แพลตฟอร์มลินุกซ์ฝังตัวสำหรับหุ่นยนต์แบบยานพาหนะไร้คนขับ

EMBEDDED LINUX PLATFORM FOR UNMANNED VEHICLE ROBOT

กีรติ ศรีนา

สถิตธรรม สังข์ทอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2553

ปริญญานิพนธ์ปีการศึกษา 2553

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง แพลตฟอร์มลินุกซ์ฝังตัวสำหรับหุ่นยนต์แบบยานพาหนะไร้คนขับ

EMBEDDED LINUX PLATFORM FOR UNMANNED VEHICLE ROBOT

ผู้จัดทำ

1. นายกีรติ ศรีนา รหัสนักศึกษา 50010136

2. นายสถิตธรรม สังข์ทอง รหัสนักศึกษา 50011616

อาจารย์ที่ปรึกษา

(ผศ. อภิเนตร อูนากูล)

แพลตฟอร์มลินุกซ์ฝังตัวสำหรับหุ่นยนต์แบบยานพาหนะไร้คนขับ

นายกีรติ ศรีนา 50010136
นายสถิตธรรม สังข์ทอง 50011616
ผศ.อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2553

บทคัดย่อ

โครงงานนี้เป็นโครงงานที่พัฒนาแพลตฟอร์ม (platform) โดยใช้ลินุกซ์ฝังตัว ซึ่งสามารถนำไปพัฒนาไปเป็นหุ่นยนต์แบบยานพาหนะไร้คนขับ โดยที่แพลตฟอร์มนี้จะประกอบไปด้วยซอร์ฟแวร์เฟรมเวิร์คแพลตฟอร์ม และฮาร์ดแวร์แพลตฟอร์ม โดยที่ผู้ใช้จะสามารถจัดการฮาร์ดแวร์ได้อย่างง่ายดายอาทิ เช่น การรับ อินพุต การส่ง เอาท์พุต ของ จีพีไอโอ (GPIO), อินเตอร์รัพท์, การสร้างสัญญาณพีดับเบิลยูเอ็ม (PWM), การสั่งการมอเตอร์ และเซอร์โวมอเตอร์, การรับค่าจากเซ็นเซอร์, การแสดงผลแอลซีดี (LCD) และรับอินพุตจากผู้ใช้งานผ่านทางหน้าจอสัมผัส (touch screen), มีลักษณะการพัฒนาส่วนประกอบต่างๆที่มีรูปแบบของเสียบแล้วเล่นได้เลย (plug and play) ผ่านพอร์ตยูเอสบี (usb) การติดต่อกับอุปกรณ์บันทึกโดยใช้ยูเอสบี หรือเอสดีการ์ด (SD-card), และการรับภาพจากกล้องเว็ปแคม (webcam) เป็นต้น โดยที่ใช้บอร์ดทดลอง Mini 2440 ของ Friendly Arm ที่ใช้ซีพียูอาร์ม 9 เป็นหน่วยประมวลผลการทำงานซึ่งมีประสิทธิภาพสูงในการทำงาน และแพลตฟอร์มนี้ใช้ลินุกซ์เป็นระบบปฏิบัติการ และซอร์ฟแวร์อื่นๆที่ฟรีซึ่งทำหน้าในการจัดการทรัพยากรต่างๆ โครงงานนี้ยังครอบคลุมถึงการพัฒนาหุ่นยนต์ตัวอย่างที่มีการใช้ความสามารถต่างๆของแพลตฟอร์มนี้ได้อย่างหลากหลาย รวมไปถึงการทำเอกสาร และคู่มือการพัฒนาเพื่อรวบรวมองค์ความรู้ที่จะเป็นประโยชน์ต่อผู้ที่สนใจต่อไป

EMBEDDED LINUX PLATFORM FOR UNMANNED VEHICLE ROBOT

Mr. Kirati	Srina	50010136
Mr. Sathittham	Sangthong	50011616
Asst.Prof. Apinetr	Unakul	Advisor

Academic Year 2010

ABSTRACT

This project is the project that develops embedded linux platform which can be developed to unmanned vehicle robot. This platform have software framework and users will can manage the hardware easily such as taking input and send output of GPIO , interrupt , building PWM signals , commanding motors and servo motors , taking value from sensor , showing information by graphic LCD , taking input from user by touch screen , development of component that have the format of plug and play by USB port , the connection with mass storage device by use USB or MMC and taking picture from webcam , etc. By using Friendly Arm-Mini2440 with Arm 9 CPU which have high effective in the work. This platform use Linux that is freeware operating system which manage resources. This project cover example robot which is development that has using various ability of platform , including doing document and application node for collect the knowledge which will be advantage to developer and persons that they take an interest this project.

กิตติกรรมประกาศ

โครงงานนี้ได้รับความรู้ ข้อคิดเห็น และความช่วยเหลืออย่างดียิ่งจาก ผศ.อภิเนตร อูนากูล อาจารย์ที่ปรึกษาโครงงานมาโดยตลอด ผู้พัฒนาขอกราบขอบพระคุณมา ณ ที่นี้

ขอขอบคุณนายไพฑูรย์ ชัยโชคอนันต์(พี่ตู่) ที่ให้ความรู้และคำปรึกษา เกี่ยวกับวงจรอิเล็กทรอนิกส์ ลินุกซ์ และคำแนะนำในการพัฒนาโครงงาน

ขอขอบคุณนายสถาพร จันทมงคล ที่ให้ความช่วยเหลือทุกๆเรื่อง ทั้งคำปรึกษาในการทำหุ่นยนต์และด้านลินุกซ์สำหรับหุ่นยนต์ในการพัฒนาโครงงานนี้

ขอขอบคุณและขอบใจ ครอบครัวและเพื่อนๆของผู้พัฒนา ที่คอยให้กำลังใจและความช่วยเหลือในหลายๆด้าน จนโครงงานนี้ประสบความสำเร็จอย่างมาก

สุดท้ายนี้ขออุทิศความดีทั้งหมดที่เกิดจากโครงงานนนี้ให้แก่นางสาวคมขวัญ จรกิจ (ดิว หรือดาว) เพื่อนผู้ซึ่งเป็นที่รักตลอดกาลที่ได้จากโลกนี้ไปก่อนที่จะได้ร่วมโครงงานนี้ด้วยกัน และขอขอบคุณคุณพ่อ และคุณแม่น้องดาวที่ให้กำลังใจเสมอมา

กีรติ ศรีนา

สถิตธรรม สังข์ทอง

สารบัญ

หน้า

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญตาราง VI

สารบัญรูป VII

บทที่ 1 บทนำ 1

1.1 ความเป็นมาของปัญหา 1

1.2 วัตถุประสงค์ของโครงงาน 1

1.3 ประโยชน์ที่คาดว่าจะได้รับ 1

1.4 ขอบเขตของโครงงาน 2

1.5 วิธีการดำเนินงาน 2

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง 3

2.1 ความหมาย, การศึกษาตัวอย่าง และลักษณะโดยรวมของโครงงาน 3

2.2 ยูเอสบี (USB : Universal Serial Bus) 12

2.3 เอ็มเบดเดดลินุกซ์ 21

บทที่ 3 แนวทาง และการพัฒนาแพลตฟอร์ม 51

3.1 ภาพรวมของระบบ 51

3.2 การพัฒนาฮาร์ดแวร์ และซอฟแวร์ของแพลตฟอร์ม 52

3.3 กรณีศึกษาการออกแบบระบบ Coyote UAV ด้วย UML 62

3.4 การพัฒนาโมดูลต่างๆของหุ่นยนต์ให้เป็นอุปกรณ์แบบยูเอสบี 70

3.5 กรณีศึกษาวี-ยูเอสบี 78

สารบัญ (ต่อ)

หน้า

บทที่ 4 การทดลองและผลการทดลอง 92

4.1 การทดสอบบอร์ด 92

4.2 การทดลองโมดูลอุปกรณ์แบบยูเอสบี 92

4.3 การทดสอบแพลตฟอร์ม 93

4.4 การทดสอบการทำงานหุ่นยนต์ตัวอย่าง 97

บทที่ 5 บทวิจารณ์และสรุป 98

5.1 บทวิจารณ์ และสรุปผล 98

5.2 ปัญหาอุปสรรคและแนวทางแก้ไข 98

5.3 แนวทางการพัฒนาต่อ 98

บรรณานุกรม 99

ภาคผนวก ก. 102

ภาคผนวก ข. 137

ภาคผนวก ค. 138

สารบัญตาราง

ตาราง หน้า

2.1 เปรียบคุณลักษณะทางเน็ตเวิร์คของลินุกซ์ 27

3.1 ข้อมูลของ 54

3.2 เงื่อนไข ก่อน-หลัง ของส่วนติดต่อของ Perform Optical Surveillace 68

3.3 อุปกรณ์ที่ใช้ 79

3.4 อุปกรณ์ที่ใช้ 83

3.5 อุปกรณ์ที่ใช้ 87

3.6 CDC-IO Instruction Set 89

3.7 อุปกรณ์ที่ใช้ 91

4.1 การทดสอบจะใช้บอร์ด Mini 2440 92

4.2 การทดลองโมดูลอุปกรณ์แบบยูเอสบี 93

สารบัญรูป

รูป หน้า

2.1 หุ่นยนต์แบบฐานอยู่กับที่ไม่สามารถเคลื่อนที่ได้ 4
2.2 หุ่นยนต์เคลื่อนที่ได้แบบใช้ล้อ 4
2.3 หุ่นยนต์เคลื่อนที่ได้แบบใช้ขา 5
2.4 หุ่นยนต์อุตสาหกรรม 5
2.5 หุ่นยนต์ช่วยป้อนอาหาร 6
2.6 หุ่นยนต์ช่วยแพทย์ในการผ่าตัด (Da Vinci Surgical System) 6
2.7 หุ่นยนต์บินได้ขนาดเล็ก (Tiny Flying Robot) 7
2.8 หุ่นยนต์เพื่อการศึกษา (Programmeerbare Robot Met Bluetooth) 7
2.9 หุ่นยนต์สำรวจใต้น้ำ (Underwater Robotics at the Australian Centre for Field Robotics) 8
2.10 หุ่นยนต์เพื่อการบันเทิง (Sony's Latest Aibo Entertainment Robot) 8
2.11 Gladiator Tactical Unmanned Ground Vehicle 9
2.12 The Guardium UGV 10
2.13 MQ-9 Reaper unmanned aerial vehicle 10
2.14 cutlet Remotely operated underwater vehicle 11
2.15 Blackghost AUV 11
2.16 USB Logo 12
2.17 USB Topology : Tiered Star 17
2.18 Device Descriptors 19
2.19 แนวโน้มการใช้ Embedded OS 22
2.20 แนวโน้มอัตราการใช้งาน Linux 22
2.21 ส่วนประกอบของระบบลินุกซ์ 28
2.22 ตัวอย่างระบบฝังตัว 29
2.23 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 1 30
2.24 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 2 30
2.25 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 3 31
2.26 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 4 31
2.27 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 5 31
2.28 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 6 32

สารบัญรูป (ต่อ)

รูป	หน้า
2.29 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 8	32
2.30 การติดตั้งบอร์ด	33
2.31 การกำหนดค่าเริ่มต้นในส่วนพอร์ตอนุกรม	33
2.32 ขั้นตอนทำงานของบูทโหลดเดอร์	34
2.33 การโหลด ลินุกซ์เคอร์เนล	35
2.34 ใช้คำสั่ง flinfo	39
2.35 ใช้คำสั่ง printenv	39
2.36 เมื่อใช้คำสั่ง setenv	39
2.37 saveenv	40
2.38 ใช้คำสั่ง tftp	40
2.39 ใช้คำสั่ง erase	40
2.40 ใช้คำสั่ง cp	40
2.41 ใช้คำสั่ง ls-l	42
2.42 Code Hello World ใน 2.6 ,2.4	43
2.43 มองอุปกรณ์เป็นไฟล์	44
2.44 struct file operations	45
2.45 โครงสร้างหน่วยความจำแฟลชทั่วๆไปในระบบฝังตัว	46
2.46 สถาปัตยกรรมเอ็มทีดี	48
2.47 การเชื่อมต่ออินเตอร์รัพท์	49
3.1 ภาพรวมของระบบ	51
3.2 Deployment Diagram ภาพรวมของระบบ	51
3.3 บอร์ดทดลอง Friendly Arm Mini2440	53
3.5 ซอฟแวร์เลเยอร์ (Software Layer)	55
3.6 แผนที่และตำแหน่งที่ได้รับมอบหมาย	57
3.7 ลำดับการปฏิบัติภาระกิจ	58
3.8 ข้อมูลที่ได้จากการรันลิลุกซ์ผ่านโปรแกรมเทระเทิร์ม (Teara Term Web)	59
3.9 basic message passing	60
3.10 Class Diagram for Operational Programming Layer	61

สารบัญรูป (ต่อ)

รูป **หน้า**

3.11 Class Diagram for Tactical Programming Layer 61

3.12 Class diagram for main program use another class 62

3.13 ชั้นการทำงานของ TOPCASED 63

3.14 Use Cases Diagram ในชั้นบนสุด 63

3.15 Use Cases Diagram ในส่วนของ Execute Mission 64

3.16 Use Cases Diagram ในส่วนของ Fly UAV 64

3.17 ตัวอย่างโครงสร้าง Use Cases Diagram ในส่วนของ UAV Parametric 65

3.18 ตัวอย่างโครงสร้าง Use Cases Diagram ในส่วนของ QoS requirement 65

3.19 Use Cases Diagram ในส่วนของ CUAV Scenario1 66

3.20 Block Diagram ของ Perform Optical Surveillance 67

3.21 ส่วน interface ของ Perform Optical Surveillance 68

3.22 กิจกรรมของ Perform Optical Surveillance 69

3.23 state chart ของ Perform Optical Surveillance 70

3.24 วี-ยูเอสวี 71

3.25 วงจรตัวอย่างสำหรับวี-ยูเอสบี 72

3.26 แผนผังการพัฒนาวี-ยูเอสบี 72

3.27 apt-get update 74

3.28 Apt-get update 74

3.29 apt-get install build-essential gcc g++ 75

3.30 Qt SDK 75

3.31 Qt SDK Download 76

3.32 properties 76

3.33 Permissions 77

3.34 Nokia Qt SDK Setup 78

3.35 USBasp 78

3.36 วงจร USBasp 80

3.37 ISP pins 80

3.38 USBmot 82

สารบัญรูป (ต่อ)

รูป	หน้า
3.39 แบบวงจร USBmot	84
3.40 วงจร USBmot ที่ทดลองทำ	84
3.41 USBmot Host Software	85
3.42 คอมพิวเตอร์พอร์ตเสมือนด้วยซอฟแวร์ยูเอสบี	86
3.43 วงจร CDC-232 สำหรับ ATmega8/48/88-20	86
3.44 วงจร CDC -232 ที่ทดลองทำ	87
3.45 วงจร CDC -232 ที่ต่อกับ GPS	88
3.46 Hypo Terminal	88
3.47 CDC-IO สำหรับ ATmega8/48/88-20	90
3.48 วงจรตัวอย่าง	90
3.49 วงจรที่ใช้ทดสอบ	90
4.1 ภาพรวมของระบบรีคอนโรบอทที่ใช้ทดสอบ	93
4.2 ส่วนที่สนใจได้แก่ส่วนของรีคอนไนส์เส้นซ์โรบอทและเบสสเตชั่นบางส่วน	94
4.3 ส่วนติดต่อผู้ใช้งานที่ Base Station	95
4.4 หุ่นยนต์ตัวอย่าง Reconnaissance Robot	96
4.5 หน้าจอของโปรแกรมส่วนของผู้ติดต่อกับผู้ใช้ในขณะทดสอบ	97
ก.1 Friendly Arm Mini2440 Board	102
ก.2 การตั้งค่าโปรแกรม PuTTy	103
ก.3 การลง USB Driver: Yes, This time Only	103
ก.4 การลง USB Driver: Install from a list or specific location (Advanced)	104
ก.5 การลง USB Driver: Choose your search and installation options	104
ก.6 การลง USB Driver: Searching for SEC S3C2410X Test B/D	105
ก.7 การลง USB Driver: Hardware Installation	105
ก.8 การลง USB Driver: Completing the Found New Hardware Wizard	106
ก.9 USB:OK	106
ก.10 Backup Nand Flash	107
ก.11 Backup: USB Port>Transmit/Restore	107
ก.12 Restore NAND Flash	108

สารบัญรูป (ต่อ)

รูป หน้า

ก.13 Restore: USB Port>Transmit/Restore 108
ก.14 เข้า root user ด้วย คำสั่ง su 109
ก.15 ดาวโหลด cross tool chain และแตกไฟล์ 109
ก.16 แตกไฟล์ cross tool chain เสร็จ 109
ก.17 เพิ่ม path ของ cross compile 110
ก.18 ทดสอบ path ที่เพิ่มไป 110
ก.19 ดึง Source Code uboot 110
ก.20 กำหนดตัวแปร CROSS_COMPILE 111
ก.21 ทำการ compile uboot 111
ก.22 Compile uboot เสร็จแล้วจะได้ u-boot.bin 111
ก.23 เลือก q จาก vivi menu 112
ก.24 upload ไฟล์ u-boot.bin ด้วย dwn.exe 112
ก.25 nand scrub 113
ก.26 nand createbbt 113
ก.27 dynenv set 40000 114
ก.28 ดึง Source code Kernel 115
ก.29 แก้ไขไฟล์ drivers/video/fbmem.c 115
ก.30 สร้าง .config สำหรับ mini2440 115
ก.31 แก้ไขไฟล์ ../kernel-bin/.config 116
ก.32 สร้าง Kernel 116
ก.33 สร้าง uImage 116
ก.34 Copy lib modules ที่สร้างไว้แล้วไปเก็บไว้ยัง directory ที่ต้องการ 117
ก.35 เชื่อมต่อกับ Host ที่ /mnt/sdb2 117
ก.36 สร้าง kernel 117
ก.37 เตรียม file system พื้นฐานของ debian 118
ก.38 เตรียม sd card เพื่อถ่าย file system 118
ก.39 fdisk เพื่อสร้าง partition 119
ก.40 สร้าง partition 119

สารบัญรูป (ต่อ)

รูป หน้า

ก.41 Format sdb1 เป็น FAT และ sdb2 เป็น ext3 120
ก.42 แตกไฟล์ File System 120
ก.43 copy uImage(kernel) และ lib modules ไปยัง partition1 และ 2 120
ก.44 เตรียมพร้อมก่อนการติดตั้ง Debain File System 121
ก.45 การ load uImage จาก sd card partition 1 และ connect rootfs จาก partition 2 121
ก.46 load uImage จาก sd card partition 1 และ connect rootfs จาก partition 2 122
ก.47 ติดตั้ง debootstrap 122
ก.48 เตรียมสิ่งที่จำเป็นสำหรับการบูท Debian File System 122
ก.49 กำหนด nameserver 123
ก.50 Reboot เข้า u-boot menu 123
ก.51 load uImage จาก fat32 partition ด้วยคำสั่ง fatload และรัน kernel 123
ก.52 mount file system 124
ก.53 คำสั่ง mkswap /dev/mmcblk0p3 125
ก.54 คำสั่ง free 125
ก.55 NetGear รุ่น WG111v3 126
ก.56 apt-get update 126
ก.57 เสียบ usb wifi adapter 126
ก.58 lsusb 127
ก.59 ifconfig -a 127
ก.60 การ config ระบบเพื่อเพิ่ม Driver 128
ก.61 Wireless LAN (IEEE 802.11) 128
ก.62 สร้าง uImage และสร้าง Modules 129
ก.63 depmod -a 129
ก.64 ข้อความที่ปรากฏที่ console หลังจากรู้จัก usb wifi adapter 130
ก.65 ยืนยันด้วยคำสั่ง ifconfig -a 131
ก.66 การเซ็ทอัพ usb wifi adapter 132
ก.67 iwconfig 133
ก.68 ต่อเข้ากับระบบ network ด้วยคำสั่ง dhclient wlan0 134

สารบัญรูป (ต่อ)

รูป **หน้า**

ก.69 การปลด eth0 ออกจากระบบและใช้ wlan0 135
ก.70 แก้ไขไฟล์ /etc/network/interfaces 136
ค.1 UML Diagrams 138
ค.2 Class Diagram 139
ค.3 Component Diagram 139
ค.4 Composite Structure Diagram 140
ค.5 Deployment Diagram 140
ค.6 Object Diagram 141
ค.7 Package Diagram 141
ค.8 Activity Diagram 142
ค.9 State Machine Diagram 142
ค.10 Use-Cases Diagram 143
ค.11 Communication Diagram 143
ค.12 Sequence Diagram 144
ค.13 Timing Diagrams 144

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหา

เนื่องจากผู้จัดทำเป็นสมาชิกชุมนุมโรบอท คณะวิศวกรรมศาสตร์ มีประสบการณ์ในการพัฒนาหุ่นยนต์มามากมาย พบว่าหุ่นยนต์ที่ถูกพัฒนาขึ้นในประเทศไทยส่วนใหญ่เป็นหุ่นยนต์อย่างง่ายไม่ค่อยมีความซับซ้อน ความสามารถไม่สูงนัก ครั้นจะพัฒนาหุ่นยนต์ที่มีความสามารถมาก มีความซับซ้อนสูงนั้น ก็มีค่าใช้จ่ายที่สูงตามความซับซ้อนขึ้นมามาก โครงงานนี้จึงมุ่งเน้นในการพัฒนาแพลตฟอร์มที่มีประสิทธิภาพ เพื่อลดความซับซ้อนในการพัฒนา ลดค่าใช้จ่าย และสามารถใช้แพลตฟอร์มของโครงงานนี้ในการพัฒนาหุ่นยนต์ได้ง่ายมากยิ่งขึ้น โดยโครงงานนี้ได้มุ่งเน้นไปที่การพัฒนาแพลตฟอร์มสำหรับหุ่นยนต์แบบยานพาหนะไร้คนขับ

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อสร้างแพลตฟอร์มที่สามารถนำไปใช้ในการพัฒนาหุ่นยนต์แบบยานพาหนะไร้คนขับได้ง่ายมากยิ่งขึ้น ลดเวลา รวมทั้งค่าใช้จ่ายในการพัฒนา และสามารถนำไปใช้งานได้จริง
2) เพื่อสร้างเอกสารและคู่มือการใช้งานแพลตฟอร์มที่ได้จากโครงงานนี้ เพื่อประโยชน์แก่ผู้ที่สนใจนำไปพัฒนาหุ่นยนต์แบบยานพาหนะไร้คนขับต่อไป

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1) แพลตฟอร์มที่สามารถนำไปใช้ในการพัฒนาหุ่นยนต์แบบยานพาหนะไร้คนขับได้ง่ายยิ่งขึ้น ลดเวลา รวมทั้งค่าใช้จ่ายในการพัฒนา และสามารถนำไปใช้งานได้จริง
2) ผู้ที่สนใจในการพัฒนาหุ่นยนต์แบบยานพาหนะไร้คนขับ สามารถนำเอาเอกสารและคู่มือการใช้งานแพลตฟอร์มที่ได้จากโครงงานนี้ไปใช้งานได้จริง

1.4 ขอบเขตของโครงงาน

โครงงานนี้จะศึกษาและพัฒนาแพลตฟอร์มที่ใช้ในการพัฒนาหุ่นยนต์แบบยานพาหนะไร้คนขับ ซึ่งแบ่งเป็น การพัฒนาทางด้านฮาร์ดแวร์ และการพัฒนาทางด้านซอฟแวร์

การพัฒนาทางด้านฮาร์ดแวร์ ครอบคลุมถึงการออกแบบโมดูลวงจรแบบยูเอสบี การออกแบบพีซีบี การจัดหาและติดตั้งอุปกรณ์อิเล็กทรอนิกต่างๆบนบอร์ดพีซีบี การทดสอบและแก้ไขอุปกรณ์

การพัฒนาทางด้านซอฟแวร์ ครอบคลุมถึงการออกแบบโครงสร้างซอฟแวร์โดยใช้ยูเอ็มแอล การพัฒนาดีไวซ์ไดร์เวอร์ การพัฒนาเฟิร์มแวร์ การสร้างไลบรารี่ การพัฒนาแอปพลิเคชั่นของโปรแกรม และการทำคลอสคอมไพล์

โครงงานนี้ยังครอบคลุมถึงการพัฒนาหุ่นยนต์แบบยานพาหนะไร้คนขับ เพื่อใช้เป็นตัวอย่างในการทดสอบแพลตฟอร์มว่าใช้งานได้จริง รวมไปถึงการทำเอกสารและคู่มือการใช้งานเพื่อรวบรวมองค์ความรู้ที่จะเป็น ประโยชน์ต่อผู้ที่สนใจต่อไป

1.5 วิธีการดำเนินงาน

แบ่งเป็นลำดับขั้นตอนดังต่อไปนี้

1) ศึกษา ทำความเข้าใจเกี่ยวกับเนื้อหา ความหมาย และองค์ประกอบต่างๆของโครงงาน
2) ศึกษาทฤษฎีที่เกี่ยวข้องกับโครงงาน และทำการทดลอง
3) ทำการวิเคราะห์ และออกแบบโครงงานตามจุดประสงค์ พร้อมทั้งวางแผนงานในการดำเนินงาน
4) สร้างและพัฒนาโครงงานตามแผนงานที่ได้กำหนดไว้
5) สรุปและวิเคราะห์ผลการดำเนินงาน
6) วางแผนการพัฒนาโครงงานในขั้นต่อไป
7) จัดทำเอกสาร

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ความหมาย, การศึกษาตัวอย่าง และลักษณะโดยรวมของโครงงาน

2.1.1 ความหมาย

หุ่นยนต์ (Robot) หมายถึง เครื่องจักรหรืออุปกรณ์ที่สามารถเคลื่อนไหวได้ โดยมีการทำงานจากโปรแกรมการตัดสินใจและสามารถปรับเปลี่ยนโปรแกรมการทำงาน ให้ทำงานได้หลากหลายหน้าที่เพื่อตอบสนองต่อข้อมูล หรือสัญญาณที่ได้จากสิ่งแวดล้อมสามารถใช้งาน หรือทำงานได้แทนมนุษย์ซึ่งอาจทำงานได้ด้วยตนเอง หรือทำงานตามลำดับการทำงานที่ได้มีการตั้งไว้ล่วงหน้า เช่น หุ่นยนต์คล้ายมนุษย์ หุ่นยนต์อุตสาหกรรม หุ่นยนต์รักษาความปลอดภัย หุ่นยนต์ทางการแพทย์ เป็นต้น

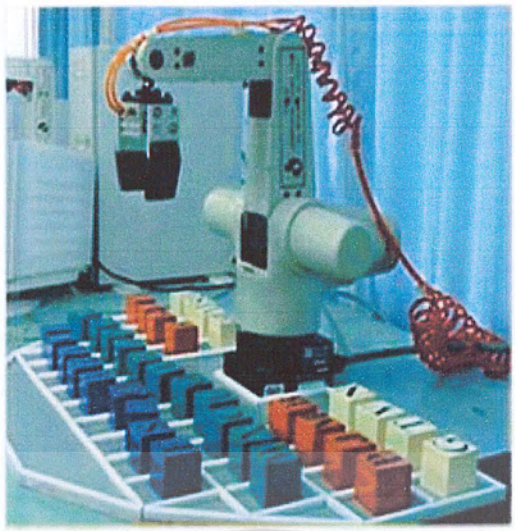
วิทยาการหุ่นยนต์ (Robotics) คือ ศาสตร์และเทคโนโลยีที่เกี่ยวข้องกับหุ่นยนต์ ทั้งส่วนการออกแบบ การผลิตและการประยุกต์ใช้งาน โดยวิทยาการหุ่นยนต์ต้องใช้ความรู้ทั้งทางด้านอิเล็กทรอนิกส์ การออกแบบเครื่องจักรกล การออกแบบระบบควบคุม และการพัฒนาโปรแกรมคอมพิวเตอร์

ระบบอัตโนมัติ (Automation) คือ ระบบที่ออกแบบด้วยกลไก อิเล็กทรอนิกส์ ไฟฟ้าและคอมพิวเตอร์ ให้สามารถทำงานได้ด้วยตนเอง ซึ่งมนุษย์อาจจะเกี่ยวข้องเพียงการกำหนดเงื่อนไขหรือเป้าหมายในการทำงาน ส่วนใหญ่เพื่อช่วยในกระบวนการผลิตในโรงงานอุตสาหกรรมและนำมาใช้งานแทนที่แรงงานมนุษย์

2.1.2 ประเภทของหุ่นยนต์

การแบ่งประเภทหุ่นยนต์นั้นขึ้นอยู่กับหัวข้อการแบ่งประเภท เช่นถ้าแบ่งตามสถานะของการเคลื่อนที่จะสามารถแบ่งออกเป็น 2 ประเภท

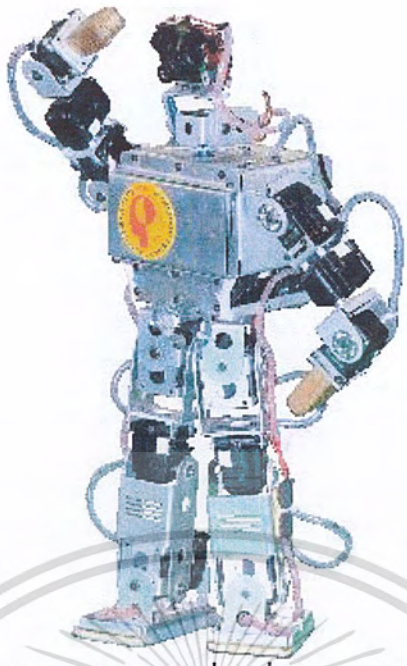
1) หุ่นยนต์แบบฐานอยู่กับที่ไม่สามารถเคลื่อนที่ได้ หุ่นยนต์ประเภทนี้จะมีฐานยึดติดกับที่ไม่สามารถเคลื่อนที่ หรือย้ายตำแหน่งได้ส่วนใหญ่จะเป็นหุ่นยนต์ที่มีลักษณะเป็นแขนกลหรือหุ่นยนต์อุตสาหกรรมที่ใช้งานในการหยิบจับและเคลื่อนย้ายชิ้นงานดังรูป 2.1



รูป 2.1 หุ่นยนต์แบบฐานอยู่กับที่ไม่สามารถเคลื่อนที่ได้

2) หุ่นยนต์ที่สามารถเคลื่อนที่ได้ หุ่นยนต์ประเภทนี้สามารถเคลื่อนย้ายตำแหน่งโดยอาจใช้ล้อติดที่ฐาน หรือเคลื่อนที่โดยใช้ขา ดังแสดงได้ดังรูป 2.2 และ 2.3 ตามลำดับ

รูป 2.2 หุ่นยนต์เคลื่อนที่ได้แบบใช้ล้อ



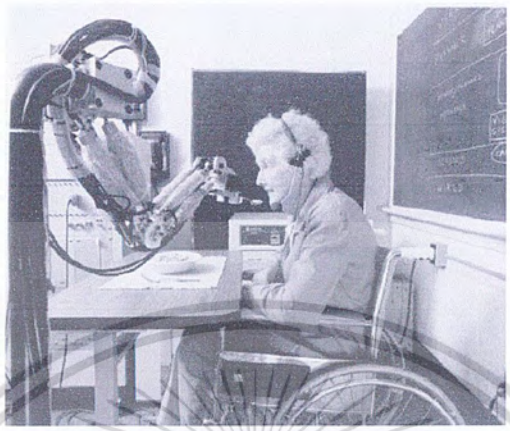
รูป 2.3 หุ่นยนต์เคลื่อนที่ได้แบบใช้ขา

ถ้าแบ่งตามประเภทการใช้งาน จะสามารถแบ่งออกได้เป็น

1) หุ่นยนต์อุตสาหกรรม หุ่นยนต์ประเภทนี้ได้แก่แขนกลที่ใช้ในโรงงานอุตสาหกรรม ส่วนใหญ่ใช้แทนแรงงานคนในงานด้านการเคลื่อนย้ายสิ่งของ การเชื่อม การพ่นสี เป็นต้น ตัวอย่างแสดงดังรูป 2.4 หุ่นยนต์ประเภทนี้สามารถยกสิ่งของที่มีขนาดหนักได้ ทำงานได้รวดเร็วและมีความแม่นยำสูง

รูป 2.4 หุ่นยนต์อุตสาหกรรม

2) หุ่นยนต์บริการ หุ่นยนต์ประเภทนี้เน้นช่วยมนุษย์ในเรื่องบริการ อำนวยความสะดวก ทั้งในสำนักงานและบ้าน เช่นหุ่นยนต์ประชาสัมพันธ์ หุ่นยนต์ป้อนอาหารแสดงดังรูป 2.5 เป็นต้น



รูป 2.5 หุ่นยนต์ช่วยป้อนอาหาร

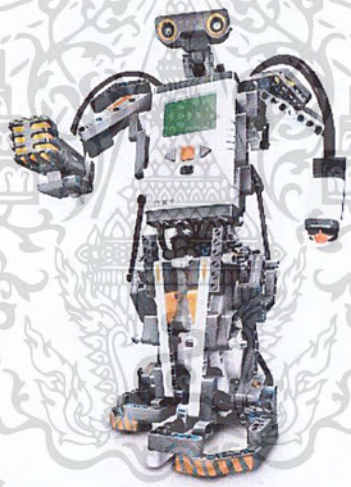
3) หุ่นยนต์ใช้ในการแพทย์ หุ่นยนต์ประเภทนี้ช่วยอำนวยความสะดวกทางการแพทย์ เริ่มตั้งแต่หุ่นยนต์ช่วยเรื่องกายภาพบำบัด การเดิน การหยิบของให้ผู้ป่วย ตลอดจนช่วยแพทย์ในการผ่าตัดแสดงดังรูป 2.6 เป็นต้น

รูป 2.6 หุ่นยนต์ช่วยแพทย์ในการผ่าตัด (Da Vinci Surgical System)

4) หุ่นยนต์ใช้ในการทหาร หุ่นยนต์ประเภทนี้ช่วยการทหารในส่วนของการสอดแนม ในลักษณะของหุ่นยนต์บินได้ขนาดเล็กแสดงดังรูป 2.7

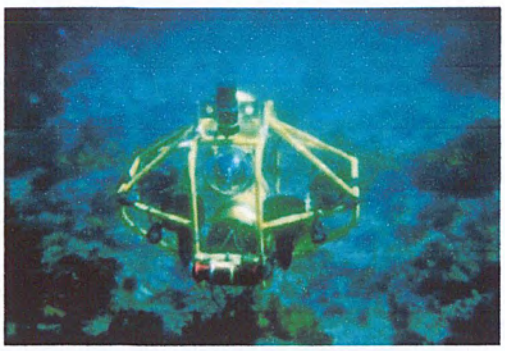
รูป 2.7 หุ่นยนต์บินได้ขนาดเล็ก (Tiny Flying Robot)

5) หุ่นยนต์เพื่อการศึกษา หุ่นยนต์ประเภทนี้ช่วยในการศึกษาพื้นฐานการทำงานของหุ่นยนต์ ทั้งในเรื่องส่วนประกอบด้านทางกล อิเล็กทรอนิกส์ และคอมพิวเตอร์ ดังแสดงในรูป 2.8



รูป 2.8 หุ่นยนต์เพื่อการศึกษา (Programmeerbare Robot Met Bluetooth)

6) หุ่นยนต์เพื่อการสำรวจ หุ่นยนต์ประเภทนี้ถูกใช้ในการสำรวจเก็บข้อมูล ซึ่งมีทั้งบนดิน บนอากาศหรือใต้น้ำ แสดงดังรูป 2.9



รูป 2.9 หุ่นยนต์สำรวจใต้น้ำ

7) หุ่นยนต์เพื่อการบันเทิง หุ่นยนต์ประเภทนี้ได้รับการพัฒนาให้สามารถตอบโต้กับคนได้เสมือนเป็นเพื่อนเล่นหรือสัตว์เลี้ยง ซึ่งมีในรูปแบบของสุนัข แมว และแมลง เป็นต้น ตัวอย่างแสดงได้ดังรูป 2.10

รูป 2.10 หุ่นยนต์เพื่อการบันเทิง (Sony's Latest Aibo Entertainment Robot)

จะเห็นได้ว่าหุ่นยนต์ และระบบอัตโนมัติได้เข้ามามีบทบาทมากขึ้น ไม่จำกัดเพียงหุ่นยนต์ที่ใช้ในโรงงานอุตสาหกรรมซึ่งเน้นการเพิ่มผลผลิตเท่านั้น แต่ขยายการใช้งานในส่วนของการเพิ่มคุณภาพชีวิตของคนให้ดีขึ้น เช่นการบริการอำนวยความสะดวกภายในบ้านและสำนักงาน ซึ่งอยู่ในรูปแบบหุ่นยนต์ดูดฝุ่น หุ่นยนต์ตัดหญ้า เป็นต้น นอกจากนี้ยังมีหุ่นยนต์ที่ช่วยเหลือทางการแพทย์ การสำรวจ การรักษาความปลอดภัย ตลอดจนหุ่นยนต์เพื่อความบันเทิงในรูปแบบหุ่นยนต์สัตว์เลี้ยง หุ่นยนต์เพื่อการศึกษา หรือหุ่นยนต์ช่วยในการประชาสัมพันธ์ เป็นต้น

2.1.3 ยานพาหนะไร้คนขับ (Unmanned Vehicle)

ยานพาหนะไร้คนขับเป็นหุ่นยนต์ประเภทหนึ่งที่มีลักษณะคล้ายยานพาหนะ ถูกพัฒนาขึ้นมาใช้ทั้งทางด้านพลเรือนและการทหาร ทำงานแทนมนุษย์ในส่วนต่างๆ ทั้งที่เป็นส่วนที่เข้าถึงยาก เป็นอันตราย และช่วยขยายสมรรถภาพของมนุษย์ โดยแบ่งได้เป็น 3 ประเภทใหญ่ๆ ดังนี้

2.1.3.1 Unmanned Ground Vehicles (UGV)

เป็นยานพาหนะไร้คนขับภาคพื้นดินหรือ "รถยนต์อัจริยะไร้คนขับ" โดยทั่วไปสามารถแบ่งได้เป็น 2 ประเภท

1) Teleoperated UGV เป็นยานพาหนะไร้คนขับภาคพื้นดินที่ควบคุมด้วยมนุษย์ผ่านระบบสื่อสารทางไกล โดยการตัดสินใจจะขึ้นอยู่กับผู้ควบคุมหุ่นยนต์โดยอาจจะรับข้อมูลภาพวีดีโอจากกล้อง ค่าจากเซนเซอร์ต่างๆ ตัวอย่างหุ่นยนต์ประเภทนี้คือ หุ่นยนต์กู้ระเบิด

รูป 2.11 Gladiator Tactical Unmanned Ground Vehicle

2) Autonomous UGV เป็นยานพาหนะไร้คนขับภาคพื้นดินที่เป็นหุ่นยนต์ทำงานแบบอัตโนมัติ (Autonomous Robot) โดยที่มีความสามารถทำงานได้เองเมื่อไม่มีมนุษย์ควบคุม เรียนรู้ได้เองจากสภาพแวดล้อม เดินทางจากจุดหนึ่งไปยังอีกจุดหนึ่งได้เอง ตรวจจับวัตถุที่น่าสนใจ หรือแม้กระทั่งสามารถซ่อมตัวเองได้โดยไม่มีผู้ช่วย เป็นต้น

รูป 2.12 The Guardium UGV

2.1.3.1 Unmanned Aerial Vehicles (UAV)

เรียงอีกอย่างหนึ่งว่า Remotely piloted Vehicles (RPV) เป็นอากาศยานที่สามารถบินได้โดยที่ไม่มีมนุษย์อยู่บนเครื่อง หรือที่รู้จัก "เครื่องบินไร้คนขับ" โดยสามารถควบคุมการปฏิบัติการได้จากระยะที่ห่างไกล ส่วนใหญ่มักใช้ในการทหาร

รูป 2.13 MQ-9 Reaper unmanned aerial vehicle

สามารถแบ่งตามการทำงานได้ 6 ประเภท

1) เล็งเป้าและเหยื่อล่อ (Target and decoy)
2) ลาดตระเวน (Reconnaissance)
3) ต่อสู้ (Combat)
4) การขนส่ง (Logistics)
5) วิจัยและพัฒนา (Research and Development)
6) พลเมืองและโฆษณา (Civil and Commercial)

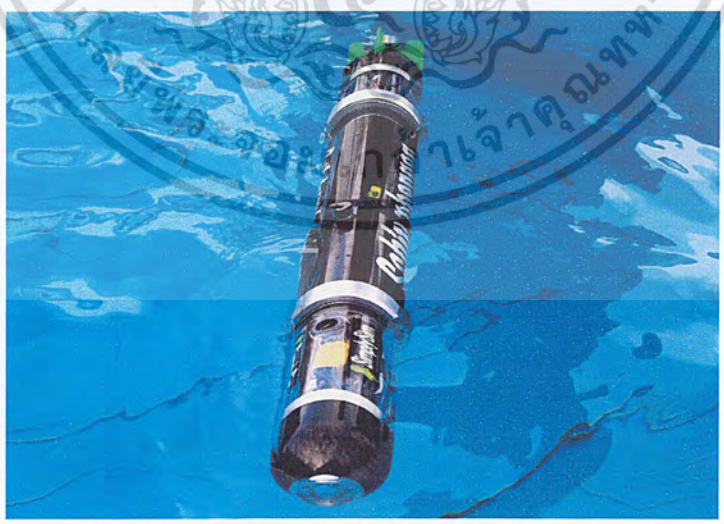
2.1.3.2 Unmanned Maritime Vehicles (UMV) หรือ Unmanned Underwater Vehicles (UUV)

ยานพาหนะไร้คนขับทางทะเล สามารถแบ่งได้เป็น 2 ประเภท คือ

1) Remotely Operated Underwater Vehicles (ROVs) เป็นยานพาหนะไร้คนขับทางทะเล ที่ควบคุมจากผู้ควบคุมที่อยู่บนบก

รูป 2.14 Cutlet Remotely Operated Underwater Vehicle

2) Autonomous Underwater Vehicles (AUVs) เป็นยานพาหนะไร้คนขับทางทะเล ที่สามารถทำงานได้เองอัตโนมัติ



รูป 2.15 Blackghost AUV

2.2 ยูเอสบี (USB : Universal Serial Bus)

2.2.1 ความเป็นมาของระบบบัส USB

รูป 2.16 USB Logo

ระบบบัส USB เป็นระบบบัสที่ถูกออกแบบมาให้มีความง่ายในการเชื่อมต่อ และทรงประสิทธิภาพในการสื่อสารข้อมูลกับอุปกรณ์หลายๆชนิด โดยปราศจากข้อจำกัด และการขัดขวางของการอินเทอร์เฟส (Interface) ทางด้านฮาร์ดแวร์ ในโลกของ USB จะไม่มีการใช้ดิพสวิตซ์ (Dip Switch) หรือแม้แต่จั๊มเปอร์ (Jumper) เพื่อการกำหนดค่าทางด้านฮาร์ดแวร์แต่อย่างใด ซึ่งการตั้งค่าการทำงานต่างๆจะถูกกระทำโดยระบบปฏิบัติการอย่างอัตโนมัติ นอกจากนี้การเชื่อมต่ออุปกรณ์ USB เข้ากับระบบยังสามารถทำในขณะที่เครื่องคอมพิวเตอร์ยังคงทำงานอยู่ได้ ซึ่งการใช้งานลักษณะนี้คือรูปแบบของระบบปลั๊กแอนด์เพลย์ (Plug and Play) อย่างแท้จริง และในส่วนของการเพิ่มจำนวนพอร์ตการสื่อสารข้อมูลก็สามารถทำได้อย่างง่ายดายด้วยการใช้ USB ฮับ (USB Hub) มาต่อพ่วงเข้ากับระบบ จุดเด่นที่น่าสนใจอีกประการหนึ่งของระบบบัสชนิดนี้ก็คือ ความเร็วในการส่งถ่ายข้อมูลซึ่งสูงถึง 480 เมกะบิตต่อวินาทีสำหรับ USB เวอร์ชั่น 2.0 ซึ่งสูงกว่าการเชื่อมต่อแบบขนานและอนุกรมเป็นอย่างมาก

2.2.2 จุดเด่นหลักๆ ของระบบบัส USB

1) ผู้ใช้สามารถนำอุปกรณ์ I/O มาต่อเข้ากับพอร์ต USB ในขณะที่เครื่องคอมพิวเตอร์กำลังทำงานอยู่ได้ (Hot-Pluggable)
2) ง่ายต่อการใช้งาน ซึ่งเครื่องคอมพิวเตอร์จะสามารถรู้จัก และจดจำอุปกรณ์ I/O ที่ถูกนำมาต่อเข้าไปในระบบโดยผ่านทางดีไวซ์ไดร์ฟเวอร์ (Device Driver) ที่เหมาะสม อีกทั้งการตั้งค่าต่างๆ จะเป็นไปอย่างอัตโนมัติ
3) ลดความสับสนของการเชื่อต่อด้วยการใช้คอนเน็คเตอร์ (Connector) เพียงชนิดเดียว

4) ประสิทธิภาพการส่งถ่ายข้อมูลด้วยการส่งถ่ายข้อมูลที่มีความเร็วถึง 480 เมกะบิตต่อวินาทีสำหรับอุปกรณ์แบบไฮสปีด (High Speed) ซึ่งมีค่าสูงกว่าพอร์ตขนานและอนุกรมหลายเท่า

5) สามารถต่ออุปกรณ์ได้ถึง 127 ตัว

6) สายเคเบิลที่ใช้เชื่อมต่อจะมีสายของแหล่งจ่ายไฟรวมอยู่ภายในตัวมันด้วย

7) มีการจัดการพลังงานที่ชาญฉลาด ซึ่งอุปกรณ์ต่างๆ จะมีการลดการใช้กำลังงานของตัวมันเองลงเมื่อไม่มีการใช้งาน

8) มีการตรวจจับและแก้ไขความผิดพลาดของข้อมูลอย่างอัตโนมัติ

2.2.3 การส่งข้อมูลภายในบัสยูเอสบี 1.0/1.1

ยูเอสบีเป็นการส่งข้อมูลและเชื่อมต่อในรูปแบบของบัส คือ อุปกรณ์ทุกๆ ตัวที่ต่อเชื่อมจะต้องส่งสัญญาณรวมกันไปในสายส่งสัญญาณเพียงคู่เดียว ดังนั้นอุปกรณ์ทุกๆ ตัวจะต้องส่งข้อมูลเรียงลำดับเพื่อป้องกันการชนกันของข้อมูล และทำให้ในช่วงเวลาหนึ่งๆ จะมีข้อมูลวิ่งไปได้เพียงทิศทางเดียวเท่านั้น

จังหวะการรับส่งข้อมูลของระบบบัสยูเอสบีทั้งหมดจะถูกควบคุมโดยโฮสต์ซึ่งก็คือตัว แพลตฟอร์มที่เป็นจุดรวมของอุปกรณ์ต่างนั่นเอง ดังนั้นจึงไม่สามารถเชื่อมต่อโฮสต์ 2 ตัวเข้าด้วยกันโดยตรงได้ เพราะจะทำให้เกิดการชนกันของข้อมูลภายในบัสเนื่องจากแต่ละโฮสต์ก็จะพยายามกำหนดจังหวะในการรับส่งของตนเองขึ้นมา ดังนั้นจึงต้องมีอุปกรณ์ตัวกลางเพื่อซิงโครไนซ์ตัวเองเข้ากับโฮสต์ ทั้ง 2 ฝั่งให้ได้

การรับส่งข้อมูลจะถูกกำหนดเป็นเฟรม (frame) โดยทุกๆ 1 มิลลิวินาทีจะเกิดการรับส่งข้อมูลขึ้น 1 เฟรมในแต่ละเฟรมจะถูกแบ่งแยกออกเป็นแพ็กเกจและเริ่มต้นการทำงานของแต่ละเฟรมโดยโฮสต์จะส่งสัญญาณเริ่มต้นเฟรม (SOF : start of frame) ออกไปให้อุปกรณ์ทุกตัวรู้จังหวะการเริ่มเฟรม หลังจากนั้นโฮสต์ก็จะเริ่มส่งหรือรับข้อมูลต่างๆ ตามที่ได้จัดลำดับความสำคัญไว้ อุปกรณ์ต่างๆ ที่อยู่ภายในบัสจะต้องทำตามจังหวะที่โฮสต์กำหนดไว้เท่านั้น การส่งข้อมูลกลับไปยังโฮสต์จะทำได้เมื่อได้รับการถามหรือร้องขอจากโฮสต์

แต่เนื่องจากแต่ละเฟรมจะต้องรับส่งข้อมูลให้เสร็จภายใน 1 มิลลิวินาทีนั่นหมายความว่าข้อมูลของอุปกรณ์ทุกๆ ตัวที่เชื่อมต่อกับบัสจะต้องถูกกำหนดขนาดไม่ให้ใหญ่เกินกว่าที่จะสามารถรับส่งได้ภายใน 1 มิลลิวินาทีและเล็กพอที่จะทำให้อุปกรณ์ทุกๆ ตัวสามารถใช้งานบัสไปพร้อมๆ กันได้ ดังนั้น ยูเอสบีจึงจำเป็นต้องอาศัยซอฟแวร์ที่เข้ามาจัดการในด้านนี้ และต้องอาศัยฮาร์ดแวร์ที่จะคอยกระจายการส่งและรวบรวมการรับข้อมูลจากทุกๆ อุปกรณ์ในระบบ ซึ่งซอฟแวร์และฮาร์ดแวร์ที่จำเป็นกับยูเอสบีมีดังนี้

2.2.3.1 ส่วนซอฟแวร์

1) ไดรเวอร์อุปกรณ์ยูเอสบี (USB device drivers)

2) ไดรเวอร์ยูเอสบี (USB driver)

3) ไดรเวอร์โฮสต์คอนโทรลเลอร์ (USB host controller driver)

2.2.3.2 ส่วน hardware

1) ยูเอสบีโฮสต์คอนโทรลเลอร์ (USB host controller) / รูทฮับ (root hub)

2) ยูเอสบีฮับ (USB hub)

3) อุปกรณ์ยูเอสบี (USB device)

2.2.4 ส่วนประกอบทาง Software

1) ไดรเวอร์อุปกรณ์ยูเอสบี คือ โปรแกรมเก็บข้อมูลที่จำเป็นในการติดต่อไปยังอุปกรณ์แต่ละตัว เมื่อโปรแกรมใดมีความต้องการจะติดต่อกับอุปกรณ์ต่างๆ จะต้องแจ้งความต้องการนั้นๆ มายังไดรเวอร์อุปกรณ์ยูเอสบี เนื่องจากตัวไดร์เวอร์นี้จะรู้ว่าถ้าต้องการติดต่ออุปกรณ์จะต้องติดต่อผ่านเอ็นพอยท์ (endpoint) ใดด้วยรูปแบบใด (การทำงานของอุปกรณ์ยูเอสบีจะติดต่อสั่งงานผ่านเอ็นพอยท์ของตัวอุปกรณ์ ซึ่งอุปกรณ์ต่างๆ จะมีชนิดและจำนวนเอ็นพอยท์ที่ต่างกัน) ดังนั้นอุปกรณ์แต่ละตัวจะต้องมีไดรเวอร์อุปกรณ์ยูเอสบีเฉพาะตัว เช่น ถ้าต้องการติดต่อกับอุปกรณ์คีย์บอร์ดตัวไดรเวอร์อุปกรณ์ยูเอสบีก็จะรู้ว่าต้องรับส่งข้อมูลแบบความเร็วต่ำ โดยใช้รูปแบบการถ่ายทอดข้อมูลแบบอินเตอร์รัพท์ผ่านเอ็นพอยท์ตัวหนึ่งของคีย์บอร์ดและตรวจสอบข้อมูลการกดเป็นช่วงๆ ระยะห่างค่าหนึ่งแต่ในบางอุปกรณ์พื้นฐานของเครื่องคอมพิวเตอร์ เช่น เมาส์, คีย์บอร์ด จะมีการบรรจุไดรเวอร์อุปกรณ์ยูเอสบีเหล่านั้นไว้ในไบออส (BIOS) ของคอมพิวเตอร์เรียบร้อยแล้วจึงไม่ต้องติดตั้งไดรเวอร์เพิ่มเติมสำหรับอุปกรณ์นี้

2) ไดรเวอร์ยูเอสบี การทำงานของยูเอสบีนั้นเป็นการต่อร่วมกันของอุปกรณ์หลายๆ ชนิดบนสายสัญญาณเพียงคู่เดียว ดังนั้นการส่งข้อมูลของอุปกรณ์แต่ละชนิดจะต้องมีการแบ่งสรรปันส่วนกันไปอย่างพอเหมาะพอดี เพื่อให้อุปกรณ์ทุกตัวสามารถทำงานไปได้พร้อมๆ กัน และแน่นอนว่าต้องมีซอฟแวร์ที่เข้ามาทำหน้าที่นี้ ซึ่งก็คือ ไดรเวอร์ยูเอสบีนั่นเอง ไดรเวอร์อุปกรณ์ยูเอสบีของอุปกรณ์แต่ละตัวจะส่งการร้องขอเพื่อการติดต่อ (request) ลงมายังไดรเวอร์ยูเอสบี และเมื่อไดรเวอร์ยูเอสบีรับทราบความต้องการการติดต่อของอุปกรณ์ครบทุกตัวที่เชื่อมอยู่บนบัสแล้ว ก็จะพิจารณาว่า ในรอบการรับส่งข้อมูลหนึ่งๆ นั้นอุปกรณ์แต่ละตัวสามารถรับส่งข้อมูลได้มากเท่าใด หากปริมาณข้อมูลที่ต้องการรับส่งมีขนาดใหญ่มากก็จะตัดแบ่งออกเป็นส่วนๆ แล้วเก็บไว้เพื่อรอส่งในรอบถัดไป โดยปริมาณข้อมูลที่ส่งได้ของอุปกรณ์แต่ละตัวจะถูกพิจารณาจากชนิดของการ

ถ่ายทอดข้อมูล (transfer type) ว่าอุปกรณ์ใดใช้การถ่ายทอดข้อมูลแบบใดและการรับส่งข้อมูลชนิดนั้นมีลำดับความสำคัญมากน้อยเพียงใด

3) ไดรเวอร์โฮสต์คอนโทรลเลอร์ หลังจากไดรเวอร์ยูเอสบีพิจารณาแล้วว่าอุปกรณ์แต่ละตัวส่งข้อมูลได้เท่าใดบ้าง มันจะส่งข้อมูลของอุปกรณ์แต่ละตัวที่จะติดต่อในรอบการติดต่อนั้นๆ มายังไดรเวอร์โฮสต์คอนโทรลเลอร์ จากนั้นไดรเวอร์โฮสต์คอนโทรลเลอร์จะจัดเรียงลำดับข้อมูลของอุปกรณ์แต่ละชนิดลงในเฟรมข้อมูล เพิ่มเติมส่วนประกอบต่างๆ ของเฟรมข้อมูลให้ครบตามมาตรฐานการถ่ายทอดข้อมูลแบบยูเอสบีแล้วส่งข้อมูลทั้งหมดไปยัง ฮาร์ดแวร์ยูเอสบี โฮสต์คอนโทรลเลอร์เพื่อส่งข้อมูลทั้งหมดออกไปยังอุปกรณ์ต่างๆ

2.2.5 ส่วนประกอบทางฮาร์ดแวร์

2.2.5.1 ยูเอสบีโฮสต์คอนโทรลเลอร์

มีหน้าที่สร้างสัญญาณข้อมูลทางไฟฟ้า แล้วส่งต่อไปยังรูทฮับ(root hub) เพื่อกระจายออกไปยังอุปกรณ์ต่างๆ โดยมันจะสร้างสัญญาณการติดต่อข้อมูลรูปแบบต่างๆตามที่ไดรเวอร์โฮสต์คอนโทรลเลอร์กำหนดมาให้ จากนั้นแปลงข้อมูลที่จะส่งจากแบบขนานเป็นแบบอนุกรมเพื่อใช้ในการส่งต่อไป เมื่อสัญญาณมาถึงรูทฮับ รูทฮับก็จะส่งสัญญาณนั้นออกไปยังบัสเพื่อส่งต่อไปยังอุปกรณ์ต่างๆนอกจากนั้นรูทฮับยังทำหน้าที่สำคัญอีก 4 อย่างคือ

1) ควบคุมการใช้พลังงานของอุปกรณ์ที่มาต่อ
2) ตรวจสอบการเชื่อมต่อของอุปกรณ์ว่ามีอุปกรณ์ต่ออยู่หรือไม่
3) เปิดการใช้งานพอร์ต (enable) เมื่อมีอุปกรณ์ต่ออยู่ และปิดการใช้งาน (disable) เมื่อปลดอุปกรณ์ออกไปแล้ว
4) รายงานสถานะของแต่ละพอร์ต เมื่อไดรเวอร์โฮสต์คอนโทรลเลอร์ร้องขอมา

2.2.5.2 ยูเอสบีฮับ

หน้าที่หลักๆคือ ขยายการเชื่อมต่อให้อุปกรณ์จำนวนมากๆ สามารถเชื่อมต่อเข้ากับระบบบัสได้ โดยการทำงานหลักมี 2 ส่วน คือ ทำหน้าที่เป็นตัวทวนสัญญาณ (repeater) และตัวจัดการพลังงาน (power management) ในส่วนการทวนสัญญาณ ยูเอสบีฮับจะต้องรับสัญญาณจากโฮสต์มา แล้วส่งกระจายออกไปยังพอร์ตทุกๆ พอร์ต และรับสัญญาณจากแต่ละพอร์ต แล้วจับมารวมกันเพื่อส่งกลับไปที่โฮสต์สำหรับส่วนของการจัดการพลังงานนั้นมีหน้าที่เหมือนกับรูทฮับก็คือ ตรวจสอบว่ามีการเชื่อมต่ออยู่ของอุปกรณ์ที่พอร์ตใดบ้าง หากมีอุปกรณ์ต่ออยู่ก็เปิดการใช้งานพอร์ตนั้นๆ หากไม่มีอุปกรณ์ต่ออยู่ก็ปิดการใช้งาน ตรวจสอบการเชื่อมต่อหรือปลดออกของอุปกรณ์เพื่อรายงานผลให้โฮสต์คอนโทรลเลอร์ร้องขอ และป้องกันอุปกรณ์ที่ต่ออยู่ในแต่ละพอร์ตไม่ให้ดึงกระแสไฟฟ้าเกินกว่าที่กำหนด

2.2.5.3 อุปกรณ์ยูเอสบี

อุปกรณ์ต่างๆ ที่เชื่อมต่อกับคอมพิวเตอร์ด้วยพอร์ตยูเอสบี นั่นเอง สามารถแบ่งเป็น 2 ชนิดตามความเร็วในการถ่ายทอดข้อมูลคือ

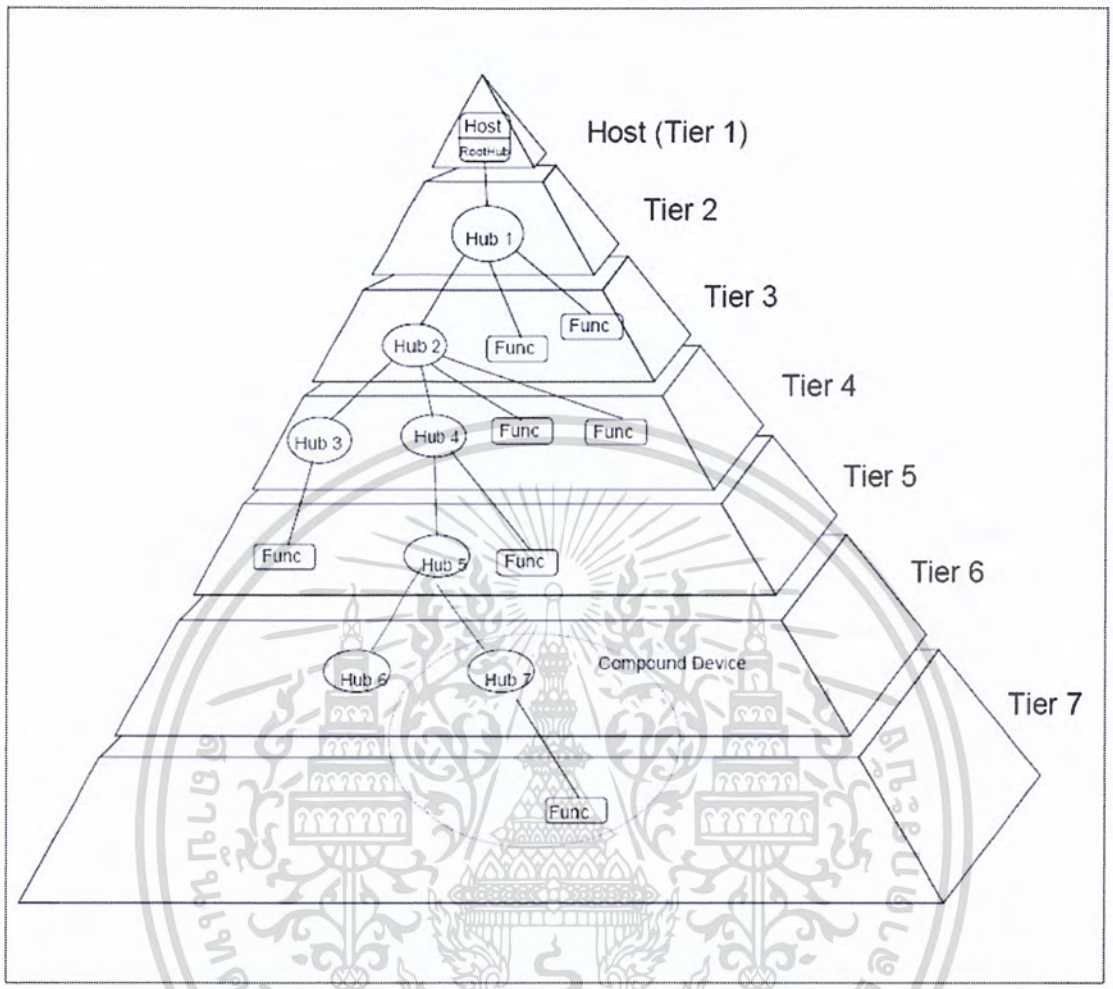
1) อุปกรณ์ความเร็วต่ำ (low speed devices) ถ่ายทอดข้อมูลด้วยความเร็ว 1.5 เมกะบิทต่อวินาที
2) อุปกรณ์ความเร็วเต็มที่ (full speed devices) รับส่งข้อมูลด้วยความเร็ว 12 เมกะบิทต่อวินาที

ในปัจุบันนี้อุปกรณ์ยูเอสบีที่มีจำหน่ายอยู่ในท้องตลาดมีอยู่เป็นจำนวนมาก เช่น คีย์บอร์ด , เมาส์ , จอยสติ๊ก เหล่านี้คืออุปกรณ์ยูเอสความเร็วต่ำ ส่วนจอมอนิเตอร์ , ลำโพง , เครื่องพิมพ์ , กล้องถ่ายรูปดิจิตอล , ซีดีรอมไดรฟ , เครื่องเล่นเอ็มพี3 จัดเป็นอุปกรณ์ยูเอสบีความเร็วสูง อุปกรณ์บางตัวจะบรรจุความสามารถของยูเอสบีฮับ เข้าไปด้วยทำให้สามารถนำอุปกรณ์อื่นๆ มาเชื่อมต่อได้เหมือนกับการต่อเข้ากับฮับอุปกรณ์เหล่านี้ (Compound USB devices) ตัวอย่างของอุปกรณ์ที่มีฮับอยู่ภายในได้แก่ จอมอนิเตอร์ , เครื่องพิมพ์ เป็นต้น

นอกจากการแบ่งชนิดของอุปกรณ์ตามความเร็วในการถ่ายทอดข้อมูลแล้ว ยังแบ่งอุปกรณ์ยูเอสบีตามการใช้พลังงานของตัวอุปกรณ์เองก็ได้ซึ่งสามารถแบ่งได้อีก 2 ชนิด คือ

1) อุปกรณ์ที่ใช้ไฟเลี้ยงจากบัส (bus powered device) คือ อุปกรณ์ที่ใช้ไฟเลี้ยงจากบัสโดยตรง ไม่ต้องมีแหล่งจ่ายไฟภายนอกเพิ่มเติม
2) อุปกรณ์ที่ใช้ไฟเลี้ยงจากตัวเอง (self powered device) คือ อุปกรณ์ที่มีแหล่งจ่ายไฟในตัวไม่ต้องอาศัยไฟเลี้ยงจากบัส

2.2.6 โครงสร้างการเชื่อมต่อ (Topology)



รูป 2.17 USB Topology : Tiered Star

โทโพโลยี หรือการจัดการเชื่อมต่อบนระบบบัส USB จะเป็นแบบไทร์สตาร์ (Tiered Star) หรือสตาร์แบบลำดับชั้น โครงสร้างนี้ประกอบไปด้วยโครงสร้างแบบสตาร์หลายๆ ชุด เชื่อมต่อกันอยู่ โดยมีฮับเป็นจุดศูนย์กลางของโครงสร้างย่อยๆ เหล่านั้น ที่ส่วนบนสุดของสตาร์จะเป็นโฮสคอนโทรลเลอร์ ซึ่ง USB จะมีโฮสเพียงตัวเดียวเท่านั้นที่ทำหน้าที่ควบคุมการติดต่อสื่อสารทั้งหมดของระบบ และมีฮับเป็นตัวเพิ่มจำนวนของพอร์ตที่ใช้เชื่อมต่อกับอุปกรณ์ ทำให้ระบบสามารถเชื่อมต่ออุปกรณ์ได้มากขึ้น

2.2.7 การติดต่อระหว่างอุปกรณ์และโฮสต์

การถ่ายทอดสัญญาณ (Transfer type) ของยูเอสบีนั้นแบ่งออกเป็น 4 ชนิดตามขนาดชนิดของข้อมูลและจังหวะการส่งข้อมูลดังนี้

2.2.7.1 การถ่ายทอดสัญญาณแบบไอโซโครนัส (Isochronous transfer)

ใช้กับการรับส่งข้อมูลที่ต้องการความต่อเนื่องสูง มีอัตราเร็วในการส่งข้อมูลคงที่ ไม่มีการรับประกันความถูกต้องของข้อมูล หากเกิดความผิดพลาดจะไม่มีการพยายามส่งข้อมูลนั้นไปใหม่ การรับส่งแบบนี้จะต้องจองการรับส่งเฟรมข้อมูลในแต่ละ 1 มิลลิวินาที ไว้ตลอดเวลาด้วย เช่น ข้อมูลเสียงเพลง , ลำโพง เป็นต้น โดยมีการซิงโครไนซ์จังหวะการรับส่งเฟรมข้อมูล แบ่งเป็น 3 วิธี คือ

1) อะซิงโครนัส (Asynchronous) สัญญาณนาฬิกาของอุปกรณ์ไม่จำเป็นต้องเข้าจังหวะกับสัญญาณข้อมูลของยูเอสบีแต่อัตราการส่งข้อมูลจะต้องถูกกำหนดไว้ตายตัวที่ค่าใดค่าหนึ่ง ซึ่งกำหนดตอนเริ่มต้นการเชื่อมต่อ ไม่สามารถเปลี่ยนแปลงได้ เช่น ไมโครโฟนที่อัตราการสุ่มข้อมูล (sampling rate) เป็นอิสระไม่ขึ้นกับสัญญาณเอสโอเอฟ (SOF) ของยูเอสบี เป็นต้น
2) ซิงโครนัส (Synchronous) สัญญาณนาฬิกาของอุปกรณ์จะต้องเข้าจังหวะกับสัญญาณข้อมูลและเอสโอเอฟของยูเอสบี เช่น โมเด็ม (modem ISDN 64 kb/s) จะต้องเกิดการ ซิงโครไนซ์ สัญญาณเอสโอเอฟของโฮสต์เข้ากับสัญญาณนาฬิกาของระบบไอเอสดีเอ็น (ISDN) ก่อนจึงจะถ่ายทอดข้อมูลที่ความเร็วคงที่ 64 กิโลบิทต่อวินาทีได้
3) อะแดพทีฟ(Adaptive) วิธีนี้เป็นวิธีที่ตัวอุปกรณ์มีความสามารถมากที่สุด เพราะสามารถปรับความเร็วในการถ่ายทอดข้อมูลได้ในช่วงเวลาที่กว้างมาก ไม่ถูกกำหนดตายตัวไว้ค่าใดค่าหนึ่ง สามารถเปลี่ยนแปลงความเร็วในระหว่างการใช้งาน เช่น ซีดีรอม (CDRom) ที่สามารถเปลี่ยนอัตราการสุ่มข้อมูลได้

2.2.7.2 การถ่ายทอดสัญญาณแบบบัลก์ (Balk transfer)

ใช้กับการถ่ายทอดข้อมูลที่มีปริมาณมากๆ แต่ไม่ต้องการความต่อเนื่องของข้อมูล และไม่ต้องการอัตราการส่งข้อมูลที่คงที่ การส่งช้าหรือเร็วจะส่งผลกระทบในด้านเวลาที่ต้องใช้มากขึ้นเท่านั้น แต่ข้อมูลไม่ได้มีความเสียหายแต่อย่างใด เพราะในการส่งข้อมูลแต่ละครั้งจะมีการตรวจสอบความถูกต้องและจะเกิดการส่งใหม่เมื่อมีความผิดพลาดเกิดขึ้น เช่น ซีดีรอม (ส่วนที่ไม่ใช่การอ่านข้อมูลเป็นเพลง) , เครื่องพิมพ์ หรือ สแกนเนอร์

2.2.7.3 การถ่ายทอดสัญญาณแบบอินเทอร์รัพต์ (Interrupt transfer)

ใช้กับการถ่ายทอดข้อมูลที่มีจำนวนน้อยครั้งและปริมาณของข้อมูลไม่มาก โดยอาศัยวิธีการวนอ่านข้อมูลจากอุปกรณ์เป็นระยะๆ หรือการโพลลิ่ง (Polling) โดยอัตราการวนอ่านนี้จะต้องไม่ช้าเกินไปเพราะจะทำให้เกิดการสูญเสียข้อมูล และไม่เร็วเกินไปเพราะจะทำให้ใช้

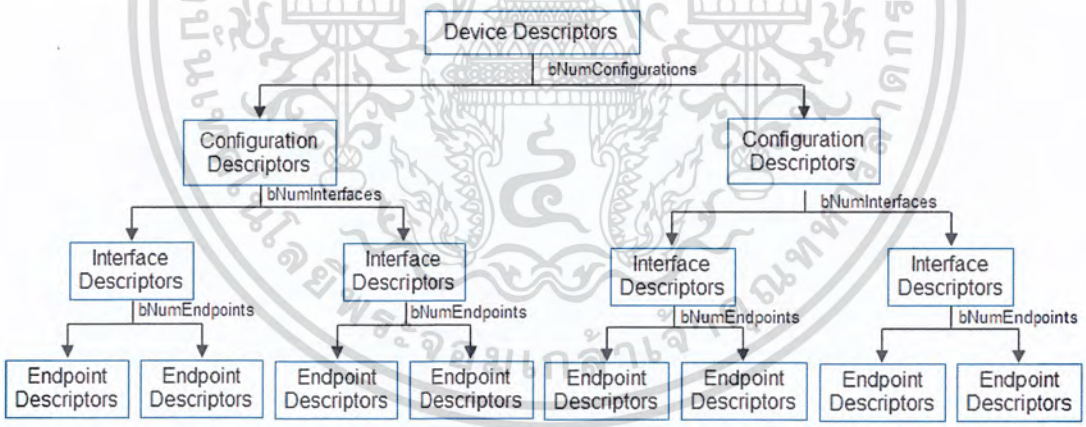
แบนด์วิดท์ (bandwidth) มากเกินความจำเป็น ถ้าหากอุปกรณ์ไม่มีข้อมูลที่จะส่งก็จะส่งแน็ค (NAK : Negative Acknowledgement) ตอบกลับไป เช่น คีย์บอร์ด เป็นต้น

2.2.7.4 การถ่ายทอดสัญญาณควบคุม (Control transfer)

มีความสำคัญมาก เพราะใช้สำหรับรับส่งคำสั่งควบคุมการทำงานทั้งหมดของอุปกรณ์ทุกๆ ตัวตั้งแต่เริ่มแรกเมื่ออุปกรณ์ถูกต่อเข้ากับระบบ เกิดการอ่านเดสคริปเตอร์ ต่างๆ เมื่อพิจารณาแล้วว่าสามารถรองรับการทำงานได้หรือไม่ ถ้าได้ก็จะทำการตั้งค่าการทำงานต่างๆ และเริ่มการทำงาน ทั้งหมดที่กล่าวมานี้จะใช้การถ่ายทอดสัญญาณควบคุมเพื่อติดต่อกับการควบคุมเอ็นพอยท์ (endpoint control) ซึ่งเป็นกุญแจหลักทั้งสิ้น

2.2.8 เดสคริปเตอร์, ความหมาย, ชนิดและรูปแบบการใช้งาน

อุปกรณ์แต่ละตัวมีคุณสมบัติและการทำงานที่แตกต่างกัน โฮสต์จำเป็นต้องรู้คุณสมบัติทั้งหมดของอุปกรณ์แต่ละตัวเพื่อให้การสั่งงานเป็นไปได้อย่างถูกต้อง และเนื่องจากบัสข้อมูลทั้งหมดถูกใช้งานรับส่งข้อมูลร่วมกันระหว่างอุปกรณ์ทุกๆ ตัว สิ่งที่โฮสต์จำเป็นต้องรู้อีกอย่างก็คือ ปริมาณที่ต้องการส่ง หรือแบนด์วิดท์ของอุปกรณ์แต่ละตัวในบัส ดังนั้นเมื่อมีอุปกรณ์ตัวใหม่ต่อเข้ากับบัสก็สามารถใช้ข้อมูลเหล่านี้พิจารณาได้ว่าจะรองรับอุปกรณ์นั้นๆ ได้หรือไม่ ข้อมูลเหล่านี้รวมเรียกว่า ดีไวซ์เดสคริปเตอร์(Device descriptor)



รูป 2.18 Device Descriptors

2.1.8.1 ดีไวซ์เดสคริปเตอร์

ในอุปกรณ์แต่ละตัวจะมี ดีไวซ์เดสคริปเตอร์เพียงชุดเดียวเท่านั้น ภายในจะระบุข้อมูลที่ใช้ในการเชื่อมต่อขั้นแรก (default descriptor pipe) เพื่อใช้ในการกำหนดข้อมูลสถานะของอุปกรณ์เข้ากับโฮสต์ นอกจากนั้นยังเก็บข้อมูลของข้อกำหนดในโหมดการทำงานต่างๆ ของอุปกรณ์รวมถึงจำนวนคอนฟิกกูเรชั่น (configuration) ด้วย เพราะในครั้งแรกที่อุปกรณ์เชื่อมต่อเข้า

กับบัสนั้นโฮสต์ ไม่มีทางรู้ได้เลยว่า ต้องติดต่อกับอุปกรณ์ที่เอ็นพอยท์ใด จึงจำเป็นต้องขอข้อมูลส่วนนี้ก่อนที่จะติดต่อกับส่วนอื่น

2.1.8.2 คอนฟิกกูเรชั่นเดสคริปเตอร์ (Configuration descriptor)

ใช้เก็บข้อมูลที่จำเป็นของการทำงานในแต่ละโหมดการทำงานและเก็บจำนวนอินเทอร์เฟซ ที่ใช้งานในโหมดนั้นๆ เช่น อุปกรณ์บางตัวที่มีการทำงาน 2 โหมด คือ โหมดใช้พลังงานสูง และโหมดประหยัดพลังงานเดสคริปเตอร์ ตัวนี้จะเก็บข้อมูลที่จำเป็นในการตั้งค่าต่างๆ ของโฮสต์ที่ต้องการเลือกใช้โหมดการทำงานแต่โหมดของอุปกรณ์

2.1.8.3 อินเทอร์เฟซเดสคริปเตอร์(Interface descriptor)

ในแต่ละโหมดการทำงานหรือ คอนฟิกกูเรชั่น อาจจะมีการเชื่อมต่อหรืออินเทอร์เฟซ 1 แบบหรือมากกว่าเพื่อใช้งานในหน้าที่ต่างๆ เช่น ซีดีรอม โดยในตัวซีดีรอมจะมีการรับส่งข้อมูลปริมาณมากๆ (mass storage), การส่งข้อมูลเสียงออดิโอ และการส่งข้อมูลภาพจะเห็นได้ว่า มีอินเทอร์เฟซที่ใช้งานแยกกัน ภายในอินเทอร์เฟซ เดสคริปเตอร์จะบรรจุข้อมูลการใช้งานอินเทอร์เฟซนั้นๆ โดยข้อมูลเหล่านี้จะระบุว่าอุปกรณ์ถูกจัดอยู่ในคลาสหรือซับคลาสใดและมีเอ็นพอยท์ จำนวนเท่าใดบ้างที่ใช้งานในอินเทอร์เฟซนี้บ้าง

2.1.8.4 เอ็นพอยท์เดสคริปเตอร์ (Endpoint descriptor)

ใช้เก็บข้อมูลคุณสมบัติของแต่ละเอ็นพอยท์ เช่น ใช้การถ่ายทอดสัญญาณแบบใด และถ่ายทอดข้อมูลได้มากที่สุดครั้งละเท่าใด

2.1.8.5 สตริงเดสคริปเตอร์ (String descriptor)

เป็นเดสคริปเตอร์ที่ไม่ได้อยู่ในโครงสร้างหลัก เพราะเดสคริปเตอร์ชนิดนี้จะเก็บข้อมูลตัวอักษรที่สามารถอ่านเข้าใจได้ไว้อธิบายส่วนต่างๆ ของเดสคริปเตอร์ทั้งสี่ตัวข้างต้น เช่น เก็บชื่อบริษัทผู้ผลิต และ/หรือ หมายเลขประจำตัวของอุปกรณ์ เป็นต้น

2.1.8.6 คลาสสเป็กซิฟิกเดสคริปเตอร์ (Class-specific descriptor)

เป็นเดสคริปเตอร์พิเศษที่มีเฉพาะอุปกรณ์บางตัวที่จัดอยู่ในบางคลาสเท่านั้น ภายในเก็บข้อมูลเฉพาะของคลาสนั้นๆ ที่อยู่นอกเหนือจากเดสคริปเตอร์พื้นฐาน 4 ชนิดแรก

2.1.8.7 สรุปการใช้งานของเดสคริปเตอร์

1) ข้อมูลทั่วไปของตัวอุปกรณ์จะเก็บอยู่ในเดสคริปเตอร์ของอุปกรณ์ หรือเดรไวซ์เดสคริปเตอร์
2) ข้อมูลการทำงานของแต่ละโหมดจะถูกเก็บอยู่ในคอนฟิกกูเรชั่นเดสคริปเตอร์
3) ข้อมูลหน้าที่การทำงานเก็บอยู่ในอินเทอร์เฟซเดสคริปเตอร์
4) ข้อมูลการทำงานของแต่ละเอ็นพอยท์จะถูกเก็บอยู่ในเอ็นพอยท์เดสคริปเตอร์

2.2.9 การตรวจสอบการเชื่อมต่อของอุปกรณ์

ใช้การตรวจสอบจากการเปลี่ยนแปลงของระดับสัญญาณในมายสัญญาณดีลบ (D-) และดีบวก (D+) โดยสายข้อมูลทั้งสองที่ด้านฮับจะถูกต่อพูลดาวน์ (pull-down) ด้วยความต้านทาน 15 กิโลโอห์ม ซึ่งจะส่งผลให้สายสัญญาณทั้งสองมีระดับแรงดับเป็น 0 โวลต์ในขณะที่ไม่มีอุปกรณ์ใดๆ มาเชื่อมต่อ

ที่ด้านอุปกรณ์ สำหรับอุปกรณ์ความเร็วต่ำสายสัญญาณดีลบ จะต่อตัวต้านทาน 1.5 กิโลโอห์ม พูลอัพ (pull-up) กับแรงดัน 3.0-3.6 โวลต์ ส่วนอุปกรณ์ความเร็วสูงสายสัญญาณดีบวกจะถูกพูลอัพแทน ดังนั้นเมื่อมีการต่ออุปกรณ์ตัวใหม่เข้าไปจะทำให้มีการเปลี่ยนแรงดันของสายสัญญาณเพิ่มจาก 0 โวลต์ ขึ้นไปที่ 90% ของไฟเลี้ยงจึงทำให้ฮับรู้ว่ามีการเชื่อมต่ออุปกรณ์เข้าไป

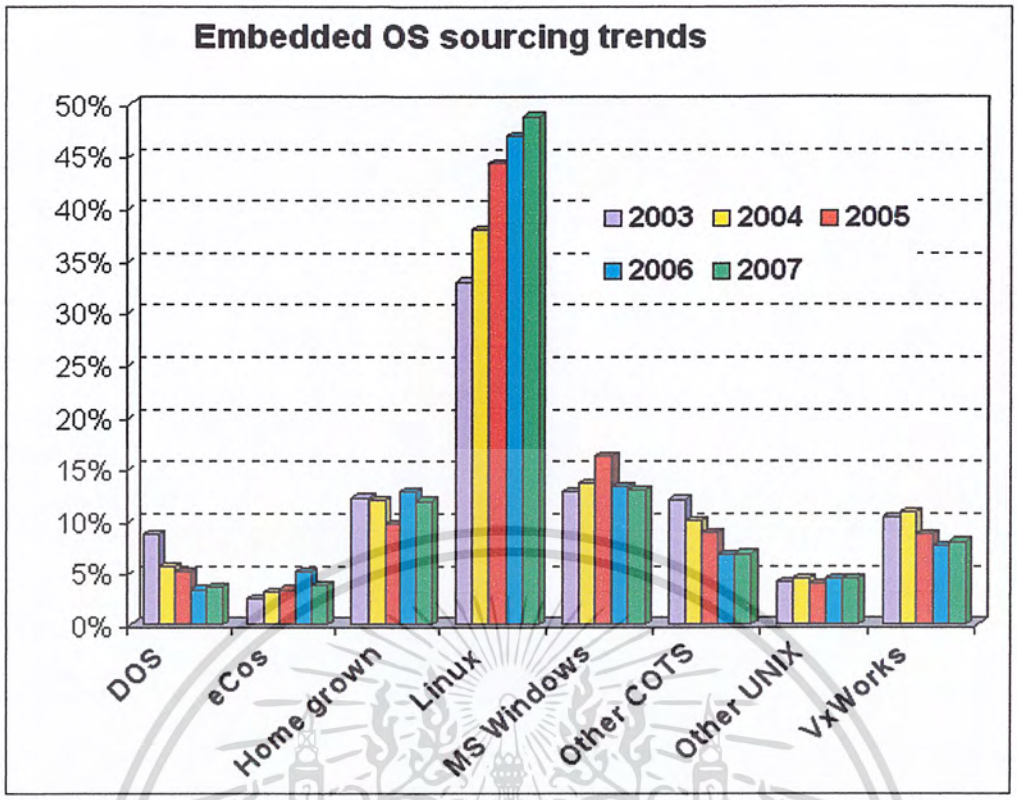
2.2.10 เทคนิคการเข้ารหัสสัญญาณ

การเข้ารหัสสัญญาณของยูเอสบี นั้นใช้เทคนิค 2 อย่างประกอบกันคือ เอ็นอาแซดไอ (NRZI : Non Return To Zero Inverted) และการเติมบิทสตัฟ (bit stuffing)

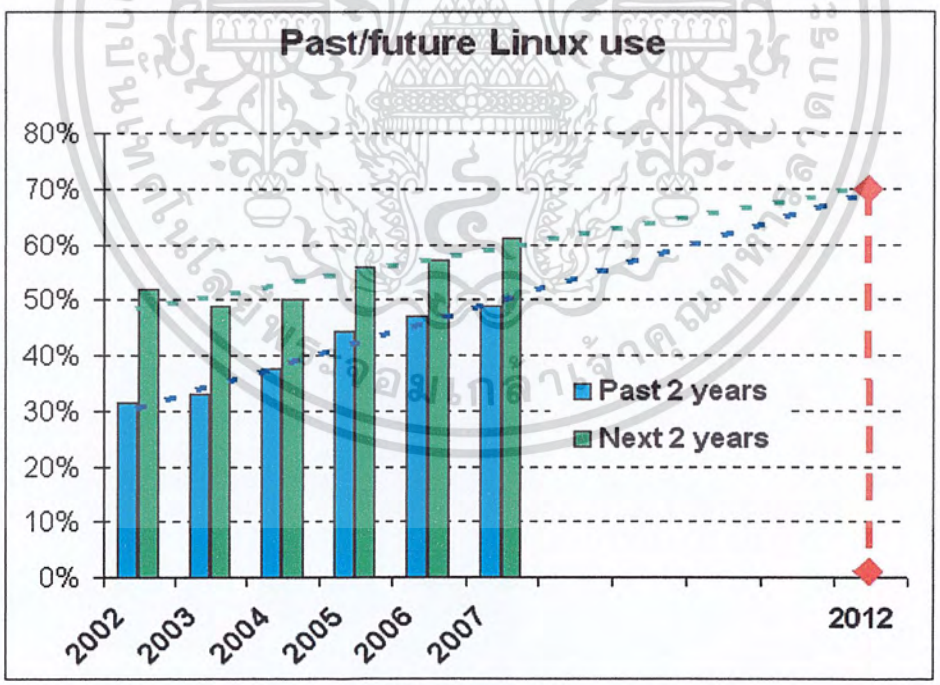
1) เอ็นอาแซดไอ การเปลี่ยนแปลงระดับสัญญาณจากระดับสูงไประดับต่ำหรือระดับต่ำไประดับสูงจะถือเป็นข้อมูล “0” แต่ถ้าไม่เกิดการเปลี่ยนแปลงจะถือเป็นข้อมูล “1” ซึ่งกรณีนี้ถ้าข้อมูลเป็น “1” ติดต่อกันหลายๆ ตัวอาจทำให้เกิดความผิดพลาดในจังหวะการอ่านข้อมูลได้จึงต้องมีการเติมบิทสตัฟเข้ามาช่วย
2) การเติมบิทสตัฟ เมื่อข้อมูลเป็น “1” ติดต่อกัน 6 บิท จะมีการเติมบิท “0” เข้าไปข้างหลัง และเวลาอ่านข้อมูลเมื่ออ่านเจอ “1” ทั้งหมด 6 บิทก็จะทำการอ่านบิทที่ตามหลังมาว่าเป็น “0” หรือไม่ ถ้าเป็น “0” จะไม่นำมาคิด แต่ถ้าไม่ใช่ “0” จะถือว่าข้อมูลนั้นผิดพลาด

2.3 เอ็มเบดเดดลินุกซ์

สาเหตุที่โครงงานได้เลือกใช้เอ็มเบดเดดลินุกซ์ก็เพราะส่วนแบ่งการตลาดในส่วนของอุปกรณ์สมองกลฝังตัวต่างๆนั้นมีอัตราส่วนแบ่งการตลาดที่สูงและมีแนวโน้มในการเติบโตที่สูงเป็นการแสดงให้เห็นถึงกระแสนิยมที่คนยังคงให้ความสนใจในเอ็มเบดเดดลินุกซ์อยู่มาก ประกอบกับต้นทุนทางด้านซอฟแวร์ที่ต่ำ โดยที่ลินุกซ์เคอร์เนลเป็นแบบโอเพ้นซอร์สจึงไม่เสียค่าใช้จ่ายในเริ่มต้นพัฒนาในส่วนนี้ จากส่วนแบ่งตลาดที่สูงเราจึงเห็นอุปกรณ์ฝังตัวที่ใช้ระบบปฏิบัติการลินุกซ์อย่างแพร่หลาย เช่น สวิตซ์ เร้าเตอร์ ปริ้นเตอร์ มือถือ ยูเอวี ยูจีวี แท็บเล็ด เป็นต้น และอีกหนึ่งสิ่งที่กำลังมาแรงในตอนนี้คือระบบปฏิบัติการแอนดรอยด์ที่ถูกพัฒนาบนพื้นฐานของลินุกซ์ และมีเหตุผลอื่นๆอีกมากที่ทำให้โครงงานเลือกใช้เอ็มเบดเดดลินุกซ์ ไม่ว่าจะเป็น ความยืดหยุ่นของระบบปฏิบัติการ หรือการเพิ่มขึ้นของจำนวนฮาร์ดแวร์และซอฟแวร์ อันเป็นเหตุให้เลือกลินุกซ์มาเป็นระบบปฏิบัติการในแพลตฟอร์มของโครงงานในครั้งนี้



รูป 2.19 แนวโน้มการใช้ Embedded OS



รูป 2.20 แนวโน้มอัตราการใช้งาน Linux

2.3.1 โครงสร้างสถาปัตยกรรมเคอร์เนลของลินุกซ์

ถึงแม้ลินุกซ์เคอร์เนล จะออกมาหลายตัวอาทิเช่น เรดแฮด (Red Hat), ซูซี (SuSE), เดเบียน (Debian) แต่สถาปัตยกรรมโครงสร้างพื้นฐานนั้นก็คล้ายคลึงกัน หรือจะแตกต่างกันเพียงเล็กน้อยลินุกซ์เคอร์เนลสามารถถูกแบ่งออกเป็นส่วนๆดังต่อไปนี้

1) ชั้นฮาร์ดแวร์นามธรรม (Hardware abstraction layer)
2) หน่วยจัดการหน่วยความจำ (Memory manager)
3) หน่วยจักการตารางเวลา (Scheduler)
4) ไฟล์ซิสเต็ม (File system)
5) ระบบอินพุทเอาต์พุท (IO subsystem)
6) ระบบเครือข่าย (Networking subsystem)
7) ไอพีซี (IPC)

จึงสรุปแต่ละส่วน และรายละเอียดในการใช้งานในระบบฝังตัว สั้นๆดังต่อไปนี้

2.3.1.1 ชั้นฮาร์ดแวร์นามธรรม

โดยแท้จริงแล้วชั้นฮาร์ดแวร์นามธรรมนั้นก็คือฮาร์ดแวร์แพลตฟอร์ม นั่นเอง ดังนั้นไดร์เวอร์ต่างๆ สามารถถูกใส่ลงไปอย่างง่ายในฮาร์ดแวร์ที่ต่างๆกันไป สถาปัตยกรรมที่ดังๆก็ได้แก่ อาร์ม (ARM) และพาวเวอร์พีซี (PowerPC) ซึ่งมีสัญลักษณ์ ของโครงสร้างข้อมูลที่ดี และเอพีไอทำให้ง่ายในการพอร์ตสู่บอร์ดใหม่อย่างง่าย

โพรเซสเซอร์ฝังตัว ซึ่งสนับสนุน ลินุกซ์เคอร์เนล 2.6

1) MIPS
2) PowerPC
3) ARM
4) M68K
5) CRIS
6) V850
7) SuperH

ชั้นฮาร์ดแวร์นามธรรมนั้นจะมี ส่วนประกอบฮาร์ดแวร์ต่อไปนี้

1) หน่วยประมวลผล, เคช และเอ็มเอ็มยู (Processor, cache, and MMU)
2) การเซ็ทอัพเมมโมรี่แม็พ(Setting up the memory map)
3) ส่วนสนับสนุนการยกเว้นและการตอบสนองอินเตอร์รัพท์ (Exception and interrupt handling support)
4) ดีเอ็มเอ (DMA)
5) ไทม์เมอร์ (Timers)

6) ซิสเต็มคอนโซล (System console)

7) การจัดการบัส (Bus management)

8) การจัดการพลังงาน (Power management)

2.3.1.2 หน่วยจัดการหน่วยความจำ

หน่วยจัดการหน่วยความจำในลินุกซ์จะมีหน้าทีรับผิดชอบสำหรับการควบคุมการเข้าถึงทรัพยากรหน่วยความจำฮาร์ดแวร์ หน่วยจัดการหน่วยความจำจะรับผิดชอบเตรียมการจัดหาหน่วยความจำที่เป็นไดนามิก ให้กับเคอร์เนลซัปซิสเต็ม อาทิเช่นไดร์เวอร์, ไฟล์ซิสเต็ม, และ ชั้นของเน็ทเวิร์ค หน่วยจัดการหน่วยความจำจะมีซอร์ฟแวร์ที่จำเป็นเพื่อจัดการหน่วยความจำเสมือน ให้กับโปรแกรมแอปพลิเคชั่น กระบวนการหนึ่งๆใน ลินุกซ์ซัปซิสเต็มจะกระทำบนตำแหน่งเสมือน ซึ่งตำแหน่งเสมือน นี้ กระบวนการ หนึ่งๆ สามารถจะทับซ้อนกับกระบวนการอื่นๆได้

ลินุกซ์เคอร์เนลแบ่งหน่วยความจำทั้งหมดออกเป็นเพจ ซึ่งมีขนาดแต่ละเพจเป็น 4 กิโลไบต์ ถึงแม้ว่าเพจจะถูกเข้าถึงโดยเคอร์เนล แต่ก็มีบางอันถูกใช้ในโดยเคอร์เนลเอง นอกจากนั้นก็จะถูกใช้โดยแอปพลิเคชั่น

2.3.1.3 หน่วยจัดตารางเวลา

จะจัดการการทำมัลติทาส และได้ถูกปล่อยออกมาโดยได้มุ่งไปที่หลักการจัดตารางเวลาแบบคาดเดาได้ (deterministic scheduling policy) ก่อนจะมาพูดถึงความเป็นมา มาทำความเข้าใจความหมายของรายการต่อไปนี้

1) เคอร์เนลเธรด (Kernel thread) : ก็คือกระบวนการซึ่งไม่ได้ทำในส่วนยูซเซอร์คอนเท็กซ์ แต่จะทำงานในส่วนของเคอร์เนลสเปซ

2) ยูสเซอร์โพรเซส (User process) : ยูสเซอร์โพรเซสหนึ่งจะมี พื้นที่ตำแหน่ง ซึ่งเป็นหน่วยความจำเสมือนจะสามารถเข้าสู่โหมดเคอร์เนลก็ต่อเมื่อเกิด อินเตอรัพต์, เอ็กเซ็บชั่น, หรือ ซิสเต็มคอลล์ถูกทำงาน

3) ยูสเซอร์เธรด (User thread) : เธรดจะทำงานแตกต่างกันไปซึ่งจะถูกแม็ปเข้ากับ ยูสเซอร์โพรเซส หนึ่งๆ ยูสเซอร์เธรดจะแชร์ คอมมอนเท็กซ์, ข้อมูล, และ พื้นที่ฮีป (heap) มีพื้นที่สแต็ก (stack) ที่แยกต่างหาก ทรัพยากรอื่นๆอาทิเช่น ไฟล์, ซิกแนลแฮนเดิล จะถูกแชร์กันข้ามเธรด

2.3.1.4 ลินุกซ์

เริ่มต้นเป็นที่นิยมเป็นที่ต้องการสำหรับงาน ระบบทันเวลา ผลลัพธ์ก็คือหน่วยจัดตารางเวลาจึงปรับปรุงแก้ไขและกลายมาเป็นแบบคาดเดาได้ (deterministic) ต่อไปเป็นการแสดง ไมล์สโตน(mile stone) ที่สำคัญของการพัฒนาการในลินุกซ์เคอร์เนล และลักษณะสำคัญ

1) เริ่มจาก 1.3.55 เคอร์เนล สนับสนุนการทำราวด์โรบิน (round robin) และ การจัดตารางเวลาโดยใช้ไฟโฟ่เป็นหลัก (FIFO-based scheduling) พร้อมด้วย การจัดตารางแบบแบ่งเวลา (classic time-sharing scheduler) ง่ายในการดิสเอเบิลเพจจิ้ง (disable paging) สำหรับเลือกขอบเขต หน่วยความจำของแอปพลิเคชั่น อีกด้วย เป็นเหตุให้หน่วยความจำ ถูกล๊อค ก็เพราะเพจจิ้งสร้างจากระบบ ที่คาดเดาไม่ได้ (nondeterministic)
2) 2.0 เคอร์เนล มีฟังก์ชันใหม่คือ nanosleep() อนุญาตให้โพรเซสหยุดพักเป็นเวลาสั้นๆ
3) 2.2 เคอร์เนล สนับสนุน โพซิกเรียลไทม์ซิกเนล (Posix real-time signals)
4) 2.4 เคอร์เนล ถูกปรับปรุงอย่างมาก และมุ่งไปที่ การจัดตารางเวลาที่ทันเวลา และในที่สุดก็กลายมาเป็น2.6 เคอร์เนล
5) 2.6 เคอร์เนล มีตารางเวลาขึ้นมาใหม่ซึ่งนำการคาดเดาได้ เข้าไปในนโยบายการจัดการเวลา และ ลักษณะทันเวลา อาทิเช่น โพซิกไทม์เมอร์ (POSIX timers) อีกด้วย

2.3.1.5 ไฟล์ซิสเต็ม

บนลินุกซ์ ไฟล์ซิสเต็มทั้งหลายถูกจัดการโดยวีเอฟเอส (VFS : Virtual File System) ซึ่งวีเอฟเอส จะจัดการมุมมองข้อมูลที่ถูกเก็บไว้ในแต่ละอุปกรณ์บนระบบ สิ่งที่ทำคือทำการแยกมุมมองของผู้ใช้งานโดยใช้ ซิสเต็มคอลล์พื้นฐาน แต่ก็อนุญาต ให้นักพัฒนาเคอร์เนล ทำการอิมพรีเม้นท์ ลอจิคอลไฟล์ซิสเต็ม บนอุปกรณ์

อุปกรณ์ที่เป็นลินุกซ์ใดๆ หรือ ระบบฝังตัว ใดๆจะต้องมีไฟล์ซิสเต็ม อย่างน้อย 1 ไฟล์ซิสเต็ม อันนี้จะไม่เหมือนกับระบบทันเวลาที่ไม่จำเป็นจะต้องมีไฟล์ซิสเต็มใดๆ ลินุกซ์จะต้องมีไฟล์ซิสเต็ม ก็มีสาเหตุมาจาก

1) แอปพลิเคชั่นต่างก็มีหลายตัว เนื่องจากเหตุนี้จึงจำเป็นต้องมีพื้นที่สตอเรจ(starage space) ในไฟล์ซิสเต็ม
2) ดีไวซ์ส่วนมากเข้าถึงแบบเป็นไฟล์

พร้อมด้วยลินุกซ์ สนับสนุนไฟล์ซิสเต็มแต่ละชนิดเช่นแฟลซและรอมสำหรับ ระบบฝังตัว ยังสนับสนุนเอ็นเอฟเอส (NFS) อีกด้วย อันซึ่งอนุญาตให้ ไฟล์ซิสเต็ม บนเครื่องโฮสถูกเม้าท์สู่ ระบบฝังตัว ลินุกซ์สนับสนุนเมโมรี่เบสไฟล์ซิสเต็ม (memory-base file system) อันซึ่งใช้ใน ระบบฝังตัวอีกเช่นกัน ต่อไปเป็นการแสดงชนิดต่างๆที่ใช้ใน ไฟล์ซิสเต็มฝังตัว

1) EXT2
2) CRAMFS

3) ROMFS

4) RAMFS

5) JFFS2

6) PROCFS

7) DEVFS

2.3.1.6 ระบบอินพุทเอาต์พุท

ระบบอินพุทเอาต์พุทบนลินุกซ์จัดการการเชื่อมต่ออย่างง่ายๆ และแบบยุ่งยากกับ ดีไวซ์บนบอร์ด มีสามชนิดดีไวซ์ที่ถูกสนับสนุนโดย หน่วยอินพุทเอาต์พุทบน

1) ชาเล็กเตอร์ดีไวซ์ (Character devices) สนับสนุนดีไวซ์แบบลำดับ

2) บล๊อกดีไวซ์ (Block devices) สนับสนุนดีไวซ์แบบแรนดอม, บล๊อกดีไวซ์ เป็นหัวใจในการสร้างไฟล์ซิสเต็ม

3) เน็ทเวิร์คดีไวซ์ (Network devices) สนับสนุน การเชื่อมต่อที่เป็นเน็ตเวิร์ค

2.3.1.7 ระบบเครือข่าย

หนึ่งในหลายๆจุดแข็งของลินุกซ์ ก็คือการมีการสนับสนุน โปโตคอลเน็ทเวิร์ค ที่หลากหลาย ตารางต่อไปเป็นการแสดงคุณลักษณะของแต่ละเคอร์เนลเวอร์ชั่น

ตาราง 2.1 เปรียบคุณลักษณะทางเน็ตเวิร์คของลินุกซ์

	Kernel Availability		
Feature	*2.2*	*2.4*	*2.6*
Layer 2			
Support for bridging	Yes	Yes	Yes
X.25	Yes	Yes	Yes
LAPB	Experimental	Yes	Yes
PPP	Yes	Yes	Yes
SLIP	Yes	Yes	Yes
Ethernet	Yes	Yes	Yes
ATM	No	Yes	Yes
Bluetooth	No	Yes	Yes
Layer 3			
IPV4	Yes	Yes	Yes
IPV6	No	Yes	Yes
IP forwarding	Yes	Yes	Yes
IP multicasting	Yes	Yes	Yes
IP firewalling	Yes	Yes	Yes
IP tunneling	Yes	Yes	Yes
ICMP	Yes	Yes	Yes
ARP	Yes	Yes	Yes
NAT	Yes	Yes	Yes
IPSEC	No	No	Yes
Layer 4 (and above)			
UDP and TCP	Yes	Yes	Yes
BOOTP/RARP/DHCP	Yes	Yes	Yes

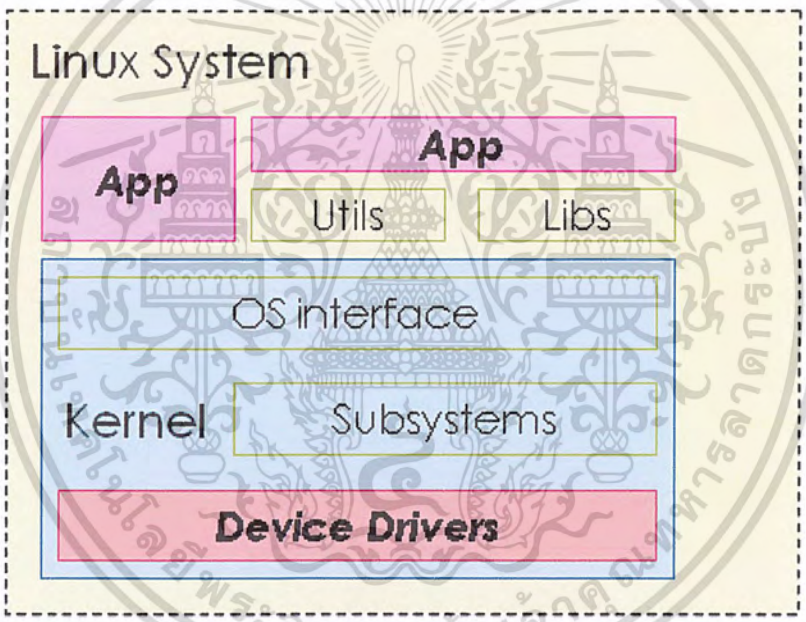
2.3.1.8 ไอพีซี (IPC : Interprocess communication)

ไอพีซีภายในลินุกซ์นั้นจะประกอบด้วยสัญญาน(ใช้สำหรับ asynchronous communication), ไปป์, และ ซ็อกเก็ต เช่นเดียวกับ System V IPC mechanisms ประกอบด้วย การแชร์หน่วยความจำ, แมสเซส คิว (message queues), และ เซมาฟอร์(semaphores) ซึ่งเคอร์เนล 2.6ได้เพิ่มสนับสนุน POSIX-type message queues เข้าไป

2.3.2 ยูสเซอร์ สเปซ (User Space)

ยูสเซอร์ สเปซ ในลินุกซ์จะมีหลักการดังต่อไปนี้

1) โปรแกรม : จะเป็น อิมเมจ และอยู่บนไฟล์ซิสเต็ม เมื่อแอปพลิเคชั่นต้องการขึ้นมาทำงาน อิมเมจจะถูกโหลดลงสู่หน่วยความจำและรัน
2) หน่วยความจำเสมือน: อนุญาตให้แต่ละโพรเซสมี พื้นที่ตำแหน่ง และยังอนุญาตคุณสมบัติพิเศษอาทิเช่น แชร์ไลบรารี่ (shared library) โพรเซสหนึ่งๆจะมีเมมโมรี่แม็ปใน ตำแหน่งพื้นที่เสมือน ซึ่งจะต้องมีหนึ่งเดียว และไม่อิสระแยกจากเคอร์เนลเมโมรี่แม็ป (kernel memory map)
3) ซิสเต็มคอลล์ : เป็นการส่งให้เคอร์เนลทำงาน ดังนั้นเคอร์เนลสามารถประมวลผลบริการต่างๆ ในนามของแอปพลิเคชั่นที่เรียกใช้



รูป 2.21 ส่วนประกอบของระบบลินุกซ์

2.3.3 ความหมายของระบบสมองกลฝังตัว

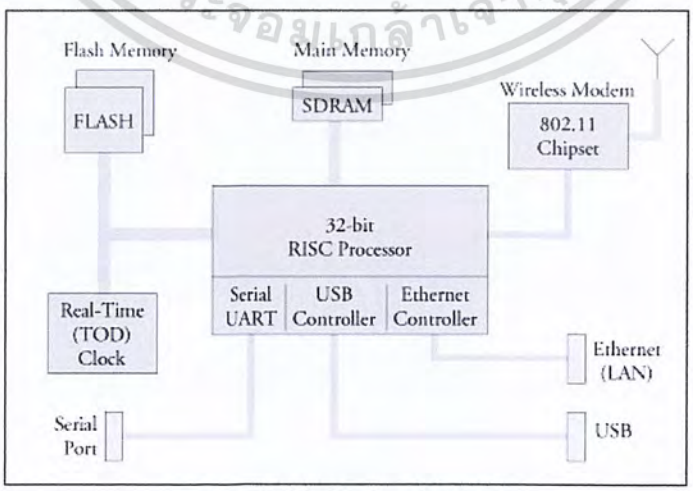
เป็นระบบอิเล็กทรอนิกส์ที่ใช้สำหรับงานควบคุมรวมถึงการแสดงผลการทำงานต่าง ๆ โดยที่ระบบเหล่านี้ถูกใช้เป็นส่วนหนึ่งของระบบและอุปกรณ์ควบคุม เครื่องมือ เครื่องจักรต่าง ๆ การที่ใช้คำว่า “ระบบสมองกลฝังตัว” เนื่องจากระบบเหล่านี้เป็นส่วนหนึ่งของระบบใหญ่ ในหลายกรณีที่ผู้ใช้ทั่วไปอาจไม่ทราบว่าอุปกรณ์ควบคุม เครื่องมือ เครื่องจักรรวมถึงระบบใดที่ใช้งานเป็นประจำเหล่านั้นเป็นระบบสมองกลฝังตัว ในบางครั้งแม้แต่ผู้ที่มีความรู้ทางด้านเทคนิคก็ไม่สามารถระบุได้แน่ชัดว่าสิ่งใดมีระบบสมองกลฝังตัวอยู่ จนกว่าจะมีการทำงานและตรวจสอบกับระบบและอุปกรณ์ควบคุมนั้นระยะหนึ่งเลยที่เดียว ระบบสมองกลฝังตัว นี้แม้ไม่ใช่เครื่อง

คอมพิวเตอร์ แต่ก็มีระบบคอมพิวเตอร์อยู่ภายใน อาจจะเป็นเพียงไมโครโพรเซสเซอร์ (Microprocessor) หรือชิป (chip) ธรรมดาหรือโพรเซสเซอร์ (Processor) ที่ประกอบด้วย ชิป (Chip) ที่มีวงจรซับซ้อน โดยจะมีหลักการทำงาน คือ มีสัญญาณข้อมูลเข้า (Input) จากอุปกรณ์ เซนเซอร์ (Sensor) เข้าสู่ระบบ และมีสัญญาณผลลัพธ์ (Output) ของระบบไปควบคุมบังคับสวิตซ์เครื่องควบคุมต่าง ๆ เช่นสวิตซ์เครื่องจักร หรือ วาล์วควบคุมทิศทางการไหลของท่อทางต่าง ๆ

ระบบฝังตัว นั้นมีรูปร่าง ขนาด ที่หลากหลาย จากขนาดที่ใหญ่สุด มัลติเพิล-แรค ดาต้า สตอเรจ (multiple-rack data storage) หรือ เน็ทเวิร์ค พาเวอร์เฮาส(network powerhouse) ไปจนถึงขนาดเล็กๆ อาทิเช่น เอ็มพีสาม (MP3) ส่วนบุคคล หรือ เซลลูล่า แฮดเซ็ต (cellular handset) ของนั้นมักจะมีคุณสมบัติของ ระบบฝังตัว ประกอบด้วย

1) มีเครื่องประมวลผม
2) ระบบฝังตัว ส่วนใหญ่ต้องทันเวลา
3) พัฒนา และตรวจสอบข้อผิดพลาดยาก
4) ถูกดีไซน์ เฉพาะเจาะจงใน แอปพลิเคชั่น หรือวัตถุประสงค์
5) มีขนาดทรัพยากรที่จำกัด อาทิเช่น หน่วยความจำ(ROM, RAM) อินพุทเอาต์พุท
6) อาจจะมีพลังงาน ที่จำกัด
7) มักจะมุ่งที่งานนั้นๆ โดยที่ปราศจากมนุษย์เข้ามาเกี่ยวข้อง

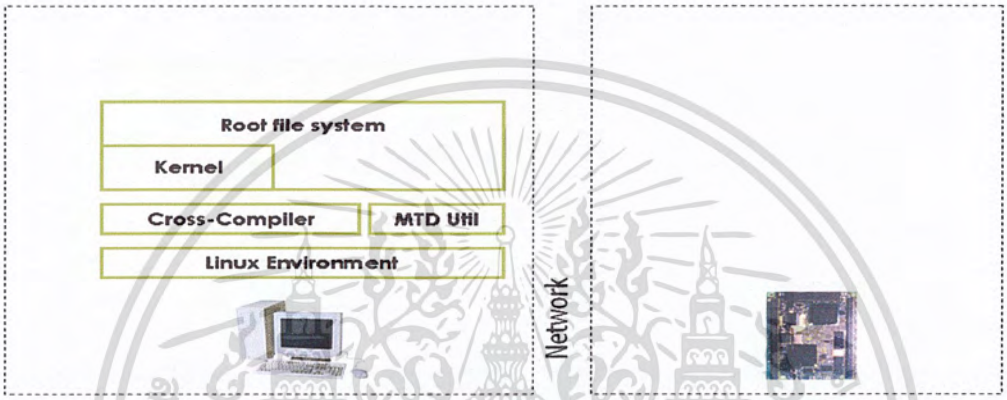
แสดงไดอะแกรมของระบบฝังตัวซึ่งเป็นตัวอย่างอย่างง่ายๆของสถาปัตยกรรมฮาร์ดแวร์ ซึ่งอาจจะเจอในวายเลสแอ็คเซสพอยท์ (wireless access point) ระบบนี้มี 32 บิท ริส โพรเซสเซอร์ เป็นหัวใจของระบบ, มีหน่วยความจำแฟลซใช้ในการเก็บข้อมูล มีหน่วยความจำหลักเป็นเอสดีแรม (SDRAM), โมดูล เรียว ไทม์ คือก (real-time clock), อีเทอร์เน็ท (Ethernet) , วายเลส โมเด็ม (Wirless modem), พอร์ตอนุกรม (Serial port) และ ยูเอสบี อินเทอร์เฟส (USB interface)



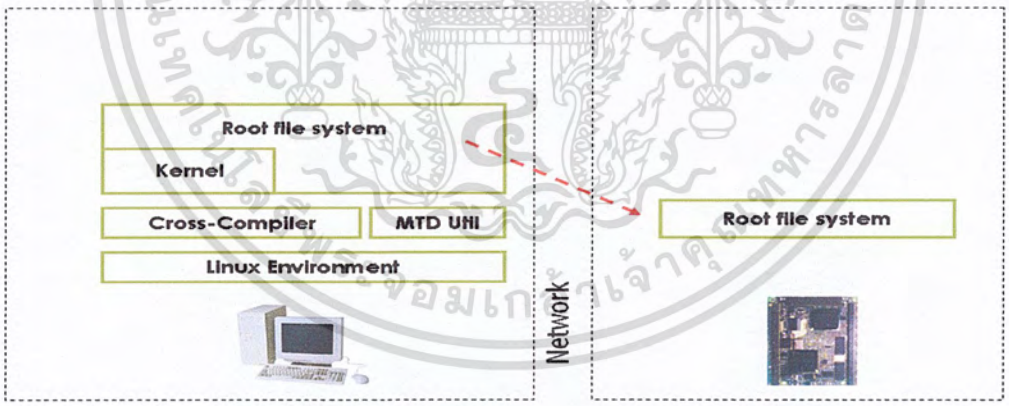
รูป 2.22 ตัวอย่างระบบฝังตัว

2.3.4 ลำดับการพัฒนา

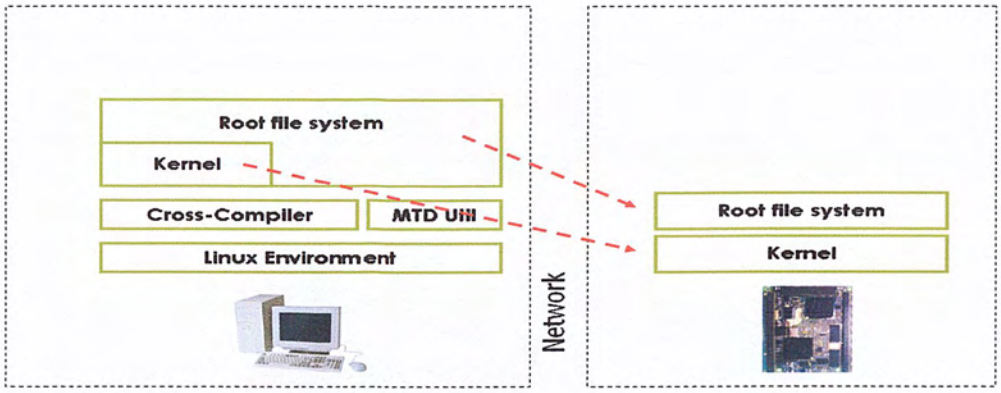
เป็นแนวทางลำดับของการพัฒนาระบบฝังตัวซึ่งจะต้องสร้างสิ่งต่างที่เครื่องพีซี ประกอบด้วย ลินุกซ์ เอ็นวิรอนเม็นท์ (linux environment), คอสคอมไพล์ (cross-compiler), เอ็มทีดียูทิริตี้ (MTD Utilities), ลินุกซ์เคอร์เนล (Linux kernel), รูทไฟล์ซิสเต็ม (Root file system) จากนั้นก็ทำการโหลดรูทไฟล์ซิสเต็ม, ลินุกซ์เคอร์เนล สู่บอร์ดตามลำดับ ในการสร้างแอปพลิเคชั่น และ โมดูล ต่างๆ อาทิเช่นดีไวซ์ไดร์เวอร์นั้นจะต้องเขียน และคอมไพล์ที่เครื่องพีซีโดยใช้ คอสคอมไพล์ จากนั้นจึงโหลดลงบอร์ดต่อไป



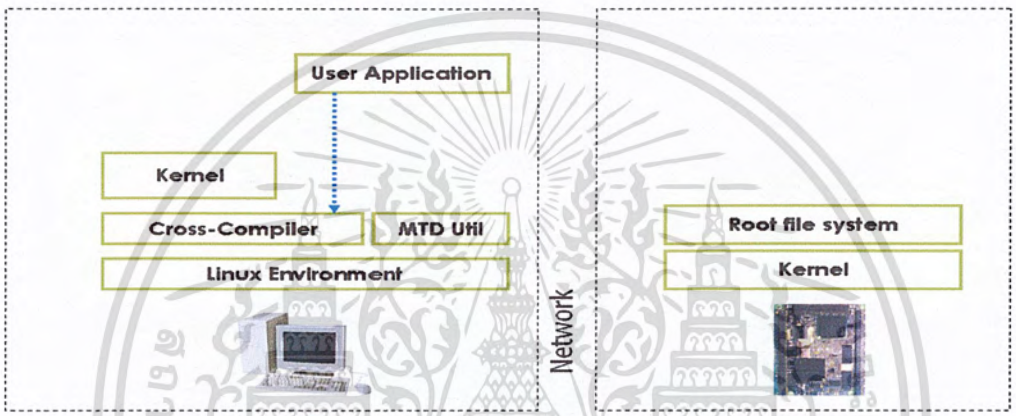
รูป 2.23 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 1



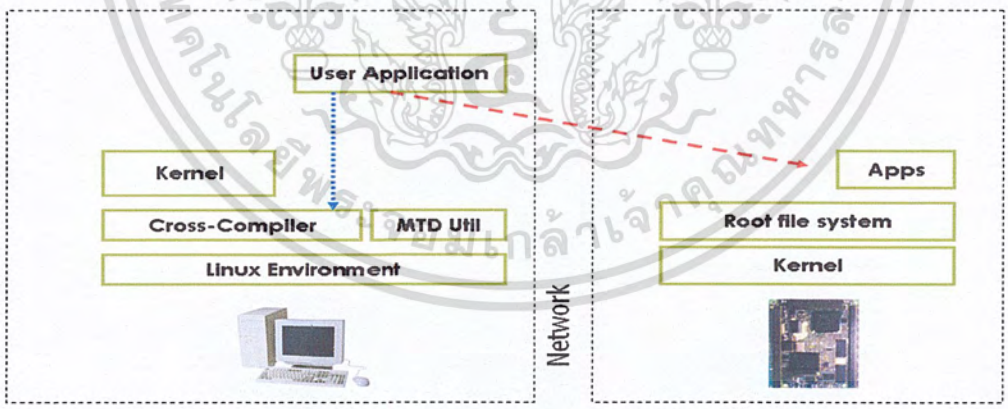
รูป 2.24 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 2



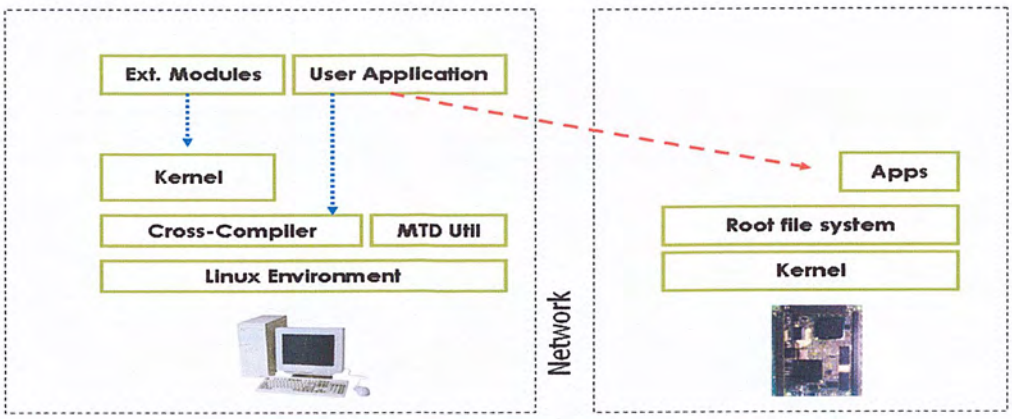
รูป 2.25 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 3



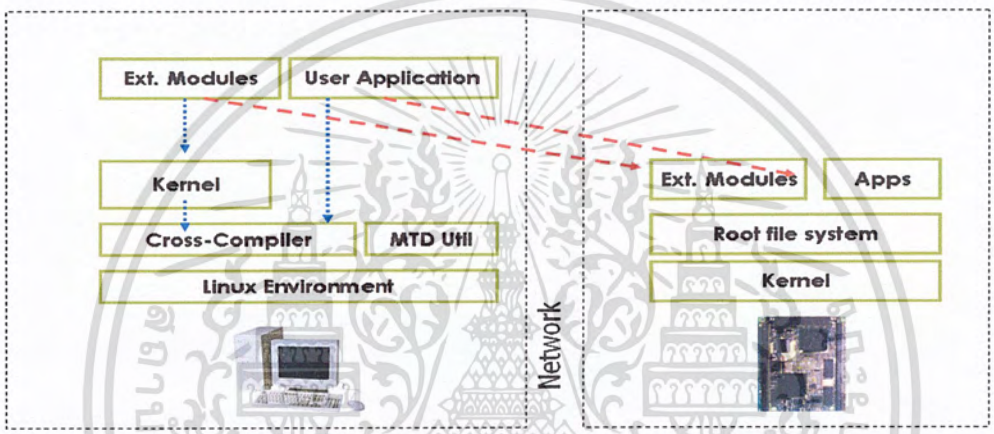
รูป 2.26 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 4



รูป 2.27 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 5



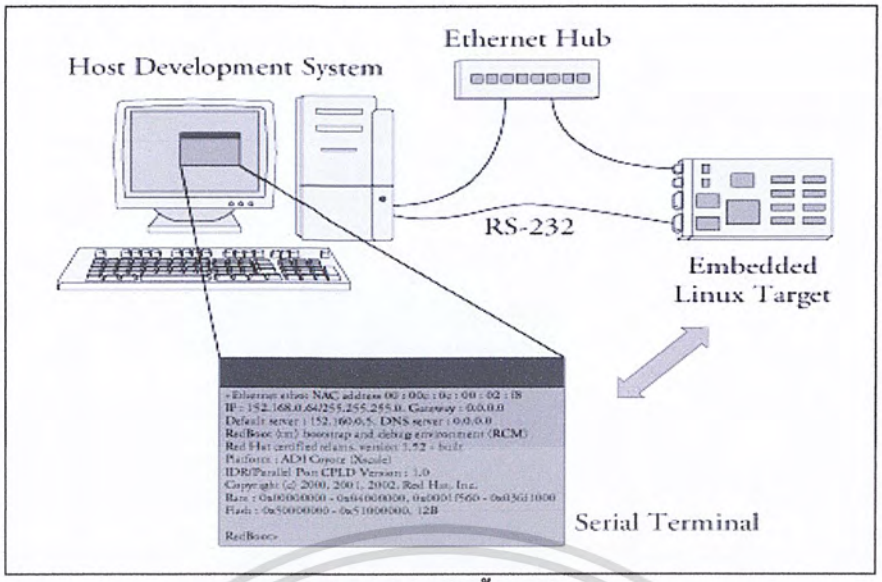
รูป 2.28 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 6



รูป 2.29 ขั้นตอนในการพัฒนาเอ็มเบดเดดลินุกซ์ขั้นที่ 8

2.3.5 การติดตั้ง เอ็มเบดเดด ลินุกซ์ (Embedded Linux development setup)

รูปนี้แสดงการติดตั้งแบบง่ายๆ มีเครื่องที่ใช้ในการพัฒนา (host development) ซึ่ง ลินุกซ์ จะรันอยู่แล้วแต่ชนิดของลินุกซ์ ที่ชื่นชอบ อาทิเช่น เช่น เรดแฮด, ซูซี, เดเบียน เป็นต้น บอร์ดเอ็มเบดเดดลินุกซ์ นั้นๆ จะถูกเชื่อมต่อกับพีซีผ่านทางพอร์ตอนุกรม (RS-232 serial) หรือจะเชื่อมต่อกับบอร์ดผ่านทางอีเทอร์เน็ทผ่านทาง ฮับ หรือ สวิตซ์ เครื่องพัฒนาจะประกอบไปด้วย ทูล (tools) และ ยูทิริตี้ (utilities) ตามแล้วแต่ ชนิดของ เอ็มเบดเดด ลินุกซ์ นั้น



รูป 2.30 การติดตั้งบอร์ด

เมื่อเริ่มทำงานเปิดสวิตซ์พาเวอร์ขึ้นมานั้น ในช่วงขณะนี้ บูทโหลดเดอร์ จะเป็นตัวที่ควบคุมการทำงานของโพรเซสเซอร์ทั้งหมด กระทำการให้ฮาร์ดแวร์ถูกกำหนดค่าเริ่มต้น ประกอบด้วยโพรเซสเซอร์ และหน่วยความจำ ถูกติดตั้ง, กำหนดค่ายูอาร์ท (UART) ไปควบคุมพอร์ตอนุกรม และ กำหนดค่าเริ่มต้นอีเทอร์เน็ทคอนโทรเลอร์และจะแสดงค่าต่างๆออกทาง พอร์ตอนุกรม

```
U-Boot 1.1.4 (Mar 18 2006 - 20:36:11)

AMCC PowerPC 440EP Rev. B
Board: Yosemite - AMCC PPC440EP Evaluation Board
        VCO: 1066 MHz
        CPU: 533 MHz
        PLB: 133 MHz
        OPB: 66 MHz
        EPB: 66 MHz
        PCI: 66 MHz
I2C:   ready
DRAM:  256 MB
FLASH: 64 MB
PCI:   Bus Dev VenId DevId Class Int
In:    serial
Out:   serial
Err:   serial
Net:   ppc_4xx_eth0, ppc_4xx_eth1

=>
```

รูป 2.31 การกำหนดค่าเริ่มต้นในส่วนพอร์ตอนุกรม

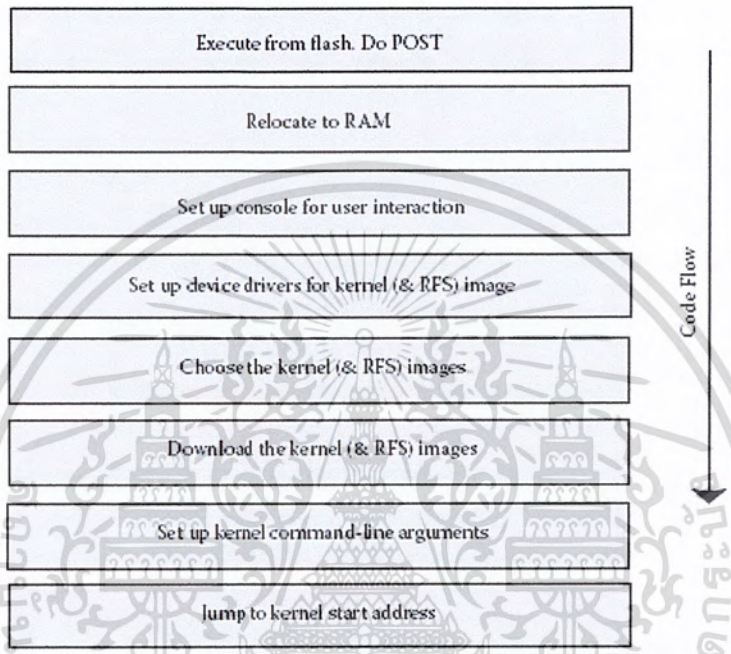
การกำหนดค่าเริ่มต้นของฮาร์ดแวร์

1) คอนฟิคความเร็วซีพียู

2) หน่วยความจำถูกกำหนดค่าเริ่มต้น, และหาขนาดหน่วยความจำของบอร์ด

3) เซ็ตพอร์ตอนุกรม สำหรับ บูทคอนโซล์

4) ตรวจสอบฮาร์ดแวร์ (POST : Power ON Self-Test)



รูป 2.32 ขั้นตอนทำงานของบูทโหลดเดอร์

หลังจากทำขั้นตอนข้างบนเสร็จเรียบร้อยแล้วนั้น ขั้นตอนต่อไปเป็นการโหลด ลินุกซ์เคอร์เนล

บูทเคอร์เนลหลังจากที่บูทโหลดเดอร์ทำการกำหนดค่าเริ่มต้นเสร็จสิ้น ต่อไปบูทโหลดเดอร์ก็จะโหลด และบูท ลินุกซ์เคอร์เนล ซึ่งบูทโหลดเดอร์ ทั้งหลาย จะมี คำสั่งในการโหลด และ เอ็คซีคิว (execute) ตัวโอเอส อิมเมจ รูปต่อไปเป็นการแสดงตัวอย่างหนึ่ง ที่บูทโหลดเดอร์ (u-boot) โหลด และ บูทลินุกซ์เคอร์เนล

```
=> tftpboot 200000 uImage-440ep
ENET Speed is 100 Mbps - FULL duplex connection
Using ppc_4xx_eth0 device
TFTP from server 192.168.1.10; our IP address is 192.168.1.139
Filename 'uImage-amcc'.
Load address: 0x200000
Loading: #################################################################
         ########################################
done
Bytes transferred = 962773 (eb0d5 hex)

=> bootm 200000
## Booting image at 00200000 ...
   Image Name:   Linux-2.6.13
   Image Type:   PowerPC Linux Kernel Image (gzip compressed)
   Data Size:    962709 Bytes = 940.1 kB
   Load Address: 00000000
   Entry Point:  00000000
   Verifying Checksum ... OK
   Uncompressing Kernel Image ... OK
Linux version 2.6.13 (chris@junior) (gcc version 4.0.0 (DENX ELDK 4.0 4.0.0))
  #2 Thu Feb 16 19:30:13 EST 2006
AMCC PowerPC 440EP Yosemite Platform
...
< Lots of Linux kernel boot messages, removed for clarity >
...
amcc login:    <<< This is a Linux kernel console command prompt
```

รูป 2.33 การโหลด ลินุกซ์เคอร์เนล

บูทโหลดเดอร์จะต้องหาที่อยู่ของเคอร์เนล อิมเมจอันซึ่งอาจจะอยู่ในระบบแฟลช หรือ อาจจะอยู่บนเน็ทเวิร์กไม่ว่าจะอยู่ไหนอิมเมจจะต้องถูกโหลดสู่หน่วยความจำในกรณีของอิมเมจที่ถูกบีบอัดมา อิมเมจจะต้องถูกคลายก่อน และถ้าอินนิเชียลแรมดิส (initial randisk) บูทโหลดเดอร์จะต้องโหลดอิมเมจของอินนิเชียลแรมดิส สู่หน่วยความจำอีกด้วย ตำแหน่งที่อยู่หน่วยความจำที่จะถูกเคอร์เนลอิมเมจโหลดนั้น บูทโหลดเดอร์จะเป็นผู้ตัดสินว่าจะอยู่ตรงไหนโดยอ่านจากอีเอ็ฟเอ็ฟ เฮดเดอร์(EFF header) ของเคอร์เนลอิมเมจ

ก่อนที่จะถึงช่วงลอกอินช่วงซีเรียล เทอร์มินอล (serial terminal) ลินุกซ์จะทำการเม้าท์รูทไฟล์ซิสเต็ม รูทไฟล์ซิสเต็มนั้นจะบรรจุ โปรแกรมแอปพลิเคชั่น, ระบบ ไลบรารี, และ ยูทิริตี้ ที่สร้างขึ้นจาก ระบบจีเอ็นยู/ลินุกซ์ (GNU/Linux system)

มีรูทไฟล์ซิสเต็มสามชนิดที่ใช้ในระบบฝังตัว

1) อินนิเชียลแรมดิส

2) Network-based file system using NFS

3) Flash-based file system

จนกระทั่งมาถึงตรงนี้เคอร์เนลกำลังเอ็กซีคิวโค้ดกระทำการกำหนดค่าเริ่มต้นต่างๆ มากมายภายใน คอนเท็กท์ (context) ที่เรียกกันว่า เคอร์เนลคอนเท็กท์ เคอร์เนลนั้นจะเข้าถึงหน่วยความจำแบบฟิซิคอล และ อินพุทเอาต์พุท ซับซิสเต็ม ทั้งหมด

เมื่อลินุกซ์เคอร์เนลกระทำการกำหนดค่าเริ่มต้นเสร็จสิ้น และทำการเม้าท์รูทไฟล์ซิสเต็ม โดยปกติจะเรียกโปรแกรมแอปพลิเคชั่น ขึ้นมาทำงาน ก็คือ init เมื่อเคอร์เนลล์รัน init จะรันอยู่ในส่วนของยูซเซอร์ หรือยูซเซอร์คอนเท็กท์โหลดการทำงานจะอยู่ในส่วนของยูซเซอร์การเข้าถึงซิสเต็มจะต้องใช้ kเคอร์เนลซิสเต็มคอลล์ในการเรียกใช้บริการเคอร์เนลอาทิเช่น ดีไวซ์และ ไฟล์อินพุตเอาต์พุทเป็นต้น ยูซเซอร์สเปซโพรเซสหรือโปรแกรมจะทำงานบนพื้นที่หน่วยความจำเสมือน

2.3.6 ระบบไฟล์

ในหัวข้อนี้จะกล่าวถึงระบบไฟล์ที่เป็นที่นิยมกัน โดยส่วนใหญ่จะใช้ แฟลซ (flash-based) และนอกจากนั้นก็จะใช้แรม (momory-based) แบบใช้แรมสามารถถูกใช้ช่วงเวลาบูทเพื่อที่จะถึงรูทไฟล์ซิสเต็ม และสำหรับเก็บข้อมูลที่เป็น โวเลนไทล์ (volatile) ที่ไม่จำเป็นต้องบันทึก

2.3.6.1 แรมดิสค์ (Ramdisk)

แรมดิสค์เป็นสิ่งทางเลือกหนึ่งของระบบไฟล์ของลินุกซ์ โดยการจำลองให้มีฮาร์ดดิสค์โดยใช้หน่วยความจำ จะใช้แรมดิสค์เมื่อไม่มีอุปกรณ์ความจำอาทิเช่นฮาร์ดดิสค์ หรือแฟลซ สำหรับเก็บรูทไฟล์ซิสเต็ม โดยแท้จริงแล้วแรมดิสค์ไม่ใช้ไฟล์ซิสเต็มแต่ใช้กลไกการทำงานโดยสามารถโหลดไฟล์ซิสเต็มลงสู่หน่วยความจำและใช้งานรูทไฟล์ซิสเต็ม

1) อินนิทอาร์ดี (Initrd) ได้จัดเตรียมกลไกกระบวนการโดยบูทโหลดเดอต์ทำการ โหลดเคอร์เนลอิมเมจ กับ รูทไฟล์ซิสแต็มลงสู่หน่วยความจำการใช้อินนิทอาร์ดีจะต้องมีขั้นตอนที่จะต้องดำเนินการดังต่อไปนี้
2) สร้างอินนิทอาร์ดีอิมเมจ และแพ็คมันไว้กับเคอร์เนล อิมเมจคุณจะต้องสร้างแรมดิสค์อิมเมจบนเครื่องโฮส ต่อไปคุณจะต้องแพ็คแรมดิสค์อิมเมจกับเคอร์เนล และออปชั่นจะต้องตรงกับตัวแพลตฟอร์มเมื่อทำการตอนบูทเคอร์เนล โดยทั่วไปจะมีอีแอลอี เซ็คชั่น (ELE section) เรียกว่า .initrd ซึ่งจะเป็นตัวจับแรมดิสค์อิมเมจ อันซึ่งสามารถทำให้บูทโหลดเดอร์ใช้ได้โดยตรง
3) เปลี่ยนจากบูทโหลดเดอร์ เป็นโหลดอินนิทอาร์ดี อินนิจอาร์ดียังมีกลไกให้คุณสามารถที่จะสลับเปลี่ยนไปใช้รูทไฟล์ซิสเต็มใหม่หลังจากช่วงของระบบทำงาน คุณสามารถใช้อินนิทอาร์ดีสำหรับการกู้และอัพเกรดอีกด้วยเมื่อผลิตภัณฑ์ถูกปล่อยออกไป

2.3.6.2 แรมเอ็ฟเอส (RAMFS)

บ่อยครั้งที่ระบบฝังตัวมีไฟล์ และไม่ต้องการมาใช้อีกเมื่อรีบูต และเพียงต้องการมาเก็บไว้ในไดเรคทรอรี /tmp การที่เก็บไฟล์ไว้ในหน่วยความจำแทนที่จะเก็บไว้ในแฟรชก็เพราะว่าการใช้งานแฟรชนั้นมีราคาสูง คุณควรใช้ แรมเอ็ฟเอส แทนไม่มีการจำกัดขนาด

2.3.6.3 คอมเพรซแรมเอ็ฟเอส (CRAMFS(Compressed RAM File System))

นี่เป็นไฟล์ซิสเต็มที่มีประโยชน์มากสำหรับแฟรชและได้ออกมาในเคอร์เนล 2.4 การอ่านมีการบีบอัดสูง คอมเพรซแรมเอ็ฟเอส เป็นไฟล์ซิสเต็มที่มีกฎเกณฑ์กล่าวคือมันจะต้องกระทำผ่านบัฟเฟอร์ แคช (buffer cache)เพื่อบอกบล็อก ดีไวซ์ (block device)ในการเข้าถึงข้อมูล คุณจะต้องเปิดการใช้งานเอ็มทีดี (MTD) บล็อก ดีไวซ์ ไดร์เวอร์ โหมด คอมเพรซแรมเอ็ฟเอส ใช้แซสลิบ (zlib) สำหรับการบีบอัด และบีบอัดทุกๆ 4 กิโลไบต์(page size) ในการจะเบิร์นคอมเพรซแรมเอ็ฟเอสอิมเมจสู่แฟรชสามารถทำได้โดยใช้โปรแกรมที่เรียกว่า mkcramfs

2.3.6.4 เจเอ็ฟเอ็ฟเอส (Journaling Flash File System-JFFS และ JFFS2)

เราต้องการแฟรชไฟล์ซิสเต็มก็เพราะ

1) ข้อมูลไม่สูญหายเมื่อระบบไม่ทำงาน
2) ใช้ MTD-level เอพีไอในการเข้าถึงชั้นของแฟรชได้โดยตรง

ในปี1999 Axis Communication ได้ปล่อย เจเอ็ฟเอ็ฟเอส สำหรับ ลินุกซ์เคอร์เนล 2.0และมีคุณลักษณะข้างบนทั้งหมด และได้ใส่ลงมาใน 2.2 และ 2.4 ในเวลาต่อมา แต่มันยังขาดการสนับสนุนการบีบอัด โครงการ เจเอ็ฟเอ็ฟเอสทู (JFFS2) จึงได้เริ่มกำเนิดขึ้นมา เจเอ็ฟเอ็ฟเอสทูถูกปล่อยลงสู่เคอร์เนล2.4 และแซงหน้า เจเอ็ฟเอ็ฟเอส เพราคุณลักษณะที่โดดเด่นกว่า ทั้งเจเอ็ฟเอ็ฟเอส และ เจเอ็ฟเอ็ฟเอสทู เป็น log-structure file system ก็คือเปลี่ยนจาก เรคคอร์ด (recorded)กลายเป็น ล๊อก (log)

ภายในล๊อกจะบรรจุสิ่งต่อไปนี้

1) การระบุตัวตน (Identification)
2) เวอร์ชัน ซึ่งจะเป็นหนึ่งเดียวกับล๊อก เป็นส่วนหนึ่งของรายละเอียดไฟล์
3) เมตาดาต้า (Metadata) อาทิเช่น ไทม์สแตมป์ (timestamp)
4) ข้อมูล และขนาดของข้อมูล
5) ออปเซ็ต (Offset)ของข้อมูลในไฟล์
6) การเขียนไฟล์จะสร้างล๊อก และเมื่อต้องการที่จะอ่านล๊อกจะกลับมากลายเป็นไฟล์โดยใช้ ขนาดข้อมูล และออปเซ็ตไฟล์ก็จะถูกสร้างขึ้นมาใหม่ จากนั้นล๊อกก็จะกลายมารอการทำลายโดยการ์เบก คอลเลคชั่น (garbage collection)

คุณลักษณะสำคัญของระบบไฟล์แบบเจเอ็ฟเอ็ฟเอสทูดังต่อไปนี้

1) การจัดการการลบบล็อก (Management of erase block):ใน เจเอ็ฟเอ็ฟเอสทู การลบบล็อกมีสามลิสต์ คลีน (clean), เดอร์ตี้ (dirty), และฟรี (free) ลิสต์clean จะบรรจุเพียงแต่ วาลิด ล็อก(valid log) ลิสต์dirtyบรรจุเพียงหนึ่งหรือหลายล็อกที่ข้อมูลกำลังจะหมดไปเมื่อการ์เบก คอลเลคชั่นถูกเรียกทำงาน ลิสต์freeบรรจุ โน ล็อก (no log) และจากนี้ไปสามารถที่จะถูกเรียกใช้ไปเป็น ล็อกใหม่
2) การ์เบก คอลเลคชั่น:เจเอ็ฟเอ็ฟเอสทู การ์เบก คอลเลคชั่น เกิดขึ้นในเธรดที่แยกต่างหาก อันซึ่งจะเริ่มมาคุณเม้าท์ เจเอ็ฟเอ็ฟเอสทูไฟล์ซิสเต็ม เจเอ็ฟเอ็ฟเอสทูจะสงวนห้าบล็อกสำหรับการทำการ์เบก คอลเลคชั่น
3) การบีบอัด (Compression):ข้อแตกต่างระหว่าง เจเอ็ฟเอ็ฟเอส กับ เจเอ็ฟเอ็ฟเอสทู ก็ตรงที่ เจเอ็ฟเอ็ฟเอสทูมีการบีบอัดโดยใช้ แซสลิบ (zlib) และ รูบิน (rubin)

ทั้งเจเอ็ฟเอ็ฟเอสและเจเอ็ฟเอ็ฟเอสทูไฟล์ซิสเต็มอิมเมจ สามารถถูกสร้างขึ้นจากคำสั่ง mkfs.jffs และ mkfs.jffs2 คำสั่งทั้งสองจะกระทำบนเครื่องโฮส โฮสจะต้องโหลดสู่บอร์ดแล้วเบิร์นสู่แฟรชในภายหลัง

2.3.6.5 เอ็นเอ็ฟเอส (NFS-Network File System)

เอ็นเอ็ฟเอสสามารถใช้ในการเม้าท์ไฟล์ซิสเต็มบนเน็ทเวิร์กในช่วงของการพัฒนา นักพัฒนามันจะใช้เอ็นเอ็ฟเอสเนื่องจาก การใช้แฟลชมีราคาค่อนข้างสูงเนื่องจากแฟรชมีข้อจำกัดของการเขียนที่จำกัด แต่เอ็นเอ็ฟเอส ไม่จำกัดขนาดเนื่องจากสตอเรจ (storage)ทั้งหมดจะอยู่ที่รีโมท (remote) เซิร์ฟเวอร์หรือเครื่องโฮส

ลินุกซ์เคอร์เนล ได้จัดเตรียมกลไกสำหรับ การเม้าท์แบบอัติโนมัติ (automatically mounting) โดยใช้เอ็นเอ็ฟเอสซึ่งจะต้องทำตามขั้นตอนดังต่อไปนี้

1) คอนฟิคออปชั่น CONFIG_NFS_FS อันซึ่งจะอนุญาตให้สามารถเม้าท์ไฟล์ซิสเต็ม จาก รีโมทเซิร์ฟเวอร์ และ CONFIG_ROOT_NFS อันซึ่งอนุญาตให้ใช้ เอ็นเอ็ฟเอส ในตอนเวลาที่สร้างเคอร์เนล
2) คอนฟิคหมายเลขไอพีหรืออาจจะใช้ BOOTP, RARP, หรือ DHCP ในการ คอนฟิคแบบอัตโนมัติผ่านเน็ตเวิร์ก (turn on network auto configuration)
3) ตัวแปรเคอร์เนลคอมมานด์ไลน์ (Kernel command-lind parameter) จะต้องเป็นชนิดเอ็นเอ็ฟเอส

สำหรับรายละเอียดของรูปแบบคำสั่งของตัวแปรอาร์กิวเมนท์ คอมมานด์ไลนด์ นั้นจะถูกอธิบายไว้ในไฟล์ Docomentation/nfsroot.txt ใน kernel source tree

2.3.7 ยูบูท (U-boot)

ยูบูท ได้ให้คำสั่งมากมายสำหรับการโปรแกรม การลบ การป้องกันหน่วยความจำแฟลชซึ่งในจะขอกล่าวถึงและชี้นำการใช้งานคำสั่งและแก้ไขตัวแปรสภาพแวดล้อม

จะใช้คำสั่ง 'flinfo' ในการที่จะเข้าถึงอุปกรณ์แฟลช,ข้อมูลอุปกรณ์ซึ่งจะประกอบไปด้วย ขนาด,ขนาดการลบในแต่ละบล๊อก และการแม็ปหน่วยความจำ

```
U-Boot>  flinfo
Bank # 1: CFI conformant FLASH (32 x 16)  Size: 8 MB in 32 Sectors
 Erase timeout 16384 ms, write timeout 3 ms, buffer write timeout 3 ms, buffer size 32
  Sector Start Addresses:
    28000000 (RO) 28040000 (RO) 28080000      280C0000      28100000
    28140000      28180000      281C0000      28200000      28240000
    28280000      282C0000      28300000      28340000      28380000
    283C0000      28400000      28440000      28480000      284C0000
    28500000      28540000      28580000      285C0000      28600000
    28640000      28680000      286C0000      28700000      28740000
    28780000      287C0000
U-Boot>
```

รูป 2.34 ใช้คำสั่ง flinfo

ผลลัพธ์จะแสดงข้อมูลจำนวนมาก ตัวอย่างเช่น เลย์เอาต์ เซกชั่นของแฟลช, ตำแหน่งเริ่มต้นของเซกชั่น สำหรับในตัวอย่างนี้จะมี 256 กิโลไบต์ 32 เซกชั่นเริ่มที่ตำแหน่ง 0x28000000 รวมเป็น 8 เมกะไบต์

จะใช้คำสั่ง 'printenv' ซึ่งคำสั่งนี้จะแสดงค่าตัวแปลสภาพแวดล้อม

```
OMAP5912 OSK # printenv
baudrate=115200
bootdelay=20
ethaddr=00:0E:99:xx:xx:xx
bootcmd=bootp; bootm
stdin=serial
stdout=serial
stderr=serial
```

รูป 2.35 ใช้คำสั่ง printenv

จะใช้คำสั่ง 'setenv' เป็นการแก้ไขค่าของตัวแปรสภาพแวดล้อมซึ่งถ้าคุณใส่อาร์กิวเมนท์ไปตัวเดียวจะเป็นการลบตัวแปรสภาพแวดล้อมนั้นออกไป

```
OMAP5912 OSK # setenv bootcmd 'cp 0x00040000 0x10000000 0x00200000;bootm'
```

รูป 2.36 เมื่อใช้คำสั่ง setenv

จะใช้คำสั่ง 'saveenv' เมื่อใดก็ตามที่คุณทำการเปลี่ยนแปลงค่าของตัวแปรสภาพแวดล้อมการกระทำนั้นๆจะกระทำบนแรมเพียงอย่างเดียว เมื่อใดก็ตามที่คุณทำการรีบูทสิ่งที่ได้เปลี่ยนแปลงนั้นก็จะหายหมด ถ้าคุณต้องการที่จะเปลี่ยนแปลงอย่างถาวร คุณควรที่จะใช้คำสั่งนี้ ซึ่งจะเป็นการบันทึกค่าตัวแปรสภาพแวดล้อมเมื่อเราทำการแก้ไขโดยใช้คำสั่ง 'setenv'

```
OMAP5912 OSK # saveenv
Saving Environment to Flash...
Un-Protected 1 sectors
Erasing Flash...
Erasing sector  1 ...  done
Erased 1 sectors
Writing to Flash.../done
Protected 1 sectors
```

รูป 2.37 saveenv

จะใช้คำสั่ง 'tftp' เป็นการโอนถ่ายข้อมูลโดยใช้โปรโตคอลทีเอ็ฟทีพี ซึ่งจะต้องโหลดไฟล์มาเก็บไว้ที่แรมก่อน

```
OMAP5912 OSK # tftp 0x10000000 osk1/u-boot-1.1.2.bin
Using MAC Address 00:0E:99:xx:xx:xx
TFTP from server 192.168.1.10; our IP address is 192.168.1.56
Filename 'osk1/u-boot-1.1.2.bin'.
Load address: 0x10000000
Loading: ##################
done
Bytes transferred = 87764 (156d4 hex)
```

รูป 2.38 ใช้คำสั่ง tftp

จะใช้คำสั่ง 'erase' ซึ่งก็คือการลบข้อมูลออกจากแฟลชนั้นเอง

```
OMAP5912 OSK # erase 0x01000000 0x001fffffff
Erasing sector 128 ...  done
...
```

รูป 2.39 ใช้คำสั่ง erase

จะใช้คำสั่ง 'cp' ซึ่งก็คือการก๊อปปี้ข้อมูลโดยจะต้องระบุตำแหน่งจุดตั้งต้น, จุดก๊อปปี้ และก็จำนวนครั้ง

```
OMAP5912 OSK # cp 0x10000000 0x00040000 0x00200000
Copy to Flash...\done
```

รูป 2.40 ใช้คำสั่ง cp

จะใช้คำสั่ง 'bootm' ซึ่งคำสั่งนี้จะเป็นการเริ่มทำงานของซิสเต็มอิมเมจ คำสั่งนี้จะโหลดไฟล์อิมเมจไปไว้นะที่ตำแหน่งหนึ่งๆแล้วทำการแตก(uncompressing)

คำสั่งที่กล่าวถึงนี้เป็นคำสั่งที่ควรที่จะรู้ไว้ใช้งานโดยคำสั่งที่กล่าวถึงไปนั้นเป็นคำสั่งที่ต้องใช้งาน จึงได้ยกตัวอย่างคำสั่งเพียงเท่านี้ คำสั่งและข้อมูลเกี่ยวกับยูบูทนั้นยังมีอีกมากมาย

2.3.8 แอปพลิเคชั่นสำหรับเอ็มเบดเดดลินุกซ์

ในหัวข้อนี้จะขอกล่าวถึงบางแอปพลิเคชั่น ที่จะใช้ในเอ็มเบดเดดลินุกซ์ซิสเต็ม บิวซีบล็อก (Busybox)

โปรแกรมบิวซีบล็อกเป็นมัลติคอลล์ (multicall) โปรแกรม บิวซีบล็อกบรรจุโปรแกรมหลักๆที่เป็นที่รู้จักดังต่อไปนี้

1) เชลล์ (Shells) อาทิเช่น the ash, lash, hush, และอื่นๆ
2) คอร์ยูทิริตี้ (Core utilities) อาทิเช่น cat, chmod, cp, dd, mv, ls, pwd, rm,
3) โพรเซสคอนโทร และ มอนิทอริ่ง ยูทิริตี้ (Process control and monitoring utilities) อาทิเช่น ps, kill, และอื่นๆ
4) โมดูลโหลดดิ่งยูทิริตี้ (Module-loading utilities) อาทิเช่น lsmod, rmmod, modprobe, insmod,and depmod
5) ซิสเต็มทูล (System tools) อาทิเช่น reboot, init, และอื่นๆ
6) เน็ตเวิร์ก ยูทิริตี้ (Networking utilities) อาทิเช่น ifconfig, route, ping, tftp, httpd,
7) telnet, wget, udhcpc (dhcp client), และอื่นๆ
8) ยูทิริตี้สำหรับการลอกอิน และพลาสเวิรด์ (Log-in and password management utilities) อาทิเช่น login, passwd,adduser, deluser, และอื่นๆ
9) อาร์คฮิเวอร์ ยูทิริตี้ (Archival utilities) อาทิเช่น ar, cpio, gzip, tar, และอื่นๆ
10) ซิสเต็ม ล็อกอินยูทิริตี้ (System logging utilities) อาทิเช่น syslogd

การสร้างบิวซีบล็อกแบ่งเป็นสองขั้นตอนดังนี้

1) คอนฟิค:ใช้คำสั่ง make menuconfig แล้วเลือกว่าจะเอาอะไรบ้าง
2) สร้างบิวซีบล็อก:ใช้คำสั่ง make

ขั้นตอนต่อไปเป็นการติดตั้ง บิวซีบล็อก ลงสู่เป้าหมายของคุณ กระทำได้โดยการเรียก บิวซีบล็ฮก กับออปชั่น –intall ยกตัวอย่าง

```
busybox mount –n –t proc /proc /proc
busybox –install –s.
```

คำสั่งการติดตั้งบิวซีบล็อกนั้นจะสร้าง ซอร์ฟ ลิงค์ (soft link) ยกตัวอย่างเช่น หลังจากติดตั้งเสร็จแล้วใช้คำสั่ง ls –l ในไดเรคทรอรี /bin จะแสดงผลลัพธ์ออกมาดังนี้

```
-rwxr-xr-x   1 0      0     1065308 busybox
lrwxrwxrwx   1 0      0           7 init -> busybox
lrwxrwxrwx   1 0      0          12 ash -> /bin/busybox
lrwxrwxrwx   1 0      0          12 cat -> /bin/busybox
lrwxrwxrwx   1 0      0          12 chmod -> /bin/busybox
lrwxrwxrwx   1 0      0          12 cp -> /bin/busybox
lrwxrwxrwx   1 0      0          12 dd -> /bin/busybox
lrwxrwxrwx   1 0      0          12 echo -> /bin/busybox
```

รูป 2.41 ใช้คำสั่ง ls-l

ทีนีลอกอิน(Tinylogin) เป็นมัลติคอลล์โปรแกรม เช่นเดียวกับบิวซีบล็อกและใช้สำหรับการพัฒนายูนิกซ์ ล็อกอิน(UNIX log-in) และ เข้าถึงแอปพลิเคชั่น แสดงฟังก์ชันที่ใช้ทีนีล็อกอิน

1) เพิ่ม และ ลบ ผู้ใช้งาน
2) ล็อกอิน และ เกตตี้แอปพลิเคชั่น (getty application)
3) เปลี่ยนพลาสเวิร์ด

เอ็ฟทีพี เซิร์ฟเวอร์(Ftp Server) ใช้ในการก๊อปปี้ไฟล์จากหรือไปที่เอ็มเบดเดดซิสเต็ม

2.3.9 เคอร์เนล โมดูล

เคอร์เนล โมดูลจะถูกเพิ่มเข้าไปแบบไดนามิคขณะเคอร์เนลกำลังทำงาน มีส่วนประกอบสามส่วนก็คือ

1) โมดูล อินเทอร์เฟส/เอพีไอ (Module interface/APIs)
2) การสร้างโมดูล (Module building)
3) การโหลดโมดูลเข้าและนำโมดูลออก (Module loading and unloading)

ทั้งสามส่วนได้มีการเปลี่ยนแปลงจาก 2.4 เคอร์เนล ไปสู่ 2.6 เคอร์เนล

2.3.9.1 โมดูลเอพีไอ

ภาพข้างล่างแสดงตัวอย่างเคอร์เนลของ 2.4 และ 2.6 kernel โมดูลทั้งสองจะแสดงสตริง Hello world ทุกๆครั้งเมื่อทำการโหลด และจะแสดงสตริง Bye world ทุกๆครั้งที่โมดูลถูกเอาออก

2.3.9.2 การเข้า (Entry) และการออก (exit) ฟังก์ชัน

ทุกๆโมดูลจะต้องมีฟังก์ชัน ลงชื่อ และ ออก ซึ่งจะถูกเรียกอย่างอัตโนมัติโดยเคอร์เนล เมื่อโมดูลถูกโหลด และ ถูกเอาออกโดยลำดับ ใน 2.4 เคอร์เนล ใช้ฟังก์ชัน

init_module() และcleanup_module()

ในการ ลงชื่อ และเอาออก ส่วนใน 2.6 เคอร์เนลจะใช้มาโคร module_init() และ module_exit()

2.3.9.3 การส่งผ่านตัวแปร (Parameter passing)

ทุกๆโมดูลสามารถถูกส่งตัวแปรได้ โดยการส่งทางคอมมานไลด์ เมื่อโมดูลถูกโหลด ใน2.4 เคอร์เนล จะใช้มาโคร MODULE_PARM ส่วน2.6 เคอร์เนล จะใช้มาโคร module_param()

```
/* 2.4 kernel based module */

static int excount = 1;
MODULE_PARM(excount,"i");
static int init_module(void)
{
  int i;
  if(excount <= 0) return -EINVAL;
  for(i=0; i<excount;i++)
    printk("Hello world\n");
  return 0;
}

static void cleanup_module(void)
{
  printk("Bye world\n");
}

/* 2.6 kernel based module code */

MODULE_LICENSE("GPL");
module_param(excount, int, 0);
static int init_module(void)
{
  int i;
  if(excount <= 0) return -EINVAL;
  for(i=0; i<excount;i++)
    printk("Hello world\n");
  return 0;
}

static void cleanup_module(void)
{
printk("Bye world\n");
}

module_init(init_module);
module_exit(cleanup_module);
```

รูป 2.42 Code Hello World ใน 2.6 ,2.4

ทุกๆโมดูลมีการใช้การนับแสดงจำนวนการอ้างถึงโมดูล จำนวน เท่ากับ 0 หมายความว่าโมดูลยังไม่ได้ถูกโหลด

ทุกๆโมดูลต้องประกาศ กระทำได้โดยใช้มาโคร MODULE_LICENSE

2.3.10 ชาเล็กเตอร์ดีไวซ์ไดร์เวอร์ (Charactor device driver)

ชาเล็กเตอร์ดีไวซ์ไดร์เวอร์จะถูกเข้าใช้งานโดยผ่านทางชื่อในไฟล์ซิสเต็ม ชื่อเหล่านี้เรียกว่า สเปเชียลไฟล์ (special file) หรือ ดีไวซ์ไฟล์ หรือ โหนด ซึ่งอยู่ในไดเรคทรอรี /dev ซึ่งจะมีเครื่องหมายเป็น "c" คอลัมท์ แรกของผลลัพธ์เมื่อใช้คำสั่ง ls –l Block device ก็อยู่ใน/dev เช่นกัน แต่จะมีเครื่องหมายเป็น "b" เมื่อคุณใช้คำสั่ง ls –l จะเห็นเลขสองตัว ก่อน วันเดือนปี เลขทั้งสองก็คือ หมายเลขดีไวซ์หลัก และ หมายเลขดีไวซ์รอง ซึ่งจะเป็นเฉพาะแต่ละดีไวซ์ ต่อไปเป็นการแสดงเลขทั้งสอง

```
crw-rw-rw-   1 root     root       1,   3 Apr 11  2002 null
crw-------   1 root     root      10,   1 Apr 11  2002 psaux
crw-------   1 root     root       4,   1 Oct 28 03:04 tty1
crw-rw-rw-   1 root     tty        4,  64 Apr 11  2002 ttyS0
crw-rw----   1 root     uucp       4,  65 Apr 11  2002 ttyS1
crw--w----   1 vcsa     tty        7,   1 Apr 11  2002 vcs1
crw--w----   1 vcsa     tty        7, 129 Apr 11  2002 vcsa1
crw-rw-rw-   1 root     root       1,   5 Apr 11  2002 zero
```

รูป 2.43 มองอุปกรณ์เป็นไฟล์

หมายเลขดีไวซ์หลัก จะมีความสัมพันธ์กับดีไวซ์ลินุกซ์เคอร์เนล สมัยใหม่จะอนุญาตให้ดีไวซ์หลายตัวแชร์หมายเลขดีไวซ์หลักได้ แต่โดยจะยังคงหนึ่งหมายเลขดีไวซ์หลักต่อหนึ่งดีไวซ์ ส่วนหมายเลขดีไวซ์รองถูกใช้โดยเคอร์เนลเพื่อหาดีไวซ์ที่แท้จริง ภายในkernelจะมีชนิดข้อมูล dev_t ซึ่งถูกประกาศใน linux/types.h ถูกใช้เพื่อการดึงหมายเลขdevice ทั้งในส่วนของหมายเลขดีไวซ์หลัก และ หมายเลขดีไวซ์รอง ในเวอร์ชัน 2.6.0 เคอร์เนล dev_t จะมีทั้งหมด 32บิท โดยแบ่งออเป็น 12บิท สำหรับ หมายเลขดีไวซ์หลัก และ 20บิทสำหรับ หมายเลขดีไวซ์รอง ภายใน linux/kdev_t.h จะมีมาโครที่ใช้ในการดึงเลข หมายเลขดีไวซ์หลัก และ หมายเลขดีไวซ์รอง โดยใช้

MAJOR(dev_t dev);

MINOR(dev_t dev);

MKDEV(int major, int minor); จะได้ค่าออกมาเป็นชนิดข้อมูล dev_t

สิ่งแรกที่จะต้องทำเมื่อทำการติดตั้งชาเล็กเตอร์ดีไวซ์ไดร์เวอร์จำเป็นต้องมีการใช้ฟังก์ชัน register_chrdev_region ซึ่งถูกประกาศอยู่ใน linux/fs.h สุดท้ายแล้วชื่อควรจะปรากฏใน /proc/devices และ sysfs register_chrdev_region จะส่งค่าคืนเป็น 0 ถ้าทำการติดตั้งอย่างเสร็จสมบูรณ์ ในส่วนของกรณีผิดพลาด ค่าที่ส่งคืนจะเป็นเลขติดลบ

```
int register_chrdev_region(dev_t first, unsigned int count,char *name);
```

register_chrdev_region จะเป็นฟังก์ชันที่ใช้เมื่อรู้หมายเลขดีไวซ์ ที่ต้องการ แต่ถ้าไม่รู้จะใช้หมายเลขดีไวซ์หลักอะไร ลินุกซ์ก็มีฟังก์ชันที่ใช้ในการจองหมายเลขดีไวซ์แบบไดนามิค ก็คือฟังก์ชัน alloc_chrdev_region

```
int alloc_chrdev_region(dev_t *dev, unsigned int firstminor,unsigned int count,
char *name);
```

ฟังก์ชันนี้ dev เป็นผลลัพธ์อย่างเดียว และจะได้ค่าหมายเลขดีไวซ์หลักออกมา และค่าหมายเลขดีไวซ์รองที่ใส่เข้าไปนั้นโดยปกติแล้วจะใช้

ส่วนการปล่อยหมายเลขดีไวซ์ จะใช้ฟังก์ชัน unregister_chrdev_region

```
void unregister_chrdev_region(dev_t first, unsigned int count);
```

2.3.11 ไฟล์ โอเปอเรชัน (File Operation)

จากที่ผ่านมาได้กล่าวถึงแต่หมายเลขดีไวซ์ที่จะเอาไปใช้ แต่ไม่ได้ยังไม่ได้กล่าวว่าจะติดต่อกระทำการใดๆกับดีไวซ์ได้อย่างไร file_operatoins structureเป็นตัวที่จะจัดการติดตั้งการเชื่อมต่อ สตรักเจอร์ (structure) ซึ่งถูกประกาศไว้ใน linux/fs.h จะเป็นรวบรวม พอยท์เตอร์ (pointer) ของฟังก์ชัน ตัวอย่าง ไฟล์ โอเปอเรชั่นสตรักซึ่งภายในจะมีฟิวที่เป็นพอยท์เตอร์ใช้ไปยังฟังก์ชันที่ถูกเขียนขึ้นเพื่อปฏิบัติงานหนึ่งๆ หรือถ้าเป็น นูล (NULL) ก็จะหมายความว่าไม่สนับสนุนการปฏิบัติงานชนิดนั้นๆ

```
struct file_operations scull_fops = {
    .owner =    THIS_MODULE,
    .llseek =   scull_llseek,
    .read =     scull_read,
    .write =    scull_write,
    .ioctl =    scull_ioctl,
    .open =     scull_open,
    .release =  scull_release,
};
```

รูป 2.44 struct file operations

การรีจิสเตอร์ชาเล็กเตอร์ดีไวซ์ไดร์เวอร์ ก่อนที่เคอร์เนลจะเรียกโอเปอเรชั่นต่างๆ ของดีไวซ์นั้น ก่อนอื่นจะต้องจอง และ รีจิสเตอร์สตรักก่อน ก่อนอื่นจะต้องinclude <linux/cdev.h> และจะมีสองฟังก์ชันในการกำหนดค่าเริ่มต้นคือ

```
void cdev_init(struct cdev *cdev, struct file_operations *fops);
```

และสุดท้ายใช้ฟังก์ชัน

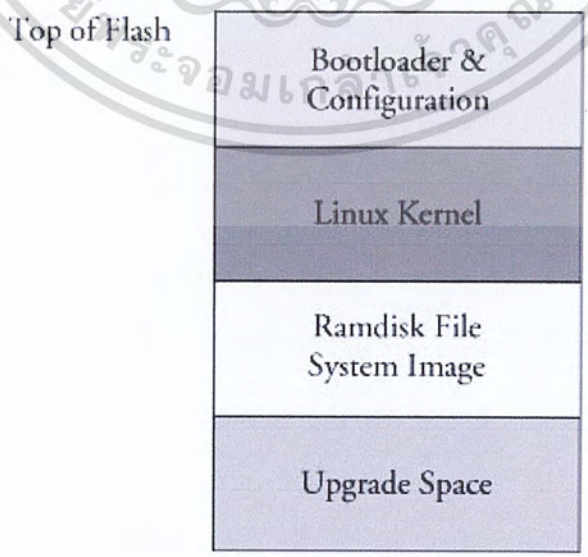
```
int cdev_add(struct cdev *dev, dev_t num, unsigned int count);
```

2.3.12 เอ็มเบดเดด สตอเรจ (Embedded Storage)

จากแนวทางที่สืบต่อกันมา สตอเรจ ในระบบฝังตัวจะใช้รอมในการอ่านโค้ดอย่างเดียว และ เอ็นวีแรม(NVRAM) สำหรับการอ่านและเขียนข้อมูล แต่อย่างไรก็ตามได้กลายมาเป็นเทคโนโลยีแฟลซ อันซึ่ง มีความเป็น นอนวาเลนไทล์ (nonvolatile)อย่างสูง ประกอบกับราคาไม่สูง แฟลซ จึงมาใช้ในระบบฝังตัวอย่างมากขึ้น

การแม็ปแฟลซ ในระบบฝังตัวนั้นแฟลซจะถูกใช้โดยทั่วไปดังต่อไปนี้

1) ส่วนเก็บบูทโหลดเดอร์
2) ส่วนเก็บโอเอสอิมเมจ
3) ส่วนเก็บแอปพลิเคชั่นอิมเมจ ไลบรารี่อิมเมจ
4) ส่วนเก็บการอ่านเขียนไฟล์



รูป 2.45 โครงสร้างหน่วยความจำแฟลซทั่วๆไปในระบบฝังตัว

บ่อยครั้งที่บูทโหลดเดอร์จะอยู่ในส่วนบนสุด หรือล่างสุดของหน่วยความจำแฟลซ และก็เรียงตามลำดับคือลินุกซ์เคอร์เนลและไฟล์ซิสเต็มเอ็มเมจ

หนึ่งในการเพิ่มพูนประสิทธิภาพของแฟลซไฟล์ซิสเต็มนั้นกระทำได้ แฟรซบล๊อกนั้นมี ระยะเวลาการใช้งานที่ถูกจำกัด จึงต้องใส่วิธีการในการลบบล๊อกแฟลซที่แตกต่างกันไป

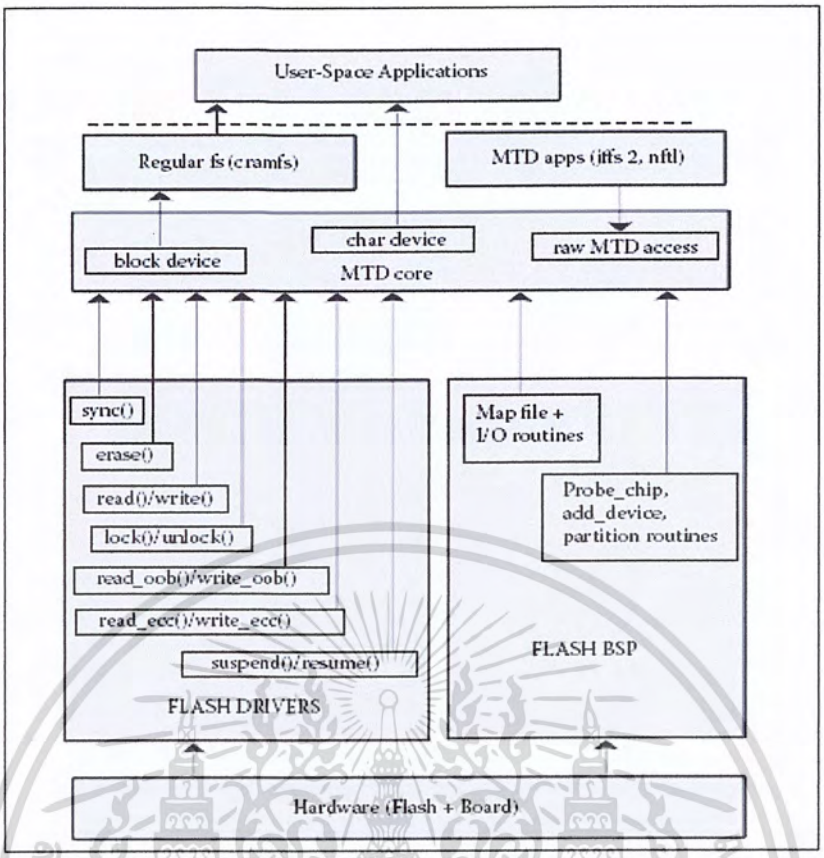
ข้อจำกัดอื่นอาจเกิดขึ้น เนื่องจากสถาปัตยกรรมของแฟลซซึ่งมีความเสี่ยง ที่ข้อมูลจะสูญหาย ในระหว่างที่ ไฟไม่ทำงานหรือ พรีมาเตอร์ชัตดาวน์ (premature shutdown) ก็เนื่องจากว่าขนาด แฟลซบล็อก ค่อนข้างที่จะใหญ่ และขนาดของไฟล์โดยเฉลี่ยแล้วจะเล็กมาก เมื่อเทียบกับขนาดของบล็อก เมื่อถูกเขียนลงไป ซึ่งในการเขียนแต่ละครั้งจะต้องกระทำเป็นบล็อก เช่นนั้นในการเขียนไฟล์เล็กๆซึ่งมีขนาดเท่ากับ 8กิโลไบต์ จะต้องลบ และเขียนใหม่เป็นบล็อกซึ่งมีขนาด 64 กิโลไบต์ หรือ 128กิโลไบต์

หนึ่งในหลายแฟลซไฟล์ซิสเต็มที่นิยมในวันนี้นั้นจะใช้ เจเอ็ฟเอ็ฟเอสทู หรือ Journaling Flash File System2 ซึ่งมีลักษณะสำคัญหลายอย่างที่ได้มุ่งไปที่การปรับปรุงประสิทธิภาพทั้งหมดเพิ่มอายุการใช้งานและ ลดทอนความเสี่ยงที่ข้อมูลจะหายเมื่อเกิดไฟดับ

2.3.13 สถาปัตยกรรม เอ็มทีดี

เอ็มทีดี เป็นระบบย่อย ที่ใช้จัดการอุปกรณ์สตอเรจบนบอร์ด สถาปัตยกรรม เอ็มทีดีถูกแบ่งออกเป็นส่วนดังต่อไปนี้

1) เอ็มทีดี คอล (MTD core):จะทำหน้าที่จัดการการเชื่อมต่อระหว่างไดร์เวอร์แแฟลซแบบระดับ(Low-level flash driver) ต่ำ กับ แอปพลิเคชั่น มันจะอิมพลีเมนท์ เป็น ชาเล็กเตอร์ และ บล็อกดีไวซ์ โหมด
2) ไดร์เวอร์แแฟลซแบบระดับต่ำ: ในส่วนนี้จะกล่าวถึงเพียงแต่ในส่วนที่เกี่ยวกับ นอร์-เบสด์ (NOR-) และ แนนด์-เบสด์ (NAND-based)แฟลซชิป
3) บีเอสพี สำหรับ แฟลซ (BSP for flash) : แฟลซสามารถเชื่อมต่อกับบอร์ด ยกตัวอย่างเช่น นอร์แฟลซ สามารถถูกเชื่อมต่อโดยตรงกับบัสโพรเซสเซอร์ หรืออาจจะเชื่อมต่อกับ พีซีไอบัสชั้นบีเอสพีจะทำให้แฟลซไดร์เวอร์ทำงาน กับบอร์ดหรือโพรเซสเซอร์ใดๆได้
4) เอ็มทีดีแอปพลิเคชั่น: สามารถให้ โมดูลย่อย อาทิเช่น เจเอ็ฟเอ็ฟเอส2 , เอ็นเอ็ฟทีแอล (NFTL), แอปพลิเคชั่นส่วนยูซเซอร์สเปซ จัดการใดๆอาทิเช่น อัพเกรด



รูป 2.46 สถาปัตยกรรมเอ็มทีดี

2.3.14 หน่วยจัดการอินเตอร์รัพท์

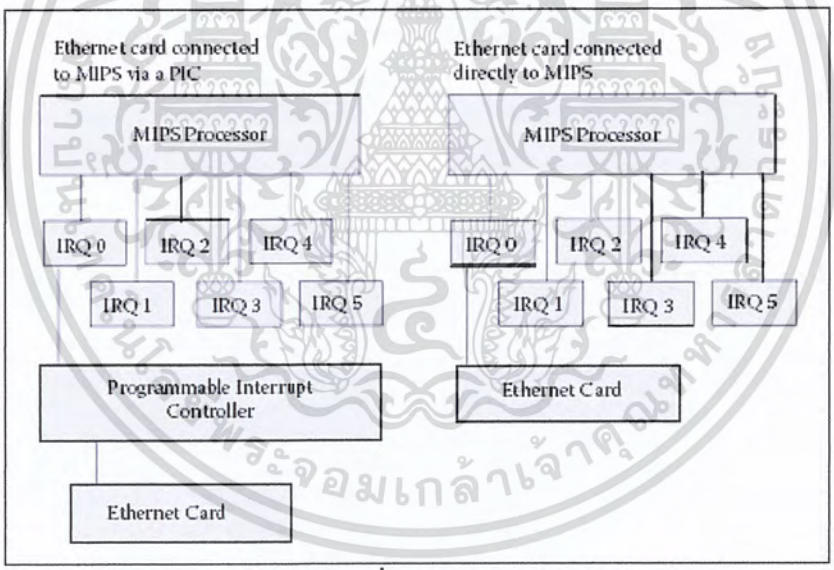
ทุกๆบอร์ดจะมีหน่วยจัดการอินเตอร์รัพท์โดยมากแล้วจะเป็น พีไอซี (PIC : Programmable Interrupt Controller) ในหัวข้อนี้จะกล่าวถึงขั้นตอนอย่างละเอียดนำไปสู่การเขียนโปรแกรมอินเตอร์รัพท์คอนโทรเลอร์ในลินุกซ์ ก่อนที่จะไปถึงการรายละเอียดการเขียนโปรแกรมมาทำความเข้าใจฟังก์ชันการทำงานพื้นฐานของ พีไอซี ก่อน

1) ไมโครโปรเซสเซอร์ โดยปกติจะมีจำนวนอินเตอร์รัพท์ที่จำกัด และอาจจะไม่พอถ้ามีอุปกรณ์บนบอร์ด และทั้งหมดต้องการการอินเตอร์รัพท์ อินเตอร์รัพท์คอนโทรเลอร์จึงเข้ามาช่วยเหลือในการแก้ปัญหานี้ โดยอนุญาตให้อินเตอร์รัพท์ทั้งหลายถูกมัลติเพลคซ์ บนเส้นทางอินเตอร์รัพท์เดียว ทำให้ขยายความสามารถของโพรเซสเซอร์
2) พีไอซี จัดการลำดับความสำคัญของอินเตอร์รัพท์ทำให้เป็นประโยชน์ในรายของกรณีที่ โพรเซสเซอร์ ไม่สนับสนุนลำดับความสำคัญของการอินเตอร์รัพท์เมื่อโพรเซสเซอร์กำลังจัดการทำงานกับอินเตอร์รัพท์ที่มีลำดับความสำคัญสูง พีไอซีจะไม่ส่งมอบอินเตอร์รัพท์ที่มีลำดับความสำคัญต่ำให้แก่โพรเซสเซอร์ดังนั้นเมื่อมี

สองอุปกรณ์เกิดการอินเตอร์รัพท์ พีไอซีจะดูลำดับความสำคัญและส่งอินเตอร์รัพท์ที่มีลำดับความสำคัญสูงแก่ซีพียู

3) ลักษณะการทริกเกอร์ ก็คือฮาร์ดแวร์จะตรวจรับการอินเตอร์รัพท์สองวิธีคือ เลเวล (level) และ เอดจ์ ทริกเกอร์ (edge trigger) พีไอซีสามารถแปลงจากเอดจ์ ไปเป็นเลเวล อินเตอร์รัพท์ โดยใช้การแลชอินพุตอินเตอร์รัพท์ 8259A เป็นตัวอย่างของพีไอซี ไอซีตัวนี้เป็นที่นิยม และเป็นฮาร์ดแวร์ที่เก่ง และได้ถูกนำไปใช้ในบอร์ดเอ็มเบดเดด กันอย่างแพร่หลาย แสดงคุณลักษณะพื้นฐานของ 8259A controller ดังต่อไปนี้

ลินุกซ์เคอร์เนล จะถือปฏิบัติว่าอินเตอร์รัพท์ ทั้งหมดเป็นลอจิคอลอินเตอร์รัพท์ ลอจิคอลอินเตอร์รัพท์ที่เชื่อมโดยตรงกับโพรเซสเซอร์ หรือผ่านทางพีไอซีในกรณีทั้งสองเมื่ออินเทอรัพต์ถูกรีจิสเตอร์จะต้องทำผ่านฟังก์ชัน request_irq(), ดีไวซ์ไดร์เวอร์ จะต้องส่งหมายเลขอินเตอร์รัพต์ แก่ฟังก์ชัน จำนวนลอจิคอลอินเตอร์รัพท์ขึ้นอยู่กับโพรเซสเซอร์ แต่ละตัว การแม็ประหว่าง ลอจิคอลอินเตอร์รัพท์ กับ ฟีซิคอลอินเตอร์รัพท์ เป็นหน้าที่ความรับผิดชอบของบีเอสพี



รูป 2.47 การเชื่อมต่ออินเตอร์รัพท์

แกนหลักของ บีเอสพีอินเตอร์รัพท์ มีสองโครงสร้างข้อมูลคือ

1) Interrupt controller descriptor hw_initerrupt_type : โครงสร้างนี้จะถูกประกาศไว้ใน include/linux/irq.h ทุกๆ ฮาร์ดแวร์อินเตอร์รัพต์ต้องทำการอิมพลีเมนท์โครงสร้างนี้ ฟิวที่สำคัญในโครงสร้างข้อมูลมีดังต่อไปนี้

1.1) Start-up: เป็น พอยท์เตอร์ ไปที่ฟังก์ชันที่ใช้ในการร้องขอเมื่ออินเตอร์รัพท์ ถูกโปรพ (probe)หรือถูกเรียก ใช้ฟังก์ชัน request_irq()

1.2) Shutdown:เป็น พอยท์เตอร์ ไปที่ฟังก์ชันที่ใช้ในการร้องขอเมื่ออินเตอร์รัพท์ถูกปลดปล่อย ใช้ฟังก์ชัน free_irq()

1.3) Enable: เป็น พอยท์เตอร์ ไปที่ฟังก์ชันที่ทำการเปิดอินเตอร์รัพท์

1.4) Disable:เป็น พอยท์เตอร์ ไปที่ฟังก์ชันที่ใช้ในการปิดอินเตอร์รัพท์

1.5) Ack:เป็น พอยท์เตอร์ไปที่ฟังก์ชัน controller-specific ที่ใช้ในการ ตอบรับอินเตอร์รัพท์

1.6) End:เป็น พอยท์เตอร์ ไปที่ฟังก์ชันที่จะเรียกขึ้นมา เมื่ออินเตอร์รัพท์ถูกบริการเสร็จสิ้น

2) IRQ descriptor irq_desc_t : ถูกประกาศไว้ใน include/linux/irq.h อีกเช่นกัน ทุกๆลอจิคอลอินเตอร์รัพท์ จะต้องใช้โครงสร้างนี้ ฟิวที่สำคัญในโครงสร้างข้อมูลมีดังต่อไปนี้

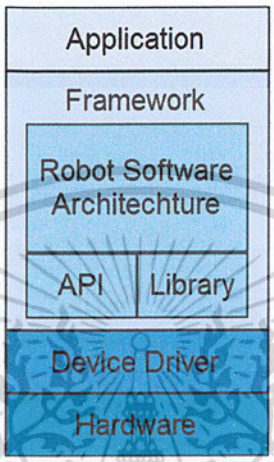
2.1) Status : สถานะของแหล่งกำเนิดอินเตอร์รัพท์

2.2) Handler : เป็นพอยท์เตอร์ ไปที่อินเตอร์รัพท์คอนโทรเดสคริปเตอร์

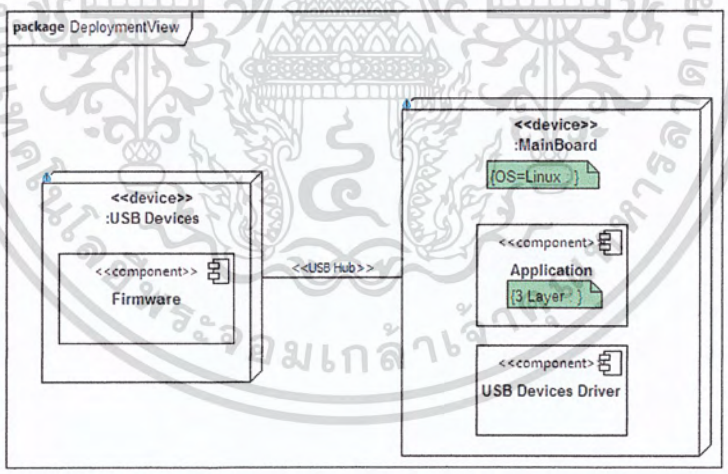
2.3) Action : รายการการทำงานของไออาร์คิว (IRQ action list)

บทที่ 3
แนวทาง และการพัฒนาแพลตฟอร์ม

3.1 ภาพรวมของระบบ



รูป 3.1 ภาพรวมของระบบ



รูป 3.2 Deployment Diagram ภาพรวมของระบบ

สำหรับภาพรวมของแพลตฟอร์มลินุกซ์ฝังตัวสำหรับหุ่นยนต์แบบยานพาหนะไร้คนขับนั้น แบ่งออกเป็น 2 ส่วน ได้แก่

3.1.1 เอ็มเบดเดดโฮส (Embedded Host)

เอ็มเบดเดดโฮส (Embedded Host) หรือบอร์ดหลัก เป็นบอร์ดที่ใช้ระบบปฏิบัติลินุกซ์เป็นตัวจัดการทรัพยากรต่างๆ รวมทั้งเชื่อมโมดูลอุปกรณ์ยูเอสบี (USB Devices) ต่างๆเข้า

ด้วยกัน โดยติดต่อผ่านทางช่องยูเอสบีโฮส (USB Host) ของบอร์ดหลักที่เสริมจำนวนช่องด้วยยูเอสบีฮับ (USB Hub) ในบอร์ดหลักนี้จะเป็นบอร์ดที่ใช้งานโปรแกรมหลัก (Execute main program) ของหุ่นยนต์ โดยมีดีไวซ์ไดรเวอร์ (Device Drivers) ทำงานร่วมกับ LibUSB ทำหน้าที่เป็นตัวเชื่อมต่อบอร์ดหลักกับโมดูลอุปกรณ์ยูเอสบี (USB Devices) และมีดีไวซ์ไลบรารี่ (Device Library) เป็นที่เก็บฟังก์ชั่นการทำงานต่างๆของแต่ละโมดูลอุปกรณ์ยูเอสบี (USB Devices)

3.1.2 ยูเอสบีดีไวซ์ (USB Devices)

ยูเอสบีดีไวซ์ (USB Devices) หรือโมดูลอุปกรณ์ยูเอสบี เป็นอุปกรณ์ที่ถูกพัฒนาบนฮาร์ดแวร์ตระกูล AVR Microcontroller โดยจัดการอุปกรณ์ต่างๆบนบอร์ดผ่านเฟิร์มแวร์ (Firmware) ที่พัฒนามาพร้อมกับ V-USB ที่เป็นซอฟแวร์ที่ใช้ในการจัดการกับยูเอสบี ทำให้ไม่ต้องใช้อุปกรณ์เสริมอื่นๆในการติดต่อกับยูเอสบี (USB)

แพลตฟอร์มลินุกซ์ฝังตัวสำหรับหุ่นยนต์แบบยานพาหนะไร้คนขับนี้ เป็นแพลตฟอร์มที่มีความยืดหยุ่นค่อนข้างสูง โดยถือว่าเป็นแพลตฟอร์มที่ไม่ขึ้นกับฮาร์ดแวร์ และเป็นแพลตฟอร์มที่เป็นแบบเสียบแล้วเล่นได้เลย (Plug and play) สำหรับโครงงานนี้มีส่วนที่พัฒนาดังนี้

1) วงจรโมดูลอุปกรณ์ยูเอสบี (USB Device)
2) เฟิร์มแวร์ของโมดูลอุปกรณ์ยูเอสบี (USB Device Firmware)
3) ดีไวซ์ไดรเวอร์(Device Drivers)
4) ดีไวซ์ไลบรารี (Device Library)

3.2 การพัฒนาฮาร์ดแวร์ และซอฟแวร์ของแพลตฟอร์ม

3.2.1 การพัฒนาฮาร์ดแวร์ของแพลตฟอร์ม

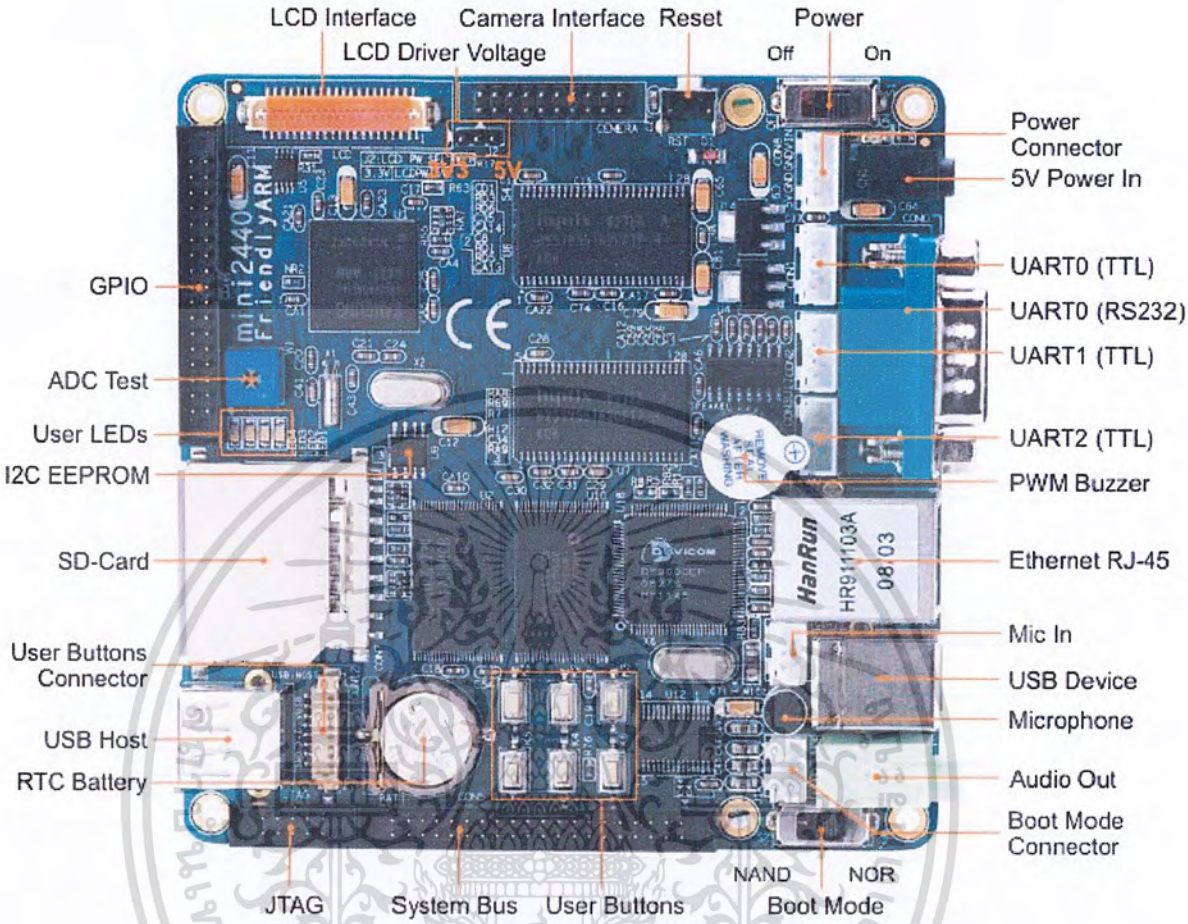
เนื่องจากโครงงานนี้เป็นโครงงานที่ใช้ลินุกซ์เป็นพื้นฐานสำคัญในการพัฒนาดังนั้นฮาร์ดแวร์ที่นำมาใช้เป็นโฮสมีความจำเป็นที่จะต้องสามารถใช้งานลินุกซ์เคอร์เนลได้ และต้องมียูเอสบีโฮสบนฮาดแวร์เป็นอย่างน้อย 1 ช่อง มีแนวทาง 2 แนวทางในการพัฒนาดังนี้

1) ออกแบบวงจรสำหรับแพลตฟอร์มใหม่โดยใช้อุปกรณ์อิเล็กทรอนิกส์ต่างๆ
2) นำวงจรสำเร็จรูปมีในท้องตลาดมาใช้งาน

จากการวิเคาะห์แล้วนั้นพบว่าการออกแบบวงจรใหม่นั้นมีความยุ่งยาก และใช้เวลานานเกินไปทางโครงงานจึงตัดสินใจเลือกใช้วงจรสำเร็จรูปมีในท้องตลาด ทางโครงงานได้เลือกบอร์ดทดลอง Friendly Arm Mini2440 ด้วยเหตุผลดังต่อไปนี้

1) สามาถใช้งานลินุกซ์ได้
2) มียูเอสบีโฮสบนบอร์ดทดลอง
3) มีอยู่แล้วทำให้ไม่ต้องซื้อใหม่

3.2.1.1 ข้อมูลทั่วไปเกี่ยวกับบอร์ดทดลอง Friendly Arm Mini2440



รูป 3.3 บอร์ดทดลอง Friendly Arm Mini2440

FriendlyARM Mini 2440 เป็นบอร์ดทดลองใช้ซีพียู with S3C2440 ARM9 ของ Samsung ที่ 400. The board measures 10 cm x 10 cm, ideal for learning about ARM systems.

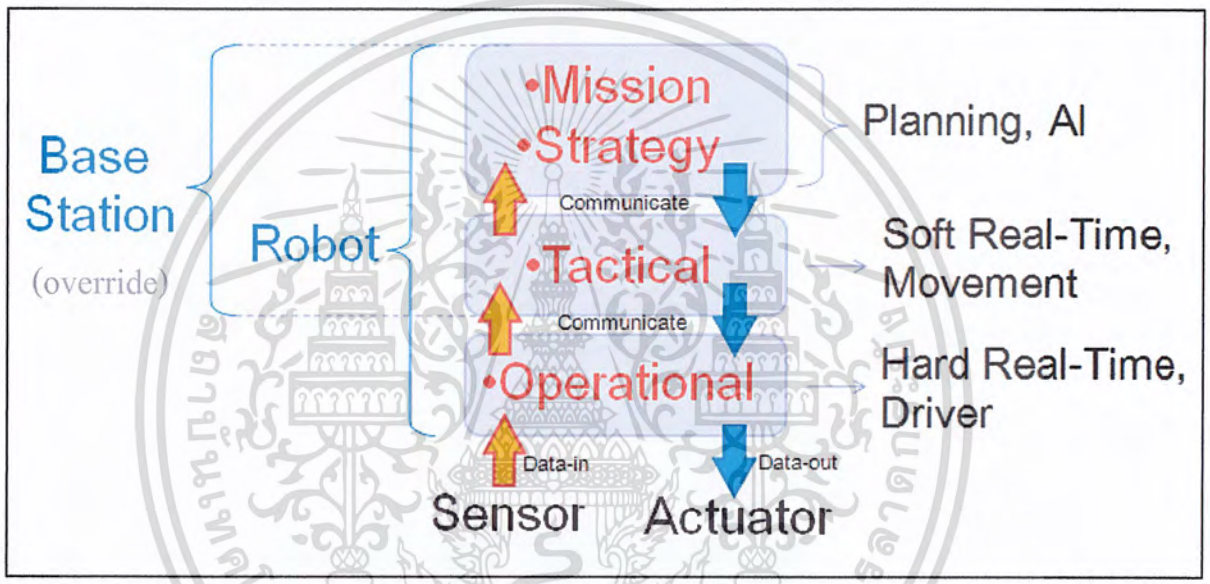
ตาราง 3.1 ข้อมูลของบอร์ดทดลอง Friendly Arm Mini2440

CPU	Samsung S3C2440A, 400MHz, Max. 533Mhz
SDRAM	- 64M SDRAM - 32bit DataBus - SDRAM Clock 100MHz
Flash	- 256M Nand Flash, - 2M Nor Flash,BIOS installed
LCD	- 4 wire resistive touch screen interface - Up to 4096 color STN,3.5 inches to12.1 inches, up to 1024x768 pixels - Up to 64K color TFT,3.5 inches to 12.1 inches, up to1024x768 pixels
Interface and Resource	- 1 10/100M Ethernet RJ-45(DM9000) - 3 Serial Port - 1 USB Host , 1 USB Slave Type B - 1 SD Card Interface - 1 Stereo Audio out, 1 Micro I - 1 20-Pin JTAG - 4 USER LEDs, 6 USER buttons - 1 PWM Beeper - 1 POT can be used for A/D converter adjust - 1 AT24C08 for I2C test, 1 20-Pin Camera Interface - 1 Battery for RTC - Power In(5V), with switch and lamp
Oscillator Frequency	- 12MHz
RTC	- Internal
Expand Interface	- 1 34-Pin 2.0mm GPIO - 1 40-Pin 2.0mm System Bus
Dimension	- 100 x 100(mm)
OS Support	- Linux 2.6 - Android - WinCE 5 and 6

3.2.2 การพัฒนาซอฟแวร์ของแพลตฟอร์ม

แนวคิดในการออกแบบซอฟแวร์ในแพลตฟอร์มนี้ได้ออกแบบซอฟแวร์บนตัวหุ่นยนต์ให้สามารถรันโปรแกรมหลายๆโปรแกรมหรือหลายๆเทรด พร้อมๆกัน โดยมีการติดต่อรับ-ส่งข้อมูลระหว่างกันหากจำเป็น โดยทำการแบ่งกลุ่มโปรแกรมออกตามพฤติกรรมของโปรแกรมหรืองานหลักๆของโปรแกรม เป็น 3 ชั้นใหญ่ๆดังนี้

1) ชั้นบน – แผนการ, ภาระกิจ (Mission&Strategy)
2) ชั้นกลาง – ยุทธวิธี (Tactical)
3) ชั้นล่าง – ดำเนินการ (Operational)



รูป 3.4 ซอฟแวร์เลเยอร์ (Software Layer)

3.2.2.1 ชั้นบน – แผนการ, ภาระกิจ (Mission&Strategy)

เป็นกลุ่มโปรแกรมที่เรียกว่าชั้นบน งานหลักๆของโปรแกรมกลุ่มนี้ จะทำการกำหนดและตรวจสอบภารกิจหลักของหุ่นยนต์ เป็นการประมวลเพื่อสร้างแผนการหรือวิธีการในการแก้ไขปัญหาให้ลุล่วงตามภารกิจที่ได้รับมอบหมาย โดยปกติจะทำงานก่อนเริ่มภารกิจหรือเข้ามาแก้ไขปัญหาในกรณีที่แผนการที่วางไว้ก่อนหน้านี้ไม่สามารถดำเนินการต่อได้เนื่องจากเกิดปัญหาต่างๆในขณะปฏิบัติการ โปรแกรมกลุ่มนี้จะทำการรับ-ส่งข้อมูลกับกลุ่มโปรแกรมชั้นกลางและมีการตื่นมาทำงานตามกำหนดการน้อยที่สุด เช่น ภารกิจเก็บภาพทั้ง 10 จุดตามตำแหน่งที่ได้กำหนดไว้ โปรแกรมจะทำการคำนวณหาระยะทางที่ใกล้ที่สุด หรือลำดับการเก็บภาพ เป็นต้น

3.2.2.2 ชั้นกลาง – ยุทธวิธี (Tactical)

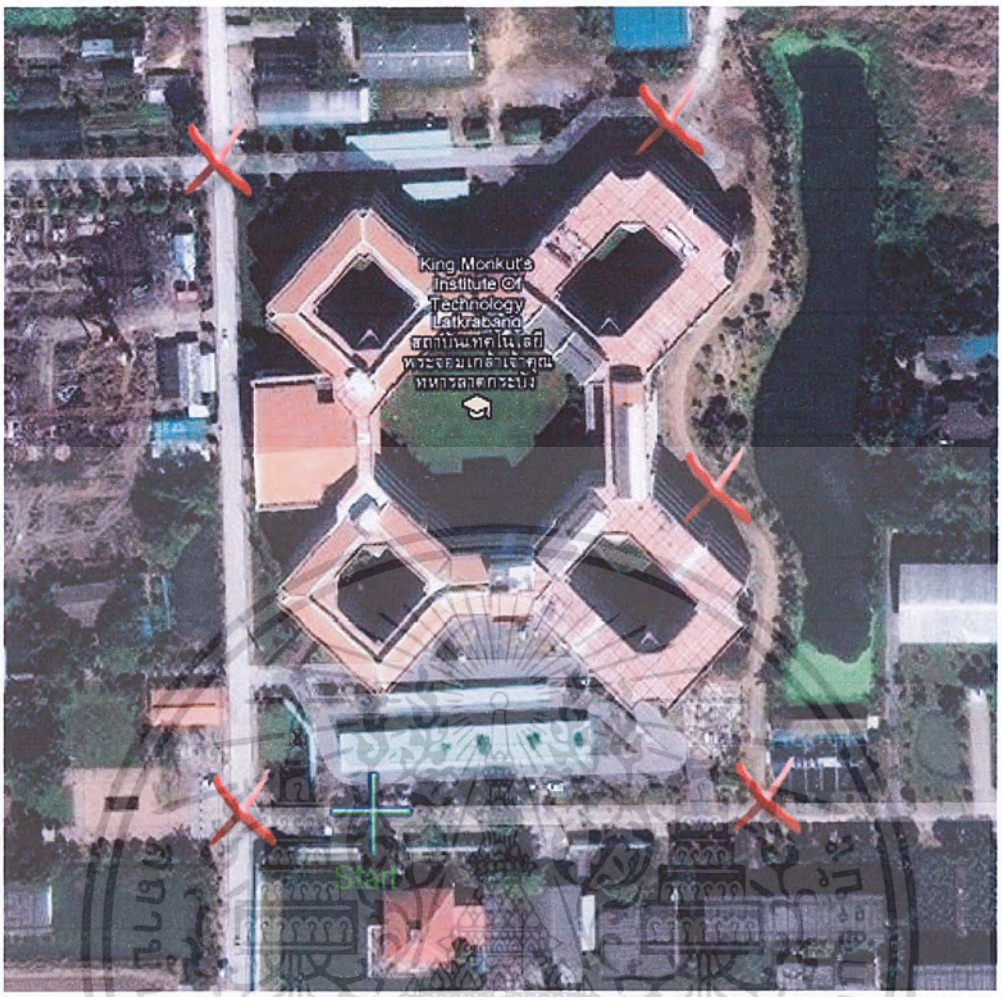
เป็นกลุ่มโปรแกรมที่เรียกว่าชั้นกลาง งานหลักๆของโปรแกรมกลุ่มนี้ จะทำการปฏิบัติงานส่วนย่อยของภาระกิจหลักซึ่งอาจเป็นงานที่มีรายละเอียดสูง มีปฏิสัมพันธ์กับสภาพแวดล้อมมากว่าโปรแกรมระดับแผนงาน มักจะเป็นภารกิจขนาดสั้นๆหรือชั่วคราว โปรแกรมกลุ่มนี้จะรับคำสั่งและรายงานผลการดำเนินการหรือข้อข้อผิดพลาดให้กับโปรแกรมชั้นบน และสั่งงานและรับผลการดำเนินงานจากโปรแกรมในกลุ่มดำเนินการ โปรแกรมกลุ่มนี้มีการตื่นมาทำงานเกือบตลอดเวลา เช่น เดินไปยังตำแหน่งใดตำแหน่งหนึ่งโดยใช้ข้อมูลจากโปรแกรมในกลุ่มดำเนินการ และหากใกล้ชนวัตถุใดอาจจะทำหนดให้ทำการหลบหลีกอัตโนมัติและกลับเข้าเส้นทางเดิมหรือคำนวณการเดินทางใหม่ เป็นต้น

3.2.2.3 ชั้นล่าง – ดำเนินการ (Operational)

เป็นกลุ่มโปรแกรมที่เรียกว่าชั้นล่าง งานหลักๆของโปรแกรมกลุ่มนี้ จะรับผิดชอบการในการสั่งการอุปกรณ์โดยจะต้องพยายามทำคำสั่งที่ได้รับมอบหมายมา พร้อมทั้งควบคุมคุณภาพข้อมูลที่ได้รับจากเซนเซอร์ให้มีความถูกต้องและเหมาะสมที่จะนำไปใช้งานต่อไป โดยจะทำการติดต่อติดต่อ ระบุตัวตน สั่งงาน และรับค่าจากอุปกรณ์ภายนอกทั้งหมด โดยแต่ละโปรแกรมที่รันหมายถึงอุปกรณ์แต่ละอย่างที่ทำการเชื่อมต่อกับตัวบอร์ด โปรแกรมกลุ่มนี้จะรับคำสั่งและรายงานผลการดำเนินการหรือข้อข้อผิดพลาดให้กับโปรแกรมชั้นกลาง สั่งงานและรับค่าจากตัวอุปกรณ์ โปรแกรมกลุ่มนี้จะทำงานอยู่ตลอดเวลา เช่น ยูเอสบีไวไฟหนึ่งอันก็จะมีโปรแกรม 1 โปรแกรมที่ใช้ในการเชื่อมต่ออุปกรณ์และ กำหนดลักษณะการเชื่อมต่อต่างๆ หรือไอพี (IP) ต่างๆ เป็นต้น

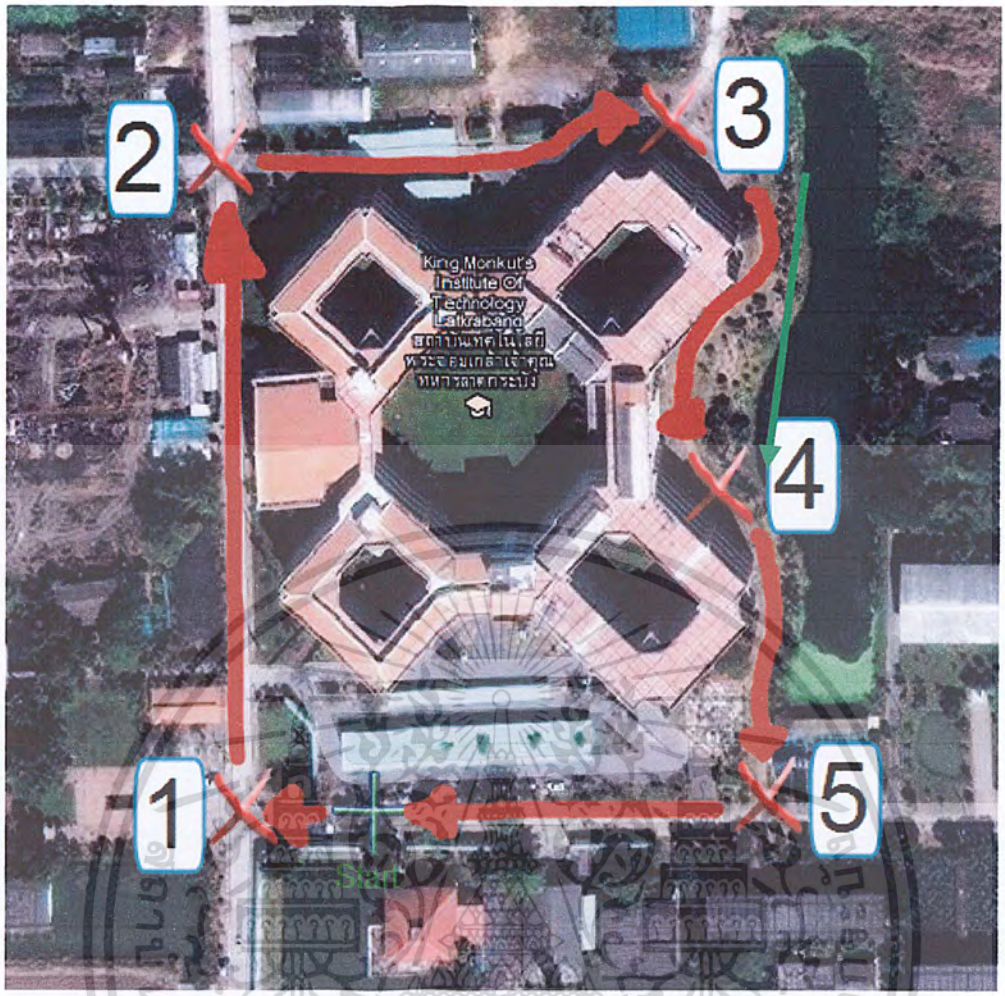
3.2.2.4 ตัวอย่างภารกิจและหน้าที่การดำเนินงานของโปรแกรมแต่ละชั้น

ภารกิจตัวอย่างกำหนดให้หุ่นยนต์เดินทางไปยังจุดต่างๆ 5 จุด ที่ได้กำหนดไว้ในแผนที่พร้อมทั้งถ่ายภาพรอบด้าน และกลับมายังจุดเริ่มต้น



รูป 3.5 แผนที่และตำแหน่งที่ได้รับมอบหมาย

เมื่อได้รับภารกิจแล้วโปรแกรมชั้นบนสุดจะเป็นผู้รับผิดชอบในการกำหนดเป้าหมายโดยคำนวณหาลำดับขั้นตอนการปฏิบัติภารกิจ จากจุดเริ่มต้นไปยังจุดต่างๆและกลับมายังจุดเริ่มต้นอีกครั้งหนึ่ง เมื่อประมวลผลเสร็จจะได้แผนการดำเนินการตามลำดับดังรูป



รูป 3.6 ลำดับการปฏิบัติภาระกิจ

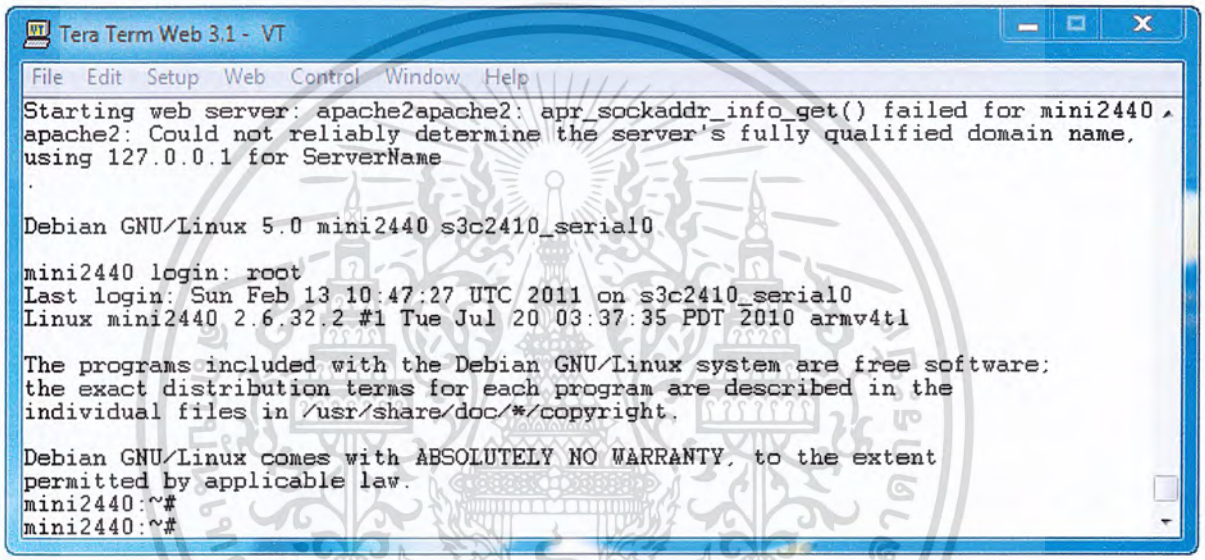
ส่วนโปรแกรมชั้นกลางหรือชั้นยุทธวิธีนั้นจะได้รับคำสั่งจากชั้นบนให้เดินทางจากจุดเริ่มต้นไปยังจุดที่1 เมื่อเสร็จภารกิจจะส่งผลการทำงานกลับไปยังโปรแกรมชั้นบน จากนั้นโปรแกรมชั้นบนจะสั่งให้ถ่ายรูปรอบตัว เมื่อทำภารกิจเสร็จจะส่งผลการทำงานกลับไปยังโปรแกรมชั้นบน ทำเช่นนี้ไปเรื่อยๆจนครบทั้ง 5 จุด โดยการทำงานของโปรแกรมในชั้นกลางจะเห็นได้ชัดเมื่อทำการเดินทางจากจุดที่ 3 ไปยังจุดที่ 4 โดยเริ่มแรกจะเดินทางตามลูกศรสีเขียวแต่เมื่อเดินทางได้ซักพักจะพบว่าเดินติดต้นไม้ โปรแกรมในชั้นกลางนี้จะค่อยๆเดินอ้อมจนถึงจุดที่4 ตามภารกิจที่ได้รับมอบหมาย

ส่วนโปรแกรมชั้นล่างจะรับผิดชอบสั่งการล้อให้หมุนตามที่ได้รับคำสั่งจากโปรแกรมชั้นกลาง และรับค่าตำแหน่งจีพีเอส (GPS) หาค่าความถูกต้องแล้วส่งไปให้โปรแกรมชั้นกลางเพื่อสั่งดำเนินการต่อไป หรือหากสั่งเดินไปข้างด้านแล้วไม่สามารถเดินได้เนื่องจากอาจจะติด

ก้อนหินขนาดเล็กที่เซนเซอร์ไม่สามารถตรวจสอบได้ ก็จะทำการส่งข้อผิดพลาดไปยังโปรแกรมชั้นกลางเพื่อหาทางดำเนินการหาทางแก้ไขต่อไป

3.2.2.5 ระบบปฏิบัติการของแพลตฟอร์มของบอร์ดหลักบนหุ่นยนต์

สำหรับแพลตฟอร์มนี้จะใช้ระบบปฏิบัติการลินุกซ์ (Linux) เคอร์เนล2.6 (Kenel2.6) ขึ้นไป โดยเลือกตัวที่เหมาะสมกับอุปกรณ์และบอร์ดหลัก และสามารถติดตั้งลงบนบอร์ดที่เลือกมาเป็นบอร์ดหลักของหุ่นยนต์ได้ โดยจะใช้จอแสดงผลแอลซีดี (LCD) ขนาดเล็ก หรืออุปกรณ์อื่นๆร่วมด้วยหรือไม่ก็ได้แล้วแต่ความถนัดและประโยชน์ของผู้นำไปใช้



รูป 3.7 ข้อมูลที่ได้จากการรันลินุกซ์ผ่านโปรแกรมเทระเทิร์ม (Teara Term Web)

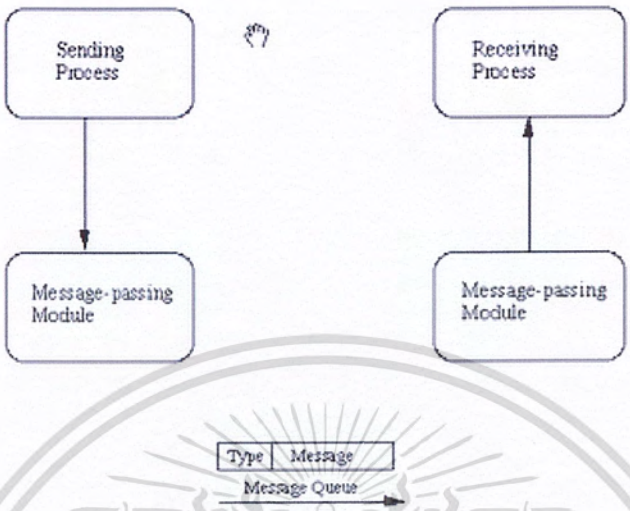
3.2.2.6 การพัฒนาโปรแกรมบนหุ่นยนต์

สำหรับการเขียนโปรแกรมในแพลตฟอร์มนี้จะเขียนภายใต้แนวคิดโปรแกรมมิ่งเชิงวัตถุ (Object-Oriented Programming) โดยใช้ภาษาซีพลัสพลัส (C++) เป็นภาษาหลักในการพัฒนา และใช้ไลบราอื่นๆหากจำเป็น โดยสามารถใช้การเขียนและสร้างโปรแกรมข้ามแพลตฟอร์ม (Cross-compile) หพรือทำการเขียนหรือสร้างบนบอร์ดหลักของหุ่นยนต์ได้เลยหากบอร์ดหลักที่นำมาใช้นั้นมีประสิทธิภาพเพียงพอต่อการใช้งานในการพัฒนา

3.2.2.7 การติดต่อสื่อสารระหว่างโปรแกรมในแพลตฟอร์ม

สำหรับการติดต่อสื่อสารกันระหว่างโปรแกรมในแพลตฟอร์มจะใช้แบบใดก็ได้แล้วแต่ผู้พัฒนาหรือผู้นำแพลตฟอร์มไปใช้สะดวกพัฒนา ในที่นี้ผู้พัฒนาได้ใช้หลักการเมสเสส

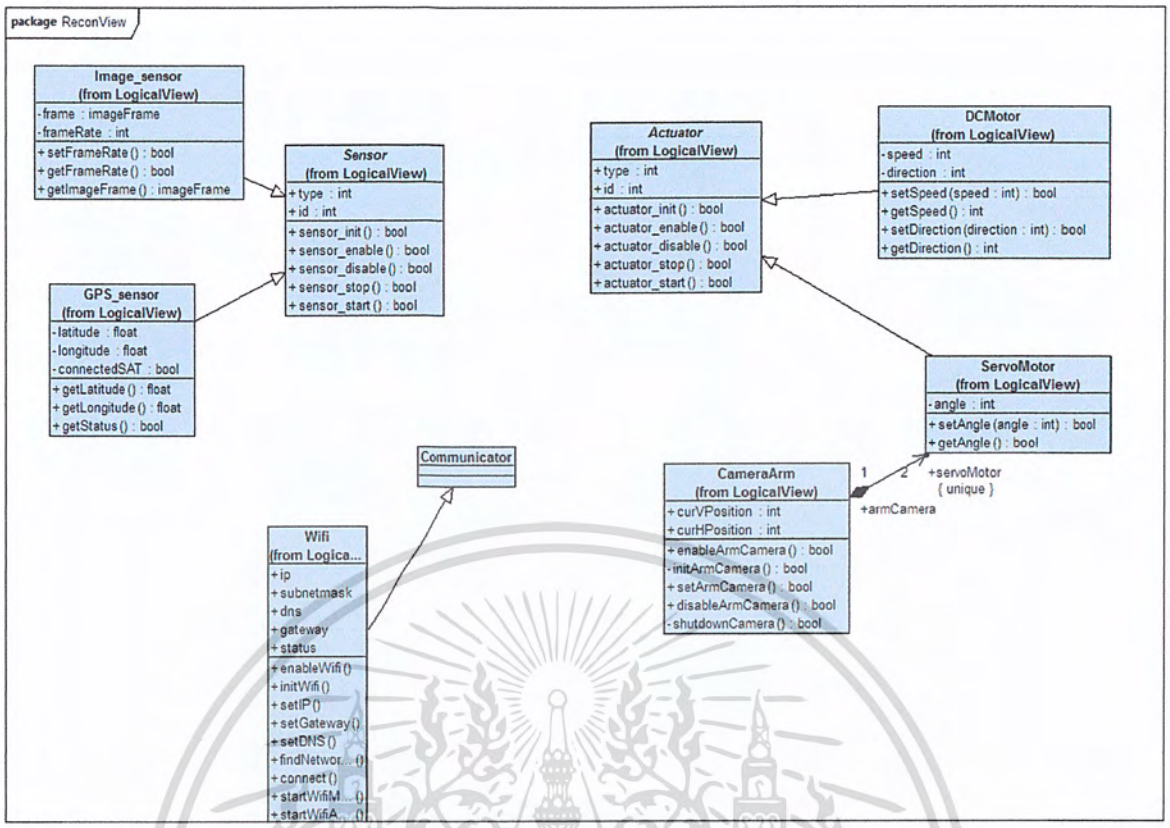
คิว (Message Queue) เพราะมีความสะดวก ไม่ซับซ้อน โดยคีย์ที่ใช้นั้นให้ผู้ที่ใช้แพลตฟอร์มใช้คีย์ตามที่ผู้พัฒนาได้กำหนดไว้ในเอกสารประกอบการใช้งาน



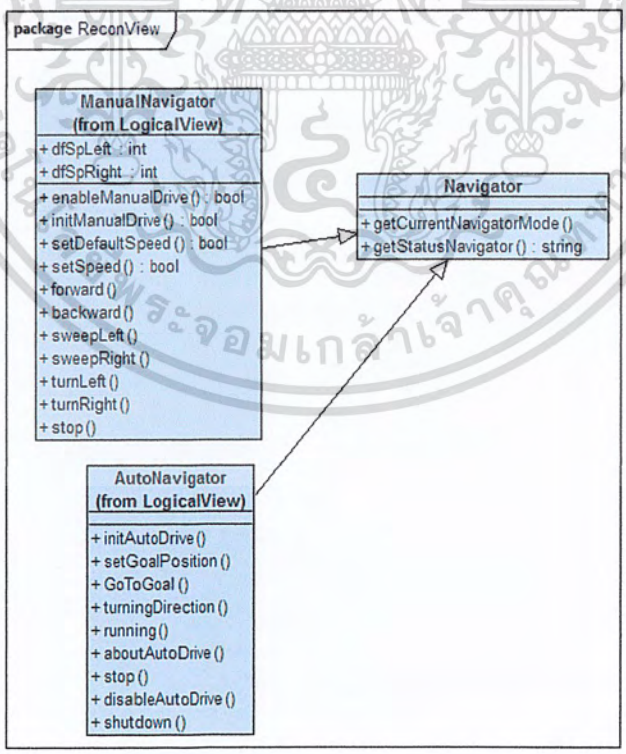
รูป 3.8 basic message passing

3.2.2.8 ตัวอย่างยูเอ็มแอลที่ใช้ออกแบบคลาสของโปรแกรมต่างๆในแพลตฟอร์ม

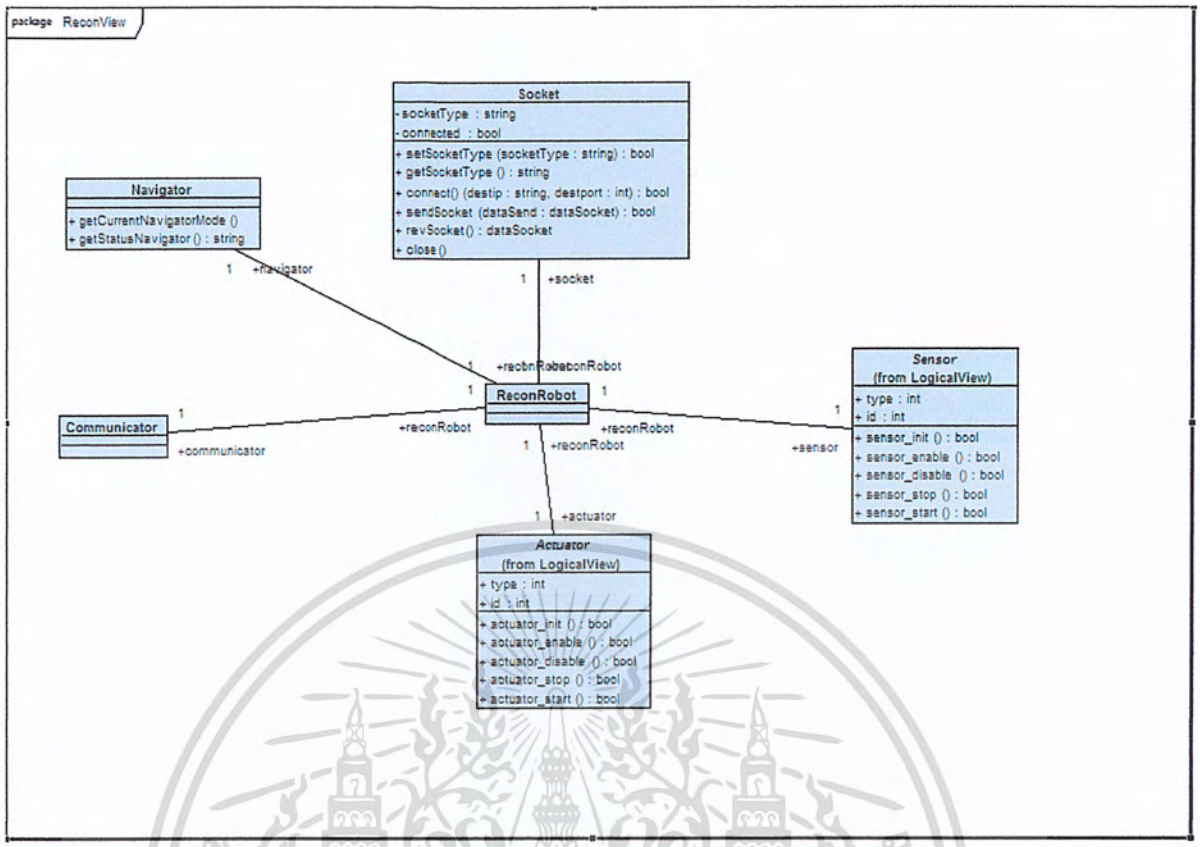
เป็นตัวอย่างการใช้โปรแกรมอีคลิปส์ (Eclipse) ร่วมกับท็อปเคส (Topcased) ในการออกแบบคลาสของแต่ละโปรแกรมเพื่อช่วยเพิ่มประสิทธิภาพในการออกแบบให้มีความสมบูรณ์ยิ่งขึ้น



รูป 3.9 Class Diagram for Operational Programming Layer



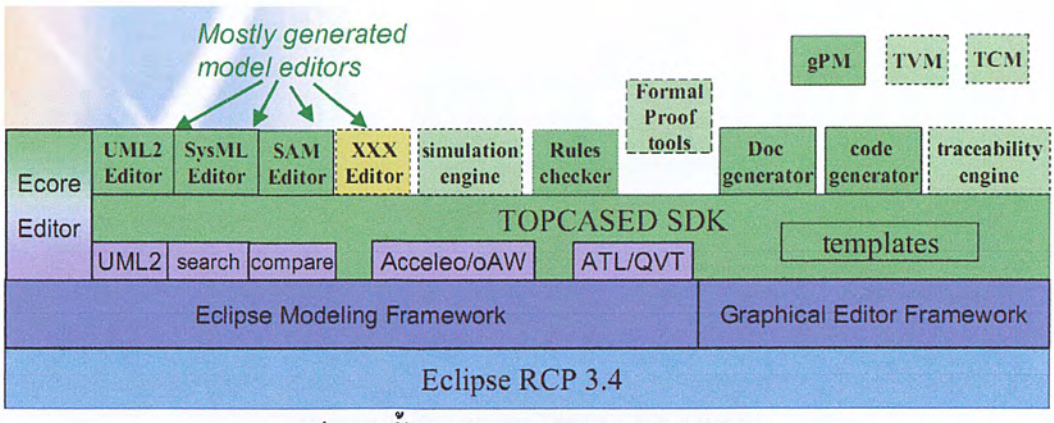
รูป 3.10 Class Diagram for Tactical Programming Layer



รูป 3.11 Class diagram for main program use another class

3.3 กรณีศึกษาการออกแบบระบบ Coyote UAV ด้วย UML

สำหรับการศึกษากรณีศึกษาการออกแบบระบบด้วย Coyote UAV ด้วย UMLนั้น เป็นกรณีศึกษาตามหนังสือชื่อว่า Real-Time UML Workshop for Embedded Systems เขียนโดย Bruce Powel Douglass, Ph.D พิมพ์ในปีพุทธศักราช 2550 เพื่อเป็นแนวทางในการออกแบบระบบของแพลตฟอร์มของโครงงาน โดยในหนังสือได้ใช้โปรแกรม Rhapsody เป็นเครื่องมือในการออกแบบทางโครงงานได้นำมาทำใน Topcased (Toolkit in OPen-source for Critical Applications & SystEms Development) ซึ่งเป็นฟรีโปรแกรมที่ทำงานควบคู่กับโปรแกรม Eclipse โดยมีแผนผังการทำงานดังรูปด่างล่าง

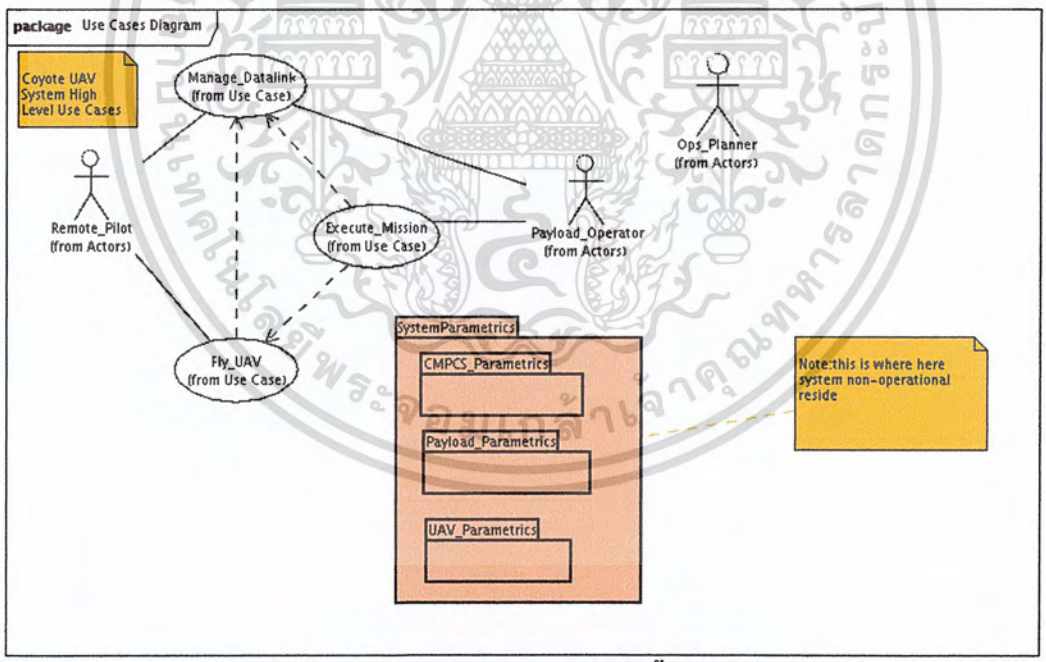


รูป 3.12 ชั้นการทำงานของ TOPCASED

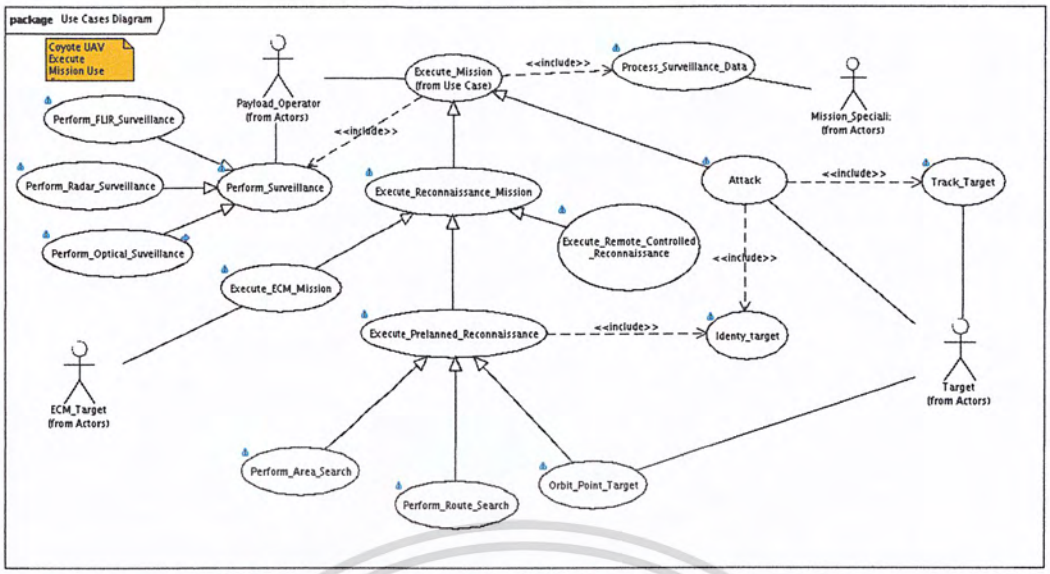
โดยมีการศึกษาตามหนังสือดังนี้

3.3.1 ระบุ Use Case ของ Coyote UAV

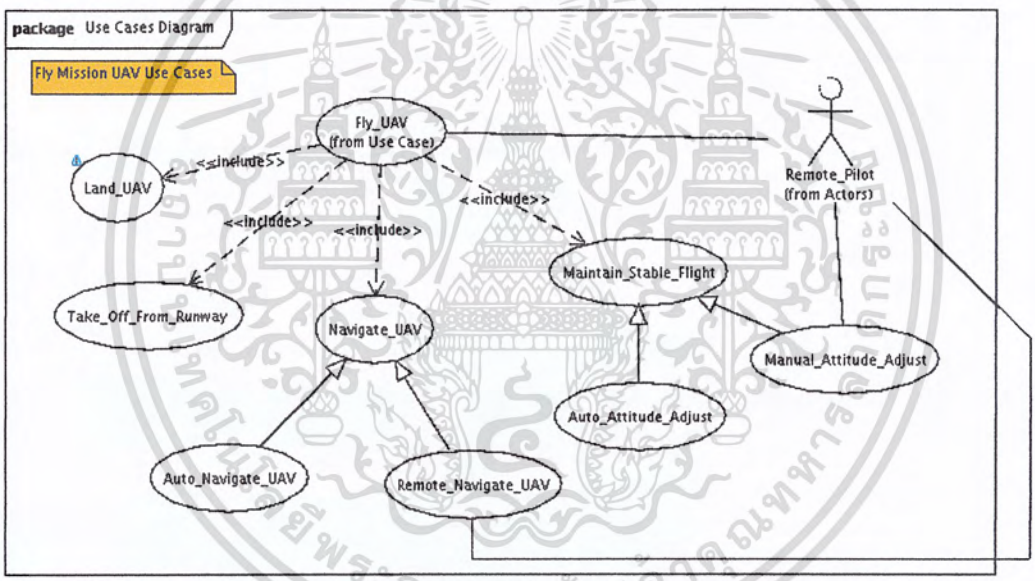
ในปัญหานี้ได้ทำการศึกษา Use Case Diagram ที่ใช้ในระบบ พบว่าในการออกแบบ Coyote UAV นั้น ทำให้เกิด Use Case มากมาย จึงต้องแบ่ง Use Case Diagram ออกเป็นหลายชั้นหลายระดับ ดังตัวอย่างที่แสดงข้างล่างนี้



รูป 3.13 Use Cases Diagram ในชั้นบนสุด



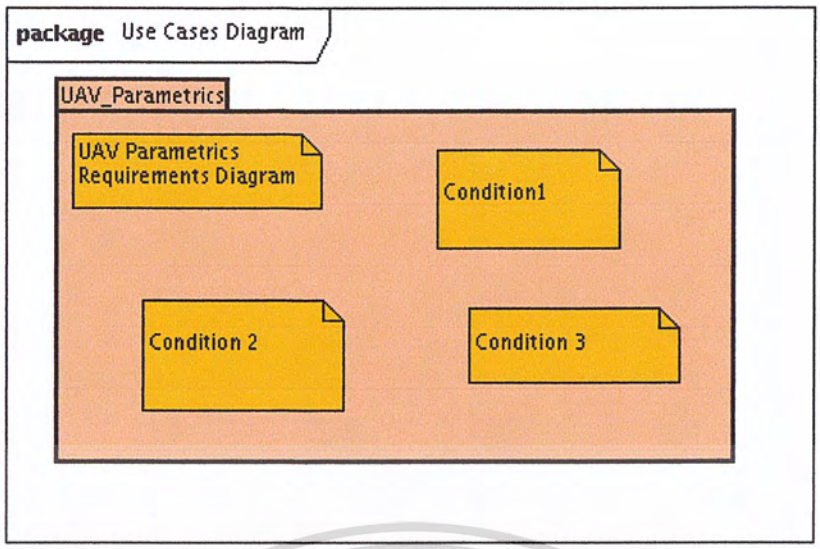
รูป 3.14 Use Cases Diagram ในส่วนของ Execute Mission



รูป 3.15 Use Cases Diagram ในส่วนของ Fly UAV

3.3.2 การระบุตัวแปลต่างๆของระบบ

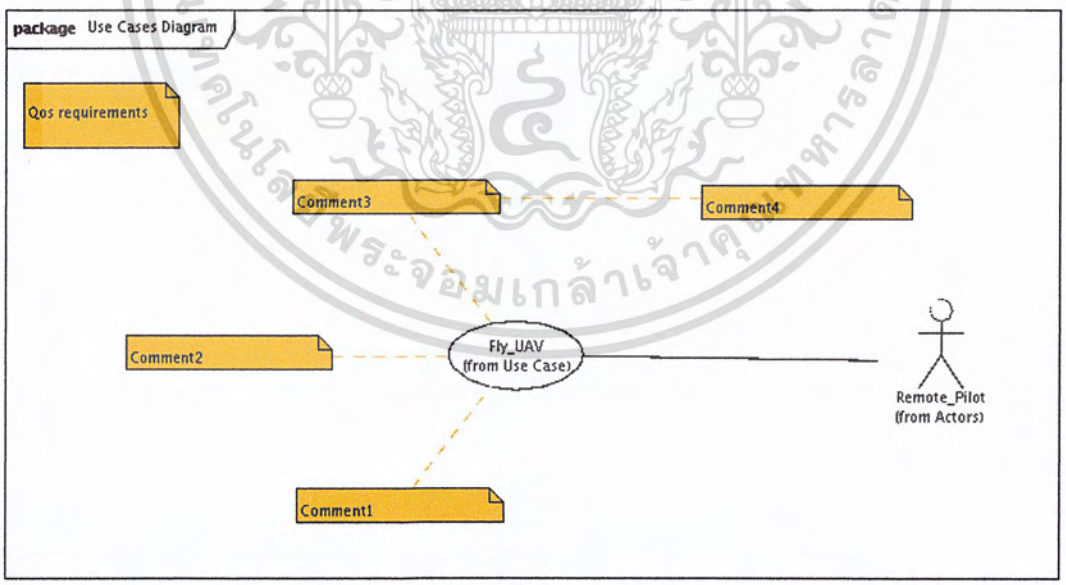
เป็นการระบุตัวแปลต่างๆที่จำเป็นต้องใช้ในระบบ หรือเป็นสิ่งที่ระบบห้ามทำ หรือทำได้ โดยมีลักษณะคล้ายการทำหมายเหตุ



รูป 3.16 ตัวอย่างโครงสร้าง Use Cases Diagram ในส่วนของ UAV Parametric

3.3.3 QoS (Quality of Service) requirement

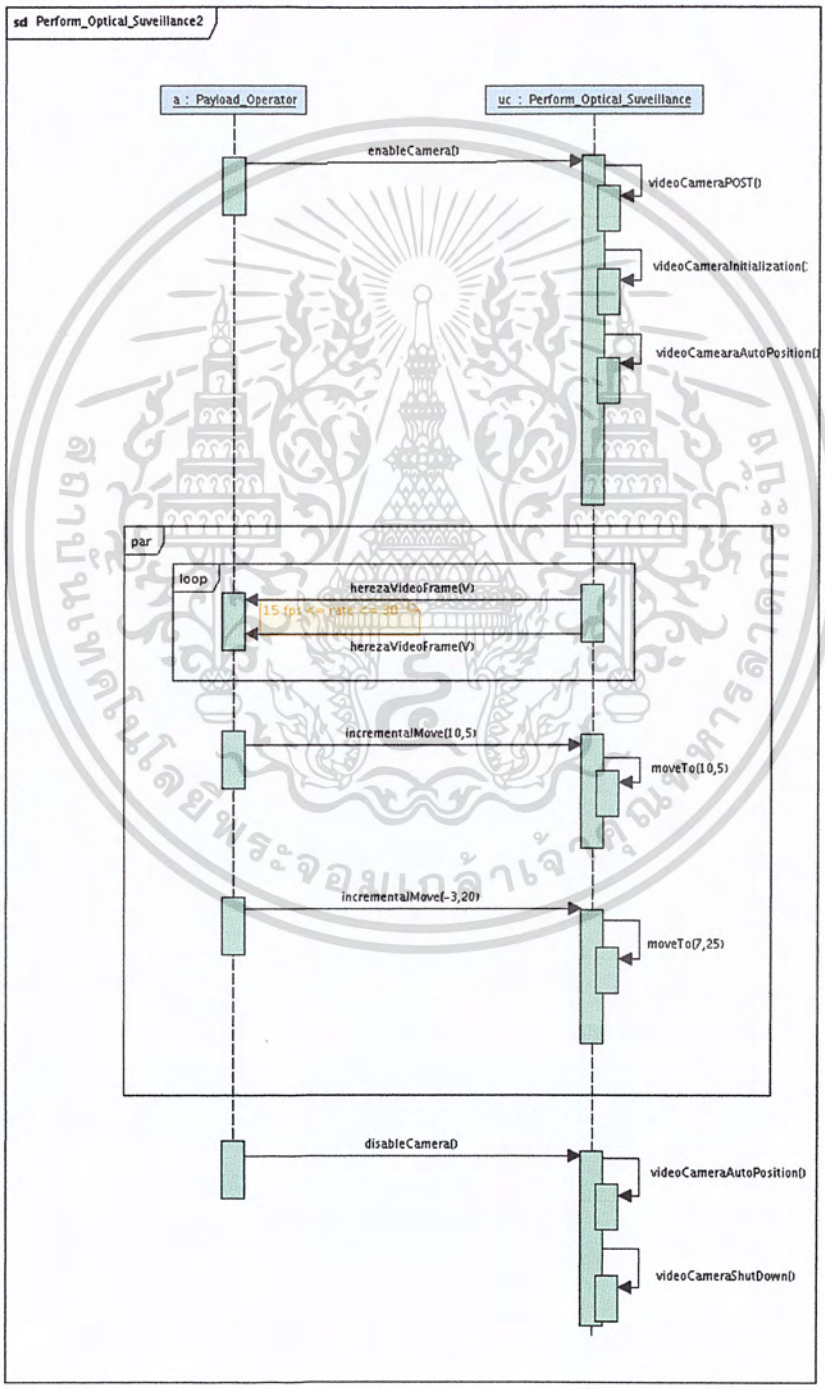
QoS requirement คือ ข้อกำหนดแบบนึงที่ระบบพึงมี โดยเฉพาะระบบที่เป็นแบบเรียลไทม์ (Real-Time Systems) เพราะเป็นข้อกำหนดเกี่ยวกับปราณต่างๆในระบบ เช่น ความเร็วภาพพื้นดินขั้นต่ำที่พึงมี ความเร็วในการรับและส่งข้อมูล เพดานบินและความเร็วสูงสุด เป็นต้น ดังแสดงตามตัวอย่างข้างล่างนี้



รูป 3.17 ตัวอย่างโครงสร้าง Use Cases Diagram ในส่วนของ QoS requirement

3.3.4 แผนภาพของแผนการทำงาน (Scenaios)

ในที่นี้เป็นกรณีศึกษาเฉพาะส่วนของ CUAV Optical Surveillance โดยแสดงให้เห็นถึงลำดับการทำงาน การเรียกใช้ และการส่งข้อมูล ของแต่ละการทำงาน โดยระบบสามารถมีแผนภาพของแผนการทำงาน (Scenaios) ได้หลายอัน รูปด้านล่างนี้แสดงให้เป็นถึงการการทำงานในส่วนของการเปิดใช้กล้อง การส่งภาพกลับ การควบคุมกล้อง และการปิดการใช้กล้อง โดยมีผู้เกี่ยวข้องกับงานคือ Payload Operator



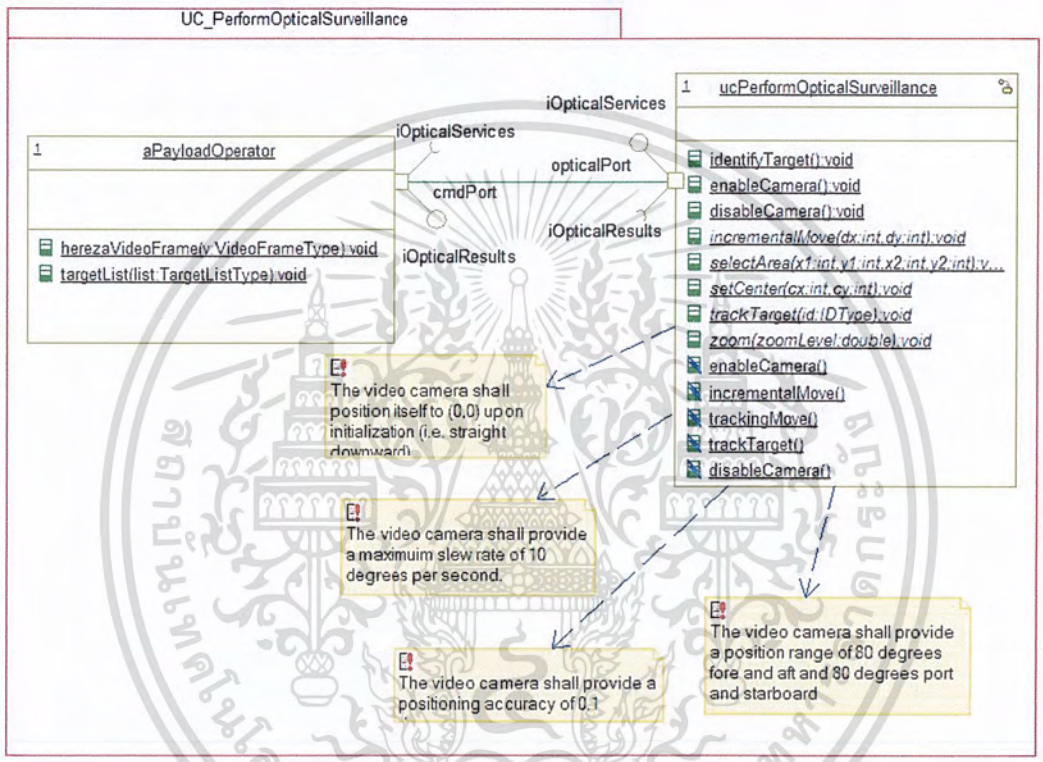
รูป 3.18 Use Cases Diagram ในส่วนของ CUAV Scenario1

3.3.5 สร้างมุมมองการดำเนินงาน

แบ่งออกเป็น 5 ขั้นตอนดังนี้

3.3.5.1 เขียน Block Diagram

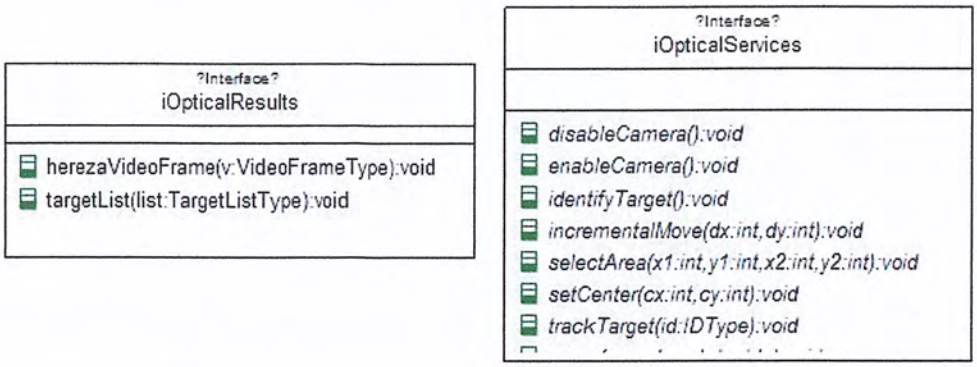
ในที่นี้เป็นการศึกษา Block Diagram ของ Perform Optical Surveillance โดยทั้ง Use-Cases และ Actor ได้ถูกเขียนให้อยู่ในรูปแบบของ block (objects) และระบุพอร์ต (port) และส่วนติดต่อ (interface) อย่างชัดเจน



รูป 3.19 Block Diagram ของ Perform Optical Surveillance

3.3.5.2 กำหนดส่วนติดต่อ (Interface)

สำหรับในส่วนติดต่อนี้จะบรรจุไปด้วยการดำเนินการที่เป็น public เท่านั้น เพื่อแสดงให้เห็นว่าสามารถใช้ใช้งานส่วนไหนได้บ้าง



รูป 3.20 ส่วน interface ของ Perform Optical Surveillance

3.3.5.3 กำหนดเงื่อนไข ก่อน-หลัง ของส่วนติดต่อ

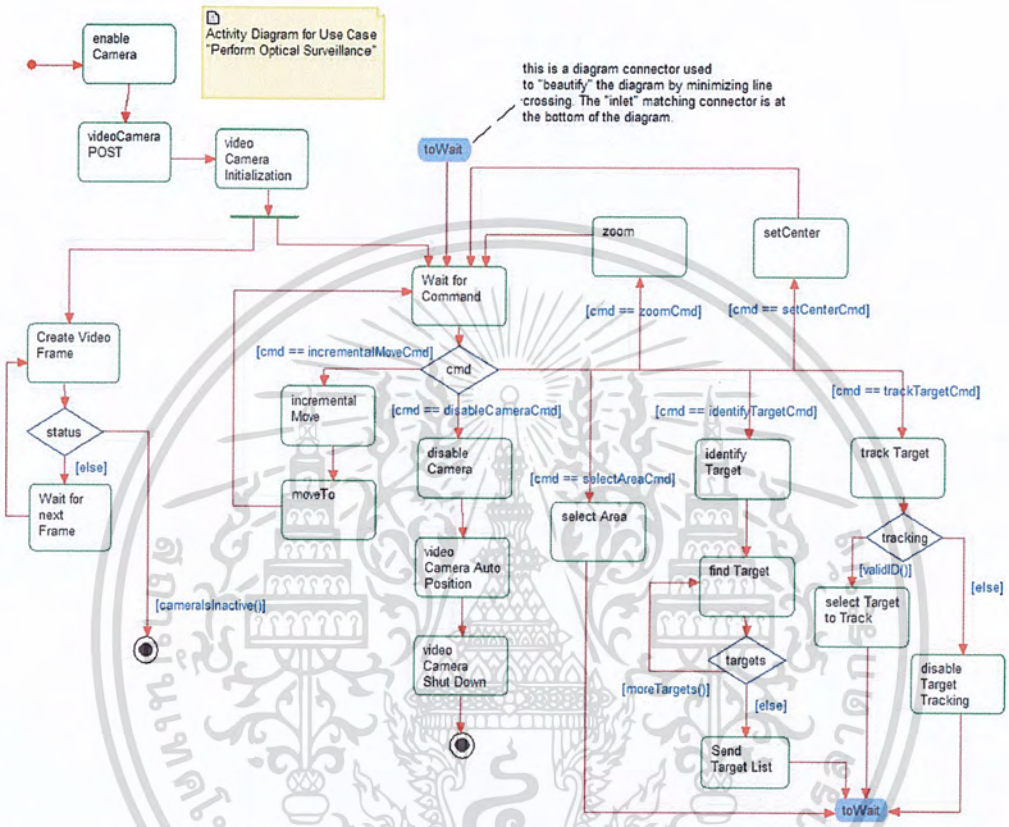
ตัวอย่างตารางที่เขียนเงื่อนไขก่อน-หลังของการทำงานในส่วนติดต่อ โดยจะต้องเป็นจริงทั้ง ก่อนและหลังที่ถูกเรียกใช้งาน

ตาราง 3.2 เงื่อนไข ก่อน-หลัง ของส่วนติดต่อของ Perform Optical Surveillace

Interface	Service	Pre-condition	Post-condition
iOpticalResults	herzaVideoFrame	Video camera is operating	Frame is stored in frame buffer for viewing and analysis
	targetList	Targeting enabled; identifyTarget command previously sent	List of targets is stored for analysis
iOpticalServices	disableCamera	None	Camera is returned to starting position and shut down
	enableCamera	None	Camera is powered and a Power On Self Test is performed, If errors occur, then error codes are sent; otherwise, the camera is initialized and centered.
	identifyTarget	Camera is operating and search area or route has been specified.	System identifies a list of targets, each with a unique target ID.
	incrementalMove	Camera is operating.	If the incremental move is within the range of the gimbaled camera mounting, then the camera is moved incrementally +x units and +y units. If the commanded position is out of range, then the camera will move to the closest valid position.
	selectArea	Camera is enabled	If the area is within the limits of the system, then the search area is defined. If not, then an error is sent and the closest matching search area will be selected.
	setCenter	Camera is enabled	If specified point is in range, the point is set to the be the search center. If not, an error is returned and the center remains unchanged.
	trackTarget	Camera is on and potential targets have been tagged with IDs	If the target ID is known to the system, then it will track that target within the center of the visual field with a combination of flight maneuvers and camera gimbal movement, depending on the state of the CAUV.
	zoom	Camera is on.	If the zoom parameter is within bounds, then the camera shall zoom in or out, auto adjusting focus. If the zoom level is out of range, then the camera shall clip at the minimum or maximum zoom, whichever is closer. The units are percentage, so that a parameter value of 300 means 3x magnification, 50 means 0.5 magnification.

3.3.5.4 สร้าง Use-Case Activity Diagram *ตาม* Scenarios *ทั้งหมดที่มี*

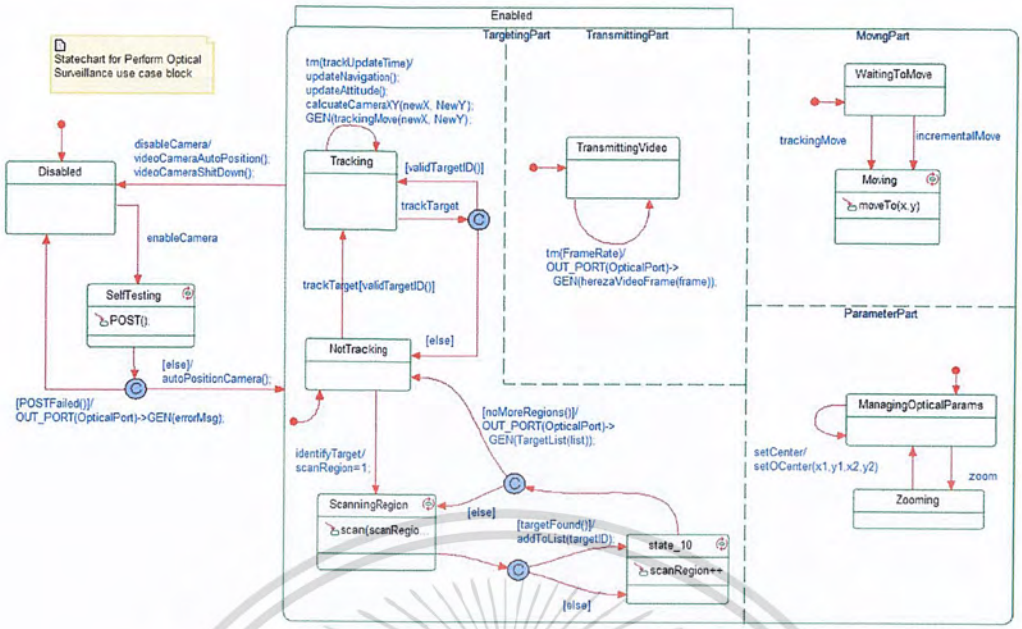
นี่คือการแสดงภาพรวมทั้งหมดของทุกๆ Scenarios ในหนึ่งเดียว ถือว่าเป็นประโยชน์อย่างมาก รูปด้างล่างนี้แสดงกิจกรรมทั้งหมดที่อาจเกิดขึ้นได้ใน Perform Optical Surveillance



รูป 3.21 กิจกรรมของ Perform Optical Surveillance

3.3.5.5 สร้างแผนภาพแสดงสถานะ*เครื่องจักร* (state machine) *ของ* Use-Case Block

งานในขั้นสุดท้ายคือสร้าง statechart สำหรับ use-case block ที่เรียกใช้งานพฤติกรรมอย่างถูกสถานการณ์ ถูกเวลา และถูกลำดับก่อน-หลัง



รูป 3.22 state chart ของ Perform Optical Surveillance

3.4 การพัฒนาโมดูลต่างๆของหุ่นยนต์ให้เป็นอุปกรณ์แบบยูเอสบี

เป็นลักษณะการพัฒนาชิ้นส่วนต่างๆ ของหุ่นยนต์ให้เป็นรูปแบบของ ยูเอสบีดีไวซ์ ซึ่งสามารถปั๊กอินเข้าทางพอร์ตยูเอสบี เพื่อใช้งาน ซึ่งจะเป็นการนำข้อดีต่างของยูเอสบีมาใช้กับการพัฒนาหุ่นยนต์ เช่น

1) มีความเร็วในการรับส่งข้อมูลสูง ยูเอสบี1.0 = 1.5 เมกะบิทต่อวินาที , ยูเอสบี 1.1 = 12 เมกะบิทต่อวินาที , ยูเอสบี 2.0 = 480 เมกะบิทต่อวินาที
2) มีลักษณะของ ปั๊กแอนด์เพล์ (สามารถเรียนรู้ได้ว่าเป็นดีไวซ์แบบไหน และทำงานอย่างไรจากการอ่าน คำอธิบาย)
3) มีลักษณะเป็นบัสทำให้สามารถเชื่อมต่อได้หลายอุปกรณ์ในเวลาเดียวกัน รวมถึงสามารถต่อยูเอสบีฮับ เพื่อเพิ่มพอร์ตในการเชื่อมต่อได้(สามารถเชื่อมต่ออุปกรณ์ได้สูงสุดถึง 127 ตัว ถ้าการจัดการด้านพลังงานดีพอ)
4) ประหยัดขา หรือจีพีไอโอที่ใช้ในการเชื่อมต่อ
5) เมื่อเปลี่ยนชิ้นส่วนฮาร์ดแวร์ของยูเอสบีดีไวซ์ก็ไม่จำเป็นต้องเปลี่ยนแปลงดีไวซ์ไดร์เวอร์และ แอปพลิเคชั่น

3.4.1 การพัฒนาอุปกรณ์ยูเอสบีด้วยวี-ยูเอสบี (V-USB)

รูป 3.23 วี-ยูเอสวี

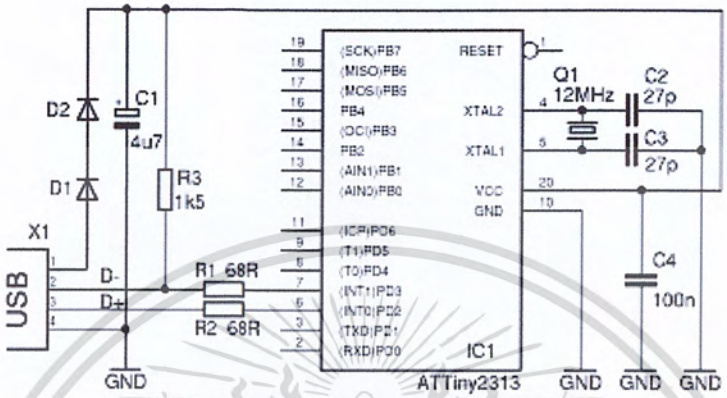
วี-ยูเอสบี หรือพอร์ตยูเอสบีเสมือนสำหรับเอวีอาร์ไมโครคอนโทรลเลอร์ (V-USB : Virtual USB port for AVR Microcontroller) เป็นชุดซอฟแวร์ที่ใช้สำหรับจำลองไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ให้สามารถจัดการกับโปรโตคอลของยูเอสบีแบบโลว์สปีด 1.1 ได้ โดยที่ไม่จำเป็นต้องใส่ชิปอื่นเพิ่ม และสามารถนำไปพัฒนาได้อย่างเสรี ภายใต้การอนุญาตของจีเอ็นยู (GNU Genneral Public License)

3.4.1.1 ลักษณะเด่นของวี-ยูเอสบี

1) ทำงานได้กับยูเอสบีแบบโลว์สปีด 1.1
2) ทำงานกับไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์ ที่มีความจำประเภทเฟรชเฟรชเมมโมรี่อย่างน้อย 2 กิโลไบต์ มีหน่วยความจำประเภท RAM อย่างน้อย 128 bytes และ Clock rate อย่างน้อย 12 MHz
3) ไม่มี UART, timer หรือ ฮาร์ดแวร์พิเศษเพิ่มเติม
4) ใช้ภาษาซีในการเขียนฟังก์ชันขั้นสูงและมีคอมเมนต์ประกอบโดยละเอียด
5) ใช้เพียง 1150 ถึง 1400 ไบต์ ของขนาดโค้ด
6) โดยพื้นฐาน สามารถส่งข้อมูลได้ถึง 254 ไบต์ และสามารถปรับเพิ่มได้
7) สนับสนุน Multiple endpoints
8) เป็น Open Source
9) ตัวอย่างโปรเจ็ค ที่ทดลองกับอุปกรณ์และโฮส มีตัวอย่างทั้งใน Linux, Max OS X และ Windows
10) ฟรี USB indentifiers (Vendor-ID, Product-ID pairs)

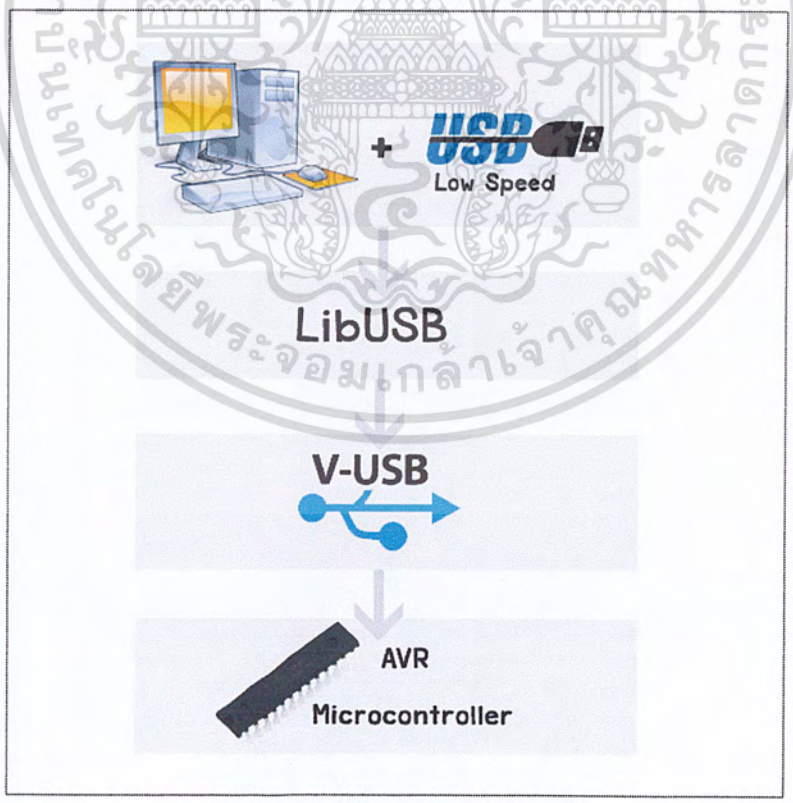
3.4.1.2 ฮาร์ดแวร์

วงจรข้างต้นเป็นวงจรตัวอย่าง แสดงเฉพาะสายสัญญาณไฟฟ้าของอุปกรณ์ โดย D1 และ D2 เป็นการแทนที่ชิปเรคกูเลเตอร์ (regulator chip) 3.3 โวลต์ เช่น LED แบบง่ายๆ ถ้าต้องการทำงานเอวีอาร์ที่ 5 โวลต์ เพิ่มไดโอดที่ขา D+ และขา D- เพื่อจำกัดกระแส



รูป 3.24 วงจรตัวอย่างสำหรับวี-ยูเอสบี

3.4.1.3 แนวทางการพัฒนาด้วย V-USB



รูป 3.25 แผนผังการพัฒนาวี-ยูเอสบี

3.4.1.4 ซอฟแวร์สำหรับการพัฒนาบนวินโดว์

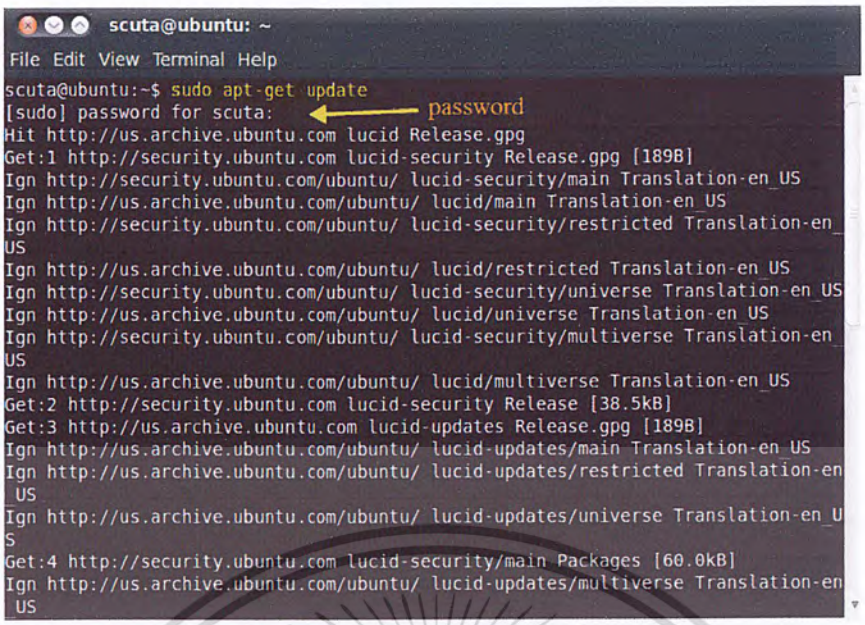
สามารถดาวโหลดได้จาก http://winavr.sourceforge.net/

1) LibUSB เป็นไลบรารีสำหรับผู้ใช้งานในระบบยูเซอร์ เลเวล (User level) ที่ต้องการติดต่อกับอุปกรณ์ยูเอสบี ที่สามารถทำงานได้หลายระบบปฏิบัติการ ซึ่ง LibUSB นั้นเป็นโอเพนซอร์ซ (Open Source) ภายใต้การอนุญาต GNU Lesser General Public License version 2.1 สามารถดาวโหลดได้จาก http://www.libusb.org
2) V-USB เป็นไลบรารีที่ใช้จำลองพอร์ตยูเอสบีสำหรับไมโครคอนโทรลเลอร์ตระกูลเอวีอาร์สามารถดาวโหลดได้จาก http://www.obdev.at/products/**vusb**/index.html
3) MinGW เป็น GNU สำหรับวินโดว์ สามารถดาวโหลดได้จาก http://www.mingw.org
4) WinAVR เป็นโปรแกรมโอเพนซอร์ซสำหรับพัฒนาไมโครคอนโทรลเลอร์เอวีอาร์บนระบบปฏิบัติการวินโดว์ ซึ่งรวมคอมไพล์เลอร์ GNU GCC สำหรับภาษาซี และซีพลัสพลัส

3.4.1.5 ซอฟแวร์สำหรับการพัฒนา Ubutu 10.4

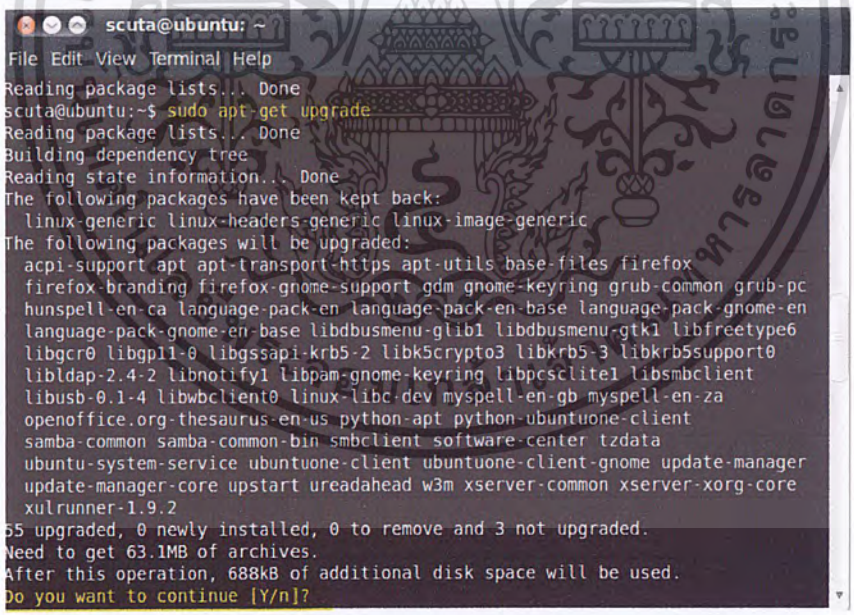
ในโครงงานนี้จะใช้ Ubuntu 10.04 ในการพัฒนาส่วนของโค้ดเป็นหลัก เนื่องจาก Ubuntu เป็นระบบปฏิบัติการลินุกซ์ที่เป็น Opensource มีผู้พัฒนาอยู่มากมายทั่วโลก และเวอร์ชั่น 10.04 เป็นเวอร์ชั่นล่าสุดในขณะนี้ ซึ่งเนื้อหาในส่วนนี้จะไม่อธิบายถึงการลง ubuntu

1) Apt-get update เป็นการอับเดตข้อมูลของแพ็คเกจ ด้วยคำสั่ง sudo apt-get update



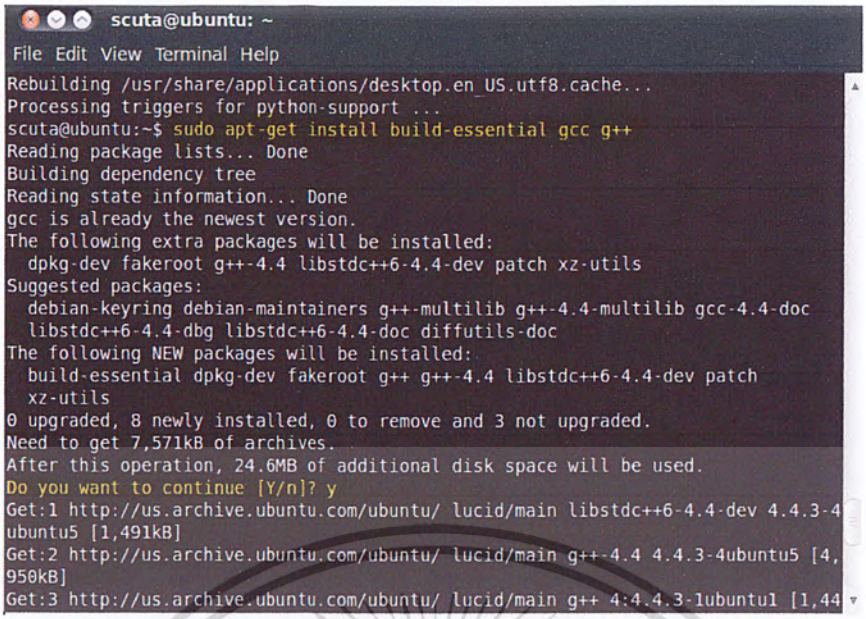
รูป 3.26 apt-get update

2) Apt-get upgrade เป็นการอับเกรดข้อมูลของแพ็คเกจ ด้วยคำสั่ง sudo apt-get upgrade



รูป 3.27 Apt-get update

3) Apt-get install gcc g++ เป็นคอมไพลเลอร์ภาษาซีและซีพลัสพลัส



รูป 3.28 apt-get install build-essential gcc g++

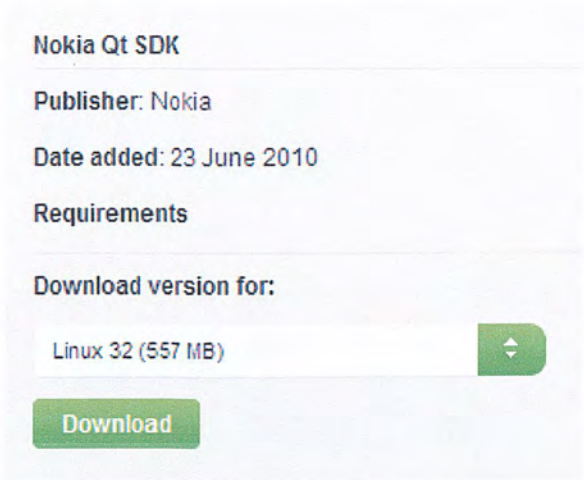
4) Qt SDK

รูป 3.29 Qt SDK

การติดตั้ง Qt SDK

1) เปิด Ubuntu10.04 และเปิด Firefox เข้าเว็บ http://www.forum.nokia.com/info/sw.nokia.com/id/e920da1a-5b18-42df-82c3-907413e525fb/Nokia_Qt_SDK.html

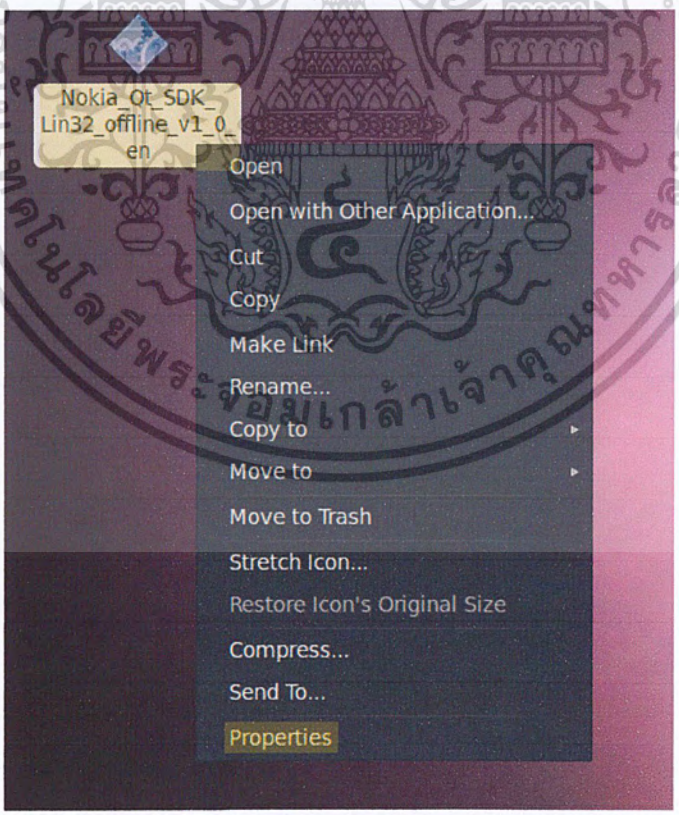
2) โหลด Qt SDK เวอร์ชั่น offine สำหรับ Linux



รูป 3.30 Qt SDK Download

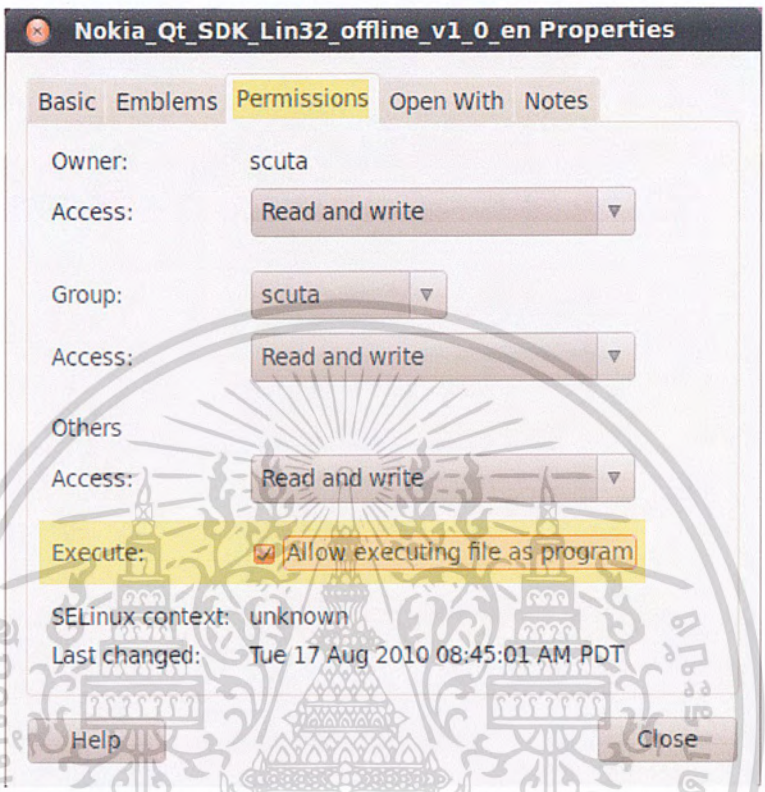
3) เมื่อทำการดาวโหลดจาก Firefox เสร็จ ที่หน้าต่างดาวโหลด ให้คลิกขวา เลือก "open containing folder

4) คลิกขวาที่ไฟล์ Nokia_Qt_SDK_Lin32_offline_v1_0_en เลือก "Properties"



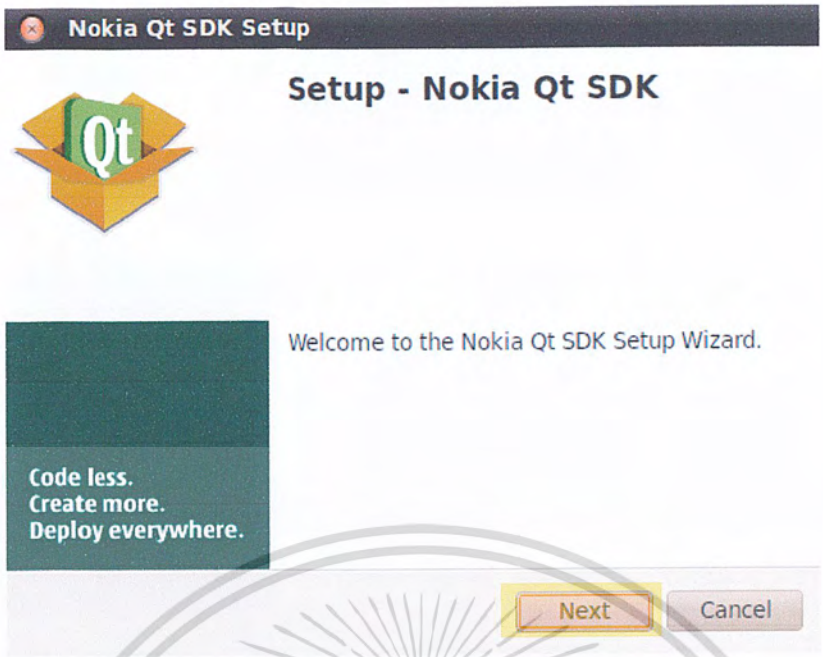
รูป 3.31 properties

5) เลือกแทป Permissions และ ทำเครื่องหมายถูกที่ช่อง "Execute"



รูป 3.32 Permissions

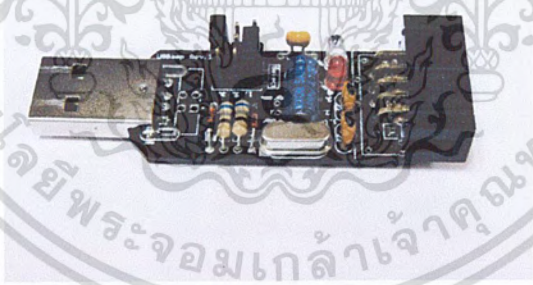
6) คลิก "Close" และดับเบิ้ลคลิกไฟล์เพื่อเริ่มติดตั้ง SDK โดยการกด Next ไปเรื่อยๆ (คล้ายกับการลงโปรแกรมในวินโดว์)



รูป 3.33 Nokia Qt SDK Setup

3.5 กรณีศึกษาวี-ยูเอสบี

3.5.1 USBasp



รูป 3.34 USBasp

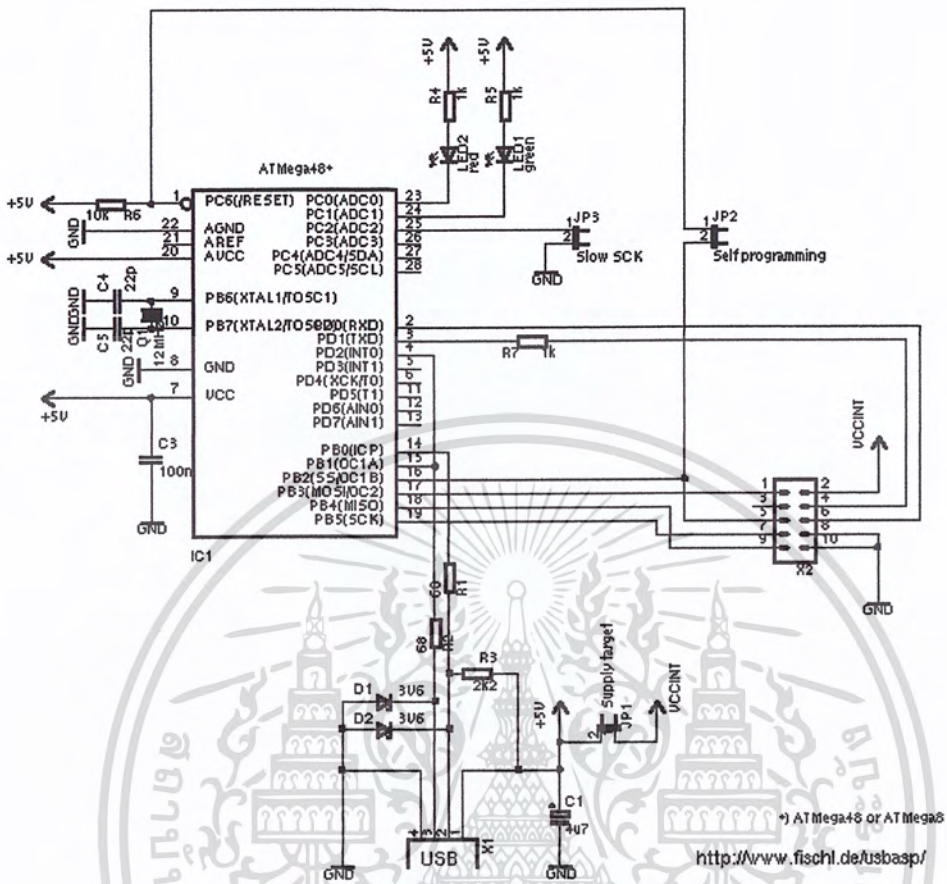
เป็นเครื่องโปรแกรมสำหรับไมโครคอนโทรเลอร์ตระกูล AVR มีความสามารถในการโปรแกรม AVR 8 bit ได้เกือบทุกเบอร์ พัฒนาโดย Thomas Fischl ผู้พัฒนาได้พัฒนาภายใต้เงื่อนไข GPL2(General Public License Source Code) ผู้พัฒนาได้เผยแพร่วงจรและ Source Code ที่เว็บไซต์ http://www.fischl.de/usbasp/ โดยใช้เพียง Atmega8 หรือ Atmega48 ตัวเดียวทำหน้าที่ทั้ง ISP และ USB Controller

3.5.1.1 อุปกรณ์ที่ใช้

ตาราง 3.3 อุปกรณ์ที่ใช้

ลำดับ	รายการ	จำนวน	รายละเอียด	รหัส ES
1.	PCB	1		
2.	Atmega8	1	Surface mount	ATMEGA8A-AU
3.	Resistor 68 Ohm	2	1/4 watt	
4.	Resistor 1k Ohm	5	Surface mount	YR-0805-1/8W-5%-1K
5.	1N5227B	2	3.6v Zener Diode	1N5227B-TAP
6.	Capacitor 22 pF	2	ceramic	T220KCHF051L
7.	Capacitor 4.7 uF	1	electrolytic	EGS475M1HD11RRSAP
8.	Capacitor 0.1 uF	1	Ceramic Multilayer	SR215C104KARAP4-90
9.	LED	2		
10.	Box Header 10 pin	1		SCP10GS31/RH
11.	Crystal 12MHz	1		HC49S-12M-LF
12.	Wafer	1		
13.	Pin Jumper	2		MJ2MH/RH
14.	USB Connect	1		UDRA04MTKNN/RH

3.5.1.2 แบบวงจร USBasp



รูป 3.35 วงจร USBasp

3.5.1.3 ISP Connector

รูป 3.36 ISP pins

ISP Connector เรียกขาการใช้งานดังนี้

ขาหมายเลข 1. MOSI

ขาหมายเลข 2. Vcc

ขาหมายเลข 3. NC ไม่ได้เชื่อมต่อ

ขาหมายเลข 4. TX หรือเชื่อมต่อ GND บน Target Board (*)

ขาหมายเลข 5. NC ไม่ได้เชื่อมต่อ

ขาหมายเลข 6. RX หรือเชื่อมต่อ GND บน Target Board (*)

ขาหมายเลข 7. SCK

ขาหมายเลข 8. GND

ขาหมายเลข 9. MISO

ขาหมายเลข 10. GND

หมายเหตุ (*) ยังไม่ได้ใช้งานในขณะนี้ให้เชื่อมต่อ GND

3.5.1.4 ขาจั๊มเปอร์

1) JP1 Supply From Target Board เมื่อเชื่อมต่อ Jumper นี้ USBAsp จะจ่ายไฟเลี้ยงจาก USB ไปยัง ISP Connector(ขา 2 ของ ISP Connector) เมื่อนำ ISP Connector ไปต่อ กับบอร์ดทดลองแล้วจะทำให้ใช้แหล่งจ่ายไฟเดียวกัน Target Board ในกรณีที่ Target Board หรือ บอร์ดทดลองมีข้อผิดพลาด อาจจะทำให้ USBAsp เสียหายได้ ดังนั้นควรตรวจสอบให้แน่ใจว่า บอร์ดทดลองมีความ น่าเชื่อถือ และสามารถใช้งานได้ตามปกติ และไม่ควรต่อ JP 1 ในกรณีที่ไม่จำเป็น

2) JP2 Update Firmware เมื่อเชื่อมต่อ Jumper นี้จะส่งผลให้ USBASP เข้าสู่โหมด Update Firmware เมื่อมีการจ่ายไฟเลี้ยง หรือเชื่อมต่อ สายUSB ใหม่ ดังนั้นเมื่อเชื่อต่อ Jumper นี้ USBASP จะไม่สามารถทำงานเป็น ISP ได้จนกว่าจะนำ Jumper ออก และจ่ายไฟเลี้ยง หรือเชื่อต่อสาย USB ใหม่อีกครั้งหนึ่ง ส่วนการ Update Firmware นั้น ให้นำเครื่องโปรแกรมอื่น ๆ เสียบเข้าที่ ISP Connector ของ USBAsp แล้วเสียบ Jumper นี้ หาก เครื่องโปรแกรมที่นำมาใช้ไม่มีการจ่ายกระแสออกมาด้วย ต้องต่อสาย USB เพื่อจ่ายไฟเลี้ยงให้ USBAsp

หากมีการเสียบ Jumper นี้แล้วต่อสาย USB จะทำให้ MS-Windows รายงานความผิดพลาดที่เกิดขึ้นบน USB Bus ซึ่งไม่ต้องสนใจ เพราะ USBAsp ได้เข้าสู่โหมด Update Firmware เรียบร้อยแล้ว ควรแน่ใจว่าได้เขียน Fuse Bits ให้กับ USBAsp ถูกต้องซึ่งจะมีค่าดังนี้ USBAsp

rev2 hfuse = 0xC9 , lfuse =0xEF

หากใช้ เครื่องโปรแกรม USBAsp rev2 ให้เรียกใช้งาน Avrdude ด้วย Command นี้เพื่อตั้งค่า FuseBits หากจำเป็นจะต้องเสียบ JP3 ก็ควรเสียบด้วย เพื่อให้ USBAsp rev2 ทำงานในโหมด Slow Clock และพิมพ์ คำสั่งดังนี้ Avrdude –c usbasp –p m8 –U hfuse:w:0xC9 –U lfuse:w:0xEF กด Enter

3) JP3 slow Clock Programming เมื่อมีการใช้ USBASP ที่ทำงานดด้วย Crystal หรือ Oscillator ความเร็วต่ำ มาก ควรจะเสียบ Jumper ที JP3 ด้วย การเสียบ Jumper นี้จะทำให้ USBAsp ลดความเร็วในการโปรแกรมลง

3.5.2 USBmot

รูป 3.37 USBmot

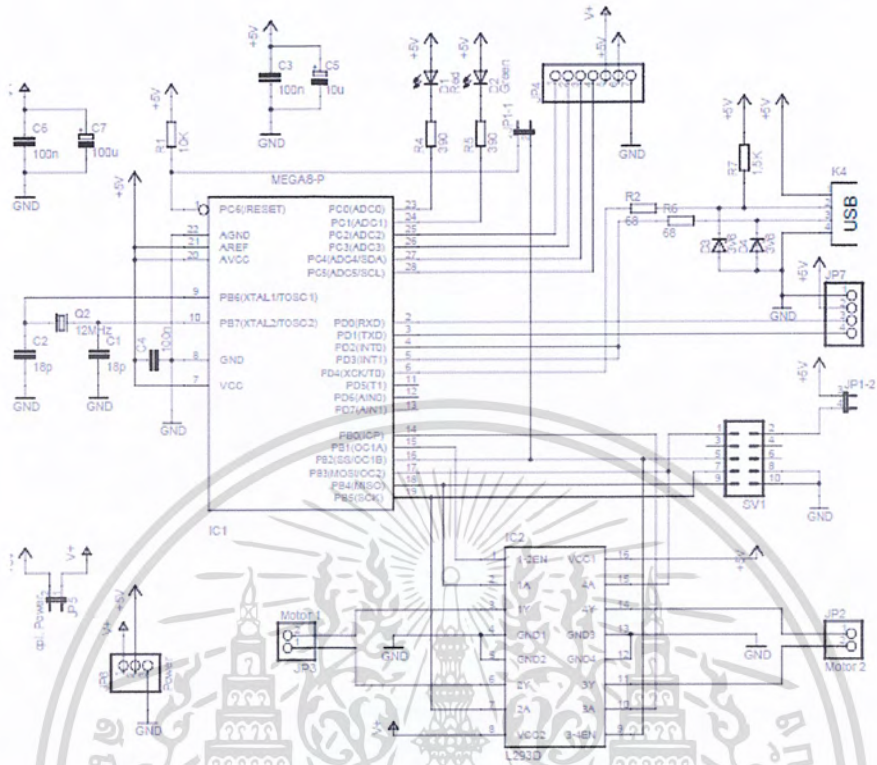
เป็นอุปกรณ์ USB ที่ควบคุมมอเตอร์เล็กๆ(600 mA, 1.2 A peak) จำนวน 2 ตัว โดยใช้ L293D และ Atmega Microcontroller ใช้ Firmware USB Driver ต่อเข้ากับโฮสผ่าน USB และควบคุมความเร็วด้วย PWM

3.5.2.1 อุปกรณ์ที่ใช้

ตาราง 3.4 อุปกรณ์ที่ใช้

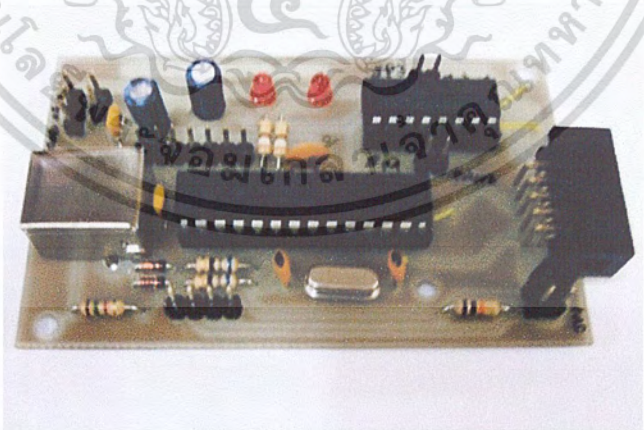
ลำดับ	รายการ	จำนวน	รายละเอียด	รหัส ES
1.	PCB	1		
2.	Atmega8	1	Dip	ATMEGA8A-PU
3.	L293D	1	Motor Driver	L293D
4.	Resistor 68 Ohm	2	1/4 watt	
5.	Resistor 1.5k Ohm	1	1/4 watt	
6.	Resistor 390 Ohm	2	1/4 watt	
7.	Resistor 10k Ohm	1	1/4 watt	
8.	Socket 16 Pin	1	Dual Wipe Contact	ISS16T1/03/RH
9.	Socket 28 Pin	1	Dual Wipe Contact	ISS28T1/03/RH
10.	1N5227B	2	3.6v Zener Diode	1N5227B-TAP
11.	Capacitor 22 pF	2	Ceramic	T220KCHF051L
12.	Capacitor 10 uF	1	Electrolytic	EGS106M1CD11RRSAP
13.	Capacitor 100 uF	1	Electrolytic	EGF107M1CE11TUSAP
14.	Capacitor 0.1 uF	1	Ceramic Multilayer	SR215C104KARAP4-90
15.	LED	2		
16.	Box Header 10 pin	1		SCP10GS31/RH
17.	Crystal 12MHz	1		HC49S-12M-LF
18.	Wafer	1		
19.	Pin Jumper	2		MJ2MH/RH
20.	USB Connect	1	Type B	UDRB04FTWNY/RH
21.	USB Wire	1	สายแบบ A,B	

3.5.2.2 แบบวงจร



รูป 3.38 แบบวงจร USBmot

3.5.2.3 วงจร USBmot ที่ทดลองทำ



รูป 3.39 วงจร USBmot ที่ทดลองทำ

3.5.2.4 การเซตจั๊มเปอร์

1) JP 1-1: Self programming: ต้องเซตบิตถ้าต้องการที่จะทำการโปรแกรมลงในอุปกรณ์ และทำเอาออกถ้าจะใช้งาน

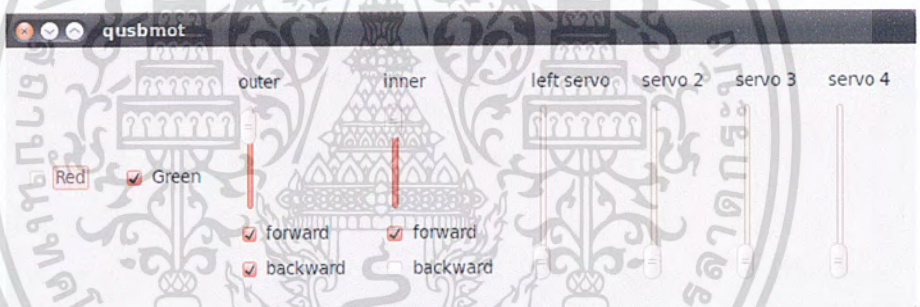
2) JP 1-2: Programmer Power: ใช้ไฟในอุปกรณ์จากแห่งจ่ายของเครื่องโปรแกรม

3) JP 5: coupled power: ต่อมอเตอร์กับแรงดันลอจิก อาจทำให้ใช้ได้กับมอเตอร์ขนาดเล็ก (ควรระวังกระแสไม่ให้ต่ำหรือสูงเกินไป) หรือ ใช้กับแหล่งจ่ายไฟ USB ที่มีกำลังสูง

3.5.2.5 การ Compile

ในการ Compile ทั้งที่เป็น Firmware และ Host Software จำเป็น AVR build chain (เช่น avr-gcc และ avrdude) ต้องมี Compiler ที่สามารถ Compile C++ ได้ และ ชุดพัฒนาของ Qt (Qt SDK) รวมถึง libusb

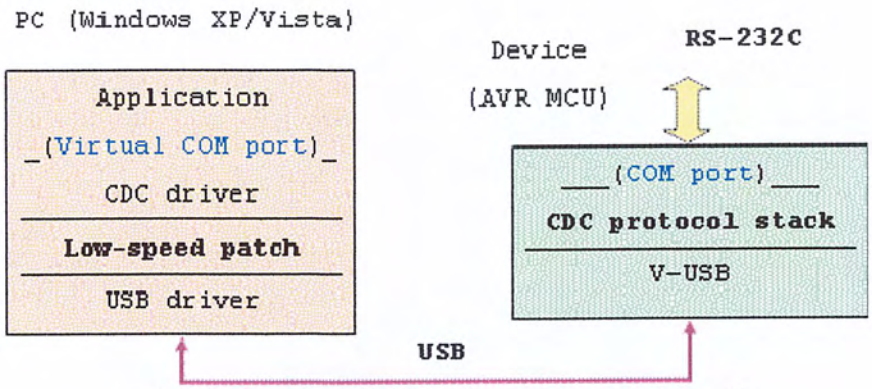
3.5.2.5 Host Software



รูป 3.40 USBmot Host Software

3.5.3 CDC-232

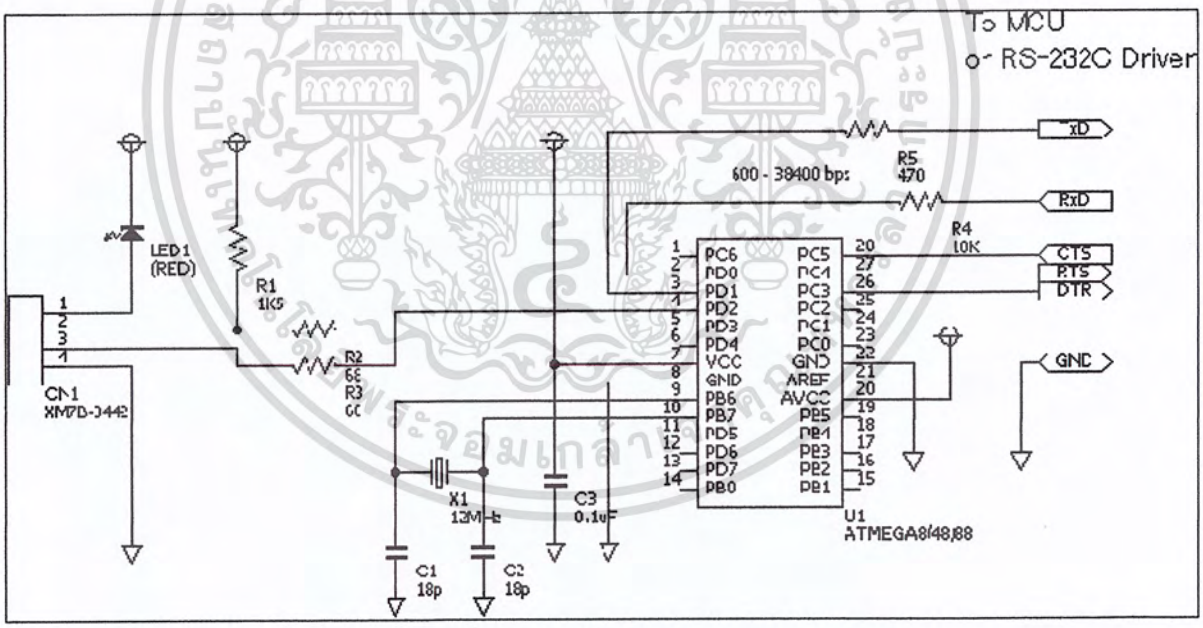
CDC-232 เป็นวงจรที่ใช้สร้างคอมพิวเตอร์พอร์ต (Virtual COM port) เสมือนสำหรับเครื่องคอมพิวเตอร์ที่ไม่มีพอร์ต RS-232 ผ่านไมโครคอนโทรลเลอร์ AVR ให้สื่อสารแบบ RS-232 ผ่านยูเอสบีได้หลังจากที่ลงไดร์เวอร์



รูป 3.41 คอมพิวเตอร์พอร์ตเสมือนด้วยซอฟแวร์ยูเอสบี

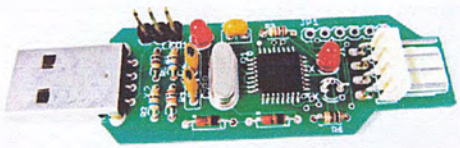
3.5.3.1 วงจร

วงจรนี้เป็นวงจรสำหรับไมโครคอนโทรลเลอร์ ATmega8/48/88/168 โดยที่ LED สีแดง ทำหน้าที่จำกัดแรงดันไฟยูเอสบีจาก 5 โวลต์ เหลือ 3.3 โวลต์ เพื่อให้กับไมโครคอนโทรลเลอร์ กระแสมี 10 มิลลิแอมป์ ซึ่งไม่พอในการขับเคลื่อนวงจรอื่น เวลาวงจรนี้เข้ากับไมโครคอนโทรลเลอร์ตัวอื่น ให้ต่อกราว TxD และ RxD



รูป 3.42 วงจร CDC-232 สำหรับ ATmega8/48/88-20

3.5.3.2 วงจร CDC -232 ที่ทดลองทำ



รูป 3.43 วงจร CDC -232 ที่ทดลองทำ

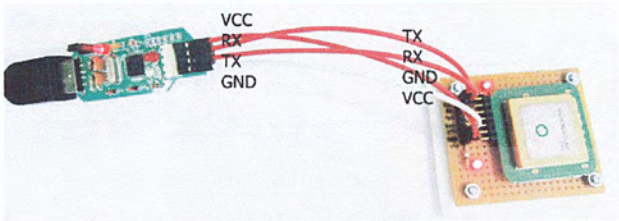
3.5.3.3 อุปกรณ์ที่ใช้

ตาราง 3.5 อุปกรณ์ที่ใช้

ลำดับ	รายการ	จำนวน	รายละเอียด	รหัส ES
1.	PCB	1		
2.	Atmega8	1	QFP	ATMEGA8A-AU
3.	Resistor 68 Ohm	2	1/8 watt	
4.	Resistor 1.5k Ohm	1	1/8 watt	
5.	Resistor 470 Ohm	1	1/8 watt	
6.	Resistor 10k Ohm	1	1/8 watt	
7.	1N5227B	2	3.6v Zener Diode	1N5227B-TAP
8.	Capacitor 22 pF	2	Ceramic	T220KCHF051L
9.	Capacitor 0.1 uF	1	Ceramic Multilayer	SR215C104KARAP4-90
10.	Crystal 12MHz	1		HC49S-12M-LF
11.	Red LED	3		
12.	Wafer	1		
13.	Pin Jumper	2		MJ2MH/RH
14.	USB Connect	1	Type B	UDRB04FTWNY/RH
15.	USB Wire	1	สายแบบ A,B	

3.5.3.4 การใช้งาน

1) ต่อวงจรเข้า CDC-232 เข้ากับดีไวซ์ที่จะทำการรับส่งค่า เช่น GPS โดยต่อสลับ Tx Rx ของ CDC-232 กับดีไวซ์ด้วย

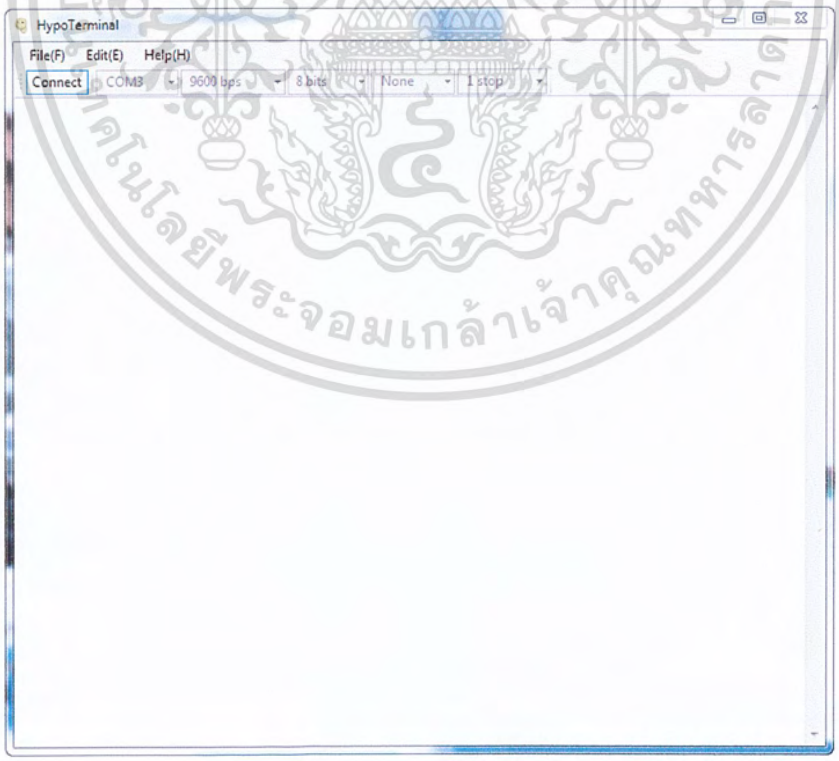


รูป 3.44 วงจร CDC -232 ที่ต่อกับ GPS

2) เสียบ CDC -232 เข้ากับพอร์ตยูเอสบี

3) ติดตั้งไดรฟ์เวอร์ของ CDC-232 ในวินโดว์

4) ใช้โปรแกรม Terminal ต่างๆ เช่น Hyper Terminal, Hypo Terminal, PuTTY เพื่อใช้อ่านค่าเวลารับส่งข้อมูล โดยเลือก COM Port ของ CDC ให้ถูกต้อง เลือก baudrate ที่ 9600 bps



รูป 3.45 Hypo Terminal

3.5.4 CDC-IO

เป็นวงจรที่ทำให้ไมโครคอนโทรลเลอร์ AVR ธรรมดา สามารถติดต่อ IO ได้แบบ RS-232 ผ่านพอร์ตยูเอสบีแบบ 1.1 ซึ่งสามารถนำไปประยุกต์ใช้งานได้หลากหลาย

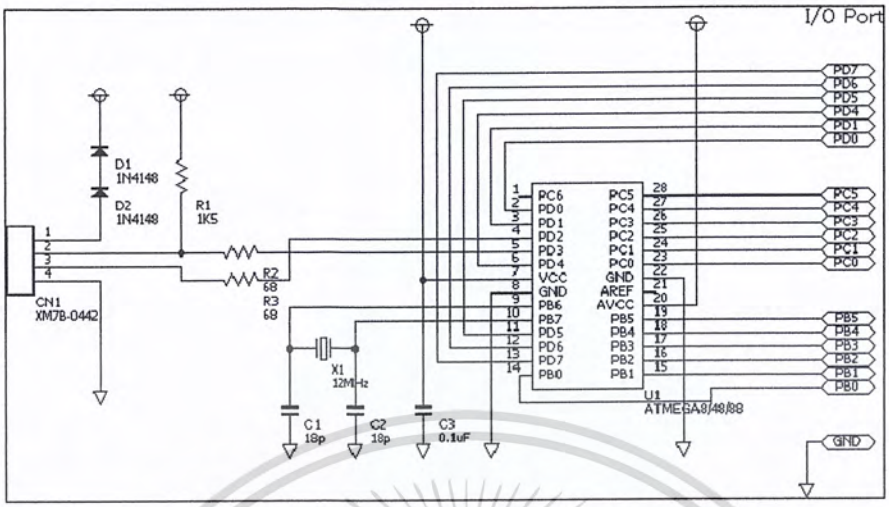
ตาราง 3.6 CDC-IO Instruction Set

Function	Command	Format	Response
Who	@	@	"cdc-io", CR-LF
Get	?	address ?	data, CR-LF
Set	=	data address =	CR-LF
AND & Set	&	data address &	CR-LF
OR & Set	|	data address |	CR-LF
EX-OR & Set	^	data address ^	CR-LF
Set Double	$	data2 data1 address $	CR-LF

3.5.4.1 การใช้งาน

1) แอดเดรสและข้อมูลก่อหน้าสามารถนำกลับมาใช้ได้ เพียงใช้คำสั่ง char เพื่อวนซ้ำ
2) ถ้าต้องการเขียนลง EEPROM ให้ใช้คำสั่ง “Set Double”
3) หลีกเลี่ยงการเปลี่ยนบิตที่ใช้ร่วมกับสัญญาณยูเอสบี ในการเปลี่ยนทิศทางของพอร์ตให้ใช้ ‘&’, ’|’ หรือ ‘^’ ถ้าพอร์ตใช้เป็นเอาต์พุตให้ใช้ PIN* เพื่อทอกเกิ้ลบิต

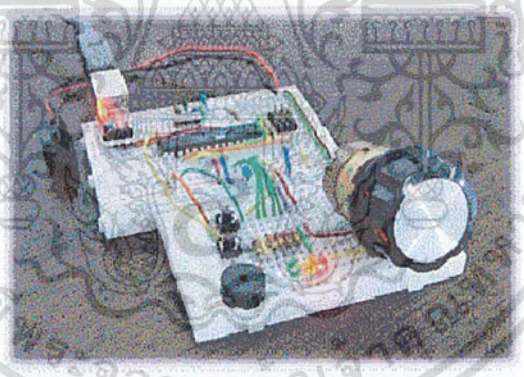
3.5.4.2 วงจร



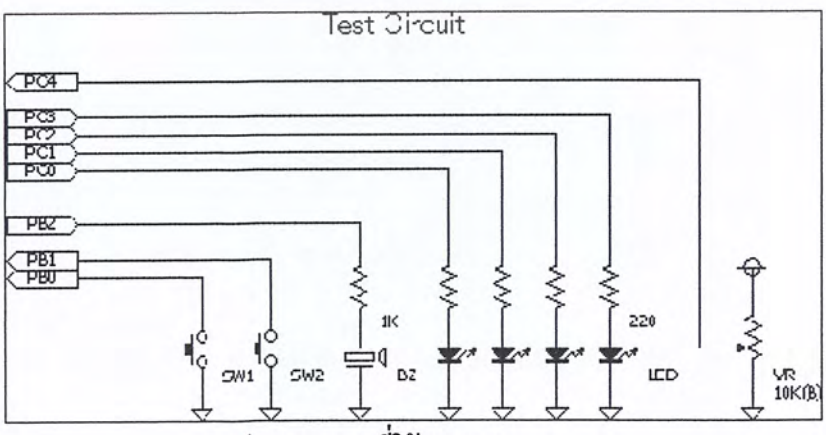
รูป 3.46 CDC-IO สำหรับ ATmega8/48/88-20

3.5.4.3 ตัวอย่างการทดลอง

เป็นวงจรสั่งเพิ่ม-ลดเสียงของบัสเซอร์และจังหวะการกระพริบของ LED



รูป 3.47 วงจรตัวอย่าง



รูป 3.48 วงจรที่ใช้ทดสอบ

3.5.4.4 อุปกรณ์ที่ใช้

ตาราง 3.7 อุปกรณ์ที่ใช้

ลำดับ	รายการ	จำนวน	รายละเอียด	รหัส ES
1.	PCB	1		
2.	Atmega8	1	DIP	ATMEGA8A-PU
3.	Resistor 68 Ohm	2	1/8 watt	
4.	Resistor 1.5k Ohm	1	1/8 watt	
5.	1N4148	2	DIODE:100V,200mA	1N4148TA
6.	Capacitor 22 pF	2	Ceramic	T220KCHF051L
7.	Capacitor 0.1 uF	1	Ceramic Multilayer	SR215C104KARAP4-90
8.	Crystal 12MHz	1		HC49S-12M-LF
9.	Box Header 10 pin	1		SCP10GS31/RH
10.	Socket 28 Pin	1	Dual Wipe Contact	ISS28T1/03/RH
11.	Wafer	1		
12.	USB Connect	1	Type B	UDRB04FTWNY/RH

บทที่ 4

การทดลองและผลการทดลอง

4.1 การทดสอบบอร์ด

การทดสอบจะใช้บอร์ด Mini 2440 ทดสอบได้ดังนี้

ตาราง 4.1 การทดสอบจะใช้บอร์ด Mini 2440

โมดูล	ผลการทดสอบ
CPU : Samsung S3C2440	ซีพียูสามารถทำงานได้ โดยใช้ Debian
Serial	สามารถรับอินพุทและส่งเอาต์พุทได้อย่างถูกต้อง
USB	สามารถทำการเชื่อมต่อกับยูเอสบีไวซ์ได้
SD Card	สามารถบูตไฟล์ซิสเต็มจาก SC Card ได้
Ethernet	ยังไม่สามารถใช้งานได้ เนื่องจากยังไม่มีแอปพลิเคชั่นใช้ในการทดสอบ
Audio	ยังไม่สามารถใช้งานได้ เนื่องจากยังไม่มีแอปพลิเคชั่นใช้ในการทดสอบ
PWM	สามารถส่งสัญญาณ PWM ได้
Interrupt	สามารถใช้งาน Interrupt ได้

4.2 การทดลองโมดูลอุปกรณ์แบบยูเอสบี

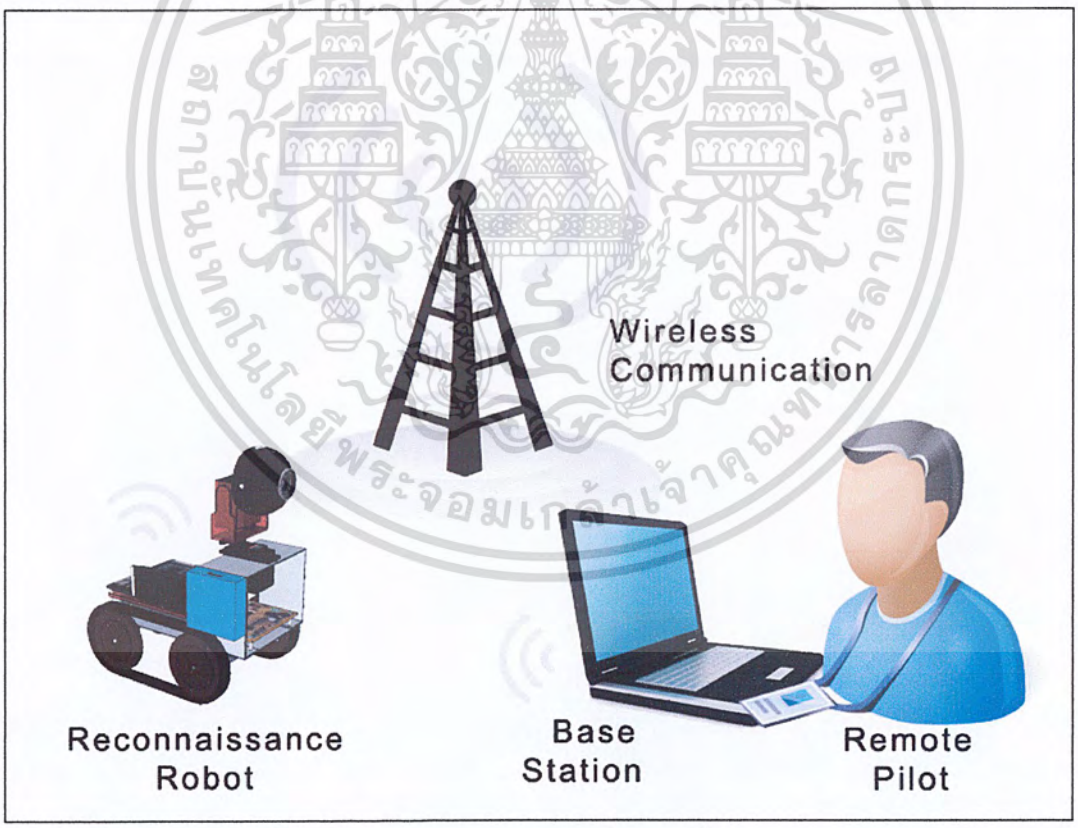
หลังจากที่ได้พัฒนาตัวฮาร์ดแวร์ของยูเอสบีดีไวซ์ แล้วได้ทำการทดสอบดังนี้

ตาราง 4.2 การทดลองโมดูลอุปกรณ์แบบยูเอสบี

โมดูล	ผลการทดสอบ
USBasp	สามารถโปรแกรมไมโครคอนโทรลเลอร์ Atmega 8 ได้ปกติ และสามารถอับเดตโปรแกกรมให้ตัวอุปกรณ์เองได้
USBmot	สามารถทำงานจากการสั่งงานบนโฮสโปรแกรมได้ปกติ
USB-CDC	สามารถติดต่อกับคอมพิวเตอร์เพื่อสร้างพอร์ตเสมือนได้ และสามารถส่งข้อมูลแบบ RS-232 ได้
USB-IO	สามารถติดต่อกับคอมพิวเตอร์เพื่อส่งข้อมูล IO ได้

4.3 การทดสอบแพลตฟอร์ม

โดยการทดสอบมีภาพรวมของระบบมีส่วนที่เกี่ยวข้องต่างๆดังรูป

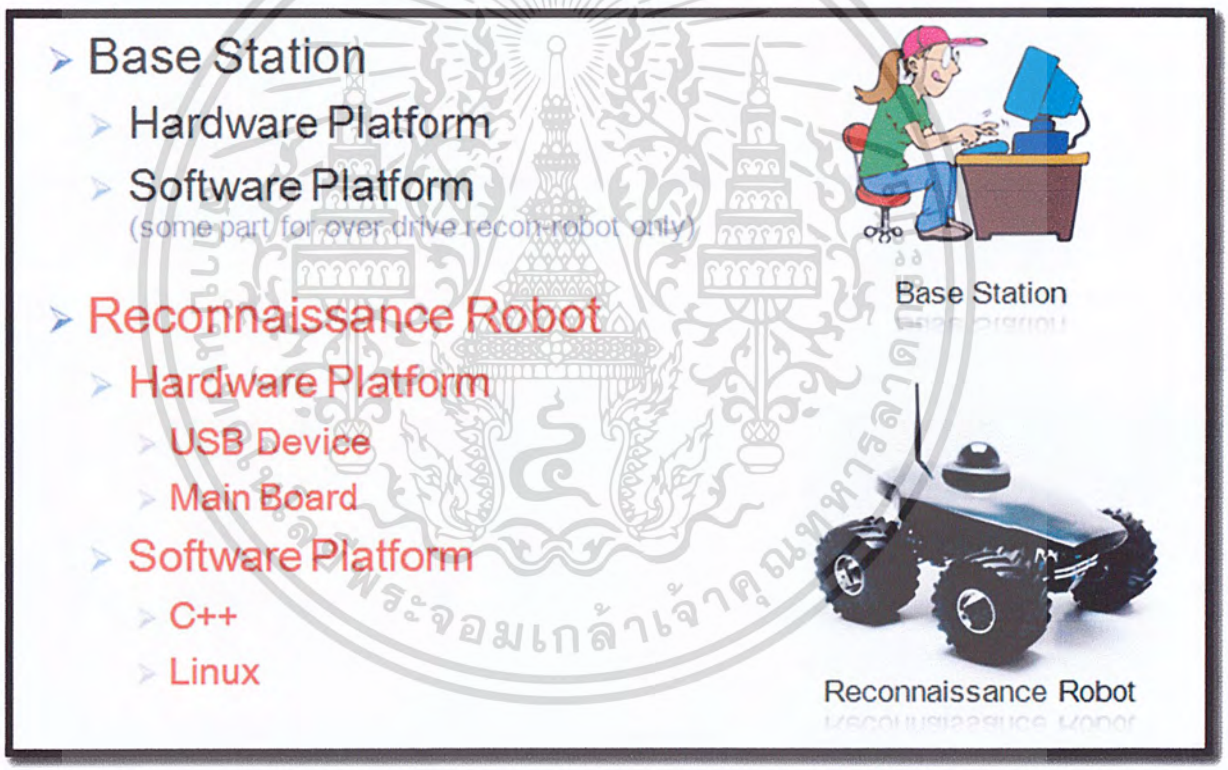


รูป 4.1 ภาพรวมของระบบรีคอนโรบอทที่ใช้ทดสอบ

โดยแบ่งออกเป็น 4 ส่วนที่เกี่ยวข้องใหญ่ๆ ได้แก่

1) **ผู้ปฏิบัติการ (Remote Pilot)** ผู้ดำเนินการในการปฏิบัติภาระกิจ
2) **ส่วนติดต่อผู้ใช้ (Base Station)** อุปกรณ์ที่ใช้ในการติดต่อกันระหว่างรีโมทไพลอทกับรีคอนไนส์เส้นซ์โรบอท
3) **การติดต่อสื่อสาร (Communication)** ระบบย่อยที่เป็นตัวกลางที่ใช้ในการติดต่อสื่อสารระหว่างเบสเสตชั่นกับรีคอนไนส์เส้นโรบอท ซึ่งในที่นี้เลือกใช้ไวไฟ (Wifi)
4) **หุ่นยนต์สอดแนม (Reconnaissance Robot)** หุ่นยนต์ที่ใช้ในการทดสอบแพลตฟอร์ม โดยหน้าที่หลักคือรับคำสั่งไปถ่ายทอดภาพจากพื้นที่เป้าหมาย

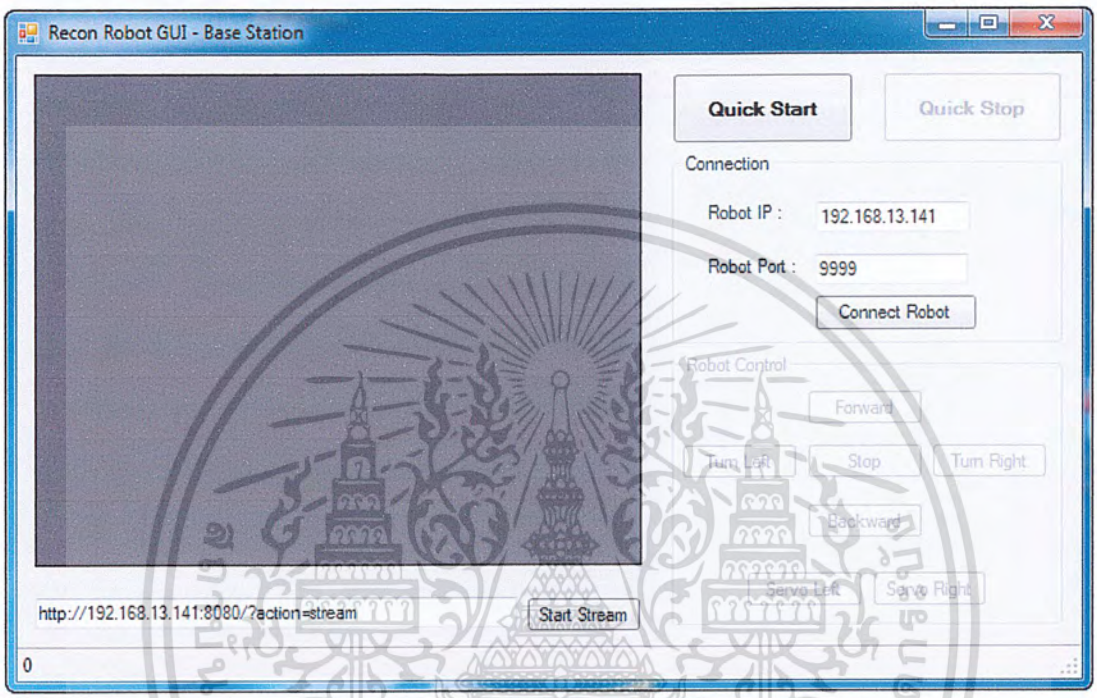
สำหรับขอบเขตในการทดลองนั้นได้ทำการการแบ่งเป็น 2 ส่วนใหญ่ๆดังรูป



รูป 4.2 ส่วนที่สนใจได้แก่ส่วนของรีคอนไนส์เส้นซ์โรบอทและเบสสเตชั่นบางส่วน

4.3.1 ส่วนติดต่อผู้ใช้ (Base Station)

โดยในส่วนของส่วนติดต่อผู้ใช้ นั้นได้พัฒนาเฉพาะส่วนของซอฟแวร์บางส่วนเท่านั้นเพื่อใช้แสดงการทำงานโดยใช้โปรแกรมวิชวลซีชาป (Visual C#) ในการเขียนโปรแกรมเพื่อใช้ติดต่อกับกล้องบนตัวหุ่น และบังคับหุ่นยนต์เคลื่อนที่ไปตามที่ต้องการ



รูป 4.3 ส่วนติดต่อผู้ใช้งานที่ Base Station

4.3.2 หุ่นยนต์สอดแนม (Reconnaissance Robot)

ได้มีการพัฒนาทั้งในส่วนของฮาดแวร์และซอฟแวร์ โดยฮาดแวร์ได้นำยูเอสบีดีไวส์ (USB Device) มาใช้ด้วยกัน 4 อย่างได้แก่ กล้องเว็บแคม, บอร์ดไดรว์, เซอร์โวคอนโทรล, ยูเอสบีไวไฟ ส่วนของซอฟแวร์นั้นได้พัฒนาโปรแกรมใน 2 ชั้น คือชั้นของการดำเนินงาน (Operation Layer) ได้พัฒนาโปรแกรมของแต่ละยูเอสบีดีไวว์เพื่อใช้ติดต่อและสั่งงานยูเอสบีดีไวส์นั้นๆ ชั้นของยุทธวิธี (Tactical Layer) ได้พัฒนาให้รับคำสั่งจากยูเอสบีไวไฟโดยชั้นของการดำเนินงาน และแปลงคำสั่งแล้วสั่งงานต่อไปยังบอร์ดไดรหรือ กล้อง เพื่อสั่งงานต่อไป โดยการปฏิบัติงานของหุ่นยนต์สอดแนมข้อมูลนั้นแบ่งออกเป็น 4 ขั้นตอนใหญ่ๆดังนี้

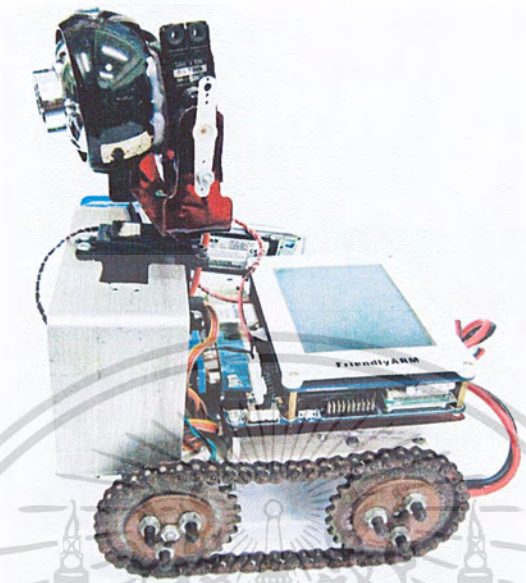
1) วางแผน (Planning)

2) สร้างและประกอบหุ่นยนต์โจรกรรม (Building)

3) ติดตั้ง ปรับแต่ง และทดสอบอุปกรณ์ต่างๆ (Setting)

4) ปฏิบัติการ (Excuting)

ซึ่งในการทดลองนี้เราจะทดลองในขั้นตอนของการ ปฏิบัติการ เท่านั้นเพื่อทดสอบแพลตฟอร์ม



รูป 4.4 หุ่นยนต์ตัวอย่าง Reconnaissance Robot

จากรูปเป็นหุ่นยนต์สอดแนม (Reconnaissance Robot) อย่างง่าย ซึ่งพัฒนาขึ้นมาเพื่อใช้ในการทดสอบแพลตฟอร์มของโครงงาน ว่าสามารถนำไปสร้างเป็นหุ่นยนต์ได้จริง มีความสะดวกจริง และมีประสิทธิภาพจริงหรือไม่ โดยหุ่นยนต์ตัวอย่างนี้มีอุปกรณีดังต่อไปนี้

แพลตฟอร์มเอมเบดเดดลินุกซ์โรบอท	1	ตัว
Mini2440	1	อัน
ยูเอสบีฮับ	1	ตัว
เว็บแคม	1	ตัว
Servo Motor	2	ตัว
DC Motor	2	ตัว
USB wireless Adapter	1	ตัว
GPS Module	1	อัน
Digital Compass Module	1	อัน
Wireless Router	1	อัน
Battery Li-Po	2	ก้อน
Host Computer	1	เครื่อง

4.4 การทดสอบการทำงานหุ่นยนต์ตัวอย่าง

4.4.1 การทดสอบการทำงานของโมดูลการเคลื่อนที่

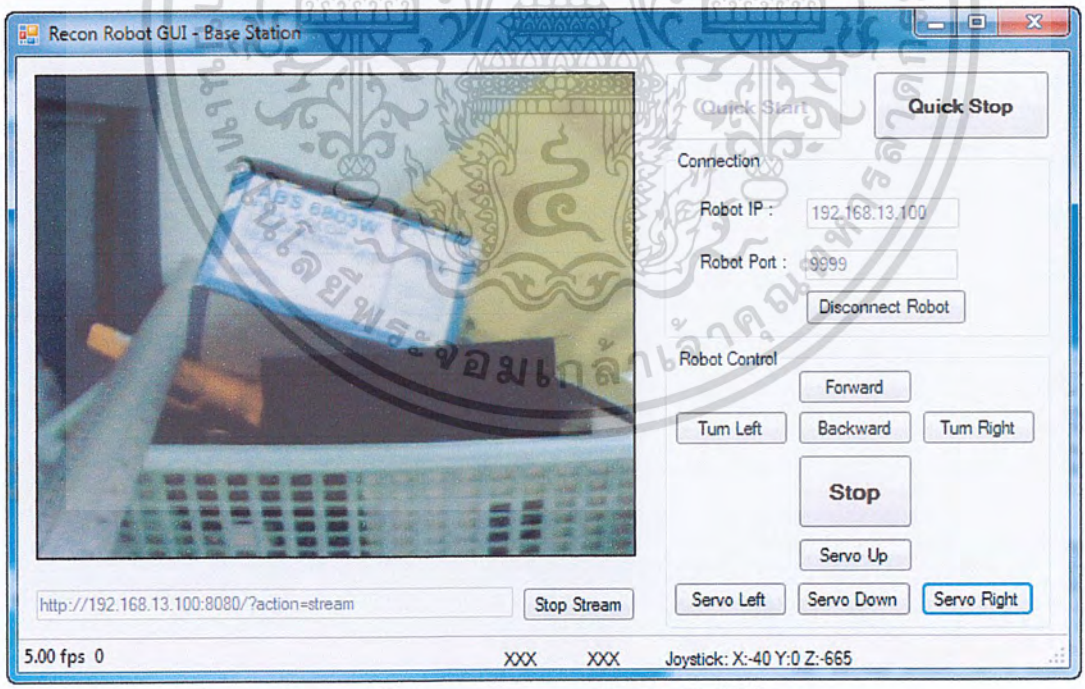
สามารถนำแพลตฟอร์มไปใช้งานได้ โดยหุ่นยนต์สามารถเคลื่อนที่ตามคำสั่งที่ได้รับภาระกิจ หรือรับคำสั่งตรงจาก Base Station ได้

4.4.2 การทดสอบการทำงานการสื่อสารข้อมูล

สามารถนำแพลตฟอร์มไปใช้งานได้ โดยหุ่นยนต์สามารถสื่อสารระหว่าง Base Station แบบ UDP ผ่านระบบ Internet ด้วย Wireless Router เพื่อส่งข้อมูลและภาพพื้นที่ที่ถูกสำรวจกลับไปได้ แต่มีข้อเสียตรงที่ถ้าใช้นานๆ จะเกิดอาการค้าง เนื่องจากปัญหาระบบการสื่อสาร และแบนวิธของอินเทอร์เน็ตก็มีผลต่อการสื่อสารเช่นกัน

4.4.3 การทดสอบโมดูลกล้องเว็บแคม

สามารถนำแพลตฟอร์มไปใช้งานได้ โดยหุ่นยนต์สามารถส่งภาพกลับมายัง Base Station ได้ ที่ 12-15 เฟรม/วินาที ซึ่งขึ้นอยู่กลับความเร็วของอินเทอร์เน็ต ถ้าอินเตอร์เน็ตช้า จะทำให้ภาพค้าง บังคับหุ่นยนต์ได้ยาก



รูป 4.5 หน้าจอของโปรแกรมส่วนของผู้ติดต่อกับผู้ใช้ในขณะทดสอบ

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทวิจารณ์ และสรุปผล

โครงงานนี้มีความคืบหน้าในระดับที่น่าพอใจ โดยเป็นไปตามจุดประสงค์และเป้าหมายของโครงการที่ได้วางแผนไว้ในต้นของโครงการ โดยผลที่ได้คือ ฮาร์ดแวร์แพลตฟอร์มสามารถใช้งานได้ตัวลินุกซ์และซอฟแวร์สามารถทำงานร่วมกันได้อย่างดี แต่ยังมีบางโมดูลที่ยังไม่สามารถทำการทดสอบได้เนื่องจากบางโมดูลไม่มีดีไวซ์ไดร์เวอร์ในการทดสอบและฮาร์ดแวร์ยังมีข้อผิดพลาดอยู่ เฟรมเวิร์คและไลบรารี่ สามารถใช้งานได้ง่ายและสุดท้ายสามารถนำไปใช้สร้างหุ่นยนต์ได้จริง

5.2 ปัญหาอุปสรรคและแนวทางแก้ไข

ปัญหาและอุปสรรคที่พบในการทำโครงงานได้แก่

1) เนื่องจากผู้จัดทำโครงงานไม่มีความคุ้นเคยกับ Linux และ OOP ทำให้ใช้เวลาในการทำความเข้าใจค่อนข้างนาน

2) การสร้างแพลตฟอร์มขึ้นมานั้นใช้เวลามาก เนื่องจากต้องหาส่วนประกอบต่างๆที่เหมาะสม

แนวทางการแก้ไข

1) ศึกษาทฤษฎีที่เกี่ยวข้องให้มากกว่านี้ และหนักกว่านี้เพื่อลดเวลาที่สูญเสียไป

2) ศึกษาความรู้ต่างๆให้มากขึ้นเพื่อเปิดมุมมองใหม่ๆ

5.3 แนวทางการพัฒนาต่อ

1) ออกแบบและพัฒนายูเอ็มแอลของแพลตฟอร์ม

2) ออกแบบและพัฒนาดีไวซ์ไดรเวอร์เพิ่มเติม

3) ออกแบบและพัฒนาโมดูลอุปกรณ์ยูเอสบีเพิ่มเติมให้หลากหลายมากขึ้น

4) ออกแบบและพัฒนาไลบรารี่ให้สนับสุนการทำงานของแพลตฟอร์ม

5) นำแพลตฟอร์มไปสร้างหุ่นยนต์ที่มีประสิทธิภาพ

บรรณานุกรม

วรเทพ ไพบูลย์รัตนากร. **สัมผัสโลก USB ด้วย Ezy USB Module.** พิมพ์ครั้งที่ 1. กรุงเทพฯ : บริษัทแอส ทรอน ลอจิก รีเสิร์ชแอนด์ดิวิลอปเมนต์

Bruce Powel Douglass, Ph.D. 2007. **Real-Time UML Workshop for Embedded Systems.** Burlington,MA: Newnes

Bruce Powel Douglass, Ph.D. 2004. **Real time UML:advance in the UML for real-time systems.** USA: Pearson Education, Inc

อภิเนตร อูนากูล. 2546. **กระบวนการและวิธีการพัฒนาเว็บแอพพลิเคชันโดยใช้ UML.** กรุงเทพฯ: แผนกตำรา คณะวิศวกรรมศาสตร์ สจล.

อรรถสิทธิ์ อารยางกูร และพิเชฐ มาภู่. 2550. "แพลตฟอร์มลินุกซ์ฝังตัวสำหรับหุ่นยนต์." ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

ARM Ltd and ARM German GmbH. **USB-Based Microcontrollers.** [Online]. Available : http://www.keil.com/usb/chips.asp

Objective Development Software GmbH. **V-USB A Firmware-Only USB Driver for Atmel AVR Microcontrollers.** [Online]. Available : http://www.obdev.at/products/vusb

Andreas Goeizer. **Usbmot.** [Online]. Available : http://andreas.goelzer.de/usbmot

Thomas Fischl. **USBasp – Programmer for Atmel AVR Controllers.** [Online]. Available : http://www.fischl.de/usbasp/

Admin. **มีอะไรใหม่ใน USB3.0.** [Online]. Available : http://www.allaboutnotebook.com/2010/what-new-in-usb3/

ชิต เหล่าวัฒนา. **สมองคนสู่สมองกลอัจฉริยะ.** [Online]. Available :
http://fibo.kmutt.ac.th/fiboweb07/thai/index.php?option=com_content&task=view&id=623&Itemid=128

Wikipedia. **Unmanned aerial vehicle.** [Online]. Available :
http://en.wikipedia.org/wiki/Unmanned_aerial_vehicle

Wikipedia. **Unmanned ground vehicle.** [Online]. Available :
http://en.wikipedia.org/wiki/Unmanned_ground_vehicle#Teleoperated_UGV

Wikipedia. **Unmanned underwater vehicle.** [Online]. Available :
http://en.wikipedia.org/wiki/Unmanned_underwater_vehicle

Linux Devices. **Snapshot of the embedded Linux market -- April, 2007.** [Online]. Available :
http://www.linuxfordevices.com/c/a/Linux-For-Devices-Articles/Snapshot-of-the-embedded-Linux-market-April-2007/

Project4fun.com. **Backup/Restore ด้วย Supervivi.** [Online]. Available :
http://www.project4fun.com/node/13

Project4fun.com. **Cross Toolchain สำหรับ mini2440.** [Online]. Available :
http://www.project4fun.com/node/10

Project4fun.com. **การสร้างและติดตั้ง uboot สำหรับ mini2440.** [Online]. Available :
http://www.project4fun.com/node/12

Project4fun.com. **การใช้ usb wireless adapter กับ mini2440.** [Online]. Available :
http://www.project4fun.com/node/36

Project4fun.com. **การสร้าง Kernel สำหรับ mini2440.** [Online]. Available :
http://www.project4fun.com/node/11

Project4fun.com.**การสร้าง Debian File System สำหรับ mini2440.** [Online]. Available :

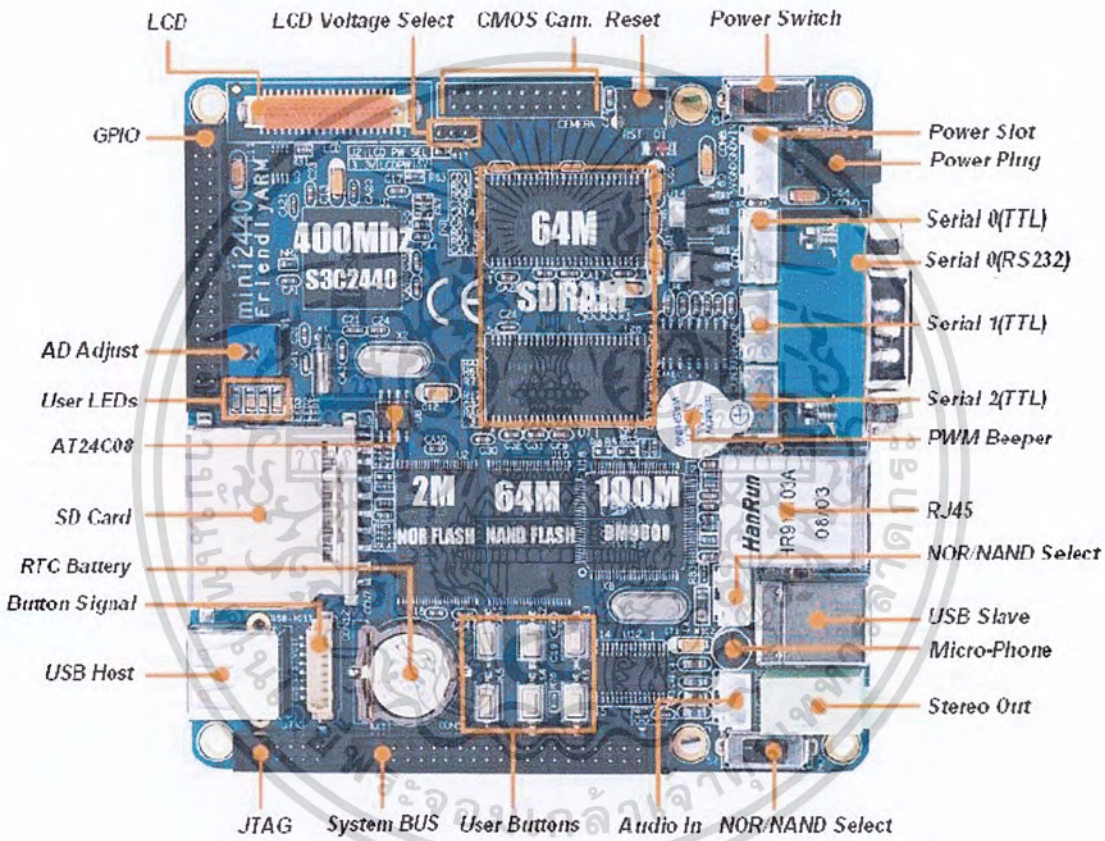
http://www.project4fun.com/node/18

ภาคผนวก ก.

การพัฒนา ซอฟแวร์ ของ Mini 2440

ก.1 Backup/Restore ด้วย Supervivi

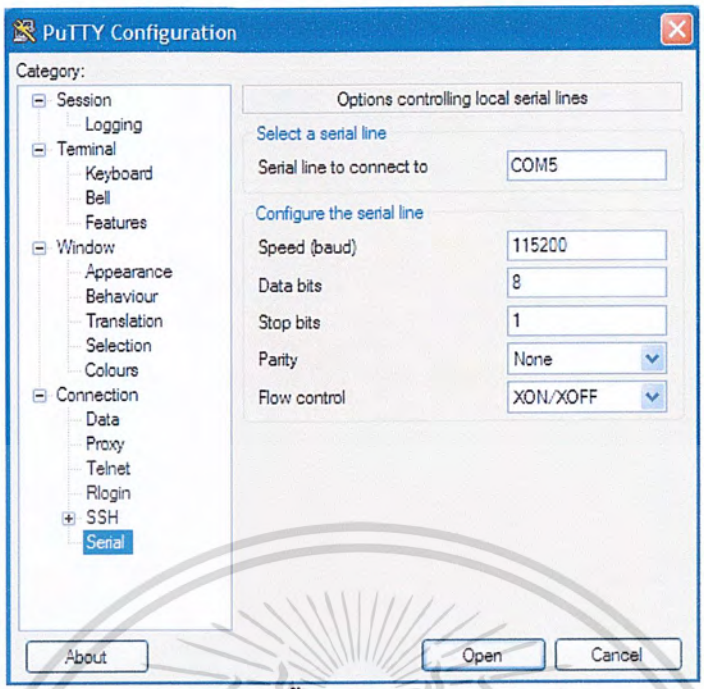
Supervivi เป็น Bootloader ที่มาพร้อมกับบอร์ด mini2440 ซึ่งเราสามารถใช้ในการ backup และ restore NAND flash ดังนี้



รูป ก.1Friendly Arm Mini2440 Board

ก.1.1 ติดตั้ง usb driver ที่ใช้ในการ upload/download ระหว่าง supervivi กับ pc host

1) ปิดสวิตซ์ไฟบอร์ด
2) เลื่อนสวิตซ์ NOR/NAND ไปที่ NOR
3) ต่อสาย Serial0 (RS232) ไปยังคอมพิวเตอร์
4) เปิดโปรแกรม putty แล้วต่อกับ R232 ด้วย Buadrate 115200, n, 8, 1



รูป ก.2 การตั้งค่าโปรแกรม PuTTy

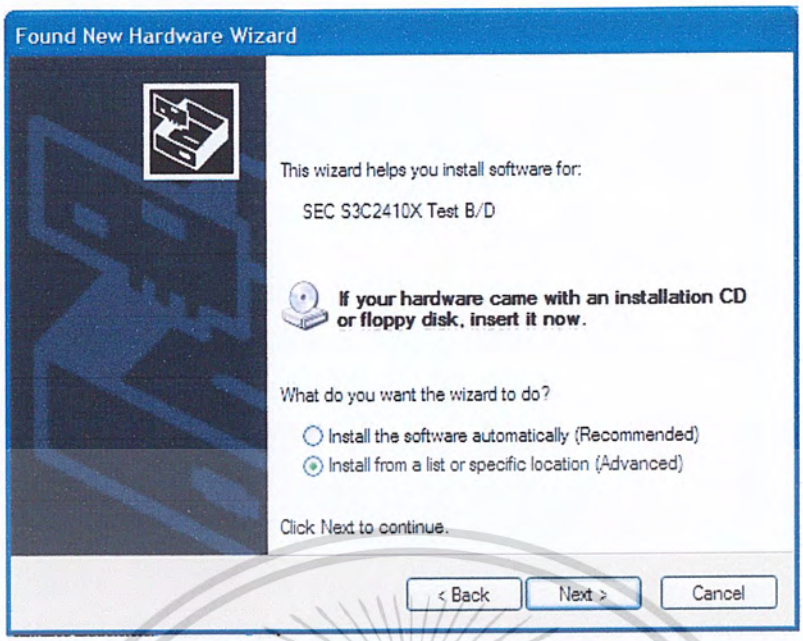
5) ต่อสาย USB Slave กับคอมพิวเตอร์

6) Download และ unzip ไฟล์ dwn และ usb driver ได้จาก http://project4fun.com/sites/default/files/dwn_driver.zip

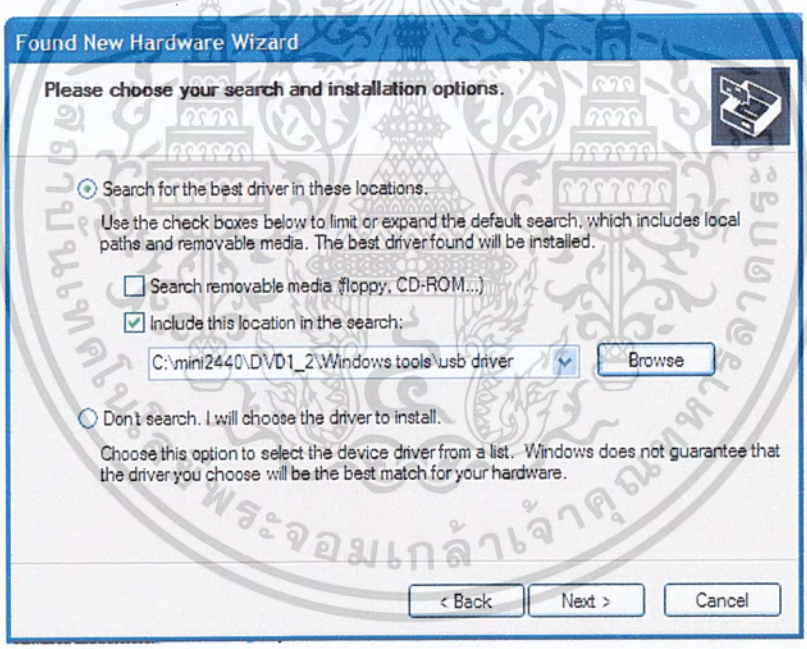
7) เปิดสวิตซ์ไฟบอร์ด

8) คอมพิวเตอร์จะถามหา Driver ให้ติดตั้ง driver จากไฟล์ usb driver ที่ download

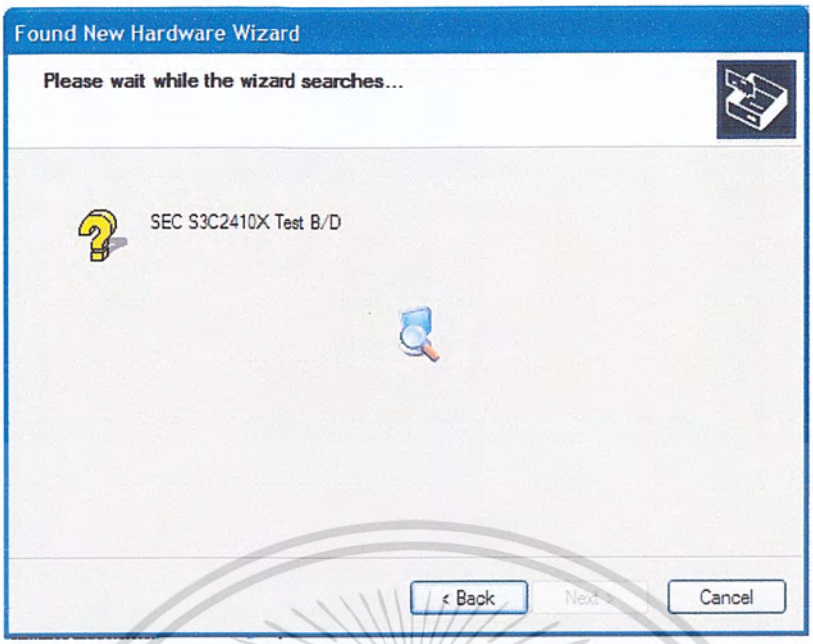
รูป ก.3 การลง USB Driver: Yes, This time Only



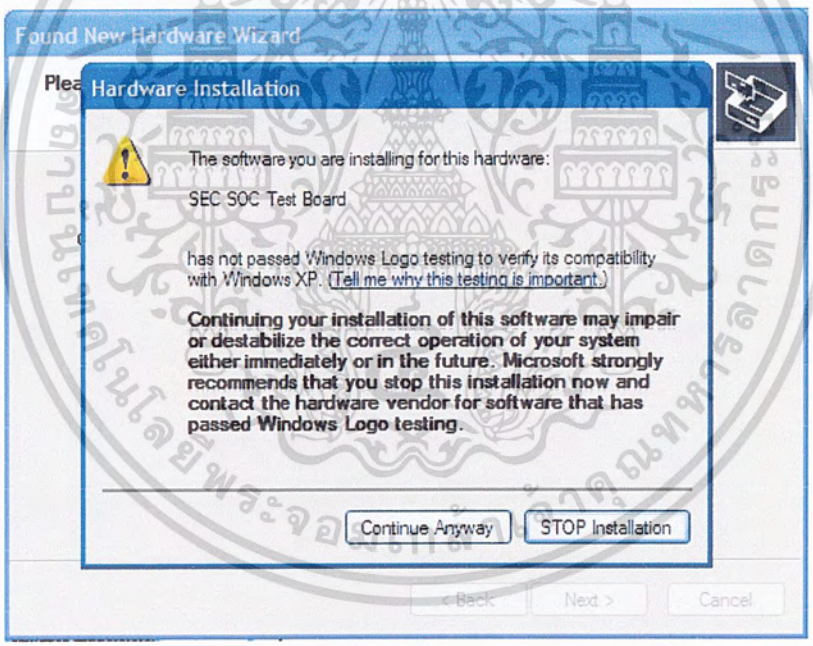
รูป ก.4 การลง USB Driver: Install from a list or specific location (Advanced)



รูป ก.5 การลง USB Driver: Choose your search and installation options



รูป ก.6 การลง USB Driver: Searching for SEC S3C2410X Test B/D

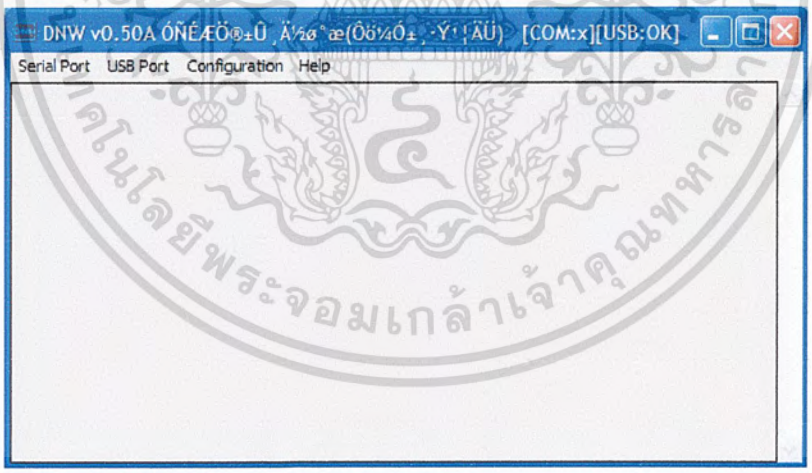


รูป ก.7 การลง USB Driver: Hardware Installation

รูป ก.8 การลง USB Driver: Completing the Found New Hardware Wizard

ก.1.2 ตรวจสอบความถูกต้องของการติดตั้ง driver

ด้วยการเรียกใช้โปรแกรม dwn ที่ได้จาก zip file จะเห็น USB:OK ตรง Header ของโปรแกรม

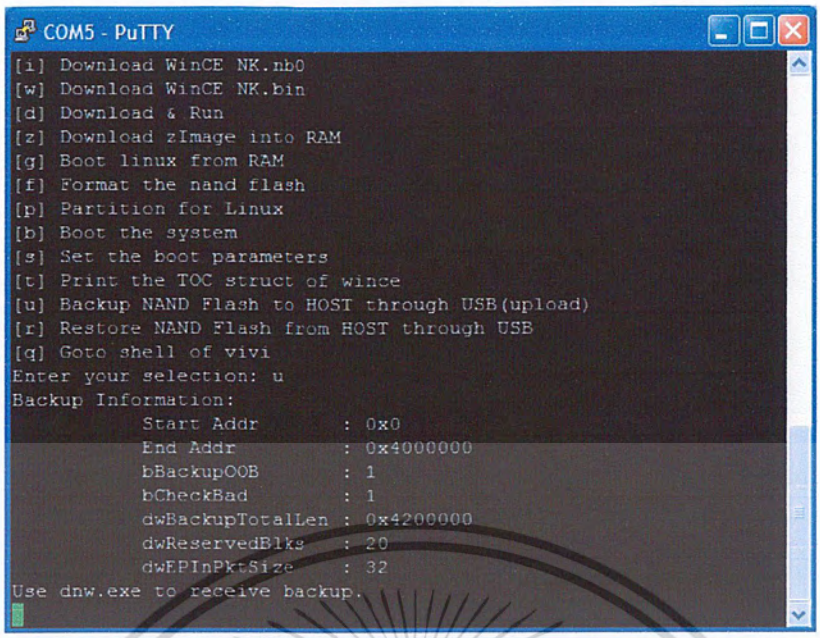


รูป ก.9 USB:OK

มาถึงจุดนี้แสดงว่าระบบเราพร้อมแล้วที่จะ backup หรือ restore NAND flash

ก.1.3 การ Backup NAND Flash

ให้เลือก u จากเมนูของ vivi



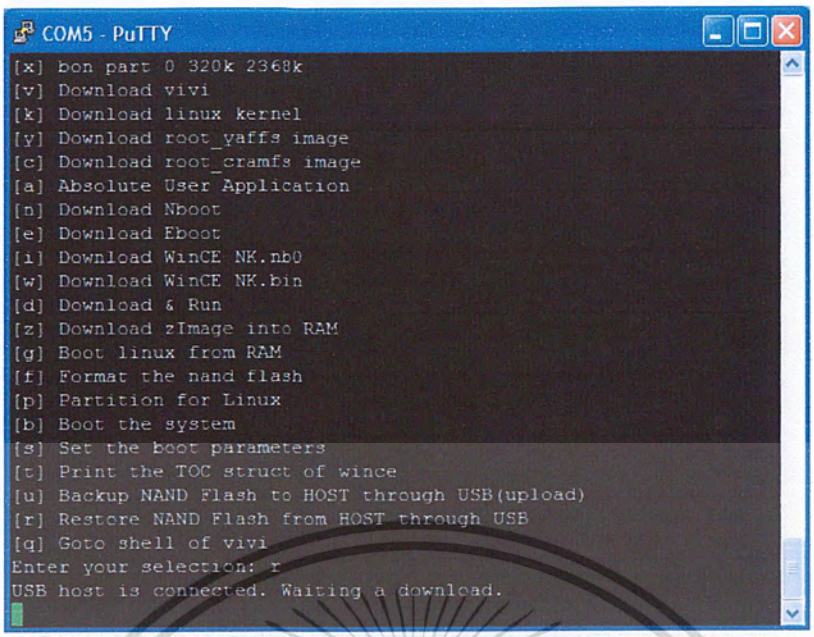
รูป ก.10 Backup Nand Flash

ไปยัง dwn แล้วเลือก Backup Nand Flash to File แล้วกำหนดชื่อและตำแหน่งที่จะเก็บไฟล์

รูป ก.11 Backup: USB Port>Transmit/Restore

ก.1.4 การ Restore NAND Flash

ให้เลือก r จากเมนูของ vivi



รูป ก.12 Restore NAND Flash

ไปยัง dwn แล้วเลือก Transmit/Restore

รูป ก.13 Restore: USB Port>Transmit/Restore

ปิดสวิตซ์ไฟบอร์ดและโยก switch เลือก NOR/NAND กลับไปยัง NAND ในการใช้งานตามปกติ

ก.2 Cross Toolchain สำหรับ mini2440

Cross Tool Chain สำหรับใช้กับ mini2440 มีหลายตัวแต่ที่นิยมใช้คือจากโปรเจ็ค Angstorm

http://qtextended.org/downloads/toolchains/arm920t-eabi.tgz

ก.2.1 การติดตั้ง Toolchain

การติดตั้งให้ทำด้วย root user หากอยู่ใน mode user ธรรมดาให้ใช้คำสั่ง su แล้วตามด้วย root password

```
$ su
Password:
```

รูป ก.14 เข้า root user ด้วย คำสั่ง su

จากนั้นไปยัง directory /opt หากยังไม่มี direcotry /opt ให้สร้างด้วยคำสั่ง mkdir /opt แล้วทำการ download cross tool chain พร้อมกับแตกไฟล์

```
# cd /opt
# wget http://qtextended.org/downloads/toolchains/arm920t-eabi.tgz
# tar xzvf arm920t-eabi.tgz -C /
```

รูป ก.15 ดาวโหลด cross tool chain และแตกไฟล์

ตรวจสอบไฟล์ที่แตกออกมาจะเห็น directory toolchains เพิ่มขึ้นมา ซึ่งภายใต้ toolchains/arm920t-eabi/bin จะเป็นไฟล์ที่ใช้ในการ cross compile

รูป ก.16 แตกไฟล์ cross tool chain เสร็จ

หลังจากติดตั้งแล้วเราจำเป็นต้องเพิ่ม path ของ cross compile สำหรับการเรียกใช้งาน ดังนี้ หากยังอยู่ใน root user ให้ออกจาก root user ไปยัง user ธรรมดาด้วยปุ่ม ctrl-d และเพิ่มข้อความ **export PATH=$PATH:/opt/toolchains/arm920t-eabi/bin** เข้าไปในระบบดังนี้

```
$ echo "export PATH=\$PATH:/opt/toolchains/arm920t-eabi/bin" >> .bashrc
```

รูป ก.17 เพิ่ม path ของ cross compile

หลังจากนั้นทุกครั้งที่เข้า shell console ไฟล์ .bashrc จะทำการเพิ่ม path ให้เราโดยอัตโนมัติ ทดสอบ cross compile โดยคำสั่ง

```
$ arm-angstrom-linux-gnueabi-gcc
arm-angstrom-linux-gnueabi-gcc: no input files
```

รูป ก.18 ทดสอบ path ที่เพิ่มไป

จะเห็นว่าโปรแกรมจะถูกเรียกใช้งาน เป็นอันว่าเสร็จสิ้นการติดตั้ง cross compile tool chain นอกจาก Cross Tool Chain ที่กล่าวมาแล้ว ยังมีอีกหลายตัวที่สามารถลองใช้ http://www.codesourcery.com/sgpp/lite/arm/portal/package3696/public/arm-none-linux-gnueabi/arm-2008q3-72-arm-none-linux-gnueabi-i686-pc-linux-gnu.tar.bz2

ก.3 การสร้างและติดตั้ง u-boot สำหรับ mini2440

ดึง Source code มาจากคลังเก็บด้วยคำสั่ง

```
$ mkdir uboot
$ cd uboot
$ git clone git://repo.or.cz/u-boot-openmoko/mini2440.git
```

รูป ก.19 ดึง Source Code uboot

กำหนดตัวแปร CROSS_COMPILE เพื่อให้ make ใช้ Cross Tool Chain ของ arm สำหรับการ Compile

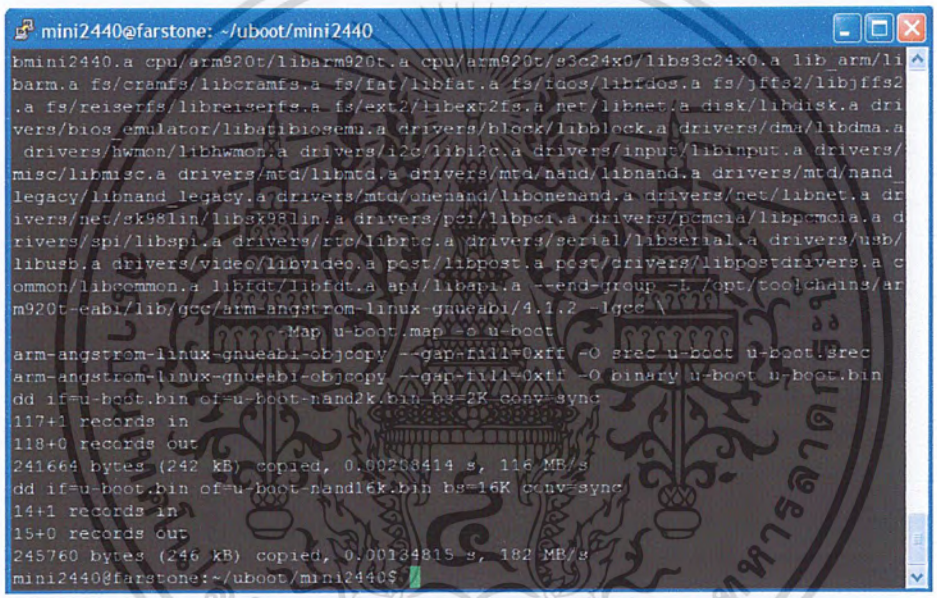
```
$ export CROSS_COMPILE=arm-angstrom-linux-gnueabi-
```

รูป ก.20 กำหนดตัวแปร CROSS_COMPILE

ทำการ compile uboot

```
$ cd mini2440
$ make mini2440_config
$ make all
```

รูป ก.21 ทำการ compile uboot



รูป ก.22 Compile uboot เสร็จแล้วจะได้ u-boot.bin

เมื่อเสร็จสิ้นการ compile เราจะได้ไฟล์ u-boot.bin ซึ่งเราจะได้นำไปใช้ในการติดตั้งบน NAND Flash ต่อไป

ก.3.1 การติดตั้ง u-boot.bin บน NAND Flash

เตรียมพร้อมระบบด้วยการ

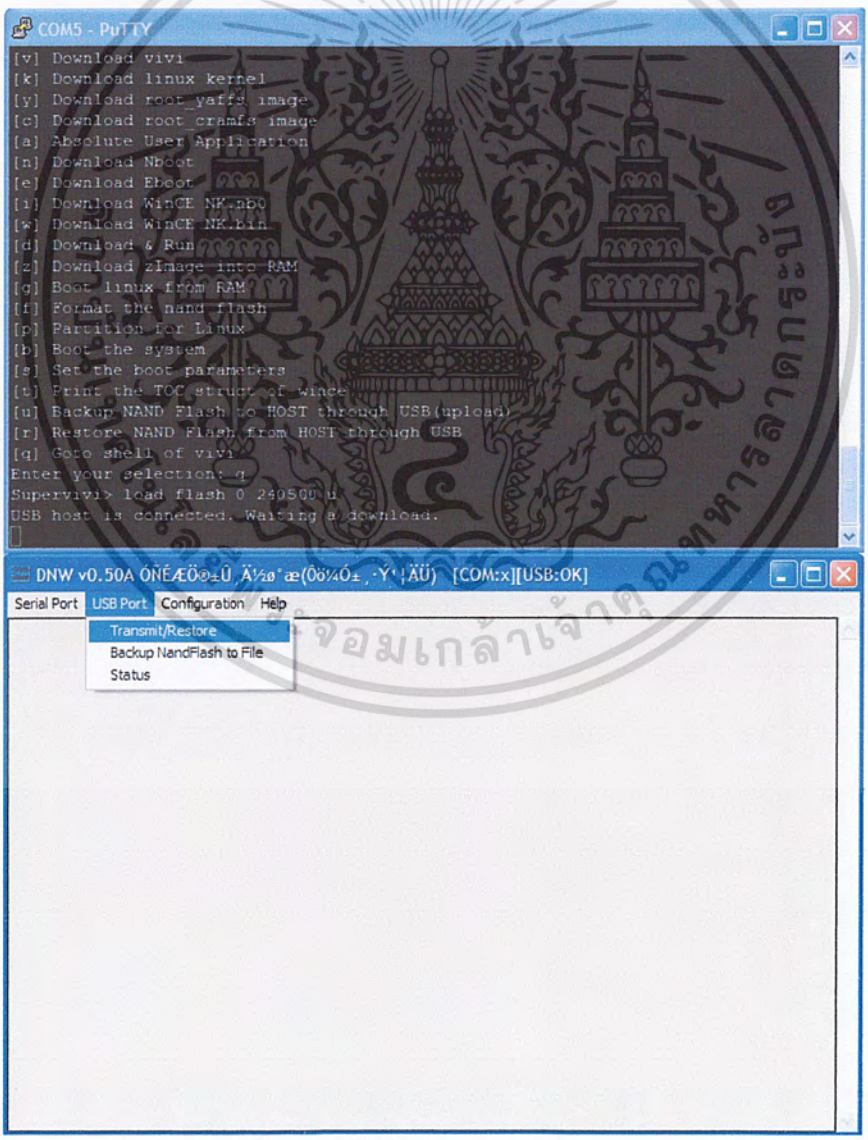
1) ต่อ USB Client ของ mini2440 ไปยัง PC Host พร้อมทั้ง Start program dwn.exe
2) ต่อ Serial port mini2440 ไปยัง PC Host พร้อมทั้งรันโปรแกรม Terminal emulator อย่างเช่น putty หรือ Hyperterminal ด้วย Buadrate 115200

ก.3.2 บันทึก u-boot ลง NAND Flash

1) ปิดสวิตซ์ไฟบอร์ด
2) เลื่อนสวิตซ์ NOR/NAND ไปที่ **NOR**
3) เปิดสวิตซ์ไฟบอร์ด เลือก q จาก vivi menu แล้วพิมพ์ข้อความข้างล่างโดย ให้แทนที่ 240500 ด้วยขนาดไฟล์จริงของ u-boot.bin ที่ได้จากการคอมไพล์ข้างต้น จากนั้นให้ upload ไฟล์ u-boot.bin ด้วย dwn.exe

```
Supervivi> load flash 0 240500 u
USB host is connected. Waiting a download.
```

รูป ก.23 เลือก q จาก vivi menu



รูป ก.24 upload ไฟล์ u-boot.bin ด้วย dwn.exe

u-boot จะถูกบันทึกไปยัง NAND Flash

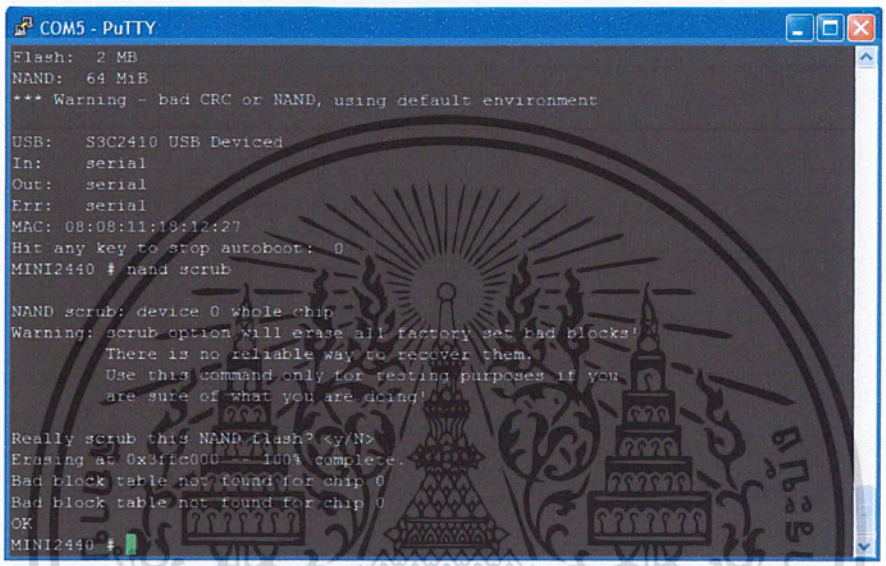
ก.3.3 สร้าง bbt (Bad Block Table) ของ NAND Flash

1) Turn Power Off

2) Switch NOR/NAND Selector to **NAND**

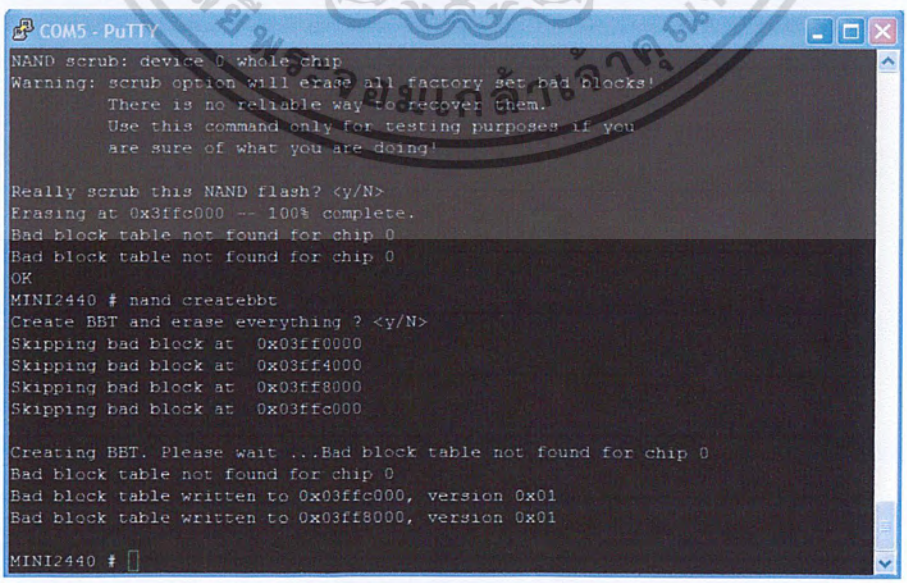
3) Turn Power Switch on

mini2440 จะบูทเข้าสู่ u-boot ให้ใช้คำสั่ง nand scrub เพื่อลบ bbt



รูป ก.25 nand scrub

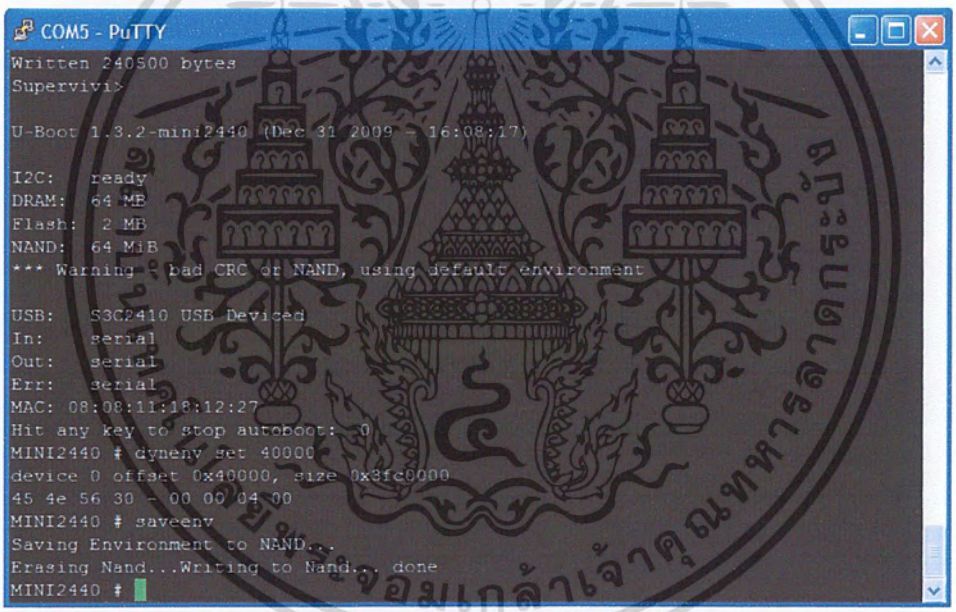
ใช้คำสั่ง nand createbbt เพื่อสร้าง bad block table ใหม่



รูป ก.26 nand createbbt

ก.3.4 สร้าง Environment สำหรับ u-boot

1) จากขั้นตอนที่ 2 โปรแกรม u-boot ที่ถูกบันทึกไว้จะถูกลบไปจาก NAND Flash ดังนั้นเราจึงจำเป็นต้องทำตามขั้นตอนที่ 1 ใหม่อีกครั้งเพื่อบันทึกโปรแกรม u-boot กลับเข้าไปใน NAND Flash
2) ปิดสวิตซ์ไฟบอร์ด
3) เลื่อนสวิตซ์ NOR/NAND ไปที่ **NAND**
4) เปิดสวิตซ์ไฟบอร์ด
5) mini2440 จะบูท u-boot โปรดสังเกตว่าจะมีข้อความ *** Warning - bad CRC or NAND, using default environment
6) ใช้คำสั่ง dynenv set 40000 เพื่อทำการสร้าง environment ใหม่ และตามด้วยคำสั่ง saveenv



รูป ก.27 dynenv set 40000

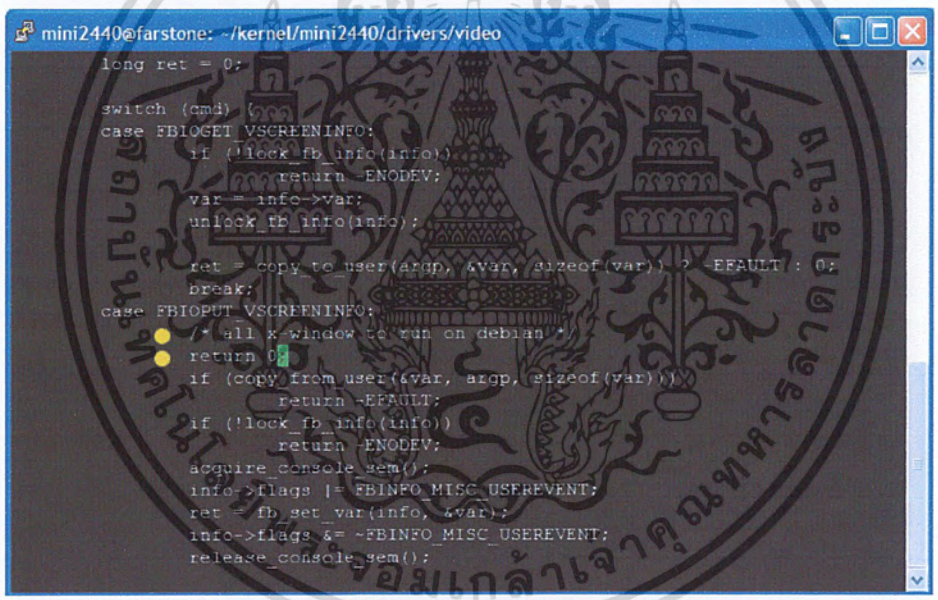
ก.4 การสร้าง Kernel สำหรับ mini2440

ดึง Source code มาจากคลังเก็บด้วยคำสั่ง

```
$ mkdir kernel

$ cd kernel

$ sudo apt-get install git-core

$ git clone git://repo.or.cz/linux-2.6/mini2440.git

$ cd mini2440

$ mkdir -p ../kernel-bin
```

รูป ก.28 ดึง Source code Kernel

สำหรับ x-window กับ debian file system ต้องแก้ไขไฟล์ drivers/video/fbmem.c เพื่อแก้บั๊กด้วยการเพิ่มบรรทัด return 0; ดังนี้



รูป ก.29 แก้ไขไฟล์ drivers/video/fbmem.c

สร้าง .config สำหรับ mini2440

```
$ CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/

mini2440_defconfig
```

รูป ก.30 สร้าง .config สำหรับ mini2440

สำหรับผู้ที่ไม่ชอบ font ขนาดเล็กที่ display ที่ lcd ให้แก้ไขไฟล์ดังนี้

```
แก้ไขไฟล์ ../kernel-bin/.config

เปลี่ยนบรรทัด
CONFIG_FONT_MINI_4x6=y
เป็น
# CONFIG_FONT_MINI_4x6 is not set

บันทึกไฟล์แล้วทำขั้นต่อไป
```

รูป ก.31 แก้ไขไฟล์ ../kernel-bin/.config

สร้าง Kernel โดยกำหนดให้ output เก็บไว้ที่ ../kernel-bin/

```
$ CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/
```

รูป ก.32 สร้าง Kernel

สร้าง uImage สำหรับ u-boot ด้วย mkimage (mkimage สร้างจาก การสร้างและติดตั้ง uboot สำหรับ mini2440)

```
$ ~/uboot/mini2440/tools/mkimage -A arm -O linux -T kernel -C none -a 0x30008000 -e
0x30008000 -d ../kernel-bin/arch/arm/boot/zImage uImage
```

รูป ก.33 สร้าง uImage

สิ่งหนึ่งที่มากับการคอมไพล์เสมอคือ lib modules ซึ่งจะต้องคู่กันกับ version ของ kernel และจะต้องถูกนำไปรวมในการสร้าง file system เราสามารถ copy lib modules ที่สร้างไว้แล้วไปเก็บไว้ยัง directory ที่ต้องการ (ในกรณีนี้คือ ../kernel-modules) ดังนี้

```
$ mkdir -p ../kernel-modules
$ CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/
INSTALL_MOD_PATH=../kernel-modules modules_install
```

รูป ก.34 Copy lib modules ที่สร้างไว้แล้วไปเก็บไว้ยัง directory ที่ต้องการ

สมมติว่า file system ของ mini2440 อยู่บน sd card และ เชื่อมต่อกับ Host ที่ /mnt/sdb2 ให้ทำดังนี้

```
$ sudo cp -Rpf ../kernel-modules/* /mnt/sdb2
```

รูป ก.35 เชื่อมต่อกับ Host ที่ /mnt/sdb2

ภายใต้ /mnt/sdb2 ก็ควรจะมี directory ดังนี้ /lib/modules/2.6.32-rc8

หมายเหตุ 2.6.32-rc8 คือ version ของ kernel ซึ่งอาจจะเป็นค่าอื่นถ้า kernel คุณมีเวอร์ชั่นที่ต่างออกไปถ้าหากมี File system ต่ออยู่กับ /mnt/sdb2 อยู่แล้วในระหว่างการสร้าง kernel ก็ให้ใช้คำสั่งนี้ได้โดยตรงเลย

```
# CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/
INSTALL_MOD_PATH=/mnt/sdb2 modules_install
```

รูป ก.36 สร้าง kernel

ก.5 การสร้าง Debian File System สำหรับ mini2440

Linux Debian Distribution เป็นระบบที่มี packages มากมายพร้อมใช้งาน นอกจากนี้ debian ยังรองรับ ARM ด้วย บทความนี้แสดงการสร้าง File System ของระบบ Debian เพื่อใช้วิ่งบน Mini2440

การสร้าง File System จำเป็นต้องใช้คำสั่ง sudo ในฐานะ root ดังนั้นหากท่านใช้คำสั่ง sudo แล้วพบ error username is not in the sudoers file. ให้แก้ไขไฟล์ /etc/sudoers แล้วเพิ่มเติมข้อความ **username ALL=(ALL) ALL** ดังนี้ โดยให้แทน username ด้วยชื่อของ user ที่คุณใช้ในการสร้าง file system

```
# User privilege specification
root    ALL=(ALL) ALL
```

ก.5.1 เตรียม file system พื้นฐานของ debian

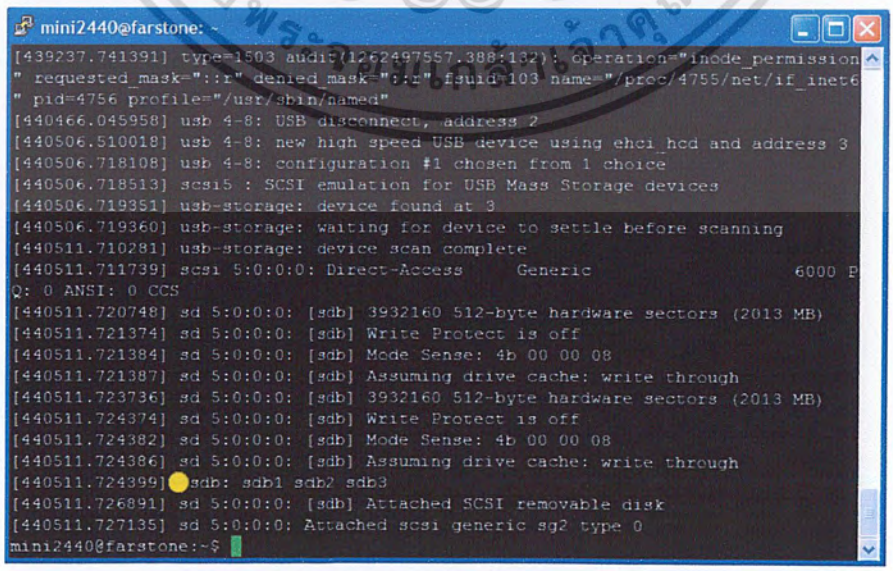
สำหรับการติดตั้งระบบที่สมบูรณ์ในขั้นตอนถัดไป โดยขั้นตอนนี้จะทำบน PC Host

```
$ mkdir ~/armel-rootfs
$ sudo debootstrap --arch=armel --include=ifupdown,udev,procps,netbase,vim-tiny,module-init-
tools,wget,openssh-server,screen,apmd --foreign lenny ~/armel-rootfs
http://ftp.de.debian.org/debian
$ cd ~/armel-rootfs
$ tar cfjv ../armel-rootfs.tar.bz2 *
$ cd ..
```

รูป ก.37 เตรียม file system พื้นฐานของ debian

ก.5.2 เตรียม SD card

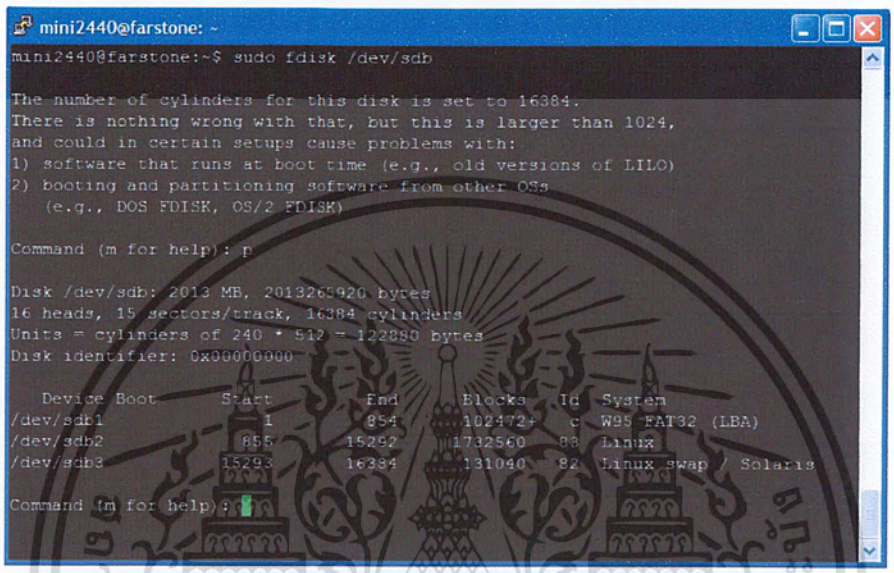
เพื่อถ่าย file system ที่สร้างในข้อ ก.5.1 และนำไปติดตั้งต่อขั้นที่สองบนเครื่อง mini2440 เสียบ sd card บน linux host และใช้คำสั่ง dmesg เพื่อตรวจสอบดูว่า sd card ที่เราเพิ่งเสียบเข้าไปใช้ device อะไร จากตัวอย่างจะเห็นว่า sd card ที่เพิ่งต่อกับเครื่องจะถูกมองเป็น sdb



รูป ก.38 เตรียม sd card เพื่อถ่าย file system

จากข้อมูลของ sd card ซึ่งต่อกับระบบเป็น /dev/sdb เราจะใช้คำสั่ง fdisk เพื่อสร้าง partition ที่มีลักษณะดังนี้

1) sdb1 - type c (fat32) ขนาด 100MB
2) sdb2 - type 83 (Linux) ขนาดเท่ากับที่เหลือ ลบด้วย 128MB
3) sdb3 - type 82 (Linux swap) ขนาด 128MB

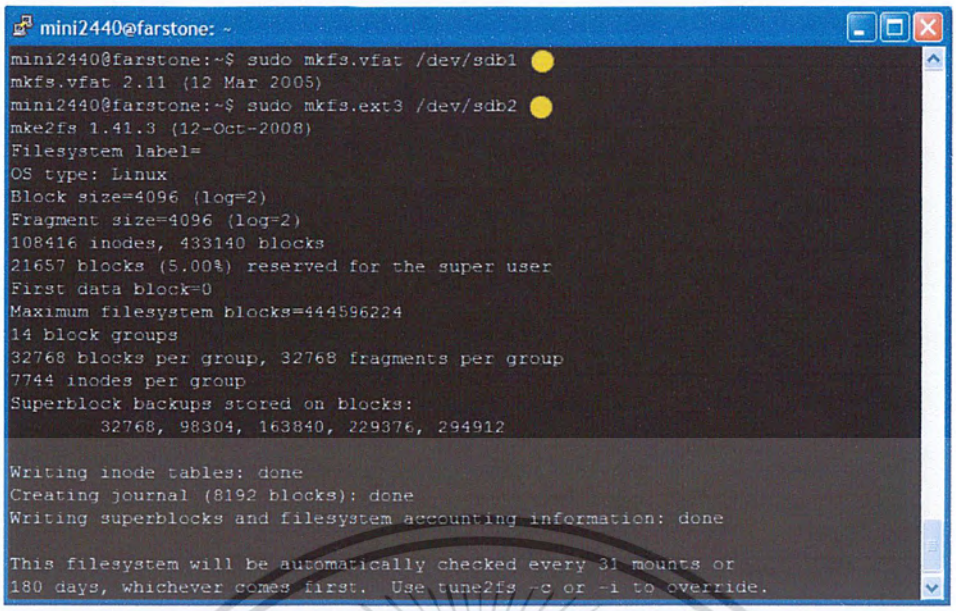


รูป ก.39 fdisk เพื่อสร้าง partition

หลังจากสร้าง partition สามอันตามข้างต้นแล้ว ให้ Format sdb1 เป็น FAT และ sdb2 เป็น ext3 ดังนี้

```
$ sudo mkfs.vfat /dev/sdb1

$ sudo mkfs.ext3 /dev/sdb2
```

รูป ก.40 สร้าง partition



รูป ก.41 Format sdb1 เป็น FAT และ sdb2 เป็น ext3

ต่อด้วยการแตกไฟล์ File System ที่ได้จากขั้นตอนที่ 1 ลงไปยัง partition ที่สอง

```
$ sudo mount /dev/sdb2 /mnt/sdb2
$ cd /mnt/sdb2
$ sudo tar -xpjvf ~/armel-rootfs.tar.bz2
$ sync
```

รูป ก.42 แตกไฟล์ File System

copy uImage(kernel) และ lib modules (ที่ได้มาจาก การสร้าง Kernel สำหรับ mini2440) ไปยัง partition1 และ 2

```
$ sudo mount /dev/sdb1 /mnt/sdb1
$ sudo cp ~/kernel/mini2440/uImage /mnt/sdb1
$ sudo cp -Rp ~/kernel/kernel-modules/lib/* /mnt/sdb2/lib
$ sync
```

รูป ก.43 copy uImage(kernel) และ lib modules ไปยัง partition1 และ 2

เตรียมพร้อมก่อนการติดตั้ง Debain File System ต่อในขั้นที่สอง

```
$ su
# cd /mnt/sdb2
# echo "proc /proc proc none 0 0" >> etc/fstab
# echo "mini2440" > etc/hostname
# echo '127.0.0.1 localhost.localdomain localhost' > etc/hosts
# mknod dev/console c 5 1
# echo 'deb http://ftp.de.debian.org/debian lenny main' > etc/apt/sources.list
# cd /
# umount /mnt/sdb1
# umount /mnt/sdb2
```

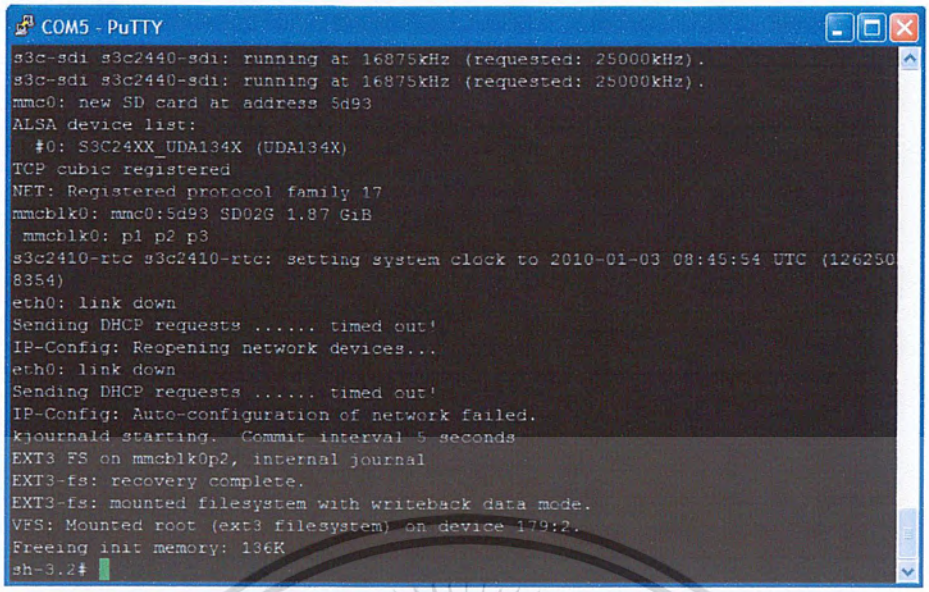
รูป ก.44 เตรียมพร้อมก่อนการติดตั้ง Debain File System

การติดตั้ง debian file system ในขั้นที่สองจะต้องทำบน mini2440 ดังนั้นให้ถอด sd card จาก PC Host แล้วเสียบเข้าช่อง sd card ของ mini2440 จากนั้นเปิดเครื่องเพื่อบูทระบบเข้า u-boot แล้วพิมพ์ข้อความดังนี้

```
setenv bootcmd 'mmcinit ; fatload mmc 0:1 0x31000000 uimage ; bootm 0x31000000'
setenv bootargs 'console=ttySAC0,115200 noinitrd root=/dev/mmcblk0p2 rootwait=4 rw ip=dhcp
init=/bin/sh'
boot
```

รูป ก.45 การ load uImage จาก sd card partition 1 และ connect rootfs จาก partition 2

โดยคำสั่งข้างต้นจะทำให้ mini2440 จะทำการ load uImage จาก sd card partition 1 และ connect rootfs จาก partition 2



รูป ก.46 load uImage จาก sd card partition 1 และ connect rootfs จาก partition 2

หลังจากนั้นทำการติดตั้ง debootstrap ภาคสองดังนี้

```
sh-3.2# mount /proc /proc -t proc
sh-3.2# export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
sh-3.2# /debootstrap/debootstrap --second-stage
```

รูป ก.47 ติดตั้ง debootstrap

เตรียมสิ่งที่จำเป็นสำหรับการบูทเพื่อวิ่ง Debian File System เต็มระบบในขั้นตอนสุดท้าย

```
sh-3.2# mknod dev/ttySAC0 c 204 64
sh-3.2# echo ttySAC0 >> /etc/securetty
sh-3.2# printf "T0:123:respawn:/sbin/getty 115200 ttySAC0\n" >> /etc/inittab
sh-3.2# printf "auto eth0\niface eth0 inet dhcp\n" >> /etc/network/interfaces
```

รูป ก.48 เตรียมสิ่งที่จำเป็นสำหรับการบูท Debian File System

กำหนด nameserver ให้กับระบบเพื่อการต่อเน็ท ในกรณีนี้ต่อการต่ออินเตอร์เน็ทกระทำผ่าน router ที่มี ip 192.168.1.1

```
sh-3.2# printf "nameserver 192.168.1.1" >> /etc/resolv.conf
```

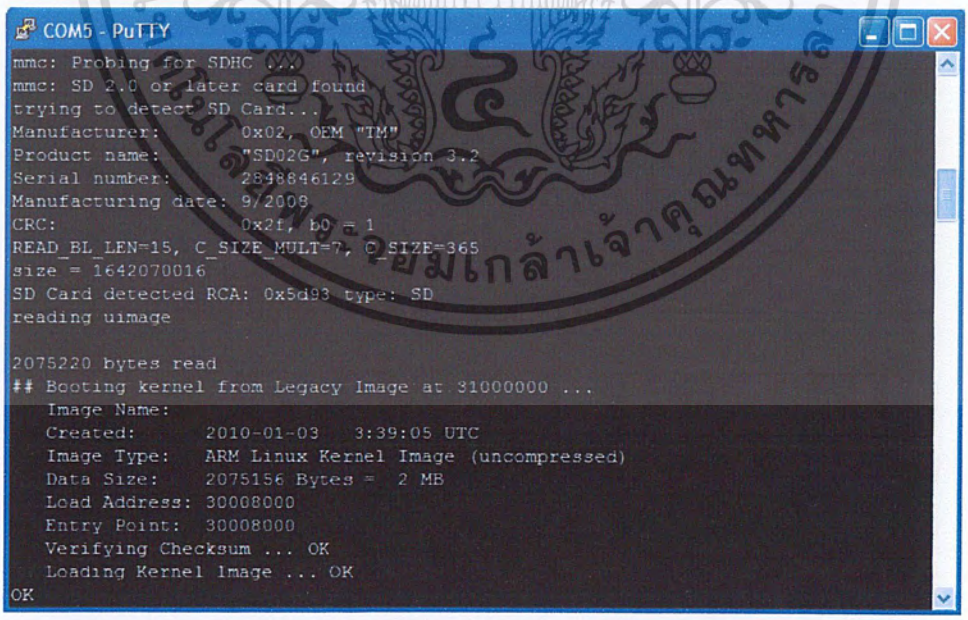
รูป ก.49 กำหนด nameserver

reboot เข้า u-boot menu และใส่ค่าดังนี้สำหรับการบูทเพื่อวิ่ง Full Debian System จาก sd card พร้อมทั้งใช้คำสั่ง saveenv เพื่อบันทึกฐาวร

```
setenv bootcmd 'mmcinit ; fatload mmc 0:1 0x31000000 uimage ; bootm 0x31000000'
setenv bootargs 'console=ttySAC0,115200n8 rootdelay=3 root=/dev/mmcblk0p2 rw
rootfstype=ext3 rootwait'
saveenv
boot
```

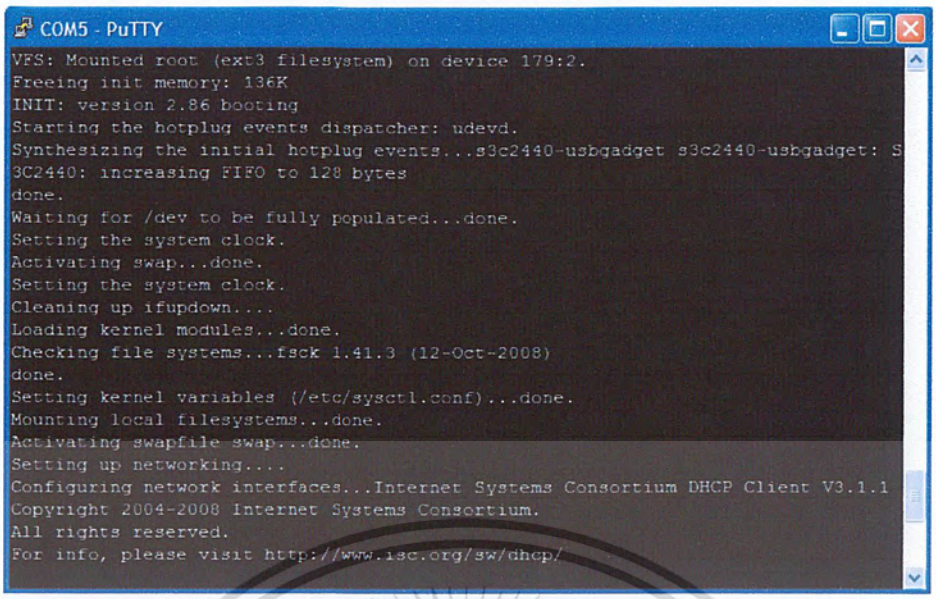
รูป ก.50 Reboot เข้า u-boot menu

เมื่อเจอคำสั่งบูท u-boot จะทำตามคำสั่งที่กำหนดไว้ใน bootcmd คือการ load uImage จาก fat32 partition ด้วยคำสั่ง fatload และรัน kernel



รูป ก.51 load uImage จาก fat32 partition ด้วยคำสั่ง fatload และรัน kernel

จากนั้นก็จะ mount file system ตามที่กำหนดไว้ใน bootargs คือ partition ที่ 2



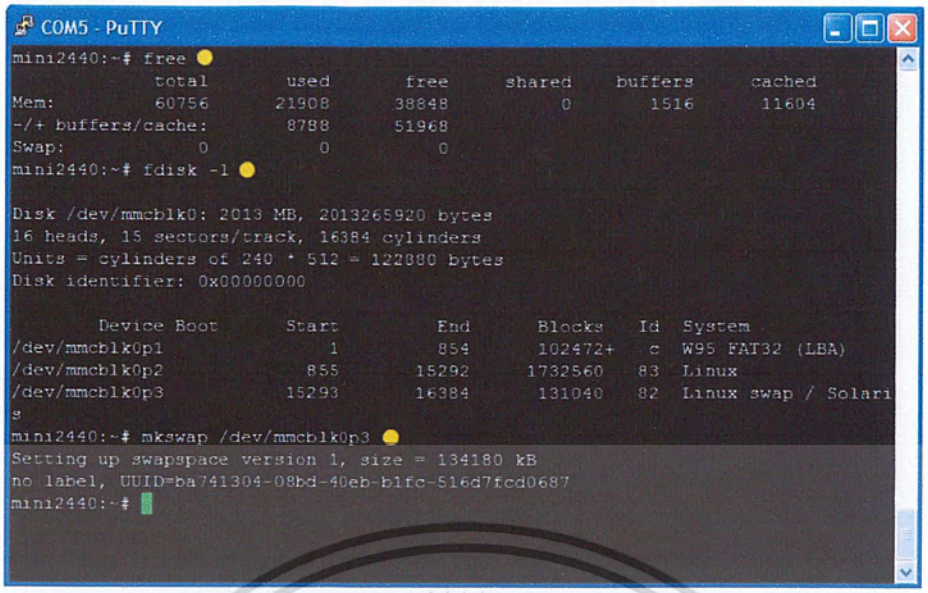
รูป ก.52 mount file system

ในกรณีที่ท่านเจอคำเตือนระหว่างบูทว่า .udev/ already exists on the static /dev! (warning). ให้ทำดังนี้

1) shutdown mini2440 แล้วถอด sd card ไปต่อกับ PC Host
2) sudo mount /dev/sdb2 /mnt/sdb2 ในกรณีที่ระบบมอง /dev/sdb2
3) sudo rm -Rf /mnt/sdb2/dev/.udev
4) sudo umount /mnt/sdb2
5) ถอด sd card กลับไปวิ่งบน mini2440 ปัญหาก็จะหมดไป

สำหรับขั้นตอนสุดท้าย เราจะทำการเซ็ทอัพ swap partition ให้กับระบบ

1) ใช้คำสั่ง free เพื่อตรวจสอบหน่วยความจำของระบบ จะสังเกตุเห็นว่า swap ไม่ได้เปิดใช้งาน
2) ใช้คำสั่ง fdisk -l เพื่อตรวจหา swap partition
3) แจ้งให้ระบบรู้ถึง swap partition ด้วยคำสั่ง mkswap /dev/mmcblk0p3 และเปิดการใช้ด้วยคำสั่ง swapon /dev/mmcblk0p3



รูป ก.53 คำสั่ง mkswap /dev/mmcblk0p3

ใช้คำสั่ง free อีกครั้งจะเห็นการใช้งาน swap ของระบบ

รูป ก.54 คำสั่ง free

ก.6 การใช้ usb wireless adapter กับ mini2440

เนื่องจากบอร์ด mini2440 ไม่ได้มาพร้อมกับความสามารถในการต่อเน็ทเวิร์คแบบไร้สาย ดังนั้นหากต้องการ ที่จะใช้เน็ทเวิร์คบนบอร์ด mini2440 แบบไร้สายจำเป็นต้องใช้ usb wireless adapter บทความนี้จะนำเสนอการเลือกใช้ usb wireless adapter การสร้างจัดเตรียม driver ตลอดจนถึงการเซ็ทอัพเพื่อใช้งาน

ก.6.1 การเลือกใช้ usb adapter

เราจะต้องเลือกใช้รุ่นที่ chip ภายในของมันมี Driver ที่รองรับโดย linux kernel มิฉนั้นแล้วท่านจะต้องไปหา driver เอาเอง ก่อนที่จะซื้อให้ตรวจสอบได้จากที่นี่ครับ

http://linuxwireless.org/en/users/Devices/USB

rtl8187	LevelOne	WNC-0305USB	0x0bda	0x8187
rtl8187	NetGear	WG111v2	0x0846	0x6a00
rtl8187	NetGear	WG111v3	0x0846	0x4260
rt73usb	Alfa	AWUS036S	0x148f	0x2573
rt73usb	Edimax	EW7318USG	0x148f	0x2573

รูป ก.55 NetGear รุ่น WG111v3

จากตัวอย่าง เราจะใช้ usb wifi adapter ของ NetGear รุ่น WG111v3 ซึ่งจะใช้ Driver rtl8187 เป็นหลักในการสาธิต

ก.6.2 ติดตั้งโปรแกรมที่จำเป็น

ระบบที่จะใช้จะเป็น Debian Distribution สำหรับ mini2440 เราเริ่มต้นด้วยการติดตั้งโปรแกรมที่จำเป็นก่อน

```
# apt-get update
# apt-get install usbutils wireless-tools
```

รูป ก.56 apt-get update

ก.6.3 เสียบ usb wifi adapter เข้ากับบอร์ด

จะเห็นข้อความดังนี้ปรากฏบน serial console

```
mini2440:~# usb 1-1: new full speed USB device using s3c2410-ohci and address 2
usb 1-1: configuration #1 chosen from 1 choice
```

รูป ก.57 เสียบ usb wifi adapter

ใช้คำสั่ง lsusb เพื่อตรวจสอบรายละเอียดของ usb devices ที่ต่ออยู่กับระบบ จะเห็นว่าระบบรายงานข้อมูลของ usb wifi ตรงกับรายละเอียดในข้อที่ 1

```
mini2440:~# lsusb
Bus 001 Device 002: ID 0846:4260 NetGear, Inc. WG111(v3) 54 Mbps Wireless [RealTek
RTL8187B]
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

รูป ก.58 lsusb

ใช้คำสั่ง ifconfig -a เพื่อตรวจสอบว่าระบบของเรามี Device Driver สำหรับ usb wifi ตัวนี้หรือเปล่า

```
mini2440:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 01:02:03:04:05:07
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:192 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:33147 (32.3 KiB)  TX bytes:836 (836.0 B)
          Interrupt:51 Base address:0x300

lo        Link encap:Local Loopback
          LOOPBACK  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

รูป ก.59 ifconfig -a

ถ้าระบบของเรามี Device Driver ที่รองรับ เราจะเห็น device ที่ชื่อ wlan0 ในกรณีนี้ปรากฏว่ามีเพียง eth0 ดังนั้นเราจำเป็นต้องติดตั้ง Device Driver เพิ่มเติมสำหรับ usb wifi adapter ตัวนี้

ก.6.4 สร้าง Device Driver สำหรับ usb wifi

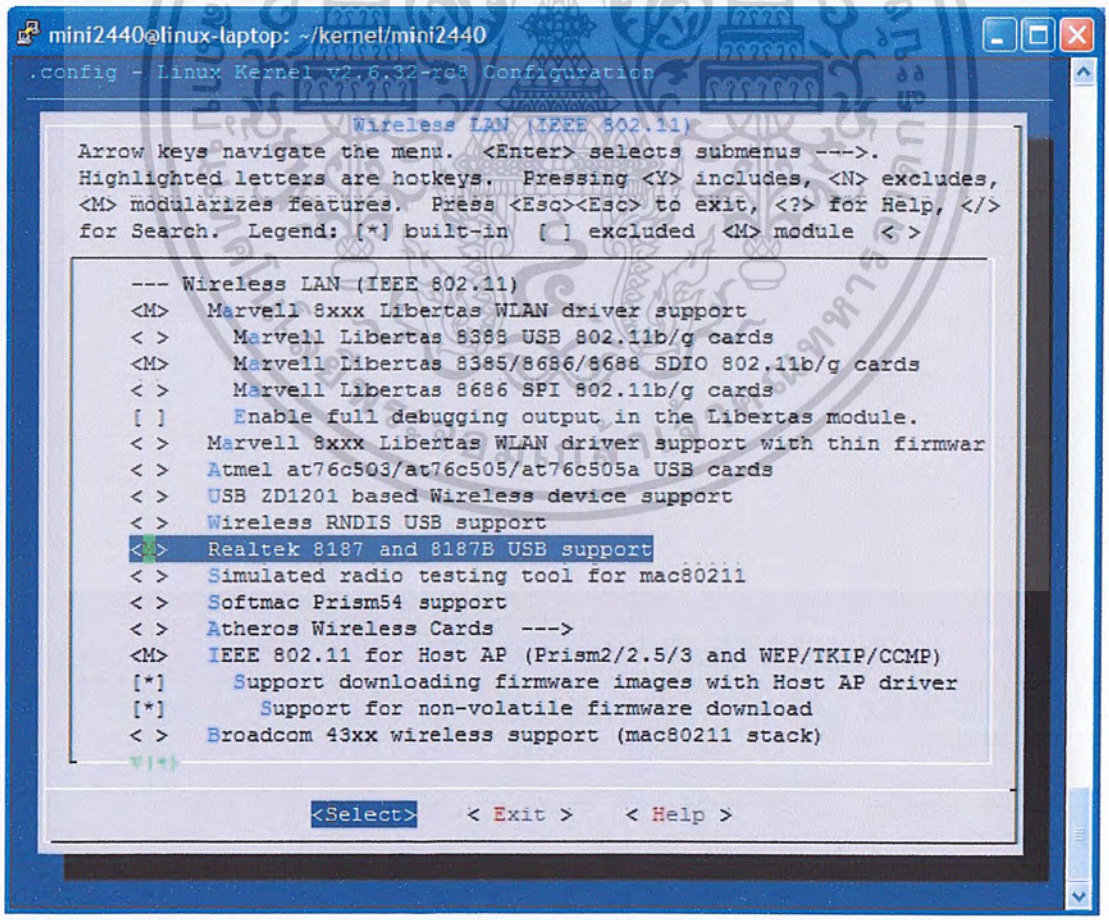
ของเราด้วยการแก้ไข config ในขั้นตอนการสร้าง kernel ดังนี้

1) ให้ทำตามขั้นตอน การสร้าง Kernel สำหรับ mini2440 จนกระทั่งก่อนขั้นตอนการสร้าง uImage
2) ทำการ config ระบบเพื่อเพิ่ม Driver ด้วยคำสั่งดังนี้

```
$ cd ~/kernel/mini2440
$ CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/
menu_config
```

รูป ก.60 การ config ระบบเพื่อเพิ่ม Driver

เลือกเมนูตามลำดับดังนี้ Device Drivers/Network device support/Wireless LAN/Wireless LAN (IEEE 802.11)



รูป ก.61 Wireless LAN (IEEE 802.11)

จากข้อมูลในข้อ 1 และ 3 แสดงให้เห็นว่า usb adapter ของเราใช้ Driver RTL8187B ดังนั้นให้เลื่อน cursor ไปยัง Realtek 8187 and 8187B USB support และเคาะ space bar เพื่อเลือกให้สร้าง Module

เลือก exit ไปจนกระทั่งก่อนออกจากโปรแกรมและ save config หลังจากนั้นทำการสร้าง uImage และสร้าง Modules ตามขั้นตอนดังนี้

```
$ CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/
$ ~/uboot/mini2440/tools/mkimage -A arm -O linux -T kernel -C none -a 0x30008000 -e
0x30008000 -d ../kernel-bin/arch/arm/boot/zImage uImage
$ rm -Rf ../kernel-modules/*
$ CROSS_COMPILE=arm-angstrom-linux-gnueabi- ARCH=arm make O=../kernel-bin/
INSTALL_MOD_PATH=../kernel-modules modules_install
```

รูป ก.62 สร้าง uImage และสร้าง Modules

เมื่อเสร็จสิ้นขั้นตอนนี้เราจะได้ uImage อยู่ใน ~/kernel/mini2440 และ Device Drivers Modules อยู่ใน ~/kernel/kernel-modules

ก.6.5 ทำการแทนที่ uImage เดิมด้วย uImage ตัวใหม่

คัดลอกไฟล์ภายใต้ ~/kernel/kernel-modules ไปยัง root directory ของ mini2440 หลังจากบูท mini2440 แล้วให้ใช้คำสั่ง depmod -a เพื่อทำการ update module dependency list

```
# depmod -a
```

รูป ก.63 depmod -a

หลังจากนั้นให้เสียบ usb wifi adapter เข้าไปใหม่อีกครั้ง ข้อความที่ปรากฏที่ console จะแสดงให้เห็นว่าระบบรู้จัก usb wifi adapter ของเราแล้ว

```
mini2440:~# usb 1-1: new full speed USB device using s3c2410-ohci and address 3
usb 1-1: configuration #1 chosen from 1 choice
cfg80211: Using static regulatory domain info
cfg80211: Regulatory domain: US
    (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp)
    (2402000 KHz - 2472000 KHz @ 40000 KHz), (600 mBi, 2700 mBm)
    (5170000 KHz - 5190000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
    (5190000 KHz - 5210000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
    (5210000 KHz - 5230000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
    (5230000 KHz - 5330000 KHz @ 40000 KHz), (600 mBi, 2300 mBm)
    (5735000 KHz - 5835000 KHz @ 40000 KHz), (600 mBi, 3000 mBm)
cfg80211: Calling CRDA for country: US
phy0: hwaddr 01:02:03:04:05:06, RTL8187BvE V0 + rtl8225z2
rtl8187: Customer ID is 0x00
Registered led device: rtl8187-phy0::tx
Registered led device: rtl8187-phy0::rx
rtl8187: wireless switch is on
usbcore: registered new interface driver rtl8187
```

รูป ก.64 ข้อความที่ปรากฏที่ console หลังจากรู้จัก usb wifi adapter

ยืนยันอีกครั้งด้วยคำสั่ง ifconfig -a จะสังเกตุเห็น device ที่ชื่อ wlan0 ปรากฏในระบบ

```
mini2440:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 01:02:03:04:05:07
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:205 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34728 (33.9 KiB)  TX bytes:836 (836.0 B)
          Interrupt:51 Base address:0x300

lo        Link encap:Local Loopback
          LOOPBACK  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

wlan0     Link encap:Ethernet  HWaddr 01:02:03:04:05:06
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

รูป ก.65 ยืนยันด้วยคำสั่ง ifconfig -a

ก.6.6 การเซ็ทอัพ usb wifi adapter ของเราให้ต่อเข้ากับระบบเน็ทเวิร์ค ในระบบตัวอย่าง Router ที่ใช้ทำการ encrypt แบบ WEP ข้อมูลที่จำเป็นต้องใช้ในการเซ็ทอัพมีดังนี้

1) ssid ของเน็ทเวิร์ค
2) key
3) channel

ซึ่ง ssid และ channel ถ้าหากไม่ทราบสามารถหาได้ด้วยคำสั่งดังนี้

```
# ifconfig wlan0 up
# iwlist wlan0 scanning
wlan0     Scan completed :
        Cell 01 - Address: 01:02:03:04:05:06
                Channel:4
                Frequency:2.427 GHz (Channel 4)
                Quality=70/70  Signal level=-23 dBm
                Encryption key:on
                ESSID:"project4fun"
                Bit Rates:1 Mb/s; 2 Mb/s; 5.5 Mb/s; 11 Mb/s; 6 Mb/s
                        9 Mb/s; 12 Mb/s; 18 Mb/s
                Bit Rates:24 Mb/s; 36 Mb/s; 48 Mb/s; 54 Mb/s
                Mode:Master
                Extra:tsf=00002e71f47aee11
                Extra: Last beacon: 1675ms ago
```

รูป ก.66 การเซ็ทอัพ usb wifi adapter

สำหรับ key นั้นหากท่านเป็นเจ้าของเน็ทเวิร์ค ท่านต้องรู้อยู่แล้วเพราะท่านเป็นคนกำหนดเอง แต่หากใช้เน็ทเวิร์คอื่น ก็ต้องติดต่อขอ key จากเจ้าของระบบก่อนครับ ในกรณีนี้สมมติให้ key = 1234567890

เมื่อได้ข้อมูลมาแล้ว เราทำการทดลองตรวจสอบและเซ็ทค่าให้กับ wlan0 ของเราด้วยคำสั่ง iwconfig ดังนี้

```
mini2440:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11bg  Mode:Managed  Access Point: Not-Associated
          Tx-Power=27 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off

mini2440:~# iwconfig wlan0 essid project4fun
mini2440:~# iwconfig wlan0 key 1234567890
mini2440:~# iwconfig wlan0 channel 4
mini2440:~# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11bg  ESSID:"project4fun"
          Mode:Managed  Frequency:2.427 GHz  Access Point: 01:02:03:04:05:06
          Bit Rate=1 Mb/s   Tx-Power=27 dBm
          Retry  long limit:7   RTS thr:off   Fragment thr:off
          Encryption key:1234-5678-90
          Power Management:off
          Link Quality=70/70  Signal level=-23 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

รูป ก.67 iwconfig

<u>หมายเหตุ</u> ลองใช้คำสั่ง iwconfig --help เพื่อดู option ทั้งหมด

เมื่อพร้อมแล้วก็ทำการต่อเข้ากับระบบ network ด้วยคำสั่ง dhclient wlan0

```
mini2440:~# dhclient wlan0
Internet Systems Consortium DHCP Client V3.1.1
Copyright 2004-2008 Internet Systems Consortium.
All rights reserved.
For info, please visit http://www.isc.org/sw/dhcp/

Listening on LPF/wlan0/01:02:03:04:05:06
Sending on   LPF/wlan0/01:02:03:04:05:06
Sending on   Socket/fallback
DHCPREQUEST on wlan0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
bound to 192.168.1.143 -- renewal in 39321 seconds.
mini2440:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 01:02:03:04:05:07
          inet addr:192.168.1.101  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:196 errors:0 dropped:0 overruns:0 frame:0
          TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:34547 (33.7 KiB)  TX bytes:836 (836.0 B)
          Interrupt:51 Base address:0x300

wlan0     Link encap:Ethernet  HWaddr 01:02:03:04:05:06
          inet addr:192.168.1.143  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:350 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:74271 (72.5 KiB)  TX bytes:420 (420.0 B)
```

รูป ก.68 ต่อเข้ากับระบบ network ด้วยคำสั่ง dhclient wlan0

เมื่อตรวจสอบด้วย ifconfig จะเห็นว่า wlan0 ต่อกับระบบโดยได้รับ ip address 192.168.1.143

เราจะทำทดสอบด้วยการปลด eth0 ออกจากระบบและใช้ wlan0 ในการติดต่อ network อย่างเดียวดังนี้

```
mini2440:~# ifconfig eth0 down
mini2440:~# ifconfig
wlan0     Link encap:Ethernet  HWaddr 01:02:03:04:05:06
          inet addr:192.168.1.143  Bcast:192.168.1.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:571 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:117970 (115.2 KiB)  TX bytes:420 (420.0 B)

mini2440:~# ping google.com
PING google.com (74.125.65.99) 56(84) bytes of data.
64 bytes from gx-in-f99.1e100.net (11.122.33.44): icmp_seq=1 ttl=50 time=16.5 ms
64 bytes from gx-in-f99.1e100.net (11.122.33.44): icmp_seq=2 ttl=50 time=19.4 ms
64 bytes from gx-in-f99.1e100.net (11.122.33.44): icmp_seq=3 ttl=50 time=16.4 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 16.413/17.460/19.463/1.424 ms
mini2440:~#
```

รูป ก.69 การปลด eth0 ออกจากระบบและใช้ wlan0

ผลจากการ ping แสดงว่าเราสามารถที่จะใช้ usb wireless adapter ของเราติดต่อสื่อสารกับอินเตอร์เน็ทผ่านเน็ทเวิร์คไร้สายได้เป็นที่เรียบ ร้อย

ก.6.7 ตั้งค่าให้ usb wireless adapter ติดต่อกับเน็ทเวิร์ค โดยอัตโนมัติทุกครั้งที่เราบูท mini2440

แก้ไขไฟล์ /etc/network/interfaces โดยใส่ # เพื่อ comment ในส่วนของ eth0 และเพิ่มเติมส่วนของ wlan0 แทนที่ดังนี้

```
# Used by ifup(8) and ifdown(8). See the interfaces(5) manpage or
# /usr/share/doc/ifupdown/examples for more information.
# auto eth0
# iface eth0 inet dhcp

auto wlan0
iface wlan0 inet dhcp
        wireless-key 1234567890
        wireless-essid project4fun
        wireless-channel 4
```

รูป ก.70 แก้ไขไฟล์ /etc/network/interfaces

ทุกครั้งที่บูท mini2440 จะใช้ wlan0 ในการต่อเข้ากับ wireless network โดยอัตโนมัติ สำหรับผู้ที่ใช้ encyption ในแบบอื่นๆให้ดูการเซ็ทอัพได้จาก เอกสารอ้างอิง

ภาคผนวก ข.

การเขียนโปรแกรมแบบ OOP

ข.1 ความหมาย

การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming , OOP) เป็นการเขียนโปรแกรมโดยมีแนวคิดว่า โปรแกรมประกอบขึ้นด้วยวัตถุ(object) ที่ทำกิจกรรมและใช้ข้อมูลร่วมกัน เพื่อให้บรรลุวัตถุประสงค์ที่กำหนดไว้ โดยผู้ที่จะเขียนโปรแกรมเชิงวัตถุได้ดี จำเป็นต้องสามารถ วิเคราะห์แยกแยะและจัดกลุ่ม ข้อมูลและกิจกรรมของโปรแกรมออกเป็นวัตถุย่อย (Object Oriented Analysis and Design) การสร้างแบบจำลองข้อมูลและกิจกรรมของวัตถุด้วยภาษา UML (Unifield Modeling Language) จากนั้นจึงนำโมเดลที่ได้มาสร้างเป็นโปรแกรม โดยใช้ภาษาเชิงวัตถุ ตัวอย่างเช่น java , c++ , php , vb.net ,c#.net ,javascript ฯลฯ[1] อีกความหมายที่น่าสนใจคือ การเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming, OOP) คือหนึ่งในรูปแบบการเขียนโปรแกรมคอมพิวเตอร์ ที่ให้ความสำคัญกับ วัตถุ ซึ่งสามารถนำมาประกอบกันและนำมาทำงานรวมกันได้ โดยการแลกเปลี่ยนข่าวสารเพื่อนำมาประมวลผลและส่งข่าวสารที่ได้ไปให้ วัตถุ อื่นๆที่เกี่ยวข้องเพื่อให้ทำงานต่อไป แนวคิดการเขียนโปรแกรมแบบดังเดิมมักนิยมใช้ การเขียนโปรแกรมเชิงกระบวนการ (Procedural Programming) ซึ่งให้ความสำคัญกับขั้นตอนกระบวนการที่ทำ โดยแบ่งโปรแกรมออกเป็นส่วนๆตามลำดับขั้นตอนการทำงาน แต่แนวคิดการเขียนโปรแกรมเชิงวัตถุนั้นให้ความสำคัญกับ ข้อมูล(data) และ พฤติกรรม(behavior) ของวัตถุ และความสัมพันธ์กันระหว่างวัตถุกันมากกว่า[3]

ข.2 ข้อดีของ OOP เรื่อง Parameter

ข้อดีของ OOP เรื่อง Parameter ที่จะกล่าวถึงก็คือเรื่องของจำนวน parameter ที่เราจะทำการส่งผ่านเข้าไปทำงานในฟังก์ชั่น หากมีจำนวนไม่มากแค่ 2-3 ตัวก็คงไม่มีปัญหาอะไร แต่หากมีจำนวนมากๆถึง 5-10 ขึ้นไป การทำฟังก์ชั่นใช้งานในกรณีนี้คงไม่ใช่แนวคิดที่ดีแน่ แต่หากเป็นการเขียนโปแกรมเชิงวัตุ(OOP) จะไม่เกิดปัญหานี้ เพราะเราสามารถสร้างฟังก์ชั่น(method) ที่รับค่า parameter เป็น object ซึ่งใน object นี้ก็จะประกอบไปด้วยข้อมูลที่เราจะทำการส่งผ่านเข้ามาทำงานในฟังก์ชั่น หรือพูดง่ายๆก็คือกำหนดค่าใน object อาจเป็น 5-10 ขึ้นไปจากนั้นก็ส่ง object นี้เป็น parameter เพียงตัวเดียว

ภาคผนวก ค.

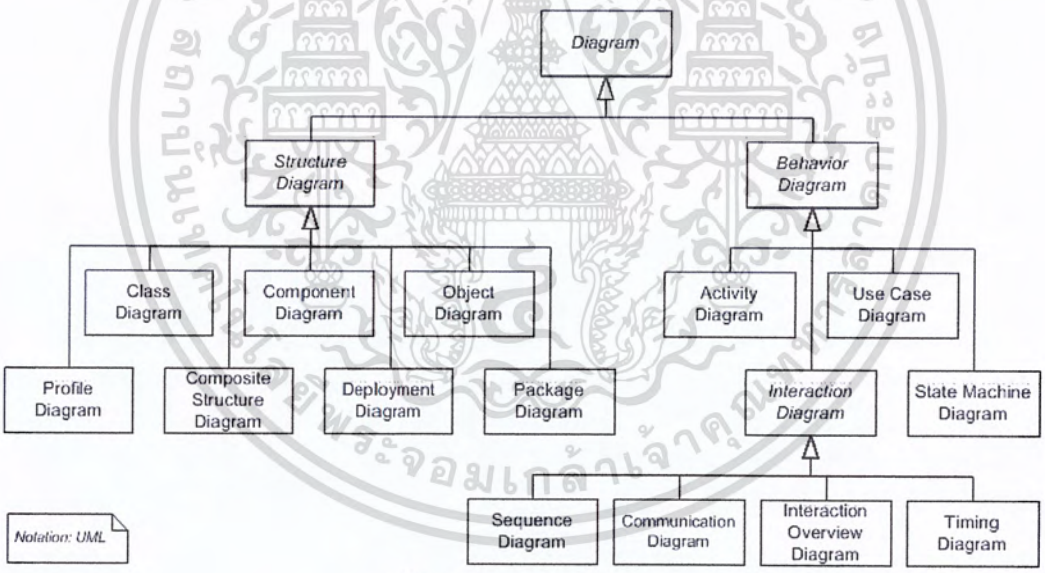
ยูเอ็มแอล (UML: Unified Modeling Language)

ค.1 ความหมาย

ยูเอ็มแอล (UML: Unified Modeling Language) เป็นสัญลักษณ์อันเป็นหนึ่งเดียวกันที่ใช้อธิบาย แสดงรายละเอียด จำลองการสร้าง และจัดการกับเอกสารต่างๆ ในระบบการทำงานจริง เพื่อให้การออกแบบซอฟต์แวร์ที่แทนระบบการทำงานจริงนั้นทำได้โดยง่าย และปรับปรุงวิธีการทำงานที่มีอยู่เดิมให้ดียิ่งขึ้น ยูเอ็มแอลมักใช้เป็นการอธิบายและนำเสนอแนวความคิดของการเขียนโปรแกรมเชิงวัตถุ ก่อนนำไปเขียนโปรแกรมจริง[1]

ค.2 แผนภาพ (Diagrams)

ใน UML 2.2 มีแผนภาพทั้งหมด 3 ประเภทใหญ่ สามารถจัดกลุ่มได้ดังนี



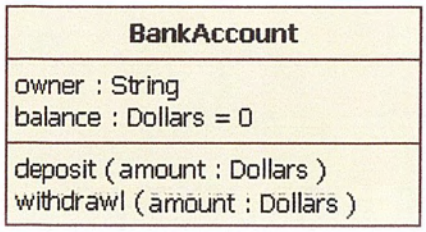
รูป ค.1 UML Diagrams

ค.2.1 แผนภาพประเภทโครงสร้าง (Structural Diagrams)

แผนภาพโครงสร้างนั้นแสดงถึงสิ่งต่างๆที่มีอยู่ในระบบของโมเดลนั้นๆ ถูกใช้อย่างกว้างขวางในการทำเอกสารแสดงสถาปัตยกรรมระบบซอฟแวร์

ค.2.2 Class diagram

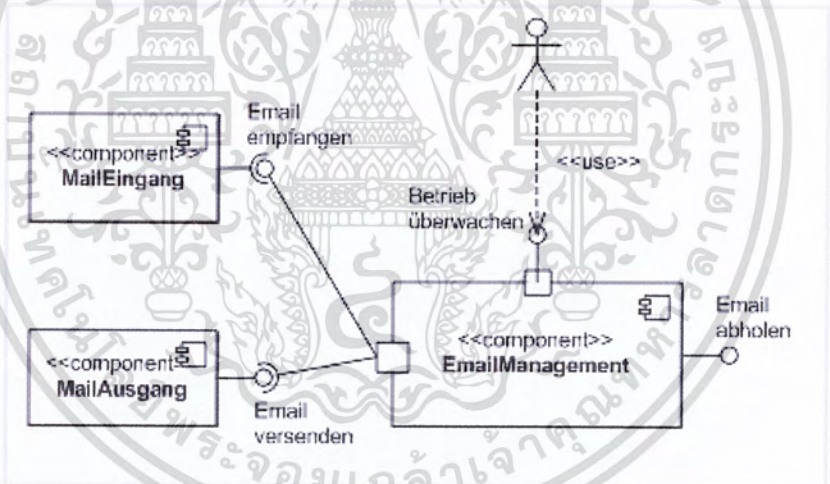
คลาสไดอะแกรม อธิบายถึงโครงสร้างของระบบโดยแสดงระบบของคลาส คุณสมบัติ และความสัมพันธ์ระหว่างคลาส



รูป ค.2 Class Diagram

ค.2.3 Component diagram

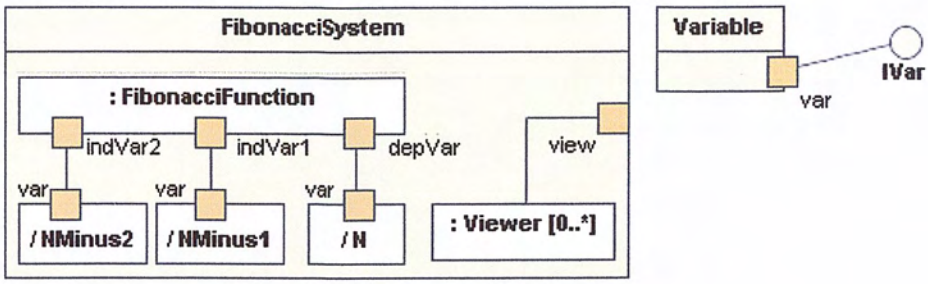
คอมโพเน้นไดอะแกรม อธิบายถึงวิธีการแบ่งระบบออกเป็นส่วนต่างๆพร้อมทั้งแสดงถึงความพาดพิงกันของแต่ละส่วน



รูป ค.3 Component Diagram

ค.2.4 Composite structure diagram

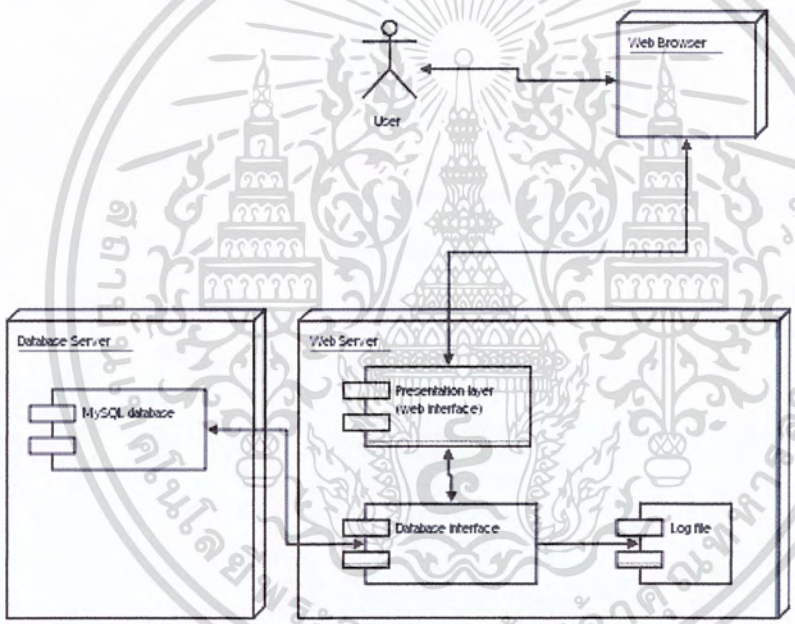
คอมโพไซต์สตัคเจอร์ไดอะแกรม อธิบายถึงโคลงสร้างภายในของคลาส และแสดงถึงความสามารถในการทำงานของคลาสที่เป็นไปได้



รูป ค.4 Composite Structure Diagram

ค.2.5 Deployment diagram

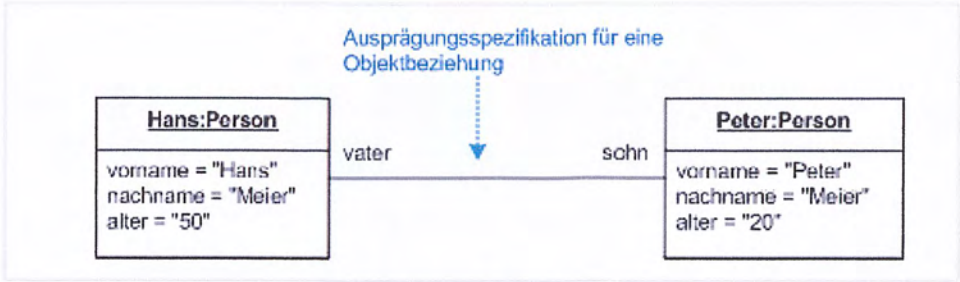
ดีพลอยเม้นท์ไดอะแกรม อธิบายถึงอุปกรณ์ที่จำเป็นต้องใช้ในการทำงาน และสภาพแวดล้อมในขณะทำงาน พร้อมทั้งสิ่งต่างๆที่ใช้งานอยู่บนอุปกรณ์



รูป ค.5 Deployment Diagram

ค.2.6 Object diagram

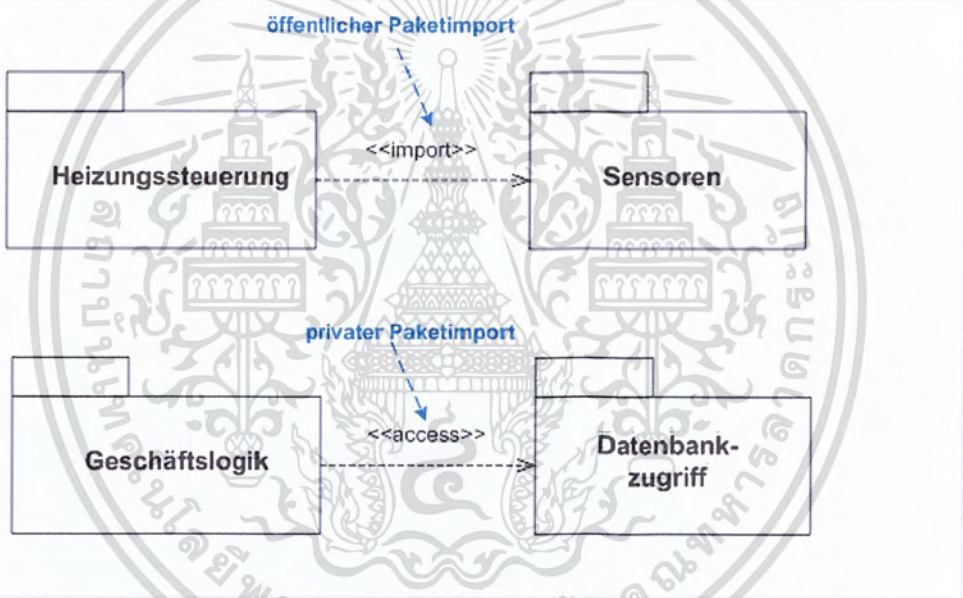
ออปเจ็คต์ไดอะแกรม แสดงถึงโครงสร้างของระบบโมเดล ณ เวลาที่ต้องการ โดยอาจจะแสดงทั้งโครงสร้างหรือแสดงเพียงส่วนใดส่วนหนึ่งของระบบก็ได้



รูป ค.6 Object Diagram

ค.2.7 Package diagram

แพคเก็จไดอะแกรม อธิบายถึงวิธีการแบ่งระบบออกเป็นกลุ่มตามตรรกะโดยแสดงถึงความเกี่ยวข้องกันของแต่ละแพคเก็จด้วย



รูป ค.7 Package Diagram

ค.2.8 Profile diagram

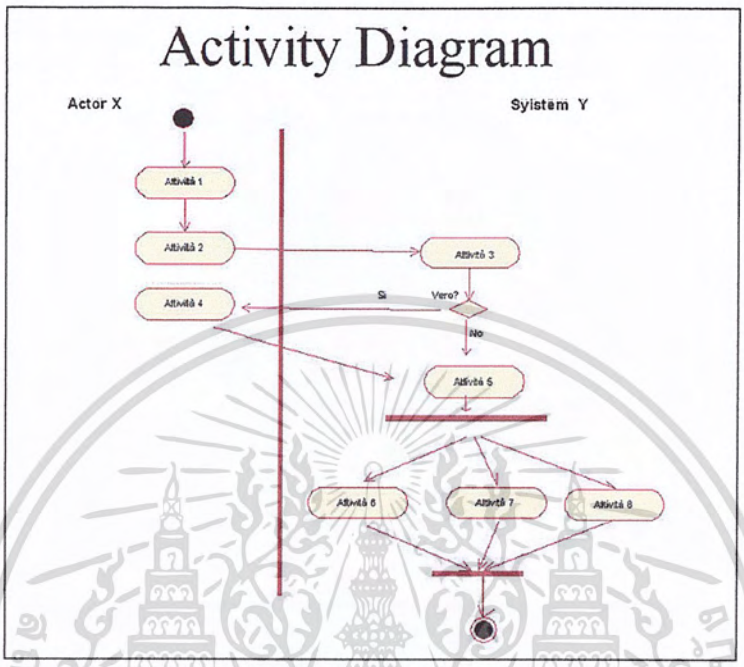
แผนภาพโปรไฟล์ ทำงานในระดับเมตาโทเดล (metamodel) เพื่อแสดงเสตอริโอไทป์ (stereotypes) เป็นคลาสที่มีกฎตายตัวและรูปแบบแพคเกจตายตัว

ค.2.9 แผนภาพประเภทพฤติกรรม (Behavioral Diagrams)

แผนภาพพฤติกรรมนั้นแสดงถึงสิ่งที่เกิดขึ้นในโมเดลนั้น โดยแสดงถึงพฤติกรรมหรือสิ่งที่เกิดขึ้นในระบบ ถูกใช้อย่างแพร่หลายในการอธิบายถึงฟังชั่นต่างๆของซอฟแวร์

ค.2.10 Activity diagram

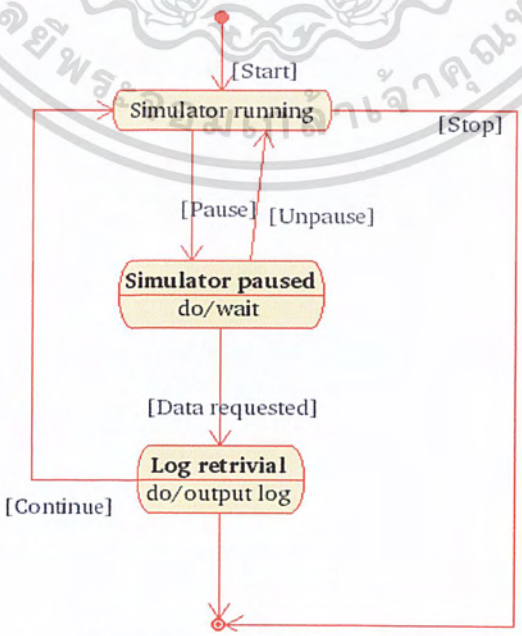
แอคทิวิตีไดอะแกรม ใช้อธิบายการทำงานอย่างเป็นขั้นตอนตามลำดับขั้นของส่วนต่างๆในระบบ รวมถึงแสดงการควบคุมระบบทั้งหมดอีกด้วย



รูป ค.8 Activity Diagram

ค.2.11 UML state machine diagram

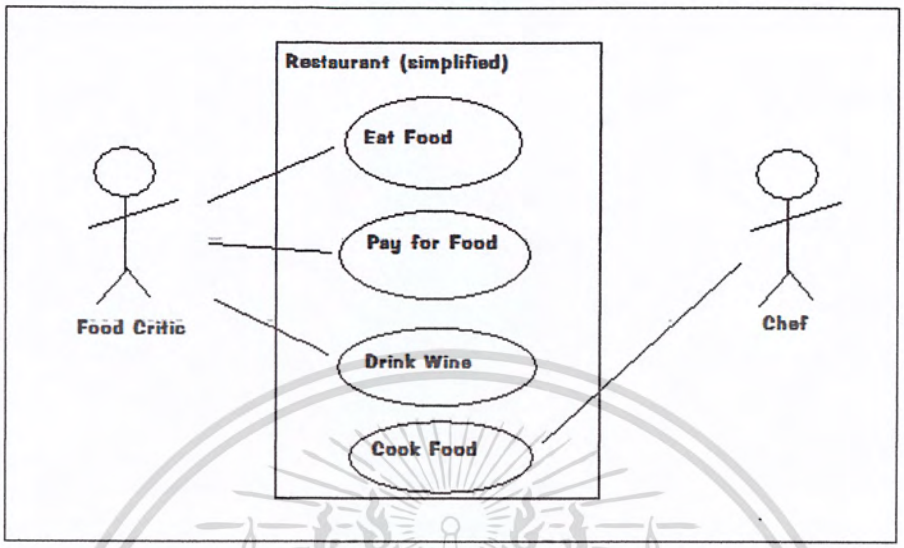
แผนภาพเสตจแมชชีน แสดงถึงสภาพและการแปลสภาพของระบบ



รูป ค.9 State Machine Diagram

ค.2.12 Use case diagram

เป็นแผนภาพที่ใช้ที่แสดงปฏิสัมพันธ์ระหว่างระบบงานและสิ่งที่อยู่นอกระบบงาน



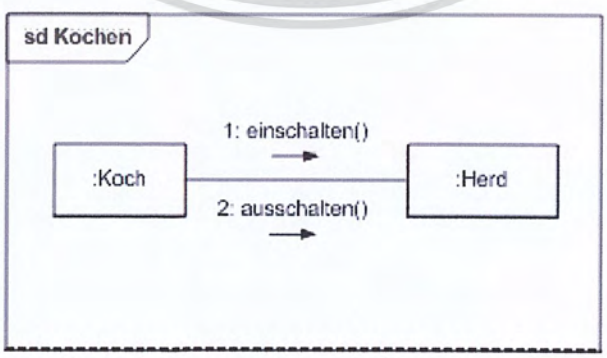
รูป ค.10 Use-Cases Diagram

ค.2.13 แผนภาพประเภทการโต้ตอบ (Interaction Diagrams)

แผนภาพประเภทการโต้ตอบ เป็นแผนภาพที่ใช้ที่แสดงขั้นตอนการทำงานของ ยูสเคส (use case) เช่นเดียวกับ Sequence Diagram และ Collaboration Diagram แต่จะเน้นไปที่งานย่อยของวัตถุ โดยจะมีกระบวนการทำงานคล้ายกับ Flowchart

ค.2.14 Communication diagram

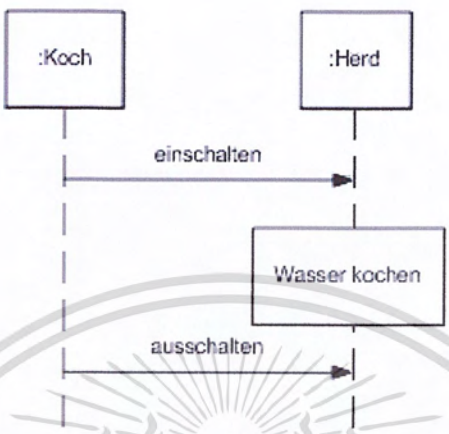
แสดงถึงปฏิสัมพันธ์ระหว่างวัตถุหรือระหว่างภาคส่วนใสรูปของแมสเสซอย่างเป็นลำดับ โดยแสดงการทำงานร่วมกันของคลาส ซีเคว้น และยูสเคส



รูป ค.11 Communication Diagram

ค.2.15 Sequence diagram

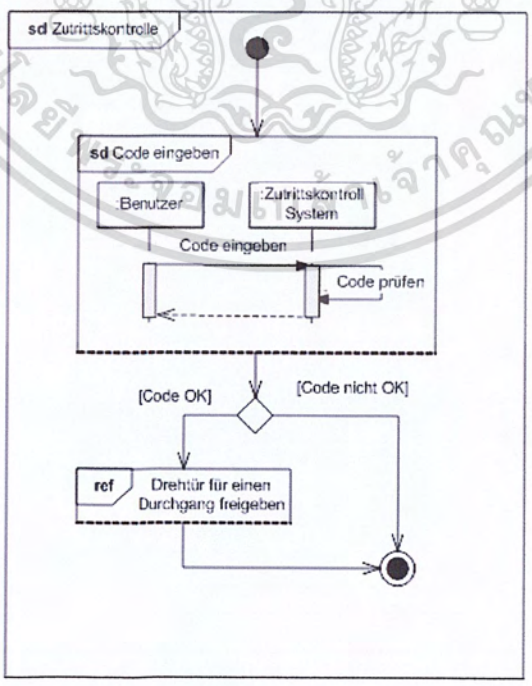
แสดงถึงการติดต่อสื่อสารกันของวัตถุต่างๆ ควบคู่ไปกับแสดงการมีอยู่ของแต่ละวัตถุ



รูป ค.12 Sequence Diagram

ค.2.16 Timing diagrams

เป็นชนิดพิเศษของอินเตอร์แอคชั่น ไดอะแกรม โดยจะโฟกัสเฉพาะเจาะจงในบางช่วงเวลา



รูป ค.13 Timing Diagrams