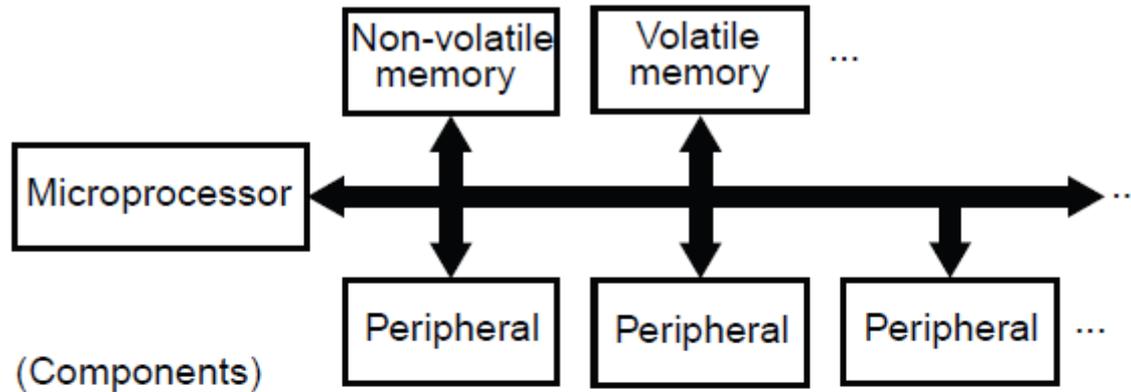
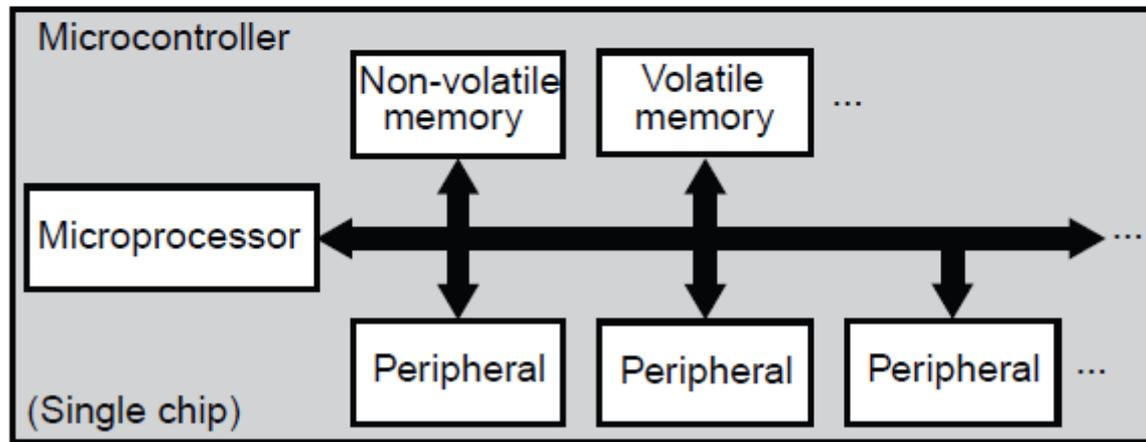


Computer vs. microcontroller

What are the differences between a generic computer and a microcontroller?



(a) General computer.



(b) Microcontroller.

Almost a smartphone: ARM Cortex A

Application (A) oriented processors are systems on a chip (SoC):

- Midway between a generic computer and microcontroller.
- Has an operating system (e.g. Linux).
- Does not have a non-volatile memory to store programs.
- Choice of peripherals not suited for process control.
- More expensive than microcontrollers



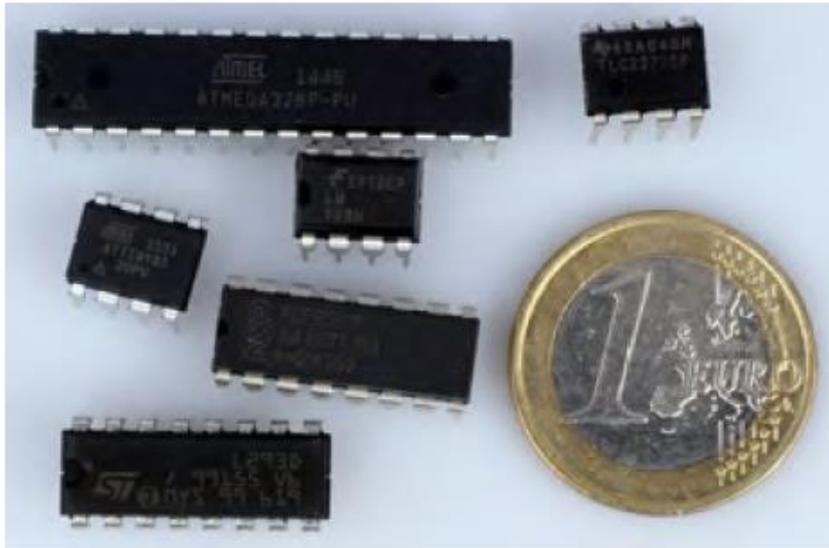
(a) Raspberry Pi.



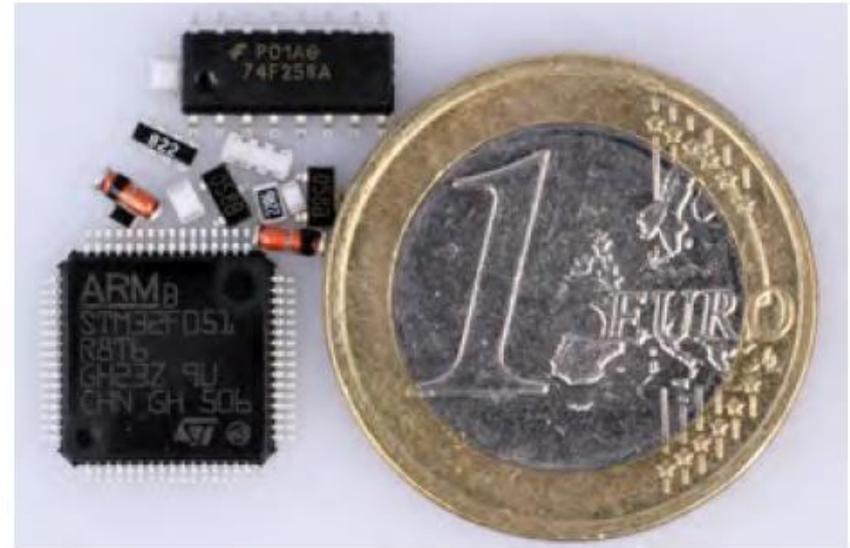
(b) BeagleBoard-xm.

Figure: Prototyping boards with high performance ARM Cortex A processors.

Electronics packages



(a) DIP package



(b) SMD package

Figure: Component packages: DIP and SMD compared to a coin for scale.

Microcontroller powerhouses: ARM Cortex M

M as in microcontroller unit (MCU).

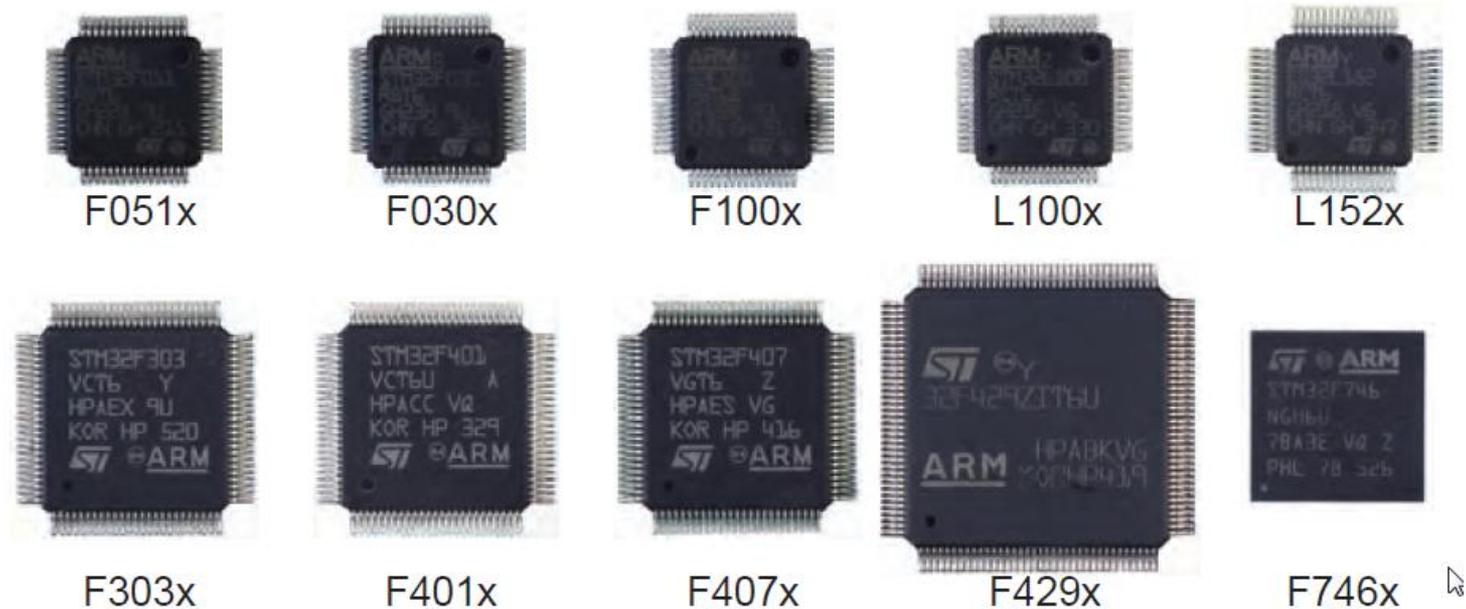


Figure: Microcontrollers with 32-bit ARM Cortex M architecture, manufactured by ST Microelectronics. The numbers designate the type of the microchip.

ARM Cortex M development boards



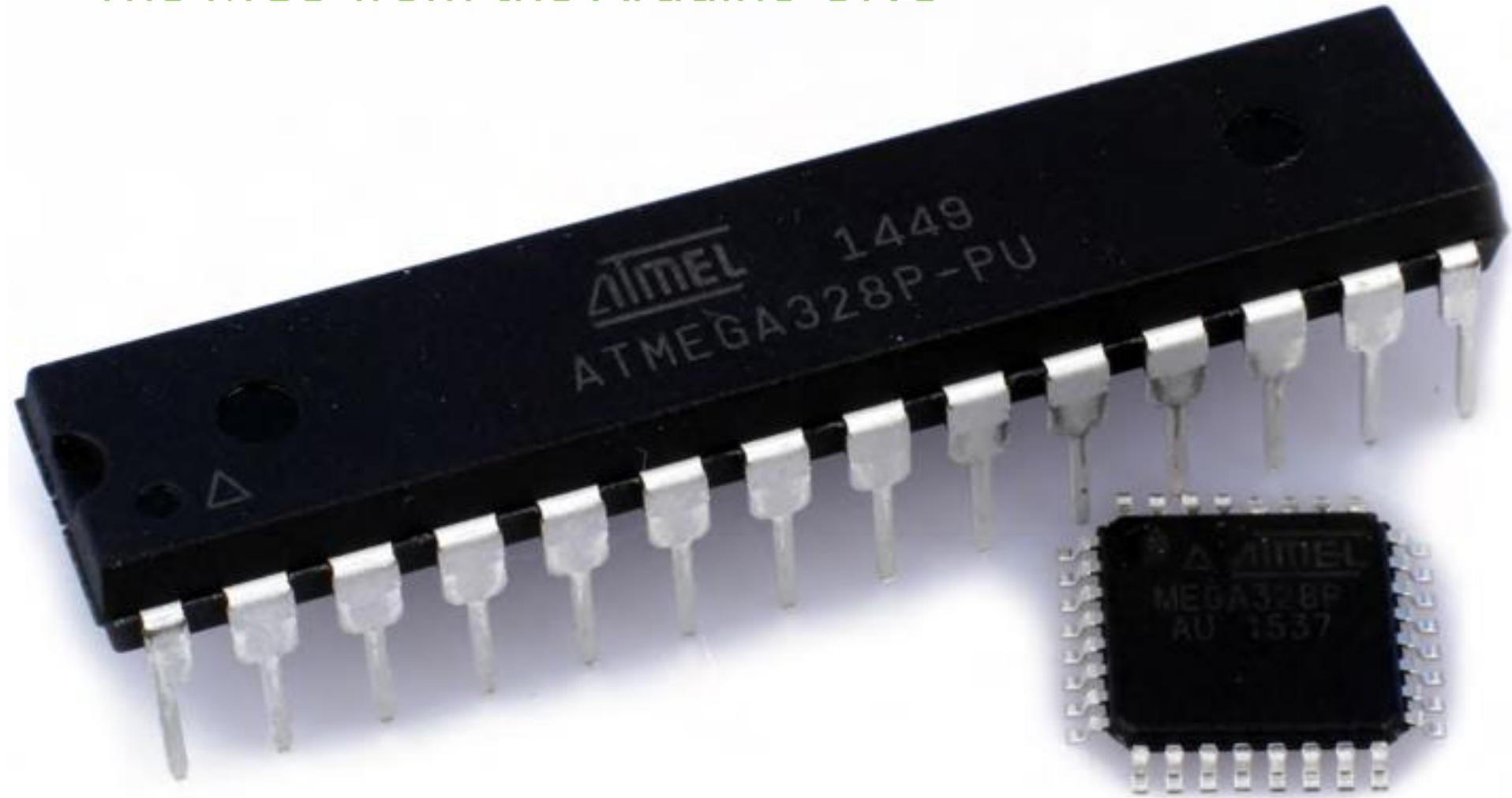
(a) ST Microelectronics Discovery.



(b) Freescale Freedom.

Figure: Development boards with 32-bit ARM Cortex M architecture microcontrollers.

The MCU from the Arduino UNO



Arduino tour

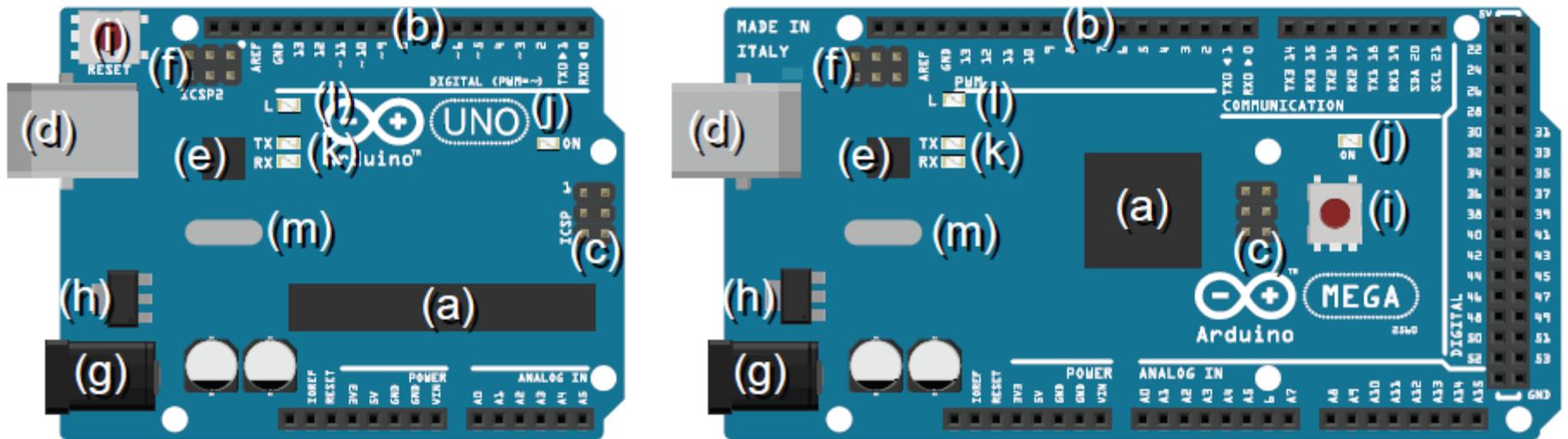


Figure: Overview of the Arduino Uno and Mega 2560 hardware.

Meet the AVR microcontroller

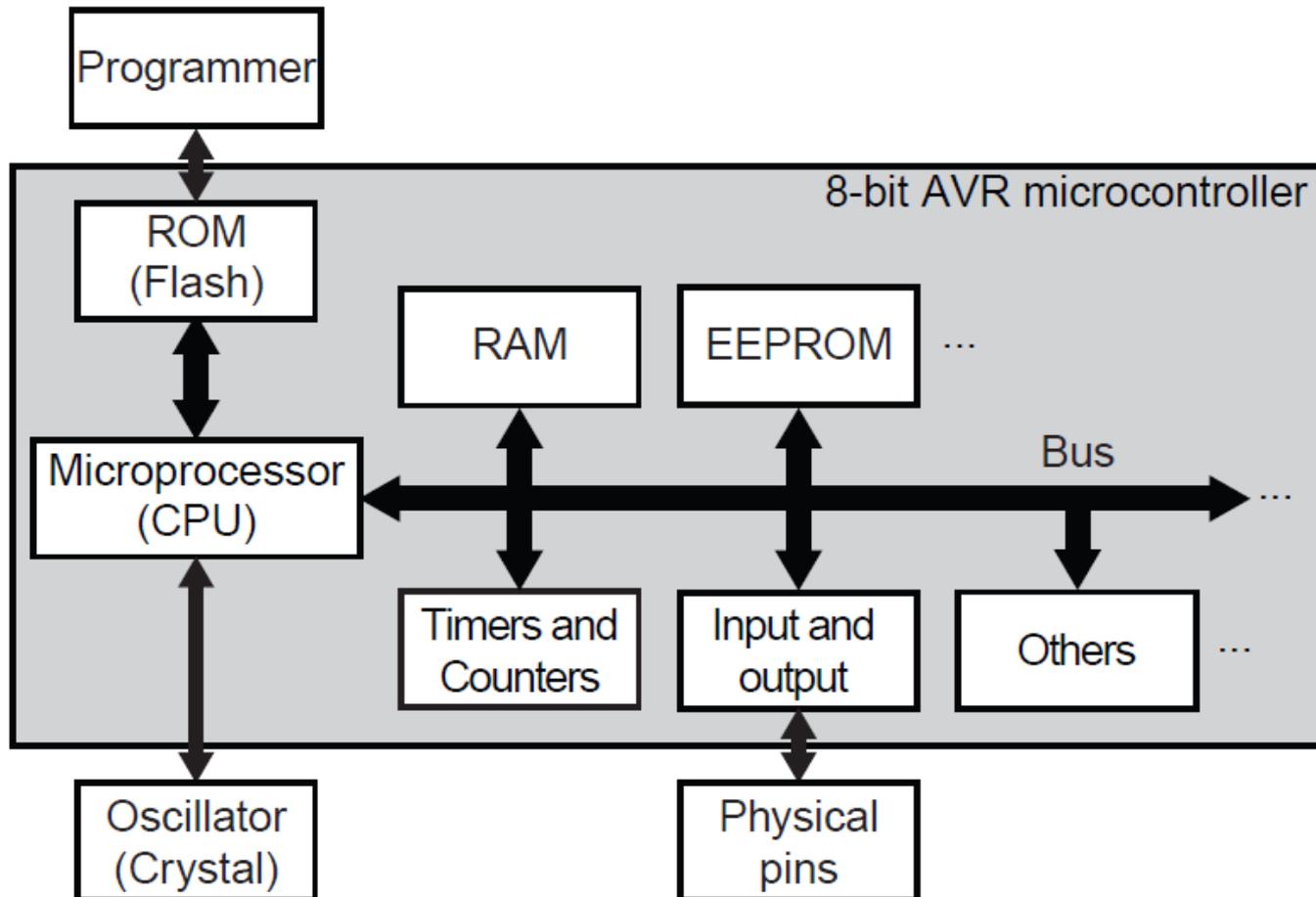


Figure: Simplified schematics of the AVR ATmega-series microcontrollers located in most Arduino boards.

Arduino Uno specs.

- 8-bit RISC architecture
- 32 kB ROM (Flash)
- 2 kB RAM
- 16 MHz clock speed (max 20 MHz for the MCU)
- 14 digital pins, 6 analog
- Programmed and powered by USB
- Can be powered and programmed externally too!

Arduino shields



(a) Arduino Ethernet Shield.



(b) Arduino Motor Shield.

Figure: Expansion boards made by Arduino.

Third party shields



(a) Prototyping shield.



(b) Adafruit Data Logging Shield.



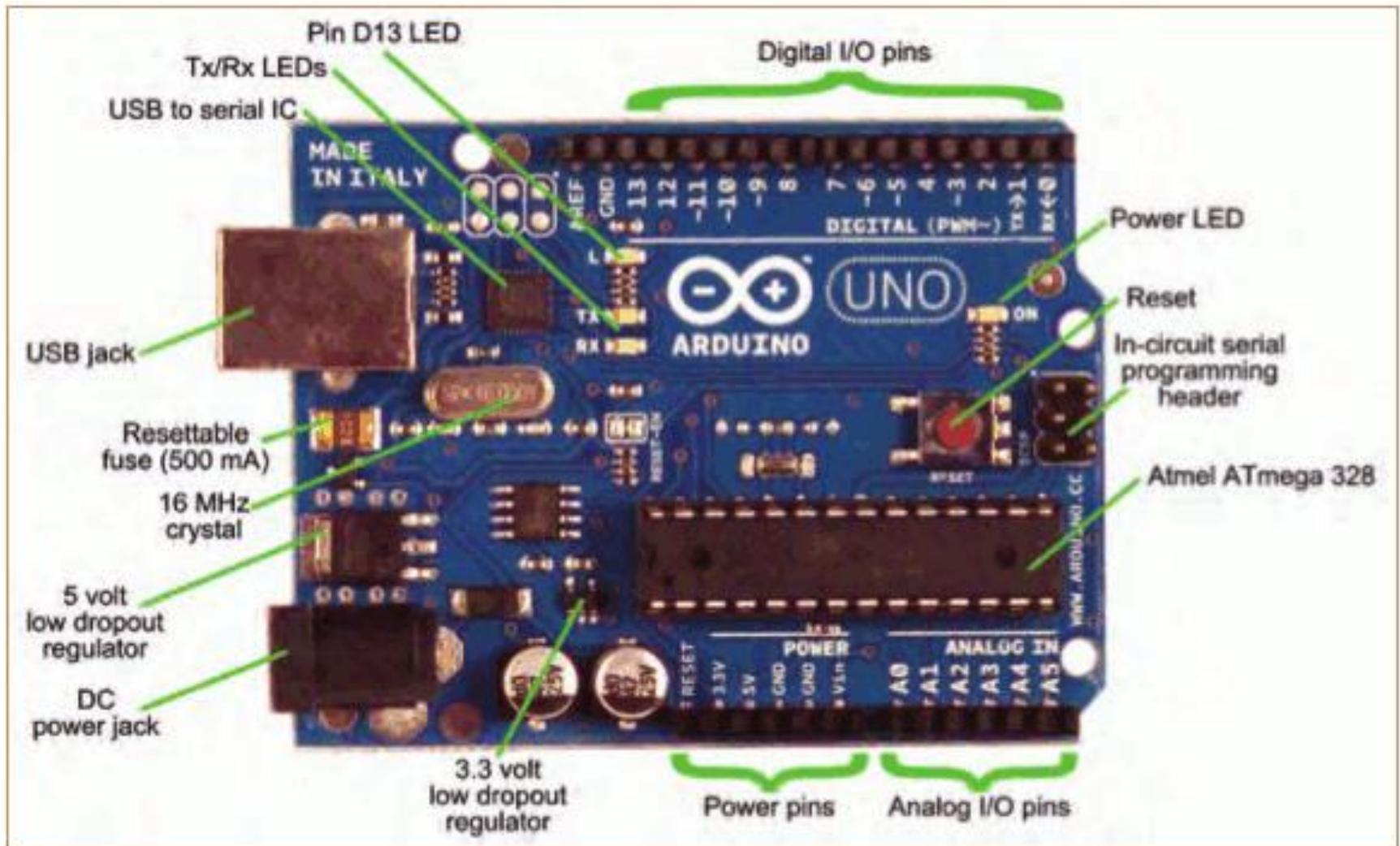
(c) Digilent Analog Shield.



(d) Seed Studio XBee Shield.

Meet Arduino Uno

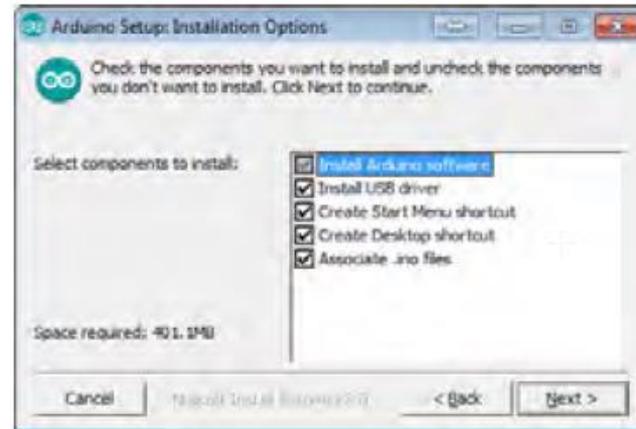
Open Source electronic prototyping platform based on flexible easy to use hardware and software.



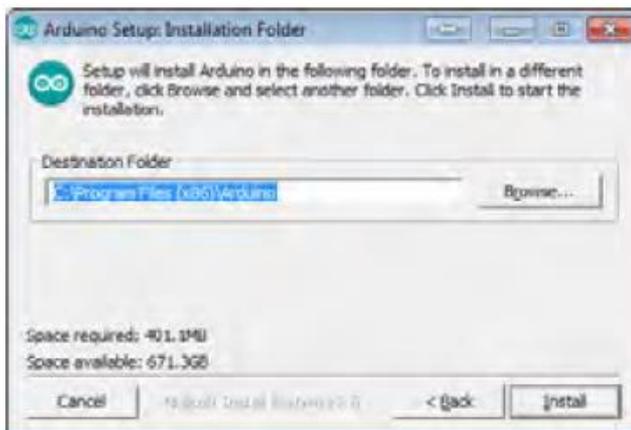
Installation (Windows)



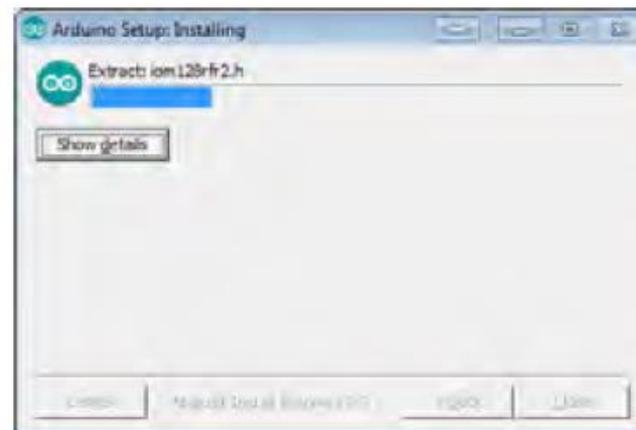
(a) License agreement.



(b) Options.



(c) Installation folder.



(d) Installation progress.

Drivers

- Windows will automatically install drivers on first connection.
- Linux, macOS does not need drivers.



(g) Searching for the driver.



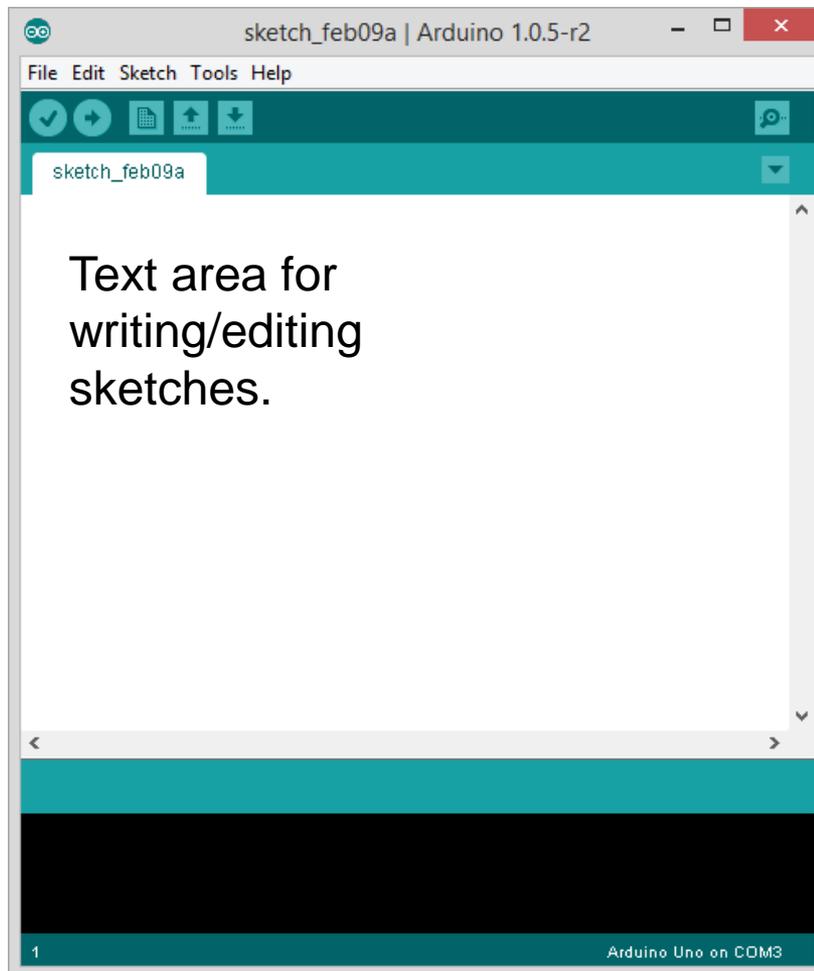
(h) Drivers found and installed.

Figure: A Windows machine will automatically install the drivers necessary to talk to your Arduino board.

Introduction to Software

- Arduino microcontrollers are programmed using the Arduino IDE (Integrated Development Environment)
- Can be downloaded for free from <http://arduino.cc/en/Main/Software>
- Arduino programs, called “sketches”, are written in a programming language similar to C and C++
- Every sketch must have a `setup()` function (executed just once) followed by a `loop()` function (potentially executed many times);
- `//` makes “comments”. In order to code to make it easier to read (technically optional, but actually required (by me))
- Many sensors and other hardware devices come with prewritten software – look on-line for sample code, libraries (of functions), and tutorials

Parts of the IDE main screen



← Name of current sketch

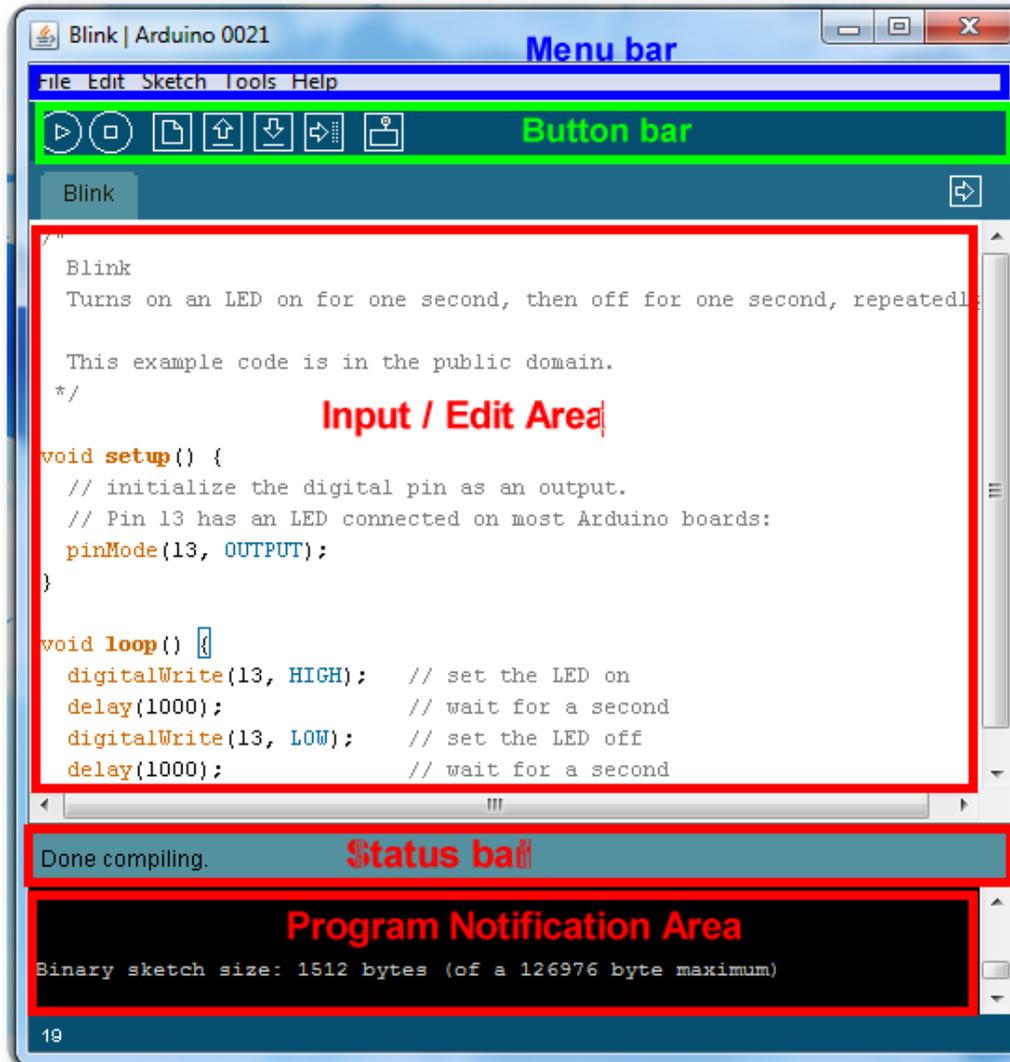
← Main menus

← Action buttons/icons

-  Verify (AKA compile)
-  Upload (send to Arduino)
-  Start a new sketch
-  Open a sketch (from a file)
-  Save current sketch (to a file)
-  Open Serial Monitor window

← Error messages and other feedback show up here.

Arduino IDE



See: <http://arduino.cc/en/Guide/Environment> for more information

Empty sketch

```
1 void setup() {  
2     // put your setup code here, to run once:  
3 }  
4  
5 void loop() {  
6     // put your main code here, to run ...  
7     repeatedly:  
7 }
```

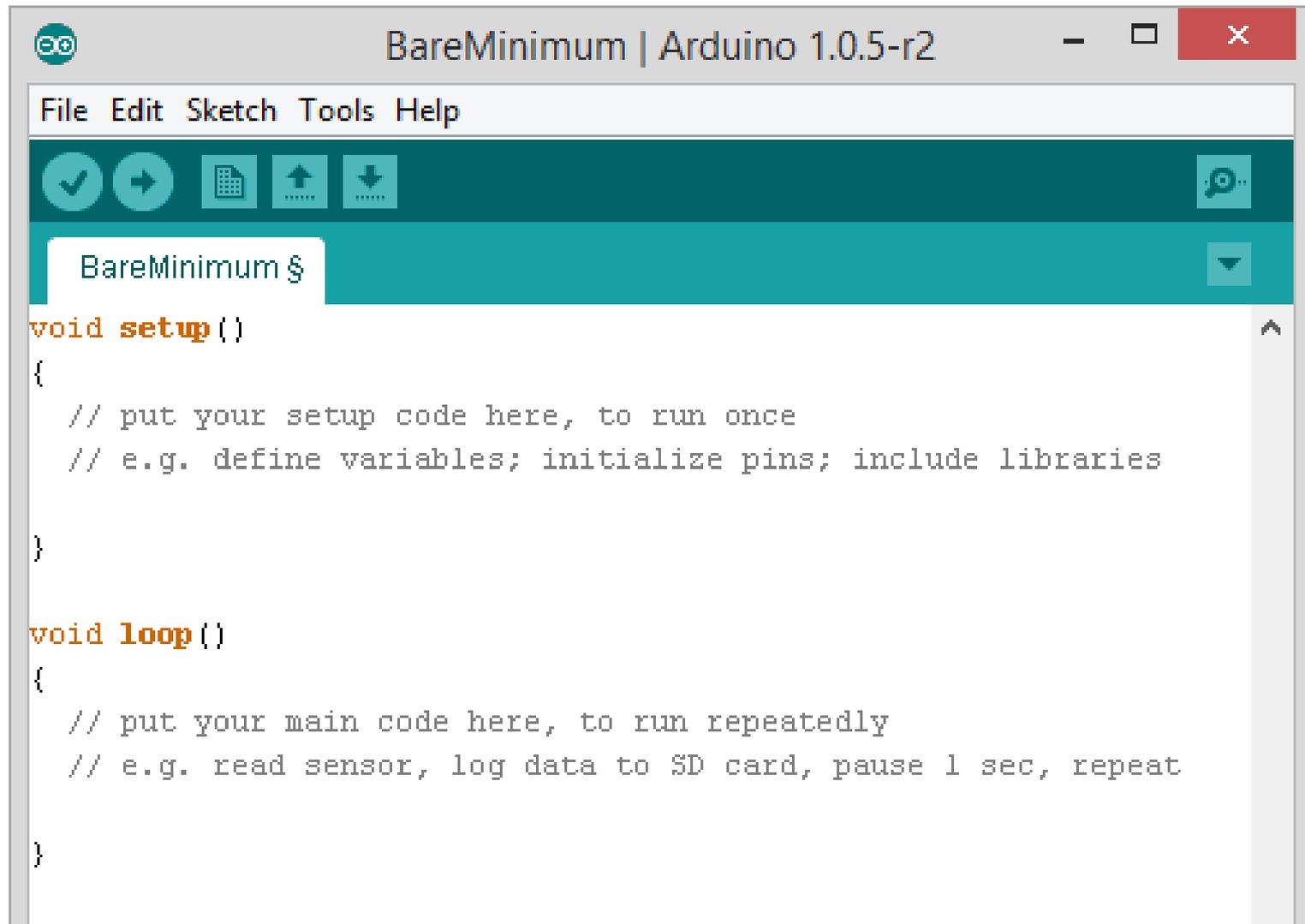
Bare minimum code

`setup` : `setup()` function is called when a sketch starts. Use it to initialize the variables, pin modes, start using libraries, etc. The `setup` function will only run once, after each power up or reset of the Arduino board.

`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.

`void` keyword is used only in function declarations. It indicates that the function is expected to return no information to the function from which it was called.

BareMinimum – sketch organization



The image shows a screenshot of the Arduino IDE window titled "BareMinimum | Arduino 1.0.5-r2". The window has a menu bar with "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for checkmark, play, grid, upload, and download. A dropdown menu is open, showing "BareMinimum \$". The main text area contains the following code:

```
void setup()  
{  
  // put your setup code here, to run once  
  // e.g. define variables; initialize pins; include libraries  
}  
  
void loop()  
{  
  // put your main code here, to run repeatedly  
  // e.g. read sensor, log data to SD card, pause 1 sec, repeat  
}
```

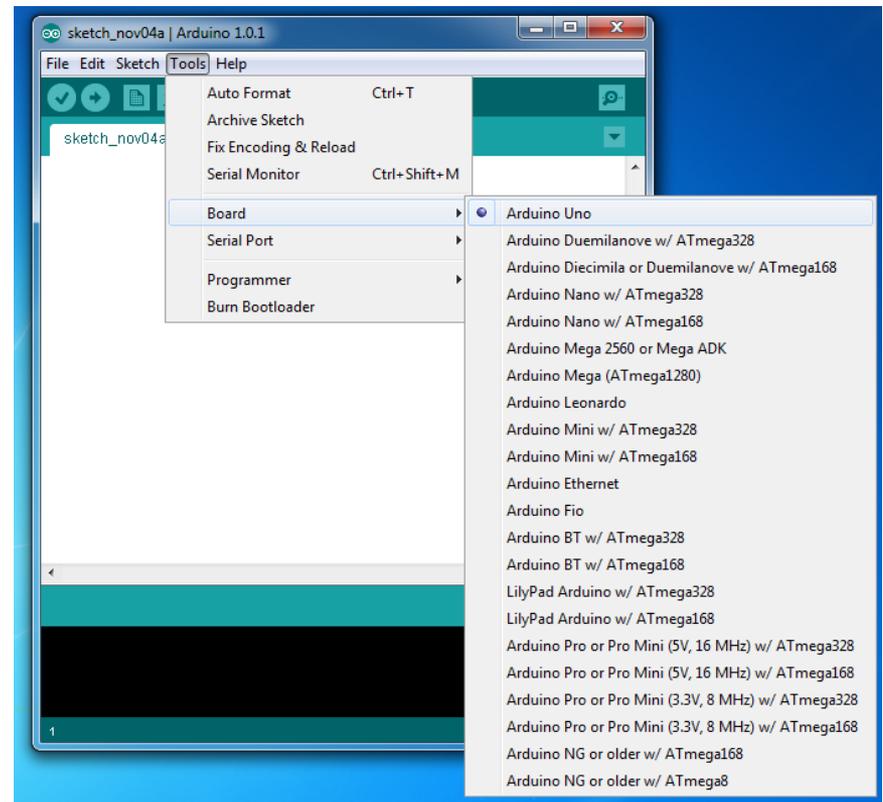
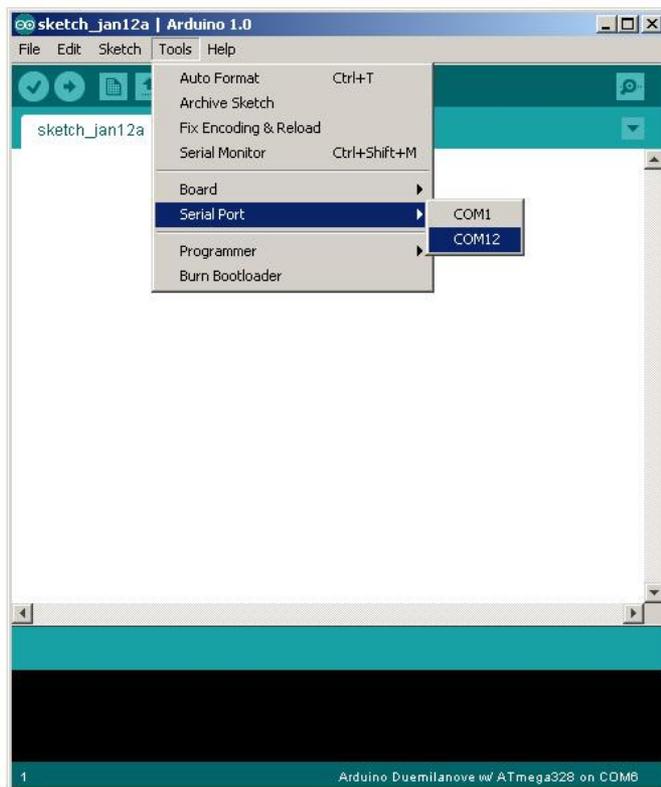
Verify compilation

Verify the syntax of an empty sketch with an empty `setup()` and `loop()` function.

- Click the checkbox icon.
- “Compiling sketch...” —look at the progress at the command line.
- Arduino IDE reports ROM and RAM usage.
- Program is compiled and checked for syntax errors, but is not uploaded to the Arduino board.

Select serial port and board

- You must select the virtual serial port in the Arduino IDE
- Port used for communicating with the Arduino.
- Tools -> Port -> (Select port).
- Linux and macOS have different names than COM1:
 - ▶ Linux `/dev/ttyACMXXX`.
 - ▶ Mac: `/dev/cu.usbmodemXXXX`.

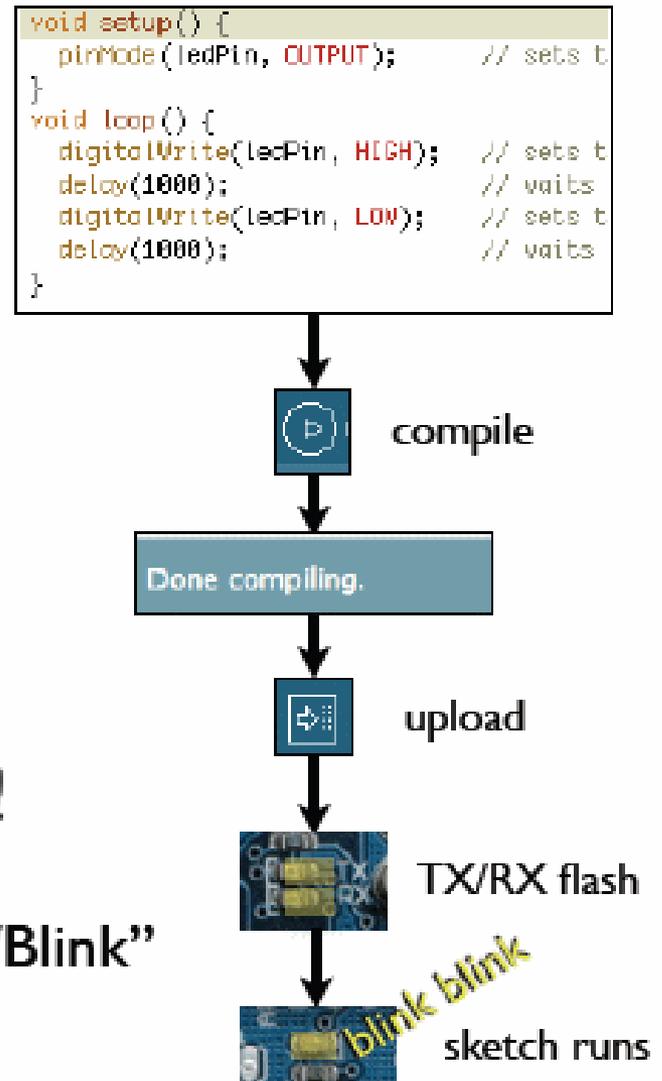


Using Arduino

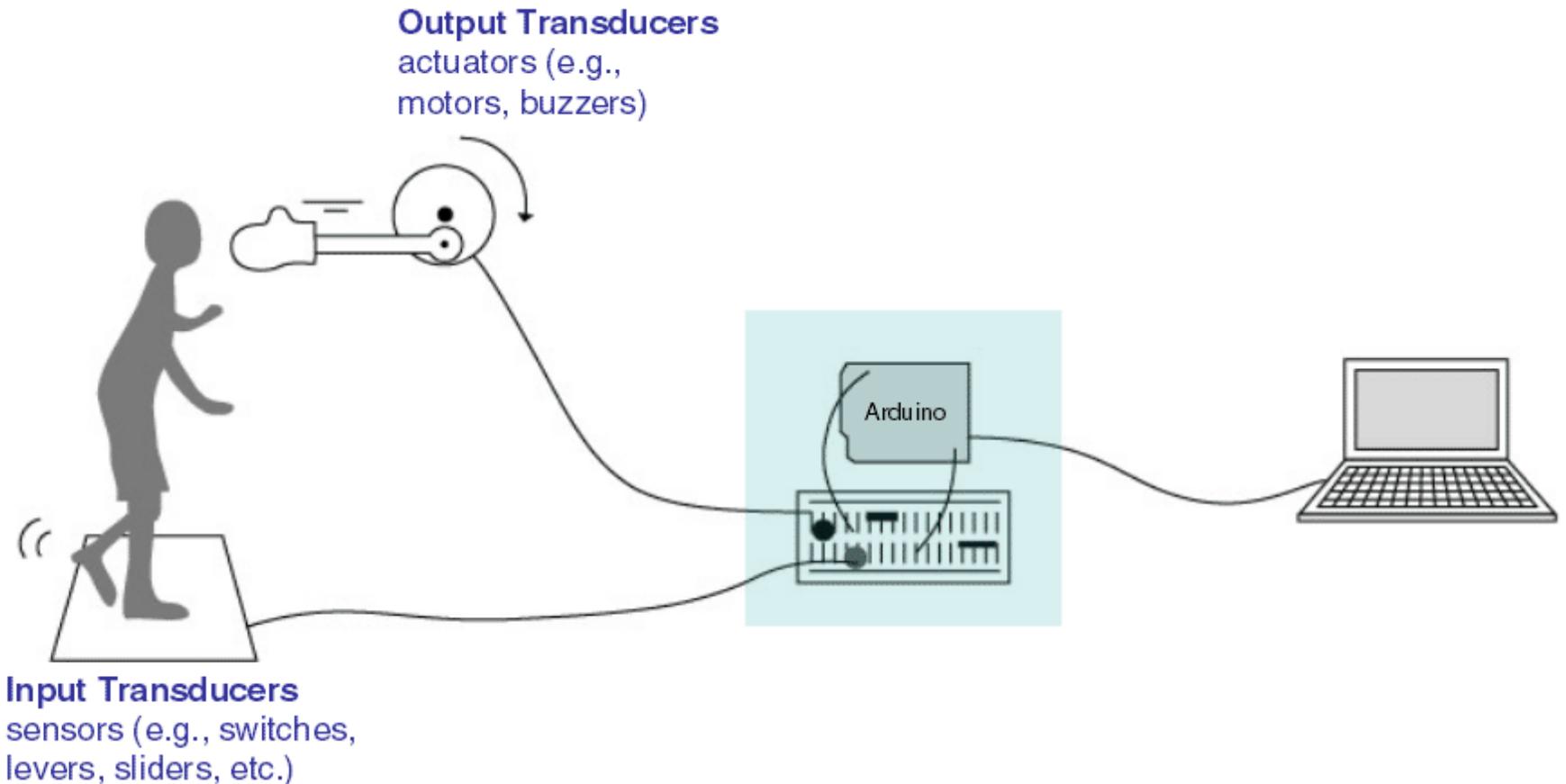
- Write your sketch
- Press Compile button (to check for errors)
- Press Upload button to program Arduino board with your sketch

Try it out with the “Blink” sketch!

Load “File/Sketchbook/Examples/Digital/Blink”

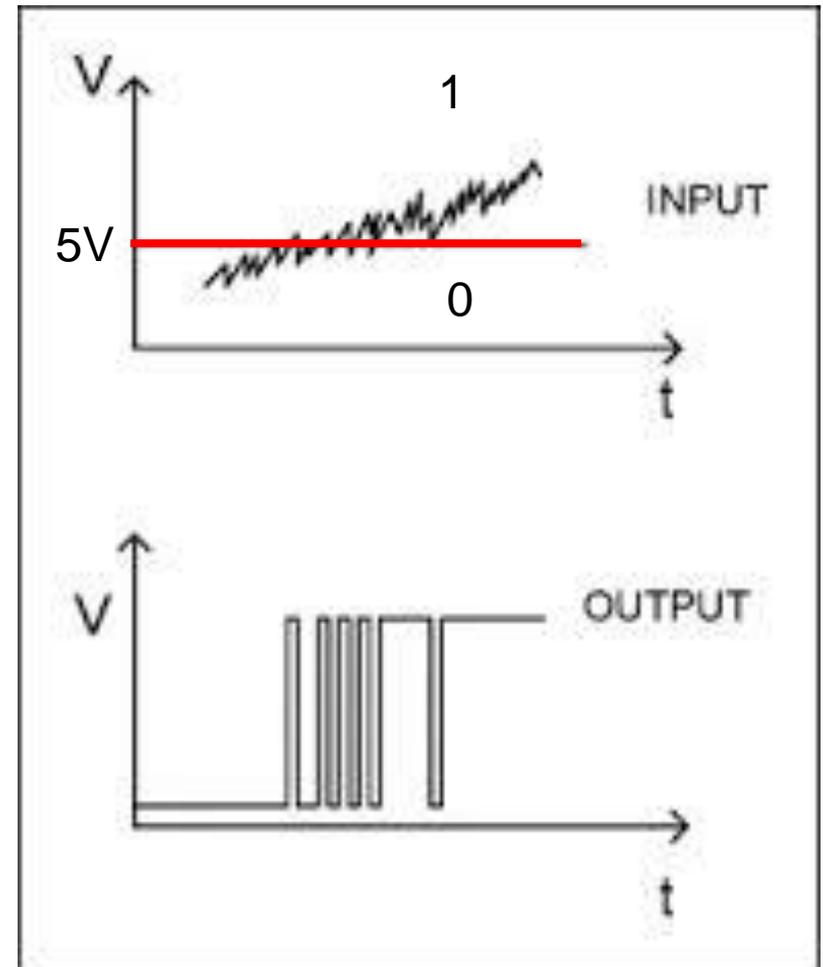


Input/Output

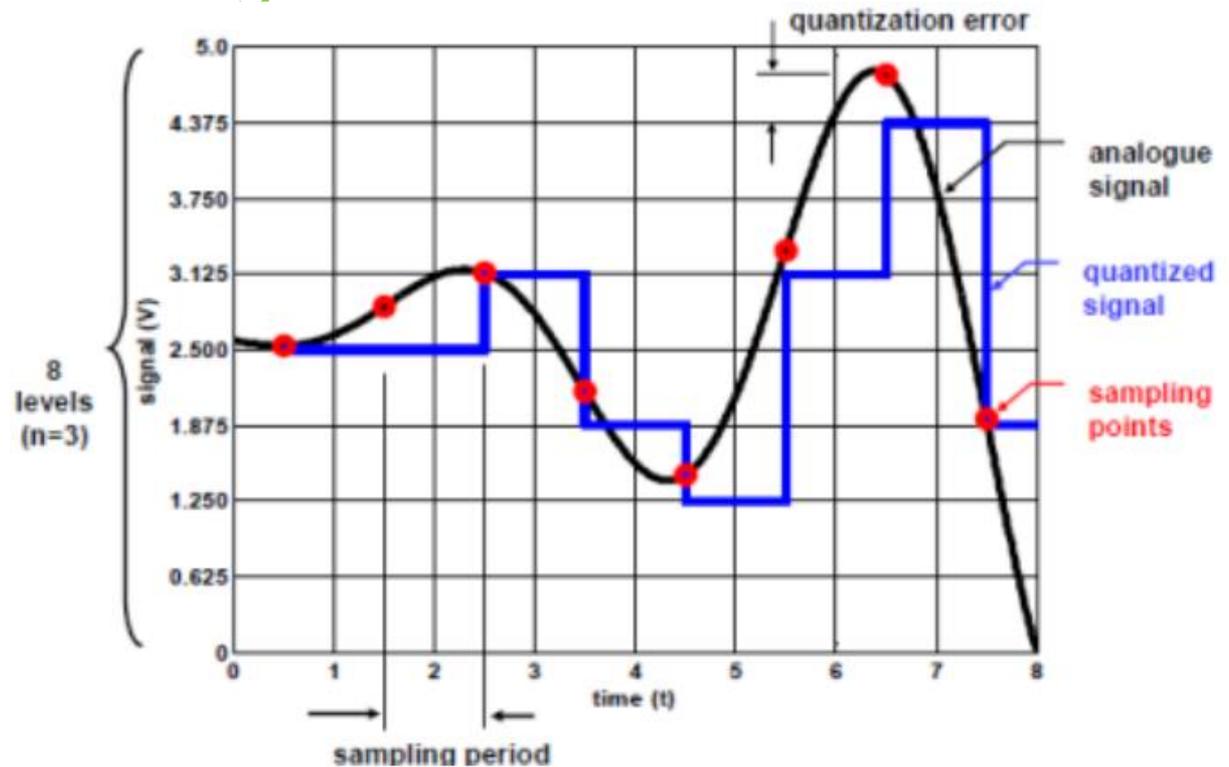


Digital Input/Output

- What is analog what is digital?
- Analog is continuous range of voltage values (not just 0 or 5V)
- Digital IO is binary valued - it's either on or off, 1 or 0
- Internally, all microprocessors are digital.



Quantization the signal



Resolution: the number of different voltage levels (i.e., *states*) used to discretize an input signal

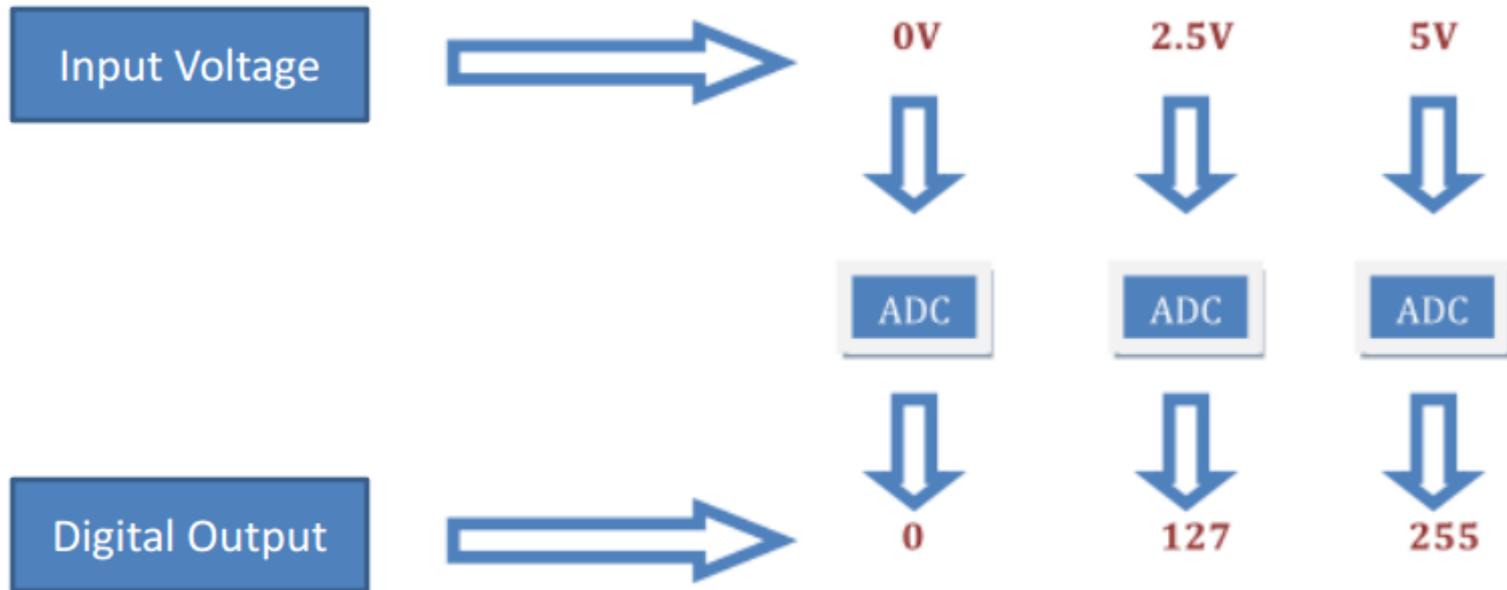
Resolution values range from 256 states (8 bits) to 4,294,967,296 states (32 bits)

The Arduino uses 1024 states (10 bits)

Smallest measurable voltage change is $5V/1024$ or 4.8 mV

Maximum sample rate is 10,000

Converting Analog Value to Digital



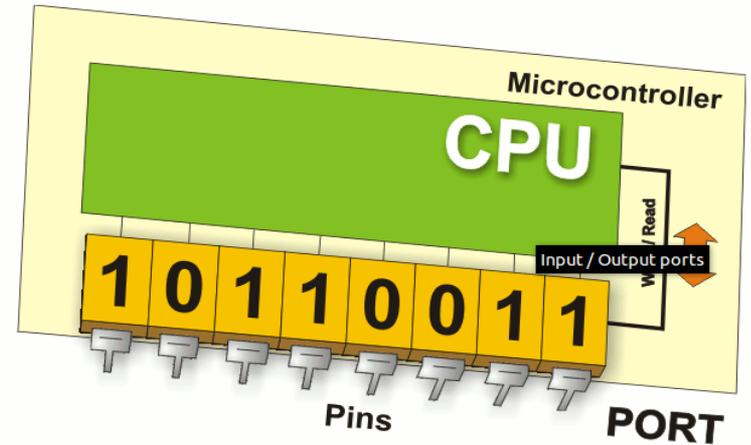
ADC in Arduino

- The Arduino Uno board contains 6 pins for ADC
- 10-bit analog to digital converter
- This means that it will map input voltages between 0 and 5 volts into integer values between 0 and 1023

Important functions

- `Serial.println(value);`
Prints the value to the Serial Monitor on your computer
- `pinMode(pin, mode);`
Configures a digital pin to read (input) or write (output) a digital value
- `digitalRead(pin);`
Reads a digital value (HIGH or LOW) on a pin set for input
- `digitalWrite(pin, value);`
Writes the digital value (HIGH or LOW) to a pin set for output

Arduino Digital I/O



`pinMode(pin, mode)`

Sets pin to either INPUT or OUTPUT

`digitalRead(pin)`

Reads HIGH or LOW from a pin

`digitalWrite(pin, value)`

Writes HIGH or LOW to a pin

Electronic stuff

Output pins can provide 40 mA of current

PinMode

- A pin on arduino can be set as input or output by using pinMode function.
- `pinMode(13, OUTPUT);` // sets pin 13 as output pin
- `pinMode(13, INPUT);` // sets pin 13 as input pin

Reading/writing digital values

- `digitalWrite(13, LOW); // Makes the output voltage on pin 13 , 0V`
- `digitalWrite(13, HIGH); // Makes the output voltage on pin 13 , 5V`
- `int buttonState = digitalRead(2); // reads the value of pin 2 in buttonState`

Reading/Writing Analog Values

- `analogRead(A0); // used to read the analog value from the pin A0`
- Analog output 0-5V
5V => 255
0V => 0
- Pin 3 gives 5V
`analogWrite(3, 255);`

analogWrite();

5V	255
4V	x

$$x = (4 * 255) / 5 = 204$$

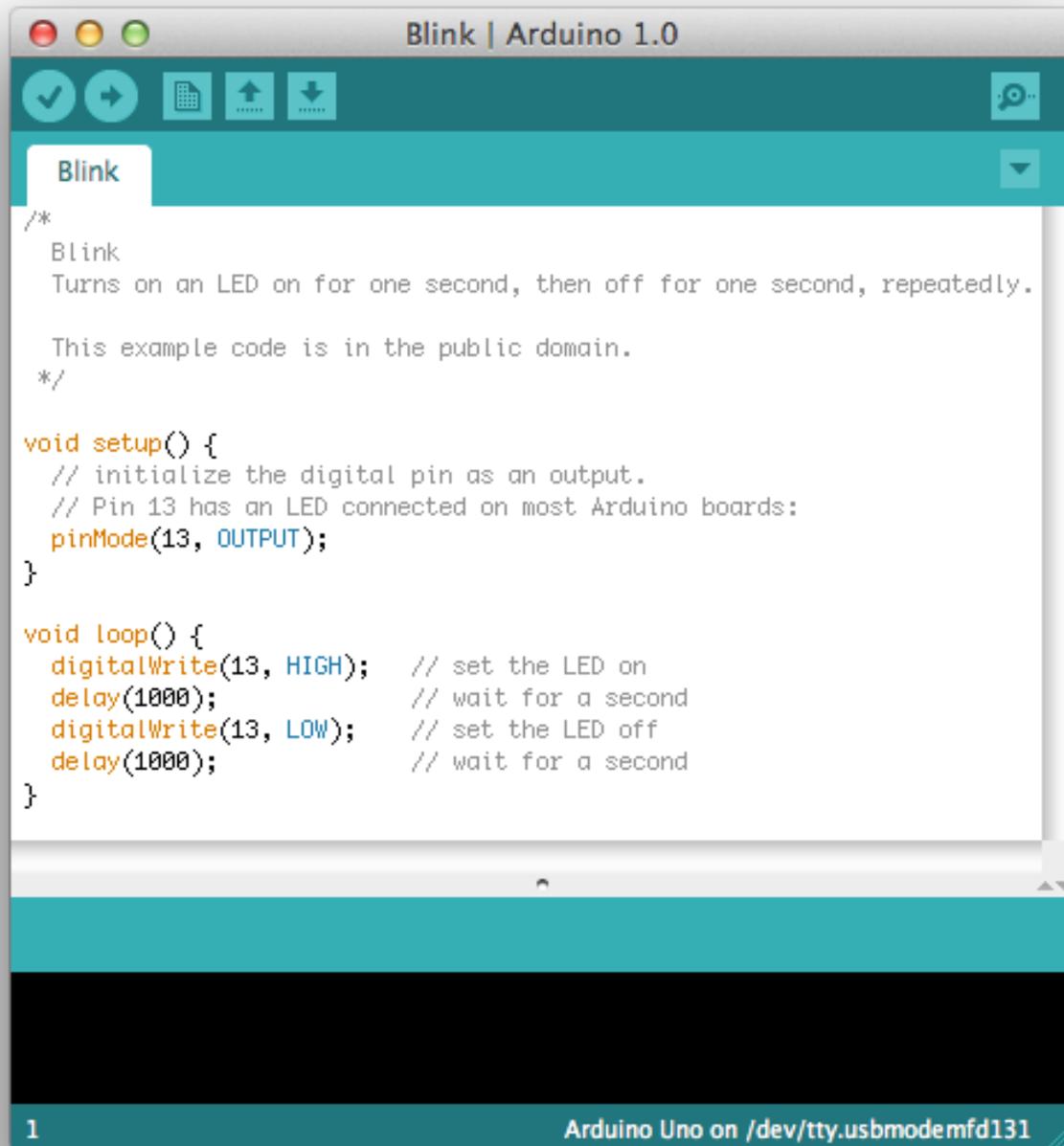
analogWrite(6 , 204);

Blinking LED

To blink an LED is the “Hello world” example for microcontrollers. This example helps to familiarize you with the basic syntax and verifies successful deployment (See: *ledblink/ledblink.ino*).

```
1 void setup() {
2   pinMode(D13, OUTPUT);    // Pin D13 out
3 }
4
5 void loop() {
6   digitalWrite(D13, HIGH); // turn the on
7   delay(1000);             // wait 1 s
8   digitalWrite(D13, LOW);  // turn off
9   delay(1000);             // wait 1 s
10 }
```

Our First Program



The image shows a screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 1.0". The code editor displays the following C++ code for a program named "Blink":

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

  This example code is in the public domain.
  */

void setup() {
  // initialize the digital pin as an output.
  // Pin 13 has an LED connected on most Arduino boards:
  pinMode(13, OUTPUT);
}

void loop() {
  digitalWrite(13, HIGH); // set the LED on
  delay(1000);           // wait for a second
  digitalWrite(13, LOW); // set the LED off
  delay(1000);          // wait for a second
}
```

At the bottom of the IDE, the status bar shows "1" on the left and "Arduino Uno on /dev/tty.usbmodemfd131" on the right.

Making an on-board and external LED blink

- Place an LED on the breadboard (don't put both legs into a single (connected) column)
- Wire the LED's negative (shorter) lead to a 560 Ohm "safety resistor" then wire the other end of the resistor to ground (GND)
- Wire the LED's positive (longer) lead to digital Pin 13 on the Arduino
- Plug in the Arduino with the USB cable and run the Arduino IDE software on the computer
- Under `Tools:Board` make sure `Arduino Uno` is selected
- Under `Tools:Serial Port` select the correct COM port (e.g. COM3)
- Look at the code. Note `delay(1000)` waits 1000 millisec = 1 second
- Verify (AKA Compile) the code , then Upload it to the Uno – notice that it runs immediately (and will re-run if you power cycle) – the sketch will stay in the Uno memory until overwritten or erased
- Discuss how the sketch works (in general terms: `setup()`, `loop()`, liberal use of comments); digital HIGH is 5V and LOW is 0V (ground)
- This is an example of "control (output) using digital pin 13"

Using LEDs

```
void setup()
{
  pinMode(77, OUTPUT);    //configure pin 77 as output
}
// blink an LED once
void blink()
{
  digitalWrite(77,HIGH); // turn the LED on
  delay(500); // wait 500 milliseconds
  digitalWrite(77,LOW); // turn the LED off
  delay(500); // wait 500 milliseconds
}
```

Creating infinite loops

```
void loop() //blink a LED repeatedly
{
  digitalWrite(77,HIGH); // turn the LED on
  delay(500); // wait 500 milliseconds
  digitalWrite(77,LOW); // turn the LED off
  delay(500); // wait 500 milliseconds
}
```