Методические рекомендации по работе с STM32

Подготовлено студентами Колледжа Современных Технологий им М. Ф. Панова

Год 2023 группой СОБ/21 года

При поддержке В.Г. Макарова

**Введение**

STM32 это семейство 32-битных микроконтроллеров STMicroelectronics на базе ядра ARM, различные модули, периферия и программное обеспечение.

Микроконтроллеры STM используется во многих областях профессиональной деятельности.

В отличии от Arduino, программирование STM32 требует глубокого погружения.

К основным преимуществам можно отнести:

1. Низкая стоимость
2. Удобство использования
3. Выборка среды разработки
4. Взаимозаменяемые чипы
5. Высокая производительность

Из недостатков:

1. Высокий порог вхождения

От студентов-первопроходцев в сфере программирования микроконтроллеров STM32, прилагаем методические рекомендации для создания различных проектов.

**Желаем удачи в освоении материалов**

**НАЧАЛО РАБОТЫ**

Театр начинается с вешалки, а создание проекта с выбора среды разработки

STM поддерживает несколько рабочих сред разработки

Мы будем использовать STM32CubeIDE.

ШАГ 1

Загрузите архив установочной программы с официального сайта (<https://www.st.com/en/development-tools/stm32cubeide.html>) или с стороннего сайта (https://st.alx52.ru/)

ВНИМАНИЕ

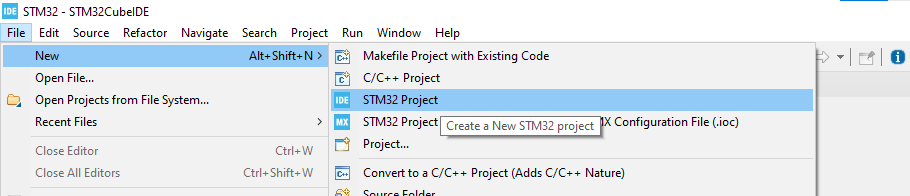
УСТАНОВКА ПРОГРАММЫ С ОФИЦИАЛЬНОГО САЙТА НЕДОСТУПНА ДЛЯ РОССИИ

Установка возможна только с стороннего сайта

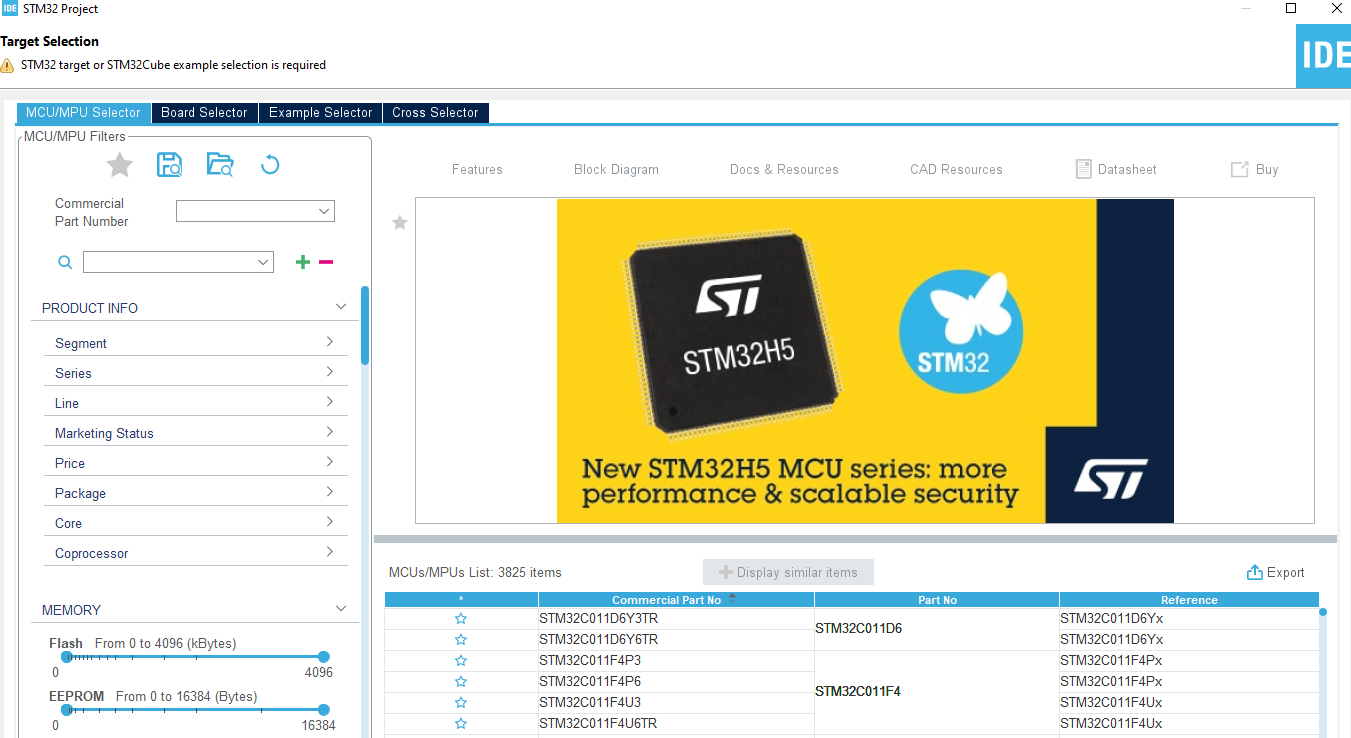
Авторы не приветствуют использование пиратского софт

ШАГ 2

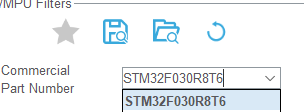
После установки программного обеспечения приступаем к созданию нового проекта



В окне необходимо выбрать модель и версию микроконтроллера



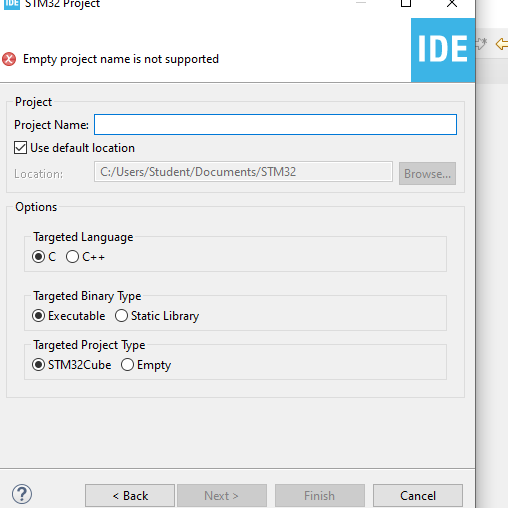
В наших руках была плата с микроконтроллером **STM32F030R8T6**



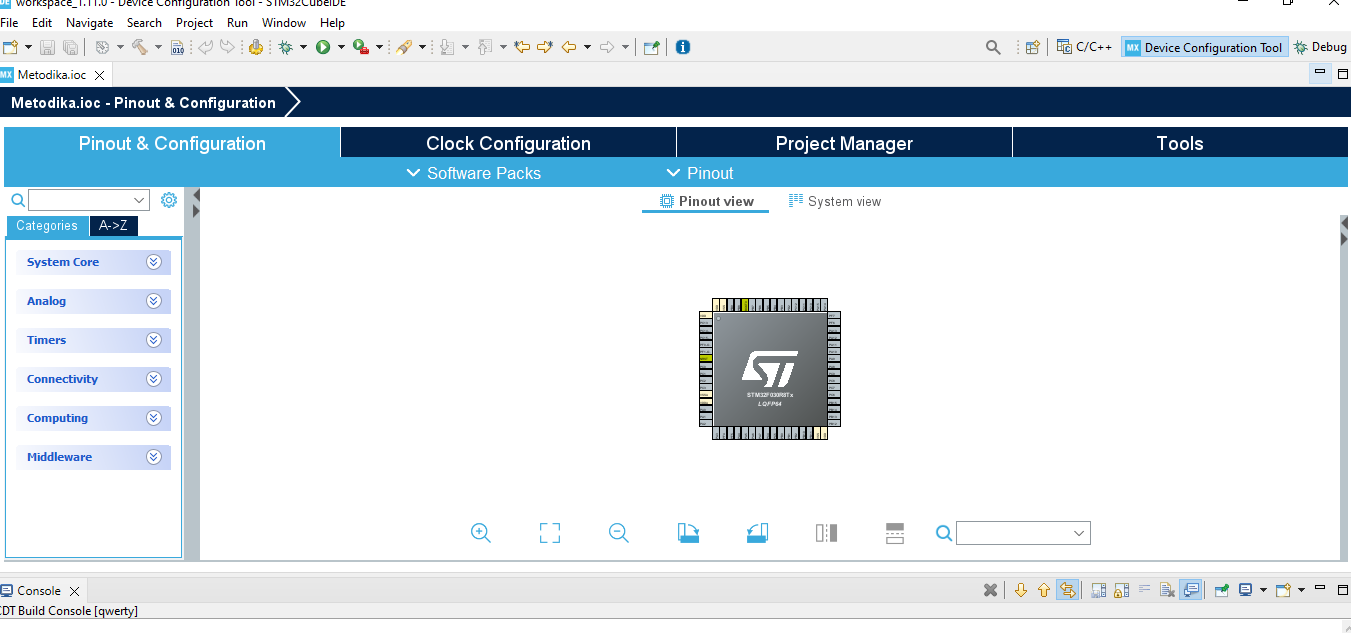
Называем проект

Так же можно выбрать место размещение проекта

Язык программирования и среду разработки



Нажимаем Finish



Перед нами окно инициализации

Здесь настраивается возможности микропроцессора

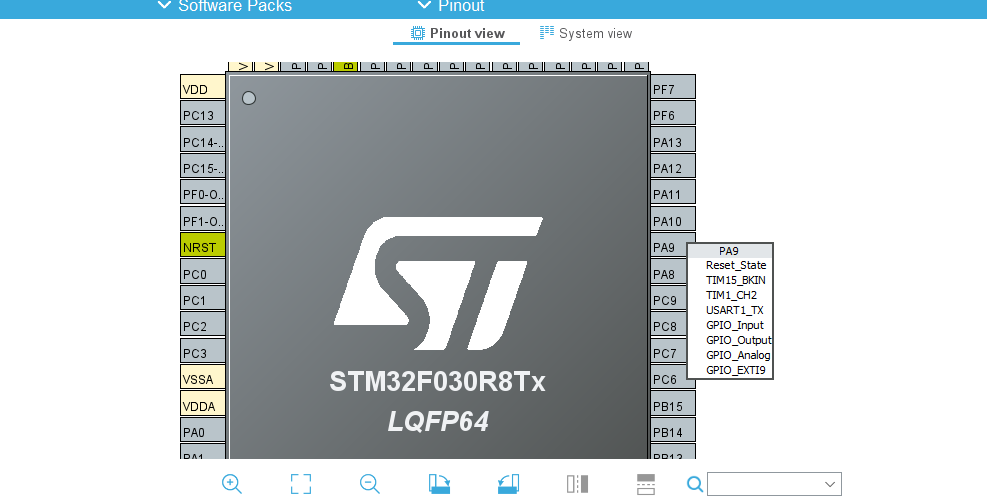
Инициализируются как “ножки процессора”, так и пины на которые, можно получить данные так и отправить

Для первого проекта нам будет достаточно использовать дискретный пин PA9

Наводим на мышку на нужный пин

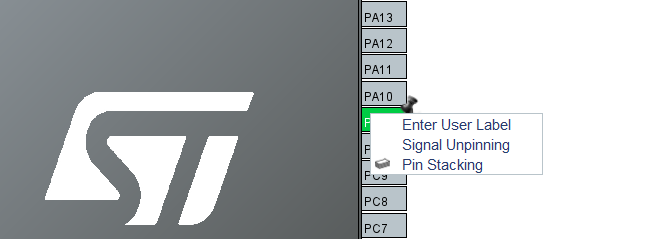
ЛКМ и выбираем нужный режим

В нашем случае инициализируем **GPIO\_Output**

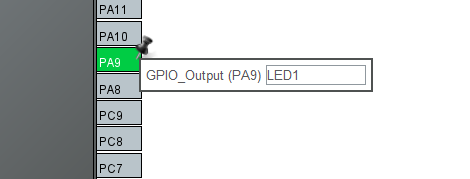


По нажатию ПКМ мы можем переименовать пин на удобное нам название

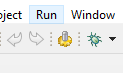
Выбираем пункт **Enter User Label**



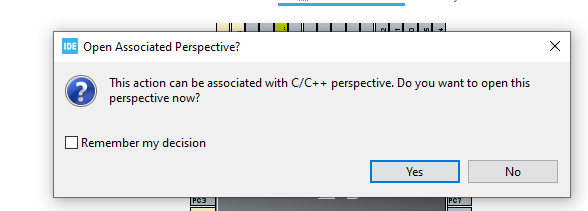
Для удобства назовем пин LED1



По нажатию “золотой шестерни” программа автоматически сгенерирует начальный код



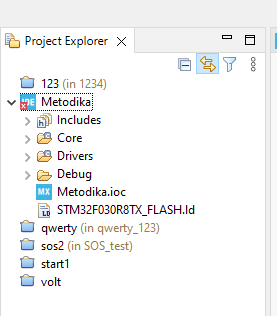
Подтверждаем компиляцию проекта



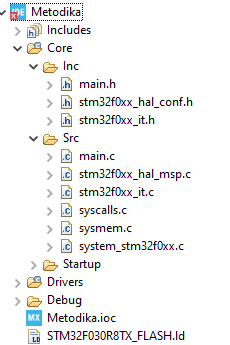
Справой стороны показан **Project Explorer**

Данное окно позволяет открывать файлы, относящиеся к проекту или проектам

Обратите внимание, что для удобства, файлы находятся в папках



Рассмотрим папки подробнее



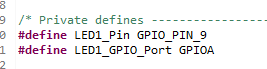
В папке Core, есть ещё две папки **Inc** и **Src**

В файле **main.h** содержится информация о инициализированных портах и подключенных библеотеках

Подключена библиотека **HAL**



Инициализирован пин и порт

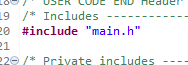


**LED1\_Pin** это инициализированный пин

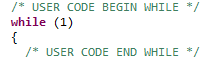
**LED1\_GPIO\_Port** это инициализированный порт

В файле main.c прописывается сам код

Подключение файла с библиотеками и портами

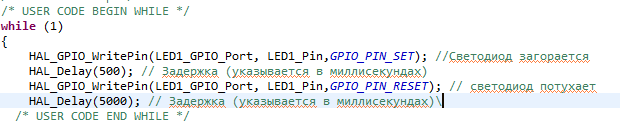


В данном сегменте будет выполнение кода, пока есть питание



ВНИМАНИЕ! ПИСАТЬ КОД НЕОБХОДИМО ТОЛЬКО В ВЫДЕЛИНЫХ СЕГМЕНТАХ, ИНАЧЕ ПРИ СЛЕДУЮЩЕЙ КОМПИЛЯЦИИ КОД ПРОПАДЕТ

Прописываем код



Чтобы проверить код на ошибки и сохранить его, прожимаем **CTRL+B** или нажимаем на “молоточек”

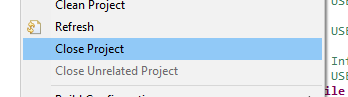


Если вы используете имя пользователя в компьютере на Кириллице, то не получится сразу загрузить в отладочную плату

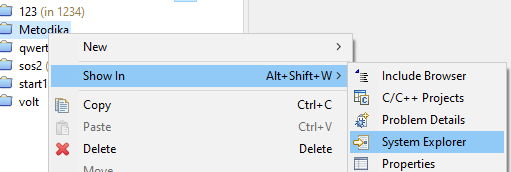
Это исправляется следующими действиями

1. Скопируйте проект в папку путь до которой не имеет Кириллицу
2. Удалите проект из Project explorer, предварительно сохранив его и закрыв

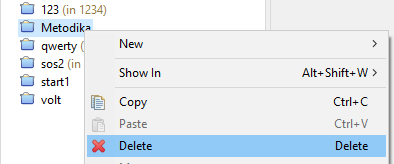
ПКМ и **Close Project**



1. Что бы найти проект достаточно нажать ПКМ - **Show In – System Explorer**

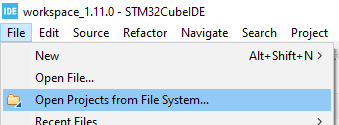


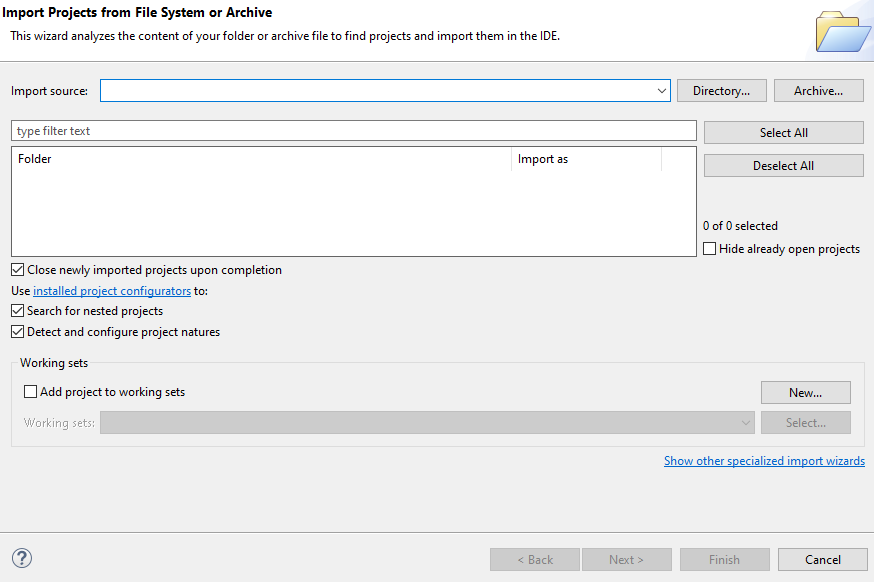
Не забудьте удалить проект из изначальной папки



1. Импортируем проект обратно

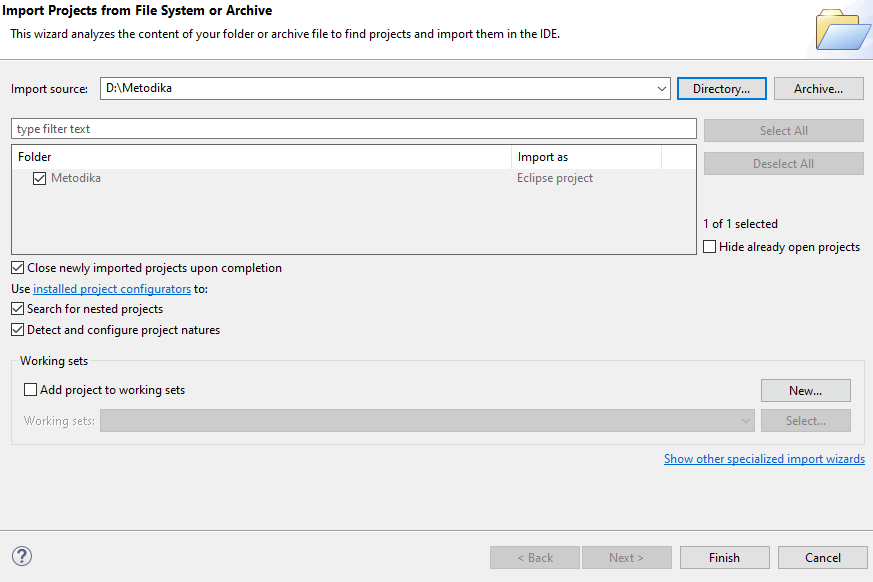
**File-Open Projects from File System…**



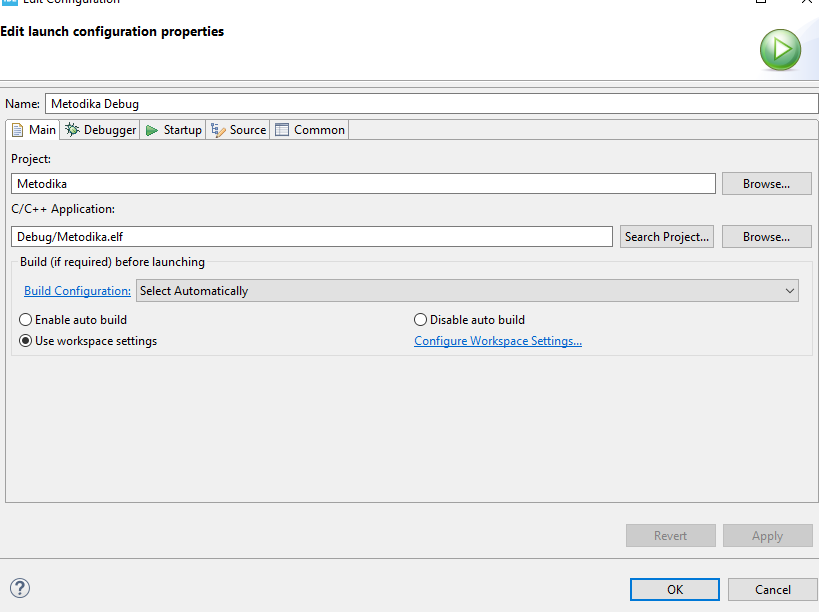


Среда разработки позволяет импортировать как папки (**Directory**), так и архивы (**Archive**)

1. Выбираем импортированные файлы и нажимаем **Finish**



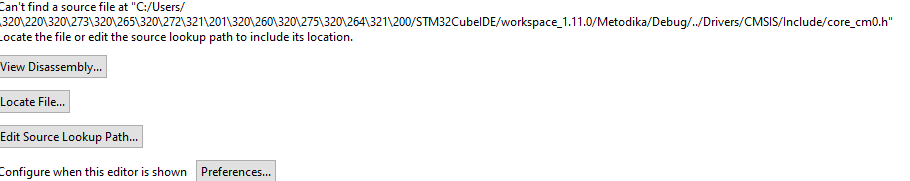
Подключаем плату отладки к компьютеру

Загружаем проект в плату нажатием на “зеленую стрелочку” 

Нажимаем **OK**

Если не найден какой-либо из файлов, то указываем путь до него вручную

Нажимаем **Locate File** и выбираем нужный файл

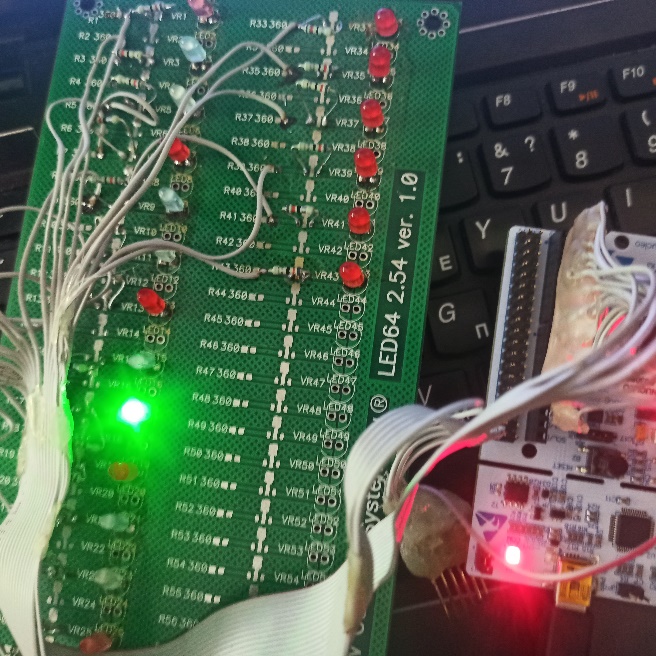


Загружаем проект повторно

Отключаем питание платы

Подключаем плату отладки к отладочной плате

В итоге должно получится следующее

Светодиод загорается на 0,5 секунды каждые 5 секунд

**АНАЛОГОВО-ЦИФРОВОЙ ПРЕОБРАЗОВАТЕЛЬ**

Усложним задачу

Сделаем так, чтобы светодиод не горел при отсутствии напряжения, если напряжение 3,3 V, то светодиод мигает, если напряжение 5 V, тогда светодиод горит

Для считывания внешнего напряжения потребуется АЦП (аналогово-цифровой преобразователь), к счастью он уже встроен STM-ку.

Следовательно, достаточно просто инициализировать его

(Заметки Авторов: Когда мы изучали АЦП и пытались его инициализировать, мы перебрали тонны материалов в попытках понять, как он работает. Мы работали над этим на протяжении целой недели. Начали мы конечно с понятия и принципа его работы, как его инициализировать, настроить и откалибровать. В итоге у нас наконец токи получилось и проект заработал.)

Вообще АЦП имеет огромное множество функций работы, но нам достаточно простого измерения напряжения. Если АЦП не один, а несколько он имеет 2 варианта работы **независимый** и **парный**

В независимом режиме каждый из АЦП получает значение с своего инициализированного пина.

В парном режиме АЦП могут получать значения с одного пина или один АЦП будет запускать другой.

Так же у АЦП можно настроить два варианта работы с каналами – регулярный **(regular channels)** и инжектированные**(Injected channels).**

Инжекторованный работает так: каждый инициализированный аналоговый пин имеет свою ячейку хранения в памяти микроконтроллера

У регулярных каналов всего одна ячейка хранения, поэтому значения со всех пинов будут записываться, затирая старая значения. Для корректной работы потребуется вовремя “забирать” значения из ячейки

АЦП имеет внутренний входы, такие как термопару в микроконтроллере и измерение напряжения на самой плате

Довольно теории, переходим к практике

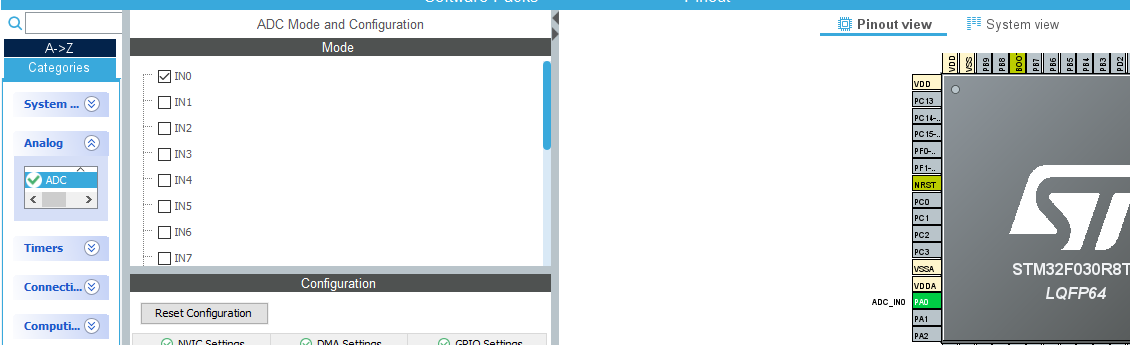
Шаг 1

Инициализация и первичная настройка АЦП

Для работы аналогово-цифрового преобразователя потребуется инициализировать аналоговый пин соответственно. И пин для светодиода.

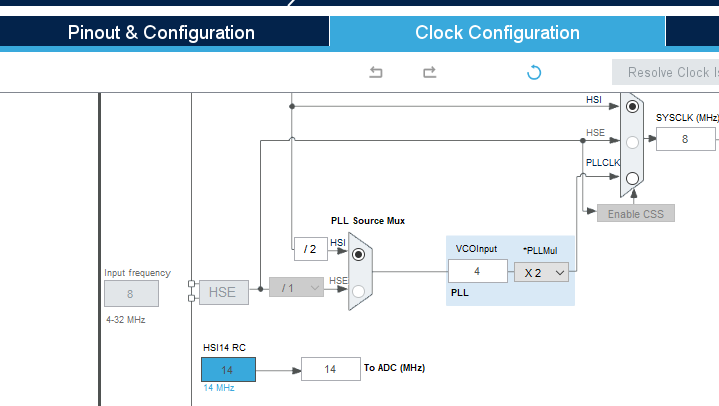
В Categories выбираем пункт Analog, подпункт ADC (обычно их несколько но в нашем МК (микроконтроллере) он только один :(

Выбираем инициализируемый пин (IN0 или любой из предложенных средой разработки)



ВНИМАНИЕ! Максимальная частота работы АЦП 14 MHz!

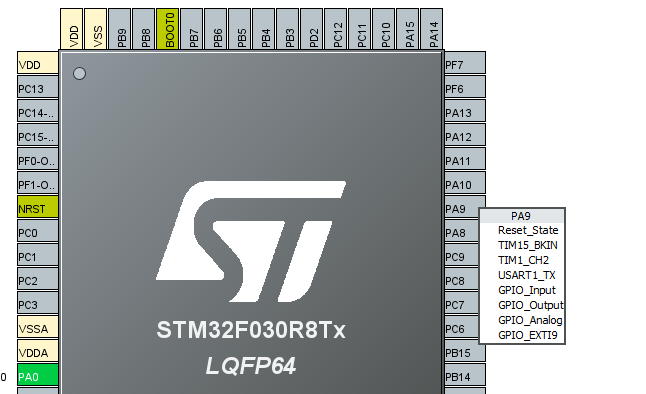
Что бы проверить правильность частоты, переходим в вкладку **Clock Configuration**



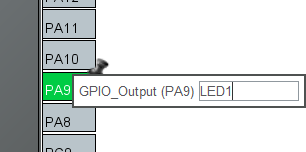
Ищем пункт **To ADC (Mhz)**

У нас частоту изменить невозможно (оно и к лучшему :) )

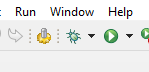
****

Незабываем про светодиод 

Называем его LED1



Нажимаем на “золотую шестерню” для генерации начального кода



Инструкция и базовые команды “Как работать с АЦП?” описаны в файле **stm32f0xx\_hal\_adc.c** (Путь до файла …/Drivers/ STM32F0xx\_HAL\_Driver/Src/stm32f0xx\_hal\_adc.c)

В файле описаны тип и вид АЦП, режимы работы, допустимое измерение, основные команды и так далее…

Рассмотрим характеристики нашего АЦП

Наш АЦП 12-битной конфигурации. Следовательно, максимальное цифровое значение 4095 (212 = 4096, но мы считаем и ноль поэтому максимальное значение 4095);

Есть самокалибровка

Возможность измерять напряжение от 0 до 3,3 V;

Внимание! Максимальное напряжение, подаваемое на АЦП не должно превышать 3,3 V. Для измерение больших величин потребуется делитель величин

Краткий алгоритм работы с АЦП

1. Открываем пин
2. Запускаем АЦП
3. Дожидаемся окончания преобразования
4. “Вытаскиваем” значение
5. Выключаем АЦП

Переходим к коду

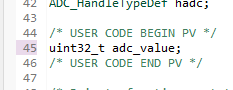
Объявим переменную до начала выполнения основного кода

Команда uint32\_t [название\_переменной];

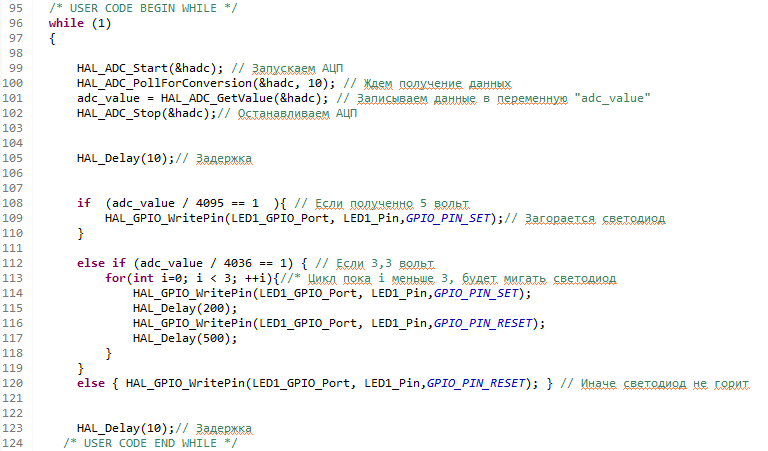
Прописывается в поле

USER CODE BEGIN PV

USER CODE END PV



Основной код пишется в поле **while(1)**



Проверяем проект на ошибки, нажатием “Молоточка”



После проверки на ошибки загружаем проект не через “зеленую стрелочку”

, а через “дебагер” (Debug) 

Он позволяет отладить проект “в реальных условиях”, а в нашем случае показать промежуточное значение переменной “adc\_value”.

После загрузки в режиме “дебага” плата будет постоянно мигать

Это нормально, значит плата готова к проверке проекта

Подключаем плата расширения, но не подключаем пин к напряжению

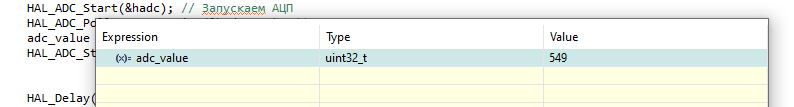
Нажатие кнопки  запустит поэтапное выполнение кода

Нажатие этой же  приостанавливает выполнение

Зная это - попробуем “поймать” промежуточное значение

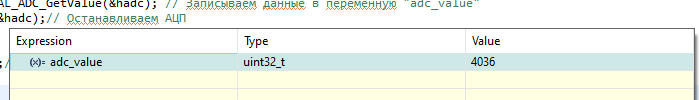
Алгоритм выполнения

1. Запускаем поэтапное выполнение кода
2. Ждем 3 секунды
3. Ставим на паузу
4. Проверяем значение

Наводим мышь на нашу переменную 

Значение под Value и есть нужное нам значение

Теперь подключим пин к напряжению 3,3 V с помощью провода-перемычки и снова запустим выполнение кода

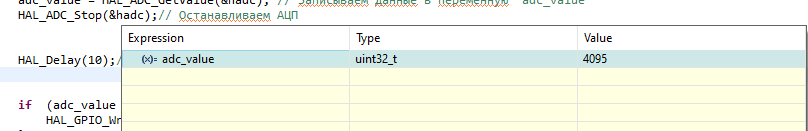
У нас это значение 4036 

Значение у вас может отличаться, рекомендуем откалибровать это в формуле.

Тепрь подключаем на 5V и повторяем алгаритм

Теперь светодиод должен гореть постояно

Останавливаем выполнение кода и проверяем значение переменной



В нашем случае значение 4095

Если вы все сделали правильно и отредактировали код под свои переменные данные

То у вас должно получиться следущее

5V – Светодиод горит постоянно

3,3V – Светодиод моргает

Нет питания – Светодиод не горит

**Кнопки и таймеры**

Задача: При нажатие кнопки загарается светодиод; При отпускании светодиод гаснет

В STM32 кнопки определяются как простой ключ (выключатель)

Следовательно, чтобы запрограммировать действие по кнопке достаточно просто заставить ее замыкать контакты

У нас в плате есть 2 встроенные кнопки

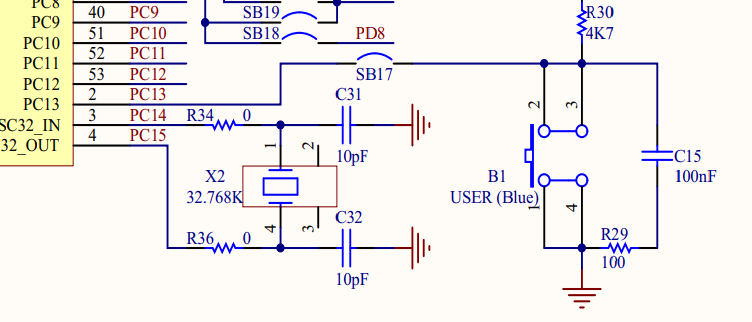
RESET (черная) и USER (синяя)

Кнопку RESET трогать не советуем (оставьте ее для быстрого перезапуска проекта)

А вот USER нам идеально подходит

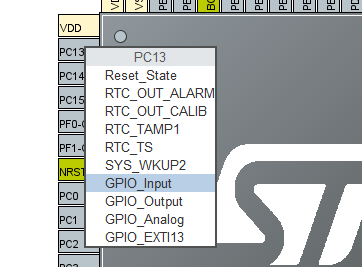
Находим схему нашей отладочной платы (прикладываем ссылку на datasheet нашей платы <https://files.amperka.ru/datasheets/nucleo-usermanual.pdf>)

Находим нужную кнопку



Кнопка USER выходит на пин PC13

Инициализуем его как вход



И переменнуем в BTN



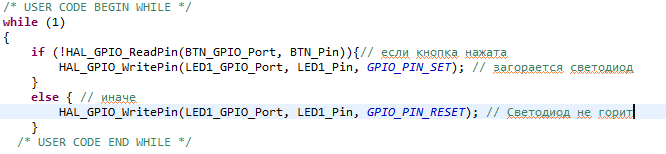
Незабываем про светодиод



Нажимаем на “золотую шестерню” для генерации начального кода



Прописываем код



Если вы сделали все правильно, то когда вы нажимаете кнопку загорается светодиод, если не нажали, светодиод не горит

Усложним задачу

Светодиод загорается когда нажатие будет длительным (2 секунды) и гаснуть при таком же длительном зажатии

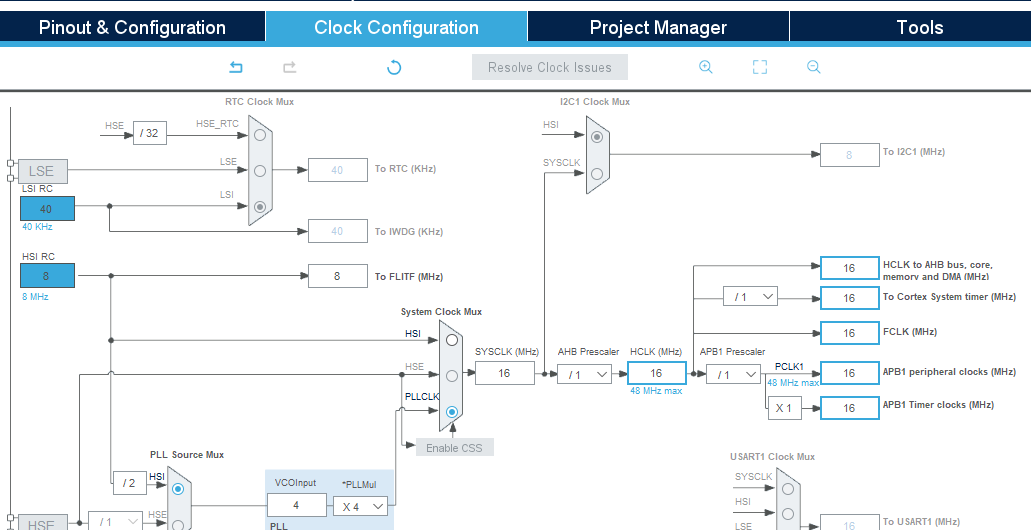
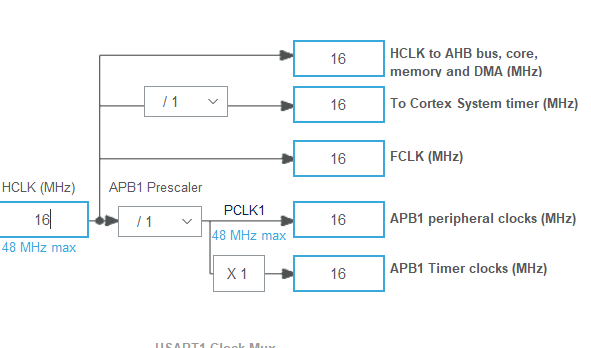
Но как отсчитать нужное время?

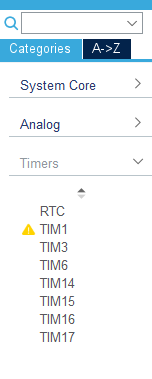
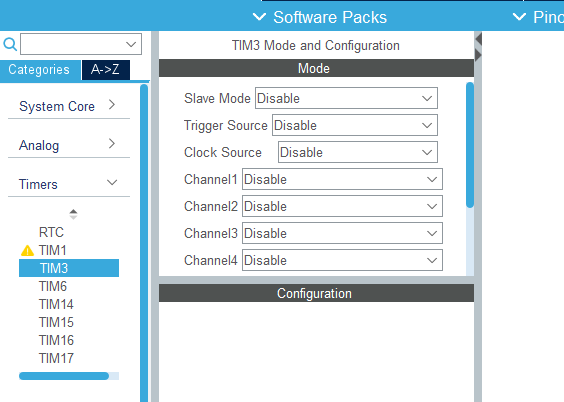
Нужен **таймер**

В STM32 есть множество таймеров, но мы воспользуемся самым простым

Инициализируем таймер под номером 3 (TIM3)

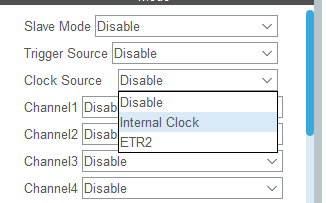
Настроем тактирование внутреннего таймера на 16 Mhz

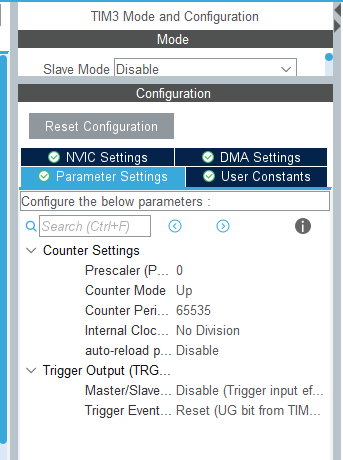
 

В меню настроек таймеров настроим таймер на тактирование от внутренний шины

Internal Clock это тактирование от внутренний шины



Теперь доступны расширенные настройки



Prescaler это предделитель

В нем мы указываем - (частоту \*100) - 1

При частоте в 16 MHz мы получим истенную частоту в 100GHz

Скорость счета получается 10мкс

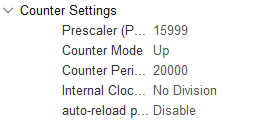
Couter Mode – этвечает за порядок счета

UP – это счет с 0 до N

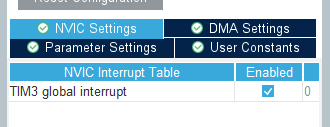
DOWN – это счет с N до 0

Counter Period это конечное число для счета

Учитывая скорость счета ставим 2000 мкс = 2 секунды

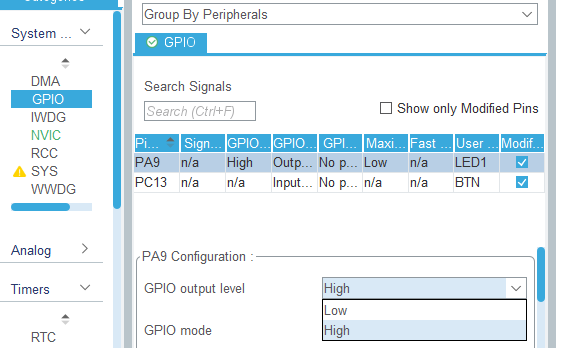


В пункте NVIC Settings ставим сброс до 0 при перепонении таймера



И меняем начальное положение светодиода на High

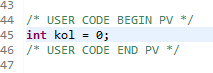
Через Sistem – GPIO – GPIO output level

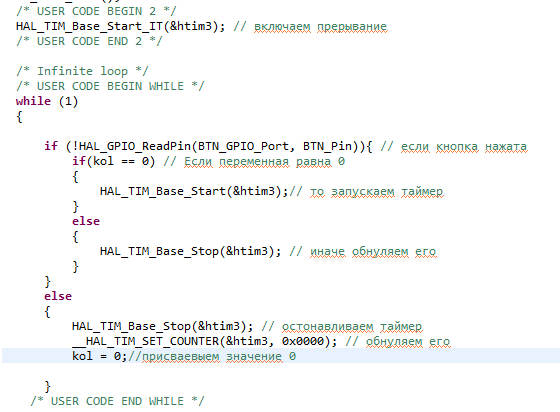


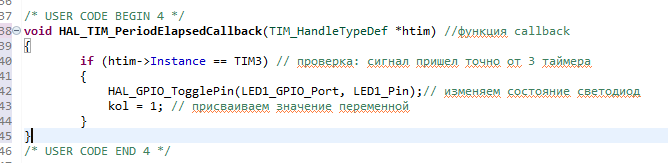
Нажимаем на “золотую шестерню” для генерации начального кода



Прописываем код







Загружаем код в отладочную плату 

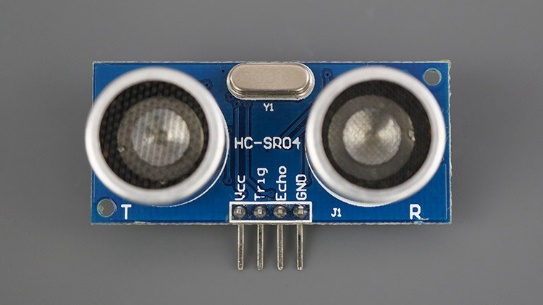
Если вы правильно прописали код, то xерез 2 секунды у нас загорается светодиод. Если же продолжить держать кнопку то ничего происходить не будет. Повторное нажатие и удержание выключит светодиод.

**Ультрозвуковой датчик**

Новая задача

Измерить расстояние с помощью ультрозвука

Мы используем ультрозвуковой датчик “HC-SR04”



Датчик питается от 5 вольт и имеет собственный интерфейс взаимодействия.

Подключение к датчику осуществляется посредством 4 ножек:

Vcc – питание 5v

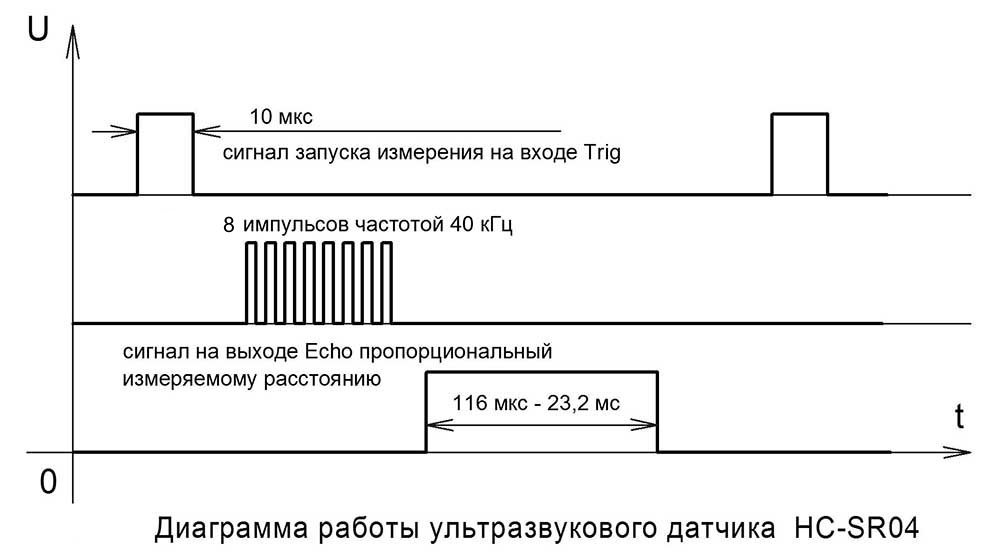
Trig – успускание сигнала

Echo – принимание сигнала

GND – заземление

Работа интерфейса

1. Посылается сигнал (10 микросикунд) для запуска Trig
2. Если Echo принимает его обратно, то выдает импульс



Что бы измерить расстояние нужно знать скорость и время

Скорость звука равна около 340 метров в секунду

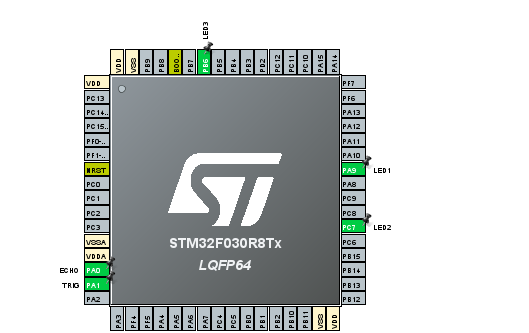
А время мы сможем измерить с помощью таймера (смотри тему раньше)

Инициализируем пины

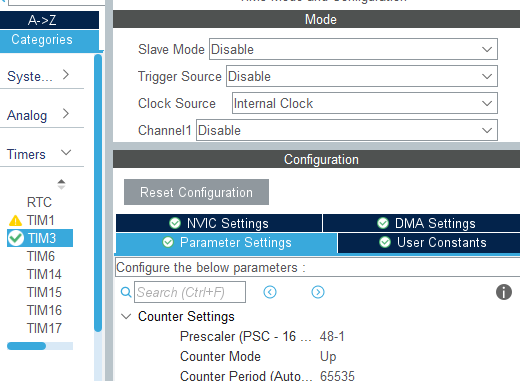
ECHO на GPIO\_Input

Trig на GPIO\_Output

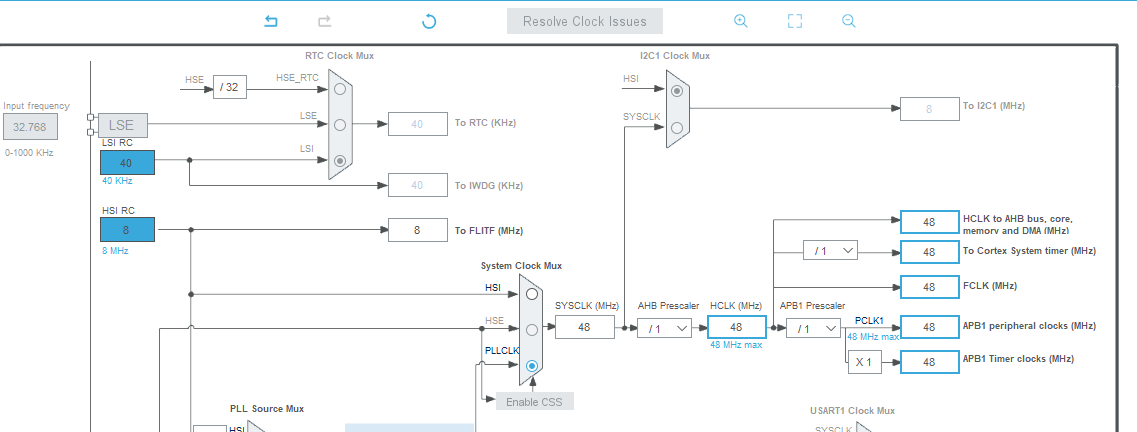
И светодиоды на GPIO\_Output



Настраиваем таймер



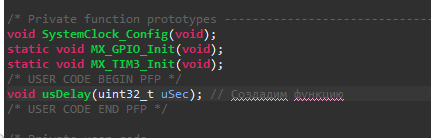
И тактовую частоту

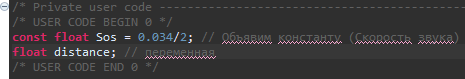


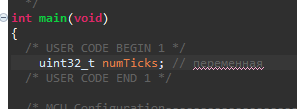
Нажимаем на “золотую шестерню” для генерации начального кода

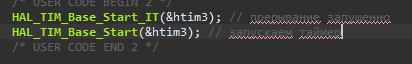


Приступаем к написанию кода



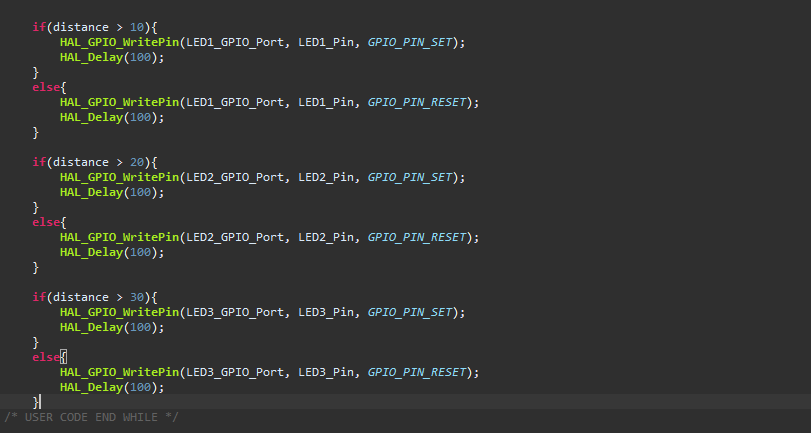


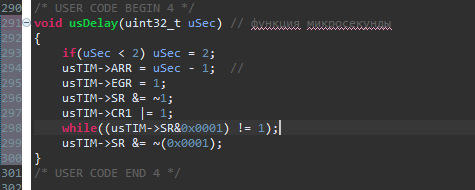


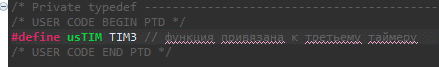




Далее следует блок кода, при выполнении которого будут загорятся светодиоды в порядке возрастания расстояния







Если вы сделали всё правильно, то если расстояние:

Меньше 10 см - светодиоды не горят

Больше 10 см – горит один светодиод

Больше 20 см – горит 2 светодиода

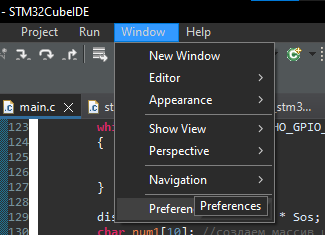
Больше 30 см – горят все светодиоды

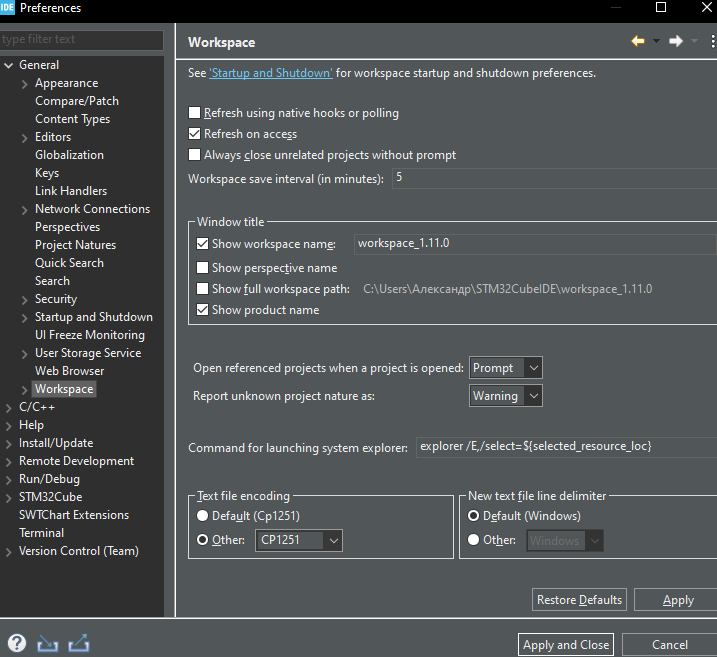
**ЖК- дисплей mt-16s2h**

[DATASHEET](https://docs.yandex.ru/docs/view?tm=1698745193&tld=ru&lang=ru&name=MT-16S2H.pdf&text=mt-1652h&url=https%3A%2F%2Fwww.melt.com.ru%2Fdocs%2FMT-16S2H.pdf&lr=213&mime=pdf&l10n=ru&sign=742a82ffbe286dd2e26e08b3e5b74cbc&keyno=0&nosw=1&serpParams=tm%3D1698745193%26tld%3Dru%26lang%3Dru%26name%3DMT-16S2H.pdf%26text%3Dmt-1652h%26url%3Dhttps%253A%2F%2Fwww.melt.com.ru%2Fdocs%2FMT-16S2H.pdf%26lr%3D213%26mime%3Dpdf%26l10n%3Dru%26sign%3D742a82ffbe286dd2e26e08b3e5b74cbc%26keyno%3D0%26nosw%3D1)

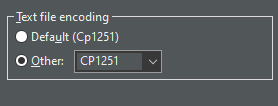
ВНИМАНИЕ!

Перед началом работы необходимо поменять работу кодировки интерпретатора на CP1251





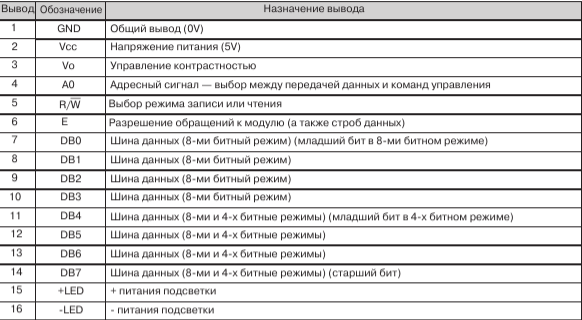
ПРИПИСЫВАТЬ КОДИРОВКУ НУЖНО ВРУЧНУЮ



Новая глава – новый модуль.

Жидкокристаллический дисплей (LCD) mt-16s2h имеет 16 столбцов и 2 строки (16/2). Собственные интерфейс, 2 режима работы (4 и 8 бит), имеет встроенные символы как на латинице, так и на кириллице. От разработчика нам досталось 16 пинов для управления дисплеем.

Информация о каждом из пинов привидяна в таблице



Кодирование символов происходит с помощью 16 – ричной систем е исчисления. Но шифровка отображаемых символов (алфавит символов) и их запись в микроконтроллер происходит в уже при загрузке кода в микроконтроллер.

Таблица символов можно найти в [DATASHEET](https://docs.yandex.ru/docs/view?tm=1698745193&tld=ru&lang=ru&name=MT-16S2H.pdf&text=mt-1652h&url=https%3A%2F%2Fwww.melt.com.ru%2Fdocs%2FMT-16S2H.pdf&lr=213&mime=pdf&l10n=ru&sign=742a82ffbe286dd2e26e08b3e5b74cbc&keyno=0&nosw=1&serpParams=tm%3D1698745193%26tld%3Dru%26lang%3Dru%26name%3DMT-16S2H.pdf%26text%3Dmt-1652h%26url%3Dhttps%253A%2F%2Fwww.melt.com.ru%2Fdocs%2FMT-16S2H.pdf%26lr%3D213%26mime%3Dpdf%26l10n%3Dru%26sign%3D742a82ffbe286dd2e26e08b3e5b74cbc%26keyno%3D0%26nosw%3D1) на странице 8

(Заметка разработчиков: Мы нашли этот дисплей совершенно случайно и захотели его запустить. Проверка работоспособности на Arduino была успешно. Но там была готовая библиотека со всеми пинами и командами, а у нас такой не было. Долгие поиски в интернете успехов не дали, пришлось писать библиотеку с абсолютного нуля. Неделя страданий и библиотека готова. От таких страданий мы вас избавим. ПРИКЛАДЫВАЕМ ГОТОВУЮ БИБЛЕОТЕКУ В ПРИЛОЖЕНИИ К МЕТОДИЧКЕ (путь до файла ...Приложение к методическим рекомендациям/Библиотеки/ЖК-экраны/mt-16s2h)

**Создаем новый проект**

Для корректной работы библиотеки необходимо правильно назвать пины.

Все пины на GPIO\_Output

Пины общего назначения (их называем вне зависимости от выбранного режима работы)

RW

EN

A0

ДЛЯ РАБОТЫ В 4 BIT-ом режиме

DB4

DB5

DB6

DB7

ДЛЯ РАБОТЫ В 8 BIT-ом режиме

DB0

DB1

DB2

DB3

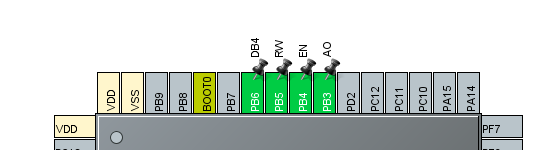
DB4

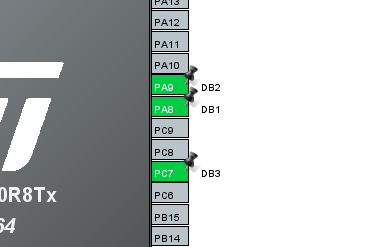
DB5

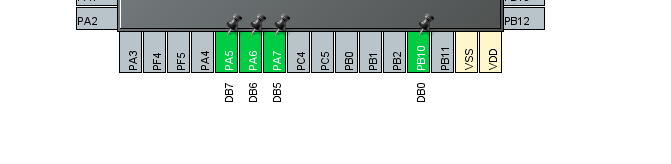
DB6

DB7

Для удобства подключения мы расположили пины так







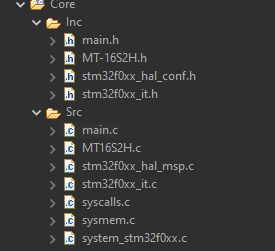
Нажимаем на “золотую шестерню” для генерации начального кода



Переходим к импортированию библеотеки

**Импортирование библиотеки**

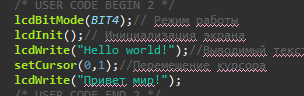
Файл с расширение .с импортируем в проект с папкой **Src,** а расширение .h в папку **Inc**



Прописываем подключение файла библиотеки в файле main.c



**Переходим к написанию коду**



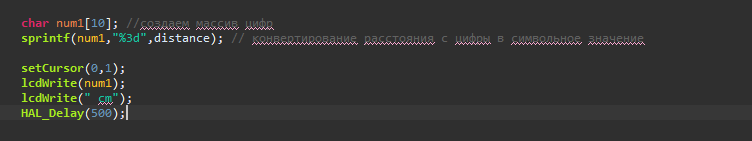
Если вы сделали всё правильно то, у вас должно получиться следующее:



Усложним задачу

Пусть на экран выводится расстояние, считанное с ультразвука

Копируем код с ультразвука в наш проект и добавим пару строчек кода



В итоге должно получится следующее

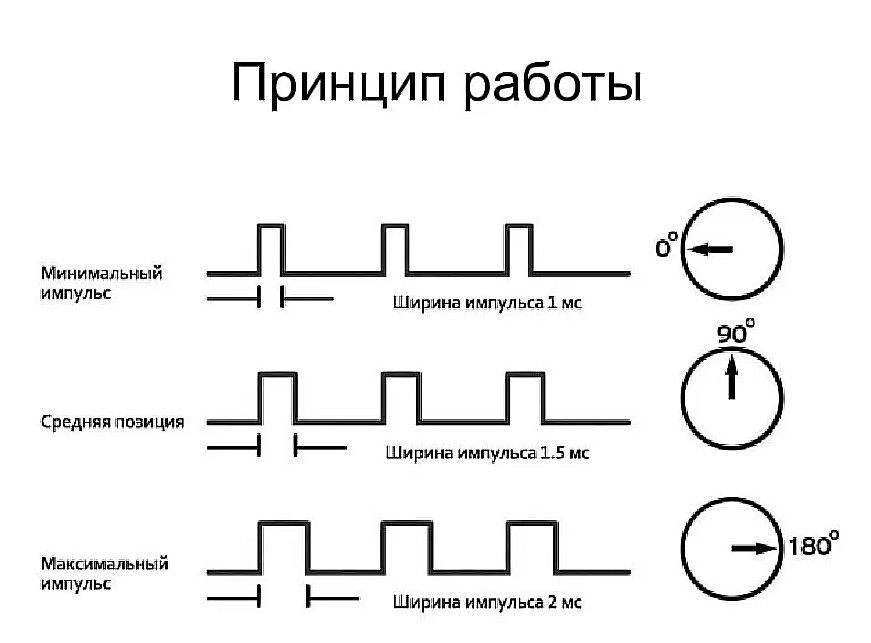


**Сервопривод**

Новый проект: Заставим работать сервопривод и вставать на определенные углы

Теория:

Сервопривод работает за счет ШИМ сигнала (Широтно Импульсная Модуляция), в зависимсти от частоты и времяни подачи этого сигнала – происходит повор привода

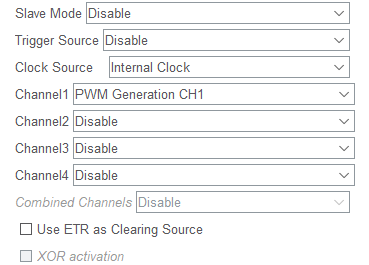


У нас сервопривод модели **SG90** ([DATASHEET](https://docs.yandex.ru/docs/view?tm=1699441570&tld=ru&lang=en&name=sg90-9g-servo-motor-mini-datasheet.pdf&text=sg90%20сервопривод%20даташит&url=https%3A%2F%2Fpdf.direnc.net%2Fupload%2Fsg90-9g-servo-motor-mini-datasheet.pdf&lr=213&mime=pdf&l10n=ru&sign=0d3a98d06d7ad04136d583be9013714a&keyno=0&nosw=1&serpParams=tm%3D1699441570%26tld%3Dru%26lang%3Den%26name%3Dsg90-9g-servo-motor-mini-datasheet.pdf%26text%3Dsg90%2B%25D1%2581%25D0%25B5%25D1%2580%25D0%25B2%25D0%25BE%25D0%25BF%25D1%2580%25D0%25B8%25D0%25B2%25D0%25BE%25D0%25B4%2B%25D0%25B4%25D0%25B0%25D1%2582%25D0%25B0%25D1%2588%25D0%25B8%25D1%2582%26url%3Dhttps%253A%2F%2Fpdf.direnc.net%2Fupload%2Fsg90-9g-servo-motor-mini-datasheet.pdf%26lr%3D213%26mime%3Dpdf%26l10n%3Dru%26sign%3D0d3a98d06d7ad04136d583be9013714a%26keyno%3D0%26nosw%3D1)) и для подходит частота в 50 Gz.

Переходим к настройке таймеров и пинов.

Тактирование от внутренней шины

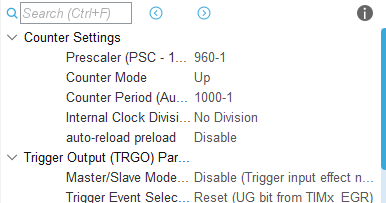
И выход тактирования на первый канал



Для частоты в 50 Gz нужно выставить:

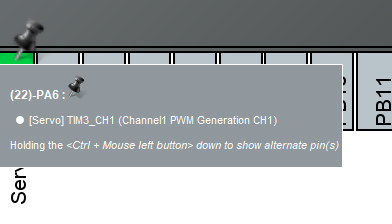
Предделитель на 960-1

Конечное число счёта на 1000-1



Необходимо найти пин с выходом нужной шины

В нашей плате это PA6 (мы назвали его Servo)



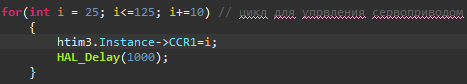
Нажимаем на “золотую шестерню” для генерации начального кода



Переходим к написанию кода



А в цикле while прописываем следующее



Для изменения градусов нужно менять i+=

**OLED дисплей 1306**

Есть 2 версии: они отличаются интерфейсами взаимодействия (I2C и SPI). Так же есть различие в размере экранов (128\*32 и 128\*64 пикселей).

Рассмотрим дисплей с интерфейсом I2S и размером 128\*32 пикселя

Используется готовая библиотека Oled1306 (найти библиотеку можно найти по адресу (…Методичка STM32\Приложение к методическим рекомендациям\Библиотеки\ЖК-экраны\Oled1306 32)

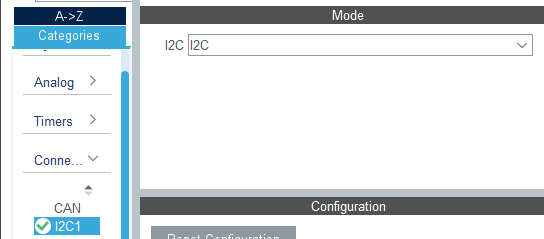
I2S один из самых удобных интерфейсов

Всего 4 пина (5v,GND,SDA,SLC) и простота настройки делают его фаворитом среди интерфейсов

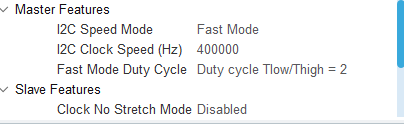
Настройка портов:

В категориях выбираем Connectivity-I2C1-Mode I2C

Режим Fast mode



Режим Fast mode



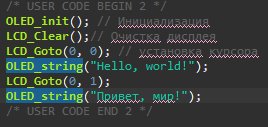
Обратим внимание на пины

IDE автоматически выдаст нужные пины

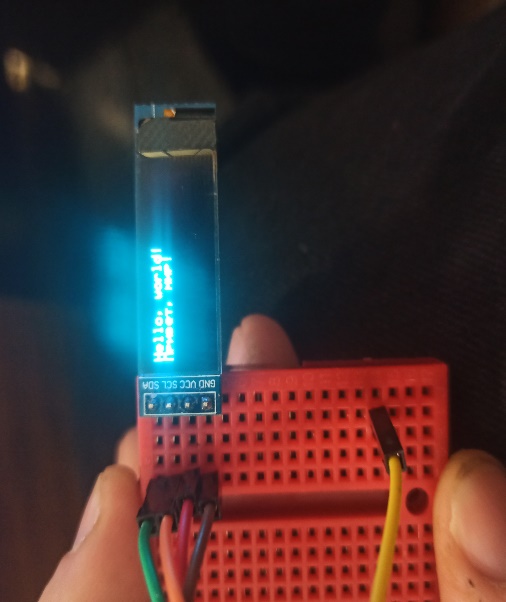
К ним и подключаемся

Нажимаем на “золотую шестерню” для генерации начального кода





Если вы сделали всё правильно, то дисплей должен показывать:



**USART интерфейс. Bluetooth HC-06**

*ВНИМАНИЕ!*

*Данная глава требует от читателя навыка пайки.*

*Если вы не умеете или плохо паяйте, пожалуйста обратитесь к лицам имеющие навыки пайки. Авторы не несут ответственности за сопутствующий ущерб. Глава имеет информационный характер и не призывает к действиям. Все действия вы делаете на свой страх и риск.*

(Универсальный синхронный приемник-передатчик)

USART простой последовательный порт легкий для освоения и понимания, ещё проще им пользоваться (если вы работали с Arduino, то вы уже знаете как с ним работать).

Ярчайший представителей модулей с интерфейсом USART это – Bluetooth модуль HC-06. Им и будем передавать данные.

Проект:

С помощью смартфона, через связь bluetooth, управлять светодиодом, отправляя команды.

Для корректного обмена данными необходимо правильно оперировать параметрами настройки

1. Скорость передачи данных (Обычно 9600 бит в секунду)
2. Кол-во Стоп-бит (По умолчанию 1)
3. Чётность

Эти параметры мы укажем в настройках проекта

Воспользуемся самым распространенным типом подключения -Асинхронным

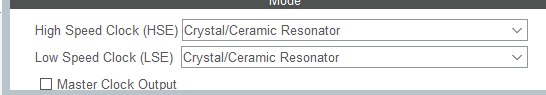
Пены Rx и Tx надо поменять местами.

Подключаем их так: Rx -> Tx

Tx->Rx

Настраиваем пины

Включаем тактирование



Включаем USART (мы воспользовались usart2)

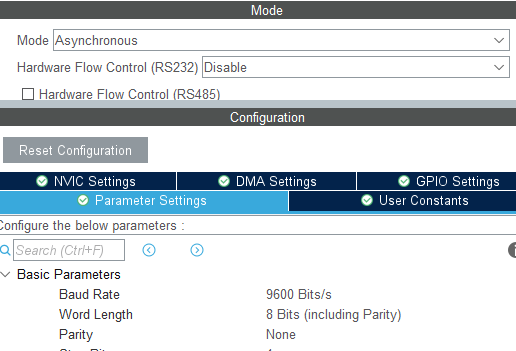
*Внимание!*

*Некоторые пины интерфейса USART не работают по умолчанию*

*В плате STM32 F030R8T6 Nucleo не работает usart2 как пины (он подключен как последовательный интерфейс виртуального COM порта для отладки).*

*Чтобы работали пины необходимо перепаять перемычки на обратной стороне платы (SB13 и SB14 на места SB62 и SB63). На других платах перемычки могут отличатся. Смотрите Datasheet для вашей платы.*

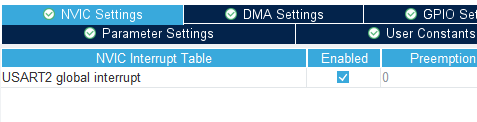
Mode выбираем Asynchronous



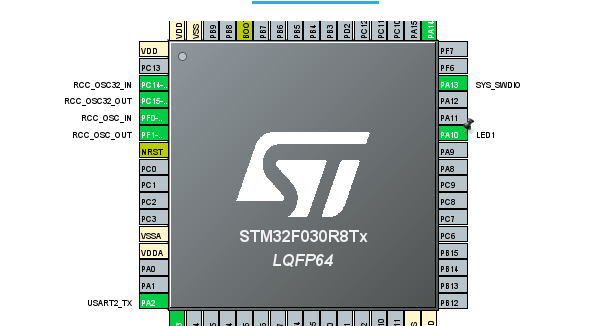
Baud Rate ставим 9600

Остальное без изменений

Незабываем включить глобальное прерывание



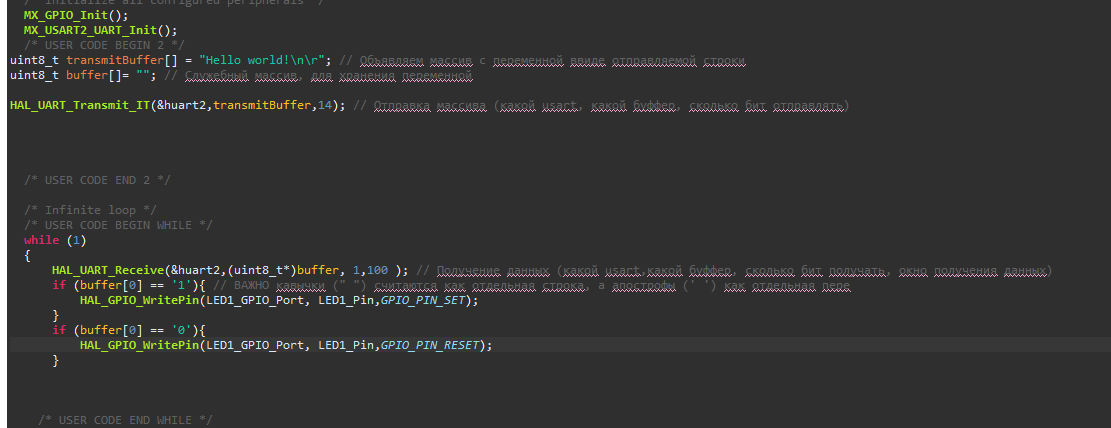
Не забываем про GPIO\_Pin\_Output для светодиода



Нажимаем на “золотую шестерню” для генерации начального кода



Пропишем код



Если вы сделали всё правильно, то:

1. Подключаемся к модулю по Bluetooth
2. В магазине приложений скачиваем терминал Bluetooth
3. При отправке (1) через телефон должен загораться светодиод
4. При отправке (0) через телефон светодиод должен гаснуть