

Arduino



Designed
Intelligence
Group

TU / **e**

Technische Universiteit
Eindhoven
University of Technology

- **Arduino Hardware**
- **Blink an LED**
- **Digital Input**
- **Analog Input**
- **Analog Output**
- **Serial Communication**
 - **Using Serial Libraries**
 - **Using Firmata**

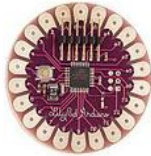
Why Arduino?

- **Physical Computing**
 - uses electronics
 - to prototype new materials
 - for designers and artists.
- **Tinkering**
- **Patching**
- **Community**
 - Blog, Forum, Playground (wiki)

Hardware



Arduino Uno



Arduino LilyPad



Arduino Ethernet



Arduino Nano



Arduino BT



Arduino Mini



Arduino Mega 2560



Arduino Fio



Arduino BT



Arduino Mini



USB/Serial Light Adapter



Arduino Pro Mini



Arduino Mega ADK



Arduino Pro



USB/Serial Light Adapter



Arduino Pro Mini



Designed
Intelligence
Group

TU/e

Technische Universiteit
Eindhoven
University of Technology

UNO



Designed
Intelligence
Group

TU/e

Technische Universiteit
Eindhoven
University of Technology

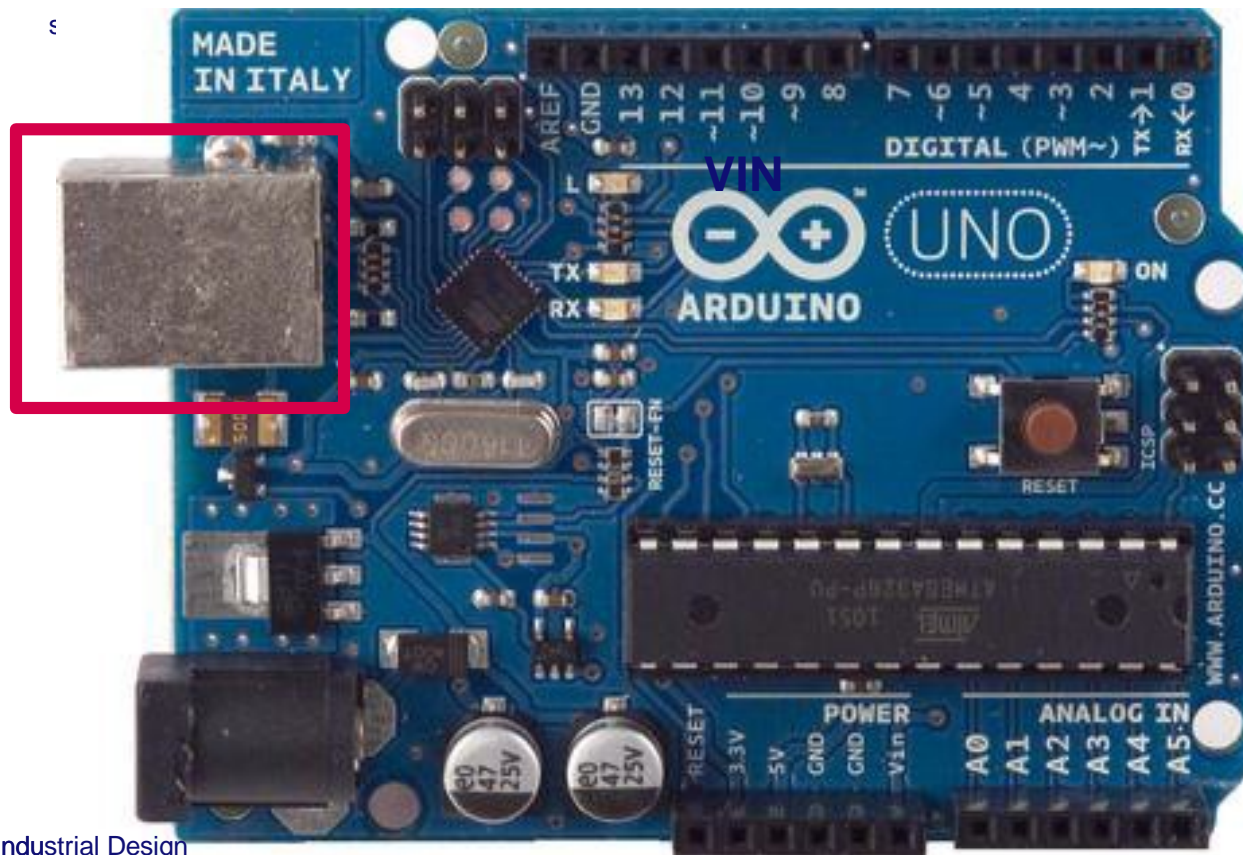
UNO

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (VIN) (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM (Static RAM)	2 KB (ATmega328)
EEPROM (Electrically erasable programmable ROM)	1 KB (ATmega328)
Clock Speed	16 MHz



UNO

- Power: USB Power supply (5V)



UNO

- Power: external power supply (7V-12V)



UNO

- **Power: VIN, input or supply, depending on external power source. (7-12V)**



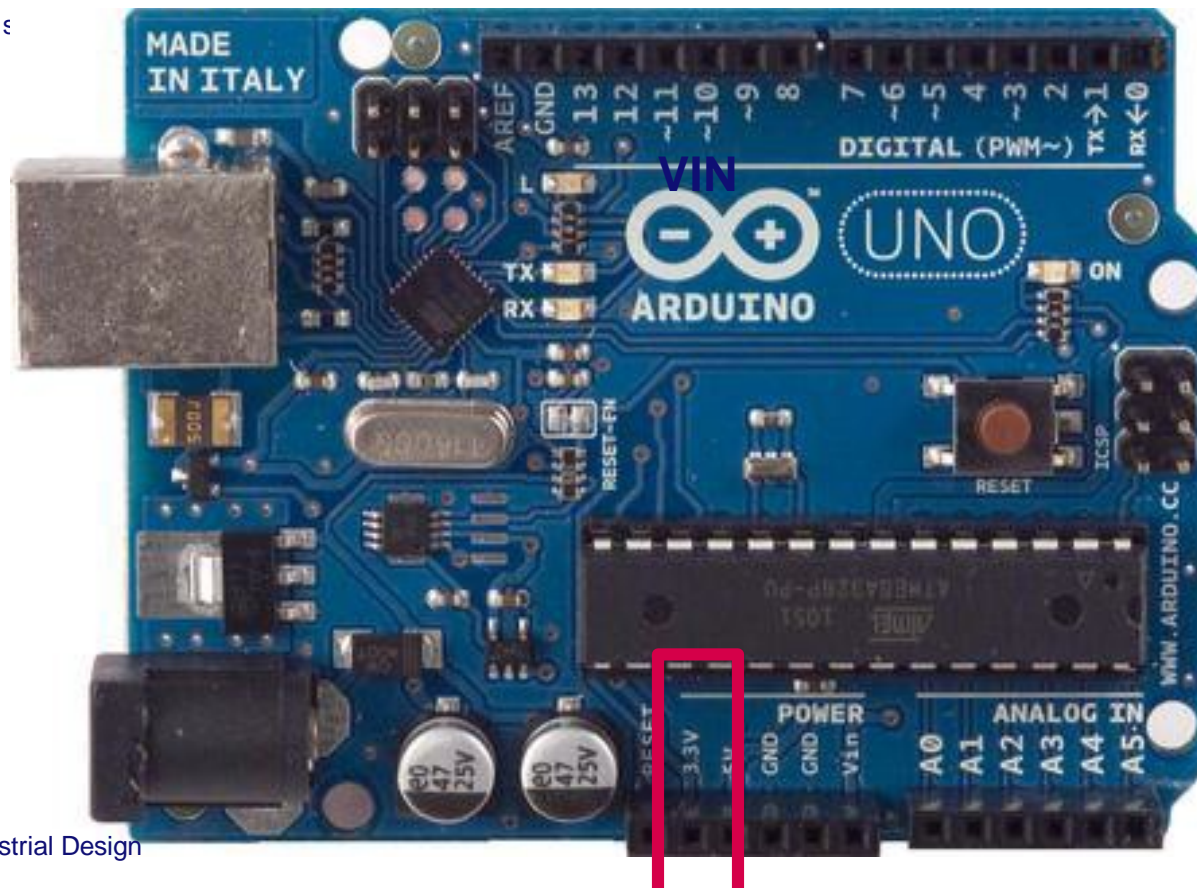
UNO

- Power: 5V supply



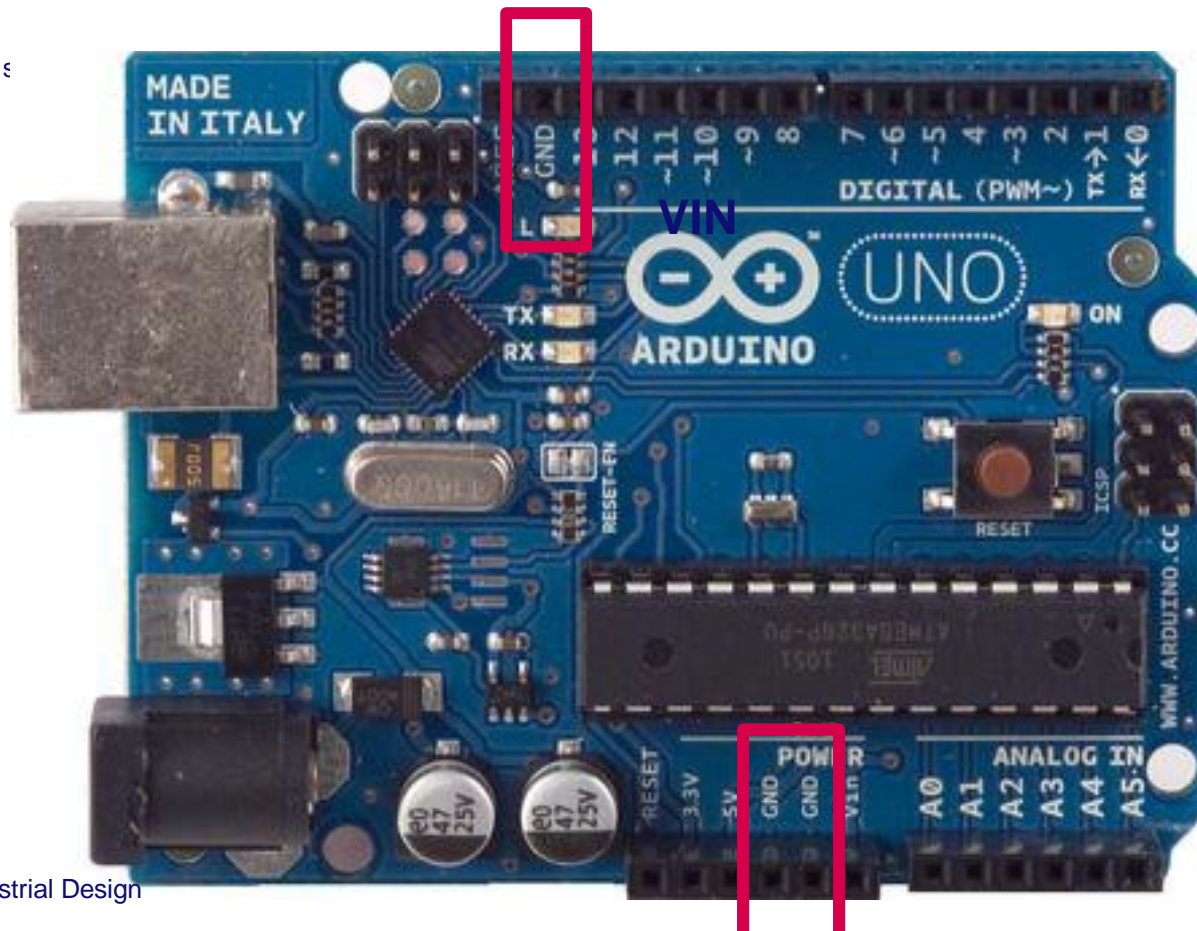
UNO

- Power: 3.3V supply



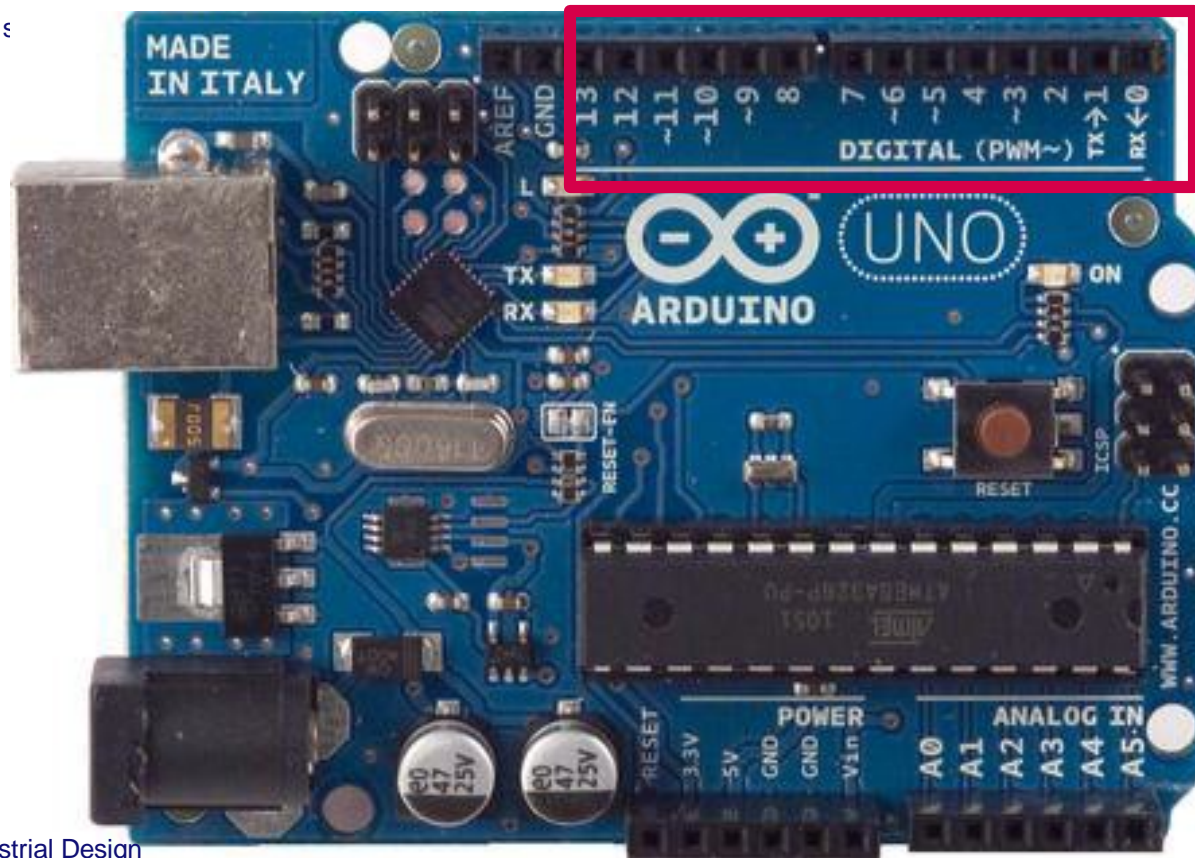
UNO

- Power: GND pins



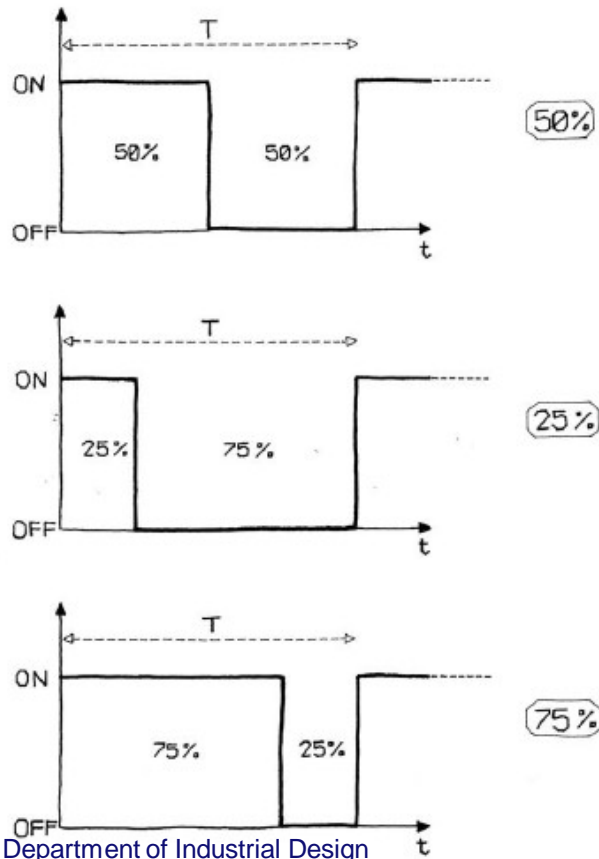
UNO

- Digital I/O Pins 14 (of which 6 provide PWM output)



UNO

- Digital I/O Pins 14 (of which 6 provide PWM output)
 - PWM (Pulse-width modulation)



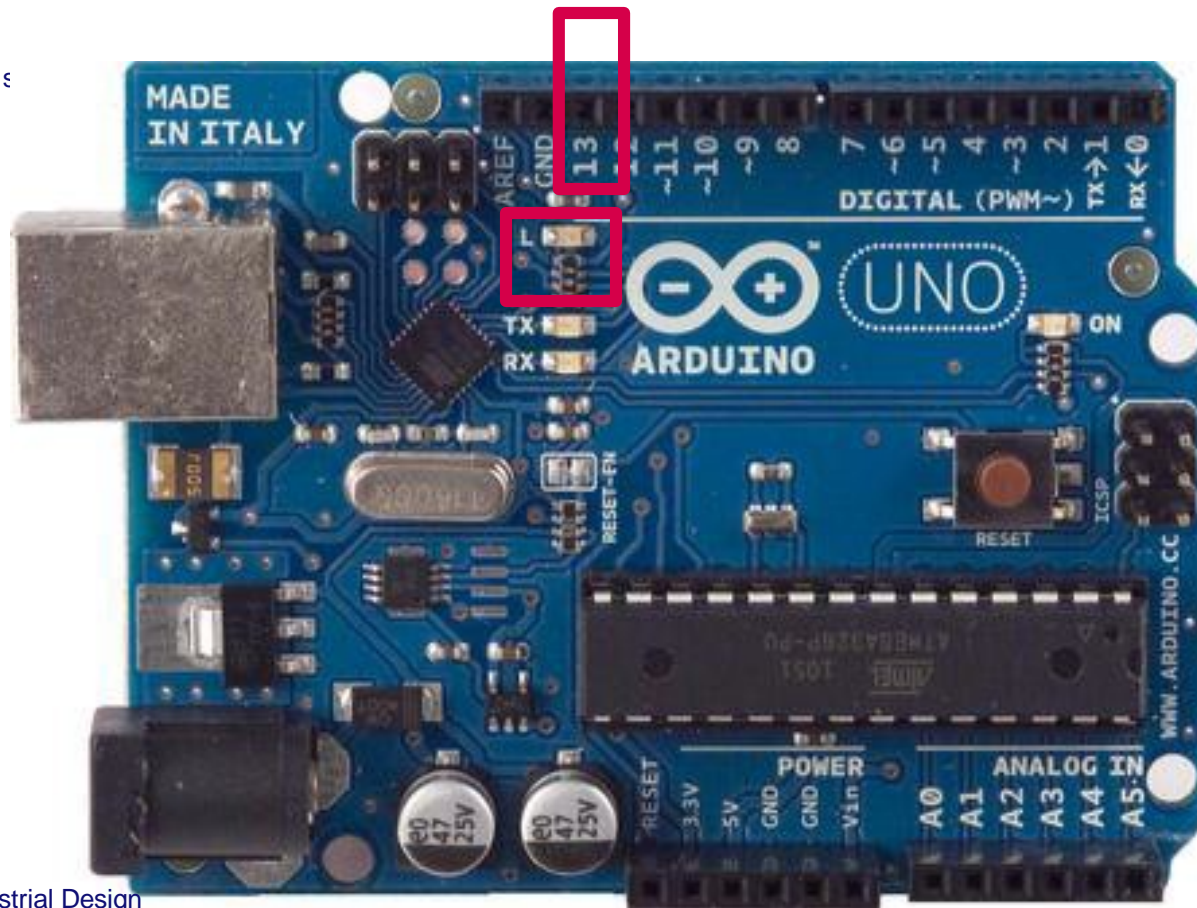
UNO

- Serial: 0 (RX) and 1 (TX)



UNO

- LED: 13



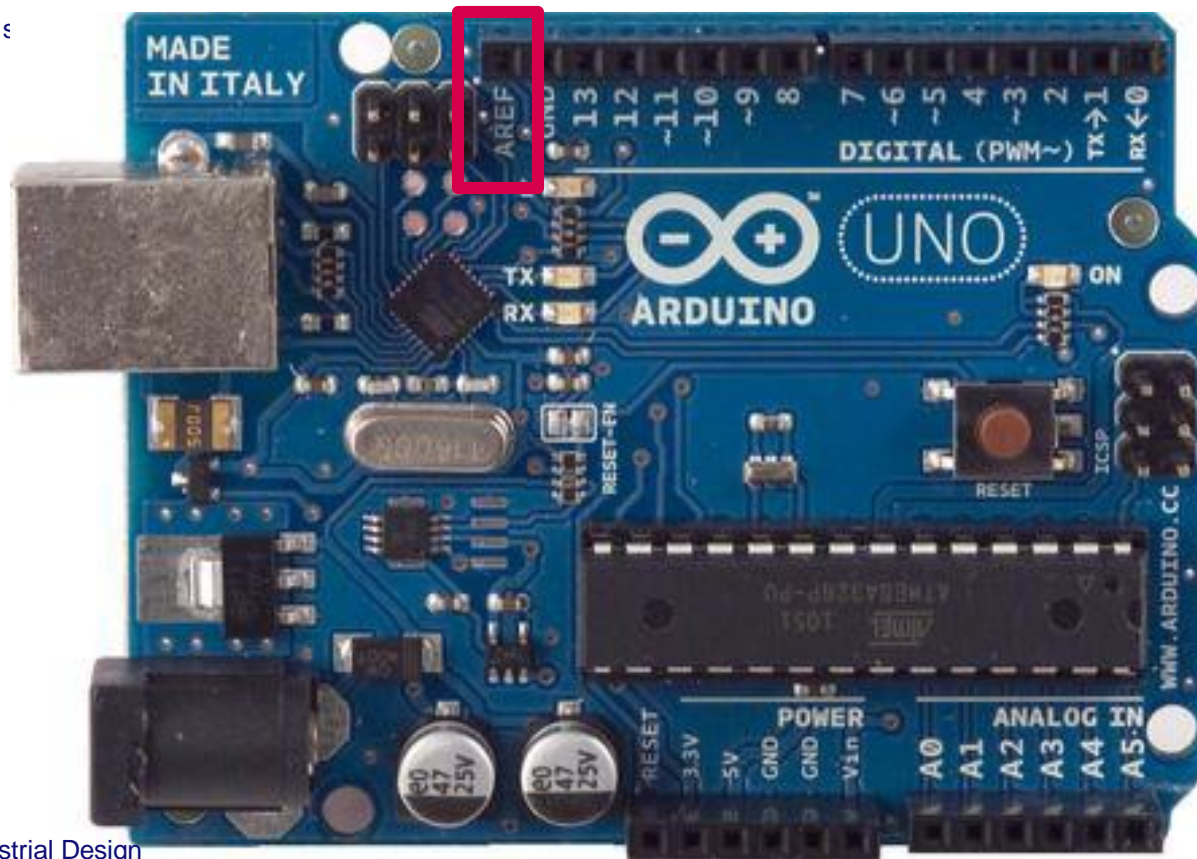
UNO

- 6 analog inputs, 10 bits of resolution (i.e. 1024 different values)



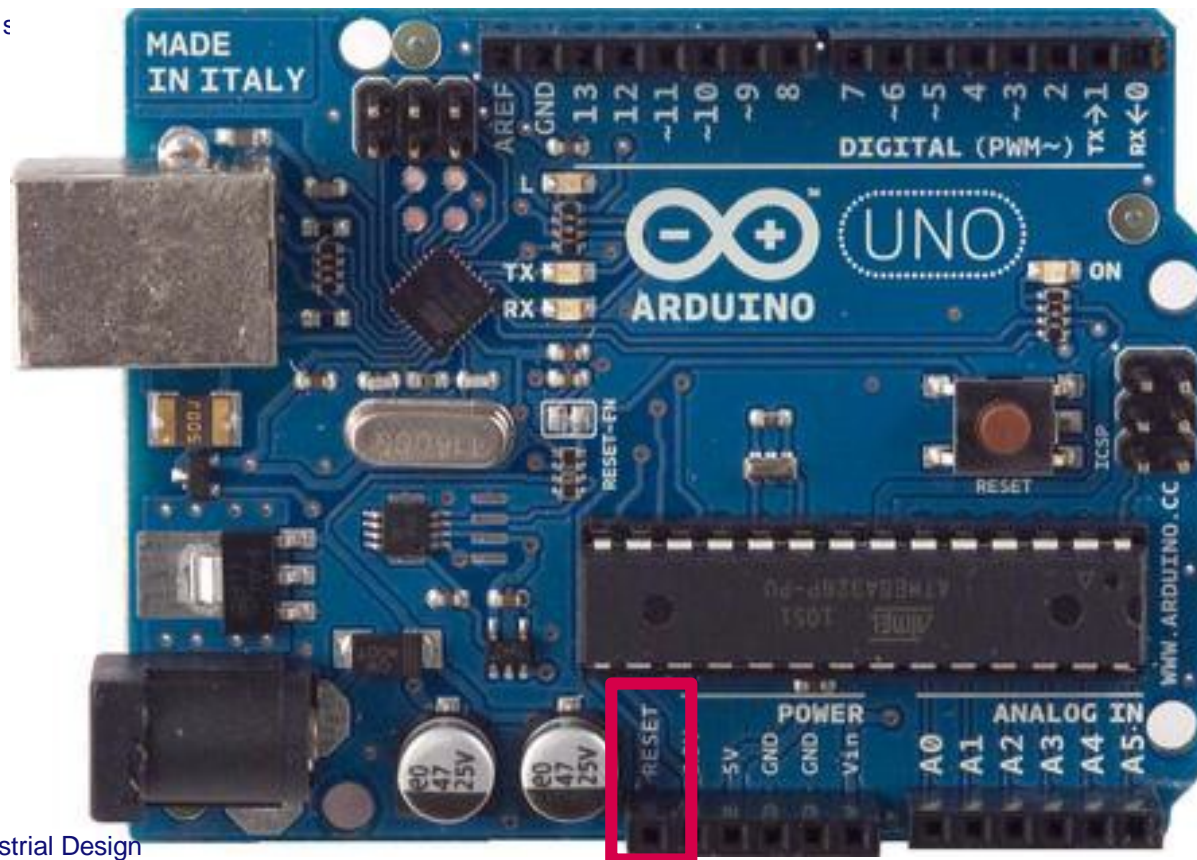
UNO

- **AREF:** Reference voltage for the analog inputs



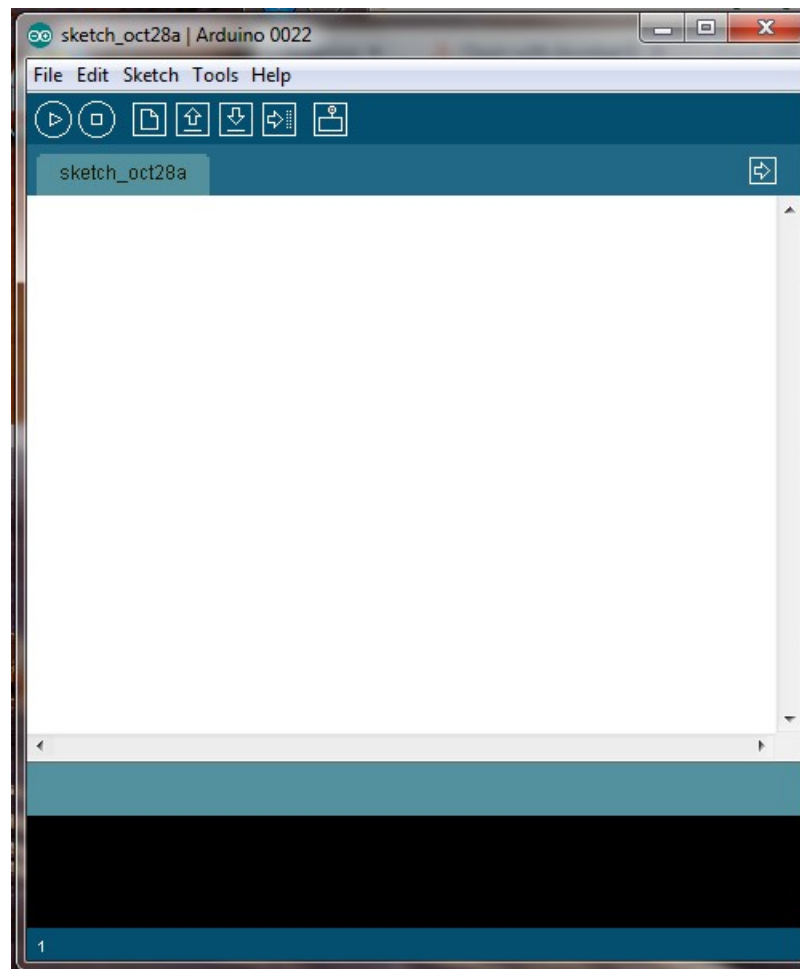
UNO

- **Reset. LOW to reset the microcontroller**



Software: IDE

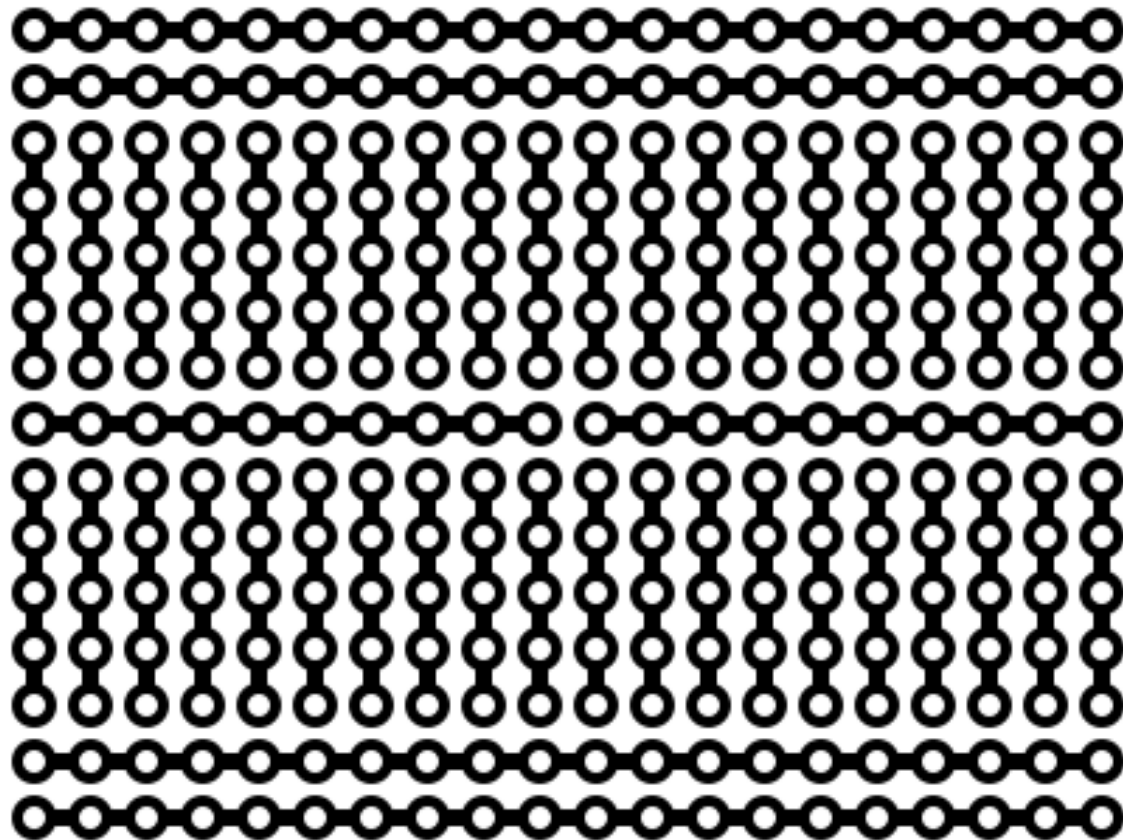
- <http://arduino.cc/en/Main/Software>



Driver Installation and Port Identification

- **Refer to the instructions in**
 - **“Getting Started with Arduino”, page 23-26**

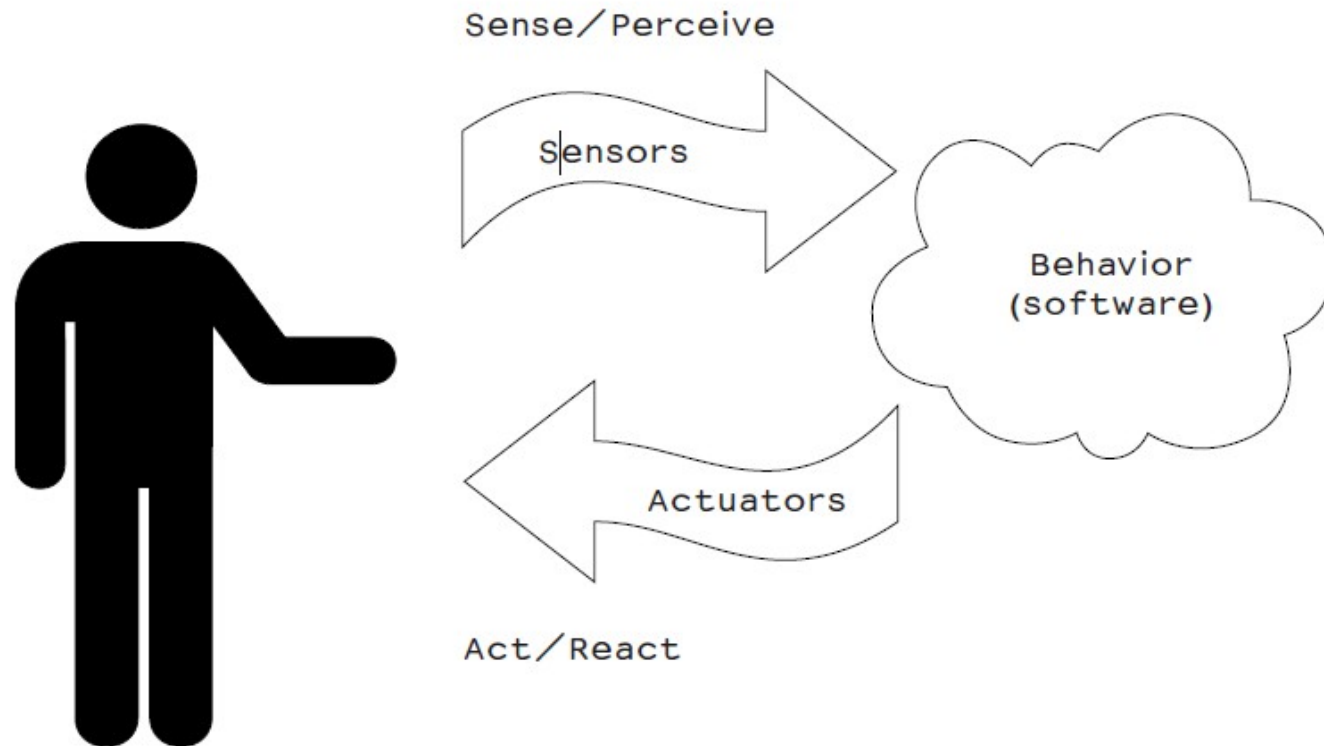
Breadboard



Color coding of the resistors

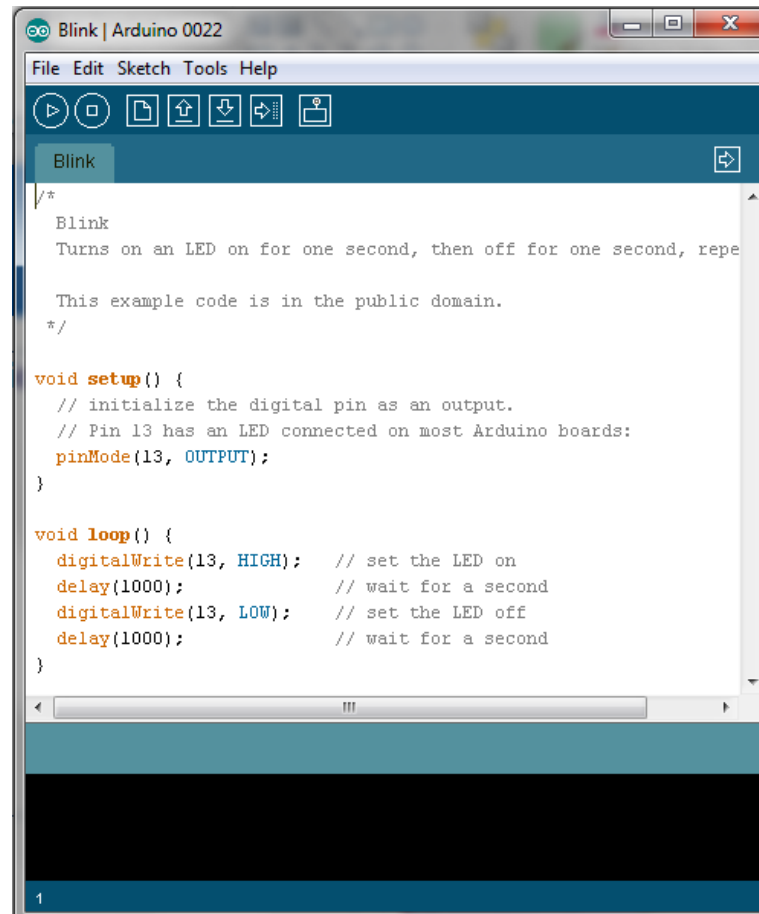
<p> </p> <p> Color Codes </p>	<p> </p> <p> 4 Band Resistors </p>	<p> </p> <p> 5 Band Resistors </p>	<p> </p> <p> 6 Band Resistors </p>
--------------------------------------	---	---	---

Really getting started



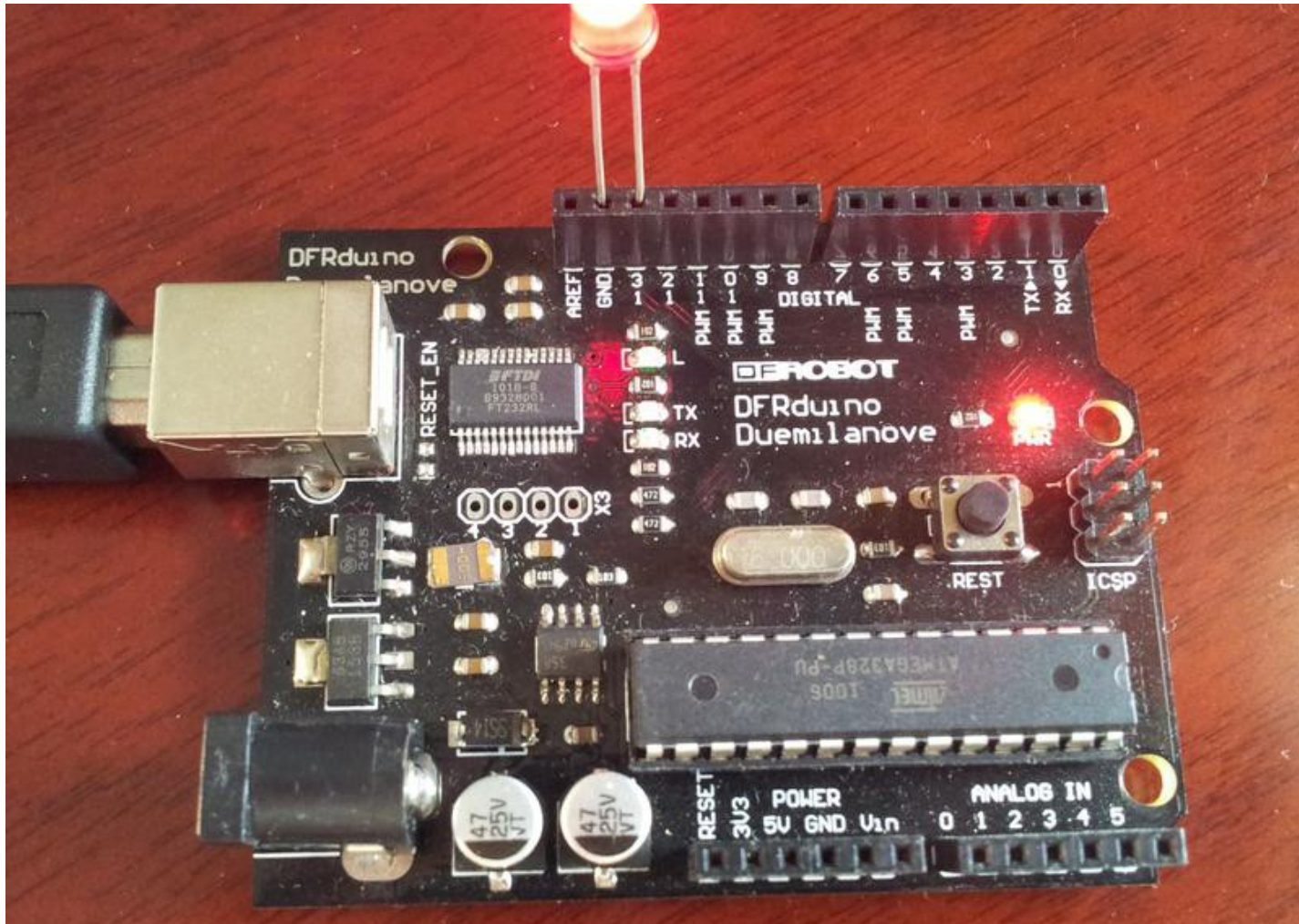
Blinking an LED

- **File>Examples>Basics>Blink**
 - **LED: light-emitting diode**

A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 0022". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for running, saving, opening, and other functions. The main text area shows the "Blink" sketch. The code is as follows:

```
/*  
  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  
}
```

Blinking an LED



Blink an LED

- **#define LED 13**

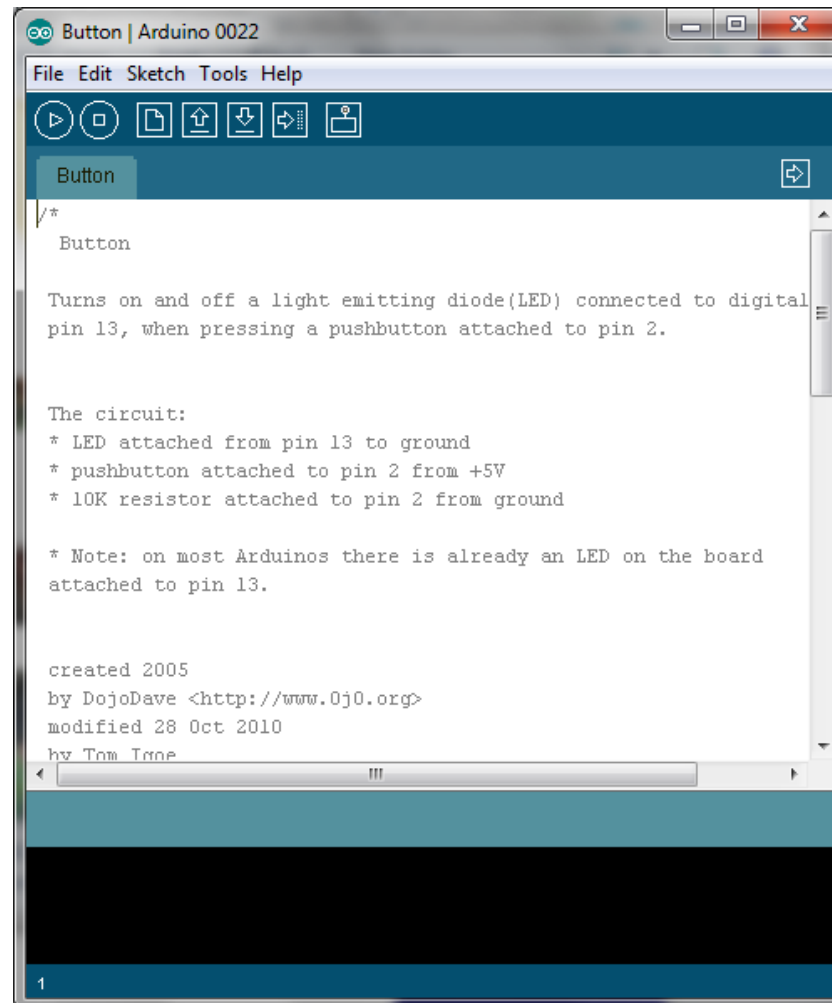
```
#define LED 13

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

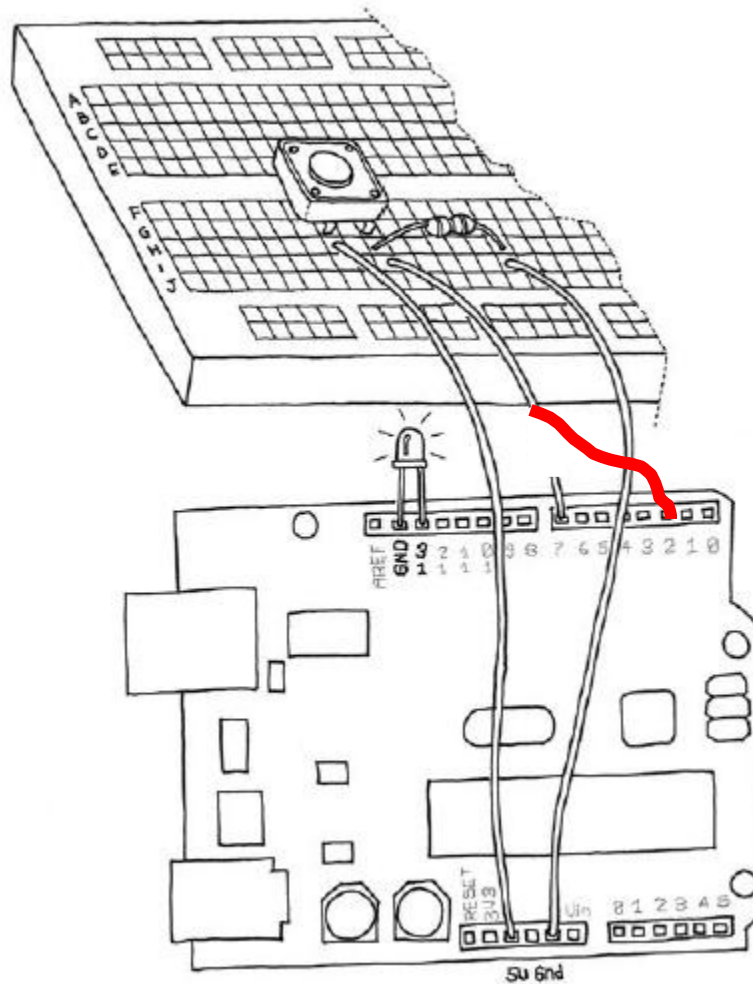
void loop() {
    digitalWrite(LED, HIGH);    // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(LED, LOW);     // set the LED off
    delay(1000);                // wait for a second
}
```


Button to control the LED

- **File>Examples>Digital>Button**



Button to control the LED



Button to control the LED

```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin = 13;      // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```

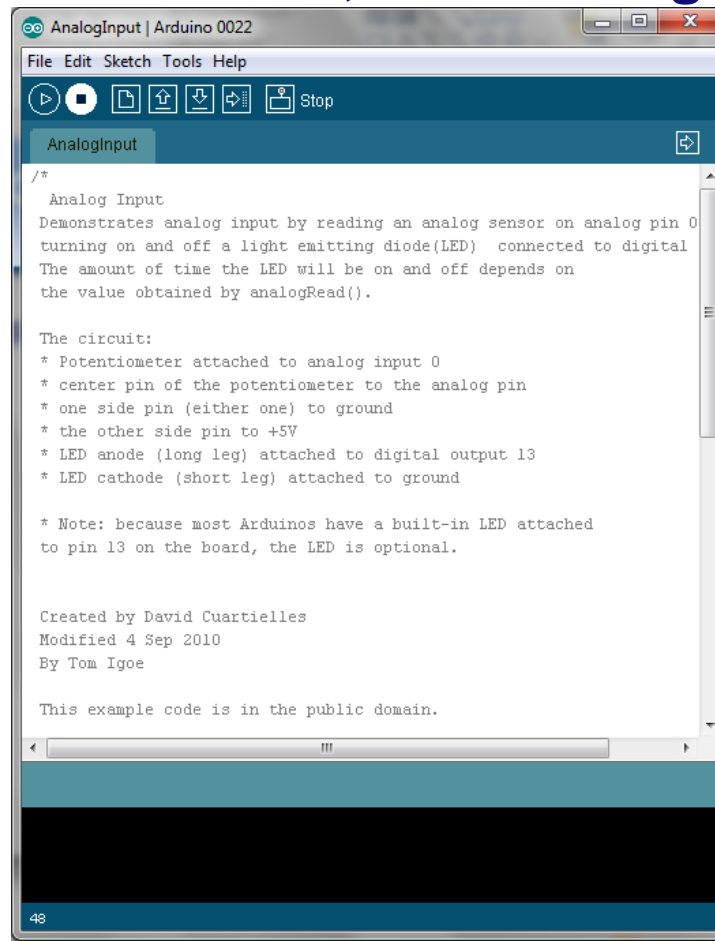
Button to control the LED

```
void loop(){
  // read the state of the pushbutton value:
  buttonState = digitalRead(buttonPin);

  // check if the pushbutton is pressed.
  // if it is, the buttonState is HIGH:
  if (buttonState == HIGH) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```


Analog I/O

- **File>Examples>Analog>AnalogInput**
 - Instead of a potentiometer, we use a light sensor



The screenshot shows the Arduino IDE interface with the 'AnalogInput' sketch loaded. The window title is 'AnalogInput | Arduino 0022'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. Below the menu bar is a toolbar with icons for running, uploading, and other functions. The sketch text is as follows:

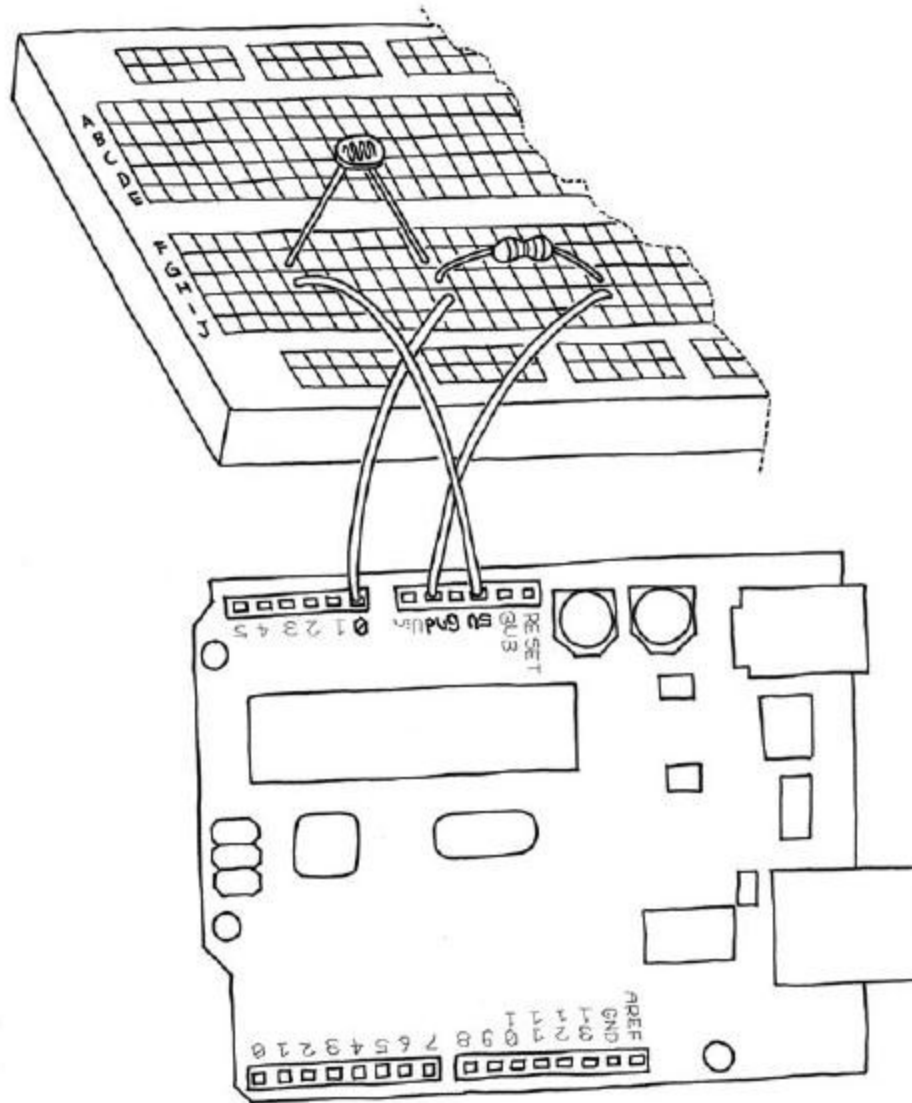
```
/*
  Analog Input
  Demonstrates analog input by reading an analog sensor on analog pin 0
  turning on and off a light emitting diode(LED) connected to digital
  The amount of time the LED will be on and off depends on
  the value obtained by analogRead().

  The circuit:
  * Potentiometer attached to analog input 0
  * center pin of the potentiometer to the analog pin
  * one side pin (either one) to ground
  * the other side pin to +5V
  * LED anode (long leg) attached to digital output 13
  * LED cathode (short leg) attached to ground

  * Note: because most Arduinos have a built-in LED attached
  to pin 13 on the board, the LED is optional.

  Created by David Cuartielles
  Modified 4 Sep 2010
  By Tom Igoe

  This example code is in the public domain.
*/
```



Analog I/O

```
int sensorPin = A0;    // select the input pin for the potentiometer
int ledPin = 13;       // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor

void setup() {
    // declare the ledPin as an OUTPUT:
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // read the value from the sensor:
    sensorValue = analogRead(sensorPin);
    // turn the ledPin on
    digitalWrite(ledPin, HIGH);
    // stop the program for <sensorValue> milliseconds:
    delay(sensorValue);
    // turn the ledPin off:
    digitalWrite(ledPin, LOW);
    // stop the program for for <sensorValue> milliseconds:
    delay(sensorValue);
}
```

Analog I/O

```
int sensorPin = A0;    // select the input pin
int ledPin = 11;       // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor

void setup() {
    // declare the ledPin as an OUTPUT:
    pinMode(ledPin, OUTPUT);
}

void loop() {
    // read the value from the sensor:
    sensorValue = analogRead(sensorPin);
    // turn the ledPin on
    analogWrite(ledPin, sensorValue/4);
}
```


Serial Communication

- We are going to use the Serial library from Processing to talk to Arduino
 - <http://processing.org/reference/libraries/serial/index.html>
- In Processing
 - File>Examples>Books>Getting Started>Ex_11_06
 - You can not run this program in Processing
 - Copy the code to Arduino software, upload to Arduino.

Serial Communication

Ex_11_06

```
// Example 11-06 from "Getting Started with Processing"  
// by Reas & Fry. O'Reilly / Make 2010  
  
// Note: This is code for an Arduino board, not Processing
```

```
int sensorPin = 0; // Select input pin  
int val = 0;
```

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

```
void loop() {  
  val = analogRead(sensorPin) / 4; // Read value from sensor  
  Serial.print(val, BYTE); // Print variable to serial port  
  delay(100); // Wait 100 milliseconds  
}
```

The 'BYTE' keyword is no longer supported.

As of Arduino 1.0, the 'BYTE' keyword is no longer supported.
Please use Serial.write() instead.

Serial Communication

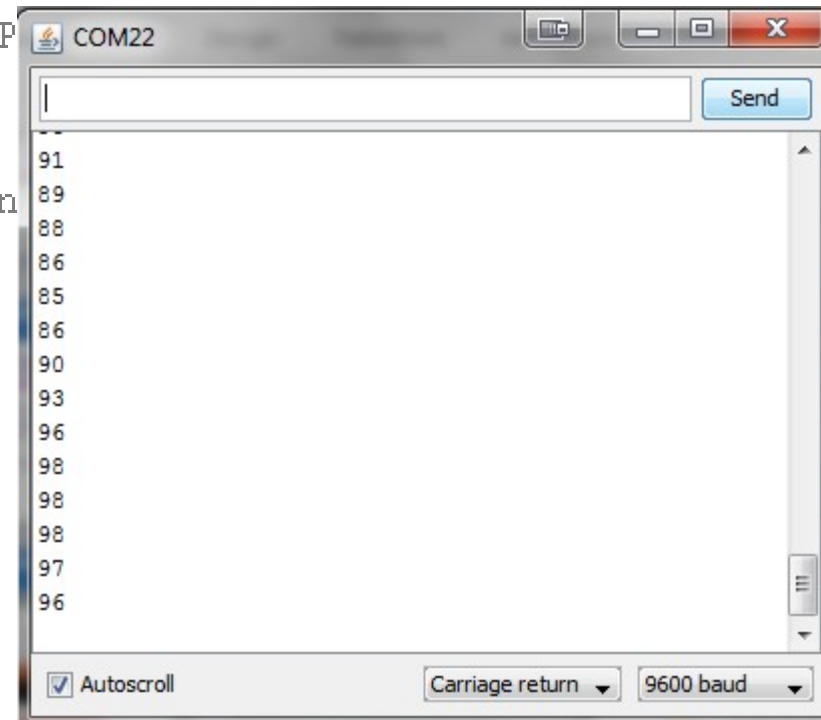
```
// Example 11-06 from "Getting Started with P
// by Reas & Fry. O'Reilly / Make 2010

// Note: This is code for an Arduino board, n

int sensorPin = 0; // Select input pin
int val = 0;

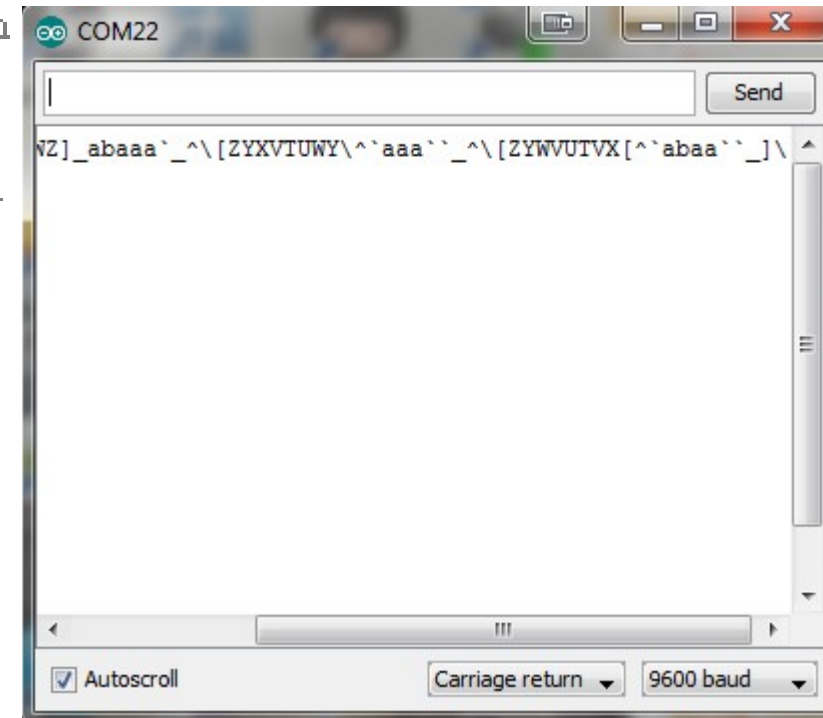
void setup() {
  Serial.begin(9600); // Open serial port
}

void loop() {
  val = analogRead(sensorPin) / 4; // Read value from sensor
  Serial.println(val); // Print variable to serial port
  delay(100); // Wait 100 milliseconds
}
```



Serial Communication

```
// Example 11-06 from "Getting Started with  
// by Reas & Fry. O'Reilly / Make 2010  
  
// Note: This is code for an Arduino board,  
  
int sensorPin = 0; // Select input pin  
int val = 0;  
  
void setup() {  
  Serial.begin(9600); // Open serial port  
}  
  
void loop() {  
  val = analogRead(sensorPin) / 4; // Read value from sensor  
  Serial.write(val); // Print variable to serial port  
  delay(100); // Wait 100 milliseconds  
}
```



Serial Communication

- **In Processing**
 - **File>Examples>Books>Chapter 11>Ex_11_07**

Serial Communication

```
import processing.serial.*;

Serial port; // Create object from Serial class
float val; // Data received from the serial port

void setup() {
    size(440, 220);
    // IMPORTANT NOTE:
    // The first serial port retrieved by Serial.list()
    // should be your Arduino. If not, uncomment the next
    // line by deleting the // before it. Run the sketch
    // again to see a list of serial ports. Then, change
    // the 0 in between [ and ] to the number of the port
    // that your Arduino is connected to.
    //println(Serial.list());
    String arduinoPort = Serial.list()[0];
    port = new Serial(this, arduinoPort, 9600);
}

void draw() {
    if (port.available() > 0) { // If data is available,
        val = port.read(); // read it and store it in val
        val = map(val, 0, 255, 0, height); // Convert the value
    }
    rect(40, val-10, 360, 20);
}
```

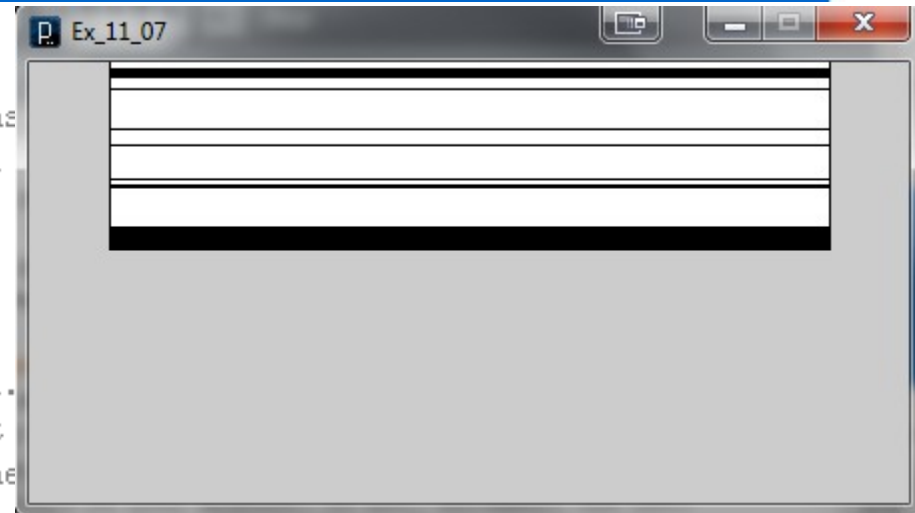
Serial Communication

```
import processing.serial.*;

Serial port; // Create object from Serial class
float val;   // Data received from the serial

void setup() {
    size(440, 220);
    // IMPORTANT NOTE:
    // The first serial port retrieved by Serial.
    // should be your Arduino. If not, uncomment
    // line by deleting the // before it. Run the
    // again to see a list of serial ports. Then, change
    // the 0 in between [ and ] to the number of the port
    // that your Arduino is connected to.
    println(Serial.list());
    String arduinoPort = Serial.list()[1];
    port = new Serial(this, arduinoPort, 9600);
}

void draw() {
    if (port.available() > 0) { // If data is available,
        val = port.read();      // read it and store it in val
        val = map(val, 0, 255, 0, height); // Convert the value
    }
    rect(40, val-10, 360, 20);
}
```



Serial Communication

```
import processing.serial.*;

Serial port; // Create object from Serial class
float val; // Data received from the serial port

void setup() {
    size(440, 220);
    // IMPORTANT NOTE:
    // The first serial port retrieved by Serial.
    // should be your Arduino. If not, uncomment
    // line by deleting the // before it. Run the
    // again to see a list of serial ports. Then,
    // the 0 in between [ and ] to the number of the port
    // that your Arduino is connected to.
    println(Serial.list());
    String arduinoPort = Serial.list()[1];
    port = new Serial(this, arduinoPort, 9600);
}

void draw() {
    background(0);
    if (port.available() > 0) { // If data is available,
        val = port.read(); // read it and store it in val
        val = map(val, 0, 255, 0, height); // Convert the value
    }
    rect(40, val-10, 360, 20);
}
```



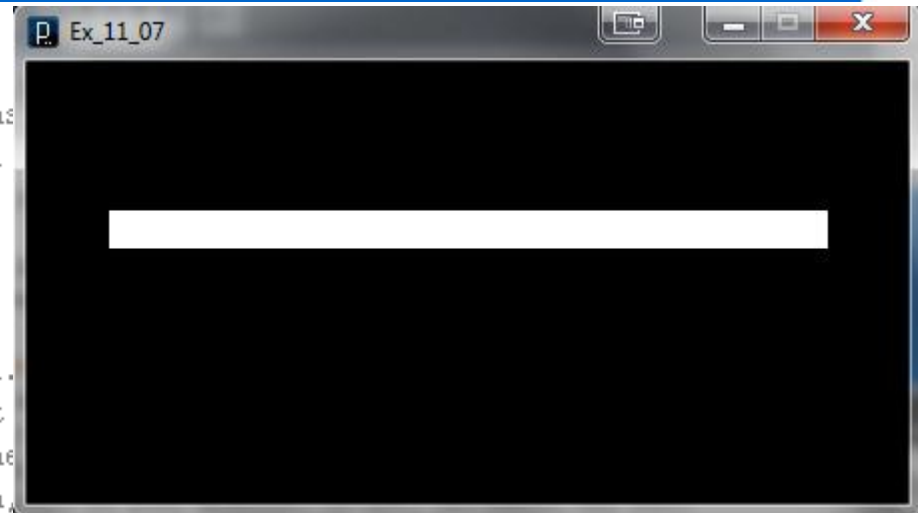
Serial Communication

```
import processing.serial.*;

Serial port; // Create object from Serial class
float val; // Data received from the serial port

void setup() {
  size(440, 220);
  // IMPORTANT NOTE:
  // The first serial port retrieved by Serial.list()
  // should be your Arduino. If not, uncomment
  // line by deleting the // before it. Run the
  // again to see a list of serial ports. Then,
  // the 0 in between [ and ] to the number of the port
  // that your Arduino is connected to.
  //println(Serial.list());
  //String arduinoPort = Serial.list()[1];
  port = new Serial(this, "COM22", 9600);
}

void draw() {
  background(0);
  if (port.available() > 0) { // If data is available,
    val = port.read(); // read it and store it in val
    val = map(val, 0, 255, 0, height); // Convert the value
  }
  rect(40, val-10, 360, 20);
}
```



Serial Communication (Firmata)

- **Introducing Firmata**
 - <http://playground.arduino.cc/Interfacing/processing>

Download

Library for Processing v2.0: [processing2-arduino.zip](#) (Updated 6 Nov. 2013)
(properties file here: [processing2-arduino.txt](#))

Library for Processing v1.5: [processing-arduino.zip](#) (Updated 11 Nov. 2011)
(properties file here: [processing-arduino.txt](#))

Note: if you run Linux, you need to change Arduino.jar into arduino.jar, because Linux is case sensitive and it does not work if you don't change this letter (Arduino.jar is in the folder "library" of this Processing Library).

Serial Communication (Firmata)

Instructions

1. Unzip the library and copy the "arduino" folder into the "libraries" sub-folder of your Processing Sketchbook. (You can find the location of your Sketchbook by opening the Processing Preferences. If you haven't made a "libraries" sub-folder, create one.)
2. Run Arduino, open the Examples > Firmata > StandardFirmata sketch, and upload it to the Arduino board.
3. Configure Processing for serial: <http://processing.org/reference/libraries/serial/>
4. In Processing, open one of the examples that comes with with the Arduino library.
5. Edit the example code to select the serial port used by Arduino. Specifically, change the

[0] in this line

```
arduino = new Arduino(this, Arduino.list()[0], 57600);
```

To find the correct item in the array, run this code in Processing:

```
import processing.serial.*;  
import cc.arduino.*;  
println(Arduino.list());
```

The output window will enumerate your serial ports. Select the number corresponding to the serial port in your Arduino environment found under Tools > Serial Port.

6. Run the example.

Serial Communication (Firmata)

- **Introducing Firmata**

Example

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

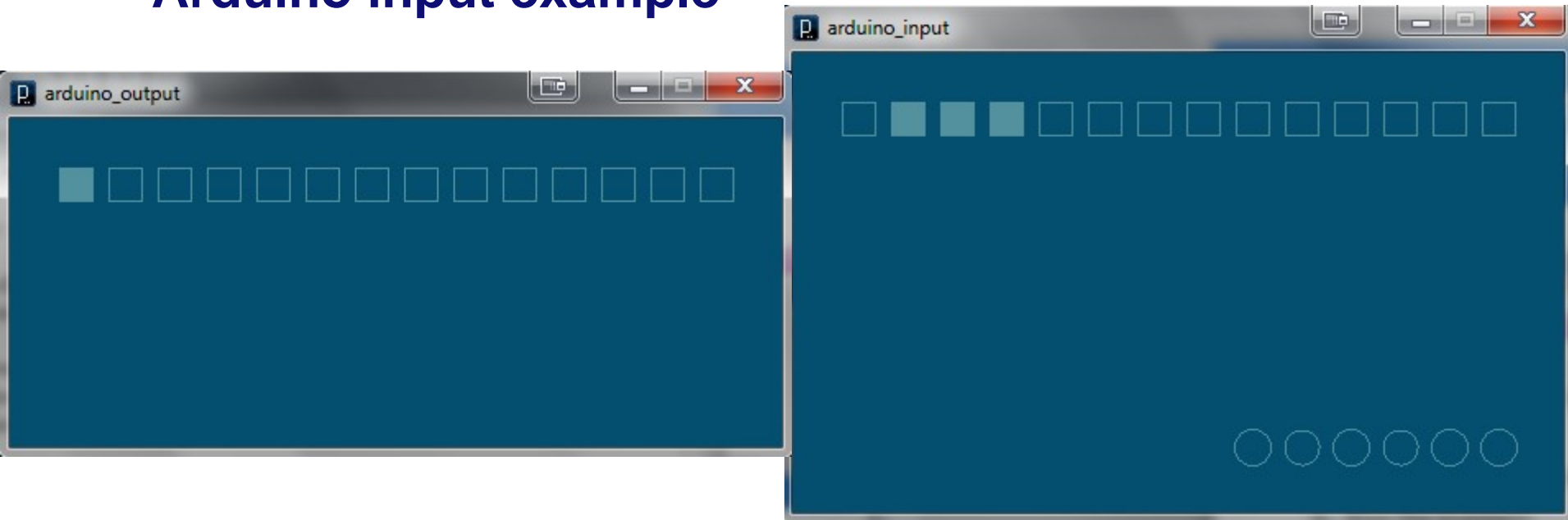
Arduino arduino;
int ledPin = 13;

void setup()
{
  //println(Arduino.list());
  arduino = new Arduino(this, Arduino.list()[0], 57600);
  arduino.pinMode(ledPin, Arduino.OUTPUT);
}

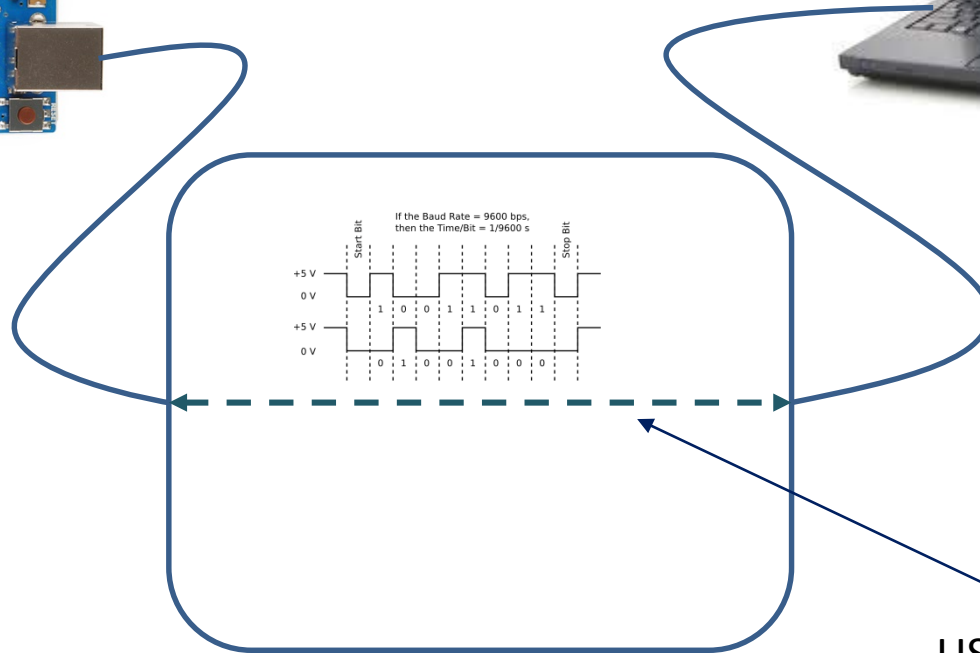
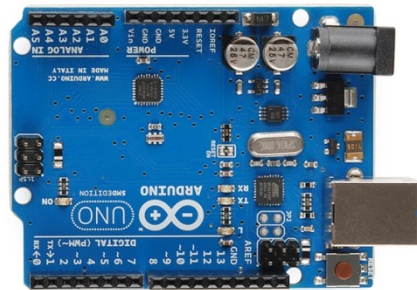
void draw()
{
  arduino.digitalWrite(ledPin, Arduino.HIGH);
  delay(1000);
  arduino.digitalWrite(ledPin, Arduino.LOW);
  delay(1000);
}
```

Serial Communication (Firmata)

- **Arduino output example**
- **Arduino input example**



Arduino \leftrightarrow notebook



USB cable

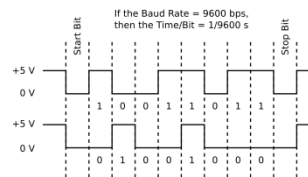
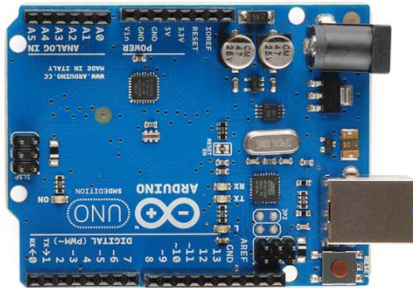


Designed
Intelligence
Group

TU/e

Technische Universiteit
Eindhoven
University of Technology

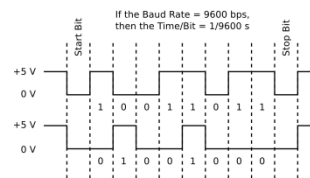
Arduino \leftrightarrow notebook





Scenario

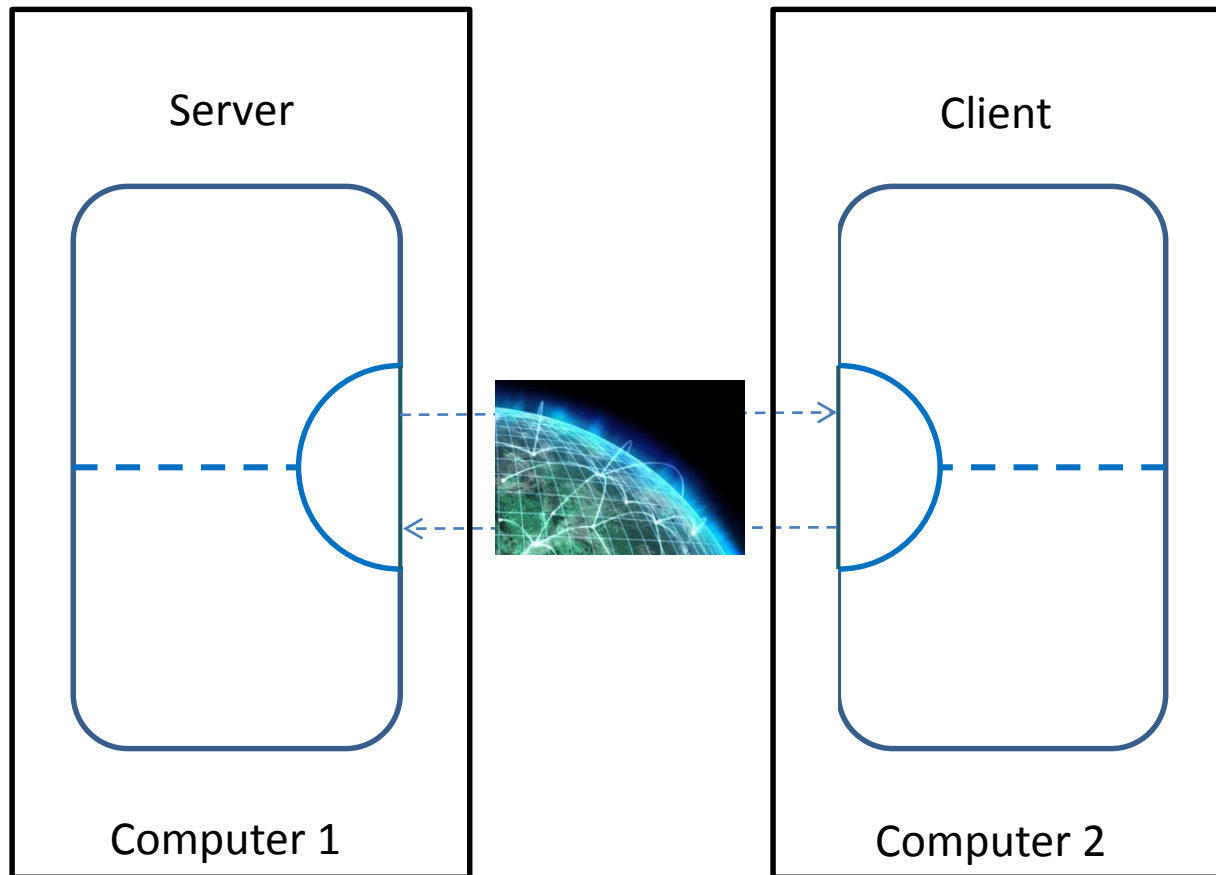
- `Integer.parseInt()`
- `Integer.valueOf()`
- `Integer.parseInt()`
- `Integer.parseInt()`



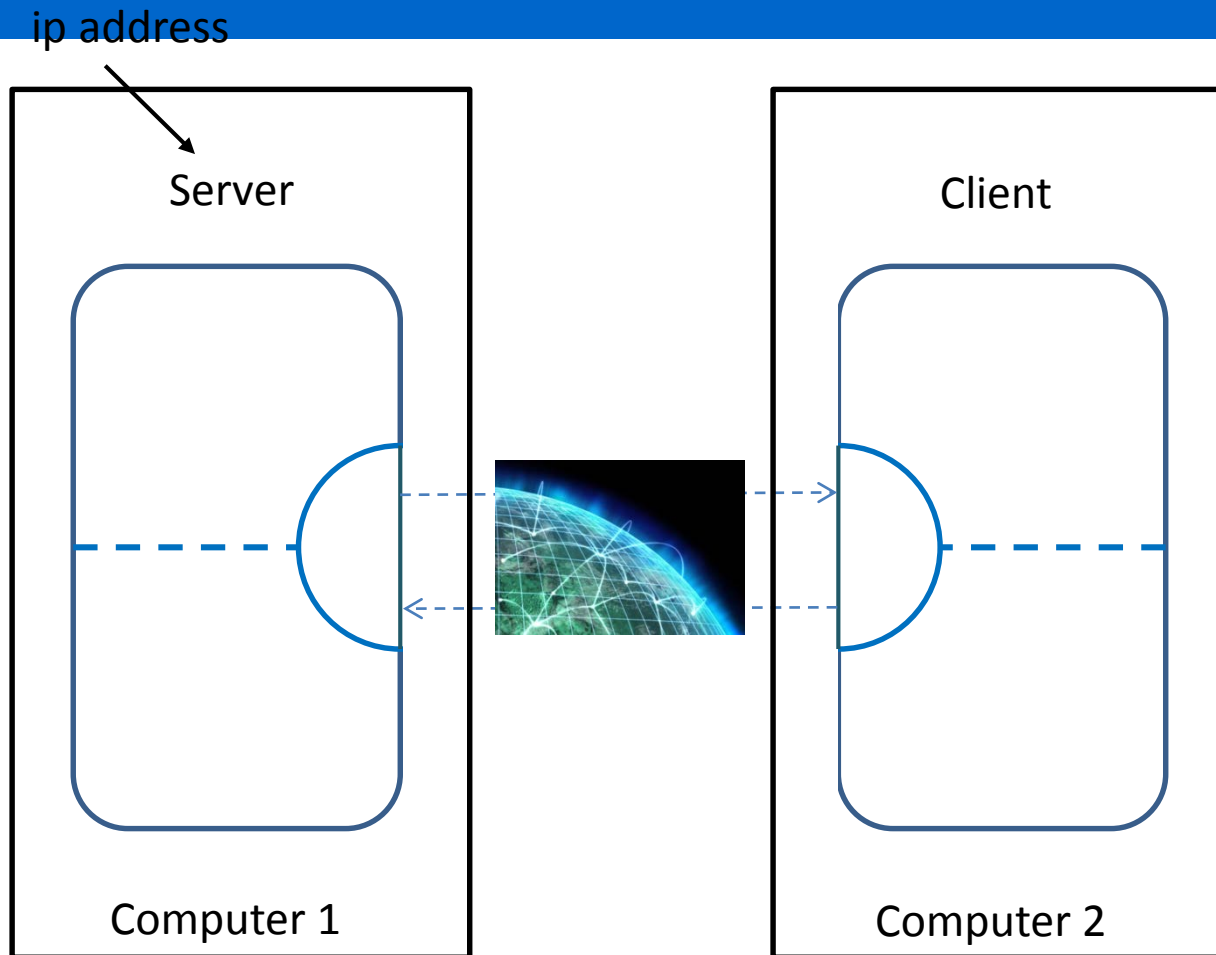
USB cable



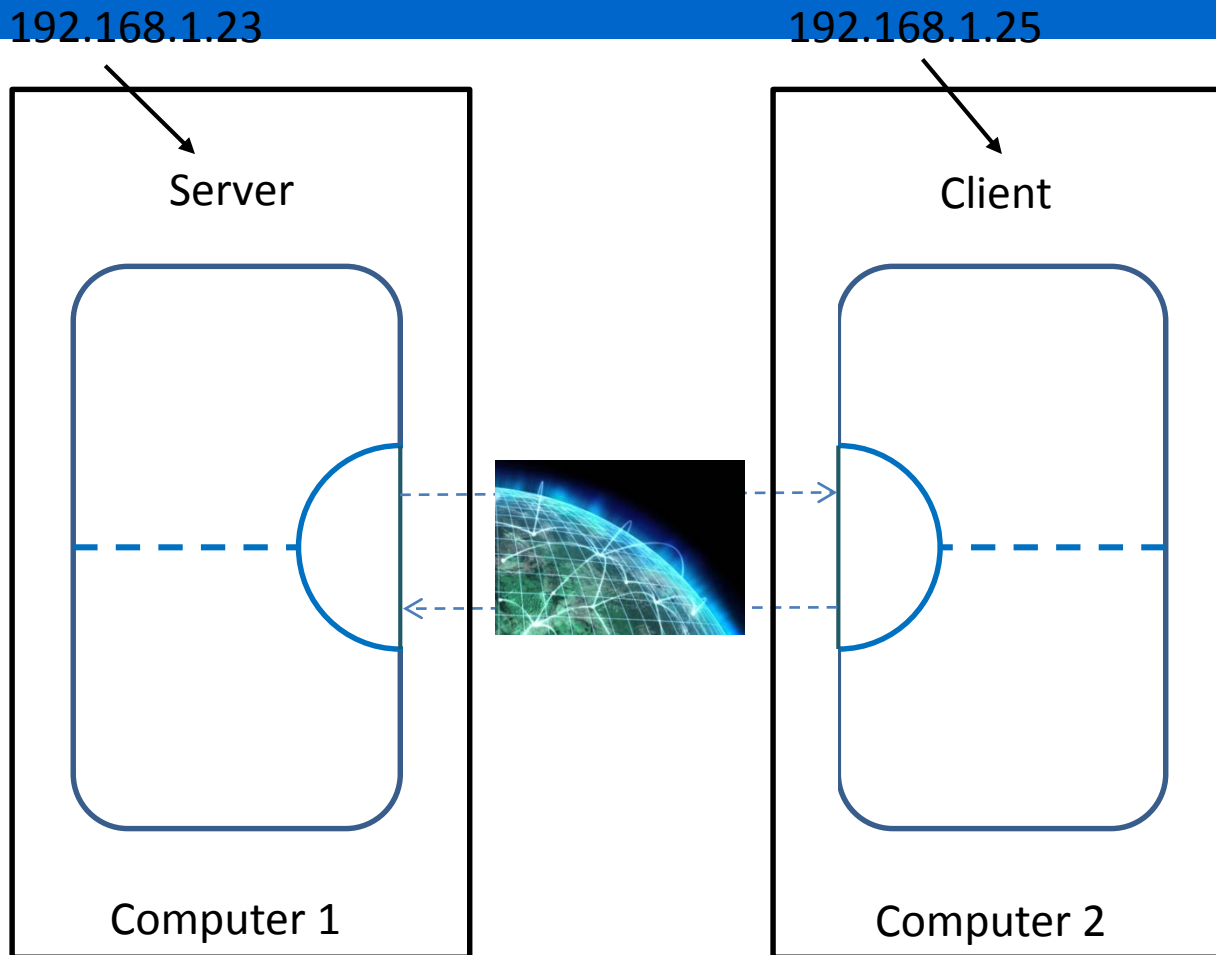
Socket communication



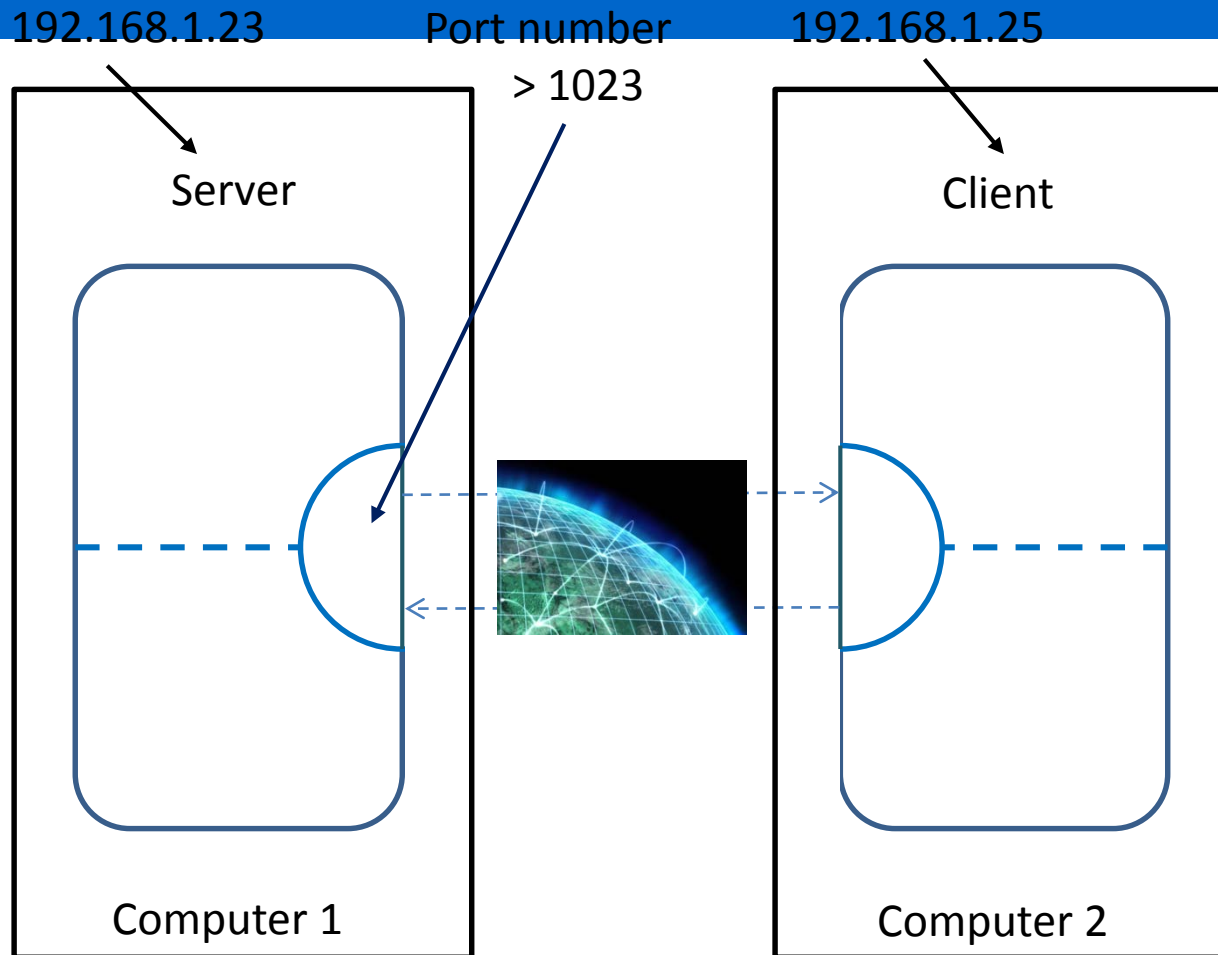
Socket communication



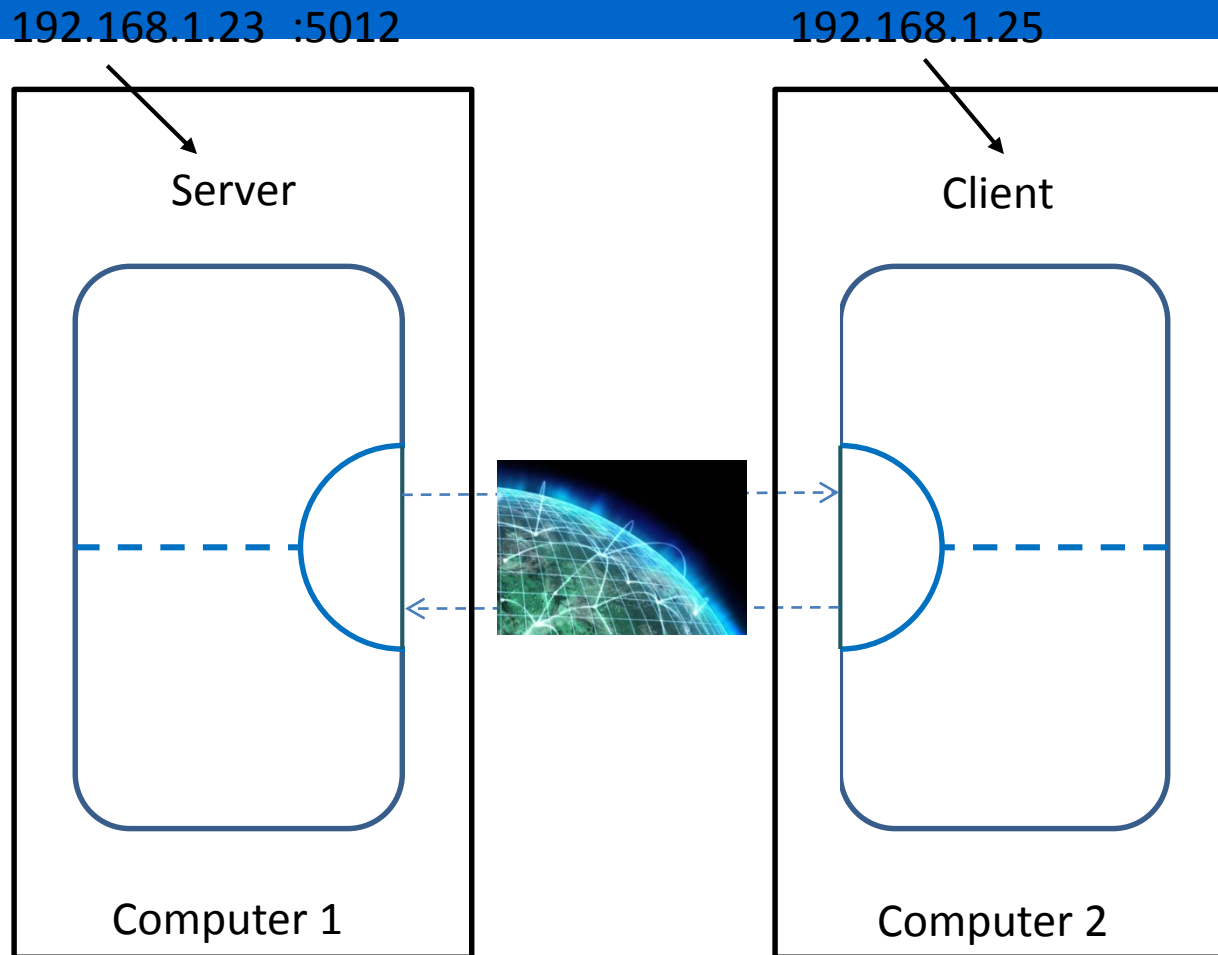
Socket communication



Socket communication



Socket communication



Ways to connect to Internet

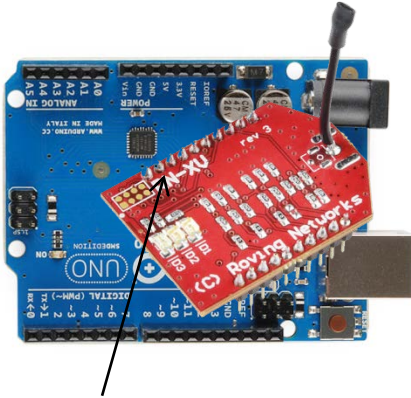


Designed
Intelligence
Group

TU/e

Technische Universiteit
Eindhoven
University of Technology

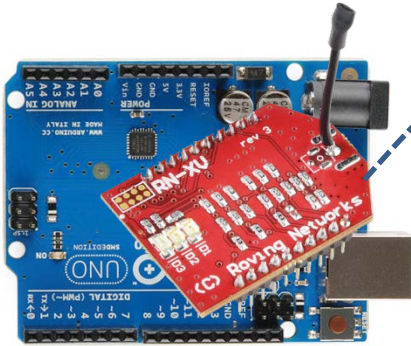
Ways to connect to Internet



Wifi Module



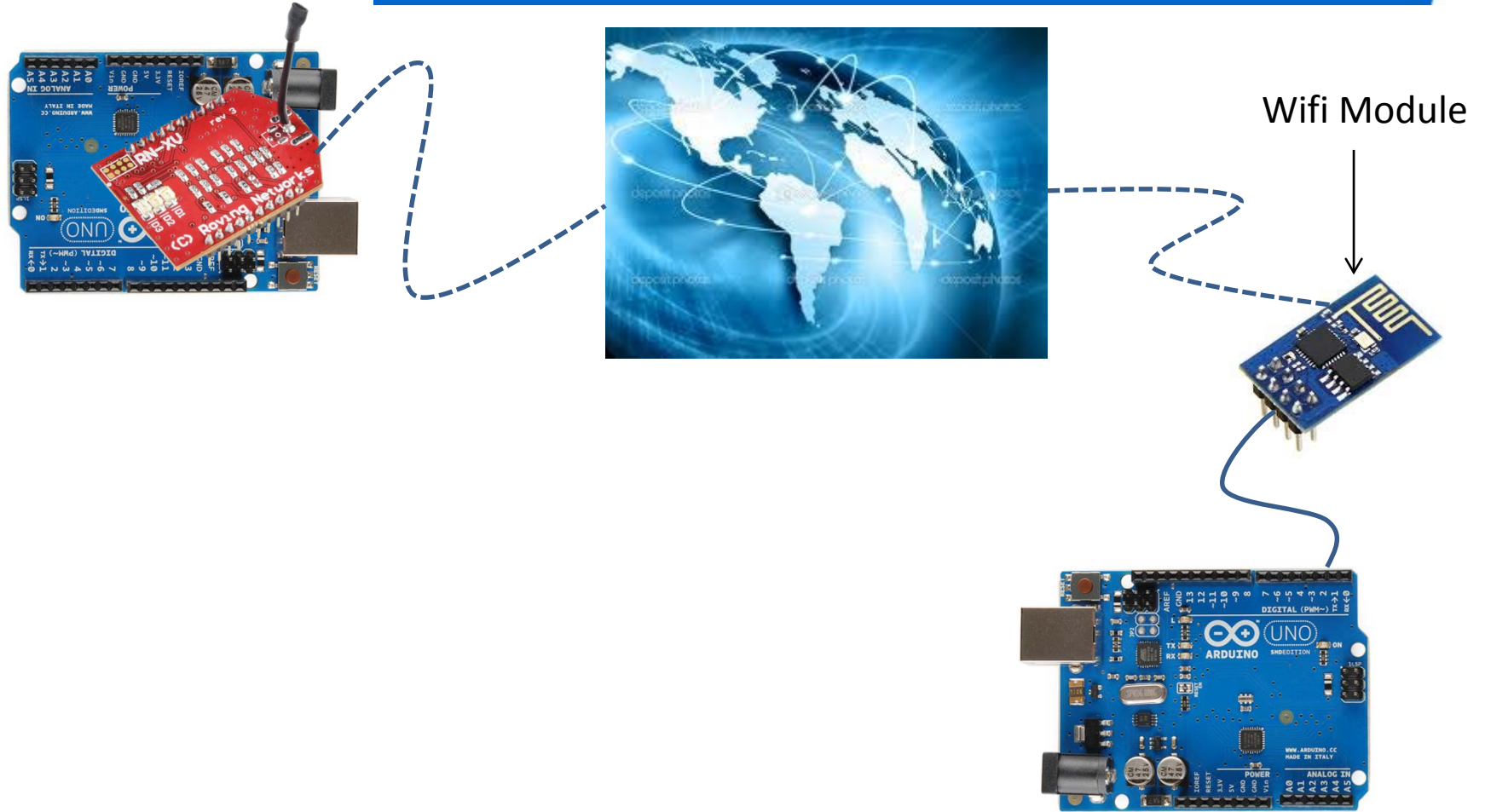
Ways to connect to Internet



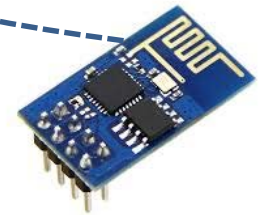
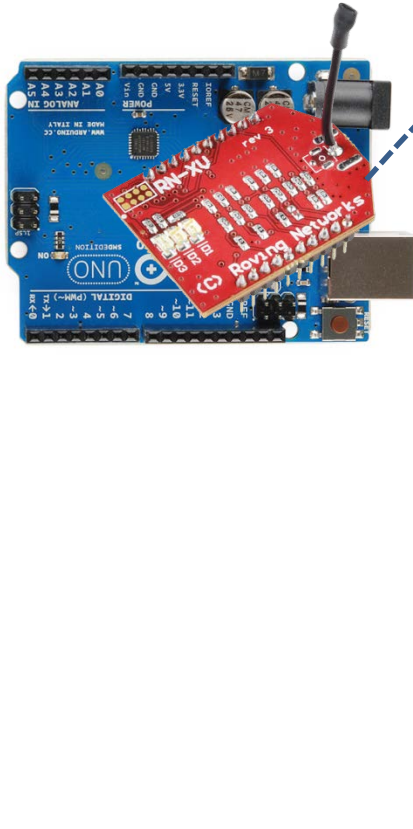
Ways to connect to Internet



Ways to connect to Internet



Ways to connect to Internet



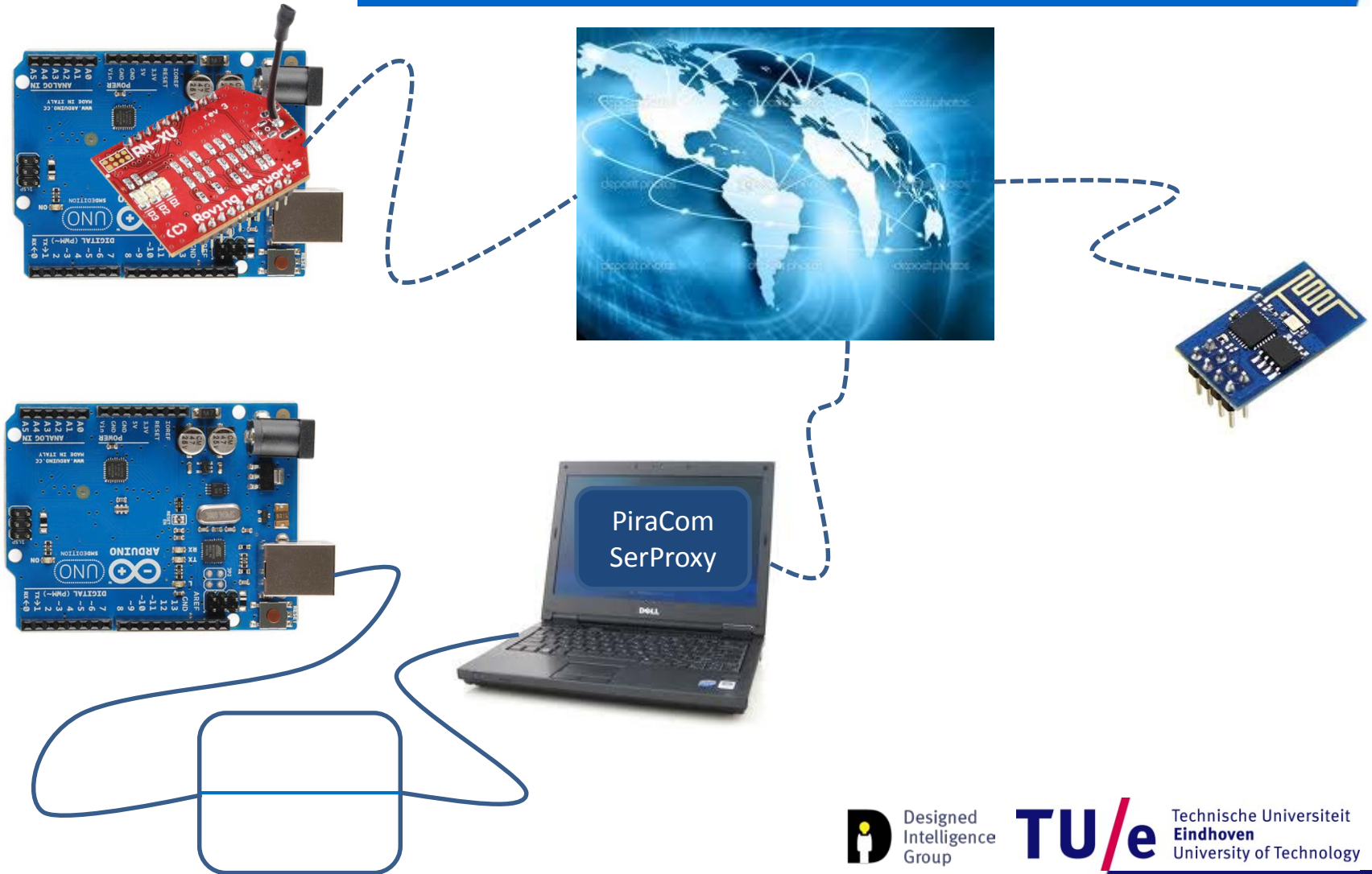
Ways to connect to Internet



Ways to connect to Internet

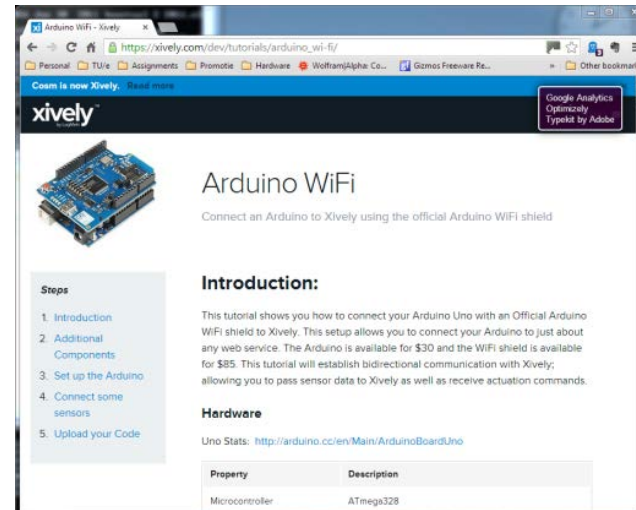


Ways to connect to Internet



Possibilities

- Make sensors available for reading worldwide
- Make actuators available worldwide
- Send sensor values to server location where it can be read by many.



Additional Resources

You may look at:

- Processing Electronics tutorial
<https://processing.org/tutorials/electronics/>
- Processing Network tutorial
<https://processing.org/tutorials/network/>
- Tutorial serial communication Arduino – Processing
<https://www.youtube.com/watch?v=g0pSfyXOXj8&feature=kp>
- Tutorial wireless communication (Zigbee)
<https://www.youtube.com/watch?v=vKVNmA8C6m8>
- Xively/Pachube/Cosm
<http://xively.com/>, <https://xively.com/dev/tutorials/>
- ESP8266
<http://www.instructables.com/id/ESP8266-mini-Tutorial/?ALLSTEPS>

And Many many more...



Designed
Intelligence
Group

TU/e

Technische Universiteit
Eindhoven
University of Technology

That was Arduino



Technische Universiteit
Eindhoven
University of Technology