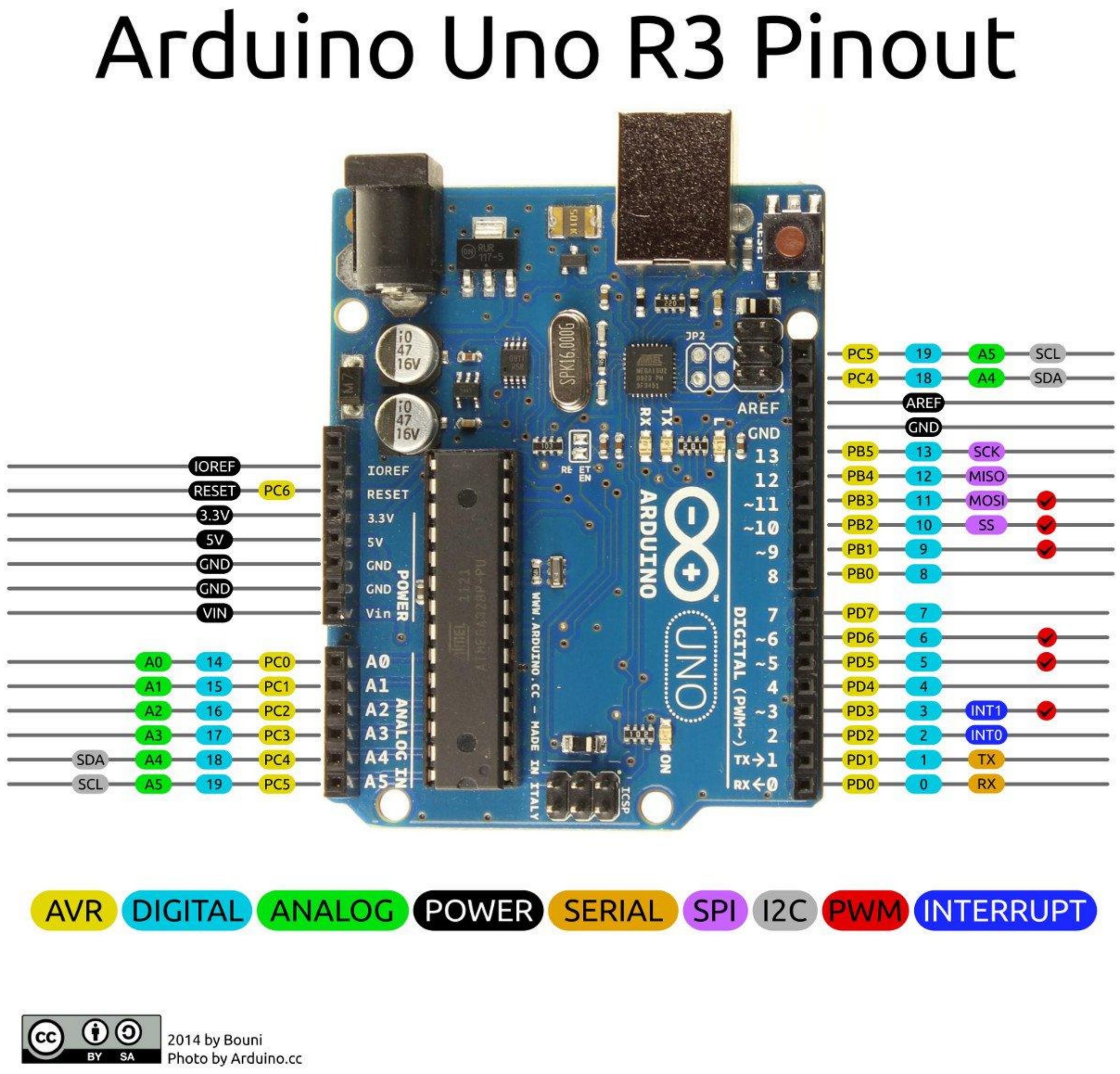
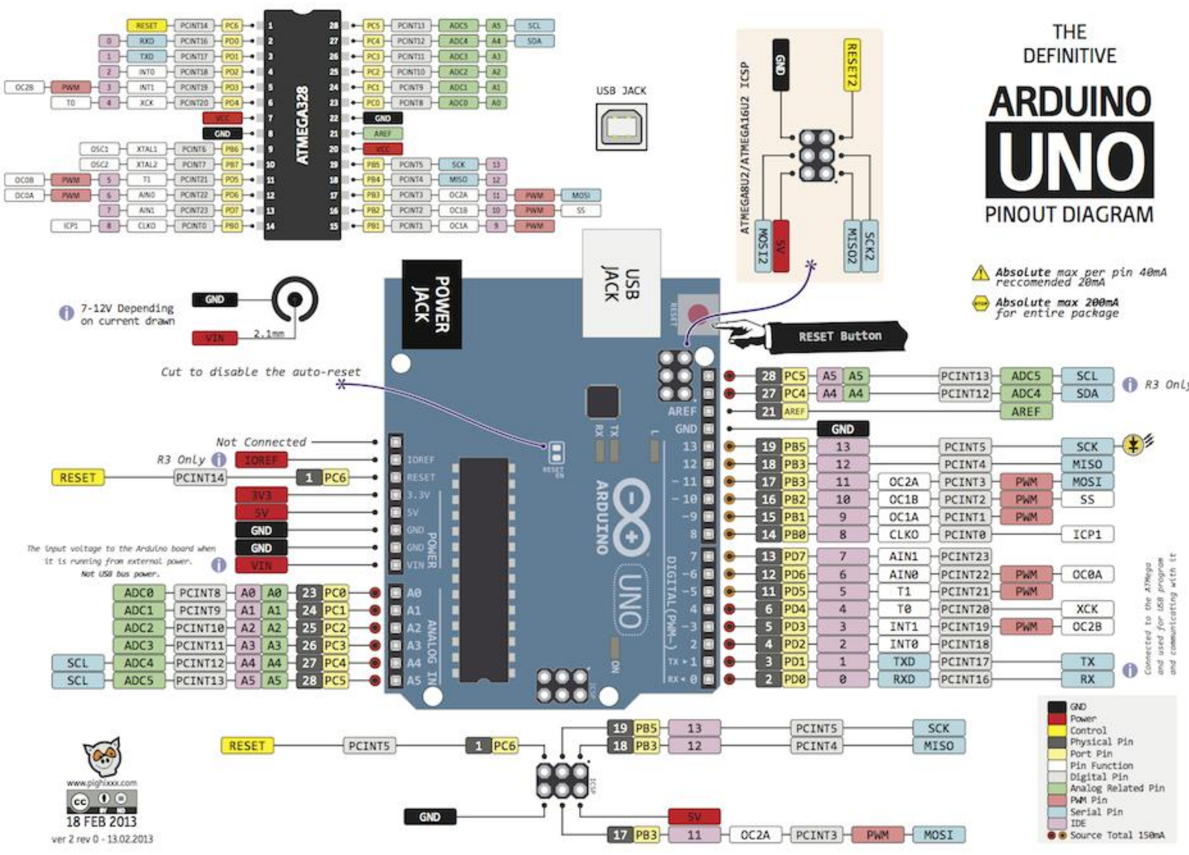
Why Arduino?

- **Inexpensive** - Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and even the pre-assembled Arduino modules cost less than $50
- **Cross-platform** - The Arduino Software (IDE) runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller systems are limited to Windows. Simple, clear programming environment - The Arduino Software (IDE) is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. For teachers, it's conveniently based on the 2rocessing programming environment, so students learning to program in that environment will be familiar with how the Arduino IDE works.
- **Open source and extensible software** - The Arduino software is published as open source tools, available for extension by experienced programmers. The language can be expanded through C++ libraries, and people wanting to understand the technical details can make the leap from Arduino to the AVR C programming language on which it's based. Similarly, you can add AVR-C code directly into your Arduino programs if you want to.
- **Open source and extensible hardware** - The plans of the Arduino boards are published under a Creative Commons license, so experienced circuit designers can make their own version of the module, extending it and improving it. Even relatively inexperienced users can build the breadboard version of the module in order to understand how it works and save money.

Arduino UNO R3

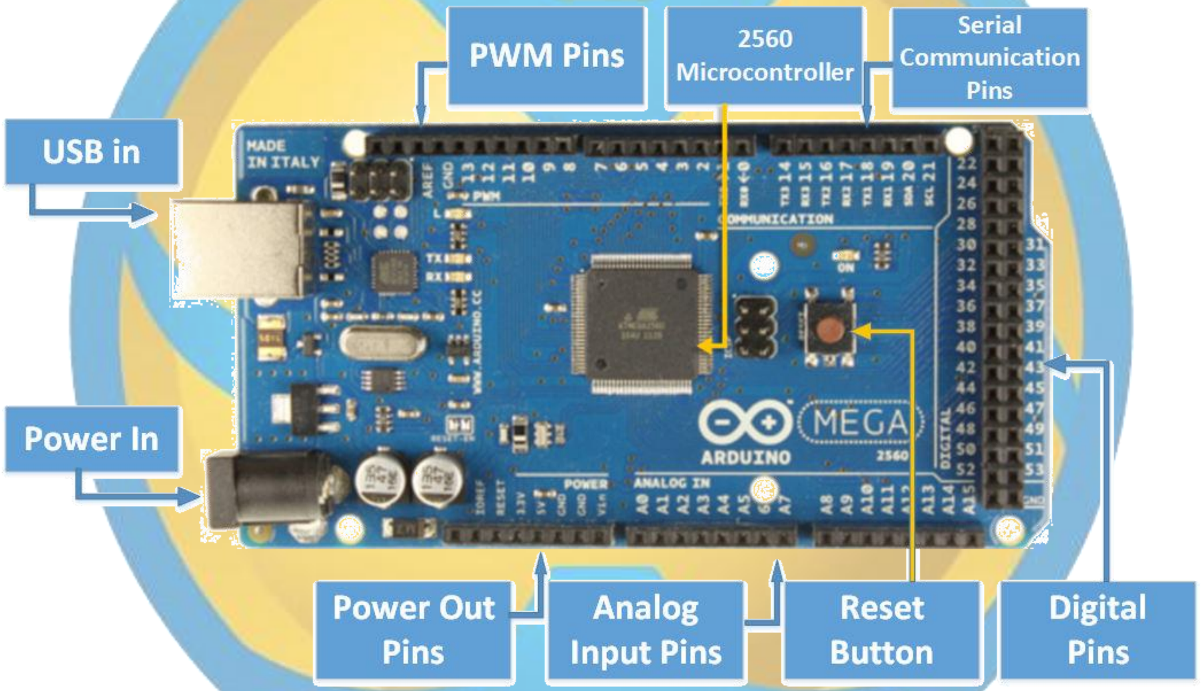
Arduino/Genuino Uno is a microcontroller board based on the ATmega328P . It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.. You can tinker with your UNO without worring too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.




THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK
ATMEGA8U2/ATMEGA16U2 ICSP
GND
RESET2
MOSI2
5V
MISO2
SCK2
Absolute max per pin 40mA reccomended 20mA
Absolute max 200mA for entire package
POWER JACK
7-12V Depending on current drawn
GND
VIN
2.1mm
RESET Button
Cut to disable the auto-reset
Not Connected
R3 Only
IOREF
RESET
PCINT14
PC6
3V3
5V
GND
VIN
The input voltage to the Arduino board when it is running from external power. Not USB bus power.
ADC0
ADC1
ADC2
ADC3
ADC4
ADC5
SCL
SDA
PCINT8
PCINT9
PCINT10
PCINT11
PCINT12
PCINT13
A0
A1
A2
A3
A4
A5
PC0
PC1
PC2
PC3
PC4
PC5
AREF
PB5
PB4
PB3
PB2
PB1
PB0
PD7
PD6
PD5
PD4
PD3
PD2
PD1
PD0
SCK
MISO
MOSI
SS
ICP1
OC2A
OC1B
OC1A
CLKO
AIN1
AIN0
T1
T0
INT1
INT0
TXD
RXD
PWM
OC0A
OC0B
XCK
OC2B
TX
RX
PCINT0
PCINT1
PCINT2
PCINT3
PCINT4
PCINT5
PCINT16
PCINT17
PCINT18
PCINT19
PCINT20
PCINT21
PCINT22
PCINT23
Connected to the ATMega and used for USB programming and communicating with it
ARDUINO UNO
DIGITAL (PWM~)
ANALOG IN
POWER
ICSP
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA
www.pighixxx.com
18 FEB 2013
ver 2 rev 0 - 13.02.2013

Arduino MEGA 2560

The Arduino/Genuino Mega 2560 is a microcontroller board based on the ATmega2560. It has 54 digital input/output pins (of which 15 can be used as 2WM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICS2 header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega 2560 board is compatible with most shields designed for Arduino/Genuino Uno and the former boards Duemilanove or Diecimila.



Arduino Shields

To extend the capabilities of Arduino board, addons namely shields are used. For convenience, these shields can be divided into four groups.

Utilitarian Shields – *Any shield that extends the utilities of the Arduino by adding better I/O capabilities or power management is included in this group.*

The Base Shield

Modeled on Grove platform for Arduino, the Base Shield extends the headers on the Arduino into a set of Grove connectors, allowing full compatibility with all Grove modules. This opens up a huge variety of I/O possibilities, in a neat and easy to use package!

The Energy Shield

The energy shield gives so much flexibility when it comes to powering an Arduino project. It can accept a huge variety of power sources and interface them safely with your Arduino. It can also accept multiple sources at once and be used to charge mobile devices! 2ower options are now limitless!

The Motor Shield

The motor shield provides the ideal interface between an Arduino and multiple motors, containing all necessary on-board regulation, switching circuitry, and even a self-healing fuse. Everything needed to protect your precious motors and provide easy control!

The Relay Shield

To control high power devices, you need switches capable of dealing with that power. The relay shield provides four fully insulated and high power switches that are easily controllable via the Arduino meaning you can control most high power devices!

The Screw Shield

Sometimes you need the ability to easily have access to all the pins on the Arduino. The Screw Shield allows just that, by providing a screw terminal connection for each pin on the Arduino. It's also fully stackable, so you can easily access pins and use them simultaneously!

The SD Card Shield

Memory storage can add so much to a project, so whether you're logging data, or using the Arduino to stream it, this shield is the perfect choice! It's compatible with SD, SDHC and MicroSD cards, and uses a simple S2I interface, leaving you with plenty of room for expansion!

Communication – *Any shield that sends or receives data is included in this group.*

The NOC Shield

The NFC shield allows you to harness the power of Near Field Communication, meaning you can create and program you own NFC tags and devices. Create a league of ID tagged robots, or even program your own e-poster, there's so much you can do with this shield!

The Wifi Shield

This shield makes use of a powerful wireless networking module, adding a whole host of networking capabilities to the Arduino. This shield only requires a connection to the serial port on the Arduino, for ultimate networking ease. It's also fully stackable, so there's no limit to the expansion possibilities!

The GPRS Shield

The G2RS Shield gives the Arduino the ability to use the GSM/G2RS cell phone network to make and receive both calls and texts! The shield also features improved power efficiency and optimized antenna design!

The XBee Shield

Xbees can be notoriously difficult to interface with Arduino, so this shield provides a seamless connection between the two. All power regulation is taken care of, and even software serial ports can be used!

Entertainment – *Any shield that involves user interface, display, or some form of funky lighting, falls into this group!*

The E-Ink Shield

The E-Ink shield uses the technology of Electronic 2aper (the same technology used in E-book readers) to create an ultra-low power and highly readable display for your Arduino. This display can even retain text without power being applied!

The EL Shield

Electro-luminescence (EL) is a fascinating technology that can be harnessed to create dazzling light displays. The EL Shield provides an easy interface for the Arduino to control these displays, meaning you can create beautiful and dazzling displays!

The Music Shield

By using a high spec Audio Codec chip, the Music shield can add great quality audio functionality to any Arduino. The shield can even support a wide range of music formats, and even includes an onboard SD Card slot for easily storing your tunes!

Beginner – *The world of Arduino can be daunting to those less experienced! Any shield that's designed to ease a beginner into this world falls into this group.*

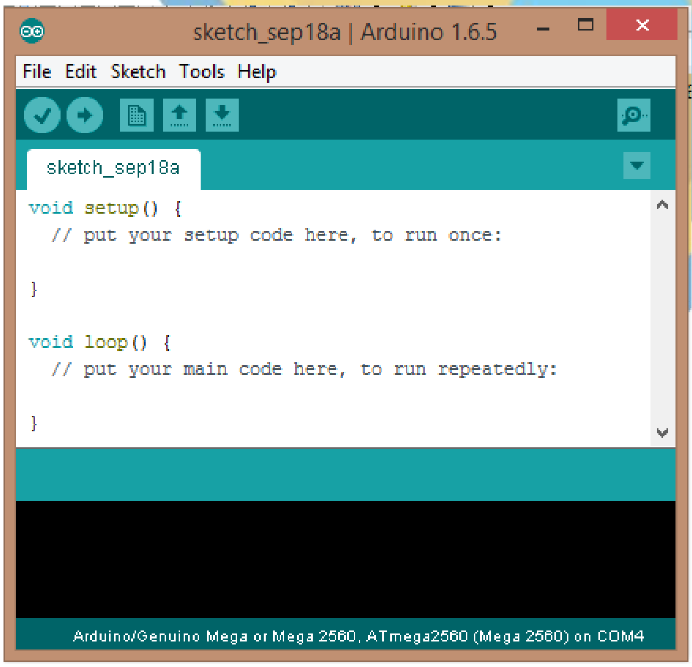
The Tick Tock Shield

This shield starts as a kit, that a beginner can build to gain the basic skills needed for full on Arduino tinkering. The kit includes a variety of input and output components, a fully detailed soldering guide, and a Real-Time clock IC, allowing you to build your own digital clock once you're more proficient!

The Shield Bot

One of the newest shields, the Shield bot is a little different to the other's we've seen. It's basically a robot rover chassis that the Arduino plugs straight into! The Shield Bot has all the power regulation for easy battery integration, motor drive circuitry for easy motor control, and even built in IR sensors for line following. Thanks to the various ports, it's also easy to add more functionality. This is such a great place to start for anyone interested in the world of robotics!

Arduino IDE



```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM4

This is the window of Arduino IDE 1.6.5. The Arduino IDE can be downloaded from arduino.cc It contains all the necessary programming tools for all the boards that arduino has designed. Regardless of the type of the board, the code follows a common structure as shown below.

Arduino Program Structure

The previous section looked at different code elements in the Arduino/C language. Now you'll see how they come together into a complete program. Open a fresh version of the Blink example by clicking on the upward pointing arrow in the Toolbar and then selecting 01.Basics → Blink.

Each Arduino program has three main parts:

1. variable declaration section
2. setup section
3. loop section

These sections are analogous to the ingredient list (variable declaration), preparation steps (setup), and cooking steps (loop) in a recipe. Here is a diagram showing where the three parts are located in the code:

When your program runs, it will first define your variables (the ingredients that you need), then execute the setup section once (set everything up to begin cooking), and then execute the loop section over and over (actually do the cooking).

1. Variable Declaration Section

2rograms often begin with introductory comments that explain what the program is doing. These comments come before the variable declarations. It is a good idea to begin every program with comments like these so that when you return to your program later you'll know what it does.
Variables are usually declared at the beginning of a program immediately following the introductory comments. All of the variables that you are using in your code should be listed here, before the setup and loop sections.

2. Setup Section

The setup section, which runs once when the program begins, follows the variable declaration section. Statements that lay the foundation for actions that happen later on in the program are put in the setup section. In particular, pinMode statements are almost always in this section. The section begins with the line void setup() {. Note that all statements in the setup section are placed between an open curly bracket '{' right after void setup() and a closed curly bracket '}' at the end of the section.

3. Loop Section

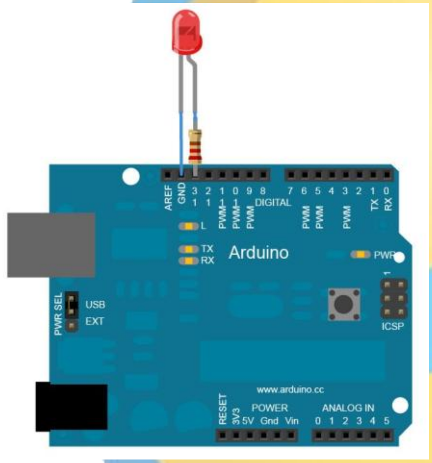
After the setup section runs, the loop section runs over and over until the board is turned off or reprogrammed—hence the word loop. The statements that carry out the main action of your program are in this section.As with the setup section, statements in the loop section are placed between open and closed curly brackets. These curly brackets tell the computer when the loop section begins (with the opening curly bracket) and when it ends (with the closing curly bracket).

Blink

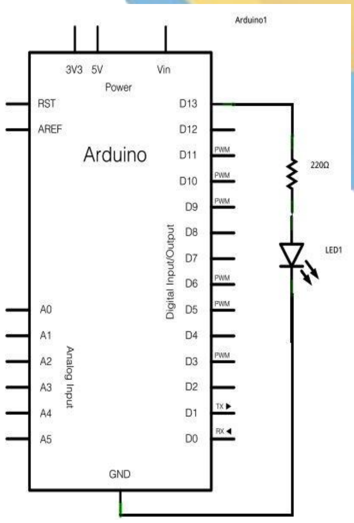
This example shows the simplest thing you can do with an Arduino or Genuino to see physical output: it blinks an LED.
Hardware Required

- Arduino or Genuino Board
- LED
- 220 ohm resistor

Circuit
To build the circuit, connect one end of the resistor to Arduino pin 13. Connect the long leg of the LED (the positive leg, called the anode) to the other end of the resistor. Connect the short leg of the LED (the negative leg, called the cathode) to the Arduino GND, as shown in the diagram and the schematic below.
Most Arduino boards already have an LED attached to pin 13 on the board itself. If you run this example with no hardware attached, you should see that LED blink.
The value of the resistor in series with the LED may be of a different value than 220 ohm; the LED will lit up also with values up to 1K ohm.



Schematic

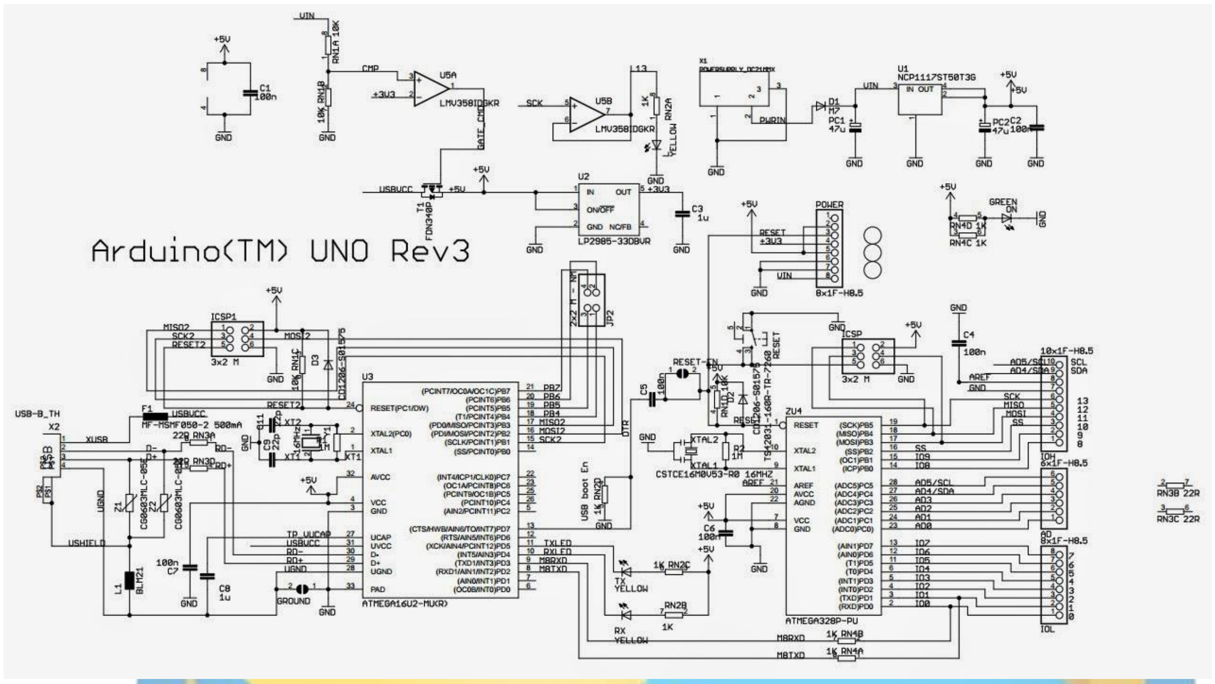


Code

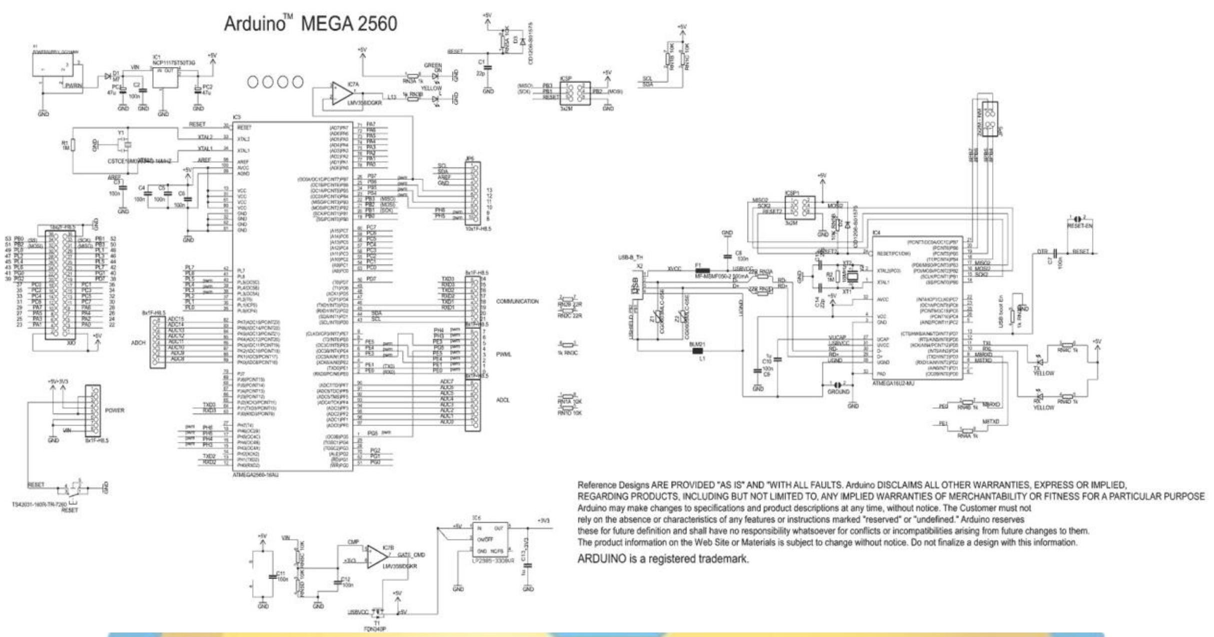
After you build the circuit plug your Arduino or Genuino board into your computer, start the Arduino Software (IDE) and enter the code below. You may also load it from the menu File/Examples/01.Basics/Blink. The first thing you do is to initialize pin 13 as an output pin with the line
pinMode(13, OUT2UT);
In the main loop, you turn the LED on with the line:
digitalWrite(13, HIGH);
This supplies 5 volts to pin 13. That creates a voltage difference across the pins of the LED, and lights it up. Then you turn it off with the line:
digitalWrite(13, LOW);
That takes pin 13 back to 0 volts, and turns the LED off. In between the on and the off, you want enough time for a person to see the change, so the delay() commands tell the board to do nothing for 1000 milliseconds, or one second. When you use the delay() command, nothing else happens for that amount of time. Once you've understood the basic examples, check out the BlinkWithoutDelay example to learn how to create a delay while doing other things.
Once you've understood this example, check out the DigitalReadSerial example to learn how read a switch connected to the board.

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

 Most Arduinos have an on-board LED you can control. On the Uno and
 Leonardo, it is attached to digital pin 13. If you're unsure what
 pin the on-board LED is connected to on your Arduino model, check
 the documentation at http://www.arduino.cc

  This example code is in the public domain.

 modified 8 May 2014
 by Scott Fitzgerald
 */


// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUT2UT);
}

// the loop function runs over and over again forever
void loop() ·
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

Architecture of Arduino UNO R3



Architecture of Arduino Mega 2560



Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS. Arduino DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE Arduino may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

ARDUINO is a registered trademark.