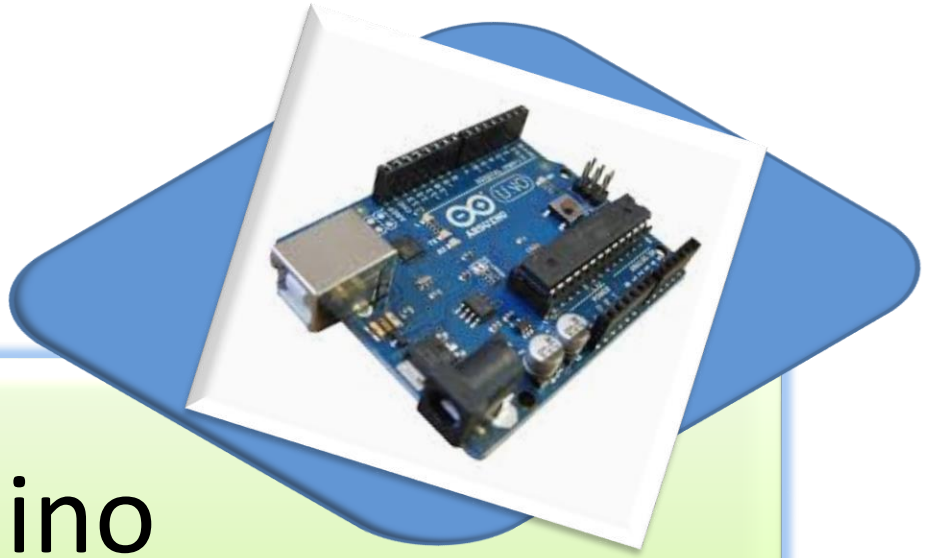


Arduino

Prof. Maurício Dias

Percepção e Ação

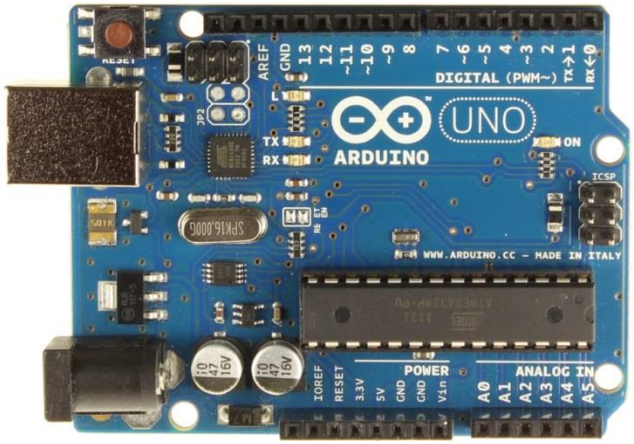
- Sistemas embarcados
- Atuação:
  - Sensores: percebem o que acontece com o mundo externo
  - Atuadores: atuam no mundo externo

Arduino

- Hardware livre
- Software livre
- Diferentes modelos
- Simples e confiável
- Robusto o suficiente para a maioria das atividades de robótica livre

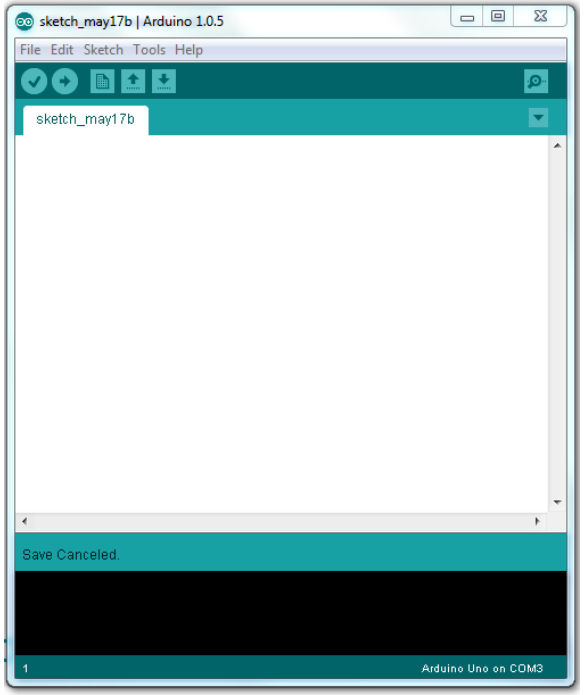
Arduino + Software + Conexão

Vista de cima



Vista de baixo

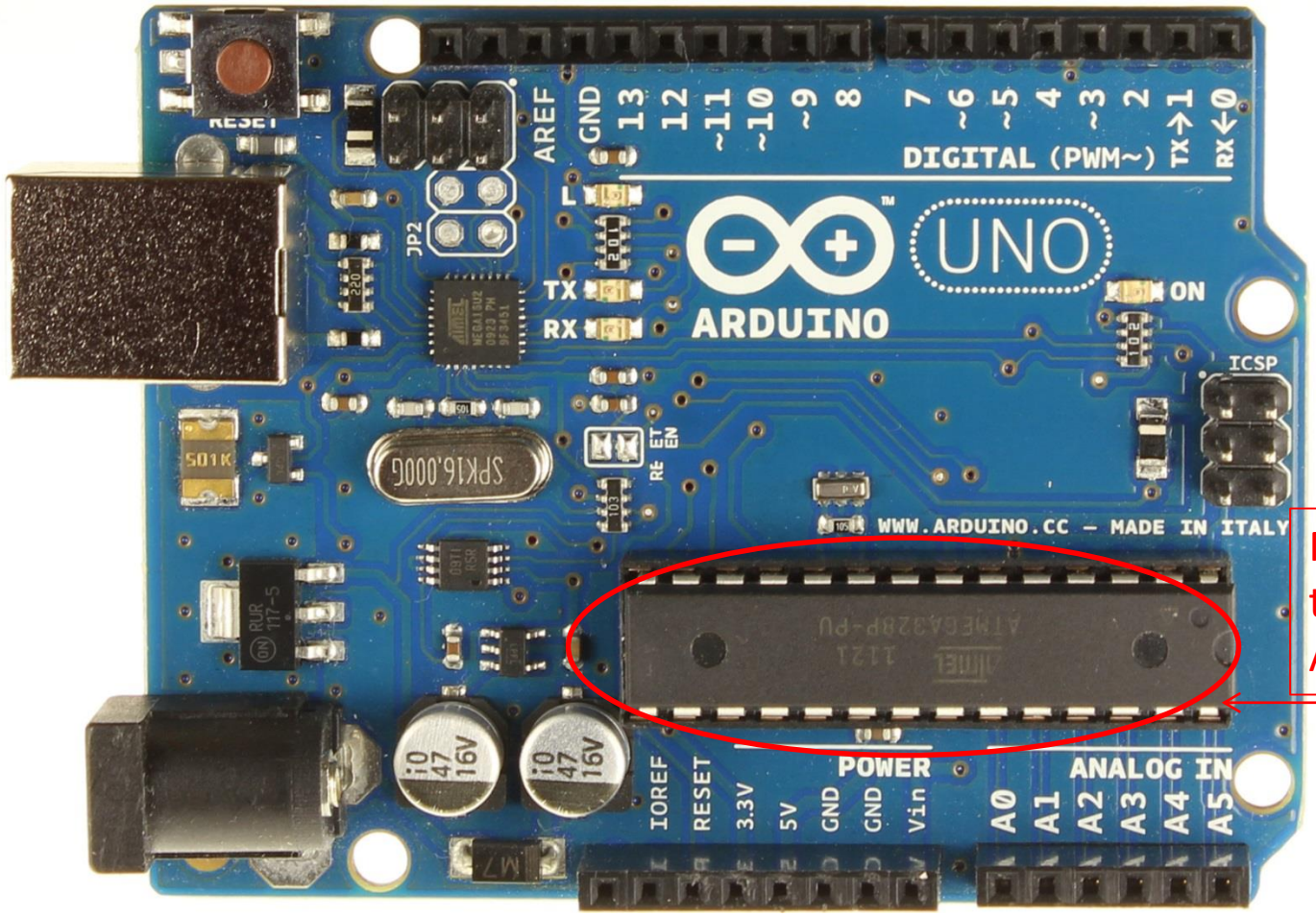
Software de programação



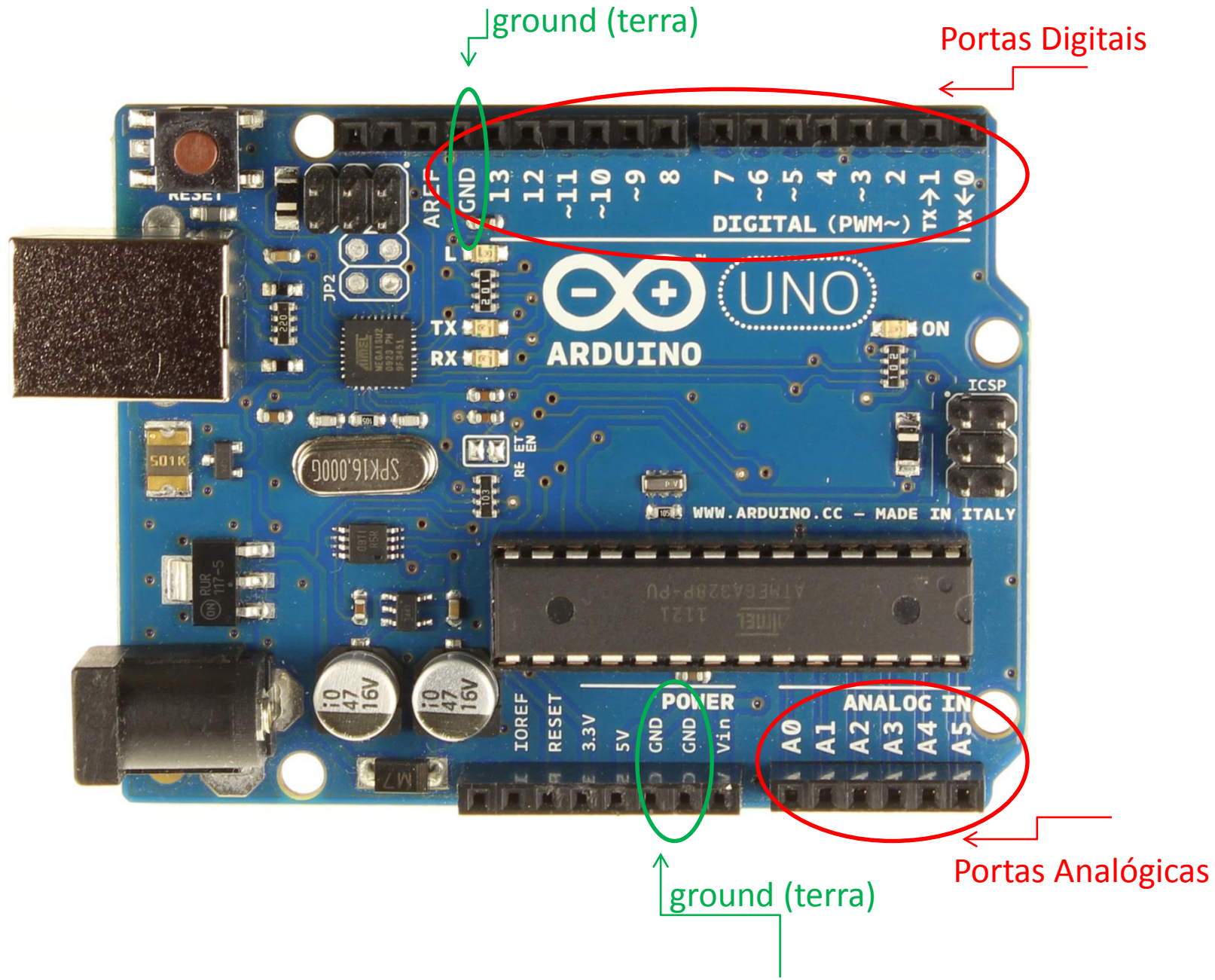
Cabo USB

Arduino Uno

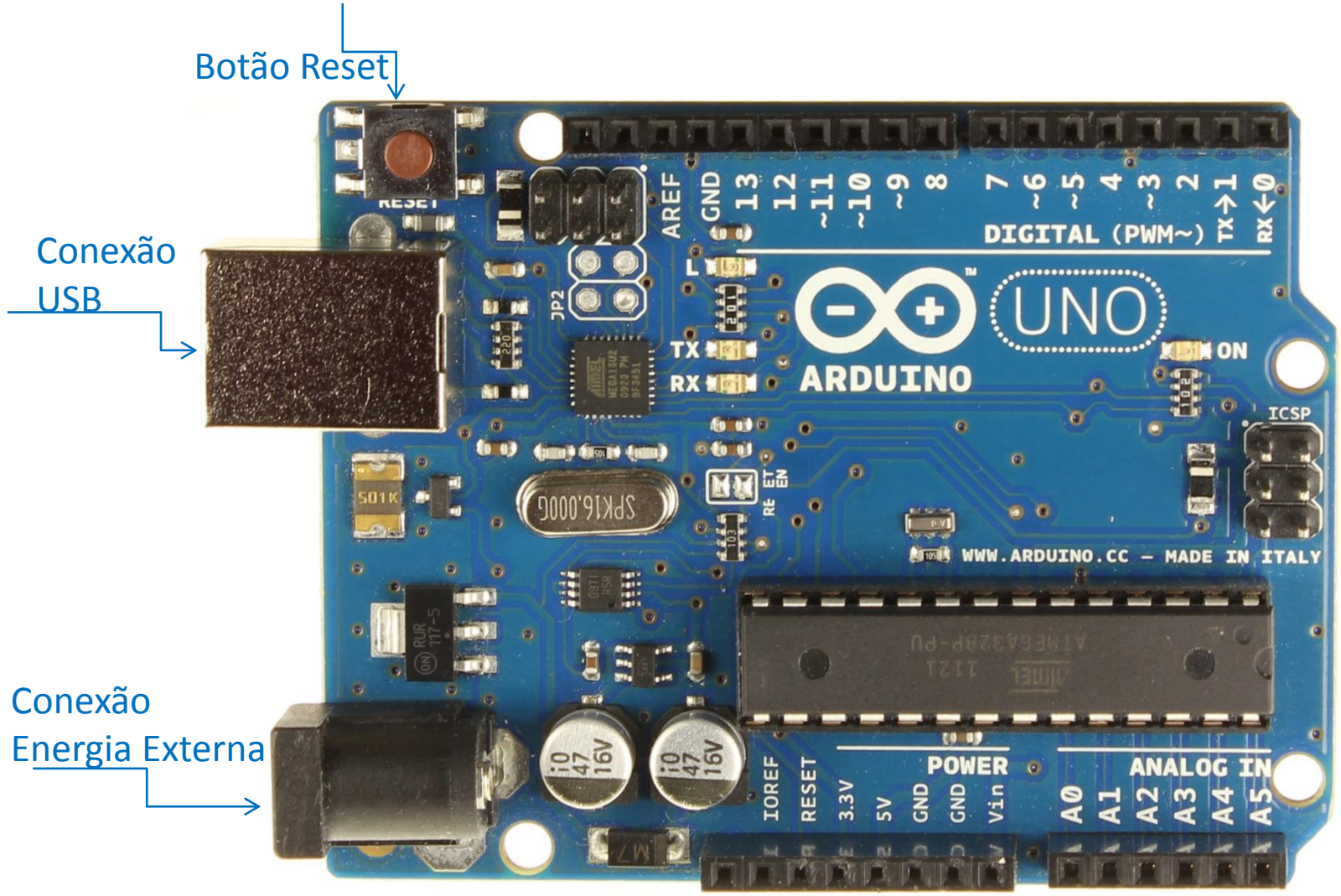
Microcontrolador	ATmega328
Tensão de funcionamento	5V
Tensão de entrada (recomendado)	7-12V
Tensão de entrada (limites)	6-20V
Pinos de entra Digital (I O)	14 (dos quais 6 oferecem saída PWM)
Pinos de entrada analógica	6
Corrente DC por pinos (IO)	40 mA
Corrente DC para 3.3V Pin	50 mA
Memória Flash	32 KB (ATmega328), dos quais 0,5 KB utilizados pelo carregador de inicialização
RAM	2 KB (ATmega328)
EPROM	1 KB (ATmega328)
Velocidade do relógio	16 MHz

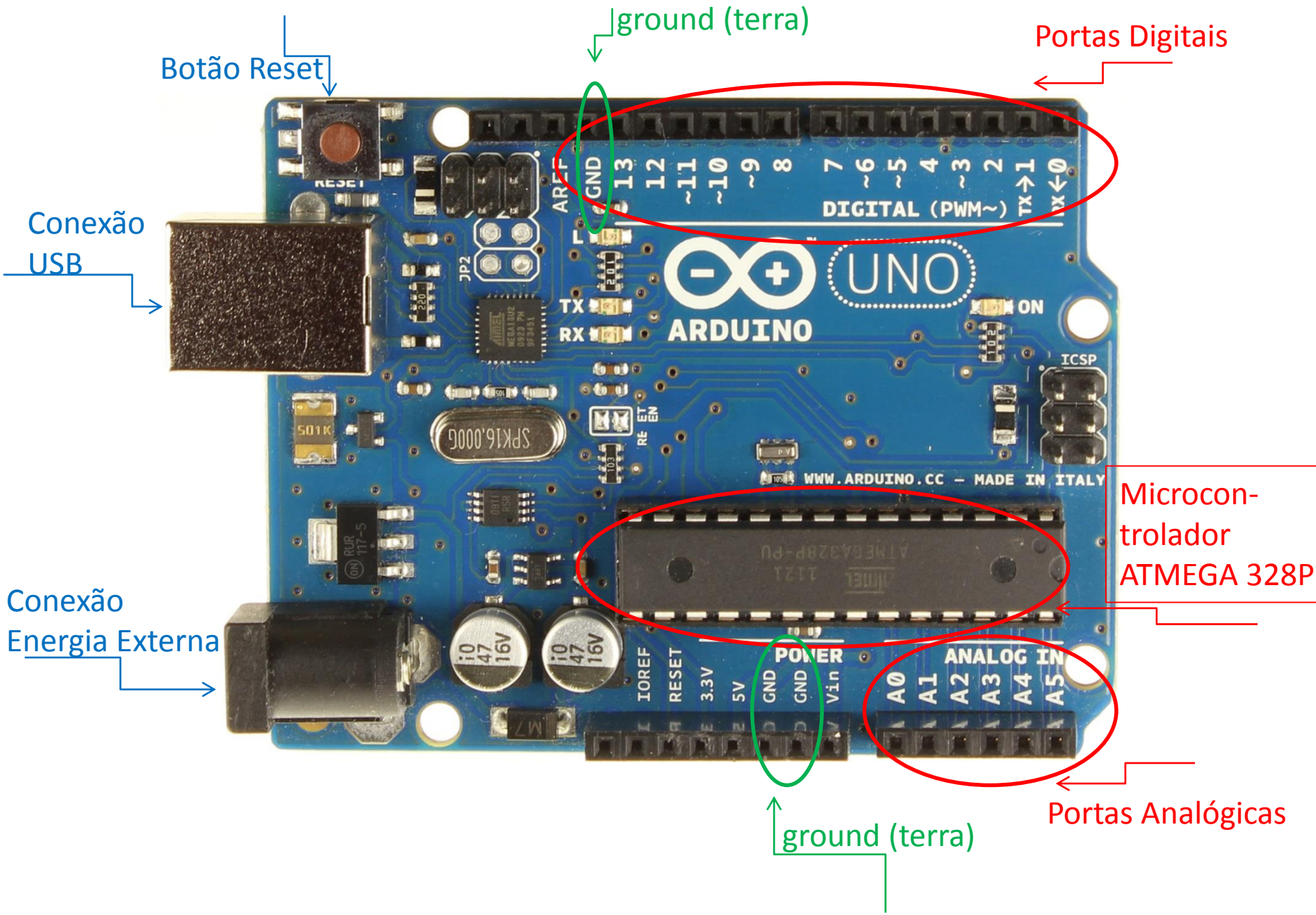


Datasheet do Atmega em http://www.atmel.com/Images/doc8161.pdf



IO (Input e Output) = Portas de entrada e saída de dados


Botão Reset
Conexão USB
Conexão Energia Externa
RESET
AREF
GND
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
DIGITAL (PWM~)
JP2
TX
RX
UNO
ARDUINO
ON
ICSP
WWW.ARDUINO.CC – MADE IN ITALY
POWER
ANALOG IN
IOREF
RESET
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5


Botão Reset
ground (terra)
Portas Digitais
Conexão USB
Conexão Energia Externa
Microcon-trolador ATMEGA 328P
Portas Analógicas
ground (terra)
DIGITAL (PWM~)
ARDUINO
UNO
POWER
ANALOG IN
WWW.ARDUINO.CC – MADE IN ITALY

Controlador

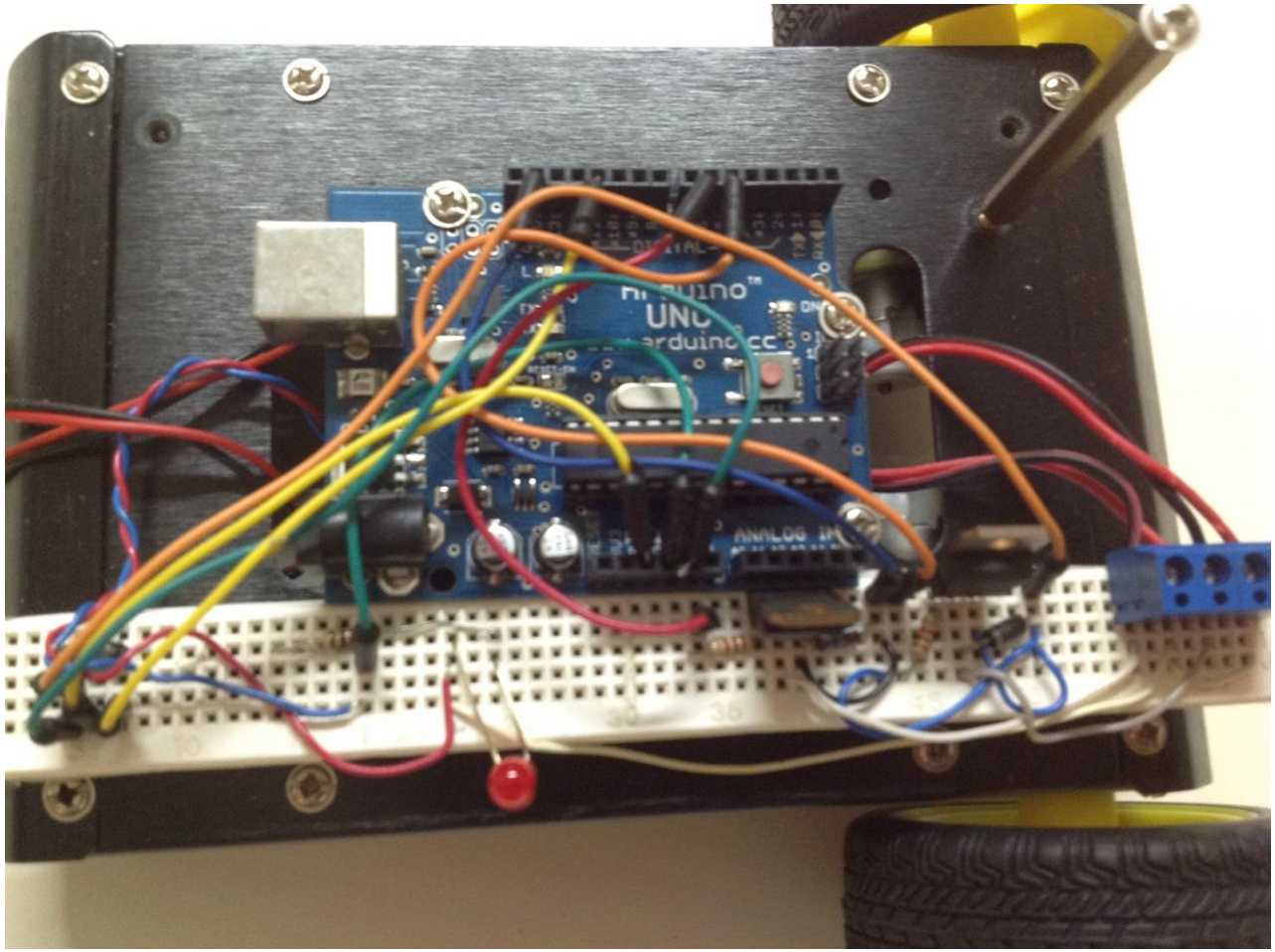
ATmega 328	Características

Parâmetros	Valor
Flash	32 Kbytes
RAM	2 Kbytes
Números de Pinos	28
Frequência Máxima de Operação	20 MHz
CPU	8-bit AVR
Número de Canais	16
Max I/O Pins	26
USB Interface	Não
USB Speed	Não

Pinagem


ATmega328 - Pinagem do Arduino
RESET
0 (RX)
1 (TX)
2
3 (PWM)
4
VCC
GND
CRISTAL
CRISTAL
5 (PWM)
6 (PWM)
7
8
(PCINT14/RESET) PC6 1
(PCINT16/RXD) PD0 2
(PCINT17/TXD) PD1 3
(PCINT18/INT0) PD2 4
(PCINT19/OC2B/INT1) PD3 5
(PCINT20/XCK/T0) PD4 6
VCC 7
GND 8
(PCINT6/XTAL1/TOSC1) PB6 9
(PCINT7/XTAL2/TOSC2) PB7 10
(PCINT21/OC0B/T1) PD5 11
(PCINT22/OC0A/AIN0) PD6 12
(PCINT23/AIN1) PD7 13
(PCINT0/CLKO/ICP1) PB0 14
28 PC5 (ADC5/SCL/PCINT13)
27 PC4 (ADC4/SDA/PCINT12)
26 PC3 (ADC3/PCINT11)
25 PC2 (ADC2/PCINT10)
24 PC1 (ADC1/PCINT9)
23 PC0 (ADC0/PCINT8)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK/PCINT5)
18 PB4 (MISO/PCINT4)
17 PB3 (MOSI/OC2A/PCINT3)
16 PB2 (SS/OC1B/PCINT2)
15 PB1 (OC1A/PCINT1)
Analog5
Analog4
Analog3
Analog2
Analog1
Analog0
GND
AREF
VCC
13
12
11 (PWM)
10 (PWM)
9 (PWM)

O Arduino não funciona só!

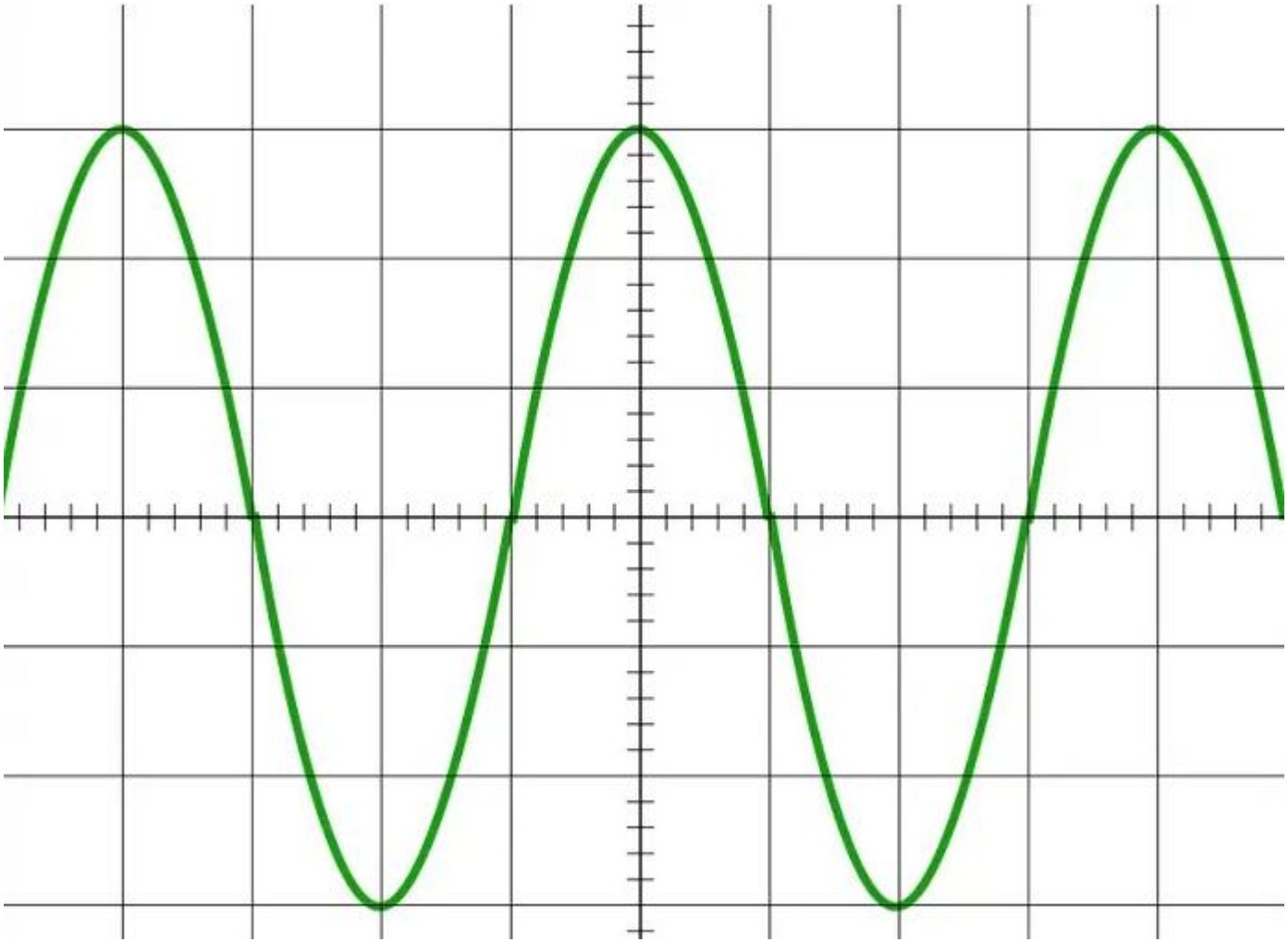


Sinal Analógico

- O antigo formato analógico é composto por um sinal contínuo, que varia em função do tempo. É possível representá-lo com uma curva, que apresenta intervalos com valores que variam entre 0 e 10. Uma das principais características deste tipo de sinal é que ele passa por todos os valores intermediários possíveis (0.01 , 0.566 , 4.565 , 8.55...), o que resulta em uma faixa de frequência bem maior e por isso não tão confiável e com qualidade inferior, devido à oscilação.

- É muito comum representar este tipo de onda como uma série de parábolas que se alternam sobre um centro. Resumindo, ele é um sinal que é caracterizado pela variação contínua de suas grandezas em um determinado espaço de tempo.

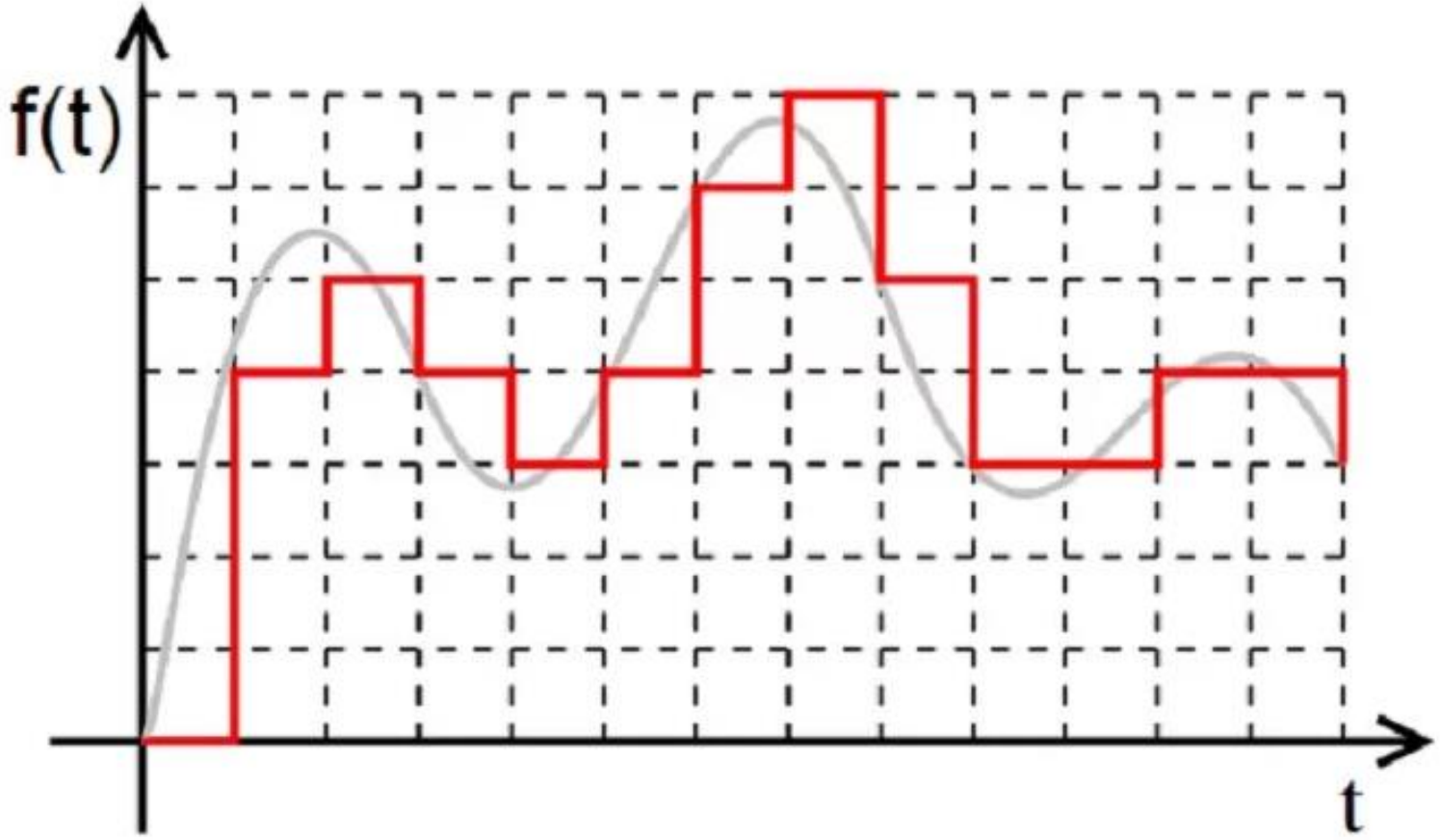
Sinal Analógico



Sinal Digital

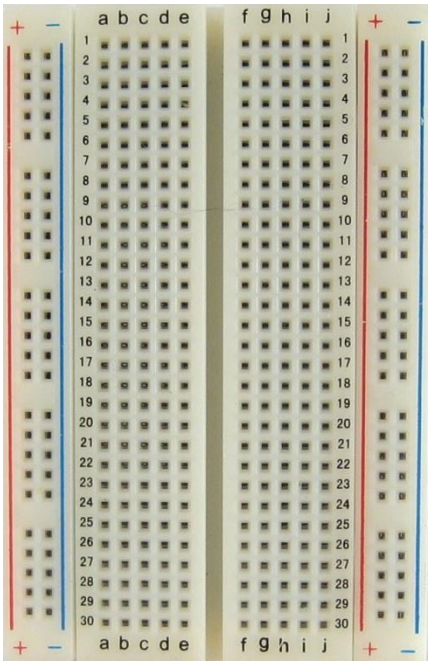
- O sinal digital tem valores discretos, com números descontínuos no tempo e na amplitude. Enquanto o formato analógico apresenta variações infinitas entre cada um de seus valores, o digital assumirá sempre os valores discretos (0,1,2,3,4,5,6,7,8,9,10), diminuindo a faixa de frequência entre eles e a oscilação.

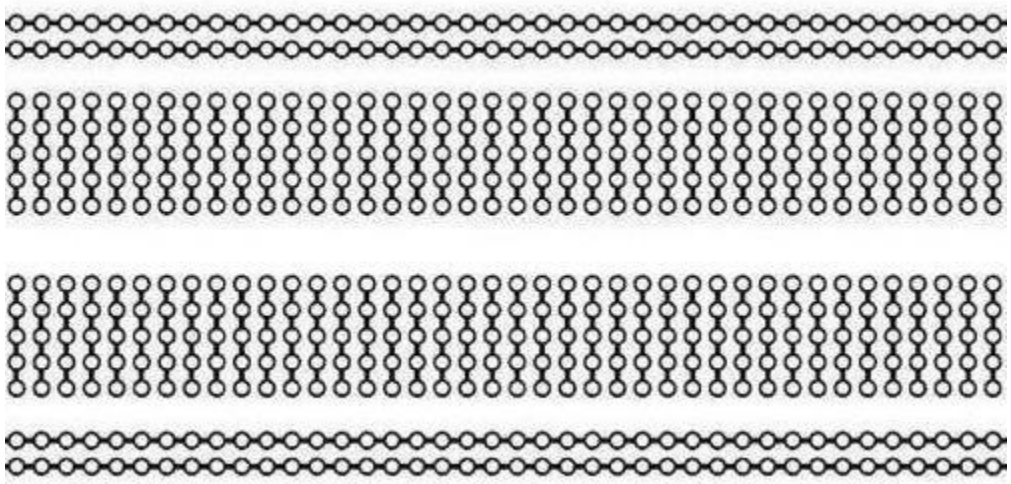
Sinal Digital



Protoboard

- Testes devem ser realizados numa placa de prototipagem onde os componentes são fixados sem a necessidade de solda.





Internamente, a placa tem comunicação metálica conforme esquema acima.

Push-Button

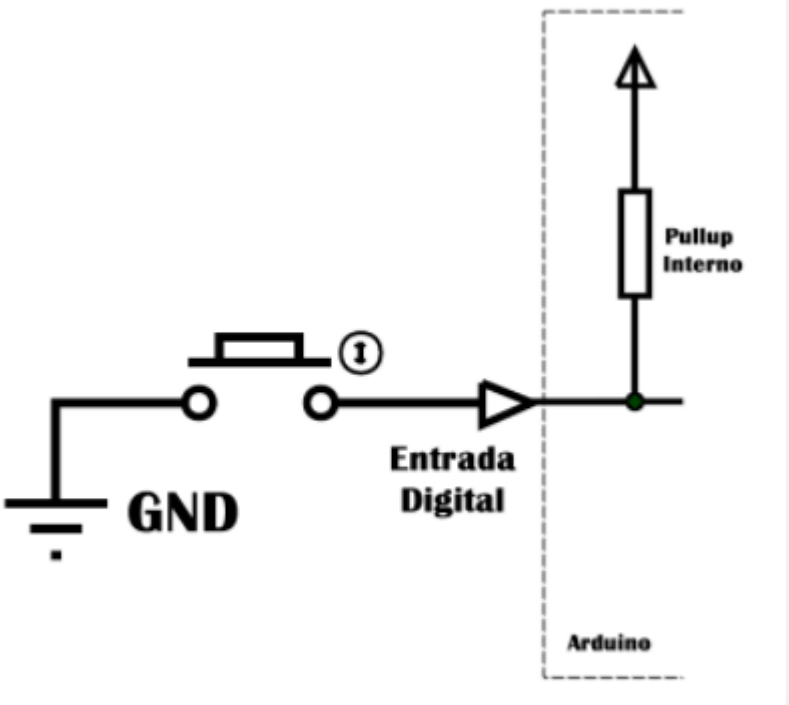


Figura 1 – Exemplo de esquema de ligação de um botão.

Fonte:
http://blog.vidadesilicio.com.br/arduino/basico/botoes/

Push-Button - Bounce

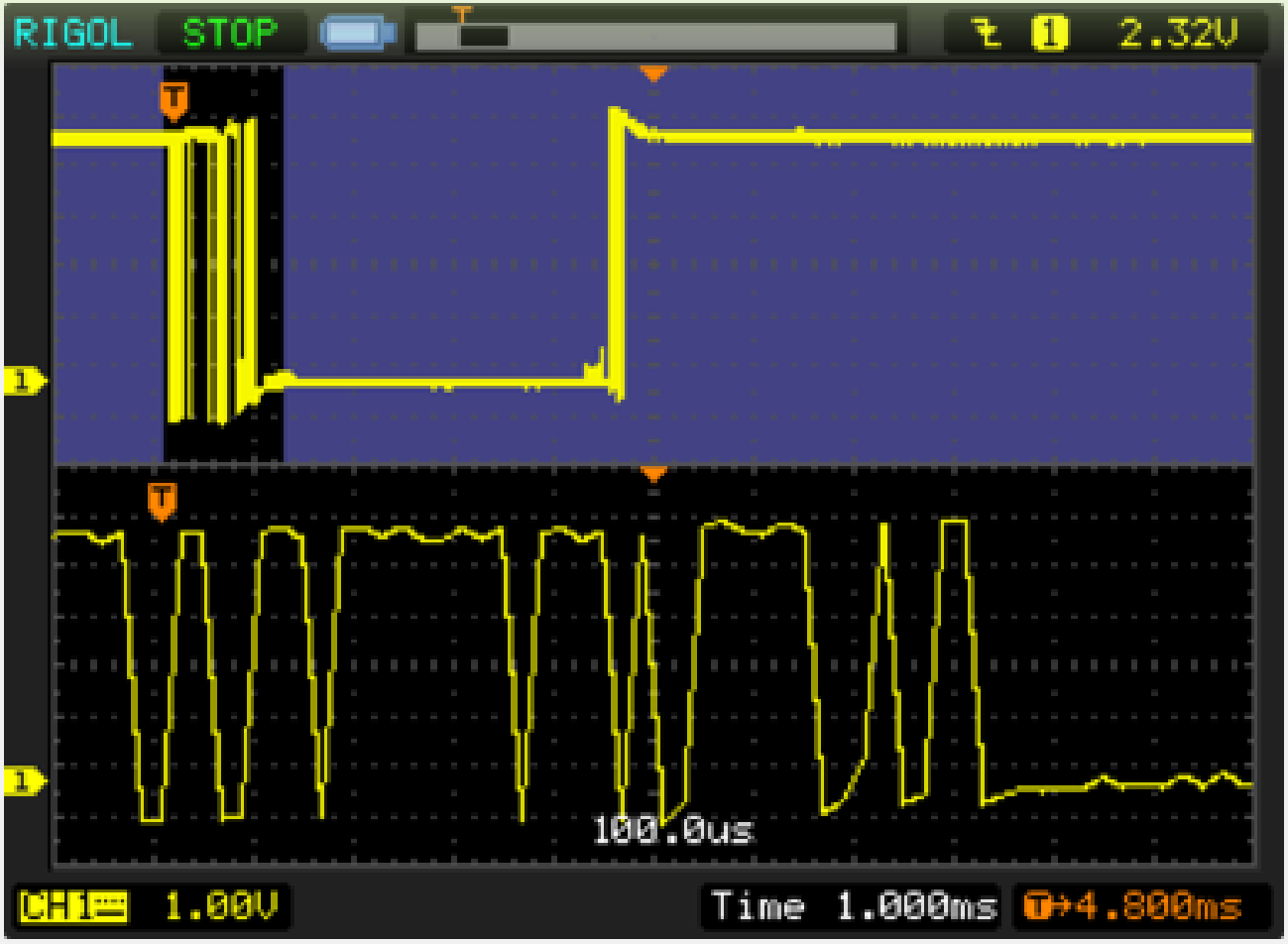


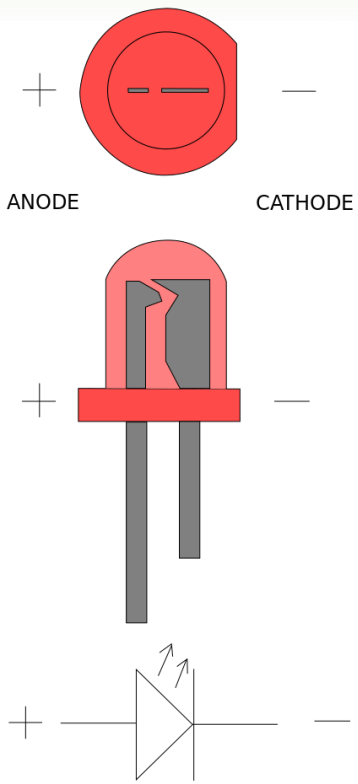
Figura 3 – Forma de onda de um botão ao ser pressionado.

Eletrônica básica

Eletricidade é um fluxo (deslocamento) de elétrons por um material bom condutor.

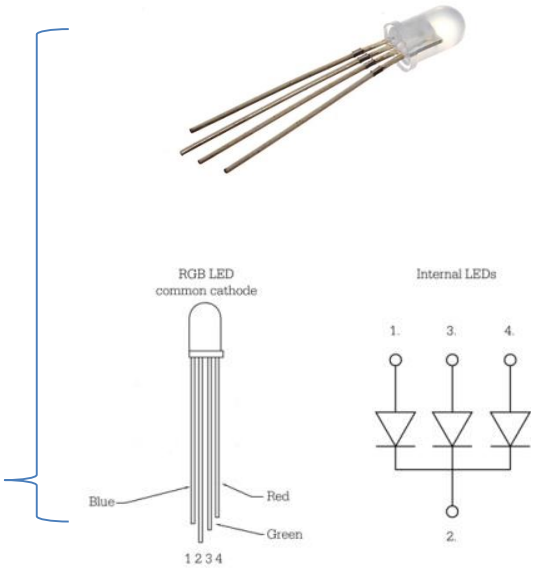
- Tensão ou voltagem (***V***) = a diferença de potencial (ddp) entre dois pontos de um condutor. É medido em **V**(volts).
- Corrente (***i***) = a quantidade de elétrons que passa por segundo, num segmento do condutor. É medida em **A** (amperes).
- Resistência (***R***) = a dificuldade oferecida pelo material à passagem da corrente elétrica. É medida em **Ω** (ohms).

LEDs



- O LED (*Light Emitting Diode),* por ser um diodo e não uma lâmpada, deixa passar a corrente num único sentido.

- O LED RGB emite, numa mesma peça, as cores vermelho, verde e azul e tem o formato à direita:

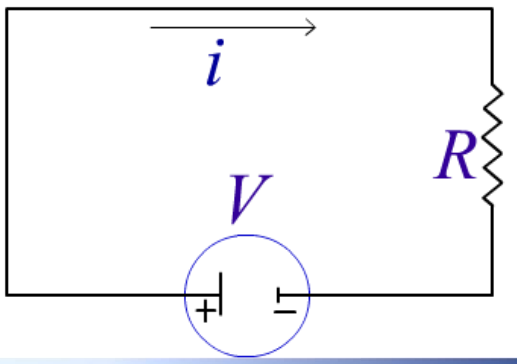


Lei de Ohm

O Arduino:

- trabalha com tensão entre 1,8V a 5,5 V;
- as intensidade de corrente máxima serão de aproximadamente 50 mA para 3 V a 300 mA para 5V;
- e para obtermos o valor da resistência R, usamos a **Lei de Ohm**:

$$V = i\,R \qquad \text{ou} \qquad R = \frac{V}{i}$$



Exemplo de aplicação da Lei de Ohm

- Para conectar um LED no Arduino, necessitamos associá-lo a um resistor para que a tensão de trabalho (5V) do Arduino produza a corrente mais aproximada possível daquela suportada pelo LED para que este não queime.

- Cada LED tem uma resistência interna. A condição ideal de trabalho são para os seguintes valores aproximados de tensão e corrente (i):

LED	Tensão	Corrente
Vermelho e infravermelho	1,8 V	15mA = 0,015A
Verde, amarelo e laranja	2,1 V	20 a 25 mA
Azul e branco	2,7 V a 4,2 V	15 a 30 mA

Assim, se usarmos um LED verde podemos calcular o valor do resistor que deve ser associado, através da Lei de Ohm ($R = \frac{V}{i}$)

$$R = \frac{V_{FONTE} - V_{LED}}{i}$$

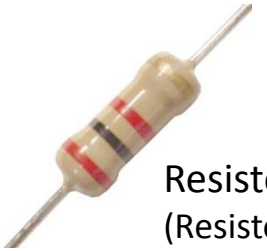
$$R = \frac{5V - 2{,}1V}{0{,}02\ A}$$

$$R = 145\ \Omega\ (145\ ohms)$$

NOTA: Os resistores são apresentados em valores fixos. Então opta-se por aquele mais próximo. Neste caso, usa-se um de 100 Ω.

Resistores

- Resistores limitam a passagem de corrente elétrica, impedindo que alguns componentes venham a ser danificados por excesso de tensão elétrica.

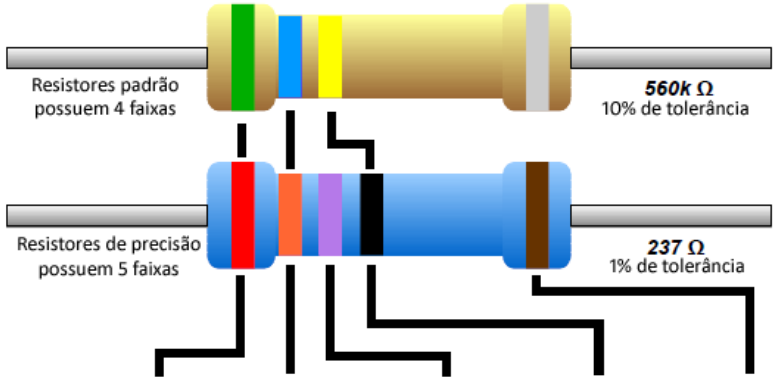


Resistor de carvão
(Resistor de valor fixo)

Potenciômetro
(Resistor variável)

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda

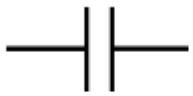


Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

Imagem extraída de http://www.eletronicadidatica.com.br/

Capacitores

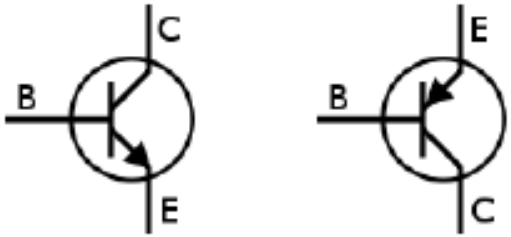
Os capacitores são componentes que armazenam energia em forma de campo elétrico. São formados por duas placas metálicas com um dielétrico (isolante) no meio. A unidade de medida é o Farad (F), porém como 1 Farad é algo bem grande, comumente encontramos capacitores na casa dos mF (miliFarad), μF (microFarad) e pF (picoFarad).



Representação de um capacitor em circuito elétrico

Capacitor eletrolítico bastante comum no mercado

Transistores



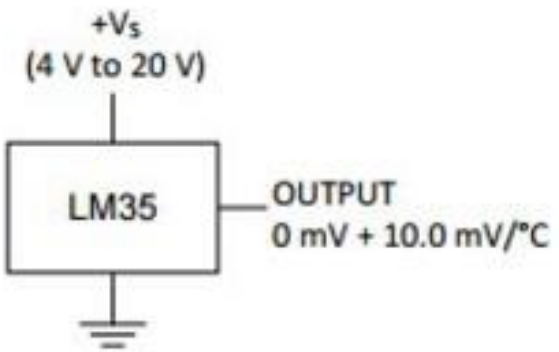
(a) NPN (b) PNP

$$I_C + I_B = I_E$$

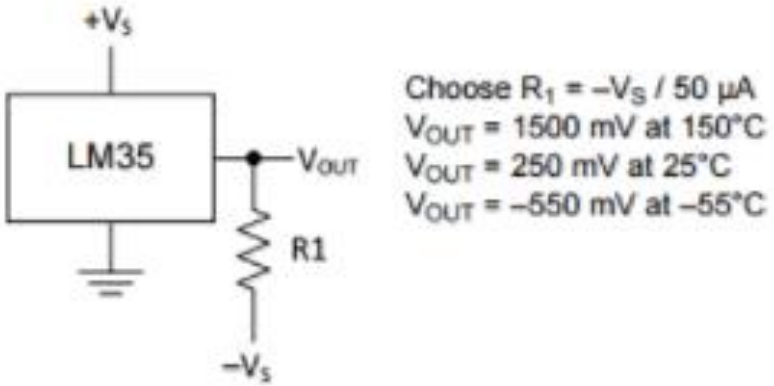
$$I_C = \beta I_B,$$

Figura 2.12: Componente 2N3904: transistor NPN com encapsulamento TO-92

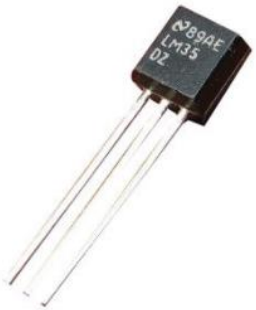
Sensor de Temperatura



Modo básico – (2°C a 150°C)



Modo escala completa – (-55°C a 150°C)



Fonte:
http://blog.vidadesilicio.com.br/arduino/basico/lm35-medindo-temperatura-com-arduino/

Sensor de Luz

Fig 2 – Um típico LDR

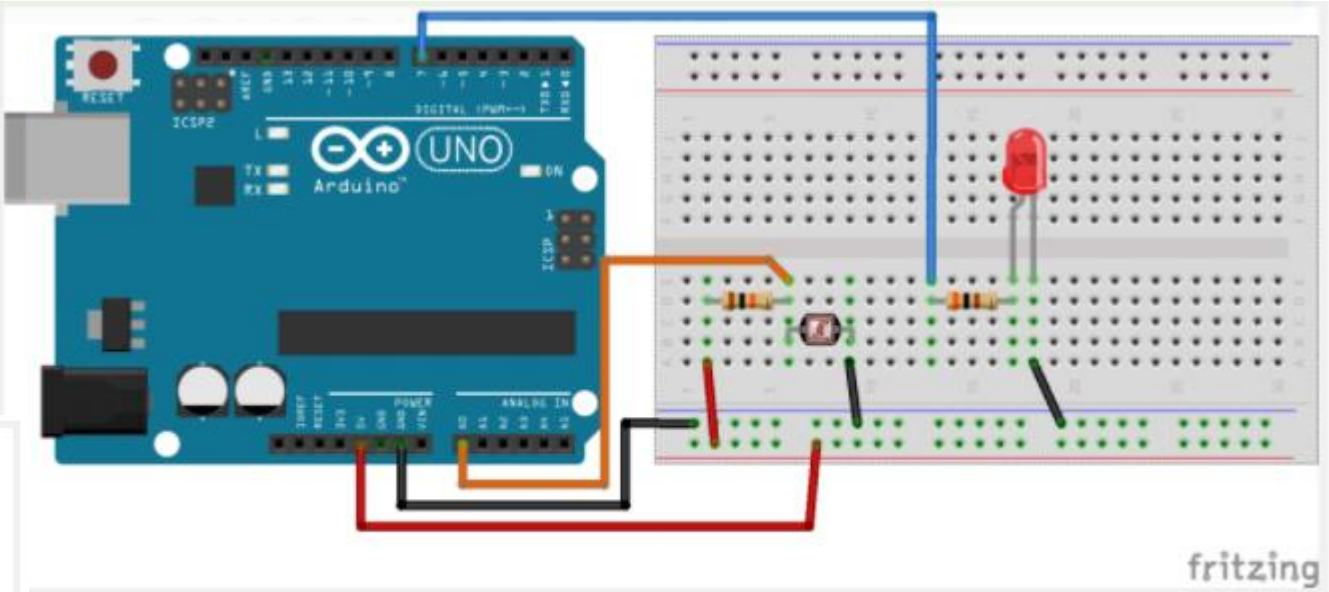
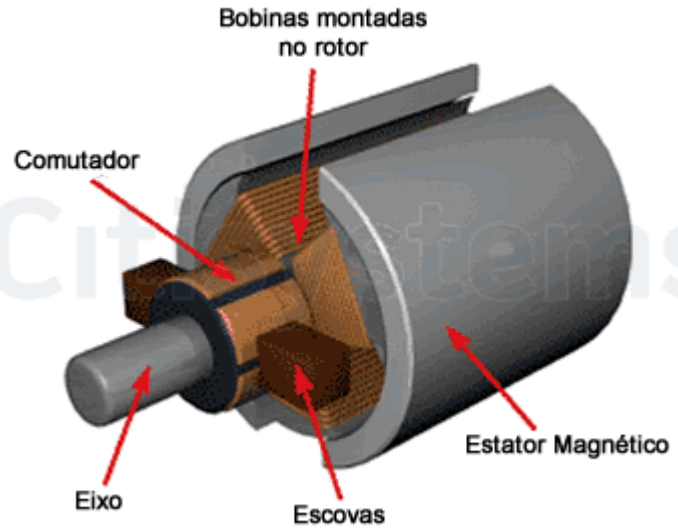
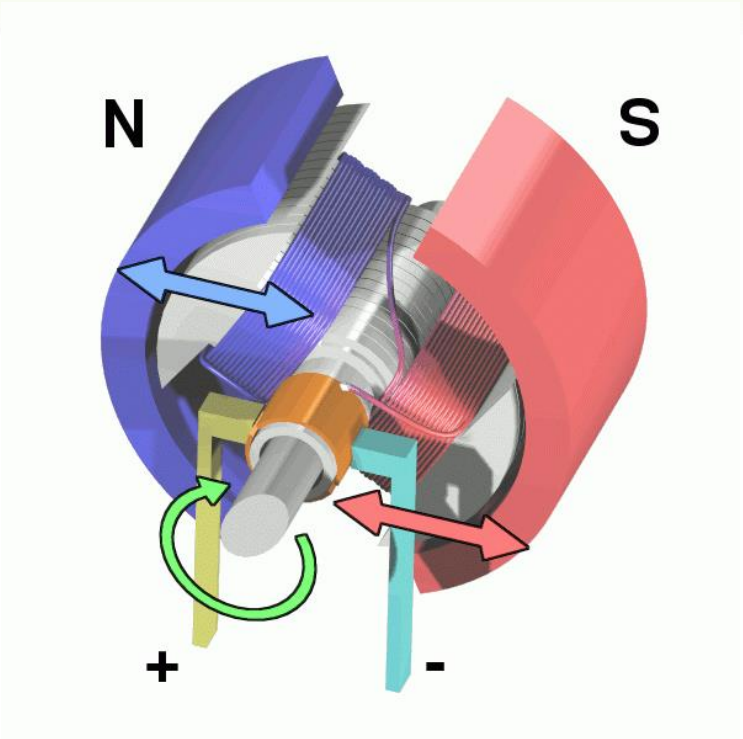


Fig 3 – Esquema de montagem

Fonte:
http://blog.vidadesilicio.com.br/arduino/basico/sensor-de-luz-ldr/

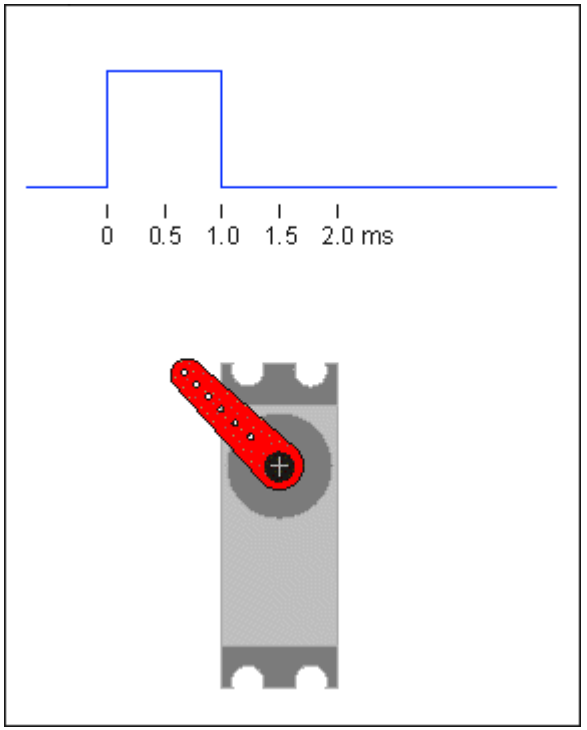
Motor CC





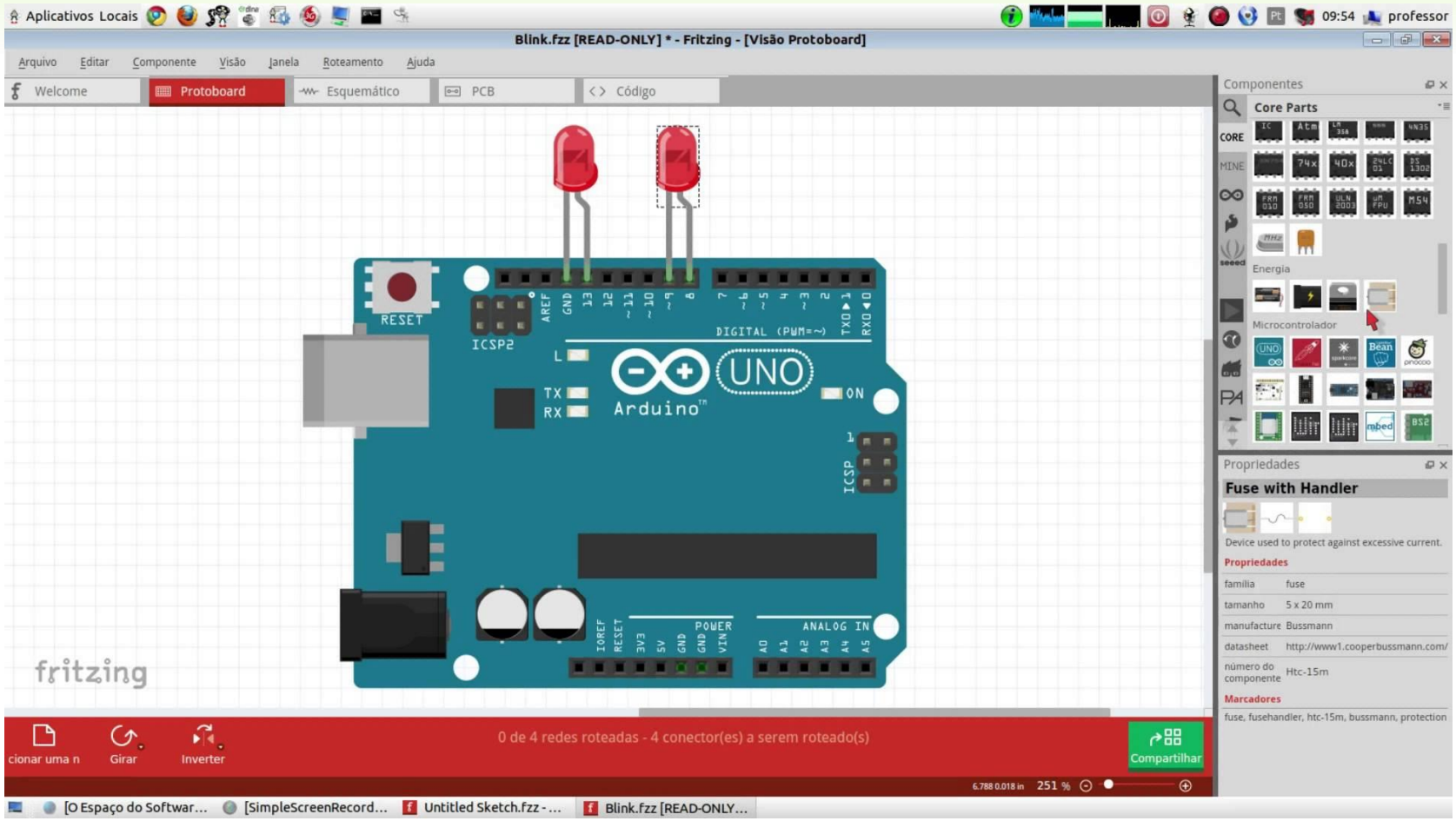
Fonte: https://www.citisystems.com.br/motor-cc/

Servomotor



Fonte: http://www.arduinoecia.com.br/2013/06/controlando-um-servo.html

Diagramas


Aplicativos Locais
09:54
professor
Blink.fzz [READ-ONLY] * - Fritzing - [Visão Protoboard]
Arquivo
Editar
Componente
Visão
Janela
Roteamento
Ajuda
Welcome
Protoboard
Esquemático
PCB
Código
Componentes
Core Parts
Energia
Microcontrolador
Propriedades
Fuse with Handler
Device used to protect against excessive current.
Propriedades
família fuse
tamanho 5 x 20 mm
manufacture Bussmann
datasheet http://www1.cooperbussmann.com/
número do componente Htc-15m
Marcadores
fuse, fusehandler, htc-15m, bussmann, protection
RESET
ICSP2
DIGITAL (PWM=~)
UNO
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN
fritzing
cionar uma n
Girar
Inverter
0 de 4 redes roteadas - 4 conector(es) a serem roteado(s)
Compartilhar
251 %
[O Espaço do Softwar...
[SimpleScreenRecord...
Untitled Sketch.fzz - ...
Blink.fzz [READ-ONLY...

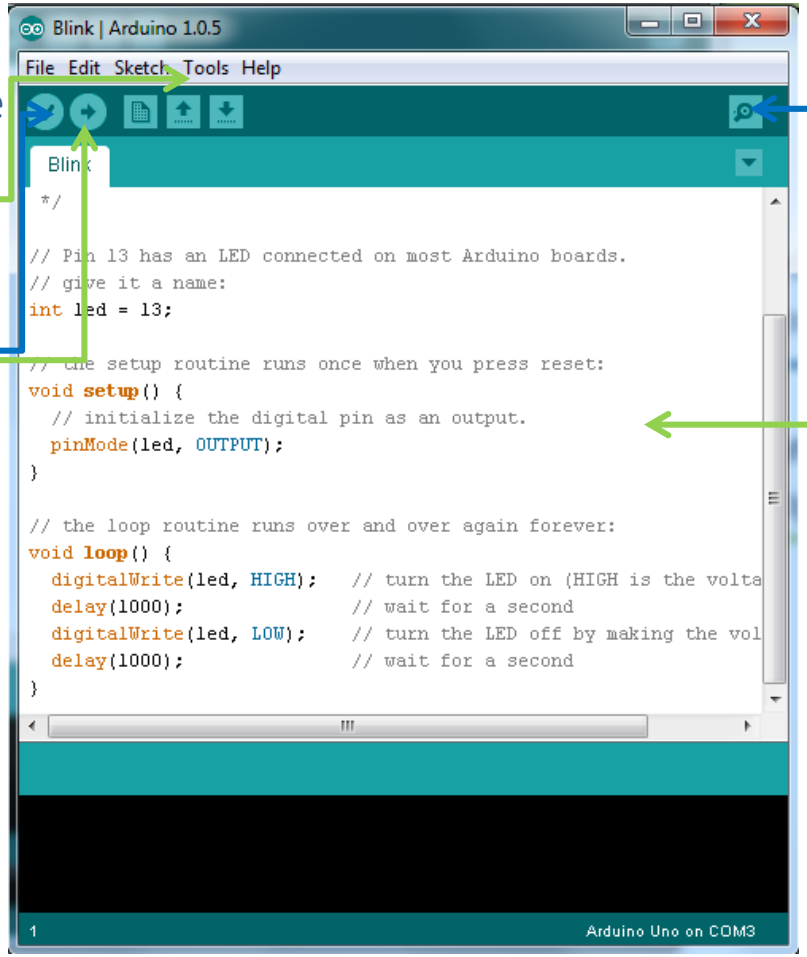
Conceitos Básicos

Software de Programação

Tools - seleciona o tipo de Arduino e a porta COM

Verify – compila a programação

Upload – envia a programação para o Arduino



Serial Monitor – monitor de dados

Sketch – palco da programação

Ambiente de Desenvolvimento Integrado do Arduino [Integrated Development Environment (IDE)

Setup e Loop

```
01  // Variáveis de 32 bits sem sinal
02  uint32_t referencia = 0;
03  uint32_t tempoAtual;
04  void setup() {
05  pinMode(13,OUTPUT); // Configura o pino 13 (LED) como saída
06  }
07
08  void loop()
09  {
10       tempoAtual = millis(); // Captura o tempo atual
11  // Verifica se já se passaram 1 segundo (1000 milissegundos)
12  // desde a última vez em que o LED foi invertido
13       if(tempoAtual - referencia >= 1000) {
14          // Se sim, inverte o LED e atualiza a referencia
15          digitalWrite(13,!digitalRead(13));
16          referencia = millis();
17       }
18  }
```

Pinos

INPUT – *Configura o pino como entrada;*

INPUT_PULLUP – *Configura o pino como entrada e ativa o resistor de pull-up interno;*

OUTPUT – *Configura o pino como saída.*

```
1  pinMode(13,OUTPUT); // Configura o pino 13 (LED) como saída digital.
2  pinMode(A0,INPUT); // Configura o pino A0 como entrada digital.
```

Fonte:
httphttp://blog.vidadesilicio.com.br/arduino/basico/entradas-e-saidas-digitais-2/

Pinos

> *HIGH – Estado lógico **alto** (**+5 V**);*
>
> *LOW – Estado lógico baixo (**GND**);*

```
1  pinMode(13,OUTPUT); //Configura o pino 13 (LED) como saída digital
2  digitalWrite(13,HIGH); // Altera o pino 13 para o estado lógico alto (+5V)
```

Fonte:
httphttp://blog.vidadesilicio.com.br/arduino/basico/entradas-e-saidas-digitais-2/

Delay

Criando pausas entre comandos

```
1  digitalWrite(13,HIGH); //Configura o pino 13 no estado lógico alto (+5V)
2  delay(1000); // Aguarda 1 segundo (1000 milisegundos)
3  digitalWrite(13,LOW); // Configura o pino 13 no estado lógico baixo (0V)
```

Fonte:
httphttp://blog.vidadesilicio.com.br/arduino/basico/entradas-e-saidas-digitais-2/

Medindo tempo de Execução

```
tempo = millis(); // Retorna o tempo em milissegundos e armazena na variável tempo
```

```
tempo = micros(); // Retorna o tempo em milissegundos e armazena na variável tempo
```

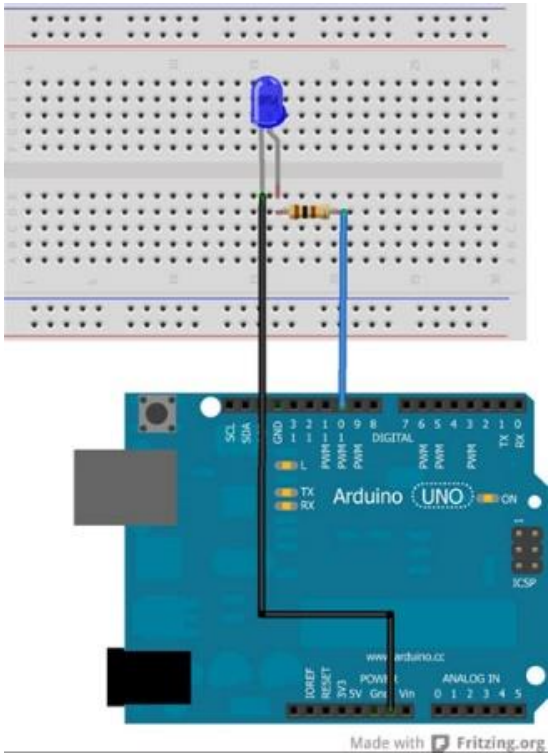
```
tempoAtual = millis(); // Captura o tempo atual</pre>
// Verifica se já se passaram 1 segundo (1000 milissegundos)
// desde a última vez em que o LED foi invertido
if(tempoAtual - referencia > 1000) {
  // Se sim, inverte o LED e atualiza a referencia
  digitalWrite(13,!digitalRead(13));
  referencia = millis();
}
```

Fonte:
httphttp://blog.vidadesilicio.com.br/arduino/basico/entradas-e-saidas-digitais-2/

Exemplos de Projetos

LED com efeito fader

- 1 LED
- 1 Resistor 100 ohms
- Jumpers.

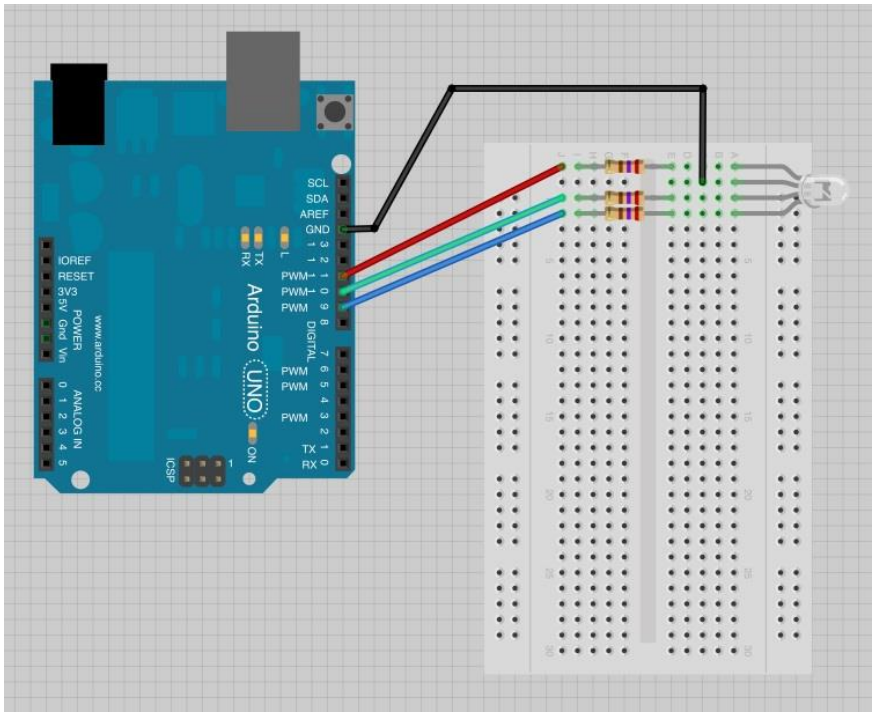



```
const int LED = 10;
int brilho = 0;
int fader = 5;
boolean acende = true;
boolean apaga = false;

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

void loop() {
   analogWrite(LED,brilho);
if (acende == true) {
   if (brilho < 255) {
     brilho = brilho + fader;
} else {
     acende = false;
     apaga = true;
   }
  }
 if (apaga == true) {
  if (brilho > 0) {
   brilho = brilho - fader;
} else {
   acende = true;
   apaga = false;
  }
 }
 delay (30);
}
```

Exemplo retirado de: http://www.comofazerascoisas.com.br/projeto-arduino-como-piscar-led-com-efeito-fader.html

Controle de um LED RGB

1 LED RGB
3 resistores 330 ohm
Jumpers.

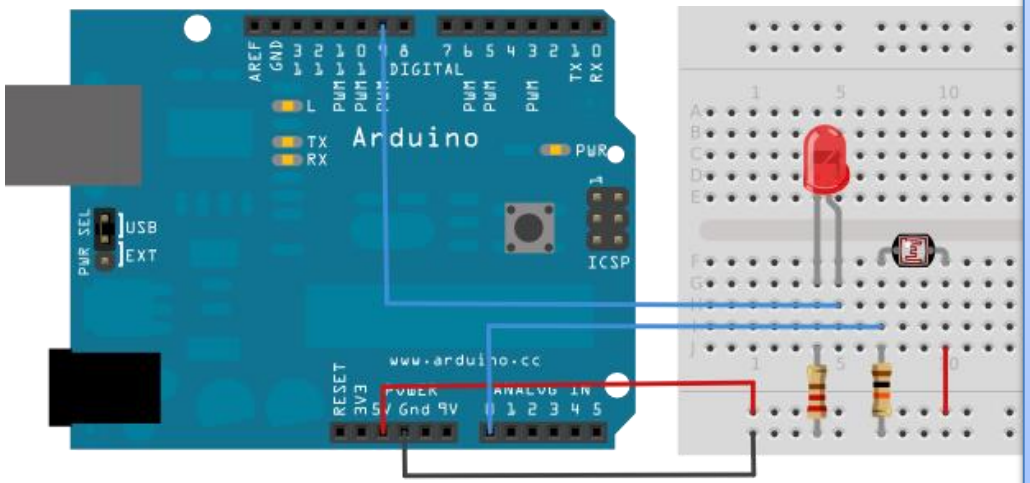



```
const int ledAzul = 8;
const int ledVerde = 9;
const int ledVermelho = 10;

void setup() {
  pinMode(ledAzul,OUTPUT);
  pinMode(ledVerde,OUTPUT);
  pinMode(ledVermelho,OUTPUT);
}
void loop() {
  digitalWrite(ledAzul,HIGH);
  delay(500);
  digitalWrite(ledAzul,LOW);
      digitalWrite(ledVerde,HIGH);
      delay(500);
      digitalWrite(ledVerde,LOW);
  digitalWrite(ledVermelho,HIGH);
  delay(500);
  digitalWrite(ledVermelho,LOW);
  }
```

Exemplo adaptado de https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/parts

Controle de um LED com Sensor de Luz (LDR)

1 LED
1 resistor 100 ohms
1 resistor 1k ohms
1 LDR
Jumpers.

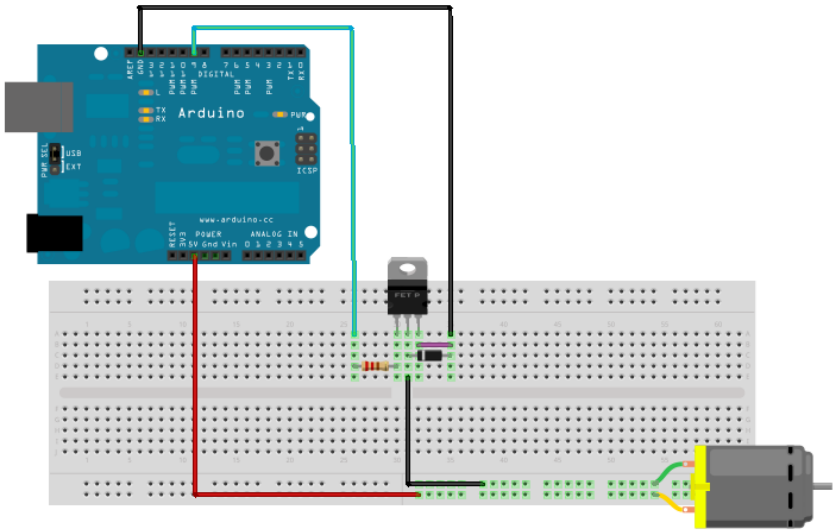



```
const int sensorPin = A0;
const int ledPin = 13;
int sensorValue = 0;
int sensorMin = 1023;
int sensorMax = 0;

void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  Serial.begin(9600);

while (millis() < 5000) {
    sensorValue = analogRead(sensorPin);
    if (sensorValor > sensorMax) {
      sensorMax = sensorValue;
    }
    if (sensorValue < sensorMin) {
      sensorMin = sensorValue;
    }
  }
digitalWrite(13, LOW);
}

void loop() {
sensorValue = analogRead(sensorPin);
sensorValue = map(sensorValue, sensorMin,
sensorMax, 0, 255);
sensorValue = constrain(sensorValue, 0, 255);
analogWrite(ledPin, sensorValue);
  {
int sensorValue = analogRead(A0);
    Serial.println(sensorValue);
  delay(1);}
}
```

Exemplo retirado de https://www.arduino.cc/en/Tutorial/Calibration

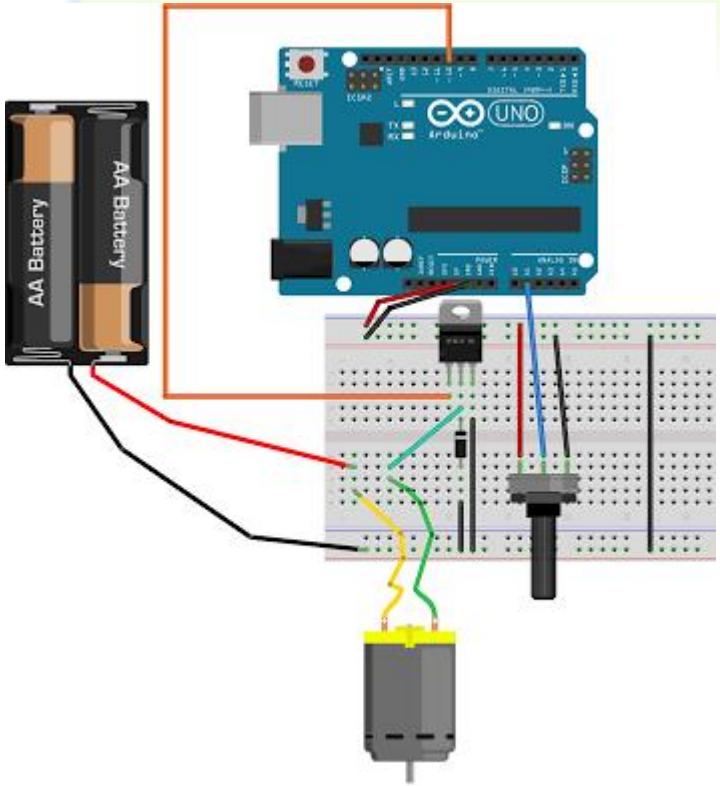
Movimentando um motor de Corrente Contínua com TIP

1 Motor DC 5v.
1 Transistor TIP 122 ou TIP 120
1 Resistor 2.2k ohms
1 Diodo IN4004 ou IN4007
Jumpers.



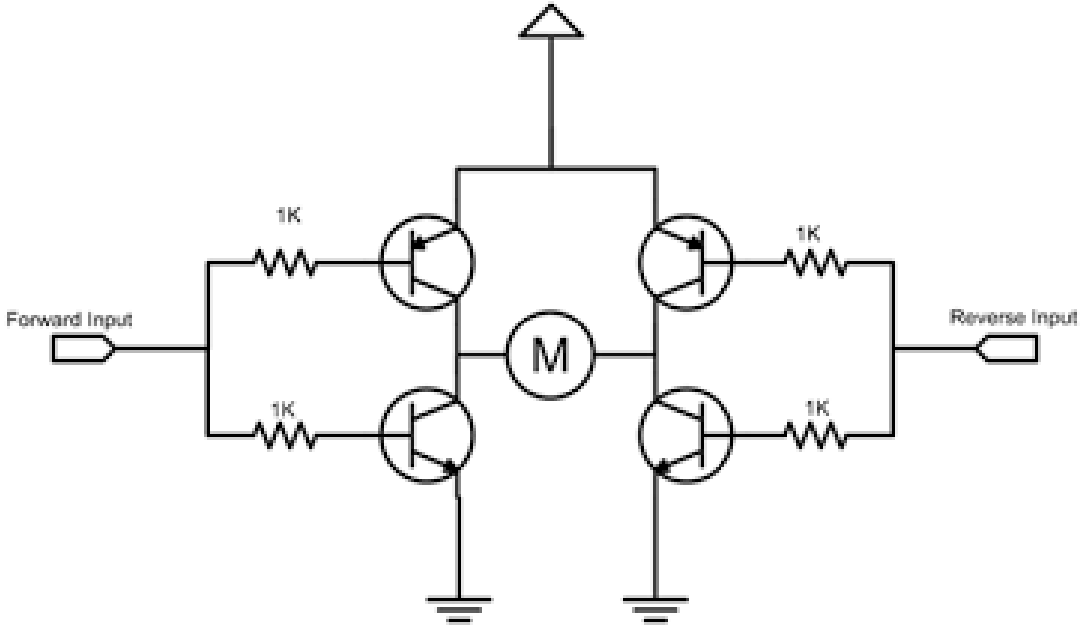

```
int MotorPin =  9;

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

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

Controle do Motor CC com Potenciômetro

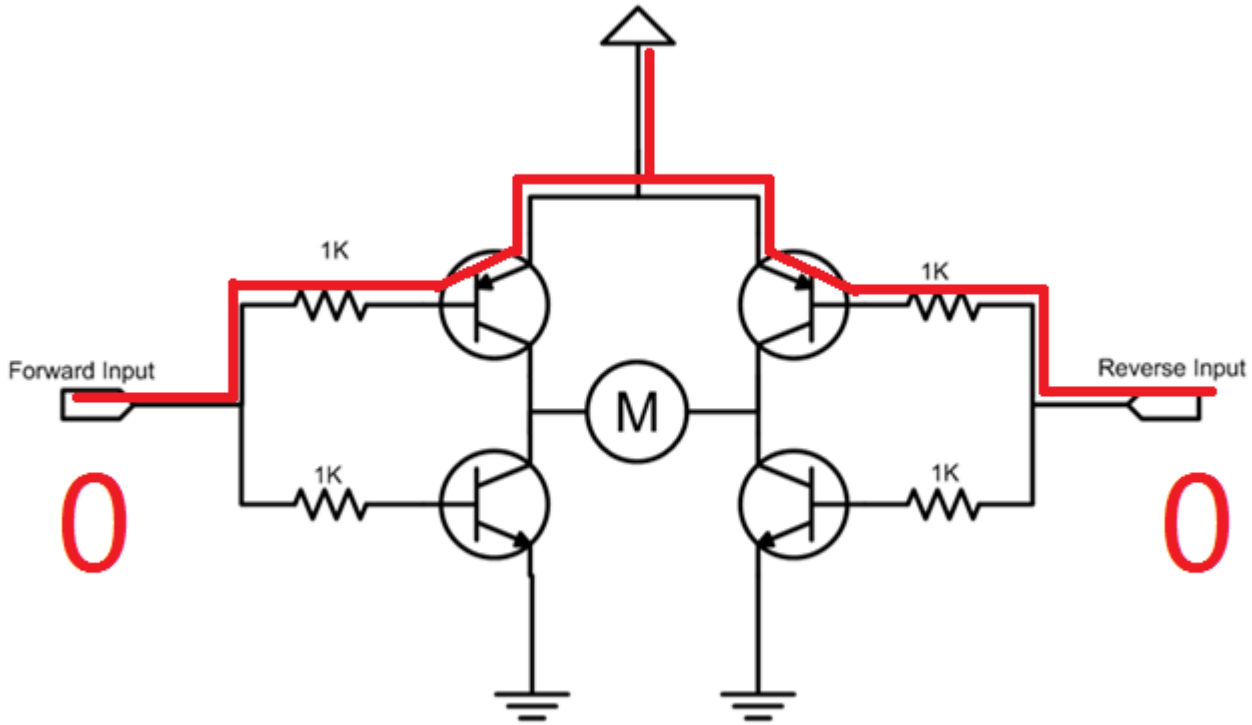



```
int const potenciometro = 1;
int const transTIP120 = 10;
int valPotenciometro = 0;

  void setup() {
  pinMode(transTIP120, OUTPUT);
}
void loop() {
    valPotenciometro = analogRead(potenciometro) / 4;
    analogWrite(transTIP120, valPotenciometro);

}
```

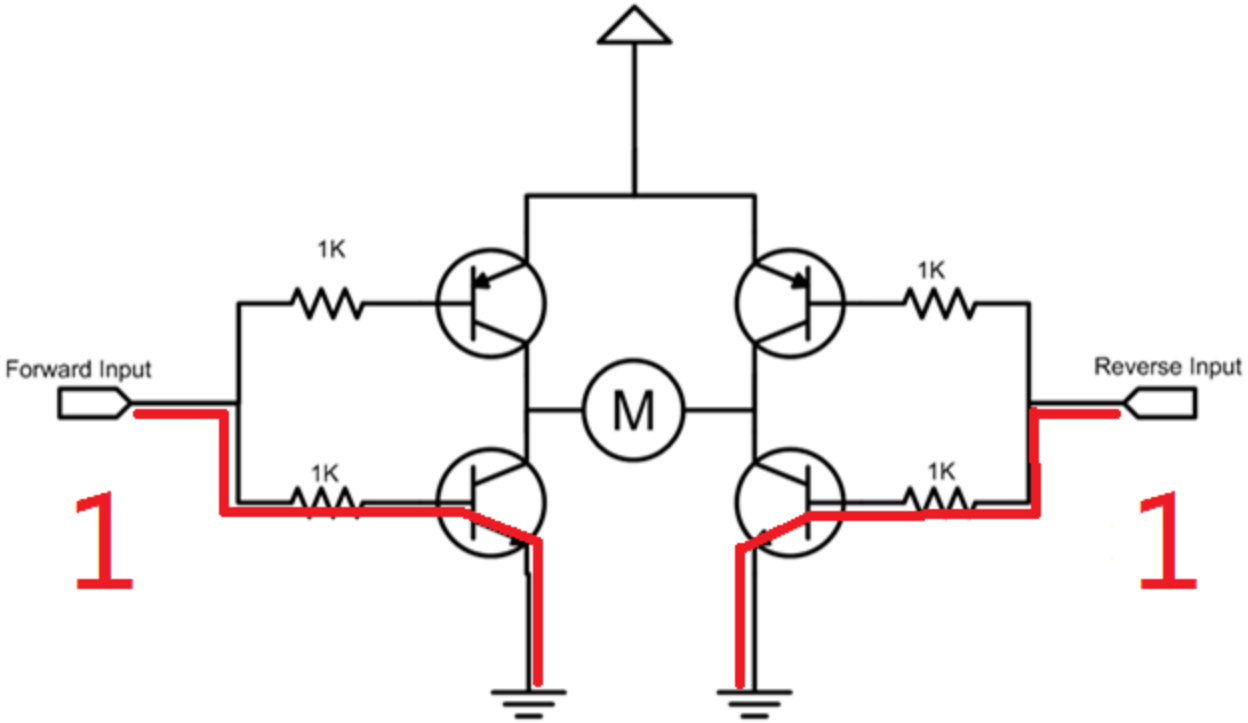
Ponte H



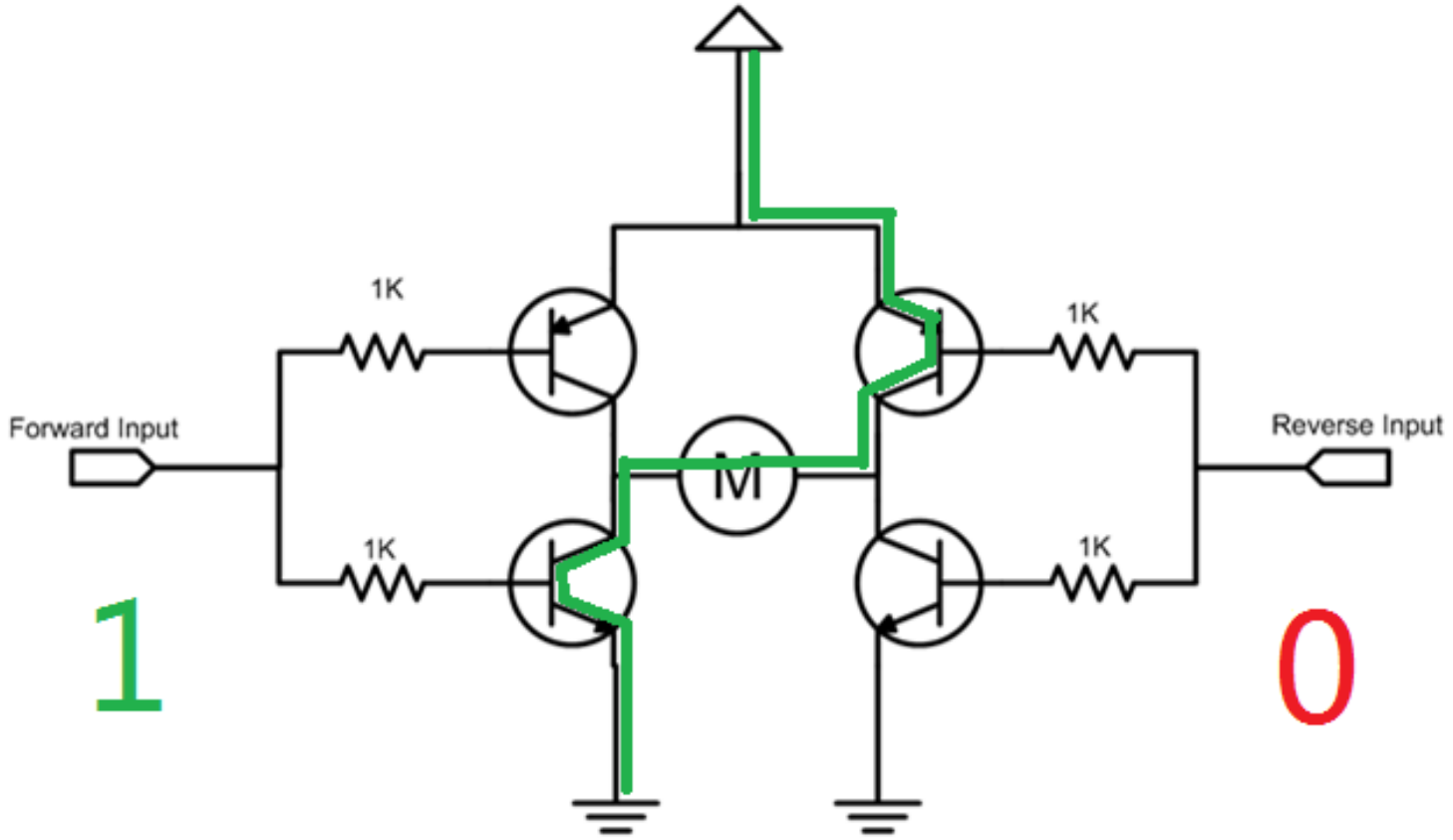
Ponte H



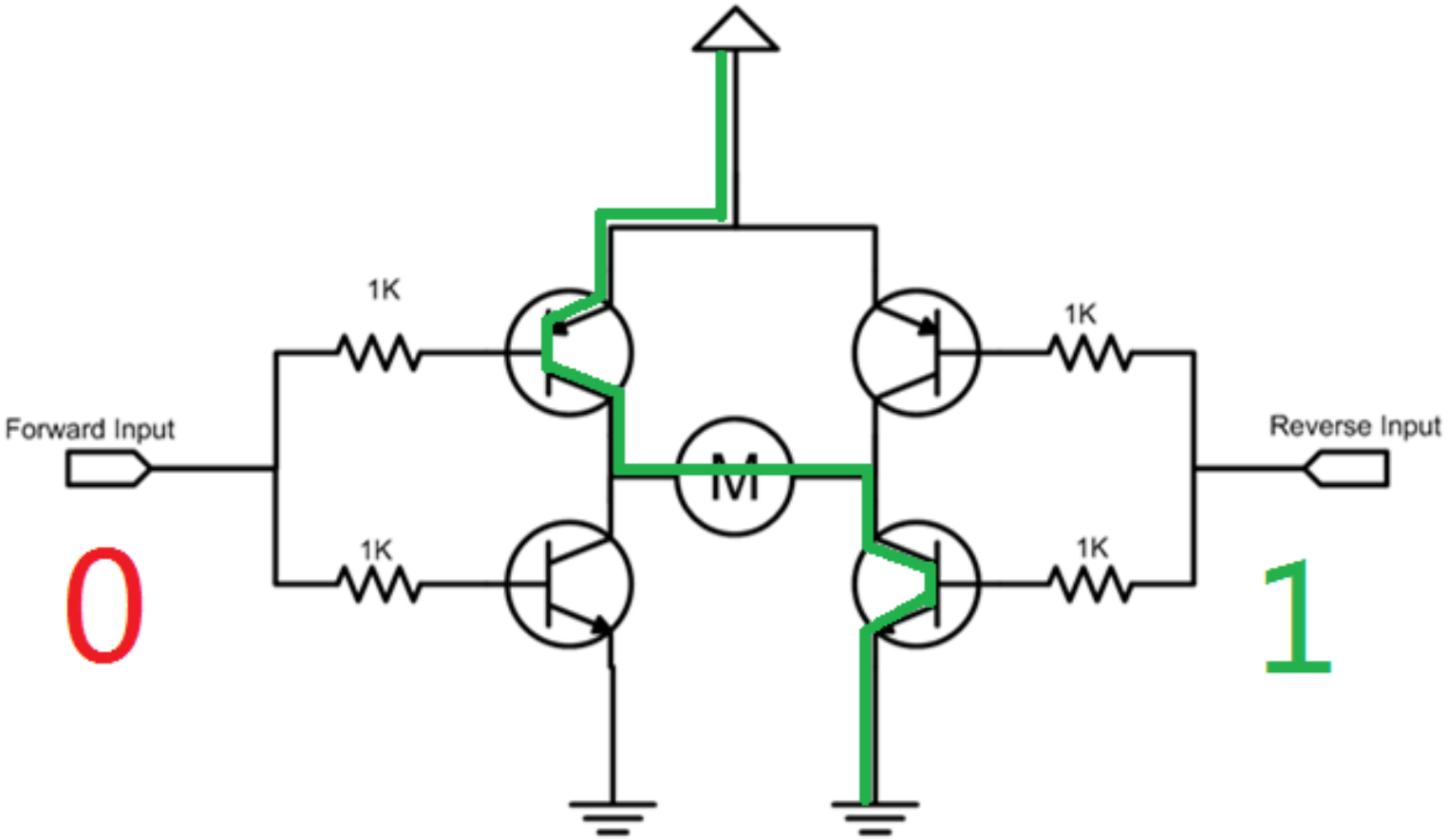
Ponte H



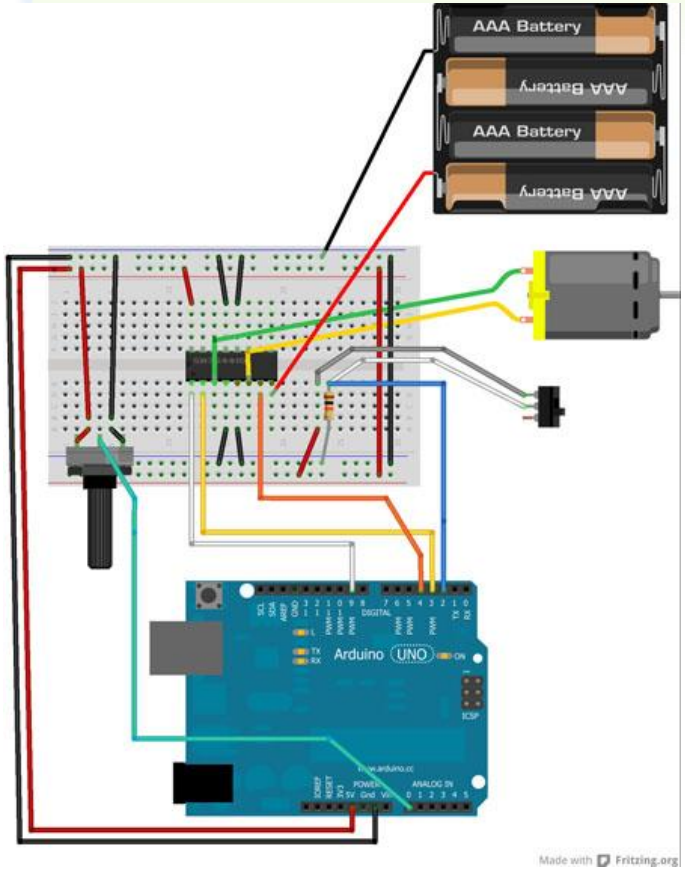
Ponte H



Ponte H

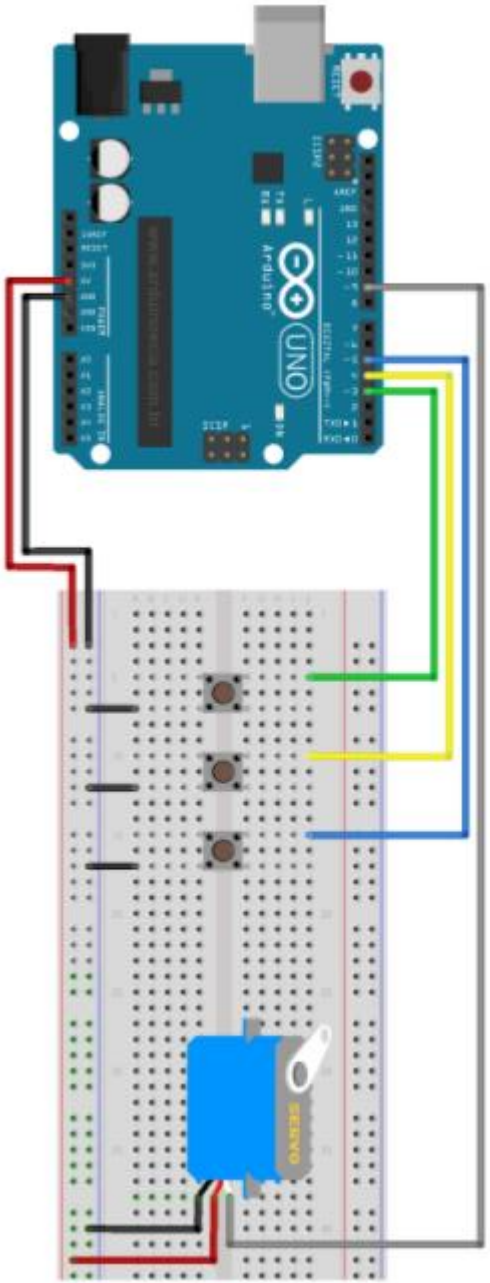


Controle do Motor CC
Ponte H – L293D




```
#define chaveMudaSentRot 2
#define motorCCEnt1 3
#define motorCCEnt2 4
#define controleL293D 9
#define controleVelocPot 0
int potVelPotencia = 0;

void setup() {
  pinMode(chaveMudaSentRot,INPUT);
  pinMode(motorCCEnt1,OUTPUT);
  pinMode(motorCCEnt2,OUTPUT);
  pinMode(controleL293D,OUTPUT);
}
void loop() {
    potVelPotencia  = analogRead(controleVelocPot) / 4;

    analogWrite(controleL293D,potVelPotencia);

   if (digitalRead(chaveMudaSentRot) == HIGH) {
   digitalWrite(motorCCEnt1,LOW);
   digitalWrite(motorCCEnt2,HIGH);
  } else {
   digitalWrite(motorCCEnt1,HIGH);
   digitalWrite(motorCCEnt2,LOW);
  }
}
```

Exemplo retirado de http://www.comofazerascoisas.com.br/



Controle do Servomotor

```
13  void setup()
14  {
15    // Pino de dados do servo conectado ao pino 9 do Arduino
16    myservo.attach(9);
17    //Define o pino como entrada
18    pinMode(ChaveEsquerda, INPUT);
19    //Aciona o resistor pull-up interno
20    digitalWrite(ChaveEsquerda, HIGH);
21    pinMode(ChaveCentral, INPUT);
22    digitalWrite(ChaveCentral, HIGH);
23    pinMode(ChaveDireita, INPUT);
24    digitalWrite(ChaveDireita, HIGH);
25  }
26
27  void loop()
28  {
29   //Le o valor da Chave Esquerda (On/Off)
30   val=digitalRead(ChaveEsquerda);
31     //Caso a chave seja pressionada, movimenta o servo
32     if(val!=1)
33      {
34         myservo.write(60);  //Move o servo para o angulo de 60 graus
35         delay(15);          //Delay para o servo atingir a posiçao
36      }
37
38   val=digitalRead(ChaveCentral);
39     if(val!=1)
40      {
41         myservo.write(90);  //Move o servo para o angulo de 90 graus
42         delay(15);
43      }
44
45   val=digitalRead(ChaveDireita);
46     if(val!=1)
47      {
48         myservo.write(120);  //Move o servo para o angulo de 120 graus
49         delay(15);
50      }
51  }
```

Exemplo retirado de http://www.comofazerascoisas.com.br/

Ver Links

- http://blog.vidadesilicio.com.br/apostila-arduino-basico/
- https://www.arduino.cc/en/Main/Software
- https://www.youtube.com/watch?v=TphX2gHBeCA
- http://fritzing.org/home/