

Workshop Basic Arduino

Class 1 – Arduino

Fundamentals

MSc. David Velásquez Rendón

Contents

1. Introduction.
2. Arduino MEGA 2560 Pinout.
3. Electronics Common Notations.
4. Variables in Arduino Programming.
5. Typical Operators for Arduino Programming.
6. Arduino Program Structure.
7. Arduino common used commands.
8. Arduino common statements.

3 Arduino

► ¿What is?

- Open source electronics platform.
- Allows creation of electronics prototypes.
- Based on open source software.
- Ease of use.
- Allows to do sequences and mathematics operations.
- Used for automation.

► Arduino Types



Arduino Uno



Arduino Leonardo



Arduino Due



Arduino Yún



Arduino Micro



Arduino Robot



Arduino Esplora



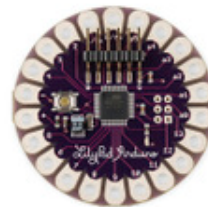
Arduino Mega 2560



Arduino Ethernet



Arduino Mini



LilyPad Arduino

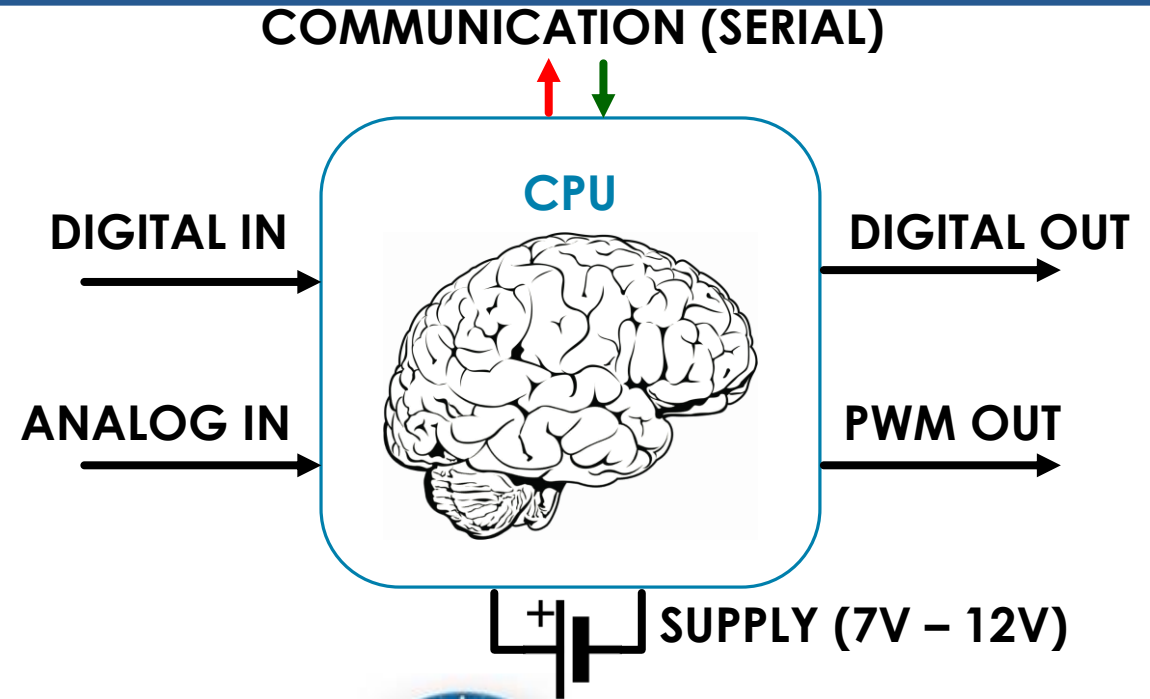


Arduino Nano

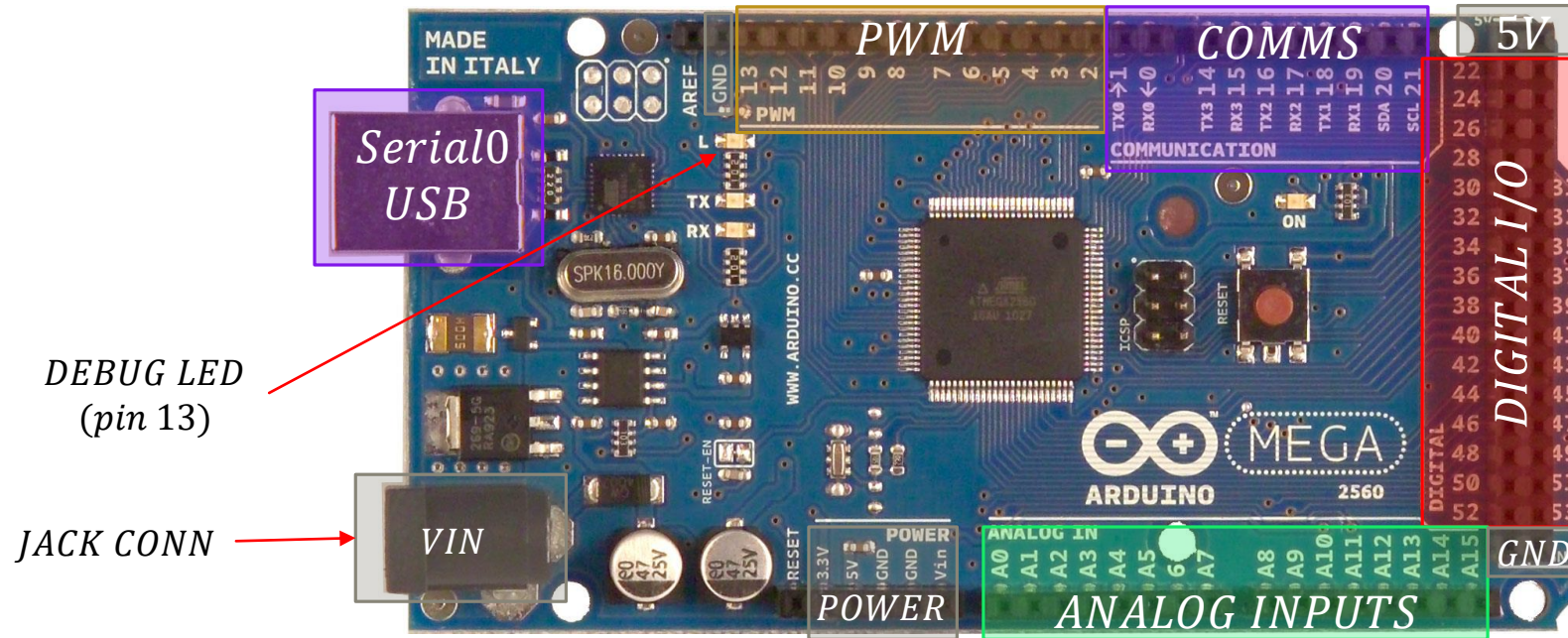


Arduino MKR1000

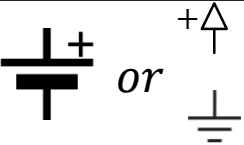
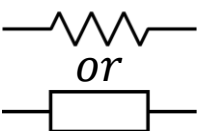
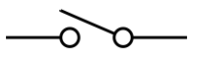
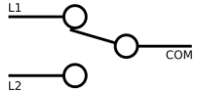
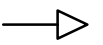
IoT



4 Arduino MEGA 2560 Pinout



- Based on the microcontroller **ATMEGA 2560**.
- **Supply:** Using Jack connector with voltage range from **7V to 12V**. Also it can be powered via USB but with low consumption.
- **Maximum current per I/O digital pin:** 40 mA.
- **Maximum current for 3.3V supply pin:** 50 mA.
- **Digital Inputs and Outputs:** 54 pins (**0V or 5V**).
- **Analog Inputs:** 16 pins (from **0V to 5V** of analog voltage).
- **PWM Outputs:** 15 pins (Considered in the 54 digital I/O pins). *Can be used as normal digital I/O pins too.*
- **Oscillator frequency/Flash Memory/RAM/ROM:** 16 MHz / Flash 256 Kb/ SRAM 8 Kb / EEPROM 4 Kb.
- **Communications:** USB CDC (Serial) + I2C + SPI.

<i>Symbol</i>	<i>Description</i>
	Power supply: Cell or battery
	Resistor
	Switch: Single Pole Single Throw (SPST)
	Switch: Single Pole Double Throw (SPDT)
	Output Pin

<i>Cable Color</i>	<i>Notation</i>
RED	Positive (+)
BLACK	Negative (-) / GND
BLUE GREEN	Sensor/Actuator Signal

Variables

NAME	SINTAX	SIZE	RANGE		EXAMPLE
			WITHOUT SIGN	WITH SIGN	
Boolean	<code>boolean</code>	1 bit	false True	N/A	<code>boolean state = false;</code>
Char ¹	<code>char</code> <code>unsigned char</code>	8 bits (1 byte)	0 a 255	-128 a 127	<code>char myChar = 'A';</code> <code>char myChar = 65;</code> Both examples are equivalent
Byte	<code>byte</code>	8 bits (1 byte)	0 a 255	N/A	<code>byte hello = B00000111;</code> <code>byte hello = 7;</code> B indicates binary notation B00000111 is equal to 7 in decimal.
Integer	<code>int</code> <code>unsigned int</code>	16 bit (2 bytes)	0 a 65535	-32768 a 32767	<code>unsigned int counter = 0;</code>
Long ²	<code>long</code> <code>unsigned long</code>	32 bit	0 a 4,294,967,295	-2,147,483,648 a 2,147,483,647	<code>unsigned long number = 20000;</code>
Float ³	<code>float</code>	32 bit	N/A	-3.4028235E+38 a 3.4028235E+38	<code>float temperature = 88.5;</code>

¹Check ASCII table (<http://www.asciitable.com/index/asciifull.gif>)

²Timing vars are commonly declared as unsigned long.

³Check Arduino documentation for more info (http://arduino.cc/en/Reference/Float#.UxOT7_I5Njl)

EQUIVALENTS

word	unsigned int
short	int

Typical Operators

	SYMBOL	DESCRIPTION
ARITHMETIC	=	Assignment
	+	Addition
	-	Subtraction
	*	Multiplication
	/	Division
	%	Module
COMPARATIVE	==	Equal to: $x == y$ is equivalent to: x is equal to y ?
	!=	Not equal to: $x != y$ is equivalent to: x is not equal to y ?
	<	Less than
	>	Greater than
	<=	Less than or equal to
	>=	Greater than or equal to
BOOLEANS	&&	AND
		OR
	!	Negation (NOT)
ACCUMULATORS	++	Increment: $y = x ++$ is equivalent to: $y = x + 1$
	--	Decrement: $y = x --$ is equivalent to: $y = x - 1$
	+=	Addition assignment: $y += x$ is equivalent to: $y = y + x$
	-=	Subtraction assignment: $y -= x$ is equivalent to: $y = y - x$
	*=	Multiplication assignment: $y *= x$ is equivalent to: $y = y * x$
	/=	Division assignment: $y /= x$ is equivalent to: $y = y / x$

Arduino Program Structure

Library declaration(e.g: `#include <SFEMP3Shield.h>`)

I/O Pin Labeling (e.g: `#define LEDPIN 13`)

Constant declaration (e.g: `const unsigned int contMax = 10;`)

Variable declaration (e.g: `float temperature = 0;`)

Subroutines or functions declaration:

Example for subroutine:

```
void readSens() {  
    //Example of a subroutine that reads the analog value from 0 to 1023 and converts it  
    //from 0 to 100 degrees storing it in the float variable named temperature.  
    y = analogRead(1); //Analog read from pin A1  
    temperature = y*100.0/1023.0; //Float to degree Celsius conversion  
}
```

Example for function:

```
int sum(int x, int y) { //Example of a function that sums two numbers "x" y "y" and returns the result as int  
    return x + y;  
}
```

Pin configuration and cleaning:

```
void setup() {  
    //CONFIGURATION: Indicate which pins are inputs and which are outputs "pinMode(PIN,OUTPUT o INPUT);" without quotes.  
    //CLEANING: For safety, it is important to clean used outputs with the purpose that they are turned off at the  
    //beginning of the program. Use the function "digitalWrite(PIN,LOW);" without quotes.  
    //COMMUNICATIONS: For example, for communications with the computer, use the function "Serial.begin(BAUDIOS);" without quotes.  
}
```

Infinite loop (Main program - Execution):

```
void loop() {  
    //Main program  
}
```

Note: Each line of Arduino takes approximately **63 nanoseconds** to execute (16 MHz Crystal)

Arduino IDE

Software download: <http://arduino.cc>

Compile

Load program
to Arduino

Save

Serial Monitor

Code Editor

Compiler error log

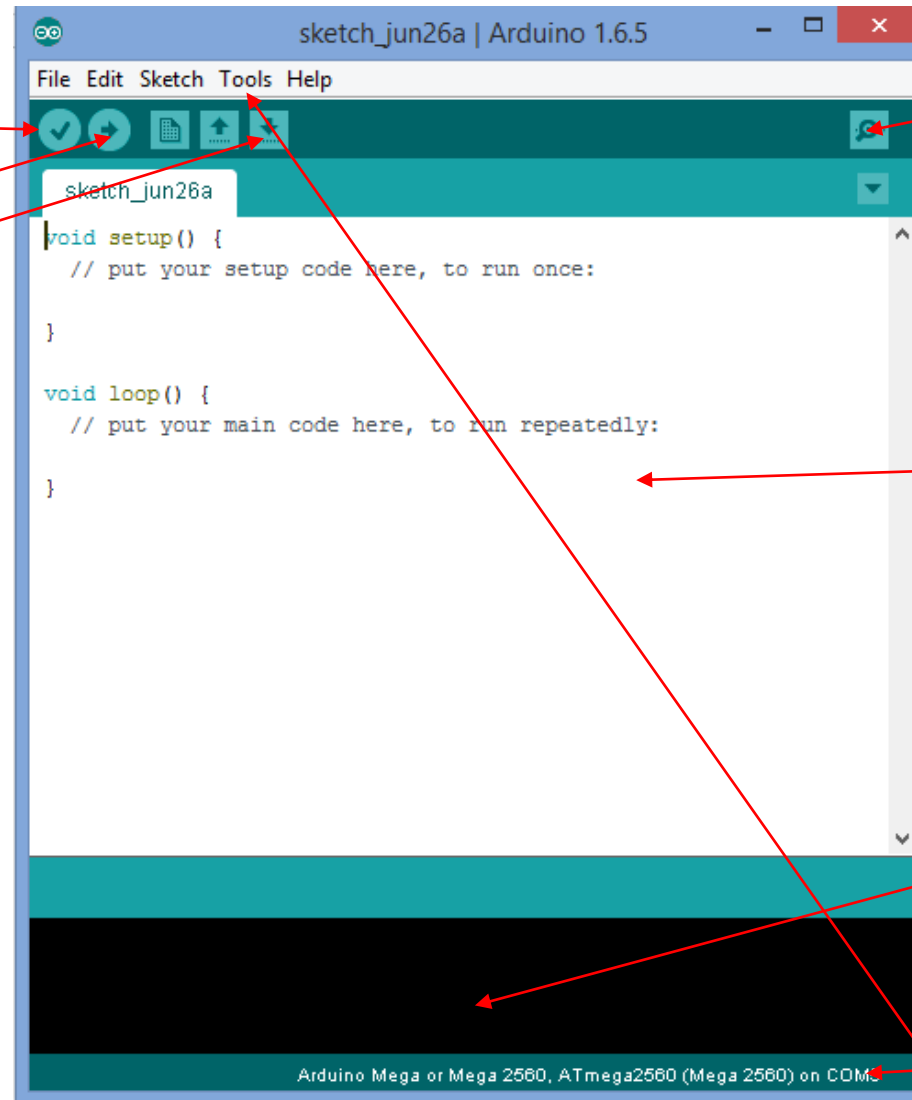
Selected ARDUINO board
and COM port
**Change it via Tools Menu*

Useful hotkeys:

Ctrl + U Compiles and uploads the code.

Ctrl + T Auto-indents the code.

For Mac use CMD instead of Ctrl key



Arduino common used commands

- Arduino functions are declared using Camel Case: First letter starts in minuscules and when another word is written starts with capital letter. E.g. iPod, getMode, etc.

- **pinMode**

- Configures the specified pin as input or output
- Syntax: **pinMode**(pin, mode);
 - pin: The pin # that will be configured
 - mode: Determines if the pin is an input or an output. Receives **INPUT** or **OUTPUT**



- **digitalWrite**

- Writes a logical state to an output pin: a HIGH logic state (5V) or a LOW logic state (0V)
- Syntax: **digitalWrite**(pin, value);
 - pin: The pin # that will be written
 - value: **HIGH** or **LOW**

- **digitalRead**

- Reads and returns the logic state value of a digital input pin
- Syntax: **digitalRead**(pin)
 - pin: The input pin # that will be read
 - Returns **HIGH** or **LOW** depending on the logic state value of the input pin that was read

- **delay**

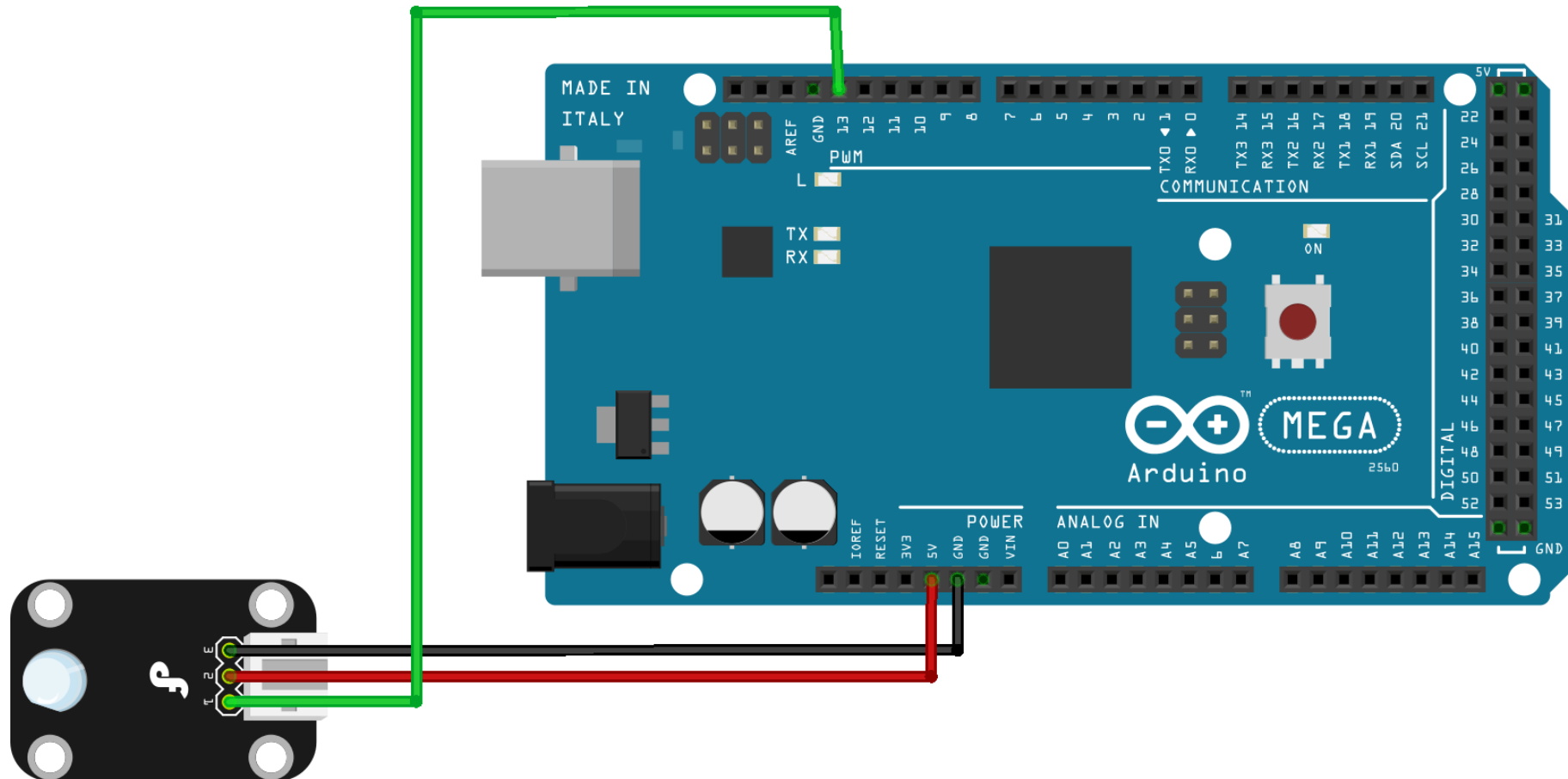
- Pauses the program execution for a desired time (in milliseconds)
- Not recommended to use because it pauses the whole program execution for the input time
- Syntax: **delay**(ms) ;
 - ms: The number of milliseconds that is desired to pause the program (var type: **unsigned long**)

Arduino Serial Monitor commands

- Serial is a **communications protocol** that can be used to program Arduino.
- For **debugging the code**, use the “**Serial Monitor**” by **printing text** or **variables** in Arduino code. Here are some commands to use the Serial Monitor:
- Serial.begin**
 - Initializes** Serial communication at a desired **speed**.
 - Syntax: **Serial.begin**(bauds);
 - bauds: Baud rate or speed to transfer data. Default value is 9600 bauds
- Serial.print**
 - Prints** specified **variable** or **text** through serial communication.
 - Syntax: **Serial.print**(val);
 - val: Value to print, can be a variable or a text using quote marks “Hello World”.
 - If used multiple times it will concatenate the next variable or text to the previous printed data.
- Serial.println**
 - Prints and change line** specified **variable** or **text** through serial communication.
 - Syntax: **Serial.println**(val);
 - val: Value to print, can be a variable or a text using quote marks “Hello World”.
- Serial.available**
 - Returns** the **number of bits** available (**incoming**) to be read from the Serial port.
 - Syntax: **Serial.available**();
 - Commonly used as a trigger function to read an input command from Serial Port.
- Serial.read**
 - Returns** the **next character available** from the **Serial Port buffer**
 - Syntax: **Serial.read**();
 - Commonly used to read incoming data.
 - Use multiple times to read all the buffer incoming data.
- Serial.readBytesUntil**
 - Function to **read multiple incoming bytes** from Serial Port buffer
 - Syntax:
Serial.readBytesUntil(terchar,arrayToStore,numBytes);
 - terchar: termination character to end reading. It's commonly used the '\n' which is the end of line.
 - arrayToStore: array variable to store incoming read buffer.
 - numByres: maximum bytes to read.

Example 1.1A – Arduino common used commands

- ▶ In PIN 13 there is a LED (L1) connected. Blink the LED $\frac{1}{2}$ second ON and $\frac{1}{2}$ second OFF.
 1. Connect the LED (L1 in the Arduino code) in PIN 13 (cf Picture).
 2. Execute the code of the next slide to blink the LED $\frac{1}{2}$ sec. ON and $\frac{1}{2}$ sec. OFF.



```
//I/O pin labeling
#define L1 13 //Label LED connected in pin 13 as "L1"

//Constant declaration
unsigned long TBLINK = 500; //Blink constant TBLINK initialized on
                             //500 ms

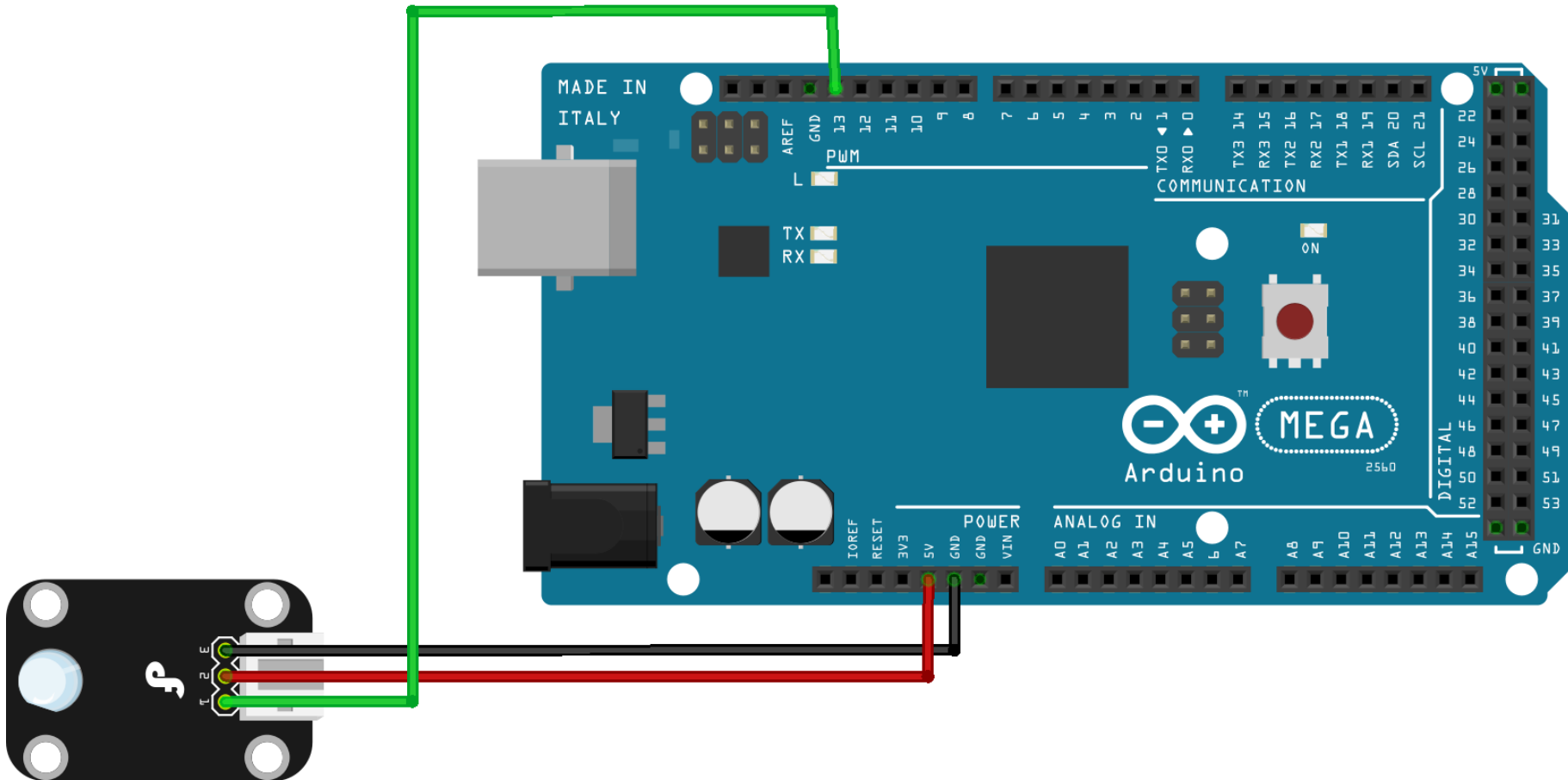
void setup() {
    //I/O Pin Configuration
    pinMode(L1, OUTPUT); //Set pin L1 as Output

    //Output cleaning
    digitalWrite(L1, LOW); //Turn OFF L1
}

void loop() {
    digitalWrite(L1, HIGH); //Turn ON L1
    delay(TBLINK); //Delay of TBLINK milliseconds(500 ms)
    digitalWrite(L1, LOW); //Turn OFF L1
    delay(TBLINK); //Delay of TBLINK milliseconds(500 ms)
}
```

Example 1.1 B – Arduino common used commands

- In PIN 13 there is a LED (L1) connected. Blink the LED 1/2 second ON and 1/2 second OFF. Print the LED status on the Serial Monitor.
1. Connect the LED (L1 in the Arduino code) in PIN 13 (cf Picture).
 2. Execute the code of the next slide to blink the LED 1/2 sec. ON and 1/2 sec. OFF.
 3. Print the current LED status using the Serial Monitor



```
//I/O pin labeling
#define L1 13 //Label LED connected in pin 13 as "L1"

//Constant declaration
unsigned long TBLINK = 500; //Blink constant TBLINK initialized on
                             //500 ms

void setup() {
    //I/O Pin Configuration
    pinMode(L1, OUTPUT); //Set pin L1 as Output

    //Output cleaning
    digitalWrite(L1, LOW); //Turn OFF L1

    //Communications
    Serial.begin(9600); //Start Serial communications with PC at 9600 bauds
}

void loop() {
    digitalWrite(L1, HIGH); //Turn ON L1
    Serial.println("LED ON"); // (Debug) Print current LED status
    delay(TBLINK); //Delay of TBLINK milliseconds(500 ms)
    digitalWrite(L1, LOW); //Turn OFF L1
    Serial.println("LED OFF"); // (Debug) Print current LED status
    delay(TBLINK); //Delay of TBLINK milliseconds(500 ms)
}
```


If statement

- Used in conjunction with a comparison operator.
- Tests if a condition is met, and in the positive case, executes desired actions and then it continues with the program.
- Syntax:

```
if (condition) {  
    //Do something here  
}  
else if (othercondition) {  
    //Do something else if the first condition wasn't met but the othercondition was met  
}  
else {  
    //Do something here in other case  
}
```

- Example with digital input pins

```
if (digitalRead(pin) == HIGH) {  
    //Do something here if the current pin  
    //logic state is HIGH  
}
```

- Example with internal variables

```
if (temperature > 25) {  
    //Do something here if temperature is  
    //greater than 25 degree Celsius  
}
```

Common **error** using if statement:

```
if (pin == HIGH) {
```



The if statement needs to ask if the actual **PIN STATE** is **HIGH**.
Not like shown at the left, where it's asked if the **PIN NUMBER** is
equals to **HIGH**.

Switch statement

- ▶ Allows to do different actions depending of different variable values.
- ▶ Similar to do multiple if..else if statements for the same var but with different values.
- ▶ Each case is the possible value that the variable can have and it's ended with a break;
- ▶ Sintax:

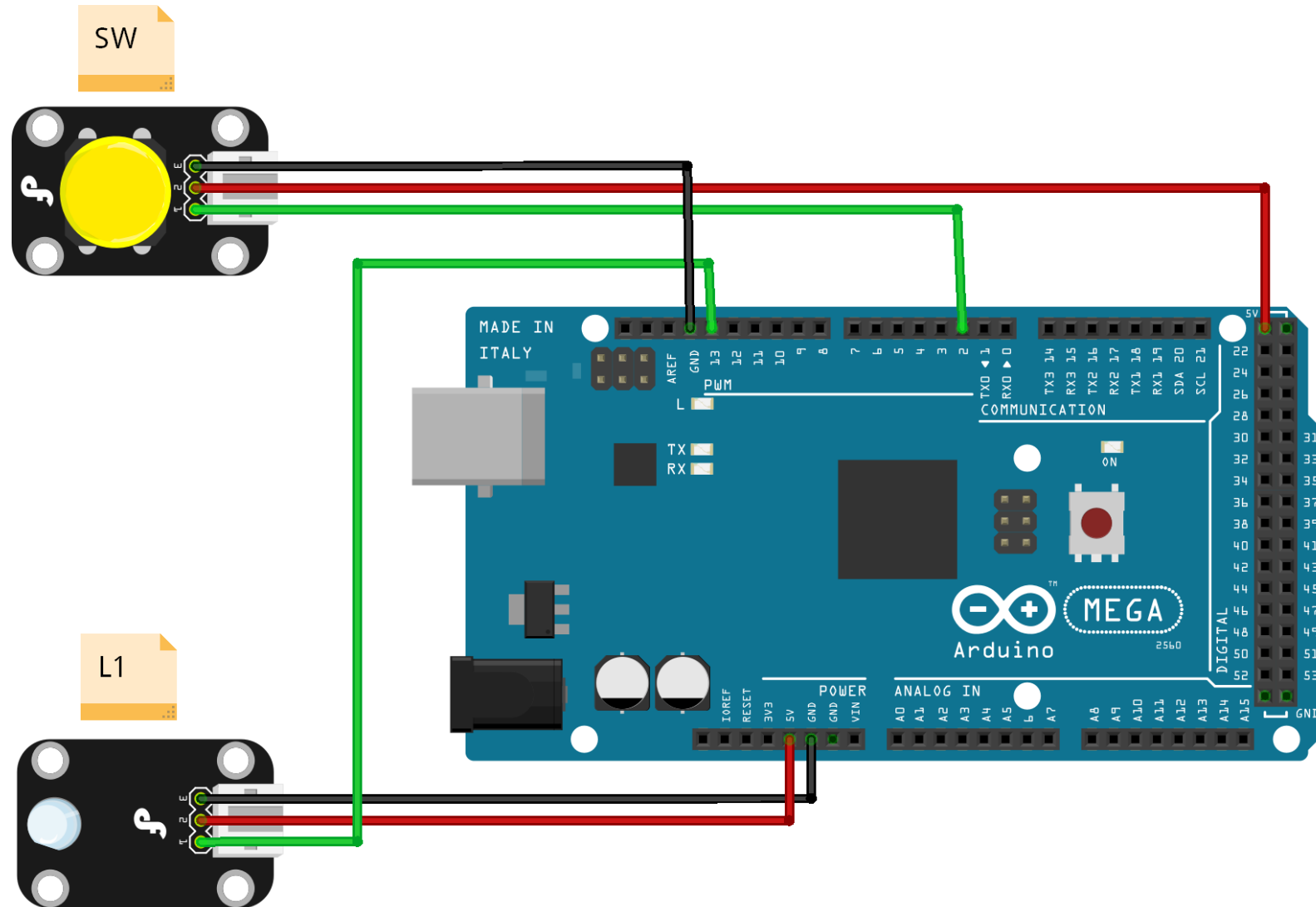
```
switch (var) {  
    case 0:  
        //Do something here if var is equal to zero  
        break;  
    case 1:  
        //Do something here if var is equal to one  
        break;  
    case 2:  
        //Do something here if var is equal to two  
        break;  
}
```

- ▶ It is possible to check the case with labels instead of raw numbers predefined at the beginning of the program with #define.

```
switch (var) {  
    case label1:  
        //Do something here if var is equal to the label1  
        break;  
    case label2:  
        //Do something here if var is equal to the label2  
        break;  
}
```

Example 1.2 – If statement with digital input

- Example: In the PIN 2 there is a button or switch (SW) and in the PIN 13 there is a LED (L1). Turn ON the LED if the switch is activated, in other case, turn off the LED



Example 1.2 – If statement with digital input

```
//I/O pin labeling
#define SW 2 //Switch "SW" connected on pin 2
#define L1 13 //LED "L1" connected on pin 13

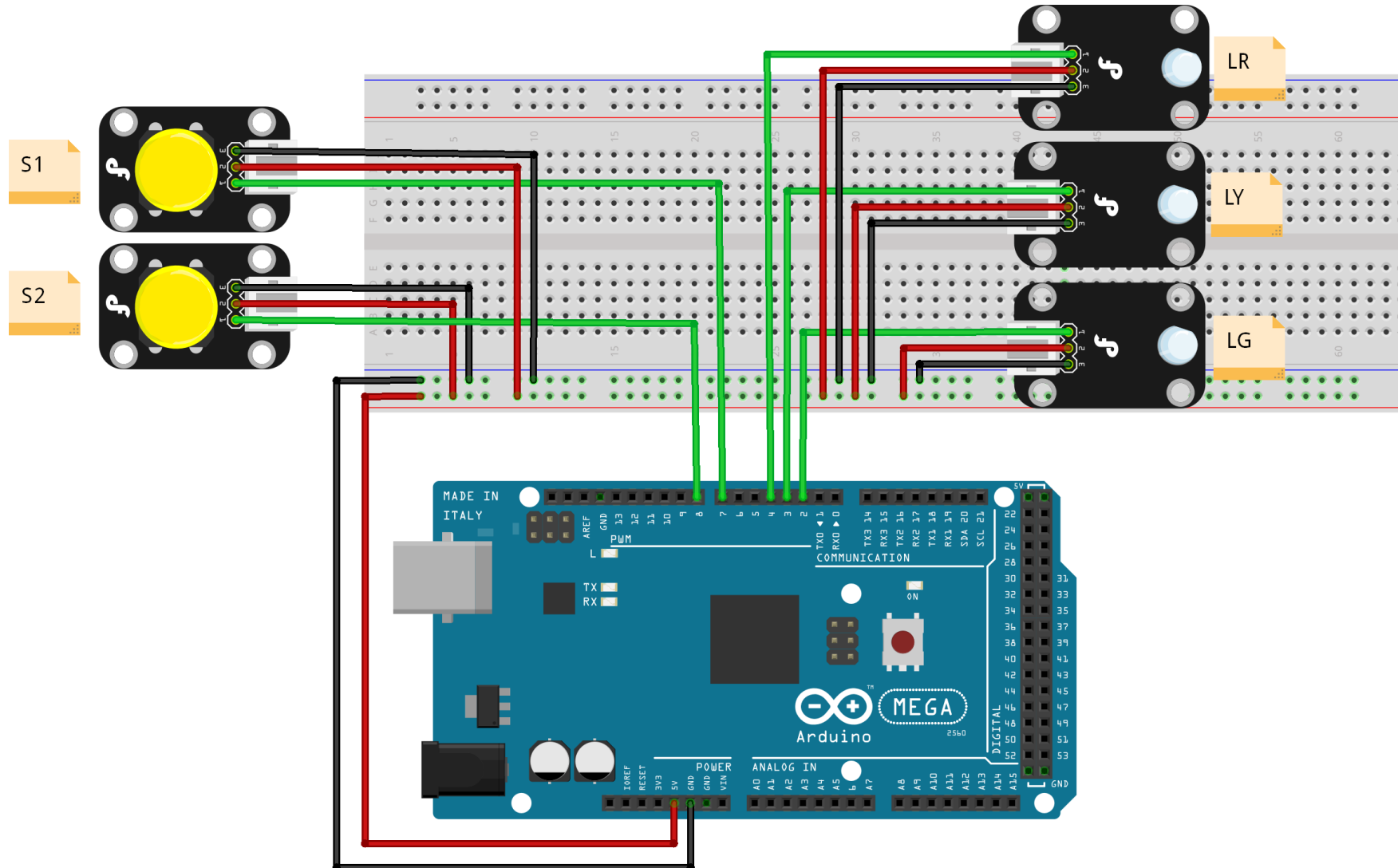
void setup() {
  //I/O Pin Configuration
  pinMode(SW, INPUT); //SW as INPUT
  pinMode(L1, OUTPUT); //L1 as OUTPUT

  //Output cleaning
  digitalWrite(L1, LOW); //Turn OFF L1
}

void loop() {
  if (digitalRead(SW) == HIGH) { //Check if SW is in logic state HIGH
    digitalWrite(L1, HIGH); //If it is, turn ON L1
  }
  else { //In other case
    digitalWrite(L1, LOW); //If the SW is OFF, turn OFF L1
  }
}
```

Example 1.3 – Traffic light

- Write a program that controls a traffic light with 2 maintenance switches. If both switches are activated, the Red light will blink. In other case the traffic light will work normally (Red, Yellow, Green).



```
//I/O pin labeling
#define LR 4 //Red LED "LR" connected on digital pin 4
#define LY 3 //Yellow LED "LY" connected on digital pin 3
#define LG 2 //Green LED "LG" connected on digital pin 2
#define S1 7 //Switch "S1" connected on digital pin 7
#define S2 8 //Switch "S2" connected on digital pin 8

//Constant declaration
const unsigned long TRV = 5000; //Time constant from Red to Green
initialized on 5000 ms
const unsigned long TVA = 2500; //Time constant from Green to Yellow
initialized on 2500 ms
const unsigned long TAR = 1000; //Time constant from Yellow to Red
initialized on 1000 ms

const unsigned long TIT = 5000; //Time constant for blinking initialized on
5000 ms

void setup() {
  //Pin configuration
  pinMode(LR, OUTPUT); //LR as output
  pinMode(LY, OUTPUT); //LY as output
  pinMode(LG, OUTPUT); //LG as output
  pinMode(S1, INPUT); //S1 as input
  pinMode(S2, INPUT); //S2 as input

  //Physical Output Cleaning
  digitalWrite(LR, LOW); //Turn off LR
  digitalWrite(LY, LOW); //Turn off LY
  digitalWrite(LG, LOW); //Turn off LG
}
```

```
void loop() {
  if (digitalRead(S1) == HIGH && digitalRead(S2) == HIGH) { //If both
maintenance switchs are ON, blink
    //Turn OFF all LEDs
    digitalWrite(LR, LOW);
    digitalWrite(LY, LOW);
    digitalWrite(LG, LOW);
    delay(TIT); //Delay of TIT msecs (5000msecs)

    //Turn ON red led
    digitalWrite(LR, HIGH);
    digitalWrite(LY, LOW);
    digitalWrite(LG, LOW);
    delay(TIT); //Delay of TIT msecs (5000msecs)
  }
  else { //In other case
    //Normal traffic light sequence
    digitalWrite(LR, HIGH);
    digitalWrite(LY, LOW);
    digitalWrite(LG, LOW);
    delay(TRV); //Delay of TRV msecs (5000msecs)

    digitalWrite(LR, LOW);
    digitalWrite(LY, LOW);
    digitalWrite(LG, HIGH);
    delay(TVA); //Delay of TVA msecs (2500msecs)

    digitalWrite(LR, LOW);
    digitalWrite(LY, HIGH);
    digitalWrite(LG, LOW);
    delay(TAR); //Delay of TAR msecs (1000msecs)
  }
}
```

Thanks!