

Лабораторная работа №2

Использование среды Mbed

Теоретическая часть

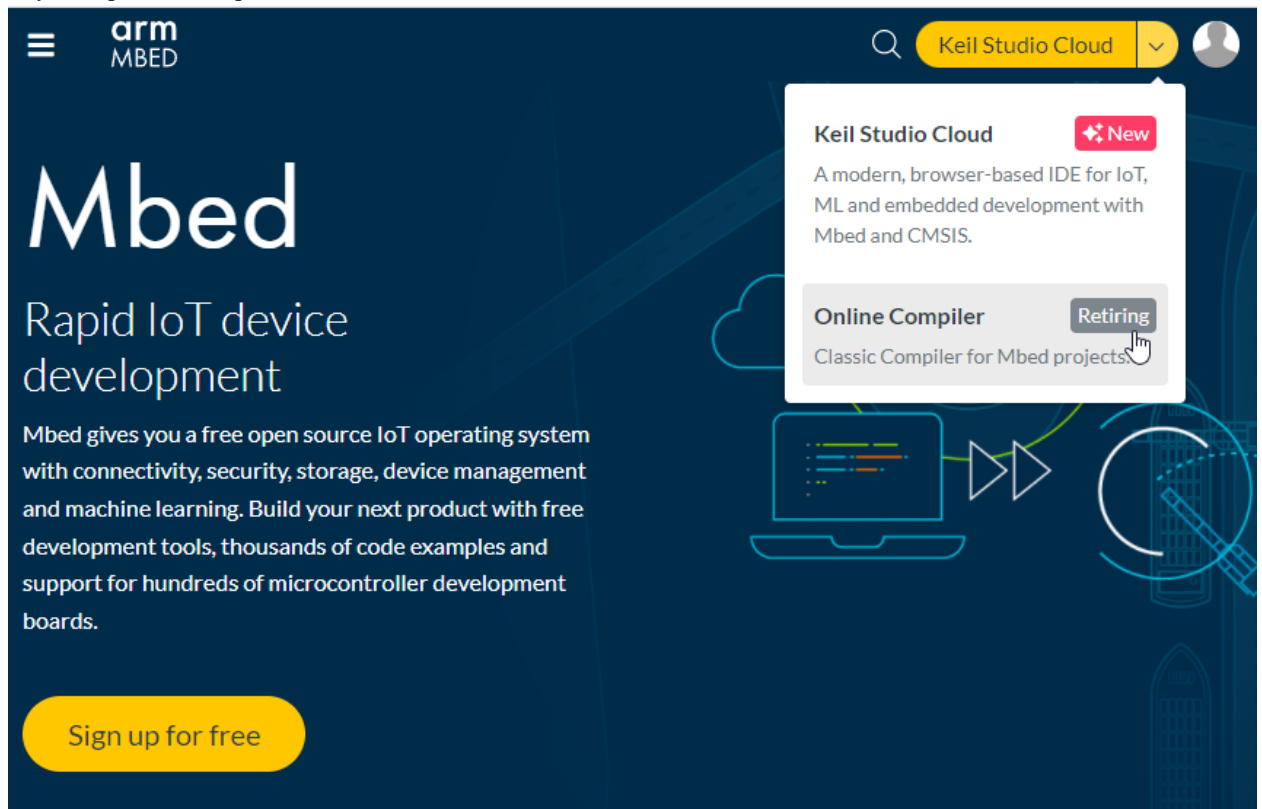
Mbed — программная платформа компании ARM, ориентированная на процессорные ядра этой компании. Платформа охватывает IDE, компиляторы, библиотеки, аппаратную привязку к популярным модулям и платам на основе контроллеров с ядром ARM.

Отличительной чертой mbed является вариант работы, при котором программист не нуждается в установке специального инструментария для программирования контроллеров. В этом варианте для написания кода используется онлайн IDE, работающая через браузер. Компиляция также производится онлайн. Облачное хранение кода поддерживается онлайн системой контроля версий Mercurial. Программист скачивает на свой компьютер уже двоичный исполняемый файл, который может тут же загрузить на одну из поддерживаемых плат. На некоторых из предлагаемых плат установлен USB-загрузчик, имитирующий флеш-накопитель: запись файла на который приводит к прошивке программной памяти микроконтролера. Таким образом, программист не нуждается ни в каких специальных программных или аппаратных инструментах, а работа может быть выполнена даже на планшете.

Платформа состоит из двух частей: ядра, над которым работает команда профессиональных программистов, и компонентов, в создании которых может принять участие любой желающий.

Регистрация в системе Mbed

Для начала работы необходимо пройти регистрацию на сайте <https://os.mbed.com>. Переходим по ссылке, нажимаем на кнопку “Sign up for free”, далее выбираем “Sign up” и вводим все требуемые данные, соглашаемся с условиями и нажимаем кнопку Signup. После этого, на открывшейся странице нажимаем на кнопку Compiler для перехода в Online IDE.

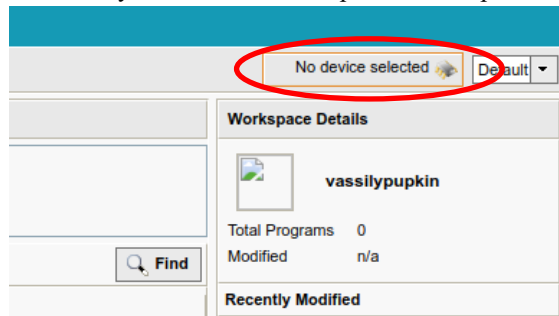


Выбор отладочной платы

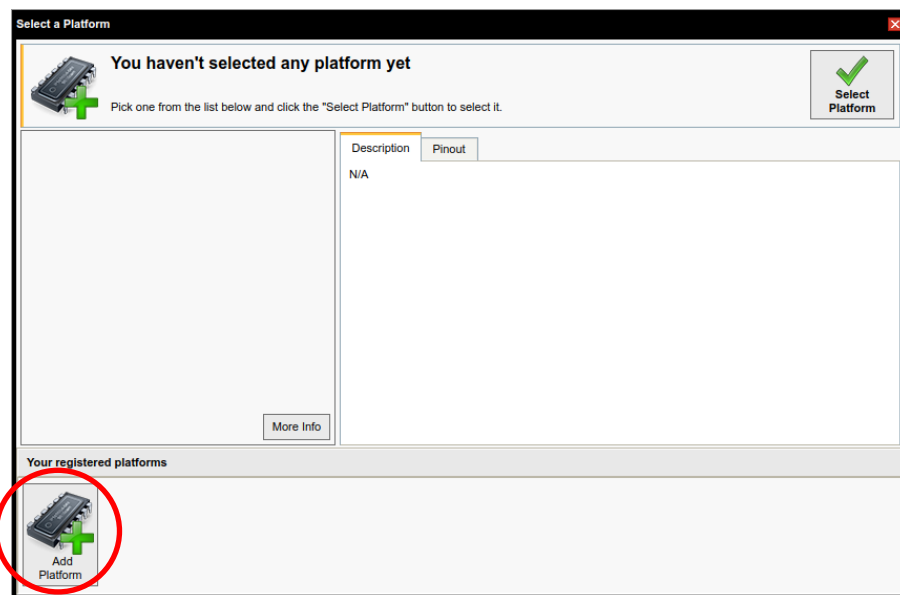
При первом запуске Online IDE необходимо выбрать тип отладочной платы для которой система будет генерировать исполняемый код.

В наших лабораторных работах мы будем использовать отладочную плату Nucleo-L053R8 на базе микроконтроллера STM32 L053R8Tx. Эта плата выполнена в форм-факторе Arduino и позволяет использовать множество совместимых плат расширений, сенсоров и исполнительных устройств. Одновременно с этим она позволяет использовать весь потенциал микроконтроллеров STM32.

Для выбора платы необходимо щелкнуть по кнопке выбора платы в правом верхнем углу окна:



В открывшемся окне выбора платы нажать кнопку «Add Board»:



Из открывшегося списка выбрать NUCLEO-L053R8:

Filter

mbed Enabled

☒ mbed Enabled

mbed OS support

☐ mbed OS 2

☐ mbed OS 5

Target vendor


☐ ARM
☐ Atmel
☐ Maxim Integrated
☐ NXP Semiconductors
☐ Nordic Semiconductor ASA
☐ Nuvoton
☐ Renesas
☒ STMicroelectronics
☐ Silicon Labs
☐ WIZnet
☐ u-blox AG

Platform vendor

☐ ARM
☐ Atmel
☐ BBC Make it Digital Campaign
☐ CQ Publishing Co.,Ltd.
☐ Delta
☐ Embedded Artists
☐ Esptel
☐ JkSoft
☐ Maxim Integrated
☐ MultiTech
☐ NXP Semiconductors
☐ Nordic Semiconductor ASA
☐ Nuvoton
☐ Outrageous Circuits
☐ RedBearLab
☐ Renesas


Boards

Showing 31 of 114 ([Show all](#))




NUCLEO-F103RB

- Cortex-M3, 72MHz
- 128-KB Flash, 20-KB SRAM
- CAN USB




NUCLEO-L152RE

- Cortex-M3, 32MHz
- 512-KB Flash, 80-KB SRAM
- LCD DAC OPAMP USB




NUCLEO-F030R8

- Cortex-M0, 48MHz
- 64-KB Flash, 8-KB SRAM




NUCLEO-F401RE

- Cortex-M4 + FPU, 84MHz
- 512-KB Flash, 96-KB SRAM
- USB_OTG_FS SDIO




NUCLEO-F302R8

- Cortex-M4 + FPU, 72MHz
- 64-KB Flash, 16-KB SRAM
- DAC OPAMP CAN USB




NUCLEO-L053R8

- Cortex-M0+, 32MHz
- 64-KB Flash, 8-KB SRAM
- LCD DAC USB



NUCLEO-F411RE

- Cortex-M4 + FPU, 100MHz
- 512-KB Flash, 128-KB SRAM
- USB_OTG_FS SDIO



NUCLEO-F334R8

- Cortex-M4 + FPU, 72MHz
- 64-KB Flash, 16-KB SRAM
- DAC OPAMP CAN

В окне с описанием отладочной платы необходимо щелкнуть по кнопке «Add to your mbed compiler» внизу правой панели:

[Boards](#) » [NUCLEO-L053R8](#)

NUCLEO-L053R8

Affordable and flexible platform to ease prototyping using a STM32L053R8T6 microcontroller.



Overview


The STM32 Nucleo board provides an affordable and flexible way for users to try out new ideas and build prototypes with any STM32 microcontroller line, choosing from the various combinations of performance, power consumption and features.

The Arduino™ connectivity support and ST Morpho headers make it easy to expand the functionality of the STM32 Nucleo open development platform with a wide choice of specialized shields.

The STM32 Nucleo board does not require any separate probe as it integrates the ST-LINK/V2-1 debugger/programmer.

Table of Contents

1. Overview
2. Microcontroller features
3. Nucleo features
4. Board pinout
5. Supported shields
6. Getting started
7. Technical references
8. Known limitations
9. Tips and Tricks

 To compile a program for this board using Mbed CLI, use `nucleo_l053r8` as the target name.

Board Partner



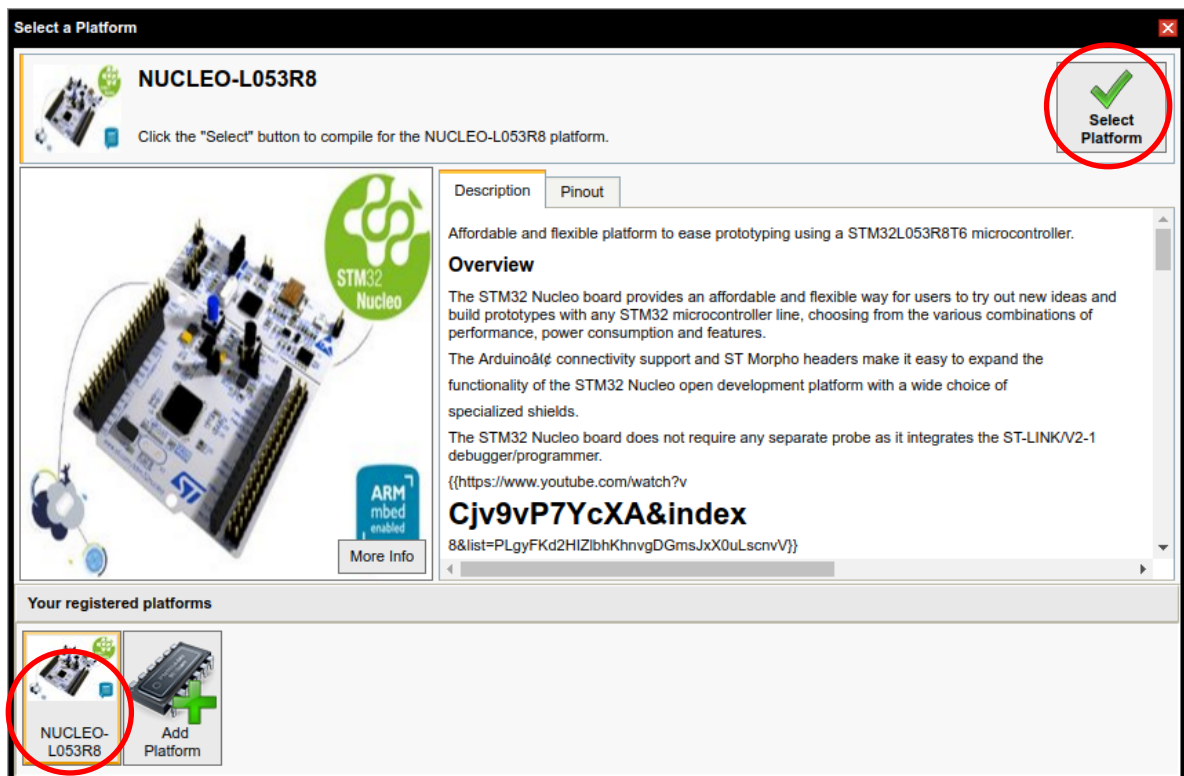
ST

A world leader in providing the semiconductor solutions that make a positive contribution to people's lives, both today and in the future.

 Add to your Mbed Compiler

Если все прошло успешно, то вы должны увидеть сообщение «Platform 'NUCLEO-L053R8' is now added to your account!»

После этого можно вернуться в онлайн компилятор, повторно нажать на кнопку выбора платы и в открывшемся окне выбрать добавленную:



Выполнив эти подготовительные шаги, можно приступить к основной части задания.

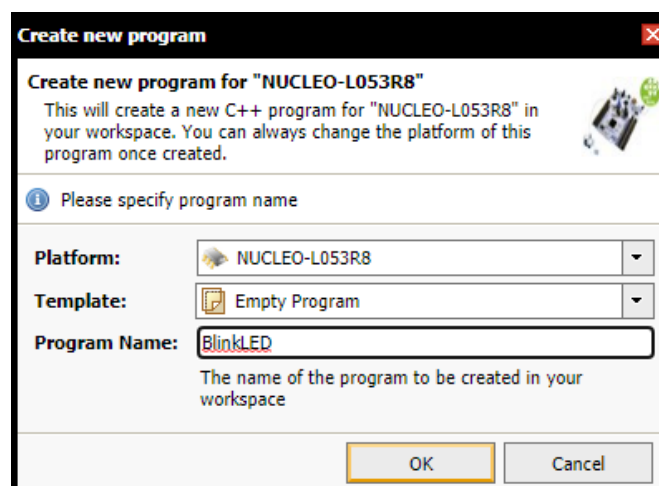
Разработка с использованием mbed

Mbed сочетает в себе слой высокоуровневых абстракций для аппаратуры, обширную библиотеку решений типовых задач и достаточное количество примеров, которые можно легко импортировать, тестировать и модифицировать для своих потребностей. Вся разработка в mbed ведется на языке C++03 (C++11 и C++14 доступны при компиляции в MbedStudio).

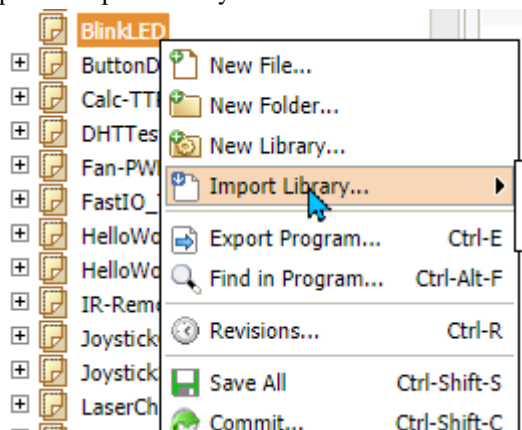
В общем случае для создания проекта на базе mbed нужно выполнить следующие шаги:

1. Создать пустой проект для выбранной платформы
2. Импортировать библиотеку mbed
3. Модифицировать и доработать имеющийся код
4. Скомпилировать программу и сохранить бинарный код на отладочную плату

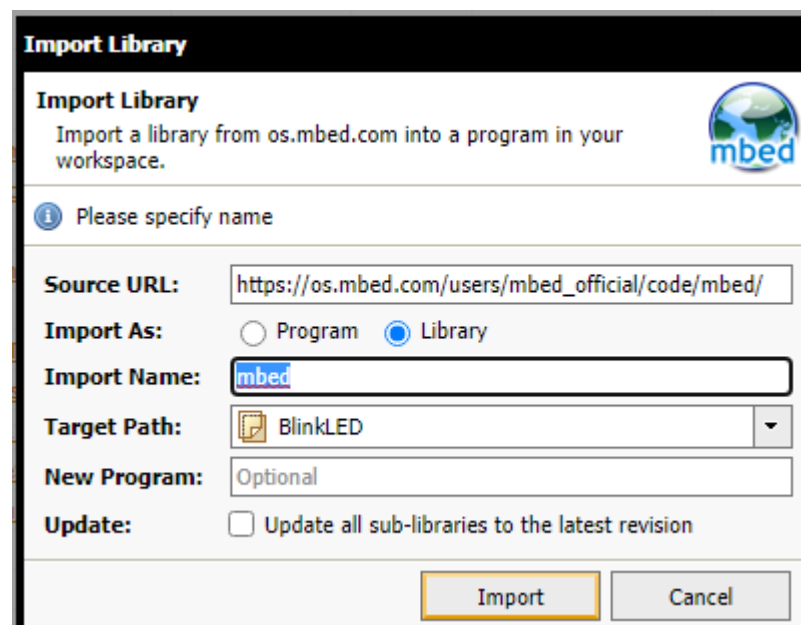
Запустим простейший пример для мигания светодиодом на плате. Для этого нажимаем кнопку «New» в левом верхнем углу экрана. Выбираем шаблон «Empty Program», называем проект BlinkLED и нажимаем OK:



Затем необходимо подключить библиотеку Mbed к проекту. Для этого щелкаем правой кнопкой по проекту в левой панели и выбираем “Import Library”

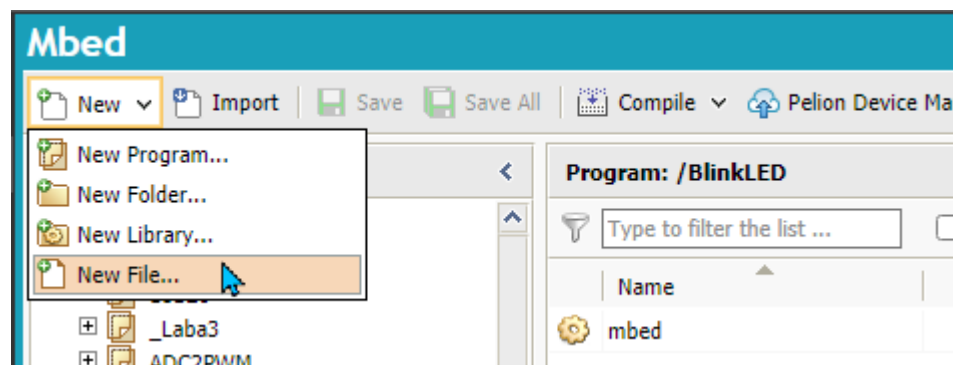


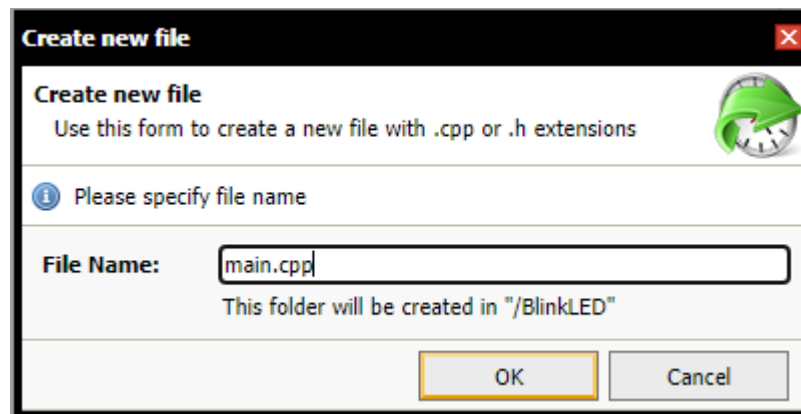
В открывшемся мастере подключения библиотек в поле поиска вводим “mbed”, выбираем первый пункт от автора “mbed official” и нажимаем кнопку “Import!”.



В диалоге импорта соглашаемся с параметрами по умолчанию и нажимаем “Import”.

Затем нужно создать новый файл в котором будем писать программу. Назовем его main.cpp.





Теперь в файл нужно добавить код, который будет мигать светодиодом:

```
#include "mbed.h"

DigitalOut myled(LED1);

int main() {
    while(1) {
        myled = 1; // LED is ON
        wait(0.2); // 200 ms
        myled = 0; // LED is OFF
        wait(1.0); // 1 sec
    }
}
```

Здесь можно выделить 3 основные части.

1. Подключение заголовочного файла `mbed.h` делает доступными все объявления, типы и классы системы `mbed`
2. Объявление переменной `myled` типа `DigitalOut` создает объект позволяющий управлять цифровым выходом микроконтроллера (проще говоря, позволяет подавать положительное напряжение на одну из клемм)
3. Функция `main` в которой выполняется основной цикл программы. Происходит циклическое включение выхода (`myled = 1`), временная задержка, а затем выключение выхода (`myled = 0`)

Теперь можно нажать кнопку «Compile» и сохранить получившийся файл на отладочную плату. Если все пройдет успешно, то зеленый светодиод на плате начнет мигать с частотой 1.2 Гц.

Работа с портами ввода-вывода (GPIO)

Самые простые действия с микроконтроллерами производятся посредством управления портами ввода\вывода. Порты ввода/вывода (General Purpose Input Output, GPIO) – предназначены для общения микроконтроллера с внешними устройствами. С их помощью мы передаем информацию другим устройствам и принимаем информацию от них. В зависимости от типа, микроконтроллер может иметь на своем борту от одного до семи GPIO. Каждому порту ввода/вывода присвоено буквенное обозначение – A, B, C, D, E, F, G. Все порты в микроконтроллере равнозначные восьмиразрядные (содержат восемь линий, они же выводы, они же разряды, они же биты) и двунаправленные – могут как передавать, так и принимать информацию. GPIO в микроконтроллере обслуживают все его устройства, в том числе и периферийные. Поэтому, в зависимости от того какое устройство будет работать с портом он может принимать и передавать или цифровую информацию, или аналоговую.

Порты классифицируются по типу сигнала:

- цифровые порты – которые работают с цифровыми сигналами – логическими “нулями” и логическими “единицами”
- аналоговые порты – которые работают с аналоговыми сигналами – использующими плавно весь

диапазон входных напряжений от нуля вольт до напряжения питания МК

- смешанные порты – они и используются в наших МК, могут оперативно переключаться с режима “цифровой порт” в режим “аналоговый порт”, и обратно.

В технической литературе и схемам GPIO обозначаются следующим образом:

- “P” – первая буква, означающая слово “порт”
- “A” (B, C, D, E, F, G) – вторая буква, обозначающая конкретный порт
- “0” (1, 2, 3, 4, 5, 6, 7) – третий символ – цифра, обозначающая конкретный вывод (регистр, бит) порта.

К примеру: “порт A” – PA, “пятый разряд порта A” – PA5.

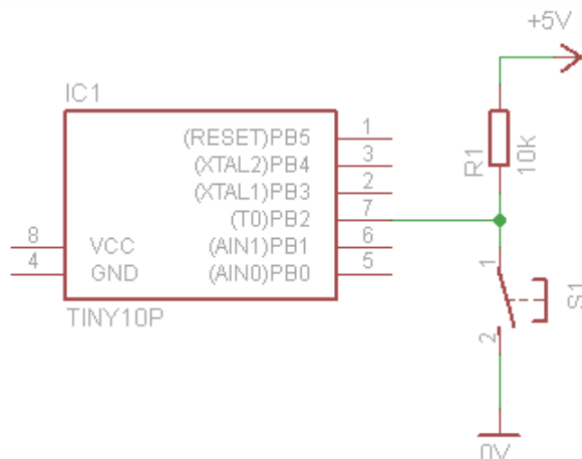
Если в МК есть несколько портов, то не обязательно их имена могут идти по порядку – A, B, C. Может быть и так – B, C, D. Поэтому пугаться искать где же порт A не надо.

Кроме того, хотя порты восьмиразрядные, выводов у порта не обязательно должно быть 8, может быть и меньше, к примеру 3 – PA0, PA1, PA2. В таком случае порт называют неполным, или урезанным.

В этой лабораторной работе мы будем рассматривать только работу в режиме цифрового порта. Для этого в mbed есть два базовых класса – DigitalIn (цифровой вход) и DigitalOut (цифровой выход). Используя класс DigitalIn – мы можем узнать какой уровень напряжения подается на выбранный нами вход и таким образом определить нажата ли кнопка, замкнут ли переключатель и тд. Используя класс DigitalOut мы можем управлять состоянием выхода, и устанавливать на нем высокий или низкий уровень напряжения.

Кнопки

В целом, кнопка представляет собой простой переключатель который замыкает электрическую цепь при нажатии. Как правило схема подключения кнопки выглядит следующим образом:



Здесь кнопка S1 подключена с одной стороны к 7 ножке микроконтроллера (PB2), а с другой к земле. Резистор R1, подключенный к источнику напряжения, – называется подтягивающим резистором и выполняет важную роль. Когда кнопка замкнута, напряжение на входе микроконтроллера падает до 0В, так как цепь замыкается на землю. В те моменты, когда кнопка не нажата и цепь PB2->S1->0V разомкнута, ток течет через резистор R1 и «подтягивает» напряжение на входе микроконтроллера до уровня +5В. Если бы этого резистора не было и источник питания был бы подключен напрямую ко входу микроконтроллера, то при нажатии на кнопку, происходило бы короткое замыкание, так как источник напряжения соединялся бы напрямую с землей. А если бы мы вообще отключили источник питания от входа микроконтроллера, то при размыкании кнопки вход оставался бы в «подвешенном» состоянии не подключенном ни к +, ни к –, и микроконтроллер не мог бы достоверно узнать замкнута кнопка или разомкнута.

На плате NUCLEO-L053R8 расположено две кнопки, они подписаны как USER и RESET. Как следует из названия, кнопка USER – является пользовательской, то есть доступной для использования в программе. А кнопка RESET – используется для перезагрузки прибора, аналогично кнопке RESET на ПК.

Для работы с кнопками в mbed используется класс DigitalIn – цифровой вход. Объявляется кнопка следующим образом:

```
DigitalIn myButton(USER_BUTTON);
```

Здесь myButton – это название объекта (экземпляра класса) по которому мы в дальнейшем сможем обращаться в программе. USER_BUTTON – это идентификатор ножки микроконтроллера к которой подключена кнопка USER. Этот идентификатор объявлен в библиотеках среды mbed, и для каждой

отладочной платы соответствует кнопке, которую может использовать пользователь. Вместо идентификатора USER_BUTTON можно использовать явное название ножки, в нашем случае это – PC_13, то есть Порт C, ножка 13, результат будет аналогичный.

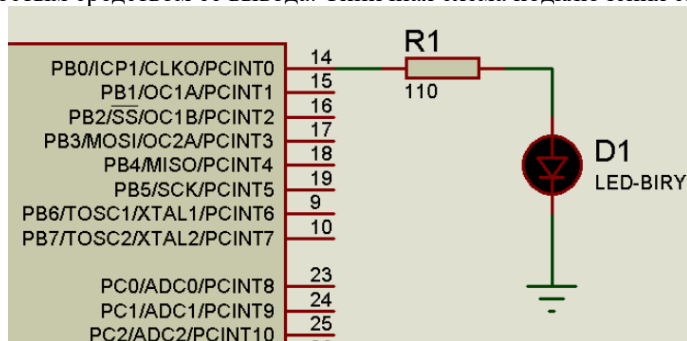
После того как мы создали объект кнопки и привязали его к нужной ножке, мы можем анализировать значение на выбранном входе. Согласно схеме показанной выше, при нажатой кнопке на входе мк будет – 0В, или состояние логического нуля, при отжатой кнопке, на входе будет +5В, или состояние логической единицы. Соответственно, код для обработки нажатия кнопки может выглядеть следующим образом:

```
if (myButton == 0) {
    //то что нужно сделать
    //если кнопка нажата
}
else {
    //то что нужно сделать
    //если кнопка отжата
}
```

Следует учитывать, что в данном случае анализируются текущие состояния входов, и если подобный код помещен в бесконечный цикл опроса состояния кнопки, то действия связанные с текущим состоянием кнопки будут выполняться на каждой итерации, что, как правило, является нежелательным. Чаще всего вам требуется сравнивать предыдущее состояние кнопки с текущим и выполнять какие-либо действия только при смене состояния.

Светодиоды

Как кнопка является самым простым средством ввода информации, так и светодиод является самым простым средством ее вывода. Типичная схема подключения светодиода выглядит следующим образом:



К выход №14 (PB0) микроконтроллера подключен светодиод D1. Резистор R1 в цепи называется токоограничивающим резистором, и служит для ограничения протекающего в цепи тока. Границы допустимого и максимального тока варьируются от типа светодиода, как правило, это ток порядка 10-30 мА. Номинал резистора, который необходимо добавить в цепь можно вычислить, используя закон Ома:

$$R = \frac{U}{I}$$

Напряжение логической единицы контроллера – 3,3В. Соответственно для получения тока номиналом 30мА необходим резистор номиналом $3,3/0,03 = 110$ Ом. Чем ниже сопротивление резистора, тем ярче будет гореть светодиод (пока не сгорит).

После того как светодиод подключен к микроконтроллеру, его можно включать и выключать при помощи программы. Для этого в mbed используется класс DigitalOut (цифровой выход), предназначенный для переключения состояния выхода из состояния логического нуля и обратно. В листинге ниже приведен пример кода, мигающий светодиодом с частотой 1 Гц:

```
#include "mbed.h"

DigitalOut led(LED1);

int main() {
    while(1) {
        led = 1;    //включаем светодиод
        wait(0.5);  //ожидаем 500мс
        led = 0;    //выключаем светодиод
        wait(0.5);  //ожидаем 500мс
    }
}
```



```
}  
}
```

С помощью классов DigitalIn и DigitalOut уже можно строить достаточно сложные системы управления, так вместо кнопки могут применяться различные датчики с выходом типа сухой контакт, а вместо светодиодов могут использоваться реле и другие управляющие приборы.

Варианты

1. Помигать светодиодом 3 раза с частотой 2 Гц при нажатии на кнопку
2. Помигать светодиодом 5 раз с частотой 1 Гц при нажатии на кнопку
3. Мигать светодиодом с частотой 3 Гц, пока не нажата кнопка, когда кнопка нажата – не мигать. Светодиод должен остаться в том состоянии в котором был на момент нажатия.
4. Мигать светодиодом с частотой 5 Гц, при нажатии на кнопку мигать с частотой 10 Гц, при повторном нажатии снова мигать с частотой 5 Гц и тд.
5. Мигать светодиодом с частотой 5 Гц если нажата кнопка
6. Мигать светодиодом с частотой 1 Гц, при каждом нажатии на кнопку увеличивать частоту на 1 Гц
7. Мигать светодиодом с частотой 2 Гц, при нажатии на кнопку включить светодиод на 5 секунд, затем снова перейти к миганию
8. Мигать светодиодом с частотой 2 Гц если кнопка отжата, если кнопка нажата то мигать с частотой 5 Гц
9. Мигать светодиодом с частотой 100 Гц и скважностью 90%. При нажатии на кнопку уменьшать скважность на 10%. Дойдя до нуля снова установить 90%.
10. Мигать светодиодом с частотой 75 Гц и скважностью 10%. При нажатии на кнопку увеличивать скважность на 10%. Дойдя до 100% снова установить 10%.