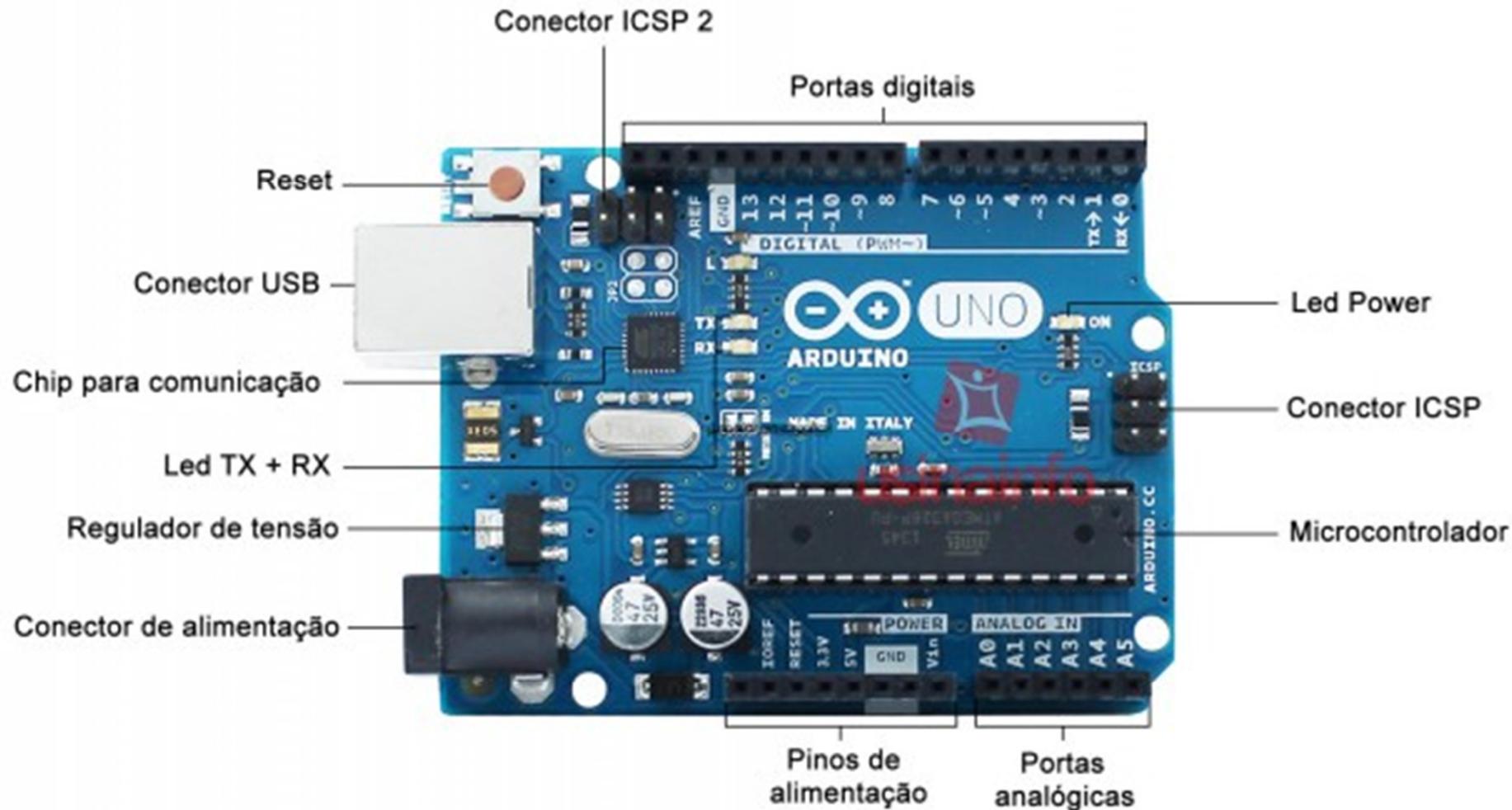




# Entrada/Saída Digitais

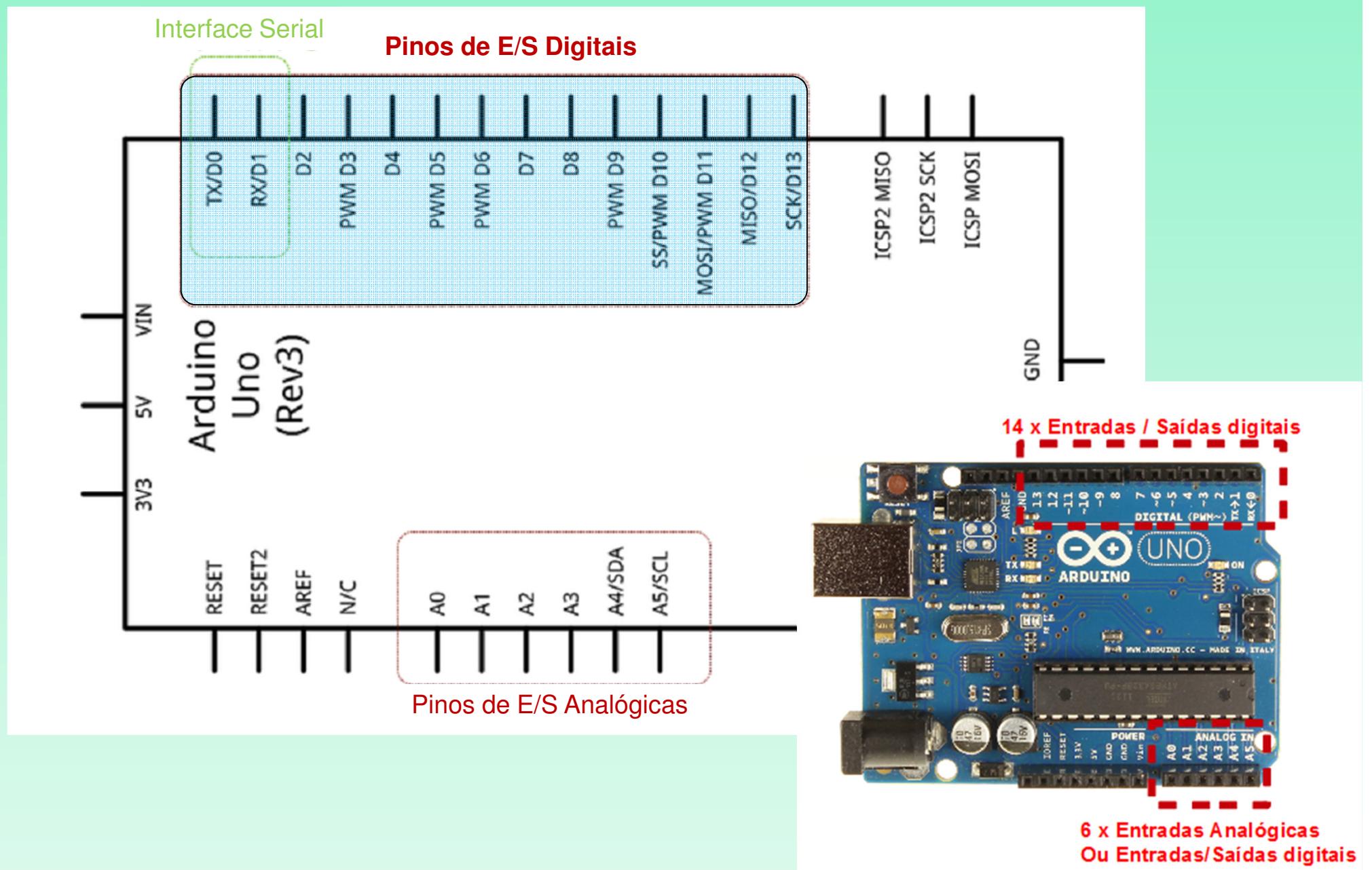
Aula 2

# Arduino UNO



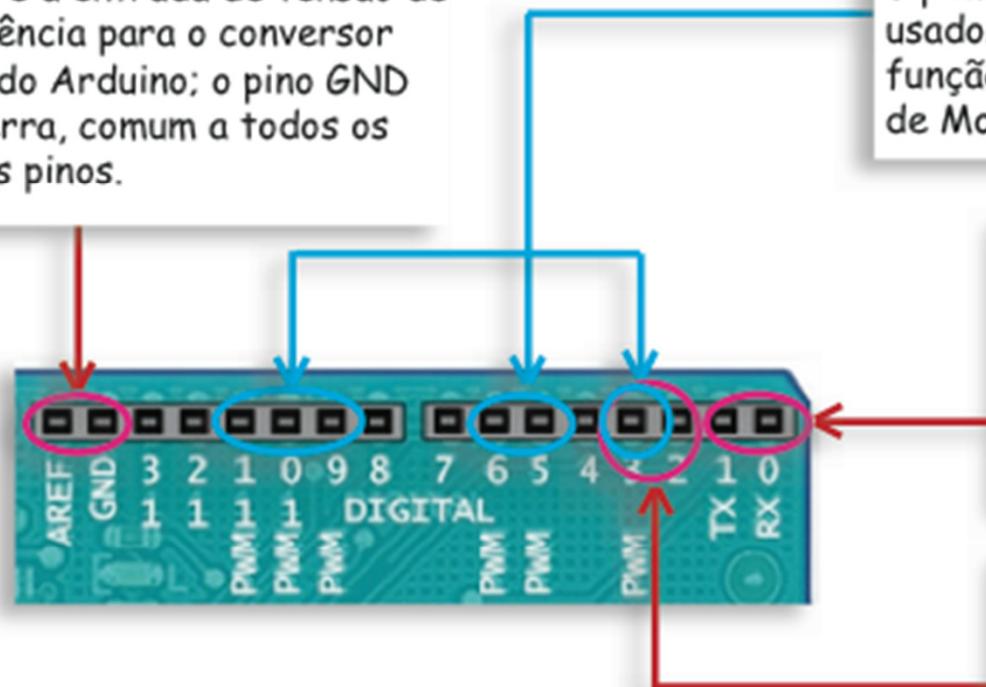
ICSP: In-Circuit Serial Programming

# HW



# HW

**Pinos AREF e GND:** o pino AREF é a entrada de tensão de referência para o conversor A/D do Arduino; o pino GND é o terra, comum a todos os outros pinos.



**Pinos 3, 5, 6, 9, 10 e 11 (PWM):** 6 pinos dos 14 pinos digitais podem ser usados para gerar sinais analógicos com a função `analogWrite( )` utilizando a técnica de Modulação por Largura de Pulso (PWM).

**Pinos 0 e 1:** os dois primeiros pinos digitais são conectados a USART do microcontrolador do Arduino para comunicação serial com um computador.

**Pinos 2 e 3:** pinos que chamam uma ISR (Interrupt Service Routine) para tratar uma interrupção com a função `attachInterrupt( )` nesses pinos.

# Ambiente de Desenvolvimento Integrado IDE

- Teste de Funcionamento do Arduino

- Carregue o arq. *Blink.pde*: *File > Sketchbook > Examples > Digital > Blink*
- Compile o código: *Sketch > Verify/Compile*
- Envie ao Arduino (*upload*): *File > Upload to I/O*
- Execução: Led vermelho piscando a cada segundo

Blink sketch

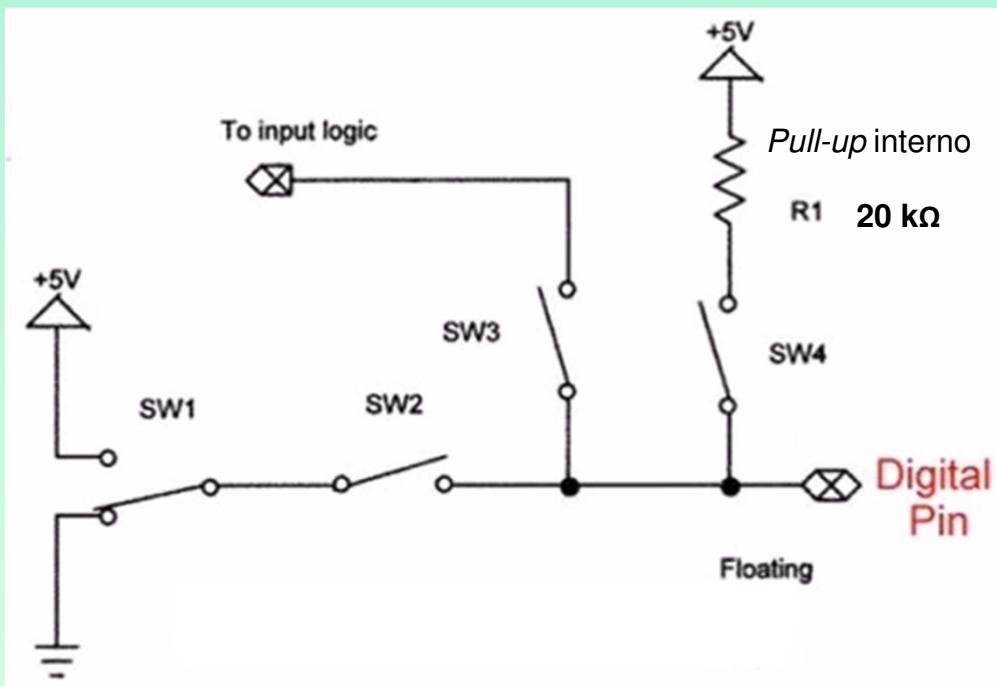
```
int pinoLED = 13;

void setup() {
    pinMode(pinoLED, OUTPUT);
}

void loop() {
    digitalWrite(pinoLED, HIGH);
    delay(500);
    digitalWrite(pinoLED, LOW);
    delay(500);
}
```

# Comandos Básicos

- Entrada/Saída Digital (pinos de 0 a 13)<sup>1</sup>
  - Somente valores HIGH (1) ou LOW (0)
  - Cada pino pode ser configurado como Entrada (INPUT) ou Saída (OUTPUT), não simultaneamente – default: INPUT



Após *power-on* os pinos do Arduino estarão todos em alta impedância,  $> 100\text{ M}\Omega$ .

Leitura de pinos digitais abertos (sem conexão) resultará em valores HIGH/LOW aleatórios (ruído do ambiente ou acoplamento capacitivo)

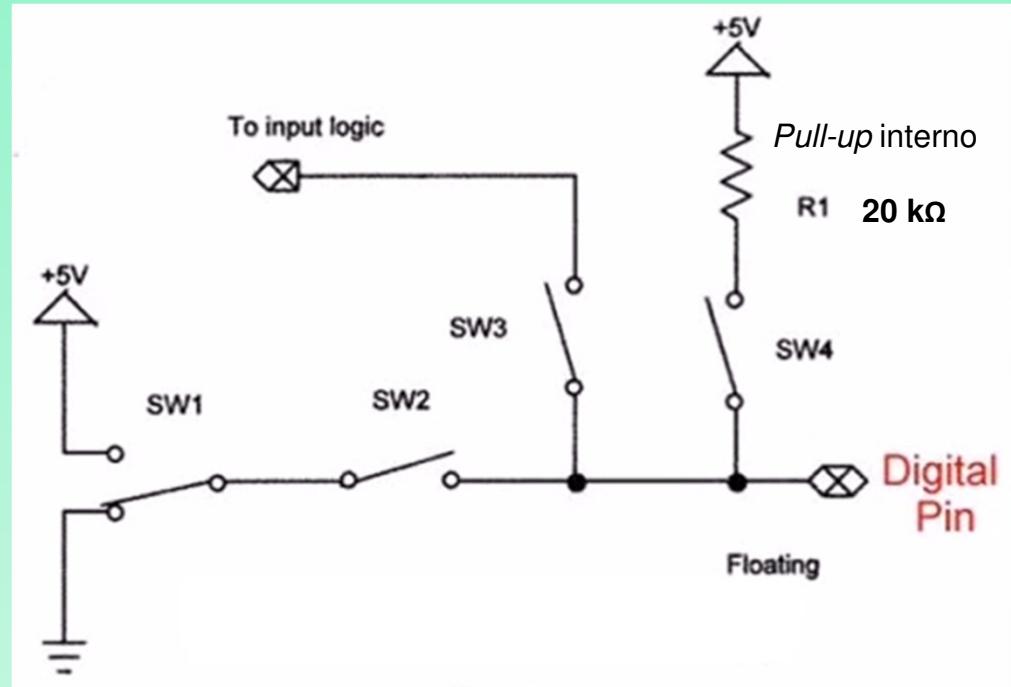
Conekte um pino de entrada num estado conhecido se nenhuma entrada estiver presente: *pull-up* ou *pull-down*

A função `pinMode(pino,estado)` configura os pinos como entrada ou saída, estado: `INPUT`, `OUTPUT` ou `INPUT_PULLUP`

<sup>1</sup> Se os pinos 0 e 1 (TX/RX) forem usados para comunicação serial então não poderão ser usados para E/S Digital

# Comandos Básicos

- Entrada/Saída Digital
  - Configurações possíveis



Estado	SW1	SW2	SW3	SW4
INPUT	X	Aberta	Fechada	Aberta
INPUT_PULLUP	X	Aberta	Fechada	Fechada
OUTPUT	LOW – GND HIGH – Vcc	Fechada	Aberta	Aberta

# Comandos Básicos

- Entrada/Saída Digital (pinos de 0 a 13)
  - Função para configuração de Pinos Digitais
    - **pinMode**(numeroPino, estadoPino);
  - Exemplo:
    - Entrada:                      `pinMode(13, INPUT);`
    - Saída:                        `pinMode(13, OUTPUT);`
    - Entrada com *pull-up*:      `pinMode(13, INPUT_PULLUP);`
  - Função para realizar uma **Saída Digital**
    - **digitalWrite**(numeroPino, HIGH/LOW);

# Comandos Básicos

- Entrada/Saída Digital (pinos de 0 a 13)
  - Função para realizar uma **Entrada**
    - var = **digitalRead**(numeroPino) ;
  - Se o pino estiver aberto (sem nível definido) então a função pode retornar HIGH ou LOW (indefinido), e esse valor pode mudar randomicamente

```
val = digitalRead(pinoEnt); // lê o pino de Entrada  
digitalWrite(pinoLed, val); // ajusta LED ao valor da chave
```

# Exemplo de Sketch

```
/*  
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  
}
```

Prepara o pino 13 para saída de tensões

Coloca o pino 13 com nível alto de tensão (5V) = acende o LED

Diz ao microcontrolador para ficar inerte durante 1000ms = 1s

Coloca o pino 13 com nível baixo de tensão (0V) = desliga o LED

# Exemplo de Sketch

```
/*
  DigitalReadSerial: Lê uma entrada digital no pino 2, e
                      mostra o resultado no Monitor Serial
*/

void setup() {
  Serial.begin(9600);
  pinMode(2, INPUT);
}

void loop() {
  int valorSensor = digitalRead(2);
  Serial.println(valorSensor, DEC);
  delay(1000);
}
```

<http://arduino.cc/en/Reference/Serial>  
<http://arduino.cc/en/Reference/PinMode>  
<http://arduino.cc/en/Reference/DigitalRead>  
<http://arduino.cc/en/Serial/Println>

**DEC** - decimal  
**HEX** - hexadecimal  
**OCT** - octal  
**BIN** - binário

# Exemplo de Sketch

```
/* Pisca-pisca sem espera ocupada: liga/desl LED sem usar delay().  
O uC pode cuidar de outras tarefas enquanto o tempo desejado é alcançado  
Circuito: LED ligado ao pino 13 e ao terra (GND)  
* Obs: na maioria dos Arduinos já existe um LED na placa conectado ao  
pino 13, assim nenhum HW é necessário para este exemplo */  
const int pinoLED = 13;           // número do pino do LED  
byte EstadoLED = LOW;           // estado do LED (LOW ou HIGH)  
long tempoAnt = 0;              // último tempo que o LED foi atualizado  
long intervalo = 1000;          // intervalo p/ piscar o LED (miliseg.)  
  
void setup() {  
    pinMode(pinoLED, OUTPUT);    // conf. pino digital como saída  
}  
  
void loop() {  
    // coloque aqui o código que precisa rodar o tempo todo...  
    unsigned long tempoAtual = millis();  
    if(tempoAtual - tempoAnt > intervalo) {  
        tempoAnt = tempoAtual;      // salva tempo que o LED piscou  
        if (EstadoLED == LOW) {      // inverte o estado anterior do LED  
            EstadoLED = HIGH;  
        } else  
            EstadoLED = LOW;  
        digitalWrite(pinoLED, EstadoLED);  
    }  
}
```

EstadoLED = !EstadoLED;

# Prática 1

# Prática 1

Piscar LED ligado ao pino digital número 10:

$$I_{\text{máx}} = 20 \text{mA} / \text{pino}$$
$$R = V/I = 5 / 0,02 = 150 \Omega$$

	Material	Conexões
Protopboard		
LED Verm.		
Res. 150Ω		
Jumpers		

# Prática 1

Piscar um LED ligado ao pino digital número 10

## Código

```
int ledPin = 10;

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

void loop() {
    digitalWrite(ledPin, HIGH);
    delay(1000);
    digitalWrite(ledPin, LOW);
    delay(1000);
}
```

### Anatomy of a C function

Datatype of data returned,  
any C datatype.

"void" if nothing is returned.

Parameters passed to  
function, any C datatype.

```
Function name
int myMultiplyFunction(int x, int y){
    int result;
    result = x * y;
    return result;
} ← Curly braces required.
```

# **Prática 2**

# Prática 2

- Piscar LED ligado ao pino digital número 10 para mostrar uma mensagem em código Morse: “S.O.S”



	Material	Conexões
Protopboard		
LED Verm.		
Res. 150Ω		
Jumpers		

```
int ledPin = 10;

// run once, when the sketch starts
void setup()
{
    // sets the digital pin as output
    pinMode(ledPin, OUTPUT);
}

// run over and over again
void loop()
{
    // 3 dits
    for (int x=0; x<3; x++) {
        digitalWrite(ledPin, HIGH);      // sets the LED on
        delay(150);                   // waits for 150ms
        digitalWrite(ledPin, LOW);     // sets the LED off
        delay(100);                  // waits for 100ms
    }

    // 100ms delay to cause slight gap between letters
    delay(100);
    // 3 dahs
    for (int x=0; x<3; x++) {
        digitalWrite(ledPin, HIGH);      // sets the LED on
        delay(400);                   // waits for 400ms
        digitalWrite(ledPin, LOW);     // sets the LED off
        delay(100);                  // waits for 100ms
    }

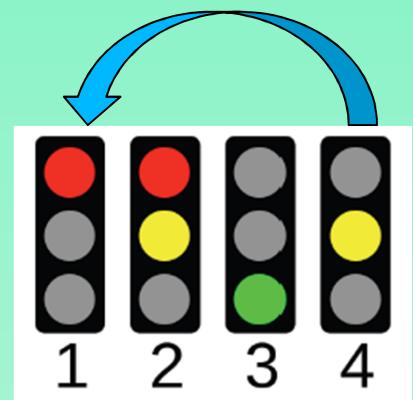
    // 100ms delay to cause slight gap between letters
    delay(100);
    // 3 dits again
    for (int x=0; x<3; x++) {
        digitalWrite(ledPin, HIGH);      // sets the LED on
        delay(150);                   // waits for 150ms
        digitalWrite(ledPin, LOW);     // sets the LED off
        delay(100);                  // waits for 100ms
    }

    // wait 5 seconds before repeating the SOS signal
    delay(5000);
}
```

# Prática 3

# Prática 3

- Simular um semáforo de 4 estágios, usando LEDs
  - 1 seg de intervalo nas trocas: 2→3 e 4→1
  - 5 seg nas trocas: 1→2 e 3→4



Conexões

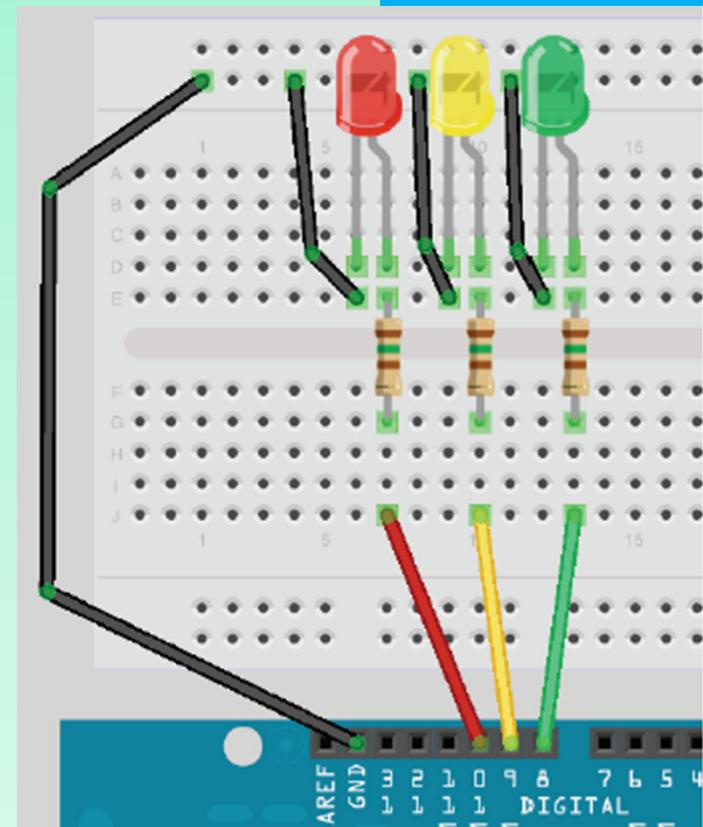
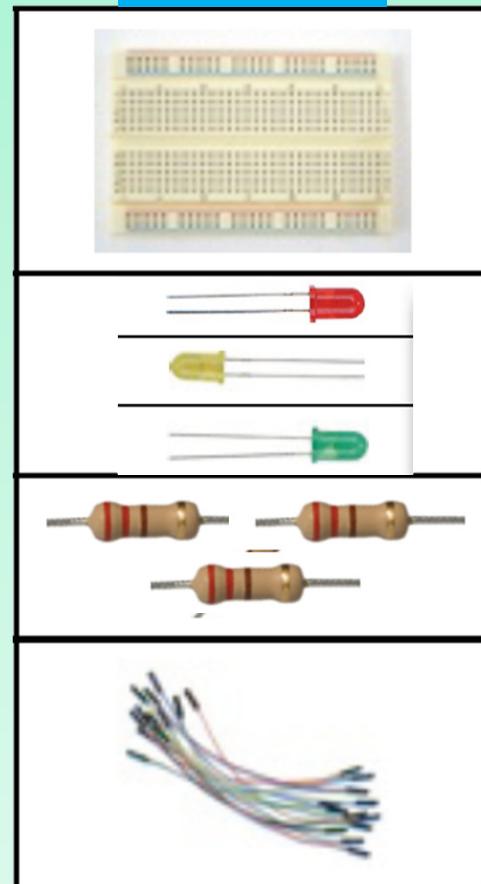
Material

Protopboard

LEDs Verm.,  
Amar., Verde

3 Res. 220Ω

Jumpers



```
int ledDelay = 10000; // delay in between changes
int redPin = 10;
int yellowPin = 9;
int greenPin = 8;

void setup() {
    pinMode(redPin, OUTPUT);
    pinMode(yellowPin, OUTPUT);
    pinMode(greenPin, OUTPUT);
}

void loop() {

    // turn the red light on
    digitalWrite(redPin, HIGH);
    delay(ledDelay); // wait 5 seconds

    digitalWrite(yellowPin, HIGH); // turn on yellow
    delay(2000); // wait 2 seconds

    digitalWrite(greenPin, HIGH); // turn green on
    digitalWrite(redPin, LOW); // turn red off
    digitalWrite(yellowPin, LOW); // turn yellow off
    delay(ledDelay); // wait ledDelay milliseconds

    digitalWrite(yellowPin, HIGH); // turn yellow on
    digitalWrite(greenPin, LOW); // turn green off
    delay(2000); // wait 2 seconds

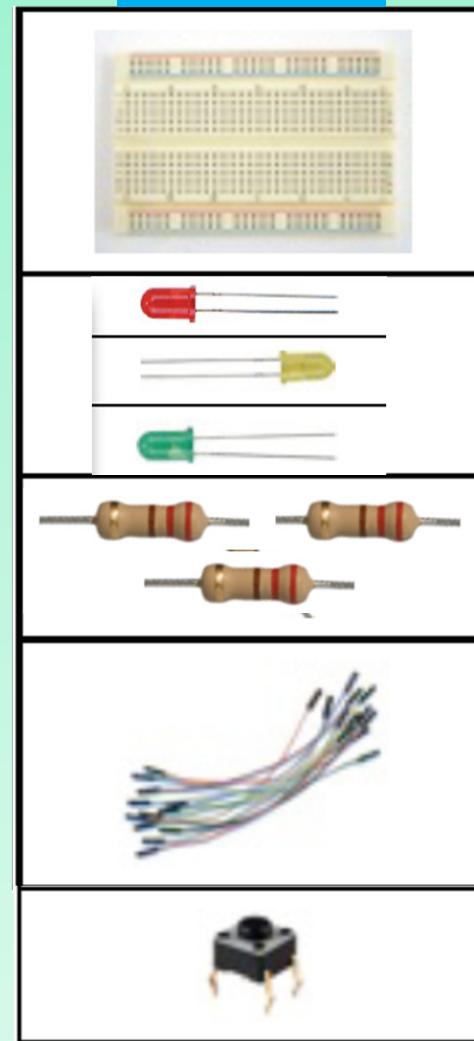
    digitalWrite(yellowPin, LOW); // turn yellow off
}
```

# Prática 4

# Prática 4

- Idem à prática anterior, incluindo semáforo para pedestres
- Interação com o Arduino
  - Pedestre aperta botoeira e aguarda pelo sinal verde para atravessar a rua
  - Intervalo mínimo entre intervenções de pedestres: 5 seg
  - Em cada intervenção de pedestre
    - Sinal verde para veículos se apaga
    - Sinal amarelo para veículos se acende por 1 seg
    - Sinal vermelho para veículos se acende
    - Por segurança, espere mais 1 seg
    - Sinal verde para pedestre se acende por *crosstime*, e pisca por 5 seg e se apaga
    - Sinal vermelho para pedestre se acende
    - Sinal vermelho para veículos se apaga e o verde para veículos se acende

## Material



Protopboard

2 LEDs Verm.,  
1 LED Amar.,  
2 LEDs Verde

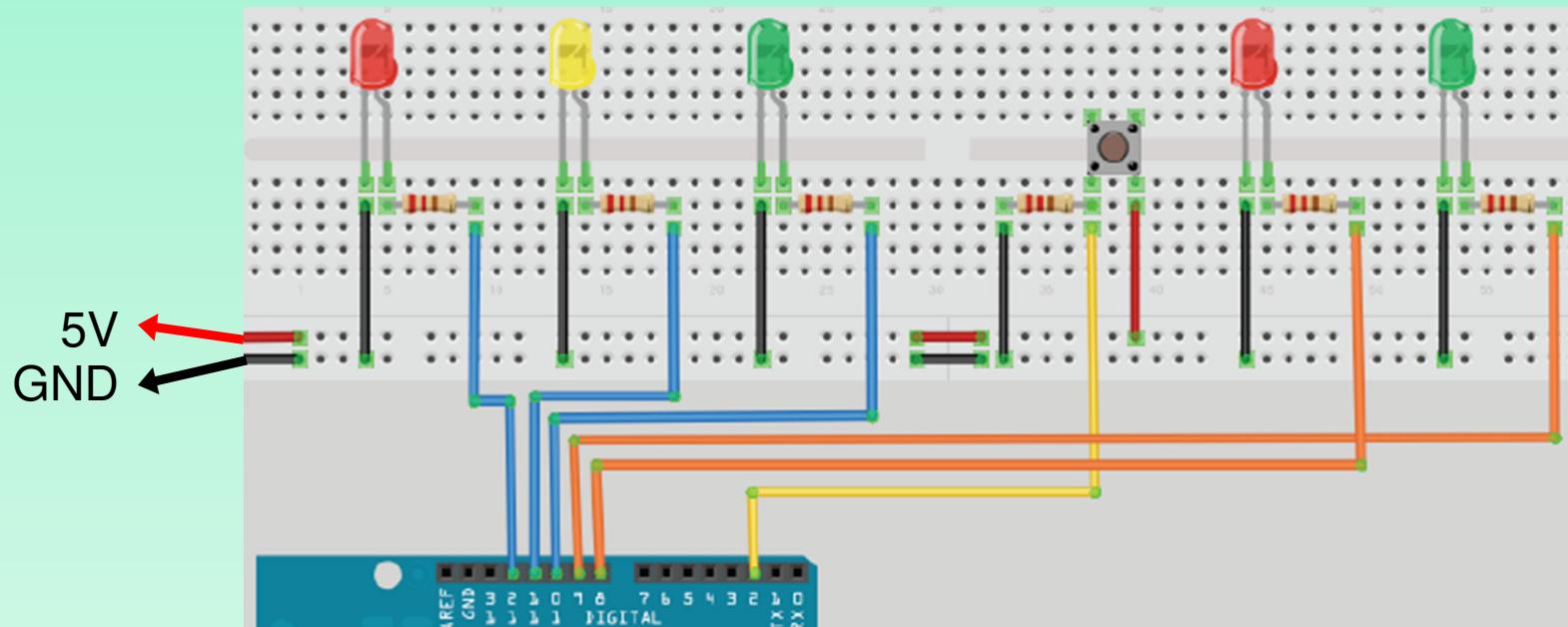
Res. 150Ω

Jumpers

Chave  
push-button

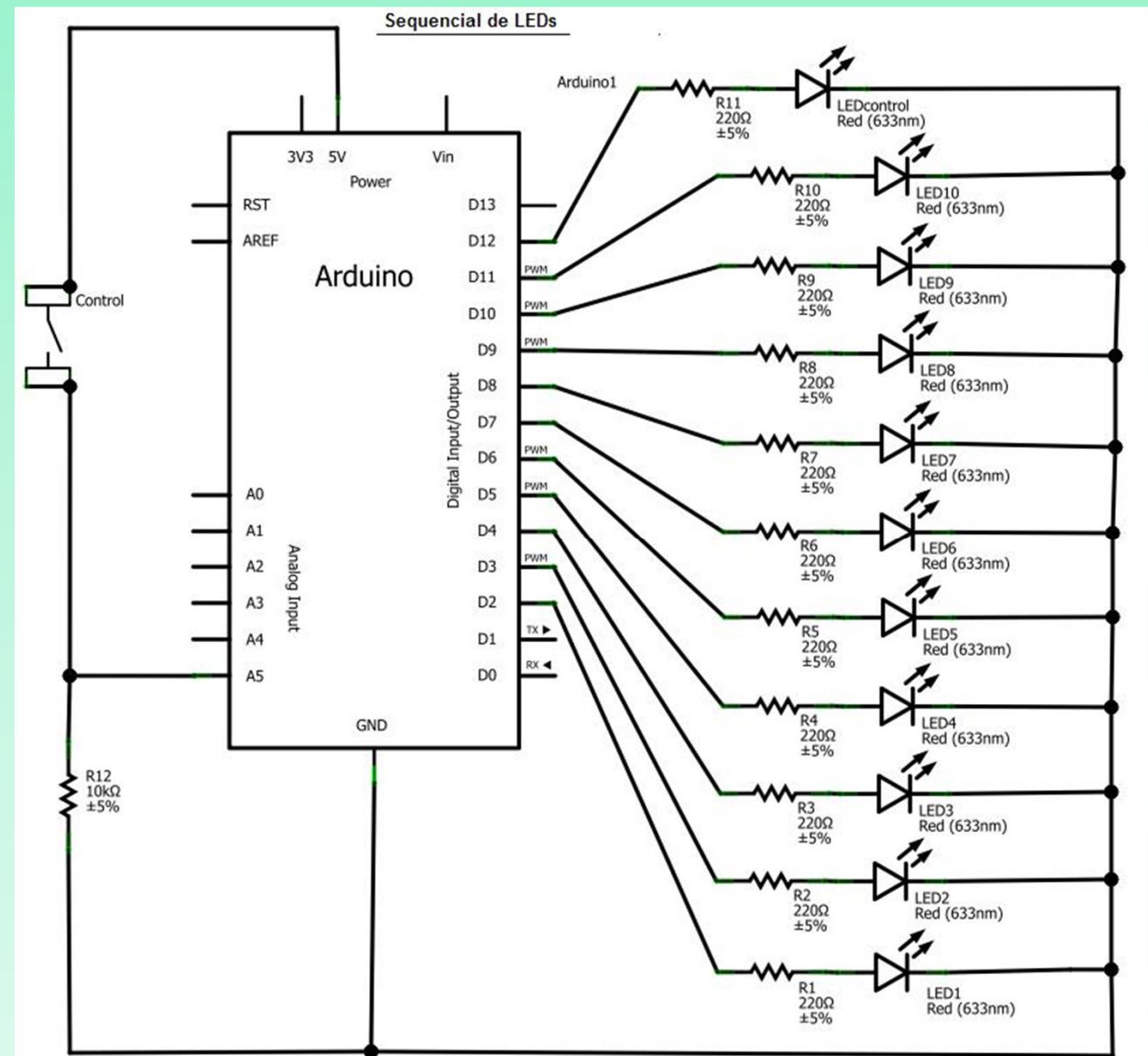
# Prática 4

## Conexões



# Prática 4

## Esquemático



## Prática 4

```
int carRed = 12; // assign the car lights
int carYellow = 11;
int carGreen = 10;
int pedRed = 9; // assign the pedestrian lights
int pedGreen = 8;
int button = 2; // button pin
int crossTime = 5000; // time allowed to cross
unsigned long changeTime; // time since button pressed

void setup() {
    pinMode(carRed, OUTPUT);
    pinMode(carYellow, OUTPUT);
    pinMode(carGreen, OUTPUT);
    pinMode(pedRed, OUTPUT);
    pinMode(pedGreen, OUTPUT);
    pinMode(button, INPUT); // button on pin 2
    // turn on the green light
    digitalWrite(carGreen, HIGH);
    digitalWrite(pedRed, HIGH);
}

void loop() {
    int state = digitalRead(button);
    /* check if button is pressed and it is
    over 5 seconds since last button press */
    if (state == HIGH && (millis() - changeTime) > 5000) {
        // call the function to change the lights
        changeLights();
    }
}
```

```
void changeLights() {
    digitalWrite(carGreen, LOW); // green off
    digitalWrite(carYellow, HIGH); // yellow on
    delay(2000); // wait 2 seconds

    digitalWrite(carYellow, LOW); // yellow off
    digitalWrite(carRed, HIGH); // red on
    delay(1000); // wait 1 second till its safe

    digitalWrite(pedRed, LOW); // ped red off
    digitalWrite(pedGreen, HIGH); // ped green on
    delay(crossTime); // wait for preset time period

    // flash the ped green
    for (int x=0; x<10; x++) {
        digitalWrite(pedGreen, HIGH);
        delay(250);
        digitalWrite(pedGreen, LOW);
        delay(250);
    }
    // turn ped red on
    digitalWrite(pedRed, HIGH);
    delay(500);

    digitalWrite(carYellow, HIGH); // yellow on
    digitalWrite(carRed, LOW); // red off
    delay(1000);
    digitalWrite(carGreen, HIGH);
    digitalWrite(carYellow, LOW); // yellow off

    // record the time since last change of lights
    changeTime = millis();
    // then return to the main program loop
}
```

# Tipos de Dados

Data type	RAM	Number Range
<u>void keyword</u>	N/A	N/A
<u>boolean</u>	1 byte	0 to 1 (True or False)
<u>byte</u>	1 byte	0 to 255
<u>char</u>	1 byte	-128 to 127
<u>unsigned char</u>	1 byte	0 to 255
<u>int</u>	2 byte	-32,768 to 32,767
<u>unsigned int</u>	2 byte	0 to 65,535
<u>word</u>	2 byte	0 to 65,535
<u>long</u>	4 byte	-2,147,483,648 to 2,147,483,647
<u>unsigned long</u>	4 byte	0 to 4,294,967,295
<u>float</u>	4 byte	-3.4028235E+38 to 3.4028235E+38
<u>double</u>	4 byte	-3.4028235E+38 to 3.4028235E+38
<u>string</u>	1 byte + x	Arrays of chars
<u>array</u>	1 byte + x	Collection of variables

Atmega168 = 1kB de SRAM  
 Atmega328 = 2kB de SRAM

Tipo	Tam. (bits)	Tam. (bytes)	Mínimo	Máximo
<b>byte</b>	8	1	0	255
<b>unsigned int</b>	16	2	0	65535
<b>int</b>	16	2	-32768	32767
<b>unsigned long</b>	32	4	0	4294967295
<b>long</b>	32	4	-2147483648	2147483647

Operadores Lógicos:

&&	Logical AND
	Logical OR
!	NOT

```

if (x==5 && y==10) {....}
if (x==5 || y==10) {....}
if (!x) {.....}
if (x==5 && (y==10 || z==25)) {..}

```