

Towards Sustainable IoT Nodes

Green Days at Lyon

Hugo Reymond¹ and Antoine Bernabeu²

¹INRIA ²LS2N

27 mars 2023



Inria



UMR

IRISA



Plan

- 1 Introduction
- 2 Intermittent computing
- 3 An Energy Consumption Aware Runtime
- 4 Maximizing the energy efficiency of an intermittent program
- 5 Conclusion



IoT devices are mostly powered by batteries → millions of batteries replaced each year

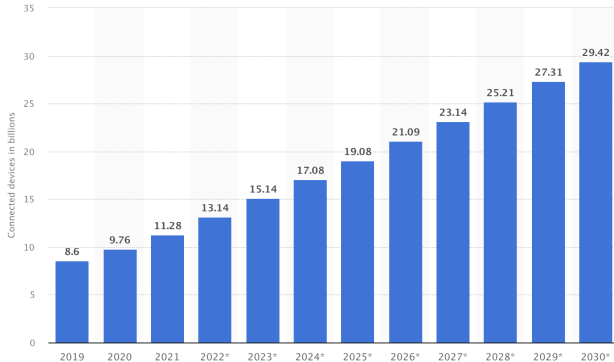
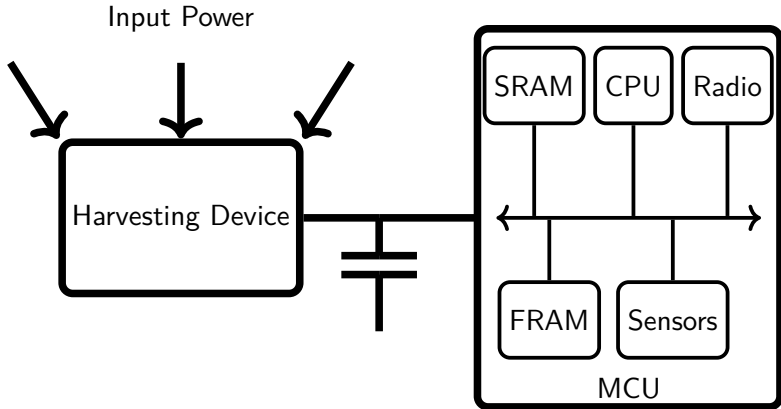


FIGURE 1 – Number of IoT devices [Source : Statistica 2023]



Batteryless Architecture



Use-case example : The *Mithræum of Circus Maximus*[1]



Context

- ▶ UNESCO archaeological site
- ▶ Temperature, humidity and vibration need to be monitored
- ▶ Intermittent computing techniques enable batteryless devices



Use-case example : The *Mithræum* of Circus Maximus[1]



Context

- ▶ UNESCO archaeological site
- ▶ Temperature, humidity and vibration need to be monitored
- ▶ Intermittent computing techniques enable batteryless devices

Battery sensing limits :

- ▶ Battery sensors require regular maintenance
- ▶ Restricted access to the site

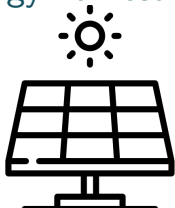


Plan

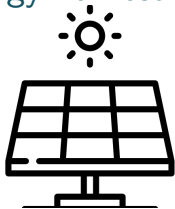
- 1 Introduction
- 2 Intermittent computing**
- 3 An Energy Consumption Aware Runtime
- 4 Maximizing the energy efficiency of an intermittent program
- 5 Conclusion



Energy harvesting



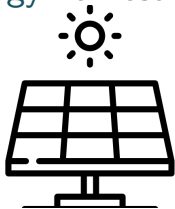
Energy harvesting



- ▶ Maintenance-free
- ▶ Reduced environmental footprint (no battery needed)



Energy harvesting



+



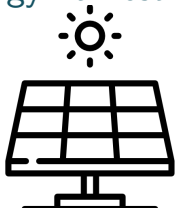
-

- ▶ Maintenance-free
- ▶ Reduced environmental footprint (no battery needed)

- ▶ Scarce energy
- ▶ Variable energy



Energy harvesting



+

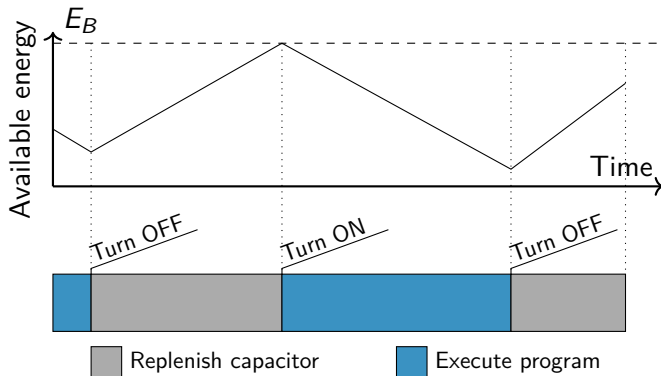
-

- ▶ Maintenance-free
- ▶ Reduced environmental footprint (no battery needed)
- ▶ Scarce energy
- ▶ Variable energy

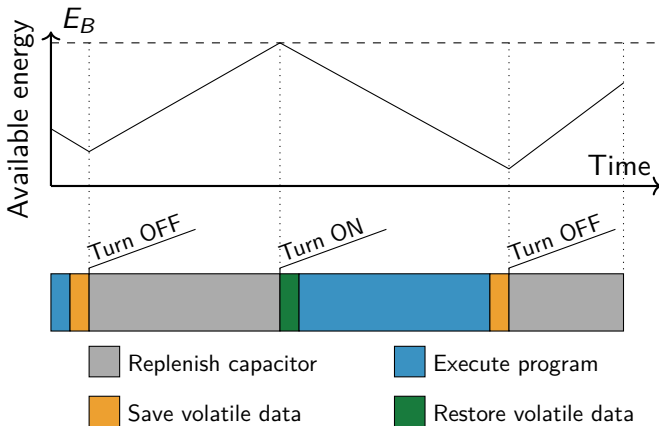
Not enough energy to power the platform continuously



Intermittent Computing



Intermittent Computing



Checkpointing

```
1  int sum = 0;
2  for(int i=offset; i<SIZE; i++)
3      sum += array
4  /* ..... */
5  class = f(sum);
```

Save checkpoint here?

Save checkpoint here?

Save checkpoint here?

- ▶ Saving too often \Rightarrow High overhead
- ▶ Not saving before power failure \Rightarrow No progress



Contributions

- ▶ RESURRECT : An Energy Consumption Aware Runtime
- ▶ SCHEMATIC : A joint checkpoint placement and memory allocation technique



Plan

- 1 Introduction
- 2 Intermittent computing
- 3 An Energy Consumption Aware Runtime**
- 4 Maximizing the energy efficiency of an intermittent program
- 5 Conclusion



RESURRECT

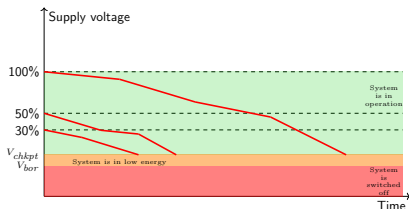
RESURRECT An Energy Consumption Aware Runtime

- ▶ *Power failure immunity* execution model : run code only when energy level is high enough to complete
- ▶ Optimal schedule computed offline based on worst-case energy consumption
- ▶ Runtime adaptation of schedule to account for actual execution parameters and harvested energy

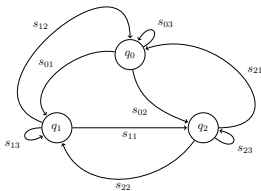


Offline Schedule Computation [2]

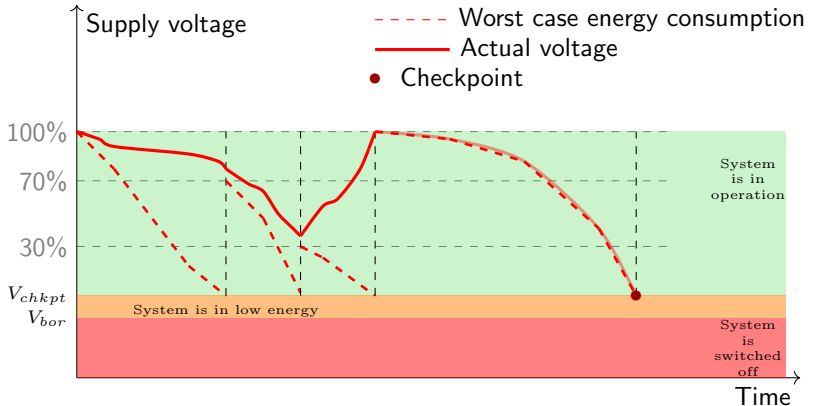
- ▶ Goal : maximize forward progress under energy consumption constraint
- ▶ Worst-case based approach : worst-case energy consumption per task, no harvesting
- ▶ Schedule computation as an optimization problem in time Petri nets with costs and rewards



$s_{01} = t_1$
 $s_{02} = t_1 \rightarrow t_2$
 $s_{03} = t_1 \rightarrow t_2 \rightarrow t_3$
 $s_{11} = t_2$
 $s_{12} = t_2 \rightarrow t_3$
 $s_{13} = t_2 \rightarrow t_3 \rightarrow t_1$
 $s_{21} = t_3$
 $s_{22} = t_3 \rightarrow t_1$
 $s_{23} = t_3 \rightarrow t_1 \rightarrow t_2$



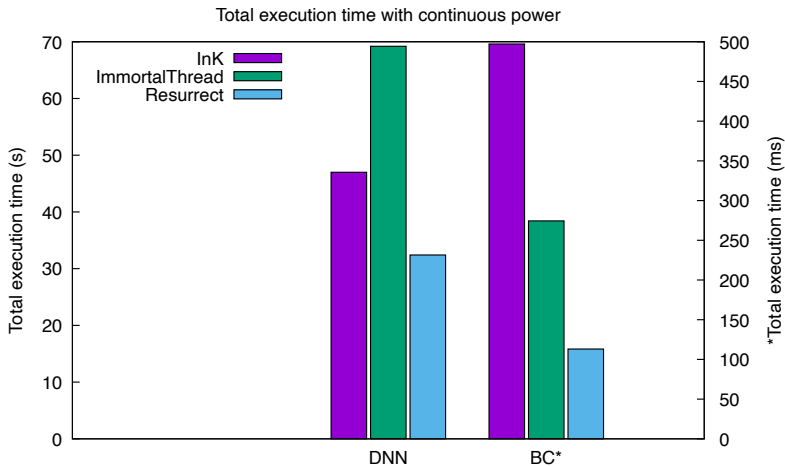
Online Adaption



- ▶ Integration of the mixed offline-online scheduling in TrampolineRTOS



Performances



Plan

- 1 Introduction
- 2 Intermittent computing
- 3 An Energy Consumption Aware Runtime
- 4 Maximizing the energy efficiency of an intermittent program**
- 5 Conclusion



SCHEMATIC

SCHEMATIC

Compile-time joint checkpoint placement and
memory allocation



SCHEMATIC

SCHEMATIC

Compile-time joint checkpoint placement and
memory allocation

1. Ensure forward progress



SCHEMATIC

SCHEMATIC

Compile-time joint checkpoint placement and
memory allocation

1. Ensure forward progress
2. Take the most out of the dual memory architecture



SCHEMATIC

SCHEMATIC

Compile-time joint checkpoint placement and memory allocation

1. Ensure forward progress
2. Take the most out of the dual memory architecture
3. Reduce overall energy consumption



Checkpoint placement and memory allocation

```
1  int sum = 0;
2  for(int i=offset;i<SIZE; i++)
3      sum += array[i];
4  /* ..... */
5  class = f(sum);
```



Checkpoint placement and memory allocation

```
1  int sum = 0;
2  for(int i=offset; i<SIZE; i++)
3      sum += array[i];
4  /* ..... */
5  class = f(sum);
```

Save checkpoint here?

Save checkpoint here?

Save checkpoint here?



Checkpoint placement and memory allocation

```
1 int sum = 0;
2 for(int i=offset; i<SIZE; i++)
3     sum += array[i];
4 /* ..... */
5 class = f(sum);
```

energy consumed
<
energy available?



Checkpoint placement and memory allocation

Allocation : SRAM or FRAM ?

```
1 int sum = 0;
2 for(int i=offset; i<SIZE; i++)
3     sum += array[i];
4 /* ..... */
5 class = f(sum);
```

Allocation : SRAM or FRAM ?



Checkpoint placement and memory allocation

Allocation : SRAM or FRAM ?

```
1 int sum = 0;
2 for(int i=offset; i<SIZE; i++)
3     sum += array[i];
4 /* ..... */
5 class = f(sum);
```

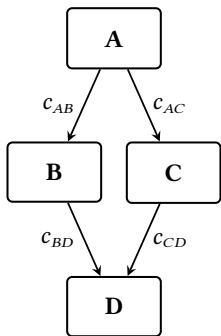
Allocation : SRAM or FRAM ?

Memory allocation parameters :

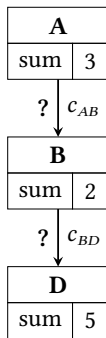
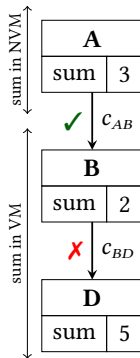
- ▶ number of accesses
- ▶ size
- ▶ liveliness



Eloise algorithm



a. Analyzed CFG

b. Path (A, B, D)
before analysisc. Path (A, B, D)
after analysis

□ Basic block

? Potential checkpoint location

✓ Checkpoint location enabled ✗ Checkpoint location disabled



Performance

- ▶ SCHEMATICensure forward progress even in extreme-intermittency setup
- ▶ Overall, 51% reduction of the energy consumption



Plan

- 1 Introduction
- 2 Intermittent computing
- 3 An Energy Consumption Aware Runtime
- 4 Maximizing the energy efficiency of an intermittent program
- 5 Conclusion



Conclusion

- ▶ Battery-less devices : Low-maintenance sensing with a reduced environmental impact



Conclusion

- ▶ Battery-less devices : Low-maintenance sensing with a reduced environmental impact
- ▶ Intermittent computing : require adaptation



Conclusion

- ▶ Battery-less devices : Low-maintenance sensing with a reduced environmental impact
- ▶ Intermittent computing : require adaptation



Conclusion

- ▶ Battery-less devices : Low-maintenance sensing with a reduced environmental impact
- ▶ Intermittent computing : require adaptation

NOP project

- ▶ Propose solutions to handle intermittent execution
- ▶ Use-case : Bird sound recognition



References I



AFANASOV, M., AND AL.

Battery-less zero-maintenance embedded sensing at the mithræum of circus maximus.

In 18th Conference on Embedded Networked Sensor Systems (2020), SenSys '20.



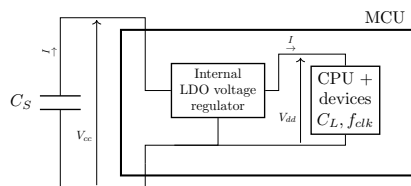
BERNABEU, A., BÉCHENNEC, J.-L., BRIDAY, M., FAUCOU, S., AND ROUX, O.

Cost-optimal timed trace synthesis for scheduling of intermittent embedded systems.

Discrete Event Dynamic Systems (Dec. 2022).

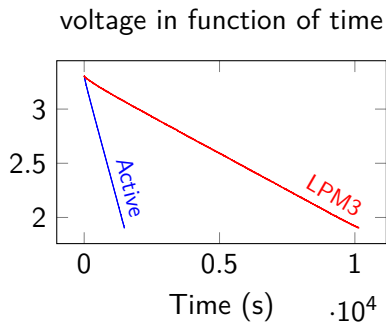


Energy Consumption Model

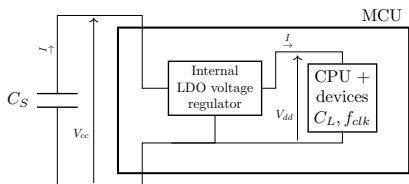


$$\frac{dV_{cc}}{dt} = \frac{f_{clk} \times C_L \times V_{dd}}{C_S}$$

super-capacitor voltage (V)



Energy Consumption Model



$$\frac{dV_{CC}}{dt} = \frac{f_{clk} \times C_L \times V_{dd}}{C_S}$$

super-capacitor voltage (V)

