Where do Events Come From?

Reactive and Energy-Efficient Programming From the Ground Up

(In-Progress Paper)

www.ceu-lang.org

Francisco Sant'Anna

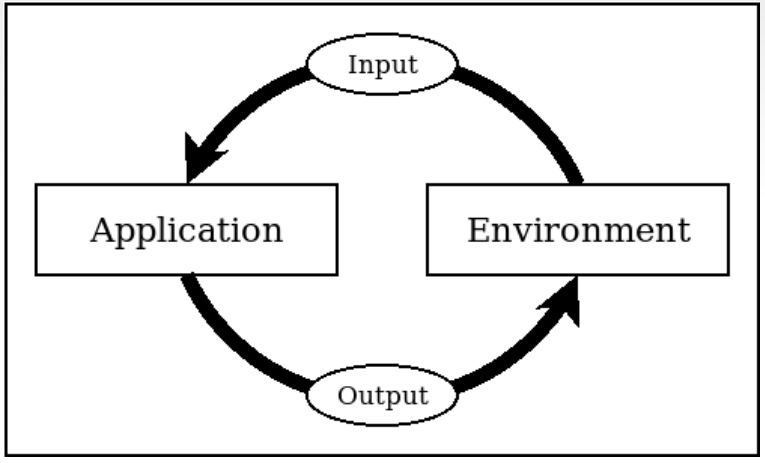
Rio de Janeiro State University

francisco@ime.uerj.br

@_fsantanna

Reactive and Event-Based Systems

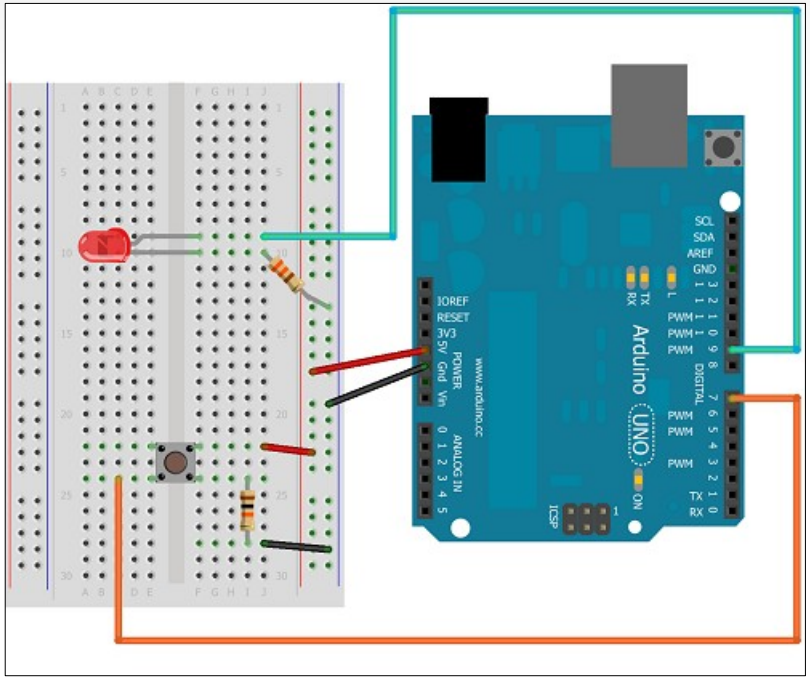
- Interact with sensors and actuators
- Represented as input & output events
- An *Environment* groups the I/O peripherals as a single entity
- Application and Environment are connected through an event loop



Céu - Arduino

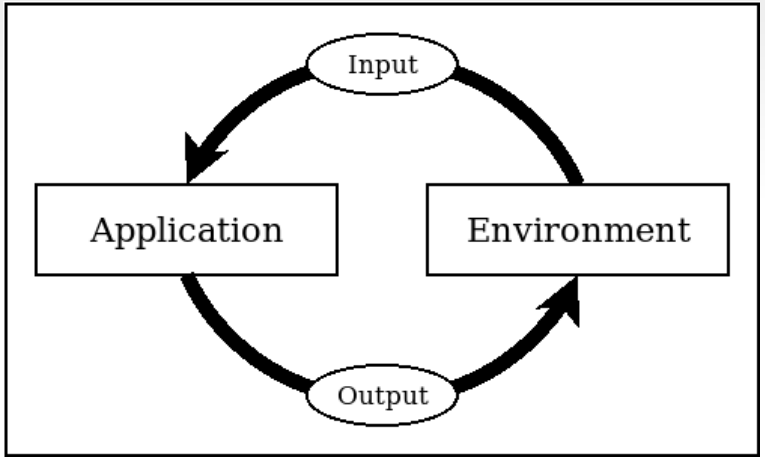
Blink the LED every second, stop on a button press.

```
input  high/low IN_07;
output high/low OUT_09;
```



The Environment

- Typically implemented in a host language (e.g., C)
- Controls the main event loop
- Invokes entry points into the reactive runtime (e.g. *callbacks*)
- Rigid component that evolves in separate from the application
- *Can we implement the Environment and Application together?*



Goal

- Take control of the whole event loop
  - From **input** generation, reaction, up to **output** effects
- New **asynchronous** interrupt handler primitive
  - In the context of the **synchronous** language Céu:
    - Prevent race conditions
    - Provide automatic standby for applications

Synchronous - Properties

```
input  high/low IN_07;
output high/low OUT_09;

par/or do
    await IN_07;
with
    loop do
        await 1s;
        emit OUT_09(high);
        await 1s;
        emit OUT_09(low);
    end
end
```

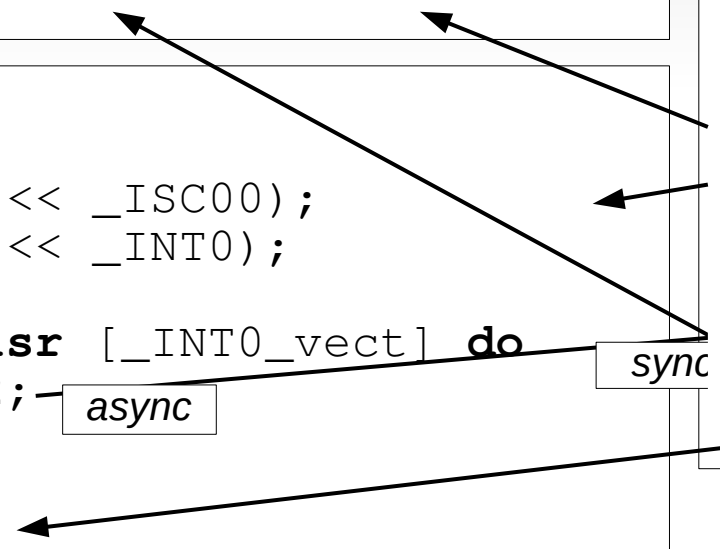
- Atomicity
  - Non-preemptive reactions
  - Environment must await
- Responsiveness
  - Loops must contain awaits
  - Application eventually yields

Asynchronous ISRs in Céu

```
// out.ceu

_pinMode(9, _OUTPUT);
output (high/low v) OUT_09 do
   _digitalWrite(9, v);
end
```

```
// in.ceu

_EICRA |= (1 << _ISC00);
_EIMSK |= (1 << _INT0);

spawn async/isr [_INT0_vect] do
   emit IN_02;
end
```

```
// app.ceu

input  none      IN_02;
output high/low OUT_09;
```



Preventing Race Conditions

```
// usart.ceu

input none USART_RX;
var[32] byte rx_buf;

spawn async/isr [_USART_RX_vect] do
    rx_buf = rx_buf .. [_UDR0];
    if $rx_buf == 1 then
        emit USART_RX;
    end
end
```

```
// app.ceu

#include “usart.ceu”

loop do
    await USART_RX;
    atomic do
        var int i;
        loop i in [0 -> $rx_buf[ do
            // uses rx_buf[i]
        end
        $rx_buf = 0;
    end
end
```

Standby Considerations

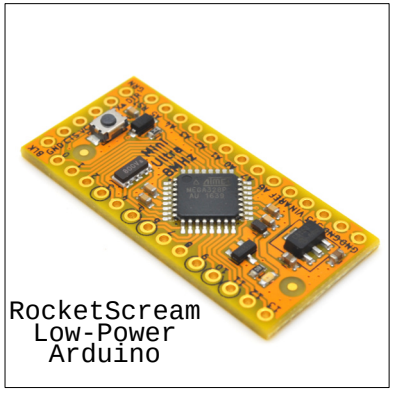
```
input  high/low IN_07;
output high/low OUT_09;

par/or do
    await IN_07;
with
    loop do
        await 1s;
        emit OUT_09(high);
        await 1s;
        emit OUT_09(low);
    end
end
```

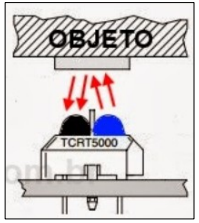
- Programs are always awaiting
- Only awakes from interrupts
- Automatic standby is possible

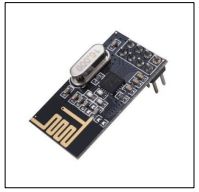
Initial Results

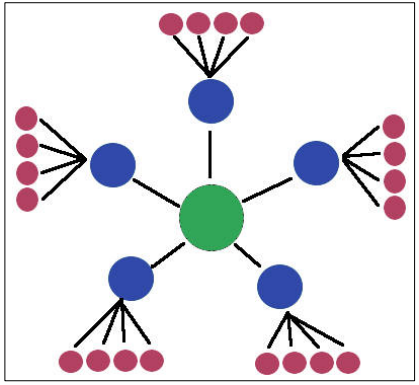
	Arduino	Céu		OBS
		M1	M2	
Empty	3.7	0.002		No activity.
Blink	6.0	3.1		Least efficient mode b/c of TIMER1.
Sensor	11.4	7.7		Most efficient mode b/c of INT2.
Radio	19.5	15.8	3.0	Alternates INT2 <-> TIMER1.
Protocol	19.6	15.9		Consumption dominated by the Radio.

(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
```

Where do Events Come From?

Reactive and Energy-Efficient Programming From the Ground Up

(In-Progress Paper)

www.ceu-lang.org

Francisco Sant'Anna

Rio de Janeiro State University

francisco@ime.uerj.br

@_fsantanna