



Introdução ao Arduino

Pessanha Santos
ASPOF EN-AEL

Programa

1. **Introdução** à plataforma de desenvolvimento *Arduino*
2. Análise à sua **constituição**
3. **Software** de desenvolvimento Arduino
4. **Estrutura** e **Sintaxe** do seu código
5. Estudo de alguns **exemplos**



Introdução

Plataforma de desenvolvimento Arduino

O que é o *ARDUINO*?

- Plataforma de desenvolvimento
 - Simples placa de circuito impresso com um **microcontrolador ATmega** da ATMEL (mas vamos já ver exemplos!?!??).
- Características principais
 - **Simplicidade** de utilização (Programação, utilização...);
 - *Cross-platform*;
 - Baixo **custo**;
 - *Open-Source*.
- A possibilidade de **actuar** no Mundo que nos rodeia.

Arduino o que consigo fazer?

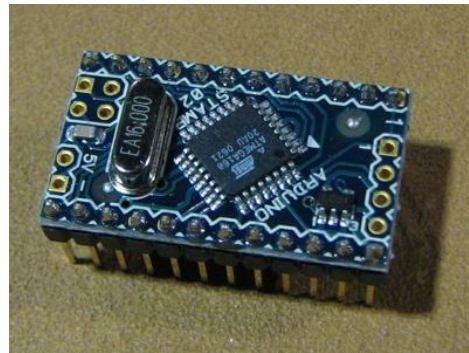
- Ler valores provenientes de **sensores**
 - Acelerómetros, LDR, ultra-sons, entre muitos outros.
- **Actuar** no Mundo exterior
 - Leds, Motores, Displays(LCD), entre muitos outros.
- Capacidade de efectuar **protótipos** rapidamente e com grande **simplicidade**.
- E muito mais.....

O que é o *Arduino*? Exemplos ?

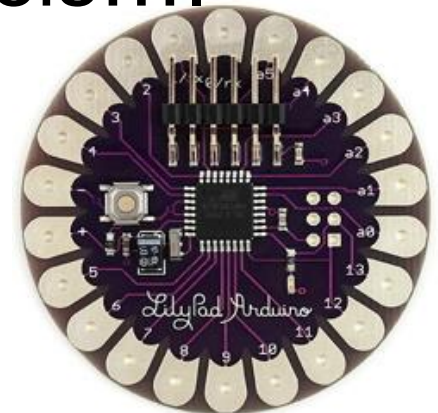
- Algumas apresentações possíveis....



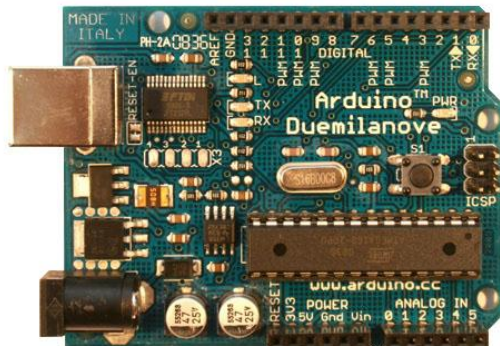
Nano



Mini



LilyPad

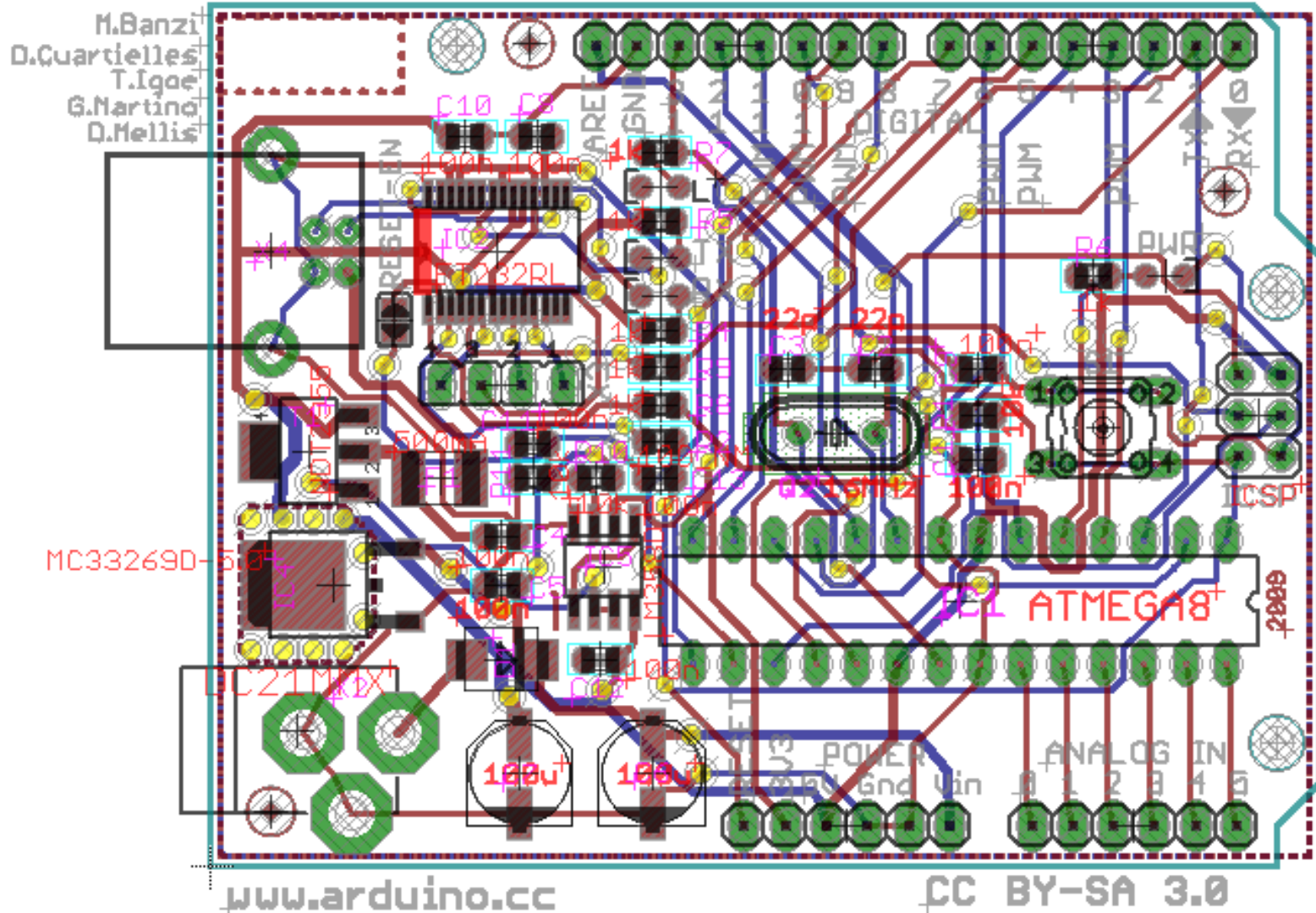


Duemilanove

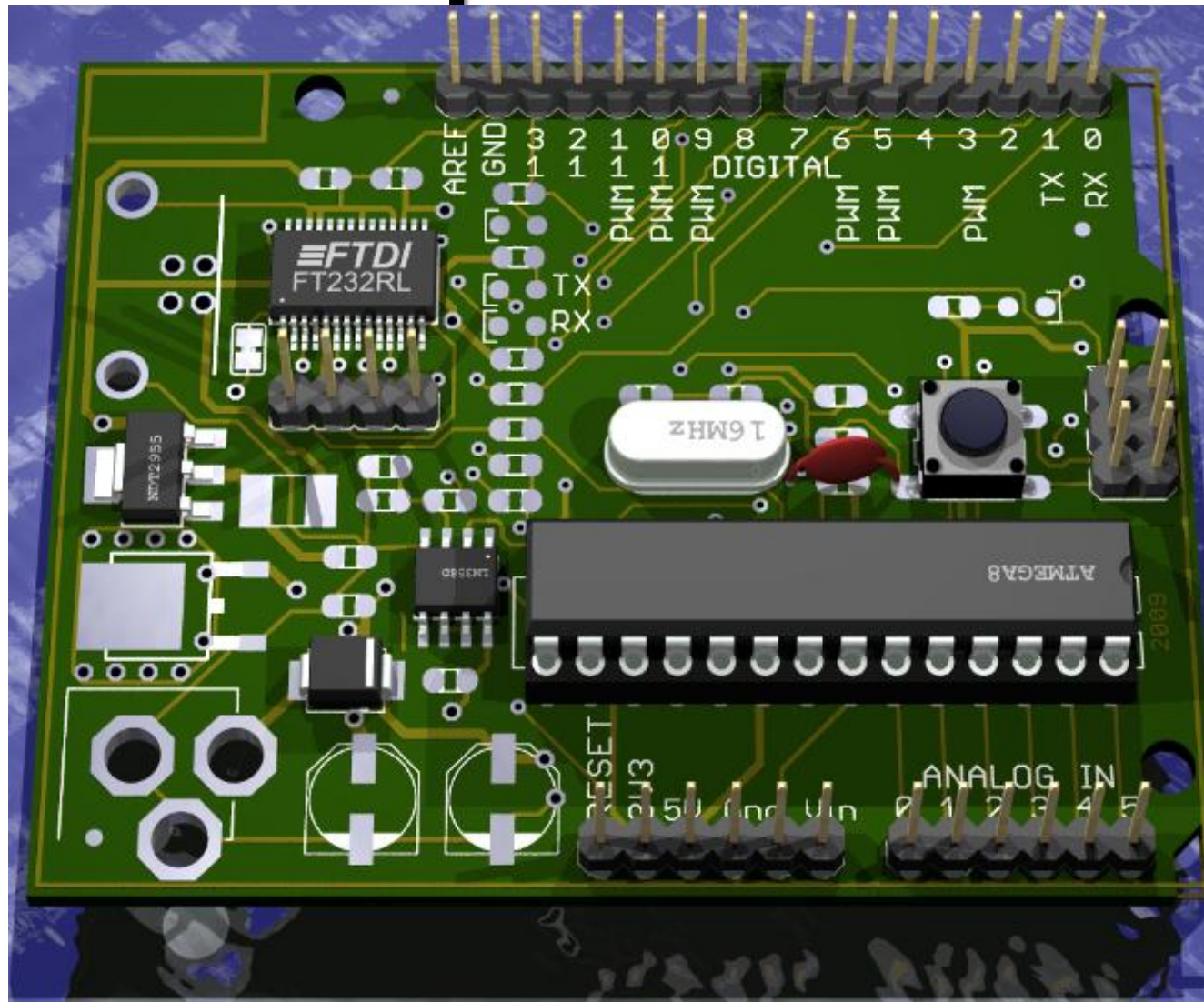


Mega

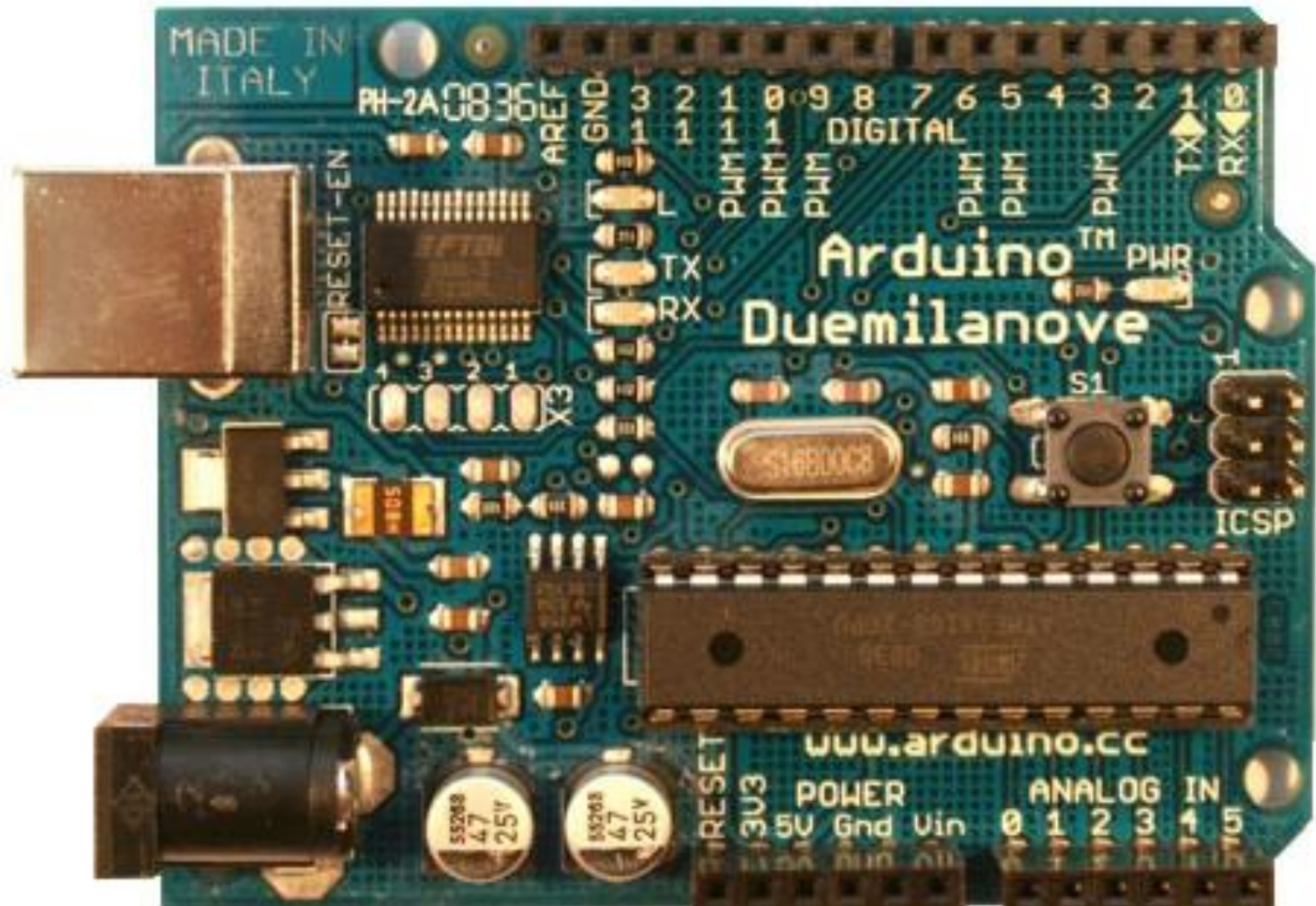
Arduino Duemilino



Ou mais simplesmente...



Ou “ainda” mais simplesmente...



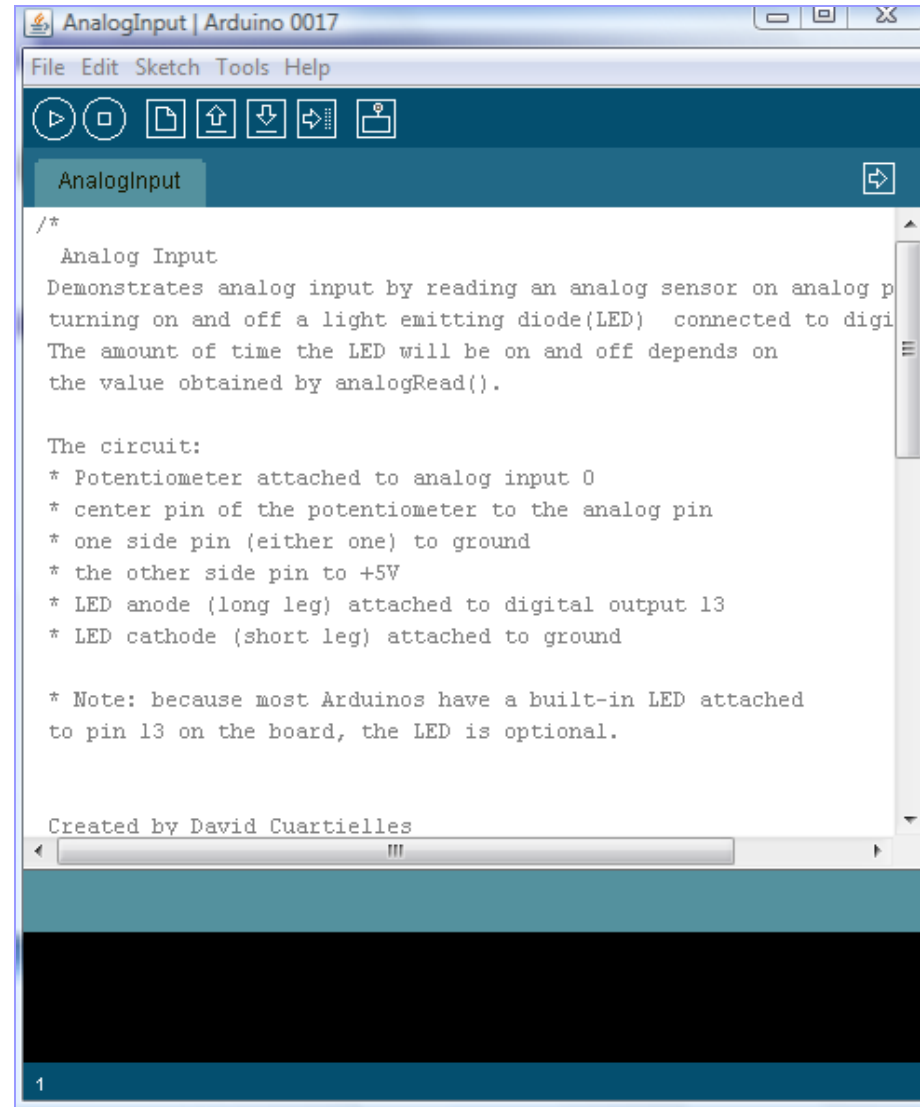
O que é o *Arduino*?

■ Software.....

Arduino *alpha*

An open project written, debugged and supported by Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino and David Mellis

Based on processing by Casey Reas and Ben Fry



```
File Edit Sketch Tools Help
AnalogInput
/*
  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
  pin 13. 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.

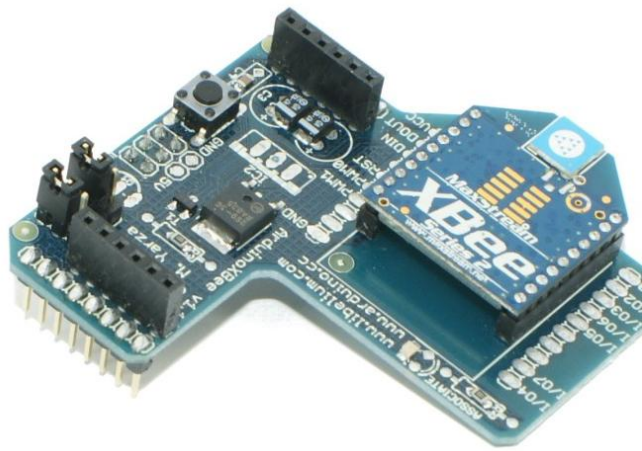
  Created by David Cuartielles
*/
```

Vantagens...

- Claramente ser uma ferramenta *Open-Source* (Software e Hardware);
- Tem uma *enorme comunidade* de seguidores por todo o Mundo (Permite uma *constante actualização e inovação*);
- Pode operar sem a presença de um computador (*standalone*);
- Possibilidade de expandir a sua capacidade através da utilização de *shields*.

O que são *shields*? Exemplos ?

- Algumas apresentações possíveis....



Zigbee



Inputshield



Ethernet

Mas existem muitas mais....

Bibliografia (Alguns exemplos...)

■ Livros de texto

- **Making Things Talk**, Tom Igoe, O'REILLY, 2007.
- **Getting started with Arduino**, Massimo Banzi, O'REILLY, 2007.
- **Programming Interactivity**, Joshua Noble, O'REILLY, 2009.

■ Outros...

- **Site oficial** (www.arduino.cc) ;
- Alguns Fóruns (Lusorobótica, Portugal-a-Programar).



Hardware

Análise à sua constituição

Microcontroladores utilizados

Modelo	Microcontrolador utilizado
<i>Arduino Duemilínove</i>	ATmega168 ou ATmega328
<i>Arduino Diecimília</i>	ATmega168
<i>Arduino Mega</i>	ATmega1280
<i>Arduino Nano</i>	ATmega168 ou ATmega328
<i>LilyPad</i>	ATmega168V
<i>Pro</i>	ATmega168 ou ATmega328
<i>Pro mini</i>	ATmega168

- Basicamente baseia-se em três modelos de microcontrolador: ATmega168, ATmega328 e ATmega1280

Microcontroladores utilizados (Exemplos de apresentação)



ATmega168
(PDIP)






ATmega1280
(TQFP)



ATmega328
(PDIP)

ATmega168/328/1280

Algumas características

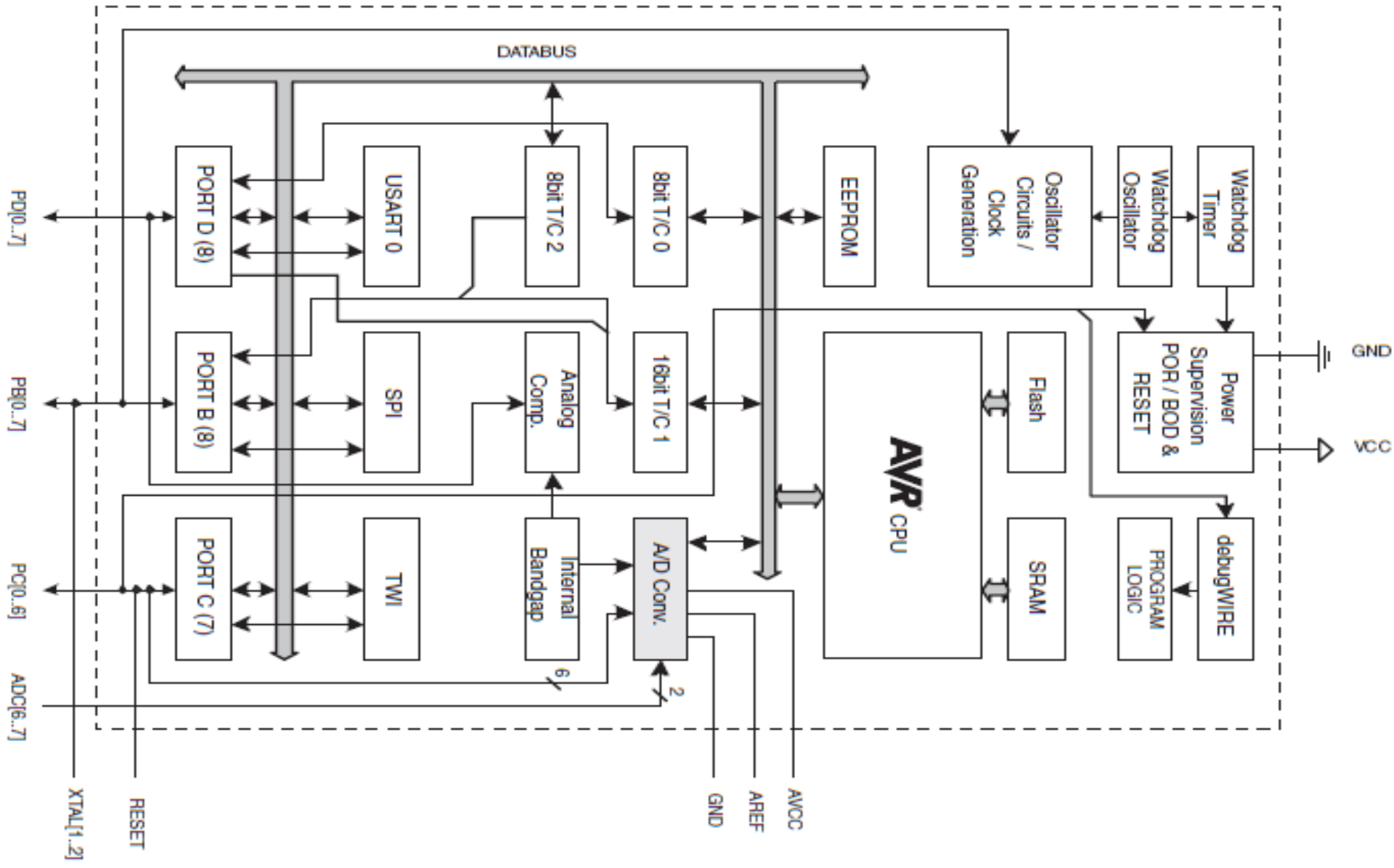
ATmega168 		ATmega328 		ATmega1280 	
Flash	16 KB	Flash	32 KB	Flash	128 KB
SRAM	1 KB	SRAM	2 KB	SRAM	8 KB
EEPROM	512 bytes	EEPROM	1 KB	EEPROM	4 KB
Clock máximo	20 MHz	Clock máximo	20 MHz	Clock máximo	16 MHz
ADC	10 bit	ADC	10 bit	ADC	10 bit
Consumo a 25°C (Modo activo)	250 µA 1 MHz (1.8 V)	Consumo a 25°C (Modo activo)	0.2 mA 1 MHz (1.8 V)	Consumo a 25°C (Modo activo)	500 µA 1 MHz (1.8 V)
Outros	PWM	Outros	PWM	Outros	PWM
	I ² C		I ² C		I ² C
	SPI		SPI		SPI
	RS232		RS232		RS232

ATmega168/328/1280

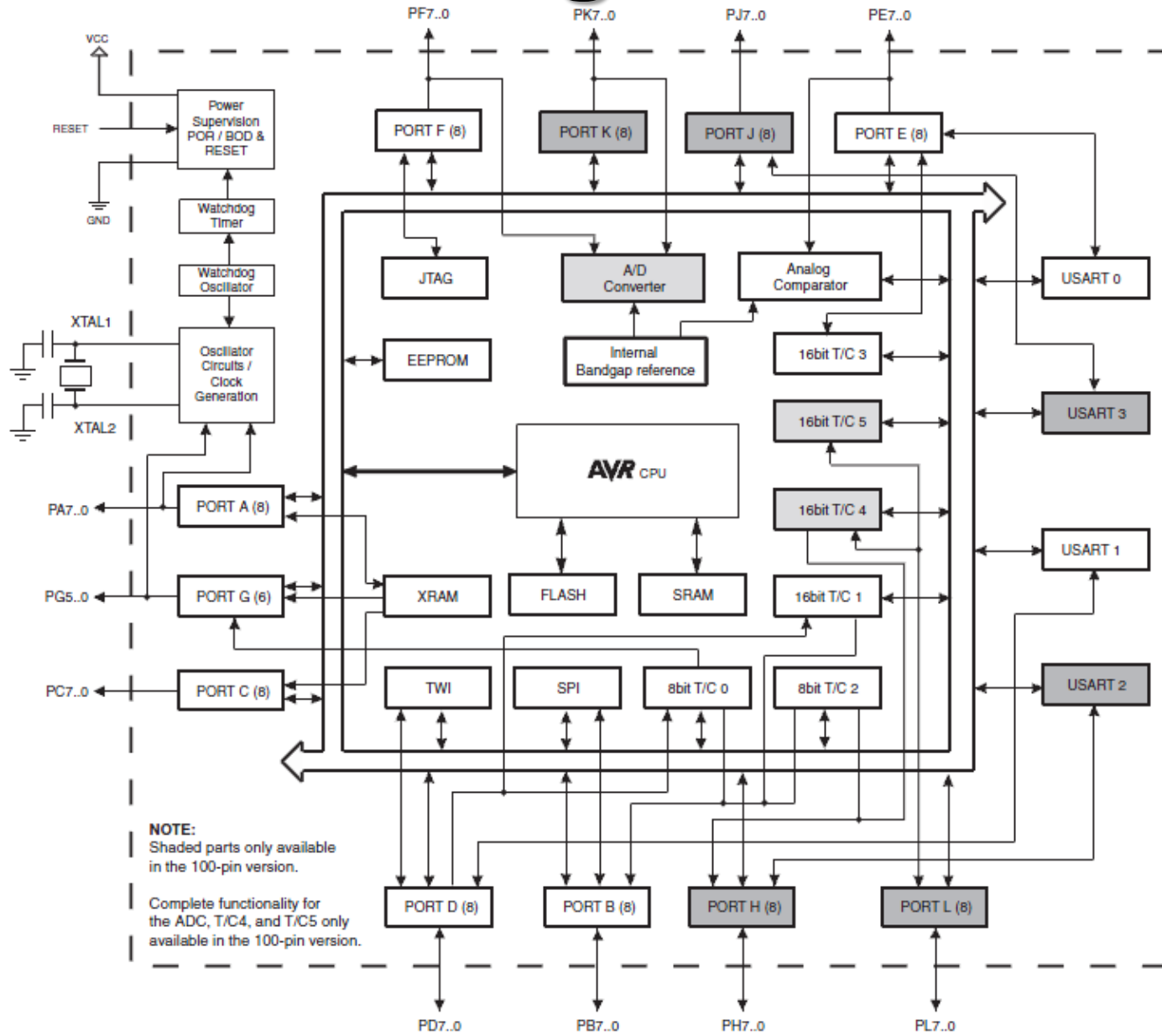
Algumas características (Cont.)

- Existe claramente uma **diferença** em termos de **memória disponível** (SRAM, Flash e EEPROM);
- O ATmega328 apresenta a **mesma arquitectura** do ATmega168 mas com **diferentes capacidades** em termos de **quantidade de memória** disponível;
- **Consumo energético** do ATmega1280 é **inferior** ao do ATmega328 nas **mesmas condições** de funcionamento;
- **Todos os modelos** apresentados possuem a **mesma resolução** no seu conversor A/D.

ATmega168/328



ATmega1280



Arduino Duemilinue vs Mega

<i>Arduino Duemilinue</i> 		<i>Arduino Mega</i> 	
Microcontrolador	Atmega168/328	Microcontrolador	ATmega1280
Tensão de operação	5V	Tensão de operação	5V
Tensão de entrada (limites)	6-20V	Tensão de entrada (limites)	6-20V
Pinos de I/O digital	14	Pinos de I/O digital	54
Pinos analógico	6	Pinos analógico	16
Pinos PWM	6	Pinos PWM	14
Corrente DC por pino de I/O	40 mA	Corrente DC por pino de I/O	40 mA
Corrente DC (3.3V)	50 mA	Corrente DC (3.3V)	50 mA

Arduino Duemilinueve vs Mega

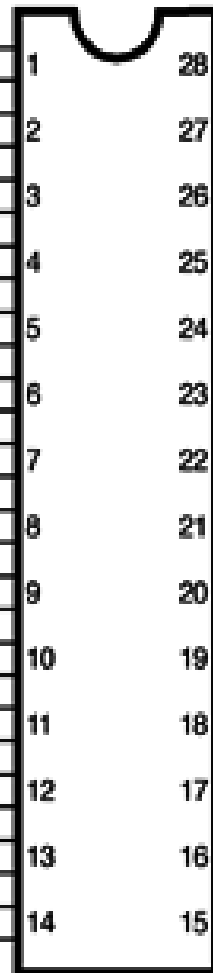
- Existe uma diferença clara em termos de número de **pinos analógicos e digitais** disponíveis, com **vantagem** para o *Arduino Mega*;
- O Arduino Mega apresenta **maiores dimensões** que o Duemilinueve, o que dependendo da **aplicação** pode um factor importante;
- **Funcionam** ambos com as **mesmas tensões** de **alimentação**;
- Mas não podemos esquecer que o **ATmega1280** (Arduino Mega) apresenta uma **maior quantidade de memória** disponível (EEPROM, SRAM e Flash).

Atmega168/328

Análise ao seu pinout vs *Arduino*

Arduino function

reset	(PCINT14/RESET) PC6	1
digital pin 0 (RX)	(PCINT16/RXD) PD0	2
digital pin 1 (TX)	(PCINT17/TXD) PD1	3
digital pin 2	(PCINT18/INT0) PD2	4
digital pin 3 (PWM)	(PCINT19/OC2B/INT1) PD3	5
digital pin 4	(PCINT20/XCK/T0) PD4	6
VCC	VCC	7
GND	GND	8
crystal	(PCINT6/XTAL1/TOSC1) PB6	9
crystal	(PCINT7/XTAL2/TOSC2) PB7	10
digital pin 5 (PWM)	(PCINT21/OC0B/T1) PD5	11
digital pin 6 (PWM)	(PCINT22/OC0A/AIN0) PD6	12
digital pin 7	(PCINT23/AIN1) PD7	13
digital pin 8	(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)

Arduino function

analog input 5
analog input 4
analog input 3
analog input 2
analog input 1
analog input 0
GND
analog reference
VCC
digital pin 13
digital pin 12
digital pin 11(PWM)
digital pin 10 (PWM)
digital pin 9 (PWM)

Arduino Duemilino

Referência Analógica

Pinos digitais

Power led

Cristal

Botão de reset

FTDI

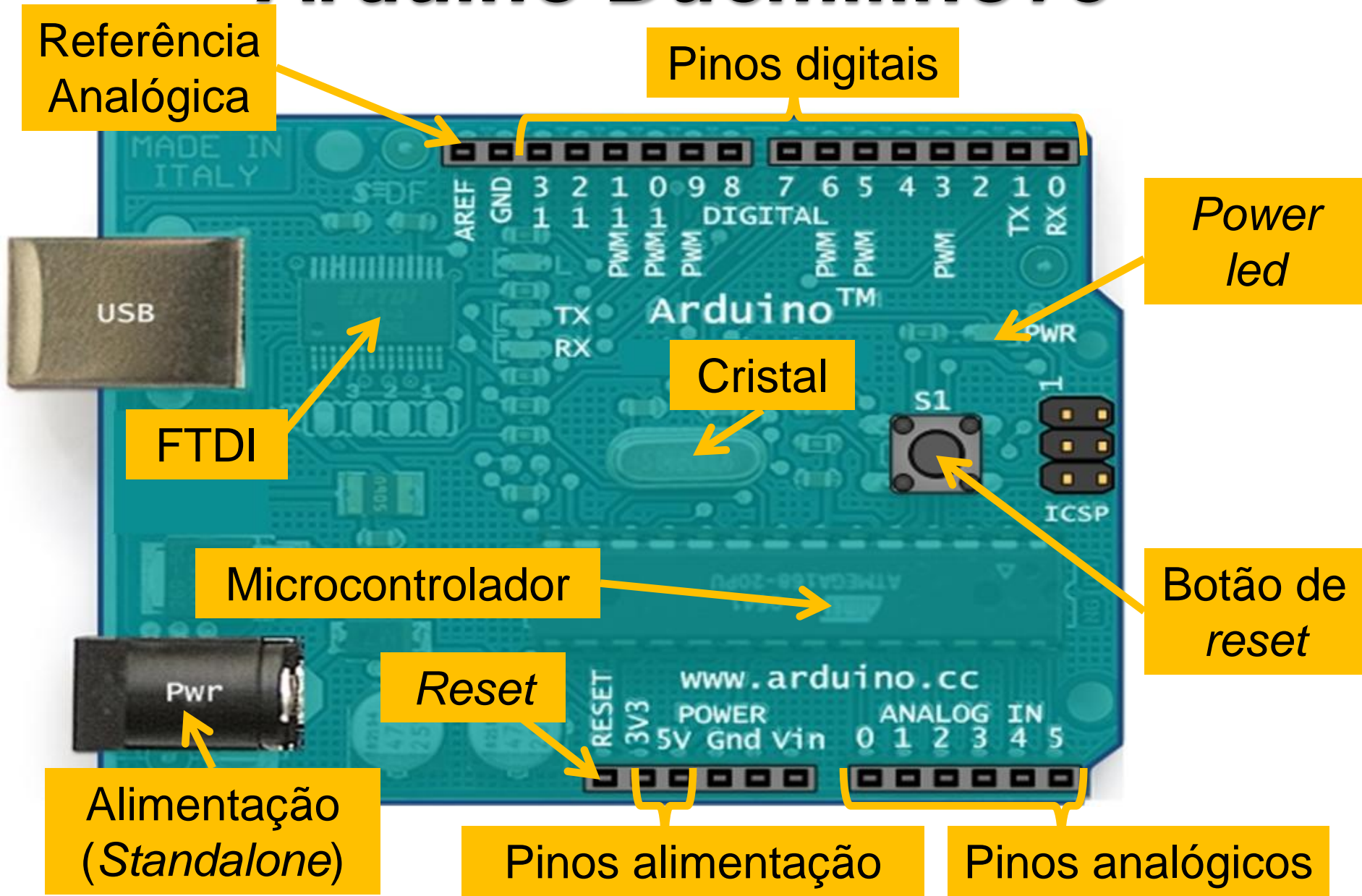
Microcontrolador

Reset

Alimentação (Standalone)

Pinos alimentação

Pinos analógicos



Arduino Mega

Referencia Analógica

Pinos digitais (PWM)

Pinos de comunicação

FTDI

ICSP

Pinos digitais

Microcontrolador

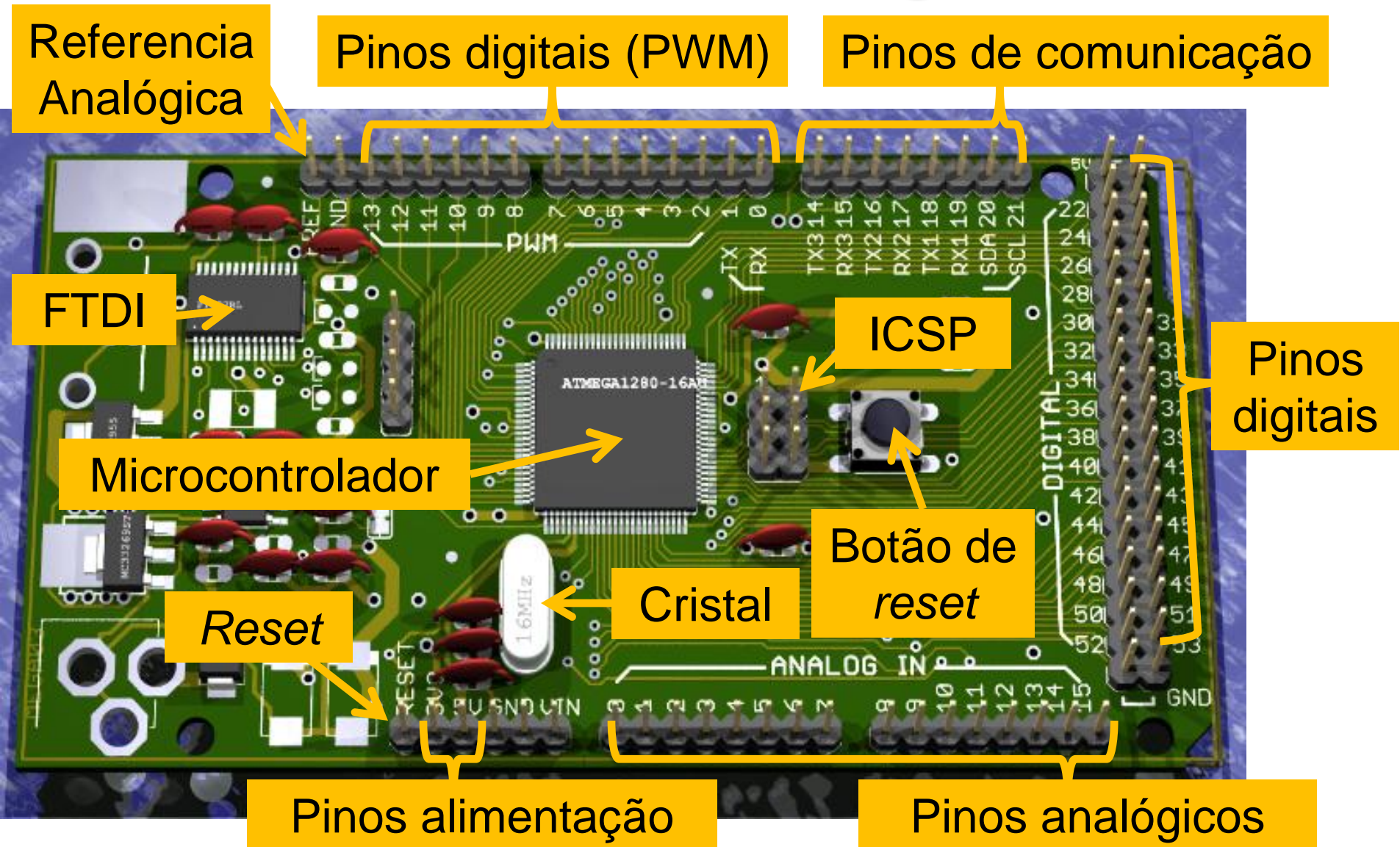
Botão de reset

Reset

Cristal

Pinos alimentação

Pinos analógicos





Software

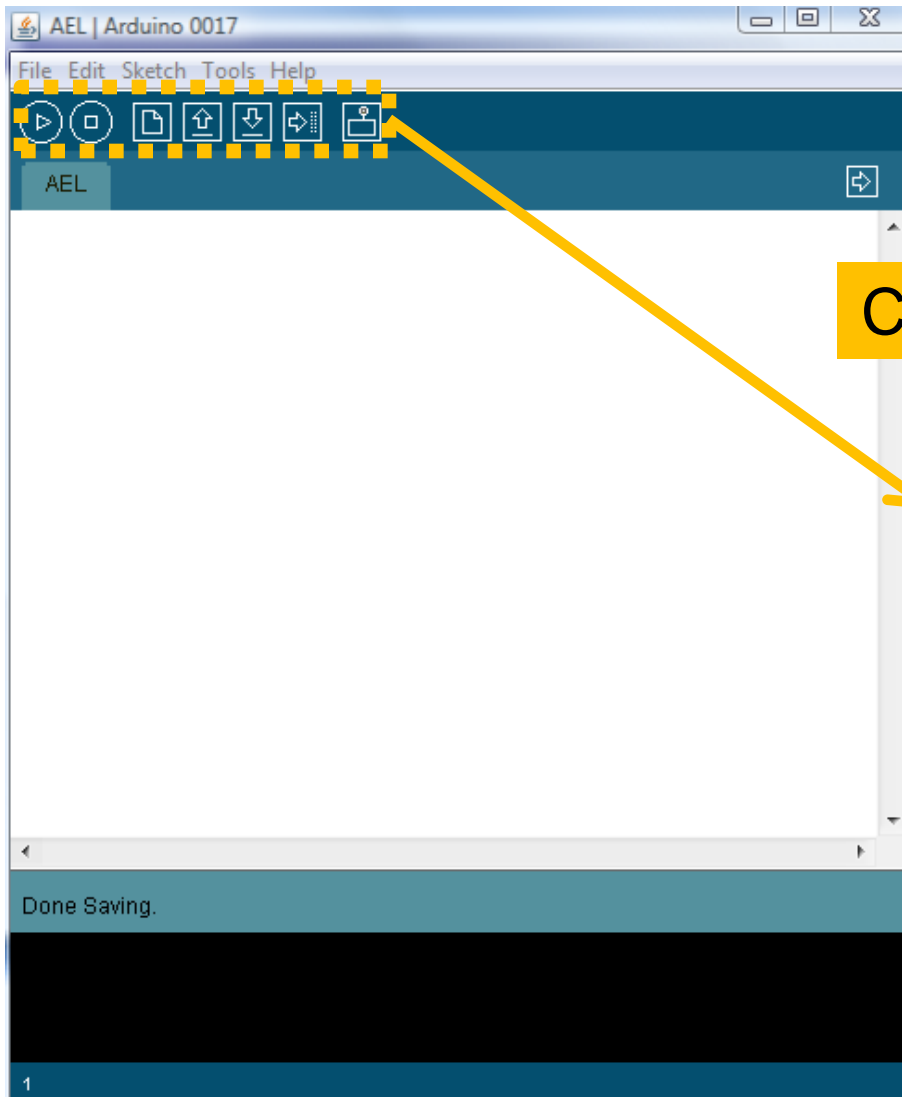
Uma abordagem à sua utilização

Software de desenvolvimento

Algumas características

- Disponível para **download** directamente do **site oficial** Arduino (www.arduino.cc);
- *Open-Source*;
- *Cross-platform*;
- Ambiente de desenvolvimento escrito em **java**;
- Sintaxe utilizada baseada na **linguagem de programação** de alto nível **C** (**Basicamente é C.....**);
- Enorme **simplicidade** de **utilização**, devido ao **bootloader** previamente gravado no microcontrolador.

Software de desenvolvimento



Compilar

Novo

Upload



Abrir

Parar
Compilar

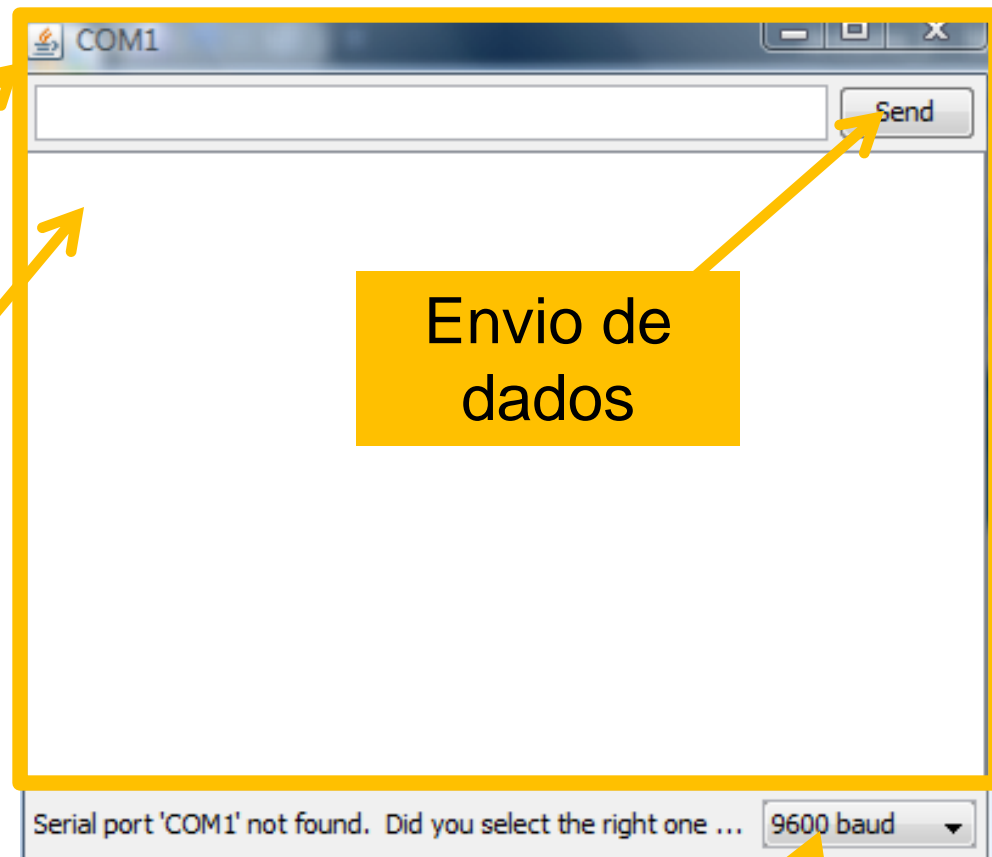
Leitura
porta série

Leitura da porta série (Software Arduino)

- Possibilita também a **leitura e envio** de dados utilizando a **porta série**



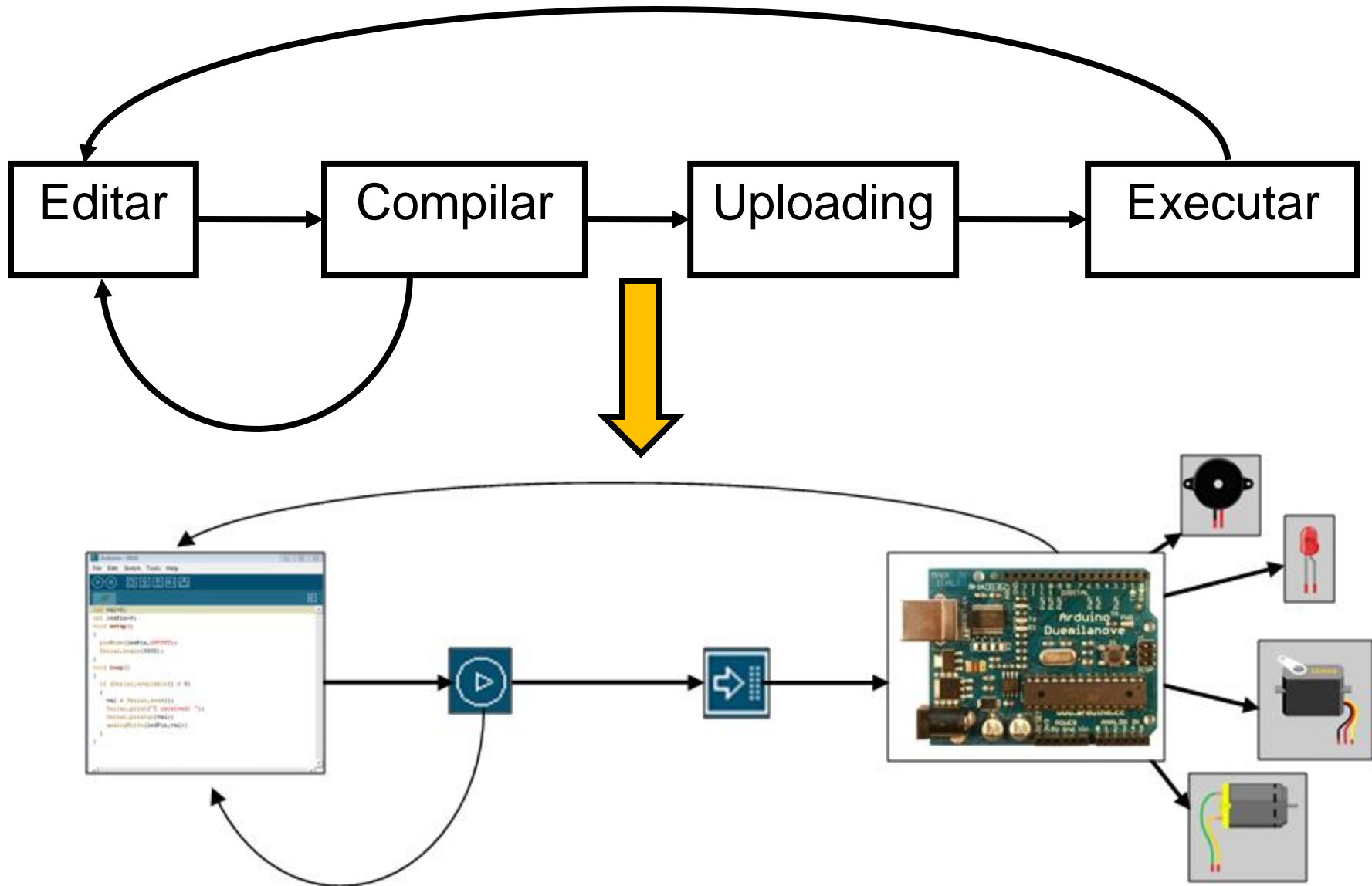
Visualizar
dados
recebidos



Envio de
dados

Seleccionar a Baud rate

Ciclo de desenvolvimento





Estrutura e Sintaxe

Instruções e estrutura do código a utilizar

Estrutura do *sketch*

//Declaração de bibliotecas

```
#include <Client.h>
#include <Ethernet.h>
#include <Server.h>
```

Declaração de bibliotecas

//Declaração de variáveis globais

```
int i=0;
float x=5.67;
```

Declaração de variáveis globais

```
void setup() {
//Instrução 1
//Instrução 2
}
```

Função setup

```
void loop() {
//Instrução 3
//Instrução 4
}
```

Função loop

Obrigatoriamente do
tipo - void

Funções Importantes

- A função **void setup()** é apenas executada **uma vez**, sendo utilizada para:
 - Inicialização de **variáveis**;
 - Inicialização de da utilização de **bibliotecas**;
 - Definição dos **pinos** a utilizar;
 - Início do uso da **comunicação série**.
- A função **void loop()** é uma função executada em **loop**. Apenas outras funções, cuja chamada é feita ao executar esta função, serão executadas.

Funções Analógicas e Digitais

//Definição do pino “Número do pino” como “INPUT” ou “OUTPUT”

pinMode(Número do Pino, Modo);

EX: pinMode(13, OUTPUT);

//Definição do pino “Número do pino” como “HIGH” ou “LOW”

digitalWrite(Número do Pino, Modo);

EX: digitalWrite(13, OUTPUT);

//Permite a leitura do valor digital presente no “Número do pino”

Variável do tipo integer = digitalRead(Número do Pino);

EX: leitura = digitalRead(5);

// Permite a leitura do valor analógico presente no “Número do pino”

Variável do tipo integer = analogRead(Número do Pino);

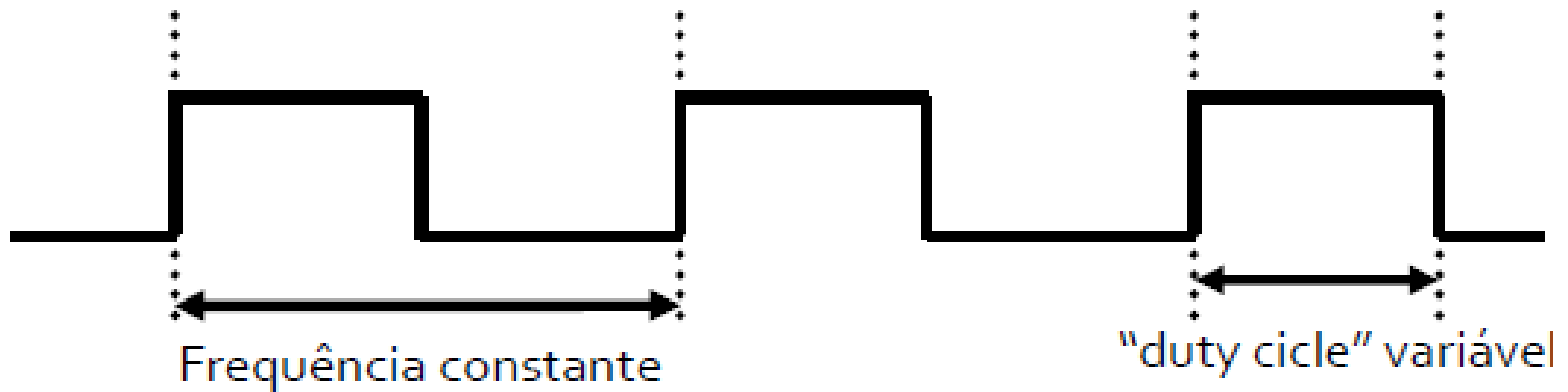
EX: leitura = analogRead(2);

//Permite a criação de um pulso PWM com o duty cycle definido pelo “valor”

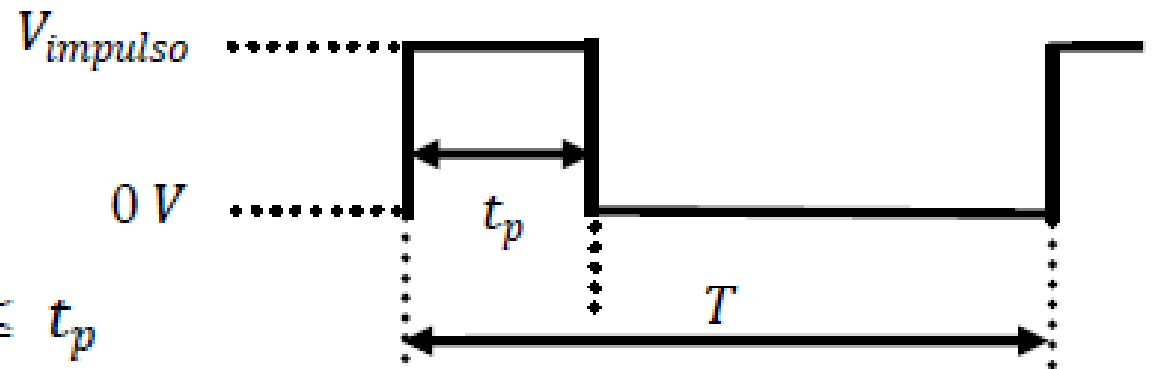
analogWrite(Número do Pino, valor);

EX: analogWrite(11,255);

PWM – Pulse Width Modulation

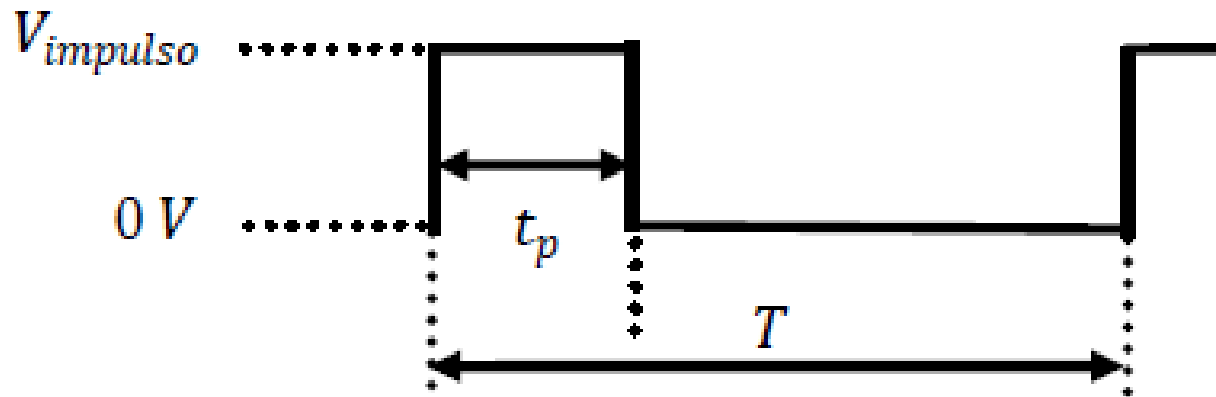


$$V_{dc} = \frac{1}{T} \int_0^T V(t) dt$$



$$V(t) = \begin{cases} V_{pulso} & \Rightarrow 0 \leq t \leq t_p \\ 0 & \Rightarrow t_p < t \leq T \end{cases}$$

PWM – Pulse Width Modulation



$$V_{dc} = \left(\int_0^{t_p} V_{impulso} dt + \int_{t_p}^T 0 dt \right) \Leftrightarrow V_{dc} = \frac{t_p}{T} * V_{impulso}$$

- Podemos então concluir que a **tensão média** V_{dc} é directamente **proporcional** ao **duty cycle** do sinal **PWM**.

Ciclo if....else....

//Ciclo que é utilizado para descrever uma condição

```
If(condição){  
Instrução 1;  
Instrução 2;  
}  
else{  
Instrução 3;  
Instrução 4;  
}
```

$X == Y$	X igual a Y
$X != Y$	X diferente de Y (não igual)
$X > Y$	X maior que Y
$X >= Y$	X maior ou igual a Y
$X < Y$	X menor que Y
$X <= Y$	X menor ou igual a Y

A condição referida anteriormente tem de respeitar as condições descritas na tabela acima. No caso descrito em cima se a **condição** se **verificar** o **instrução 1 e 2** é executada, **caso contrário** são executadas a **instrução 3 e 4**.

Ciclo for

//Ciclo que é utilizado quando se pretende executar um determinado conjunto de instruções um certo número de vezes

```
for( inicialização; condição; Incremento a efectuar){  
  Instrução 1;  
  Instrução 2;  
  (.....)  
}
```

$X == Y$	X igual a Y
$X != Y$	X diferente de Y (não igual)
$X > Y$	X maior que Y
$X >= Y$	X maior ou igual a Y
$X < Y$	X menor que Y
$X <= Y$	X menor ou igual a Y

A condição referida anteriormente tem de respeitar as condições descritas na tabela acima. A **inicialização** da variável apenas é **efectuada** no **inicio** do ciclo, sendo a **cada execução** do ciclo efectuado o respectivo **incremento** na variável de controlo do ciclo.

Ciclo switch / case

//Ciclo que é normalmente utilizado para descrever uma lista de casos possíveis para uma determinada variável

```
switch(variável){
```

```
case 1:
```

```
Instrução a executar quando a variável for 1 (variável == 1)
```

```
break;
```

```
case 2:
```

```
Instrução a executar quando a variável for 2 (variável == 2)
```

```
break;
```

```
(.....)
```

```
default:
```

```
Conjunto de instruções a executar se nenhuma das condições for verificada. A utilização desta condição é opcional.
```

```
break;
```

```
}
```

Ciclo while

//Ciclo que é utilizado quando se pretende executar um determinado conjunto de instruções um certo número de vezes

```
while(condição){  
Instrução 1;  
Instrução 2;  
}
```

Ciclo do.....while

//Ciclo bastante semelhante ao ciclo while mas a condição apenas é testada no fim do ciclo, sendo sempre executado o ciclo pelo menos uma vez

```
do{  
Instrução 1;  
Instrução 2;  
(.....)  
while(condição);
```



Exemplos

Aplicação do exposto anteriormente a casos práticos

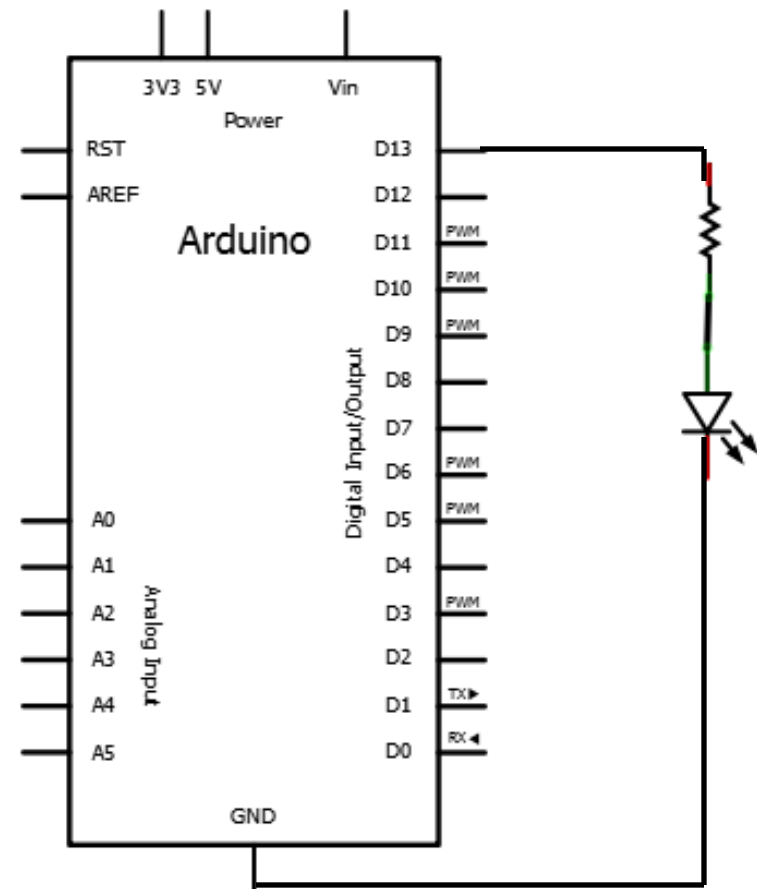
Exercício 1

- Faça com que um led acenda e apague com uma frequência de 2 Hz.

//Declaração de variáveis globais
int ledpin=13;

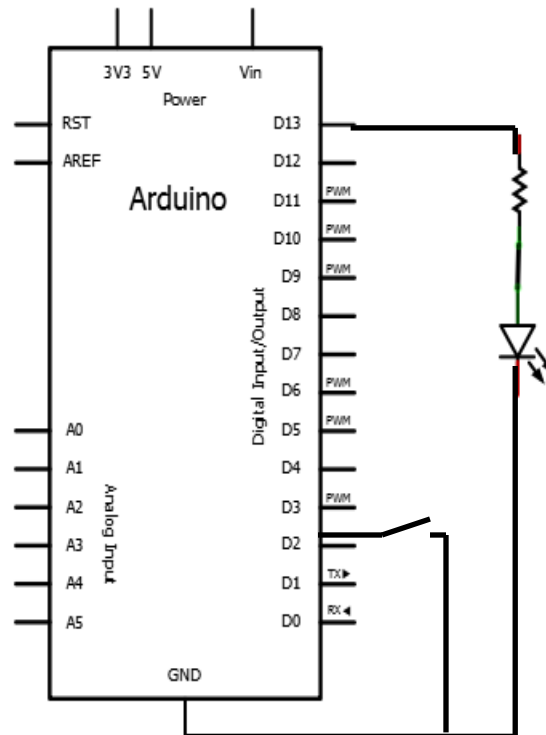
```
void setup() {  
  pinMode(ledPin,OUTPUT);  
}
```

```
void loop() {  
  digitalWrite(ledPin,HIGH);  
  Delay(500);  
  digitalWrite(ledPin,LOW);  
  Delay(500);  
}
```



Exercício 2

■ Utilize um sinal digital de entrada (HIGH ou LOW) para fazer com que o led acenda ou apague (valor digital de entrada HIGH o led liga, valor digital de entrada LOW o led encontra-se desligado).



Exercício 2 - Resolução

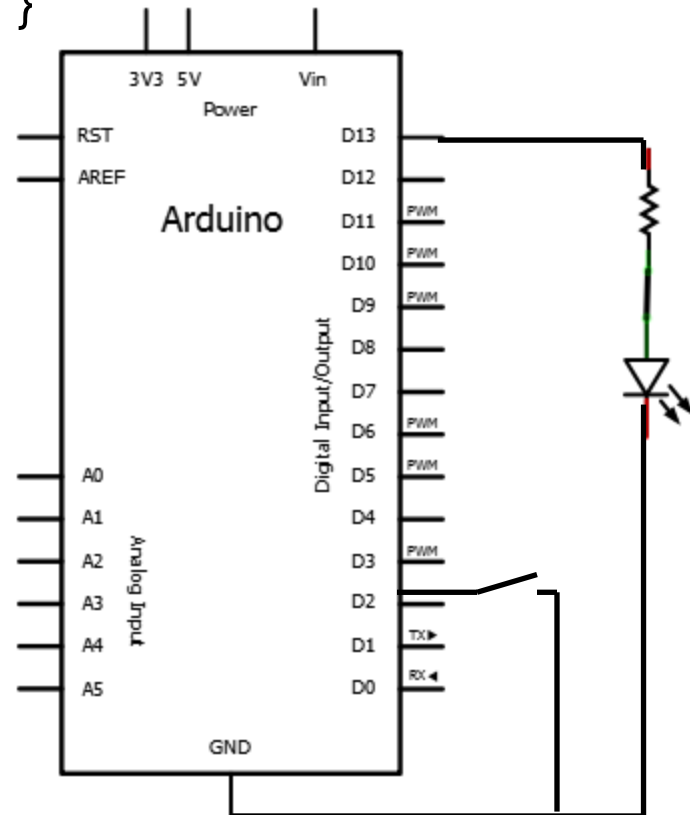
//Declaração de variáveis globais

```
int ledPin = 13;  
int comando=2;  
int val=0;
```

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

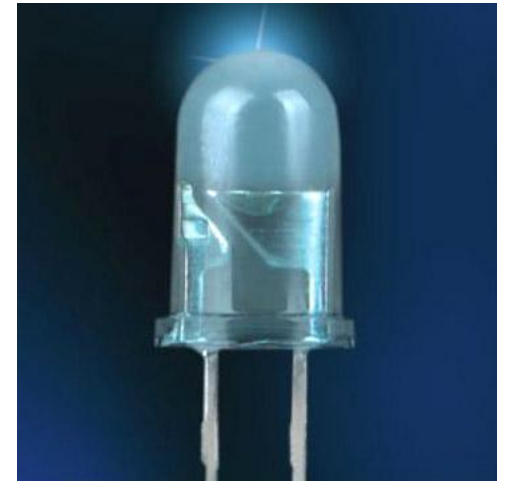
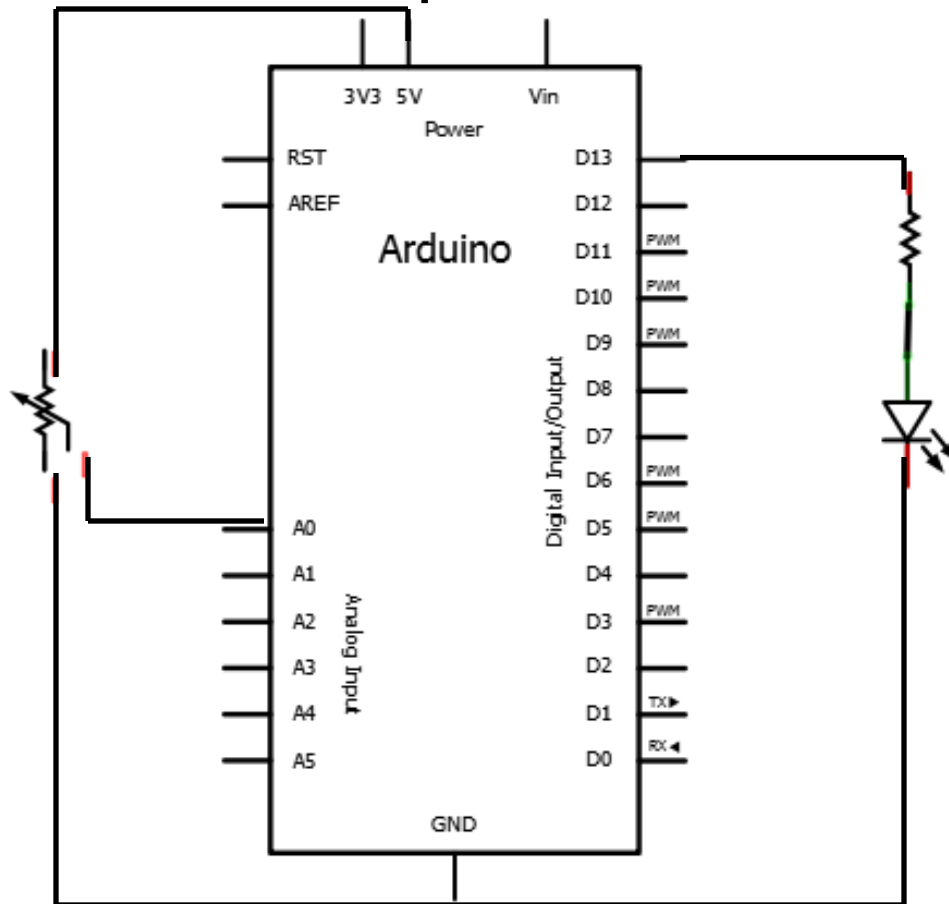
```
void loop() {  
  val=digitalRead(comando);  
  if (val==LOW)  
  {  
    digitalWrite(ledPin,LOW);  
  }  
}
```

```
else{  
  digitalWrite(ledPin, HIGH);  
}
```



Exercício 3

Recorrendo a leituras sucessivas ao valor de saída de um simples potenciômetro faça um regulador de luminosidade para o nosso tão *famoso* led.



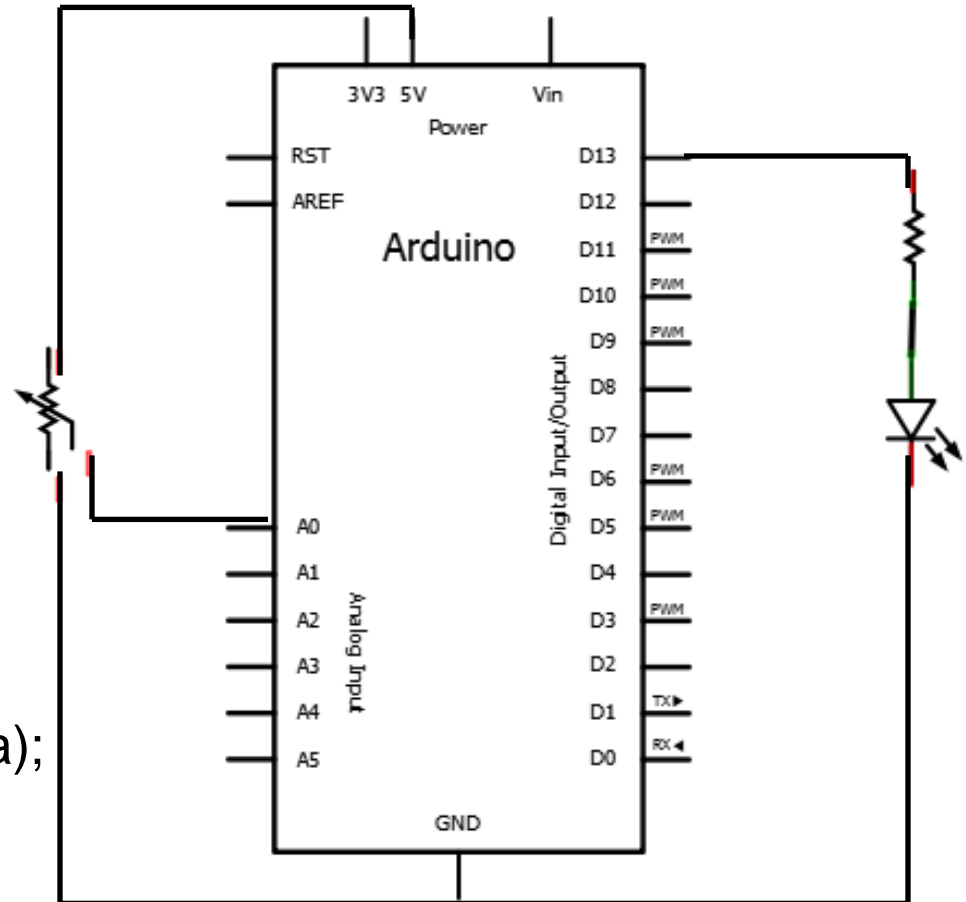
Exercício 3 - Resolução

//Declaração de variáveis globais

```
int ledPin = 13;  
int comando=2;  
int val=0;
```

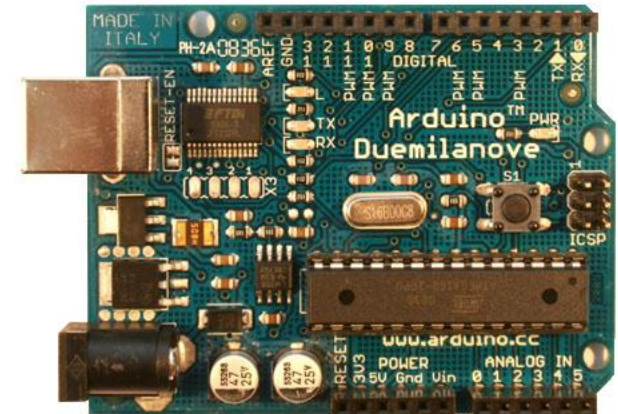
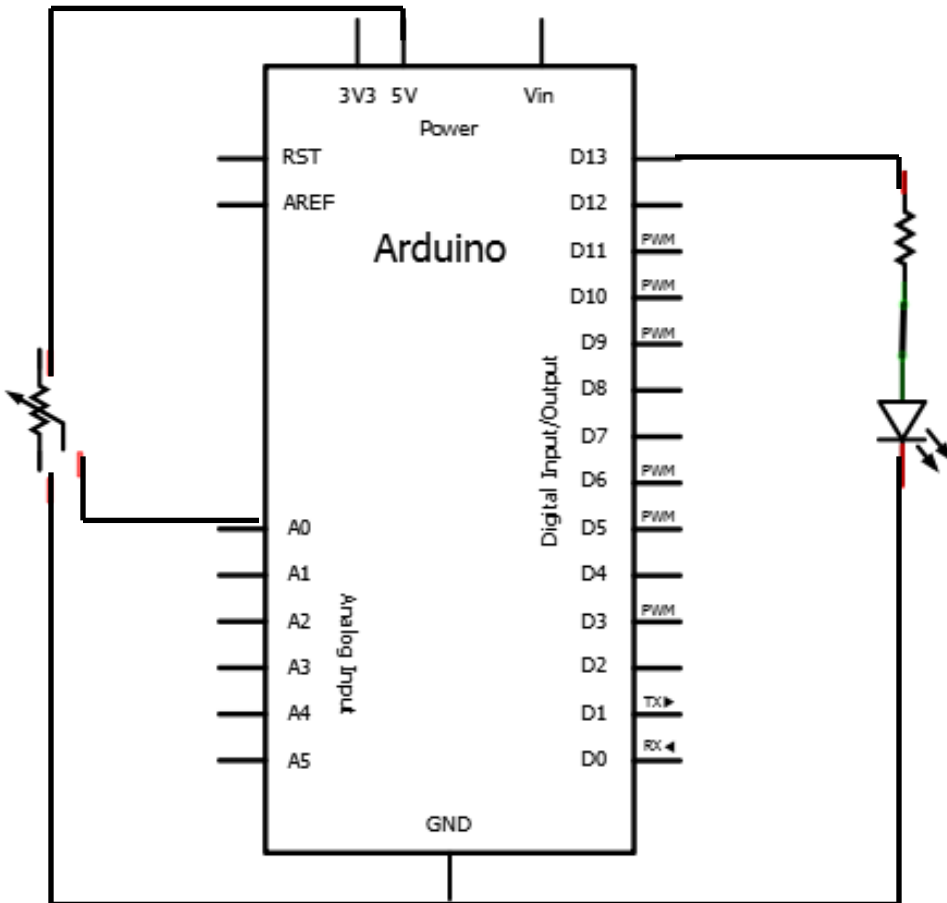
```
void setup() {  
  pinMode(entrada_analogica,INPUT)  
  pinMode(ledPin,OUTPUT);  
}
```

```
void loop() {  
  val=analogRead(entrada_analogica);  
  val=(val/4);  
  analogWrite(ledPin,val);  
}
```



Exercício 4

■ Obtenha o valor, em tempo real, da variável utilizada para controlar a luminosidade do led.



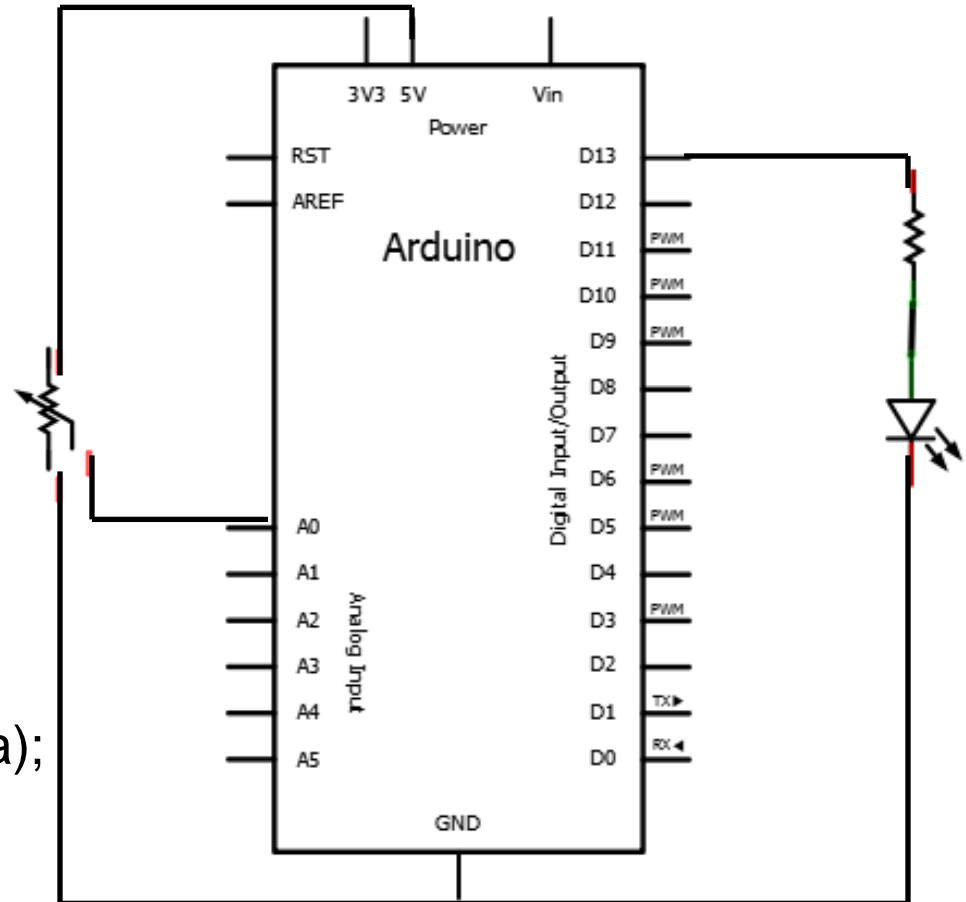
Exercício 4 - Resolução

//Declaração de variáveis globais

```
int ledPin = 13;  
int comando=2;  
int val=0;
```

```
void setup() {  
  pinMode(entrada_analogica,INPUT)  
  pinMode(ledPin,OUTPUT);  
  Serial.begin(9600);  
}
```

```
void loop() {  
  val=analogRead(entrada_analogica);  
  val=(val/4);  
  analogWrite(ledPin,val);  
  Serial.println(val);  
}
```





FIM