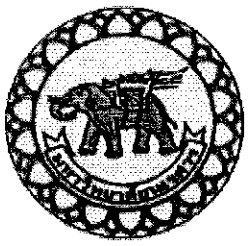


เครื่องวัดการเลื่อนของดิน

SOIL EROSION MEASUREMENT

นายทวีชาติ ธนกุลปริยา รหัส 48380341

นายธนากร วรินทรา รหัส 48364401

ห้องสมุดคณะวิศวกรรมศาสตร์
วันที่รับ 25 / มี.ค. 2553 /
เลขทะเบียน ร 5004214
เลขเรียกหนังสือ วิ
มหาวิทยาลัยนเรศวร
2551

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2551

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	เครื่องวัดการเลื่อนของดิน			
ผู้ดำเนินโครงงาน	นายทวีชาติ	ธนกุลปริยา	รหัส	48380341
	นายธนากร	วรินทรา	รหัส	48364401
อาจารย์ที่ปรึกษา	ผศ.ดร. ยงยุทธ	ชนบดีเฉลิมรุ่ง		
อาจารย์ที่ปรึกษาร่วม	ดร.ศุภวรรณ	พลพิทักษ์ชัย		
สาขาวิชา	วิศวกรรมไฟฟ้า			
ภาควิชา	วิศวกรมไฟฟ้าและคอมพิวเตอร์			
ปีการศึกษา	2551			

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง)

.....กรรมการ

(ดร.อัครพันธ์ วงศ์กังแห)

.....กรรมการ

(อาจารย์แสงชัย มังกรทอง)

หัวข้อโครงงาน	เครื่องวัดการเลื่อนของดิน
ผู้ดำเนินโครงงาน	นายทวีชาติ ธนกุลปริยา รหัส 48380341
	นายธนากร วรินทรา รหัส 48364401
อาจารย์ที่ปรึกษา	ผศ.ดร.ยงยุทธ ชนบดีเฉลิมรุ่ง
สาขาวิชา	วิศวกรรมไฟฟ้า
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2551

บทคัดย่อ

โครงงานชิ้นนี้เกิดขึ้นเนื่องจาก ความต้องการระบบที่สามารถทำการเตือนภัยล่วงหน้าก่อนเกิดดินถล่มแบบฉับพลันได้ ซึ่งในช่วงเวลาปัจจุบันนั้น สภาพอากาศได้แปรเปลี่ยนไปจากเดิมมาก เกิดฝนตกอย่างหนักในหลายพื้นที่ ทำให้มีการสูญเสียทั้งชีวิตและทรัพย์สินที่มีสาเหตุมาจากน้ำท่วมและดินถล่ม จึงเกิดแนวคิดที่จะสร้างระบบเตือนภัยโดยผ่านการเก็บข้อมูลและวิเคราะห์ข้อมูล เพื่อทำการส่งข้อมูลที่ได้รับไปทำการแจ้งเตือนภัยให้กับประชาชนที่อาศัยในบริเวณพื้นที่ที่อันตรายจากภัยพิบัติโดยใช้อุปกรณ์ที่ได้รับการประยุกต์ใช้งานในแบบต่าง ๆ ดังนั้นจึงได้มีการนำเครื่องตรวจวัด เพื่อตรวจสอบการเลื่อนตัวของดินมาใช้ในการเก็บข้อมูลเพื่อส่งข้อมูลไปยังศูนย์รับข้อมูลที่ได้จากอินพุท เพื่อเก็บข้อมูลและนำมาวิเคราะห์และนำข้อมูลที่ได้ทำการแจ้งเตือนภัยธรรมชาติ ต่อผู้ที่เสี่ยงภัยและบริเวณใกล้เคียง

Project title Soil Erosion Measurement

Name Mr. Taweechad Tanakunpariyar ID. 48380341

Mr. Tanakon Warinta ID. 48364401

Project advisor Asst. Prof. Dr. Yongyut Chonbodeechalearmroong

Major Electrical Engineering

Department Electrical and Computer Engineering

Acadamic year 2008

ABSTRACT

This project is created because of the requirements of a system that can warn of danger in advance before a landslide will happen. Now the weather is very changeable, it rains hard in many areas. The floods and landslides can cause loss of life and assets that bring about the idea to build a warning system by storing, analyzing and sending data to warn resident people in dangerous areas against the disaster. Thus this leads to our project to measure soil movement, send data via radio frequency, and analyze the data for giving information and warning people in the risky area and the surrounding.

กิตติกรรมประกาศ

ขอขอบพระคุณ ผศ.ดร. ยงยุทธ ชนบดีเฉลิมรุ่ง อาจารย์ที่ปรึกษาที่คอยให้คำปรึกษาและให้ความช่วยเหลือตลอดจนคำแนะนำต่างๆในการทำโครงงานชิ้นนี้ สุดท้ายต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ พี่ๆ ทุกคนที่ยังไม่ได้เอ่ยนามที่ให้คำแนะนำและให้การสนับสนุน ผู้จัดทำโครงงานให้สามารถทำโครงงานชิ้นนี้จนสำเร็จลุล่วงไปได้ด้วยดี

นายทวีชาติ ธนกุลปริยา รหัส 48380341

นายธนากร วรินทรา รหัส 48364401

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันเหตุการณ์ " น้ำท่วม-ดินถล่ม"ในภาคเหนือ ก็ได้สร้างความสูญเสียให้กับคนไทย โดยเฉพาะในภาคเหนืออีกครั้ง จากภาวะน้ำป่าที่พัดพาดิน โคลนรวมถึงซากไม้หรือซุงที่กำลังถกเถียงกันในขณะนี้มาทับถมบ้านเรือนประชาชนในจังหวัดอุตรดิตถ์ แพร่ สุโขทัย และจังหวัดพิษณุโลก โดยเฉพาะที่ จังหวัดอุตรดิตถ์ ในพื้นที่อำเภอลับแล , ท่าปลาและอำเภอเมืองที่ถูกน้ำท่วมอย่างที่ไม่เคยปรากฏในประวัติศาสตร์มาก่อน

สร้างความเสียหายให้กับชีวิตความเป็นอยู่อย่างมหาศาล บางรายบ้านถูกน้ำพัดหายไปทั้งหลัง คนในครอบครัวถูกน้ำบวกกับโคลนถาถมร่างมิดไปกลับตา ปัญหาที่เกิดขึ้น นอกจากจะสร้างความเดือดร้อนในแง่ของเศรษฐกิจแล้ว ยังส่งผลต่อสภาพจิตใจอย่างมากด้วย เพราะหลายรายบ้าน สวน และไร่นา ได้รับความเสียหาย

ดังนั้นทางคณะผู้จัดทำ จึงได้ทำเครื่องวัดการเลื่อนของดิน เพื่อวัดการเลื่อนของดินบนเชิงเขา และแจ้งเตือนภัย ให้ประชาชนในพื้นที่เสี่ยงภัยและละแวกใกล้เคียงได้รับทราบ ก่อนที่จะเกิดดินถล่ม

โครงงานนี้มีอุปกรณ์อิเล็กทรอนิกและอุปกรณ์สื่อสาร คือส่วนที่เป็นเครื่องวัดการเลื่อนตัวของดินซึ่งใช้หลักการเปรียบเทียบการเปลี่ยนแปลงของตัวต้านแบบละเอียด เพื่อตรวจสอบการเลื่อนของดิน และส่งค่าไปยังเครื่องรับสัญญาณเพื่อนำข้อมูลส่งไปยังเครื่องรับสัญญาณเตือนภัยอื่นๆ ต่อไป

1.2 วัตถุประสงค์ของโครงการ

1. เพื่อศึกษาการเลื่อนตัวของดิน
2. เพื่อศึกษาเครื่องมือวัด โดยใช่หลักการการเปลี่ยนแปลงของตัวต้านทานปรับค่าได้
3. เพื่อศึกษาการส่งข้อมูลแบบไร้สาย
4. เพื่อลดความเสียหายที่เกิดจากแผ่นดินถล่ม
5. เพื่อศึกษาและพัฒนาอุปกรณ์ที่วัดการเลื่อนตัวของดิน และเครื่องส่งข้อมูลเตือนภัย

1.3 ขอบเขตของโครงงาน

1. ศึกษาการเลื่อนของดินและการส่งข้อมูลแบบไร้สาย
2. พัฒนาอุปกรณ์ที่ใช้ในการวัดการเลื่อนตัวของดิน โดยใช่หลักการการเปลี่ยนแปลงของตัวต้านทานปรับค่าได้
3. พัฒนาอุปกรณ์แบบไร้สาย โดยการส่งสัญญาณระหว่างเครื่องวัดกับฐานข้อมูลเพื่อใช้ในการประมวลผลและแจ้งเตือนภัยต่อไป

1.4 ขั้นตอนการดำเนินและแผนการดำเนินงานตลอดโครงงาน

รายละเอียด	ปี 2551							ปี 2552			
	มิ.ย	ก.ค	ส.ค	ก.ย	ต.ค	พ.ย	ธ.ค	ม.ค	ก.พ	มี.ค	เม.ย
1.รวบรวมข้อมูล	←	—	—	→							
2.ศึกษาและออกแบบการทำงานของเครื่องวัดและเครื่องส่ง				←	—	—	→				
3.จัดทำอุปกรณ์และทำการทดลองเครื่องวัดและเครื่องส่ง					←	—	—	—	→		
4.จัดทำรายงานและสรุปผลการทำงาน										←	→

1.5 ประโยชน์ที่คาดว่าจะได้รับจากโครงงาน

1. เข้าใจหลักการการเลื่อนตัวของดิน
2. เข้าใจหลักการสื่อสารแบบไร้สาย
3. สามารถนำอุปกรณ์วัดไปติดตั้งในพื้นที่เสี่ยงภัยดินถล่มได้
4. เพื่อแจ้งเตือนการเลื่อนตัวของดินก่อนที่จะเกิดดินถล่มไปยังประชาชนในพื้นที่เสี่ยงภัยและละแวกใกล้เคียง
5. เพื่อเป็นประโยชน์ในการลดความสูญเสียต่อพื้นที่เสี่ยงภัยการถล่มของดิน

1.6 งบประมาณ

1. ถ่ายเอกสารและค่าเข้าเล่มโครงงานฉบับสมบูรณ์	500	บาท
2. ค่าอุปกรณ์ในการทำโครงงาน	9,000	บาท
3. ค่าหนังสือ	300	บาท
4. ค่าพิมพ์เอกสาร	200	บาท
รวมเป็นเงิน (หนึ่งหมื่นบาทถ้วน)	10,000	บาท

หมายเหตุ : ถัวเฉลี่ยทุกรายการ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

จากแนวคิดของคณะผู้จัดในการสร้างเครื่องวัดการเลื่อนของดิน โดยอาศัยหลักการจากการเปลี่ยนของตัวต้านทานปรับค่าได้กับตัว Referent (เป็นตัวปรับค่าได้เช่นกัน) และเชื่อมต่อกับไมโครคอนโทรลเลอร์เพื่อจะนำมาข้อมูลที่ได้จากเครื่องวัดมาแปลงค่าและเปรียบเทียบและส่งไปยังเครื่องรับสัญญาณเตือนภัยอื่นๆ ต่อไป เนื่องจากโครงานนี้เป็นเครื่องการเลื่อนของดินเพื่อเป็นการเตือนภัยจากเหตุดินถล่ม (land slides) จึงของกล่าวถึงทฤษฎีพื้นฐานเกี่ยวกับการเกิดดินถล่มและหลักการทำงานของอุปกรณ์ที่เกี่ยวข้องกับโครงงาน ซึ่งมีดังต่อไปนี้

2.1 ปรากฏการณ์ทางธรรมชาติ

แผ่นดินถล่ม (land slides)

นิยาม

แผ่นดินถล่มเป็นปรากฏการณ์ธรรมชาติของการสึกกร่อนชนิดหนึ่ง ที่ก่อให้เกิดความเสียหายต่อบริเวณพื้นที่ที่เป็นเนินสูงหรือภูเขาที่มีความลาดชันมาก เนื่องจากขาดความสมดุลในการทรงตัวบริเวณดังกล่าว ทำให้เกิดการปรับตัวของพื้นดินต่อแรงดึงดูดของโลกและเกิดการเคลื่อนตัวขององค์ประกอบธรณีวิทยาบริเวณนั้นจากที่สูงลงสู่ที่ต่ำ แผ่นดินถล่มมักเกิดในกรณีที่มีฝนตกหนักมากบริเวณภูเขาและภูเขานั้นอุ้มน้ำไว้จนเกิดการอิ่มตัว จนทำให้เกิดการพังทลาย

ประเภทของแผ่นดินถล่ม

แบ่งตามลักษณะการเคลื่อนตัวได้ 3 ชนิดคือ

1. แผ่นดินถล่มที่เคลื่อนตัวอย่างแผ่นดินถล่มที่เคลื่อนตัวอย่างช้าๆ เรียกว่า Creep เช่น Sacrificial Creep
2. แผ่นดินถล่มที่เคลื่อนตัวอย่างรวดเร็วเรียกว่า Slide หรือ Flow เช่น Sacrificial Slide
3. แผ่นดินถล่มที่เคลื่อนตัวอย่างฉับพลัน เรียกว่า Fall Rock Fall นอกจากนี้ยังสามารถแบ่งออกได้ตามลักษณะของวัสดุที่ล่วงหล่นลงมาได้ 3 ชนิด คือ "แผ่นดินถล่มที่เกิดจากการเคลื่อนตัวของผิวหน้าดินของภูเขา" แผ่นดินถล่มที่เกิดจากการเคลื่อนที่ของวัตถุที่ยังไม่แข็งตัว "แผ่นดินถล่มที่เกิดจากการเคลื่อนตัวของชั้นหิน.

แผ่นดินถล่มในประเทศไทย

แผ่นดินถล่มในประเทศไทย ส่วนใหญ่มักเกิดภายหลังฝนตกหนักมากบริเวณภูเขาซึ่งเป็นต้นน้ำลำธาร บริเวณตอนบนของประเทศ โดยเฉพาะในภาคเหนือและภาคตะวันออกเฉียงเหนือ มีโอกาสเกิดแผ่นดินถล่มเนื่องมาจากพายุหมุนเขตร้อนเคลื่อนผ่านในระหว่างเดือนกรกฎาคมถึงสิงหาคม ในขณะที่ภาคใต้จะเกิดในช่วงฤดูมรสุมตะวันออกเฉียงเหนือ ระหว่างเดือนพฤศจิกายนถึงธันวาคม

ปัจจัยที่ส่งเสริมความรุนแรงของแผ่นดินถล่ม

1. ปริมาณฝนที่ตกบนภูเขา
2. ความลาดชันของภูเขา
3. ความสมบูรณ์ของป่าไม้
4. ลักษณะทางธรณีวิทยาของภูเขา

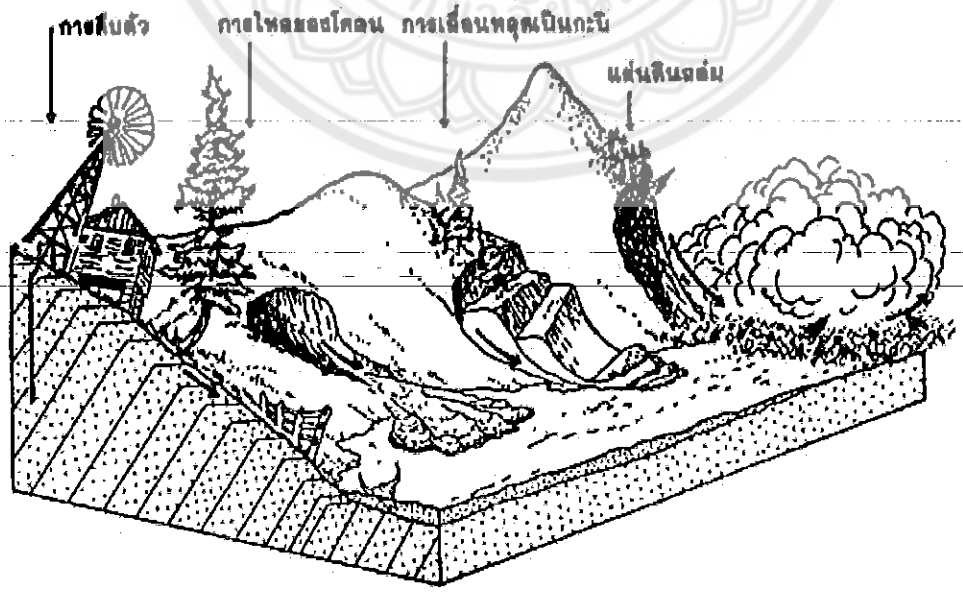
ลำดับเหตุการณ์ของการเกิดแผ่นดินถล่ม

เมื่อฝนตกหนักน้ำซึมลงไปในดินอย่างรวดเร็ว ในขณะที่ดิน อิ่มน้ำ แรงยึดเกาะระหว่างมวลดินจะลดลง ระดับน้ำใต้ผิวดินสูงขึ้นจะทำให้แรงต้านทานการเลื่อนไหล ของดินลดลง เมื่อน้ำใต้ผิวดินมีระดับสูงก็จะไหลภายในช่องว่างของดิน ลงตามความชันของลาดเขา เมื่อมีการเปลี่ยนความชัน ก็จะเกิดเป็นน้ำผุด และเป็นจุดแรกที่มีการเลื่อนไหลของดิน เมื่อเกิดดินเลื่อนไหลแล้วก็จะเกิดต่อเนื่องขึ้นไปตามลาดเขา

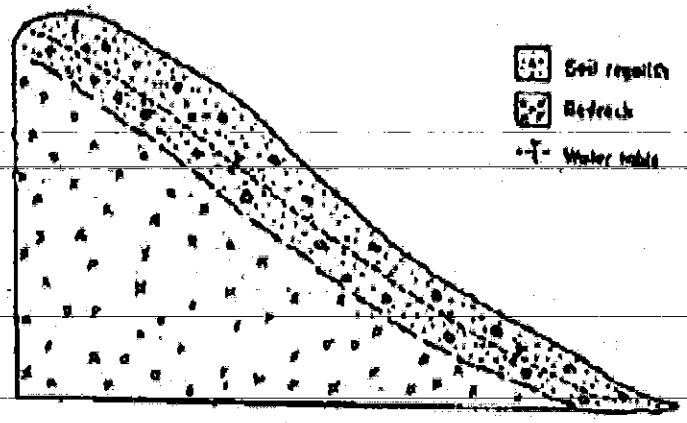
ปัจจัยสำคัญที่เป็นสาเหตุของการเกิดแผ่นดินถล่ม

- ลักษณะของดินที่เกิดจากการผุพังของหินบนลาดเขา
- ลาดเขาที่มีความลาดชันมาก (มากกว่า 30 เปอร์เซ็นต์)
- มีการเปลี่ยนแปลงสภาพป่า

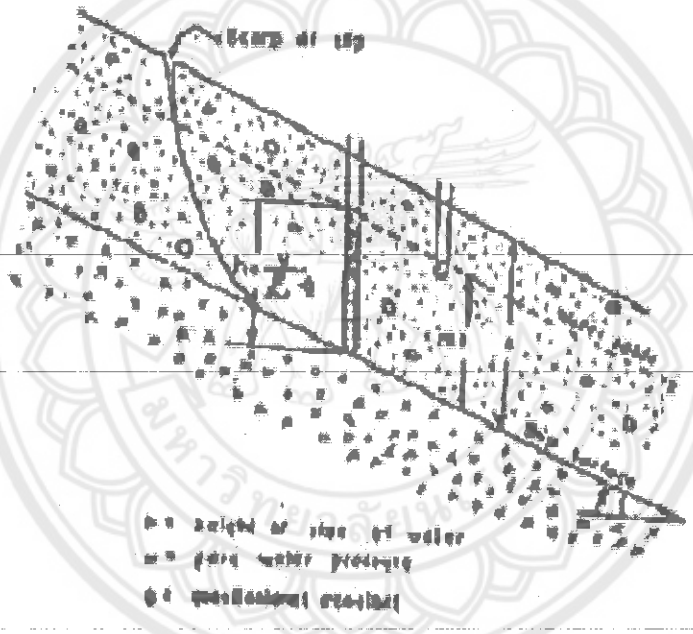
หมายเหตุ : จากปัจจัยที่สำคัญและได้ถูกกระตุ้นโดยปริมาณน้ำฝนที่ตกหนักมาก เป็นสาเหตุที่สำคัญที่ทำให้เกิดภัยพิบัติแผ่นดินถล่ม



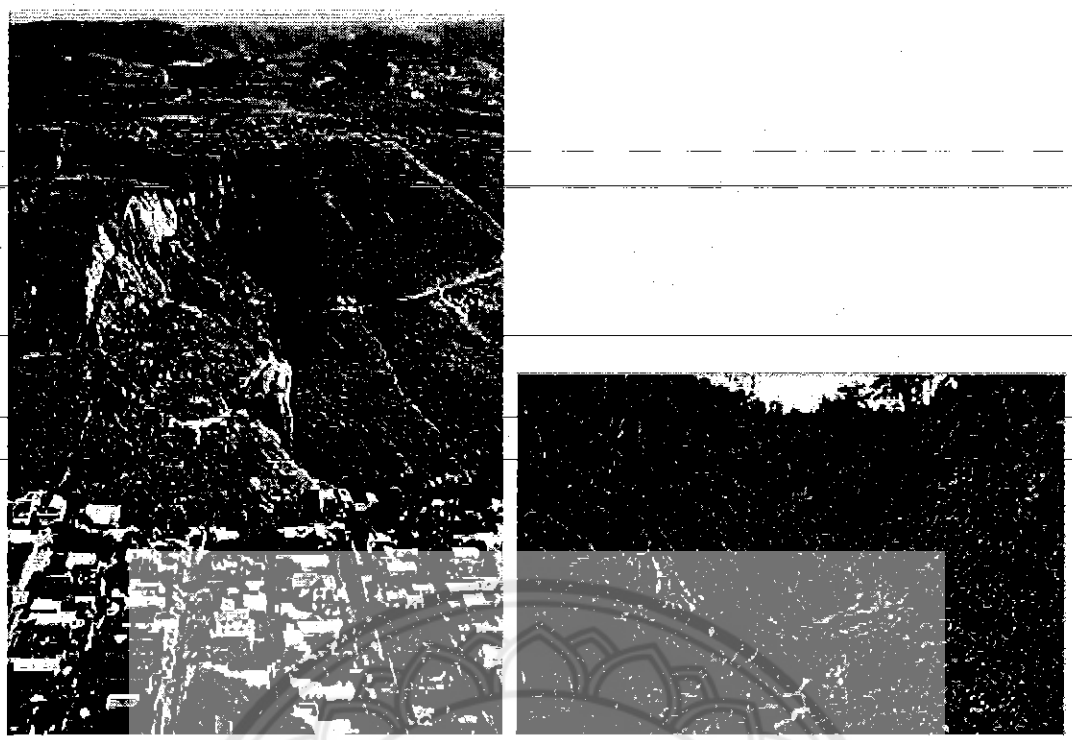
รูป2.1 ลักษณะการเกิดแผ่นดินถล่มและการเคลื่อนตัวของดิน



รูปที่ 2.2 ลักษณะการเลื่อนของดิน



รูปที่ 2.3 ลักษณะการเลื่อนของดิน 2



รูปที่ 2.4 ภาพเหตุการณ์การเกิดแผ่นดินถล่ม

2.2 อุปกรณ์อิเล็กทรอนิกส์ที่เกี่ยวข้อง

ตัวต้านทานชนิดปรับค่าได้ (Variable Value Resistor)

การปรับปุ่มควบคุมระดับความดัง หรือ วอลลุม (Volume) ซึ่งอุปกรณ์ดังกล่าวนี้เป็นตัวอย่างของตัวต้านทานชนิดปรับค่าได้ประเภทหนึ่ง

ตัวต้านทานชนิดเปลี่ยนค่าได้โดยอาศัยกลไก ตัวต้านทานชนิดนี้เปลี่ยนค่าได้โดยอาศัยกลไกมีอยู่ 2 แบบ ได้แก่

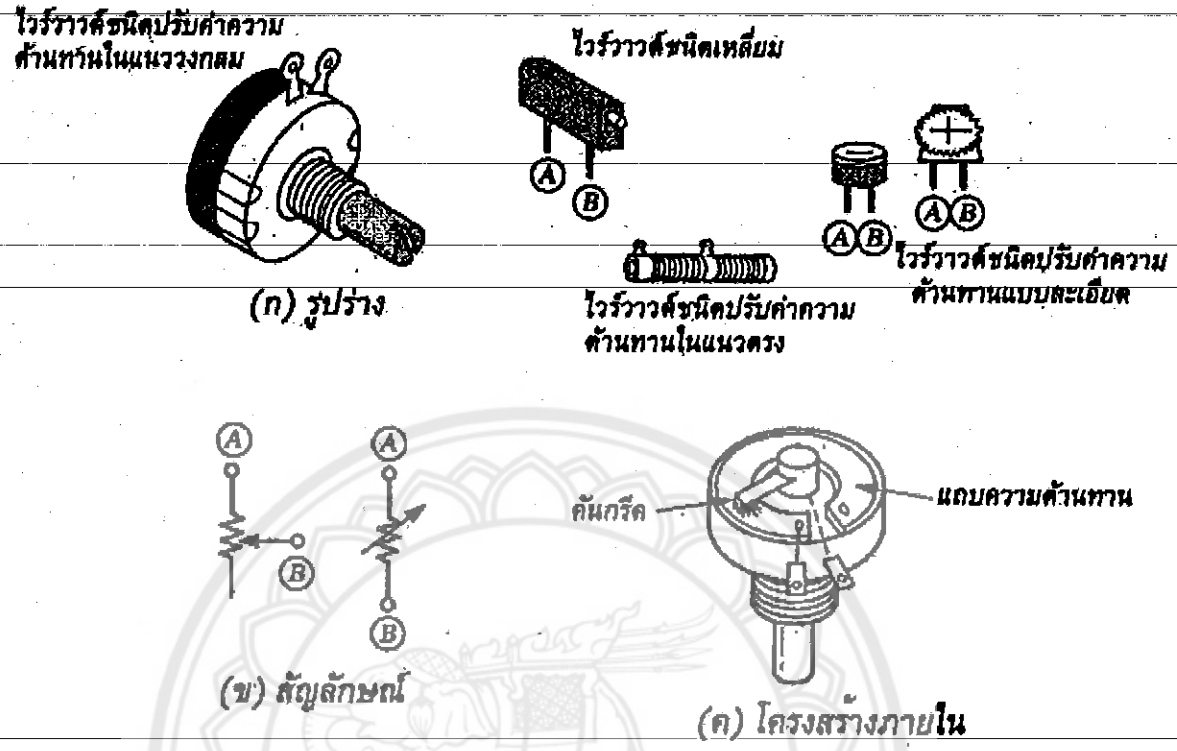
- รีโอสตัส (Rheostat)
- โพเทนชิโอมิเตอร์ (Potentiometer)

รีโอสตัส (2 ขั้ว : A และ B)

รูปลักษณะของรีโอสตัสแบบต่างๆ ดังรูป ก ส่วนสัญลักษณ์ของรีโอสตัส ดังแสดงในรูป ข ส่วนรูป ค จะแสดงโครงสร้างภายในของรีโอสตัสแบบวงกลม ซึ่งจะเห็นว่าปลายอีกด้านหนึ่งของผิวสัมผัสเมื่อคันกรีดเคลื่อนที่ออกห่างไปจากบริเวณส่วนที่ขั้วต่ออยู่ จะทำให้ความต้านทานเพิ่มขึ้น ซึ่งจะแสดงตามรูป ง ซึ่งคันกรีดจะเคลื่อนที่ต่ำลงโดยการหมุนแกนตามเข็มนาฬิกา

ด้วยเหตุนี้กระแสไฟฟ้าจึงไหลผ่านได้น้อยเนื่องจากค่าความต้านทานที่มีค่ามาก ในทางกลับกันถ้า

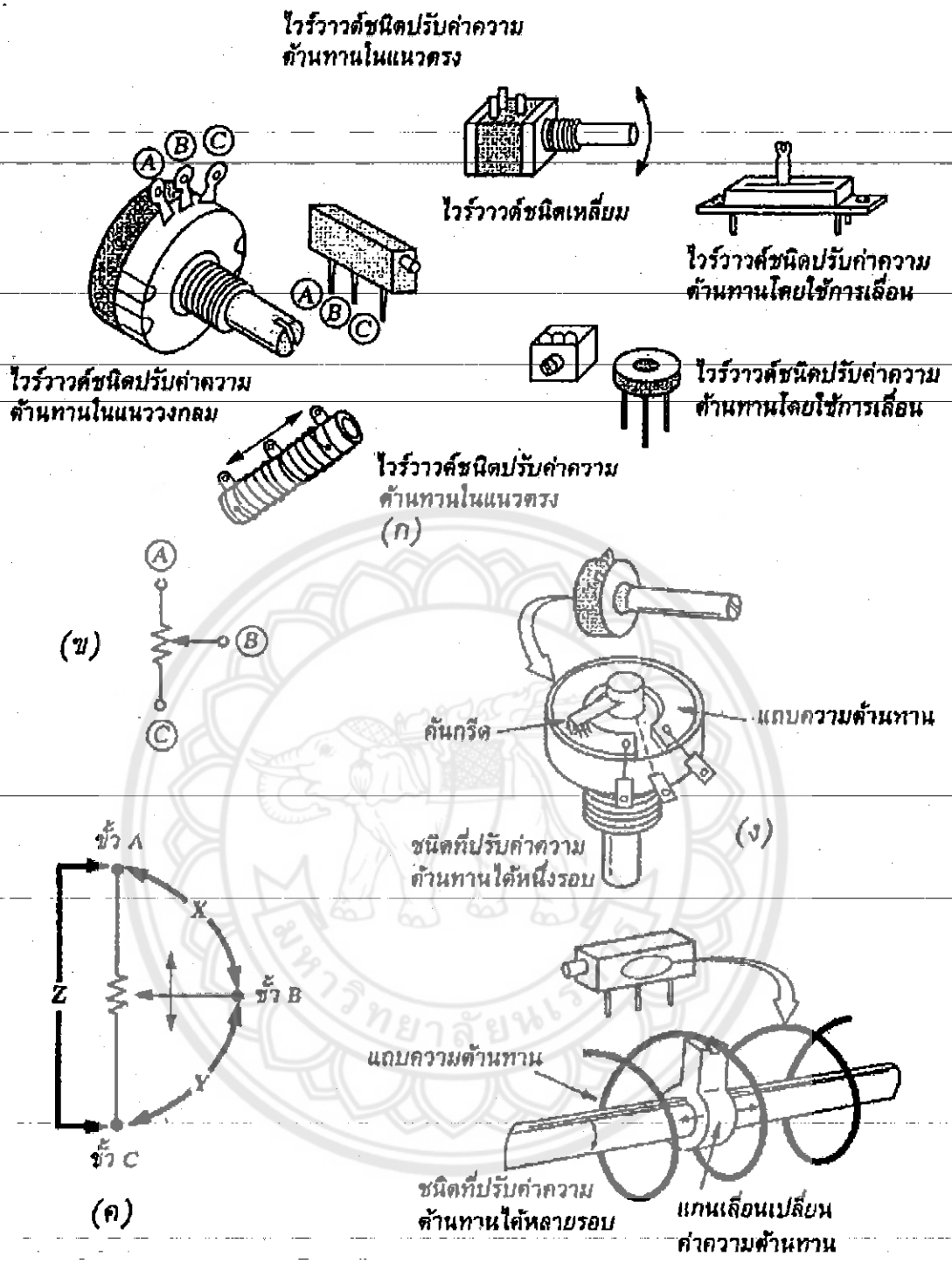
คันกรีดเคลื่อนที่เข้าใกล้ส่วนปลายที่มีขั้วต่ออยู่จะทำให้ค่าความต้านทานลดลง ดังแสดงในรูป จ ซึ่งคันกรีดจะเคลื่อนที่ขึ้นโดยการหมุนแกนทวนเข็มนาฬิกาและกระแสไฟฟ้าที่ไหลผ่านรีโอสตัสในกรณีนี้จะมีค่ามากเนื่องจากค่าความต้านทานที่ลดลงนั่นเอง



รูปที่ 2.5 ลักษณะของตัวต้านทานปรับค่าได้แบบต่างๆ

โพเทนชิโอมิเตอร์ (3 ขั้ว : A,B และ C)

รูปแสดงลักษณะภายนอกของโพเทนชิโอมิเตอร์แบบต่างๆ ซึ่งบางครั้งนิยมเรียกอุปกรณ์ชนิดนี้ว่า พอต (Pot) ดังแสดงในรูป ข ความแตกต่างระหว่างโพเทนชิโอมิเตอร์และรีโอสตัส คือจำนวนขั้วต่อใช้งาน ซึ่งขั้วต่อของโพเทนชิโอมิเตอร์จะมี 3 ขั้ว โดยการนำไปใช้งานสามารถต่อค่าความต้านทานได้ 3 แบบ ได้แก่ ระหว่าง A และ B (X) ระหว่าง B และ C (Y) และระหว่าง C และ A (Z) ส่วนที่เพิ่มเข้ามาที่ทำให้โพเทนชิโอมิเตอร์แตกต่างไปจากรีโอสตัส คือ ขั้วที่ 3 ที่ต่อเข้ากับปลายอีกด้านหนึ่งของแถบค่าความต้านทาน



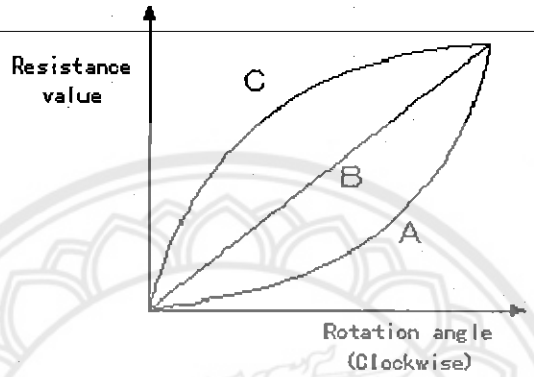
รูปที่ 2.6 หลักการทำงานของตัวต้านทานปรับค่าได้

เราสามารถแบ่งคุณลักษณะการทำงานของ Variable Value Resistor เป็น 3 ประเภทใหญ่ๆด้วยกับ คือ

1.แบบ A หรือ แบบ Log ซึ่งมีคุณสมบัติตามสมการ Log ครับ คือค่าจะกระโดดในช่วงท้ายครับตามรูปข้างล่างครับ

2.แบบ B หรือ แบบ Linear ซึ่งมีคุณสมบัติเป็นเชิงเส้น คือ แกน X =1 แกน Y = 1 X = 2 Y=2 คือเปลี่ยนแปลงไปเป็นตามลำดับไม่กระโดด ครับ

3.แบบ C เป็น แบบ ผกผันกลับจากแบบ A



รูปที่ 2.7 กราฟเปรียบเทียบลักษณะการทำงานของ Variable Value Resistor

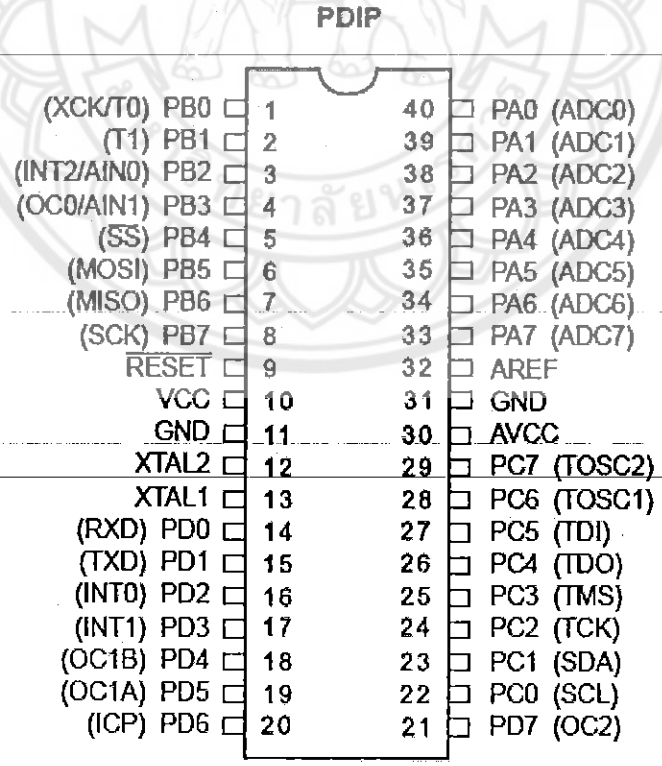
สำหรับการนำไปใช้งาน เราสามารถที่จะสลับกันใช้งานได้ทั้ง 3 แบบครับ แต่ว่าเรานิยมใช้แบบ A ส่วนปรับความดังเบา ที่ตำแหน่ง Gain หรือ Volume และนิยมใช้แบบ B กับภาค ToneControl พวก ปรับทุ่มแหลมครับ Treble Bass ครับ นอกจากนี้ยังต้องคำนึงถึงเรื่องของกายภาพ ในการใช้งาน VR แต่ละแบบ

2.3 Microcontroller ATmega32

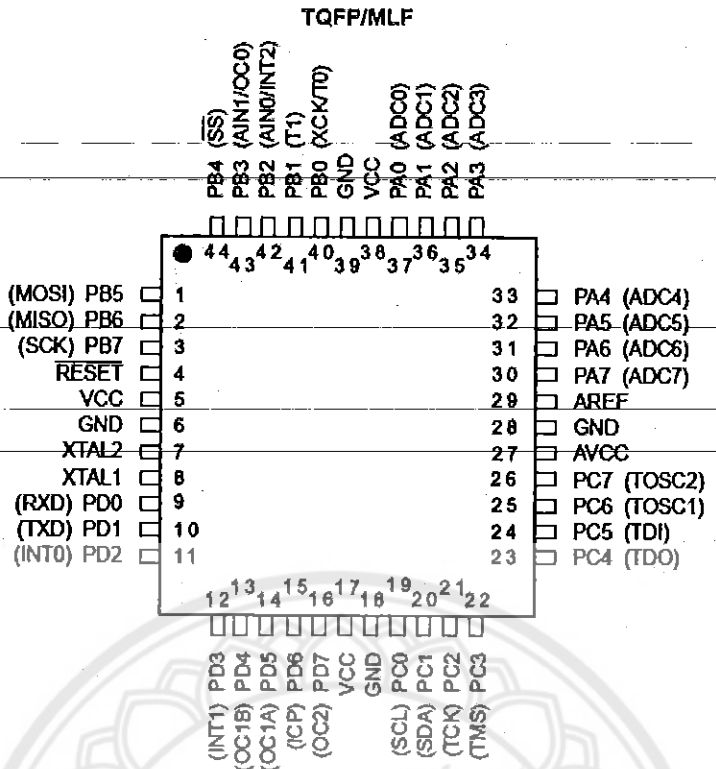
ในโครงการนี้จะใช้ไมโครคอนโทรลเลอร์ของ Atmega32 เป็นไมโครคอนโทรลเลอร์ (MCU) ที่ได้รวบรวมอุปกรณ์สนับสนุนการทำงานของ CPU ไว้มากมาย อาทิเช่น Analog to Digital, SPI , UART , Timer , Counter , PWM ซึ่งอุปกรณ์สนับสนุนการทำงานเหล่านี้ทำให้ MCU สามารถทำงานได้กว้างและใช้อุปกรณ์ต่อร่วมจากภายนอกน้อยมาก และ สามารถประมวลคำสั่งได้ภายใน 1 clock ในบทนี้จะนำเสนอข้อมูลบางส่วนที่เป็นการทำงานภายในของ AVR - MCU แนะนำคุณสมบัติและขาต่อใช้งานของไมโครคอนโทรลเลอร์ สถาปัตยกรรมภายในและรีจีสเตอร์ใช้งานทั่วไป ตำแหน่ง I/O รีจีสเตอร์สถานะและการใช้งาน EEPROM การรีเซ็ตและการอินเตอร์รัพท์ การสื่อสารอนุกรม การเปรียบเทียบสัญญาณอนาลอกและการแปลงสัญญาณอนาลอกเป็นดิจิตอล การทำงานของพอร์ตอินพุต/เอาท์พุต การทำงานของTimer / Counter & Watch dog และการใช้กลุ่มคำสั่งต่าง ๆ

2.3.1. คุณสมบัติและขาต่อใช้งานของไมโครคอนโทรลเลอร์

- สถาปัตยกรรมภายในถูกออกแบบให้ใช้สถาปัตยกรรมแบบ RISE (Reduce Instruction Set Computer) RISE คือ ทำให้การประมวลผลมีความเร็ว 1 คำสั่ง / 1 Clock หรือ CPU สามารถประมวลคำสั่งได้ 1 MIPS / MHz
- มีคำสั่งในการควบคุมการทำงานของไมโครคอนโทรลเลอร์จำนวน 118 คำสั่ง
- หน่วยความจำแบบ FLASH สำหรับบันทึก PROGRAM MEMORY ขนาด 32 Kbytes (ATMEGA 32)
- หน่วยความจำแบบ EEPROMสำหรับบันทึก DATA MEMORY ขนาด 1024 Byte (ATMEGA 32)
- หน่วยความจำแบบ RAM ขนาด 2K Byte (ATMEGA 32)
- ระบบการเปลี่ยนสัญญาณ ANALOG TO DIGITAL ขนาด 10 บิท จำนวน 8 CHANNEL
- ความถี่สัญญาณนาฬิกา 0 - 16 MHz (ATMEGA 32)
- ระบบการตรวจจับระดับสัญญาณอนาลอก (Analog Comparator)
- TIMER/COUNTER ขนาด 16 บิท 1 CHANNEL
- TIMER/COUNTER ขนาด 8 บิท 2 CHANNEL
- Vcc: 4.5 - 5.5 for ATMEGA 32



รูปที่ 2.8 Pinout ATmega32



รูปที่ 2.9 แบบตัวถังของ ATmega32

2.3.2. คุณสมบัติการแปลงสัญญาณอนาลอกเป็นสัญญาณดิจิตอล

- 10 – bit Resolution
- 0.5 LSB Integral Non – linearity
- 65 – 260 us Conversion Time
- Up to 15 KBPS at Maximum Resolution
- 8 Multiplexed Single Ended input Channels
- 0 – Vcc ADC input voltage rang
- Free running or Single Conversion Mode
- Interrupt on ADC Conversion complete

ใน Atmega 32 จัดให้มีวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลขนาด 10 บิต 8 Channel ซึ่งแต่ละ Channel จะรับสัญญาณเข้ามาทางแต่ละขาของพอร์ต A โดยในระบบจะมีวงจร SAMPLE HOLD เพื่อช่วยให้สัญญาณอนาลอกที่รับเข้ามาเพื่อแปลงเป็นสัญญาณดิจิตอลมีระดับสัญญาณคงที่ โดยวงจรแปลงสัญญาณอนาลอกเป็นดิจิตอลจะมีแหล่งจ่ายไฟและกราวด์แยกกันต่างหากจากแหล่งจ่ายไฟของระบบ ซึ่งในการใช้งานจริงไม่ควรให้ความแตกต่างของแรงดันไฟของวงจรอนาลอกและแรงดันไฟของระบบแตกต่างกันเกิน +/- 0.3 V. ซึ่งในการใช้งานจะต้องจ่ายแรงดันไฟอ้างอิงและกราวด์ที่ขา AREF ในช่วงของระดับแรงดัน Avcc – GND

2.3.3 การทำงานของ ATMEGA 32 กับการแปลงสัญญาณจากอนาลอกเป็นดิจิตอล

ในส่วนของการแปลงสัญญาณอนาลอกเป็นดิจิตอล สามารถแบ่งการทำงานได 2 Mode คือ

1. Single Conversion Mode

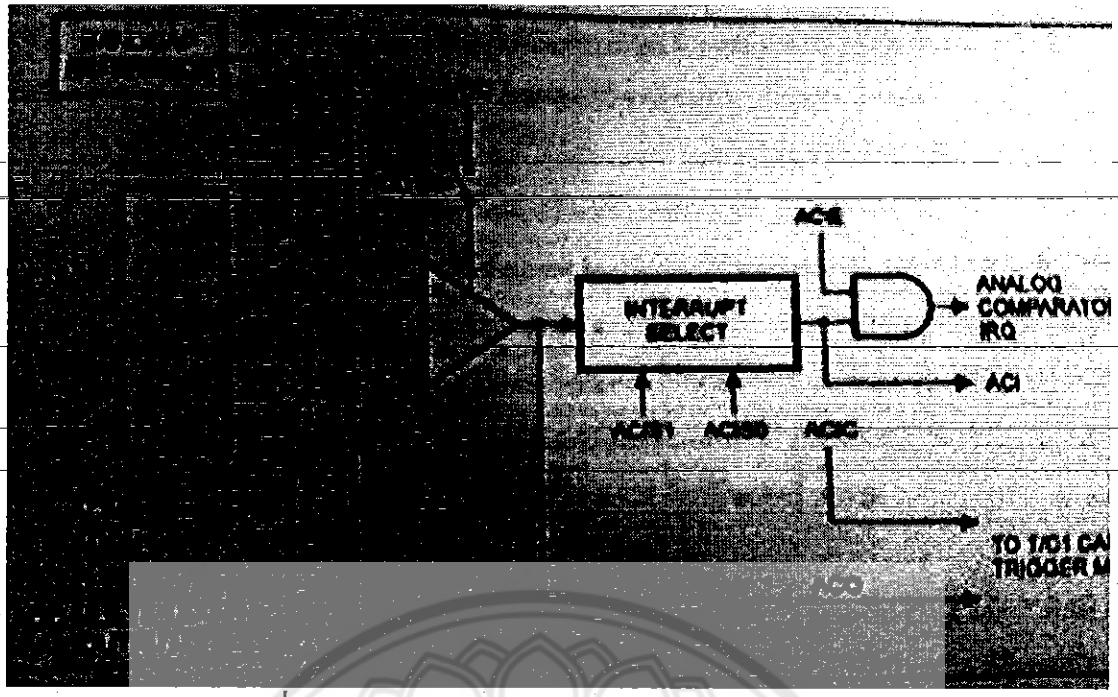
2. Free Running Mode

การทำงาน Single Conversion Mode ผู้ใช้ต้องเป็นผู้กำหนดการใช้งานขึ้นเอง แต่ในส่วนของ Free Running Mode วงจร Analog to digital จะเป็นตัวจัดการอ่านข้อมูลและเก็บใน ADC Data Register ซึ่งบิต ADFR ใน Register ADCSR จะเป็น บิตที่ใช้เลือก Mode การใช้งานของวงจร Analog to digital สำหรับการกำหนดให้วงจร Analog to digital ทำงานนั้น สามารถทำได้โดยการเซ็ตบิต ADEN ในรีจีสเตอร์ ADCHRA ให้เป็น 1 โดยบิตนี้จะเป็น 1 ไปตลอดจนกระทั่ง Conversion ของสัญญาณจะเรียบร้อยแล้วจึงทำให้บิตนี้เป็น 0 โดยอัตโนมัติ แต่ถ้าเป็นการเปลี่ยน Channel ของการแปลงสัญญาณขณะที่ Channel เดิมยัง Conversion อยู่ วงจร Analog to digital จะ Conversion สัญญาณ Channel เดิมให้เสร็จก่อนแล้วจึง Conversion สัญญาณ Channel ถัดไป โดยข้อมูลที่ได้จากการแปลงสัญญาณอะนาลอกเป็นดิจิตอลจะเก็บไว้ในรีจีสเตอร์ ADCH และ ADCL

2.3.4 การใช้งานโมดูลแปลงสัญญาณอนาลอกป็นดิจิตอล

ไมโครคอนโทรลเลอร์ เบอร์ ATmega 32 มีโมดูลแปลงสัญญาณอะนลอกเป็นดิจตอลหรือ ADC (Analog to Digital Converter) ความละเอียดขนาด 10 บิต (10-bit Resolution) ที่แรงดัน +5V หมายถึงเมื่อแปลงสัญญาณเป็นอะนาลอกเป็นดิจิตอลแล้วจะได้ค่าตัวเลขอยู่ระหว่าง 0-1024 โดยมีรูปแบบการแปลงสัญญาณอะนาลอกเป็นดิจิตอลแบบบซัคเซสซีฟ แอพพร็อกซิเมชั่น (Successive Approximation ADC) คือการแปลงแบบประมาณค่า โดยการสุ่มค่าดิจิตอลแล้วแปลงเป็นแรงดันอะนาลอกภายในโมดูล เพื่อใช้เปรียบเทียบกับแรงดันอะนาลอกด้านอินพุต เมื่อเปรียบเทียบได้ค่าแรงดันเท่ากัน โมดูล ADC จะให้ผลลัพธ์ออกมาเป็นค่าดิจิตอล ซึ่งการใช้วิธีการนี้เป็นที่นิยมเพราะมีความเที่ยงตรงสูงและการทำงานได้อย่างรวดเร็ว

โมดูล ADC จะมีจำนวน 8 ช่องสัญญาณใช้หลักการมัลติเพล็กซ์ (Multiplexer) เพื่อเลือกการทำงานในแต่ละช่อง กำหนดไว้ที่ขาพอร์ต A โดยมีแรงดันอินพุตระหว่าง 0 V (GND) ถึง VCC (แรงดันอินพุตที่ขา VCC ของไมโครคอนโทรลเลอร์ AVR) ผ่านวงจรสุ่มและเก็บค่า (Sample and Hold) วงจรโมดูล ADC แสดงดังรูปที่ 2.8



รูปที่ 2.10 บล็อกไดอะแกรมเปรียบเทียบแรงดันอะนาลอก

รีจิสเตอร์ที่เกี่ยวข้องกับโมดูลเปรียบเทียบแรงดันอะนาลอก

1. **รี**สจิสเตอร์ ADMUX (ADC Multiplexer Selection Register)

   รีจิสเตอร์กำหนดแรงดันอ้างอิง (Voltage Reference) รูบแบบการเก็บข้อมูลและการกำหนดอินพุทอะนาลอกอ้างอิงด้านบวกและลบ

2. **รีจิ**สเตอร์ ADCSRA (ADC Control and Status Register)

   รีจิสเตอร์กำหนดและแสดงสถานะการทำงานของโมดูล ADC

3. **รีจิ**สเตอร์ ADCL และ ADCH (The ADC Data Register)

   รีจิสเตอร์เก็บข้อมูลที่ได้จากการแปลงสัญญาณอะนาลอกเป็นดิจิตอล

4. **รีจิ**สเตอร์ SFIOR (Special Function IO Register)

   รีจิสเตอร์กำหนดการกระตุ้นจากแหล่งสัญญาณภายนอกให้กับโมดูล ADC

รายละเอียดของรีเตอร์ที่เกี่ยวข้องกับการใช้งานโมดูลดังนี้

1. **รีสจิสเตอร์ ADMUX (ADC Multiplexer Selection Register)**

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

ตารางที่ 2.1 ตารางรีสจิสเตอร์ ADMUX (ADC Multiplexer Selection Register)

- บิตที่ 7-6 : บิต REFS1:0 (Reference Selection Bits)

  บิตที่กำหนดแหล่งแรงดันอ้างอิงสำหรับโมดูล ADC การกำหนดรายละเอียดตามตารางที่ 2.2 ดังนี้

REFS1	REFS0	แหล่งแรงดันอ้างอิง
0	0	AREF , ปิดการใช้งานแรงดันอ้างอิงภายใน (Vref)
0	1	ใช้แรงดัน AVCC กับตัวเก็บประจุภายนอกที่ขา AREF
1	0	สงวนไว้
1	1	ใช้แรงดันอ้างอิงภายในที่ 2.56 V กับตัวเก็บประจุภายนอกที่ขา AREF

ตารางที่ 2.2 แรงดันอ้างอิงสำหรับโมดูล ADC

- บิตที่ 5 : บิต ADLAR (ADC Left Adjust Result)

  บิตกำหนดรูปแบบการเก็บข้อมูลในรีจิสเตอร์ ADC Data ขนาด 16 บิต มี 2 รูปแบบคือการเก็บชิดบิตสูงสุด (MSB bits) หรือเก็บชิดบิตต่ำสุด (LSB bits) ดูเพิ่มเติมในรีจีสเตอร์ ADC Data Register (ADCH, ADCL)

- บิตที่ 4:0 บิต MUX4:0 (Analog Channel Gain Selection Bits)

  บิตกำหนดขาแรงดันอะนาลอกอินพุตด้านบวกและด้านลบ

  รายละเอียดแสดงดังตารางที่ 2.3

MUX4.0	อินพุตเดี่ยว	ความแตกต่างของอินพุท ด้านบวก	ความแตกต่างของอินพุทด้าน ลบ	Gain
00000	ADC0			
00001	ADC1			
00010	ADC2	N/A		
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			

01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010[1]		ADC0	ADC0	200x
01011[1]		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110[1]	N/A	ADC2	ADC2	200x
01111[1]		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.22 V (V_{BG})	N/A		
11111	0 V (GND)			

ตารางที่ 2.3 การเลือกช่องสัญญาณอินพุตอะนาลอกและตัวคูณอัตราขยาย

หมายเหตุ (1) : ไม่ได้ทดสอบบน ATmega แบบ PDIP ทดสอบเฉพาะแบบ TQFP และ QFN/ML

โดยที่ผลการแปลงสัญญาณอะนาลอกเป็นดิจิตอล คำนวณผลลัทธ์ได้จากสูสตรต่อไปนี้

- เมื่อทำงานในโหมดสัญญาณเดี่ยว

$$ADC = \frac{V_{IN} * 1024}{V_{REF}}$$

โดย Vin คือ แรงดันด้านขาอินพุต

Vref คือ แรงดันอ้างอิง

รายละเอียดการกำหนดแรงดันอินพุทและแรงดันอ้างอิง ตามตารางที่ 13

- ถ้าทำงานในช่องที่มีความแตกต่างของสัญญาณอินพุตด้านบวกและลบ

$$ADC = \frac{(V_{POS} * V_{NEG}) * 512}{V_{REF}}$$

โดย Vpos คือ แรงดันอินพุตด้านบวก

Vneg คือ แรงดันอินพุตด้านลบ

Vref คือ แรงดันอ้างอิง

GAIN คือ ตัวคูณอัตราขยาย

2. รีจิสเตอร์ ADCSRA (ADC Control and Status Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

ตารางที่ 2.4 รีจิสเตอร์ ADCSRA (ADC Control and Status Register)

บิตที่ 7 : บิต ADEN (ADC Enable)

เซตบิต ADEN เป็น "1" เพื่อเปิดการใช้งานโมดูล ADC

- บิตที่ 6 : บิต ADSC (ADC Start Conversion)

เซตบิต ADSC เป็น "1" เพื่อกำหนดให้โมดูลเริ่มต้นแปลงสัญญาณอะนาลอกเป็นดิจิตอล เมื่อแปลงเสร็จสมบูรณ์บิต ADSC จะถูกเซตเป็น "0" การเซตค่า "0" จะไม่มีผลใดๆกับโมดูล

- บิตที่ 5 : บิต ADATE (ADC Auto Trigger Enable)

เซตบิต ADATE เป็น "1" เพื่อเปิดการเปิดการกระตุ้น (Trigger) สัญญาณอัติโนมัติ โดยแหล่งสัญญาณในการกระตุ้นกำหนดบิต ADTS ที่รีจิสเตอร์ SFIOR

- บิตที่ 4 : บิต ADIF (ADC Interrupt Flag)

  บิต ADIF จะถูกเซตเป็น "1" โมดูลแปลงสัญญาณอะนาลอกเป็นดิจิตอลเสร็จสมบูรณ์และข้อมูลได้ถูกเขียนไปที่รีจีสเตอร์ ADCD (ADCH, ADCL) แล้ว หากมีการเปิดใช้งานอินเตอร์รัปต์เนื่องจากโมดูล ADC และเปิดใช้งานอินเตอร์รัปต์ โดยรวมจะส่งผลให้เกิดอินเตอร์รัปต์ขึ้น

- บิตที่ 3 : บิต ADIE (ADC Interrupt Enable)

  เซตบิต ADIE เป็น "1" เพื่อเปิดการใช้งานอินเตอร์รัปต์เนื่องจากโมดูล ADC (ต้องเปิดอินเตอร์รัปต์โดยรวมด้วย)

- บิตที่ 2:0 : บิต ADPS2:0 (ADC Prescaler Select Bits)

  บิตกำหนดปรีสเกลเลอร์สำหรับใช้ในการหารสัญญาณนาฬิกาสำหรับโมดูล ADC รายละเอียดแสดงดังตารางที่ 2.5

ADPS2	ADPS1	ADPS0	ปรีสเกลเลอร์หารสัญญาณความถี่ (XTAL) และสัญญาณนาฬิกาโมดูล ADC
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

ตารางที่ 2.5 ปรีสเกลเลอร์สำหรับโมดูล

3. รีจิสเตอร์ ADCL และ ADCH (The ADC Data Register)

เมื่อ ADLAR=0

บิตที่	15	14	13	12	11	10	9	8
ชื่อบิต	-	-	-	-	-	-	ADC9	ADC8
ชื่อบิต	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
บิตที่	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R	R	R	R
	R	R	R	R	R	R	R	R
ค่าเริ่มต้น	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

ตารางที่ 2.6 รีจิสเตอร์ ADCL และ ADCH (The ADC Data Register) เมื่อ ADLAR=0

- บิตที่ 9:0: บิต ADC9:0 (ADC Data Register)

  ผลลัพธ์ของข้อมูลที่ได้จากโมดูล ADC จะเก็บทางขวาสุดของข้อมูล โดยการเซตบิต ADLAR (ADC Left Adjust Result) เป็น "0" ในรีจีสเตอร์ ADMUX

เมื่อ ADLAR =1

บิตที่	15	14	13	12	11	10	9	8
ชื่อบิต	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ชื่อบิต	ADC1	ADC0	-	-	-	-	-	-
บิตที่	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R	R	R	R
	R	R	R	R	R	R	R	R
ค่าเริ่มต้น	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0

ตารางที่ 2.7 รีจิสเตอร์ ADCL และ ADCH (The ADC Data Register) เมื่อ ADLAR=1

- บิตที่ 15:6: บิต ADC9:0 (ADC Data Register)

  ผลลัพธ์ของข้อมูลที่ได้จากโมดูล ADC จะเก็บทางขวาสุดของข้อมูล โดยการเซตบิต ADLAR (ADC Left Adjust Result) เป็น "1" ในรีจีสเตอร์ ADMUX

4. รีจิสเตอร์ SFIOR (Special Function IO Register)

บิตที่	7	6	5	4	3	2	1	0
ชื่อบิต	ADTS2	ADTS1	ADTS0	-	ACME	PUD	PSR2	PSR10
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
ค่าเริ่มต้น	0	0	0	0	0	0	0	0

ตารางที่ 2.8 รีจิสเตอร์ SFIOR (Special Function IO Register)

- บิตที่ 7:5: บิต ADTS2:0 (ADC Auto Trigger Source)

  บิตที่กำหนดการกระตุ้นโมดูล ADC จากแหล่งสัญญาณภายนอกโดยขึ้นอยู่กับบิต ADATE ในรีจีสเตอร์ ADCSRA หาก ADATE ถูกเซตเป็น "1" การหำหนดบิตกระตุ้นการทำงานของโมดูล ADC ขะขึ้นอยู่กับสัญญาณจากภายนอกตามบิต ADTS2:0 หากบิต ADCSRA ถูกเซตเป็น "0" บิต ADST2:0 จะไม่มีผลใดๆ กับโมดูล ADC การกำหนดแหล่งกระตุ้นสัญญาณจากภายนอกกำหนดได้กังตารางที่ 2.9

ADTS2	ADTS1	ADTS0	แหล่งกระตุ้นสัญญาณอัตโนมัติ
0	0	0	โหมดทำงานอิสระ
0	0	1	เปรียบเทียบแรงดันอะนาลอก
0	1	0	อินเตอร์รัปต์เนื่องจากสัญญาณภายนอก ช่องที่ 0
0	1	1	โมดูลเปรียบทียบสัญญาณของไทเมอร์/เคาน์เตอร์ 0
1	0	0	ไทเมอร์/เคาน์เตอร์ 0 โอเวอร์โฟลว์
1	0	1	โมดูลเปรียบเทียบสัญญาณของไทเมอร์/เคาน์เตอร์
1	1	0	ไทเมอร์/เคาน์เตอร์ 1 โอเวอร์โฟลว์
1	1	1	โมดูลตรวจจับสัญญาณอินพุตของไทเมอร์

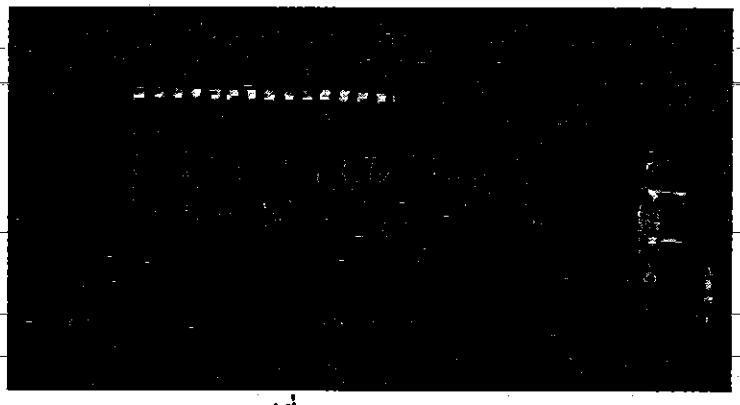
ตารางที่ 2.9 แหล่งกระตุ้นสัญญาณอัตโนมัติของโมดูล ADC

- บิตที่ 4 : บิต RES (Reserved Bot)

  บิตนี้สงวนไว้ไม่ได้ใช้งาน อ่านค่าได้เป็น "0"

2.4 จอแสดงผล

LCD ที่ใช้เป็นแบบ 16 ตัวอักษร 1 บรรทัดซึ่งมีลักษณะดังรูปที่ 2.2



รูปที่ 2.11 จอแสดงผล

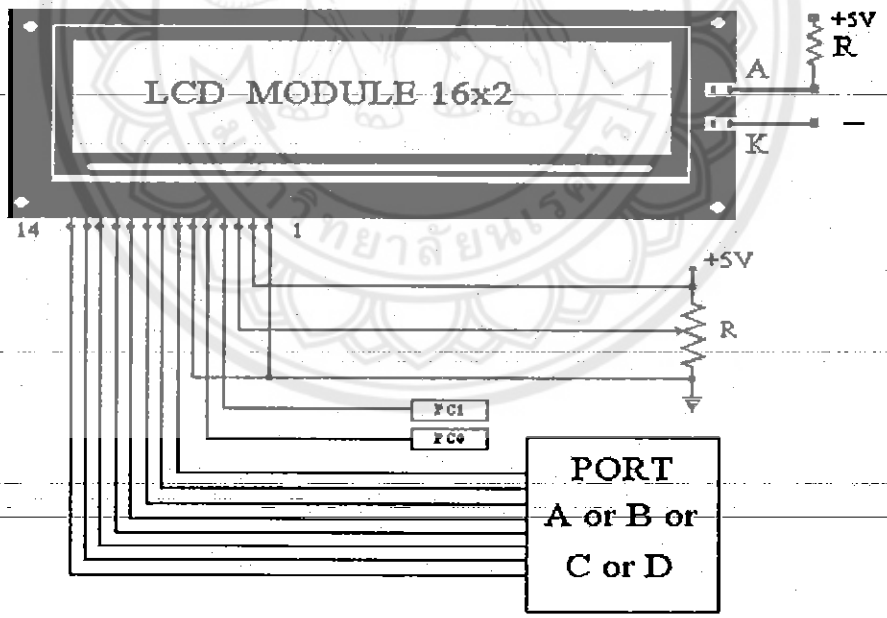
การติดต่อกับ LCD Module

มีอยู่ด้วยกัน 2 แบบ

1. แบบ 8 บิต

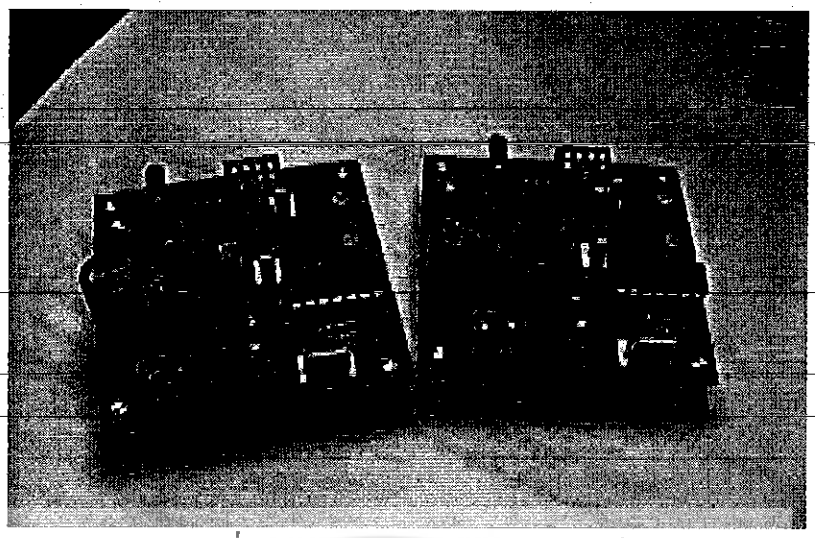
2. แบบ 4 บิต

ซึ่งในโครงงานนี้ซึ่งจะใช้การต่อ LCD Module แบบ 8 บิต ซึ่งจะแสดงการต่อดังในรูปที่ 2.3



รูปที่ 2.12 การแสดงการต่อ LCD แบบ 8 บิต

2.5 โมเดมอเนกประสงค์

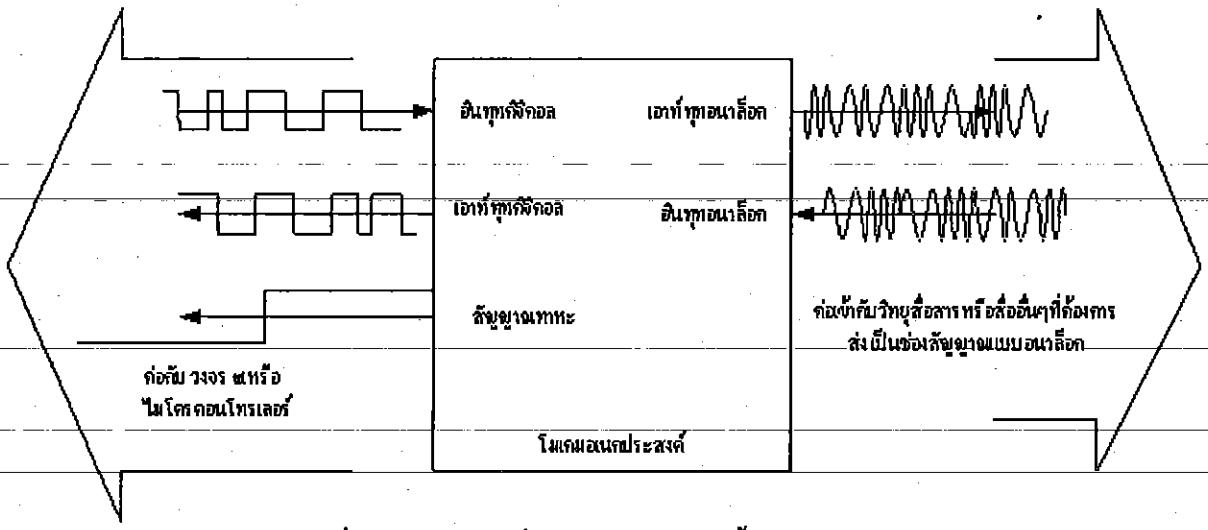


รูปที่ 2.13 รูปภาพโมเดมอเนกประสงค์

คณะสมบัติ

1. ความเร็วในการส่งสัญญาณ 1200 bps
2. มอดูเลทเชิงความถี่ (FSK)
3. รับสัญญาณดิจิตอลอินพุทและส่งสัญญาณดิจิตอลเอาท์พุทเป็นแบบ TTL (ต่อใช้งานกับไมโครคอนโทรเลอร์ได้โดยตรง)
4. มีช่องสัญญาณตรวจสอบสัญญาณคลื่นพาหะ (ใช้สำหรับตรวจสอบว่าเครื่องส่งหยุดส่งสัญญาณหรือยัง)
5. มีภาคจ่ายไฟให้วงจรในตัว ผู้ใช้สามารถใช้แหล่งจ่ายได้หลายรูปแบบ
6. สามารถปรับความเร็วให้เหมาะสมกับการใช้งานได้
7. สามารถสื่อสารแบบ HALF และ FULL DUPLEX ได้ชิพที่ใช้เป็นมาตรฐาน CCI

หลักการที่จะนำวงจรนี้ไปประยุกต์ใช้งาน

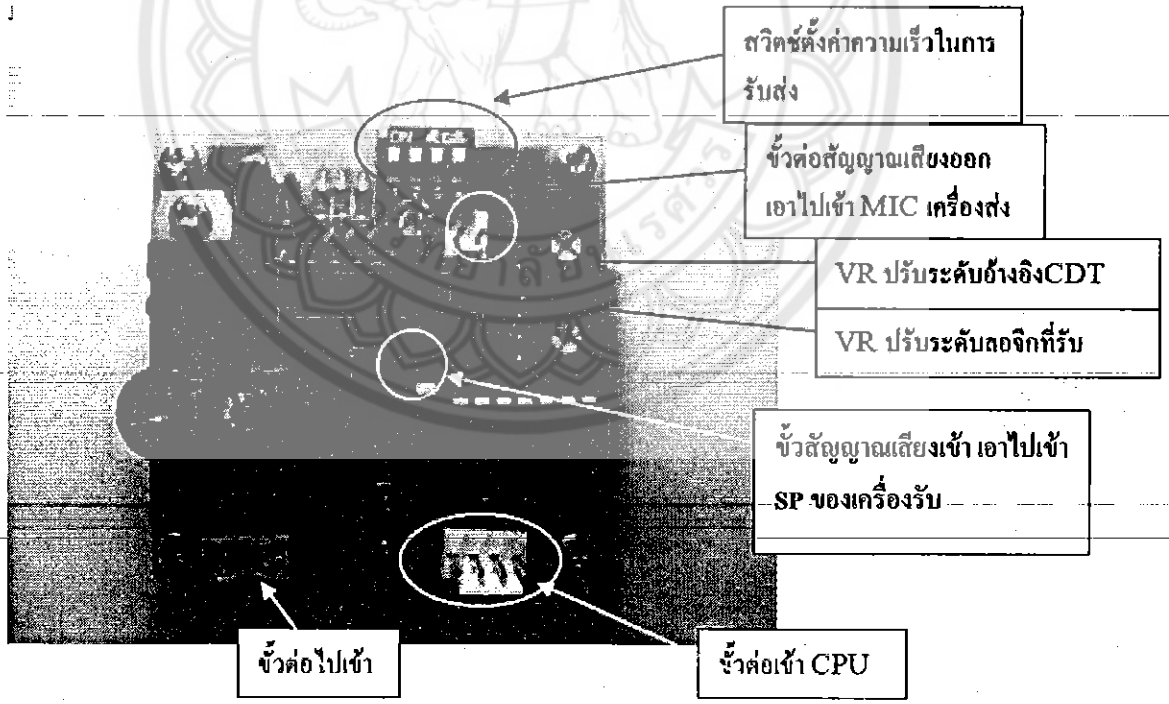


รูปที่ 2.14 หลักการที่จะนำวงจรโมเดมนี้ไปประยุกต์ใช้งาน

หลักการมีอยู่ว่า โดยทั่วไปแล้วช่องสัญญาณในการติดต่อสื่อสารแบบไร้สายจะเป็นแบบอะนาลอกอยู่แล้ว (คลื่นวิทยุสื่อสาร) ดังนั้นเมื่อต้องการส่งสัญญาณที่เป็นดิจิตอลไปในช่องสัญญาณเหล่านี้ก็จำเป็นที่จะต้องทำการมอดูเลทสัญญาณแบบดิจิตอล เพื่อให้สัญญาณดิจิตอลกลายเป็นสัญญาณอะนาลอกก่อน จากนั้นจึงทำการมอดูเลทสัญญาณดิจิตอลที่เปลี่ยนเป็นอะนาลอกแล้ว เข้ากับคลื่นวิทยุสื่อสาร เพื่อส่งออกอากาศ หลายคนคงสงสัยว่าทำไม จึงไม่นำสัญญาณดิจิตอลมามอดูเลทกับคลื่นวิทยุสื่อสารเลย เหตุผลก็มีอยู่ว่าสัญญาณดิจิตอลเองมีองค์ประกอบของความถี่อะนาลอกที่มากๆ (Harmonic) แต่ช่องสัญญาณที่จะส่งออกไปมีแบนด์วิท (ความกว้างของความถี่ที่ใช้งาน) ที่จำกัด ดังนั้นหากเรานำสัญญาณดิจิตอลมามอดกับคลื่นวิทยุสื่อสารเลย ก็จะเกิดสัญญาณลบกวนสูงมาก และเมื่อสัญญาณไปถึงปลายทาง ก็จะเกิดความผิดเพี้ยนขึ้น แต่ถ้าหากเรานำสัญญาณดิจิตอลมาเปลี่ยนเป็นสัญญาณอะนาลอกในย่านของช่องสัญญาณที่รับได้ เช่น เปลี่ยนสัญญาณดิจิตอลให้อยู่ในย่านของความถี่ ตั้งแต่ 300 - 3.4 KHz (ย่านเสียงของคนเราในระบบโทรศัพท์) ก่อนก็จะสามารถส่งสัญญาณนั้นไปได้ในระบบโทรศัพท์ เช่น โมเดมที่เราใช้เล่นอินเตอร์เน็ต เป็นต้น วิทยุสื่อสารสื่อสารก็เช่นกัน หากต้องการจะส่งสัญญาณดิจิตอลไปในก็ต้องนำมาเปลี่ยนเป็นอะนาลอกก่อนเช่นกัน โดยเทคนิคการเปลี่ยนสัญญาณดิจิตอลเป็นอะนาลอกหรือเรียกว่าการมอดูเลทแบบดิจิตอลนั้น ทางเทคนิคมีด้วยกันหลายวิธีมาก เช่น FSK, PSK, QPSK, BPSK, DM อื่นๆ อีกมากมาย ซึ่งการจะเลือกใช้วิธีการใดก็ขึ้นอยู่กับความเหมาะสมของการนำไปใช้งาน แต่ละแบบก็จะมีข้อดีข้อเสียที่แตกต่างกันไป จากที่ทราบแล้วว่าหลักการการส่งสัญญาณดิจิตอลไปในสื่อต่างๆ สามารถทำได้อย่างไร ไปบ้างแล้ว ก็คงจะพอเข้าใจว่าวงจรที่ผมนำเสนอมีประโยชน์อย่างไร ไม่ว่าจะนำมาประยุกต์เข้ากับวิทยุสื่อสารสื่อสาร ก็สามารถส่งสัญญาณดิจิตอลไปได้ ยิ่งถ้าหากกำลังส่งของวิทยุสื่อสารสื่อสารสูงมากๆ ก็จะทำให้เราสามารถส่งสัญญาณดิจิตอลที่เราจะนำไปควบคุมอุปกรณ์อะไรสักอย่าง ครอบคลุมพื้นที่ได้ไกลมากขึ้น อาจจะเป็น 10 กิโลเมตรขึ้นไป ก็สามารถทำได้

วงจรที่นำเสนอ เมื่อดูตามคุณสมบัติแล้วก็เพียงพอต่อการนำไปใช้งานส่งสัญญาณควบที่ไม่เร็วมากนัก คือ ในหนึ่งวินาทีสามารถส่งสัญญาณควบคุมได้สูงสุด 1200 Bit ถ้าคิดเป็นตัวอักษรก็ประมาณ 150 ตัวอักษรต่อหนึ่งวินาที หรือ 150 bps ก็นับว่าไม่ช้ามากนัก การใช้งานถ้าหากเป็นโครงงานที่สร้างด้วย MCS51 หรือไมโครคอนโทรเลอร์ตระกูลต่างๆ ก็สามารถนำวงจรนี้ไปต่อร่วมกับ Port (RS232) ของ ไมโครคอนโทรเลอร์ได้เลยหรือ Port อื่นๆที่ต้องการ จากนั้นอาจนำเอาท์พุทไปต่อที่ MIC ของเครื่องส่งวิทยุสื่อสารก็ได้ และชุดรับก็นำสัญญาณจากลำโพงของเครื่องรับมาต่อเข้ากับอินพุทอะนาลอก เพียงเท่านี้เมื่อมีการส่งสัญญาณดิจิตอลเข้ามาที่ชุดส่งๆ ก็จะส่งสัญญาณอะนาลอกไฟผสมกับคลื่นวิทยุสื่อสารแล้วส่งออกอากาศ เมื่อชุดรับรับได้ก็จะส่งสัญญาณดิจิตอลคืนออกมาให้ ระยะทางขึ้นอยู่กับกำลังส่งของเครื่องวิทยุสื่อสารและระบบสายอากาศครับ และวงจรที่ออกแบบมายังสามารถปรับระดับสัญญาณรบกวนได้อีกด้วย ก็คือ เมื่อในบริเวณเดียวกันมีการใช้คลื่นวิทยุสื่อสารความถี่เดียวกันอาจจะทำให้เกิดการรบกวนกันของความถี่ วงจรนี้สามารถปรับระดับความแรงของสัญญาณที่ต้องการรับได้ เพื่อไม่ให้สัญญาณที่อ่อนกว่าเข้ามา

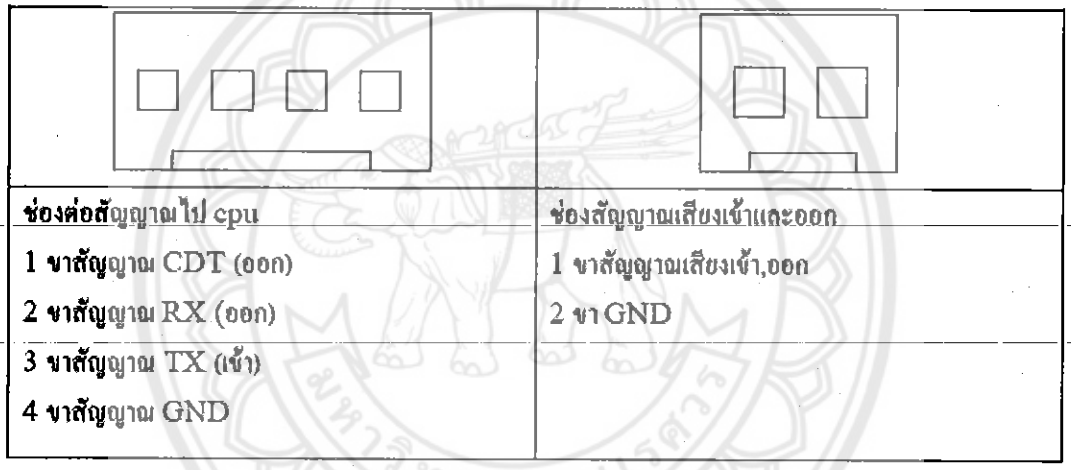
การใช้งานโมเดมอเนกประสงค์



รูปที่ 2.15 การใช้งานโมเดมอเนกประสงค์

การใช้งานเริ่มจากต่อไฟเลี้ยงเข้าวงจรตั้งแต่ 6Vdc-15Vdc หรือ AC ตั้งโหมดความเร็วของวงจรด้วยการปรับสวิตช์ตำแหน่ง 2-4 ตามตาราง สวิตช์หมายเลข 1 เป็นสวิตช์ปิด,เปิด IC หลักที่ทำหน้าที่เป็นตัวมอดูเลทและดีมอดูเลท (เนื่องจาก ICมีความไวต่อไฟฟ้าสถิตมากและอาจเสียหายได้ง่ายเมื่อมีการต่อใช้งานที่ไม่ถูกวิธี เมื่อต้องการใช้งานวงจรให้เลื่อนสวิตช์ตำแหน่งที่ 1 ไปที่ ON ตามรูป ไฟจะเข้าไปเลี้ยง IC วงจรทำงานได้ จากนั้นต่อสัญญาณเสียงออกไปเข้าที่ช่อง MIC ของเครื่องส่งวิทยุ และต่อช่องสัญญาณเสียงเข้ามาจากลำโพงของเครื่องรับส่งวิทยุ ในกรณีที่ส่งสัญญาณทางเดียวด้านเครื่องส่งไม่ต้องต่อช่องสัญญาณเสียงเข้าก็ได้ และในกรณีเป็นภาครับอย่างเดียวไม่จำเป็นต้องต่อเสียงออกไปเข้าช่อง MIC ก็ได้ดูรูปประกอบ

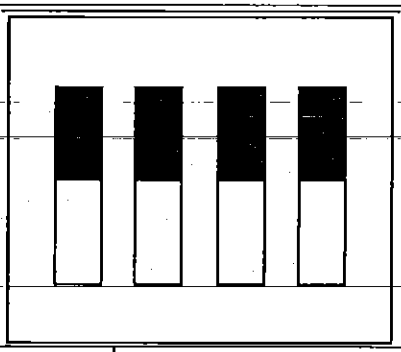
เมื่อต่อวงจรและ set ระดับแรงดันอ้างอิงแล้วก็สามารถนำสัญญาณจากไมโครคอนโทรเลอร์ต่อเข้าในช่องเสียบได้เลย เท่านี้ก็สามารถส่งสัญญาณดิจิตอลผ่านเครื่องวิทยุสื่อสารได้แล้ว



ช่องต่อสัญญาณไป cpu	ช่องสัญญาณเสียงเข้าและออก
1 ขาสัญญาณ CDT (ออก)	1 ขาสัญญาณเสียงเข้า,ออก
2 ขาสัญญาณ RX (ออก)	2 ขา GND
3 ขาสัญญาณ TX (เข้า)	
4 ขาสัญญาณ GND	

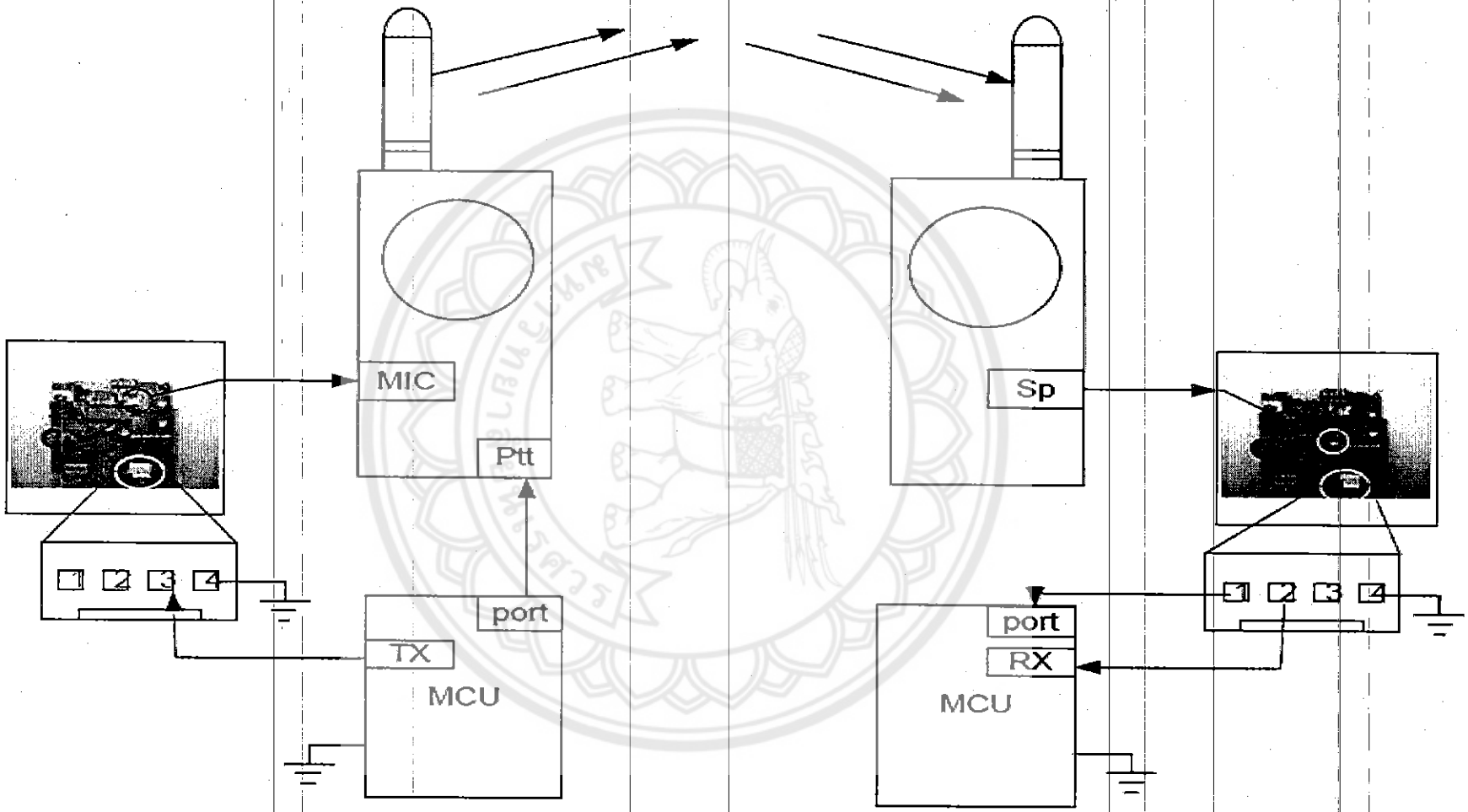
รูปที่ 2.16 รูปแสดงขาของช่องสัญญาณไป cpu และ ช่องสัญญาณเสียงเข้าและออก

i5004214 ค.2
ปร
ท1930
2551



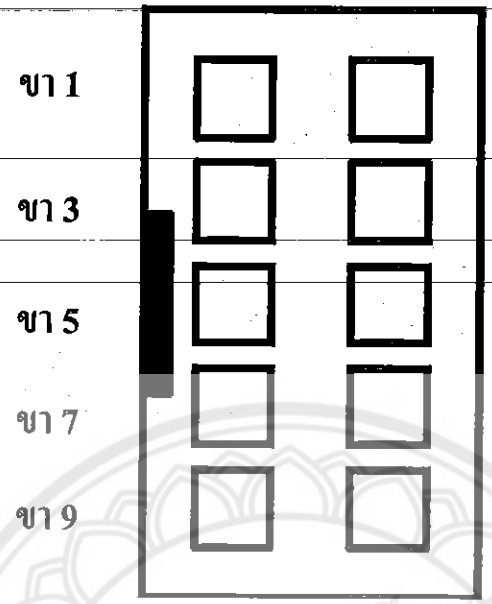
สวิทช์ตำแหน่งที่ 1 = Power IC			ON			
ตำแหน่ง Sw			ความเร็ว		ความถี่	
2	3	4	รับ	ส่ง	รับ	ส่ง
ON	ON	ON	1200	1200	M 1300	M 1300
					S 2100	S 2100
ON	ON	OFF	75	1200	M 390	M 1300
					S 450	S 2100
ON	OFF	ON	75	600	M 390	M 1300
					S 450	S 1700
ON	OFF	OFF	600	600	M 1300	M 1300
					S 1700	S 1700
OFF	ON	ON	1200	75	M 1300	M 390
					S 2100	S 490
OFF	ON	OFF	600	75	M 1300	M 390
					S 1700	S 450
OFF	OFF	ON	75	75	M 390	M 390
					S 450	S 450
OFF	OFF	OFF	1200	ไม่ส่ง	M 1200	ไม่ส่ง
					S 2200	ไม่ส่ง

ตารางที่ 2.10 ตารางการทำงานของขาโมเดม



รูปที่ 2.17 ภาพหลักการต่อวงจรเพื่อใช้งานโมเดมเอนกประสงค์

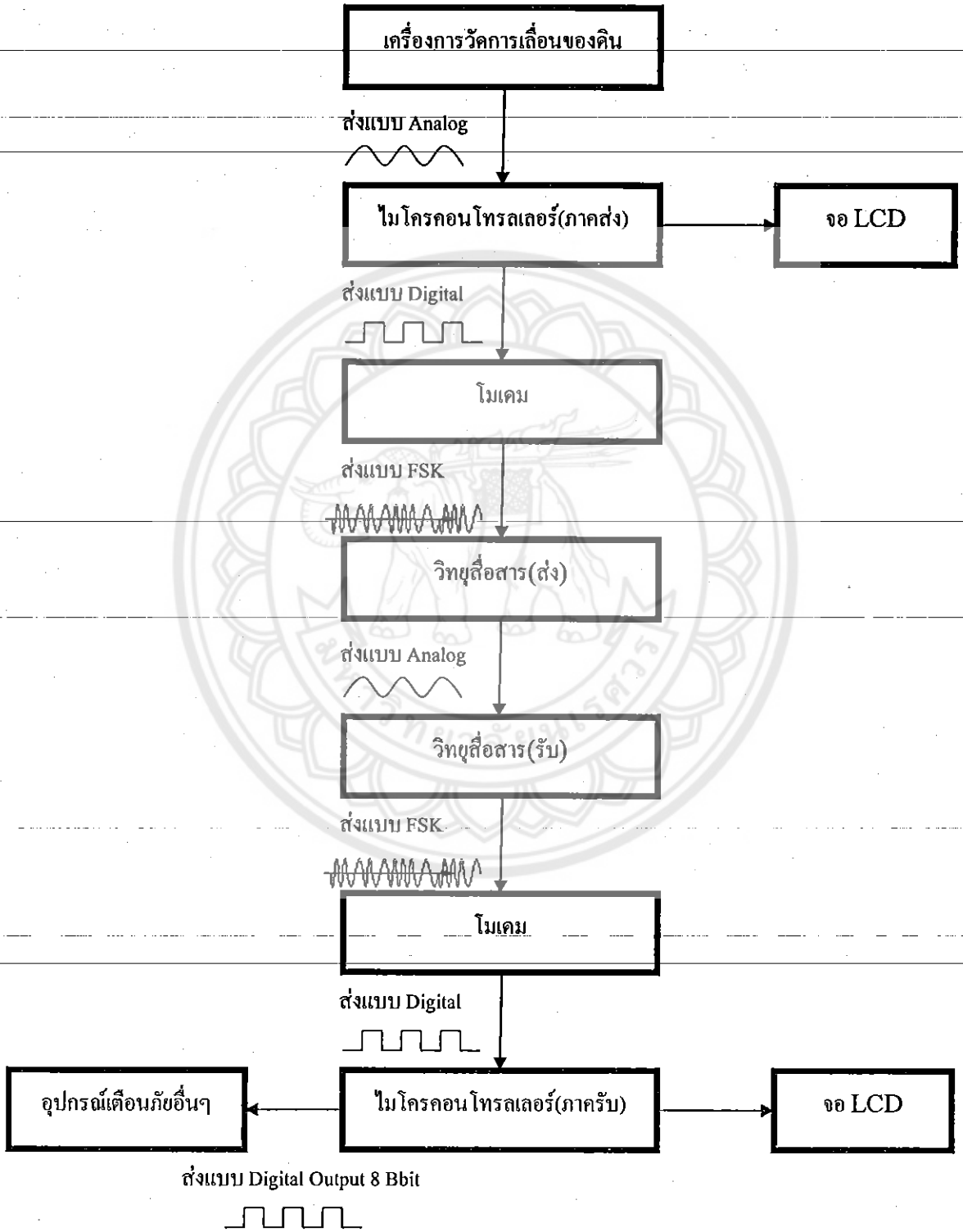
การใช้งานพอร์ท Header 5X2



รูปที่ 2.18 ภาพแสดงตำแหน่งขาของแต่ละขาตั้งงแต่ 1-10

บทที่ 3
วิธีการดำเนินโครงงาน

3.1 หลักการทำงานทั่วไป



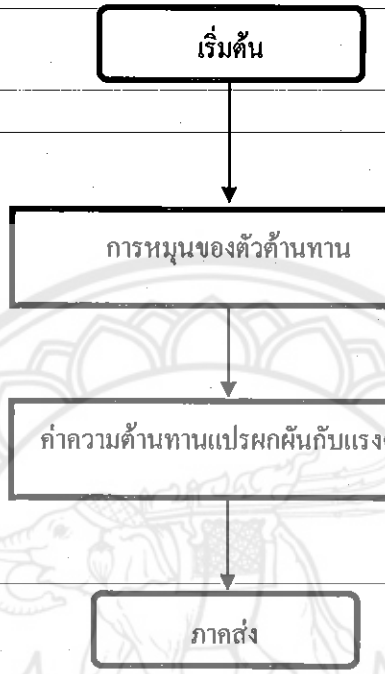
รูปที่ 3.1 บล๊อกไดอะแกรมหลักการทำงานทั่วไป

หลักการทำงานของแต่ละส่วน

จากบล็อกไดอะแกรม จะเห็นว่าเราสามารถแบ่งการทำงานของโครงการได้ออกเป็น 3 ส่วน ทำหน้าที่ดังต่อไปนี้

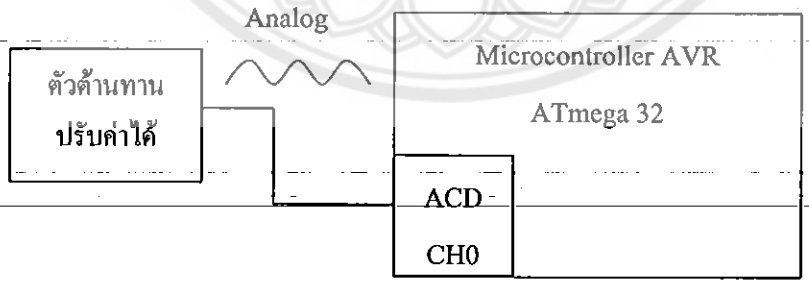
3.2 หลักการทำงานของเครื่องวัดการเลื่อนของดิน

3.2.1 บล็อกไดอะแกรมเครื่องวัดการเลื่อนของดิน



รูปที่ 3.2 บล็อกไดอะแกรมหลักการเครื่องวัดการเลื่อนของดิน

3.2.2 การออกแบบ

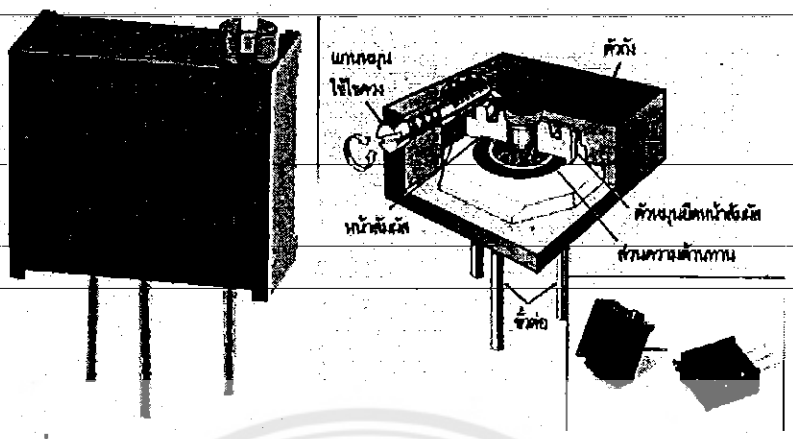


รูปที่ 3.3 รูปออกแบบเครื่องวัดการเลื่อนของดิน

*** สาย ACD_CH0 เป็นสายที่ใช้การส่งสัญญาณในลักษณะ Analog

3.2.3 การเลือกใช้ตัวต้านทานปรับค่าได้

ทางคณะผู้จัดทำได้เลือกใช้ตัวต้านทานปรับค่าได้ขนาด 500 โอห์ม รอบหมุน 20 รอบมีลักษณะดังรูป 3.4

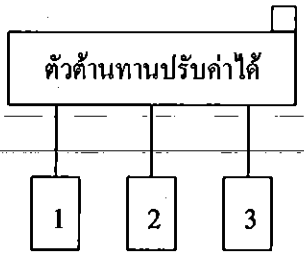


รูปที่ 3.4 รูปตัวต้านทานปรับค่าได้ขนาด 500 โอห์ม รอบหมุน 20 รอบ

Technical specification			
Power rating	0.5W	Operating Temperature	-55 to 150°C
Voltage Maximum	300V	End resistance	2 Ω max
Effective Travel (Nominal)	25 turns nom.	Contact resistance variation	1% or 1Ω (whichever is greater)
Resistance Tolerance	±10%	Linear or Logarithmic	Linear
Temperature Coefficient	±100ppm/°C	Lead Pitch	2.54 mm
Operating Torque	35 mNm	Pins	0.51 x 0.03 mm
Rotational Life	200 cycles	Dimensions	10 x 9.52 x 4.83 mm

ตารางที่ 3.1 คุณสมบัติของตัวตานทานปรับค่าได้

3.2.4 การนำไปใช้งาน



1. VCC
2. ACD_CH0
3. GND

รูปที่ 3.5 แสดงขาของตัวต้านทานปรับค่าได้

3.2.5 อุปกรณ์เครื่องมือวัดการเลื่อนของดิน

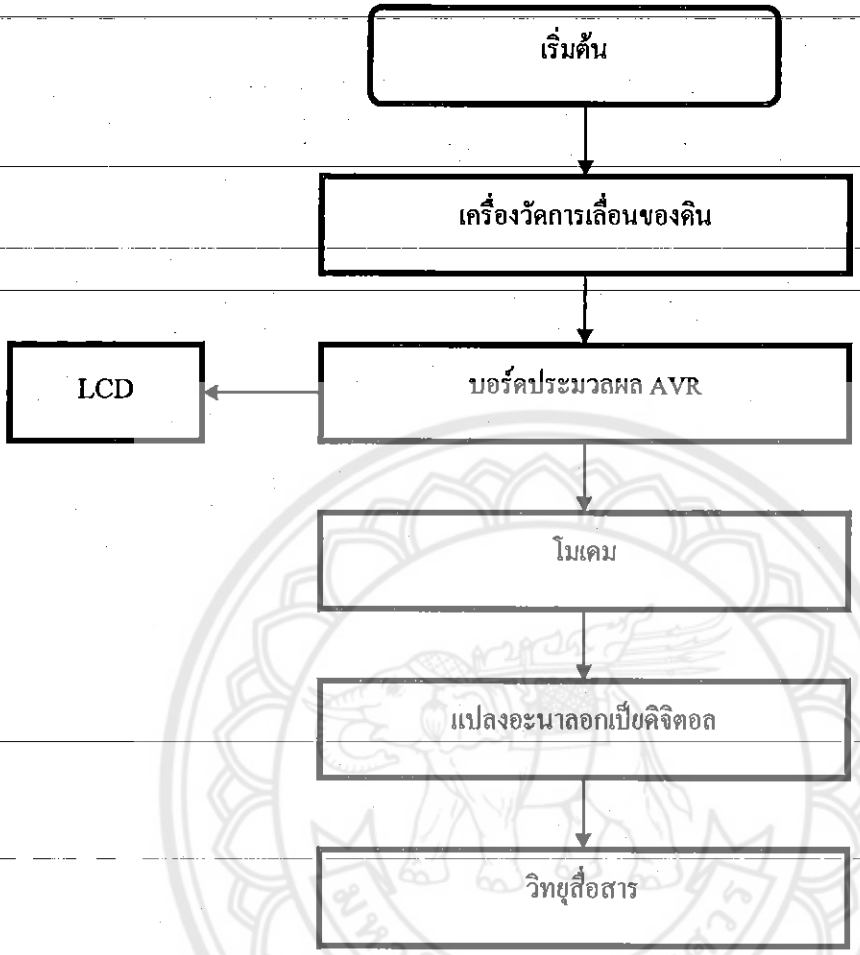
1. ตัวต้านทานปรับค่าได้
2. โซ่
3. เฟื่อง
4. แกนเหล็ก

3.2.6 หลักการทำงาน

หลักการทำงานของเครื่องวัดการเลื่อนของดิน คือ เมื่อสายโซ่ถูกดึงออกไปทำให้เกิดการหมุนของตัวต้านทานปรับค่าได้ทำให้เกิดแรงดัน ซึ่งค่าความต้านทานจะแปรผกผันกับแรงดัน เมื่อได้ค่าแรงดันแล้ว ค่าแรงดันจะถูกส่งไปยังภาคส่งและประมวลผลต่อไป

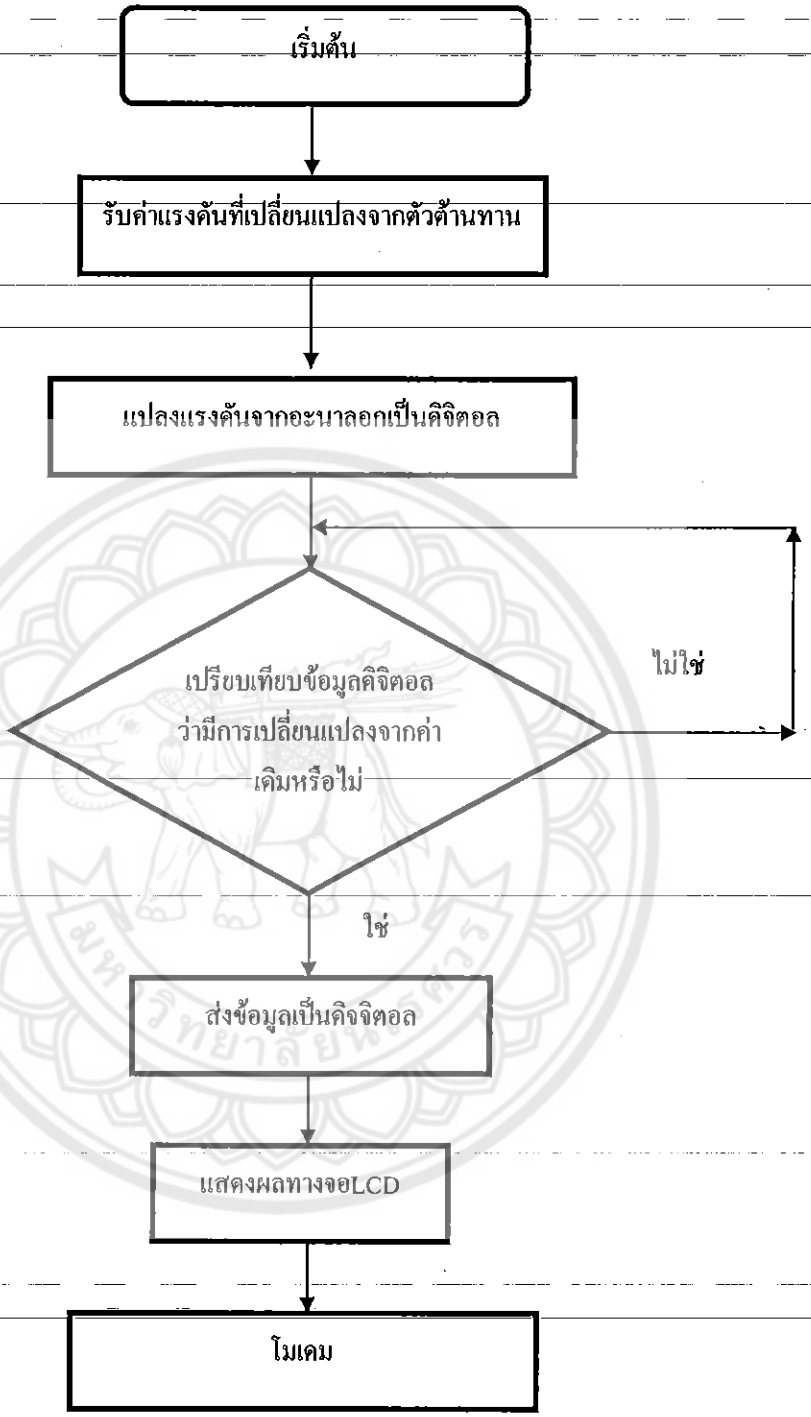
3.3 หลักการทำงานของภาคส่งและประมวลผล

3.3.1 บล็อกไดอะแกรมภาคส่ง



รูป 3.6 บล็อกไดอะแกรมภาคส่ง

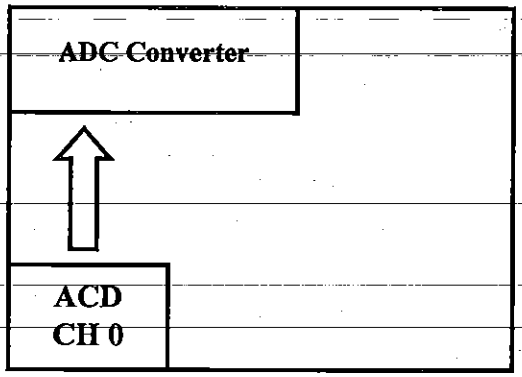
3.3.2 ส่วนของโปรแกรมของบอร์ดไมโครคอนโทรลเลอร์ AVR ภาคส่ง



รูป 3.7 การทำงานของไมโครคอนโทรลเลอร์ภาคส่ง

3.3.3 การออกแบบ

ส่วนการแปลงอะนาลอกเป็นดิจิตอล



รูปที่ 3.8 ภาพการแปลงอะนาลอกเป็นดิจิตอล

การคำนวณหาค่า ADC

จาก $$ADC = \frac{V_{IN} * 1024}{V_{REF}}$$

โดย ADC คือ ค่าข้อมูลบิต

Vin คือ แรงดันด้านขาอินพุต

Vref คือ แรงดันอ้างอิง มีค่าเท่ากับ 2.8 V จากการออกแบบวงจร

$$ADC = \frac{V_{IN} * 1024}{V_{REF}} = \frac{4.5 * 1024}{2.8} = 1645.7142$$

ดังนั้น เราจะได้ความต่างศักย์ 1 bit เท่ากับ

$$Volt / bit = \frac{Vcc}{ADC} = \frac{4.5}{1645.7142} = 0.002734 \text{ V}$$

ดังนั้นเมื่อคำนวณจะได้ออกมาดังตาราง 3.2

ระยะทาง (cm)	ข้อมูลบิต	แรงดัน ADC
0	0 - 6	0.00 - 0.016404
1	7 - 11	0.019138 - 0.030074
2	12 - 18	0.032808 - 0.049212
3	19 - 37	0.051946 - 0.101158
4	38 - 53	0.103892 - 0.144902
5	54 - 59	0.147636 - 0.161306
6	60 - 76	0.16404 - 0.207784
7	77 - 93	0.210518 - 0.254262
8	94 - 108	0.256996 - 0.295272
9	109 - 122	0.298006 - 0.333548
10	123 - 136	0.336282 - 0.371824
11	137 - 161	0.374558 - 0.440174
12	162 - 173	0.442908 - 0.472982
13	174 - 181	0.475716 - 0.494854
14	182 - 199	0.497588 - 0.544066
15	200 - 220	0.5468 - 0.60148
16	221 - 232	0.604214 - 0.634288
17	233 - 247	0.637022 - 0.675298
18	248 - 256	0.678032 - 0.699904
19	257 - 274	0.702638 - 0.749116
20	275 - 297	0.75185 - 0.811998
21	298 - 311	0.814732 - 0.850274
22	312 - 328	0.853008 - 0.896752
23	329 - 345	0.899486 - 0.94323
24	346 - 357	0.945964 - 0.976038
25	358 - 372	0.978772 - 1.017048

ตารางที่ 3.2 ตารางค่าข้อมูลบิตที่โปรมแกรมกำหนดเมื่อเทียบกับแรงดัน

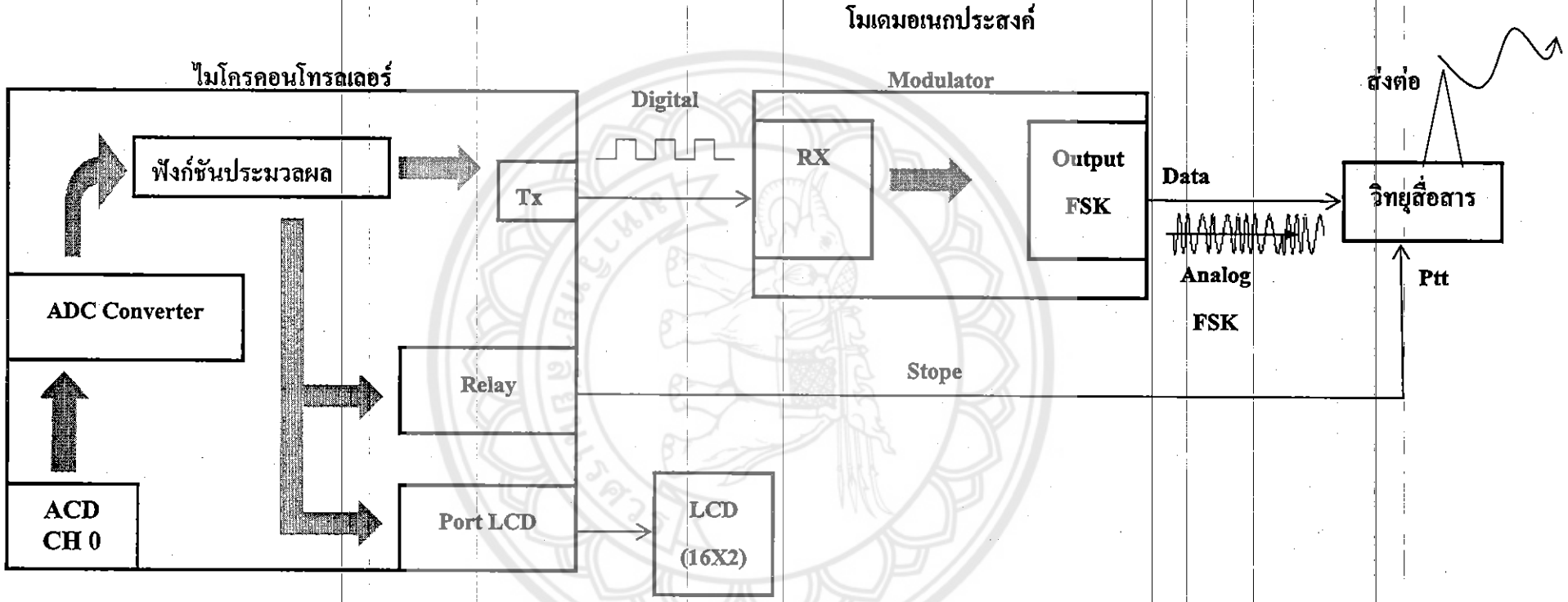
3.3.4 หลักการทำงาน

หลักการทำงานของตัววัดการเลื่อนของดิน เริ่มจากตัวต้านทานปรับค่าได้แบบละเอียด 500Ω (ขนาดการหมุน 20 รอบ) เกิดการเปลี่ยนแปลง ค่าความต่างศักดิ์ (V) ก็จะเปลี่ยนแปลงด้วย ค่าความต่างศักย์ (V) ที่เปลี่ยนแปลงจะเข้าสู่ไมโครคอนโทรลเลอร์เพื่อเปลียนข้อมูลจากอนาลอก เป็นดิจิตอลเพื่อเทียบออกมาอยู่ในค่าของบิตข้อมูลและทำการหาค่าเฉลี่ยออกมา และเทียบกับค่า ข้อมูลบิตที่กำหนดหนดไว้ในโปรแกรม และเปลี่ยนข้อมูลจากดิจิตอลเป็นอนาลอกเพื่อส่งไปยัง ตัวรับต่อไป

ระยะทาง(cm)	ข้อมูลบิต	ระยะทาง(cm)	ข้อมูลบิต
0	0-6	13	181
1	11	14	199
2	18	15	220
3	37	16	232
4	53	17	247
5	59	18	256
6	76	19	274
7	93	20	297
8	108	21	311
9	122	22	328
10	136	23	345
11	161	24	357
12	173	25	372

ตารางที่ 3.3 ตารางค่าข้อมูลบิตที่โปรมแกรมกำหนดเพื่อใช้ในการเปรียบเทียบข้อมูลบิต

3.3.5 การออกแบบส่วนไมโครคอนโทรลเลอร์(ภาคส่ง)ละภาคเอาท์พุท



รูปที่3.9 การทำงานของไมโครคอนโทรลเลอร์ ภาคส่ง

3.3.6 อุปกรณ์ไมโครคอนโทรลเลอร์ ภาคส่ง

ประกอบด้วยอุปกรณ์และหลักการทำงานดังต่อไปนี้

1. แผงควบคุมไมโครคอนโทรลเลอร์ ATmega 32

2. โมเดม

3. วิทยุสื่อสาร

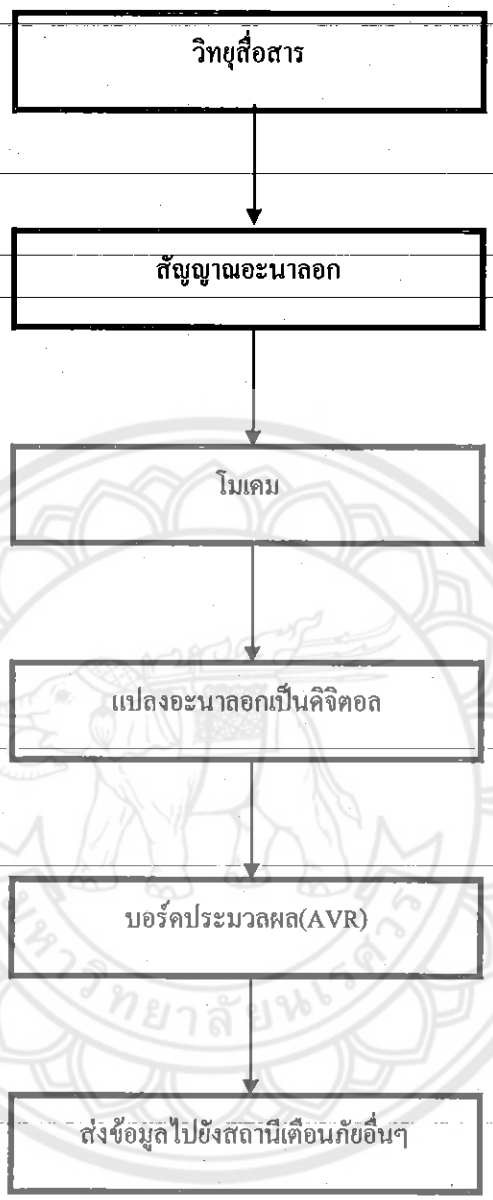
4. จอ LCD

3.3.7 หลักการทำงาน

เมื่อมีการเปลี่ยนแปลงของตัวต้านทานแล้ว ก็จะเกิดแรงดันขึ้นแรงดันนี้จะถูกส่งมายัง Port ACD CH0 และถูกส่งไปยัง ADC Converter เพื่อทำการเปลี่ยนค่าแรงดันให้เป็นข้อมูลดิจิตอล ข้อมูลนี้ก็จะถูกส่งปยัง หน่วยประมวลผลเพื่อทำการประมวล ว่ามีการเปลี่ยนแปลงหรือไม่เมื่อมีการเปลี่ยนถึงจุดที่กำหนดไว้ข้อมูลนี้ก็จะถึงส่งไปยัง port Tx เพื่อส่งข้อมูลไปยังโมเดมผ่าน port Rx โมเดมก็จะทำการ Modulator สัญญาณจากดิจิตอลเป็นอะนาลอกในรูปแบบ FSK สัญญาณจะออกทาง port out เข้าไปยัง port Ptt ในวิทยุสื่อสารเพื่อส่งไปยังภาครับ โดยมี Relay เป็นตัวกำหนดการส่งข้อมูลและข้อมูลที่มาจากหน่วยประมวลผลก็ยังแสดงผลบนจอ LCD ผ่าน port LCD16X2

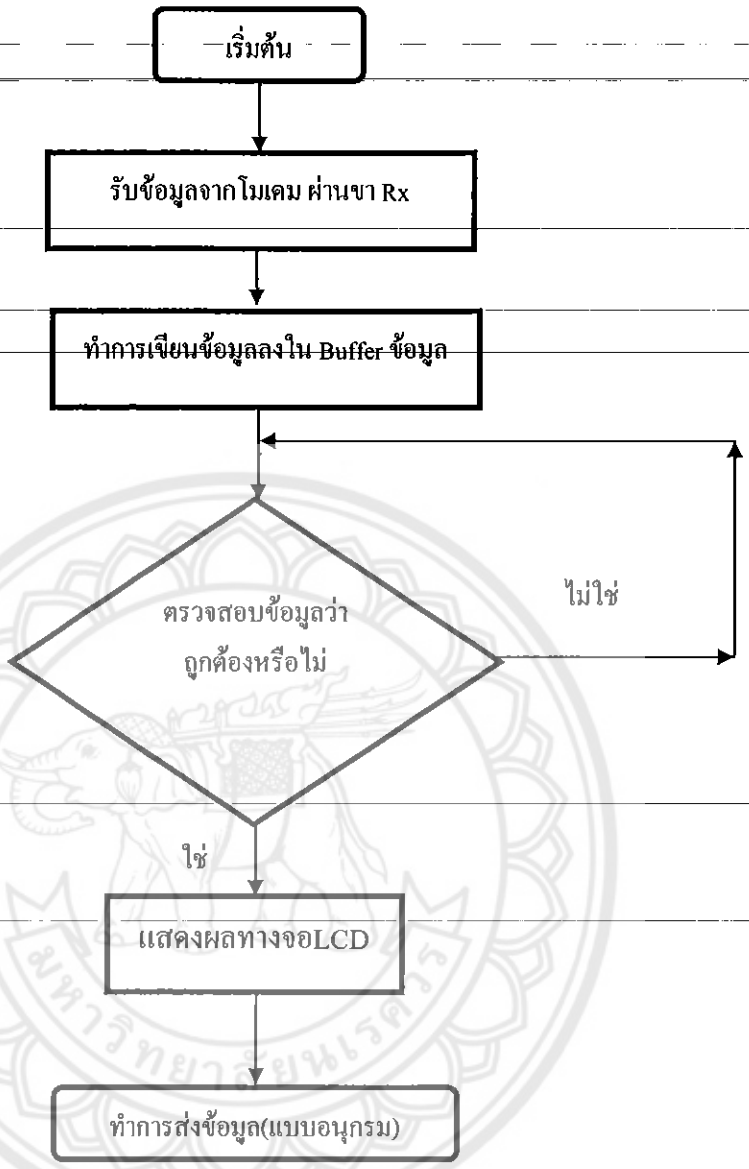
3.4 หลักการทำงานของภาครับข้อมูล

3.4.1 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์ ภาครับ



รูปที่ 3.10 บล็อกไดอะแกรมไมโครคอนโทรลเลอร์ ภาครับ

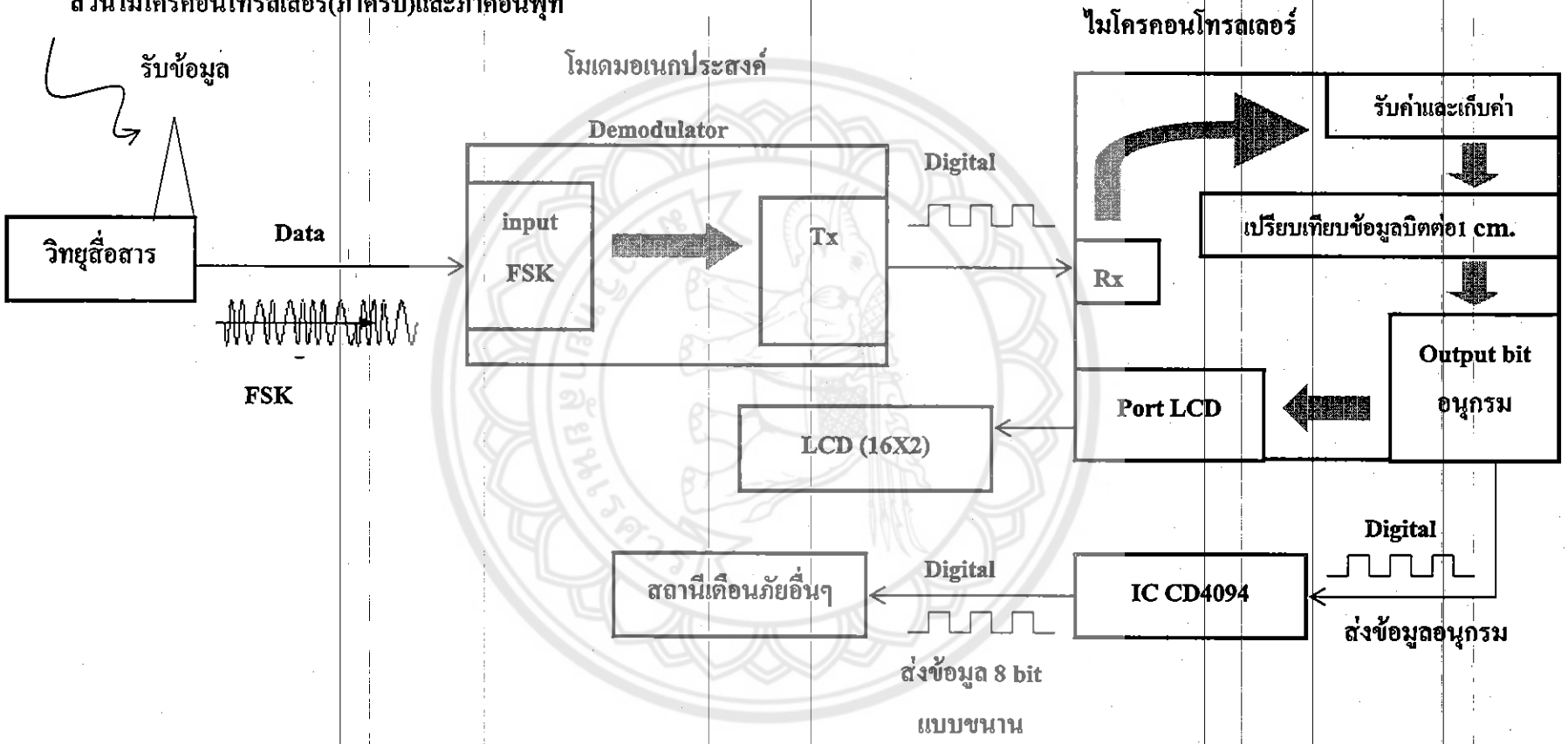
3.4.2 ส่วนของโปรแกรมของบอร์ดไมโครคอนโทรลเลอร์ AVR ภาครับ



รูปที่ 3.11 ส่วนของโปรแกรมของบอร์ดไมโครคอนโทรลเลอร์ AVR ภาครับ

3.4.3 การออกแบบ

ส่วนไมโครคอนโทรลเลอร์(ภาครับ)และภาคอินพุท



รูปที่3.12 การทำงานของไมโครคอนโทรลเลอร์ ภาครับ

3.4.4 อุปกรณ์ไมโครคอนโทรลเลอร์ ภาครับ

ประกอบด้วยอุปกรณ์และหลักการทำงานดังต่อไปนี้

1. แผงควบคุมไมโครคอนโทรลเลอร์ ATmega 32

2. โมเดม

3. วิทยุสื่อสารสื่อสาร

4. จอ LCD

3.4.5 หลักการทำงาน

เมื่อวิทยุได้รับสัญญาณอะนาลอกที่ภาคส่งส่งมาแล้ว สัญญาณนี้จะถูกส่งเข้าไปยัง

โมเดมผ่าน port input เพื่อให้โมเดมทำการ Demodulator จากสัญญาณอะนาลอกเป็นดิจิตอลและ

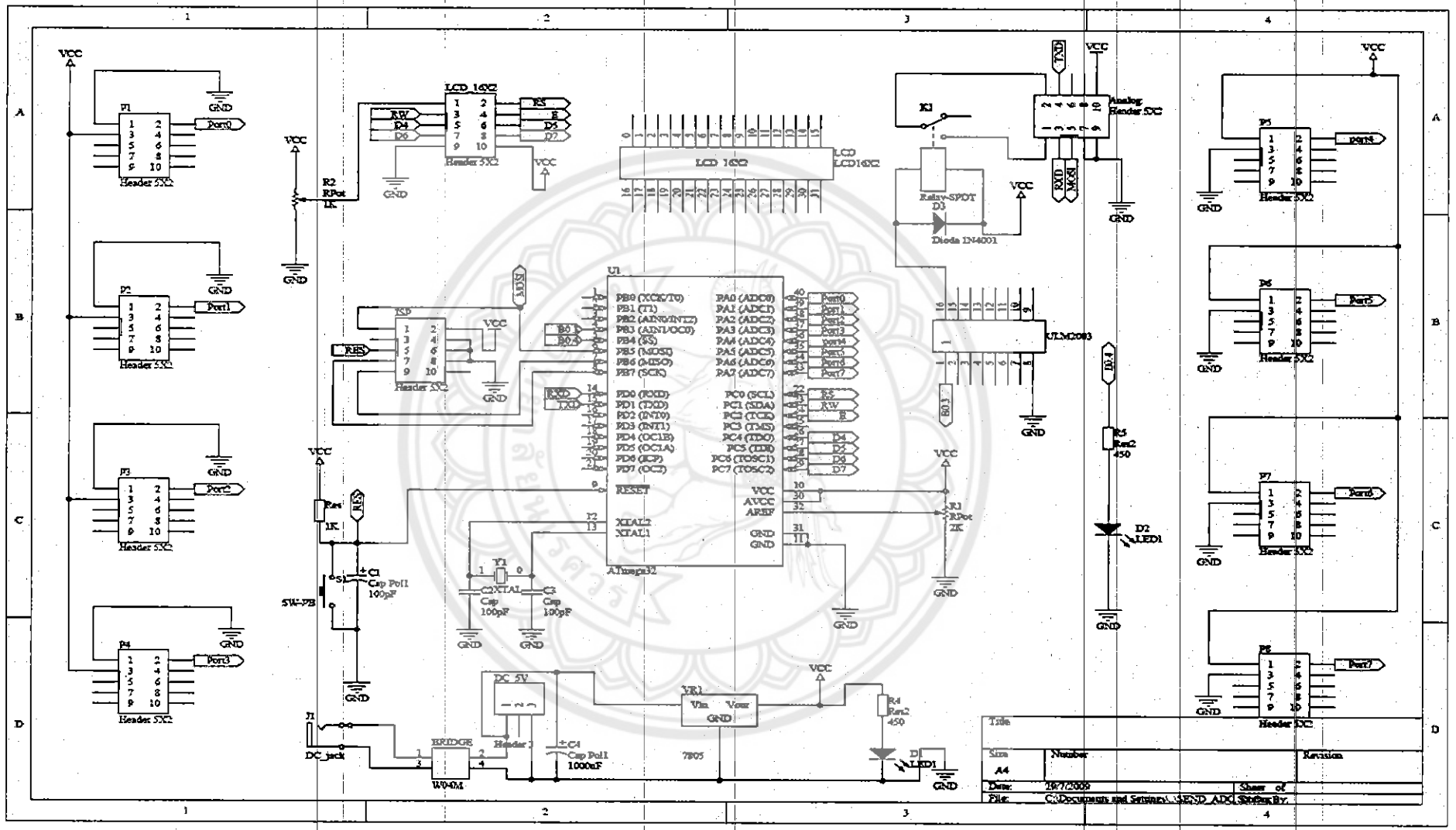
ส่งออกผ่านทาง port Tx เข้าไปยังไมโครคอนโทรลเลอร์ทาง port Rx สัญญาณดิจิตอลที่ได้จะถูก

นำไปเก็บไว้และนำไปเปรียบเทียบสัญญาณบิตต่อ 1 ซม.เมื่อข้อมูลถูกต้องแล้ว ข้อมูลที่ได้จะถูก

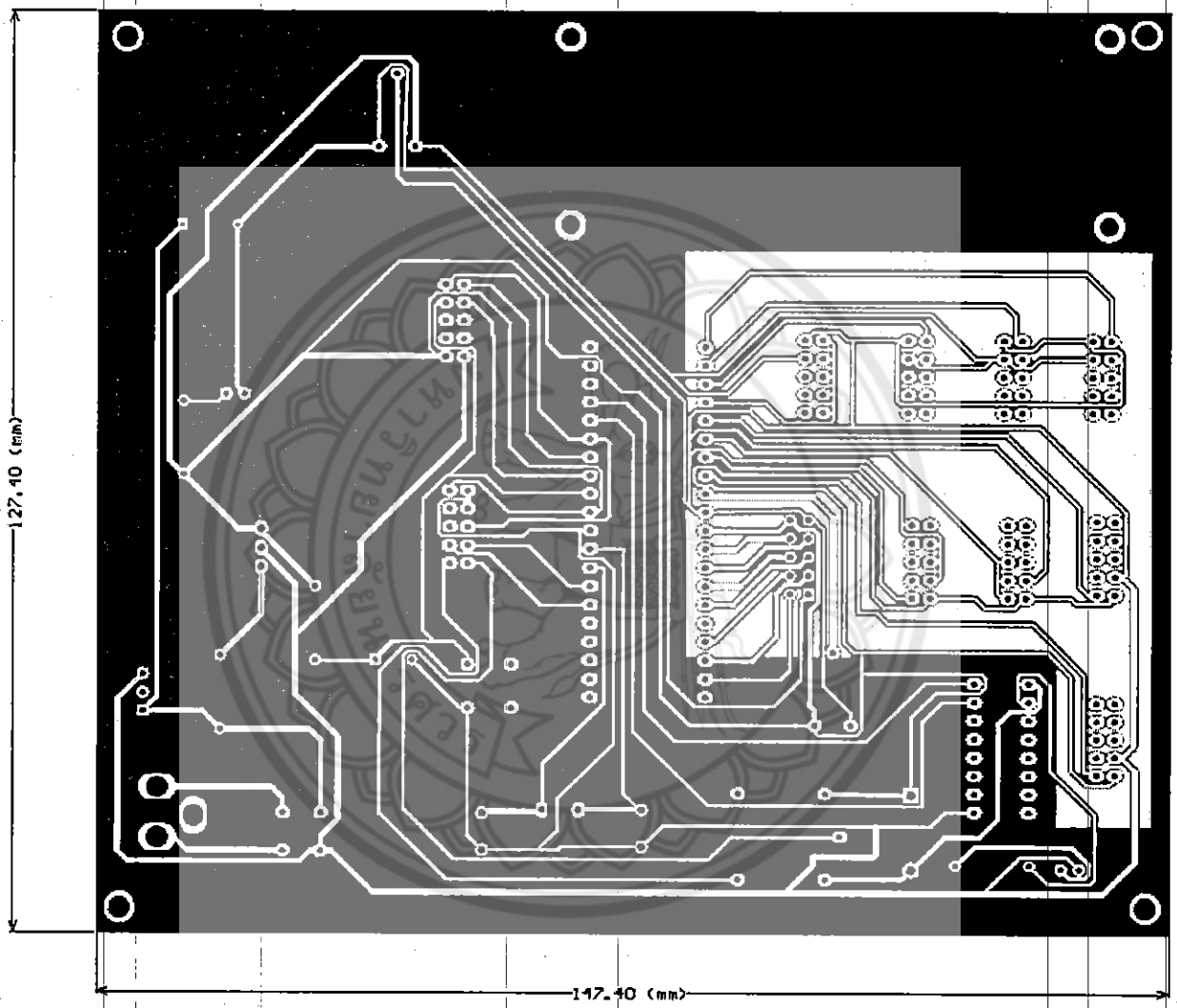
ส่งไปยัง output bit อนุกรม และแสดงผลทางจอ LCD ข้อมูลที่ได้จาก output bit อนุกรมจะถูกส่งไป

ยัง IC CD4094 เพื่อทำการเป็นจาก output bit อนุกรม เป็น output bit ขนาน เพื่อส่งต่อไปยังสถานี

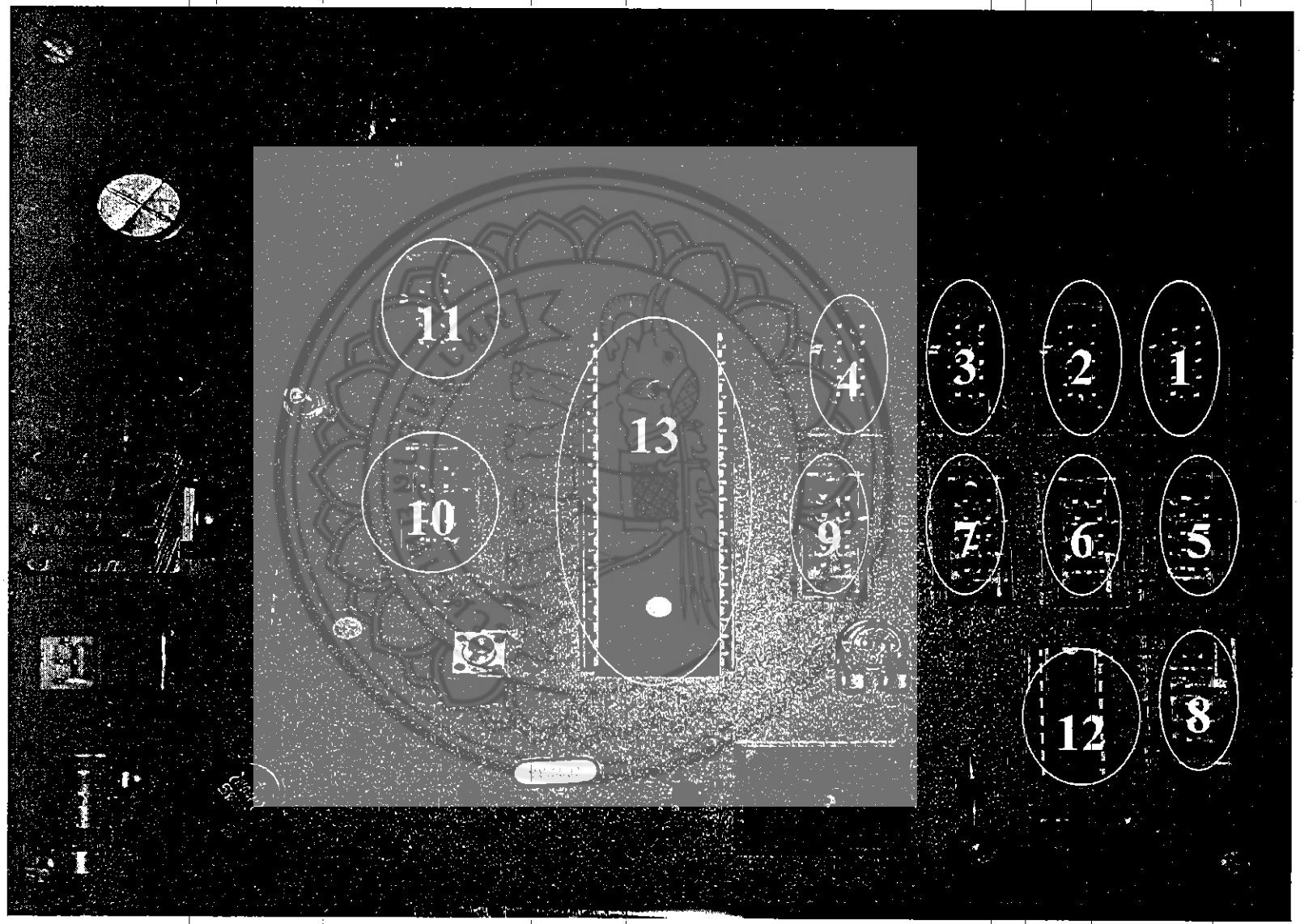
เตือนภัยอื่นๆ



รูปที่ 3.13 วงจรเครื่องมือวัดการเลื่อนของดินจากการออกแบบ



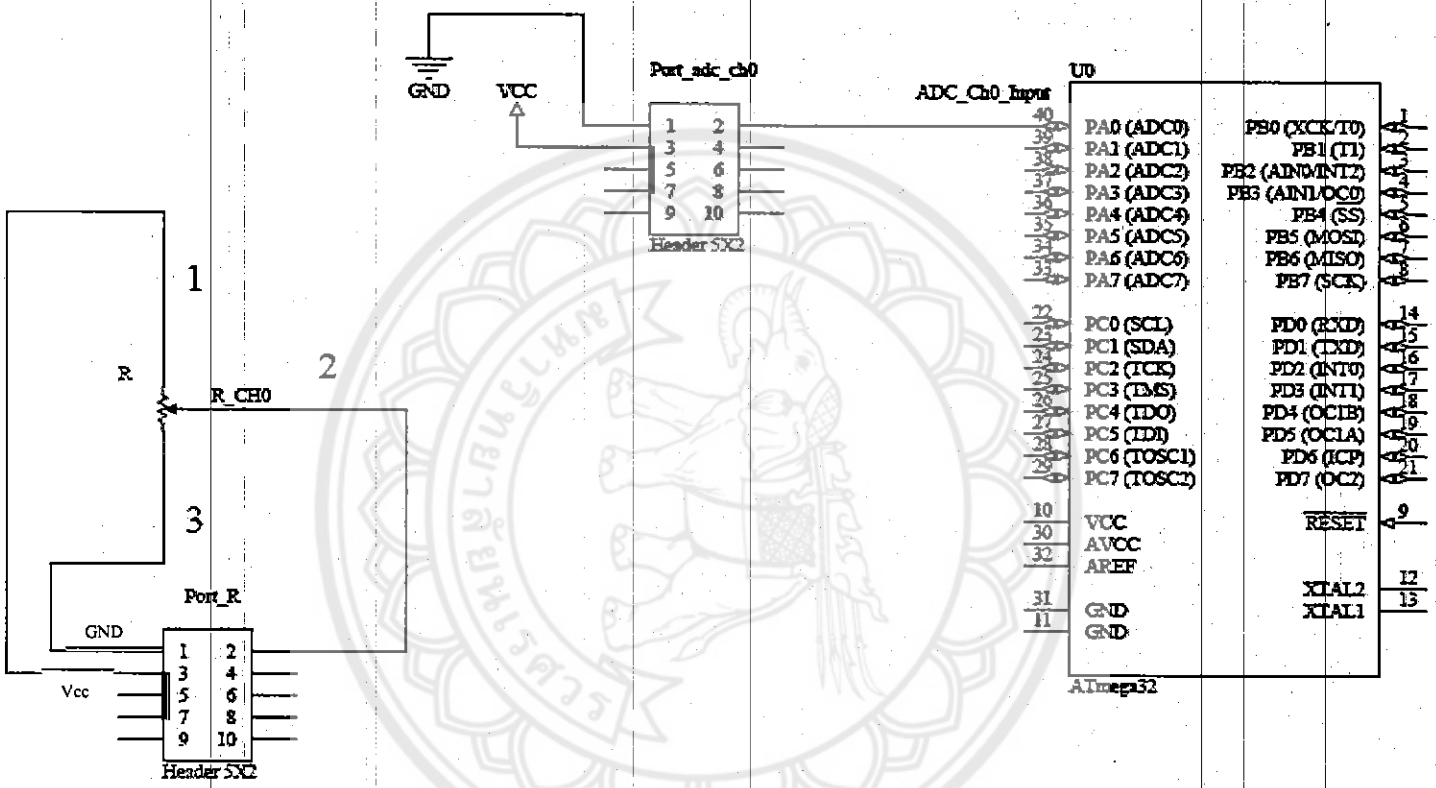
รูปที่ 3.14วงจรเครื่องมือวัดการเลื่อนของดิน(PCB)



รูปที่ 3.15 วงจรเครื่องมือวัดการเลื่อนของดิน

จากรูปที่ 3.15 สามารถอธิบายส่วนต่างๆได้ดังนี้

1. พอร์ท Channel 0 (A/D), [Port0] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้
2. พอร์ท Channel 1 (A/D), [Port1] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้
3. พอร์ท Channel 2 (A/D), [Port2] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้
4. พอร์ท Channel 3 (A/D), [Port3] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้
5. พอร์ท Channel 4 (A/D), [Port4] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้ เป็นพอร์ทสำรองเมื่อพอร์ท 0-3 เกิดเสียหาย
6. พอร์ท Channel 5 (A/D), [Port5] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้ เป็นพอร์ทสำรองเมื่อพอร์ท 0-3 เกิดเสียหาย
7. พอร์ท Channel 6 (A/D), [Port6] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้ เป็นพอร์ทสำรองเมื่อพอร์ท 0-3 เกิดเสียหาย
8. พอร์ท Channel 7 (A/D), [Port7] เป็นพอร์ที่รับข้อมูลมาจากตัวต้านทานปรับค่าได้ เป็นพอร์ทสำรองเมื่อพอร์ท 0-3 เกิดเสียหาย
9. พอร์ท (Output D/A) เป็นพอร์ทที่แปลงข้อมูลจากดิจิตอลเป็นอะนาลอกเพื่อส่งข้อมูลไปยังภาครับ
10. พอร์ท จอ LCD,[LCD_16X2] เป็นพอร์ทที่นำข้อมูลไปแสดงบนจอ LCD
11. พอร์ท ISP (In-System Programming),[Port ISP] เป็นพอร์ทที่ใช้ในการ Burn ข้อมูลจากโปรแกรมลงในไมโครคอนโทรลเลอร์
12. IC เบอร์ ULN2003 เป็นไอซีที่ใช้ในการดึงกระแสลงกราวน์เพื่อป้อนกระแสให้กับรีเลย์และลดความเสียหายของพอร์ทไมโครคอนโทรลเลอร์ ATmega32
13. ไมโครคอนโทรลเลอร์ ATmega32 เป็นหน่วยประมวลผลข้อมูลที่ได้รับจากพอร์ทชาแนลต่างๆ



รูปที่ 3.16 รูปแสดงการเชื่อมต่อระหว่าง ตัวต้านทานปรับค่าได้ขนาด 500 โอห์มรอบหมุน 20 รอบ (Port_R) กับบอร์ดไมโครคอนโทรลเลอร์ AVR (Port_adc_ch0)

การทำงานของขาตัวต้านทานปรับค่าได้แต่ละขา

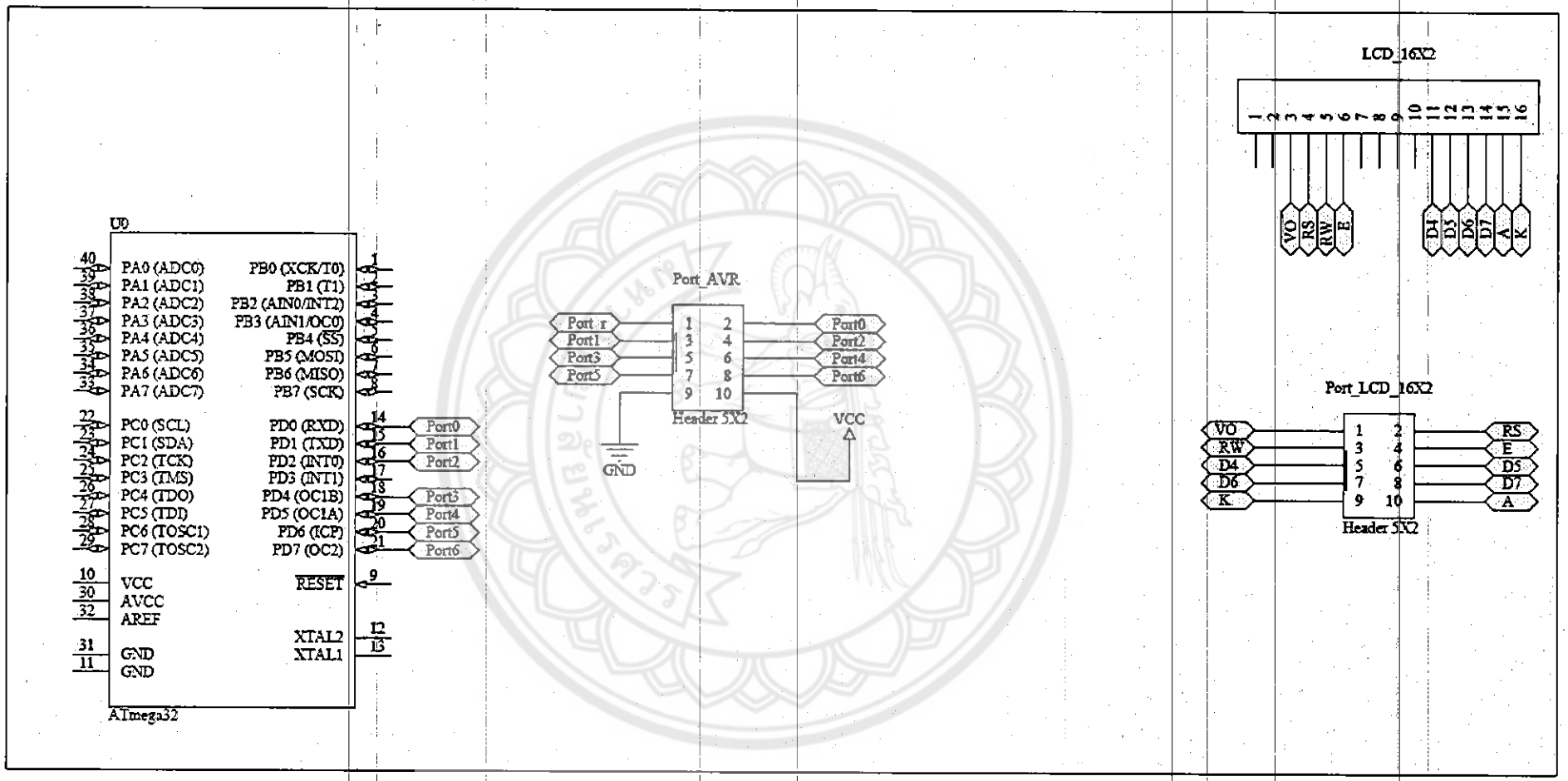
ขาอุปกรณ์ R	ขา Port	Port	คำอธิบาย Port	ขาไมโครคอนโทรลเลอร์
1	3	Vcc	ไฟเลี้ยง 4.5 V	-
2	2	adc_ch0	ขาแปลงอนาลอกเป็นดิจิตอล ch0	40(PA0)
3	1	GND	GND	-

ตารางที่ 3.4 การทำงานของขาตัวต้านทานปรับค่าได้แต่ละขา

*** ในกรณีที่ต้องการใช้ Port ข้อมูลการแปลงอะนาลอกเป็นดิจิตอลเพิ่มเติม จะต้องเพิ่ม Port ดังนี้

Port	คำอธิบาย Port	ขาไมโครคอนโทรลเลอร์
adc_ch1	ขาแปลงอนาลอกเป็นดิจิตอล ch1	39(PA1)
adc_ch2	ขาแปลงอนาลอกเป็นดิจิตอล ch2	38(PA2)
adc_ch3	ขาแปลงอนาลอกเป็นดิจิตอล ch3	37(PA3)
adc_ch4	ขาแปลงอนาลอกเป็นดิจิตอล ch4	36(PA4)
adc_ch5	ขาแปลงอนาลอกเป็นดิจิตอล ch5	35(PA5)
adc_ch6	ขาแปลงอนาลอกเป็นดิจิตอล ch6	34(PA6)
adc_ch7	ขาแปลงอนาลอกเป็นดิจิตอล ch7	33(PA7)

ตารางที่ 3.5 การทำงานในกรณีที่ต้องการใช้ Port ข้อมูลการแปลงอะนาลอกเป็นดิจิตอลเพิ่มเติม

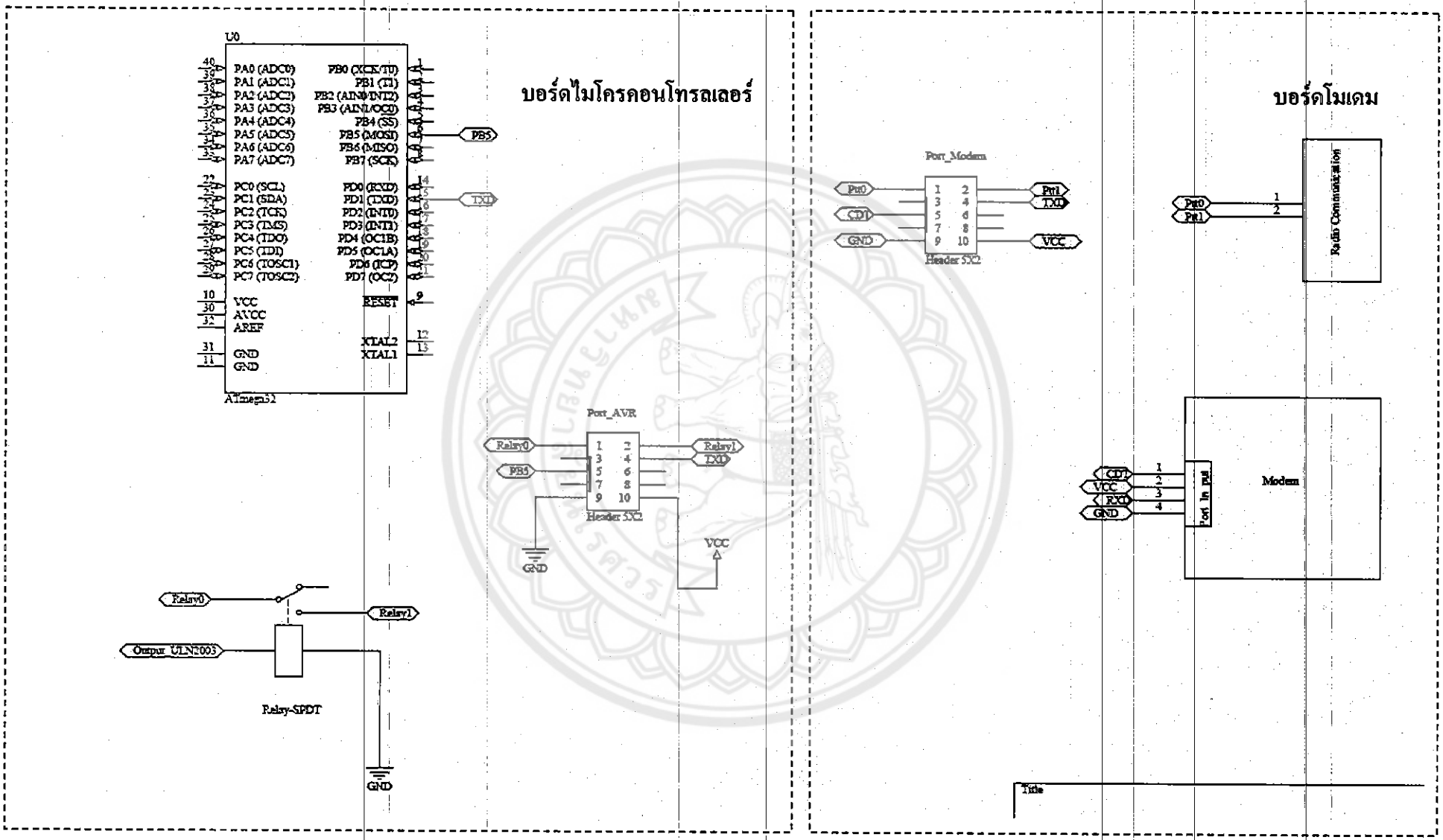


รูปที่ 3.17 แสดงการเชื่อมต่อระหว่าง Port_AVR กับ Por_LCD_16X2 ของภาคส่ง

การทำงานของขาอุปกรณ์ LCD แต่ละขา

ขาอุปกรณ์ LCD	ขา Port	Port	คำอธิบาย Port	ขาไมโครคอนโทรลเลอร์
1	-	-	-	-
2	-	-	-	-
3	1	VO	เป็นขาที่ใช้ปรับการแสดงผลของ LCD	-
4	2	RS	อ่านข้อมูลของ LCD	14
5	3	RW	เขียนข้อมูลของ LCD	15
6	4	E	ขาเปิด/ปิด ข้อมูล	16
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	-	-	-	-
11	5	D4	ขาข้อมูล	18
12	6	D5	ขาข้อมูล	19
13	7	D6	ขาข้อมูล	20
14	8	D7	ขาข้อมูล	21
15	10	A	Vcc ไฟเลี้ยง 4.5 V	-
16	9	K	GND	-

ตารางที่ 3.6 การทำงานของ LCD แต่ละขา

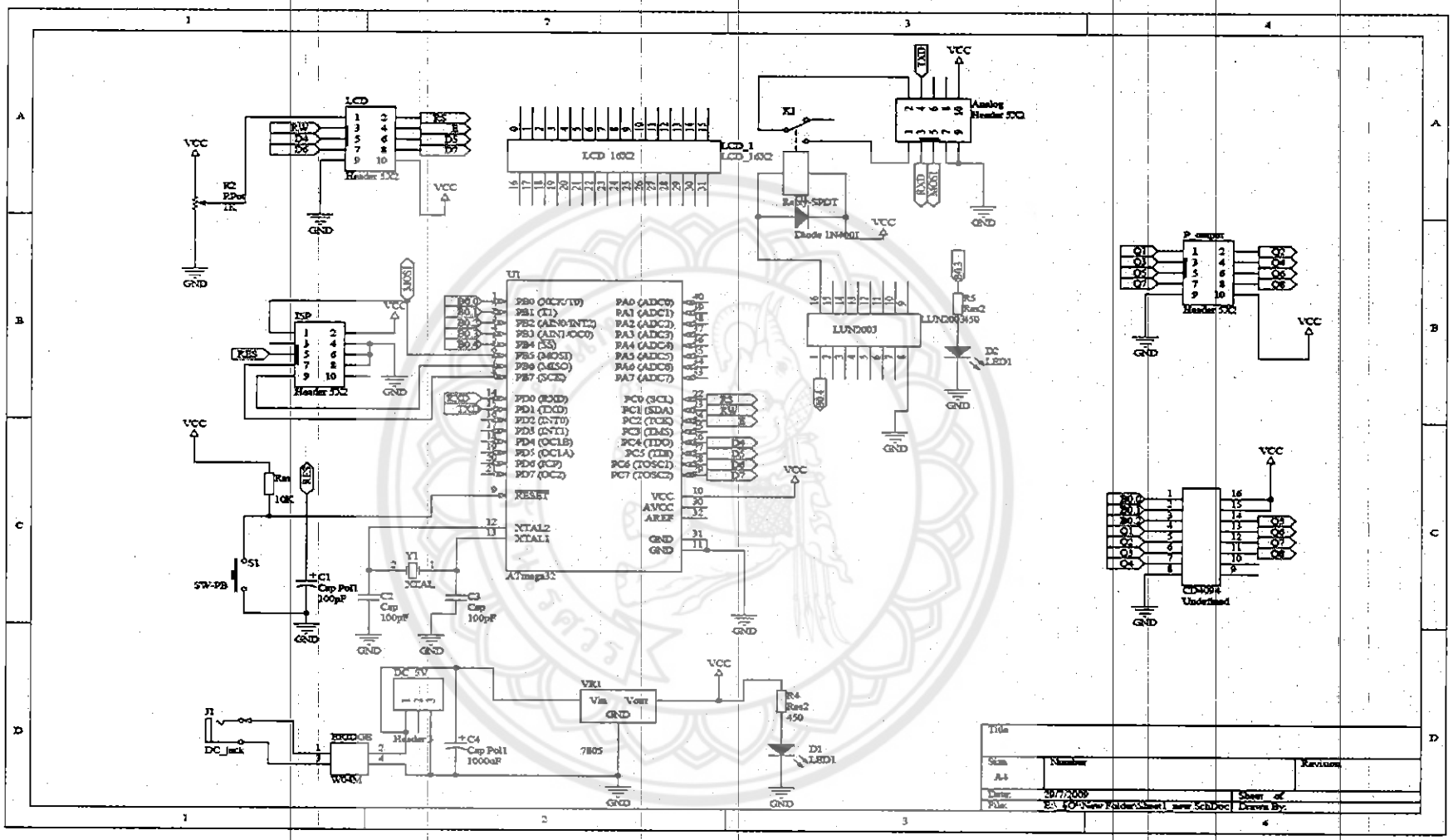


รูปที่ 3.18 การเชื่อมต่อระหว่างบอร์ดไมโครคอนโทรลเลอร์(Port_AVR) กับโมเดม (Port_Modem) ของภาคส่ง

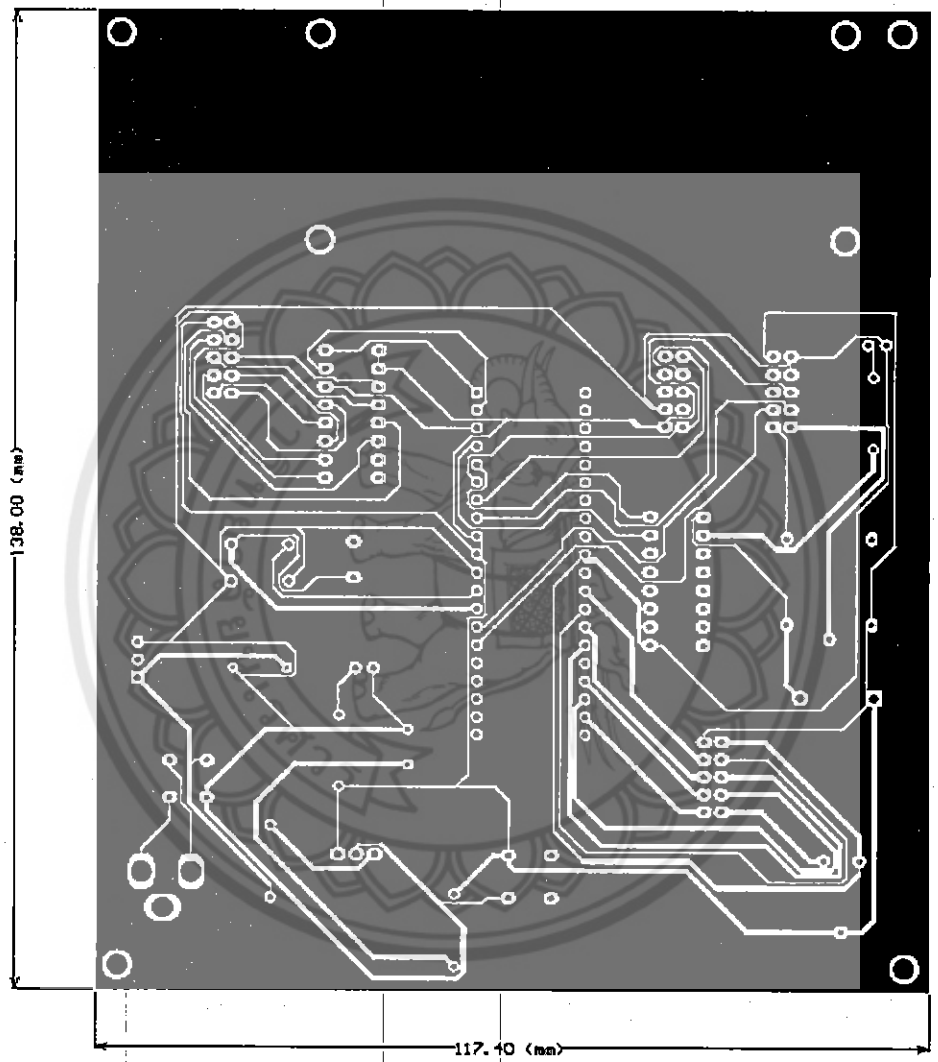
การทำงานของขาอุปกรณ์ โมเดม (ภาคส่ง) แต่ละขา

ขาบอร์ดโมเดม	ขา Port	Port_Modem	คำอธิบาย Port	ขา PPT วิทยุสื่อสาร	ขาไมโครคอนโทรลเลอร์
1	5	CDT	ขาตรวจสอบข้อมูล	-	6
2	10	Vcc	ไฟเลี้ยง 4.5 V	-	-
3	4	TXD	ขาส่งข้อมูล	-	15
4	9	GND	GND	-	-
-	1	Ptt0	ขาต่อกับ Relay เมื่อ On/Off Ptt	1	-
-	2	Ptt1	ขาต่อกับ Relay เมื่อ On/Off Ptt	2	-

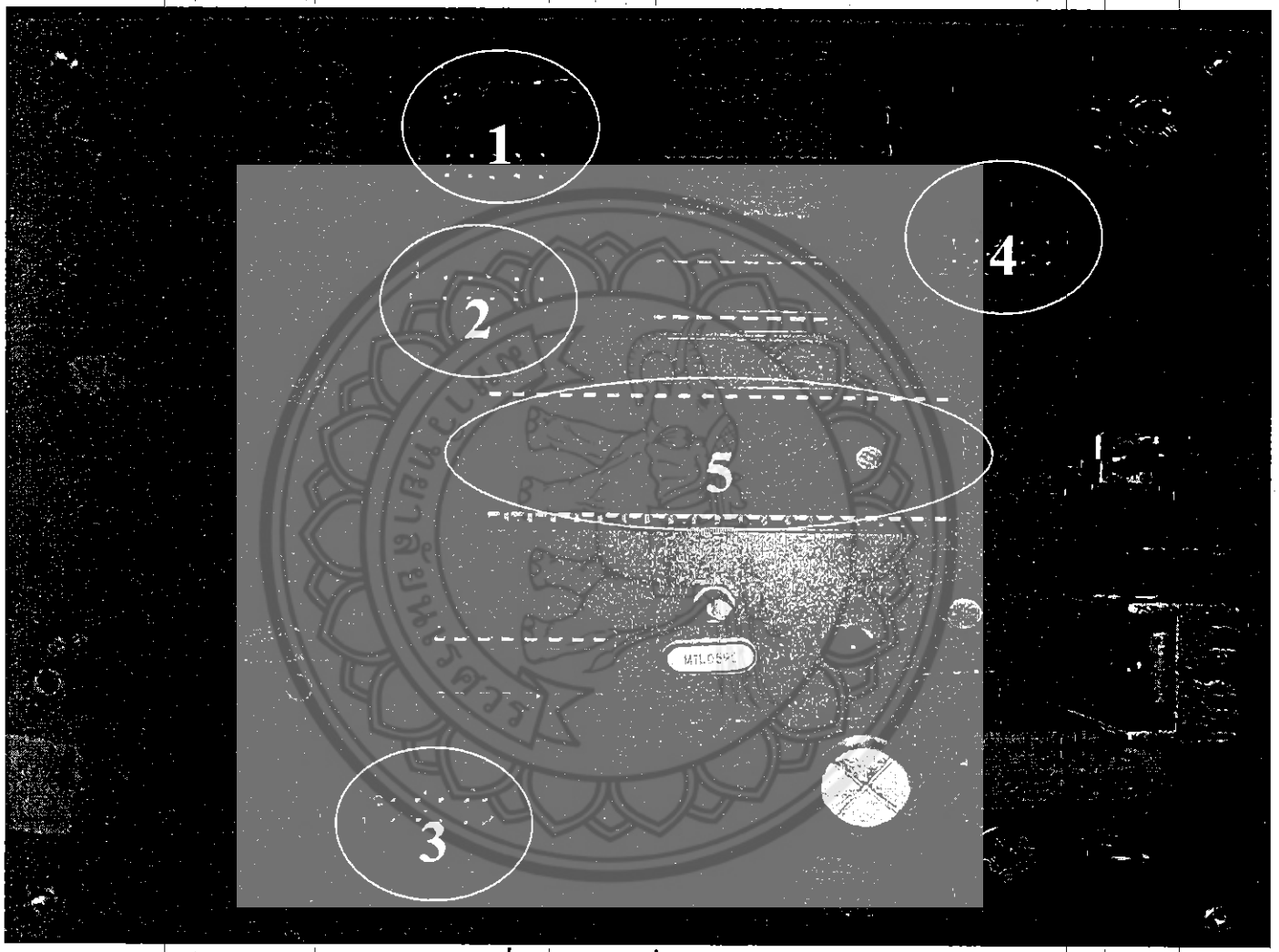
ตารางที่ 3.7 การทำงานของขาโมเดม(ภาคส่ง)แต่ละขา



รูปที่ 3.19 วงจรเครื่องรับสัญญาณที่ได้จากการออกแบบ



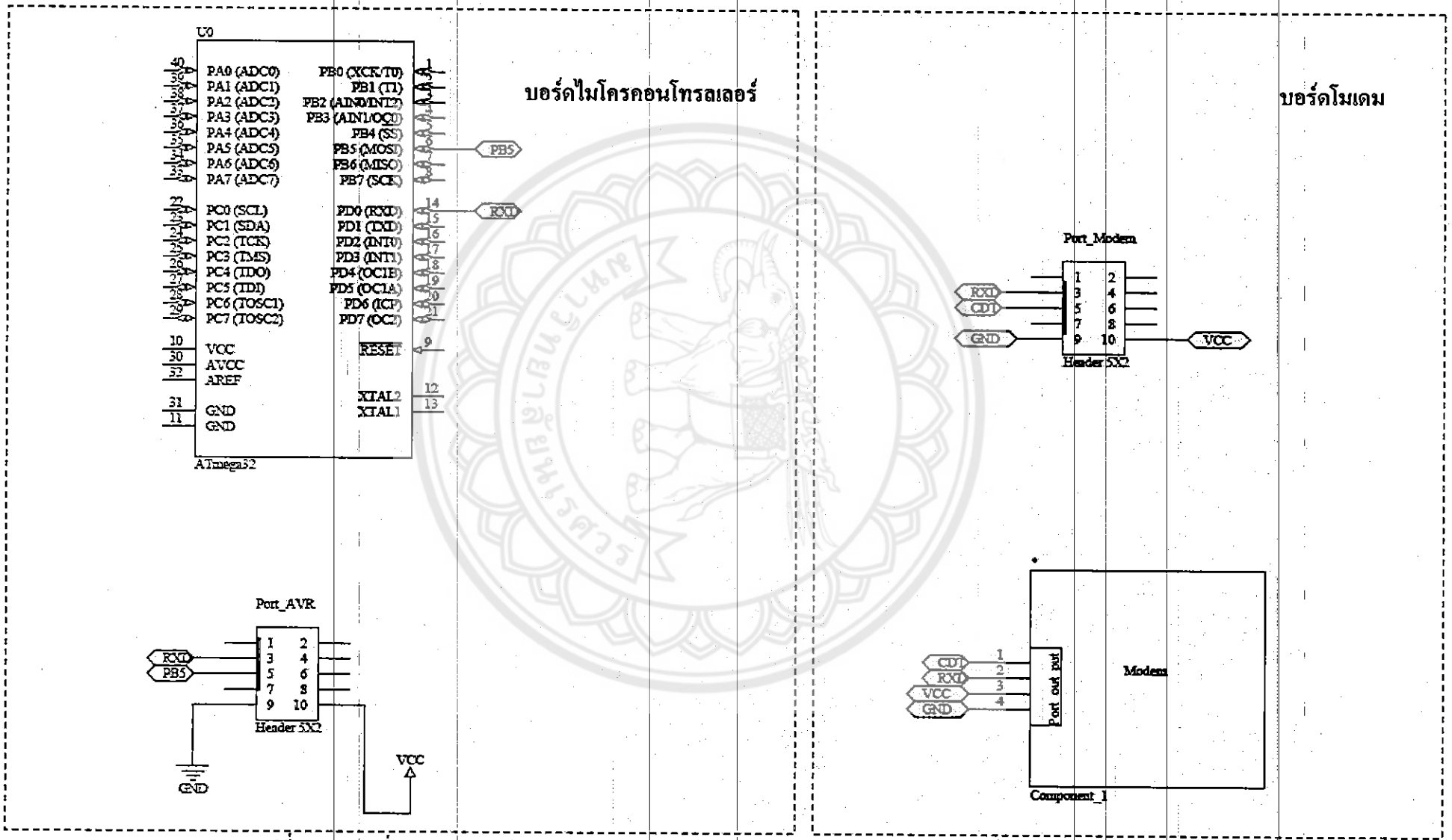
รูปที่ 3.20 วงจรเครื่องรับสัญญาณที่ได้จากการออกแบบ (PCB)



รูปที่ 3.21 วงจรเครื่องรับสัญญาณ

จากรูปที่ 3.21 สามารถอธิบายส่วนต่างๆได้ดังนี้

1. พอร์ท (input A/D),[Port Modem] เป็นพอร์ทที่แปลงข้อมูลจากอะนาลอกเป็นดิจิตอลเพื่อส่งข้อมูลไปยังไมโคคอนโทรลเลอร์
2. พอร์ท ISP (In-System Programming),[ISP] เป็นพอร์ทที่ใช้ในการ Burn ข้อมูลจากโปรแกรมลงในไมโครคอนโทรลเลอร์
3. พอร์ท output 8 bit,[P_output] เป็นพอร์ทที่ส่งข้อมูลแบบขนานขนาด 8 บิตไปยังสถานนีเตือนภัยอื่นๆ
4. พอร์ท จอ LCD,[LCD] เป็นพอร์ทที่เชื่อมต่อไปยังจอ LCD เพื่อแสดงผลข้อมูล
5. ไมโครคอนโทรลเลอร์ ATmega32 เป็นหน่วยประมวลผลข้อมูลที่ได้รับจากพอร์ทชาแนลต่างๆ



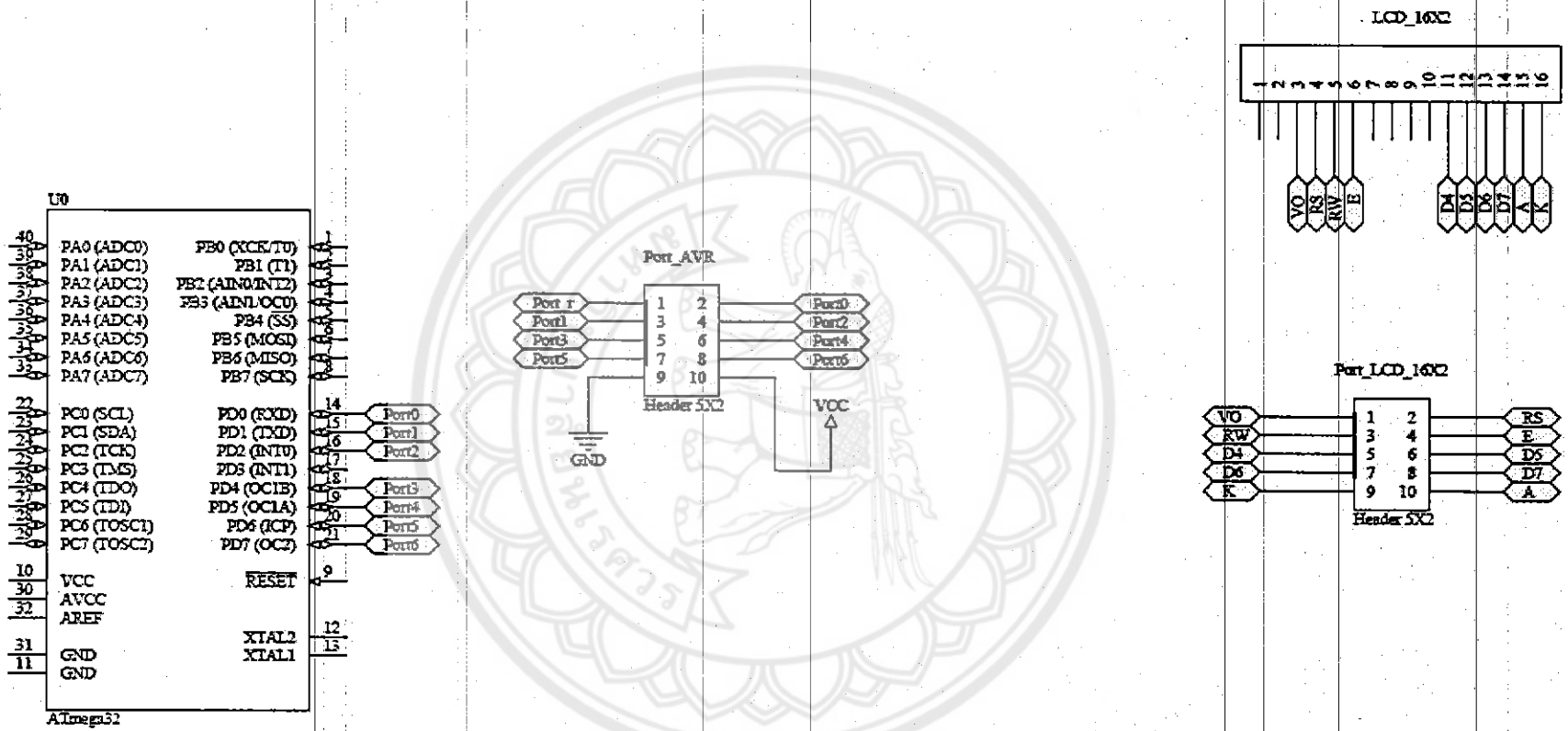
รูปที่ 3.22 การเชื่อมต่อระหว่างไมโครคอนโมรลเลอร์ AVR (Port_AVR) กับโมเดม (Port_Modem) ของภาครับ

การทำงานของขาอุปกรณ์ โมเดม (ภาครับ) แต่ละขา

ขาบอร์ดโมเดม	ขา Port	Port_Modem	คำอธิบาย Port	ขา PPT วิทยุสื่อสาร	ขาไมโครคอนโทรลเลอร์
1	5	CDT	ขาตรวจสอบข้อมูล	-	6
2	3	TXD	ขารับข้อมูล	-	14
3	10	Vcc	ไฟเลี้ยง 4.5 V	-	-
4	9	GND	GND	-	-

ตารางที่ 3.8 การทำงานของขาโมเดม(ภาครับ)แต่ละขา

มหาวิทยาลัยนเรศวร



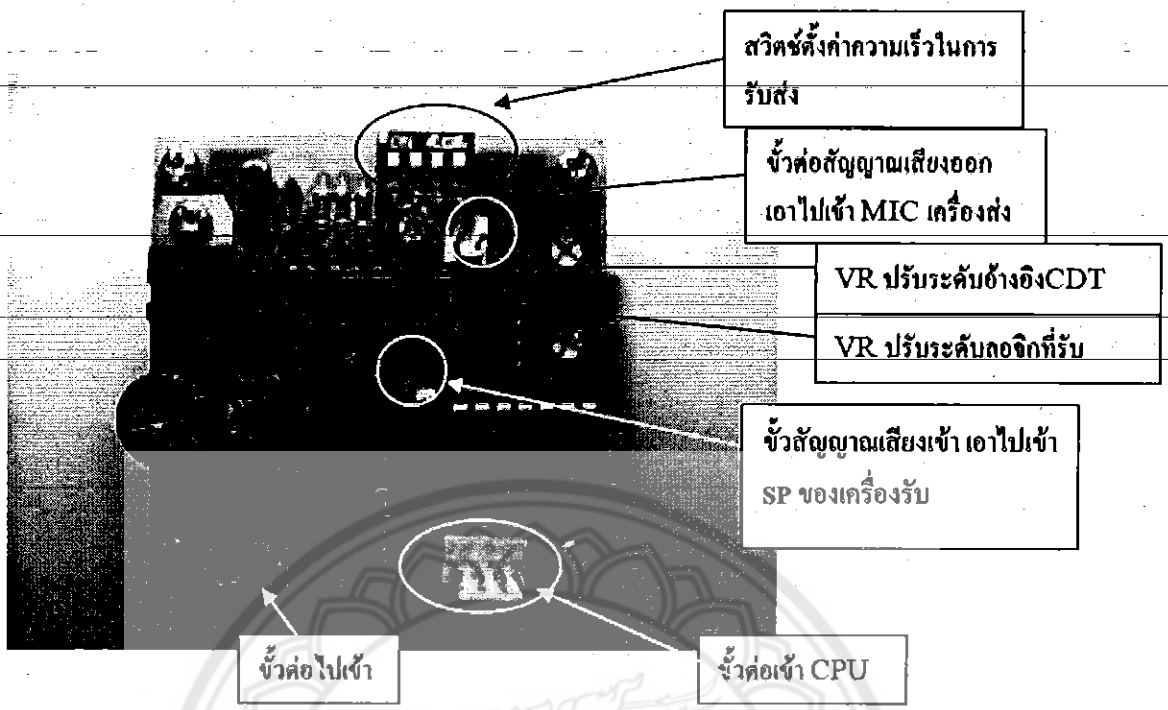
รูปที่ 3.23 การเชื่อมต่อระหว่างไมโครคอนโทรลเลอร์ (Port_AVR) กับ จอ LCD (Port_LCD_16X2)

การทำงานของขาอุปกรณ์ LCD แต่ละขา

ขาอุปกรณ์ LCD	ขา Port	Port	คำอธิบาย Port	ขาไมโครคอนโทรลเลอร์
1	-	-	-	-
2	-	-	-	-
3	1	VO	เป็นขาที่ใช้ปรับการแสดงผลของ LCD	-
4	2	RS	อ่านข้อมูลของ LCD	14
5	3	RW	เขียนข้อมูลของ LCD	15
6	4	E	ขาเปิด/ปิด ข้อมูล	16
7	-	-	-	-
8	-	-	-	-
9	-	-	-	-
10	-	-	-	-
11	5	D4	ขาข้อมูล	18
12	6	D5	ขาข้อมูล	19
13	7	D6	ขาข้อมูล	20
14	8	D7	ขาข้อมูล	21
15	10	A	Vcc ไฟเลี้ยง 4.5 V	-
16	9	K	GND	-

ตารางที่ 3.9 การทำงานของ LCD แต่ละขา

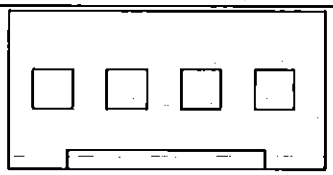
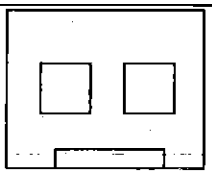
3.5 การทำงานของโมเดมอเนกประสงค์



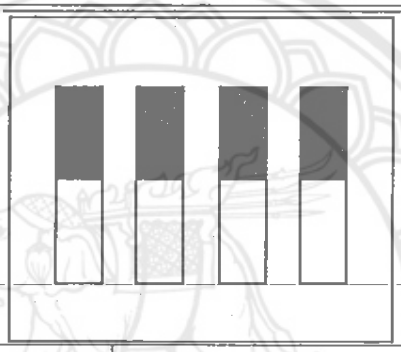
รูปที่ 3.24 โมเดมอเนกประสงค์

การใช้งานเริ่มจากต่อไฟเลี้ยงเข้าวงจรตั้งแต่ 6Vdc-15Vdc หรือ AC ตั้งโหมดความเร็วของวงจรด้วยการปรับสวิตช์ตำแหน่ง 2-4 ตามตาราง สวิตช์หมายเลข 1 เป็นสวิตช์ปิด,เปิด IC หลักที่ทำหน้าที่เป็นตัวมอดูเลทและดีมอดูเลท (เนื่องจาก ICมีความไวต่อไฟฟ้าสถิตมากและอาจเสียหายได้ง่ายเมื่อมีการต่อใช้งานที่ไม่ถูกวิธี เมื่อต้องการใช้งานวงจรให้เลื่อนสวิตช์ตำแหน่งที่ 1 ไปที่ ON ตามรูป ไฟจะเข้าไปเลี้ยง IC วงจรทำงานได้ จากนั้นต่อสัญญาณเสียงออกไปเข้าที่ช่อง MIC ของเครื่องส่งวิทยุ และต่อช่องสัญญาณเสียงเข้ามาจากลำโพงของเครื่องรับส่งวิทยุ ในกรณีที่ส่งสัญญาณทางเดียวด้านเครื่องส่งไม่ต้องต่อช่องสัญญาณเสียงเข้าก็ได้ และในกรณีเป็นภาครับอย่างเดียวไม่จำเป็นต้องต่อเสียงออกไปเข้าช่อง MIC ก็ได้ดูรูปประกอบ

เมื่อต่อวงจรและ Set ระดับแรงดันอ้างอิงแล้วก็สามารถนำสัญญาณจากไมโครคอนโทรเลอร์ต่อเข้าในช่องเสียบได้เลย เท่านี้ก็สามารถส่งสัญญาณดิจิตอลผ่านเครื่องวิทยุสื่อสารได้แล้ว

	
ช่องต่อสัญญาณไป cpu 1 ขาสัญญาณ CDT (ออก) 2 ขาสัญญาณ RX (ออก) 3 ขาสัญญาณ TX (เข้า) 4 ขาสัญญาณ GND	ช่องสัญญาณเสียงเข้าและออก 1 ขาสัญญาณเสียงเข้า,ออก 2 ขา GND

รูปที่ 3.25 รูปแสดงขาของช่องสัญญาณไป cpu และ ช่องสัญญาณเสียงเข้าและออก

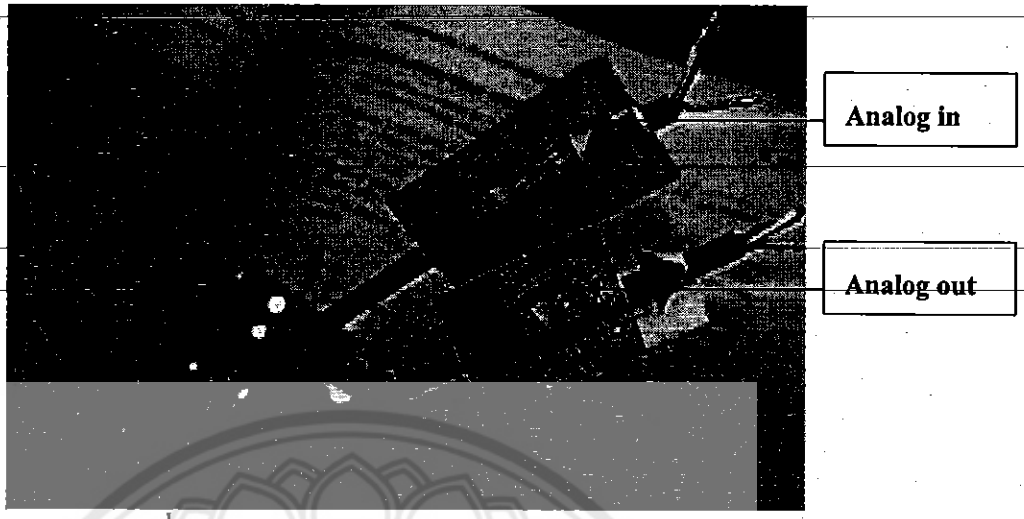


สวิทช์ตำแหน่งที่ 1 = Power IC			ON			
ตำแหน่ง Sw			ความเร็ว		ความถี่	
2	3	4	รับ	ส่ง	รับ	ส่ง
ON	ON	ON	1200	1200	M 1300	M 1300
					S 2100	S 2100

ตารางที่ 3.10 การทำงานของขาตัวต้านทานปรับค่าได้แต่ละขา

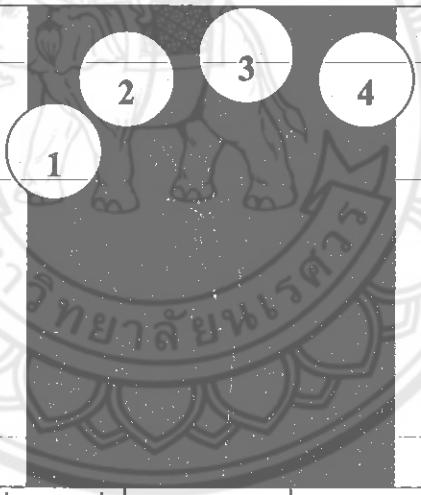
3.5.1 การเชื่อมต่อโมเดมกับวิทยุสื่อสาร

การเชื่อมต่อจากโดเดมไปยังวิทยุสื่อสารโดยใช้สาย Analog in และ Analog out ในการเชื่อมต่อ ดังรูป 3.26



รูปที่ 3.26 รูปแสดงสาย Analog in และ Analog out

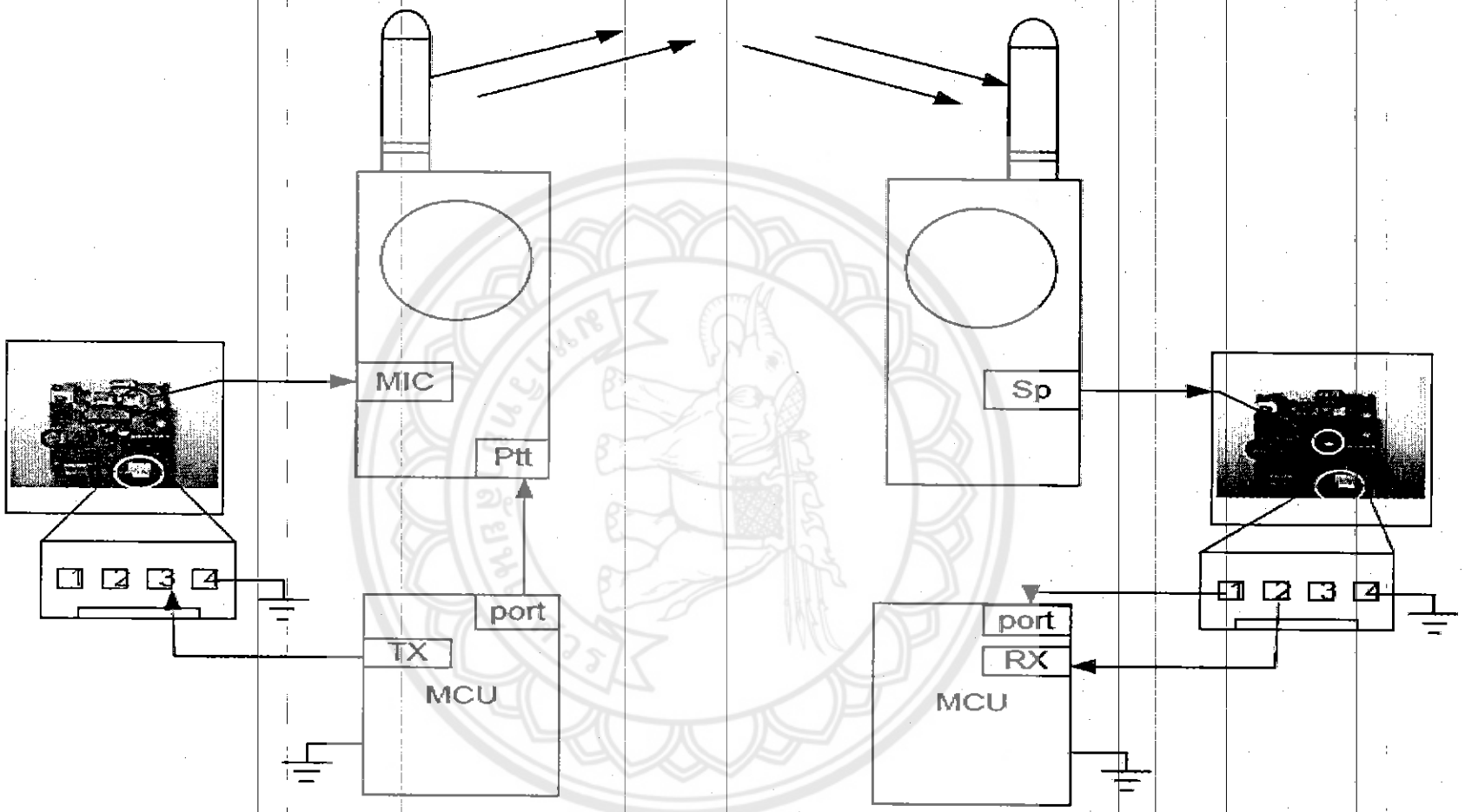
และสายวิทยุสื่อสารที่ใช้ในการเชื่อมต่อนั้นเราได้นำสายหูฟังของวิทยุมาตัดและแยกสายไฟฟ้าออกเป็น ดังตารางที่ 3.8



สีของสายหูฟังวิทยุ	ชื่อ port ที่ใช้งาน	Analog in	Analog out	Ptt
แดง(4)	MIC		×	
ทอง(1)	Ptt			×
เขียว(2)	SP	×		
น้ำเงิน(3)	GND	×	×	×

*** × แทนมีการเชื่อมต่อ

ตารางที่ 3.11 การทำงานของสายวิทยุสื่อสารแต่ละสาย



รูปที่ 3.27 ภาพหลักการต่อวงจรเพื่อใช้งานโมเดมเอนกประสงค์

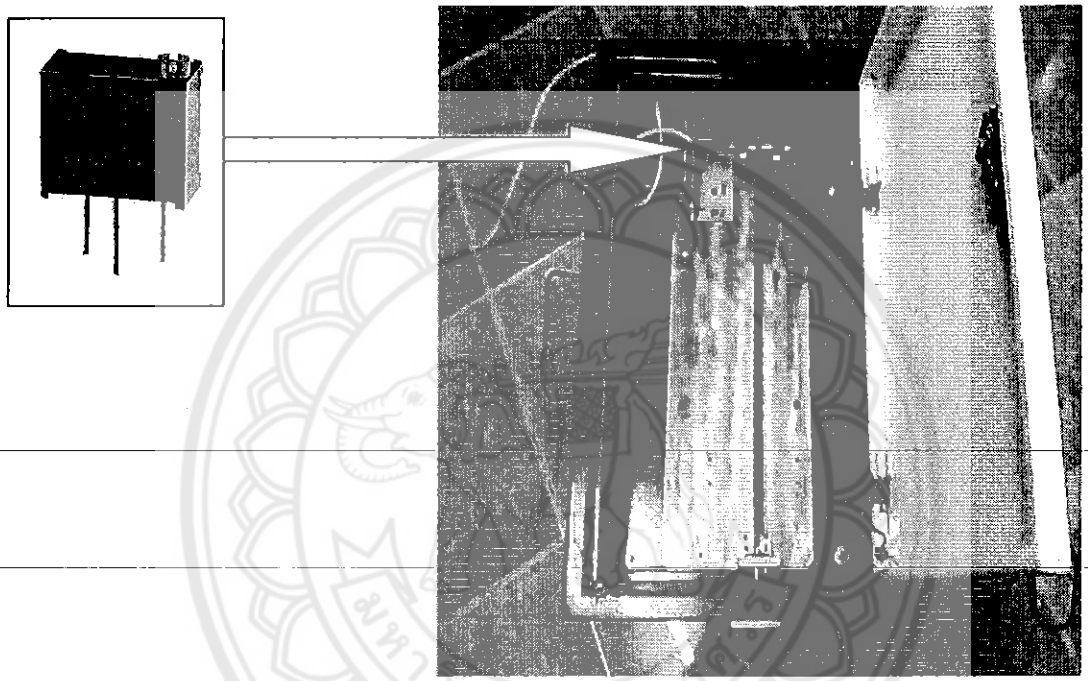
บทที่ 4

ผลการทดลอง

4.1 ระบบการทำงาน

ประกอบด้วย 3 ส่วนหลักๆคือ

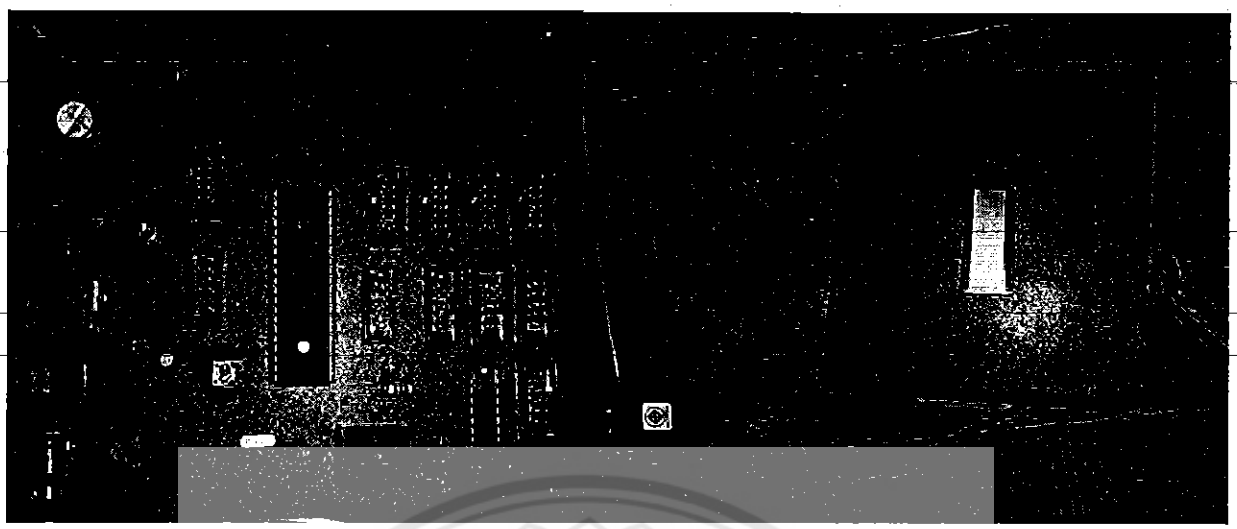
4.1.1.เครื่องวัดการเลื่อนของดิน



รูปที่ 4.1 เครื่องวัดการเลื่อนของดิน

ประกอบด้วยตัวต้านทานปรับค่าได้ขนาด 500 โอห์ม รอบหมุน 20 รอบ ซึ่งถูกต่อกับสายไฟฟ้าที่มีความต้านทานต่ำ แกนหมุนของตัวต้านทานถูกต่อเข้ากับแกนเหล็กซึ่งแกนเหล็กนี้จะเชื่อมอยู่กับเฟืองเหล็กและจะถูกถึงด้วยโซ่ ซึ่งโซ่นี้จะถูกดึงออกไปเมื่อมีการเลื่อนของดินเกิดขึ้น

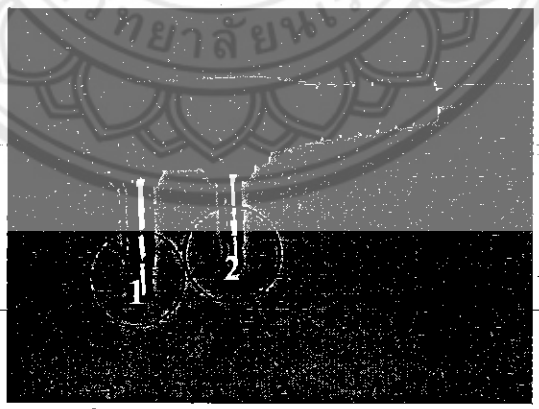
4.1.2. เครื่องส่งข้อมูลและประมวลผล



รูปที่ 4.2 ภาคส่งละประมวลผล

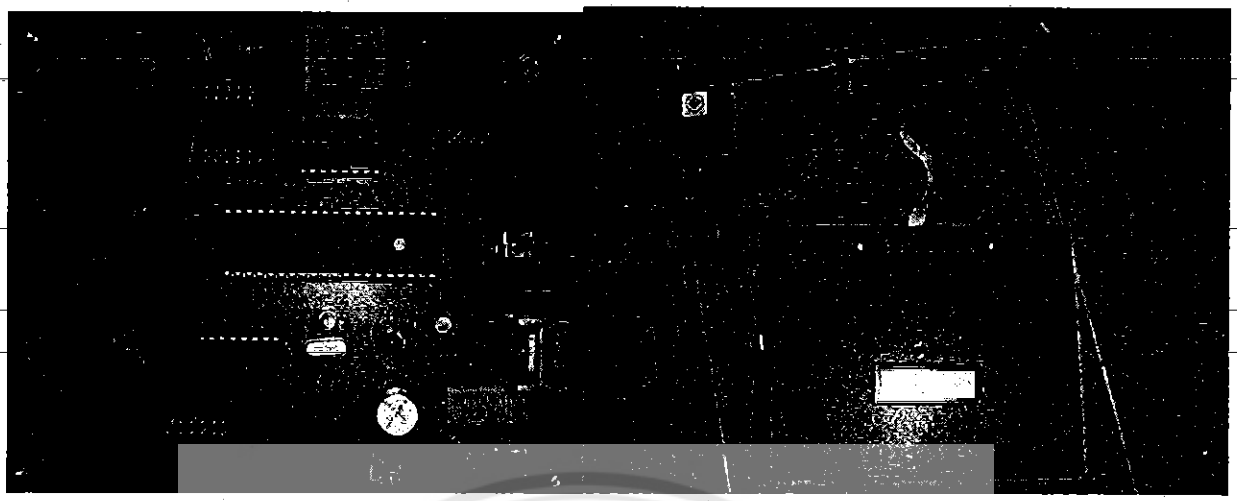
ประกอบด้วยแผงไมโครคอนโทรลเลอร์ และโมเดม ซึ่งถูกบรรจุลงในกล่องมีสายอแดปเตอร์เพื่อต่อเข้ากับไฟฟ้า และสายวิทยุสื่อสารซึ่งเป็นสายที่ใช้ในการควบคุมการเปิด/ปิดของพอร์ท Ptt ของวิทยุสื่อสารและพอร์ทข้อมูลอินพุทเข้าวิทยุสื่อสารทางสาย MIC อุปกรณ์ชิ้นนี้จะทำหน้าที่นำข้อมูลจากเครื่องวัดการเลื่อนของดินมาประมวลผลและแสดงออกทางจอ LCD และส่งข้อมูลนี้ต่อไปยังภาครับ

*** อธิบายสายวิทยุสื่อสาร



รูปที่ 4.3 รูปแสดงสายวิทยุสื่อสารหมายเลข 1 คือ port Ptt และหมายเลข 2 คือ port MIC

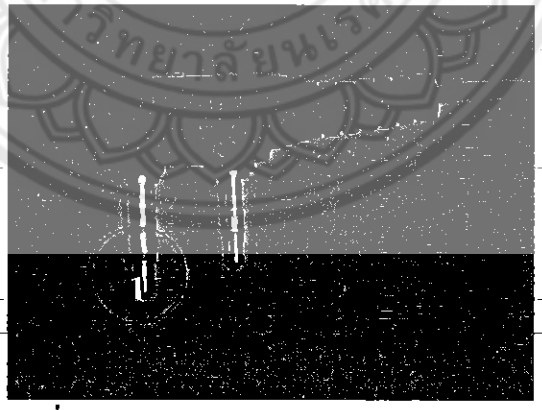
4.1.3.เครื่องรับข้อมูล



รูปที่ 4.4 ภาครับข้อมูล

ประกอบด้วยแผงไมโครคอนโทรลเลอร์ และโมเดม ซึ่งถูกบรรจุลงในกล่องมีสายอแดปเตอร์เพื่อต่อเข้ากับไฟฟ้า และสายวิทยุสื่อสารซึ่งเป็นสายพอร์ทรับข้อมูลอินพุทเข้าวิทยุสื่อสารทางสาย SP อุปกรณ์ชิ้นนี้จะทำหน้าที่รับข้อมูลจากเครื่องส่งและประมวลผลและแสดงออกทางจอ LCD และส่งข้อมูลนี้ต่อไปยังอุปกรณ์เตือนภัยอื่นๆ

*** อธิบายสายวิทยุสื่อสาร



รูปที่ 4.5 รูปแสดงสายวิทยุสื่อสารหมายเลข 1 คือ port SP หมายเลข 2 ขอสงวนไว้ไม่ได้ใช้งาน

4.2 วิธีทำการทดลอง

1. ต่ออุปกรณ์เครื่องวัดการลื่อนของดินเข้ากอนเครื่องส่งข้อมูลและประมวลผล
2. ต่ออุปกรณ์เครื่องส่งและเครื่องรับเข้ากับวิทยุสื่อสาร ต่ออแดปเตอร์เข้ากับปลั๊กไฟฟ้า
3. เมื่อต่ออุปกรณ์เครื่องส่งเสร็จแล้วหน้าจอ LCD จะขึ้นคำว่า "***Start****"
4. เมื่อต่ออุปกรณ์เครื่องรับเสร็จเรียบร้อยแล้วหน้าจอ LCD จะขึ้นคำว่า "DATA MAX= 0 CM" ที่ขึ้นแบบนี้เนื่องจากเครื่องรับจะรับขึ้นมูลที่มากที่สุด(ระยะทาง cm) ที่เครื่องส่งส่งมาเครื่องรับ
5. ทำการดึงโซ่ออกโดยใช้ไม้บรรทัดเทียบระยะกับโซ่ โดยเริ่มจาก 1 cm และดูหน้าจอ LCD ที่เครื่องส่งว่าเครื่องมีการส่งข้อมูล 1 cm ที่บิตข้อมูลที่เท่าไรและบันทึกผลลงในตาราง
6. ทำการตรวจสอบว่าเครื่องรับมือการรับข้อมูลจากเครื่องส่งหรือไปเมื่อความส่ง ส่งข้อมูลระยะ 1 cm ออกไป
7. ทำการทดลองซ้ำ 5-6 โดยเปลี่ยนระยะเป็น 2,3,4,...,25 cm และบันทึกผลลงในตาราง

4.3 ผลการทดลอง

4.3.1 ผลภาควัดการเลื่อนของดิน

ทางคณะผู้จัดทำได้ทำการวัดค่าความผิดพลาดที่เกิดจากการเลื่อนของดินทั้งหมด 3 ครั้งและแสดงออกมาเป็นได้ดังตารางต่อไปนี้

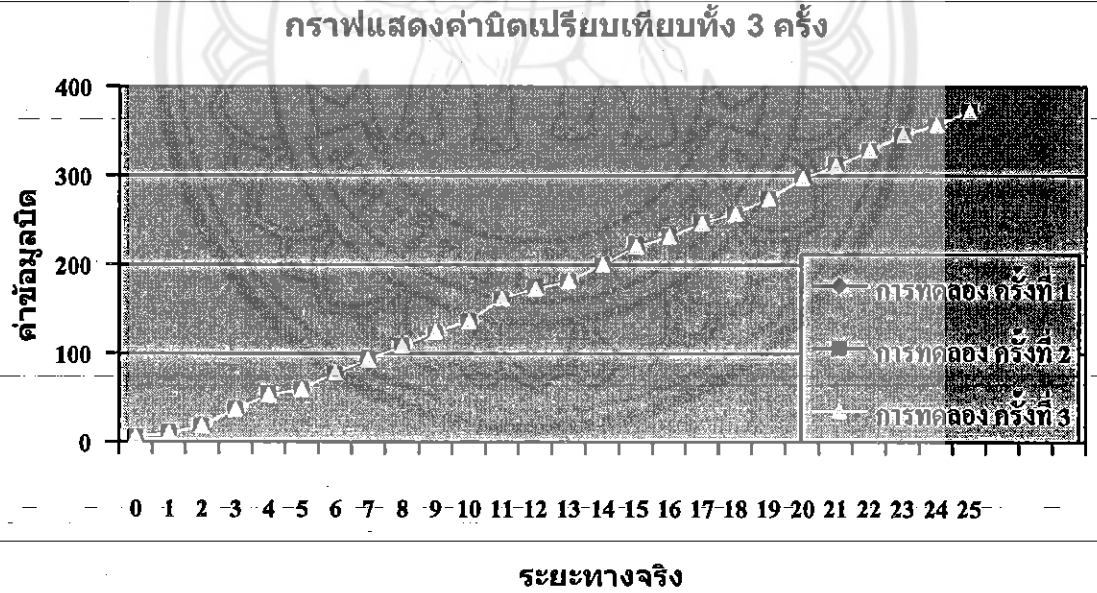
การทดลอง

ภาคเครื่องวัดการเลื่อนของดิน			
ระยะทางจริง	ค่าข้อมูลบิต		
	การทดลองครั้งที่ 1	การทดลองครั้งที่ 2	การทดลองครั้งที่ 3
0	0-6	0-6	0-6
1	11	11	10
2	18	18	18
3	37	37	37
4	53	54	53
5	59	59	59
6	76	76	78
7	93	94	93
8	108	108	108
9	122	122	124
10	136	136	136
11	161	161	162
12	173	175	173
13	181	181	181
14	199	199	200
15	220	222	220
16	232	232	232
17	247	248	247
18	256	256	257
19	274	274	274
20	297	297	297

ภาคเครื่องวัดการเลื่อนของดิน			
ระยะทางจริง	ค่าข้อมูลบิต		
	การทดลองครั้งที่	การทดลองครั้งที่	การทดลองครั้งที่
	1	2	3
21	311	313	311
22	328	328	328
23	345	345	345
24	357	357	357
25	372	372	372

ตารางที่ 4.1 ผลการทดลองภาคเครื่องวัดการเลื่อนของดิน

สามารถแสดงไปเป็นกราฟได้ดังรูป



รูปที่ 4.6 กราฟแสดงระหว่างระยะจริงกับระยะทางที่เครื่องวัดวัดได้

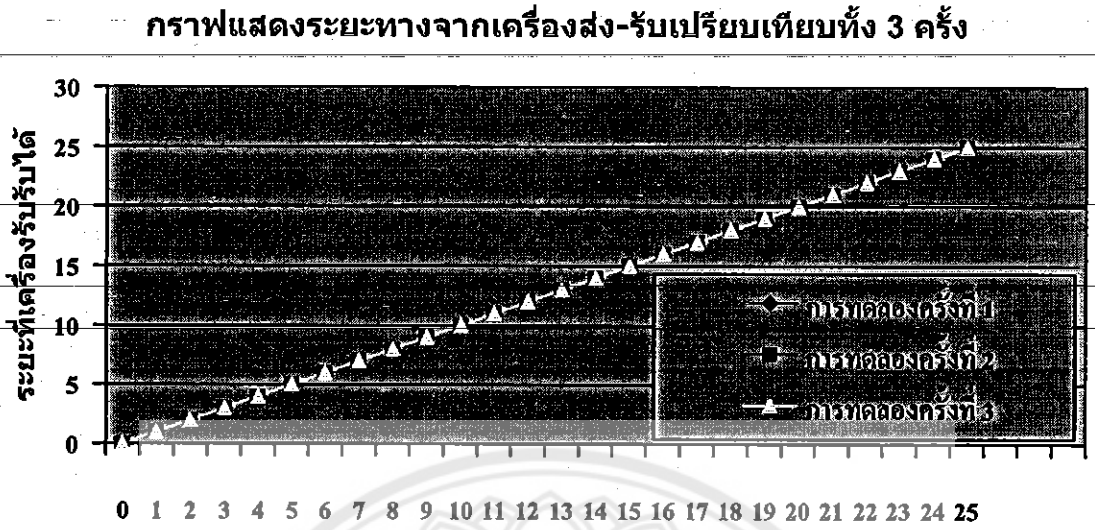
4.3.2 ผลการทดลองภาครับข้อมูล

จากการทดลองรับข้อูลจากภาคส่ง 3 ครั้งสามารถแสดงได้ดังตารางที่ 4.2

ข้อมูลที่เครื่องวัดส่ง (ระยะทาง cm)	การทดลองครั้งที่ 1 (ระยะทาง cm)	การทดลองครั้งที่ 2 (ระยะทาง cm)	การทดลองครั้งที่ 3 (ระยะทาง cm)
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4
5	5	5	5
6	6	6	6
7	7	7	7
8	8	8	8
9	9	9	9
10	10	10	10
11	11	11	11
12	12	12	12
13	13	13	13
14	14	14	14
15	15	15	15
16	16	16	16
17	17	17	17
18	18	18	18
19	19	19	19
20	20	20	20
21	21	21	21
22	22	22	22
23	23	23	23
24	24	24	24
25	25	25	25

ตารางที่ 4.2 ผลการทดลองภาครับข้อมูล

สามารถแสดงไปเป็นกราฟได้ดังรูป



รูปที่ 4.7 กราฟแสดงระหว่างระยะที่เครื่องวัดส่งกับระยะทางที่เครื่องรับวัดได้

บทที่ 5

สรุปผลการทดลอง

5.1 สรุปผลการดำเนินงานโครงงาน

จากโครงงานการเครื่องวัดการเลื่อนของดินสามารถข้อสรุปจากผลการดำเนินงานดังนี้

1. เครื่องวัดสามารถทำการวัดระยะได้ตามระยะจริง
2. ภาคส่งสามารถประมวลผลข้อมูล และทำการส่งข้อมูลได้ถูกต้องแม่นยำ
3. ภาครับสามารถรับข้อมูลจากส่ง และทำการประมวลผลได้อย่างถูกต้องแม่นยำ
4. สามารถส่งข้อมูลไปยังสถานีอื่นๆได้ถูกต้อง

5.2 ผลที่ได้รับจากการทำโครงงาน

1. เกิดความรู้ ความเข้าใจการทำงานของตัวต้านทานปรับค่าได้, การเขียนโปรแกรมภาษา c AVR และการเชื่อมต่อไมโครคอนโทรลเลอร์มากยิ่งขึ้น
2. เกิดความรู้ ความเข้าใจในวิชาพื้นฐานที่มีอยู่เดิม สามารถนำมาใช้และพัฒนาตนเองต่อไป

5.3 ปัญหาและแนวทางการแก้ไข

จากการทำโครงงานการเครื่องวัดการเลื่อนของดินพบปัญหาและอุปสรรคต่างๆ ดังนี้

1. การดำเนินงานมีความล่าช้า เนื่องจากขาดความรู้ความเข้าใจในการเลือกอุปกรณ์ และการออกแบบวงจรที่เหมาะสมกับโครงงาน
2. การดำเนินงานในด้านโปรแกรม บางครั้งเกิดปัญหาข้อผิดพลาดในด้านการพัฒนาโปรแกรมทำให้เกิดความล่าช้าในการดำเนินงาน แก้ไขโดยศึกษาด้วยตนเองและสอบถามผู้ที่เชี่ยวชาญในด้านนี้

5.4 ข้อจำกัดของระบบ

1. ระบบมีระยะทางการทำงานจำกัด ไม่เกิน 25 cm.
2. วิทยุสามารถส่งได้ไม่เกิน 3 km.
3. ไม่สามารถใช้ย่านความถี่วิทยุร่วมกับย่านความถี่อื่นๆได้เพราะอาจมีการรบกวนของสัญญาณ

5.5 ข้อเสนอแนะในการพัฒนาต่อไป

1. ควรมีการพัฒนาอุปกรณ์ที่ใช้อย่างสม่ำเสมอ เพื่อให้อุปกรณ์มีความสมบูรณ์และพร้อม ใช้งานได้มากยิ่งขึ้น
2. ควรมีการตรวจเช็คอุปกรณ์อิเล็กทรอนิกส์อย่างสม่ำเสมอ เนื่องจากอุปกรณ์เหล่านี้มีโอกาสที่จะเกิดข้อผิดพลาดได้ง่าย ถ้าหากการดูแลรักษาไม่ดีเท่าที่ควร
3. ควรมีการพัฒนาวงจรให้สามารถใช้งานได้ดีขึ้นและละเอียดมากขึ้น

เอกสารอ้างอิง

(1) Data Sheet ATMEGA32 ATMEGA32.pdf

(2) SEVEN DARLINGTON ARRAYS ULN2001A-ULN2002A.pdf

(3) คู่มือการใช้งานโมเดมอเนกประสงค์ MAN.pdf

(4) PX-4000 mini AVR In-System programmer 8000288.pdf

(5) รายงานโครงงานหมายเลข EE 2003 -70 ATMEGA32-1.pdf

(6) คู่มือ ET-BASE51 AC3 คู่มือ ET-BASE51 AC3.pdf

(6) AVR ISP : เบื้องต้น : http://www.electoday.com/bbs/archiver/?tid-180.html

(7) http://www.thaiamp.com/Main.htm

(8) ประจิน พลังสันติกุล.การเขียนโปรแกรมควบคุมไมโครคอนโทรลเลอร์ AVR.ครั้งที่ 1.กรุงเทพ

ภาคผนวก ก.

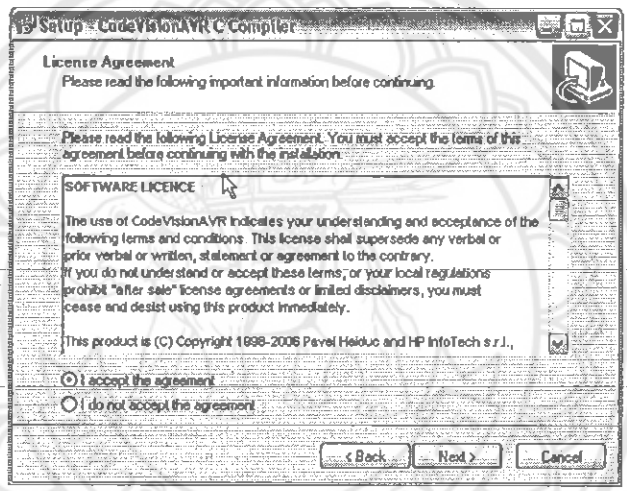
การลงโปรแกรม

1. การติดตั้งโปรแกรม CodeVisionAVR

1). ติดตั้งโปรแกรม CodeVision AVR ลงบน PC โดยเปิดไฟล์ set up ตามภาพ

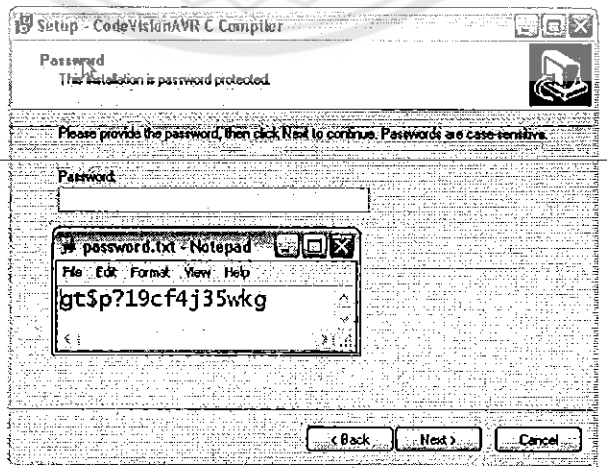
รูปที่ 1.1 setup.exe file

2). เลือกที่ช่อง "I accept the agreement" แล้วกด next



รูปที่ 1.2 Setup – CodeVisionAVR Complier(1)

3). ทำการใส่ password ลงไป แล้วคลิกที่ next

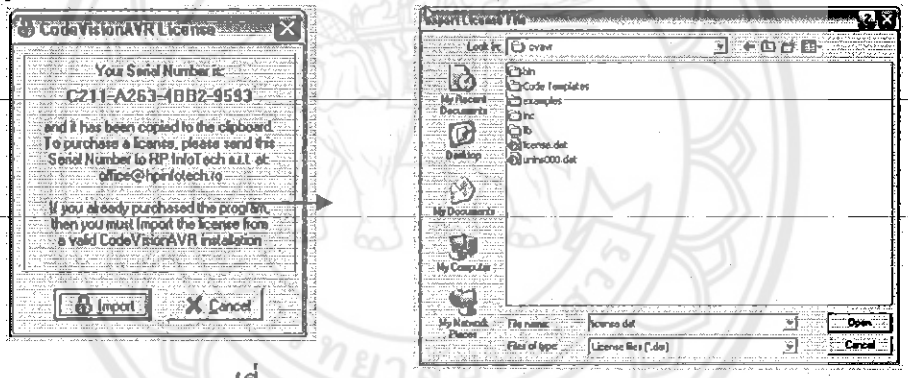


รูปที่ 1.3 Setup – CodeVisionAVR Complier(2)

4). คลิก next ไปเรื่อยๆจนกว่าจะเจอหน้าต่างดังภาพ

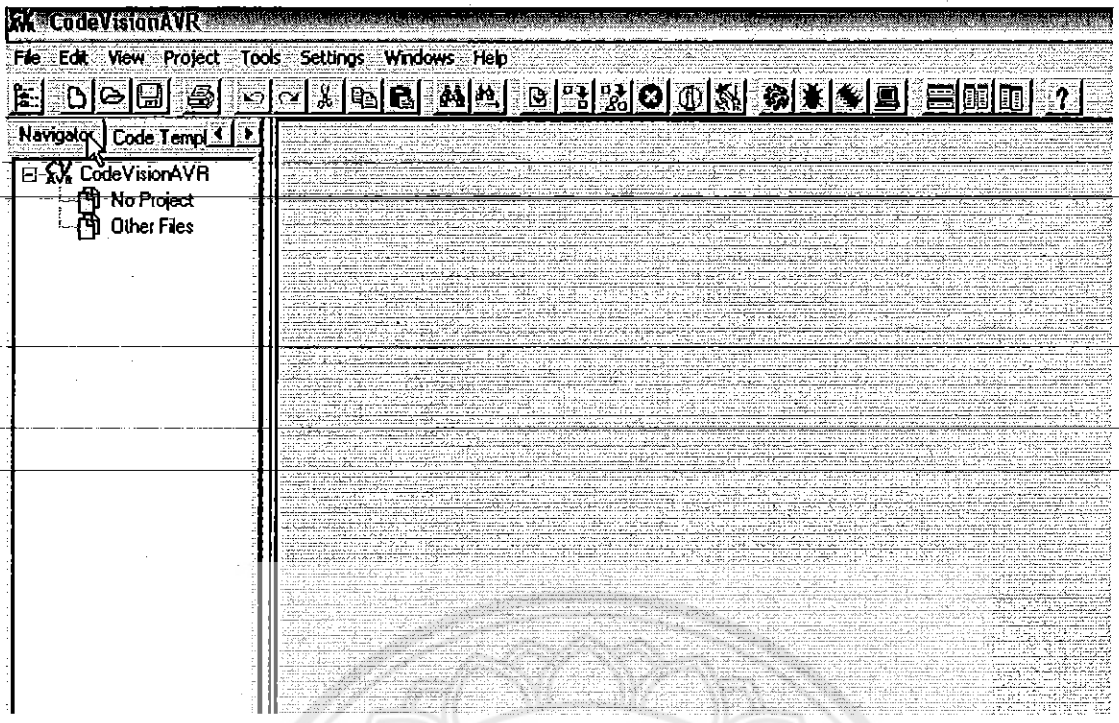
รูปที่ 1.4 Setup – CodeVisionAVR Complier(3)

5). ทำการ Copy license.dat ไปไว้ที่ C:\Cvavr จากนั้นทำการ IMPORT ดังภาพด้านล่าง



รูปที่ 1.5 Setup – CodeVisionAVR Complier(4)

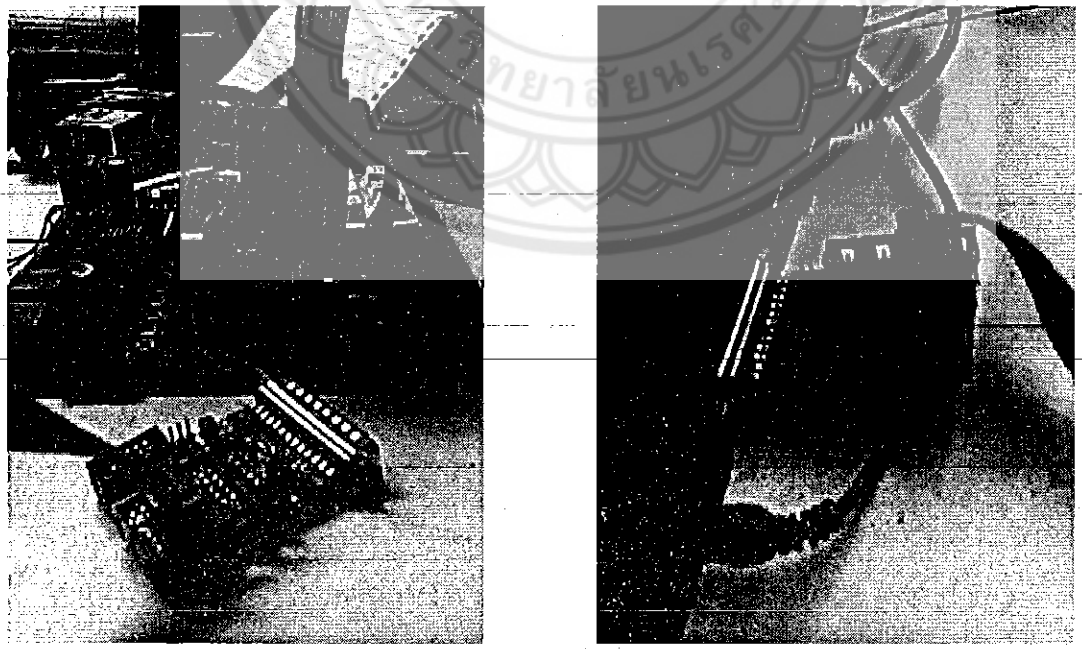
6). กด OPEN File license.dat จากนั้นกด OK



รูปที่ 1.6 Setup – CodeVisionAVR Complier(5)

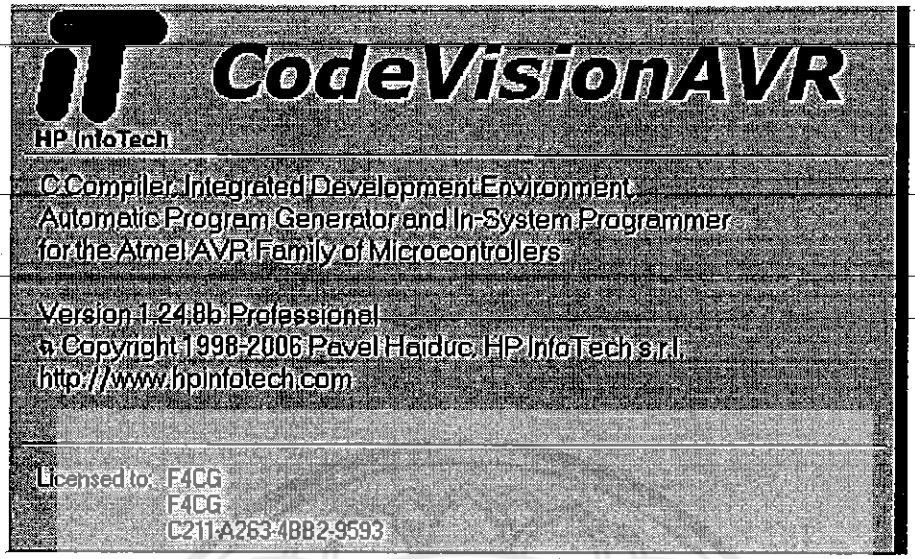
2. การ Burn โปรแกรม

1). ต่ออุปกรณ์ตามภาพ



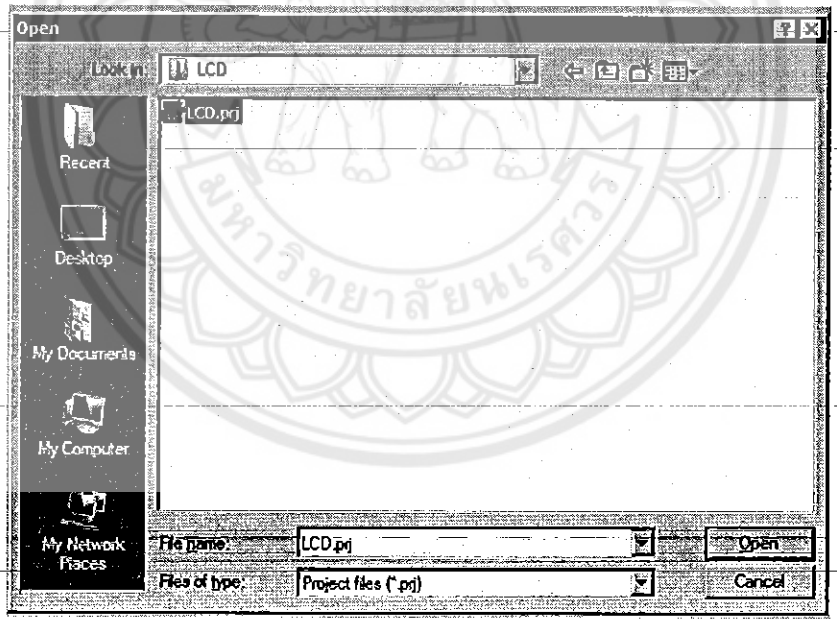
รูปที่ 2.1 Setup – AVR ISP Burn (1)

2). เปิดโปรแกรม CodeVisionAVR



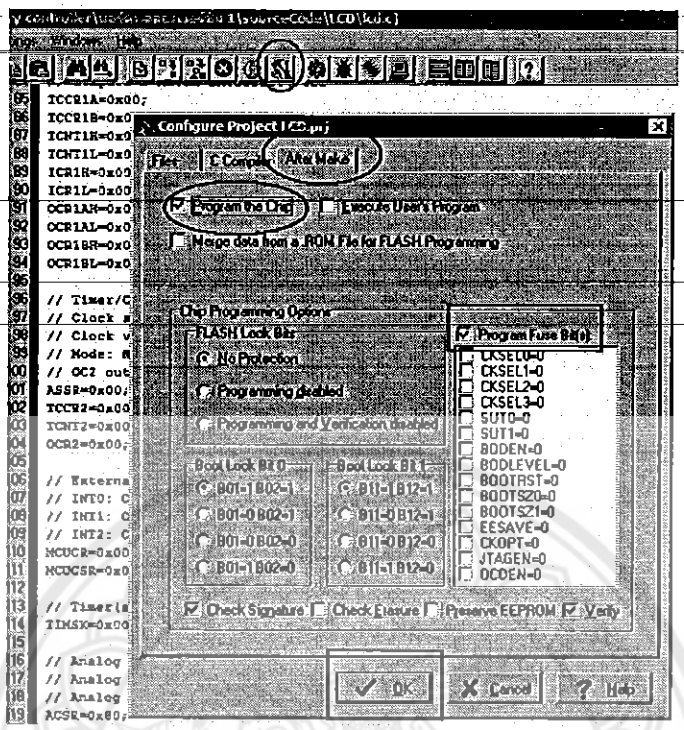
รูปที่ 2.2 Setup – AVR ISP Burn (2)

3). ทำการ Open Object ที่ได้สร้างเอาไว้



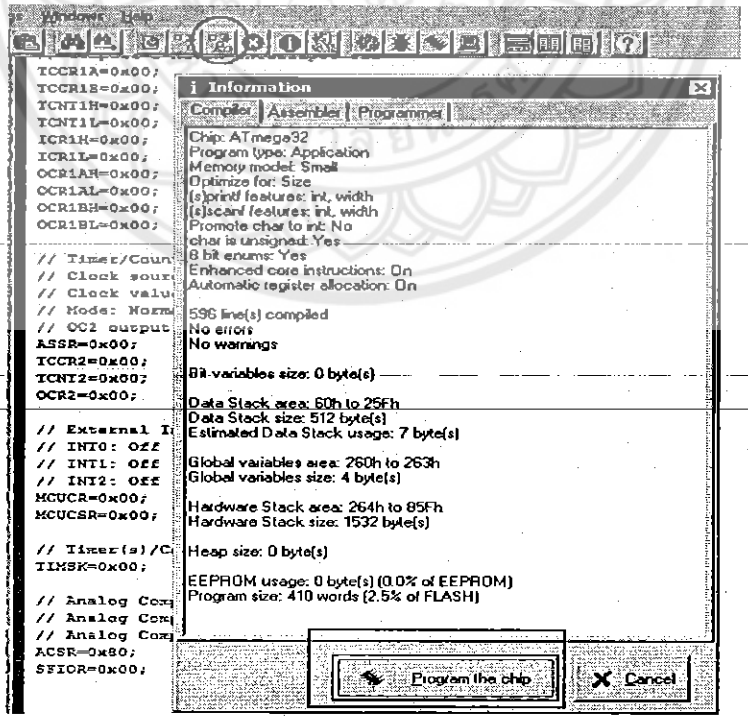
รูปที่ 2.3 Setup – AVR ISP Burn (3)

4). ทำการ set ค่าตามภาพ



รูปที่ 2.4 Setup -- AVR-ISP-Burn (4)

5). ทำการ set ค่าตามภาพ จากนั้นจึงทำการ burn โปรแกรมที่เขียนได้ลงในบอร์ด



ภาคผนวก ข.

Code ของโปรแกรม

ภาคเครื่องวัดและส่งข้อมูล

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com
Chip type           : ATmega32
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 512
*****************************************************/
#include <mega32.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
#define Client 1 //กำหนดให้ตัวแปร Client มีค่าเท่ากับ 1
#define  LED  PORTB.4 //กำหนดให้ตัวแปร LED เป็นตัวแปรขา PB4
#define  RELAY  PORTB.3 //กำหนดให้ตัวแปร RELAY เป็นตัวแปรขา PB3
int   DIV =0; //ประกาศตัวแปร DIV มีค่าเท่ากับ 0
char temp[16]; //ประกาศตัวแปร temp แบบอาร์เรย์ 16 ตัว
unsigned int read0=0,read1=0,read2=0,read3=0,data_send;
unsigned int data_send0=0;
unsigned int read_max=0;
int val1=0,val2,i;
```

ประกาศค่าตัวแปร

```
char cnt_read_ok=0;
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
#endasm
#include <lcd.h>
#define RXB8 1 //กำหนดขา RXB ของ LCD ให้เป็น port PC1 ของไมโครคอนโทรลเลอร์ ATmega32
#define TXB8 0 //กำหนดขา TXB ของ LCD ให้เป็น port PC0 ของไมโครคอนโทรลเลอร์ ATmega32
#define UPE 2 //กำหนดขา UPE ของ LCD ให้เป็น port PC2 ของไมโครคอนโทรลเลอร์ ATmega32
#define OVR 3 //กำหนดขา OVR ของ LCD ให้เป็น port PC3 ของไมโครคอนโทรลเลอร์ ATmega32
#define FE 4 //กำหนดขา FE ของ LCD ให้เป็น port PC4 ของไมโครคอนโทรลเลอร์ ATmega32
#define UDRE 5 //กำหนดขา UDRE ของ LCD ให้เป็น port PC5 ของไมโครคอนโทรลเลอร์ ATmega32
#define RXC 7 //กำหนดขา RXC ของ LCD ให้เป็น port PC7 ของไมโครคอนโทรลเลอร์ ATmega32

#define FRAMING_ERROR (1<<FE) //กำหนดตัวแปร FRAMING_ERROR แล้วมีการกำหนดตัวแปร FE=1
#define PARITY_ERROR (1<<UPE) //กำหนดตัวแปร PARITY_ERROR แล้วมีการกำหนดตัวแปร UPE=1
#define DATA_OVERRUN (1<<OVR) //กำหนดตัวแปร DATA_OVERRUN แล้วมีการกำหนดตัวแปร OVR=1
#define DATA_REGISTER_EMPTY (1<<UDRE) //กำหนดตัวแปร DATA_REGISTER_EMPTY แล้วมีการ
กำหนดตัวแปร UDRE=1
#define RX_COMPLETE (1<<RXC) //กำหนดตัวแปร RX_COMPLETE แล้วมีการกำหนดตัวแปร RXC=1

// USART Receiver buffer
#define RX_BUFFER_SIZE 8 //กำหนดตัวแปร RX_BUFFER_SIZE มีค่าเท่ากับ 8
char rx_buffer[RX_BUFFER_SIZE]; //กำหนดตัวแปร rx_buffer มีค่าอาเรย์เท่ากับ 8

#if RX_BUFFER_SIZE<256 //ถ้า RX_BUFFER_SIZE<256
//กำหนดตัวแปร char rx_wr_index,rx_rd_index,rx_counter เป็นชนิด unsigned char
unsigned char rx_wr_index,rx_rd_index,rx_counter;
#else//ถ้าไม่ใช่
//กำหนดตัวแปร char rx_wr_index,rx_rd_index,rx_counter เป็นชนิด unsigned int
unsigned int rx_wr_index,rx_rd_index,rx_counter;
#endif
```

ฟังก์ชัน LCD โดยประกาศใช้ LCD ใช้ Port C ใน
ไมโครคอนโทรลเลอร์

```
// USART Transmitter interrupt service routine
interrupt [USART_TXC] void usart_tx_isr(void)//เปิดใช้ Interupt ของ USART
{
if (tx_counter)//เช็คค่า tx_counter เท่ากับ 1 หรือไม่
  {
  --tx_counter;
  UDR=tx_buffer[tx_rd_index];
  if (++tx_rd_index == TX_BUFFER_SIZE) tx_rd_index=0;//กำหนดให้ตัวแปรเพิ่มค่า 1 ค่าแล้วทำการตรวจสอบว่ามีค่า
เท่ากับ TX_BUFFER_SIZE หรือไม่ ถ้าใช่ให้ค่าของตัวแปร tx_rd_index เท่ากับ 0
  };
}
#ifndef _DEBUG_TERMINAL_IO_
// Write a character to the USART Transmitter buffer
#define _ALTERNATE_PUTCHAR_
#pragma used+
void putchar(char c)//ประกาศฟังก์ชันย่อย putchar
{
while (tx_counter == TX_BUFFER_SIZE); //เมื่อ tx_counter เท่ากับ TX_BUFFER_SIZE
#asm("cli")
if (tx_counter || ((UCSRA & DATA_REGISTER_EMPTY)==0))
//ถ้า ค่า UCSRA มา (ANDข้อมูล) กับ DATA_REGISTER_EMPTY และ ทั้งหมดมา (ORข้อมูล) กับ tx_counter เท่ากับ 0หรือไม่
{
  tx_buffer[tx_wr_index]=c; //กำหนดค่าตัวแปร tx_buffer เท่ากับ C
  if (++tx_wr_index == TX_BUFFER_SIZE) tx_wr_index=0; ;//กำหนดให้ตัวแปรเพิ่มค่า 1 ค่าแล้วทำการตรวจสอบว่ามี
ค่าเท่ากับ TX_BUFFER_SIZE หรือไม่ ถ้าใช่ให้ค่าของตัวแปร tx_wr_index เท่ากับ 0
  ++tx_counter;
  }
else //ถ้าใช่
  UDR=c; //ให้ค่า UDC เท่ากับ 0
#asm("sei")
}
#pragma used-
```

```
#endif
// Standard Input/Output functions
#define ADC_VREF_TYPE 0x00 //กำหนดค่าตัวแปร ADC_VREF_TYPE เท่ากับศูนย์
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)//ประกาศค่าฟังก์ชันย่อย read_adc แล้วทำการรับค่าจากตัวแปร adc_input
{

ADMUX=adc_input | (ADC_VREF_TYPE & 0xff
//นำค่าในตัวแปร ADC_VREF_TYPE มา(AND ข้อมูล) กับ 11111111₂และทั้งหมดมา (OR ข้อมูล) กับ adc_input
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=0x40; //นำค่าในตัวแปร ADCSRA มา (OR ข้อมูล) กับค่า 40₁₆ แล้วนำค่าที่ได้เก็บไว้ในตัวแปร ADCSRA
// Wait for the AD conversion to complete
while ((ADCSRA & 0x10)==0); //ทำซ้ำจนกว่าข้อมูลของตัวแปร ADCSRA มา (AND ข้อมูล) กับค่า 10₁₆ ค่าเป็นศูนย์
ADCSRA|=0x10; ; //นำค่าในตัวแปร ADCSRA มา (OR ข้อมูล) กับค่า 10₁₆ แล้วนำค่าที่ได้เก็บไว้ในตัวแปร ADCSRA
return ADCW; //กลับไปที่ ADCW
}
// Declare your global variables here
void main(void) //ประกาศใช้ฟังก์ชันย่อย
{
// Declare your local variables here
// Input/Output Ports initialization
// Port A initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTA=0x00;
DDRA=0x00;
// Port B initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTB=0x00;
```

```
DDRB=0x18;
// Port C initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTC=0x00;
DDRC=0x00;
// Port D initialization
// Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
// State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
PORTD=0x00;
DDRD=0x00;

// Timer/Counter 0 initialization
// Clock source: System Clock
// Clock value: Timer 0 Stopped
// Mode: Normal top=FFh
// OC0 output: Disconnected
TCCR0=0x00;
TCNT0=0x00;
OCR0=0x00;
// Timer/Counter 1 initialization
// Clock source: System Clock
// Clock value: Timer 1 Stopped
// Mode: Normal top=FFFFh
// OC1A output: Discon.
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
```

```
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;
// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;
// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: Off
// USART Transmitter: On
```

```
// USART Mode: Asynchronous
// USART Baud Rate: 1200
UCSRA=0x00;
UCSRB=0x48;
UCSRC=0x86;
UBRRH=0x02;
UBRRL=0x3F;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;//กำหนดให้รีจิสเตอร์ ACSR มีค่าเท่ากับ 80_16
SFIOR=0x00; ;//กำหนดให้รีจิสเตอร์ SFIOR มีค่าเท่ากับ 00_16
// ADC initialization
// ADC Clock frequency: 172.800 kHz
// ADC Voltage Reference: AREF pin
ADMUX=ADC_VREF_TYPE & 0xff;
ADCSRA=0x86; ;//กำหนดให้รีจิสเตอร์ ADCSR มีค่าเท่ากับ 86_16
// LCD module initialization
lcd_init(16);//กำหนดค่า 16 ให้กับฟังก์ชัน lcd_init
// Global enable interrupts
#asm("sei")
   lcd_gotoxy(0,0);  //กำหนดตำแหน่ง LCD(0,0)
   lcd_putsf("  ADC  sending  ");// LCD แสดง ADC Sending
   LED=0;
   delay_ms(400);
   LED=1;
   delay_ms(400);
   LED=0;
   delay_ms(400);
   LED=1;
   delay_ms(400);
```

กำหนดความเร็วในการรับข้อมูลเท่ากับ 1200 bps โดยคำนวณจาก XTAL 11.0592 MHz

LED ติดดับสลับกันโดย delay time 0.45s

```
        LED=0;
        delay_ms(400);
        LED=1;
        delay_ms(400);
        lcd_gotoxy(0,0);   //กำหนดตำแหน่ง LCD(0,0)
        lcd_putsf(" waitting data  ");");// LCD แสดง waitting data
      // Global enable interrupts
      #asm("sei") //ประกาศค่าตัวแปรภาษา Asemble
      LED=0;
      DIV = 1;
      val2=0;
      val1=1;

      while (1)
          {
           //lcd_clear();
              delay_us(100);  //หน่วงเวลา 100 µS
                  for(i=0;i<63;i++)
                  {
                   read0 =  read0 + read_adc(0);          //read_adc_ch0;
                   delay_us(220);
                  }
                  for(i=0;i<63;i++)
                  {
                   read1 =  read1 + read_adc(1);          //read_adc_ch1;
                   delay_us(220);
                  }
                  for(i=0;i<63;i++)
                  {
                   read2 =  read2 + read_adc(2);          //read_adc_ch2;
                   delay_us(220);
                  }
```

LED ติดดับสลับกัน โดย delay time 0.45s

ทำการรับค่า ADC 63 ค่าที่ chanal 0
เก็บค่าไว้ในตัวแปร read 0

ทำการรับค่า ADC 63 ค่าที่ chanal 1
เก็บค่าไว้ในตัวแปร read 1

ทำการรับค่า ADC 63 ค่าที่ chanal 2
เก็บค่าไว้ในตัวแปร read 2

```
        for(i=0;i<63;i++)
        {
         read3 = read3 + read_adc(3);        //read_adc_ch3;   } ทำการรับค่า ADC 63 ค่าที่ chanal 3
         delay_us(220);                                        } เก็บค่าไว้ในตัวแปร read 3
        }
        read0 = read0/63;
        read1 = read1/63;                                      } ทำการหาร read0 ถึง read 3 ด้วย 63
        read2 = read2/63;
        read3 = read3/63;
        delay_us(100);


      if((read0==5)&&(read1==5)&&(read2==5)&&(read3==5)) //ทำการตรวจสอบว่าค่า read0-read3 มีค่าเท่ากับ 5
หรือไม่
        {
          lcd_clear(); //clear หน้าจอ LCD
          lcd_gotoxy(0,1); //กำหนดตำแหน่งการแสดงผล LCD โดย X=0,Y=1
          lcd_putsf("*****START*****#"); //ให้ LCD แสดงผล  *****START*****
          delay_ms(1500);
          //lcd_clear();
          };
        lcd_clear();
        lcd_gotoxy(0,0);
        sprintf(temp,"CH0:%i",read0); //เก็บค่าตัวแปร read0 พร้อมกับ CH0 ไว้ในตัวแปร temp
        lcd_puts(temp);//แสดงค่าตัวแปรที่อยู่ในตัวแปร temp แสดงผลใน LCD
        lcd_gotoxy(8,0);//แสดงผล lcd ตำแหน่ง (8,0)
        sprintf(temp,"CH1:%i",read1); //เก็บค่าตัวแปร read1 พร้อมกับ CH1 ไว้ในตัวแปร temp
        lcd_puts(temp); //แสดงค่าตัวแปรที่อยู่ในตัวแปร temp แสดงผลใน LCD
        lcd_gotoxy(0,1); //แสดงผล lcd ตำแหน่ง (0,1)
        sprintf(temp,"CH2:%i",read2); //เก็บค่าตัวแปร read2 พร้อมกับ CH2 ไว้ในตัวแปร temp
        lcd_puts(temp); //แสดงค่าตัวแปรที่อยู่ในตัวแปร temp แสดงผลใน LCD
```

```
lcd_gotoxy(8,1); //แสดงผล lcd ตำแหน่ง (8,1)
sprintf(temp,"CH3:%i",read3); //เก็บค่าตัวแปร read3 พร้อมกับ CH3 ไว้ในตัวแปร temp
lcd_puts(temp); //แสดงค่าตัวแปรที่อยู่ในตัวแปร temp แสดงผลใน LCD
delay_us(100);
if(read0<5)                          ถ้า read0 มีค่าน้อยกว่า 5 ให้แสดงผล LCD
{                                    ตำแหน่ง(0,0) ว่า "CH0:NC"
lcd_gotoxy(0,0);
lcd_putsf("CH0:NC");
}
if(read1<5)                          ถ้า read1 มีค่าน้อยกว่า 5 ให้แสดงผล LCD
{                                    ตำแหน่ง(8,0) ว่า "CH0:NC"
lcd_gotoxy(8,0);
lcd_putsf("CH1:NC");
}
if(read2<5)                          ถ้า read2 มีค่าน้อยกว่า 5 ให้แสดงผล LCD
{                                    ตำแหน่ง(0,1) ว่า "CH0:NC"
lcd_gotoxy(0,1);
lcd_putsf("CH2:NC");
}
if(read3<5)                          ถ้า read3 มีค่าน้อยกว่า 5 ให้แสดงผล LCD
{                                    ตำแหน่ง(8,1) ว่า "CH0:NC"
lcd_gotoxy(8,1);
lcd_putsf("CH3:NC");
}
 if(read0==5)                        ถ้า read0 มีค่าเท่ากับ 5 ให้แสดงผล LCD
{                                    ตำแหน่ง(0,0) ว่า "CH0:ST"
lcd_gotoxy(0,0);
lcd_putsf("CH0:ST");
}
```

```
        if(read1==5)
        {
        lcd_gotoxy(8,0);
        lcd_putsf("CH1:ST");
        }
        if(read2==5)
        {
        lcd_gotoxy(0,1);
        lcd_putsf("CH2:ST");
        }
        if(read3==5)
        {
        lcd_gotoxy(8,1);
        lcd_putsf("CH3:ST");
        }
```

ถ้า read1 มีค่าเท่ากับ 5 ให้แสดงผล LCD ตำแหน่ง(8,0) ว่า "CH0:ST"

ถ้า read2 มีค่าเท่ากับ 5 ให้แสดงผล LCD ตำแหน่ง(0,1) ว่า "CH0:ST"

ถ้า read3 มีค่าเท่ากับ 5 ให้แสดงผล LCD ตำแหน่ง(8,1) ว่า "CH0:ST"

```
        if ((read0>=read1)&&(read0>=read2)&&(read0>=read3))//เปรียบเทียบว่าค่าตัวแปร read 0 มีค่ามากกว่า
read1,read2,read3 หรือไม่
            read_max = read0; //ถ้าใช่ให้กำหนด read_max=read0
        delay_us(100);
        if ((read1>=read0)&&(read1>=read2)&&(read1>=read3)) ))//เปรียบเทียบว่าค่าตัวแปร read 1 มีค่ามากกว่า
read0,read2,read3 หรือไม่
            read_max = read1; //ถ้าใช่ให้กำหนด read_max=read1
        delay_us(100);
        if ((read2>=read1)&&(read2>=read0)&&(read2>=read3)) ))//เปรียบเทียบว่าค่าตัวแปร read 2 มีค่ามากกว่า
read1,read0,read3 หรือไม่
            read_max = read2; //ถ้าใช่ให้กำหนด read_max=read2
        delay_us(100);
        if ((read3>=read2)&&(read3>=read1)&&(read3>=read0)) ))//เปรียบเทียบว่าค่าตัวแปร read 3 มีค่ามากกว่า
read1,read2,read0 หรือไม่
            read_max = read3; //ถ้าใช่ให้กำหนด read_max=read2
```

```
        delay_us(100);
        ////////// Process bit:data cm./////////////****
      if(read_max<11)
{data_send = 0;};
```
ถ้า ค่าตัวแปร read_max น้อยกว่า 11 ตัวแปร data_send จะมีค่าเท่ากับ 0

```
if((read_max >= 11)&&(read_max < 18)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 11-17 ตัวแปร
{data_send = 1;}; //data_send จะมีค่าเท่ากับ 1
if((read_max >= 18)&&(read_max < 37)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 18-36 ตัวแปร
{data_send = 2;}; //data_send จะมีค่าเท่ากับ 2
 if((read_max >= 37)&&(read_max < 53)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 37-52 ตัวแปร
{data_send = 3;}; //data_send จะมีค่าเท่ากับ 3
if((read_max >= 53)&&(read_max < 59)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 53-58 ตัวแปร
{data_send = 4;}; //data_send จะมีค่าเท่ากับ 4
if((read_max >= 59)&&(read_max < 76)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 59-75 ตัวแปร
{data_send = 5;}; //data_send จะมีค่าเท่ากับ 5
 if((read_max >= 76)&&(read_max < 93)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 76-92 ตัวแปร
{data_send = 6;}; //data_send จะมีค่าเท่ากับ 6
 if((read_max >= 93)&&(read_max < 108)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 93-108 ตัวแปร
{data_send = 7;}; //data_send จะมีค่าเท่ากับ 7
if((read_max >= 108)&&(read_max < 122)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 108-121 ตัวแปร
{data_send = 8;}; //data_send จะมีค่าเท่ากับ 8
 if((read_max >= 122)&&(read_max < 136)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 122-135 ตัวแปร
{data_send = 9;}; //data_send จะมีค่าเท่ากับ 9
 if((read_max >= 136)&&(read_max < 151)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 136-150 ตัวแปร
{data_send = 10;}; //data_send จะมีค่าเท่ากับ 10
if((read_max >= 151)&&(read_max < 173)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 151-172 ตัวแปร
{data_send = 11;}; //data_send จะมีค่าเท่ากับ 11
 if((read_max >= 173)&&(read_max < 181)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 173-180 ตัวแปร
{data_send = 12;}; //data_send จะมีค่าเท่ากับ 12
 if((read_max >= 181)&&(read_max < 199)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 181-198 ตัวแปร
{data_send = 13;}; //data_send จะมีค่าเท่ากับ 13
if((read_max >= 199)&&(read_max < 220)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 199-219 ตัวแปร
{data_send = 14;}; //data_send จะมีค่าเท่ากับ 14
```

```
if((read_max >= 220)&&(read_max < 232)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 220-231 ตัวแปร
{data_send = 15;}; //data_send จะมีค่าเท่ากับ 15
if((read_max >= 232)&&(read_max < 247)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 232-246 ตัวแปร
{data_send = 16;}; //data_send จะมีค่าเท่ากับ 16
if((read_max >= 247)&&(read_max < 256)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 247-255 ตัวแปร
{data_send = 17;}; //data_send จะมีค่าเท่ากับ 17
if((read_max >= 256)&&(read_max < 274)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 256-273 ตัวแปร
{data_send = 18;}; //data_send จะมีค่าเท่ากับ 18
if((read_max >= 274)&&(read_max < 297)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 274-296 ตัวแปร
{data_send = 19;}; //data_send จะมีค่าเท่ากับ 19
if((read_max >= 297)&&(read_max < 311)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 297-310 ตัวแปร
{data_send = 20;}; //data_send จะมีค่าเท่ากับ 20
if((read_max >= 311)&&(read_max < 328)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 311-327 ตัวแปร
{data_send = 21;}; //data_send จะมีค่าเท่ากับ 21
if((read_max >= 328)&&(read_max < 345)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 328-344 ตัวแปร
{data_send = 22;}; //data_send จะมีค่าเท่ากับ 22
if((read_max >= 345)&&(read_max < 357)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 345-356 ตัวแปร
{data_send = 23;}; //data_send จะมีค่าเท่ากับ 23
if((read_max >= 357)&&(read_max < 372)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 357-371 ตัวแปร
{data_send = 24;}; //data_send จะมีค่าเท่ากับ 24
if((read_max >= 372)&&(read_max < 375)) //ถ้าค่าตัวแปร read_max อยู่ในช่วง 372-374 ตัวแปร
{data_send = 25;}; //data_send จะมีค่าเท่ากับ 25
if(read_max>375) ) //ถ้าค่าตัวแปร read_max มีค่ามากกว่า 375 ตัวแปร
{
lcd_clear();//clear หน้าจอ LCD
lcd_gotoxy(0,0);
lcd_putsf("   CHECK_ERROR ");// แสดงข้อมูล จอ LCD "CHACK_ERROR"
data_send = 55; //กำหนดให้ data_send มีค่าเท่ากับ 55
delay_ms(500);
}
delay_us(100);
if (data_send != data_send0)//ตรวจเช็คว่า data_send ไม่เท่ากับ data_send0
```

```
                {
                 cnt_read_ok++;// เพิ่มค่า cnt_read_ok ที่ละ1 ค่า
                 if (cnt_read_ok > 3_) // เพิ่มค่า cnt_read_ok มีค่ามากกว่า 3 หรือไม่
                 {


                  lcd_clear();
                  delay_ms(800);
                   lcd_gotoxy(0,0);
                   lcd_putsf("   Send   Data  ");
                   lcd_gotoxy(0,1);
                   lcd_putsf("                ");// ให้หน้าจอ LCD แสดงค่าว่าง
                   cnt_read_ok=0;
                   delay_ms(200);
                   RELAY=1;
                   delay_ms(800);
                   lcd_gotoxy(0,1);


            sprintf(temp,"Data = $%i%i%i!     ",data_send/100,((data_send%100)/10),(data_send%10));
          //delay_ms(300);
              //ทำการเก็บค่ารวมทั้งตัวอักษร "Data =$___!"จะเก็บค่าตัวแปรโดย
              %;ตัวแรกมีค่าเท่ากับ data_send/100
              %;ตัวที่ 2 มีค่าเท่ากับ (data_send/100)/10
              %;ตัวที่ 3 มีค่าเท่ากับ data_send/10
                   printf("%s ",temp);
                   lcd_puts(temp);
                   delay_ms(200);
                  LED = 0;
                   delay_ms(300);
                   RELAY = 0;
                   val2=data_send;
                   delay_ms(100);
                   data_send0 = val2;
```

```
            delay_ms(500);
            LED = 1 ;
            delay_ms(500);
        };
            };delay_us(500);
        };
    }
```

ภาครับ

```
/*****************************************************
This program was produced by the
CodeWizardAVR V2.03.4 Standard
Automatic Program Generator
© Copyright 1998-2008 Pavel Haiduc, HP InfoTech s.r.l.
http://www.hpinfotech.com

Project :
Version :
Date    : 29/7/2009
Author  :
Company :
Comments:


Chip type           : ATmega32
Program type        : Application
Clock frequency     : 11.059200 MHz
Memory model        : Small
External RAM size   : 0
Data Stack size     : 512
*****************************************************/

#include <mega32.h>
#include <mega32.h>
#include <delay.h>
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

#define  LED  PORTB.3 //กำหนดตัวแปร LED เท่ากับ port PB3 ของไมโครคอนโทรลเลอร์ ATmega32
#define  RELAY  PORTB.4 //กำหนดตัวแปร Relay เท่ากับ port PB4 ของไมโครคอนโทรลเลอร์ ATmega32

#define  strobe   PORTB.0   //กำหนดตัวแปร strobe เท่ากับ port PB0 ของไมโครคอนโทรลเลอร์ ATmega32
#define  clk      PORTB.2    //กำหนดตัวแปร clk เท่ากับ port PB2 ของไมโครคอนโทรลเลอร์ ATmega32
#define  dat      PORTB.1    //กำหนดตัวแปร dat เท่ากับ port PB1 ของไมโครคอนโทรลเลอร์ ATmega32
char data_max[3],temp_data[5],ch,index;
char temp[16];
char outdata =0;
char command_send=0;

#define INCOME  PINB.5 //กำหนดตัวแปร INCOME แบบรับข้อมูล เท่ากับ port PB5 ของไมโครคอนโทรลเลอร์
ATmega32
// Alphanumeric LCD Module functions
#asm
  .equ __lcd_port=0x15 ;PORTC
```

```
#endasm
#include <lcd.h>

#define RXB8 1 //กำหนดขา RXB ของ LCD ให้เป็น port PC1 ของไมโครคอนโทรลเลอร์ ATmega32
#define TXB8 0 //กำหนดขา TXB ของ LCD ให้เป็น port PC0 ของไมโครคอนโทรลเลอร์ ATmega32
#define UPE 2 //กำหนดขา UPE ของ LCD ให้เป็น port PC2 ของไมโครคอนโทรลเลอร์ ATmega32
#define OVR 3 //กำหนดขา OVR ของ LCD ให้เป็น port PC3 ของไมโครคอนโทรลเลอร์ ATmega32
#define FE 4 //กำหนดขา FE ของ LCD ให้เป็น port PC4 ของไมโครคอนโทรลเลอร์ ATmega32
#define UDRE 5 //กำหนดขา UDRE ของ LCD ให้เป็น port PC5 ของไมโครคอนโทรลเลอร์ ATmega32
#define RXC 7 //กำหนดขา RXC ของ LCD ให้เป็น port PC7 ของไมโครคอนโทรลเลอร์ ATmega32

#define FRAMING_ERROR (1<<FE) //กำหนดตัวแปร FRAMING_ERROR แล้วมีการกำหนดตัวแปร FE=1
#define PARITY_ERROR (1<<UPE) //กำหนดตัวแปร PARITY_ERROR แล้วมีการกำหนดตัวแปร UPE=1
#define DATA_OVERRUN (1<<OVR) //กำหนดตัวแปร DATA_OVERRUN แล้วมีการกำหนดตัวแปร OVR=1
#define DATA_REGISTER_EMPTY (1<<UDRE) //กำหนดตัวแปร DATA_REGISTER_EMPTY แล้วมีการ
กำหนดตัวแปร UDRE=1
#define RX_COMPLETE (1<<RXC) //กำหนดตัวแปร RX_COMPLETE แล้วมีการกำหนดตัวแปร RXC=1


// USART Receiver buffer
#define RX_BUFFER_SIZE 8 //กำหนดตัวแปร RX_BUFFER_SIZE มีค่าเท่ากับ 8
char rx_buffer[RX_BUFFER_SIZE]; //กำหนดตัวแปร rx_buffer มีค่าอาเรย์เท่ากับ 8

#if RX_BUFFER_SIZE<256 //ถ้า RX_BUFFER_SIZE<256
unsigned char rx_wr_index,rx_rd_index,rx_counter;
//กำหนดตัวแปร char rx_wr_index,rx_rd_index,rx_counter เป็นชนิด unsigned char
#else//ถ้าไม่ใช่
unsigned int rx_wr_index,rx_rd_index,rx_counter;
//กำหนดตัวแปร char rx_wr_index,rx_rd_index,rx_counter เป็นชนิด unsigned int
#endif

// This flag is set on USART Receiver buffer overflow
bit rx_buffer_overflow;

// USART Receiver interrupt service routine
interrupt [USART_RXC] void usart_rx_isr(void)// ฟังก์ชัน Interrupt
{
char status,data
status=UCSRA; // กำหนดตัวแปร status มีค่าเท่ากับรีจิสเตอร์ UCSRA
data=UDR; // กำหนดตัวแปร data มีค่าเท่ากับรีจิสเตอร์ UDR

if ((status & (FRAMING_ERROR | PARITY_ERROR | DATA_OVERRUN))==0)
```

```
//ถ้ามีค่าในตัวแปร FRAMING_ERROR (ORข้อมูล) กับค่าในตัวแปร PARITY_ERROR (ORข้อมูล) กับค่าในตัวแปร
DATA_OVERRUN และทั้งหมด (AND ข้อมูล) กับตัวแปร status มีค่าเท่ากับศูนย์หรือไม่
  {
   rx_buffer[rx_wr_index]=data; //ถ้าเท่ากับศูนย์ ให้เก็บ data ไว้ในตัวแปร rx_buffer

   lcd_putchar(data); //เก็บค่า data ไว้ใน Buffer LCD
   if (data == '$')//ถ้าค่าdataเท่ากับ $
   {
      index=0;//ให้กำหนดค่า index=0
      ch=0;//ให้กำหนดค่า ch=0
   }
   else if (data == '!') //ถ้าไม่ใช่ data เท่ากับ '!' หรือไม่
   {
      if (index >=2)//ถ้าค่าของตัวแปร index มีค่ามากกว่า 2
      {
      //if(data=='A')
      command_send=1;
      data_max[0] = temp_data[0];
      data_max[1] = temp_data[1];
      data_max[2] = temp_data[2];


      }
   }

   else if ((data >= '0') && (data <='9'))  //ถ้าไม่ใช่ data อยู่ในช่วง 0 ถึง 9 หรือไม่
   {
      temp_data[index] = data; //กำหนดตัวแปร temp_data มีค่าเท่ากับ index
      index++;  // ค่า index เพิ่มขึ้นทีละ 1
   }
   }
   if (++rx_wr_index == RX_BUFFER_SIZE) rx_wr_index=0;
    //ถ้าค่าตัวแปร rx_wr_index มีค่าเพิ่มขึ้น 1 ค่าแล้วมีค่าเท่ากับ RX_BUFFER_SIZE ให้ค่าตัวแปร rx_wr_indexเท่ากับศูนย์
   if (++rx_counter == RX_BUFFER_SIZE)
  //ถ้าค่าตัวแปร rx_counter มีค่าเพิ่มขึ้น 1 ค่าแล้วมีค่าเท่ากับ RX_BUFFER_SIZE
  {
     rx_counter=0; //ให้ rx_counter เท่ากับ 0
     rx_buffer_overflow=1;// ให้ rx_buffer_overflow=1
     };
  };


#ifndef _DEBUG_TERMINAL_IO_
// Get a character from the USART Receiver buffer
```

```
#define _ALTERNATE_GETCHAR_
#pragma used+
char getchar(void) // ประกาศฟังชันก์ย่อย getchar
{
char data;//กำหนดข้อมูล data เป็นแบบตัวอักษร
while (rx_counter==0);//เมื่อ rx_counter มีค่าเท่ากับ 0
data=rx_buffer[rx_rd_index];//ข้อมูล data จะเท่ากับ rx_buffer
if (++rx_rd_index == RX_BUFFER_SIZE) rx_rd_index=0;
#asm("cli")
--rx_counter;//ค่าตัวแปร rx_counter
#asm("sei")
return data;// ส่งค่าในตัวแปร data กลับไป
}
#pragma used-
#endif

// Standard Input/Output functions
#include <stdio.h>


// Declare your global variables here
void strob(void)// ประกาศค่าฟังชันก์ย่อย strob
{

        strobe=1; delay_ms(2);
        strobe=0; delay_ms(2);
}

void pulse(void) // ประกาศค่าฟังชันก์ย่อย pulse
{
        clk=1; delay_ms(2);
        clk=0; delay_ms(2);
}

void display(char dp) // ประกาศค่าฟังชันก์ย่อย display และมีการรับค่าเข้ามาในฟังก์ชันผ่านตัวแปร dp
{
        unsigned char tm,tm2;
        char ch=8;
        tm =  dp;
        tm2 = (dp&0x01);//กำหนดให้ค่าในตัวแปร dp ที่รับเข้ามา AND กับ 1(16) นำค่าที่ได้เก็บไว้ในตัวแปร tm2
        while(ch)
        {
                dat=tm2; delay_ms(2);
                pulse();//เรียกใช้ฟังก์ชัน pule
        tm = tm>>1;//ทำการเลื่อน bit tm ไปทีละ 1 bit แล้วเก็บค่าไว้ในตัวแปร tm
```

```
        tm2 = (tm & 0x01); );//กำหนดให้ค่าในตัวแปร tm ที่รับเข้ามา AND กับ 1₁₆ นำค่าที่ได้เก็บไว้ในตัวแปร tm2
        ch--;  delay_ms(2);

        }
    }


    void main(void)
    {
    // Declare your local variables here

    // Input/Output Ports initialization
    // Port A initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTA=0x00;
    DDRA=0x00;

    // Port B initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTB=0x00;
    DDRB=0x1f;

    // Port C initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTC=0x00;
    DDRC=0x00;

    // Port D initialization
    // Func7=In Func6=In Func5=In Func4=In Func3=In Func2=In Func1=In Func0=In
    // State7=T State6=T State5=T State4=T State3=T State2=T State1=T State0=T
    PORTD=0x00;
    DDRD=0x00;

    // Timer/Counter 0 initialization
    // Clock source: System Clock
    // Clock value: Timer 0 Stopped
    // Mode: Normal top=FFh
    // OC0 output: Disconnected
    TCCR0=0x00;
    TCNT0=0x00;
    OCR0=0x00;

    // Timer/Counter 1 initialization
    // Clock source: System Clock
    // Clock value: Timer 1 Stopped
    // Mode: Normal top=FFFFh
    // OC1A output: Discon.
```

```
// OC1B output: Discon.
// Noise Canceler: Off
// Input Capture on Falling Edge
// Timer 1 Overflow Interrupt: Off
// Input Capture Interrupt: Off
// Compare A Match Interrupt: Off
// Compare B Match Interrupt: Off
TCCR1A=0x00;
TCCR1B=0x00;
TCNT1H=0x00;
TCNT1L=0x00;
ICR1H=0x00;
ICR1L=0x00;
OCR1AH=0x00;
OCR1AL=0x00;
OCR1BH=0x00;
OCR1BL=0x00;

// Timer/Counter 2 initialization
// Clock source: System Clock
// Clock value: Timer 2 Stopped
// Mode: Normal top=FFh
// OC2 output: Disconnected
ASSR=0x00;
TCCR2=0x00;
TCNT2=0x00;
OCR2=0x00;

// External Interrupt(s) initialization
// INT0: Off
// INT1: Off
// INT2: Off
MCUCR=0x00;
MCUCSR=0x00;

// Timer(s)/Counter(s) Interrupt(s) initialization
TIMSK=0x00;
// USART initialization
// Communication Parameters: 8 Data, 1 Stop, No Parity
// USART Receiver: On
// USART Transmitter: Off
// USART Mode: Asynchronous
```

```
// USART Baud Rate: 1200
UCSRA=0x00;
UCSRB=0x90;
UCSRC=0x86;
UBRRH=0x02;
UBRRL=0x3F;

// Analog Comparator initialization
// Analog Comparator: Off
// Analog Comparator Input Capture by Timer/Counter 1: Off
ACSR=0x80;
SFIOR=0x00;

// LCD module initialization
lcd_init(16);//นำค่า 16 เข้าไปในฟังก์ชัน lcd_init

// Global enable interrupts
#asm("sei")
PORTB.3 = 1;
while (1)
    {
    //==============receive from client ============

          if(INCOME==0)
          {
           LED = 0;
           delay_ms(1000);
           LED = 1;
          };

            lcd_clear();
            outdata = ((data_max[0]-'0')*100) + ((data_max[1]-'0')*10)+((data_max[2]-'0')*1);
            if(outdata == 48)
```

//นำค่าตัวแปร data_max[0] ลบด้วย '0'หรือ 48 คูณด้วย 100 บวกกับ data_max[1]ลบด้วย '0'หรือ 48 คูณด้วย 100 บวก
กับ data_max[2] ลบด้วย '0'หรือ 48 คูณด้วย 1

เช่น data_max[0]=48

data_max[1]=50

data_max[2]=49

จะคำนวณได้ outdata = ((48-48)*100)+((50-48)*100)+((49-48)*1)=021

```
{
          outdata = 0;
          }

          if(outdata == 55)
          {
          lcd_clear();
          lcd_gotoxy(0,0);
```

กำหนดความเร็วในการรับข้อมูลเท่ากับ 1200 bps โดยคำนวณจาก XTAL 11.0592 MHz

```
                lcd_putsf("   CHECK_ERROR  ");//แสดงผล LCD
                lcd_gotoxy(0,1);
                lcd_putsf("                ");// แสดงผล LCD
                delay_ms(1500);
                outdata = 0xff;
                }


                lcd_gotoxy(0,0); //กำหนดตำแหน่งแสดง LCD

                sprintf(temp,"DATA MAX = %icm.",(char)outdata);
                //กำให้ตัวแปร temp เก็บตัวอักษร "DATA MAX =  cm"และค่าของ outdata ไว้ในตัวแปร
                lcd_puts(temp); //แสดงค่าตัวแปร temp ทาง LCD
                delay_ms(20);
                lcd_gotoxy(0,1); ); //กำหนดตำแหน่งแสดง LCD
                sprintf(temp,"                ",(char)outdata);  //กำให้ตัวแปร temp เก็บเก็บค่าว่าง
                lcd_puts(temp); //แสดงค่าว่าง ทาง LCD

                display(0x01);// ส่งค่า 1 ให้กับฟังก์ชัน display
                display(outdata);// ส่งค่าตัวแปร outdata ใหกับฟังก์ชัน display
                strob(); //เรียกใช้ฟังชั่น strob
                delay_ms(500);



        }delay_ms(100);

    }
```

ภาคผนวก ค

อุปกรณ์ที่ใช้ในการสร้างบอร์ดไมโรคอนโทรลเลอร์

แบ่งได้ 2ส่วนคือ

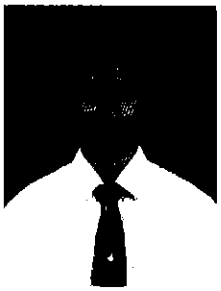
1.ตัวส่งและประมวลผลข้อมูล

	จำนวน
1. ไมโครคอนโทลเลอร์ AVR Mega32	1
2. จอแสดงผล LCD	1
3. Header 5x2	11
4. IC ULN2003	1
5. IC 7805	1
6. ไมโครสวิทซ์	1
7. ไดโอด IN4001	1
8. หลอด LED (red)	1
9. หลอด LED (green)	1
10. ตัวเก็บประจุ 100uf ,1000uf	1
11. ตัวเก็บประจุ 100pf	2
12. ตัวกำเนิดสัญญาณนาฬิกา 11.0592 MHz	1
13. ตัวต้านทานปรับค่าได้ 2k,1k	1
14. ตัวต้านทานปรับค่าได้อย่างละเอียด 20 รอบ 500	1
15. ตัวต้านทาน 450	2
16. ตัวต้านทาน 10k	1
17. DC_Jack	1
18. W04M	1
19. Header 3x1	1
20. Relay 5V.	1

2.ตัวรับข้อมูล

	จำนวน
1. ไมโครคอนโทลเลอร์ AVR Mega32	1
2. จอแสดงผล LCD	1
3. Header 5x2	4
4. IC ULN2003	1
5. IC 7805	1
6. ไมโครสวิทซ์	1
7. ไดโอด IN4001	1
8. หลอด LED (red)	1
9. หลอด LED (green)	1
10. ตัวเก็บประจุ 100uf ,1000uf	1
11. ตัวเก็บประจุ 100pf	2
12. ตัวกำเนิดสัญญาณนาฬิกา 11.0592 MHz	1
13. ตัวต้านทานปรับค่าได้ 1k	1
14. IC 4094	1
15. ตัวต้านทาน 450	2
16. ตัวต้านทาน 10k	1
17. DC_Jack	1
18. W04M	1
19. Header 3x1	1
20. Relay 5V.	1

ประวัติผู้เขียนโครงงาน



ชื่อ นายทวีชาติ ธนกุลปริยา

ภูมิลำเนา 116/1 ม.1 บ.ท่าฟ้าเหนือ ต.สระ อ.เชียงม่วน จ.พะเยา

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนท่าฟ้าเหนือ
- จบระดับมัธยมศึกษาจากโรงเรียนปงรัชดาภิเษก
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
- สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : tory_gvnine@hotmail.com

ชื่อ นายธนากร วรินทรา

ภูมิลำเนา 444/6 ต.เชียงคาน อ.เชียงคาน จ.เลย 42110

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนเมืองเลย
- จบระดับมัธยมศึกษาจากโรงเรียนจุฬาภรณ์ราชวิทยาลัย เลย
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
- สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail : electrical_worm@hotmail.com