Um Microkernel para Arduino

Francisco Sant'Anna

www.ceu-lang.org

LabLua – PUC-Rio
www.lua.inf.puc-rio.br

Sobre nós...

- Linhas de pesquisa:
  - Linguagens de Programação
  - Sistemas Concorrentes e Distribuídos

- Linguagem de Programação Reativa “Céu”
  - Plataformas Embarcadas
  - Redes de Sensores Sem Fio

Microkernel (?!)

- Sistema Operacional
  - Gerencia recursos de hardware e provê serviços comuns:
    - Escalonamento de processos (*Scheduler*)
    - Comunicação entre processos (*IPC*)
    - Drivers de I/O
    - Sistema de Arquivos
    - Protocolos de Rede, etc.

- Arquitetura de Microkernel (vs Monolítica)
  - Scheduler + IPC (~15 system calls // Linux has ~200)

Por quê? Como?

Objetivos

- Multi-aplicações (protocolos)
- Reprogramação remota
- Programação Reativa
- Segurança, Determinismo

Desafios

- 16MHz, (32+4)Kb
- Sem MMU
- Distribuição Espacial

Proposta

- Co-design Linguagem+SO
- Comunicação por eventos
- Escalonamento cooperativo
- Sem memória estática
- Pilha única
- Fila de eventos única

Céu: Hello World!

```
var int dt = 1000;

_pinMode(13, _OUTPUT);

loop do
  _digitalWrite(13, _HIGH);
  await (dt)ms;
  _digitalWrite(13, _LOW);
  await (dt)ms;
end
```

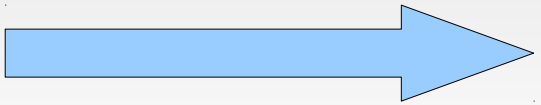
```
var int dt = 1000;

_pinMode(13, _OUTPUT);

par do
   loop do
      _digitalWrite(13, _HIGH);
      await (dt)ms;
      _digitalWrite(13, _LOW);
      await (dt)ms;
   end
with
   loop do
      var int light = _analogRead(0);
      dt = _map(light, 0,1023, 100,1000);
      await 200ms;
   end
end
```

- Chamadas ao Arduino prefixadas com "_" (underscore).
- Comando **await** suspende a linha de execução até o evento acontecer.
- Múliplas linhas de execução com composições **par**, **par/or**, **par/and**.

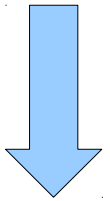
Não requer sistema operacional!!!

Céu “standalone”

```
// light.ceu
var int dt = 1000;
_pinMode(13, _OUTPUT);
par do
   loop do
      _digitalWrite(13, _HIGH);
      await (dt)ms;
      _digitalWrite(13, _LOW);
      await (dt)ms;
   end
with
   loop do
      var int light = _analogRead(0);
      dt = _map(light, 0,1023, 100,1000);
      await 200ms;
   end
end
```

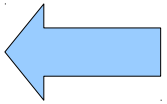


Compilador Céu

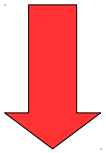


```
// light.ino
void setup () {
   <...>
}
void loop () {
   <...>
}
```

Compilador Arduino

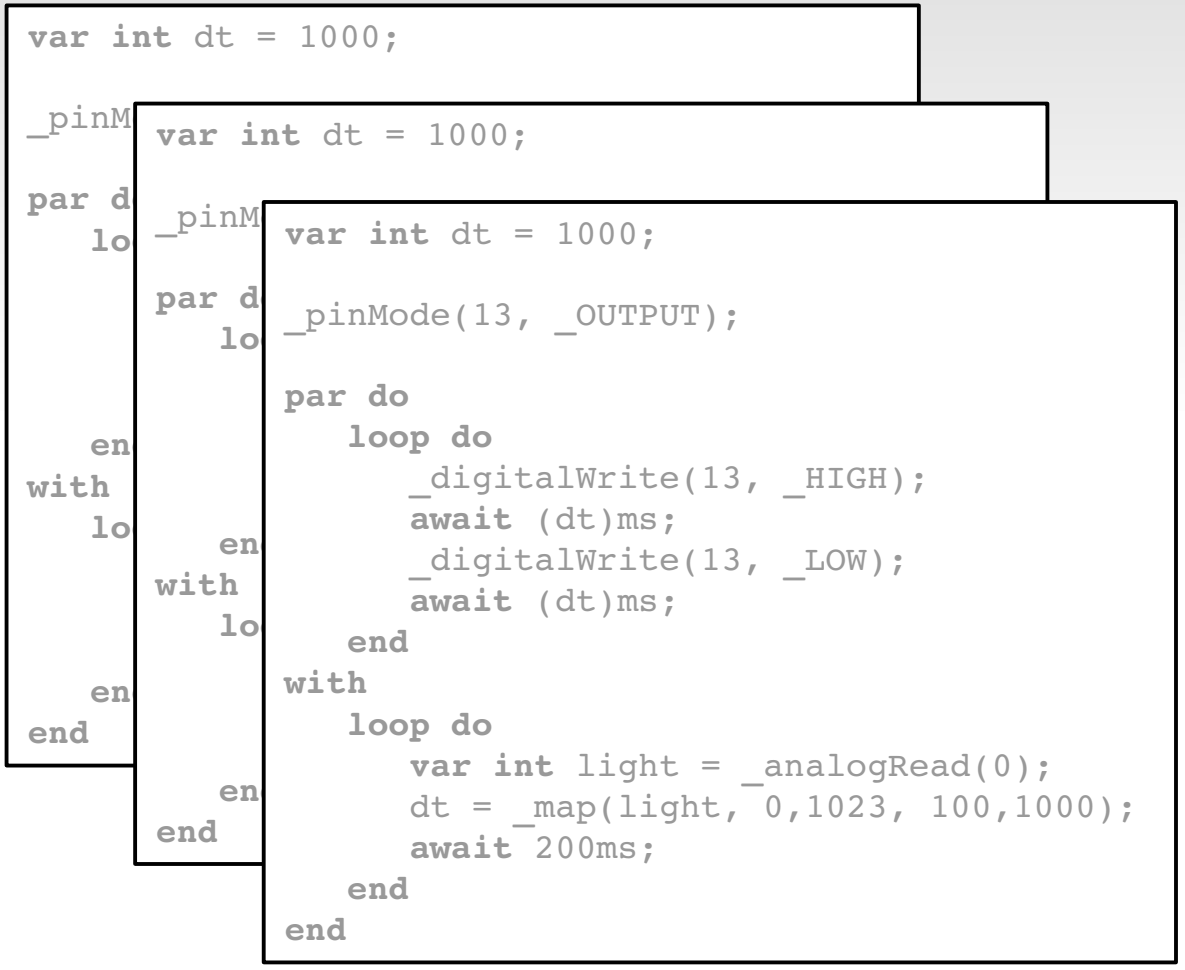


```
// light.hex
01010110101011010101
01010111010101010101
01011101110101101011
01010110101010110101
01010101101010101010
```



“make”

Múltiplas Aplicações



```
var int dt = 1000;

_pinMode(13, _OUTPUT);

par do
   loop do
      _digitalWrite(13, _HIGH);
      await (dt)ms;
      _digitalWrite(13, _LOW);
      await (dt)ms;
   end
with
   loop do
      var int light = _analogRead(0);
      dt = _map(light, 0,1023, 100,1000);
      await 200ms;
   end
end
```

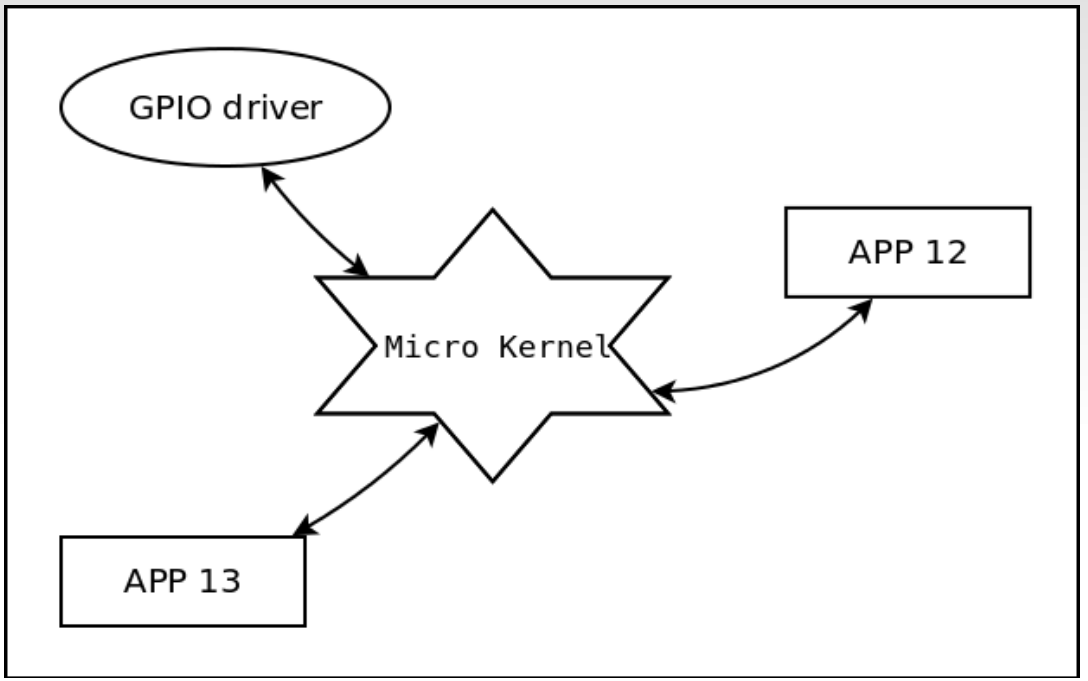
- Aplicações compiladas e carregadas em separado.
- Possivelmente por programadores diferentes.
- Código FLASH duplicado
- Recursos compartilhados
- **Um SO pode fazer o "meio de campo"**

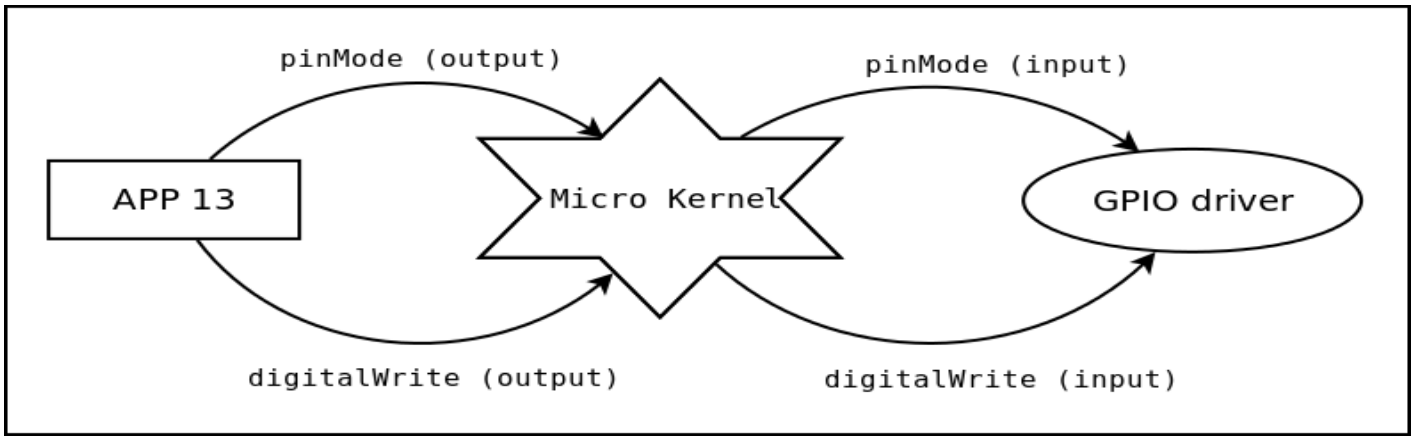
OS: Hello World!

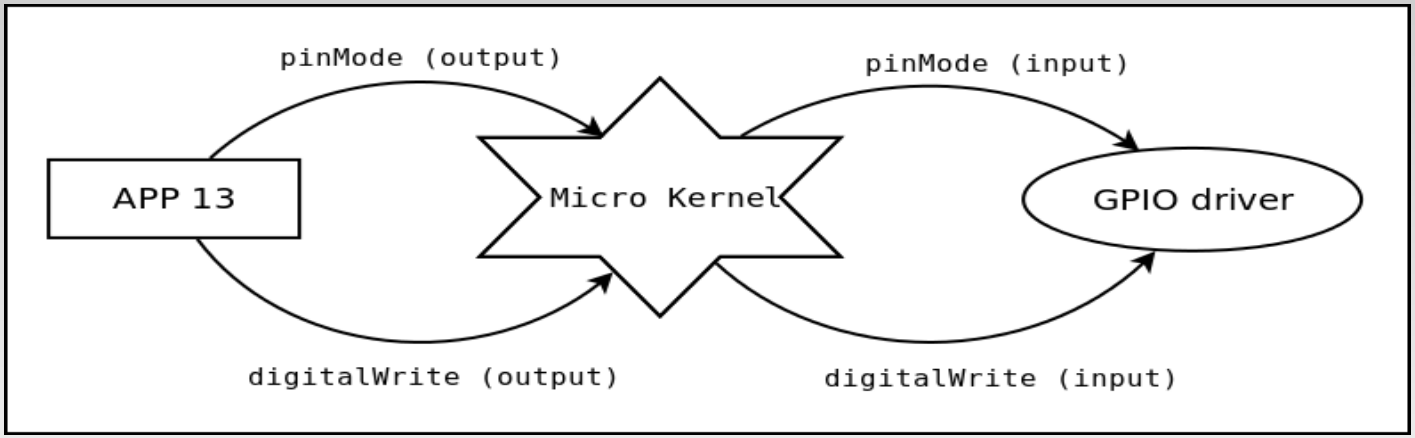
```
_pinMode(12, _OUTPUT);
loop do
end
```

```
_pinMode(13, _OUTPUT);
loop do
  _digitalWrite(13, _HIGH);
  await 1s;
  _digitalWrite(13, _LOW);
  await 1s;
end
```

```
// GPIO driver
_pinMode      ?
_digitalWrite ?
```







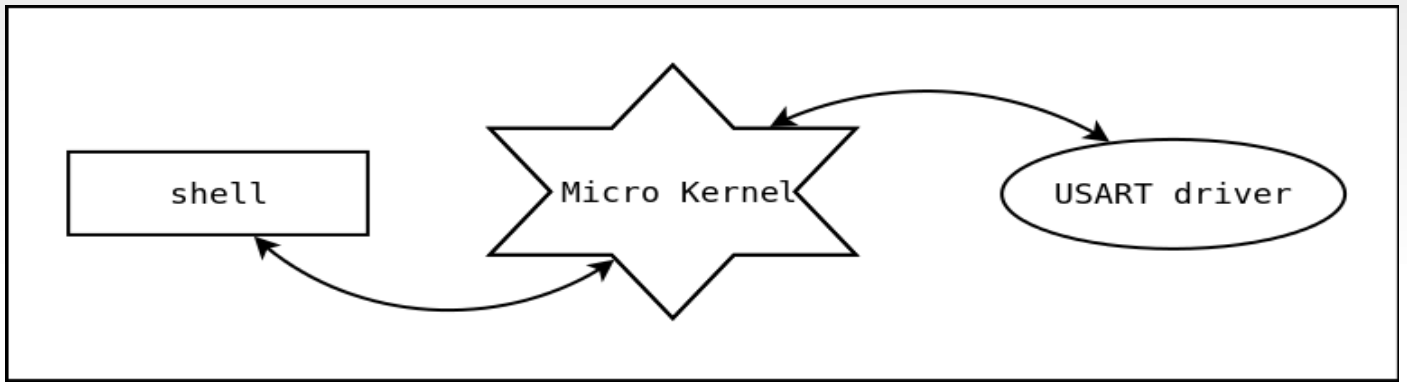

```
// APP 13
_pinMode(13, _OUTPUT);
loop do
  _digitalWrite(13, _HIGH);
  await 1s;
  _digitalWrite(13, _LOW);
  await 1s;
end
```

```
// APP 13
output (int pin, int mode)=>void PIN_MODE;
output (int pin, int val) =>void DIGITAL_WRITE;
call PIN_MODE => (13,OUTPUT);
loop do
   call DIGITAL_WRITE => (13,HIGH);
   await 1s;
   call DIGITAL_WRITE => (13,LOW);
   await 1s;
end
```

```
// GPIO driver
input (int pin, int mode)=>void PIN_MODE do
    _pinMode(pin, mode);
end
input (int pin, int val)=>void DIGITAL_WRITE do
    _digitalWrite(pin, val);
end
```

DEMO

- Image básica: *mk* + *serial* + *shell*



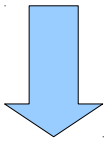
- Carga dinâmica: *gpio* + *app1* + *app2*

Com. entre Processos

Calls:

- execução imediata
- unicast

```
// app13.ceu
output (int pin, int mode)=>void PIN_MODE;
output (int pin, int val) =>void DIGITAL_WRITE;
call PIN_MODE => (13,OUTPUT);
loop do
   call DIGITAL_WRITE => (13,HIGH);
   await 1s;
   call DIGITAL_WRITE => (13,LOW);
   await 1s;
end
```



Comandos rápidos que não bloqueiam a CPU.

```
// gpio.ceu
input (int pin, int mode)=>void PIN_MODE do
    _pinMode(pin, mode);
end
input (int pin, int val)=>void DIGITAL_WRITE do
    _digitalWrite(pin, val);
end
```

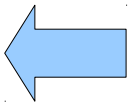
Com. entre Processos

Events:

- enfileirado / deferido
- multicast

Notificação aleatória / não requisitada.

```
// app13.ceu
<...>
input (int,int) DIGITAL_CHANGED;
var int dt = 1000;
par do
   <...>
with
   loop do
      var int pin;
      var int v;
      (pin, v) = await DIGITAL_CHANGED
                 until pin == 2;
      dt = MAX(100, dt-50);
   end
end
```



```
// gpio.ceu
output (int,int) DIGITAL_CHANGED;
<...>
every 50ms do
   loop i, 14 do
      if _chkMode(i, _INPUT) then
         var int v = _digitalRead(i);
         if _chkValue(i,v) then
            _chgValue(i,v);
            emit DIGITAL_CHANGED => (i,v);
         end
      end
   end
end
```

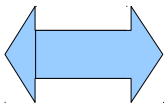
Com. entre Processos

Requests:

- enfileirado / deferido
- unicast

Notificação esperada / requisitada.

```
// app13.ceu
<...>
output/input (int)=>int ANALOG_READ;
var int dt = 1000;
par do
   <...>
with
   loop do
      var int v = (request ANALOG_READ=>0);
      dt = _map(v, 0,1023, 100,1000);
      await 200ms;
   end
end
```



```
// adc.ceu
input/output (int pin)=>int READ do
   _ADMUX  = (pin & 0x7);
   _ADCSRA = _ADCSRA | (1<<_ADSC);
   async do
      while (_ADCSRA & (1<<_ADSC));
   end
   var int low  = _ADCL;
   var int high = _ADCH;
   return (high<<8) | low;
end
```

“Takeaways”

- Céu: alternativa para o C/C++ do Arduino
  - Modelo reativo
  - Composições paralelas
  - Garantias de segurança (não discutidas hoje)
- Microkernel
  - Aplicações desenvolvidas em separado
  - Carga remota de código
  - Pensado em conjunto com Céu

Obrigado!

Francisco Sant'Anna


@fsantanna_puc


francisco.santanna@gmail.com

www.ceu-lang.org

LabLua – PUC-Rio

www.lua.inf.puc-rio.br