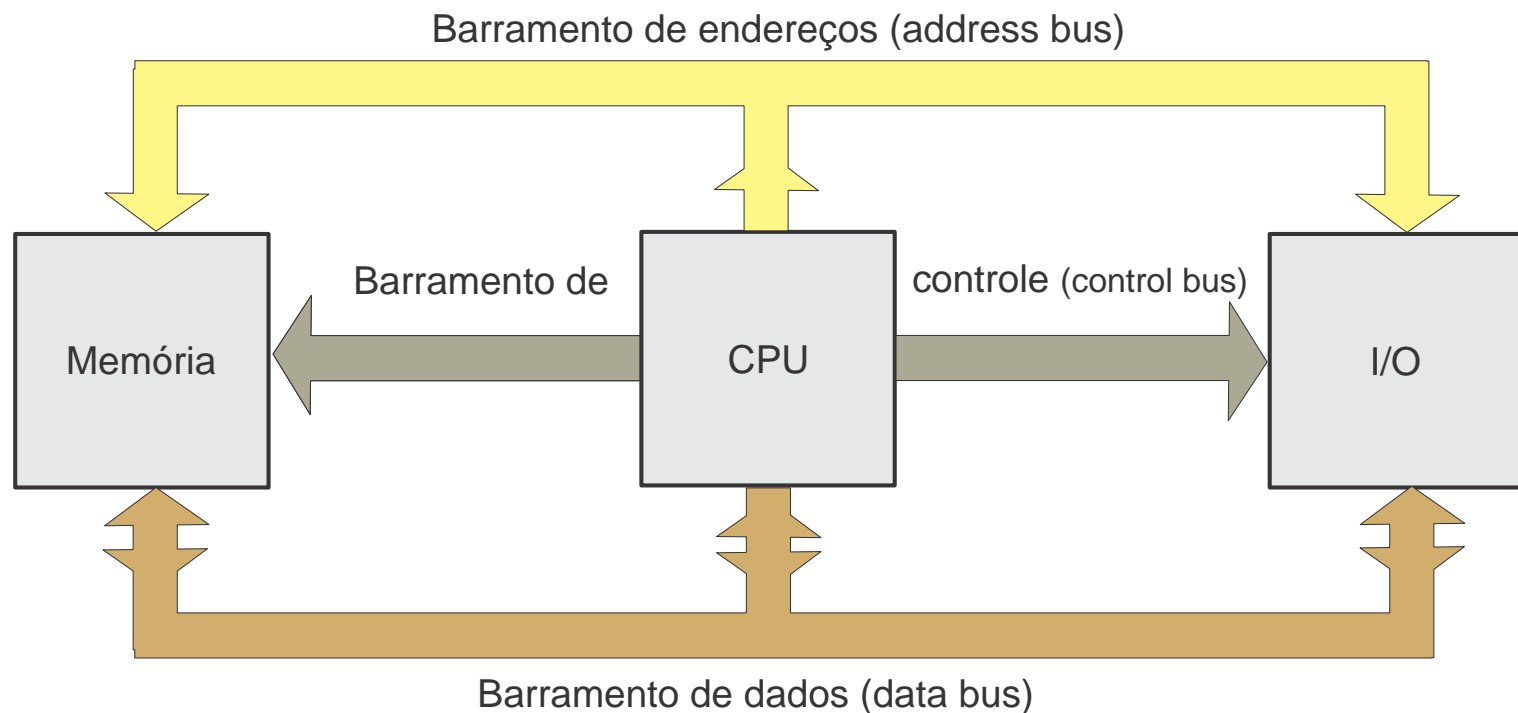


ARDUINO PRIMEIROS PASSOS

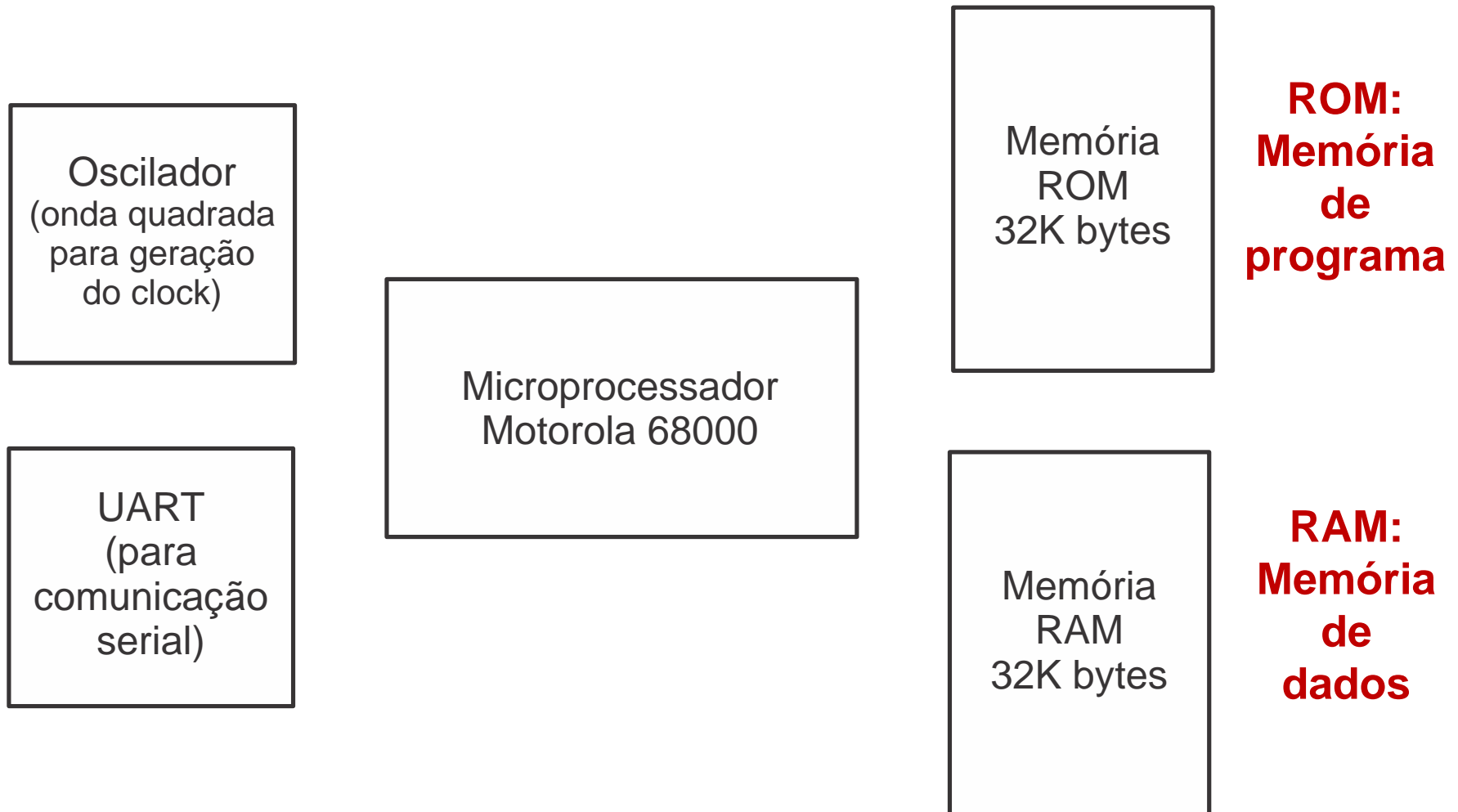
EL66J

Computador

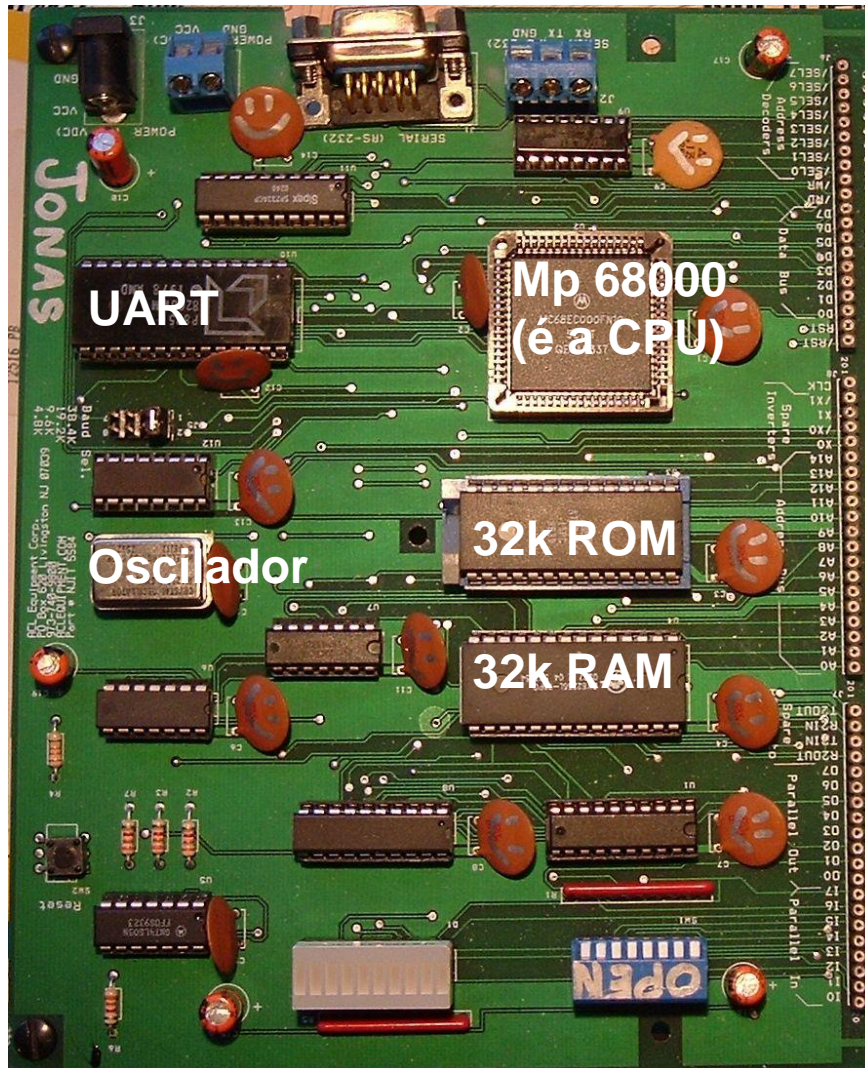
- Um computador é um sistema digital que possui uma CPU (central processing unit), memória e portas de entrada e saída (I/O). Estes três módulos comunicam-se através dos barramentos de dados, endereços e controle.



Computador • Exemplo de um computador específico.



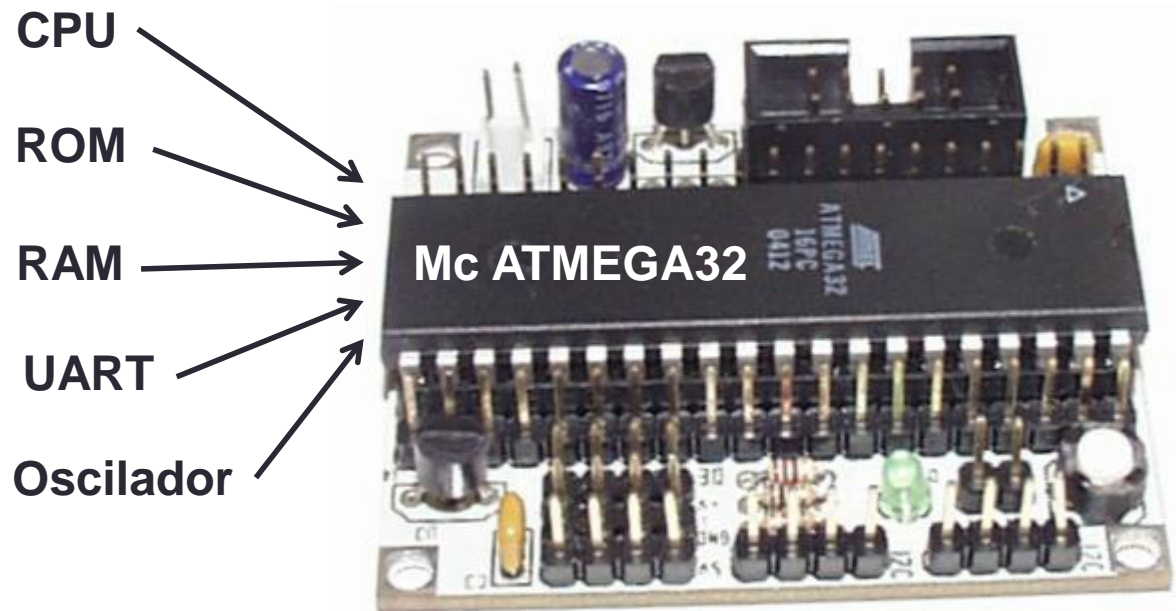
Microprocessador



- Se construído com um Microprocessador, o computador do exemplo necessitaria CIs externos de RAM, ROM, UART e Oscilador.

Microcontrolador (Mc)

- Um Mc possui RAM, ROM, UART e Oscilador embutidos.
- Por isso, um Mc também é chamado de *single-chip computer*.

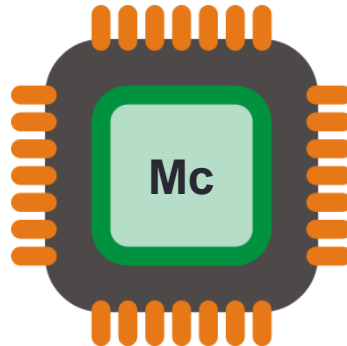
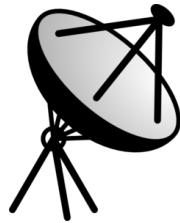
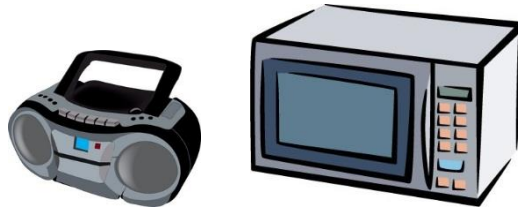


<http://www.d8apro.com/NewKits.html>

- Há muitas opções de recursos embutidos, também chamados de periféricos embutidos ou periféricos integrados.

Sistemas embarcados

- São sistemas digitais microcontrolados para **aplicações específicas.**



Tem microcontrolador pra todo lado!

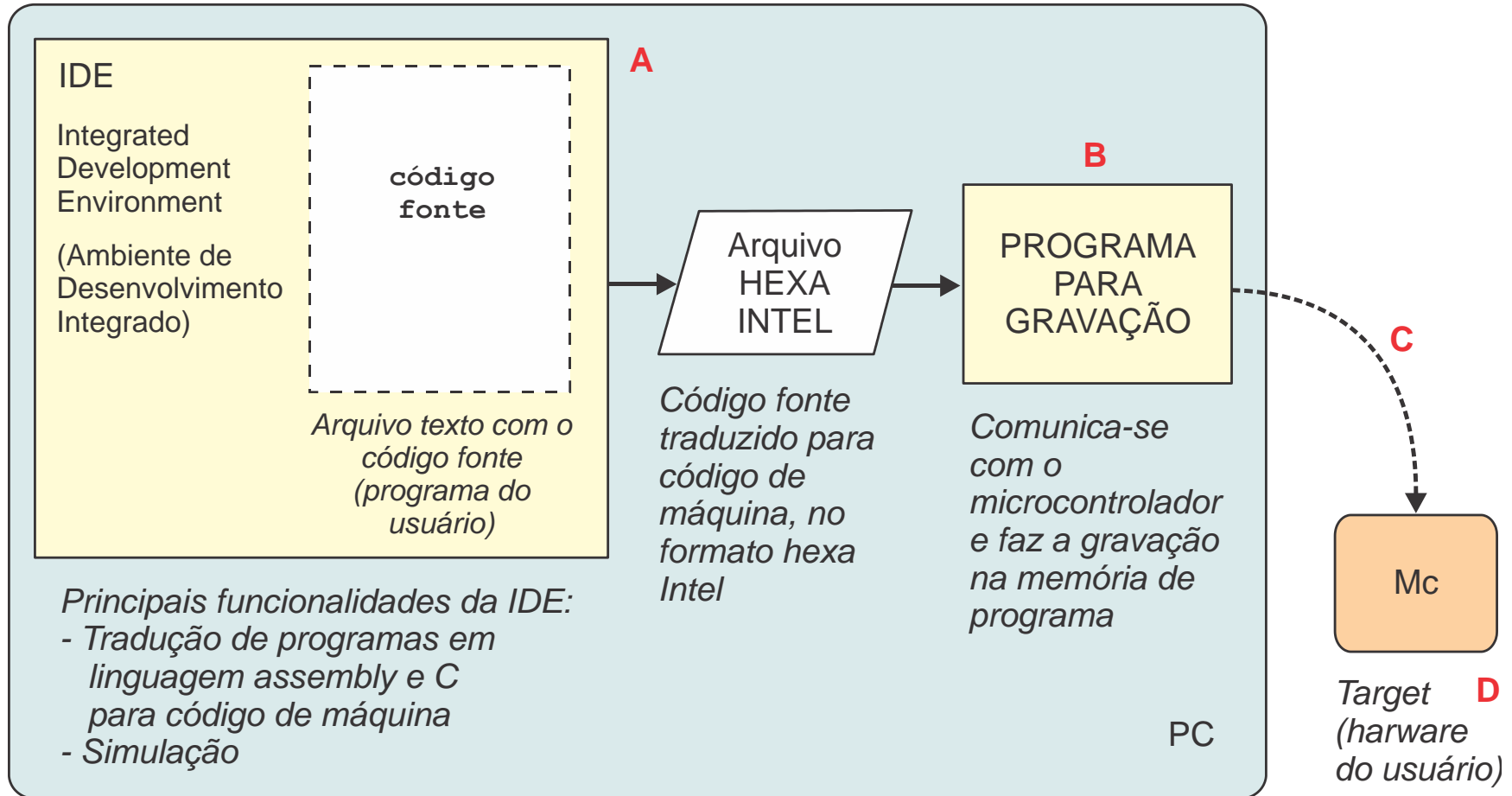
Principal critério de classificação dos microcontroladores (segmentos de mercado)

- Quanto ao número de bits de dados
 - 4 bits
 - 8 bits
 - 16 bits
 - 32 bits

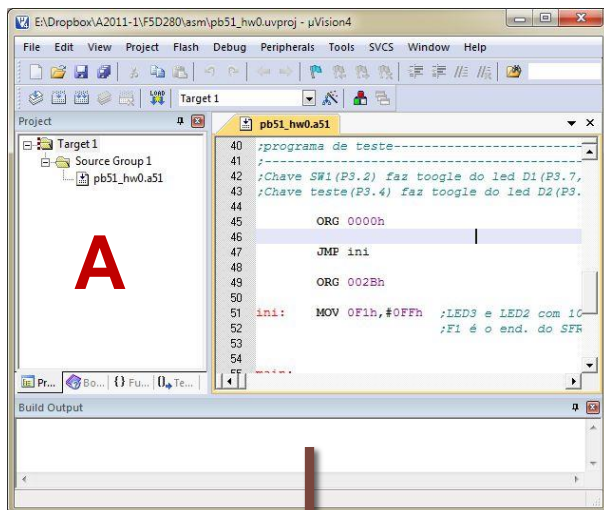
Exemplos de fabricantes



Desenvolvimento - principais passos e ferramentas

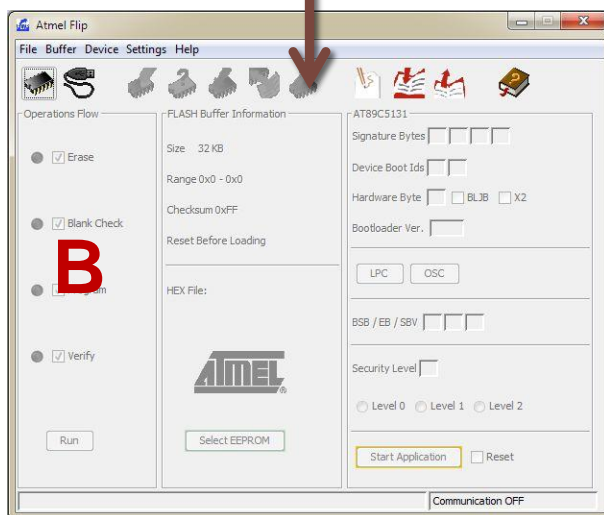


Desenvolvimento - exemplo específico



Keil ou outra IDE
(para desenvolver os programas)

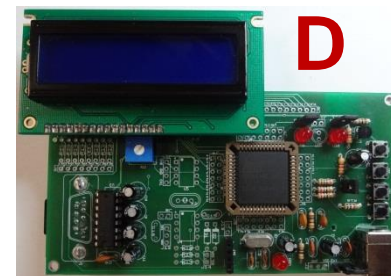
Arquivo .hex



USB

C

Target
Placa P51 USB

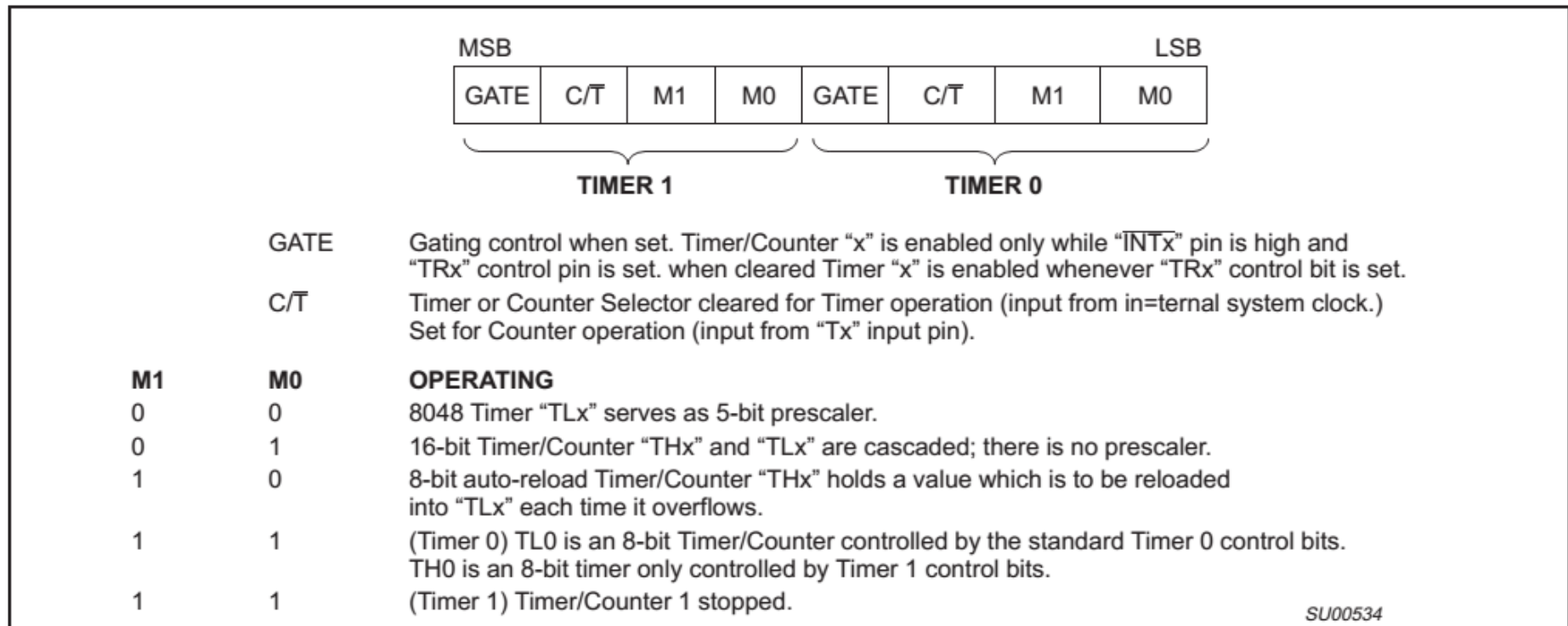


Microcontrolador
AT89C5131

Flip (para gravar os programas no Mc)

Desenvolvimento - programação

- Em geral, é necessário conhecer detalhadamente a arquitetura interna do microcontrolador, o que costuma ser muito trabalhoso!
- Por exemplo, para programar os timers/counters do microcontrolador 8051:
- Slide 1 de 4 (Philips Semiconductor, Datasheet, 80C51 family hardware description, 1997)



SU00534

Figure 6. Timer/Counter Mode Control (TMOD) Register

Desenvolvimento - programação

- Em geral, é necessário conhecer detalhadamente a arquitetura interna do microcontrolador, o que costuma ser muito trabalhoso!
- Por exemplo, para programar os timers/counters do microcontrolador 8051:
- Slide 2 de 4 (Philips Semiconductor, Datasheet, 80C51 family hardware description, 1997)

		MSB							LSB
		TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
BIT	SYMBOL	FUNCTION							
TCON.7	TF1	Timer 1 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or clearing the bit in software.							
TCON.6	TR1	Timer 1 Run control bit. Set/cleared by software to turn Timer/Counter on/off.							
TCON.5	TF0	Timer 0 overflow flag. Set by hardware on Timer/Counter overflow. Cleared by hardware when processor vectors to interrupt routine, or by clearing the bit in software.							
TCON.4	TR0	Timer 0 Run control bit. Set/cleared by software to turn Timer/Counter on/off.							
TCON.3	IE1	Interrupt 1 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.							
TCON.2	IT1	Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.							
TCON.1	IE0	Interrupt 0 Edge flag. Set by hardware when external interrupt edge detected. Cleared when interrupt processed.							
TCON.0	IT0	Interrupt 0 Type control bit. Set/cleared by software to specify falling edge/low level triggered external interrupts.							

SU00536

Figure 8. Timer/Counter Control (TCON) Register

Desenvolvimento - programação

- Em geral, é necessário conhecer detalhadamente a arquitetura interna do microcontrolador, o que costuma ser muito trabalhoso!
- Por exemplo, para programar os timers/counters do microcontrolador 8051:
- Slide 3 de 4 (Philips Semiconductor, Datasheet, 80C51 family hardware description, 1997)

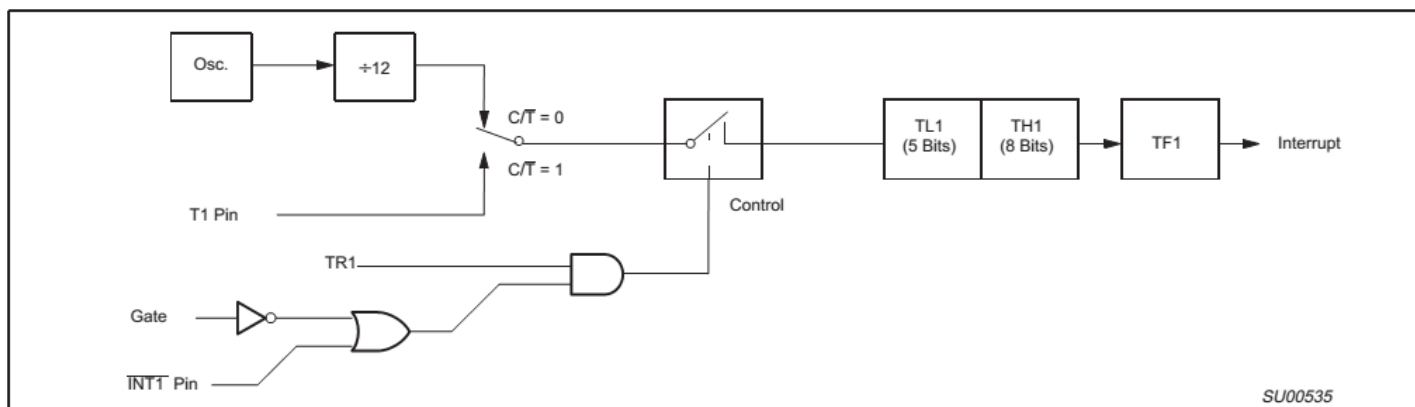


Figure 7. Timer/Counter Mode 0: 13-Bit Counter

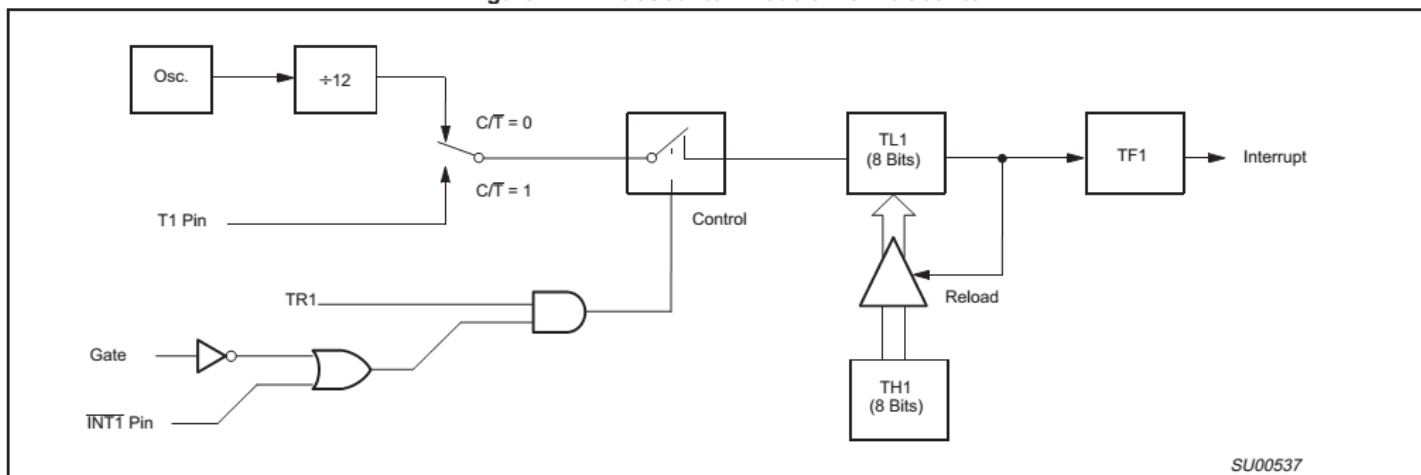


Figure 9. Timer/Counter Mode 2: 8-Bit Auto-Load

Desenvolvimento - programação

- Em geral, é necessário conhecer detalhadamente a arquitetura interna do microcontrolador, o que costuma ser muito trabalhoso!
- Por exemplo, para programar os timers/counters do microcontrolador 8051:
- Slide 4 de 4 (Philips Semiconductor, Datasheet, 80C51 family hardware description, 1997)

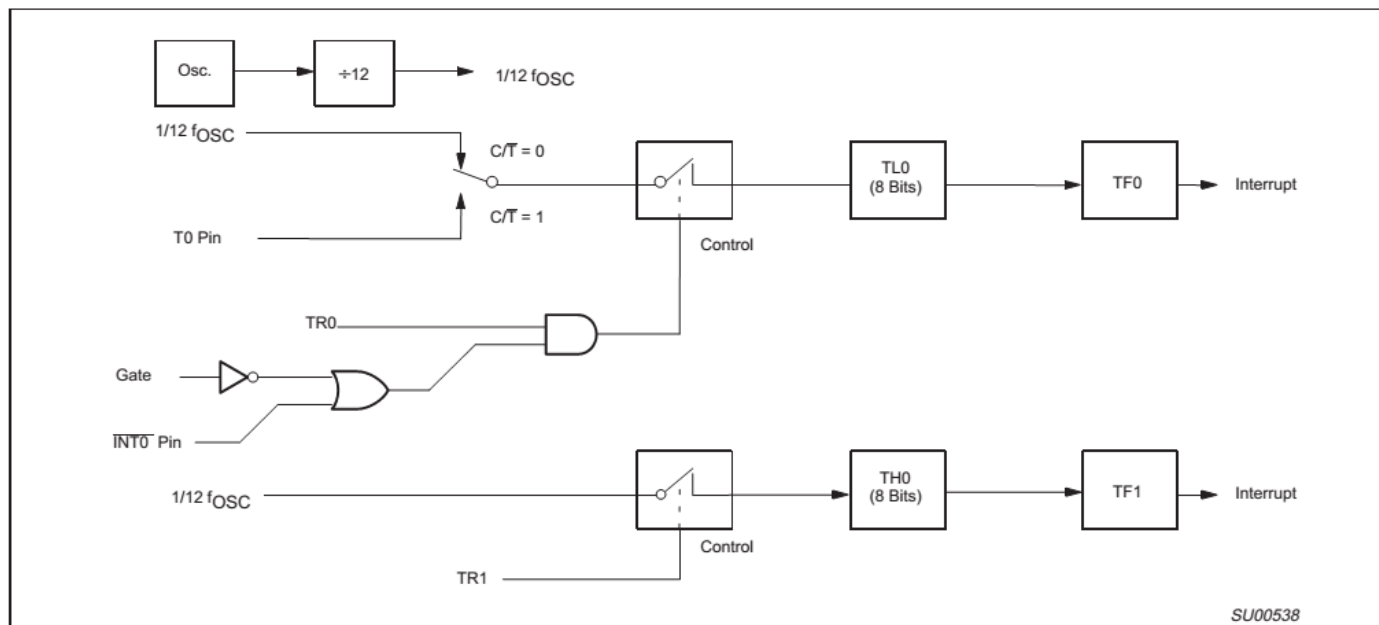
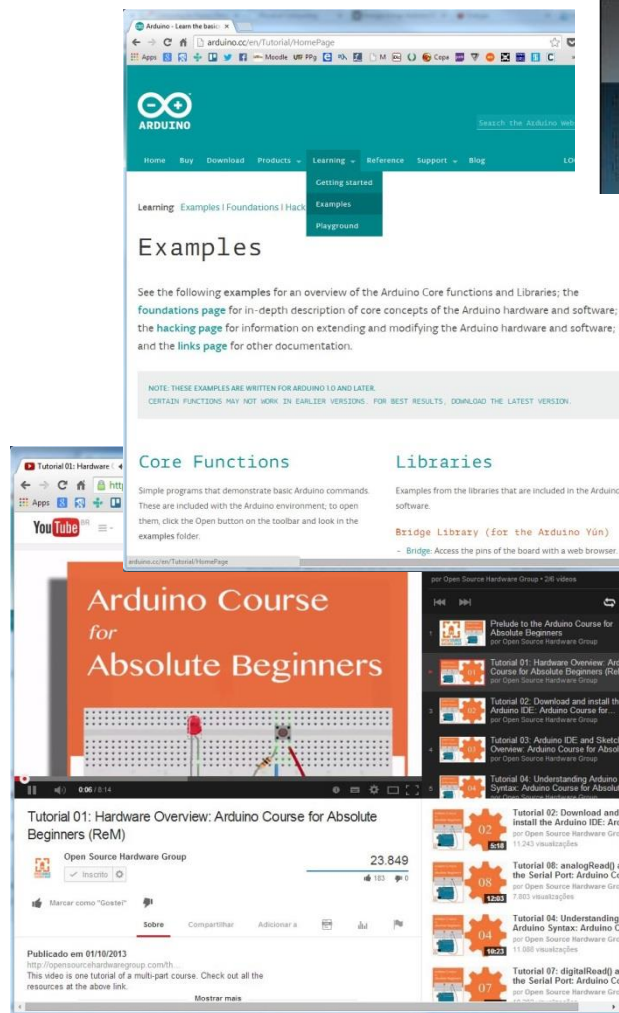
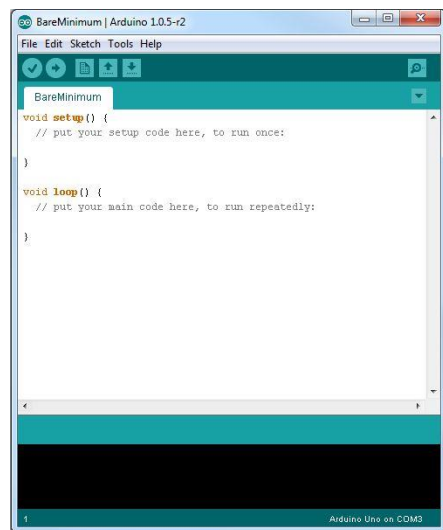


Figure 10. Timer/Counter 0 Mode 3: Two 8-Bit Counters

Arduino e similares

- “*Computação física (physical computing)*: conjunto de ferramentas e métodos para construir sistemas digitais capazes sensorear e atuar no mundo físico, mais efetivamente que um computador convencional.”
- “O Arduino é uma plataforma open-source de computação física baseada em uma placa simples com um microcontrolador e um ambiente de desenvolvimento para escrever programas para a placa.”
- “Exemplos de outras plataformas: Parallax Basic Stamp, Netmedia's BX-24, Phidgets, MIT's Handyboard, Texas Instrument's Energia.”
- “Estas plataformas transformam todos os detalhes complicados da programação de microcontroladores em um pacote fácil de usar.”

Arduino Placa + IDE + muitos recursos



<http://arduino.cc/en/Main/ArduinoBoardUno>
<http://arduino.cc/en/Tutorial/HomePage>
http://www.amazon.com/s/ref=nb_sb_noss_1?url=search-alias%3Dstripbooks&field-keywords=arduino&sprefix=ardui%2Cstripbooks&rh=i%3Astripbooks%2Ck%3Aarduino
https://www.youtube.com/watch?v=09zfRaLEasY&index=2&list=PLYutciIGBqC34bfjBdYch49oyU-B_tH

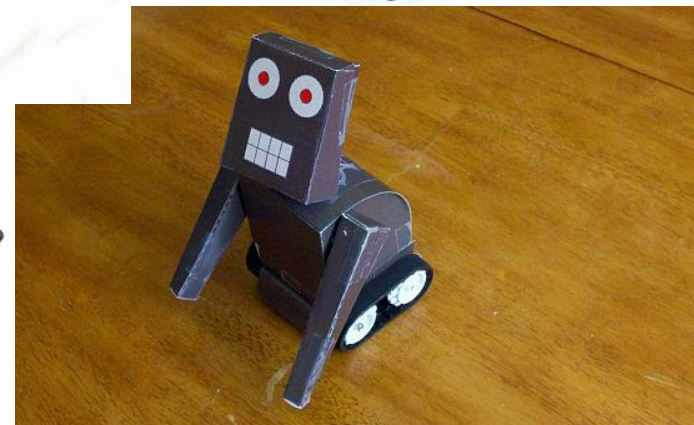
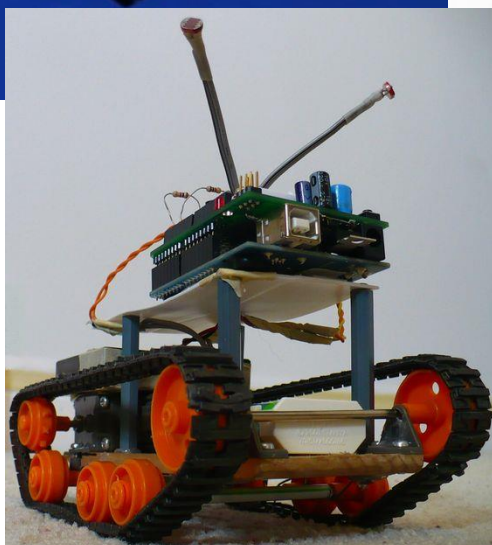
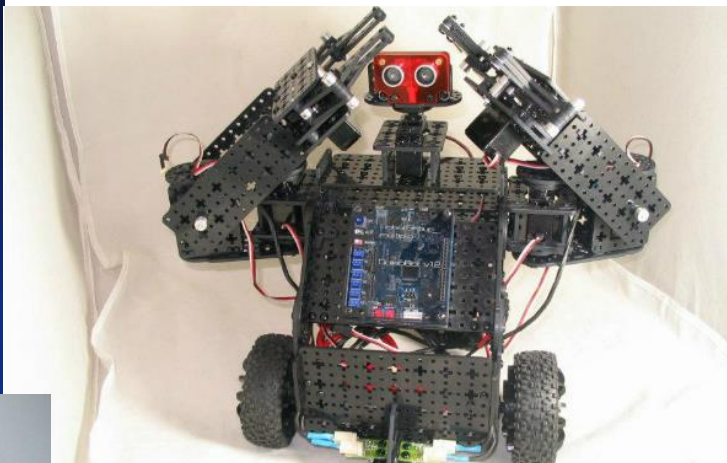
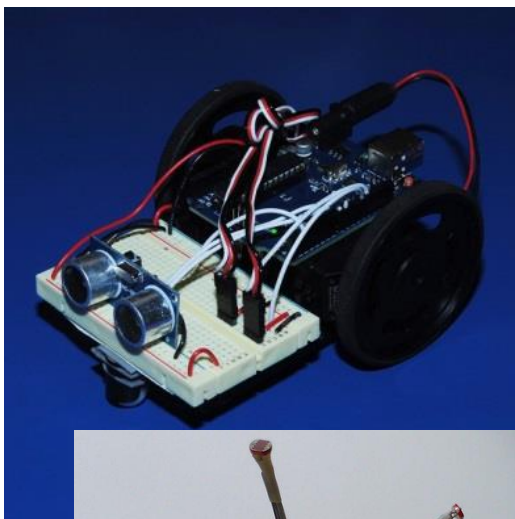
Arduino – exemplos de projetos

Mande um tweet para a sua cafeteira



Arduino – exemplos de projetos

Robozinhos de todos os sabores



<http://www.instructables.com/id/Tweet-a-Pot-Twitter-Enabled-Coffee-Pot/>
<http://www.robotshop.com/en/dfrobotshop-rover-20-arduino-mecanum-robot-basic.html>
<http://www.instructables.com/file/FL5YH1ZGLE1MKLC>
<http://www.roboticmagazine.com/toys-fun/n8-kit-build-all-the-robots-you-can-imagine>
<http://duino4projects.com/beginners-guide-to-building-arduino-robots-with-bluetooth-and-android/>
<http://arduino.cc/en/Main/Robot>

Arduino – exemplos de projetos

Pedale por aí em segurança



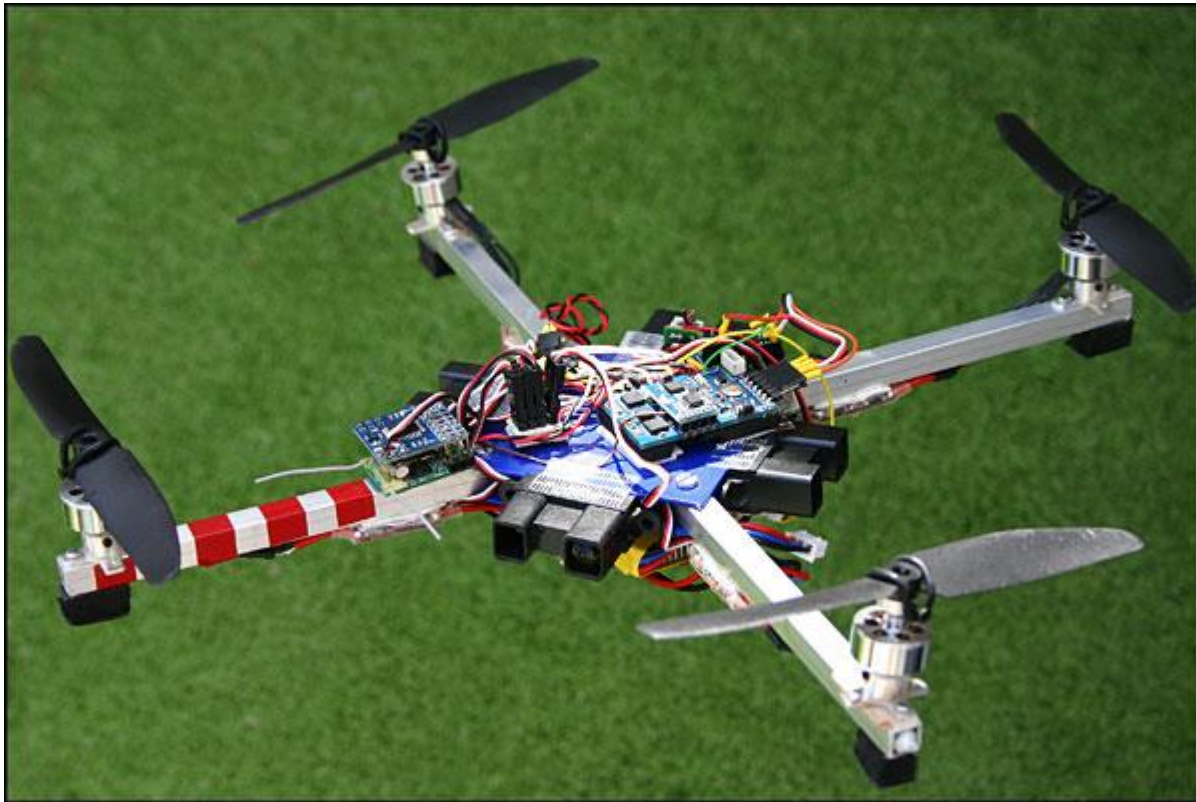
Arduino – exemplos de projetos

Cortador de grama por controle remoto



Arduino – exemplos de projetos

Quadcopter



Arduino – exemplos de projetos

Capacete do Daft Punk



Arduino – exemplos de projetos

Dispenser de comida de gato



Arduino – exemplos de projetos

Robo que sobe em árvores



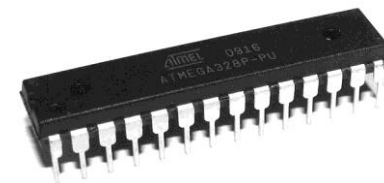
Arduino – exemplos de projetos

Cavalinho que cospe fogo

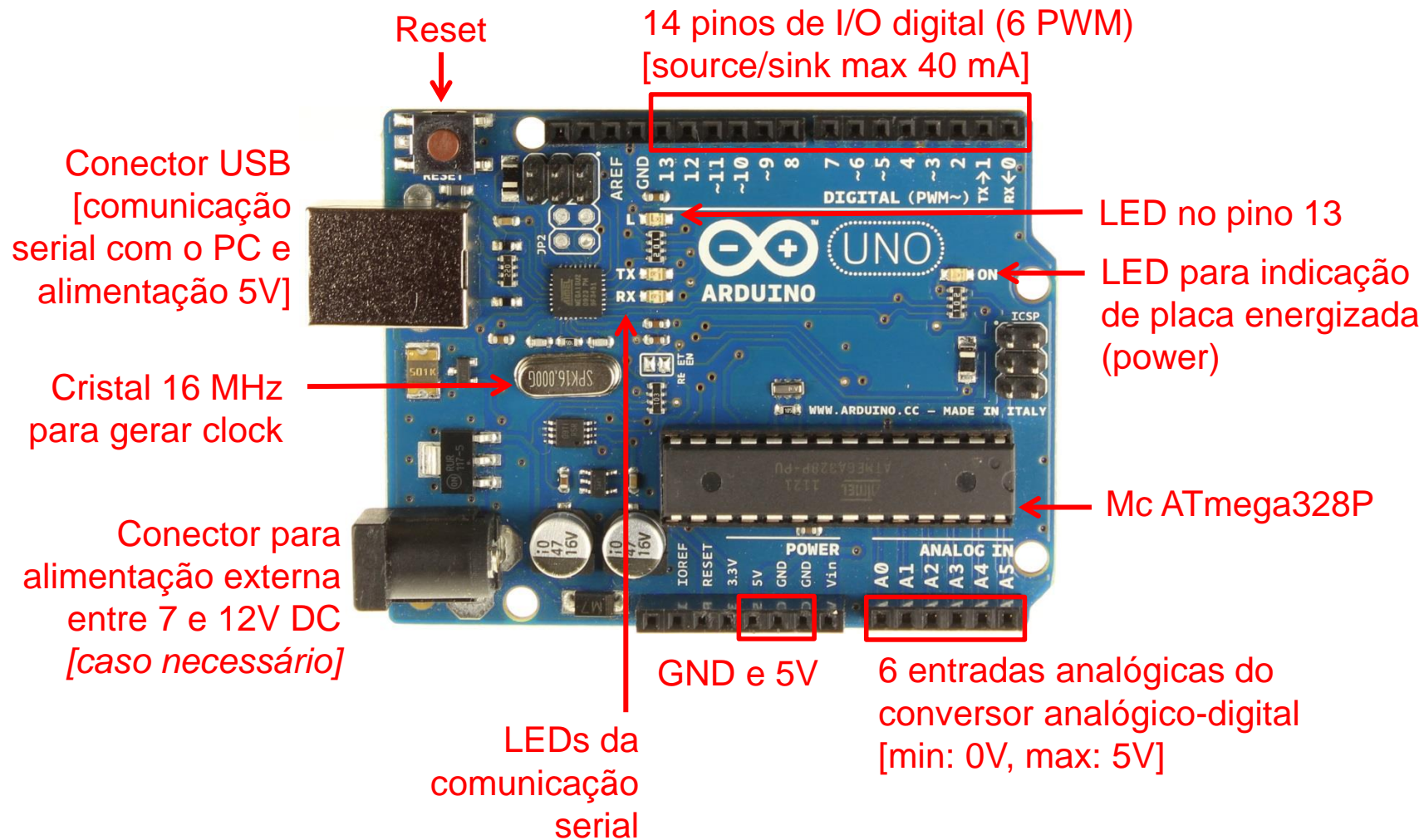


Arduino Uno

- O Arduino Uno é baseado no microcontrolador Atmel, família AVR, modelo ATmega328.
- Principais características do microcontrolador:
 - ATmega328P-PU: encapsulamento DIP 28 pinos
 - 8 bits
 - Clock de até 20 MHz
 - Alimentação 1,8 a 5,5 V
 - 32 KBytes de memória de programa Flash
 - 2 KBytes de memória de dados (RAM)
 - 23 pinos de I/O
 - Conversor analógico-digital de 6 canais, 10 bits
 - PWM de 6 canais

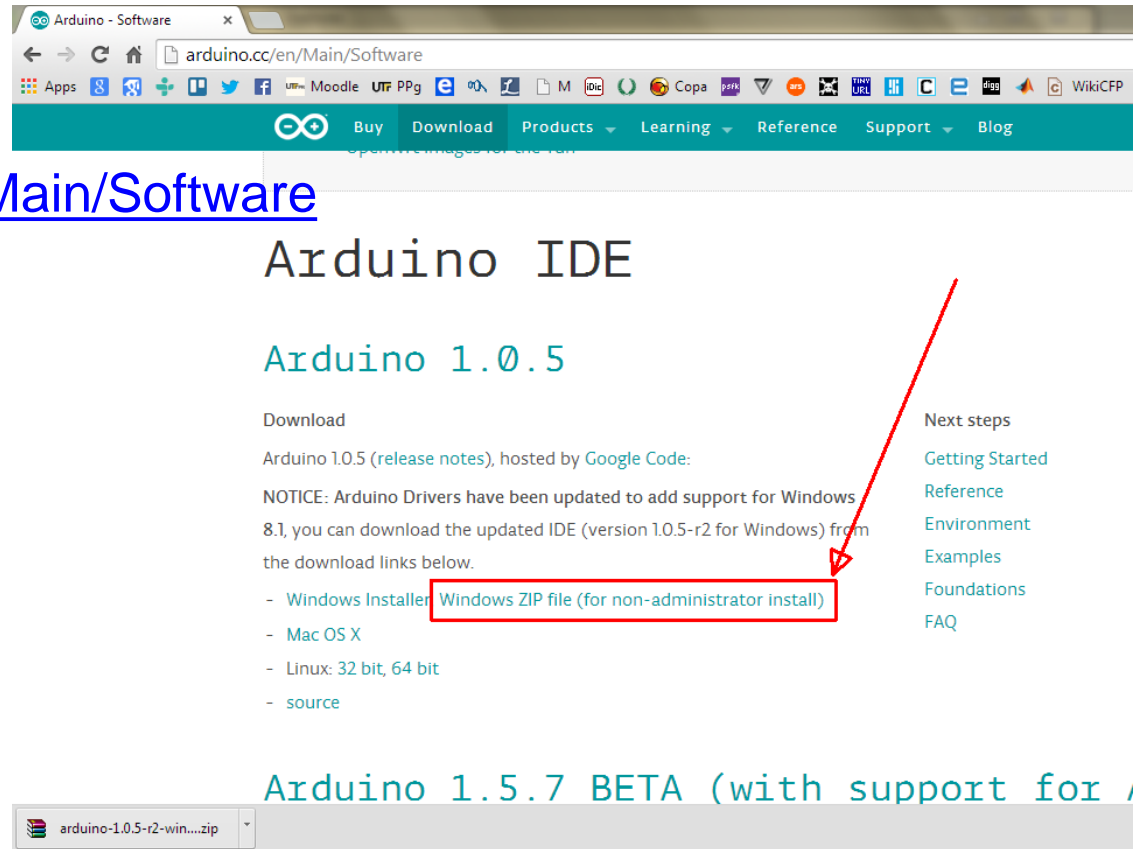


Arduino Uno



Instalando o IDE (software do Arduino)

1. Download em <http://arduino.cc/en/Main/Software>



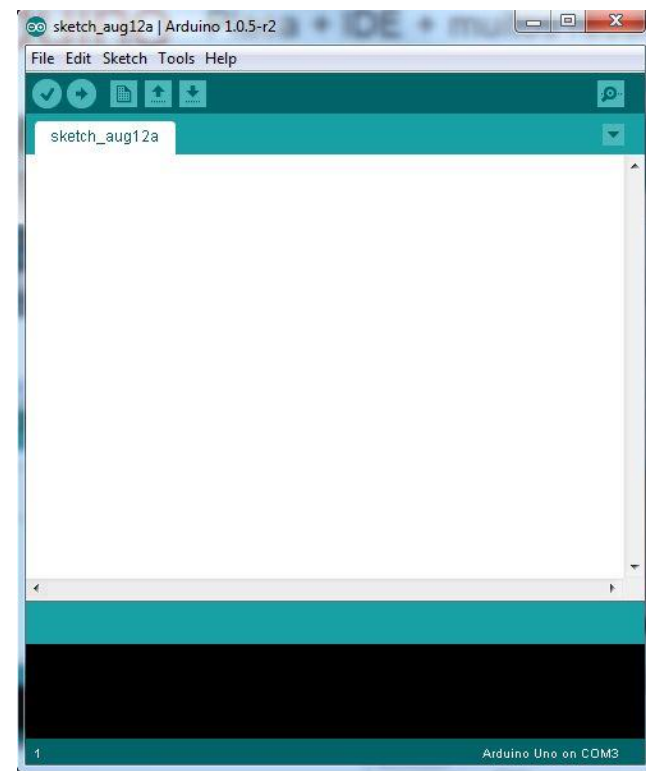
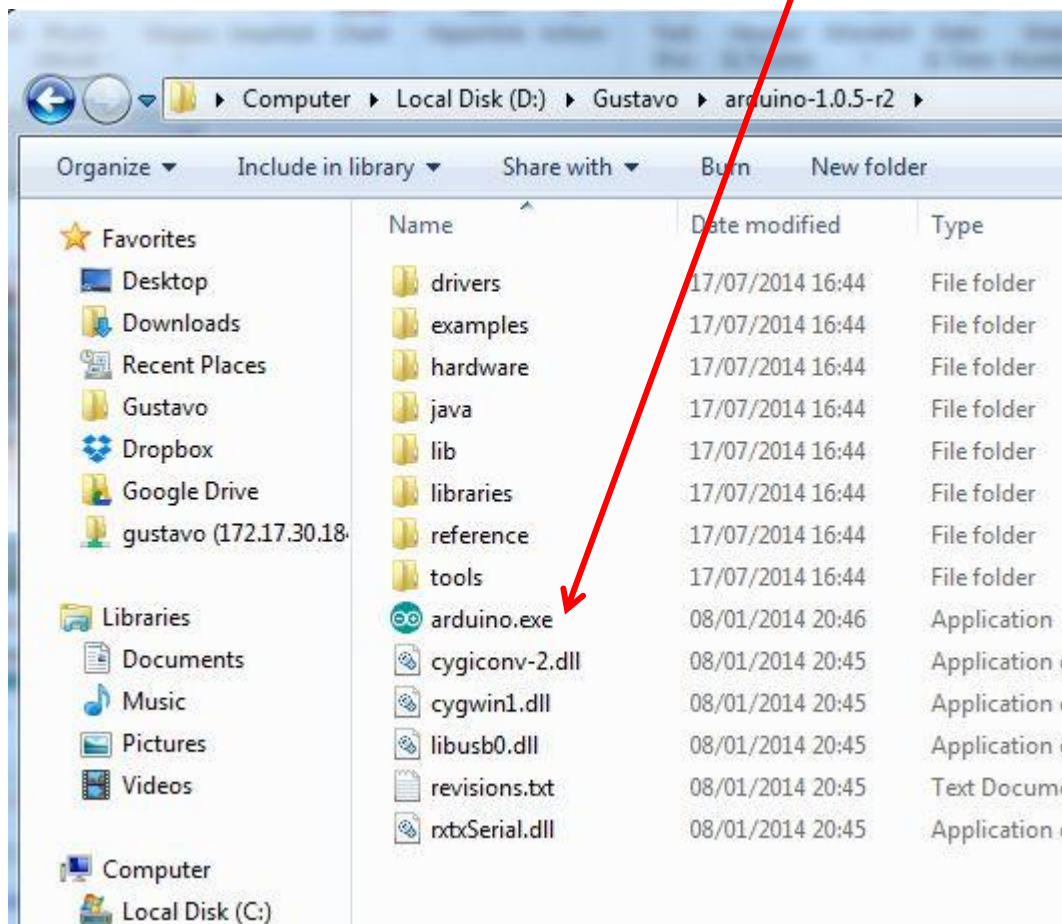
- Ou link direto da versão para Windows:
<http://arduino.googlecode.com/files/arduino-1.0.5-r2-windows.zip>

Instalação o IDE (software do Arduino)

2. Descompactar [em um lugar que vc ache depois!].
3. Conectar a placa do Arduino ao computador com o cabo USB. Aguarde...
4. Vai aparecer uma mensagem do Windows de dispositivo não encontrado. Não mexer!
5. Abrir o Control panel → System → Device manager.
6. Procurar um 'Unknown device' ou um 'Arduino UNO (COMxx)'. Pode estar em 'Ports' ou 'Other devices'.
7. Clicar com o botão direito → Update driver software → Browse my computer.
8. Navegar até o folder descompactado e lá dentro → folder 'Drivers' (é só deixar este folder selecionado) → Ok [o objetivo é fazer o Windows enxergar o 'arduino.inf'. Nesse folder só tem este arquivo. O Windows vai pegar ele] → Next
9. Clicar em 'Install' ou algo parecido na tela de confirmação.
10. Para conferir: no Control Panel → System → Device Manager → Ports há agora um 'Arduino UNO COM3'.

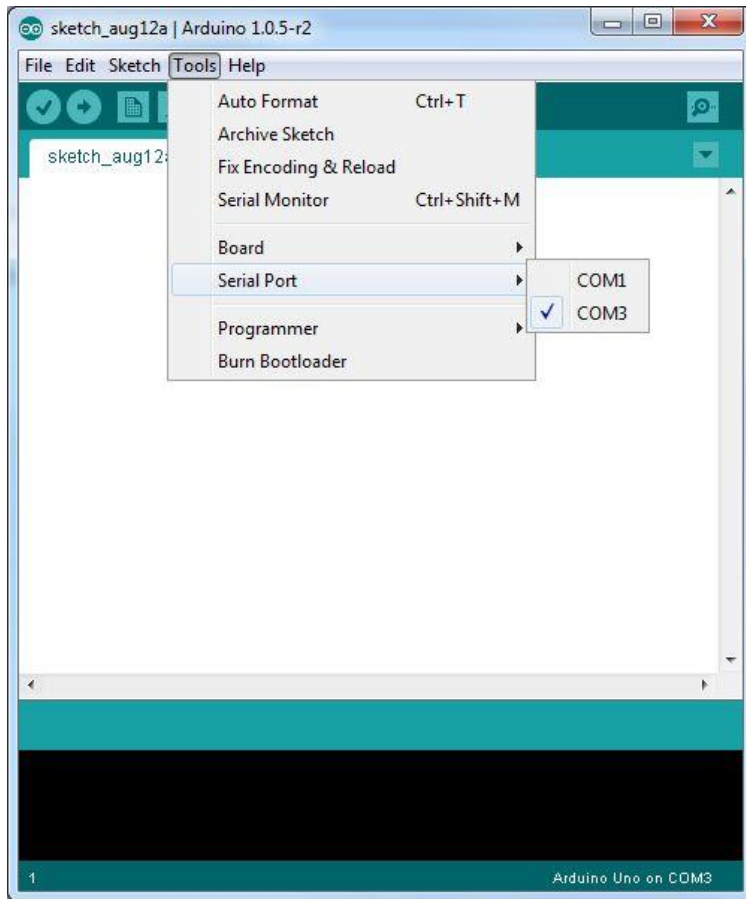
Abrindo o IDE

- Para abrir o IDE: executar o arquivo 'arduino.exe' dentro da pasta descompactada. Double click

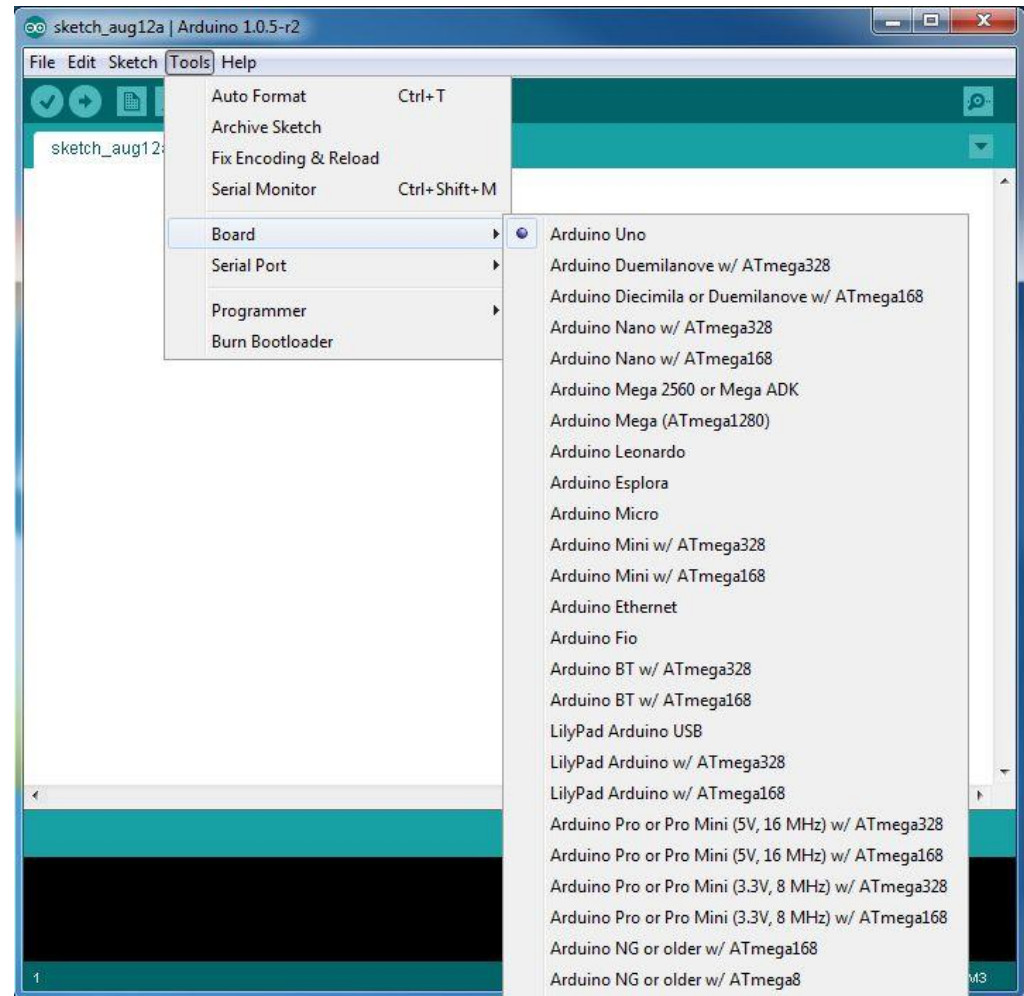


Configurando

1. Tools → Serial Port → COM3

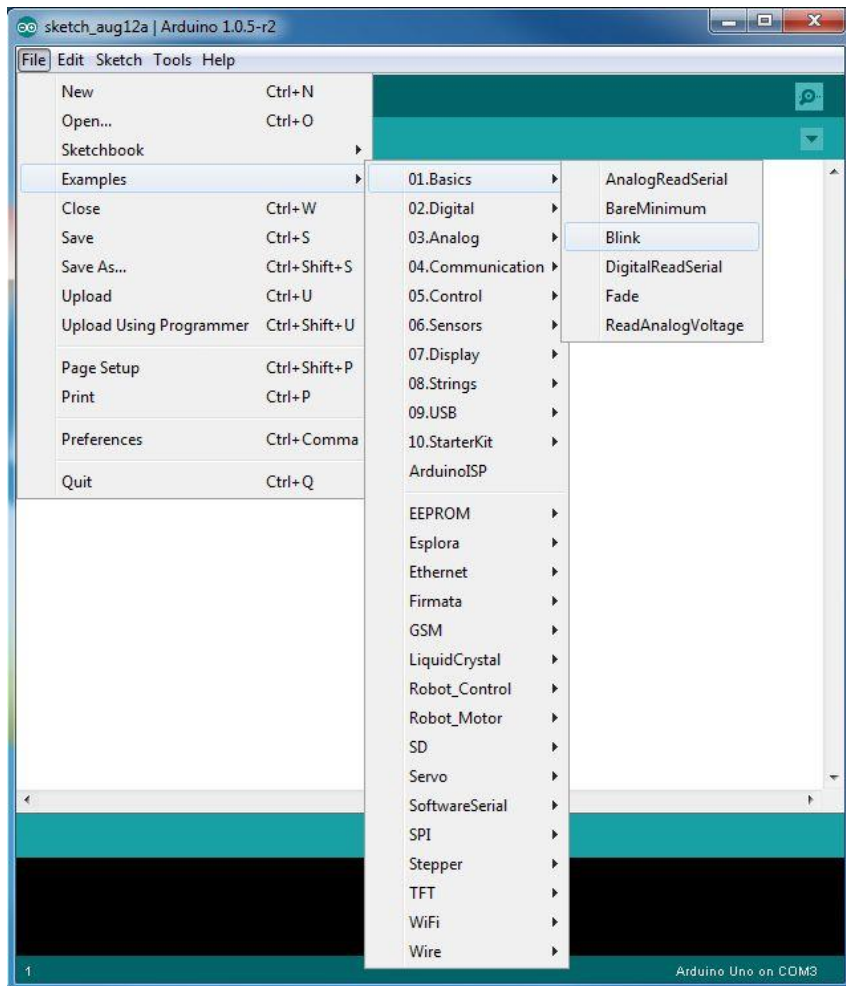


2. Tools → Board → Arduino Uno

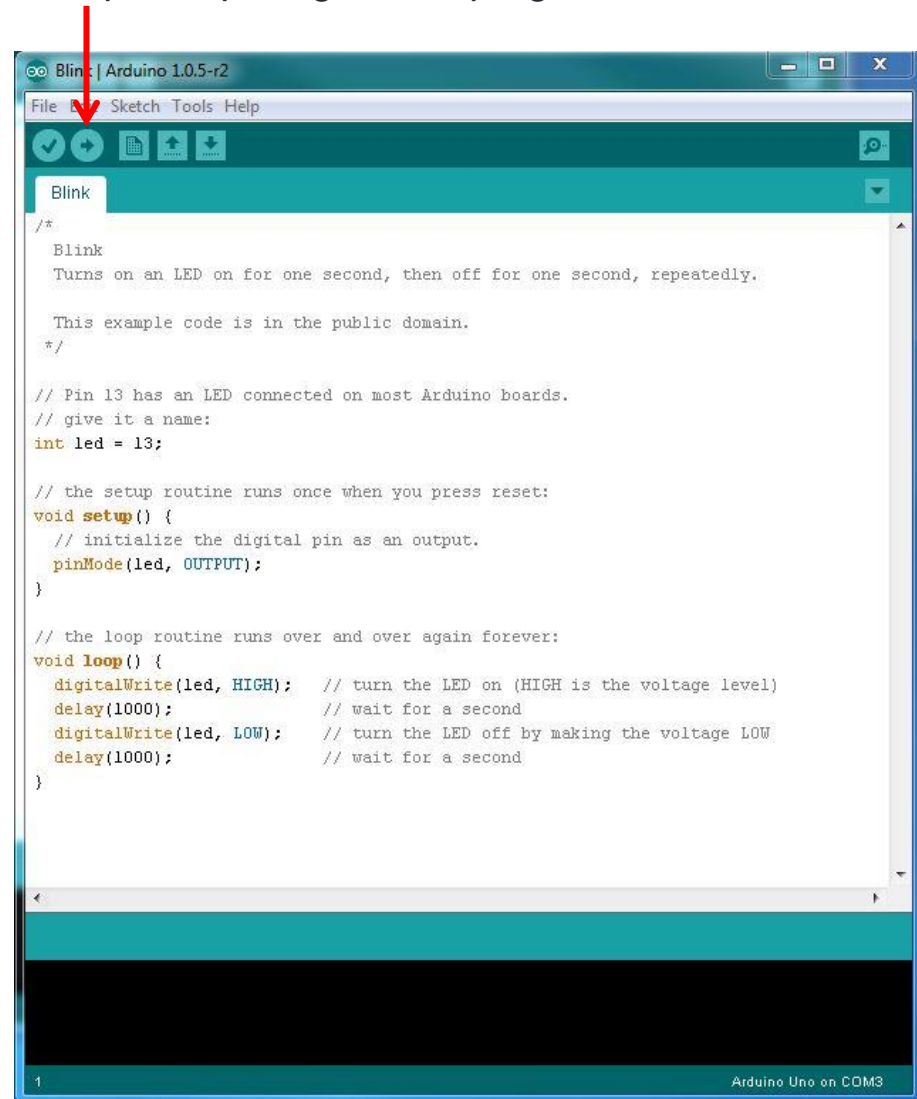


Testando

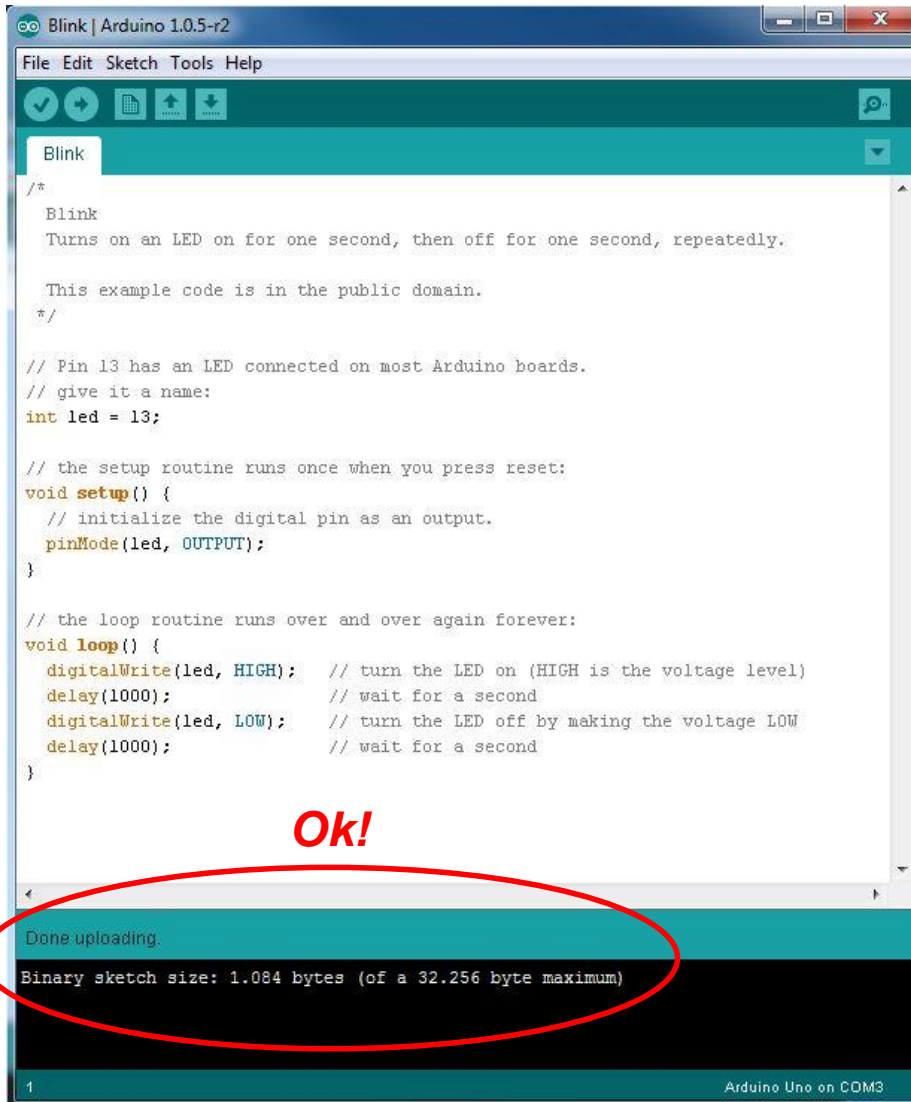
1. File → Examples → 01.Basics → Blink



2. Conectar o Arduino na USB e clicar no 'Upload' para gravar o programa no Arduino



Testando



```
/*
Blink
Turns on an LED on for one second, then off for one second, repeatedly.

This example code is in the public domain.
*/

// Pin 13 has an LED connected on most Arduino boards.
// give it a name:
int led = 13;

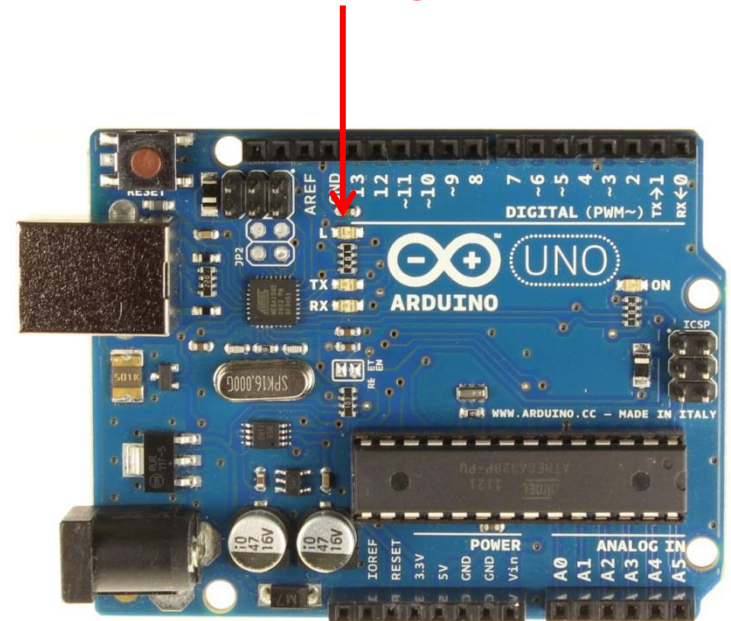
// the setup routine runs once when you press reset:
void setup() {
  // initialize the digital pin as an output.
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000);             // wait for a second
  digitalWrite(led, LOW);  // turn the LED off by making the voltage LOW
  delay(1000);             // wait for a second
}
```

Done uploading.

Binary sketch size: 1.084 bytes (of a 32.256 byte maximum)

Blinking :-)



Entendendo o Sketch *Blink.ino*

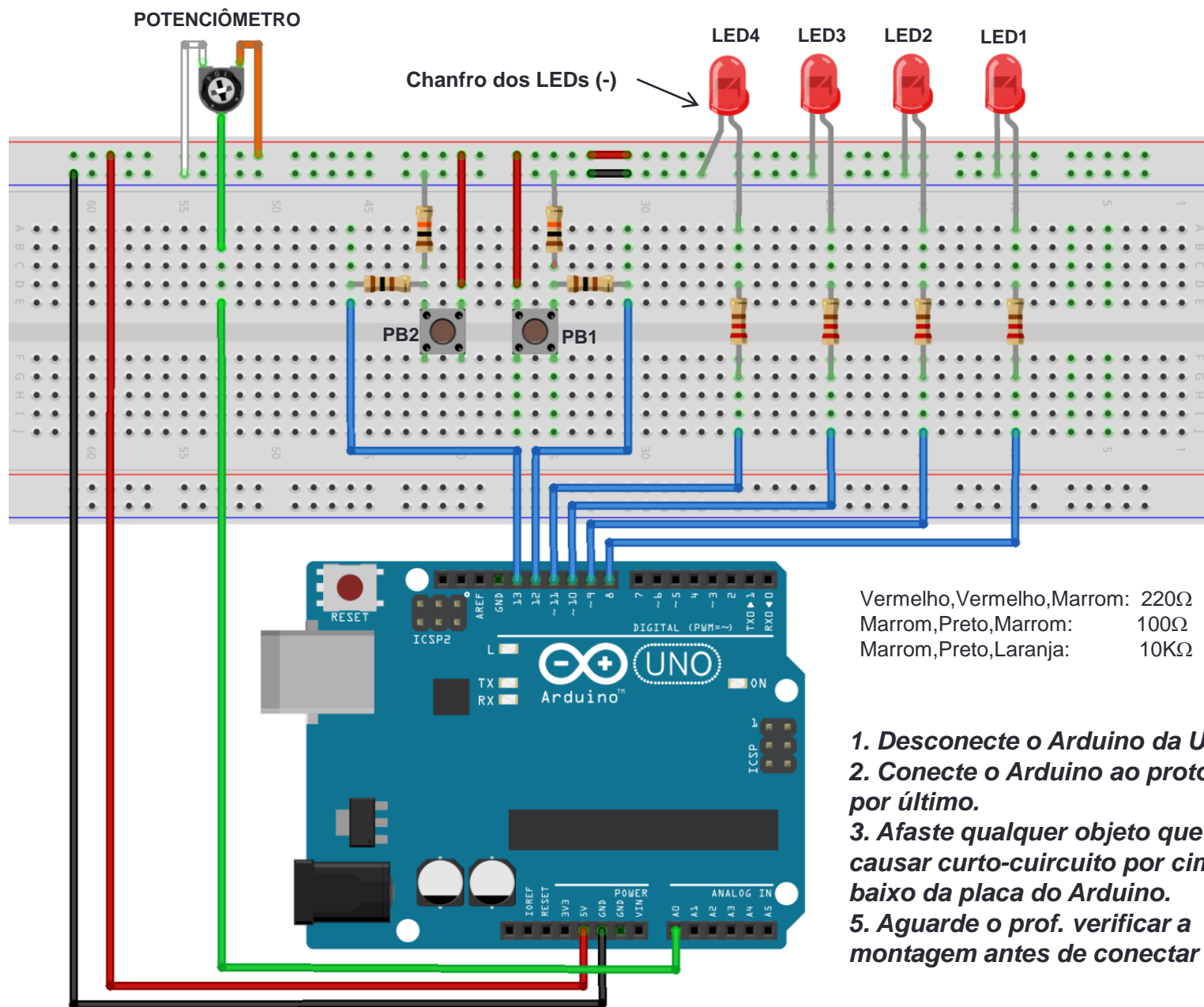
```
/* Blink
   Turns on an LED on for one second, then off for one second, repeatedly.
   This example code is in the public domain. */

// Pin 13 has an LED connected on most Arduino boards. Give it a name:
int led = 13;

// the setup routine runs once when you press reset:
void setup()
{
    // initialize the digital pin as an output.
    pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop()
{
    digitalWrite(led, HIGH);    // turn the LED on (HIGH is the voltage level)
    delay(1000);                // wait for a second
    digitalWrite(led, LOW);     // turn the LED off by making the voltage LOW
    delay(1000);                // wait for a second
}
```





1. Desconecte o Arduino da USB.
2. Conecte o Arduino ao protoboard por último.
3. Afaste qualquer objeto que possa causar curto-circuito por cima e por baixo da placa do Arduino.
5. Aguarde o prof. verificar a montagem antes de conectar na USB.

Sketch01

```
//Sketch01
//Pisca LED1

const int LED1 = 8; //LED1 no pino 8

void setup()
{
  pinMode(LED1, OUTPUT); //configura pino como saída
}

void loop()
{
  digitalWrite(LED1, HIGH); //liga o led
  delay(1000);               //espera 1 seg
  digitalWrite(LED1, LOW);   //desliga o led
  delay(1000);               //espera 1 seg
}
```

Com o *Sketch01* aberto: Save As... → *Sketch02*

```
//Sketch02
```

```
//Pisca LED1, LED2, LED3, LED4 simultaneamente
```

Sketch03

```
//Sketch03
//Pisca LED1, LED2, LED3, LED4 simultaneamente
//Usa array e 'for'
const int LEDS[] = {8,9,10,11}; //array de pinos para leds
void setup(){
    //'for' para configurar pinos como saída
    for(int i=0; i<4; i++){
        pinMode(LEDS[i], OUTPUT);
    }
}
void loop(){
    //'for' para ligar os leds
    for(int i=0; i<4; i++){
        digitalWrite(LEDS[i], HIGH);
    }
    delay(1000); //espera 1 seg
    //'for' para desligar os leds
    for(int i=0; i<4; i++){
        digitalWrite(LEDS[i], LOW);
    }
    delay(1000); //espera 1 seg
}
```

Com o *Sketch03* aberto: Save As... → *Sketch04*

```
//Sketch04  
//Liga os leds sequencialmente indo e voltando:  
//LED1->LED2->LED3->LED4->LED3...
```

Sketch05

```
//Sketch05
//Liga leds sequencialmente.
//Ao segurar PB1 inverte a sequencia.

const int LEDS[] = {8,9,10,11};
//pushbutton 1 no pino 12
const int PB1 = 12;
//para receber valor do pushbutton
int state = 0;

void setup(){
  for(int i=0; i<4; i++){
    pinMode(LEDS[i], OUTPUT);
  }
  //pino PB1 como entrada
  pinMode(PB1, INPUT);
}
```



```
void loop(){

  state = digitalRead(PB1); //lê PB1

  if(state == HIGH){
    //do menor para o maior
    for(int i=0; i<4; i++){
      digitalWrite(LEDS[i], HIGH);
      delay(200); //espera 0,2 seg
      digitalWrite(LEDS[i], LOW);
    }
  }
  else{
    //do maior para o menor
    for(int i=3; i>=0; i--){
      digitalWrite(LEDS[i], HIGH);
      delay(200); //espera 0,2 seg
      digitalWrite(LEDS[i], LOW);
    }
  }
}
```

Com o *Sketch05* aberto: Save As... → *Sketch06*

```
//Sketch06  
//Liga leds sequencialmente.  
//Ao segurar PB1 inverte a sequencia.  
//Ao segurar PB2 altera o tempo.
```


Sketch07

```
//Sketch07
//Um toque em PB1
//inverte o estado do LED1
//(toggle)

const int LED1 = 8;
const int PB1 = 12;
//estado atual do pb
int currState = 0;
//estado anterior do pb
int prevState = 0;
//0->LED off, 1->LED on

int value = 0;
void setup() {
    pinMode(LED1, OUTPUT);
    pinMode(PB1, INPUT);
}
```



```
void loop(){

    currState = digitalRead(PB1); // lê

    // testa transição
    if ((currState == HIGH) && (prevState == LOW)){
        value = 1 - value; //inverte estado do LED
        delay(20); //debounce
    }

    //agora o estado atual ficou velho!
    prevState = currState;

    if (value == 1) {
        digitalWrite(LED1, HIGH);
    } else {
        digitalWrite(LED1, LOW);
    }
}
```

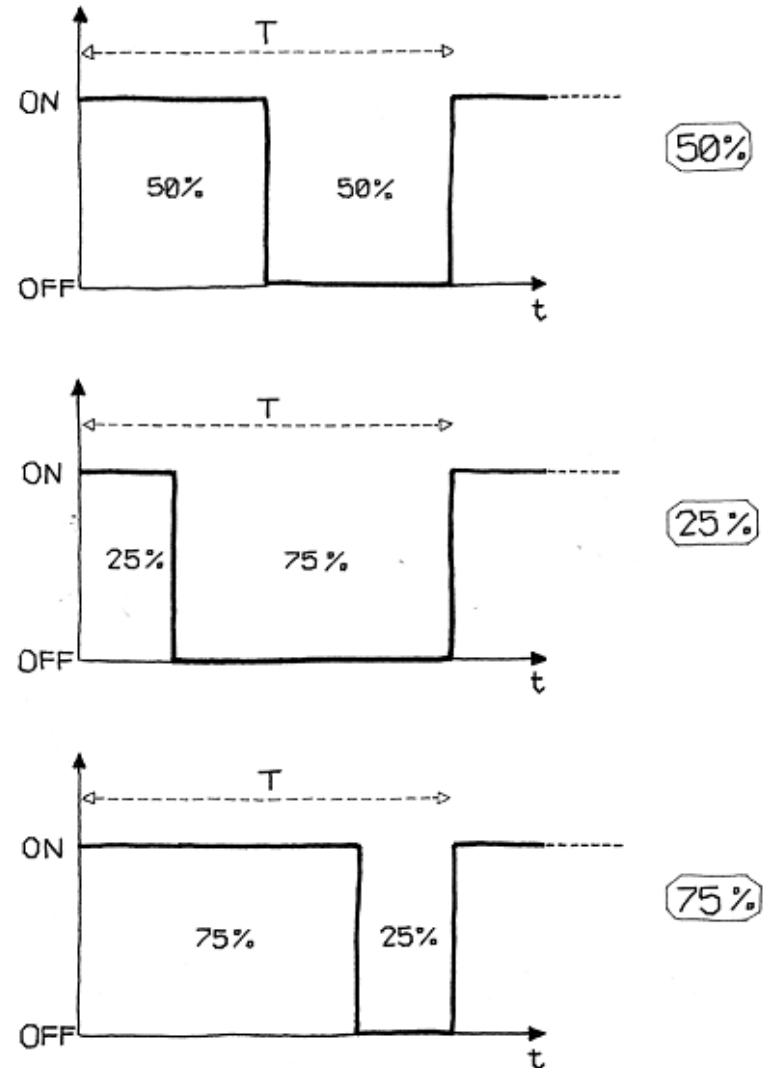
Com o *Sketch07* aberto: Save As... → *Sketch08*

```
//Sketch08
```

```
//Cada toque em PB1 desliga led atual e liga próximo.
```

PWM

- Pulse Width Modulation (modulação por largura de pulso)



"The frequency of the PWM signal on most pins is approximately 490 Hz. On the Uno and similar boards, pins 5 and 6 have a frequency of approximately 980 Hz."

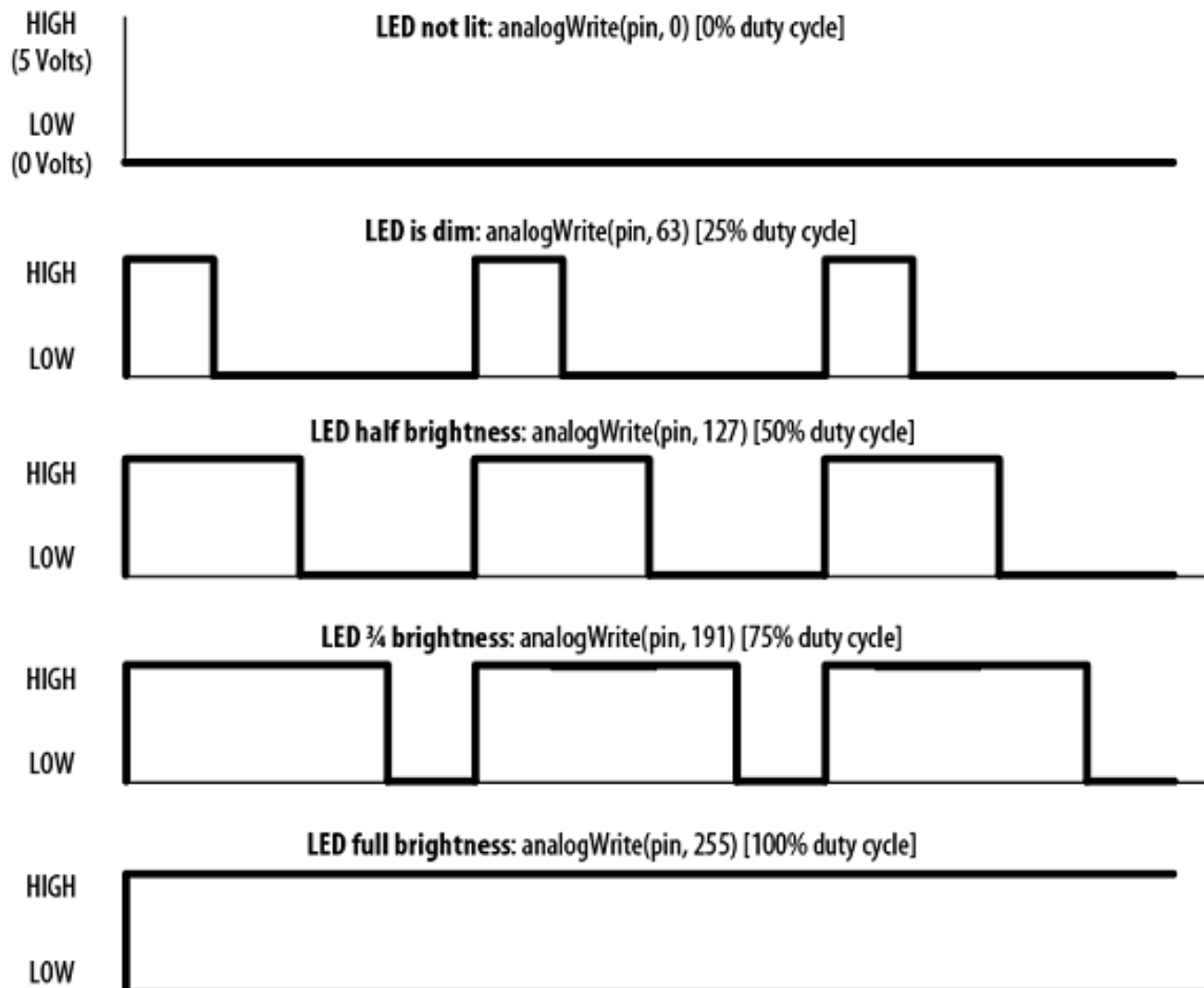
<http://arduino.cc/en/Reference/analogWrite>

Getting Started with Arduino, Massimo Banzi, O'Reilly, 2nd ed, 2011, Fig. 5-

3<http://www.amazon.com/Getting-Started-Arduino-Massimo-Banzi/dp/1449309879>

PWM

- Pulse Width Modulation (modulação por largura de pulso)



Sketch09

```
//Sketch09
//Controla a intensidade do LED2 por PWM
//Massimo Banzi, Getting Started with Arduino
//O'Reilly, 2nd ed, 2011, p.57.

const int LED2 = 9; // the pin for the LED
int i = 0; // We'll use this to count up and down

void setup() {
  pinMode(LED2, OUTPUT); // tell Arduino LED is an output
}

void loop(){
  for (i = 0; i < 255; i++) { // loop from 0 to 254 (fade in)
    analogWrite(LED2, i); // set the led brightness
    delay(10); // Wait 10ms because analogWrite
                // is instantaneous and we would
                // not see any change
  }
  for (i = 255; i > 0; i--) { // loop from 255 to 1 (fade out)
    analogWrite(LED2, i); // set the LED brightness
    delay(10); // Wait 10ms
  }
}
```


Com o *Sketch09* aberto: Save As... → *Sketch10*

```
//Sketch10
```

```
//PWM crescente no LED2 e decrescente no LED3
```

Sketch11

```
//Sketch11
//PWM no LED2 usando seno
//Michael Roberts, Beginning Arduino,
//Listing 3-3, p.59.

const int LED2 = 9;
float sinVal;
int ledVal;

void setup() {
  pinMode(LED2, OUTPUT);
}

void loop(){
  //from 0 to 179 deg in order to get positive values only
  for (int x=0; x<180; x++) {
    //convert deg to rad then obtain sin value
    sinVal = sin(x*(3.1412/180));
    //convert to int in order to use in analogWrite
    ledVal = int(sinVal*255);
    analogWrite(LED2, ledVal);
    delay(25);
  }
}
```

Com o *Sketch11* aberto: Save As... → *Sketch12*

```
//Sketch12
```

```
//PWM no LED2 usando seno, depois PWM linear no LED3
```

ADC • Analog-to-digital converter



- ADC de 6 canais (6 entradas analógicas)
- Resolução de 10 bits
- Faixa de entrada: 0 a 5V
- $5/1024 \approx 4,9 \text{ mV}$

```
analogRead()
```

```
0V → 0000000000 = 0
```

```
5V → 1111111111 = 1023
```

Sketch13

```
//Sketch13
//Pisca LED1 em uma taxa especificada pelo
//potenciômetro na entrada analógica 0
//Massimo Banzi, Getting Started with Arduino
//O'Reilly, 2nd ed, 2011, p.64.

const int LED = 8; // the pin for the LED
int val = 0; // variable used to store the value
               // coming from the pot

void setup() {
    pinMode(LED, OUTPUT); // LED is as an OUTPUT
                          // Note: Analogue pins are
                          // automatically set as inputs
}

void loop() {
    val = analogRead(0); // read the value from the pot

    digitalWrite(LED, HIGH); // turn the LED on
    delay(val); // stop the program for some time
    digitalWrite(LED, LOW); // turn the LED off
    delay(val); // stop the program for some time
}
```


Com o *Sketch13* aberto: Save As... → *Sketch14*

```
//Sketch14  
//O brilho do LED2 é controlado pelo  
//potenciômetro na entrada analógica 0.  
//Usa a função map().
```

Use a função *map*:

```
map(value, fromLow, fromHigh,  
toLow, toHigh)
```

Description

Re-maps a number from one range to another. That is, a **value** of **fromLow** would get mapped to **toLow**, a value of **fromHigh** to **toHigh**, values in-between to values in-between, etc.

<http://arduino.cc/en/Reference/map>

Sketch15

```
//Sketch15
//O brilho do LED2 é controlado pelo
//potenciômetro na entrada analógica 0.
//Usa a função map().
//Envia valor do brilho do led pela serial a cada 200 ms

const int LED2 = 9;
const int PB1 = 12;
int val = 0;
int newRange = 0;

void setup() {
  pinMode(LED2, OUTPUT);
  Serial.begin(9600); // 9600 bits/seg
}

void loop() {
  val = analogRead(0); // read the value from the pot
  newRange = map(val, 0, 1023, 0, 255);
  analogWrite(LED2, newRange); // brilho do led

  Serial.print("brilho = "); //envia string
  Serial.println(newRange); //envia valor e new line

  delay(200);
}
```

Com o *Sketch15* aberto: Save As... → *Sketch16*

```
//Sketch16
//O brilho do LED2 é controlado pelo
//potenciômetro na entrada analógica 0.
//Usa a função map().
//Envia valor do brilho do led pela serial a cada toque em PB1.
```

Sketch17

```
//Sketch17
//Recebe da serial:
//a,s,d,f -> toggle leds 1,2,3,4

const int LEDS[] = {8,9,10,11};
int byteRead;
int toggle0, toggle1 = 0;
int toggle2, toggle3 = 0;

void toggleLed(int ledIdx, int value){
    if (value == HIGH){
        digitalWrite(LEDS[ledIdx], HIGH);
    }
    else{
        digitalWrite(LEDS[ledIdx], LOW);
    }
}

void setup() {
    for(int i=0; i<4;i++){
        pinMode(LEDS[i], OUTPUT);
    }
    Serial.begin(9600); // 9600 bits/seg
}
```



```
void loop() {
    //Verifica se tem carac no buffer
    if(Serial.available() > 0){

        //retira um carac do buffer
        byteRead = Serial.read();

        switch (byteRead){
            case 'a':
                toggle0 = 1 - toggle0;
                toggleLed(0, toggle0);
                break;
            case 's':
                toggle1 = 1 - toggle1;
                toggleLed(1, toggle1);
                break;
            case 'd':
                toggle2 = 1 - toggle2;
                toggleLed(2, toggle2);
                break;
            case 'f':
                toggle3 = 1 - toggle3;
                toggleLed(3, toggle3);
                break;
        }
    }
}
```

`Serial.available()`: Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer. <http://arduino.cc/en/Serial/Available>



Com o *Sketch17* aberto: Save As... → *Sketch18*

```
//Sketch18  
//Recebe da serial: a,s,d,f -> toggle leds 1,2,3,4.  
//Envia pela serial o estado do led que acabou de ser "tooglado".
```