**POLITECNICO DI TORINO**

**Master’s Degree in Computer Engineering**



**Energy Management for IoT 01UDGOV**

**Lab 1 Report**

**Group-8**

**Supervisor:**

**Prof. Massimo Poncino**

**Candidates:**

**Diego Porto (s313169)**

**Mohamed elsayed Mahmoud elsayed**

**elsisy (s330451)**

**A.Y. 2024/25**

**Table of Contents**

[1. Timeout Policy 2](#_Toc191239625)

[1.1. RUN-IDELE states 3](#_Toc191239626)

[1.1.1 Workload 1- (fast sensor response’s) 3](#_Toc191239627)

[1.1.2 Workload 2 – (slow sensor response’s) 4](#_Toc191239628)

[1.2. RUN-SLEEP states 6](#_Toc191239629)

[1.2.1 Summary of Insights – WORKLAOD-1 11](#_Toc191239630)

[1.2.2 Summary of Insights – WORKLAOD-2 12](#_Toc191239631)

[1.3. Three PSM STATE MACHINE 12](#_Toc191239632)

[1.3.1 Examples 12](#_Toc191239633)

[2. History Policy 13](#_Toc191239634)

[2.1. Examples and results 14](#_Toc191239635)

1

# Timeout Policy

In the first part of the timeout policy, we will run the simulator using the two given workloads from two different sensor responses with the default timeout policy. The transition constraint only allows transitions between the **RUN** and **IDLE** states. The PSM file used is dpm-simulator/example/psm.txt.

The differences in the sensor's response times between workload\_1.txt and workload\_2.txt can significantly impact the effectiveness and behaviour of the DPM policy.

For example:

● Fast Sensor (workload\_1.txt): With a response time of 4ms, it is better for the system to remain in the RUN state rather than transitioning to IDLE.

● Slow Sensor (workload\_2.txt): With a response time of 100ms, it is more convenient for the system to switch to IDLE or even sleep in the meantime, as the overhead caused by the transition may still be less than staying in the RUN state constantly.

Shorter timeout values may be more effective in managing power consumption, as the system can quickly transition to low-power states when the sensor is inactive. However, we must also avoid frequent transitions, as this can be counterproductive due to the high transition overhead.

## RUN-IDELE states

### Workload 1- (fast sensor response’s)

This workload uses "fast" sensors, meaning the value is returned after 4ms. The file contains two values on each line: the first value is the arrival time of the task, and the second value represents the duration of that task. The following four different timeout values are considered: 0ms, 1ms, 4ms, 20ms, and 40ms.

The command to run the first workload with a timeout of 1ms is:

Shell. /dpm\_simulator -t 1 -psm example/psm.txt -wl ../workloads/workload\_1.txt > results/workload\_1/workload\_1ms

The results of the simulation are stored in the results folder, divided by workload. In the output, the **[sim]** section provides detailed results of the simulation:

* **Active and Inactive Time**: The total time the system spent in active and inactive states.
* **Total Time**: The total simulation time, both with and without DPM.
* **State Times**: The total time spent in each state (Run, Idle).
* **Timeout Waiting Time**: The time spent waiting for the timeout to expire.
* **Transitions Time**: The total time spent in state transitions.
* **Number of Transitions**: The total number of state transitions.
* **Energy for Transitions**: The total energy consumed during state transitions.
* **Total Energy**: The total energy consumption with and without DPM. Results for Different Timeout Values In this first workload, we can observe the following values:

Timeout 1ms:

Number of Transitions: 138

Total Energy with DPM: 0.6997882530J

Timeout Waiting Time: 0.086400s

Timeout 4ms:

Number of Transitions: 18

Total Energy with DPM: 0.7032023760J

Timeout Waiting Time: 0.256100s

Timeout 20ms:

Number of Transitions: 18

Total Energy with DPM: 0.7070946960J

Timeout Waiting Time: 0.400100s

Timeout 40ms:

Number of Transitions: 18

Total Energy with DPM: 0.7119600960J

Timeout Waiting Time: 0.580100s

Timeout 60ms:

Number of Transitions: 18

Total Energy with DPM: 0.7168254960J

Timeout Waiting Time: 0.760100s

As shown, different timeout values affect the number of state transitions, total energy consumption, and the time spent in each state. Shorter timeouts lead to more frequent transitions and overall lower energy consumption, while longer timeouts result in fewer transitions and higher energy consumption. This occurs because the overhead caused by the transition between

The RUN and IDLE states has a lower impact on energy consumption compared to staying in the RUN state during that time.

**Conclusion,** overall, from the results, we observed that lower timeout values lead to more frequent state

Transitions, resulting in higher transition time values and energy consumption. However, they also lead to

Lower overall energy consumption. On the other hand, higher timeout values result in fewer transitions, which

Means lower transition times and energy consumption, but they also lead to higher overall energy

Consumption. Given these results, for the first workload (workload\_1, fast sensors), despite the high number of

Transitions, it is better to keep the timeout threshold at lower values, such as 0 or 1ms.

### Workload 2 – (slow sensor response’s)

This workload uses "slow" sensors, meaning the value is returned after 100 ms. The file workload\_2.txt

Contains two values on each line: the first value is the arrival time of the task, and the second value represents

The duration of that task. Due to the longer response time of the sensors, the timeout values will differ

Compared to those used in Workload 1. The timeout values are as follows: 80ms, 100ms, 120ms, 140ms, 160ms,

180ms, and 200ms.

Summary of Results for Different Timeout Values Total Energy Consumption: The energy consumption without

DPM is always Total Energy w/o DPM = 30.1388136000J.The total energy consumption with DPM for different

Timeout values as follows,

timeout\_80ms:

Number of transitions = 194 Energy for transitions = 0.0019400000J

Tot. Energy w DPM = 0.9073267669J

timeout\_80ms: Timeout waiting time = 7.922800s

Total time in state Run = 10.469800s

Total time in state Idle = 1081.439100s

timeout\_100ms:

timeout\_100ms: Number of transitions = 56 Energy for transitions = 0.0005600000J

Tot. Energy w DPM = 0.9464934979J

timeout\_100ms: Timeout waiting time = 9.421700s

Total time in state Run = 11.968700s

Total time in state Idle = 1079.995400s

timeout\_120ms:

Number of transitions = 20 Energy for transitions = 0.0002000000J

Tot. Energy w DPM = 0.9552641029J

timeout\_120ms: Timeout waiting time = 9.759200s

Total time in state Run = 12.306200s

Total time in state Idle = 1079.671900s

timeout\_140ms:

Number of transitions = 20 Energy for transitions = 0.0002000000J

Tot. Energy w DPM = 0.9606701029J

timeout\_140ms: Timeout waiting time = 9.959200s

Total time in state Run = 12.506200s

Total time in state Idle = 1079.471900s

timeout\_160ms:

Number of transitions = 20 Energy for transitions = 0.0002000000J

Tot. Energy w DPM = 0.9660761029J

timeout\_160ms: Timeout waiting time = 10.159200s

Total time in state Run = 12.706200s

Total time in state Idle = 1079.271900s

timeout\_180ms:

Number of transitions = 20 Energy for transitions = 0.0002000000J

Number of transitions = 20 Energy for transitions = 0.0002000000J

Tot. Energy w DPM = 0.9714821029J

timeout\_180ms: Timeout waiting time = 10.359200s

Total time in state Run = 12.906200s

Total time in state Idle = 1079.071900s

timeout\_200ms:

Number of transitions = 20 Energy for transitions = 0.0002000000J

Tot. Energy w DPM = 0.9768881029J

timeout\_200ms: Timeout waiting time = 10.559200s

Total time in state Run = 13.106200s

Total time in state Idle = 1078.871900s

Conclusion Optimal Timeout Value: As the timeout value increases, the total energy consumption with DPM

Slightly increases. For instance, the energy consumption with DPM is 0.9073J at 80ms and increases to 0.9769J

At 200ms. This indicates that shorter timeout values are more energy-efficient. Number of Transitions:

The number of state transitions decreases significantly as the timeout value increases. For example, there are

194 transitions at 80ms compared to only 20 transitions at 200ms. While fewer transitions can reduce the

Overhead and energy consumption associated with state changes, it is still beneficial to switch to the IDLE state

Despite the transition overhead. Timeout Waiting Time: The timeout waiting time increases with higher timeout

Values. This suggests that higher timeout values lead to longer periods of inactivity before transitioning to

A low-power state.

Final Conclusion: For workload\_2, similar to workload\_1, shorter timeouts lead to more frequent transitions but

Overall lower energy consumption. On the other hand, longer timeouts result in fewer transitions but higher

Energy consumption. Additionally, as expected, the energy consumed with DPM is significantly lower than

Without DPM across all timeout values for both workloads.

## RUN-SLEEP states

To modify the timeout policy to enable transitions to SLEEP, the following steps need to be taken:

1. **Modify the dpm\_decide\_state function**:  
   Inside dpm\_policies.c, change the relevant logic to include the new macro PSM\_STATE\_SLEEP, which is defined in the psm.h file.
2. **Modify the dpm\_timeout\_params struct**:  
   In dpm\_policies.h, update the struct to create an array of two double numbers, timeout[2], which will represent the timeouts for both **RUN** and **SLEEP** states.
3. **Edit the utilities.h file**:  
   Update the struct in utilities.h to reflect the changes made in dpm\_policies.h.

Once these modifications are made, you can compare the results in the results/workload\_2 folder. The primary focus will be on energy consumption and the time spent in different states (RUN, IDLE, SLEEP).

“psm.h”

typedef struct {

/\* Day2: you can add/change stuff here \*/

psm\_time\_t timeout[2];

} dpm\_timeout\_params;

“utilities.h”

tparams->timeout[0] = atof(argv[++cur]);

tparams->timeout[1] = atof(argv[++cur]);

printf("the time out values as input are <<<<<<<<<<<>>>>>>>>>>>>>>\n");

printf("the value of time out idle is %f \n",tparams->timeout[0]);

printf("the value of time out sleep is %f \n",tparams->timeout[1]);

“dpm\_policies.c”

if((t\_curr > t\_inactive\_start + tparams.timeout[1]) && (tparams.timeout[1] > tparams.timeout[0]))

\*next\_state = PSM\_STATE\_SLEEP;

else if((t\_curr > t\_inactive\_start + tparams.timeout[0]) && (tparams.timeout[1] < tparams.timeout[0]))

\*next\_state = PSM\_STATE\_IDLE;

else {

\*next\_state = PSM\_STATE\_RUN;}

break;

**Comparison for workload\_2 with IDLE and SLEEP**

**Timeout\_0ms vs timeout\_0ms\_sleep**

**Total Time in States**  
timeout\_0ms: More time in Idle state (1089.350400s), no time in Sleep state.  
timeout\_0ms\_sleep: All inactive time spent in Sleep state (1088.938200s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_0ms: 0.009900s  
timeout\_0ms\_sleep: 0.009900s

**Transitions Time**  
timeout\_0ms: 0.079200s  
timeout\_0ms\_sleep: 0.495000s

**Number of Transitions**  
timeout\_0ms: 198transitions.  
timeout\_0ms\_sleep: 198 transitions.

**Energy for Transitions**  
timeout\_0ms: 0.0019800000J  
timeout\_0ms\_sleep: 0.1999800000J

**Total Energy with DPM**  
timeout\_0ms: 0.6934801680J  
timeout\_0ms\_sleep: 0.3685548779J

**Timeout\_80ms vs timeout\_80ms\_sleep**

**Total Time in States**  
timeout\_80ms: More time in Idle state (1081.439100s), no time in Sleep state.  
timeout\_80ms\_sleep: All inactive time spent in Sleep state (1081.076900s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_80ms: 7.922800s  
timeout\_80ms\_sleep: 7.906200s

**Transitions Time**  
timeout\_80ms: 0.077600s  
timeout\_80ms\_sleep: 0.460000s

**Number of Transitions**  
timeout\_80ms: 194 transitions.  
timeout\_80ms\_sleep: 184 transitions.

**Energy for Transitions**  
timeout\_80ms: 0.0019400000J  
timeout\_80ms\_sleep: 0.1858400000J

**Total Energy with DPM**  
timeout\_80ms: 0.9073267669J  
timeout\_80ms\_sleep: 0.5716452409J

**Timeout\_100ms vs timeout\_100ms\_sleep:**

**Total Time in States**  
timeout\_100ms: More time in Idle state (1079.995400s), no time in Sleep state.  
timeout\_100ms\_sleep: All inactive time spent in Sleep state (1079.949900s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_100ms: 9.421700s  
timeout\_100ms\_sleep: 9.353200s

**Transitions Time**  
timeout\_100ms: 0.022400s  
timeout\_100ms\_sleep: 0.140000s

**Number of Transitions**  
timeout\_100ms: 56 transitions.  
timeout\_100ms\_sleep: 56 transitions.

**Energy for Transitions**  
timeout\_100ms: 0.0005600000J  
timeout\_100ms\_sleep: 0.0565600000J

**Total Energy with DPM**  
timeout\_100ms: 0.9464934979J  
timeout\_100ms\_sleep: 0.4822010109J

**Comparison for workload\_1 with IDLE and SLEEP**

**Timeout\_0ms vs timeout\_0ms\_sleep:**

**Total Time in States**  
timeout\_0ms: More time in Idle state (1079.576000s), no time in Sleep state.  
timeout\_0ms\_sleep: All inactive time spent in Sleep state (1079.443000s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_0ms: 7.922800s  
timeout\_0ms\_sleep: 7.906200s

**Transitions Time**  
timeout\_0ms: 0.068000s  
timeout\_0ms\_sleep: 0.205000s

**Number of Transitions**  
timeout\_0ms: 170 transitions.  
timeout\_0ms\_sleep: 82 transitions.

**Energy for Transitions**  
timeout\_0ms: 0.0017000000J  
timeout\_0ms\_sleep: 0.0828200000J

**Total Energy with DPM**  
timeout\_0ms: 0.6979953200J  
timeout\_0ms\_sleep: 0.2607854300J

**Timeout\_1ms vs timeout\_1ms\_sleep:**

timeout\_1ms: More time in Idle state (1079.510900s), no time in Sleep state.  
timeout\_1ms\_sleep: All inactive time spent in Sleep state (1079.414800s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_1ms: 0.086400s  
timeout\_1ms\_sleep: 0.047300s

**Transitions Time**  
timeout\_1ms: 0.055200s  
timeout\_1ms\_sleep: 0.190000s

**Number of Transitions**  
timeout\_1ms: 138 transitions.  
timeout\_1ms\_sleep: 76 transitions.

**Energy for Transitions**  
timeout\_1ms: 0.0017000000J  
timeout\_1ms\_sleep: 0.0767600000J

**Total Energy with DPM**  
timeout\_1ms: 0.6997882530J  
timeout\_1ms\_sleep: 0.2559152120J

**Timeout\_4ms vs timeout\_4ms\_sleep:**

timeout\_4ms: More time in Idle state (1079.388800s), no time in Sleep state.  
timeout\_4ms\_sleep: All inactive time spent in Sleep state (1079.383400s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_4ms: 0.256100s  
timeout\_4ms\_sleep: 0.223700s

**Transitions Time**  
timeout\_4ms: 0.007200s  
timeout\_4ms\_sleep: 0.045000sv

**Number of Transitions**  
timeout\_4ms: 18 transitions.  
timeout\_4ms\_sleep: 18 transitions.

**Energy for Transitions**  
timeout\_4ms: 0.0001800000J  
timeout\_4ms\_sleep: 0.0181800000J

**Total Energy with DPM**  
timeout\_4ms: 0.7032023760J  
timeout\_4ms\_sleep: 0.2022010260J

**Timeout\_20ms vs timeout\_20ms\_sleep:**

**Total Time in States**  
timeout\_20ms: More time in Idle state (1079.244800s), no time in Sleep state.  
timeout\_20ms\_sleep: All inactive time spent in Sleep state (1079.239400s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_20ms: 0.400100s  
timeout\_20ms\_sleep: 0.367700s

**Transitions Time**  
timeout\_20ms: 0.007200s  
timeout\_20ms\_sleep: 0.045000s

**Number of Transitions**  
timeout\_20ms: 18 transitions.  
timeout\_20ms\_sleep: 18 transitions.

**Energy for Transitions**  
timeout\_20ms: 0.0001800000J  
timeout\_20ms\_sleep: 0.0181800000J

**Total Energy with DPM**  
timeout\_20ms: 0.7070946960J  
timeout\_20ms\_sleep: 0.2061624660J

**Timeout\_40ms vs timeout\_40ms\_sleep:**

**Total Time in States**  
timeout\_40ms: More time in Idle state (1079.064800s), no time in Sleep state.  
timeout\_40ms\_sleep: All inactive time spent in Sleep state (1079.059400s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_40ms: 0.580100s  
timeout\_40ms\_sleep: 0.547700s

**Transitions Time**  
timeout\_40ms: 0.007200s  
timeout\_40ms\_sleep: 0.045000s

**Number of Transitions**  
timeout\_40ms: 18 transitions.  
timeout\_40ms\_sleep: 18 transitions.

**Energy for Transitions**  
timeout\_40ms: 0.0001800000J  
timeout\_40ms\_sleep: 0.0181800000J

**Total Energy with DPM**  
timeout\_40ms: 0.7119600960J  
timeout\_40ms\_sleep: 0.2111142660J

**Timeout\_60ms vs timeout\_60ms\_sleep:**

**Total Time in States**  
timeout\_60ms: More time in Idle state (1078.884800s), no time in Sleep state.  
timeout\_60ms\_sleep: All inactive time spent in Sleep state (1078.879400s), no time in Idle state.

**Timeout Waiting Time**  
timeout\_60ms: 0.760100s  
timeout\_60ms\_sleep: 0.727700s

**Transitions Time**  
timeout\_60ms: 0.007200s  
timeout\_60ms\_sleep: 0.045000s

**Number of Transitions**  
timeout\_60ms: 18 transitions.  
timeout\_60ms\_sleep: 18 transitions.

**Energy for Transitions**  
timeout\_60ms: 0.0001800000J  
timeout\_60ms\_sleep: 0.0181800000J

**Total Energy with DPM**  
timeout\_60ms: 0.7168254960J  
timeout\_60ms\_sleep: 0.2160660660J

### Summary of Insights – WORKLAOD-1

**Energy Consumption**

The simulations with the "sleep" parameter (timeout = 4ms) show significantly lower total energy consumption with DPM compared to their counterparts without the "sleep" parameter. Similarly, the simulations with the "IDLE" parameter (timeout = 0ms) show significantly lower total energy consumption with DPM compared to their counterparts without the "IDLE" parameter.

Therefore, the best timeout value for Sleep mode with respect to workload\_1 is 4ms, which is particularly convenient given that the average response time is 4ms for fast sensor responses.

For the IDLE state, a timeout value of 0ms is also efficient, as it logically facilitates transitions from RUN to IDLE, thus saving energy overall. However, a timeout of 4ms for IDLE could also be more beneficial than 0ms, considering that the number of transitions is 18 instead of 82. Both the 0ms timeout for IDLE and the 4ms timeout for Sleep have been tested, and the results are based on the experiment.

In conclusion, the optimal configuration is either:

* 4ms Sleep / 4ms Idle, or
* 4ms Sleep / 0ms Idle.

### Summary of Insights – WORKLAOD-2

**Energy Consumption**

The simulations with the "sleep" parameter (timeout = 0ms\_sleep) show significantly lower total energy consumption with DPM compared to their counterparts without the "sleep" parameter.  
Similarly, the simulations with the "IDLE" parameter (timeout = 0ms\_IDLE) also show significantly lower total energy consumption with DPM compared to their counterparts without the "IDLE" parameter.

## Three PSM STATE MACHINE

In this three-state PSM (Power State Management) machine, it is not a binary state machine that only transitions between IDLE and RUN, or between SLEEP and RUN. Instead, the transitions can involve moving from IDLE to RUN to SLEEP, or from SLEEP to RUN to IDLE, with flexibility in the state transitions. Additionally, the system allows transitions from RUN to any state, subject to constraints.

The code edits are designed to handle timeouts for both the SLEEP and IDLE states. Specifically, tparams.timeout[1] corresponds to the SLEEP timeout, while tparams.timeout[0] corresponds to the IDLE timeout. The transition logic works as follows: if the SLEEP timeout is greater than the IDLE timeout, the system will prioritize the SLEEP state. Otherwise, it will transition to the IDLE state.

During the workload, the PSM process will adjust based on the system's idle duration. This means the system will alternate between some time spent in IDLE and some time in SLEEP, optimizing energy savings. The state transitions will follow the constraints described in the report, ensuring the best energy-saving configuration while respecting the transition conditions.

### Examples

For example, when the best value for SLEEP is 4ms and IDLE is either 4ms or 0ms for workload\_1, and when the values for the IDLE timeout are set to 0ms and the SLEEP timeout to 4ms in the three-state system, the result will be as follows:

Energy w DPM = 0.1976708950J

Energy for transitions = 0.0195400000J

N. of transitions = 154

Transitions time = 0.099400s

Timeout waiting time = 0.006800s

Total time in state Sleep = 1079.376200s

Total time in state Idle = 0.170100s

Total time in state Run = 2.930800s

As shown, the resulting energy value will be lower when the system is in SLEEP mode during idle time with a 4ms timeout, which saves energy compared to other configurations. For instance, with timeout\_4ms\_sleep, the energy consumption is 0.2022010260J, whereas setting the IDLE timeout to 0ms results in energy consumption of 0.6979953200J.

The second example is for workload\_2, where both the SLEEP and IDLE timeouts are set to 0ms. When these values are passed to the three-state machine, the result is that the PSM is always in SLEEP mode. However, the outcome is either good or at least comparable to the SLEEP mode. In fact, the result of using 0/0 timeouts will be equivalent to setting the system to SLEEP mode for workload\_2.

if(tparams.timeout[1] > tparams.timeout[0] && (t\_curr > t\_inactive\_start + tparams.timeout[1]))

{

if((prev\_state == PSM\_STATE\_SLEEP) || (prev\_state == PSM\_STATE\_RUN) )

{

\*next\_state = PSM\_STATE\_SLEEP;

}

else {

\*next\_state = PSM\_STATE\_RUN;}

}

else if((t\_curr > t\_inactive\_start + tparams.timeout[0]))

{ if((prev\_state == PSM\_STATE\_IDLE) || (prev\_state == PSM\_STATE\_RUN) )

{ \*next\_state = PSM\_STATE\_IDLE;

}

else {

\*next\_state = PSM\_STATE\_RUN;}

} else {

\*next\_state = PSM\_STATE\_RUN; } break;

# History Policy

The **History Policy** is a predictive policy that estimates the transition time based on the previous workload pattern. The computed value, **Tpred**, is compared to the thresholds for **IDLE** and **SLEEP**. After making a decision, the system can independently transition to one of the low-power states.

In the code section related to DPM history in the **dpm\_policies.c** file, **Tpred** is calculated using one of two possible methods: the **polynomial method** and the **regression method**. Both methods can be activated using **#ifdef** macros in the header section of **dpm\_policies.c**.

Once the time prediction is calculated, the next step is to decide whether the system should transition to **SLEEP** or **IDLE**. This decision is made by evaluating specific conditions. For example, to transition to **SLEEP**, the predicted time must be higher than the **time breakeven** and the **sleep threshold**, and there must be higher priority for **SLEEP** than for **IDLE**. If these conditions are met, the system will transition to **SLEEP**. If not, the system will transition to **IDLE**, provided that the **time prediction** is higher than both the **breakeven time for IDLE** and the **IDLE threshold**.

If the prediction fails to meet the conditions for either **SLEEP** or **IDLE**, the system will transition to **RUN**. This ensures that the transition is either from **SLEEP** to **RUN**, **IDLE** to **RUN**, or vice versa, as necessary. The code has been edited to allow transitions between the three states, based on these conditions. The code works as well in two state transition as well

## Examples and results

With workload\_1, the best threshold timeout policy was 4ms for SLEEP mode and 0ms for IDLE. When combining 4ms for SLEEP mode with 0ms for IDLE in the three-state timeout policy, the resulting energy consumption was 0.1976708950J.

In the case of the History Policy using the polynomial method, with the following coefficients:

* k1 = 0.9
* k2 = 0.80
* k3 = 0.70
* k4 = 0.60
* k5 = 0.50

**Example 1:**

And with a threshold of 4ms for SLEEP and 0ms for IDLE, the results are as follows:

Energy w DPM = 0.2587870080J

Energy for transitions = 0.0808200000J

N. of transitions = 82

Transitions time = 0.200800s

Timeout waiting time = 0.004100s

Total time in state Sleep = 1079.444700s

Total time in state Idle = 0.002500s

Total time in state Run = 2.928100s

The result is generally quite good and close to the result of the three-state timeout policy. Changes in the coefficients lead to either more energy savings or an increase in energy consumption for DPM. However, based on practical examples, in the case of **4ms for sleep** and **0ms for idle**, the energy consumption typically ranges from **0.25J to 0.27J**.

**Example 2:**

Since the predictive three-state machine can also function as a binary machine—transitioning between RUN and IDLE, or SLEEP and RUN—with a threshold of 1ms for the IDLE period, the system can operate under predictive history DPM. If the SLEEP threshold is ignored (disabled, set to 0), the machine will only trigger between IDLE and RUN.

Using the following coefficients:

* k1 = 0.9
* k2 = 0.80
* k3 = 0.70
* k4 = 0.60
* k5 = 0.50

The results are as follows:

* Predictive\_1ms\_idle: Energy with DPM = 0.6980541630J
* Predictive\_1ms\_sleep: Energy with DPM = 0.2587870080J
* Timeout\_1ms: Energy = 0.6997882530J
* Timeout\_1ms\_sleep: Energy = 0.2559152120J

For workload**\_1**, the results of the predictive model are close to the timeout policy. The predictive model performs better in terms of the IDLE threshold compared to the timeout idle policy, but it doesn't perform as well as the timeout sleep policy, although the results are still close.

For workload\_2, using the same coefficients, the results for the IDLE and SLEEP thresholds are better than using the timeout policy for either IDLE or SLEEP.

**Example 3:**

* Predictive\_80ms\_idle: Energy with DPM = 0.7203257410J
* Predictive\_80ms\_sleep: Energy with DPM = 0.3670321360J
* Timeout\_80ms: Energy = 0.9073267669J
* Timeout\_80ms\_sleep: Energy = 0.5716452409J

**Example 4:**

● **Predictive\_100ms\_idle**: Energy with DPM **= 0.7203257410J**

● **Predictive\_100ms\_sleep**: Energy with DPM **= 0.3670321360J**

● **Timeout\_100ms:** Energy **= 0.9464934979J**

● **Timeout\_100ms\_sleep:** Energy **= 0.4822010109J**

|  |
| --- |
| if(t\_curr > t\_inactive\_start)  {  #ifdef T\_Pre\_Poly  for( int i =0; i < DPM\_HIST\_WIND\_SIZE; i++)  {  T\_Pre += hparams.alpha[i] \* pow(history[i],i);  }  #endif  #ifdef T\_Pre\_N\_Reg  for( int i =1; i < DPM\_HIST\_WIND\_SIZE; i++)  {  T\_Pre += hparams.alpha[i]\*(history[DPM\_HIST\_WIND\_SIZE-(i)]);  }  T\_Pre += hparams.alpha[0];  #endif  if((T\_Pre >= T\_B\_E\_S) && (T\_Pre >= Thr\_Sleep) && (Thr\_Sleep > Thr\_Idle ))  {  if((prev\_state == PSM\_STATE\_SLEEP) || (prev\_state == PSM\_STATE\_RUN) )  {  \*next\_state = PSM\_STATE\_SLEEP;  }  else {  \*next\_state = PSM\_STATE\_RUN;}  }  else if((T\_Pre >= T\_B\_E\_I) && (T\_Pre >= Thr\_Idle))  {    if((prev\_state == PSM\_STATE\_IDLE) || (prev\_state == PSM\_STATE\_RUN) )  {  \*next\_state = PSM\_STATE\_IDLE;  }  else {  \*next\_state = PSM\_STATE\_RUN;}  }  }  else {  \*next\_state = PSM\_STATE\_RUN;  }  break;  #endif |