PRACTICE

### 1.Installing and configuring Arduino in Windows

### 2.Arduino connection and programming

### 3. Introduction to the Arduino UNO and the Arduino IDE development environment

### 4.Arduino. "Hello World!"the program on the arduino

### 5.Connecting analog sensors to Arduino

### 6.Connecting analog sensors to Arduino, reading sensor readings

### 7.Using the BMP180 pressure sensor

### 8. Development of a device for ECOIMPACT IoT based on an Arduino microcontroller

### 9. Connecting the device to the local ECOIMPACT server

### 10. Connecting to WiFi using the ESP8266 NodeMCU microcontroller

### 11. Development of IoT devices using Bluetooth LE

### 12.Lamp with push-button control

### 13. Lamp with push-button control

### 14. Orange Pi 2G

### 15. «Modeling. Creating a 3D model in Sketch Up»

**1. INSTALLING AND CONFIGURING ARDUINO IN WINDOWS**

Hardware part

The Arduino Board consists of an Atmel AVR microcontroller (ATmega328 and ATmega2560 in new versions and ATmega168 in old versions) and an element binding for programming and integration with other circuits. Each Board must have a 5 V linear voltage stabilizer and a 16 MHz quartz oscillator (in some versions, a ceramic resonator). The microcontroller is pre-flashed loader, so an external programmer is not needed.

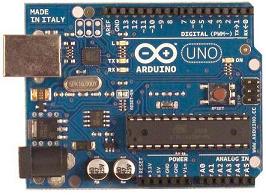
[](http://webstm32.sytes.net/user-files/arduino_uno_large.JPG)

Figure 8. [Arduino UNO R3](http://webstm32.sytes.net/user-files/cxuno.pdf)

At the conceptual level, all boards are programmed via RS-232 (serial connection), but the implementation of this method differs from version to version. The Serial Arduino Board contains a simple inverting circuit for converting RS-232 signal levels to TTL levels, and Vice versa. Current boards like Diecimila are programmed via USB, which is done thanks to a USB-to-serial Converter chip like FTDI FT232. In some variants, such as the Arduino Mini or unofficial Boarduino, programming requires the connection of a separate USB-to-serial Board or cable.

Arduino boards allow you to use most of the I/O pins of the microcontroller in external circuits. For example, the UNO Board has 14 digital inputs/outputs ("LOW" -0V and "HIGH" -5V), 6 of which can produce a PWM signal, and 6 analog inputs(0-5V). These pins are accessible at the top of the Board via 0.1-inch "Mama" type connectors. Several external expansion boards known as "shields" are available on the market.

Software

The Arduino integrated development environment is a cross-platform Java application that includes a code editor, compiler, and firmware transfer module to the Board.

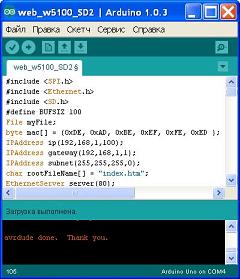
[](http://webstm32.sytes.net/user-files/aprog.jpg)

Figure 9. Arduino integrated development environment

The development environment is based on the Processing programming language and is designed for programming by beginners who are not familiar with software development. The programming language is similar to that used in the Wiring project. Strictly speaking, this is C/C++, supplemented by some libraries. Programs are processed using a preprocessor and then compiled using AVR-GCC.

## 1.  Installation Arduino IDE

First, you need to install the integrated development environment Arduino — Arduino IDE on your computer.

Installing the Arduino IDE using the installer will save you from most of the potential problems with the drivers and software environment.

**2. Running the Arduino IDE**

After you have downloaded and installed the Arduino IDE

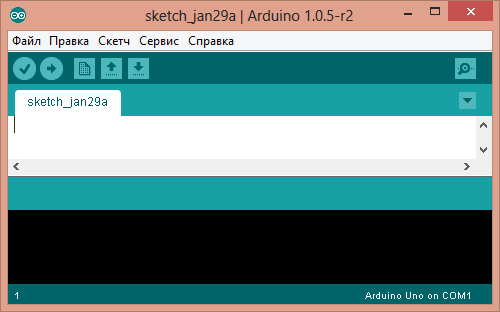
[](http://wiki.amperka.ru/_media/arduino-%D0%B1%D1%8B%D1%81%D1%82%D1%80%D1%8B%D0%B9-%D1%81%D1%82%D0%B0%D1%80%D1%82:%D0%BF%D1%80%D0%B5%D0%B4%D1%83%D0%BF%D1%80%D0%B5%D0%B6%D0%B4%D0%B5%D0%BD%D0%B8%D0%B5_com1.png)

Figure 10. Arduino IDE

Before us is the Arduino IDE window. Please note — we have not yet connected our Arduino Uno Board to the computer, and in the lower right corner there is already an inscription "Arduino Uno on COM1". So the Arduino IDE tells us that it is currently configured to work with the target Arduino Uno Board. And when the time comes, the Arduino IDE will look for the Arduino Uno on the COM1 port.

We will change these settings later.

**Did something go wrong?**

Arduino IDE does not start? The JRE (Java Runtime Environment) is probably not installed correctly on the computer. Refer to step (1) to reinstall the Arduino IDE: the installer will do all the work of deploying the JRE.

3. Connecting an Arduino to a computer

After installing the Arduino IDE, it's time to connect the Arduino Uno to your computer.

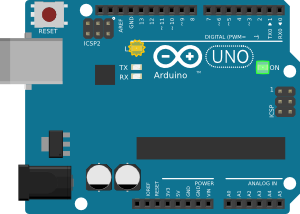
[](http://wiki.amperka.ru/_media/arduino-%D0%B1%D1%8B%D1%81%D1%82%D1%80%D1%8B%D0%B9-%D1%81%D1%82%D0%B0%D1%80%D1%82:arduino_leds.png)

Figure 11. Arduino Uno

Connect the Arduino Uno to the computer via a USB cable. You will see the "ON" led on the Board light up and the "L" led blink. This means that the Board is powered up, and the Arduino Uno microcontroller has started to run the factory-stitched program "Blink" (led flashing).

To configure the Arduino IDE to work with the Arduino Uno, we need to know what COM port number the Arduino Uno computer assigned. To do this, go to the" device Manager "Windows and open the tab" Ports (COM and LPT)". We should see the following picture:

This means that the operating system recognized our Arduino Uno Board as a COM port, selected the correct driver for it, and assigned this COM port number 7. If we connect another Arduino Board to the computer, the operating system will assign it a different number. Therefore, if you have multiple Arduino boards, it is very important not to get confused about the COM port numbers

**Configuring the Arduino IDE to work with the Arduino Uno**

Now we need to tell the Arduino IDE that the Board it is going to communicate with is on the COM port "COM7".

To do this, go to the menu "Tools" → "Serial port" and select the port "COM7". Now the Arduino IDE knows — something is on the "COM7" port. And with this "something" she will soon have to communicate.

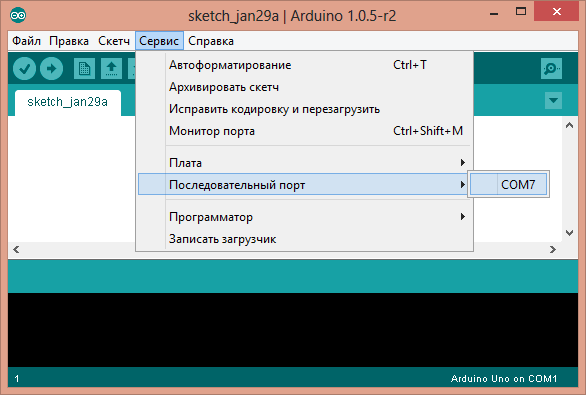
[](http://wiki.amperka.ru/_media/arduino-%D0%B1%D1%8B%D1%81%D1%82%D1%80%D1%8B%D0%B9-%D1%81%D1%82%D0%B0%D1%80%D1%82:%D0%B2%D1%8B%D0%B1%D0%BE%D1%80_com-%D0%BF%D0%BE%D1%80%D1%82%D0%B0.png)

Figure 12. The Tools Menu

To make sure that the Arduino IDE has no doubts, you must explicitly state: "We will use the Arduino Uno!". To do this, go to the menu "Tools" → "Board" and select our "Arduino Uno".

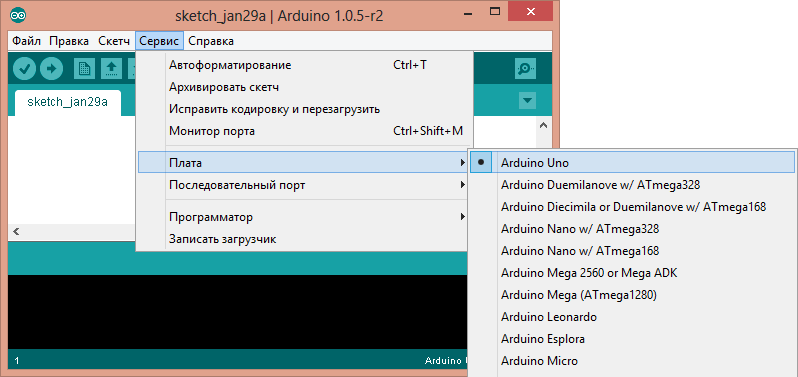
[](http://wiki.amperka.ru/_media/arduino-%D0%B1%D1%8B%D1%81%D1%82%D1%80%D1%8B%D0%B9-%D1%81%D1%82%D0%B0%D1%80%D1%82:%D0%B2%D1%8B%D0%B1%D0%BE%D1%80_%D0%BF%D0%BB%D0%B0%D1%82%D1%8B.png)

Figure 13. Service - > Fee

5. Loading the first sketch

The environment is configured, the Board is connected. Now you can proceed to loading the sketch.

The Arduino IDE contains a lot of ready-made examples in which you can quickly see the solution of a problem. There is a simple example of "Blink" in it. Let's choose him.

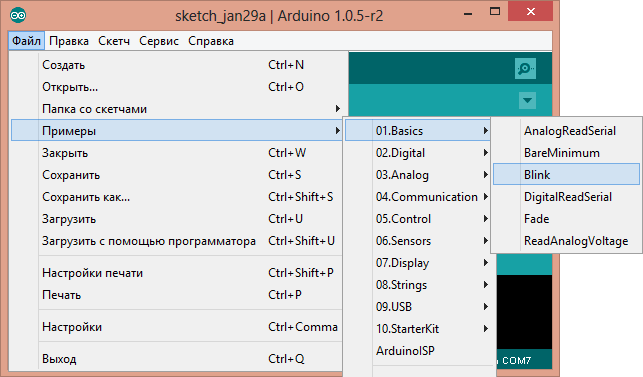
[](http://wiki.amperka.ru/_media/arduino-%D0%B1%D1%8B%D1%81%D1%82%D1%80%D1%8B%D0%B9-%D1%81%D1%82%D0%B0%D1%80%D1%82:%D0%B2%D1%8B%D0%B1%D0%B8%D1%80%D0%B0%D0%B5%D0%BC_%D0%BC%D0%B0%D1%8F%D1%87%D0%BE%D0%BA_.png)

Figure 14. File «Blink».

Slightly modify the code to see the difference with the factory led flashing.

Instead of a line:

Delay (1000);

write:

delay (100);

Full code version:

/\*

Flicker

Turns on the led for one second, then turns off for one second, several times.

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(100); // wait for a second

digitalWrite(led, LOW); // turn the LED off by making the voltage LOW

delay(100); // wait for a second

}

Now the "L" led should light up and go out for a tenth of a second. That is 10 times faster than the factory version.

Let’s upload our sketch to the Arduino Uno and check if it's true? After loading, the led will flash faster. It means that everything worked out. Now you can safely move on to "Experiments".

Arduino: Board selection, connection and first program

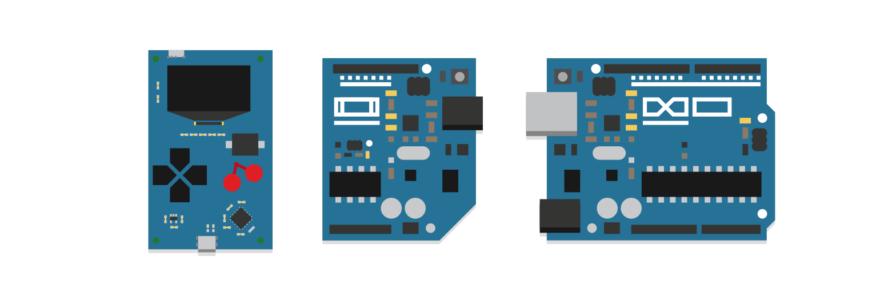


Figure 15. Arduino

Arduino is an open source electronic platform that allows you to interact with the world around you. Thanks to it, you can create everything that comes to mind-from simple electronic toys and automation of everyday life to the electronic filling of a combat robot for competitions, controlled by the power of thought.

What does Arduino consist of?

At the hardware level, this is a series of mounted boards, the brain of which is the AVR family of microcontrollers.

Boards have everything you need on Board for comfortable work, but their functionality is often not enough. To make your project more interactive, you can use various modules and extension boards that are compatible with the Arduino platform. This includes sensors (temperature, lighting, moisture, gas / smoke, atmospheric pressure), input devices (keyboards, joysticks, touch panels) and output devices (segment indicators, LCD / TFT displays, led matrices).

At the software level, the Arduino platform is a free development environment for the Arduino IDE. Microcontrollers should be programmed in C++, with some differences and easing created for quick adaptation of beginners. The development environment takes over the compilation of the program code and firmware of the microcontroller.

There is also s4a.cat -service based on Scratch, which allows you to more clearly conduct development on Arduino. It is suitable for teaching children, as well as if you just want to create a simple device without learning the Arduino programming language and various documentation. For other cases, it is better to stick to the traditional development process.

**Do you need to be able to solder?**

Knowledge in the field of electrical installation is welcome, but not mandatory. Simple Arduino-based devices are often made as a layout. For this purpose, a solderless breadboard is used, on which the modules are switched with the Arduino Board using jumpers.

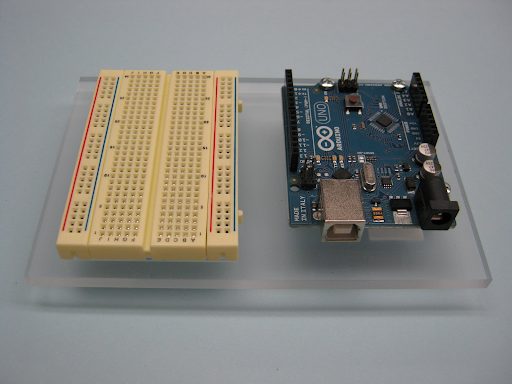


Figure 16.400-hole breadboard

There are also sets that include the Arduino Board (original or from a third-party manufacturer), the layout Board, jumpers and various radio elements, sensors, modules. For example, this:

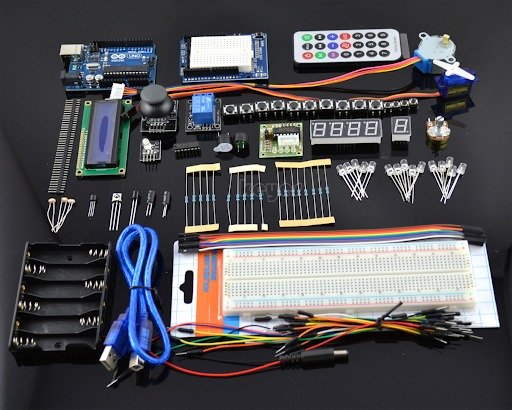


Figure 17. charges

By manufacturer

There are both official versions of Arduino boards and third-party boards. The original boards are of high quality product, but the price is also higher. They are made only in Italy and the United States, as evidenced by the inscription on the Board itself.

On the example of the most popular Board Arduino UNO:

Original Board. It is delivered only in a branded box, has the company logo, on the ports of the Board — marking. The price from the manufacturer is 20 €.

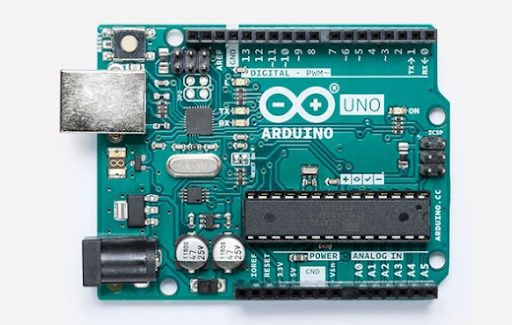


Figure 18. Original Board

Original Arduino UNO Board.

The fee from the third-party manufacturer. The quality is worse, but the price starts from 150 rubles. The quality of the Board may affect its performance in the future. Although this is rare, but the Board may not work "out of the box" - it all depends on the integrity of the manufacturer and the seller. To work with such boards, you need the CH340 driver, which is freely available. In all other respects, the development process is identical to the development process on the original boards.



Figure 19. Arduino UNO Board from a third-party manufacturer.

As intended

The UNO Board has enough ports for most projects. However, sometimes UNO capabilities may not be sufficient, and sometimes they may be redundant. For this reason, both original and third-party manufacturers produce a large number of boards that differ in the characteristics of the microcontroller, the number of ports and functionality.



Figure 20. Various Arduino boards.

The most popular of them:

\* Arduino Nano-the difference with UNO is only in the design. Nano is smaller.

\* Arduino Mega-Board based on a powerful microcontroller. It has a large number of ports.

\* Arduino Micro-has built-in support for USB connection, and therefore can be used as a HID device (keyboard, mouse, MIDI device).

\* Arduino Ethernet-has the ability to connect to the network via an Ethernet wire. The Board also has a slot for microSD cards.

\* Arduino Mini-slightly inferior to UNO in performance. The advantage of the Board is its miniature design.

• Arduino Due is a Board based on a 32-bit ARM microcontroller. Has a performance advantage over the rest.

\* Arduino LilyPad-the form factor allows the Board to be used in garments and textiles.

\* Arduino Yún — " it was necessary to put Linux...". It has support for the Linux distribution, built-in support for Ethernet and Wi-Fi, a slot for microSD. Like Micro, it has built-in support for USB connection.

**SOFTWARE installation**

After selecting the necessary Board, you need to install the free Arduino IDE development environment, which can be found on the official website, as well as, if necessary, the CH340 driver.

The Arduino Create cloud platform has recently opened, which covers most of the development stages (from idea to build). You do not need to install anything on your computer, the platform takes care of everything you need. In the first place — an online code editor.

Arduino Create has access to training materials and projects. You will be able to communicate with professionals and help beginners.



Figure 21. Arduino IDE development environment

**Features of programming on the Arduino platform**

**Terms**

Software code for Arduino is called sketches. Sketches have two main methods: setup() and loop (). The first method is automatically called after enabling / resetting the microcontroller. It initializes ports and various modules and systems. The loop() method is called in an infinite loop throughout the entire operation of the microcontroller.

Ports are an integral part of any microcontroller. Through them, the interaction of the microcontroller with external devices. On the software side, ports are called pins. Any pin can work in the input mode (for further reading the voltage from it) or in the output mode (for further setting the voltage on it).

Any pin works with two logical States: LOW and HIGH, which is equivalent to a logical zero and one, respectively. Some ports have a built-in ADC, which allows you to read the analog signal from the input (for example, the value of an alternating resistor). Also, some pins can work in PWM mode, which allows you to set the analog output voltage. Usually the pin functionality is indicated on the label of the Board itself.

Main function

For basic work with the Board, the Arduino library has the following functions:

\* pinMode (PIN, type) — specifies the purpose of a specific pin PIN (value type INPUT-input, OUTPUT-output);

\* digitalWrite(PIN, state) - sets the logical state of the PIN output (state LOW-0, HIGH-1);

\* digitalRead (PIN) - returns the logical state from the PIN input (LOW-0, HIGH-1);

\* analogWrite (PIN, state) - sets the analog voltage at the PIN (state) output between 0 and 255);

\* analogRead (PIN) - returns the value of the analog signal level from the PIN input (limits depend on the bit depth of the built-in ADC. Typically, the bit depth is 10 bits, so the return value is between 0 and 1023);

\* delay (ms) - pauses the sketch execution for a specified number of milliseconds;

\* millis () — returns the number of milliseconds after the start of the microcontroller.

Otherwise, the programming process in Arduino is the same as in standard C++.

Six

Writing the first program

Instead of all the usual Hello World's in Arduino decided to run a sketch Blink, which can be found in The file→Examples→01.Basics→Blink. There you can also find many other educational sketches on different topics.

Almost all boards have an led, the pin number of which is contained in the variable LED\_BUILTIN. It can be used for debugging purposes. In the next sketch, an example of controlling such an led will be considered.

Consider the blink sketch:

// This function starts when the microcontroller starts

void setup() {

// Assign pin output. The LED\_BUILTIN pin has a built-in led placed on the Board

pinMode(LED\_BUILTIN, OUTPUT);

}

// This function is called cyclically

void loop() {

digitalWrite(LED\_BUILTIN, HIGH); / / Turn on the led (supply it with logic 1-the supply voltage of the microcontroller)

delay(1000); / / Waiting for a second

digitalWrite(LED\_BUILTIN, LOW); / / Turn off the led (supply it with a logical 0-ground voltage)

delay(1000); / / Waiting for a second

}

**Insertion**

After writing, you need to "pour" the sketch on the microcontroller. As already mentioned, the Arduino platform takes over the entire process of firmware microcontroller - you only need to connect the Board to the computer.

Before flashing the microcontroller, you need to select your Board from the list in the IDE. This is done in the Tools→Payment tab. Most of the existing boards are already there, but if necessary, you can add others through the Board Manager.

After that, you need to connect the Arduino Board to any USB port on your computer and select the appropriate port in the Tools tab→Port.

Now you can start flashing the microcontroller. To do this, just click the Download button, or go to the Sketch→Download tab. After clicking, the code will be compiled, and if there are no compilation errors, the firmware of the microcontroller will start. If all the steps are completed correctly, the led on the Board will flash with a period and an interval of 1 second.

**Data exchange with a computer**

All Arduino boards have the ability to exchange information with the computer. The exchange takes place over a USB cable — no additional "buns" are required. We need a Serial class that contains all the necessary functions. Before working with the class, you must initialize the serial port and specify the data transfer rate (by default, it is 9600). To send text data, the well-known print() and println () methods exist in the Serial class. Consider the following sketch:

void setup() {

Serial.begin(9600); / / Initialize the serial port at 9600 baud

}

void loop() {

Serial.println ("T for Tproger"); / / Send a message on the serial port and translate to a new line

delay(1000); / / Waiting for a second

}

The Arduino IDE has a port Monitor. You can run it through The tools→port Monitor. After opening it, make sure that the Monitor is running at the same speed that you specified when initializing the serial port in the sketch. This can be done in the lower panel of the Monitor. If everything is correctly configured, a new line "T for Tproger" should appear every second in the Monitor. You can use data exchange with your computer to debug your device.

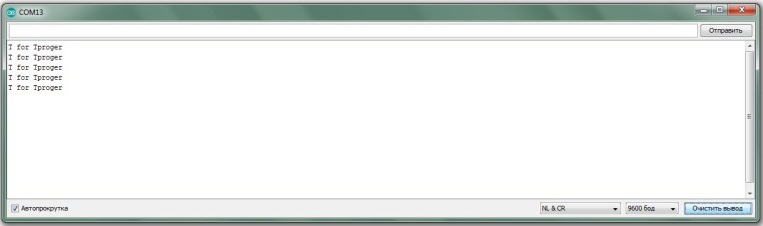


Figure 22. Tools→Monitor

Information on the computer side can not only be received, but also sent. To do this, consider the following sketch:

void setup() {

Serial.begin(9600); / / Initialize the serial port at 9600

pinMode(LED\_BUILTIN, OUTPUT); / / Initialize port with led as output

}

void loop() {

if (Serial.available () > 0) / / If the buffer has a byte to read, then...

switch (Serial.read ()) { // Read bytes from the buffer

case '1': digitalWrite(LED\_BUILTIN, HIGH); / / If the byte is '1' - turn on the led

break;

case '0': digitalWrite(LED\_BUILTIN, LOW); / / If the byte is '0' - turn off the led

break;

case 'T': Serial.println ("proger"); / / If the byte is equal to 'T' - send the text "proger" on the serial port"

}

}

Flash the microcontroller and return to the port Monitor. Enter 1 in the upper field and click Send. After that, the led on the Board should light up. Turn off the led by sending 0 from the Monitor. If we send the t character, we should get the string "proger" in response.

This way you can send information from the computer to the Arduino and back. Similarly, you can implement a connection between two Arduino.

Eight

And how to connect modules?

To work with sensors and modules, their manufacturers create special libraries. They serve for easy integration of modules into your system. You can connect the library using a zip file or using the Library Manager.

However, a large number of sensors are binary, i.e. you can read information from them with a simple function digitalRead ().

**2. ARDUINO CONNECTION AND PROGRAMMING**

Does learning about microcontrollers seem complicated and incomprehensible? Before the advent of Arudino-it was really not easy and required a certain set of programmers and other equipment.

[](http://electrik.info/img/arduino3.jpg)

Figure 23. Arduino

What is Arduino?

This is a kind of electronic constructor. The initial goal of the project is to allow people to easily learn how to program electronic devices, while devoting minimal time to the electronic part.

Assembly of complex circuits and connection of boards can be carried out without a soldering iron, and with the help of jumpers with detachable connections "dad" and "mom". Thus it can be connected as hinged elements, and expansion boards, which in the lexicon of arduinschikov called simply "Shields" (shield).

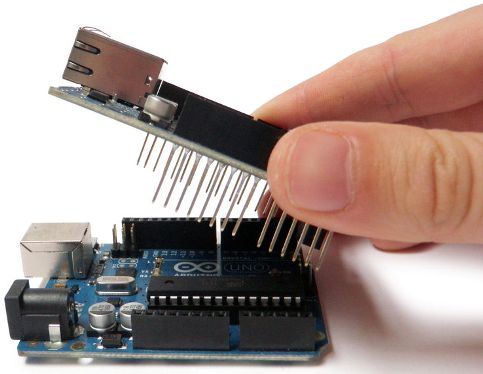


Figure 24. shield

What is the first Arduino Board to buy for a beginner?

The base and most popular Board is the Arduino Uno. This fee is the size of a credit card. Rather large. Most of the shields that are on sale are perfect for it. There are sockets on the Board for connecting external devices.

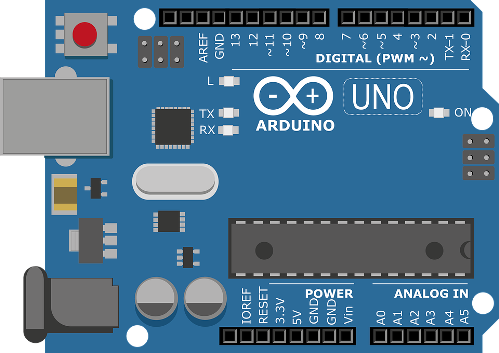


Figure 25. Atmega328.

In domestic stores in 2017, its price is about 4-5 dollars. On modern models, its heart is the Atmega328.

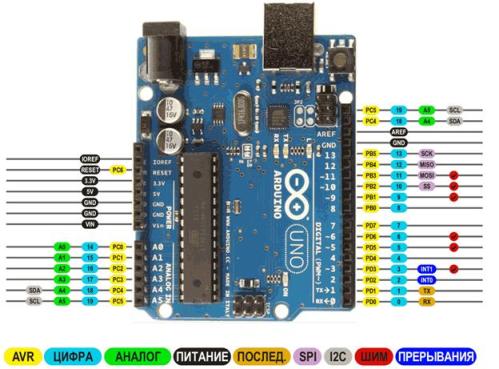
[](http://electrik.info/img/arduino1.gif)

Figure 26. Arduino UNO

The image of the Arduino Board and decoding the functions of each pin, Arduino UNO pinout. The microcontroller on this Board is a long chip in the dip28 case, which means that it has 28 legs.

The next most popular fee is almost two times cheaper than the previous one-2-3 dollars. This is an Arduino Nano Board. The current boards are built on the same Atmega328, they are functionally similar to UNO, the differences in size and the solution of matching with USB, more on this later. Another difference is that to connect to the Board devices provided plug, in the form of needles.

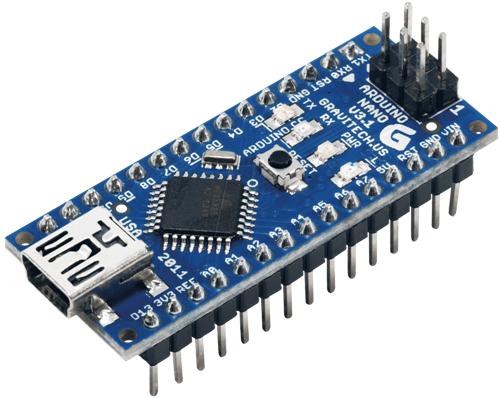


Figure 27. Plug

The number of pins (legs) of this Board is the same, but you can see that the microcontroller is made in a more compact case TQFP32, in the case added ADC6 and ADC7, the other two "extra" legs duplicate the power bus. Its size is quite compact-about the size of your thumb.

Aruino Nano pinout

The third most popular Board is the Arduino Pro Mini, it does not have a USB port for connecting to a computer, as I will tell you later.



Figure 28. Size comparison of Arduino Nano and Pro Mini

This is the smallest Board of all considered, otherwise it is similar to the previous two, and its heart is still Atmega328. We will not consider other boards, since this is an article for beginners, and the comparison of boards is the subject of a separate article.

Arduino Pro Mini pinout, in the upper part of the connection diagram USB-UART, pin "GRN" - divorced on the reset circuit of the microcontroller, can be called differently, for what it is necessary you will learn further.

**Overall results:**

If UNO is convenient for layout, then Nano and Pro Mini are convenient for final versions of your project, because they take up little space.

**How to connect an Arduino to a computer**

The Arduino Uno and Nano are connected to the computer via USB. In this case, there is no hardware support for the USB port, it uses a level conversion scheme, usually called USB-to-Serial or USB-UART (rs-232). At the same time, a special Arduino loader is stitched into the microcontroller, which allows you to flash on these buses.

The Arduino Uno implements this binding on a microcontroller with support for USB-ATmega16U2 (AT16U2). It turns out that an additional microcontroller on the Board is needed to flash the main microcontroller.

In the Arduino Nano, this is implemented by the FT232R chip, or its ch340 counterpart. This is not a microcontroller — it is a level Converter, this fact makes it easier to build an Arduino Nano from scratch with your own hands.

Usually the drivers are installed automatically when the Arduino Board is connected. However, when Chinese copy of the Arduino Nano was bought, the device was identified, but it did not work, on the Converter was pasted a round sticker with data about the release date, it is unknown whether this was done on purpose, but when it was peeled off, the marking CH340 was seen.

Before that, the author did not come across this and thought that all USB-UART converters are assembled on FT232. The author had to download the drivers, they are very easy to find by searching for "Arduino ch340 drivers". After a simple installation - everything worked!

The microcontroller can be powered through the same USB port, i.e. if you connect it to the adapter from a mobile phone-your system will work.

**3. INTRODUCTION TO THE ARDUINO UNO AND THE ARDUINO IDE DEVELOPMENT ENVIRONMENT**

task:

- Explore the Arduino Board;

- Learn the Arduino Programming environment;

- Demonstrate the work with the program Arduino On the example of laboratory work;

- Answer control questions;

- Issue a report.

Theoretical part

The image of the Arduino Board is Shown



Figure 29. ArduinoUNO

## General information

## The Arduino UNO Controller is built on the ATmega328. The platform has 14 digital inputs/outputs (6 of which can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connector, a power connector, an ICSP connector, and a reset button. To work, you must connect the platform to the computer via a USB cable, or supply power using an AC/DC adapter or battery.

## Unlike all previous boards that used the FTDI USB microcontroller for USB communication, the new Arduino Uno uses the ATmega8U2 microcontroller.

## Table 3. Characteristics

|  |  |
| --- | --- |
| Microcontroller | ATmega328 |
| Operating voltage | 5 В |
| Input voltage (recommended) | 7-12 В |
| Input voltage (limit) | 6-20 В |
| Digital Inputs/Outputs | 14 (6 of which can be used as PWM outputs) |
| Analog input | 6 |
| Direct current through input / output | 40 мА |
| Constant current for 3.3 V output | 50 мА |
| Flash memory | 32 KB (ATmega328) of which 0.5 KB is used for the loader |
| RAM | 2 Кб (ATmega328) |
| EEPROM | 1 Кб (ATmega328) |
| Clock frequency | 16 MHz |

**Arduino Uno circuit diagram**

The schematic diagram is presented in Appendix B

Food

The Arduino Uno can be powered via a USB connection or from an external power supply. The power supply is selected automatically.

External power (not USB) can be supplied via an AC/DC voltage Converter (power supply) or a rechargeable battery. The voltage Converter is connected via a 2.1 mm connector with a Central positive pole. The battery wires are connected to the GND and Vin terminals of the power connector.

The platform can operate with an external power supply from 6 V to 20 V. If the power supply voltage is lower than 7 V, the 5V output can output less than 5 V, and the platform can work unstable. When using a voltage higher than 12 V, the voltage regulator may overheat and damage the Board. Recommended range from 7 V to 12 V.

**Power outputs:**

\* VIN. The input is used to supply power from an external source (in the absence of a 5V USB connector or other regulated power supply). The supply voltage is supplied through this pin.

\* 5V. Adjustable voltage source used to power the microcontroller and components on the Board. Power can be supplied from the VIN output via a voltage regulator, or from a USB connector, or other regulated 5 V voltage source.

• 3V3. 3.3 V output voltage generated by the built-in controller on the Board. The maximum current consumption is 50 mA.

• GND. The conclusions of the ground.

Memory

The ATmega328 microcontroller has 32 KB of flash memory, of which 0.5 KB is used for bootloader storage, as well as 2 KB of RAM (SRAM) and 1 KB of EEPROM (which is read and written using the EEPROM library).

**Inputs and Outputs**

Each of the 14 digital UNO pins can be configured as an input or output using the pinMode(),digitalWrite(), and digitalread () functions . The pins work at a voltage of 5 V. Each pin has a load resistor (disabled by default) of 20-50 k Ohm and can pass up to 40 mA. Some conclusions have special functions:

\* Serial bus: 0 (RX) and 1 (TX). The pins are used to receive (RX) and transmit (TX) TTL data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL serial bus chip.

\* External interrupt: 2 and 3. These pins can be configured to call an interrupt either on the lower value, on the front or back edge, or when the value changes. For more information, see the description of theattachinterrupt () function.

\* PWM: 3, 5, 6, 9, 10, and 11.Any of the pins is provided with an 8-bit resolution using the analogwrite () function.

\* SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). By means of these pins, SPI communication is carried out, for which the SPI library is used.

\* LED: 13. Built-in led connected to digital output 13. If the value on the output has a high potential, the led is lit.

The Uno platform has 6 analog inputs (designated as A0 .. A5), each with a resolution of 10 bits (i.e. can take 1024 different values). Standard outputs have a measurement range of up to 5 V relative to the ground, but it is possible to change the upper limit by means of the AREF output and the analogReference () function. Some conclusions have additional functions:

\* I2C: 4 (SDA) and 5 (SCL).Through the pins, an I2C (TWI) connection is made, which is created using the Wire library.

An additional pair of pins on the Board:

• AREF.The reference voltage for the analog inputs. Used with the analogreference () function.

\* Reset. A low signal level on the output resets the microcontroller. It is usually used to connect the reset button on the expansion Board, which closes access to the button on the Arduino Board itself.

**Communication**

Several devices are installed on the Arduino Uno platform to communicate with a computer, other Arduino devices, or microcontrollers. The ATmega328 supports a UART TTL (5V) serial interface supplied with pins 0 (RX) and 1 (TX). The ATmega8U2 chip installed on the Board directs this interface via USB, programs on the computer side "communicate" with the Board via a virtual COM port. ATmega8U2 firmware uses standard USB COM drivers, no exterior drivers are required, but on Windows you will need to connect the file ArduinoUNO.inf. Arduino serial monitor allows you to send and receive text data when connected to the platform. The RX and TX LEDs on the platform will flash when transmitting data via the FTDI chip or USB connection (but not when using serial transmission via pins 0 and 1).

With the SoftwareSerial library, it is possible to create a serial data transfer through any of the UNO digital pins.

The ATmega328 supports I2C (TWI) and SPI interfaces. The Arduino includes the Wire library for easy use of the I2C bus.

**Programming**

The platform is programmed via Arduino SOFTWARE. From the Tools > Board menu, select "Arduino Uno" (according to the installed microcontroller). Detailed information can be found in the reference and instructions on the developer's website.

The ATmega328 microcontroller comes with a recorded bootloader, making it easy to write new programs without using external programmers. Communication is carried out by the original STK500 Protocol.

It is possible not to use the loader and program the microcontroller via ICSP pins (in-circuit programming).

Automatic (software) reboot

UNO is designed in such a way that before writing a new code, the reboot is performed by the Arduino program itself on the computer, and not by pressing a button on the platform. One of the DTR lines of the ATmega8U2 chip that control the data flow (DTR) is connected to the reset pin of the ATmega328 microcontroller via a 100 nF capacitor. Activating this line, i.e. giving a low-level signal, restarts the microcontroller. The Arduino program, using this function, loads the code with a single click of the Upload button in the programming environment itself. The low-level DTR signal is coordinated with the start of code writing, which reduces the bootloader timeout.

The function has another application. UNO reboots every time you connect to an Arduino program on a Mac X or Linux computer (via USB). The boot loader runs for the next half-second after the reboot. During programming, the first few bytes of code are delayed to prevent the platform from receiving incorrect data (all but the new program code). If you are debugging a sketch written to the platform on a one-time basis, or entering any other data at the first start, make sure that the program on the computer waits for a second before transmitting the data.

On Uno, it is possible to disable the automatic reload line by breaking the corresponding line. The chip contacts at both ends of the line can be connected for recovery purposes. The line is marked "RESET-EN". It is also possible to disable the automatic reset by connecting a 110 Ohm resistor between the 5 V source and this line.

Current protection of the usb connector

The Arduino UNO has A self-repairing fuse (automatic) that protects the computer's USB port from short-circuit and overcurrent currents. Although almost all computers have similar protection, however, this fuse provides an additional barrier. The fuse is triggered when a current of more than 500 mA passes through the USB port and opens the circuit until normal current values are restored.

Physical characteristic

The length and width of the Uno PCB are 6.9 and 5.3 cm, respectively. The USB connector and power connector go beyond these dimensions. Four holes in the Board allow you to fix it on the surface. The distance between digital pins 7 and 8 is 0.4 cm, although it is 0.25 cm between the other pins

**Arduino development environment**

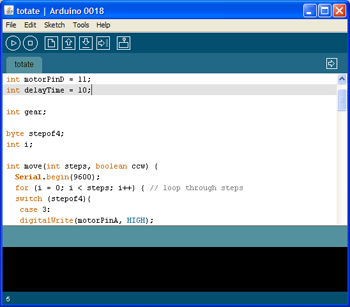


Figure 30. ArduinoIDE software editor window

The Arduino development environment consists of a built-in text editor for software code, a message area, a text output window(console), a toolbar with buttons for frequently used commands, and several menus. To download programs and communicate, the development environment is connected to the Arduino hardware.

A program written in the Arduino environment is called a sketch. The sketch is written in a text editor that has tools for cutting/inserting, searching / replacing text. When saving and exporting a project, explanations appear in the message area, and errors may also be displayed. The text output window (console) shows Arduino messages that include full error reports and other information.

Toolbar button

The toolbar buttons allow you to check and record the program, create, open and save a sketch, and open serial bus monitoring:

Verify/Compile program code checking for errors, compilation.

Stop stops monitoring the serial bus (Serial monitor) or dimming other buttons.

New creating a new sketch.

Open the access menu for all sketches in Notepad. Opens by clicking in the current window.

Note: due to an error in Java, this menu cannot be scrolled; if you need to open a sketch from this list, go to the File | Sketchbook menu.

Save save the sketch.

Upload to I/O Board Compiles the program code and uploads it to the Arduino device. The download description is given below.

Serial monitor Opens serial bus monitoring (Serial monitor).

Additional commands are grouped into five menus: File, Edit, Sketch, Tools, Help. The availability of the menu is determined by the work that is currently being performed.

Edit

\* Copy for discourse Copies to the clipboard suitable for posting on the forum sketch code with syntax highlighting.

\* Copy as HTML Copies the sketch code to the clipboard as HTML code to be placed on web pages.

**Sketch**

\* Verify / Compile check the sketch for errors.

\* Import library Adds the library to the current sketch by inserting the #include Directive into the sketch code. For more information, see the library descriptions below (Libraries).

\* Show Sketch folder Opens the folder containing the sketch file on the desktop.

\* Add File...Adds a file to the sketch (the file will be copied from the current location). The new file appears in a new tab in the sketch window. The file can be deleted from the sketch using the bookmarks menu.

Tools

\* Auto Format this option optimizes the code, for example, lines up the opening and closing brackets vertically and places a statement between them.

\* Board select the platform used. The list with the description of the platforms is provided below.

\* Serial Port the Menu contains a list of serial data transfer devices (real and virtual) on the computer. The list is updated automatically every time you open the Tools menu.

\* Burn Boot loader points in this menu allow you to write the Bootloader to the microcontroller on the Arduino platform. This action is not required in the current work with Arduino, but it is useful if you have a new ATmega (without the loader). Before recording, we recommend that you check the correct platform selection from the menu. When using an AVR ISP, you must select the appropriate port for the programmer from the Serial Port menu.

**Serial bus monitoring (Serial Monitor)**

Displays data sent to the Arduino platform (USB card or serial bus card). To send data, enter the text and press the Send or Enter button. Then select the transfer rate from the drop-down list corresponding to the value Serial.begin in the sketch. On Mac or Linux, the Arduino platform will restart (the sketch will start over) when the serial bus monitoring is connected.

It is possible to exchange information with the platform via Processing, Flash, MaxMSP, etc. (see the interface descriptions page for details).

**Settings**

Some settings are changed in the Preferences window(the Arduino menu on Mac or File on Windows and Linux). The rest of the settings are located in the file whose location is specified in the Preferences window.

The basics of Arduino programming

**Syntax**

1 . Each expression ends with a character; a semicolon. For example: a = b+c;

2. The body of functions and compound operators (if, else, for, while) is separated by curly brackets (similar to BeginEnd in Pascal). For example: if (a>0) { b = a+1; }

3. Strings are separated by regular double quotes." Example: println(“some text”);

4. Symbols are separated by single quotes: symbol = ‘a’;

5. Libraries are connected using the construction: #include <math.h>

6. Commentaries in the program start with the characters / / two straight slashes. Example: //this is my program

Data type

Variable Declaration in c++ is performed using a construction of the form: variable type variable name; Example: int x, y; / / two variables x and y are declared, having an integer type 1 . Integers byte from 0 to 255 int from 32 768 to 32 767 word from 0 to 65535 long from 2 1 47 483 648 to 2 1 47 483 647 2. Fractional float numbers from 3.4028235 E+38 to 3.4028235 E+38 double are equivalent to float in the current version of Arduino 3. Arrays Arrays in c++ are defined by a construction of the type: element type\_name [size]; Example: int numbers[10]; / / specifies an array of ten integers 4. Strings and char characters; Strings in c++ are arrays with elements of the char type. Example: char my\_str[10]; / / string of ten characters 5. Other types void is an empty type; boolean false is true (false or true).

**Operating personnel**

1 . Comparison operators = = equality != inequality < less than <= less than or equal to > more >= greater than or equal to

2. If(a>0) { ... commands executed if the condition is true }e lse { ... commands executed otherwise }

3. Cycles for (k=0; k<3; k=k+1) { ... commands executed at each step of the cycle } in parentheses, the initial value of the iterator k=0; the condition for continuing the cycle k<3 (as long as the iterator is less than three); the action on the iterator during each step k=k+1 (increase by one at each step).

**Functions**

function\_type function\_name (arguments) { commands executed within the function return function result\_function;} function\_type return value type. For example, the standard sin function has a float return type. function name

**4. ARDUINO. "HELLO WORLD!" THE PROGRAM ON THE ARDUINO**

For the first lesson, we do not need anything except the Arduino itself and the usb cable to it.

How banal it will sound, but let's say with the help of Arduino " Hello World!". With this simple experiment, we will take our first step into the world of Arduino.

For the lesson we will need :

1. Arduino uno controller

2. USB cable (in most cases, it comes with the controller)

First, we need to connect the Arduino to the computer, and then write the program code in this environment "Hello World!". Copy it and paste it into the previously installed IDE.

int val; / / Set the variable val to track the keystroke

int ledpin = 13; / / set the digital input/output interface 13 - this is our led

void setup ()

{

Serial.begin (9600); / / Set the com port exchange rate to 9600

pinMode (ledpin, OUTPUT); / / Set ledpin = 13 as the output interface

}

void loop ()

{

val = Serial.read (); / / Read the command sent from the computer via the Arduino IDE console

if (val == 'R') / / Set the condition letter to the letter "R", when you press it in the console will light up the led and the line " Hello World!"

{

digitalWrite (ledpin, HIGH); / / Turn on the led on the 13 output of the Board

delay (500);

digitalWrite (ledpin, LOW); / / Turn off the led on the 13 output of the Board

Serial.println ("Hello World!"); / / Write to the console " Hello World!"

}

}

Or you can pour it into the Arduino directly from the browser from the below suggested window. In this case, you must first install the drivers for your Board.

int val; / / Set the variable val to track the keystroke

int ledpin = 13; / / set the digital input/output interface 13 - this is our led

void setup ()

{

Serial.begin (9600); / / Set the com port exchange rate to 9600

pinMode (ledpin, OUTPUT); / / Set ledpin 13 as the output interface

}

void loop ()

{

val = Serial.read (); / / Read the command sent from the computer via the Arduino IDE console

if (val == 'R') / / Set the condition letter to the letter "R", when you press it in the console will light up the led and the line " Hello World!"

{

digitalWrite (ledpin, HIGH); / / Turn on the led on the 13 output of the Board

delay (2000);

digitalWrite (ledpin, LOW); / / Turn off the led on the 13 output of the Board

Serial.println ("Hello World!"); / / Write to the console " Hello World!"

}

}

The data code waits for a signal in the form of the letter "R" to be sent to the virtual com port of the arduino, then the built-in led on the Board (output number 13 of the Board) is lit for 2000ms (2 seconds), and then in the monitor of the Arduino port UNO returns us the inscription " Hello World!"

Let's analyze the program code.

In the first two lines, our variables val and ledpin are set, and the second variable is immediately assigned the value 13 - this is the number of the output on which the built-in led on the Board is located.

Next, in the mandatory void setup () procedure, use the Serial command.begin (9600) we set the speed of data exchange with our Board. While we do not plan to transfer large amounts of data, so we will set a small speed of 9600, which will have a positive impact on the stability of data exchange with the computer. The pinMode (ledpin, OUTPUT) command tells the microcontroller that pin 13 (ledpin variable) is intended for information output, in our case we will turn on and off the voltage on the led.

The main code of the program is executed in the mandatory void loop () procedure, it executes the code inside it in a circle to infinity.

Read data from the com port-val = Serial.read ();

If the com port has the letter R - if (val == 'R')

that

Turn on the led on the 13 output of the Board-digitalWrite (ledpin, HIGH);

for 2 seconds-delay (2000);

Turn off the led on the 13 output of the Board-digitalWrite (ledpin, LOW);

Writing in com port Hello World! - Serial.println ("Hello World!");

**5. CONNECTING ANALOG SENSORS TO ARDUINO**

There are a huge number of sensors available for the Arduino platform. This article will focus on analog sensors. They are so called because the signal at their output is measured by the value of the voltage. Arduino converts analog signal values to digital using the built-in ADC (Analog-to-digital Converter) with a 10-bit capacity (most boards). The number of possible voltage values in this case will be equal to 1024: from 0 (0 V) to 1023 (5 V). The minimum possible recorded voltage change will be 5V/1024 = 0.004883 V (4.883 mV).The bit depth is not always 10 bits, for example, the Arduino Due ADC is 12-bit (4096 values).

Usually these sensors have the following outputs:

– G (GND).Minus the supply voltage.

– V (VTG or +5V, VCC).Plus the supply voltage.

– S (AO (Analog Output), SIG, OUT).Analog output.

Connecting to Arduino

G connects to ground (GND pin). V is connected to the +5V or 3V3 pin (read the sensor description). S is connected to a software-defined analog pin (not all Arduino pins can receive an analog signal), and the status of this pin is read in the Arduino IDE using the analogRead () function.

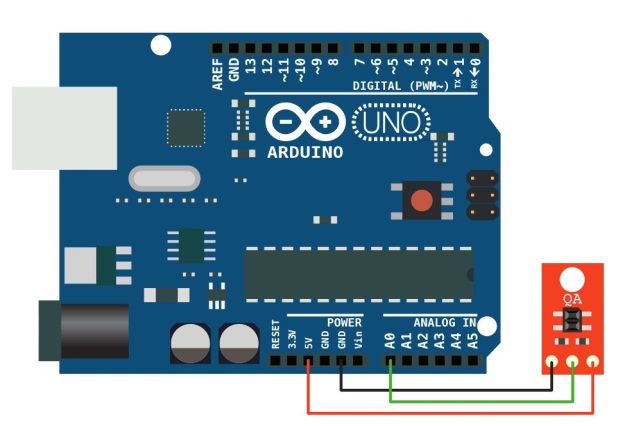
[](https://voltiq.ru/wp-content/uploads/connect-digital-sensors-to-arduino-1.jpg)

Figure 31. Arduino

Sample program code for reading sensor readings. The sensor readings will be output to the computer's serial port. You can view them using the port monitor in the Arduino IDE.

Connecting analog sensors to ArduinoArduino

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | const int analogPin = 0;    void setup() {      Serial.begin(9600);      Serial.println("Analog Sensor Test");      Serial.println();  }    void loop() {      int analogValue = analogRead(analogPin    Serial.println(analogValue);    delay(500);    } |

In the program, you can control any parameters when changing the state of the sensors. So now let's give a more obvious example. When a value less than 500 is received from the sensor, the Arduino Board's built-in led (pin 13) will light up.

Connecting analog sensors to ArduinoArduino

|  |
| --- |
| const int analogPin = 0;   void setup() {   }   void loop() {      int analogValue = analogRead(analogPin);      if (analogValue < 500)    {       digitalWrite(13, HIGH);    }      else   {       digitalWrite(13, LOW);    }     delay(100);   } |

*Please note! Sensors have different output voltage ranges. When making programs, you need to take this into account. The highest value is not always 1023.*

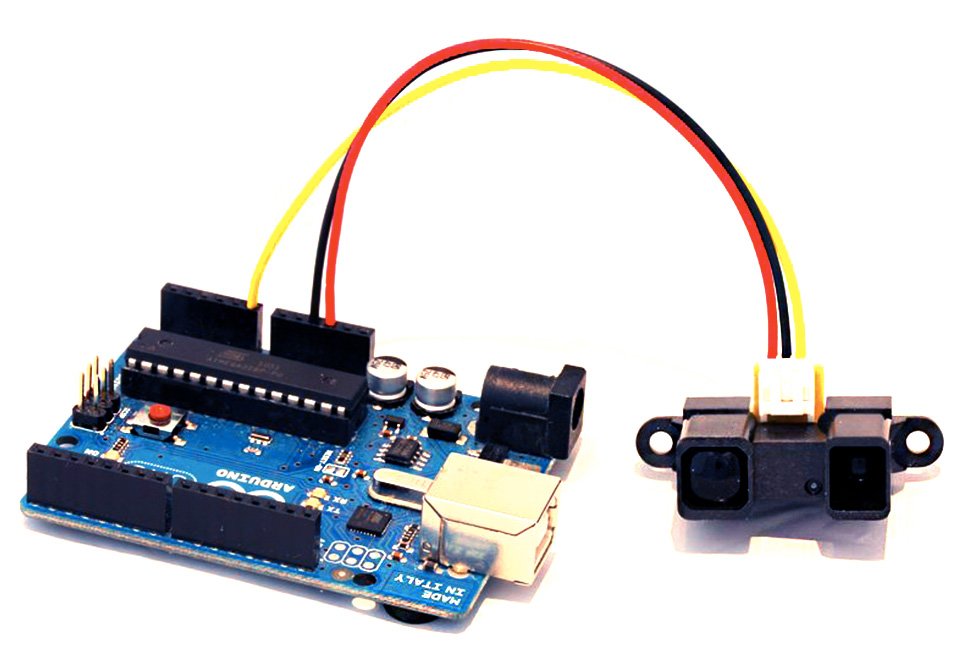


Figure 32. combined sensors

There are also combined sensors. They have both digital and analog outputs. When a certain value is reached on the analog output, the signal on the digital output reverses the value. These are, for example, volume sensors.

**6. CONNECTING ANALOG SENSORS TO ARDUINO, READING SENSOR READINGS**

Sensors are used to measure quantities, environmental conditions, and reactions to changes in States and positions. At their output, there may be both digital signals consisting of ones and zeros, and analog signals consisting of an infinite set of voltages in a certain interval.

Both digital and analog inputs of the microcontroller can be used to read digital values, in our case on the Arduino Board. Analog sensors must be connected via an analog-to-digital Converter (ADC). ATMEGA328, which is installed in most ARDUINO boards, contains a built-in ADC in its circuit. You can choose from as many as 6 analog inputs.

If this is not enough, you can use an additional external ADC to connect to the digital inputs, but this will complicate the code and increase its volume, due to the addition of processing algorithms and ADC control. The topic of analog-to-digital converters is wide enough that you can make a separate article or cycle about them. It is easier to use a Board with a large number of them or multiplexers. Let’s look at how to connect analog sensors to an Arduino.

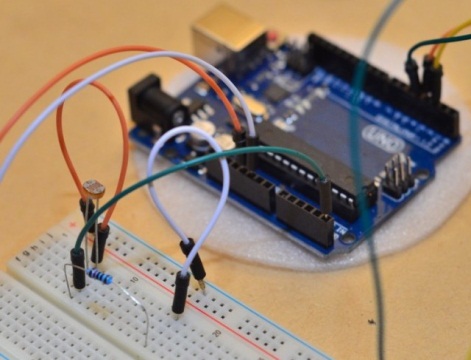


Figure 33. Analog sensor

General scheme of analog sensors and their connection

The sensor can even be an ordinary potentiometer. In fact – it is a resistive position sensor, which is used to control the level of liquids, the angle of inclination, and the opening of something. It can be connected to an Arduino in two ways.

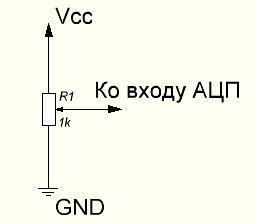


Figure 34. Diagram of analog sensors and their connection

The circuit above will allow you to read values from 0 to 1023, due to the fact that all the voltage falls on the potentiometer. Here the principle of the voltage divider works, in any position of the engine, the voltage is distributed over the surface of the resistive layer linearly or on a logarithmic scale (depends on the potentiometer), the part of the voltage that remains between the output of the slider (sliding contact) and the ground (gnd) falls on the input. On the breadboard this connection looks like this:

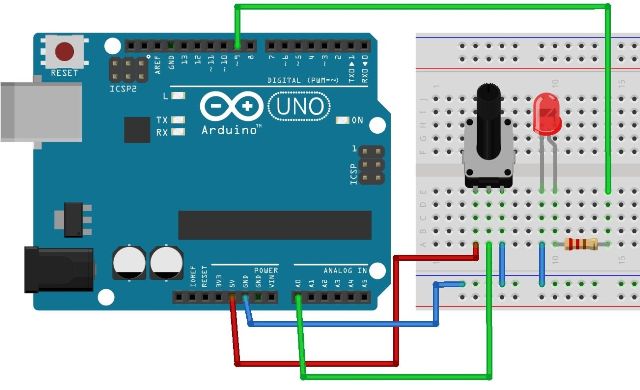


Figure 35. Connection options

The second option is connected according to the scheme of the classical resistive divider, here the voltage at the point of maximum resistance of the potentiometer depends on the resistance of the upper resistor.

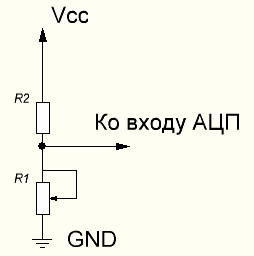


Figure 36. Connection

In General, the resistive divider is very important not only in the field of working with microcontrollers, but also in electronics in General. Below you can see the General scheme, as well as the calculated ratios for determining the value of the stress on the lower arm.

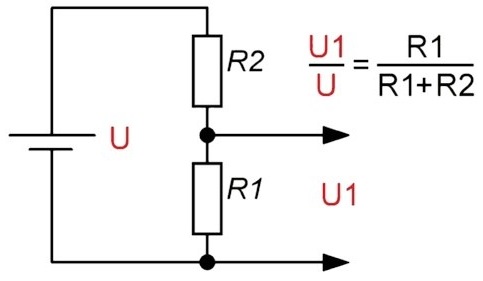


Figure 37. Connection

This connection is typical not only for the potentiometer, but for all analog sensors, because most of them work on the principle of changing the resistance (conductivity) under the influence of external sources-temperature, light, radiation of various kinds, etc.

Below is the simplest connection diagram of a thermistor, in principle, on its basis you can make a thermometer. But the accuracy of its readings will depend on the accuracy of the table of conversion of resistance to temperature, the stability of the power supply and the coefficients of change in resistance (including the upper arm resistor) under the same temperature. This can be minimized by selecting the optimal resistances, their power and operating currents.

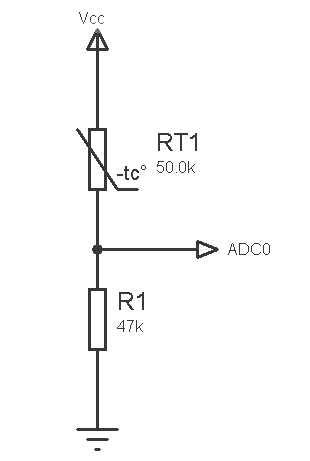


Figure 38. The simplest connection diagram of a thermistor

In the same way, you can connect photodiodes, phototransistors as a light sensor. Photo electronics has found applications in sensors that determine the distance and presence of an object, one of these we will consider later.

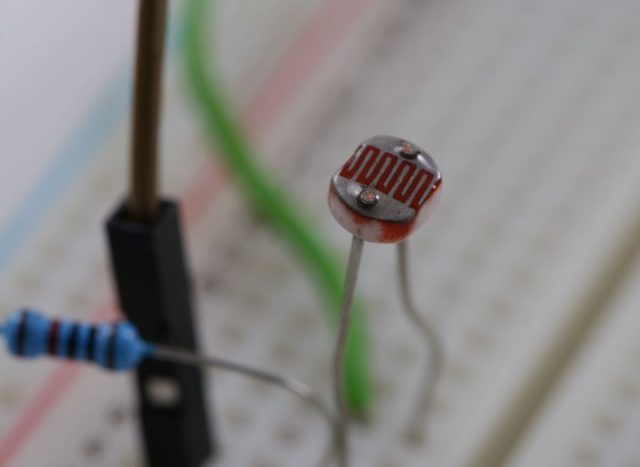


Figure 39. to connect photodiodes

The picture shows the connection of the photoresistor to the Arduino.

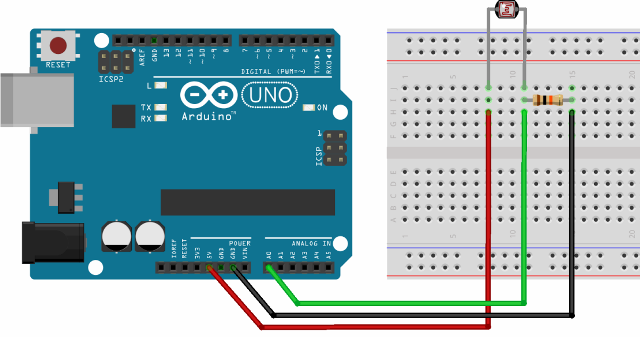


Figure 40. Connecting a photoresistor to an Arduino

**Program part**

All analog signals are read from the same ports using the analogRead () command. It is worth noting that the Arduino UNO and other models on the 168 and 328 ATmega 10-bit ADC. This means that the microcontroller sees the input signal as a number from 0 to 1023-a total of 1024 values. If we consider that the supply voltage is 5 volts, the input sensitivity:

5/1024=0.0048 V or 4.8 mV

That is, with a value of 0 at the input, the voltage is 0, and with a value of 10 at the input-48 mV.

In some cases, to convert the values to the desired level (for example, for transmission to the PWM output), 1024 is divided by a number, and as a result of division, the necessary maximum should be obtained. More clearly, the map function works (source, LF, HF, VHF, VHF), where:

• bass – the lower number to the transformation function;

\* RF-upper;

\* VHF-lower number after processing by the function (output);

\* VHF-upper

PP Practical application for converting a function of the input value for transmission to PWM (maximum value 255, for converting data from the ADC to the PWM output 1024 divided by 4):

Option 1 – the division.

int x;

x = analogRead(pot) / 4;

// a number from 0 to 1023 will be received

// divide it by 4, you get an integer from 0 to 255 analogWrite (led, x);

Option 2-the MAP function-opens up more possibilities, but more on that later.

void loop()

{ int val = analogRead(0);

val = map(val, 0, 1023, 0, 255);

analogWrite(led, val); }

Or even shorter:

analogWrite (led, map(val, 0, 1023, 0, 255))

Not all sensors have 5 Volts at the output, i.e. the number 1024 is not always convenient to divide to get the same 256 for the PWM (or any other). It can be both 2 and 2.5 volts and other values when the maximum of the signal is, for example, 500.

Popular analog sensors

The General view of the sensor for Arduino and its connection is shown below:

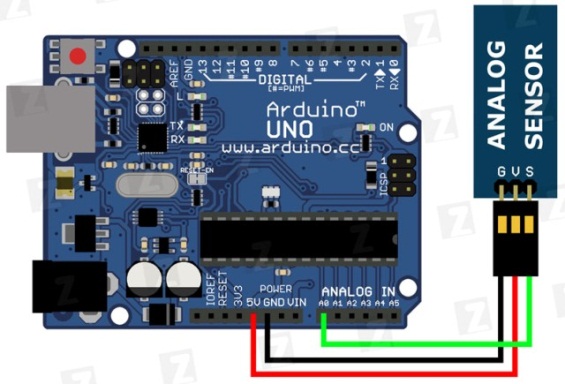


Figure 41. General view of the sensor for Arduino and its connection.

Usually there are three outputs, there may be a fourth – digital, but these are features.

Decoding of analog sensor pin designations:

\* G-minus power, common bus, ground. Can be denoted as GND, «-»;

\* V-plus power supply. Can be designated as Vcc, Vtg, «+»;

\* S-output signal, possible symbols-Out, SGN, Vout, sign.

Beginners for the development of reading the value of sensors choose projects of various thermometers. These sensors are available in digital versions, such as DS18B20, and analog-all kinds of chips such as LM35, TMP35, TMP36 and others. Here is an example of a modular design of such a sensor on the Board.

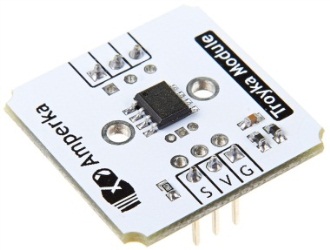


Figure 42. Chips

The accuracy of the sensor from 0.5 to 2 degrees. Built on the TMP36 chip, like many of its analogues, its output values are equal to 10 mV / °C. At 0°, the output signal is 0 V, and then added to 10 mV per 1 degree. That is, at 25.5 degrees, the voltage is 0.255 V, there may be a deviation within the error and the proper heating of the IC crystal (up to 0.1°C)

However, for the quality of thermometer you cannot simply read the values and display them on LCD indicator, or serial port for communication with PC, for the stability of the output signal of the whole system we need to average the values from the sensors, both analog and digital to a certain extent, while not degrading their performance and accuracy (there is a limit). This is due to the presence of noise, interference, unstable contacts (for resistive sensors based on the potentiometer, see the fault of the water level sensor or fuel tank in the car).

Codes for working with most sensors are quite voluminous, so I will not give them all, they are easy to find in the network on the request "name sensor + Arduino".

The next sensor that is often used by Arduino robot builders is the line sensor. It is based on photoelectronic devices, such as phototransistors.



Figure 43. The sensor line

With their help, the robot that moves along the line (used in automated production for the delivery of parts) determines the presence of a white or black stripe. In the right part of the picture you can see two devices similar to LEDs. One of them is an led that can emit in the invisible spectrum, and the second is a phototransistor.

Light is reflected from a surface if it is dark-the phototransistor does not receive the reflected stream, and if light receives and it opens. The algorithms that you put in the microcontroller process the signal and determine the correctness and direction of movement and correct them. Similarly, arranged and optical mouse, which you probably hold in your hand reading these lines.

An adjacent sensor will be added -a distance sensor from the company Sharp, also used in robotics, as well as in the conditions of monitoring the position of objects in space (with the corresponding TX error).



Figure 44. Distance sensor

Works on the same principle. Libraries and examples of sketches and projects with them in large numbers are on sites dedicated to Arduino.

**Conclusion**

The use of analog sensors is very simple, and with the easy-to-learn Arduino programming language, you will quickly learn simple devices. This approach has significant disadvantages in comparison with digital counterparts. This is due to a large spread of parameters, which causes problems when replacing the sensor. You may need to edit the source code of the program.

However, some analog devices are composed of reference voltage sources and current stabilizers, which has a positive effect on the final product and repeatability of devices in mass production. All problems can be avoided by using digital devices.

Digital circuitry as such reduces the need for tuning and adjusting the circuit after Assembly. This gives you the opportunity to build several identical devices on the same source code, the parts of which will give the same signals, with resistive sensors this is rare.

**7. USING THE BMP180 PRESSURE SENSOR**

Learn how to use the BMP180 pressure sensor with an Arduino Board

1. Install libraries to work with the BMP180 sensor

2. Assemble the installation and develop a sketch for Arduino to read the BMP180 sensor

Tools for performing work

1. Computer with Internet connection

2. Arduino Board with USB interface (for example, Arduino Uno)

Theoretical part

BMP180-combined sensor for measuring atmospheric pressure and temperature. The measured pressure range is 300-1100hPa with an accuracy of 1hPa, the temperature is 0-65 degrees Celsius with an accuracy of 0.1 degrees. The sensor is connected to the Arduino via the I2C bus. depending on the manufacturer, there are several ready-made BMP180 sensor modules that differ in the number of pins and power supply voltage. This work uses the Gy-68 module with 4 pins and a supply voltage of 3.3 volts.

**Execution of work**

Installing the library for working with the BMP180 sensor

Create a new sketch and save it with the name BMP180Test. Check whether the Board and port are selected correctly. Download the library to work with the sensor BMP180 "Adafruit BMP085 Unified" through the library Manager.

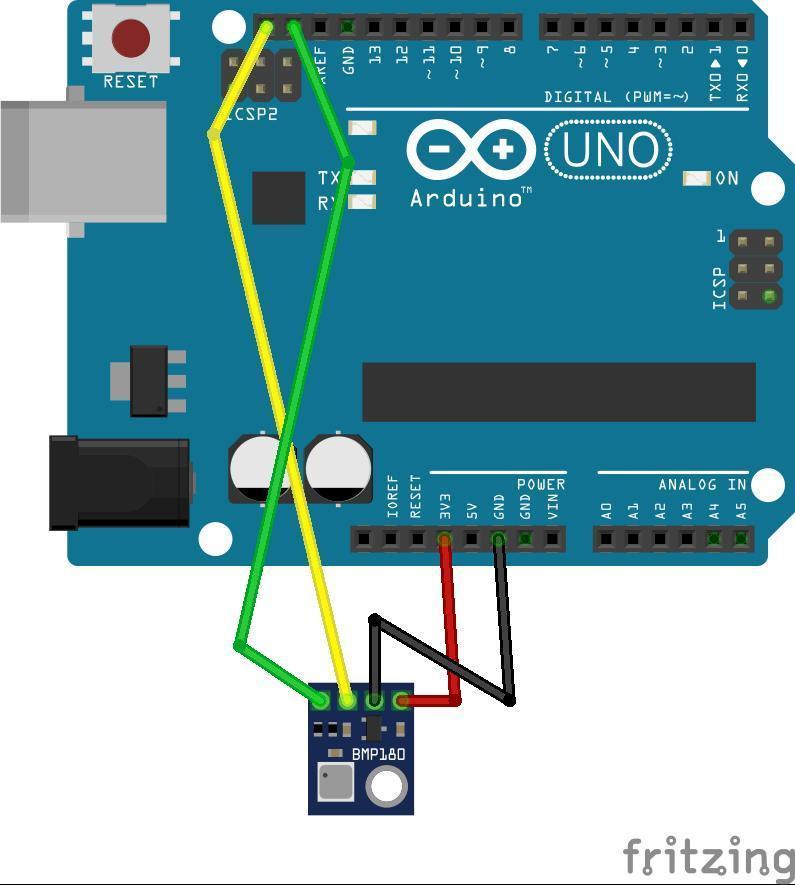
Build install and develop a sketch for Arduino for reading BMP180 sensor readings  
  
[](https://ecoimpact-ple.com/images/452_dguoq2pHSDTvrSRUtO8lnA.jpeg)

Figure 45. Connect the BMP180 sensor to the Arduino Board according to the scheme

Go to the Arduino IDE. Connect the library "Adafruit BMP180 Unified".

Before the setup() function, add the global variable bmp

Adafruit\_BMP085\_Unified bmp(10085);

In the setup() function, start the serial port and initialize the BMP180 sensor

Serial.begin(9600);

bmp.begin();

Next, in the loop() function, read the pressure value

sensors\_event\_t event;

bmp.getEvent(&event);

Output pressure in hPa

Serial.print("Pressure (hPa): ");

Serial.print (event.pressure);

Read the temperature value

float temperature;

bmp.getTemperature(&temperature);

and output it

Serial.print("Temp (\*C): ");

Serial.println(temperature);

Adding a delay

delay(1000);

**8. DEVELOPMENT OF A DEVICE FOR ECOIMPACT IOT BASED ON AN ARDUINO MICROCONTROLLER**

1. Install lib for the Protocol of interaction with the application ECOIMPACT.

2. Development of the sketch.

3. Checking the interaction with the device via the serial port monitor.

Tools for performing work

1. A computer with an Internet connection.

2. Arduino Board with USB output (for example, Arduino Uno).

Theoretical part

In order to simplify the development of IoT devices and unify their connection to the ECOIMPACT platform, a simple text Protocol was developed for the interaction of the device with the computer through various communication channels designed for serial transmission of information. The Protocol is intended for the organization of interaction between two devices (point-to-point) through the exchange of simple text messages. Using text messages makes it easier to develop and debug a device, as it allows you to interact with it without specialized software from the command line or serial port monitor.

Execution of work

Install lib for the Protocol of interaction with the application ECOIMPACT

1. Download the archive with the source code at https://github.com/ooolms/wl\_iot\_framework

2. Unzip this archive, you should see the folder wl\_iot\_framework.

3. Install the library through the library Manager: select the menu item " Sketch - > library Management - > Add .ZIP library" and find the archive of the ARpc.zip inside the wl\_iot\_framework folder in the ArduinoIdeLibrary subfolder.

Sketch development

As part of the work, a sketch will be created that allows the led to flash from the ECOIMPACT application interface and transmits "measurements" once every half-second (generated two-dimensional signal (sin(t); cos (t)) ). In addition, the device will have another "sensor" - led flashing counter.

To communicate with a computer via a serial port, we will create an ARpc class object and define two callback functions for It - one for processing commands from the PC, the second for sending messages to the computer. These functions will be called by the library itself if necessary. To uniquely identify the device it is also necessary to specify the ID and name of the device. And to provide the ability to control the device, you need to develop an xml description of the device control panel.

Create a new sketch and save it under the name ekoimpaktest. Check that the Board and port are correct. Connect the arpc library to the sketch (sketch - > connect library - > arpc). The desired # Include < arpcdevice should appear at the beginning of the file.h>.

Generate a unique identifier in the UUID format (for example, you can use the service https://www.uuidgenerator.net/version4, when you open the page at the top will be ready UUID). Adding two global variables for the device ID and name:

constant type char \*device name= "led\_blink\_test"; / / device name constant device ID ARpcUuid ("{xxxxxxxx-XXXX-XXXX-XXXX-xxxxxxxxxx}"); / / device ID

xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx replace with the resulting UUID, curly brackets should remain. We also define additional global variables:

int ledPin=13; / / led pin uint32\_t blinkscount = 0; / /number of blinks

In order to make the device management interface available to the device in the future, you must also develop an XML description. PDF PDF-description (link from the theoretical part). Our scenario requires a single button that sends the "blink" command to the controller, which is described as follows:

constant type char \* interfaceStr=" < control><group name= \ "control device\" > < control name=\ "blink \" command=\ "blink\" />< / groups>< / elements>";

Also, to get data from the device, you need to prepare a description of the sensors. For our device it looks like this:

const char \*sensorsDef= " <sensors>"

"<sensor name=\ "blinks\_count\" type=\ "u32\_sv\" / > " / / sensor blinks\_count "< sensor name = \ "sin\_x\" Type = \ "f32\_sv\_d2 \" / > " / / sensor sin\_x(two-dimensional) < / Sensors>";

Define a class for sending messages to the PC via the serial port:

WriteCallback class :public ARpcIWriteCallback{public: virtual void writeData(const char \*data, unsigned long sz) { Serial.write (data, sz);} virtual void writeStr(const char \*str) {Serial.print (str); }}wcb;

An object of this class is used when the library needs to pass some data from the device. In this case, this is the data that we will see in the port monitor. Note prin println, since this function adds an extra line feed that will interfere with normal message processing.

Next we declare an object of the Armr class and pass it references to the variables and functions created above:

ARpcDevice dev(300, & DSP, & device ID,device name);

The class Defines the processing of commands passed to the device. The library will use an object of this class when commands are sent to the device, such as those we entered in the port monitor. The handler function accepts the command, the command arguments, and the number of arguments:

CommandCallback class

:public ARpcIDevEventsCallback {

public:

virtual void processCommand(constant type char \*UMK, constant type char \*parameter args [], unsigned character argsCount)

{

if (strcmp (cmd,"blink")==0)/ / BL Blink, check that there is an argument

{

digitalWrite(13, HIGH);

delay(500);

digitalWrite(13, low level);

++blinksCount;

development.disp ().writeMeasurement ("blinks\_count", String(blinksCount).c\_str());

development.disp ().writeOk();

}

another Dev.disp ().writeErr ("unknown cmd"); / / unknown command

}

}CCB; Here we process one command - "blink", when it comes to blink the regular led on port 13 and pass a new "measurement" of the blink counter.

Then a function is prepared for generating sin and cos counts

int t=0;

float sVal[2];

void writeSinVal()

{

sVal[0]=sin(0.1\*t);

sVal[1]=cos(0.1\*t);

dev.disp().writeMeasurementB ("sin\_x", sVal, 2);

++t;

}

300-buffer size for a single message. You cannot send a message larger than the specified size to the controller.

The size of the buffer should be selected based on the available memory. On microcontrollers with a large amount of memory, you can use a larger buffer size.

Enter the pin and serial port initialization in the setup() function and set the description of the sensors and the control interface:

void setup()

{

Serial.begin(9600);

pinMode(ledPin, OUTPUT);

dev.disp().installDevEventsHandler(&ccb);

dev.disp().setControls(interfaceStr);

dev.disp().setSensors(sensorsDef);

}

Finally, in the loop() function, you need to check the serial port for new data, pass it to the parser object, generate a new sin count, and then make a delay of half a second so that the counts are not generated too much часто.poizzzennn@mail.ru

void loop()

{

while (Serial.available())

dev.putByte(Serial.read());

writeSinVal();

delay(500);

}

Upload the resulting sketch to the microcontroller.

Checking the interaction with the device via the serial port monitor.

Open the port monitor. It should regularly receive the message "meas" with the new values of sin and cos.

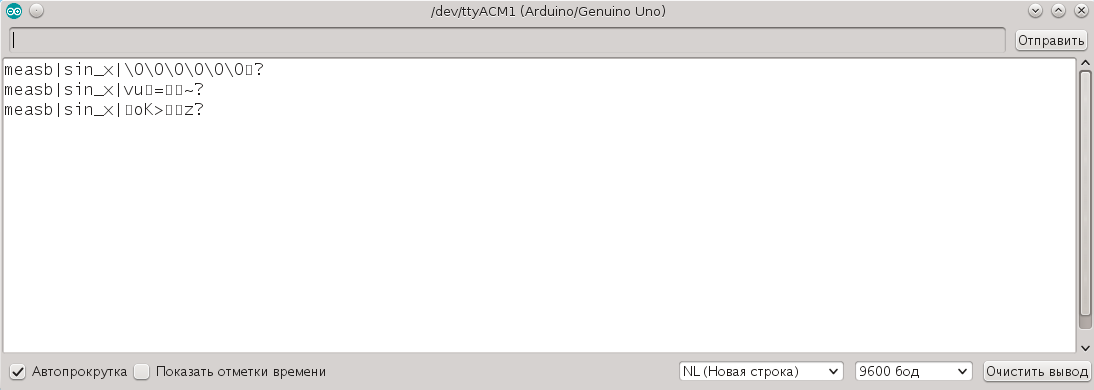
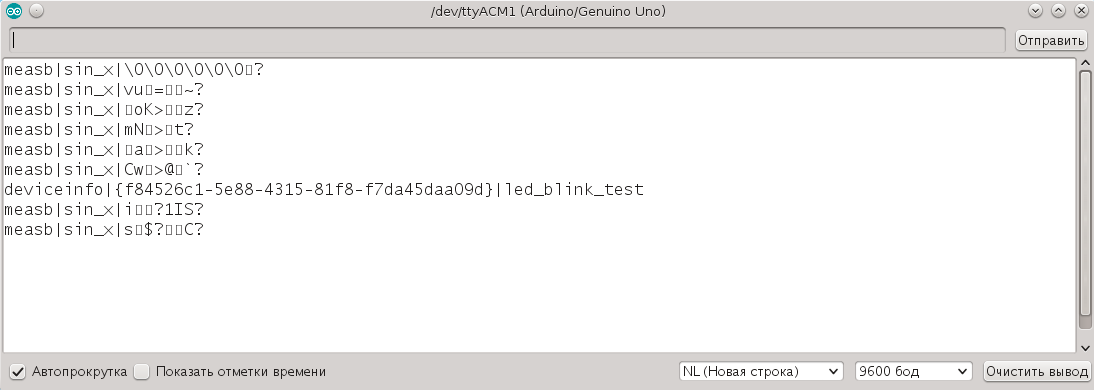
[](https://ecoimpact-ple.com/images/1904_HpRaVu9dzLvY6WOdHScYXg.jpeg)

Figure 46. Values of sin and cos.

Check that "New line" is selected at the bottom, not "No end of line".

Write in the input field "identify" and click Send. The device info message should appear in the

response.

[](https://ecoimpact-ple.com/images/1905_j7u6VMSGjK9KxyNd4L4w.jpeg)

Pic 47.Device ID and name

This message must contain the device ID and name specified in the sketch.

**9. CONNECTING THE DEVICE TO THE LOCAL ECOIMPACT SERVER**

Learn how to connect a device to a configured local ECOIMPACT server

Work tasks

1. Connect to a PC with a local server installed via a remote terminal and connect the device to the local server

2. Create storage for values from the sensor

3. Bind storage to the ECOIMPACT cloud

Tools for performing work

1. A computer with an Internet connection.

2. An Arduino Board with a USB output (for example, an Arduino Uno) and a loaded sketch from the lab work "developing a device for Alterozoom IoT".

Theoretical part

The local ECOIMPACT server is a self-contained Autonomous point for connecting devices and collecting and processing data. It also provides the ability to duplicate data coming to the local storage in the cloud storage ECOIMPACT.

Execution of work

Connect to a PC with a local server installed via a remote terminal and connect the device to the local server

Note: if several people are using the same local server at the same time, you must use different device names by changing the sketch (replace led\_blink\_test, for example, with led\_blink\_test2, led\_blink\_test3, etc.). Accordingly, the same way to replace the device name in all the commands given below in the work.

To connect to a PC with a local service installed, use the SSH Protocol (and the same utility under Linux). Putty is available for the Windows operating system. When connecting to a PC with a local server, you need to specify a username and password.

Note: if you are using a pre-configured PC with a server, ask the administrator for the username and password.

The next step is to physically connect the device to a PC with a local server using a USB cable.

Then run the command in the open terminal

wliotproxy list\_identified and make sure that the device with the desired ID and name led\_blink\_test is present in the list.

Creating storage for sensor values

First you need to make sure that the list of sensors from the device is available. Executed command wliotproxy list\_sensors led\_blink\_test and check that there are two sensors with the names blinks\_count and sin\_x.

Create a storage for the sin\_x sensor with the command

wliotproxy add\_storage led\_blink\_test sin\_x last\_n\_values add\_global\_time --N=1000

Here "led\_blink\_test" is the name of the device, " blinks\_count "is the name of the sensor on the device," last\_n\_values "and" --N=100 " mean that the last 100 values will be stored, and add\_global\_time means that when writing data, you need to add a global time stamp in the local server.

**10. LINKING STORAGE TO THE ECOIMPACT CLOUD**

To bind the storage to the Ecoimpact server, you must make a preliminary preparation.

You must log in to the remote server under the desired account (for example, [test@example.com](mailto:test@example.com)) wliotproxy-alterozoom-auth test@example.com

The application will ask for the user's password during operation, and in case of successful authentication, it will issue a message with the user ID. By calling the same application without parameters, you can make sure that the user is present in the list. If everything is OK, you need to restart the server systemctl restart wliotproxyd

You can then link the storage to a remote server

wliotproxy data\_export add led\_blink\_test sin\_x alterozoom email:test@example.com host:ecoimpact-ple.com

10. Connecting to WiFi using the ESP8266 NodeMCU microcontroller

Learn how to connect to a WiFi network with an Arduino microcontroller and transfer data over WiFi.

1. Add support for NodeMCU microcontroller based on ESP8266 WiFi module

2. Develop a sketch for connecting to a WiFi network and transmitting data wirelessly.

Tools for performing work

1. Computer with Internet connection

2. The microcontroller NodeMCU V3 on the basis of EPS8266-12E

Theoretical part

The NodeMCU Board is based on the esp8266 WiFi module.

[](https://ecoimpact-ple.com/images/458_LomKDuXVJPl11utoFjSw.jpeg)

Figure 48. The Pinout of the Board

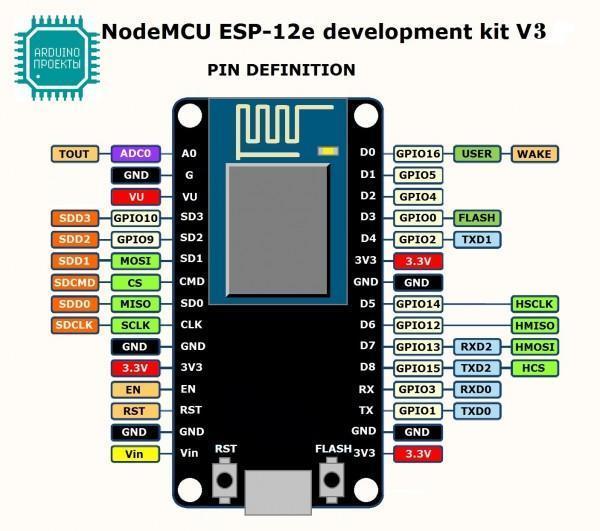
[](https://ecoimpact-ple.com/images/459_AomOWdGg1HnzQ2DOuoXKw.jpeg)

Figure 49. D6 (GPIO12)

Support for the Arduino IDE development environment has been added to work with this Board. Almost all libraries for regular Arduino also work for this Board, but its Pinout is different from the Pinout of Arduino boards. So, the Board has one analog input (A0), which runs a 10-bit ADC.

The digital pins 0-15 of the Arduino correspond to the outputs designated GPIO1-GPIO16. For example, if the Arduino IDE uses pin 12, it will correspond to the pin marked on the Board as D6 (GPIO12). The GPIO1 — GPIO5, GPIO10, GPIO12-GPIO15 pins support PWM.

1. Board features:
2. support WiFi protocols 802.11 b/g/n
3. Wi-Fi Direct( P2P), soft-AP
4. built-in TCP/IP stack
5. SDIO 2.0, SPI, UART
6. built-in PLL, controllers, and power management system
7. Rated voltage: 3.3 V
8. Input voltage: 3.7–20V
9. Maximum current consumption: 220 mA
10. D9, D10-UART
11. D1, D2 — I2C (D1-SCL(SCK), D2 - SDA)
12. D5–D8-SPI
13. The main site of the project: https://github.com/esp8266/Arduino
15. Execution of work
16. Adding support for the NodeMCU microcontroller based on the ESP8266 WiFi module
17. To add support for NodeMCU in the Arduino IDE you need to perform the following steps:
18. 1. Open settings (File - > Settings)
19. 2. Find additional links for the Board Manager and click the edit button on the right (highlighted in the screenshot)

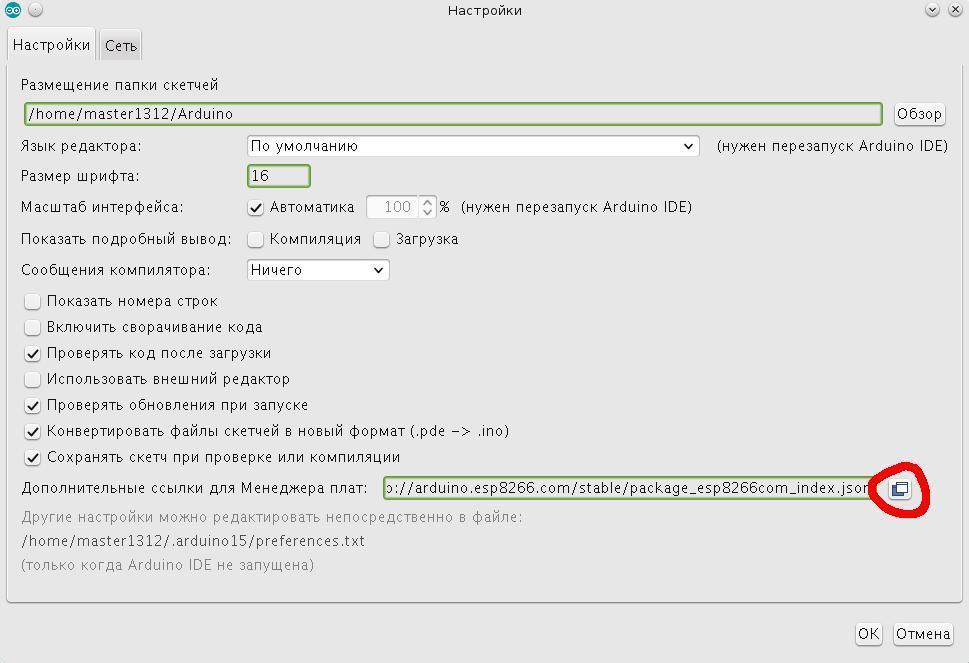
[](https://ecoimpact-ple.com/images/461_sjbbkPkbdvSia4HtUmhvQ.jpeg)

Figure 50. Open settings (File - > Settings)

Adding the URL: http://arduino.esp8266.com/stable/package\_esp8266com\_index.json

Save and go to the Board Manager (Tools - > Board: - > Board Manager)

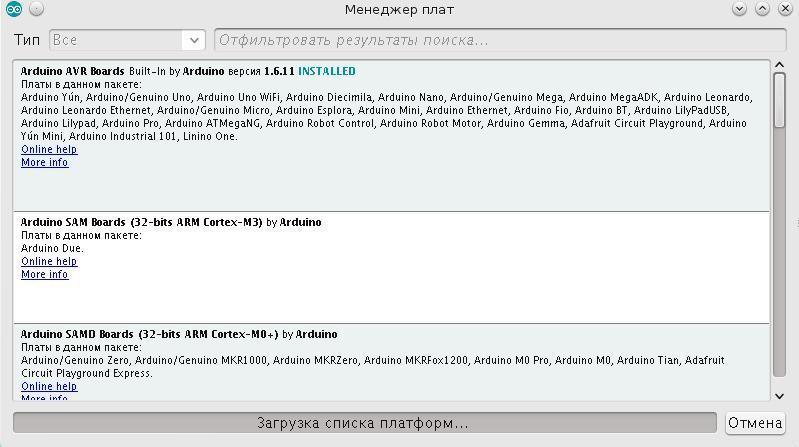
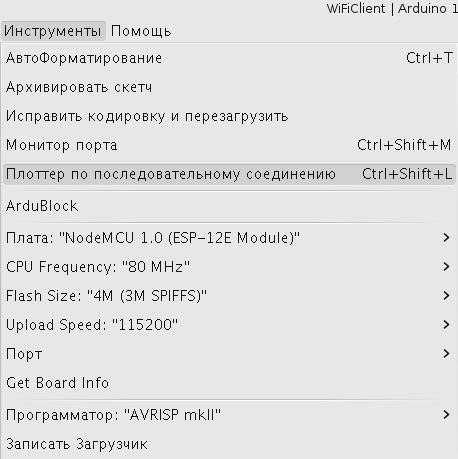
[](https://ecoimpact-ple.com/images/462_zoIX3oFCqMElSoENi0lA.jpeg)

Figure 51. Tools -> Board: - > Board Manager

Enter "esp8266" in the search and install the esp8266 package

Create a new sketch and save it as Esp8266Test.

Select the Board "NodeMCU 1.0 (ESP-12E Module)" (Tools - > Board:). Below, in the "Tools" menu, enter the settings for CPU Frequency, Flash Size and Upload Speed as shown in the screenshot (most likely, these will be exactly the same, if the Board differs from the one shown above, the settings may differ).

[](https://ecoimpact-ple.com/images/460_SpfSiqVnZtPA3O2H1dNZNw.jpeg)

# Figure 52. Select the desired port

# Development of a sketch for connecting to a WiFi network and transmitting data over a wireless communication channel.

# Sketch to work with WiFi, we use libraries similar to the standard WiFi library for ordinary Arduino. When working with a regular Arduino and some WiFi shield, similar header files will be connected, only without the prefix "ESP8266".

# Plug-in header files

# #include <ESP8266WiFi.h>

# #include <ESP8266HTTPClient.h>

# Specify the parameters of the WiFi network

# const char \*essid= " WIFI\_ESSID";

# const char \*key= " WIFI\_KEY";

# WIFI\_ESSID and WIFI\_KEY are replaced with the name and password of the used WiFi network.

# In the setup() function, run Serial and connect to WiFi

# Serial.begin(9600);

# Wi-Fi access.begin(essid, key);

# while (WiFi.status ()!= WL\_CONNECTED)

# {

# delay(500);

# Serial.print(".");

# }

# Serial.println ("WiFi connected");

# In the loop function, we execute a GET request to the web page of the remote server. To do this, connect to the server

# WiFiClient client;

# if (! client. connect ("wl.unn.ru", 80))

# {

# Serial.println ("connection failed");

# return;

# }

# Send a manually generated GET request

# client.print("GET /laboratory/?page=1 HTTP/1.1\r\nConnection: close\r\n\r\n");

# Waiting for a response from the server

# unsigned long timeout = millis();

# while (client.available ()==0)

# {

# if(millis()-timeout>5000)

# {

# Serial.println (">>>Client Timeout !");

# client.stop();

# return;

# }

# }

# Output the answer and sleep for 10 seconds

# while (client.available())

# {

# String line=client.readStringUntil('\r');

# Serial.print(line);

# }

# delay(10000);

# Load the sketch on the Board and open the port Manager. There every ten seconds should appear the code of the web page, which came in response from the server wl.unn.ru.

# Additional task: develop a sketch for scanning available WiFi networks

# 11. DEVELOPMENT OF IOT DEVICES USING BLUETOOTH LE

# Bluetooth technology is vigorously making its way into the Internet of things. Part of this technology, called Bluetooth LE (Bluetooth Low Energy, aka Bluetooth Smart, aka BLE) directly positions itself as an ideal choice for IoT (Internet of things). It's hard not to agree. BLE already knows how to route Internet traffic, determine coordinates in the premises, connect industrial programmable logic controllers, support WEB servers, connect scales, thermometers, heart rate monitors, oximeters, tonometers and a lot of other things. C BLE automatically solves many problems inherent in solutions using Wi-Fi. It will not be long before devices with BLE can be organized in a MESH network, using a technology similar to ZigBee. This is already reflected in the Bluetooth 5.0 specification, so when developing IoT module, an absolute preference to BLE was given as opposed to using Wi-Fi. It is considered that the peripheral part of the BLE network using the example of the K66BLEZ debugging module.

# Here we would like to describe my development route from almost complete ignorance of BLE to the production of a serial product

# The k66blez module uses the MKW40Z160 chip (48 MHz Cortex-M0+, 160 KB Flash, 20 KB RAM) produced by dig NXP as a ble receiver-transmitter. The chip is interesting because, along with BLE, it can also work as a receiver-transmitter of signals of the 802.15.4 standard. And the 802.15.4 standard is known to be the carrier in ZigBee technology. The ZigBee stack for MKW40Z is not released directly, but it is already offered to the firmware where 802.15.4 works simultaneously with BLE

# The MKW40 chip is replaced by the MKW41 chip with 128 KB RAM, 512 KB Flash and support for all popular protocols: BLE 4.2, BLE Mesh, ZigBee, Thread, IPv6 6LoBLE. There is no open documentation for the new chip yet, but it promises to be pin compatible with the MKW40.

# THE ble MKW40 chip on the module connects to the main MK66 microcontroller via the SPI and I2C interfaces. the I2C Interface also connects the chip to the charger chip. The main communication channel is implemented on the SPI interface with a bit rate of 6 Mbit / s.

# Program debugging in the MKW40 chip can be performed via the SWD interface using the JTAG adapter and via the UART0 debugging interface also output to the x4 debugger connector.

# NXP provides more than two dozen examples of implementation of various applications on the MKW40 chip, including: pressure, glucose, temperature, proximity sensors, heart rate meters, etc. There are applications for wireless UART and wireless bootloader.

# We did a deep refactoring of the NXP framework for these chips and created new profiles with demo programs on Windows PC that do not require a separate adapter on the PC side. But more on that later.

# Bluetooth LE is difficult to learn. The reason is a voluminous specification and a large number of its brief retellings in the documentation of manufacturers, immediately beginning with unusual terminology. So let's start with it.

# Interpretation and translation of terms and abbreviations, slang.

# \* Pairing-binding (pairings). The process of creating one or more shared secret keys by BLE device pairs for subsequent traffic encryption. The user is involved in this process when the system asks for a PIN code.

# \* Bonding-binding. The process of saving shared secret keys for use in subsequent trusted connections of BLE device pairs.

# \* Device authentication-check (authentication) for the fact that two devices have the same secret keys.

# \* Advertising-The process of broadcasting ble device notification packets (advertising). In these packages, the device reports its name and address, reports on the services it provides, and special information.

# \* Scanning-the process of receiving advertising packets from other ble devices during passive scanning. When scanning is active, it sends packets of requests for additional information from devices running in the advertising mode.

# \* Profile-profile. A set of lists of functions, properties, behaviors, and roles for a set of levels in a particular Protocol stack.

# \* UUID — universally unique identifier. 128-bit unique attribute identifier.

# • BLE Host-host. The part of the ble stack software that runs on the main processor that runs both the main application and the bridge functionality to the main application. The host contains GAP, GATT, GATT database, L2CA.

# • BLE Controller-controller. Part of the ble stack software running on a Bluetooth radio chip.

# • HCI-Host Controller Interface. Protocol or API depending on the context for interaction between THE ble host and THE ble controller.

# \* GAP-Generic Access Profile, a typical access profile. This is usually immediately referred to as a layer. But it's rather strange to call a profile a layer. In the source code, this is represented as a set of macros, declarations, and functions for establishing and maintaining communication between BLE devices.

# \* GATT-Generic Attribute Profile, a typical attribute profile. In the source code, this is a set of functions for exchanging data between devices. Attributes are units of data of different types (strings, numbers, structures...) organized as a hierarchical tree whose nodes are services, characteristics, descriptors, etc. an Attribute is characterized by having a unique UUID.

# • L2CA-Logical Link Control and Adaptation Layer. The software layer with the corresponding Protocol is responsible for establishing and maintaining logical communication channels. It deals with forwarding planning, error control, packet segmentation, thread management, and packet multiplexing between top-level protocols. Is part of the ble host.

# \* SMP-Security Manager Protocol. The Protocol used for pairings. Works on a dedicated channel in L2CA.

# • LTK — Long-Term Key. Secret key used when encrypting ble traffic.

# \* IRK-Identity Resolving Key. The key to decrypt the real address of the device from the confusing public one.

# \* CSRK-Connection Signature Resolving Key. Key for signing messages.

# • RAND-64-bit random variable used to generate LTK

# \* EDIV-16-bit random variable used to generate LTK

# \* MITM-man-in-the-middle. An attempt to open a third party joint secret key of two devices by embedding in the communication channel between the devices as an intermediate link.

# \* Message integrity - protection against fake messages.

# \* Framework — this is what we call software in the source code, designed to simplify the creation of applications on a specific hardware platform with specific libraries of communication Protocol stacks. It usually includes BSP (board support package), HAL (hardware abstraction layer), OSA (OS abstraction layer), intermediate software (middleware) such as: memory managers, file systems, schedulers and timers, and so on.

# Analysis of competing solutions

# When choosing a chip for BLE, I did a little analysis of offers from the most famous manufacturers. Most of all, we were interested in the composition of the proposed software, frameworks, and tools for compiling, building, and debugging projects for the ARM core. An important factor was the continuity with the IAR environment and the RTOS MQX framework that are used when developing an application on the main processor of the module.

# NORDIC SEMICONDUCTOR provides an SDK for the nRF51822 chip with a Cortex-M0 core. Compiled in IAR, KEIL, GCC. The ble stack is represented by a monolithic library without source code called SoftDevice where all APIs are implemented: GAP, GATT, L2CA, HCI. A framework with drivers is built around this library. The framework comes with two RTOS: RL-ARM RTX from Keil, and FreeRTOS. The framework uses protobuf serialization technology and segger RTT debugging. instead, the nrf5 IoT SDK is offered. It includes sources for MQTT, COAP, TLS (taken from the mbed project), cJSON, lwip (free TCP/IPv4/IPv6 Protocol stack), socket interface, and IPv6 adapter. There is also 6LoWPAN, but without the source code.

# Texas Instruments

# ARM only makes 2-core ble CC2640 chips (Cortex-M3 and Cortex-M0), but corresponding to the Bluetooth 4.2 specification.For download, the SimpleLink Bluetooth low energy Software Stack 2.2.0 SDK is Compiled by the native Code Composer Studio development environment, as well as in the IAR environment. Comes with its own RTOS TI-RTOS 2.16 and a developed framework around the ble stack libraries. The SDK as one of the scenarios involves the use of an external application processor-Simple Application Processor (SAP). The CC2640 chip itself is referred to as the Simple Network Processor (SNP). They communicate over a Protocol called the Unified Network Processor Interface (NPI). On the CC2640 side, TI-RTOS is required, and on the SAP processor side, RTOS can be used as desired. The SDK provides NPI Protocol sources for both the SAP and SNP sides. This is the SimpleLink technology.

# The ble stack itself is divided into 3 precompiled libraries without source code: host, controller, HCI. All three libraries only run on the Cortex-M3 processor, which is part of the CC2640 chip. In addition to studying TI-RTOS, the user here will need to learn a special software mechanism or Protocol for interacting with the BLE stack called iCall.

# Microchip-Atmel

# produces Bluetooth LE atbtlc1000 chips on Cortex-M0 core. The entire stack in the chips is written to ROM. There are no open tools for programming these chips on the Atmel website. Atmel instead suggests using an external microcontroller to interact with the ATBTLC1000. Software for an external microcontroller and examples are in the Atmel Software Framework package. Compiled in Atmel Studio (GCC shell) or in IAR.

# Cypress Semiconductor Corp.

produces families of programmable BLE chips on the Cortex-M0 — PSoC 4 core: PSoC 4XX8 and PRoC CYBL1XX7X supporting the Bluetooth 4.2 specification. Projects for chips are created in a special IDE PSoC Creator. Cypress chips differ in that there is no ready-made configuration of peripherals (UART, SPI, I2S, PWM, etc.), it must be created from library elements in the circuit editor with the addition of software libraries. This is intended to provide some flexibility. Although it adds a considerable amount of work to the developer. The configured project can be compiled by one of the following toolchains: GCC, IAR, Keil. BLE there is one of the libraries. The BLE stack is delivered as a precompiled monolithic library without source code combining THE ble host, ble controller, and HCI. However, the company has laid out the source code of applications for Android and iOS working with BLE.

Silicon Labs produces EFR32 Blue Gecko Bluetooth Smart SoCs on an ARM Cortex-M4 core supporting the Bluetooth 4.2 specification, EFR32BG1P332F256GMxx type Chips can deliver power up to 19.5 dBm and combine a separate 868 MHz radio channel with power up to 20 dBm and sensitivity -121.4 dBm. The chip of Silicon Labs chips is a huge selection of alternative pin functions and a system called the Peripheral Reflex System (PRS). Although the peripherals cannot be created as the chips from Cypress, but its connection to the pins is almost arbitrary, the presence of PRS makes it possible to interact with each other peripherals without involving the processor. The BLE stack from Silicon Labs is able to accept the results of profile generation by the Bluetooth Developer Studio program, which will be discussed below. Silicon Labs offers two Bluetooth stacks. One of them is designed for Bluegiga modules and supports in addition to BLE and regular Bluetooth. The second stack complies with specification 4.2 and only LE. The BLE stack is delivered as a monolithic precompiled library with no source code. For the option with an external microcontroller, a serial Protocol and API are offered in the source code. It can be compiled in both GCC, IAR, and Keil. Everything is done in a single development environment Simplicity Studio V4. The accompanying stack framework is not supported by RTOS. But in the source code of Simplicity Studio you can find such pearls as Speex under 8 kbit/sec suitable for voice transmission over BLE and a powerful window GUI from Segger.

STMicroelectronics makes BlueNRG network controller chips based on Cortex-M0 containing a ble stack according to the Bluetooth 4.1 specification. The chips themselves are not programmable, but have a serial application command interface (ACI) through which an external microcontroller must communicate with them. A framework has been developed for ACI, and it can be included as part of the proprietary stm32cube development environment from ST.

CSR PLC does not make BLE chips on ARM Cortex, but is interested in its implementation of MESH networks on Bluetooth modules. Video here. The sources of various BLE applications for Android and iOS are laid out. There is an SDK.

Renesas

makes BLE chips on its 16-bit rl78 core. The BLE stack is issued only to premium users. All its own-compiler, RTOS, host microcontroller. But there is a plugin for Bluetooth Developer Studio Dialog Semiconductor

offers, as they claim, the smallest BLE chips. However, the chips with Flash memory DA14583 (the rest only with ROM) cannot be called the smallest-5x5 mm. Core Cortex-M0. The maximum power of 0 dBm. Support Bluetooth 4.1 specification. To get the SDK from the company, you need to register and pass the verification. But with these chip parameters, I didn't even try to get the SDK.

Therefore, the sources MQTT, COAP, TLS, SPEEX, LwIP, and so on. we are not interested in the different SDKs, they can be found freely on Github without binding to specific frameworks. Support for the Bluetooth 4.2 specification does not give much because it is not yet possible to use it on a PC.

The ble stack for Kinetis, like others, comes in the form of precompiled libraries. Around these libraries, a multitasking framework is built that includes drivers and a layer of hardware abstraction in the source code that is independent of the operating system. The framework can be configured to run without an operating system, or it can use one. Immediately in delivery, the framework is adapted for FreeRTOS. But it interacts with FreeRTOS through an auxiliary set of functions called the operating system abstraction layer (OS abstraction, OSA).

With OSA, you can substitute any other operating system that supports message Queuing, crowding, flags, and timers instead of FreeRTOS. For example, RTOS MQX. The stack is compiled, oddly enough, only in the IAR environment. Fortunately, in my case, this is not a problem. More interesting is that the stack is divided into two libraries-BLE host and BLE controller. And the ble host library can run on a different chip. Libraries interact with each other in this case through the HCI Protocol. i. e. where other manufacturers come up with another communication Protocol for application interaction on an external microcontroller with a ble stack (remember SimpleLink), NXP offers a standard solution. And most importantly, with this approach, moving the BLE host to a more powerful external microcontroller significantly increases the capabilities of our GATT database and services.

Briefly about BLE

The open version of the Bluetooth specification version 4.2 is available here. The description of the lower BLE level (Controller level) is included in it as "Vol.6 Core System Package [Low Energy Controller volume] " from page 2544. The upper level (Host level) with the description of the att Protocol and GATT profile is in the part " vol.3 Core System Package [Host volume]" of the document with page 1693.

**Range of frequencies used**

Three frequencies (in the figure above, indicated by channel numbers 37,38,39) are allocated for broadcast unaddressed parcels, and the rest for transmitting packets when establishing logical communication channels between devices. A well-known feature of Bluetooth is that when transmitting packets, each subsequent packet is transmitted on a different frequency selected pseudo-randomly from the list of allowed ones. All data in BLE packages can be encrypted and authenticated. Also apply the dynamic random generation of addresses for the devices and their identification using a hash, i.e. having intercepted the address of the device in the air, we will not be able to use it for longer than 15 minutes, because the address will change during this time according to an unknown algorithm for us.

BLE modules can work as unidirectional transmitters, i.e. without installing a bidirectional connection, just broadcast some data in the form of ad packets, such as temperature. To do this, you can use the data type in Advertising packages designated as Manufacturer Specific Data. A computer or tablet can receive data from hundreds of such transmitters without the need to search, establish a connection, enter a pincode etc.

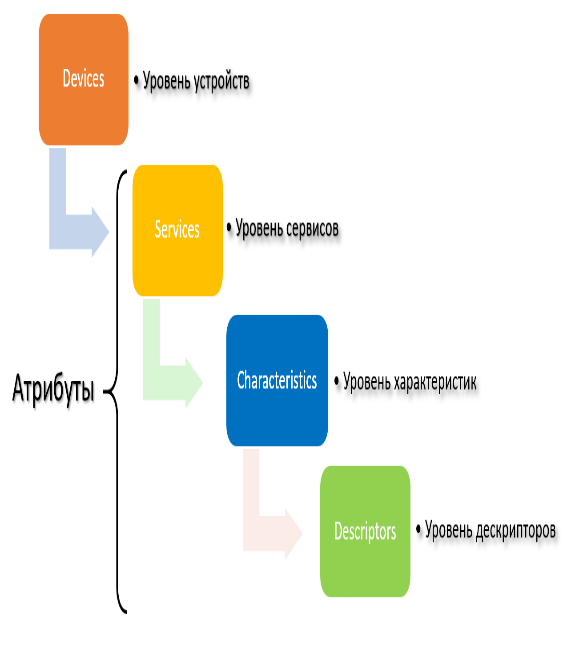
the only way to transmit data without setting a communication channel is to transmit in the request-response mode (request-scanrequest package, module response-ScanResponce package). This BLE is significantly different from Wi-Fi, where even for the simplest thermometer, you need to establish a connection that takes up the resources of the router.

Ble Protocol stack

The figure below shows the BLE as seen by the microcontroller programmer. The ble stack consists of two software parts: Host and Controller. The Host software handles high-level data management and connection management functions, while the Controller manages the physical peripherals of the transceiver, works with secret keys, and handles other low-level functions. The named parts are connected by the HCI software interface (Host Controller Interface). In a PC implementation, part of the Host runs on a computer, and part of the Controller runs in a Bluetooth hardware transceiver, and the HCI Protocol is most often transmitted over USB. In implementation on the microcontroller both parts work on one chip, and the HCI interface turns simply into direct data transfer from the task (program module) of the host to the task (program module) of the controller and back In fact the programmer sees several sets of APIs working at the Host level: called GATT, GAP, L2CA, SMP, HCI. The GAP API allows you to set the device's operating mode — Central, Peripheral, Observer, Broadcaster-and establish a connection when needed. And with the help of the GATT API, you can directly transmit and receive useful data and parse it.

Most existing devices still support VERSION ble 4.1, despite the existence of version 4.2.All the differences between version 4.2 and the previous one are related to improvements in the BLE part: increased speed, the ability to transmit IP Protocol and HTTP traffic, increased cryptographic protection and non-recognition for external observers.

An important feature of BLE compared to Wi-Fi is the specification of not only the communication channel, but also the application applications that use it. This is called profiles and services. Profiles with services describe device roles, the purpose of data, the composition and format of data, data protection, the order, types, and events of exchange, and not just how data is transmitted. This allows you not to invent any of the protocols when developing, for example, a body temperature sensor or a pulse meter. The specifications are already given, it remains on the device side only to fill in the necessary fields to send the measurement results. Customers of such devices in the form of smartphones, tablets, PCs or kitchen appliances will recognize this data automatically and display it or use it accordingly. All thanks to the fact that all manufacturers are guided by the same ble specifications about how the temperature or pulse data is presented and how to work with them. But there is still room for the developer's imagination, as profiles have mechanisms for extending functionality. The following is a rough hierarchy of attributes in a BLE device.

[](https://habrastorage.org/getpro/geektimes/post_images/7df/ca0/0fe/7dfca00fe6427b57e47b1ba1e10c4767.png)Figure 53. Attribute tree

Below is a slightly more detailed typical attribute tree. This is not a complete tree, most are omitted because it would take up too much space. Colors highlight the levels of the tree; each attribute has a unique number-UUID. Recording standard numbers is reduced to 16-bit. In this picture, all numbers are standard. GAP and GATT profiles are also presented as services with their standard characteristics. Each service can have its own security model and authorization. The entire tree in the device is stored as a database called the GATT database, usually as a simple cross-referenced table

The characteristics of services have many characteristics, as shown below. We’ll have to apologize for the tautology here, but there really is some kind of terminology crisis in BLE. In short, the characteristics belonging to the service can be specified permissions to read, write, the need for notifications, confirmations, signatures, and so on. BLE is a serious technology, so much has been done to ensure security and maximum formalization, which should in turn make it easier to achieve compatibility.

Data exchange between BLE devices is performed by recording and reading characteristic values. There are no streaming channels such as TCP or UART. And if the devices have them, then they are organized by higher-level software add-ons.

**Development tool**

Development tools offered by the Bluetooth Special Interest Group (Bluetooth SIG) website — https://www.bluetooth.com/develop-with-bluetooth/developer-resources-tools the following useful tools are available on the website of the main standardization organization — Bluetooth SIG:

**Bluetooth Developer Studio**

Bluetooth Developer Studio is a tool that allows you to correctly create and insert profiles, services, characteristics and descriptors in the ble implementation of the device, i.e. create a database. If you buy an additional hardware Bluetooth adapter for $ 99, the program allows you to intercept, decrypt and display Bluetooth Protocol packets. There are also features in the program for debugging and testing of created services. Since the approved profiles are described in great detail in BLE, even minor errors about the format, numbering, availability, and so on. these profiles will cause compatibility issues. But even for non-standard profiles, it is very difficult to do without a tool that accurately constructs a tree of services, characteristics, descriptors in compliance with all specifications. It is easy to get confused in the names of services, characteristics, descriptors and their multi-byte unique numbers-UUID. The result of the tool in particular are generated XML files describing profiles, services, characteristics and descriptors in the user's project. These XML files are directly used by Silicon Labs ' Simplicity IDE for integration into embedded projects for their chips. Another result of the tool can be the source code for the device working with the BLE database. But to do this, the user needs to write their own plugin in JavaScript. The program will also provide the user plug-in with access to the database via a special JavaScript API. There are a number of ready-made plug-ins that generate various source text files suitable for compiling in third-party environments and software frameworks for solutions based on the NXP Kinetis KW40Z Connectivity Software framework.

**Application accelerator**

Application accelerator 2.1-a set of demo projects with source code for different operating systems Android 6.0, Blackberry, iOS 9, Tizen 2.4 and Windows 10. For Windows 10, projects are only available for the Visual Studio development environment under the Universal Windows Platform (UWP) architecture. That is, these projects cannot be compiled under Windows Forms or WPF c .NET frameworks. And bridges for translating ordinary Windows applications into UWP are just being created. It should be noted that UWP makes it possible to place applications in the Windows Store, but it does not create a simple executable .an exe file that you can simply copy and run. The first launch of the UWP application is always accompanied by an installation. All this creates difficulties for the developer. And the functionality of demo projects leaves much to be desired.

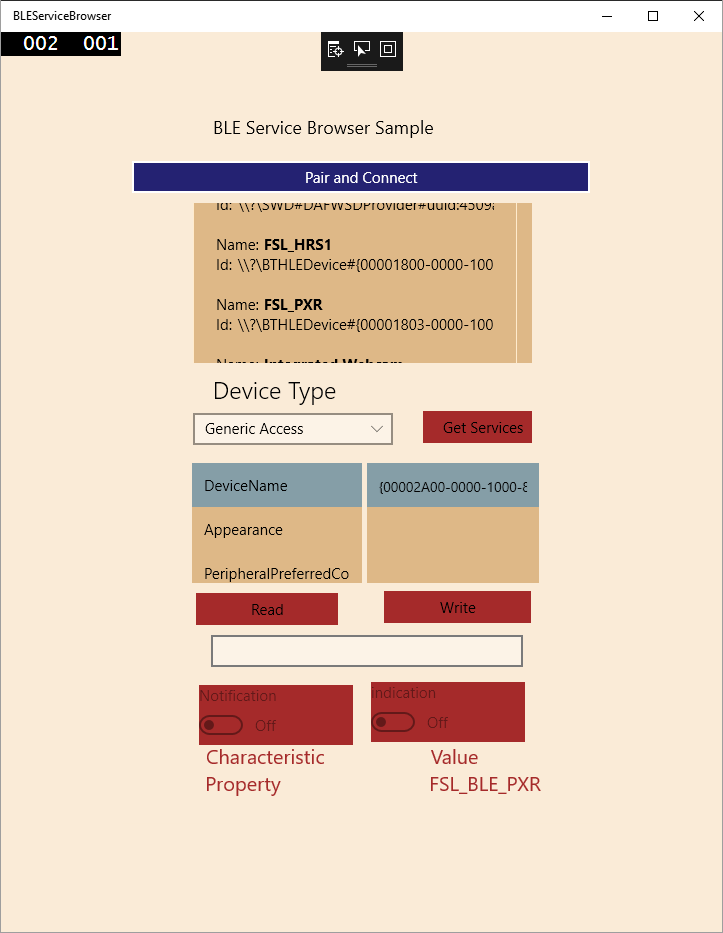
[[](https://habrastorage.org/getpro/geektimes/post_images/fd8/213/747/fd82137474453134c3bd6744473908a8.png)](https://habrastorage.org/getpro/geektimes/post_images/fd8/213/747/fd82137474453134c3bd6744473908a8.png)

Figure 54. Windows-BLEServiceBrowser

Above is a screenshot of the only demo application for Windows — BLEServiceBrowser.

Gateway Smart Starter Kit

Gateway Smart Starter Kit - project gateway BLE devices to the WEB server and the WEB server itself implements the user interface for the network BLE devices. Everything is implemented in the Node environment.js. It is proposed to deploy on a raspberry Pi 2 model B microcomputer with The raspbian Jessie operating system. Directly connecting the Raspberry Pi to BLE devices uses the Bluetooth HCI sockets interface to the L2CAP level and the USB HCI adapter. To run under Windows, you need to install a special replacement for the standard Bluetooth HCI driver. The solution works on a very limited number of hardware adapter types due to the limitations of the HCI driver.

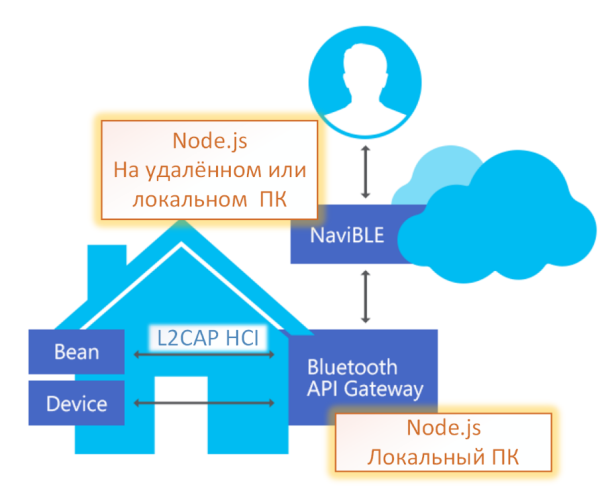


Figure 55. Bluetooth HCI drivers

**12. LAMP WITH PUSH-BUTTON CONTROL**

Parts list

* 1 Arduino Uno Board
* 1 solderless breadboard
* 2 clock buttons
* 1 220 Ohm resistor
* 1 led
* 7 wires " Papa-Papa»

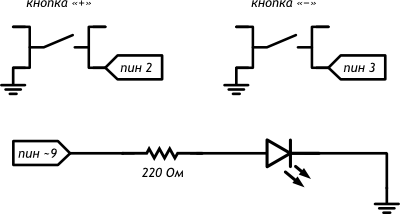


Figure 56. Schematic diagram

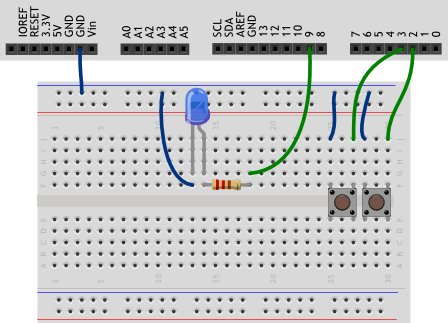


Figure 57. Diagram on the layout

Pay attention

* If you rework the circuit from the previous experiment, please note that this time we need to connect the led to the port that supports PWM.

Sketch

p110\_plus\_minus\_light.ino

#define PLUS\_BUTTON\_PIN 2

#define MINUS\_BUTTON\_PIN 3

#define LED\_PIN 9

int brightness = 100;

boolean plusUp = true;

boolean minusUp = true;

void setup()

{

pinMode(LED\_PIN, OUTPUT);

pinMode(PLUS\_BUTTON\_PIN, INPUT\_PULLUP);

pinMode(MINUS\_BUTTON\_PIN, INPUT\_PULLUP);

}

void loop()

{

analogWrite(LED\_PIN, brightness);

// respond to clicks using a function written by us

plusUp = handleClick(PLUS\_BUTTON\_PIN, plusUp, +35);

minusUp = handleClick(MINUS\_BUTTON\_PIN, minusUp, -35);

}

// Own function with 3 parameters: pin number with button

// (buttonPin), state before check (wasUp), and gradation

// brightness when clicking on the button (delta). The function returns

// (eng. return) back to the new, current state of the button

boolean handleClick(int buttonPin, boolean wasUp, int delta)

{

boolean isUp = digitalRead(buttonPin);

if (wasUp && !isUp) {

delay(10);

isUp = digitalRead(buttonPin);

// if there was a click, change the brightness from 0 to 255

if (!isUp)

brightness = constrain(brightness + delta, 0, 255);

}

return isUp; // return the value back to the calling code

}

Explanations to the code

We can use not only built-in functions, but also create our own. This is reasonable when we need to repeat the same actions in different places of the code, or, for example, we need to perform the same actions on different data, as in this case: to process the signal from digital ports 2 and 3.

* You can define your own functions anywhere in the code outside the code of other functions. In our example, we defined a function after the loop.
* To define a proper function, we need:
* Declare what type of data it will return. In our case, it is a boolean. If the function only performs some actions and does not return any value, use the void keyword
* Assign a name — identifier to the function. The same rules apply here as for naming variables and constants. It is accepted to call functions in the same style as variable.
* list the parameters passed to the function in parentheses, specifying the type of each parameter. This is a Declaration of variables that are visible within the newly created function, and only within it. For example, if we try to access wasUp or isUp from loop() in this experiment, we will get an error message from the compiler. Similarly, variables declared in a loop are not visible to other functions, but their values can be passed as parameters.
* between a pair of curly braces write the code executed by the function
* If a function must return a value, use the return keyword to specify which value to return. This value must be of the type we have declared
* So-called global variables, i.e. variables that can be accessed from any function, are usually declared at the beginning of the program. In our case, it is brightness.
* Inside the handleClick function that we created, the same thing happens as in the "Push-button switch" experiment.
* Because at the brightness increment step 35, after no more than eight consecutive clicks on one of the buttons, the value of the brightness + delta expression will go beyond the interval [0, 255]. Using the constraint function, we limit the allowed values for the brightness variable to the specified interval limits.
* In the expression plusUp = handleClick (PLUS\_BUTTON\_PIN, plusUp, +35) , we refer to the variable plusUp twice. Since = puts the value of the right operand in the left, it is first calculated that handleClick will return. So when we pass plusUp to it as a parameter, it still has the old value calculated from the last call to handleClick.
* Inside handleClick, we calculate a new led brightness value and write it to the global brightness variable, which is simply passed to analogWrite on each loop iteration.

**13. LAMP WITH PUSH-BUTTON CONTROL**

Parts list

* 1 Arduino Uno Board
* 1 solderless breadboard
* 2 clock buttons
* 1 220 Ohm resistor
* 1 led
* 7 wires " Papa-Papa»

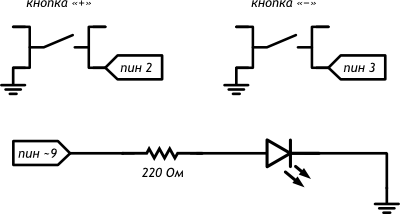


Figure 58. Schematic diagram

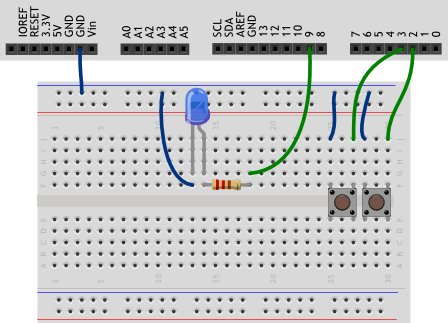


Figure 59. Diagram on the layout

Pay attention

If you rework the circuit from the previous experiment, please note that this time we need to connect the led to the port that supports PWM.

Sketch

p110\_plus\_minus\_light.ino

#define PLUS\_BUTTON\_PIN 2

#define MINUS\_BUTTON\_PIN 3

#define LED\_PIN 9

int brightness = 100;

boolean plusUp = true;

boolean minusUp = true;

void setup()

{

pinMode(LED\_PIN, OUTPUT);

pinMode(PLUS\_BUTTON\_PIN, INPUT\_PULLUP);

pinMode(MINUS\_BUTTON\_PIN, INPUT\_PULLUP);

}

void loop()

{

analogWrite(LED\_PIN, brightness);

// respond to clicks using a function written by us

plusUp = handleClick(PLUS\_BUTTON\_PIN, plusUp, +35);

minusUp = handleClick(MINUS\_BUTTON\_PIN, minusUp, -35);

}

// Own function with 3 parameters: pin number with button

// (buttonPin), state before check (wasUp), and gradation

// brightness when clicking on the button (delta). The function returns

// (eng. return) back to the new, current state of the button

boolean handleClick(int buttonPin, boolean wasUp, int delta)

{

boolean isUp = digitalRead(buttonPin);

if (wasUp && !isUp) {

delay(10);

isUp = digitalRead(buttonPin);

// if there was a click, change the brightness from 0 to 255

if (!isUp)

brightness = constrain(brightness + delta, 0, 255);

}

return isUp; // return the value back to the calling code

}

Explanations to the code

* We can use not only built-in functions, but also create our own. This is reasonable when we need to repeat the same actions in different places of the code, or, for example, we need to perform the same actions on different data, as in this case: to process the signal from digital ports 2 and 3.
* You can define your own functions anywhere in the code outside the code of other functions. In our example, we defined a function after the loop.
* To define a proper function, we need:
* Declare what type of data it will return. In our case, it is a boolean. If the function only performs some actions and does not return any value, use the void keyword
* Assign a name — identifier to the function. The same rules apply here as for naming variables and constants. It is accepted to call functions in the same style as variable.
* list the parameters passed to the function in parentheses, specifying the type of each parameter. This is a Declaration of variables that are visible within the newly created function, and only within it. For example, if we try to access wasUp or isUp from loop() in this experiment, we will get an error message from the compiler. Similarly, variables declared in a loop are not visible to other functions, but their values can be passed as parameters.
* between a pair of curly braces write the code executed by the function

If a function must return a value, use the return keyword to specify which value to return. This value must be of the type we have declared

* So-called global variables, i.e. variables that can be accessed from any function, are usually declared at the beginning of the program. In our case, it is brightness.
* Inside the handleClick function that we created, the same thing happens as in the "Push-button switch" experiment.
* Because at the brightness increment step 35, after no more than eight consecutive clicks on one of the buttons, the value of the brightness + delta expression will go beyond the interval [0, 255]. Using the constraint function, we limit the allowed values for the brightness variable to the specified interval limits.
* In the expression plusUp = handleClick (PLUS\_BUTTON\_PIN, plusUp, +35) , we refer to the variable plusUp twice. Since = puts the value of the right operand in the left, it is first calculated that handleClick will return. So when we pass plusUp to it as a parameter, it still has the old value calculated from the last call to handleClick.
* Inside handleClick, we calculate a new led brightness value and write it to the global brightness variable, which is simply passed to analogWrite on each loop iteration

# 14. ORANGE PI 2G

# https://habrastorage.org/webt/59/ca/c2/59cac247b4fb7547619158.jpeg

Figure 60. Orange Pi 2G

This single-Board for $ 10 has a modest size (67x42mm), contains: Cortex-A5 1.0 GHz, 256MB of RAM, 512MB NAND, and most importantly-built-in WI-FI and 2G adapters!

Download and install SketchUp on your computer

You can download the sketch From the official website of the program – http://sketchup.com.

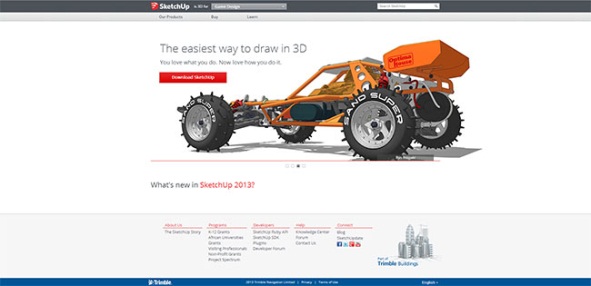


Figure 61. Download SketchUp

To do this, select SketchUp Make from the Our Products menu and then click the red Download SketchUp button.

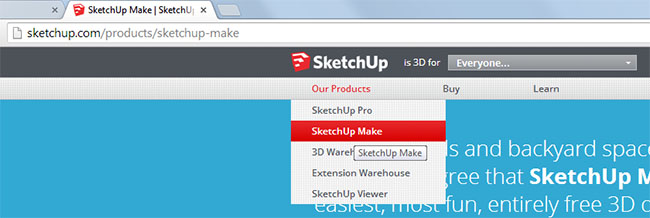


Figure 62. Download SketchUp.

According to the logic of the program should start. You will be asked to specify why you need it at all, enter your contacts, select the operating system, and so on.

And only after filling out the entire form and the next click Download will begin downloading the editor. It weighs a little-72.2 MB (current version for today).

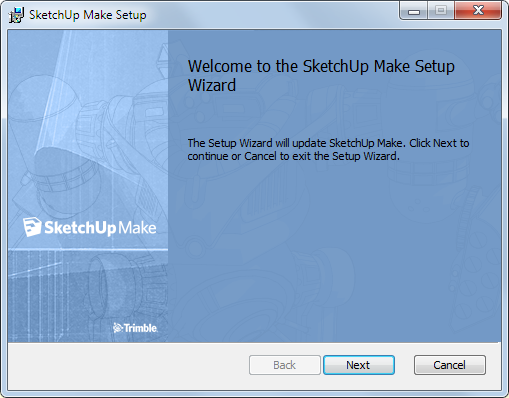


Figure 63. Next

It remains only to press the Next button all the time and put a tick under the license agreement in one place.

Familiarity with the interface

In order to better understand all that I will describe further, I recommend downloading a specially prepared demo file with a simple model of the apartment. Practice on it, so it will be much clearer than just reading an article.

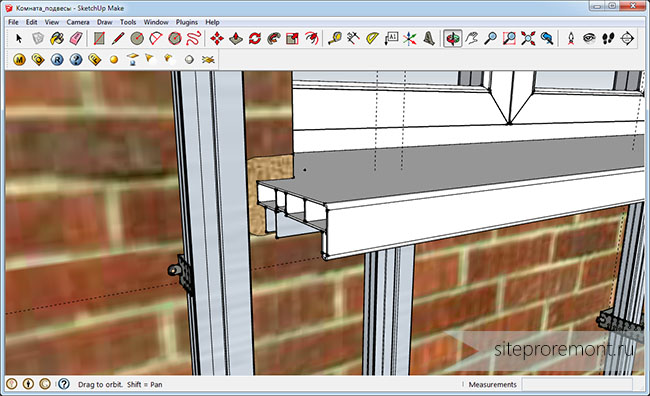


Figure 64. Familiarity with the interface

As you can see, everything is in English. It's not great, but I'll try to explain what's going on. At the top we see the usual text menu. Let's go over his points quickly.

File (File). This tab traditionally contains the items "Create", "Open", "Save", "Import", "Export", "print", "Exit".

Edit (To Edit). In addition, nothing unusual. By going to the tab, you can "Cut", "Copy", "Paste", "Delete", "Select all", "Hide/Unhide" object (Hide/Unhide), "Lock/Unlock" (Lock/Unlock), "Create a component", "Create a group" of objects. These last specific points will be discussed in other lessons; it is too early for us to discuss them.

Edit (To Edit). In addition, nothing unusual. By going to the tab, you can "Cut", "Copy", "Paste", "Delete", "Select all", "Hide/Unhide" object (Hide/Unhide), "Lock/Unlock" (Lock/Unlock), "Create a component", "Create a group" of objects. These last specific points will be discussed in other lessons; it is too early for us to discuss them.

The Toolbars menu is responsible for displaying various editor panels. For example, the drawing panel or the shadow panel. I have only activated the Large Tool Set, since it contains all the main buttons, and Shadow (Shadows). To begin with, these two panels will be enough for you.

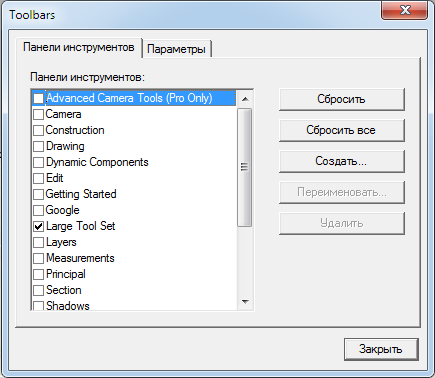


Figure 65. Menu Toolbars

By default, the 3D modeling program displays Hidden geometry, Section Planes & Cuts (I don't know what it is, honestly), Axes, Guides (Auxiliary lines). Hidden Geometry is also an auxiliary line inside the object, which helps to more accurately represent its shape. You can safely remove this jackdaw. Section Planes & Cuts-too. I recommend leaving the axes for a better representation of the volume, as well as the Auxiliary lines.

The Shadows item is responsible for dynamic shading of the model depending on the selected time of day and time of year, which are displayed in the same panel. That is, the program has its own sun, the position of which, as in reality, depends on time. But not from real time, but from the chosen one. Note that on complex models, when you turn on the Shadows, the computer may slow down.

Fog is the fog. By activating this option, you will be able to see close objects well, but not far away. The greater the distance from the camera to the object, the more it will be immersed in the "fog". This reduces the load on the computer.

Edge Style-style of lines (faces). You can choose whether to show them at all, or do only planes (without them really inconvenient). Back Edges (K key) – show all lines, even those that are behind the planes. They will be displayed in dotted lines.

Face Style – style of planes. You can choose to display them: transparent planes, planes without fill, color-filled planes without texture, textured planes (by default), monochrome. Leave it as default.

We will not consider the item Component Edit yet, as well as Animation.

The next tab is Camera. Let's also go through the points.

Previous/Next – the previous / next position of the camera (your eyes on the model, so to speak).

Standart Views-standard views of the 3D model: top, bottom, right, left, etc.

Parallel Projection/Perspective/Two-Point Perspective-select the volume display option. Choose Perspective as the most realistic construction.

The items Orbit, Pan, Zoom, Field of View, Zoom window, Zoom extents, Position camera, Walk, Look around are duplicated on the Large tool set panel

Orbit-rotation of the 3D model around the point on the screen where you fixed the left mouse button. This function is also performed by default by clicking on the mouse wheel. Clamped and move the mouse-the model rotates.

Pan-move the 3D model without rotation. You sort of grab it with your hand and pull.

Zoom – if you select this tool, hold down the left key and start moving the mouse forward/backward, the scale of the model will increase/decrease. The same function is performed by default by scrolling the mouse wheel.

Zoom window-a tool for selecting the area of the model that you want to zoom in. The smaller the area you select, the larger the magnification. That is, the zone you selected will take up the entire screen.

Field of view - not duplicated on the panel. It is difficult to explain humanly what this function does. But if you try it, it changes the field of view. That is, it changes its angle. It is easier to explain with pictures

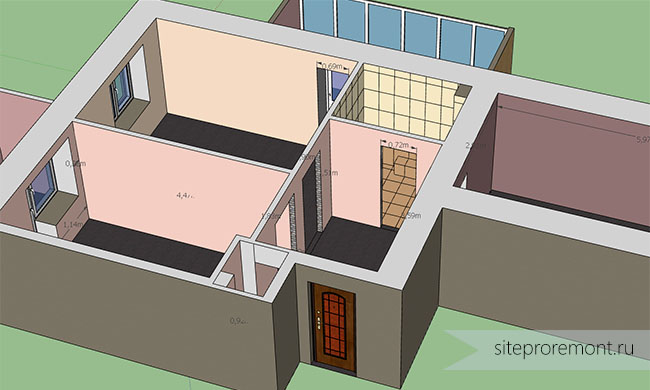


Figure 66. By 15 degree angle

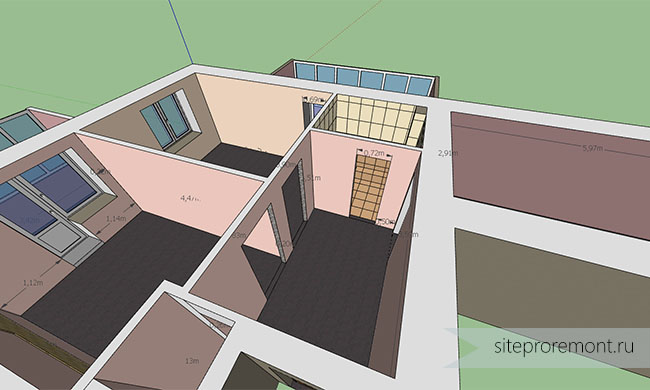


Figure 67. By 80 degrees

How it works: select the tool, hold down the left key and move the mouse forward/backward. You can track the degree value in the status bar in the lower right corner of the program window.

Zoom extents-changes the scale of the model so that it would all fit on the screen.

Position camera-select the camera location on the model. When you select this tool, your cursor will take the form of a decile man, which can be placed anywhere in the model. In our example-in any room. And the subsequent view of the model on the screen will be the view from its "eyes". At the same time, his eyes are always located at a height of 168 cm from the floor. That is, it's like the eyes of a real person. His name is Vasya.

Walk - function "walk" on the model. Let’s say we put our Vasya in the bathroom. By selecting the Walk tool, holding down the left key and moving the mouse, you can make it walk around the virtual apartment.

Look around-model inspection tool. That is, Vasya stands in one place and turns his head.

Everything presented in the Draw tab is duplicated on the Large tool set.

**Line** - you won't believe it, line. Click on the starting point and the end point, a line is built between them.

**Arc** - creating an arc. The arc is built in three clicks: the first two set the beginning and end, the third - the size of the bend.

**Freehand**-freehand line.

**Rectangle** - a rectangle. The first click sets the starting point, the second-the end point on the diagonal.

**Circle** - creates a circle. The first click determines their center, the second-the radius.

**Polygon** is an equilateral polygon. After activating this tool, you will first need to set the number of sides. To do this, simply press the corresponding number on the keyboard and confirm the selection by pressing Enter. Then make the first click on the desired location on the model, thereby setting the geometric center of the figure. The second click sets the radius of the construction described around the polygon.

You can specify the exact dimensions and radius of all the shapes described above. To do this, instead of the last click, just type the desired size on the keyboard and confirm with the Enter key. The values you enter will be displayed in the status bar at the bottom right. For a rectangle, you need to enter two values: length and width, separated by a semicolon. By default, all dimensions in the program are specified in millimeters. (Note, in the demo file for convenience, the dimensions are specified in meters!)

Next, we analyze the Tools menu item. It contains all the main tools for editing 3D model components. All of them are for convenience, again, duplicated in the Large tool set.

Select the selection tool.

Eraser-eraser. By selecting it and clicking on the object, you will make it disappear.

Paintbucket - a tool for filling a plane with a color or texture.

Move - a tool for moving a component.

Rotate-rotates the component around any of the three axes of space.

Scale-changes the size of the component.

Push/Pull - tool for drawing / pulling faces of an object. With its help, it is convenient, for example, to model the skirting Board. Set the shape of the section and pulled out of it a whole plinth. You can pull it back in the same way.

Follow me is the same pull tool, but it follows a pre-selected path, not just a straight line.

Offset-if you can say so, it is a tool that evenly "squeezes" the selected plane beyond its boundaries:

**Рис 68. Offset**

Tape measure – normal roulette. Hold down the left key at the starting point and move the cursor to the end.The length is displayed on the screen in the cursor area.

Protractor-square. Measures the angle in three clicks. The first click is at the top of the corner, the others are on the sides.

Axes-axes. The tool allows you to set your own coordinate system.

Dimensions - tool for drawing dimensions. Works in three clicks. Two extreme points and the distance of the removal of dimensions

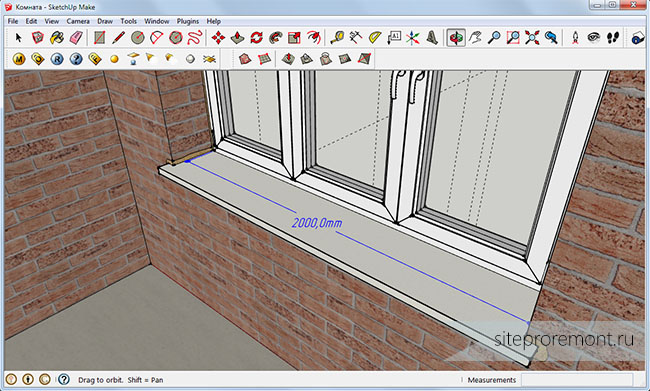


Figure 69. Dimensions

Text is a signature creation tool. By default it immediately outputs the value of the area of the plane or the length of the line:

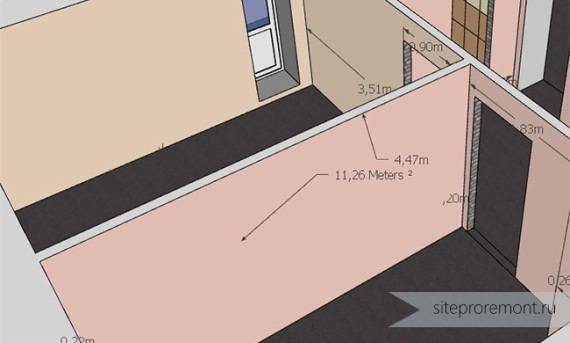


Figure Text

3DText - allows you to place three-dimensional text on the model. For example, I used this tool to count the number of Wallpaper to buy, signed the number of paintings. Here's what 3D text looks like:

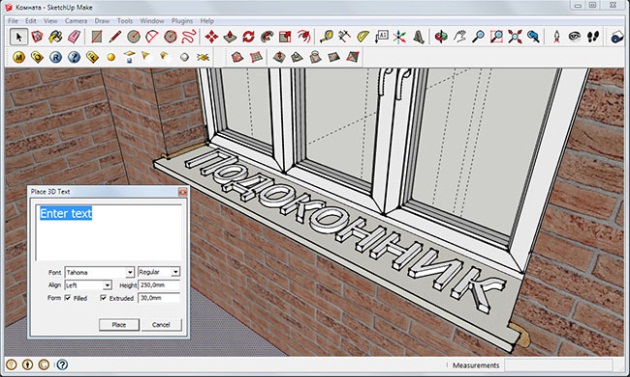


Figure 71. 3DText

There are a few more items in the Tools menu, but we won't look at them, as they are unlikely to be needed. Go to the next menu.

It’s called Window. All items in this menu are responsible for calling a window.

Model info-information about the entire 3D model. Here you can also set its basic properties, such as fonts, their size, how to display dimension lines, etc.

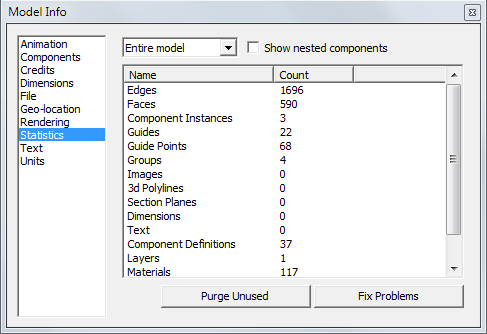


Figure 72. Model info

Entity info-properties of a specific model component. To plane this square, the line length, arc radius and number of segments that can be changed. The fact is that SketchUp by default draws arcs and circles quite roughly, breaking them into a small number of segments. You can use the arc properties to set any number of segments, so your shape will look much better. The Entity info window is also accessible by right-clicking on the object.

Materials-displays all available colors and textures to fill. You can add and edit them.

Components - library of ready-made components. Such as, for example, furniture items, cars, etc. you can download components from the "cloud" storage Warehouse directly to your model. Let’s Say Opel Astra:



Figure 73. Components - library of ready-made components

Styles - allows you to select the display styles of the model. Pencil, marker, different line thickness, and so on. Pure pampering. Layers, Outlines, Scenes we will omit, I do not use them.

Shadows-opens the window for managing the time of day and season of the year. In this way, you can control the "natural" lighting in the model.

Fog-enables " fog " settings. The very function of the fog has already been described.

Match photo)

Soften Edges-softens the edges, or rather creates the illusion of softened edges.

Instructor-calling SketchUp help via the Internet. Only it's in English…

Preferences – the main settings of the program, such as the time of AutoSave of the model, setting the quality of graphics, selecting a template for displaying the model, etc.

Extension Warehouse is an online library of extensions for SketchUp. What is there just is not, it is better for you to dig yourself and try something, but at a later stage of development of the 3D editor.

**15. «MODELING. CREATING A 3D MODEL IN SKETCH UP»**

Today we will look at creating a model of a building (house).

The model today will be built on the principle of "Quick Start", as it is called in English-language manuals – a specific simple example of modeling from scratch to save the finished result demonstrates a minimum of tools and techniques.

First, let's look at the drawing area of the program-we see the axes of the 3-dimensional space of the scene-red, green and blue, which corresponds to the traditional designations X, Y, Z. Their intersection, of course – is the origin of coordinates (coordinate origin). So far, it is only important for us to understand that we will be guided in the three-dimensional space of the scene, relying on these axes and planes formed by each pair of axes.

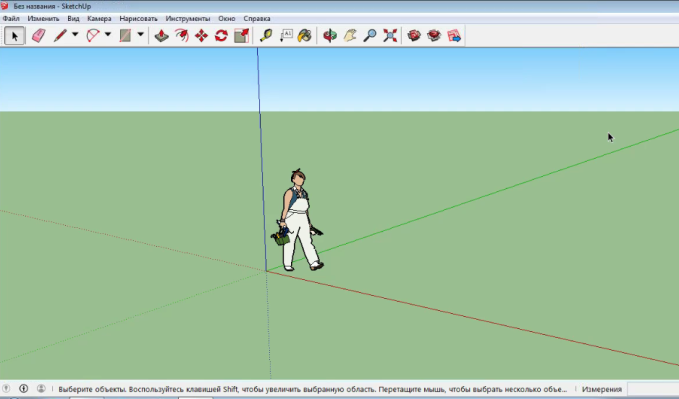


Figure 74.Coordinate origin

To view the scene of constantly used tools in this group also have the appropriate buttons, but in practice it is much faster and more convenient to use similar options, using the middle key (wheel) of the three-button mouse. In this case, it is possible to manage the review options without leaving the tool that is currently activated, which means that we get rid of unnecessary mouse clicks:

hello_html_25b323e8.gif *Rotation (Orbit) - press and hold, move*;

hello_html_5c3a2f31.gif *Pan-Shift + press and hold to move ;*

hello_html_m7cc03da5.gif *Magnifier (Zoom) - rotate the wheel: from itself – increase, on itself-decrease.*

We start working in the perspective view of Iso, and check that the menu Samegavklyuchena option Perspective.

Also for viewing the scene-view control. We will constantly view the objects of the scene as if through the viewfinder of the camera, aiming it at objects from different sides, moving, approaching and moving away. To do this, use a group of special review tools – click on the corresponding button in the toolbar, hold the left mouse button and act:

hello_html_m706963e2.gif *Zoom Window-the part of the drawing area indicated by a rectangular window is enlarged to the full screen*;

hello_html_m8b1f8cb.gif *Show all (Zoom Extends) - shows all objects in the scene*;

hello_html_m8cb4385.gif *Previous, next view – shows the views selected before and after the current view*

***Let our first object be something extremely simple, such as a house. Let's start with the Foundation of the house.***

hello_html_74f0458d.gif*Прямоугольник (Rectangle)*. We build a rectangular area on the plane.

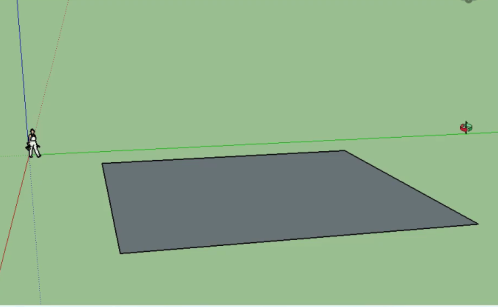


Figure 75. Rectangle

Now let's transform a flat 2D shape into a 3D object. To do this, use the main инструмент из hello_html_68308f00.gif*(Push/Pull)*. Place it on the surface of the square (it will "light up" with points), hold the mouse button, and pull up (note that this option works strictly in the direction of the axes, we have-blue), and let go in the right place.

hello_html_m4c54d987.gif *(Line)*draw the details of the roof, click at the starting point > pull > click at the end point (this principle works almost all tools). The first line is built. Building a second line.

hello_html_68308f00.gif pull out the base of the roof and build the side lines of the roof

hello_html_m4c54d987.gif*Pencil,. Closed lines become a plane*

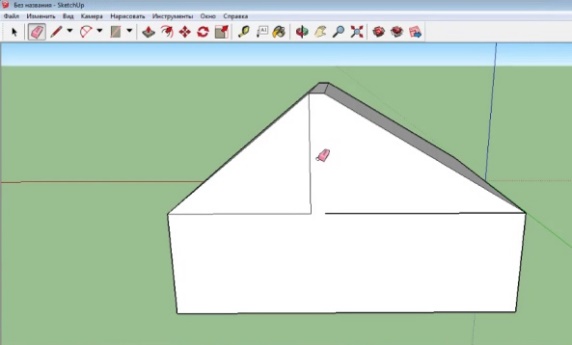
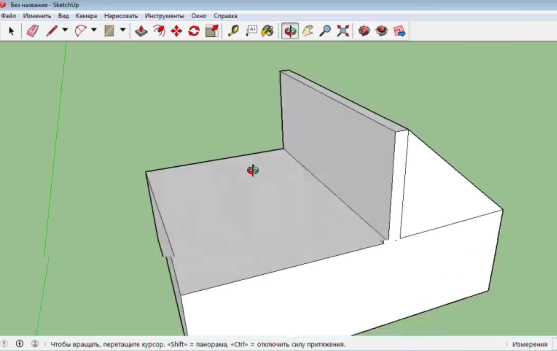


Figure 76. Closed lines become a plane

At any time, we can then press Esc to exit the tool, or immediately continue to consistently draw the remaining parts of the roof. Immediately pay attention to the first important point of working in SketchUp – any actions appear hints about the tool used (in the status bar at the bottom of the working window). In addition, the movement of the cursor relative to the axes is "commented" – in our case, the corresponding color of the line (green or red) and the position (dotted lines) relative to neighboring objects, and we also see text hints next to the cursor and on the control (end, middle) points of edges and surfaces of construction elements.

As soon as we complete the construction of the sides of the rectangle (close it), the space inside is "filled" with the surface.

hello_html_m12487d41.png *Eraser. Excess lines are removed with the tool*

Now we go to fill the roof, to do this, select the Fill tool hello_html_4dd2b9fd.png *Fill, this tool allows you to choose not just a color, but allows you to choose the material of the roof, tiles, apply the fill*.

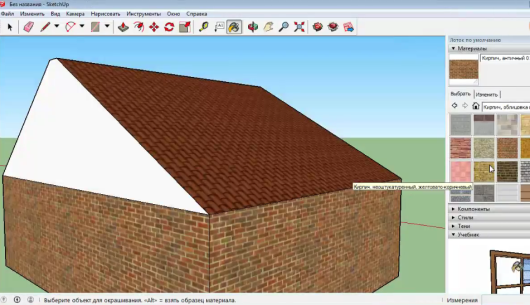
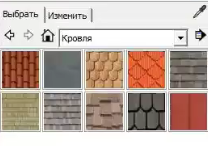


Figure 77.Apply the fill to the walls of the house

In order to build a window, select the pencil tool on the toolbar hello_html_m4c54d987.gif *Карандаш,* прорисовываем детали окна. Инструментом

hello_html_68308f00.gif*Push/Pull*

Описание: hello_html_m12487d41.png eraser

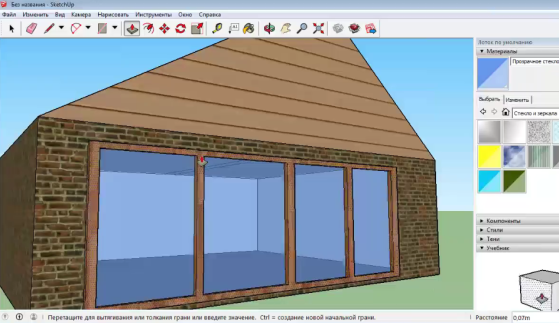
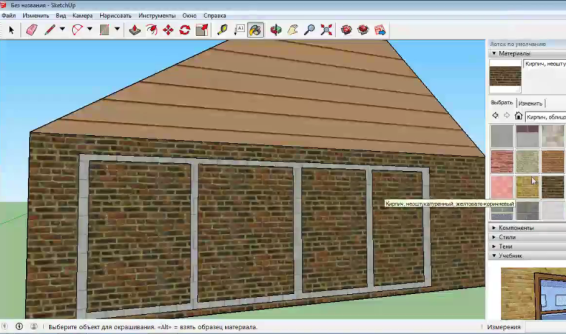


Figure 78. Casting

We use another tool for moving modifications – hello_html_m2ffeed22.gif*Moving( Move),* which, in addition to its direct purpose, can also duplicate the object in a new place – of course, to build a second window, the most rational way is to duplicate the already finished window. Click on the window object, hold down the Ctrl key, and move the resulting copy to the desired location. Do not forget to control the movement parallel to the axis (for example, green), focusing on the hint-a dotted line of the corresponding color and a text hint.

Along the way, let's consider the options for selecting individual objects (in our case, groups) with the Select tool. After all, with all the elementary options, it is used so often that rational methods of its application can significantly speed up the work as a whole. So: single-click selection on the object and multiple-consecutive clicks on objects with the Shift key held down standard and do not require explanations. But the choice of "stretching rectangle" has very convenient options – if we build it from left to right, then we choose what completely falls inside, and if from right to left-everything that intersects any of the sides of the rectangle.

We hope that the General idea of the principle of work appeared. At the same time, we used only a few of the most "running" tools. Naturally, as with any other editor, it is possible to use a different sequence of actions, tools, construction techniques, etc.to achieve the same result. And in the end, of course, we save our construction in the "native" format of the program \*.skp. (File > Save) and at the same time we can export, having chosen a spectacular angle, in any public raster format, for example JPG (File > Export > 2D Graphic).

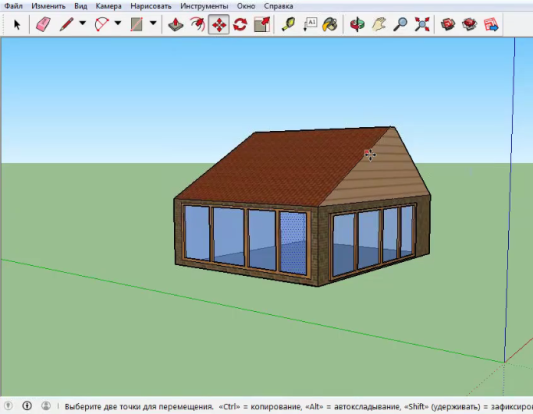


Figure 79. *File > Export > 2D Graphic*.