

# *Software Embarcado*

## 01 - Arduino

Francisco Sant'Anna

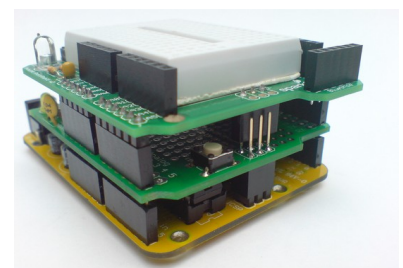
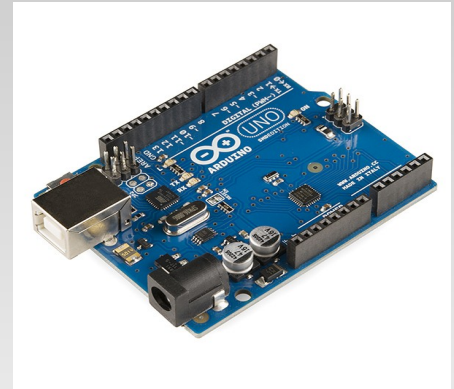
Sala 6020-B

`francisco@ime.uerj.br`

`http://github.com/fsantanna-uerj/SE`

# 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.

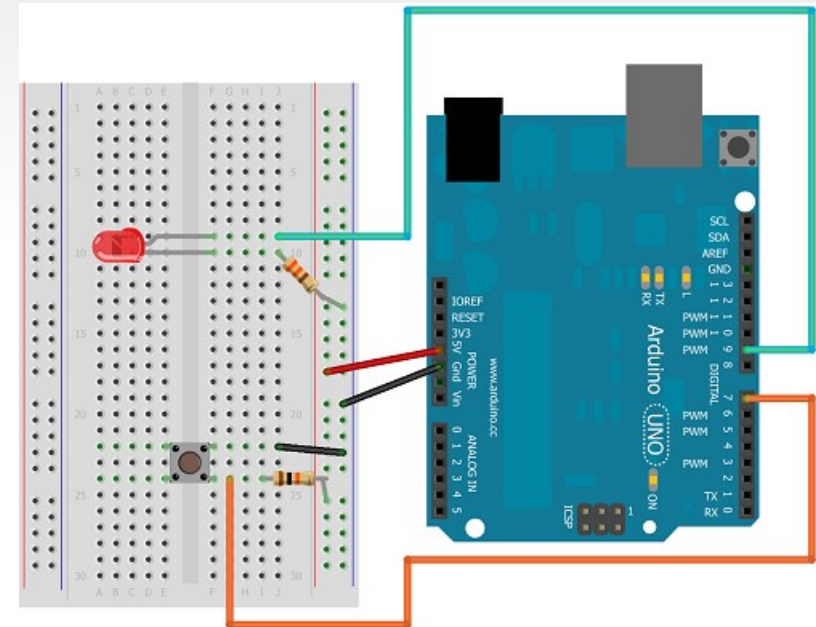


# I/O básico

```
// configura pino para I/O  
pinMode(7, INPUT);  
pinMode(9, OUTPUT);
```

```
// lê o pino  
int val = digitalRead(7);
```

```
// escreve no pino  
digitalWrite(9, HIGH);
```

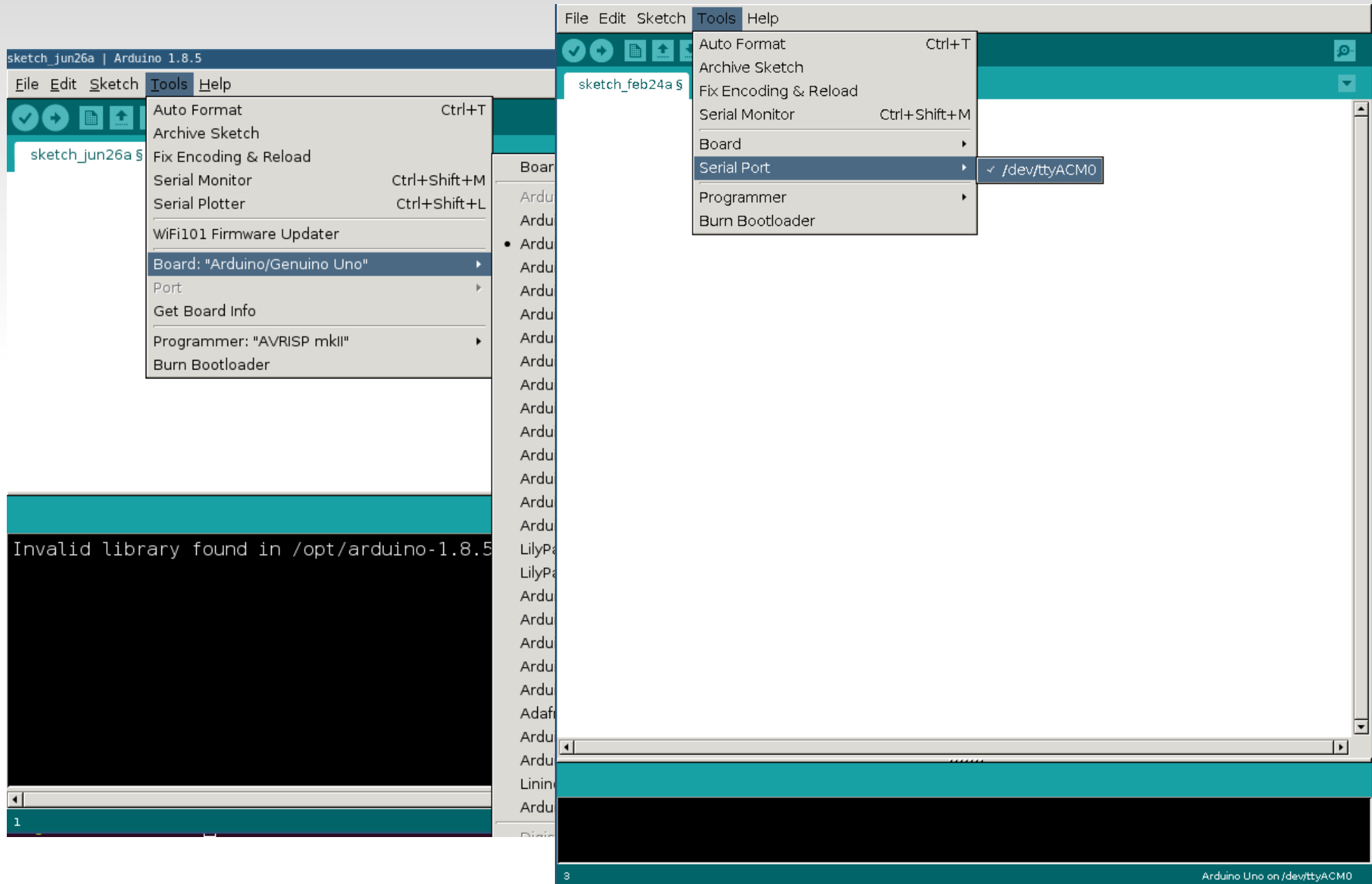


# Arduino IDE

- Abrir o terminal :

```
$ /opt/reativos/arduino-1.8.1/arduino
```

# Arduino IDE



# Hello World: output

- Piscar o LED a cada 1 segundo
- [github.com/fsantanna-uerj/SE/tree/master/Code/01-arduino/blink/](https://github.com/fsantanna-uerj/SE/tree/master/Code/01-arduino/blink/)

```
#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
}
```

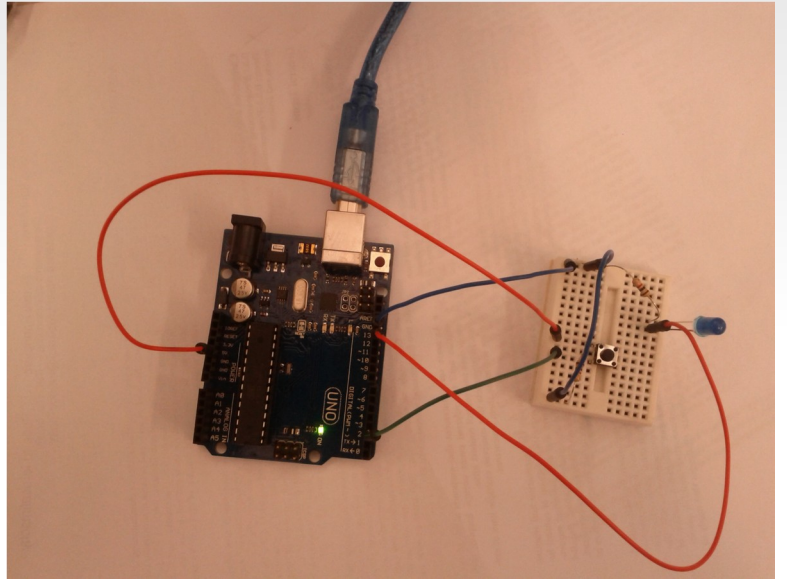
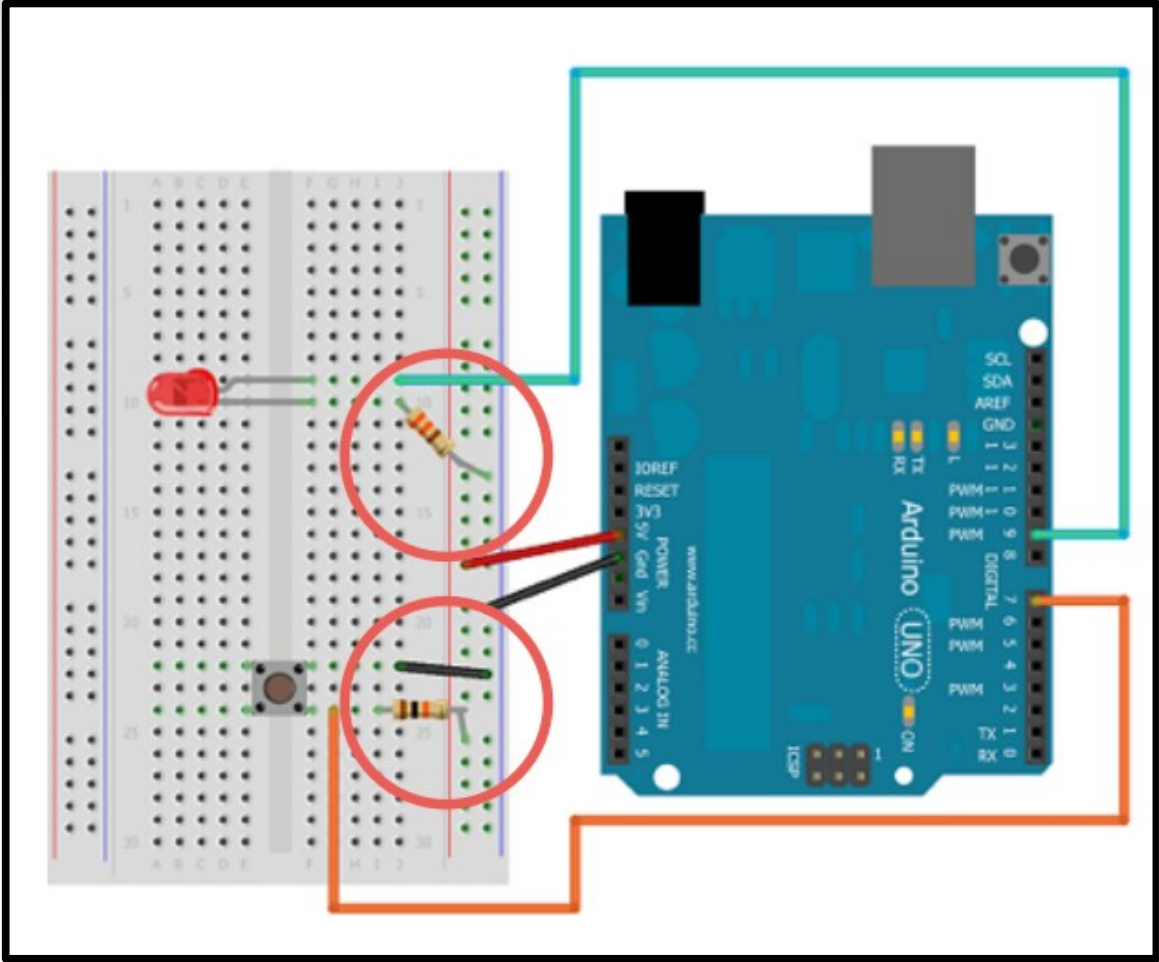
# Hello World: input

- Fazer o LED acompanhar o estado do botão
- [github.com/fsantanna-uerj/SE/tree/master/Code/01-arduino/button/](https://github.com/fsantanna-uerj/SE/tree/master/Code/01-arduino/button/)

```
#define LED_PIN 13
#define BUT_PIN 2

void setup () {
    pinMode(LED_PIN, OUTPUT);           // Enable pin 13 for digital output
    pinMode(BUT_PIN, INPUT);            // Enable pin 2 for digital input
}

void loop () {
    int but = digitalRead(BUT_PIN);    // Read button state
    digitalWrite(LED_PIN, but);        // Copy state to LED
}
```



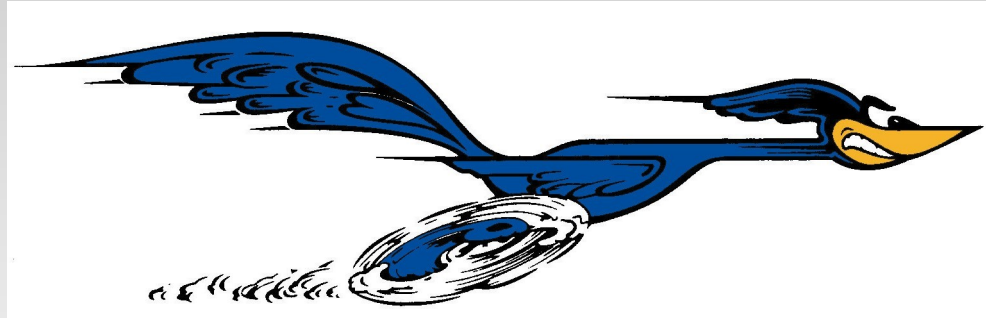


# Exercício 1

- Piscar o LED a cada 1 segundo
- Parar ao pressionar o botão, mantendo o LED aceso para sempre (mesmo após soltar o botão)

```
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);  
    }  
}
```

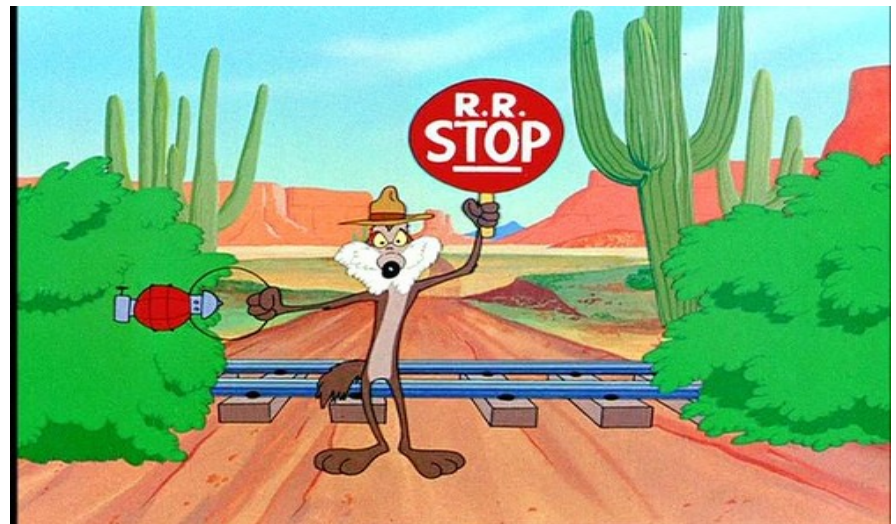
- Programa não reativo!



# Programa Reativo

VS

# Chamadas Bloqueantes



# 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 () {  
    millis(); // 1,2,4,5,...  
    delay(1);  
}
```

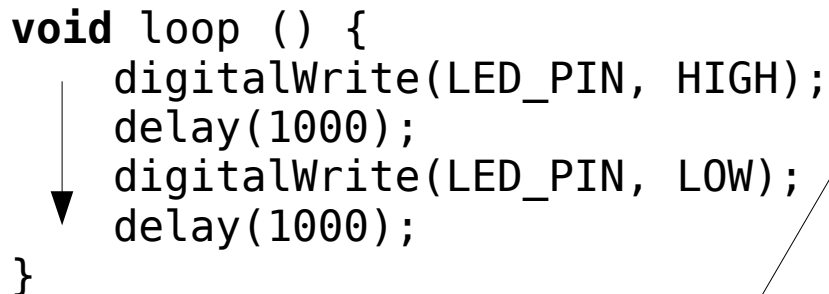
# Exercício 1 - Reativo

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

# Inversão de Controle

- Aplicação (programador) => Ambiente (dispositivos)
- Programação sequencial => Variáveis globais de estado

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



```
int state = 1;  
unsigned long old;  
void loop () {  
    unsigned long now = millis();  
    if (now >= old+1000) {  
        old = now;  
        state = !state;  
        digitalWrite(LED_PIN, state);  
    }  
}
```

- inicialização
- decodificação
- codificação

# Tradeoff

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