сообщения, если после окончания работы загрузчика вы получили сообщение [SUCCESS], значит процесс завершился без ошибок.

15) После загрузки приложения в память МК Амур, на отладочной плате ELBEAR ACE-UNO должен мигать пользовательский светодиод, а при нажатии и удержании пользовательской кнопки — светить (на плате NUKE МІК32 v0.3 один светодиод будет мигать постоянно, а при нажатии или удержании пользовательской кнопки будет светить второй светодиод).

### 3.2. Типовая структура проекта

Структура типового проекта PlatformIO состоит из шести папок (каталогов) и двух файлов.





.pio/	Служебный каталог, предназначен для хранения файлов сборки, а также внешних библиотек, используемых в проекте.
.vscode/	Служебный каталог, содержит конфигурационные файлы для редактора VS Code. Данные файлы генерируются PlatformIO автоматически, не рекомендуется вручную вносить в их изменения.
Include/	Каталог проекта, предназначен для хранения заголовочных файлов проекта.
lib/	Каталог проекта, предназначен для хранения внутренних библиотек проекта. Структура данного каталога жестко регламентирована, подробнее см. файл README, расположенный в данной папке.
src/	Каталог проекта, предназначен для хранения исходных кодов программ. Только один файл из данного каталога должен содержать функцию main().
test/	Каталог проекта, предназначен для хранения файлов модульных тестов (фреймворк Unity).
.gitignore	Служебный файл системы управления версиями (СУВ) git, содержит информацию о каталогах, которые не должны индексироваться.
platformio.ini	Файл конфигурации проекта PlatformIO.

Конфигурационный файл проекта platformio.ini играет важную роль в настройке параметров сборки, загрузки, отладки и других инструментов. Рассмотрим содержимое файла, создаваемое по умолчанию при создании нового проекта для МК Амур, а также наиболее часто используемые параметры

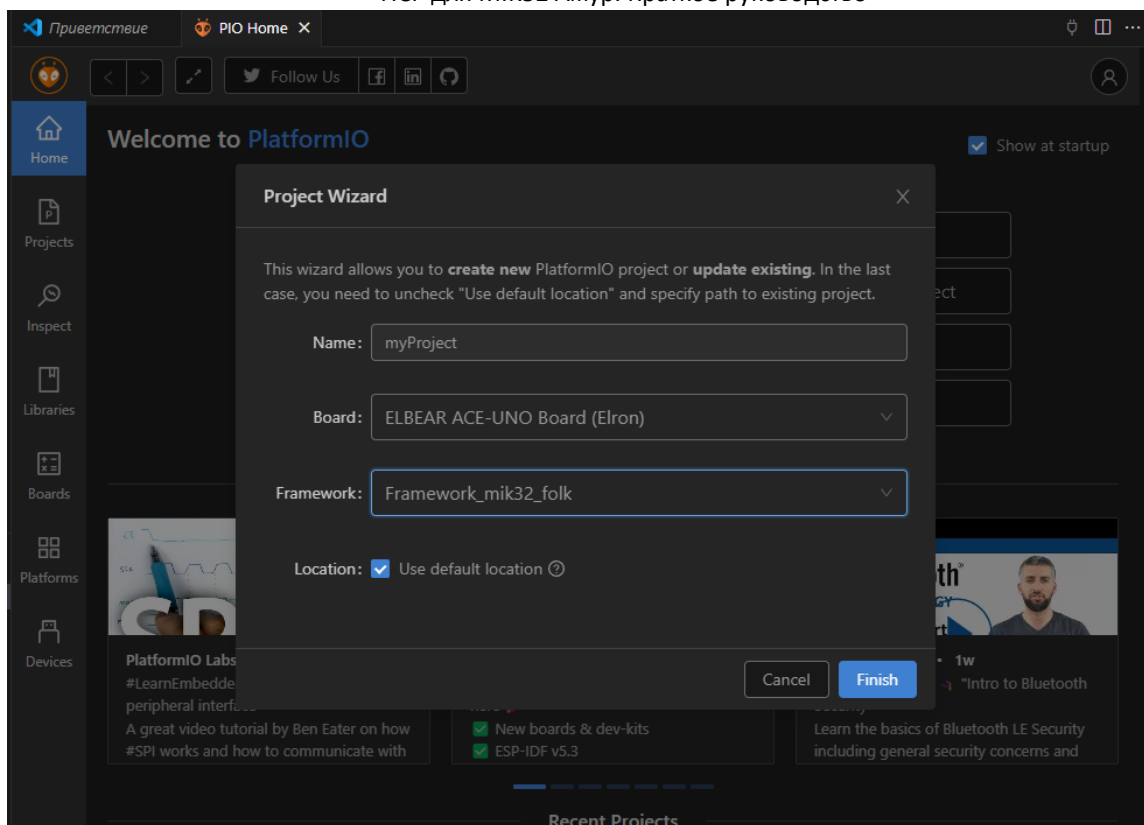
Параметр	
platform	Платформа, для которой создан проект
board	Вид отладочной платы, для которой создан проект
framework	Используемая в проекте программная платформа разработки приложений
build_type	<p>Переключает режимы оптимизации компилятора между отладкой (debug) и выпуском (release). При сборке проекта, соответственно будут использоваться опции компилятора -Og или -Os. По умолчанию (если параметр не задан) используется режим выпуска (Release)</p> <p><b>Важно!</b> При проведении отладки приложения с использованием программатора обязательно включайте режим отладки (debug). Так как компиляция с опцией -Os приводит к непредсказуемому поведению отладчика из-за отсутствия в бинарном файле приложения отладочной информации, необходимой для его работы.</p>

build_flags	Устанавливает опции управления препроцессором компилятора языка Си (например, -DMIK32_SPIFI, -DBOARD_LITE)
board_debug.ldscript	Задаёт сценарий компоновщика, который будет использоваться при сборке приложения в режиме отладки (debug), доступные значения: – ram, загрузка и отладка приложения в ОЗУ; – eeprom, загрузка и отладка приложения в EEPROM; – spifi, загрузка и отладка приложения во внешней памяти.
board_build.ldscript	Задаёт сценарий компоновщика, который будет использоваться для сборки приложения в режиме выпуска (release), доступные значения: – ram, загрузка и выполнение приложения в ОЗУ; – eeprom, загрузка и выполнение приложения в EEPROM; – spifi, загрузка и выполнение приложения во внешней памяти.
monitor_port	Устанавливает номер последовательного порта для текстовой отладки (например: в ОС Windows — COM3, в ОС Linux — ttyUSB1)
monitor_speed	Устанавливает скорость передачи информации через последовательный порт для текстовой отладки.
test_port	Устанавливает номер последовательного порта для вывода сообщений модульных тестов (например: в ОС Windows — COM3, в ОС Linux — ttyUSB1)
test_speed	Устанавливает скорость передачи информации через последовательный порт при выполнении модульных тестов.
debug_speed	Устанавливает скорость работы программатора

### 3.3. Создание нового (пустого) проекта

Для создания нового пустого проекта выполните следующие действия:

- 1) Запустите приложение VS Code.
- 2) Откройте боковую панель PlatformIO.
- 3) Откройте домашнюю страницу PlatformIO (команда «Open»).
- 4) В разделе «Quick Access» нажмите кнопку «New Project».
- 5) В открывшемся диалоговом окне в поле «Name» укажите название для вашего проекта (например, FirstEmptyApp), в выпадающем списке «Board» выберите модель вашей отладочной платы, например «ELBEAR ACE-UNO» (используйте функцию поиск на вводе), поле «Framework» заполнится автоматически значением «Framework\_mik32\_folk».



По умолчанию, новый проект будет сохранен в папке проектов PlatformIO (см. 5 подраздела 3.1), если вы хотите изменить место хранения проекта, снимите галочку с поля «Location» и выберите необходимый каталог с помощью диалогового окна.

- 6) Для создания проекта нажмите кнопку «Finish».
- 7) В диалоговом окне «Вы доверяете авторам файлов в этой папке» нажмите кнопку «Да».

Ваш новый пустой проект готов. Созданный проект имеет структуру аналогичную описанной в подразделе 3.1, за исключением отсутствующего файла main.c. Данный файл не создается автоматически при генерировании пустого проекта. Отредактируйте содержимое файла конфигурации проекта platformio.ini в соответствии с задачами вашего проекта.