สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

ควบคุมกล้องผ่านอินเตอร์เน็ต

Camera Control via Internet

โดย

นาย ธนะพงษ์ จุลธรรมาศน์

นาย นพดล วงศ์บุญนาค

เลขหมู่.....
เลขทะเบียน 42277
วัน, เดือน, ปี 16 พ.ค. 2545

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ควบคุมกล้องผ่านอินเตอร์เน็ต

Camera Control via Internet

โดย

นาย ธนะพงษ์ จุลธรรมาศน์

นาย นพดล วงศ์บุญนาค

อาจารย์ที่ปรึกษา

รศ. อิทธิชัย อรุณศรีแสงไชย

ปริญญานิพนธ์สำหรับปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ควบคุมกล้องผ่านอินเตอร์เน็ต

Camera Control via Internet

นาย ธนะพงษ์ จุลธรรมาศน์ รหัสประจำตัว 40010295

นาย นพดล วงศ์บุญนาค รหัสประจำตัว 40010356

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

ลงชื่อ ..... อาจารย์ที่ปรึกษา

(รศ. อิทธิชัย อรุณศรีแสงไชย)

วันที่ 20/ธ.ค./49

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ ปีการศึกษา 2543

ภาควิชา อิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ควบคุมกล้องผ่านอินเตอร์เน็ต

ผู้จัดทำ

1. นาย ธนะพงษ์ จุลธรรมาศน์ รหัสประจำตัว 40010295
2. นาย นพดล วงศ์บุญนาค รหัสประจำตัว 40010356

.....อาจารย์ที่ปรึกษา

(รศ. อิทธิชัย อรุณศรีแสงไชย)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ควบคุมกล้องผ่านอินเตอร์เน็ต

นาย ธนะพงษ์ จุลธรรมาศน์
นาย นพดล วงศ์บุญนาค
รศ. อิทธิชัย อรุณศรีแสงไชย อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

ในโครงงานนี้เป็นการศึกษาถึงวิธีการทำงานในการติดต่อระหว่างเครื่องServerและเครื่อง Client รวมทั้งวิธีการในการรับและส่งภาพ โดยให้มีการทำการรับและส่ง code กันระหว่างเครื่อง Server และเครื่อง client เพื่อนำไปใช้ในการควบคุมทิศทางการหมุนตลอดจนสามารถแสดงองศาการหมุนของกล้องได้ สำหรับในส่วนของการรับและส่งภาพก็ได้แสดงถึงวิธีการทำงานในส่วนของการรับและส่งภาพของกล้องจากการควบคุมของเครื่องServerและเครื่องClientให้สามารถรับและส่งภาพได้อย่างรวดเร็ว รวมทั้งยังแสดงให้เห็นถึงการนำโครงงานนี้ไปประยุกต์ใช้ในชีวิตประจำวันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Camera Control via Internet

Mr. Thanapong Jullathammas

Mr. Noppadon Wongbunnak

Association Prof. Ittichai Arungsrisangchai

(Advisor)

2nd Semester , Education Year 2000

Abstract

This Project is shown the studies of communication between server and client computers, include the ways to send and receive the images. The code is sent and received between server and client computer in order to control the rotation of camera, and to show the rotated angles of camera. In addition, for parts of sending and receiving of the images can show the ways to send and receive the images from server and client computer as quickly. Moreover, is also shown the application of this project in the future.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

เรื่อง **หน้า**

บทคัดย่อ

Abstract

สารบัญข้อมูล

สารบัญรูปภาพ

สารบัญข้อมูล

บทที่ 1 บทนำ 1

บทที่ 2 เครือข่ายอินเตอร์เน็ต 3

2.1 สถาปัตยกรรมอินเตอร์เน็ต (Internet Architecture) 3

2.2 OSI โมเดล 4

2.3 โครงสร้างชั้นเน็ตเวิร์ก (Network Layer Structure) 6

2.4 รูปแบบมาตรฐานโปรโตคอลของอินเตอร์เน็ต 7

2.5 Internet IP 8

2.5.1 Address Structure 8

2.5.2 รูปแบบของข้อมูล 9

2.5.3 การแบ่งส่วนย่อยของข้อมูลและการประกอบขึ้นใหม่ (Fragmentation and Reassembly) 10

2.5.4 การเลือกเส้นทาง (Routing) 12

2.6 ซ็อกเก็ต (Socket) 13

2.6.1 เอพีไอ (API) ข้อกำหนดในการติดต่อสื่อสาร 13

2.6.2 ชนิดของ Socket 15

2.6.3 Socket Address 16

2.6.4 Socket System Call 17

บทที่ 3 พื้นฐานไมโครคอนโทรลเลอร์ และ RS-232 30

3.1 ไมโครคอนโทรลเลอร์ 30

3.2 การสื่อสารพอร์ตอนุกรม RS-232 33

3.2.1 มาตรฐานอนุกรม RS-232 34

3.2.2 การอินเตอร์เฟซตามมาตรฐานอนุกรม RS-232 34

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4 หลักการออกแบบและการสร้าง 35
4.1 ชุดควบคุมกล้อง 35
4.2 โปรแกรมการติดต่อใช้งานระหว่างผู้ควบคุมและผู้ให้บริการ 37
4.2.1 โปรแกรมควบคุมด้านผู้ใช้ 37
4.2.2 โปรแกรมด้านให้บริการ 39
4.2.3 ส่วนส่งภาพ 40
4.2.4 ส่วนรับภาพ 41
บทที่ 5 สรุปผล 43
5.1 ชุดควบคุมกล้อง 43
5.2 ส่วนการติดต่อระหว่าง Server และ Client 43
5.2.1 การทำงานที่เครื่อง Server 43
5.2.2 การทำงานที่เครื่อง Client 43
5.2.3 การรับภาพ 43
ภาคผนวก
กิตติกรรมประกาศ
หนังสืออ้างอิง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญรูปภาพ

	หน้า
รูปที่ 1.1 แสดงการเชื่อมต่ออุปกรณ์ควบคุมผ่านเครือข่ายอินเตอร์เน็ต	2
รูปที่ 2.1 แสดงสถาปัตยกรรมของ Internet	3
รูปที่ 2.2 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI Model	4
รูปที่ 2.3 Network Layer Structure	6
รูปที่ 2.4 Internetwide IP Schemetic	7
รูปที่ 2.5 โครงสร้างของ Address ที่ใช้งานใน class ต่างๆ ของเครือข่ายทั้งหมดความยาว 32 บิต	8
รูปที่ 2.6 Internet datagram format and contents	9
รูปที่ 2.7 Internet fragmentation and reassembly	11
รูปที่ 2.8 รูปแบบทั่วไปของการเลือกเส้นทางภายใน Host	13
รูปที่ 2.9 แสดงการเปรียบเทียบระหว่าง Socket และ SUN RPC application programming	14
รูปที่ 2.10 แสดงลักษณะทั้งสามแบบของ Socket	16
รูปที่ 2.11 แสดงโครงสร้างของข้อมูลที่ใช้ในการกำหนด Socket Address	17
รูปที่ 2.12 แสดงวิธีการแบบ connectionless application protocol โดยใช้ UDP	21
รูปที่ 2.13 แสดงวิธีการแบบ connection-oriented application protocol โดยใช้ TCP	23
รูปที่ 3.1 แสดงสถาปัตยกรรมของ 89C2501	30
รูปที่ 3.2 แสดงการต่อ Oscillator	33
รูปที่ 4.1 แสดงชุดควบคุมกล้อง (1)	36
รูปที่ 4.2 แสดงชุดควบคุมกล้อง (2)	37
รูปที่ 4.3 แสดงการวางรูปแบบของโปรแกรมด้าน Client	38
รูปที่ 4.4 แสดงโปรแกรมฝั่ง Server	40
รูปที่ 4.5 แสดงโปรแกรมจับภาพ	41
รูปที่ 4.6 แสดงโปรแกรม index.htm	42

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

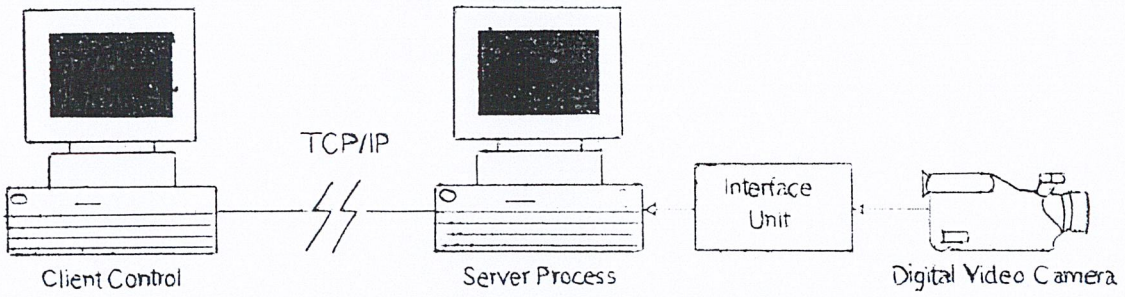
บทที่ 1
บทนำ

เนื่องด้วยในโลกยุคปัจจุบันนี้มีสิ่งจำเป็นที่มนุษย์จะต้องเรียนรู้และเข้าใจอยู่ 2 อย่างคือ internet และ biotechnology ซึ่งในโครงงานนี้จะเน้นในเรื่องของอินเตอร์เน็ต เพราะอินเตอร์เน็ตได้เข้ามามีบทบาทในชีวิตประจำวันของมนุษย์มากขึ้น ดังนั้นการทำให้เครื่องมือหรืออุปกรณ์ใดๆที่ใช้ประโยชน์จากการติดต่อผ่านอินเตอร์เน็ต ซึ่งจะได้รับประโยชน์อย่างมากมายที่เห็นได้อย่างชัดเจนก็คงเป็นความครอบคลุมการใช้งานและการควบคุมระยะไกล (Remote Control) ซึ่งเครือข่ายอินเตอร์เน็ตนั้นได้ครอบคลุมไปทุกมุมโลกแล้วในขณะนี้และจำนวนผู้ใช้งานก็มีอัตราเพิ่มขึ้นอย่างรวดเร็ว เราจึงใช้คุณสมบัติตรงนี้ของอินเตอร์เน็ตเข้ามาประยุกต์ใช้ในชีวิตประจำวันได้

ในงานที่ต้องการความสะดวกในการติดต่อหรือควบคุมระยะไกล (Remote Control)ไม่มีการสื่อสารชนิดใดที่จะทำได้ง่ายและรวดเร็วกว่าการติดต่อโดยสื่ออื่นๆ ซึ่งเราไม่จำเป็นต้องอยู่ในสถานที่อันตรายเนื่องจากความไม่ปลอดภัยจากสถานที่ทำงานหรือความไม่สะดวกจากการที่ต้องเดินทางไปปฏิบัติงาน การควบคุมสามารถที่จะทำได้ในสถานที่ที่เหมาะสมได้การควบคุมตรวจสอบหรือดูแลก็ทำได้ง่ายขึ้นเพียงแต่เพิ่มในส่วนของอินเตอร์เน็ตเข้ามาทำงานร่วมกับชุดควบคุมหรือที่เรียกกันว่า Software อีก เพียงเท่านี้เครื่องมือเหล่านี้ก็สามารถที่จะสามารถควบคุมการทำงานผ่านเครือข่ายอินเตอร์เน็ต การควบคุมสามารถที่จะทำได้ทุกมุมโลกที่อินเตอร์เน็ตเข้าถึงซึ่งมีความสะดวกมาก ยกตัวอย่างเช่น บริษัทรักษาความปลอดภัยแห่งหนึ่งได้รับมอบหมายในการจัดเวรยามคอยเฝ้าดูสถานที่ต่างๆ ตลอด 24 ชั่วโมง ทั้งหมด 10 สถานที่ที่ตั้งซึ่งอยู่ไกลกันมาก ดังนั้นบริษัทนี้จึงจำเป็นต้องจ้างเวรยามจำนวนมากที่คอยเฝ้าดูแลสถานที่ทั้งหมด สถานที่ละ 3 คนต่อวัน รวมหนึ่งวันก็ 30 คน และต้องเสียค่าใช้จ่ายจำนวนมากต่อวัน และถ้าบริษัทนี้ได้ใช้กล้องควบคุมผ่านเครือข่ายอินเตอร์เน็ต ติดตั้งไว้ทั้ง 10 ที่ และใช้เพียงยามคนเดียวต่อการเฝ้าดู การเคลื่อนไหวพร้อมๆกันทั้ง 10 ที่ จึงเป็นการลดจำนวนบุคลากรตรงนี้ได้มากเลยทีเดียว

โครงการนี้ได้นำเอาแนวคิดจากตัวอย่างข้างบนมาทำการศึกษาและสร้างเครื่องมือชุดนี้ขึ้นมาเพื่อให้ได้มาซึ่งวัตถุประสงค์ที่ต้องการ โดยหลักการทำงานอย่างคร่าวๆ สามารถ แบ่งเป็นส่วนต่างๆได้ดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



1. ส่วนควบคุม (Client Control) จะประกอบไปด้วยคอมพิวเตอร์ ที่เชื่อมต่อเข้ากับเครือข่ายอินเตอร์เน็ต และได้บรรจุโปรแกรมคุมการทำงานในส่วนของโครงงานนี้เอาไว้

2. ส่วนประมวลผล (Server Process) จะเป็นคอมพิวเตอร์ที่ทำให้เป็น Server ที่เอาไว้สำหรับติดต่อกับกับส่วนของส่วนควบคุมโดยผ่านเครือข่ายอินเตอร์เน็ตและนำผลของการประมวลผลส่งไปยังส่วนควบคุมกล้องซึ่งจะได้กล่าวถึงต่อไป

3. ส่วนควบคุมกล้อง (Camera Control Unit) จะเป็นส่วนที่รับคำสั่งควบคุมการทำงานจากส่วนประมวลผลมาปฏิบัติตามแล้วและจะรายงานผลตำแหน่งของกล้องด้วยว่าขณะนี้อยู่ที่ตำแหน่งใด

4. กล้องรับภาพดิจิตอล (Digital CCD Camera) ทำหน้าที่แปลงภาพให้เป็นสัญญาณดิจิตอลเพื่อส่งไปยังคอมพิวเตอร์

จากการทำงานเบื้องต้นข้างบนนี้ บทต่อๆไปจะกล่าวถึงทฤษฎี ที่ใช้ในการออกแบบและหลักการออกแบบในส่วนต่างๆในบทต่อไป

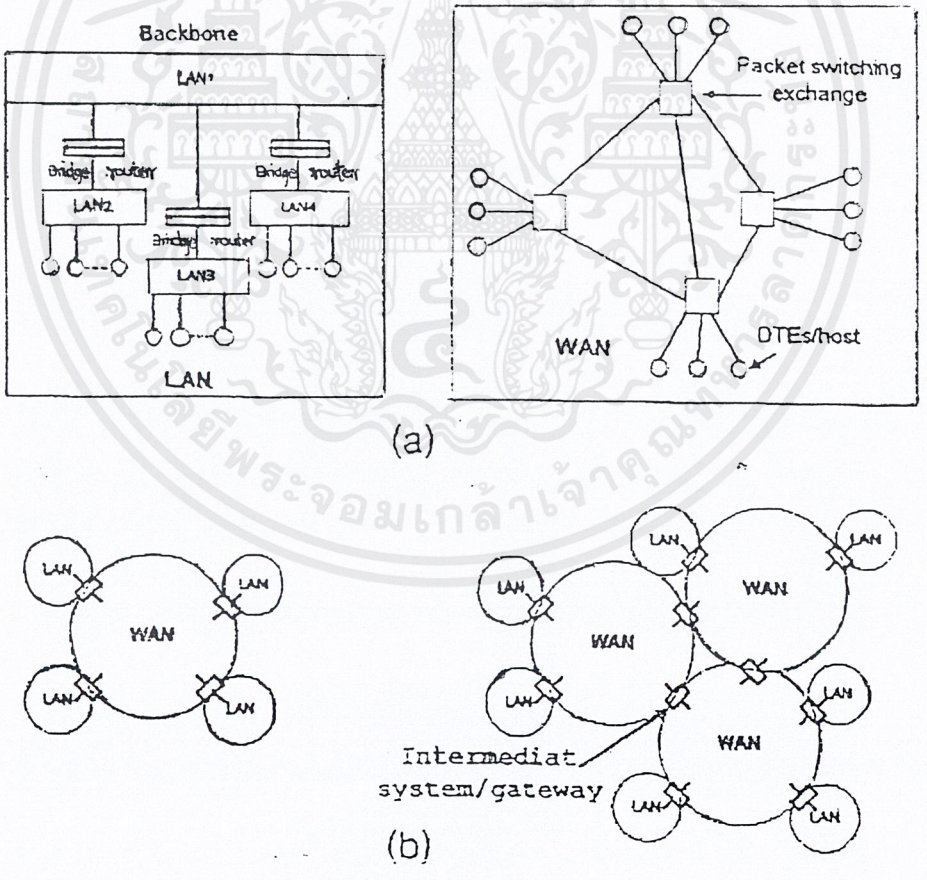
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
เครือข่ายอินเตอร์เน็ต

Internet คือการที่เครือข่าย 2 หรือมากกว่า เชื่อมต่อเข้าด้วยกันและทำงานเสมือนเป็นเครือข่ายเดียวกันโดยNetwork ที่เป็นส่วนประกอบของ Internet คือ Subnetwork หรือ Subnet ซึ่งอาจจะเป็นเครือข่าย Local Area Network หรือ Wide Area Network อุปกรณ์ที่ใช้ในการเชื่อมต่อ 2 เครือข่ายเข้าด้วยกันก็คือ intermediate system หรือ internetworking unit การเชื่อมโยงระหว่างระบบที่แตกต่างกันจำเป็นต้องมีมาตรฐานในการติดต่อกันซึ่งเรียกเป็นศัพท์เฉพาะว่า โปรโตคอล

2.1 สถาปัตยกรรมอินเตอร์เน็ต (Internet Architectures)

สถาปัตยกรรมพื้นฐานของ Internet แสดงดังรูปที่ 2.1



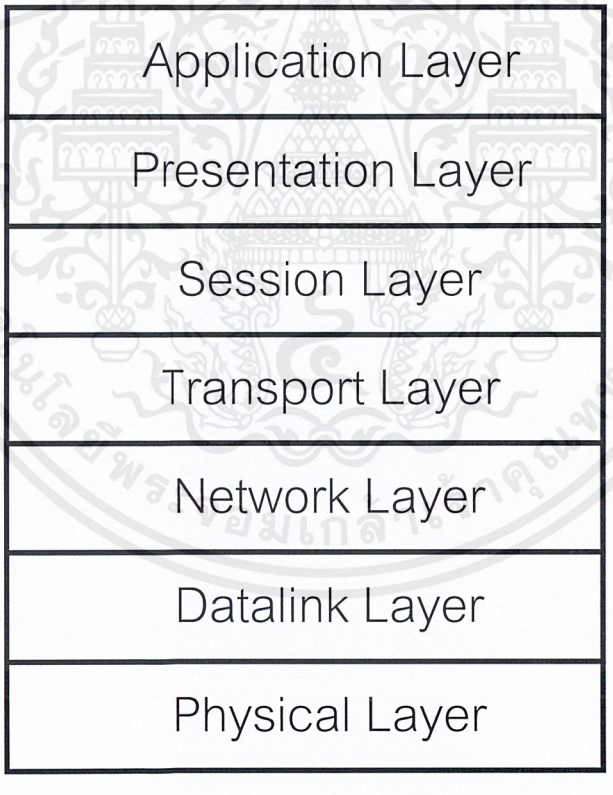
รูปที่ 2.1 แสดงสถาปัตยกรรมของ internet (a) Single LAN and WAN (b) interconnect

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในรูป (a) แสดงตัวอย่าง 2 ตัวอย่างของเครือข่ายเดี่ยว (Single network) ซึ่งอย่างแรกเป็น side-wide LANซึ่งประกอบขึ้นมาจากชุดของ LANS ซึ่งถูกต่อเข้ากับเครือข่ายหลัก ซึ่งอุปกรณ์ที่ใช้ต่อ LAN เข้ากับเครือข่ายหลัก ถ้า LAN ทุกเครือข่ายมีระบบเดียวกันก็จะใช้ bridge ถ้าเป็น LAN ที่แตกต่างกันก็จะใช้ router ตัวอย่างที่ 2 เป็นตัวอย่างของ WAN เดี่ยวๆ ในรูป (b) แสดงถึงเครือข่ายอินเตอร์เน็ตซึ่งประกอบไปด้วย เครือข่ายทั้ง 2 ชนิดข้างต้น

2.2 Operating System Interconnection (OSI) โมเดล

องค์กรมาตรฐานสากลISOได้กำหนดมาตรฐานของเครือข่ายโดยจัดแบ่งกิจกรรมของเครือข่ายออกเป็นงานย่อยๆและกำหนดโมเดลแบ่งเป็นชั้นๆตามลำดับเรียกว่ามาตรฐาน OSIโดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อยๆ ก็จะช่วยในการออกแบบ และการใช้งานเครือข่ายรวมถึง การเชื่อมโยงกันเป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน ดังในรูปที่ 2.2



รูปที่ 2.2 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI model

ในแต่ละชั้นของOSIโมเดลจะมีการติดต่อสื่อสารกันเป็นชั้นๆตามลำดับลงมาเช่น Application Layer ก็จะติดต่อสื่อสารกับ Presentation Layer ตามลำดับไปจนถึงชั้นแรกสุดคือ Physical Layer

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Application Layer เป็นชั้นบนสุดของโมเดล เป็นส่วนที่จะทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้เป็นไปได้ตามต้องการ ตัวอย่าง ของเครือข่ายเช่น ระบบ e-mail, การโอนถ่ายข้อมูล, การขอเข้าใช้ระบบคอมพิวเตอร์ในเครือข่ายเป็นต้น

Presentation Layer มีการกำหนดหน้าที่ไม่ชัดเจนนักและมีการนำไปใช้ไม่มาก ซึ่งหน้าที่หลักก็คือ เป็นส่วนที่จัดรูปแบบและนำเสนอข้อมูล ให้เป็นไปตามต้องการ รวมถึงการจัดแปลงข้อมูล ในรูปมาตรฐาน ASCII หรือ EBCDIC, การลดขนาดข้อมูล, การเข้ารหัส หรือถอดรหัสของข้อมูล แต่ส่วนใหญ่แล้ว แอปพลิเคชั่นจะจัดการแทนได้

Session Layer เป็นชั้นที่จัดการในเรื่อง "การติดต่อแต่ละครั้ง" หรือ session ให้ระบบคอมพิวเตอร์ทั้ง 2 ฝั่ง โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อครั้งนั้นๆ เป็นไปได้โดยไม่มีปัญหา จนถึงเลิกการติดต่อเมื่อเสร็จงาน

Transport Layer ทำหน้าที่ควบคุมปริมาณและรายละเอียด วิธีการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้ายที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูลซึ่งส่วนของ PCT ในโพรโตคอลทำงานในระดับนี้

Network Layer ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูล ระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน Package ข้อมูล ผ่านอุปกรณ์ต่างๆไปยังเครือข่ายย่อย ได้อย่างถูกต้องตามที่ต้องการนอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูลและกั้นหรือกรอง Package ข้อมูลที่ส่งไปยังเครือข่ายเดียวกันไม่ให้ข้ามไปยังเครือข่ายอื่นซึ่งจะช่วยลดปริมาณข้อมูลที่จะวิ่งบนเครือข่ายได้ส่วนหนึ่ง โพรโตคอล IP ,TCP/IP , และ IPX เป็นโพรโตคอลที่ทำงานอยู่ใน layer นี้

Data link layer ทำหน้าที่เรียกใช้หรือกำหนดช่องทาง ในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet, Tokenring , หรือ FDDI เป็นต้น รวมถึงการลำดับและอัตราการรับส่งข้อมูลหรือ flow control และสถานที่ที่จะส่งข้อมูลไป ทั้งนี้ Data link layer จะเป็นชั้นแรกที่จัดการแปลงข้อมูลจาก bit ให้เป็น package โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้อง ในกรณีที่ส่งข้อมูลออกไป หรือในกรณีที่อ่านข้อมูลที่เข้ามา ก็จะตรวจสอบผ่าน checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วนและถ้าได้รับ package ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งานและจะบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

Physical Layer รับผิดชอบดูแลในรายละเอียด ในการส่งข้อมูลในด้าน hardware เช่น การควบคุม network Interface Card การส่งสัญญาณผ่านสานสัญญาณแบบต่างๆ การเชื่อมต่อเข้ากับเครือข่ายแบบต่างๆ โดยใช้ Physical Layer จะจัดสร้างสัญญาณทางไฟฟ้า, สัญญาณเสียง หรือ สัญญาณที่จะเป็นในการสื่อสารโดยตรง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เนื่องจาก Network Layer เป็นชั้นที่โพรโตคอล IP, TCP/IP ทำงานอยู่จะกล่าวในรายละเอียดต่อไป

2.3โครงสร้างชั้น เน็ตเวิร์คหน้าที่ของ Network Layer ในแต่ละ End System จะเป็นตัวจัดการการติดต่อแบบ end-to-end ของการบริการ internet wide ไปยังผู้ใช้บริการ

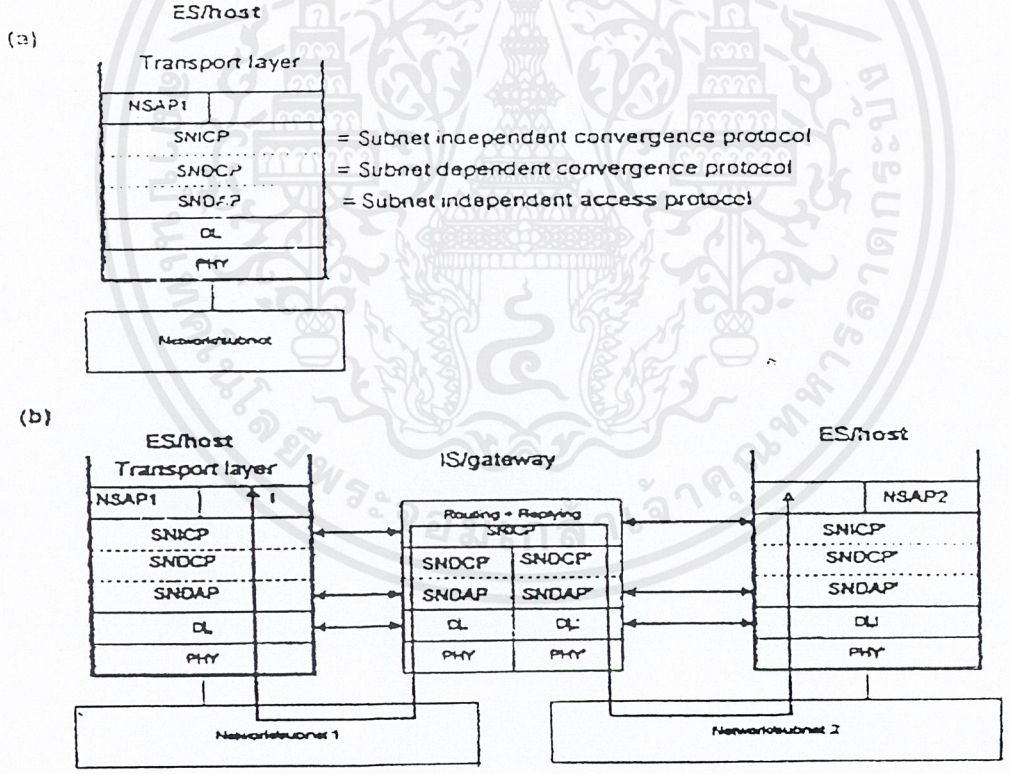
โดย ISO ได้จัด network layer เป็น 3 โพรโตคอล ซึ่งจะทำงานร่วมกันเพื่อให้ บริการใน network layer ได้แก่

Subnetwork independent convergence protocal (SNICP)

Subnetwork dependent convergence protocal (SNDCP)

Subnetwork dependent access protocal (SNDAP)

โดยที่โครงสร้างแสดงดังรูปที่ 2.3



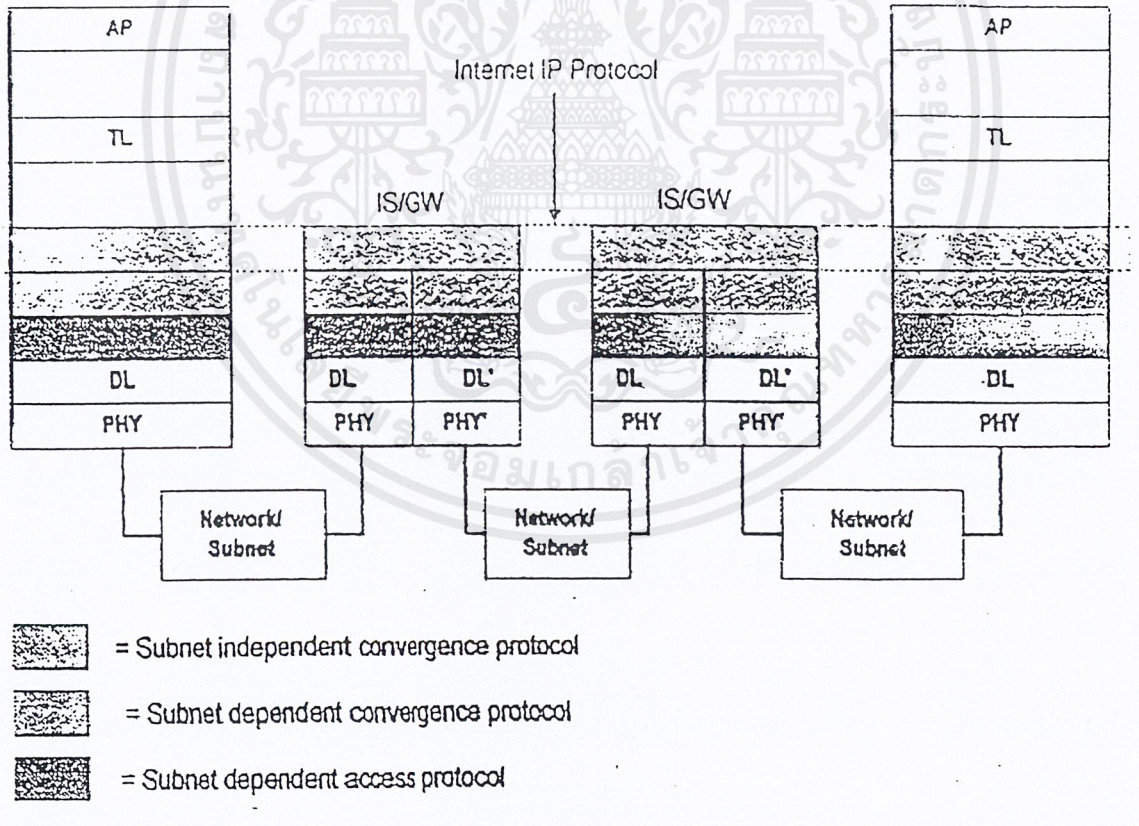
รูปที่ 2.3 Network layer structure (a) Sublayer Protocal (b) IS structure

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยที่ SNICP จะเป็นตัวสนับสนุนจัดการให้ผู้ใช้บริการ สามารถ Interface กับ Internet ซึ่งมันจะมีหน้าที่เป็นตัวประสานฟังก์ชั่นที่จำเป็นในการเลือกเส้นทางและถ่ายทอดข้อมูลของผู้ใช้ข้าม Internetซึ่งการทำงานของมันไม่ขึ้นอยู่กับคุณสมบัติเฉพาะของเครือข่ายย่อย SNDAP จะเป็นตัวโพรโตคอลที่ติดต่อกับ เครือข่ายย่อย ที่มีลักษณะเฉพาะใน internet เช่น X.25 package layer protocal สำหรับเครือข่าย X.25 ซึ่งใช้บ่อย ใน LAN เพราะว่าการบริการและการทำงานของ SNDAP แตกต่างจาก NETWORK แบบอื่นๆ sublayer ที่อยู่ตรงกลางคือ SNDCP จะเป็นตัวจัดการระหว่าง SNICP และ SNDAP

2.4 รูปแบบมาตรฐานโปรโตคอลของอินเตอร์เน็ต

อินเตอร์เน็ตโพรโตคอลซึ่งถูกใช้ในอินเตอร์เน็ต มีโพรโตคอลก็คือ TCP/IP ซึ่งรวมถึง Transport และ application โพรโตคอล ซึ่งทั้งหมดของ TCP/IP จะกำหนดให้เหมาะกับการใช้ในการเชิงสาธารณะ ซึ่งรูปแบบโดยทั่วไปแสดงดังรูปที่ 2.4



รูปที่ 2.4 Internetwide IP schemetic

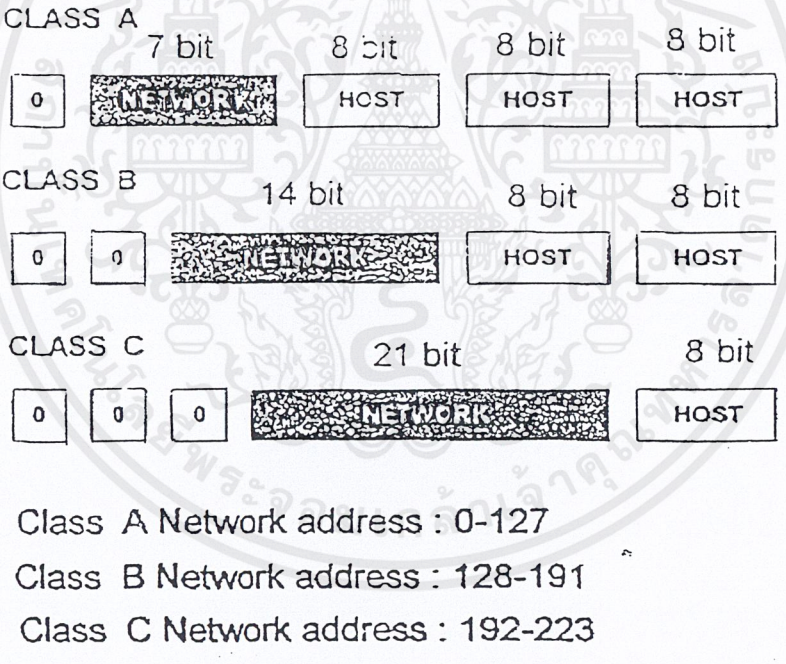
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

IPเป็น internet wide protocal ซึ่งทำให้สอง transport protocal ที่ต่างสถานที่และต่าง ES/hosts กันสามารถแลกเปลี่ยนหน่วยข้อมูลกันได้ ซึ่งหมายถึงว่า หลายๆ network/subnet และ IS/gateways ที่แตกต่างกันสามารถติดต่อสื่อสารกันได้อย่างสมบูรณ์

2.5 Internet IP

2.5.1Address structure

ในศัพท์ของ ISO เมื่อ 2 เครือข่าย ติดต่อกันด้วย host/ES ที่ต่อกับอินเตอร์เน็ต เครือข่ายเหล่านี้ติดต่อกันได้ โดยใช้ network service access point address และ subset point of attachment สำหรับใน TCP/IP ก็จะมี IP address และ NPA address ตามลำดับ โดย NPA address จะแตกต่างกันในแต่ละชนิดของ network/subset ขณะที่ IP address จะเป็นรูปแบบเดียวกัน โครงสร้างของ IP address แสดงดังรูปแบบที่ 2.5



รูปที่ 2.5 โครงสร้างของ Address ที่ใช้ใน class ต่างๆ ของเครือข่ายทั้งหมดความยาว 32 บิต

IP address นี้ มีการจัดแบ่งออกเป็นทั้งหมด 5 ระดับ แต่ที่ใช้งานทั่วไปจะมีเพียง 3 ระดับ คือ ระดับ A, ระดับ B และระดับ C ซึ่งจะแบ่งตามขนาดความใหญ่ของเครือข่าย ถ้าเครือข่ายใดมีจำนวนเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่มาก จะมีหมายเลขอยู่ในระดับA และลดหลั่นกันมาในระดับB และระดับC ตามลำดับ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

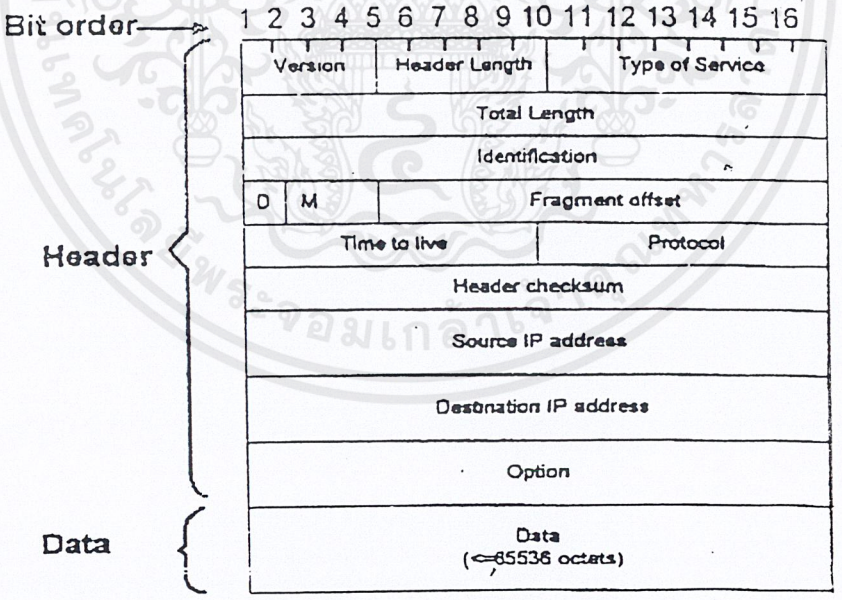
จากรูปจะเห็นว่าหมายเลข IP ของระดับ A มีตัวแรกเป็นศูนย์และหมายเลขของเครือข่ายขนาด 7 bit และมีเลขหมายเครื่องคอมพิวเตอร์ ขนาด 24 bit ทำให้ในหนึ่งเครือข่ายของระดับ A สามารถมีคอมพิวเตอร์เชื่อมต่ออยู่ ในเครือข่ายได้ถึง 16 ล้านเครื่อง แต่ในระดับ A นี้ จะมีหมายเลขเครือข่ายได้ 128 ตัวเท่านั้น ทั่วโลก ซึ่งก็คือจะมีเครือข่ายใหญ่แบบนี้เพียง 128 เครือข่ายเท่านั้น

สำหรับระดับ B จะมีหมายเลขเครือข่ายแบบ 14 bit และหมายเลขเครื่องคอมพิวเตอร์แบบ 16 bit ดังนั้นจึงสามารถมีคอมพิวเตอร์เชื่อมต่อในเครือข่ายระดับ B แต่ละเครือข่ายได้ถึง กว่า 65,000เครื่อง และสุดท้ายคือระดับ C ซึ่งมีหมายเลขคอมพิวเตอร์แบบ 8 bit และมีหมายเลขเครือข่ายแบบ 21 bit ส่วน 3 bit แรกบังคับว่าต้องเป็น 110 ดังนั้นในแต่ละเครือข่าย ระดับ C จะมีจำนวนเครื่องต่อเชื่อมได้เพียงไม่เกิน 254 เครื่องในแต่ละเครือข่าย

จะเห็นได้ว่าเมื่อเครือข่ายและเครื่องคอมพิวเตอร์ที่ต่ออยู่ในอินเตอร์เน็ตมีหมายเลข IP address ให้ใช้อ้างอิงได้ไม่ซ้ำกันและมีความหมายให้ทราบถึงขนาดเครือข่ายแล้วการติดต่อส่งผ่านข้อมูลจึงกระทำได้ไม่สับสน

2.5.2รูปแบบของข้อมูล

รูปแบบของ IP data unit ก็คือ datagram ซึ่งโครงสร้างของ datagram เป็นดังรูปที่ 2.6



รูปที่ 2.6 Internet datagrams format and contents

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

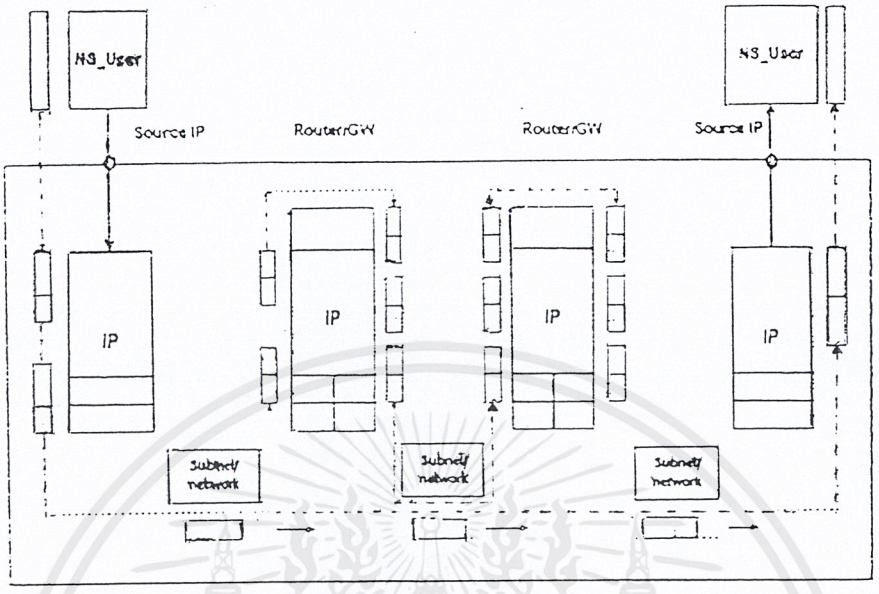
หน่วยข้อมูล IP แต่ละหน่วยจะประกอบด้วย ส่วนของข้อมูลที่รับมาจากส่วนของงาน TCP หรือ UDP และส่วนของข้อมูลนำทาง ซึ่งมีรายละเอียดดังนี้

- Version หมายเลขรุ่นของข้อกำหนด IP
- Header Length ความยาวของข้อมูลนำทาง
- Type of service วิธีการจัดการกับข้อมูล
- Total Length ความยาวของหน่วยข้อมูล
- Identification, Flags และ Fragment offset รายละเอียดที่เกี่ยวกับการแบ่งย่อยข้อมูลซึ่งจะถูกนำมาใช้ในการรวบรวมข้อมูล
- Time to live เวลาสูงสุดที่ใช้ในการเดินทาง ซึ่งกำหนดมาจากต้นทาง เวลานี้จะลดลงเรื่อยๆ ในระหว่างทาง ถ้าลดลงไปถึงศูนย์ หน่วยข้อมูลนั้นจะถูกกำจัดไป
- Protocal ชนิดของข้อมูลเป็น UDP หรือ TCP
- Header Checksum ค่าตรวจสอบข้อมูลนำทาง
- IP address หมายเลข internetwide IP ของเครื่องต้นทางและปลายทาง
- Option ข้อมูลอื่นๆเช่น ข้อมูลเกี่ยวกับการรักษาความปลอดภัย บันทึกเส้นทางเดินของข้อมูล และเวลาที่ข้อมูลเดินทางมาถึงเป็นต้น

2.5.3 การแบ่งส่วนย่อยของข้อมูลและการประกอบขึ้นใหม่

ขนาดข้อมูลของผู้ใช้ซึ่งอ้างอิงกับ NSDU มีความจุได้ถึง 64k หรือ 65,536 bytes แต่ขนาดของหน่วยข้อมูลที่สามารถใช้ติดต่อกัน ในระบบที่ต่างกันสามารถมีได้ตั้งแต่ 128 bytes สำหรับระบบ X.25 packat switching จนถึง 8,000 bytes สำหรับบาง LAN ดังนั้น กระบวนการ Fragmentation และ Reassembly จึงถูกนำมาใช้เพื่อทำให้ขนาดของข้อมูลเล็กลง และสามารถส่งไปในระบบได้ และเมื่อถึงปลายทาง IP ก็จะทำการประกอบข้อมูลขึ้นมาใหม่ ก่อนที่จะส่งผ่านไปยังผู้ใช้ปลายทาง ดังรูปที่ 2.7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.7 internet fragmentation and reassembly

อันดับแรก IP ใน host ต้นทางจะแยกข้อมูลของผู้ใช้, NSDU เป็น Datagram ซึ่งมี Address กำกับเป็นเฉพาะส่วนๆไป ซึ่งจะถูกออกคำสั่งโดย เครือข่ายที่มันติดต่ออยู่ด้วยและส่ง Datagram ไปยัง IP ใน Gateway ตัวแรก โดยที่ IP ใน Gateway จะไม่ Reassemble NSUD แต่จะปรับปรุงในขอบเขตที่เหมาะสม และส่ง Datagram ที่ได้รับตรงไปยังเครือข่ายที่ 2 หรือทำการFragment datagram ให้มีขนาดเล็กลง ซึ่งขั้นตอนนี้จะถูกทำซ้ำที่ Gateway ตัวต่อไป โดยในรูปที่ 7 Network หลังสุดสามารถรองรับขนาดของ Packet ได้มากกว่า Packet ที่มันได้รับข้อมูลจึงถูกส่งได้โดยตรง โดยที่จะมีการปรับปรุงในบางส่วน Header ของ Datagram เท่านั้น จากนั้น IP ใน host ปลายทางจะทำการประกอบข้อมูลที่มันได้รับขึ้นมาใหม่ และส่งผลที่ได้รับก็คือ NSDU ไปยังผู้ใช้

ในการคิดค่าเวลาสูงสุดที่ Host ต้นทางกำหนดให้ Gateway รอ Datagram ระหว่างแต่ละการ Assembly ซึ่งก็คือ time to live ซึ่งจะถูกนำติดไปที่ส่วนของ Header ของ Datagram ซึ่งจะถูกตั้งค่าโดย IP ใน Host ต้นทาง ซึ่งจะมีค่าลดลงเรื่อยๆในแต่ละขั้นตอน ของการ Process Datagram ถ้า Datagram ถูก Fragment ค่าปัจจุบันจะถูกนำไปใส่ในส่วน Header ของ Datagram ตัวใหม่ ถ้ามันถึงค่า 0 ที่จุดใดๆ ระหว่าง Process ใน Gateway การ Reassembly ก็จะล้มเหลวและทุกๆการ Fragment ที่เกี่ยวกับ NSDU ก็จะถูกตัดทิ้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่า time to live ในแต่ละ Datagram จะเป็นจำนวนเท่าของ 1 วินาที โดยที่จำนวนของมันจะถูกลดลงในแต่ละ IP ซึ่งจะเปลี่ยนแปลงไปตามค่าเวลาจริงในการส่งถ่ายข้อมูลของเครือข่ายที่ติดต่อด้วย

2.5.4 การเลือกเส้นทาง (Routing)

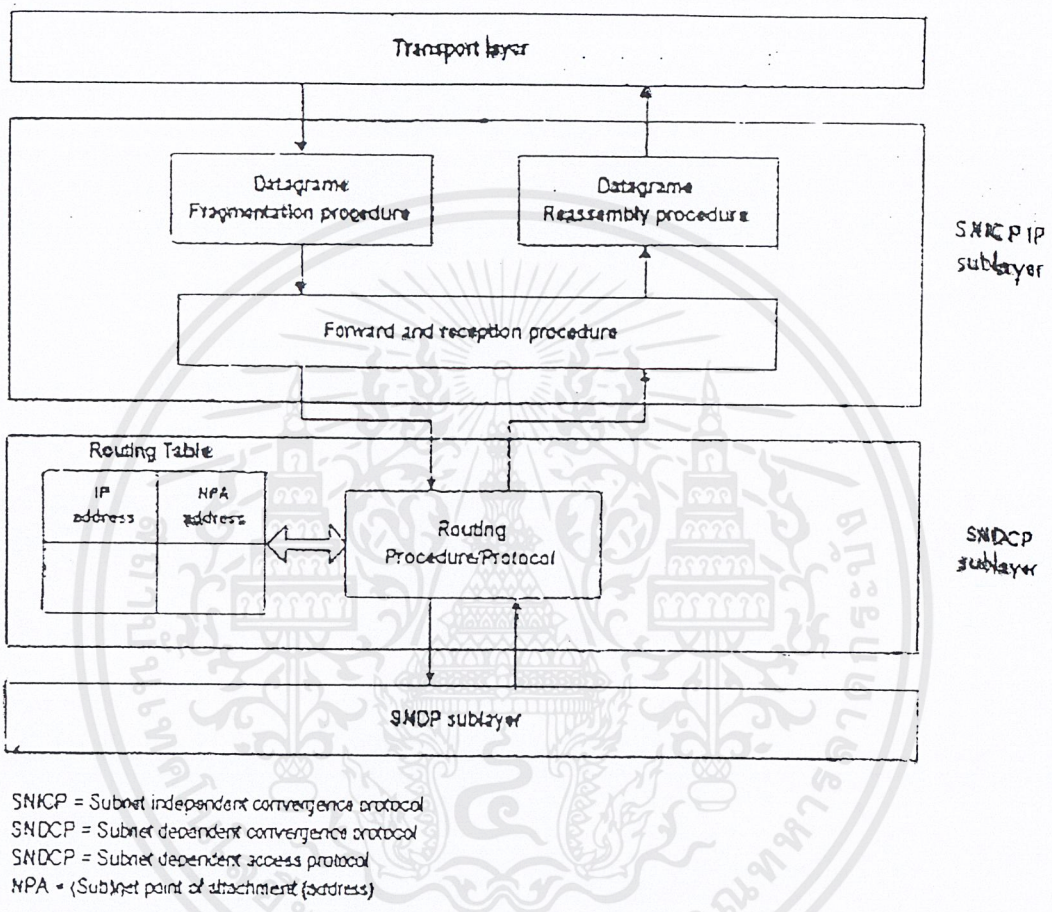
ในแต่ละ Network ใน Internetจะมีชนิดของ PA Address ที่แตกต่างกัน ซึ่งระบบ host หรือ gateway ที่ถูกต่อเข้ากับ network จะสามารถส่ง Datagramไปยังระบบอื่นได้โดยตรงเฉพาะ network ที่เหมือนกันเท่านั้นในการเลือกเส้นทางให้ datagram ข้ามไปยังหลายๆ network IP ในแต่ละ internetwork gateway ต้องรู้ PA addresss ของ host ปลายทาง

ซึ่งมี 2 วิธีการพื้นฐานที่ถูกใช้ในการหาเส้นทางภายในinternetคือ centralized และ distirduted ด้วยวิธีการ centralized routing ข้อมูลเกี่ยวกับการเลือกเส้นทาง ที่เกี่ยวข้องกับแต่ละ gateway จะถูก download จาก site ส่วนกลาง โดยใช้ข้อมูล network และ special network management โดย network management จะพยายามตรวจสอบ network และ host ที่ถูกเพิ่มเข้าและถอดออก และข้อบกพร่องที่จะถูกวินิจฉัยและตรวจซ่อม

ด้วยวิธีการ Distributed routing ทุกๆ host และ gateway จะร่วมกันในการแบ่งปันวิธีการในการรับประกันว่า ข้อมูลเกี่ยวกับการเลือกเส้นทางในแต่ละ system, host และ gateway จะถูกทำให้ทันสมัยและสอดคล้องกัน ข้อมูลเกี่ยวกับการเลือกเส้นทางจะถูกจดจำไว้โดยแต่ละระบบ ในรูปของ routing table ซึ่งจะมี NPA address ไว้ใช้ในการส่งแต่ละ Datagram ซึ่ง internet จะใช้วิธีการแบบนี้

ขั้นตอนการ routing ที่เกี่ยวกับ IR ขั้นตอนแรกจะอ่าน IP address ปลายทางจากภายใน Datagram และใช้มันในการหาการตอบสนอง PA address ของ host หรือ gateway จาก routing table ในส่วนที่เพิ่มเติมชุดของ routing protocal จะถูกใช้เพิ่มและรักษาส่วนที่อยู่ในแต่ละ routing table ในแบบของ distributed ซึ่งรูปแบบทั่วไปที่ถูกใช้ภายใน host IP แสดงดังรูปที่ 2.8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.8 รูปแบบทั่วไปของการเลือกเส้นทางภายใน Host

2.6 ซ็อกเก็ต (Socket)

2.6.1 เอพีไอ (API) ข้อกำหนดในการติดต่อสื่อสาร

TCP/IPเป็นsoftwareในการเชื่อมโยงเครือข่ายชนิดหนึ่งที่สนับสนุนโปรแกรมประยุกต์ในการอินเตอร์เฟสสำหรับการติดต่อสื่อสารบนอินเตอร์เน็ตซึ่งจะอธิบายชนิด ของ APIs ในที่นี้เพียง 2 ชนิด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Socket System Calls

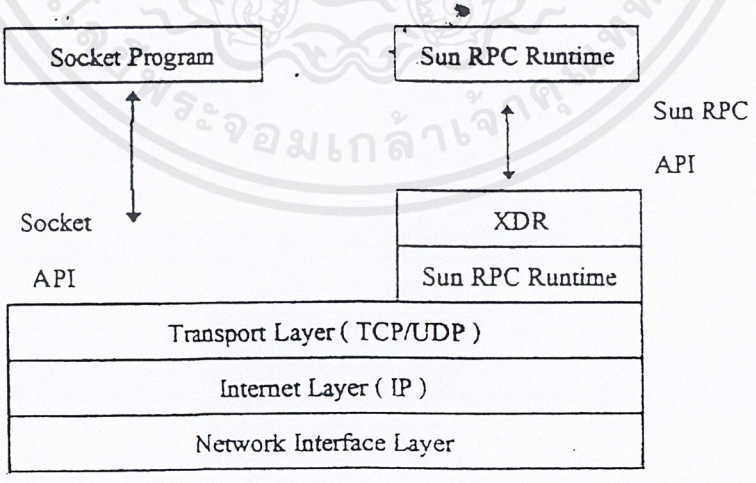
ซ็อกเก็ตเข้าถึงข้อกำหนดในการติดต่อสื่อสารของเครือข่าย ซึ่งโดยทั่วๆไปจะเกี่ยวข้องกับการเปลี่ยนแปลง BSD ของ UNIX ซึ่งเป็นอุปกรณ์อันดับแรกของซ็อกเก็ต API การอินเตอร์เฟสของซ็อกเก็ตได้จัด API สำหรับการติดต่อสื่อสารของโครงข่าย ซึ่งเป็น API ที่ใกล้เคียงกับการทำงาน ของ I/O มากๆ

การอินเตอร์เฟสของซ็อกเก็ตเป็นเครื่องมือที่ใช้งานอย่างกว้างขวางสำหรับระบบปฏิบัติการ UNIX และระบบปฏิบัติการอื่นๆ โปรแกรมที่เข้าถึงซ็อกเก็ตสามารถที่จะใช้งานเป็นเครื่องมือสำหรับการติดต่อสื่อสารบนโครงข่ายอินเตอร์เน็ต โดยอยู่ในสภาพแวดล้อมที่แตกต่างกันได้และติดต่อสื่อสารได้กับโฮสทุกชนิด ซ็อกเก็ต API โดยทั่วไปมักใช้สำหรับโครงข่ายที่ใช้งานบน TCP/IP และก็สามารถใช้งานได้ดีกับโครงข่ายแบบอื่นๆ

Remote Procedure Call Facilities

Remote Procedure Call (RPC) เป็นชนิดหนึ่งของ API ซึ่งยอมให้ผู้เขียนโปรแกรมสำหรับโครงข่ายใช้ procedure call ที่มีโครงสร้างอย่างง่ายๆ เพื่อที่จะควบคุมกันระหว่างโพรซิเยอร์ของโฮสที่มีการทำงานที่แตกต่างกันบนอินเตอร์เน็ต สิ่งอำนวยความสะดวกของ RPC ที่ใช้กันอย่าง กว้างขวางบนสภาวะ TCP/IP ได้แก่

- SUN RPC
- Hewlett-Packard RPC



รูปที่ 2.9 แสดงการเปรียบเทียบระหว่าง ซ็อกเก็ต และ SUN RPC application programming

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซ็อกเก็ตอินเตอร์เฟสเป็นการเริ่มพัฒนาสำหรับหลายๆ BSD ของระบบปฏิบัติการ UNIX โดยการใช้ซ็อกเก็ตอินเตอร์เฟส จะมี 2 โปรแกรมประยุกต์ คือ

1. การทำงานที่ Local System

2.การทำงานที่ Remote System

ซึ่งสามารถติดต่อสื่อสารกับโฮสอื่นๆได้ด้วยวิธีมาตรฐาน ซ็อกเก็ตอินเตอร์เฟสที่ใช้กันอย่างกว้างขวาง คือ API ที่ใช้งานบน TCP/IP

ซ็อกเก็ต API เป็นอุปกรณ์ที่ใช้สำหรับการเชื่อมต่อระหว่างผู้ขอใช้บริการและผู้ให้บริการระหว่าง 2 ส่วนประกอบประยุกต์ที่ทำงานบนระบบของคอมพิวเตอร์ที่แตกต่างกัน

การติดต่อระหว่างผู้ขอใช้บริการและผู้ให้บริการแต่ละครั้งจะเรียกฟังก์ชั่นขึ้นมาจัดการการเชื่อมต่อระหว่างผู้ขอใช้บริการและผู้ให้บริการนั้นๆ การเชื่อมต่อระหว่าง 2 ส่วนประกอบ (component) ประยุกต์นี้เป็นพื้นฐานบนการใช้โครงสร้างข้อมูลที่ถูกเรียกว่า ซ็อกเก็ต เพื่อเป็นการบริการการติดต่อสื่อสารโดยใช้ TCP/IP

ส่วนประกอบของผู้ขอใช้บริการและผู้ให้บริการ (Client-Server component) จะเรียกฟังก์ชั่นที่ใช้ส่งและรับข้อมูลบนอินเตอร์เน็ต ซึ่งคล้ายกับวิธีการการเรียกฟังก์ชั่นที่ใช้กับไฟล์ I/O ทั่วๆไป

ซ็อกเก็ต API สามารถที่จะใช้งานได้ในโปรโตคอลอื่นๆที่นอกเหนือจาก TCP/IP แต่ในที่นี้จะอธิบายเฉพาะโปรโตคอล TCP/IP เท่านั้น

2.6.2 ชนิดของ Socket

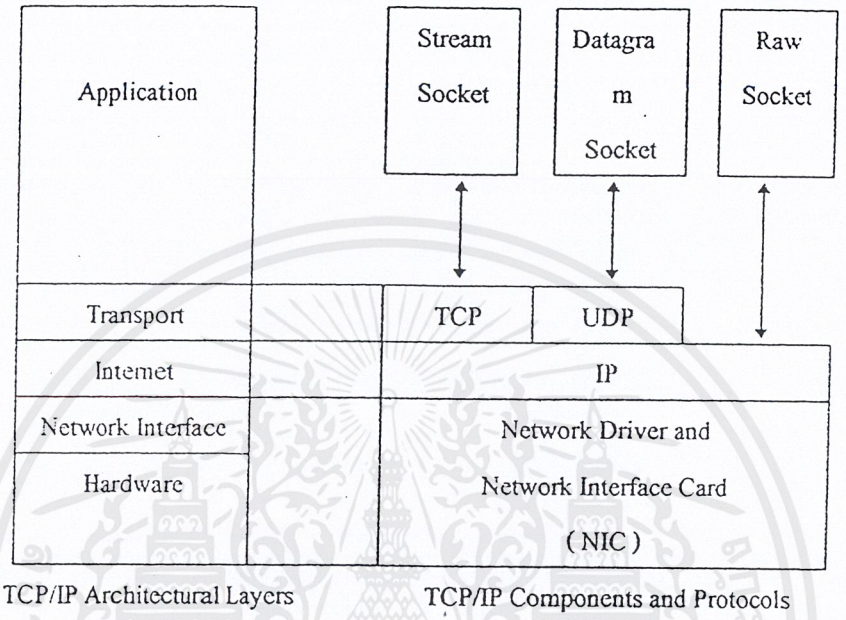
ซ็อกเก็ตที่ใช้ในการติดต่อสื่อสารบน TCP/IP จะมีอยู่ 3 แบบด้วยกันดังนี้

1. Stream Socket คือ ซ็อกเก็ตที่ใช้โปรโตคอล TCP ชั้นทรานสปอร์ตในการติดต่อ ซ็อกเก็ตชนิดนี้จะใช้การติดต่อแบบ Connection Oriented ซึ่งการรับส่งข้อมูลแบบนี้สามารถเชื่อถือได้

2. Datagram Socket คือ ซ็อกเก็ตที่ใช้โปรโตคอล UDP ชั้นทรานสปอร์ตในการติดต่อ Datagram Socket จะเชื่อถือไม่ค่อยได้ รูปแบบของข้อมูลที่ส่งจะถูกแบ่งออกเป็นส่วนๆโดยแต่ละอันไปปลายทางโดยอิสระจากกัน

3. Raw Socket คือซ็อกเก็ตที่อยู่ภายใต้ชั้น IP และ ICMP ซ็อกเก็ตชนิดนี้จะใช้เฉพาะการส่งที่มีจุดมุ่งหมายพิเศษ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.10 แสดงลักษณะทั้งสามแบบของซ็อกเก็ต

2.6.3 Socket Address

ระบบซ็อกเก็ตโดยทั่วๆไปจะอ้างอิงโครงสร้างของข้อมูลแบบพอยเตอร์ (pointer) ที่ประกอบไปด้วยข้อมูลต่างๆซึ่งก็คือซ็อกเก็ตแอดเดรสนั่นเอง

ซ็อกเก็ตแอดเดรสจะใช้เฉพาะการสื่อสารแบบ one-end

สำหรับโปรโตคอล TCP/IP จะมีโครงสร้างของซ็อกเก็ตแอดเดรส ดังนี้

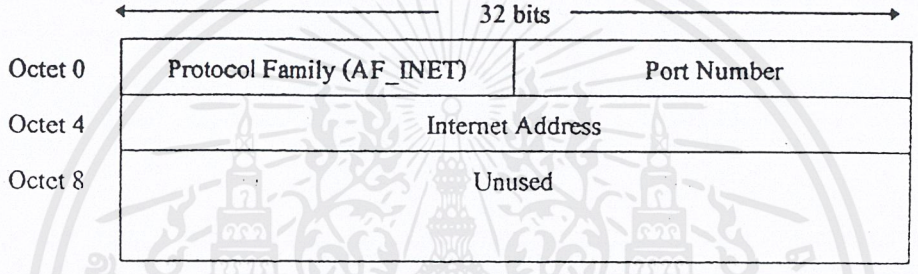
1. Family เป็นเลขจำนวนเต็ม 16 บิท ที่ชี้เฉพาะโปรโตคอล เช่น โปรโตคอล TCP/IP คือ AF_INET, UNIX คือ AF_UNIX

2. พอร์ต เป็นเลขจำนวนเต็ม 16 บิท ที่ระบุหมายเลขพอร์ตสำหรับใช้ในกระบวนการต่างๆ สำหรับผู้ให้บริการ หมายเลขพอร์ตที่ใช้จะถูกกำหนดโดยตัวผู้ให้บริการเอง ซึ่งผู้ขอใช้บริการที่ต้องการติดต่อกับผู้ให้บริการโดยใช้โปรแกรมนั้นจะต้องรู้หมายเลขพอร์ตที่ใช้ติดต่อจึงจะสามารถติดต่อกันได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

3. Address เป็นเลขจำนวนเต็ม 32 บิท ซึ่งภายในคือ ไอพีแอดเดรสที่อยู่ในรูปแบบของเลขฐานสอง ในการทำงานเพื่อติดต่อกัน ผู้ขอใช้บริการต้องรู้ไอพีแอดเดรสของผู้ให้บริการจึงจะสามารถทำการติดต่อกันได้ แต่ละการติดต่อของผู้ขอใช้บริการ-ผู้ให้บริการต้องมีการกำหนดซ็อกเก็ตแอดเดรส ซึ่งซ็อกเก็ตแอดเดรสก็หมายรวมถึงซ็อกเก็ตแอดเดรสของผู้ให้บริการและซ็อกเก็ตแอดเดรสของผู้ขอใช้บริการ กระบวนการติดต่อของผู้ขอใช้บริการและผู้ให้บริการต้องมีการชี้เฉพาะไปที่โปรโตคอลชั้นทรานสปอร์ตเพื่อใช้บริการติดต่อสื่อสาร รูปที่ 2.11 จะแสดงความหมายของชนิดข้อมูลและโครงสร้างของข้อมูลที่จะใช้ในการกำหนดซ็อกเก็ตแอดเดรส



รูปที่ 2.11 แสดงโครงสร้างของข้อมูลที่ใช้ในการกำหนดซ็อกเก็ตแอดเดรส

2.6.4 Socket System Call

ซ็อกเก็ต API ให้ Socket System Call เป็นโปรแกรมประยุกต์ที่ทำหน้าที่เรียกบริการการร้องขอที่จะทำการติดต่อสื่อสาร Socket System Call พื้นฐานที่ใช้กันทั่วไปอธิบายได้ดังนี้

Socket System Call

1. **accept.** ถูกใช้ใน connection -orientation application protocol ที่ใช้ TCP โดยใช้ในขบวนการผู้ให้บริการ ซึ่งผู้ให้บริการจะรอการร้องขอเพื่อติดต่อจากผู้ขอใช้บริการ
2. **bind.** ถูกใช้ร่วมกับ Endpoint identifier (internet address และหมายเลขพอร์ต) ซึ่งเลขจำนวนเต็มเป็นตัวอธิบายในการเปิดซ็อกเก็ต
3. **close.** ใช้ในการปิดซ็อกเก็ต โดยในรายละเอียดจะคล้ายกับวิธีปิดไฟล์
4. **connect.** ถูกใช้ใน connection-oriented application protocol ที่ใช้ TCP โดยขบวนการผู้ขอใช้บริการจะสร้างการติดต่อแบบ TCP กับผู้ให้บริการ
5. **gethostbyaddr.** จะใช้เมื่อต้องการข้อมูลเกี่ยวกับโฮสซึ่งรู้อินเตอร์เน็ตแอดเดรสแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

42277

6. **gethobyname.** จะใช้เพื่อให้ได้อินเตอร์เน็ตแอดเดรสของ remote host ซึ่งเรารู้ host name แล้ว
7. **gethostid.** ถูกใช้พื่อให้ได้อินเตอร์เน็ตแอดเดรสของ local host
8. **gethostname.** ถูกใช้เพื่อให้ได้ชื่อของ local host
9. **getpeername.** ถูกใช้เพื่อให้ได้ Endpoint address ซึ่งเกี่ยวข้องกับขบวนการด้านรีโมท
10. **getsockname.** ถูกใช้เพื่อให้ได้ชื่อที่ซ็อกเก็ตให้มา
11. **getprotobyname.** ถูกใช้เพื่อให้ได้ค่าเลขจำนวนเต็มซึ่งเป็นค่าของโปรโตคอลที่บอกชื่อของโปรโตคอล
12. **getservbyname.** ถูกใช้เพื่อให้ได้ค่าเลขจำนวนเต็มซึ่งเป็นค่าที่ให้ชื่อของการบริการของโปรแกรมประยุกต์บนชั้น แอพพลิเคชั่น ของโมเดล TCP/IP ที่รู้จักกันดี
13. **getsockopt.** ถูกใช้เพื่อให้ได้รายชื่อพารามิเตอร์ของซ็อกเก็ต
14. **listen.** ใช้ในการติดต่อแบบ connection-oriented application protocol ที่ใช้ TCP เป็นขบวนการของผู้ให้บริการที่จะแสดงให้เห็นว่าผู้ให้บริการจะยอมรับการติดต่อจากผู้ขอใช้บริการ เมื่อมีการร้องขอเพื่อที่จะสร้างการติดต่อแบบ TCP
15. **read.** ถูกใช้เพื่อที่จะรับข้อมูลมาจากซ็อกเก็ต
16. **readv.** เป็นอีกตัวเลือกหนึ่งของคำสั่ง read เพื่อที่จะรับข้อมูลใส่ไว้ในบัฟเฟอร์ที่ไม่ติดกัน
17. **recv.** เป็นอีกตัวเลือกหนึ่งของ read เพื่อที่จะรับข้อมูลที่กำลังเข้ามาจากซ็อกเก็ตและใช้แฟลกมาเป็นข้อกำหนดเพิ่มเข้าไปด้วย
18. **recvfrom.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะรับ user datagram จาก datagram socket เพื่อที่จะให้ได้ที่อยู่ซ็อกเก็ตของผู้ส่ง
19. **recvmsg.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะรับ user datagram จาก datagram socket โดยในบัฟเฟอร์จะประกอบด้วยส่วนหัวของ user datagram ซึ่งก็คือส่วนหนึ่งของข้อมูลของ user datagram
20. **select.** ถูกใช้เพื่อให้ขบวนการรอคอย (wait) สมบูรณ์
21. **send.** ใช้เป็นอีกทางเลือกของ write เพื่อที่จะส่งข้อมูลบนซ็อกเก็ต และใช้เงื่อนไขของแฟลกเพิ่มมาด้วย
22. **sendmsg.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะส่ง user datagram ไปบน datagram socket โดยในบัฟเฟอร์จะประกอบด้วยส่วนหัวของ user datagram ซึ่งก็คือส่วนหนึ่งของข้อมูลของ user datagram

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

23. **sendto.** ใช้กับ connectionless application protocol ที่ใช้แบบ UDP เพื่อที่จะส่งข้อมูลบน datagram socket และระบุ socket address ของ remote process, sendto จะถูกเรียกมาเพื่อให้สัมพันธ์กับการสร้างการติดต่อในเวลาที่จะส่งข้อมูล
24. **setsockopt.** ใช้เพื่อกำหนดค่าสำหรับ พารามิเตอร์ต่างๆของ ซ็อกเก็ต
25. **shutdown.** ใช้กับ connection-oriented application protocol ที่ใช้แบบ TCP โดย shutdown จะทำการยกเลิกการติดต่อที่ TCP สร้างขึ้น
26. **socket.** ใช้เพื่อเริ่มต้นโครงสร้างทางข้อมูลของซ็อกเก็ต ทำการแยกแยะโปรโตคอล ที่ใช้งานสำหรับการติดต่อสื่อสาร ในชั้นทรานสปอร์ต และได้ค่าจำนวนเต็มที่สามารถบอกรายละเอียดได้ โดยหลังจากที่ระบบการเรียกของ ซ็อกเก็ต สามารถใช้ตัวเลขนี้อ้างอิงถึงซ็อกเก็ตได้
27. **write.** ใช้เพื่อส่งข้อมูลบนซ็อกเก็ต
28. **writev.** ใช้เพื่อเป็นอีกทางเลือกหนึ่งของ write เพื่อที่จะส่งข้อมูลจาก บัฟเฟอร์ที่ไม่ติดกัน

2.6.5 Application Protocol

สามารถมีซ็อกเก็ต API ได้ทั้ง 2 แบบ คือทั้ง Datagram Socket และ Stream Socket ที่จะใช้ในการออกแบบโปรแกรม โดยมีโปรโตคอลการติดต่อแบ่งเป็น 2 ประเภทดังนี้

Connectionless Application Protocol

เป็นโปรแกรมประยุกต์ ผู้ขอใช้บริการและผู้ให้บริการที่ใช้ Connectionless Application Protocol จะใช้กับ Datagram Socklet เพื่อจะติดต่อสื่อสารบนการติดต่อ UDP ดังแสดงในรูป 2.11 แสดงให้เห็นถึงลำดับของ Socket System Call ที่ใช้แบบ Connectionless Application Protocol การทำแบบนี้จะไม่มีการตรวจสอบความผิดพลาดและการส่งซ้ำ (Retransmission)เนื่องจากเป็นการบริการแบบ UDP

พิจารณากระบวนการของผู้ให้บริการจะใช้ Socket, Bind, System Calls และรับข้อมูลโดยใช้ Recvfrom ซึ่ง System Calls ทั้ง 5 กระบวนการนี้ต้องการการเชื่อมต่อที่สมบูรณ์ระหว่างกระบวนการผู้ขอใช้บริการและกระบวนการผู้ให้บริการ สำหรับ Connectionless Application Protocol นี้การเชื่อมต่อจะยังไม่สมบูรณ์จนกระทั่งถึงเวลาที่กระบวนการผู้ขอใช้บริการได้ส่งข้อมูลไปยังกระบวนการผู้ให้บริการแล้วจริงๆ

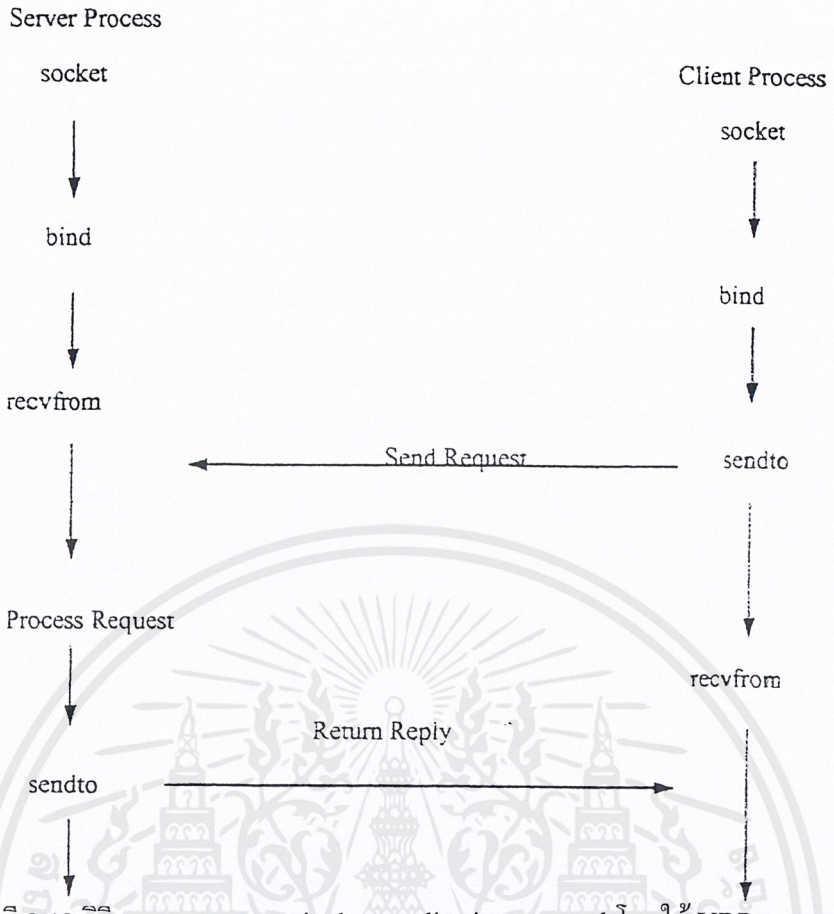
พิจารณากระบวนการของผู้ขอใช้บริการจะใช้ Socket, Bind, Sendto System Calls เพื่อให้การเชื่อมต่อระหว่างกระบวนการผู้ขอใช้บริการและกระบวนการผู้ให้บริการนั้นสมบูรณ์ ซึ่งการเชื่อมต่อจะสมบูรณ์เมื่อเวลาที่ผู้ขอใช้บริการส่ง datagram อันดับแรกไปถึงผู้ให้บริการแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Server System Calls

1. ผู้ให้บริการจะสั่งให้ socket system call เข้าถึงข้อมูลของโปรโตคอล ใน 5-tuple ที่สัมพันธ์กัน ซึ่งในที่สุดแล้วก็จะถูกสร้างเพื่อสนับสนุนการติดต่อสื่อสารระหว่างกระบวนการผู้ขอใช้บริการและกระบวนการผู้ให้บริการใน Connectionless Application Protocol จะเห็นว่า Socket System Call เจาะจงที่จะใช้ UDP
2. ผู้ให้บริการจะสั่งให้ bind system call (ระบุที่อยู่ซ็อกเก็ตของผู้ให้บริการ) ทำแทนในส่วนที่เหลือของข้อมูลข่าวสารที่กำลังซ่อมแซม sever half-association ของผู้ให้บริการบรรจุไปด้วยโปรโตคอลที่มีลักษณะเฉพาะตัว ในตอนนี้ internet address และหมายเลขพอร์ตของผู้ให้บริการกำลังทำงานอยู่
3. ผู้ให้บริการจะสั่งให้ทำ recvfrom system call เพื่อให้ผู้ให้บริการรับข้อมูลจากผู้ขอใช้บริการที่กำลังเข้ามา recvfrom system call เป็นสาเหตุให้ผู้ให้บริการ กระบวนการรอคอยจนกระทั่งข้อมูลของผู้ขอใช้บริการมาถึง ซึ่งมันก็คือความต้องการที่จะขอใช้บริการจากทางผู้ให้บริการ
4. เมื่อ user datagram จากผู้ขอใช้บริการมาถึง ผู้ให้บริการจะทำกระบวนการ request process ซึ่งจะเห็นว่าส่วนหัวข้อมูลของ user datagram ที่ส่งมาจะประกอบไปด้วย internet address และหมายเลขพอร์ตของผู้ขอใช้บริการ ข่าวสารเหล่านี้จะจัดหาหนทางไปยังผู้ให้บริการจนกระทั่ง 2 ชิ้นสุดท้ายของข้อมูลถูกร้องขอเพื่อให้ 5-tuple สมบูรณ์ ซึ่งจะสัมพันธ์ระหว่างด้านผู้ขอใช้บริการและผู้ให้บริการ
5. ผู้ให้บริการตอบกลับไปยังผู้ขอใช้บริการ 1 ครั้งหรือมากกว่า โดยใช้คำสั่ง sendto system call ซึ่งจะทำการส่ง user datagram กลับไปหาผู้ขอใช้บริการ sendto system call จะใช้ socket address ของผู้ขอใช้บริการและเป็นตัวอ้างอิงเพื่อที่จะตกลงตอบรับการติดต่อ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.12 วิธีการแบบ connectionless application protocol โดยใช้ UDP

Client system calls

1. ผู้ขอใช้บริการจะสั่งให้ socket system call เข้าถึงข้อมูลของโปรโตคอลสำหรับ half-association อนึ่งใน connectionless application protocol จะเห็นว่า socket system call จะใช้ UDP
2. ผู้ขอใช้บริการจะสั่งให้ bind system call ระบุที่อยู่ซ็อกเก็ตของผู้ขอใช้บริการแล้วซึ่งเต็มไปด้วยส่วนที่เหลือของข้อมูลข่าวสารที่มีลักษณะ half-association ของผู้ขอใช้บริการ กระบวนการผู้ขอใช้บริการเป็นชนิดที่ร้องขอ networking software ที่จะเป็นแหล่งของ internet address ของผู้ขอใช้บริการและจัดหาหมายเลขพอร์ตที่ไม่ได้ใช้เอามาใช้ชั่วคราว
3. ผู้ขอใช้บริการจะสั่งให้ sendto system call ทำการส่ง user datagram ไปยังผู้ให้บริการ ทางด้านผู้ขอใช้บริการจำเป็นต้องรู้ที่อยู่ซ็อกเก็ตของผู้ให้บริการและที่อยู่ซ็อกเก็ตนี้จะต้องถูกอ้างอิงใน sendto system call
4. ถ้าผู้ขอใช้บริการคาดหวังว่าจะได้รับข้อมูลจากผู้ให้บริการ มันจะต้องสั่งให้เรียก recvfrom ซึ่งเป็นสาเหตุให้ผู้ขอใช้บริการคอยจนกระทั่งข้อมูลจากผู้ให้บริการจะมาถึง
5. หลังจากนั้นกระบวนการติดต่อกันระหว่างผู้ขอใช้บริการและผู้ให้บริการจะสิ้นสุดลงเป็นอันสิ้นสุดกระบวนการ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Connection-Oriented Application Protocol

เป็นโปรแกรมประยุกต์ผู้ขอใช้บริการและผู้ให้บริการที่ใช้ Connection-Oriented Application Protocol จะใช้กับ Stream Socket เพื่อที่จะติดต่อสื่อสารบนการติดต่อบน TCP ดังรูปที่ 2.12

การส่งข้อมูลโดยใช้ Connection-Oriented อันดับแรกผู้ให้บริการจะเตรียมตัวให้พร้อมด้วยตัวของมันเอง และรอการร้องขอการติดต่อจากผู้ขอใช้บริการ ส่วนผู้ขอใช้บริการจะร้องขอการติดต่อและส่งข้อมูลไปยังผู้ให้บริการภายใต้การทำงานของ TCP จะมีการตรวจสอบความผิดพลาด มีการส่งข้อมูลซ้ำใหม่และมีการควบคุมการไหลของข้อมูล

พิจารณากระบวนการของผู้ให้บริการจะใช้ Socket, Bind System Call ต่อมาหากมีการติดต่อจะใช้ Accept System Call เพื่อยอมรับการติดต่อนั้น วิธี Connection-Oriented Application Protocol จะสมบูรณ์เมื่อการเชื่อมต่อสำเร็จ

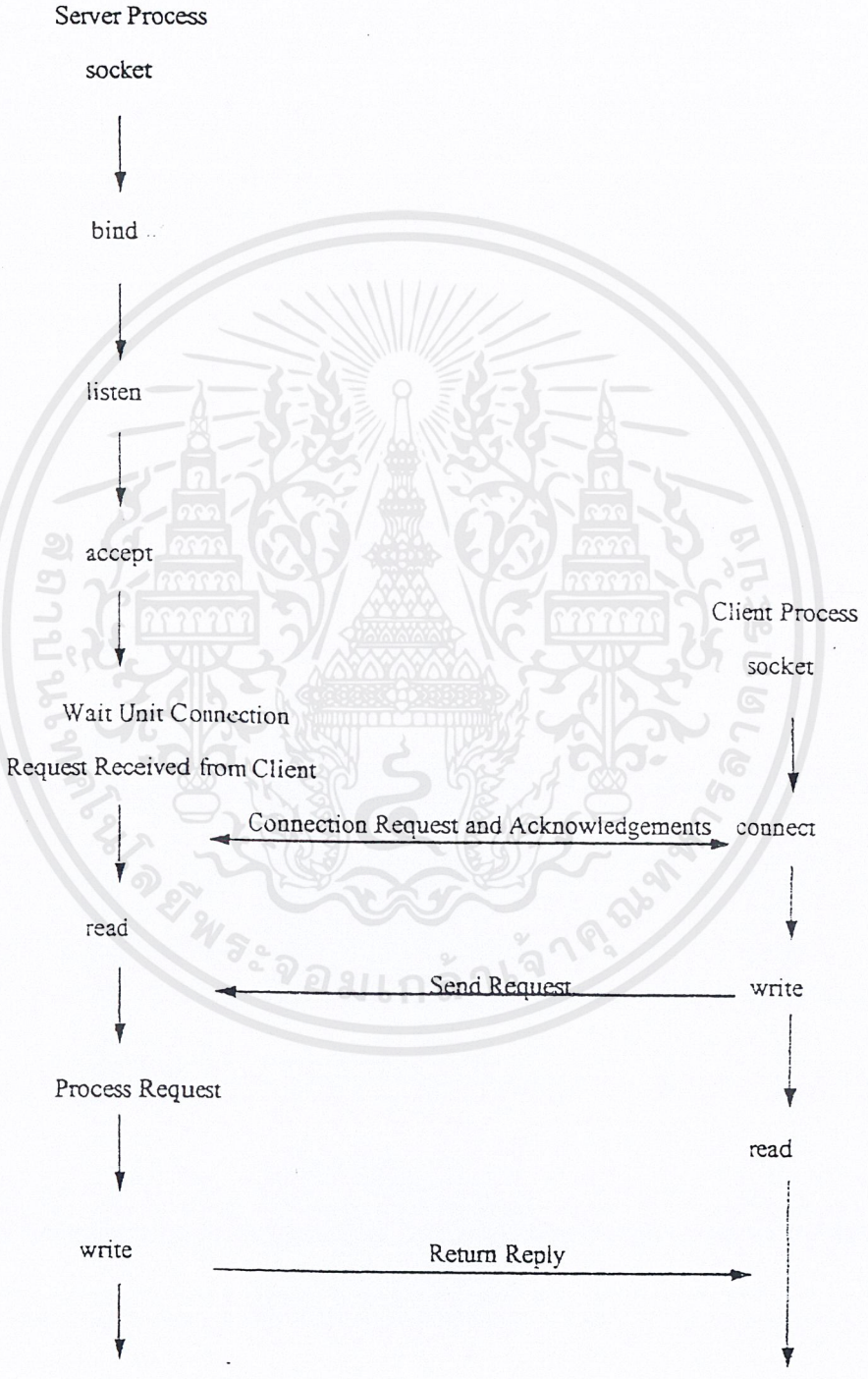
พิจารณากระบวนการของผู้ขอใช้บริการจะใช้ Socket, Connect System calls จะเห็นว่ามี Three way Handshake ซึ่งจะเริ่มเมื่อผู้ขอใช้บริการใช้ Connect System Call ตามด้วยการแลกเปลี่ยน ข้อมูลระหว่างผู้ขอใช้บริการและผู้ให้บริการ ดังนั้นการติดต่อนี้จะสมบูรณ์เมื่อเวลาที่ผู้ขอใช้บริการใช้ Connect System Call

Sever System Calls

1. ผู้ให้บริการจะสั่งให้ socket system call เข้าถึงข้อมูลของโปรโตคอลใน 5-tuple ที่สัมพันธ์กัน ซึ่งในที่สุดแล้วก็จะถูกสร้างเพื่อสนับสนุนการติดต่อสื่อสารระหว่างกระบวนการผู้ขอใช้บริการและกระบวนการผู้ให้บริการใน connection-oriented application จะเห็นว่า socket system call เจาะจงที่จะใช้ TCP
2. ผู้ให้บริการจะสั่งให้ bind system call (ระบุที่อยู่ซ็อกเก็ตของผู้ให้บริการ) ทำแทนในส่วนที่เหลือของข้อมูลข่าวสารที่กำลังซ่อมแซม sever half-association. โดย half-association ของผู้ให้บริการบรรจุไปด้วยโปรโตคอลที่มีลักษณะเฉพาะตัว ในตอนนี้ internet address และหมายเลขพอร์ตของผู้ให้บริการกำลังทำงานอยู่
3. ผู้ให้บริการจะสั่งให้ทำ listen system call ซึ่งแสดงว่าผู้ให้บริการกำลังรอการร้องขอจากผู้ขอใช้บริการเพื่อการติดต่อ TCP
4. ผู้ให้บริการจะสั่งให้ทำ accept system call เมื่อมีการร้องขอการสร้างการเชื่อมต่อ TCP ที่ได้รับจาก client accept system call จะอ้างอิงซ็อกเก็ตแอดเดรสที่เกิดจากซ็อกเก็ต แอดเดรสของกระบนการผู้ขอใช้บริการเมื่อมีการร้องขอสำหรับการเชื่อมต่อที่ได้รับจริงๆ
5. ผู้ให้บริการจะสั่งให้ทำ read system call หรือ recv system call เพื่ออนุญาตให้ผู้ให้บริการรับข้อมูลจากผู้ขอใช้บริการบนการเชื่อมต่อ TCP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6. ผู้ให้บริการสามารถสั่งให้ทำ write system call หรือ send system call เพื่อที่จะอนุญาตให้ผู้ให้บริการส่งข้อมูลไปยังผู้ขอใช้บริการได้



รูปที่ 2.13 วิธีการแบบ connection-oriented application protocol โดยใช้ TCP

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Client system calls

1. ผู้ขอใช้บริการจะขอให้ socket system call เข้าถึงข้อมูลของโปรโตคอลสำหรับ half-association อนึ่งใน connection-oriented application protocol จะเห็นว่า socket system call จะใช้ TCP
2. ผู้ขอใช้บริการจะสั่งให้ connect system call ที่อ้างอิงซ็อกเก็ตแอดเดรสของผู้ขอใช้บริการที่จะสร้างการเชื่อมต่อ TCP กับผู้ให้บริการ connect system call เกิดได้จากการมีข้อมูลไหลกลับ และอันดับ 4 ระหว่างกระบวนการผู้ขอใช้บริการและกระบวนการผู้ให้บริการเพื่อที่จะสร้างกระบวนการ three way handshake ขึ้นอย่างสมบูรณ์เพื่อที่จะต้องการสร้างการติดต่อข้อมูลที่แลกเปลี่ยนระหว่างกระบวนการ handshake รวมทั้งข้อมูลที่ต้องการเพื่อที่จะสร้างการเชื่อมต่อที่สมบูรณ์ระหว่างผู้ขอใช้บริการและผู้ให้บริการ กระบวนการผู้ขอใช้บริการต้องรู้ซ็อกเก็ตแอดเดรสของกระบวนการผู้ให้บริการในการออกคำสั่ง connect system call และเป็นชนิดที่ร้องขอ networking software ที่จะเป็นแหล่งของ internet address ของผู้ขอใช้บริการและจัดหาหมายเลขพอร์ตที่ไม่ได้ใช้เอามาใช้ชั่วคราว
3. ผู้ขอใช้บริการจะสั่งให้ write system call หรือ send system call เพื่อที่จะส่งข้อมูลไปยังผู้ให้บริการบนการเชื่อมต่อ TCP ที่ถูกสร้างขึ้น
4. ผู้ขอใช้บริการสามารถสั่งให้ read system call หรือ recv system call เพื่อที่จะรับข้อมูลที่เข้ามาจากผู้ให้บริการ
5. หลังจากนั้นกระบวนการติดต่อกันระหว่างผู้ขอใช้บริการและผู้ให้บริการจะสิ้นสุดลง เป็นอันสิ้นสุดกระบวนการ

2.6.6กระบวนการ system call พื้นฐาน

system call ในหัวข้อนี้ใช้บน TCP/IP และจะให้ค่าผลลัพธ์เป็น integer ซึ่งค่านั้นจะบอกได้ถึงความผิดพลาดที่เกิดขึ้นในการทำงานของซ็อกเก็ตนั้น

Socket system call

ใช้สำหรับสร้างซ็อกเก็ตและสร้างการติดต่อระหว่าง 2 กระบวนการติดต่อโดยมีการกำหนดรูปแบบดังนี้

Socket (family,type,protocol)

- family : แสดง family ของโปรโตคอลที่ใช้ในการติดต่อ เช่น โปรโตคอล TCP/IP คือ AF_INET
- type : แสดงชนิดของซ็อกเก็ตที่ใช้ เช่น stream socket คือ SOCK_STREAM
- proyocol : ชี้เฉพาะโปรโตคอลที่ใช้ในการติดต่อ ทั่วๆไปแล้ว จะกำหนดให้เท่ากับศูนย์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง

การสร้าง stream socket สำหรับ connection – oriented application protocol โดยใช้ TCP

stream_ sockfd = socket (AF_INET,SOCK_STREAM,0) ;

การสร้าง datagram socket สำหรับ connectionless application protocol โดยใช้ UDP

dgrm_sockfd= socket (AF_INET , SOCK_DGRAM,0);

Bind System Call

จะสร้างแอดเดรสและพอร์ตสำหรับการติดต่อ โดยที่ผู้ให้บริการจะบันทึก internet address และหมายเลขพอร์ตด้วย software ของการติดต่อสื่อสาร TCP/IP ส่วนกระบวนการของผู้ขอใช้บริการที่มีการติดต่อแบบ Connectionless Application Protocol ทั่วๆไปจะใช้ Bind เพื่อบันทึก internet address และสร้างหมายเลขพอร์ตชั่วคราวที่ผู้ให้บริการสามารถส่งข้อมูลกลับมาที่กระบวนการผู้ขอใช้บริการ

ผู้ขอใช้บริการสามารถร้องขอ software ของการติดต่อ TCP/IP เพื่อจะให้ internet address ของผู้ขอใช้บริการเองและสร้างหมายเลขพอร์ตชั่วคราวที่ยังไม่ได้มีการใช้ระหว่างการติดต่อ แต่ในการติดต่อแบบ Connection-Oriented Application Protocol กระบวนการผู้ขอใช้บริการไม่จำเป็นที่จะต้องใช้ Bind ซึ่งถ้าผู้ขอใช้บริการไม่เรียกใช้ Bind

Connection System Call จะถูกเรียกใช้โดยอัตโนมัติเพื่อให้กระบวนการ Binding สมบูรณ์ รูปแบบแสดงได้ดังนี้

bind (socket, local_address, address_length)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- local address : แสดง local address ที่ bind สร้างขึ้น
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสในหน่วยไบต์

ตัวอย่าง

bind (sockfd, (struct sockaddr*)&own_addr, sizeof (own_addr)):

Connect System Call

หลังจากมีการสร้างซ็อกเก็ตแอดเดรสแล้ว ซ็อกเก็ตเป้าหมายจะทำการติดต่อโดยใช้ Connect System Call ซึ่งทั่วๆไปแล้วจะออกโดยกระบวนการผู้ขอใช้บริการ ในการติดต่อแบบ Connection-Oriented Application Protocol ซึ่งจะมีกระบวนการ Three way handshake รวมทั้งแลกเปลี่ยนอัตโนมัติทั้งข้อมูลที่จำเป็นและเชื่อมต่ออย่างสมบูรณ์ระหว่างกระบวนการทั้งสอง แสดงรูปแบบได้ดังนี้

connect (socket, destination_address, address_length)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- destination address : แสดงแอดเดรสปลายทางที่ต้องการติดต่อ
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสปลายทางในหน่วยไบต์

ตัวอย่าง

connect (sockfd, (struct sockaddr*) & serv_addr,sizeof (serv_addr));

Listen System Call

Listen System Call ออกโดยกระบวนการผู้ให้บริการในการติดต่อแบบ Connection-Oriented Application Protocol โดยใช้โปรโตคอล TCP โดยจะใช้ Listen System Call เพื่อบอกแก่ software ในการติดต่อแบบ TCP/IP ซึ่งกระบวนการผู้ให้บริการจะยอมรับ (Accept) จากกระบวนการผู้ขอใช้บริการ สำหรับการสร้างการติดต่อบน TCP แสดงรูปแบบได้ดังนี้

Listen (socket, queue_length)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- queue length : เป็นตัวเลขที่ชี้ให้เห็นว่ามีการร้องขอการติดต่อเพื่อรอคอยกระบวนการตอบรับจำนวนเท่าใด ตัวแปรนี้มีค่ามากที่สุดเท่ากับ 5

ตัวอย่าง

listen (sockfd, 5);

Accept System Call

ส่วนใหญ่แล้ว Accept System Call จะถูกออกโดยกระบวนการทางผู้ให้บริการใน Connection-Oriented Application Protocol เป็นผลให้กระบวนการทางผู้ให้บริการต้องรอการร้องขอที่จะเข้ามาจากผู้ขอใช้บริการ รูปแบบแสดงได้ดังนี้

accept (socket, client_address, address_length)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- client address : แอดเดรสของผู้ขอใช้บริการที่ต้องการทำการติดต่อ
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสของผู้ขอใช้บริการในหน่วยไบต์

ผลที่ได้จะสร้างโครงสร้างข้อมูลซ็อกเก็ตใหม่และคืนค่าตัวแปรของซ็อกเก็ตที่สามารถใช้หลังจากอ้างอิงโครงสร้างข้อมูลของซ็อกเก็ตใหม่

ตัวอย่าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

newsockof = accept (sockfd, (struct sockaddr*) & cli_addr, & client);

Write and Send System Call

Write and Send System Call ใช้สำหรับส่งข้อมูลไปยังอีกฝ่ายหนึ่ง Write และ Send คล้ายกัน แต่ Send System Call จะประกอบด้วยตัวแปรที่มากกว่า มีรูปแบบดังนี้

write (socket, buffer, length)

send (socket, buffer_address, length_flags)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลในการส่ง
- length : จำนวนไบต์ที่กำหนดเพื่อใช้ในการส่ง
- flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการส่ง

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่ส่งได้

ตัวอย่าง

nwritten = write (sockfd, buffer, noctets);

nwritten = send (sockfd, buffer, noctets, sendflag);

Sendto System Call

Sendto System Call คล้ายกับ Send และ Write System Call แต่จะใช้กับการติดต่อแบบ Connectionless Application Protocol บน UDP Sendto System Call จะสร้างการเชื่อมต่อระหว่าง 2 กระบวนการซึ่งการติดต่อจะสมบูรณ์เมื่อข้อมูลได้ถูกส่งออกไปแล้ว มีรูปแบบดังนี้

sendto (socket, buffer_address, length, flags, destination, address_length)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลในการส่ง
- length : จำนวนไบต์ที่ใช้เก็บข้อมูลในการส่ง

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่ส่งได้จริง

ตัวอย่าง

nwritten = sendto (sockfd, buffer, noctets, sendflag, (struct sockaddr*) & serv_addr,

sizeof (serv_addr));

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Read and Recv System Call

Read and Recv System Call ใช้สำหรับการยอมรับ (Accept) ข้อมูลที่เข้ามาจากอีกฝ่ายหนึ่ง โดยที่ Recv system Call ทำหน้าที่คล้ายกับ Read System Call แต่มีตัวแปรที่มากกว่า รูปแบบแสดงได้ดังนี้

read (socket, buffer, length)

recv (socket, buffer_address, length, flags)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลที่ใช้ในการรับ
- length : จำนวนที่กำหนดเพื่อใช้ในการรับ
- flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการรับ

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่รับได้จริง

ตัวอย่าง

nread = read (sockfd, buffer, bufflength);

nread = recv (sockfd, buffer, bufflength, recvflag);

Recvfrom System Call

Recvform System Call คล้ายกับ Recv และ Read System Call แต่จะใช้การติดต่อแบบ Connectionless Application Protocol ที่ใช้ UDP Recvform System Call จะสร้างการเชื่อมต่อระหว่าง 2 กระบวนการ ซึ่งการติดต่อจะสมบูรณ์เมื่อได้รับข้อมูลเข้ามาเรียบร้อยแล้ว มีรูปแบบดังนี้

Recvform (socket, buffer_address, length, flags,source_address,address_length)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call
- buffer address : เป็นบัฟเฟอร์ที่ใช้เก็บข้อมูลในการรับ
- length : จำนวนไบต์ที่กำหนดเพื่อใช้ในการรับ
- flags : เป็นตัวที่ใช้กำหนดเจาะจงทางเลือกพิเศษสำหรับการรับ
- source addess : แสดงแอดเดรสที่ต้องการส่งข้อมูล
- address length : เป็นเลขจำนวนเต็มแสดงความยาวของแอดเดรสในหน่วยไบต์

ผลที่ได้จาก System Call นี้คือ จำนวนไบต์ที่รับได้จริง

ตัวอย่าง

nwritten = recvform (sockfd, buffer, bufflength, recvflag (struct sockaddr*) &

serv_addr, & length);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Close System Call

การเรียกใช้ Close System Call นั้นจะทำหลังจากมีการเชื่อมต่อที่สมบูรณ์แล้ว การเรียกใช้ System Call นี้ จะเกิดผลดีในการจัดเตรียมเนื้อที่ในหน่วยความจำ ในการติดต่อแบบ Connection-Oriented Application Protocol ที่ใช้ TCP จะเรียก Close System Call หลังจากการติดต่อสมบูรณ์แล้วโดยที่มั่นใจว่าข้อมูลต่างๆที่ต้องการนั้นได้ถูกส่งออกไปเรียบร้อยแล้วก่อนที่การติดต่อจะสิ้นสุดลง มีรูปแบบดังนี้

close (socket)

- socket : แสดงถึงซ็อกเก็ตที่กำหนดขึ้นใน Socket System Call

ตัวอย่าง

close (sockfd);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

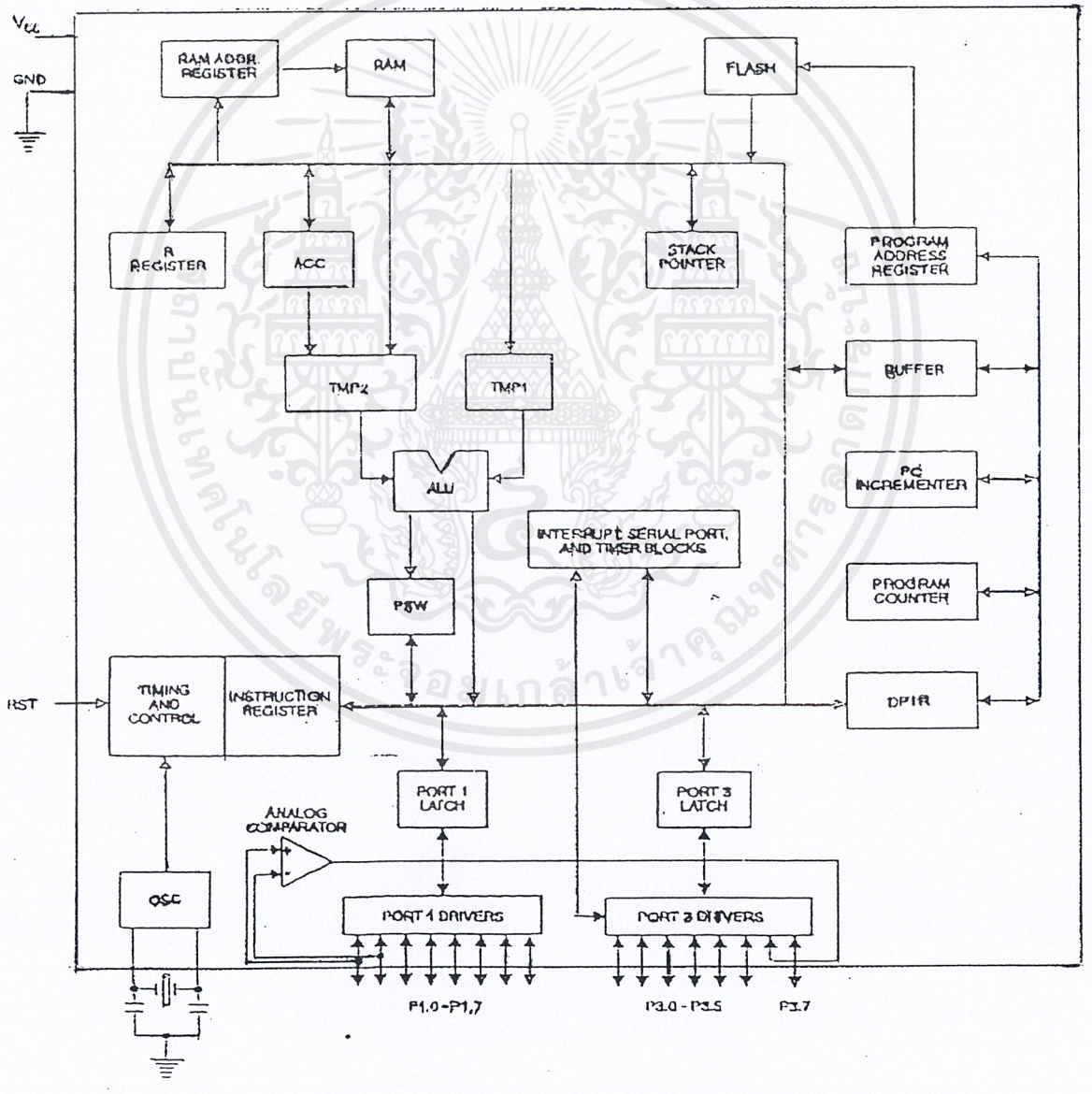
บทที่ 3
พื้นฐานไมโครคอนโทรลเลอร์ และ RS - 232

3.1 ไมโครคอนโทรลเลอร์

เป็นส่วนควบคุมการทำงานของ ชุดควบคุมมอเตอร์ ซึ่งหน้าที่ของไมโครคอนโทรลเลอร์

1. รับคำสั่งควบคุมจาก คอมพิวเตอร์ มาประมวลเพื่อที่จะทำงานต่อไป
2. จัดเก็บข้อมูลตำแหน่งของมอเตอร์ทั้งสองตัวเพื่อที่จะส่งไปให้ Computer

Block Diagram



รูปที่ 3.1 แสดงสถาปัตยกรรมของ 89C2501

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

89C51 ไมโครคอนโทรลเลอร์ แต่ละข้างจะมีขาอยู่ข้างละ 20 ขา รวมกันทั้งหมด 40 ขา โดย V ขาที่ 40 เป็นขาที่ป้อนไฟเลี้ยง + 5 โวลท์ เข้าไปเพื่อให้วงจรรวมทำงานได้ ระดับแรงดันของ ลอจิก 0 และ 1 ของ 89C51 จึงต่อเข้ากับอุปกรณ์ ลอจิกแบบ TTL ได้โดยตรง พอร์ต 1 เป็นพอร์ตขนานขนาด 8 บิต อยู่ที่ขา 1 ถึง 8 เริ่มจากบิต 0 ถึง บิต 7 ตามลำดับ แต่ละขาจะเขียนว่า P1.0,P1.1,.....P1.7 หมายถึงบิต 7 ของพอร์ต 1 ซึ่งเป็นบิตที่มีนัยสำคัญสูงสุด และ P1.0 คือบิต 0 ของพอร์ต 1 เป็นบิตที่มีนัยสำคัญน้อยที่สุด พอร์ต 0 นี้ ใช้ได้ทั้งการรับส่ง ตำแหน่งของข้อมูลกับหน่วยความจำหรือใช้เป็นพอร์ตรับส่ง ข้อมูลก็ได้ ข้อมูลที่ถูกส่งออกทางพอร์ต 1 จะถูกค้างค่าไว้ที่ขาพอร์ต

วงจรภายในส่วน Timing และ Control จะเป็นตัวสร้างสัญญาณมาควบคุมวงจร เพื่อให้การทำงานแต่ละอย่างข้างต้น เมื่อแต่ละบิตของพอร์ต 1 มีการทำงาน คือรับส่งข้อมูลกับ Data Memory หรือใช้รับข้อมูลจาก Program Memory และใช้รับส่งข้อมูลผ่านทาง พอร์ตโดยตรง ในกรณีที่ไม่มีการใช้หน่วยความจำ Program Memory หรือ Data Memory ภายนอกแล้ว วงจร Timing and Control จะทำให้สภาวะลอจิกของขา Control เป็น 1 ซึ่งทำให้สวิทซ์ MUX อยู่ในตำแหน่งข้างบนเมื่อพอร์ต 1 จะส่งข้อมูลซึ่งเป็นค่าตำแหน่งหน่วยความจำหรือข้อมูลที่จะเขียนออกไปยังหน่วยความจำภายนอก ก็จะส่งค่าดังกล่าวมายัง ADDR/DATA ถ้าข้อมูลที่ส่งมาเป็น 1 ก็จะทำให้สัญญาณออกจาก AND GATE เป็น 1 และสัญญาณที่ออกจาก Inverter ดังนั้น FET ตัวบนเปิด (สภาวะเปิด ของ FET คือความต้านทานระหว่างขา D กับ S มีค่าต่ำมากเสมือนวงจรปิด) ส่วน FET ตัวล่างปิด (สภาวะปิดของ FETคือ ความต้านทานระหว่างขา D กับ S มีค่าสูงมากเสมือนกับเป็นวงจรเปิด) สภาวะลอจิกที่ขา P1.X PIN จะเป็น 1 แต่ถ้าข้อมูลที่ส่งออกมายัง ADDR/DATA เป็น 1 ก็จะทำให้สัญญาณจาก AND GATE เป็น 0 และสัญญาณที่ออกจาก Inverter เป็น 1 ดังนั้น FET ตัวบนจะปิด ส่วน FET ตัวล่างจะเปิด ทำให้สภาวะลอจิกที่ขา P0.X PIN เป็น 0 เมื่อ 89C51 ต้องการใช้พอร์ต 0 สำหรับอ่านข้อมูลจากหน่วยความจำภายนอก ก็จะทำได้โดยวงจร Timing andControl ทำให้สภาวะลอจิกของสัญญาณ Control ในรูปเป็น 0 ทำให้เอาท์พุท จาก AND GATE เป็น 0 FET ตัวบนจะปิด และสวิทซ์ MUX จะอยู่ในตำแหน่งข้างล่างดังนั้น FET ตัวล่างจะเปิดหรือปิด ก็แล้วแต่ ข้อมูลที่ขา Q มีสภาวะลอจิก 0 ทำให้ FET ตัวล่างปิดดังนั้น ขา P0.X PIN จะอยู่ในสภาวะอิมพีแดนซ์สูง เพราะ FET ทั้ง 2 ตัวปิด

พอร์ต 1 เป็นพอร์ตขนานขนาด 8 บิต คือ ขา P1.0 หมายถึงบิต 0 ของพอร์ต 1 เป็นบิตนัยสำคัญต่ำสุด และบิต P1.7 หมายถึง บิตที่ 7 ของพอร์ต 1 ซึ่งเป็นบิตนัยสำคัญสูงสุด

ส่วนที่ 1 พอร์ต 1 Latch ซึ่งจะมีการทำงานเหมือนส่วนที่ 1 ของพอร์ต 0 ส่วนที่ 2 คือ พอร์ต 1 Driver นี้ จะมีตัวต้านทานต่ออยู่เป็น Internal Pull Up พอร์ต 1 นี้จะใช้ทำหน้าที่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เป็น ตัวรับส่งข้อมูลเท่านั้น ข้อมูลที่ถูกส่งออกมาทางพอร์ต 1 จะถูก Latchไว้ แล้วส่งออกไปทาง แต่ละขาก่อนที่จะอ่านข้อมูลเข้าไปทางพอร์ต 1 จะต้องเขียน 1 ไปยังทุกบิต ของพอร์ต 1 เสียก่อน เพื่อให้ FET อยู่ในสภาวะปิดก่อน มิฉะนั้นแล้วถ้ามีข้อมูล 0 ส่งมาค้างอยู่ก็จะทำให้ FET อยู่ในสภาวะเปิด ดังนั้น ถ้าสัญญาณภายนอกส่งเข้ามาที่ขานี้ก็จะถูกลัดลงกราวน์ โดยไม่สนใจว่าสภาวะลอจิก ของสัญญาณที่เข้ามาจะเป็นอะไร ข้อมูลที่อ่านเข้าไปจะเป็น 0 เสมอ

พอร์ต 3 คือ ขา P3.0 ถึง P3.7 ตามลำดับ โดยข้อมูลที่เขียนมายังพอร์ต 3 ทาง Internal Bus เหมือนกับพอร์ตอื่นๆ และพอร์ต 3 จะมี Internal Pull Up อยู่ทุกบิตแต่พอร์ต 3 นี้ แต่ละบิตจะใช้ในการทำงาน อื่นได้โดยใช้คำสั่งควบคุมการทำงาน และจะมีส่วนที่มีสัญญาณ Alternative Output Functuon ที่สร้างมาจาก ส่วน Timing andControl สัญญาณ Alternative Output Function เป็นสัญญาณที่ส่งออกในกรณี ที่ใช้พอร์ต 3 ทำงานใน ฟังก์ชั่นอื่น และจุด Alternative Input Function เป็นจุดที่จะเอาสัญญาณไปเข้ากับ ส่วนอื่นตามการทำงานของบิตนั้น แต่ละบิตของพอร์ต 3 จะมีฟังก์ชั่นอื่นๆ ดังนี้

P3.0/RXD (Serial Input Port) เป็นขาที่ใช้รับส่งข้อมูล

P3.1/TXD (Serial Output Port) เป็นขาที่ใช้ส่งข้อมูลแบบอนุกรม

P3.2/INTO (External Interrup) ใช้รับสัญญาณขัดจังหวะภายนอก

P3.4 /TO (Timer/Counter 0External Interrupt) ขารับสัญญาณเข้าไปในวงจร Timer/Counter ที่ทำหน้าที่นับจำนวนไซเคิล ของสัญญาณ TO นี้หรือ สัญญาณนาฬิกาก็ได้

P3.5/T1 (Timer/Counter 1 External Input) ขารับสัญญาณเข้าไปยัง Timer/Counter 1ซึ่งมีการทำงานเหมือนกับ TO

P3.7/RD ขาสัญญาณข้อมูล

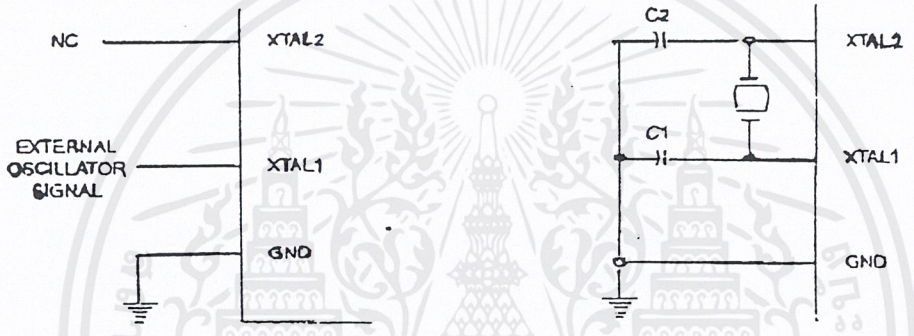
RST ขารีเซ็ทขานี้จะใช้ทำการรีเซ็ทการทำงานของ 89c51 ที่ขา RST ภายใน 89c51 จะมีตัวต้านทานต่อระหว่างขานี้ กับกราวน์ ถ้าป้อนสัญญาณที่สภาวะลอจิก 1 เข้าไปที่ขานี้จะเป็นการรีเซ็ทการทำงานของไมโครคอนโทรลเลอร์ ดังนั้นจึงสามารถต่อตัวเก็บประจุภายนอกระหว่างขา RST กับไฟเลี้ยง +5 โวลท์ เพื่อให้เกิดการรีเซ็ท เมื่อเริ่มป้อนไฟเลี้ยงให้กับ 89c2051 ซึ่งเรียกว่า “Power On Reset “

XTAL 1 ขาที่ 5 ขานี้จะต่อเข้ากับของ Inverting Amplifier (วงจรขยายแบบป้อนกลับเฟสสัญญาณ) ที่ประกอบเป็นวงจรออสซิลเลเตอร์ ซึ่งจะเห็นว่าวงจรภายในยออสซิลเลเตอร์ NAND GATEจะทำหน้าที่เป็นวงจรขยาย แบบกลับเฟสของสัญญาณที่จะควบคุมให้มีการออสซิลเลตหรือไม่ก็ขึ้นกับสัญญาณ PD ซึ่งต่อมาจากบิต PD ของรีจีสเตอร์ POON ถ้าต้องการใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สัญญาณนาฬิกา จากภายนอกมาเป็นสัญญาณนาฬิกา ควบคุมการทำงานของ 89c2051 ก็ให้ป้อนสัญญาณเข้ามาที่จุดนี้ แต่ถ้าต้องการใช้วงจร ออสซิเลเตอร์ภายในก็ให้ต่อ Crytal

XTAL 2 ขาที่ 4 ขานี้เป็นจุดเอาท์พุท ของวงจรขยายแบบกลับเฟส สัญญาณที่ประกอบเป็นวงจรออสซิเลเตอร์ ซึ่งอินพุทคือขาXTAL1 ถ้าจะใช้สัญญาณนาฬิกาที่สร้างมาจากภายนอกมาเป็นสัญญาณนาฬิกาของ 89c2051 แล้ว ให้ปล่อยขานี้ลอยไว้แล้ว ป้อนสัญญาณนาฬิกาจากภายนอก เข้ามาที่ขา XTAL 1 ดังรูปที่ 3.2



รูปที่ 3.2 การต่อ Oscillator

3.2 สื่อสารพอร์ตอนุกรม RS-232

ลักษณะของการส่งข้อมูลแบบอนุกรมนั้น ข้อมูลจะถูกส่งออกมาทีละบิต จากอุปกรณ์ส่งไปยังอุปกรณ์รับ ช่องสัญญาณในการส่งข้อมูลอาจจะใช้เพียง 1 หรือ 2 ช่องสัญญาณเท่านั้น ทำให้ค่าใช้จ่ายในการสื่อสารถูกกว่าแบบขนาน แต่อัตราการส่งข้อมูลจะล่าช้ากว่าการส่งแบบขนาน ในการส่งข้อมูลแบบอนุกรม ข้อมูลที่ต้องการส่งจะอยู่ในลักษณะเป็นไบต์จะถูกทยอยส่งทีละบิต และทางอุปกรณ์รับจะต้องรับข้อมูลเข้ามาทีละบิต แล้วมารวมกันเป็นไบต์ ซึ่งทางด้านอุปกรณ์รับจะต้องคอยตรวจสอบว่า บิตใดเป็นบิตเริ่มแรกของไบต์นั้น การตรวจสอบขึ้นอยู่กับรูปแบบของรหัสของบิตที่ใช้ ซึ่งในการรับส่งข้อมูลแบบอนุกรมระหว่าง ไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอกนั้น จำเป็นต้องมีมาตรฐานในการรับส่ง ซึ่งมาตรฐานที่นิยมมากที่สุดคือ มาตรฐาน RS-232

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.1 มาตรฐาน RS-232

เพื่อที่จะทำให้อุปกรณ์จากผู้ผลิตต่างกัน ทำงานร่วมกันได้ มาตรฐานหลายชนิด จึงได้รับการออกแบบขึ้น มาตรฐานที่ใช้กันอย่างกว้างขวางมากที่สุด คือ RS-232C ซึ่งปกติไมโครคอมพิวเตอร์จะมีพอร์ตที่เป็นแบบอนุกรมอยู่ในตัวอยู่แล้วและจะทำหน้าที่รับส่งข้อมูลในแบบอนุกรม

ตามจุดประสงค์ของมาตรฐาน RS-232C นั้น เพื่อจะสามารถเชื่อมต่อกันระหว่างอุปกรณ์รับส่งข้อมูลปลายทาง เช่น พอร์ตของคอมพิวเตอร์หลักหรืออุปกรณ์ปลายทาง กับอุปกรณ์สื่อสารข้อมูล หมายถึงอุปกรณ์ที่สามารถแปลงรูปคลื่น ดิจิตอลไปเป็นสัญญาณที่เหมาะสมสำหรับการส่งผ่านสายโทรศัพท์ หรือตัวกลางการสื่อสารอย่างอื่น โดยกระบวนการที่เรียกว่า มอดูเลชั่น

3.2.2 การอินเตอร์เฟสตามมาตรฐาน RS-232C

มาตรฐาน RS-232C ใช้สายสัญญาณเพียงเส้นเดียวในการส่งสัญญาณ โดยสัญญาณที่ส่งๆไปได้ทิศทางเดียว สำหรับความเร็วและระยะทางของการเชื่อมต่อ RS-232C สามารถเชื่อมต่อการถ่ายโอนข้อมูลได้จาก 0 - 20K bps บิตต่อวินาที ซึ่งเพียงต่อสำหรับไมโครคอมพิวเตอร์ที่มีขนาดอัตราบอด 110 ถึง 9600 ความยาวของสายเชื่อมสัญญาณระหว่าง DTC และ DCE โดยสัญญาณตามมาตรฐานของ RS-232 จำกัดอยู่แค่ 50 ฟุตหรือประมาณ 15 เมตร ซึ่งเพียงพอต่อการสื่อสารของไมโครคอมพิวเตอร์กับอุปกรณ์ภายนอก สำหรับแรงดันของระดับสัญญาณจะถูกกำหนดลงใน 2 บริเวณนี้ คือ

- แรงดันไฟบวก (สถานะ SPACE) อยู่ระหว่าง +5 ถึง +15 โวลท์สำหรับเอาท์พุต และระหว่าง +3 และ +15 โวลท์สำหรับอินพุท ความแตกต่างมีไว้เผื่อกรณีที่แรงดันไฟฟ้าสูญหายเนื่องจากความยาวของสายสัญญาณในทำนองเดียวกัน

- แรงดันไฟลบ (สถานะ MARK) ถูกกำหนดไว้ระหว่าง -5 ถึง -15 โวลท์สำหรับเอาท์พุตและ –3 ถึง –15 โวลท์สำหรับอินพุท

ซึ่งจะเห็นได้ว่าถ้าให้สายสัญญาณยาวเกินไป ระดับแรงดันไฟฟ้าจะตกลงเกินขอบเขตที่กำหนด นอกจากนี้ ความจุไฟฟ้าที่เกิดขึ้น จะมีผลกับคุณภาพของสัญญาณ โดยการทำให้การเปลี่ยนสถานะจาก แรงดันไฟบวกไปเป็นแรงดันไฟลบนั้นไม่ชัดเจน เนื่องจาก RS-232C ไม่ได้ออกแบบให้นำไปใช้กับระยะทางไกล ดังนั้นถ้าอุปกรณ์อยู่ห่างกันมาก อาจจำเป็นต้องใช้โมเดมหรือวิธีการอื่นๆแทน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
หลักการออกแบบและสร้าง

ในส่วนของเทอมนี้เราได้ทำการออกแบบและสร้างในส่วนของ ชุดควบคุมกล้องและส่วนของประมวล และจะได้อธิบายดังต่อไปนี้

4.1 ชุดควบคุมกล้อง

ใช้ไมโครคอนโทรลเลอร์ เป็นตัวควบคุมชุดขับ stepping motor โดยวงจรขับจะใช้ ic เบอร์ ULN2803 เป็นตัวขับ ซึ่งภายในจะเป็นทรานซิสเตอร์ 8 ชุด ต่อแบบ darington ส่วนของ Stepping Motorที่ใช้จะมีขนาดเล็ก และหาซื้อได้ง่ายราคาถูก ซึ่งมีคุณสมบัติคือ ใช้ไฟเลี้ยง 12 โวลท์ และมี 1.8 ดีกรีต่อ 1 สเต็ป จำนวนสองตัว เพื่อทำงานในการหมุนทางด้านแนวซ้ายขวากับแนวบนล่าง ดังรูปในภาคผนวก

สำหรับโปรแกรมที่เอาไว้ควบคุมการทำงานของสเต็ปปิ้งมอเตอร์ทั้ง 2 ตัว ซึ่งโปรแกรมนี้จะถูกเก็บเอาไว้ที่ 89C51โปรแกรมแอสเซมบลี้หลักการออกแบบของเขียนโปรแกรมนี้และสามารถดูได้จาก Flowchart ในภาคผนวก

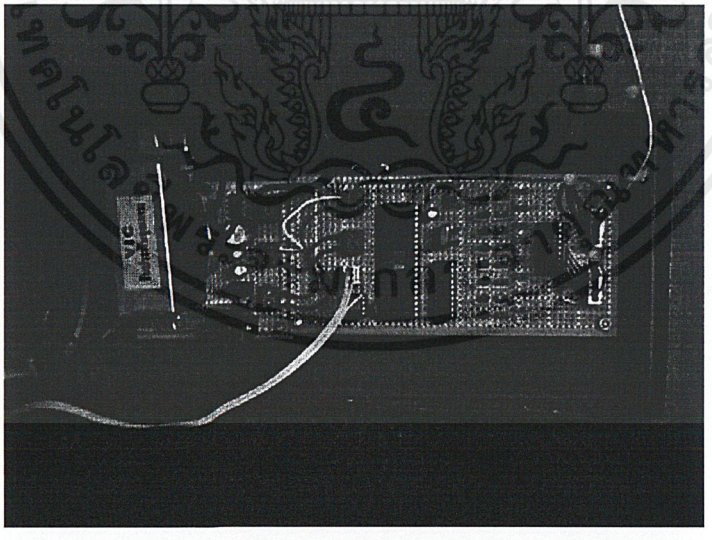
เริ่มจากการที่เราต้องกำหนดค่าเริ่มต้นของพารามิเตอร์ต่างๆที่จำเป็นในการกำหนดรูปแบบการทำงาน เช่น ความเร็วของสเต็ปปิ้งมอเตอร์ ตำแหน่งสูงสุดและต่ำสุดของการหมุน ตำแหน่งแอดเดรสต่างๆที่ใช้ในการเก็บข้อมูลของโปรแกรม เมื่อได้กำหนดค่าพารามิเตอร์ต่างๆแล้วก็จะเป็นส่วนของการรอรับค่าคอมพิวเตอร์โดยผ่านทาง RS232 ซึ่งเป็นพอร์ตอนุกรมการรับค่าจากคอมพิวเตอร์จะรอรับเพียงแค่ ไบต์เดียว ซึ่งจะเป็นรหัสที่ใช้ในการกำหนด ให้ 89C51 ทำงานตามรหัสที่ส่งมา ซึ่งจะมีทั้งหมด 4 รหัสด้วยกัน คือ 3Fh สำหรับการหมุนขวา 3Eh สำหรับการหมุนซ้าย 38h สำหรับการหมุนบน 3Ch สำหรับการหมุนล่าง 3Bh เมื่อได้รับรหัสที่ส่งมาให้แล้ว ก็จะทำการตรวจสอบว่าเป็นรหัสสำหรับทำอะไร ถ้ารหัสที่ได้รับมาถูกต้องตามที่ได้กำหนดไว้หรือเปล่า ถ้าถูกต้องก็จะทำการเข้าไปทำโปรแกรมย่อย ที่ใช้ในการเคลื่อนที่แต่ละส่วนของมอเตอร์ และในการทำงานในส่วนของโปรแกรมย่อยนี้ จะมีการตรวจสอบตำแหน่งของมอเตอร์ว่าอยู่ในขอบเขตที่กำหนดไว้ในตอนแรกหรือเปล่า ถ้ามากไป ก็จะทำการหยุดหมุน และในระหว่างการหมุน ก็จะทำการส่งตำแหน่งของมอเตอร์ กลับไปยังคอมพิวเตอร์ตลอดการหมุน Control Board ก็จะทำการหมุนไปเรื่อยๆจนกว่าจะได้รับคำสั่งให้หยุดหมุนจากคอมพิวเตอร์ ลักษณะโปรแกรมนี้จะเป็นโปรแกรมวนลูปไม่รู้จบ ทำไปเรื่อยๆจนกว่าจะมีการรีเซ็ทการทำงาน และในส่วนโปรแกรมย่อยที่ทำหน้าที่ส่งสัญญาณออกไปควบคุมการหมุนมอเตอร์ จะประกอบด้วยการทำงาน 2 ส่วนคือ การนำรูปแบบการหมุนของมอเตอร์แต่ละตัว (รูปแบบของการหมุนแตกต่างกัน เนื่องจากการใช้งานที่แตกต่างกัน) มาทำการประมวล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผล เพื่อให้ได้สัญญาณที่จะส่งออกไป ขับมอเตอร์ทั้ง 2 ตัว และอีกส่วนคือส่วนที่ คอยหน่วงเวลาสำหรับ การขับมอเตอร์ออกไป แต่ละสเต็ปของการหมุน ถ้าพูดง่ายๆก็จะเป็นการควบคุมความเร็วของการหมุนของมอเตอร์ทั้ง 2 ตัว เมื่อได้สัญญาณ และความเร็วในการหมุนที่แน่นอน ก็จะส่งข้อมูลออกไปที่พอร์ต 1 ของ 89c51 เมื่อได้ส่งสัญญาณไปแล้วก็จะทำการรายงานผลของการหมุนไปที่คอมพิวเตอร์อีกครั้งหนึ่ง โดยจะนำเอาข้อมูลของการหมุนที่เก็บเอาไว้ที่ตำแหน่งที่กำหนด ในตอนต้นโปรแกรมส่งออกไป โดยข้อมูลที่ส่งออกไปก็มี ตำแหน่งซ้ายขวา ตำแหน่งบนล่าง โดยผ่านทางพอร์ตอนุกรม RS-232 ไปยังคอมพิวเตอร์ และโปรแกรมที่คอมพิวเตอร์ก็จะนำข้อมูลเหล่านี้มาวิเคราะห์และแสดงผลต่อไป

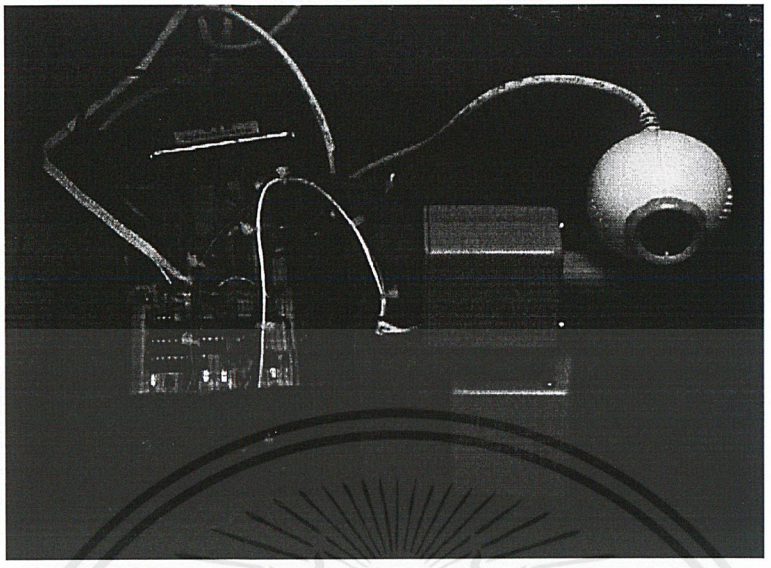
สำหรับในส่วนของคอมพิวเตอร์ที่เป็น Server Control จะใช้ PC ธรรมดาที่มีคุณสมบัติพอใช้ได้ เพื่อที่ความสามารถที่จะรันโปรแกรม Delphi 4 ได้อย่างไม่มีปัญหา

ในการทำงานของโปรแกรมนี้ หลักที่สำคัญคือรับข้อมูลจาก Control Board มาทำการประมวลผลและแสดงผล และรับคำสั่งจากผู้ใช้งาน ไปควบคุมการหมุนของมอเตอร์อีกที และที่สำคัญคือการแสดงผลภาพจากกล้องวีดีโอ ซึ่งรายละเอียดปลีกย่อยของโปรแกรมได้อยู่ในส่วนของภาคผนวกแล้ว



รูปที่ 4.1 ชุดควบคุมกล้อง(1)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูป 4.2 ชุดควบคุมกล้อง(2)

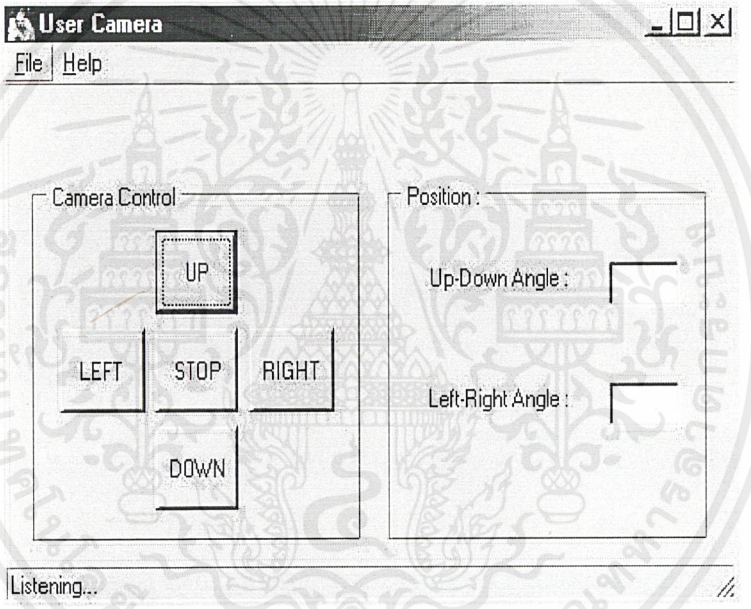
4.2 โปรแกรมการติดต่อใช้งานระหว่างผู้ควบคุมและผู้ให้บริการ

จากที่ได้อธิบายมา จะเป็นส่วนของการติดต่อระหว่างเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น Server กับตัวชุดจับกล้อง โดยจะผ่านทางพอร์ตอนุกรม RS232 ต่อไปก็จะเป็นการออกแบบในส่วนของการติดต่อระหว่างผู้ใช้ กับส่วนที่เป็น Server โปรแกรมที่ใช้ในโครงงานนี้ จะเป็นโปรแกรมในส่วนที่ติดต่อ กับผู้ใช้งานมีอยู่ 2 โปรแกรมคือโปรแกรมทางด้านผู้ควบคุม(ผู้ใช้งาน) และโปรแกรมทางด้านผู้ให้บริการ Server โดยจะเริ่มจากออกแบบโปรแกรมในส่วนที่เป็นผู้ใช้ก่อน โดยจะมีรายละเอียดดังต่อไปนี้

4.2.1 โปรแกรมควบคุมด้านผู้ใช้ เนื่องจากโปรแกรมที่ทางผู้จัดทำ ได้เขียนบนระบบปฏิบัติการ Windows 98 ดังนั้นโปรแกรมที่เรานำมาสร้าง โปรแกรมควบคุมจะใช้ โปรแกรม Delphi 4 ซึ่งสามารถทำงานบนระบบนี้ได้ เริ่มจากได้สร้างโปรแกรม โดยจะทำการออกแบบโครงสร้าง ของโปรแกรมนี้ก่อน โดยคิดจากฟังก์ชั่นการทำงานทั้งหมด ที่จำเป็นในการทำงาน ในการควบคุมการหมุนของกล้อง ที่อยู่ด้านปลายทางอีกที โดยมีฟังก์ชั่นการทำงานที่เห็นคือ ส่วนของการควบคุมทิศทางการหมุน การแสดงผลตำแหน่งของกล้อง และจะบอกความผิดพลาด ที่เกิดบนตัวชุดจับกล้องอันเนื่องมาจากสาเหตุ การที่ชุดจับกล้องหมุนเกินตำแหน่ง ที่ตั้งไว้ที่โปรแกรม โดยโปรแกรมได้ระยะตำแหน่ง ของการหมุนไว้เป็นดังนี้คือ หมุนได้ 60 องศา จากตำแหน่งจุดศูนย์กลาง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทั้ง 2 ตัว เมื่อการแจ้งการเกิดข้อผิดพลาดขึ้นมา ผู้ใช้ไม่สามารถที่จะสั่งไปในทิศทางนั้นได้ ต้องเปลี่ยนทิศทางการควบคุมให้มีทิศตรงกันข้ามจากเดิม และส่วนต่อมาก็จะเป็นส่วนของ โปรแกรมที่ใช้สำหรับติดต่อกับชุดจับกล้อง โดยชุดจับกล้องตัวนี้จะอยู่คนละที่กับด้านของผู้ใช้นี้ ซึ่งการติดต่อนี้จะควบคุมผ่านเครือข่ายอินเตอร์เน็ต โดยตัวโปรแกรมก็จะเป็นการส่งรหัสควบคุมการหมุนไปยังชุดควบคุมกล้องซึ่งอยู่ไกลออกไป และจะคอยรับข้อมูลที่จะบอกตำแหน่งปัจจุบันของมอเตอร์ ที่เราได้สั่งไป เมื่อได้กำหนดการทำงานทั้งหมดแล้ว ก็จะเป็นการเริ่มสร้างส่วนของโปรแกรมชุดนี้ โดยจะมีรูปแบบตามรูปข้างนี้



รูปที่ 4.3 แสดงการวางรูปแบบของโปรแกรมด้าน client

โดยทำการจัดรูปแบบของโปรแกรม ตามรูปที่4.1แล้วเขียนโปรแกรมต่างๆที่ได้กำหนดไว้ตามหน้าที่การใช้งานโดยส่วนประกอบที่สำคัญในส่วนนี้ก็คงเป็นตัว Client Socket และ Server Socket จะเป็นตัวทำงานในด้านอินเตอร์เน็ตทั้งหมด ส่วนชุดแป้นกดจะเป็นแบบชุดควบคุมทิศทางการหมุนของมอเตอร์ และสุดท้ายจะเป็นชุดแสดงผลตำแหน่งของมอเตอร์ ที่เราได้สั่งไป และได้ทำการเขียนโปรแกรมการทำงาน ซึ่งได้แสดงไว้ที่ภาคผนวกแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.2.2 โปรแกรมด้านให้บริการ

โดยในส่วนนี้จะเป็นส่วนที่เป็น ตัวจัดการเกี่ยวกับการควบคุม การรับส่งข้อมูล การจัดการเกี่ยวกับภาพ ที่จะส่งไปยังจุดหมายปลายทาง โดยจะมีชุดจับกล้องรวม อยู่ที่ส่วนนี้ด้วย แต่เราจะไม่พูดถึงในส่วนนี้ เพราะได้กล่าวมาแล้วในตอนต้น โดยจะกล่าวถึงโปรแกรมที่ใช้ควบคุมในส่วนนี้จะเป็นตัวคอยควบคุมติดต่อกับชุดจับกล้องโดยผ่านทาง พอร์ตอนุกรม RS232 และรับภาพจากกล้อง CCD โดยผ่านทางพอร์ต USB และสุดท้ายจะเป็นการรับคำสั่ง ควบคุมจากผู้ใช้ โดยโปรแกรมจะเริ่มจากการกำหนดลักษณะของหน้าที่การทำงานทั้งหมดอาจจะแบ่งหน้าที่การทำงานภายในโปรแกรมนี้ออกได้เป็น 3 ส่วนใหญ่ ได้แก่

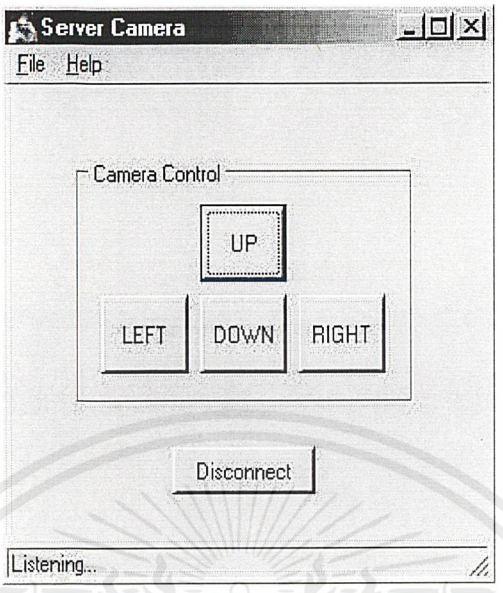
1) ส่วนการติดต่อกับชุดจับกล้อง โดยที่ชุดจับกล้องที่ได้ออกแบบจะติดต่อโดยผ่านทางพอร์ตอนุกรม RS 232 แต่ในการเขียนโปรแกรมโดยใช้ Delphi 4 นั้น Component ที่เกี่ยวกับ RS 232 นี้ไม่มีมาให้ในส่วนของโปรแกรมที่เป็นสแตนดาร์ดทั่วไป จึงได้ใช้ Component ที่ได้ Download มาและเมื่อนำมาทดลองใช้ก็สามารถทำงานในส่วนนี้ได้เป็นอย่างดี

2) ส่วนในการรับภาพจากกล้อง CCD ในส่วนนี้ถือว่าเป็นส่วนที่สำคัญมาก เพราะจุดประสงค์หลักของโครงงานนี้จะเกี่ยวกับการส่งภาพ และนี่ก็เป็นปัญหาที่คล้ายกับการโปรแกรมในส่วนแรกที่ได้กล่าวมาในในข้อที่แล้วคือ Component ที่ทำการรับภาพไม่มี สรุปก็คือต้องไปหา Download จากอินเตอร์เน็ตเหมือนที่ผ่านมา และเมื่อได้ Component นี้มาก็ทำการทดสอบการทำงาน ทั้งการรับภาพจากกล้องวีดีโอ และการบันทึกภาพเพื่อที่จะส่งต่อไป ปรากฏว่าทำงานได้ดังที่ตั้งใจเอาไว้

3) ส่วนในการติดต่อผ่านอินเตอร์เน็ต ก็จะใช้ Component ที่ได้มีไว้ให้แล้วใน Delphi4 โดยที่จะใช้เป็นตัว ClientSocket กับ SeverSocket ทั้งสองตัวนี้จะเป็นตัวทำหน้าที่ในการรับคำสั่งควบคุม และส่งข้อมูลตำแหน่งของมอเตอร์ไปยังผู้สั่งงาน การทำงานในส่วนนี้ก็สามารถทำงานได้ถูกต้อง

เมื่อได้เขียนโปรแกรมทั้งหมดแล้วในส่วนนี้ก็จะมีการวางรูปแบบดังรูปต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



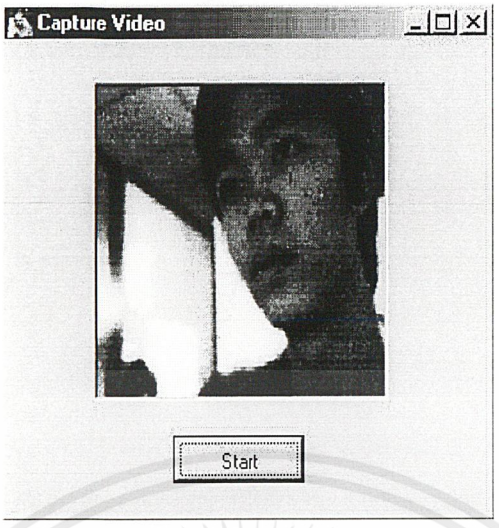
รูปที่ 4.4 โปรแกรมฝั่ง server

จากรูปที่ 4.2 จะเห็นได้ว่าจะมี Component ที่ถูกเพิ่มเข้ามาอยู่สองตัว คือ RS 232 และ IMAGEEN ซึ่งทั้งสองตัวนี้ไม่มีอยู่ในโปรแกรมที่เป็นมาตรฐานของ Delphi 4 จึงต้องทำการไป Download มา ในส่วนของตัวซอสโค้ดโปรแกรมได้แสดงไว้ในส่วนของภาคผนวกแล้ว

4.2.3 ส่วนส่งภาพ

หลักการคือใช้วิธีการจับภาพจากกล้องวีดีโอแล้วบันทึกไว้ในไฟล์ โดยจะกำหนดระยะเวลาในการจับภาพและบันทึกภาพ และจะใช้ไฟล์ html ที่มีการ refresh ภาพใหม่ในทุกๆช่วงเวลา โดยต้องทำเครื่อง server ให้เป็น web server เพื่อที่จะสามารถส่งไฟล์ html ไปได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.5 โปรแกรมจับภาพ

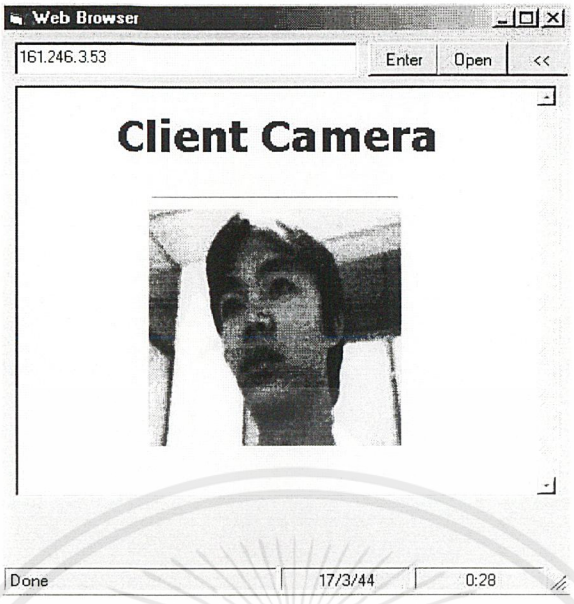
ในโปรเจ็คนี้จะใช้โปรแกรม PWS (Personal Web Server) เพื่อทำให้คอมพิวเตอร์เป็นเครื่อง Server โดยกำหนดให้ใช้ไฟล์ index.htm เป็นไฟล์ให้บริการ เมื่อคอมพิวเตอร์เครื่องอื่นจะขอใช้บริการ ก็จะเข้ามาที่ไฟล์นี้ ส่วนโปรแกรมที่ใช้ในการจับภาพและบันทึกภาพสร้างโดยโปรแกรม Delphi 4 โดยจะกำหนดการจับภาพและบันทึกภาพเป็นลูปทุกๆช่วงเวลา 1 วินาที

4.2.4 ส่วนรับภาพ

ใช้โปรแกรม Web browser ในการเรียกไฟล์ HTML จากเครื่อง Server มาแสดงบนเครื่อง client ในโปรเจ็คนี้จะใช้โปรแกรม Internet Explorer เป็นตัวแสดงไฟล์ index.htm และจะมีการ Upload ภาพใหม่ทุกๆ 1 วินาที

เมื่อมีการ connect จากเครื่อง client จะมีข้อความบอก client เครื่องอื่นว่าขณะนี้ ชุดควบคุมกล้องถูกใช้อยู่ และเมื่อมีการ disconnect เกิดขึ้น ก็จะมีข้อความบอกเครื่อง client เครื่องอื่นว่าขณะนี้สามารถเข้าใช้งานชุดควบคุมกล้องได้แล้ว โดยการติดต่อระหว่างเครื่อง client และ เครื่อง server จะติดต่อผ่านทาง Web Browser

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.6 โปรแกรม index.htm

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปผล

จากการที่ได้ลงมือสร้างโครงงานนี้ก็ได้ทดสอบคุณสมบัติต่างๆของโครงงานนี้ว่าได้ตามที่ออกแบบหรือคิดไว้หรือเปล่า โดยจะแยกผลการทดลองแต่ละส่วนเพื่อที่จะให้เห็นคุณสมบุติกันอย่างชัดเจน เริ่มที่ชุดควบคุมกล้องก่อน ผลการทดลองมีดังนี้

5.1 ชุดควบคุมกล้อง ซึ่งเป็นคล้ายๆแขนกล ทางผู้จัดทำได้ทำการออกแบบมาเฉพาะกล้องเท่านั้น โดยโครงสร้างทำมาจากพลาสติกเพราะมีราคาถูกและง่ายต่อการสร้างมาก ได้ทำตามที่ได้ออกแบบไว้ทุกอย่างซึ่งแสดงไว้ในรูปที่ 4.1 และ 4.2 ชุดควบคุมการหมุน ซึ่งประกอบไปด้วย Control Board ต่ออินเตอร์เฟซกับชุดขับมอเตอร์ จากการทดสอบโปรแกรมพบว่าการควบคุมสามารถทำได้ดี โดยควบคุมที่คอมพิวเตอร์และได้ตรวจสอบฟังก์ชั่นการทำงานต่างๆที่ได้ออกแบบไว้ได้อย่างครบถ้วน

5.2 ส่วนการติดต่อระหว่าง Server และ Client

5.2.1 การทำงานที่เครื่อง Server

โปรแกรมสามารถควบคุมการเคลื่อนไหวของกล้องได้อย่างราบเรียบและนุ่มนวล และสามารถรับ code จาก client มาทำงานได้อย่างถูกต้อง เมื่อ client disconnect ออกไป server ก็สามารถรับการ connect จาก client เครื่องอื่นได้ตามปกติ โดยก่อนการ disconnect ต้องทำการคืนตำแหน่งเดิมให้กับกล้องก่อน

5.2.2 การทำงานที่เครื่อง client

เมื่อโปรแกรมส่ง code ควบคุมให้กับ server ก็สามารถรับ feedback ได้ และแสดงองศาการหมุนของกล้องได้อย่างถูกต้อง

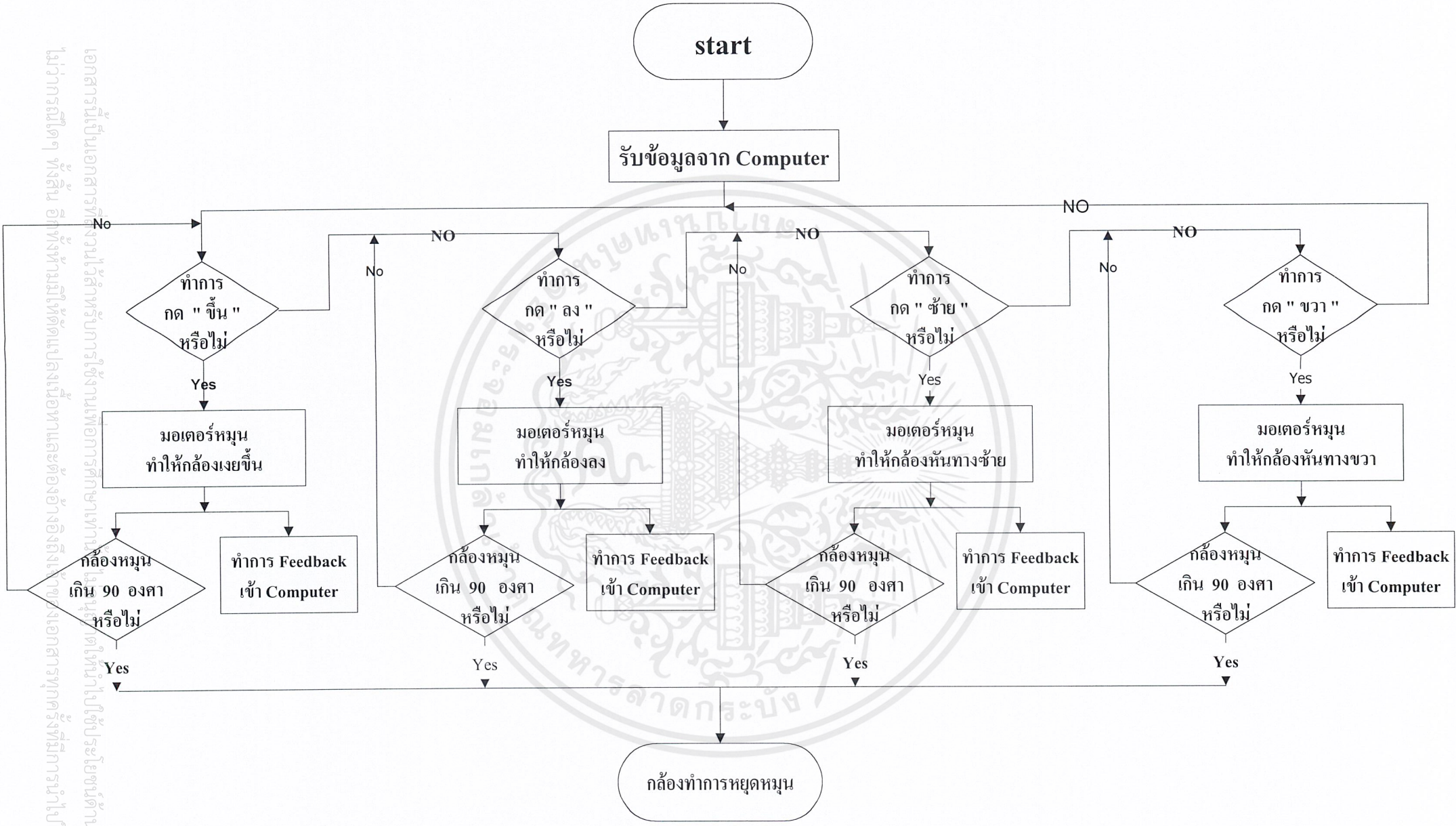
5.2.3 การรับภาพ

การรับภาพสามารถรับภาพได้ทุกๆ 1 วินาที แต่จะมีความผิดพลาดคือ ในภาพบางวินาทีจะไม่สามารถแสดงภาพได้ ซึ่งเนื่องมาจากการที่ไฟล์ index.htm จะไปดึงไฟล์ภาพมาจากเครื่อง server ซึ่งเป็นขณะเดียวกับที่โปรแกรมจับภาพทำการบันทึกภาพ จึงทำให้ไม่สามารถแสดงภาพได้

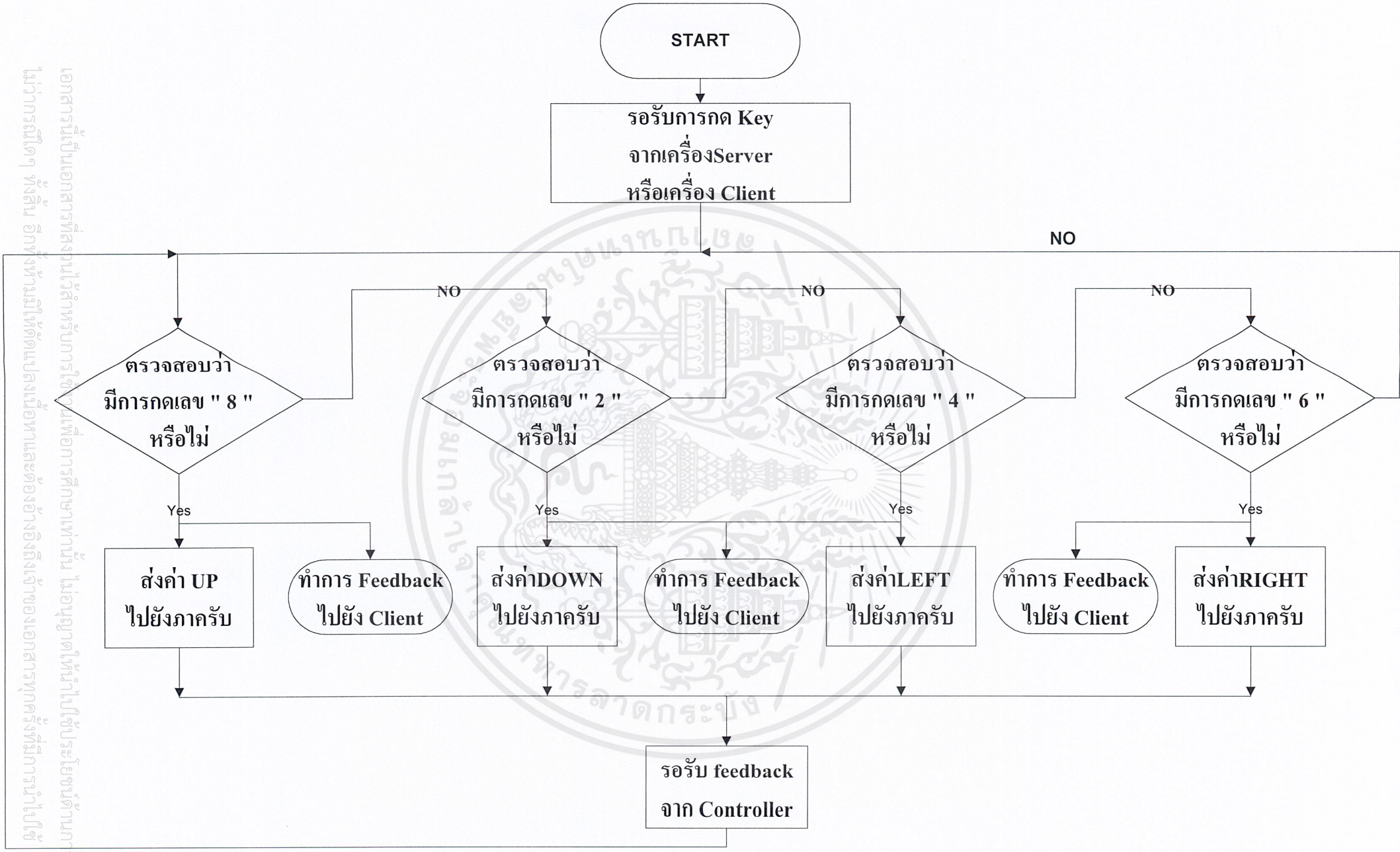
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

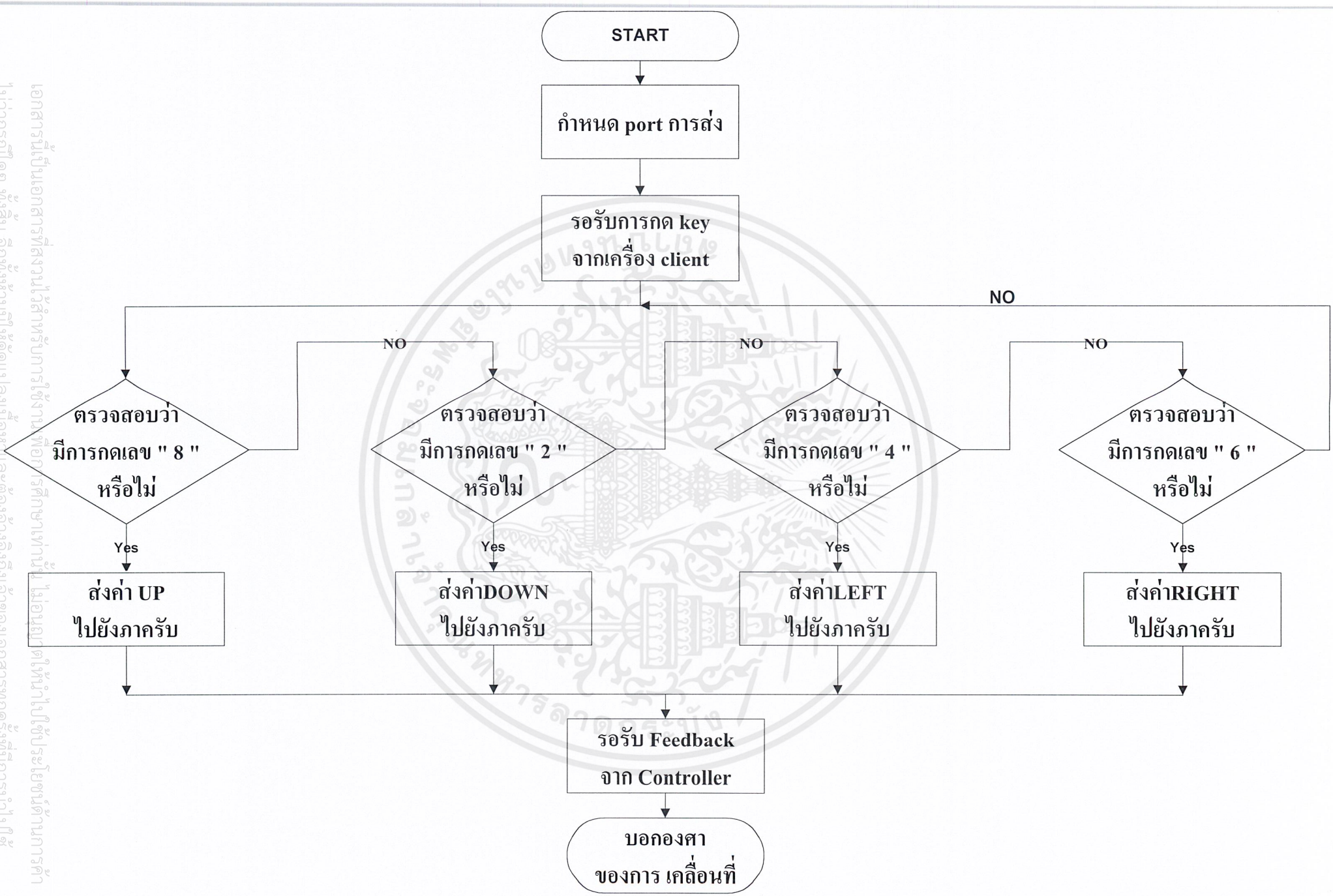
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



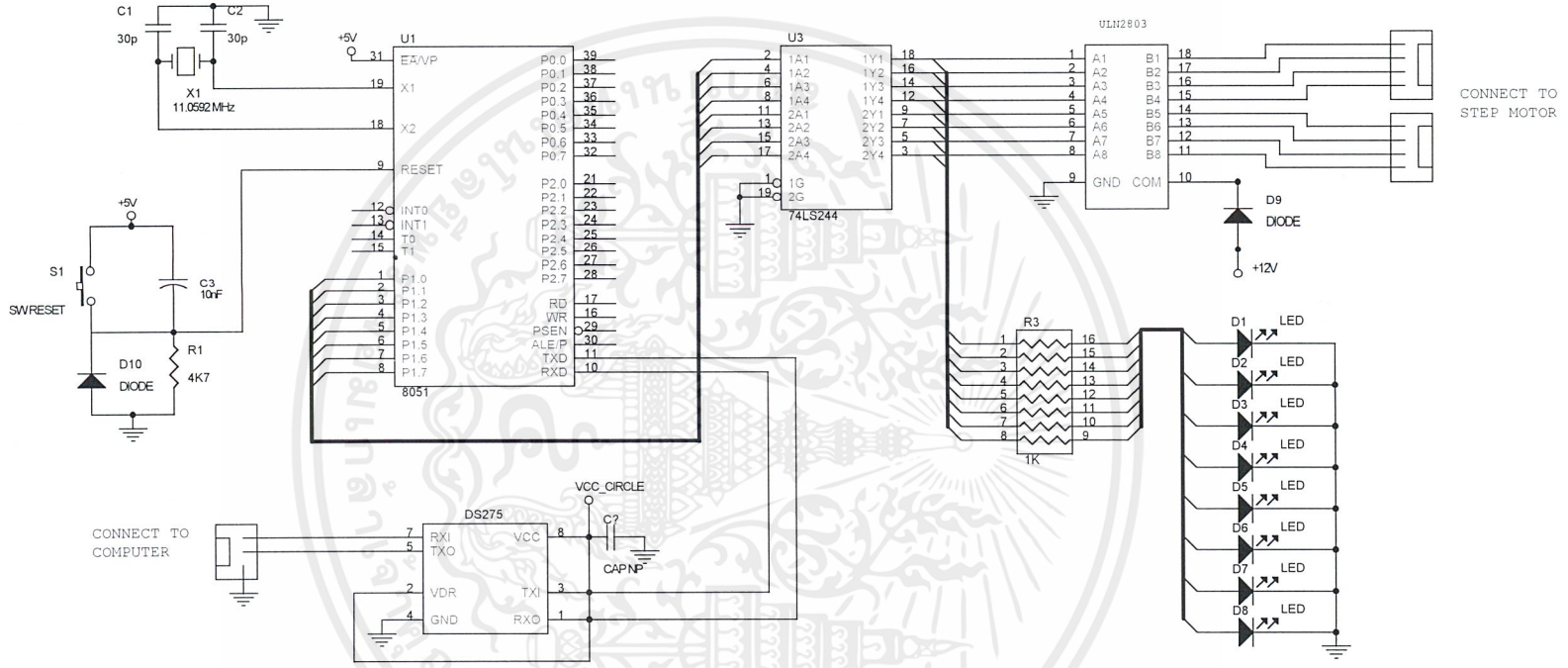
Flowchart แสดงการทำงานของภาครับ (micro controller)



Flowchart แสดงการทำงานของเครื่อง Server

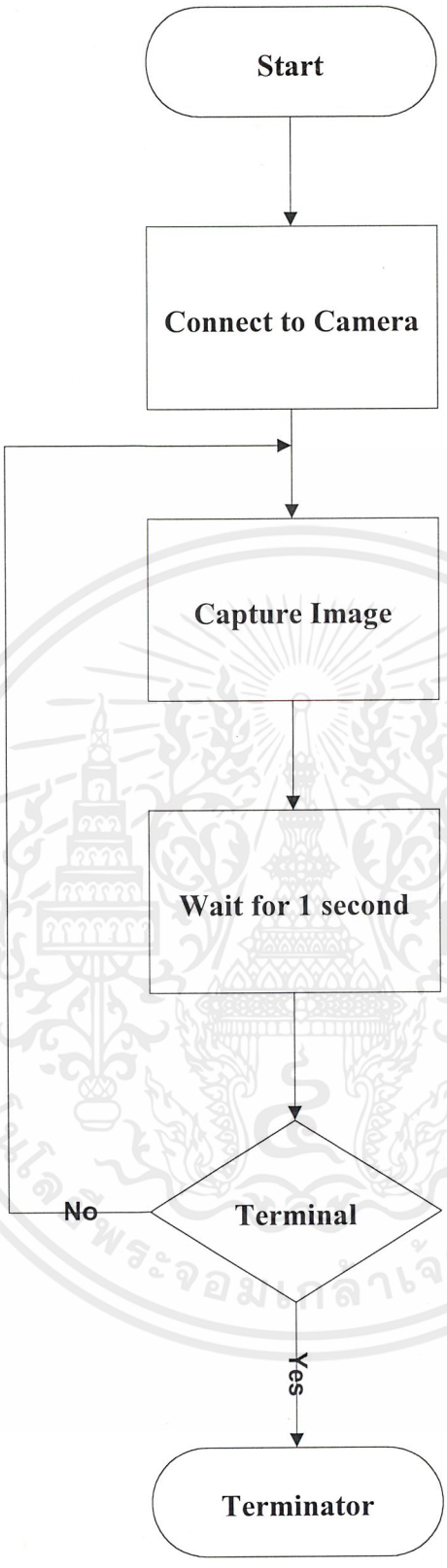


Flowchart แสดงการทำงานของเครื่อง Client



วงจรชุดควบคุมกล้อง

Title	<Title>		
Size B	Document Number <Doc>		Rev <RevCode>
Date: Friday, March 09, 2001	Sheet 1 of 1		



Block diagram โปรแกรมจับภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
UP              EQU   00111100B
DOWN            EQU   00111011B
LEFT            EQU   00111110B
RIGHT           EQU   10111111B
                ORG 0000H
                MOV P1,#11111111B     ;initial port

MAIN:           MOV PCON,#00H
                MOV SCON,#0D0H
                MOV TMOD,#20H
                MOV TH1,#0FDH
                SETB TR1
                MOV A,#0
                MOV R0,A
                MOV R1,A
                MOV R2,A
                MOV R3,#00001111B
                MOV R4,#11110000B
                MOV P1,#10011001B

WAIT:           SET REN
                CLR RI

WAIT1:          JNB RI,WAIT
                MOV A,SBUF
                CLR REN


ROTATE_UP:      CJNE A,#UP,ROTATE_DOWN
                MOV DPTR,#UPDOWN
                CJNE R0,#7,UPJ
                MOV R0,#0FFH
UPJ:            INC R0
                MOV A,R0
                MOVC A,@A+DPTR
                MOV R2,A
                MOV A,P1
                ANL A,R3
                ORL A,R2
                MOV P1,A
                ACALL DELAY_100mS
                AJMP WAIT

ROTATE_DOWN:    CJNE A,#DOWN,ROTATE_LEFT
                MOV DPTR,#UPDOWN
                CJNE R0,#0,DOWNJ
                MOV R0,#8
DOWNJ:          DEC R0
                MOV A,R0
                MOVC A,@A+DPTR
                MOV R2,A
                MOV A,P1
                ANL A,R3
                ORL A,R2
                MOV P1,A
                ACALL DELAY_100mS
                AJMP WAIT

ROTATE_LEFT:    CJNE A,#LEFT,ROTATE_RIGHT
                MOV DPTR,#LEFTRIGHT
                CJNE R1,#7,LEFTJ
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
                MOV R1,#0FFH
LEFTJ:          INC R1
                MOV A,R1
                MOVC A,@A+DPTR
                   MOV R2,A
                    MOV A,P1
                    ANL A,R4
                    ORL A,R2
                MOV P1,A
                ACALL DELAY_100mS
                AJMP WAIT

ROTATE_RIGHT:  CJNE A,#RIGHT,WAIT
                MOV DPTR,#LEFTRIGHT
                CJNE R1,#0,RIGHTJ
                MOV R1,#8
RIGHTJ:          DEC R1
                MOV A,R1
                MOVC A,@A+DPTR
                   MOV R2,A
                   MOV A,P1
                   ANL A,R4
                   ORL A,R2
                MOV P1,A
                ACALL DELAY_100mS
                AJMP WAIT

DELAY_100mS:   MOV R7,#100
DELAY_100mS_1:  MOV R6,#0E6H
DELAY_100mS_2:  NOP
                NOP
                DJNZ R6,DELAY_100mS_2
                DJNZ R7,DELAY_100mS_1
                JNB TI,$
                CLR TI
                MOV A,P1
                MOV SBUF,A
                RET


LEFTRIGHT: DB 09H,01H,03H,02H,06H,04H,0CH,08H
UPDOWN: DB 090H,080H,0C0H,040H,060H,020H,030H,010H
                 END
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unit pong_1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, Buttons, ScktComp, ExtCtrls, ComCtrls, jpeg, CPDrv;

type
  TServerForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    FileConnectItem: TMenuItem;
    FileListenItem: TMenuItem;
    StatusBar1: TStatusBar;
    Bevel1: TBevel;
    Panel1: TPanel;
    N1: TMenuItem;
    Disconnect1: TMenuItem;
    ServerSocket: TServerSocket;
    ClientSocket: TClientSocket;
    GroupBox1: TGroupBox;
    Upbutton: TButton;
    Leftbutton: TButton;
    Downbutton: TButton;
    Rightbutton: TButton;
    Disbutton: TButton;
    Edit1: TEdit;
    rs232: TCommPortDriver;
    Help1: TMenuItem;
    About1: TMenuItem;
    Image1: TImage;
    procedure FileListenItemClick(Sender: TObject);
    procedure FileConnectItemClick(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ServerSocketError(Sender: TObject; Number: Smallint;
      var Description: string; Scode: Integer; const Source,
      HelpFile: string; HelpContext: Integer; var CancelDisplay: Wordbool);
    procedure Disconnect1Click(Sender: TObject);
    procedure ClientSocketConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketClientRead(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ServerSocketClientConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
    procedure ServerSocketClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure UpbuttonClick(Sender: TObject);
    procedure LeftbuttonClick(Sender: TObject);
    procedure DownbuttonClick(Sender: TObject);
    procedure RightbuttonClick(Sender: TObject);
    procedure UpbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure DownbuttonKeyPress(Sender: TObject; var Key: Char);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    procedure LeftbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure RightbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure DisbuttonClick(Sender: TObject);
    procedure Edit1Change(Sender: TObject);
  protected
    IsServer: Boolean;
  end;

var
  ServerForm: TServerForm;
  Server,data: String;
  count1,count2: integer;

implementation

{$R *.DFM}

procedure TServerForm.FileListenItemClick(Sender: TObject);
begin
  FileListenItem.Checked := not FileListenItem.Checked;
  if FileListenItem.Checked then
  begin
    ClientSocket.Active := False;
    ServerSocket.Active := True;
    Statusbar1.Panels[0].Text := 'Listening...'
  end
  else
  begin
    if ServerSocket.Active then
      ServerSocket.Active := False;
    Statusbar1.Panels[0].Text := '';
  end;
end;

procedure TServerForm.FileConnectItemClick(Sender: TObject);
begin
  if ClientSocket.Active then ClientSocket.Active := False;
  if InputQuery('Computer to connect to', 'Address Name:', Server) then
    if Length(Server) > 0 then
      with ClientSocket do
      begin
        Host := Server;
        Active := True;
      end;
end;

procedure TServerForm.Exit1Click(Sender: TObject);
begin
  ServerSocket.Close;
  ClientSocket.Close;
  Close;
end;

procedure TServerForm.FormCreate(Sender: TObject);
begin
  FileListenItemClick(nil);
  data:='8';
  count1:=0;
  count2:=0;
  image1.Picture.LoadFromFile('C:\Inetpub\wwwroot\Not.bmp');
  Image1.Picture.SaveToFile('C:\Inetpub\wwwroot\connect.bmp');
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
end;

procedure TServerForm.ServerSocketError(Sender: TObject; Number: Smallint;
  var Description: string; Scode: Integer; const Source, HelpFile: string;
  HelpContext: Integer; var CancelDisplay: Wordbool);
begin
  ShowMessage(Description);
end;

procedure TServerForm.Disconnect1Click(Sender: TObject);
begin
  ClientSocket.Close;
  FileListenItemClick(nil);
end;

procedure TServerForm.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteHost;
end;

procedure TServerForm.ServerSocketClientRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  data:=socket.ReceiveText;
  if data='8'then edit1.Text:='8';
  if data='6'then edit1.Text:='6';
  if data='2'then edit1.Text:='2';
  if data='4'then edit1.Text:='4';
  if data='5'then edit1.Text:='5';
  if data='1'then edit1.Text:='1';
  if edit1.Text='8' then Upbutton.Click;
  if edit1.Text='2' then Downbutton.Click;
  if edit1.Text='4' then Leftbutton.Click;
  if edit1.Text='6' then Rightbutton.Click;
  if edit1.Text='5' then disbutton.Click;
  if edit1.Text<>'1' then edit1.Clear;
end;

procedure TServerForm.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  IsServer := True;
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteAddress;
end;

procedure TServerForm.ServerSocketClientConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  edit1.Clear;
  image1.Picture.LoadFromFile('C:\Inetpub\wwwroot\Used.bmp');
  Image1.Picture.SaveToFile('C:\Inetpub\wwwroot\connect.bmp');
end;

procedure TServerForm.ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  statusbar1.Panels[0].text:='Client Disconnect';
  FileListenItemClick(nil);
end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure TServerForm.ClientSocketError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  statusbar1.Panels[0].text:='Error connecting to : ' + Server;
  ErrorCode := 0;
end;

procedure TServerForm.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  ServerSocket.Active := False;
  FileListenItem.Checked := not FileListenItem.Checked;
  FileListenItemClick(nil);
end;

procedure TServerForm.UpbuttonClick(Sender: TObject);
var j,k:integer;
begin
 if (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8'))
then
   if count1<=70 then
    if rs232.Connect then
     if rs232.SendChar('8') then
      begin
       count1:=count1+1;
       for j:=0 to 20000 do
            for k:=0 to 800 do;
      end;
end;

procedure TServerForm.LeftbuttonClick(Sender: TObject);
var j,k:integer;
begin
  if (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8'))
then
   if count2<=70 then
     if rs232.Connect then
       if rs232.SendChar('4') then
         begin
          count2:=count2+1;
          for j:=0 to 20000 do
            for k:=0 to 600 do;
         end;
end;

procedure TServerForm.DownbuttonClick(Sender: TObject);
var j,k:integer;
begin
  if (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8'))
then
   if count1>=-70 then
    if rs232.Connect then
     if rs232.SendChar('2') then
      begin
       count1:=count1-1;
       for j:=0 to 20000 do
            for k:=0 to 800 do;
      end;
end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure TServerForm.RightbuttonClick(Sender: TObject);
var j,k:integer;
begin
  if (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8')) or (rs232.readchar('8'))
then
   if count2>=-70 then
     if rs232.Connect then
       if rs232.SendChar('6') then
        begin
         count2:=count2-1;
         for j:=0 to 20000 do
             for k:=0 to 700 do;
        end;
end;

procedure TServerForm.UpbuttonKeyPress(Sender: TObject; var Key: Char);
begin
     case key of
    '8' : UpButton.Click;
    '2' : DownButton.Click;
    '4' : LeftButton.Click;
    '6' : RightButton.Click;
   end;
end;

procedure TServerForm.DownbuttonKeyPress(Sender: TObject; var Key: Char);
begin
    case key of
    '8' : UpButton.Click;
    '2' : DownButton.Click;
    '4' : LeftButton.Click;
    '6' : RightButton.Click;
   end;
end;

procedure TServerForm.LeftbuttonKeyPress(Sender: TObject; var Key: Char);
begin
    case key of
    '8' : UpButton.Click;
    '2' : DownButton.Click;
    '4' : LeftButton.Click;
    '6' : RightButton.Click;
   end;
end;

procedure TServerForm.RightbuttonKeyPress(Sender: TObject; var Key: Char);
begin
    case key of
    '8' : UpButton.Click;
    '2' : DownButton.Click;
    '4' : LeftButton.Click;
    '6' : RightButton.Click;
   end;
end;

procedure TServerForm.DisbuttonClick(Sender: TObject);
var i,j,k:integer;
begin
   if count2>0 then
    for i:=0 to count2 do
     if rs232.Connect then
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
      if rs232.SendChar('6') then
         for j:=0 to 20000 do
           for k:=0 to 900 do;
   if count2<0 then
    for i:=0 downto count2 do
     if rs232.Connect then
       if rs232.SendChar('4') then
          for j:=0 to 20000 do
            for k:=0 to 900 do;
    if count1>0 then
    for i:=0 to count1 do
     if rs232.Connect then
       if rs232.SendChar('2') then
          for j:=0 to 20000 do
            for k:=0 to 900 do;
   if count1<0 then
    for i:=0 downto count1 do
     if rs232.Connect then
       if rs232.SendChar('8') then
          for j:=0 to 20000 do
            for k:=0 to 900 do;
    count1:=0;count2:=0;
end;

procedure TServerForm.Edit1Change(Sender: TObject);
begin
  if edit1.Text<>'1' then
   ServerSocket.Socket.Connections[0].SendText(edit1.Text)
  else
   begin
    rs232.Disconnect;
    serversocket.Close;
    clientsocket.Close;
    filelistenitem.Checked:=false;
    FileListenItemClick(nil);
    image1.Picture.LoadFromFile('C:\Inetpub\wwwroot\Not.bmp');
    Image1.Picture.SaveToFile('C:\Inetpub\wwwroot\connect.bmp');
   end;
end;

end.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unit TestClienC1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  Menus, StdCtrls, Buttons, ScktComp, ExtCtrls, ComCtrls;

type
  TClientForm = class(TForm)
    MainMenu1: TMainMenu;
    File1: TMenuItem;
    Exit1: TMenuItem;
    FileConnectItem: TMenuItem;
    FileListenItem: TMenuItem;
    StatusBar1: TStatusBar;
    Bevel1: TBevel;
    Panel1: TPanel;
    N1: TMenuItem;
    Disconnect1: TMenuItem;
    ServerSocket: TServerSocket;
    ClientSocket: TClientSocket;
    GroupBox1: TGroupBox;
    Upbutton: TButton;
    Leftbutton: TButton;
    Rightbutton: TButton;
    Downbutton: TButton;
    Button1: TButton;
    Help1: TMenuItem;
    About1: TMenuItem;
    GroupBox2: TGroupBox;
    UDangle: TEdit;
    LRangle: TEdit;
    Label1: TLabel;
    Label2: TLabel;
    Button2: TButton;
    procedure FileListenItemClick(Sender: TObject);
    procedure FileConnectItemClick(Sender: TObject);
    procedure Exit1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure ServerSocketError(Sender: TObject; Number: Smallint;
      var Description: string; Scode: Integer; const Source,
      HelpFile: string; HelpContext: Integer; var CancelDisplay: Wordbool);
    procedure Disconnect1Click(Sender: TObject);
    procedure ClientSocketConnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketRead(Sender: TObject; Socket: TCustomWinSocket);
    procedure ServerSocketAccept(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure ClientSocketError(Sender: TObject; Socket: TCustomWinSocket;
      ErrorEvent: TErrorEvent; var ErrorCode: Integer);
    procedure ServerSocketClientDisconnect(Sender: TObject;
      Socket: TCustomWinSocket);
    procedure UpbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure UpbuttonClick(Sender: TObject);
    procedure DownbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure DownbuttonClick(Sender: TObject);
    procedure LeftbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure LeftbuttonClick(Sender: TObject);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    procedure RightbuttonKeyPress(Sender: TObject; var Key: Char);
    procedure RightbuttonClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button1KeyPress(Sender: TObject; var Key: Char);
    procedure Button2Click(Sender: TObject);
  protected
    IsServer: Boolean;
  end;

var
  ClientForm: TClientForm;
  Server: String;
  data,udstr,lrstr:string;
  udcount,lrcount,upcount,downcount,leftcount,rightcount: real;
  initial: boolean;

implementation

{$R *.DFM}

procedure TClientForm.FileListenItemClick(Sender: TObject);
begin
  FileListenItem.Checked := not FileListenItem.Checked;
  if FileListenItem.Checked then
  begin
    ClientSocket.Active := False;
    ServerSocket.Active := True;
    Statusbar1.Panels[0].Text := 'Listening...'
  end
  else
  begin
    if ServerSocket.Active then
      ServerSocket.Active := False;
    Statusbar1.Panels[0].Text := '';
  end;
end;

procedure TClientForm.FileConnectItemClick(Sender: TObject);
begin
  if ClientSocket.Active then ClientSocket.Active := False;
  if InputQuery('Computer to connect to', 'Address Name:', Server) then
    if Length(Server) > 0 then
      with ClientSocket do
      begin
        Host := Server;
        Active := True;
      end;
  initial:=false;
end;

procedure TClientForm.Exit1Click(Sender: TObject);
begin
  ServerSocket.Close;
  ClientSocket.Close;
  Close;
end;

procedure TClientForm.FormCreate(Sender: TObject);
begin
  FileListenItemClick(nil);
  data:='8';
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
 upcount:=0;
 downcount:=0;
 leftcount:=0;
 rightcount:=0;
 udcount:=0;
 lrcount:=0;
end;

procedure TClientForm.ServerSocketError(Sender: TObject; Number: Smallint;
  var Description: string; Scode: Integer; const Source, HelpFile: string;
  HelpContext: Integer; var CancelDisplay: Wordbool);
begin
  ShowMessage(Description);
end;

procedure TClientForm.Disconnect1Click(Sender: TObject);
begin
  Button2.Click;
  //ClientSocket.Close;
  //FileListenItemClick(nil);
end;

procedure TClientForm.ClientSocketConnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteHost;
end;

// Feedback Num of key
procedure TClientForm.ClientSocketRead(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  data:=socket.ReceiveText;
  //if data='5' then disconnect1.Click;
  if data='8' then
   if ((upcount-downcount)<70) then upcount:=upcount+1;
  if data='2' then
   if ((downcount-upcount)<69) then downcount:=downcount+1;
  if data='4' then
   if ((leftcount-rightcount)<70) then leftcount:=leftcount+1;
  if data='6' then
   if ((rightcount-leftcount)<69) then rightcount:=rightcount+1;
  udcount:=(upcount-downcount)*0.9;
  lrcount:=(leftcount-rightcount)*0.9;
  str(udcount:1:2,udstr);
  str(lrcount:1:2,lrstr);
  UDangle.text:=udstr;
  LRangle.text:=lrstr;
end;

procedure TClientForm.ServerSocketAccept(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  IsServer := True;
  Statusbar1.Panels[0].Text := 'Connected to: ' + Socket.RemoteAddress;
end;

procedure TClientForm.ClientSocketDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  statusbar1.Panels[0].text:='Now Client Disconnect';
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
  FileListenItemClick(nil);
end;

procedure TClientForm.ClientSocketError(Sender: TObject;
  Socket: TCustomWinSocket; ErrorEvent: TErrorEvent;
  var ErrorCode: Integer);
begin
  statusbar1.Panels[0].text:='Eror connect to :'+Server;
  ErrorCode := 0;
end;

procedure TClientForm.ServerSocketClientDisconnect(Sender: TObject;
  Socket: TCustomWinSocket);
begin
  ServerSocket.Active := False;
  FileListenItem.Checked := not FileListenItem.Checked;
  FileListenItemClick(nil);
end;

procedure TClientForm.UpbuttonKeyPress(Sender: TObject; var Key: Char);
begin
   case key of
    '8' : Upbutton.Click;
    '2' : Downbutton.Click;
    '4' : Leftbutton.Click;
    '6' : Rightbutton.Click;
    '5' : button1.Click;
   end;
end;
// send Up signal
procedure TClientForm.UpbuttonClick(Sender: TObject);
begin
   ClientSocket.Socket.SendText('8');
end;

procedure TClientForm.DownbuttonKeyPress(Sender: TObject; var Key: Char);
begin
  case key of
    '8' : Upbutton.Click;
    '2' : Downbutton.Click;
    '4' : Leftbutton.Click;
    '6' : Rightbutton.Click;
    '5' : button1.Click;
   end;
end;

// Send Down signal
procedure TClientForm.DownbuttonClick(Sender: TObject);
begin
   ClientSocket.Socket.SendText('2');
end;

procedure TClientForm.LeftbuttonKeyPress(Sender: TObject; var Key: Char);
begin
    case key of
     '8' : Upbutton.Click;
     '2' : Downbutton.Click;
     '4' : Leftbutton.Click;
     '6' : Rightbutton.Click;
     '5' : button1.Click;
    end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
end;

// Send Left signal
procedure TClientForm.LeftbuttonClick(Sender: TObject);
begin
  ClientSocket.Socket.SendText('4');
end;

procedure TClientForm.RightbuttonKeyPress(Sender: TObject; var Key: Char);
begin
  case key of
   '8' : Upbutton.Click;
   '2' : Downbutton.Click;
   '4' : Leftbutton.Click;
   '6' : Rightbutton.Click;
   '5' : button1.Click;
  end;
end;

// Send Right signal
procedure TClientForm.RightbuttonClick(Sender: TObject);
begin
  ClientSocket.Socket.SendText('6');
end;

// return status and Disconnect
procedure TClientForm.Button1Click(Sender: TObject);
begin
  ClientSocket.Socket.SendText('5');
  Initial:=true;
end;

procedure TClientForm.Button1KeyPress(Sender: TObject; var Key: Char);
begin
  if key='5' then button1.Click;
end;

procedure TClientForm.Button2Click(Sender: TObject);
begin
 if initial then
 begin
 upcount:=0;
 downcount:=0;
 leftcount:=0;
 rightcount:=0;
 udcount:=0;
 lrcount:=0;
 ClientSocket.Socket.SendText('1');
 LRangle.Text:='0.00';
 UDangle.Text:='0.00';
 initial:=false;
 upbutton.SetFocus;
 end
 else
   Showmessage('You must click initial before disconnect');
end;

end.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
// Program Capture Image//

unit Capture Image;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Buttons, uSpnThrd, ImageEnView, VideoCap, ImageEnIO;

type

  TFSaveVideo = class(TForm)
    BitBtn1: TBitBtn;
    Video: TImageEnVideoView;
    ImageSV: TImageEnIO;
    procedure BitBtn1Click(Sender: TObject);
    procedure FormCloseQuery(Sender: TObject; var CanClose: Boolean);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  FSaveVideo: TFSaveVideo;

implementation

{$R *.DFM}

procedure TFSaveVideo.BitBtn1Click(Sender: TObject);
begin

   TSpinLine.create(video,ImageSV);
end;

procedure TFSaveVideo.FormCloseQuery(Sender: TObject;
  var CanClose: Boolean);
begin

   video.ShowVideo:=false; canclose:=true;
end;

end.
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unit uSpnThrd;

interface

uses
  Classes, Graphics, Windows, extctrls, ImageEnView, VideoCap, ImageEnIO, SysUtils;



type


  TSpinLine = class(TThread)
  public
    { Public declarations }
    constructor Create(var Video1:TImageEnVideoView;var ImageSV1:TImageEnIO);

  private
    { Private declarations }
    Video2 : TImageEnVideoView;
    ImageSV2 : TImageEnIO;
    procedure DoSaveVideo;
    procedure DoVideoThread;
  protected
    procedure Execute; override;
  end;

implementation

//- Save Routine ------------------------------------------------------------


procedure TSpinLine.DoSaveVideo;
var
   i,j : integer;
   fs : Tfilestream;
begin
   if not( terminated ) then
    if deletefile('C:\Inetpub\wwwroot\1.bmp') then
    begin
    //fs:=Tfilestream.Create('1.txt',fmcreate or fmopenwrite );
    Video2.frozen:=true;
    ImageSV2.SaveToFile('C:\Inetpub\wwwroot\1.bmp');
    Video2.frozen:=false;
    end;
    for i:=1 to 20000 do
     for j:=1 to 15000 do;
end;


//-----------------------------------------------------------------------------

constructor TSpinLine.Create(var video1:TImageEnVideoView;var ImageSV1:TImageEnIO);
begin
  Video2:=Video1;ImageSV2:=ImageSV1;
  Video2.ShowVideo :=true;
  FreeOnTerminate := True;
  inherited Create(False);
end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure TSpinLine.Execute;
begin
  DoVideoThread;
end;

procedure TSpinLine.DoVideoThread;
begin
  repeat
    DoSaveVideo;
    sleep(20);
  until ( Terminated );
end;

end.
```


เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

รายงานฉบับนี้สำเร็จลงได้ด้วยความช่วยเหลืออย่างดียิ่งจากหลายๆฝ่าย ซึ่งผู้จัดทำใคร่ขอขอบคุณทุกๆท่านที่มีส่วนร่วมสนับสนุน ช่วยเหลือและแนะนำในทุกๆด้าน

ขอขอบพระคุณ รศ. อิทธิชัย อรุณศรีแสงไชย อาจารย์ที่ปรึกษาโครงงานนี้ที่ได้เสียสละเวลา ให้คำปรึกษาและข้อเสนอแนะที่เป็นประโยชน์ ตลอดจนห้องทดลองและการทำงาน ให้การทำโครงงานชิ้นนี้สำเร็จลุล่วงได้ด้วยดี

ขอกราบขอบพระคุณบิดา มารดา ผู้ให้โอกาสและคอยให้กำลังใจเสมอมา

ขอขอบคุณเพื่อนๆ ห้อง A-404 ที่ช่วยเหลือให้คำปรึกษาและให้กำลังใจเสมอมา

สุดท้ายขอขอบพระคุณภาควิชาอิเล็กทรอนิกส์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่สนับสนุนการทำโครงงานในครั้งนี้

คุณค่าและประโยชน์อันพึงมีจากรายงานฉบับนี้ ผู้จัดทำขอมอบแด่ผู้มีพระคุณทุกท่าน

ธนะพงษ์ จุลธรรมาศน์

นพดล วงศ์บุญนาค

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หนังสืออ้างอิง

1. ธีรวัฒน์ ประกอบผล, " การประยุกต์ใช้งานไมโครคอนโทรลเลอร์ " , ส่วนตำราสนับสนุนเทคนิคอุตสาหกรรม สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น)
2. Pat Bonner, " Networking Programming with Window Sockets " , Prentice-hall INC. , ISBN 0-13-230152, pp. 49-78
3. Markus Pope, " Programming Internet Protocol " ,Prima Publishing, ISBN 0-7615-0773-6 pp. 167-184

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้