# Arduino Basics Workshop

Written by Mark Webster

In this workshop learn what is a microcontroller and learn how to upload a simple program to an Arduino microcontroller.

## Microcontrollers and Arduinos

Arduinos are a popular, easy to use family of microcontrollers. A microcontroller is a little single chip computer which has input and output pins for digital and analog communication with the outside world. A microcontroller has memory and a CPU, and can run simple computer programs. Unlike a regular laptop or desktop computer, it does not have an operating system like Windows, iOS, or Android. An Arduino Uno has an ATmega328P microcontroller chip with some additional support chips around it for voltage control, quartz oscillator, leds, a reset button, and USB communication. The advantages of a microcontroller are they are low power and run just one small computer program continuously. With no operating system to boot up, they start running almost immediately on power up. For example, they might control the ABS brakes in a car, or the buttons and display on a microwave oven.

Arduinos have been around since 2005 which is a long time in computer years. Many variations of the original Arduino have been released. Also, several other computer manufacturers have produced microcontrollers which compete with the Arduino that are smaller or faster or have additional features like wifi or bluetooth communication.


(Images from Wikipedia, Arduino Article)

Since the Arduino hardware is open source, many manufacturers have made copies or clones of the original Arduino boards. Sometimes the clones are better than the original, and are much less expensive. Check user reviews before purchasing a clone board since quality control varies.

### Inline Question: Arduino vs Laptop

What is the difference between an Arduino and a regular laptop computer?

## Arduino Shields

The Arduino hardware can be extended with shields and modules. A shield connects on top of an Arduino, usually an Arduino Uno, and gives additional functionality.

Multiple shields can be stacked. In this example the top shield contains a solderless breadboard.

Dragino Lora Shield allows the user to send data and reach extremely long ranges at low data-rates.

Screw-terminal breakout shield in a wing-type format

Adafruit Motor Shield with screw terminals for connection to motors

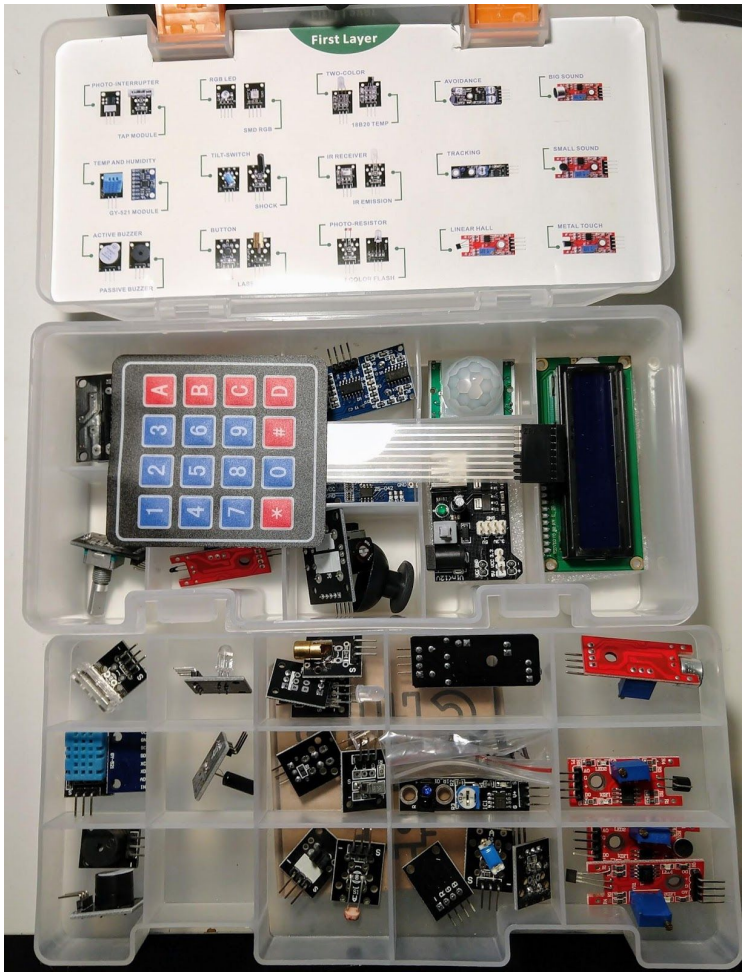

Adafruit Datalogging Shield with a Secure Digital (SD) card slot and real-time clock (RTC) chip

(Images from Wikipedia, Arduino Article)

For example, the data logging shield provides an SD holder and a real time clock. The combination of these two features makes it easy to use an Arduino for gathering real time sensor data, with a time stamp, and saving it in a format that can be read by a laptop computer later.

**Arduino Modules**

Unlike shields which sit on top of an Arduino Uno, modules are designed to connect with the Arduino using jumper wires. Modules can be sensors or motor controllers. Many sensors also connect with jumper wires.



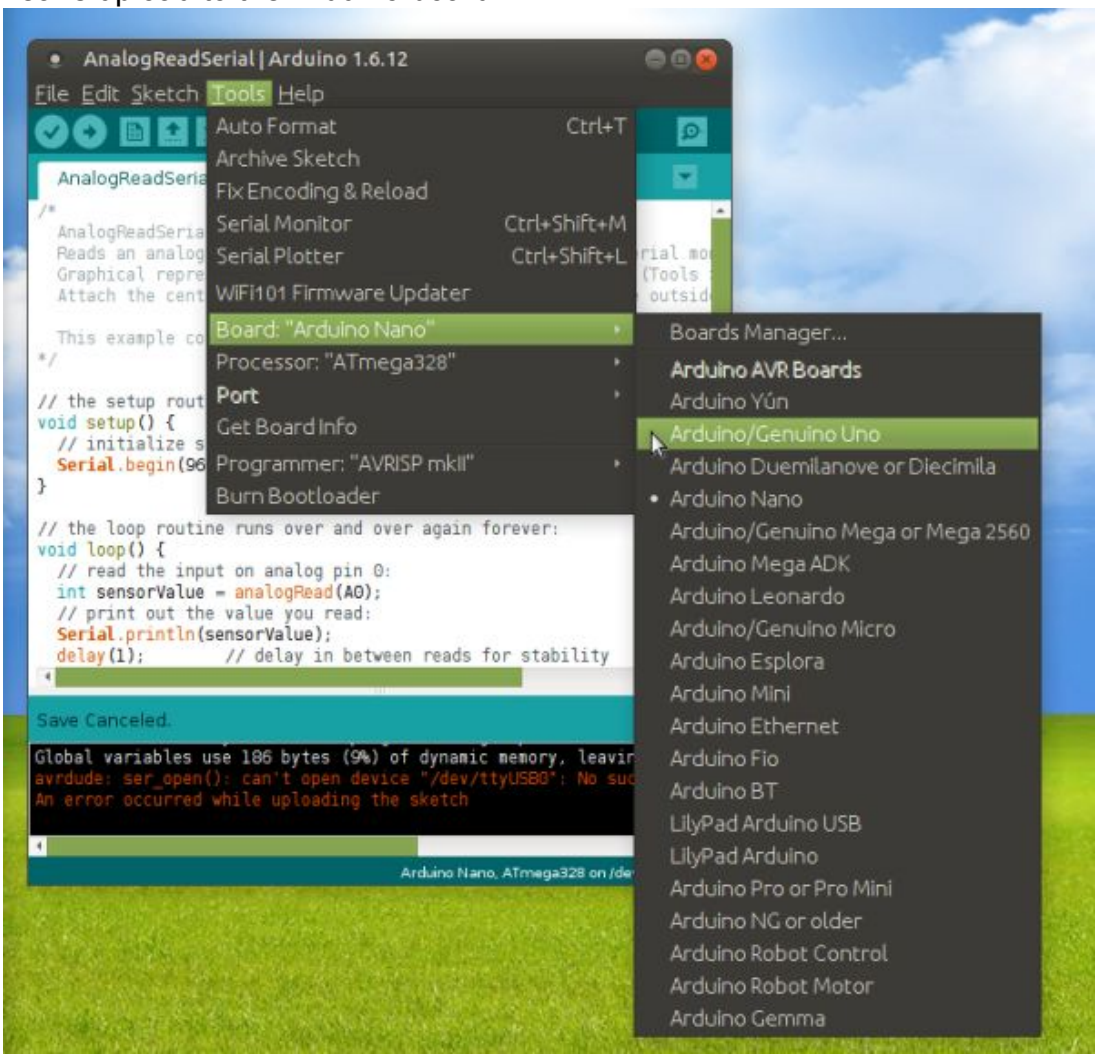

Sensor kit with many kinds of sensor modules.

## Arduino IDE

What made the Arduino so appealing to Makers and artists was the simple software, the Arduino IDE. It is written in Java and is thus cross platform so it can be run on Windows, Macintosh, or Linux. Since it is open source anyone can read and contribute to the source code for the Arduino IDE.
The icon on the desktop looks something  like:

When the Arduino IDE program is launched it looks like:

The check-mark menu icon at the top left means compile the program. The right arrow icon means upload to the Arduino board.




The top menus let you load or save programs, add libraries, select a board type, and select a serial port to use.

In this example screen capture, on the Tools menu the Board selection sub-menu is selected. The standard Arduino boards are shown here. If you have non-Arduino boards installed using the Board Manager, you might see Attiny, ESP32, Teensy 3.6, or something else.

An Arduino program or “sketch” has two main parts: a setup() function which is executed once when the Arduino starts, and a loop() function which runs forever and has the main part of the code.

The bottom of the IDE window gives error messages and information about memory usage.

At the very bottom is lists the Arduino type being used and the serial port being used.

Arduino programs or “sketches” usually are written in the C programming language, but can use C++ if desired.

Download the Arduino IDE for free from: https://www.arduino.cc/en/main/software

Inline Task: Launch Arduino IDE

Find the desktop icon, search for the program, or double click on a *.ino file and launch the Arduino IDE.

## C Programming

C programs have the same basic components as all computer programming languages.

1) There is a way to add comments. In C  if a line starts with a // then the entire line is a comment. All lines between /* and */ will be treated as comments.

2) Values can be stored in a named location in memory called a variable.
In the C language, variables must be given a type, meaning you define what kind of data will be stored in that memory location. It can be an Integer, Float, Char, Byte, or array of these data types. In C, variable names are case sensitive. avalue and AVALUE are different variables.

3) There is a way to make logical decisions or change which parts of the program are executed based on what data is stored in the variable.

```
If( height < 60 ) {
   // the person is short
} else {
  // the person is normal
}
```

4) All programs have a way to loop, or to repeat some part of the program a specified number of times.

```
for( I = 0; I<20; I++ ) {
   // this stuff is repeated 20 times
}
```

or

```
I = 0;
while( I<20 ) {
    // Do some stuff
    // increment I
    I++;
}
```

Although C has these basic 4 features all programming languages share, it does have many advanced features related to bit operations and input/output.

## Arduino Language Extensions

The Arduino IDE extends the standard C language with commands unique to programming Arduino microcontrollers. The cheat sheet covers all of the features. Here are a few common ones:

```
Serial.begin( speed_to_use ); // in the setup() function, start up the serial communication port
Serial.println("This string is sent out the USB serial port.");
        // The USB serial port can be viewed using the Serial Monitor on the tools menu
incoming_byte = Serial.read();   // read a byte from the serial port

data_value = analogRead( pin_to_read );          // read value from an analog input pin
analogWrite( pin_to_write, value_to_write);  // write a pulse width to a PWM pin
data_value = digitalRead( pin_to_read );  // Read a 1 (HIGH) or 0 (LOW) from a digital pin
digitalWrite( pin_to_write, high_or_low_value);  // Write a 1 or 0 to a digital pin

delay( milliseconds_to_pause );  // have the program wait for a few milliseconds

#include <library_to_use>
// include a library to add functionality, such as <Servo.h> to control servo motors
```

There are many more subtleties to the Arduino language extensions that can be explored in books, blogs, and YouTube videos.

Something which helped make the Arduino so popular is all the example programs to be found on the web or other sources. For instance, built into the Arduino IDE and into most libraries are example programs which provide a functioning example of different capabilities.

Inline Question: How can you learn the C or C++ language?

(Many possible answer… Online tutorials, book, open source PDF book, workshops, CIS course)

## How the IDE Compiles Programs

Behind the scenes, the Arduino IDE does several tasks every time the user clicks compile and upload.

First, the IDE adds several standard libraries and a main program to call the setup() and loop() functions.

The IDE also finds and includes any extra libraries #include in the user's program.

Then the IDE cross compiles all the C / C++ code down to byte code for the specific board being used. It uses something like the avr-gcc program to do this. This of course is board specific.

The IDE then uploads the compiled code to the Arduino board typically using the avrdude program.

Any error messages are displayed in the IDE status window at the bottom.

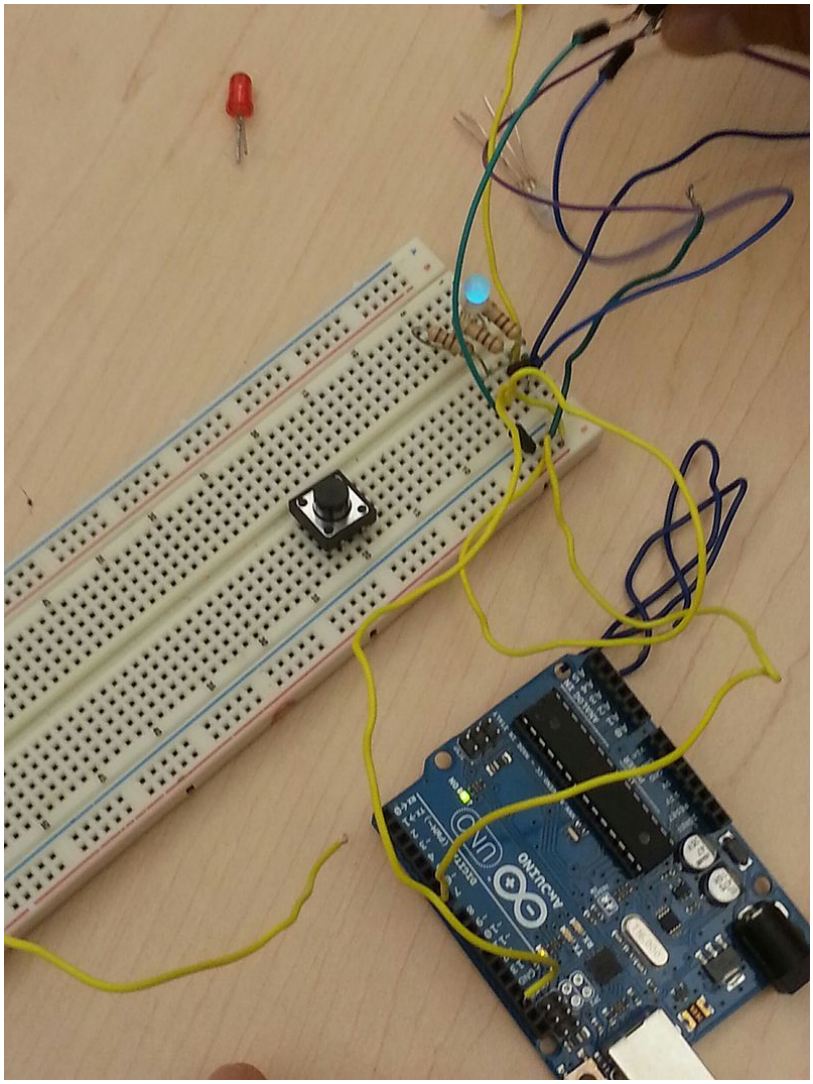
Inline Question: What icon(s) or keyboard shortcut(s) will compile the Arduino program?

Two icons. Sketch menu also has compile and upload options, and lists the shortcuts.

## Create the Circuit

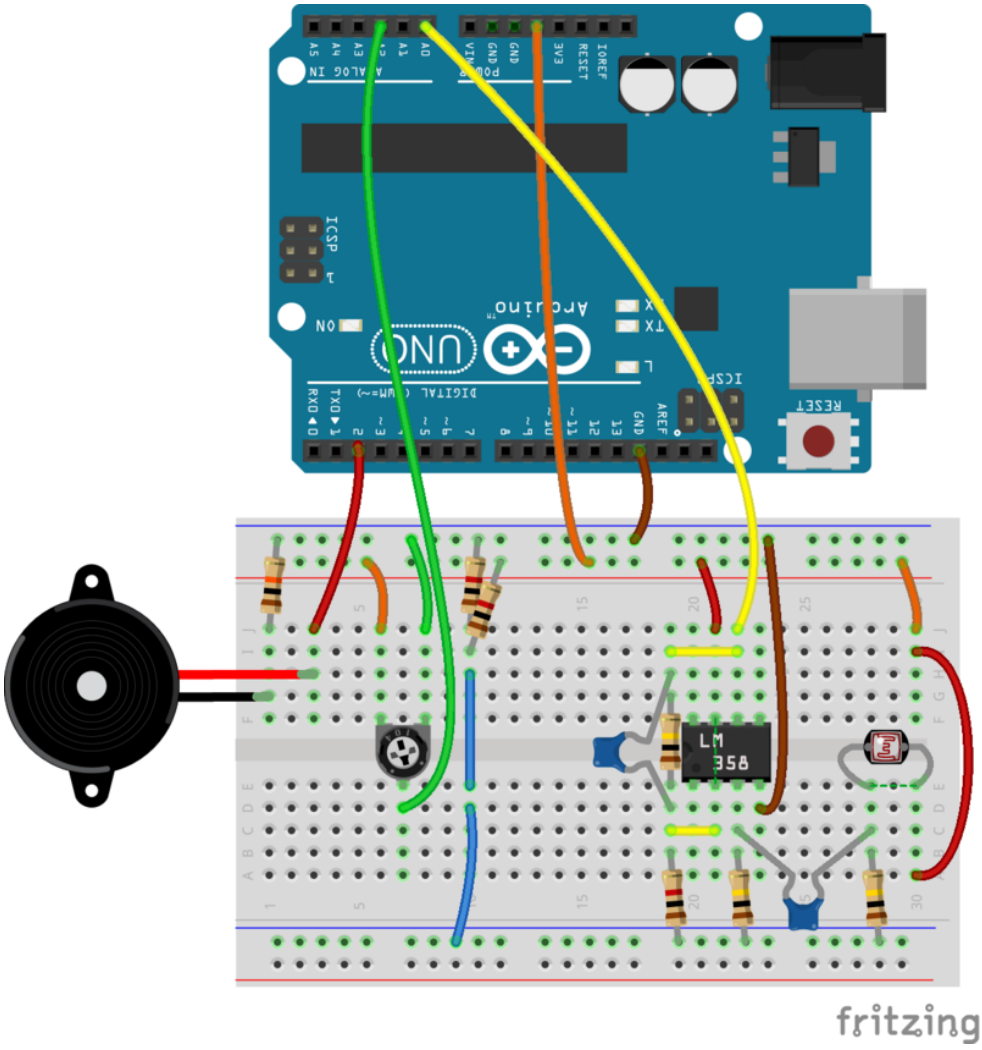
The fun begins when the Arduino is connected to some electronics devices such as sensors, displays, or motors. Typically, these devices combined into a circuit that is prototyped on a breadboard with jumper wires.

The picture below is of a pushbutton circuit on a breadboard, connected to an arduino uno clone.



By NathanMinnick58 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=35728553

Frequently, instead of a photograph of the circuit, users will usually create a diagram of the circuit, often using the Fritzing open source software.




(Image from Wikimedia Commons, By Felix Sempe - Own work, CC BY-SA 4.0,)

Inline Task: Look at breadboard and wires. Connect wires to an Arduino.

## The Project Development Process

1) Have an idea that solves some problem or looks cool

2) Research any similar projects on the Internet or in books

3) Make a block diagram of what components are required for the project and their connections

4) Identify and purchase components that will complete the block diagram

5) Prepare to assemble the hardware and circuits to see the total physical layout. This may involve 3D printing cases or making chassis.

6) Write the code and assemble the hardware in small testable chunks called units. Debug each unit.

7) Connect all the units and assemble the completed project. It may mean transferring the breadboard circuit to a more permanent construction method.

8) Test, debug and show off the completed project to friends! You may want to share your project on the internet or at Maker Faires to other Arduino enthusiasts.

Typically, 1/3 of the total project time is spent designing, 1/3 is spent making the prototype, and 1/3 is spent debugging the many unexpected problems.

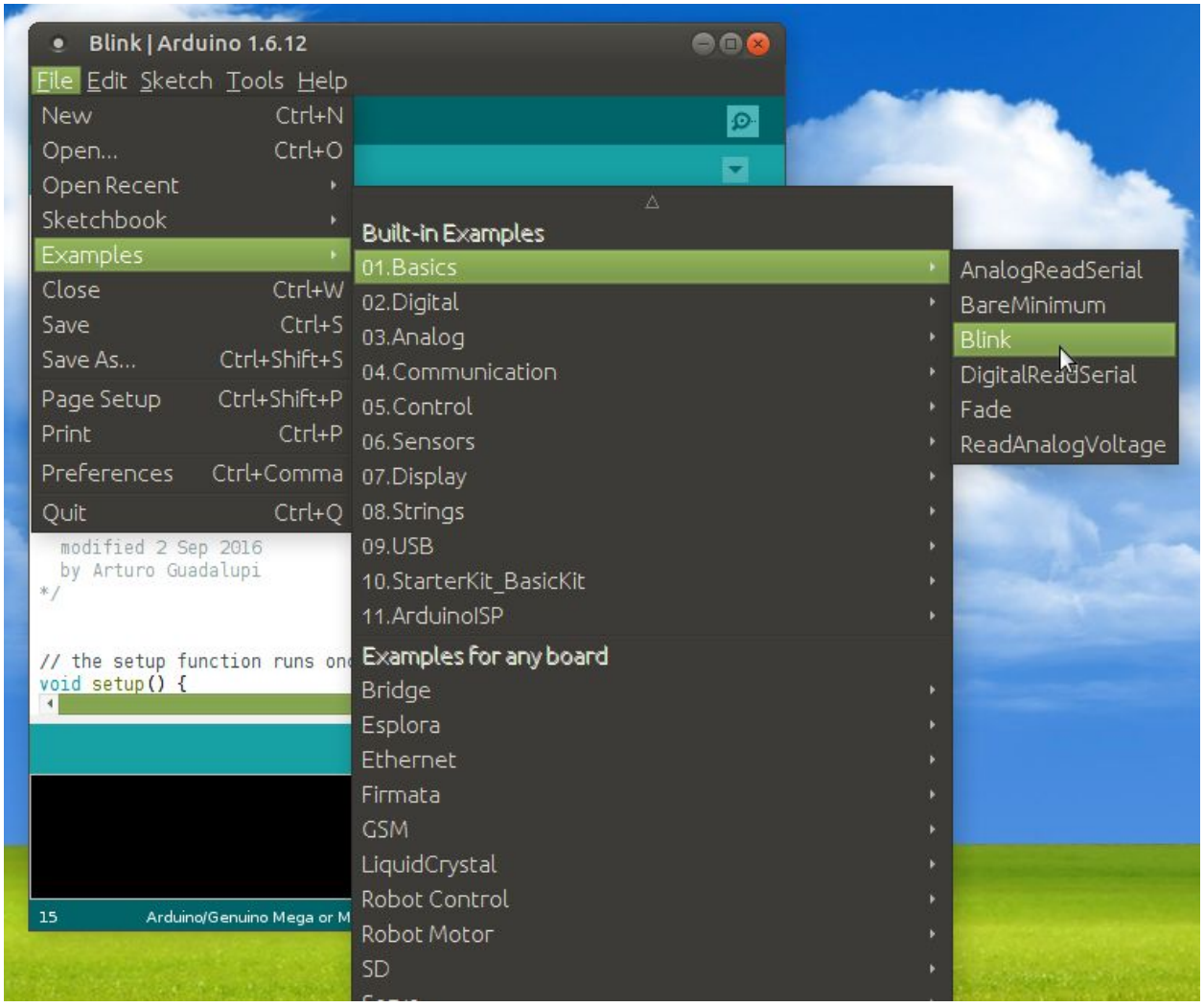
### Inline Task: Make up an Arduino project and describe the steps to create it.

(Invent an example) (One example: Gardening bot to automatically water a plant when it is dry. Search the internet to find similar projects. The modules are sensor, water pump, the physical framework to hold everything, and the plant. Need to buy the parts, write code,  etc.
Make a guess about the time needed.)

## Examples:

### Example 1:

Open the Arduino IDE and open the Blink example sketch.




Connect an Arduino Uno to the computer. Compile the Blink program. Download and verify the built-in LED is blinking.
Notice the messages in the debug window at the bottom of the Arduino IDE describing how much of different types of memory was used by the sketch.

### Example 2:

After downloading the Blink program, disconnect the Arduino from the laptop or Raspberry Pi. Connect a 9V battery to the Arduino power jack. Notice the light starts blinking. All programs downloaded to the Arduino are permanently stored in flash memory, and will run endlessly whenever the Arduino is given power.

### Example 3:

Hardware: Breadboard, red led, 330 ohm resistor, two wires
Connect an led and resistor to digtal output pin 9. Modify the BLINK sketch to use pin 9 instead of LED_BUILTIN.

## Example 4:

Connect a potentiometer between +5 volts and ground. The positive voltage and ground are both pins on the Arduino. The middle pin of the potentiometer is connected to analog input pin A0 on the Arduino.
Load and download the example sketch Examples -> Basics -> AnalogReadSerial.
Open the Tools menu “Serial Monitor” and look at the values as the potentiometer is turned.
Open the Tools menu “Serial Plotter” and look at a graph of the values as the potentiometer is turned.