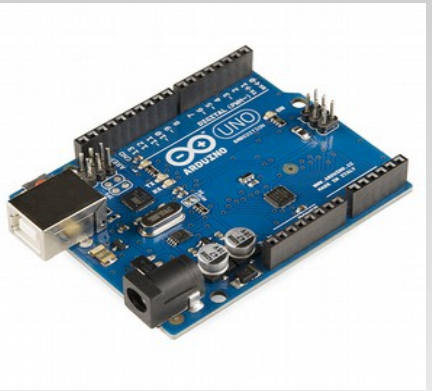
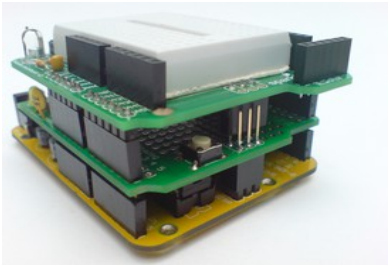
Arduino

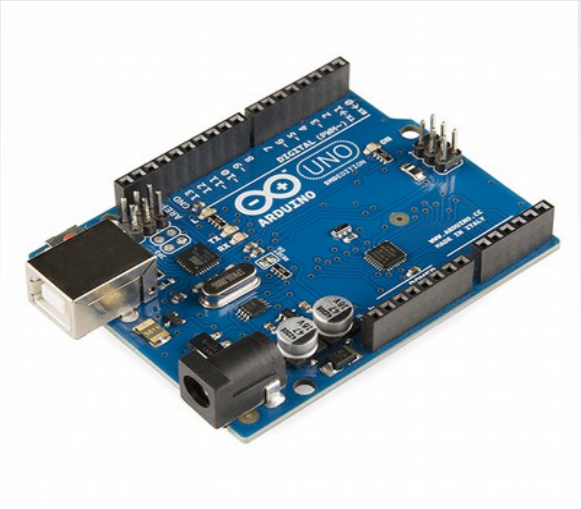


- *Single-board microcontroller*
- Microcontrolador
  - CPU, Memória, Serial, I/O
- Placa
  - Conectores, Fonte, USB, LEDs
- IDE
  - Compilador, Bibliotecas, Editor, *Burner*
  - *http://arduino.cc/en/Reference/HomePage*
- Shields
  - Display, Ethernet, Sensores, etc.



arduino UNO

- processador ATmega328P
  - 8 bits
- memória flash: 32KB
  - programa
- memória SRAM: 2K
  - dados
- clock: 16MHz



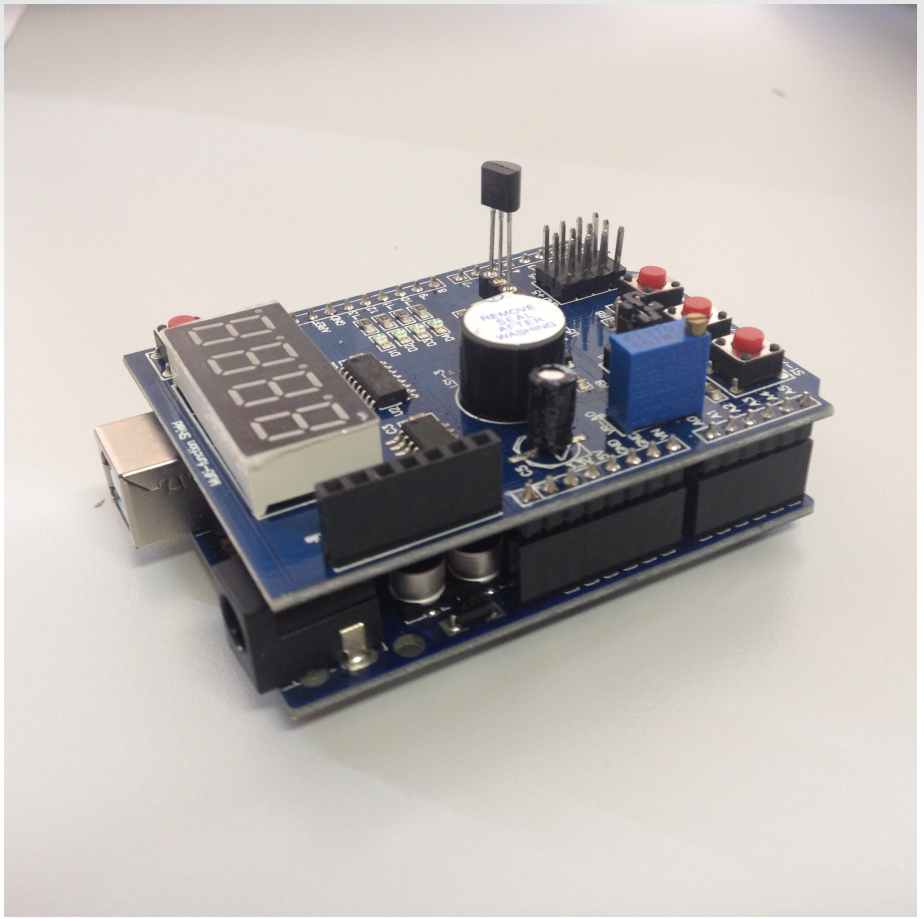
memória (bem) limitada

I/O básico

```
// configura pino para I/O
pinMode(7, INPUT);
pinMode(A2, OUTPUT);

// lê o pino
int val = digitalRead(A2);

// escreve no pino
digitalWrite(13, HIGH);
```



shield usado no curso

- 4 leds
- buzina
- 3 botões (chaves)
- display com 4 dígitos de 7 segmentos
- potenciômetro
- interface para sensor de temperatura
- interface para receptor infravermelho

pinos

- 4 leds: pinos 10, 11, 12, 13
- buzina: 3
- botões: A1, A2, A3
- potenciômetro: A0
- pinos livres: 5, 6, 9, A5

Hello World: output

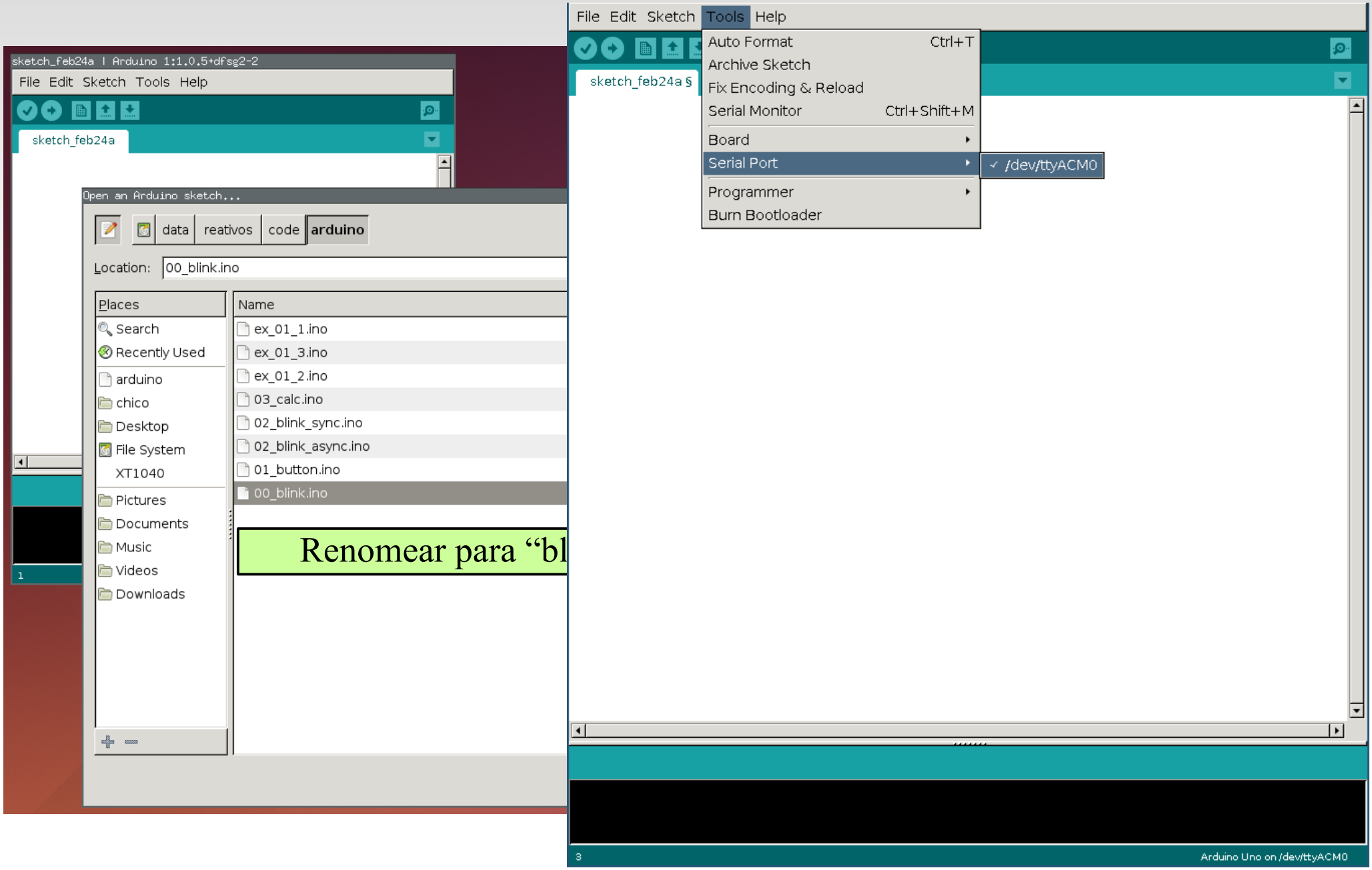
- Piscar o LED a cada 1 segundo
- `sr-17/code/arduino/00_blink.ino`

```
#define LED_PIN 13

void setup () {
   pinMode(LED_PIN, OUTPUT);     // Enable pin 13 for digital output
}

void loop () {
   digitalWrite(LED_PIN, HIGH);  // Turn on the LED
   delay(1000);                  // Wait one second (1000 milliseconds)
   digitalWrite(LED_PIN, LOW);   // Turn off the LED
   delay(1000);                  // Wait one second
}
```

Arduino IDE


sketch_feb24a | Arduino 1:1.0.5+dfsg2-2
File Edit Sketch Tools Help
sketch_feb24a
Open an Arduino sketch...
data
reativos
code
arduino
Location:
00_blink.ino
Places
Search
Recently Used
arduino
chico
Desktop
File System
XT1040
Pictures
Documents
Music
Videos
Downloads
Name
ex_01_1.ino
ex_01_3.ino
ex_01_2.ino
03_calc.ino
02_blink_sync.ino
02_blink_async.ino
01_button.ino
00_blink.ino
Renomear para “bl
File Edit Sketch Tools Help
sketch_feb24a §
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Serial Monitor Ctrl+Shift+M
Board
Serial Port
/dev/ttyACM0
Programmer
Burn Bootloader
3
Arduino Uno on /dev/ttyACM0

programando o arduino

- ambiente IDE*
  - informações sobre uso de memória
  - IDE cria diretório com arquivos de um *sketch*
- ambiente pré-processa programa e passa para compilador C/C++ (avr-gcc)
  - bibliotecas padrão: http://www.nongnu.org/avr-libc/user-manual/modules.html
  - bibliotecas Arduino

*para OSX: pode ser necessário instalar:
https://blog.sengotta.net/signed-mac-os-driver-for-winchiphead-ch340-serial-bridge/

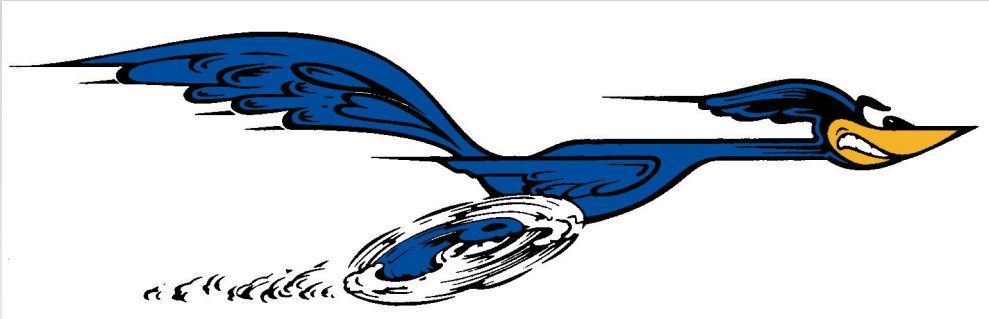
Exercício 1

- Piscar o LED a cada 1 segundo
- Parar ao pressionar o botão, mantendo o LED aceso para sempre

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

    int but = digitalRead(BUT_PIN);
    if (but) {
        digitalWrite(LED_PIN, HIGH);
        while(1);
    }
}
```

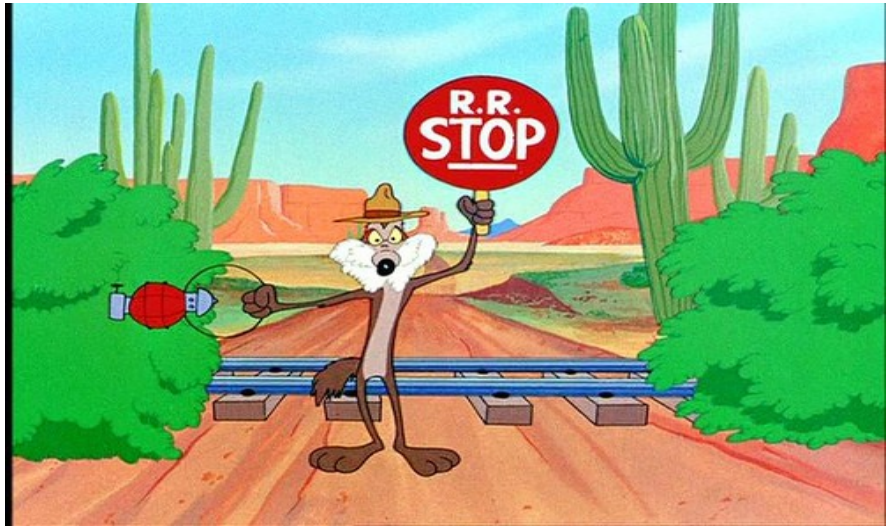
- Programa interativo!



Programa Reativo

vs

Chamadas Bloqueantes



Exercício 1 - Reativo

- Guardar *timestamp* da última mudança
- Guardar estado atual do LED

Exercício 1 - Alternativa

- Usar a função `millis()` para contar o tempo, **sem bloquear**.

millis()

Description

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

Parameters

None

Returns

Number of milliseconds since the program started (*unsigned long*)

```
void loop () {
    unsigned long time = millis();
    Serial.println(time);
    delay(1000);
}
```

https://www.arduino.cc/en/Reference/Millis

Tradeoff

- Execução sequencial com chamadas bloqueantes
  - não reativo
- Inversão de controle e variáveis de estado
  - reativo

Tarefa-02

(a conferir na próxima aula)

- Piscar o LED a cada 1 segundo

- Botão 1: Acelerar o pisca-pisca a cada pressionamento
  (somente na transição de LOW->HIGH)
- Botão 2: Desacelerar a cada pressionamento
  (somente na transição de LOW->HIGH)
- Botão 1+2 (em menos de 500ms): Parar

Modelos de Concorrência

- Modelo Assíncrono
  - ChibiOS: `http://www.chibios.org`
  - threads Java
  - Occam
- Modelo Síncrono
  - Arduino Loop
  - Céu

Modelo Assíncrono

- Por quê?
  - Como descrever e entender as partes de um sistema concorrente.
  - Vocabulário e semântica
    - execução, composição, comunicação, sincronização
- Modelo Assíncrono
  - Execução independente / Sincronização explícita
    - Threads + locks/mutexes (p-threads, Java Threads)
    - Atores + message passing (erlang, go)
- Modelo Síncrono
  - Execução dependente / Sincronização implícita
    - Arduino, Game Loops, Padrão Observer

Mini Arduino

- Projeto com sensores, atuadores, e cálculos
  - propostas até 21/3
  - `mini-arduino/PROJETO.md`
- Extras
  - interrupções
  - threads
- Apresentações em 4/4

Mini Arduino

- Entrada / Sensor
  - Distância, Movimento, Controle infra-vermelho, RTC, Acelerômetro, Teclado, Umidade, Temperatura, Luz, Botões, ...
- Saída / Atuador
  - LEDs, LCD, Motor, Servo, Buzina
- Entrada e Saída
  - Módulo RF, Serial

material disponível

Qtd	Item
2	Display LCD 16x2 (com conectores soldados)
2	Módulo RF Transmissor + Receptor 433MHz
1	Sensor de Distância Ultrassônico
1	Sensor de Movimento Presença PIR
1	Controle Remoto Ir
1	Módulo Acelerômetro
1	Módulo Bluetooth RS232 HC-6
1	Teclado Matricial De Membrana
1	Sensor De Umidade e Temperatura Dht11
2	Micro Servo 9g SG90 TowerPro
7	Sensor de Luz LDR
5	Buzzer Ativo 5V

Qtd	Item
4	Motor de passo28BYJ-48 + driver STBO811
2	Regulador 5v/3v
1	Módulo 4 Relés
2	Display 7-seg 5161A
2	DipSwitch 8
2	DipSwitch 2

Micro chaves
Potênciometros diversos
LEDs variados (pequenos e grandes)
Resistores variados
Capacitores variados

e mais empréstimos (a verificar) em
/www.inf.puc-rio.br/~abranco/eng1450/Material%20Disponível/Lista%20materia.pdf

projetos finais - exemplos

- para exemplos de projetos finais:
  - https://github.com/fsantanna-uerj/reativos-videos/