UNIT IV
OPEN PLATFORMS AND PROGRAMMING

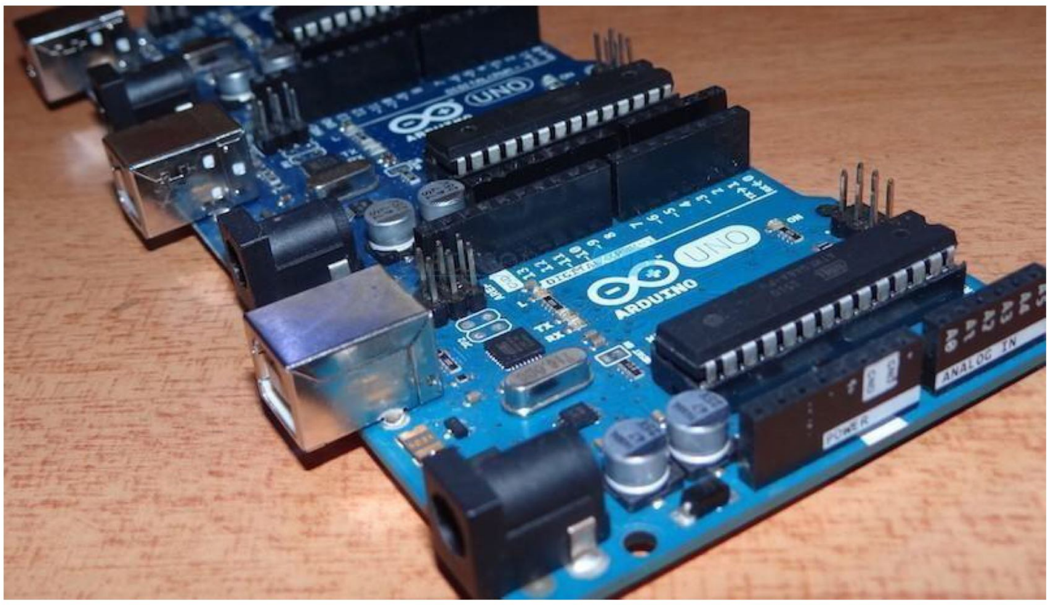
4.1.1 IOT deployment for Raspberry Pi /Arduino platform

A decade ago, working around electronics involved knowledge in physics and math, expensive lab equipment, a laboratory type setup and important of all, love for electronics. But the picture has changed over the decade or so where the above-mentioned factors became irrelevant to work around electronics except for the last part: love for electronics. One such product which made use of the above specified and many other reasons and made electronics be able reach anyone regardless of their background is "Arduino".

Introduction

Arduino is an open-source prototyping platform in electronics based on easy-to-use hardware and software. Subtly speaking, Arduino is a microcontroller based prototyping board which can be used in developing digital devices that can read inputs like finger on a button, touch on a screen, light on a sensor etc. and turning it in to output like switching on an LED, rotating a motor, playing songs through a speaker etc.

The Arduino board can be programmed to do anything by simply programming the microcontroller on board using a set of instructions for which, the Arduino board consists of a USB plug to communicate with your computer and a bunch of connection sockets that can be wired to external devices like motors, LEDs etc. The aim of Arduino is to introduce the world of electronics to people who have small to no experience in electronics like hobbyists, designers, artists etc.



Arduino is based on open source electronics project i.e. all the design specifications, schematics, software are available openly to all the users. Hence, Arduino boards can bought from vendors as they are commercially available or else you can make your own board by if you wish i.e. you can download the schematic from Arduino's official website, buy all the components as per the design specification, assemble all the components, and make your own board.

Hardware and Software

Arduino boards are generally based on microcontrollers from Atmel Corporation like 8, 16 or 32 bit AVR architecture based microcontroller.

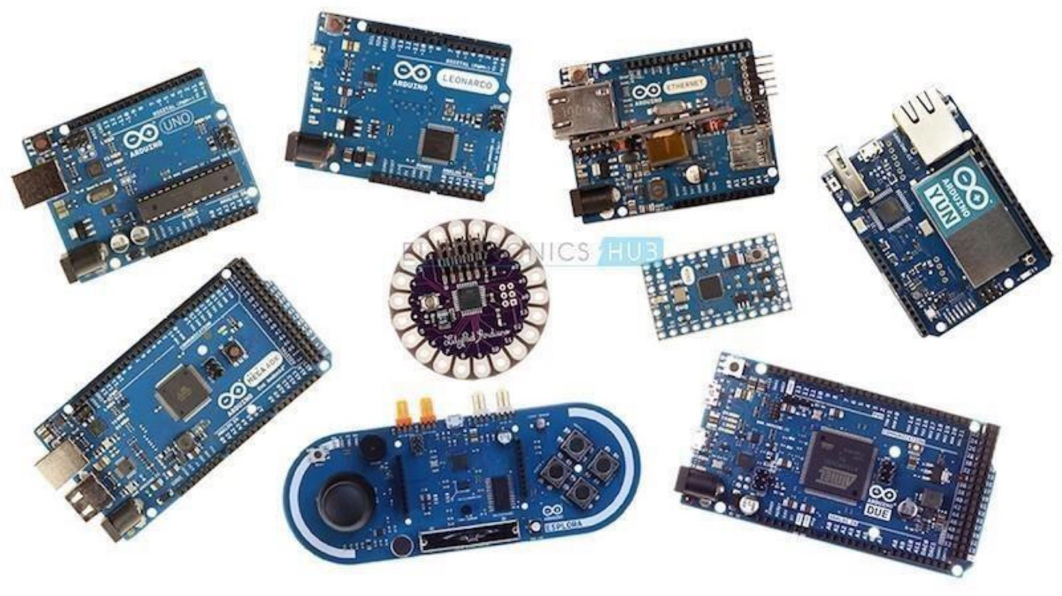
The important feature of the Arduino boards is the standard connectors. Using these connectors, we can connect the Arduino board to other devices like LEDs or add-on modules called Shields. The Arduino boards also consists of on board voltage regulator and crystal oscillator. They also consist of USB to serial adapter using which the Arduino board can be programmed using USB connection.

In order to program the Arduino board, we need to use IDE provided by Arduino. The Arduino IDE is based on Processing programming language and supports C and C++.

Types of Arduino Boards

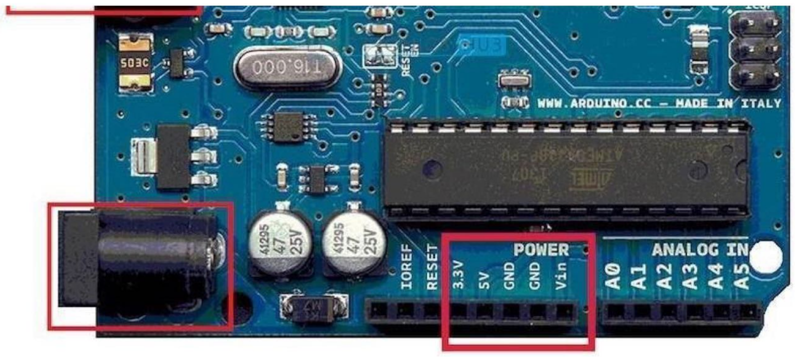
There are many types of Arduino boards available in the market but all the boards have one thing in common: they can be programmed using the Arduino IDE. The reasons for different types of boards are different power supply requirements, connectivity options, their applications etc.

Arduino boards are available in different sizes, form factors, different no. of I/O pins etc. Some of the commonly known and frequently used Arduino boards are Arduino UNO, Arduino Mega, Arduino Nano, Arduino Micro and Arduino Lilypad.



Arduino UNO

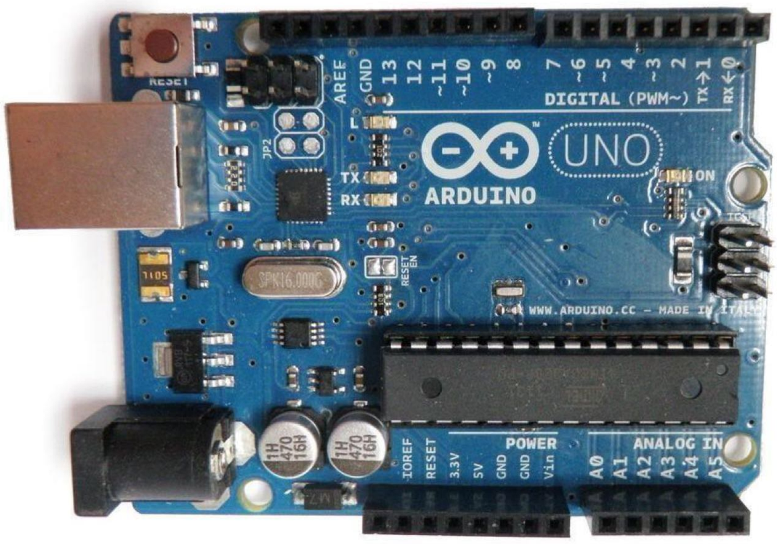
The most common version of Arduino is the Arduino Uno. This board is what most people are talking about when they refer to an Arduino. In the next step, there is a more complete rundown of its features.



Arduino Uno Features

Some people think of the entire Arduino board as a microcontroller, but this is inaccurate. The Arduino board actually is a specially designed circuit board for programming and prototyping with Atmel microcontrollers.
The nice thing about the Arduino board is that it is relatively cheap, plugs straight into a computer's USB port, and it is dead-simple to setup and use (compared to other development boards).

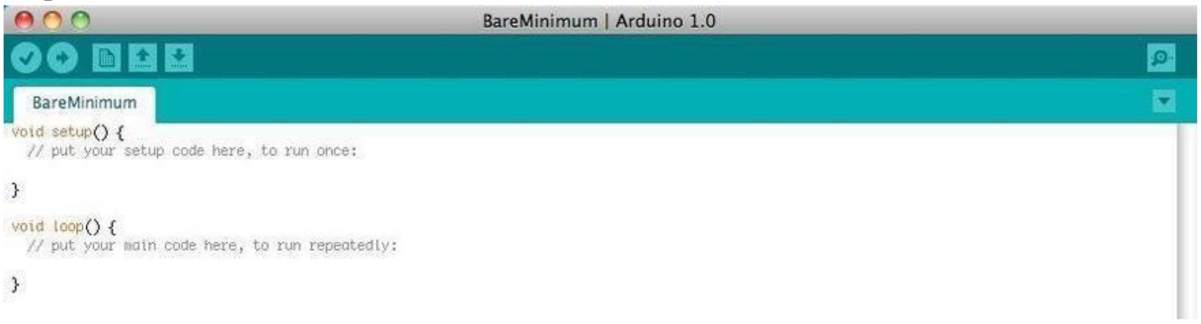
Some of the key features of the Arduino Uno include:
An open source design. The advantage of it being open source is that it has a large community of people using and troubleshooting it This makes it easy to find someone to help you debug your projects.



An easy USB interface . The chip on the board plugs straight into your USB port and registers on your computer as a virtual serial port. This allows you to interface with it as through it were a serial device. The benefit of this setup is that serial communication is an extremely easy (and time-tested) protocol, and USB makes connecting it to modern computers really convenient.

- Very convenient power management and built-in voltage regulation. You can connect an external power source of up to 12v and it will regulate it to both 5v and 3.3v. It also can be powered directly off of a USB port without any external power.
- A 16mhz clock. This makes it not the speediest microcontroller around, but fast enough for most applications.
- 32 KB of flash memory for storing your code.
- 13 digital pins and 6 analog pins. These pins allow you to connect external hardware to your Arduino. These pins are key for extending the computing capability of the Arduino into the real world. Simply plug your devices and sensors into the sockets that correspond to each of these pins and you are good to go.
- An ICSP connector for bypassing the USB port and interfacing the Arduino directly as a serial device. This port is necessary to re-bootload your chip if it corrupts and can no longer talk to your computer.
- An on-board LED attached to digital pin 13 for fast an easy debugging of code.

Step 3: Arduino IDE



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

}

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

}
```

Before you can start doing anything with the Arduino, you need to download and install the Arduino IDE (integrated development environment). From this point on we will be referring to the Arduino IDE as the Arduino Programmer.

The Arduino Programmer is based on the Processing IDE and uses a variation of the C and C++ programming languages.
You can find the most recent version of the Arduino Programmer on this page.

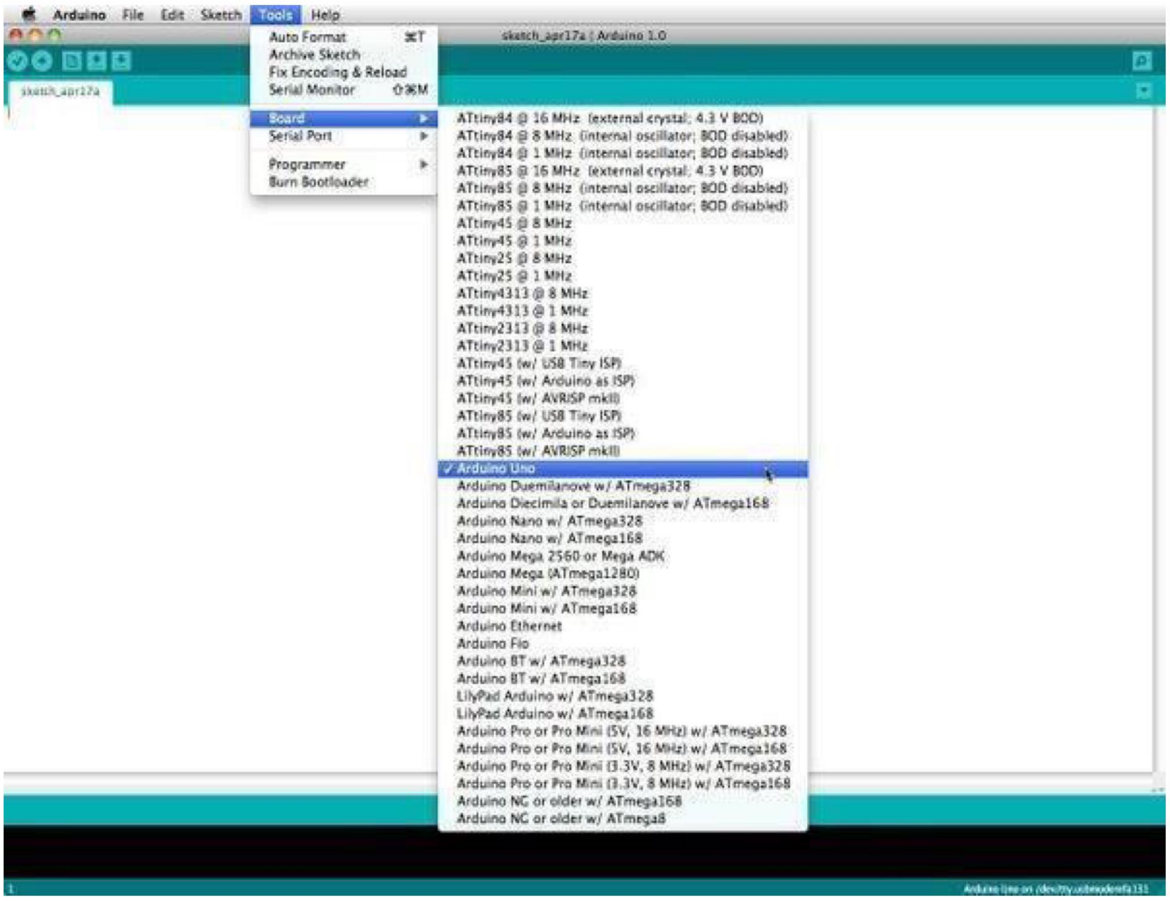
Step 4: Plug It In

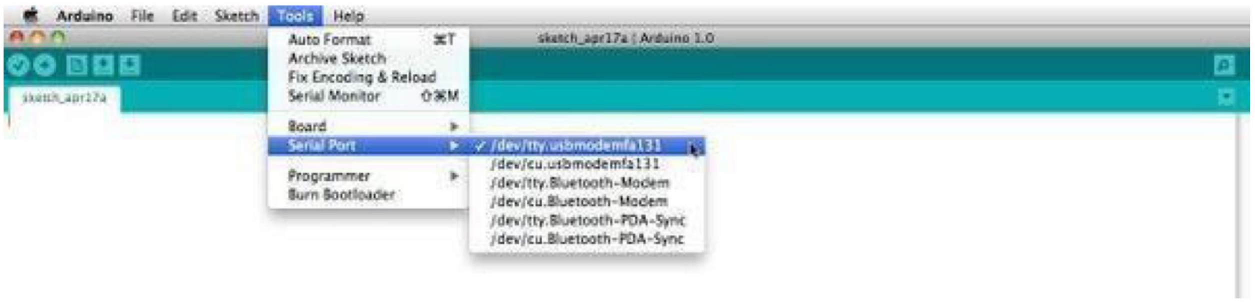
Connect the Arduino to your computer's USB port.

Please note that although the Arduino plugs into your computer, it is not a true USB device. The board has a special chip that allows it to show up on your computer as a virtual serial port whenit is plugged into a USB port. This is why it is important to plug the board in. When the board is
not plugged in, the virtual serial port that the Arduino operates upon will not be present
It is also good to know that every single Arduino has a unique virtual serial port address. This means that every time you plug in a different Arduino board into your computer, you will need toreconfigure the serial port that is in use.

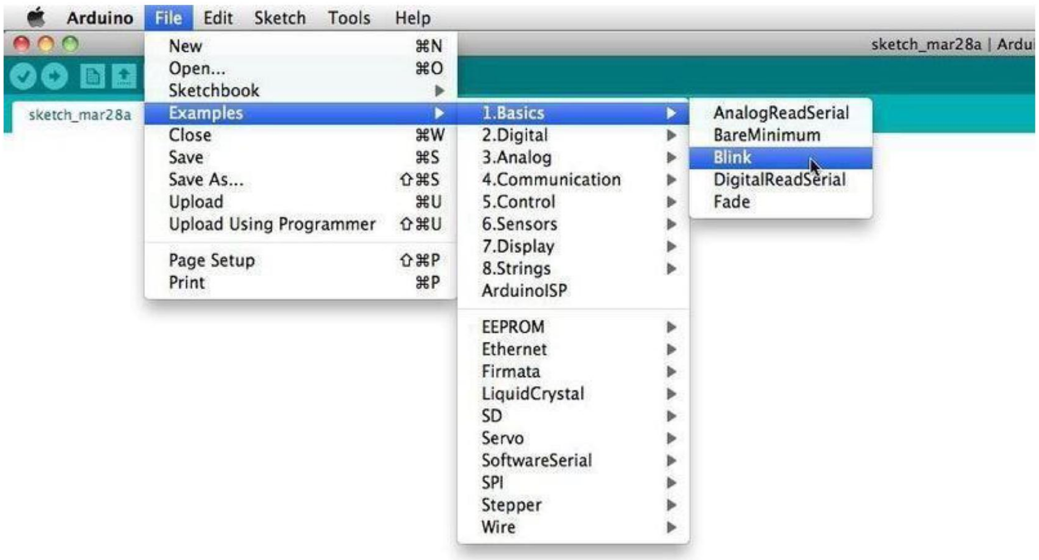
Step 5: Settings

Before you can start doing anything in the Arduino programmer, you must set the board-type andserial port.
To set the board, go to the following:
Tools --> Boards
Select the version of board that you are using. Since I have an Arduino Uno plugged in, Iobviously selected "Arduino Uno."
To set the serial port, go to the following:Tools --> Serial Port
Select the serial port that looks like:
/dev/tty.usbmodem [random numbers]


Arduino File Edit Sketch Tools Help
sketch_apr17a | Arduino 1.0
sketch_apr17a
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
ATtiny84 @ 16 MHz (external crystal; 4.3 V BOD)
ATtiny84 @ 8 MHz (internal oscillator; BOD disabled)
ATtiny84 @ 1 MHz (internal oscillator; BOD disabled)
ATtiny85 @ 16 MHz (external crystal; 4.3 V BOD)
ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)
ATtiny85 @ 1 MHz (internal oscillator; BOD disabled)
ATtiny45 @ 8 MHz
ATtiny45 @ 1 MHz
ATtiny25 @ 8 MHz
ATtiny25 @ 1 MHz
ATtiny4313 @ 8 MHz
ATtiny4313 @ 1 MHz
ATtiny2313 @ 8 MHz
ATtiny2313 @ 1 MHz
ATtiny45 (w/ USB Tiny ISP)
ATtiny45 (w/ Arduino as ISP)
ATtiny45 (w/ AVRISP mkII)
ATtiny85 (w/ USB Tiny ISP)
ATtiny85 (w/ Arduino as ISP)
ATtiny85 (w/ AVRISP mkII)
✓ Arduino Uno
Arduino Duemilanove w/ ATmega328
Arduino Diecimila or Duemilanove w/ ATmega168
Arduino Nano w/ ATmega328
Arduino Nano w/ ATmega168
Arduino Mega 2560 or Mega ADK
Arduino Mega (ATmega1280)
Arduino Mini w/ ATmega328
Arduino Mini w/ ATmega168
Arduino Ethernet
Arduino Fio
Arduino BT w/ ATmega328
Arduino BT w/ ATmega168
LilyPad Arduino w/ ATmega328
LilyPad Arduino w/ ATmega168
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
Arduino NG or older w/ ATmega168
Arduino NG or older w/ ATmega8
1
Arduino Uno on /dev/tty.usbmodemfa131


Arduino File Edit Sketch Tools Help
sketch_apr17a | Arduino 1.0
sketch_apr17a
Auto Format ⌘T
Archive Sketch
Fix Encoding & Reload
Serial Monitor ⇧⌘M
Board
Serial Port
Programmer
Burn Bootloader
✓ /dev/tty.usbmodemfa131
/dev/cu.usbmodemfa131
/dev/tty.Bluetooth-Modem
/dev/cu.Bluetooth-Modem
/dev/tty.Bluetooth-PDA-Sync
/dev/cu.Bluetooth-PDA-Sync

Step 6: Run a Sketch

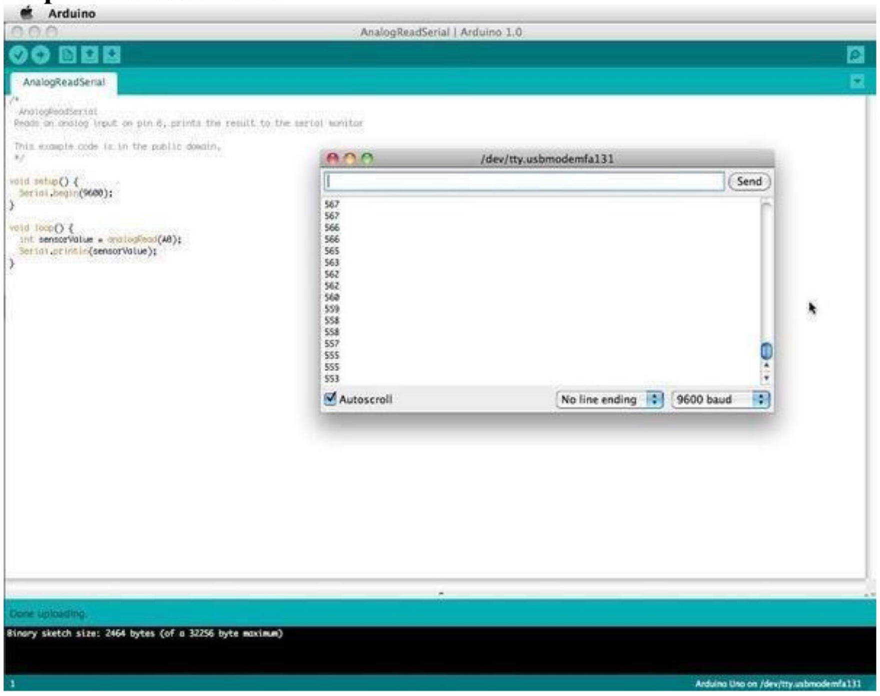


Arduino programs are called sketches. The Arduino programmer comes with a ton of example sketches preloaded. This is great because even if you have never programmed anything in your life, you can load one of these sketches and get the Arduino to do something.

To get the LED tied to digital pin 13 to blink on and off, let's load the blink example.
The blink example can be found here:
Files --> Examples --> Basics --> Blink
The blink example basically sets pin D13 as an output and then blinks the test LED on theArduino board on and off every second.

Once the blink example is open, it can be installed onto the ATMEGA328 chip by pressing theupload button, which looks like an arrow pointing to the right.
Notice that the surface mount status LED connected to pin 13 on the Arduino will start to blink. You can change the rate of the blinking by changing the length of the delay and pressing the upload button again.

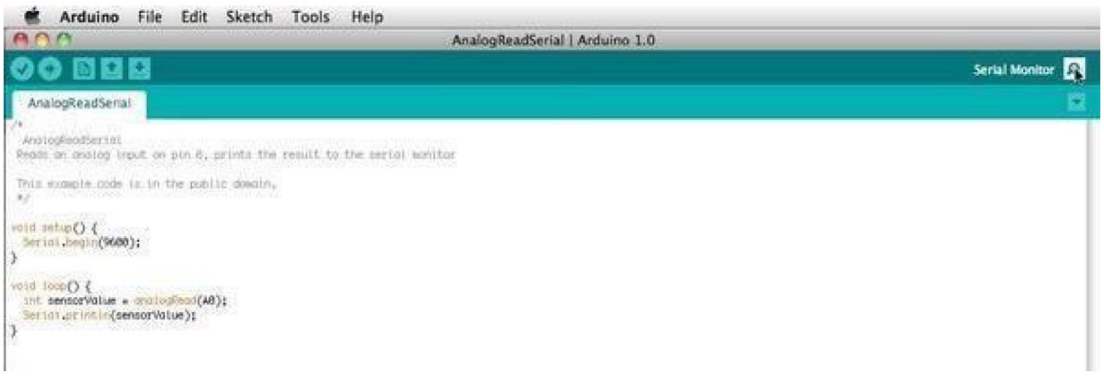
Step 7: Serial Monitor



```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

```
567
567
566
566
565
563
562
562
560
559
558
558
557
555
555
553
```

```
Done uploading.
Binary sketch size: 2464 bytes (of a 32256 byte maximum)
```



```
/*
  AnalogReadSerial
 Reads an analog input on pin 0, prints the result to the serial monitor

 This example code is in the public domain.
 */

void setup() {
  Serial.begin(9600);
}

void loop() {
  int sensorValue = analogRead(A0);
  Serial.println(sensorValue);
}
```

The serial monitor allows your computer to connect serially with the Arduino. This is important because it takes data that your Arduino is receiving from sensors and other devices and displaysit in real-time on your computer. Having this ability is invaluable to debug your code and understand what number values the chip is actually receiving.

For instance, connect center sweep (middle pin) of a potentiometer to A0, and the outer pins, respectively, to 5v and ground. Next upload the sketch shown below:
File --> Examples --> 1.Basics --> AnalogReadSerial Click the button to engage the serial monitor which looks like a magnifying glass. You can now see the numbers being read by the analog pin in the serial monitor. When you turn the knob the numbers will increase and decrease.
The numbers will be between the range of 0 and 1023. The reason for this is that the analog pinis converting a voltage between 0 and 5V to a discreet number.

Step 8: Digital In

The Arduino has two different types of input pins, those being analog and digital.To begin with, lets look at the digital input pins

Digital input pins only have two possible states, which are on or off. These two on and off states are also referred to as:

- HIGH or LOW
- 1 or 0
- 5V or 0V.

This input is commonly used to sense the presence of voltage when a switch is opened or closed. Digital inputs can also be used as the basis for countless digital communication protocols. By creating a 5V (HIGH) pulse or 0V (LOW) pulse, you can create a binary signal, the basis of all computing. This is useful for talking to digital sensors like a PING ultrasonic sensor, or communicating with other devices.

Step 9: Analog In

Aside from the digital input pins, the Arduino also boasts a number of analog input pins.
Analog input pins take an analog signal and perform a 10-bit analog-to-digital (ADC) conversionto turn it into a number between 0 and 1023 (4.9mV steps).
This type of input is good for reading resistive sensors. These are basically sensors whichprovide resistance to the circuit. They are also good for reading a varying voltage signal between0 and 5V. This is useful when interfacing with various types of analog circuitry.

If you followed the example in Step 7 for engaging the serial monitor, you have already tried using an analog input pin.

Step 10: Digital Out

A digital out pin can be set to be HIGH (5v) or LOW (0v). This allows you to turn things on and off. Aside from turning things on and off (and making LEDs blink), this form of output is convenientfor a number of applications.
Most notably, it allows you to communicate digitally. By turning the pin on and off rapidly, you are creating binary states (0 and 1), which is recognized by countless other electronic devices asa binary signal. By using this method, you can communicate using a number of differentprotocols.
Digital communication is an advanced topic, but to get a general idea of what can be done, checkout the Interfacing With Hardware page.
If you followed the example in Step 6 for getting an LED to blink, you have already tried using adigital output pin.

Step 11: Analog Out

As mentioned earlier, the Arduino has a number of built in special functions. One of thesespecial functions is pulse width modulation, which is the way an Arduino is able to create an analog-like output.
Pulse width modulation - or PWM for short - works by rapidly turning the PWM pin high (5V) and low (0V) to simulate an analog signal. For instance, if you were to blink an LED on and off rapidly enough (about five milliseconds each), it would seem to average the brightness and only appear to be receiving half the power. Alternately, if it were to blink on for 1 millisecond and then blink off for 9 millisecond, the LED would appear to be 1/10 as bright and only be receiving1/10 the voltage.
PWM is key for a number of applications including making sound, controlling the brightness of lights, and controlling the speed of motors.
To try out PWM yourself, connect an LED and 220 ohm resistor to digital pin 9, in series toground. Run the following example code:
File --> Examples --> 3.Analog --> Fading