

Digital electronics essentials

Здесь когда-нибудь будет дизайн

Занятие 8. Микроконтроллеры

Recap?

Recap?

- Цифровые схемы
- Логические уровни
- Базовые блоки
 - AND
 - OR
 - NOT
 - NAND и NOR
 - XOR
- Сложные схемы
 - Полусумматор
 - Сумматор

Контроллер

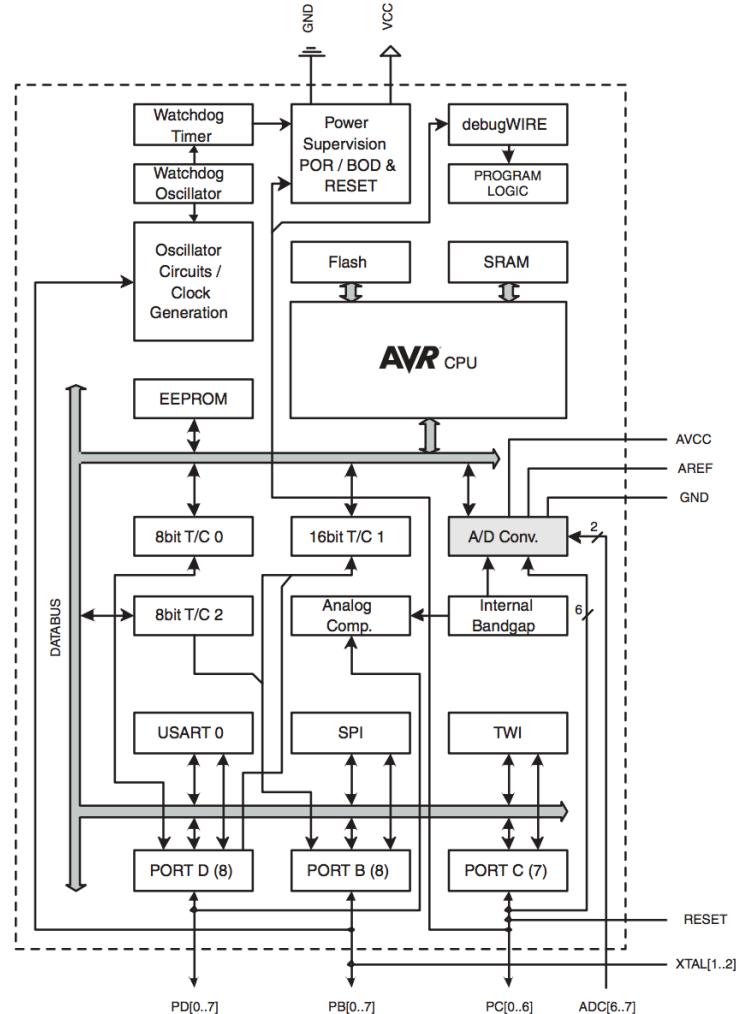
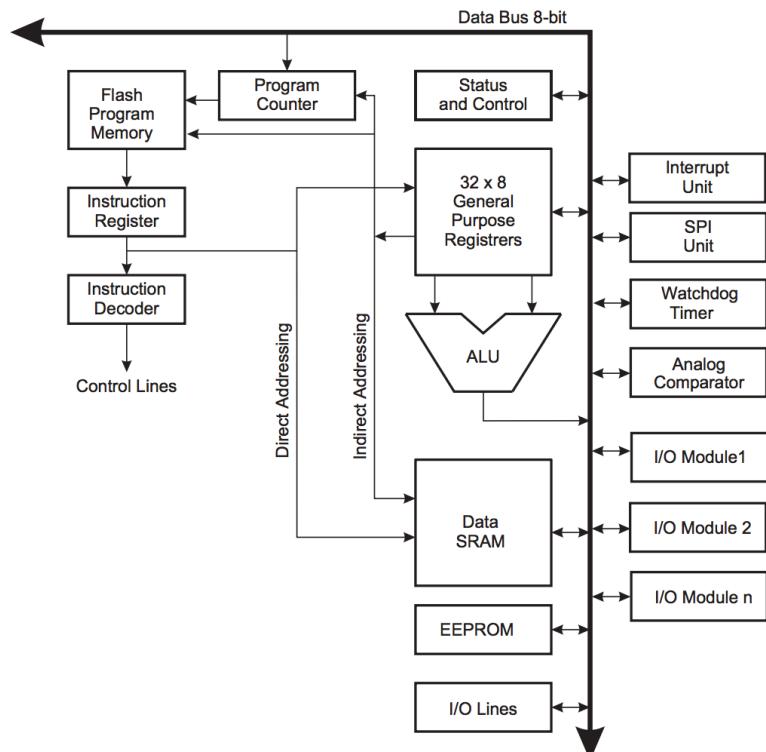
- Законченное устройство, включающее в себя
 - АЛУ (вычислительное ядро ЦПУ)
 - Память
 - Оперативную
 - (не всегда) постоянную
 - Устройства ввода-вывода

Читаем даташит

- http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Figure 2-1. Block Diagram

Структура МК



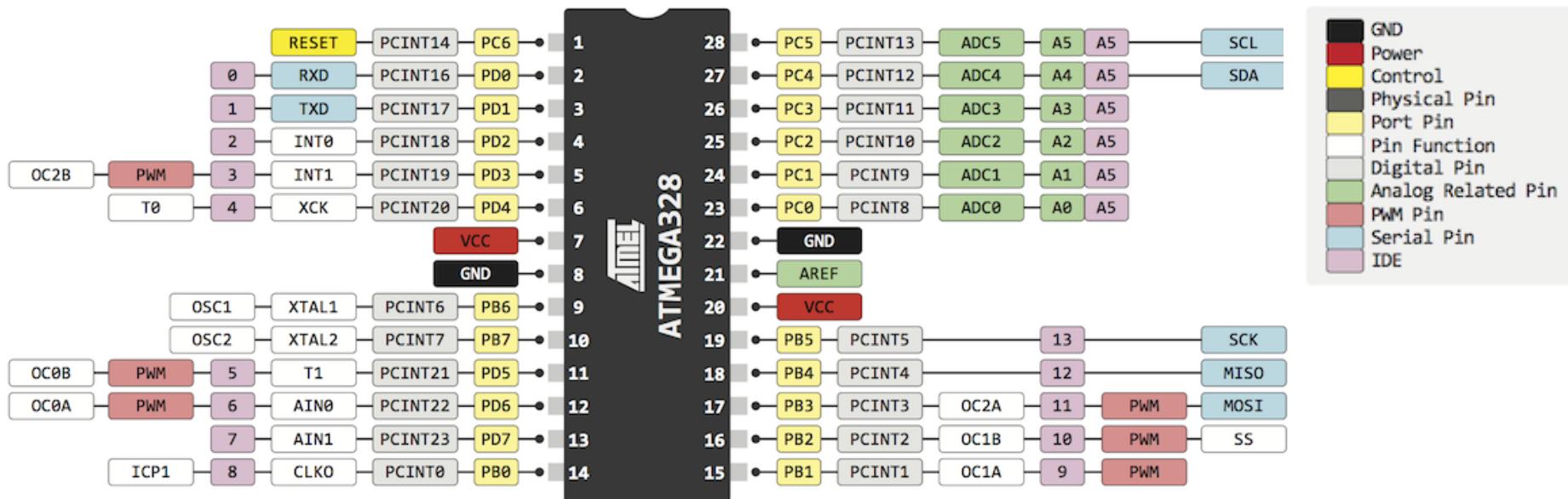
SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W								
Initial Value	0	0	0	0	0	0	0	0	

- Bit 7 – I: Global Interrupt Enable The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.
- Bit 6 – T: Bit Copy: BLD (Bit LoaD) and BST (Bit STore) instructions use the T-bit as source or destination for the operated bit. A bit from a register in the Register File can be copied into T by the BST instruction, and a bit in T can be copied into a bit in a register in the Register File by the BLD instruction.
- Bit 5 – H: Half Carry Flag The Half Carry Flag H indicates a Half Carry in some arithmetic operations. Half Carry Is useful in BCD arithmetic.
- Bit 4 – S: Sign Bit, $S = N \oplus V$ The S-bit is always an exclusive or between the Negative Flag N and the Two's Complement Overflow Flag V.
- Bit 3 – V: Two's Complement Overflow Flag The Two's Complement Overflow Flag V supports two's complement arithmetic.
- Bit 2 – N: Negative Flag The Negative Flag N indicates a negative result in an arithmetic or logic operation.
- Bit 1 – Z: Zero Flag The Zero Flag Z indicates a zero result in an arithmetic or logic operation.
- Bit 0 – C: Carry Flag The Carry Flag C indicates a carry in an arithmetic or logic operation.

Распиновка ATMega328



Сравнение систем

Архитектура	Примеры	Примечания
AVR	Arduino (AtMega 328, 2560, 32U4)	8 бит, разработана в 1996.
PIC	EasyPIC	
PIC32	EasyPIC Fusion, Mini-32	
ARM Cortex M0	mBed LPC11U24	
ARM Cortex M3	mBed LPC1768	
ARM Cortex M4	STM32Discovery	
	Espruino	Используется в модулях IskraJS (конструктор Йодо)
ARM Cortex A*	Raspberry Pi BeagleBone * ODroid	Микрокомпьютеры широкого назначения
SuperH (SH)		Используется в автомобильной электронике
x86	Intel Galileo, Intel Edison Intel NUC AMD Geode	Высокое энергопотребление, производительность, совместимость
Xtensa	ESP8266, ESP32	Удобен в качестве SoM

Выбор

- Платформы, созданные для быстрого прототипирования
- => подходят нам:
 - Arduino (AVR)
 - Espruino
 - ESP8266

Устройство Arduino

- Основана на микроконтроллере
- Управляется написанной программой
- Содержит необходимые средства программирования и отладки
 - Подключается к компьютеру без дополнительных средств

Системы лучше и больше

Как писать хороший код и не лажать

Простейший код: Blink

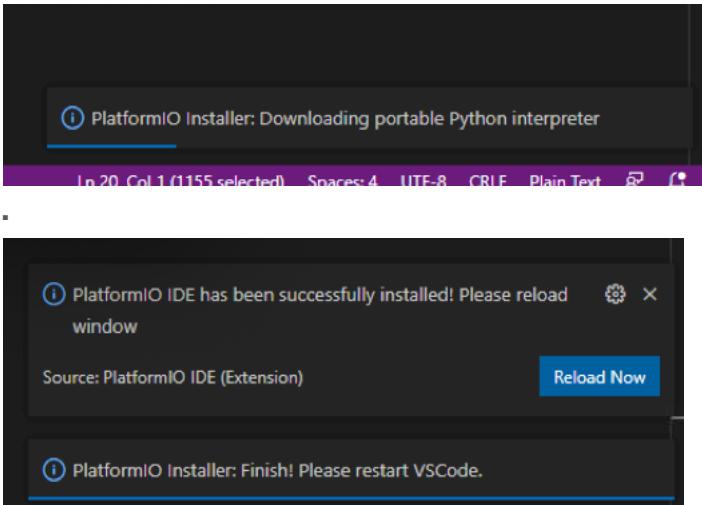
- На плате Arduino Uno и клонах на выводе 13 присутствует индикаторный светодиод

```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

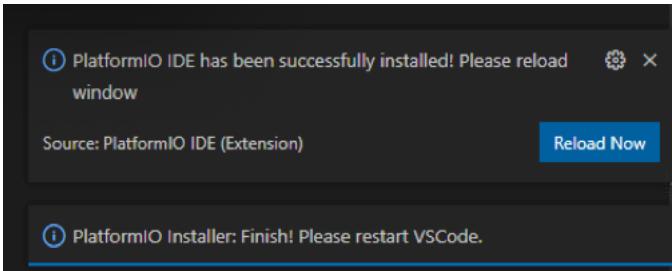
// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                         // wait for a second
    digitalWrite(LED_BUILTIN, LOW);       // turn the LED off by making the voltage LOW
    delay(1000);                         // wait for a second
}
```

Установка систем

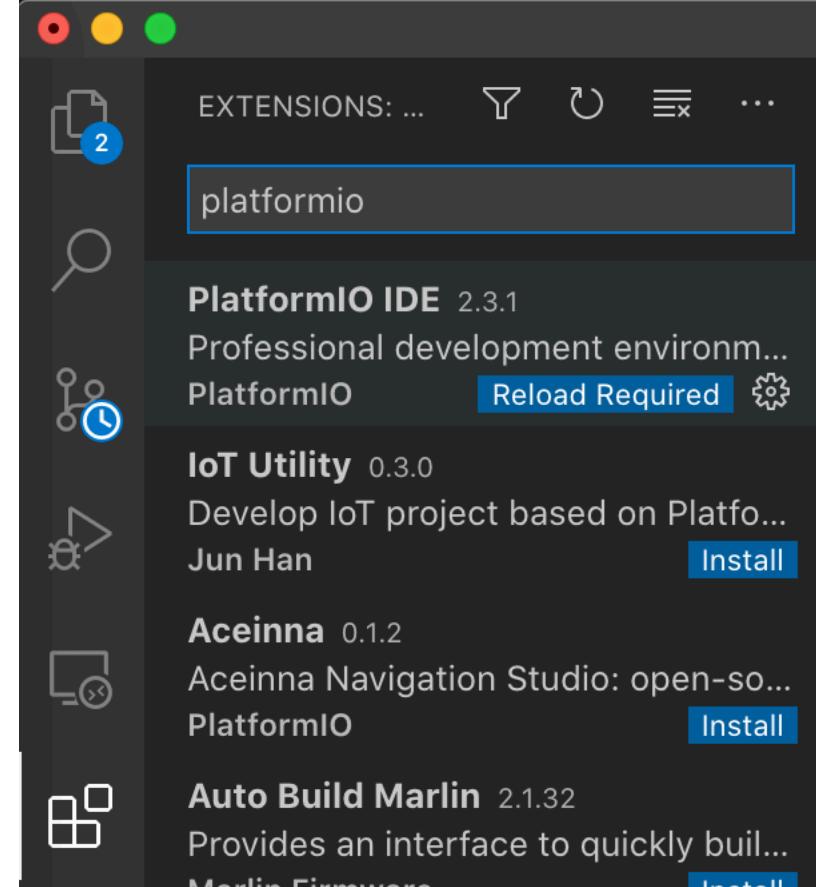
- <https://code.visualstudio.com/Download>
- Extensions -> “PlatformIO”
- ...



- ...



- Открыть PIO Home



PLATFORMIO: QUICK ACCESS

PIO Home X 2do_2021 Untitled-1 ● expert.py universal_expert __init__.py .../rvexpert

2

Open

PIO Account

Inspect

Projects & Configuration

Libraries

Boards

Platforms

Devices

Debug

Start Debugging

Toggle Debug Console

Updates

Library Updates

Platform Updates

Update All

Miscellaneous

PlatformIO Core CLI

Home Projects Inspect Libraries Boards

Welcome to PlatformIO



Core 5.1.0 · Home 3.3.4

Recent News

Platforms -> Atmel AVR -> Install

The screenshot shows the PIO Home extension interface in Visual Studio Code. The left sidebar has a dark theme with various icons for quick access. The main area is titled "PIO Home - Visual Studio Code". The "Platforms" section is currently selected. The central panel displays information about the "Atmel AVR" platform, stating: "Atmel AVR 8-bit MCUs deliver a unique combination of performance, power efficiency and design flexibility. Optimized to speed time to market-and easily adapt to new ones-they are based on the industry's most code-efficient architecture for C and assembly programming". Below this is an "Installation" section with a dropdown menu set to "3.2.0" and a large blue "Install" button. To the right of the "Install" button are tabs for "Boards", "Examples", "Packages", and "Changelog". A search bar labeled "Search board..." is positioned below these tabs. At the bottom of the central panel is a table with columns: Name, Frameworks, MCU, FRQ, ROM, RAM, and Extra. Two boards are listed: "Adafruit Bluefruit Micro" and "Adafruit Circuit Playground Classic". On the right side of the interface, there are sections for "Frameworks" (Arduino, Simba) and "Resources" (Homepage, Repository, Documentation, Vendor, Apache 2.0, GitHub). The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL.

Name	Frameworks	MCU	FRQ	ROM	RAM	Extra
Adafruit Bluefruit Micro	Arduino	ATMEGA32U4	8 Mhz	28 KB	2.5 KB	
Adafruit Circuit Playground Classic	Arduino	ATMEGA32U4	8 Mhz	28 KB	2.5 KB	



Platform has been successfully installed

Platform Manager: Installing atmelavr

Platform Manager: atmelavr @ 3.2.0 has been
installed!

Tool Manager: Installing platformio/toolchain-
atmelavr @ ~1.70300.0

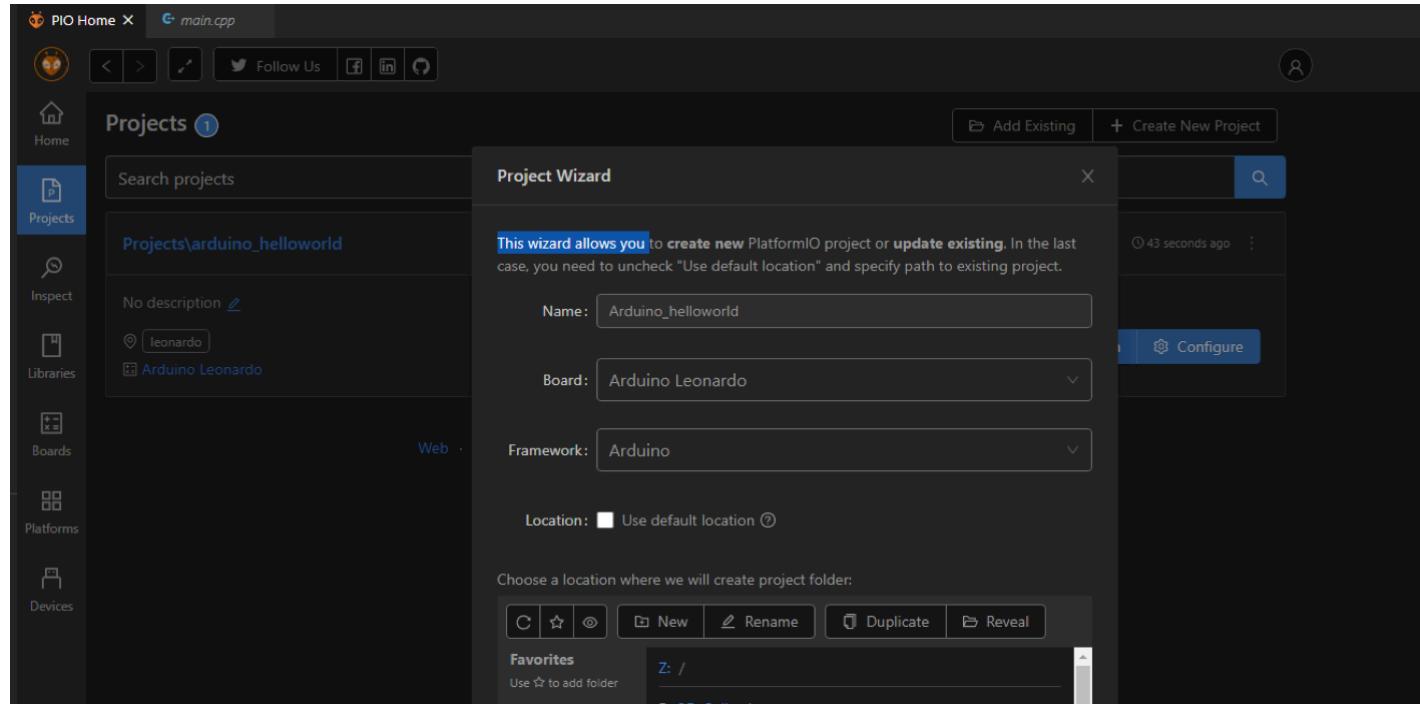
Tool Manager: toolchain-atmelavr @
1.70300.191015 has been installed!

The platform 'atmelavr' has been successfully
installed!

The rest of the packages will be installed later
depending on your build environment.

OK

Projects -> Create new project



Простейший код: Blink (src/main.cpp)

- На плате Arduino Uno и клонах на выводе 13 присутствует индикаторный светодиод

```
// the setup function runs once when you press reset or power the board
void setup() {
    // initialize digital pin LED_BUILTIN as an output.
    pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
    digitalWrite(LED_BUILTIN, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                         // wait for a second
    digitalWrite(LED_BUILTIN, LOW);       // turn the LED off by making the voltage LOW
    delay(1000);                         // wait for a second
}
```

Build: сборка и проверка валидности кода

The screenshot shows the Visual Studio Code interface with the PlatformIO extension open. The left sidebar displays the project structure under 'PROJECT TASKS' for the 'leoardo' board, including options for Build, Upload, Monitor, Upload and Monitor, and Clean. The main editor window shows the 'main.cpp' file with the following code:

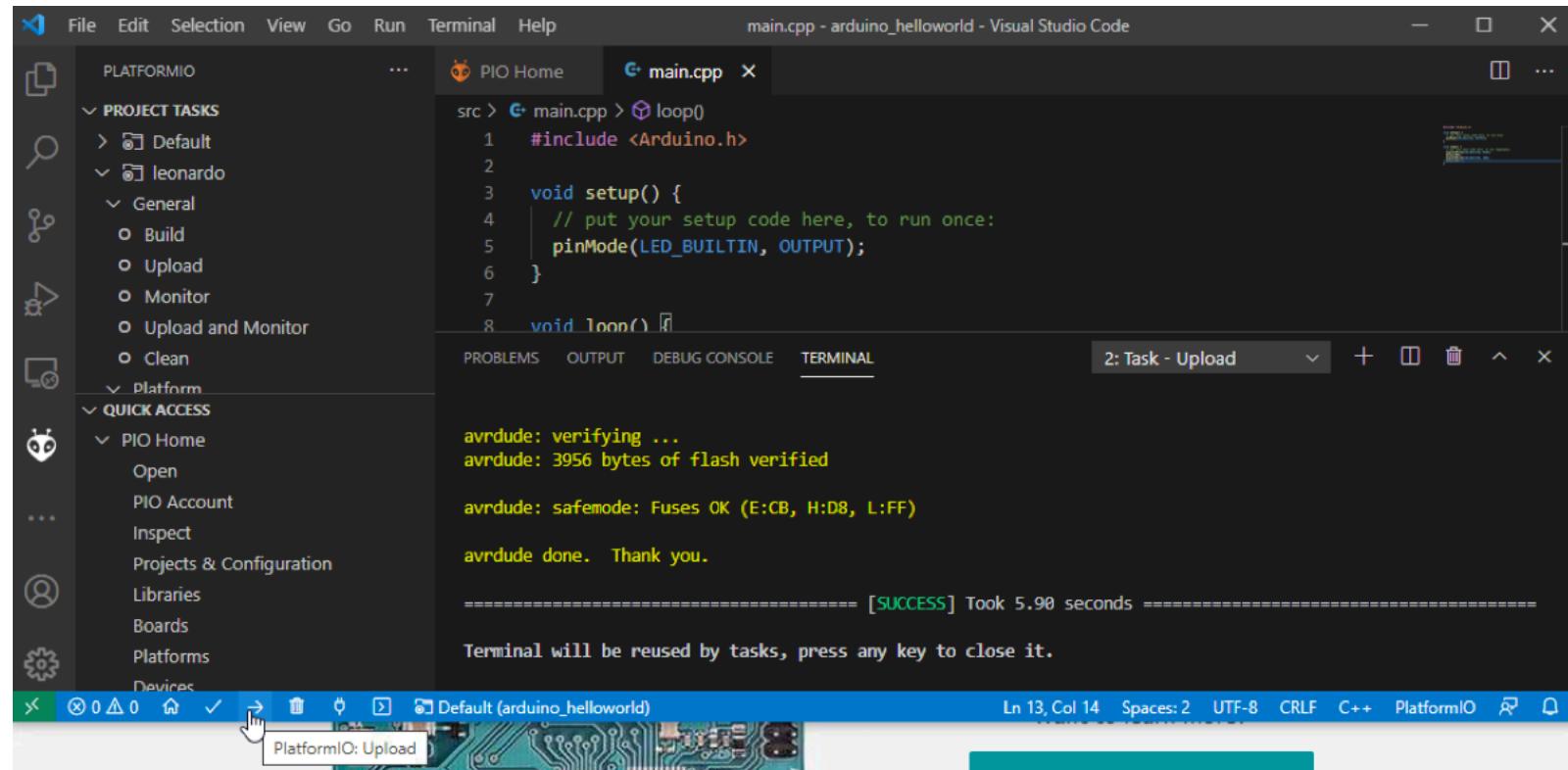
```
src > main.cpp > loop()
1 #include <Arduino.h>
2
3 void setup() {
4     // put your setup code here, to run once:
5     pinMode(LED_BUILTIN, OUTPUT);
6 }
7
8 void loop() {
```

The bottom terminal window shows the build process output:

```
Compiling .pio/build/leoardo/FrameworkArduino/wiring_shift.c.o
Archiving .pio/build/leoardo/libFrameworkArduino.a
Linking .pio/build/leoardo/firmware.elf
Building .pio/build/leoardo/firmware.hex
Checking size .pio/build/leoardo/firmware.elf
Advanced Memory Usage is available via "PlatformIO Home > Project Inspect"
RAM: [=      ] 5.8% (used 149 bytes from 2560 bytes)
Flash: [=     ] 13.8% (used 3956 bytes from 28672 bytes)
=====
[SUCCESS] Took 2.21 sec
```

The status bar at the bottom indicates the current file is 'Default (arduino_helloworld)' and provides status information: Ln 13, Col 14, Spaces: 2, UTF-8, CRLF, C++, PlatformIO.

Upload: загрузка в целевую плату



Вынос параметра в переменную

```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup() {
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
    digitalWrite(led, HIGH);      // turn the LED on (HIGH is the voltage level)
    delay(1000);                // wait for a second
    digitalWrite(led, LOW);       // turn the LED off by making the voltage LOW
    delay(1000);                // wait for a second
}
```

Вынос параметра в переменную

- Но лучше либо так

```
const int ledPin = 13
```

- Либо

```
#define PIN_LED 13
```

```
http://www.arduino.cc/en/Tutorial/BlinkWithoutDelay
*/
// constants won't change. Used here to
// set pin numbers:
const int ledPin = 13;      // the number of the LED pin

// Variables will change:
int ledState = LOW;          // ledState used to set the LED
long previousMillis = 0;      // will store last time LED was updated

// the following variables is a long because the time, measured in milliseconds,
// will quickly become a bigger number than can be stored in an int.
long interval = 1000;        // interval at which to blink (milliseconds)

void setup() {
  // set the digital pin as output:
  pinMode(ledPin, OUTPUT);
}

void loop()
{
  // here is where you'd put code that needs to be running all the time.

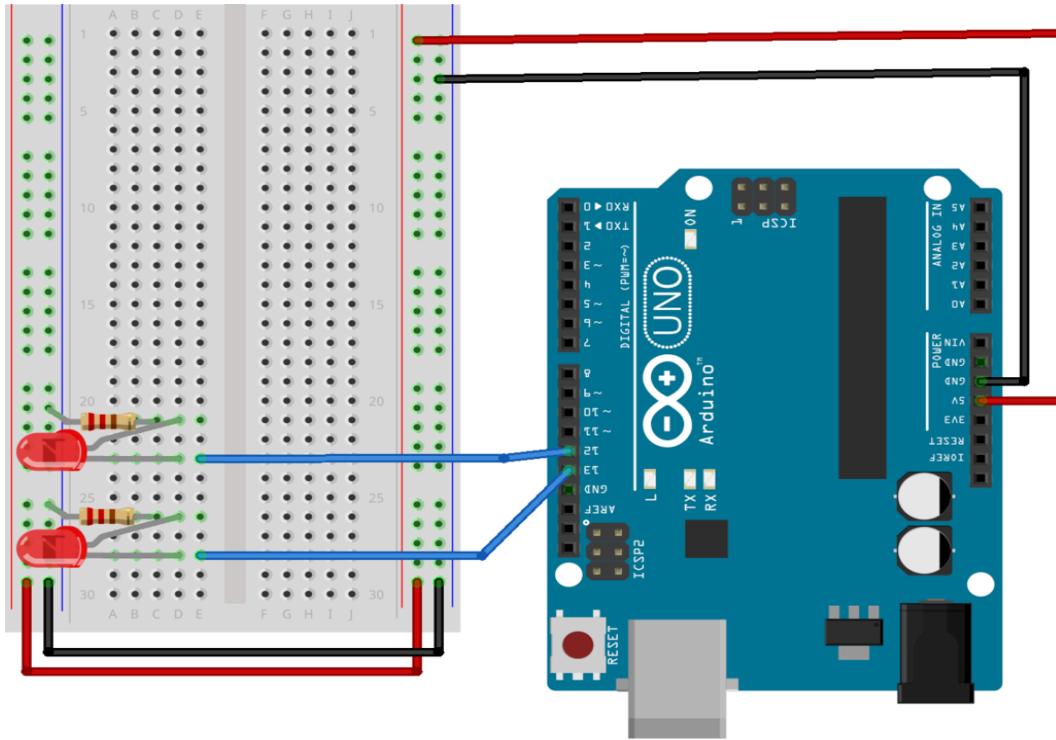
  // check to see if it's time to blink the LED; that is, if the
  // difference between the current time and last time you blinked
  // the LED is bigger than the interval at which you want to
  // blink the LED.
  unsigned long currentMillis = millis();

  if(currentMillis - previousMillis > interval) {
    // save the last time you blinked the LED
    previousMillis = currentMillis;

    // if the LED is off turn it on and vice-versa:
    if (ledState == LOW)
      ledState = HIGH;
    else
      ledState = LOW;

    // set the LED with the ledState of the variable:
    digitalWrite(ledPin, ledState);
  }
}
```

fritzing



```
class Flasher
{
    // Class Member Variables
    // These are initialized at startup
    int ledPin;      // the number of the LED pin
    long OnTime;     // milliseconds of on-time
    long OffTime;    // milliseconds of off-time

    // These maintain the current state
    int ledState;                // ledState used to set the LED
    unsigned long previousMillis; // will store last time LED was updated

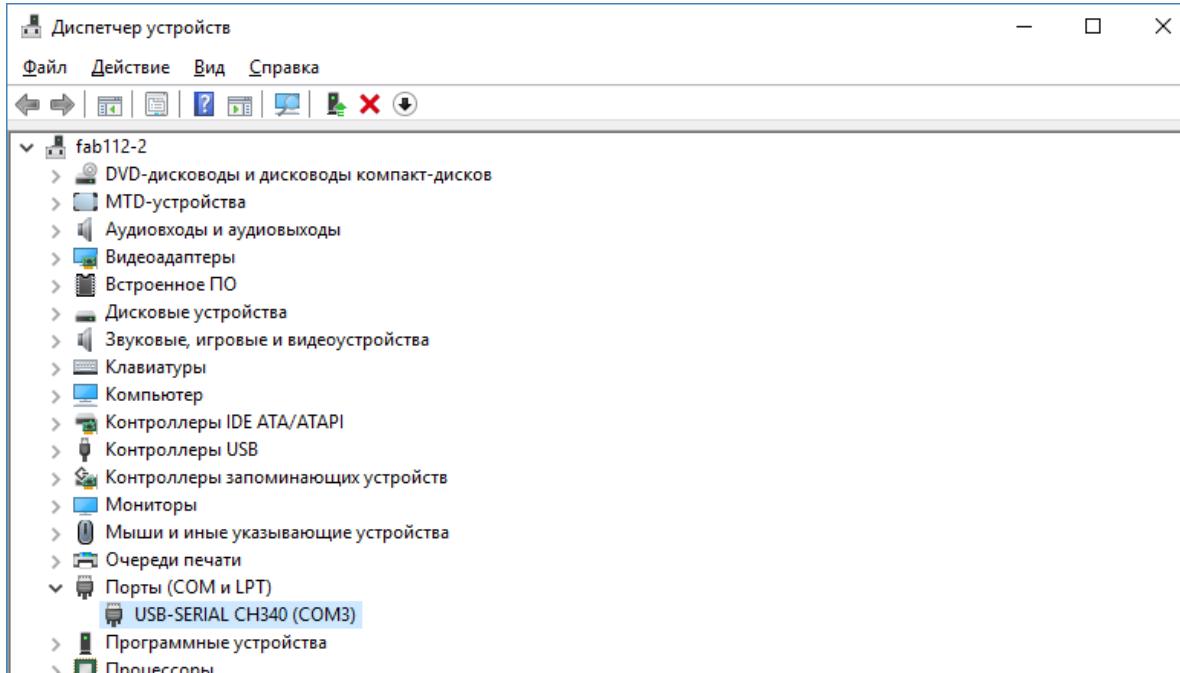
    // Constructor - creates a Flasher
    // and initializes the member variables and state
    public:
    Flasher(int pin, long on, long off)
    {
        ledPin = pin;
        pinMode(ledPin, OUTPUT);

        OnTime = on;
        OffTime = off;

        ledState = LOW;
        previousMillis = 0;
    }
};
```

Отладка: Serial

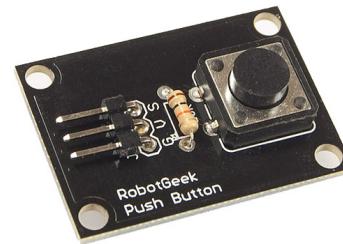
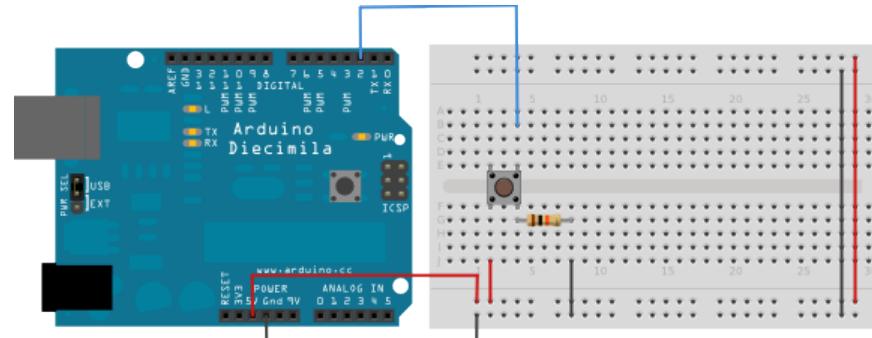
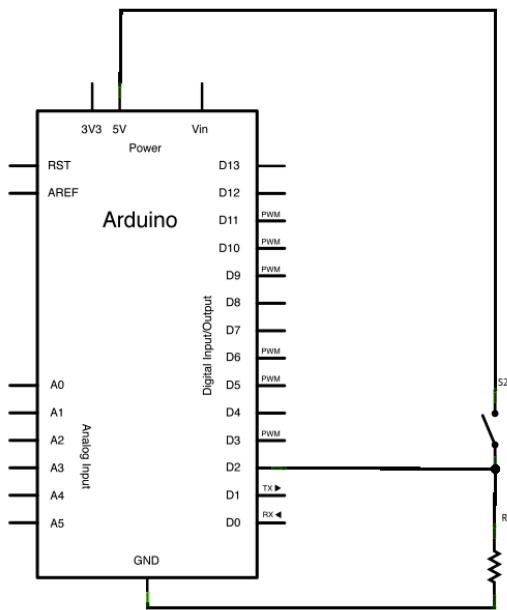
- При подключении к компьютеру Arduino выглядит как последовательный порт (Serial port)



Отладка: Serial

```
// digital pin 2 has a pushbutton attached to it. Give it a name:  
int pushButton = 2;  
  
// the setup routine runs once when you press reset:  
void setup() {  
    // initialize serial communication at 9600 bits per second:  
    Serial.begin(9600);  
    // make the pushbutton's pin an input:  
    pinMode(pushButton, INPUT);  
}  
  
// the loop routine runs over and over again forever:  
void loop() {  
    // read the input pin:  
    int buttonState = digitalRead(pushButton);  
    // print out the state of the button:  
    Serial.println(buttonState);  
    delay(1);          // delay in between reads for stability  
}
```

Схема с внешним событием

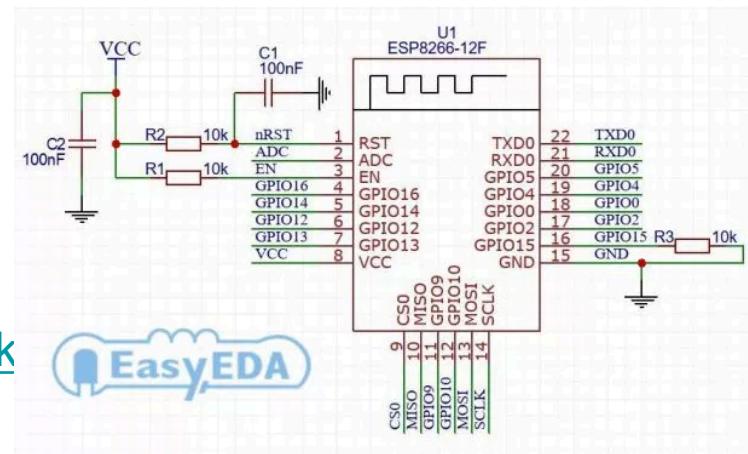


ССЫЛКИ

- <http://easyelectronics.ru/osnovy-na-palcax-chast-1.html>
- http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf
- <https://www.circuitar.com/nanoshields/modules/triac/>

ESP8266

- <https://esp8266.ru/>
- <http://www.instructables.com/id/Quick-Arduino-IDE/>
- <https://esp8266.ru/esp8266-nodemcu/>
- <https://esp8266.ru/category/esp8266/esp8266-for-beginners/>



Полезные ссылки

- http://www.chipnews.ru/html.cgi/arhiv/99_09/stat_2.htm - обзор Microchip, Atmel, Scenix от 2001 года
- http://www.chipnews.ru/html.cgi/arhiv/00_01/stat-3.htm - старые ST, NS, TI, Zilog
- <http://housea.ru/index.php/micro/14906> - старые 16-битные Hitachi, Mitsubishi, Motorola
- <http://playground.arduino.cc>