เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัว

Digital Real-time Watt Hour Meter for Building based on Raspberry Pi

โดย

นายอาสาฬห์ เขียวเม่น

นายเอกชยรัฐ ทองสาย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัว
Digital Real-time Watt Hour Meter for Building based on Raspberry Pi

โดย

นายอาสาฬห์ เขียวเม่น 54011555

นายเอกชยรัฐ ทองสาย 54011573

อาจารย์ที่ปรึกษา

รศ.ดร. พิพัฒน์ พรหมมี

ผศ.ดร. มนตรี คำเงิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2557

ปริญญานิพนธ์ปีการศึกษา 2557
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัว

Digital Real-time Watt Hour Meter for Building based on Raspberry Pi

ผู้จัดทำ

1. นายอาสาฬห์ เขียวเม่น 54011555
2. นายเอกชยรัฐ ทองสาย 54011573

..... อาจารย์ที่ปรึกษา

(รศ.ดร.พิพัฒน์ พรหมมี)

..... อาจารย์ที่ปรึกษา

(ผศ.ดร.มนตรี คำเงิน)

กิตติกรรมประกาศ

รายงานประกอบปริญญานิพนธ์เล่มนี้สำเร็จได้ลุล่วงไปได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์จาก รศ.ดร.พิพัฒน์ พรหมมี และ ผศ.ดร.มนตรี คำเงิน อาจารย์ปรึกษาโครงงานที่ได้ให้คำแนะนำที่ดีตลอดมา และเอื้อเฟื้อสถานที่ให้ในจัดทำโครงงาน ขอบคุณนิตยสารซีเอ็ด เอ็ดดูเคชัน สำหรับคำแนะนำเกี่ยวกับอุปกรณ์วัดกำลังไฟฟ้า ขอบคุณรุ่นพี่และเพื่อนๆ ทุกคนที่ให้คำปรึกษาและคอยให้ความช่วยเหลือ และที่สำคัญที่สุดขอบคุณเพื่อนในกลุ่มที่ได้จัดทำปริญญานิพนธ์ ที่ทำโครงงานนี้สำเร็จลุล่วงไปด้วยดี ท้ายนี้ผู้จัดทำขอขอบคุณทุกๆ ท่านที่มีส่วนเกี่ยวข้องมา ณ โอกาสนี้

นายอาสาฬห์ เขียวเม่น
นายเอกชยรัฐ ทองสาย
ผู้จัดทำ

เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัว
DIGITAL REAL-TIME WATT HOUR METER FOR BUILDING BASED ON RASPBERRY PI

โดย นายอาสาฬห์ เขียวเม่น 54011555
นายเอกชยรัฐ ทองสาย 54011573

อาจารย์ที่ปรึกษา รศ.ดร.พิพัฒน์ พรหมมี
ผศ.ดร.มนตรี คำเงิน

บทคัดย่อ

โครงงานนี้มีจุดมุ่งหมายเพื่อที่จะศึกษา และออกแบบเครื่องมือ เพื่อใช้ในการวัดพลังงานไฟฟ้าที่เรียกว่า เครื่องมือการวัดพลังงานไฟฟ้าภายในบ้าน ที่สามารถวัด กระแส แรงดัน และกำลังไฟฟ้าบางอย่าง เครื่องวัดกำลังไฟฟ้านี้ จะอำนวยความสะดวกสบายของผู้ใช้ในการเฝ้าดูค่าพารามิเตอร์ของพลังงานไฟฟ้า ได้หลากหลายวิธีการ เช่น โปรแกรมประยุกต์บนเว็บอินเทอร์เน็ต หรือหน้าจอแสดงผลของมิเตอร์ โดยเลือกใช้ Power IC chip เบอร์ ADE7763 เป็นตัววัด และคำนวณค่าทางไฟฟ้าต่างๆ อาทิเช่น กระแส แรงดัน และกำลังไฟฟ้าต่างๆ การวัดพลังงานไฟฟ้านี้จะเหมาะสมกับเครื่องใช้ไฟฟ้าตามบ้านเรือน นอกจากนี้ยังสามารถวัดค่ากำลังไฟฟ้าสะสมได้หรือ kWh โดยมีสมองกลฝังตัวเป็นตัวควบคุม และแสดงผลต่างๆ

ABSTRACT

A digital watt-hour meter based on embedded system is presented in this project. It is able to measure volts, ampere and apparent power in the resident. The ADE7763 power IC chip module is used for sensing the volts, ampere and power of ac-line system. The cumulative power is calculated by using embedded system in kWh. The results can be displayed by using the web browser on application software.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

สารบัญรูป(ต่อ)

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ผู้ให้บริการหอพักส่วนใหญ่จะมีปัญหาในเรื่องการจดมิเตอร์ค่าไฟตามห้องต่างๆที่อยู่ภายในหอพักของ ตนเอง เมื่อถึงเวลาสิ้นเดือน ผู้ให้บริการจะต้องทำการจดมิเตอรของห้องพักต่างๆ เพื่อที่จะนำไปคำนวณค่าไฟของ หองพักนั้นๆ ซึ่งขั้นตอนเหล่านี้อาจเกิดความผิดพลาดได้ทุกขั้นตอน ตั้งแต่การจดค่ามิเตอร และ การคำนวณ ซึ่ง โครงงานนี้จะมีรูปแบบการวัดค่าไฟเป็นแบบดิจิตอลและทำการเก็บค่าเป็นปัจจุบันตลอดเวลา (Real Time) ซึ่งทำให้เกิดความผิดพลาดน้อย และหากเกิดปัญหาของเครื่องใช้ไฟฟ้าภายในห้องพักที่อาจใช้ไฟเกินกำหนดกว่าที่ควรจะ เป็นก็จะสามารถตรวจพบได้โดยง่าย

1.2 วัตถุประสงค์

1) เพื่อศึกษาการใช้งานระบบสมองกลฝังตัว
2) เพื่อศึกษาการเชื่อมต่อระหว่างระบบสมองกลฝังตัวกับระบบเซิร์ฟเวอร์
3) เพื่อศึกษาการเชื่อมต่อระหว่างวัตต์มิเตอร์กับระบบสมองกลฝังตัว
4) สร้างวัตต์มิเตอร์เพื่อหาค่ากำลังไฟฟ้าได้

1.3 ขอบเขตของปริญญานิพนธ์

โครงงานนี้จัดทำเพื่อศึกษาและเครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัว โดยมีการเชื่อมต่อระหว่างสมองกลฝังตัวกับวงจรคำนวณค่ากำลังไฟฟ้าและหน่วยไฟฟ้าที่ใช้ไปผ่านโครงข่ายอินเตอร์เน็ต และแสดงผลผ่านทางคอมพิวเตอร์ และทำการศึกษา ดิจิตอลวัตต์มิเตอร์ โดยแสดงผลทาง Raspberry Pi จากนั้นวัดผลการทดลองเทียบกับอุปกรณ์ที่ใช้งานจริง

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัว (Digital Real-time Watt Hour Meter for building based on Raspberry Pi) เป็นการนำเอา Raspberry Pi มาประยุกต์เป็นดิจิตอลวัตต์มิเตอร์เพื่อคำนวณหาค่ากำลังไฟฟ้าและ Raspberry Pi จะส่งข้อมูลไปยังเซิร์ฟเวอร์เก็บข้อมูลต่อไป

2.1 สมองกลฝังตัว

ระบบประมวลผล ที่ใช้ชิปหรือไมโครโพรเซสเซอร์ที่ออกแบบมาโดยเฉพาะ เป็นระบบคอมพิวเตอร์ขนาดจิ๋วที่ฝังไว้ในอุปกรณ์ เครื่องใช้ไฟฟ้า และเครื่องเล่นอิเล็กทรอนิกส์ต่างๆ เพื่อเพิ่มความฉลาด ความสามารถให้กับอุปกรณ์เหล่านั้นผ่านซอฟต์แวร์ซึ่งต่างจากระบบประมวลผลที่เครื่องคอมพิวเตอร์ทั่วไป ระบบฝังตัวถูกนำมาใช้กันอย่างแพร่หลายในยานพาหนะ เครื่องใช้ไฟฟ้าในบ้านและสำนักงาน อุปกรณ์อิเล็กทรอนิกส์ เทคโนโลยีซอฟต์แวร์ เทคโนโลยีฮาร์ดแวร์ เทคโนโลยีเครือข่ายเน็ตเวิร์ก เทคโนโลยีด้านการสื่อสาร เทคโนโลยีเครื่องกลและของเล่นต่าง ๆ คำว่าระบบฝังตัวเกิดจาก การที่ระบบนี้เป็นระบบประมวลผลเช่นเดียวกับระบบคอมพิวเตอร์ แต่ว่าระบบนี้จะฝังตัวลงในอุปกรณ์อื่น ๆ ที่ไม่ใช่เครื่องคอมพิวเตอร์ ในปัจจุบันระบบสมองกลฝังตัวได้มีการพัฒนามากขึ้น โดยในระบบสมองกลฝังตัวอาจจะประกอบไปด้วยไมโครคอนโทรลเลอร์ หรือไมโครโปรเซสเซอร์ อุปกรณ์ที่ใช้ระบบสมองกลฝังตัวที่เห็นได้ชัดเช่นโทรศัพท์มือถือ และในระบบสมองกลฝังตัวยังมีการใส่ระบบปฏิบัติการต่างๆแตกต่างกันไปอีกด้วย ดังนั้น ระบบสมองกลฝังตัวอาจจะทำงานได้ตั้งแต่ควบคุมหลอดไฟจนไปถึงใช้ในยานอวกาศ

2.1.1 ระบบปฏิบัติการสำหรับสมองกลฝังตัว

ในการพัฒนาระบบสมองกลฝังตัวอาจจะมีการใช้ระบบปฏิบัติการเป็นแกนหลักในการพัฒนา หรือไม่มีการใช้ในการพัฒนาก็ได้ ระบบปฏิบัติการสำหรับระบบสมองกลฝังตัวมีหลายประเภทมากตั้งแต่ RTOS, ucOS-II จนไปถึงระบบปฏิบัติการที่มีขนาดใหญ่ขึ้นมาเช่น Linux Window CE จนถึงระบบปฏิบัติการสมัยใหม่ที่มีการพัฒนา เช่น MeeGo Android ,Raspberry Pi

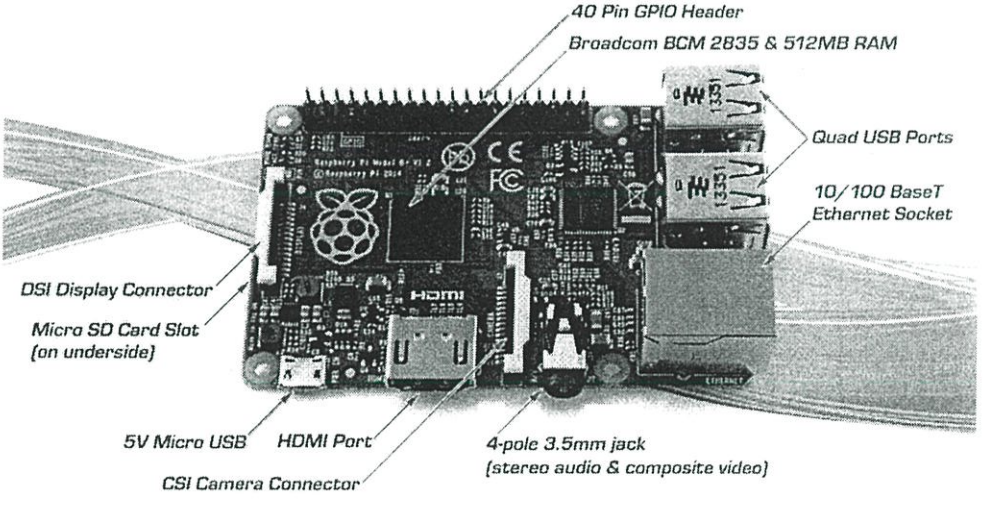
2.1.2 ภาษาที่ใช้ในการพัฒนาของระบบสมองกลฝังตัว

ในปัจจุบันมีภาษาโปรแกรมต่างๆมากมายที่ใช้ในการพัฒนาระบบสมองกลฝังตัวเช่น ภาษา assembly ภาษา C ,C++ หรือภาษาระดับสูงที่ถูกนำมาใช้ในการพัฒนาระบบสมองกลฝังตัว

ที่มีระบบปฏิบัติการเช่น JAVA หรือ Python โดยผู้ใช้สามารถเลือกใช้ภาษาในการพัฒนาระบบสมองกลฝังตัวได้ตามความเหมาะสม

2.1.3 Raspberry Pi

Raspberry Pi ก็คือคอมพิวเตอร์ขนาดจิ๋ว มีแต่สิ่งที่จำเป็นเพื่อการประมวลผล บอร์ดใช้ชิป SoC ของ Broadcom BCM2835 ซึ่งบรรจุ ARM1176JZFS พร้อมทั้งหน่วยประมวลผลเลขทศนิยม (floating point) ทำงานที่ความถี่สัญญาณนาฬิกา 700Mhz DRAM ขนาด 512 MB ซึ่งถ้าเป็น microcontroller ทั่วๆ ไป อาจมี RAM น้อยกว่านี้ (เป็น SRAM เพราะไม่มี MMU) งานบางอย่างอาจต้องใช้หน่วยความจำเยอะ เช่น งานประมวลผลภาพ Raspberry Pi จึงมีความเหมาะสม Raspberry Pi อาจมองว่ามันดูคล้าย tablet หรือ netbook ที่ไม่มีจอ ไม่มีแบตเตอรี่ ไม่มีคีย์บอร์ดหรือจอสัมผัส ก็ได้ ถ้าอยากได้ต้องหามาต่อเอง และถ้าต่อ Raspberry Pi กับ Monitor และ keyboard ก็สามารถใช้มันได้พอๆ กับ PC เหมือนกัน นอกจากนี้ยังสามารถต่ออุปกรณ์อื่นๆ ตามใช้ชอบได้เหมือนกัน และยังมี GPIO ที่รับส่งข้อมูลได้สารพัดนึกเหมือนกับพวก Microcontroller จึงอาจไปประยุกต์ใช้ในการควบคุมอย่างอื่นได้



รูปที่ 2.1 โครงสร้างของ Raspberry pi [1]

2.1.3.1 พอร์ต GPIO

เป็นพอร์ตอินพุตเอาต์พุตอเนกประสงค์ หรือ GPIO (General Purpose Input Output) ไว้ให้ใช้งานรวมทั้งหมด 21 ขา โดยมีขาสำหรับพอร์ตอินพุทและเอาต์พุทดิจิตอลปกติ ขา

เชื่อมต่อระบบบัส I2C และ SPI จึงทำให้ Raspberry Pi สามารถเชื่อมต่ออุปกรณ์อิเล็กทรอนิกส์ได้หลากหลาย

Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I2C)		DC Power 5v	04
05	GPIO03 (SCL1 , I2C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I2C ID EEPROM)		(I2C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

รูปที่ 2.2 พอร์ต GPIO ของ Raspberry Pi Model B+ [2]

2.1.3.2 พอร์ต SPI ของบอร์ด Raspberry Pi

ในการติดต่อสื่อสารกับ ADE7763 จำเป็นจะต้องใช้งาน Port SPI โดย SPI (Serial Peripheral Interface) Bus เป็นรูปแบบหนึ่งสำหรับการสื่อสารข้อมูลแบบดิจิตอลระหว่างอุปกรณ์ เป็นการรับส่งข้อมูลแบบ รับ-ส่งข้อมูลทีละบิตและใช้สัญญาณ Clock เป็นตัวกำหนดจังหวะการทำงาน (Synchronous, Bit-Serial Data Communication) ที่มีการใช้งานอย่างแพร่หลาย บอร์ด RPi Model B, rev 2.0 รองรับการใช้งานพอร์ต SPI โดยมีขาสำหรับพอร์ต SPI อยู่ที่ P1 header ได้แก่ SPI_MOSI (Master-Out, Slave-In), SPI_MISO (Master-In, Slave-Out), SPI_SCLK (Serial Clock)

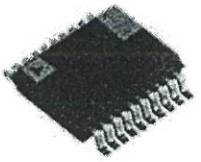
บอร์ด Raspberry Pi จะทำหน้าที่เป็นฝ่ายที่เรียกว่า SPI Master เพื่อเริ่มต้นและสื่อสารข้อมูลกับอุปกรณ์อื่นที่ทำหน้าที่เป็นฝ่าย SPI Slave และใช้ขาสัญญาณ Chip Select (หรือ

Slave Select) เพื่อเลือกสื่อสารกับอุปกรณ์ SPI Slave ได้โดยอัตโนมัติ (สัญญาณนี้ทำงานแบบ Active-Low และจะเป็น LOW ในช่วงของการส่ง-รับข้อมูลกับ SPI Slave ที่ได้เลือก) บอร์ด RPI มีขาสัญญาณแบบนี้อยู่ 2 ขา คือ SPI_CE0_N และ SPI_CE1_N สามารถใช้ต่อกับอุปกรณ์ SPI Slave ในบัสเดียวกันได้สองอุปกรณ์ (หรือได้มากกว่า ถ้าใช้ขา GPIO ด้วยเพื่อใช้เป็นสัญญาณเลือกอุปกรณ์ SPI Slave แต่ต้องคอยเปลี่ยนขาสัญญาณจาก HIGH เป็น LOW ก่อนการส่ง-รับข้อมูลใดๆ และกลับไปเป็น HIGH ในสภาวะ Idle)

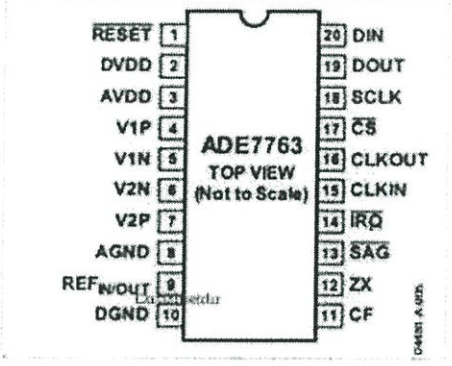
รูปที่ 2.3 แสดงพอร์ต SPI ของ Raspberry Pi [2]

2.2 IC ADE7763

ADE7763 เป็น Power IC chip ของบริษัท Analog Devices ซึ่ง IC ตัวนี้มีความสามารถในการวัดคำนวณค่าทางไฟฟ้าออกมาได้หลากหลายและมีความแม่นยำสูง มีความสามารถในการวัดแรงดัน และ กระแส รวมไปถึงค่ากำลังไฟฟ้าต่างๆ เช่น กำลังไฟฟ้าจริง (Active Power) และกำลังไฟฟ้าปรากฏ (Apparent Power) โดย Output ที่ได้นั้นจะเป็นค่าดิจิตอล ADE7763 มีส่วนที่ใช้ติดต่อกับอุปกรณ์ภายนอกแบบอนุกรม (SPI: Serial Peripheral Interface) เพื่อใช้ในการติดต่อระหว่าง Power IC กับไมโครคอนโทรลเลอร์ได้อีกด้วย โดยในส่วนของ Input จะมีสองทาง คือ ทางที่หนึ่งเป็น Input สำหรับวัดค่ากระแสไฟฟ้า (Current Channel) และอีกทางคือ Input ของแรงดันไฟฟ้า (Voltage Channel) ซึ่งทั้งสอง Channel นี้รับค่าแรงดันไฟฟ้าได้มากที่สุดไม่เกิน ±0.5 Vrms หลังจากนั้น สัญญาณจะผ่านเข้ามาที่ตัวขยายสัญญาณ (Amplifier) ต่อมา สัญญาณที่วัดได้ก็จะผ่านตัวแปลงสัญญาณจากอนาล็อกเป็นดิจิตอล จากนั้นผ่านเข้าสู่ตัวกรองความถี่สูง (High Pass Filter) เพื่อตัดสัญญาณรบกวนต่างๆ ออกและเมื่อสัญญาณผ่านวงจรอินทิเกรเตอร์ (Integrator) ทั้งสอง Channel สัญญาณจะแบ่งออกเป็นสองส่วน โดยส่วนที่หนึ่งจะนา ไปคูณกับแรงดันไฟฟ้า ผลที่ได้จะเป็นกำลังไฟฟ้าจริง (Active Power) มีหน่วยเป็นวัตต์ (Watt) และนา ค่าที่ได้เก็บไว้ในรีจิสเตอร์ซึ่งจะสะสมค่าเพิ่มขึ้นเรื่อยๆ ดังนั้นค่าในรีจิสเตอร์จึงเป็นค่าพลังงานไฟฟ้าจริง (Active Energy) ส่วนสัญญาณที่สองนั้นจะนำมาคำนวณ เป็นค่ารากของกำลังสองเฉลี่ย (RMS) ก่อนที่จะนำมาคูณกับแรงดันไฟฟ้า (RMS) เพื่อให้ได้กำลังไฟฟ้าปรากฏ (Apparent Power) มีหน่วยเป็น VA (Volt-Amp) และนำค่าที่ได้เก็บไว้ในรีจิสเตอร์ซึ่งจะสะสมค่าเพิ่มขึ้นเรื่อยๆ ดังนั้นค่าในรีจิสเตอร์จึงเป็นค่าพลังงานไฟฟ้าปรากฏ (Apparent Energy)



รูปที่ 2.4 Power IC chip (ADE7763) [3]



รูปที่ 2.5 โครงสร้างภายในของ IC ADE7763 {3}

2.3 การหาค่ากำลังไฟฟ้า

ถ้ากำหนดให้ค่าแรงดันไฟฟ้าตกคร่อมโหลดและกระแสไฟฟ้าไหลผ่านโหลด ณ เวลาใดๆ มีค่าเป็น

$$v(t) = V_p \cos(\omega_0 t)$$

$$i(t) = I_p \cos(\omega_0 t - \theta)$$

โดยที่ V_p คือ แอมปลิจูดของแรงดันไฟฟ้า

I_p คือ แอมปลิจูดของกระแสไฟฟ้า

ω_0 คือ ความเร็วเชิงมุมซึ่งมีค่าเท่ากับ $2\pi f_0$

f_0 คือ ความถี่มูลฐาน

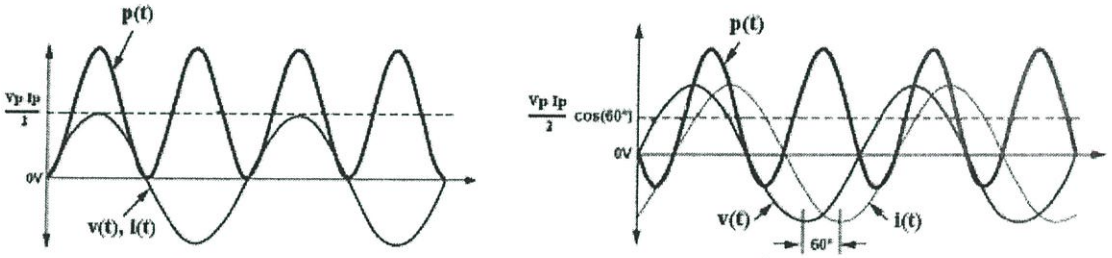
θ คือ ความต่างเฟสระหว่างแรงดันไฟฟ้าและกระแสไฟฟ้า

กำลังไฟฟ้าชั่วขณะ (instantaneous power) จะมีค่าเท่ากับ

$$p(t) = v(t)i(t) = V_p I_p \cos(\omega_0 t)\cos(\omega_0 t - \theta)$$
$$= \frac{V_p I_p}{2}[\cos\theta + \cos(2\omega_0 t - \theta)]$$

เมื่อนำ $p(t)$ ไปหาค่าเฉลี่ยในหนึ่งลูกคลื่น จะได้กำลังไฟฟ้าเฉลี่ย คือ

$$p_{av} = \frac{V_p I_p}{2}\cos\theta$$ เมื่อ $\theta = 0°$ และ $60°$ กำลังไฟฟ้าจะมีค่าเท่ากับ $\frac{V_p I_p}{2}$ และ $\frac{V_p I_p}{2}\cos 60°$ ตามลำดับดังเส้นประที่แสดงในรูปที่ 1 ก) และ 2 ข)



ก) เมื่อ $\theta = 0°$ ข) เมื่อ $\theta = 60°$

รูปที่ 2.6 แรงดันไฟฟ้า กระแสไฟฟ้า และกำลังไฟฟ้าชั่วขณะ [4]

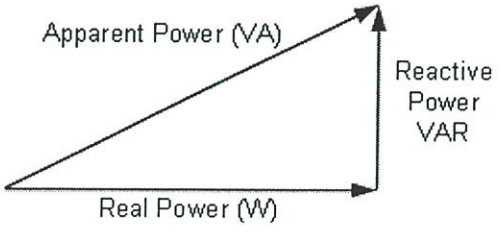
2.4 ทฤษฎีกำลังไฟฟ้า

กำลังไฟฟ้า (Electrical Power) คือ การนำเอากระแสมาคูณกับแรงดัน อาจจะมีตัวประกอบกำลังมาคูณเพิ่ม ดังนั้น กำลังไฟฟ้าประกอบไปด้วย สามส่วนหลักๆ คือ

1.ค่ากำลังไฟฟ้าที่ปรากฏ (Apparent Power) คือ ค่าที่กระแสคูณ (Cross Product) กับแรงดัน

2.ค่ากำลังไฟฟ้าจริง (Real Power) คือ ค่าที่กระแสคูณ (Cross Product) กับแรงดัน แล้วคูณด้วย Cosine ของมุมระหว่างแรงดันกับกระแสนั้นๆ

3.กำลังไฟฟ้าสุดท้าย คือ ค่ากำลังไฟฟ้าประกอบ (Reactive Power) คือ ค่าที่กระแสคูณ (Cross Product) กับแรงดัน แล้วคูณด้วย Sine ของมุมระหว่างแรงดันกับกระแสนั้นๆ ซึ่งเรียก Cosine ของมุมระหว่างแรงดันกับกระแสว่า Power factor



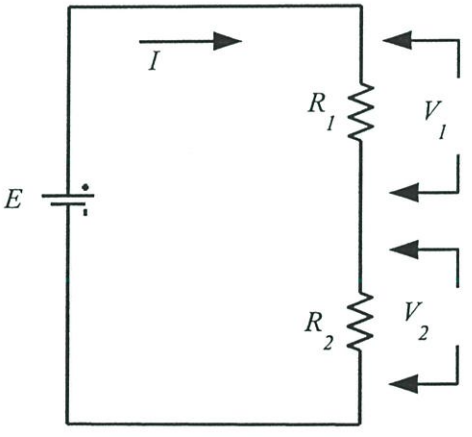
รูปที่ 2.7 กราฟแสดง power triangle

2.5 วงจรแบ่งแรงดันไฟฟ้า

วงจรแบ่งแรงดันไฟฟ้า (Voltage Divider) เป็นวงจรที่ทำหน้าที่แบ่งแรงดันไฟฟ้าออกเป็นระดับต่างๆตามความต้องการ วงจรมีลักษณะเป็นวงจรแบบอนุกรมสามารถแบ่งออกเป็น 2 ชนิดด้วยกันคือวงจร แบ่งแรงดันไฟฟ้าแบบไม่มีภาระไฟฟ้า (Unloaded Voltage Divider) วงจรแบ่งแรงดันไฟฟ้าแบบมีภาระไฟฟ้า (Loaded Voltage Divider)

2.5.1 วงจรแบ่งแรงดันไฟฟ้าแบบไม่มีภาระไฟฟ้า

วงจรแบ่งแรงดันไฟฟ้าแบบไม่มีภาระไฟฟ้า (Unloaded Voltage Divider) เป็นวงจรแบ่งแรงดันไฟฟ้าที่ยังไม่ได้ต่อภาระไฟฟ้าเราสามารถที่จะออกแบบการแบ่งแรงดันไฟฟ้าได้ตามความต้องการใช้งานในการคำนวณจึงไม่ต้องนำค่าภาระไฟฟ้ามาคำนวณด้วย



รูปที่ 2.8 วงจรแบ่งแรงดันไฟฟ้าแบบไม่มีภาระไฟฟ้า

จากรูป R_1 และ R_2 ทำหน้าที่แบ่งแรงดันไฟฟ้าออกเป็น 2 ช่วงคือ V_1 และ V_2 การคำนวณหาค่า V_1 และ V_2 ถ้าใช้สูตรการคำนวณแบบวงจรอนุกรมจำเป็นต้องคำนวณหาค่าความต้านทานรวม (R_T) และกระแสไฟฟ้า (I_T) ของวงจรเสียก่อน ซึ่งทำให้เสียเวลา เราสามารถประยุกต์สูตรที่ใช้ในการคำนวณหาค่าแรงดันไฟฟ้าตกคร่อมที่ตัวต้านทานมาใช้ในการคำนวณหาค่า V_1 และ V_2 ได้โดยไม่ใช้ค่าของกระแสไฟฟ้าของวงจรได้ดังนี้

จาก $V_1 = IR_1$ เมื่อ $I = \frac{E}{R_T}$

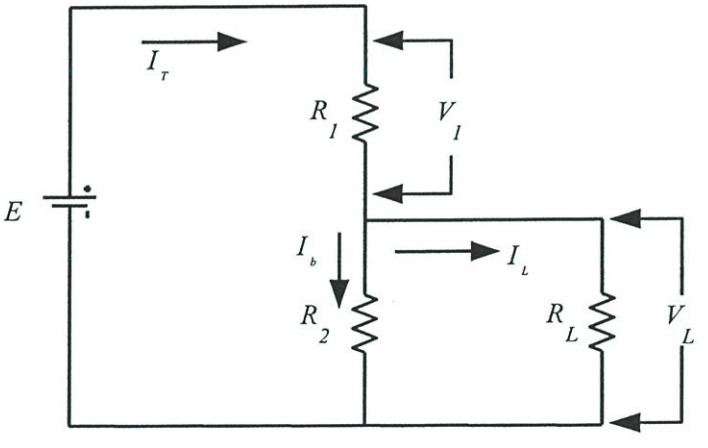
แทนค่า $I = \frac{E}{R_T}$ ในสมการจะได้ $V_1 = \frac{E}{R_T} R_1$ แต่ $R_T = R_1 + R_2$

แทนค่า $R_T = R_1 + R_2$ ในสมการจะได้ $V_1 = E \frac{R_1}{R_1 + R_2}$

ทำนองเดียวกัน $V_2 = E \frac{R_2}{R_1 + R_2}$

2.5.2 วงจรแบ่งแรงดันไฟฟ้าแบบมีภาระไฟฟ้า

วงจรแบ่งแรงดันไฟฟ้าแบบมีภาระไฟฟ้า (Loaded Voltage Divider) จะคล้ายกับวงจรแบ่งแรงดันไฟฟ้าแบบไม่มีภาระไฟฟ้าเพียงแต่มีการต่อภาระไฟฟ้าเข้าไปในส่วนที่แบ่งแรงดันไฟฟ้าไว้ ดังนั้นการคำนวณหาค่าแรงดันไฟฟ้าในวงจรจึงต้องคำนึงถึงค่าความต้านทานของภาระไฟฟ้าที่นำไปต่อด้วย โดยการให้กระแสไฟฟ้าไหลผ่านตัวต้านทานที่ต่อขนานกับภาระไฟฟ้านั้นมีค่าประมาณ 10% – 20 % ของกระแสไฟฟ้าที่ไหลผ่านภาระไฟฟ้าทั้งหมด



รูปที่ 2.9 วงจรแบ่งแรงดันไฟฟ้าแบบมีภาระไฟฟ้า

จากวงจรยุบรวม R_1 และ R_L ได้ดังสูตร

$$R_{T_1} = \frac{R_2 \times R_L}{R_2 + R_L}$$

คำนวณหาค่า R_T ได้จากสูตร

$$R_T = R_1 + R_{T_1}$$

จากสูตรการคำนวณหาค่ากระแสไฟฟ้าในวงจร

$$I_T = \frac{E}{R_T} = \frac{E}{R_1 + R_{T_1}} \quad \text{และ}$$

$$V_L = I_T R_{T_1}$$

แทนค่า I_T ในสูตร $V_L = I_T R_{T_1}$ จะได้

$$V_L = \frac{E}{R_1 + R_{T_1}} \times R_{T_1} \quad \text{หรือ}$$

$$V_L = E \times \frac{R_{T_1}}{R_T} \quad \text{เมื่อ } R_T = R_1 + R_{T_1}$$

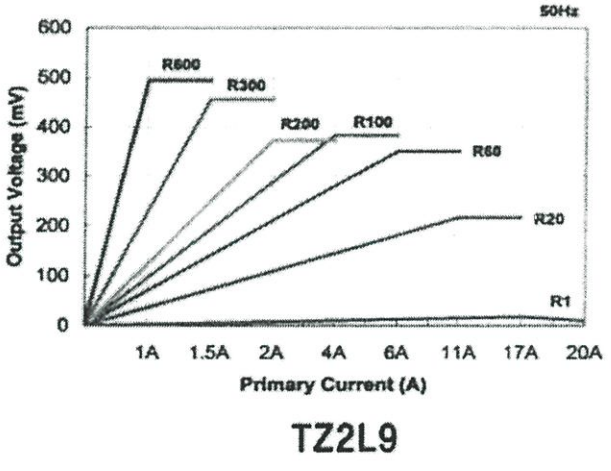
ดังนั้นจึงสามารถใช้สูตร $V_L = E \times \frac{R_{T_1}}{R_T}$ คำนวณหาค่าแรงดันไฟฟ้าของภาระไฟฟ้าได

2.6 Current Transformer

TZ2L9 เป็น อุปกรณ์เหนี่ยวนำกระแสชนิดหนึ่ง โดยใช้หลักการของสนามแม่เหล็กเพื่อลดทอนกระแสทางด้าน Primary ที่มีปริมาณสูง มาเป็นกระแสต่ำๆ (0-1A หรือ 0-5A) เพื่อให้เครื่องมือวัด หรือ อุปกรณ์ป้องกันทำงานได้ เนื่องจากใช้หลักการเหนี่ยวนำของสนามแม่เหล็ก CT จึงใช้กับไฟ AC ได้เท่านั้น โดยหม้อแปลง TZ2L9 ผลิตมาจากโรงานอัตราส่วน 1:1000 กล่าวคือเมื่อนำสายไฟจากโหลดคล้องผ่านอุปกรณ์ก็จะเกิดการเหนี่ยวนำกระแสในอัตราส่วน 1:1000 หรือให้กระแสที่เอาต์พุท 1 mA นั่นเอง

รูปที่ 2.10 TZ2L9 [4]

TZ2L9 สามารถวัดกระแสได้หลายย่านโดยเลือกย่านการวัดจากตัวต้านทานที่เอามาต่อขนานกับเอาท์พุทเรียกว่า Bunden Resistor ยิ่งค่าน้อยยิ่งดี ค่าผิดพลาดน้อย และสามารถบ่งบอกถึงย่านการวัดได้อีกด้วย โดยจากกราฟความสัมพันธ์ของกระแสและแรงดันด้านล่างจะแสดงให้ถึงเมื่อใช้กระแส 1A และใช้ตัวต้านทานที่ต่อขนานทางด้านเอาท์พุทเป็น 600 โอห์มจะให้แรงดันที่เอาท์พุทเป็น 500 mV ซึ่งอุปกรณ์ ADE7763 นั้น ด้านอินพุทจะรับแรงดันได้ไม่เกิน 0.5 Vrms จึงเลือกใช้ตัวต้านทาน 20 โอห์มเพื่อที่จะให้แรงดันทางด้านเอาท์พุทประมาณ 200 mV เพื่อป้องกันอุปกรณ์เสียหาย



รูปที่ 2.11 รูปแสดงความสัมพันธ์ของ Primary Current กับ Output Voltage ของ TZ2L9 [5]

2.7 LM 2575

เป็นวงจรแปลงโวลต์ให้ต่ำลง หรือ Step Down Voltage Regulator ที่ออกแบบให้มีขนาดเล็ก (1.4x2.4x1.1 cm.) ให้สามารถไปใช้แทนไอซี 7805 หรือ 7812 ได้โดยตรง โดยมีข้อดี คือ มีความร้อนที่เกิดขึ้นในวงจรน้อยมาก เพราะใช้หลักการ SWITCHER สามารถนำไปใช้ในวงจรต่างๆ ได้ดี ซึ่ง LM7805 นี้จะมีคุณสมบัติเหมือนกับ IC 7805 คือแปลงแรงดันจากอินพุทให้เหลือเพียง 5 V เท่านั้นเพื่อนำไปใช้กับอุปกรณ์ต่างๆ

รูปที่ 2.12 LM 2575 [6]

2.8 โปรแกรมจำลองการทำงาน PROTEUS

โปรแกรม Proteus เป็นโปรแกรมที่มีความสามารถมากอีกโปรแกรมหนึ่งในงานด้านอิเล็กทรอนิกส์ เพราะสามารถออกแบบวงจรไฟฟ้าได้ พร้อมทั้งสามารถจำลองการทำงานและออกแบบลายพิมพ์ของวงจรได้ความสามารถที่โดดเด่น Proteus นั้นจะกล่าวได้ว่าเป็นโปรแกรมที่สามารถจำลองพฤติกรรม (Simulator) การทำงานของวงจรที่ใช้ Microcontroller เบอร์ต่างๆได้มากมายโดยไม่ต้องประกอบวงจรให้เสียเวลา เพื่อพิสูจน์ว่าโปรแกรมที่เขียนขึ้นว่าใช้งานได้หรือไม่

ความเป็นมาของโปรแกรม Proteus หรือ Proteus VSN (Virtual System Modelling) เป็นโปรแกรมที่พัฒนาขึ้นโดยบริษัทแล็บเซ็นเตอร์อิเล็กทรอนิกส์จำกัด (Labcenter Electronics Ltd.) ที่ประเทศอังกฤษ โปรแกรม Proteus มีชื่อเต็มว่า Labcenter Electronics Proteus ซึ่งภายในโปรแกรมจะประกอบด้วยส่วนประกอบหลัก 2 ส่วนคือ ISIS และ ARES โปรแกรม Proteusจะมีอยู่หลายเวอร์ชันให้เลือกใช้งาน ซึ่งเวอร์ชันในปัจจุบัน คือ เวอร์ชัน 7.8 ความสามารถในการทำงานของโปรแกรม Proteus ก็คือ สามารถจำลองการทำงานของวงจรอิเล็กทรอนิกส์ได้หลายรูปแบบ ไม่ว่าจะเป็นแบบอนาล็อกและแบบดิจิตอล หรือทั้งแบบอนาล็อกและดิจิตอลผสมกัน นอกจากนี้ Proteus ยังสามารถออกแบบลายวงจรพิมพ์ (PCB) ได้อีกด้วย จุดเด่นของโปรแกรม Proteus ที่เป็นที่นิยมและชื่นชอบก็คือ การจำลองการทำงานของวงจรอิเล็กทรอนิกส์ ที่ใช้ไมโครคอนโทรลเลอร์ตระกูลต่างๆ ไม่ว่าจะเป็น PIC, MCS-51, AVR และ ARM เป็นต้น ทำให้นักเขียนโปรแกรมหรือโปรแกรมเมอร์สามารถตรวจสอบได้ว่าโปรแกรม หรือซอสโค้ด (Source Code) ที่เขียนขึ้นมานั้น สามารถสนับสนุนกับวงจรฮาร์ดแวร์ที่ต่อจำลองได้หรือไม่ ถ้าโปรแกรม (Source Code) ที่เขียนขึ้นไม่สนับสนุนกับวงจรที่ต่อจำลองโปรแกรมเมอร์ก็จะทำการพัฒนาโปรแกรม (Source Code) ที่เขียนขึ้นใหม่หรือปรับปรุงวงจรฮาร์ดแวร์ใน Proteus จนกว่าโปรแกรมที่เขียนขึ้นและฮาร์ดแวร์ที่ต่อจำลอง สามารถสนับสนุนซึ่งกันและกัน ทำให้การสร้างโครงงานต่าง ๆสามารถประหยัดเวลาและค่าใช้จ่ายเป็นอย่างมาก เพราะในอดีตการเขียนโปรแกรมขึ้นมานั้นจะต้องต่อวงจรจริงเพื่อทดสอบทำให้เสีย-เวลาและค่าใช้จ่ายมากในกรณีวงจรฮาร์ดแวร์และโปรแกรมที่เขียนขึ้นไม่สนับสนุนซึ่งกันและกัน

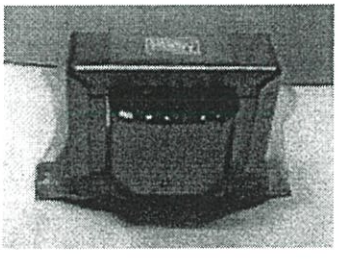
2.9 คริสตัลออสซิลเลเตอร์

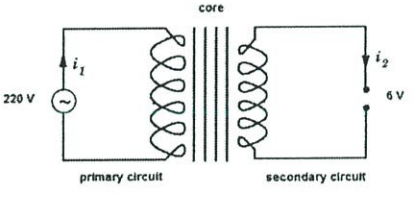
อุปกรณ์อิเล็กทรอนิกส์ประเภทนี้คือแหล่งกำเนิดสัญญาณนาฬิกาให้กับไมโครคอนโทรลเลอร์จึงจะทำให้ไมโครคอนโทรลเลอร์สามารถทำงานได้ ซึ่งแหล่งกำเนิดสัญญาณนาฬิกานี้อาจจะอยู่ภายในหรือภายนอกชิพไมโครคอนโทรลเลอร์ก็ได้ ถ้าเป็นแหล่งกำเนิดสัญญาณนาฬิกาภายในก็จำใช้สัญญาณจากออสซิลเลเตอร์ที่อยู่ภายใน ส่วนถ้าเป็นแหล่งกำเนิดสัญญาณนาฬิกาจากภายนอก ก็จะใช้ออสซิลเลเตอร์กับตัวเก็บประจุ (C) อีก 2 ตัวต่อเข้ากับขา XTAL1 และ XTAL2 ของไมโครคอนโทรลเลอร์

รูปที่ 2.13 คริสตัลออสซิลเลเตอร์ [8]

2.10 หม้อแปลง

หม้อแปลงเป็นอุปกรณ์ที่ทำหน้าที่ลดแรงดันจาก 220 VAC ไปเป็นแรงดันตามที่กำหนดไว้ เช่นหากหม้อแปลงมีตัวเลขด้านหนึ่งเขียนไว้ว่า 220V และอีกด้านหนึ่งเขียนไว้ว่า 6V แสดงว่าหม้อแปลงนี้ทำหน้าที่เปลี่ยนแรงดันจาก 220V มาเป็น 6V





รูปที่ 2.14 หม้อแปลง [9]

2.11 มัลติมิเตอร์

มัลติมิเตอร์ (Multimeter) ถือว่าเป็นเครื่องมือวัดที่จำเป็นสำหรับงานด้านอิเล็กทรอนิกส์ เพราะว่าเป็นเครื่องวัดที่ใช้ค่าพื้นฐานทางไฟฟ้าคือ แรงดันไฟฟ้า กระแสไฟฟ้าและความต้านทานไฟฟ้า ไม่ว่าจะเป็นการทดสอบหรือการตรวจซ่อมวงจรต่าง ๆ ก็จำเป็นต้องวัดค่าเหล่านั้นทั้งสิ้น มัลติมิเตอร์เป็นการรวม Voltmeter Ammeter และ Ohmmeter ไว้ในตัวเดียวกัน

รูปที่ 2.15 มัลติมิเตอร์ [10]

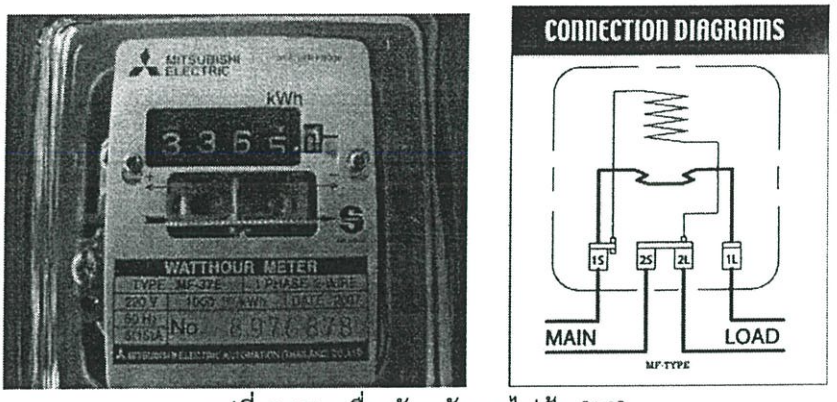
2.12 แคลมป์มิเตอร์

แคลมป์มิเตอร์ (Clamp meter) เป็นอุปกรณ์ที่ใช้ในการวัดค่ากระแสไฟฟ้า (current measurement) โดยที่เราไม่ต้องตัดต่อสายไฟแล้วทำต่อมิเตอร์ต่ออนุกรมเพื่อวัดค่ากระแสโดย clamp meter นี้จะมีส่วนคล้ายขากรรไกรเพื่อใช้สำหรับคล้องสายไฟและสามารถอ่านค่ากระแสไฟฟ้าได้เลยโดยไม่ต้องเสียเวลา

รูปที่ 2.16 แคลมป์มิเตอร์ [11]

2.13 เครื่องวัดพลังงานไฟฟ้า

เครื่องวัดพลังงานไฟฟ้า (KILOWATT HOUR METER) เป็นเครื่องวัดสำหรับวัดการใช้พลังงานไฟฟ้ามี หน่วยวัดเป็นกิโลวัตต์ชั่วโมง (kW-hrs) หรือยูนิต (unit) มักใช้เป็นเครื่องวัดหน่วยการใช้ไฟฟ้าในอาคาร บ้านเรือน หรือโรงงานอุตสาหกรรม ทั่วไป



รูปที่ 2.17 เครื่องวัดพลังงานไฟฟ้า [12]

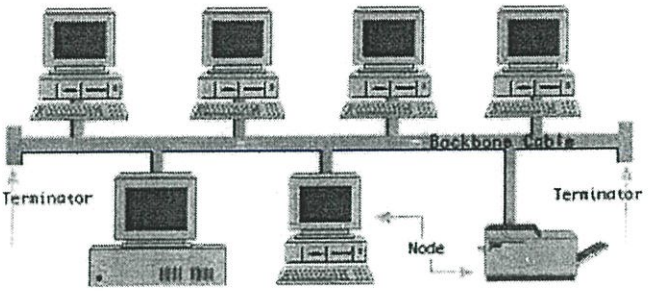
2.14 ระบบเครือข่ายอีเทอร์เน็ตแลน

ระบบเครือข่ายอีเทอร์เน็ตแลน (Ethernet LAN) เป็นเครือข่ายแบบมาตรฐานแบบแรกของโลกลักษณะการเชื่อมต่อเครือข่ายเป็นระบบแลนและมีลักษณะเป็นเส้นตรงโดยมีเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเครื่อง Server อย่างน้อย 1 เครื่อง อีเทอร์เน็ตเป็นเทคโนโลยีที่มีราคาถูกและใช้งานง่าย เมื่อต้องการเชื่อมต่อเครือข่ายอีเทอร์เน็ตก็ต้องมีการ์ดเชื่อมโยงอีเทอร์เน็ต หรือที่เรียกว่า NIC - Network Interface Card การ์ดเชื่อมโยงที่มีขายอยู่มักจะมีหัวต่อเชื่อมโยงทั้งที่เป็นแบบโคแอกเชียลและยูทีพีหรืออาจเรียกเป็นมาตรฐานกลางคือ 10BASE-2 หรือ 10BASE-T การใช้งานเครือข่ายจึงทำได้ง่ายเพียงหาสายโคแอกเชียลต่อเชื่อมและโปรแกรมไมโครซอฟต์วินโดวส์ก็ทำให้ใช้ไฟล์ร่วมกันหรือใช้เครื่องพิมพ์ร่วมกันได้ปัจจุบันมีมาตรฐาน 100BASE-T ซึ่งกำลังเป็นที่นิยมเพราะสามารถดูและและใช้งานได้ดี

2.14.1 โครงสร้างระบบเครือข่ายแลนแบบบัส

เครือข่ายแลนแบบบัส (Bus Network) มีโครงสร้างไม่ยุ่งยาก และไม่ต้องใช้เครื่องขยายสัญญาณ หรืออุปกรณ์สลับสาย สถานีต่าง ๆ จะเชื่อมต่อเข้าหาบัสโดยผ่านทางอุปกรณ์เชื่อมต่อที่เป็นฮาร์ดแวร์ การจัดส่งข้อมูลจึงสามารถส่งไปถึงทุกสถานีได้ ระบบนี้มี

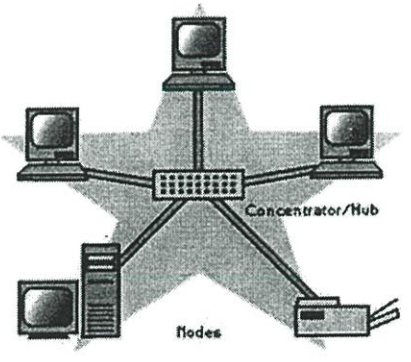
จุดอ่อนอยู่ที่ เมื่อคอมพิวเตอร์เครื่องใดเครื่องหนึ่งมีปัญหากับสายเคเบิ้ล จะทำให้ระบบรวนไปทั้งระบบ และเมื่อเพิ่มคอมพิวเตอร์ในเครือข่าย จะต้องหยุดการใช้งานทั้งหมด เพื่อตัดต่อสายใหม่ ข้อดีคือ ไม่จำเป็นต้องมีอุปกรณ์ตัวกลางในการเชื่อมโยง เช่น ฮับ (Hub) หรือสวิตช์ (Switch)



รูปที่ 2.18 โครงสร้างระบบเครือข่ายแลนแบบบัส [13]

2.14.2 โครงสร้างระบบเครือข่ายแลนแบบสตาร์

เครือข่ายแลนแบบสตาร์ (Star Network) จะมีลักษณะคล้ายกับดาวกระจาย มีอุปกรณ์ Hub เป็นศูนย์กลางการต่อเชื่อม โดยการนำสถานีต่าง ๆ มาต่อร่วมกันกับหน่วยสลับสายกลาง เพื่อเชื่อมโยงระหว่างสถานีต่างๆ ที่ต้องการติดต่อกัน ข้อดีของระบบนี้คือ เมื่อสายใดหลุดหรือขาดการต่อเชื่อม จะไม่มีผลต่อระบบทั้งหมด เป็นที่นิยมมากที่สุดในปัจจุบันนี้

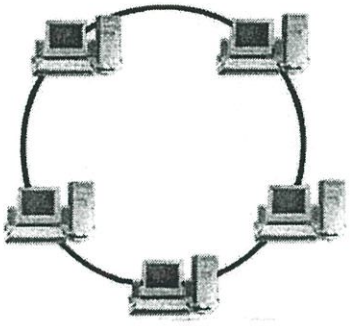


รูปที่ 2.19 โครงสร้างระบบเครือข่ายแลนแบบสตาร์ [13]

2.14.3 โครงสร้างระบบเครือข่ายแลนแบบวงแหวน (Ring Network)

เครือข่ายแลนแบบวงแหวน (Ring Network) สถานีของเครือข่ายทุกสถานีจะทำการเชื่อมโยงเครื่องขยายสัญญาณของทุกสถานีเข้าไว้ด้วยกันเป็นวงแหวน โดยด้านหนึ่งเป็นตัวรับสัญญาณ และอีกด้านหนึ่งเป็นตัวส่งสัญญาณ การส่งข้อมูลเป็นลักษณะออกทั้งสองทาง ถ้าทางไหนถึงก่อนเครื่องคอมพิวเตอร์ก็จะรับเพ็จเก็จนั้น มีความเร็วในการ รับส่งสัญญาณได้ 16 ล้านบิตต่อ

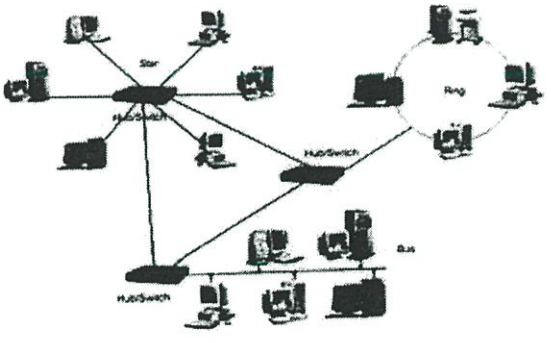
วินาที ข้อมูลจะไม่ชนกันเพราะการรับส่งมีลำดับที่แน่นอน ว่ามาจากสถานีใด จะส่งไปยังสถานีปลายทางที่ใด นิยมใช้ในเครือข่ายของเครื่องคอมพิวเตอร์ IBM โดยเฉพาะระบบธนาคาร ATM และระบบทางทหาร



รูปที่ 2.20 โครงสร้างระบบเครือข่ายแลนแบบวงแหวน [13]

2.14.4 โครงสร้างระบบเครือข่ายแลนแบบผสม (Hybrid Network)

เครือข่ายแลนแบบผสม (Hybrid Network) เป็นการเชื่อมเครือข่ายแบบดาว บัส และวงแหวนมาผสมผสานกัน เพื่อลดจุดอ่อนและเพิ่มจุดเด่นให้กับระบบเครือข่าย โดยการใช้ฮับหรือสวิตช์ เป็นอุปกรณ์ต่อเชื่อม



รูปที่ 2.21 โครงสร้างระบบเครือข่ายแลนแบบผสม [13]

2.15 Hub

Hub หรือบางทีก็เรียกว่า "รีพีตเตอร์ (Repeater)" คือ อุปกรณ์ที่ใช้เชื่อมต่อกลุ่มของคอมพิวเตอร์ Hub มีหน้าที่รับส่งเฟรมข้อมูลทุกเฟรมที่ได้รับจากพอร์ตใดพอร์ตหนึ่งไปยังทุก ๆ พอร์ตที่เหลือ คอมพิวเตอร์ที่เชื่อมต่อเข้ากับ Hub จะแชร์แบนด์วิธหรืออัตราข้อมูลของเครือข่าย ฉะนั้นยิ่งมีคอมพิวเตอร์เชื่อมต่อเข้ากับ Hub มากเท่าใด ยิ่งทำให้แบนด์วิธต่อคอมพิวเตอร์แต่ละ

เครื่องลดลง ในท้องตลาดปัจจุบันมี Hub หลายชนิดจากหลายบริษัท ข้อแตกต่างระหว่าง Hub เหล่านี้ก็เป็นจำพวกพอร์ต สายสัญญาณที่ใช้ ประเภทของเครือข่าย และอัตราข้อมูลที่ Hub รองรับได้

2.15.1 หลักการทำงานของ Hub

เมื่อใดที่มีคอมพิวเตอร์ภายในเครือข่ายต้องการส่งข้อมูล Hub ทำจะหน้าที่ในการทำสำเนาข้อมูลและส่งไปยังอุปกรณ์ต่างๆ ภายในเครือข่าย ไม่ใช่แค่คอมพิวเตอร์ แต่รวมถึงอุปกรณ์อื่นๆ ด้วยเช่น เครื่องพิมพ์ เป็นต้น เรียกว่าส่งข้อมูลไปทั้งหมด และถ้าข้อมูลนี้เป็นของอุปกรณ์ใด อุปกรณ์นั้นก็จะรับเองอัตโนมัติ และจุดด้อยของ Hub ที่ควรทราบคือ เวลามีอุปกรณ์ใดส่งข้อมูลในเครือข่ายผ่าน Hub อุปกรณ์อื่นๆ จะต้องรอให้การส่งสมบูรณ์ก่อน เปรียบเทียบได้กับถนน One-Way ห้ามส่งข้อมูลสวนทางกัน

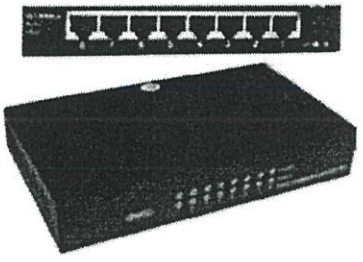
Hub นั้นทำงานในระดับ layer 1 ซึ่งเป็น layer เกี่ยวข้องกับ เรื่องของการส่งสัญญาณออกไปสู่ media หรือ สื่อกลางที่ใช้ในการสื่อสาร รวมไปถึงเรื่องของการเข้ารหัสสัญญาณเพื่อที่จะส่งออกไปเป็นค่าต่างๆในทางไฟฟ้า และ เป็น layer ที่กำหนดถึง การเชื่อมต่อต่างๆที่เป็นไปในทาง physical hub นั้น จะทำงานในลักษณะของการทวนสัญญาณ หมายถึงว่า จะทำการทำซ้ำสัญญาณนั้นอีกครั้งซึ่งเป็นคนละอย่างกับการขยายสัญญาณนะครับ พอทำแล้วก็จะส่งออกไปยังเครือข่ายที่เชื่อมต่ออยู่โดยจะมีหลักว่า จะส่งออกไปยังทุกๆ port ยกเว้น port ที่เป็นตัวส่งสัญญาณออกมาและเมื่อปลายทางแต่ละจุดรับข้อมูลไปแล้ว ก็จะต้องพิจารณา ข้อมูลที่ได้มา ว่าข้อมูลนั้นส่งมาถึงตัวเองหรือไม่ ถ้าหากไม่ใช่ข้อมูลที่จะส่งมาถึงตัวเอง ก็จะไม่รับข้อมูลที่ส่งมานั้น

การทำงานในระดับนี้ ถ้าดูในส่วนของตัว Hub เองนั้น จะเห็นได้ว่า ตัวของ Hub นั้นเวลาส่งข้อมูลออกไป จะไม่มีการพิจารณาข้อมูลอย่างพวก mac address ของ layer 2 หรือ ip address ซึ่งเป็นของ layer 3 เลย

2.15.2 ประเภทของ Hub

2.1.2.1. SmallHub มีจำนวนพอร์ต RJ-45ประมาณ 4,5,8,2 และ 16 พอร์ต แล้วแต่รุ่น เหมาะกับระบบเครือข่ายขนาดเล็กที่มีเครื่องคอมพิวเตอร์ไม่มากนัก ประมาณ 3-16 เครื่อง

2.15.2.2. Rack mount Hub ขนาดความกว้าง 19 นิ้ว พอร์ต RJ-45 มีมากตั้งแต่ 12, 16, 24 ถึง 48 พอร์ต เหมาะสำหรับใช้งานในเครือข่ายขนาดใหญ่ ที่มีเครื่องคอมพิวเตอร์ตั้งแต่ 12 เครื่องขึ้นไป Hub ใหญ่บางรุ่นจะมี Fiber Module สำหรับเชื่อมโยงอุปกรณ์ผ่านใยแก้วนำแสง



รูปที่ 2.22 Hub [14]

2.16 ภาษา Python

Python เป็นภาษาระดับสูงภาษาหนึ่ง ที่มีความสามารถสูงถูกสร้างขึ้นในปี 1989 โดย Guido van Rossum ซึ่งถูกพัฒนาขึ้นมาโดยไม่ยึดติดกับแพลตฟอร์ม กล่าวคือสามารถรันภาษา Python ได้ทั้งบนระบบ Unix, Linux, Windows NT, Windows 2000, Windows XP หรือแม้แต่ระบบ FreeBSD อีกอย่างหนึ่งภาษาตัว นี้เป็นภาษาลักษณะ Open Source เหมือนแบบ PHP

2.16.1 คุณลักษณะเด่นของภาษา Python

2.16.1.1 ภาษา Python สนับสนุนแนวแบบคิดออปเจกต์โอเรียนเทด หรือ OOP (Object Oriented Programming)

2.16.1.2 Pythonเป็นภาษาคอมพิวเตอร์ที่ไม่คิดมูลค่าการใช้งานและเป็นภาษาที่มีความยืดหยุ่นสูงมาก

2.16.1.3 โค้ดที่เขียนด้วย Python สามารถนำไปรันบนระบบปฏิบัติการอื่นๆ ได้(Portable)เช่น Linux, Ms-windows (95, 98, NT, 2000, XP), Amiga, Be-OS, OS/2, VMS, QNX, และระบบอื่นๆอีกมากมาย

2.16.1.4 Python สนับสนุนเทคโนโลยี COM ของ Ms-windows

2.16.1.5 Python รวมมาตรฐานการอินเตอร์เฟส Tkinter ซึ่งสนับสนุนบนระบบ X windows, Ms-windows และ Macintosh การใช้คำสั่ง Tkinter API ช่วยให้โปรแกรมเมอร์ไม่ต้องแก้ไขโค้ดเมื่อนำไปรันบนระบบปฏิบัติการอื่นๆ

2.16.1.6 Python เป็น Dynamic typing คือ สามารถเปลี่ยนชนิดข้อมูลได้ง่ายและสะดวก

2.16.1.7 Python มี Built-in Object Types คือ โครงสร้างของข้อมูลที่สามารถใช้ได้ใน Python ประกอบด้วย ลิสต์, ดิกชันนารี, สตริง ที่ง่ายต่อการใช้งานและมีประสิทธิภาพสูง

2.16.1.8 Pythonมีเครื่องมือต่างๆ มากมาย เช่น การประมวลผลเท็กซ์ไฟล์ การเรียงข้อมูล การเชื่อต่อสตริง การตรวจสอบเงื่อนไขของข้อความ การแทนคำ เป็นต้น

2.16.1.9 Python มีมอดูลสำหรับจัดการ Regular Expression

2.16.1.10 Python มีมอดูลที่สร้างขึ้นจากนักพัฒนาสนับสนุนมากมาย ได้แก่ COM, Image, CORBA, ORBs, XML เป็นต้น

2.16.1.11 Python จัดการหน่วยความจำอย่างอัตโนมัติ สามารถจักการพื้นที่หน่วยความจำที่ไม่ต่อเนื่องให้ทำงานได้อย่างมีประสิทธิภาพ

2.16.1.12 Python อนุญาตให้ฝังชุดคำสั่งของ Python เอาไว้ภายในโค้ตภาษา C/C++ ได้

2.16.1.13 Python อนุญาตให้โปรแกรมเมอร์สร้าง Dynamic Link Libray (DLL) เพื่อใช้ร่วมกับ Python

2.16.1.14 Python มีมอดูลสนับสนุนเกี่ยวกับเน็ตเวิร์ก โปรเซส เธรด regular, expression, xml, GUI และอื่นๆ

2.16.1.15 Pythonประกอบด้วยมอดูลสำหรับสร้าง Internet Script และติดต่อกับอินเทอร์เน็ตผ่าน Sockets, และทำหน้าที่เป็น CGI Script ตลอดจนใช้งานคำสั่ง FTP, Glopher, XML และอื่นๆอีกมาก

2.16.1.16 Python สามารถประมวลผลทางด้านวิยาศาสตร์ และวิศวกรรมศาสตร์ได้อย่างมีประสิทธิภาพ

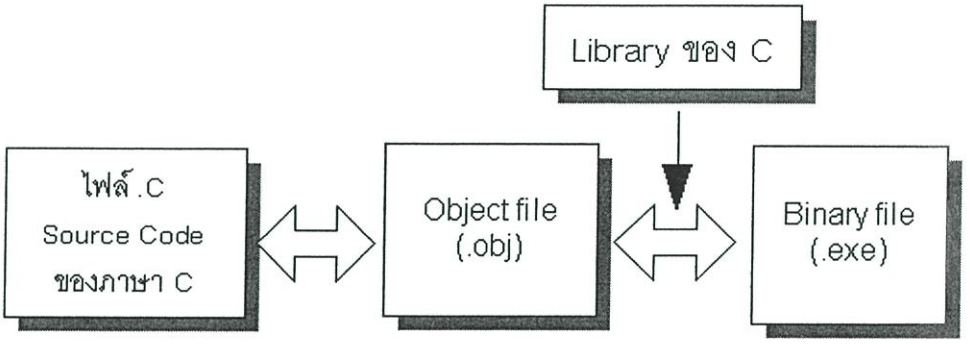
2.16.1.17 Python มีฟังก์ชันสนับสนุนฐานข้อมูล เช่น MySQL, Sybase, Oracle, Informix, ODBC และอื่นๆ

2.16.1.18 Python มีไลบรารีสนับสนุนด้านการสร้างภาพกราฟฟิก เช่น ทำภาพเบลอ หรือภาพชัด หรือเขียนข้อความบนภาพ ตลอดจนบันถึกไฟล์ในรูปแบบต่างๆ ได้อย่างสะดวกและมีประสิทธิภาพ

2.16.2 หลักการทำงานของภาษา Python

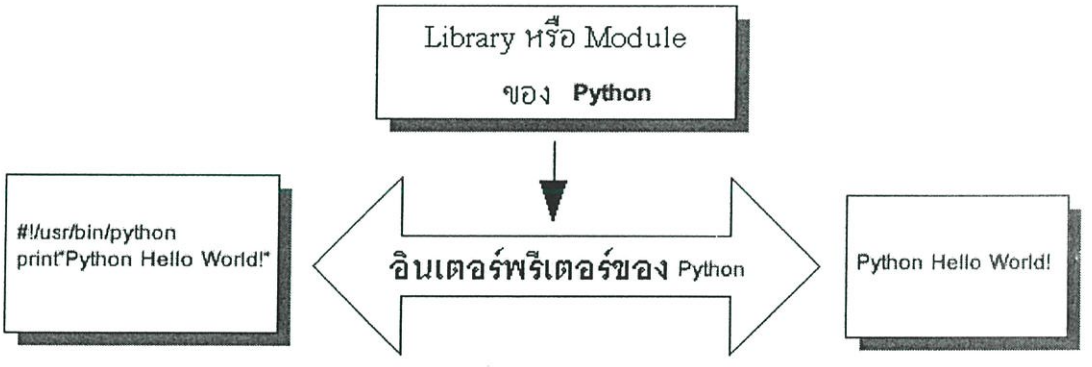
เมื่อเราได้เขียนโค้ดขึ้นมาตามโครงสร้างของโปรแกรมภาษาใดก็ตาม และการจะให้โค้ดคำสั่งเหล่านั้นทำงานได้ก็จะต้องมีตัวแปรภาษามาจัดการแปลโค้ดคำสั่ง เพื่อให้ทำงานตามที่เราต้องการ โดยลักษณะของตัวแปลภาษานั้นแบ่งได้ 2 ประเภทใหญ่ ๆ คือ

2.16.2.1. คอมไพเลอร์ (Compiler) เป็นตัวแปลภาษาสำหรับภาษา C, C++, Pascal การทำงานก็คือจะตรวจสอบความผิดพลาดของโค้ดคำสั่งตั้งแต่ต้นจนจบก่อน หรือเรียกว่าการคอมไพล์ ถ้าไม่มีข้อผิดพลาดก็จะทำการแปลโค้ดคำสั่งของเราให้เป็นไฟล์นามสกุล .obj (object file) จากนั้นก็ทำการแปลไฟล์ .obj ให้เป็นไบนารีไฟล์ .exe เพื่อทำงานต่อไป ดังตัวอย่างการทำงานของคอมไพเลอร์ภาษา C ดังรูป



รูปที่ 2.23 ตัวอย่างการทำงานของคอมไพเลอร์ภาษา C [15]

2.16.2.2. อินเตอร์พรีเตอร์ (Interperter) จะทำงานเป็นบรรทัดต่อบรรทัด คือ อ่านโค้ดคำสั่งมาบรรทัดหนึ่งแล้วก็ทำงานให้ผลออกมาเลย



รูปที่ 2.24 อินเตอร์พรีเตอร์ (Interperter) [15]

จากรูปตัวอย่างในกรณีที่มีการเรียกใช้ฟังก์ชั่นจากไลบรารี (Library) หรือโมดูล (Module) ของภาษา Python อินเตอร์พรีเตอร์ของภาษา Python ก็จะไปทำการเรียกฟังก์ชั่นเหล่านั้นให้ทำงานแล้วจึงแสดงผลการทำงานออกมา

ในส่วนของประสิทธิภาพการทำงานนั้นตัวแปลภาษาแบบคอมไพเลอร์จะทำงานได้เร็วกว่าตัวแปลภาษาแลลอินเตอร์พรีเตอร์ เพราะโค้ดคำสั่งถูกคอมไพล์และลิงค์โดยตัวแปลภาษาแบบคอมไพเลอร์ผ่านแล้วได้เป็นไฟล์ .exe ออกมา จากนั้นก็เป็นขั้นตอนการทำงานอย่างเดียว

2.17 ภาษา PHP

พีเอชพี (PHP) คือ ภาษาคอมพิวเตอร์ในลักษณะเซิร์ฟเวอร์-ไซด์ สคริปต์ โดยลิขสิทธิ์อยู่ในลักษณะโอเพนซอร์ส ภาษาพีเอชพีใช้สาหรับจัดทาเว็บไซต์ และแสดงผลออกมาในรูปแบบ HTMLโดยมีรากฐานโครงสร้างคำสั่งมาจากภาษา ภาษาซี ภาษาจาวา และ ภาษาเพิร์ล ซึ่ง ภาษาพีเอชพีนั้นง่ายต่อการเรียนรู้ ซึ่งเป้าหมายหลักของภาษานี้ คือให้นักพัฒนาเว็บไซต์สามารถเขียน เว็บเพจ ที่มีความตอบโต้ได้อย่างรวดเร็ว

2.17.1 คุณสมบัติ

การแสดงผลของพีเอชพี จะปรากฏในลักษณะHTML ซึ่งจะไม่แสดงคำสั่งที่ผู้ใช้เขียน ซึ่งเป็นลักษณะเด่นที่พีเอชพีแตกต่างจากภาษาในลักษณะไคลเอนต์-ไซด์ สคริปต์ เช่น ภาษาจาวา สคริปต์ ที่ผู้ชมเว็บไซต์สามารถอ่าน ดูและคัดลอกคำสั่งไปใช้เองได้ นอกจากนี้พีเอชพียังเป็นภาษาที่เรียนรู้และเริ่มต้นได้ไม่ยาก โดยมีเครื่องมือช่วยเหลือและคู่มือที่สามารถหาอ่านได้ฟรีบนอินเทอร์เน็ต ความสามารถการประมวลผลหลักของพีเอชพี ได้แก่ การสร้างเนื้อหาอัตโนมัติจัดการคำสั่ง การอ่านข้อมูลจากผู้ใช้และประมวลผล การอ่านข้อมูลจากดาต้าเบส ความสามารถจัดการกับคุกกี้ ซึ่งทำงานเช่นเดียวกับโปรแกรมในลักษณะCGI คุณสมบัติอื่นเช่น การประมวลผลตามบรรทัดคำสั่ง (Command line scliping)ทาให้ผู้เขียนโปรแกรมสร้างสคริปต์พีเอชพี ทางานผ่านพีเอชพีพาร์เซอร์ (PHP parser) โดยไม่ต้องผ่านเซิร์ฟเวอร์หรือเบราว์เซอร์

2.17.2 การรองรับพีเอชพี

คำสั่งของพีเอชพี สามารถสร้างผ่านทางโปรแกรมแก้ไขข้อความทั่วไป เช่น โน้ตแพด หรือ vi ซึ่งทำให้การทำงานพีเอชพี สามารถทำงานได้ในระบบปฏิบัติการหลักเกือบทั้งหมดโดยเมื่อเขียนคำสั่งแล้วนำมาประมวลผล Apache, Microsoft Internet Information Services (IIS) และอื่นๆ อีกมากมาย สำหรับส่วนหลักของ PHP ยังมี Module ในการรองรับ CGI มาตรฐานซึ่ง PHP สามารถทำงานเป็นตัวประมวลผล CGI ด้วย และ PHP มีอิสรภาพในการเลือกระบบปฏิบัติการ และเว็บเซิร์ฟเวอร์ นอกจากนี้คุณยังสามารถใช้สร้างโปรแกรมโครงสร้าง สร้างโปรแกรมเชิงวัตถุ (OOP) หรือสร้างโปรแกรมที่รวมทั้งสองอย่างเข้าด้วยกัน แม้ว่าความสามารถของคำสั่ง OOP มาตรฐานในเวอร์ชันนี้ยังไม่สมบูรณ์ แต่ตัวไลบรารีทั้งหลายของโปรแกรม และตัวโปรแกรมประยุกต์ (รวมถึง PEAR library) ได้ถูกเขียนขึ้นโดยใช้รูปแบบการเขียนแบบ OOP เท่านั้น

พีเอชพีสามารถทางานร่วมกับฐานข้อมูลได้หลายชนิด ซึ่งฐานข้อมูลส่วนหนึ่งที่รองรับได้แก่ ออราเคิล dBase PostgreSQL IBM DB2 MySQL Informix ODBC โครงสร้างของฐานข้อมูลแบบ DBX ซึ่งทาให้พีเอชพีใช้กับฐานข้อมูลอะไรก็ได้ที่รองรับรูปแบบนี้และ PHP ยังรองรับ ODBC (Open Database Connection) ซึ่งเป็นมาตรฐานการเชื่อมต่อฐานข้อมูลที่ใช้กันแพร่หลายอีกด้วย คุณสามารถเชื่อมต่อกับฐานข้อมูลต่างๆ ที่รองรับมาตรฐานโลกนี้ได้

พีเอชพียังสามารถรองรับการสื่อสารกับการบริการในโพรโทคอลต่างๆ เช่น LDAP IMAP SNMP NNTP POP3 HTTP COM (บนวินโดวส์) และอื่นๆ อีกมากมาย คุณสามารถเปิด Socket บนเครือข่ายโดยตรง และ ตอบโต้โดยใช้ โพรโทคอลใดๆ ก็ได้

2.17.3 โครงสร้างพื้นฐานของ PHP

PHP เป็นภาษาที่สามารถใช้งานร่วมกับภาษา HTML ได้ ในการเขียนรหัส (Code) โปรแกรม มีวิธีการเขียนได้หลายรูปแบบ จึงจำเป็นต้องมี สัญลักษณ์ที่บ่งบอกถึงขอบเขตของ PHP เพื่อที่จะแยกโค้ด PHP ออกจากโค้ด HTML ได้อย่างชัดเจน โดยมีรูปแบบในการเขียนแทนด้วยสัญลักษณ์ต่าง ๆ ที่เรา สามารถนามาใช้แยกโค้ด PHP ได้มีดังนี้

การเขียนพีเอชพีนั้นมีรูปแบบการเขียนอยู่หลายแบบดังนี้

1) เขียนแบบ SGML เป็นรูปแบบการเขียนที่เป็นมาตรฐานของภาษา XML โดยมีรูปแบบดังนี้

<?...?>

2) เขียนแบบ XML เป็นรูปแบบการเขียนของภาษาประเภท XML โดยมีรูปแบบดังนี้

<?php...?>

3) เขียนแบบภาษาสคลิปต์หรือการเขียนแบบจาวาสคลิปต์โดยมีรูปแบบดังนี้

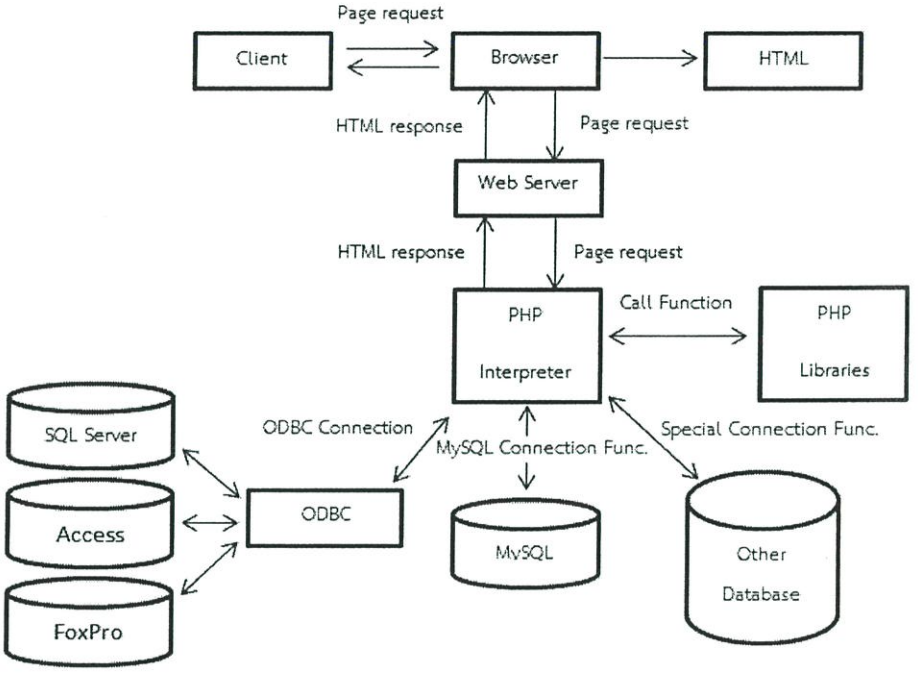
<script language="PHP">...</script>

4) เขียนแบบ ASP เป็นรูปแบบการเขียนที่เป็นมาตรฐานของภาษาประเภท ASP โดยมีรูปแบบดังนี้

<%...%>

จากรูปแบบข้างต้นทั้ง 4 รูปแบบ รูปแบบที่ได้รับความนิยมในการเขียนพีเอชพีคือ การเขียนในรูปแบบที่ 1

2.17.4 หลักการทางานของ PHP



รูปที่ 2.25 แสดงขั้นตอนการทำงาน PHP Script Request/Response [16]

2.17.4.1 จากไคลเอนต์จะเรียกไฟล์ php script ผ่านทางโปรแกรมบราวเซอร์ (Internet Explore)

2.17.4.2 บราวเซอร์จะส่งคำร้อง (Request) ไปยังเว็บเซิร์ฟเวอร์ผ่านทางเครือข่ายอินเทอร์เน็ต

2.17.4.3 เมื่อเว็บเซิร์ฟเวอร์รับคำรองขอจากบราวเซอร์แล้วก็จะนำสคริปต์ phpที่เก็บอยู่ในเซิร์ฟเวอร์มาประมวลผลด้วยโปรแกรมแปลภาษา PHP ที่เป็นอินเตอร์พรีเตอร์

2.17.4.4 กรณีที่ php script มีการเรียกใช้ข้อมูลก็จะติดต่อกับฐานข้อมูลต่างๆผ่านทาง ODBC Connection ถ้าเป็นฐานข้อมูลกลุ่ม Microsoft SQL Server, Microsoft Access, FoxProหรือใช้ Function Connection ที่มีอยู่ใน PHP Library ในการเชื่อมต่อฐานข้อมูลเพื่อดึงข้อมูลออกมาหลังจากแปลสคริปต์ PHP เสร็จแล้วจะได้รับไฟล์ HTML ใหม่ที่มีแต่แท็กHTML ไปยัง Web Server

2.17.4.5 Web Server ส่งไฟล์ HTML ที่ได้ผ่านการแปลแล้วกลับไปยังบราวเซอร์ที่ร้องขอผ่านทางเครือข่ายอินเตอร์เน็ต

2.17.4.6 บราวเซอร์รับไฟล์ HTML ที่เว็บเซิร์ฟเวอร์ส่งมาให้แปล HTML แสดงผลออกมาทางจอภาพเป็นเว็บเพจโดยใช้ตัวแปลภาษา HTML ที่อยู่ในบราวเซอร์ ซึ่งเป็นอินเตอร์พรีเตอร์เช่นเดียวกัน

2.17.5 ความสามารถของภาษา PHP

2.17.5.1 เป็นภาษาที่มีลักษณะเป็นแบบ Open source ผู้ใช้สามารถ Download และนำ Source Code ของ PHP ไปใช้ได้โดยไม่เสียค่าใช้จ่าย

2.17.5.2 เป็นสคริปต์แบบ Server Side Script ดังนั้นจึงทางานบนเว็บเซิร์ฟเวอร์ไม่ส่งผลกับการทางานของเครื่อง Client โดย PHP จะอ่านโค้ด และทางานที่เซิร์ฟเวอร์ จากนั้นจึงส่งผลลัพธ์ที่ได้จากการประมวลผลมาที่เครื่องของผู้ใช้ในรูปแบบของ HTML ซึ่งโค้ดของ PHP นี้ผู้ใช้จะไม่สามารถมองเห็นได้

2.17.5.3 PHP สามารถทางานได้ในระบบปฏิบัติการที่ต่างชนิดกัน เช่น Unix, Windoes, Mac OS หรือ Risc OS อย่างมีประสิทธิภาพ เนื่องจาก PHP เป็นสคริปต์ที่ต้องทางานบนเซิร์ฟเวอร์ ดังนั้นคอมพิวเตอร์สาหรับเรียกใช้คาสั่ง PHP จึงจาเป็นต้องติดตั้งโปรแกรมเว็บเซิร์ฟเวอร์ไว้ด้วย เพื่อให้สามารถประมวลผล PHP ได้

. 2.17.5.4 PHP สามารถทางานได้ในเว็บเซิร์ฟเวอร์หลายชนิด เช่น Personal WebServer(PWS), Apache, OmniHttpd และ Internet Information Service(IIS) เป็นต้น

2.17.5.5 ภาษา PHP สนับสนุนการเขียนโปรแกรมเชิงวัตถุ (Opject oriented Programming)

2.17.5.6 PHP มีความสามารถในการทางานร่วมกับระบบจัดการฐานข้อมูลที่หลากหลาย ซึ่งระบบจัดการฐานข้อมูลที่สนับสนุนการทางานของ PHP เช่น Oracle, MySQL, FilePro, Solid, FrontBase, M SQL และ MS SQL เป็นต้น

2.17.5.7 PHP อนุญาตให้ผู้ใช้สร้างเว็บไซต์ซึ่งทางานผ่านโปรโตคอลชนิดต่างๆ ได้เช่น LDAP, IMAP, SNMP, POP3 และ HTTP เป็นต้น

2.17.5.8 โค้ด PHP สามารถเขียน และอ่านในรูปแบบของ XML ได้

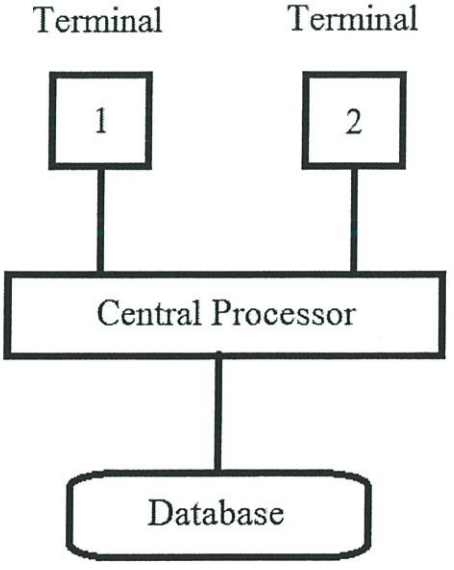
2.18 สถาปัตยกรรมของฐานข้อมูล

สถาปัตยกรรมของฐานข้อมูล แบ่งตามโครงสร้างทางกายภาพได้เป็นสองชนิดคือ เมนเฟรมคอมพิวเตอร์ (Mainframe Computer) และคอมพิวเตอร์ส่วนบุคคล (Personal Computer) แต่ถ้าเราแบ่งตามสถาปัตยกรรมของฐานข้อมูลตามจานวนผู้ใช้ สามารถแบ่งได้เป็นสองแบบด้วยกัน คือระบบฐานข้อมูลที่มีผู้ใช้เพียงคนเดียวและระบบฐานข้อมูลที่มีผู้ใช้หลายคนและในที่นี้ พิจารณาการแบ่งสถาปัตยกรรมของฐานข้อมูลตามจานวนผู้ใช้ดังนี้

1.ระบบฐานข้อมูลผู้ใช้คนเดียว โดยปกติแล้วจะเป็นคอมพิวเตอร์ส่วนบุคคล ซึ่งข้อมูลของฐานข้อมูลจะอยู่ในฮาร์ดดิสก์ ระบบจัดการฐานข้อมูลเป็นโปรแกรมเดี่ยวๆและฐานข้อมูลมีขนาดไม่ใหญ่ ระบบนี้สิ้นเปลืองทรัพยากรเนื่องจากต้องรอคาสั่งจากผู้ใช้คนเดียว

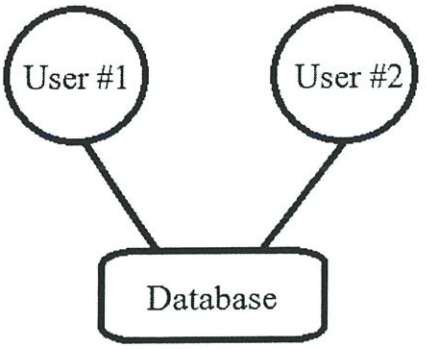
2. ระบบฐานข้อมูลผู้ใช้หลายๆคน ระบบนี้สามารถใช้ได้มากกว่าหนึ่งคนในเวลาเดียวกันทำให้มีประสิทธิภาพเนื่องจากระบบฐานข้อมูลสามารถตอบสนองต่อผู้ใช้ให้หลายๆคนโดยมีรูปแบบที่ชัดเจนได้หลายรูปแบบ เช่น Teleprocessing Client/Server และ Distributed Data Processing เป็นต้น

3. Teleprocessing ระบบนี้ มีประสิทธิภาพสูงและต่อกับเครื่องคอมพิวเตอร์ที่ปลายทางแต่ระบบนี้มีข้อเสียคือ ใช้ซีพียูตัวเดียวในการทางานทั้งหมด ดังนั้นซีพียูต้องมีประสิทธิภาพสูง และเมื่อมีการใช้งานเพิ่มขึ้นเรื่อยๆประสิทธิภาพจะลดลงอย่างมาก นอกจากนี้ยังขึ้นอยู่กับแบนด์วิดท์ (ความสามารถในการรับส่งข้อมูลระหว่างต้นทางกับปลายทาง) ซึ่งต่อระหว่างผู้ใช้ซีพียู ถ้าแบนด์วิดท์แคบประสิทธิภาพก็ลดลง และเมื่อเพิ่มผู้ใช้อีก แบนด์วิดท์ของแต่ละคนที่ได้ก็จะน้อยลง อีกระบบนี้ใช้ในเมนเฟรมคอมพิวเตอร์และได้แสดงสถาปัตยกรรมแบบ Teleprocessing ดังแสดงในรูปที่ 2.26



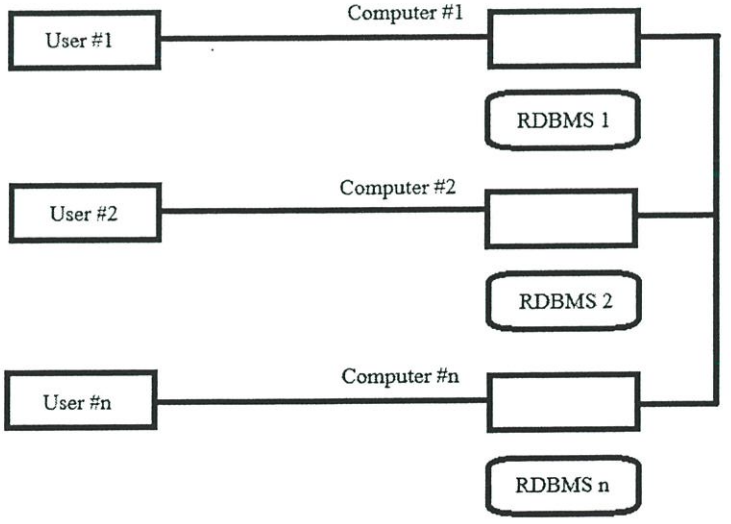
รูปที่ 2.26 สถาปัตยกรรมแบบ Teleprocessing [17]

4. Client/Server ในระบบนี้ซีพียูที่อยู่ส่วนกลางเรียกว่าเซิร์ฟเวอร์ ต่อกับเครื่องคอมพิวเตอร์หลายๆเครื่องที่เรียกว่าไคลเอ็นต์ (client) และ ได้แสดงถึงสถาปัตยกรรมแบบ Client/Server ดังแสดงในรูปที่ 2.27



รูปที่ 2.27 สถาปัตยกรรมแบบ Client/Server [17]

5. Distributed Data Processing ระบบนี้ใช้เซิร์ฟเวอร์หลายเครื่องและผู้ใช้หลายคน ซึ่งเซิร์ฟเวอร์หลายๆเครื่องนี้จะเป็นการเชื่อมต่อฐานข้อมูลจากเครื่องคอมพิวเตอร์ต่างๆให้เป็นฐานข้อมูลเดียวกัน ช่วยลดการติดต่อระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์ ผลที่ได้คือประสิทธิภาพสูงแต่ข้อเสียคือต้องมีการควบคุมอย่างระมัดระวังเพื่อให้แน่ใจว่าข้อมูลเซิร์ฟเวอร์แต่ละตัวจะสอดคล้องกันกับคอมพิวเตอร์เซิร์ฟเวอร์อื่นๆ ระบบนี้จะแสดงในรูปที่ 2.28



รูปที่ 2.28 สถาปัตยกรรมแบบ Distributed Data Processing [17]

เมื่อมีระบบฐานข้อมูลอยู่เซิร์ฟเวอร์หรือเครื่องคอมพิวเตอร์ส่วนบุคคลแล้วจาเป็นต้องมีเทคโนโลยีสาหรับเข้าถึงข้อมูลในฐานข้อมูลที่มีอยู่ที่ไคลเอ็นต์ เทคโนโลยีสาหรับเข้าถึงข้อมูลนี้เรียกว่าเทคโนโลยีการอินเตอร์เฟส

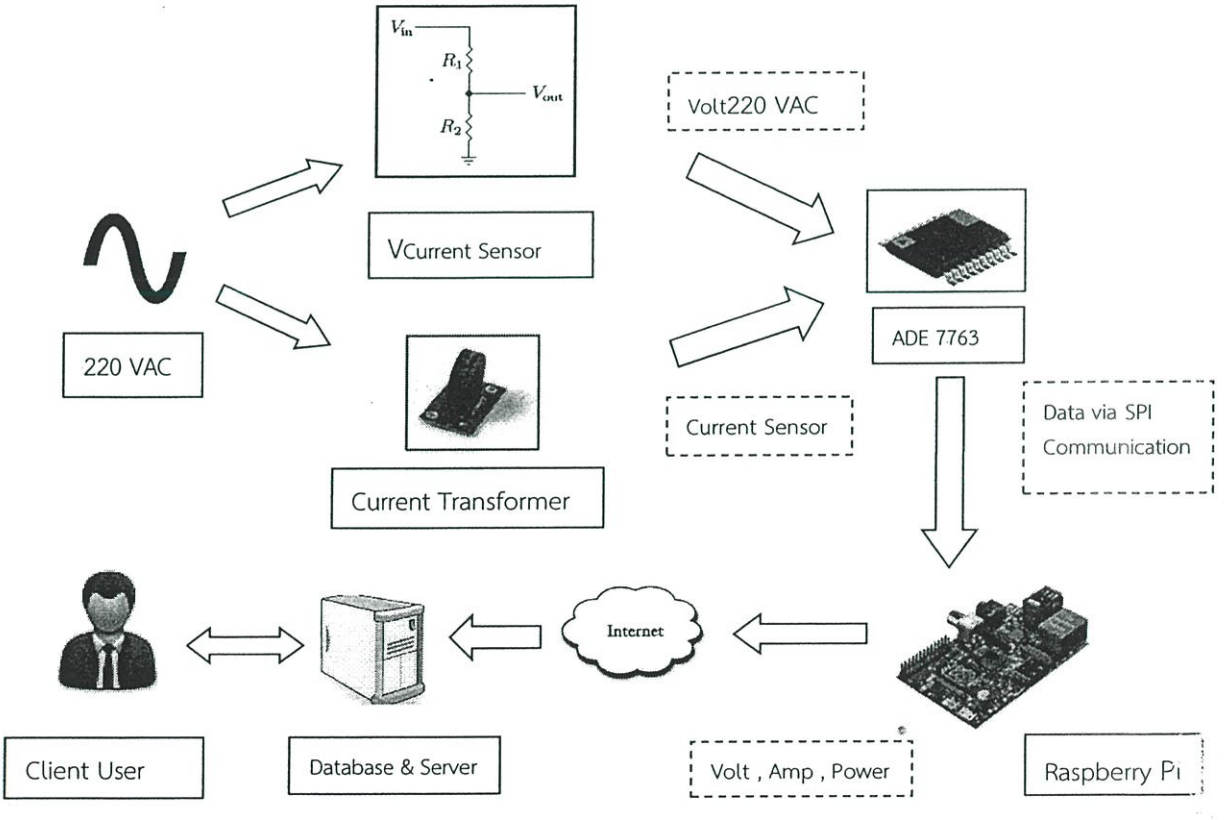
บทที่ 3

การออกแบบและการจัดทำปริญญานิพนธ์

ในการปฏิบัติงานให้สำเร็จได้ตามเป้าและภายในระยะเวลาที่กำหนด จะเป็นต้องวางแผนโครงงานงาน เพื่อให้ทราบแนวทางและขั้นตอนการปฏิบัติงาน

3.1 การออกแบบ

เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยสมองกลฝังตัวแบ่งการทำงานออกเป็น 2 ส่วนหลักคือ ส่วนคำนวณค่าทางไฟฟ้าและส่วนของระบบจัดการฐานข้อมูลและเว็บเซิร์ฟเวอร์โดย Raspberry Pi จะรับค่าแรงดัน กระแส และกำลังไฟฟ้า จาก IC ADE7763 และทำการส่งข้อมูลเพื่อบันทึกลงในระบบฐานข้อมูล โดยแสดงการทำงานของระบบ ได้ดังนี้



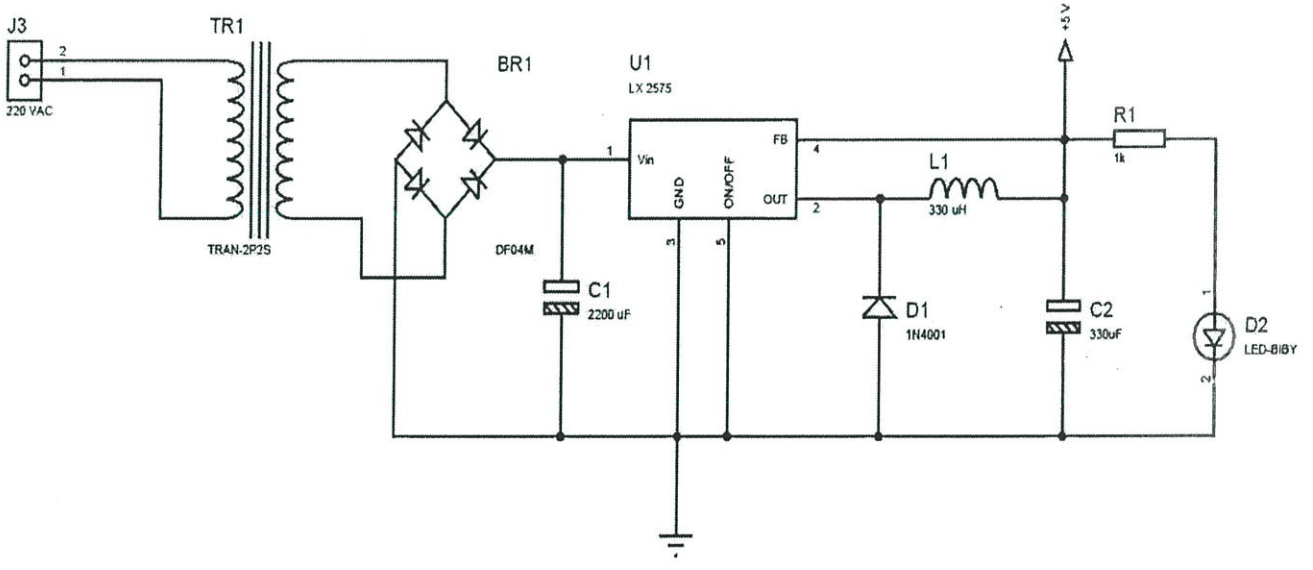
รูปที่ 3.1 บล็อกไดอะแกรมของระบบ

3.1.1 การออกแบบวงจรการคำนวณ

เครื่องวัดกำลังไฟฟ้าที่จัดทำขึ้นนั้นจะทำการออกแบบวงจรโดยแบ่งออกเป็น 3 ส่วน การทำงานหลักๆ คือ วงจรจ่ายแรงดัน วงจรการคำนวณแรงดัน กระแส กำลังไฟฟ้า และส่วนประมวลผลกลาง

3.1.1.1 วงจรจ่ายแรงดัน

ในการสร้างอุปกรณ์เครื่องวัดกำลังไฟฟ้านั้นจะต้องใช้ไฟเลี้ยงในวงจร 5V จ่ายให้แต่ละอุปกรณ์ในวงจร โดยมีต้นกำเนิดจากไฟบ้าน 220 VAC จากนั้นนำมาเข้าหม้อแปลงเพื่อลดแรงดันไฟให้เหลือประมาณ 6 VAC และถูกนำไปเปลี่ยนแรงดันกระแสตรงโดยใช้ Diode Bridge แต่แรงดัน 6 V นั้นก็ยังไม่สามารถนำไปใช้กับวงจรได้ จึงต้องมีการลดแรงดันอีกครั้งโดยใช้อุปกรณ์สวิตชิ่ง เรกกูเรเตอร์ LM2575sx โดยอุปกรณ์ตัวนี้มีทำหน้าที่เหมือนกับ IC 7805 คือลดแรงดันให้เหลือ 5V แต่จะเสถียรกว่า IC 7805 มาก โดยสามารถตรวจสอบได้ว่าวงจรทำงานได้หรือไม่โดยมีหลอด LED เพื่อทดสอบ สามารถแสดงวงจรได้ดังนี้



รูปที่ 3.2 วงจรลดแรงดันโดยใช้ไดโอดบริดจ์และ IC LM2575sx

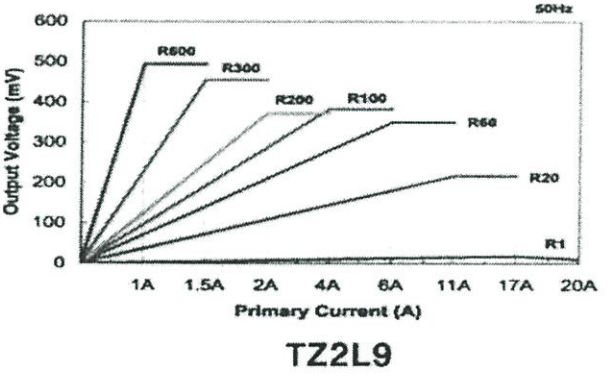
3.1.1.2 วงจรการคำนวณแรงดัน กระแส และกำลังไฟฟ้า

ในส่วนการคำนวณนี้จะใช้ IC ADE7763 เป็นอุปกรณ์ทำงานหลัก ซึ่งคุณสมบัติของ IC ADE7763 นั้นคือ สามารถที่จะหาค่าทางไฟฟ้าได้หลายค่าเช่น แรงดัน กระแส และกำลังไฟฟ้า

ปรากฏ ด้วยคุณสมบัตินี้ทางผู้จัดทำจึงนำ IC ตัวนี้มาใช้ในโครงงานเพื่อหาค่าทางไฟฟ้าต่างๆโดยจะแบ่งการทำงานออกเป็น 2 ส่วนหลักๆคือ ภาครับกระแสและแรงดัน

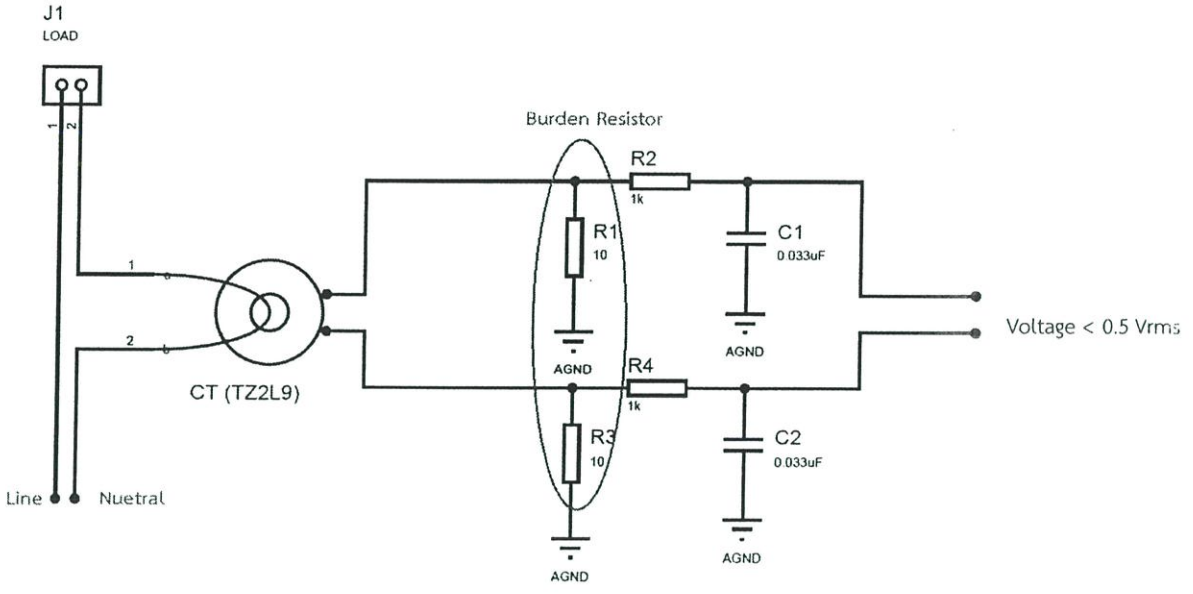
1) วงจรภาครับกระแส

เนื่องจาก Power IC Chip เบอร์ ADE7763 ไม่สามารถรับกระแสเข้ามาคำนวณตรงๆ ได้ดังนั้นจึงต้องแปลงกระแสให้เป็นแรงดันก่อนโดยใช้อุปกรณ์เหนี่ยวนำกระแสหรือ Current Transformer (TZ2L9) เป็นตัวเหนี่ยวนาสายไฟจากโหลดเพื่อเหนี่ยวนำกระแสมา จากนั้นจะถูกตัวต้านทาน R1 และ R3 ซึ่ง เป็นตัวต้านทาน Burden Resistor ซึ่งจะทำให้แรงดันไหลผ่านมีค่าไม่เกิน 0.5 Vrms (แรงดันที่ไอซีจะรับได้) โดยจะพิจารณาได้จากกราฟความสัมพันธ์จากกราฟของ แรงดันและกระแสดังต่อไปนี้



รูปที่ 3.3 กราฟแสดงความสัมพันธ์ของแรงดันและกระแส

จากกราฟสามารถอธิบายได้ว่า เมื่อตัวต้านทาน Burden ที่ต่อขนาดทางด้านเอาท์พุทของ TZ2L9 นั้นเมื่อใช้ตัวต้านทานค่า 20 โอห์มจะให้แรงดันไหลผ่านตัวต้านทานประมาณ 200 mV หรือ 0.2 Vrms และสามารถวัดกระแสได้ในค่าสูงๆอีกด้วย



รูปที่ 3.4 วงจรด้านรับกระแส

2) วงจรภาครับแรงดัน

ในส่วนภาครับแรงดันนั้นจะใช้แรงดันกระแสสลับ 7 V เป็นแรงดันขาเข้าในการทดลอง เพราะเป็นการป้องกันอุปกรณ์เสียหาย หากใช้แรงดันสูงๆ เมื่อเกิดข้อผิดพลาดจะทำให้อุปกรณ์เสียหายได้ และจากข้อกำหนดว่าแรงดันขาออกนั้นจะต้องมีไม่เกิน 0.5 Vrms เพราะฉะนั้นจึงสามารถใช้หลักการแบ่งแรงดันเพื่อคำนวณหาค่าตัวต้านทานได้

จากสมการแบ่งแรงดัน $V_1 = E\frac{R_1}{R_1 + R_3}$ โดยที่ V1 คือ แรงดันขาออก E คือ แรงดันขาเข้า และกำหนดตัวต้านทาน R_3 ให้มีค่าเท่ากับ 1k โอห์ม และให้แรงดันเอาท์พุตเป็น 0.25 V เมื่อแทนค่าต่างๆแล้วจะทำให้ได้ว่า

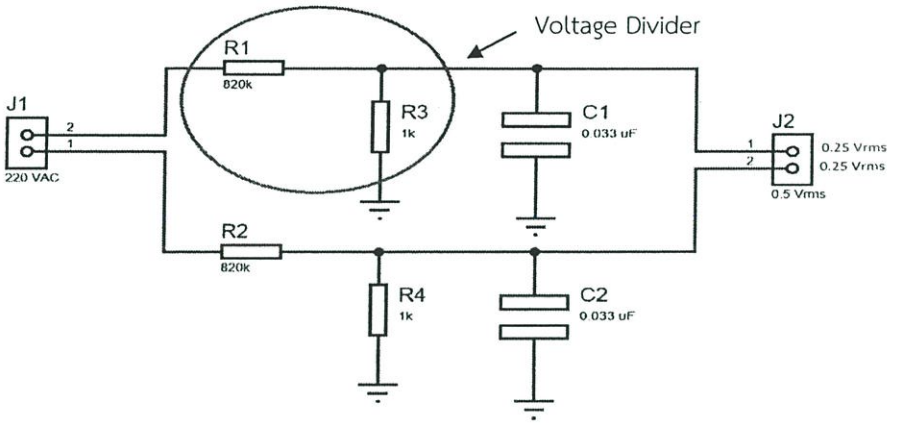
$V_1 = 0.25V, E = 220V, R_3 = 1k$

$$0.25 = \frac{220(1000)}{1000 + R_1}$$

$$0.25R_1 + 250 = 220,000$$

$$0.25R_1 = 219,750$$

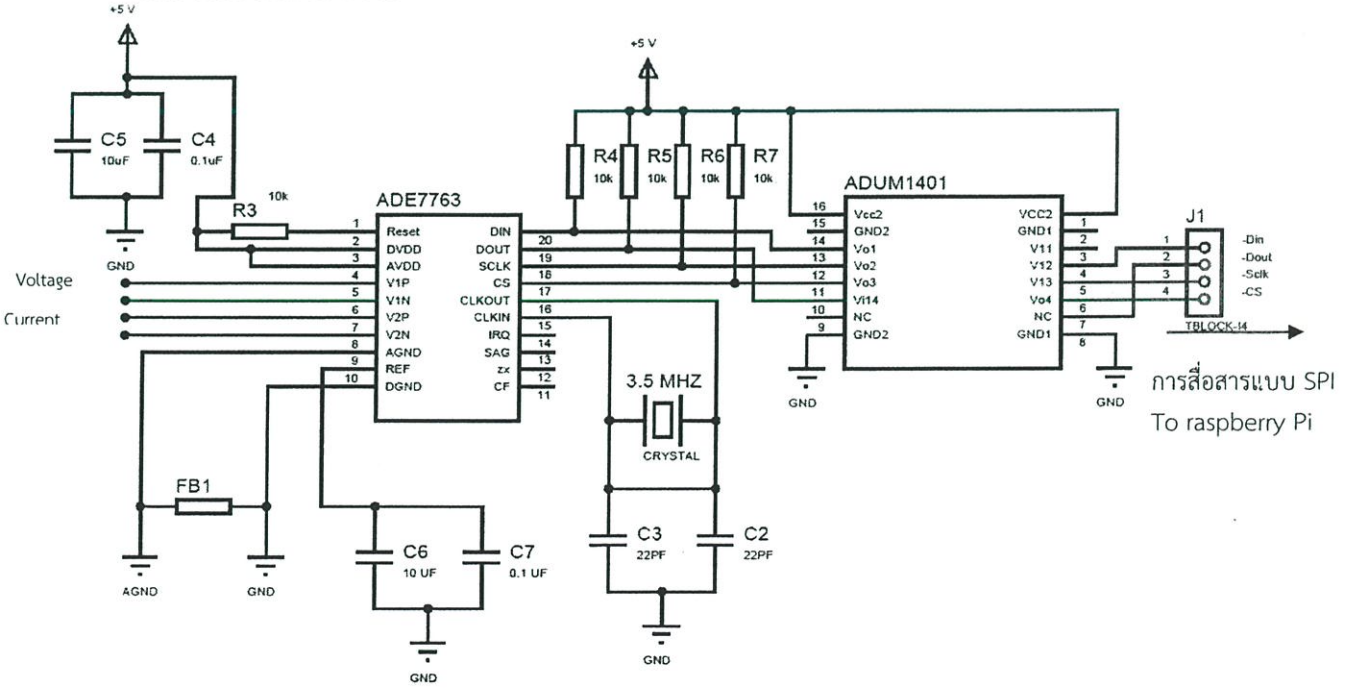
$$R_1 = 820k$$



รูปที่ 3.5 วงจรแบ่งแรงดัน

3.1.1.3 ส่วนประมวลผลกลาง

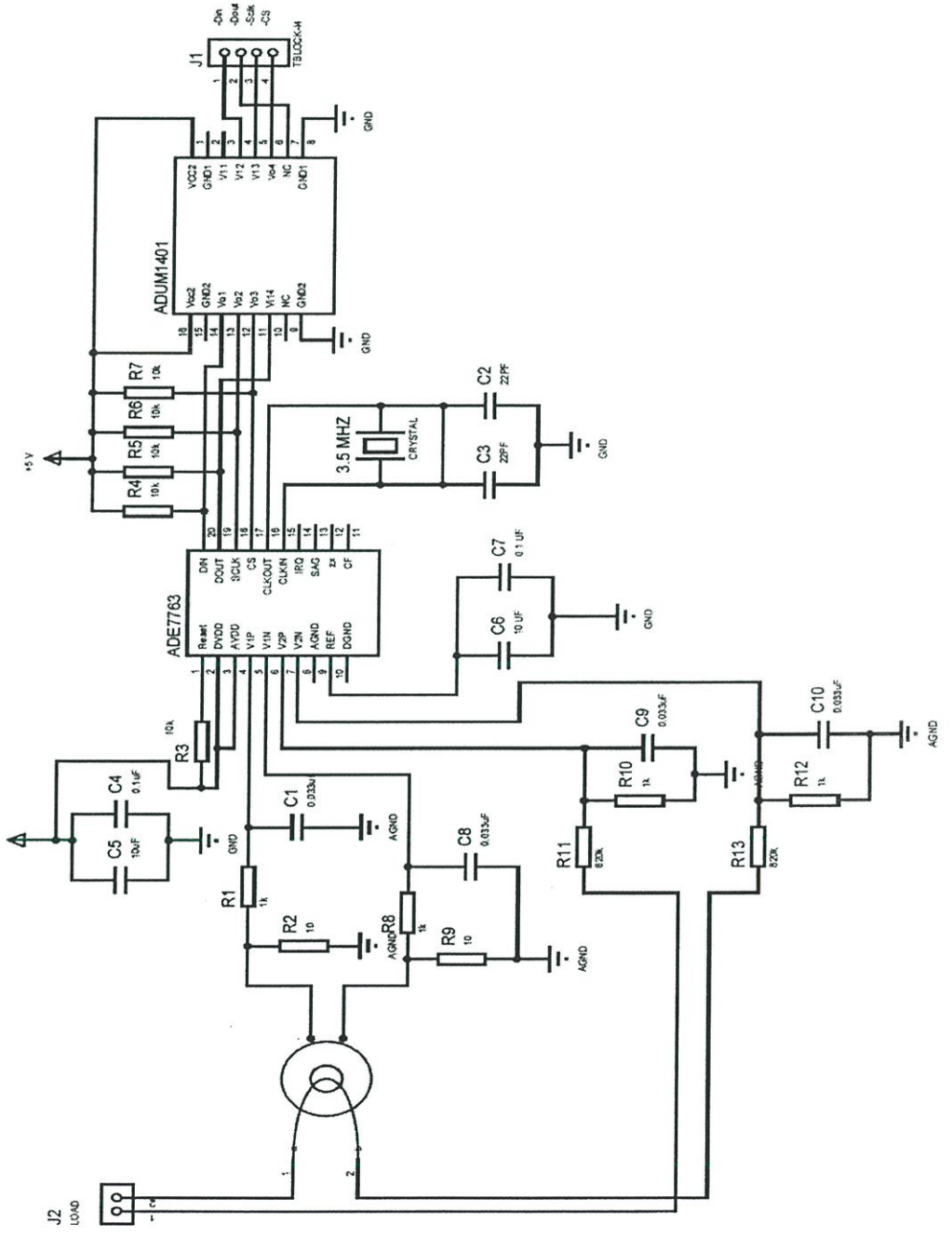
ในส่วนประมวลผลกลางนี้ จะทำหน้าที่คำนวณค่าแรงดัน กระแส และกำลังไฟฟ้าทั้งหมด โดย IC-ADE7763 จะทำหน้าที่ประมวลผลข้อมูลและส่งข้อมูลไปยัง ADuM1401 ซึ่งมีหน้าที่ลดสัญญาณรบกวนที่เกิดขึ้น จากนั้นข้อมูลจะถูกส่งไปยัง Raspberry Pi ด้วยการสื่อสารแบบ SPI จากนั้น Raspberry Pi ก็จะทำการจัดเก็บข้อมูล และแสดงผลผ่านทางจอภาพ หรือ ส่งค่าไปยัง Database Server ต่อไป



รูปที่ 3.6 วงจรคำนวณของไอซี ADE7763

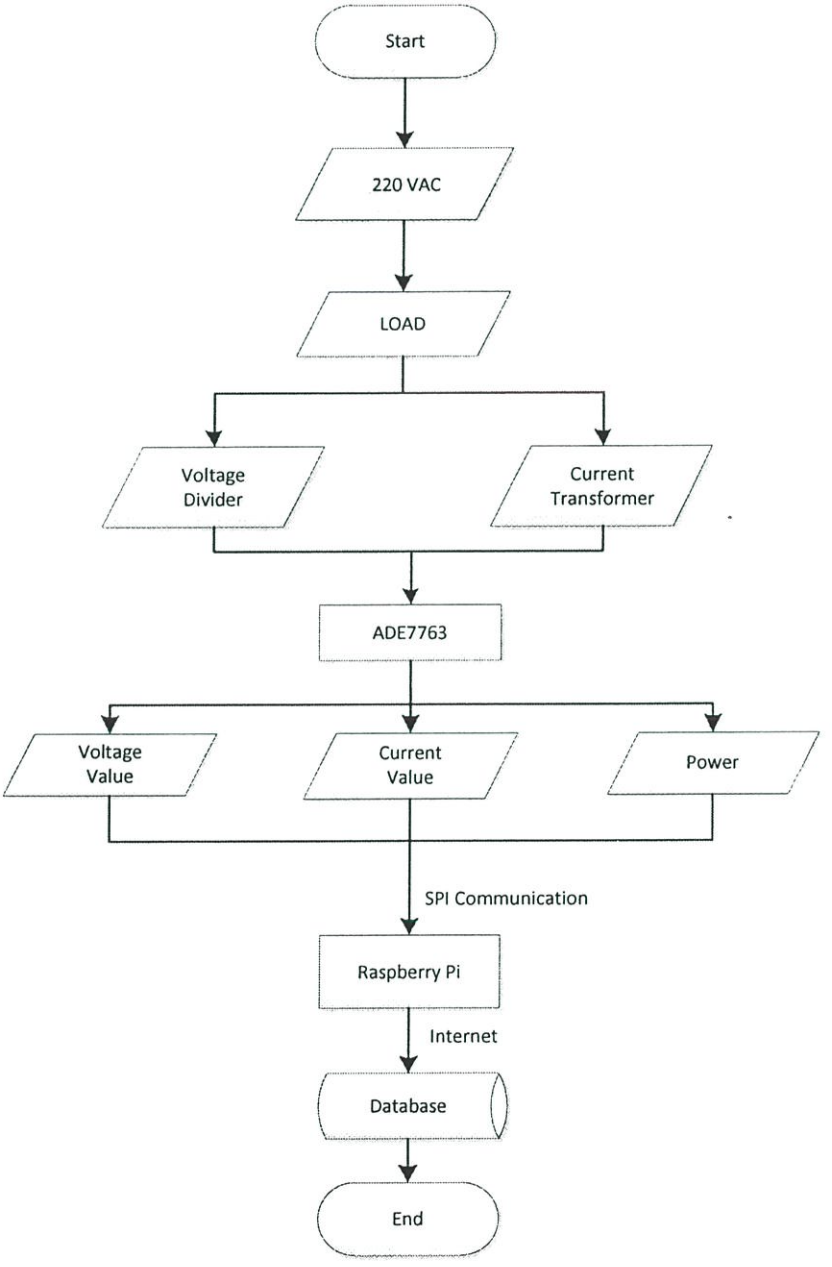
3.1.1.4 วงจรการทำงานรวม

เมื่อนำวงจรทั้งสามส่วนมารวมกันแล้วจะทำให้วงจรทำงานได้สมบูรณ์โดยแสดงได้ดังรูปที่ 3.7



รูปที่ 3.7 วงจรเครื่องวัดกำลังไฟฟ้า

จากการออกแบบวงจรการทำงานของวงจรคำนวณค่าทางไฟฟ้าสามารถแสดงเป็นแผนภาพการทำงานได้ดังรูปที่ 3.8



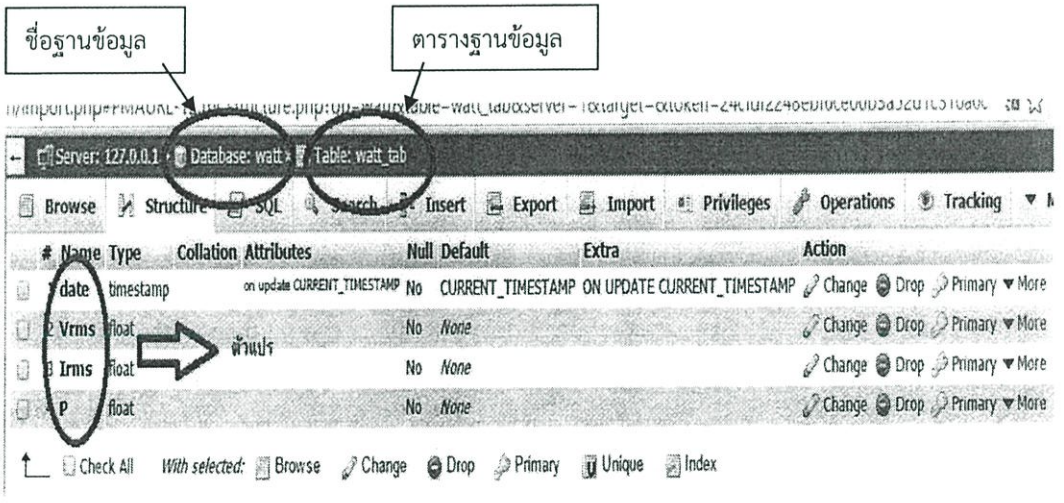
รูปที่ 3.8 โฟล์วชาร์ตการทำงานของระบบ

3.1.2 การออกแบบฐานข้อมูลและสร้างเซิร์ฟเวอร์

เป็นการสร้างและออกแบบฐานข้อมูลเพื่อทำหน้าที่ในจัดเก็บข้อมูลที่ต้องการ โดยมีการกำหนดฐานข้อมูลชื่อว่า watt และสร้างตารางฐานข้อมูลชื่อว่า watt_tab โดยในฐานข้อมูลที่จัดทำขึ้น จะทำการสร้างตัวแปรเพื่อจัดเก็บข้อมูลทั้งสิ้น 4 ตัวคือ

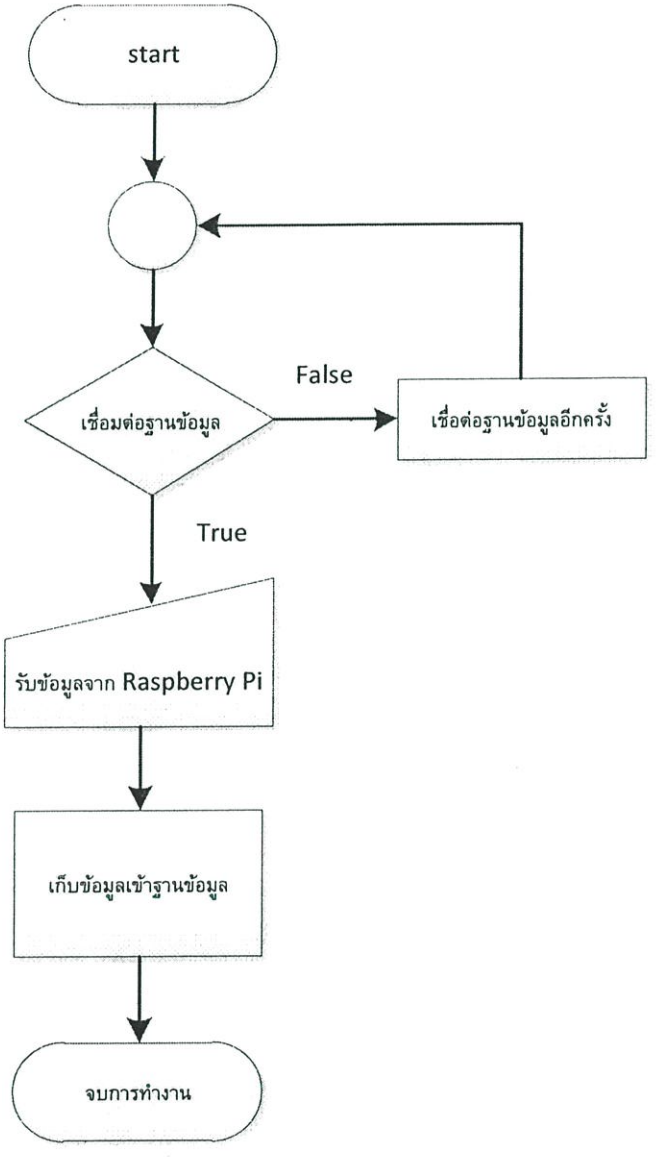
- ตัวแปร Date คือ ข้อมูลวันที่และเวลา ที่ข้อมูลถูกส่งเข้ามายังฐานข้อมูล
- ตัวแปร Vrms คือ ข้อมูลค่าแรงดันที่ถูกส่งเข้ามายังฐานข้อมูล
- ตัวแปร Irms คือ ข้อมูลค่ากระแสที่ถูกส่งเข้ามายังฐานข้อมูล
- ตัวแปร P คือ ข้อมูลค่ากำลังไฟฟ้าที่ถูกส่งเข้ามายังฐานข้อมูล

ดังแสดงในรูปที่ 3.9



รูปที่ 3.9 ฐานข้อมูลในเซิร์ฟเวอร์

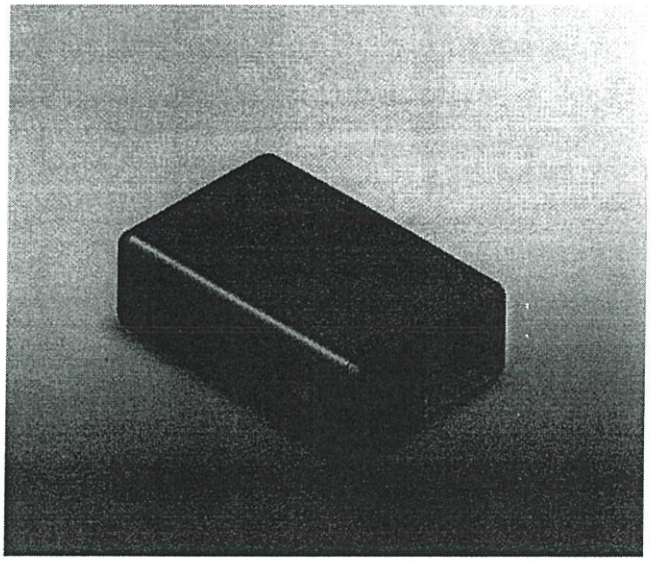
โดยโปรแกรมที่ใช้ในการรับข้อมูลที่ส่งมายังเว็บเซิร์ฟเวอร์จะรับโดยผ่านไฟล์ watt.php ซึ่งไฟล์นี้จะบันทึกค่าทางไฟฟ้าต่างๆที่ถูกส่งเข้ามาลงในฐานข้อมูล โดยข้อมูลทั้งหมดจะถูกส่งอินเทอร์เน็ตเข้าไปเก็บไว้ในตารางข้อมูล ซึ่งแสดงการทำงานได้ดังโฟลว์ชาร์ตในรูปที่ 3.10



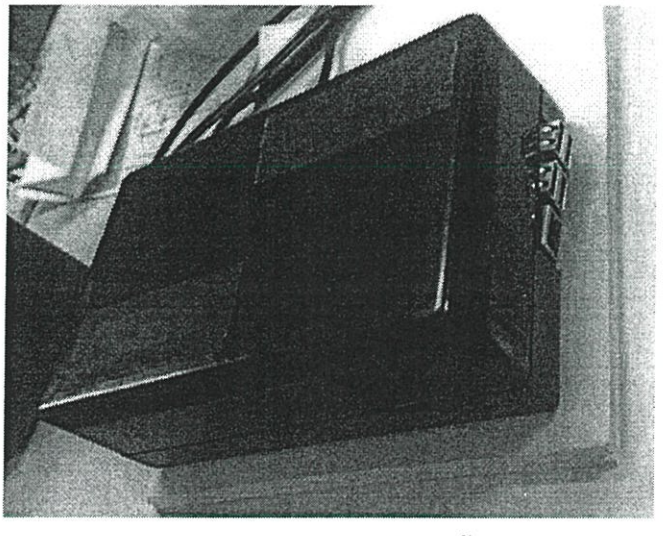
รูปที่ 3.10 โฟล์วชาร์ตขั้นตอนการทำงานของโปรแกรมที่ใช้รับข้อมูลมายังเซิร์ฟเวอร์

3.2 การออกแบบเครื่อง

ในการออกแบบ model เครื่องวัดกำลังไฟฟ้าสำหรับอาคารโดยใช้สมองกลฝังตัว ได้เลือกใช้วัสดุเป็นพลาสติกสีดำ โดยใช้โปรแกรม solidwork ในการออกแบบ



รูปที่ 3.11 model ที่ได้ออกแบบ



รูปที่ 3.12 model ที่สร้างขึ้น

3.3 เครื่องมือที่ใช้ในการทดลอง

1. Raspberry Pi
2. ADE7763
3. TZ2L9
4. คอมพิวเตอร์
5.หม้อแปลง
6. Digital Oscilloscope
7. มัลติมิเตอร์
8.Clamp Meter
9. Kilowatt Hour Meter
10. ADuM1401

3.4 การจัดเก็บผลการทดลอง

ในการจัดเก็บผลการทดลองจะแบ่งออกเป็น 3 ส่วนหลักๆ ซึ่งจะประกอบไปด้วยส่วนของ ADE7763 ส่วนของราสเบอร์รี่ไพ และในส่วนสุดท้ายจะเป็นส่วนของฐานข้อมูล

3.3.1 ส่วนของ ADE7763

การเก็บผลการทดลองในส่วนนี้ จะทำการทดสอบวัดสัญญาณแรงดันขาเข้า จะต้องมีค่าไม่เกิน 0.5 Vrms และทำการทดสอบการติดต่อระหว่าง ADE7763 กับ Raspberry piผ่านทางการเชื่อมต่อแบบ SPI ด้วยการส่งสัญญาณแบบ Loopback ด้วย Digital Oscilloscope

3.3.2 ส่วนราสเบอร์รี่ไพ

การเก็บผลการทดลองในส่วนราสเบอร์รี่ไพ จะเป็นการทดสอบการวัดค่าแรงดันไฟฟ้า กระแสไฟฟ้า และกำลังไฟฟ้าโดยวัดเทียบกับมัลติมิเตอร์ Clamp Meter และ Kilowatt Hour Meter ตามลำดับ

3.3.3 ส่วนฐานข้อมูล

การเก็บผลการทดลองในส่วนนี้ทำได้โดย เปิดเซิร์ฟเวอร์เพื่อรอรับค่าข้อมูลที่ Raspberry Pi ส่งข้อมูลเข้ามาแล้วตรวจสอบว่าค่าที่ส่งเข้ามานั้นสอดคล้องกับค่าที่อ่านได้หรือไม่

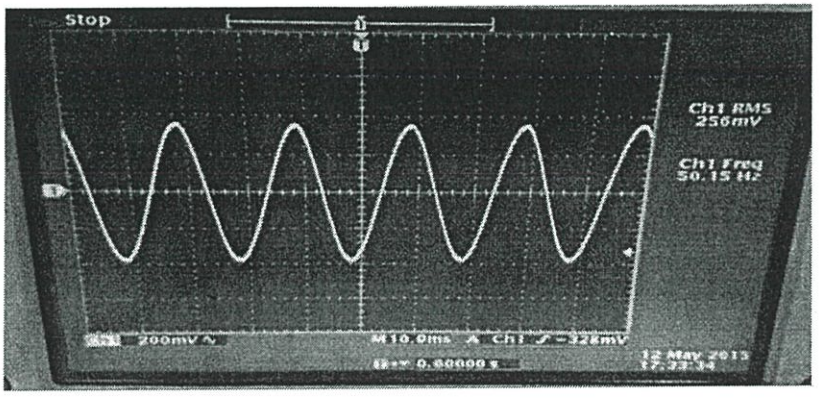
บทที่ 4

ผลการทดลอง

4.1 ผลของสัญญาณเอาท์พุตเมื่อทำการวัดค่าต่างๆจากวงจร

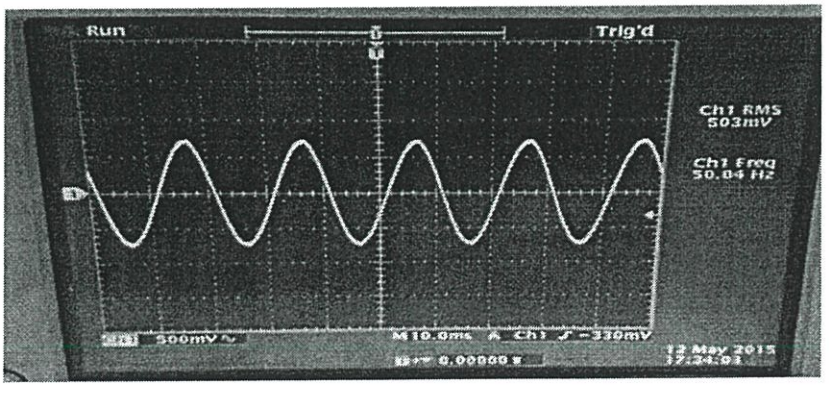
4.1.1 สัญญาณแรงดันขาเข้า

จากการออกแบบวงจรแบ่งแรงดันจาก 220 VAC ไหลผ่านวงจรแบ่งแรงดันจะให้แรงดันเอาท์พุทประมาณ 256 mV แสดงได้ดังต่อไปนี้



รูปที่ 4.1 สัญญาณเอาท์พุทจากวงจรแบ่งแรงดัน

และเมื่อวัดคร่อมทั้งสองโหนดจะได้แรงดัน 220 VAC ไหลผ่านวงจรให้แรงดันเอาท์พุทประมาณ 503 mV แสดงได้ดังต่อไปนี้



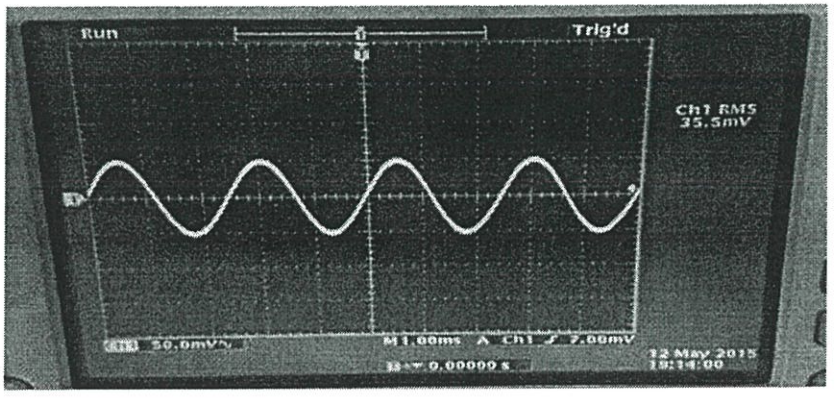
รูปที่ 4.2 สัญญาณเอาท์พุทจากวงจรแบ่งแรงดัน

4.1.2 สัญญาณแรงดันหลังจากผ่าน Current Transformer

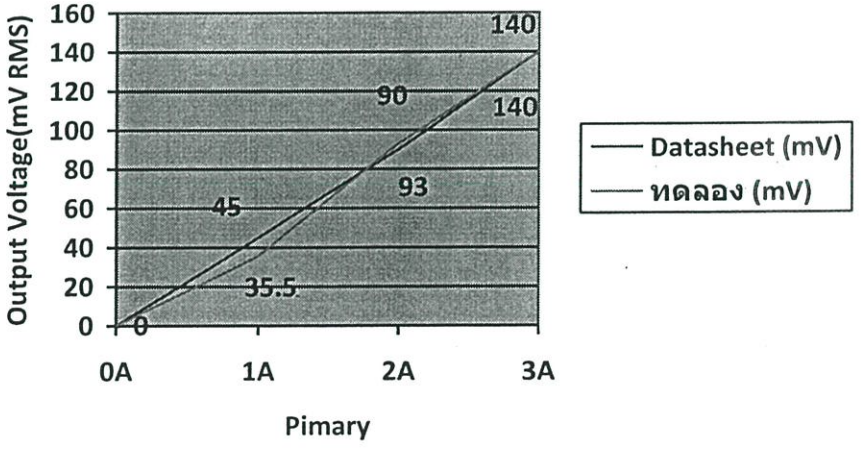
จากรูปที่ 3.3 เมื่อใช้ตัวต้านทาน 20, 60, 100, 200, 300 และ 600 โอห์ม ต่อขนานกับเอาท์พุทของ Current Transformer โดยแสดงการต่อวงจรได้ดังรูปที่ 3.4 จะสามารถให้แรงดันขาออกมีค่าต่างกัน

4.1.2.1 เมื่อใช้ตัวต้านทานขนาด 20 โอห์ม

จากกราฟเมื่อใช้ตัวต้านทาน 20 โอห์มจะให้แรงดันประมาณ 40 mV และจากผลการทดลองจะให้ค่าแรงดันเอาท์พุทเป็น 35.5 mV



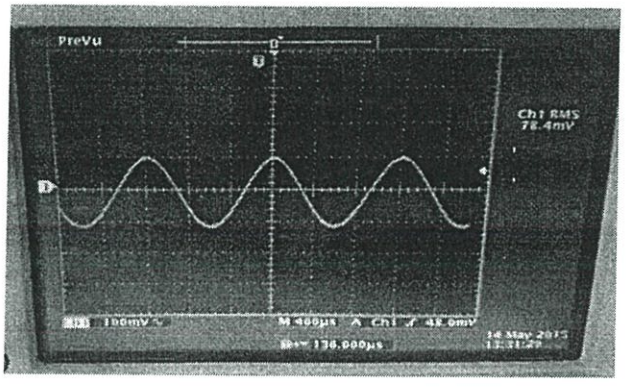
รูปที่ 4.3 สัญญาณแรงดันเอาท์พุทเมื่อใช้ตัวต้านทานขนาด 20 โอห์ม



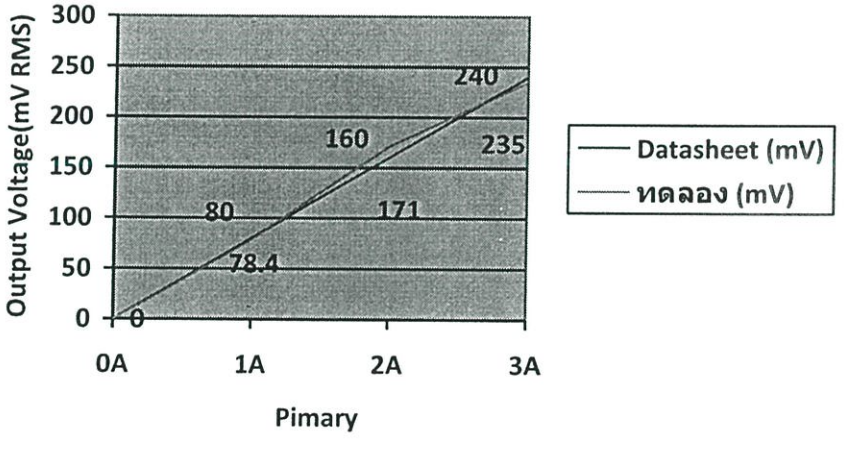
รูปที่ 4.4 กราฟแสดงความสัมพันธ์ระหว่างทฤษฎีกับการทดลองเมื่อใช้ตัวต้านทานขนาด 20 โอห์ม

4.1.2.2 เมื่อใช้ตัวต้านทานขนาด 60 โอห์ม

จากกราฟเมื่อใช้ตัวต้านทาน 60 โอห์มจะให้แรงดันประมาณ 60 mV และจากผลการทดลองจะให้ค่าแรงดันเอาท์พุทเป็น 78.4 mV



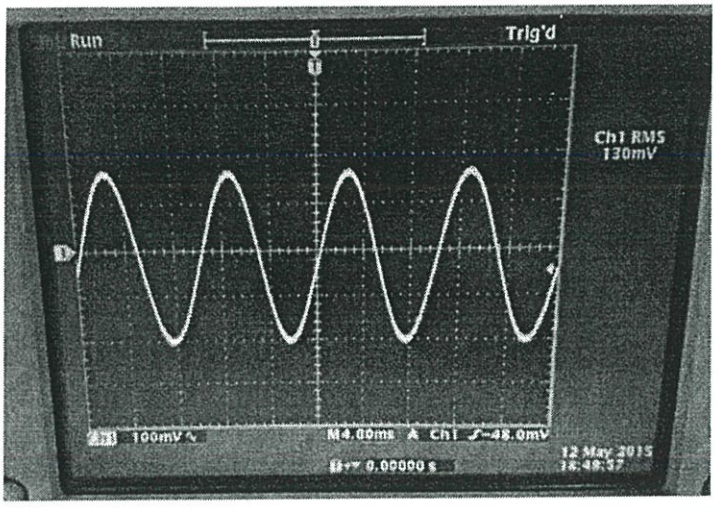
รูปที่ 4.5 สัญญาณแรงดันเอาท์พุทเมื่อใช้ตัวต้านทานขนาด 60 โอห์ม



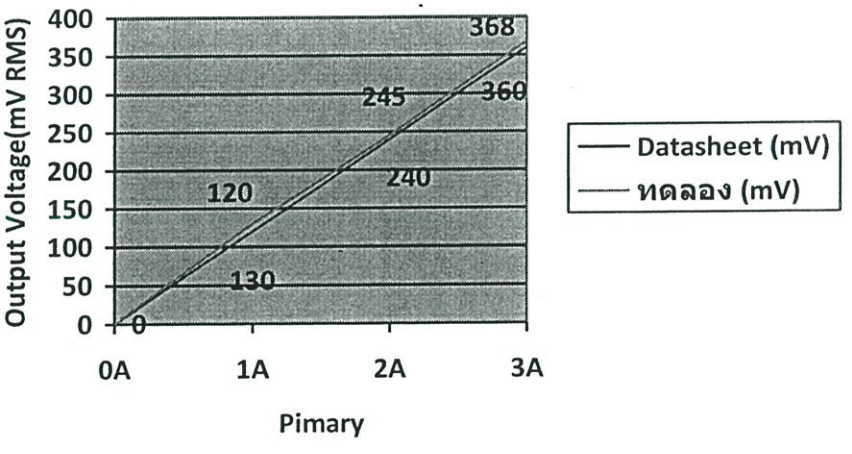
รูปที่ 4.6 กราฟแสดงความสัมพันธ์ระหว่างทฤษฎีกับการทดลองเมื่อใช้ตัวต้านทานขนาด 60 โอห์ม

4.1.2.3 เมื่อใช้ตัวต้านทานขนาด 100 โอห์ม

จากกราฟเมื่อใช้ตัวต้านทาน 100 โอห์มจะให้แรงดันประมาณ 120 mV และจากผลการทดลองจะให้ค่าแรงดันเอาท์พุทเป็น 130 mV



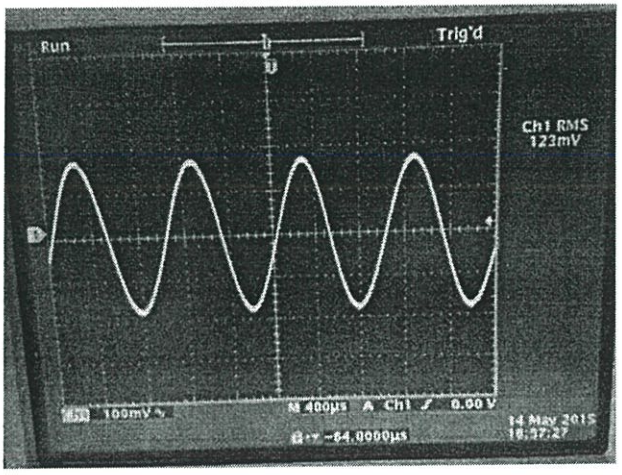
รูปที่ 4.7 สัญญาณแรงดันเอาท์พุทเมื่อใช้ตัวต้านทานขนาด 100 โอห์ม



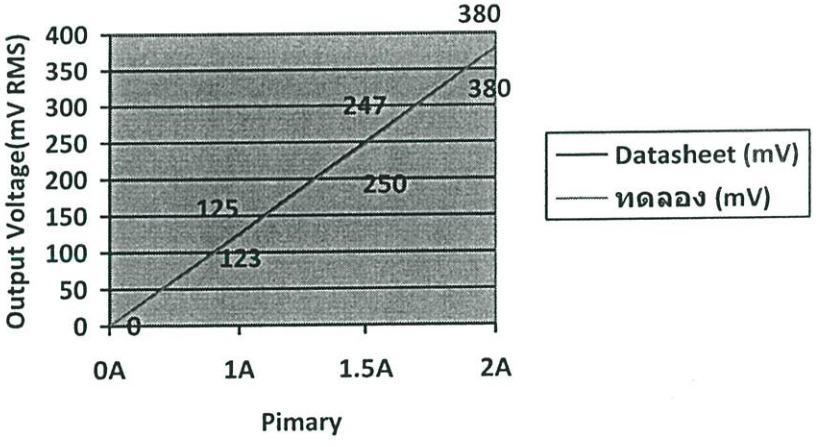
รูปที่ 4.8 กราฟแสดงความสัมพันธ์ระหว่างทฤษฎีกับการทดลองเมื่อใช้ตัวต้านทานขนาด 100 โอห์ม

4.1.2.4 เมื่อใช้ตัวต้านทานขนาด 200 โอห์ม

จากกราฟเมื่อใช้ตัวต้านทาน 200 โอห์มจะให้แรงดันประมาณ 140 mV และจากผลการทดลองจะให้ค่าแรงดันเอาท์พุทเป็น 123 mV



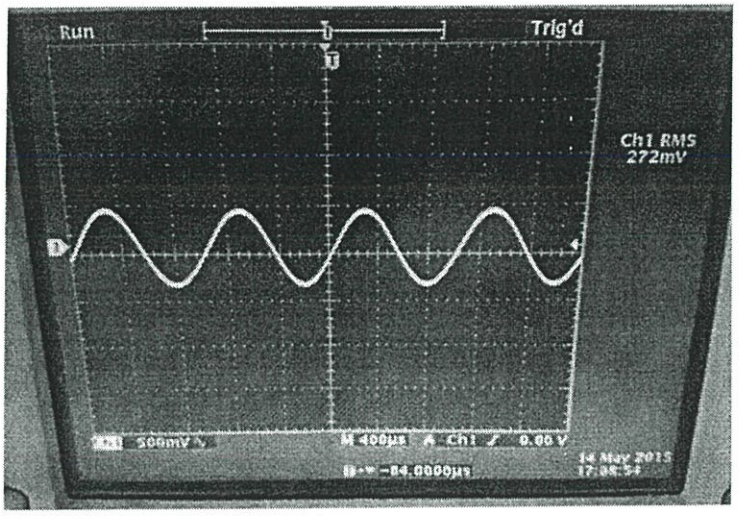
รูปที่ 4.9 สัญญาณแรงดันเอาท์พุทเมื่อใช้ตัวต้านทานขนาด 200 โอห์ม



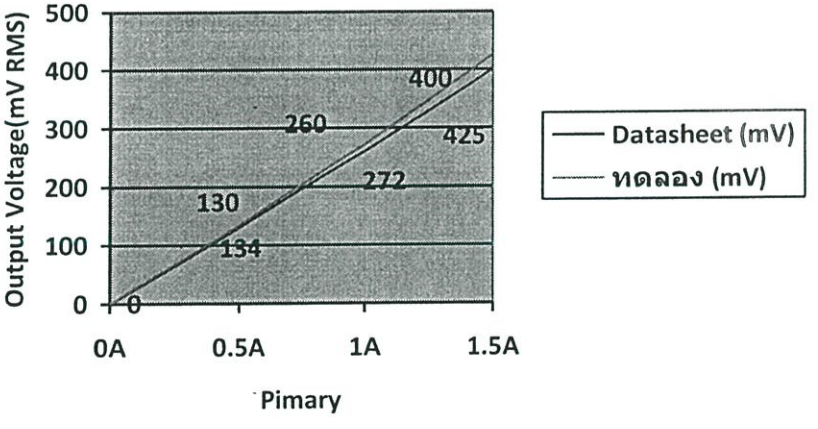
รูปที่ 4.10 กราฟแสดงความสัมพันธ์ระหว่างทฤษฎีกับการทดลองเมื่อใช้ตัวต้านทานขนาด 200 โอห์ม

4.1.2.5 เมื่อใช้ตัวต้านทานขนาด 300 โอห์ม

จากกราฟเมื่อใช้ตัวต้านทาน 300 โอห์มจะให้แรงดันประมาณ 270 mV และจากผลการทดลองจะให้ค่าแรงดันเอาท์พุทเป็น 272 mV



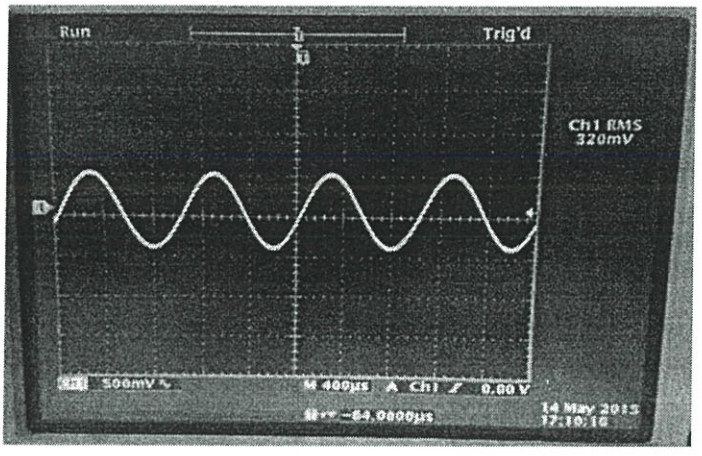
รูปที่ 4.11 สัญญาณแรงดันเอาท์พุทเมื่อใช้ตัวต้านทานขนาด 300 โอห์ม



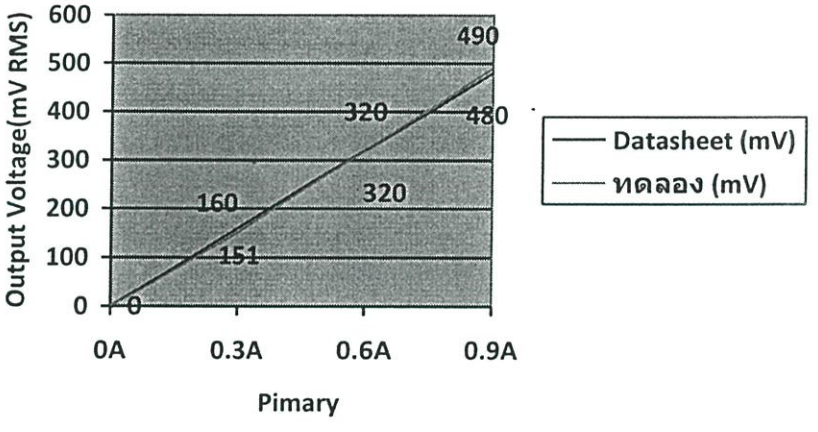
รูปที่ 4.12 กราฟแสดงความสัมพันธ์ระหว่างทฤษฎีกับการทดลองเมื่อใช้ตัวต้านทานขนาด 300 โอห์ม

4.1.2.6 เมื่อใช้ตัวต้านทานขนาด 600 โอห์ม

จากกราฟเมื่อใช้ตัวต้านทาน 600 โอห์มจะให้แรงดันประมาณ 500 mV และจากผลการทดลองจะให้ค่าแรงดันเอาท์พุทเป็น 489 mV



รูปที่ 4.13 สัญญาณแรงดันเอาท์พุทเมื่อใช้ตัวต้านทานขนาด 600 โอห์ม

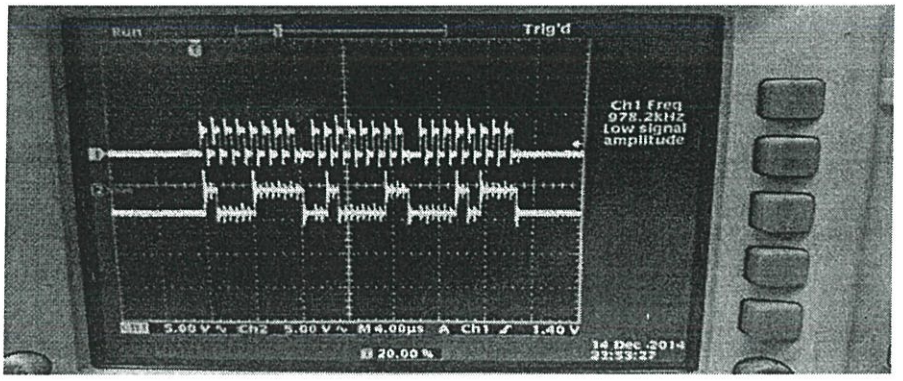


รูปที่ 4.14 กราฟแสดงความสัมพันธ์ระหว่างทฤษฎีกับการทดลองเมื่อใช้ตัวต้านทานขนาด 600 โอห์ม

4.1.3 สัญญาณทดสอบการเชื่อมต่อแบบ SPI

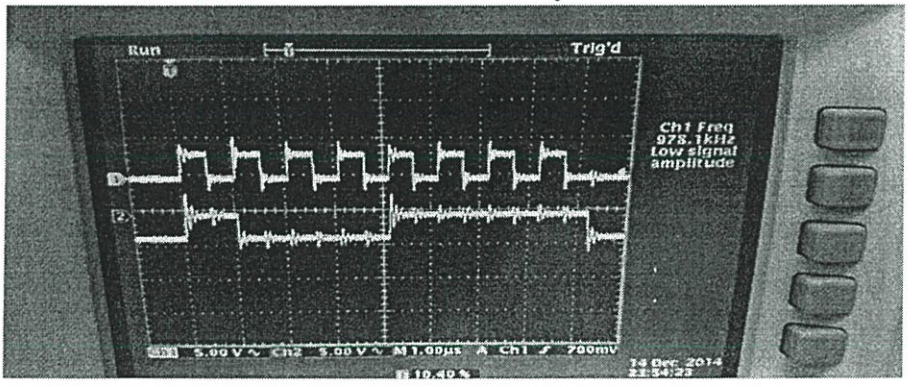
การทดสอบทำได้โดยตั้งค่าความถี่สำหรับ SCK (SPI clock speed) ให้เป็น 1MHz จะทำการส่งข้อมูลชุดละ 3 ไบต์ (0x55, 0x03, 0x11 ตามลำดับ) ในแต่ละไบต์จะส่งบิต MSB ออกไปก่อน (MSB First) และทำขั้นตอนซ้ำโดยเว้นระยะเวลา 0.1 วินาที ถ้าต่อขาสัญญาณ MOSI และ MISO เข้าด้วยกันโดยใช้ Jumper Wire ข้อมูลไบต์ที่ได้รับจากพอร์ต SPI จะได้ค่าเท่ากับข้อมูลไบต์ที่ถูกส่งออกไป (เป็นการทดลองส่ง-รับข้อมูลผ่าน SPI แบบ 'loopback')

3.4.3.1 วัดสัญญาณ SCK เทียบกับ สัญญาณข้อมูลที่ถูกส่งออกไปและรับเข้ามา แสดงได้ดังนี้



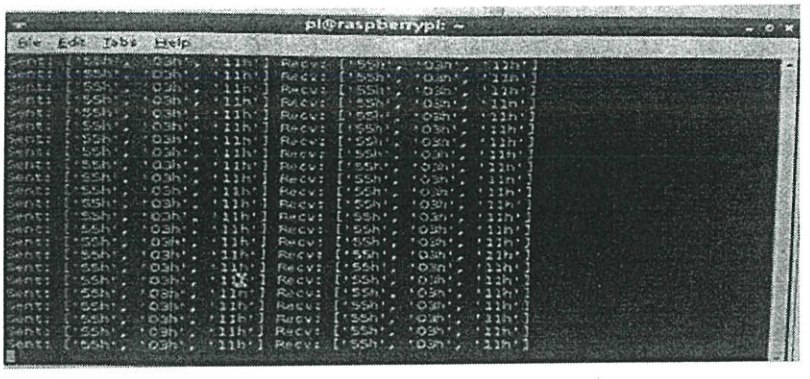
รูปที่ 4.15 สัญญาณ SCK (1) และ MOSI (2) ซึ่งมีข้อมูลไบต์ 0x55, 0x03, 0x11 ตามลำดับโดยมีความถี่เท่ากับ 1.00 MHz

3.4.3.2 แสดงสัญญาณ SCK เทียบกับสัญญาณข้อมูล 0x55



รูปที่ 4.16 สัญญาณ SCK (1) และ MOSI (2) ซึ่งมีข้อมูลไบต์ 0x55 โดยมีความถี่เท่ากับ 1.00 MHz

3.4.3.3 หน้าจอการทำงานของโปรแกรม เมื่อส่งข้อมูลออกไป และรับเข้ามาจะแสดงได้ดังนี้



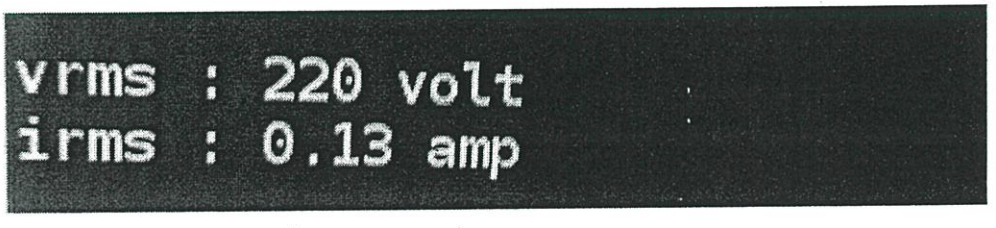
รูปที่ 4.17 หน้าจอแสดงโปรแกรมการรับส่งข้อมูลแบบ Loopback

4.2 ผลการทดลองในส่วนของ Raspberry Pi

การจัดเก็บผลการทดลองในส่วนนี้ จะเป็นการทดสอบการวัดค่าแรงดันไฟฟ้า กระแสไฟฟ้า และ กำลังไฟฟ้า โดยวัดเทียบกับมัลติมิเตอร์ Clamp Meter และ Kilowatt Hour Meter ตามลำดับ โดยมีตัวภาระเป็นเตารีดและคอมพิวเตอร์

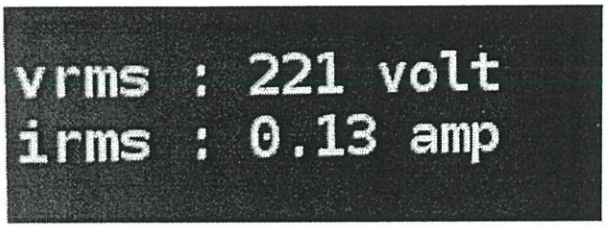
4.2.1 การวัดแรงดันไฟฟ้า

ในการทดลองส่วนนี้จะทำการทดสอบโดยจ่ายไฟให้กับตัวภาระแล้วทำการรันโปรแกรมใน Raspberry Pi จากนั้น Raspberry Pi จะแสดงผลค่าแรงดันไฟฟ้าผ่านทางหน้าจอซึ่งโดยปกติแล้วแรงดันที่วัดได้ควรจะมีค่า 220 V แสดงได้ดังนี้



รูปที่ 4.18 แรงดันที่วัดได้จาก Raspberry Pi

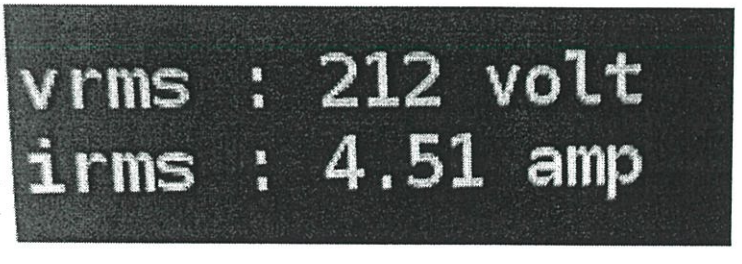
รูปที่ 4.19 แรงดันที่วัดได้จากมัลติมิเตอร์



รูปที่ 4.20 แรงดันที่วัดได้จาก Raspberry Pi มีค่า 221 V

รูปที่ 4.21 แรงดันที่วัดได้จากมัลติมิเตอร์

เมื่อเตารีดทำงานทำให้ใช้กระแสเยอะขึ้นจึงทำให้แรงดันที่วัดได้มีค่าต่ำลงแสดงได้ดังรูปที่ 4.16 และ 4.17



รูปที่ 4.22 แรงดันที่วัดได้จาก Raspberry Pi เมื่อเตารีดทำงาน

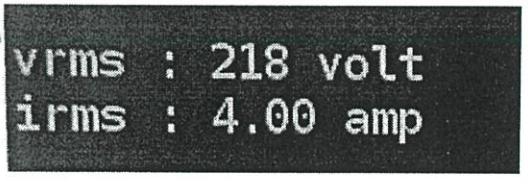
รูปที่ 4.23 แรงดันที่วัดได้จากมัลติมิเตอร์

4.2.2 การวัดกระแสไฟฟ้า

ในส่วนการวัดกระแสไฟฟ้าจะใช้ตัวภาระเป็นพัดลมและเตารีด เนื่องจากพัดลมทำงานตลอดเวลา จึงทำให้กระแสที่วัดได้มีค่าประมาณ 0.2 A และเมื่อเตารีดทำงาน กระแสที่วัดได้จะมีค่าประมาณ 4.0 A จากนั้นรันโปรแกรมเพื่อดูค่ากระแสเทียบกับ Clamp Meter

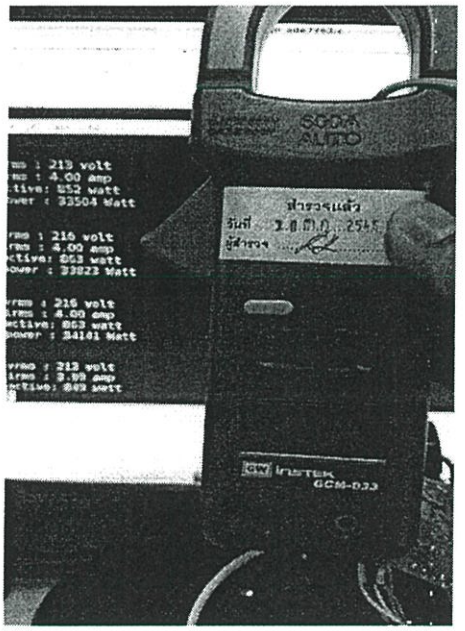
รูปที่ 4.24 กระแสที่วัดได้จาก Raspberry Pi มีค่า 0.21 A

รูปที่ 4.25 กระแสที่วัดได้จาก Clamp Meter มีค่า 0.2 A



รูปที่ 4.26 กระแสที่วัดได้จาก Raspberry Pi เมื่อเตารีดทำงาน มีค่า 4.0 A

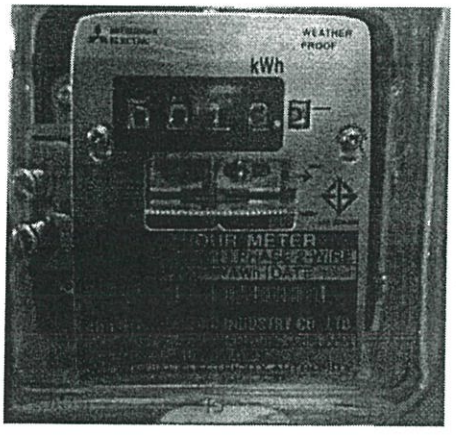
รูปที่ 4.27 กระแสที่วัดได้จาก Clamp Meter เมื่อเตารีดทำงาน มีค่า 4.0 A



รูปที่ 4.28 กระแสที่วัดได้จาก Clamp Meter เมื่อเทียบกับหน้าจอแสดงผลของ Raspberry Pi

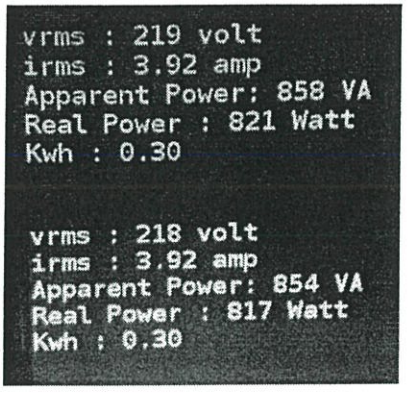
4.2.3 ส่วนการวัดกำลังไฟฟ้า

ในการจัดเก็บผลการทดลองในส่วนนี้จะทำการเก็บผลการทดลองวัดค่ากำลังไฟฟ้าที่ใช้ไปเทียบกับ Kilowatt Hour Meter ที่ใช้จริงภายในอาคารและหอพัก โดยค่าเริ่มต้นของ Kilowatt Hour Meter จะเป็น 13.60 เนื่องจากมีการใช้งานมาบ้างแล้ว แสดงผลการทดลองได้ดังนี้

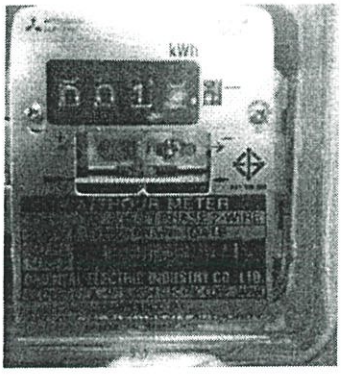


รูปที่ 4.29 ค่าเริ่มต้นของ Kilowatt Hour Meter มีค่าเป็น 13.60

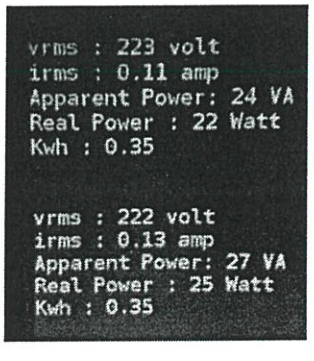
รูปที่ 4.30 เมื่อใช้กำลังไฟฟ้าไป 0.30 หน่วย ค่าที่มิเตอร์วัดได้เป็น 13.90



รูปที่ 4.31 ค่ากำลังไฟฟ้าที่วัดได้ใน Raspberry Pi มีค่า 0.30



รูปที่ 4.32 เมื่อใช้กำลังไฟฟ้าไป 0.35 หน่วย ค่าที่มิเตอร์วัดได้เป็น 13.95



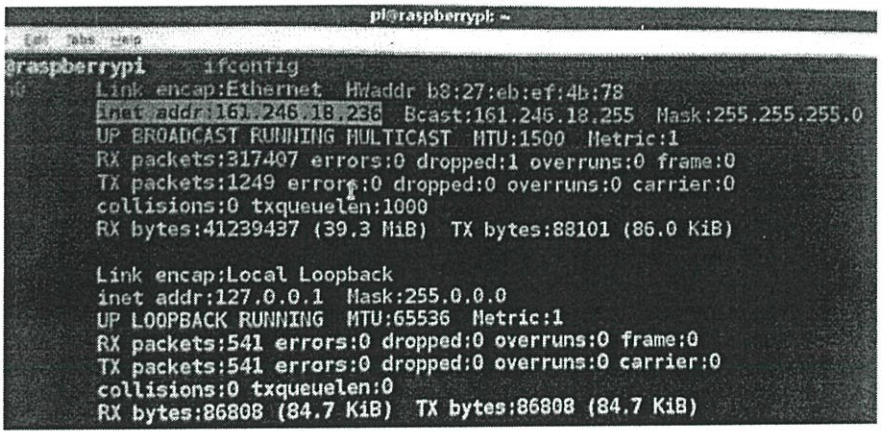
รูปที่ 4.33 ค่ากำลังไฟฟ้าที่วัดได้ใน Raspberry Pi มีค่า 0.35

4.3 ส่วนของฐานข้อมูล

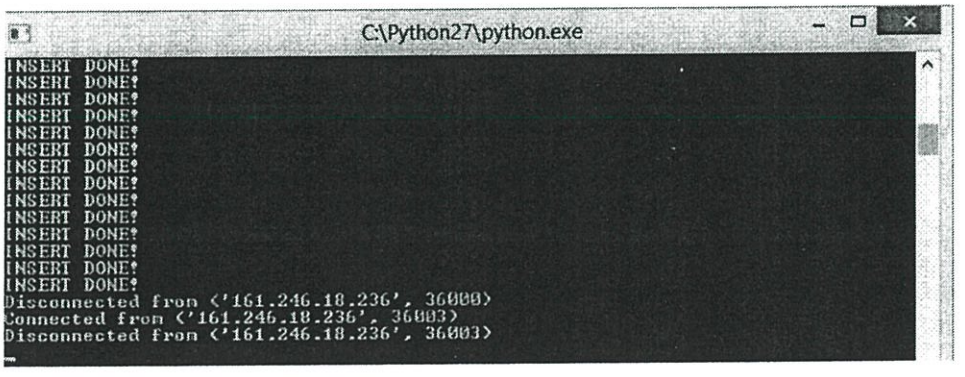
การจัดเก็บผลการทดลองในส่วนของฐานข้อมูลนี้ ทำได้โดยตรวจสอบการเชื่อมต่อว่ามีการเชื่อมต่อมาที่เซิร์ฟเวอร์หรือไม่ จากนั้นทำการตรวจสอบค่าที่ถูกส่งมาจาก Raspberry Pi ว่ามีค่าถูกต้องตามที่ Raspberry Pi แสดงผลหรือไม่ จากนั้นแสดงกราฟของข้อมูลที่เก็บไว้ในฐานข้อมูล

4.3.1 ตรวจสอบการเชื่อมต่อของอุปกรณ์

เมื่อ Raspberry Pi มีการเชื่อมต่อเข้ากับเซิร์ฟเวอร์ หน้าจอเซิร์ฟเวอร์จะมีแสดงผลทางหน้าจอว่า มีผู้เข้ามาเชื่อมต่อโดยสามารถแสดง IP Address ของผู้ที่เข้ามาทำการเชื่อมต่อด้วย แสดงได้ดังรูปที่ 4.28 และ 4.29



รูปที่ 4.34 IP Address ของ Raspberry Pi มีค่าเป็น 161.246.18.236



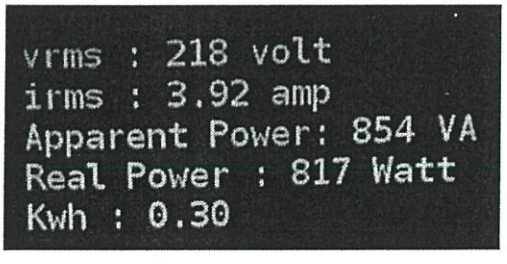
รูปที่ 4.35 แสดง IP ของ Raspberry Pi บนหน้าจอแสดงผลของเซิร์ฟเวอร์ เมื่อมีการเชื่อมต่อและยกเลิกการเชื่อมต่อ

4.3.2 ตรวจสอบค่าที่ถูกส่งมายังฐานข้อมูล

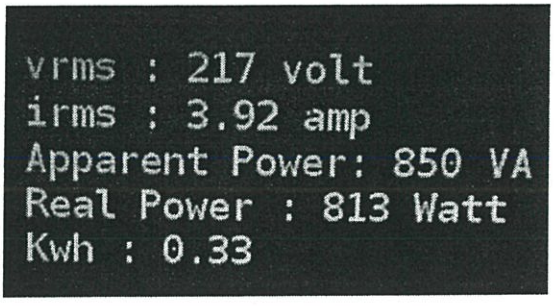
ข้อมูลที่ถูกส่งเข้ามายังฐานข้อมูล จะถูกส่งมาทุก 1 นาที เมื่อข้อมูลถูกส่งเข้ามายังฐานข้อมูลแล้ว ข้อมูลทั้งหมดที่ถูกส่งมาจะถูกเก็บไว้ในตัวแปรที่ถูกสร้างไว้ก่อนหน้าโดย ค่าแรงดันจะถูกเก็บไว้ในตัวแปร Vrms ค่ากระแสจะถูกเก็บไว้ในตัวแปร Irms และ ค่ากำลังไฟฟ้าจะถูกเก็บไว้ในตัวแปร P แสดงได้ดังรูปที่ 4.30

date	Vrms	Irms	P
2015-04-16 23:49:54	222	0.13	0.35
2015-04-16 23:48:54	224	0.13	0.34
2015-04-16 23:47:54	221	0.12	0.33
2015-04-16 23:46:53	217	3.92	0.33
2015-04-16 23:45:53	223	0.14	0.32
2015-04-16 23:44:52	222	0.14	0.31
2015-04-16 23:43:52	218	3.92	0.3
2015-04-16 23:42:51	222	0.12	0.3
2015-04-16 23:41:51	224	0.12	0.29
2015-04-16 23:40:51	218	3.93	0.28
2015-04-16 23:39:51	223	0.13	0.27
2015-04-16 23:38:50	217	3.93	0.26
2015-04-16 23:37:50	216	3.93	0.25
2015-04-16 23:36:49	216	3.93	0.25
2015-04-16 23:35:49	216	3.93	0.24
2015-04-16 23:34:48	216	3.94	0.23
2015-04-16 23:33:48	216	3.94	0.22
2015-04-16 23:32:48	220	3.95	0.21
2015-04-16 23:31:48	222	0.14	0.21
2015-04-16 23:30:47	217	3.91	0.2
2015-04-16 23:29:47	223	0.12	0.19

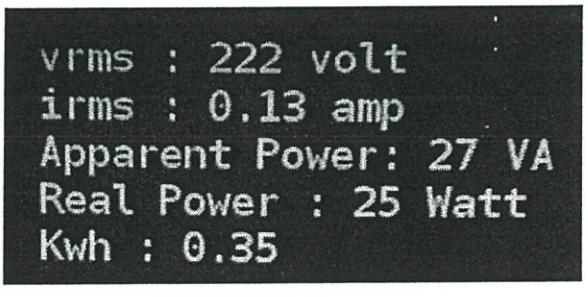
รูปที่ 4.36 ข้อมูลทั้งหมดที่ถูกส่งเข้ามายังฐานข้อมูล



รูปที่ 4.37 ค่าแรงดัน กระแส และกำลัง ที่อ่านได้จาก Raspberry Pi



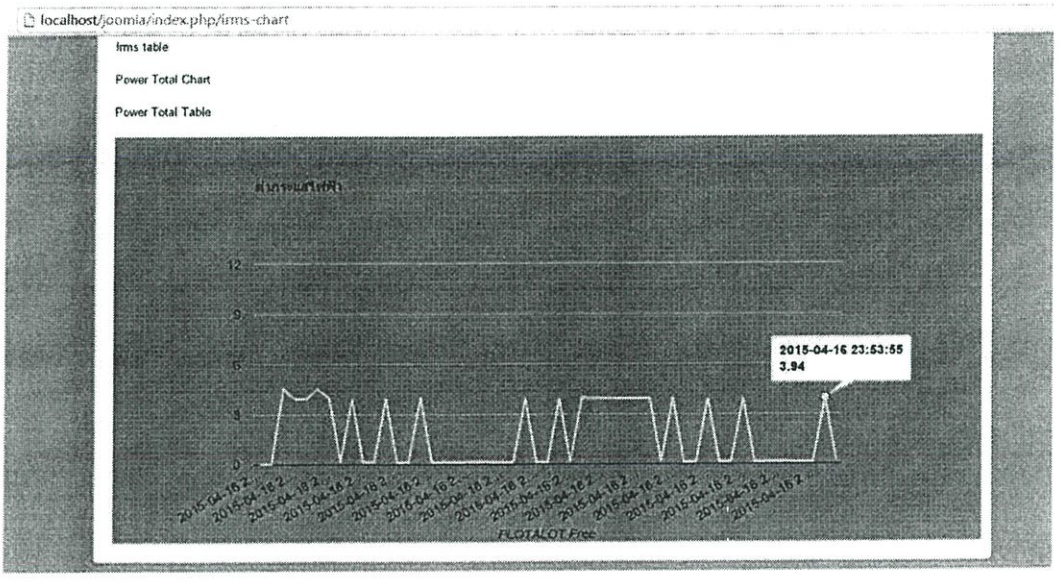
รูปที่ 4.38 ค่าแรงดัน กระแส และกำลัง ที่อ่านได้จาก Raspberry Pi



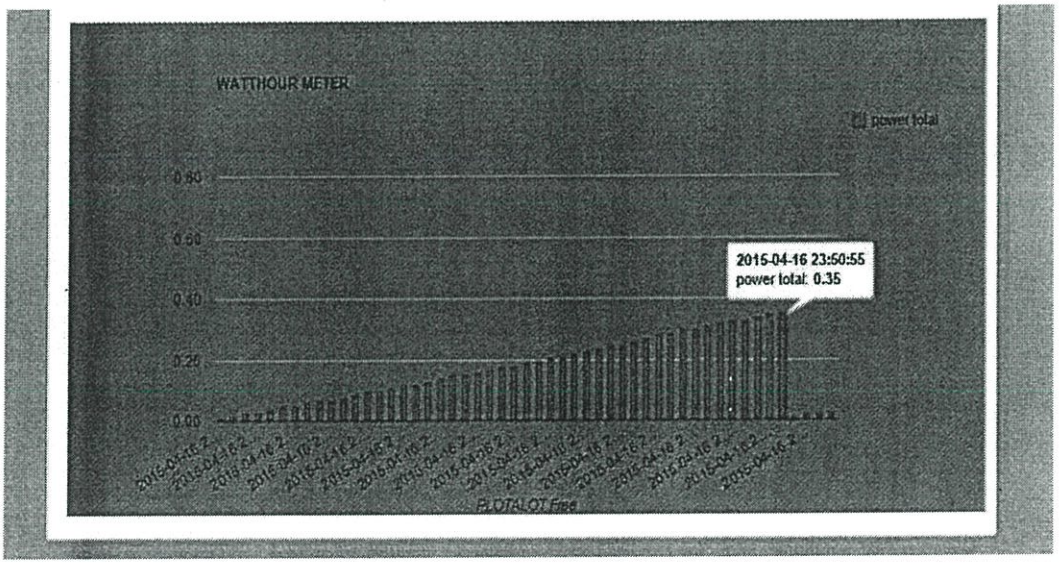
รูปที่ 4.39 ค่าแรงดัน กระแส และกำลัง ที่อ่านได้จาก Raspberry Pi

4.3.3 กราฟแสดงค่าของข้อมูล

เมื่อข้อมูลถูกส่งเข้ามาเก็บยังฐานข้อมูลแล้ว ข้อมูลเหล่านั้นจะถูกนำไปพล็อตกราฟเทียบกับเวลา เพื่อแสดงค่าผ่านทางหน้าเว็บเซิร์ฟเวอร์ เพื่อให้ผู้ใช้สามารถเรียกดูข้อมูลกระแส และกำลังไฟฟ้าที่ใช้ไปได้สะดวกยิ่งขึ้น แสดงได้ดังต่อไปนี้



รูปที่ 4.40 กราฟความสัมพันธ์ระหว่างกระแสกับเวลา



รูปที่ 4.41 กราฟความสัมพันธ์ระหว่างกำลังไฟฟ้ากับเวลา

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

โครงงานนี้เป็นโครงงานเครื่องวัดกำลังไฟฟ้าสำหรับอาคารและหอพักโดยสมองกลฝังตัว (Digital Real-time Watt Hour Meter for Apartment based on Raspberry Pi) เป็นเครื่องมือการวัดพลังงานไฟฟ้าภายในบ้าน ที่สามารถคำนวณค่าทางไฟฟ้าต่างๆ อาทิเช่น กระแส แรงดัน และกำลังไฟฟ้าต่างๆ การวัดพลังงานไฟฟ้านี้จะอำนวยความสะดวกสบายของผู้ใช้ในการเฝ้าดูค่าพารามิเตอร์ของพลังงานไฟฟ้า ซึ่ง เป็นการนำเอา Raspberry Pi มาประยุกต์เป็นดิจิตอลวัตต์มิเตอร์เพื่อคำนวณหาค่ากำลังไฟฟ้าและ Raspberry Pi จะส่งข้อมูลไปยังเซิร์ฟเวอร์เก็บข้อมูลและแสดงค่าต่างๆได้อย่างถูกต้อง

5.2 ข้อเสนอแนะ

ในการต่อวงจรวัดแรงดันและกระแส สำหรับอุปกรณ์ ADE7763 นั้น ควรระวังไม่ให้แรงดันขาออกต้องมีค่าไม่เกิน 0.5 Vrms เนื่องจากจะทำให้อุปกรณ์เสียหายได้ และในการต่อโหลดถ้าต่อโหลดมากจะทำให้แรงดันตกลงได้

บรรณานุกรม

[1] "Raspberry Pi Model B+"
http://www.element14.com/community/community/raspberry-pi/raspberry-pi-bplus/blog/2014/07/14/welcome-to-the-b

[2] "WHATS A RASPBERRY PI?"
https://celebratelife24x7.wordpress.com/2014/11/02/electronics101raspberry-pi/

[3] "Power IC chip (ADE7763)"
http://webst.st.kmutt.ac.th/~s0212431/Power%20IC%20chip%20(ADE7763).html

[4] "ค่าปริมาณที่เกี่ยวข้องกับไฟฟ้ากระแสสลับ"
https://physicspigpig.wordpress.com/%E0%B8%84%E0%B9%88%E0%B8%B%E0%8%9B%E0%B8%A3%E0%B8%B4%E0%B8%A1%E0%B8%B2%E0%B8%93%E0%B8%97%E0%B8%B5%E0%B9%88%E0%B9%80%E0%B8%B5%E0%B8%81%E0%B8%B5%E0%B9%88%E0%B8%A2%E0%B8%A7%E0%B8%82%E0%B9%89%E0%B8%AD/

[5] "TZ2L9"
http://www.es.co.th/detail.asp?prod=35700093

[6] "POWER MONITOR"
http://kll.engineering-news.org/kllfusion01/articles.php?article_id=20

[7] "LM2575t-5.0 1a 5v Simple Switcher Nsc"
http://www.rapidonline.com/electronic-components/lm2575t-5-0-1a-5v-simple-switcher-nsc-82-0658

[8] บ้านอิเล็กทรอนิกส์ "CRYSTAL OSCILLATOR"
http://www.semi-shop.com/shopping/product_group.php?group=3004

[9] "Electronic3"
http://km.srptc.ac.th/external_links.php?links=50

[10] "มัลติมิเตอร์ (Multimeter)"
http://www.un-sound.com/board/index.php?topic=1102.0

[11] "แคลมป์มิเตอร์ (clamp meter)"
http://www.pballtechno.com/category/4/clampmeter%E0%B9%81%E0%B8%84%E0%B8%A5%E0%B8%A1%E0%B8%9B%E0%B9%8C%E0%B8%A1%E0%B8%B4%E0%B9%80%E0%B8%95%E0%B8%AD%E0%B8%A3%E0%B9%8C

[12] "KiloWatt-Hour Meter"
http://www.pui108diy.com/wp/?p=154

[13] "TOPOLOGY"
http://members.tripod.com/barhoush_2/topology.htm

[14] "HUB"
http://www.com5dow.com/basic-computer/265-hub-%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3.html

[15] "ความรู้เบื้องต้นเกี่ยวกับภาษา Python"
http://python.cmsthailand.com/basic_python.html

[16] Khwanchira Nuanthong. "ภาษา PHP"
http://kuk14331.blogspot.com/2013/01/4.html

[17] ทิพย์สุดา บูระวัตรเดชา.พิชามญช์ คารัง.ประภาพร บุญเหลือง. การประยุกต์ใช้ระบบโครงข่ายโทรศัพท์เคลื่อนที่ในการตรวจและดูแลตู้สินค้าหยอดเหรียญ. ปริญญานิพนธ์วิศวกรรมศาสตรบัณฑิต. สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, 2553.

ภาคผนวก ก

โปรแกรมการคำนวณแรงดัน กระแส และกำลังไฟฟ้า

```
//////////////////////////////////////////////////////////////////////////////
// File: spi_ade7763.c
// Date: 2015-Feb-2
// To build:
//    $ gcc -o spi_ade7763 spi_ade7763.c
// To run:
//    $ ./spi_ade7763 -D /dev/spidev0.0
//////////////////////////////////////////////////////////////////////////////
#include <stdint.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <getopt.h>
#include <fcntl.h>
#include <string.h>
#include <sys/ioctl.h>
#include <linux/types.h>
#include <linux/spi/spidev.h>
#include <time.h>
#define ARRAY_SIZE(a) (sizeof(a) / sizeof(*(a)))

//----- SET SPI MODE -----
//SPI_MODE_0 (0,0) CPOL = 0, CPHA = 0, Clock idle low, data is clocked in on rising
edge, output data (change) on falling edge
//SPI_MODE_1 (0,1) CPOL = 0, CPHA = 1, Clock idle low, data is clocked in on falling
edge, output data (change) on rising edge
//SPI_MODE_2 (1,0) CPOL = 1, CPHA = 0, Clock idle high, data is clocked in on falling
edge, output data (change) on rising edge
//SPI_MODE_3 (1,1) CPOL = 1, CPHA = 1, Clock idle high, data is clocked in on rising,
edge output data (change) on falling edge
// global variables
static const char *device = "/dev/spidev0.0";
//static uint8_t  mode  = 0; // mode 0
static uint8_t  mode  = SPI_CPHA | !SPI_CPOL; // mode 1
//static uint8_t  mode  = !SPI_CPHA | SPI_CPOL; // mode 2
//static uint8_t  mode  = SPI_CPHA | SPI_CPOL; // mode 3
```

```
static uint8_t  bits  = 8;       // 8 bits
static uint32_t speed = 400000; // 400 kHz
static uint16_t delayx = 0;
static float power_temp;
static float power_temp1;
static unsigned long long avp;
static float test[20];
static long double power;
static float temp;
static uint32_t wwatt[3];
static int xx,loop=20;
static float power_old , power_new , power_overflow , power_total;
static float temp_power;
static uint16_t apparent;
static void pabort( const char *s ) {
perror(s);
abort();
}
static void spi_transfer( int fd, uint8_t *wbuf, uint8_t *rbuf, int len ) {
static struct  spi_ioc_transfer tr, *ptr = &tr;
int ret;

memset( rbuf, 0, len );
ptr->tx_buf = (unsigned long)wbuf;
ptr->rx_buf = (unsigned long)rbuf;
ptr->len    = len;
ptr->delay_usecs   = delayx;
ptr->speed_hz      = speed;
ptr->bits_per_word = bits;
ret = ioctl( fd, SPI_IOC_MESSAGE(1), &tr );
if ( ret < 1 ) {
pabort( "Can't send spi message" );
}
}
static void init_spi( int fd ) {
// Test write/read the SPI device...
```

```
// set SPI mode
if ( ioctl( fd, SPI_IOC_WR_MODE, &mode ) == -1 ) {
pabort( "Can't set spi mode" );
}
// get SPI mode
if ( ioctl( fd, SPI_IOC_RD_MODE, &mode ) == -1 ) {
pabort( "Can't get spi mode" );
}
// set bits per word
if ( ioctl( fd, SPI_IOC_WR_BITS_PER_WORD, &bits ) == -1 ) {
pabort( "Can't set bits per word" );
}
// get bits per word
if ( ioctl( fd, SPI_IOC_RD_BITS_PER_WORD, &bits ) == - 1 ) {
pabort( "Can't get bits per word" );
}
// set SPI speed
if ( ioctl( fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed ) == -1 ) {
pabort( "Can't set max speed Hz" );
}
// get SPI speed
if ( ioctl(fd, SPI_IOC_RD_MAX_SPEED_HZ, &speed ) == -1 ) {
pabort( "Can't get max speed Hz" );
}
// Now show SPI mode, bits per word, and max. clock speed
printf( "SPI mode: %d\n", mode );
printf( "bits per word: %d\n", bits );
printf( "Max speed: %d Hz (%d KHz)\n", speed, speed/1000 );
}
char *array_to_str( char *sbuf, const uint8_t *a, int len ) {
int i;
char *p = sbuf;
sprintf( p, "[" );
p++;
for ( i=0; i < len; i++ ) {
sprintf( p, "0x%02X%c", a[i], (i+1 < len) ? ',' : ']' );
```

```
p += 5;
}
return sbuf;
}
int main( int argc, char *argv[] ) {
int ret = 0, i;
int fd=0; // file descriptor for the SPI device
uint8_t volt[5] = { 0x8F, 0x43, 0x17, 0xFF, 0xFF };        //Command For read Register
Volt 0x17
uint8_t amp[5] = { 0x8F, 0x43, 0x16, 0xFF, 0xFF };         //Command For read Register
Amp 0x16
uint8_t watt[6] = { 0x8F, 0x43, 0x05, 0xFF, 0xFF, 0xFF };  //Command For read Register
power 0x05
uint8_t rbuf[5] = { 0, };

char sbuf[64];
uint16_t volt_temp;
uint16_t amp_temp;
uint8_t b=0;
uint32_t vrms;
uint32_t active;

float irms;
float volt_diff = 44.52;
float amp_diff = 3320;
float power_diff = 0.01902;

fd = open( device, O_RDWR ); // open SPI device
if (fd < 0) {
pabort( "Can't open device" );
}
init_spi( fd ); // open and initialize the SPI device
FILE *fp = fopen("data.txt","w");
while (1) {
////////////Open  File to.txt////////////
fp = fopen("data.txt","w");
```

```
if(fp != NULL){
time_t now;
time(&now);
fputs(ctime(&now), fp);
fclose(fp);
}
temp = 0;

//////////////////////Read Volt//////////////////////
for (xx=0;xx<loop;xx++){
array_to_str( sbuf, volt, ARRAY_SIZE(volt) );
//printf( "Sent: %s ", sbuf );
spi_transfer( fd, volt, rbuf, ARRAY_SIZE(volt) );
array_to_str( sbuf, rbuf, ARRAY_SIZE(rbuf) );
//printf( "Recv: %s\n", sbuf );
volt_temp = rbuf[3] << 8;
volt_temp = volt_temp+ rbuf[4];
temp = temp + volt_temp;
//printf( "volt_temp : %u\n", volt_temp );
}
vrms = temp/loop;
vrms = vrms / volt_diff;
printf( "vrms : %u volt \n", vrms );
usleep( 500000 ); // delay time
temp = 0;
////////////Send Volt to.txt////////////
fp = fopen("data.txt","a");
if(fp != NULL){
char str[10];
sprintf(str,"%d",vrms);
fputs(str, fp);
fclose(fp);
}
//////////////////////////////
```

```
//////////////////////Read Current//////////////////////
for (xx=0;xx<loop;xx++){
array_to_str( sbuf, amp, ARRAY_SIZE(amp) );
//     printf( "Sent: %s ", sbuf );
spi_transfer( fd, amp, rbuf, ARRAY_SIZE(amp) );
array_to_str( sbuf, rbuf, ARRAY_SIZE(rbuf) );
//     printf( "Recv: %s\n", sbuf );
amp_temp = rbuf[3] << 8;
amp_temp = amp_temp + rbuf[4];
temp = temp + amp_temp;
//     printf( "amp_temp : %u\n", amp_temp );
}
irms = temp / loop;
irms = irms / amp_diff;
printf( "irms : %.2f amp \n", irms );
active = vrms*irms;
printf( "Active power: %lu VA \n", active);
usleep( 500000 ); // delay time
temp = 0;

////////////Send Ampere to.txt////////////
fp = fopen("data.txt","a");
if(fp != NULL){
char str[10];
sprintf(str,"%.2f",irms);
fputs("\n", fp);
fputs(str, fp);
fclose(fp);
//////////////////////////////
}
//////////////////////Read power//////////////////////
temp_power = 0;
for (xx=0;xx<loop;xx++){
array_to_str( sbuf, watt, ARRAY_SIZE(watt) );
//     printf( "Sent: %s ", sbuf );
spi_transfer( fd, watt, rbuf, ARRAY_SIZE(watt) );
```

```
array_to_str( sbuf, rbuf, ARRAY_SIZE(rbuf) );
//printf( "Recv: %s\n", sbuf );
power_temp = rbuf[3] << 16;
wwatt[0] = power_temp;
power_temp1 = rbuf[4] << 8;
wwatt[1] = power_temp1;
wwatt[2] = rbuf[5];
avp = wwatt[0] + wwatt[1] + wwatt[2];
temp_power = temp_power + avp;
}
power_new = temp_power/loop;
if(power_old > 10000000 && power_new < 5000000)
{
power_overflow++;
}
power_old = power_new;
power_total = power_overflow * 0x1000000 + power_new;
power_total = (power_total * power_diff)/(1000*3600);
printf( "Apparent power : %lu Watt \n", apparent );
printf( "Kwh : %.2f  \n\n\n", power_total );

////////////Send Power to.txt////////////
fp = fopen("data.txt","a");
if(fp != NULL){
char str[10];
sprintf(str,"%.2f",power_total);
fputs("\n", fp);
fputs(str, fp);
fclose(fp);
}
//////////////////////////

usleep( 500000 ); // delay time
}
//  close( fd );
return (temp);}
```

ภาคผนวก ข

โปรแกรมการติดต่อและส่งข้อมูลในส่วน Client

```
import time
from socket import *
import thread
HOST = '161.246.18.155' ////// IP Server ///////
PORT = 21567
BUFSIZE = 1024
ADDR = (HOST, PORT)
tcpCliSock = socket(AF_INET, SOCK_STREAM)
tcpCliSock.connect(ADDR)
while 1:
time.sleep(60)
with open('data.txt') as f:     ////// Open File .txt //////////
lines = f.readlines()
while len(lines)<4:
f = open('data.txt')
lines = f.readlines()
txt = ', '.join(lines)
BUFF = ""+txt
data = tcpCliSock.send(BUFF)
```

ภาคผนวก ค

โปรแกรมการรับข้อมูลในส่วนของ Server

```
import asyncore
import socket
import MySQLdb
import time
clients = {}
class MainServerSocket(asyncore.dispatcher):
    def __init__(self, port):
        asyncore.dispatcher.__init__(self)
        self.create_socket(socket.AF_INET, socket.SOCK_STREAM)
        self.bind(('',port))
        self.listen(5)
    def handle_accept(self):
        newSocket, address = self.accept( )
        clients[address] = newSocket
        print "Connected from", address
        SecondaryServerSocket(newSocket)
class SecondaryServerSocket(asyncore.dispatcher_with_send):
    def handle_read(self):
        receivedData = self.recv(8192)
        #print receivedData
////////////////// Login to database//////////////////////
        db = MySQLdb.connect(host="localhost",
                             user="oak",
                             passwd="123456",
                             db="watt")
//////////////////////ReceivedData//////////////////////
        if receivedData:
            every = clients.values()
            data = receivedData.split(',')
            #print data
            cur = db.cursor()
#get_date = time.strftime("%x %X")
cur.execute("INSERT INTO watt_tab (Vrms,Irms,P) VALUES ("
                +data[1].rstrip('\n')+","+data[2].rstrip('\n')+","+data[3].rstrip('\n')+")")
#cur.execute("INSERT INTO watt_tab (date,Vrms,Ivms,P) VALUES (11,22,33,44)")
db.commit()
```

```
print "INSERT DONE!"
            #for one in every:
            #    one.send(receivedData+'\n')
else: self.close( )
        db.close()
def handle_close(self):
print "Disconnected from", self.getpeername( )
one = self.getpeername( )
del clients[one]
MainServerSocket(21567)
asyncore.loop()
```