



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**
- **Taking to Processing**

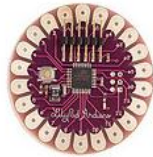
Why Arduino?

- **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



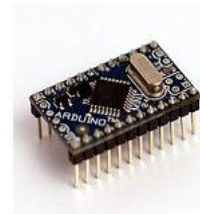
Arduino Mega 2560



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



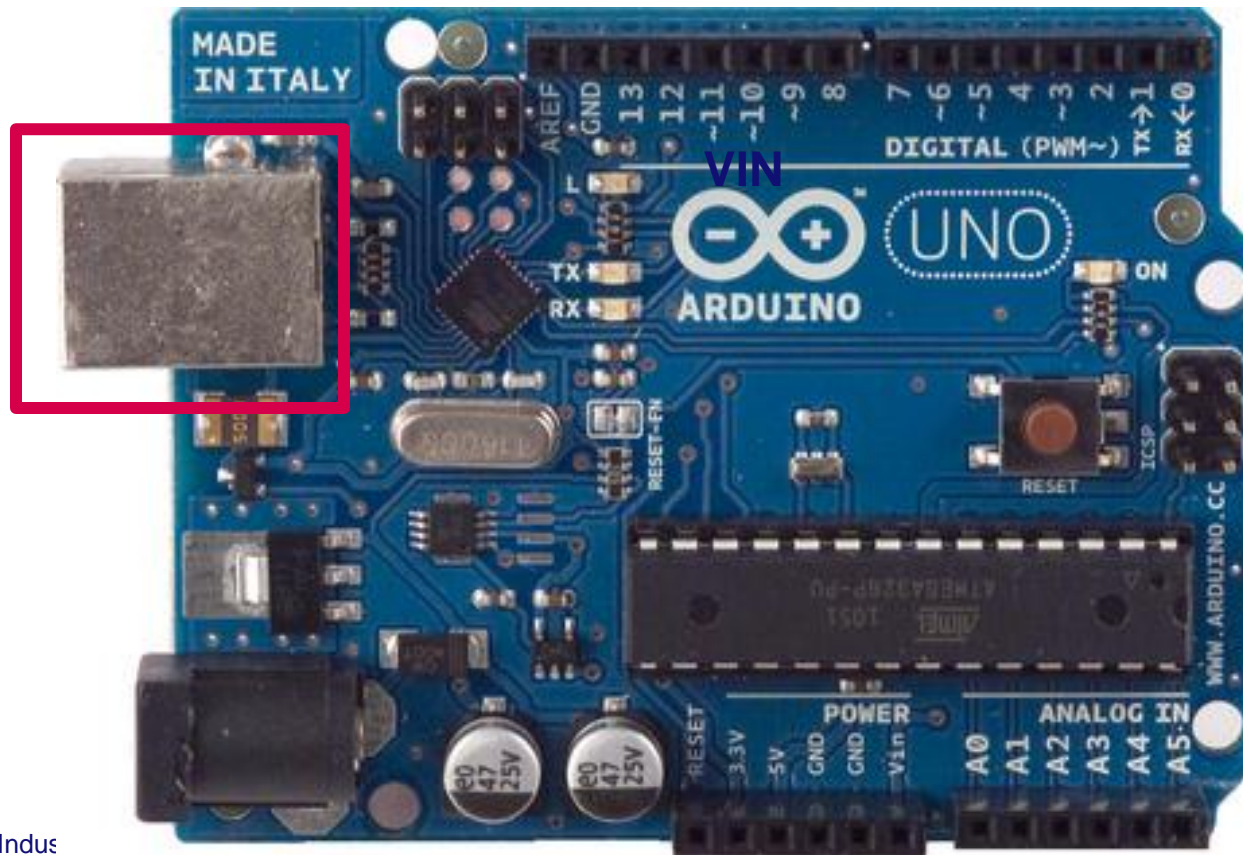
UNO

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



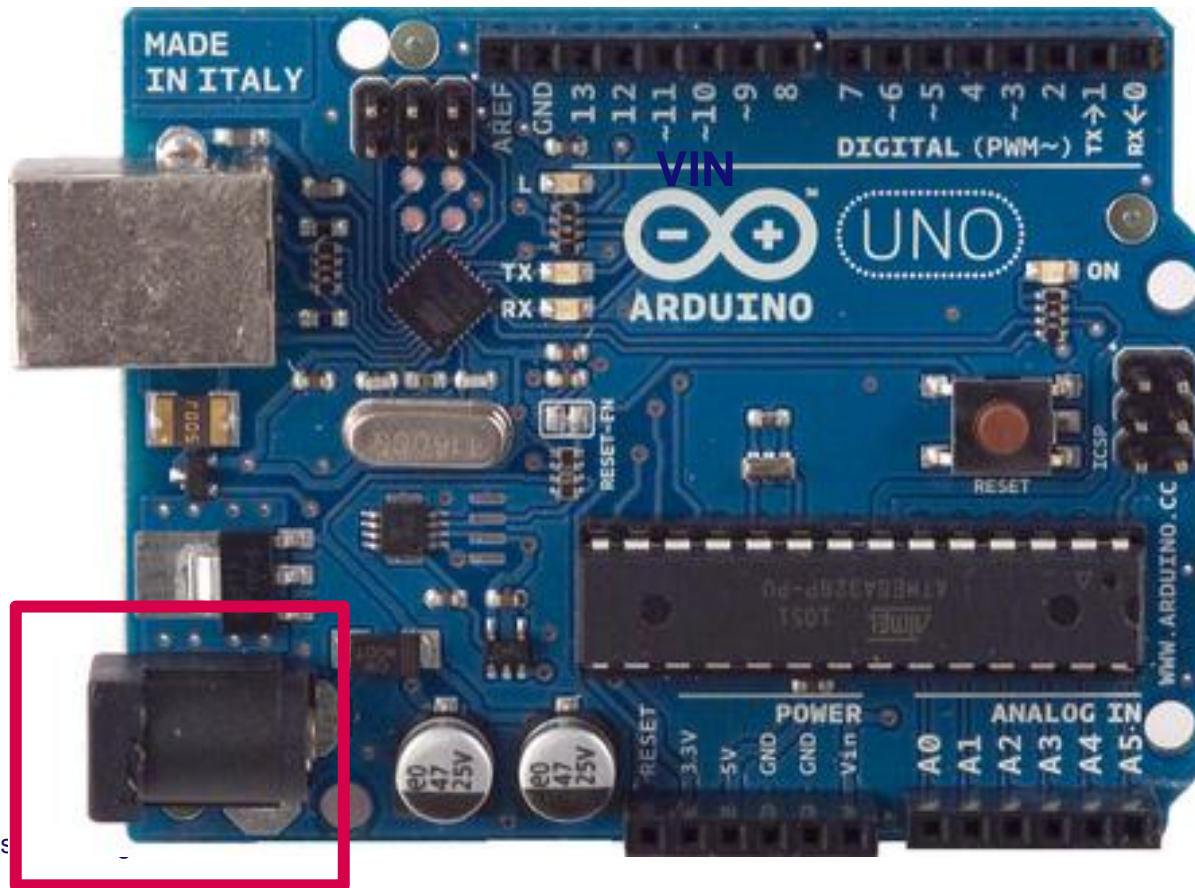
UNO

- 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



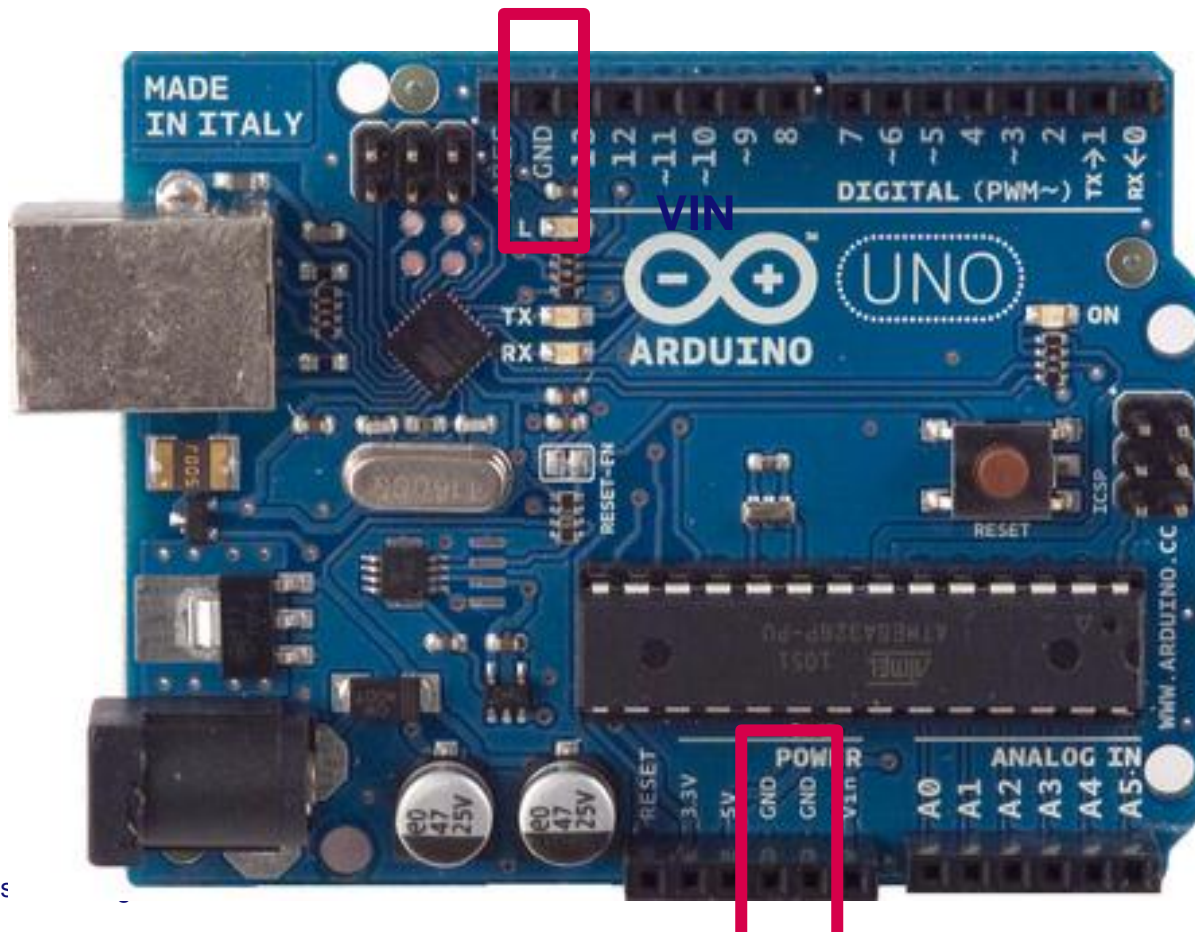
UNO

- Power: 3.3V supply



UNO

- Power: GND pins



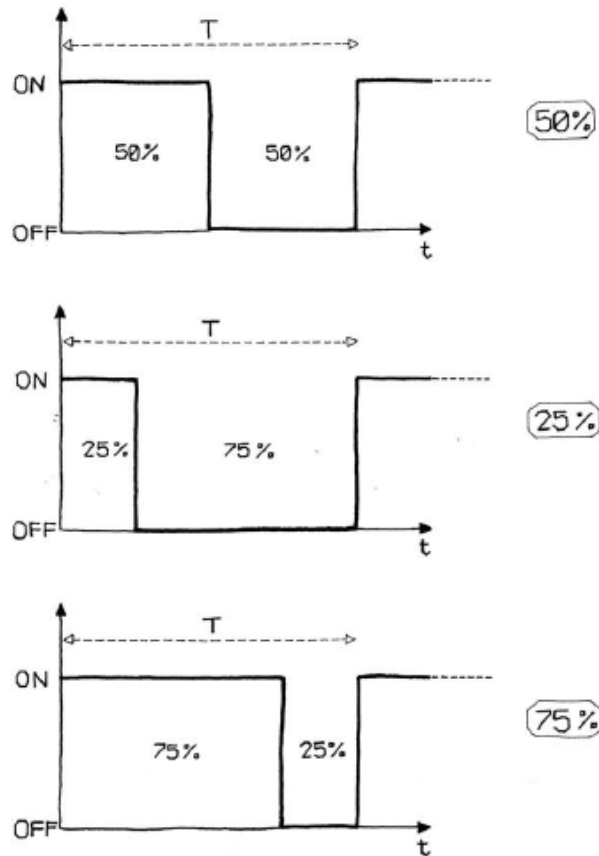
UNO

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



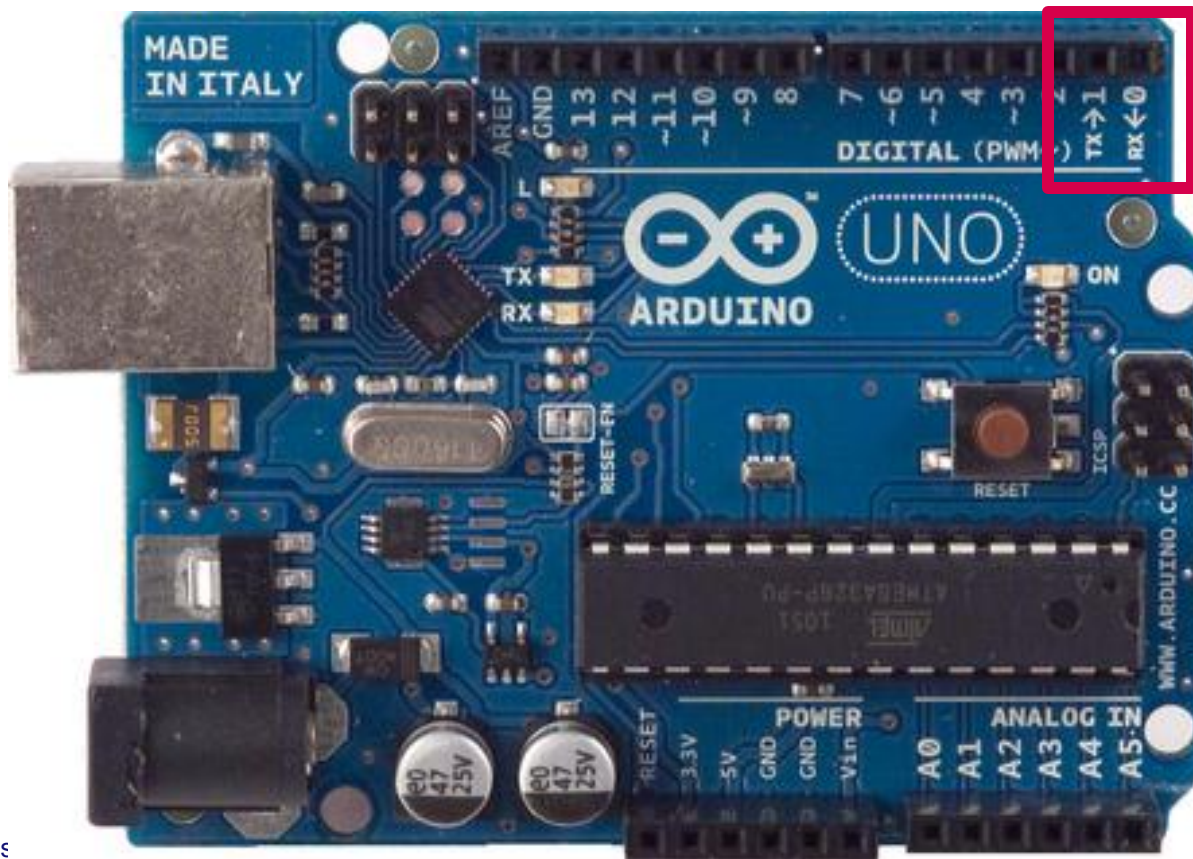
UNO

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



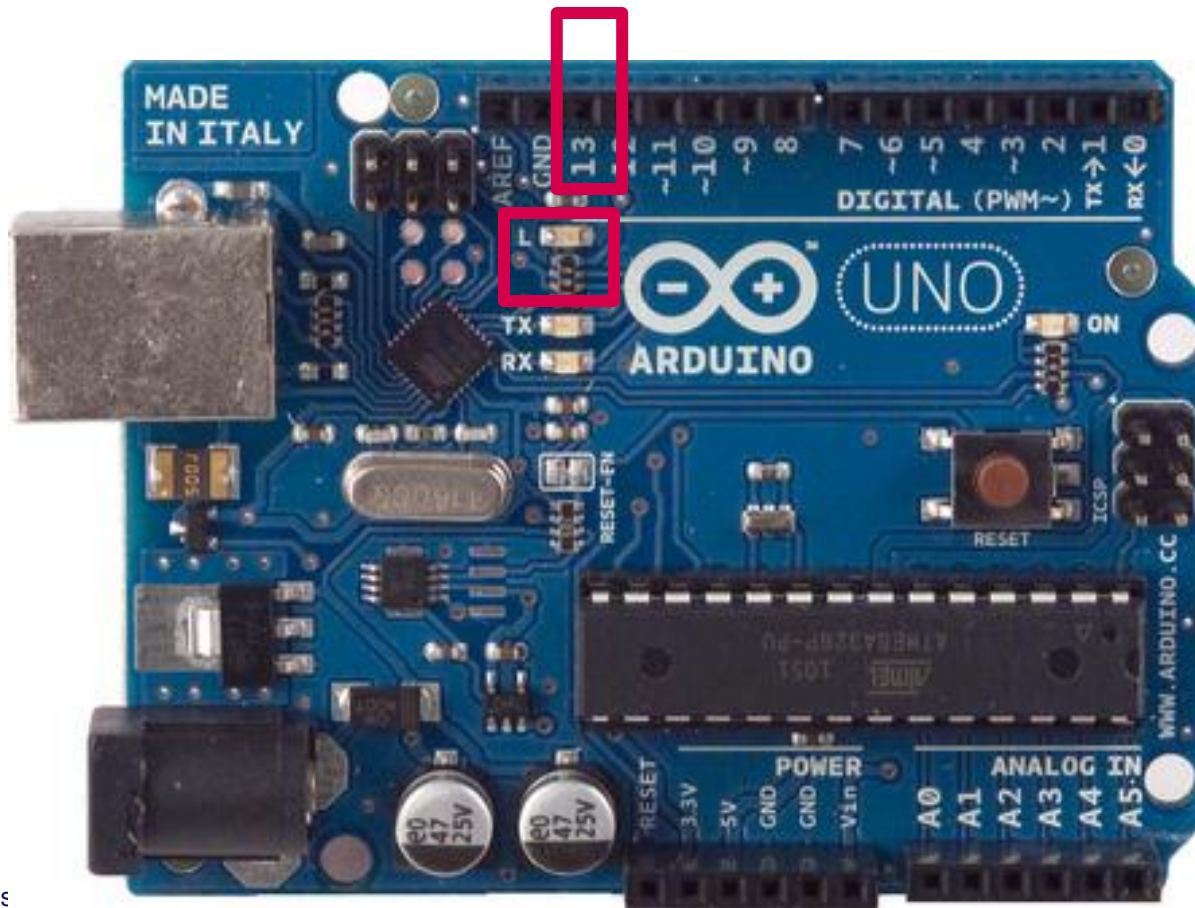
UNO

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



UNO

- LED: 13



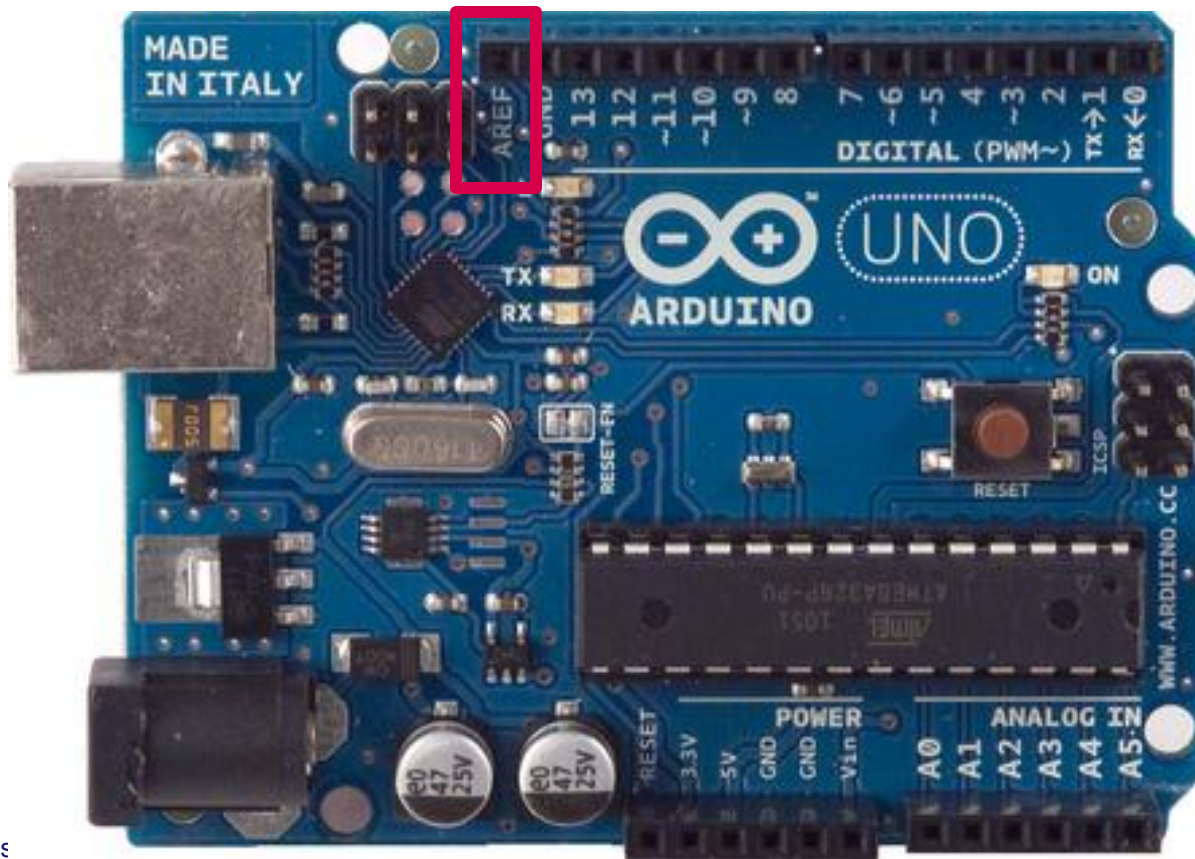
UNO

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



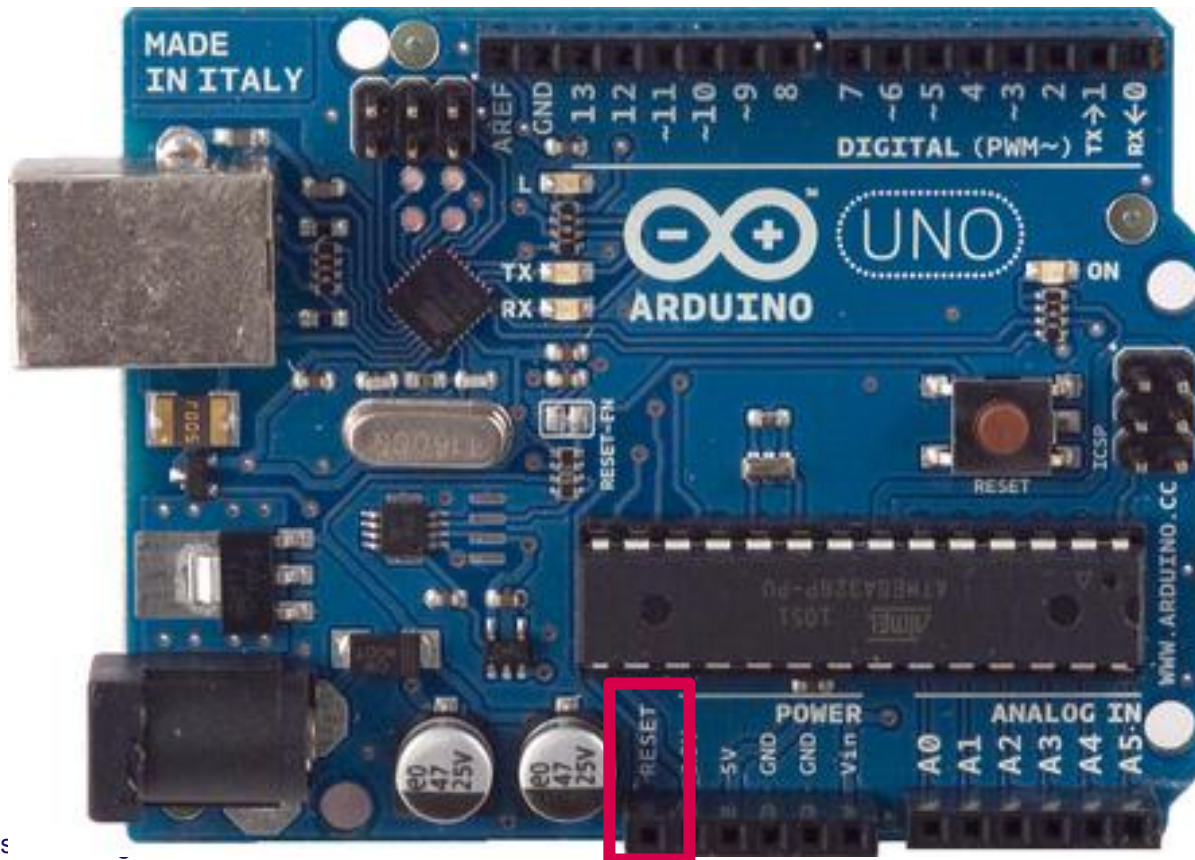
UNO

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



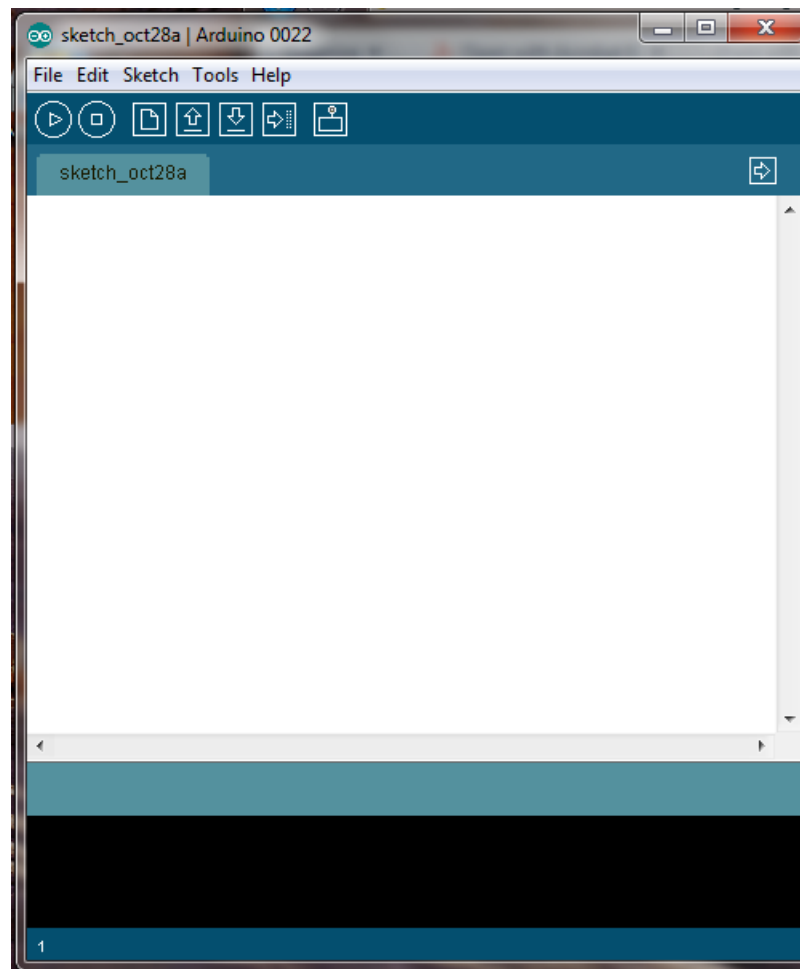
UNO

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



Software: IDE

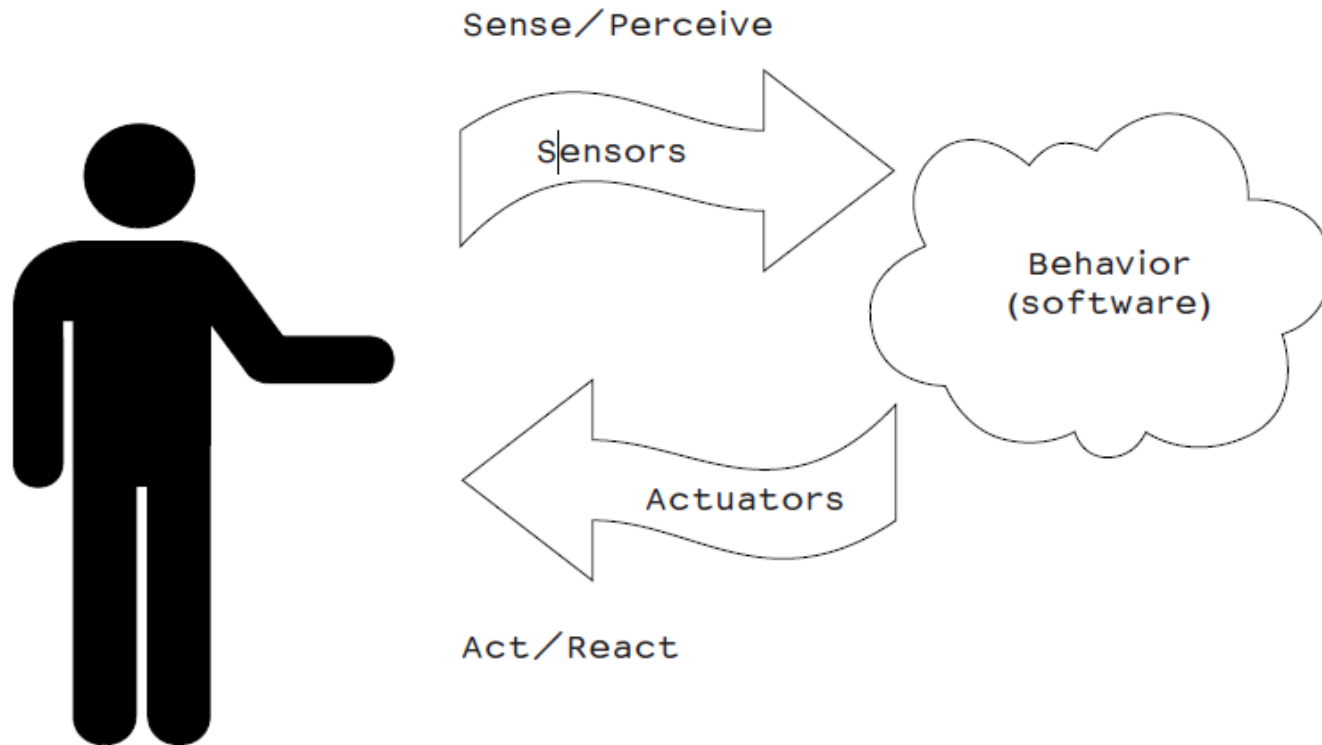
- <http://arduino.cc/en/Main/Software>



Driver Installation and Port Identification

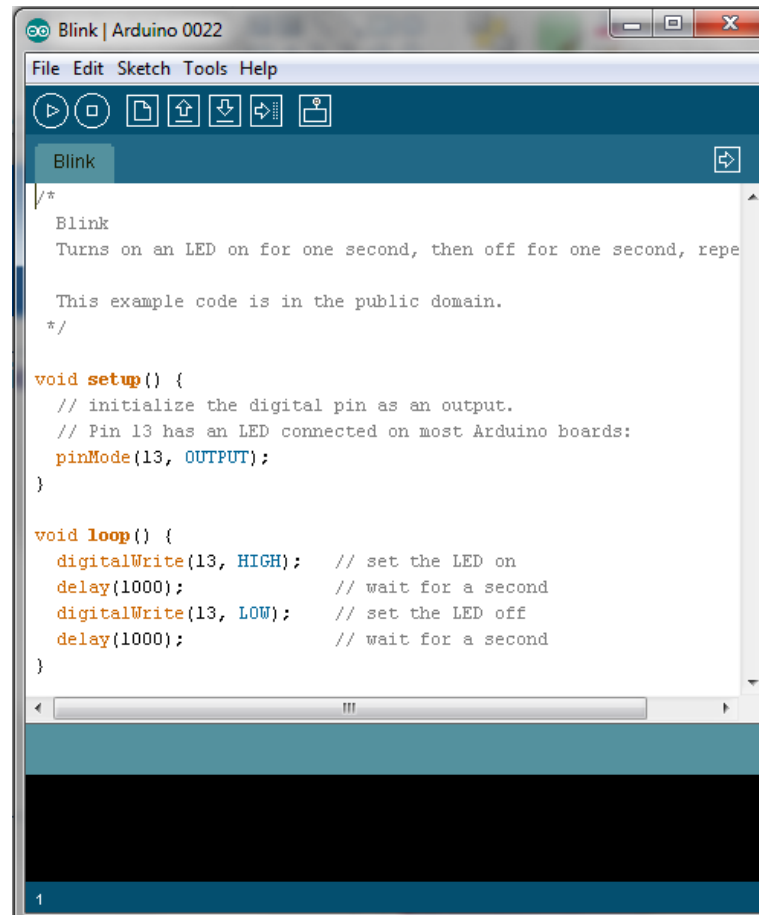
- Refer to the instructions in
 - “Getting Started with Arduino”, page 23-26

Really getting started



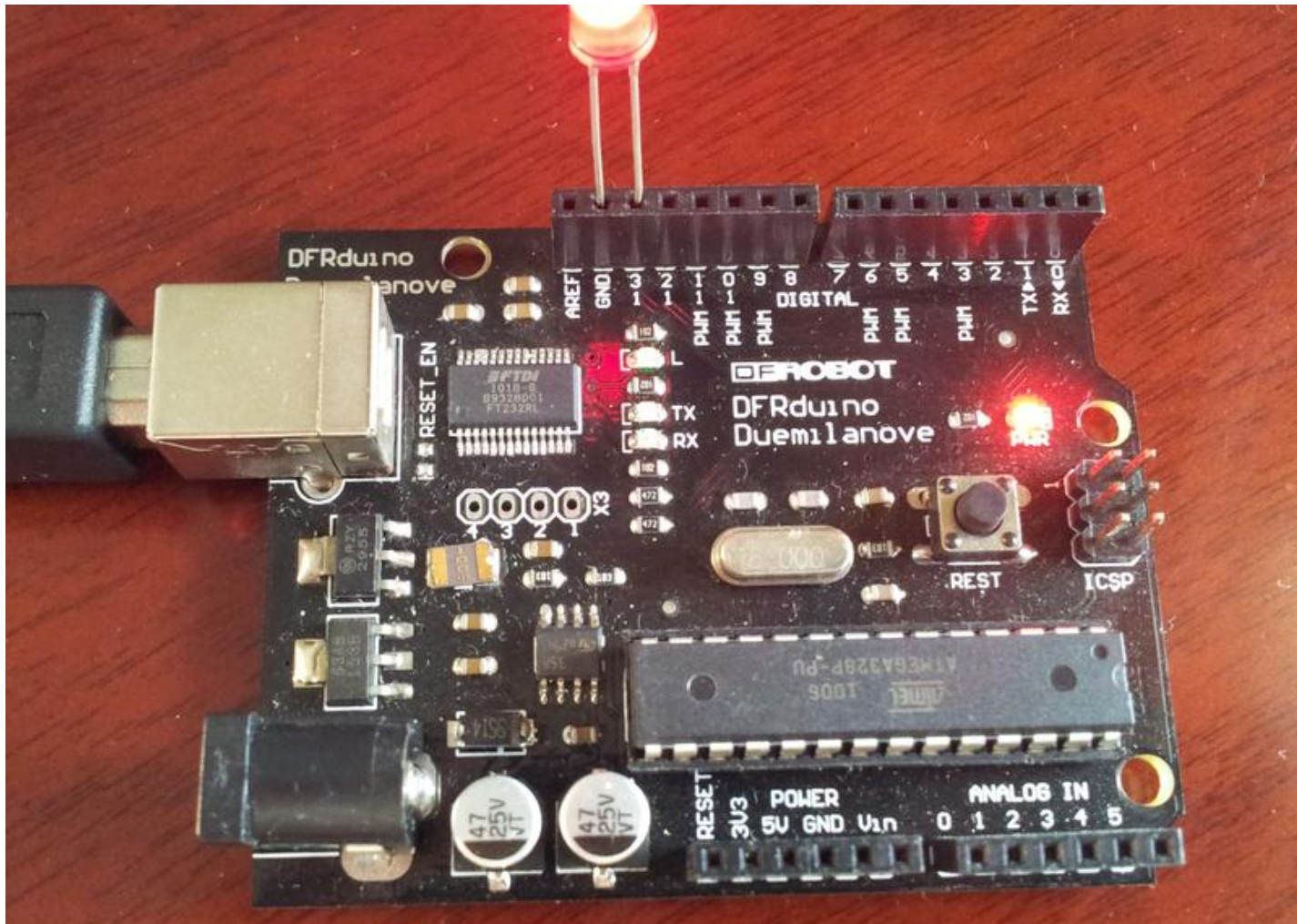
Blinking an LED

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

A screenshot of the Arduino IDE interface. The title bar reads "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 displays the "Blink" example code. The code includes a multi-line comment describing the function, a setup function to initialize pin 13 as an output, and a loop function that toggles the LED on and off with 1000ms delays. The IDE has a light blue theme.

```
/*  
  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



University of Technology

University of Technology

Blink an LED

- **#define LED 13**

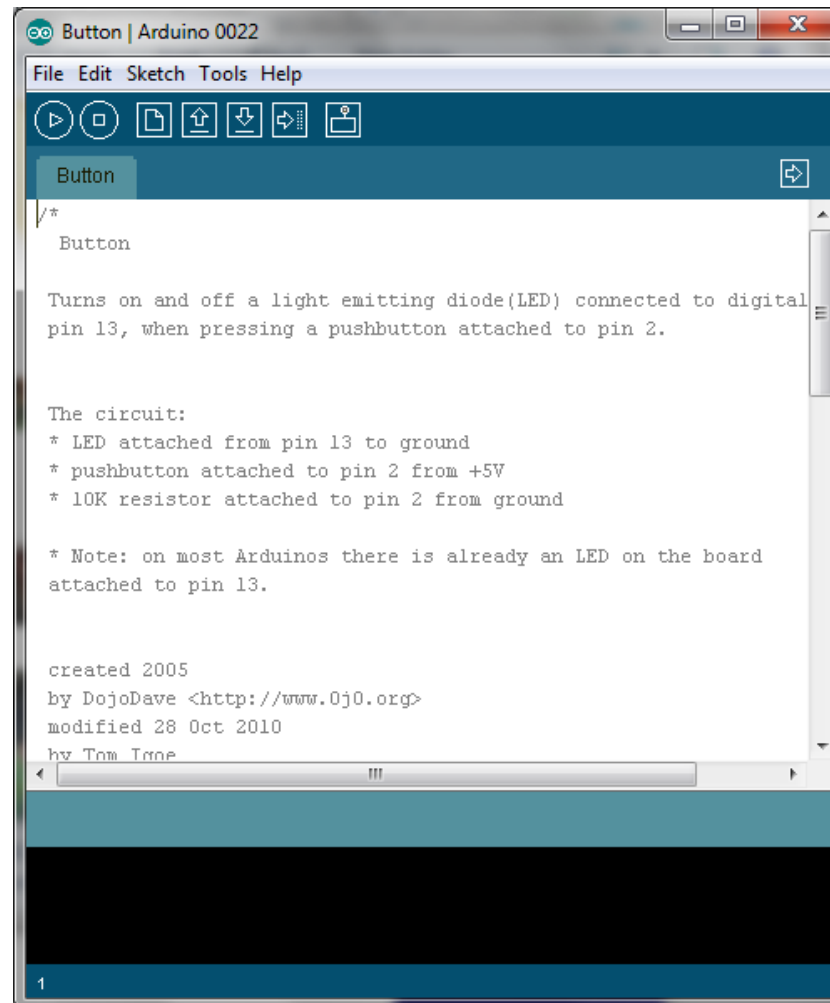
```
#define LED 13

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

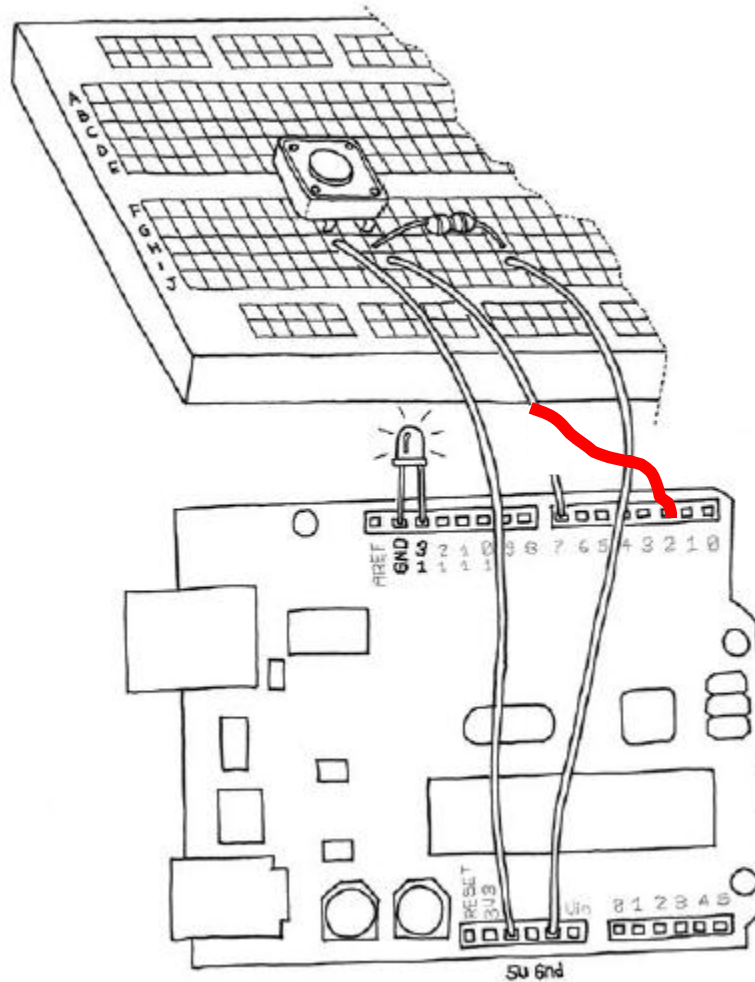
void loop() {
    digitalWrite(LED, HIGH);    // set the LED on
    delay(1000);                // wait for a second
    digitalWrite(LED, LOW);     // set the LED off
    delay(1000);                // wait for a second
}
```

Button to control the LED

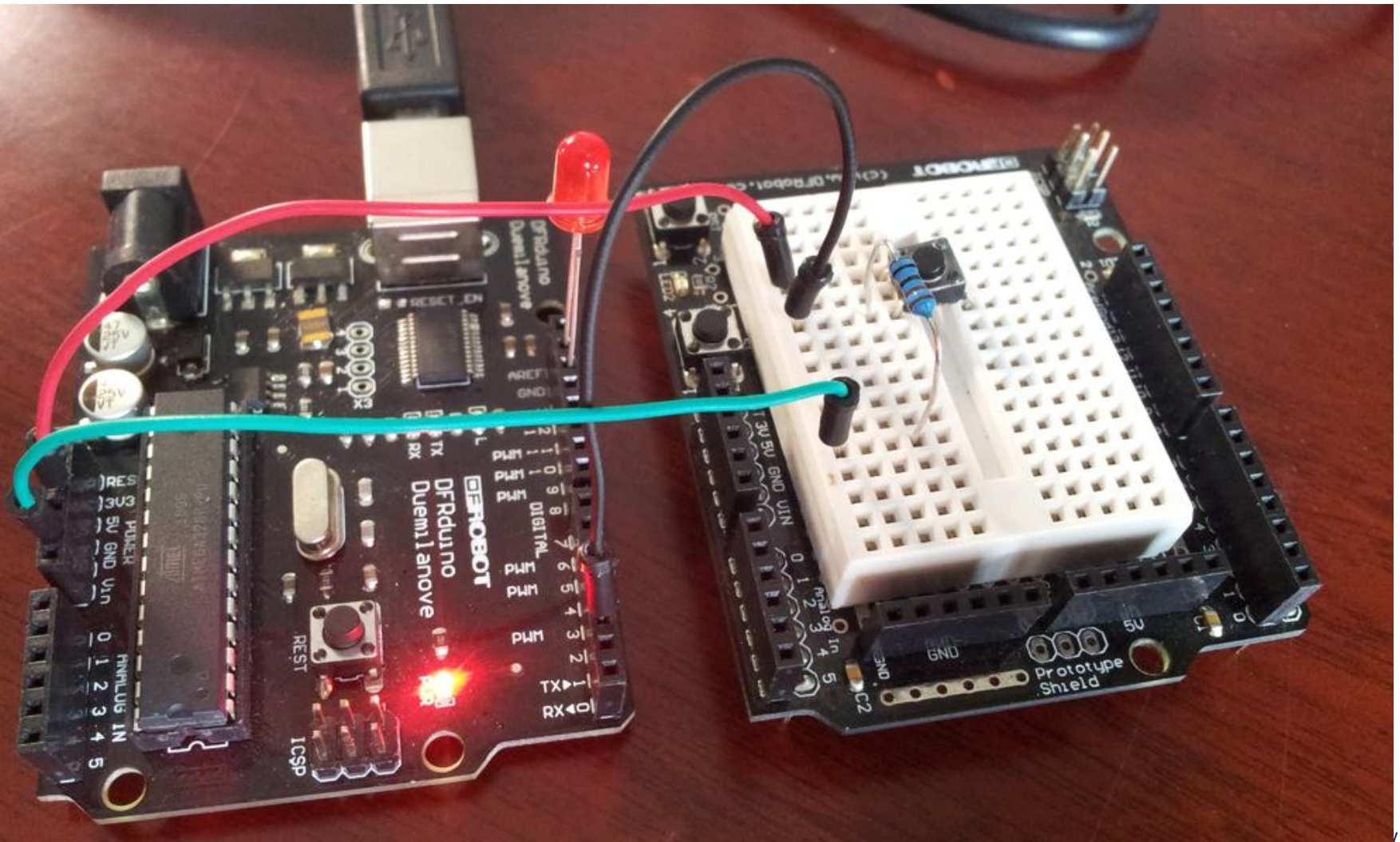
- **File>Examples>Digital>Button**



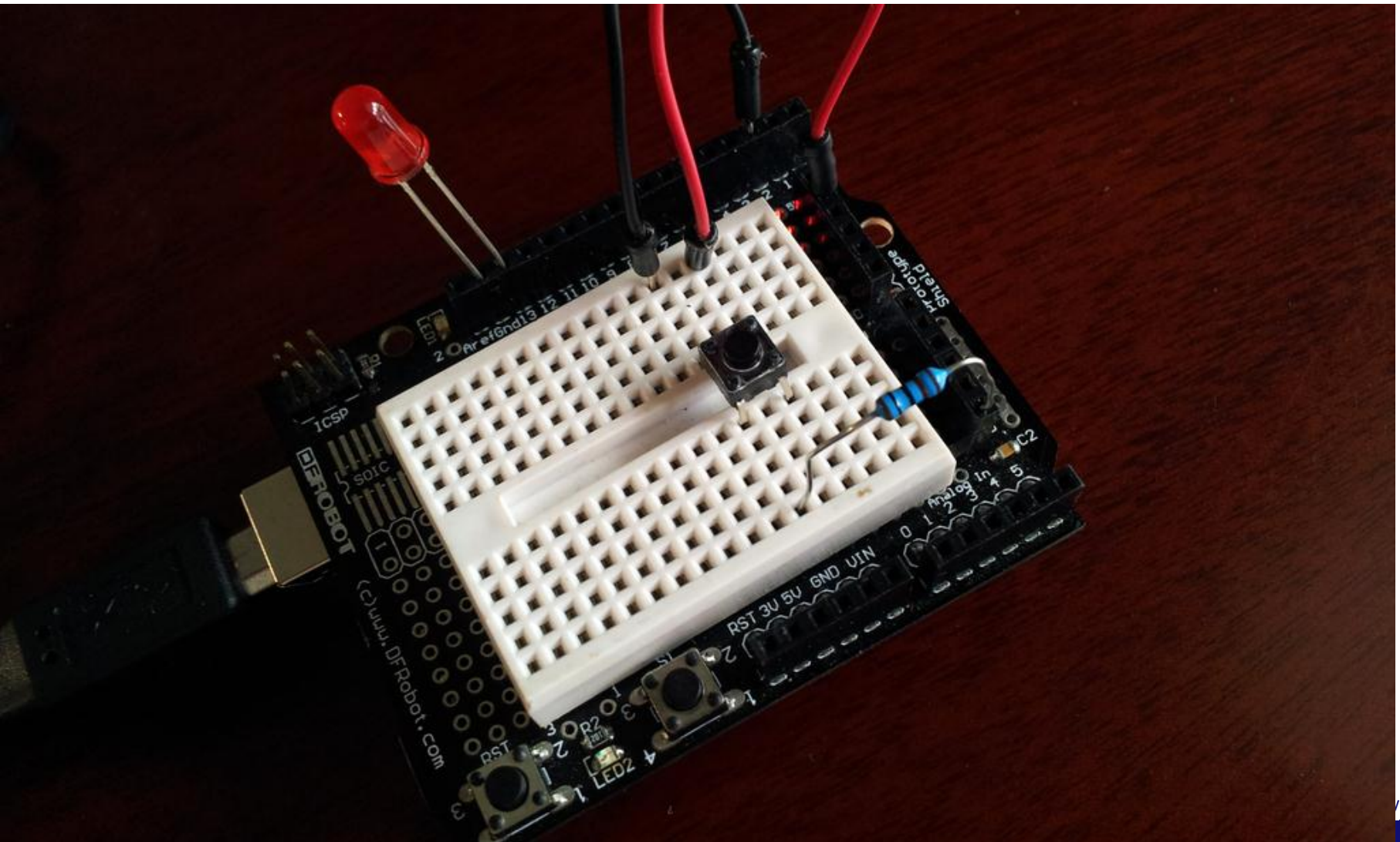
Button to control the LED



Button to control the LED



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

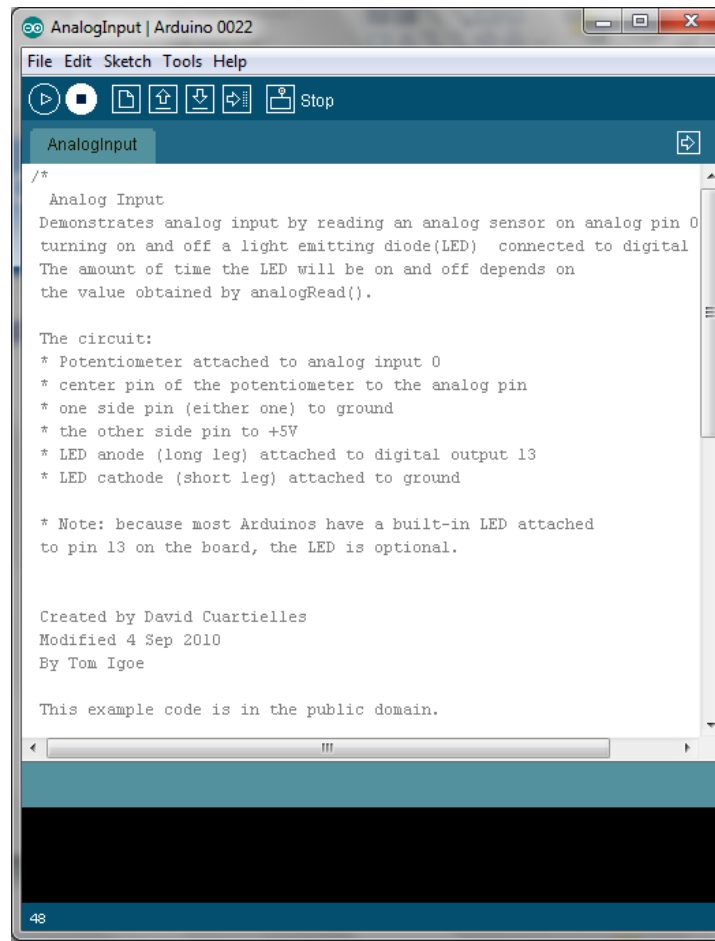
void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
  // initialize the pushbutton pin as an input:
  pinMode(buttonPin, INPUT);
}
```

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'. The toolbar contains icons for running, uploading, saving, 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().

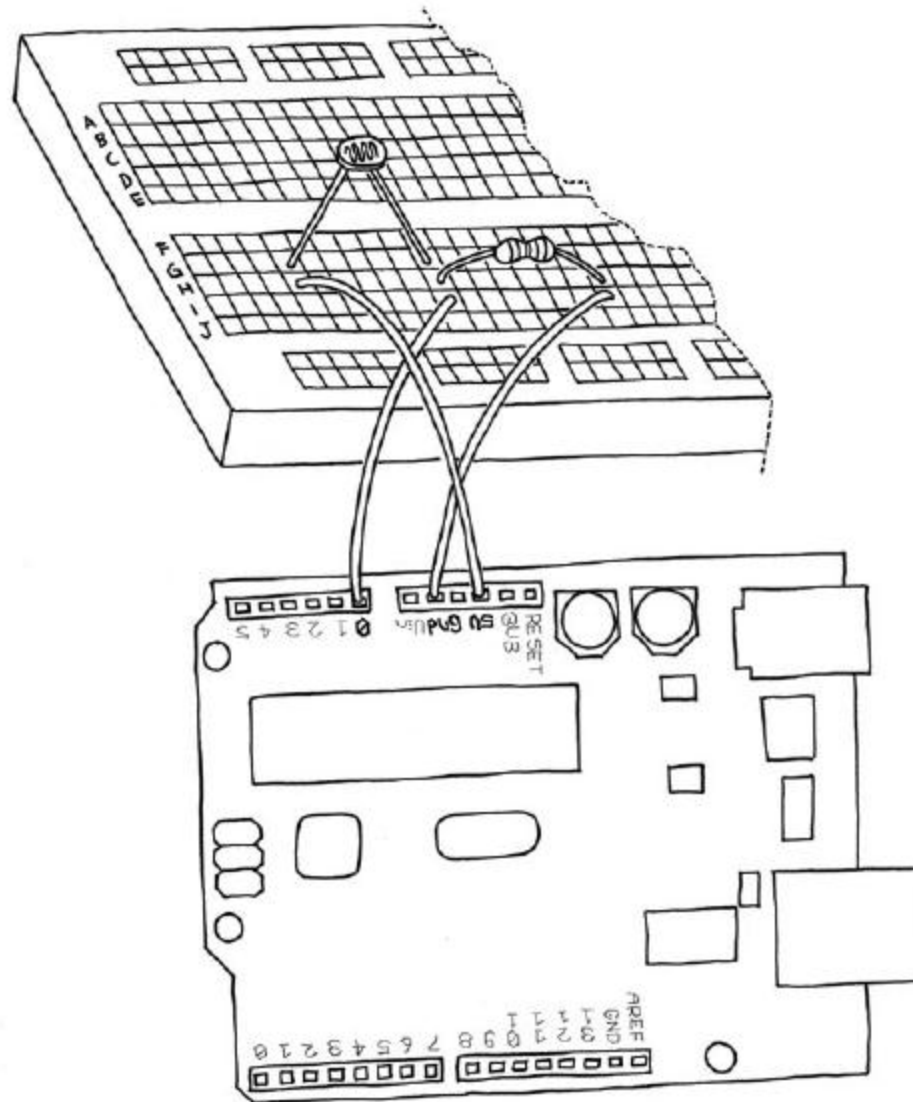
  The circuit:
  * Potentiometer attached to analog input 0
  * center pin of the potentiometer to the analog pin
  * one side pin (either one) to ground
  * the other side pin to +5V
  * LED anode (long leg) attached to digital output 13
  * LED cathode (short leg) attached to ground

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

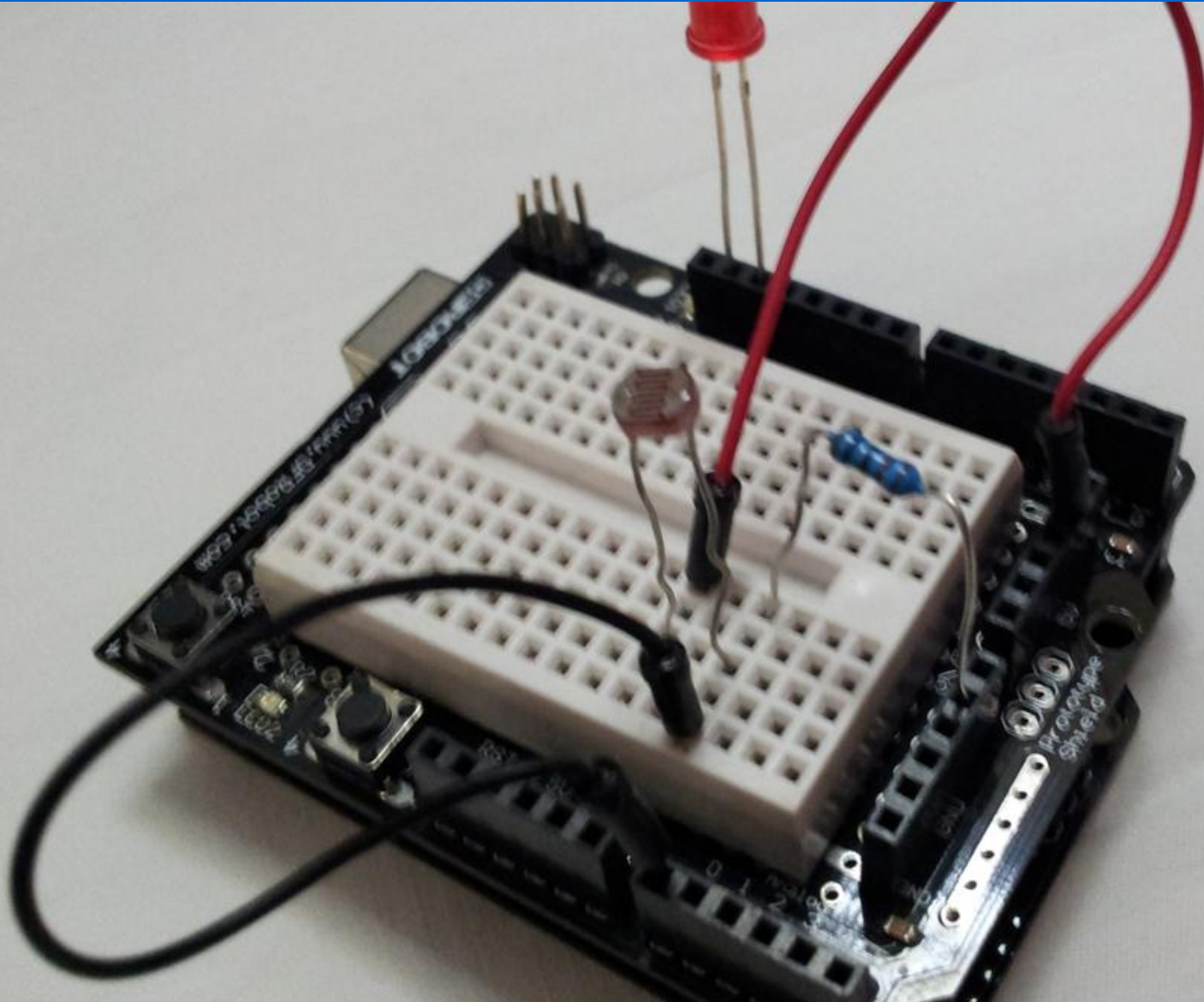
  Created by David Cuartielles
  Modified 4 Sep 2010
  By Tom Igoe

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


Analog I/O



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

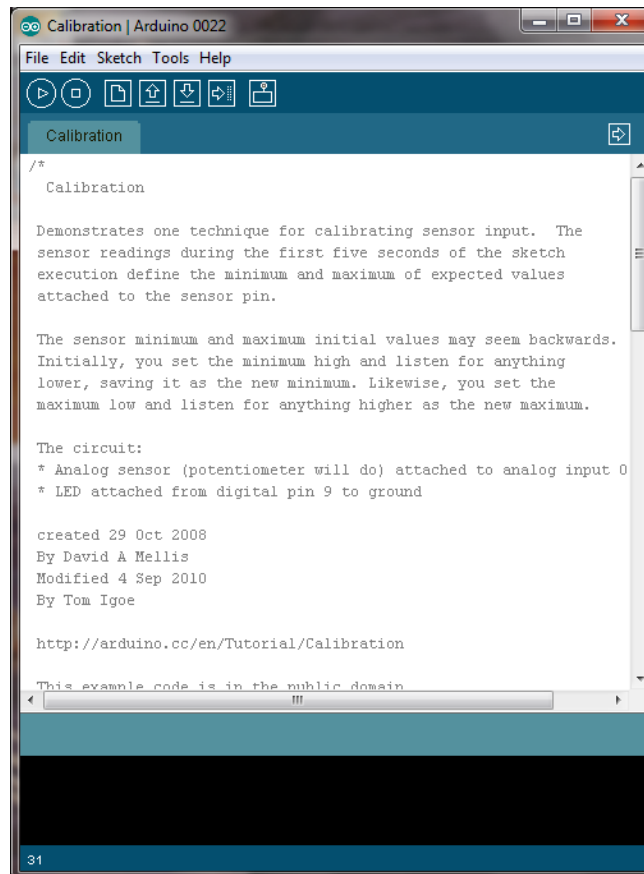
```
int sensorPin = A0;    // select the input pin
int ledPin = 11;       // select the pin for the LED
int sensorValue = 0;   // variable to store the value coming from the sensor

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);
}
```


Analog Input Calibration

- **File>Examples>Analog>Calibration**
 - **Connect an LED to pin 9, and the other to pin 13.**



The screenshot shows the Arduino IDE window titled "Calibration | 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 displays the "Calibration" sketch. The sketch includes a comment block explaining the technique for calibrating sensor input, the circuit setup (analog sensor on pin 0, LED on pin 9), and the author information (David A. Mellis, Tom Igoe). The sketch is 34 lines long.

```
/*
  Calibration

  Demonstrates one technique for calibrating sensor input. The
  sensor readings during the first five seconds of the sketch
  execution define the minimum and maximum of expected values
  attached to the sensor pin.

  The sensor minimum and maximum initial values may seem backwards.
  Initially, you set the minimum high and listen for anything
  lower, saving it as the new minimum. Likewise, you set the
  maximum low and listen for anything higher as the new maximum.

  The circuit:
  * Analog sensor (potentiometer will do) attached to analog input 0
  * LED attached from digital pin 9 to ground

  created 29 Oct 2008
  By David A. Mellis
  Modified 4 Sep 2010
  By Tom Igoe

  http://arduino.cc/en/Tutorial/Calibration

  This example code is in the public domain
  */

int sensorPin = 0;    // the pin the sensor is attached to
int ledPin = 9;       // the pin the LED is attached to

// variables will store the minimum and maximum sensor values
int sensorMin = 1023;
int sensorMax = 0;

// variables will store the current sensor readings
int currentSensorReading = 0;

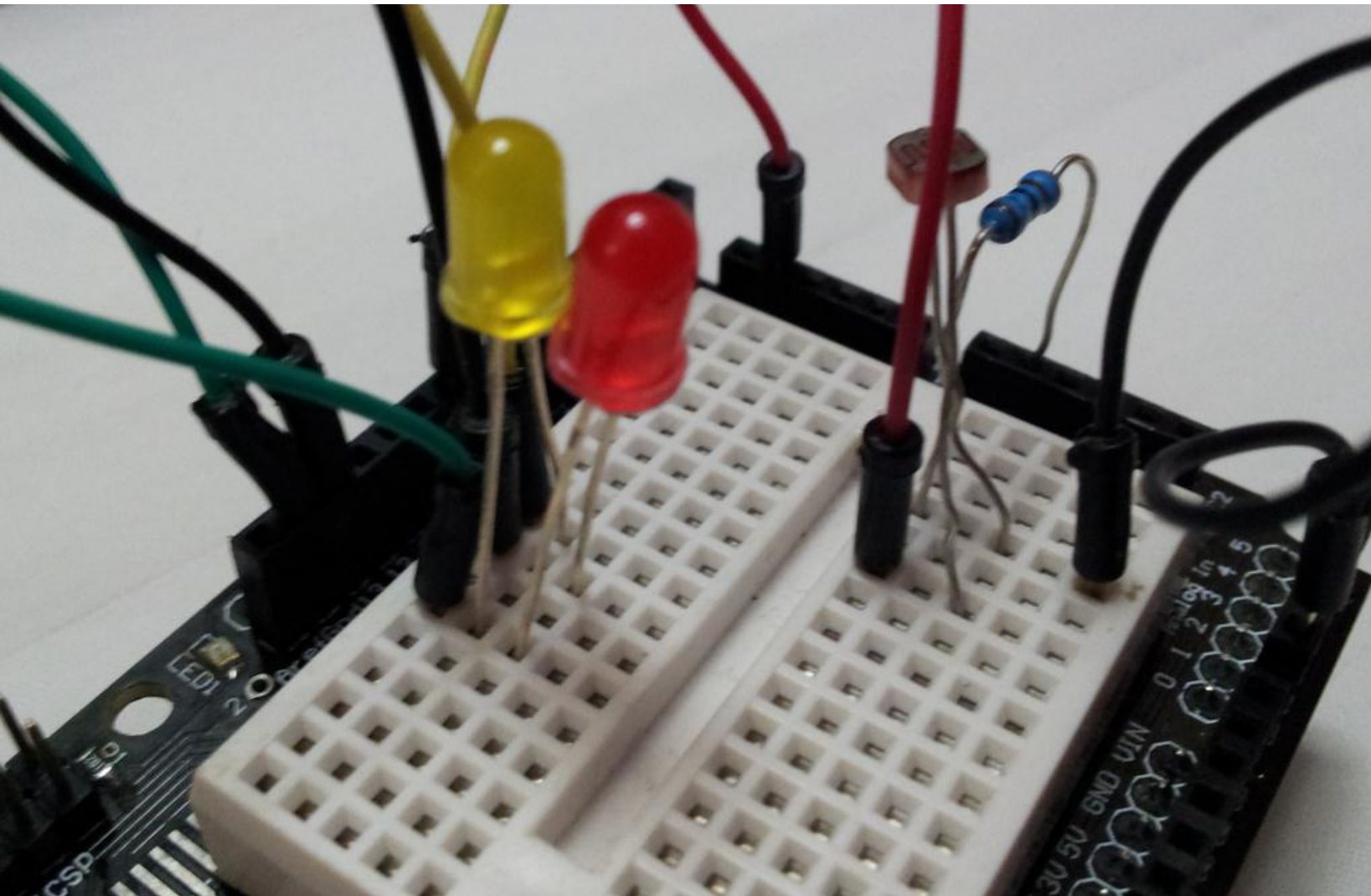
void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  // read the sensor value
  currentSensorReading = analogRead(sensorPin);

  // determine the new minimum and maximum values
  if (currentSensorReading < sensorMin) {
    sensorMin = currentSensorReading;
  }
  if (currentSensorReading > sensorMax) {
    sensorMax = currentSensorReading;
  }

  // turn the LED on if the sensor reading is between the minimum and maximum
  if (currentSensorReading > sensorMin && currentSensorReading < sensorMax) {
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }
}
```

Analog Input Calibration



Analog Input Calibration

```
// These constants won't change:
const int sensorPin = A0;    // pin that the sensor is attached to
const int ledPin = 9;        // pin that the LED is attached to

// variables:
int sensorValue = 0;         // the sensor value
int sensorMin = 1023;        // minimum sensor value
int sensorMax = 0;           // maximum sensor value
```

Analog Input Calibration

```
void setup() {  
    // turn on LED to signal the start of the calibration period:  
    pinMode(13, OUTPUT);  
    digitalWrite(13, HIGH);  
  
    // calibrate during the first five seconds  
    while (millis() < 5000) {  
        sensorValue = analogRead(sensorPin);  
  
        // record the maximum sensor value  
        if (sensorValue > sensorMax) {  
            sensorMax = sensorValue;  
        }  
  
        // record the minimum sensor value  
        if (sensorValue < sensorMin) {  
            sensorMin = sensorValue;  
        }  
    }  
  
    // signal the end of the calibration period  
    digitalWrite(13, LOW);  
}
```


Analog Input Calibration

```
void loop() {  
  // read the sensor:  
  sensorValue = analogRead(sensorPin);  
  
  // apply the calibration to the sensor reading  
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);  
  
  // in case the sensor value is outside the range seen during calibration  
  sensorValue = constrain(sensorValue, 0, 255);  
  
  // fade the LED using the calibrated value:  
  analogWrite(ledPin, sensorValue);  
}
```

Serial Communication

- **We use the same example for Calibration**

Serial Communication

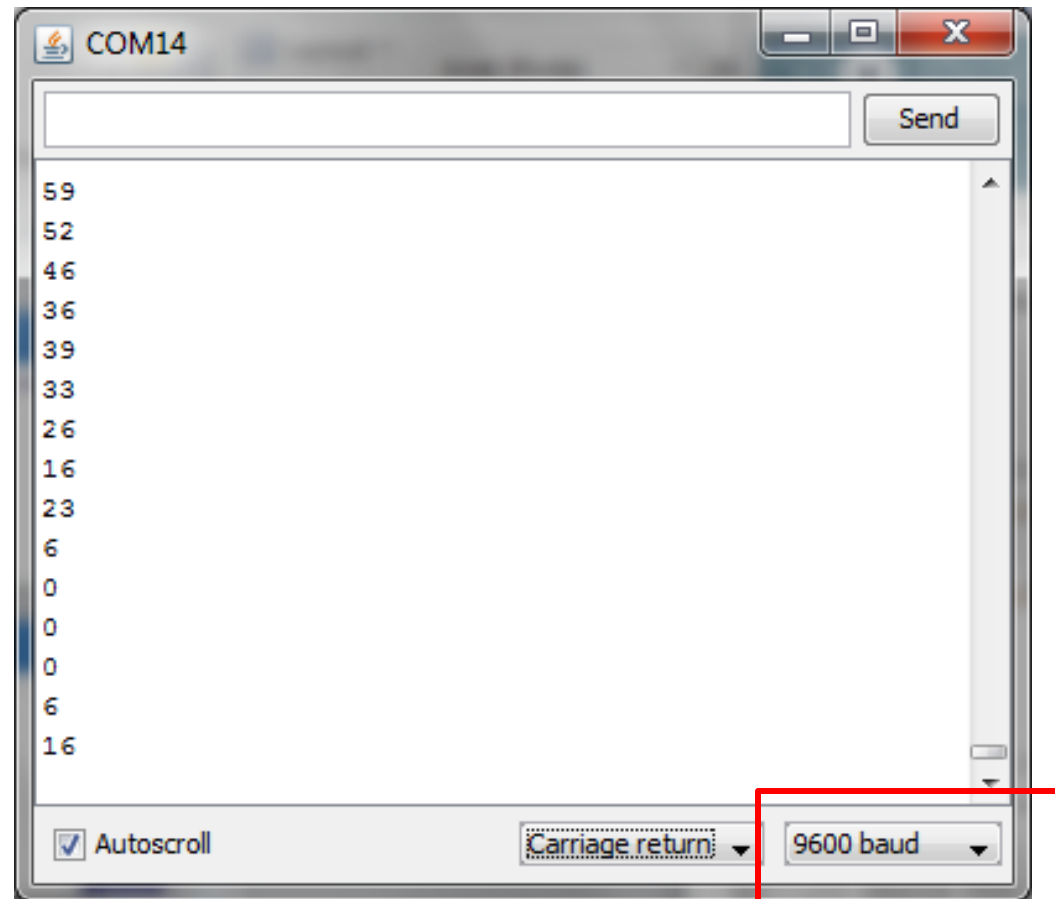
```
void setup() {  
  // turn on LED to signal the start of the calibration period:  
  
  pinMode(13, OUTPUT);  
  digitalWrite(13, HIGH);  
  
  // calibrate during the first five seconds  
  while (millis() < 5000) {  
    sensorValue = analogRead(sensorPin);  
  
    // record the maximum sensor value  
    if (sensorValue > sensorMax) {  
      sensorMax = sensorValue;  
    }  
  
    // record the minimum sensor value  
    if (sensorValue < sensorMin) {  
      sensorMin = sensorValue;  
    }  
  }  
  
  // signal the end of the calibration period  
  digitalWrite(13, LOW);  
  
  Serial.begin(9600);  
}
```

Serial Communication

```
void loop() {  
  // read the sensor:  
  sensorValue = analogRead(sensorPin);  
  
  // apply the calibration to the sensor reading  
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);  
  
  // in case the sensor value is outside the range seen during calibration  
  sensorValue = constrain(sensorValue, 0, 255);  
  
  // fade the LED using the calibrated value:  
  analogWrite(ledPin, sensorValue);  
  
  Serial.println(sensorValue);  
  delay(100);  
}
```

Serial Communication

- Try it out.



Serial Communication

- Now change it a bit. Try again the Serial Monitor

```
void loop() {  
  // read the sensor:  
  sensorValue = analogRead(sensorPin);  
  
  // apply the calibration to the sensor reading  
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);  
  
  // in case the sensor value is outside the range seen during calibration  
  sensorValue = constrain(sensorValue, 0, 255);  
  
  // fade the LED using the calibrated value:  
  analogWrite(ledPin, sensorValue);  
  
  Serial.write(sensorValue);  
  delay(100);  
}
```

Serial Communication

- Now try to receive the sensor input from Processing
- 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()[1];
    port = new Serial(this, arduinoPort, 9600);
}
```

Serial Communication

```
void draw() {  
  background(255);  
  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

- Now use the same hardware, try out
- In **Processing**
 - File>Examples>Books>Chapter 11>Ex_11_08

- **That was Arduino.**