การจัดตารางงานในระบบปฏิบัติการแบบเวลาจริงบนไมโครโพรเซสเซอร์ชนิดหลายแกน
Scheduling Tasks in Real-Time Operating Systems on Multiple Core Microprocessors

อังคาร เชี่ยวกิจวุฒิกุล (Angkhan Chiewkiwutthikul)[1] และ ประภาส จงสถิตย์วัฒนา (Prabhas Chongstitvatana)[2]
[1,2]ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
angkhan.c@student.chula.ac.th[1], prabhas.c@chula.ac.th[2]

บทคัดย่อ

บทความวิจัยนี้นำเสนอวิธีการจัดตารางงานในระบบปฏิบัติการแบบเวลาจริงบนหน่วยประมวลผลชนิดหลายแกน โดยพัฒนาต้นแบบจาก Micrium μC/OS-III ให้สามารถทำงานบนหน่วยประมวลผลชนิดหลายแกน ได้ทำการทดสอบเปรียบเทียบประสิทธิภาพการทำงานบนหน่วยประมวลผลชนิดแกนเดียวกับหน่วยประมวลผลชนิดสองแกน พบว่ามีประสิทธิภาพของระบบบนหน่วยประมวลผลชนิดสองแกนเพิ่มขึ้น 181 เปอร์เซ็นต์

คำสำคัญ: การจัดตารางงาน, ระบบปฏิบัติการ, หน่วยประมวลผลชนิดหลายแกน.

Abstract

This research developed a prototype of a scheduler in Real-Time Operating Systems on multiple core microprocessors. Micrium μC/OS-III has been modified to support multiple core microprocessors. The experiment is carried out to compare the performance of running tasks on single core and multiple core microprocessors. The result shows that the efficiency is increased by 181 percents on a dual core processor.

Keyword: task scheduling, operating systems, multicore computing.

1. บทนำ

ระบบปฏิบัติการแบบเวลาจริง (Real-Time Operating Systems) เป็นระบบปฏิบัติการที่เน้นเวลาในการทำงานและการตอบสนองการทำงาน ใช้เวลาเป็นตัววัดประสิทธิภาพของระบบ นิยมนำไปใช้ในซอฟต์แวร์แบบระบบฝังตัว (Embedded Systems) ซึ่งใช้อย่างแพร่หลายในหลายอุตสาหกรรม ทั้งอุปกรณ์ในยานพาหนะ เครื่องควบคุมต่างๆ เครื่องใช้ไฟฟ้าทั้งภายในบ้าน ภายในสำนักงานและอุปกรณ์อิเล็กทรอนิกส์ต่างๆ

ปัจจุบันระบบฝังตัวมีแนวโน้มการใช้หน่วยประมวลผลหลายแกน (Multiple Core Microprocessor) เพิ่มมากขึ้น [1] เพราะหน่วยประมวลผลชนิดหลายแกนมีประสิทธิภาพการทำงานที่ดีกว่าหน่วยประมวลผลชนิดแกนเดียว (Single Core Microprocessor) ดังนั้นจึงมีความจำเป็นที่ต้องพัฒนาระบบปฏิบัติการแบบเวลาจริงให้ใช้งานบนหน่วยประมวลผลหลายแกนเช่นกัน

งานวิจัยนี้ได้เลือกใช้ระบบปฏิบัติการ Micrium μC/OS-III [3] เป็นต้นแบบ เพราะเป็นระบบปฏิบัติการที่นิยมและถูกใช้ในหน่วยประมวลผลหลากหลายสถาปัตยกรรม เป็นโอเพนซอร์สสำหรับการศึกษาและยังไม่มีส่วนสนับสนุนการทำงานบนหน่วยประมวลผลหลายแกน

งานวิจัยจึงเสนอวิธีการจัดตารางงานบนหน่วยประมวลผลหลายแกนโดยได้พัฒนาระบบปฏิบัติการ Micrium μC/OS-III ให้สามารถทำงานหน่วยประมวลผลที่มีจำนวนหลายแกน เพื่อเพิ่มประสิทธิภาพการทำงานและให้มีการใช้ทรัพยากรอย่างคุ้มค่า

2. วรรณกรรมที่เกี่ยวข้อง

2.1 หน่วยประมวลผลชนิดหลายแกน

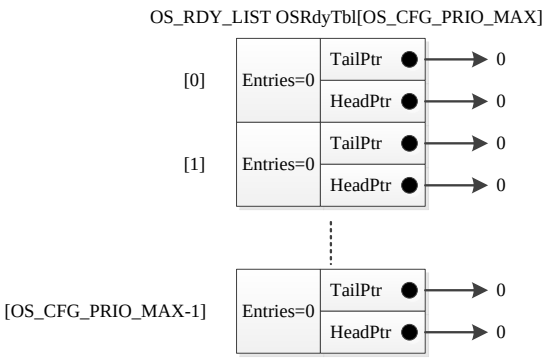
หน่วยประมวลผลแบบหลายแกน [2] คือหน่วยประมวลผลที่มีการรวมหน่วยประมวลผลกลางมากกว่า 2 ตัวขึ้นไป เข้ารวมเป็นแกนของหน่วยประมวลผลกลาง จึงทำให้หน่วยประมวลผลแบบหลายแกนมีการประมวลผลที่มีประสิทธิภาพดีขึ้น สามารถแบ่งสถาปัตยกรรมของหน่วยประมวลผลชนิดหลายแกนเป็นหน่วยประมวลผลหลายแกนชนิดแกนไม่ต่างกัน (Heterogeneous) คือมีลักษณะของแต่ละแกนไม่เหมือนกัน และหน่วยประมวลผลชนิดหลายแกนแบบเหมือนกัน (Homogeneous) คือหน่วยประมวลผลกลางที่แต่ละแกนมีลักษณะเหมือนกัน โดยทุกหน่วยประมวลผลสามารถทำงานในลักษณะที่เหมือนกันได้

งานวิจัยนี้พัฒนาบนหน่วยประมวลผล S2 : A Hypothetical 32-bit Processor Version 3 [7] เป็นหน่วยประมวลผลชนิดสองแกน ที่ถูกจำลองขึ้นเพื่อใช้ในการศึกษา โดยมีลักษณะเป็นหน่วยประมวลผลชนิดสองแกนแบบเหมือนกันและมีโปรแกรมจำลองแบบการทำงาน รวมถึงการจำลองสัญญาณการขัดจังหวะและการตั้งเวลาในหน่วยประมวลผล

2.2 Micrium µC/OS-III

ระบบปฏิบัติการ Micrium µC/OS-III [3] เป็นระบบปฏิบัติการแบบให้สิทธิ์ก่อน (Pre-emptive Real-Time Kernel) เริ่มพัฒนาขึ้นปี ค.ศ. 2009 โดยใช้ภาษา C ในการพัฒนาเป็นหลักและมีการใช้ภาษา Assembly ในบางส่วน

การกำหนดระดับความสำคัญ (Assigning Task Priorities) ใช้ค่าลำดับความสำคัญ งานสำคัญมากจะมีค่าน้อยจะได้ทำงานก่อน ส่วนงานสำคัญน้อยจะมีค่าลำดับความสำคัญมาก จำนวนงานทั้งหมดถูกกำหนดไว้ก่อนเริ่มทำงานต้องกำหนดขนาด Stack เพื่อใช้เก็บ CPU Registers และ Floating-Point Unit (FPU) Registers โครงสร้างข้อมูลเพื่อเก็บข้อมูลต่าง ๆ ของแต่ละส่วนงานว่า Task Control Blocks



ภาพที่ 1: ตาราง Ready List Pointers

การเก็บรายการส่วนงานหรือเรียกว่า รายการสถานะพร้อมทำงาน (The Ready List) ซึ่งมีสองส่วนคือ Bitmap Priority Levels ใช้บอกระดับความสำคัญที่พร้อมทำงานและ ตาราง Ready List Pointers ใช้เก็บตำแหน่ง Pointers ของ Task Control Blocks ของส่วนงานที่มีสถานะพร้อมทำงาน ลักษณะของตาราง Ready List Pointers แสดงในภาพที่ 1 [3]

ในแต่แถวแยกตามลำดับความสำคัญ โดยมีค่า Entries เป็นจำนวนส่วนงานที่มีสถานะพร้อมทำงานในระดับความสำคัญนั้น ๆ มี TailPtr และ HeadPtr เป็น Pointers สำหรับชี้ตำแหน่ง Task Control Blocks ในลิสต์

เครื่องมือจัดการทรัพยากร (Resource Management) หรือจัดการข้อมูลที่ใช้งานร่วมกันระหว่างส่วนงาน สามารถใช้งานเซมาฟอร์ (Semaphores) การปิดและเปิดสัญญาณขัดจังหวะ

2.3 การจัดตารางงาน

การจัดตารางงาน คือการทำการเลือกส่วนงานเพื่อทำงานตามความเหมาะสมหรือตามลำดับความสำคัญของส่วนงาน เพื่อให้ระบบมีประสิทธิภาพ โดยการจัดตารางงานในระบบปฏิบัติการ Micrium µC/OS-III ใช้การจัดตารางงานแบบให้สิทธิ์ก่อน (Pre-Emptive Scheduling) และใช้การจัดตารางงานแบบวนรอบ (Round-Robin Scheduling) ทำงานในส่วนงานที่มีระดับความสำคัญของส่วนงานเท่ากัน ซึ่งการจัดตารางงานทั้งสองแบบในระบบปฏิบัติการนี้ยังไม่สนับสนุนการทำงานบนหน่วยประมวลผลชนิดหลายแกน

การจัดตารางงานสำหรับทำงานบนหน่วยประมวลผลชนิดหลายแกน จึงต้องเพิ่มส่วนการตัดสินใจสำหรับเลือกส่วนงานว่าจะใช้แกนประมวลผลใด [4,5,6]

Diana Bautista [4] มีการทำการเลือกแกนหน่วยประมวลผลจากการพิจารณาค่าจาก λ คือค่าอัตราส่วนงานต่อเวลา หรือ 1/λ เป็นเวลาที่หนึ่งส่วนงานใช้บนแกนหน่วยประมวลผล โดยพิจารณาเลือกแกนหน่วยประมวนผลที่มีการถูกใช้งานน้อยกว่า วัตถุประสงค์เพื่อการประหยัดพลังงานบนหน่วยประมวลผล

James Mistry [5] ทำการจัดตารางงานจากความสัมพันธ์แกนประมวลผล (Core Affinity) และระดับความสำคัญของส่วนงาน โดยให้เลือกส่วนงานที่มีระดับความสำคัญสูงสุดและมีความสัมพันธ์กับแกนประมวลผลก่อน

Nicolas Navet [6] ทำการเลือกแกนหน่วยประมวลผลโดยใช้ค่าการใช้ประโยชน์ของแกนหน่วยประมวลผล (CPU Utilization Rate) โดยพิจารณาจากกลุ่มของส่วนงานที่ผ่านการแบ่งแยกการพิจารณาความเกี่ยวข้องกับแกนหน่วยประมวลผลแล้ว

ทั้ง 3 งานวิจัยแสดงการจัดการตารางงานที่แตกต่างกันและแสดงให้เห็นถึงความสำคัญของตัวจัดตารางงานต่อระบบ

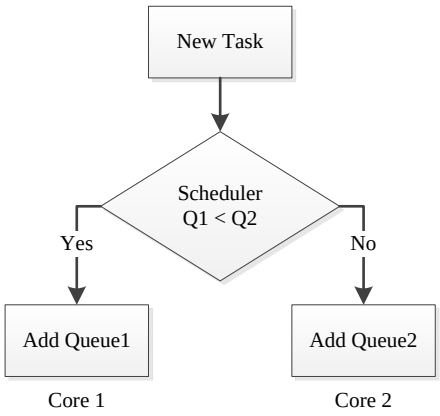
3. วิธีการดำเนินการวิจัย

วิธีการดำเนินการวิจัยประกอบด้วย 3 ขั้นตอนดังต่อไปนี้ 1. การออกแบบ, 2. การพัฒนาระบบ, 3. การทดสอบระบบ

3.1 การออกแบบ

ออกแบบการจัดตารางงานให้ตัวจัดการตารางงานทำหน้าที่เลือกส่วนงานที่เหมาะสมทำงานบนแต่ละแกนหน่วยประมวลผล แนวความคิดหลักคือกระจายงานให้ทั่วถึง ไม่ให้มีแกนใดอยู่ว่าง ข้อสมมุติฐานเบื้องต้นคือเมื่อให้งานกับแกนใดไปแล้วจะไม่มีการย้ายงานไปแกนอื่น ทั้งนี้เพื่อให้ระบบตอบสนองในเวลาจริงได้เช่นเดิม จำนวนงานทั้งหมดถูกกำหนดไว้ก่อนเริ่มงานแล้ว ตามข้อกำหนดเดิมของระบบปฏิบัติการ Micrium μC/OS-III

การกระจายงานต้องทราบภาระงานของทุกแกนก่อน ตัววัดที่ใช้บอกภาระงานคือ จำนวนงานในรายการพร้อมทำงานของแกนนั้น ๆ จากจำนวนงานที่กำลังทำงานอยู่แต่ละแกนหน่วยประมวลผล โดยจะเลือกใส่ในแกนหน่วยประมวลผลที่มีจำนวนงานน้อยกว่า มีการเพิ่มรายการสถานะพร้อมทำงานตามจำนวนแกนที่เพิ่มขึ้นหรือแยกรายการสถานะพร้อมทำงาน เป็นของแต่ละแกนหน่วยประมวลผลดังภาพที่ 2 แสดงตัวอย่างการทำงานของตัวจัดตารางงานที่ทำงานบนหน่วยประมวลผลสองแกน

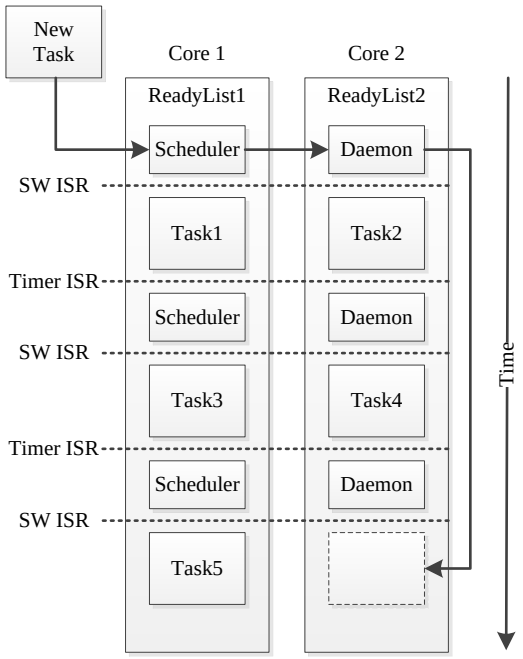


ภาพที่ 2: การทำงานของตัวจัดตารางงาน

ตัวจัดการตารางทำงานบนแกนหน่วยประมวลผลหลักมีหน้าที่จัดการส่วนงานเพียงตัวเดียวและให้การทำงานมีการใช้งานสัญญาณขัดจังหวะน้อยที่สุด ซึ่งลดความซับซ้อนของการทำงานและเพิ่มความมั่นคง จึงสร้างตัวจัดตารางงานเป็นส่วนงานชื่อ Scheduler Task ให้ทำงานบนแกนประมวลผลหลักเท่านั้น ทำหน้าที่เพิ่มส่วนงานไปยังแต่ละรายการสถานะพร้อมทำงานและสร้างส่วนงานที่รับการมอบงานจากแกนหลักซึ่งทำงานเป็นระยะอยู่ตลอดเวลาอยู่บนแกนอื่น ๆ ชื่อ Daemon Task ในแต่ละรายการสถานะพร้อมทำงานของแต่ละแกนหน่วยประมวลผลอื่น เพื่อรับคำสั่งเพิ่มส่วนงานบนแกนหน่วยประมวลผลนั้น ๆ

เนื่องจากแต่ละแกนหน่วยประมวลผลมีรายการสถานะพร้อมทำงานของตนเอง การจัดรายการส่วนงานทำหน้าที่เพียงสลับการทำงานตามรายการส่วนงานในรายการสถานะพร้อม

ทำงาน โดย Scheduler Task และ Daemon Task จะสลับเป็นส่วนงานงานอื่นด้วยสัญญาณขัดจังหวะชนิดซอฟต์แวร์และส่วนงานอื่นจะใช้สัญญาณขัดจังหวะชนิดนับเวลาเป็นตัวทำงานในส่วนการสลับส่วนงานดังภาพที่ 3



ภาพที่ 3: แสดงการขั้นตอนการเพิ่มส่วนงาน

3.2 การพัฒนา

การพัฒนาเริ่มจากการย้ายระบบปฏิบัติการ Micrium µC/OS-III ให้ทำงานบนหน่วยประมวลผล S2 โดยยังไม่ทำการแก้ไขส่วนการจัดตารางงานและยังไม่เริ่มการทำงานแกนที่สอง เพื่อทำการพัฒนาส่วน Context Switch หรือส่วนการสลับเปลี่ยนส่วนงานเป็นส่วนงานใหม่ เริ่มเมื่อได้รับสัญญาณการขัดจังหวะเกิดขึ้นและต้องการเปลี่ยนส่วนงานที่กำลังทำงาน เมื่อได้รับสัญญาณขัดจังหวะจะทำการบันทึกตำแหน่ง Stack Pointer ค่า CPU Registers และ Floating-Point Unit (FPU) Registers ในขณะนั้นลง Task Control Block ของส่วนงานที่กำลังทำงาน จากนั้นจะทำการเรียกข้อมูลจาก Task Control Block ของส่วนงานใหม่ขึ้นมาทำงานแทน

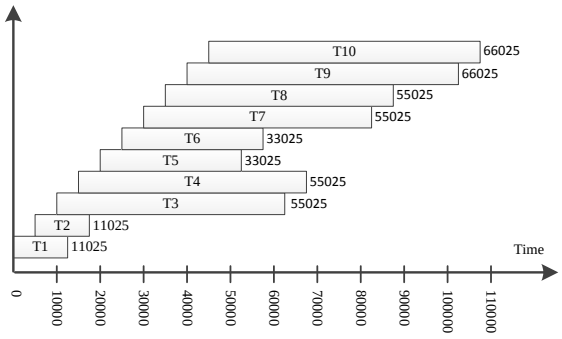
จากนั้นทำการเพิ่มรายการสถานะพร้อมทำงานตามจำนวนแกนที่เพิ่มขึ้นมา ซึ่งในงานวิจัยนี้หน่วยประมวลผล S2 เป็นหน่วยประมวลผลสองแกน จึงทำการเพิ่มแค่หนึ่งรายการ แล้วทดสอบการทำงานโดยสร้างส่วนงานมา 4 ส่วนงาน แบ่งส่วนงานให้สองรายการสถานะพร้อมทำงานเท่ากัน เพื่อทดสอบการทำงานของแกนที่สอง โดยให้แต่ละแกนทำงานแยกกันโดยทำงานตามรายการสถานะพร้อมทำงานของตน

ทำการพัฒนา Scheduler Task และ Daemon Task โดยให้อยู่ในรายการสถานะพร้อมทำงานบนแกนหลักและแกนที่สองตามลำดับ ใช้ Semaphore ในการป้องกันข้อมูลตำแหน่ง Task Control Block ที่จะเพิ่มเข้ารายการสถานะพร้อมทำงานของแกนที่สอง โดยป้องกันการใช้งานร่วมกันระหว่าง Scheduler Task และ Daemon Task

3.2 การทดสอบ

ทำการทดสอบการจัดตารางงานโดยเปรียบเวลาที่ใช้ในการทำงานบนหน่วยประมวลผลชนิดแกนเดียวและการทำงานบนหน่วยประมวลผลชนิดสองแกน จากชุดทดลอง 5 ชุด โดยแต่ละชุดมีส่วนงานจำนวน 10 ส่วนงาน

โดยแต่ละส่วนงานให้ทำการบวกนับเลขในจำนวนที่แตกต่างกัน เพื่อจำลองเวลาทำงานของแต่ละส่วนงานไม่เท่ากัน และให้เริ่มการทำงานของแต่ละส่วนงานไม่พร้อมกัน เพื่อจำลองสถานการณ์การทำงานเป็นจำนวน 5 เหตุการณ์



ภาพที่ 4: ตัวอย่างชุดทดลองที่ 1

จากภาพที่ 4 ตัวอย่างชุดทดลองที่ 1 แสดงระยะเวลาทำงานและการเริ่มทำงานของแต่ละส่วนงาน จำนวนตัวเลขคือจำนวนสัญญาณนาฬิกาของระบบ โดยส่วนงานที่ 1 และ 2 มีเวลาการทำงานที่ 11025 ส่วนงานที่ 3, 4, 7 และ 8 มีเวลาการทำงานที่

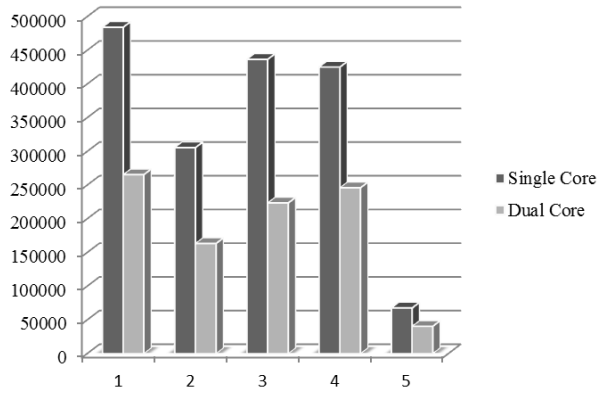
55025 ส่วนงานที่ 5 และ 6 จะมีเวลาการทำงานที่ 33025 ส่วนงานที่ 9 และ 10 จะมีเวลาการทำงานที่ 66025 แต่ละส่วนงานจะเกิดห่างกันที่ 5000

4. ผลการดำเนินงาน

ผลการทดสอบเป็นเลขจำนวนสัญญาณนาฬิกาที่ใช้ไปของส่วนงานทั้งหมดในแต่ละชุดทำงานและทำการเปรียบเทียบผลระยะเวลาการทำงานบนหน่วยประมวลผลแกนเดียวต่อการทำงานบนสองแกน ดังตารางที่ 1 และภาพที่ 5

ตารางที่ 1: ผลการทดสอบ

ลำดับ	แกนเดียว	สองแกน	เปรียบเทียบผล
1	484212	265521	182.36
2	305687	163449	187.02
3	436648	223595	195.28
4	425227	246135	172.76
5	67760	40523	167.21



ภาพที่ 5: กราฟแท่งเปรียบเทียบเวลาในการทำงาน

นำผลการเปรียบเทียบการทำงานของหน่วยประมวลผลแกนเดียวและสองแกนของทั้ง 5 ชุดมาเฉลี่ยกัน ค่าที่ได้เป็นประสิทธิภาพของการทำงานของหน่วยประมวลสองแกนเปรียบเทียบกับหน่วยประมวลผลแกนเดียวมีค่าเพิ่มขึ้นเป็น 181 เปอร์เซ็นต์

5. สรุป

วิธีการจัดตารางงานที่นำเสนอสามารถทำงานบนหน่วยประมวลผลหลายแกนและทำงานได้มีประสิทธิภาพเพิ่มขึ้น 181 เปอร์เซ็นต์จากการเปรียบเทียบการทำงานบนหน่วยประมวลผลแกนเดียวและหน่วยประมวลผลสองแกน ซึ่งค่าที่ได้ยังห่างจากค่าในอุดมคติที่ 200 เปอร์เซ็นต์ เพราะยังมีการใช้เวลาในกระบวนการจัดตารางงานและการสลับเปลี่ยนส่วนงาน

การพัฒนาต่อไปควรทดลองกับหน่วยประมวลผลที่มีจำนวนแกนมากขึ้น ลดเวลาในกระบวนการจัดตารางงานและพัฒนาส่วนการส่งข้อมูลการเพิ่มส่วนงานจาก Scheduler Task ถึง Daemon Task ที่มีจำนวนเพิ่มขึ้นหรือส่วนการควบคุมการใช้งานข้อมูลร่วมกันระหว่างแกน

6. เอกสารอ้างอิง

[1] D. Geer, “Industry Trends: Chip Makers Turn to Multicore Processors,” *IEEE Computer Society*, vol. 38, no. 5, pp. 11-13, 2005.

[2] Atsushi Hasegawa, “Renesas' Multi-Core Technology,” *http://www.renesas.com/products/mpumcu/multi_core/child/multicore.jsp*.

[3] Micruim, Jean J. Labrosse, “µC/OS-III TM The Real-Time Kernel User’s Manual,” Micrium Press: 2010.

[4] Diana Bautista, Julio Sahuquillo, Houcine Hassan, Salvador Petit, Jos´e Duato, “A Simple Power-Aware Scheduling for Multicore Systems when Running Real-Time Applications,” *Parallel and Distributed Processing (IPDPS 2008), IEEE International Symposium*, pp. 1-7, 2008.

[5] James Mistry, Matthew Naylor and Jim Woodcock, “Adapting FreeRTOS for Multicore: an Experience Report,” John Wiley & Sons, Ltd: 2010.

[6] Nicolas Navet, Aurélien Monot, Bernard Bavouxy, Françoise Simonot-Lion, “Real Multi-source and multicore automotive ECUs - OS protection mechanisms and scheduling,” *International Symposium on Industrial Electronics (ISIE2010)*, 2010.

[7] “S2: A Hypothetical 32-bit Processor Version 3,” *www.cp.eng.chula.ac.th/faculty/pjw/project/s2/s23.htm*