ระบบควบคุมกล้องและส่งภาพผ่านเครือข่ายอินเตอร์เน็ต สำหรับเทคโนโลยีการรักษาความปลอดภัย

THE CAMERA CONTROL AND IMAGE TRANSMISSION VIA INTERNET NETWORK FOR THE SECURITY TECHNOLOGY

โดย

นายสุรศักดิ์ สังข์วัฒนะ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2551

ระบบควบคุมกล้องและส่งภาพผ่านเครือข่ายอินเตอร์เน็ต
สำหรับเทคโนโลยีการรักษาความปลอดภัย
THE CAMERA CONTROL AND IMAGE TRANSMISSION
VIA INTERNET NETWORK FOR THE SECURITY TECHNOLOGY

โดย

นายสุรศักดิ์ สังข์วัฒนะ รหัส 49015173

อาจารย์ที่ปรึกษา

รศ.ดร.มนัส สังวรศิลป์

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2551

ปริญญานิพนธ์ ปีการศึกษา 2551

ภาควิชา อิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ระบบควบคุมกล้องและส่งภาพผ่านเครือข่ายอินเตอร์เน็ตสำหรับเทคโนโลยี
การรักษาความปลอดภัย

THE CAMERA CONTROL AND IMAGE TRANSMISSION VIA
INTERNET NETWORK FOR THE SECURITY TECHNOLOGY

ผู้จัดทำ

1. นายสุรศักดิ์ สังข์วัฒนะ รหัส 49015173

.....อาจารย์ที่ปรึกษา

(รศ.ดร. มนัส สังวรศิลป์)

ระบบการควบคุมกล้องและส่งภาพผ่านเครือข่ายอินเตอร์เน็ต สำหรับเทคโนโลยีการรักษาความปลอดภัย

นายสุรศักดิ์ สังข์วัฒนะ รหัส 49015173
รศ.ดร.มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2551

บทคัดย่อ

รายงานฉบับนี้จัดทำขึ้นเพื่ออธิบายการออกแบบและสร้าง เครื่องควบคุมกล้องและส่งภาพผ่านเครือข่ายอินเตอร์เน็ตโดยใช้โมดูลอินเตอร์เน็ตไมโครคอนโทรเลอร์ EREAX 100LX RISC MODULE ซึ่งเป็นอุปกรณ์ที่สามารถโปรแกรมข้อมูลที่ใช้ในการส่งภาพและควบคุมการเคลื่อนไหวของกล้อง ผ่านทางเครือข่ายอินเตอร์เน็ตโดยพัฒนาด้วยโปรแกรมภาษา C และ HTML โดยการส่งข้อมูลแบบ CGI (Command Gateway Interface)ผ่านโพรโตคอล ทีซีพี /ไอพี (TCP/IP Protocol) ซึ่งในส่วนของการส่งภาพใช้กล้องเว็บแคมเชื่อมต่อแบบ USB กับไมโครคอนโทรเลอร์ EREAX 100LX RISC ส่งไปยังผู้ใช้งานได้เห็นภาพจากโปรแกรม JAVA บนหน้าเว็บ(Webpage)ในส่วนการควบคุม เซอร์โวมอเตอร์ควบคุมการเคลื่อนไหวของกล้อง โดยรับคำสั่งจากหน้าเว็บแล้วส่งข้อมูลให้กับไมโครคอนโทรเลอร์ EREAX 100LX รับข้อมูลและส่งให้ MCS-51สร้างสัญญาณควบคุมเซอร์โวมอเตอร์ ทำให้เราสามารถเห็นภาพและควบคุมกล้องได้จากหน้าเว็บจากทุกที่ที่สามารถเข้าถึงอินเตอร์เน็ต

THE CAMERA CONTROL AND IMAGE TRANSMISSION VIA INTERNET NETWORK FOR THE SECURITY TECHNOLOGY

Mr.Surasak Sangwatthana ID. 49015173

Assoc. Prof. Dr. Manas Sangworasilp Advisor

Educational Year 2008

Abstract

The objective of this project is to design a camera control and graphics transfer system through an internet using module internet microcontroller EREAX 100LX RISC module programs data is for graphics transferred and cameras controlled. The system works through an internet network, developed by C and HTML languages by transferring CGI (Command Gateway Interface) data through TCP/IP Protocol. A web cam camera is connected through USB port and a microcontroller EREAX 100LX RISC. A user views the transferred graphics from JAVA on the webpage. For a camera movement, a servo motor controlled from the webpage transfers data to microcontroller EREAX 100LX which receives and transfers data to MCS-51 that generates signals to control this servo motor. The graphics then can be observed and the camera can be controlled from a webpage from anywhere that is accessible through an internet.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดีเนื่องจากได้รับการช่วยเหลือจาก รศ.ดร.มนัส สังวรศิลป์ อาจารย์ที่ปรึกษา และพี่นักศึกษาปริญญาโท ที่ได้คอยให้คำปรึกษาในเรื่องต่าง ๆ ที่ไม่เข้าใจ ทั้งยังช่วยสนับสนุนในเรื่องของสถานที่ในการทำโปรเจคและ เครื่องมือต่าง ๆ เป็นอย่างดี จึงขอกราบขอบพระคุณบุคคลที่ได้กล่าวไปข้างต้นนี้เป็นอย่างสูงไว้ ณ โอกาสนี้ และสุดท้ายขอกราบขอบพระคุณบิดามารดาที่ได้คอยอุปการะในเรื่องการเรียนด้วยดีตลอดมา ทั้งคอยดูแลและให้ความช่วยเหลือในทุก ๆ เรื่อง

ผู้จัดทำ

นายสุรศักดิ์ สังข์วัฒนะ

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

หน้า

สารบัญรูปภาพ (ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญ

ในยุคปัจจุบันนี้มนุษย์ได้พัฒนาเทคโนโลยีเพื่ออำนวยความสะดวกในการใช้ชีวิตประจำวันมากมายจนเรียกได้ว่าแทบจะตามไม่ทัน โดยเฉพาะเทคโนโลยีทางด้าน การสื่อสาร เช่น โทรศัพท์มือถือ Internet ที่นับวันจะเป็นส่วนหนึ่งของชีวิตประจำวันไปเลยก็ได้ เมื่อ Internet ได้เข้ามามีบทบาทกับมนุษย์เรามากขึ้น ผู้จัดทำโครงงานจึงเล็งเห็นว่าควรที่จะประยุกต์ใช้ประโยชน์จาก การติดต่อสื่อสารทางด้าน Networkให้มากที่สุดก็คงจะเป็น การรักษาความปลอดภัยและการควบคุมระยะไกล และเนื่องจากว่าทรัพย์สินที่มนุษย์หามาได้ด้วยความยากลำบากนั้น ถ้าเกิดความเสียหาย หรือถูกลักขโมยไปก็จะต้องเกิดความเสียใจ อย่างที่สุดเพื่อแก้ไขปัญหานี้ จึงเกิดโครงงานนี้ขึ้นมา โดยจะใช้ EREAX 100LX RISC MODULE ซึ่งมีความสามารถที่เป็นได้ทั้งไมโครคอลโทรลเลอร์ และเป็น Server ให้บริการแก่เครื่องคอมพิวเตอร์ ในการออกแบบนั้นผู้จัดทำโครงงานจะใช้จะใช้ Modules นี้เป็นเครื่องที่คอยส่งภาพและควบคุมอุปกรณ์ และคอยรักษาความปลอดภัยอยู่ที่บ้าน โดยเมื่อใดที่เจ้าของบ้านอยากดูว่าบ้านของตนเป็นอย่างไรบ้างก็สามารถดูได้จากโปรแกรมที่เขียนขึ้นโดย ผ่านหน้าเว็บ (Webpage)

1.2 วัตถุประสงค์

1) เพื่อศึกษาวิธีการสื่อสารผ่านเครือข่ายอินเตอร์เน็ต

2) เพื่อศึกษาการทำงานของ EREAX 100LX RISC MODULE

3) เพื่อประยุกต์ใช้งานเครือข่ายอินเตอร์เน็ต ให้เกิดประโยชน์ทางด้านการรักษาความปลอดภัย

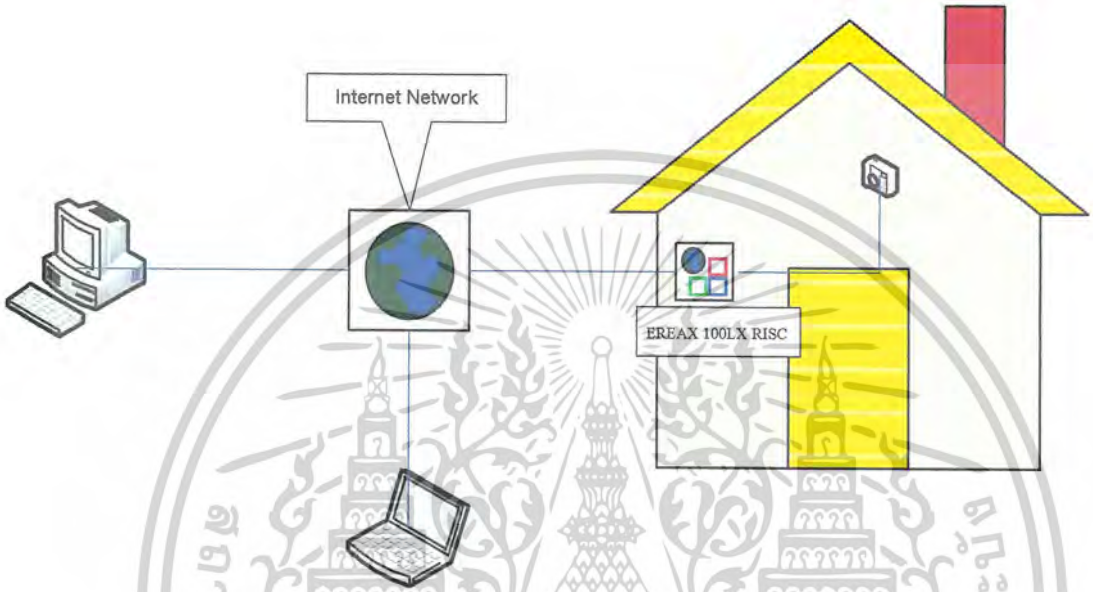
4) เพื่อนำความรู้จากการเรียน วิชา Microcontroller มาประยุกต์ใช้งานด้านการควบคุม

5) เพื่อนำความรู้จากการเรียน วิชา Data Communication มาประยุกต์ใช้งานด้านการควบคุม

6) เพื่อฝึกทักษะการแก้ปัญหาเฉพาะหน้าในการทำงานจริง

7) เพื่อเตรียมความพร้อมก่อนที่จะได้ออกไปทำงาน ทางด้านวิศวกรรม

1.3 ขอบเขตของโครงงาน



รูปที่ 1.1 แสดงขอบเขตของโครงงาน

จากรูปที่ 1.1 แสดงภาพรวมของระบบควบคุมและรักษาความปลอดภัยภายในบ้านผ่านระบบอินเตอร์เน็ต โดย EREAX 100LX RISC MODULE ที่เป็นทั้ง Microcontroller และเป็น Server ที่คอยให้บริการแก่ผู้ใช้จากภายนอก จากรูปเราออกแบบให้ Module ตัวนี้สามารถควบคุมการส่งภาพและควบคุมการหมุนของกล้องสามารถหมุนปรับมุมมองของกล้อง ซึ่งก็จะสามารถทราบได้ว่ามีใครมาหาโดยมองภาพผ่านหน้าเว็บ (Webpage) โดยโปรแกรมที่เขียนขึ้นด้วย CGI (Command Gateway Interface)

บทที่2
ทฤษฎีที่เกี่ยวข้อง

2.1 TCP/IP (Transmission Control Protocol/internet Protocol)

หลายเทคโนโลยีที่เราท่านใช้อยู่ทั่วไปมีจุดกำเนิดจากเทคโนโลยีการสงคราม IP เน็ตเวิร์คก็เป็นหนึ่งในนั้นเมื่อครั้งสงครามเย็นระหว่างสหรัฐอเมริกาและสหภาพโซเวียตกระทรวงกลาโหมภายใต้รัฐบาลกลางสหรัฐฯ ได้จ้างมหาวิทยาลัยต่างๆทำวิจัยเพื่อสร้างเครือข่ายที่ทนต่อความล้มเหลว(ด้วยระเบิดนิวเคลียร์)สิ่งที่ได้คือโพรโตคอลTCP/IPเครือข่ายคอมพิวเตอร์ที่ใช้โพรโตคอลนี้เรียกสั้น ๆว่า TCP/IP (Transmission Control Protocol/Internet Protocol) คือชุดของโพรโตคอลที่ รวมกันเป็นกลุ่มให้ใช้งานเช่น Internet Protocol (IP) , Address Resolution Protocol (ARP), Internet Control Message Protocol (ICMP), User Datagram Protocol (UDP) ฯลฯ แต่โพรโตคอลที่ มีบทบาทสำคัญคือ Internet Protocal (IP) โดยมีหลักการทำงานคือ แบ่งเนื้อข้อมูลที่ต้องการส่งเป็น ชิ้นเล็ก ๆ เรียกว่าแพ็กเก็ต ส่งแพ็กเก็ตไปยังเส้นทางที่เหมาะสมเป็นทอดจนกว่า จะถึงปลายทางแต่ละแพ็กเก็ตอาจใช้เส้นทางคนละทิศขึ้นกับการพิจารณาของเราเตอร์ในช่วงต่างๆ หากเกิด ข้อผิดพลาด ณ ช่วงการส่งใด เราเตอร์ที่รับผิดชอบการส่งช่วงนั้นจะจัดส่งแพ็กเก็ตชิ้นนั้นใหม่โดยอัตโนมัติเมื่อถึงจุดหมายระบบปลายทางจะรวบรวมแพ็กเก็ตกลับให้เป็น เนื้อข้อมูลดังเดิม ซึ่งถ้าจะว่ากันตามทฤษฎีแล้ว TCP/IP นั้นจะประกอบด้วยส่วนสำคัญอยู่ 2 ส่วน ด้วยกันก็คือ TCP หรือ Transmission Control Protocol และอีกส่วนก็คือ IP หรือ Internet Protocol นั่นเอง การแบ่งลักษณะในการทำงานก็จะแบ่งเป็นTCP มีหน้าที่ในการตรวจสอบการรับส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์ผู้รับและเครื่องคอมพิวเตอร์ผู้ส่งให้ได้รับขอมูลที่ถูกต้องและครบถ้วนหรือว่าหากมีการสูญหายของข้อมูลก็จะมีการแจ้งให้ต้นทางที่ส่งข้อมูลมารับทราบแล้วให้ทำการส่งข้อมูลมาให้ใหม่

2.2 ลักษณะการทำงาน

ลักษณะการทำงานของ IP นั้น จะทำหน้าที่ในการเลือกเส้นทางที่จะใช้ในการรับส่งข้อมูลใน ระบบเครือข่าย และทำการตรวจสอบที่อยู่ของผู้รับ โดยการใช้ข้อมูลขนาด 4 Byte เป็นตัวกำหนด แอดเดรสหรือที่เราเรียกกันว่า IP Address ซึ่งโพรโตคอล TCP จะทำงานอยู่ในชั้น Transport Layer ตัวแพ็กเก็ต TCP จะประกอบด้วย ส่วนหัว (Header) และสวนข้อมูล (Data) และโพรโตคอล IP จะ ทำงานอยู่ในชั้น Network Layer ตัวแพ็กเก็ต IP ประกอบด้วย 2 ส่วนใหญ่ ๆ คือ ส่วนหัว (IP Header) จะประกอบด้วย IP แอดเดรสของเครื่องต้นทางและปลายทาง และส่วนข้อมูล

(IP Data) จะ เป็นที่เก็บโพรโตคอล TCP เนื่องจากโพรโตคอล TCP/IP จะถูก Encapsulate ให้มาอยู่ในส่วนของ แพ็กเก็ต IP

2.3 จุดเด่นของโพรโตคอล TCP/IP

2.3.1 สามารถนำส่งข้อมูลไปถึงจุดหมายได้แม้เส้นทางบางที่เสียหาย : เป็นจุดประสงค์หลักที่ ช่วยให้ทนต่อความล้มเหลว โดยหากระหว่างการสื่อสารข้อมูลและมีเส้นทางใดเสียหายหรือล้มเหลว IP เน็ตเวิร์กจะปรับใช้เส้นทางอื่นที่ทดแทนได้เพื่อนำส่งข้อมูลให้ไปถึงปลายทางอยาง อัตโนมัติ ผู้ส่งและผู้รับข้อมูลไม่จำเป็นต้องรับรู้หรือปรับตัวแต่ประการใด

2.3.2 ไม่ขึ้นกับแพลตฟอร์มใด ๆ ไม่ว่าเครือขายนั้นเป็นเครือขายท้องถิ่นหรือเครือข่ายระหว่าง ภูมิภาค เป็นไฟล์ / พรินต์เซิร์ฟเวอร์หรือไคลเอ็นต์/เซิร์ฟเวอร์ เป็นระบบปฏิบัติการใด เน็ตเวิร์ก อินเทอร์เฟซเป็นแบบใดก็ตาม ในมุมมองของโพรโตคอล TCP/IP ก็คือ IP เน็ตเวิร์ก

2.4 จุดอ่อนของโพรโตคอล TCP/IP

2.4.1 รับส่งโดยไม่มีการรักษาความปลอดภัยเนื้อข้อมูล : การรับส่งข้อมูลด้วย IP แพ็กเก็ตไม่มี ทั้งการเข้ารหัสข้อมูลและป้องกันการปลอมแปลงใด ๆ การไม่เข้ารหัสข้อมูลอาจทำให้ผู้ไม่ประสงค์ ดีระหว่างเส้นทางที่ IP แพ็กเก็ตผานดักลอบดูเนื้อขอมูลอย่างง่ายดาย แม้ว่าเราอาจสามารถบังคับ เส้นทางของ IP แพ็กเก็ตได้ก็ไม่อาจมั่นใจได้ว่าระหว่างทางมีการดักลอบดูหรือไม่

ในเรื่องปัญหาการปลอมแปลงแบ่งออกเป็นสองกรณีคือ การปลอมแปลงหรือดัดแปลงเนื้อ ข้อมูล และการปลอมแปลงส่วนหัวของ IP แพ็กเก็ต ทั้งสองกรณีให้ผลเหมือนกันคือผู้รับได้ขอมูลที่ ผิดจากความเป็นจริง ว่าจุดประสงค์ต่างกัน หากเป็นกรณีแรกนั้น ผู้ไม่หวังดีต้องการหลอกหรือ กลั่นแกล้งให้ได้ข้อมูลผิด ๆ หากเป็นกรณีหลัง ผู้ไม่หวังดีต้องการแอบอ้างว่าข้อมูลนั้นมากจาก แหล่งที่ผู้รับไว้ใจหรือแหลงอื่นที่กลายเป็นเหยื่อของการแอบอ้างโดยไม่รู้ตัว

2.4.2 รับส่งโดยไม่คำนึงถงคุณภาพการให้บริการ: การรับส่งต่อ IP แพ็กเก็ตระหว่างเครือขาย ย่อยไปเป็นทอดนั้นใช้หลักการใครมาก่อนได้ก่อน ฉะนั้นจึงคาดเดาไม่ได้ว่าข้อมูลที่นำสงไปจะไป ถึงปลายทางเมื่อใด แม้ว่า IP เน็ตเวิร์กใช้หลักการเลือกเส้นทางที่เหมาะสมที่สุดในขณะนั้นก็ตาม หากแต่ความเหมาะสมนั้นผู้ส่งและผู้รับไม่อาจคาดการณ์หรือมีส่วนร่วมตัดสินใจได้เลยว่าจะช้าเร็ว หรือมีโอกาสที่ข้อมูลผิดพลาดมากน้อยเพียงไร

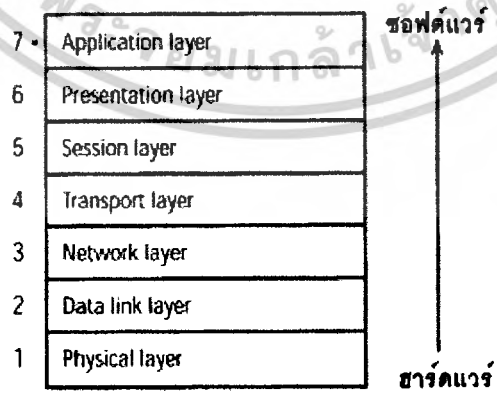
สำหรับเรื่องการรับส่งโดยไม่มีการรักษาความปลอดภัยเนื้อข้อมูลนั้น องค์กรกลางของอินเทอร์เน็ตได้ออกมาตรฐานที่ช่วยแก้ไขปัญหานี้คือ IPSec โดยมีทั้งการเข้าและถอดรหัสเนื้อข้อมูลในระดับ IP แพ็กเก็ตการตรวจสอบความถูกต้องเนื้อข้อมูลและการพิสูจน์ตนของ IP แพ็กเก็ต เพื่อป้องกันการปลอมแปลง

2.5 มาตรฐานการสื่อสารข้อมูล

การกำหนดมาตรฐานของการสื่อสารข้อมูลนั้น นับว่ามีความจำเป็นอย่างมากสำหรับระบบ เครือข่ายที่มี องค์ประกอบของอุปกรณ์ต่างๆ หลากหลายผู้ผลิต ซึ่งอุปกรณ์ทั้งหมดเหล่านั้นจะต้อง ทำงานเข้ากันได้อย่างราบรื่น การกำหนดมาตรฐานต่างๆ นั้นจะเริ่มตั้งแต่โครงสร้างพื้นฐานของ ฮาร์ดแวร์ระบบเครือข่าย ได้แก่ ระบบสายเคเบิล อุปกรณ์ในการส่งสัญญาณข้อมูล ตลอดจนถึง เครื่องเซิร์ฟเวอร์ และซอฟต์แวร์ในการสื่อสารบนระบบเครือข่าย เพื่อเป็นการรับประกันว่า ส่วนประกอบต่างๆ จะสามารถทำงานร่วมกันได้ ผู้ผลิตฮาร์ดแวร์และซอฟต์แวร์ระบบเครือข่าย จะต้องทำตามคำแนะนำตามมาตรฐานการออกแบบและสร้างผลิตภัณฑ์ ซึ่งกำหนดขึ้นโดย องค์กร มาตรฐานสากล (International Organization for Standardization - ISO) โดยมาตรฐานที่กำหนดขึ้น และได้ประกาศใช้ตั้งแต่ปี คศ.1984 เรียกว่า Open Systems Interconnection Reference Model เรียก สั้นๆ ว่า OSI Reference Model หรือ ISO/OSI Model

2.6 แบบจำลอง OSI

OSI Reference Model เป็นการกำหนดชุดของคุณลักษณะเฉพาะที่ใช้อธิบายโครงสร้างของ ระบบเครือข่าย โดยมีวัตถุประสงค์ เพื่อให้ผู้ผลิตฮาร์ดแวร์หรือซอฟต์แวร์ใดๆ ใช้เป็นโครงสร้างอ้างอิงในการสร้างอุปกรณ์ให้สามารถทำงานร่วมกันได้อย่างดีบนระบบเครือข่าย โดยมีการ จัดแบ่งเลเยอร์ของ OSI ออกเป็น 7 เลเยอร์ แต่ละเลเยอร์จะมีการโต้ตอบหรือรับส่งข้อมูลกับเลเยอร์ ที่อยู่ข้างเคียงเท่านั้น โดยเลเยอร์ที่อยู่ชั้นล่างจะกำหนดลักษณะของอินเตอร์เฟส เพื่อให้บริการกับเลเยอร์ที่อยู่เหนือขึ้นไปตามลำดับขั้น เริ่มตั้งแต่ส่วนล่างสุดซึ่งเป็นการจัดการลักษณะทางกายภาพของ ฮาร์ดแวร์และการส่งกระแสของข้อมูลในระดับบิต ไปสิ้นสุดที่แอพพลิเคชั่นเลเยอร์ในส่วนบนสุด



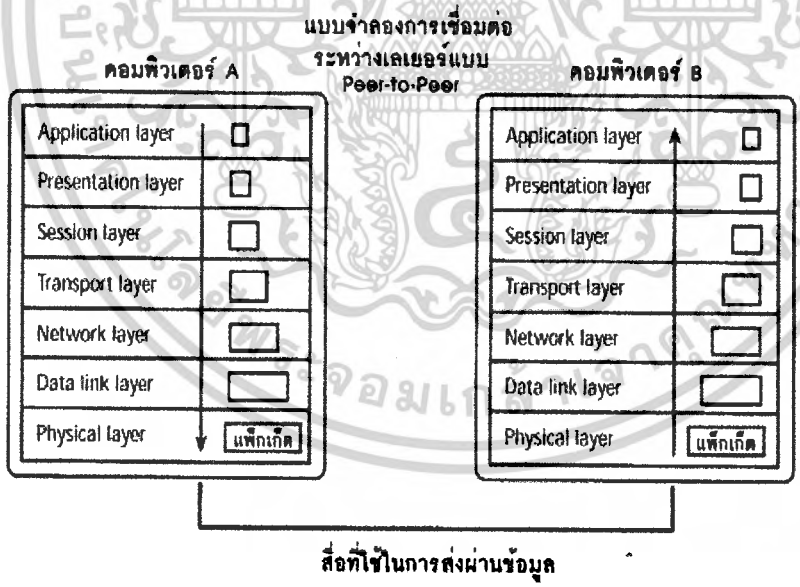
รูปที่ 2.1 OSI Reference Model

2.6.1 หลักการออกแบบเลเยอร์

- แต่ละเลเยอร์จะมีการกำหนดการทำงานอย่างละเอียดโดยมีการทำงานเป็นอิสระไม่ขึ้นต่อกัน
- ฟังก์ชันภายในเลเยอร์จะพยายามมุ่งไปสู่ข้อกำหนดมาตรฐาน (standard protocol)
- ขอบเขตของเลเยอร์จะถูกเลือกและจำกัดให้มีปริมาณการเชื่อมต่อระหว่างเลเยอร์ให้น้อยที่สุด
- จำนวนของเลเยอร์ต้องมากพอที่จะแยกฟังก์ชั่นที่จำเป็นและแตกต่างกันไม่ให้อยู่ในเลเยอร์เดียวกัน

2.6.2 การทำงานของ OSI Reference Model

การที่แพ็กเก็ตข้อมูลเดินทางจากเครื่องคอมพิวเตอร์ A ไปยังเครื่องคอมพิวเตอร์ B นั้น มีกระบวนการทำงานดังนี้



รูปที่ 2.2 การส่งแพ็กเก็ตใน OSI Reference

จากแผนผัง คอมพิวเตอร์ A และคอมพิวเตอร์ B มีโครงสร้างเป็น OSI ซึ่งมี 7 เลเยอร์ เมื่อเครื่อง คอมพิวเตอร์ A พร้อมที่จะส่งสัญญาณข้อมูลไปยังเครื่องคอมพิวเตอร์ B นั้น แต่ละเลเยอร์ในเครื่อง คอมพิวเตอร์ A จะเสมือนกับว่ามีการส่อสารกับเลเยอร์ในระดับเดียวกันบนเครื่องคอมพิวเตอร B ถึงแม้ว่าจะไม่มีการสื่อสารระหว่าง เลเยอร์เหล่านี้เกิดขึ้นจริง แต่เลเยอร์ในระดับ

ต่างๆ บนเครื่อง คอมพิวเตอร์ทั้งคู่นั้นจะทำตามกฎเกณฑ์หรือโพรโตคอล (protocol) อย่างเดียวกัน เพื่อให้มั่นใจได้ว่าแต่ละเลเยอร์บนเครื่องคอมพิวเตอร์ฝ่ายผู้รับจะได้รับแพ็กเก็ตข้อมูลแบบเดียวกันกับแพ็กเก็ต ข้อมูลที่รวบรวม โดยแต่ละเลเยอร์บนเครื่องคอมพิวเตอร์ฝ่ายผู้ส่ง โดยแพ็กเก็ตข้อมูลจะเริ่มที่ระดับ สูงสุดคือ Application Layer บนเครื่องคอมพิวเตอร์ A และเคลื่อนลงมาทีละระดับชั้นจนมาถึงชั้น ล่างสุดคือ Physical Layer การที่แพ็กเก็ตเคลื่อนผ่านจากระดับหนึ่งไปยังระดับถัดไปนั้น จะมีการ กำหนดที่อยู่ การจัดรูปแบบของข้อมูลและอื่นๆ ซึ่งแต่ละเลเยอร์จะเป็นตัวจัดการและมีกระบวนการ ของตนเอง เมื่อแพ็กเก็ตเคลื่อนตัวลงมาถึง Physical Layer ก็จะถูกแปลงให้เป็นกระแสข้อมูลแบบ อนุกรมและส่งผานสื่อกลางคือสายสัญญาณ ซึ่งเป็นเลเยอร์เดียวที่เครื่องคอมพิวเตอร์ A สื่อสารกับ เครื่องคอมพิวเตอร์ B และเมื่อสัญญาณข้อมูลมาถึงเครื่องคอมพิวเตอร์ B กระบวนการก็จะเริ่มทำใน ทางตรงข้าม คือจะทำการแยกแพ็กเก็ตออกผ่าน OSI ทั้ง 7 เลเยอร์ ส่งย้อนกลับขึ้นไปยัง Application Layer ของเครื่องคอมพิวเตอร์ B เมื่อแพ็กเก็ตเดินทางผ่านเลเยอร์ระดับต่างๆ แต่ละเลเยอร์จะแยก ข้อมูลข่าวสารตามกำหนดที่อยู่ และการจัดรูปแบบของแพ็กเก็ต จนเมื่อมาถึงเลเยอร์ระดับสูงสุดคือ Application Layer ก็จะเหลือเฉพาะข้อมูลที่เหมือนกับบน Application Layer ของเครื่อง คอมพิวเตอร์ A

เลเยอร์ 1: Appplication Layer

Layer นี้เป็นการแลกเปลี่ยนข้อมูลระหว่างโปรแกรมที่ทำงานอยู่บนคอมพิวเตอร์และบริการ อื่นๆ ในเครือข่าย เช่น Database

เลเยอร์ 2: Data Link Layer

เลเยอร์นี้มีจุดประสงค์หลักคือพยายามควบคุมการส่งข้อมูลให้เสมือนกับว่าไม่มีความผิดพลาด เกิดขึ้น เพื่อให้เลเยอร์สูงขึ้นไปสามารถนำข้อมูลไปใช้ได้อย่างถูกต้อง วิธีการคือฝ่ายผู้ส่งจะทำการ แตกข้อมูลออกเป็นเฟรมข้อมูล (data-frame) โดยจะต้องมีการกำหนดขอบเขตของเฟรม (frame boundary) โดยการเติมบิทเข้าไปยังจุดเริ่มต้นและจุดสิ้นสุดของเฟรม จากนั้นทำการส่งเฟรมข้อมูล ออกไปทีละชุดและรอรับการตอบรับ (acknowledge frame) จากผู้รับ ถ้าหากมีการสูญหายของเฟรม ข้อมูล ซึ่งอาจเนื่องมาจากสัญญาณรบกวนจากภายนอกหรือข้อผิดพลาดอื่นๆ ในกรณีนี้ฝ่ายผู้ส่ง จะต้องส่งเฟรมข้อมูลเดิมออกมาใหม่

เลเยอร์ 3: Network Layer

เป็นเลเยอร์ที่ทำหน้าที่หลักเกี่ยวข้องกับการหาเส้นทาง (routing) ในการส่งแพคเก็ตจากต้นทาง ไปยังปลายทาง ซึ่งจะมีการสลับช่องทางในการส่งข้อมูลหรือที่เรียกว่า แพ็กเกตสวิตชิ่ง (packet switching) มีการสร้างวงจรเสมือน (virtual circuit) ซึ่งคล้ายกับว่ามีเส้นทางเชื่อมโยงกันระหว่าง คอมพิวเตอร์ 2 เครื่องให้ติดต่อสื่อสารถึงกันได้โดยตรง การกำหนดเส้นทางการส่งข้อมูล

นั้น คอมพิวเตอร์ฝ่ายผู้ส่งอาจทำหน้าที่พิจารณาหาเส้นทางที่เหมาะสมในการส่งข้อมูล ตั้งแต่ต้น หรือ อาจใช้วิธีแบบไดนามิกส์ (dynamic) คือแต่ละแพคเก็ตสามารถเปลี่ยนแปลงเส้นทางได้ตลอดเวลา นอกจากนี้เครื่องคอมพิวเตอร์ฝ่ายผู้ส่งยังมีหน้าที่ในการจัดการเรื่องที่อยู่ของเครือข่าย ปลายทางโดย จะมีการแปลงที่อยู่แบบตรรกะ (logical address) ให้เป็นที่อยู่แบบกายภาพ (physical address) ซึ่งถูก กำหนดโดยการ์ดเชื่อมต่อระบบเครือข่าย

เลเยอร์ 4: Transport Layer

Transport Layer ทำหน้าที่เสมือนบริษัทขนส่งที่รับผิดชอบการจัดส่งข้อมูลโดยปราศจากความ ผิดพลาด ซึ่งมีหน้าที่หลักคือ การตรวจสอบและแก้ไขความผิดพลาดที่เกิดขึ้นในข้อมูล คอยแยกแยะ และจัดระเบียบของแพ็กเก็ต ข้อมูลให้จัดเรียงลำดับอย่างถูกต้อง และมีขนาดที่เหมาะสม นอกจากนี้ ยังทำการผนวกข้อมูลทั้งหลายให้อยู่ในรูปของ วงจรเดียวหรือเรียกว่าการมัลติเพล็กซ์ (multiplex) และมีกลไกสำหรับควบคุมการไหลของข้อมูลให้มีความสม่ำเสมอ

เลเยอร์ 5: Session Layer

จากเลเยอร์ที่ผ่านมาจะเห็นว่าการทำงานต่างๆ จะเกี่ยวพันอยู่เฉพาะกับบิตและข้อมูลเท่านั้น โดยไม่ได้สนใจเกี่ยวกับสถานะภาพการใช้งานจริงของผู้ใช้แต่อย่างใด ซึ่งหน้าที่ดังกล่าวนี้จะเกิดขึ้น ที่ Session Layer ในเลเยอร์นี้จะมีการให้บริการสำหรับการใช้งานเครื่องที่อยู่ห่างไกลออกไป (remote login) การถ่ายโอนไฟล์ระหว่างเครื่อง โดยจะมีการจัดตั้งการสื่อสารระหว่าง 2 ฝ่าย เรียกว่า Application Entities หรือ AE ซึ่งเทียบได้กับบุคคล 2 คนที่ต้องการสนทนากันทางโทรศัพท์ โดย Session Layer จะมีหน้าที่จัดการให้การสนทนาเป็นไปอย่างราบรื่น โดยการเฝ้าตรวจสอบการไหล ของข้อมูลอย่างเป็นจังหวะ ดูแลเรื่องความปลอดภัยเช่น ตรวจสอบอายุการใช้งานของรหัสผ่าน จำกัดช่วงระยะเวลาในการติดต่อ ควบคุมการถ่ายเทข้อมูลรวมถึงการกู้ข้อมูลที่เสียหายอันเกิดมาจาก เครือข่ายทำงานผิดปกติ นอกจากนี้ยังสามารถตรวจดูการใช้งานของระบบและจัดทำบัญชีรายงาน ช่วงเวลาการใช้งานของผู้ใช้ได้

เลเยอร์ 6: Presentation Layer

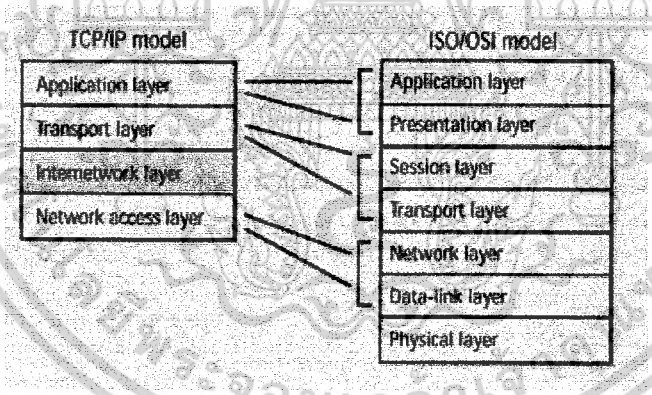
หน้าที่หลักคือการแปลงรหัสข้อมูลที่ส่งระหว่างเครื่องคอมพิวเตอร์ 2 เครื่อง ให้เป็นอักขระแบบเดียวกัน เครื่องคอมพิเตอร์ส่วนใหญ่จะใช้รหัส ASCII (American Standard Code for Information Interchange) แต่ในบางกรณีเครื่องที่ใช้รหัส ASCII อาจจะต้องสื่อสารกับเครื่องเมนเฟรมของ IBM ที่ใช้รหัส EBCDIC (Extended Binary Coded Decimal Interchange Code) ดังนั้น Presentation Layer จะทำหน้าที่แปลงรหัสเหล่านี้ให้เครื่องคอมพิวเตอร์เข้าใจได้ตรงกัน นอกจากนี้ยังสามารถทำการลดขนาดของข้อมูล (data compression) เพื่อเป็นการประหยัดเวลาในการรับส่ง และสามารถเข้ารหัสเพื่อเป็นการป้องกันการโจรกรรมข้อมูลได้อีกด้วย

เลเยอร์ 7: Application Layer

เป็นเลเยอร์บนสุดที่ทำงานใกล้ชิดกับผู้ใช้ การทำงานของเลเยอร์นี้จะเกี่ยวข้องกับโพรโตคอล ต่างๆ มากมาย ซึ่งจะมีการใช้งานที่เฉพาะตัวแตกต่างกันออกไป มีบริการทางด้านโปรแกรม ประยุกต์ต่างๆ ได้แก่ Email, file transfer, remote job entry, directory services นอกจากนี้ยังมีการ จัดเตรียมฟังก์ชั่นในการเข้าถึงไฟล์และเครื่องพิมพ์ ซึ่งเป็นการแบ่งปันการใช้ทรัพยากรณ์บนระบบ เครื่อข่าย

2.7 แบบจำลอง TCP/IP

TCP/IP Model มีแนวคิดพื้นฐานแตกต่างจาก OSI Model คือไม่ได้มีพื้นฐานของการสื่อสาร แบบการสนทนา TCP/IP Model เป็นภาพแสดงถึงโลกของระบบเครื่อข่ายสากล (Internetworking) ที่ทำการเคลื่อนย้ายและกำหนดเส้นทางให้กับข้อมูลระหว่างเครือข่ายและระหวางเครื่อง คอมพิวเตอร์ต่างๆ เมื่อเปรียบเทียบความสัมพันธ์ระหว่างทั้ง 2 โมเดล จะพบว่ามีบางเลเยอร์ที่มีการ กำหนดคุณสมบัติที่เทียบได้ใกล้เคียงกัน แต่บางเลเยอร์ก็ไม่สามารถเทียบหาความสัมพันธ์กันได้เลย

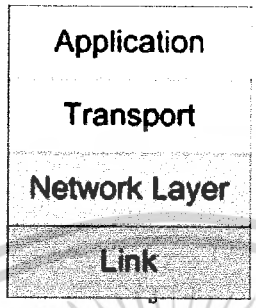


รูปที่ 2.3 เลเยอร์ต่างๆ ใน TCP/IP และ ISO/OSI Model

TCP/IP (Transmitsion Control Protocol/ InternetProtocol) เป็นชุดของโพรโตคอลที่ถูกใช้ใน การสื่อสารผ่านเครือข่ายอินเทอร์เน็ต ได้รับการพัฒนามาตั้งแต่ปี 1960 ซึ่งถูกใช้เป็นครั้งแรกในเครือ ข่าย ARPANET ซึ่งต่อมาได้ขยายการเชื่อมต่อไปทั่วโลกเป็นเครือข่ายอินเตอร์เน็ต ทำให้ TCP/IP เป็นที่ยอมรับอย่างกว้างขวางจนถึงปัจจุบัน

2.7.1 การแบ่งชั้นของ TCP/IP

TCP/IP แบ่งออกเป็น 4 เลเยอร์ ดังรูปที่ 2.4



รูปที่ 2.4 การแบ่งชั้นของ TCP/IP

ในแต่ละเลเยอร์จะมีหน้าที่ดังนี้

• **Link Layer** - เลเยอร์นี้มีหน้าที่ควบคุมการรับส่งข้อมูลในระดับฮาร์ดแวร์ของเครือข่ายรับผิดชอบการรับส่งข้อมูลในระดับกายภาพ จนถึงการแปลความจะกลับสัญญาณไฟฟ้าเป็นข้อมูลทางคอมพิวเตอร์

• **Network Layer** - ทำหน้าที่รับข้อมูลจากชั้น Transport Layer และค้นหาและเลือกเส้นทาง ระหว่างผู้รับและผู้ส่ง เทียบได้กับ Network Layer ของ OSI Model โพรโตคอลในเลเยอร์นี้ได้แก่ IP,ICMP,IGMP

• **Transport Layer** - รับผิดชอบการรับส่งข้อมูลระหว่างปลายด้านส่งและด้านรับข้อมูลและส่ง
ข้อมูลขึ้นไปให้ Application Layer นำไปใช้งาน ต่อ เทียบได้กับ Session Layer และ Transport Layer ของ OSI Model

• **Application Layer** - เป็นเลเยอร์ที่แอพพลิเคชั่นเรียกโพรโตคอลระดับล่างๆลงไป เพื่อให้บริการ ต่างๆ เช่น FTP , SMTP , Telnet , HTTP , POP

2.7.2 โครงสร้างของโพรโตคอล TCP/IP

เนื่องจาก TCP/IP เป็นชุดของโปรโตคอลประกอบด้วยโพรโตคอลหลายตัวทำงานร่วมกันในเล เยอร์ต่างๆ และมีหน้าที่แตกต่างกันออกไป ได้แก่

• **TCP : (Tranmission Control Protocol)** - อยู่ใน Transport Layer ทำหน้าที่จัดการและควบคุม การรับส่งข้อมูล และมีกลไกความคุมการ รับส่งข้อมูลให้มีความถูกต้อง (reliable) และมีการ สื่อ

สารอย่างเป็นกระบวนการ (connection-orient)

• **UDP : (User Datagram Protocol)** - อยู่ใน Transport Layer ทำหน้าที่จัดการและควบคุมการรับ ส่งข้อมูลแต่ไม่มีกลไกความคุม การรับส่ง ข้อมูลให้มีเสถียรภาพและเชื่อถือได้ (unreliable,connectionless) โดยปล่อยให้เป็นหน้าที่ของแอพพลิเคชันเลเยอร์ แต่ UDP มีข้อ ได้เปรียบในการส่งข้อมูลได้ทั้งแบบ unicast , multicast และ broadcast อีกทั้งยังทำการ ติดต่อสื่อสารได้เร็วกว่า TCP เนื่องจาก TCP ต้องเสีย overhead ให้กับขั้นตอนการสื่อสารที่ทำให้ TCP มีความน่าเชื่อถือในการรับส่งข้อมูลนั่นเอง

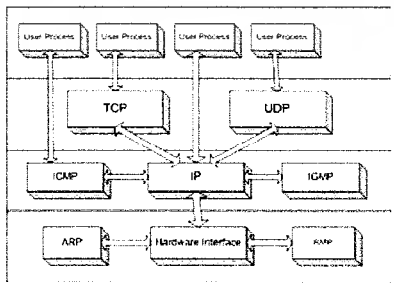
• **IP : (Internet Protocol)** - อยู่ใน Internetwork Layer เป็นโพรตคอลหลักในการสื่อสารข้อมูล มี หน้าที่ค้นหาเส้นทางระว่างผู้รับและผู้ส่ง โดยใช้ IP Address ซึ่งมีลักษณะเป็นเลขสี่ชุด แต่ละชุดมีค่า ตั้งแต่ 0-255 เช่น 172.17.3.12 ในการอ้างอิงโฮสต์ต่างๆ และกลไกการ Route เพื่อส่งต่อข้อมูลไป จนถึงจุดหมายปลายทาง

• **ICMP : (Internet Control Message Protocol)** - อยู่ใน Internetwork Layer มีหน้าที่ส่งข่าวสาร และแจ้งข้อผิดพลาดให้แก่ IP

• **IGMP : (Internet Group Management Protocol)** อยู่ในเน็ตเวิร์กเลเยอร์ ทำหน้าที่ในการส่ง UDP ดาต้าแกรมไปยัง กลุ่มของโฮสต์ หรือ โฮสต์หลายๆตัวพร้อมกัน

• **ARP : (Address Resolution Protocol)** - อยู่ใน Link Layer ทำหน้าที่เปลี่ยนระหว่าง IP แอดเดรส ให้เป็นแอดเดรสของ Network Interface เรียกว่า MAC Address ในการติดต่อระหว่างกัน MAC Address คือหมายเลขประจำของ Hardware Interface ซึ่งในโลกนี้จะ ไม่มี MAC Address ที่ซ้ำ กัน มีลักษณะเป็นเลขฐาน 16 ยาว 6 ไบต์ เช่น 23:43:45:AF:3D:78 โดย 3 ไบต์แรกจะเป็นรหัสของ ผู้ผลิต และ 3 ไบต์หลังจะเป็นรหัสของผลิตภัณฑ์

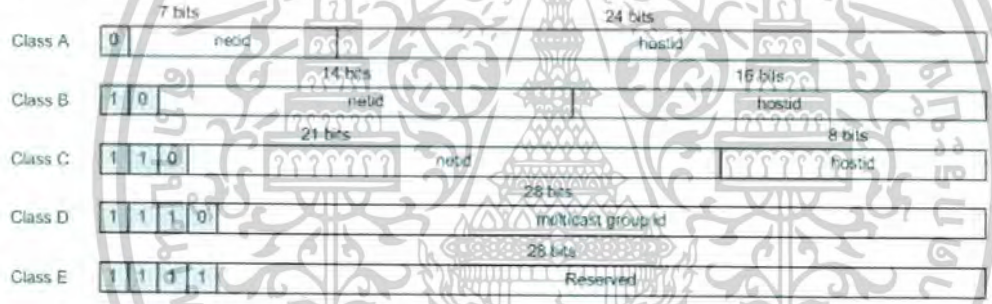
• **RARP : (Reverse ARP)** - อยู่ในลิงค์เลเยอร์เช่นกัน แต่ทำหน้าที่กลับกันกับ ARP คือเปลี่ยน ระหว่างแอดเดรสของ Network Interface ให้เป็นแอดเดรสที่ใช้โดย IP Address



รูปที่ 2.5 แสดงให้เห็นถึงความสัมพันธ์ระหว่างโพรโตคอลต่างๆใน TCP/IP

2.8 IP Addressing

ทุกอินเตอร์เฟซที่ต่ออยู่บนอินเตอร์เน็ตจะต้องมีหมายเลขประจำตัวเพื่อใช้ในการสื่อสารข้อมูล เรียกว่า Internet Address หรือเรียกย่อๆว่า IP Address โดยค่า IP Address นี้จะเป็นหมายเลขจำนวน 32 บิต แต่แทนที่จะกำหนดให้เลขทั้ง 32 บิตนั้นถูกนับต่อเนื่องกันไป ก็จะใช้วิธีการแบ่งหมายเลข ดังกล่าวออกเป็นกลุ่มของเลขขนาด 8 บิตจำนวน 4 ชุด และคั่นแต่ละชุดด้วยจุด ตัวอย่าง 172.17.3.12 นอกจากนี้ใน IP Address นั้นยังถูกแบ่งออกเป็น 2 ส่วนคือ ส่วนที่ เป็นแอดเดรส ของเน็ตเวิร์ก (Network ID) และส่วนที่เป็นแอดเดรสของโฮสต์ (Host ID) ซึ่งข้อมูลในส่วนนี้จะ ถูกใช้สำหรับ ค้นหาเส้นทางของ IP ในการที่จะขนส่งข้อมูลจากต้นทางให้ถึงปลายทางอย่างถูกต้อง เพื่อเป็นการกำหนดขนาดของเน็ตเวิร์ก สำหรับ IP Address ต่างๆดังนั้นจึงมีการจัด IP Address ในแต่ละช่วงออกเป็นคลาส (class) ต่างๆกันจาก A ถึง E เพื่อจะได้ทำการจัดสรร IP Address ได้อย่างเหมาะสมกับขนาดของเน็ตเวิร์ก



รูปที่ 2.6 การกำหนด IP Address ในคลาสต่างๆ

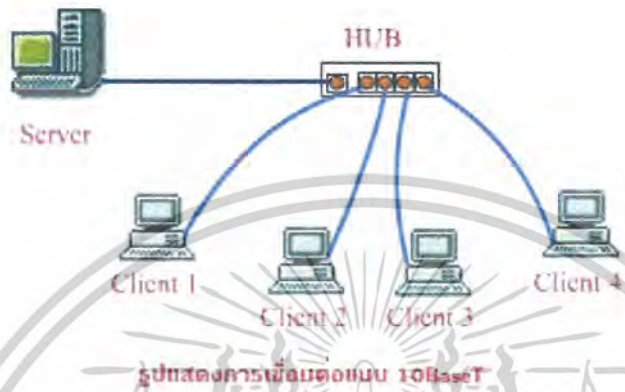
จากข้อกำหนดในการแบ่งคลาสของ IP Address หากลองนำบิตที่อยู่ในตอนต้นของ ip address ในแต่ละคลาสมาแปลงเป็น ip address ในเลขฐานสิบ จะเห็นว่าแต่ละคลาสครอบคลุม ip address ช่วงต่างๆ ดังตารางที่ 2.1

Class	IP Range	
A	0.0.0.0	- 127.255.255.255
B	128.0.0.0	- 191.255.255.255
C	192.0.0.0	- 223.255.255.255
D	224.0.0.0	- 239.255.255.255
E	240.0.0.0	- 255.255.255.255

ตารางที่ 2.1 แสดงช่วงของ IP Adress ในแต่ละคลาส

2.9 อุปกรณ์เครือข่าย

2.9.1 เซอร์เวอร์ (Server)



รูปที่ 2.7 เซอร์เวอร์

เซอร์เวอร์ หรือเรียกอีกอย่างหนึ่งว่า เครื่องแม่ข่าย เป็นเครื่องคอมพิวเตอร์หลักในเครือขาย ที่ ทำหน้าที่จัดเก็บและให้บริการไฟล์ข้อมูลและทรัพยากรอื่นๆ กับคอมพิวเตอร์เครื่องอื่น ๆ ใน เครือข่าย โดยปกติคอมพิวเตอร์ที่นำมาใช้เป็นเซอร์เวอรมักจะเป็นเครื่องที่มีสมรรถนะสูง และมีฮาร์ดดิสก์ความจำสูงกว่าคอมพิวเตอร์เครื่องอื่น ๆ ในเครือข่าย

ไคลเอนต์ (Client) หรือเรียกอีกอย่างหนึ่งว่า เครื่องลูกข่าย เป็นคอมพิวเตอร์ในเครือข่ายที่ร้อง ขอ บริการและเข้าถึงไฟล์ข้อมูลที่จัดเก็บในเซอร์เวอร์ หรือพูดง่าย ๆ ก็คือ ไคลเอนต์ เป็นคอมพิวเตอร์ ของผู้ใช้แต่ละคนในระบบเครือข่ายนั่นเอง

2.9.2 ฮับ (HUB)

รูปที่ 2.8 ฮับ

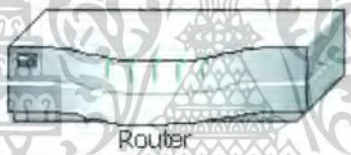
ฮับ (HUB) หรือเรียก รีพีทเตอร์ (Repeater) คืออุปกรณ์ที่ใช้เชื่อมต่อกลุ่มคอมพิวเตอร์ ฮับมี หน้าที่รับส่งเฟรมข้อมูลทุกเฟรมที่ได้รับจากพอร์ตใดพอร์ตหนึ่ง ไปยังพอร์ตที่เหลือ คอมพิวเตอร์ที่ เชื่อมต่อเข้ากับฮับจะแชร์แบนด์วิธหรืออัตราข้อมูลของเครือข่าย เพราะฉะนั้นถ้ามีคอมพิวเตอร์ เชื่อมต่อมากจะทำให้อัตราการส่งข้อมูลลดลง

2.9.3 สวิตซ์ (Switch)

รูปที่ 2.9 สวิตซ์

คืออุปกรณ์เครือข่ายที่ทำหน้าที่ในเลเยอร์ที่ 2 และทำหน้าที่ส่งข้อมูลที่ได้รับมาจากพอร์ตหนึ่งไปยังพอร์ตเฉพาะที่เป็นปลายทางเท่านั้น และทำให้คอมพิวเตอร์ที่เชื่อมต่อกับพอร์ตที่เหลือส่ง ข้อมูลถึงกันในเวลาเดียวกัน ดังนั้น อัตราการรับส่งข้อมูลหรือแบนด์วิธจึงไม่ขึ้นอยู่กับคอมพิวเตอร์ ปัจจุบันนิยมเชื่อมต่อแบบนี้มากกว่าฮับเพราะลดปัญหาการชนกันของข้อมูล

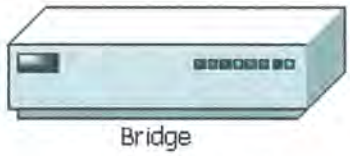
2.9.4 เราเตอร์ (Router)



รูปที่ 2.10 เราเตอร์

เป็นอุปกรณ์ที่ทำหน้าที่ในเลเยอร์ที่ 3 เราท์เตอร์จะอ่านที่อยู่ (Address) ของสถานีปลายทาง ที่ส่วนหัว (Header) ข้อแพ็กเก็ตข้อมูล เพื่อที่จะกำหนดและส่งแพ็กเก็ตต่อไป เราท์เตอร์จะมีตัวจัด เส้นทางในแพ็กเก็ต เรียกว่า เราติ้งเทเบิ้ล (Routing Table) หรือตารางจัดเส้นทาง นอกจากนี้ยังส่ง ข้อมูลไปยังเครือข่ายที่ให้โพรโตคอลต่างกันได้ เช่น IP (Internet Protocol) , IPX (Internet Package Exchange) และ AppleTalk นอกจากนี้ยังเชื่อมต่อกับเครือข่ายอื่นได้ เช่น เครือข่ายอินเตอร์เน็ต

2.9.5 บริดจ์ Bridge



รูปที่ 2.11 บริดจ์

บริดจ์ เป็นอุปกรณ์ที่มักจะใช้ในการเชื่อมต่อวงแลน (LAN Segments) เข้าด้วยกัน ทำให้สามารถขยายขอบเขตของ LAN ออกไปได้เรื่อยๆ โดยที่ประสิทธิภาพรวมของระบบ ไม่ลดลงมาก

นัก เนื่องจากการติดต่อของเครื่องที่อยู่ในเซกเมนต์เดียวกันจะไม่ถูกส่งผ่าน ไปรบกวนการจราจรของเซกเมนต์อื่น และเนื่องจากบริดจ์เป็นอุปกรณ์ที่ทำงานอยู่ในระดับ Data Link Layer จึงทำให้สามารถใช้ในการเชื่อมต่อเครือขายที่แตกต่างกันในระดับ Physical และ Data Link ได้ เช่น ระหว่าง Ethernet กับ Token Ring เป็นต้น.

บริดจ์ มักจะถูกใช้ในการเชื่อมเครือข่ายย่อย ๆ ในองค์กรเข้าด้วยกันเป็นเครือข่ายใหญ่เพียง เครือข่ายเดียว เพื่อให้เครือข่ายย่อยๆ เหลานั้นสามารถติดต่อกับเครือข่ายย่อยอื่นๆ ได้

2.9.6 เกตเวย์ (Gateway)

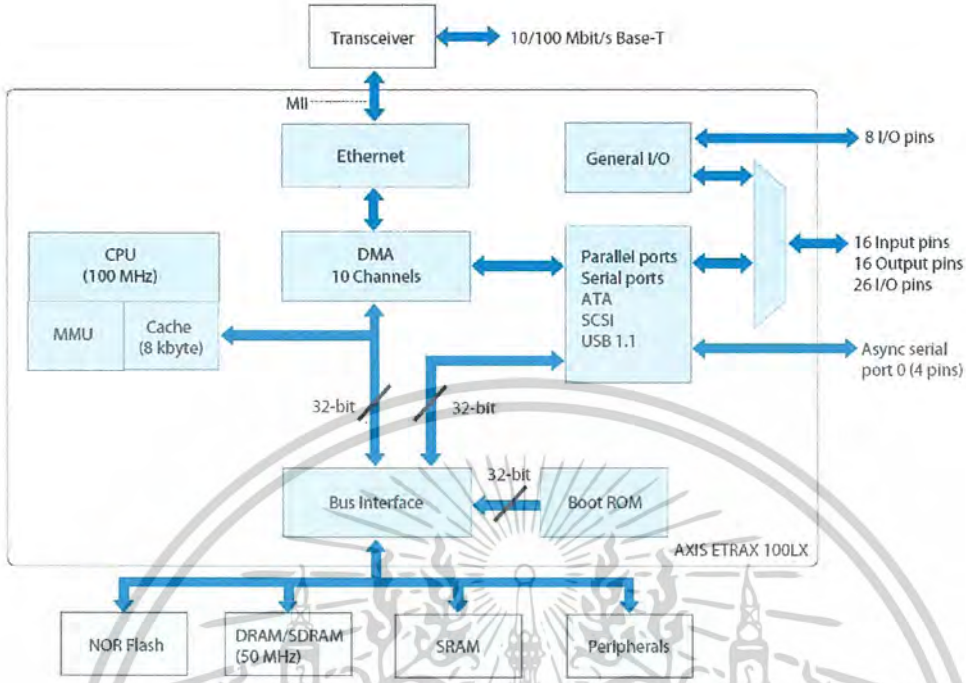
รูปที่ 2.12 เกตเวย์

เกตเวย์ เป็นอุปกรณ์ฮารดแวร์ที่เชื่อมต่อเครือข่ายต่างประเภทเข้าด้วยกัน เช่น การใช้เกตเวย์ใน การเชื่อมต่อเครือข่าย ที่เป็นคอมพิวเตอร์ประเภทพีซี (PC) เข้ากับคอมพิวเตอร์ประเภทแมคอิน ทอช (MAC) เป็นต้น

2.10 คุณสมบัติของไมโครคอนโทรลเลอร์ ETRAX 100LX RISC

2.10.1 คุณสมบัติทางด้าน Hardware

- Microprocessor ETRAX 100LX RISC Architecture 100 MIPS 32 bit
- 8 M Flash
- 32 MB SDRAM
- 1xEthernet 10/100 on board
- 1xUSB port
- 2x20 pin header for 48 DIO
- Low power 300mA @5V



รูปที่ 2.13 ระบบการทำงานของ Microprocessor ETRAX 100LX RISC

2.10.2 คุณสมบัติทางด้าน Software

- Build in Web server
- Real Linux Operating System Kernel 2.4.x or 2.6.x
- Full suit of TCP/IP ipv4, ipv6
- Firmware Upgradeable FTP, SSH and Telnet with Lan interface

2.10.3 วิธี Configuring SDK and Kernel

เข้าไปยัง Home ของ SDK

cd /home/fox/devboard-R2_01

source init_env

การกำหนด Kernel จะมีรายละเอียดและเมนูให้เลือกกำหนด ด้วยคำสั่ง make menuconfig

make menuconfig

```
    ACME FOXBOARD selection (FOXBOARD LX832)  --->
    Linux kernel (Linux 2.4.x)  --->
    Standard C Library (glibc)  --->
    Driver settings  --->
    Network Settings  --->
    Applications  --->
    Advanced Settings  --->
[*] Use pregenerated ssh keys
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
```

เลือกปรับแต่งตามต้องการ จากนั่นบันทึก

จากนั้นทำการ Config **./configure**

การกำหนด Tools ต่าง ๆของ busybox ด้วยคำสั่ง make busybox

make busybox

```
    General Configuration  --->
    Build Options  --->
    Installation Options  --->
    Archival Utilities  --->
    Coreutils  --->
    Console Utilities  --->
    Debian Utilities  --->
    Editors  --->
    Finding Utilities  --->
    Init Utilities  --->
    Login/Password Management Utilities  --->
    Miscellaneous Utilities  --->
    Linux Module Utilities  --->
    Networking Utilities  --->
    Process Utilities  --->
    Another Bourne-like Shell  --->
    System Logging Utilities  --->
    Linux System Utilities  --->
    Debugging Options  --->
---
Load an Alternate Configuration File
Save Configuration to an Alternate File
```

การกำหนดค่าของ Kernel ด้วยคำสั่ง make kernelconfig

```
Code maturity level options  --->
Loadable module support  --->
General setup  --->
Hardware setup  --->
Drivers for ETRAX 100LX built-in interfaces  --->
Memory Technology Devices (MTD)  --->
Parallel port support  --->
Plug and Play configuration  --->
Block devices  --->
Multi-device support (RAID and LVM)  --->
Networking options  --->
Telephony Support  --->
ATA/IDE/MFM/RLL support  --->
SCSI support  --->
I2O device support  --->
Network device support  --->
Amateur Radio support  --->
IrDA (infrared) support  --->
ISDN subsystem  --->
Old CD-ROM drivers (not SCSI, not IDE)  --->
Input core support  --->
Character devices  --->
Multimedia devices  --->
File systems  --->
Sound  --->
USB support  --->
ACME  --->
Kernel hacking  --->
---
```

การสร้าง image ไฟล์

./configure make

เสร็จแล้วจะได้ไฟล์ชื่อว่า fimage

2.10.4 การดาวโหลด Reflash

- ติดตั้ง WinPcap 4.0
- ติดตั้ง flashFox ไฟล์จะอยู่ใน CD-ROM: \firmware นำไฟล์มาไว้ใน เช่น c:\flash
- Copy ไฟล์ image ที่ต้องการ flash ลงที่โฟลเดอร์ c:\flash
- เข้า Command Prompt แล้วพิมพ์ตามนี้

cd\

cd flash

flashfox –i

จะมีข้อความขึ้นดังนี้

****

e100boot made by **axis.se**, ported for windows by **acmesystems.it**

feedback, bugs : **info@acmesystems.it**

****

The following devices are available

1.) Wirless USB Ethernet Adaptor

2.) Realtek RTL8139 Family Fast Ethernet Adapter

- ดูว่า LAN Card ของเครื่องเป็นรุ่นไหน จากตัวอย่างคือ 2
- พิมพ์คำสั่ง flashfox -d 2 -F -i แล้วตามด้วยชื่อไฟล์ที่ต้องการ flash

ตัวอย่าง : flashfox -d 2 -F -i fimage

- ที่ด้านหลัง Board ให้ DIP Switch 2 ให้เป็น ON จากนั้นเสียบ Adaptor

รูปที่ 2.14 การ set DIP switch

- จะมีข้อความขึ้นดังนี้

Device ID = 0x00001b49

This bootloader was built by blogic on Mon Mar 5 10:29:02 CET 2007.

Checksum of bootloader is 0x000a0ac0

Waiting for load info.

Checksum of file is 0x00001ebd

Got load info.

SET_REGISTER

0xb0000000

0x000095f8

SET_REGISTER

0xb0000004

0x00000004

SET_REGISTER

0xb000000c

0x09603737

SET_REGISTER

0xb0000008

...

...

...

0x80360000: Writing 0x00010000 bytes

0x80370000: Writing 0x00010000 bytes

0x80380000: No need to write

0x80390000: No need to write

0x803a0000: Writing 0x00010000 bytes

0x80000000: Verifying...OK

JUMP

0x00000000

END

Exiting with code 0

- จากนั้นเขี่ย Dip Switch 2 คืนตำแหน่งเดิม
- Reset Boardโดยถอด Adapter ออก แล้วเสียบใหม่อีกครั้ง

2.10.5 การใช้งานตัวแปรในโปรแกรม

ตัวแปรจะเป็นตัวที่เก็บค่าต่าง ๆ ไม่ว่าจะเป็นตัวอักษรหรือตัวเลข

การตั้งชื่อตัวแปร

ชื่อตัวแปรต้องขึ้นต้นด้วยตัวอักษรหรือเครื่องหมายขีดล่าง (Underscore , _) ที่เหลือจะเป็นตัวอะไรก็ได้

เช่น book="Unix book" คือการกำหนดให้ตัวแปร book เก็บค่าสตริง Unix book

ค่าตัวแปรที่ใช้ในการเขียน

$0 จะเป็นตัวแปรที่เก็บชื่อโปรแกรม หรือ สคริปต์ที่ทำการเรียกใช้อยู่ปัจจุบัน

$1...$9 เป็นตัวแปรที่เก็บค่าอาร์กิวเมนต์ของโปรแกรม ซึ่งค่าเหล่านี้ถูกกำหนดตามหลังมากับชื่อโปรแกรม

$* ตัวแปรที่เก็บค่าอาร์กิวเมนต์ของโปรแกรมทั้งหมด ซึ่งหมายถึง $1 ... $9

$# เก็บจำนวนของอาร์กิวเมนต์ทั้งหมด

$$ เก็บค่า โพรเซสของโปรแกรมที่เรียกใช้อยู่ปัจจุบัน

$! เก็บค่าตัวเลขโพรเซสของโปรแกรม หรือคำสั่งที่รันอยู่เบื้องหลังล่าสุด

$? เก็บค่าที่ถูกส่งคืนมาจากโปรแกรม หรือคำสั่งที่เพิ่งทำงานเสร็จ (Return Code)

$@ คล้ายกับ $* คือ เก็บค่าอาร์กิวเมนต์ของโปรแกรมทั้งหมด

การเปรียบเทียบ (expression)

คำสั่งเกี่ยวกับจำนวนเต็ม

int1 -eq int2	เป็นจริงเมื่อ int1 เท่ากับ int2
int1 -ge int2	เป็นจริงเมื่อ int1 มากกว่าหรือเท่ากับ int2
int1 -gt int2	เป็นจริงเมื่อ int1 มากกว่า int2
int1 -le int2	เป็นจริงเมื่อ int1 น้อยกว่าหรือเท่ากับ int2
int1 -lt int2	เป็นจริงเมื่อ int1 น้อยกว่า int2
int1 -ne int2	เป็นจริงเมื่อ int1 ไม่เท่ากับint2

คำสั่งเกี่ยวกับจำนวนเต็ม

int1 -eq int2	เป็นจริงเมื่อ int1 เท่ากับ int2
int1 -ge int2	เป็นจริงเมื่อ int1 มากกว่าหรือเท่ากับ int2
int1 -gt int2	เป็นจริงเมื่อ int1 มากกว่า int2
int1 -le int2	เป็นจริงเมื่อ int1 น้อยกว่าหรือเท่ากับ int2
int1 -lt int2	เป็นจริงเมื่อ int1 น้อยกว่า int2
int1 -ne int2	เป็นจริงเมื่อ int1 ไม่เท่ากับint2

คำสั่งเกี่ยวกับ String

str1 = str2	เป็นจริงเมื่อ str1 เหมือนกับ str2
str1 != str2	เป็นจริงเมื่อ str1 ไม่เหมือนกับ str2
str	เป็นจริงเมื่อ str ไม่เป็น null
-n str	เป็นจริงเมื่อ str มีความยาวมากกว่า 0
-z str	เป็นจริงเมื่อ str มีความยาวเป็น 0

คำสั่งเกี่ยวกับ file

-d filename	เป็นจริงเมื่อ filename เป็น directory
-f filename	เป็นจริงเมื่อ filename เป็น file
-r filename	เป็นจริงเมื่อ filename อ่านได้โดยโปรแกรม
-s filename	เป็นจริงเมื่อ filename มีขนาดไม่เป็น 0
-w filename	เป็นจริงเมื่อ filename เขียนได้โดยโปรแกรม
-x filename	เป็นจริงเมื่อ filename run ได้โดยโปรแกรม

คำสั่งเกี่ยวกับ Logical อื่นๆ

! expr	เป็นจริงเมื่อ exp เป็นเท็จ
exp1 && exp2	เป็นจริงเมื่อ exp1 และ exp2 เป็นจริง
exp1 || exp2	เป็นจริงเมื่อ exp1 หรือ exp2 เป็นจริง

รูปแบบเงื่อนไข

โครงสร้าง

```
if [expression ]
then   commands
elif [ exprression ]
then   commands
else
commands
fi
```

เงื่อนไข case หรือ switch

```
case string  in
   str1)
      commands;;
```

```
    str2)
        commands;;
    str3)
        commands;;
    *)
        commands;;
  esac
```

เงื่อนไขการวนลูปด้วย for

โครงสร้าง

```
  for name  [ in list ]
  do
  commands
  done
```

เงื่อนไขการวนลูปด้วย while

โครงสร้าง

```
  while [ condition ]
  do
    code block;
  done
```

การเขียน Function

คือชุดคำสั่งหลาย ๆ คำสั่งเหมือนกับ shell script แต่ฟังกชั่นจะเหมือนเป็น shell script ที่อยู่ใน shell script อีกที่หนึ่ง

ฟังกชั่นจะถูกใช้ใน กรณีที่จะต้องทำการใช้คำสั่งใด ๆ ซ้ำอยู่บ่อย ๆ แทนที่จะต้องเขียนคำสั่งนั้นบ่อย ๆ ก็ให้นำคำสั่งนั้นมาสร้างเป็น

ฟังก์ชั่น แล้วเรียกฟังก์ชั่นนั้นอีกที่

โครงสร้าง

```
  [function] fname {
      (shell commands)
  }
```

2.11 คำสั่งยูนิกส์ Command Line

ls เป็นคำสั่งที่ใช้สำหรับแสดงแฟ้มข้อมูล (เช่นเดียวกับ dir ของDOS) มากจากคำว่า list

รูปแบบคำสั่ง ls [option] [file]

option ที่มักใช้กันใน ls คือ

-l จะแสดงผลลัพธ์แบบ Long Format ซึ่งจะแสดง Permission ของแฟ้มด้วย

-a จะแสดงแฟ้มข้อมูลทั้งหมด

-F จะแสดง / หลัง Directory และ * หลังแฟ้มข้อมูลที่ execute ได้

adduser คำสั่งเพิ่ม User ให้กับระบบ Unix,Linux

รูปแบบการใช้งาน adduser -g (group) -d (Directory) (User)

useradd คำสั่งเพิ่ม User ให้กับระบบ Unix,Linux (ใช้เหมือนกับคำสั่ง adduser)

รูปแบบการใช้งาน useradd -g (group) -d (Directory) (User)

userdel คำสั่งลบ User ออกจากระบบ Unix,Linux

รูปแบบการใช้งาน userdel [option] (Username)

passwd คำสั่งกำหนดและแก้ไขรหัสผ่านของ User ของระบบ Unix,Linux

รูปแบบการใช้งาน passwd [Username]

alias คำสั่งกำหนดคำสั่งย่อของระบบ Unix,Linux (คล้ายกับคำสั่ง SETในDOSแต่สามารถใช้เป็นคำสั่ง RUNได้)

รูปแบบการใช้งาน alias [ชื่อใหม่=ข้อความ]

cp เป็นคำสั่งที่ใช้สำหรับสำเนาแฟ้มข้อมูล (เช่นเดียวกับcopyของDOS) มาจากคำว่า copy

รูปแบบคำสั่ง cp source target

cal คำสั่งแสดงปฏิทินของระบบ Unix,Linux

รูปแบบการใช้งาน cal [Enter](สั่งให้ระบบแสดง ปฏิทินเดือน ปัจจุบัน)

cat คำสั่งแสดงข้อความในFileของระบบ Unix,Linux (คล้ายกับคำสั่งTypeของDOS)

cd คำสั่งChange Directoryของระบบ Unix,Linux (คล้ายกับคำสั่งCDของDOS)

รูปแบบการใช้งาน cd [directory]

chgrp คำสั่งChange Groupของระบบ Unix,Linux (เป็นการเปลี่ยนกลุ่มเจ้าของไฟล์)

รูปแบบการใช้งาน chgrp [-chfRv] (Group) (File)

chmod คำสั่งChange Modeของระบบ UNIX, Linux (เป็นการเปลี่ยนสิทธิการเข้าถึงไฟล์)

รูปแบบการใช้งาน chmod [สิทธิ] (File)

การกำหนดสิทธิกำหนดได้ 2 ลักษณะคือ

1.กำหนดโดยใช้อักษรย่อกลุ่ม

2.ใช้รหัสเลขฐาน2แทนสิทธิ (1 คืออนุญาต)

กลุ่มผู้ใช้ User Group Other = ugo เช่น go-r-w+x คือกลุ่ม และคนอื่นไม่มีสิทธิอ่านเขียนแต่Runได้

สิทธิ์การใช้ -rwx rwx rwx = Read Write Execute

รหัสเลขฐาน 111 101 100 = 754 คือเจ้าของไฟล์ใช้ได้ครบ คน Group เดียวกันอ่าน Executeได้นอกนั้นอ่านได้อย่างเดียว

chown คำสั่งChange Ownerของระบบ Unix, Linux (เป็นการเปลี่ยนเจ้าของไฟล์)

รูปแบบการใช้งาน chown [ชื่อเจ้าของไฟล์] (ชื่อFile)

clear คำสั่ง clear ของระบบ Unix, Linux

รูปแบบการใช้งาน clear

date ใช้แสดง วันที่ และ เวลา

df คำสั่ง df ของระบบ Unix, Linux (เป็นการตรวจสอบการใช้พื้นที่บนฮาร์ดดิสก์)

รูปแบบการใช้งาน df [option] [file]

du คำสั่ง du : แสดงการเนื้อที่ใช้งาน ของแต่ละ directory โดยละเอียด

du :: เพื่อแสดงรายชื่อ directory และเนื้อที่ที่ใช้ไป

du -all :: เพื่อแสดงโดยละเอียดว่าแต่ละแฟ้มมีขนาดเท่าใด ใน directory ปัจจุบัน

du | sort -g :: แสดงการใช้พื้นที่ของแต่ละ directory พร้อม sort จากน้อยไปมาก มีหน่วยเป็น Kb

du -b :: แสดงหน่วยเป็น byte ของแต่ละ directory

ps คำสั่ง ps : แสดง Process หรือโปรแกรมที่ประมวลผลอยู่ในระบบขณะนั้น

ps :: แสดงชื่อ process ต่าง ๆ ที่ทำงานอยู่อย่างสั้น

ps -ef :: แสดงข้อมูลของ process โดยละเอียด

ps -ax :: แสดงข้อมูลของ process พร้อมชื่อโปรแกรมได้ละเอียด

ps -aux :: แสดงข้อมูลของ process พร้อมชื่อโปรแกรมและชื่อผู้สั่งได้ละเอียด

top คำสั่ง top : ใช้แสดงสถานการณ์ใช้ทรัพยากร ภายในเครื่อง

find คำสั่ง find : เมื่อไฟล์ที่ต้องการว่าอยู่ใน directory ของเครื่องเราหรือไม่

find / -name hello.pl :: ใช้ค้นหาแฟ้ม hello.pl ในทุก directory

find / -name hello* :: ใช้ค้นหาแฟ้มที่ขึ้นต้นด้วยคำว่า hello

find /bin -size 626188c :: ใช้ค้นหาแฟ้มที่มีขนาด 626188 ถ้าเป็น RH8 จะพบแฟ้ม bash

dmesg คำสั่ง dmesg ของระบบ Unix, Linux (เป็นการให้แสดงผลเหมือนตอน Boot)

รูปแบบการใช้งาน dmesg

หมายเหตุ คำสั่งนี้ ใช้ตรวจสอบ เมื่อเกิดปัญหา

cho คำสั่ง echo ของระบบ Unix, Linux เป็นการให้แสดงข้อความ

รูปแบบการใช้งาน echo (ข้อความที่ต้องการให้แสดงผล)

exit คำสั่ง exit ของระบบ Unix, Linux (ออกจากระบบยูนิกส์)

รูปแบบการใช้งาน exit

grep คำสั่ง grep ของระบบ Unix, Linux (เป็นการสั่งให้ค้นหาตามเงื่อนไข)

รูปแบบการใช้งาน grep (option)

ไฟล์ /etc/test

gzip/gunzip คำสั่ง gzip/gunzip ของระบบ Unix, Linux (เป็นการบีบอัดไฟล์หรือขยายบีบอัดไฟล์)

รูปแบบการใช้งาน gzip หรือ gunzip (-cdfhlLnNrtv19) [file]

ifconfig คำสั่ง history ของระบบ Unix, Linux เป็นการตรวจสอบกำหนดค่า Network

รูปแบบการใช้งาน ifconfig [option]

netstat คำสั่ง netstat : แสดงสถานะของเครือข่ายว่ามีโปรแกรมใดเปิดให้บริการ

route คำสั่ง route : ใช้เส้นทางการเชื่อมเครือข่าย

env คำสั่ง env : แสดงค่า environment ปัจจุบัน

kill คำสั่ง kill ของระบบ Unix, Linux (เป็นคำสั่งสำหรับยกเลิก Process)

รูปแบบการใช้งาน kill [option] (process ID)

Tail คำสั่ง tail : แสดงส่วนท้ายของแฟ้มที่มีขนาดใหญ่ ต้องข้ามกับ cat ที่ดูตั้งแต่เริ่มแฟ้ม

mkdir เป็นคำสั่งที่ใช้สำหรับการสร้าง directory มาจากคำว่า make directory

รูปแบบของคำสั่ง mkdir [option] [file]

โดย option ที่มักใช้กันใน mkdir คือ

-m จะทำการกำหนด Permissioin

-p จะทำการสร้าง Parent Directory ให้ด้วยกรณีที่ยังไม่มีการระบุ directory ในที่นี้อาจเป็น relative

หรือ absolute path ก็ได้

เป็นคำสั่งที่ใช้สำหรับการย้ายแฟ้มข้อมูลและ Directory รวมถึงการเปลี่ยนชื่อด้วย (ในทำนองเดียวกับ dos)

มาจากคำว่า move

รูปแบบคำสั่ง mv source target

mount คำสั่ง mount ของระบบ Unix, Linux (เป็นคำสั่งเชื่อมต่ออุปกรณ์เข้ากับระบบ)

รูปแบบการใช้งาน mount (-t type) DeviceDriver MountPoint

การ Mount cdrom แบบที่ 2 mount /dev/cdrom เมื่อmountแล้ว CD จะอยู่ที่ /mnt/cdrom

Rmdir เป็นคำสั่งที่ใช้สำหรับการลบ Directory มาจากคำว่า remove directory

โครงสร้างคำสั่ง rmdir [option] [file]

โดย option ที่มักใช้กันใน mkdir คือ -p จะทำการลบ Child และ Parent Directory ตามลำดับ directory ในที่นี้อาจเป็น relative หรือ absolute path ก็ได้

tar เป็นคำสั่งเพื่อการ backup และ restore file ทั้งนี้การ tar จะเก็บทั้งโครง สร้าง directory และ file permission ด้วย

(เหมาะสำหรับการเคลื่อนย้าย หรือแจกจ่ายโปรแกรมบนระบบ UNIX) มาจากคำว่า tape archive

รูปแบบคำสั่ง tar [option]... [file]... โดย option ที่มักใช้กันใน echo คือ

-c ทำการสร้างใหม่ (backup)

-t แสดงรายชื่อแฟ้มข้อมูลในแฟ้มที่ backup ไว้

-v ตรวจสอบความถูกต้องของการประมวลผล

-f ผลลัพธ์ของมาที่ file

-x ทำการ restore

whoami or who

คำสั่งใช้เพื่อแสดงว่าผู้ใช้ซึ่ง login เข้าสู่ระบบนั้น (ตัวเราเอง) login ด้วยชื่ออะไร

รูปแบบคำสั่ง/ตัวอย่าง whoami หรือ who am i (บน SUN OS หรือ UNIX บางตัวเท่านั้น)

free แสดงหน่วยความจำที่เหลืออยู่บนระบบ

โครงสร้างคำสั่ง free [-b|-k|-m]

โดย option ที่มักใช้กันใน free คือ

-b แสดงผลลัพธ์เป็นหน่วย byte

-k แสดงผลลัพธ์เป็นหน่วย kilobyte

-m แสดงผลลัพธ์เป็นหน่วย megabyte

ตัวอย่าง free free –b free -k

pwd เป็นคำสั่งที่ใช้สำหรับแสดง Directory ปัจจุบัน (ในทำนองเดียวกับการพิมพ์ cd บน DOS) มาจากคำว่า print work directory

uname คำสั่งแสดง ชื่อและรุ่นของ OS ชื่อและรุ่นของ cpu ชื่อเครื่อง

hostname คำสั่งแสดงชื่อเครื่องที่ใช้อยู่

tty แสดงหมายเลข terminal ที่ใช้งานอยู่

id ใช้แสดงชื่อและกลุ่มของผู้ใช้งาน

2.12 Operating System Component

2.12.1 Kernel คือหัวใจของระบบจะควบคุมการทำงานภายในทั้งหมดของระบบคอมพิวเตอร์ เช่น การเตรียมทรัพยากรต่างๆของระบบ การจัดเก็บข้อมูล การบริหารหน่วยความจำ การควบคุมอุปกรณ์ต่างๆที่ต่ออยู่ ตัว kernel จะขึ้นกับ ชนิดของเครื่องดังนั้นเราต้องใช้ kernel คนละตัวกันหากใช้เครื่องคนละตระกูลกัน

2.12.2 File System (FS) คือโครงสร้างการจัดเก็บข้อมูลในฮาร์ดดิสก์ เพื่อให้ OS สามารถอ่านเขียน ใช้ไฟล์ที่ต้องการได้อย่างมีประสิทธิภาพ โดยที่ OS แต่ละตัวจะมี FS ที่แตกต่างกัน เช่น

Operating System File System

DOS/Windows95 FAT12,FAT16

Windows98/95-osr FAT12,FAT16,FAT32

Windows NT NTFS,FAT16,HPFS

OS/2 FAT12,FAT16,HPFS

Linux EXT2,VFAT,HPFS,NTFS,etc.

SunOS UFS

หมายเหตุ เนื่องจาก Linux ใช้ File System แบบ Ext2 (Extended Files System 2) จึงทำให้ Linux สามารถมองเห็นดิสก์ก้อนที่ใหญ่มากมีขนาดถึง 4 เทราไบต์ (Tbytes) หรือขนาด 4000 Gbytes

2.12.3 Shell เป็น command Interpreter เป็นตัวกลางติดต่อระหว่าง user กับ kernel คอยรับคำสั่งที่จะพิมพ์เข้าไปแล้วแปลคำสั่งนั้นต่อไป นอกจากนี้แล้วยังสามารถที่จะนำเอาคำสั่งต่างๆ มาเขียนเป็นโปรแกรมเรียกว่า Shell Script และ shell ยังสามารถกำหนดทิศทาง Input / Output ได้ด้วย การเปลี่ยนทิศทางจะมีเครื่องหมายที่จำเป็นคือ

> การเปลี่ยนทิศทางของ output

< การเปลี่ยนทิศทางของ input

>> การเปลี่ยนทิศทางของ output ไปต่อท้ายไฟล์

การทำงานผ่าน shell มี 2 ลักษณะคือ

Synchronous execution เป็นการทำงานตามลำดับของคำสั่งทีละคำสั่งจนเสร็จแล้วจึงจะขึ้น prompt เพื่อป้อนคำสั่งต่อไป เรียกว่าการทำงานแบบฉากหน้า (foreground mode) เช่น

$ ls -l (เป็นการ list ดูไฟล์แบบยาวใน directory ปัจจุบัน)

Asynchronous execution จะทำงานตามคำสั่งโดยที่งานเก่าจะเสร็จแล้วหรือยังไม่เสร็จก็ตามแต่ shell จะกำหนด prompt และสร้าง shell ใหม่ขึ้นมาเพื่อรองรับงานใหม่ต่อไป เรียกว่าการทำงานแบบฉากหลัง (background mode) การทำงานแบบนี้ทำได้โดยเติมเครื่องหมาย ampersand (&) ไว้ที่ท้ายคำสั่งนั้นเช่น

$ netscape & (เรียกโปรแกรม netscape แล้วขึ้น prompt โดยไม่ต้องรอให้ออกจาก netscape ก่อน)

Shell ที่นิยมใช้

Bourne Shell (sh) เป็น starndard shell ที่มีใน unix ทุกตัวสามารถย้าย shell script ไปยัง Unix ระบบอื่นได้โครงสร้างเป็นแบบ Algol สามารถใช้งาน Procedure ได้ จะมี default prompt เป็น "$"

C Shell (csh) มีโครงสร้างคล้ายภาษา C ทำงานได้ดีกว่า bourne shell มีไฟล์ที่เก็บคำสั่งที่ใช้ไปแล้ว ทำงานกับ shell script ของ bourne shell ไม่ได้ default prompt เป็น "%"

Korn Shell (ksh) ทำงานได้ดีกว่า sh และ csh แต่ไม่ได้มีใน Unix ทุกตัว ksh มีขนาดใหญ่กว่า shell อื่น ๆ เขียน shell script ได้ง่ายขึ้นและรัดกุม เป็น Standard IEEE PDSIX 1003.2 default prompt เป็น "$"

Bourne Again Shell (bash) เป็นการพัฒนา sh ให้สามารถมีแฟ้มคำสั่งที่ใช้ไปแล้ว และเพิ่มขีดความสามารถเพิ่มขึ้นอีกหลายอย่าง (default of Linux) default prompt เป็น "$" ฯลฯ

2.12.4 Utilities คำสั่งต่างที่ทำงานได้บน ระบบงาน UNIX จึงทำให้ kernel มีขนาดเล็กเพราะจะมีเฉพาะหน้าที่สำคัญเท่านั้น

ประเภทของไฟล์ใน UNIX

ไฟล์ในระบบยูนิกซ์นั้นจะขึ้นอยู่กับผู้สร้างยูนิกซ์แต่ละตัวซึ่งมีทั้งแตกต่างและเหมือนกัน และการตั้งชื่อไฟล์ในระบบยูนิกซ์ส่วนใหญ่จะสามารถตั้งชื่อได้ยาวถึง 255 ตัวอักษรโดยที่ตัวอักษรตัวเล็กและตัวอักษรตัวใหญ่นั้นมีความแตกต่างกัน สามารถใช้ตัวเลขหรือขีดเส้นใต้ร่วมด้วยก็ได้ แต่ไม่ควรใช้เครื่องหมายเหล่านี้มาตั้งชื่อ เช่น ^ " ' , - ?] () ~ ! $ @ # <> $ / และหากไฟล์ใดที่ตั้งชื่อขึ้นต้นด้วยจุด "." จะทำให้ไฟล์นั้นเป็น hidden file คือไฟล์ที่ถูกซ่อนไว้ จะไม่สามารถมองเห็นได้โดยใช้คำสั่งทั่วไปจะต้องมี option เพิ่มเติม

Regular files คือไฟล์ทั่วไปที่สร้างขึ้นได้ด้วย Text Editor หรืออาจจะสำเนามาจากไฟล์อื่น หรืออาจจะเป็นโปรแกรมใช้งานต่างๆก็ได้

Directory files คือไฟล์ที่เก็บไฟล์ทั่วไปหรือจะเก็บไฟล์ที่เป็น Directory ด้วยกัน ที่เรียกว่า Sub Directory ก็ได้ โดยที่ Directory บนสุด (root) ของ ยูนิกซ์จะแทนด้วย " / "

Special files เป็นไฟล์พิเศษจะมีอยู่สองแบบคือ Character device file และ Block device file ทั้งสองแบบจะเป็นไฟล์ device driver โดยส่วนใหญ่จะเก็บไว้ที่ /dev แต่ไฟล์ทั้งสองจะแตกต่างกัน ที่การรับส่งข้อมูล นั่นคือ Character device file จะรับส่งข้อมูลที่ละตัวอักษร แต่ Block device file จะรับส่งข้อมูลเป็นบล็อก

UNIX demean seckets ใน BSD Unix หรือ Name pipes ใน AT&T UNIX

Symbolic Link files หรือไฟล์เชื่อมต่อ การเชื่อมต่อของไฟล์มี 2 ลักษณะคือ

2.12.4.1 Hard Link การเชื่อมต่อแบบนี้จะใช้ I-node เดียวกับไฟล์ต้นฉบับ เหมือนกับมีการสร้างไฟล์ใหม่ แต่ใช้ค่า I-node เดิม และ I-node จะมีตัวนับจำนวนไฟล์ที่เชื่อมต่อด้วย หากแก้ไขไฟล์ใดไฟล์หนึ่งจะมีผลกระทบส่งถึงกัน เพราะข้อมูลเก็บที่เดียวกัน แต่ข้อมูลต้องอยู่ที่ partition เดียวกัน ทำให้ประหยัดเนื้อที่ สามารถอ้างถึงข้อมูลได้จากหลายๆที่

2.12.4.2 Symbolic Link การเชื่อมต่อแบบนี้จะสร้าง I-node ของตัวเองขึ้นมาใหม่ เหมือนกับ shutcut ของ windows 95 โดยที่หากเปลี่ยนแปลงต้นฉบับจะมีผลกับ link file แต่ถ้าลบ link file จะไม่มีผลใดๆต่อไฟล์ต้นฉบับ สามารถใช้ได้ทั้งที่อยู่ partition เดียวกัน หรือต่าง partition กันก็ได้เราสามารถที่จะแยกประเภทของไฟล์ต่างได้โดยใช้คำสั่ง ls -l แล้วจะแสดงสัญลักษณ์ โดยจะแสดงดังนี้

Type Sysbol Create Remove

Text file - cp , mv ,etc rm

Directory p mkdir rm -r , rmdir

Character device v mknod rm

Block device b mknod rm

Unix domain socket s socket rm

Name pipes p mknod rm

link file l ln -s rm

โครงสร้างไฟล์ไดเรคเทอรีของระบบยูนิกซ์ส่วนใหญ่จะเป็นแบบ Filesystem Hierarchy Standard (FHS) โดยการจัดลำดับชั้นจะเป็นแบบต้นไม้หัวกลับ โดยเริ่มจากชั้นแรกที่เป็น ราก หรือ root เขียนแทนด้วย / ไฟล์แต่ละไฟล์อาจจะสร้างขึ้นมาเองหรือเป็นโปรแกรมก็ได้ ไฟล์ลักษณะนี้จะเป็น

ไฟล์ไดเรคเทอรี การจัดไฟล์ระบบนี้จะทำให้การจัดไฟล์เป็นระบบ ง่ายต่อการดูแลรักษา โดยจะมีโครงสร้างหลักเป็นดังนี้

/ เป็นไดเรคเทอรี root ที่เก็บไฟล์ kernel ของระบบ

/bin เป็นไดเรคเทอรีที่ใช้เก็บคำสั่งทั่วไปของระบบ

/dev เป็นไดเรคเทอรีที่ใช้เก็บไฟล์ที่เกี่ยวกับอุปกรณ์ต่างๆ

/etc เป็นไดเรคเทอรีที่ใช้เก็บไฟล์ที่เป็น config files ของเครื่อง

/etc/X11 เป็นไดเรคเทอรีที่ใช้เก็บไฟล์ที่เป็น config files ของ x windows

/etc/skel เป็นไดเรคเทอรีที่ใช้เก็บไฟล์ที่เป็นไฟล์ต้นฉบับที่จะถูกสำเนาไปยัง home user

/lib เป็นไดเรคเทอรีที่ใช้เก็บไฟล์ไลบรารี สำหรับให้โปรแกรมต่างๆเรียกใช้

/sbin เป็นไดเรคเทอรีที่ใช้เก็บไฟล์คำสั่งของผู้ดูแลระบบ

/usr เป็นไดเรคเทอรีที่ใช้เก็บไฟล์โปรแกรมของผู้ใช้ทั่วไป

/var เป็นไดเรคเทอรีที่ใช้เก็บไฟล์ข้อมูลทั่วไปของระบบ

2.13 PERMISSION

ยูนิกซ์เป็นระบบ OS ที่ใช้ไฟล์ต่างๆ ร่วมกันหากทุกคน มีสิทธิที่จะกระทำต่อทุกไฟล์เท่ากัน ย่อมจะทำให้เกิดความวุ่นวาย ดังนั้นในระบบยูนิกซ์จึงมี user id และ group id ประจำ user แต่ละคน จึงทำให้ที่ home directory ของแต่ละ user จะเป็นที่ ที่ user แต่ละคนมีสิทธิมากที่สุด เมื่อ user สร้างไฟล์ขึ้นมาก็จะทำให้ มีชื่อของผู้สร้างติดอยู่ด้วย การจำกัดสิทธิการเข้าถึงไฟล์ออกเป็น 3 กลุ่มคือ

Owner เจ้าของไฟล์หรือผู้ที่สร้างไฟล์

Group ผู้ใช้กลุ่มเดียวกับผู้ใช้ไฟล์ คือ ผู้ใช้ที่มี gid เดียวกับเจ้าของไฟล์

Other คนอื่นๆหรือใครก็ได้

สิทธิในไฟล์จะประกอบไปด้วย

Read Permission สิทธิในการอ่าน แทนด้วย r

Write Permission สิทธิในการเขียน แทนด้วย w

Execute Permission สิทธิในการ Run แทนด้วย x

user สามารถที่จะดู Permission ของไฟล์และชนิดของไฟล์ได้โดยคำสั่ง

$ ls -la

-rwxr--r-- 1 wihok Special 5223 May 12 10:10 .profile

-rwxr--r-- 1 wihok Special 2022 May 12 10:13 .kshrc

drwx------ 2 wihok Special 1024 May 12 10:34 mail

-rw-r--r-- 1 wihok Special 11211 May 12 11:01 test

จากตัวอย่างจะเห็นว่า มีทั้งหมด 7 filed ดังนี้

1 File Type และ Permission

2 จำนวน link

3 เจ้าของ (owner)

4 กลุ่ม (group)

5 ขนาดของไฟล์ (byte)

6 วัน-เวลาที่ update

7 ชื่อไฟล์

มาดูกันที่ field ที่ 1 ที่เป็น Permission โดย

อักษรตัวที่ 1 แสดงชนิดของไฟล์

อักษรตัวที่ 2-4 แสดง Owner

อักษรตัวที่ 5-7 แสดง Group

อักษรตัวที่ 8-10 แสดง Other

เช่นจากตัวอย่าง ไฟล์ .kshrc มี permission เป็น -rwxr--r-- หมายความว่า Owner สามารถที่จะ อ่าน เขียน และ Run ได้ แต่ user กลุ่มเดียวกับ owner และ other อ่านได้เพียงอย่างเดียว สังเกตได้ว่าหากไม่มี permission จะแสดงด้วย

คำสั่งเปลี่ยน Permission

การเปลี่ยน permission ของไฟล์กระทำได้โดยผู้ที่เป็น Admin ของระบบ หรือเจ้าของไฟล์นั้น โดยมีคำสั่งคือ

2.13.1 คำสั่ง chmod ใช่เปลี่ยน permission ของไฟล์มีวิธีการเปลี่ยนได้ 2 วิธี คือ

Absolute Permission

รูปแบบ $ chmod ตัวเลข filename

โดยสามารถหาตัวเลขที่มาใส่ได้จากการแทนค่าน้ำหนักของแต่ละบิตลงไปคือ

บิต r แทนน้ำหนักด้วย 4

บิต w แทนน้ำหนักด้วย 2

บิต x แทนน้ำหนักด้วย 1

บิต - แทนน้ำหนักด้วย 0

โดยหากต้องการให้ permission ใดก็แทนค่าของบิตนั้นลงไปแล้วนำเลขน้ำหนักของแต่ละบิตมารวมกัน (คิดทีละส่วนโดยแยกเป็น owner , group และ other) เช่น
จะกำหนดสิทธิไฟล์ test ให้ owner สามารถอ่าน เขียน และ Run ได้ group สามารถอ่านและ run ได้ ส่วน other สามารถ run ได้เพียงอย่างเดียวคิดได้ดังนี้

Permission rwx r-x --x

Number 7 5 1

ใช้คำสั่ง : $ chmod 751 test

Relative Permission

ผู้ใช้ไฟล์ เครื่องหมาย สิทธิ

u (เจ้าของไฟล์) + เพิ่มสิทธิ r (อ่าน)

g (กลุ่มเดียวกับเจ้าของไฟล์)

- ลดสิทธิ w (เขียน)

o (คนทั่วไปใครก็ได้)

= กำหนดสิทธิ x (Run

a (ทุกคนทุกกลุ่มที่กล่าวมา)

เช่นจะเปลี่ยน permission ของไฟล์ .kshrc จาก rwxr--r-- เป็น rwxrw-r--

$ chmod g+w .kshrc

หรือจะเปลี่ยน permission ของไฟล์ .profile จาก rwxr--r-- เป็น rwxrw-rw-

$ chmod go+w .profile

2.13.2 คำสั่ง chown ใช้เปลี่ยนผู้เป็นเจ้าของไฟล์ เช่น$ chown newuser test คือเปลี่ยน field ที่ 3 จากการใช้คำสั่ง ls -la จากเจ้าของเดิมคือ wihok เป็น newuser

2.13.3คำสั่ง chgrp ใช้เปลี่ยนกลุ่มผู้เป็นเจ้าของไฟล์ เช่น $ chgrp newgroup test คือเปลี่ยน field ที่ 4 จากการใช้คำสั่ง ls -la จากเจ้าของเดิมคือ Special เป็น newgroup

2.13.4 Text Editor

Text Editor ที่ใช้ในระบบยูนิกซ์ที่เห็นบ่อยคือ โปรแกรม pico และโปรแกรม vi แต่ pico ไม่ได้มีอยู่ใน unix ทุกตัว การใช้งานง่าย ไม่ต้องจำคำสั่งต่างเพราะมีอธิบายอยู่แล้วที่ด้านล่างหน้าจอภาพ สามารถพิมพ์ text ได้เลย แต่ text editor ที่ชื่อ vi จะเป็น text editor ที่มีอยู่ในทุกยูนิกซ์ การใช้งานค่อนข้างยาก ดังนั้นผู้เขียนจะแนะนำเฉพาะการใช้ vi เท่านั้น

การเรียกใช้งาน text editor

$ pico filename หรือ $ pico

2.14 การควบคุมการเคลื่อนที่เซอร์โว

เซอร์โว(Servo)เป็นอุปกรณ์ขนาดเล็กที่ประกอบด้วยวงจรอิเล็คทรอนิกส์และเครื่องกลไกทำหน้าที่แปลงสัญญานควบคุมไฟฟ้าจากเครื่องรับวิทยุไปเป็นการเคลื่อนที่เพื่อนำไปควบคุมอุปกรณ์บังคับการบินต่างๆ เช่น บังคับ Aileron หรือ Rudder เพื่อเลี้ยว เป็นต้น โดยที่เซอร์โวจะมีแกนหมุนติดอยู่แกนหมุนนี้จะสามารถเปลี่ยนตำแหน่งได้ด้วยการส่งสัญญาณควบคุมเข้าไปที่เซอร์โวและเซอร์โวจะยังคงตำแหน่งของแกนหมุนไว้จนกว่าสัญญาณควบคุมจะมีการเปลี่ยนแปลงไป มุมของแกนหมุนนี้สามารถเปลี่ยนแปลงได้ตั้งแต่ 0-210 องศา แต่โดยทั่วไปจะใช้งานอยู่ในช่วง 0-180 องศา สายสัญญาณที่ต่อเข้าไปที่เซอร์โวจะมีสามเส้นคือ สายสีดำจะต่อกับไฟลบ สายสีแดงจะต่อกับไฟบวก และสายสีขาวเป็นสายสัญญาณควบคุม (สีของสายไฟสำหรับเซอร์โวยี่ห้อ FUTABA ถ้าเป็นยี่ห้ออื่นจะแตกต่างจากนี้)

2.17.1 Pulse

pulse เป็นสัญญาณไฟฟ้าที่มีการเปลี่ยนแปลงในช่วงเวลาสั้นๆ ตัวอย่างเช่น ถ้าเราเปิดสวิทช์ไฟ (มีสถานะเป็น 1) เป็นเวลาหนึ่งวินาที แล้วปิดสวิทช์ไฟ (มีสถานะเป็น 0) อีกหนึ่งวินาที สลับกันไปเรื่อยๆก็จะเป็นการสร้างสัญญาณ pulse ที่มีช่วงเวลาเปิดและปิดรวมเป็น 2 วินาที หรือเรียกว่ามีคาบเวลาเท่ากับ 2 วินาที เป็นต้น ดังนั้นการวัดคาบเวลาของ pulse (ใช้สัญลักษณ์ T) จะเริ่มวัดตั้งแต่ช่วงของการเปิดสวิทช์ไฟครั้งแรกจนถึงการเปิดสวิทช์ไฟครั้งต่อไป คาบเวลายังมีความสัมพันธ์กับความถี่ (ใช้สัญลักษณ์ f) ของการเปิดและปิดไฟด้วย โดยเราสามารถคำนวณหาความถี่ของ pulse ได้จากสูตร f = 1/T duty cycle จะวัดเป็นเปอร์เซ็นต์ (%) โดยถ้าช่วงเวลาของการเปิดและปิดไฟเท่ากันเราจะเรียกว่าสัญญาณ pulse นี้มีค่า duty cycle เป็น 50% ถ้าช่วงเวลาในการเปิดไฟน้อยกว่าช่วงเวลาในการปิดไฟ ค่า duty cycle ก็จะลดลง เช่นถ้าเราเปิดไฟครึ่งวินาที แล้วปิดไฟหนึ่งวินาทีครึ่ง ค่า duty cycle จะเป็น 25% เป็นต้น ดังนั้นค่า duty cycle หาได้จาก

$$\text{duty cycle (D)} = \frac{\text{ระยะเวลาของการเปิดไฟ}}{\text{คาบเวลาของ pulse}} \times 100$$

2.17.2 ความถี่ (Frequency)

ความถี่ (Frequency หรือ f) คือจำนวน pulse ต่อหนึ่งวินาที มีหน่วยเป็นเฮิรท์ (Hz) เช่น ถ้าเราเปิดไฟและปิดไฟ 2 ครั้งในเวลาหนึ่งวินาที เราจะนับจำนวน pulse ได้เท่ากับ 2 นั้นก็คือความถี่ 2 เฮิรท์นั้นเอง เราสามารถคำนวณหาคาบเวลาของ pulse ได้จากสูตร T = 1/fโดยปกติเซอร์โวจะใช้ไฟเลี้ยงอยู่ระหว่าง 4 - 6 โวลท์ การควบคุมตำแหน่งของแกนหมุนจะใช้สัญญาณเป็น pulse สั้นๆโดยมีความกว้างของสัญญาณอยู่ระหว่าง 1 - 2 ms (มิลลิวินาที) โดยถ้าสัญญาณควบคุมมี Pulse เป็น 1 ms

แกนหมุนของเซอร์โวจะอยู่ที่ตำแหน่ง 0 องศา ถ้าเป็น 1.5 ms แกนหมุนจะอยู่ที่ตำแหน่ง 90 องศา และ 2 ms แกนหมุนจะอยู่ที่ตำแหน่ง 180 องศา หลังจากส่ง pulse ควบคุมเข้าไปที่เซอร์โวแล้วจะต้องรออีก 20 ms (0.02 วินาที) (แต่เซอร์โวสามารถรับคาบเวลาของ pulse ได้ตั้งแต่ 10 ms จนถึง 30 ms) จึงจะส่ง pulse อีกอันตามเข้าไปได้อีก จากข้อกำหนดนี้ทำให้เราสามารถคำนวนหาค่า duty cycle ได้ดังนี้

ช่วงเวลาเปิดสัญญาณต่ำสุด = 1 ms

ช่วงเวลาของสัญญาณควบคุมทั้งหมด = 20 ms

ค่า duty cycle ต่ำสุด $= \frac{1}{20} \times 100 = 5\%$

ช่วงเวลาเปิดสัญญาณสูงสุด = 2 ms

ช่วงเวลาของสัญญาณควบคุมทั้งหมด = 20 ms

ค่า duty cycle สูงสุด $= \frac{2}{20} \times 100 = 10\%$

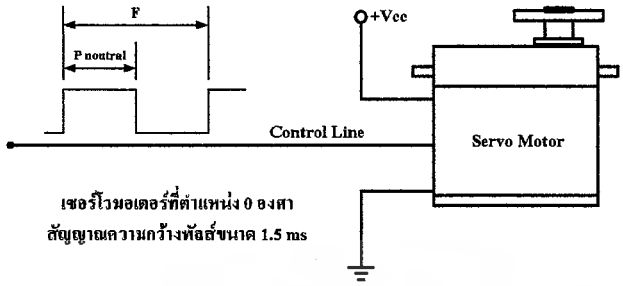
ดังนั้นเราต้องการค่า duty cycle ที่เซอร์โวต้องการอยู่ระหว่าง 5 - 10%

การเคลื่อนที่ของเซอร์โวมอเตอร์ การทำงานในส่วนของเซอร์โวมอเตอร์ที่ใช้งานในระบบการจับภาพเคลื่อนไหวโดยใช้กล้องดิจิตอล ในการควบคุมการทำงานของเซอร์โวมอเตอร์ทำได้โดยการป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ ซึ่งตำแหน่ง และทิศทางการหมุนของมอเตอร์นี้ จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้น ๆ โดยทั่วไปแล้ว ความกว้างของสัญญาณพัลส์ จะมีจุดให้อ้างอิง 3 จุด ดังนี้

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือจุดกึ่งกลางของมอเตอร์

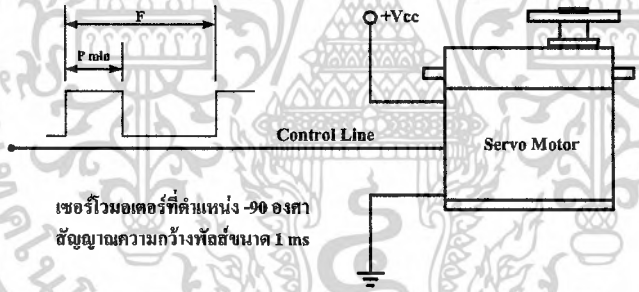
- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม -90 องศา หรือในทิศทางทวนเข็มนาฬิกา

- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม +90 องศา หรือในทิศทางทวนเข็มนาฬิกา



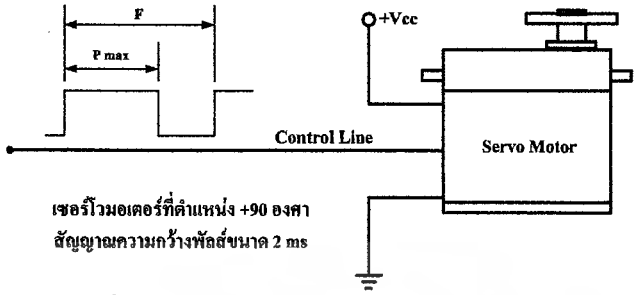
รูปที่2.15 การควบคุมเซอร์โวที่ตำแหน่ง 0 องศา

รูปที่ 2.15 แสดงสัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือจุดกึ่งกลางของมอเตอร์ ส่วนการควบคุมการทำงานของเซอร์โวมอเตอร์ จะใช้หลักการสร้างสัญญาณพัลส์ขนาดความกว้าง 1.5 ms ส่งไปควบคุมการทำงานของมอเตอร์



รูปที่2.16การควบคุมเซอร์โวที่ตำแหน่ง -90 องศา

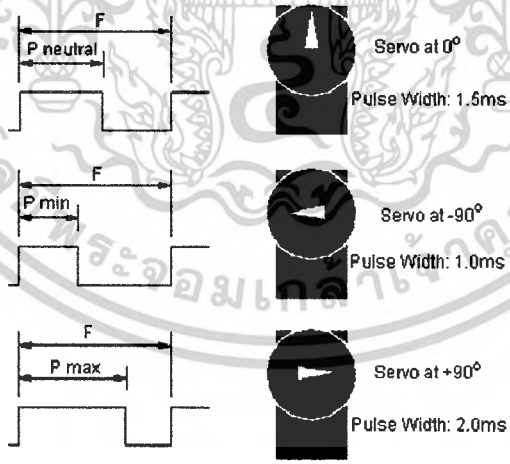
รูปที่ 2.16 แสดงสัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม -90 องศา หรือในทิศทางทวนเข็มนาฬิกา ส่วนการควบคุมการทำงานของเซอร์โวมอเตอร์ จะใช้หลักการสร้างสัญญาณพัลส์ขนาดความกว้าง 1 ms ส่งไปควบคุมการทำงานของมอเตอร์



รูปที่2.17 การควบคุมเซอร์โวที่ตำแหน่ง +90 องศา

รูปที่ 2.17 แสดงสัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม +90 องศา หรือในทิศทางทวนเข็มนาฬิกา ส่วนการควบคุมการทำงานของเซอร์โวมอเตอร์ จะใช้หลักการสร้างสัญญาณพัลส์ขนาดความกว้าง 2 ms ส่งไปควบคุมการทำงานของมอเตอร์

ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆนั้น ก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่าง ๆ โดยอ้างอิงจากจุดทั้ง 3 จุด ที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม -45 องศา จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms เป็นต้น และสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุก ๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้

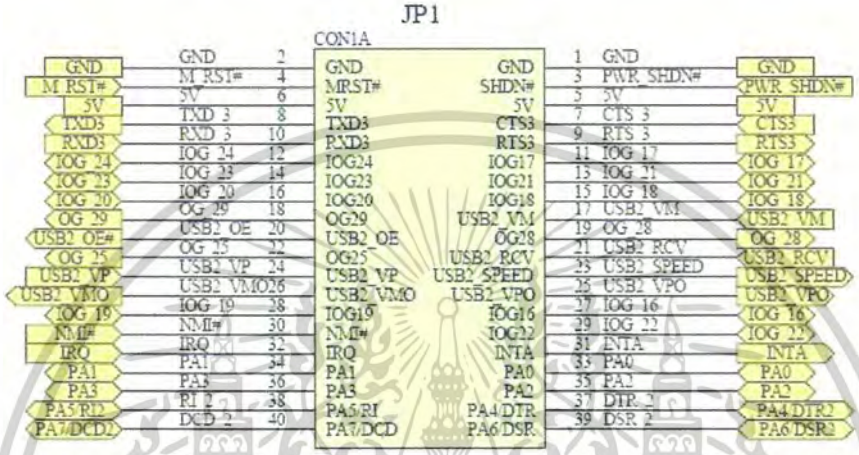


รูปที่2.18 การหมุนในตำแหนงต่างของเซอร์โว

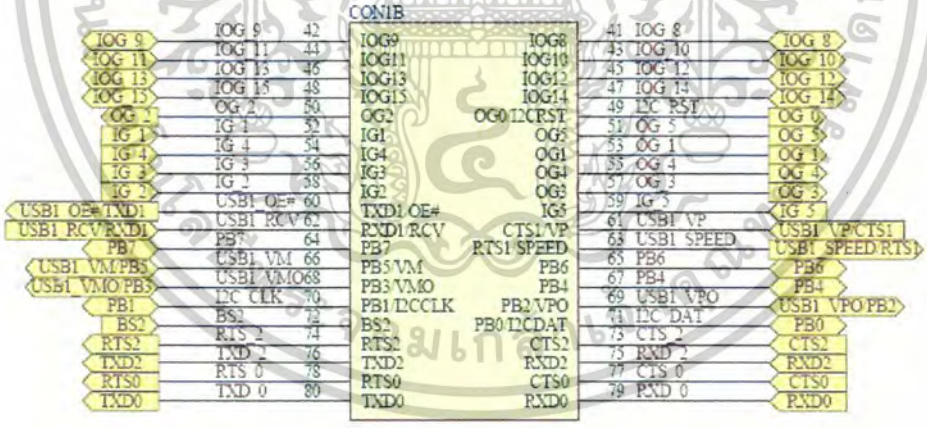
บทที่ 3

การออกแบบระบบ

3.1 กำหนดพอร์ตใช้งาน Microprocessor ETRAX 100LX RISC



รูปที่ 3.1 ETRAX 100LX RISC Connector 1



รูปที่ 3.2 ETRAX 100LX RISC Connector 2

จากโครงสร้างของ Microprocessor ETRAX 100LX RISC และข้อมูล หน้าที่ของพอร์ตต่างๆ เราสามารถเลือกพอร์ตใช้งาน โดยพอร์ตที่เราทำการเลือกนั้นจะต้อง สามารถทำหน้าที่เป็น Input/output พอร์ตได้ทำให้ส่งข้อมูลไปให้ Microprocessor MCS-51 สร้างสัญญาณ PWM เพื่อใช้ในการควบคุมความกว้างของพัลส์ที่จะส่งไปควบคุมการหมุนของ Servo Motor

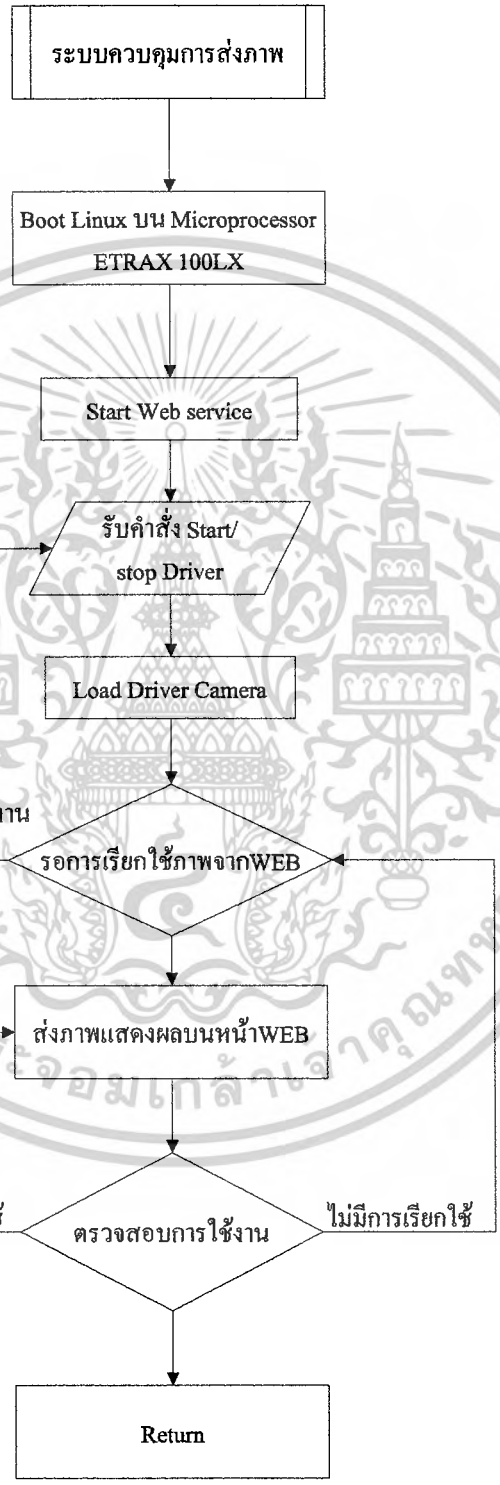
ตำแหน่งขา	ชื่อพอร์ต	ชนิดของพอร์ต	หน้าที่	หมายเหตุ
11	PG17	PORTG	เลือกใช้งาน Servo Motor	-
14	PG24	PORTG	ข้อมูลบิตที่ 1	-
13	PG21	PORTG	ข้อมูลบิตที่ 2	-
16	PG23	PORTG	ข้อมูลบิตที่ 3	-
15	PG18	PORTG	ข้อมูลบิตที่ 4	-

ตารางที่ 3.1 แสดงหน้าที่การทำงานของพอร์ตต่าง ๆ

รูปที่ 3.3 Connector MCS-51

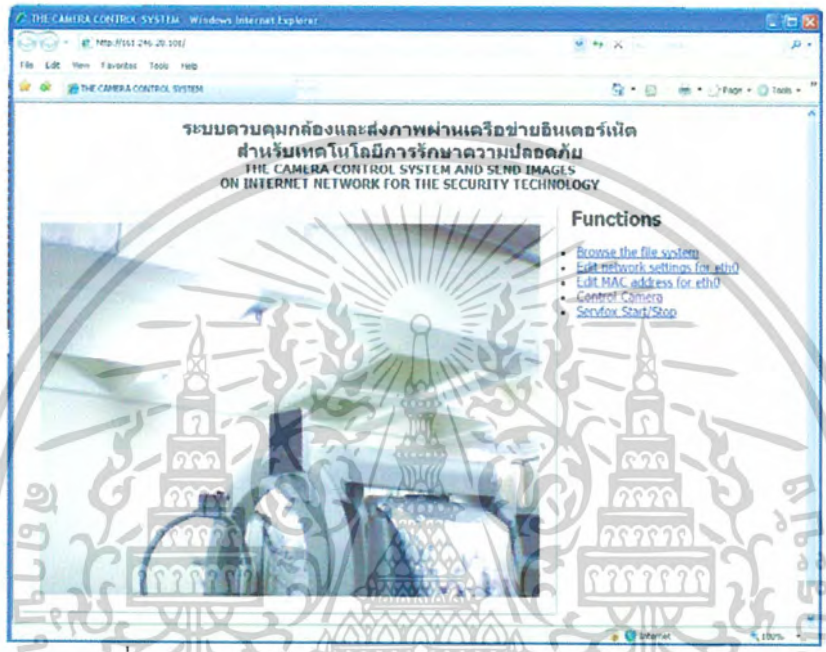
จากโครงสร้างของ Microprocessor MCS-51 และข้อมูล หน้าที่ของพอร์ตต่างๆ เราสามารถเลือกพอร์ตใช้งาน โดยพอร์ตที่เราทำการเลือกนั้นจะต้อง สามารถทำหน้าที่เป็น Input/output พอร์ตได้แต่ในการใช้งานเป็น Module PWM (Pulse Width Modulation) เพื่อใช้ในการควบคุมความกว้างของพัลส์ที่จะส่งไปควบคุมการหมุนของ Servo Motor นั้นเราจะเลือกใช้ P0 เป็น Input และ P2.0 กับ P2.1 เป็น Output ซึ่งพอร์ตที่เราทำการเลือกแสดงได้ดังรูปที่ 3.3

3.2 การเขียนโปรแกรมควบคุมการส่งภาพ

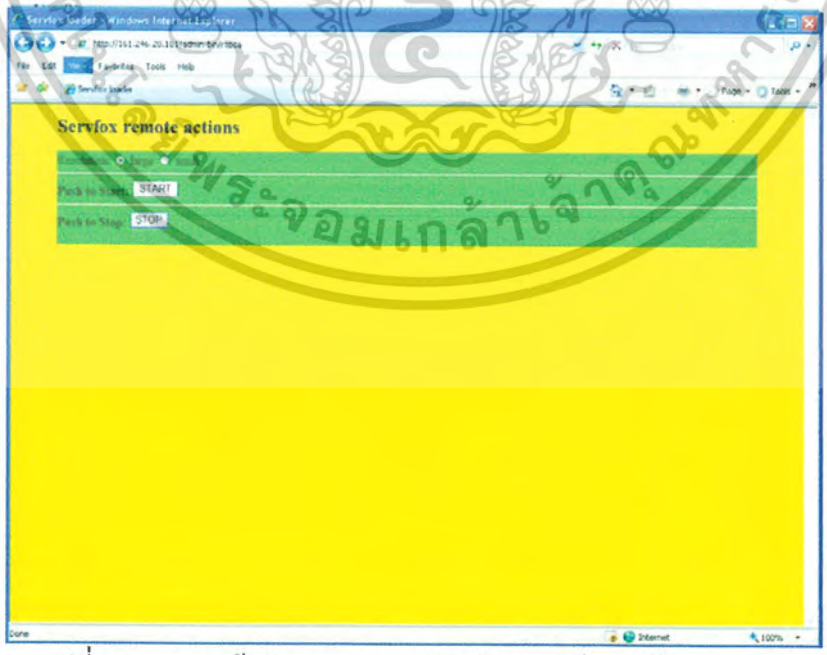


รูปที่ 3.4 Flowchart แสดงขั้นตอนการส่งภาพไปยังผู้ใช้งาน

เริ่มการทำงานจาก Microprocessor ETRAX 100LX RISC เปิดการทำงาน web service ส่งหน้าเว็บแสดงผลให้ผู้ใช้งาน ผู้ใช้งานสามารถสามารถควบคุมการเรียกใช้ Driver ของกล้องจากหน้าเว็บได้โดยตรง



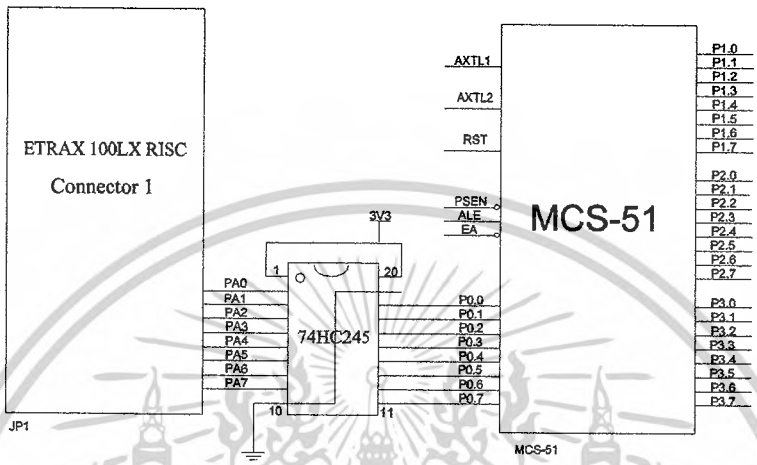
รูปที่ 3.5 แสดงหน้าต่างแสดงภาพให้ผู้ใช้งานได้เห็น



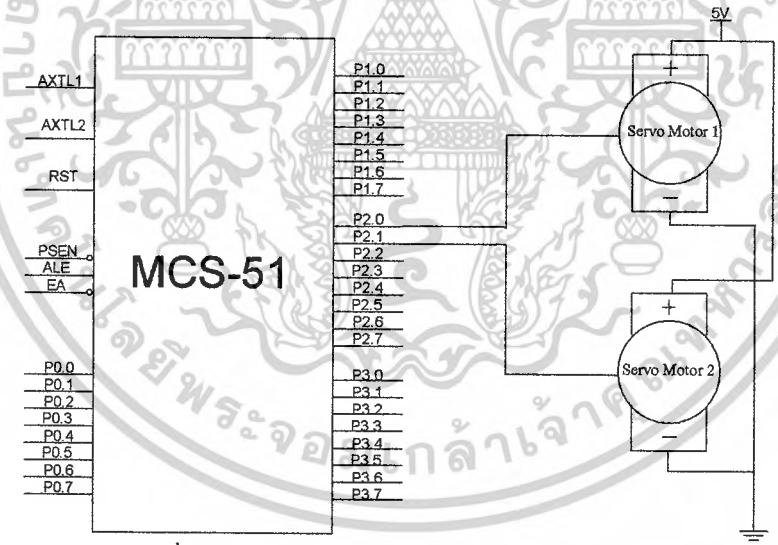
รูปที่ 3.6 แสดงหน้าต่างควบคุมการเปิด-ปิดการใช้งานกล้อง

3.3 การควบคุม Servo Motor

3.3.1 วงจรควบคุม



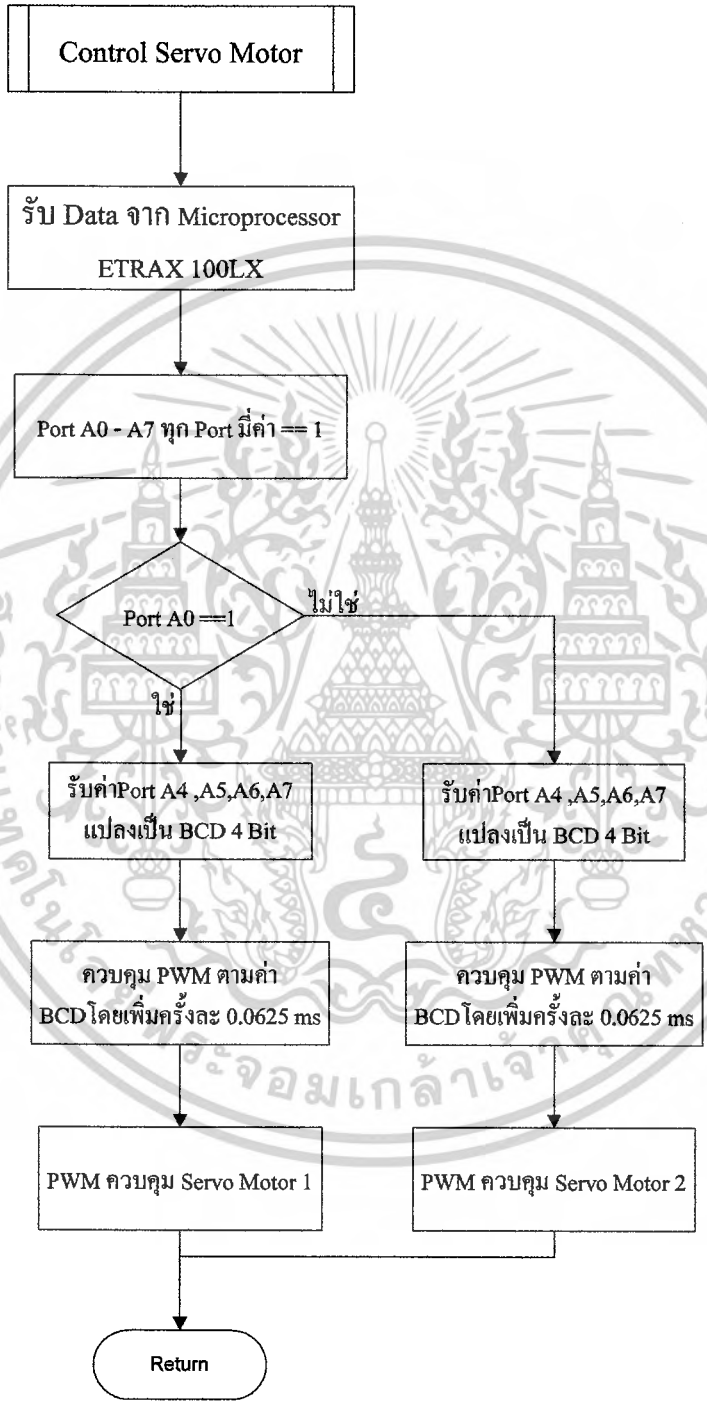
รูปที่ 3.7 วงจรเชื่อมต่อระบบควบคุม



รูปที่ 3.8 วงจรควบคุมการหมุนของ Servo Motor

จากวงจรในรูปที่ 3.8 เราจะใช้ Port A0 ถึง Port A7ของ Microprocessor ETRAX 100LX RISC เป็นพอร์ตส่งข้อมูล โดยจะใช้งานร่วมกันกับ IC 74HC245 เป็นตัวทำหน้าที่ปรับระดับแรงดันของสัญญาณเอาท์พุท Microprocessor ETRAX 100LX ที่มีแรงดัน 3.3Vให้สามารถเชื่อมต่อกับ Microprocessor MCS-51 ที่มีระดับแรงดัน 5V

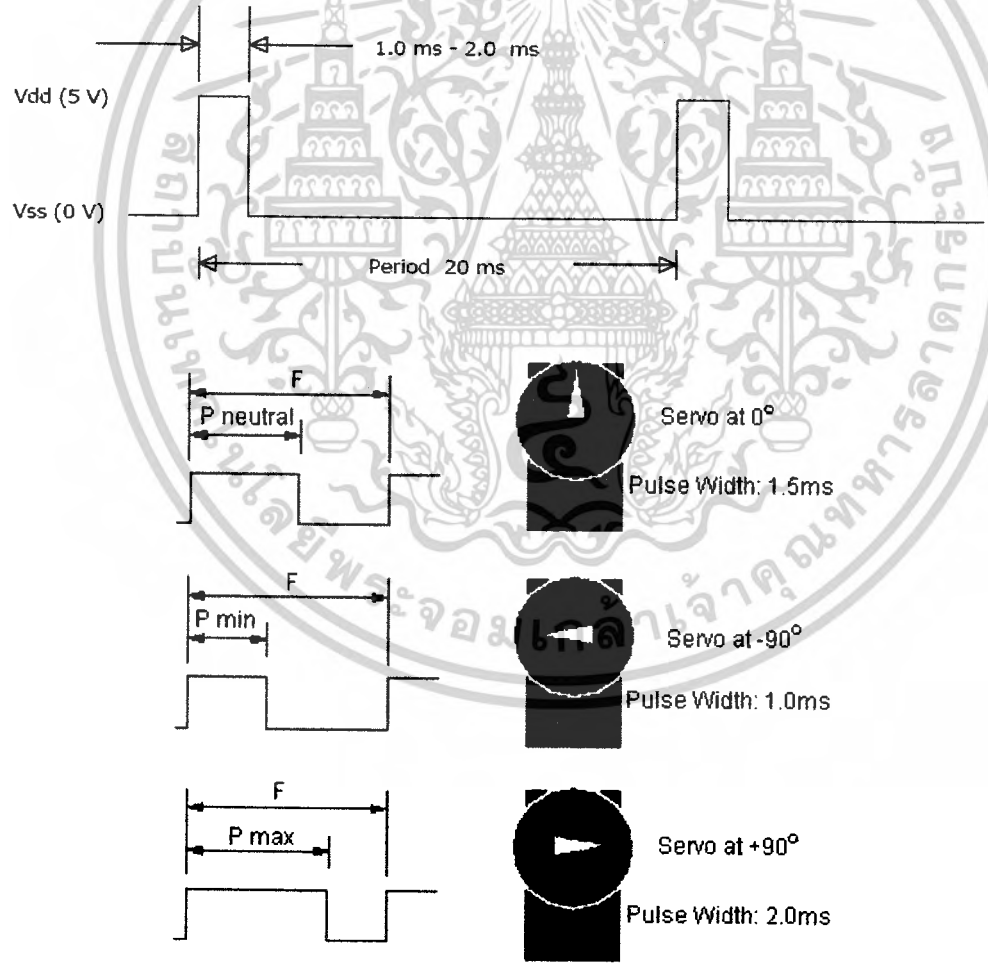
3.3.2 การเขียนโปรแกรมควบคุมการหมุนของ Servo Motor (MCS-51)



รูปที่ 3.9 Flowchart ของโปรแกรมสร้าง PWM ควบคุม Servo Motor

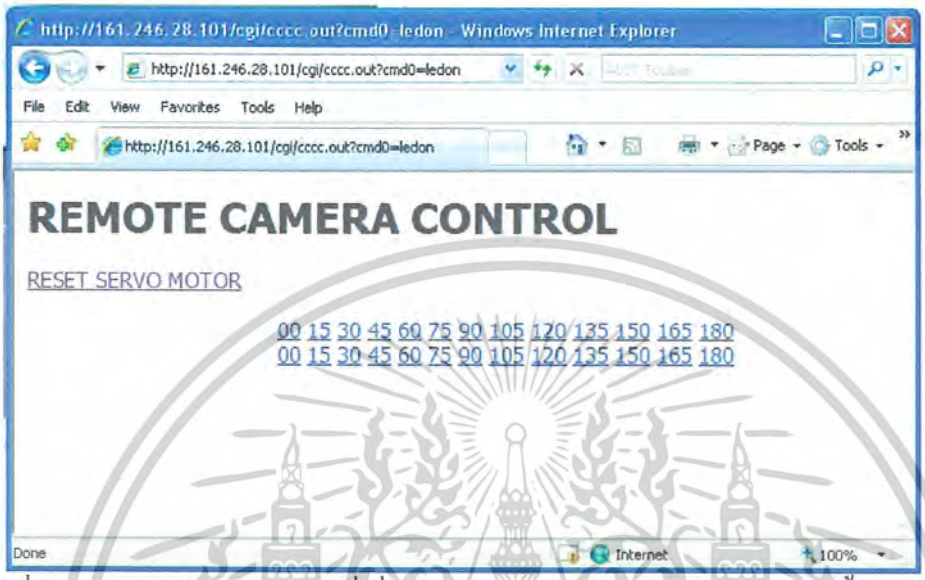
การส่งข้อมูลจากหน้าเว็บจะมีขนาด 5 บิต โดยจะเริ่มทำงานครั้งแรกเมื่อทุกบิตเป็นหนึ่งทั้งหมดหลังจากนั้นบิตแรกคือ Port A0 ทำหน้าที่เป็นบิตที่เลือกว่าข้อมูลนั้นจะไปควบคุม Servo Motor โดยถ้าเป็นศูนย์จะไปควบคุม Servo Motor 1 ถ้าเป็นหนึ่งจะไปควบคุม Servo Motor 2 หลังจากนั้นแล้วนำข้อมูลบิตที่เหลือคือ Port A4, Port A5, Port A6, Port A7 มาประมวลผลเพื่อสร้างเป็นสัญญาณความกว้างพัลส์ค่าต่าง ๆ ที่ใช้ควบคุมการทำงานของ Motor Servo ซึ่งโดยทั่วไปแล้วจะมีตำแหน่งอ้างอิงความกว้างของพัลส์อยู่ 3 จุด ดังแสดงในรูปที่ 3.10

เนื่องจาก Servo หมุนได้ 180 องศา ข้อมูลที่ใช้ควบคุมจึงมีความละเอียด 15 องศาต่อ 1 ค่า ดังนั้นก็เพราะต้องคิดความละเอียดของพัลส์ออกมาเป็นร้อยละ ของความกว้างของคาบเวลา แล้วซึ่งคือค่าเริ่มต้นของตำแหน่งที่ 0 องศา แล้วนำไปกำหนดในฟังก์ชันของ PWM ที่มีความละเอียด 4 บิต



รูปที่ 3.10 ตำแหน่งของ Motor Servoกับความกว้างของสัญญาณพัลส์

3.3.3 การเขียนโปรแกรมควบคุมการหมุนของ Motor Servo จากระบบ Network

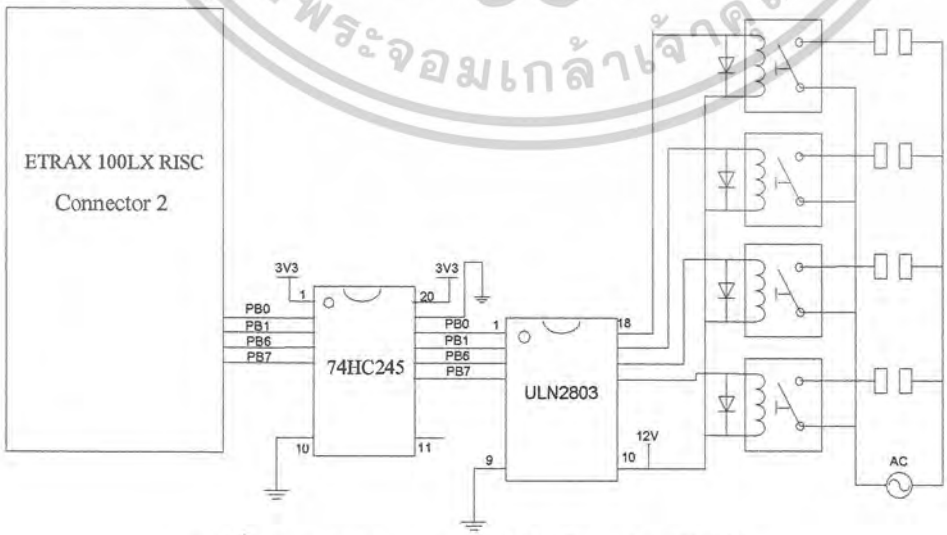


รูปที่ 3.11 แสดงหน้าต่างควบคุมที่เพิ่มส่วนการควบคุมตำแหน่ง Motor Servo ทั้งสองตัว

ใช้ตัวเลขเป็นตัวส่งค่าตำแหน่งในการควบคุม Motor Servo ซึ่งในแต่ละแกนจะสามารถส่งค่าได้ตั้งแต่ 0 จนถึง 16 แล้วทางด้าน Server และ MCS-51 ก็จะนำค่าที่ได้ไปคำนวณเพื่อสร้างเป็นความกว้างสัญญาณพัลส์ต่าง ๆ แล้วส่งไปควบคุมการหมุนของ Servo อีกทีหนึ่ง

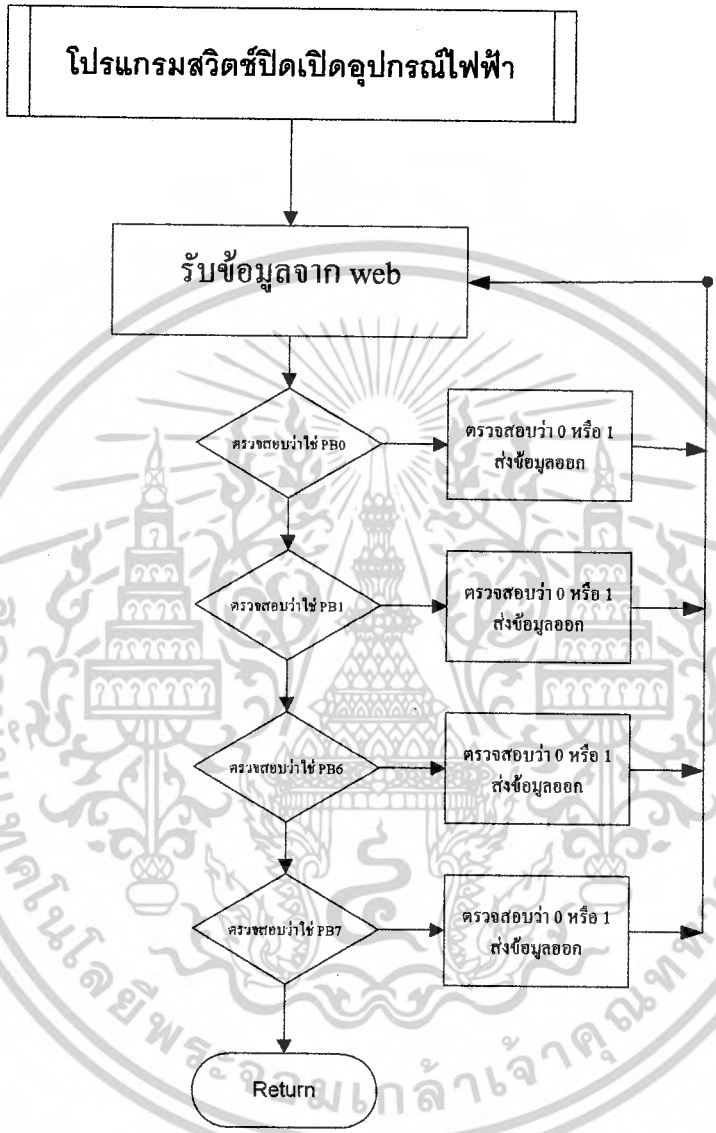
3.4 การควบคุมสวิตช์ปิดเปิดอุปกรณ์ไฟฟ้า

3.4.1 วงจรควบคุมสวิตช์



รูปที่ 3.12 วงจรควบคุมสวิตช์ปิดเปิดอุปกรณ์ไฟฟ้า

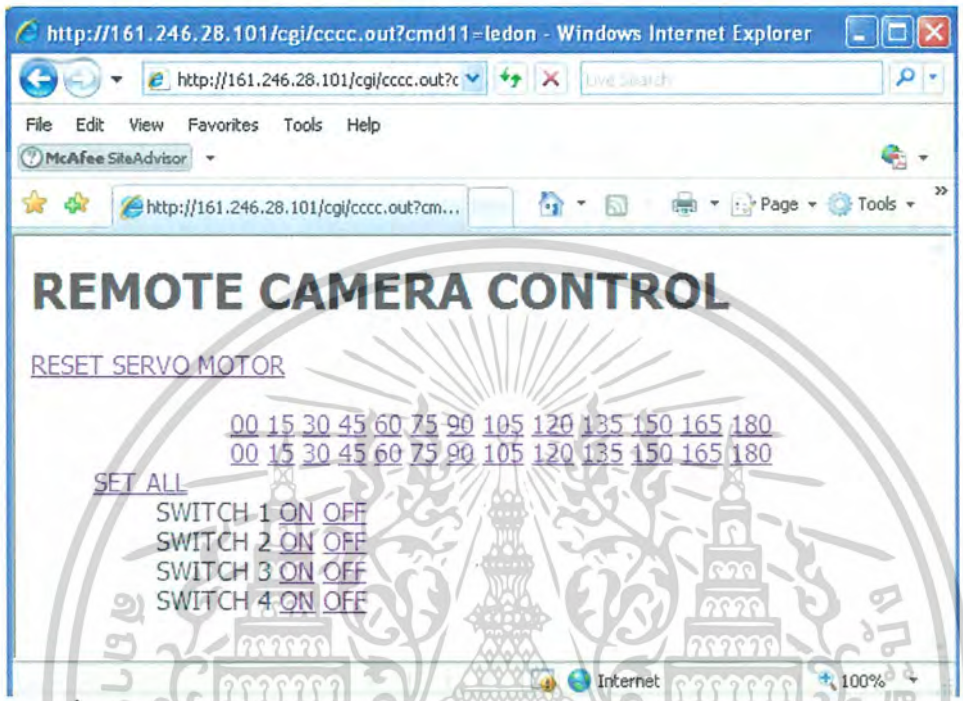
3.4.2 การเขียนโปรแกรมควบคุมสวิตช์ปิดเปิดอุปกรณ์ไฟฟ้า



รูปที่ 3.13 Flowchart ของโปรแกรมควบคุมสวิตช์ปิดเปิดอุปกรณ์ไฟฟ้า

การส่งข้อมูลจากหน้าเว็บจะมีขนาด 4 บิต โดยจะเริ่มทำงานโปรแกรมจะทำการตรวจสอบว่าเป็นคำสั่งเป็นของ Port ใดหลังจากนั้นหลังจากนั้นทำหน้าที่เป็นบิตที่เลือกว่าข้อมูลนั้นเป็นคำสั่งกำหนดค่าเอาท์พุตให้เป็นค่าตามคำสั่งจากหน้าเว็บ

3.4.3 โปรแกรมควบคุมสวิตช์ปิดเปิดอุปกรณ์ไฟฟ้าจากระบบ Network



รูปที่ 3.14 แสดงหน้าต่างโปรแกรมควบคุมสวิตช์อุปกรณ์ไฟฟ้าจากระบบ Network

จากรูป 3.14 จะเห็นได้ว่าจะแยกสวิตช์แต่ละช่องออกจากกันเพื่อที่สะดวกต่อการใช้งาน โดยโปรแกรมนี้จะอยู่ในหน้าเดียวกับโปรแกรมควบคุมเซอร์โวมอเตอร์ทำให้ง่ายต่อการใช้งาน

บทที่ 4

วิธีการทดลองและผลการทดลอง

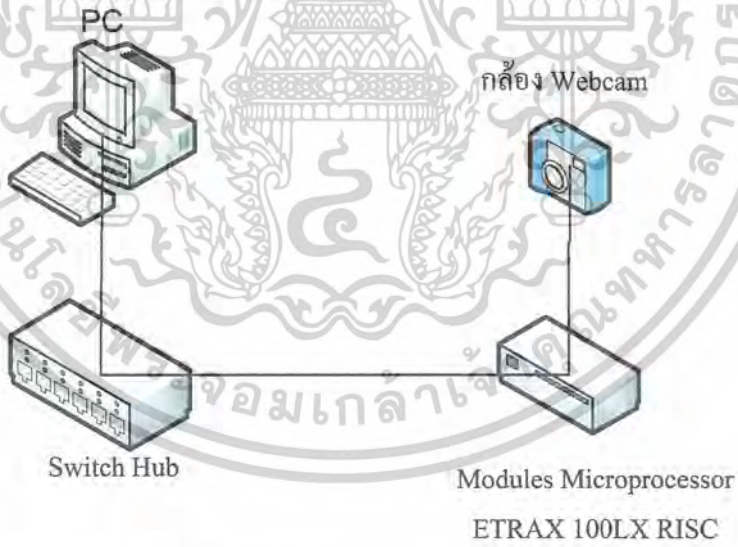
4.1 อุปกรณ์ที่ใช้ในการทดลอง

- Switch Hub	1 ตัว
-กล้อง Webcam USB	1 ตัว
-Servo Motor	2 ตัว
- สาย LAN	2 เส้น
- คอมพิวเตอร์	1 เครื่อง
- Modules Microprocessor ETRAX 100LX RISC	1 ชุด

4.2 การควบคุมอุปกรณ์การส่งภาพโดยการ Telnet

4.2.1 วิธีการทดลอง

1) ต่ออุปกรณ์ต่างๆดังรูป 4.1



รูปที่ 4.1 การทดลองการส่งภาพจากกล้องWebcam

2) ทำการตั้งค่า IP ของคอมพิวเตอร์ให้อยู่ในวงเดียวกับ IP ของ MODULE EREAX 100LX RISC

3) ทำการตั้งค่า IP ของ MODULE EREAX 100LX RISC โดยการเปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC ไปที่เมนู Edit network settings for eth0 ทำการตั้งค่าเป็น 161.246.28.101

4) ทำการโหลดไดเวอร์ของกล้องโดยการTelnet แล้วใช้คำสั่งโหลดไดเวอร์

1. insmod videodev
2. insmod ./spca5xx.ko
3. ./servfox –s 640x480

5) เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC

4.2.2 ผลการทดลอง

(ก) (ข)

รูปที่ 4.2 ผลการทดลองการส่งภาพโดยการ Telnet

จากการทดลองจะพบว่าในการ Telnet จะสามารถควบคุมการส่งภาพได้โดยจะเห็นการส่งข้อมูลภาพและเวลาที่ใช้ในการส่งภาพโดยเวลาที่ได้ในแต่ละภาพจะไม่เท่ากันขึ้นอยู่กับขนาดของภาพที่ส่งซึ่งจะเห็นได้จากรูปที่ 4.2

4.3 การควบคุมอุปกรณ์การส่งภาพโดยผ่านหน้าเว็บ

4.3.1 วิธีการทดลอง

1) ต่ออุปกรณ์ต่างๆดังรูป 4.1
2) ทำการตั้งค่า IP ของคอมพิวเตอร์ให้อยู่ในวงเดียวกับ IP ของ MODULE EREAX 100LX RISC
3) ทำการตั้งค่า IP ของ MODULE EREAX 100LX RISC โดยการเปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC ไปที่เมนู Edit network settings for eth0 ทำการตั้งค่าเป็น 161.246.28.101

4) ทำการโหลดไดเวอร์ของกล้องโดยการเปิด Internet Explorer เรียกไปยัง IP ของMODULE EREAX 100LX RISC
5) เลือก Servfox start/stop จะได้หน้าควบคุมกล้องขึ้นมาเลือก Start
6) เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC

4.3.2 ผลการทดลอง

(ก) (ข)

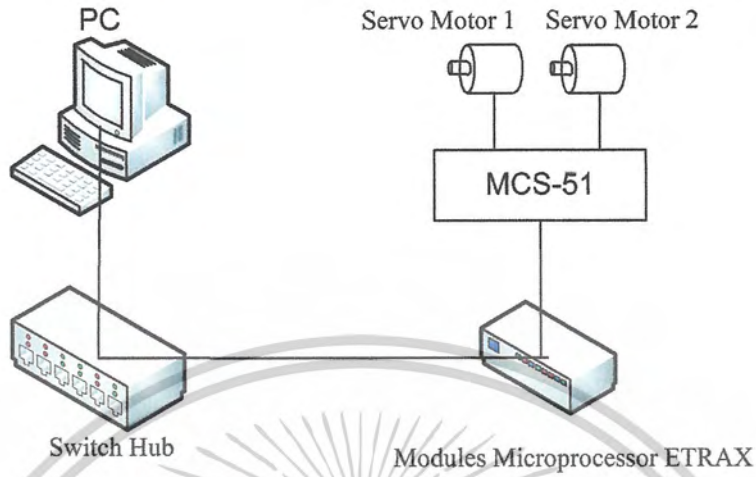
รูปที่ 4.3 ผลการทดลองการส่งภาพโดยผ่านหน้าเว็บ

จากการทดลองจะพบว่าการควบคุมจากหน้าเว็บจะสามารถควบคุมการส่งภาพได้และเห็นภาพได้เหมือนกับการควบคุมภาพจากการTelnet ซึ่งจะเห็นได้จากรูปที่ 4.3

4.4 การควบคุมการหมุนของเซอร์โวมอเตอร์

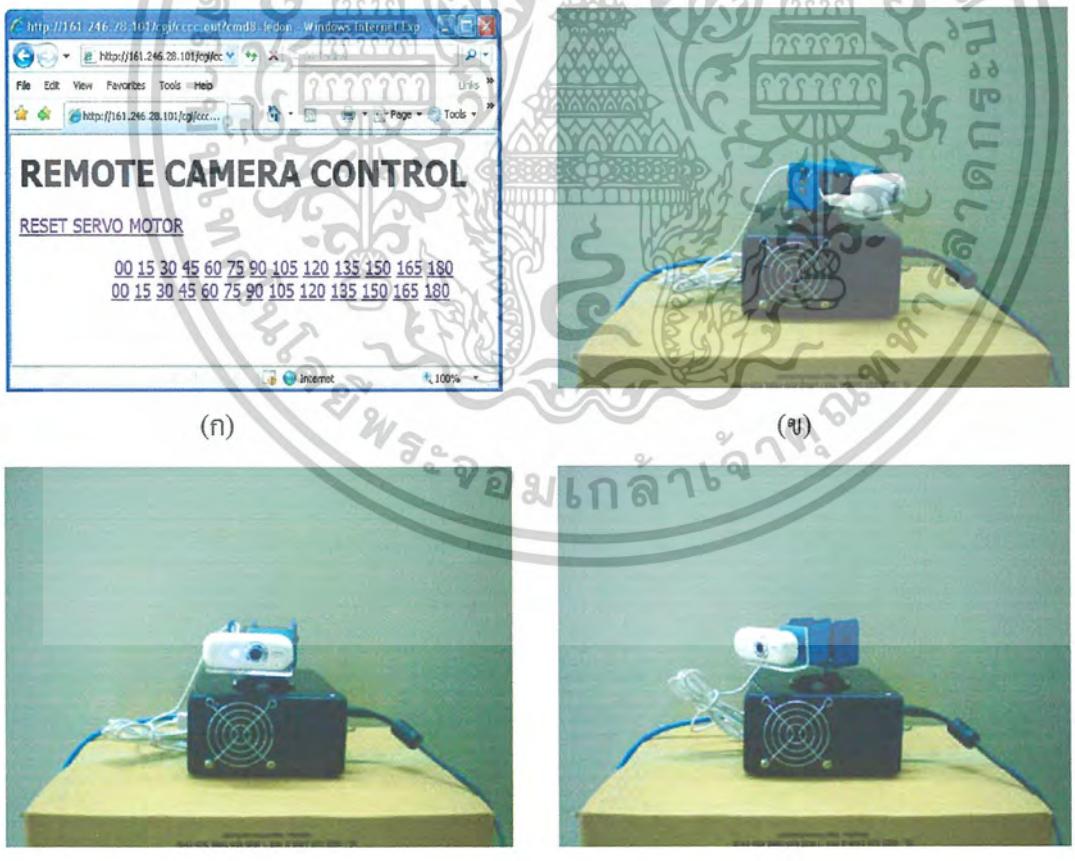
4.4.1 วิธีการทดลอง

1) ต่ออุปกรณ์ต่างๆดังรูป 4.4
2) ทำการตั้งค่า IP ของคอมพิวเตอร์ให้อยู่ในวงเดียวกับ IP ของ MODULE EREAX 100LX RISC
3) ทำการตั้งค่า IP ของ MODULE EREAX 100LX RISC โดยการเปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC ไปที่เมนู Edit network settings for eth0 ทำการตั้งค่าเป็น 161.246.28.101
4) เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC
5) เปิดหน้า Control Camera ทดสอบโดยเลือกมุมของกล้องต่างๆ



รูปที่ 4.4 การทดลองการควบคุมเซอร์โวมอเตอร์

4.4.2 ผลการทดลอง



(ก) (ข)

(ค) (ง)

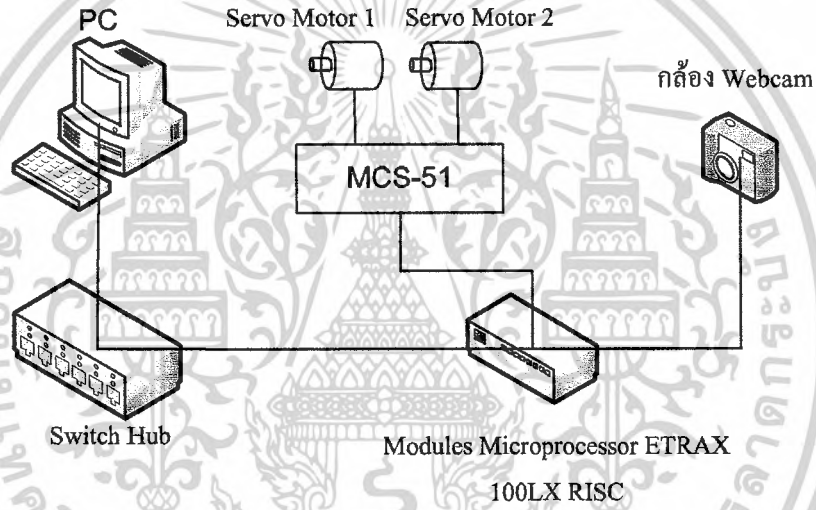
รูปที่ 4.5 ผลการทดลองการควบคุมเซอร์โวมอเตอร์

จากผลการทดลองจะเห็นว่าการปรับมุมมองของกล้องสามารถทำได้ตั้งแต่ 0 องศา โดยเพิ่มครั้งละ 15 องศา ไปจนถึง 180 องศา ตามคุณสมบัติของ เซอร์โวมอเตอร์ทั้งแกนนอน และแกนตั้ง ทำให้ได้มุมมองที่ คลอบคลุม พื้นที่ได้ครึ่งวงกลม สามารถที่จะนำไปประยุกต์ใช้ประโยชน์ได้มากมายไม่จำเป็นต้องเกี่ยวกับงานรักษาความปลอดภัยเพียงอย่างเดียว

4.5 การควบคุมการหมุนของเซอร์โวมอเตอร์พร้อมกับส่งภาพ

4.5.1 วิธีการทดลอง

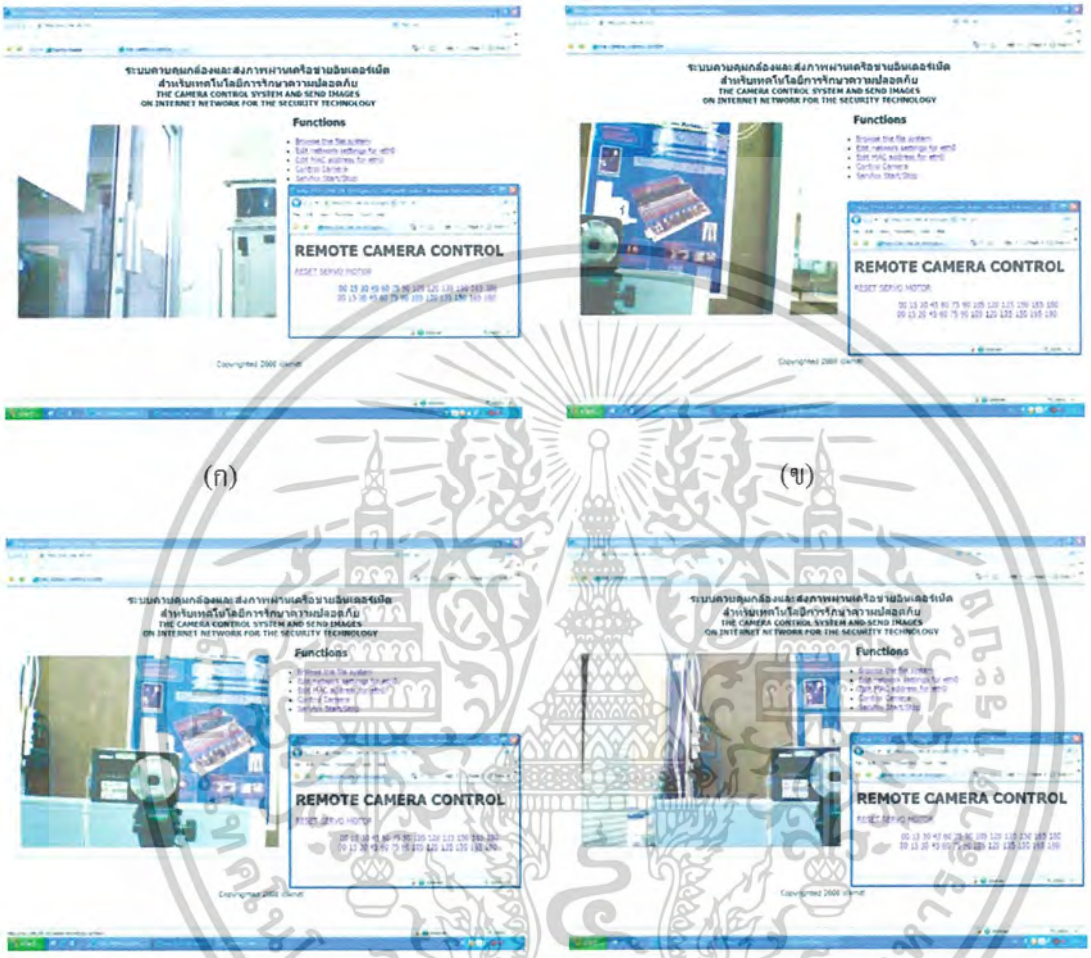
1) ต่ออุปกรณ์ดังรูป 4.6



รูปที่ 4.6 การทดลองการหมุนของเซอร์โวมอเตอร์พร้อมกับส่งภาพ

2) ทำการตั้งค่า IP ของคอมพิวเตอร์ให้อยู่ในวงเดียวกับ IP ของ MODULE EREAX 100LX RISC
3) ทำการตั้งค่า IP ของ MODULE EREAX 100LX RIS เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC
4) เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC
5) เลือก Servfox start/stop จะได้หน้าควบคุมกล้องขึ้นมาเลือก Start
6) เปิดหน้า Control Camera ทดสอบโดยเลือกมุมของกล้องต่างๆ

4.5.2 ผลการทดลอง



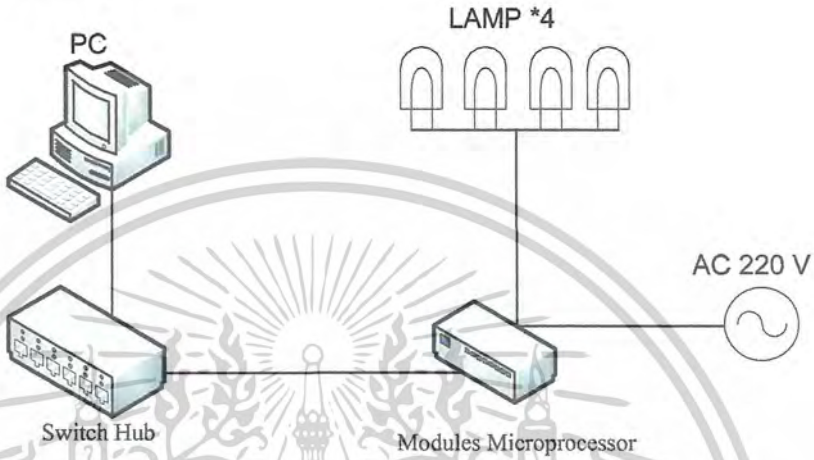
รูปที่ 4.7 ผลการทดลองการการหมุนของเซอร์โวมอเตอร์พร้อมกับส่งภาพ

จากผลการทดลองจะเห็นว่าการปรับมุมมองของกล้องจะยังสามารถส่งภาพไปพร้อมกันได้ โดยมุมที่เปลี่ยนครั้งละ 15 องศายังคงทำให้ภาพที่ได้ยังคงต่อเนื่องทำให้สามารถมองภาพเห็นภาพได้ครอบคลุมได้ทั้งหมดตั้งแต่ 0 องศา ไปจนถึง 180 องศา ตามคุณสมบัติของเซอร์โวมอเตอร์ทั้งแกนนอน และแกนตั้ง ทำให้ได้มุมมองที่คลอบคลุมพื้นที่ได้ครึ่งวงกลม สามารถที่จะนำไปประยุกต์ใช้ประโยชน์ในงานอื่นๆได้

4.6 การควบคุมการปิดเปิดสวิตช์สวิตช์อุปกรณ์ไฟฟ้า

4.6.1 วิธีการทดลอง

1) ต่ออุปกรณ์ดังรูป 4.8



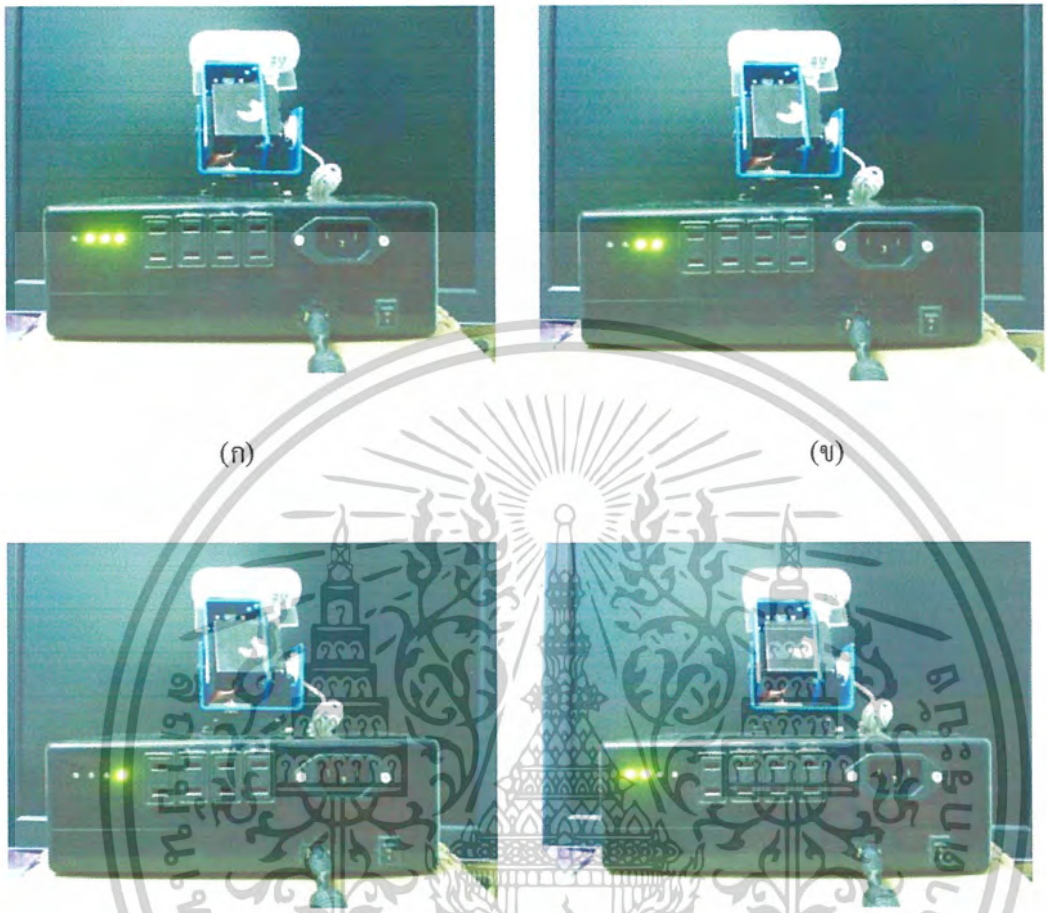
รูปที่ 4.8 การทดลองการปิดเปิดสวิตช์สวิตช์อุปกรณ์ไฟฟ้า

2) ทำการตั้งค่า IP ของคอมพิวเตอร์ให้อยู่ในวงเดียวกับ IP ของ MODULE EREAX 100LX RISC
3) การตั้งค่า IP ของ MODULE EREAX 100LX RIS เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC
4) เปิด Internet Explorer เรียกไปยัง IP ของ MODULE EREAX 100LX RISC
5) เปิดหน้า Control Camera ทดสอบโดยเลือกสวิตช์แต่ละตัวโยทำการปิดเปิด

4.6.2 ผลการทดลอง

(ก) (ข)

รูปที่ 4.9 การทดลองการปิดเปิดสวิตช์สวิตช์รีเซ็ต



(ก) (ข)

(ค) (ง)

รูปที่ 4.10 ผลการทดลองการปิดเปิดสวิตช์สวิตช์อุปกรณ์ไฟฟ้า

จากผลการทดลองจะเห็นสวิตช์แต่ละตัวสามารถควบคุมการปิดเปิดได้แยกต่อกันโดย สามารถสั่งงานผ่านระบบอินเตอร์เน็ตทำให้สามารถควบคุมอุปกรณ์ไฟฟ้าในระยะไกลได้

บทที่ 5
สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

การควบคุมอุปกรณ์การส่งภาพโดยการ Telnet

เมื่อต่อวงจรตามรูปและทำการ Telnet แล้วใช้คำสั่งโหลดไดเวอร์แล้วจะพบว่าจะเห็นภาพที่หน้าเว็บทันทีที่พิมพ์คำสั่งเสร็จโดยสามารถเลือกขนาดของภาพได้ 2 ขนาดคือ 320x240 และ 640x480 โดยจะเห็นเวลาในการส่งภาพในแต่ละเฟรมในหน้าของการ Telnet ได้ทันที

การควบคุมอุปกรณ์การส่งภาพโดยผ่านหน้าเว็บ

เมื่อต่อวงจรและเปิดหน้าเว็บขึ้นมาจะพบหน้าควบคุมการเปิดการทำงานของกล้องโดยสามารถเลือกขนาดของภาพได้ทันที ในการควบคุมการส่งภาพโดยผ่านหน้าเว็บจะสะดวกกว่าการ Telnet เพราะเปิดแค่เพียงหน้าเว็บเพียงอย่างเดียวก็สามารถควบคุมการทำงานของกล้องได้ทั้งหมดเหมือนกับกับการควบคุมโดยการ Telnet โดยไม่ต้องใช้โปรแกรมอื่นๆในการทำงาน

การควบคุมการหมุนของเซอร์โวมอเตอร์

เป็นการควบคุมการหมุนของเซอร์โวมอเตอร์ ผ่านระบบเครือข่ายคอมพิวเตอร์โดยใช้โปรแกรมที่เขียนให้แสดงผลบนหน้าเว็บทำให้สะดวกต่อการใช้งาน จากผลการทดลองจะเห็นว่าการปรับมุมมองของกล้องสามารถทำได้ตั้งแต่ 0 องศา โดยเพิ่มครั้งละ 15 องศาไปจนถึง 180 องศาตามคุณสมบัติของ เซอร์โวมอเตอร์ทั้งแกนนอน และแกนตั้ง ทำให้ได้มุมมองที่ คลอบคลุม พื้นที่ได้ครึ่งวงกลม สามารถที่จะนำไปประยุกต์ใช้ประโยชน์ได้มากมายไม่จำเป็นต้องเกี่ยวกับงานรักษาความปลอดภัยเพียงอย่างเดียว

การควบคุมการหมุนของเซอร์โวมอเตอร์พร้อมกับส่งภาพ

เป็นการควบคุมการหมุนของเซอร์โวมอเตอร์และส่งภาพไปพร้อมกัน ผ่านระบบเครือข่ายคอมพิวเตอร์โดยใช้โปรแกรมที่เขียนให้แสดงผลบนหน้าเว็บ จากการทดลองจะสามารถทำงานทั้งสองอย่างพร้อมกันและสามารถส่งภาพไปยังผู้ใช้หลายคนได้ในเวลาเดียวกันได้

การควบคุมการปิดเปิดสวิตช์ไฟฟ้าผ่านระบบอินเตอร์เน็ต

เป็นการควบคุมสวิตช์จำนวน 4 ช่องที่สามารถควบคุการทำงานได้เหมือนการควบคุมเซอร์โวมอเตอร์ โดยสามารถแยกการควบคุมได้ง่ายต่อการใช้งาน จากการทดลองสามารถนำไปประยุกต์ใช้งานในที่ห่างไกลหรืออันตรายได้

สรุปการทำงานทั้งหมดไม่ว่าจะควบคุมอุปกรณ์การส่งภาพโดยการTelnet หรือควบคุมอุปกรณ์การส่งภาพโดยผ่านหน้าเว็บ สามารถที่จะแสดงภาพผ่านหน้าเว็บและสามารถเปลี่ยนแปลงทิศทางของกล้องปลายทางได้อย่างอิสระหรือควบคุมอุปกรณ์ไฟฟ้าปลายทางผ่านเครือข่ายคอมพิวเตอร์นั้นเป็นอิสระต่อกัน ทำให้ผู้ใช้งานง่ายสะดวกและมีประสิทธิภาพ

5.2 ปัญหาและวิธีการแก้ไข

5.2.1 เกิดความร้อนของวงจรภาคจ่ายแรงดัน 5 โวลต์ที่จ่ายแรงดันให้กับ Microprocessor ETRAX 100LX RISC กล้อง USB และ Microprocessor MCS-51

การแก้ไขก็คือเปลี่ยนภาคจ่ายไฟจาก 78S05 กระแส 2 แอมแปร์เป็นแบบสวิตชิ่ง LM2575 ที่สามารถรองรับกระแสได้ถึง 3 แอมแปร์และไม่ทำให้เกิดความร้อนกับวงจร

5.2.2 การเขียนโปรแกรมของไมโครคอนโทรลเลอร์ MCS51ใช้การตรวจสอบข้อมูลแบบ 1บิตทำใหม่โปรแกรมมีขนาดใหญ่และทำงานช้า

การแก้ไขคือเปลี่ยนการตรวจสอบข้อมูลเป็นแบบ 8 บิตโดยทำการตรวจสอบข้อมูลครั้งเดียวทั้งพอร์ตทำให้ขนาดของโปรแกรมลดลงจาก 8Kbyts เหลือเพียง 3Kbyts ทำให้โปรแกรมทำงานเร็วขึ้น

5.2.3 ตอนที่ทำการเปิดการทำงานเข้าไปที่ Microprocessor ETRAX 100LX RISC จะทำการรีเซ็ตค่ากลับมาอยู่ที่ตำแหน่ง 0 องศา ทั้งในแกนนอนและแกนตั้ง ทำให้ช่วงที่ Motor Servo ทำงานนั้นกินกระแสมาก

แนวทางในการแก้ไขคือ ทำการเพิ่มโปรแกรมให้มีการรีเซ็ตโปรแกรมการเริ่มใช้ครั้งแรกและทำการแยกไฟเลี้ยงในส่วนของ Motor Servo กับส่วนของ ระบบออกจากกันซึ่งก็สามารถช่วยได้ในระดับหนึ่ง

5.3 แนวทางการนำไปประยุกต์ใช้งานอื่น ๆ

1. สามารถประยุคติดกับหุ่นยนต์ไปในสถานที่อันตราย
2. สามารถนำไปควบคุมอุปกรณ์ไฟฟ้าเช่นการ เปิด-ปิด แสงสว่าง ในสถานที่อันตราย
3. ใช้เพื่อตรวจสอบว่าลืม ปิด อุปกรณ์เครื่องใช้ไฟฟ้าตอนออกจากบ้านหรือไม่
4. สามารถนำไปประยุกต์ใช้ในงานรักษาความปลอดภัยอื่น ๆ ได้

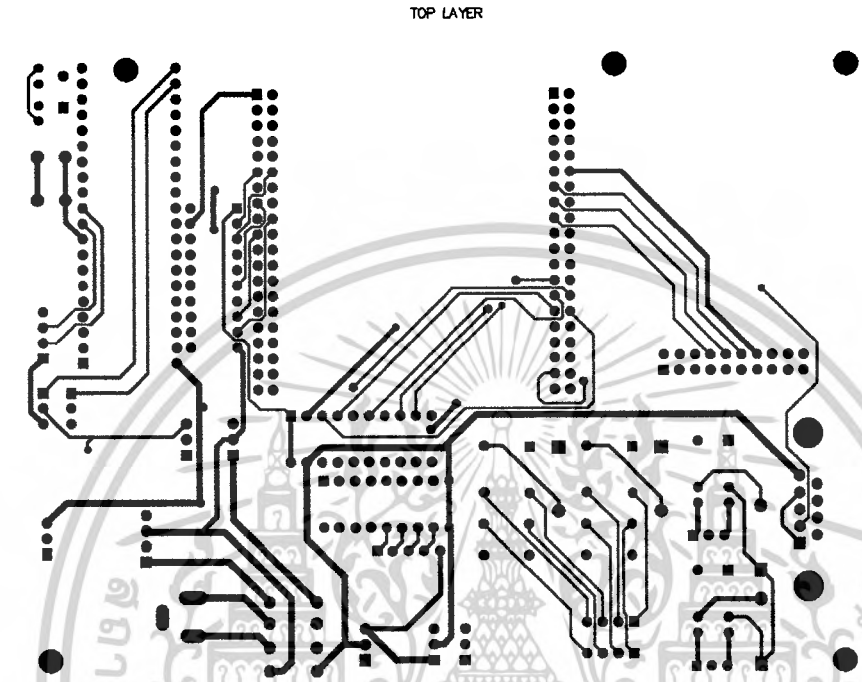
บรรณานุกรม

สิทธิโชค ยอดระยับ. **การเขียนโปรแกรม Digital Image Processing ด้วย Visual Basic.** กรุงเทพฯ : สถาบันส่งเสริมการสอนวิทยาศาสตร์และเทคโนโลยี, 2547.

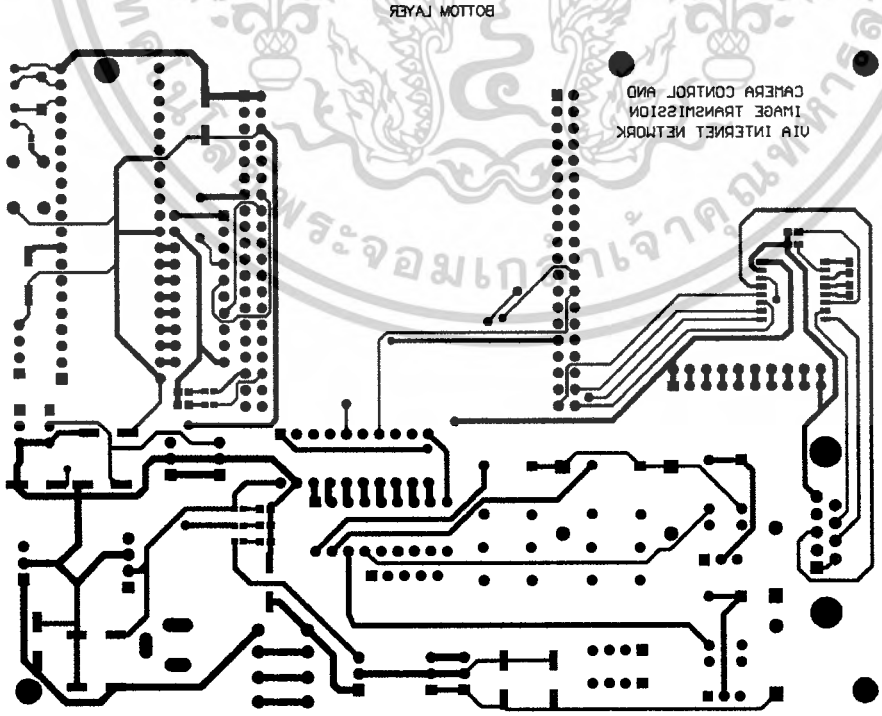
Behrouz A. Forouzan. **การสื่อสารข้อมูลและเครือข่ายคอมพิวเตอร์.** แปลโดย จักกริช พฤษการ. กรุงเทพฯ : ท้อป, 2548.

ภาคผนวก

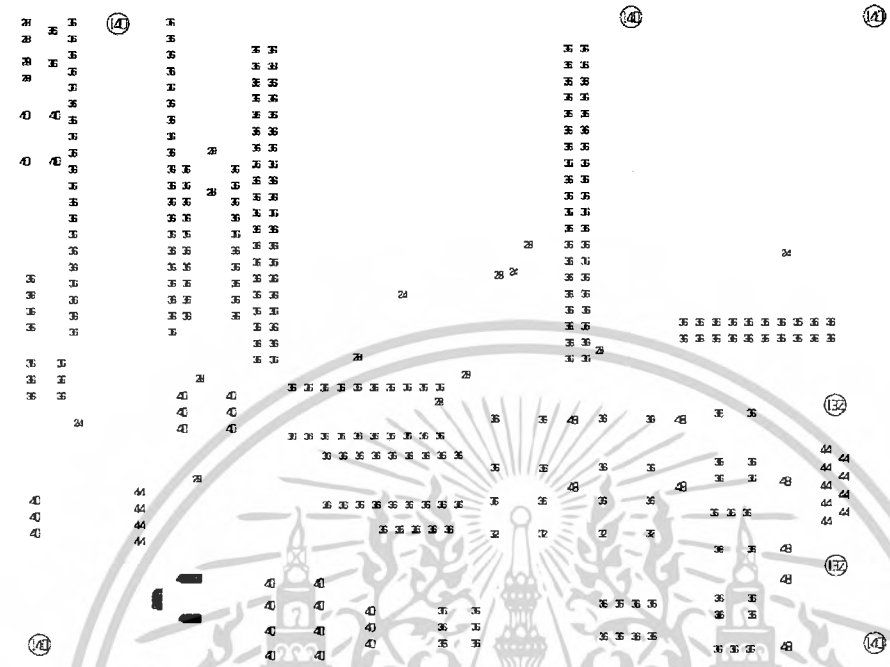
ภาคผนวก A ลายวงจร



แสดงภาพ PCB Top layer ของวงจร



แสดงภาพ PCB Bottom layer ของวงจร



แสดงภาพ PCB Place ของวงจร

แสดง PCB Koop-Out Layer

ภาคผนวก A โปรแกรม MODULE EREAX 100LX RISC

```
#include "unistd.h"
#include "sys/ioctl.h"
#include "fcntl.h"
#include "time.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "linux/gpio_syscalls.h"

// Search an httpd parameter inside the QUERY_STRING environment variable

int read_parameter(char *name, char *value, int maxlen){
  char *pos1, *pos2;
  char *query_string = getenv("QUERY_STRING");
  int success = 0;

  if(query_string){
    pos1 = strstr(query_string, name);
    if(pos1){
      pos1 += strlen(name) + 1;
      pos2 = strstr(pos1, "&");
      if(pos2){
        *pos2 = '\0';
      };
      if(strlen(pos1) >= maxlen){
        pos1[maxlen] = '\0';
      };
      strcpy(value, pos1);
```

```
    success = 1;
   };
  };
  return success;
};

int main(int argc, char *argv[]) {
  char value[128];
// control reset servo motor
  printf("Content-type: text/html\n\n");
  printf("<h1>REMOTE CAMERA CONTROL</h1>");
   if (read_parameter("cmd0",value,10)) {
                    if (strstr("ledon",value))  gpiosetdir(PORTG, DIROUT, PG17);
                    if (strstr("ledon",value))  gpiosetdir(PORTG, DIROUT, PG24);
                    if (strstr("ledon",value))  gpiosetdir(PORTG, DIROUT, PG21);
                    if (strstr("ledon",value))  gpiosetdir(PORTG, DIROUT, PG23);
                    if (strstr("ledon",value))  gpiosetdir(PORTG, DIROUT, PG18);
                    if (strstr("ledon",value))  gpiosetdir(PORTG, DIROUT, PG20);
                    if (strstr("ledon",value)) gpiosetbits(PORTG, PG17);
                    if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                    if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                    if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                    if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                    if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
  }
  printf("<a href=\"?cmd0=ledon\">RESET SERVO MOTOR</a>");
  printf("<tr>\n");
  printf("<td>\n");
  printf("</a>\n");
  printf("</a>\n");
```

```
printf("<ul>\n");
printf("<div align='center'>");
// control  servo motor 1
if (read_parameter("cmd1",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd1=ledon\">00</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd2",value,10)) {
                        if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd2=ledon\">15</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd3",value,10)) {
```

```
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd3=ledon\">30</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd4",value,10)) {
        if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd4=ledon\">45</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd5",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
```

```
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd5=ledon\">60</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd6",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd6=ledon\">75</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
  if (read_parameter("cmd7",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
 }
 printf("<a href=\"?cmd7=ledon\">90</a>");
```

```
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd8",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd8=ledon\">105</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd9",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd9=ledon\">120</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
```

```
if (read_parameter("cmd10",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd10=ledon\">135</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd11",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
 }
printf("<a href=\"?cmd11=ledon\">150</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
  if (read_parameter("cmd12",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
```

```
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd12=ledon\">165</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
   if (read_parameter("cmd13",value,10)) {
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd13=ledon\">180</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
        if (read_parameter("cmd28",value,10)) {
    if (strstr("ledon",value)) gpiosetbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
```

```
  printf("<a href=\"?cmd28=ledon\">scan X</a>");
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("<br>\n");

// control  servo motor 2

  if (read_parameter("cmd14",value,10)) {
     if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
  }
  printf("<a href=\"?cmd14=ledon\">00</a>");
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("</li>\n");
  if (read_parameter("cmd15",value,10)) {
     if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
  }
  printf("<a href=\"?cmd15=ledon\">15</a>");
```

```
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd16",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd16=ledon\">30</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd17",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd17=ledon\">45</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
```

```
if (read_parameter("cmd18",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
            if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
            if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd18=ledon\">60</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd19",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
            if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
            if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd19=ledon\">75</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd20",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
            if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
            if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
```

```
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd20=ledon\">90</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd21",value,10)) {
     if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd21=ledon\">105</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
  if (read_parameter("cmd22",value,10)) {
     if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
```

```
printf("<a href=\"?cmd22=ledon\">120</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd23",value,10)) {
     if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                 if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                 if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd23=ledon\">135</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
 if (read_parameter("cmd24",value,10)) {
     if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                 if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                 if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd24=ledon\">150</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
```

```
printf("</li>\n");
 if (read_parameter("cmd25",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
}
printf("<a href=\"?cmd25=ledon\">165</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
   if (read_parameter("cmd26",value,10)) {
    if (strstr("ledon",value)) gpioclearbits(PORTG, PG17);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG24);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG21);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG23);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG18);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG20);
}
printf("<a href=\"?cmd26=ledon\">180</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
        if (read_parameter("cmd27",value,10)) {
    if (strstr("ledon",value)) gpiosetbits(PORTG, PG17);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG24);
```

```
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG21);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG23);
                if (strstr("ledon",value)) gpiosetbits(PORTG, PG18);
                if (strstr("ledon",value)) gpioclearbits(PORTG, PG20);
  }
  printf("<a href=\"?cmd27=ledon\">scan Y</a>");
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("<br>\n");
  printf("</div>");
 if (read_parameter("cmd28",value,10)) {
    if (strstr("ledon",value))  gpiosetdir(PORTB, DIROUT, PB0);
    if (strstr("ledon",value))  gpiosetdir(PORTB, DIROUT, PB1);
    if (strstr("ledon",value))  gpiosetdir(PORTB, DIROUT, PB4);
    if (strstr("ledon",value))  gpiosetdir(PORTB, DIROUT, PB6);
    if (strstr("ledon",value))  gpiosetdir(PORTB, DIROUT, PB7);
          if (strstr("ledon",value)) gpiosetbits(PORTB, PB0);
          if (strstr("ledon",value)) gpiosetbits(PORTB, PB1);
          if (strstr("ledon",value)) gpiosetbits(PORTB, PB4);
          if (strstr("ledon",value)) gpiosetbits(PORTB, PB6);
          if (strstr("ledon",value)) gpiosetbits(PORTB, PB7);
           }
  printf("<a href=\"?cmd28=ledon\">SET ALL</a>");
  printf("</td>\n");
  printf("</a>\n");
  printf("<ul>\n");
  printf("</div>");
 // control  switch 1
  printf("SWITCH 1");
```

```
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd29",value,10)) {
        if (strstr("ledon",value)) gpiosetbits(PORTB, PB0);
        }
printf("<a href=\"?cmd29=ledon\">ON</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd30",value,10)) {
        if (strstr("ledon",value)) gpioclearbits(PORTB, PB0);
        }
printf("<a href=\"?cmd30=ledon\">OFF</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("<br>\n");
// control  switch 2
printf("SWITCH 2");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd31",value,10)) {
        if (strstr("ledon",value)) gpiosetbits(PORTB, PB1);
        }
printf("<a href=\"?cmd31=ledon\">ON</a>");
```

```
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd32",value,10)) {
        if (strstr("ledon",value)) gpioclearbits(PORTB, PB1);
         }
printf("<a href=\"?cmd32=ledon\">OFF</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("<br>\n");
// control  switch 3
printf("SWITCH 3");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd33",value,10)) {
        if (strstr("ledon",value)) gpiosetbits(PORTB, PB6);
         }
printf("<a href=\"?cmd33=ledon\">ON</a>");
printf("</td>\n");
printf("</a>\n");
printf("</a>\n");
printf("</li>\n");
if (read_parameter("cmd34",value,10)) {
        if (strstr("ledon",value)) gpioclearbits(PORTB, PB6);
         }
printf("<a href=\"?cmd34=ledon\">OFF</a>");
```

```
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("<br>\n");
// control  switch 3
  printf("SWITCH 4");
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("</li>\n");
  if (read_parameter("cmd35",value,10)) {
          if (strstr("ledon",value)) gpiosetbits(PORTB, PB7);
           }
  printf("<a href=\"?cmd35=ledon\">ON</a>");
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("</li>\n");
  if (read_parameter("cmd36",value,10)) {
          if (strstr("ledon",value)) gpioclearbits(PORTB, PB7);
           }
  printf("<a href=\"?cmd36=ledon\">OFF</a>");
  printf("</td>\n");
  printf("</a>\n");
  printf("</a>\n");
  printf("<br>\n");
  return 0;
}
```

ภาคผนวก A โปรแกรม MCS 51

```
$regfile = "89c52.dat"
```

```
$ramstart = 0
$ramsize = 256
$crystal = 11059200
Config Servos = 2 , Servo1 = P2.0 , Servo2 = P2.1 , Reload = 130
Dim X As Byte , Y As Byte , Z As Byte
Loop1:
    Servo1 = 10
    Servo2 = 10
  If P0 >= 251then
    Goto Loop2
  End If
Goto Loop1
  'motor 1
Loop2:
  If P0 = 4 Then
    Servo1 = 4
    X = 4
  End If
  If P0 = 8 Then
    Servo1 = 5
    X = 5
  End If
  If P0 = 12 Then
    Servo1 = 6
    X = 6
  End If
  If P0 = 16 Then
    Servo1 = 7
    X = 7
  End If
```

```
If P0 = 20 Then
  Servo1 = 8
  X = 8
End If
If P0 = 24 Then
  Servo1 = 9
  X = 9
End If
If P0 = 28 Then
  Servo1 = 10
  X = 10
End If
If P0 = 32 Then
  Servo1 = 11
  X = 11
End If
If P0 = 36 Then
  Servo1 = 12
  X = 12
End If
If P0 = 40 Then
  Servo1 = 13
  X = 13
End If
If P0 = 44 Then
  Servo1 = 14
  X = 14
End If
If P0 = 48 Then
  Servo1 = 15
```

```
  X = 15
End If
If P0 = 52 Then
  Servo1 = 16
  X = 16
End If
'motor 2
  If P0 = 68 Then
  Servo2 = 4
  Y = 4
End If
If P0 = 72 Then
  Servo2 = 5
  Y = 5
End If
If P0 = 76 Then
  Servo2 = 6
  Y = 6
End If
If P0 = 80 Then
  Servo2 = 7
  Y = 7
End If
If P0 = 84 Then
  Servo2 = 8
  Y = 8
End If
If P0 = 88 Then
  Servo2 = 9
  Y = 9
```

```
End If
If P0 = 92 Then
  Servo2 = 10
  Y = 10
End If
If P0 = 96 Then
  Servo2 = 11
  Y = 11
End If
If P0 = 100 Then
  Servo2 = 12
  Y = 12
End If
If P0 = 104 Then
  Servo2 = 13
  Y = 13
End If
If P0 = 108 Then
  Servo2 = 14
  Y = 14
End If
If P0 = 112 Then
  Servo2 = 15
  Y = 15
End If
If P0 = 116 Then
  Servo2 = 16
  Y = 16
End If
Call Scan1
```

```
  Call Scan2
Goto Loop2
Sub Scan1
  If P0 = 132 Then
    Z = 4
    Loop3:
    Servo1 = Z
    Wait 1
    Z = Z + 1
      If Z >= 18 Then
        Servo1 = X
        Loop4:
        If P0 = 132 Then
        Goto Loop4
        Else
        Goto Loop2
        End If
      End If
    Goto Loop3
  End If

End Sub
Sub Scan2
  If P0 = 136 Then
    Z = 4
    Loop5:
    Servo2 = Z
    Wait 1
    Z = Z + 1
      If Z >= 18 Then
```

```
        Servo2 = Y
        Loop6:
        If P0 = 136 Then
        Goto Loop6
        Else
        Goto Loop2
        End If
      End If
    Goto Loop5
  End If
End Sub
End
```