การสร้างเครื่องควบคุมเสียงรบกวนแบบไวงาน

(Implementation of an Active Acoustic Noise Control)

โดย

นายธนสาร ศรีโคตร รหัส B5304691

นายพัชรพล ช่างเหล็ก รหัส B5321544

รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงงานวิศวกรรมโทรคมนาคม

หลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546

สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2556

การสร้างเครื่องควบคุมเสียงรบกวนแบบไวงาน(Implementation of an Active Acoustic Noise Control)

คณะกรรมการสอบโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร.สมศักดิ์ วาณิชอนันต์ชัย)

ที่ปรึกษาโครงงาน

(ผู้ช่วยศาสตราจารย์ ดร. ชุติมา พรหมมาก)

กรรมการ

(อาจารย์ ดร.ธนเสฏฐ์ ทศดีกรพัฒน์)

กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นับรายงานโครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม รายวิชา 427499 โครงงานวิศวกรรมโทรคมนาคมประจำปีการศึกษา 2556

โครงงาน การสร้างเครื่องควบคุมเสียงรบกวนแบบไวงาน

(Implementation of an Active Acoustic Noise Control)

จัดทำโดย	นายธนสาร ศรีโคตร รหัส B5304691
	นายพัชรพล ช่างเหล็ก รหัส B5321544
อาจารย์ที่ปรึกษาโครงงาน	ผู้ช่วยศาสตราจารย์ ดร.สมศักดิ์ วาณิชอนันต์ชัย
สาขาวิชา	วิศวกรรมโทรคมนาคม
ภาคการศึกษาที่ 3 / 2556	

บทคัดย่อ

(Abstract)

โครงงานเล่มนี้ต้องการสร้างเครื่องกำจัดเสียงรบกวนแบบไวงานซึ่งใช้ไมโครโปรเซสเซอร์ TMS320VC5416 ซีพียูสมรรถนะสูงนี้ทำให้เราสามารถสร้างเครื่องกำจัดเสียงรบกวนแบบไวงานโดยใช้สัมประสิทธิ์วงจรกรองที่มีขนาด 32 บิต ใช้จำนวนสัมประสิทธิ์ที่มากขึ้น รวมทั้งสามารถใช้อัตราการสุ่มตัวอย่างที่เร็วขึ้น จากการทดลองแสดงให้เห็นว่าโดยการใช้โครงสร้าง Filtered-X LMS ชนิดป้อนกลับแบบง่ายๆสามารถลดทอนเสียงรบกวนช่วง 100 Hz ถึง 300 Hz ลงได้

กิตติกรรมประกาศ

(Acknowledgement)

โครงงานเล่มนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องจากได้รับความช่วยเหลืออย่างดียิ่ง จากบุคคลและกลุ่มบุคคลต่าง ๆ ได้แก่

ผู้ช่วยศาสตราจารย์ ดร.สมศักดิ์ วาณิชอนันต์ชัย อาจารย์ประจำสาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี อาจารย์ที่ปรึกษาโครงงาน ผู้ที่เป็นเจ้าของงานวิจัยการสร้างเครื่องควบคุมเสียงรบกวนแบบไวงาน (Implementation of an Active Acoustic Noise Control)ที่ให้ความช่วยเหลือในการให้แนวคิด ให้คำปรึกษา แนะนำ ชี้แนะข้อบกพร่องที่ข้าพเจ้ามองข้าม และให้กำลังใจ ตลอดจนฝึกฝนและสนับสนุนข้าพเจ้าให้มีความสามารถในการทำโครงงานจนสามารถนำเสนอผลงานให้เป็นที่รู้จักและยอมรับได้

ขอขอบคุณอาจารย์ประจำสาขาวิศวกรรมโทรคมนาคมทุกท่าน ที่สั่งสอนให้ความรู้มาโดยตลอด และยังเป็นแบบอย่างที่ดีในการใช้ชีวิตในรั้วมหาวิทยาลัย ขอขอบคุณพี่ๆ นักศึกษาปริญญาโท วิศวกรรมโทรคมนาคมทุกท่าน ที่คอยแนะนำความรู้ต่างๆ

สำหรับคุณงามความดีอันใดที่เกิดจากโครงงานฉบับนี้ ข้าพเจ้าขอมอบให้กับบิดา มารดาซึ่งเป็นที่รักและเคารพยิ่ง ผู้คอยห่วงใย ให้โอกาสให้กำลังใจและให้การสนับสนุนทางการศึกษามาโดยตลอด ตลอดจนครูอาจารย์ผู้สอนที่เคารพทุกท่านที่ได้ประสิทธิ์ประสาทวิชาความรู้และถ่ายทอดประสบการณ์ที่ดีให้แก่ข้าพเจ้าตลอดมา จนทำให้ข้าพเจ้าประสบความสำเร็จมาจนถึงบัดนี้

นายธนสาร ศรีโคตร

นายพัชรพล ช่างเหล็ก

สารบัญ

สารบัญ(ต่อ)

สารบัญรูป

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

มลพิษทางเสียง (Noise pollution) คือเสียงที่ไม่อยากได้ยิน แต่ก็ไม่สามารถหลีกเลี่ยงได้ เป็นปัญหาสำคัญที่เกิดขึ้นจากความก้าวหน้าทางเทคโนโลยีและความเจริญของสังคมเมืองโดยเฉพาะในเมืองใหญ่ๆ และในโรงงานอุตสาหกรรม ยกตัวอย่างเช่น เสียงเครื่องตัดหญ้าที่บ้าน เสียงรถยนต์ รถมอเตอร์ไซค์ในย่านการจราจรหนาแน่น เสียงเครื่องยนต์กลไกในที่ทำงาน เป็นต้น ถึงแม้ว่ามลพิษทางเสียงนี้จะไม่ได้เป็นอันตรายต่อชีวิตโดยตรงแต่ก็มีผลรบกวนต่อจิตใจ ก่อให้เกิดความรำคาญ ความเครียด ส่งผลทำให้ประสิทธิภาพในการทำงานลดลง โดยเฉพาะในงานที่ต้องการสมาธิค่อนข้างสูง นอกจากนี้ยังเป็นปัจจัยสำคัญที่มีผลต่อความปลอดภัยในการปฏิบัติงานในโรงงานอุตสาหกรรม

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างเครื่องกำจัดเสียงรบกวนแบบไวงาน ที่สามารถทำงานได้จริง
2. เพื่อศึกษาทักษะในการพัฒนาโปรแกรมประมวลผลสัญญาณเวลาจริงโดยใช้บอร์ด TMS320C5416

1.3 ขอบเขตการดำเนินงาน

1. พัฒนาและสร้างหูฟังที่สามารถกำจัดเสียงรบกวนแบบไวงาน (ANC Headsets) โดยใช้บอร์ด TMS320C5416 และไมโครโฟน กับหูฟังที่มีขายอยู่ทั่วไปในท้องตลาด
2. ทดสอบความสามารถในการลดทอนสัญญาณเสียงรบกวนความถี่เดียว ณ ความถี่ต่างๆ และเวลาที่ใช้ในการลู่เข้า

1.4 ขั้นตอนการดำเนินงาน

กิจกรรม	พ.ศ. 2557			
	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1.ศึกษา ค้นคว้าหาข้อมูล	←	→		
2.เขียนโครงงานและเสนอกับอาจารย์		←	→	
3.จัดหาอุปกรณ์ที่ใช้ในโครงงานนี้		↔		
4.ศึกษาภาษาซี เขียนโปรแกรมเครื่องกำจัดเสียงแบบไวงาน		←	→	
5.ทำการเชื่อมต่ออุปกรณ์กับบอร์ด TMS320C5416 ที่ลงโปรแกรมเครื่องกำจัดเสียงแบบไวงาน			↔	
6.ทำการทดสอบชิ้นงาน			↔	
7.สรุปผลการทดลองและเขียนรายงาน			↔	
8.นำเสนอโครงงาน				↔

ตารางที่ 1.1 ขั้นตอนการดำเนินงาน

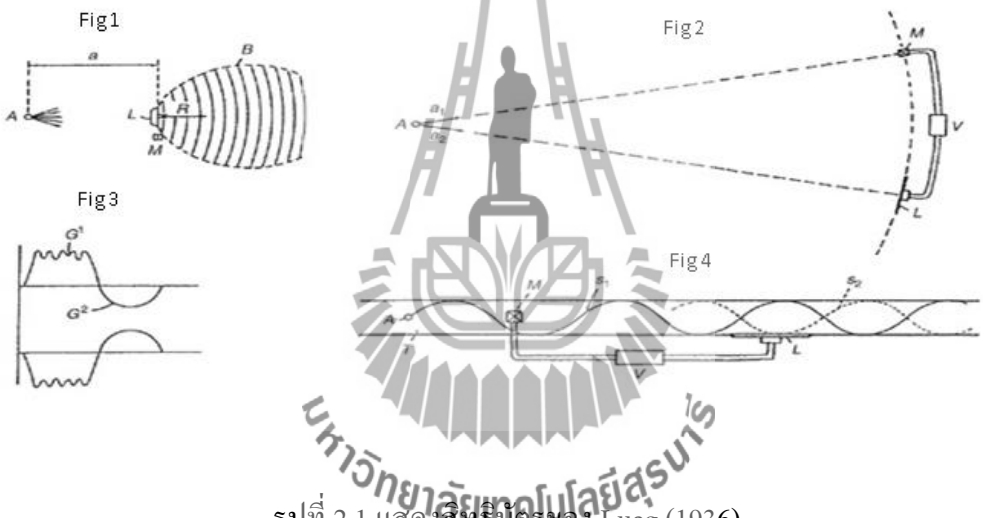
1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถสร้างเครื่องกำจัดเสียงแบบไวงานได้
2. สามารถนำไปใช้ได้จริงและเป็นประโยชน์ต่อผู้ปฏิบัติงานในสภาพแวดล้อมที่มีมลพิษทางเสียง
3. มีความรู้ความสามารถในการเขียนภาษาซีที่สูงขึ้น

บทที่ 2
ทฤษฎีพื้นฐาน

2.1 กล่าวนำ

แนวคิดเรื่องการสร้างสัญญาณเสียงเพื่อหักล้างเสียงรบกวนได้รับการเสนอและจดสิทธิบัตรครั้งแรกโดย Lueg (1936)ดังแสดงในรูปที่ 2.1 จากรูป Fig 1 ในสิทธิบัตรแสดงถึงไมโครโฟน M ตรวจวัดสัญญาณเสียงในท่อ ส่งเข้าวงจรควบคุม เพื่อนสร้างสัญญาณเสียงออกที่ลำโพง L ให้มีเฟสตรงข้ามเพื่อหักล้างกับสัญญาณเดิม Fig 3 แสดงให้เห็นว่าสัญญาณเสียงไม่จำเป็นต้องเป็นสัญญาณรูปคลื่นซายน์ สำหรับ Fig 2 และ Fig 4 แสดงให้เห็นถึงแนวคิดในการหักล้างเสียงรบกวนในอากาศ ในอากาศ 3 มิติ

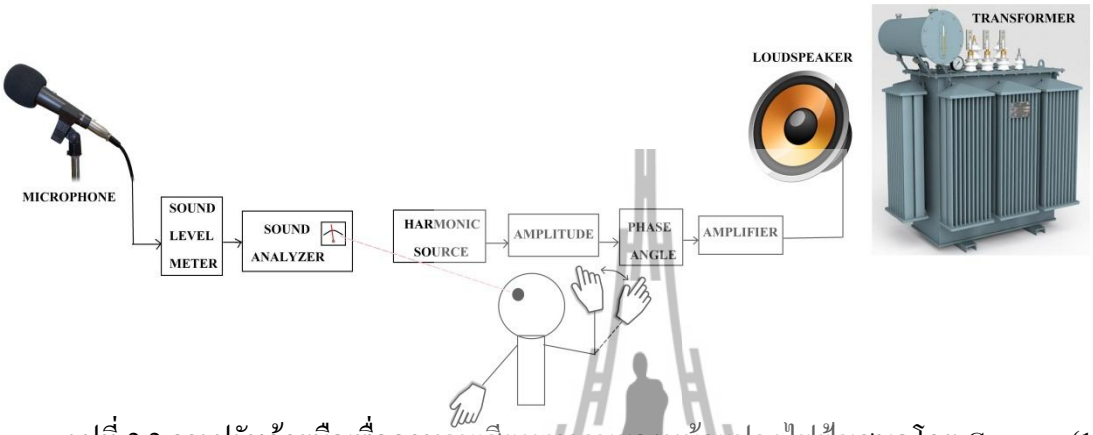


รูปที่ 2.1 แสดงสิทธิบัตรของ Lueg (1936)

ต่อมาในปี 1953 Harry Olson และ Everet May ได้ตีพิมพ์บทความเรื่อง “Electronic sound absorber ” ซึ่งได้เสนอความคิดให้ติดตั้งไมโครโฟนใกล้ๆกับลำโพงที่ทำหน้าที่เป็นที่แหล่งกำเนิดสัญญาณทุติยภูมิ และติดตั้งอุปกรณ์ทั้งสองนี้ที่ด้านหลังของที่นั่งบริเวณศีรษะ วงจรขยายจะรับสัญญาณจากไมโครโฟนและส่งออกไปยังลำโพงเพื่อหักล้างกับสัญญาณรบกวนและสร้าง “zone of quiet” ณ บริเวณศีรษะ ความสำคัญของผลงานชิ้นนี้คือ วิสัยทัศน์ที่จะนำแนวคิดนี้ไปประยุกต์ใช้กับที่นั่งของผู้โดยสารในเครื่องบินหรือรถยนต์

ในเวลาขนาดเดียวกัน William Conover (1956) ได้พยายามกำจัดเสียงรบกวนที่เกิดจากหม้อแปลงไฟฟ้าขนาดใหญ่ซึ่งจะมีความถี่ของสัญญาณรบกวนเท่ากับความถี่พื้นฐาน และฮาร์โมนิคของความถี่พื้นฐาน ในงานวิจัยของ Conover นี้ไม่ต้องใช้สัญญาณอ้างอิงที่มาจากแหล่งกำเนิดสัญญาณรบกวนปฐมภูมิ เนื่องจาก

สัญญาณรบกวนมีความถี่ที่แน่นอนและมีลักษณะเป็นสัญญาณรายคาบ ในรูปที่ 2.2 เครื่องวัดขนาด (Sound Level Meter) จะวัดขนาดของสัญญาณ และเครื่องวิเคราะห์สัญญาณ (Sound Analyzer) จะวัดเฟสของสัญญาณรบกวนจากหม้อแปลงไฟฟ้า ณ ความถี่ต่างๆ แหล่งกำเนิดสัญญาณฮาร์โมนิค (Harmonic Source) จะสร้างสัญญาณ Full wave rectified ผ่านวงจรกรองผ่านแถบ ซึ่งสามารถปรับขนาดและเฟสด้วยมือได้เป็นครั้งคราว เพื่อแก้ไขผลของลมและอุณหภูมิที่มีต่อการเดินทางของสัญญาณเสียง ความสำคัญของผลงานของ Conover คือแสดงให้เห็นถึงศักยภาพในการนำระบบควบคุมอัตโนมัติมาประยุกต์ใช้ และความเป็นไปได้ในการใช้ลำโพงและไมโครโฟนหลายๆ ชุด



รูปที่ 2.2 การปรับด้วยมือเพื่อควบคุมเสียงรบกวนของหม้อแปลงไฟฟ้าเสนอโดย Conover (1956) (ที่มา Elliott, 1993)

ผลงานทั้งสามชิ้นที่กล่าวมานี้ แม้ว่าจะมีความพยายามทำการทดลอง แต่ก็ยังไม่ประสบความสำเร็จและยังไม่สามารถใช้งานได้จริง หลังจากช่วงทศวรรษที่ 50 งานวิจัยทางด้าน ANC ก็เงียบหายไป จนกระทั่งปี 1975 (Kido ,1975) เมื่อเริ่มที่การพัฒนาและใช้งานเทคนิคการประมวลผลสัญญาณเชิงเลขหรือ Digital Signal Processing อย่างรวดเร็ว สาเหตุที่การค้นคว้าเรื่อง ANC เงียบหายไปกว่า 20 ปี ไม่ใช่เป็นเพราะว่าขาดความรู้หรือความมานะพยายาม แต่สิ่งที่ขาดไปก็คือเทคโนโลยี ในการสร้าง ANC ชนิดป้อนไปหน้า เพื่อให้ความดังสัญญาณรบกวนชนิด Pure tone เบาลง 20 dB วงจรอิเล็กทรอนิกส์จำเป็นต้องปรับตัวตามสภาพแวดล้อม และสร้างสัญญาณ Pure tone ด้วยความละเอียดของขนาด ± 0.6 dB และของเฟส ±5 องศา ซึ่งความละเอียดขนาดนี้ไม่สามารถสร้างได้โดยใช้วงจรอิเล็กทรอนิกส์ชนิดแอนาลอก ด้วยเทคโนโลยีในช่วงปี 1936 - 1970

2.2 Time invariant system

ระบบเวลาไม่เปลี่ยนแปร เป็นระบบที่มีความสัมพันธ์ระหว่างสัญญาณอินพุตกับสัญญาณเอาต์พุตที่ไม่เปลี่ยนแปรไปตามเวลา เนื่องจากคุณสมบัติของระบบไม่เปลี่ยนแปลงไปตามเวลาหรือไม่ขึ้นกับเวลา ดังนั้นไม่ว่าจะป้อนสัญญาณอินพุตเข้าสู่ระบบเวลาใดก็ตาม ผลตอบสนองที่ได้จะเหมือนเดิมเพียงแต่จะมีการเลื่อนเวลาออกไปและสอดคล้องกับจังหวะเวลาที่ป้อน ยกตัวอย่างเช่น

ถ้าระบบมีสัญญาณอินพุตเท่ากับ $x_1(n)$ ระบบจะให้สัญญาณเอาต์พุตมีค่าเป็น $y_1(n)$

ถ้าระบบมีสัญญาณอินพุตที่มีการเลื่อนในทางเวลาจากเดิม x(n) เลื่อนออกไปเป็น $x_1(n-n_0)$ สัญญาณเอาต์พุตที่ได้ก็จะมีการเลื่อนจาก y(n) ไปเป็น $y_1(n-n_0)$ ซึ่งมีการเลื่อนทางเวลาที่สอดคล้องกันกับสัญญาณอินพุต

ระบบที่เป็นเชิงเส้น (linear system) และเป็นระบบเวลาไม่เปลี่ยนแปร (time invariant system) หรือ shift invariant system จะเป็นระบบที่ง่ายต่อการประมวลผลในทางคณิตศาสตร์ ซึ่งเป็นระบบที่มีประโยชน์มากและมีการประยุกต์ใช้งานกว้างขวาง ระบบดังกล่าวข้างต้นมักจะเรียกรวมว่า ระบบเชิงเส้นเวลาไม่เปลี่ยนแปร (linear time invariant systems: LTI system หรือ linear shift invariant system: LSI systems)

2.3 Discrete-Time Convolution

ลำดับอินพุท x[n] ใดๆสามารถแสดงได้ด้วยผลรวมแบบเชิงเส้นของชุดแถวสุ่มตัวอย่างขนาด 1 หน่วย (unit sample sequence) หรือชุดแถวสัญญาณพัลส์ (impulse) ที่มีการเลื่อนทางเวลาทั้งในแบบล้าหลัง (delayed) และแบบก้าวหน้า (advanced) ดังนี้

$$X(n) = \sum_{k=-\infty}^{\infty} x(k)\delta(n-k) \qquad (2.1)$$

โดยที่ x(k) คือ ลำดับสัญญาณที่ป้อนที่ลำดับ k

δ(n-k) คือ สัญญาณพัลส์ที่มีการเลื่อนทางเวลาที่ k

x(n) คือ ผลรวมแบบเชิงเส้นของสัญญาณ

เราทราบว่าผลตอบสนองของระบบ LTI ต่อสัญญาณอินพุต $\mathrm{x(k)}\delta(n-k)$ จะได้เป็น $\mathrm{x(k)}h(n-k)$ ดังนั้นลำดับเอาต์พุต y[n] จากการป้อนจะเป็น

$$y(n) = \sum_{k=-\infty}^{\infty} \mathrm{x(k)}h(n-k) \quad (2.2)$$

โดยที่ x(k) คือ ลำดับสัญญาณที่ป้อนที่ลำดับ k

h(n-k) คือ ผลตอบสนองอิมพัลส์ที่มีการเลื่อนทางเวลาที่ k

y(n) คือ ผลรวมแบบเชิงเส้นของสัญญาณ

$$y(n) = \sum_{k=-\infty}^{\infty} \mathrm{x(n-k)}h(k) \text{หรือ} \quad (2.3)$$

โดยที่ x(n-k) คือ ลำดับสัญญาณที่ป้อนที่มีการเลื่อนทางเวลาที่ k

h(k) คือ ผลตอบสนองอิมพัลส์ที่ k

x(n) คือ ผลรวมแบบเชิงเส้นสัญญาณ

2.3.1 Finite impulse response (FIR system)

ผลตอบสนองอิมพัลส์สำหรับระบบเวลาต่อหน่วยแบบมีเหตุ (Casual system) เป็นระบบที่สัญญาณเอาต์พุตจะขึ้นอยู่กับสัญญาณอินพุตในอดีตและสัญญาณอินพุตในเวลาปัจจุบันเท่านั้นและสัญญาณเอาต์พุตยังไม่ขึ้นกับสัญญาณอินพุตในอนาคตอีกด้วย จะได้ว่า

$$y(n) = \sum_{k=0}^{M-1} h(k)\,\mathrm{x(n-k)} \text{ดังนั้น} \quad (2.4)$$

โดยที่ x(n-k) คือ ลำดับสัญญาณที่ที่ป้อนมีการเลื่อนทางเวลาที่ k

h(k) คือ ผลตอบสนองอิมพัลส์ที่ k ; h(k) = 0 k<0 and k$\geq$M

X(n) คือ ผลรวมแบบเชิงเส้นสัญญาณ

2.4 Least Mean Square Method

Least Mean Square: LMS ประกอบด้วยสองกระบวนการ คือกระบวนการกรองสร้างสัญญาณและประมาณความผิดพลาด กับ กระบวนการการปรับที่มีหน้าที่ปรับ filter tap weights โดยอัตโนมัติ ต่อไปนี้คือคำนิยามและตัวแปรที่จะใช้ในการอธิบายการทำงานของ LMS

2.4.1 การทำงานกระบวนการแรก กระบวนการกรองสร้างสัญญาณและประมาณความผิดพลาด

Filter output:

$$y(n) = \sum_{k=0}^{M-1} w(k)u(n-k) \quad (2.5)$$

Estimation error: $e(n) = d(n) - y(n)$ (2.6)

โดยที่ u(n-k) คือ ลำดับสัญญาณที่ป้อนที่มีการเลื่อนทางเวลาที่ k

w(k) คือ สัมประสิทธิ์ตัวกรองที่ k ของวงจรกรอง FIR $\hat{H}(z)$

$h(k) = 0 \quad k<0 \text{ and } k \geq M$

y(n) คือ ผลรวมแบบเชิงเส้นสัญญาณ

d(n) คือ สัญญาณที่ต้องการ

e(n) คือ ค่าความผิดพลาดของสัญญาณ

การทำงานกระบวนการสองกระบวนการการปรับที่มีหน้าที่ปรับ filter tap weights โดยอัตโนมัติ

2.4.2 LMS filter coefficient adjustment

$$w(n+1) = w(n) + \mu e(n)u(n) \quad (2.7)$$

โดยที่ w(n) คือ สัมประสิทธิ์ตัวกรองที่ n ของวงจรกรอง FIR $\hat{H}(z)$

u(n) คือ ลำดับสัญญาณที่ป้อน

e(n) คือ ค่าความผิดพลาดของสัญญาณ

μ คือ ขนาดของ Step size

ค่า μเป็นตัวหลักในการขับเคลื่อนการปรับ coefficient ของ filter

$$0 < \mu < \frac{2}{\lambda\, max} \tag{2.8}$$

$$\mu = \frac{\lambda\, max}{10}$$

นอกจากนี้ได้มีการเสนอให้ค่า

โดยที่ $\lambda\, max$ คือ eigenvalue ของ autocorrelation matrix $E[u(n)^H u(n)]$

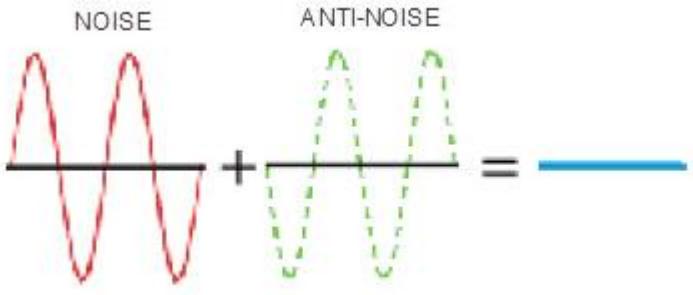
$$\lambda\, max < \sum_{i=1}^{M} \lambda\, i$$

$$= \mathbf{E}[\mathbf{u(n)}^{\mathbf{H}}\mathbf{u(n)}] \tag{2.9}$$

โดยที่ u(n) คือ ลำดับสัญญาณที่ป้อน

2.5 Active Feedback ANC

การควบคุมเสียงรบกวน (ANC) เป็นชุดที่ใช้งานอย่างกว้างขวางสำหรับเทคนิคการลดทอนเสียงรบกวน เอเอนซีเทคนิคสามารถลดทอนเสียงรบกวนด้วยการเพิ่มของสัญญาณ “anti-noise” ที่มีเฟสตรงข้ามกับเสียงรบกวน 180 องศาดีกรี ดังแสดงในรูป 2.3



รูปที่ 2.3 หารปรับตัวยกเลิกเสียงรบกวน

2.5.1 เครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMS

เป็นแอพพลิเคชั่นอย่างหนึ่งของ Adaptive noise control ทำงานโดยการกรองสัญญาณอ้างอิงด้วยฟังก์ชั่นระบบที่ประมาณค่าได้ของวิถีทุติยภูมิ $\hat{S}(z)$ ดังรูปที่ 2.4 ซึ่งจะทำให้ สัญญาณอ้างอิง x[n] ถูกหน่วงเวลาออกไปเท่าๆกับสัญญาณผิดพลาด e[n] การเพิ่มวิถีทุติยภูมิ S(z) ตามหลังวงจรกรอง W(z) ที่ปรับตัวได้โดยใช้ขั้นตอนวิธี LMS นั้นทำให้สัญญาณเศษเหลือมีค่าเท่ากับ

$$e[n] = d[n] + s[n] \otimes (w^T[n]x[n]) \qquad (2.10)$$

โดยที่ e[n] คือ สัญญาณผิดพลาด

d[n] คือ สัญญาณที่ต้องการ

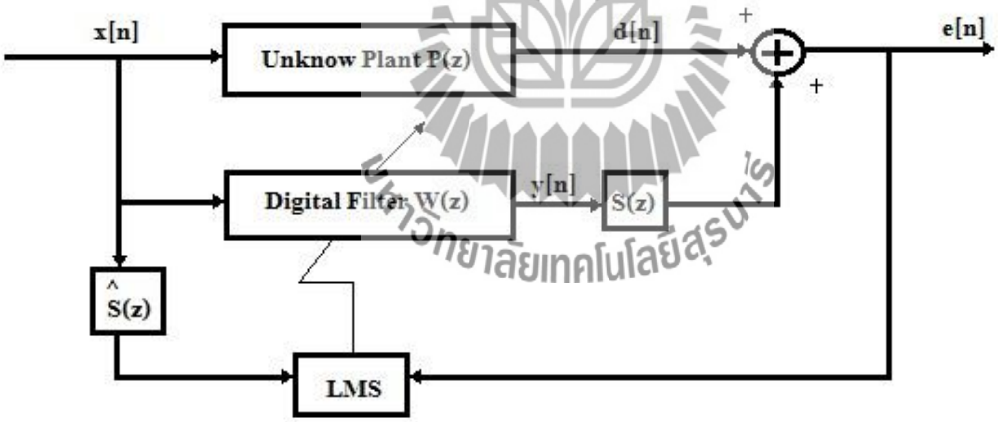
s[n] คือ ตัวกรองสัญญาณที่ได้จากการประมาณค่าวิถีทุติยภูมิ

w[n] คือ ตัวกรองสัญญาณที่ใช้ในการกลับเฟสเสียงรบกวน

x[n] คือ สัญญาณเสียงรบกวน

n คือ เป็นดัชนีเวลา (Time index)

$\otimes$ คือ การคอนโวลูชัน



รูปที่ 2.4 แผนภาพบล็อกของระบบ ANC ที่ใช้ขั้นตอนวิธี Filtered-X LMS

ดังนั้น $w[n+1] = w[n] - \mu e[n]\mathbf{x'}[n]$ (2.11)

โดยที่ e[n] คือ สัญญาณผิดพลาด

w[n] คือ สัมประสิทธิ์ตัวกรองที่ n ของวงจรกรอง FIR $\hat{H}(z)$

x[n] คือ สัญญาณเสียงรบกวน

μ คือ ขนาดของ Step size

n คือ เป็นดัชนีเวลา (Time index)

ค่า μเป็นตัวหลักในการขับเคลื่อนการปรับ coefficient ของ filter

$$0 < \mu < \frac{2}{\lambda max} \quad (2.12)$$

$$\mu = \frac{\lambda max}{10}$$

นอกจากนี้ได้มีการเสนอให้ค่า

โดยที่ λmaxคือ eigenvalue ของ autocorrelation matrix $E[u(n)^H u(n)]$

$$\lambda max < \sum_{i=1}^{M} \lambda i = \mathbf{E[u(n)^H u(n)]} \quad (2.13)$$

โดยที่ u(n) คือ ลำดับสัญญาณที่ป้อน

การใช้งานระบบ ANC ในทางปฏิบัติ เราจะไม่รู้ค่า S(z) จึงต้องประมาณค่าหา S(z) ดังนั้น สัญญาณอ้างอิงที่ใช้ในการคำนวณตามขั้นตอนวิธี LMS ในสมการที่ข้างบนจึงต้องนำ x[n] มาผ่านวิถีทุติยภูมิที่ประมาณค่าได้เสียก่อน

2.6 สรุป

จากทฤษฎีเครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMS สามารถสร้างเครื่องควบคุมเสียงรบกวนแบบไวงานโดยใช้พื้นฐานความรู้ทางด้าน digital signal processing โดยใช้ทฤษฎีหลักๆคือการคอนโวรูชันและการใช้ filterแบบLMSในการทำงานเป็นหลัก

บทที่ 3

การออกแบบและการจำลอง

3.1 กล่าวนำ

ในบทนี้จะกล่าวถึงขั้นตอนการออกแบบโปรแกรม ตามทฤษฎีเครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMSและโปรแกรมที่ใช้ในการสั่งงานควบคุมบอร์ด TMS320C5416 ผ่านคอมพิวเตอร์

3.2 Adaptive noise control using Filtered-X LMS

การทำงาน filtered-X LMS ชนิดป้อนกลับแบ่งการทำงานออกเป็น 2 เฟส คือ เฟสประมาณค่าหาแบบจำลองของวิถีทุติยภูมิแบบไม่เชื่อมตรง (off-line secondary path modeling) ดังแสดงในรูป 3.1 หลังจากการหาแบบจำลองของวิถีทุติยภูมิได้แล้วจะเข้าทำงานในเฟสกำจัดเสียงรบกวน (noise cancellation phase) ดังแสดงในรูปที่ 3.2 รายละเอียดการทำงานในแต่ละเฟสมีดังต่อไปนี้

3.2.1 เฟสประมาณค่าเพื่อ หาแบบจำลอง ของวิถีทุติยภูมิแบบไม่เชื่อมตรง (off-line secondary path modeling phase) มีขั้นตอนดังนี้

3.2.1.1 ใช้การสร้างค่าแบบ random เพื่อสร้างสัญญาณ white noise x[n] ไปส่งออกที่ลำโพง

3.2.1.2 ใช้ไมโครโฟนที่ติดตั้งวัดผลตอบสนองของวิถีทุติยภูมิ r[n]

3.2.1.3 คำนวณผลตอบสนองของวงจรกรอง FIR $\hat{H}$ (z)

$$\hat{r}[n] = \sum_{k=0}^{N-1} \hat{h}[k]x[n-k] \quad (3.1)$$

โดยที่ x[n-k] คือ ลำดับสัญญาณ white noise ที่มีการเลื่อนทางเวลาที่ k

$\hat{h}$ [k] คือ สัมประสิทธิ์ตัวกรองที่ k ของวงจรกรอง FIR$\hat{H}$ (z)

$\hat{r}$ [n] คือ ผลรวมแบบเชิงเส้นสัญญาณ

3.2.1.4 คำนวณสัญญาณผิดพลาด e[n]

$$e[n] = r[n] - \hat{r}[n] \quad (3.2)$$

โดยที่ e[n] คือ ค่าความผิดพลาดของสัญญาณ

r[k] คือ ค่า white noise ที่รับมาจากไมโครโฟน

$\hat{r}[n]$ คือ ผลรวมแบบเชิงเส้นสัญญาณ

3.2.1.5 ใช้ขั้นตอน LMS ปรับสัมประสิทธิของวงจรกรอง $\hat{h}_k[n]$

$$\hat{h}_k[n+1] = \hat{h}_k[n] + \mu e[n]x[n-k] \quad ;k = 0,1,\ldots, N-1 \quad (3.3)$$

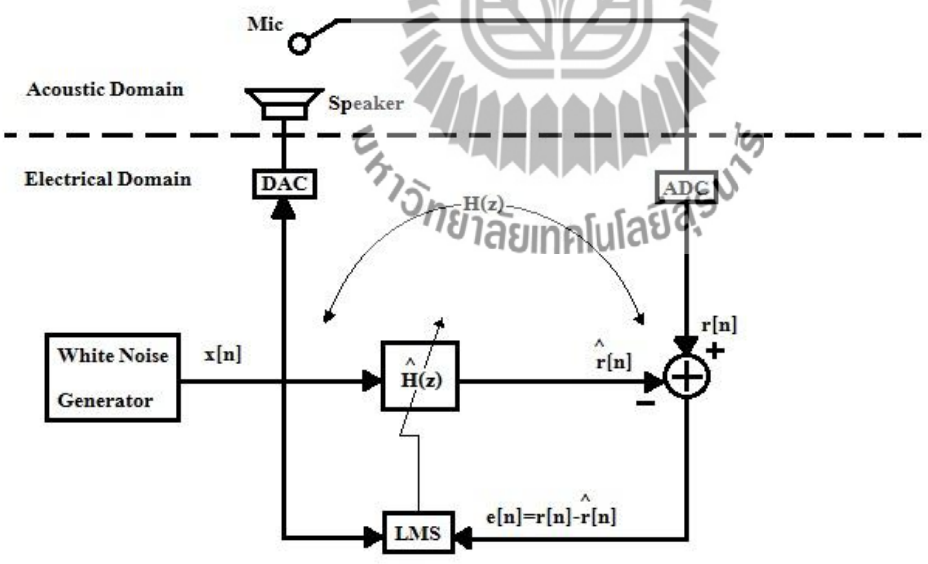
โดยที่ x[n-k] คือ ลำดับสัญญาณ white noise ที่มีการเลื่อนทางเวลาที่ k

$\hat{h}[k]$ คือ สัมประสิทธิ์ตัวที่ k ของวงจรกรอง FIR $\hat{H}(z)$

e[n] คือ ค่าความผิดพลาดของสัญญาณ

μ คือ ขนาดของ step size

3.2.1.6 ทำซ้ำขั้นตอนที่ 3.2.1.1 - 3.2.1.5 เป็นเวลา 15 วินาที แล้วจึงใช้สัมประสิทธิ์ $\hat{h}_k[n]$ เป็นค่าประมาณของผลตอบสนองอิมพัลส์ของวิถีทุติยภูมิ ในเฟสการกำจัดเสียงรบกวน



รูปที่ 3.1 เฟสประมาณค่าเพื่อ หาแบบจำลองของ วิถีทุติยภูมิแบบไม่เชื่อมตรง (off-line secondary path modeling phase)

3.2.2 เฟสกำจัดเสียงรบกวน (Noise cancellation phase) ของ Filtered X-LMS แบบป้อนกลับ มีขั้นตอนดังต่อไปนี้

3.2.2.1 ไมโครโฟนที่ติดตั้งในหูฟังวัดสัญญาณค่าผิดพลาด e[n]

3.2.2.2 ผลตอบสนองของวงจรกรองวิถีทุติยภูมิที่จำลองขึ้น

$$\hat{y}[n] = \sum_{k=0}^{N-1} \hat{h}[k] y^{-}[n-k] \quad (3.4)$$

โดยที่ $y^{-}[n-k]$ คือ ลำดับสัญญาณรบกวนที่มีการเลื่อนทางเวลาที่ k

$\hat{h}[k]$ คือ สัมประสิทธิ์ตัวกรองที่ k ของวงจรกรอง FIR $\hat{H}(z)$

$\hat{y}[n]$ คือ ผลรวมแบบเชิงเส้นสัญญาณ

3.2.2.3 คำนวณค่าประมาณของสัญญาณรบกวนที่เข้ามา

$$\hat{x}[n] = e[n] - \hat{y}[n] \quad (3.5)$$

โดยที่ $\hat{x}[n]$ คือ ค่าความผิดพลาดของสัญญาณ

e[n] คือ ค่าสัญญาณเสียงรบกวนที่รับมาจากไมโครโฟน

$\hat{y}[n]$ คือ ผลรวมแบบเชิงเส้นสัญญาณ

3.2.2.4 คำนวณสัญญาณด้านเสียงรบกวน

$$y^{-}[n] = \sum_{k=0}^{M-1} w[k] \hat{x}[n-k] \quad (3.6)$$

โดยที่ $\hat{x}[n-k]$ คือ ค่าความผิดพลาดของสัญญาณ

w[k] คือ สัมประสิทธิ์ตัวกรองที่ k ของวงจรกรอง FIR $\hat{H}(z)$

$y^{-}[n]$ คือ ผลรวมแบบเชิงเส้นสัญญาณ

3.2.2.5 คำนวณค่า filtered-x version ของ $\hat{x}[n]$

$$\bar{x}[n] = \sum_{k=0}^{N-1} \hat{h}[k]\hat{x}[n-k]$$

(3.7)

โดยที่ $\hat{x}[n-k]$ คือ ค่าความผิดพลาดของสัญญาณที่มีการเลื่อนทางเวลาที่ k

$\hat{h}[k]$ คือ สัมประสิทธิ์ตัวกรองที่ k ของวงจรกรอง FIR $\hat{H}(z)$

$\bar{x}[n]$ คือ ผลรวมแบบเชิงเส้นสัญญาณ

3.2.2.6 ใช้ขั้นตอนวิธี LMS ปรับค่าสัมประสิทธิ์ของวงจรกรอง W[n]

$$w[n+1]=w[n] - \mu e[n]\bar{x}[n-k] \quad ; k = 0,1,\ldots,M-1 \qquad (3.8)$$

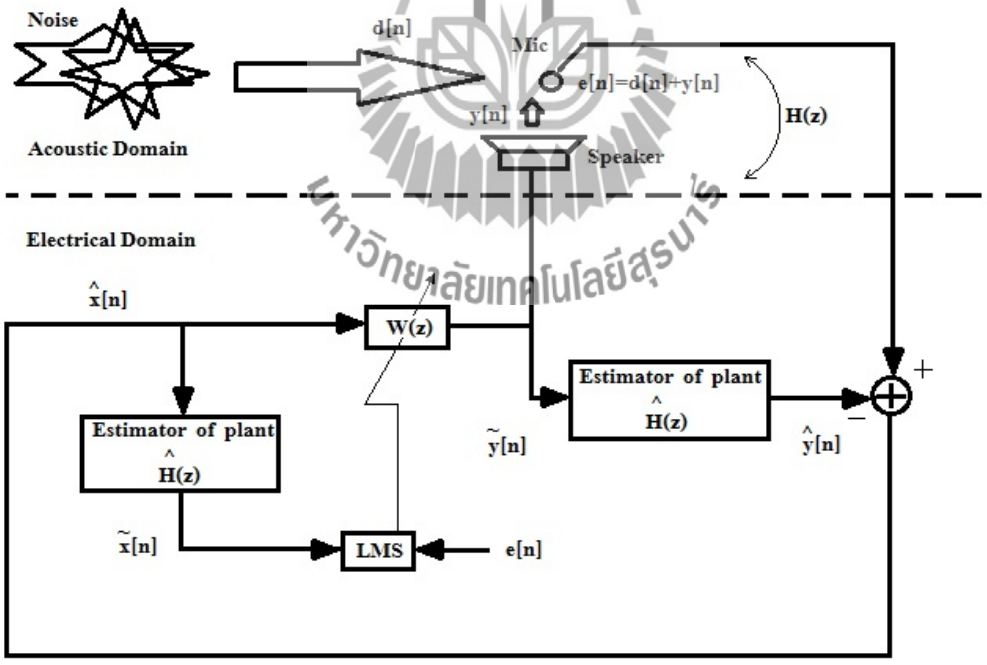
โดยที่ $\bar{x}[n-k]$ คือ ลำดับสัญญาณ white noise ที่มีการเลื่อนทางเวลาที่ k

$w[k]$ คือ สัมประสิทธิ์ตัวที่ k ของวงจรกรอง FIR $\hat{H}(z)$

$e[n]$ คือ ค่าความผิดพลาดของสัญญาณ

μ คือ ขนาดของ step size

3.2.2.7 ทำซ้ำขั้นตอนที่ 3.2.2.1 – 3.2.2.6

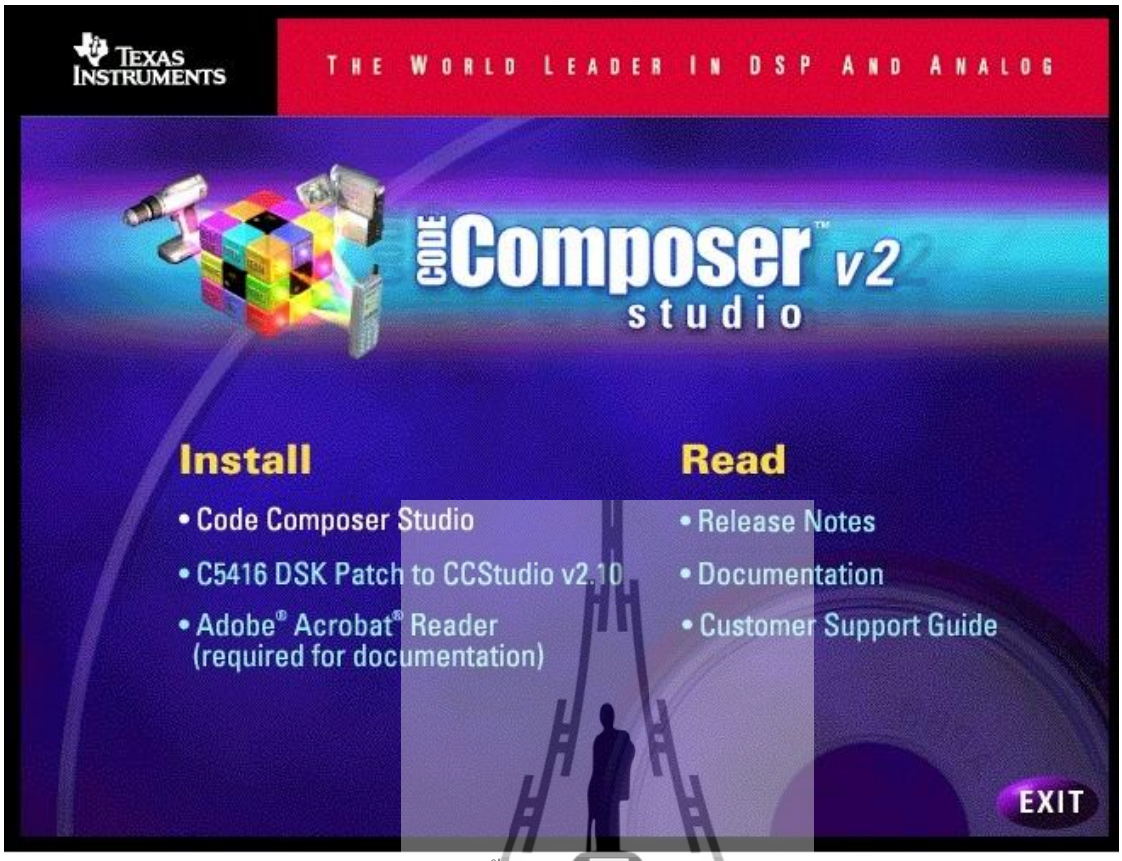


รูปที่ 3.2 เฟสกำจัดเสียงรบกวน (Noise cancellation phase)

3.3 วิธีการลงและการใช้โปรแกรม Code Composer Studio V2(CCS2)

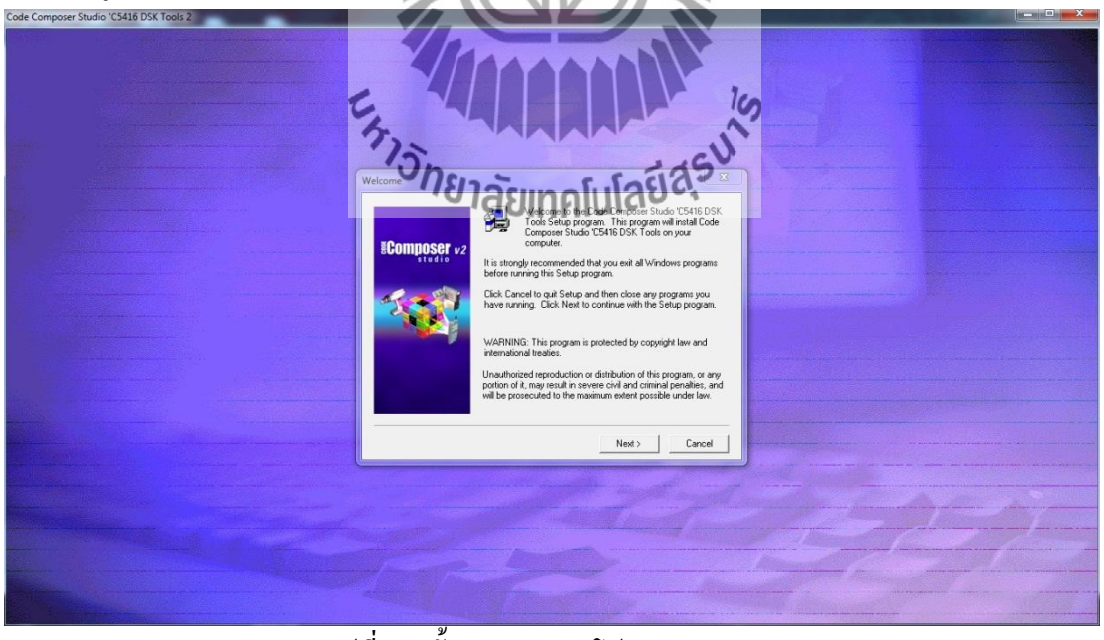
การติดตั้งโปรแกรม Code Composer Studio V2(CCS2)

3.3.1 ใส่แผ่น CD และเปิดขึ้นมาเลือก Install Code Composer Studio



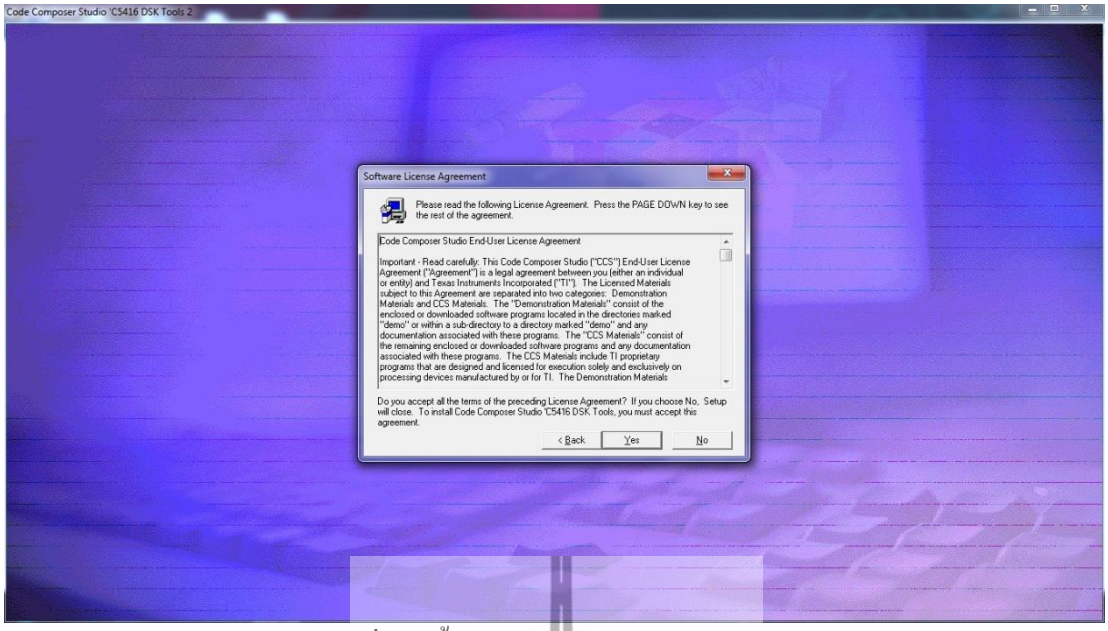
รูปที่ 3.3 ขั้นตอนการลงโปรแกรม CCS2

3.3.2 เมื่อเข้าสู่ขั้นตอนการลงกด Next



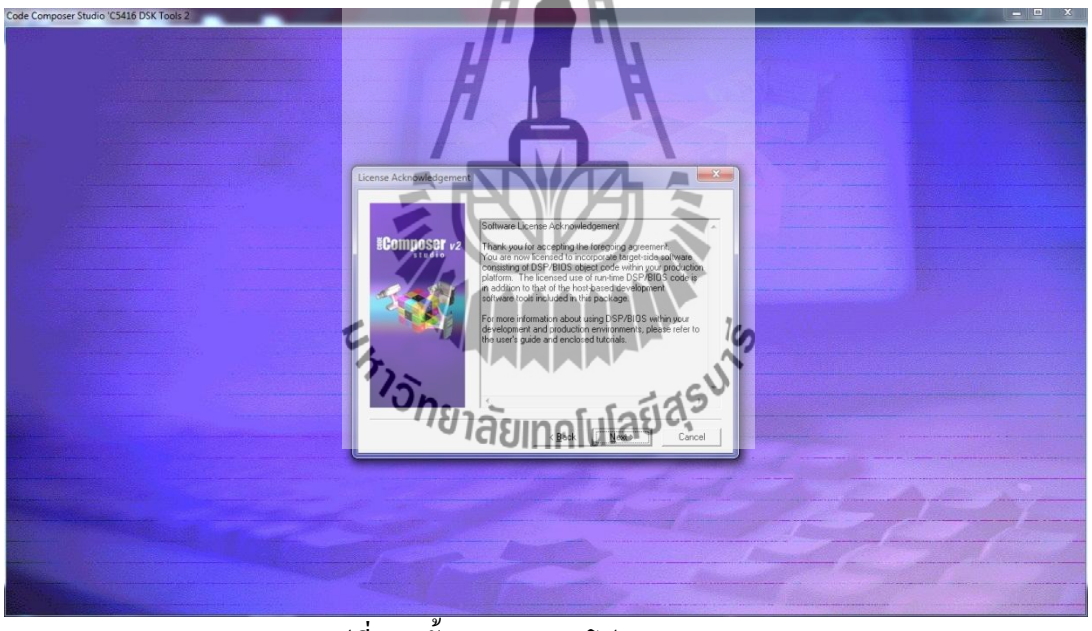
รูปที่ 3.4 ขั้นตอนการลงโปรแกรม CCS2

3.3.3 หน้า License Agreement จะบอกถึงข้อตกลงการใช้โปรแกรม กด Yes



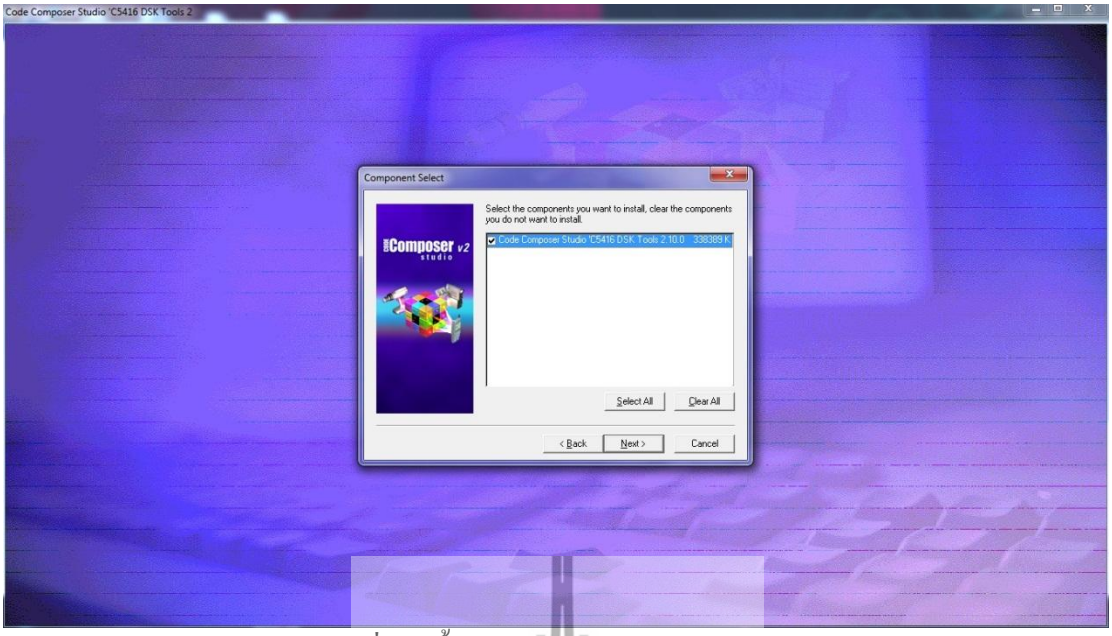
รูปที่ 3.5 ขั้นตอนการลงโปรแกรม CCS2

3.3.4 Solftware Lincense (ต่อ) กด Next



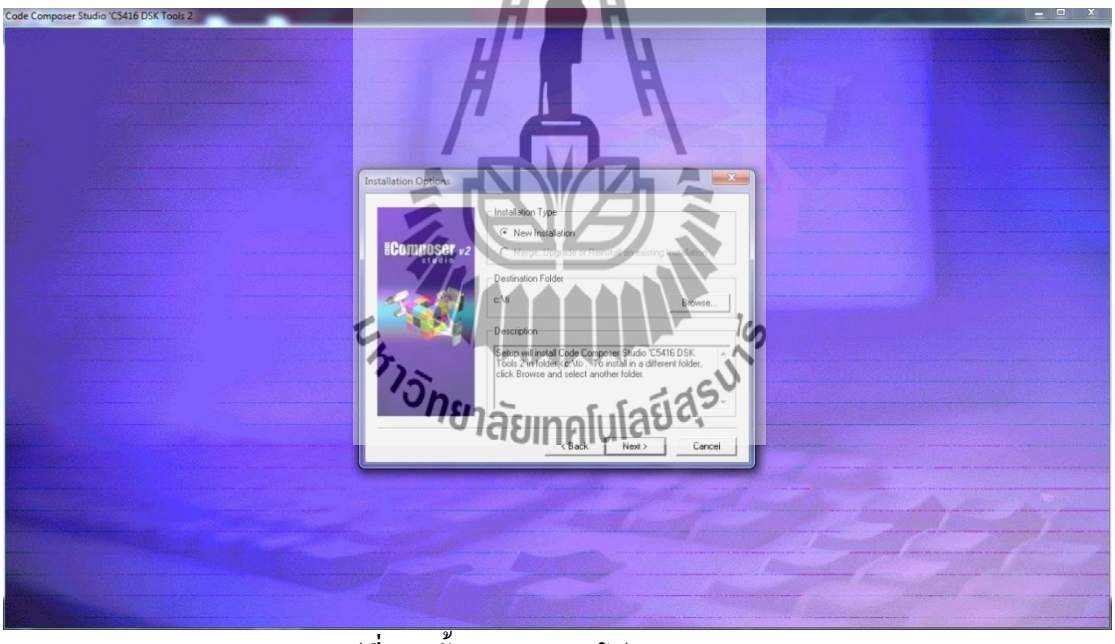
รูปที่ 3.6 ขั้นตอนการลงโปรแกรม CCS2

3.3.5 เลือก Code Composer Studio แล้วกด Next



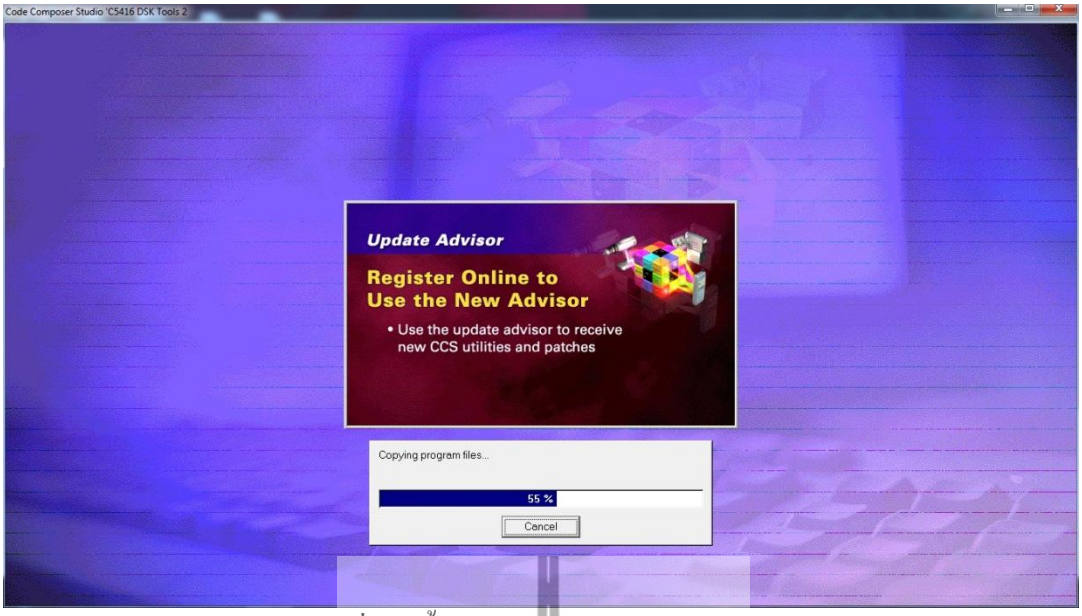
รูปที่ 3.7 ขั้นตอนการลงโปรแกรม CCS2

3.3.6 เลือกตำแหน่งที่จะลงโปรแกรมแล้วกด Next



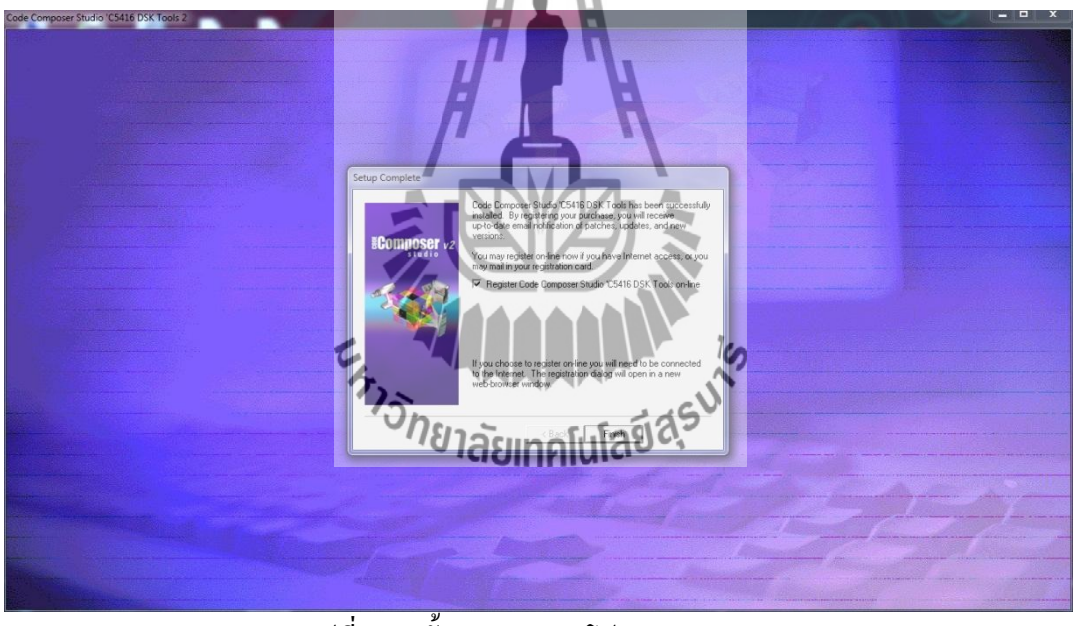
รูปที่ 3.8 ขั้นตอนการลงโปรแกรม CCS2

3.3.7 โปรแกรมจะทำการติดตั้งไปยังตำแหน่งที่ระบุ



รูปที่ 3.9 ขั้นตอนการลงโปรแกรม CCS2

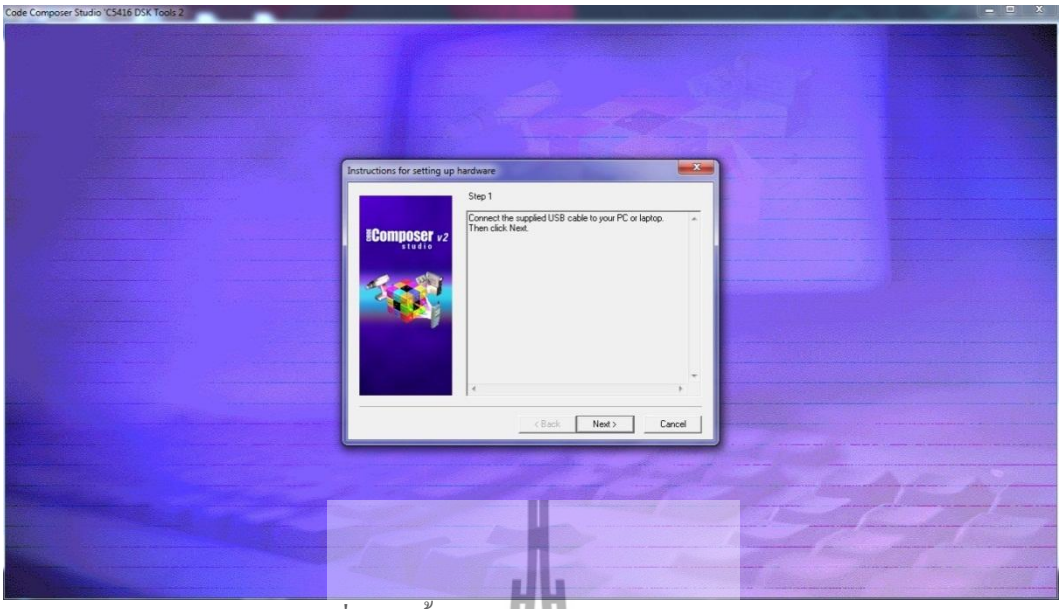
3.3.8 เมื่อเสร็จสิ้นให้กด Finish



รูปที่ 3.10 ขั้นตอนการลงโปรแกรม CCS2

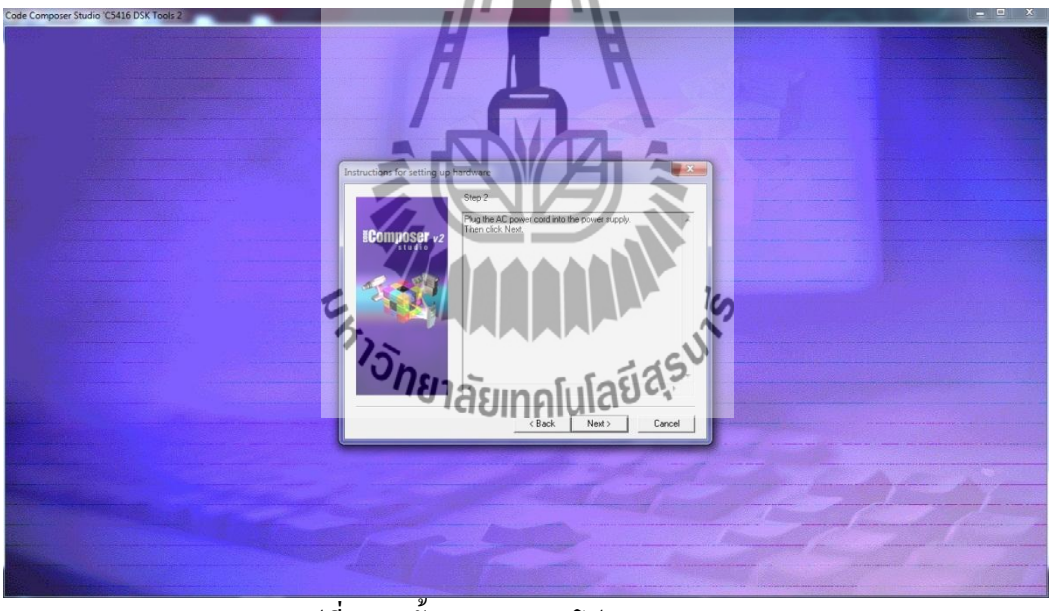
3.3.9 ทำตาม step ที่โปรแกรมสั่ง

3.3.9.1 ต่อสายUSBเข้าที่บอร์ด TMS320VC5416และคอมพิวเตอร์



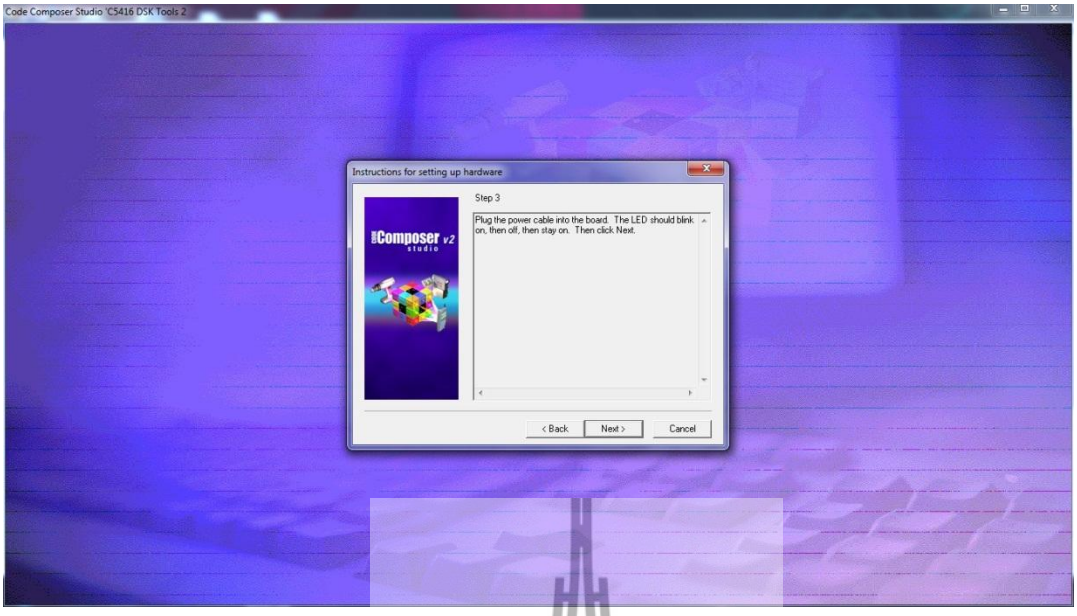
รูปที่ 3.11 ขั้นตอนการลงโปรแกรม CCS2

3.3.9.2 ต่อสายไฟACเข้าที่บอร์ดTMS320VC5416



รูปที่ 3.12 ขั้นตอนการลงโปรแกรม CCS2

3.3.9.3 เมื่อต่อสายไฟแล้วให้สังเกตไฟLEDถ้าติดก็สามารถใช้งานได้



รูปที่ 3.13 ขั้นตอนการลงโปรแกรม CCS2

3.3.10 กดปุ่มFinish เพื่อเสร็จสิ้นการติดตั้ง

รูปที่ 3.14 ขั้นตอนการลงโปรแกรม CCS2

3.3.11 วิธีการใช้โปรแกรม Code Composer Studio V2(CCS2)

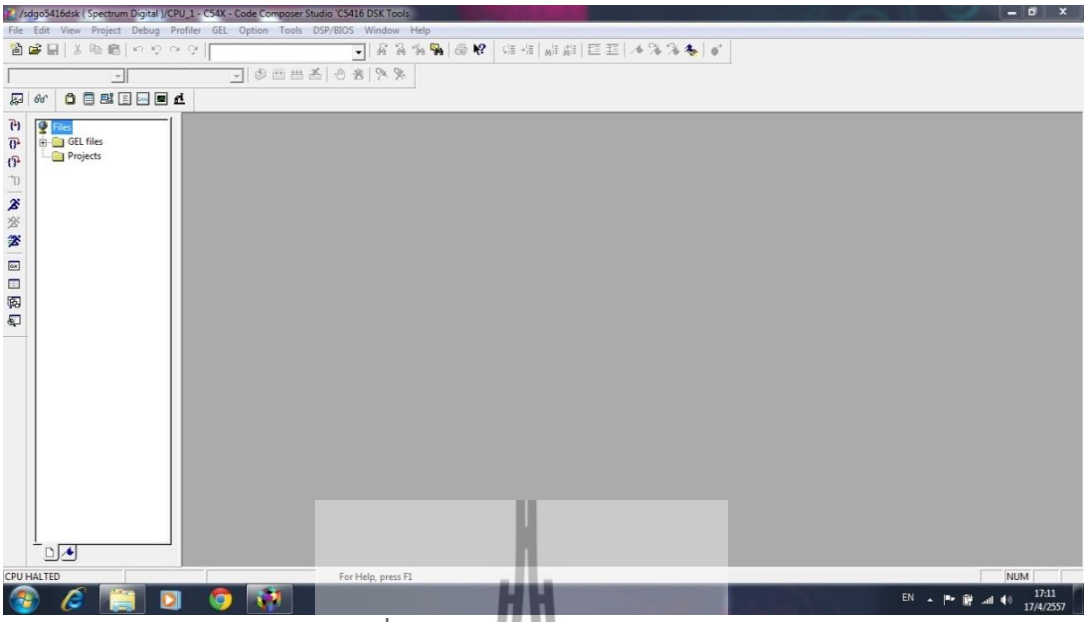
3.3.11.1เปิดโปรแกรมที่Iconดังรูป

รูปที่ 3.15 วิธีการใช้โปรแกรม CCS2

3.3.11.2 โปรแกรมจะทำการเช็คสถานการณ์เชื่อมต่อ USB

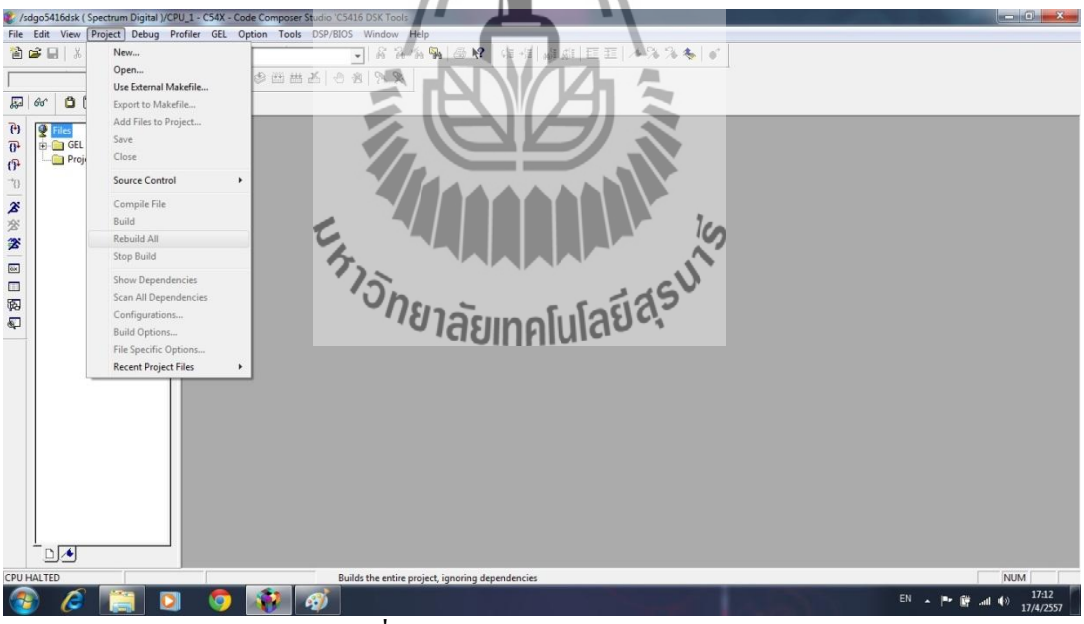
รูปที่ 3.16 วิธีการใช้โปรแกรม CCS2

3.3.11.3 หน้าตาโปรแกรม Code Composer Studio V2(CCS2)



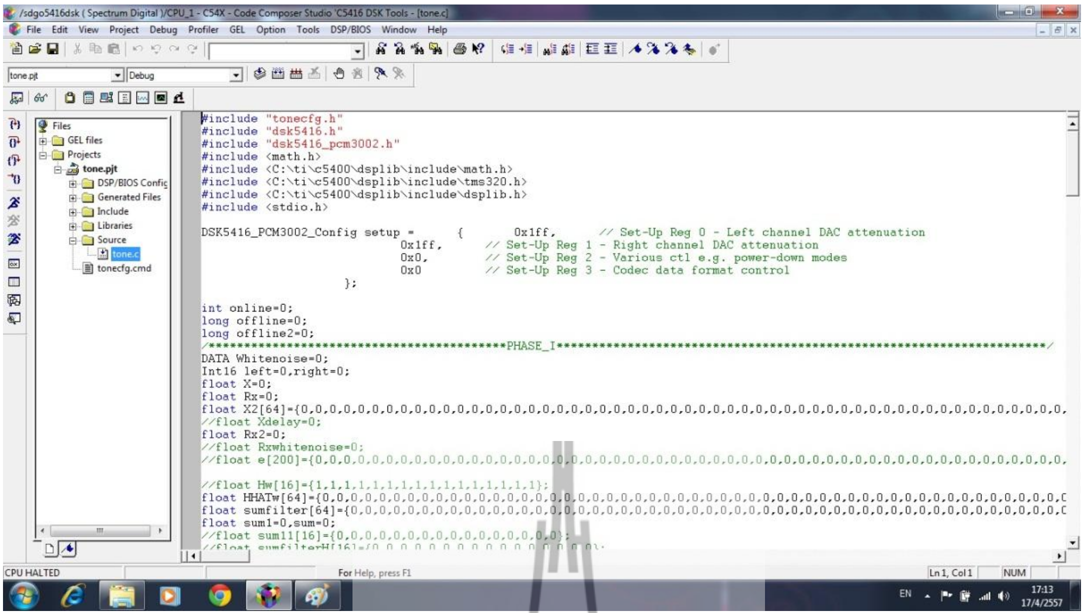
รูปที่ 3.17 วิธีการใช้โปรแกรม CCS2

3.3.11.4 ให้เลือกที่คำสั่ง Project หากต้องการสร้างProjectใหม่เลือก New แต่ถ้ามีอยู่แล้วก็สามารถกด Open ได้เลย



รูปที่ 3.18 วิธีการใช้โปรแกรม CCS2

3.3.11.5 เมื่อได้แล้วProjectจะมาอยู่ในช่องด้านซ้ายโดยจะแบ่งเป็นหลายๆส่วน และในส่วนของSource จะเป็นส่วนที่ใช้เขียนโปรแกรมต่างๆ

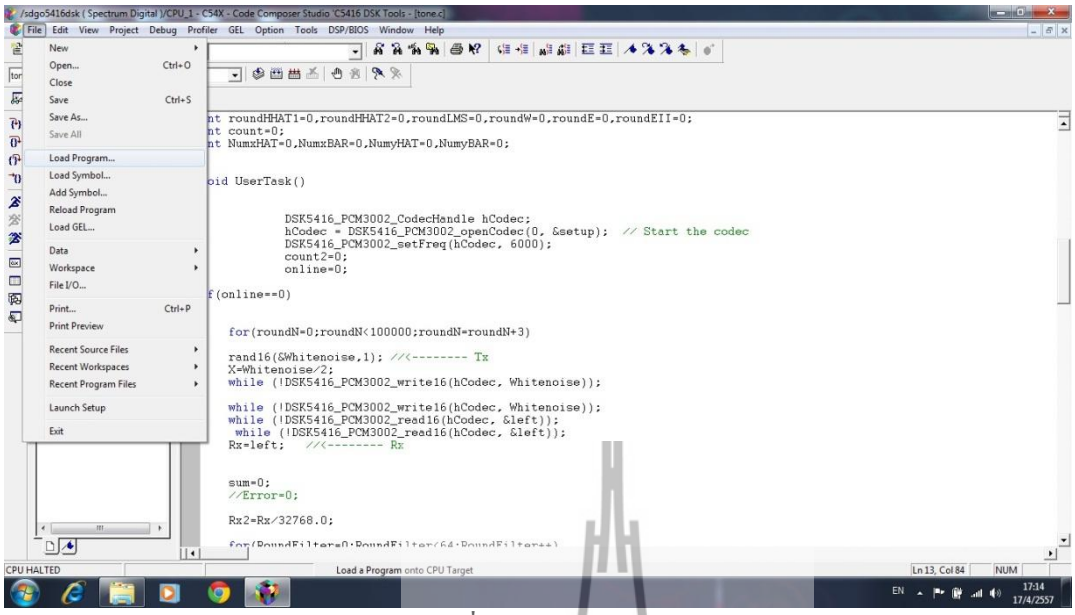


รูปที่ 3.19 วิธีการใช้โปรแกรม CCS2

3.3.11.6 เมื่อทำการเขียนโปรแกรมเสร็จให้กดปุ่มดังรูปเพื่อ complier ในขั้นตอนนี้โปรแกรมจะแสดง errorต่างๆให้เห็น(ถ้ามี)

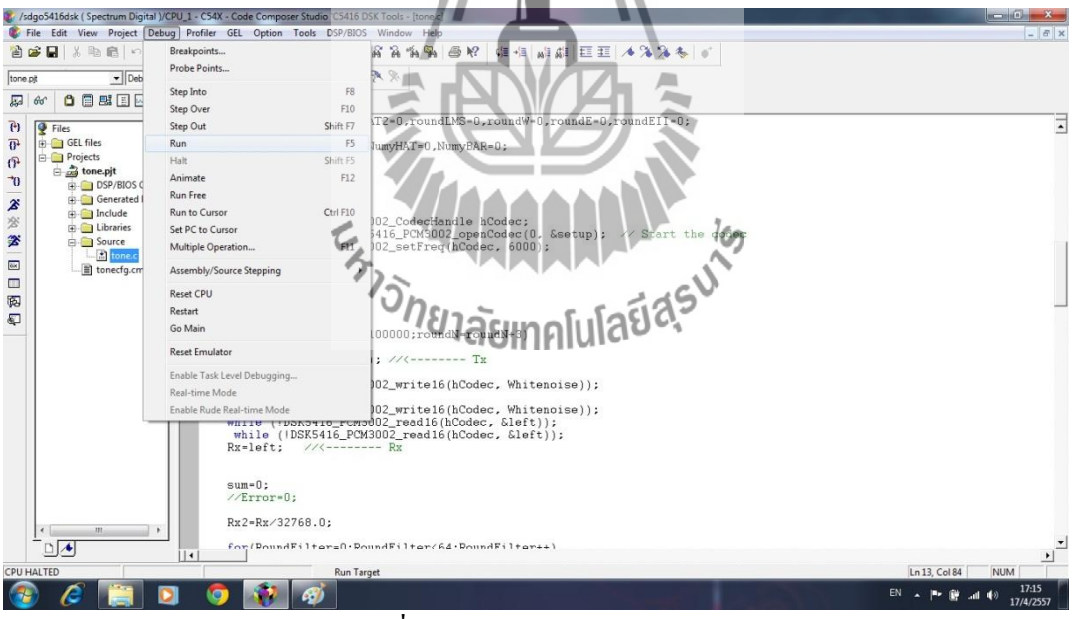
รูปที่ 3.20 วิธีการใช้โปรแกรม CCS2

3.3.11.7 เมื่อcomplierผ่านแล้วให้ทำการโหลดโปรแกรมลงบอร์ด TMS320VC5416 โดยเลือกคำสั่ง Load Program ที่เมนูFile



รูปที่ 3.21 วิธีการใช้โปรแกรม CCS2

3.3.11.8 เมื่อLoad Program เสร็จให้เลือกที่เมนู Debug เลือก Run เพื่อทำการ run โปรแกรม



รูปที่ 3.22 วิธีการใช้โปรแกรม CCS2

3.4 code ภาษาซี เครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMS สำหรับบอร์ด TMS320C5416

```
#include "tonecfg.h"
#include "dsk5416.h"
#include "dsk5416_pcm3002.h"
#include <math.h>
#include <C:\ti\c5400\dsplib\include\math.h>
#include <C:\ti\c5400\dsplib\include\tms320.h>
#include <C:\ti\c5400\dsplib\include\dsplib.h>
#include <stdio.h>
DSK5416_PCM3002_Config setup =  {       0x1ff,     // Set-Up Reg 0 - Left
channel DAC attenuation
                                0x1ff,     // Set-Up Reg 1 - Right channel
DAC attenuation
                                0x0,       // Set-Up Reg 2 - Various ctl e.g.
power-down modes
                                0x0        // Set-Up Reg 3 - Codec data format
control
                                };
int online=0;
long offline=0;
long offline2=0;
/*************************************PHASE_I*************************
************************************/
DATA Whitenoise=0;
Int16 left=0,right=0;
float X=0;
float Rx=0;
float
X2[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float Rx2=0;
```

```
float
HHATw[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float
sumfilter[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
float sum1=0,sum=0;
float Error=0;
int RoundShift=0,RoundShift2=0,RoundFilter=0,RoundLMS=0,RoundFilterH=0;
longroundN=0;
int i=0,j=0,k=0;
int count2=0;
/****************************************PHASE_II***************************
*****************************************/
Int16 leftII=0,rightII=0;
double
Ww[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double
xHATx[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double
xBARy[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double
yHATy[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
double
yBARx[64]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
doubleeII=0;
Int16 yout=0;
```

```
double sumHHAT1=0;
double sumHHAT2=0;
doublesumHHATW=0;
doublesumout=0;
int roundHHAT1=0,roundHHAT2=0,roundLMS=0,roundW=0,roundE=0,roundEII=0;
int count=0;
intNumxHAT=0,NumxBAR=0,NumyHAT=0,NumyBAR=0;
voidUserTask()
{                       DSK5416_PCM3002_CodecHandle hCodec;
                hCodec = DSK5416_PCM3002_openCodec(0, &setup);  // Start the codec
                DSK5416_PCM3002_setFreq(hCodec, 6000);
                        count2=0;
                        online=0;
if(online==0)
{
        for(roundN=0;roundN<100000;roundN=roundN+3)
{
        rand16(&Whitenoise,1); //<-------- Tx
        X=Whitenoise/2;
        while (!DSK5416_PCM3002_write16(hCodec, Whitenoise));
        while (!DSK5416_PCM3002_write16(hCodec, Whitenoise));
        while (!DSK5416_PCM3002_read16(hCodec, &left));
while (!DSK5416_PCM3002_read16(hCodec, &left));
        Rx=left;   //<-------- Rx
        sum=0;
        Rx2=Rx/32768.0;
        for(RoundFilter=0;RoundFilter<64;RoundFilter++)
        {
                sumfilter[RoundFilter]=(X2[RoundFilter])*(HHATw[RoundFilter]);
                sum=sum+(sumfilter[RoundFilter]);
        }
        Error=Rx2-sum;
```

```
		for(RoundLMS=0;RoundLMS<64;RoundLMS++)
		{

		HHATw[RoundLMS]=(HHATw[RoundLMS])+((1.0/32768.0)*Error*X2[RoundLMS]);
		}
		for(RoundShift=63;RoundShift>0;RoundShift--)
		{
			X2[RoundShift] = X2[RoundShift-1];
		}
		X2[0]=X/32768.0;
}
online=1;
}
if(online==1)
{
for(roundE=0;roundE<10;roundE=1)
{
		while (!DSK5416_PCM3002_read16(hCodec, &leftII));
		while (!DSK5416_PCM3002_read16(hCodec, &leftII));
				eII=leftII/32768.0;
			sumHHAT1=0;
			sumHHAT2=0;
			sumHHATW=0;
			sumout=0;
			for(roundHHAT1=0;roundHHAT1<64;roundHHAT1++)
			{

		sumHHAT1=sumHHAT1+(HHATw[roundHHAT1]*yBARx[roundHHAT1]);
			}
			for(NumxHAT=63;NumxHAT>0;NumxHAT--)
			{
				xHATx[NumxHAT] = xHATx[NumxHAT-1];
```

```
                }
                xHATx[0]=eII-sumHHAT1;
                for(roundW=0;roundW<64;roundW++)
                {
                        sumHHATW=sumHHATW+(Ww[roundW]*xHATx[roundW]);
                }
                yout=sumHHATW*(32768.0);
                while (!DSK5416_PCM3002_write16(hCodec, yout));
                while (!DSK5416_PCM3002_write16(hCodec, yout));
                for(NumyBAR=63;NumyBAR>0;NumyBAR--)
                {
                        yBARx[NumyBAR] = yBARx[NumyBAR-1];
                }
                yBARx[0]=sumHHATW;
                for(roundHHAT2=0;roundHHAT2<64;roundHHAT2++)
                {

        sumHHAT2=sumHHAT2+(HHATw[roundHHAT2]*xHATx[roundHHAT2]);
                }
                        for(NumxBAR=63;NumxBAR>0;NumxBAR--)
                {
                        xBARy[NumxBAR] = xBARy[NumxBAR-1];
                }
                xBARy[0]=sumHHAT2;
                for(roundLMS=0;roundLMS<64;roundLMS++)
                {
        Ww[roundLMS]=Ww[roundLMS]-((1.0/32768.0)*eII*xBARy[roundLMS]);
                }
}
}
        DSK5416_PCM3002_closeCodec(hCodec);
}
```

```
void main()
{
 //         DSK5416_int();
}
```

3.5 สรุป

ในบทนี้ได้ศึกษาเกี่ยวกับการออกแบบเครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMS โดยใช้บอร์ด TMS320C5416 ผ่านโปรแกรม CCS ในการเขียน software ในการสั่งงานให้บอร์ดควบคุมเสียงรบกวน โดยใช้ภาษาซีเป็นภาษาหลักในการเขียนโปรแกรม และไม่ใช้libraryหลักของภาษาซีที่มีให้สำเร็จแต่ทำการเขียนการทำงานทีละขั้นตอนด้วยbasicของภาษาC ทั้งหมดเนื่องจากibraryของภาษาcที่มีให้นั้นมีการทำงานที่ไม่คงที่และไม่สมบูรณ์ทางผู้จัดทำจึงทำการเขียนขึ้นมาเอง ทั้งการคอนโวรูชันหรือการจำลองการทำงานของLMSทั้งหมด

บทที่ 4

ผลการทดลอง

เครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMS โดยใช้บอร์ด TMS320C5416

4.1 กล่าวนำ

ในบท นี้เป็นการนำผล ที่ได้จากการการวัดสัญญาณเสียงรบกวนผ่านไมโครโฟน เข้าสู่บอร์ด TMS320C5416 และอ่านค่า Amplitude ของสัญญาณเสียงผ่านโปรแกรม code compressor studio ทำให้ค่าที่อ่านได้ผลดังตารางที่จะกล่าวถึงต่อไปโดยมีการแบ่งการทำงานแบบปกติและการปิดการทำงานของ LMSในเฟส2เพื่อเทียบผลการทำงานของโปรแกรม ซึ่งค่าที่วัดได้อาจมีความคาดเคลื่อนเนื่องจากใช้เครื่องมือวัดที่ไม่ได้มาตรฐาน และเสียงรบกวนจากสิ่งแวดล้อมต่างๆ

ตารางที่4.1ตารางแสดงผลขนาดของสัญญาณที่รับเข้าในกรณีไม่ผ่านการควบคุมสัญญาณเสียงและผ่านการควบคุมสัญญาณเสียง ที่ความถี่ต่างๆ F=100,150,200,250,300

	F=100		F=150		F=200	
n	Before(v)	after(v)	before(v)	after(v)	before(v)	after(v)
1	-0.016205	-0.14825	0.10025	-0.119884	0.15063	-0.0525
2	0.3472	-0.12489	0.124298	0.09504	0.11276	-0.1155
3	0.651794	-0.0561	0.338898	0.117802	0.02411	-0.0916
4	0.630829	-0.07736	0.490997	0.153402	-0.05954	-0.0589
5	0.618042	0.14773	0.53479	0.110172	-0.08115	0.02114
6	0.439484	0.14519	0.358704	-0.045883	-0.04675	0.00879
7	0.0593262	-0.00553	0.110779	-0.160609	-0.04678	0.0382
8	-0.340942	-0.03653	-0.107343	-0.218779	0.02884	-0.0485
9	-0.452551	-0.11149	-0.361365	-0.143951	0.10846	0.04674
10	-0.50307	-0.07348	-0.48407	-0.100197	0.1453	0.04028
11	-0.411353	-0.02707	-0.347754	0.1592646	0.11578	0.02733
12	-0.352873	0.013278	-0.146802	0.1858959	-0.10723	0.06761
13	0.101705	0.11806	0.1852417	0.1550152	-0.18704	-0.001225
14	0.156433	0.19076	0.0587463	0.0360886	-0.23721	0.05531
15	0.216675	0.13327	0.3050232	-0.108994	-0.23578	0.04557

	F=250		F=300	
	before(v)	after(v)	before(v)	after(v)
1	0.1232666	-0.03868	0.4551392	0.0371
2	0.36947	-0.09678	0.641052	0.1402
3	0.44085	-0.0654	0.599731	0.0624
4	0.308038	-0.05264	0.356903	-0.046
5	0.071167	0.06098	-0.066132	-0.0644
6	-0.095306	0.03451	-0.424408	-0.0521
7	-0.28125	-0.02865	-0.561823	0.0512
8	-0.38107	0.013668	-0.49518	0.1257
9	-0.35701	0.111866	-0.24316	0.0501
10	-0.19745	0.17305	0.087189	-0.0896
11	0.07058	0.114434	0.388367	-0.083
12	0.38587	-0.142294	0.568573	-0.0862
13	0.560578	-0.028776	0.529388	0.04306
14	0.42248	-0.06043	0.282043	0.08663
15	0.146332	0.153228	-0.128235	-0.0389

***หมายเหตุ** before = สัญญาณเสียงรบกวนที่ไม่ผ่านการควบคุมสัญญาณเสียงด้วยบอร์ด TMS320C5416 หน่วยเป็น v

after = สัญญาณเสียงรบกวนที่ผ่านการควบคุมสัญญาณเสียงด้วยบอร์ดTMS320C5416 หน่วยเป็น v

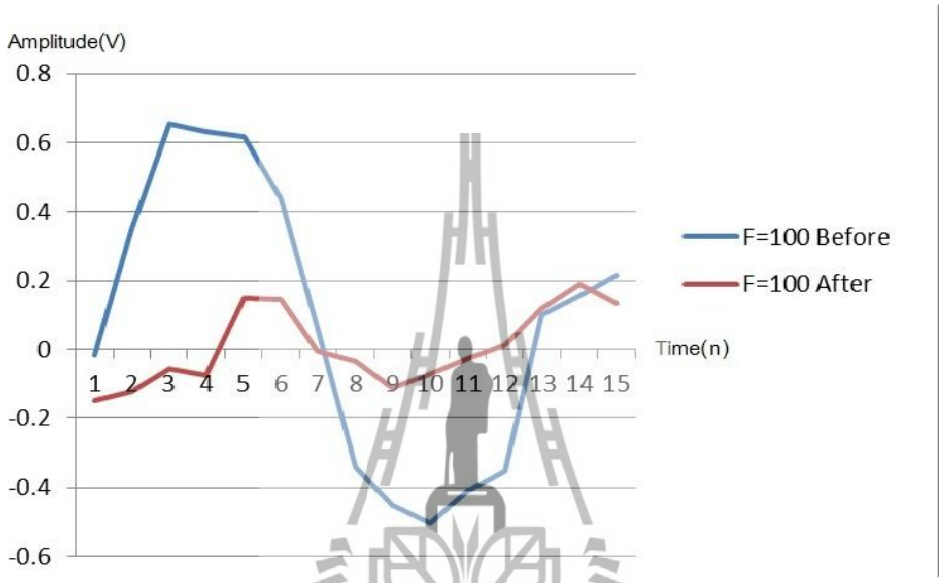
n= จำนวนช่องเก็บข้อมูลตัวอย่าง

ค่าที่ได้เป็นขนาด Amplitudeของสัญญาณเสียง

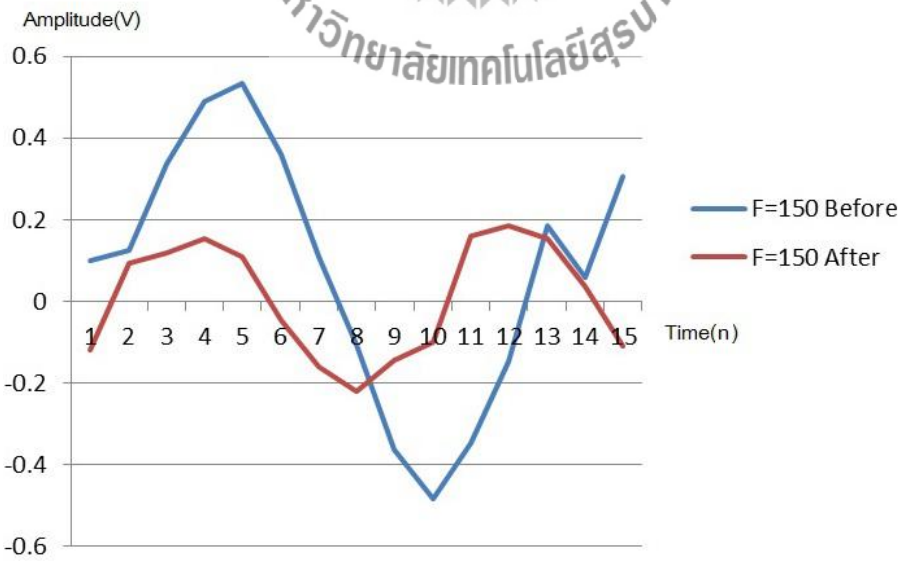
4.2กราฟแสดงผลขนาดของสัญญาณที่รับเข้าในกรณีไม่ผ่านการควบคุมสัญญาณเสียงและผ่านการควบคุมสัญญาณเสียง

***หมายเหตุ** เส้นสีฟ้า = สัญญาณเสียงรบกวนที่ไม่ผ่านการควบคุมสัญญาณเสียงด้วยบอร์ด TMS320C5416

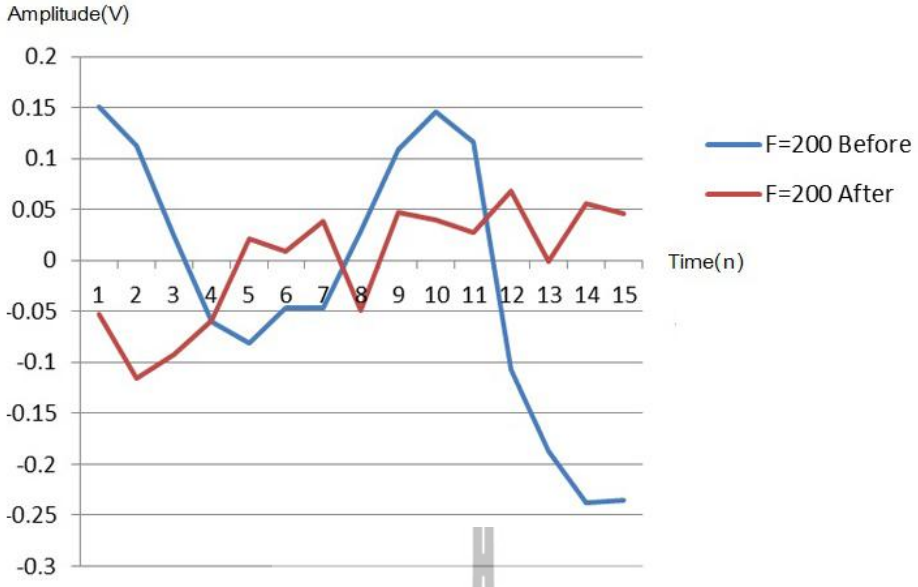
เส้นสีแดง = สัญญาณเสียงรบกวนที่ผ่านการควบคุมสัญญาณเสียง ด้วยบอร์ด TMS320C5416



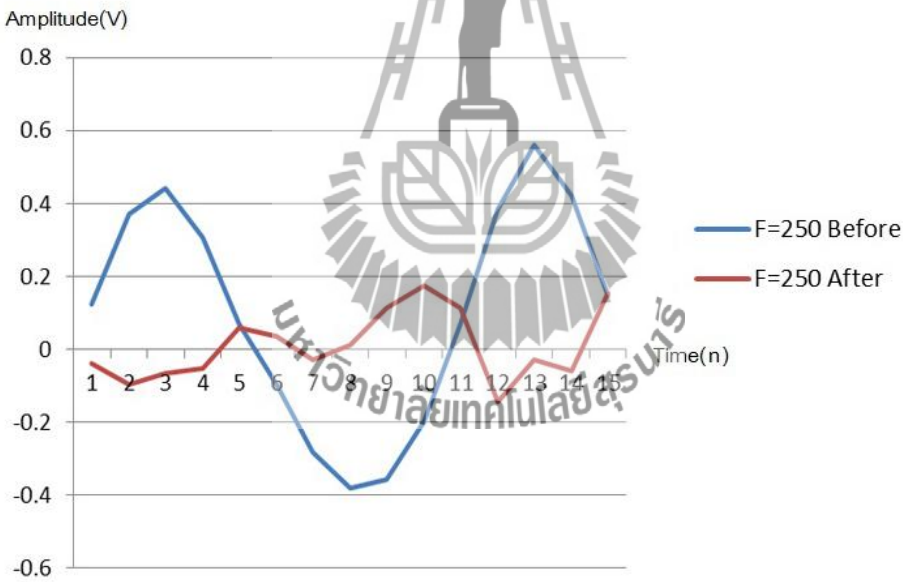
รูปที่ 4.1 กราฟแสดงผลการทำงานของโปรแกรมที่ความถี่ 100 HZ



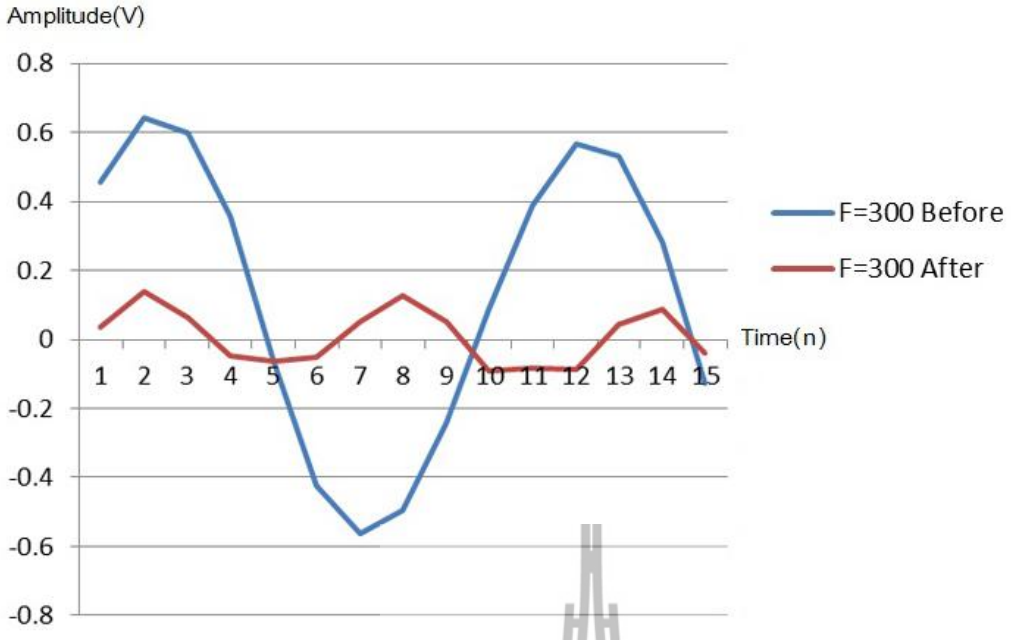
รูปที่ 4.2 กราฟแสดงผลการทำงานของโปรแกรมที่ความถี่ 150 HZ



รูปที่ 4.3 กราฟแสดงผลการทำงานของโปรแกรมที่ความถี่ 200 HZ



รูปที่ 4.4 กราฟแสดงผลการทำงานของโปรแกรมที่ความถี่ 250 HZ



รูปที่ 4.5 กราฟแสดงผลการทำงานของโปรแกรมที่ความถี่ 300 HZ

4.4 ตารางค่าการลดทอนที่ความถี่ต่างๆ

จากสูตร

ค่าการลดทอน = 20Log(Vหลังการลดทอน/Vก่อนการลดทอน)

ความถี่ (Hz)	ก่อนการลดทอน (dBW)	หลังการลดทอน (dBW)	ค่าการทดทอน(dBW)
100	1.17	-21.15	-22.32
150	1.29	-18.832	-20.122
200	-10.01	-35.26	-25.25
250	4	-16.59	-20.59
300	5.97	-17.31	-23.28

ตารางที่ 4.2 ตารางแสดงการลดทอนตามความถี่

4.5 สรุป

ในบทที่ 4 เป็นการทดสอบเครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMS โดยใช้บอร์ด TMS320C5416 จากราฟที่แสดงการทำงานหรือการที่บอร์ดTMS320C5416 ทำการลดทอนเสียงรบกวน นั้นจะเห็นได้ว่าเสียงnoiseมีขนาดลดลงจากnoiseปกติเป็นอย่างมากซึ่งคิดเป็นค่าการลดทอนจากสูตร

ค่าการลดทอน = 20Log(Vหลังการลดทอน/Vก่อนการลดทอน)

จะสามารถลดทอนเสียงรบกวนได้ประมาณ 20 dB ที่ความถี่ 100-300Hz

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 สรุป

โครงงานฉบับนี้ได้ดำเนินการศึกษาวิเคราะห์ออกแบบ เครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMSโดยใช้บอร์ด TMS320C5416 สั่งงานผ่านโปรแกรมCode Composer Studio ที่ใช้ภาษาซีเป็นหลักในการเขียนโดยตัวโปรแกรมจะเขียนด้วยBasicของภาษาCที่เข้าใจได้ง่ายมีการแบ่งการทำงานเป็นส่วนอย่างชัดเจนและแยกการทำงานเพื่อให้ดูง่าย ในส่วนของชุดหูฟังและไมโครโฟนมีขายตามท้องตลาดที่หาซื้อได้ง่าน เมื่อนำมาจัดทำเครื่องควบคุมเสียงรบกวนแบบไวงาน Filtered-X LMSสามารถตัดเสียงรบกวนได้ประมาณ 20 dBที่ความถี่ 100-300 Hz แต่เนื่องจากบอร์ดTMS320C5416 มีความเร็วในการประมวลผลในระดับหนึ่งที่ไม่สามารถรองรับการทำงานของภาษาซีที่นิยมกันในปัจจุบันทำให้การทำงานเป็นไปอย่างไม่สมบูรณ์ หรือบอร์ดไม่สามารถทำงานแบบreal time บนโครงสร้างของโค้ดภาษาซีได้จึงทำให้ผลที่ได้มีการผิดพลาดไปบ้างไม่มากก็น้อย และผู้จัดทำไม่สามารถบอกได้ว่าผิดพลาดประมาณใดเนื่องจากไม่มีเครื่องมือที่สามารถวัดค่าที่แน่นอนได้

ตารางสรุปผลการทดลอง

ความถี่ (Hz)	ก่อนการลดทอน (dBW)	หลังการลดทอน (dBW)	ค่าการทดทอน(dBW)
100	1.17	-21.15	-22.32
150	1.29	-18.832	-20.122
200	-10.01	-35.26	-25.25
250	4	-16.59	-20.59
300	5.97	-17.31	-23.28

ตารางที่ 5.1 ตารางสรุปผลการลดทอน

ค่าการลดทอน = 20Log(Vหลังการลดทอน/Vก่อนการลดทอน)

5.2 ข้อเสนอแนะ

เนื่องจากการทำงานของภาษาซีต้องการความเร็วในการทำงานค่อนข้างมากแต่ในบอร์ด TMS320C5416 มีความเร็วในการประมวลผลในระดับหนึ่งซึ่งยังไม่พอสำหรับภาษาcสมัยใหม่ทำให้บอร์ดมีการทำงานที่ไม่ทันคือรับค่าเข้ามาในโปรแกรมช้าหรือกระโดดข้ามค่าทำให้ค่าที่รับเข้ามาขาดหายหรือไม่ถูกต้องตามความจริง การประมวลผลที่ช้าทำให้ใช้เวลามากในการคำนวณค่าต่างๆเกิดการล่าช้าในการส่งข้อมูลออกไป ทำให้การทำงานนี้ไม่เป็นไปตามที่คาดหวังไว้และเกิดข้อผิดพลาดต่างๆมากมาย ดังนั้นถ้ามีการพัฒนาต่อไปควรใช้ภาษา assembly ที่มีการทำงานที่ไวกว่าภาษาCอาจจะทำงานได้ดีกว่าการที่ใช้ภาษาCทั้งหมด

5.2.1ปัญหาที่พบและวิธีแก้ไข

1. คำสั่ง Library ของภาษา C ไม่สามารถใช้งานได้โดยตรงโดยค่าที่ได้จากการใช้คำสั่ง Library เกิดการคลาดเคลื่อนทั้งหมด

วิธีการแก้ปัญหา ทำการสร้างฟังก์ชันโดยใช้คำสั่งพื้นฐานในการทำงานแทน

2. ชุดหูฟังที่ใช้ ตามท้องตลาดจะมีไมโครโฟนติดมาอยู่แล้วแต่ประสิทธิภาพไม่พอสำหรับการทำงานคือรับสัญญาณเสียงได้ไม่ชัด

วิธีการแก้ปัญหา ทำการต่อไมโครโฟนชนิดโมโนเพิ่มทั้งสองข้าง

3. ความเร็วในการประมวลผลของบอร์ดทำได้ค่อนข้างช้าไม่สามารถรองรับการทำงานของตัวแปรที่มีจำนวนมากได้

วิธีการแก้ปัญหา ลดขนาดตัวแปรลงให้มีขนาดพอเหมาะที่บอร์ดจะสามารถทำงานได้

4.ไม่สามารถตรวจวัดค่าสัญญาณเสียงในอากาศที่ถูกต้องตามจริงได้

วิธีการแก้ปัญหา ใช้การรับสัญญาณเข้ามาในบอร์ดแล้วพร็อตกราฟเพื่อดูค่าต่างๆ

บรรณานุกรม

[1] ผู้ช่วยศาสตราจารย์ ดร. สมศักดิ์ วาณิชอนันต์ชัย, รายงานการวิจัย การสร้างเครื่องควบคุมเสียงรบกวนแบบไวงาน, สาขาวิชาวิศวกรรมโทรคมนาคม, มหาวิทยาลัยเทคโนโลยีสุรนารี, 2546.

[2] Digital Signal Processing Implementations : Using DSP Microprecessors – with Examples form TMS320C54XX, Avtar Singh, 2004

[3] TMS320C54x DSP CPU and Peripherals Reference Set Volume 1, printed U.S.A., April 1999

[4] TMS320C54x Optimizing C/C++ Compiler User's Guide Literature Number: SPRU103G, October 2002

[5] TMS320C54x DSP Library Programmer's ReferenceLiterature Number: SPRU518D, October 2004

[6] http://www.ti.com/product/tms320vc5416

ประวัติผู้เขียน

นายธนสาร ศรีโคตร เกิดเมื่อวันที่ 15 สิงหาคม พ.ศ. 25 35ภูมิลำเนาอยู่ที่ ตำบล บัวใหญ่ อำเภอบัวใหญ่ จังหวัดนครราชสีมา สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียน บัวใหญ่ อำเภอบัวใหญ่ จังหวัดนครราชสีมา เมื่อปี พ.ศ. 25 52ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

นายพัชรพล ช่างเหล็ก เกิดเมื่อวันที่ 1 สิงหาคม พ.ศ. 25 34ภูมิลำเนาอยู่ที่ ตำบล ชากโดน อำเภอแกลง จังหวัดระยอง สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียน ระยองวิทยาคม อำเภอเมือง จังหวัดระยอง เมื่อปี พ.ศ. 25 52ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4 สาขาวิชาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี

ภาคผนวก

TMS320C5416 DSP Starter Kit (DSK)

TMDSDSK5416

Description

The TMS320C5416 DSP starter kit (DSK) is a low-cost development platform designed to speed the development of power-efficient applications based on TI's TMS320C54x DSPs. The kit, which provides new performance-enhancing features such as USB communications and true plug-and-play functionality, gives both experienced and novice designers an easy way to get started immediately with innovative product designs.

The C5416 DSK offers the ability to detect, diagnose and correct DSK communications issues, download and step through code faster and get a higher throughput with Real Time Data Exchange (RTDX™).

The kit includes:

• C5416 DSP Development Board -
• C5416 DSK Code Composer Studio™ v2.1 IDE
• Quick Start Guide
• Technical Reference
• Customer Support Guide
• USB Cable
• Universal Power Supply
• AC Power Cord(s)

Daughter Cards - Add new functionality to your TI or Third Party DSP development board. Evaluate a wide range of data converters, prototype boards, new I/O interfaces and many other peripheral devices. These compliant cards will allow you to connect with TI DSKs and some TI EVMs seamlessly via common connectors (TMS320 Cross-Platform Daughtercard Specification). Other Cards that are not compliant may require glue logic or special cabling to work with TI DSP development boards.

Features

The TMS320C5416 features the TMS320C5416 DSP - the designer's choice for applications that require an optimized combination of power performance and area. With 160 MIPS performance, designers can use the 160 MHz device as the foundation for a range of signal processing applications, including speech compression/decompression, speech recognition, text-to-speech conversion, fax/data conversion and echo cancellation. Other hardware features of the TMS320C5416 DSK board include:

- Embedded JTAG support via USB
- High-quality 16-/20-bit stereo codec
- Four 3.5mm audio jacks for microphone, line in, speaker and line out
- 256K words of Flash and 64K words RAM
- Expansion port connector for plug-in modules
- On-board standard JTAG interface
- +5V universal power supply

Software Tools - Designers can readily target the TMS32C5416 DSP through TI's robust and comprehensive Code Composer Studio development platform. The tools, which run on Windows© 98,

Windows 2000 and Windows XP, allow developers to seamlessly manage projects of any complexity. Code Composer Studio features for the TMS320C5416 DSK include:

• A complete Integrated Development Environment (IDE), an efficient optimizing C/C++ compiler assembler, linker, debugger, an a advanced editor with Code Mastro™ technology for faster code creation, data visualization, a profiler and a flexible project manager
• DSP/BIOS™ real-time kernel
• Target error recovery software
• DSK diagnostic tool
• "Plug-in" ability for third-party software for additional functionality

What's Included

• CCSTUDIO,Code Composer Studio (CCStudio) Integrated Development Environment (IDE) v5
• Target board specific