Первая программа

В первой главе мы научились настраивать среду программирования, собирать проект и загружать его в контроллер «Рудирон».
В этой главе мы научимся основам создания программ для контроллеров подобного типа. Написание программы связано не только со знанием языка программирования С++, но и знание электроники, схемотехники, физики и т.д.

Но сначала давайте рассмотрим скелет (структуру) любой программы:

```
main(){
        void setup(){
        }
        void loop(){
        }
    }
```

Обязательная в С++ функция main() создаётся автоматически и на экране она не отображается.
Результатом того, что видит программист будет:

```
void setup(){
}
void loop(){
}
```

Это две обязательные функции в любой программе.

- Функция setup() вызывается только один раз при старте контроллера. Именно она выставляет все базовые настройки. Что такое базовые настройки мы с вами изучим ниже.
- Функция loop() — циклическая. Она исполняет программы на протяжении всего времени работы контроллера в режиме бесконечного цикла. В этом месте и пишется сама программа.

Может напишем быстренько первую программу?!

Если бы нам надо было написать простенькую программу для компьютера, то программа была бы такой:

```
void main (void)
{
  printf ("Моя первая программа");

}
```

Программа выводит на экран сообщение – Моя первая программа.

Какая простая и очаровательная программа, да еще и текст выводит на экран! Все так просто, потому что программа будет запущена на компьютере где операционная система выполнит множество своих команд, чтобы вывести сообщение на экран. Это мы видим 4 строчки кода из них всего лишь одна команда на печать.

В контроллерах нет операционной системы, нет экрана (хотя его можно подключить, но это тоже требует множества команд), нет клавиатуры и т.д.

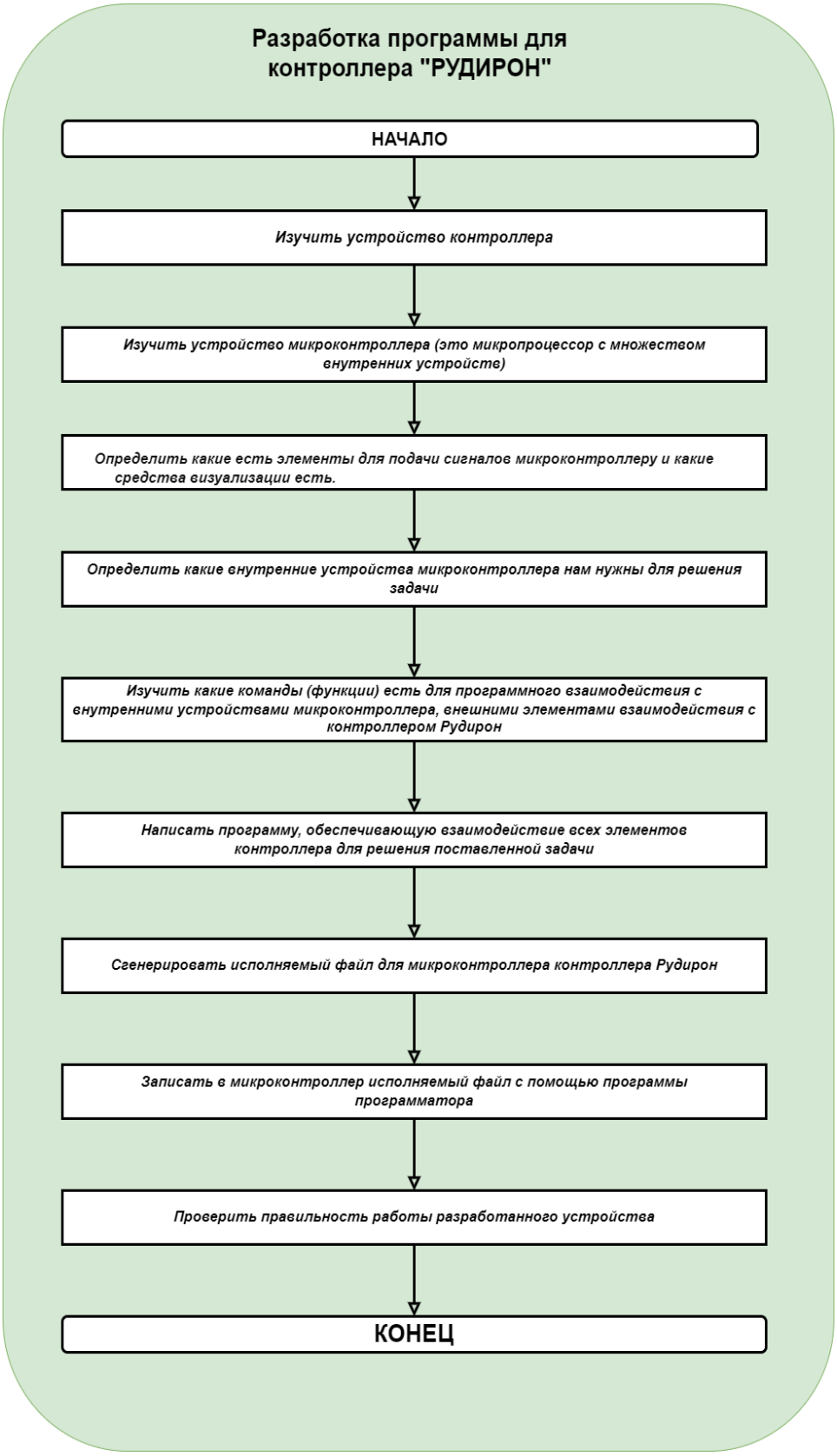
Контроллеры имеют как правило ограниченный набор средств взаимодействия с человеком. Поэтому создание программы в таком случае идет иначе.

Необходимо действовать по следующему алгоритму:

1. Изучить устройство контроллера ««РУДИРОН»»
2. Изучить устройство микроконтроллера (это микропроцессор с множеством внутренних устройств) – это сердце и одновременно мозг устройства.
3. Определить какие есть элементы для подачи сигналов микроконтроллеру и какие средства визуализации есть (светодиоды, экран и т.д.).
4. Определить какие внутренние устройства микроконтроллера нам нужны для решения задачи.
5. Изучить какие команды (функции) есть для программного взаимодействия с внутренними устройствами микроконтроллера, внешними элементами контроллера ««РУДИРОН»».
6. Написать программу, обеспечивающую взаимодействие всех элементов контроллера для решения поставленной задачи.
7. Сгенерировать исполняемый файл для микроконтроллера контроллера ««РУДИРОН»».
8. Записать в микроконтроллер исполняемый файл с помощью программатора.

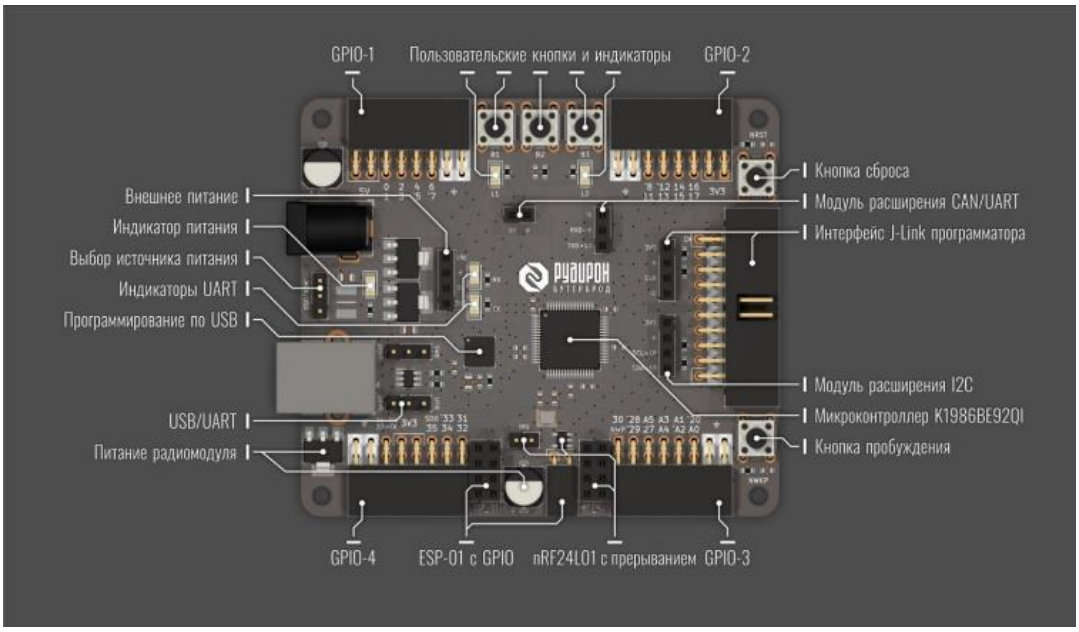
9. Проверить правильность работы разработанного устройства.

Чтобы было проще запомнить нарисуем этот алгоритм!



Кажется, все сложно и запутанно. Не удивительно так как мы с вами создаем программно-аппаратное устройство. А это требует понимания электроники, цифровой техники и программирования. На самом деле мы с вами шаг за шагом не спеша и легко это все освоим!

1. Изучаем на наш контроллер:



Для любого устройства производитель готовит описание своего устройства. Вот и для контроллера ««РУДИРОН»» можно воспользоваться техническим описанием (отдельный документ, прилагаемый к контроллеру). Скачать руководство к контроллеру можно на сайте ***www.РУДИРОН.рф***

Рассмотрим внешний вид контроллера. Мы видим множество разнообразных разъемов и надписей. Не пугаемся и идем дальше.

На плате установлены 2 светодиода. Они обозначены на плате как L1 и L2, есть кнопки, разъемы питания и т.д.

Любое устройство на плате подключено физически к микроконтроллеру.

2. Изучаем микроконтроллер

Микроконтроллер сложное устройство. Он представляет собой микропроцессор с множеством дополнительных устройств, что превращает его по сути в микрокомпьютер. Не такой мощный как настольные компьютеры и ноутбуки.

Невозможно сразу прочесть и понять тысячи страниц технического текста, описывающие микроконтроллер. Поэтому для облегчения работы с контроллером мы написали программное обеспечение, которое позволит вам легко с ним общаться без необходимости глубокого изучения всего микроконтроллера.

На первых порах нам необходима таблица с номерами выводов контроллера «РУДИРОН». Она длинная и будет приведена полностью в конце этого учебника или вы можете скачать ее с сайта ***www.РУДИРОН.рф***. Сейчас мы рассмотрим кусочек этой таблицы.

Описание портов ввода-вывода (GPIO)

Номер	**Альтернативное обозначение**	**Порт**	**ШИМ**	**Внешнее прерывание**	**Альтернативная функция**

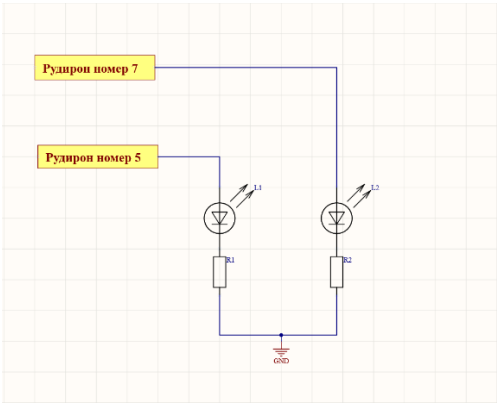
0		F3			SSP1_RXD/CAN2_TX
1		F2			SSP1_FSS/CAN2_RX
2		F1			SSP1_CLK/UART2_TXD
3		F0			SSP1_TXD/UART2_RXD
4		A1			
5	L1	A2	+		
6		A3			
7	L2	A4	+		

Посмотрим, как таблица соотносится с реальным положением элементов на плате контроллера.
В таблице описания портов ввода-вывода в столбце «Номер» можно увидеть за какими номерами закреплены светодиоды L1 – 5 (A2), L2 – 7 (A4). Номера задали разработчики «РУДИРОНа» для вашего удобства, чтобы не искать названия A2 и A4 (это названия выводов самого микроконтроллера). Так будет легче писать программу.

3. Определяем какие есть элементы для подачи визуальной информации для человека. Это светодиоды L1 и L2.

4. Внутреннее устройство микроконтроллера, которое взаимодействует с подключенными светодиодами это порты ввода и вывода GPIO. Звучит очень страшно).
   На самом деле порт ввода вывода сейчас можете рассматривать как выключатель для лампочки. Перевели выключатель в положение TRUE (включить) на лампу (светодиод) поступает электрический ток и мы видим свечение. Переводим выключатель в положение FALSE (выключить) ток перестает течь через лампу (светодиод) и мы сидим в темноте.

   Схема электрическая подключения светодиодов:



Теперь мы понимаем, что нам на следующем шаге нужны команды, которые могут работать с GPIO.

5. Какие у нас есть команды (функции) для работы с портами ввода и вывода GPIO?

   - Установка режима работы вывода порта на вывод электрического тока или на ввод - ***pinMode***(«Номер порта», «Режим ввод или вывод»);
   - Включение или выключение порта - ***digitalWrite***(«Номер порта», «Включить или выключить»);

Команды (функции) для управления большинством внутренних устройств микроконтроллера мы изучим постепенно в ходе написания программ и сборки простых устройств.

6. Написание программы:

   Задача - программа подачи напряжения 3.3 вольта на два номера контроллера (5 и 7) к которым подключены светодиоды L1 и L2.

Пример программы (включение 2-х встроенных светодиодов):

```
#define LED_L1 5
#define LED_L2 7

void setup()
{
      pinMode(LED_L1,OUTPUT);
      pinMode(LED_L2, OUTPUT);
}
```

```
void loop()
{
        digitalWrite(LED_L1, true);
        digitalWrite(LED_L2, true);
}
```

Рассмотрим детально текс программы:

a) Присваиваем осознанные имена номерам контроллера

```
#define LED_L1  5
#define LED_L2  7
```

b) Любой порт GPIO контроллера может работать на ввод либо вывод (выдача напряжения 3.3 вольта или нет).
Первая команда **pinMode(LED_L1,OUTPUT);** настраивает порт на нужный режим. Так как для «зажигания» светодиода необходимо на него с выхода контроллера подать напряжение 3.3 вольта - это режим выхода OUTPUT.

Устанавливаем режим работы вывода контроллера используя команду (более правильно говорить функцию) **pinMode:**

```
pinMode(LED_L1, OUTPUT);
pinMode(LED_L2, OUTPUT);
```

c) Устанавливаем на выходе напряжение 3.3 вольта, чтобы светодиод засветился. Для этого используем команду (функцию) **digitalWrite**.

```
digitalWrite(LED_L1, true);
digitalWrite(LED_L2, true);
```

Программу можно было написать более просто:

```
void setup()
{
      pinMode(5,OUTPUT);
      pinMode(7, OUTPUT);
}
void loop()
{
     digitalWrite(5, true);
     digitalWrite(7, true);
}
```

Как видите в командах (функциях) мы использовали номер вывода контроллера. Но это не удобно! Помнить к какому выводу контроллера подключено то или иное устройство очень тяжело. В больших проектах таких устройств могут быть десятки и легко запутаться какие надо номера выводов указывать в командах.

Для удобства написания кода программы мы можем задать выводам контроллера свои имена:

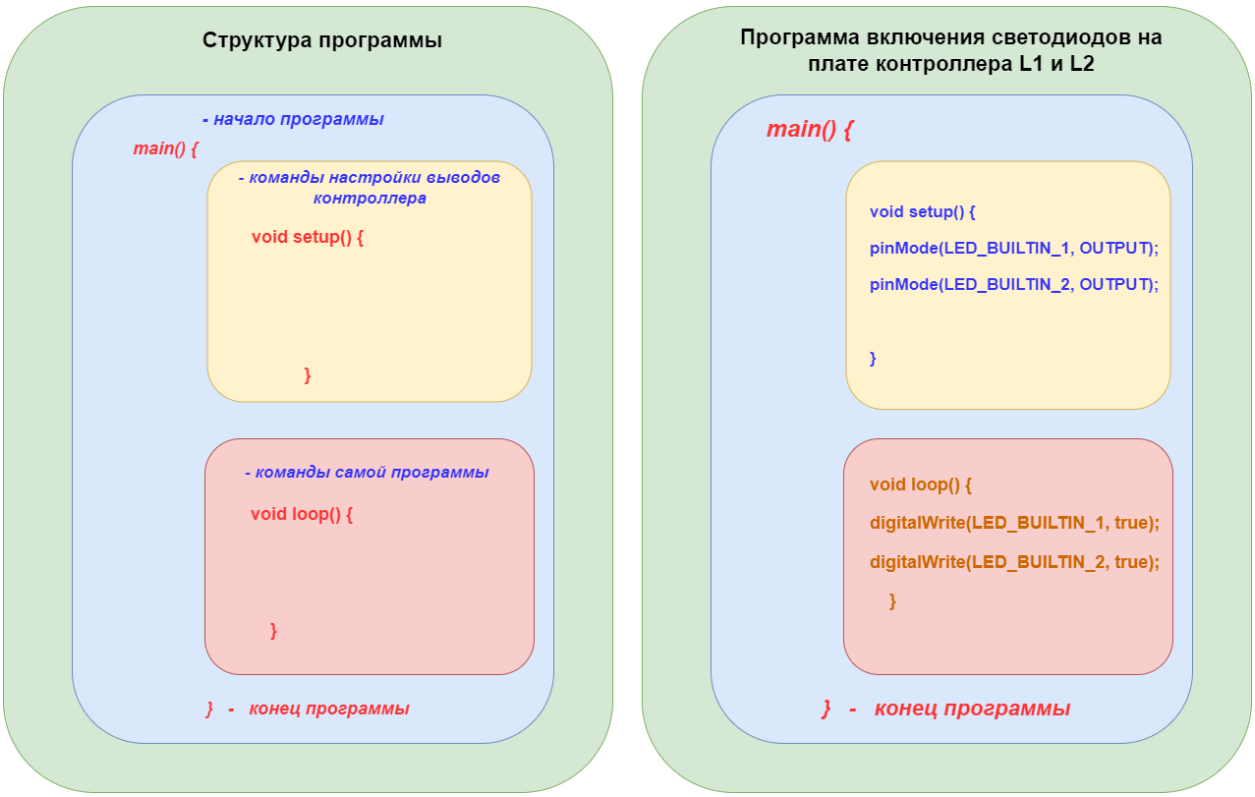
```
#define LED_L1 5
#define LED_L2 7
```

Очень часто разработчики создают заранее такие имена. Вот и в нашем случае вы можете не создавать свои имена, а воспользоваться нашими:

```
void setup()
{
        pinMode(LED_BUILTIN_1, OUTPUT);
        pinMode(LED_BUILTIN_2, OUTPUT);
}
void loop()
{
      digitalWrite(LED_BUILTIN_1, true);
      digitalWrite(LED_BUILTIN_2, true);
}
```

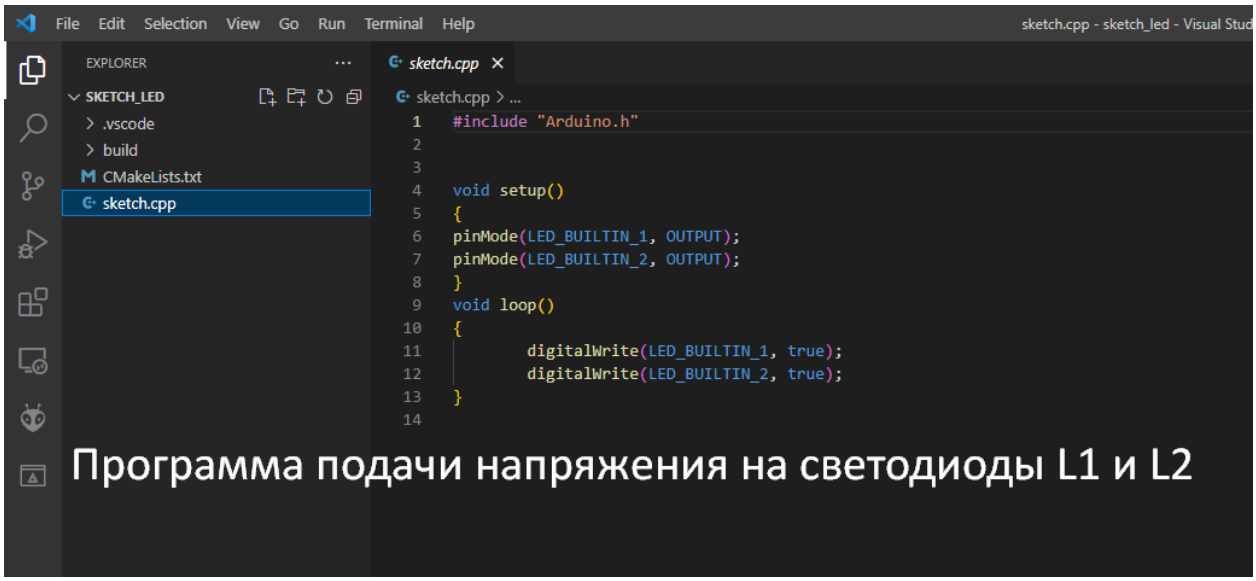
Мы создали имена LED_BUILTIN_1 и LED_BUILTIN_2 для выводов к которым подключены светодиоды на плате L1 и L2. Поэтому вам остается запомнить такие имена и использовать их в своих программах.

Давайте закрепим окончательно структуру программы для нашего контроллера ««РУДИРОН»»:



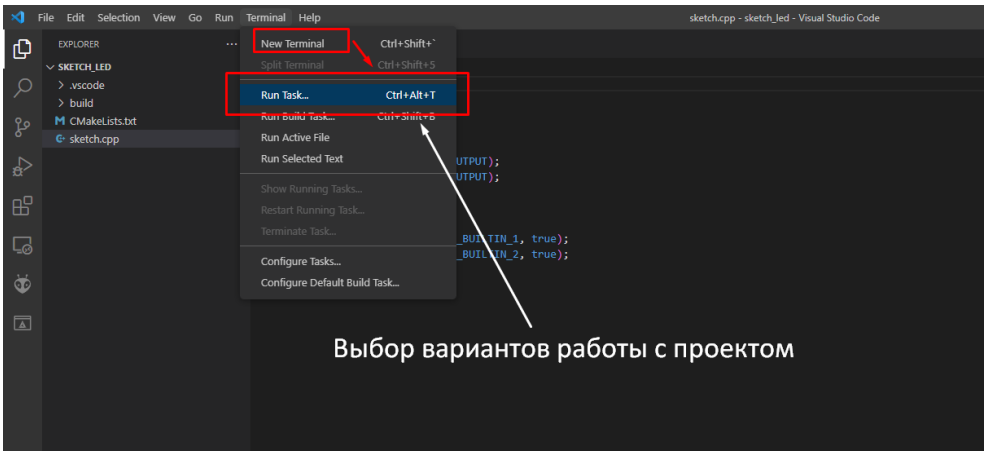
7. Генерируем исполняемый файл для контроллера «РУДИРОН».

Набираем программу в файле sketch.cpp

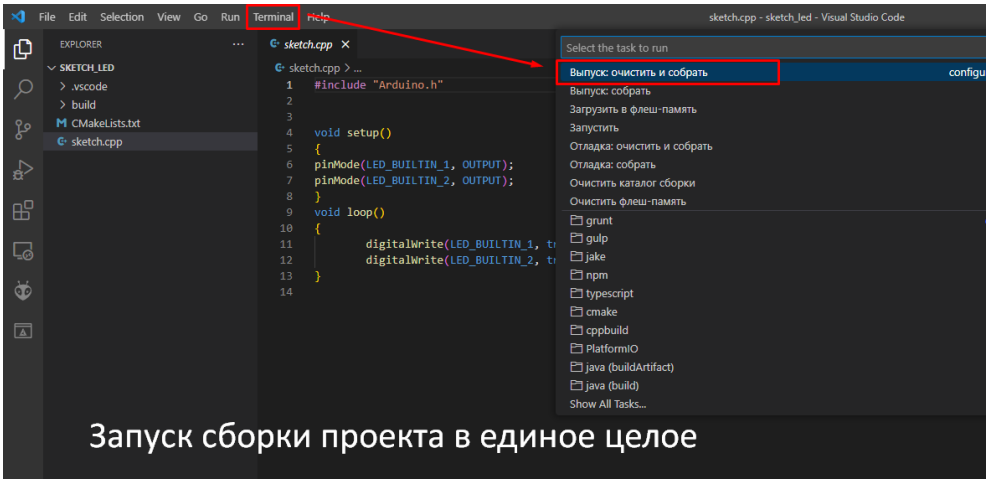


8. Записываем исполняемый файл в контроллер с помощью программатора.

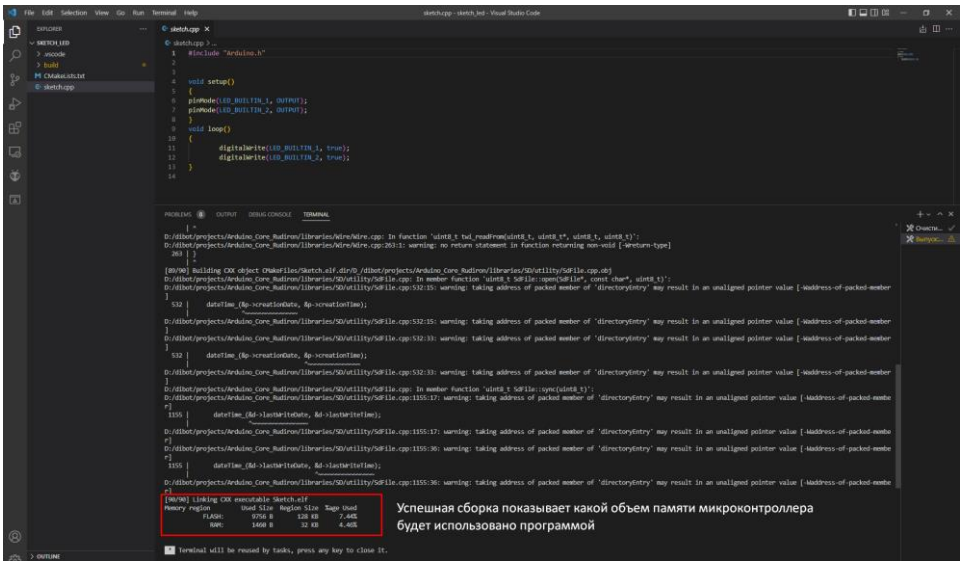
Современные программы состоят не из одного файла с текстом программы как sketch.cpp. Любая программа включает в себя множество других программ. Но чтобы эти программы попали в один исполняемый файл для микроконтроллера необходимо их все собрать в одном месте в исполняемом файле. Для этого необходимо текст программы sketch.cpp перевести в машинный код микроконтроллера и дополнить его кодом из других файлов. Этот процесс называется Сборкой проекта. Когда мы настраивали файл конфигурации проекта CMake как раз и фиксировали в этом файле как среда должна и откуда брать необходимые файлы для сборки в одно единое целое.
Для проведения сборки используем верхнее меню среды ***Terminal – Run Task.***



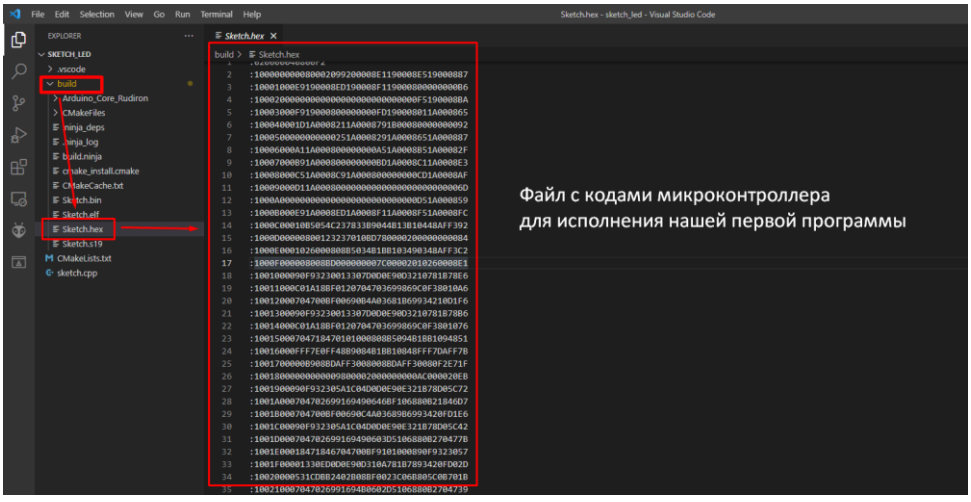
Появится меню с вариантами действий с проектом.
Мы выбираем **Выпуск!**



В случае отсутствия ошибок в программе и правильно настроенной системе сборки мы получим такой экран:



В нижнее окно терминала будет выведена маленькая текстовая таблица с указанием сколько в процентах будет занято памяти FLASH (считайте это жесткий диск с которого запускает программу компьютер) и RAM (это оперативная память как у компьютера) нашей программой.
Итогом такой сборки будет исполняемый файл для микроконтроллера ***Sketch.hex*** в директории ***build.***

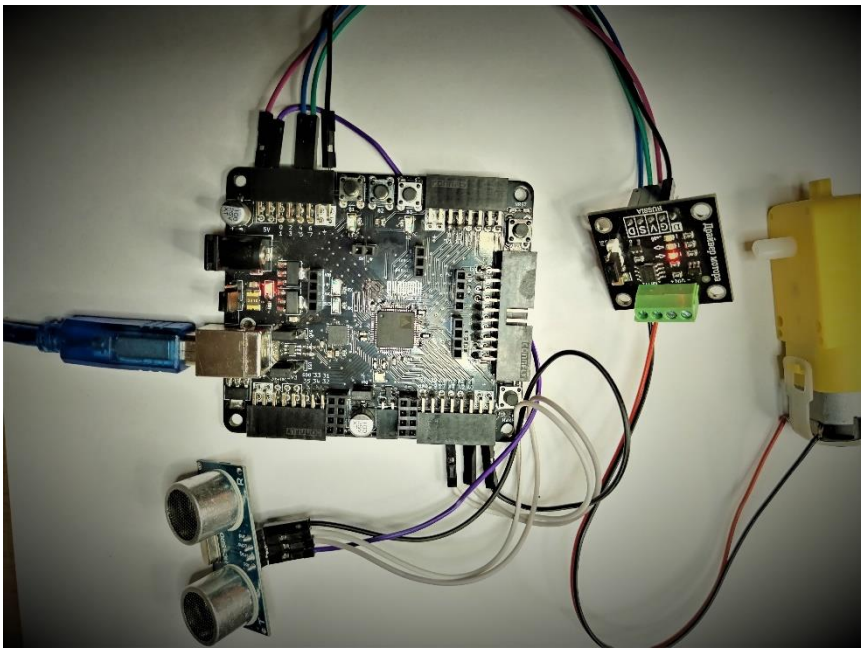


Теперь необходимо этот файл загрузить в контроллер «РУДИРОН». Вернее, сказать записать во FLASH память микроконтроллера. Обычно используется отдельная программа для этого с графическим экраном. Разработчики подключили к Visual Studio программатор и нет необходимости его отдельно вызывать и указывать файл Sketch.hex.

Достаточно вызвать знакомый нам пункт меню ***Terminal-Run Task*** и выбрать меню ***Загрузить в флеш-память.***

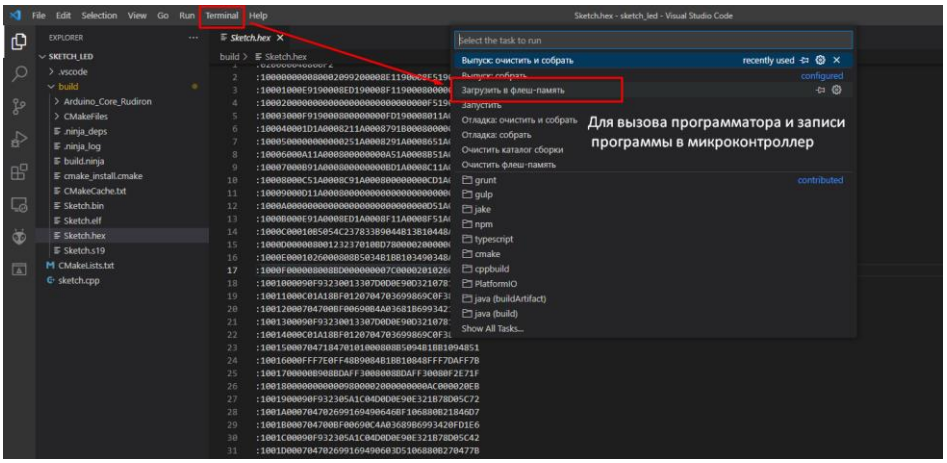
Однако мы забыли про сам контроллер. Программа, что по воздуху перенесется во FLASH память микроконтроллера?!

Перед загрузкой необходимо подключить контроллер к компьютеру через USB кабель.

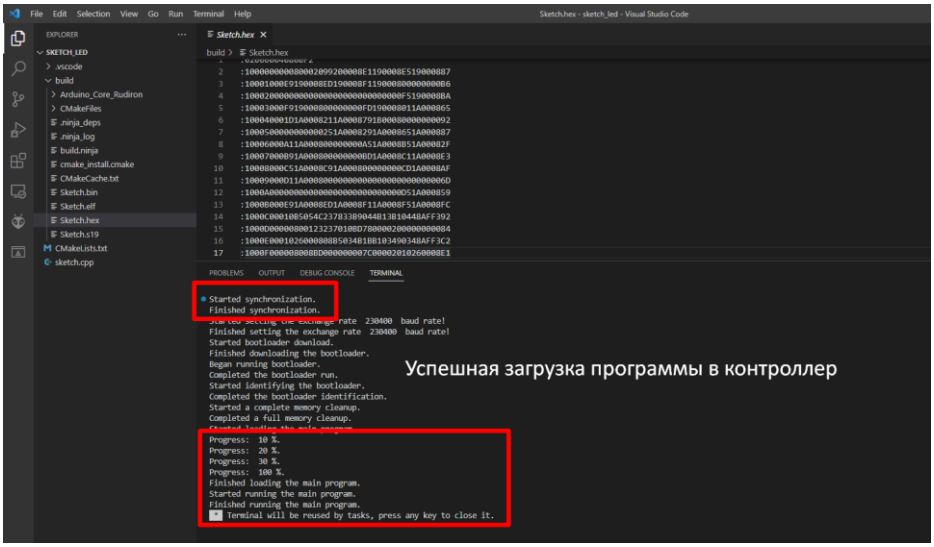


Пример подключения. Обратите внимание светодиоды L1 и L2 не светятся.

Ну а теперь можно запускать запись во флеш-память микроконтроллера.



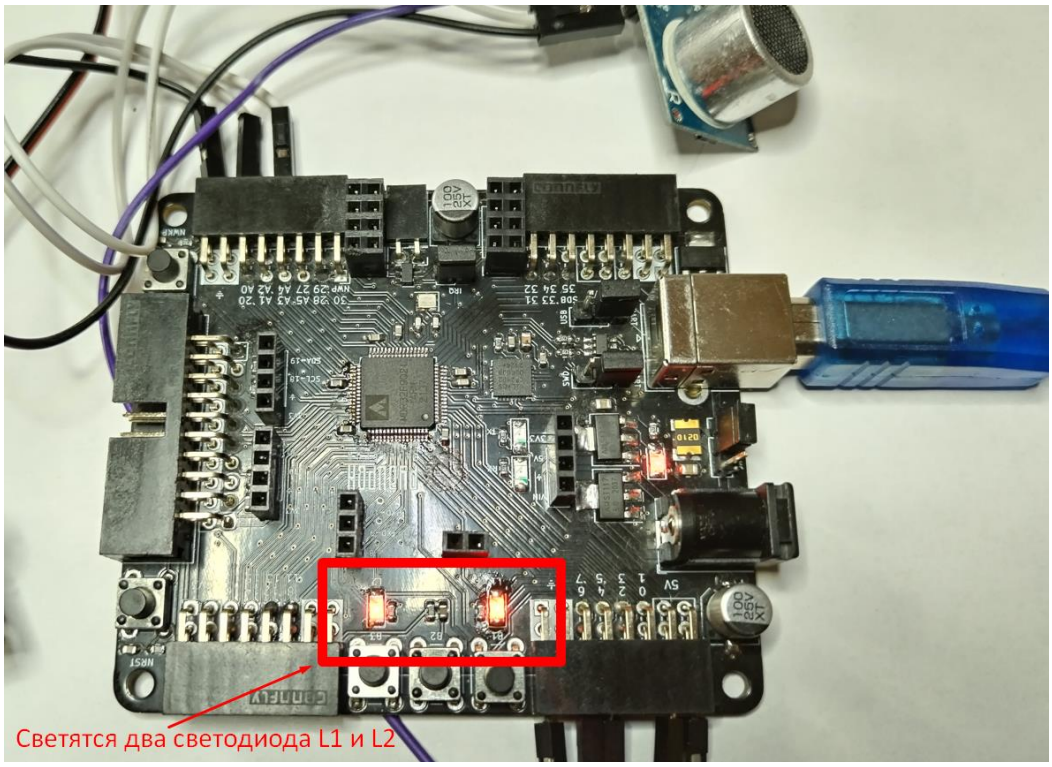
Если все сделано правильно, то получаем окно успешной загрузки.



9. Запускаем контроллер и проверяем работоспособность устройства.

Сразу после загрузки флеш-памяти программа начинает исполняться.

Посмотрим, что там с нашими двумя светодиодами:



Отлично! Первая программа написана, собрана и загружена в контроллер. И она работает!

Дальше все будет идти намного быстрее. Ты наберешь опыта и процесс создания своих устройств будет интересным и увлекательным. Первые шаги самые сложные как у ребенка, который только что встал на ножки.