



# Arduino

**TU** / **e**

Technische Universiteit  
**Eindhoven**  
University of Technology

**Where innovation starts**

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

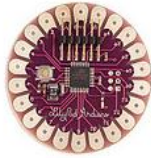
# Why Arduino?

- **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



# UNO



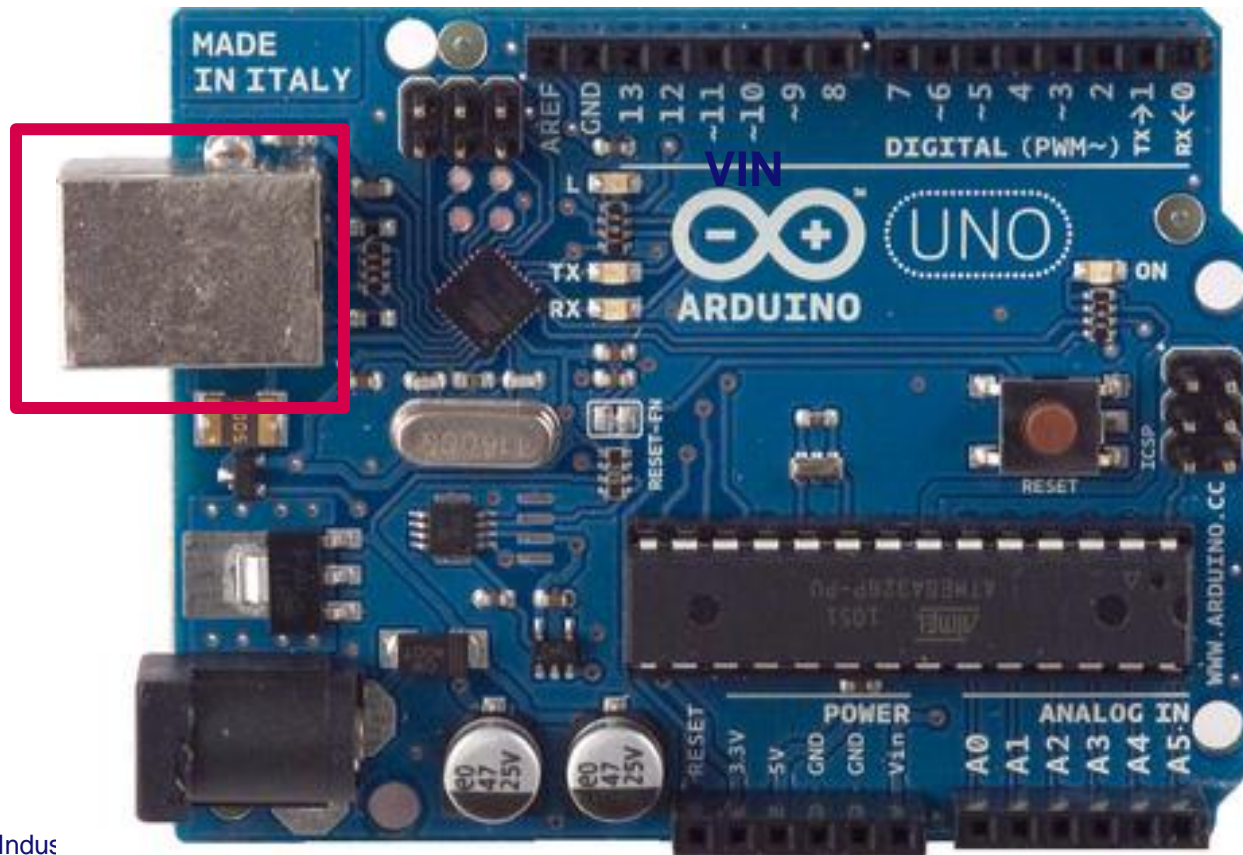
# UNO

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



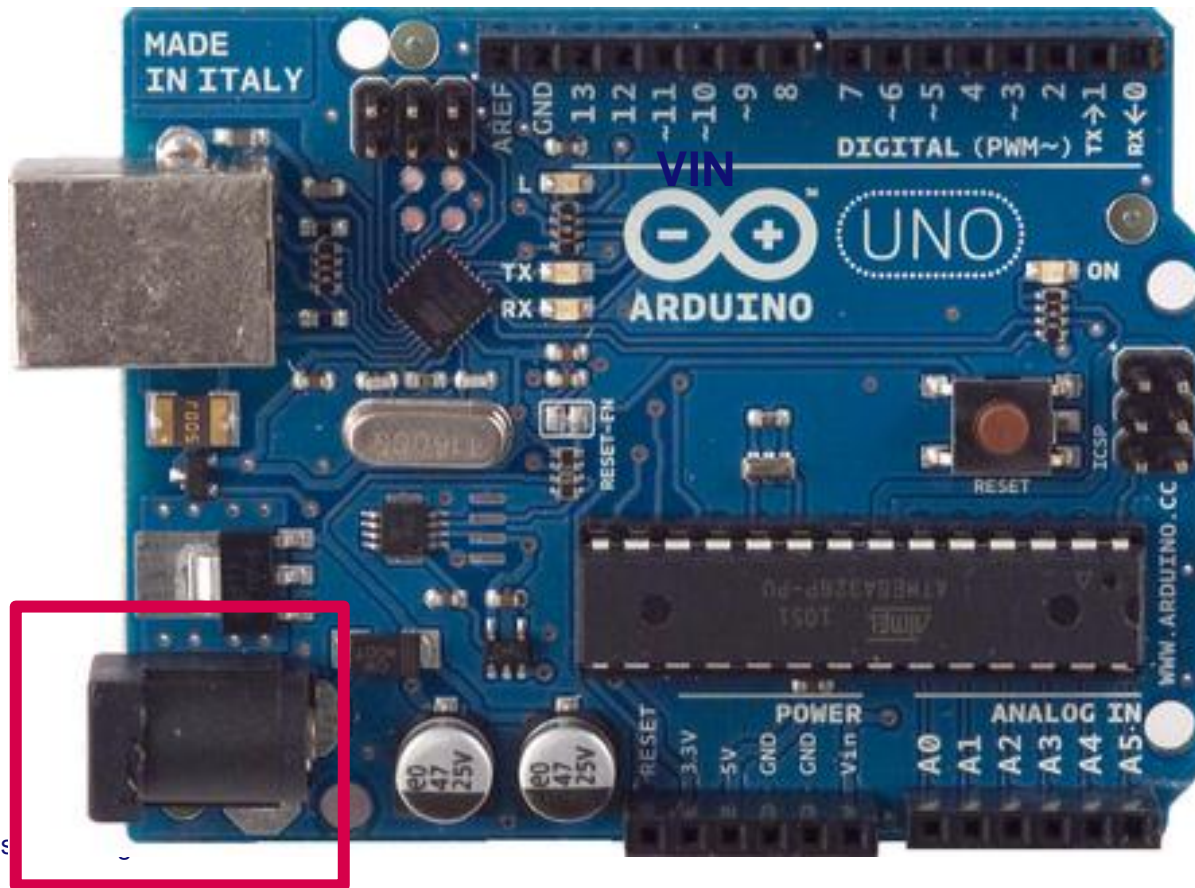
# UNO

- Power: USB Power supply (5V)



# UNO

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





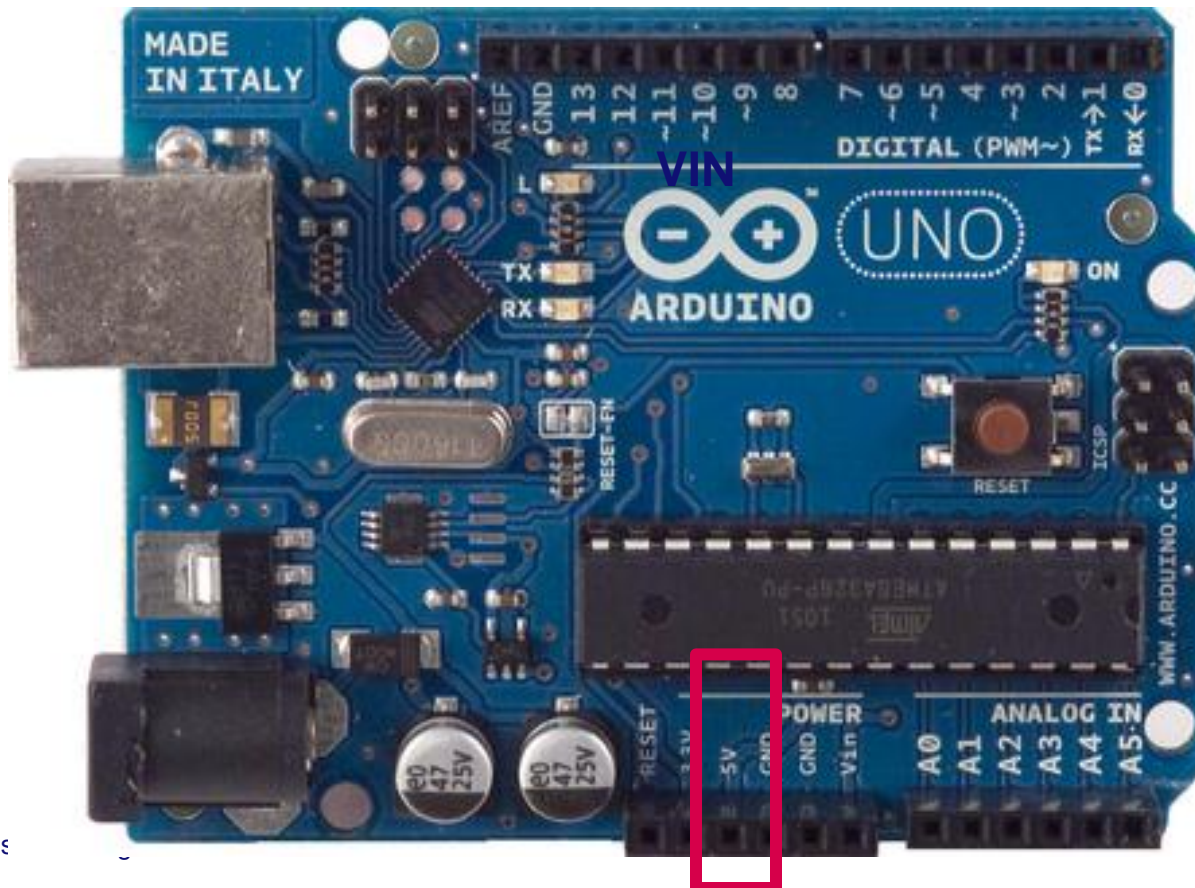
# UNO

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



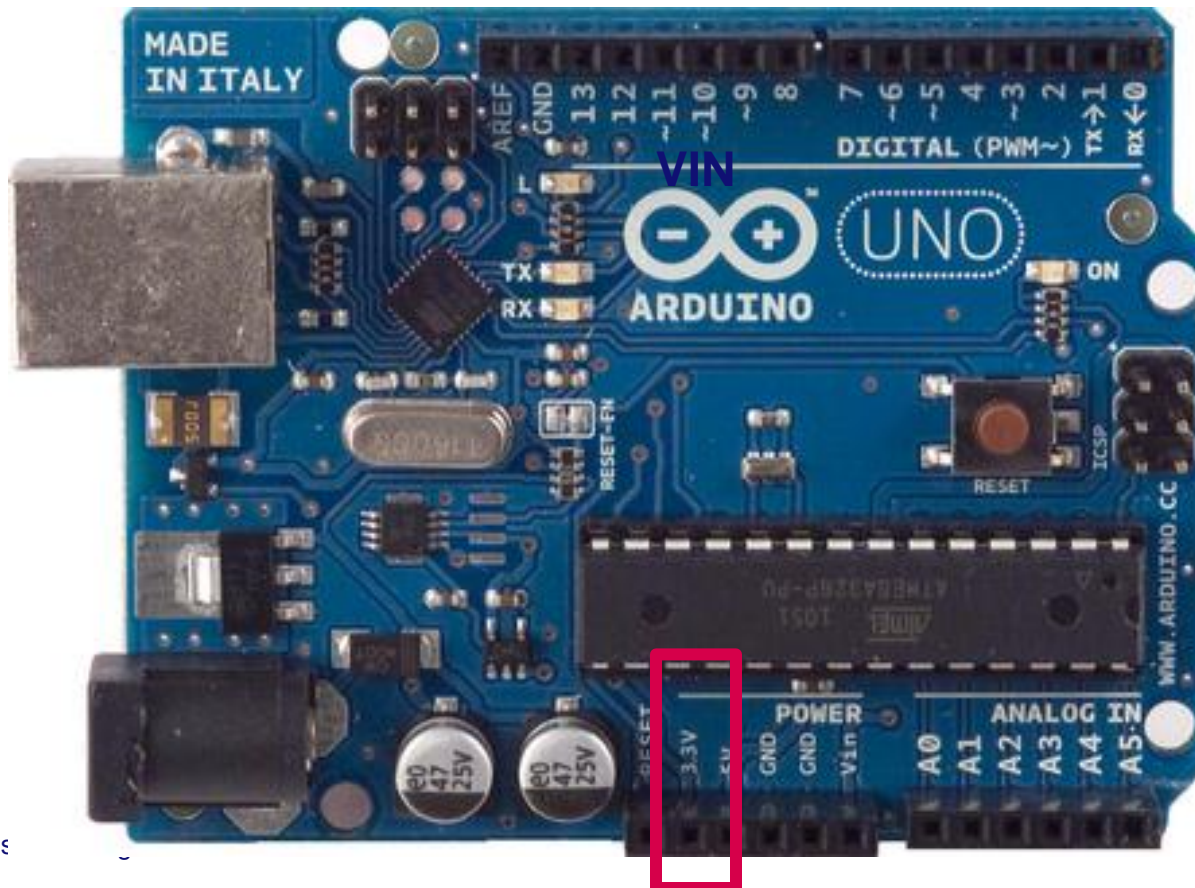
# UNO

- Power: 5V supply



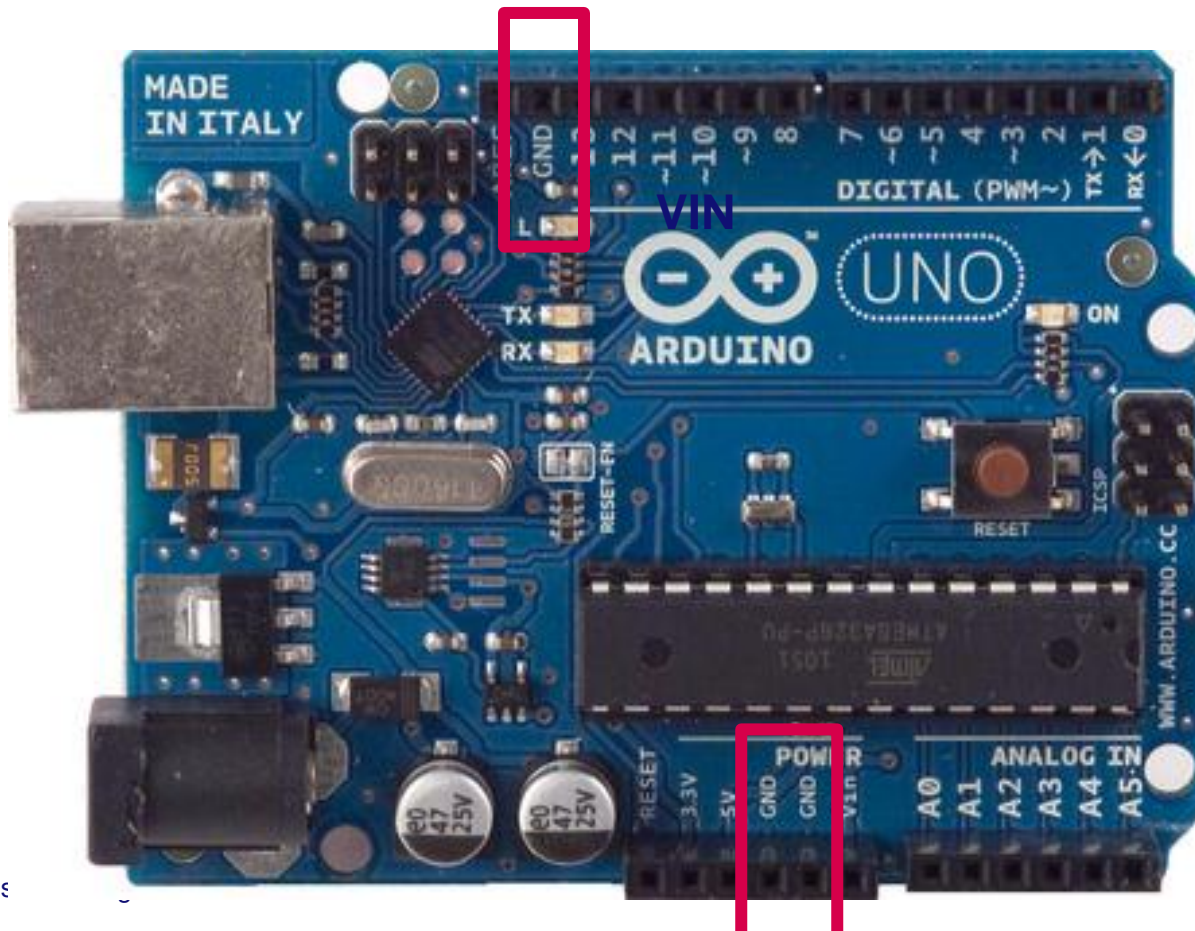
# UNO

- Power: 3.3V supply



# UNO

- Power: GND pins



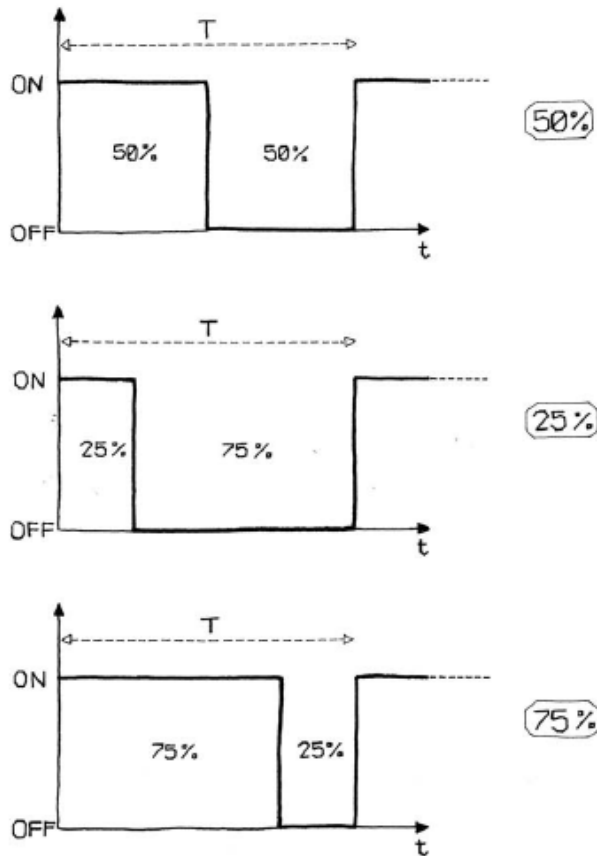
# UNO

- 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)



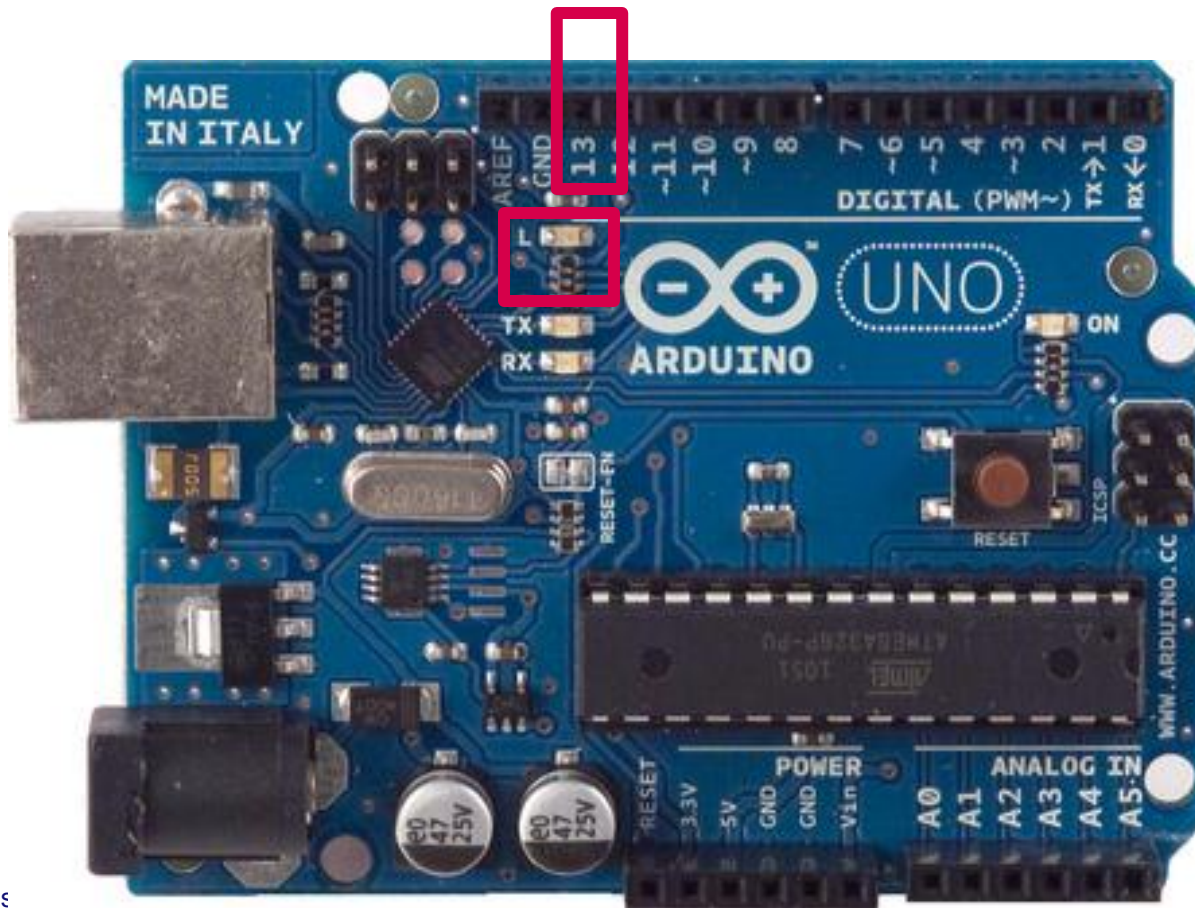
# UNO

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



# UNO

- LED: 13





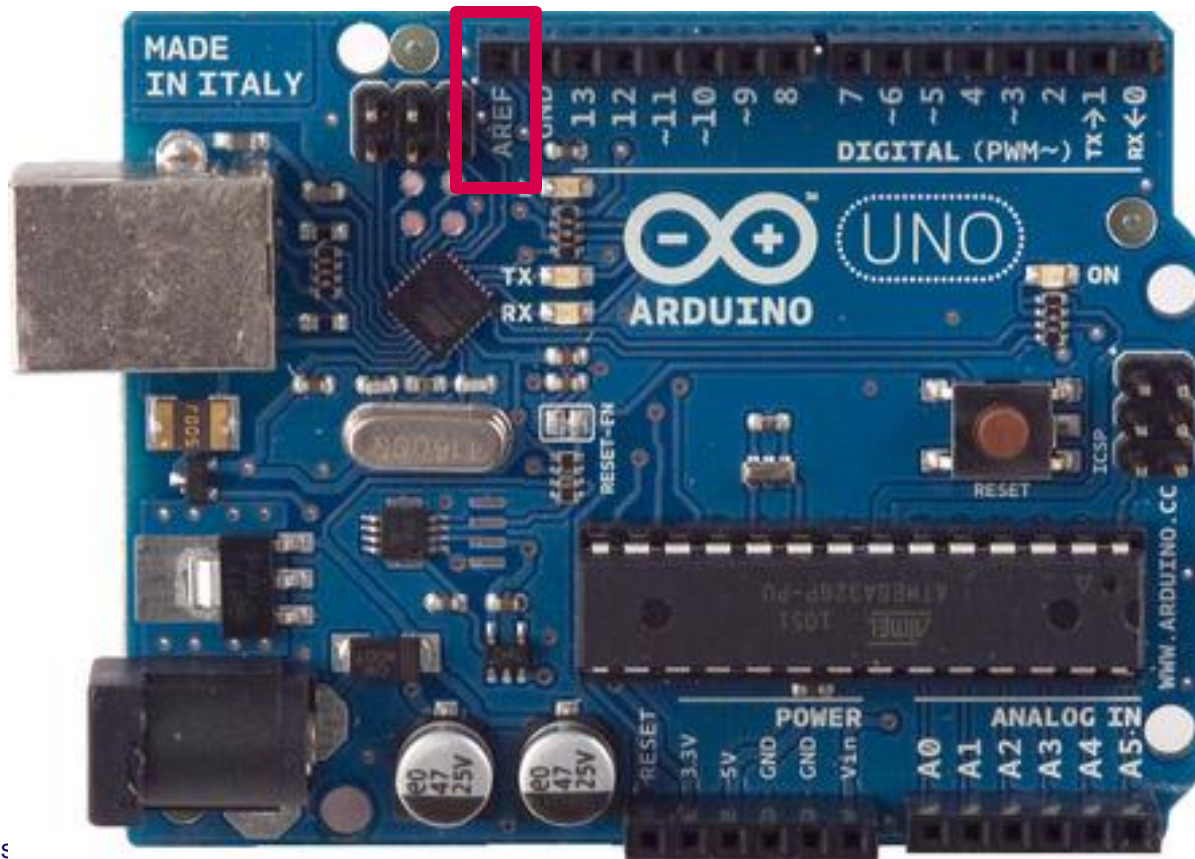
# UNO

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



# UNO

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



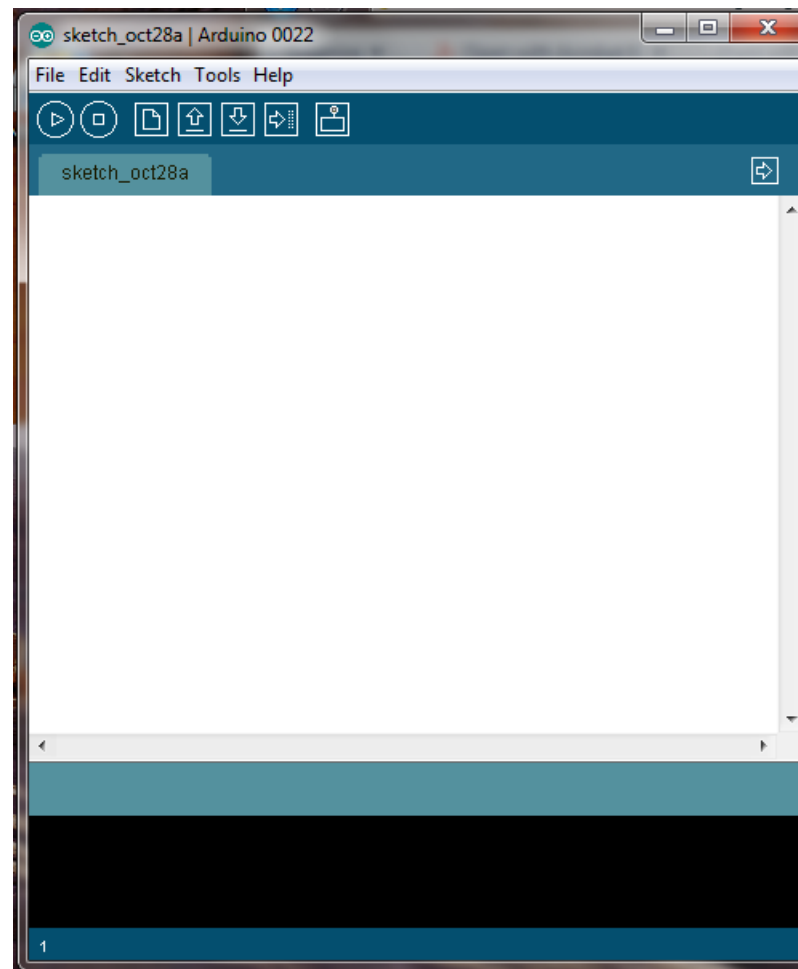
# UNO

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



# Software: IDE

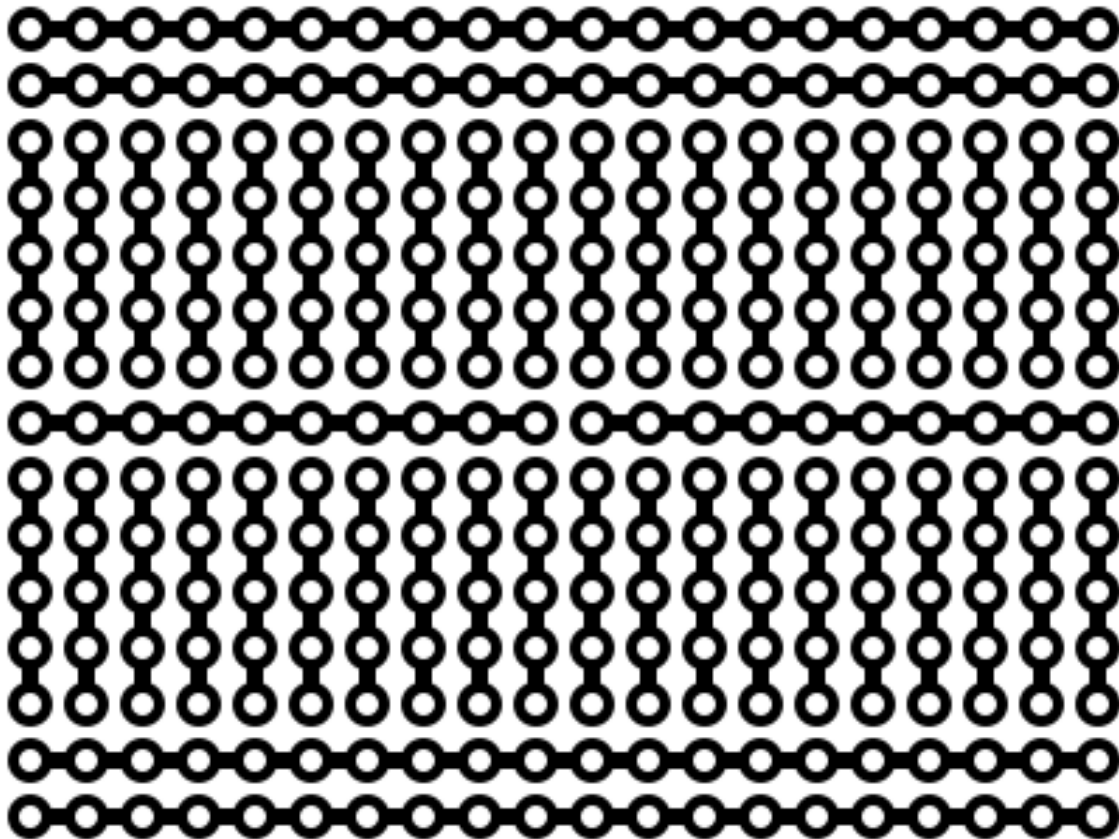
- <http://arduino.cc/en/Main/Software>





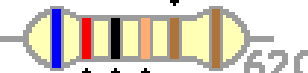
# Driver Installation and Port Identification

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

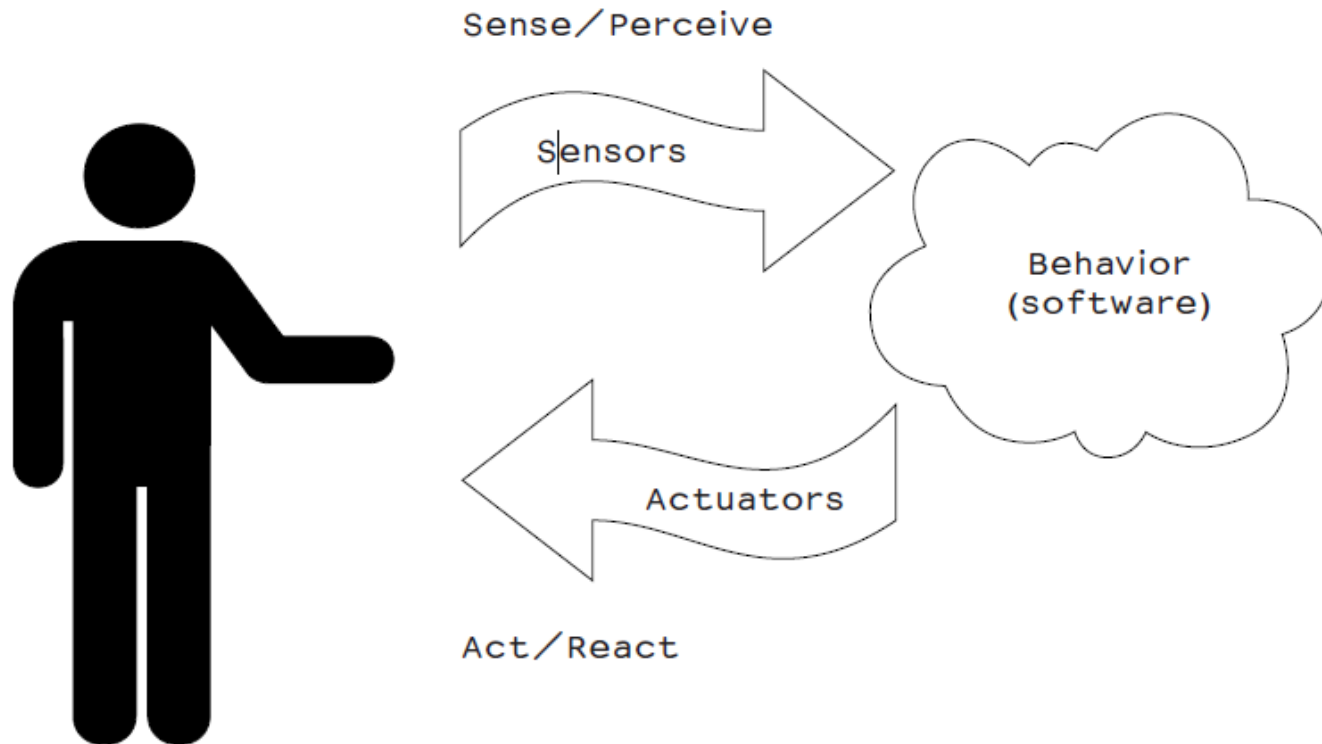
# Breadboard



# Color coding of the resistors

<p>           0 1 2 3 4 5 6 7 8 9            0 <b>Black</b>            1 <b>Brown</b>            2 <b>Red</b>            3 <b>Orange</b>            4 <b>Yellow</b>            5 <b>Green</b>            6 <b>Blue</b>            7 <b>Purple</b>            8 <b>Grey</b>            9 <b>White</b>            ±1% <b>Brown</b>            ±2% <b>Red</b>            ±5% <b>Gold</b>            ±10% <b>Silver</b> </p>	<p>           ±1%            ±2%            ±5%            ±10%         </p>  <p>27K EXAMPLE</p> <p>           0 ×1            1 1 ×10            2 2 ×100            3 3 ×1000            4 4 ×10000            5 5 ×100000            6 6 ×1000000            7 7 ÷10            8 8 ÷100            9 9         </p>	<p>           ±1%            ±2%            ±5%            ±10%         </p>  <p>15K EXAMPLE</p> <p>           0 0 ×1            1 1 1 ×10            2 2 2 ×100            3 3 3 ×1000            4 4 4 ×10000            5 5 5 ÷10            6 6 6 ÷100            7 7 7            8 8 8            9 9 9         </p>	<p>           ±1%            ±2%            ±5%            ±10%         </p>  <p>620K EXAMPLE</p> <p>           100 50            25 15            10 5            1         </p> <p>           0 0 ×1            1 1 1 ×10            2 2 2 ×100            3 3 3 ×1000            4 4 4 ×10000            5 5 5 ÷10            6 6 6 ÷100            7 7 7            8 8 8            9 9 9         </p>
<p><b>Color Codes</b></p>	<p><b>4 Band Resistors</b></p>	<p><b>5 Band Resistors</b></p>	<p><b>6 Band Resistors</b></p>

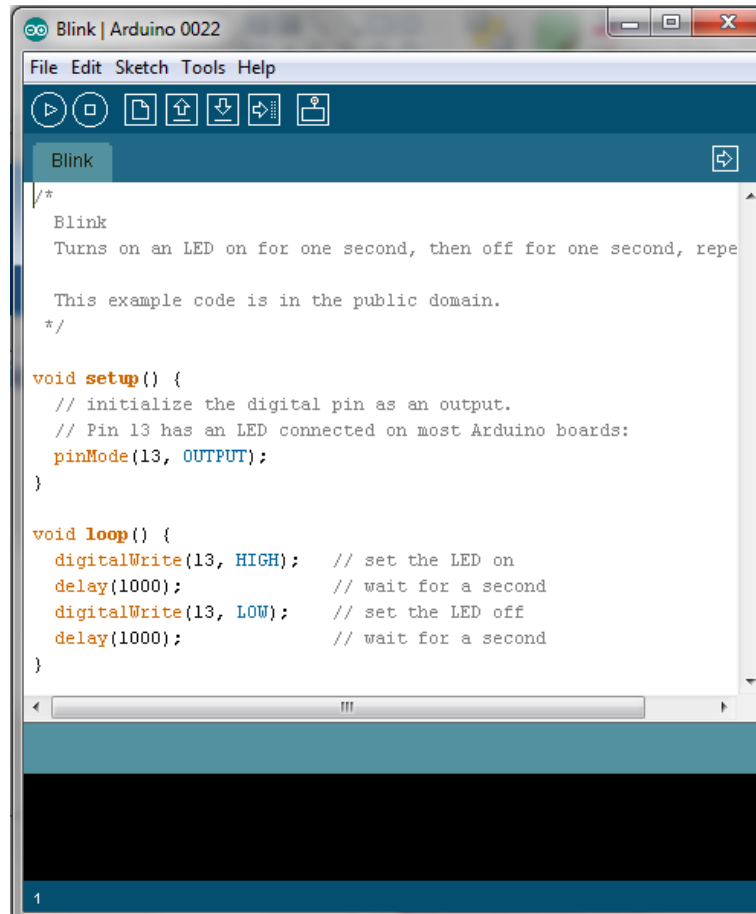
# Really getting started





# Blinking an LED

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

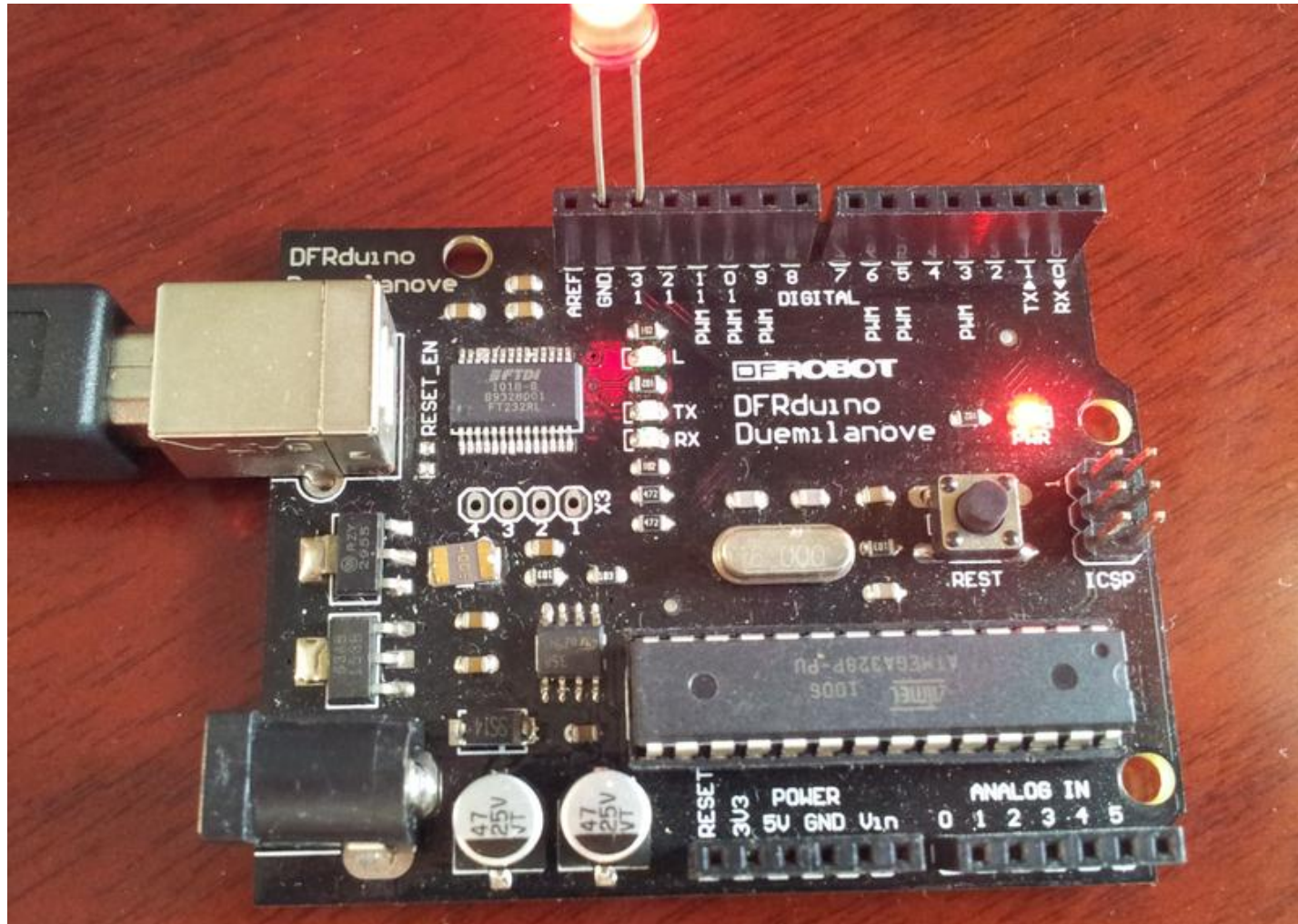
A screenshot of the Arduino IDE interface. The window title is "Blink | Arduino 0022". The menu bar includes "File", "Edit", "Sketch", "Tools", and "Help". Below the menu bar is a toolbar with icons for running, stopping, saving, and other functions. The main text area shows the code for the "Blink" sketch. The code includes a multi-line comment describing the sketch, a setup function that initializes pin 13 as an output, and a loop function that toggles the LED on and off with 1000ms delays.

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

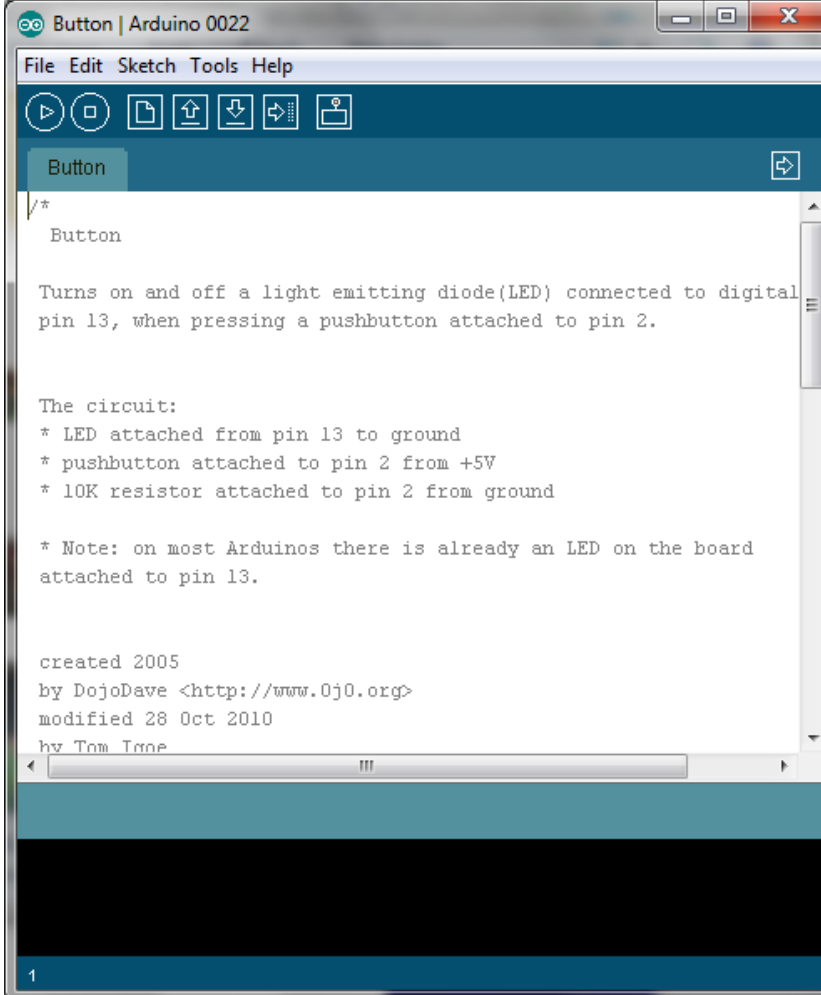
```
#define LED 13

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 | Arduino 0022
File Edit Sketch Tools Help
Button
/*
  Button

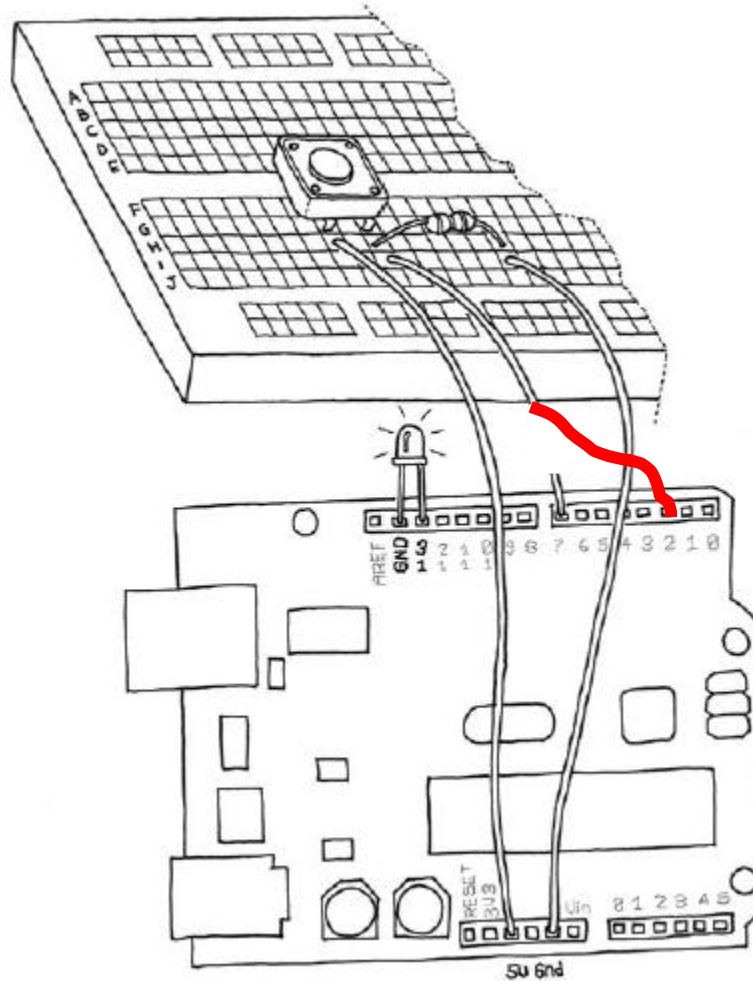
  Turns on and off a light emitting diode(LED) connected to digital
  pin 13, when pressing a pushbutton attached to pin 2.

  The circuit:
  * LED attached from pin 13 to ground
  * pushbutton attached to pin 2 from +5V
  * 10K resistor attached to pin 2 from ground

  * Note: on most Arduinos there is already an LED on the board
  attached to pin 13.

  created 2005
  by DojoDave <http://www.0j0.org>
  modified 28 Oct 2010
  by Tom Igoe
```

# 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);
}
```

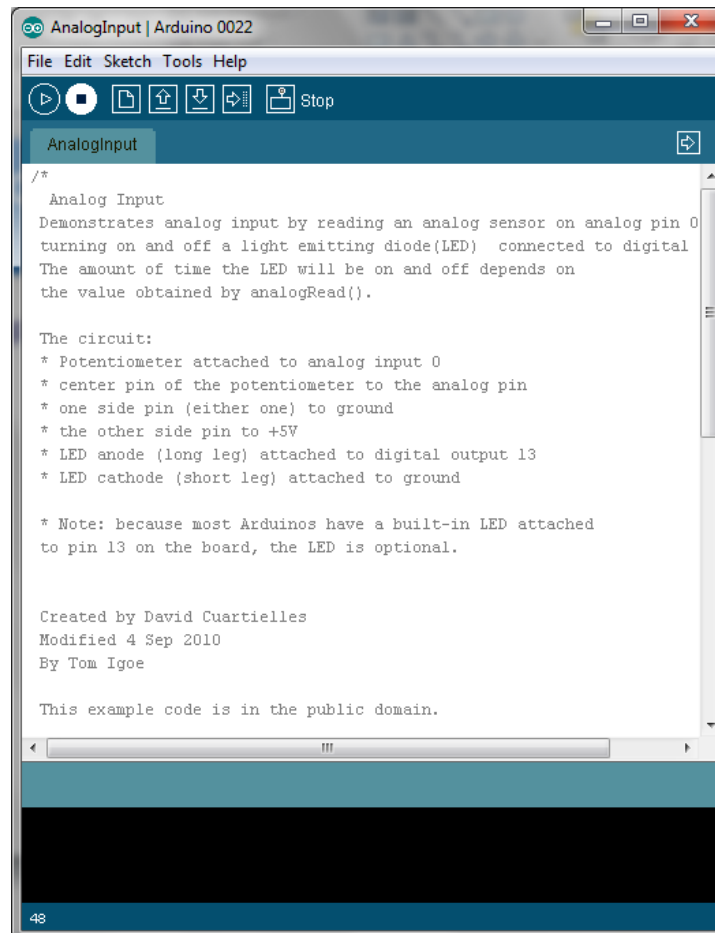
# Button to control the LED

```
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**



```
/*
  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

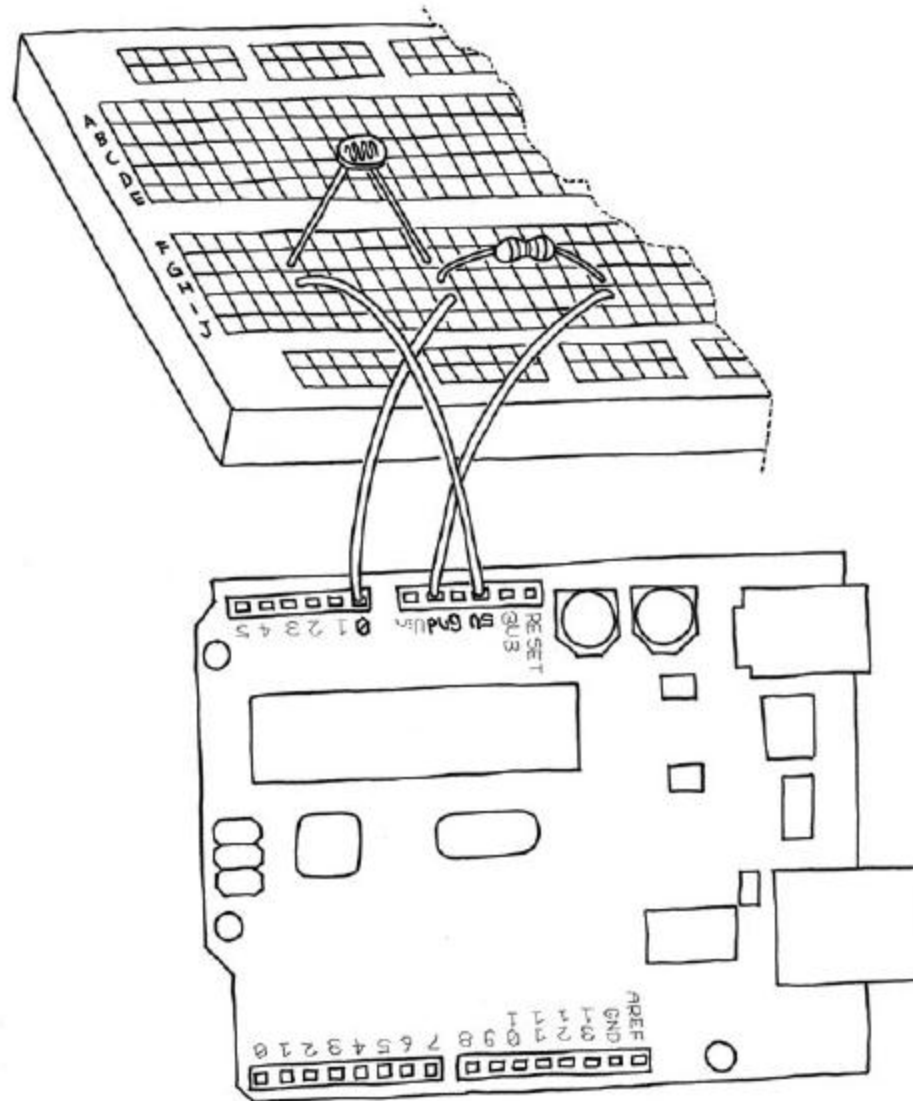
  * Note: because most Arduinos have a built-in LED attached
  to pin 13 on the board, the LED is optional.

  Created by David Cuartielles
  Modified 4 Sep 2010
  By Tom Igoe

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



# Analog I/O



# 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

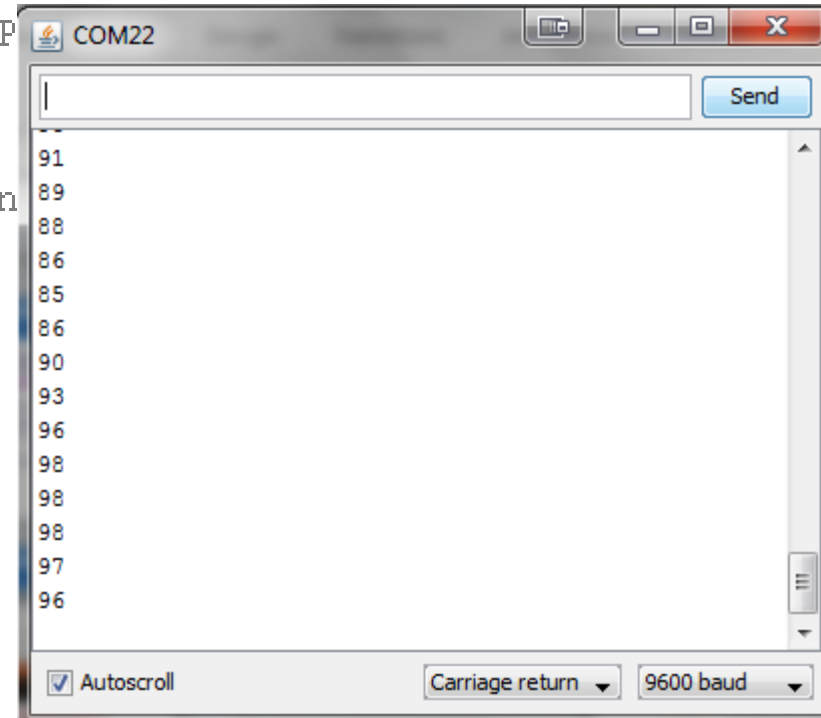
```
// Example 11-06 from "Getting Started with Processing"  
// by Reas & Fry. O'Reilly / Make 2010  
  
// Note: This is code for an Arduino board, not Processing  
  
int sensorPin = 0; // Select input pin  
int val = 0;  
  
void setup() {  
  Serial.begin(9600);  
}  
  
void loop() {  
  val = 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.

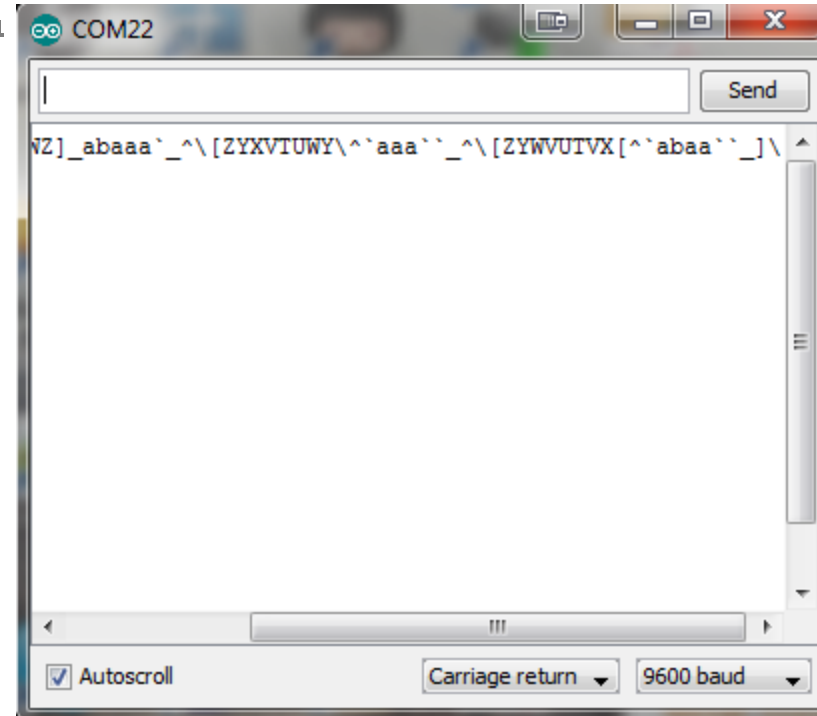
# Serial Communication

```
// 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);
}

Department }
```

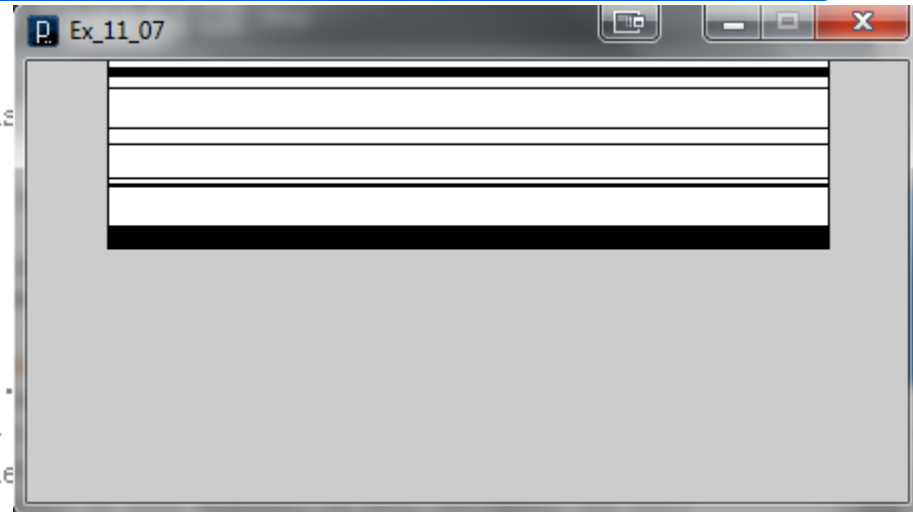
# Serial Communication

```
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

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 the
  // line by deleting the // before it. Run the sketch
  // 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

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

## Download

Processing Library: [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).

Processing library for arduinoMega: [processing-arduinomega.zip](#) (Updated 12 Apr. 2010)

# Serial Communication

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

## 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/>

# Serial Communication

- **Introducing Firmata**

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

## 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);
}

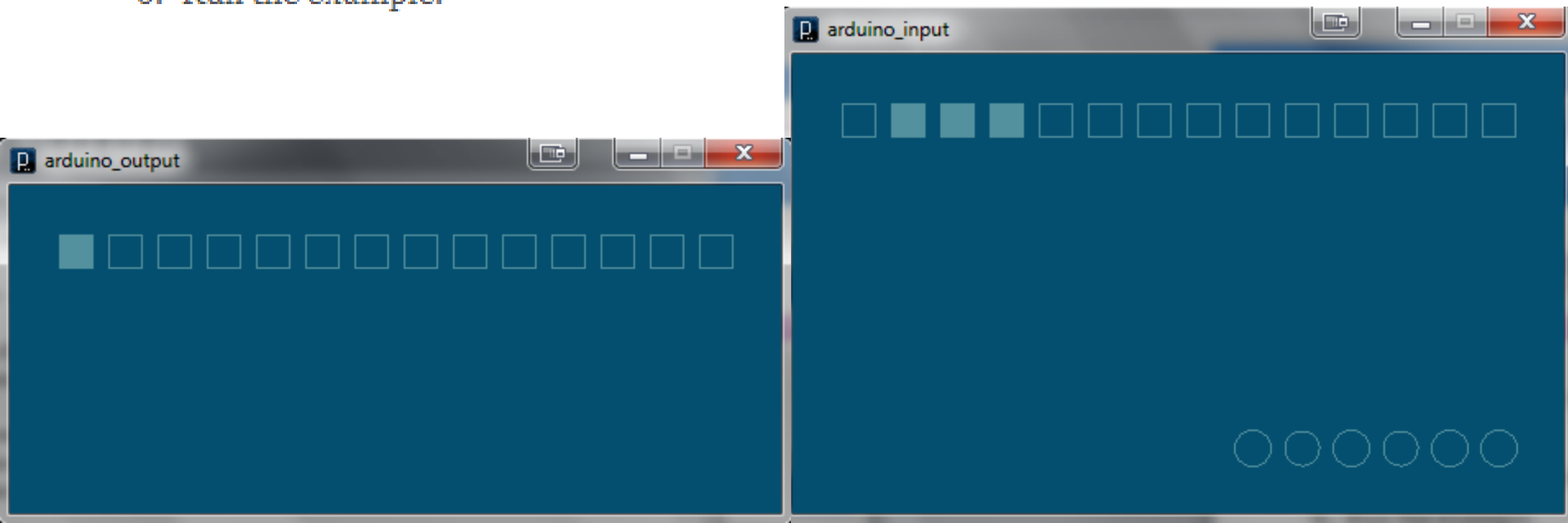
void draw()
{
  arduino.digitalWrite(ledPin, Arduino.HIGH);
  delay(1000);
  arduino.digitalWrite(ledPin, Arduino.LOW);
  delay(1000);
}
```

# Serial Communication

- **Introducing Firmata**

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

4. In Processing, open one of the examples that comes with with the Arduino library.
5. Edit the example code to select the correct serial port.
6. Run the example.







That was Arduino.

**TU** / **e**

Technische Universiteit  
**Eindhoven**  
University of Technology

**Where innovation starts**