

1.1. Лабораторная работа: начало работы с микроконтроллером

Site: [Samsung Innovation Campus](#)

Course: Internet of Things

Book: 1.1. Лабораторная работа: начало работы с микроконтроллером

Printed by: Антон Файтельсон

Date: Saturday, 21 October 2023, 7:33 PM

Table of contents

1.1.1. Описание практикума

1.1.2. Hello World — компиляция и загрузка программы

1.1.3. Подключение терминала

1.1.4. Разбираем код демо-примера

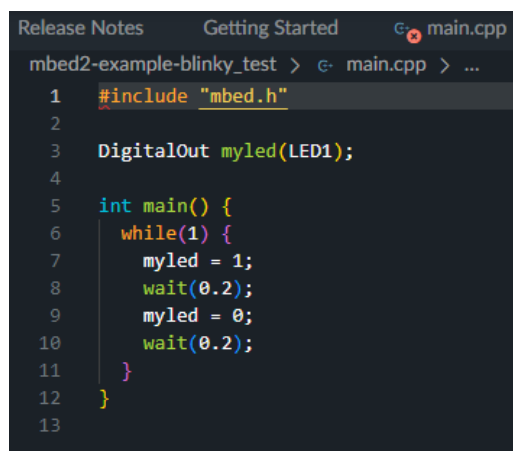
1.1.1. Описание практикума

Мы будем работать с готовыми модулями микроконтроллера STM32 - это наиболее популярное и широко используемое за счет низкой стоимости семейство микроконтроллеров, используемых и в реальной жизни, а не только в обучении. Программирование будет происходить на языке C.

Возможно, вы уже знакомы с проектом Arduino. Там можно быстро и легко выполнить базовые задачи. Но есть и существенные недостатки: за счет абстрагирования и упрощения порога вхождения, выпадают некоторые важные вещи. Например, вас приучают писать весь алгоритм программы в бесконечном цикле и щедро снабжать код задержками, тогда как в профессиональной embedded-разработке это делается через прерывания и многопоточность. Кроме того, историческая привязка проекта Arduino к микроконтроллерам Atmel делает решения на ней слабо масштабируемыми в силу высокой стоимости контроллеров. Тогда как решение-прототип на STM32 вы потом можете без труда перенести в реальное устройство, и оно будет бюджетным.

Мы будем изучать в нашем курсе несколько более современные (хотя и более сложные в освоении) программные платформы. Предлагается ARM Mbed - официальная платформа производителя (ARM), по синтаксису приближенная к Arduino, но с гораздо более широкими возможностями и более правильным стилем программирования.

На фото ниже - пример того, как выглядит самая простая программа на Mbed:



```
1 #include "mbed.h"
2
3 DigitalOut myled(LED1);
4
5 int main() {
6     while(1) {
7         myled = 1;
8         wait(0.2);
9         myled = 0;
10        wait(0.2);
11    }
12 }
13
```

После того, как мы научимся делать самые простые вещи - мигать светодиодом, считывать состояние кнопки, общаться с платой через консоль - вы попробуете написать свою собственную несложную программу - первый учебный Кейс.

Мы предлагаем проходить эту и последующие лабораторные работы, используя плату STM32Nucleo. Вот краткая информация о ней.

STM32Nucleo

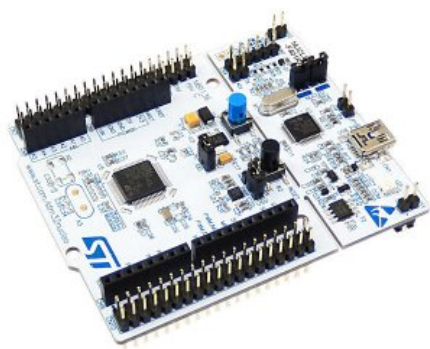
Это стандартная учебная плата от ST на настоящий момент. Рекомендуется использовать именно её. В магазинах есть большой выбор таких плат с разными характеристиками микроконтроллера, некоторые из них неудобны из-за малого количества памяти. Мы предлагаем модель NUCLEO-L152RE - в ней памяти точно хватит.

Плюсы:

- Форм-фактор Arduino
- Есть встроенный USB-UART (интерфейс для коммуникации платы и компьютера через консольный ввод/вывод)
- Легко доступна в продаже
- Очень удобно перепрошивать: определяется как флэшка, поэтому на нее достаточно просто "бросить" файл, и это работает и в Linux, и в Windows
- Совместима с Mbed "из коробки"

Минусы:

- Много различных версий с разными характеристиками (легко запутаться)
- Устаревший разъём MiniUSB (хотя, у него есть и преимущество: такой разъём сложнее сломать)



1.1.2. Hello World – компиляция и загрузка программы

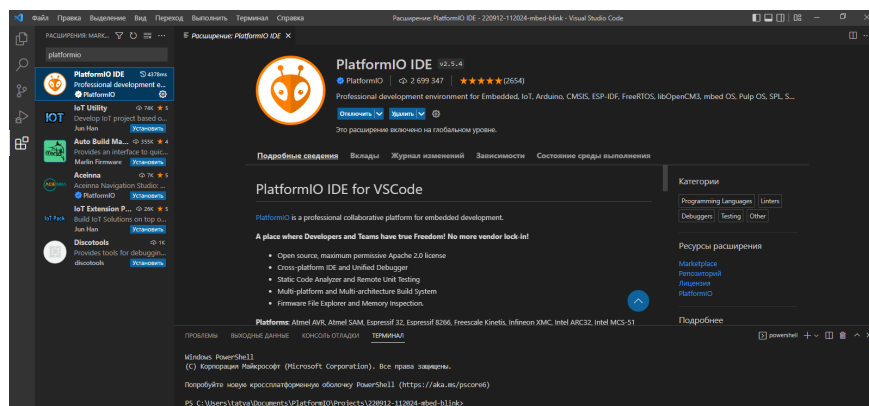
Работа в среде Platformio

Среда для программирования встраиваемых устройств Platformio - это надстройка над VSCode, открытой и бесплатной средой разработки от Microsoft. За последние годы Platformio заметно развивается и становится более открытым. К примеру, в 2019 году они сделали отладчик бесплатным, что очень важно для любого разработчика.

Platformio подходит работы с совершенно разными аппаратными платформами, такими как Mbed, ESP и так далее. В ней можно создать и скомпилировать новый проект всего за несколько кликов. В нашем курсе мы будем использовать эту среду как основную IDE.

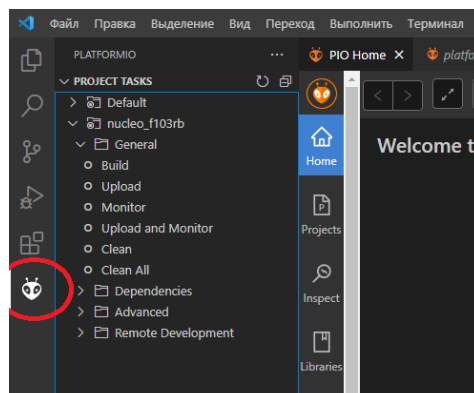
Всё, что нужно, это:

1. Установить VSCode: <https://code.visualstudio.com/>
2. Установить к нему расширение Platformio - это делается через менеджер расширений внутри VSCode
3. Перезапустить VSCode.



Расширения - самая нижняя иконка в левом меню. При поиске по расширениям по слову Platformio первый вариант как раз тот, что нам нужен.

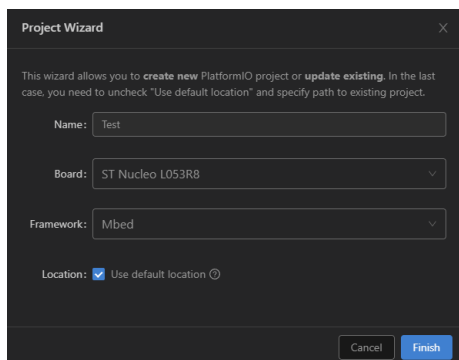
После установки расширения слева внизу появится значок муравья.



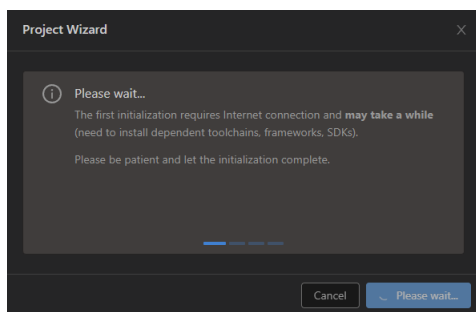
Теперь можно в главном меню выбрать создание нового проекта.



Введите имя проекта, плату (в моем случае это ST Nucleo L053R8) и фреймворк (Mbed).



При создании проекта начнут подтягиваться все библиотеки, это может занять продолжительное время.



После чего вы увидите пустой проект. Вставьте в его main.cpp следующий код:

```
#include "mbed.h"

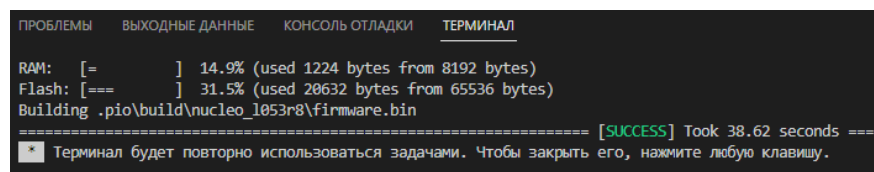
DigitalOut led(LED1);
int main()
{
    int i = 0;
    printf("Hello World !\n");
    while(1)    {
        wait_ms(1000); // 1 second
        led = !led; // Toggle LED
        printf("This program runs since %d seconds.\n", i++);
    }
}
```

Это самый простой демонстрационный код для мигания светодиодом (Blink).

Попробуем его загрузить в плату. Для этого подключите плату по USB и нажмите в нижнем меню кнопку сборки:



Первая сборка будет тоже длиться долго из-за компиляции всех библиотек. Но в результате увидите, что программа успешно собралась:



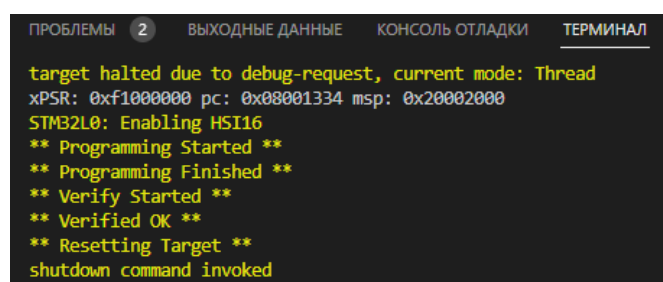
```
ПРОБЛЕМЫ  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ

RAM:  [=      ] 14.9% (used 1224 bytes from 8192 bytes)
Flash: [===    ] 31.5% (used 20632 bytes from 65536 bytes)
Building .pio\build\nucleo_l053r8\firmware.bin
===== [SUCCESS] Took 38.62 seconds =====
* Терминал будет повторно использоваться задачами. Чтобы закрыть его, нажмите любую клавишу.
```

И теперь ее можно загрузить в плату:



При первом запуске загрузки, дополнительно скачаются различные утилиты для прошивки. Наконец, программа будет загружена в плату:



```
ПРОБЛЕМЫ  2  ВЫХОДНЫЕ ДАННЫЕ  КОНСОЛЬ ОТЛАДКИ  ТЕРМИНАЛ

target halted due to debug-request, current mode: Thread
xPSR: 0xf1000000 pc: 0x08001334 msp: 0x20002000
STM32L0: Enabling HSI16
** Programming Started **
** Programming Finished **
** Verify Started **
** Verified OK **
** Resetting Target **
shutdown command invoked
```

И вы увидите мигающий светодиод!

Всё вышеописанное прекрасно показано в видео-инструкции, которую подготовил преподаватель ИТ Академии из НГТУ (Новосибирск) Илья Дубков:

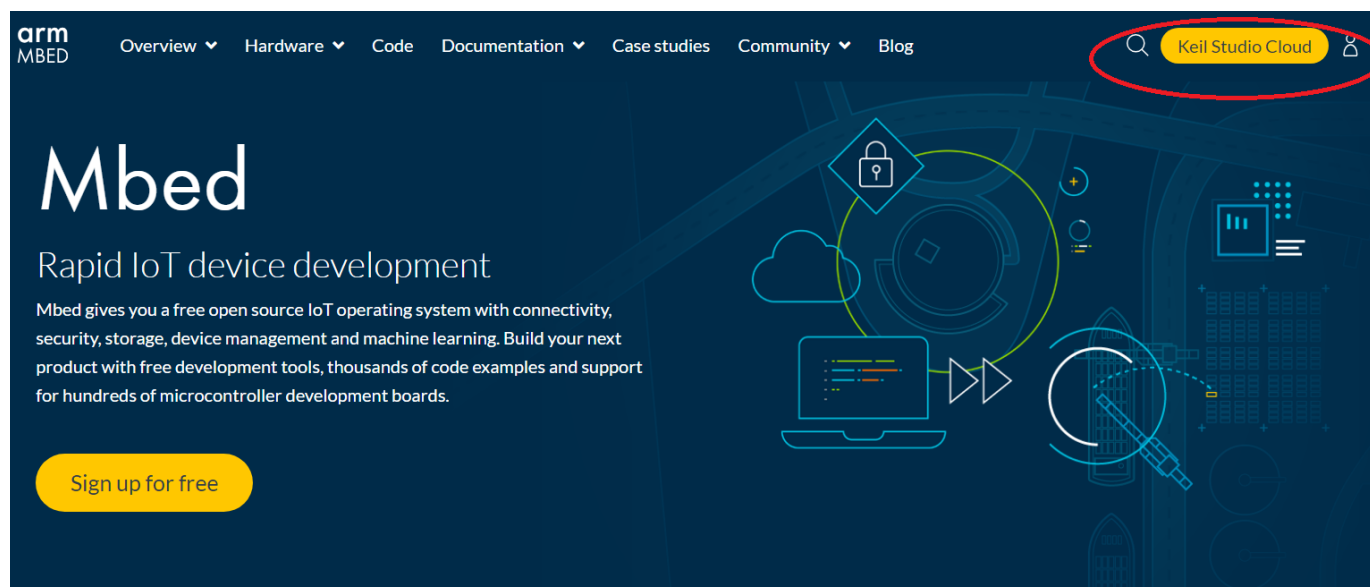
(Источник: Видео на YouTube “ПРОСОФТ: Быстрый старт с STM32 Nucleo и MBED”)

Работа в онлайн-IDE: Keil Studio

Внимание! Сайт Mbed доступен только через VPN. Если у вас есть доступ к VPN, вы можете работать по нижеприведенной инструкции. Если нет, то воспользуйтесь инструкцией для Platformio.

Для работы с Mbed в самом начале вам не понадобится даже устанавливать IDE - вы можете воспользоваться онлайн-средой

разработки Keil Studio Cloud. Для этого просто выберите соответствующий пункт на [главной странице](#) Mbed.



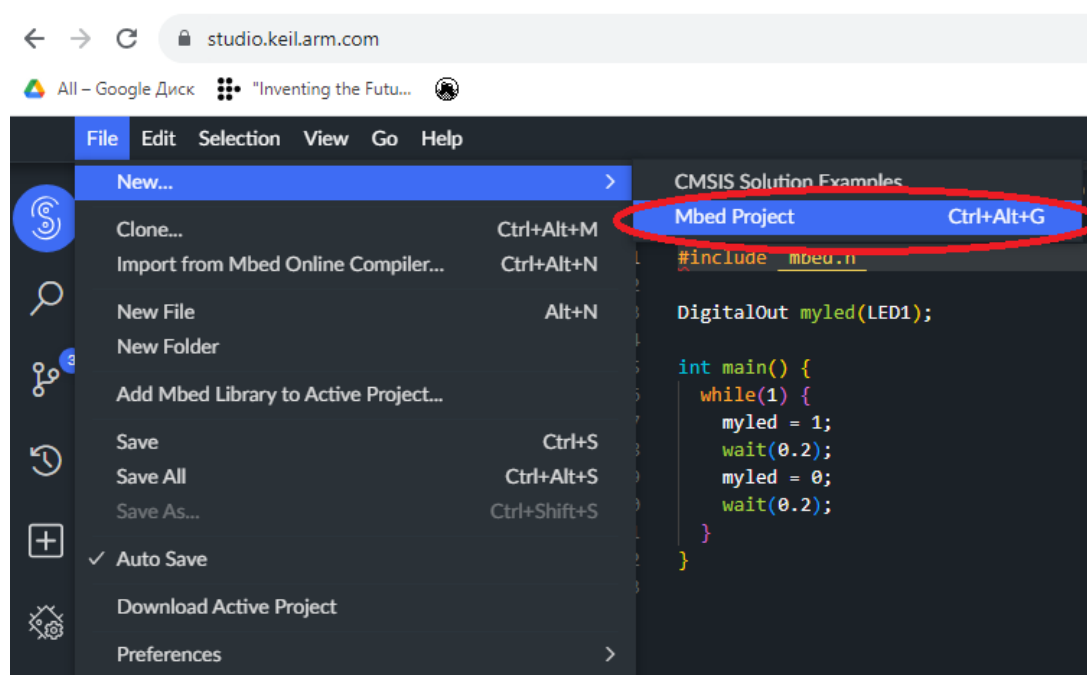
Для работы с онлайн-IDE вам понадобится создать и подтвердить аккаунт на сайте Mbed. Это тоже удобно: ваши программы будут сохраняться в облаке. Создать аккаунт достаточно просто, но вот на всякий случай видеоинструкция:

Конечно, онлайн-IDE - непривычная концепция со своими минусами (например, вы бы наверняка хотели бы программировать и в отсутствие доступа к Интернету), но для начала это ровно то, что нужно. Потом вы можете перейти на оффлайновую IDE под названием Mbed Studio, об этом чуть далее: на первых этапах она вам не понадобится. Собирать программу из консоли тоже можно - и это будет тоже затронуто в учебных материалах.

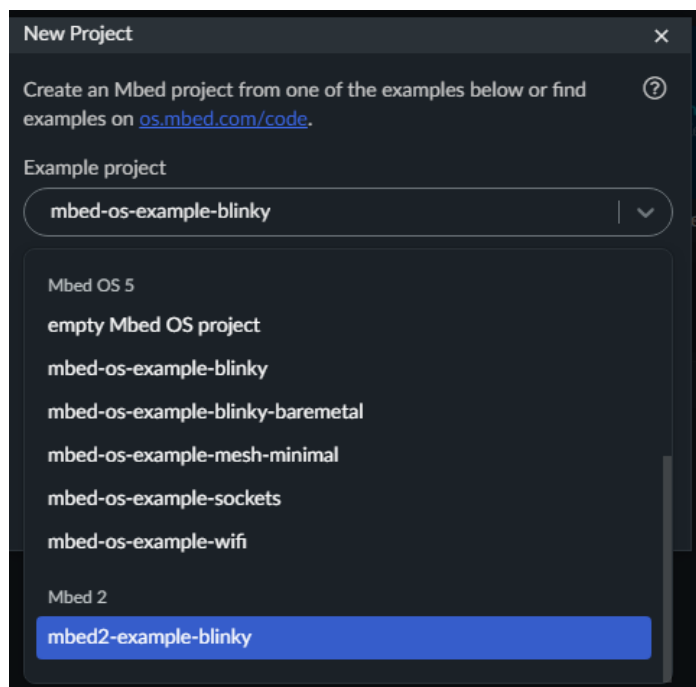
Начало работы с Keil Studio показано здесь:

То же самое, но в текстовом виде:

Создайте новый проект, нажав на File - New Mbed Project



Из типовых проектов выберите mbed2-example-blinky. Что означает цифра 2 в названии: это проект для Mbed версии 2, и его почти гарантированно поддерживают все платы. Поэтому попробовать оптимально этот вариант.

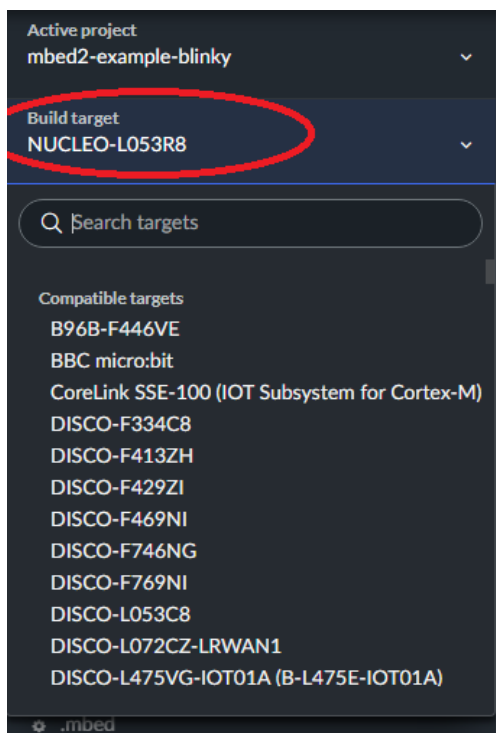


Текст программы очень прост. Она мигает светодиодом. В целом такая программа вам более чем знакома, если вы имели дело с проектом Arduino. Пока не будем разбираться, как она устроена, а просто используем ее в готовом виде:

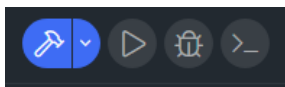
```
Release Notes  Getting Started  main.cpp
mbed2-example-blinky_test > main.cpp > ...
1  #include "mbed.h"
2
3  DigitalOut myled(LED1);
4
5  int main() {
6      while(1) {
7          myled = 1;
8          wait(0.2);
9          myled = 0;
10         wait(0.2);
11     }
12 }
13
```

Остается скомпилировать программу и скачать готовую прошивку. Но прежде нужно выбрать правильный тип платы. Нужно выбрать именно ту, которая у вас. Для примера, здесь используется STM32 Nucleo L053R8, но у вас может быть другая - посмотрите маркировку на плате и найдите вашу плату в списке.

Вверху слева, в общих настройках проекта:



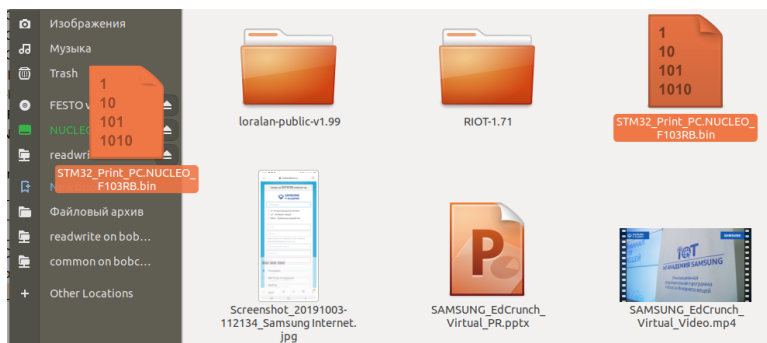
После чего можно скомпилировать и скачать программу. Значок молотка станет активным:



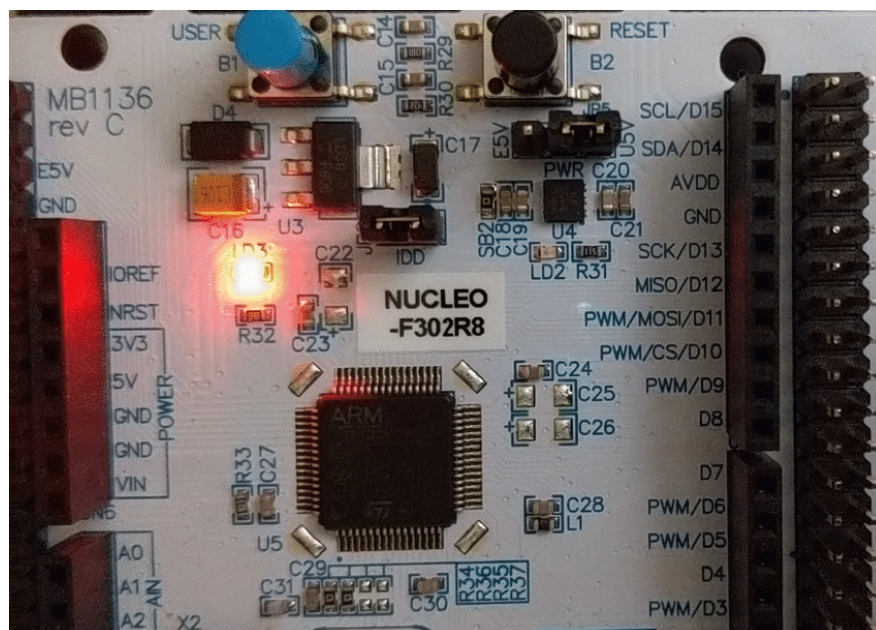
После онлайн-компиляции программа окажется у вас в папке "Загрузки".

Подключите устройство к компьютеру, используя MiniUSB-кабель. В Windows нужно дополнительно установить драйвер (система сама предложит сделать это). В Linux ничего дополнительно делать не нужно, есть поддержка на уровне ядра.

Устройство должно появиться у вас в системе как флэшка. Получившуюся прошивку вы просто "бросаете" на появившуюся в системе флэшку. Плата будет мигать красно-зеленым светодиодом, а потом перестанет – это означает, что прошивка загрузилась.



Результат: плата будет мигать встроенным светодиодом раз в секунду! Теперь вы можете поменять период мигания на свое усмотрение.



Следующий шаг: попробуйте вот эту программу:

```
#include "mbed.h"

DigitalOut led(LED1);
int main()
{
    int i = 0;
    printf("Hello World !\n");
    while(1)    {
        wait_ms(1000); // 1 second
        led = !led; // Toggle LED
        printf("This program runs since %d seconds.\n", i++);
    }
}
```

Результат: плата будет не только мигать, но и печатать, сколько секунд прошло.

1.1.3. Подключение терминала

Помимо мигания светодиодом, в коде прописан еще и вывод текста в терминал. Проверим, как это работает.

В среде Platformio

Выберите иконку терминала:



Внизу увидите окошко с печатью информации из программы:

```
--- Terminal on COM10 | 9600 8-N-1
--- Available filters and text transformations: colorize, debug, def
--- More details at https://bit.ly/pio-monitor-filters
--- Quit: Ctrl+C | Menu: Ctrl+T | Help: Ctrl+T followed by Ctrl+H
This program runs since 7 seconds.
This program runs since 8 seconds.
This program runs since 9 seconds.
This program runs since 10 seconds.
```

Со сторонней терминальной программой

Если вы не пользуетесь средой Platformio или другой IDE, то всегда можно взять любую терминальную программу на ваш выбор

Как узнать номер порта, где находится плата? В Linux нужно посмотреть в папке /dev/, там появится новое устройство, скорее всего с именем /dev/ttyACM0 или /dev/ttyUSB0. В Windows вы смотрите номер порта в Диспетчере устройств, в данном случае это оказался порт номер 8:

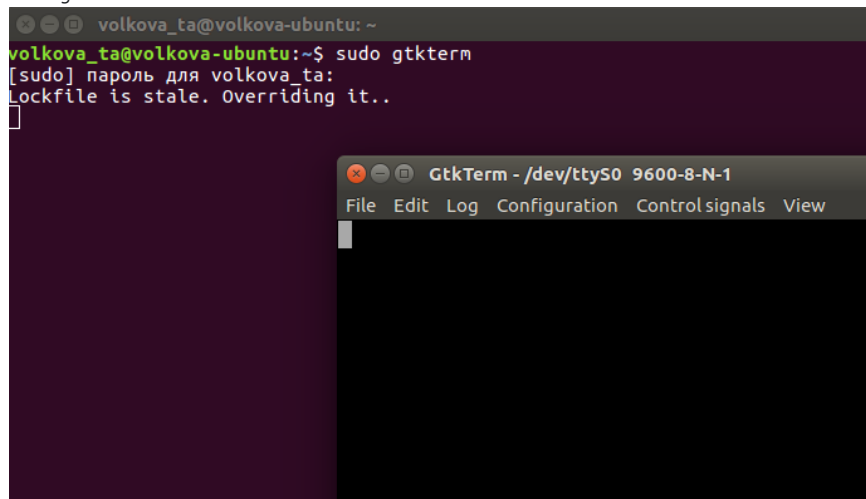
- > Переносные устройства
- ▼ Порты (COM и LPT)
 - STMicroelectronics STLink Virtual COM Port (COM8)
- > Программные устройства

Для коммуникации с устройством нужна терминальная программа. Для Windows годятся Putty, Ttermite и другие. Для Linux практически единственный удобный клиент с графическим интерфейсом - это GTKTerm, остальные - консольные (screen, minicom, picocom); можно также установить тот же самый Putty, если вы к нему привыкли.

Далее покажем всё на примере GTKTerm, но в принципе годится любая терминальная программа. Чтобы установить и запустить GTKTerm:

```
sudo apt-get install gtkterm
```

```
sudo gtkterm
```

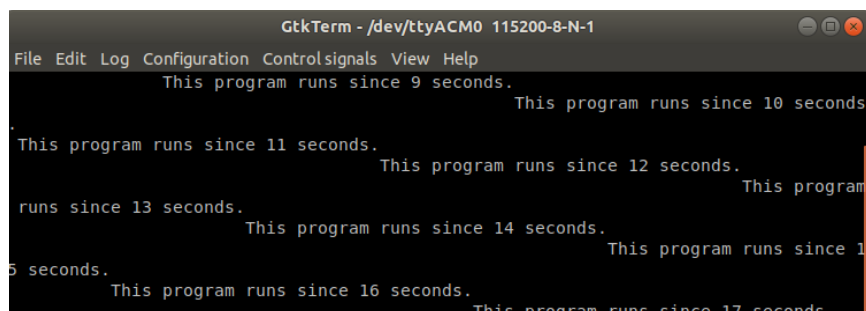


Укажите параметры соединения:

- COM-порт: /dev/ttyACM0 или аналогичное название, выбор из выпадающего списка.

- Скорость (бод): 9600

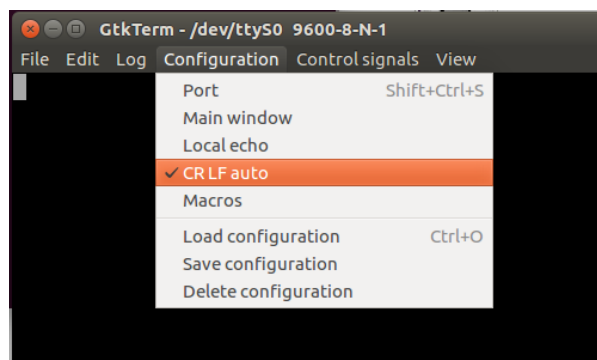
Вы увидите черное окно терминала, возможно - с задержкой.



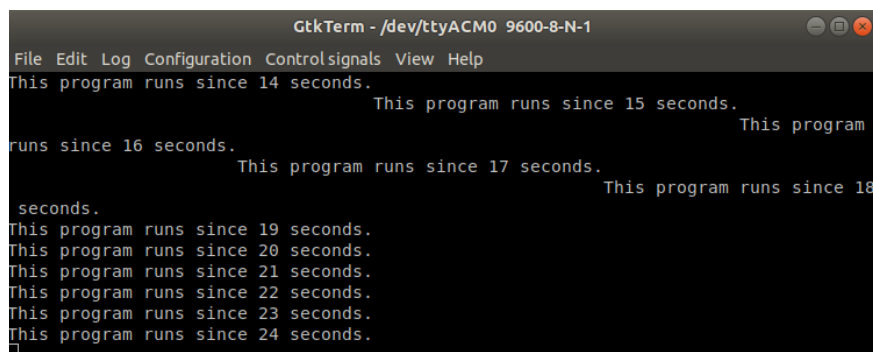
Программа будет отсчитывать время от старта.

Настройки терминальной программы

Для удобства работы можно поменять настройки эмулятора терминала. Если вы занимались программированием, к примеру, на C или Java, то знаете, что “перенос строки” (LF, line feed) и “возврат каретки” (CR, carriage return) – это два разных действия, и для удобства чтения текста в терминале следует в конце строки добавлять CR LF (“\r” и “\n”). Во встраиваемых системах зачастую, видимо по причине экономии одного байта, в конце строки ставится только LF, и строки будут выводиться «ёлочкой». Поэтому для удобства чтения нужно настроить терминал так, чтобы в любом случае в конце строки были символы CR LF. Это делается так:



После чего вы, перезагрузив плату, увидите вывод уже в удобочитаемом виде:



Или, альтернативный вариант - не забывать ставить в коде \n и \r в конце каждой строки, когда программируете!

Для удобства дальнейшей работы, сохраните параметры сессии: Configuration - Save Configuration.

Ещё одно полезное действие: чтобы каждый раз не вводить sudo при запуске терминала, можно сделать так:

```
sudo adduser MYUSERNAME dialout
sudo chmod a+rw /dev/ttyUSB0
```

(естественно, на место MYUSERNAME нужно подставить свое имя пользователя в системе. Пояснение: в Linux порты находятся в группе dialout, и пользователя нужно тоже добавить в эту группу. Также нужно разрешить запись и чтение порта.)

1.1.4. Разбираем код демо-примера

Теперь, когда всё заработало, разберем исходный код подробно

```
#include "mbed.h"

DigitalOut led(LED1);
int main()
{
    int i = 0;
    printf("Hello World !\n");
    while(1)
    {
        wait_ms(1000); // 1 second
        led = !led; // Toggle LED
        printf("This program runs since %d seconds.\n", i++);
    }
}
```

С первой строкой: `#include "mbed.h"` всё понятно. Мы подключаем библиотеку, как вы обычно подключали `stdio` или `iostream` в других языках

Далее строчка `DigitalOut led(LED1);` - в этой строке создается переменная `led`, в качестве параметра ей в конструктор передается `LED1` - это не что иное, как макрос, отсылающий к выводу микроконтроллера, к которому на плате подключен встроенный в плату светодиод. Вместо `LED1` можно подставить название любого другого вывода - например, `PA_8`, и сигнал уже будет отправляться на этот вывод. Класс `DigitalOut`, объектом которого является `led` - это класс для работы с выводами, настроенными как выходы.

Дальше начинается `main()` и в самом начале заводится переменная `int i = 0` - она будет использоваться для подсчета, сколько секунд прошло с начала программы. Для проверки первый раз печатается `Hello World`. Затем начинается бесконечный цикл.

`wait_ms(1000);` означает ждать заданное число миллисекунд, `led = !led;` - переключение переменной в противоположное значение на каждой итерации, и затем печать - сколько раз произошел инкремент счетчика. Программа никогда не завершает свою работу и крутится в бесконечном цикле. Такой стиль написания программы, когда весь код пишется в бесконечном цикле, очень характерен для платформы `Arduino`, но строго говоря, он неправильный. Мы здесь сейчас рассматриваем этот пример только потому, что он очень просто устроен. А как делать правильно - будет рассказано дальше.

В качестве упражнения на самостоятельное выполнение, попробуйте изменить период мигалки.

[Reset user tour on this page](#)