



เครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz

โดย

| | |
|------------------------|----------|
| นางสาวเมวิกา ยศวัฒน์นา | B5119493 |
| นางสาวมณฑิตา คงรอด | B5128457 |
| นายบารมี บุญยบาล | B5133383 |

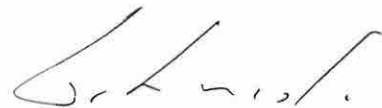
รายงานนี้เป็นส่วนหนึ่งของการศึกษาวิชา 427499 โครงการวิศวกรรมโทรคมนาคม
และวิชา 427494 โครงการวิศวกรรมโทรคมนาคม
หลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมโทรคมนาคม หลักสูตรปรับปรุง พ.ศ. 2546
สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีสุรนารี
ประจำภาคการศึกษาที่ 3 ปีการศึกษา 2554

เครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz

คณะกรรมการสอบโครงการ



(ผู้ช่วยศาสตราจารย์ ดร. ปิยาภรณ์ กระฉอดคนอก)
กรรมการ/อาจารย์ที่ปรึกษาโครงการ



(ผู้ช่วยศาสตราจารย์ ดร. ชานชัย ทองโสภ)
กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร. รังสรรค์ ทองทา)
กรรมการ

มหาวิทยาลัยเทคโนโลยีสุรนารี อนุมัติให้นำรายงานโครงการฉบับนี้ เป็นส่วนหนึ่งของการศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมโทรคมนาคม รายวิชา 427499 โครงการวิศวกรรมโทรคมนาคม และรายวิชา 427494 โครงการศึกษาวิศวกรรมโทรคมนาคม ประจำปีการศึกษา 2554

| | | | |
|------------------|---|-------------|----------|
| โครงการ | เครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ | | 1 GHz |
| โดย | นางสาวเมวิกา | ยศวฒนา รหัส | B5119493 |
| | นางสาวมณฑิตา | คงรอด รหัส | B5128457 |
| | นายบารมี บุญยุบล รหัส | | B5133383 |
| อาจารย์ที่ปรึกษา | ผู้ช่วยศาสตราจารย์ ดร. ปิยาภรณ์ กระจอดนอก | | |
| สาขาวิชา | วิศวกรรมโทรคมนาคม | | |
| ภาคการศึกษาที่ | 3/ 2554 | | |

บทคัดย่อ

โครงการนี้นำเสนอการออกแบบและการสร้างเครื่องวัดแบบรูปการแผ่ พลังงานของสายอากาศ โดยทำการออกแบบและสร้างวงจรรับคลื่นความถี่ที่แผ่ออกมาจากสายอากาศภาคส่งที่ความถี่ 1 GHz เพื่อนำมาวิเคราะห์ในรูปของค่าแรงดันแล้วแปลงให้เป็นสัญญาณดิจิทัลเพื่อทำการแสดงผลทางคอมพิวเตอร์โดยการใช้โปรแกรม Visual Basic Studio เพื่อใช้แสดงผลค่าการวัดแบบรูปการแผ่ พลังงานของสายอากาศ ซึ่งการหมุนของมอเตอร์จะถูกควบคุมด้วยวงจรไมโครคอนโทรลเลอร์

มหาวิทยาลัยเทคโนโลยีสุรนารี

กิตติกรรมประกาศ

โครงการฉบับนี้สามารถสำเร็จลุล่วงไปได้ด้วยดี เนื่องด้วยได้รับความกรุณาจากอาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร. ปิยาภรณ์ กระจงนอก ผู้ที่เป็นเจ้าของแนวคิด ในโครงการเรื่อง เครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz ซึ่งได้ให้ความช่วยเหลือเกี่ยวกับแนวคิด การดูแลเอาใจใส่ติดตามงาน ชี้แนะข้อบกพร่อง ตลอดจนช่วยฝึกฝนและให้การสนับสนุนคณะผู้จัดทำ ให้มีความสามารถในการทำโครงการจนเสนอผลงานให้เป็นที่รู้จักและยอมรับได้

ขอขอบพระคุณคณาจารย์และบุคลากรสาขาวิชาวิศวกรรมโทรคมนาคมทุกท่าน ที่ให้ความช่วยเหลือแก่คณะผู้จัดทำมาโดยตลอด พี่นักศึกษาปริญญาโท และเอกวิศวกรรมโทรคมนาคมและเพื่อนนักศึกษาสาขาวิศวกรรมโทรคมนาคมทุกคนที่เป็นกำลังใจให้มาโดยตลอด

คณะผู้จัดทำใคร่ขอขอบพระคุณทุกท่านที่ได้กล่าวมาข้างต้นไว้ ณ ที่นี้



เมวิกา ยศวัดนา

มณฑิตา กงรอด

บารมี บุญยุบล

สารบัญ

| | หน้า |
|---|------|
| บทคัดย่อ | ก |
| กิตติกรรมประกาศ | ข |
| สารบัญ | ค |
| สารบัญตาราง | ฉ |
| สารบัญรูปภาพ | ช |
| บทที่ 1 บทนำ | 1 |
| 1.1 ความเป็นมา | 1 |
| 1.2 วัตถุประสงค์ของโครงการ | 2 |
| 1.3 ขอบเขตการทำงาน | 2 |
| 1.4 ขั้นตอนการดำเนินงาน | 2 |
| บทที่ 2 ทฤษฎีที่เกี่ยวข้อง | 3 |
| 2.1 ความรู้เบื้องต้นและหลักการทำงานของสเต็ปมอเตอร์ | 3 |
| 2.1.1 โครงสร้างภายในของสเต็ปมอเตอร์ | 4 |
| 2.1.2 การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์แบบครึ่งสเต็ป (Half Step) | 4 |
| 2.1.3 วงจรขับสเต็ปมอเตอร์ (Drive Circuit) | 5 |
| 2.2 บอร์ดขับมอเตอร์ | 6 |
| 2.3 บอร์ดควบคุมการขับวงจรมอเตอร์ | 9 |
| 2.3.1 การใช้งาน AVR ATmega128 | 9 |
| 2.3.2 คุณสมบัติของบอร์ด | 9 |
| 2.3.3 โครงสร้างของบอร์ด | 10 |
| 2.3.4 ขั้วต่อสัญญาณต่างๆ | 11 |
| 2.4 วงจรขยายสัญญาณ (Amplifier) ที่ย่านความถี่ 1 GHz | 15 |
| 2.5 วงจรขยายสัญญาณด้วย Op-amp ที่ความถี่ 1 kHz | 15 |
| 2.6 วงจรเรียงกระแส (Rectifier) | 18 |
| 2.7 วงจร Envelop Detector | 18 |
| 2.8 การอ่านค่าแรงดันจากภาครับผ่านพอร์ต ADC | 19 |
| 2.9 สรุป | 19 |

สารบัญ (ต่อ)

| | หน้า |
|---|------|
| บทที่ 3 การออกแบบและสร้างวงจรวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz | 20 |
| 3.1 เครื่องกำเนิดสัญญาณ (RF Generator) | 20 |
| 3.2 สายอากาศภาคส่ง | 21 |
| 3.3 สายอากาศภาครับ | 21 |
| 3.4 วงจรควบคุมการหมุนของสายอากาศด้วยสเต็ปมอเตอร์ | 22 |
| 3.4.1 วงจรขับสเต็ปมอเตอร์ 23 | |
| 3.4.2 ชุดควบคุมการหมุน สายอากาศภาครับและสเต็ปมอเตอร์ 24 | |
| 3.5 วงจรขยายสัญญาณ (Amplifier) ที่ความถี่ 1 GHz | 25 |
| 3.6 วงจร self-biased detector | 25 |
| 3.7 วงจรขยายสัญญาณด้วย Op-Amp ที่ความถี่ 1 kHz และวงจรเรียงกระแส (Rectifier) | 28 |
| 3.8 การอ่านค่าแรงดันจากรับ | 33 |
| 3.9 การออกแบบและสร้างโปรแกรมวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz | 35 |
| บทที่ 4 การทดสอบ | 39 |
| 4.1 การติดตั้งอุปกรณ์และโปรแกรม 40 | |
| 4.2 ทำการรันโปรแกรม 42 | |
| 4.3 ขั้นตอนการใช้งาน 43 | |
| 4.4 ผลการวัดแบบรูปการแผ่พลังงานของสายอากาศ | 47 |
| 4.5 สรุปการทดสอบเครื่องวัดแบบรูปการแผ่พลังงานของสายอากาศ 5 | 4 |
| บทที่ 5 บทสรุป และข้อเสนอแนะ | 59 |
| 5.1 กล่าวนำ | 55 |
| 5.2 ปัญหาที่พบในระหว่างการทำโครงการและวิธีแก้ปัญหา | 55 |
| 5.3 ข้อเสนอแนะ | 56 |
| 5.4 แนวทางการพัฒนาต่อไป | 56 |
| 5.5 สิ่งที่ได้รับจากโครงการ | 57 |
| 5.6 บทสรุป | 57 |
| บรรณานุกรม | 59 |

สารบัญ (ต่อ)

| | หน้า |
|---|------|
| ภาคผนวก ก | 60 |
| โปรแกรม CodeVisionAVR C Compiler Evaluation | 60 |
| โปรแกรม Visual Basic Studio 2010 | 64 |
| ภาคผนวก ข | 74 |
| Datasheet | |
| -ET-BASE AVR AT mega64/128 r3 | 78 |
| ประวัติผู้เขียน | 98 |



สารบัญตาราง

| | หน้า |
|---|------|
| ตารางที่ 2.1 การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์แบบ Half Step | 5 |
| ตารางที่ 3.1 แสดงการเชื่อมต่อพอร์ตของไมโครคอนโทรลเลอร์กับบอร์ดขับเคลื่อนมอเตอร์ | 23 |
| ตารางที่ 3.2 ผลการทดสอบระดับแรงดันที่ค่า Power ต่าง ๆ ของวงจรถีเทคเตอร์ | 28 |
| ตารางที่ 3.3 แสดงผลการทดสอบระดับแรงดันที่ค่า Power ต่าง ๆ ของวงจรถีเทคเตอร์ วงจรถยายสัญญาณด้วยออปแอมป์และวงจรถเรียงกระแส | 33 |
| ตารางที่ 3.4 แสดงการเชื่อมต่อพอร์ต ADC กับ เอาท์พุทของวงจรถ้ากรับ | 34 |
| ตารางที่ 4.1 แสดงผลการทดสอบพลังงานของสายอากาศ | 50 |
| ตารางที่ 5.1 แสดงรายละเอียดของปัญหาที่พบ และวิธีแก้ปัญหของโครงการ | 55 |



สารบัญรูปภาพ

| | หน้า |
|-------------|------|
| รูปที่ 1.1 | |
| รูปที่ 2.1 | 3 |
| รูปที่ 2.2 | 4 |
| รูปที่ 2.3 | 4 |
| รูปที่ 2.4 | 5 |
| รูปที่ 2.5 | 6 |
| รูปที่ 2.6 | 7 |
| รูปที่ 2.7 | 8 |
| รูปที่ 2.8 | 8 |
| รูปที่ 2.9 | 8 |
| รูปที่ 2.10 | 9 |
| รูปที่ 2.11 | 10 |
| รูปที่ 2.12 | 13 |
| รูปที่ 2.13 | 13 |
| รูปที่ 2.14 | 14 |
| รูปที่ 2.15 | 15 |
| รูปที่ 2.16 | 15 |
| รูปที่ 2.17 | 17 |
| รูปที่ 2.18 | 18 |
| รูปที่ 2.19 | 19 |
| รูปที่ 3.1 | 20 |
| รูปที่ 3.2 | 20 |
| รูปที่ 3.3 | 21 |
| รูปที่ 3.4 | 22 |
| รูปที่ 3.5 | 22 |

และระนาบสนามแม่เหล็ก: H-Plane (x-y) ของสายอากาศไดโพล

สารบัญรูปภาพ (ต่อ)

| | หน้า |
|--|------|
| รูปที่ 3.6 ไมโครคอนโทรลเลอร์และชุดวงจรจับมือเตอร์ | 23 |
| รูปที่ 3.7 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ ATMEGA 128 และบอร์ดจับมือเตอร์ | 24 |
| รูปที่ 3.8 (ก) วงจรจับมือเตอร์ | 24 |
| (ข) วงจรควบคุมวงจรจับมือเตอร์ | |
| รูปที่ 3.9 ชุดควบคุมการหมุนของสายอากาศด้วยสเต็ปมอเตอร์ | 25 |
| รูปที่ 3.10 วงจรขยายสัญญาณ (Amplifier) ที่ความถี่ 1 GHz | 26 |
| รูปที่ 3.11 วงจร self-biased detector | 26 |
| รูปที่ 3.12 สัญญาณข่าวสารที่ถูกคิมอคูเลตแล้ว 28 | |
| รูปที่ 3.13 กราฟแสดงความสัมพันธ์ระหว่างพลังงาน และแรงดัน | 29 |
| รูปที่ 3.14 สัญญาณอินพุตของวงจขยายสัญญาณด้วยออปแอมป์ | 29 |
| รูปที่ 3.15 วงจรขยายสัญญาณด้วย Op-Amp แบบ Non-inverting | 29 |
| รูปที่ 3.16 สัญญาณเอาต์พุต | 30 |
| รูปที่ 3.17 วงจรเรียงกระแส | 31 |
| รูปที่ 3.18 ลายวงจขยายสัญญาณด้วยออปแอมป์ และวงจรเรียงกระแส | 31 |
| รูปที่ 3.19 วงจรขยายสัญญาณด้วยออปแอมป์ และวงจรเรียงกระแส | 31 |
| รูปที่ 3.20 สัญญาณเอาต์พุตเมื่อผ่านวงจขยายสัญญาณด้วย Op-Amp | 33 |
| และวงจรเรียงกระแส | |
| รูปที่ 3.21 แสดงความสัมพันธ์ระหว่างกำลังงานส่งและแรงดันหลังผ่านวงจรเรียงกระแส | 34 |
| รูปที่ 3.22 วงจรย่อยต่างๆ บรรจุรวมชุดอุปกรณ์ภาครับสัญญาณ | 35 |
| รูปที่ 3.23 Flow Chart ควบคุมการหมุนจากไมโครคอนโทรลเลอร์ (AVR ATmega 128) | 36 |
| รูปที่ 3.24 Flow Chart การทำงานของโปรแกรม Visual Basic Studio 2010 | 37 |
| รูปที่ 3.25 การออกแบบโปรแกรมวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz | 37 |
| รูปที่ 4.1 แสดงระบบการทดสอบทั้งหมด | 39 |
| รูปที่ 4.2 แสดงอุปกรณ์ภาคส่ง | 40 |
| รูปที่ 4.3 แสดงอุปกรณ์ภาครับ | 40 |
| รูปที่ 4.4 แสดงการเลือกใช้ความถี่บนเครื่องส่ง (Generator) | 41 |

สารบัญรูปภาพ (ต่อ)

| | หน้า |
|--|------|
| รูปที่ 4.5 แสดงการพร้อมใช้งานของอุปกรณ์ทั้งภาคส่งและภาครับ | 41 |
| รูปที่ 4.6 เปิดโปรแกรม Visual Basic Studio 2010 | 42 |
| รูปที่ 4.7 โปรแกรมวัดรูปแบบการแผ่พลังงานที่ความถี่ 1 GHz | 42 |
| รูปที่ 4.8 การเปิดโปรแกรม My Computer | 43 |
| รูปที่ 4.9 แสดงการเชื่อมต่อของ COM PORT กับ COMPUTER | 43 |
| รูปที่ 4.10 แสดงการเชื่อมต่อโปรแกรมพร้อมใช้งาน | 44 |
| รูปที่ 4.11 แสดงการเริ่มใช้งานโปรแกรม | 44 |
| รูปที่ 4.12 แสดงการหยุดโปรแกรมชั่วคราว | 45 |
| รูปที่ 4.13 เสร็จสิ้นการทำงาน | 45 |
| รูปที่ 4.14 แสดงการบันทึกข้อมูล | 46 |
| รูปที่ 4.15 แสดงการเคลียร์ค่าข้อมูล | 46 |
| รูปที่ 4.16 แสดงค่าพลังงานและมุม (องศา) | 47 |
| รูปที่ 4.17 แสดงค่าพลังงาน ในหน่วย dB และหาพลังงานสูงสุด | 47 |
| รูปที่ 4.18 แสดงการเลือกข้อมูลที่ใช้ในการพล็อตกราฟ | 48 |
| รูปที่ 4.19 แสดงแบบรูปการแผ่พลังงานที่ได้จากการพล็อตกราฟ | 48 |
| รูปที่ 4.20 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล เมื่อมอเตอร์ทำงานเสถียรที่ 0.9° | 50 |
| รูปที่ 4.21 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล เมื่อมอเตอร์ทำงานเสถียรที่ 1.8° | 51 |
| รูปที่ 4.22 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล เมื่อมอเตอร์ทำงานเสถียรที่ 3.6° | 52 |
| รูปที่ 4.23 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล เมื่อมอเตอร์ทำงานเสถียรที่ 7.2° | 53 |

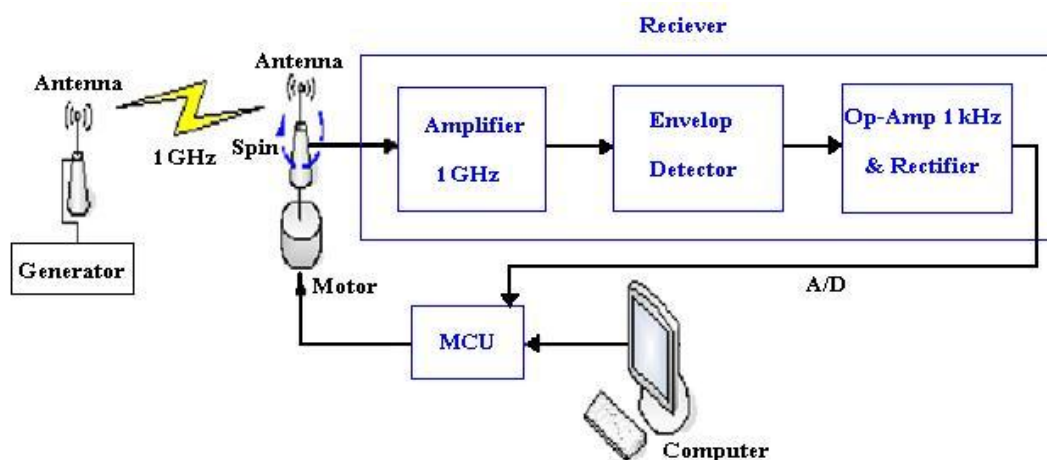
บทที่ 1

บทนำ

1.1 ความเป็นมา

เนื่องจากห้องปฏิบัติการวิศวกรรมโทรคมนาคม มหาวิทยาลัยเทคโนโลยีสุรนารี ไม่มีชุดอุปกรณ์สำหรับวัดแบบรูปการแผ่พลังงาน ดังนั้นในการทดลองจึงใช้คนในการหมุนปรับองศาเพื่อหาค่าแบบรูปการแผ่พลังงาน แล้วนำค่าที่ได้มาพล็อตกราฟ โดยในการทำการทดลองดังกล่าวนี้มีความยุ่งยากและเสียเวลา คณะผู้จัดทำจึงได้จัดทำโครงงานขึ้นมาเพื่อช่วยในการทำการทดลองให้มีความสะดวกสบายและมีความรวดเร็วมากขึ้น

เมื่อทำการจ่ายสัญญาณจากเครื่องส่งสัญญาณ (Generator) มาที่สายอากาศภาคส่ง ดังแสดงในรูปที่ 1.1 สายอากาศจะทำหน้าที่แปลงสัญญาณทางไฟฟ้า ให้เป็นคลื่นแม่เหล็กไฟฟ้า และแผ่กระจายคลื่นออกมา ซึ่งภาครับจะมีสายอากาศ ทำหน้าที่คอยรับสัญญาณ ในทิศทางต่าง ๆ ตั้งแต่ 0 - 360 องศา และส่งค่าสัญญาณมาที่เครื่องรับสัญญาณ ซึ่งประกอบด้วยวงจรถ่ายสัญญาณ ทำหน้าที่เพิ่มระดับสัญญาณ จากนั้นใช้วงจรดีเทคเตอร์ (Detector) เพื่อคิมอดสัญญาณข่าวสาร และขยายสัญญาณด้วยวงจรออปแอมป์ จากนั้นวงจรเรียงกระแสจะทำการแปลงสัญญาณกระแสสลับให้เป็นกระแสตรง เพื่อให้ทำงานกับบอร์ดไมโครคอนโทรลเลอร์ ในการกำหนดสตีปการหมุนนั้นใช้คำสั่งที่ออกแบบจากไมโครคอนโทรลเลอร์หรือโปรแกรม Code Vision AVR C Compiler Evaluation และทำการแสดงผลไปที่คอมพิวเตอร์โดยใช้ชุดคำสั่งที่ได้ทำการออกแบบโดยโปรแกรม Visual Basic Studio แสดงค่าการวัดซึ่งจะออกมาในแบบรูปการแผ่พลังงาน



รูปที่ 1.1 แผนผังไดอะแกรมของโครงการ

1.2 วัตถุประสงค์ของโครงการ

1. เพื่อศึกษาและออกแบบวงจรเพื่อใช้ในการออกแบบและสร้างวงจรรับคลื่นความถี่ 1 GHz
2. เพื่อควบคุมการหมุนของมอเตอร์ด้วยโปรแกรม Code Vision AVR C Compiler Evaluation
3. เพื่อแสดงผลการวัดแบบรูปการแผ่พลังงานด้วยโปรแกรม Visual Basic Studio 2010
4. เพื่อนำอุปกรณ์มาอำนวยความสะดวกในการนำไปใช้ทดลองในห้องปฏิบัติการ สาขาวิชาวิศวกรรมโทรคมนาคม

1.3 ขอบเขตการทำงาน

1. ศึกษาแบบรูปการแผ่พลังงานของสายอากาศ
2. ออกแบบและสร้างเครื่องวัดแบบรูปการแผ่พลังงานของสายอากาศ ให้เชื่อมต่อกับเครื่องคอมพิวเตอร์เพื่อใช้งานได้ตามวัตถุประสงค์ที่กำหนด
3. ออกแบบและสร้างวงจรรับที่ความถี่ 1 GHz พร้อมวงจรประกอบอื่นๆ
4. ออกแบบโปรแกรม Visual Basic Studio เพื่อรับค่าของแรงดันมาแสดงผลของแบบรูปการแผ่พลังงานของสายอากาศที่วัดได้ทางคอมพิวเตอร์

1.4 ขั้นตอนการดำเนินงาน

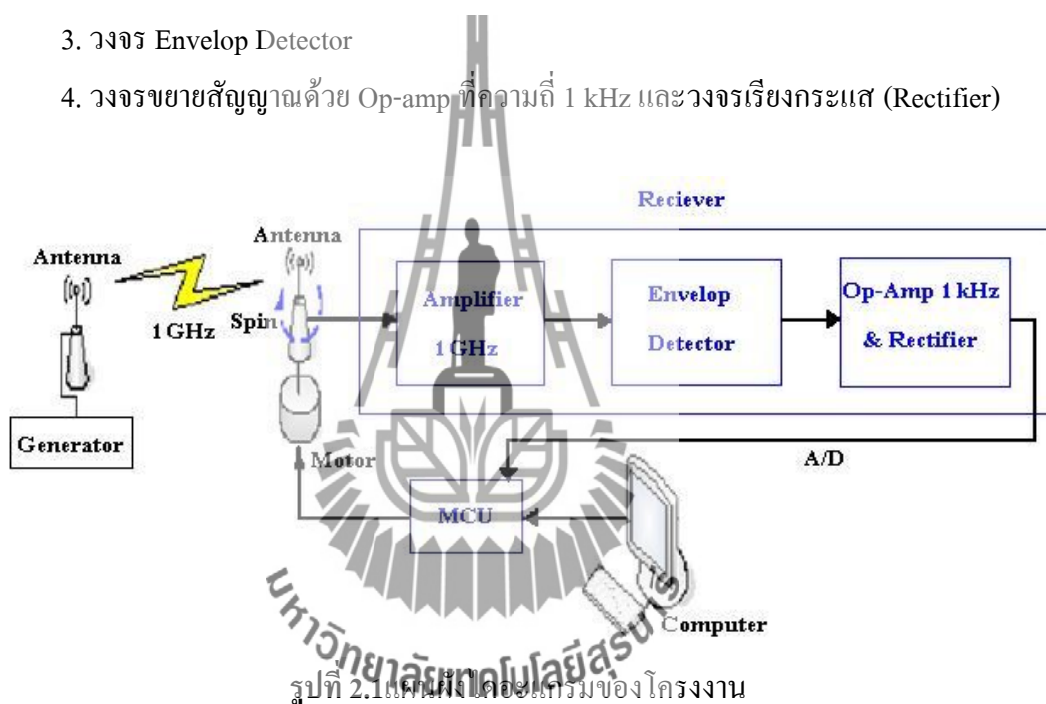
1. ศึกษาค้นหาข้อมูลเกี่ยวกับการสร้างเครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz
2. วางแผนโครงการในการสร้างเครื่องวัดแบบรูปการแผ่พลังงานของสายอากาศ
3. เขียนโครงการและนำเสนอโครงการกับอาจารย์ที่ปรึกษา
4. ทำการศึกษาการใช้งานโปรแกรม Visual Basic และ ไมโครคอนโทรลเลอร์
5. ทำการออกแบบเครื่องวัดและโปรแกรมในการควบคุม
6. ทำการสร้างอุปกรณ์เครื่องรับสัญญาณสายอากาศ
7. นำเครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz มาเชื่อมต่อกับเครื่องคอมพิวเตอร์
8. ทำการทดสอบและแก้ไขการทำงานของชุดอุปกรณ์
9. สรุปผลการทดลอง ประเมินผลและจัดทำรายงานโครงการ
10. นำเสนอโครงการ

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

โครงการนี้ออกแบบเครื่องวัดแบบรูปการแผ่พลังงานคลื่นที่ความถี่ 1 GHz โดยได้ทำการออกแบบและสร้างวงจรรับคลื่นความถี่ที่แผ่ออกมาจากสายอากาศภาคส่ง ที่ความถี่ 1GHzซึ่งวงจรภาครับมีองค์ประกอบที่สำคัญ4 ส่วน ดังแสดงในรูปที่ 2.1

1. วงจรควบคุมการหมุนของสายอากาศด้วยสเต็ปมอเตอร์
2. วงจรขยายสัญญาณ(Amplifier) ที่ความถี่1 GHz
3. วงจร Envelop Detector
4. วงจรขยายสัญญาณด้วย Op-amp ที่ความถี่ 1 kHz และวงจรเรียงกระแส (Rectifier)



รูปที่ 2.1 แผนผังโดยรวมของโครงการ

2.1 ความรู้เบื้องต้นและหลักการทำงานของสเต็ปมอเตอร์

สเต็ปมอเตอร์ เป็นมอเตอร์ที่มีลักษณะคือ เมื่อ เราป้อนแรงดัน ไฟฟ้าให้กับมอเตอร์ทำให้ หมุนเพียงเล็กน้อยตามเส้นรอบวงและหยุดซึ่ง ต่างจากมอเตอร์ทั่วไปที่จะหมุนทันทีและตลอดเวลา เมื่อป้อนแรงดัน ไฟฟ้าข้อดีของสเต็ปมอเตอร์คือสามารถกำหนดตำแหน่งของการหมุนได้ด้วยตัวเลข เป็นองศาหรือระยะทางอย่างละเอียดโดยใช้สัญญาณที่สร้างจาก คอมพิวเตอร์หรือ ไมโครคอนโทรลเลอร์

สเต็ปมอเตอร์เป็นมอเตอร์ที่ขับเคลื่อนด้วยพัลส์ ลักษณะการขับเคลื่อนโดยจะหมุนรอบ แกนได้ 360° มีลักษณะไม่ต่อเนื่อง แต่มีลักษณะเป็นสเต็ป โดยแต่ละสเต็ปจะขับเคลื่อนได้ลักษณะ

ต่าง ๆ เช่น 0.45° , 0.9° , 1.35° หรือ 1.8° ตามลำดับแล้วแต่ละโครงสร้างของมอเตอร์ ลักษณะที่นำมอเตอร์ไปใช้นั้นต้องการตำแหน่งที่แม่นยำ ดังรูปที่ 2.2 เป็นภาพของสเต็ปมอเตอร์



รูปที่ 2.2 มอเตอร์แบบมีสาย 4 เส้น

2.1.1 โครงสร้างภายในของสเต็ปมอเตอร์

โครงสร้างของขั้วแม่เหล็ก บนสเตเตอร์ (Stator) ทำมาจากแผ่นเหล็กวงแหวนที่มีซี่ยื่นออกมาประกอบกันเป็นชั้นๆ โดยที่แต่ละชั้นนั้นจะมีขดลวดพันสวมอยู่ เมื่อมีการป้อนกระแสผ่านขดลวด ทำให้เกิดสนามแม่เหล็กไฟฟ้า (Electromagnetic) ถ้าเพิ่มจำนวนของขั้วแม่เหล็กมากขึ้นจะเพิ่มจำนวนของสเต็ปต่อวงจรรอบมากขึ้นดังรูปที่ 2.3



รูปที่ 2.3 โครงสร้างของสเต็ปมอเตอร์

2.1.2 การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์แบบครึ่งสเต็ป (Half Step)

การควบคุมการหมุนของสเต็ปมอเตอร์แบบครึ่งสเต็ปนี้จะทำให้เราสามารถเพิ่มความละเอียดในการควบคุมการหมุนของสเต็ปมอเตอร์ได้อีกเท่าตัวทำให้เราสามารถควบคุมตำแหน่งในการหยุดของสเต็ปมอเตอร์ได้แม่นยำมากยิ่งขึ้น การควบคุมมอเตอร์แบบครึ่งสเต็ปนี้เป็น การผสมผสานระหว่าง การควบคุมแบบเต็มสเต็ป (Full Step) 1 เฟส กับการควบคุมแบบเต็มสเต็ป 2 เฟส เข้าไว้ด้วยกันดังตารางที่ 2.1

ตารางที่ 2.1 การสั่งงานควบคุมการหมุนของสเต็ปมอเตอร์แบบไฮบริดจ์ (Hybrid step)

| สเต็ปที่ | ขดลวดที่ 1 | ขดลวดที่ 2 | ขดลวดที่ 3 | ขดลวดที่ 4 |
|----------|------------|------------|------------|------------|
| 1 | ON | - | - | - |
| 2 | ON | ON | - | - |
| 3 | - | ON | - | - |
| 4 | - | ON | ON | - |
| 5 | - | - | ON | - |
| 6 | - | - | ON | ON |
| 7 | - | - | - | ON |
| 8 | ON | - | - | ON |

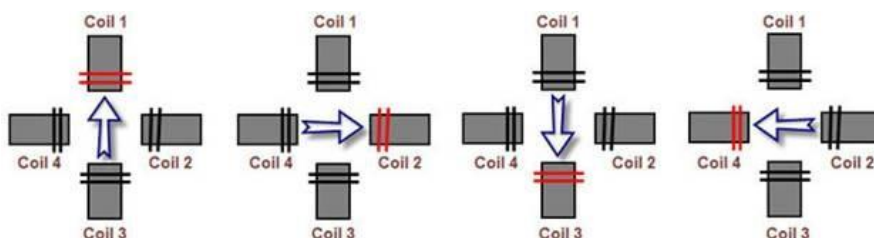
2.1.3 วงจรขับสเต็ปมอเตอร์ (Drive Circuit)

วงจรขับมอเตอร์ เป็นส่วนหนึ่งของระบบที่รับเอาคำสั่งจาก ไมโครคอนโทรลเลอร์ ที่เป็นสัญญาณอนาล็อกกระแสต่ำมาเพิ่มกระแสให้สูงขึ้นเพื่อขับมอเตอร์ โดยคุณสมบัติต่างๆ ของวงจรขับมอเตอร์ก็จะขึ้นอยู่กับขนาดและชนิดของมอเตอร์ที่ใช้

วิธีการขับสเต็ปมอเตอร์ให้หมุนโดยการกระตุ้นเฟส

ในการควบคุมสเต็ปมอเตอร์เพื่อที่จะให้ทำการหมุน มีวิธีการควบคุมกระแสไฟที่จ่ายให้กับขดลวดสเตเตอร์ในแต่ละเฟสของสเต็ปมอเตอร์ อย่างเป็นลำดับที่แน่นอน โดยถ้าหากเราต้องการให้กระแสไหลในเฟสใดๆ ก็จะทำให้สถานะของเฟสนั้นๆ เป็นสถานะลอจิก "1" และในการกระตุ้นเฟสของสเต็ปมีอยู่ด้วยกัน 2 แบบคือ

1. การกระตุ้นเฟสแบบเต็มสเต็ป (Full Step Drive) ซึ่งสามารถแบ่งการกระตุ้นเฟสออกได้ เป็นอีก 2 วิธีด้วยกันคือ

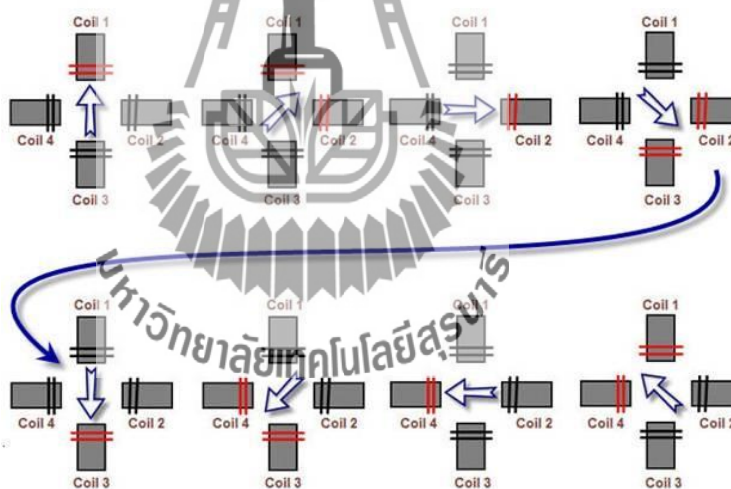


รูปที่ 2.4 การกระตุ้นเฟสแบบเต็มสเต็ป

1.1 การกระตุ้นเฟสแบบเต็มสเต็ป 1 เฟส (Single-Phase Driver) จะเป็นการป้อนกระแสไฟให้กับขดลวด ของสเต็ปมอเตอร์ทีละขด โดยจะป้อนกระแสเรียงตามลำดับกันไป ดังนั้น กระแสที่ไหลในขดลวดจะทำการไหลในทิศทางเดียวกันทุกขดลักษณะเช่นนี้จึงทำให้แรงขับของ สเต็ปมอเตอร์มีน้อย

1.2 การกระตุ้นเฟสแบบเต็มสเต็ป 2 เฟส (Two-Phase Driver) เป็นการป้อนกระแส ให้กับขดลวด 2 ขด ของสเต็ปมอเตอร์พร้อมๆกันไป และจะกระตุ้นเรียงถัดกันไปเช่นเดียวกับแบบ หนึ่งเฟส ดังนั้นการกระตุ้นแบบนี้จึงต้องใช้กำลังไฟมากขึ้น และจะทำให้มีแรงบิดของมอเตอร์ มากกว่าการกระตุ้นแบบ 1 เฟส

2. การกระตุ้นเฟสแบบ ครึ่งสเต็ป (Half Step Drive) คือการกระตุ้นเฟสแบบเต็มสเต็ป 1 เฟส และ 2 เฟส เรียงลำดับกันไป แรงบิดที่ได้จากการกระตุ้นเฟสแบบนี้จะมีเพิ่มมากขึ้น เพราะช่วงของ สเต็ปมีระยะสั้นลง ในการกระตุ้นแบบนี้ เราจะต้องมีการกระตุ้นที่เฟสถึง 2 ครั้ง จึงจะได้ระยะของสเต็ปเท่ากับการกระตุ้นเพียงครั้งเดียวของแบบเต็มสเต็ป 2 แบบแรกความละเอียดของการหมุน ตำแหน่งองศาต่อสเต็ป ก็เป็นสองเท่าของแบบแรก ความถูกต้องของตำแหน่งที่กำหนดจึงมีมากขึ้น



รูปที่ 2.5 การกระตุ้นเฟสแบบครึ่งสเต็ป

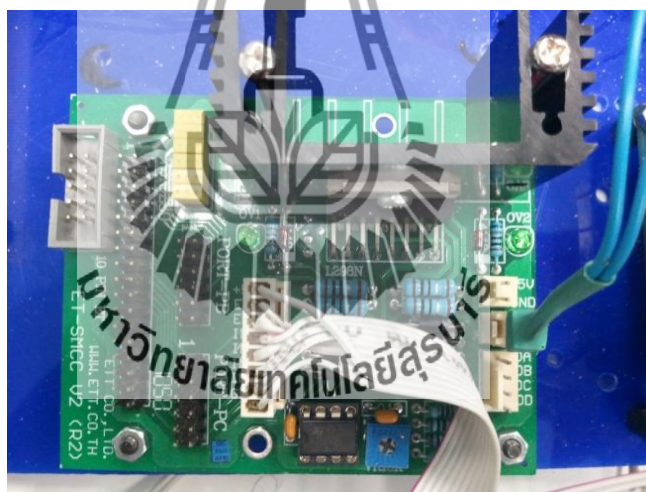
2.2 บอร์ดขับมอเตอร์

บอร์ด ET-SMCC V2.0 (R2) สามารถทำงานร่วมกับบอร์ดไมโครโปรเซสเซอร์และ ไมโครคอนโทรลเลอร์ต่างๆ ได้อย่างง่ายดาย โดยที่ตัวบอร์ดเองต้องการสัญญาณในการควบคุมการทำงานทั้งหมด 6 เส้นสัญญาณโดยสามารถรับสัญญาณ ลอจิกแบบ TTL มาตรฐานได้โดยตรง ซึ่ง บอร์ดสามารถขับกระแสให้กับ สเต็ปมอเตอร์แบบ Bipolar Stepper Motor ได้ทันที โดยไม่ต้อง คัดแปลงใดๆ โดยในส่วนของวงจรขับกระแสให้กับมอเตอร์บนบอร์ด ET-SMCC V2.0 (R2) นั้นจะ

ใช้ IC เบอร์ L298N ของ SGS-THOMSON และบอร์ด ET-SMCC V2.0 (R2) นี้จะใช้ได้กับสเต็ปมอเตอร์แบบ 2 ขั้ว หรือมอเตอร์ที่มีสาย 4 เส้นคือ Bipolar Stepper Motor เท่านั้น

คุณสมบัติของบอร์ด ET-SMCC V2.0 (R2)

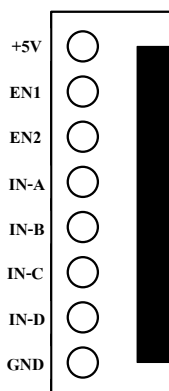
| | | |
|-----------------------|---|---|
| Channel Control | : | 2 Channel (1 Bipolar Stepper Motor หรือ 2 DC Motor) |
| Step Frequency | : | สูงสุด 40 kHz |
| Motor Control Type | : | Bipolar Stepper Motor หรือ DC Motor |
| Output Driver Current | : | 4 A / Phase |
| Input Logic Control | : | Standard TTL Logic Level |
| Power Supply | | |
| Logic Supply | : | +5 Vdc / 20mA |
| Motor Supply | : | Motor Supply (Maximum 50 Vdc / 4A) |
| PCB Size | : | 7.6 Cm. X 8.4 Cm. |



รูปที่ 2.6 แสดงบอร์ด ET-SMCC V2.0 (R2)

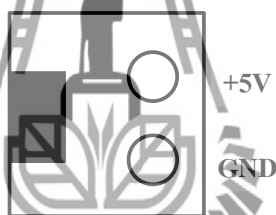
บอร์ด ET-SMCC V2.0 (R2) มีจุดเชื่อมต่อสัญญาณ INPUT/OUTPUT และแหล่งจ่ายไฟเลี้ยงรวมกันทั้งหมด 4 จุด โดยมีรายละเอียดดังนี้

CN1 เป็นขั้ว Connector แถวเดียวแบบ CPA ขนาด 8 PIN ใช้สำหรับเชื่อมต่อกับบอร์ดไมโครคอนโทรลเลอร์ เพื่อส่งสัญญาณมาควบคุมการทำงานของบอร์ด ET-SMCC V2.0 (R2)



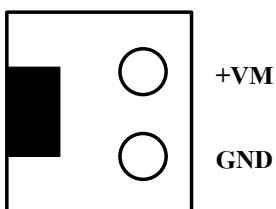
รูปที่ 2.7 แสดง CN1 เป็นขั้ว Connector แถวเดียวแบบ CPA ขนาด 8 PIN

CN2 เป็นขั้ว Connector ขนาด 2 PIN ใช้สำหรับต่อแหล่งจ่ายไฟเลี้ยงให้กับบอร์ด ET-SMCC V2.0 (R2) ซึ่งแหล่งจ่ายไฟเลี้ยงนี้ต้องเป็นขนาด +5 Vdc สามารถจ่ายกระแสได้ประมาณ 20 mA. เป็นอย่างน้อย โดยแหล่งจ่ายไฟที่จุดนี้จะเชื่อมต่อเป็นจุดเดียวกับ +5 Vdc ของ CN1 ดังนั้นในการเชื่อมต่ออาจเลือกใช้ไฟเลี้ยงจากจุดใดจุดหนึ่งเพียงจุดเดียวก็ได้



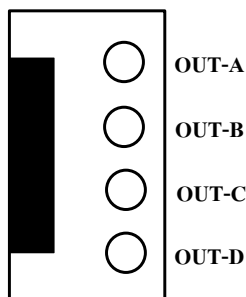
รูปที่ 2.8 CN2 เป็นขั้ว Connector ขนาด 2 PIN

CN3 เป็นขั้ว Connector ขนาด 2 PIN ใช้สำหรับเชื่อมต่อกับไฟเลี้ยงมอเตอร์ โดยขนาดของไฟเลี้ยงที่จุดนี้จะขึ้นอยู่กับขนาดความต้องการของมอเตอร์ว่าต้องการแหล่งจ่ายไฟขนาดเท่าไร (ขนาดสูงสุดที่ใช้ได้ต้องไม่เกิน 50 Vdc/4A)



รูปที่ 2.9 CN3 เป็นขั้ว Connector ขนาด 2 PIN

CN4 เป็นขั้ว Connector ขนาด 4 PIN ใช้สำหรับต่อกับขดลวดของมอเตอร์ ซึ่งมอเตอร์ที่จะใช้กับบอร์ด ET-SMCC V2.0 (R2) นี้ จะต้องเป็น DC Motor หรือ Stepper Motor แบบ 2 ขั้ว (Bipolar Stepper Motor) เท่านั้น



รูปที่ 2.10 CN4 เป็นหัว Connector ขนาด 4 PIN

ในกรณีที่เรากำลังต้องการต่อใช้งานกับ DC Motor นั้น จะสามารถใช้ได้กับ DC Motor 2 ตัว โดยต่อหัว OUT-A และ OUT-B เข้ากับมอเตอร์ตัวที่ 1 ส่วนหัว OUT-C และ OUT-D จะใช้ต่อกับมอเตอร์ตัวที่ 2 ซึ่ง DC Motor ทั้ง 2 ตัวที่ใช้จะต้องเป็น DC Motor ที่ใช้แหล่งจ่ายไฟเท่ากันด้วย

Note หากต้องการต่อใช้งานบอร์ด ET-SMCC V2.0 (R2) กับบอร์ด Basic Stamp ที่ชื่อ CP-BS2P40 และ CP-JRBS2P40 สามารถทำการเชื่อมต่อทางหัว 10 PIN ET ของทั้ง 2 บอร์ดได้ทันที และเมื่อทำการเขียนโปรแกรมเพื่อควบคุมมอเตอร์ให้ทำงานจะต้องใช้คำสั่งที่อ้างถึงขา AUXILIARY I/O ของ Basic StampBS2P40

2.3 บอร์ดควบคุมการขับเคลื่อนมอเตอร์

2.3.1 การใช้งาน AVR ATmega128

ET-BASE AVR ATmega64/128 r3 เป็นบอร์ดไมโครคอนโทรลเลอร์ในตระกูล AVR ของบริษัท Atmel ซึ่งบอร์ดนี้เลือกใช้ MCU เบอร์ ATmega64 และเบอร์ ATmega128 ขนาด 64 Pin โดยในบอร์ด ET-BASE AVR ATmega64/128 r3 นี้จะเน้นการใช้งานทรัพยากรของตัว MCU เองเป็นหลักซึ่งจะมีการต่อขาสัญญาณ I/O ออกมาจัดเรียงให้เป็นพอร์ต PA, PB, PC, PD, PE, PF และพอร์ต ET-CLCD เพื่อสะดวกต่อการใช้งานพร้อมทั้งพอร์ตสำหรับดาวน์โหลดโปรแกรมนอกจากนี้ยังได้เพิ่มวงจร Line Driver RS-232 เข้าไปด้วยเพื่อให้สามารถใช้งานทางด้านพอร์ตอนุกรม RS-232 ได้ง่ายและสะดวกยิ่งขึ้น

2.3.2 คุณสมบัติของบอร์ด

1. เลือกใช้ MCU ตระกูล AVR เบอร์ ATmega64, ATmega128 ของ Atmel ซึ่งเป็น MCU ขนาด 8-Bit โดยเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาแบบ XTAL ค่า 16 MHz ซึ่งคุณสมบัติเด่นๆ ของ MCU ได้แก่ มีหน่วยความจำ Flash สำหรับเขียนโปรแกรม 64 Kbytes สำหรับ ATmega64 และ 128K Bytes สำหรับ ATmega128 และมี RAM 4 Kbytes และมีหน่วยความจำข้อมูลถาวรแบบ

EEPROM ขนาด 2K Bytes สำหรับ ATmega64 และ 4K Bytes สำหรับ ATmega128 ซึ่งสามารถลบและเขียนซ้ำได้กว่า 100,000 ครั้ง

- จำนวน I/O สูงสุดถึง 53 I/O Pins
- มีวงจรสื่อสาร SPI จำนวน 1 ช่อง, I2C จำนวน 1 ช่อง, Programmable Serial USARTs จำนวน 2 ช่อง
- มี ADC ขนาด 10-Bit จำนวน 8 ช่อง
- มี Timers/Counters 8-Bit จำนวน 2 ช่อง, Timers/Counters 16-Bit จำนวน 2 ช่อง, 8-Bit PWM 2 ช่อง, Watchdog Timer, Real Time Counter

2. I/O PORT 10 PIN จำนวน 6 PORT ดังนี้ PA, PB, PC, PD, PE, PF

3. พอร์ต ISP LOAD สำหรับโปรแกรม MCU (ต้องใช้ร่วมกับ ET-AVR ISP หรือเครื่องโปรแกรม ISP อื่นที่มีการจัดเรียงขาสัญญาณเหมือนกัน)

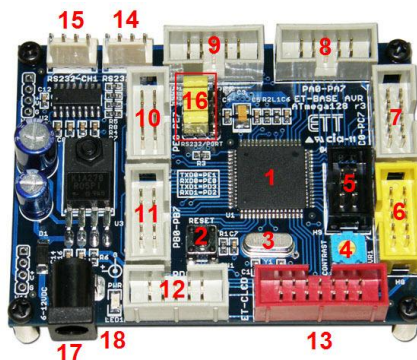
4. วงจร Line Driver สำหรับพอร์ตสื่อสารอนุกรม RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่องจะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1) เพื่อให้ผู้ใช้สามารถต่อทดลองการติดต่อสื่อสาร RS232

5. วงจรเชื่อมต่อแสดงผล LCD แบบ Character (ET-CLCD) พร้อม VR ปรับความเข้มของ LCD ซึ่งใช้การเชื่อมต่อวงจรกับ LCD แบบ 4 Bit Interface

6. วงจร Regulate ขนาด +5V / 2A สำหรับใช้งานเป็นแหล่งจ่ายไฟเลี้ยงวงจรให้กับจอแสดงผล LCD และอุปกรณ์ I/O ต่างๆ ที่ใช้กับแหล่งจ่ายขนาด +5V พร้อม LED แสดงสถานะสีแดง

7. ขนาด PCB Size เล็กเพียง 8x6 cm

2.3.3 โครงสร้างของบอร์ด



รูปที่ 2.11 โครงสร้างบอร์ด AVR ATmega128

- หมายเลข 1 คือ MCU เบอร์ ATmega64 หรือ ATmega128 ซึ่งเป็น MCU ตระกูล AVR จาก ATMEL
- หมายเลข 2 คือ Switch RESET ใช้สำหรับ Reset การทำงานของ MCU
- หมายเลข 3 คือ Crystal ค่า 16 MHz
- หมายเลข 4 คือ ตัวต้านทานสำหรับปรับค่าความเข้มให้ LCD
- หมายเลข 5 พอร์ต AVR ISP (6 PIN) ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU
- หมายเลข 6 พอร์ต AVR ISP (10 PIN) ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU
- หมายเลข 7 คือ PORTC มีขนาด 8 Bit คือ PC0-PC7
- หมายเลข 8 คือ PORTA มีขนาด 8 Bit คือ PA0-PA7
- หมายเลข 9 คือ PORTF มีขนาด 8 Bit คือ PF0-PF7
- หมายเลข 10 คือ PORTE มีขนาด 8 Bit คือ PE0-PE7
- หมายเลข 11 คือ PORTB มีขนาด 8 Bit คือ PB0-PB7
- หมายเลข 12 คือ PORTD มีขนาด 8 Bit คือ PD0-PD7
- หมายเลข 13 คือ พอร์ต ET-CLCD สำหรับเชื่อมต่อกับ LCD ชนิด Character Type ซึ่งใช้การเชื่อมต่อแบบ 4 Bit
- หมายเลข 14 และ 15 คือ ขั้วต่อ RS232 สำหรับใช้งานทั่วไป
- หมายเลข 16 คือ จัมเปอร์สำหรับเลือกใช้งาน RS232 หรือ พอร์ต IO
- หมายเลข 17 คือ ขั้วต่อแหล่งจ่ายไฟสำหรับเลี้ยงวงจรของบอร์ด
- หมายเลข 18 คือ LED Power ซึ่งใช้แสดงสถานะของแหล่งจ่ายไฟ +5VDC

2.3.4 ขั้วต่อสัญญาณต่างๆ

สำหรับขั้วต่อสัญญาณของพอร์ต I/O จาก MCU นั้นจะถูกออกแบบและจัดเตรียมไว้ผ่านทางขั้วต่อแบบ IDC-Header ขนาด 10 Pin (2×5) จำนวน 6 ชุดคือ PA, PB, PC, PD, PE, PF ตามลำดับ โดยที่ขั้วต่อสัญญาณแต่ละชุดจะประกอบไปด้วยสัญญาณของ I/O ที่เชื่อมต่อมาจากขาสัญญาณของ MCU โดยตรงทั้งหมด โดยจุดเชื่อมต่อกับสัญญาณภายนอกบอร์ดมีดังนี้

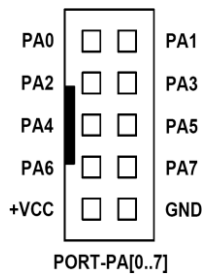
- ขั้วต่อแหล่งจ่ายไฟสำหรับเลี้ยงวงจรของบอร์ด
- ขั้วต่อ PORTA มีขนาด 8 Bit คือ PA0-PA7
- ขั้วต่อ PORTB มีขนาด 8 Bit คือ PB0-PB7
- ขั้วต่อ PORTC มีขนาด 8 Bit คือ PC0-PC7

- ขั้วต่อ PORTD มีขนาด 8 Bit คือ PD0-PD7
- ขั้วต่อ PORTE มีขนาด 8 Bit คือ PE0-PE7
- ขั้วต่อ PORTF มีขนาด 8 Bit คือ PF0-PF7
- ขั้วต่อ ET-CLCD สำหรับเชื่อมต่อกับ LCD ชนิด Character Type
- ขั้วต่อ RS232 จำนวน 2 ช่องโดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0)

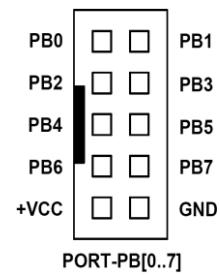
จำนวน 1 ช่องส่วนที่เหลืออีก 1 ช่องจะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1) เพื่อให้ผู้ใช้สามารถทดลองการติดต่อสื่อสาร RS232

- ขั้วต่อ AVR ISP ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU

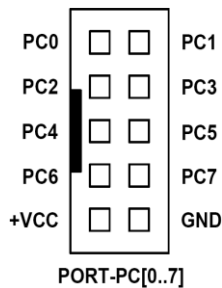
พอร์ต PA มีขนาด 8 บิต



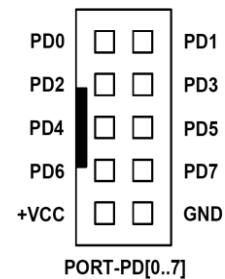
พอร์ต PB มีขนาด 8 บิต



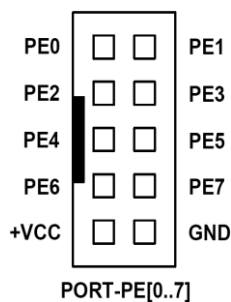
พอร์ต PC มีขนาด 8 บิต



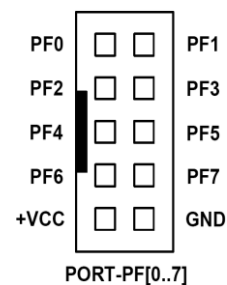
พอร์ต PD มีขนาด 8 บิต



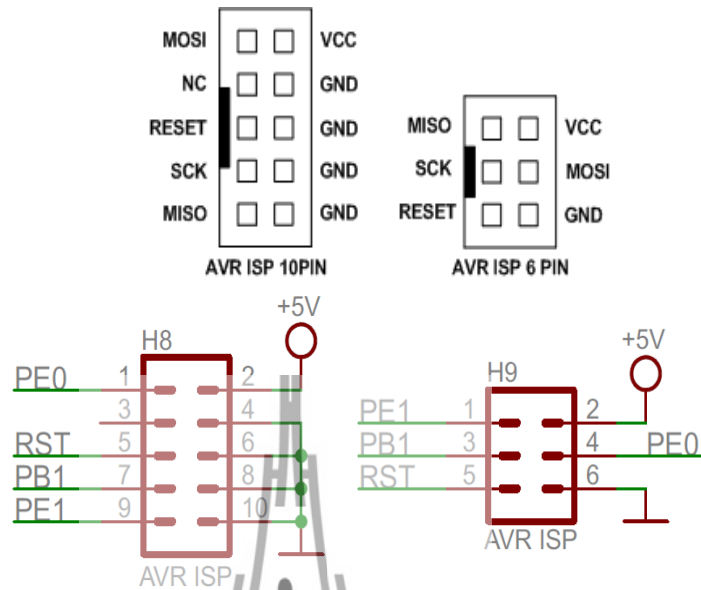
พอร์ต PE มีขนาด 8 บิต



พอร์ต PF มีขนาด 8 บิต

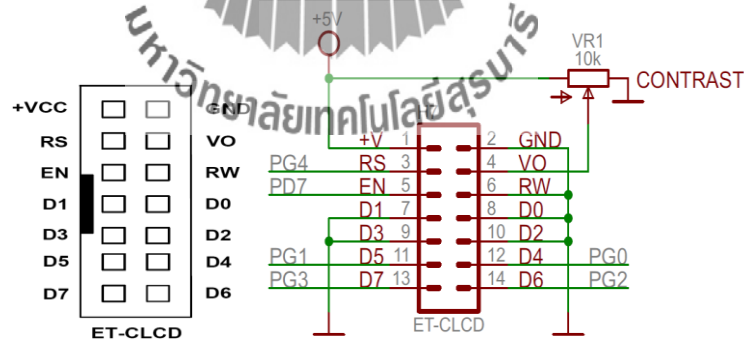


พอร์ต AVR ISP



รูปที่ 2.12 วงจรส่วนที่เชื่อมต่อกับ AVR ISP

พอร์ต ET-CLCD ใช้กับ Character Type LCD โดยใช้ในการเชื่อมต่อแบบ 4 บิตโดยสัญญาณที่ใช้เชื่อมต่อกับ LCD จะเป็นสัญญาณจากพอร์ต PG และ PD (PD7) โดยในการเชื่อมต่อสายสัญญาณจากขั้วต่อของพอร์ต LCD ไปยังจอแสดงผล LCD นั้นให้ยึดชื่อขาสัญญาณเป็นจุดอ้างอิงโดยให้ต่อสัญญาณที่มีชื่อตรงกันเข้าด้วยกันให้ครบทั้ง 14 เส้น

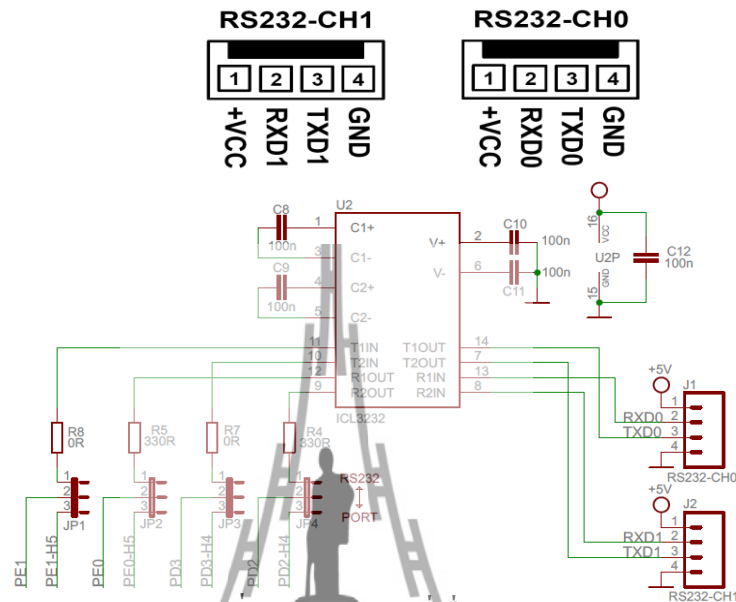


| | | | | | | | | | | | | | |
|-----|------|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| GND | +VCC | VO | RS | RW | EN | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |

รูปที่ 2.13 รายละเอียดของพอร์ตเชื่อมต่อ ET-CLCD

การจัดเรียงขาสัญญาณของ Character LCD มาตรฐาน

พอร์ต RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ส่วนที่เหลืออีก 1 ช่องจะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1)

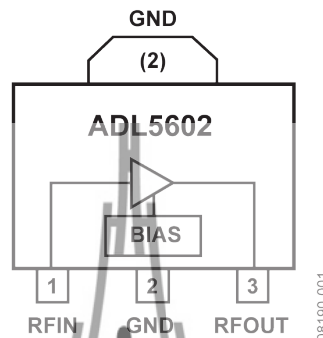


รูปที่ 2.14 วงจรส่วนที่เชื่อมต่อกับ RS232

การดาวน์โหลด Hex File ให้กับ MCU นั้นจำเป็นจะต้องใช้ ET-AVR ISP หรือเครื่องโหลดโปรแกรมแบบ ISP อื่นๆเช่น AVRISP ของ ATMEL เพื่อใช้ในการดาวน์โหลด Hex File ให้กับ MCU ตระกูล AVR ของ Atmel โดยใช้วิธีการแบบ Serial Programming ซึ่งการดาวน์โหลด Hex File ในกรณีที่ใช้ ET-AVR ISP จะอาศัยช่องทางทางพอร์ตขนานของคอมพิวเตอร์โดยที่จะต้องใช้งานร่วมกับ ETCAP10P ของอีทีทีและ Software ที่ใช้ร่วมกับ ET-AVR ISP ก็คือ PonyProg2000 ซึ่ง PonyProg2000 เป็นโปรแกรม Download ข้อมูลแบบ HEX File ให้กับ CPU ตระกูล AVR โดยใช้วิธีการแบบ Serial Programming ซึ่งสามารถใช้งานกับบอร์ดตระกูล AVR ของอีทีทีได้เป็นอย่างดี ซึ่งวิธีการใช้งานโปรแกรมโดยทั่วไปนั้นสามารถศึกษาได้จาก Help ของโปรแกรมได้เอง

2.4 วงจรขยายสัญญาณ(Amplifier) ที่ย่านความถี่1GHz

วงจขยายสัญญาณ (Amplifier)ทำหน้าที่ขยายสัญญาณที่ออกมาจากสายอากาศภาค รับให้มีค่าสูงมากเพียงพอที่จะใช้งาน กับวงจรถิเทคเตอร์เพื่อให้วงจรถิเทคเตอร์สามารถ แยกสัญญาณคลื่นพาหะความถี่ 1 GHz ออกให้เหลือเพียงสัญญาณข้อมูลความถี่ 1 kHz โดยใช้MMIC เบอร์ ADL5602 มีคุณสมบัติคือ จะทำงานที่แรงดันไฟฟ้า 4.75V – 5.25V กำลังงานไฟฟ้าขาเข้า ไม่เกิน 16 dBm อุณหภูมิ $-40^{\circ}C \leq T_a \leq +85^{\circ}C$ และอัตราขยายเฉลี่ย 20.2 dB

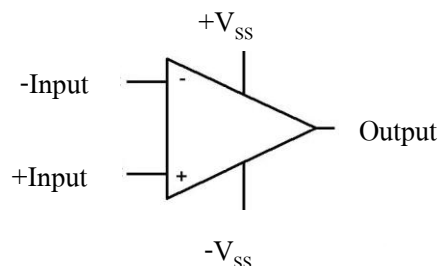


รูปที่ 2.15 วงจรภายใน MMIC เบอร์ ADL5602

2.5 วงจรขยายสัญญาณด้วย Op-amp ที่ความถี่ 1 kHz

ออปแอมป์เป็นไอซีแบบหนึ่งของตระกูลไอซีแบบลิเนียร์ที่ทำหน้าที่ได้สารพัดประโยชน์ แต่โดยพื้นฐานแล้วออปแอมป์ถูกสร้างขึ้นมาเพื่อขยายสัญญาณออปแอมป์จะขยายความแตกต่างระหว่างแรงเคลื่อนหรือสัญญาณ (AC หรือ DC) ที่ป้อนเข้าที่อินพุตทั้งสองแรงเคลื่อนหรือสัญญาณที่ต้องการจะขยายสามารถป้อนเข้าทางอินพุตใดอินพุตหนึ่งหรือทั้งสองอินพุตก็ได้สัญลักษณ์และชื่อขาใช้งานพื้นฐานของออปแอมป์แสดงดังรูปที่ 2.16 โดยประกอบด้วย

- ขั้วอินพุตบวก(Non-inverting)
- ขั้วอินพุตลบ(Inverting)
- ขั้วเอาต์พุต(Output)
- ขั้วแรงดันไฟเลี้ยงบวกและลบซึ่งปกติไม่ได้แสดงในสัญลักษณ์

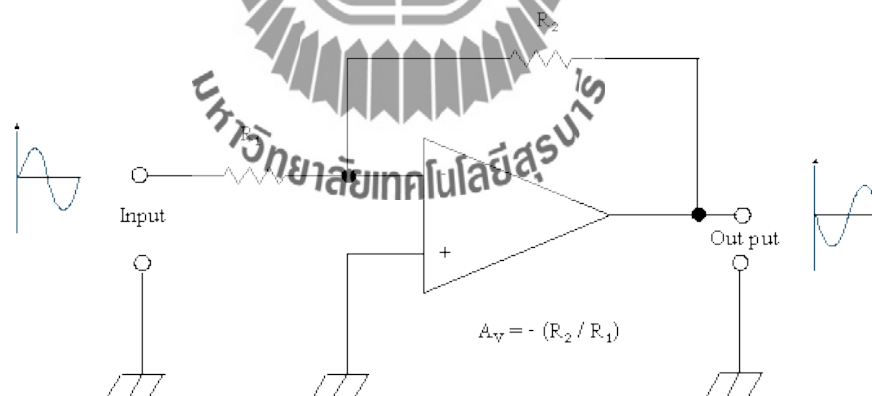


รูปที่ 2.16 สัญลักษณ์และชื่อขาใช้งานพื้นฐานของออปแอมป์

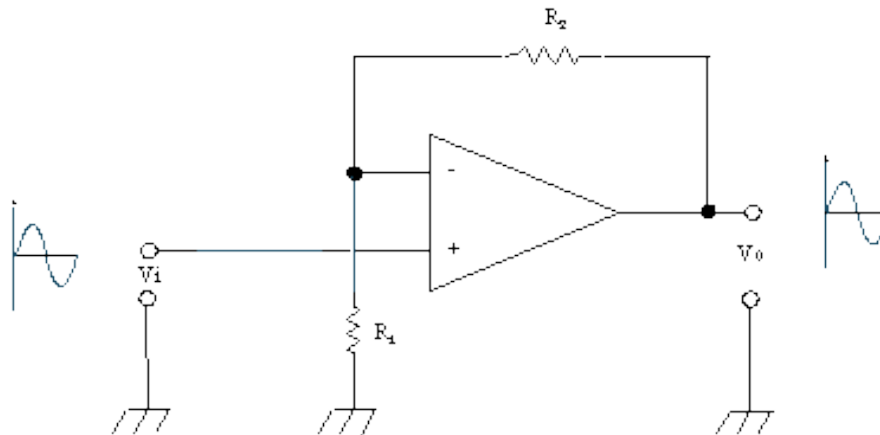
คุณสมบัติของออปแอมป์ในทางอุดมคติ

1. อัตราขยายมีค่าสูงมากเป็นอนันต์ ($AV = \infty$)
2. อินพุตอิมพีแดนซ์มีค่าสูงมากเป็นอนันต์ ($Z_i = \infty$)
3. เอาท์พุตอิมพีแดนซ์มีค่าต่ำมากเท่ากับศูนย์ ($Z_o = 0$)
4. ความกว้างของแบนด์วิดท์ (Bandwidth) ในการขยายสูงมาก ($BW = \infty$)
5. สามารถขยายสัญญาณได้ทั้งสัญญาณ AC และ DC
6. การทำงานไม่ขึ้นกับอุณหภูมิ

เมื่อศึกษาคุณสมบัติของออปแอมป์ในอุดมคติแล้วพบว่า ออปแอมป์ได้รวมข้อดีของ วงจรขยายไว้ได้อย่างครบถ้วน เนื่องจากมีอัตราขยายเป็นอนันต์และสามารถขยายสัญญาณได้ทั้ง ไฟฟ้ากระแสสลับและไฟฟ้ากระแสตรง การนำไปใช้งานในบางครั้งเมื่อต้องการลดอัตราขยายก็สามารถกระทำได้โดยการป้อนกลับ (Feed Back) เพื่อมาลดอัตราขยายลง และข้อดีอีกประการหนึ่งก็คือ อินพุตอิมพีแดนซ์สูงมาก จึงทำให้เหมือนไม่มีกระแสอินพุตไหลเลยลักษณะเช่นนี้จึงทำให้วงจรทางอินพุตไม่ไหลตรงส่งกำลังในส่วนหน้า เช่นเดียวกันที่เอาท์พุตอิมพีแดนซ์เป็นศูนย์สามารถนำไปเชื่อมต่อกับวงจรอื่นได้ดี ดังแสดง วงจรขยายออปแอมป์แบบกลับเฟส (Inverting Amplifier) ในรูปที่ 2.17 (ก) และ วงจรขยายออปแอมป์แบบไม่กลับเฟส (Non-Inverting Op-amp) ในรูปที่ 2.17 (ข)



(ก) วงจรขยายออปแอมป์แบบกลับเฟส (Inverting Amplifier)



(ข) วงจรขยายออปแอมป์แบบไม่กลับเฟส(Non-Inverting Op-amp)

รูปที่ 2.17 วงจรขยายออปแอมป์

วงจรขยายออปแอมป์แบบกลับเฟส (Inverting Amplifier) จากรูปที่ 2.17(ก) ในวงจรขยายออปแอมป์นั้นสามารถที่จะกำหนดอัตราการขยายของวงจรได้โดยการใช้ วงจรเนกาทีฟฟีดแบ็ค (Negative Feedback) เมื่อเราป้อนสัญญาณเข้าทางขากลับเฟส (ขา -) แรงดันด้านทางออกจะมีมุมเฟสต่างไปจากแรงดันทางเข้า 180 องศา ซึ่งมีลักษณะตรงข้าม สัญญาณตรงกันข้ามนี้จะถูกป้อนกลับผ่าน R_2 เข้ามายังขาอินเวอร์ตอีกครึ่งหนึ่ง ตรงจุดนี้จะทำให้สัญญาณเกิดการหักล้างกัน อัตราการขยายก็จะลดลง ถ้าตัวต้านทานที่เป็นตัวป้อนกลับมีค่ามาก จะทำให้สัญญาณป้อนกลับมีขนาดเล็ก อัตราการขยายออกจึงสูง ถ้าตัวต้านทานที่ป้อนกลับมีค่าน้อยสัญญาณป้อนกลับไปได้มาก อัตราการขยายก็จะลดลง ฉะนั้นอัตราส่วนของความต้านทาน R_1 และ R_2 จะเป็นตัวกำหนดอัตราการขยายของวงจร โดยไม่ขึ้นกับอัตราการขยายของออปแอมป์ ซึ่งสามารถหาอัตราการขยายแรงดันได้จากสูตร

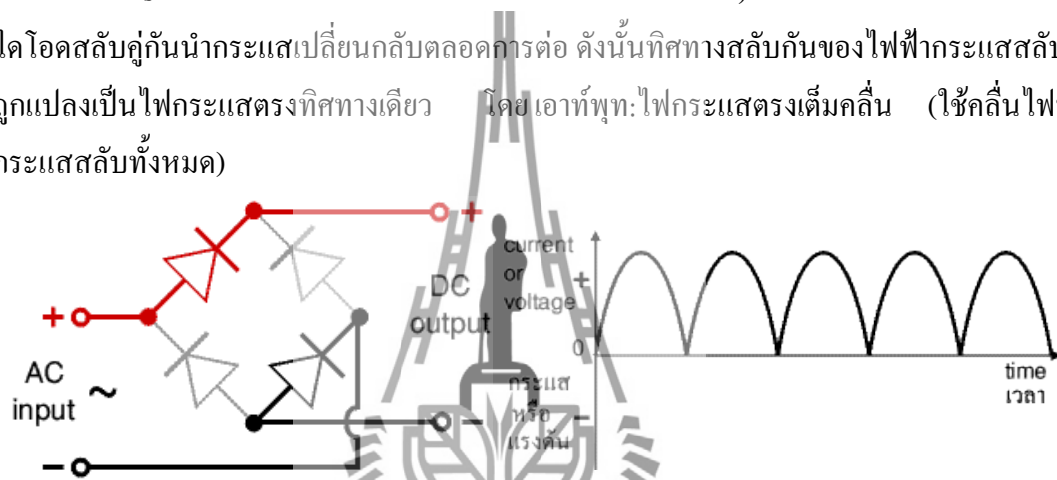
$$A_v = -\frac{R_2}{R_1} \quad (2.1)$$

วงจรขยายออปแอมป์แบบไม่กลับเฟส Non-Inverting Op-amp จากรูปที่ 2.17(ข) วงจรขยายนี้เป็นวงจรขยายอีกแบบหนึ่งที่ต้องการเฟสในการขยายเป็นเฟสเดียวกัน ดังนั้นการป้อนสัญญาณอินพุตจึงต้องป้อนเข้าที่ขาอินพุตไม่กลับเฟส (+) ซึ่งเมื่อขยายออกที่เอาต์พุตแล้วจะได้สัญญาณเอาต์พุตที่มีเฟสเหมือนเดิม ดังนั้นในวงจรขยายแบบไม่กลับเฟสนี้การป้อนกลับเพื่อลดอัตราการขยายจึงยังคงต้องป้อนไปยังขาอินเวอร์ต (-) เพื่อให้เกิดการหักล้างของสัญญาณกันภายในตัวไอซีออปแอมป์ โดยสามารถหาอัตราการขยายของวงจรได้จากสูตร

$$A_v = \frac{R_f}{R_i} + 1 \quad (2.2)$$

2.6 วงจรเรียงกระแส (Rectifier)

วงจรเรียงกระแสแบบบริดจ์ (Bridge rectifier) สามารถใช้ไดโอดเดี่ยวสี่ตัวมาต่อกันหรือสามารถใช้ไดโอดบริดจ์แบบแพ็คเกจสำเร็จรูปก็ได้ เรียกว่าการเรียงกระแสแบบเต็มคลื่นเพราะใช้คลื่นไฟฟ้ากระแสสลับทั้งหมด(ทั้งด้านบวกและด้านลบ) ตัวเรียงกระแสแบบบริดจ์จะเกิดแรงดันตกคร่อม 1.4V เพราะไดโอดแต่ละตัวจะตกคร่อมเท่ากับ 0.7V ขณะนำกระแสและบริดจ์มีการนำกระแสสองตัวพร้อมกันดังแสดงในรูปที่ 2.18 ตัวเรียงกระแสแบบบริดจ์จัดแบ่งตามกระแสสูงสุดที่สามารถผ่านได้และแรงดันกลับสูงสุดที่ทนได้ (ในการเลือกใช้งานอย่างน้อยต้องสูงเป็นสามเท่าของแรงดันแหล่งจ่าย rms นั้นคือวงจรเรียงกระแสจะสามารถทนแรงดันขอคได้) วงจรเรียงกระแสแบบบริดจ์ไดโอดสลับคู่กันนำกระแสเปลี่ยนกลับตลอดการต่อ ดังนั้นทิศทางสลับกันของไฟฟ้ากระแสสลับจึงถูกแปลงเป็นไฟกระแสตรงทิศทางเดียว โดยเอาท์พุท: ไฟกระแสตรงเต็มคลื่น (ใช้คลื่นไฟฟ้ากระแสสลับทั้งหมด)



รูปที่ 2.18 วงจรเรียงกระแสแบบบริดจ์และเอาท์พุท: ไฟกระแสตรงเต็มคลื่น

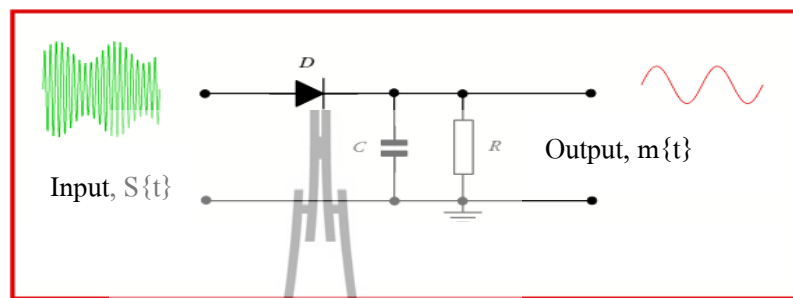
2.7 วงจร Envelop Detector

วงจร Envelope Detector ใช้สำหรับ การ คัดเลือกสัญญาณ AM ซึ่งทำหน้าที่คัดสัญญาณพาหะออก เหลือเฉพาะ สัญญาณข่าวสาร ส่งต่อไปยังภาคขยายสัญญาณด้วยออปแอมป์และวงจรเรียงกระแสต่อไป โดยวงจร Envelop Detector ประกอบด้วย ไดโอด และวงจรกรองสัญญาณความถี่ต่ำ (Low Pass Filter) ดังแสดงในรูป 2.19

สำหรับวงจรกรองความถี่ต่ำ เป็นวงจรกรองสัญญาณไฟฟ้าที่ยอมให้ความถี่ตั้งแต่ 0 Hz ถึงความถี่ f_c (ความถี่ Cutoff คือความถี่ที่วงจรกรองยอมให้กำลังของสัญญาณผ่านได้ครึ่งหนึ่งของกำลังที่ยอมให้ผ่านได้สูงสุด) ผ่านไปยังขั้วเอาท์พุทของวงจรได้ ส่วนความถี่ที่สูงกว่า f_c ความถี่จะไม่ผ่านไปยังขั้วเอาท์พุทของวงจร

วงจรกรองสัญญาณไฟฟ้าแบบพาสซีฟจะประกอบไปด้วยอุปกรณ์ทางพาสซีฟ (Passive Device) เป็นหลักได้แก่ ตัวเก็บประจุ และตัวเหนี่ยวนำ (บางครั้งอาจจะมีตัวต้านทานประกอบรวมอยู่ด้วย) ข้อดีของวงจรกรองสัญญาณไฟฟ้าแบบพาสซีฟคือ สามารถตอบสนองความถี่ได้สูงมาก

และสามารถใช้งานได้โดยไม่ต้องใช้แหล่งจ่ายไฟใดๆทั้งสิ้น ในความเป็นจริงแล้ว สัญญาณที่ออกมาจากเอาต์พุตของวงจรกรองสัญญาณไฟฟ้าแบบพาสซีฟจะเกิดการสูญเสียขึ้นเนื่องจากค่าอิมพีแดนซ์ของวงจรและเมื่อพิจารณาถึงการส่งผ่านของแถบความถี่จะบ่งบอกได้อย่างชัดเจนว่าเป็นวงจรที่มีการส่งผ่านไม่ดีนัก อย่างไรก็ตามสามารถแก้ไขปรับปรุงได้ โดยเพิ่มอุปกรณ์เข้าไปแต่สิ่งที่จะตามมาคือการออกแบบที่ซับซ้อนยุ่งยากมากยิ่งขึ้น



รูปที่ 2.19 แสดงวงจร Envelope Detector

2.8 การอ่านค่าแรงดันจากภาครับผ่านพอร์ต ADC

การสั่งงานมอเตอร์โดยไมโครคอนโทรลเลอร์ AVR เบอร์ ATmega128 ซึ่งมีโมดูลแปลงสัญญาณแอนาลอกเป็นดิจิตอล หรือ ADC (Analog to Digital Converter) ความละเอียดขนาด 10 บิต (10-bit Resolution) ที่แรงดัน +5V หมายถึงเมื่อแปลงสัญญาณดิจิตอลแล้วจะได้ค่าตัวเลขอยู่ระหว่าง 0-1024 โดยมีรูปแบบการแปลงสัญญาณแอนาลอกเป็นดิจิตอลแบบ Successive Approximation (ADC) คือการแปลงแบบประมาณค่า โดยการสุ่มค่าดิจิตอลแล้วแปลงเป็นแรงดันแอนาลอกภายในโมดูล เพื่อให้เปรียบเทียบกับแรงดันแอนาลอกด้านอินพุต เมื่อเปรียบเทียบได้ค่าแรงดันเท่ากัน โมดูล ADC จะให้ผลลัพธ์ออกมาเป็นค่าดิจิตอล ซึ่งการใช้วิธีการนี้เป็นที่นิยมเพราะมีความเที่ยงตรงสูงและทำงานได้อย่างรวดเร็ว

2.9 สรุป

เครื่องวัดแบบรูปการแผ่กระจาย พลังงานของสายอากาศที่ความถี่ 1GHz โดยวงจรภาครับมีองค์ประกอบที่สำคัญคือ วงจรควบคุมการหมุนของสายอากาศด้วยสเต็ปมอเตอร์ วงจร Envelope Detector วงจรขยายสัญญาณด้วย Op-amp และวงจรเรียงกระแส

บทที่ 3

การออกแบบและสร้างวงจรวัดแบบรูปการแผ่พลังงานที่ความถี่ 1GHz

เนื้อหาในบทนี้จะกล่าวถึงการออกแบบและการสร้าง วงจรวัดสัญญาณที่ความถี่ 1GHz โดยจะพิจารณาแยกทีละวงจรแล้วทำการทดสอบการใช้งานได้จริงของวงจรมานั้นๆก่อนที่จะนำมาประกอบกัน

3.1 เครื่องกำเนิดสัญญาณ (RF Generator)

เครื่องกำเนิดสัญญาณซึ่งทำการมอดูเลตแบบ AM (Amplitude Modulation) มีคลื่นพาหะ (Carrier Frequency) ความถี่ 1 GHz กับสัญญาณข่าวสาร (Message Frequency) ความถี่ 1 kHz ที่มีลักษณะสัญญาณเป็น Pulse มีกำลังส่งสูงสุดประมาณ 0dBm ดังรูปที่ 3.2



รูปที่ 3.1 เครื่องกำเนิดสัญญาณ (RF Generator)



รูปที่ 3.2 สัญญาณ AM จากเครื่องส่ง

3.2 สายอากาศภาคส่ง

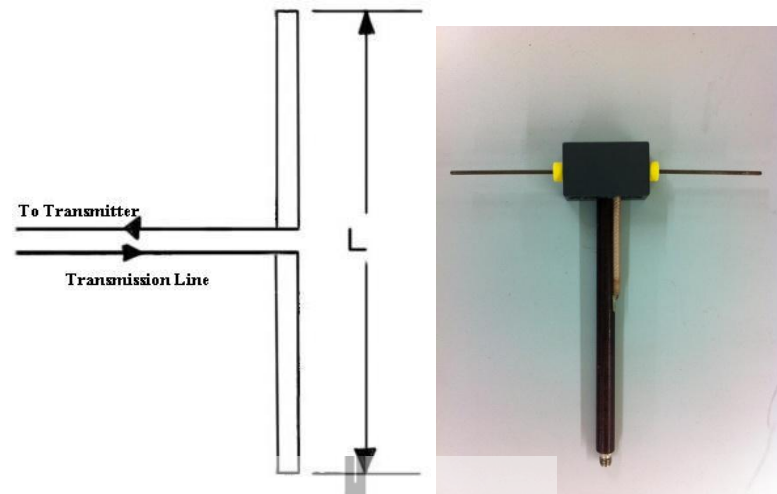
สายอากาศภาคส่งที่นำมาใช้ในการทดลอง เป็นสายอากาศ Yagi-Uda หรือสายอากาศ ก้างปลา มี 4 Element ซึ่งเป็นสายอากาศแบบทิศทางเดียวสามารถรับ-ส่งคลื่นได้ดีในทิศทางที่กำหนด และจะมีอัตราขยาย (gain) สูงกว่าประเภทอื่น อัตราขยาย (gain) เป็นความสามารถของสายอากาศ ในการรับส่งคลื่นวิทยุ สายอากาศแต่ละแบบมีอัตราขยายแตกต่างกัน สายอากาศแบบทิศทางเดียวจะมีอัตราขยายมากกว่าสายอากาศแบบกึ่งรอบตัว และแบบรอบตัวโดยลำดับ ลักษณะการใช้งาน จึงแตกต่างกันไป สายอากาศที่มีอัตราขยายสูงจะสามารถรับ-ส่งคลื่นวิทยุได้ดีมากมีหน่วยวัดอัตรา การขยายได้แก่ dB



รูปที่ 3.3 สายอากาศภาคส่ง
มหาวิทยาลัยเทคโนโลยีสุรนารี

3.3 สายอากาศภาครับ

สายอากาศภาครับที่นำมาใช้ในการทดลอง เป็น สายอากาศไดโพล (Dipole Antenna) ซึ่งเป็นสายอากาศอย่างง่าย ที่มีองค์ประกอบเป็นแท่งโลหะ 2 แท่งวางเป็นแนวเส้นตรงที่มีความยาว L ดังรูปที่ 3.4 โดยจุดกึ่งกลางของตัวไดโพลจะถูกต่อเข้ากับ ตัวป้อน โดยใช้สายส่งเป็นตัวกลางในการเชื่อมต่อ กระแส เชนผิวจะไหลไปยังขั้วหนึ่งของไดโพล และไหลกลับมายังอีกขั้วหนึ่งของ ไดโพลดังแสดงในรูปที่ 3.4 ซึ่งมีทิศทางตรงข้ามกับทิศทางของกระแสที่ส่งไปยังขั้วแรกของไดโพล



รูปที่ 3.4 สายอากาศภาครับ

3.4 วงจรควบคุมการหมุนของสายอากาศด้วยสแต็ปมอเตอร์

ส่วนของการควบคุมการหมุนของสายอากาศด้วยสแต็ปมอเตอร์ ในส่วนนี้จะเป็นตัวควบคุมการทำงานของระบบ โดยประกอบด้วยส่วนการทำงานต่างๆดังนี้

1. บอร์ดไมโครคอนโทรลเลอร์
2. บอร์ดขับเคลื่อนมอเตอร์



รูปที่ 3.5 ไมโครคอนโทรลเลอร์และชุดวงจรขับเคลื่อนมอเตอร์

รูปที่ 3.5 แสดงชุดวงจรควบคุมการหมุนของสแต็ปมอเตอร์ ซึ่งจะทำหน้าที่ควบคุมการหมุนของสแต็ปมอเตอร์เป็นแบบครึ่งสเต็ป โดยมีการเชื่อมต่อตามตารางที่ 3.1

ตารางที่ 3.1 แสดงการเชื่อมต่อพอร์ตของไมโครคอนโทรลเลอร์กับบอร์ดขับเคลื่อนมอเตอร์

| พอร์ตของไมโครคอนโทรลเลอร์ | พอร์ตของบอร์ดขับเคลื่อนมอเตอร์ |
|---------------------------|--|
| PA0 | INA เพื่อเชื่อมต่อไมโครคอนโทรลเลอร์กับ Out-A |
| PA1 | INB เพื่อเชื่อมต่อไมโครคอนโทรลเลอร์กับ Out-B |
| PA2 | INC เพื่อเชื่อมต่อไมโครคอนโทรลเลอร์กับ Out-C |
| PA3 | IND เพื่อเชื่อมต่อไมโครคอนโทรลเลอร์กับ Out-D |
| PA4 | EN1 เพื่อเปิดการใช้งานพอร์ต Out-A และ Out-B |
| PA5 | EN2 เพื่อเปิดการใช้งานพอร์ต Out-C และ Out-D |
| VCC | 5V |
| GND | GND |

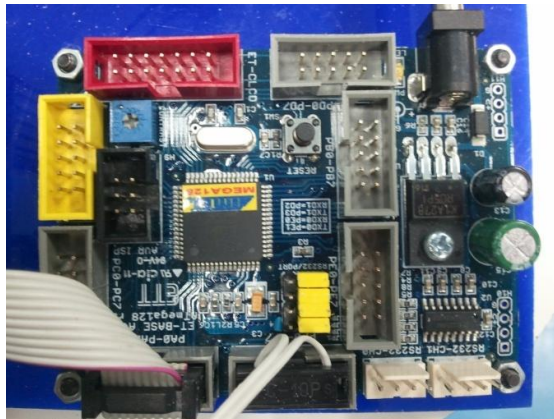


รูปที่ 3.6 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ ATMEGA 128 และบอร์ดขับเคลื่อนมอเตอร์

3.4.1 วงจรขับเคลื่อนมอเตอร์



(ก) วงจรขับเคลื่อนมอเตอร์



(ข) วงจรควบคุมวงจรขับเคลื่อนมอเตอร์

รูปที่ 3.7 ชุดควบคุมการหมุนของมอเตอร์

3.4.2 ชุดควบคุมการหมุนสายอากาศคาร์บและสเต็ปมอเตอร์

ในส่วนของสายอากาศคาร์บนี้ จะนำไปใช้ภายในห้องปฏิบัติการวิศวกรรมโทรคมนาคม ซึ่งจะใช้สายอากาศ ที่มีอยู่แล้วหมุนด้วยสเต็ปมอเตอร์ (0.9°/Step) ซึ่ง ถูกควบคุมการ หมุนจาก ไมโครคอนโทรลเลอร์ (AVR ATmega128) ที่บรรจุลงในกล่องเดียวกับวงจรขับเคลื่อนสเต็ปมอเตอร์ ดังรูปที่ 3.7 โดยใช้โปรแกรมภาษา C ในการเขียนเพื่อควบคุมการหมุนของสเต็ปมอเตอร์ให้หมุนทีละครั้งสเต็ปจะได้ครั้งละ 0.45° แล้วใช้โปรแกรม Visual Basic Studio 2010 ในการเขียนเพื่อรับค่า กำลังงานของสนามแม่เหล็กไฟฟ้า ที่สายอากาศรับได้ แล้วนำมาแสดงเป็นแบบรูปการแผ่พลังงานของสายอากาศ

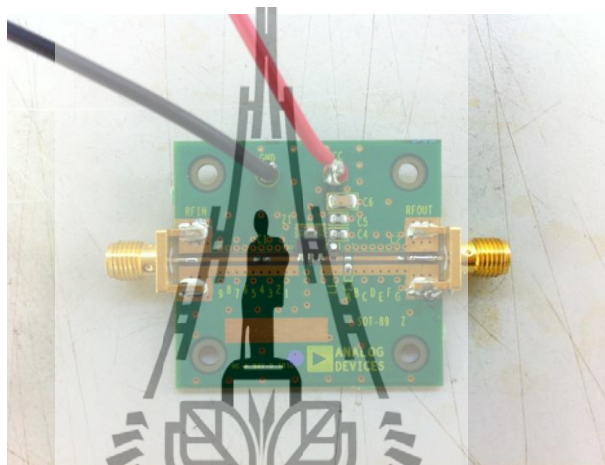


รูปที่ 3.8 ชุดควบคุมการหมุนของสายอากาศด้วยสเต็ปมอเตอร์

การทดสอบการหมุนของสเต็ปมอเตอร์ว่าหมุนที่ละครั้งสเต็ปจะได้ครั้งละ 0.45° จริงหรือไม่ นั้น ทำได้โดยการเขียน โปรแกรมภาษา C ให้สเต็ปมอเตอร์ หมุนไป 800 สเต็ปแล้วสเต็ปมอเตอร์หมุนครบหนึ่งรอบแล้วกลับมาที่ตำแหน่งเดิมแสดงว่าสเต็ปมอเตอร์ หมุนที่ละครั้งสเต็ปจะได้ 0.45° เนื่องจาก $0.45^\circ/\text{step} \times 800 \text{ step} = 360^\circ$

3.5 วงจรขยายสัญญาณ (Amplifier) ที่ความถี่ 1GHz

วงจขยายสัญญาณ (Amplifier) ที่ความถี่ 1GHz ต้องป้อนแรงดันไฟฟ้าไม่เกิน 5V และกำลังงานไม่เกิน 16dBm

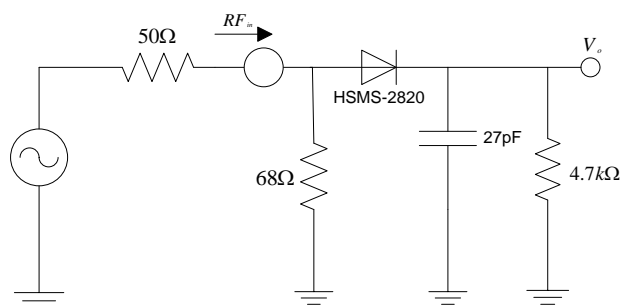


รูปที่ 3.9 วงจขยายสัญญาณ (Amplifier) ที่ความถี่ 1 GHz

เนื่องจากสัญญาณที่สายอากาศภาครับได้ มีค่าสัญญาณที่ต่ำมากจนวงจร Self-bias detector ไม่สามารถถอดสัญญาณขาออกจากสัญญาณพาหะได้ เราจึงนำวงจขยายสัญญาณที่ความถี่ 1GHz มาใช้เพื่อเพิ่มแอมพลิจูด (Amplitude) ทำให้สามารถถอดสัญญาณได้

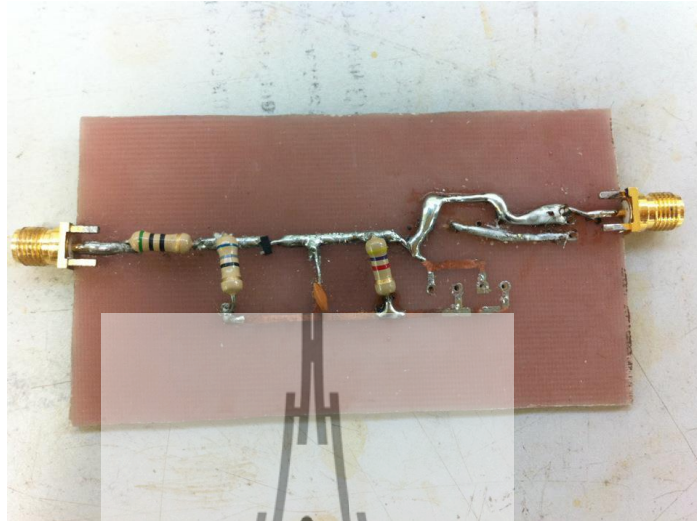
3.6 วงจร self-biased detector

ในการออกแบบและการสร้างวงจร Self-bias detector แสดงดังรูปที่ 3.6 (ก)



(ก) ลายวงจร self-biased detector

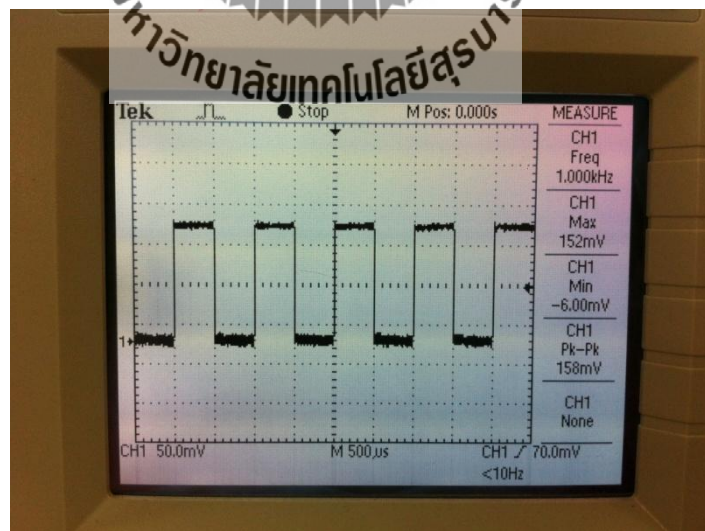
ลาย วงจรบนแผ่น วงจรพิมพ์ และ ทำการบัดกรีอุปกรณ์อิเล็กทรอนิกส์ต่างๆ ได้วงจร ดังรูปที่ 3.10 (ข)



(ข) วงจร self-biased detector

รูปที่ 3.10 วงจร self-biased detector

จากรูปที่ 3.11 เป็นวงจรที่เราได้ทำการออกแบบเพื่อ แยกสัญญาณข่าวสารกับสัญญาณ พหุระ รูปที่ 3.12 แสดงสัญญาณข่าวสารหลังจากผ่านวงจร Self-bias detector ซึ่งสัญญาณดังกล่าวจะมี ความถี่ 1 kHz

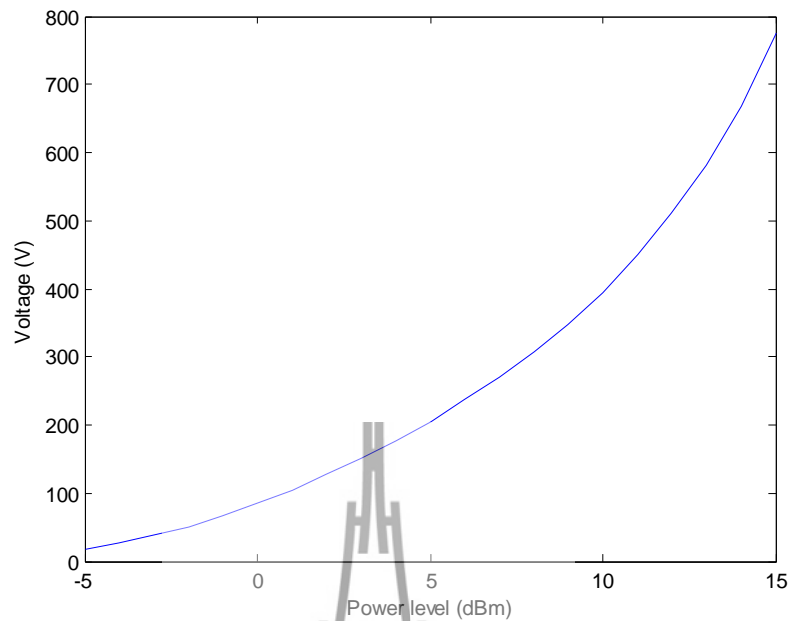


รูปที่ 3.11 สัญญาณข่าวสารที่ถูกตีมอดูเลตแล้ว

ตารางที่ 3.2 ผลการทดสอบระดับแรงดันที่ค่า Power ต่าง ๆ ของวงจรดีเทคเตอร์

| Power level (dBm) | ครั้งที่ 1 (mv) | ครั้งที่ 2 (mv) | ครั้งที่ 3 (mv) | ค่าเฉลี่ย (mv) |
|----------------------|--------------------|--------------------|--------------------|-------------------|
| -5 | 19.2 | 17.5 | 17.5 | 18.1 |
| -4 | 27.8 | 26.2 | 26.0 | 26.8 |
| -3 | 38.6 | 37.6 | 36.7 | 37.6 |
| -2 | 51.7 | 52.1 | 49.7 | 51.2 |
| -1 | 66.8 | 67.4 | 65.2 | 66.5 |
| 0 | 84.4 | 85 | 83.7 | 84.4 |
| 1 | 104.3 | 104.7 | 104.2 | 104.4 |
| 2 | 126.8 | 126.7 | 126.9 | 126.8 |
| 3 | 150.8 | 150.9 | 151.3 | 151.0 |
| 4 | 177.0 | 177.1 | 177.5 | 177.2 |
| 5 | 205.4 | 205.6 | 205.9 | 205.6 |
| 6 | 236.2 | 236.5 | 236.8 | 238.0 |
| 7 | 269.9 | 270.2 | 270.4 | 270.1 |
| 8 | 307.1 | 307.4 | 307.7 | 307.4 |
| 9 | 348.3 | 348.6 | 348.7 | 348.5 |
| 10 | 394.1 | 394.5 | 394.8 | 394.5 |
| 11 | 450.0 | 450.0 | 451.0 | 450.3 |
| 12 | 510.0 | 511.0 | 511.0 | 510.3 |
| 13 | 581.0 | 582.0 | 583.0 | 582.0 |
| 14 | 667.0 | 668.0 | 668.0 | 667.6 |
| 15 | 775.0 | 777.0 | 778.0 | 776.6 |

ตารางที่ 3.2 แสดงผลการทดสอบระดับแรงดันที่กำลังส่งค่าต่างๆ เมื่อสัญญาณผ่านวงจรดีเทคเตอร์ พบว่าระดับสัญญาณในการทดสอบแต่ละครั้งมีค่าแตกต่างกันเล็กน้อย เนื่องจากมีสัญญาณรบกวน ดังนั้นจึงทำการวัดสัญญาณทั้งหมด 3 ครั้ง และหาค่าเฉลี่ย รูปที่ 3.12 แสดงกราฟความสัมพันธ์ระหว่างค่ากำลังงานขาเข้าและระดับแรงดันขาออกของวงจรดีเทคเตอร์



รูปที่ 3.12 กราฟแสดงความสัมพันธ์ระหว่างพลังงานและแรงดัน

จากตารางที่ 3.2 และกราฟรูปที่ 3.12 แสดงให้เห็นว่าเมื่อเพิ่มกำลังส่งมากขึ้น ทำให้ระดับแรงดันที่เอาท์พุทของวงจรเพิ่มขึ้น และจากการทดสอบเมื่อมีสัญญาณข่าวสารความถี่ 1 kHz มอดูเลตกับสัญญาณพาหะความถี่ 1 GHz แบบ AM เข้ามาวงจรจะทำการถอดสัญญาณข่าวสารที่ความถี่ 1 kHz ออกมาได้

3.7 วงจรขยายสัญญาณด้วย Op-Amp ที่ความถี่ 1 kHz และวงจรเรียงกระแส (Rectifier)

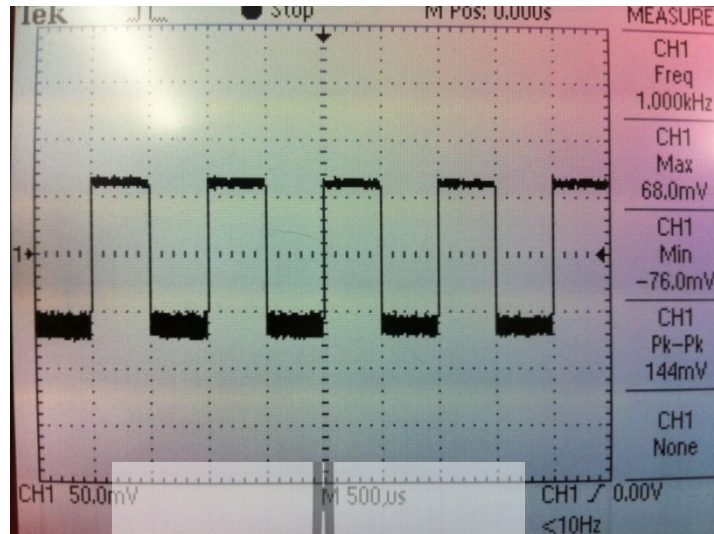
การออกแบบและสร้าง วงจรขยายสัญญาณด้วย Op-Amp ที่ความถี่ 1 kHz ซึ่งในโครงการนี้จะใช้แบบ Non-inverting สามารถคำนวณอัตราขยายได้จากสมการที่ 2.2

กำหนดให้ $R_f = 100k\Omega$

$$R_i = 10k\Omega$$

ดังนั้นเราจะได้อัตราขยาย
$$A_v = \frac{100k}{10k} + 1$$

11 เท่า



รูปที่3.13 สัญญาณอินพุตของวงจรรขยายสัญญาณด้วยออปแอมป์

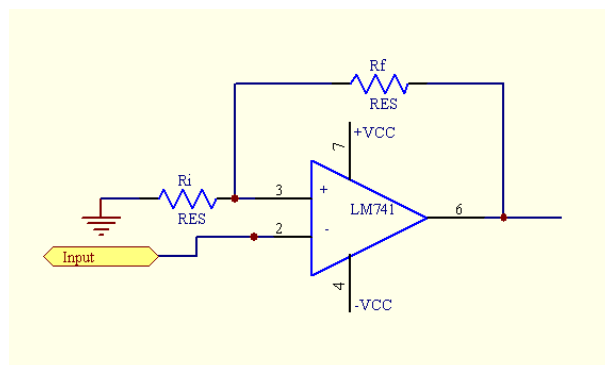
จากรูปที่ 3.11 และ 3.13พบว่าเกิดการสูญเสียในสายส่งประมาณ 14mV จึงทำการออกแบบวงจรรออปแอมป์มาขยายสัญญาณเมื่อทำการคำนวณพบว่าจะได้อัตราขยายเท่ากับ

$$A_v = \frac{V_o}{V_i}$$

$$V_o = 11 \times 144mV$$

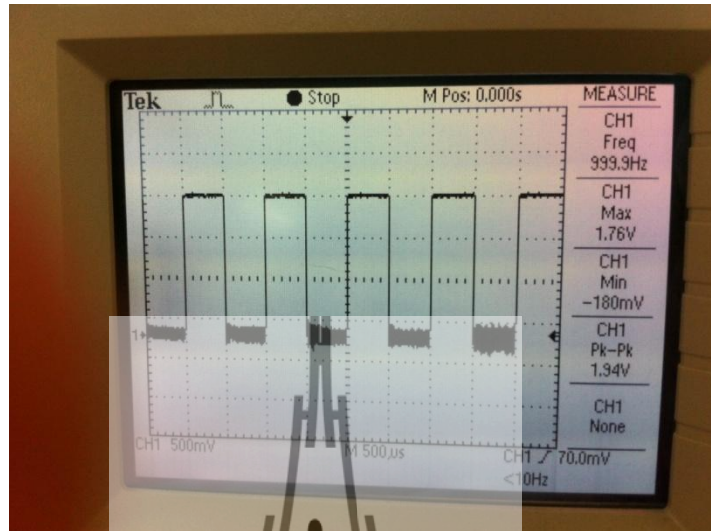
$$= 1.584V$$

เพราะฉะนั้น จะได้แรงดันเอาต์พุตเท่ากับ 1.584 V



รูปที่3.14 วงจรรขยายสัญญาณด้วยOp-AmpแบบNon-inverting

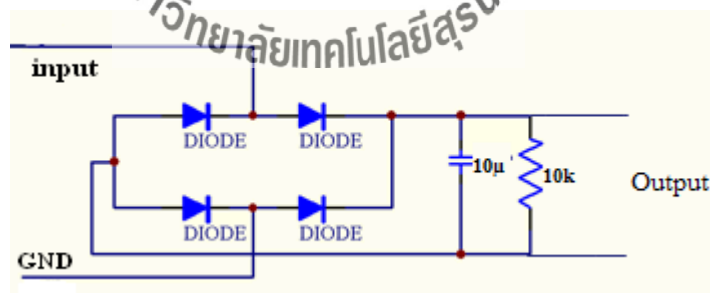
เมื่อสัญญาณผ่านวงจรขยายสัญญาณด้วยออปแอมป์จะได้สัญญาณเอาต์พุตดังรูปที่ 3.15 ทำให้ระดับสัญญาณเพิ่มขึ้นจาก 0.144 V เป็น 1.94 V



รูปที่ 3.15 สัญญาณเอาต์พุต

เนื่องจากสัญญาณหลังผ่านวงจรขยายสัญญาณด้วยออปแอมป์ มีลักษณะเป็นสัญญาณกระแสสลับ จึงต้องแปลงให้เป็นสัญญาณกระแสตรง เพื่อให้บอร์ดไมโครคอนโทรลเลอร์อ่านค่าระดับสัญญาณได้ ซึ่งการแปลงให้เป็นสัญญาณกระแสตรงดังกล่าว ต้องใช้วงจรเรียงกระแส

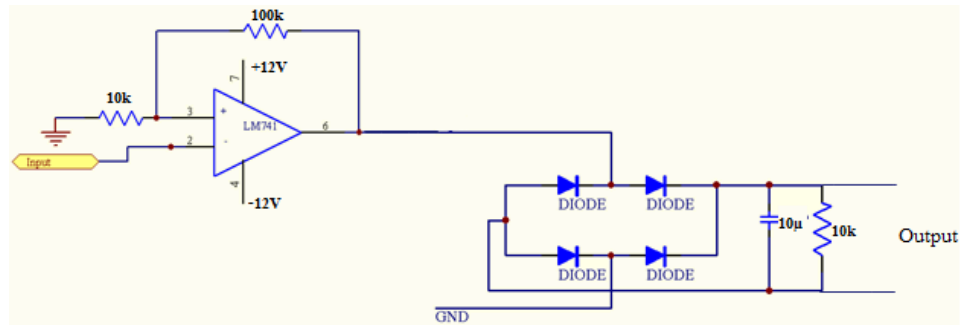
การออกแบบวงจรเรียงกระแส



รูปที่ 3.16 วงจรเรียงกระแส

เมื่อรวมวงจรขยายสัญญาณด้วยออปแอมป์และวงจรเรียงกระแส

แสดงดังรูปที่ 3.17

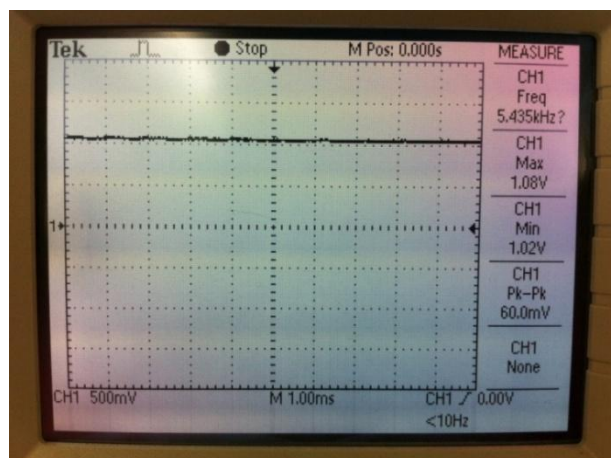


รูปที่3.17 ลายวงจรขยายสัญญาณด้วยออปแอมป์และวงจรเรียงกระแส

เมื่อได้ลายวงจรบนแผ่นPrint แล้วก็ทำการบัดกรีอุปกรณ์อิเล็กทรอนิกส์ต่างๆ จะได้วงจรดังรูปที่3.18 และสัญญาณที่ผ่านวงจรเรียงกระแสแสดงดังรูปที่ 3.19



รูปที่3.18 วงจรขยายสัญญาณด้วยออปแอมป์และวงจรเรียงกระแส

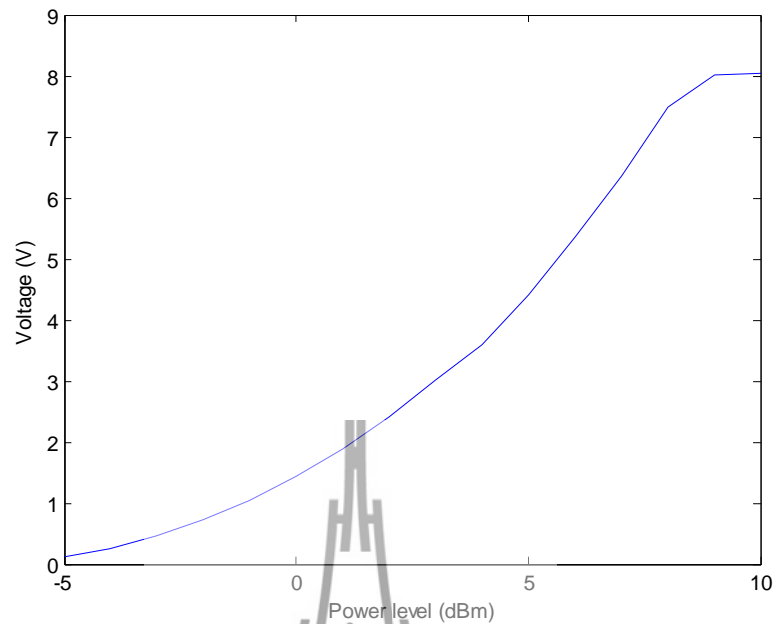


รูปที่3.19 สัญญาณเอาท์พุทเมื่อผ่าน วงจรขยายสัญญาณด้วย Op-Amp และวงจรเรียงกระแส

ตารางที่ 3.3 แสดงผลการทดสอบระดับแรงดันที่ค่า Power ต่าง ๆ ของวงดีเทคเตอร์ วงจรขยาย สัญญาณด้วยออปแอมป์ และวงจรเรียงกระแส

| Power level (dBm) | ครั้งที่ 1 (v) | ครั้งที่ 2 (v) | ครั้งที่ 3 (v) | ค่าเฉลี่ย (v) |
|----------------------|-------------------|-------------------|-------------------|------------------|
| -5 | 0.12 | 0.14 | 0.12 | 0.13 |
| -4 | 0.26 | 0.26 | 0.26 | 0.26 |
| -3 | 0.46 | 0.47 | 0.47 | 0.47 |
| -2 | 0.73 | 0.73 | 0.72 | 0.73 |
| -1 | 1.05 | 1.05 | 1.05 | 1.05 |
| 0 | 1.43 | 1.44 | 1.43 | 1.43 |
| 1 | 1.89 | 1.89 | 1.88 | 1.89 |
| 2 | 2.41 | 2.41 | 2.14 | 2.41 |
| 3 | 3.01 | 3.01 | 3.01 | 3.01 |
| 4 | 3.70 | 3.70 | 3.69 | 3.67 |
| 5 | 4.48 | 4.48 | 4.48 | 4.48 |
| 6 | 5.37 | 5.37 | 5.36 | 5.36 |
| 7 | 6.36 | 6.37 | 6.36 | 6.36 |
| 8 | 7.51 | 7.50 | 7.50 | 7.50 |
| 9 | 8.03 | 8.02 | 8.02 | 8.02 |
| 10 | 8.04 | 8.03 | 8.04 | 8.03 |

ตารางที่ 3.3 แสดงผลการทดสอบระดับแรงดันที่กำลังส่งค่าต่างๆ เมื่อสัญญาณผ่านวงจรถิเทคเตอร์ วงจรขยายสัญญาณด้วยออปแอมป์ และวงจรเรียงกระแสพบว่าระดับสัญญาณในการทดสอบแต่ละครั้งมีค่าใกล้เคียงกันและ รูปที่ 3.20 แสดงกราฟความสัมพันธ์ระหว่างค่ากำลังงานขาเข้า และ ระดับแรงดันขาออกของวงจร



รูปที่ 3.20 แสดงความสัมพันธ์ระหว่างกำลังงานส่งและแรงดันหลังผ่านวงจรเรียงกระแส

จากตารางที่ 3.3 และกราฟรูปที่ 3.20 แสดงให้เห็นว่าเมื่อเพิ่มกำลังส่งมากขึ้นระดับแรงดันที่เอาต์พุตของวงจรจะเพิ่มขึ้นด้วยและจากการทดสอบเมื่อมีสัญญาณข่าวสารความถี่ 1 kHz มอดูเลตกับสัญญาณพาหะความถี่ 1 GHz แบบ AM เข้ามาวงจรจะทำการถอดสัญญาณข่าวสารที่ความถี่ 1 kHz ออกมาได้ แล้วผ่านวงจรขยายสัญญาณด้วยออปแอมป์และวงจรเรียงกระแส ทำให้ได้สัญญาณกระแสตรง ที่มีค่าแรงดันเพิ่มขึ้นตามกำลังงานของเครื่องส่งซึ่งแรงดันอยู่ในช่วงระหว่าง 0 – 8 V (ซึ่งไมโครคอนโทรลเลอร์สามารถอ่านค่าได้ในช่วง 0 – 5 V)

3.8 การอ่านค่าแรงดันจากภาครับ

การอ่านค่าแรงดันจากภาครับ สามารถทำได้จากการใช้บอร์ดไมโครคอนโทรลเลอร์อ่านค่าแรงดันจากภาครับ ซึ่งสามารถอ่านค่าผ่านพอร์ต ADC ของบอร์ดไมโครคอนโทรลเลอร์ ซึ่งทำการต่อพอร์ตดังตารางนี้

ตารางที่ 3.4 แสดงการเชื่อมต่อพอร์ต ADC กับ เอาต์พุตของวงจรภาครับ

| พอร์ต ADC ของบอร์ดไมโครคอนโทรลเลอร์ | เอาต์พุตของวงจรภาครับ |
|-------------------------------------|-----------------------|
| PF0 | Output |
| GND | GND |

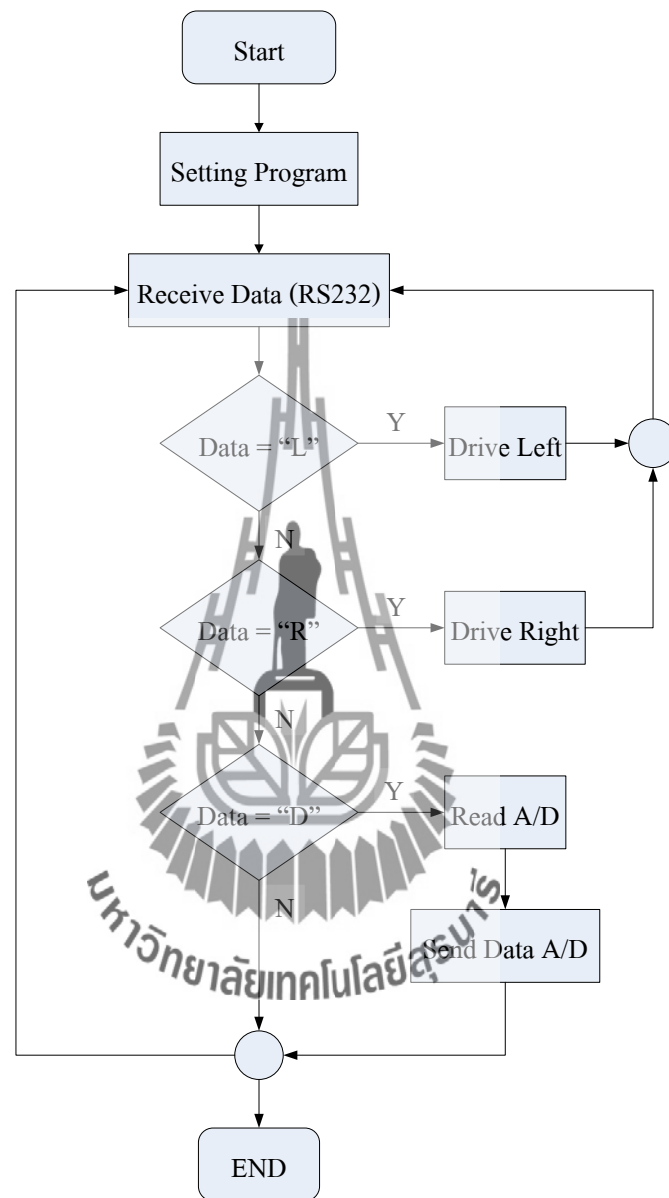
เมื่อทำการทดสอบวงจรย่อยแต่ละวงจรที่เป็นองค์ประกอบของวงจรรับสัญญาณ ที่ความถี่ 1GHz แล้ว จากนั้นนำวงจรย่อยต่างๆมาต่อกัน และได้ทำการทดสอบวงจรโดยรวมอีกครั้ง พบว่าวงจรย่อยแต่ละวงจรสามารถนำมาใช้งานร่วมกันได้ เนื่องจากเอาท์พุทที่ได้ในแต่ละวงจรมันให้ผลเป็นไปตามที่ต้องการ



รูปที่ 3.21 วงจรย่อยต่างๆ ประกอบกันเป็นชุดอุปกรณ์ภาครับสัญญาณ

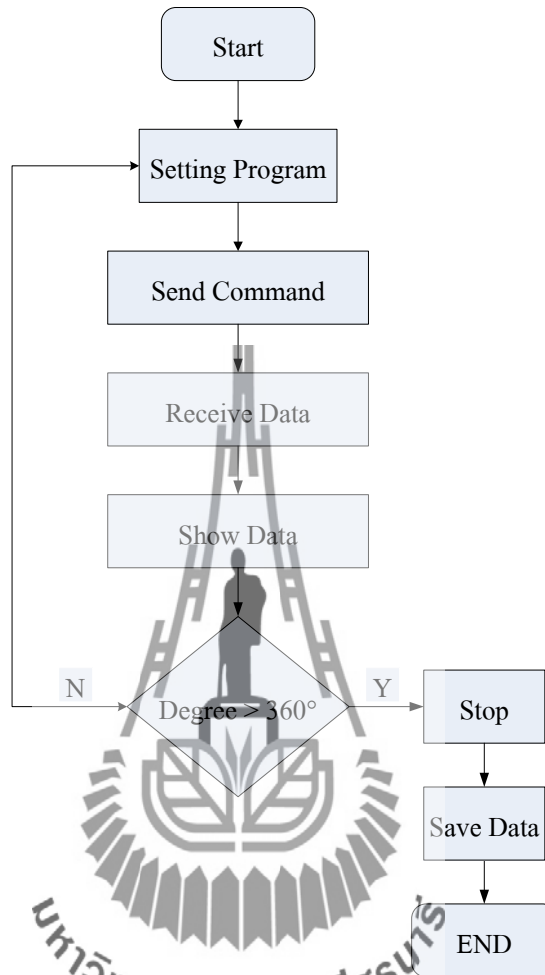
3.9 การออกแบบและสร้างโปรแกรมวัดแบบรูปการแผ่พลังงานที่ความถี่ 1GHz

Flow Chart ควบคุมการหมุนจากไมโครคอนโทรลเลอร์ (AVR ATmega128) ดังรูป 3.22



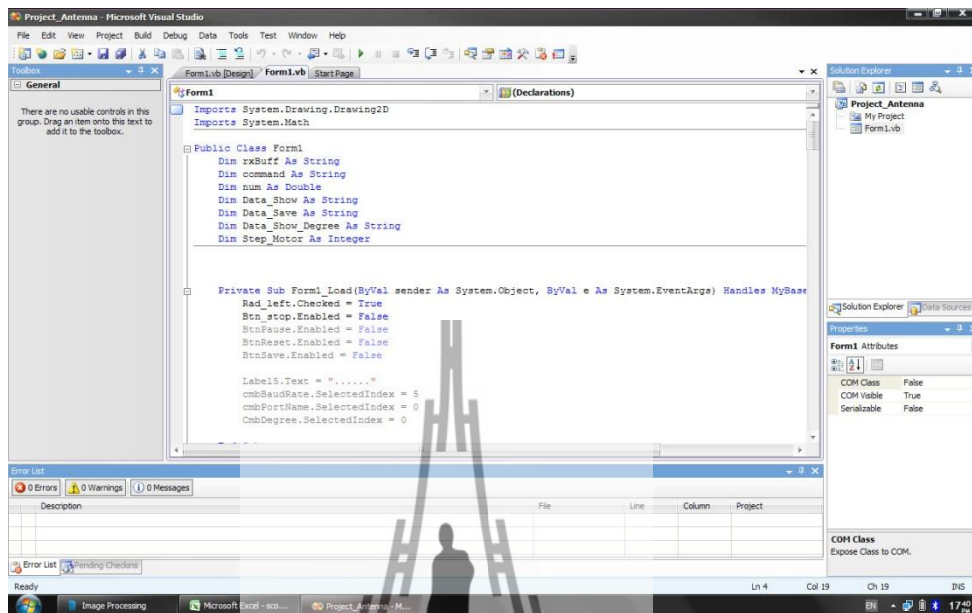
รูปที่ 3.22 Flow Chart ควบคุมการหมุนจากไมโครคอนโทรลเลอร์ (AVR ATmega128)

Flow Chart การทำงานของโปรแกรม Visual Basic Studio 2010 แสดงดังรูปที่ 3.23

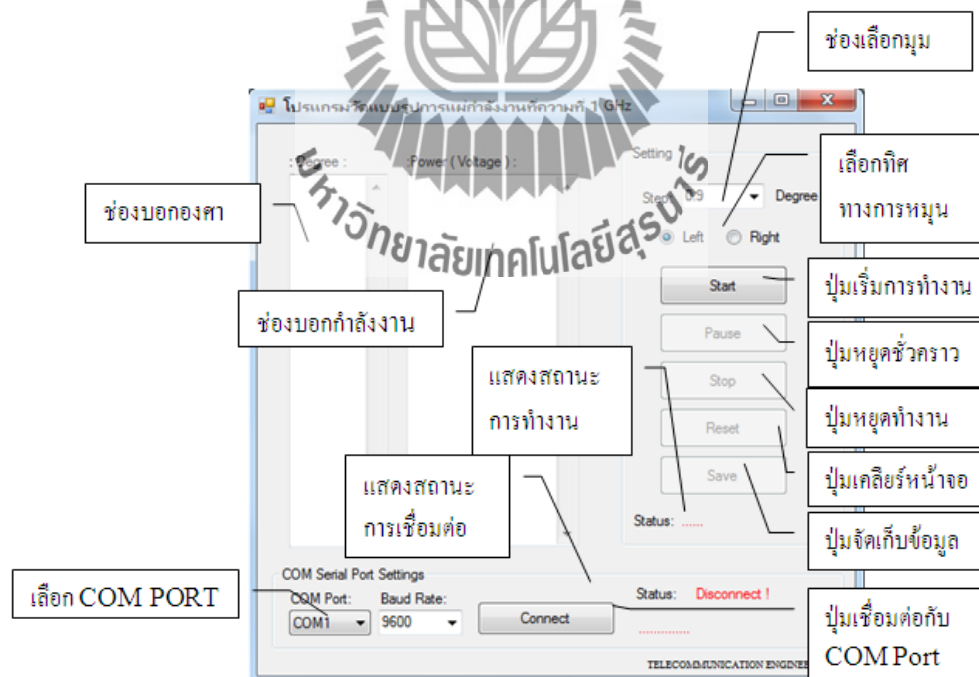


รูปที่ 3.23 Flow Chart การทำงานของโปรแกรม Visual Basic Studio 2010

เมื่อทำการเปิดโปรแกรมการออกแบบและสร้างโปรแกรมวัดแบบรูปการแผ่พลังงานที่ความถี่ 1GHz แสดงดังรูปที่ 3.24 ซึ่งได้การออกแบบจะแสดงอยู่ในภาคผนวก ก



รูปที่ 3.24 การออกแบบโปรแกรมวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz



รูปที่ 3.25 ส่วนแสดงผลเครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz

จากรูปที่ 3.25 ส่วนแสดงผลเครื่องวัดแบบรูปการแผ่พลังงานที่ความถี่ 1 GHz ซึ่งประกอบด้วยส่วนประกอบต่าง ๆ ดังนี้

1. ส่วนแสดงองศา

2. ส่วนแสดงค่า แรงดันมีหน่วยเป็น โวลต์(V)

3. ส่วนของ Com Serial Port Settings ประกอบด้วย

- ช่องเลือก Com Port
- ช่องเลือก Baud Rate
- ปุ่ม Connect
- ส่วนแสดงสถานะการเชื่อมต่อ

4. ส่วนของการตั้งค่า (Setting) ประกอบด้วย

- ช่องเลือกองศาในการทำงานของมอเตอร์
- ทิศทางการหมุนของมอเตอร์
- ปุ่มเริ่มการทำงาน
- ปุ่มหยุดชั่วคราว
- ปุ่มหยุดการทำงาน
- ปุ่มเคลียร์หน้าจอ
- ปุ่มบันทึกข้อมูล
- ส่วนแสดงสถานะการทำงาน

บทที่ 4

การทดสอบ

ในบทนี้จะเป็นการนำวงจรทั้งหมดที่ได้ ทำการ ออกแบบและสร้างขึ้น มาประกอบเข้าเป็น เครื่องรับสัญญาณที่ความถี่ 1 GHz แล้วเชื่อมต่อกับเครื่องคอมพิวเตอร์ด้วย Serial Port เพื่อทำการ วัดแบบรูปการแผ่พลังงาน ของสายอากาศ โดยสายอากาศภาครับจะหมุน ตั้งแต่มุม 0° - 360° การ เชื่อมต่อวงจรและอุปกรณ์ต่างๆ แสดงดังรูปที่ 4.1

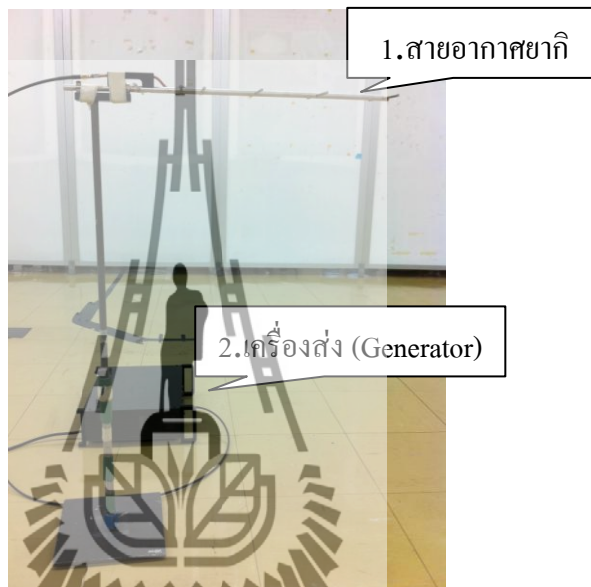


รูปที่ 4.1 แสดงระบบการทดสอบทั้งหมด

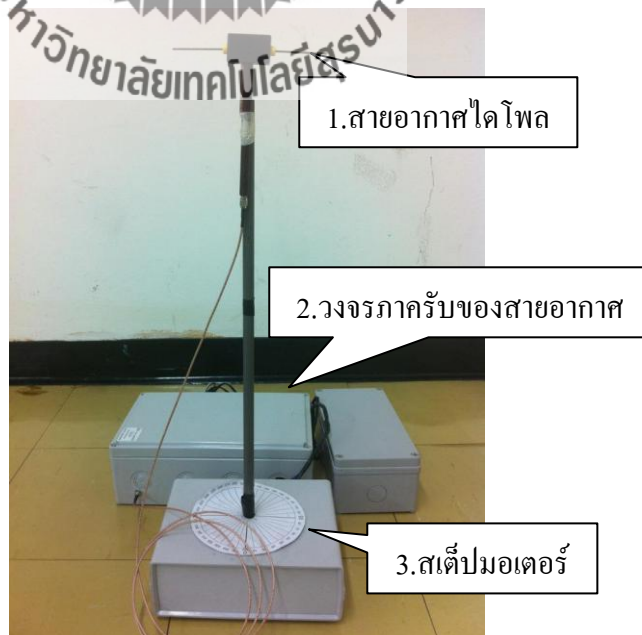
4.1 การติดตั้งอุปกรณ์และโปรแกรม

1. การติดตั้งอุปกรณ์

การทดสอบระบบรวมทั้งหมดจะให้สายอากาศในภาครับหมุนตั้งแต่ 0° ถึง 360° เพื่อวัดพลังงานของสายอากาศทั้งในระนาบสนามไฟฟ้า (E-plane) และระนาบสนามแม่เหล็ก (H-plane) ที่ระยะห่างระหว่างสายอากาศภาครับและภาคส่งเท่ากับ 1 เมตรแล้วนำค่าที่ได้ไปประมวลผลต่อในคอมพิวเตอร์เพื่อวัดแบบรูปการแผ่พลังงานของสายอากาศ รูปที่ 4.2 และ 4.3 แสดงอุปกรณ์ภาคส่งและอุปกรณ์ภาครับ ตามลำดับ



รูปที่ 4.2 แสดงอุปกรณ์ภาคส่ง



รูปที่ 4.3 แสดงอุปกรณ์ภาครับ



รูปที่ 4.4 แสดงการเลือกใช้ความถี่บนเครื่องส่ง (Generator)

ขั้นตอนการต่ออุปกรณ์ภาคส่ง จากรูปที่ 4.4 เครื่องส่ง (Generator) เลือกใช้ที่ความถี่ 1 GHz

- ต่อสายโคแอกเซียลที่เอาต์พุทของเครื่องส่งกับสายอากาศ

- เลือก MODE 

- เลือก RF POWER 


ขั้นตอนการต่ออุปกรณ์ภาครับ

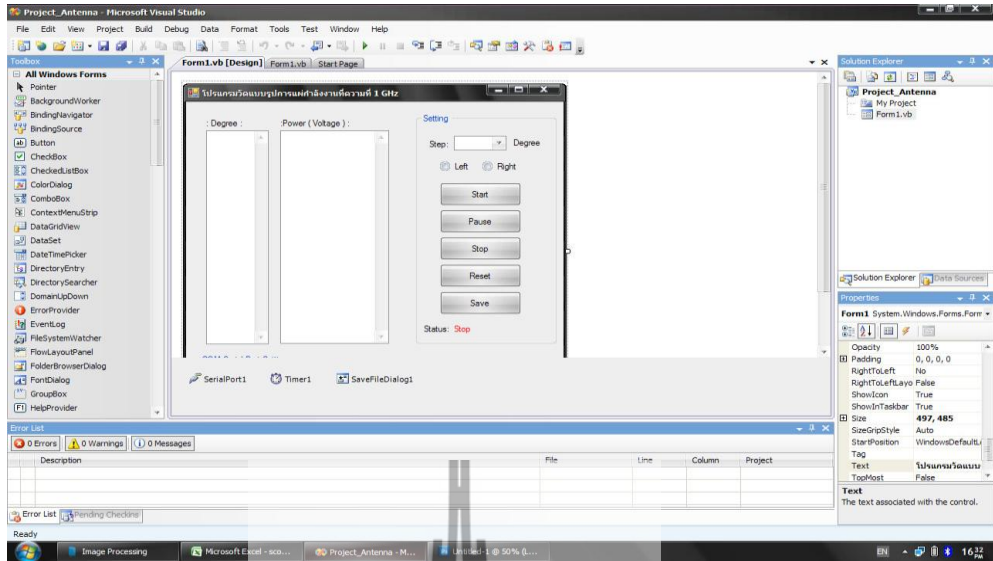
- ต่อสายโคแอกเซียลจากสายอากาศได้ โพลกับชุดอุปกรณ์ภาครับจากนั้นกดเปิดเครื่องส่ง



รูปที่ 4.5 แสดงการพร้อมใช้งานของอุปกรณ์ทั้งภาคส่งและภาครับ

2. เปิดโปรแกรม Visual Basic Studio 2010

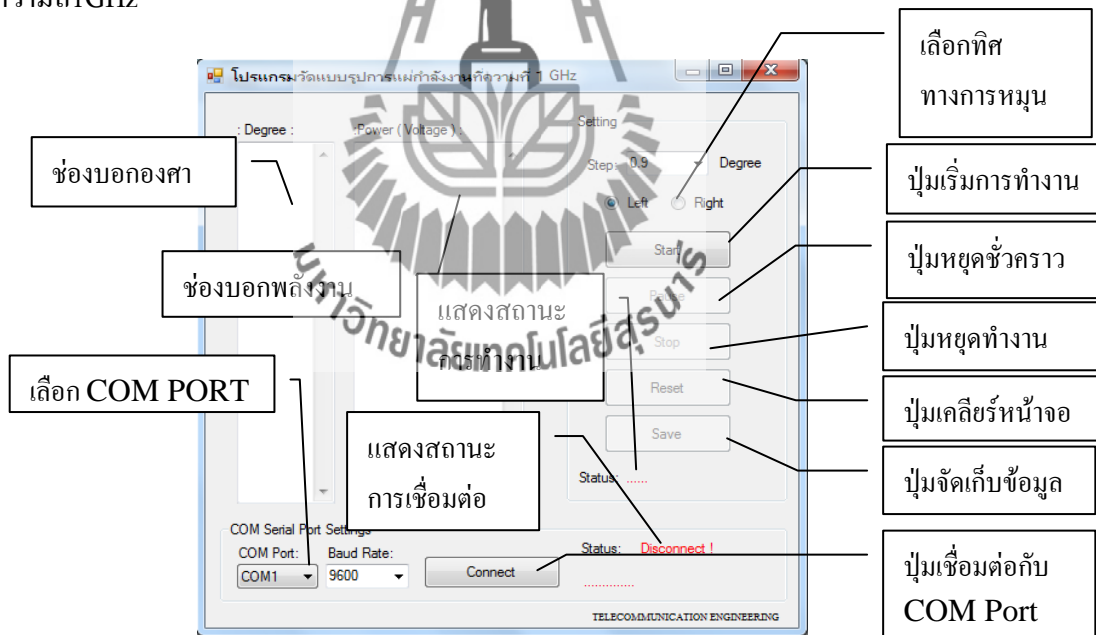
เปิดโปรแกรม Visual Basic Studio 2010 โดยคลิกที่  จะได้นหน้าต่างดังรูปที่ 4.6 เพื่อใช้ในการเชื่อมต่อกับไมโครคอนโทรลเลอร์ เพื่อทำการรับค่าพลังงานของสายอากาศ



รูปที่ 4.6 เปิดโปรแกรม Visual Basic Studio 2010

4.2 ทำการรันโปรแกรม

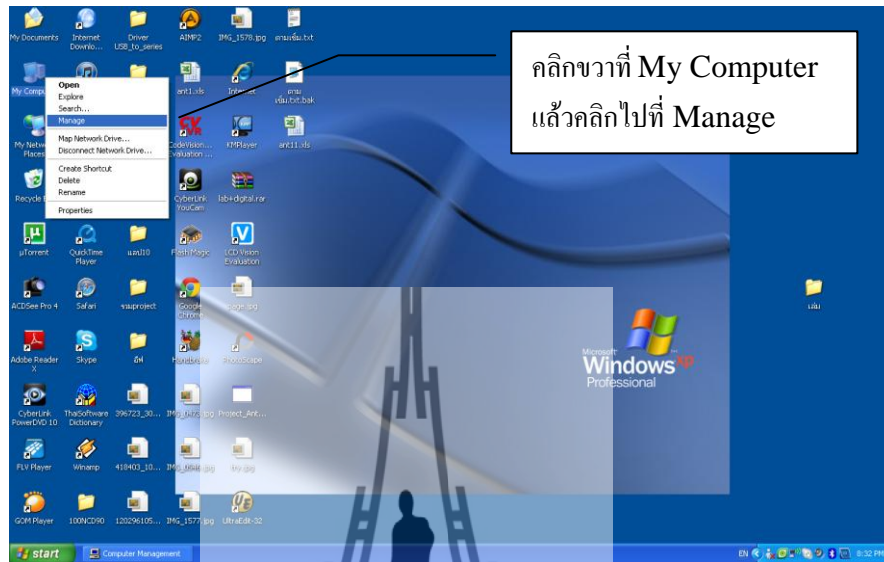
เมื่อทำการเปิดโปรแกรมการออกแบบและสร้างโปรแกรมวัดรูปแบบการแผ่พลังงานที่ความถี่ 1GHz



รูปที่ 4.7 โปรแกรมวัดรูปแบบการแผ่พลังงานที่ความถี่ 1 GHz

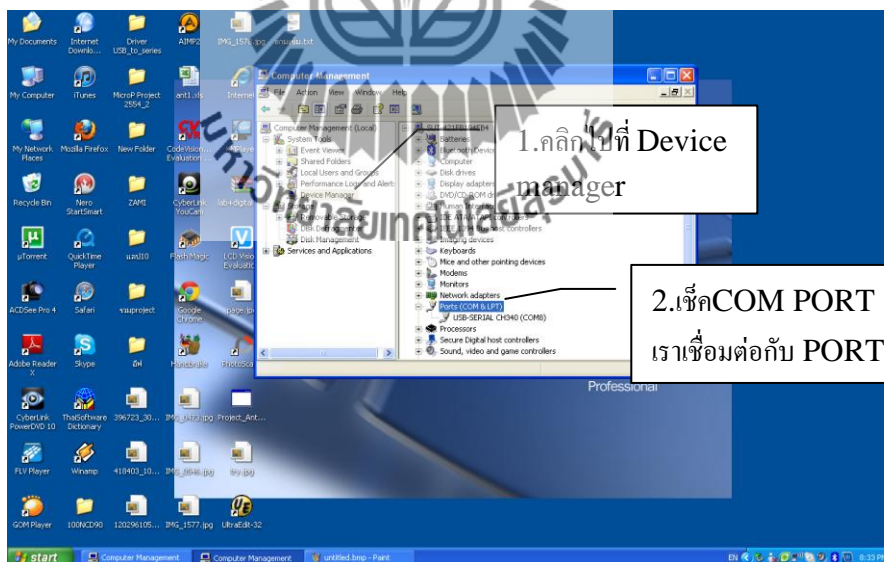
4.3 ขั้นตอนการใช้งาน

1. ให้เลือก COM PORT ที่จะทำการเชื่อมต่อกับ PORT RS232 และจำนวน Buad rate ดังรูป 4.8



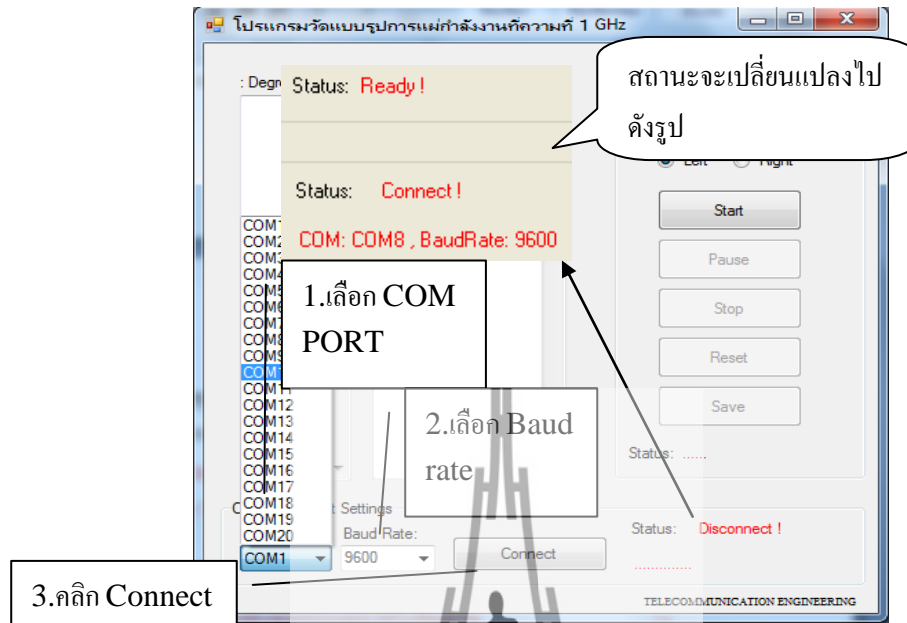
รูปที่ 4.8 การเปิดโปรแกรม My Computer

จะแสดงหน้าต่าง Computer Management



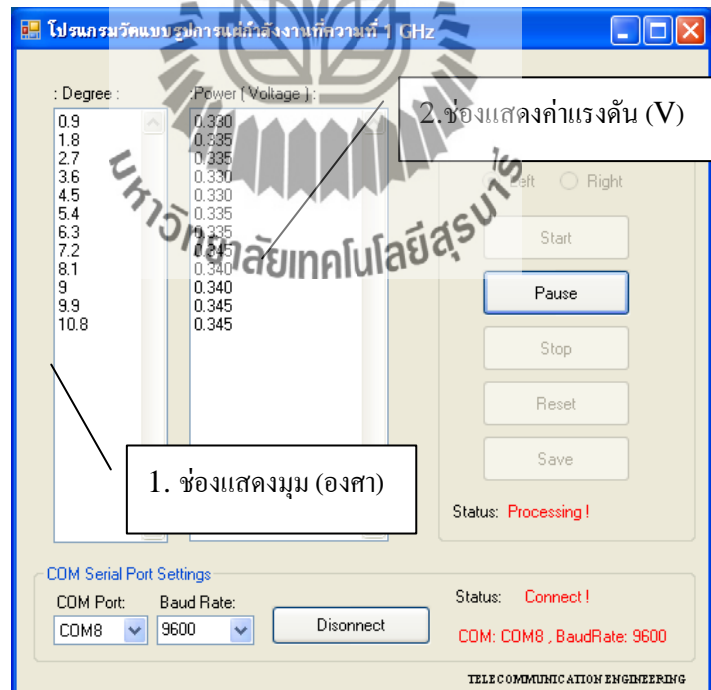
รูปที่ 4.9 แสดงการเชื่อมต่อของ COM PORT กับ COMPUTER

2. ไปที่โปรแกรมที่เราออกแบบให้ปฏิบัติดังนี้



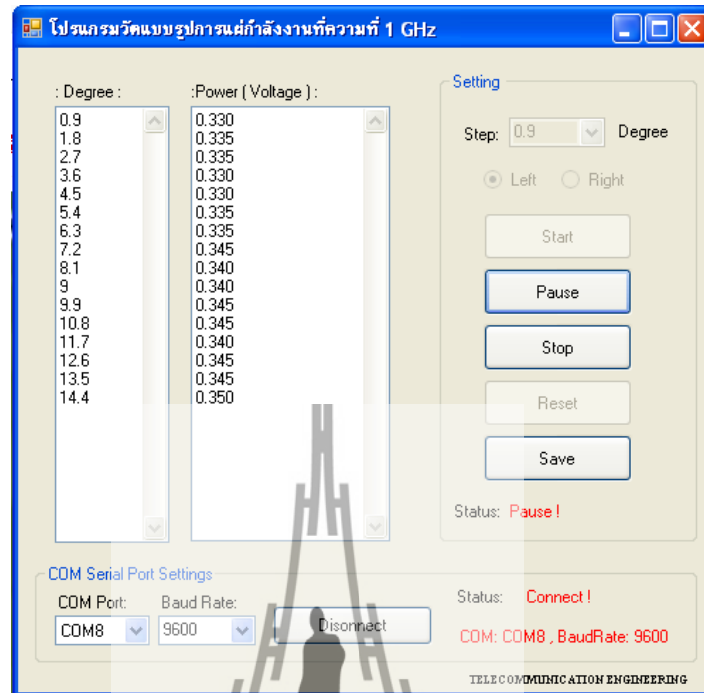
รูปที่ 4.10 แสดงการเชื่อมต่อโปรแกรมพร้อมใช้งาน

3. ขั้นตอนต่อไปคลิก Start แล้วโปรแกรมจะทำงานดังรูปที่ 4.12



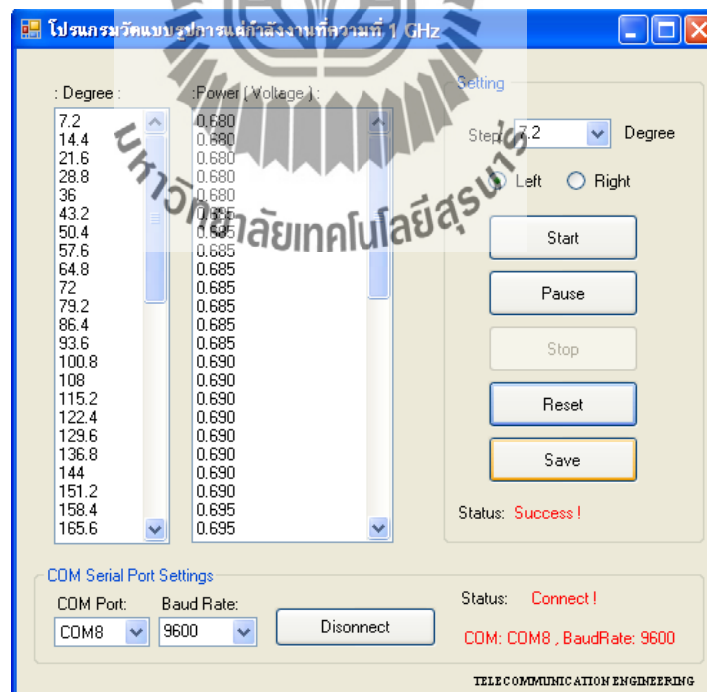
รูปที่ 4.11 แสดงการเริ่มใช้งานโปรแกรม

4. การหยุดโปรแกรมชั่วคราวโดยการคลิก Pause



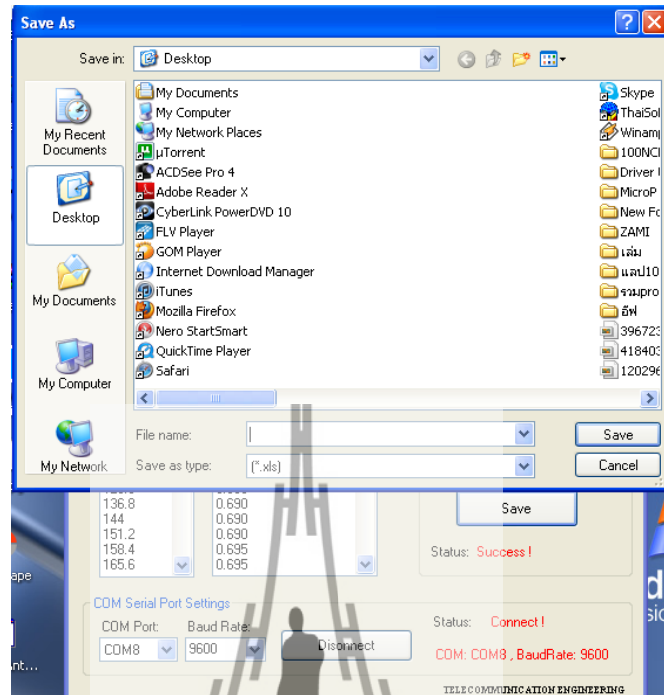
รูปที่ 4.12 แสดงการหยุดโปรแกรมชั่วคราว

5. เมื่อโปรแกรมทำงานเสร็จ สถานะจะขึ้นว่า Success!



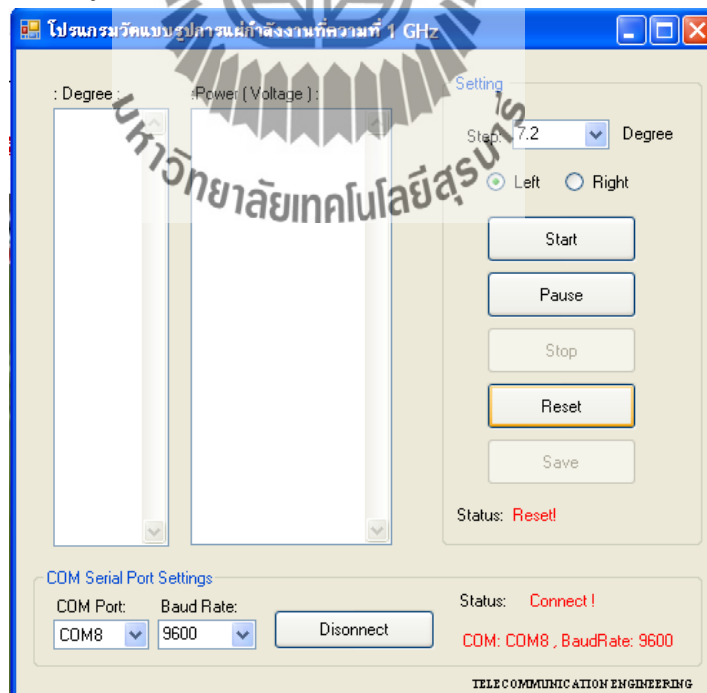
รูปที่ 4.13 เสร็จสิ้นการทำงาน

6. การบันทึกข้อมูล ให้คลิกปุ่ม Save แล้วพิมพ์ชื่อไฟล์นามสกุล .xls



รูปที่ 4.14 แสดงการบันทึกข้อมูล

7. การเคลียร์หน้าต่างข้อมูล ให้คลิก Reset จะแสดงสถานะว่า Reset !



รูปที่ 4.15 แสดงการเคลียร์ค่าข้อมูล

4.4 ผลการวัดแบบรูปการแผ่พลังงานของสายอากาศ

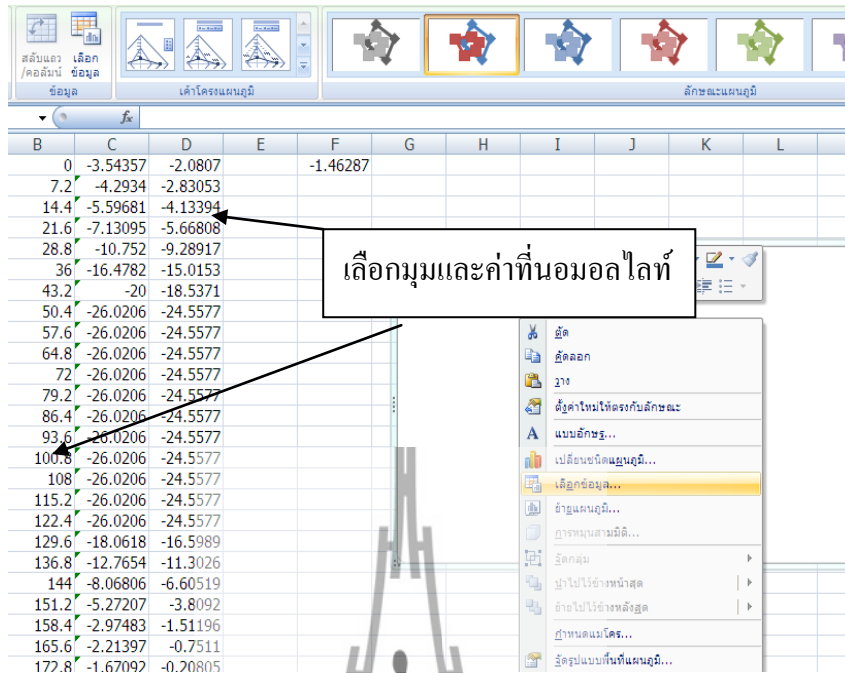
1. นำผลการวัดพลังงานที่บันทึกไว้ใน Microsoft Office Excel ซึ่งจะแสดงค่าแรงดันในหน่วยโวลต์
2. บอกมุมมองตามการทำงานของมอเตอร์
3. นำค่าพลังงานมาทำการคำนวณโดยใช้สมการ $20 \log (V)$ เพื่อเปลี่ยนหน่วยให้เป็น dB
4. นำค่าพลังงานสูงสุดที่ได้จากหน่วย dB มาทำการนอมอลไลซ์ และบอกมุมมองตามการทำงานของมอเตอร์นำค่ามาพล็อตกราฟเพื่อแสดงผลแบบ Polar Plot ที่ Normalize

| | A | B | C |
|----|-------|-------|---|
| 1 | 0.665 | 0 | |
| 2 | 0.61 | 7.2 | |
| 3 | 0.525 | 14.4 | |
| 4 | 0.44 | 21.6 | |
| 5 | 0.29 | 28.8 | |
| 6 | 0.15 | 36 | |
| 7 | 0.1 | 43.2 | |
| 8 | 0.05 | 50.4 | |
| 9 | 0.05 | 57.6 | |
| 10 | 0.05 | 64.8 | |
| 11 | 0.05 | 72 | |
| 12 | 0.05 | 79.2 | |
| 13 | 0.05 | 86.4 | |
| 14 | 0.05 | 93.6 | |
| 15 | 0.05 | 100.8 | |
| 16 | 0.05 | 108 | |
| 17 | 0.05 | | |
| 18 | 0.05 | | |
| 19 | 0.125 | | |
| 20 | 0.23 | | |
| 21 | 0.395 | | |

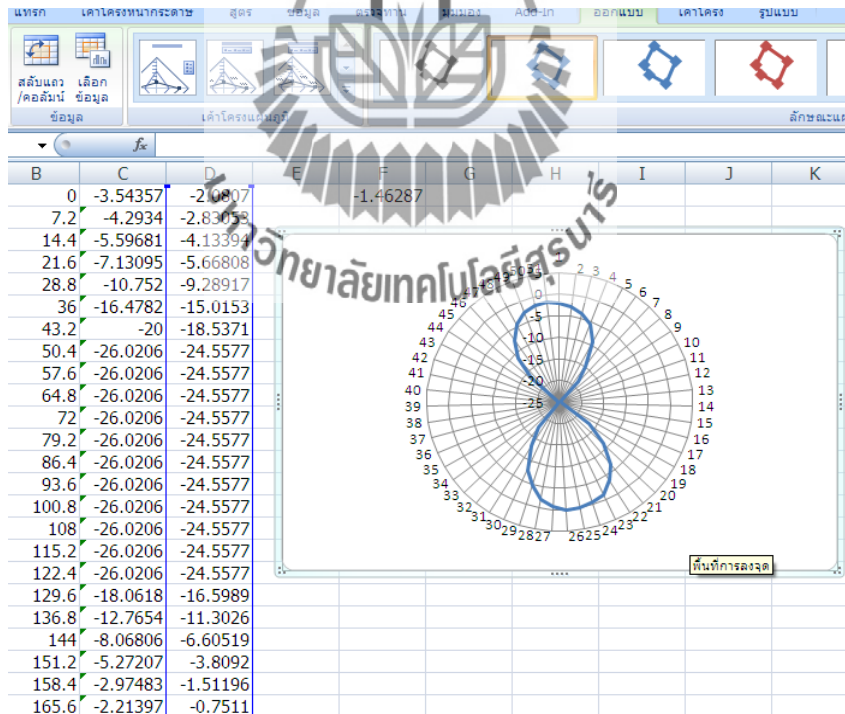
รูปที่ 4.16 แสดงค่าพลังงานและมุม (องศา)

| | A | B | C | D | E | F |
|----|-------|-------|----------|---|---|----------|
| 1 | 0.665 | 0 | -3.54357 | | | -1.46287 |
| 2 | 0.61 | 7.2 | -4.2934 | | | |
| 3 | 0.525 | 14.4 | -5.59681 | | | |
| 4 | 0.44 | 21.6 | -7.13095 | | | |
| 5 | 0.29 | 28.8 | -10.752 | | | |
| 6 | 0.15 | 36 | -16.4782 | | | |
| 7 | 0.1 | 43.2 | -20 | | | |
| 8 | 0.05 | 50.4 | -26.0206 | | | |
| 9 | 0.05 | 57.6 | -26.0206 | | | |
| 10 | 0.05 | 64.8 | -26.0206 | | | |
| 11 | 0.05 | 72 | -26.0206 | | | |
| 12 | 0.05 | 79.2 | -26.0206 | | | |
| 13 | 0.05 | 86.4 | -26.0206 | | | |
| 14 | 0.05 | 93.6 | -26.0206 | | | |
| 15 | 0.05 | 100.8 | -26.0206 | | | |
| 16 | 0.05 | 108 | -26.0206 | | | |
| 17 | 0.05 | 115.2 | -26.0206 | | | |
| 18 | 0.05 | 122.4 | -26.0206 | | | |
| 19 | 0.125 | 129.6 | -18.0618 | | | |
| 20 | 0.23 | 136.8 | -12.7654 | | | |
| 21 | 0.395 | 144 | -8.06806 | | | |
| 22 | 0.545 | 151.2 | -5.27207 | | | |
| 23 | 0.71 | 158.4 | -2.97483 | | | |
| 24 | 0.775 | 165.6 | -2.21207 | | | |

รูปที่ 4.17 แสดงค่าพลังงานในหน่วย dB และหาพลังงานสูงสุด



รูปที่ 4.18 แสดงการเลือกข้อมูลที่ใช้ในการพล็อตกราฟ



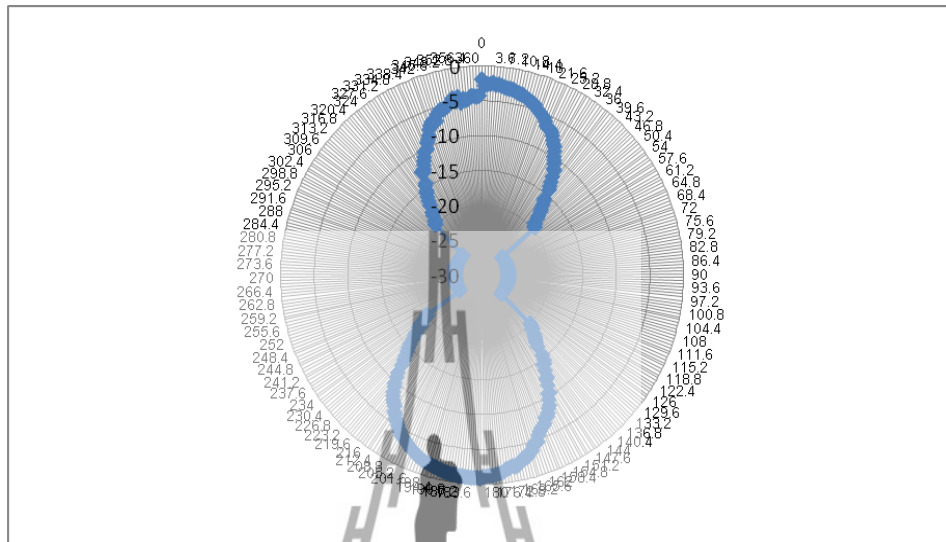
รูปที่ 4.19 แสดงแบบรูปการแผ่พลังงานที่ได้จากการพล็อตกราฟ

ตัวอย่างตารางแสดงผลทดสอบการวัดแบบรูปการแผ่พลังงานของสายอากาศ เมื่อสลับปี
มอเตอร์หมุนทีละ 1.8°

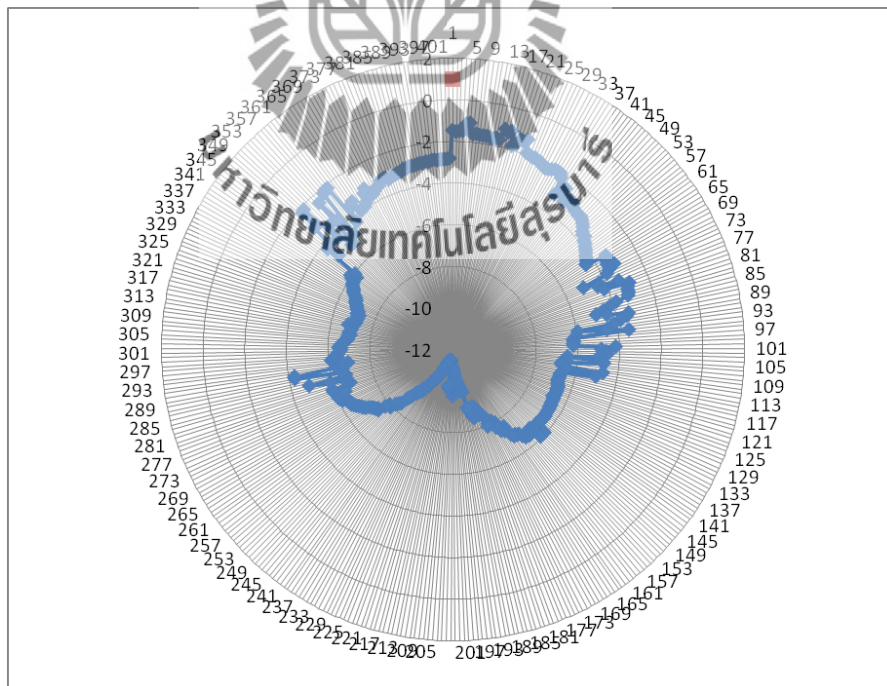
ตารางที่ 4.1 แสดงผลการทดสอบแรงดันของสายอากาศ

| มุม (องศา) | ระนาบ สนามไฟฟ้า (V) | ระนาบ สนามแม่เหล็ก (V) | มุม (องศา) | ระนาบ สนามไฟฟ้า (V) | ระนาบ สนามแม่เหล็ก (V) |
|---------------|---------------------------|------------------------------|---------------|---------------------------|------------------------------|
| 0 | 0.665 | 1.367 | 187.2 | 0.77 | 0.54 |
| 7.2 | 0.61 | 1.362 | 194.4 | 0.66 | 0.58 |
| 14.4 | 0.525 | 1.352 | 201.6 | 0.495 | 0.655 |
| 21.6 | 0.44 | 1.307 | 208.8 | 0.35 | 0.72 |
| 28.8 | 0.29 | 1.242 | 216 | 0.19 | 0.745 |
| 36 | 0.15 | 1.162 | 223.2 | 0.11 | 0.77 |
| 43.2 | 0.1 | 1.67 | 230.4 | 0.05 | 0.785 |
| 50.4 | 0.05 | 1.77 | 237.6 | 0.05 | 0.8 |
| 57.6 | 0.05 | 0.91 | 244.8 | 0.05 | 0.79 |
| 64.8 | 0.05 | 0.855 | 252 | 0.05 | 0.77 |
| 72 | 0.05 | 0.83 | 259.2 | 0.05 | 0.785 |
| 79.2 | 0.05 | 0.835 | 266.4 | 0.05 | 0.835 |
| 86.4 | 0.05 | 0.87 | 273.6 | 0.05 | 0.915 |
| 93.6 | 0.05 | 0.905 | 280.8 | 0.05 | 0.94 |
| 100.8 | 0.05 | 0.93 | 288 | 0.05 | 1.42 |
| 108 | 0.05 | 0.91 | 295.2 | 0.05 | 1.112 |
| 115.2 | 0.05 | 0.845 | 302.4 | 0.05 | 1.212 |
| 122.4 | 0.05 | 0.78 | 309.6 | 0.05 | 1.297 |
| 129.6 | 0.125 | 0.695 | 316.8 | 0.125 | 1.362 |
| 136.8 | 0.23 | 0.615 | 324 | 0.23 | 1.427 |
| 144 | 0.395 | 0.56 | 331.2 | 0.355 | 1.472 |
| 151.2 | 0.545 | 0.5 | 338.4 | 0.5 | 1.482 |
| 158.4 | 0.71 | 0.495 | 345.6 | 0.61 | 1.472 |
| 165.6 | 0.775 | 0.47 | 352.8 | 0.675 | 1.482 |
| 172.8 | 0.825 | 0.48 | 360 | 0.69 | 1.492 |
| 180 | 0.845 | 0.51 | | 0.655 | 1.477 |

ผลการ วัดแบบรูปการแผ่ พลังงานของ สายอากาศในระนาบ สนามแม่เหล็กและ ระนาบ สนามไฟฟ้าโดยภาคส่งสายอากาศยัก และภาครับสายอากาศไดโพลที่มุมเออร์หมุนทีละ 0.9° , 1.8° , 3.6° และ 7.2°

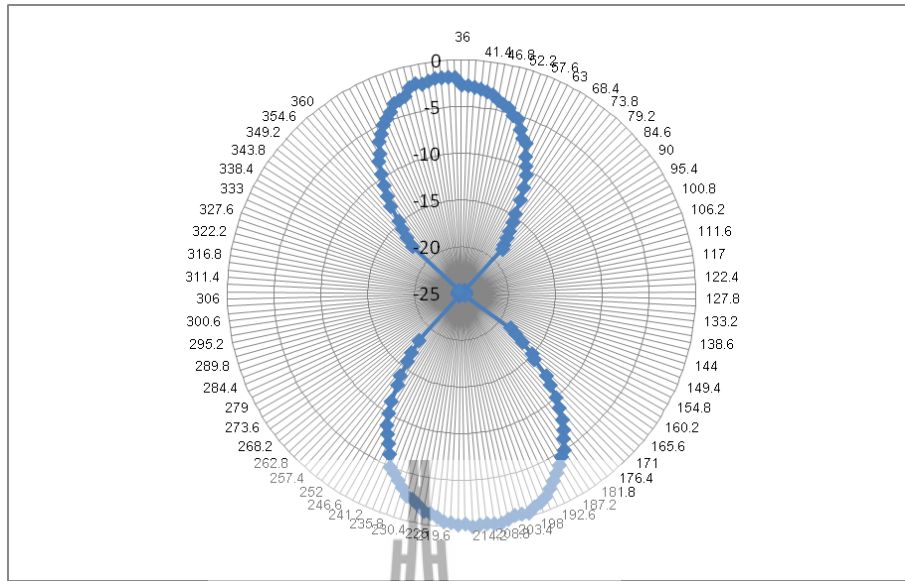


(ก) ระนาบสนามไฟฟ้า

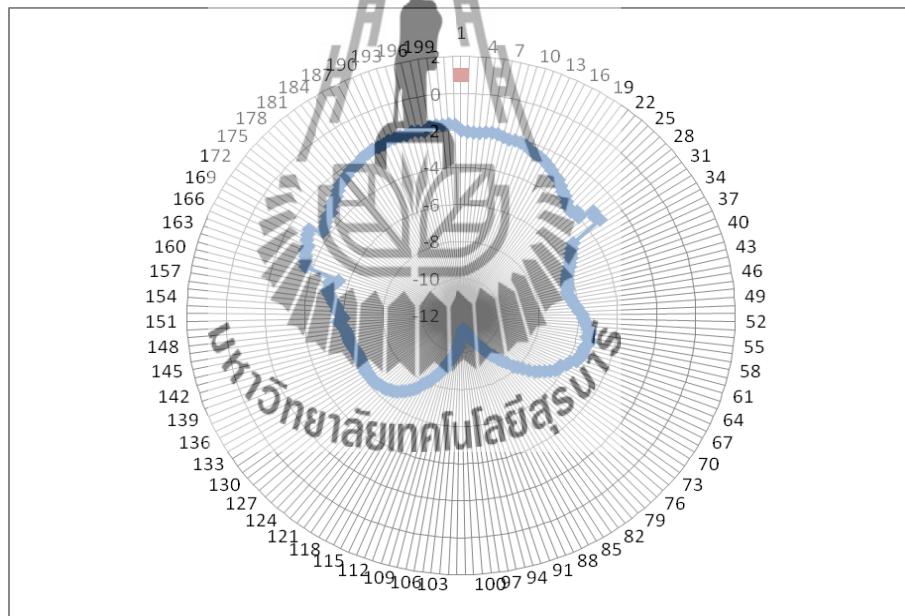


(ข) ระนาบสนามแม่เหล็ก

รูปที่ 4.20 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล
เมื่อมอเตอร์ทำงานเสถียรที่ 0.9°

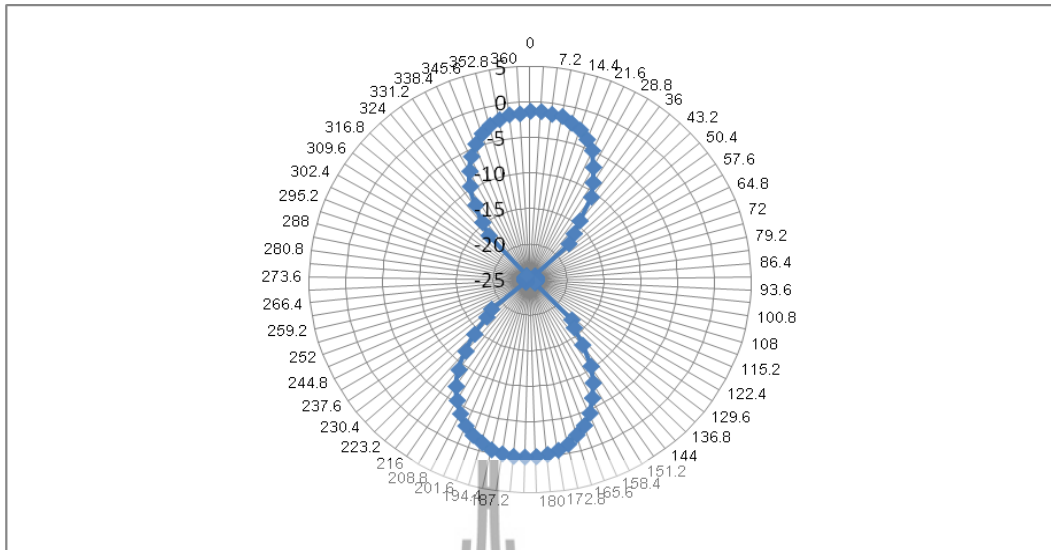


(ก) ระบายสนามไฟฟ้า

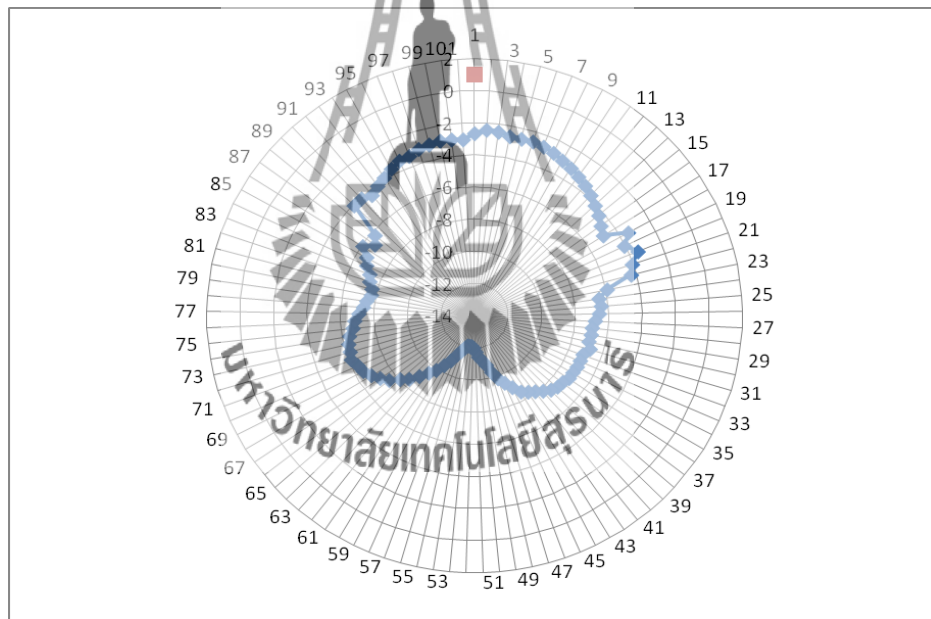


(ข) ระบายสนามแม่เหล็ก

รูปที่ 4.21 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล
เมื่อมอเตอร์ทำงานเสถียรที่ 1.8°

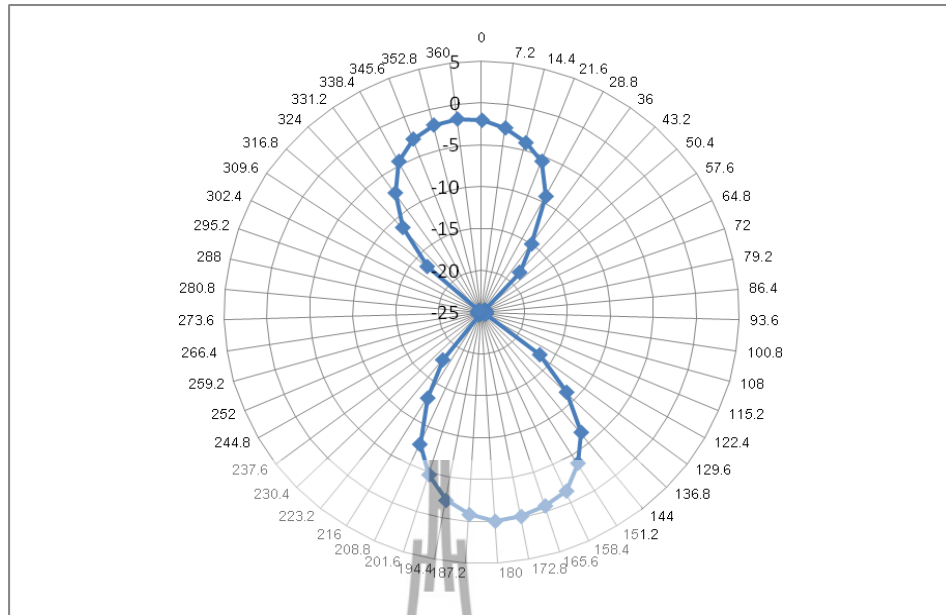


(ก) ระบายสนามไฟฟ้า

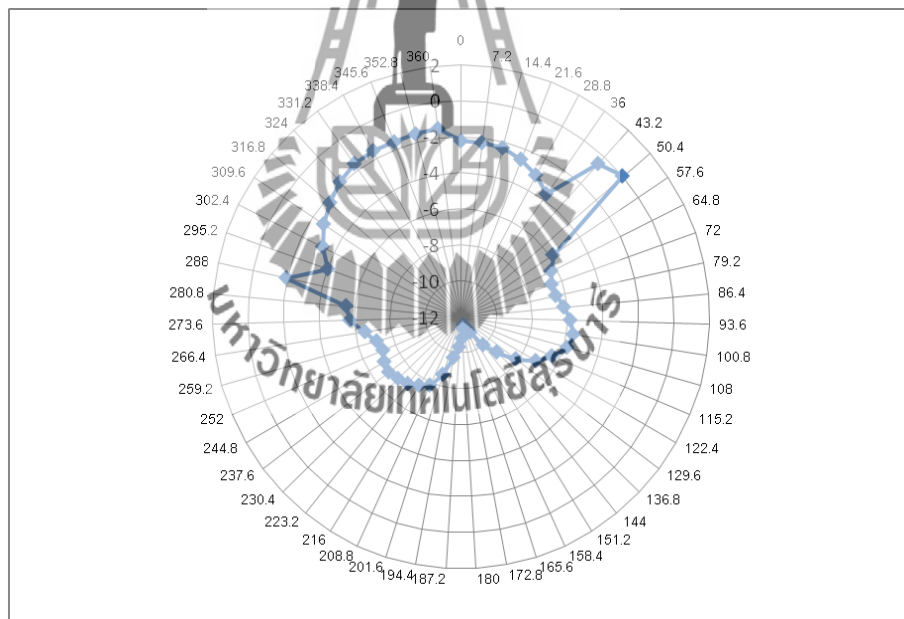


(ข) ระบายสนามแม่เหล็ก

รูปที่ 4.22 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล
เมื่อมอเตอร์ทำงานเสถียรที่ 3.6°



(ก) ระนาบสนามแม่ไฟฟ้า

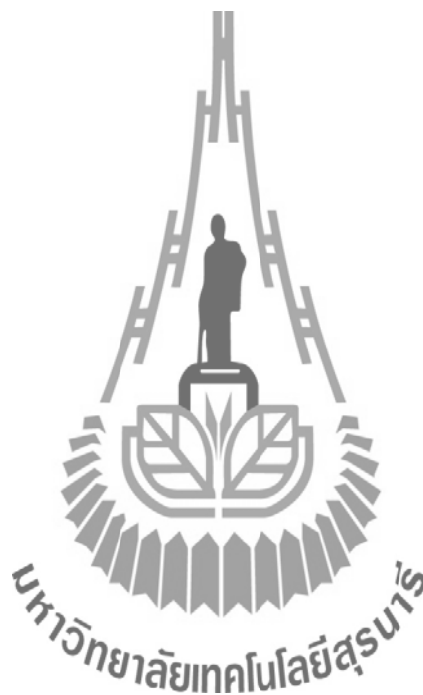


(ข) ระนาบสนามแม่เหล็ก

รูปที่ 4.23 ผลการทดสอบแบบรูปการแผ่พลังงานของสายอากาศไดโพล
เมื่อมอเตอร์ทำงานเสถียรที่ 7.2°

4.5 สรุปการทดสอบเครื่องวัดแบบรูปการแผ่พลังงานของสายอากาศ

เครื่องวัดแบบรูปการแผ่พลังงานของสายอากาศที่ออกแบบและสร้างขึ้นมานี้ สามารถวัดแบบรูปการแผ่พลังงาน แล้วแสดงผลทางคอมพิวเตอร์ได้แบบ Polar Plot ที่ Normalize แล้ว จากการทดสอบได้ทำการวัดแบบรูปการแผ่พลังงานของสายอากาศหลายๆรอบ ในระนาบสนามไฟฟ้าและระนาบแม่เหล็ก พบว่าแบบรูปการแผ่พลังงานที่วัดได้ในแต่ละครั้งมีแบบรูปที่คล้ายคลึงกันทั้งในสองระนาบ แสดงว่าเครื่องวัดแบบรูปการแผ่พลังงานของสายอากาศนี้มีความเที่ยงตรงและแม่นยำ สามารถนำมาใช้งานในการวัดแบบรูปการแผ่พลังงานของสายอากาศได้



บทที่ 5

บทสรุป และข้อเสนอแนะ

5.1 กล่าวนำ

ในบทนี้จะกล่าวถึงระบบรวมของ เครื่องวัดแบบรูปการแผ่กระจายพลังงานความถี่ 1 GHz โดยได้อธิบายถึง ปัญหาที่พบในระหว่างการทำโครงการ วิธีแก้ปัญหา ข้อเสนอแนะ แนวทางการพัฒนา และบทสรุปของโครงการที่จัดทำขึ้น

5.2 ปัญหาที่พบในระหว่างการทำโครงการและวิธีแก้ปัญหา

ตารางที่ 5.1 แสดงรายละเอียดของปัญหาที่พบ และวิธีแก้ปัญหของโครงการ

| ปัญหาที่พบ | สาเหตุและแนวทางแก้ไข |
|---|---|
| 1. วงจรขยายสัญญาณ (Amplifier) ที่ความถี่ 1 GHz | <p><u>สาเหตุ</u> เนื่องจาก การออกแบบวงจรขยายสัญญาณมีขนาดเล็กมาก ซึ่งทำให้ไม่สามารถพิมพ์ลายวงจรเพื่อนำมากัดลงแผ่นปรินต์ได้ และขนาดของไดโอดที่ใช้มีขนาดเล็กมาก ซึ่งยากต่อการบัดกรีวงจร</p> <p><u>แนวทางแก้ไข</u> ขอ ยืมจากห้องปฏิบัติการวิศวกรรมโทรคมนาคม</p> |
| 2. คณะผู้จัดทำโครงการไม่มีความรู้พื้นฐานในการเขียนโปรแกรม | <p><u>สาเหตุ</u> เนื่องจากไม่เคยเขียนและใช้โปรแกรม Visual Basic Studio 2010 มาก่อน</p> <p><u>แนวทางแก้ไข</u> ทำความเข้าใจและทำการศึกษาการใช้โปรแกรมจากหนังสือและการขอคำปรึกษาผู้มีความรู้</p> |
| 3. วงจร Detector | <p><u>สาเหตุ</u> เนื่องจาก ไดโอดของวงจรเป็นแบบชิฟใช้ในย่านความถี่สูงจึงมีขนาดเล็กและหาซื้อได้ยาก</p> <p><u>แนวทางแก้ไข</u> ควรระวังเรื่องความร้อนในการบัดกรีด้วย เช่น ไม่ควรจี้ ขาไดโอดเป็นเวลานานๆ เพราะอาจจะทำให้ ไดโอดพังได้</p> |

| ปัญหาที่พบ | สาเหตุและแนวทางแก้ไข |
|--------------------------------------|--|
| 4. บอร์ด Microcontroller | <p>สาเหตุ เนื่องจากบอร์ดไมโครคอนโทรลเลอร์ไม่สามารถอ่านค่าของสัญญาณได้ เพราะค่าแรงดันที่ได้มีค่าต่ำมาก</p> <p>แนวทางแก้ไข เนื่องจากค่าแรงดันที่ได้มีค่าต่ำมากจนบอร์ดไมโครคอนโทรลเลอร์ไม่สามารถอ่านได้ จึงต้องใช้วงจรขยายสัญญาณด้วยออปแอมป์ 1 kHz</p> <p>เพื่อขยายสัญญาณให้ค่าแรงดันสูงจนไมโครคอนโทรลเลอร์สามารถอ่านค่าได้</p> |
| 5. การสูญเสียในสายส่งและปีจจัยภายนอก | <p>สาเหตุ เนื่องจากสายส่งที่ใช้ในการทดสอบมีการสูญเสียภายในสายและปีจจัยภายนอก จึงส่งผลให้สัญญาณที่วัดออกมาเกิดการผิดเพี้ยน</p> <p>แนวทางแก้ไข ควรเลือกใช้สายที่มีความยาวสั้นลง เพราะสายส่งยังมีขนาดยาวยังมีการสูญเสียมาก หรือใช้สายส่งที่มีการสูญเสีย น้อย แต่สายที่มีการสูญเสียน้อยก็จะมีราคาแพงตามไปด้วย</p> |

5.3 ข้อเสนอแนะ

เนื่องจาก IC และไดโอดมีขนาดเล็ก ควรที่จะระมัดระวังในการบัดกรี เพราะเวลาบัดกรีขาของ IC และไดโอด อาจติดกันทำให้เกิดการ สัดวงจร ได้ และควรระวังเรื่องความร้อนในการบัดกรีด้วย เช่น ไม่ควรจี้ขา ICและไดโอด เป็นเวลานานๆ อาจทำให้ ICและไดโอดพังได้

5.4 แนวทางการพัฒนาต่อไป

- 1.เพิ่มชุดวงจรขยายสัญญาณที่ภาคส่งและภาครับ ให้มากขึ้น เพื่อ ทำให้สัญญาณมีความแรงมากขึ้น สามารถรับ- ส่งสัญญาณได้ระยะทางที่ไกลขึ้น
- 2.ปรับปรุง สเต็ปมอเตอร์ให้หมุนอย่างราบเรียบ ซึ่งจะทำให้ผลในการวัดแบบรูปการแผ่กระจายพลังงานของสายอากาศมีความเที่ยงตรงและแม่นยำยิ่งขึ้น

- 3.ปรับปรุง และ พัฒนาเครื่องวัดแบบรูปการแผ่กระจายพลังงานของสายอากาศ ให้สามารถใช้งานได้ในหลายๆ ย่านความถี่ เช่น ที่ย่านความถี่ 10 GHz
- 4.สามารถเขียนโปรแกรมเพิ่มการรับค่าจากการหมุนของสเต็ปมอเตอร์ โดยสเต็ปมอเตอร์ที่ใช้ในโครงงานนั้น แต่ สเต็ปจะขับเคลื่อนได้ 0.45° , 1.35° หรือ 1.8° ซึ่งแล้วแต่ละโครงสร้างของมอเตอร์

5.5 สิ่งที่ได้รับจากโครงงาน

- 1.ได้รับความรู้เกี่ยวกับ โปรแกรม Visual Basic Studio 2010 และสามารถพัฒนาให้เกี่ยวข้องกับหัวข้อโครงงานได้
- 2.ได้รับความรู้เกี่ยวกับ โปรแกรม Code Vision AVR C Compiler Evaluation และสามารถพัฒนาให้เกี่ยวข้องกับหัวข้อโครงงานได้
- 3.สามารถทำงานร่วมกันเป็นทีม
- 4.นำผลงานมาใช้ในห้องปฏิบัติการสาขาวิชาวิศวกรรมโทรคมนาคม

5.6 บทสรุป

จากการทดสอบเมื่อทำการต่อชุดอุปกรณ์และวงจรต่างๆ ในระบบรวมทั้งหมด แล้วทำการวัดแบบรูปการแผ่กระจายพลังงานของสายอากาศที่ความถี่ 1 GHz โดยกำหนดให้ระยะห่างระหว่างสายอากาศภาครับและภาคส่งห่างกันในระยะ Far field ซึ่งจะมีค่าประมาณ 1 เมตร แล้วทำการวัดค่าพลังงานของสนามแม่เหล็กไฟฟ้าที่ภาครับสามารถรับได้ ทั้งในระนาบ E และระนาบ H เมื่อทำการวัดค่าซ้ำหลายๆ รอบ พบว่าแบบรูปการแผ่กระจายพลังงานที่วัดได้ในแต่ละครั้งมีความคล้ายคลึงกัน ทั้งในสองระนาบ แสดงว่าเครื่องวัดแบบรูปการแผ่กระจายพลังงานของสายอากาศนี้มีประสิทธิภาพและมีความเที่ยงตรงแม่นยำที่สามารถนำมาใช้งานได้จริง

ก่อนที่จะทำการต่อชุดอุปกรณ์และวงจรต่างๆในระบบ จะทำการทดสอบชุดอุปกรณ์และวงจรข้างต้น ซึ่งได้ผลตามบทที่ 3 อธิบายได้ดังต่อไปนี้

5.6.1 วงจรควบคุมการหมุนของสายอากาศด้วยสเต็ปมอเตอร์

สเต็ปมอเตอร์ที่ถูกควบคุมด้วย ไมโครคอนโทรลเลอร์ จะหมุนสายอากาศทีละครั้งสเต็ปโดยจะได้มุม 0.45° ในแต่ละครั้งสเต็ป ดังนั้นสเต็ปมอเตอร์จะหมุน 800 สเต็ปทำให้ได้มุมทั้งหมด 360°

5.6.2 วงจรขยายสัญญาณ (Amplifier) ที่ความถี่ 1 GHz

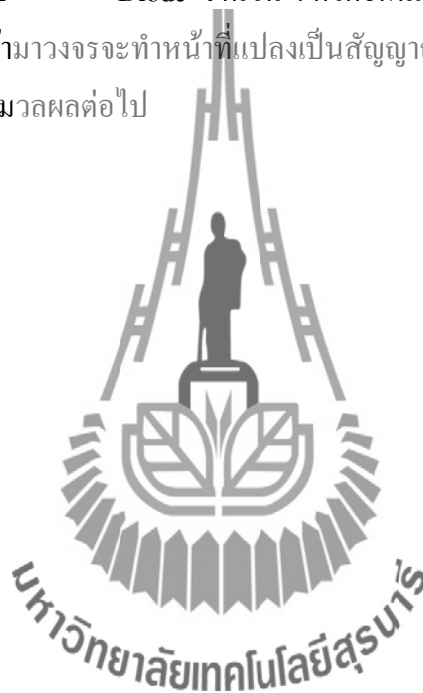
วงจรขยายสัญญาณ (Amplifier) ที่ความถี่ 1 GHz ทำหน้าที่ขยายสัญญาณที่แพร่
ออกมาจากสายอากาศภาคส่งให้มีค่าพลังงานสูงขึ้น เพื่อให้วงจรดีเทคเตอร์สามารถทำงาน

5.6.3 วงจร Detector

วงจร Detector เป็นวงจรที่ทำการดีมอดูเลตสัญญาณ ซึ่งสัญญาณที่ได้ออกมา
นั้นเป็นสัญญาณข่าวสารที่ความถี่ 1 kHz และในวงจรนี้ยังสามารถกรองสัญญาณความถี่ต่ำ

5.6.4 วงจรขยายสัญญาณด้วย Op-amp ที่ความถี่ 1 kHz และวงจรเรียงกระแส

ภายในวงจรจะใช้ Diode จำนวน 4 ตัวต่อกันแบบบริดจ์ เมื่อมีสัญญาณข่าวสารที่
ความถี่ 1 kHz เข้ามาวงจรจะทำหน้าที่แปลงเป็นสัญญาณ DC เพื่อส่งให้โปรแกรมใน
คอมพิวเตอร์ประมวลผลต่อไป



บรรณานุกรม

- [1] <http://www.inventor.in.th/2012/2011-07-06-15-43-20/7-2011-07-03-05-53-51/71-2011-09-15-15-58-45.html>
- [2] http://km.mvc.ac.th/files/1103221212525715_1103240992435.pdf
- [3] <http://www.bloggang.com/viewblog.php?id=megachan&date=27-07-2008&group=1&gblog=5>
- [4] <http://www.etteam.com/product/avr/ManTh-ET-BASE%20AVR%20ATmega64-128%20r3.pdf>
- [5] http://www.etteam.com/product/avr/avr-base-atmega128/ET-BASE%20AVR%20ATmega64-128_Schematic.pdf
- [6] <http://icelectronic.com/beginner/study/powersup.htm#rectifier>
- [7] http://www.kmitl.ac.th/~kpteeraw/data_com/datacom_52/Filter.htm
- [8] ผศ.ดร.รังสรรค์ วงศ์สรรค์. เอกสารประกอบการเรียนวิชาวิศวกรรมสายอากาศ. มหาวิทยาลัยเทคโนโลยีสุรนารี, นครราชสีมา.2547



ภาคผนวก ก

โปรแกรม CodeVisionAVR C Compiler Evaluation ที่ใช้ในการ
ควบคุมการหมุนของสเต็ปมอเตอร์

```

#include <mega128.h> // ATmega128 MCU
#include <stdio.h> // Standard Input/Output functions
#define ADC_VREF_TYPE 0xC0 // ADC Used Internal Reference

unsigned int read_adc(unsigned char adc_input); // Read ADC Result
void FW_Step(int step);
void delay (void);

void main(void)
{
    char uart_data;
    unsigned int j=1,val,val1,val2; // ADC Result

    DDRA = 0x7F;

    UCSR0A=0x00;
    UCSR0B=0x18;
    UCSR0C=0x86;
    UBRR0H=0x00;
    UBRR0L=0x67;

    ADMUX=ADC_VREF_TYPE;
    ADCSRA=0x87;
    SFIOR&=0xEF;

```



```

PORTA = 0x00;
delay();
while (1)
{
    uart_data = getchar();
    if(uart_data=='s')
    {
        printf("\rStop");
        PORTA = 0x00;
    }
    if(uart_data=='d')
    {
        val = read_adc(0);
        val1=(val*5)/1023;
        val2=(val*5)%1023;
        if (val < 20)
        { printf("%u.05\n",val1); }
        else
        { printf("\r%u.%u",val1,val2); }
    }
    Else
    {
        if(uart_data=='l')
        {
            if(j<8){j++;}
            else if(j==8){j=1;}
            FW_Step(j);
        }
        if(uart_data=='r')
        {

```

```

        if(j>1){j--;}
        else if(j==1){j=8;}
        FW_Step(j);
    }
}
}
}

/*****/;
/*****/ Read the AD conversion result *****/;
/*****/;
unsigned int read_adc(unsigned char adc_input) // Read Result ADC
{
    ADMUX=adc_input|ADC_VREF_TYPE;
    ADCSRA|=0x40; // Start the AD conversion
    while ((ADCSRA & 0x10)==0); // Wait for the AD conversion to
                                // complete
    ADCSRA|=0x10;
    return ADCW;
}

/*****/;
/*****/FW-STEP*****/;
/*****/;

void FW_Step(int step)
{
    switch(step)
    {
        case 1: PORTA=0b00000001;delay();break;
        case 2: PORTA=0b00000011;delay();break;
        case 3: PORTA=0b00000010;delay();break;
        case 4: PORTA=0b00000110;delay();break;
    }
}

```


โปรแกรม Visual Basic Studio 2010
ที่ใช้ในการออกแบบโปรแกรมวัดแบบรูปการแผ่พลังงานของสายอากาศ

```
Imports System.Drawing.Drawing2D
```

```
Imports System.Math
```

```
Public Class Form1
```

```
    Dim rxBuff As String
```

```
    Dim command As String
```

```
    Dim num As Double
```

```
    Dim Data_Show As String
```

```
    Dim Data_Save As String
```

```
    Dim Data_Show_Degree As String
```

```
    Dim Step_Motor As Integer
```

```
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
        Handles MyBase.Load
```

```
            Rad_left.Checked = True
```

```
            Btn_stop.Enabled = False
```

```
            BtnPause.Enabled = False
```

```
            BtnReset.Enabled = False
```

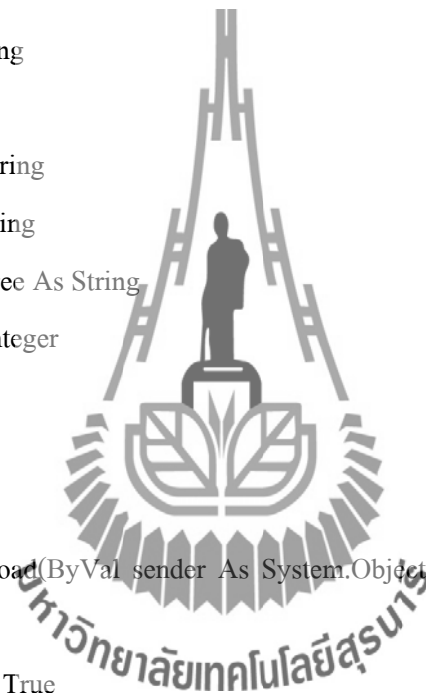
```
            BtnSave.Enabled = False
```

```
            Label5.Text = "....."
```

```
            cmbBaudRate.SelectedIndex = 5
```

```
            cmbPortName.SelectedIndex = 0
```

```
            CmbDegree.SelectedIndex = 0
```



End Sub

```
Private Sub button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
```

```
Handles btn_connect.Click
```

```
    If btn_connect.Text = "Connect" Then
```

```
        btn_connect.Text = "Disonnect"
```

```
        label2.Text = "Connect !"
```

```
        Label4.Text = "COM: " + cmbPortName.Text + " , " + "BaudRate: " +  
cmbBaudRate.Text
```

```
        SerialPort1.PortName = cmbPortName.Text
```

```
        SerialPort1.BaudRate = cmbBaudRate.Text
```

```
        SerialPort1.Parity = IO.Ports.Parity.None
```

```
        SerialPort1.StopBits = IO.Ports.StopBits.One
```

```
        SerialPort1.DataBits = 8
```

```
        SerialPort1.Open()
```

```
        Label5.Text = "Ready !"
```

```
    Else
```

```
        btn_connect.Text = "Connect"
```

```
        label2.Text = "Disconnect !"
```

```
        Label4.Text = ""
```

```
        Label5.Text = "....."
```

```
        SerialPort1.Close()
```

```
    End If
```

End Sub


```
Private Sub Btn_start_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Btn_start.Click
```

```
    BtnReset.Enabled = False
```

```
    Step_Motor = CmbDegree.SelectedIndex
```

```
    num = 0
```

```
    Data_Show = ""
```

```
    Data_Show_Degree = ""
```

```
    Data_Save = ""
```

```
    If Rad_left.Checked = True Then
```

```
        command = "l"
```

```
    Else
```

```
        command = "r"
```

```
    End If
```

```
    If SerialPort1.IsOpen = True Then
```

```
        Label5.Text = "Processing !"
```

```
        Btn_start.Enabled = False
```

```
        Btn_stop.Enabled = True
```

```
        BtnPause.Enabled = True
```

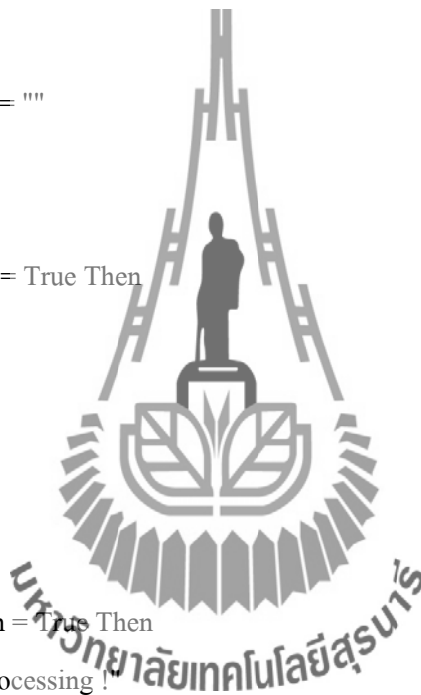
```
        CmbDegree.Enabled = False
```

```
        Rad_left.Enabled = False
```

```
        Rad_right.Enabled = False
```

```
        Timer1.Enabled = True
```

```
        BtnSave.Enabled = False
```



```

Else
    MessageBox.Show("เชื่อมต่อ Serial Port ก่อน !", "ผิดพลาด", MessageBoxButtons.OK,
    MessageBoxIcon.Error)
    Btn_start.Enabled = True
    Btn_stop.Enabled = False
    CmbDegree.Enabled = True
    Rad_left.Enabled = True
    Rad_right.Enabled = True
    BtnPause.Enabled = False
    BtnSave.Enabled = True

End If

End Sub

Private Sub Btn_stop_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Btn_stop.Click

    Label5.Text = "Ready !"
    BtnReset.Enabled = True
    Btn_stop.Enabled = False
    Btn_start.Enabled = True
    BtnPause.Enabled = False
    Timer1.Enabled = False
    CmbDegree.Enabled = True
    Rad_left.Enabled = True
    Rad_right.Enabled = True
    BtnSave.Enabled = True

```



End Sub

Private Sub Timer1_Tick(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles Timer1.Tick

SerialPort1.Write("d")

Do

rxBuff = ""

System.Threading.Thread.Sleep(50)

rxBuff = (SerialPort1.ReadExisting)

System.Threading.Thread.Sleep(50)

Loop While rxBuff = ""

Data_Save = Data_Save & CStr(rxBuff)

Data_Show = Data_Show & CStr(rxBuff) & vbCrLf

TextBox1.Text = Data_Show

System.Threading.Thread.Sleep(50)

If num < 360 Then

Select Case Step_Motor

Case 0

SerialPort1.Write(command)

System.Threading.Thread.Sleep(100)

SerialPort1.Write(command)

System.Threading.Thread.Sleep(100)


```
num = (num + 3.6)
```

Case 3

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```
SerialPort1.Write(command)
```

```
System.Threading.Thread.Sleep(100)
```

```

SerialPort1.Write(command)
System.Threading.Thread.Sleep(100)
SerialPort1.Write(command)
System.Threading.Thread.Sleep(100)

```

```

num = (num + 7.2)

```

```

End Select

```

```

Data_Show_Degree = Data_Show_Degree & CStr(num) & vbCrLf
TextBox2.Text = Data_Show_Degree

```

```

Else

```

```

Timer1.Enabled = False
BtnReset.Enabled = True
Btn_stop.Enabled = False
Btn_start.Enabled = True
CmbDegree.Enabled = True
Rad_left.Enabled = True
Rad_right.Enabled = True
BtnSave.Enabled = True
Label5.Text = "Success !"

```

```

End If

```

```

End Sub

```

```

Private Sub BtnReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnReset.Click

```

```

Label5.Text = "Reset!"
TextBox1.Text = ""
TextBox2.Text = ""
Timer1.Enabled = False
BtnSave.Enabled = False
End Sub

```

```

Private Sub BtnPause_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles BtnPause.Click
    If Label5.Text = "Processing !" Then
        Label5.Text = "Pause !"
        Btn_stop.Enabled = True
        BtnSave.Enabled = True
        Timer1.Enabled = False
    Else
        Label5.Text = "Processing !"
        Btn_stop.Enabled = False
        BtnSave.Enabled = False
        Timer1.Enabled = True
    End If
End Sub

```

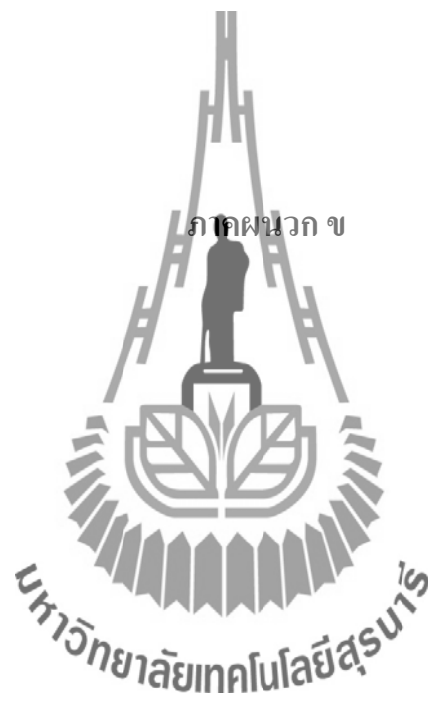
```

Private Sub SaveFileDialog1_FileOk(ByVal sender As System.Object, ByVal e As
System.ComponentModel.CancelEventArgs) Handles SaveFileDialog1.FileOk
    Dim writer As New IO.StreamWriter(SaveFileDialog1.FileName) 'Creates the 'writer'
    writer.Write(Data_Save) 'This will write it.
    writer.Close() 'Closes it.
End Sub

```

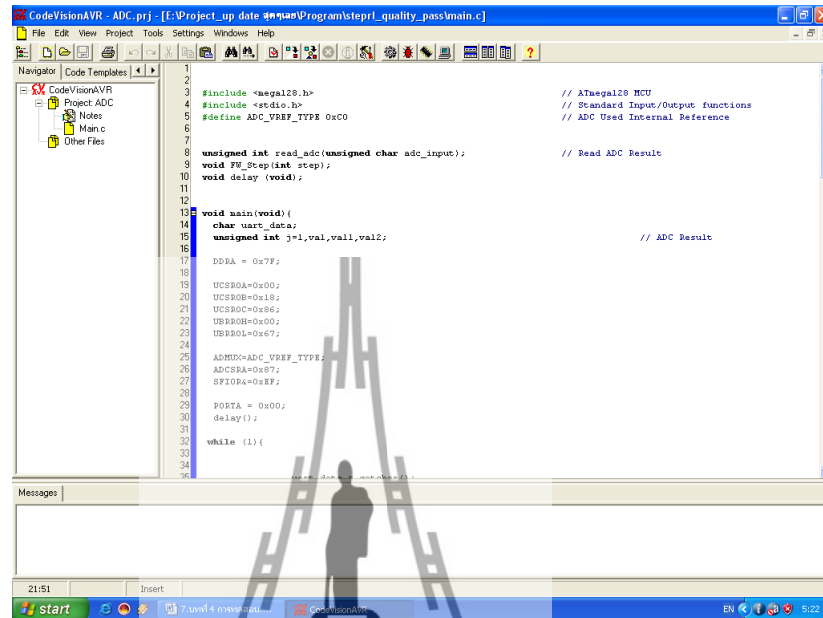
```
Private Sub BtnSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)  
Handles BtnSave.Click  
    SaveFileDialog1.ShowDialog()  
  
End Sub  
End Class
```





ปิดโปรแกรมภาษา C

- 1.เปิดโปรแกรม เพื่อใช้ในการควบคุมการหมุนของสเต็ปมอเตอร์
แล้วทำการ Compile

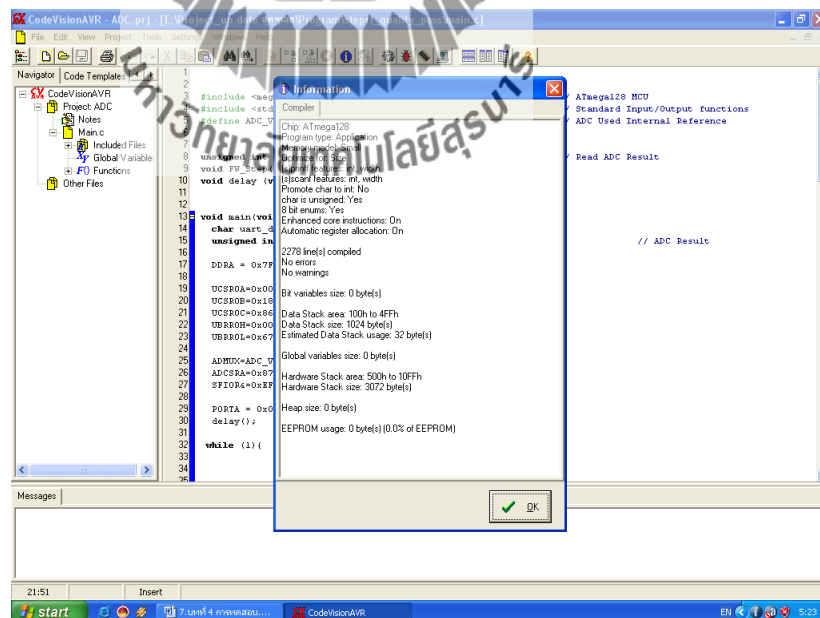


```

1  #include <mega128.h> // ATmega128 MCU
2
3  #include <stdio.h> // Standard Input/Output functions
4  #define ADC_VREF_TYPE 0x00 // ADC Used Internal Reference
5
6
7
8  unsigned int read_adc(unsigned char adc_input); // Read ADC Result
9  void FW_Step(int step);
10 void delay(void);
11
12
13 void main(void) {
14     char uart_data;
15     unsigned int j=1, val, val1, val2; // ADC Result
16
17     DDRA = 0x7F;
18
19     UCSRA=0x00;
20     UCSRB=0x18;
21     UCSRC=0x86;
22     UBRR0H=0x00;
23     UBRR0L=0x67;
24
25     ADMUX=ADC_VREF_TYPE;
26     ADCSRA=0x87;
27     SFIOR=0x8F;
28
29     PORTA = 0x00;
30     delay();
31
32     while (1) {
33
34
35

```

แสดงเปิดโปรแกรมภาษา C



```

1  #include <mega128.h> // ATmega128 MCU
2
3  #include <stdio.h> // Standard Input/Output functions
4  #define ADC_VREF_TYPE 0x00 // ADC Used Internal Reference
5
6
7
8  unsigned int read_adc(unsigned char adc_input); // Read ADC Result
9  void FW_Step(int step);
10 void delay(void);
11
12
13 void main(void) {
14     char uart_data;
15     unsigned int j=1, val, val1, val2; // ADC Result
16
17     DDRA = 0x7F;
18
19     UCSRA=0x00;
20     UCSRB=0x18;
21     UCSRC=0x86;
22     UBRR0H=0x00;
23     UBRR0L=0x67;
24
25     ADMUX=ADC_VREF_TYPE;
26     ADCSRA=0x87;
27     SFIOR=0x8F;
28
29     PORTA = 0x00;
30     delay();
31
32     while (1) {
33
34
35

```

Information

Compiler

Chip: ATmega128

Program type: Application

Enhanced core instructions: On

Automatic register allocation: On

2278 line(s) compiled

No errors

No warnings

8x variables size: 0 byte(s)

Data Stack area: 100h to 4FFh

Data Stack size: 1024 byte(s)

Estimated Data Stack usage: 32 byte(s)

Global variables size: 0 byte(s)

Hardware Stack area: 500h to 10FFh

Hardware Stack size: 3072 byte(s)


Heap size: 0 byte(s)

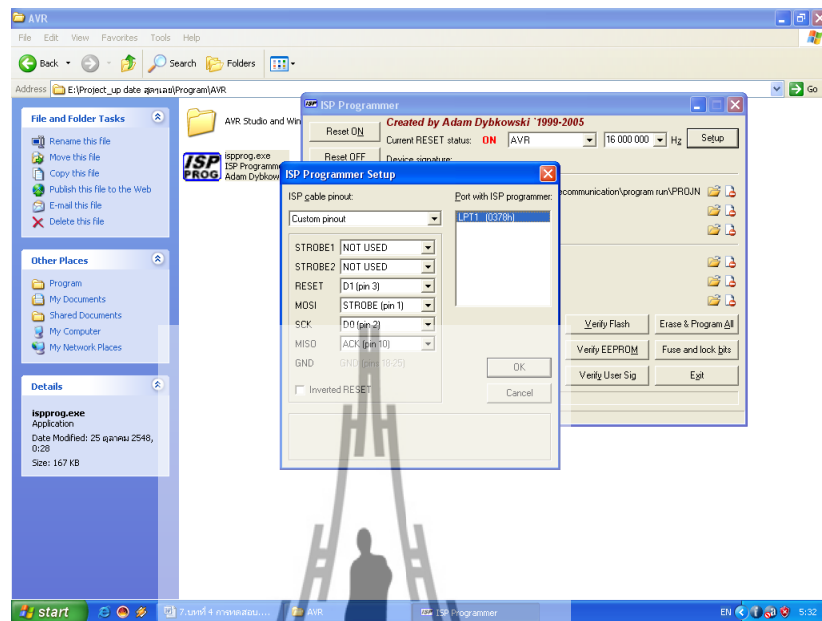
EEPROM usage: 0 byte(s) (0.0% of EEPROM)

OK

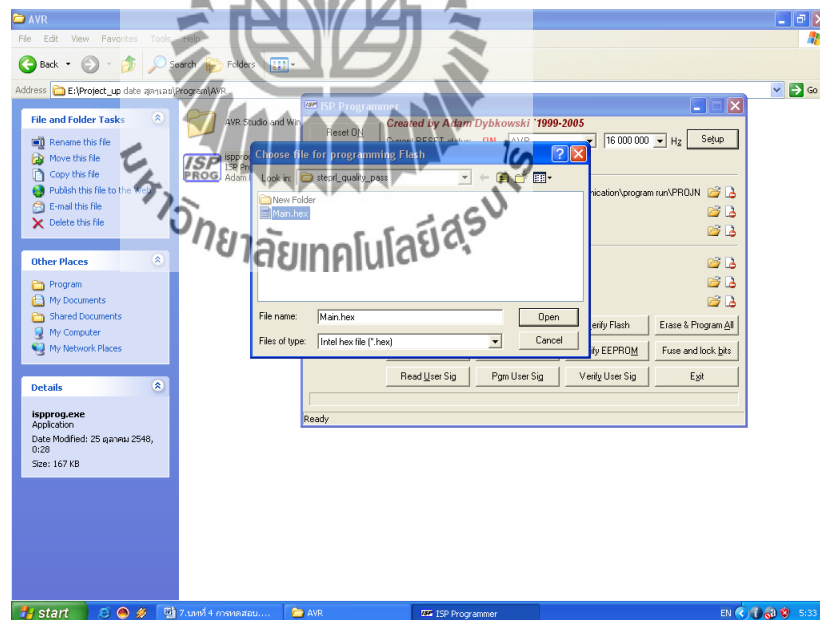
ทำการ Compile

2. ทำการโหลดโปรแกรมภาษา C ลงในบอร์ดไมโครคอนโทรลเลอร์

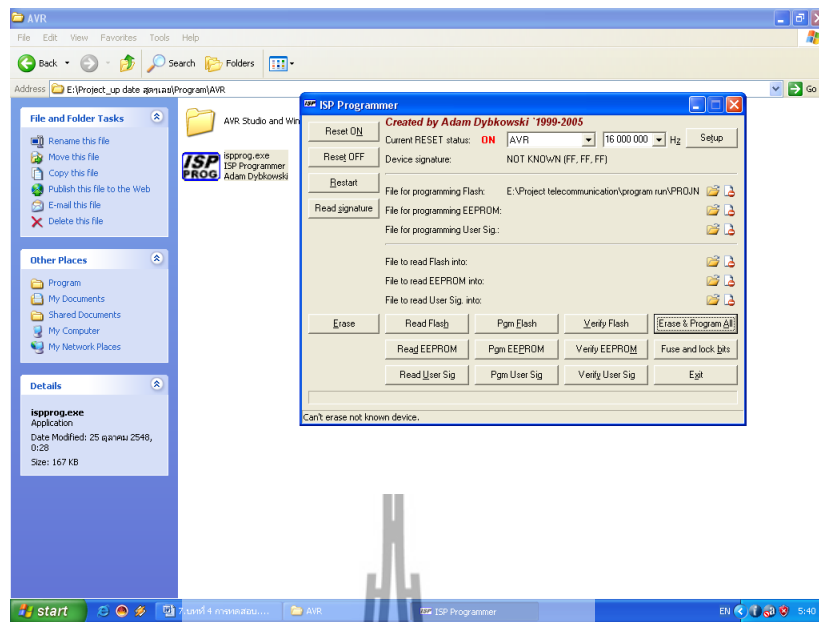
โดยใช้  และต้อง Setup ค่าต่างๆ ก่อน แล้วจึงโหลดโปรแกรม



Setup ค่าต่างๆ



เปิดไฟล์ที่ได้จากการ Compile



ทำการโหลดโปรแกรมภาษา C



ET-BASE AVR ATmega64/128 r3

ET-BASE AVR ATmega64/128 r3 เป็นบอร์ดไมโครคอนโทรลเลอร์ในตระกูล AVR ของบริษัท Atmel ซึ่งบอร์ดนี้เลือกใช้ MCU เบอร์ ATmega64 และ เบอร์ ATmega128 ขนาด 64 Pin โดยในบอร์ด ET-BASE AVR ATmega64/128 r3 นี้จะเน้นจะเน้นการใช้งานทรัพยากรของตัว MCU เองเป็นหลัก ซึ่งจะมีการต่อขาสัญญาณ I/O ออกมาจัดเรียงให้เป็นพอร์ต PA,PB,PC,PD,PE,PF และพอร์ต ET-CLCD เพื่อสะดวกต่อการใช้งาน พร้อมทั้งพอร์ตสำหรับดาวน์โหลดโปรแกรม นอกจากนี้ยังได้เพิ่มวงจร Line Driver RS-232 เข้าไปด้วยเพื่อให้สามารถใช้งานทางด้านพอร์ตอนุกรม RS-232 ได้ง่ายและสะดวกยิ่งขึ้น

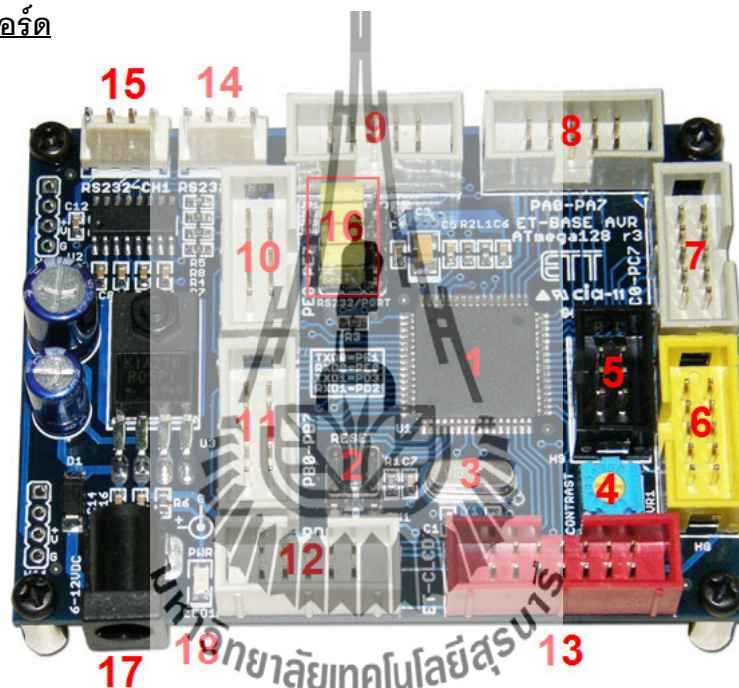
คุณสมบัติของบอร์ด

- เลือกใช้ MCU ตระกูล AVR เบอร์ ATmega64 , ATmega128 ของ Atmel ซึ่งเป็น MCU ขนาด 8-Bit โดยเลือกใช้แหล่งกำเนิดสัญญาณนาฬิกาแบบ XTAL ค่า 16 MHz ซึ่งคุณสมบัติเด่น ๆ ของ MCU ได้แก่
 - มีหน่วยความจำ Flash สำหรับเขียนโปรแกรม 64 KBytes สำหรับ ATmega64 และ 128K Bytes สำหรับ ATmega128 และมี RAM 4 KBytes
 - มีหน่วยความจำข้อมูลถาวรแบบ EEPROM ขนาด 2K Bytes สำหรับ ATmega64 และ 4 K Byte สำหรับ ATmega128 ซึ่งสามารถลบและเขียนซ้ำได้กว่า 100,000 ครั้ง
 - จำนวน I/O สูงสุดถึง 53 I/O Pins
 - มีวงจรสื่อสาร SPI จำนวน 1 ช่อง , I2C จำนวน 1 ช่อง , Programmable Serial USARTs จำนวน 2 ช่อง
 - มี ADC ขนาด 10-Bit จำนวน 8 ช่อง
 - มี Timers/Counters 8-Bit จำนวน 2 ช่อง , Timers/Counters 16-Bit จำนวน 2 ช่อง , 8-Bit PWM 2 ช่อง , Watchdog Timer , Real Time Counter
- I/O PORT 10 PIN จำนวน 6 PORT ดังนี้ PA,PB,PC,PD,PE,PF
- พอร์ต ISP LOAD สำหรับโปรแกรม MCU (ต้องใช้ร่วมกับ ET-AVR ISP หรือเครื่องโปรแกรม ISP อื่นที่มีการจัดเรียงขาสัญญาณเหมือนกัน)
- วงจร Line Driver สำหรับพอร์ตสื่อสารอนุกรม RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่อง จะต่อกับ

สัญญาณ PD2(RXD1) และ PD3(TXD1) เพื่อให้ผู้ใช้สามารถต่อทดลองการติดต่อสื่อสาร RS232

- วงจรเชื่อมต่อกับจอแสดงผล LCD แบบ Character (ET-CLCD) พร้อม VR ปรับความเข้มของ LCD ซึ่งใช้การเชื่อมต่อกับวงจรกับ LCD แบบ 4 Bit Interface
- วงจร Regulate ขนาด +5V / 2A สำหรับใช้งานเป็นแหล่งจ่ายไฟเลี้ยงวงจรให้กับจอแสดงผล LCD และอุปกรณ์ I/O ต่างๆที่ใช้กับแหล่งจ่ายขนาด +5V พร้อม LED แสดงสถานะสีแดง
- ขนาด PCB Size เล็กเพียง 8 X 6 cm

โครงสร้างของบอร์ด



- หมายเลข 1 คือ MCU เบอร์ ATmega64 หรือ ATmega128 ซึ่งเป็น MCU ตระกูล AVR จาก ATMEL
- หมายเลข 2 คือ Switch RESET ใช้สำหรับ Reset การทำงานของ MCU
- หมายเลข 3 คือ Crystal ค่า 16 MHz
- หมายเลข 4 คือ ตัวต้านทานสำหรับปรับค่าความเข้มให้ LCD
- หมายเลข 5 พอร์ต AVR ISP (6 PIN) ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU
- หมายเลข 6 พอร์ต AVR ISP (10 PIN) ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU
- หมายเลข 7 คือ PORTC มีขนาด 8 Bit คือ PC0-PC7

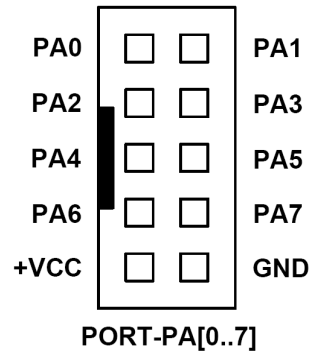
- หมายเลข 8 คือ PORTA มีขนาด 8 Bit คือ PA0-PA7
- หมายเลข 9 คือ PORTF มีขนาด 8 Bit คือ PF0-PF7
- หมายเลข 10 คือ PORTE มีขนาด 8 Bit คือ PE0-PE7
- หมายเลข 11 คือ PORTB มีขนาด 8 Bit คือ PB0-PB7
- หมายเลข 12 คือ PORTD มีขนาด 8 Bit คือ PD0-PD7
- หมายเลข 13 คือ พอร์ต ET-CLCD สำหรับเชื่อมต่อกับ LCD ชนิด Character Type ซึ่งใช้การเชื่อมต่อแบบ 4 Bit
- หมายเลข 14 และ 15 คือ ขั้วต่อ RS232 สำหรับใช้งานทั่วไป
- หมายเลข 16 คือ จัมเปอร์ สำหรับเลือกใช้งาน RS232 หรือ พอร์ต IO
- หมายเลข 17 คือ ขั้วต่อแหล่งจ่ายไฟสำหรับเลี้ยงวงจรของบอร์ด
- หมายเลข 18 คือ LED Power ใช้สำหรับแสดงสถานะของแหล่งจ่ายไฟ +5VDC

ขั้วต่อสัญญาณต่าง ๆ

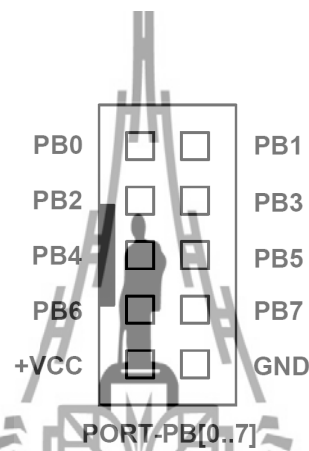
สำหรับขั้วต่อสัญญาณของพอร์ต I/O จาก MCU นั้นจะถูกออกแบบและจัดเตรียมไว้ผ่านทางขั้วต่อแบบ IDC-Header ขนาด 10 Pin (2X5) จำนวน 6 ชุด คือ PA,PB,PC,PD,PE,PF ตามลำดับ โดยที่ขั้วต่อสัญญาณแต่ละชุด จะประกอบไปด้วยสัญญาณของ I/O ที่เชื่อมต่อนอกจากขาสัญญาณของ MCU โดยตรงทั้งหมด โดยจุดเชื่อมต่อกับสัญญาณภายนอกบอร์ดมีดังนี้

- ขั้วต่อแหล่งจ่ายไฟสำหรับเลี้ยงวงจรของบอร์ด
- ขั้วต่อ PORTA มีขนาด 8 Bit คือ PA0-PA7
- ขั้วต่อ PORTB มีขนาด 8 Bit คือ PB0-PB7
- ขั้วต่อ PORTC มีขนาด 8 Bit คือ PC0-PC7
- ขั้วต่อ PORTD มีขนาด 8 Bit คือ PD0-PD7
- ขั้วต่อ PORTE มีขนาด 8 Bit คือ PE0-PE7
- ขั้วต่อ PORTF มีขนาด 8 Bit คือ PF0-PF7
- ขั้วต่อ ET-CLCD สำหรับเชื่อมต่อกับ LCD ชนิด Character Type
- ขั้วต่อ RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่อง จะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1) เพื่อให้ผู้ใช้สามารถทดสอบการติดต่อสื่อสาร RS232
- ขั้วต่อ AVR ISP ใช้สำหรับดาวน์โหลด Hex File ให้กับ MCU

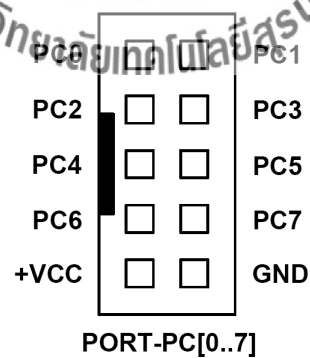
พอร์ต PA มีขนาด 8 บิต



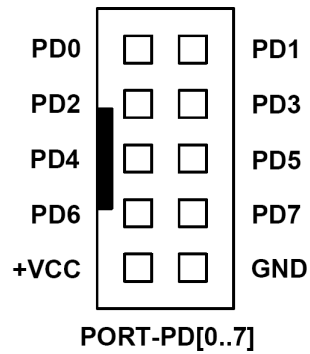
พอร์ต PB มีขนาด 8 บิต



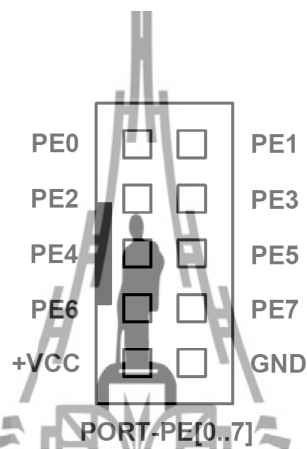
พอร์ต PC มีขนาด 8 บิต



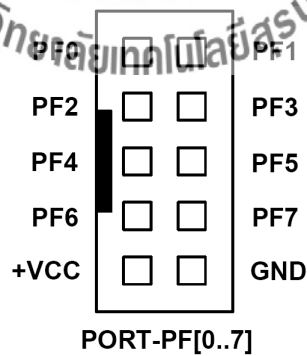
พอร์ต PD มีขนาด 8 บิต



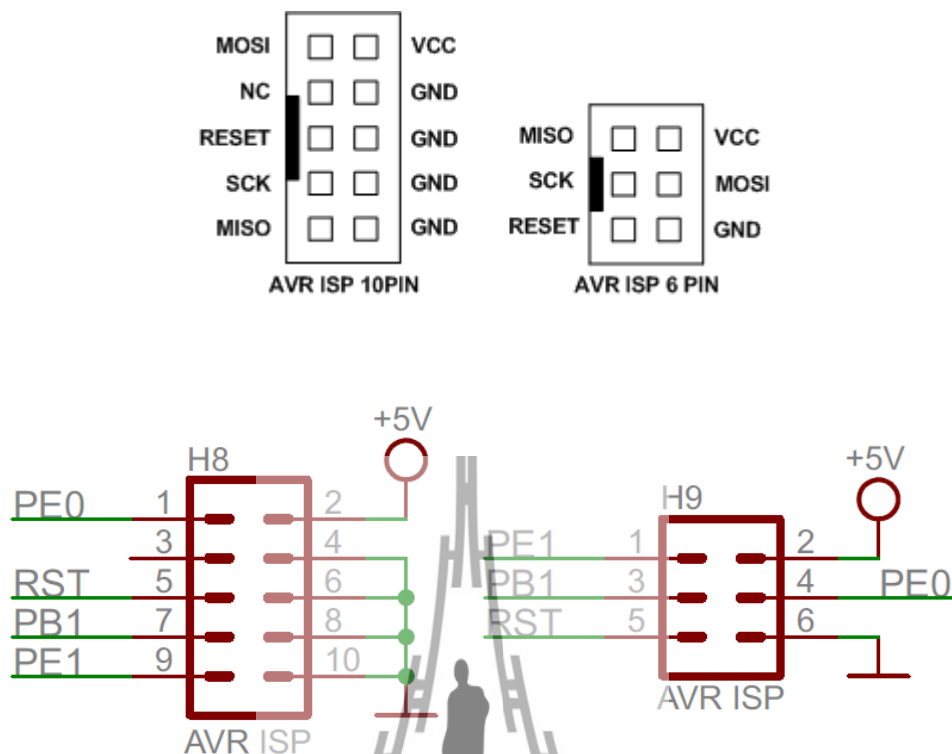
พอร์ต PE มีขนาด 8 บิต



พอร์ต PF มีขนาด 8 บิต

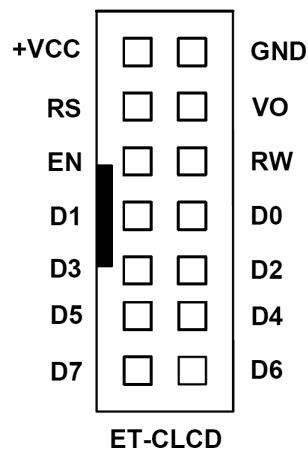


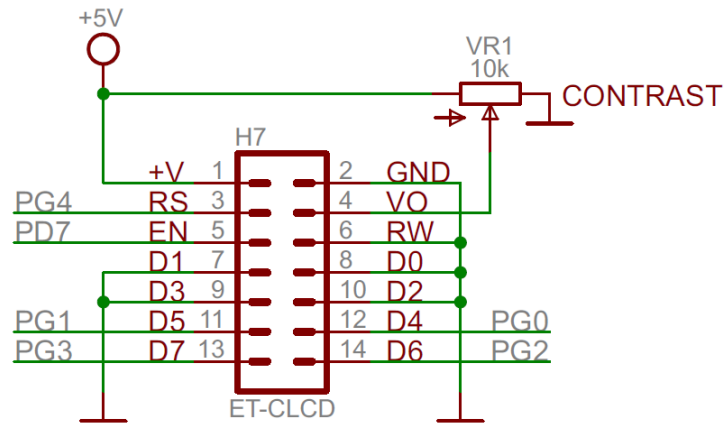
พอร์ต AVR ISP



รูปแสดง วงจรส่วนที่เชื่อมต่อกับ AVR ISP

พอร์ต ET-CLCD ใช้กับ Character Type LCD โดยให้การเชื่อมต่อแบบ 4 บิต โดยสัญญาณที่ใช้เชื่อมต่อกับ LCD จะเป็นสัญญาณจากพอร์ต PG และ PD (PD7) โดยในการเชื่อมต่อสายสัญญาณจากขั้วต่อของพอร์ต LCD ไปยังจอแสดงผล LCD นั้นให้ยึดข้อขาสัญญาณเป็นจุดอ้างอิง โดยให้ต่อสัญญาณที่มีชื่อตรงกันเข้าด้วยกันให้ครบทั้ง 14 เส้น

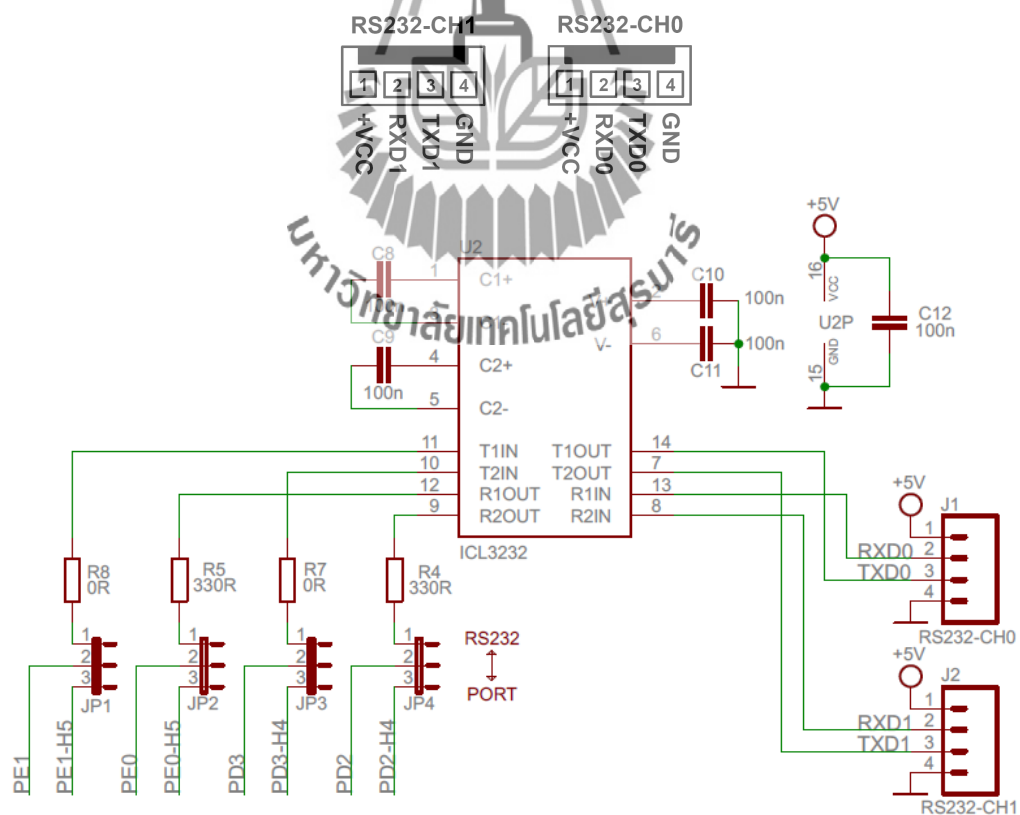




| | | | | | | | | | | | | | |
|-----|------|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| GND | +VCC | VO | RS | RW | EN | D0 | D1 | D2 | D3 | D4 | D5 | D6 | D7 |

แสดงการจัดเรียงขาสัญญาณของ Character LCD มาตรฐาน

พอร์ต RS232 จำนวน 2 ช่อง โดยเชื่อมต่อกับสัญญาณ PE0(RXD0) และ PE1(TXD0) จำนวน 1 ช่อง ส่วนที่เหลืออีก 1 ช่อง จะต่อกับสัญญาณ PD2(RXD1) และ PD3(TXD1)

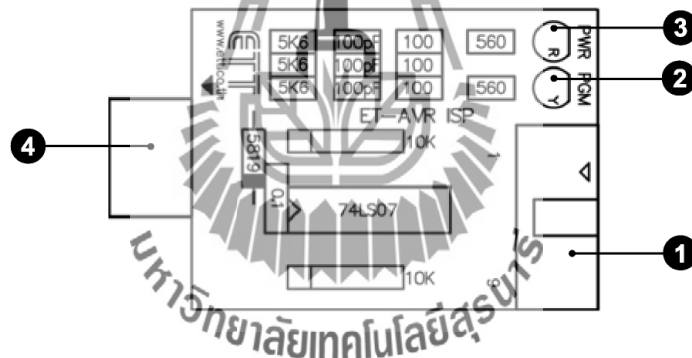


รูปแสดง วงจรส่วนที่เชื่อมต่อกับ RS232

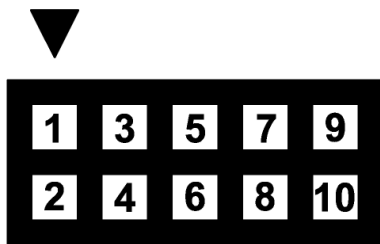
การดาวน์โหลด Hex File ให้กับ MCU

การดาวน์โหลด Hex File ให้กับ MCU นั้นจำเป็นจะต้องใช้ ET-AVR ISP หรือเครื่องโปรแกรมแบบ ISP อื่นๆ เช่น AVRISP ของ ATMEL เพื่อใช้ในการดาวน์โหลด Hex File ให้กับ MCU ตระกูล AVR ของ Atmel โดยใช้วิธีการแบบ Serial Programming ซึ่งการดาวน์โหลด Hex File ในกรณีที่ใช้ ET-AVR ISP จะกระทำผ่านทางพอร์ตขนานของคอมพิวเตอร์ โดยที่จะต้องใช้งานร่วมกับ ET-CAP10P ของอีทีที และ Software ที่ใช้ร่วมกับ ET-AVR ISP ก็คือ PonyProg2000 ซึ่ง PonyProg2000 เป็นโปรแกรม Download ข้อมูลแบบ HEX File ให้กับ CPU ตระกูล AVR โดยใช้วิธีการแบบ Serial Programming ซึ่งสามารถใช้งานกับบอร์ดตระกูล AVR ของ อีทีที ได้เป็นอย่างดี ซึ่งวิธีการใช้งานโปรแกรมโดยทั่วไปนั้น สามารถศึกษาได้จาก Help ของโปรแกรมได้เอง โดยในที่นี้จะขอแนะนำให้ทราบถึงวิธีการ Setup โปรแกรม PonyProg2000 เพื่อใช้งานกับบอร์ดตระกูล AVR ของ อีทีที ซึ่งสามารถใช้งานได้กับบอร์ดตระกูล AVR ทุกรุ่นของ อีทีที

โครงสร้างของบอร์ด ET-AVR ISP



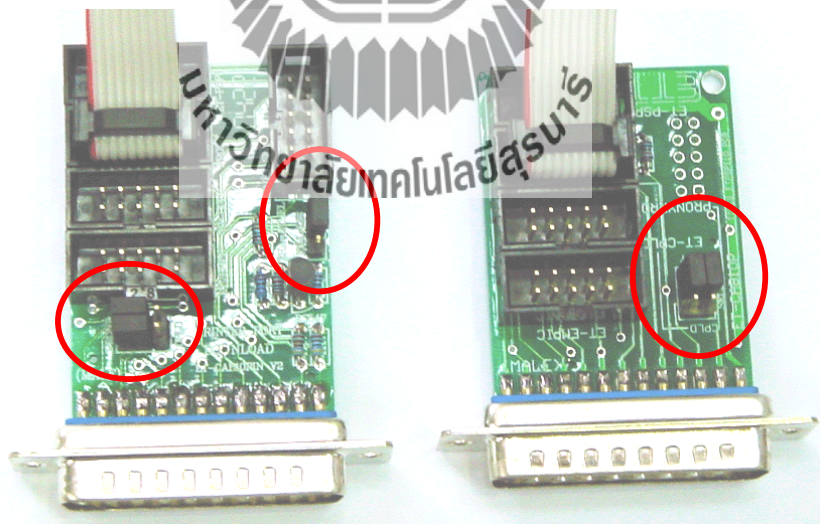
- **หมายเลข 1** คือ พอร์ตสำหรับเชื่อมต่อกับ ET-CAP10P ของอีทีที เพื่อโปรแกรม Hex File ให้กับ MCU
- **หมายเลข 2** คือ LED PGM (สีเขียว) แสดงสถานะของการโปรแกรมหรือดาวน์โหลด Hex File ลง MCU
- **หมายเลข 3** คือ LED PWR (สีแดง) แสดงสถานะของไฟเลี้ยงบอร์ด
- **หมายเลข 4** คือ พอร์ตสำหรับเชื่อมต่อกับบอร์ด Target ซึ่งสามารถใช้โปรแกรม Hex File ให้กับบอร์ด ET-BASE AVR ATmega64/128 r3โดยเสียบบอร์ด ET-AVR ISP เข้าที่ พอร์ต AVR ISP ซึ่งมีการจัดเรียงขาสัญญาณดังรูป



| ตำแหน่งขา | ชื่อสัญญาณ |
|-----------|--------------|
| 1 | MOSI |
| 2 | VCC |
| 3 | ไม่ได้ใช้งาน |
| 4,6,8,10 | GND |
| 5 | RESET |
| 7 | SCK |
| 9 | MISO |

การเชื่อมต่ออุปกรณ์สำหรับโปรแกรม Hex File

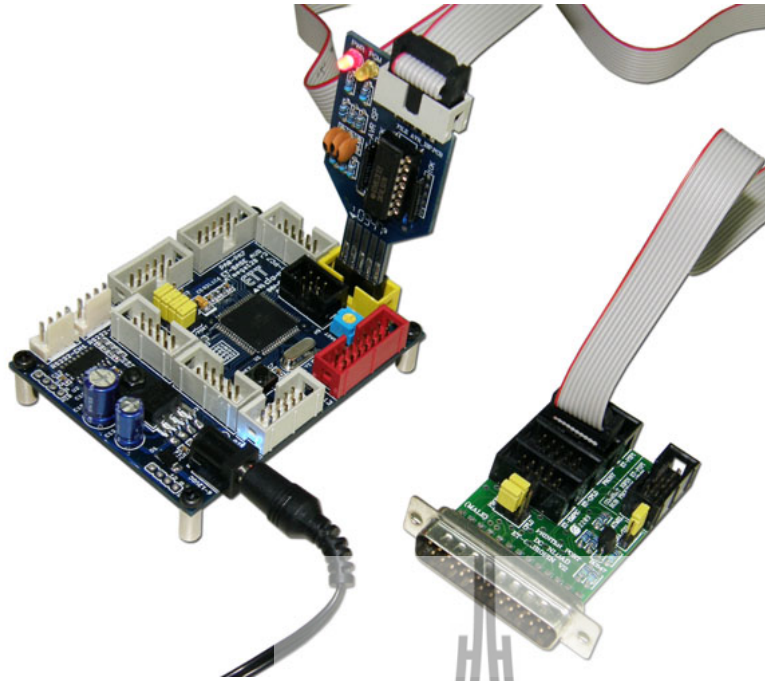
การโปรแกรมโค้ด (Hex File) ให้กับ AVR MCU ต้องใช้งานร่วมกับ ET-CAB10PIN และโปรแกรม PonyProg2000 โดยต่อ ET-CAP10PIN เข้ากับพอร์ต Printer พร้อมทั้งเลือก Jumper สำหรับใช้งานกับโปรแกรม PonyProg2000 แล้วต่อสาย Download ที่ขั้วต่อ AVR ISP Download ของบอร์ด พร้อมทั้งจ่ายไฟเข้าบอร์ดให้เรียบร้อย ถ้ามีการต่ออุปกรณ์ภายนอกที่พอร์ต PB ให้ปลดออกก่อน โดยการเชื่อมต่อจะมีลักษณะดังรูปต่อไปนี้



(ซ้าย) ET-CAP10P V2.0

(ขวา) ET-CAP10P V1.0

รูปแสดง การเลือก Jumper และการต่อสาย Download ของ ET-CAP10P เพื่อใช้กับ AVR



รูปแสดงการต่อ ET-AVR ISP เข้ากับ ET-BASE AVR ATmega64/128 r3 โดยการต่อบอร์ดทั้งสองเข้าด้วยกันนั้นจะให้สังเกตที่ตำแหน่งขา 1 จะต้องตรงกัน

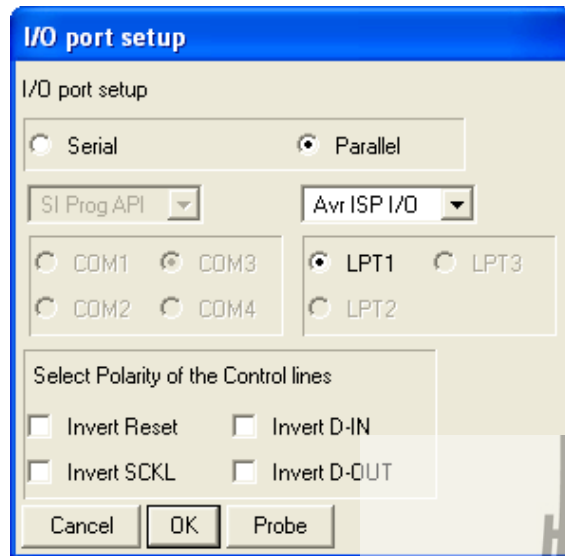
การ Program ให้ Board ET-BASE AVR ATmega64/128 r3 ด้วยโปรแกรม PonyProg2000

โปรแกรม PonyProg2000 เป็นโปรแกรม Download ข้อมูลแบบ HEX File ให้กับ CPU ตระกูล AVR โดยใช้วิธีการแบบ Serial Programming ซึ่งสามารถใช้งานกับบอร์ดตระกูล AVR ของ อีทีที ได้เป็นอย่างดี ซึ่งวิธีการใช้งานโปรแกรมโดยทั่วไปนั้น สามารถศึกษาได้จาก Help ของโปรแกรมได้เอง โดยในที่นี้จะขอแนะนำให้ทราบถึงวิธีการ Setup โปรแกรม PonyProg2000 เพื่อใช้งานกับบอร์ดตระกูล AVR ของ อีทีที ซึ่งสามารถใช้งานได้กับบอร์ดตระกูล AVR ทุกรุ่นของ อีทีที

สำหรับกรณีที่ใช้ CPU ตระกูล AVR เบอร์ ATmega64/128 นั้น จะมีข้อควรระวังอยู่อย่างหนึ่ง เนื่องจากโครงสร้างภายในของ ATmega64/128 นั้นจะมี Fuse Bit สำหรับกำหนดเงื่อนไขการทำงานของ CPU รวมอยู่ด้วยหลายบิต ซึ่ง Fuse Bit ต่างๆเหล่านี้ บางบิตจะมีผลต่อการ Download แบบ Serial Programming ด้วย เนื่องจากถ้าเลือกกำหนดคุณสมบัติของ Fuse Bit ไม่ถูกต้องอาจทำให้ไม่สามารถส่งโปรแกรม CPU ตัวนั้นด้วยวิธีการ Serial Programming ได้อีก นอกจากนี้จะนำ CPU ตัวนั้นไปแก้ไข Fuse Bit ด้วยเครื่องโปรแกรมแบบ Parallel ให้ได้ค่าที่ถูกต้องเสียก่อน

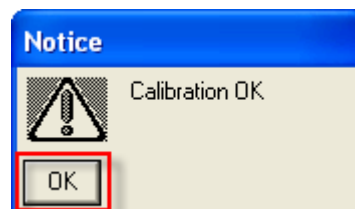
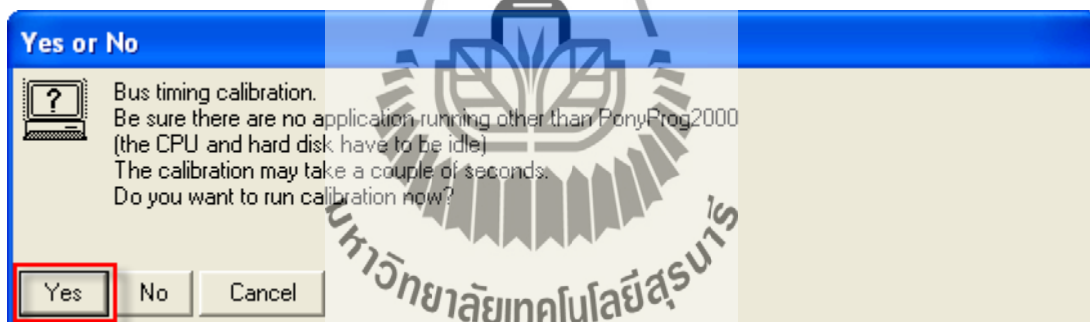
โดยในการสั่งโปรแกรม CPU ตระกูล AVR ที่ใช้งานกับบอร์ดของ อีทีที นั้น ถ้าใช้การโปรแกรมด้วยโปรแกรมของ “PonyProg2000” จะต้องกำหนด Option ของโปรแกรมเพื่อให้สามารถใช้งานกับบอร์ดของ อีทีที ดังนี้

1. กำหนด Setup → Interface Setup... เป็นดังนี้



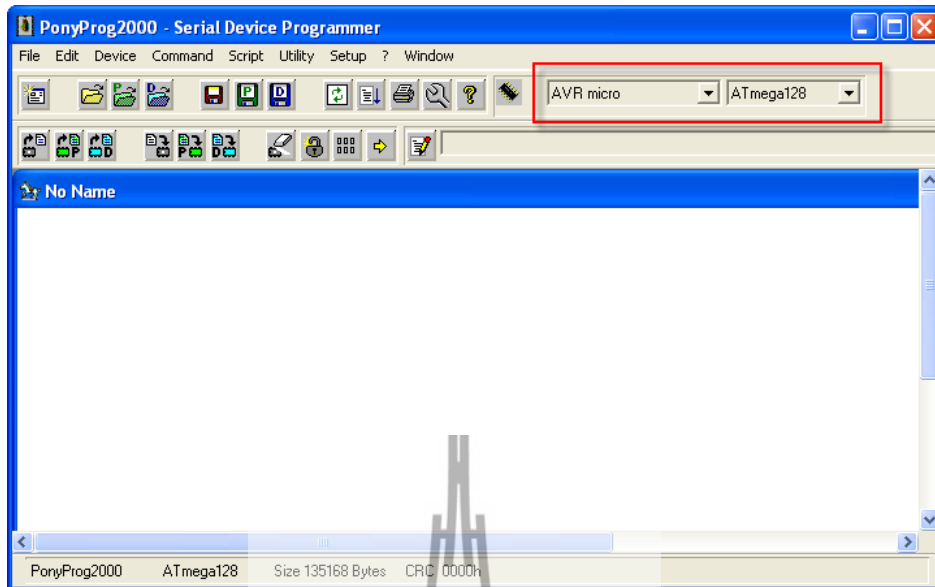
- ให้เลือก I/O Port เป็น Parallel และเลือกรูปแบบการโปรแกรมเป็น Avr ISP I/O
- ให้เลือก Printer Port ตามที่ต่อจริง เช่น LPT1 ในกรณีที่ใช้กับ Printer Port LPT1
- ส่วนของ Polarity Control Line ไม่ต้องเลือก
- การ Setup นี้ทำเพียงครั้งเดียวตอนเริ่มใช้งานโปรแกรมในครั้งแรกเท่านั้น

2. สั่งให้โปรแกรม PonyProg2000 ทำการคำนวณหาค่าความเร็วที่เหมาะสมสำหรับการส่งสัญญาณไปโปรแกรม CPU โดยเลือกจาก Setup → Calibration

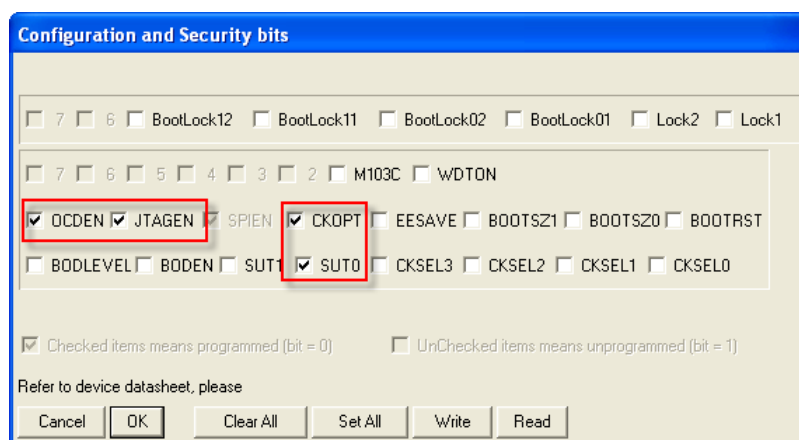


- การสั่ง Calibration จะกระทำเพียงครั้งเดียวในตอนเรียกใช้งานโปรแกรมครั้งแรกเท่านั้น

3. เลือกกำหนดเบอร์ CPU จาก Device → AVR Micro → Atmega64 หรือ ATmega128



4. เลือกกำหนด Command → Security and Configuration Bits โดยถ้าเป็น AVR เบอร์อื่นๆ สามารถกำหนดการทำงานของ Fuse Bit ได้ตามต้องการ โดยสามารถศึกษารายละเอียดของ Fuse Bit ต่างๆได้จาก Data Sheet ของ CPU ที่ใช้ได้เอง แต่ในกรณีที่ใช้งานกับ Atmega64/128 นั้นต้องระมัดระวังในการเลือกกำหนด Fuse Bit ให้ถูกต้องด้วย ซึ่งถ้ากำหนดผิดพลาดส่งผลให้ไม่สามารถสั่งโปรแกรม CPU ด้วยวิธีการ Serial Programming ได้อีก เมื่อเลือกดังรูปแล้วกดปุ่ม Write (ขั้นตอนนี้ทำเพียงครั้งเดียวเท่านั้น ครั้งต่อไปก็สามารถข้ามไปได้เลย ในกรณีที่ใช้บอร์ด ET-BASE AVR ATmega64/128 r3 ไม่จำเป็นต้องทำขั้นตอนนี้เพราะทางบริษัทได้ตั้งค่าไว้เรียบร้อยแล้ว ยกเว้นผู้ใช้งานต้องการเปลี่ยนค่า)



รูปแสดง การเลือกกำหนด Fuse Bit เพื่อใช้กับ CPU เบอร์ ATmega64/128

ความหมายของ Fuse Bit ต่างๆ ของ ATmega64/128

- ในกรณีที่เลือก [✓] ที่หน้า Fuse Bit ตัวใด หมายถึงการกำหนดให้ Fuse Bit นั้นๆ มีค่าเป็น "0" หรือการสั่งโปรแกรม Fuse Bit นั้นๆ
- ในกรณีที่ไมเลือก [✓] ที่หน้า Fuse Bit ตัวใด หมายถึงการกำหนดให้ Fuse Bit นั้นๆ มีค่าเป็น "1" หรือสั่งไม่โปรแกรม Fuse Bit นั้นๆ

ความหมายของ Fuse Bit ของ ATmega64/128 ที่มีผลต่อ Serial Programming

- SPIEN เป็น Serial Programming Enable Bit ซึ่งจะต้องสั่งโปรแกรม Fuse Bit นี้ไว้เสมอ เพื่อให้สามารถสั่ง Download โปรแกรมให้กับ CPU ด้วยวิธีการ In-System Serial Programming ได้ ซึ่งตามปกติแล้ว Fuse Bit นี้จะถูกสั่งโปรแกรมมาจากโรงงานอยู่แล้ว และไม่สามารถสั่งลบหรือแก้ไข Fuse Bit นี้ได้ด้วยโหมด Serial Programming แต่ถ้ามีการนำ CPU ไปโปรแกรมด้วยเครื่องแบบ Parallel Programming จะต้องไม่ลืมสั่งโปรแกรม Fuse Bit นี้ไว้ด้วยเสมอทุกครั้ง
- OCDEN และ JTAGEN ทั้งสองบิตนี้จะใช้ในกรณีที่ต้องการ Debug การทำงานของ MCU และโปรแกรมผ่านทาง JTAG Interface ซึ่งต้องร่วมกับ AVR JTAG Debugger ซึ่งถ้าไม่ได้ใช้งานก็ไม่จำเป็นต้องเลือกทั้งสองบิตนี้
- CKOPT เป็น Oscillator Option Bit ถ้าสั่งโปรแกรม Fuse Bit นี้จะเป็นการกำหนดให้ CPU ทำงานที่ย่านความถี่ 16MHz แต่ถ้าไม่ได้สั่งโปรแกรม Fuse Bit นี้จะเป็นการกำหนดให้ CPU ทำงานที่ย่านความถี่เป็น 8MHz ซึ่งถ้าใช้กับบอร์ดมาตรฐานของอีทีที จะใช้ XTAL เป็นแหล่งกำเนิดความถี่ ดังนั้นควรสั่งโปรแกรมค่า Fuse Bit นี้ไว้ เพื่อให้ CPU สามารถทำงานได้ที่ย่านความถี่ของ XTAL ตั้งแต่ 1.0MHz-16.0MHz
- CKSEL3...0 เป็น Select Clock Source Bit ใช้ร่วมกันสำหรับเลือกแหล่งกำเนิดและย่านของความถี่ที่จะใช้กับ CPU ซึ่งในกรณีใช้งานกับบอร์ดมาตรฐานของอีทีที ต้องเลือกเป็น External Crystal ค่า 1.0 MHz - 16.0 MHz ซึ่งถ้าเลือกเป็นอย่างอื่นจะทำให้การทำงานของโปรแกรมผิดพลาด **และที่สำคัญถ้าเลือกแหล่งกำเนิดความถี่ผิด เช่น เลือกเป็น External Clock หรือ External RC Oscillator จะทำให้ CPU ไม่สามารถทำงานได้ เนื่องจากไม่มีการต่อสัญญาณนาฬิกาจากภายนอกไว้ให้ และจะทำให้ไม่สามารถสั่งโปรแกรม CPU ตัวนั้นด้วยวิธีการแบบ Serial Programming ได้อีก** จนกว่าจะมี

การนำ CPU ไปแก้ไขค่า Fuse Bit เพื่อเลือกแหล่งกำเนิดสัญญาณนาฬิกาเป็น External Crystal ให้ถูกต้องเสียก่อน

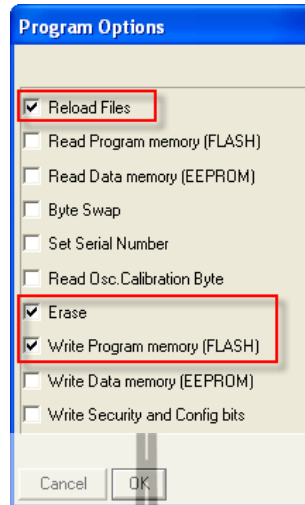
| แหล่งกำเนิดสัญญาณนาฬิกาของ AVR Atmega128 | การกำหนด Fuse Bit ของ CKSEL[3...0] (0=Program,1=Un-Program) |
|---|--|
| External Crystal/Ceramic Resonator | 1111-1010 |
| External Low Frequency Crystal | 1001 |
| External RC Oscillator | 1000-0101 |
| Calibrated Internal RC Oscillator | 0100-0001 |
| External Clock | 0000 |

ตารางแสดง การเลือกแหล่งกำเนิดความถี่จาก Fuse Bit CKSEL [3...0]

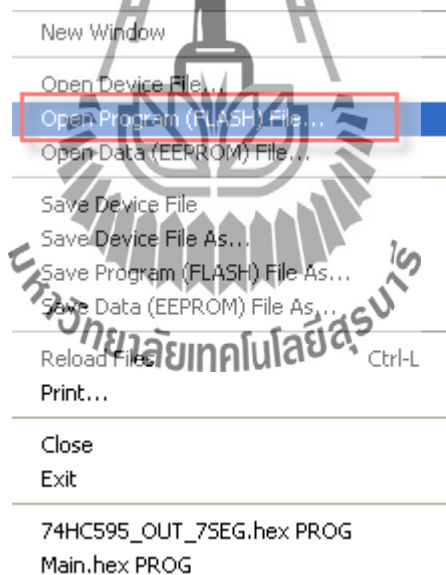
หมายเหตุ

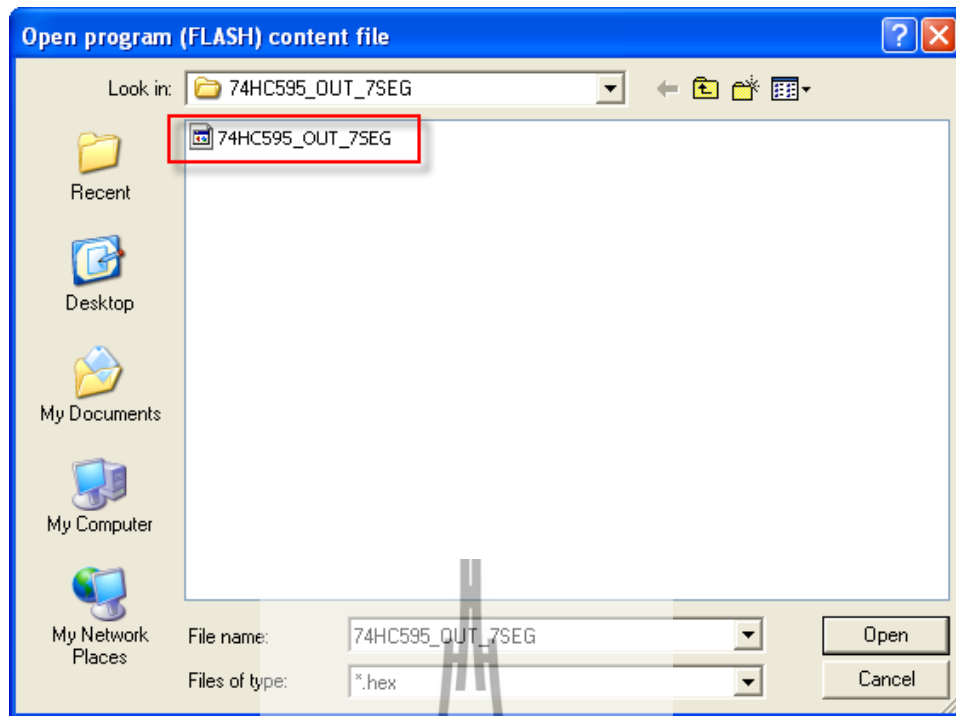
- ค่า 1 หมายถึง การสั่งไม่โปรแกรม Fuse Bit นั้นๆ โดยไม่ต้องใส่เครื่องหมาย [√] หน้า Fuse Bit
- ค่า 0 หมายถึง การสั่งโปรแกรม Fuse Bit นั้นๆ โดยการเลือกเครื่องหมาย [√] หน้าชื่อ Fuse Bit
- ควรสั่งโปรแกรม Fuse Bit ของ CKOPT เพื่อให้ใช้งานที่ย่านความถี่ 1.0MHz-16.00MHz
- ห้ามสั่งโปรแกรม Fuse Bit ของ CKSEL[3..0] เพราะจะทำให้การทำงานไม่ถูกต้อง ตัวอย่างเช่น ถ้าเลือกสั่งโปรแกรม Fuse Bit ของ CKSEL[3..0] ให้มีค่าเป็น 0 ทั้งหมด ซึ่งหลังจากโปรแกรม PonyProg2000 ทำการเขียนค่า Fuse Bit นี้ให้กับ CPU เรียบร้อยแล้ว จะทำให้ CPU ไม่สามารถใช้งานได้กับบอร์ดของอีทีที อีก และจะไม่สามารถสั่งโปรแกรมแก้ไขค่า Fuse Bit ใหม่ให้กับ CPU ด้วยวิธีการแบบ Serial Programming ได้อีก เนื่องจาก CPU ไม่สามารถทำงานได้อีก เพราะว่าการสั่งโปรแกรม Fuse Bit ของ CKSEL[3..0] ให้เป็น 0 ทั้งหมด จะเป็นการสั่งให้ CPU ทำงานด้วยความถี่ของสัญญาณนาฬิกาจากภายนอก (External Clock) ซึ่งจะทำให้วงจรกำเนิดความถี่ของ External Crystal หยุดทำงาน โดย CPU จะรอรับสัญญาณนาฬิกาจากภายนอกเพียงอย่างเดียวเท่านั้น แต่บอร์ดของอีทีที จะใช้สัญญาณนาฬิกาจาก วงจร Crystal (External Crystal) เท่านั้น ดังนั้นเมื่อ CPU ไม่สามารถเริ่มต้นทำงานได้ ก็จะทำให้เราไม่สามารถสั่งโปรแกรมแก้ไขค่า Fuse Bit ที่ถูกต้องให้กับ CPU ด้วยวิธีการแบบ Serial Programming ได้อีก ซึ่งจะต้องนำ CPU ตัวนั้น ไปทำการแก้ไขค่า Fuse Bit ด้วยเครื่องโปรแกรมแบบ Parallel เสียก่อนจึงจะสามารถนำมาใช้งานกับวิธีการโปรแกรมแบบ Serial Programming ได้เหมือนเดิม
- ตำแหน่ง Fuse Bit ของ Lock[2..1] สามารถกำหนดได้ตามต้องการ

5. เลือกกำหนด Command → Program Option เป็นดังนี้

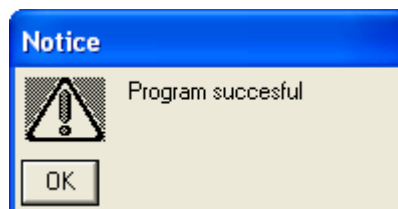
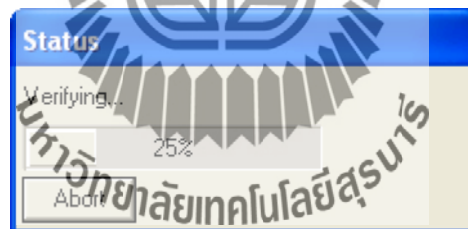


6. สั่งเปิดไฟล์สำหรับที่จะใช้โปรแกรมให้กับ CPU โดยเลือกจาก File → Open Program (FLASH) File... → พร้อมทั้งระบุชื่อและที่อยู่ของ HEX File ที่จะใช้โปรแกรมให้เรียบร้อย





7. สั่งเริ่มต้นโปรแกรมข้อมูลให้กับ CPU โดยเลือก Command → Program จากนั้นโปรแกรมจะเริ่มทำงานตามคำสั่งที่เราเลือกกำหนดไว้ในข้อ 5 คือ Load File → Erase → Write Program memory (FLASH) ตามลำดับ ซึ่งให้รอจนการทำงานของโปรแกรมเสร็จสมบูรณ์



ซึ่งหลังจากการโปรแกรมเสร็จเรียบร้อยแล้ว CPU จะเริ่มต้นทำงานตามข้อมูลในโปรแกรมที่สั่ง Download ให้ทันที

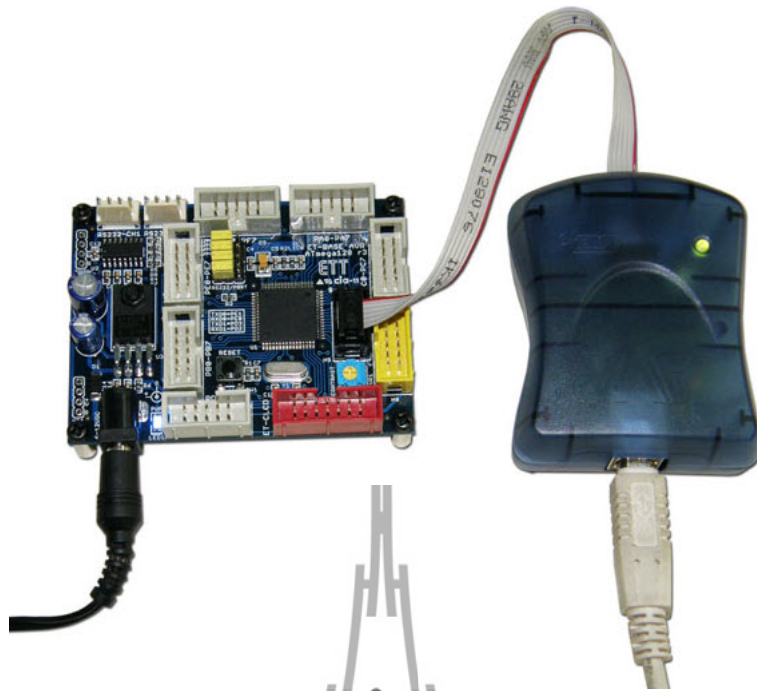
การตรวจสอบเบื้องต้นเมื่อไม่สามารถดาวน์โหลดโปรแกรมได้

ถ้าเกิดการ Error ในขั้นตอนของการโปรแกรมให้ตรวจสอบปัญหาดังนี้

- อ่านคู่มือการใช้งานบอร์ด และคู่มือวิธีการ Download โปรแกรม AVR ด้วย PonyProg2000 ให้ละเอียด
- ตรวจสอบการเชื่อมต่อของสายสัญญาณต่างๆ และ ในการ Download โปรแกรมโดยใช้ PonyProg2000 นั้น จะต้องใช้งานร่วมกับชุด Cable Download รุ่น ET-CAP10PIN ของ ETT ด้วย ซึ่งต้องมีการกำหนด JUMPER ให้เป็น PonyProg ให้ถูกต้องด้วย (รายละเอียดหน้า 9)
- ตรวจสอบการจ่ายไฟเลี้ยงให้กับบอร์ด
- ตรวจสอบการตั้งค่าต่างๆของโปรแกรมสำหรับดาวน์โหลด PonyProg2000
- ตรวจสอบว่ามีการนำสัญญาณจากพอร์ต PORT-PB ของ CPU ไปต่อไว้กับอุปกรณ์ภายนอก ในขณะที่สั่ง Download หรือไม่ ตัวอย่างเช่น ต่อกับ LED หรือ นำสัญญาณจากพอร์ต PB ไปต่อไว้กับวงจรอื่นๆในขณะที่สั่ง Download ข้อมูลอยู่



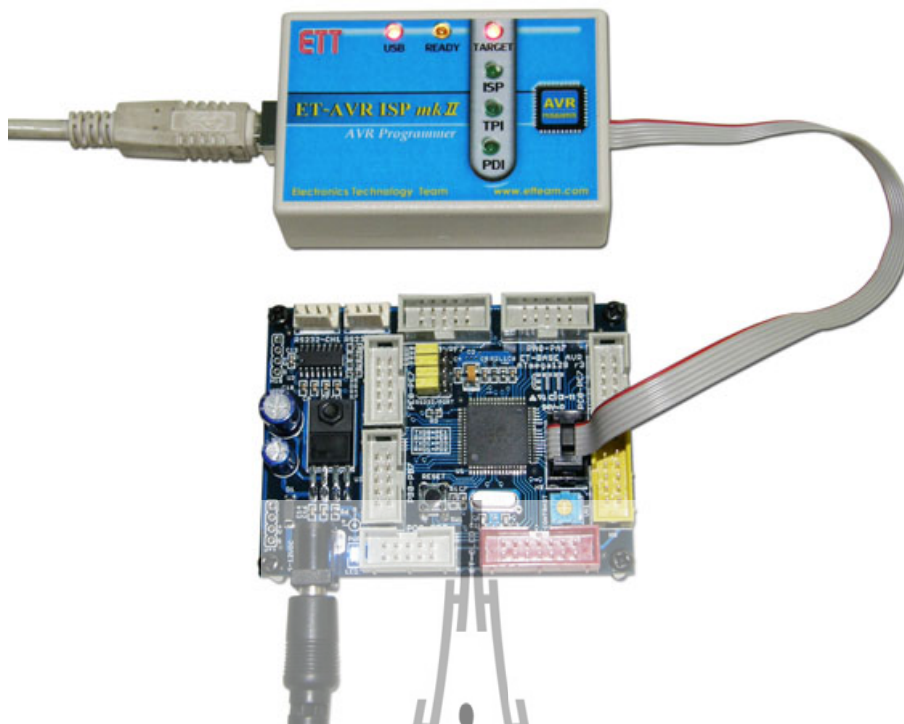
การใช้ร่วมกับเครื่องโปรแกรมอื่นๆ



ตัวอย่างการใช้งานร่วมกับ AVRISP mkII ของบริษัท ATMEL

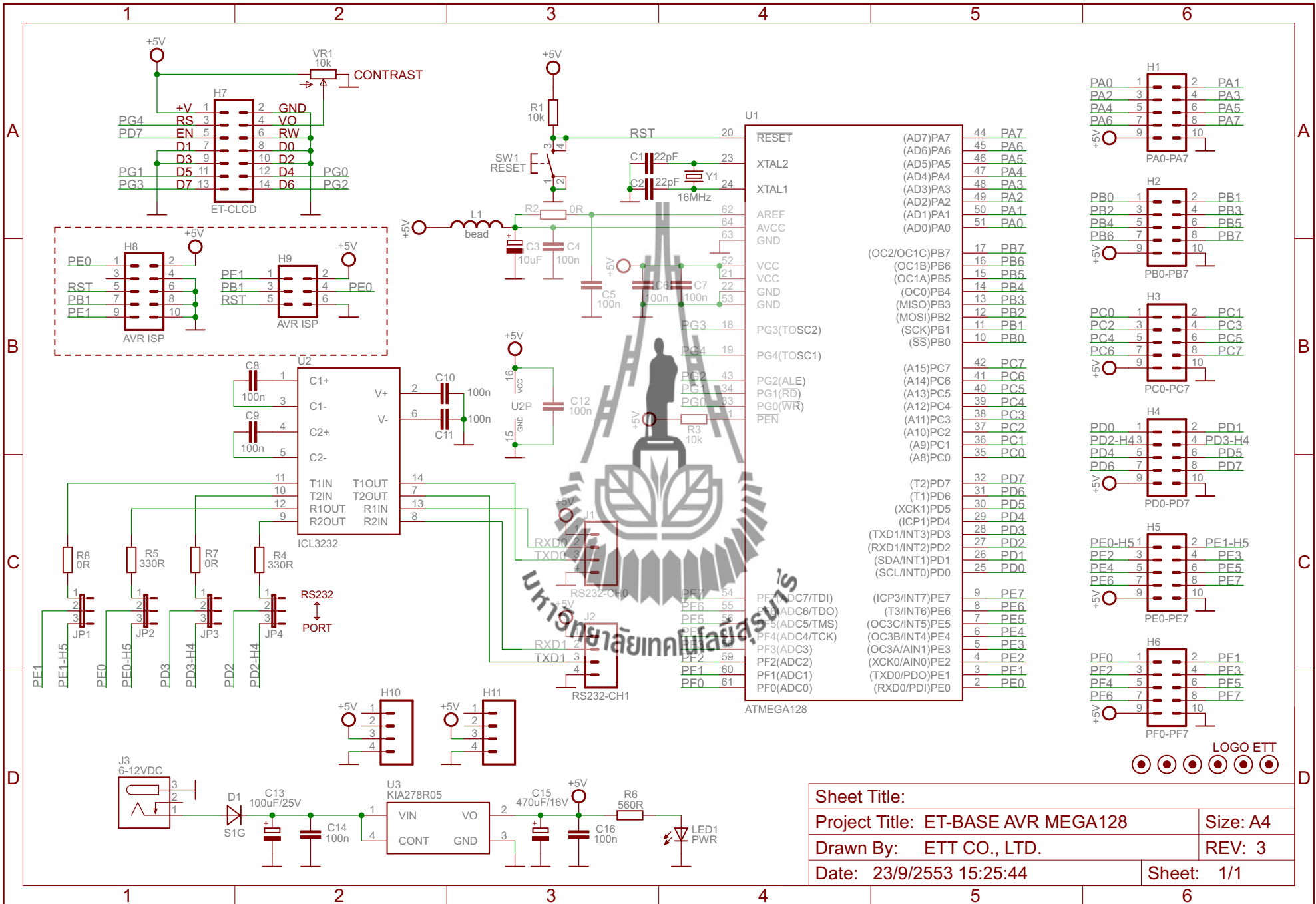


การใช้งานร่วมกับ ET-AVR ISP USB V1.0 ของบริษัท ETT



การใช้งานร่วมกับ ET-AVR ISP mkII ของบริษัท ETT





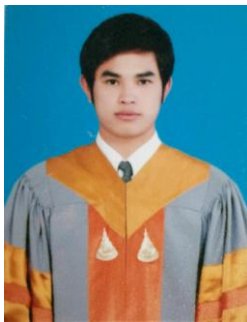
Sheet Title: ET-BASE AVR MEGA128
Project Title: ET-BASE AVR MEGA128
Drawn By: ETT CO., LTD.
Date: 23/9/2553 15:25:44

Size: A4
REV: 3
Sheet: 1/1

ประวัติผู้เขียน



นางสาวเมวิกา ขวัดหนา เกิดเมื่อวันที่ 1 เมษายน พ.ศ. 2533
 ภูมิลำเนาอยู่ที่ ตำบลในเมือง อำเภอเมืองนครราชสีมา จังหวัดนครราชสีมา
 สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนมารีย์วิทยา อำเภอเมือง
 นครราชสีมา จังหวัดนครราชสีมา เมื่อปี พ.ศ. 2550 ปัจจุบันเป็นนักศึกษาชั้น
 ปีที่ 4 สาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์
 มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา



นายบารมี บุญขุบล เกิดเมื่อวันที่ 22 สิงหาคม พ.ศ.2533
 ภูมิลำเนาอยู่ที่ ตำบลบางตลาด อำเภอปากเกร็ด จังหวัดนนทบุรี
 สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนเบญจมราชานุสรณ์
 อำเภอเมือง จังหวัดนนทบุรี เมื่อปีพ.ศ.2550 ปัจจุบันเป็นนักศึกษาชั้นปี
 ที่ 4 สาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์
 มหาวิทยาลัยเทคโนโลยีสุรนารี จังหวัดนครราชสีมา



นางสาวฉันทิตา คงรอด เกิดเมื่อวันที่ 6 มีนาคม พ.ศ.2533
 ภูมิลำเนาอยู่ที่ ตำบลบ้านเมือง อำเภอเมือง จังหวัดพิษณุโลก
 สำเร็จการศึกษาระดับมัธยมปลายจากโรงเรียนเฉลิมขวัญสตรีพิษณุโลก
 อำเภอเมือง จ.พิษณุโลก เมื่อปีพ.ศ.2550 ปัจจุบันเป็นนักศึกษาชั้นปีที่ 4
 สาขาวิศวกรรมโทรคมนาคม สำนักวิชาวิศวกรรมศาสตร์ มหาวิทยาลัย
 เทคโนโลยีสุรนารี จังหวัดนครราชสีมา