Synchronous Programming of Device Drivers for Global Resource Control in Embedded Operating Systems

Nicolas Berthier

Supervisors: Florence Maraninchi & Laurent Mounier

Synchrone Team

Synchron 2010

Synchronous Programming of Device Drivers for Global Resource Control in Wireless Sensor Network Operating Systems

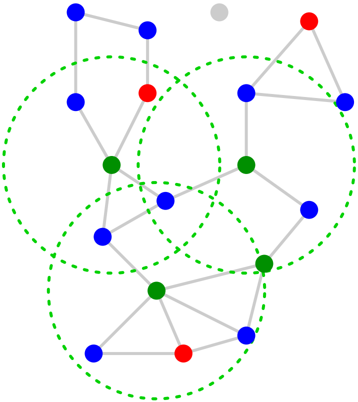
Nicolas Berthier

Supervisors: Florence Maraninchi & Laurent Mounier

Synchrone Team

Synchron 2010

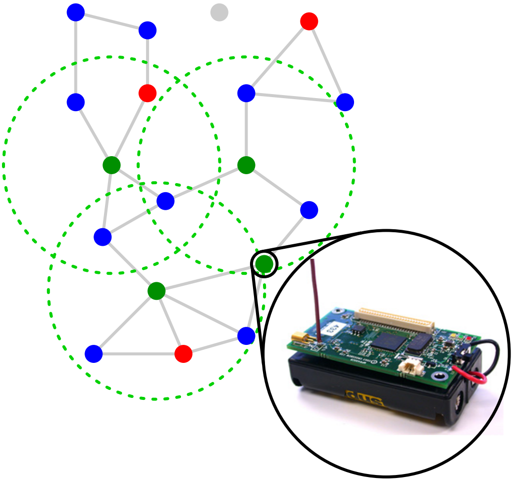
Context: Wireless Sensor Networks



Components

- μ-Controller (MCU)
- Radio Transceiver(s)
- Sensors
- Battery
- ...

Context: Wireless Sensor Networks



Components

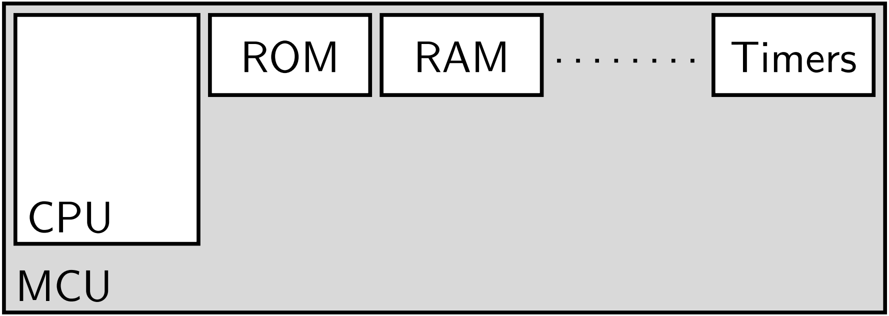
- μ-Controller (MCU)
- Radio Transceiver(s)
- Sensors
- Battery
- ...

Constraints, Problems

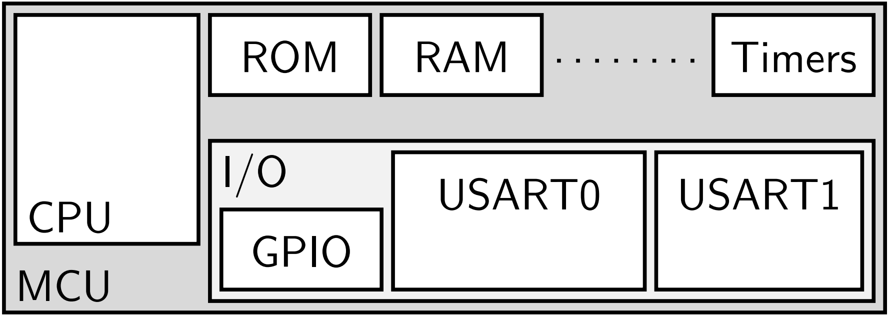
- Slow Computations
- Small Memory
- Battery-Awareness
- ...

Example WSN Hardware Platform

Wsn430

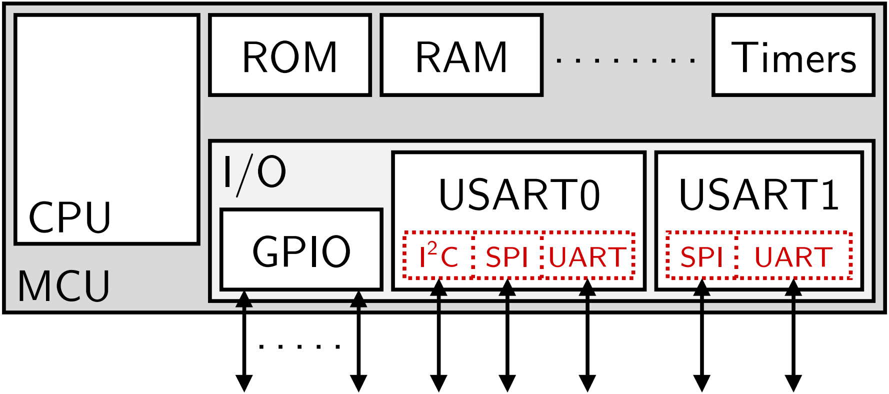


Example WSN Hardware Platform
Wsn430



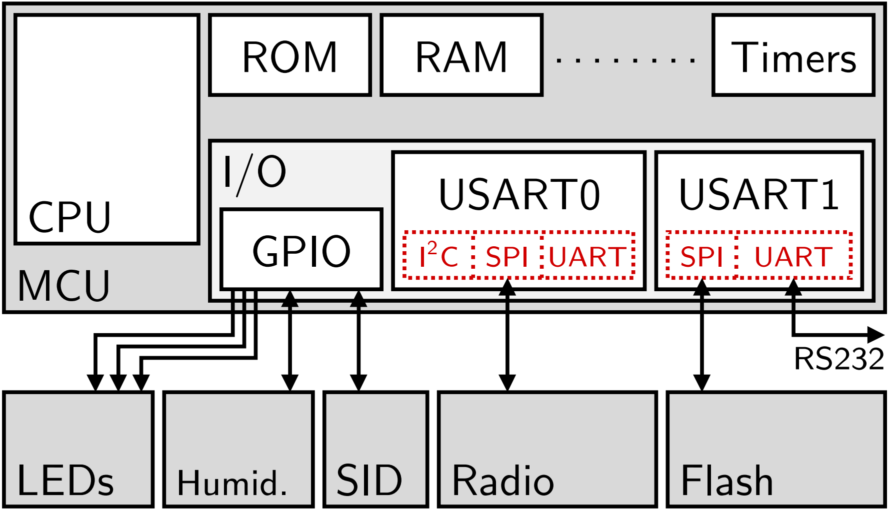
Example WSN Hardware Platform

Wsn430



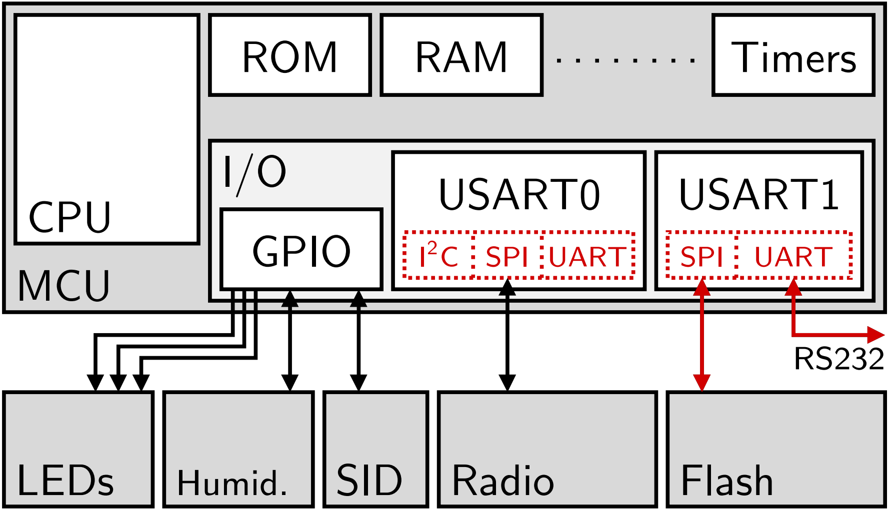
Example WSN Hardware Platform

Wsn430



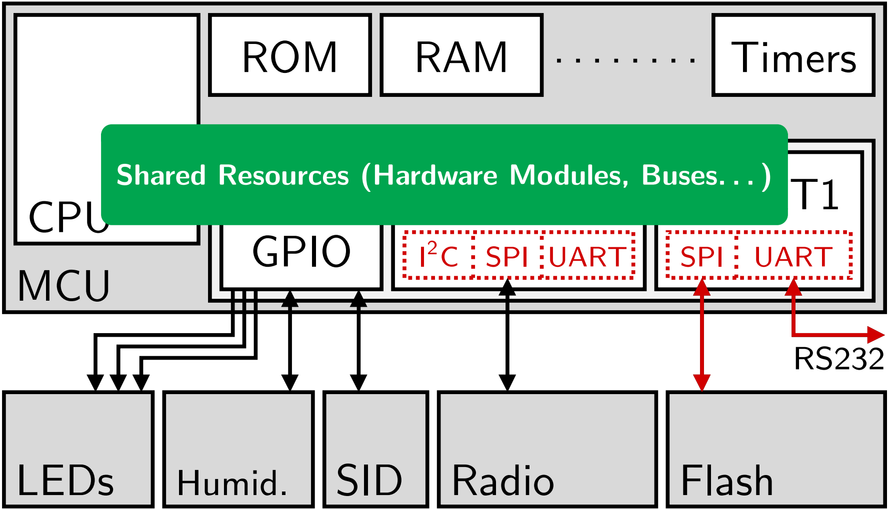
Example WSN Hardware Platform

Wsn430



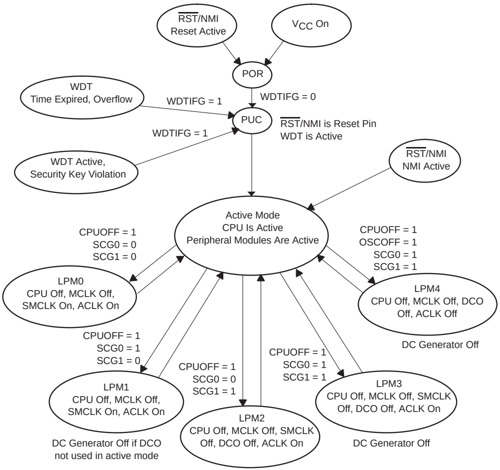
Example WSN Hardware Platform

Wsn430



Hardware Behavior: MCU Automaton

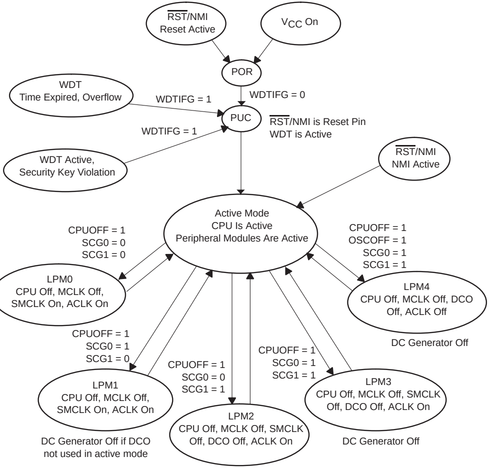
TI MSP430 Operating Modes



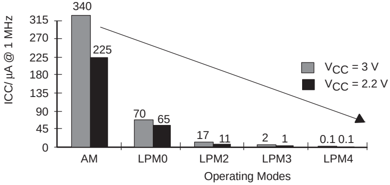
- Discrete States

Hardware Behavior: MCU Automaton

TI MSP430 Operating Modes

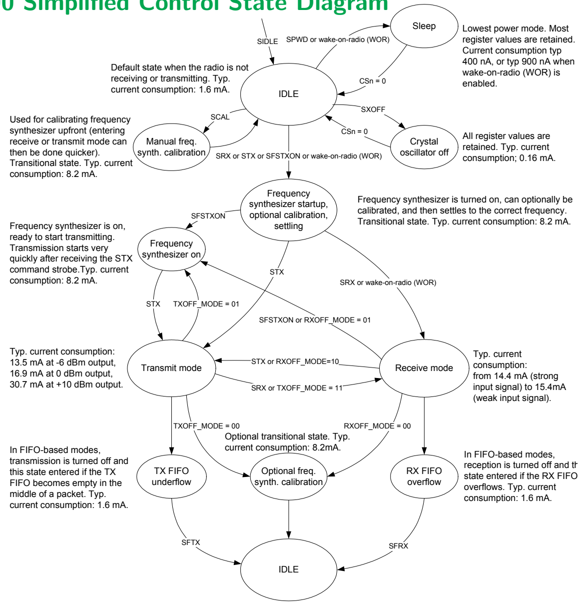


- Discrete States
- Power Consumption



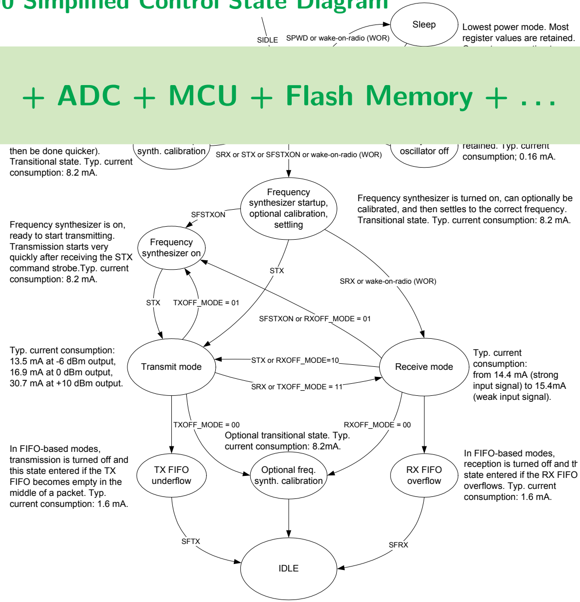
Hardware Behavior: Radio Automaton

Chipcon CC1100 Simplified Control State Diagram



Hardware Behavior: Radio Automaton

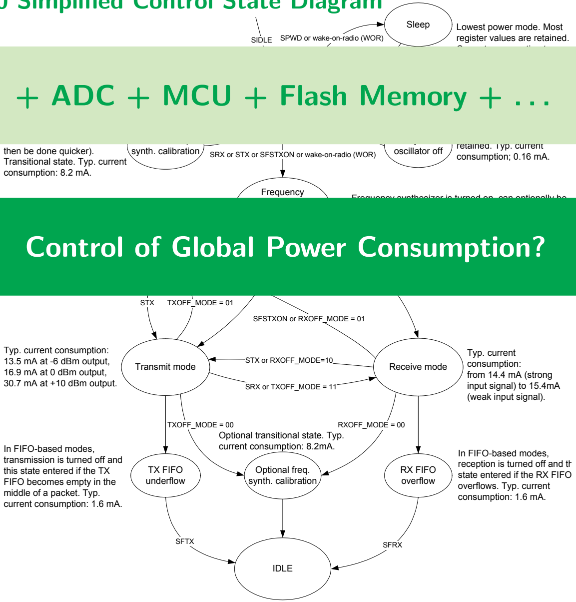
Chipcon CC1100 Simplified Control State Diagram



+ ADC + MCU + Flash Memory + . . .

Hardware Behavior: Radio Automaton

Chipcon CC1100 Simplified Control State Diagram



+ ADC + MCU + Flash Memory + ...

Control of Global Power Consumption?

Programming WSNs: Usual Practice

Applications

Operating System Support / Abstractions

- Multitasking
- System Services
- Hardware Device Drivers

Programming WSNs: Usual Practice

Applications

Operating System Support / Abstractions

- Multitasking
- System Services (Network Stack, File Systems. . .)
- Hardware Device Drivers

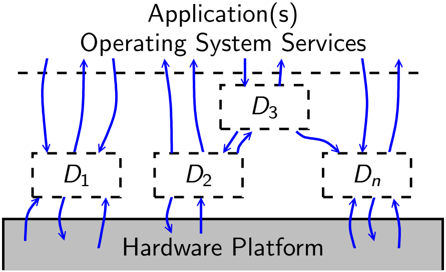
Programming WSNs: Usual Practice

Applications

Operating System Support / Abstractions

- Multitasking
- System Services (Network Stack, File Systems. . .)
- Hardware Device Drivers

Operating Systems Programming for WSNs



- Device Drivers designed *Locally*

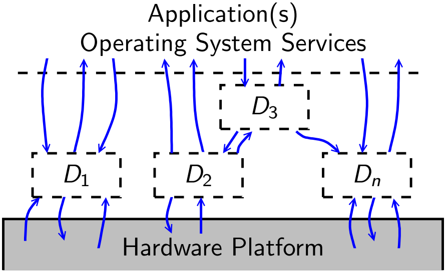
Programming WSNs: Usual Practice

Applications

Operating System Support / Abstractions

- Multitasking
- System Services (Network Stack, File Systems. . .)
- Hardware Device Drivers

Operating Systems Programming for WSNs



- Device Drivers designed *Locally*
- *Ad hoc* Solutions for Resource Management & Power-Awareness

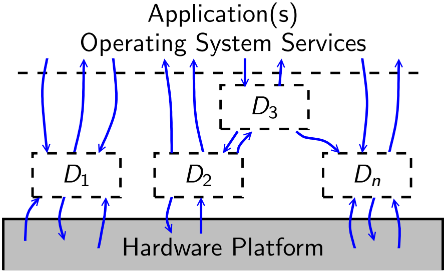
Programming WSNs: Usual Practice

Applications

Operating System Support / Abstractions

- Multitasking
- System Services (Network Stack, File Systems. . .)
- Hardware Device Drivers

Operating Systems Programming for WSNs



- Device Drivers designed *Locally*
- *Ad hoc* Solutions for Resource Management & Power-Awareness

  ⇒ **Decentralized Knowledge!**

Problems

Recap

Problems

Recap

- Shared Resources

Problems

Recap

- Shared Resources
- Power Management

Problems

Recap

- Shared Resources
- Power Management

➨ **Need for Global Control!**

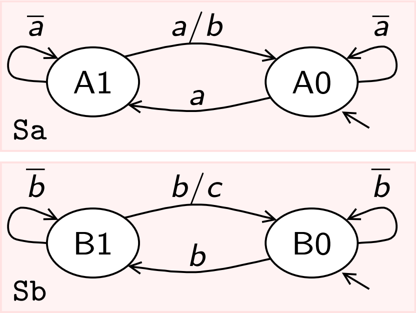
Outline

- **Preliminary Remarks**
- **Proposal**
- **Implementation**
- **Summary**

Outline

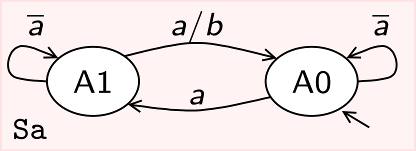
Communicating Boolean Mealy Machines

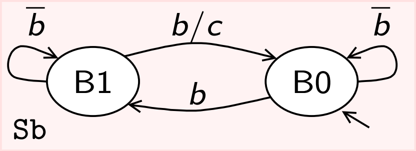
Synchronous Product

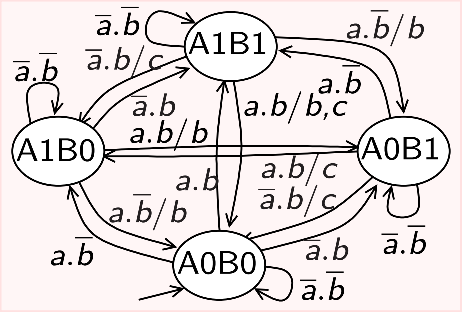


Communicating Boolean Mealy Machines

Synchronous Product

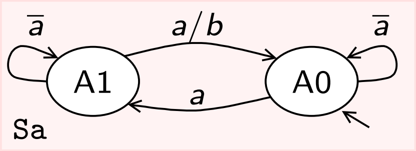


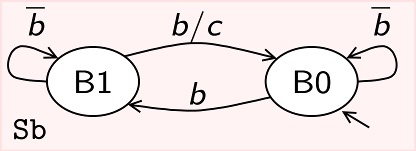


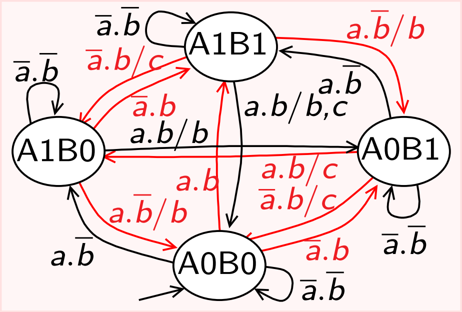


Communicating Boolean Mealy Machines

Synchronous Product

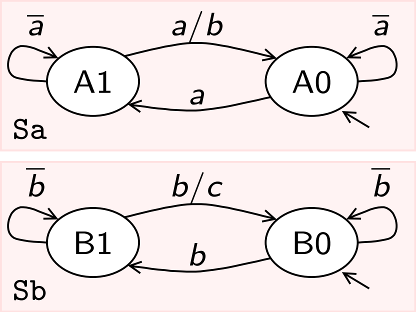


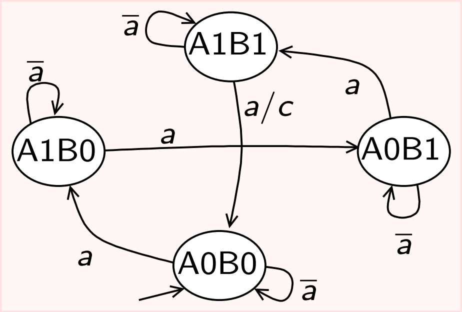




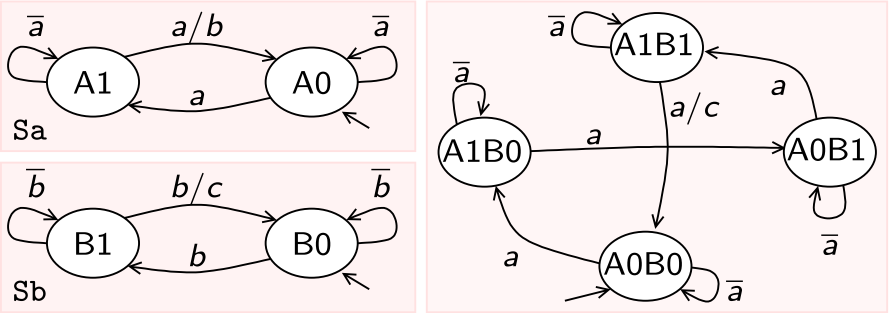
Communicating Boolean Mealy Machines

Synchronous Product





Communicating Boolean Mealy Machines




```
node SE (a: bool) returns (c: bool);
var inB0, inB1, inA0, inA1, m_A0, m_B0, A0, B0, b: bool;
let
  inA1 = not m_A0;      inA0 = m_A0;      b = a and inA1;
  inB1 = not m_B0;      inB0 = m_B0;      c = b and inB1;
  A0 = not a and inA0 or a and inA1; m_A0 = true -> pre A0;
  B0 = not b and inB0 or b and inB1; m_B0 = true -> pre B0;
tel;
```

Communicating Boolean Mealy Machines

Reactive Kernel

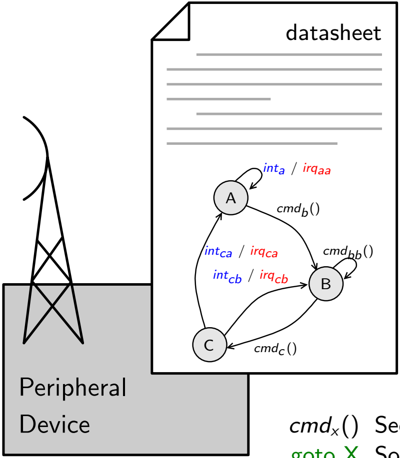
```
bool M1, M2, INIT;                // state variables
void init () { INIT = 1; } // initialization
```

Communicating Boolean Mealy Machines

Reactive Kernel

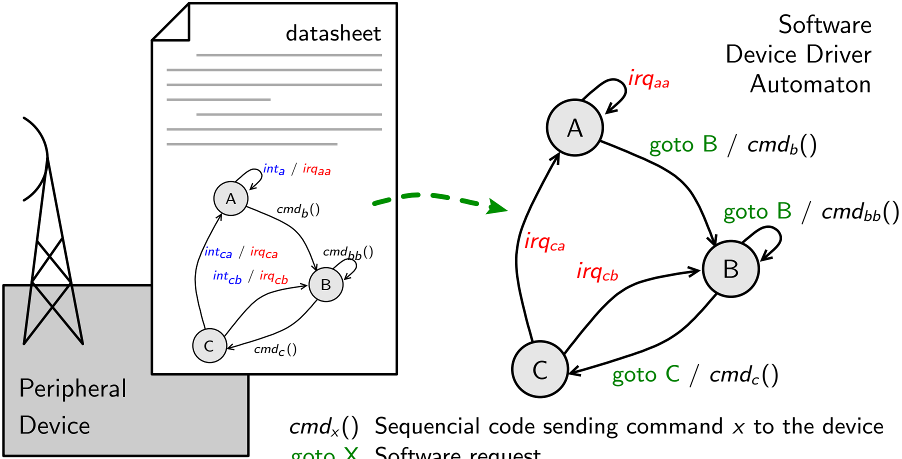
```
bool M1, M2, INIT;                // state variables
void init () { INIT = 1; } // initialization

void run_step (bool a) {
  bool L1, L2, L3, L4, L5, L6;

  L2 = INIT | M1;     L5 = INIT | M2;
  L4 = ~L5 &  a;      L1 = ~L2 &  L4;
  L6 =  L5 & ~a;      L3 =  L2 & ~L4;

  main_O_c (L1);
  M1 = L3 | L1;       M2 = L6 | L4;
  INIT = 0;
}
```

From Automata to Device Drivers



$cmd_x()$	Sequencial code sending command x to the device
goto X	Software request
irq_i	Hardware request / signal
int_j	Internal device event (*e.g.* end of transmission, etc.)

From Automata to Device Drivers



$cmd_x()$	Sequencial code sending command x to the device
goto X	Software request
irq_i	Hardware request / signal
int_j	Internal device event (*e.g.* end of transmission, etc.)

Outline

Principles of the Solution

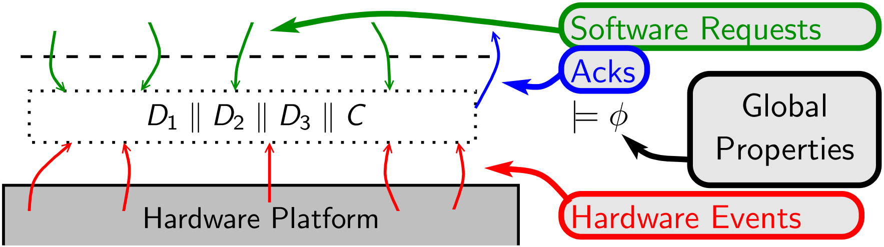
(Para-)Virtualization Concept

- Interception and Control of Software Operations
- *Global* Resource Control $\Rightarrow$ Centralized Knowledge
- May Forbid (or Enforce) Operations

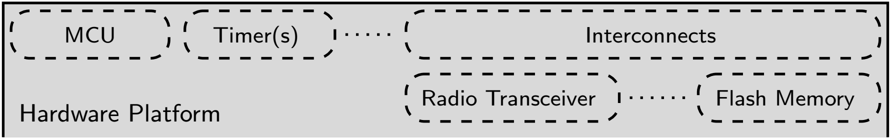
Principles of the Solution

(Para-)Virtualization Concept

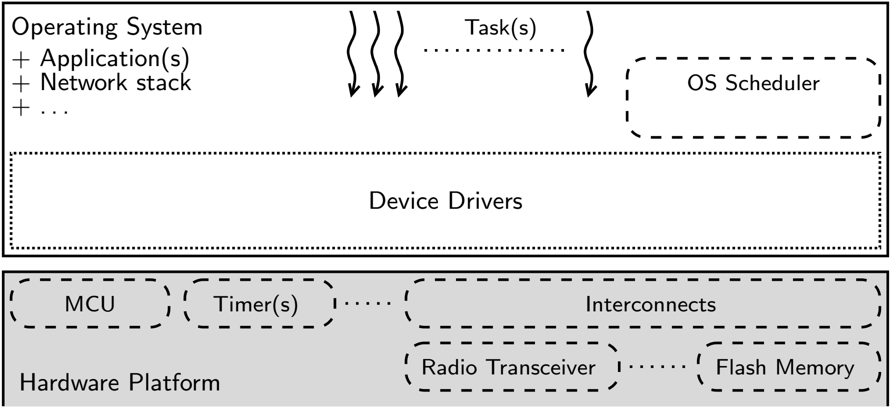
- Interception and Control of Software Operations
- *Global* Resource Control $\Rightarrow$ Centralized Knowledge
- May Forbid (or Enforce) Operations



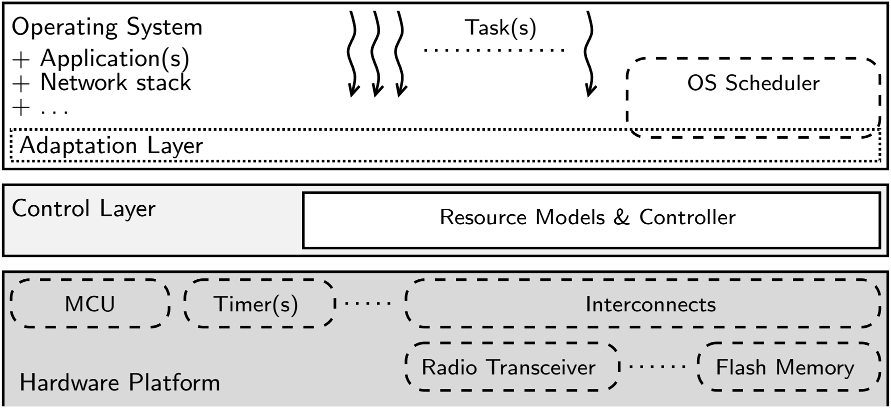
Overview


MCU
Timer(s)
Interconnects
Radio Transceiver
Flash Memory
Hardware Platform

Overview


Operating System
+ Application(s)
+ Network stack
+ . . .
Task(s)
OS Scheduler
Device Drivers
MCU
Timer(s)
Interconnects
Radio Transceiver
Flash Memory
Hardware Platform

Overview


Operating System
+ Application(s)
+ Network stack
+ . . .
Task(s)
OS Scheduler
Adaptation Layer
Control Layer
Resource Models & Controller
MCU
Timer(s)
Interconnects
Radio Transceiver
Flash Memory
Hardware Platform

Structure: Adaptation Layer

Modified part of the Operating System

- Simplified Device Drivers

Structure: Adaptation Layer

Modified part of the Operating System

- Simplified Device Drivers

Interacts with the Control Layer

- Emitting *software requests* to the Control Layer

Structure: Adaptation Layer

Modified part of the Operating System

- Simplified Device Drivers

Interacts with the Control Layer

- Emitting *software requests* to the Control Layer

- Receiving *output events* from the Control Layer
  - Notifications (Hardware Events, Acknowledgments)

Structure: Adaptation Layer

Modified part of the Operating System

- Simplified Device Drivers

Interacts with the Control Layer

- Emitting *software requests* to the Control Layer
  - Using on_sw()
- Receiving *output events* from the Control Layer
  - Notifications (Hardware Events, Acknowledgments)

```
turn_adc_on ()
  if (on_sw (adc_on) = ack_a)
    return success;
  timer_wait (some time);    // Consider we can
  turn_adc_on ();            // try again later
```

Structure: Adaptation Layer

Modified part of the Operating System

- Simplified Device Drivers

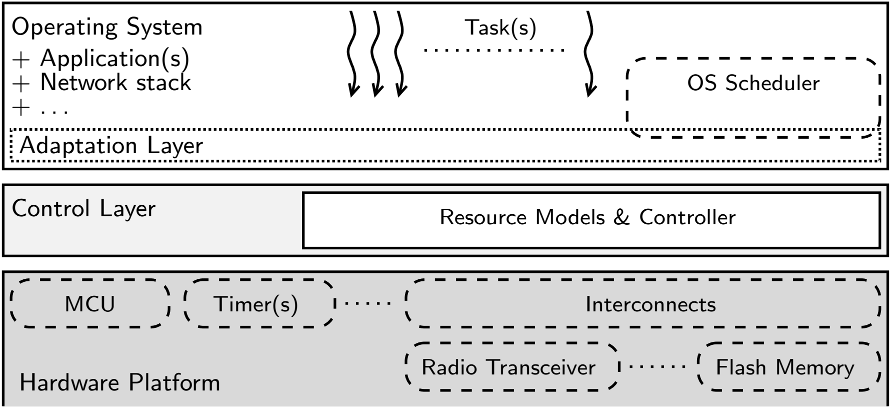
Interacts with the Control Layer

- Emitting *software requests* to the Control Layer
  - Using on_sw()
- Receiving *output events* from the Control Layer
  - Notifications (Hardware Events, Acknowledgments)

```
turn_adc_on ()
  if (on_sw (adc_on) = ack_a)
    return success;
  timer_wait (some time);    // Consider we can
  turn_adc_on ();            // try again later
```

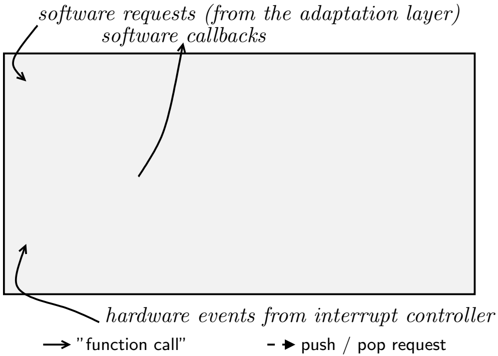
  - Callbacks (Virtual IRQs)

Overview


Operating System
+ Application(s)
+ Network stack
+ . . .
Task(s)
OS Scheduler
Adaptation Layer
Control Layer
Resource Models & Controller
MCU
Timer(s)
Interconnects
Radio Transceiver
Flash Memory
Hardware Platform

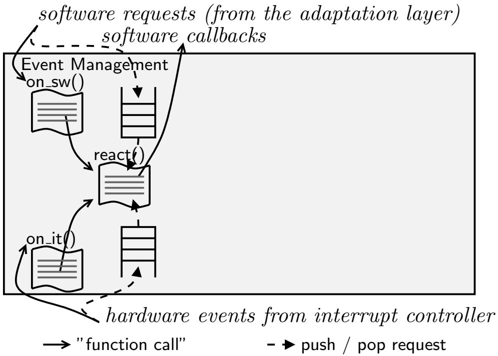
Structure: Control Layer

- Receives
  - *software requests*
  - *hardware requests* (IRQs)
- Emits *notifications*
- Manages the Peripheral Devices



Structure: Control Layer

- Receives
  - *software requests*
  - *hardware requests* (IRQs)
- Emits *notifications*
- Manages the Peripheral Devices

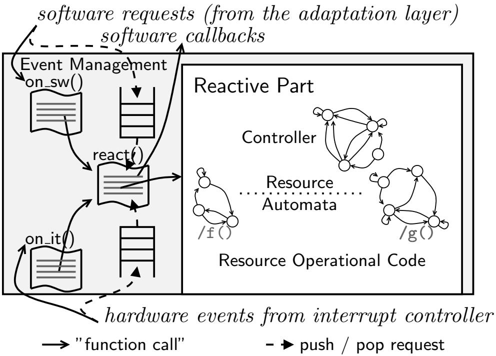


Event Management Part

- Handle *request queues*

Structure: Control Layer

- Receives
  - *software requests*
  - *hardware requests* (IRQs)
- Emits *notifications*
- Manages the Peripheral Devices



Event Management Part

- Handle *request queues*
- Executes the Reactive Part

Reactive Part

- Device Drivers Machines
  - *Reactive Kernel*
- Resource Operational Code

Principles of the Solution

(Para-)Virtualization Concept

- Interception and Control of Software Operations
- *Global* Resource Control $\Rightarrow$ Centralized Knowledge
- May Forbid (or Enforce) Operations

Principles of the Solution *(cont'd)*

(Para-)Virtualization Concept

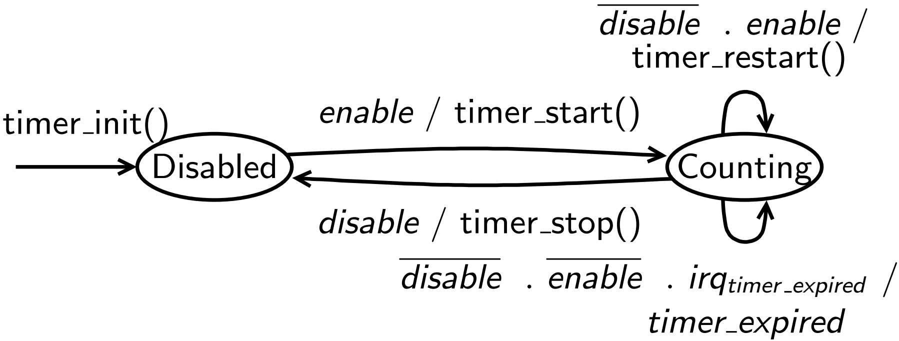
- Interception and Control of Software Operations
- *Global* Resource Control $\Rightarrow$ Centralized Knowledge
- May Forbid (or Enforce) Operations

Key Elements (Boolean Mealy Machines)

- Resource Automata
  - Inputs: Software Requests...
  - Outputs: Low-Level Code, Notifications...
- Controller

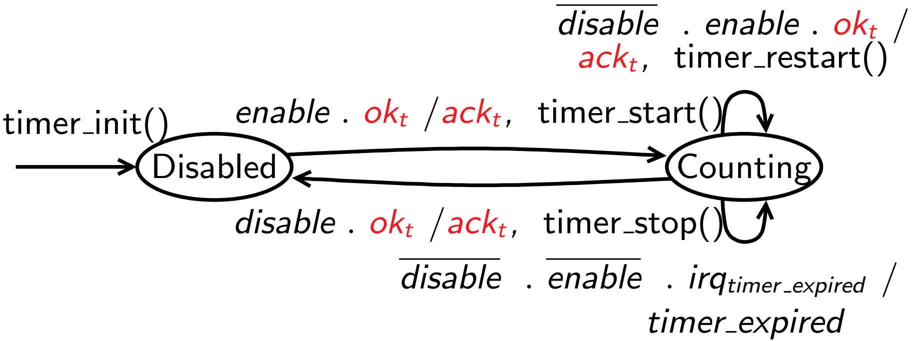
Example of Uncontrollable Automaton

Timer



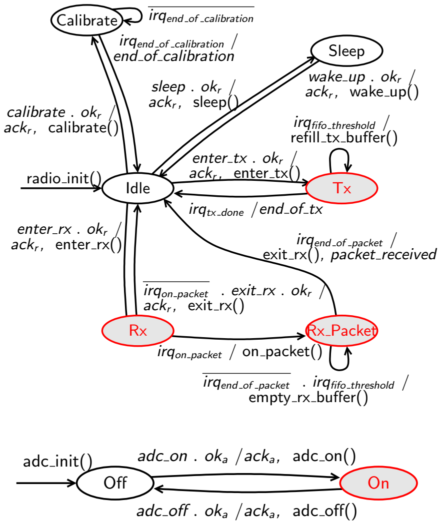
Example of Controllable Automaton

Timer



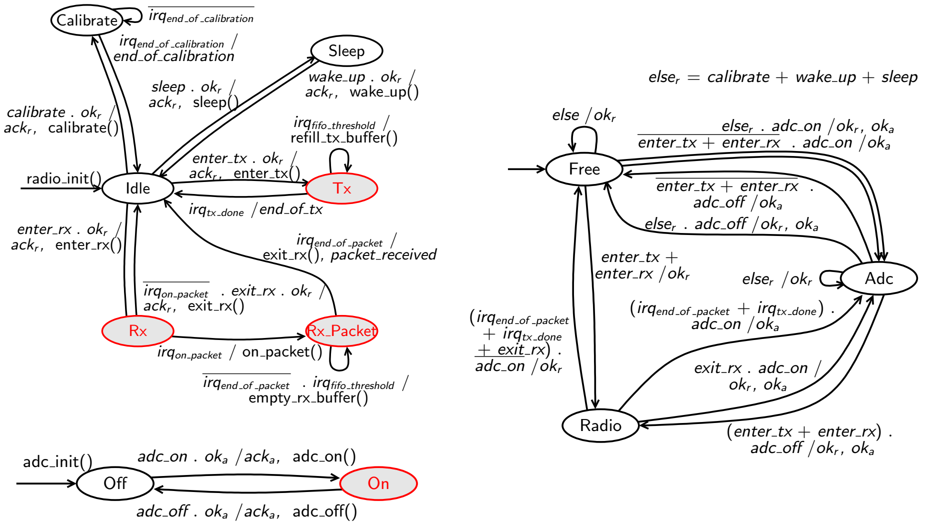
Exclusion of Energy-greedy States: Example

Radio Transceiver || ADC



Exclusion of Energy-greedy States: Example

Radio Transceiver ∥ ADC ∥ Controller



Principles of the Solution *(cont'd)*

(Para-)Virtualization Concept

- Interception and Control of Software Operations
- *Global* Resource Control ⇒ Centralized Knowledge
- May Forbid (or Enforce) Operations

Key Elements (Boolean Mealy Machines)

- Resource Automata
  - Inputs: Software Requests...
  - Outputs: Low-Level Code, Notifications...
- Controller

Principles of the Solution *(cont'd)*

(Para-)Virtualization Concept

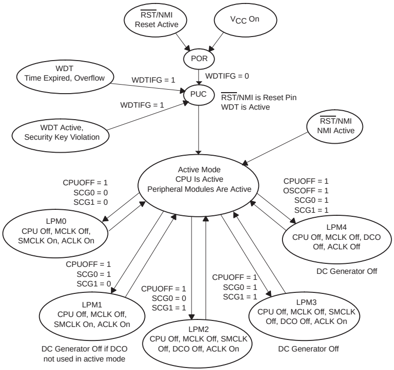
- Interception and Control of Software Operations
- *Global* Resource Control $\Rightarrow$ Centralized Knowledge
- May Forbid (or Enforce) Operations

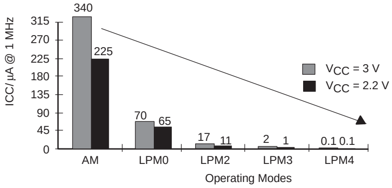
Key Elements (Boolean Mealy Machines)

- Resource Automata
  - Inputs: Software Requests & Approval Signals
  - Outputs: Low-Level Code, Notifications & Acknowledgments
- Controller
  - Inputs: Software & Hardware Requests
  - Outputs: Approval Signals

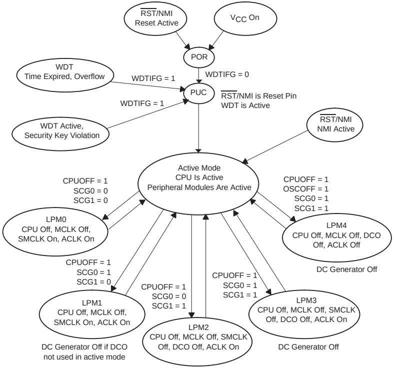
Enforcing Global Properties $\rightsquigarrow$ Designing the Controller

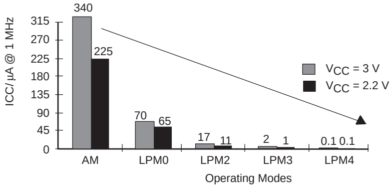
How to: Selecting the Best MCU Low-Power Mode





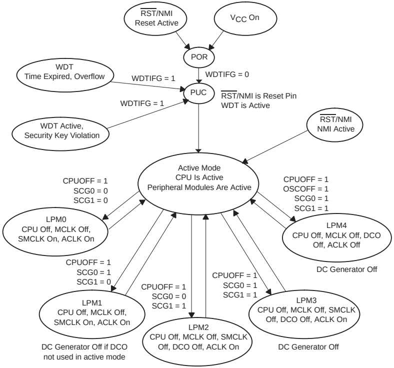
How to: Selecting the Best MCU Low-Power Mode

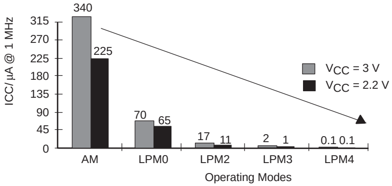




- Reducing Energy Consumption
  - ...as usual...

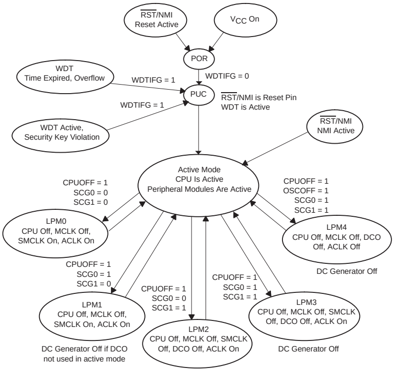
How to: Selecting the Best MCU Low-Power Mode

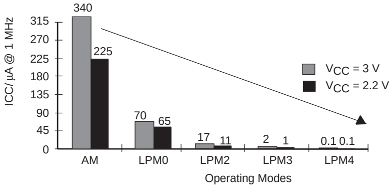




- Reducing Energy Consumption
  - . . . as usual. . .
- Wake-up time
  - Latency property

How to: Selecting the Best MCU Low-Power Mode





- Reducing Energy Consumption
  - . . . as usual. . .
- Wake-up time
  - Latency property
- To be sure to wake up!
  - Potential IRQs?

How to: Other Possibilities

- Mutual Exclusion of Accesses to Shared Resources
  - Safety Property

How to: Other Possibilities

- Mutual Exclusion of Accesses to Shared Resources
  - Safety Property
- Controlling Guest Tasks / Resources
  - $\rightsquigarrow$ Modification of the Guest Scheduler
  - Allowing Direct Access to the Resources

How to: Other Possibilities

- Mutual Exclusion of Accesses to Shared Resources
  - Safety Property
- Controlling Guest Tasks / Resources
  - $\rightsquigarrow$ Modification of the Guest Scheduler
  - Allowing Direct Access to the Resources
- Booking Controller
  - Slightly more complex (to use)... fits in the model however

Outline

Implementation

Proof of Concept

- Rough Implementation
- Resource Automata and Controller encoded in LUSTRE
- Multithreaded, CONTIKI
- Targetting Wsn430 Platform

Implementation

Proof of Concept

- Rough Implementation
- Resource Automata and Controller encoded in LUSTRE
- Multithreaded, CONTIKI
- Targetting Wsn430 Platform

Practicable?

- Extra Memory Footprint: 1.5 to 2.5 KB
- Timing Overhead: One Reaction $\approx$ 1,600 CPU cycles

Comparable to Solutions using Decentralized Control

Outline

Summary

Global Resource Control

- Synchronous Programming... in Wireless Sensor Networks!

Summary

Global Resource Control

- Synchronous Programming... in Wireless Sensor Networks!
- Many Possible Extensions $\rightsquigarrow$ Powerful

Summary

Global Resource Control

- Synchronous Programming... in Wireless Sensor Networks!
- Many Possible Extensions $\leadsto$ Powerful
- Para-Virtualizaton Concept $\leadsto$ Flexible Framework

Summary

Global Resource Control

- Synchronous Programming... in Wireless Sensor Networks!
- Many Possible Extensions ⇝ Powerful
- Para-Virtualizaton Concept ⇝ Flexible Framework

Implementation

- Proof of Concept
- Practicable
- Device Drivers Revealed "*easier*" to Develop

Perspectives

Evaluation

- Efficiency to Reduce Power Consumption?

Soon in the Senslab Testbed...

Perspectives

Evaluation

- Efficiency to Reduce Power Consumption?

Soon in the Senslab Testbed...

Automated Control

- Using Controller Synthesis

Perspectives

Evaluation

- Efficiency to Reduce Power Consumption?

Soon in the Senslab Testbed...

Automated Control

- Using Controller Synthesis

Synchronous Approach

- "More-Lustre" Solutions?
- Monitoring
- Other Domains (Real-Time...)

Thank you

Questions ?

Outline

- **Example Execution**

Example Execution