Céu:
Uma Linguagem de Programação Voltada ao Mundo Exterior

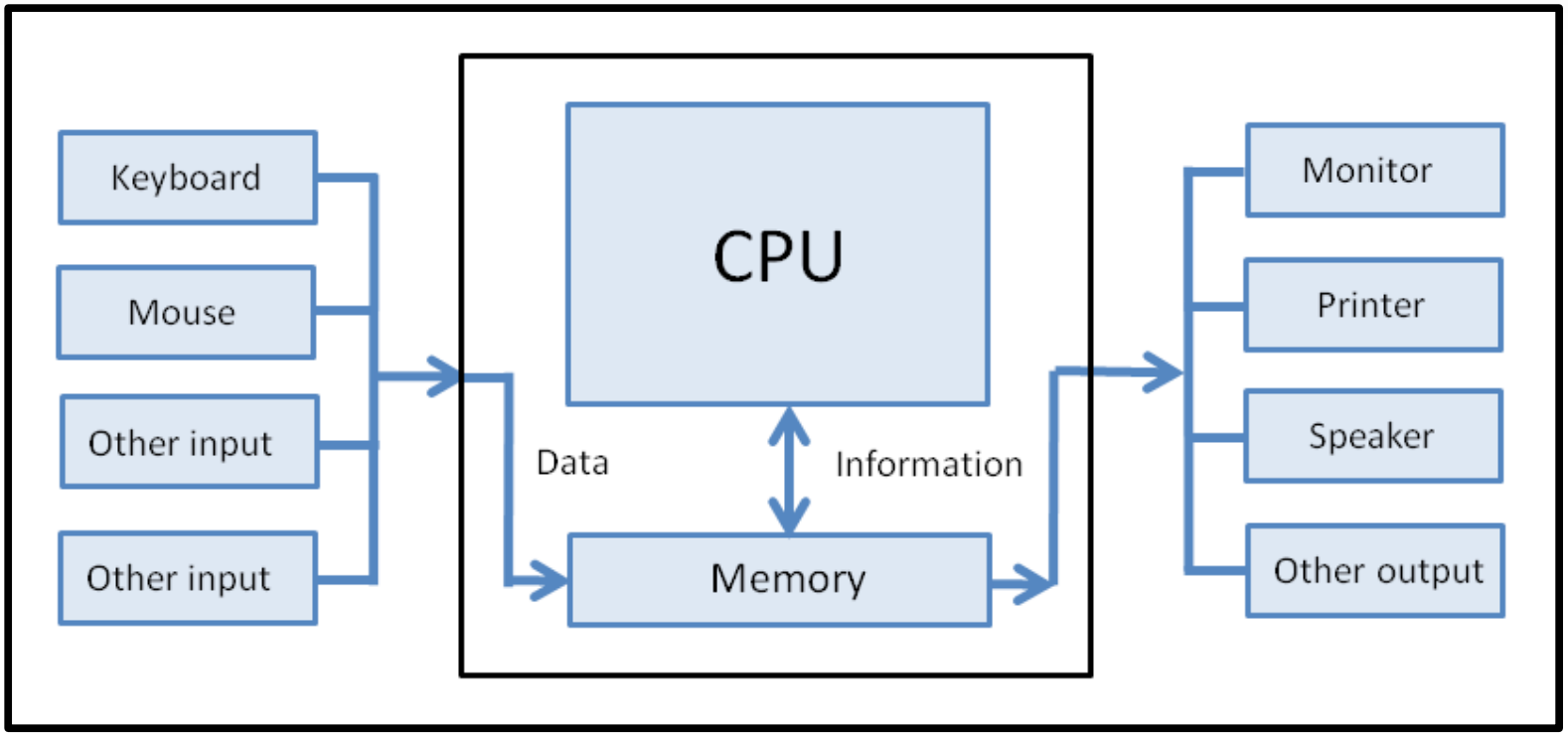
Francisco Sant'Anna

francisco@ime.uerj.br

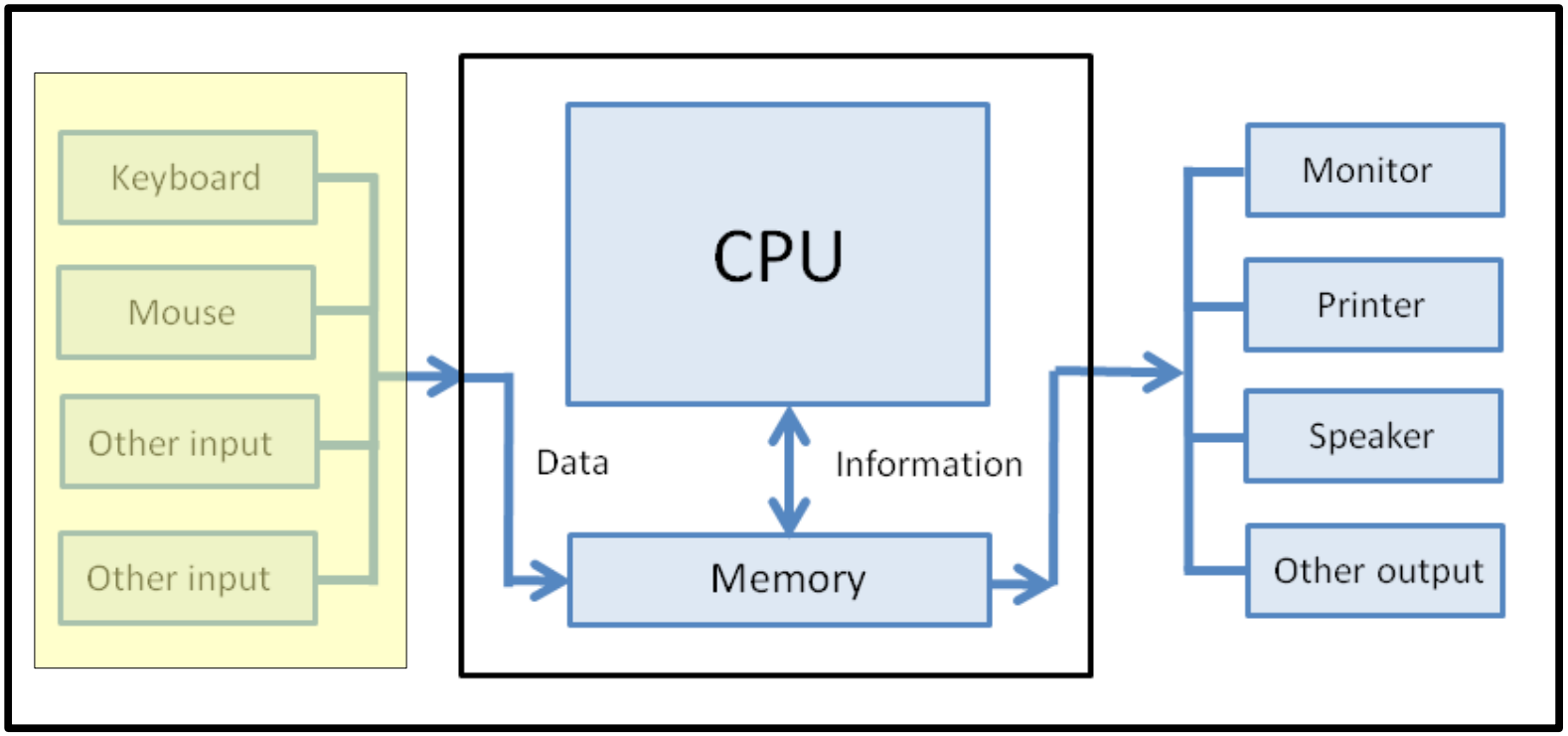
www.ceu-lang.org

```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```

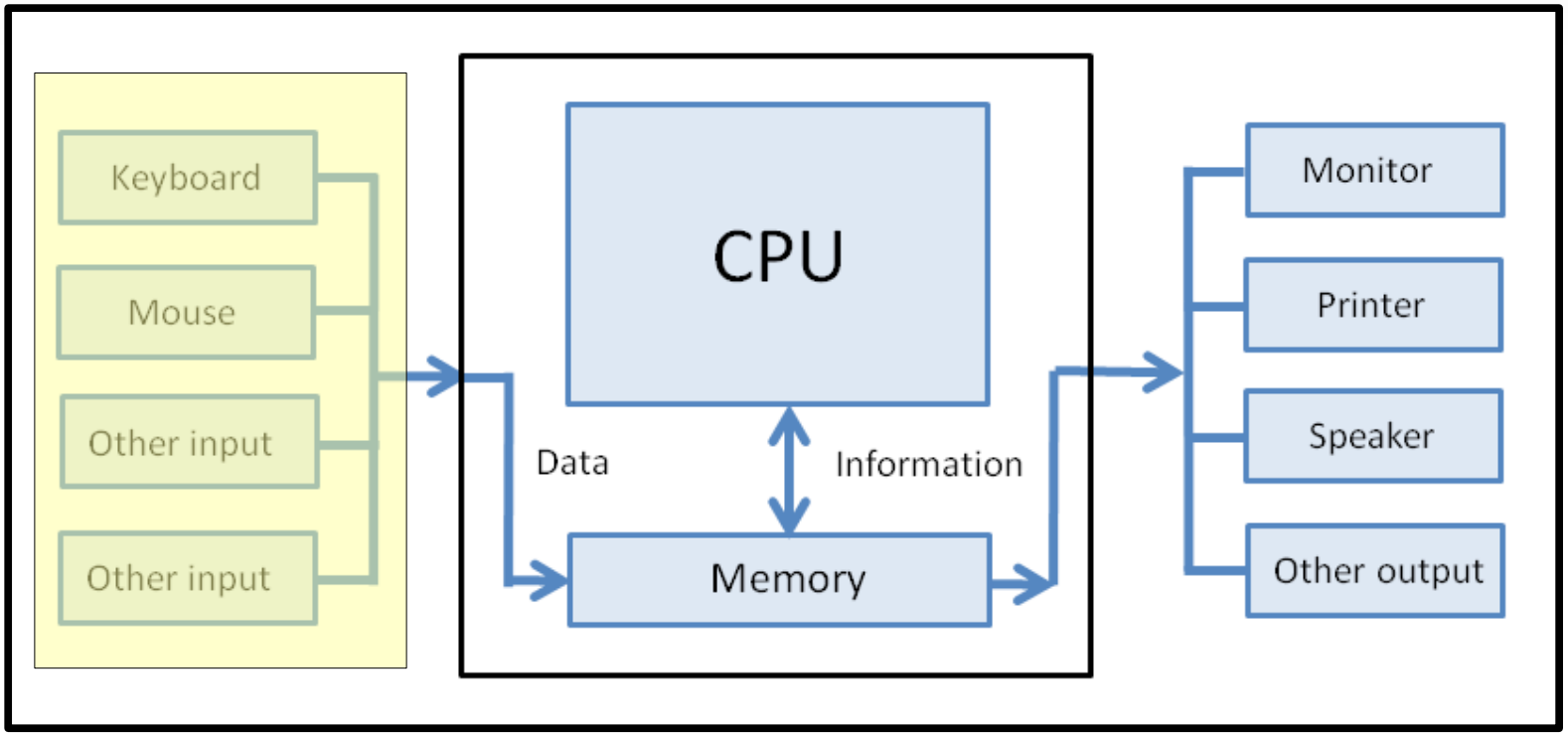
```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



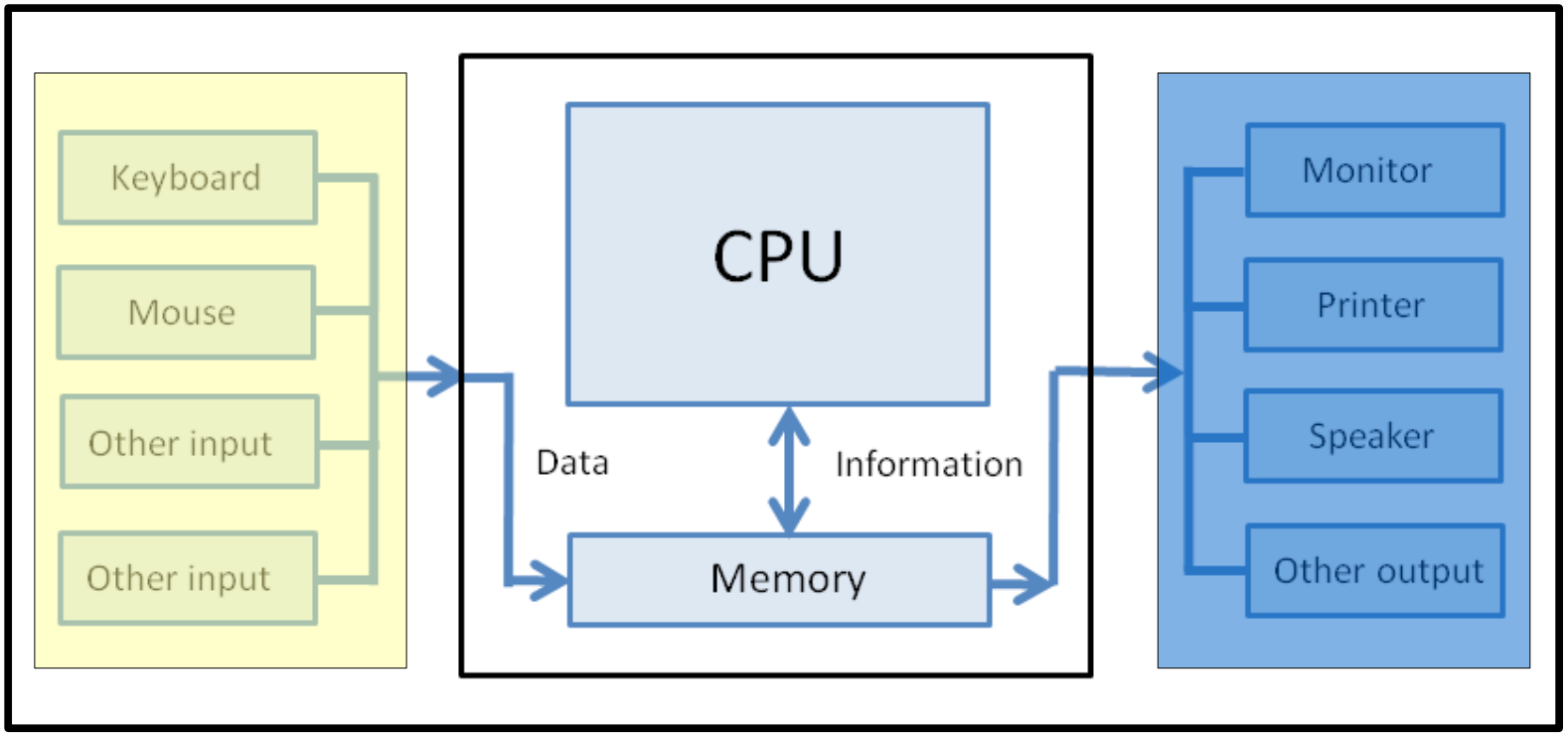
```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



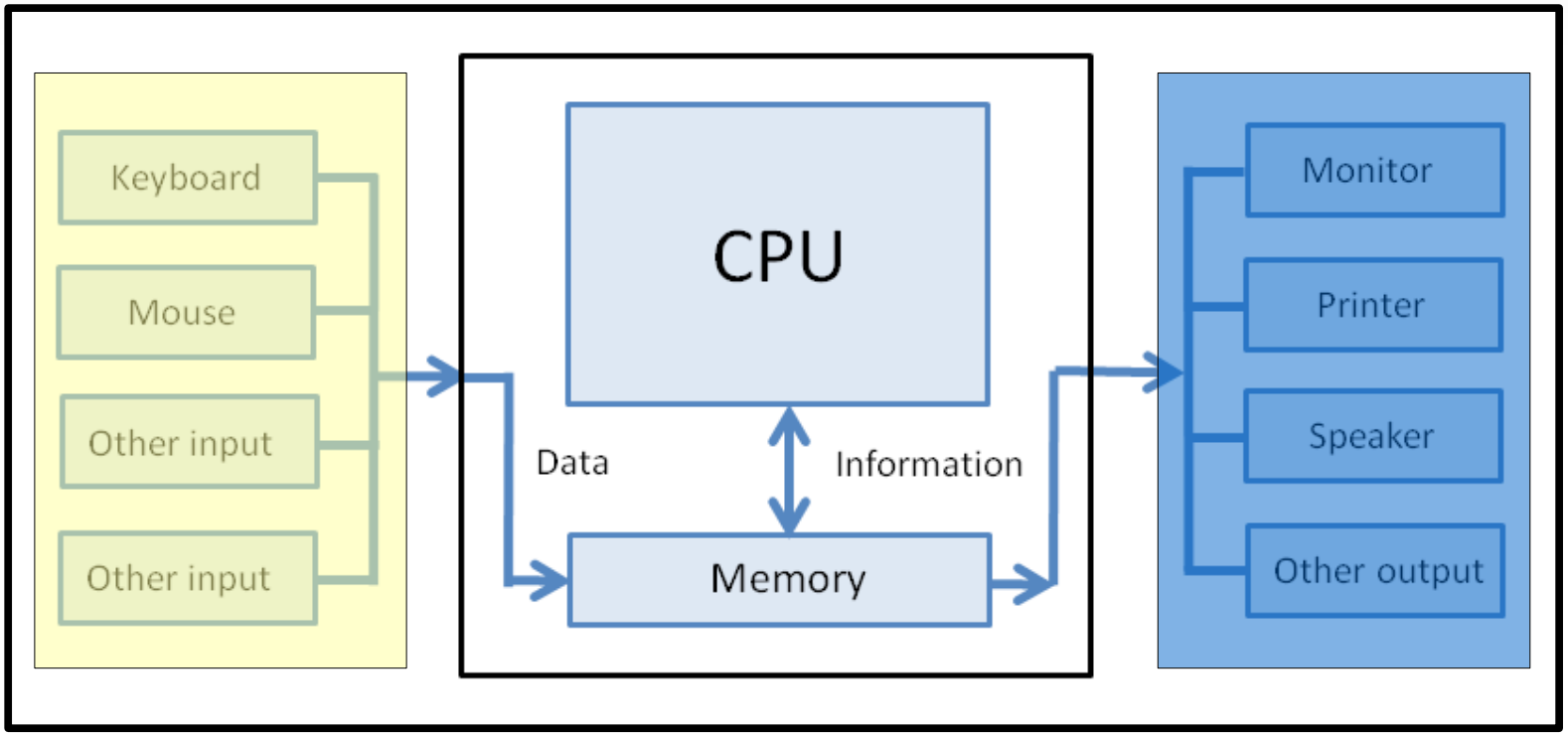
```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



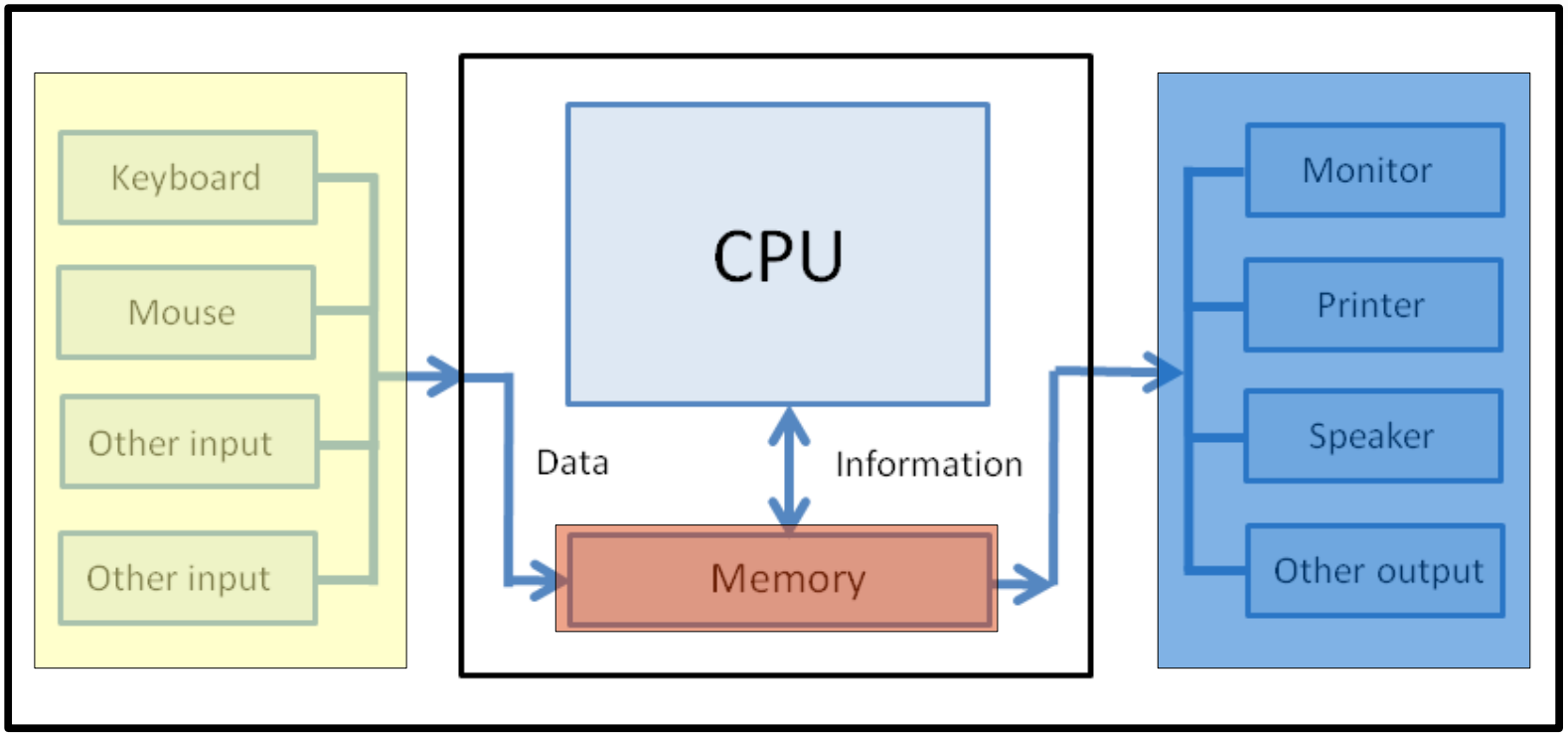
```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



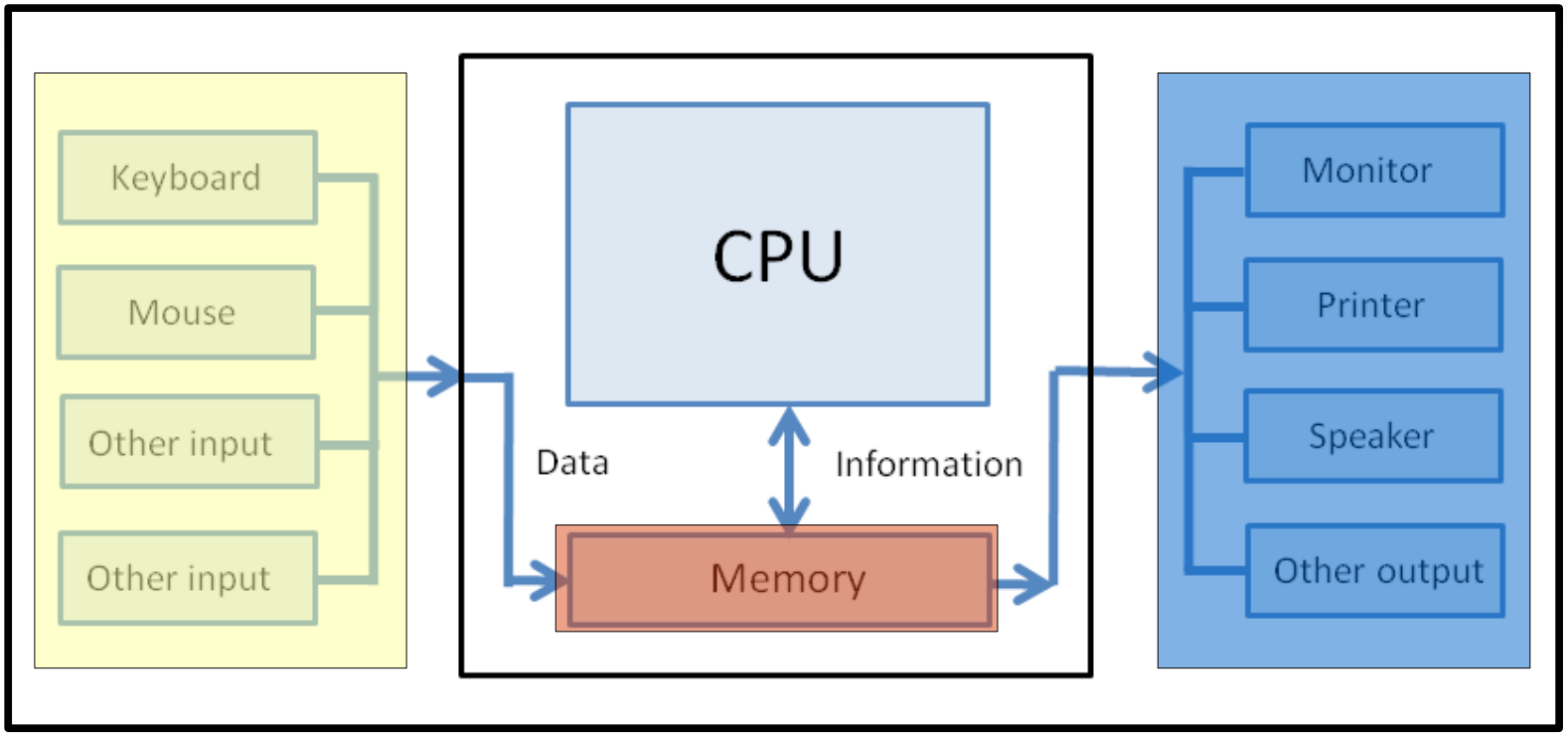
```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



```c
int n;
scanf("%d", &n);
int soma = 0;
for (int i=1; i<=n; i++) {
    soma += pow(i,2);
}
printf("soma = %d\n", soma);
```



Sistemas Transformacionais

Sistemas Transformacionais

- Transformam uma entrada em uma saída

Sistemas Transformacionais

- Transformam uma entrada em uma saída
- Origem (razão de existência) da computação

Sistemas Transformacionais

- Transformam uma entrada em uma saída
- Origem (razão de existência) da computação
  - Cálculos pesados

Sistemas Transformacionais

- Transformam uma entrada em uma saída
- Origem (razão de existência) da computação
  - Cálculos pesados
- Base da formação educacional em computação

Sistemas Transformacionais

- Transformam uma entrada em uma saída
- Origem (razão de existência) da computação
  - Cálculos pesados
- Base da formação educacional em computação
  - Algoritmos e estruturas de dados

Sistemas Transformacionais

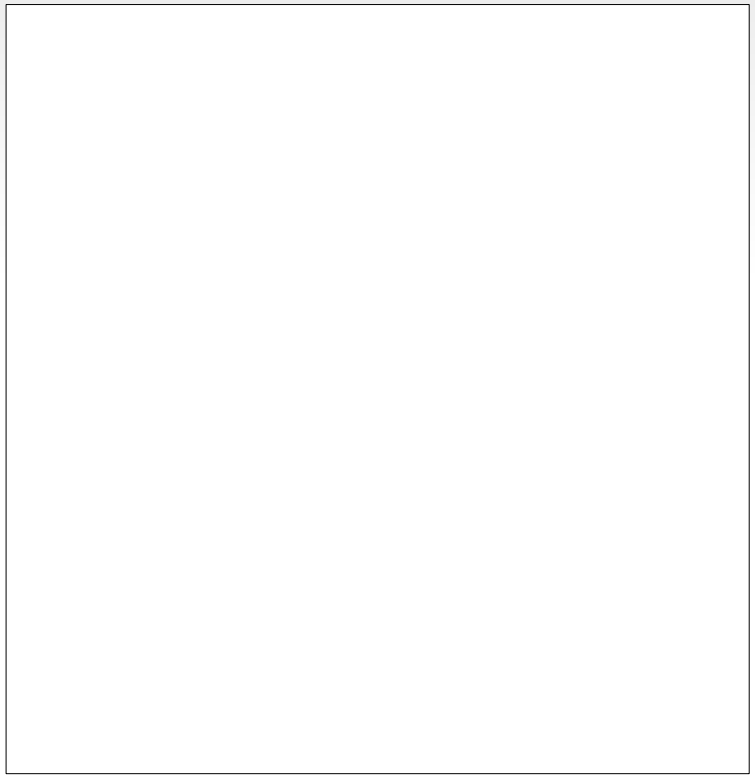
- Transformam uma entrada em uma saída
- Origem (razão de existência) da computação
  - Cálculos pesados
- Base da formação educacional em computação
  - Algoritmos e estruturas de dados
- Influência no design das linguagens

Sistemas Transformacionais

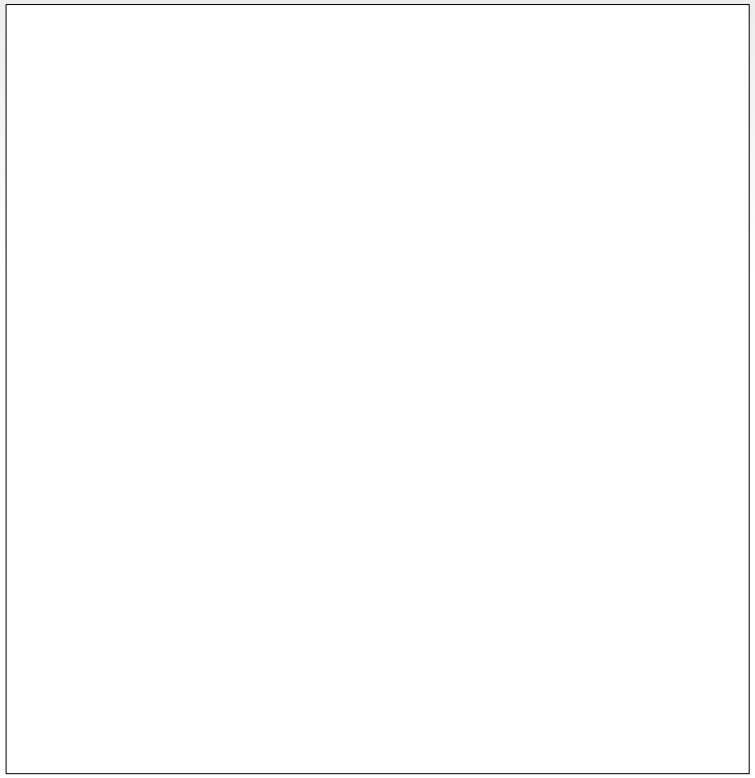
- Transformam uma entrada em uma saída
- Origem (razão de existência) da computação
  - Cálculos pesados
- Base da formação educacional em computação
  - Algoritmos e estruturas de dados
- Influência no design das linguagens
  - Vocabulário e modelo de execução

Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

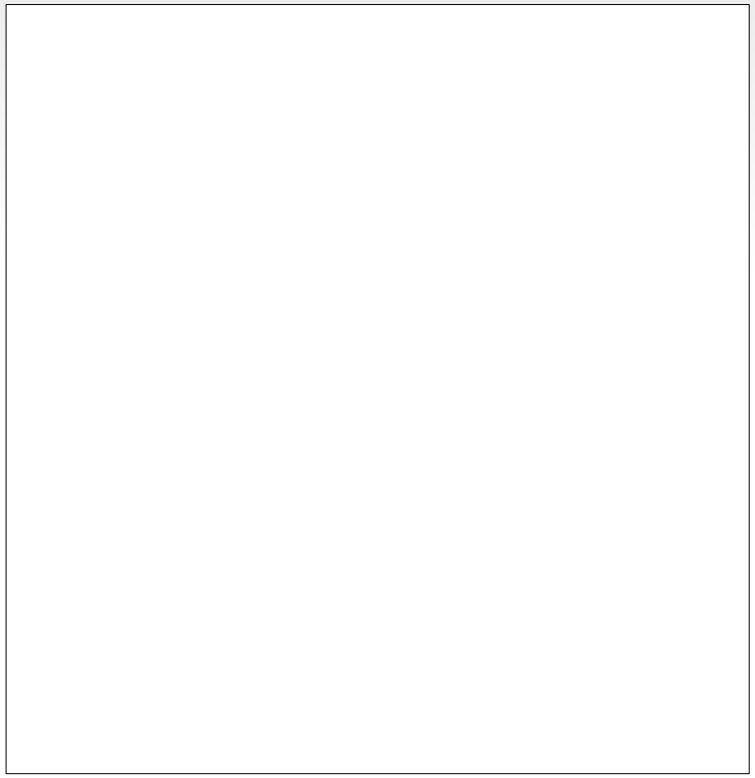
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.



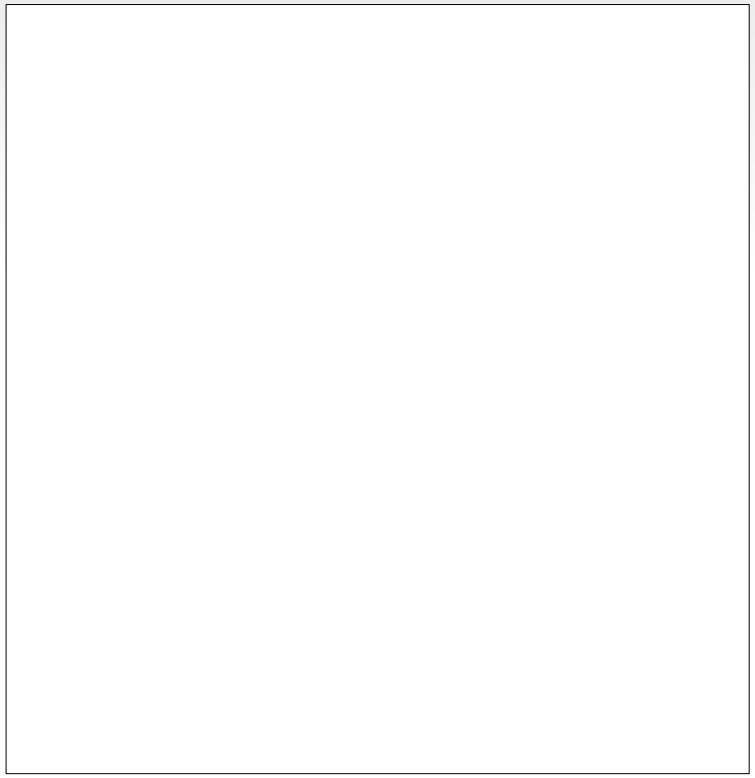
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.



Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.



Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

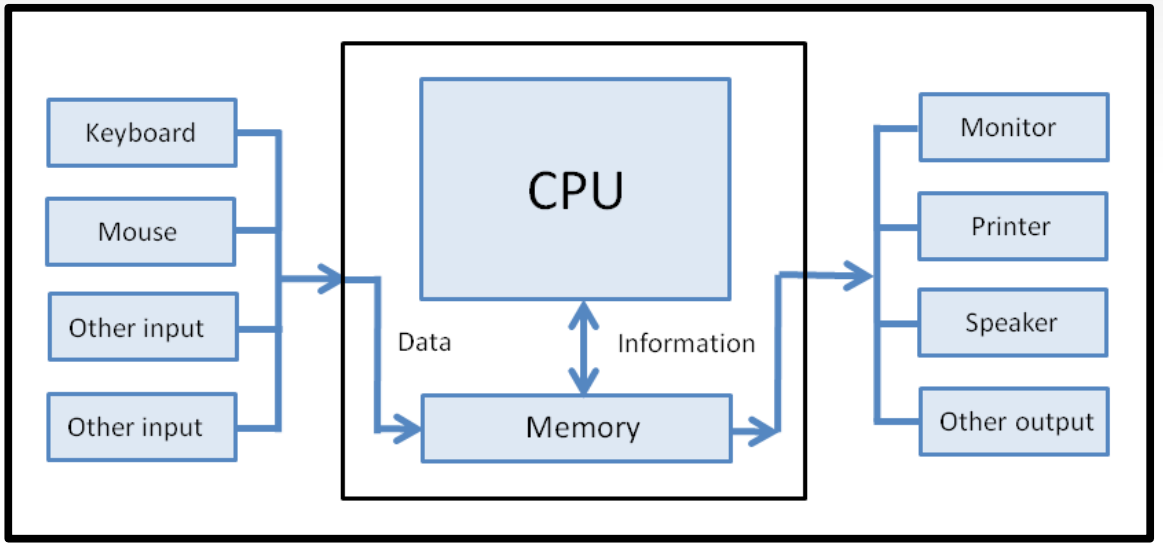


Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
   await BUTTON;
with
   await 1min;
with
   loop do
      await 1s;
      emit LED(high);
      await 1s;
      emit LED(off);
   end
end
```

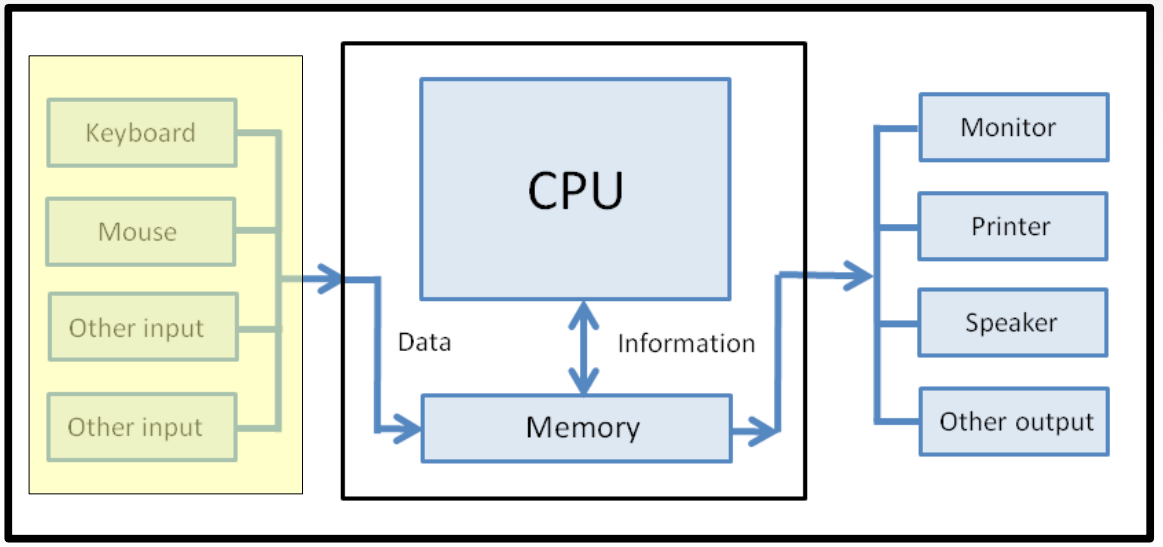
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



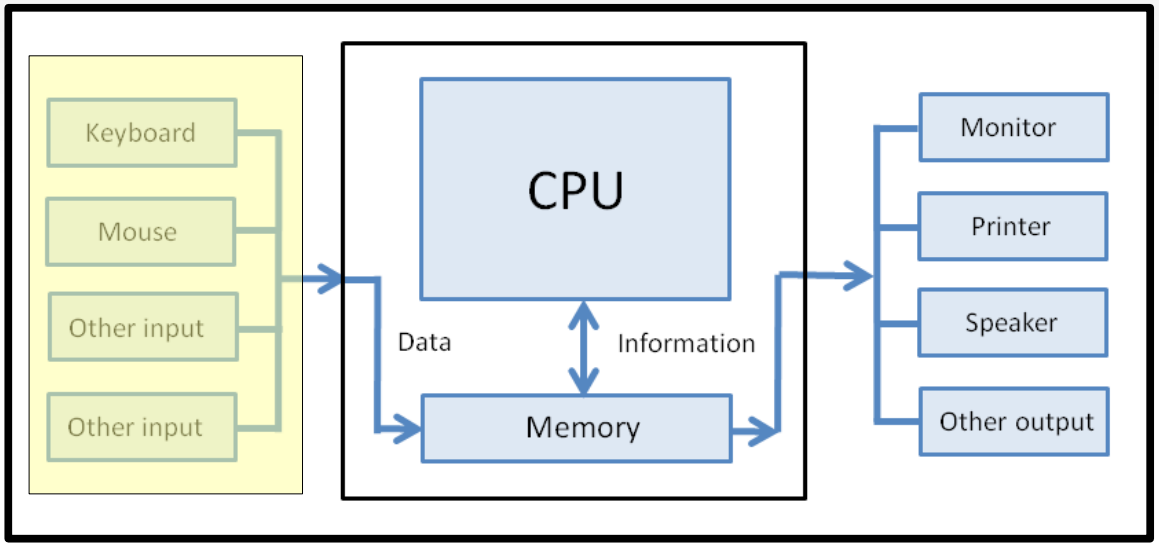
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



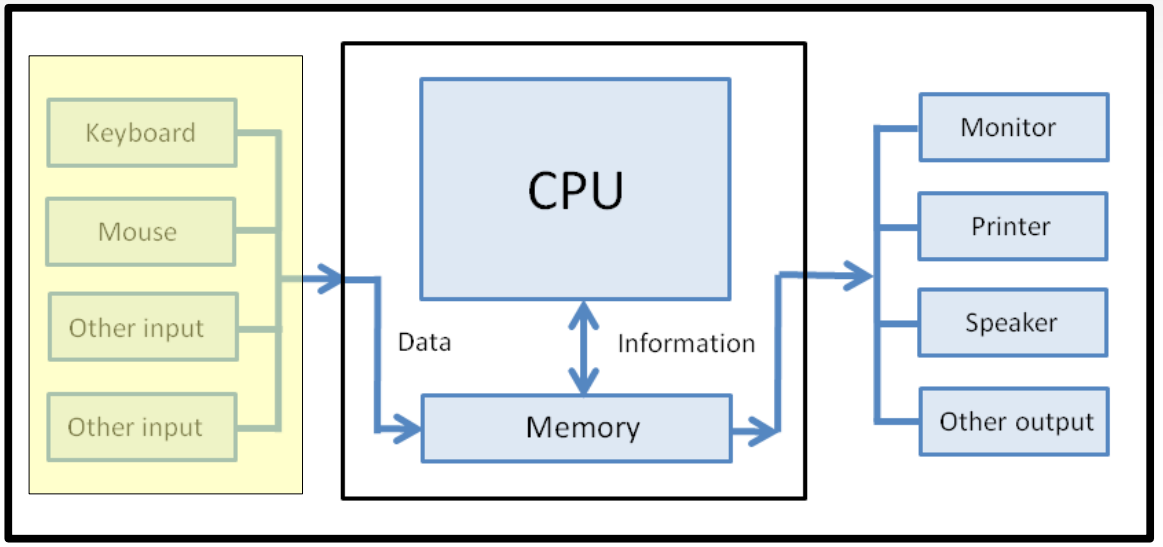
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



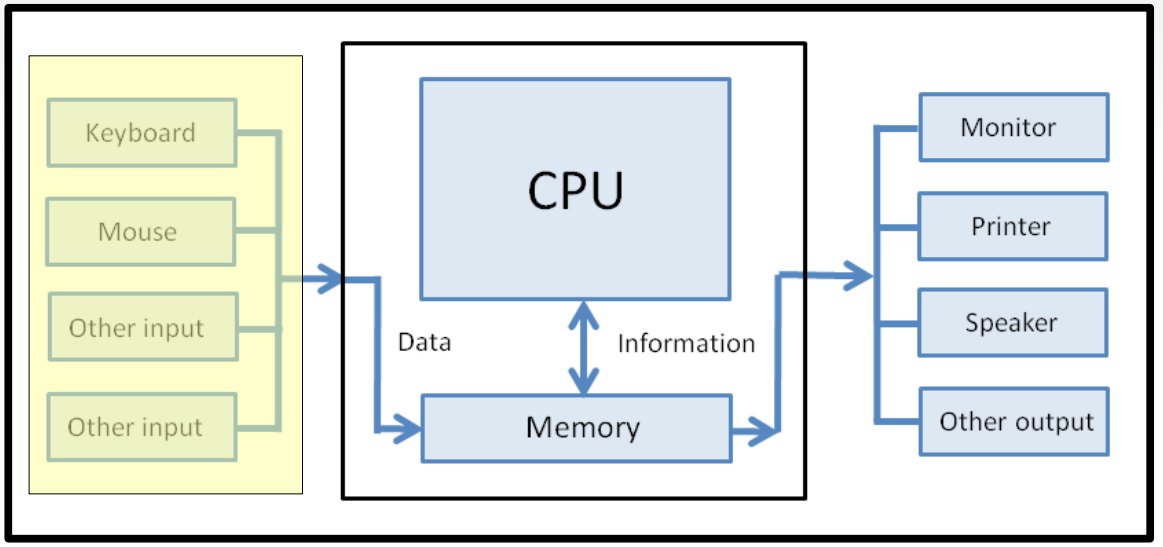
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



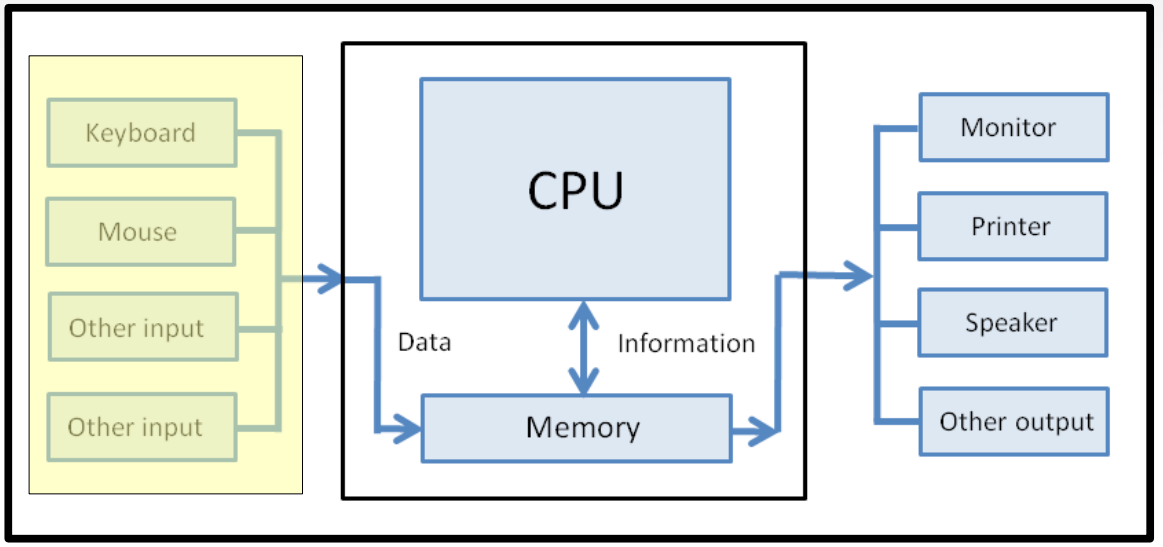
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



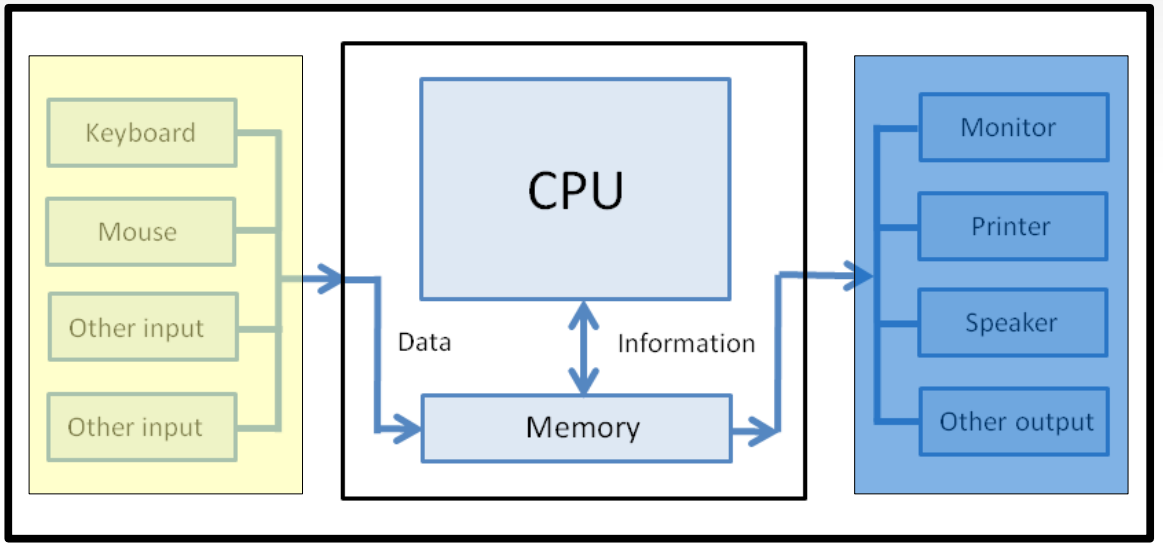
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



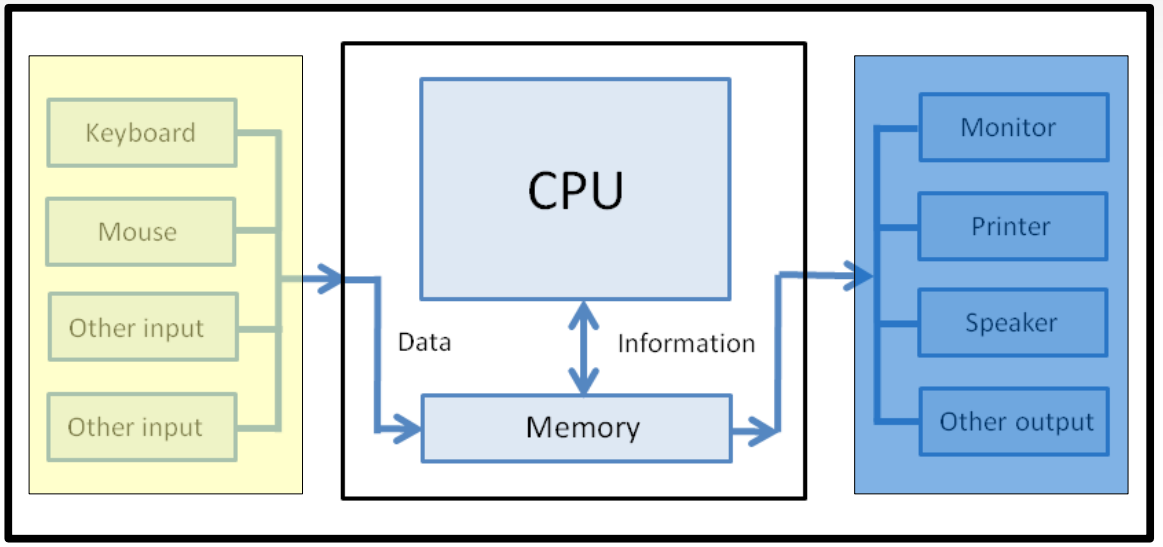
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



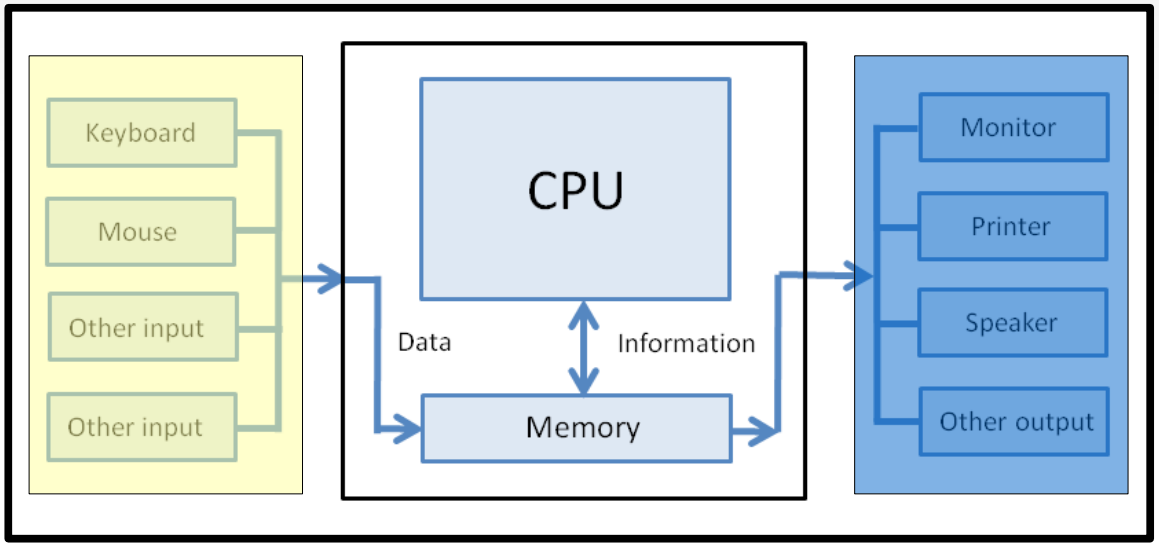
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



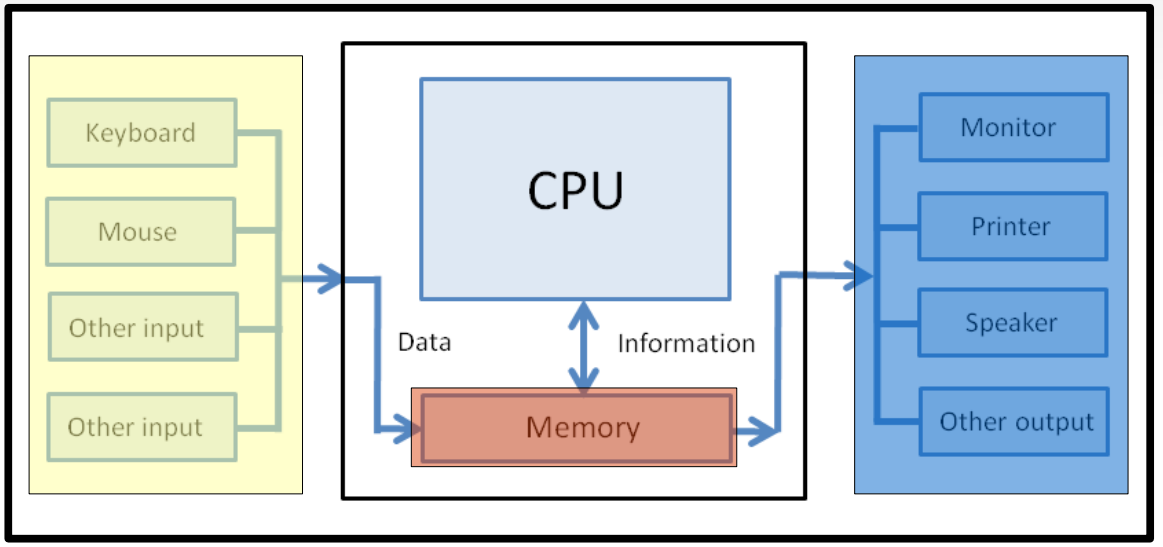
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



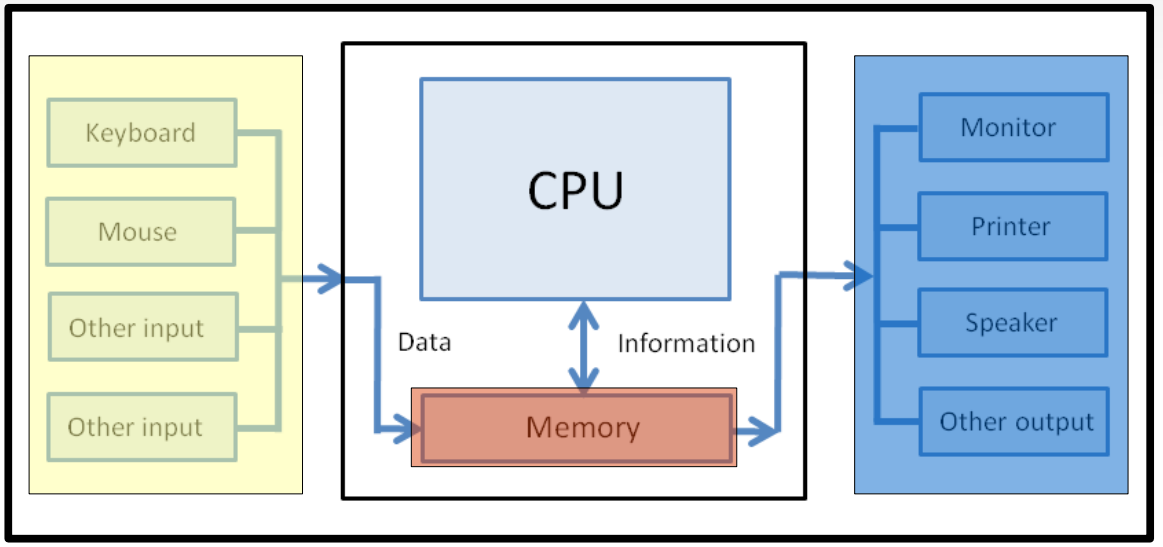
Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



Piscar um LED com frequência de 1 segundo.
Interromper ao apertar o botão ou após 1 minuto.

```
par/or do
    await BUTTON;
with
    await 1min;
with
    loop do
        await 1s;
        emit LED(high);
        await 1s;
        emit LED(off);
    end
end
```



www.ceu-lang.org

Sistemas Reativos

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo
- “Novas” aplicações

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo
- “Novas” aplicações
  - GUIs, jogos, *apps*, robôs, internet das coisas

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo
- “Novas” aplicações
  - GUIs, jogos, *apps*, robôs, internet das coisas
- Negligenciados na formação em computação

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo
- “Novas” aplicações
  - GUIs, jogos, *apps*, robôs, internet das coisas
- Negligenciados na formação em computação
  - Ao menos na base (fundamentos)

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo
- “Novas” aplicações
  - GUIs, jogos, *apps*, robôs, internet das coisas
- Negligenciados na formação em computação
  - Ao menos na base (fundamentos)
- Pouca influência no design das linguagens

Sistemas Reativos

- Interagem continuamente e em **tempo real** com o mundo externo
- “Novas” aplicações
  - GUIs, jogos, *apps*, robôs, internet das coisas
- Negligenciados na formação em computação
  - Ao menos na base (fundamentos)
- Pouca influência no design das linguagens
  - Vocabulário e modelo de execução

	Transformacional (C)	Reativo (Céu)
Vocabulário		
Modelo de Execução		

	Transformacional (C)	Reativo (Céu)

Vocabulário	`scanf(...)`
	`printf(...)`
	`sin(...)`	

Modelo de Execução		

	Transformacional (C)	Reativo (Céu)

Vocabulário	`scanf(...)`
	`printf(...)`
	`sin(...)`
		`await` `BUTTON`
		`emit` `LED`
		`call` `Sin(...)`

Modelo de Execução		

	Transformacional (C)	Reativo (Céu)

Vocabulário	`scanf(...)`
	`printf(...)`
	`sin(...)`
		`await` `BUTTON`
		`emit` `LED`
		`call` `Sin(...)`

Modelo de Execução	Sequencial Von Neumann	

	Transformacional (C)	Reativo (Céu)

Vocabulário	`scanf(...)`
	`printf(...)`
	`sin(...)`
		`await` `BUTTON`
		`emit` `LED`
		`call` `Sin(...)`

Modelo de Execução	Sequencial Von Neumann	Concorrente **`par/or`**

	Transformacional (C)	Reativo (Céu)

Vocabulário	`scanf(...)`
	`printf(...)`
	`sin(...)`
		`await` `BUTTON`
		`emit` `LED`
		`call` `Sin(...)`

Modelo de Execução	Sequencial Von Neumann	Concorrente **`par/or`**
	“Olhar para dentro”	

	Transformacional (C)	Reativo (Céu)

Vocabulário	`scanf(...)`
	`printf(...)`
	`sin(...)`
		`await` `BUTTON`
		`emit` `LED`
		`call` `Sin(...)`

Modelo de Execução	Sequencial Von Neumann	Concorrente **`par/or`**
	“Olhar para dentro”	“Olhar para fora”

Visão Geral de Céu

Visão Geral de Céu

- Reativa
  - ambiente no controle: *eventos*

Visão Geral de Céu

- Reativa
  - ambiente no controle: *eventos*
- Imperativa
  - sequências, laços, atribuições

Visão Geral de Céu

- Reativa
  - ambiente no controle: *eventos*
- Imperativa
  - sequências, laços, atribuições
- Concorrente
  - múltiplas linhas de execução: *trilhas*

Visão Geral de Céu

- Reativa
  - ambiente no controle: *eventos*
- Imperativa
  - sequências, laços, atribuições
- Concorrente
  - múltiplas linhas de execução: *trilhas*
- Síncrona
  - trilhas sincronizam em cada evento externo

Visão Geral de Céu

- Reativa
  - ambiente no controle: *eventos*
- Imperativa
  - sequências, laços, atribuições
- Concorrente
  - múltiplas linhas de execução: *trilhas*
- Síncrona
  - trilhas sincronizam em cada evento externo
  - **trilhas estão sempre esperando**

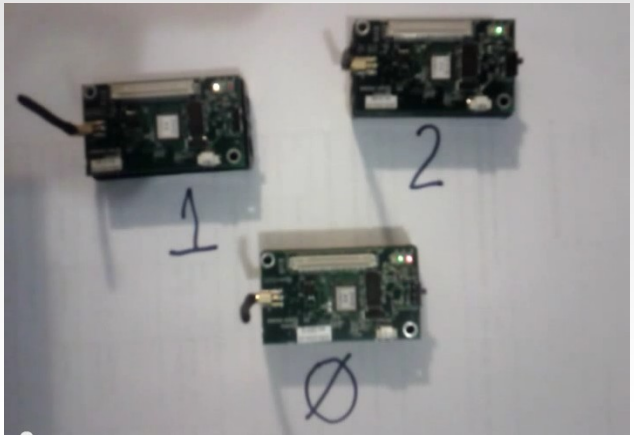
Visão Geral de Céu

- Reativa
  - ambiente no controle: *eventos*
- Imperativa
  - sequências, laços, atribuições
- Concorrente
  - múltiplas linhas de execução: *trilhas*
- Síncrona
  - trilhas sincronizam em cada evento externo
  - **trilhas estão sempre esperando**
- Determinística
  - sempre produz o mesmo resultado para uma dada linha de tempo

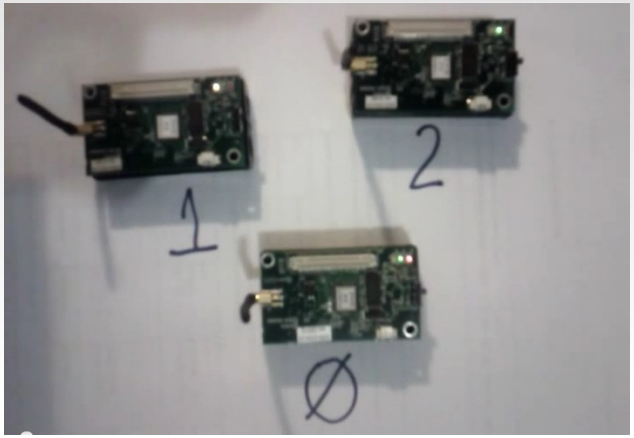
Usos de Céu

Usos de Céu

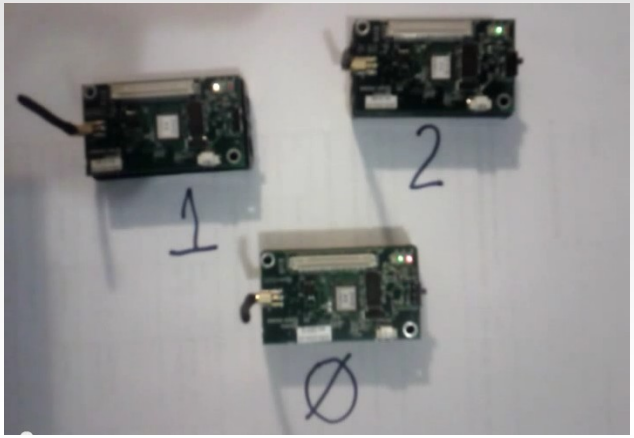
Usos de Céu



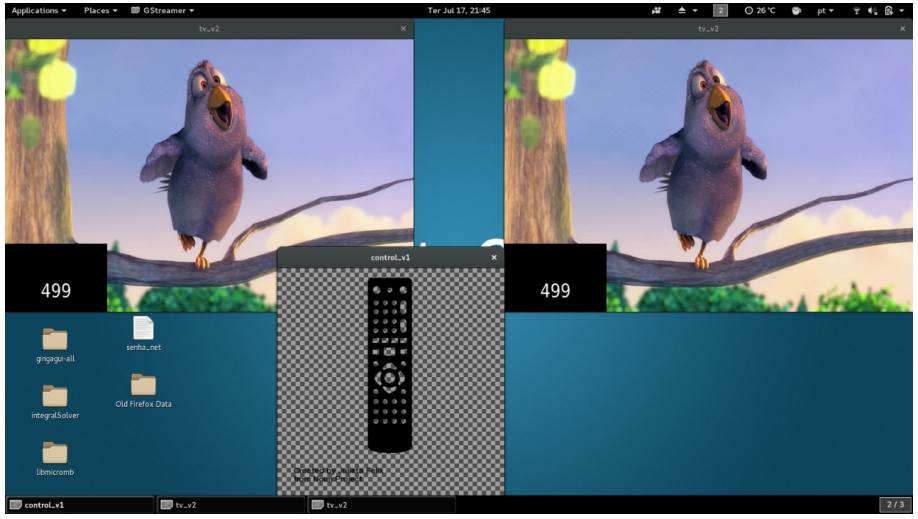
Usos de Céu



Usos de Céu


1
2
0


Released: 3/20 Out: 3 Saved: 0/10
5
5
5
10
4
5
40


Applications
Places
GStreamer
Ter Jul 17, 21:45
tv_v2
control_v1
499
499
gingagui-all
senha_net
integralSolver
Old Firefox Data
libmicromb
2 / 3

Eficiência Energética para Software IoT: Uma abordagem no Nível de Linguagem de Programação

Francisco Sant'Anna

francisco@ime.uerj.br

www.ceu-lang.org

IoT e Consumo

IoT e Consumo

- 15 bilhões de dispositivos conectados “tradicionais” em 2015 (e.g., celulares, TVs inteligentes).

IoT e Consumo

- 15 bilhões de dispositivos conectados “tradicionais” em 2015 (e.g., celulares, TVs inteligentes).
- 75 bilhões até 2025 com a IoT (e.g., lâmpadas e vestíveis inteligentes.

IoT e Consumo

- 15 bilhões de dispositivos conectados “tradicionais” em 2015 (e.g., celulares, TVs inteligentes).
- 75 bilhões até 2025 com a IoT (e.g., lâmpadas e vestíveis inteligentes.
- **Maior parte do consumo em modo em espera (“standby”).**

IoT e Consumo

- 15 bilhões de dispositivos conectados “tradicionais” em 2015 (e.g., celulares, TVs inteligentes).
- 75 bilhões até 2025 com a IoT (e.g., lâmpadas e vestíveis inteligentes.
- **Maior parte do consumo em modo em espera (“standby”).**
- Standby de dispositivos conectados é uma das seis frentes do “Plano de Eficiência Energética” da AIE/G20.
    - https://www.iea-4e.org/projects/g20

Objetivos

Objetivos

1. Eficiência energética através do uso rigoroso de **standby**.

Objetivos

1. Eficiência energética através do uso rigoroso de **standby**.
2. Foco em arquiteturas **restritas** que formam a IoT.

Objetivos

1. Eficiência energética através do uso rigoroso de **standby**.
2. Foco em arquiteturas **restritas** que formam a IoT.
3. Mecanismos de standby no nível de **linguagem de programação** para maior escalabilidade.

Objetivos

1. Eficiência energética através do uso rigoroso de **standby**.
2. Foco em arquiteturas **restritas** que formam a IoT.
3. Mecanismos de standby no nível de **linguagem de programação** para maior escalabilidade.
4. Mecanismos de standby **transparentes**/não intrusivos para reduzir as barreiras de adoção.

Abordagem Geral

(standby, restritos, linguagem, transparente)

Abordagem Geral
(standby, restritos, linguagem, transparente)

- Forçar estados ociosos de execução

Abordagem Geral
(standby, restritos, linguagem, transparente)

- Forçar estados ociosos de execução
- Inferir *sleep mode* mais profundo

Abordagem Geral
(standby, restritos, linguagem, transparente)

- Forçar estados ociosos de execução
- Inferir *sleep mode* mais profundo
- Colocar o dispositivo para dormir

Abordagem Geral
(standby, restritos, linguagem, transparente)

- Forçar estados ociosos de execução
- Inferir *sleep mode* mais profundo
- Colocar o dispositivo para dormir
- Acordar somente por interrupções

Um Exemplo Ilustrativo

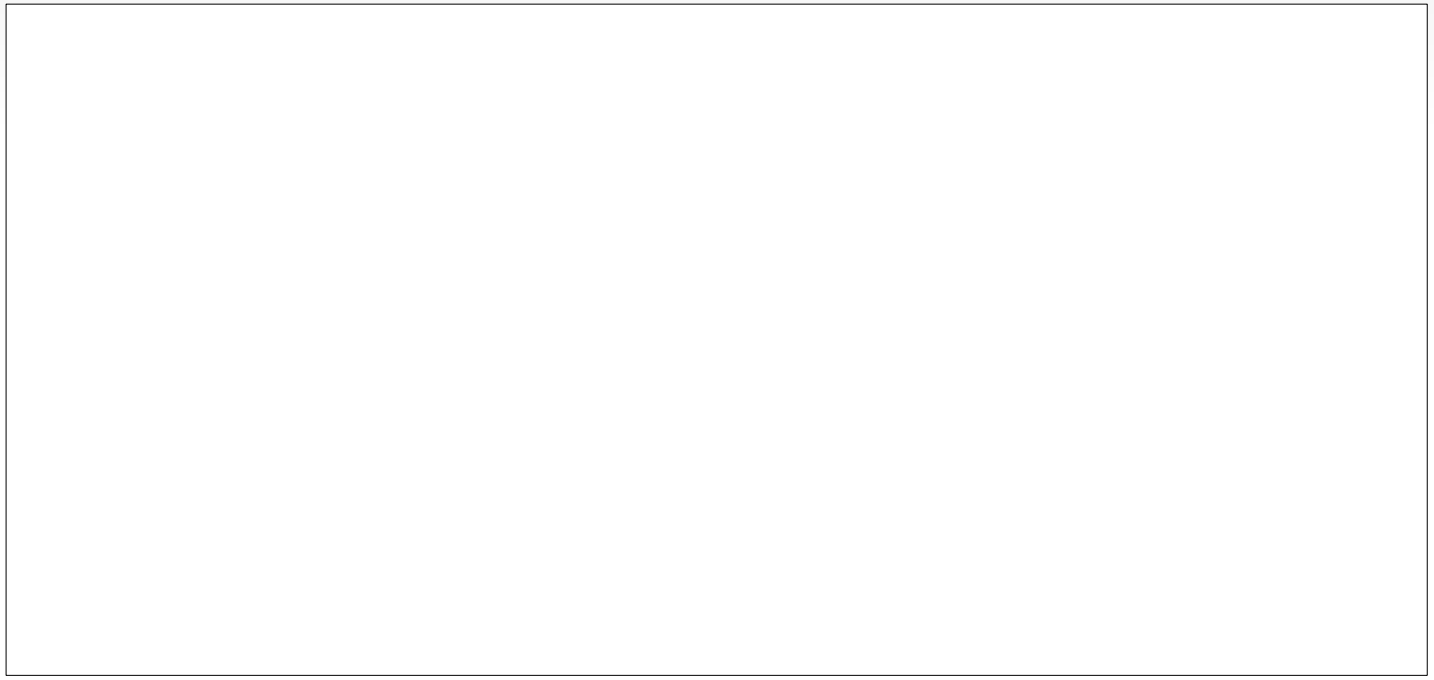
A cada segundo, ler um sensor e transmitir o valor.

Parar assim que receber uma mensagem.

Um Exemplo Ilustrativo

A cada segundo, ler um sensor e transmitir o valor.

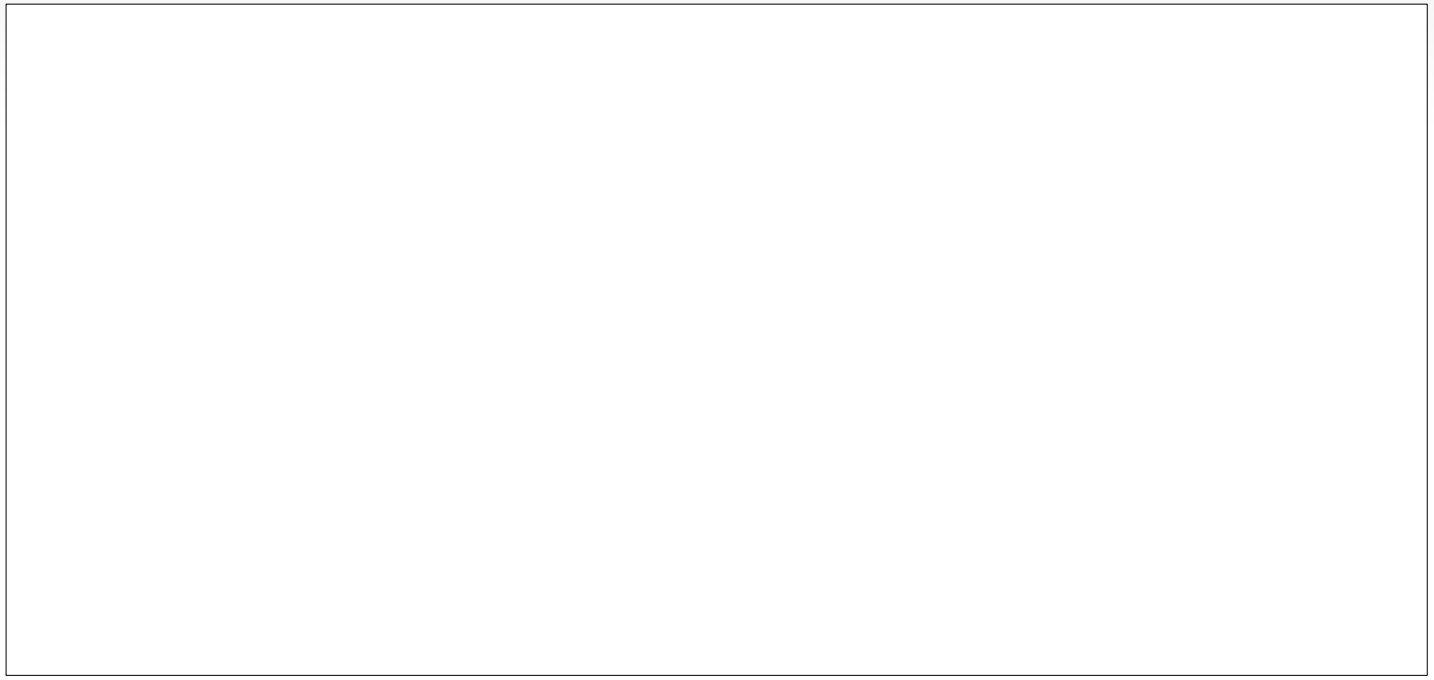
Parar assim que receber uma mensagem.



Um Exemplo Ilustrativo

A cada segundo, ler um sensor e transmitir o valor.

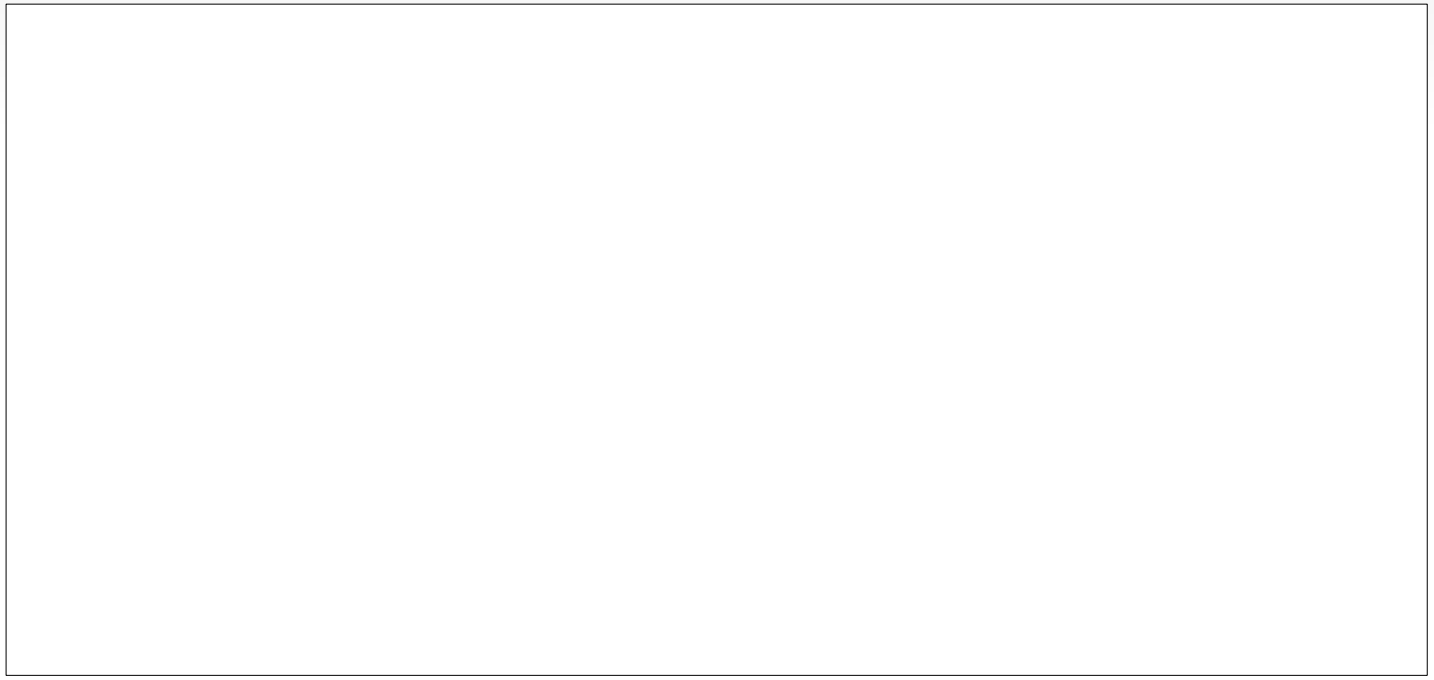
Parar assim que receber uma mensagem.



Um Exemplo Ilustrativo

A cada segundo, ler um sensor e transmitir o valor.

Parar assim que receber uma mensagem.



Um Exemplo Ilustrativo

A cada segundo, ler um sensor e transmitir o valor.

Parar assim que receber uma mensagem.

```
par/or do
    await RadioAvail();
with
    loop do
        await 1s;
        int v = await AnalogRead();
        await RadioWrite(v);
    end
end
```

Um Exemplo Ilustrativo

A cada segundo, ler um sensor e transmitir o valor.

Parar assim que receber uma mensagem.

```
par/or do
    await RadioAvail();
with
    loop do
        await 1s;
        int v = await AnalogRead();
        await RadioWrite(v);
    end
end
```

Um Exemplo Ilustrativo

A cada segundo, ler um sensor e transmitir o valor.

Parar assim que receber uma mensagem.

```
par/or do
    await RadioAvail();
with
    loop do
        await 1s;
        int v = await AnalogRead();
        await RadioWrite(v);
    end
end
```

Initial Results

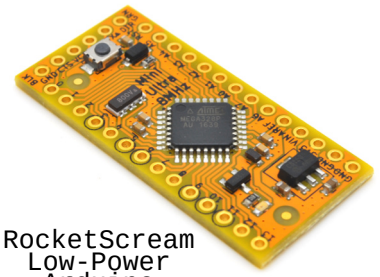
	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)

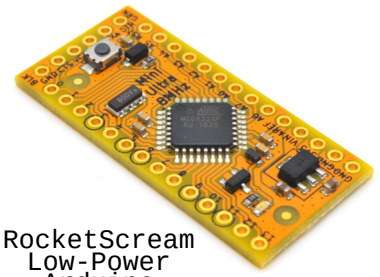


RocketScream
Low-Power
Arduino

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)

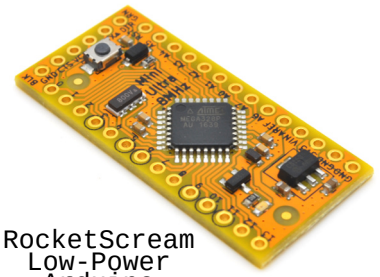


RocketScream
Low-Power
Arduino

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



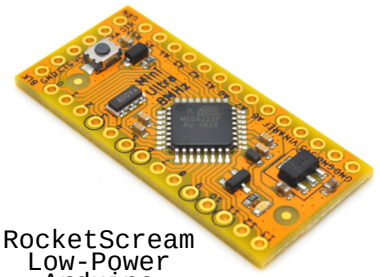
RocketScream
Low-Power
Arduino

```
await FOREVER;
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



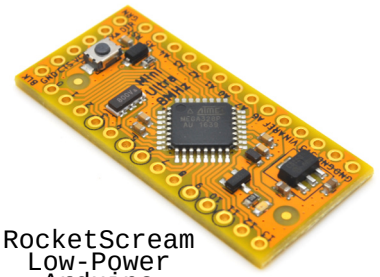
RocketScream
Low-Power
Arduino

```
await FOREVER;
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



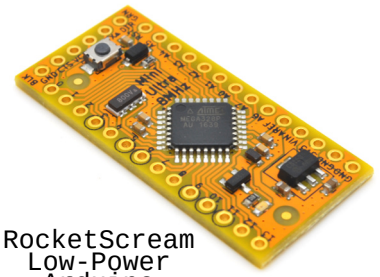
RocketScream
Low-Power
Arduino

```
await FOREVER;
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino

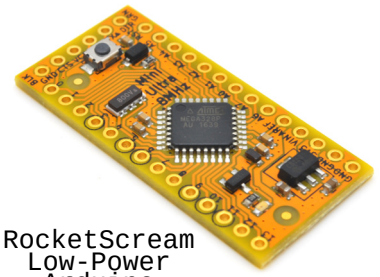
```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino

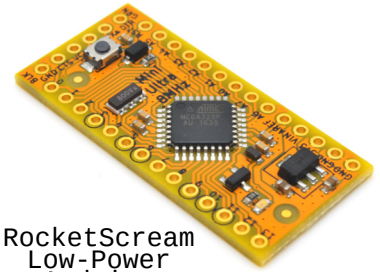
```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino

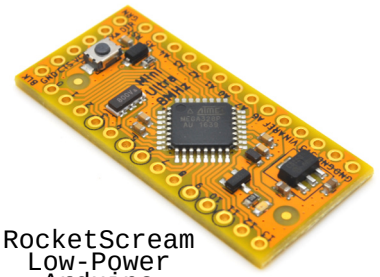
```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino

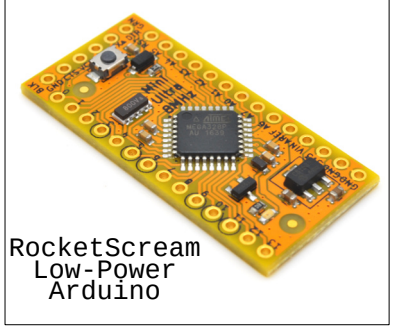
```
await FOREVER;
```

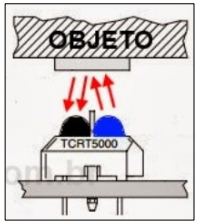
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)


RocketScream
Low-Power
Arduino



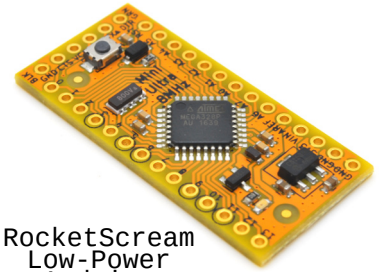

```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

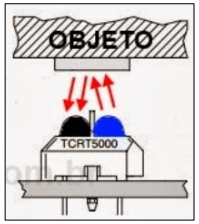
Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino




```
await FOREVER;
```

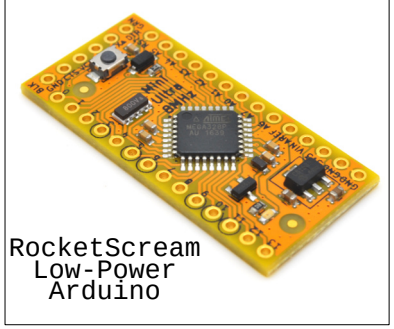
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

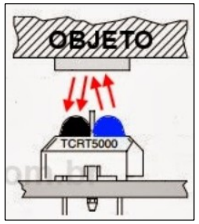
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)


RocketScream
Low-Power
Arduino




```
await FOREVER;
```

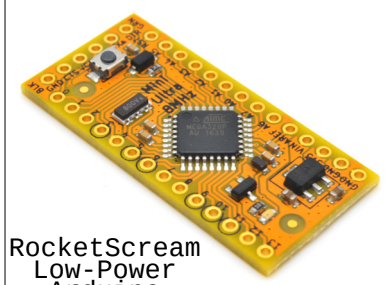
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

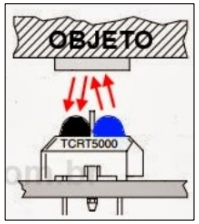
Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino




```
await FOREVER;
```

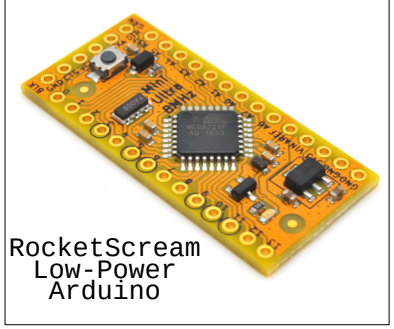
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

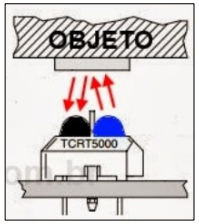
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)


RocketScream
Low-Power
Arduino




```
await FOREVER;
```

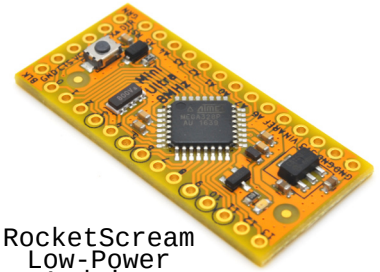
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

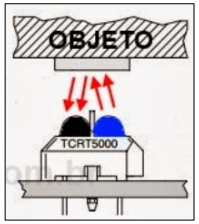
Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)



RocketScream
Low-Power
Arduino




```
await FOREVER;
```

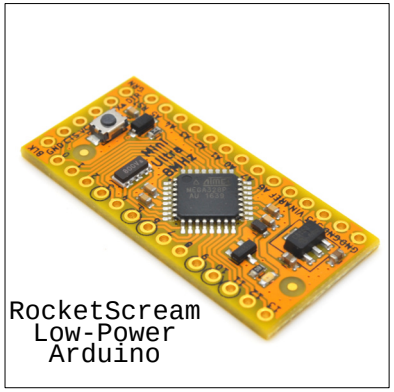
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

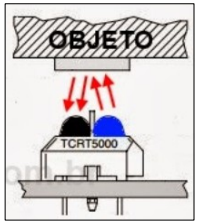
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)


RocketScream
Low-Power
Arduino



```
await FOREVER;
```

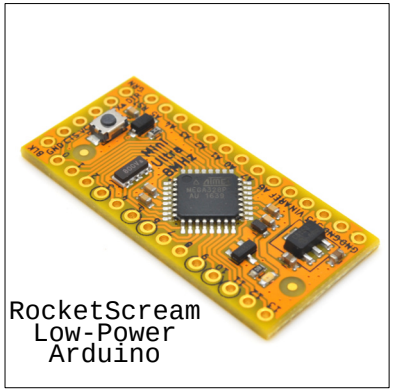
```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

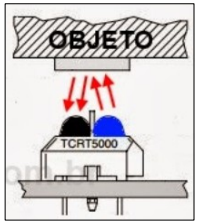
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)





```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

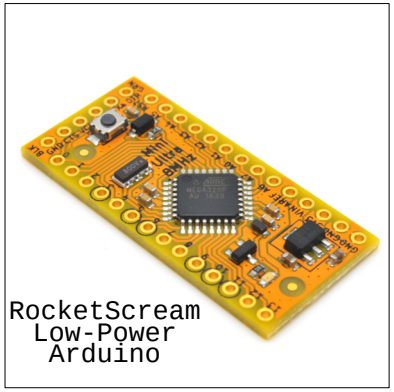
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

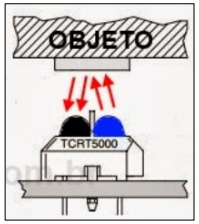
```
loop do
   await 1s;
   <...>
   await Nrf24l01_TX(...);
   <...>
   await Nrf24l01_RX(...);
   <...>
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)


RocketScream
Low-Power
Arduino



```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

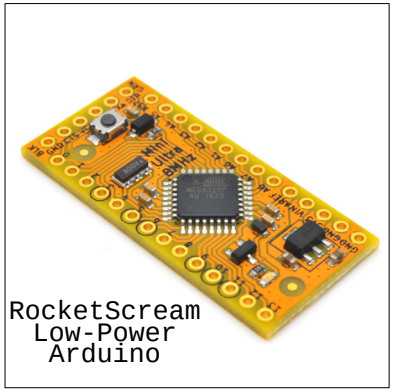
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

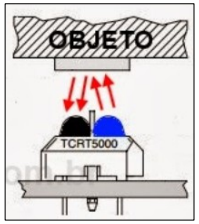
```
loop do
   await 1s;
   <...>
   await Nrf24l01_TX(...);
   <...>
   await Nrf24l01_RX(...);
   <...>
end
```

Initial Results

	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)


RocketScream
Low-Power
Arduino



```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

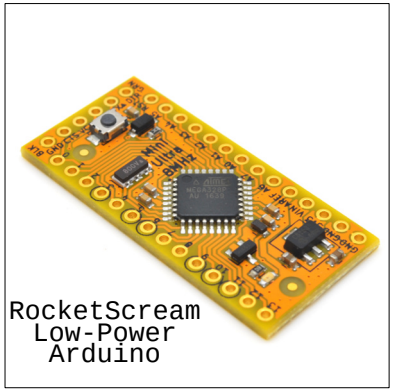
```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

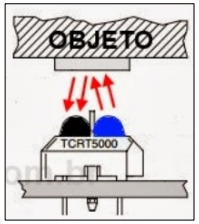
```
loop do
   await 1s;
   <...>
   await Nrf24l01_TX(...);
   <...>
   await Nrf24l01_RX(...);
   <...>
end
```

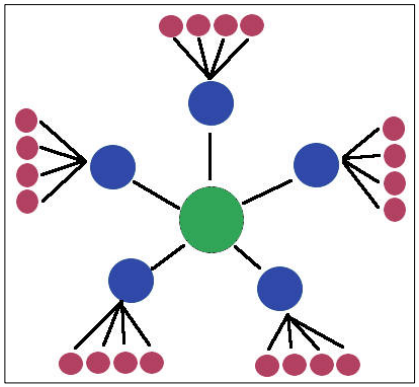
Initial Results

	Arduino	Céu M1	Céu M2	OBS
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(Consumption in mA)







```
await FOREVER;
```

```
loop do
   emit PIN(13,high);
   await 1s;
   emit PIN(13,low);
   await 1s;
end
```

```
emit PIN(13, _digitalRead(2));
loop do
   var bool v = await Pin(2);
   emit PIN(13, v);
end
```

```
loop do
   await 1s;
   <...>
   await Nrf24l01_TX(...);
   <...>
   await Nrf24l01_RX(...);
   <...>
end
```

General Approach

(standby, constrained, programming language, transparent)

```
par/or do
  await RadioAvail();
with
  loop do
    await 1s;
    int v = await AnalogRead();
    await RadioWrite(v);
  end
end
```

General Approach
(standby, constrained, programming language, transparent)

- Enforce idle states of execution

```
par/or do
  await RadioAvail();
with
  loop do
    await 1s;
    int v = await AnalogRead();
    await RadioWrite(v);
  end
end
```

General Approach
(standby, constrained, programming language, transparent)

- Enforce idle states of execution
  - Céu enforces a reactive model of execution
- Infer deepest sleeping mode

```
par/or do
  await RadioAvail();
with
  loop do
    await 1s;
    int v = await AnalogRead();
    await RadioWrite(v);
  end
end
```

General Approach

(standby, constrained, programming language, transparent)

- Enforce idle states of execution
  - Céu enforces a reactive model of execution
- Infer deepest sleeping mode

```
par/or do
  await RadioAvail();
with
  loop do
    await 1s;
    int v = await AnalogRead();
    await RadioWrite(v);
  end
end
```

General Approach

(standby, constrained, programming language, transparent)

- Enforce idle states of execution
  - Céu enforces a reactive model of execution
- Infer deepest sleeping mode
  - Céu has a semantics amenable to analysis
- Put device to sleep

```
par/or do
  await RadioAvail();
with
  loop do
    await 1s;
    int v = await AnalogRead();
    await RadioWrite(v);
  end
end
```

General Approach

(standby, constrained, programming language, transparent)

- Enforce idle states of execution
  - Céu enforces a reactive model of execution
- Infer deepest sleeping mode
  - Céu has a semantics amenable to analysis
- Put device to sleep

```
par/or do
   await RadioAvail();
with
   loop do
      await 1s;
      int v = await AnalogRead();
      await RadioWrite(v);
   end
end
```

General Approach

(standby, constrained, programming language, transparent)

- Enforce idle states of execution
  - Céu enforces a reactive model of execution
- Infer deepest sleeping mode
  - Céu has a semantics amenable to analysis
- Put device to sleep
  - Céu has an energy-aware runtime
- Only awake from interrupts

```
par/or do
   await RadioAvail();
with
   loop do
      await 1s;
      int v = await AnalogRead();
      await RadioWrite(v);
   end
end
```

General Approach

(standby, constrained, programming language, transparent)

- Enforce idle states of execution
  - Céu enforces a reactive model of execution
- Infer deepest sleeping mode
  - Céu has a semantics amenable to analysis
- Put device to sleep
  - Céu has an energy-aware runtime
- Only awake from interrupts
  - Céu provides interrupt service routines (ISRs)

```
par/or do
   await RadioAvail();
with
   loop do
      await 1s;
      int v = await AnalogRead();
      await RadioWrite(v);
   end
end
```

Eficiência Energética para Software IoT: Uma abordagem no Nível de Linguagem de Programação

Francisco Sant'Anna

francisco@ime.uerj.br

www.ceu-lang.org