

# Creative Electronics

Adding hardware – Arduino

20-5-2015



Technische Universiteit  
**Eindhoven**  
University of Technology

**Where innovation starts**

# Contents

- **Why Arduino**
- **Arduino Hardware**
- **Blink a LED**
- **Digital Input**
- **Analog Input**
- **Analog Output**
- **Communication**

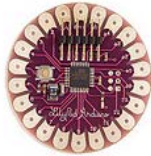
# Why Arduino?

- **Physical Computing**
  - interfaces to sensors / actuators
  - prototypes behavior
  - entry level for designers and artists.
- **Large community**
  - Blog, Forum, Examples

# 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

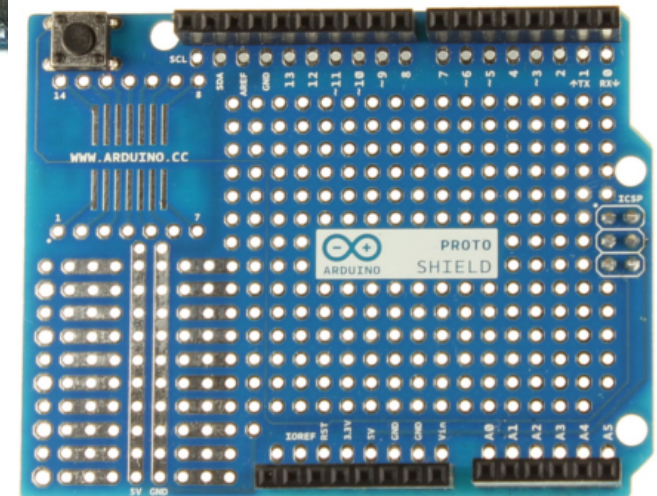
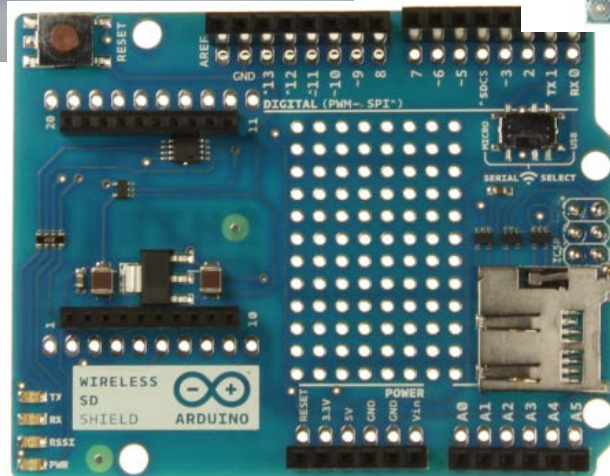
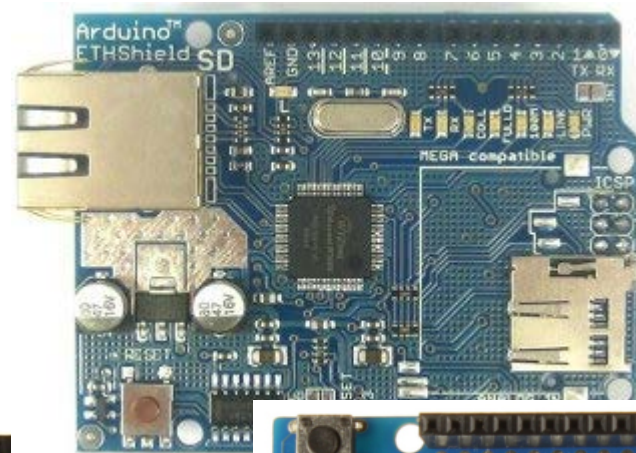
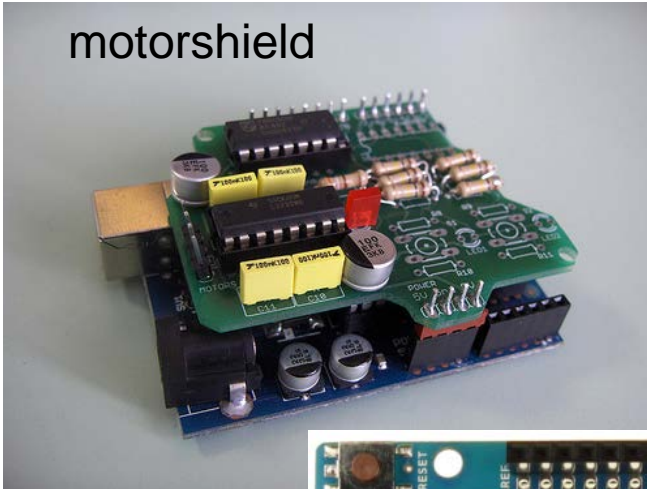




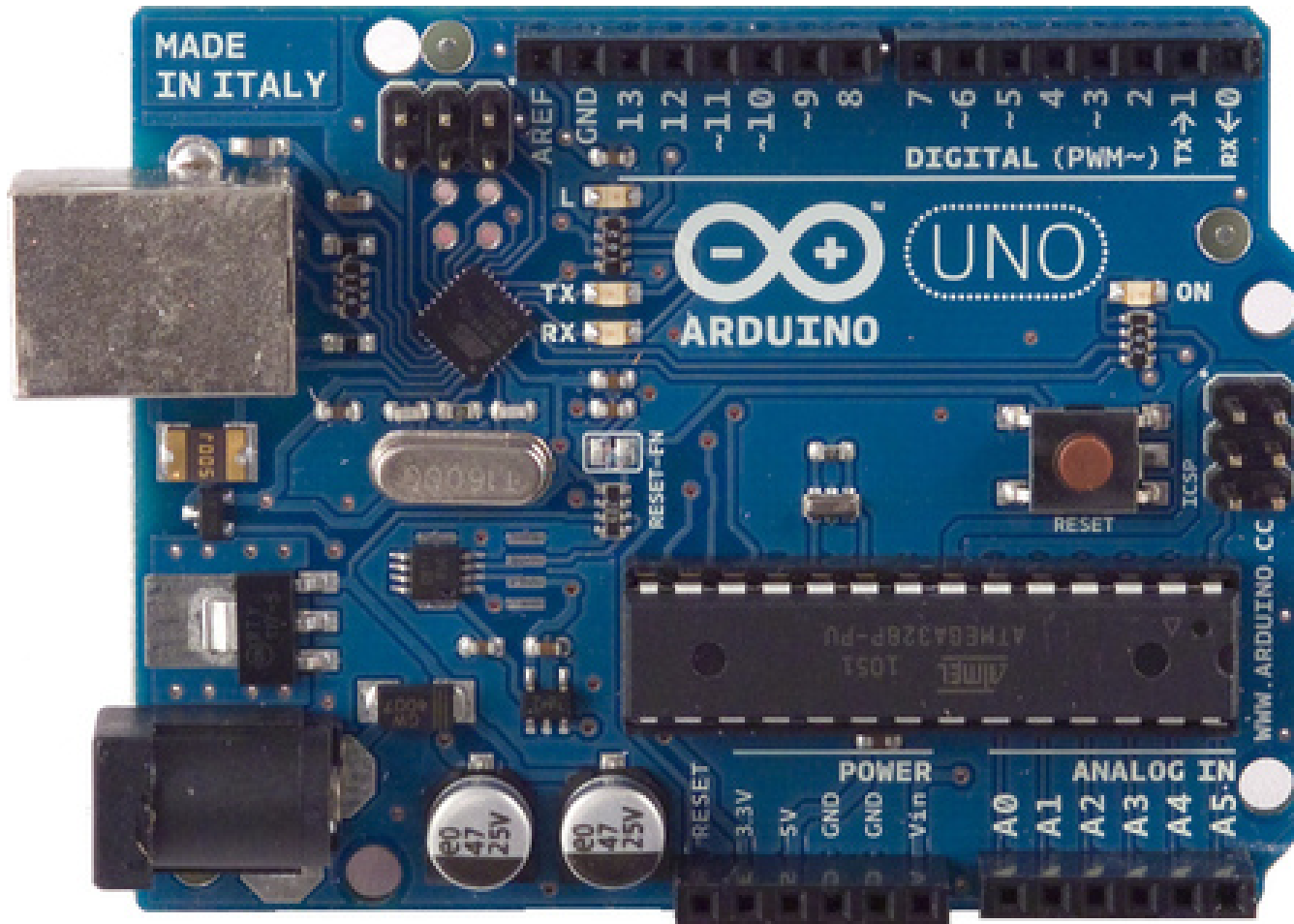
# Additional hardware: shields

- Add-on electronics modules for improving functionality

motorshield



# UNO



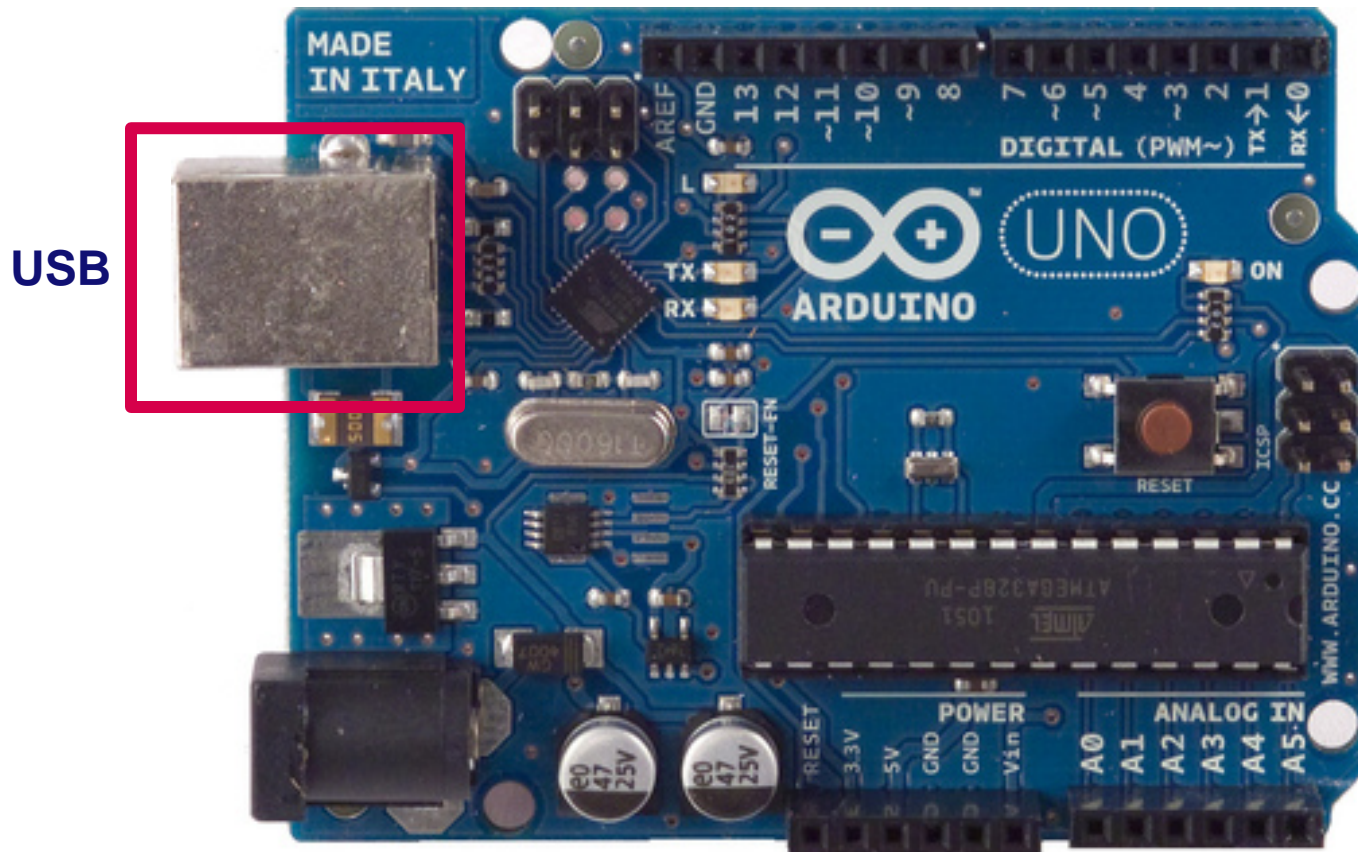
# UNO

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

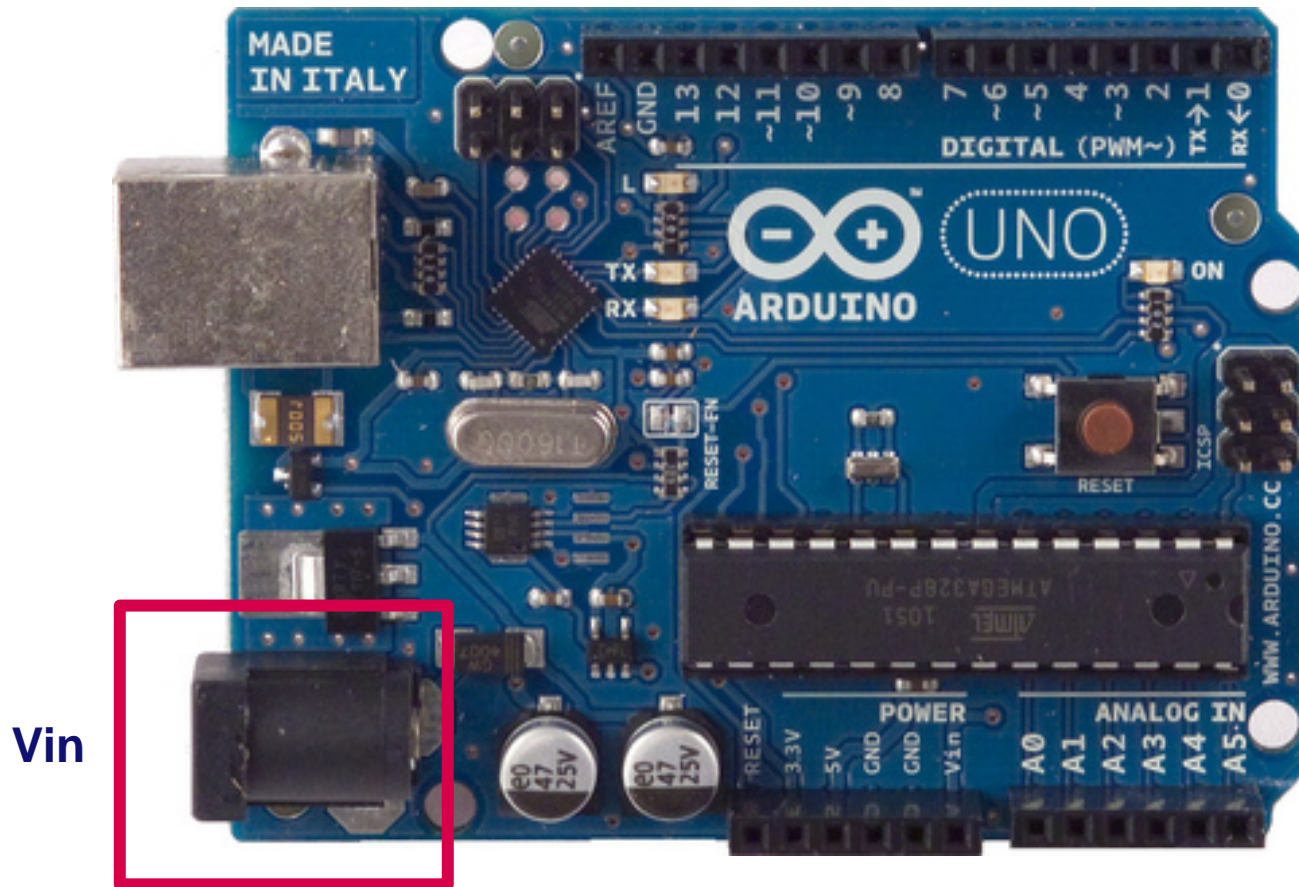
- Power: USB Power supply (5V)





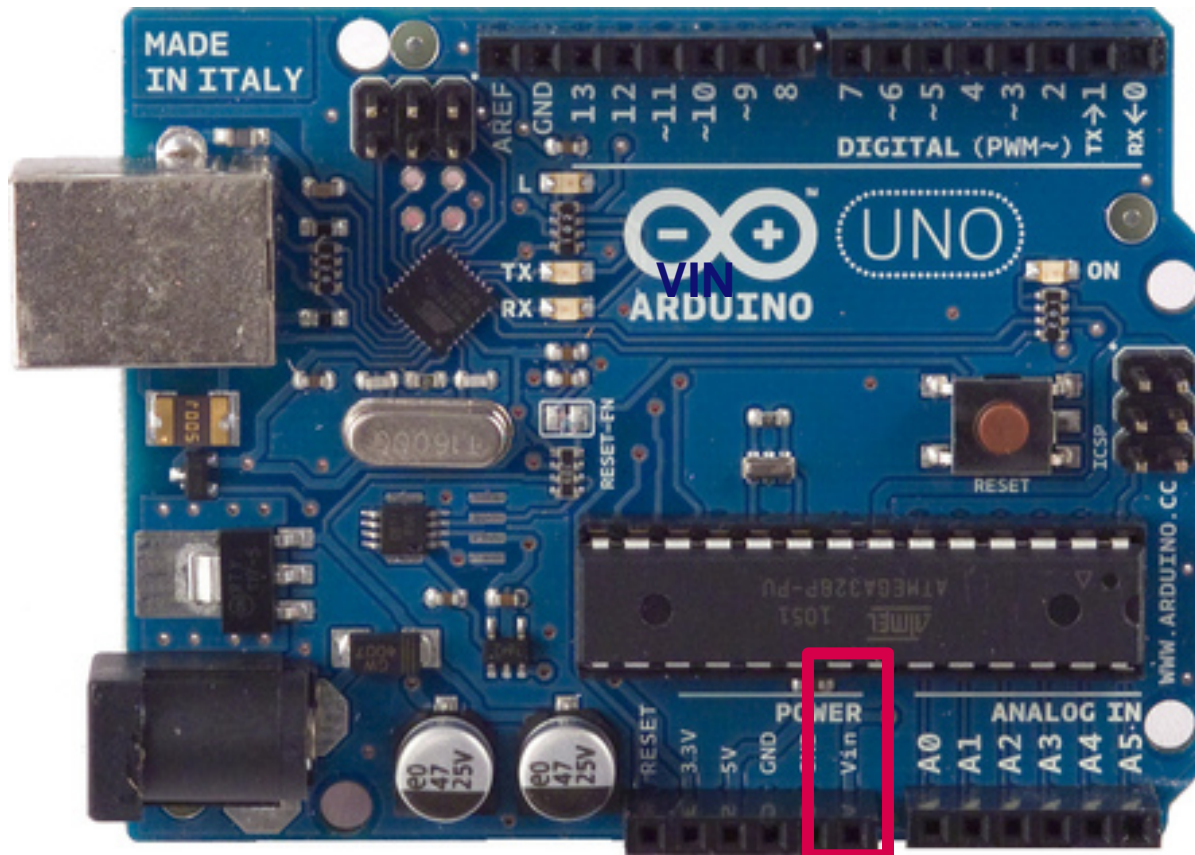
# UNO

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



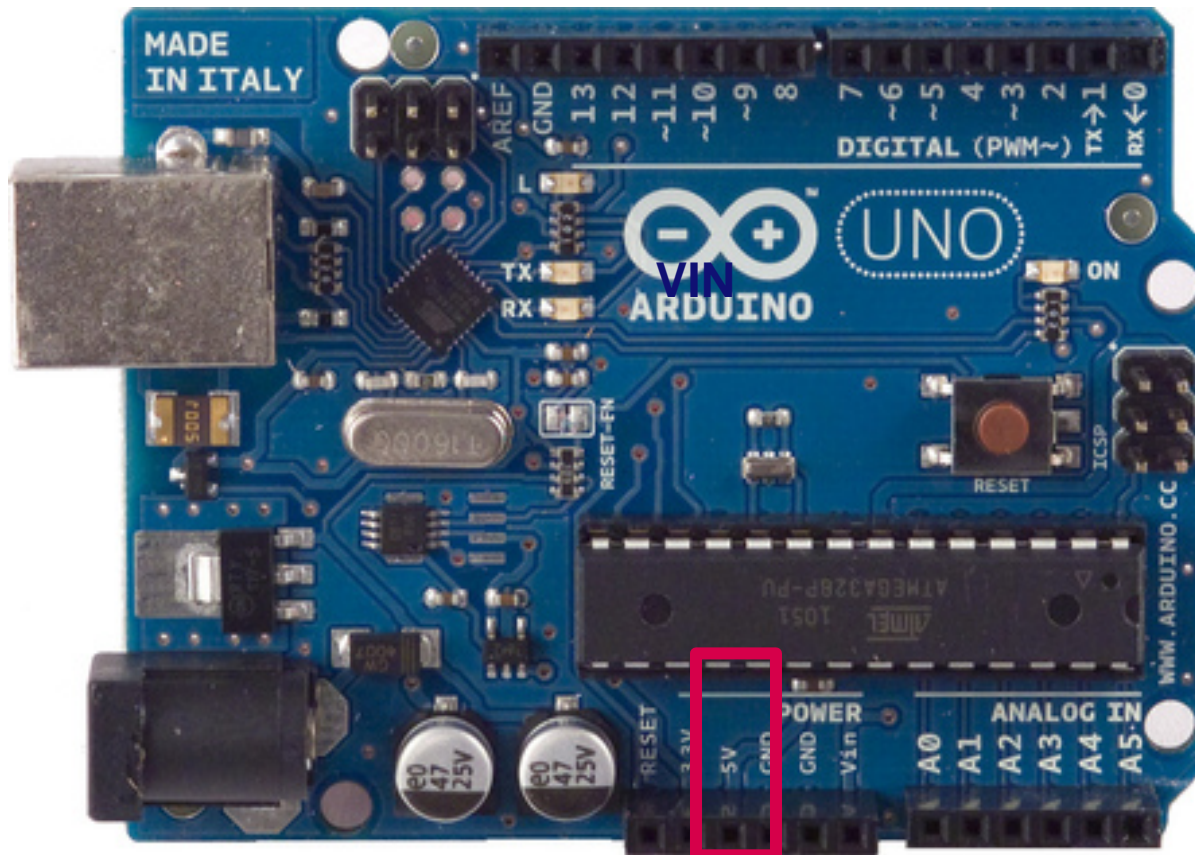
# UNO

- Power:  $V_{in}$ , depends on external source. (7-12V)



# UNO

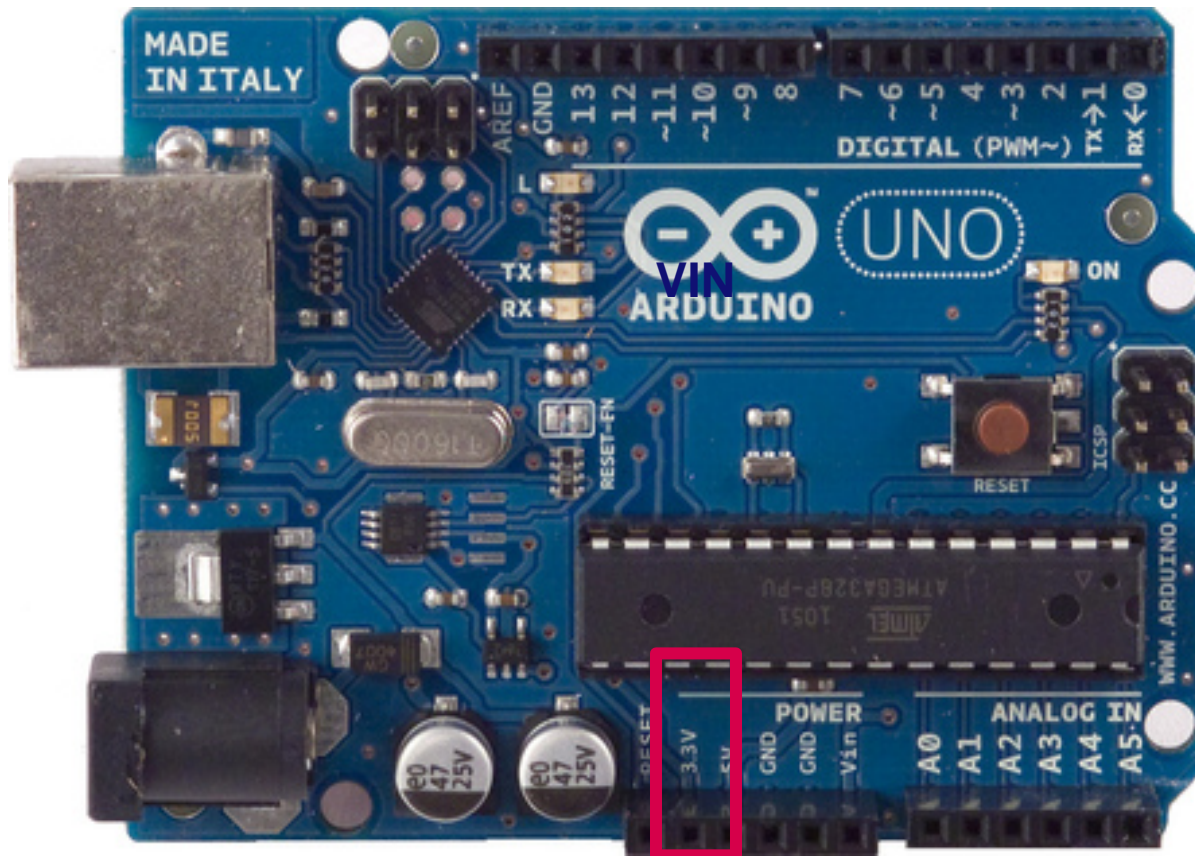
- Power: 5V supply





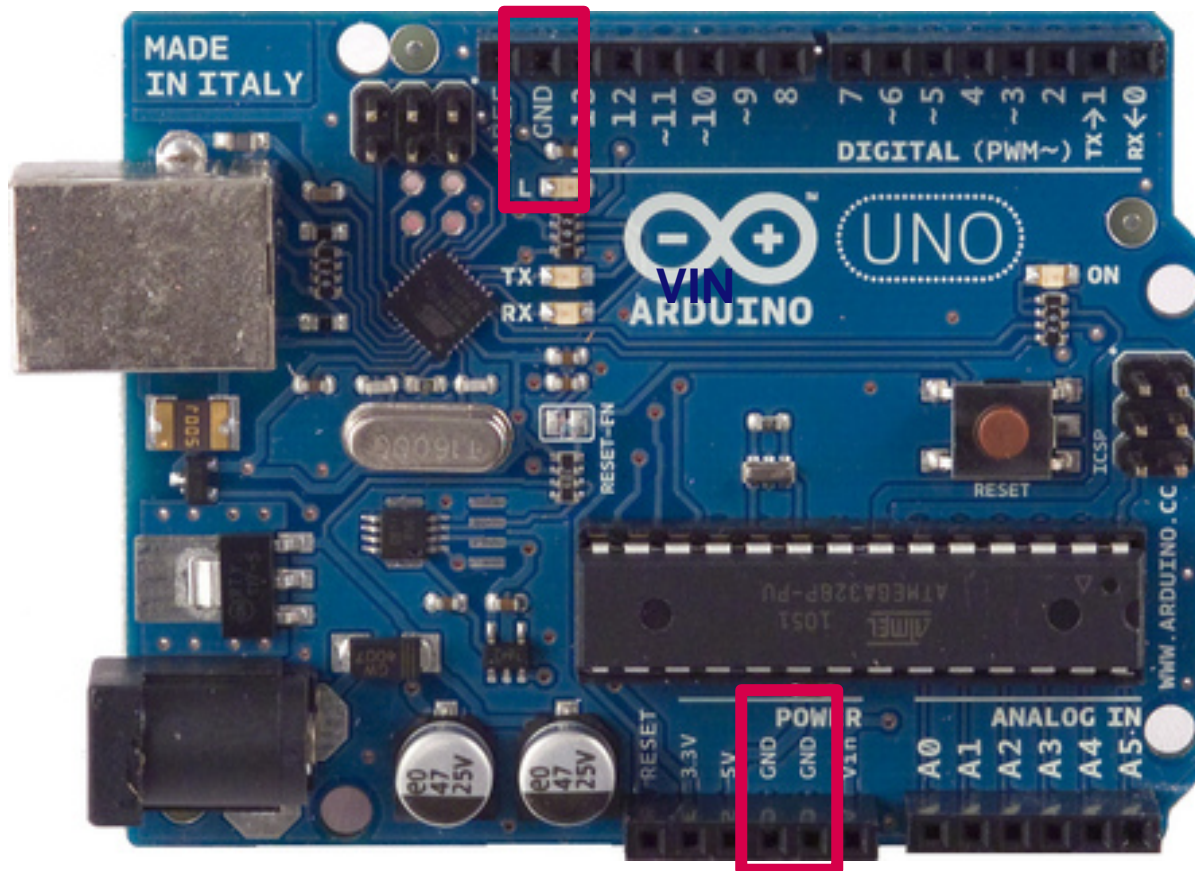
# UNO

- Power: 3.3V supply



# UNO

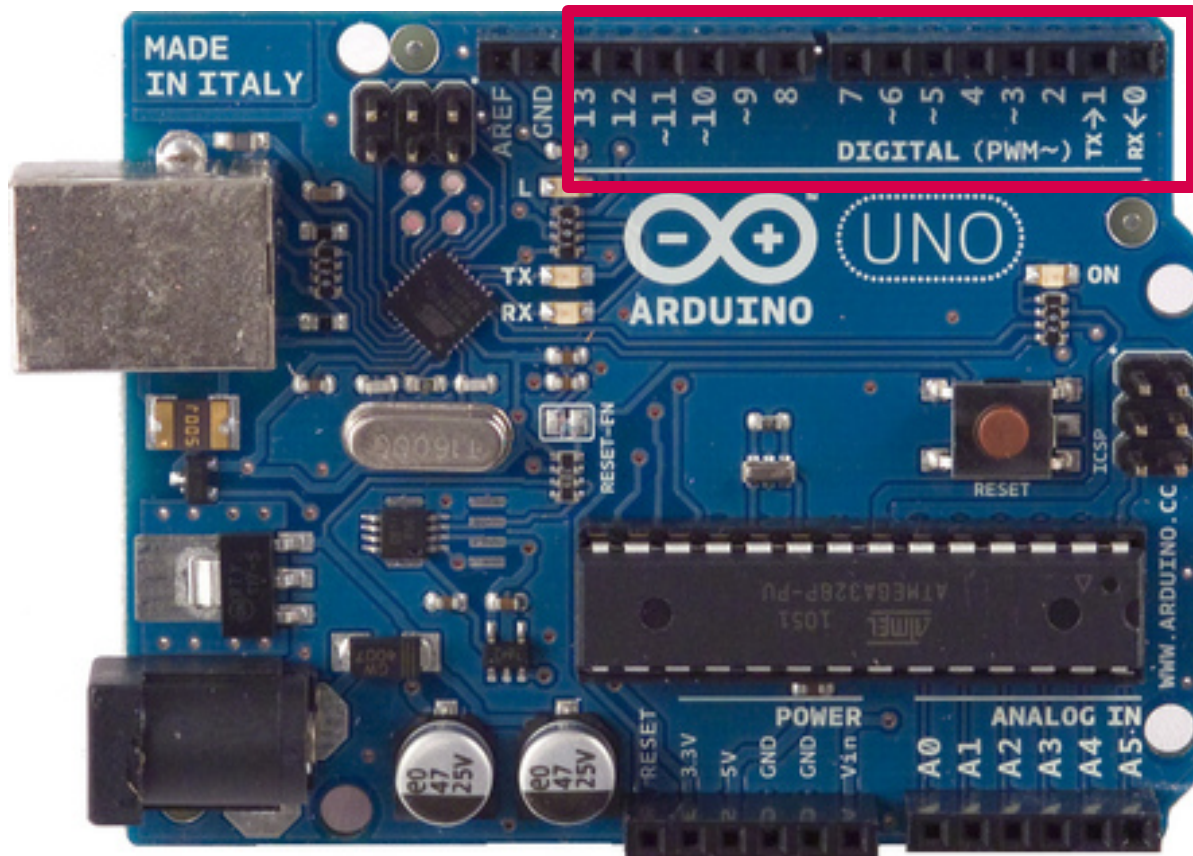
- Power: GND pins





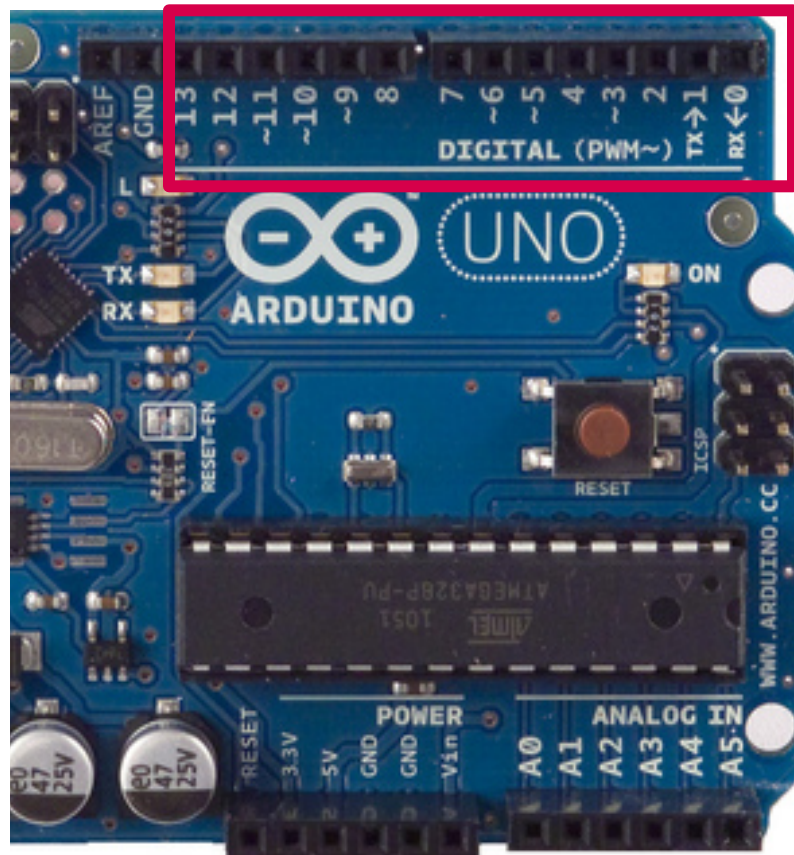
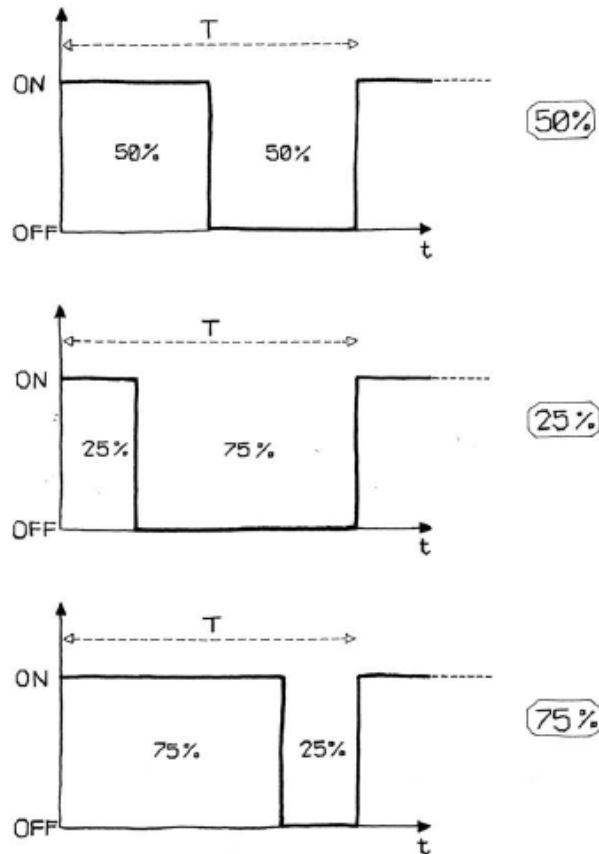
# UNO

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



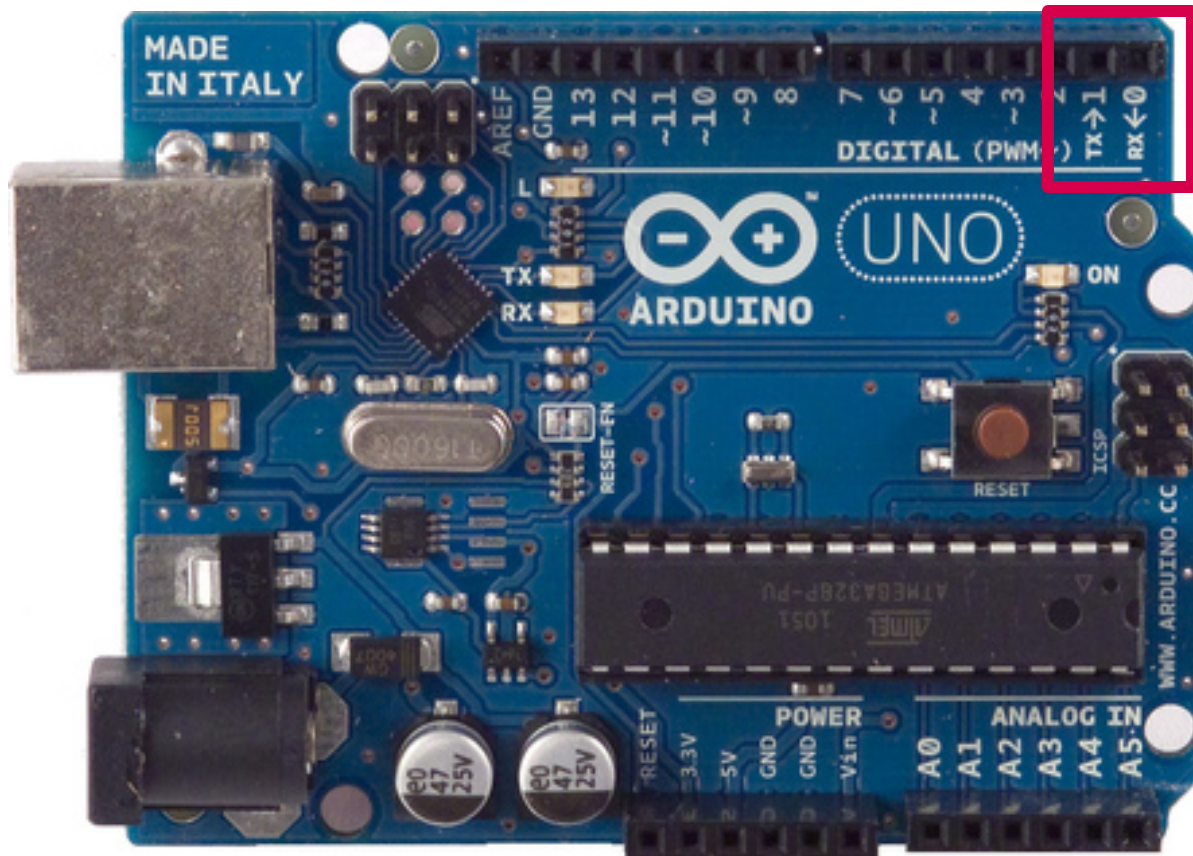
# UNO

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



# UNO

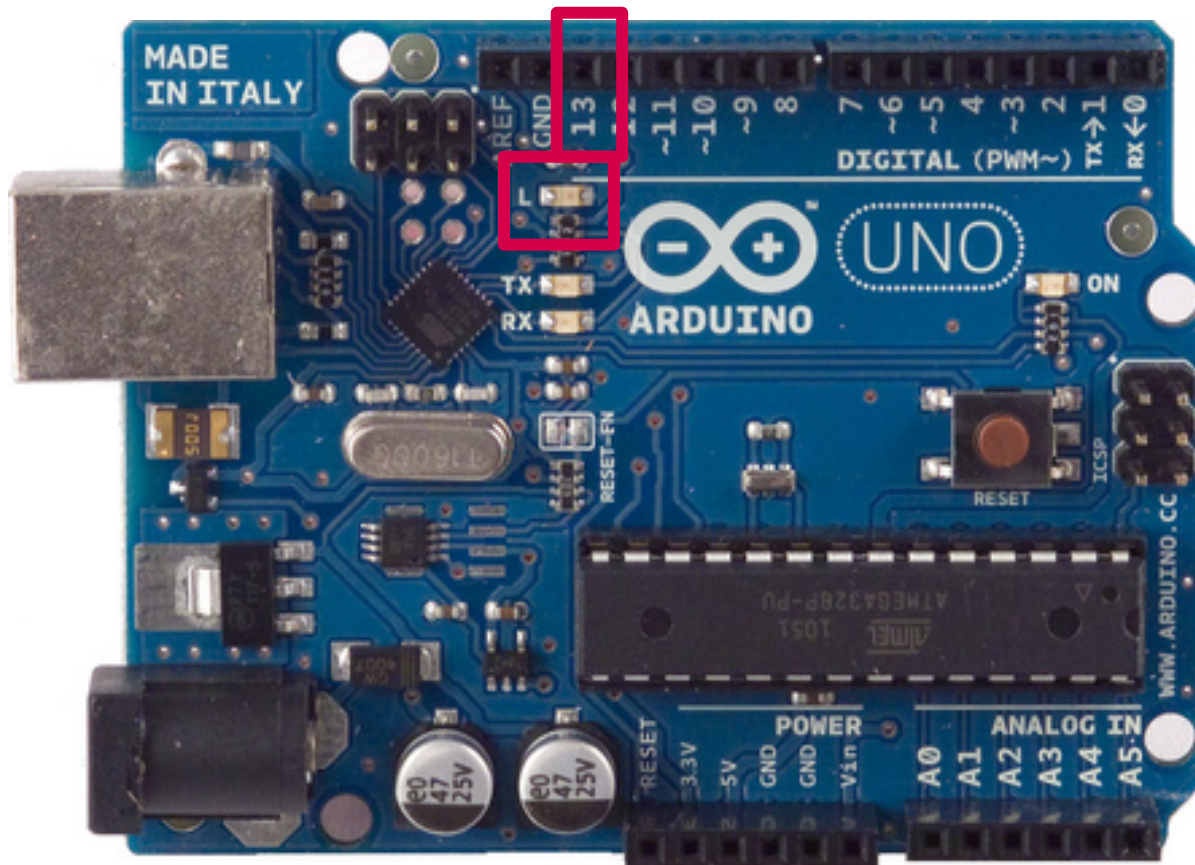
- Serial: 0 (RX) and 1 (TX)





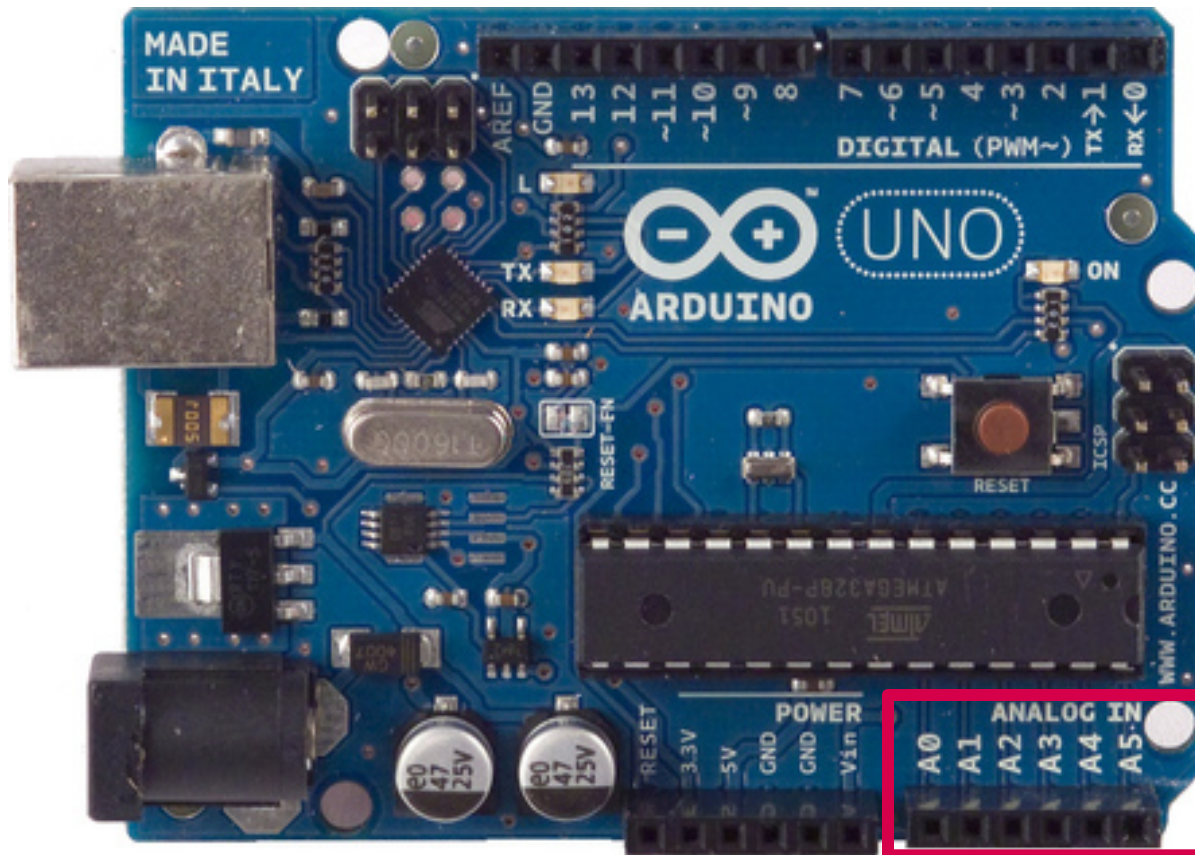
# UNO

- LED: Pin 13



# UNO

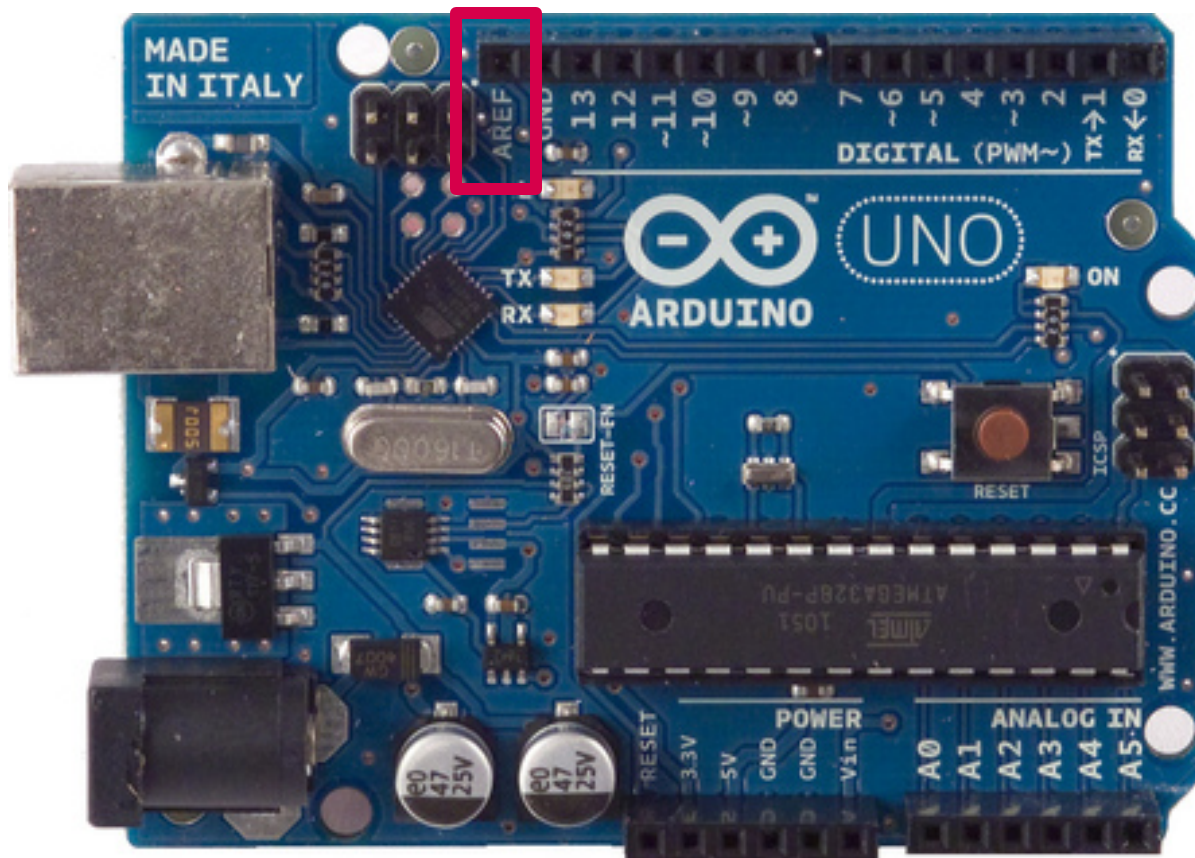
- 6 analog inputs, 10 bits resolution (1024 values)





# UNO

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

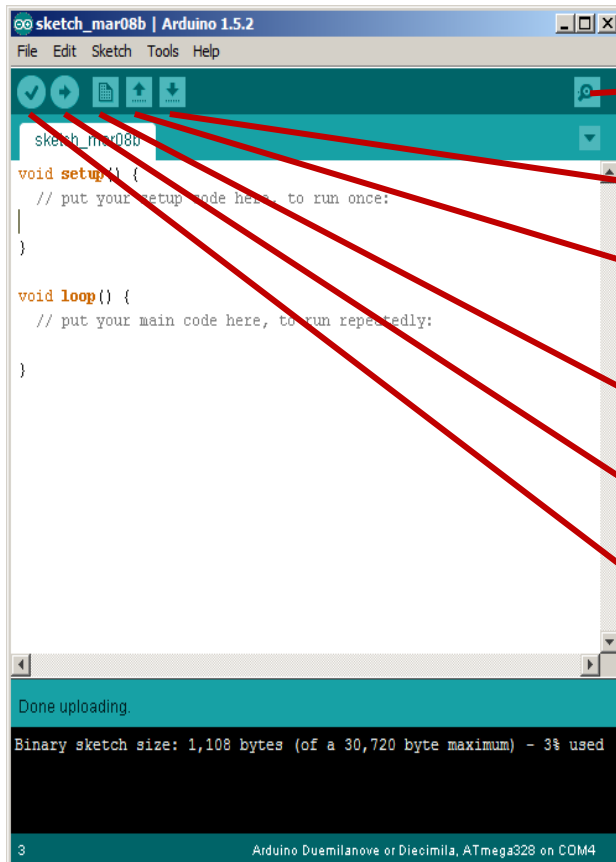


- **Reset. LOW to reset the microcontroller**



# Software: IDE

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



Serial Monitor

Save Sketch

Open Sketch  
(old: \*.pde, new:  
\*.ino)

New Sketch

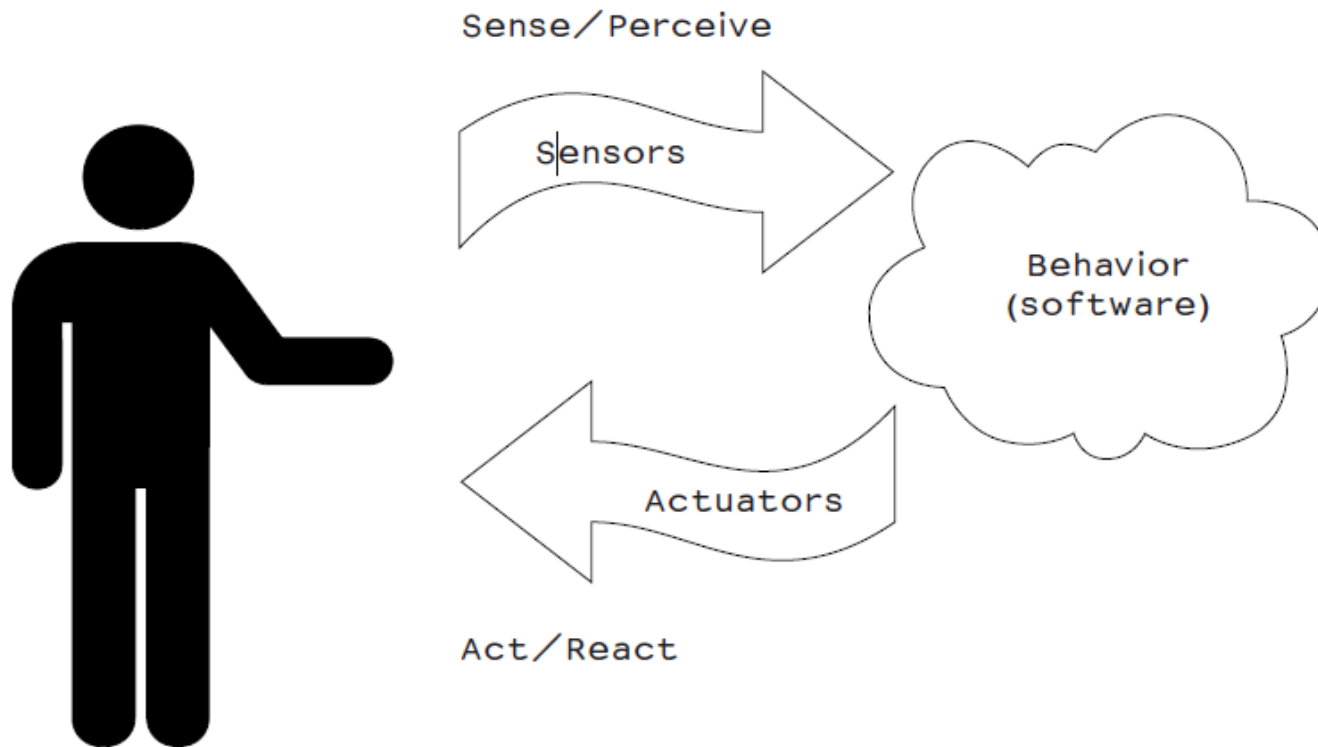
Compile and Upload

Verify (Compile)

# Driver Installation and Port Identification

- Refer to the instructions in
  - “Getting Started with Arduino” book, page 23-26
  - <http://arduino.cc/en/Guide/HomePage>
- Check the wiki
  - <http://wiki.id.tue.nl/creapro/CreativeProgrammingAssignment201504>

# Really getting started

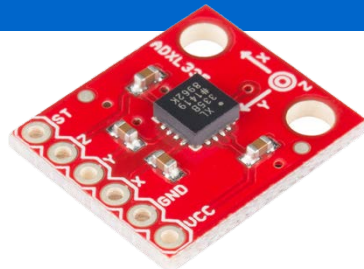




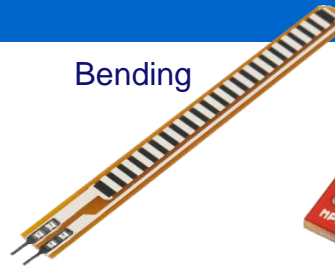
# Some common sensors



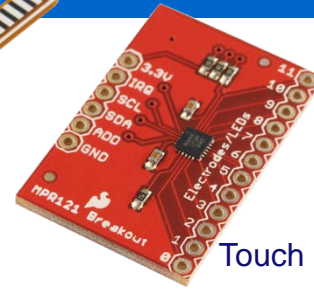
Position (GPS)



Acceleration



Bending



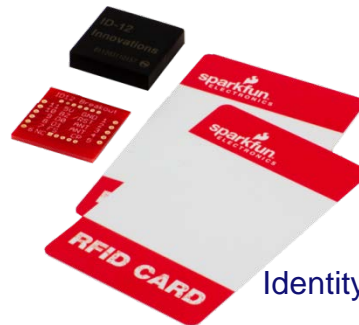
Touch



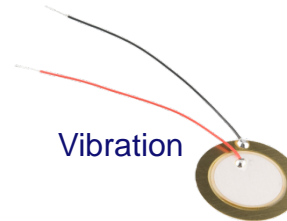
Distance



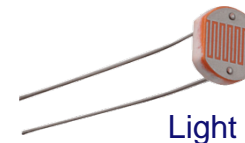
Force



Identity



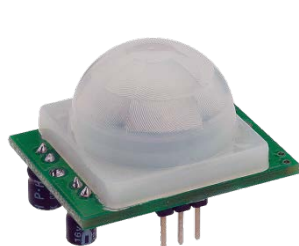
Vibration



Light



Sound



Movement



Temperature



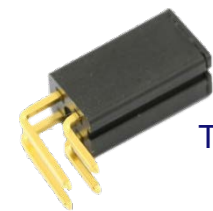
On-off status



Setting or Orientation



Rotation



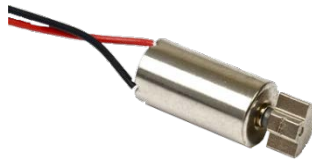
Tilt



# Some common actuators



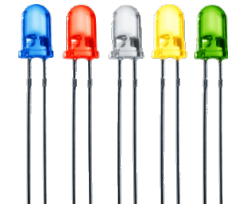
DC motor



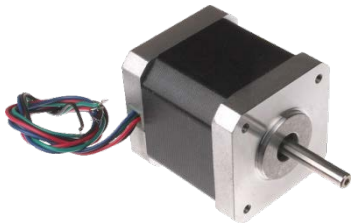
Vibration motor



Display



Led



Stepper motor



Solenoid



Buzzer



Heating pad



Servo motor

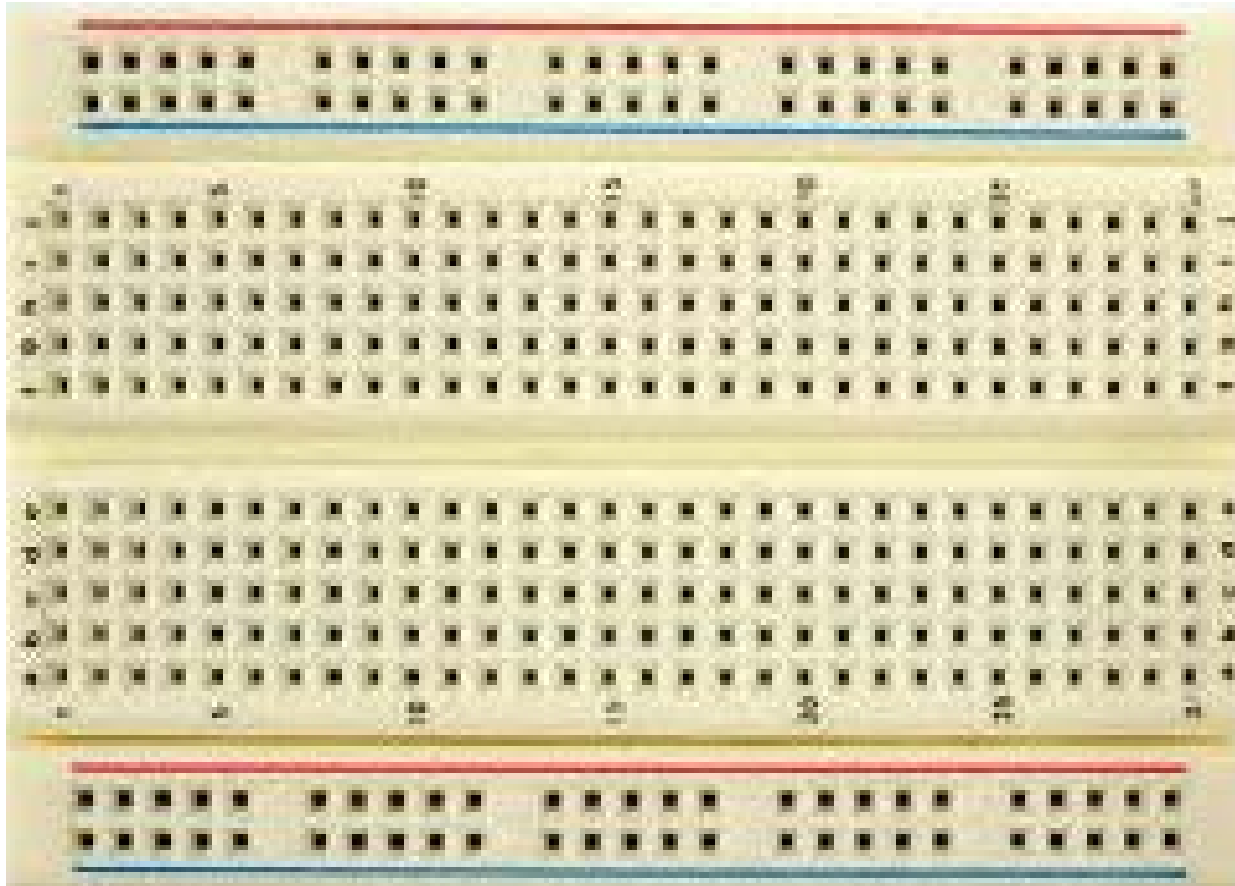


Fan



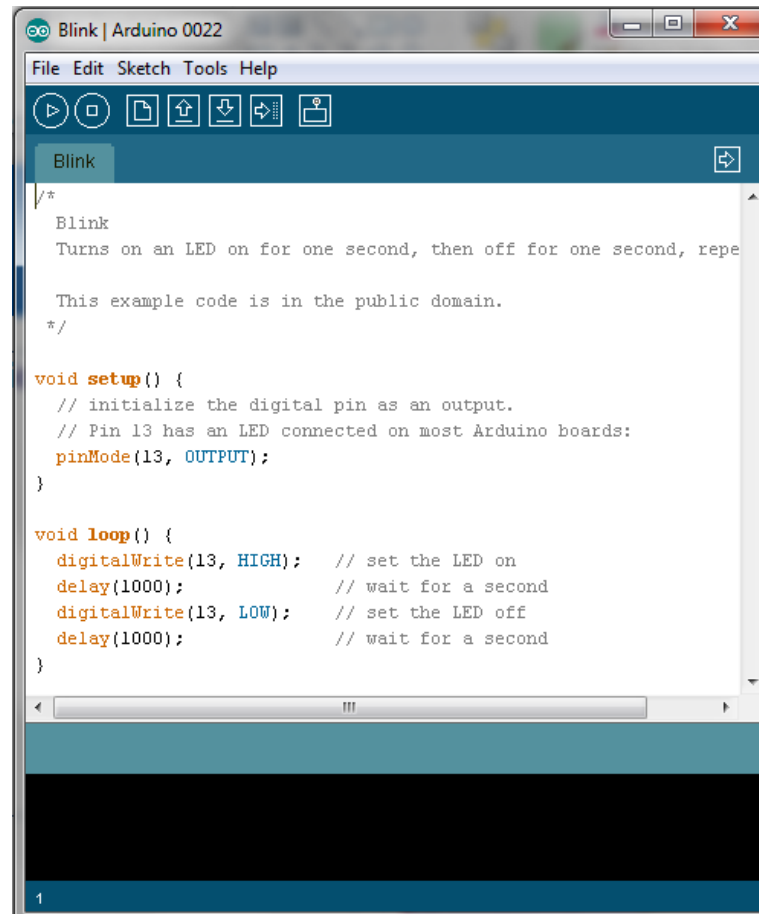
Speaker

# Breadboard



# Blinking a LED

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



The screenshot shows the Arduino IDE interface with the 'Blink' example code loaded. The window title is 'Blink | Arduino 0022'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar contains icons for running, stopping, saving, and other sketch functions. The code editor displays the following code:

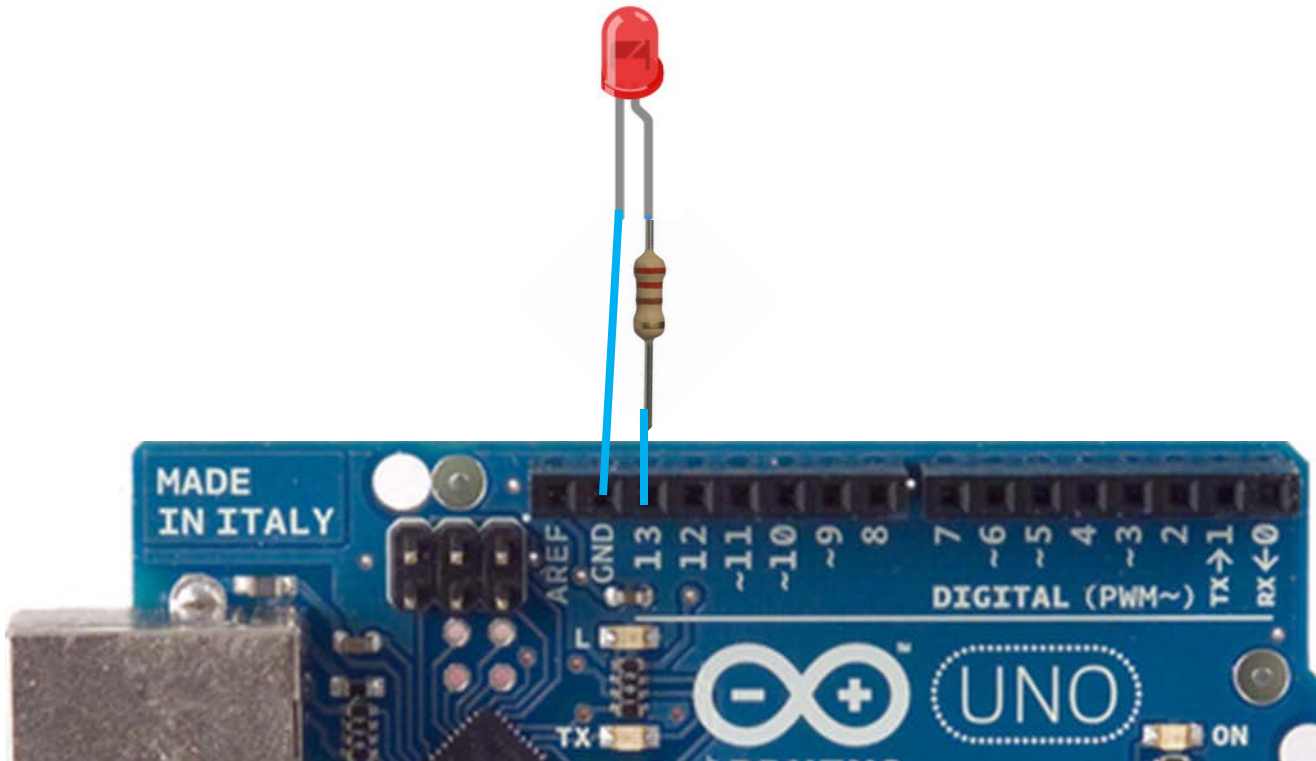
```
/*
 * Blink
 * Turns on an LED on for one second, then off for one second, repeatedly.
 *
 * This example code is in the public domain.
 */

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
}
```

The bottom of the window shows a dark area representing the serial monitor or a terminal window.

# Blinking a LED





# Blinking a LED

- **#define LED 13**

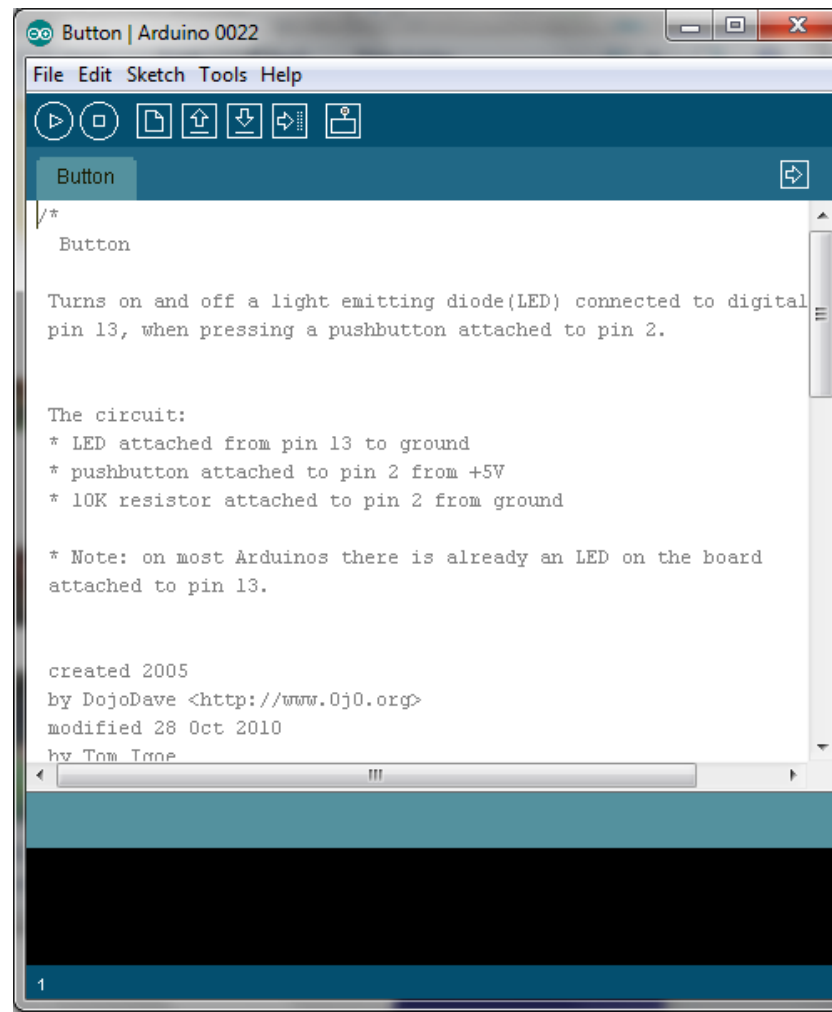
```
#define LED 13

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

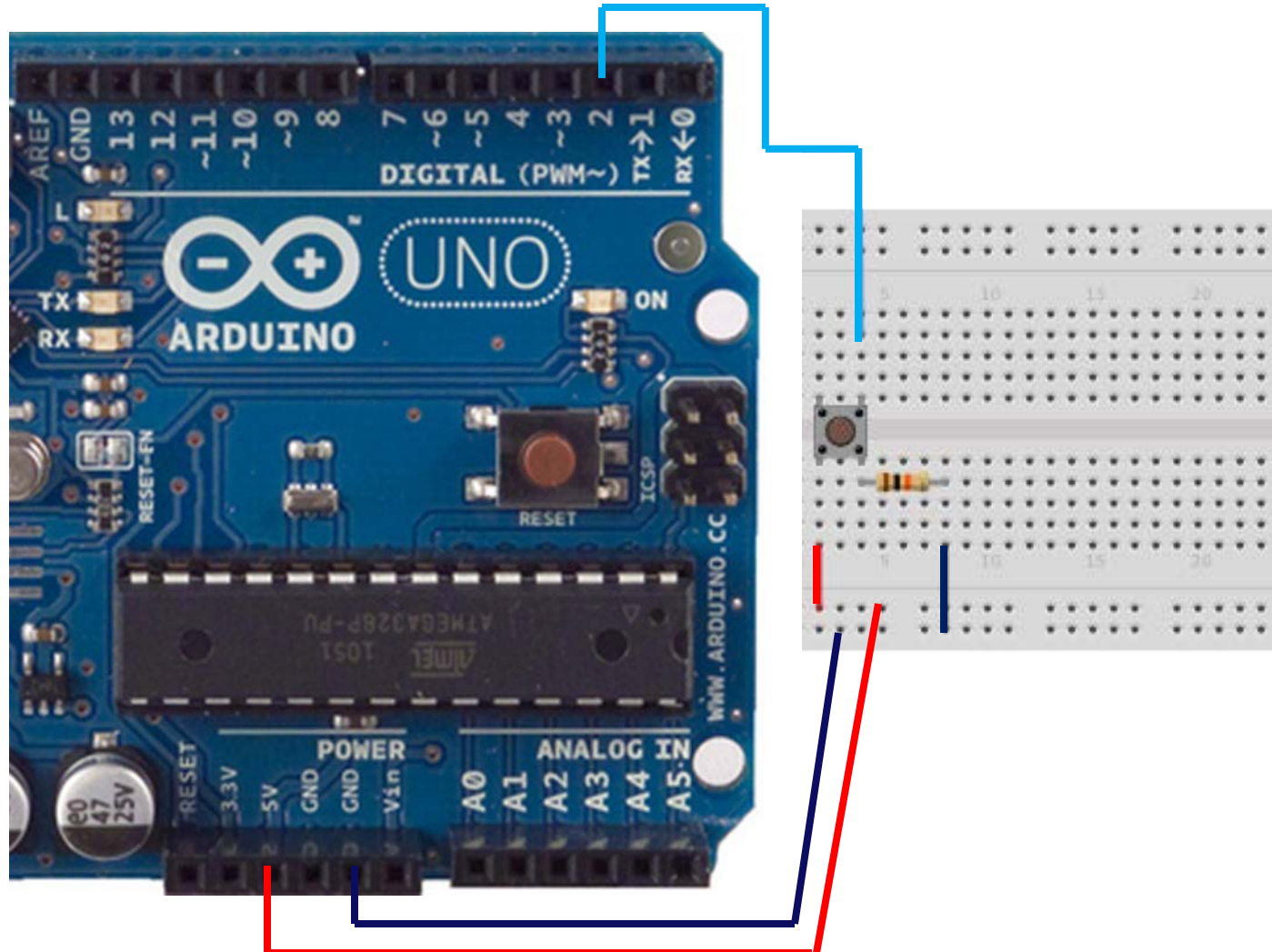
void loop() {
    digitalWrite(LED, HIGH);    // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(LED, LOW);     // set the LED off
    delay(1000);                // wait for a second
}
```

# Button to control a LED

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



# Button to control a LED



# Button to control a 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);
}
```

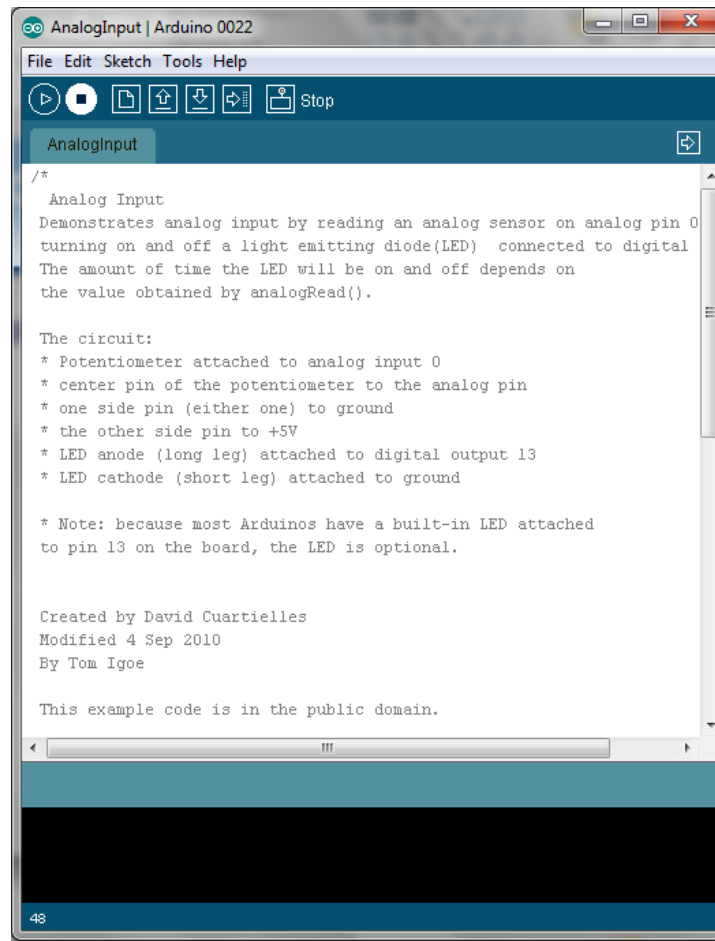
# Button to control a LED

```
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);  
  }  
}
```

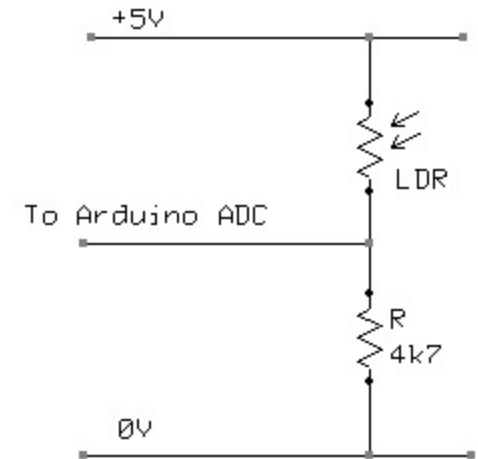
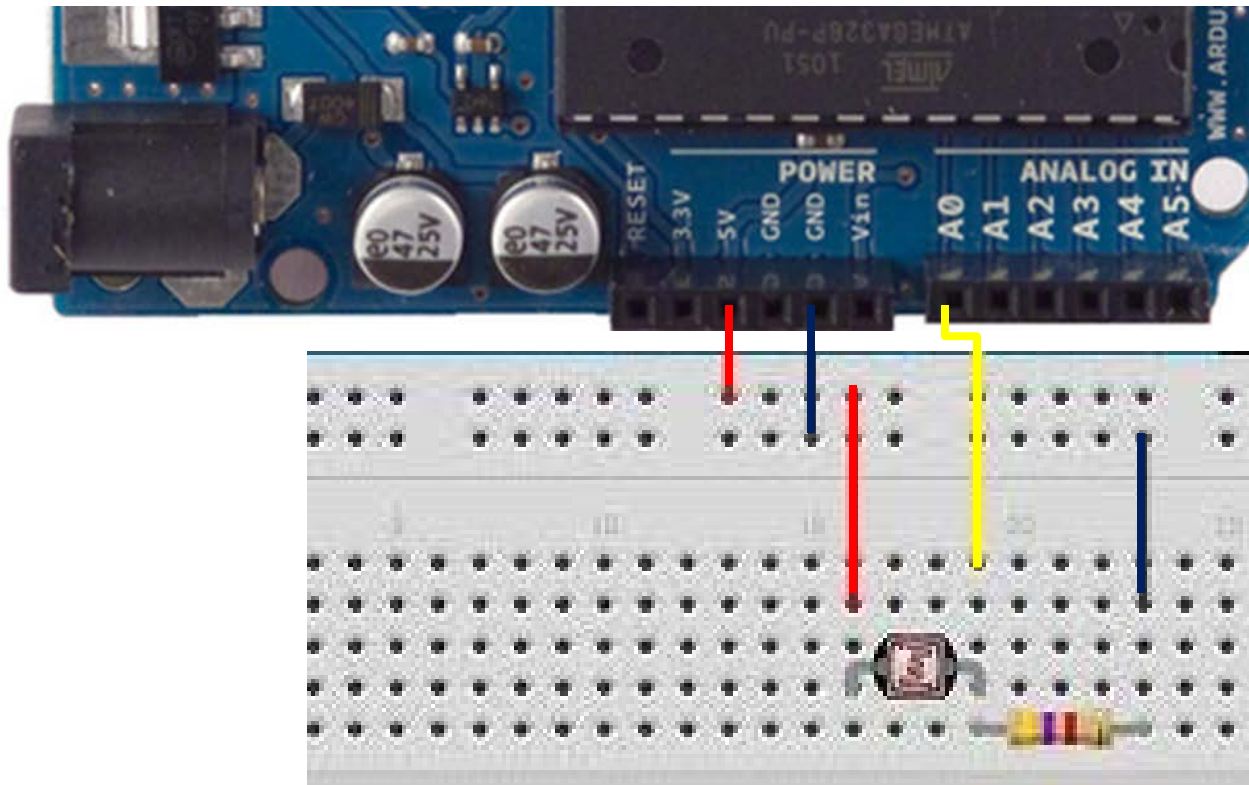


# Analog I/O

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



# Analog I/O



# Analog Input

```
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);
}
```

PWM pin!!!



# Serial Communication

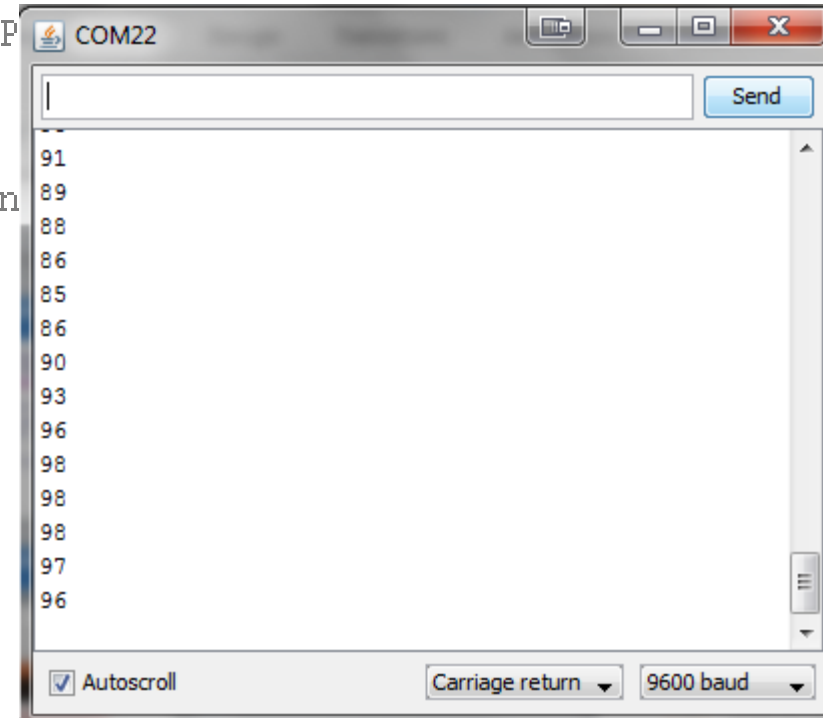
```
// Example 11-06 from "Getting Started with P
// by Reas & Fry. O'Reilly / Make 2010

// 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

- **Allows Arduino to communicate**
  - **Directly**
    - via USB cable / COM port
  - **Wireless**
    - via WiFi module
    - via Xbee
    - via bluetooth module

# Take care

- **Arduino:**
  - **Serial.print** does translation to **ASCII text**
    - int value 12 into 2 characters '1' and '2'
    - float value 1.23 as 4 characters '1', '.', '2' and '3'
    - String value "Hi!" as 3 characters 'H', 'i' and '!'
  - **Serial.println** does similar + end of line
  - **Serial.write** writes a single *byte*
- **Processing/Arduino**
  - **Serial monitoring and communication to Processing cannot be done at the same time...**

# Examples

- **Bad code / Good code**



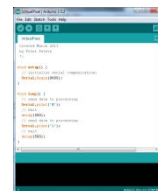
- **Arduino communicating to Processing**



SerialPrint



SensorGraph



SensorRead





That was Arduino.

**TU/e**

Technische Universiteit  
Eindhoven  
University of Technology

Where innovation starts