

Programming *“Arduino”*

Create Microcontroller-Based Projects

Instructor / Facilitator - Alan Rux

Programming “Arduino”

- This is a “**Hands-On** Course”, the instructor will guide you on where to find the information you need.
- **Lectures will be “ON-LINE”** (24/7 web access)
- The instructor will not do it for you in the classroom and then you repeat what was showed.
- You will be given assignments.
- The classroom time will be group discussion, the ***instructor will act as an facilitator.***
- The laboratory time you will apply what you learned.
- Team or Group participation is encouraged.

What is a Microcontroller?

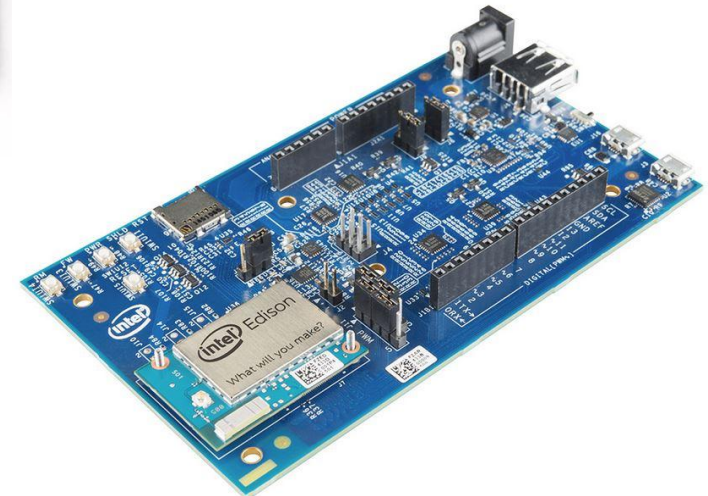
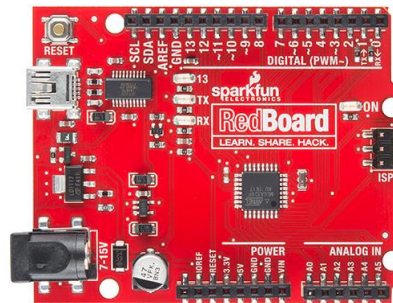
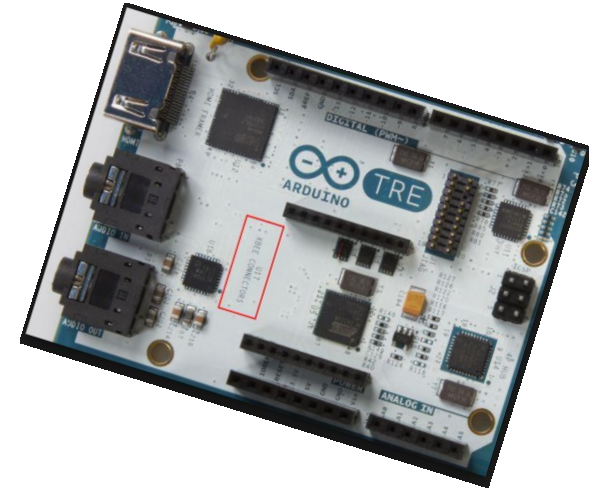
A microcontroller is a very small computer that has digital electronic devices (peripherals) built into it that helps it control things. These peripherals allow it to sense the world around it and drive the actions of external devices. An example of a use for a microcontroller is to sense a temperature and depending on the value sensed it could either turn on a fan if things were too warm or turn on a heater if things were too cool.

You might already be aware that microcontrollers are in common devices like cell phones, microwave ovens, and alarm clocks that have buttons for you to input information and displays to tell you things. But there are even more microcontrollers embedded in things where you never see them.

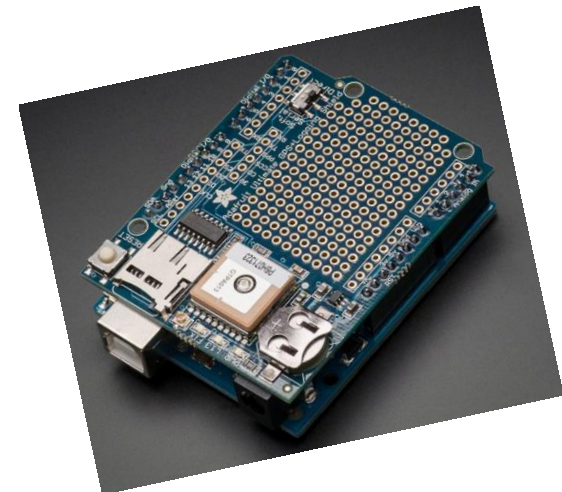
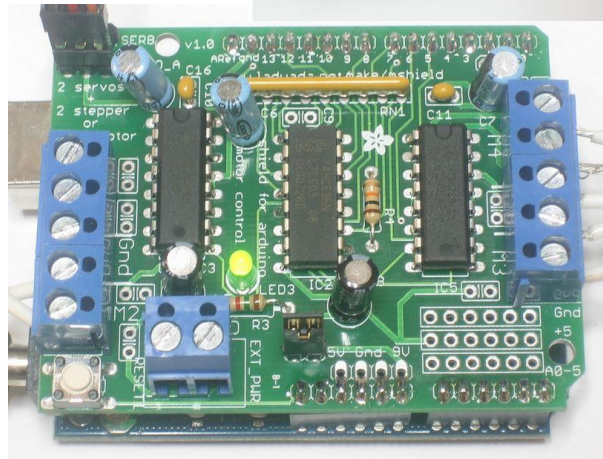
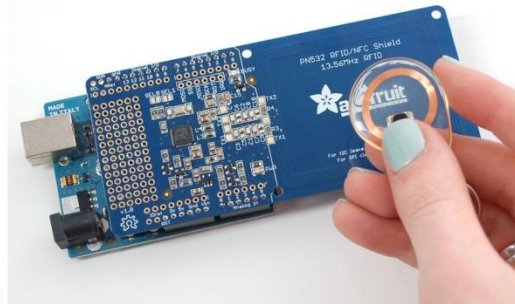
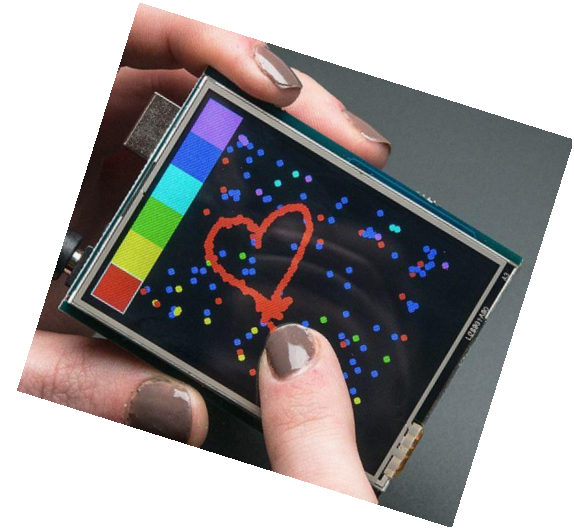
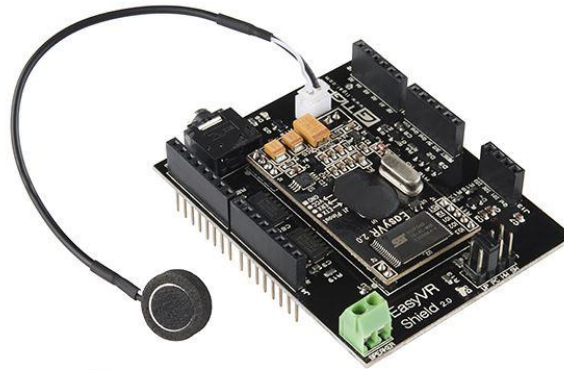
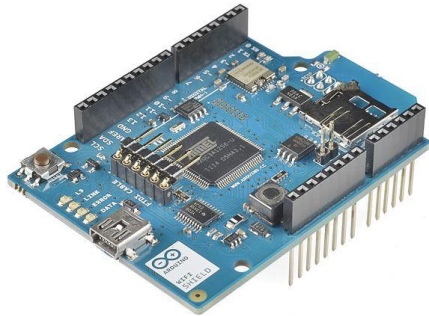
For example there are 30 or more microcontrollers in an automobile. These do everything from sensing the oxygen intake to setting the fuel air mixture to measuring the cabin temperature for controlling the air conditioning levels

Open-Source Electronics Prototyping Platform based
on flexible, easy-to-use hardware & software

Arduino prototyping “Platform”

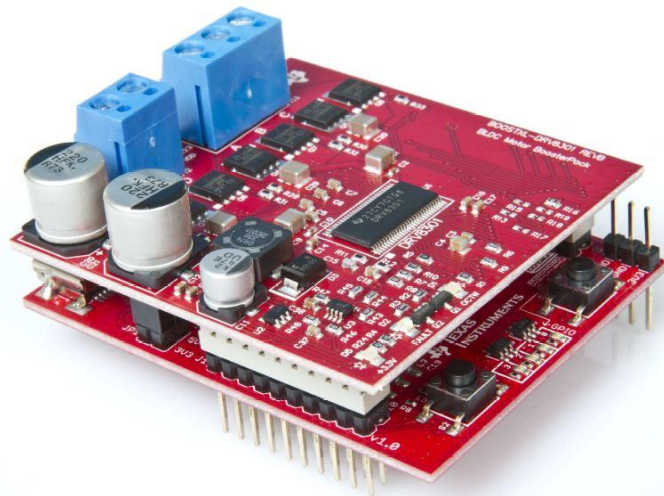
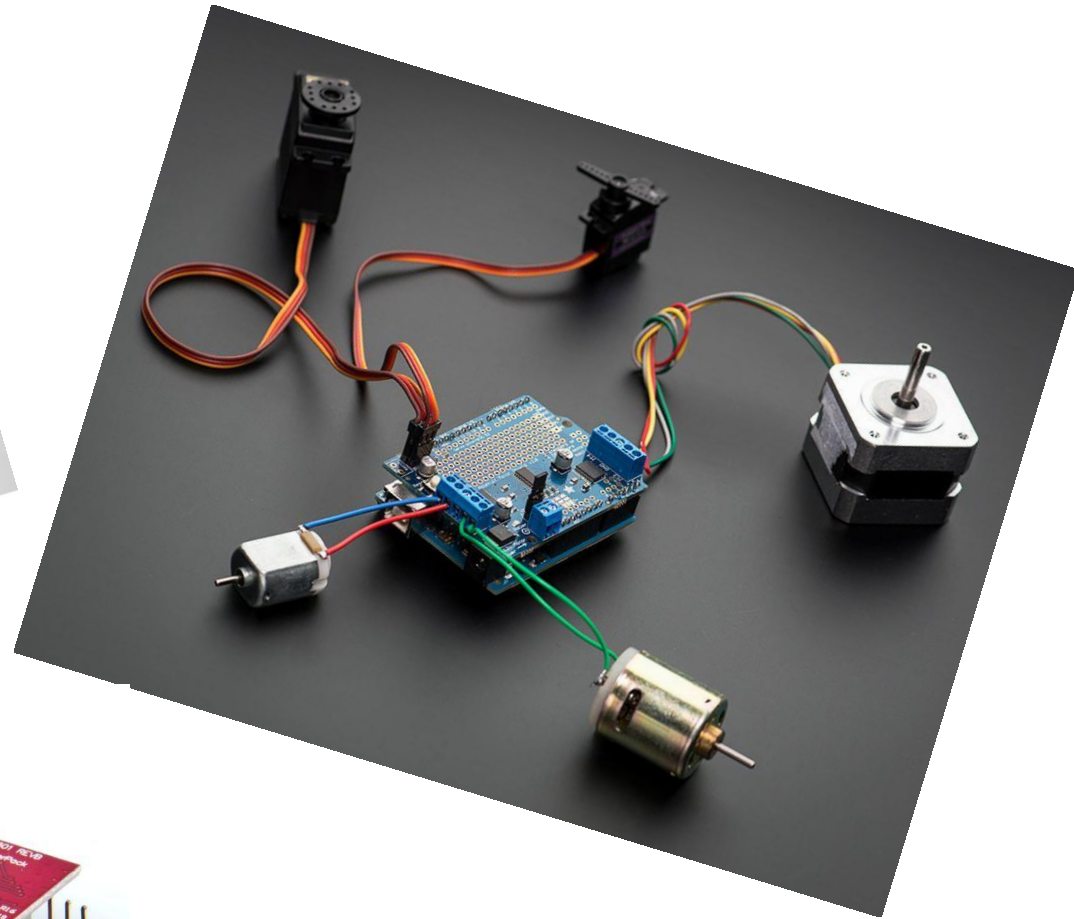
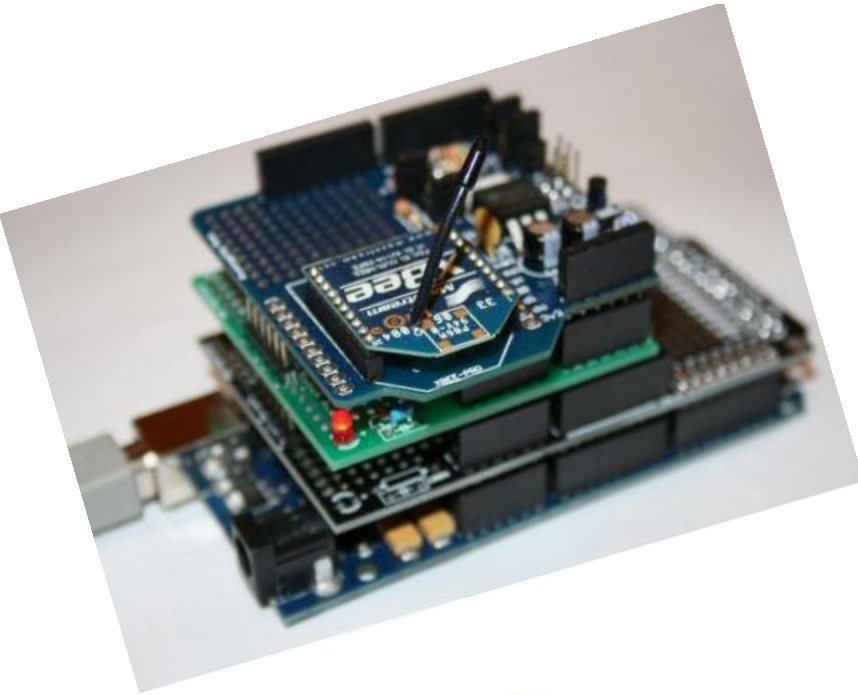


SHIELDS Arduino platform modules (over 500 and growing)



learning by doing

SHIELDS Arduino platform modules



New World Engineering ***solutions***

**Motor development
in the palm of your hand**

NEW InstaSPIN-FOC™ MCU LaunchPad
and Motor Drive BoosterPack for under \$70

► Buy now
at Mouser



 TEXAS INSTRUMENTS

Speed development, and improve efficiency for
advanced designs

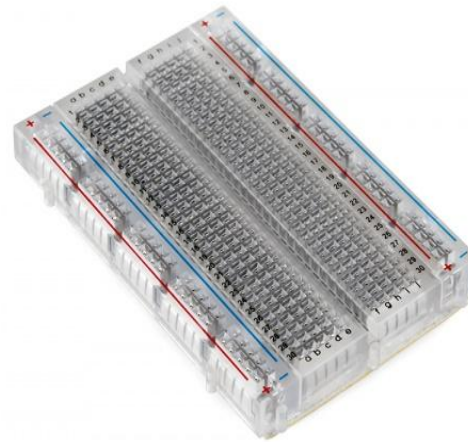
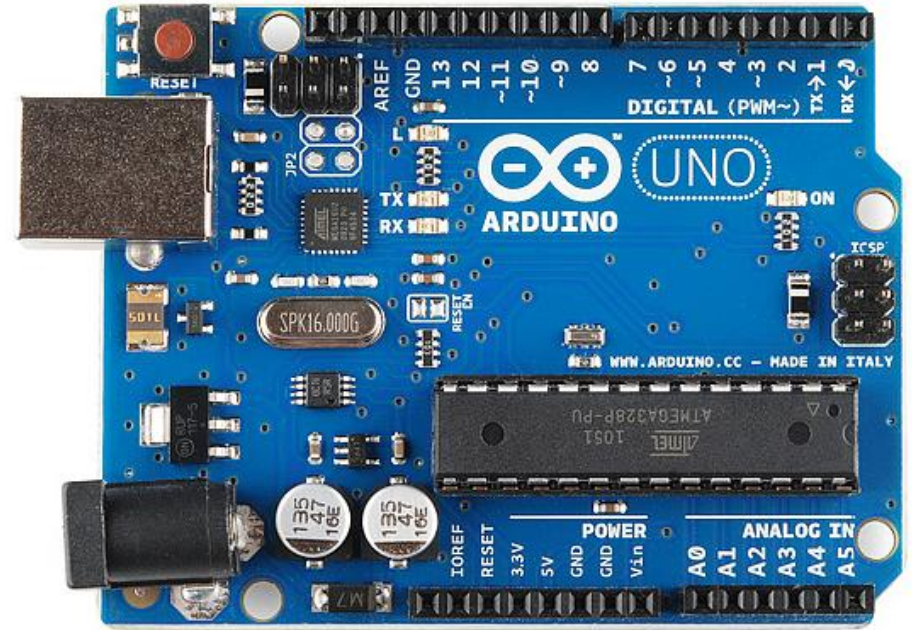
Electrostatic Discharge Hazard

Electrostatic discharge (ESD) is the sudden flow of electricity between two objects caused by contact, an [electrical short](#), or [dielectric breakdown](#). ESD can be caused by a buildup of static electricity by tribocharging, or by electrostatic induction.



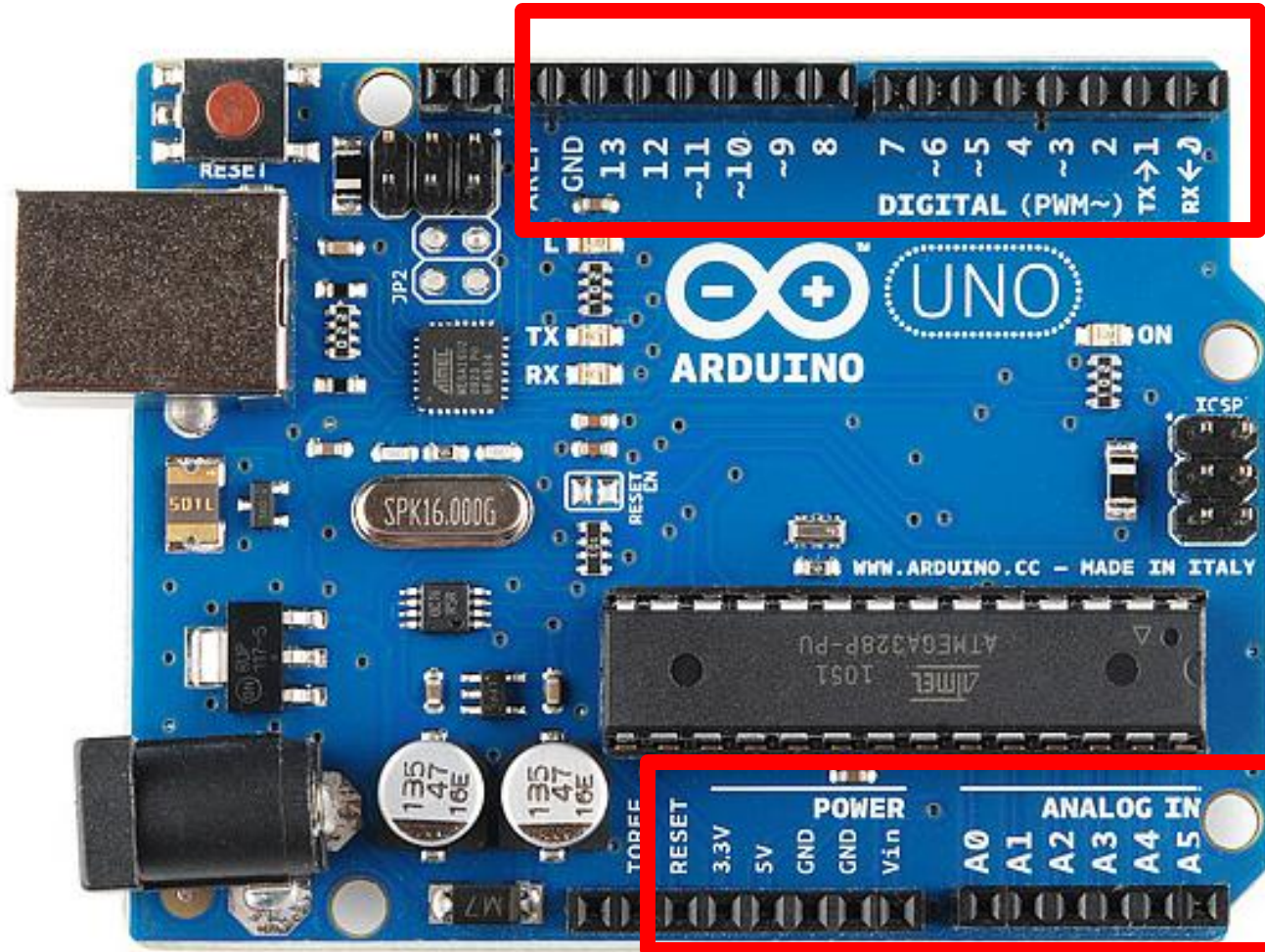
Minimum tools to get Started

- Arduino platform Board
- Solderless Breadboard
- USB Cable



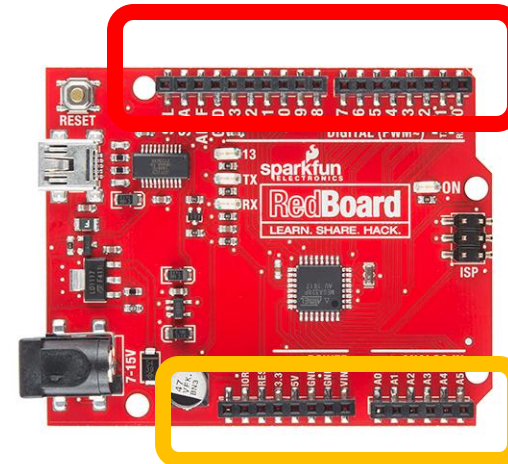
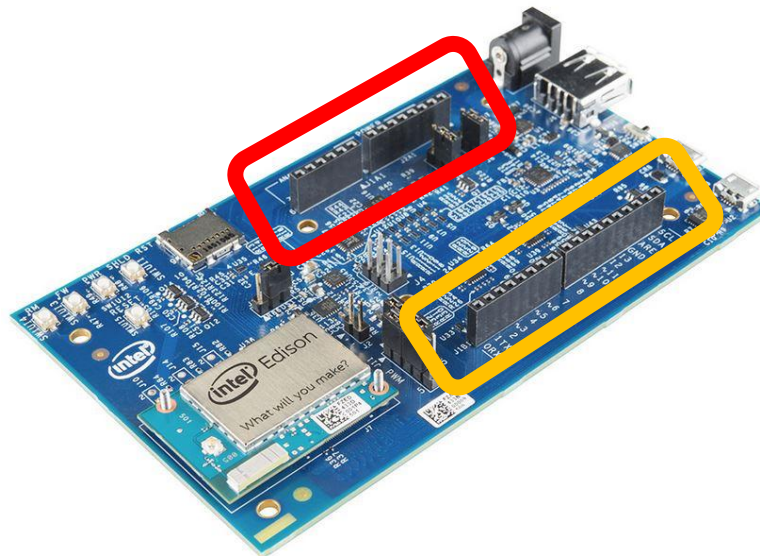
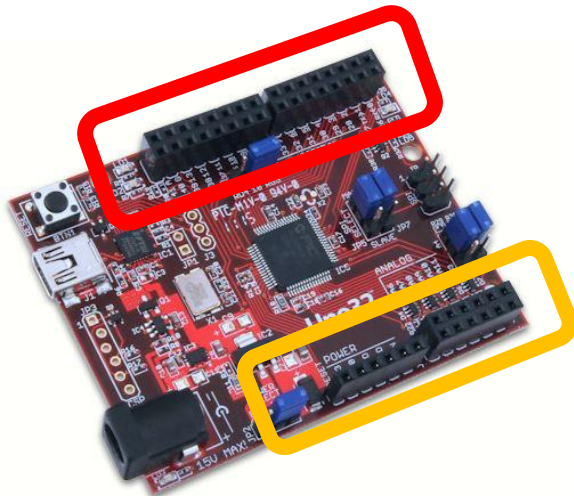
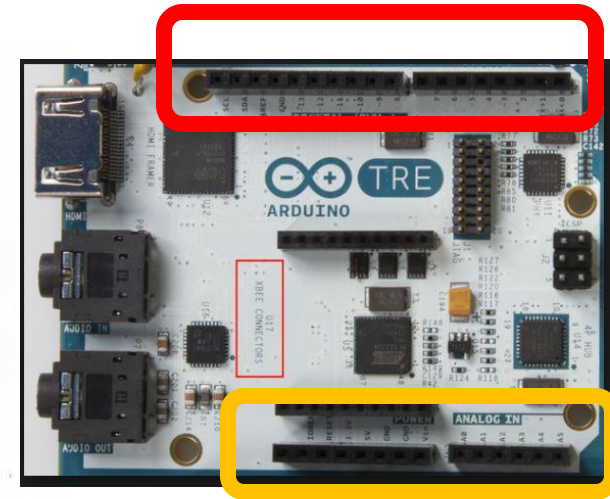
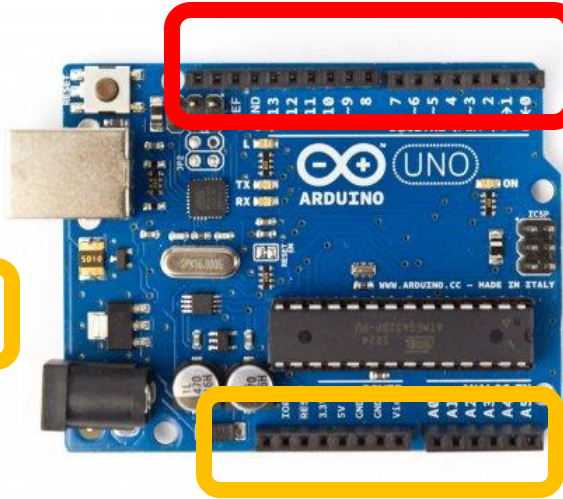
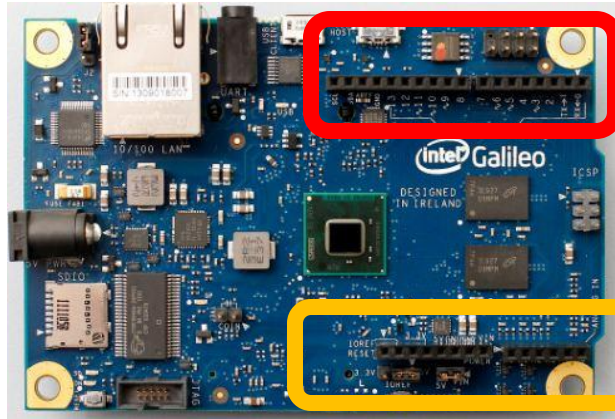
Arduino

“Platform”

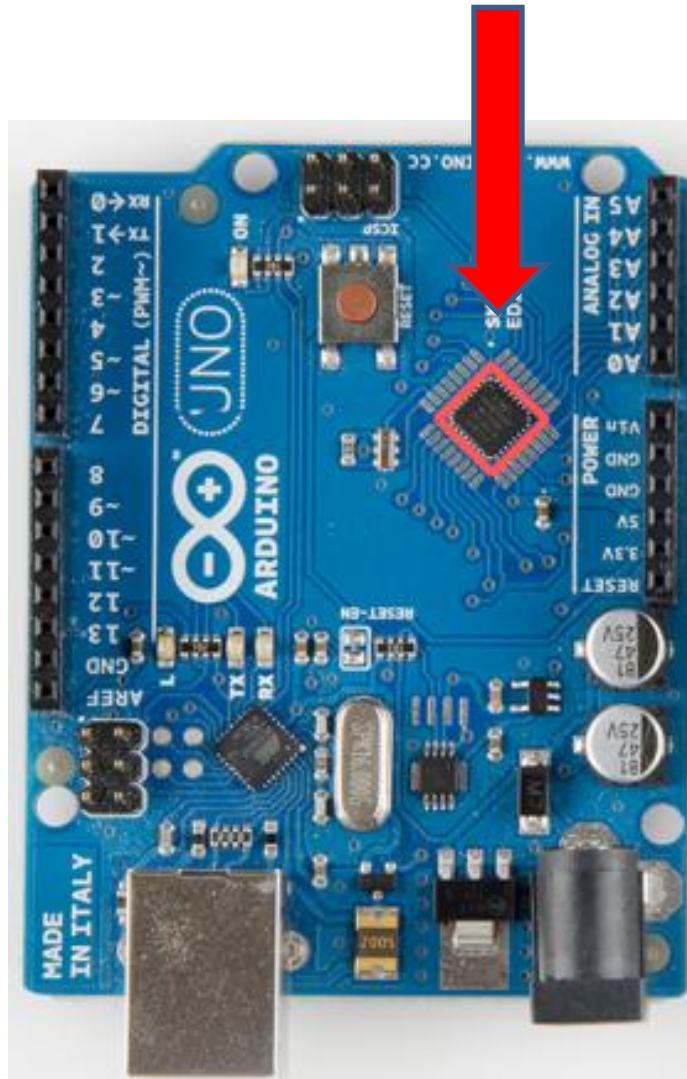


Power Analog In - Digital I/O - Pulse With Modulation

Arduino “Platform”



SMD Arduino Uno



*Plastic
Through the Hole
Arduino Uno*



External Power
Jack

USB - B
Connector

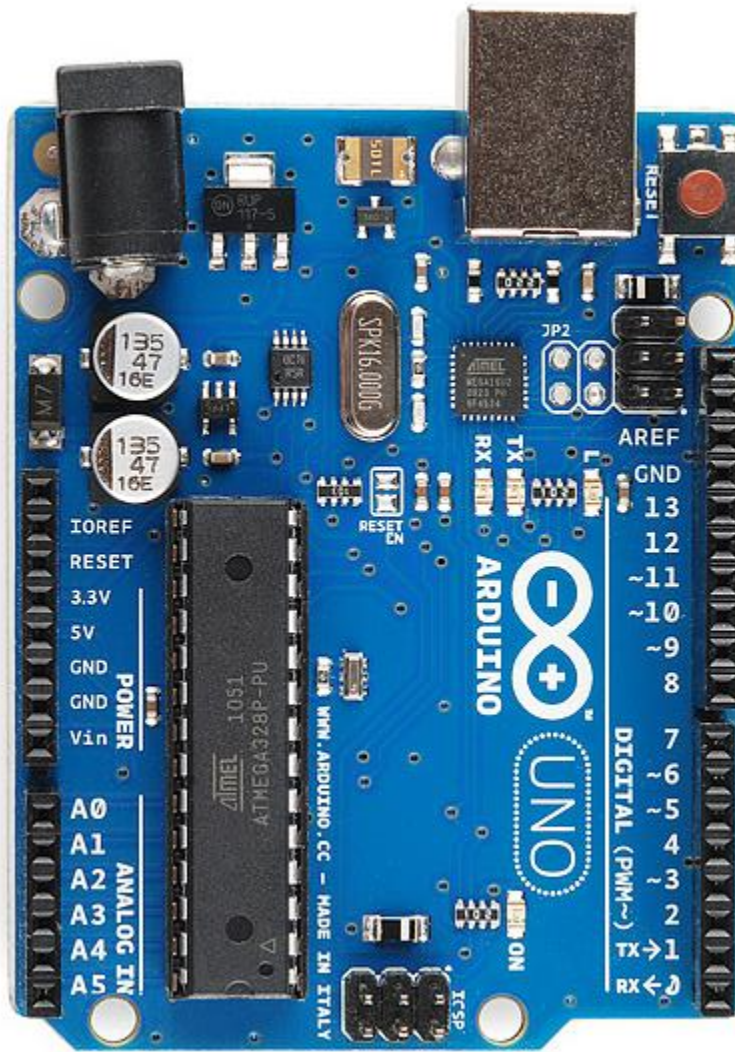
Reset
Button

Power
Pins

6 Analog
Input Pins

Digital I/O
14 pins,
6 provide
PWM output

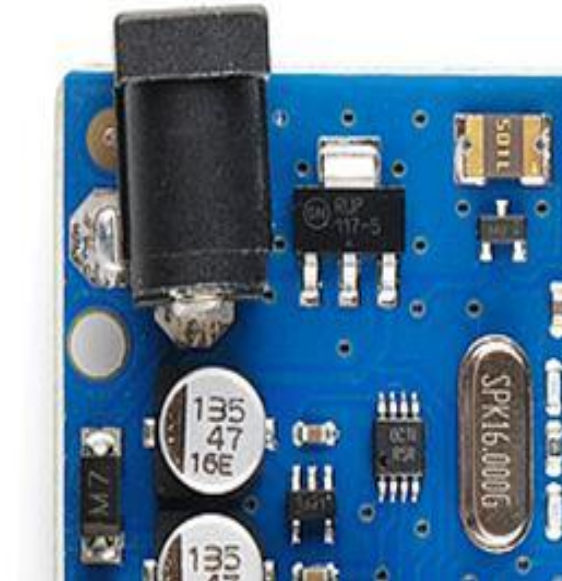
Serial Programming
Connector (not used)



External Power Jack

- Power Jack of Vin pin
- +7 to +12 V.D.C.
- 200 to 500ma. current

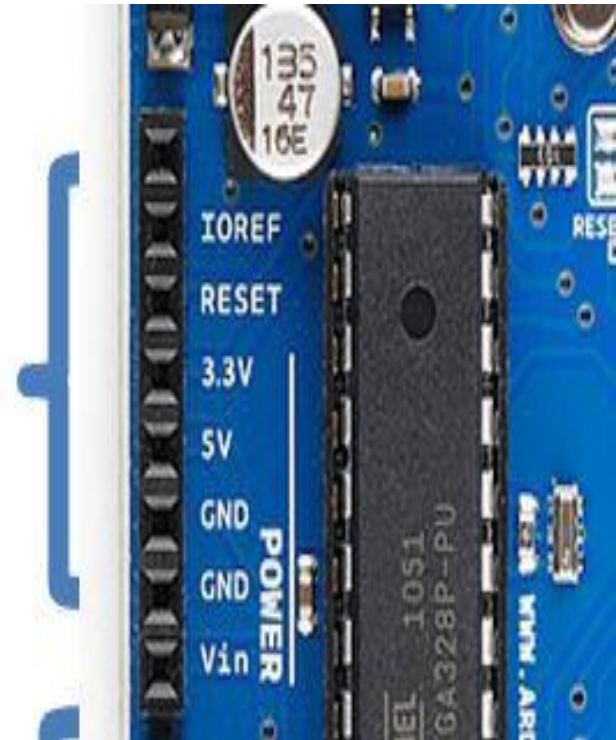
External Power
Jack



“Platform” Power Pins

- I/O Ref. = no connection
- Reset , when low,
0 volts = reset chip
- 3.3V ,50 ma. for 3.3 V shields
- 5V , 300 ma. for shields
- Grounds = 0 volts
- Vin = 7 to 12 V.D.C.
(same as external power jack)

Power
Pins



“Platform” Analog Inputs

- Six Channels A/D converter
- Input 0v. To + 5v
- 1024 bits conversion (0-1023)
- .004889 volts / step

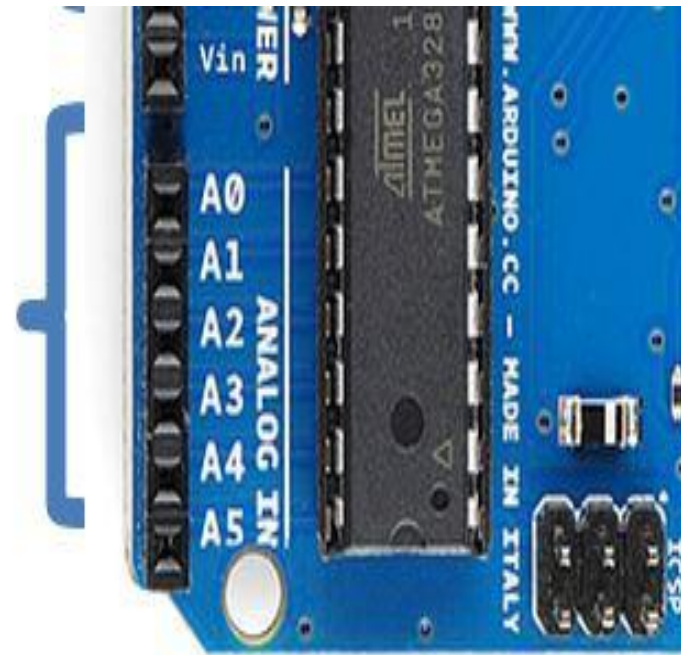
0 bits = 0 volts

256 bits = + 1.25 volts

512 bits = +2.5 volts

1023 bits = + 5 volts

6 Analog
Input Pins



“Platform” Digital I /O Pins

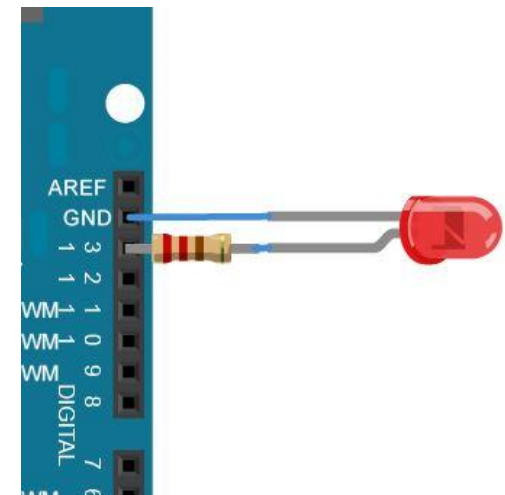
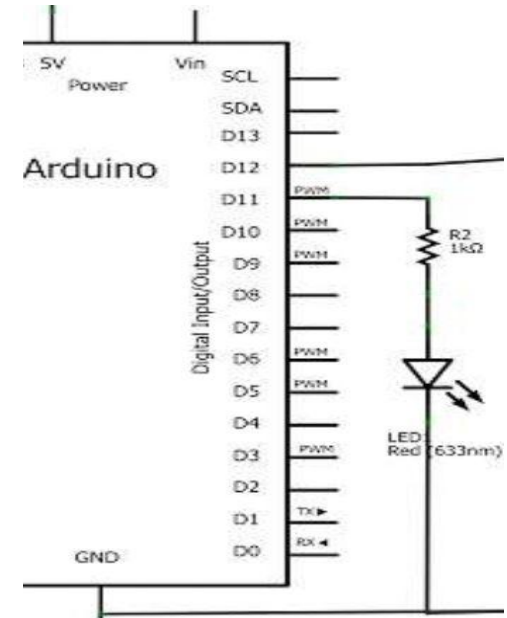
- D0 to D13 = 14 pins
- “0” = 0 v., “1” = +5v.
- Six pins = PWM output
D3,D5,D6,D9,D10,D11
- AREF. = A/D ref. voltage
- I/O pin 0 = RX, I/O pin=TX
(serial com, port)
- I/O current limited
(see next slide)




Digital I/O
14 pins,
6 provide
PWM output

Digital I /O Pins current limit

- 40 ma. Max current / pin
- **200 ma. Max for IC
(all pins total)**
- **< 10ma. Recommended**
- Use 560 ohm resistor to
1k ohm resistor



Atmel - ATmega328 website



WorldwideCommunitiesmyAtmelLog InCart

ProductsApplicationsTechnologiesDesign SupportAbout AtmelBuy

Microcontrollers
AVR 8- and 32-bit MCUs
32-bit AVR UC3 MCUs
AVR XMEGA MCUs
→megaAVR MCUs
tinyAVR MCUs
Battery Management MCUs
Automotive AVR MCUs
SMART ARM-based MCUs
8051 Architecture

Touch Solutions

Automotive

Wireless Connectivity

Smart Energy

Memory

Security ICs

Digital Broadcast

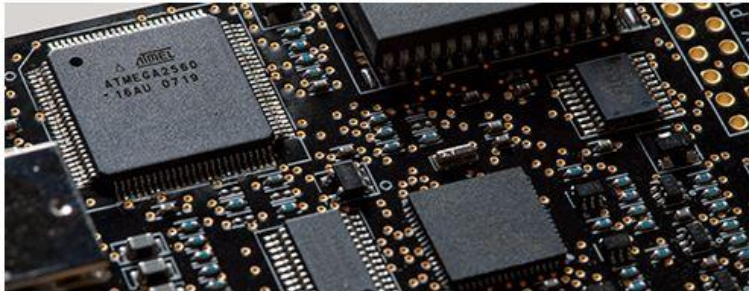
More Products

Microcontroller Selector

Home > Products > Microcontrollers > AVR 8- and 32-bit MCUs > megaAVR MCUs

ATmega328P

OverviewParametersToolsDocumentsApplications



Get Started

We'll tell you all you need to know to start evaluating and working with this product.

» Start Now

» Contact Sales


» Request Samples

» Sign-up for News


Documents for ATmega328P

Datasheet

PDFSoftwareDescription



ATmega48A/PA/88A/PA/168A/PA/328/P Summary
(file size: 570KB, 31 pages, revision G, updated: 02/2013)



ATmega48A/PA/88A/PA/168A/PA/328/P Complete
(file size: 35MB, 660 pages, revision G, updated: 02/2013)

Application Note

Related Items

» Third Party Support

» Consultants

» University Program

» AVR Knowledge Base

» Technical Support

» What's Changed

» Mature Devices

Atmel ATmega328

[http://www.atmel.com
/devices/atmega328.aspx](http://www.atmel.com/devices/atmega328.aspx)



Atmel 8-bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash

**ATmega48A; ATmega48PA; ATmega88A; ATmega88PA;
ATmega168A; ATmega168PA; ATmega328; ATmega328P**

SUMMARY

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
 - 131 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 20 MIPS Throughput at 20MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
 - 256/512/1024/2048 Bytes EEPROM

Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf (SECURED) - Adobe Reader

File Edit View Window Help

My Files

Bookmarks

- Features
- 1. Pin Configurations
- 2. Overview
- 3. Resources
- 4. Data Retention
- 5. About Code Examples
- 6. Capacitive Touch Sensing
- 7. AVR CPU Core
- 8. AVR Memories
- 9. System Clock and Clock Options
- 10. Power Management and Sleep Modes

Atmel

Atmel 8-bit Microcontroller with 4/8/16/32KBytes In-System Programmable Flash

**ATmega48A; ATmega48PA; ATmega88A; ATmega88PA;
ATmega168A; ATmega168PA; ATmega328; ATmega328P**

Features

- High Performance, Low Power Atmel® AVR® 8-Bit Microcontroller Family
- Advanced RISC Architecture
 - 131 Powerful Instructions – Most Single Clock Cycle Execution
 - 32 x 8 General Purpose Working Registers
 - Fully Static Operation
 - Up to 20 MIPS Throughput at 20MHz
 - On-chip 2-cycle Multiplier
- High Endurance Non-volatile Memory Segments
 - 4/8/16/32KBytes of In-System Self-Programmable Flash program memory
 - 256/512/1024/2048 Bytes EEPROM
 - 512/1K/2K/4K Bytes Internal SRAM
 - Write/Erase Cycles: 10,000 Flash/100,000 EEPROM
 - Data retention: 20 years at 85°C/100 years at 25°C⁽¹⁾
 - Optional Boot Code Section with Independent Lock Bits
 - In-System Programming by On-chip Boot Program

Arduino Platform Software

<http://arduino.cc/en/Main/Software>



Arduino IDE

Arduino 1.0.5

Download

Arduino 1.0.5 ([release notes](#)), hosted by [Google Code](#):

NOTICE: Arduino Drivers have been updated to add support for Windows 8.1, you can download the updated IDE (version 1.0.5-r2 for Windows) from the download links below.

- [Windows Installer, Windows ZIP file \(for non-administrator install\)](#)
- [Mac OS X](#)
- [Linux: 32 bit, 64 bit](#)
- [source](#)

Next steps

[Getting Started](#)

[Reference](#)

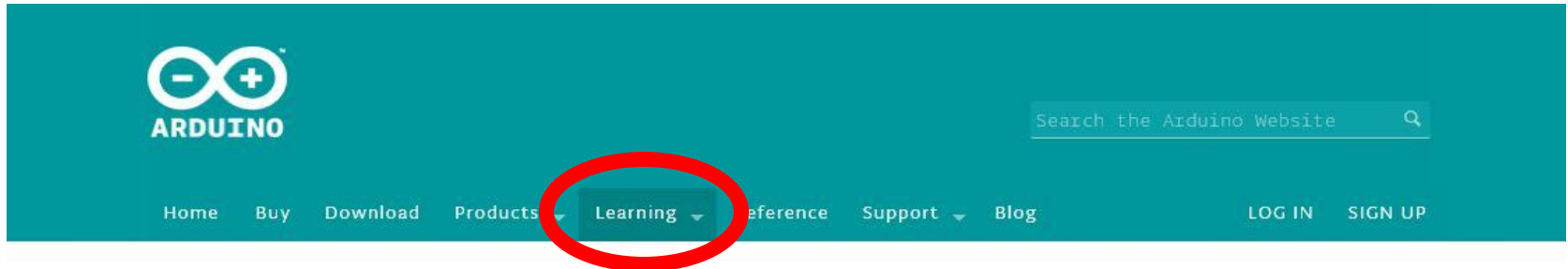
[Environment](#)

[Examples](#)

[Foundations](#)

[FAQ](#)

Getting Started on Windows



Getting Started with Arduino on Windows

This document explains how to connect your Arduino board to the computer and upload your first sketch.

- 1 | [Get an Arduino board and USB cable](#)
- 2 | [Download the Arduino environment](#)
- 3 | [Connect the board](#)
- 4 | [Install the drivers](#)
- 5 | [Launch the Arduino application](#)
- 6 | [Open the blink example](#)
- 7 | [Select your board](#)
- 8 | [Select your serial port](#)
- 9 | [Upload the program](#)

Language Reference



Reference [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

Language Reference

Arduino programs can be divided in three main parts: *structure*, *values* (variables and constants), and *functions*.

Structure

- `setup()`
- `loop()`

Control Structures

- `if`
- `if...else`
- `for`
- `switch case`
- `while`
- `do...while`

Variables

Constants

- `HIGH` | `LOW`
- `INPUT` | `OUTPUT` | `INPUT_PULLUP`
- `LED_BUILTIN`
- `true` | `false`
- integer constants
- floating point constants

Data Types

Functions

Digital I/O

- `pinMode()`
- `digitalWrite()`
- `digitalRead()`

Analog I/O

- `analogReference()`
- `analogRead()`
- `analogWrite()` - *PWM*

Language Reference



Reference [Language](#) | [Libraries](#) | [Comparison](#) | [Changes](#)

loop()

After creating a `setup()` function, which initializes and sets the initial values, the `loop()` function does precisely what its name suggests, and loops consecutively, allowing your program to change and respond. Use it to actively control the Arduino board.

Example

```
const int buttonPin = 3;

// setup initializes serial and the button pin
void setup()
{
  Serial.begin(9600);
  pinMode(buttonPin, INPUT);
}

// loop checks the button pin each time,
// and will send serial if it is pressed
void loop()
{
  if (digitalRead(buttonPin) == HIGH)
```


Arduino Development Environment

Sketches



Arduino Development Environment

The Arduino development environment contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions, and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

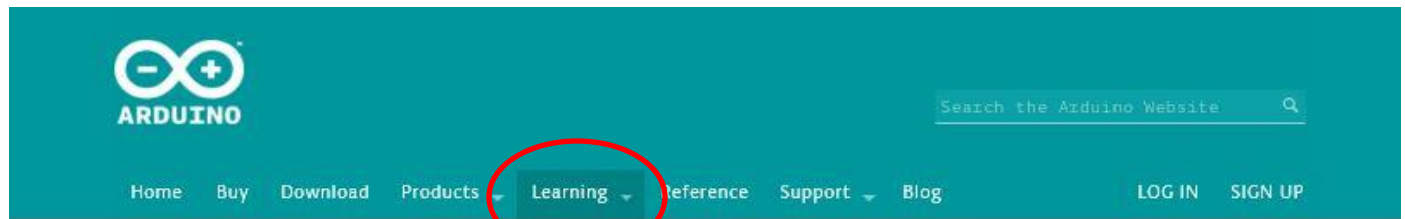
Writing Sketches

Software written using Arduino are called **sketches**. These sketches are written in the text editor. Sketches are saved with the file extension `.ino`. It has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino environment including complete error messages and other information. The bottom righthand corner of the window displays the current board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

NB: Versions of the IDE prior to 1.0 saved sketches with the extension `.pde`. It is possible to open these files with version 1.0, you will be prompted to save the sketch with the `.ino` extension on save.

Foundations

Fritzing



Learning [Examples](#) | [Foundations](#) | [Hacking](#) | [Links](#)

Examples > Basics

Bare Minimum code needed to get started

This example contains the bare minimum of code you need for an Arduino sketch to compile: the `setup()` method and the `loop()` method.

Hardware Required

- [Arduino Board](#)

Circuit

Only your Arduino Board is needed for this example.

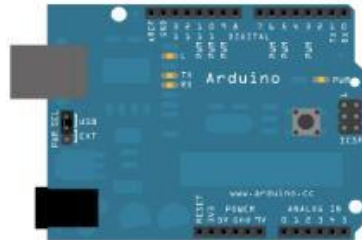


image developed using [Fritzing](#). For more circuit examples, see the [Fritzing project page](#).

Code

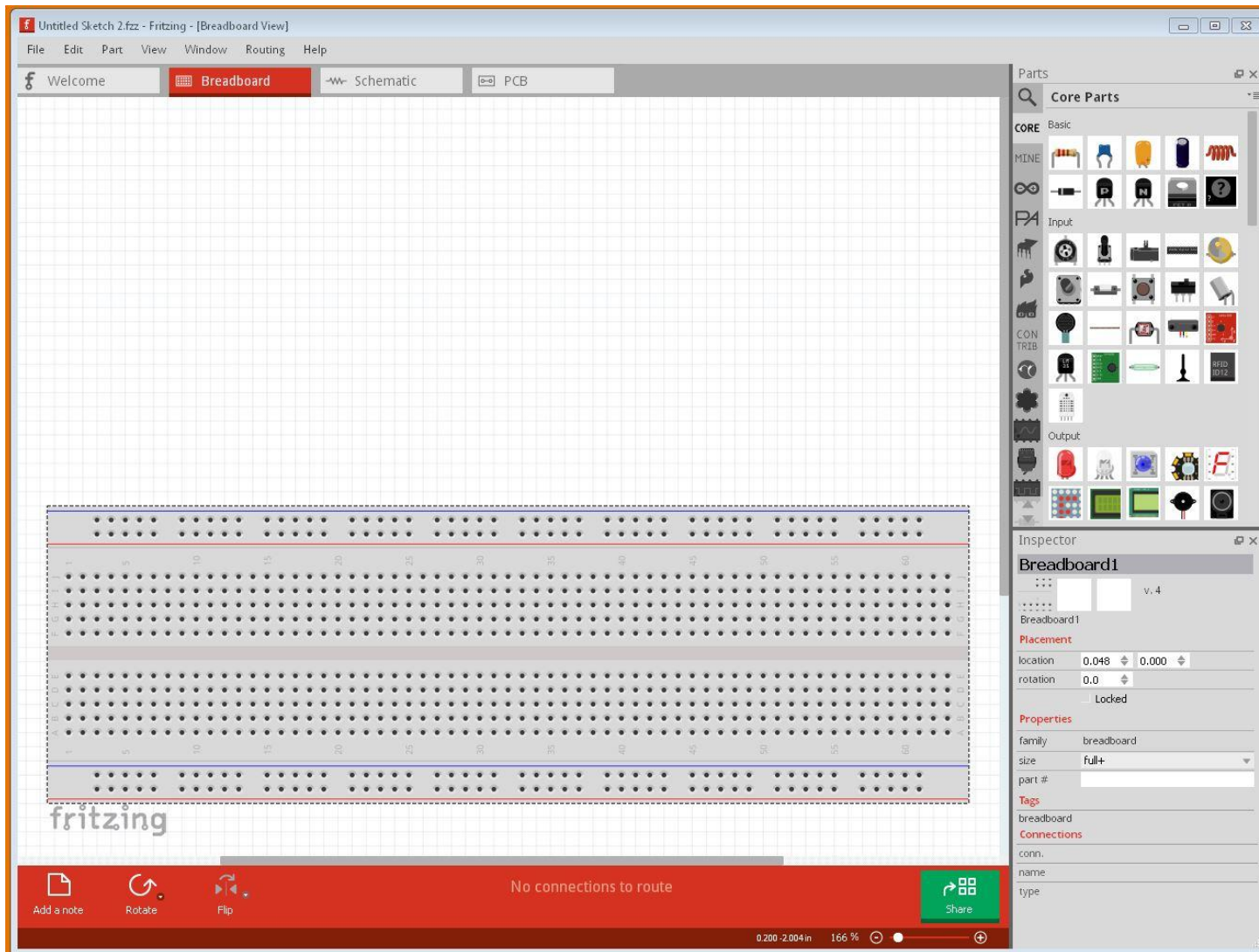
The `setup()` function is called when a sketch starts. Use it to initialize variables, pin modes, start using libraries, etc.

The setup function will only run once, after each powerup or reset of the Arduino board.

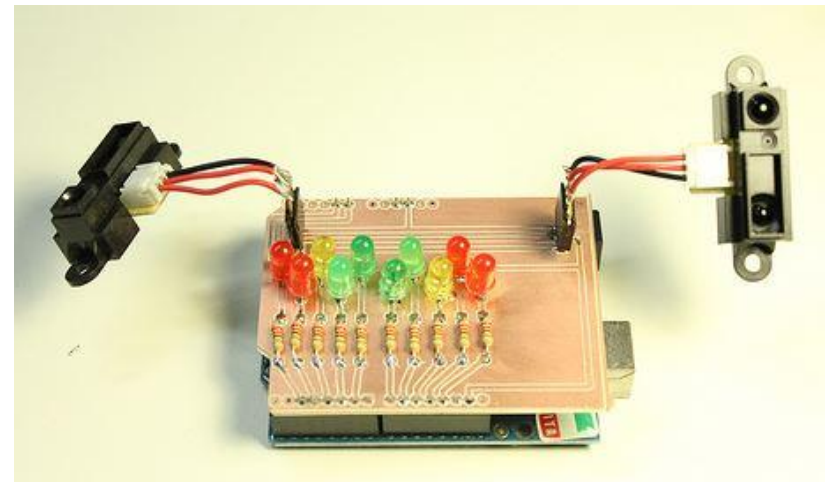
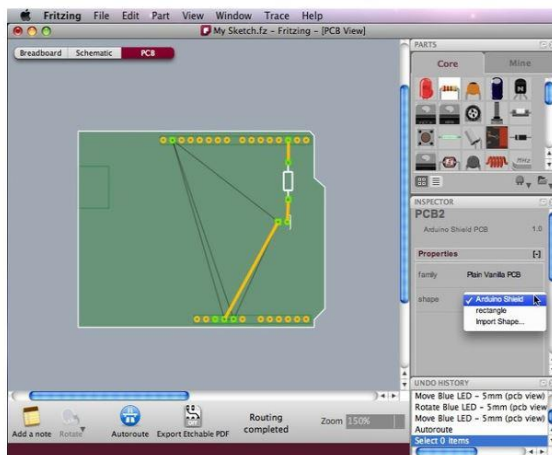
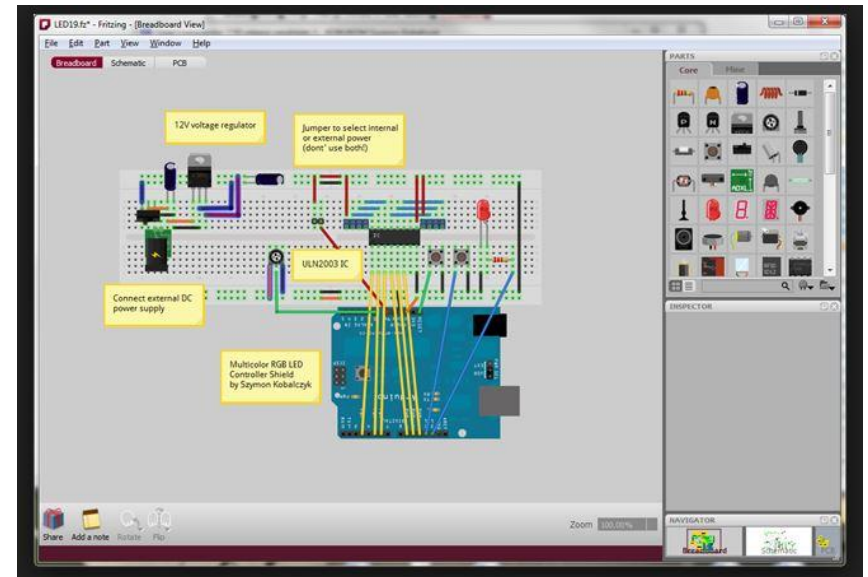
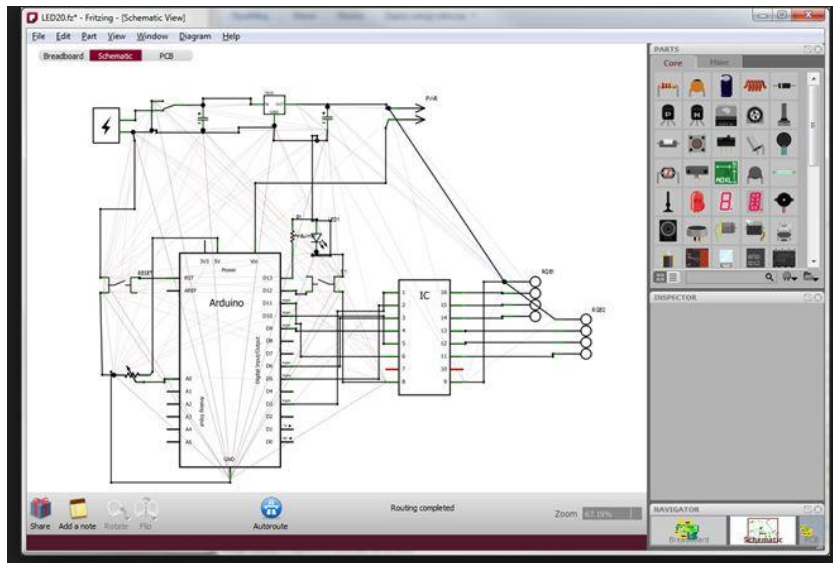


Fritzing is an [open-source hardware initiative](#) that makes electronics accessible as a creative material for anyone, a software tool, a community website and services in the spirit of [Processing](#) and [Arduino](#), fostering a creative ecosystem that allows users to *document* their prototypes, *share* them with others, and layout and *manufacture* professional pcbs.

Fritzing Breadboard



Fritzing Shield development



Core Functions examples



[Learning](#) [Examples](#) | [Foundations](#) | [Hacking](#) | [Links](#)

Examples

See the following [examples](#) for an overview of the Arduino Core functions and Libraries; the [foundations page](#) for in-depth description of core concepts of the Arduino hardware and software; the [hacking page](#) for information on extending and modifying the Arduino hardware and software; and the [links page](#) for other documentation.

NOTE: THESE EXAMPLES ARE WRITTEN FOR ARDUINO 1.0 AND LATER.
CERTAIN FUNCTIONS MAY NOT WORK IN EARLIER VERSIONS. FOR BEST RESULTS, DOWNLOAD THE LATEST VERSION.

Core Functions

Simple programs that demonstrate basic Arduino commands. These are included with the Arduino environment; to open them, click the Open button on the toolbar and look in the examples folder.

I. Basics

- [BareMinimum](#): The bare minimum of code needed to start an Arduino sketch.
- [Blink](#): Turn an LED on and off.
- [DigitalReadSerial](#): Read a switch, print the state out to the

Libraries

Examples from the libraries that are included in the Arduino software.

Bridge Library (for the Arduino Yún)

- [Bridge](#): Access the pins of the board with a web browser.
- [Console ASCII Table](#): Demonstrates printing various formats to the Console.
- [Console Pixel](#): Control an LED through the Console.
- [Console Read](#): Parse information from the Console and repeat it back.

Exercises

- ❑ The Arduino website is divided into six main sections. First visit the Main Site arduino.cc/en/Main.
- ❑ Visit the Getting Started section and get acquainted with what they have to say about themselves.
- ❑ Visit the Learning section and note the extensive list of links to projects. This will come in handy later when you want to do things not discussed in this book.
- ❑ Visit the Reference section. This provides links to the documentation for the Arduino functions library.
- ❑ Visit the Blog at arduino.cc/blog. This is where the team members post interesting things about the Arduino.

Exercises

- ❑ Visit the Playground. This is a wiki where Arduino users can contribute. This is another huge resource that you will want to use later as you gain more experience with the Arduino.
- ❑ Visit the forum. This is a great place to get good answers to your questions.
- ❑ However, if you aren't familiar with typical forum etiquette I strongly recommend you first visits: *How to ask questions that have a better chance of getting an answer*: <http://www.catb.org/~esr/faqs/smart-questions.html>

Exercises

- ☐ Go to arduino.cc/en/Main/Software and download the latest software for your particular computer system.
- ☐ Open the Arduino IDE (Integrated Development Environment)
- ☐ Compile a Sketch to Blink an LED on I/O pin 13
- ☐ Upload the sketch to the Arduino to blink the LED attached to pin 13

Exercises guide

<http://arduino.cc/en/Guide/Windows>
<http://arduino.cc/en/Guide/Windows>

Getting Started with Arduino on Windows

This document explains how to connect your Arduino board to the computer and upload your first sketch.

- 1 | Get an Arduino board and USB cable
- 2 | Download the Arduino environment
- 3 | Connect the board
- 4 | Install the drivers
- 5 | Launch the Arduino application
- 6 | Open the blink example
- 7 | Select your board
- 8 | Select your serial port
- 9 | Upload the program