

COMMUNICATION BETWEEN PROCESSING & ARDUINO

Processing& Arduino通讯

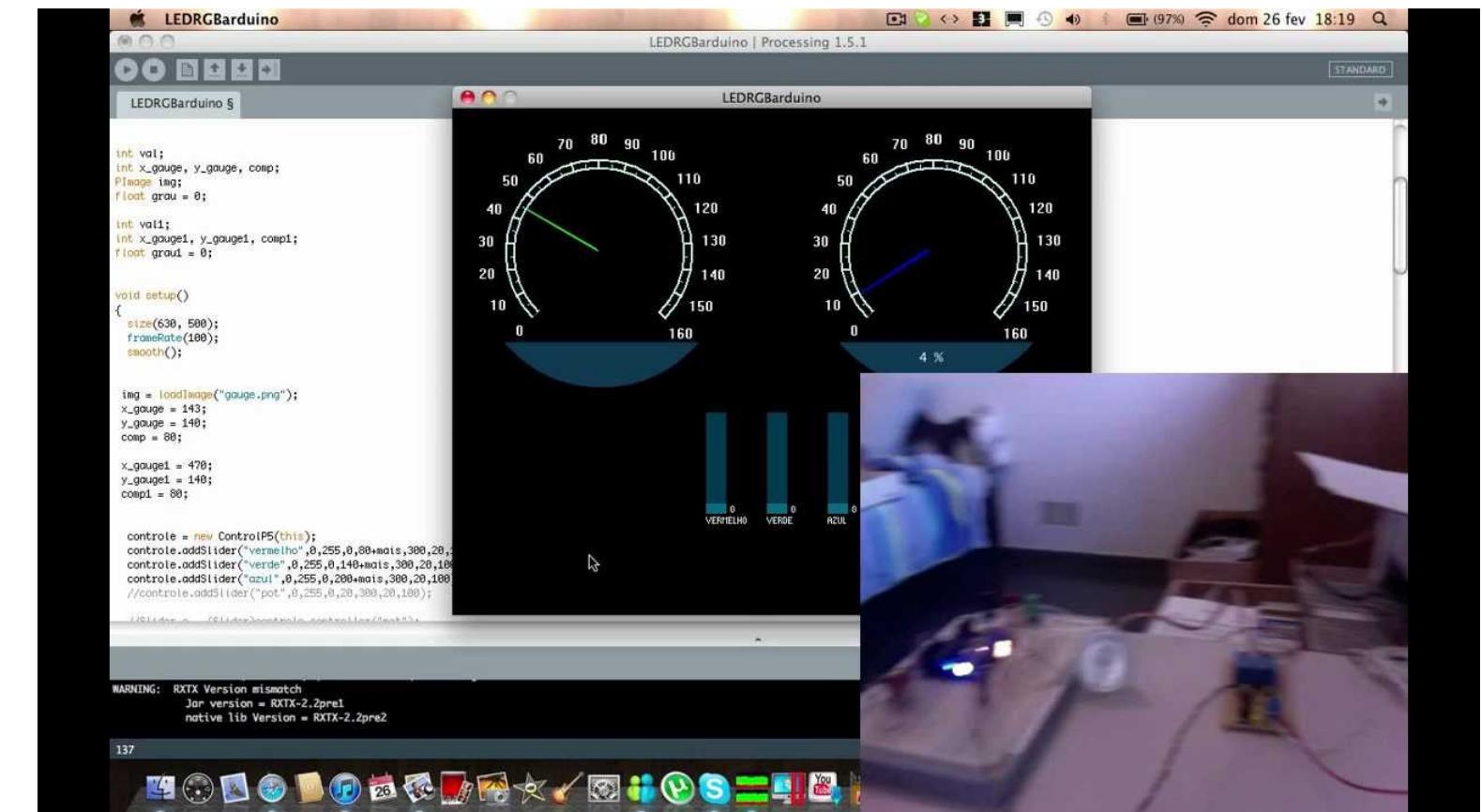
Day04_2

COMMUNICATION BETWEEN PROCESSING & ARDUINO

COMMUNICATION BETWEEN PROCESSING & ARDUINO

Communication between Processing and Arduino can happen in the following ways:

- Firmata
 - Mouse & Keyboard Emulation
键盘鼠标事件模拟
 - Serial Communication
串口通信



COMMUNICATION BETWEEN PROCESSING & ARDUINO

- Firmata

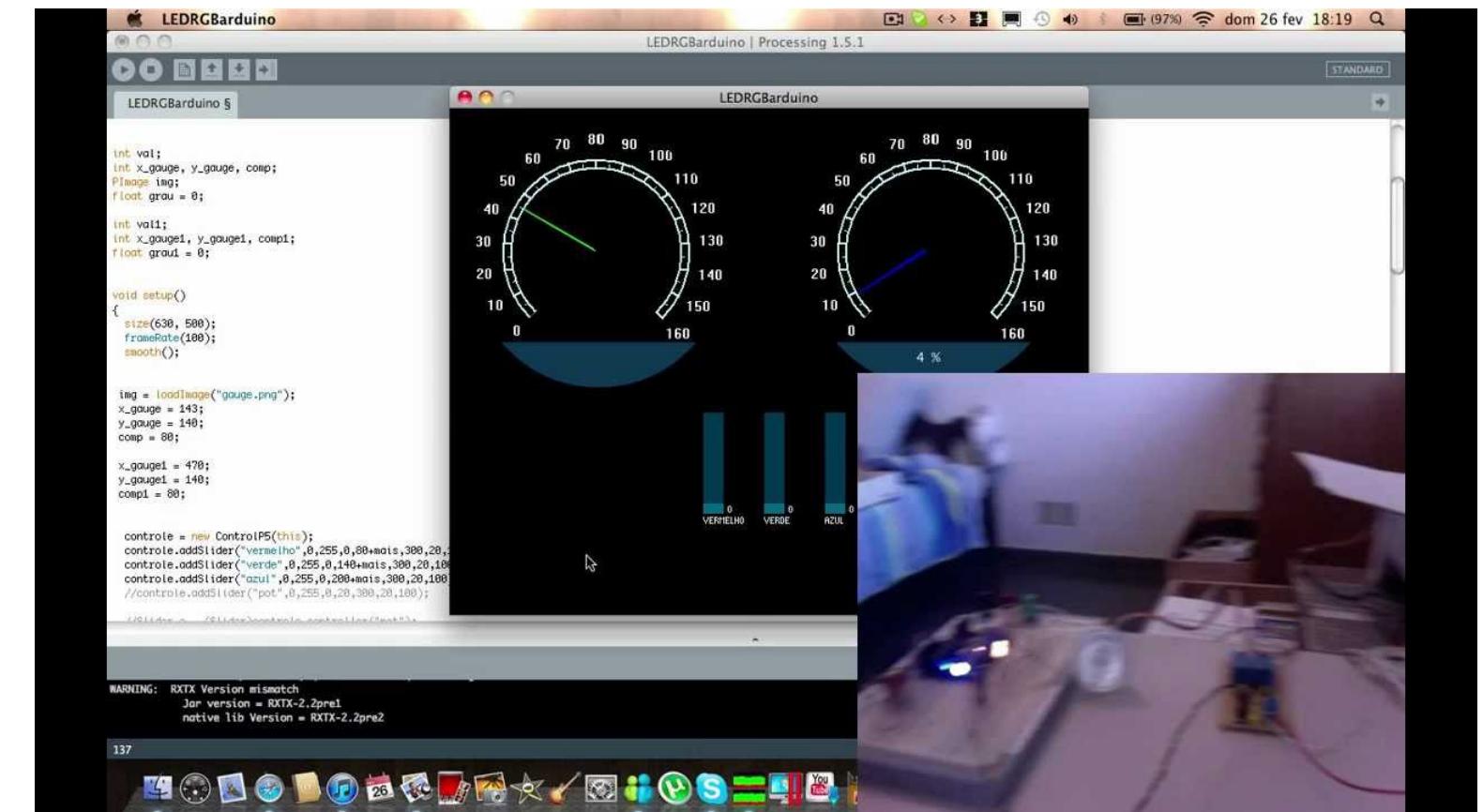
- Great for simple inputs (buttons) and outputs (LEDs)
方便实现简单输入（按钮）输出（LED）
- Can't do libraries and complex sensors / complex outputs!
无法使用libraries和复杂的输出

- Leonardo / Due (Red Arduinos) 特殊型号Arduino板

- Great for input to replace mouse/ keyboard
方便替换鼠标和键盘

- Serial Communication 串口通信

- Great for Sensors and complex outputs (motors)
方便使用传感器和复杂输出（如电机）



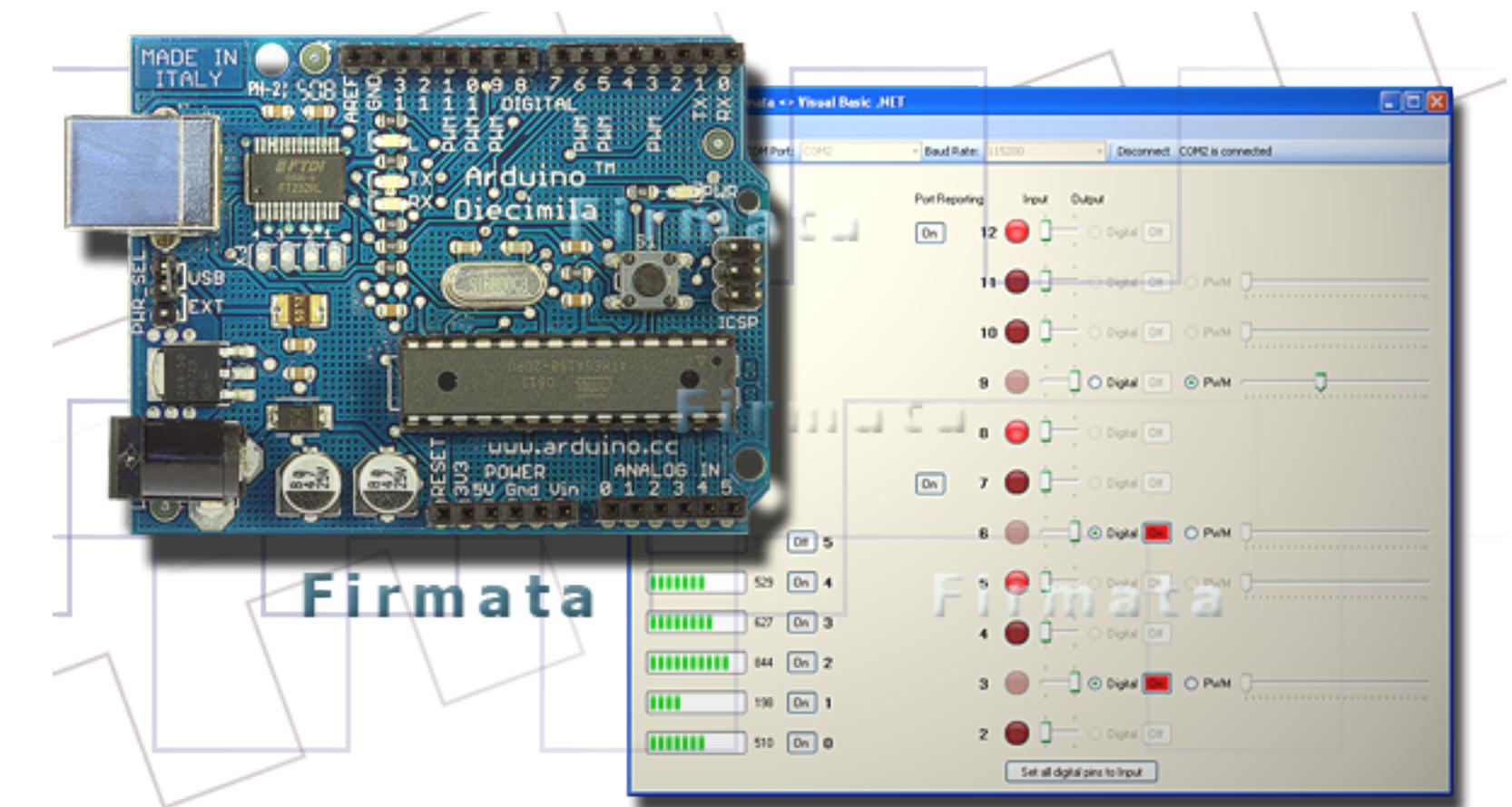
FIRMATA

FIRMATA

Firmata is a protocol for communicating between microcontrollers and computers.

To use Firmata, first install the libraries in both Processing and Arduino.

Firmata是一种微控制器和电脑之间通信的协议，使用Firmata需要先分别给Processing和Arduino加载library文件



ARDUINO FIRMATA LIBRARY

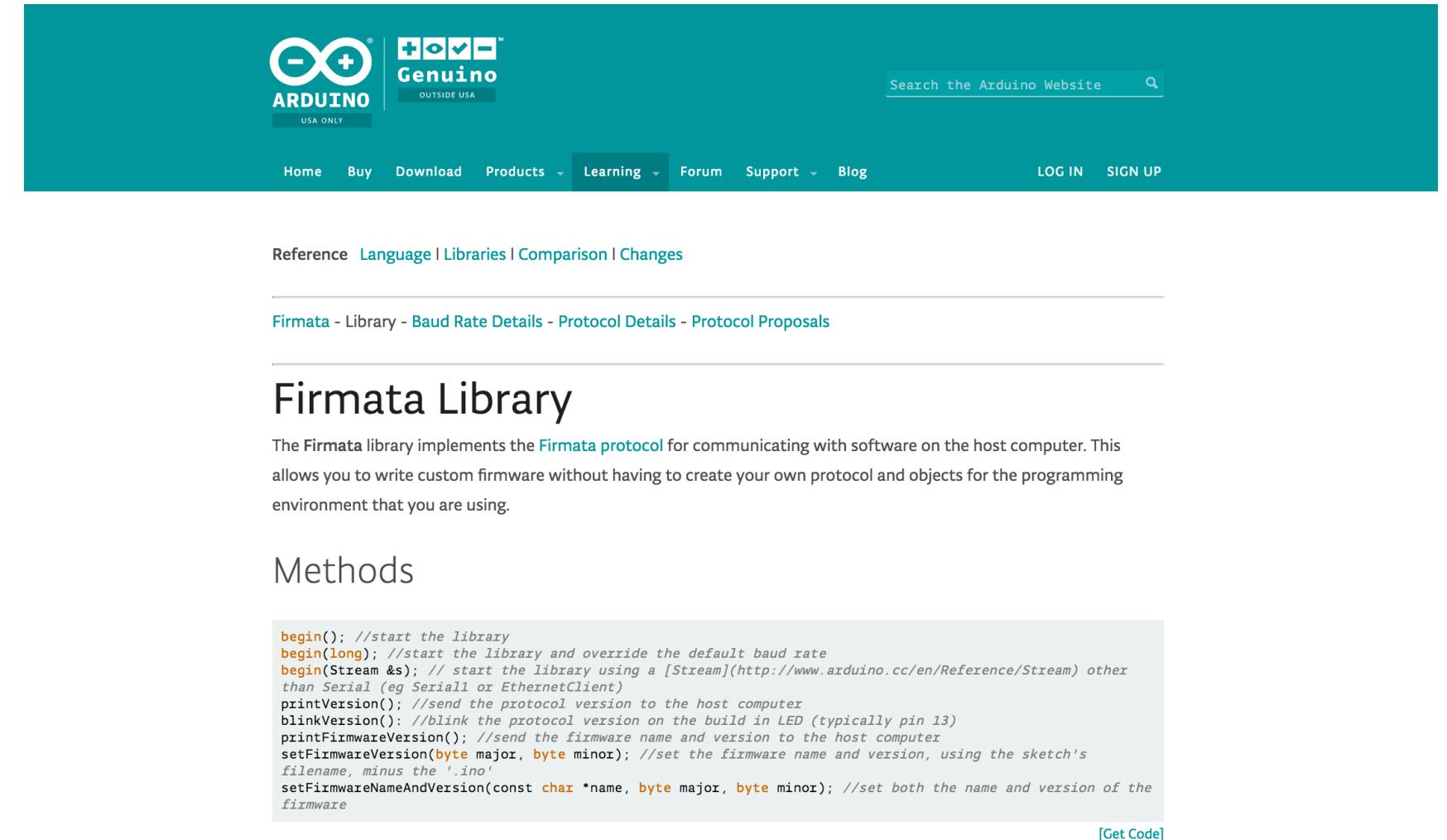
Step 1:

Download / Update Firmata on the Arduino IDE.

You can get to the library window by going to

Sketch > Include Library > Manage Libraries

And search for Firmata.



The screenshot shows the Arduino website's header with the Arduino and Genuino logos, a search bar, and navigation links for Home, Buy, Download, Products, Learning, Forum, Support, and Blog. Below the header, a breadcrumb trail indicates the current page: Firmata - Library - Baud Rate Details - Protocol Details - Protocol Proposals. The main content area is titled "Firmata Library" and describes the protocol for communicating with software on the host computer. It includes a section titled "Methods" with a code snippet for the Firmata library:

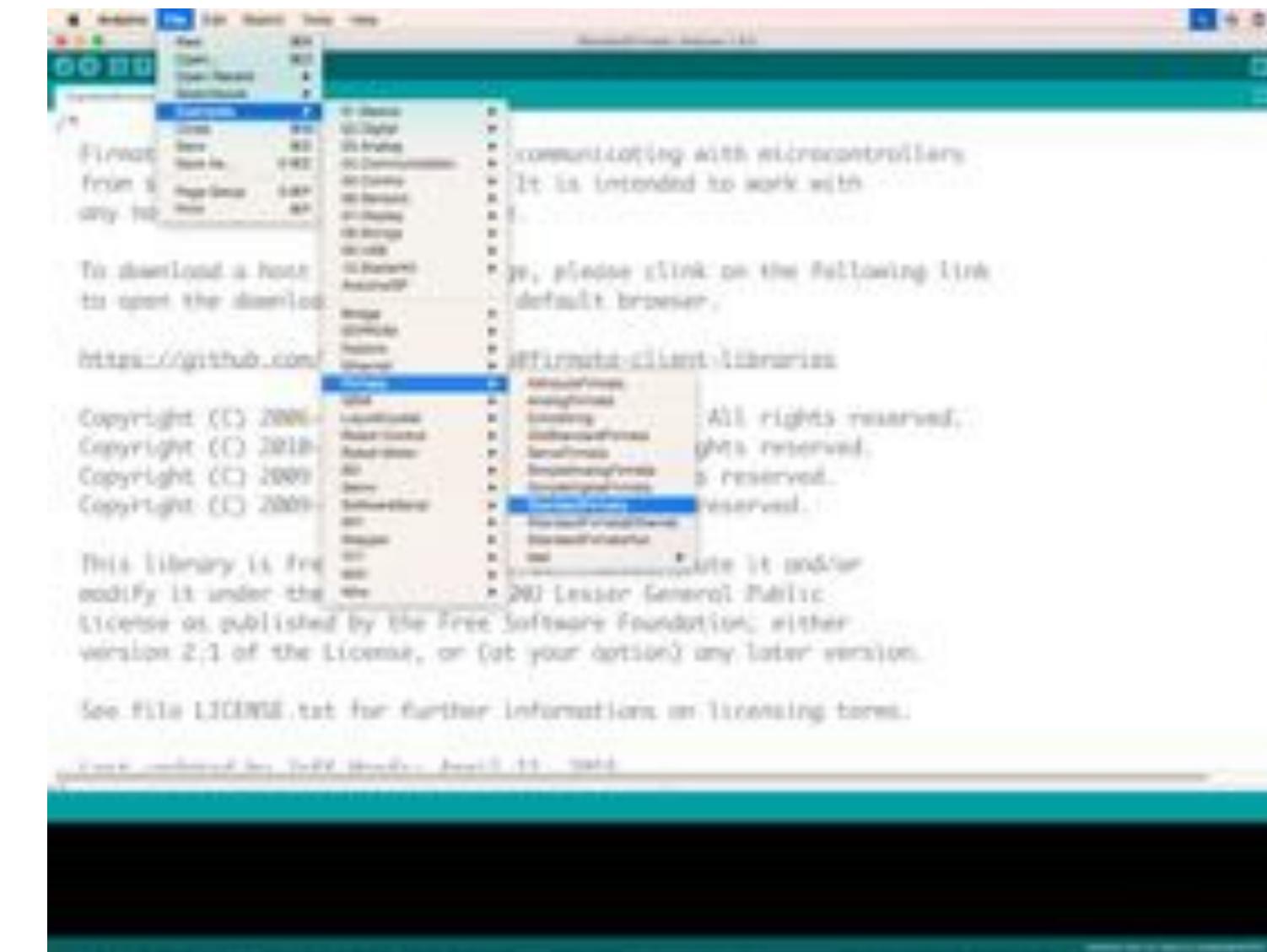
```
begin(); //start the library
begin(long); //start the library and override the default baud rate
begin(Stream &s); // start the library using a [Stream](http://www.arduino.cc/en/Reference/Stream) other
than Serial (eg Serial1 or EthernetClient)
printVersion(); //send the protocol version to the host computer
blinkVersion(); //blink the protocol version on the build in LED (typically pin 13)
printFirmwareVersion(); //send the firmware name and version to the host computer
setFirmwareVersion(byte major, byte minor); //set the firmware name and version, using the sketch's
filename, minus the '.ino'
setFirmwareNameAndVersion(const char *name, byte major, byte minor); //set both the name and version of the
firmware
```

[\[Get Code\]](#)

STANDARD FIRMATA EXAMPLE

Step 2:

Once the Firmata libraries are installed, upload the Standard Firmata sketch to your Arduino.



SCREENSHOT

PROCESSING FIRMATA LIBRARY

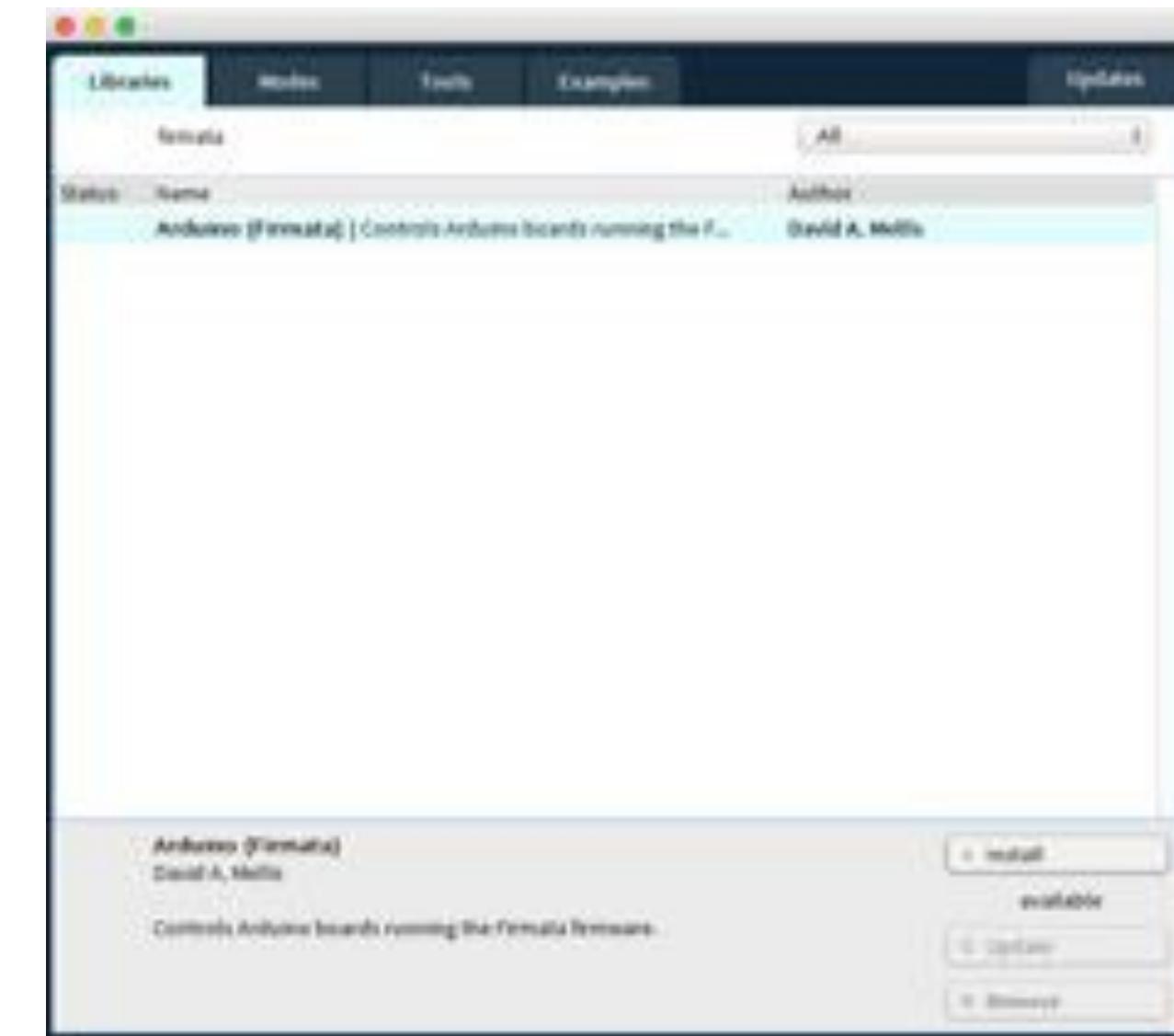
Step 3:

Download / Update Firmata on the Processing IDE.

You can get to the library window by going to

Sketch > Import Library > Add Library And search

for Firmata.



SCREENSHOT

PROCESSING FIRMATA EXAMPLE

Step 4:

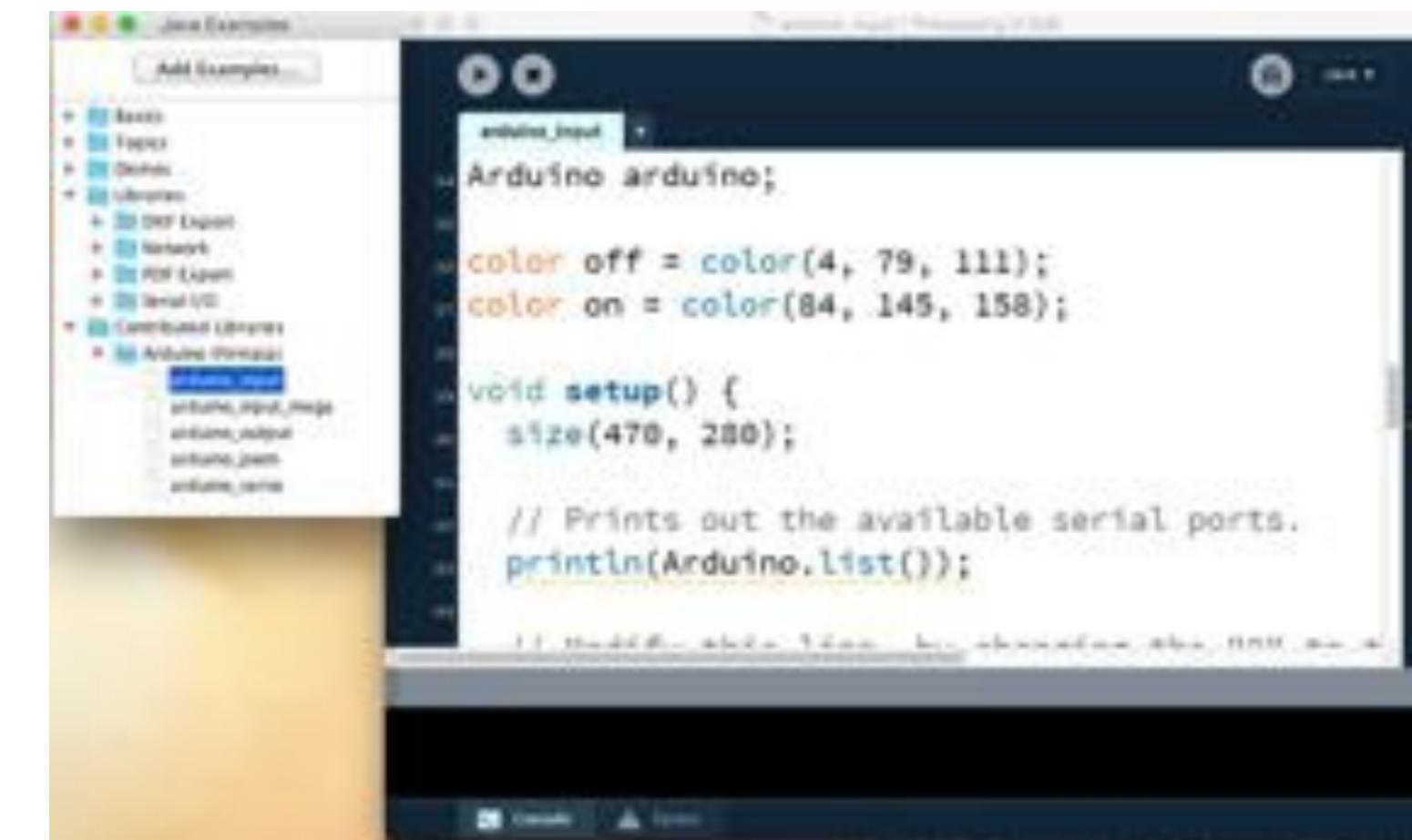
Run the Firmata “arduino_input” example on Processing.

You may need to modify the following line: arduino = new

Arduino(this, "...", 57600); with the name of the serial

port

such as "/dev/tty.usbmodem621"



The screenshot shows the Processing IDE interface. On the left, the 'Examples' browser lists various examples under categories like 'Basic', 'Fitter', 'Demos', 'Libraries', and 'Contributed Libraries'. The 'arduino_input' example is selected, highlighted in blue. The main workspace on the right contains the code for the 'arduino_input' sketch. The code includes declarations for colors (color off and color on), initializes an Arduino object, sets up the window size, and prints available serial ports.

```
arduinio arduino;

color off = color(4, 79, 111);
color on = color(84, 145, 158);

void setup() {
    size(470, 280);

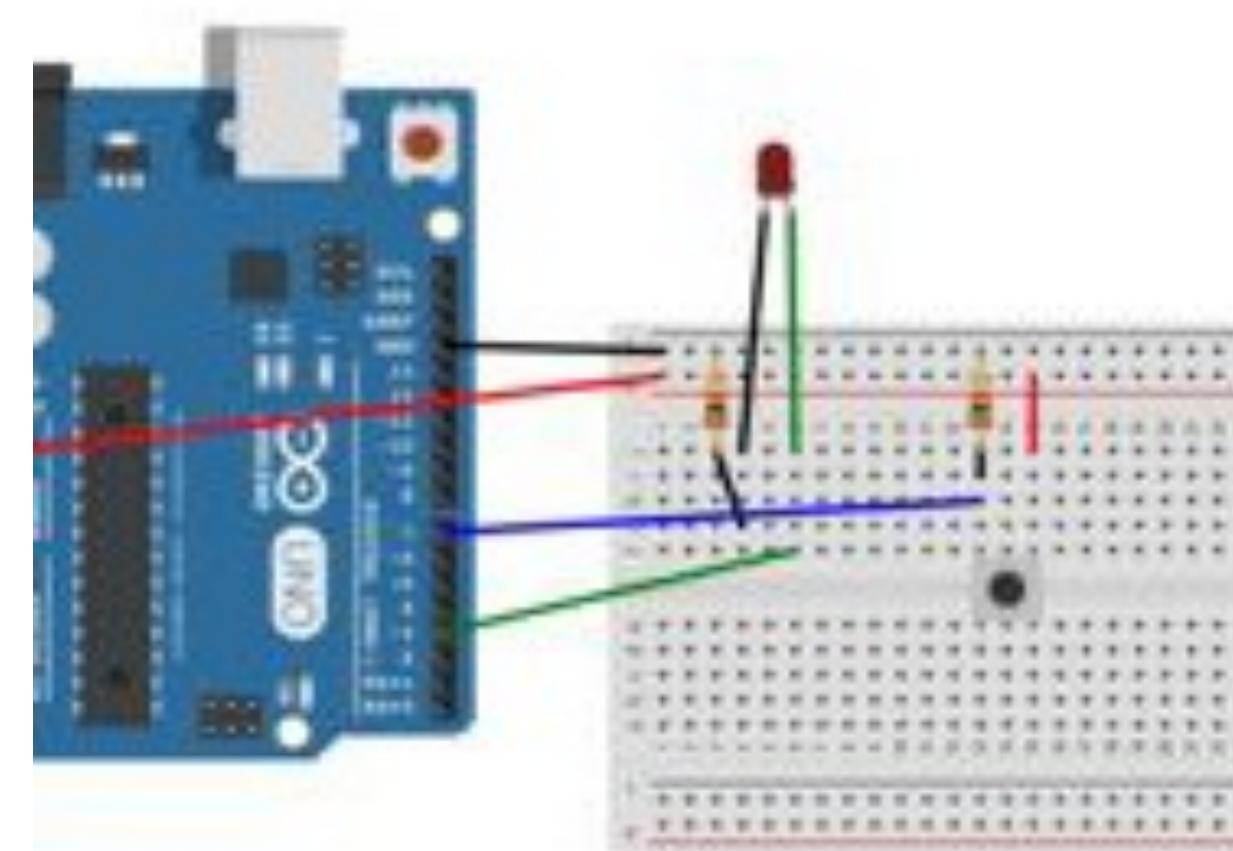
    // Prints out the available serial ports.
    println(Arduino.list());
```

SIMPLE CIRCUIT

Connect an LED and a button to test out the Firmata library.

You can also try a simple sensor like the Vibration sensor into the analog inputs.

But complex sensors and outputs like the Accelerometer and the Servo motor are not good to control with Firmata.



PROCESSING FIRMATA EXAMPLE

```
// Receives Digital Data from Arduino  
// Into Processing  
  
import processing.serial.*;  
import cc.arduino.*;  
Arduino arduino;  
  
void setup() {  
size(470, 280);  
arduino = new Arduino(this,Arduino.list()[3],57600);  
println(Arduino.list()[3]);  
arduino.pinMode(7, Arduino.INPUT);  
}  
  
void draw()  
{ background(0);  
if (arduino.digitalRead(7) == Arduino.HIGH)  
{ rect(10, 10, 200, 200);  
}  
}
```

PROCESSING FIRMATA EXAMPLE

```
// Sends Digital Data out of Processing
// Into Arduino

import processing.serial.*;
import cc.arduino.*;
Arduino arduino;

void setup(
{ size(470,280);
  arduino = new Arduino(this, Arduino.list()[3], 57600);
  println(Arduino.list()[3]);
  arduino.pinMode(3, Arduino.OUTPUT);
}

void draw() { background(0);
  if(mousePressed) {
    arduino.digitalWrite(3,Arduino.HIGH);
  } else { arduino.digitalWrite(3,Arduino.LOW);
  }
}
```

MOUSE & KEYBOARD EMULATION | 键盘鼠标事件模拟

MOUSE & KEYBOARD EMULATION | 键盘鼠标事件模拟

Certain Arduino boards are capable of emulating a USB Human Interface Device (HID), for example a mouse or keyboard.

This allows an Arduino to control any piece of software running on a computer, including your Processing sketches.

某些Arduino具有模拟USB交互设备（HID）的能力，比如键盘和鼠标，通过模拟键鼠交互，你可以使用arduino控制任何软件包括processing程序



PHOTO BY POCKAFWYE

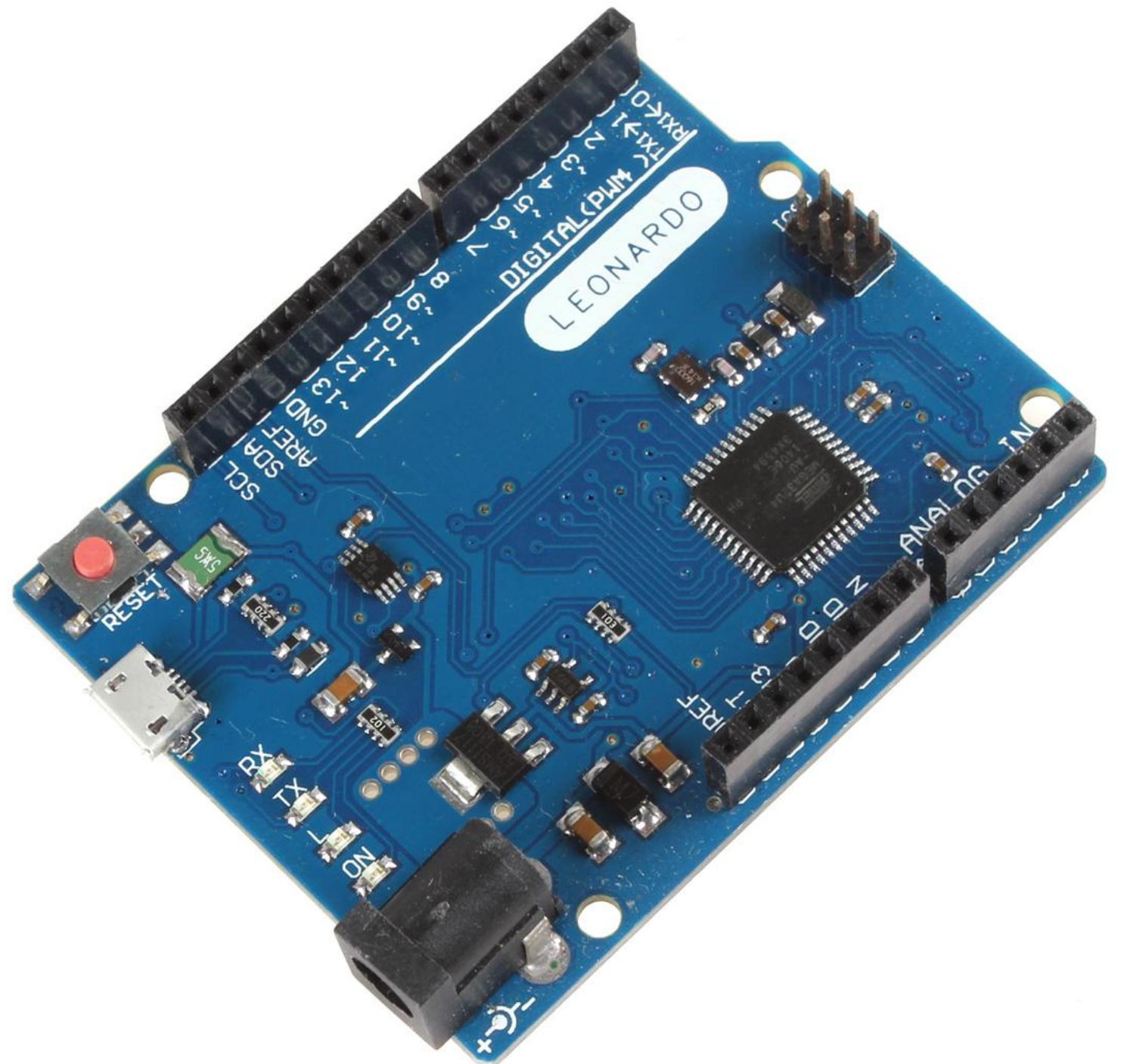
LEONARDO, MICRO & DUE

The 3 Arduinos that are designed with USB HID capability are the Leonardo, Micro, and Due.

可以实现模拟键盘鼠标的arduino板型号为：

Leonardo, Micro, and Due

我们有的设备中有5块Arduino leonardo

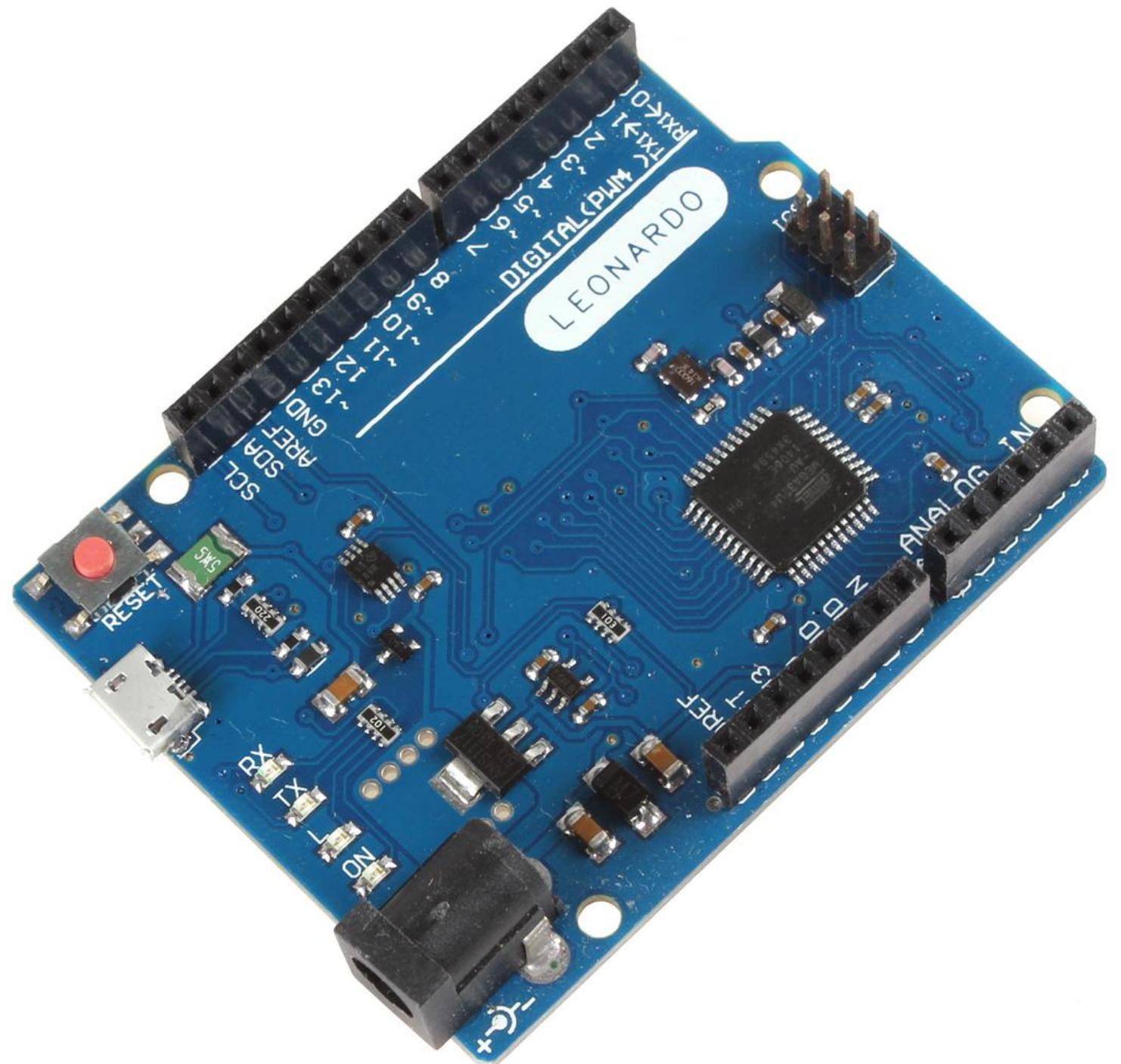


ARDUINO LEONARDO

These Arduinos need a USB Micro connection.

In-class: experiment with mouse and keyboard emulation.

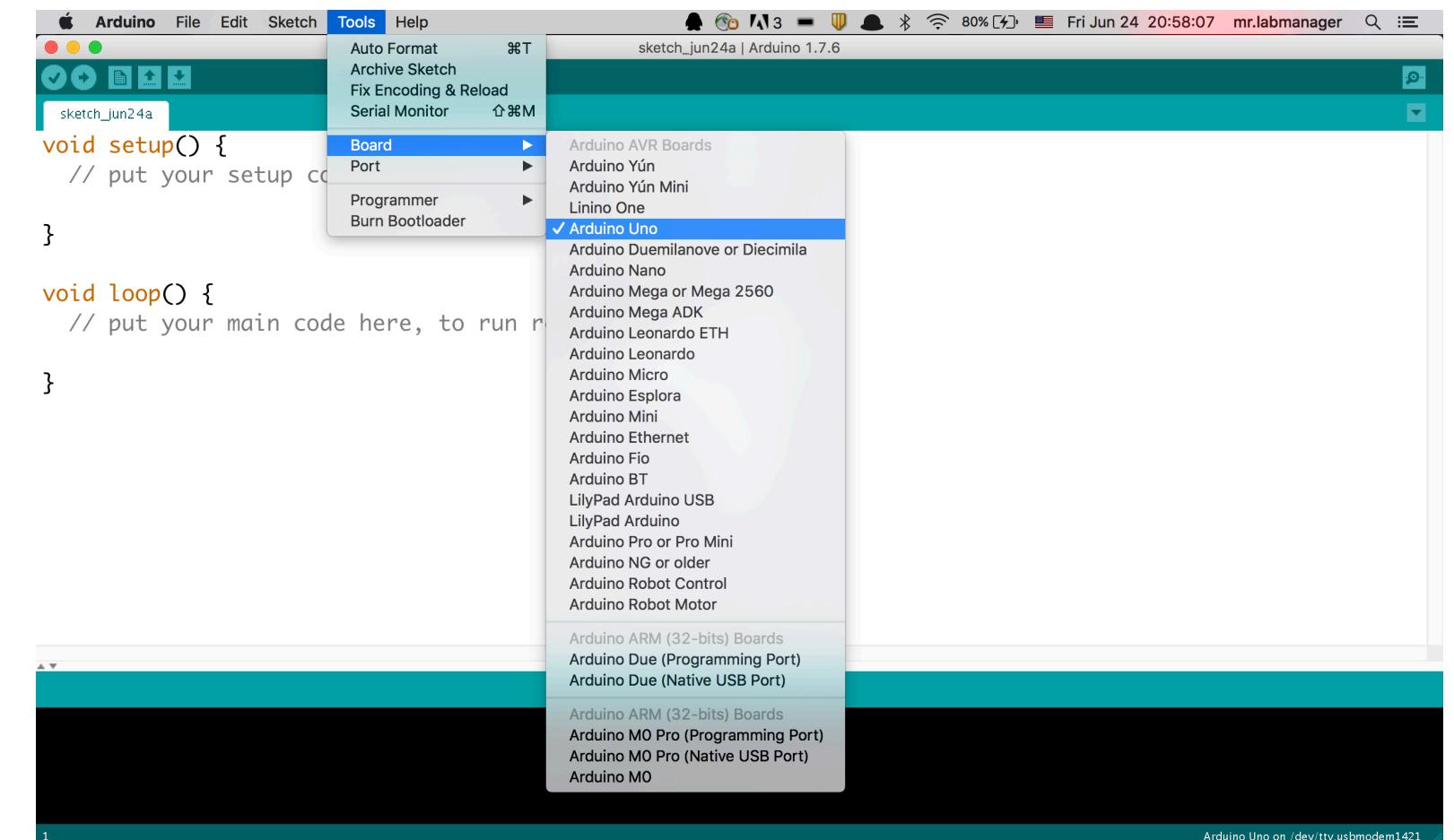
LEONARDO 使用micro USB线连接电脑，也是采购的设备



ARDUINO LEONARDO

Don't forget to select the correct board under Tools > Board > Arduino Leonardo

使用LEONARDO时注意更改Tools> Board下的型号为
Arduino Leonardo



ARDUINO MOUSE LIBRARY

Mouse.h

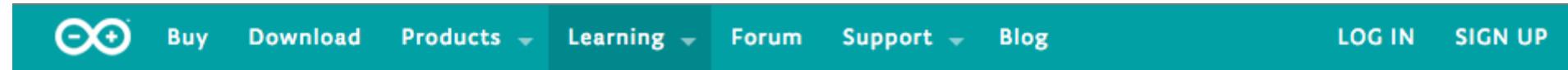
The Arduino Mouse Library allows an Arduino to act like a mouse by initiating clicks and even moving the cursor around.

Library, 用以实现Arduino模拟鼠标功能，可以触发鼠标点击甚至移动光标



PHOTO BY YUM9ME

ARDUINO MOUSE LIBRARY



Mouse

The mouse functions enable a Leonardo, Micro, or Due to control cursor movement on a connected computer. When updating the cursor position, it is always relative to the cursor's previous location.

- [Mouse.begin\(\)](#)
- [Mouse.click\(\)](#)
- [Mouse.end\(\)](#)
- [Mouse.move\(\)](#)
- [Mouse.press\(\)](#)
- [Mouse.release\(\)](#)
- [Mouse.isPressed\(\)](#)

```
Mouse.begin();  
Mouse.click();  
Mouse.click(button);  
//MOUSE_LEFT (default)  
//MOUSE_RIGHT  
//MOUSE_MIDDLE  
Mouse.move(x,y,wheel);  
//x, int, amount to move along the x axis  
//y, int, amount to move along the y axis  
//wheel, int, amount to move scroll wheel
```

MOUSE LIBRARY EXAMPLE

```
#include "Mouse.h"

void setup() {
pinMode(7, INPUT);
Mouse.begin();
}

Void loop() {
if(digitalRead(7) == HIGH) {
    Mouse.click();
}
}
```

ARDUINO KEYBOARD LIBRARY

Keyboard.h:

The Arduino Keyboard Library allows an Arduino to act like a keyboard by initiating key presses.

Library, 用以实现Arduino模拟键盘功能，可以触发键盘按键点击



ARDUINO KEYBOARD LIBRARY



Keyboard

The keyboard functions enable a Leonardo, Micro, or Due to send keystrokes to an attached computer.

Note: Not every possible ASCII character, particularly the non-printing ones, can be sent with the Keyboard library.

The library supports the use of modifier keys. Modifier keys change the behavior of another key when pressed simultaneously. [See here](#) for additional information on supported keys and their use.

- [Keyboard.begin\(\)](#)
- [Keyboard.end\(\)](#)
- [Keyboard.press\(\)](#)
- [Keyboard.print\(\)](#)
- [Keyboard.println\(\)](#)
- [Keyboard.release\(\)](#)
- [Keyboard.releaseAll\(\)](#)
- [Keyboard.write\(\)](#)

```
Keyboard.begin();
```

```
Keyboard.press('a'); //char
```

```
Keyboard.print("hello"); //string
```

KEYBOARD LIBRARY EXAMPLE

```
#include "Keyboard.h"
void setup() {
pinMode(7, INPUT);
Keyboard.begin();
}
void loop() {
if(digitalRead(7) == HIGH) {
    Keyboard.press('n');
    delay(100);
    Keyboard.release('n');
}
}
```

SERIAL COMMUNICATION|串口通信

Serial is a form of digital communication that involves sending a sequence of bits, one at a time, between electronic devices.

串口指设备之间同时发送比特序列的通信方式



PHOTO BY MARTIN CATHRAE

BITS & BYTES | 比特&字节

A bit is the smallest unit of digital information, representing a single binary value, either 0 or 1.

A Byte is a slightly larger unit of digital information, representing eight bits, between 0 and 255.

Bytes are also a Variable data type in processing and Arduino.

比特：二进制一位数值如0, 1

字节：和比特大一级的单位，8比特，可以表达00000000~11111111，转化为十进制为0~255



PHOTO BY MARKROUND

UART

In serial communication, a Universal Asynchronous Receiver Transmitter (UART) is a device that converts a byte of data into individual bits and sends it out by modulating the voltage either high or low to represent a 1 or 0.

Typically, another UART on the receiving end converts the bits back into bytes.



PHOTO BY MARKDEMERS

USB|通用串行总线

Universal Serial Bus (USB) is a modern communications protocol that can support multiple devices attached to a single port at very high data rates. USB can also support older serial devices through a converter.

Different Arduino boards have varying converters like this.

连接计算机系统与外部设备的一种串口总线标准，也是一种输入输出接口的技术规范，被广泛地应用于个人电脑和移动设备等信息通讯产品，并扩展至摄影器材、数字电视（机顶盒）、游戏机等其它相关领域。

USB也可以通过转接头支持其他类型的串口设备，不同的arduino使用不同的转接头



PHOTO BY MARCELO ALVES

DATA RATE

A data rate is the speed at which communication occurs.

In order for serial communication to be successful the transmitting and receiving sides must both be set to communicate at the same speed.

With a speed of 9600 bits per second (baud) the voltage is interpreted every 1/9600th of a second.

指数据传输速率，串口通信需要发送和接收端速率保持同步

9600比特 / 秒

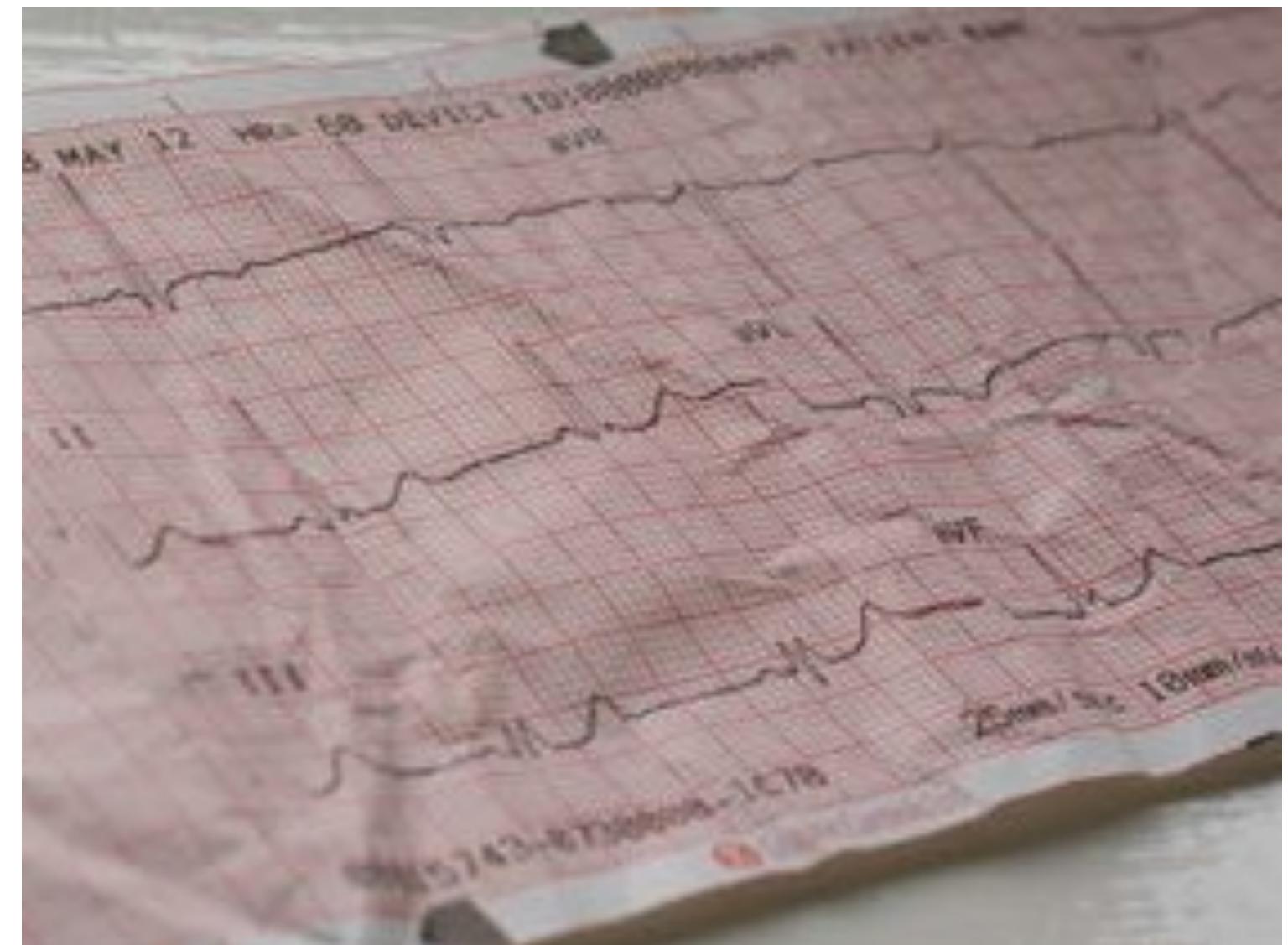


PHOTO BY CROSS
DUCK

PORT | 端口

A port is the physical interface through which serial communication occurs.

When you plug in your Arduino it appears as an available serial port on your computer.

A port can only be connected to one device at a time.

端口是串口通信发生的物理界面，连接arduino和电脑后可以找到可用端口，一个端口不能同时被两个或以上设备使用



PHOTO BY BERT VANDIJK

BUFFER | 缓冲

A buffer is a place in the memory of an electronic device where bytes are temporarily stored as they arrive from another device.

缓冲器为暂时置放输出或输入数据的内存。缓冲器内数据自存储设备（如硬盘），放置在缓冲器中，须待机送至CPU或其他运算设备



PHOTO BY KIEL ESON

PROTOCOLS | 通讯协议

A protocol defines the rules that govern the interpretation of whatever communication is taking place. Although we have little control over serial communication itself, what we can control is what is being sent and what it means.

Protocol specification should be determined based on the complexity of the data being transmitted.

网络传输协议或简称为传送协议，是指计算机通信或网络设备的共同语言。传送协议也存在于计算机的其他形式通信，例如：面向对象编程里面对象之间的通信；操作系统内不同程序之间的消息，都需要有一个传送协议，以确保传信双方能够沟通无间。



PHOTO BY GORDON TARPLEY

BINARY PROTOCOLS

A binary Protocol is a protocol in which every value being transmitted fits within a single byte and does not need to be interpreted.

指所有的数值使用单字节传输，并且不用被interpret

```
101010101101010010101011010101101010101011010  
1010101010110101010101101110101010101101010101  
1010101101010101010110101010101101010101011010  
110101010101000010101011010100101010101011010101  
101001010000101010101110010110010100101010110100  
00011010100101001101000101010101101010101010101  
0011010101101010100100101010101010101010110101  
1010101011010100101011010101101010101011010  
101010101101010101011011010101010110101010101  
1010101101010101010110101010101011010101010110  
1101010101000010101011010100101010101011010101  
1010010100001010101011100101100101001010110100  
000110101001010011010001010101011010101010101  
0011010101101010100100101010101010101010110101  
10101010110101010110111010101010110101010101  
1010101101010101010110101010101011010101010110  
1101010101000010101011010100101010101010110101  
1010010100001010101011100101100101001010110100  
000110101001010011010001010101011010101010101  
0011010101101010100100101010101010101010110101
```

BINARY PROTOCOLS

00000000 / 11111111

ARDUINO SERIAL PORTS | 串口通信端口

The Arduino Uno has a single hardware serial port which uses digital pins 0 (receive) and 1 (transmit).

Other Arduinos may have multiple serial ports.

You can also configure software based serial ports if you need to communicate with multiple devices simultaneously.

UNO只有一个端口用于串口通信，使用0, 1号引脚通信，其他可能型号的arduino有更多的端口,可以同时和多个设备通信

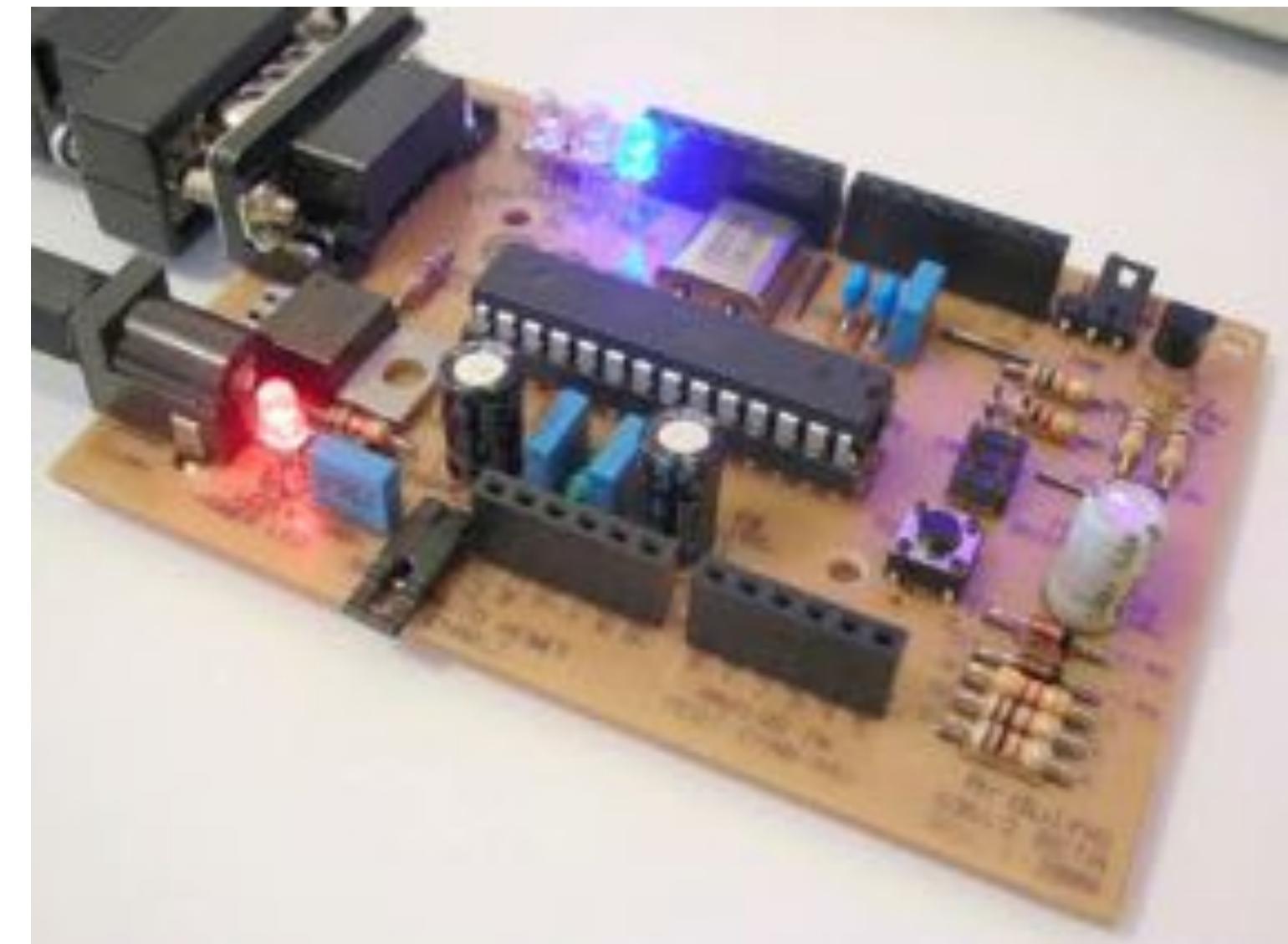


PHOTO BY ADILSON AKASHI

SERIAL FUNCTIONS | 串口function

Arduino has many built-in Serial Functions that facilitate data transmission and receipt.

Before using serial, you must first call `Serial.begin()` and define a data rate.

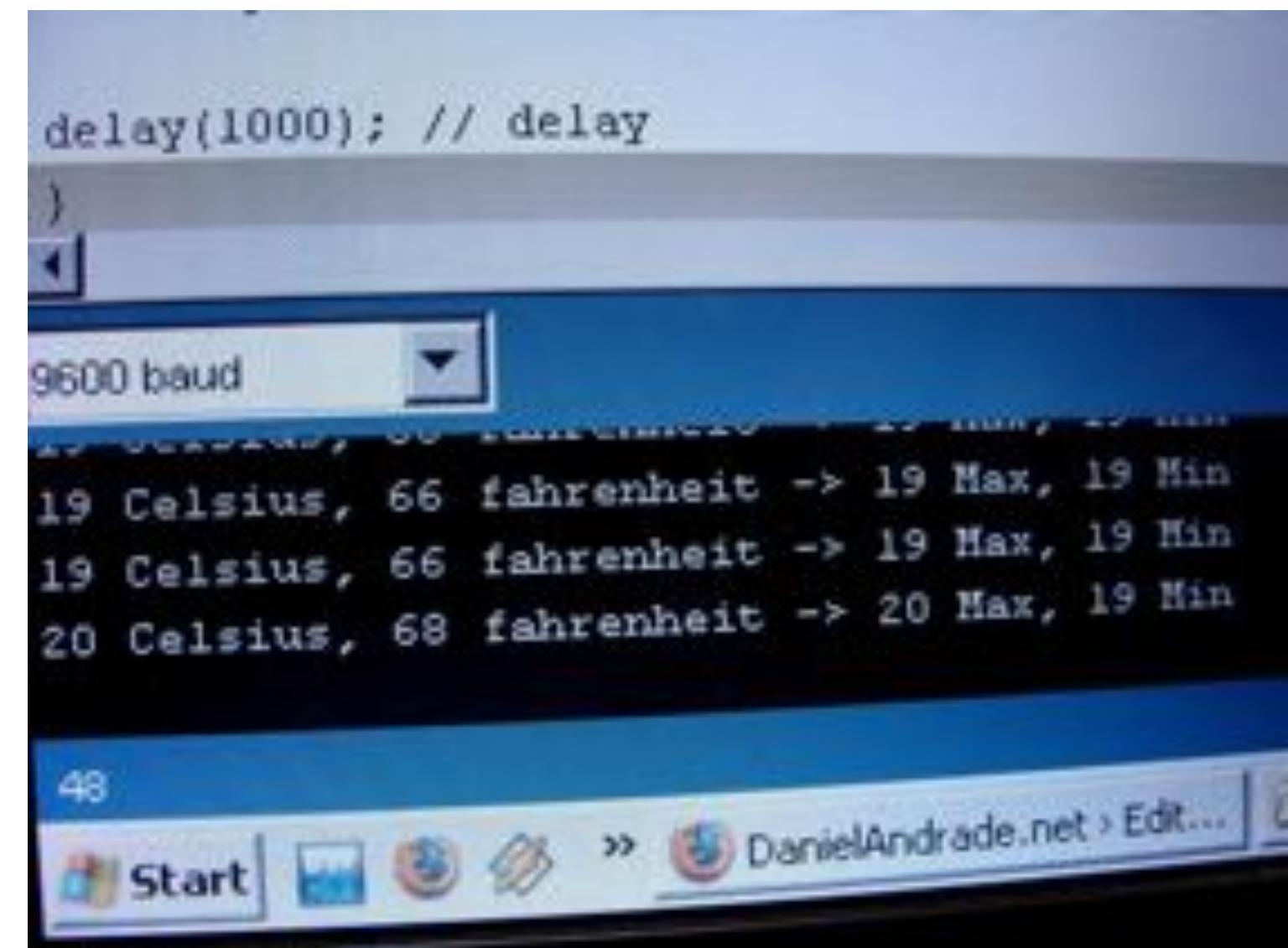


PHOTO BY DANIEL SPILLERE ANDRADE

ARDUINO SERIAL EXAMPLE

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

void loop() {
    int sensorValue = analogRead(A0);
    sensorValue = sensorValue/4;
    Serial.write(sensorValue);
}
```

PROCESSING SERIAL LIBRARY

Processing has a Serial Library, also with many built-in functions.

Before using the Serial function in Processing you must first import the Serial library, create a serial variable, and define the port and data rate.



Cover

Download

Exhibition

Reference

Libraries

Tools

Environment

Tutorials

Examples

Books

Handbook

Overview

People

Shop

» Forum

» GitHub

» Issues

» Wiki

» FAQ

» Twitter

» Facebook

Serial

The Serial library reads and writes data to and from external devices one byte at a time. It allows two computers to send and receive data. This library has the flexibility to communicate with custom microcontroller devices and to use them as the input or output to Processing programs. The serial port is a nine pin I/O port that exists on many PCs and can be emulated through USB.

Issues related to the Serial library on different platforms are documented on the [Processing Wiki](#). The source code is available on the [processing GitHub repository](#).

When sending data to the console, such as via `print()` or `println()`, note that the console is relatively slow. It does not support high-speed, real-time output (such as at 60 frames per second). For real-time monitoring of serial values, render those values to the Processing window during `draw()`.

Serial

Serial

available()

read()

readChar()

readBytes()

readBytesUntil()

readString()

readStringUntil()

buffer()

bufferUntil()

last()

lastChar()

write()

clear()

stop()

list()

Serial Event

serialEvent()

PROCESSING SERIAL EXAMPLE

```
import processing.serial.*;
Serial myPort;
void setup() {
    myPort = new Serial(this, Serial.list()[0], 9600);
}
void draw() {
    while (myPort.available() > 0)
        { int inByte = myPort.read();
        println(inByte);
        }
}
```