Concurrent Event-driven Programming in occam-pi for the Arduino

Christian L. Jacobsen, Matthew Jadud, Omer Kilic, and Adam Sampson

University of Copenhagen
Allegheny College
University of Kent
University of Abertay Dundee

June 21st, 2011
Overview

1. Introduction
2. The Arduino
3. Implementation
4. Implementation
5. Interrupts
6. Case Study: A Room Usage Monitor
7. Conclusions and Future Work
8. Questions
Introduction
A very brief history occam

1. occam on the Transputer
2. occam on SPARC, Alpha, etc.
3. occam-$\pi$ on x86
4. occam-$\pi$ on the Transterpreter
   - PPC, x86, ARM, MSP430, H8, ...
   - Mac, Linux, Windows, ...
The Arduino

Hardware
The Arduino
A very brief overview

- Open hardware
- Based on the ATmega 328
  - 8-bit, 16MHz
  - 32KB flash, 2KB RAM, 1KB EEPROM
  - 6 ADC, 1 USART, SPI/I²C, 14 GPIO
The Arduino
A very brief overview

Figure: The Arduino, an open hardware platform.
The Arduino

A very brief overview

Figure: Concurrency.cc board
The Arduino

Goals in porting to the Arduino

- Provide challenging environments for exploring concurrent control.
- Provide authentic educational context for learning about concurrency.
- Increase project awareness by engaging a large community of artists and makers.
An IDE with a simplified hardware programming model and libraries to support the platform are a critical part of adoption and learning for newcomers.
The Arduino
Variants, community, businesses
A Program

How do we code for the Arduino?
boolean state[4] = {false, false, false, false};
unsigned long prev = 0;

void setup () {
    Serial.begin(9600);
    for (int i = 0; i < 4; i++) {
        pinMode(10+i, OUTPUT);
    }
}

void toggle (int pin) {
    boolean val = state[pin - 10];
    digitalWrite(pin, !val);
    state[pin - 10] = !val;
}
void loop () {
    unsigned long time = millis();
    
    if (time != prev) {
        if ((time % 200) == 0) { toggle(13); }
        if ((time % 300) == 0) { toggle(12); }
        if ((time % 400) == 0) { toggle(11); }
        if ((time % 500) == 0) { toggle(10); }
        prev = time;
    }
}
A Program
In occam

#INCLUDE "plumbing.module"

PROC main ()

    PAR
        blink (13, 500)
        blink (12, 400)
        blink (11, 300)
        blink (10, 200)

    :
Implementation

How do we run on the Arduino?
Implementation

... in one slide...

Concurrent Event-driven Programming in occam-pi for the Arduino

Interrupts

Interrupts
Interrupts

INITIAL INT vintr IS (-1):
  SEQ
    set.interrups (avr.pin, vintr)
  WHILE TRUE
    ...
    wait.for.interrupt (vintr, any)
    ...

Interrupts
Power Consumption

- The VM is not busy-waiting for external events
- It can tell if there is nothing to run (but there are processes waiting on timers or interrupts)
- The VM can then enter one of the AVR's power down modes automagically
Interrupts

A hardware interrupt is raised.
Interrupts

It is lifted into a “soft interrupt” in the VM.

Concurrent Event-driven Programming in occam-pi for the Arduino

Interrupts

A natural occam-pi rescheduling point is reached.
Interrupts

wait.for.interrupt is unblocked.
Monitoring Room Usage

Monitoring Room Usage
Monitoring Room Usage

Context

- 25 Environmental Science students.
- Goal: Build, deploy, and analyze results from a sensor.
- Time: 4 weeks to design, prototype, develop, and ship.
Monitoring Room Usage

Sensor

Figure: The Environmental Sensor
Monitoring Room Usage

Results

Question

How much energy is wasted by lighting in empty rooms?

Answer

Up to 47%.

Outcome

Changes to automation configuration and future building configuration.
Monitoring Room Usage
The role of interrupts

Motion detection.
A $7$ passive IR motion sensor used to indicate room occupancy.

Scheduled readings.
A $15$ real-time clock module to schedule periodic checks of light levels in the room.

Design pattern.
A data collection pipeline with two possible triggers.
Conclusions and Future Work

We have:

• occam-pi on a cheap popular piece of hardware
• interrupts exposed to the user
• high level plug-and-play interface (Plumbing)
• small case studies

And want to look more at:

• power consumption
• performance
• bigger cases
• plug-and-play programming
Q & A

Thanks for listening.