ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล
MARINE ENVIRONMENTAL MONITORING SYSTEM

โดย

นางสาวปิยากร ศรีคำรุณ

นางสาวปุณยาพร จันทร์อบ

นางสาวพงษ์นภา ภิรักจรรยากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล
MARINE ENVIRONMENTAL MONITORING SYSTEM

โดย

นางสาวปิยากร ศรีคำรุณ
นางสาวปุณยาพร จันทร์อบ
นางสาวพงษ์นภา ภิรักจรรยากุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2558

ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล
MARINE ENVIRONMENTAL MONITORING SYSTEM

โดย

นางสาวปิยากร	ศรีคำรุณ	55010777
นางสาวปุณยาพร	จันทร์อบ	55010782
นางสาวพงษ์นภา	ภิรักจรรยากุล	55010804

อาจารย์ที่ปรึกษา

รศ.ดร.พิพัฒน์ พรหมมี

ผศ.ดร.มนตรี คำเงิน

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2558

ผ่านการตรวจรูปเล่มแล้ว

(.....)

อาจารย์ที่ปรึกษา

17/05/59

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ผ่านการตรวจชิ้นงานแล้ว

(.....)

กรรมการผู้ตรวจชิ้นงาน

17.5.59.

KMITL วิศวกรรมโทรคมนาคม Telecommunications Engineering

ปริญญานิพนธ์ปีการศึกษา 2558
ภาควิชาวิศวกรรมโทรคมนาคม
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
เรื่อง ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล
MARINE ENVIRONMENTAL MONITORING SYSTEM

ผู้จัดทำ

1. นางสาวปิยากร ศรีคำรุณ 55010777
2. นางสาวปุณยาพร จันทร์อบ 55010782
3. นางสาวพงษ์นภา ภิรักจรรยากุล 55010804

..... อาจารย์ที่ปรึกษา
(รศ.ดร.พิพัฒน์ พรหมมี)

..... อาจารย์ที่ปรึกษา
(ผศ.ดร.มนตรี คำเงิน)

กิตติกรรมประกาศ

รายงานประกอบโครงงานเล่มนี้สำเร็จได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์จาก รศ.ดร.พิพัฒน์ พรหมมี และ ผศ.ดร. มนตรี คำเงิน อาจารย์ที่ให้คำปรึกษาโครงงานที่ได้ให้คำแนะนำที่ดีตลอดมา และเอื้อเฟื้อสถานที่ให้ในการจัดทำโครงงาน ขอขอบพระคุณ คุณดนัย ภิรักจรรยากุล ที่ให้ความอนุเคราะห์ให้คำปรึกษา เอื้อเฟื้ออุปกรณ์และสถานที่ ในการทำตัวทุ่นลอย ขอบคุณบิดามารดาที่คอยให้กำลังใจ ขอบคุณรุ่นพี่และเพื่อนๆ ทุกคนที่ให้คำปรึกษาและคอยให้ความช่วยเหลือ และที่สำคัญที่สุดขอบคุณเพื่อนในกลุ่มที่ได้จัดทำโครงงานนี้ และได้ผ่านความยากลำบากมาด้วยกัน ทำโครงงานนี้สำเร็จลุล่วงไปด้วยดี ท้ายนี้ผู้จัดทำขอขอบคุณทุกๆ ท่านที่มีส่วนเกี่ยวข้องมา ณ โอกาสนี้

นางสาวปิยากร ศรีคำรุณ
นางสาวปุณยาพร จันทร์อบ
นางสาวพงษ์นภา ภิรักจรรยากุล
ผู้จัดทำ

ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล
MARINE ENVIRONMENTAL MONITORING SYSTEM

โดย นางสาวปิยากร ศรีคำรุณ 55010777
นางสาวปุณยาพร จันทร์อบ 55010782
นางสาวพงษ์นภา ภิรักจรรยากุล 55010804

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

บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการสร้างทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล โดยใช้ Arduino Mega 2560 เป็นอุปกรณ์หลักในการควบคุมการทำงาน มีการติดตั้ง GPS Tracking เพื่อระบุตำแหน่งของทุ่น ใช้เซ็นเซอร์ต่างๆตรวจวัดค่าเพื่อเก็บข้อมูล และประยุกต์ใช้เซลล์แสงอาทิตย์เป็นแหล่งจ่ายพลังงานให้กับระบบ ข้อมูลที่ตรวจวัดได้แก่ อุณหภูมิผิวน้ำทะเล ความสูงของระดับน้ำทะเล ความเร็วลม ทิศทางลม อุณหภูมิอากาศ และความกดอากาศ ซึ่งข้อมูลที่ตรวจวัดได้จากทุ่นจะถูกส่งผ่านระบบอินเตอร์เน็ตแบบไร้สายไปจัดเก็บในฐานข้อมูลกลางทุกๆ 3 นาที แสดงผลในรูปแบบตารางบนฐานข้อมูล เพื่อนำข้อมูลที่ได้มาใช้ในการประเมินสถานการณ์ร่วมกับข้อมูลอื่นๆและติดตามผลกระทบด้านสิ่งแวดล้อมทางทะเล

ABSTRACT

This project presents marine environmental monitoring system using Arduino Mega 2560 with GPS tracking to determine the location of the buoy. Different sensors are used to collect measured data and the solar cells are used as the power source for the whole system. The measured data of sea surface temperature, the height of the sea level, wind speed, wind direction, air temperature, humidity and air pressure will be sent over the internet to the database every three minutes. Results are displayed in tabular form on the database which uses to forecast and planning of the environment situation and natural marine impacts. It is ensured that can be announced to the people to prevent the further natural disasters.

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันการพยากรณ์มีบทบาทมากขึ้นในชีวิตประจำวัน นอกจากนี้ยังมีภัยพิบัติที่เกิดขึ้นอย่างไม่ทันตั้งตัว เช่น การเกิดมรสุมทางทะเล อุณหภูมิน้ำเปลี่ยนฉับพลัน ซึ่งอาจเกิดภัยร้ายแรงในท้องทะเลได้ ไม่ว่าจะร้ายแรงมากหรือน้อยก็สามารถเกิดขึ้นได้ทุกเวลา โดยที่มนุษย์ไม่ทันตั้งตัว เราควรจะหันมาตระหนักถึงภัยธรรมชาติต่างๆที่เกิดขึ้น เพื่อเร่งแก้ไขหรือป้องกันให้ทันการ จากโครงงาน เรื่อง ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล เป็นการเก็บค่าข้อมูลต่างๆที่ทำการสำรวจ นั่นคือ อุณหภูมิน้ำ อุณหภูมิอากาศ ความกดอากาศ ความชื้น ละติจูด ลองจิจูด วันที่ เวลา ความเร็วลม และลักษณะคลื่น ข้อมูลทั้งหมดจะถูกส่งผ่านทางระบบอินเทอร์เน็ตไร้สายมายังฐานข้อมูลบนคอมพิวเตอร์ ซึ่งอุปกรณ์ต่างๆ ภายในทุ่นจะใช้แหล่งพลังงานมาจากแผงโซล่าร์เซลล์ที่แปลงพลังงานแสงอาทิตย์เป็นพลังงานไฟฟ้า แล้วนำมาเก็บในแบตเตอร์รี่ขนาดเล็ก ส่วนด้านการประมวลผลใช้ตัว Arduino Mega 2560 เป็นอุปกรณ์หลัก

1.2 วัตถุประสงค์

1) เพื่อศึกษาหลักการทำงานไมโครคอนโทลเลอร์ อุปกรณ์ทางอิเล็กทรอนิกส์ต่างๆ ได้แก่ Arduino Uno, GPS Tracking, Sensor ต่างๆ และ Solar cell

2) เพื่อสร้างสรรค์ผลงานทุ่นสำรวจด้วยอุปกรณ์อิเล็กทรอนิกส์เพื่อนำไปใช้งานในการสำรวจ วัดค่าตามที่ได้ติดตั้ง อุปกรณ์เซ็นเซอร์ ไว้เพื่อนำข้อมูลมาทำการเก็บค่าและประเมินผลในขั้นถัดไป

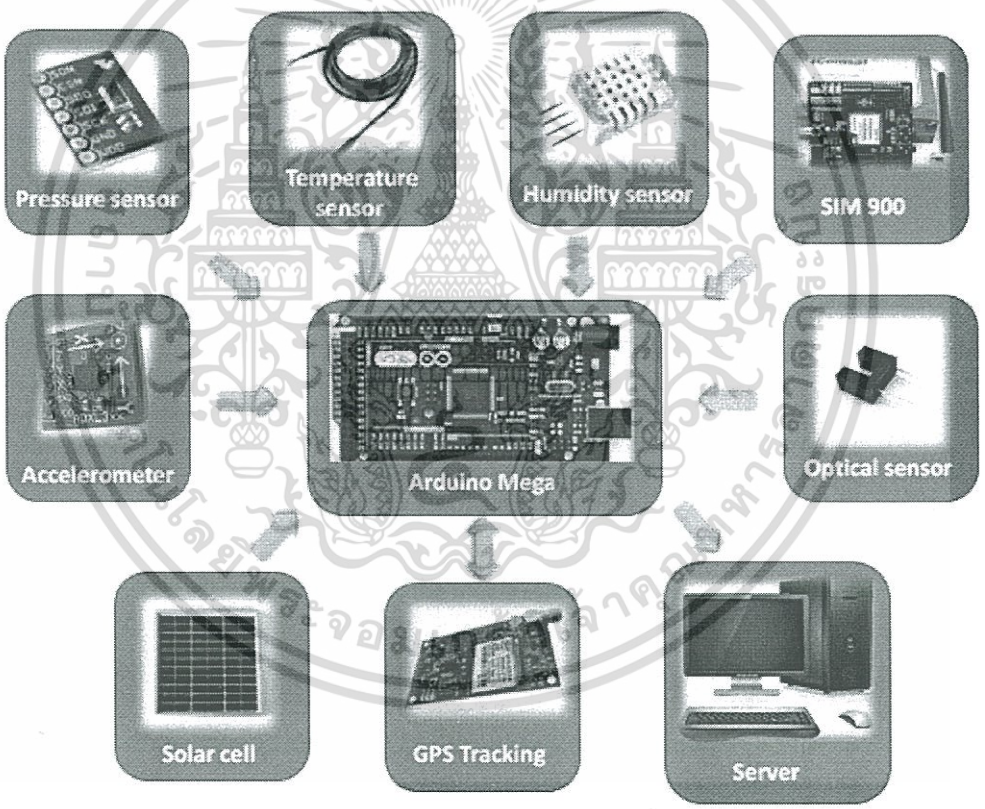
3) เพื่อศึกษาการทำงานในการรับ-ส่งข้อมูลในระยะทางไกล,การระบุตำแหน่งบนพื้นโลก และการใช้พลังงานจากแสงอาทิตย์

4) ข้อมูลจากทุ่นสำรวจสามรถเผยแพร่สู่หน่วยงานที่เกี่ยวข้องทั้งในส่วนสถาบันการศึกษา ภาครัฐและเอกชน และถูกนำไปใช้ประโยชน์ในงานและโครงการต่างๆ เช่น การพยากรณ์สภาพอากาศ การวางแผนจัดการ และติดตาม ผลกระทบทางด้านสิ่งแวดล้อม

5) การติดตามผลกระทบของสภาวะโลกร้อนต่อการเปลี่ยนแปลงของอุณหภูมิน้ำทะเล ตลอดจนเป็นข้อมูล สภาพแวดล้อมทางทะเล เพื่อสนับสนุนด้านกิจกรรมการท่องเที่ยวและการอนุรักษ์

1.3 ขอบเขตของปริญญานิพนธ์

สามารถเก็บค่าข้อมูลต่างๆที่ทำการสำรวจ นั่นคือ อุณหภูมิน้ำ อุณหภูมิอากาศ ความกดอากาศ ความชื้น ละติจูด ลองจิจูด วันเวลา ข้อมูลทั้งหมดจะถูกส่งผ่านทางระบบอินเทอร์เน็ตแบบไร้สายมายังฐานข้อมูลบนคอมพิวเตอร์ นอกจากนี้ อุปกรณ์ต่างๆ ภายในทุ่นใช้แหล่งพลังงานมาจากแผงโซล่าเซลล์ที่แปลงพลังงานจากแสงอาทิตย์มาเป็นพลังงานไฟฟ้า แล้วนำมาเก็บในแบตเตอร์รี่ขนาดเล็ก โดยใช้ Arduino เป็นตัวควบคุมการทำงานหลักไปยังอุปกรณ์ต่างๆ สำหรับในเทอมนี้ได้ทำการติดตั้งตัวเซ็นเซอร์ต่างๆ อันได้แก่ Temperature sensor, Pressure sensor ,Humidity sensor ,Accelerometer ,Optical Sensor , GPS โมดูล, Sim900 เข้ากับ Arduino Mega 2560 พร้อมทั้งใช้ตัวเซ็นเซอร์ในการเก็บค่าข้อมูลจากสภาพแวดล้อม และส่งข้อมูลผ่านทางอินเทอร์เน็ตไร้สายเข้าไปเก็บค่ายัง Server ได้สำเร็จ



รูปที่ 1.1 ภาพรวมของโครงงาน

บทที่ 2

ทฤษฎีและหลักการที่เกี่ยวข้อง

ทุ่นสำรวจข้อมูลสภาพแวดล้อมทางทะเล (Marine Environmental Monitoring System) เป็นการนำเอา Arduino Mega 2560 มาเป็นตัวประมวลผลหลัก โดยเชื่อมต่อกับตัว **Sensor** ต่างๆ และเชื่อมต่อกับ **SIM900 GSM /GPRS Module** เพื่อนำเอาค่าต่างๆที่ได้จาก Sensor ส่งข้อมูลผ่านอินเตอร์เน็ตมายังฐานข้อมูลในคอมพิวเตอร์เพื่อเก็บค่าและสามารถแจ้งเตือนได้ โดยใช้พลังงานที่ได้จากแผงโซล่าร์เซลล์เป็นแหล่งจ่ายไฟให้กับอุปกรณ์ทั้งหมดนี้

2.1 ความรู้เบื้องต้นเกี่ยวกับไมโครคอนโทรลเลอร์ Arduino Mega 2560

2.1.1 ความหมายของไมโครคอนโทรลเลอร์ Arduino Mega 2560

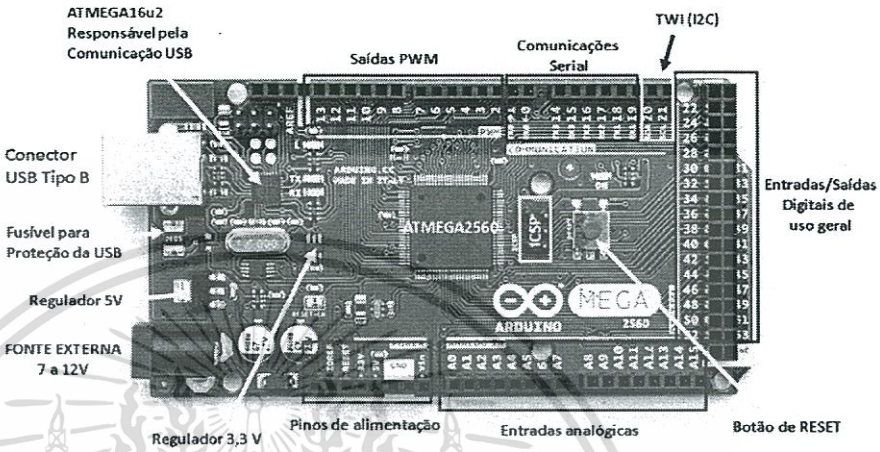
ไมโครคอนโทรลเลอร์ (Microcontroller) เป็นอุปกรณ์อิเล็กทรอนิกส์แบบหนึ่งที่รวมเอา หน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุต วงจรส่ง สัญญาณเอาท์พุต รวมถึงหน่วยความจำวงจรกำเนิดสัญญาณนาฬิกาไว้ด้วยกัน ทำให้สามารถ นำไปใช้งานแทนวงจรอิเล็กทรอนิกส์ที่ซับซ้อนได้เป็นอย่างดี โดยไมโครคอนโทรลเลอร์มาจากคาสองคำรวมกันคือ “ไมโคร” ซึ่งหมายถึง ไมโครโปรเซสเซอร์ เป็นอุปกรณ์ประมวลผลข้อมูลขนาดเล็กภายในประกอบด้วยหน่วยประมวลผลกลางหรือ CPU ประกอบด้วยหน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรเชื่อมต่อหน่วยความจำ วงจรกำเนิดสัญญาณนาฬิกา อีกคำหนึ่งคือคำว่า “คอนโทรลเลอร์”หมายถึง อุปกรณ์ควบคุม ดังนั้นไมโครคอนโทรลเลอร์จึงเป็นอุปกรณ์ที่ใช้ในการควบคุม โดยที่สามารถเขียนโปรแกรมเพื่อกำหนดรูปแบบการควบคุมได้อย่างเป็นอิสระ

ไมโครคอนโทรเลอร์ในปัจจุบันก็มีอยู่หลายยี่ห้อ เช่น PIC ของบริษัทไมโครชิพ Z80 MCS-51 ARM-Cortex AVR และ อื่นๆอีกมาก Arduino เป็นไมโครคอนโทรเลอร์ชนิดหนึ่งที่มีเอกลักษณ์เฉพาะตัวที่ต่างจากอุปกรณ์อื่นๆ คือ การเป็น Open Source

Arduino Mega 2560 เป็นบอร์ดไมโครคอนโทรลเลอร์ที่ทำงานบนพื้นฐานของ ATmega2560 เป็น MCU ประจำบอร์ด RUN ความถี่ 16 MHz, 100PIN TQFP 256 kbyte Flash ทั้งนี้บอร์ด Mega 2560 ยังมีความหน่วยความจำแบบ Flash ทำให้สามารถเขียนคำสั่งโปรแกรมเข้าไปได้มาก ในการใช้เขียนโปรแกรม (8 kbyte สำหรับ Bootloader) / 8 kbyte SRAM / 4 kbyte EEPROM รองรับการพัฒนาโปรแกรมด้วยภาษาซี ออกแบบมาสำหรับงานที่ต้องใช้ I/O มากๆ การรับสัญญาณจาก Sensor หรือควบคุมมอเตอร์ Servo หลายๆตัว เขียนโปรแกรมบนซอฟท์แวร์ Arduino IDE และโปรแกรมผ่านพอร์ต USB [1]

2.1.2 โครงสร้างของบอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560

2.1.2.1 บอร์ดไมโครคอนโทรลเลอร์ Arduino Mega 2560



รูปที่ 2.1 โมดูล Arduino Mega 2560 [2]

2.1.2.2 โครงสร้างและหน้าที่ของพอร์ตต่างๆบนโมดูล Arduino Mega 2560

1) Input and Output ในแต่ละ digital pins ทั้ง 54 pins บนบอร์ด Arduino Uno สามารถเป็นได้ทั้ง input และoutput โดยจะทำงานที่แรงดัน 5 V และให้กระแสสูงสุด 40 mA

2) Serial: 0 (Rx) และ 1(Tx); Serial 1: 19(Rx) และ 18 (Tx); Serial 2: 17 (Rx) และ16(Tx); Serial 3:15 (Rx) และ 14 (Tx) ใช้สำหรับรับ (Rx) และส่ง(Tx) TTL serial dataโดย pin 0 และ 1 จะถูกเชื่อมต่อไปยัง corresponding pins ของ ATmega16U2 USB-to-TTL serial chip

3) External Interrupts: 2 (interrupt 0) , 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), 21 (interrupt 2) pins เหล่านี้สามารถที่จะกำหนดค่าที่เรียกinterrupt ในค่าต่ำๆ, ขอบขาขึ้นและลง หรือเปลี่ยนแปลงค่า

4) PWM: 2 ถึง 13 และ 44 ถึง 46 ให้ output PWM output 8-bits

5) SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS) ใช้สำหรับรองรับการสื่อสารแบบ SPI โดยที่ไม่เกี่ยวข้องกันกับ ICSP header ซึ่งจะมีลักษณะคล้ายกับ Uno, Duemilanove และ Diecimila

6) LED 13 : เป็น build-in LED ที่เชื่อมต่อกับ digital pin 13 เมื่อ pin มีค่าเป็น HIGH LEDจะติด , แต่เมื่อ pin เป็น LOW LED จะดับ

7) TWI : 20 (SDA) and 21 (SCL). รองรับการเชื่อมต่อแบบ TWI(I2C)

8) บอร์ด Mega2560 มี 16 analog inputs แต่ละ pins ให้ความละเอียด 10 bits

9) AREF. แรงดันอ้างอิง สำหรับ analog input

10) Reset ใช้ในการ reset ไมโครคอนโทรลเลอร์ โดยทั่วไปจะใช้โดยการเพิ่มปุ่ม reset ไว้บน shield เพื่อป้องกันปุ่มที่อยู่บนบอร์ด

2.2 โปรแกรม Arduino

2.2.1 ทฤษฎีภาษา Arduino

"Arduino" เป็นภาษาอิตาลีซึ่งเป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ ตระกูล AVR แบบ Open source ที่ได้รับการปรับปรุงมาจากโครงการ Open source ของAVR Arduino มีจุดเด่นในเรื่องของความง่ายในการเรียนรู้และใช้งาน เนื่องจากมีการออกคำสั่งต่างๆขึ้นมาสนับสนุนการใช้งาน ด้วยรูปแบบที่ง่ายไม่ซับซ้อน ซึ่งแม้ว่า Arduino จะมีรูปแบบการใช้งานคล้ายๆกับไมโครคอนโทรลเลอร์อย่าง BasicStap ของ Parallax แต่ก็มีจุดเด่นกว่ารายอื่นคือ ราคาไม่แพง โปรแกรมที่ใช้พัฒนาของ Arduino สามารถรองรับการทางานทั้ง Window, Linux และ OSx คำสั่งง่ายต่อการใช้งาน แต่สามารถนำไปใช้งานจริงๆ กับส่วนที่มีความซับซ้อนมากๆได้ และยังสามารถสร้างคำสั่งรวมถึง Library ใหม่ๆ ขึ้นมาใช้งานเองได้ เมื่อมีความชำนาญมากขึ้น

สำหรับการเขียนโปรแกรมของ Arduino นั้นใช้ภาษา C++ ซึ่งเป็นรูปแบบของภาษาซี ประยุกต์รูปแบบหนึ่งที่มีโครงสร้างการทำงานของตัวภาษาโปรแกรมโดยรวมคล้ายกับภาษาซีมาตรฐานทั่วไป เพียงแต่ได้มีการปรับปรุงเพื่อลดความยุ่งยากในการใช้งานลดลง และให้ผู้ใช้สามารถ ใช้งานเขียนโปรแกรมได้ง่าย สะดวกมากกว่าการเขียนภาษาซีแบบมาตรฐาน แต่ในความเป็นจริง นั้นโปรแกรมดังกล่าวไม่ใช่ C-compiler โดยตรง เนื่องจาก Arduino จะมีลักษณะการทำงาน เช่นเดียวกับ Text Editor ของภาษา C++ ตัวหนึ่งโดยจะทางานร่วมกับ Utility บางส่วนที่ Arduino สร้างขึ้นมารองรับโดย Arduino จะใช้รูปแบบการทำงานของ Editor เป็นฉากหน้า ในการติดต่อสื่อสารกับผู้ใช้เท่านั้น ส่วนเบื้องหลังจริงๆแล้ว Arduino จะไปเรียกใช้ตัวแปลภาษาซี และ Utility อื่นที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ตระกูล AVR อีกทีหนึ่ง Arduino จะเลือกใช้คอมไพลเลอร์ของ "GNU AVR-GCC Toolchain" ร่วมกับ Library function ของ "avr-libc" ส่วน Utility ที่ใช้ในการอัพโหลดโค้ดให้กับ AVR จะใช้ในส่วนของ "AVRDude"

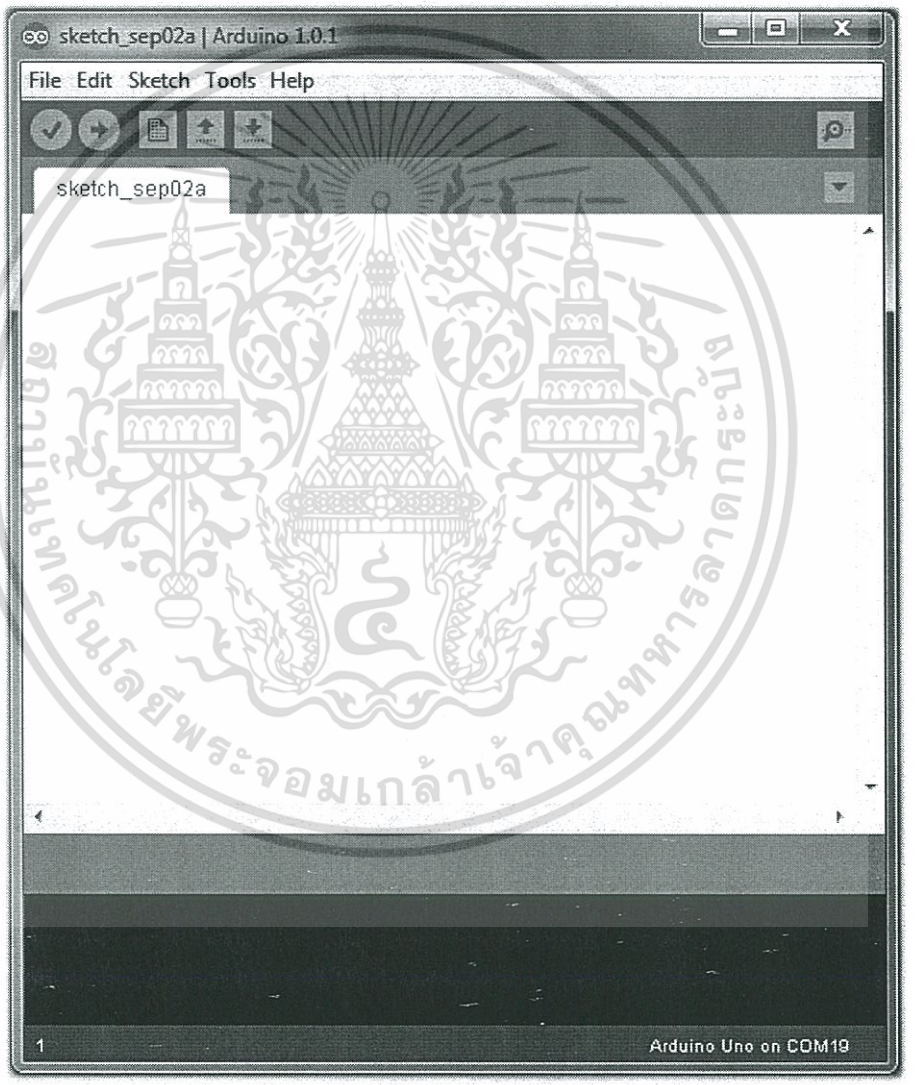
ภาษาซีของ Arduino จะจัดแบ่งรูปแบบโครงสร้างของการเขียนโปรแกรมเป็นส่วนย่อยๆ หลายๆส่วน โดยเรียกแต่ละส่วนว่า “ฟังก์ชั่น” และเมื่อนำฟังก์ชันมารวมเข้าด้วยกัน ก็จะเรียกว่า “โปรแกรม” โดยโครงสร้างการเขียนโปรแกรมของ Arduino ทุกๆ โปรแกรมจะประกอบไปด้วย ฟังก์ชั่นจำนวนเท่าใดก็ได้ แต่อย่างน้อยสุดต้องมี 2 ฟังก์ชั่น คือ Setup( ) และ Loop( ) 6

- Setup( ):เป็นฟังก์ชั่นบังคับที่ต้องกำหนดให้มีทุกๆโปรแกรม ถึงแม้ว่าบางโปรแกรม จะไม่ต้องการใช้งานก็ยังจำเป็นต้องประกาศไว้เสมอ เพียงแต่ไม่ต้องเขียนคำสั่งใดๆไว้ หลังวงเล็บปีกกา { } ที่ใช้เป็นตัวกำหนดขอบเขตของฟังก์ชั่น โดยฟังก์ชันนี้จะใช้สำหรับ บรรจุคำสั่งที่ต้องการให้โปรแกรมทำงานเพียงรอบเดียว ตอนเริ่มต้นทำงานของ โปรแกรมครั้งแรกเท่านั้น ซึ่งได้แก่คำสั่งเกี่ยวกับการ Setup ค่าการทำงานต่างๆ เช่น การกำหนดหน้าที่ของการใช้งานของ PinMode และค่า Baudrate สาหรับการใช้งาน สื่อสารพอร์ตอนุกรม เป็นต้น

- Loop( ) : เป็นส่วนฟังก์ชั่นบังคับที่ต้องกำหนดให้มีในทุกโปรแกรม เช่นเดียวกันกับ ฟังก์ชั่น Setup( ) โดยฟังก์ชั่น Loop( ) นี้จะใช้ในการบรรจุคำสั่งที่ต้องการให้โปรแกรม ทำงานเป็นวนรอบซ้ำๆ กันไปไม่รู้จบ ซึ่งเปรียบเทียบได้กับฟังก์ชั่น main ( ) ใน ANSCI-C นั่นเอง 7 [3]

2.2.2 พื้นฐานจำเป็นในการใช้โปรแกรม Arduino IDE

โปรแกรม Arduino IDE (IDE นั้นย่อมาจาก Integrated Development Environment) ซึ่งสามารถใช้งานได้ทั้งบนระบบปฏิบัติการ Window (XP Vista 7 8) ทั้ง 32 และ 64 บิต, Mac OS X และ Linux เรียกได้ว่าใช้งานได้กับทุกระบบปฏิบัติการ และเป็นอิสระจากการทำงานของ OS ทุกชนิด ทำให้ไม่ต้องมีการ Install โปรแกรมให้ยุ่งยากเหมือนโปรแกรมอื่น สามารถ Download มา จากนั้น Unzip ไว้ใน Directory ที่ต้องการก็สามารถเริ่มใช้งานโปรแกรมได้

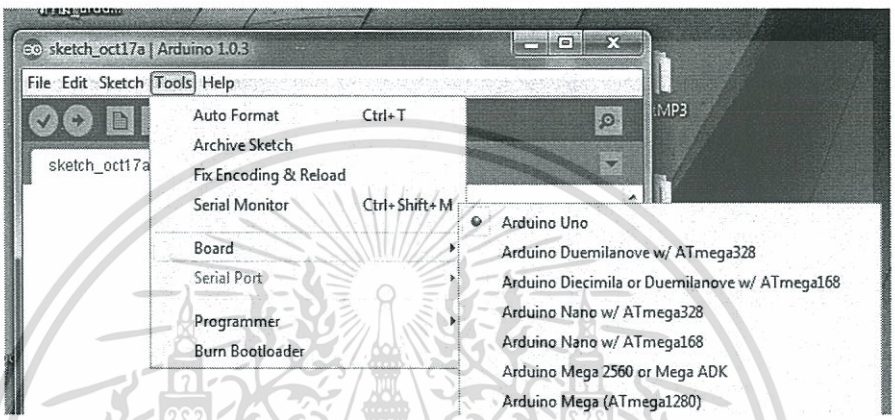


รูปที่ 2.2 แสดงการเปิดโปรแกรม Arduino IDE [3]

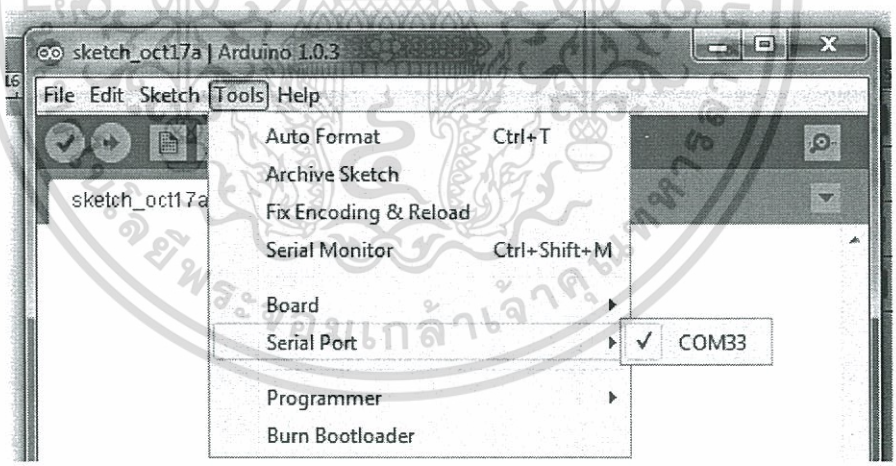
2.2.3 อธิบายการทำงานของโปรแกรม Arduino IDE

1) เขียนโปรแกรมบนคอมพิวเตอร์ ผ่านทางโปรแกรม Arduino IDE ซึ่งสามารถดาวน์โหลดได้จาก Arduino.cc/en/main/software

2) หลังจากที่เขียนโค้ดโปรแกรมเรียบร้อยแล้ว ให้ผู้ใช้งานเลือกรุ่นบอร์ด Arduino ที่ใช้และหมายเลข Com port

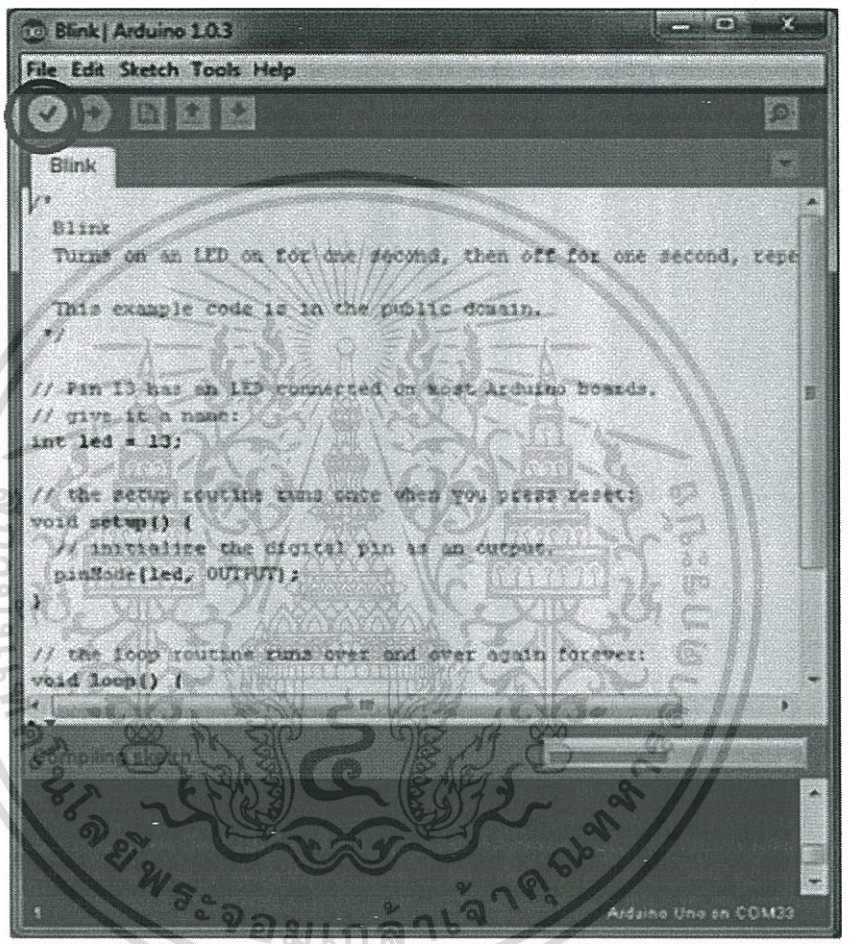


รูปที่ 2.3 แสดงการเลือกรุ่นบอร์ด Arduino ที่ต้องการ upload [3]



รูปที่ 2.4 แสดงการเลือกหมายเลข Comport ของบอร์ด [3]

3) กดปุ่ม Verify เพื่อตรวจสอบความถูกต้อง และ Compile โค้ดโปรแกรม จากนั้นกดปุ่ม Upload โค้ด โปรแกรมไปยังบอร์ด Arduino ผ่านทางสาย USB เมื่ออัพโหลดเรียบร้อยแล้ว จะแสดงข้อความแถบข้างล่าง “Done uploading” และบอร์ดจะเริ่มทำงานตามที่เขียนโปรแกรมไว้ได้ทันที [3]

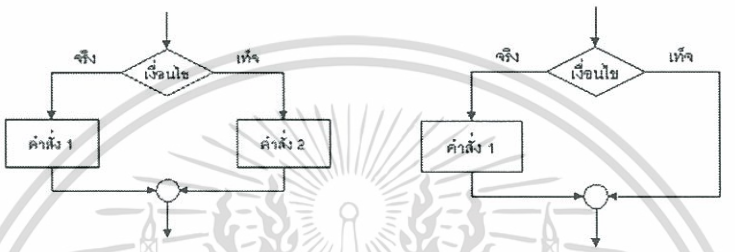


รูปที่ 2.5 กดปุ่ม Verify เพื่อตรวจสอบความถูกต้อง และ Compile โค้ดโปรแกรม [3]

2.3 ภาษาซีที่ใช้ในการเขียนโปรแกรม

2.3.1 คำสั่ง if

คำสั่ง if เป็นคำสั่งที่ใช้ในการเขียนแบบเงื่อนไข โดยจะให้ว่าประโยคเงื่อนไขดังกล่าวมีอยู่ 2 ลักษณะ คือถ้าเงื่อนไขเป็นจริงเกิดเหตุการณ์หนึ่ง แต่ถ้าไม่จริงจะเกิดอีกเหตุการณ์หนึ่งกับประโยคในลักษณะที่ถ้าเงื่อนไขเป็นจริงจึงจะเกิดเหตุการณ์ขึ้นเท่านั้น ทั้ง 2 ลักษณะสามารถเขียนเป็นผังงานของงานได้ดังรูปที่ (ก) และ (ข)



(ก) (ข)

รูปที่ 2.6 แสดงผังงานของประโยคเงื่อนไข [4]

จากผังงานทั้ง 2 จะมีรูปแบบการเขียนคำสั่ง if เกิดขึ้น 2 แบบ

2.3.1.1 คำสั่ง if แบบไม่ซับซ้อน

ในกรณีที่ประโยคเงื่อนไขมีการทำงานเฉพาะเงื่อนไข ที่เป็นจริงเท่านั้น โดยไม่มีการทำงานใดในเงื่อนไขที่เป็นเท็จ ดังแสดงในรูปที่ 3.1 (ข) สามารถเขียนแทนด้วยคำสั่ง if โดยไม่ต้องใส่คำสั่ง else แสดงดังรูปแบบ

```
if ( เงื่อนไข )

    คำสั่งที่ 1;
```

แต่ถ้าเงื่อนไขเป็นจริงแล้วมีการทำคำสั่งมากกว่า 1 คำสั่งขึ้นไป ก็ใช้รูปแบบของเครื่องหมาย{ } ซึ่งใช้ในกรณีที่มีคำสั่งที่ต้องทำในเงื่อนไขและการวนซ้ำมากกวา 1 คำสั่ง เพื่อแสดงขอบเขตของการทำงานนั้น [4]

2.3.1.2 คำสั่ง if แบบซับซ้อน

ในบางกรณีประโยคเงื่อนไขอาจจะมีความซับซ้อน มีการเปรียบเทียบ เงื่อนไขเดียวกันกับหลายค่า

2.3.2 คำสั่ง if-else

คำสั่ง if ในรูปแบบแรกจะคำสั่งที่ต้องทำทั้งในกรณีที่เงื่อนไขเป็นจริงและเป็นเท็จ โดยใช้นิพจน์ตรรกศาสตร์มาเป็นเครื่องมือช่วยในการตรวจสอบเงื่อนไข มีรูปแบบคำสั่ง คือ

if (เงื่อนไข)

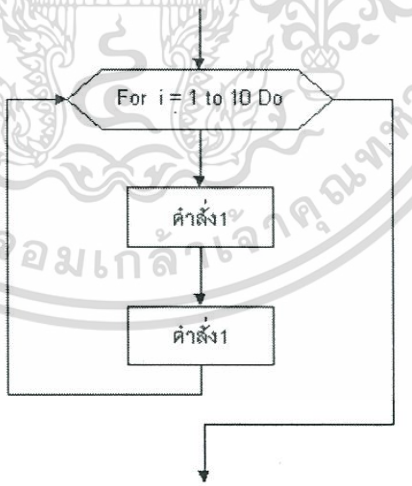
คำสั่งที่ 1;

else

คำสั่งที่ 2;

2.3.3 คำสั่ง for

คำสั่งวนซ้ำเป็นคำสั่งใช้แก้ปัญหาโจทย์ในลักษณะที่มีการทำงานเดิมซ้ำกันหลายๆ ครั้ง ซึ่งเขียนในรูปแบบของผังงานได้ดังรูปที่ 2.7



รูปที่ 2.7 แสดงผังงานของคำสั่ง for [4]

คำสั่ง for เป็นคำสั่งวนซ้ำในลักษณะที่รู้จำนวนรอบของการวนซ้ำที่แน่นอน โดยแบ่งรูปแบบหลักออกเป็น 3 ส่วน ได้แก่

- ส่วนที่ใช้กำหนดค่าเริ่มต้นหรือกำหนดค่าตัวนับของการวนซ้ำ
- ส่วนที่ตรวจเงื่อนไขการวนซ้ำ
- ส่วนของการจัดการค่าตัวนับของการวนซ้ำ

```
for ( กำหนดค่าตัวนับ ; เงื่อนไขการวนซ้ำ ; จัดการค่าตัวนับ )
{
คำสั่ง1;
คำสั่ง2;
}
```

ขั้นตอนของการทำงานเมื่อพบคำสั่ง for มีดังนี้

1). ทำคำสั่งในการกำหนดค่าตัวนับ

2). ตรวจสอบเงื่อนไขการวนซ้ำ หากเป็นเท็จจะหยุดและออกจากการทำงานของ คำสั่ง for ไปทำงานคำสั่งหลังจากนั้น

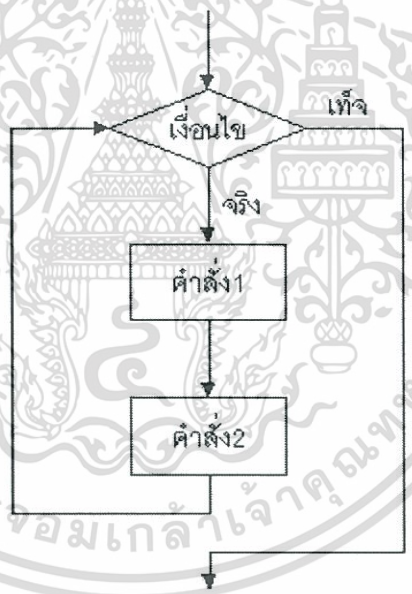
3). กรณีเงื่อนไขการวนซ้ำเป็นจริง จะทำคำสั่งในขอบเขตของ for นั้น คือ ภายใต้ เครื่อง { } จนกระทั่งหมด และไปทำคำสั่งจัดการค่าตัวนับ ซึ่งอาจจะเป็นการเพิ่มค่าหรือลดค่าตัวนับ หลังจากนั้นจะกลับทำตรวจสอบเงื่อนไขการวนซ้ำในขั้นตอนที่ 2 ทำเช่นนี้เรื่อยไปจนกระทั่งเงื่อนไขการวนซ้ำเป็นเท็จ

หากคำสั่งที่ต้องทำในการวนซ้ำมีเพียง 1 คำสั่ง รูปแบบการเขียนจะเขียนเครื่องหมาย { } ครอบคำสั่งนั้นไว้หรือไม่ก็ได้แต่ถ้ามีคำสั่งที่ต้องทำซ้ำมากกว่า 1 คำสั่ง จะต้องมีเครื่องหมาย { } แสดงขอบเขตของการทำวนซ้ำเสมอ

2.3.4 คำสั่ง while

คำสั่ง while เป็นคำสั่งวนซ้ำ มักใช้ในกรณีที่ต้องทำงานซ้ำกันหลายๆ ครั้ง โดยไม่ทราบจำนวนรอบของการทำซ้ำที่แน่นอน ตัวอย่างเช่น ต้องการรับข้อมูลเลขจำนวนเต็มบวกจากผู้ใช้จำนวนหนึ่งเพื่อนำมาหาค่าเฉลี่ยของตัวเลขที่ป้อนเข้ามาทั้งหมด

การทำงานดังกล่าวจะต้องมีการทำงานวนซ้ำเพื่อรับข้อมูลและหาผลรวมของข้อมูลที่รับเข้ามานั้น ในกรณีเช่นนี้ผู้ที่ใช้งานโปรแกรมซึ่งป้อนจำนวนข้อมูลเข้าสู่ระบบอาจจะป้อนข้อมูลในจำนวนที่ไม่เท่ากัน หากต้องการเขียนโปรแกรมเพื่อให้ทำงานกับผู้ใช้คนใด ๆ มักจะใช้คำสั่ง while เข้ามาช่วยในการเขียนโปรแกรม ในกรณีตัวอย่างเราทราบว่าข้อมูลที่รับเข้ามาต้องเป็นข้อมูลจำนวนเต็มบวกเท่านั้นจึงจะนำมาหาค่าเฉลี่ย ซึ่งผู้เขียนโปรแกรมสามารถตั้งเงื่อนไขว่า หากมีการป้อนข้อมูลเป็นเลขจำนวนเต็มลบให้แสดงว่าผู้ใช้ต้องการหยุดการป้อนข้อมูลนั้น การทำงานของคำสั่ง while สามารถเขียนแสดงด้วยผังงานดังรูปที่ 2.8



รูปที่ 2.8 ผังงานของการทำงานคำสั่ง while [4]

หากแทนผังงานดังกล่าวด้วยคำสั่ง while สามารถเขียนรูปแบบของคำสั่ง while ได้ดังนี้

```
while ( เงื่อนไข )

{

คำสั่ง1 ;

คำสั่ง2 ;

}
```

จากรูปแบบคำสั่งดังกล่าว จะเกิดการทำงานคำสั่ง 1 และคำสั่ง 2 เมื่อมีการตรวจสอบว่าเงื่อนไขเป็นจริง และจะทำงานซ้ำเช่นนี้ไปจนกว่าเงื่อนไขนั้นจะเป็นเท็จ หากคำสั่งที่ทำซ้ำมี มากกว่า 1 คำสั่งจะต้องใช้เครื่องหมายแสดงขอบเขต คือ { } ครอบคำสั่งที่ต้องการให้ทำซ้ำทั้งหมด แต่ถ้ามีคำสั่งที่ต้องทำซ้ำเพียงคำสั่งเดียว ผู้เขียนโปรแกรมใส่เครื่องหมาย { } หรือไม่ก็ได้ [4]

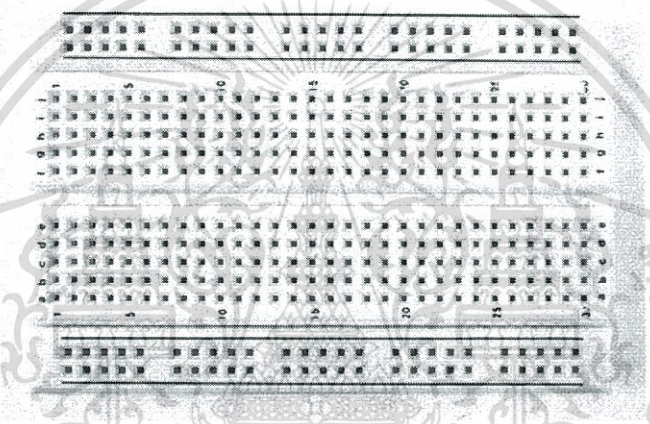
2.4 เครื่องคอมพิวเตอร์

เครื่องคอมพิวเตอร์นับเป็นสิ่งสำคัญที่สุดในการพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์ ไม่ว่าจะเป็นไมโครคอนโทรลเลอร์ตระกูลไหนก็ตาม ปัจจุบันคอมพิวเตอร์ที่ใช้ในการพัฒนาโปรแกรมไมโครคอนโทรลเลอร์ที่นิยมใช้มีอยู่ 2 ประเภทคือ คอมพิวเตอร์ส่วนตัว (PC) และคอมพิวเตอร์พกพา (Notebook) ซึ่งเครื่องคอมพิวเตอร์เหล่านี้ จะใช้ในการติดตั้งและใช้ในงานซอร์ฟแวร์ต่างๆที่เกี่ยวข้องกับการพัฒนาโปรแกรมบนไมโครคอนโทรลเลอร์ เช่น โปรแกรม Text Editor, โปรแกรมคอมไพเลอร์, ไมโครคอนโทรลเลอร์, โปรแกรมควบคุมบอร์ด, โปรแกรมจำลองการทำงานของวงจร เป็นต้น [5]

รูปที่ 2.9 เครื่องคอมพิวเตอร์ [5]

2.5 โปรโตบอร์ด

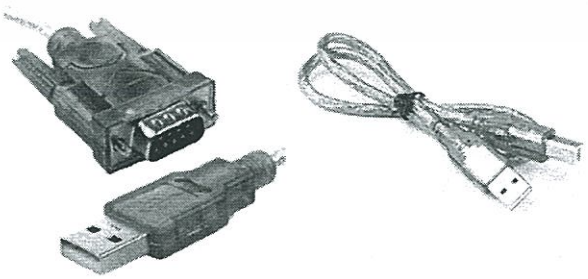
ในการสร้างชิ้นงานทางอิเล็กทรอนิกส์โดยเฉพาะในงานไมโครคอนโทรลเลอร์ ก็จำเป็นจะต้องมีการต่อวงจรไมโครคอนโทรลเลอร์กับอุปกรณ์ภายนอก เพื่อทดสอบว่าโปรแกรมที่เขียนขึ้นทำให้ไมโครคอนโทรลเลอร์สามารถทำงานได้หรือไม่ซึ่งโปรโตบอร์ดนี้เอง จะอำนวยความสะดวกให้สำหรับขั้นตอนนี้ เพราะสามารถใช้สายไฟอุปกรณ์อิเล็กทรอนิกส์ต่างๆและชิปไมโครคอนโทรลเลอร์ต่อรวมกันเป็นวงจรได้โดยไม่ต้องอาศัยการบัดกรีเหมือนกับการต่อวงจรบนแผ่น PCB ทำให้แก้ไขวงจรและโปรแกรมได้ง่าย เพราะไม่แน่ใจว่าโปรแกรมที่เขียนขึ้นและวงจรจะให้ฟังก์ชั่นตามที่ต้องการ เมื่อได้ฟังก์ชั่นการทำงานที่ถูกต้องแล้วจึงค่อยนำอุปกรณ์ทั้งหมดติดลงบนแผ่น PCB เพื่อให้ได้ชิ้นงานตามที่ต้องการ [6]



รูปที่ 2.10 โปรโตบอร์ด [6]

2.6 USB

Universal Serial Bus (USB) เป็นข้อกำหนดมาตรฐานของบัสการสื่อสารแบบอนุกรม เพื่อใช้ในการเชื่อมต่อกับอุปกรณ์ เช่น การเชื่อมต่อระหว่าง เครื่องคอมพิวเตอร์กับบอร์ด Arduino และการเชื่อมต่อ ระหว่าง วงจร Max232 กับ PC คอมพิวเตอร์

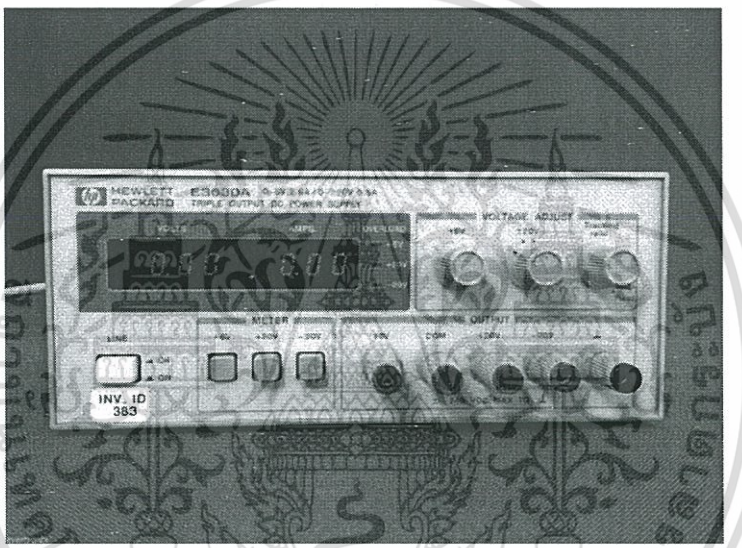


(ก) (ข)

รูปที่ 2.11 (ก) สาย USB RS232 (ข) สาย USB Arduino [7]

2.7 แหล่งจ่ายไฟ

อุปกรณ์อิเล็กทรอนิกส์ ไม่ว่าจะเป็น Arduino หรือ Sensor ต่างๆ จะทำงานก็ได้ต้องมีแหล่งจ่ายไฟเลี้ยงให้กับมัน Power supply ทำหน้าที่ เป็นแหล่งจ่ายไฟแรงดันต่ำ DC ซึ่งแปลงจาก 220V AC มาเป็น 12V DC เพื่อเป็นแหล่งจ่ายไฟแรงดันต่ำ ให้กับอุปกรณ์และวงจรต่างๆ ในโครงงานนี้ และใช้ Power Bank ในการเป็นแหล่งจ่ายไฟโดยกักเก็บจากแผงโซล่าร์เซลล์ ซึ่งพลังงานที่เก็บนั้นเป็นพลังงานจากการเปลี่ยนพลังงานแสงอาทิตย์ให้เป็นพลังงานไฟฟ้า

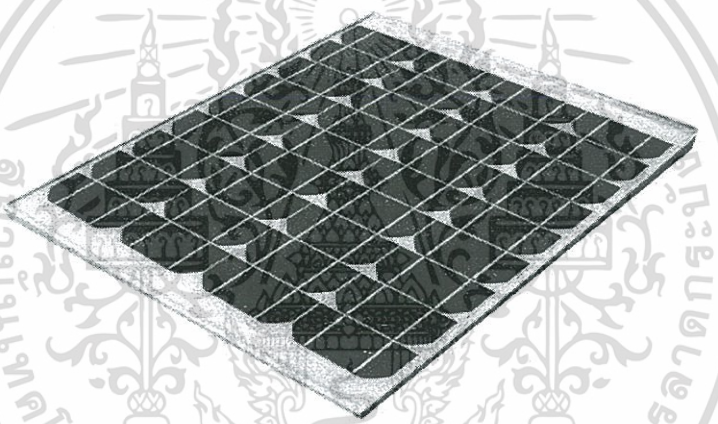


รูปที่ 2.12 Power supply [8]

รูปที่ 2.13 Power Bank [9]

2.8 Solar Cell

อุปกรณ์สำหรับเปลี่ยนพลังงานแสงอาทิตย์ให้เป็นพลังงานไฟฟ้า โดยการนำสารกึ่งตัวนำ เช่น ซิลิกอน ซึ่งมีราคาถูกที่สุดและมีมากที่สุดบนพื้นโลกมาผ่านกระบวนการทางวิทยาศาสตร์เพื่อผลิตให้เป็นแผ่นบางบริสุทธิ์ และทันทีที่แสงตกกระทบบนแผ่นเซลล์ รังสีของแสงที่มีอนุภาคของพลังงานประกอบที่เรียกว่า โฟรตอน (Proton) จะถ่ายเทพลังงานให้กับอิเล็กตรอน (Electron) ในสารกึ่งตัวนำจนมีพลังงานมากพอที่จะกระโดดออกมาจากแรงดึงดูดของอะตอม (atom) และเคลื่อนที่ได้อย่างอิสระ ดังนั้นเมื่ออิเล็กตรอนเคลื่อนที่ครบวงจรจะทำให้เกิดไฟฟ้ากระแสตรงขึ้น เมื่อพิจารณาลักษณะการผลิตไฟฟ้าจากเซลล์แสงอาทิตย์พบว่า เซลล์แสงอาทิตย์จะมีประสิทธิภาพการผลิตไฟฟ้าสูงที่สุดในช่วงเวลากลางวัน ซึ่งสอดคล้องและเหมาะสมในการนำเซลล์แสงอาทิตย์มาใช้ผลิตไฟฟ้า เพื่อแก้ไขปัญหาการขาดแคลนพลังงานไฟฟ้าในช่วงเวลากลางวัน [10]



รูปที่ 2.14 แผง Solar Cell [10]

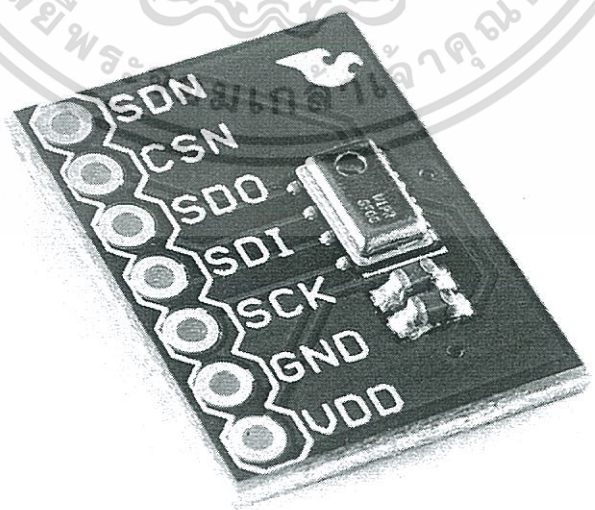
2.9 Temperature Sensor - Waterproof (DS18B20)

Temperature Sensor - Waterproof (DS18B20) เป็นหัววัดอุณหภูมิโดยใช้ Chip DS18B20 ที่ทำหัวเซ็นเซอร์ให้กันน้ำด้วย มีช่วงการวัดอยู่ที่ -55°C เป็น +125°C ใช้งานติดต่อผ่านทาง 1 Wire interface

รูปที่ 2.15 Temperature Sensor - Waterproof (DS18B20) [11]

2.10 MPL115A1 Barometric Pressure Sensor Breakout

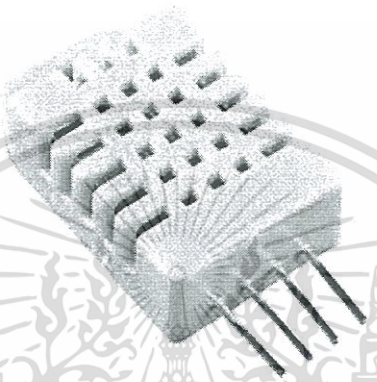
MPL115A1 เป็นเซ็นเซอร์วัดความกดอากาศแบบ Digital ที่ใช้เทคโนโลยี MEMs เพื่อให้ได้ความแม่นยำในการวัดใน ช่วง 50kPa and 115 kPa



รูปที่ 2.16 MPL115A1 Barometric Pressure Sensor Breakout [12]

2.11 DHT22 Temperature Humidity Sensor

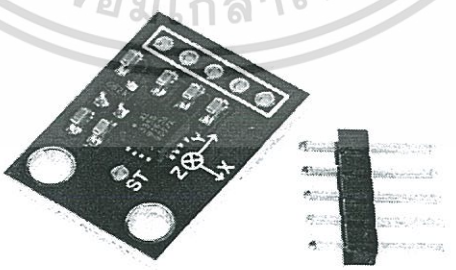
อุปกรณ์เซนเซอร์สำหรับวัดอุณหภูมิและความชื้นสัมพัทธ์ (Temperature & Relative Humidity Sensor) เป็นอุปกรณ์ที่สามารถนำมาประยุกต์ใช้งานทางด้านระบบสมองกลฝังตัวได้หลากหลาย เช่น การวัดและควบคุมอุณหภูมิและความชื้น ระบบบันทึกข้อมูลเกี่ยวกับอุณหภูมิและความชื้นในห้อง เป็นต้น



รูปที่ 2.17 DHT22 Temperature Humidity Sensor [13]

2.12 Accelerometer Module GY-61 ADXL335

GY-61 เป็นโมดูล Accelerometers บนโมดูลประกอบด้วยชิป ADXL335 เอาต์พุตที่ได้เป็นแบบ Analog ใช้ในการหาค่าของการเคลื่อนไหวทั้ง 3 แกน ใช้สำหรับการวัดค่าความเร่งที่คงที่จากการเอียง หรือการวัดความเร่งที่เปลี่ยนแปลงไปจากการเคลื่อนไหว การกระแทก หรือการสั่นสะเทือน



รูปที่ 2.18 Accelerometer Module GY-61 ADXL335 [14]

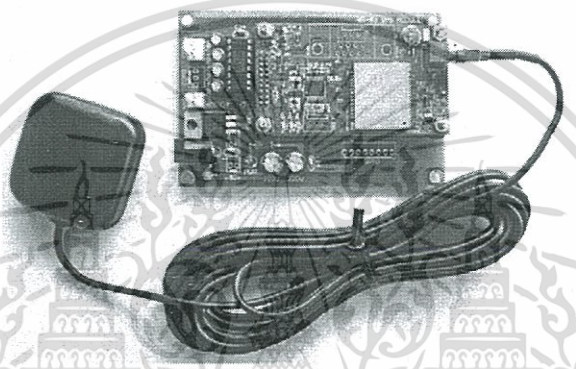
2.13 OPTICAL SENSOR

เซนเซอร์ชนิดใช้แสง (optical sensor) หรือ photo sensor โดยทั่วไปใช้ในงานการตรวจจับการเคลื่อนไหว การตรวจจับวัตถุ และการตรวจสอบขนาดรูปร่างของวัตถุ เซนเซอร์ชนิดนี้ทำงานโดยอาศัยหลักการส่งและรับแสง มีส่วนประกอบสำคัญ 2 ส่วนคือ ตัวส่งแสง (emitter) และตัวรับแสง (receiver) ลักษณะการตรวจจับเกิดจากการที่ลำแสงจากตัวส่งแสง ส่งไปสะท้อนกับวัตถุหรือถูกขวางกั้นด้วยวัตถุ ส่งผลให้ตัวรับแสงรู้สภาวะที่เกิดขึ้นและเปลี่ยนแปลงสภาวะของสัญญาณทางด้านเอาต์พุตเพื่อนำไปใช้งานต่อไป อุปกรณ์ที่เป็นตัวรับแสงส่วนใหญ่นิยมใช้โฟโต้ไดโอด (photo diode) หรือ โฟโต้ทรานซิสเตอร์ (photo transistor) ส่วนตัวส่งแสงนั้นโดยทั่วไปใช้ LED (Light Emitting Diode) เนื่องจากการต่อใช้งานร่วมกับวงจรอิเล็กทรอนิกส์ทำได้ง่าย สะดวกในการบำรุงรักษา ใช้กระแสไฟฟ้าต่ำ และไม่ได้รับผลกระทบจากสภาวะรอบข้างไม่ว่าจะเป็นสนามแม่เหล็ก ความถี่ ความร้อน ความชื้น หรือการสั่นสะเทือน [15]

รูปที่ 2.19 เซนเซอร์ชนิดใช้แสง Optical Sensor [15]

2.14 GPS GR-83 Module

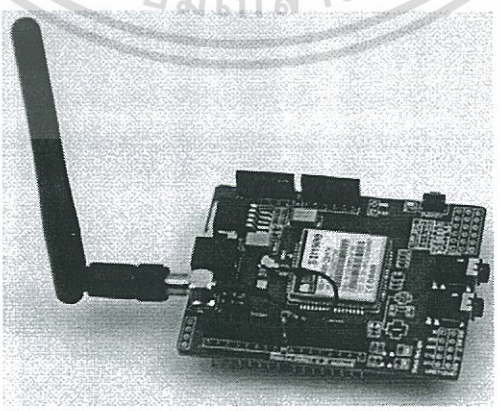
GR-83 GPS Module คือโมดูลรับสัญญาณดาวเทียม GPS เพื่อการบอกตำแหน่งพิกัด ณ จุดใด ๆ และเวลามาตรฐานโลก รับสัญญาณจากดาวเทียมได้ 20 ดวง ส่งข้อมูลออกทาง RS232 (COMA) ความเร็วในการสื่อสาร Baud rate = 4800 ,Data bits : 8, Parity = None ,Stop bit = 1 รับข้อมูล RTCM-104 DGPS (COMB) ความเร็วในการสื่อสาร Baud rate = 9600 ,Data bits : 8, Stop bit = 1 รับสัญญาณดาวเทียมที่ระดับความสูงถึง 18,000 เมตร (60,000 ฟุต) สามารถเลือกใช้แหล่งจ่ายไฟแบบ 5 VDC (สาย L/P2) หรือ Adapter 9 VDC



รูปที่ 2.20 GPS GR-83 Module [16]

2.15 SIM900 GSM / GPRS Module

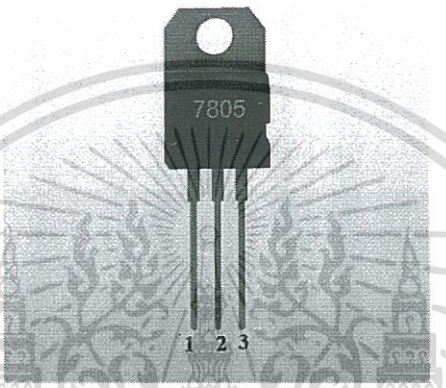
ตัวโมดูลสามารถรองรับสัญญาณ GSM/GPRS ได้ 4 ช่วงสัญญาณ ได้ที่ 850, 900, 1800, 1900 เมกะเฮิรตซ์ โมดูลตัวนี้สามารถทำให้คุณใช้สัญญาณ GSM แบบเดียวกับการใช้โทรศัพท์ในการรับส่งข้อมูลจากที่ต่างๆได้แม้จะไม่มีสัญญาณ internet/3G



รูปที่ 2.21 **SIM900 GSM / GPRS Module** [17]

2.16 IC 7805

ไอซีเรกูเลเตอร์ 7805 จะเสมือนกับมีตัวต้านทานปรับค่าได้ต่ออนุกรมกับวงจรและจะเปลี่ยนแปลงค่าไปเรื่อยๆเพื่อให้แรงดันทางด้านเอาท์พุทคงที่เสมอ ยกตัวอย่างเช่นใช้ไฟเลี้ยง 10 V จ่ายผ่านไอซีเรกูเลเตอร์โดยมีไฟทางเอาท์พุทอยู่ที่ 5 V ฉะนั้น ถ้าอุปกรณ์ที่ต่อทางเอาท์พุทของไอซีเรกูเลเตอร์ใช้กำลังงาน 5 Watt ไอซีเรกูเลเตอร์ก็จะมีกำลังไฟ 5 watts ตกอยู่บนไอซีตัวนี้ด้วย



รูปที่ 2.22 IC 7805 [18]

2.17 โปรแกรม Proteus

โปรแกรม Proteus หรือ Proteus VSM (Virtual System Modeling) เป็นโปรแกรมที่พัฒนาขึ้น โดยบริษัท แล็บเซ็นเตอร์อิเล็กทรอนิกส์ จำกัด (Labcenter Electronics Ltd.) ที่ประเทศอังกฤษ โปรแกรม Proteus มีชื่อเต็มว่า Labcenter Electronics Proteus ซึ่งภายในโปรแกรมจะประกอบด้วยส่วนประกอบหลัก 2 ส่วน คือ ISIS และ ARES

ความสามารถในการทำงานของโปรแกรม Proteus ก็คือ สามารถจำลองการทำงานของวงจรอิเล็กทรอนิกส์ได้หลากหลายรูปแบบ ไม่ว่าจะเป็นแบบอนาล็อกและแบบดิจิตอล หรือทั้งแบบอนาล็อกและดิจิตอลผสมกัน นอกจากนี้ Proteus ยังสามารถออกแบบลายวงจรพิมพ์ (PCB) ได้อีกด้วย จุดเด่นของโปรแกรม Proteus ที่เป็นที่นิยมและชื่นชอบก็คือ การจำลองการทำงานของวงจรอิเล็กทรอนิกส์ ที่ใช้ไมโครคอนโทรลเลอร์ตระกูลต่าง ๆ ไม่ว่าจะเป็น PIC, MCS-51, AVR และ ARM เป็นต้น ทำให้นักเขียนโปรแกรมหรือโปรแกรมเมอร์สามารถตรวจสอบได้ว่าโปรแกรม หรือซอสโค้ด (Source Code) ที่เขียนขึ้นมานั้น สามารถสนับสนุนกับวงจรฮาร์ดแวร์ที่ต่อได้หรือไม่ ถ้าโปรแกรม (Source Code) ที่เขียนขึ้น ไม่สนับสนุนกับวงจรฮาร์ดแวร์ที่ต่อโปรแกรมเมอร์ก็จะทำการพัฒนาโปรแกรม (Source Code) ที่เขียนขึ้นใหม่ หรือปรับปรุงวงจรฮาร์ดแวร์ใน Proteus จนกว่าโปรแกรมที่เขียนขึ้นและฮาร์ดแวร์ที่ต่อ สามารถสนับสนุนซึ่งกันและกัน ทำให้การสร้างโครงงานต่าง ๆ สามารถประหยัดเวลาและค่าใช้จ่ายเป็นอย่างมาก เพราะในอดีตการเขียนโปรแกรมขึ้นมานั้น จะต้องต่อวงจรจริงเพื่อทดสอบ ทำให้เสียเวลาและค่าใช้จ่ายมาก ในกรณีที่วงจรฮาร์ดแวร์และโปรแกรมที่เขียนขึ้นไม่สนับสนุนซึ่งกันและกัน [19]

2.18 โปรแกรมสร้างโดเมนเนม NO-IP

No-IP เป็นโปรแกรม สำหรับผู้ใช้งานคอมพิวเตอร์ที่ต้องการแปลงสัญญาณอินเตอร์เน็ตที่มาจากผู้ให้บริการอินเตอร์เน็ตของเรา ซึ่งเป็นแบบ Dynamic (ซึ่งจะเปลี่ยนไปเรื่อยๆ) ให้เป็นแบบ Static IP (คือค่าคงที่) มันจึงมีประโยชน์สำหรับผู้ที่ต้องการสร้างคอมพิวเตอร์ของเราให้เป็น Web Server หรือ FTP Server ด้วยบริการอินเตอร์เน็ตของ ISP ในปัจจุบันนั้นเอง [20]

2.19 โปรแกรม AppServ

AppServ คือโปรแกรมที่รวบรวมเอา Open Source Software หลายๆ อย่างมารวมกันโดยมี Package หลักดังนี้

- Apache
- PHP
- MySQL
- phpMyAdmin

โปรแกรมต่างๆ ที่นำมารวบรวมไว้ทั้งหมดนี้ ได้ทำการดาวน์โหลดจาก Official Release ทั้งสิ้น โดยตัว AppServ จึงให้ความสำคัญว่าทุกสิ่งทุกอย่างจะต้องให้เหมือนกับต้นฉบับ เราจึงไม่ได้ตัดทอนหรือเพิ่มเติมอะไรที่แปลกไปกว่า Official Release แต่อย่างได้ เพียงแต่มีบางส่วนเท่านั้นที่เราได้เพิ่มประสิทธิภาพการติดตั้งให้สอดคล้องกับการทำงานแต่ละคน โดยที่การเพิ่มประสิทธิภาพนี้ไม่ได้ไปยุ่ง ในส่วนของ Original Package เลยแม้แต่น้อยเพียงแต่เป็นการกำหนดค่า Config เท่านั้น เช่น Apache ก็จะเป็นในส่วนของ httpd.conf, PHP ก็จะเป็นในส่วนของ php.ini, MySQL ก็จะเป็นในส่วนของ my.ini ดังนั้นเราจึงรับประกันได้ว่าโปรแกรม AppServ สามารถทำงานและความเสถียรของระบบ ได้เหมือนกับ Official Release ทั้งหมด

จุดประสงค์หลักของการรวมรวบ Open Source Software เหล่านี้เพื่อทำให้การติดตั้งโปรแกรมต่างๆ ที่ได้กล่าวมาให้ง่ายขึ้น เพื่อลดขั้นตอนการติดตั้งที่แสนจะยุ่งยากและใช้เวลานาน โดยผู้ใช้งานเพียงดับเบิ้ลคลิก setup ภายในเวลา 1 นาที ทุกอย่างก็ติดตั้งเสร็จสมบูรณ์ระบบต่างๆ ก็พร้อมที่จะทำงานได้ทันทีทั้ง Web Server, Database Server เหตุผลนี้จึงเป็นเหตุผลหลักที่หลายๆ คนทั่วโลก ได้เลือกใช้โปรแกรม AppServ แทนการที่จะต้องมาติดตั้งโปรแกรมต่างๆ ที่ละส่วนไม่ว่าจะเป็นผู้ที่ความชำนาญในการติดตั้ง Apache, PHP, MySQL ก็ไม่ได้เป็นเรื่องง่ายเสมอไป เนื่องจากการติดตั้งโปรแกรมที่แยกส่วนเหล่านี้ให้มารวมเป็นชิ้นอันเดียวกัน ก็ใช้เวลาค่อนข้างมากพอสมควร แม้แต่ตัวผู้พัฒนา AppServ เอง ก่อนที่จะ Release แต่ละเวอร์ชั่นให้ดาวน์โหลด ต้องใช้ระยะเวลาในการติดตั้งไม่น้อยกว่า 2 ชั่วโมง เพื่อทดสอบความถูกต้องของระบบ [21]

บทที่ 3

การออกแบบและการจัดทำโครงงาน

3.1 การออกแบบ

หลักการทำงานประกอบด้วย การแจ้งเตือนอุณหภูมิน้ำ อุณหภูมิอากาศ ความดัน ค่าละติจูด ลองจิจูด ความเร็วลม และความสูงของคลื่น โดยจะเก็บข้อมูลและสามารถทำการแจ้งเตือนได้ ข้อมูลทั้งหมดจะถูกส่งผ่านทางระบบอินเทอร์เน็ตแบบไร้สายมายังฐานข้อมูลบนคอมพิวเตอร์ นอกจากนี้ภายในทุ่นจะใช้แหล่งพลังงานมาจากแผงโซล่าเซลล์ที่จะแปลงพลังงานจากแสงอาทิตย์มาเป็น พลังงานไฟฟ้าแล้วนำมาเก็บในแบตเตอร์รี่ขนาดเล็ก ส่วนด้านการประมวลผลจะใช้ตัว Arduino Mega 2560 เป็นอุปกรณ์หลักในการประมวลผล โดยจะอธิบายการออกแบบและการทำงานแยกเป็นส่วนต่างๆดังรูปที่ 3.1

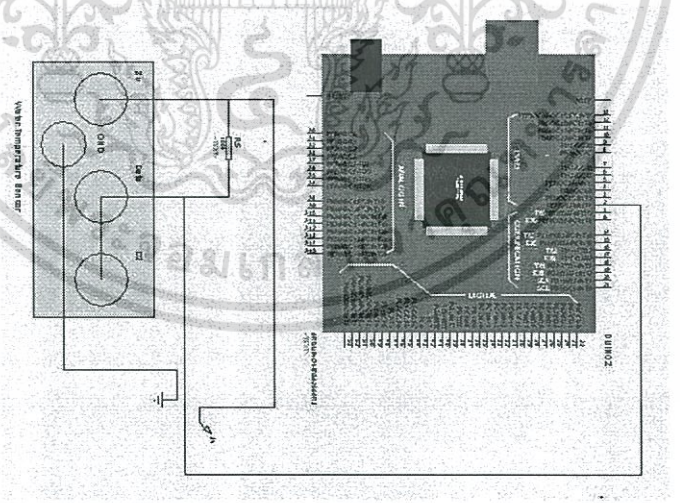
รูปที่ 3.1 บล็อกไดอะแกรมของการทำงาน

3.1.1 อุปกรณ์ที่ใช้ในการทดลอง

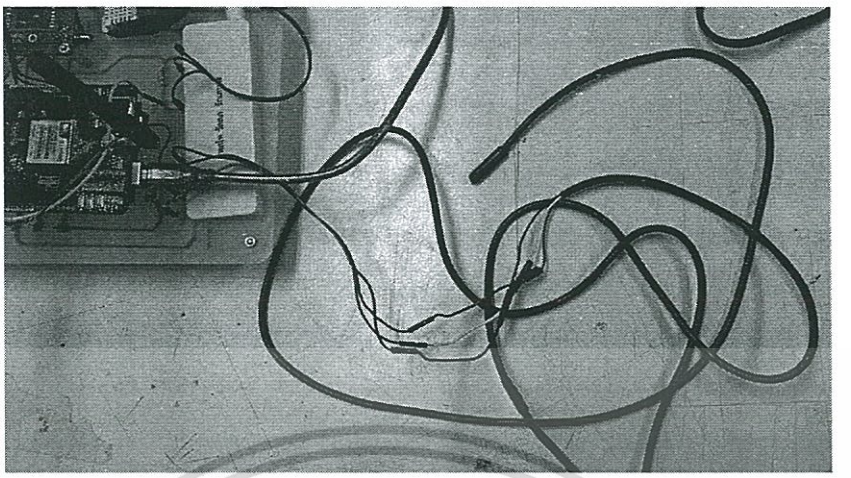
3.1.1.1 Arduino Mega 2560
3.1.1.2 Temperature Sensor (bs18B20)
3.1.1.3 Pressure Sensor (MPL 115A1)
3.1.1.4 Humidity Sensor DHT22
3.1.1.5 Sim900 S2-1040S-Z1K0C
3.1.1.6 Solar cell
3.1.1.7 Power bank
3.1.1.8 LM7805
3.1.1.9 Accelerometer
3.1.1.10 Optical sensor

3.1.2 การออกแบบการเชื่อมต่อระหว่าง Arduino กับ Temperature Sensor

Temperature Sensor Waterproof (DS18B20) เป็นหัววัดอุณหภูมิโดยใช้ Chip DS18B20 ที่ทำหัวเซ็นเซอร์ให้กันน้ำ มีช่วงการวัดอยู่ที่ -55°C ถึง +125°C ใช้งานติดต่อผ่านทาง 1 Wire แสดงวงจรการเชื่อมต่อระหว่าง Arduino กับ Temperature Sensor ดังรูปที่ 3.2 โดยมีวงจรจริงการเชื่อมต่อระหว่าง Arduino กับ Temperature Sensor ดังรูปที่ 3.3 และใช้คำสั่งในการทำงานดังรูปที่ 3.4



รูปที่ 3.2 รูปวงจรการเชื่อมต่อระหว่าง Arduino กับ Temperature Sensor ใน Proteus



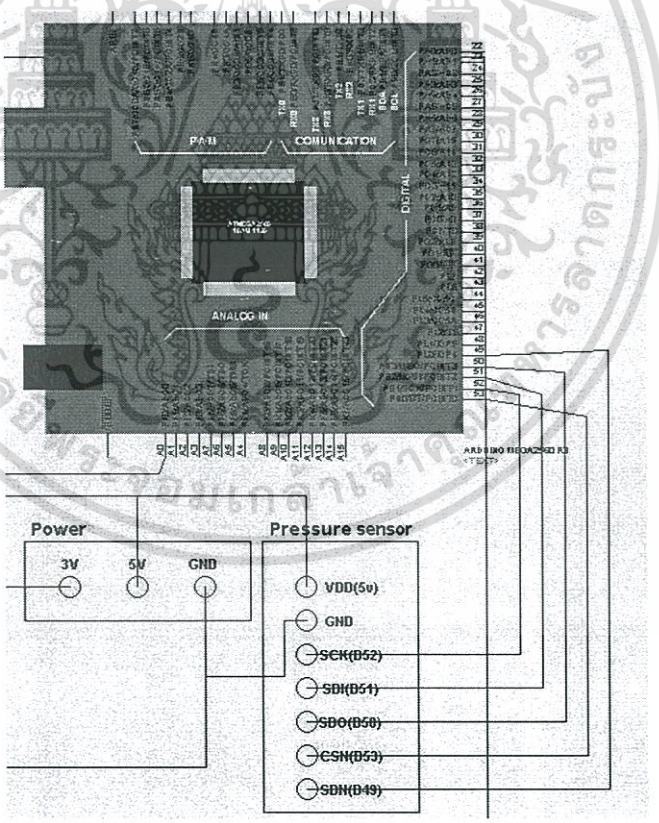
รูปที่ 3.3 แสดงวงจรจริงของการเชื่อมต่อระหว่าง Arduino กับ Temperature Sensor

```
float getTemp(){
 //returns the temperature from one DS18S20 in DEG Celsius

 byte data[12];
 byte addr[8];

 if ( !ds.search(addr)) {
   //no more sensors on chain, reset search
   ds.reset_search();
   return 31.5;
 }

 if ( OneWire::crc8( addr, 7) != addr[7]) {
   Serial.println("CRC is not valid!");
   return 31.5;
 }

 if ( addr[0] != 0x10 && addr[0] != 0x28) {
   Serial.print("Device is not recognized");
   return 31.5;
 }

 ds.reset();
 ds.select(addr);
 ds.write(0x44,1); // start conversion, with parasite power on at the end

 byte present = ds.reset();
 ds.select(addr);
```

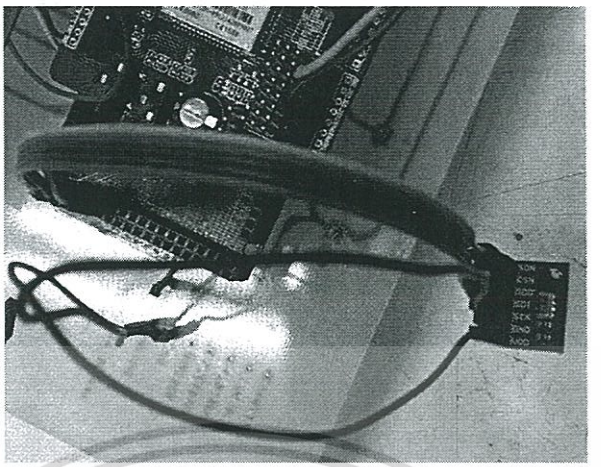
รูปที่ 3.4 คำสั่งโปรแกรม Arduino โดยใช้ LibraryOneWire.h

3.1.3 การออกแบบการเชื่อมต่อระหว่าง Arduino กับ Pressure Sensor (MPL115A1)

Pressure Sensor MPL115A1 เป็นเซ็นเซอร์วัดความกดอากาศแบบ Digital ที่ใช้เทคโนโลยี MEMS เพื่อให้ได้ความแม่นยำในการวัดใน ช่วง 50 kPa and 115 kPa โดยกินกระแสไฟฟ้าเฉลี่ยที่ 10 micro Amp ค่าที่วัดออกมา ได้ทั้งอุณหภูมิ และความดัน จะออกมาทาง SPI bus ช่วงอุณภูมิที่ทำงานคือ -40 C ถึง 105 C โดยเชื่อมต่อกับ Arduino mega ใช้ไฟเลี้ยง 5 volt ขา SCK เข้ากับ Digital 52 ของ Arduino SDI กับ Digital 51 SDO กับ Digital 50 CSN กับ Digital 53SDN กับ Digital 49 แสดงค่าความดันในหน่วย kpa และ mmHg ซึ่งมีวงจรการเชื่อมต่อเข้ากับ Arduino ดังรูปที่ 3.5 และมีวงจรจริงของการเชื่อมต่อระหว่าง Arduino กับ Pressure Sensor ดังรูปที่ 3.6 และใช้คำสั่ง Arduino ในการเก็บค่าจาก Pressure Sensor MPL115A1 ดังรูปที่ 3.7



รูปที่ 3.5 รูปวงจรการเชื่อมต่อระหว่าง Arduino กับ Pressure Sensor



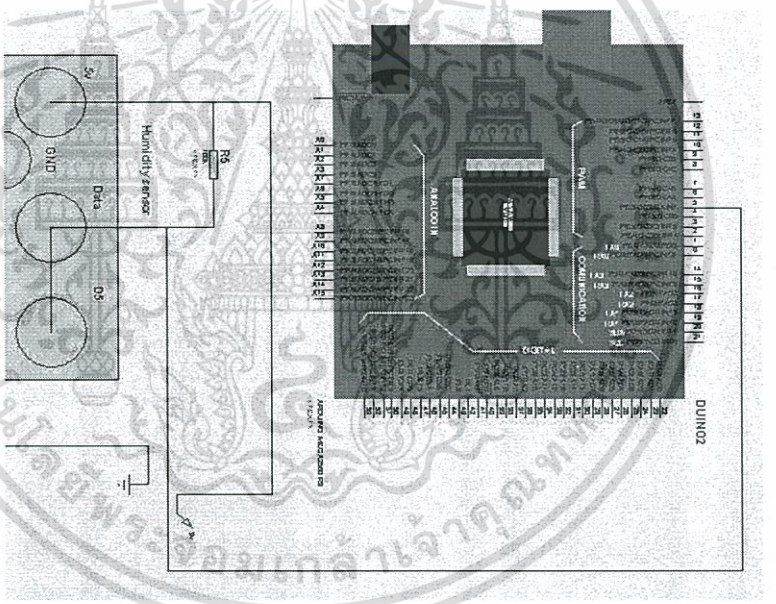
รูปที่ 3.6 รูปวงจรจริงของการเชื่อมต่อระหว่าง Arduino กับ Pressure Sensor

```
  altitude_ft = calculateAltitudeFt(pressure_pKa);

  // put the MPL115A1 to sleep, it has this feature why not use it
  // while in shutdown the part draws ~1uA
  digitalWrite(MPL115A1_ENABLE_PIN, LOW);

  // print table of altitude, pressures, and temperatures to console
  Serial.print(altitude_ft);
  Serial.print(" ft | ");
  Serial.print(FT_TO_M(altitude_ft));
  Serial.print(" m | ");
  Serial.print(KPA_TO_INHG(pressure_pKa), 2);
  Serial.print(" in Hg | ");
  Serial.print(KPA_TO_MMHG(pressure_pKa), 0);
  Serial.print(" mm Hg | ");
  Serial.print(KPA_TO_PSIA(pressure_pKa), 2);
  Serial.print(" psia | ");
  //Serial.print(KPA_TO_KGCM2(pressure_pKa), 3);
  ///Serial.print(" kg/cm2 | ");
  Serial.print(pressure_pKa, 1);
  Serial.print(" kPa | ");

  // At a res of -5.35 counts/ºC, digits lower than 0.1ºC are not significant
  Serial.print(temperature_c, 1);
  Serial.print(" C | ");
  Serial.print(DEGC_TO_DEGF(temperature_c), 1);
  Serial.print(" F\n");
  Serial.println(temperature);
```

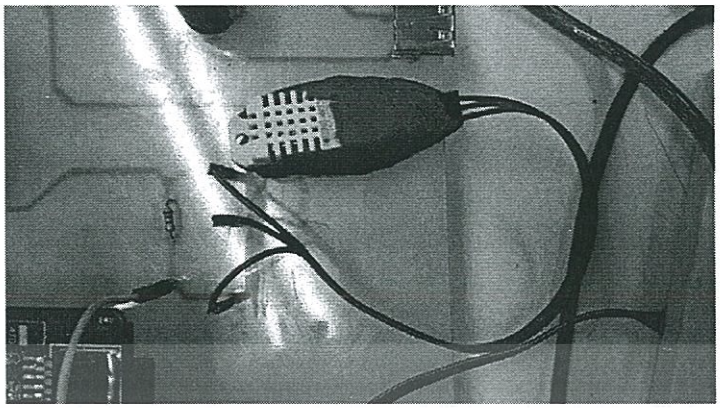
รูปที่ 3.7 คำสั่งโปรแกรม Arduino โดยใช้ LibrarySPI.h และ OneWire.h

3.1.4 การออกแบบการเชื่อมต่อระหว่าง Arduino กับ Humidity Sensor DHT22

DHT22 High Accuracy Digital Temperature and Humidity Sensor ใช้สำหรับวัด อุณหภูมิและความชื้น โดยต่อไฟเลี้ยง 5 volt และขา Data เข้ากับ Digital 5 ของ Arduino โดยทำการออกแบบวงจรการเชื่อมต่อระหว่าง Arduino กับ Humidity Sensor DHT22 ในโปรแกรม Proteus ดังรูปที่ 3.8 และวงจรจริงดังรูปที่ 3.9 ทำการเขียนคำสั่งโปรแกรม Arduino เพื่ออ่านค่าความชื้นและอุณหภูมิจากเซนเซอร์ DHT22 ดังรูปที่ 3.10 และให้แสดงค่าความชื้นในอากาศออกมาในรูปของเปอร์เซ็นต์ (%) แปลว่า ปริมาณไอน้ำที่มีอยู่ในอากาศบริเวณนั้นประกอบด้วยสัดส่วนของน้ำอยู่เท่าใด นอกจากนี้ยังใช้เซ็นเซอร์ตัวนี้ในการวัดอุณหภูมิของอากาศแสดงออกมาในหน่วยองศาเซลเซียส และองศาฟาเรนไฮต์



รูปที่ 3.8 วงจรการเชื่อมต่อระหว่าง Arduino กับ Humidity Sensor DHT22 ในโปรแกรม Proteus



รูปที่ 3.9 วงจรจริงในการเชื่อมต่อระหว่าง Arduino กับ Humidity sensor DHT22

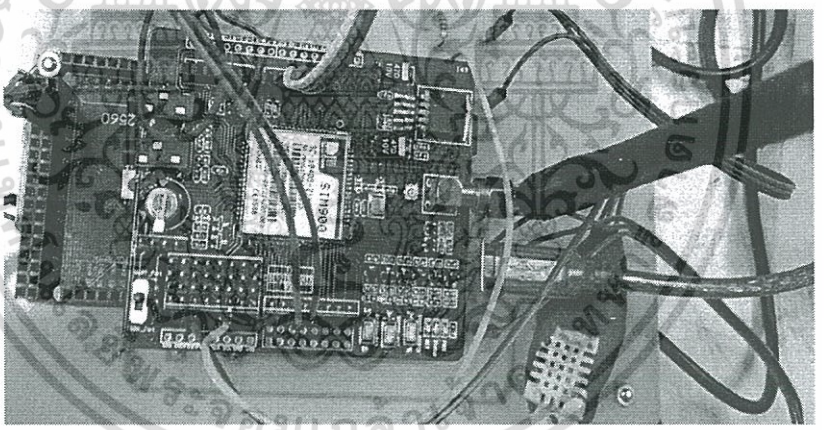
```
stat.total++;
switch (chk)
{
case DHTLIB_OK:
    stat.ok++;
    break;
case DHTLIB_ERROR_CHECKSUM:
    stat.crc_error++;
    Serial.print("Checksum error,\t");
    break;
case DHTLIB_ERROR_TIMEOUT:
    stat.time_out++;
    Serial.print("Time out error,\t");
    break;
case DHTLIB_ERROR_CONNECT:
    stat.connect++;
    Serial.print("Connect error,\t");
    break;
case DHTLIB_ERROR_ACK_L:
    stat.ack_l++;
    Serial.print("Ack Low error,\t");
    break;
case DHTLIB_ERROR_ACK_H:
    stat.ack_h++;
    Serial.print("Ack High error,\t");
    break;
default:
```

รูปที่ 3.10 คำสั่งโปรแกรม Arduino กับ Humidity sensor DHT22 โดยใช้ Librarydht.h

3.1.5 การออกแบบการเชื่อมต่อระหว่าง SIM900 GSM/GPRS Module และ Arduino MEGA 2560

ตัวโมดูลนี้สามารถรองรับสัญญาณ GSM/GPRS ได้ 4 ช่วงสัญญาณ ได้ที่ 850, 900, 1800, 1900 เมกะเฮิรตซ์ โมดูลตัวนี้สามารถทำให้ใช้สัญญาณ GSM แบบเดียวกับการใช้โทรศัพท์ในการรับส่งข้อมูลจากที่ต่างๆได้ ซึ่งการเชื่อมต่อเราจะนำมาประกบกับตัว Arduino mega โดยใช้ขา Digital 9 ในการเปิดปิดและจ่ายไฟให้กับตัวโมดูล และนำมาใช้ในการส่งข้อมูลจากเซ็นเซอร์ต่างๆมาเข้าตัวเซิฟเวอร์ โดยทำการต่อวงจรจริงระหว่าง Arduino กับ SIM900 ดังรูปที่ 3.11

การสร้าง Server และ Database เริ่มจากการตั้งตัวอุปกรณ์โดยใช้เป็นตัว Wireless Router ผ่านทางสายแลนเข้ากับตัวคอมพิวเตอร์ดังรูปที่ 3.12 และจากนั้นทำการ Fix IP ที่ตัวTCP/IPv4 ของ Local Area Connection เพื่อทำการ forward port ทำการเซ็ต Router ให้รู้ว่าเมื่อมีข้อมูลส่งมาที่ port นี้ ให้ส่งข้อมูลนั้นต่อไปที่ไหน ในเครือข่ายหรือฐานข้อมูลที่เราตั้งไว้ ในที่นี้เราจะใช้ Port 80 ซึ่งเป็น Internet port (HTTP) มีขั้นตอนดังรูปที่ 3.13

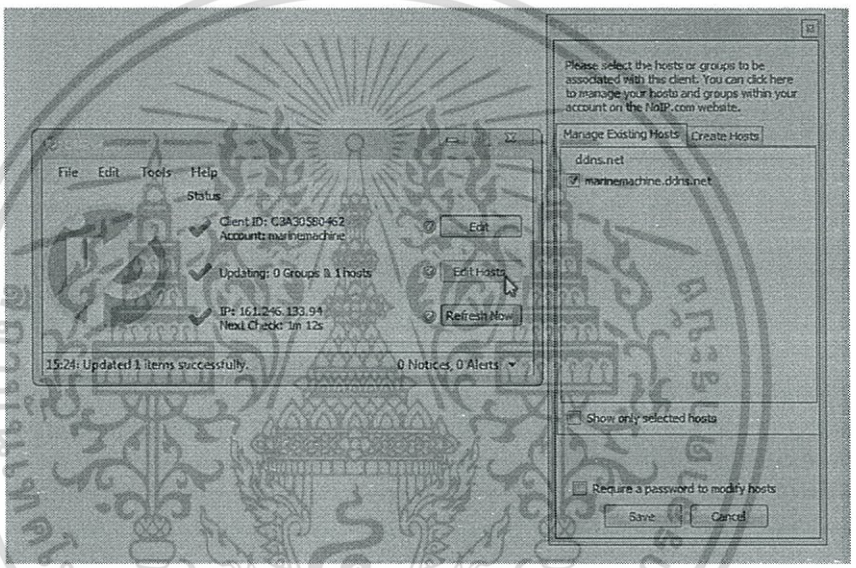


รูปที่ 3.11 วงจรจริงของการเชื่อมต่อระหว่าง Arduino กับ SIM900

รูปที่ 3.12 แสดงตัว Router ที่ทำการต่อตรงเข้าคอมพิวเตอร์ด้วยสาย LAN เพื่อทำการติดตั้ง

รูปที่ 3.13 การทำ Forward port ให้กับตัว Router

ทำการติดตั้งโปรแกรมโปรแกรมที่ใช้ทำการจำลองเครื่องคอมพิวเตอร์เป็นเครื่อง Server ต้องมี Apache PHP และ MySQL มารวมเป็นอาจจะเรียกว่า Package ก็ได้ ซึ่งมีให้เลือกใช้งานอยู่หลายตัวเช่น Appserv, WAMP และ XAMPP เป็นต้น และเมื่อติดตั้งโปรแกรมเหล่านี้เสร็จเรียบร้อยแล้ว เครื่องคอมพิวเตอร์ส่วนบุคคลที่เรียกว่า localhost ก็สามารถทำงานเหมือนกับเครื่อง Sever ได้แล้ว ในที่นี้เราใช้ตัว Appserv นอกจากนี้ต้องใช้โปรแกรม No-ip ดังรูปที่ 3.14 เพื่อสร้าง domain name ซึ่งคือชื่อที่ถูกเรียกแทนการเรียกเป็นหมายเลขอินเทอร์เน็ต (IP Address) เนื่องจากการจดจำหมายเลข IP ถึง 16 หลัก ทำให้ยุ่งยาก จึงนำชื่อที่เป็นตัวอักษรมาใช้แทน เพื่อจะ fix IP ในการเข้าไปหน้าเว็บไซต์ฐานข้อมูลของเรา



รูปที่ 3.14 แสดงการใช้โปรแกรม No-ip ในการสร้าง Domain name

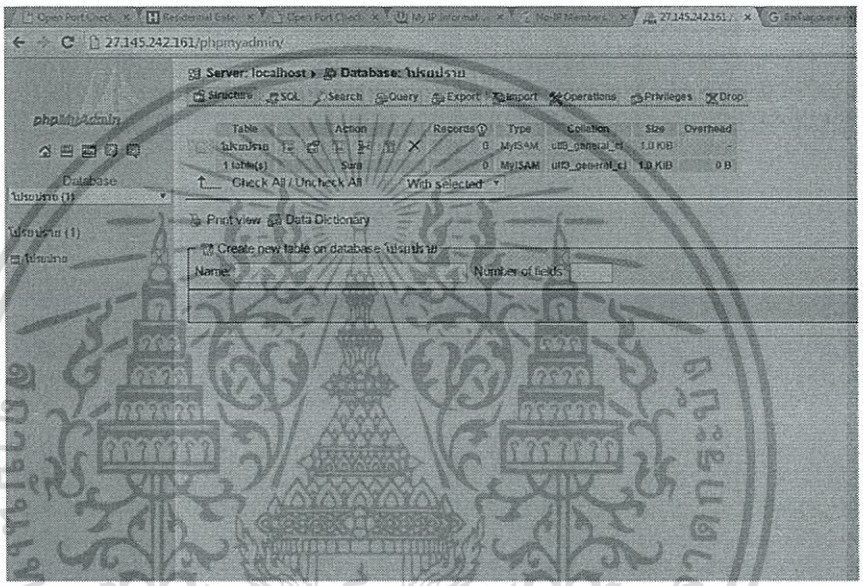
ระหว่างการทำขั้นตอนการสร้างตัว Server ต้องเปิดตัว Apeche และ MySQL ของ Appserve ไว้ด้วย หลังจากการทำ port forward เรียบร้อยให้เช็ค IP global และ Port 80 ว่ามีการใช้งานอยู่จริงจาก Canyouseeme.org ดังรูปที่ 3.15

รูปที่ 3.15 แสดงการเช็ค IP Global และ Port 80 มีการ forwarding เรียบร้อย

หลังจากนั้นนำตัว IP global ที่ได้ (27.165.242.161) ใส่ในช่อง URL จะปรากฏภาพดังรูปที่ 3.16 ซึ่งแปลว่าเราสามารถนำไปสร้างตัว localhost/phpMyadmin ต่อไปได้

รูปที่ 3.16 แสดงหน้า Web Server ของเราที่สามารถใช้งานต่อได้เพื่อทำเป็นฐานข้อมูลต่อไป

ทำการสร้างตารางเพื่อเก็บ Data ใน localhost/phpMyadmin ดังรูปที่ 3.17 เป็นส่วนต่อประสานที่สร้างโดยภาษาพีเอชพี ซึ่งใช้จัดการฐานข้อมูล MySQL ผ่านเว็บเบราว์เซอร์ โดยสามารถที่จะทำการสร้างฐานข้อมูลใหม่ หรือทำการสร้าง TABLE ใหม่ๆ และยังมี function ที่ใช้สำหรับการทดสอบการ query ข้อมูลด้วยภาษา SQL พร้อมกันนั้น ยังสามารถทำการ insert delete update หรือแม้กระทั่งใช้คำสั่งต่างๆเหมือนกับกันการใช้ภาษา SQL ในการสร้างตารางข้อมูล



รูปที่ 3.17 แสดงการสร้างตารางเก็บข้อมูลบน localhost/phpMyadmin

หลังจากนั้นทำการใช้คำสั่งดังรูปที่ 3.18 เพื่อทำการเชื่อมต่อระหว่างตัว Server และ Sim900 ด้วย php เพื่อส่งข้อมูลที่เป็นตัวแปรเข้าสู่ตาราง Server โดยสร้าง php สองไฟล์ ไฟล์แรกเป็นของ dbconnect.php เป็นการเชื่อมต่อเข้าสู่ Server โดยใช้โดเมนเนมชื่อ http://dometeeboss.ddns.net/phpmyadmin ในตารางชื่อ marinemachine

dbconnect.php - Notepad

File Edit Format View Help

```
<?php
$MyUsername = "root";
$MyPassword = "root";
$MyHostname = "localhost";
$My_db="[illegible]";

$dbh = mysqli_connect($MyHostname , $MyUsername , $MyPassword ,$My_db);
$selected = mysqli_select_db($dbh,"marinemachine");
?>
```

รูปที่ 3.18 แสดงคำสั่งใน dbconnect.php เพื่อเชื่อมต่อตัวอุปกรณ์กับ Server

ไฟล์ php ส่วนที่สอง add_data.php เป็นการกำหนดตัวแปรแต่ละเซนเซอร์เข้าสู่ตารางใน server ซึ่งใช้คำสั่งดังรูปที่ 3.19

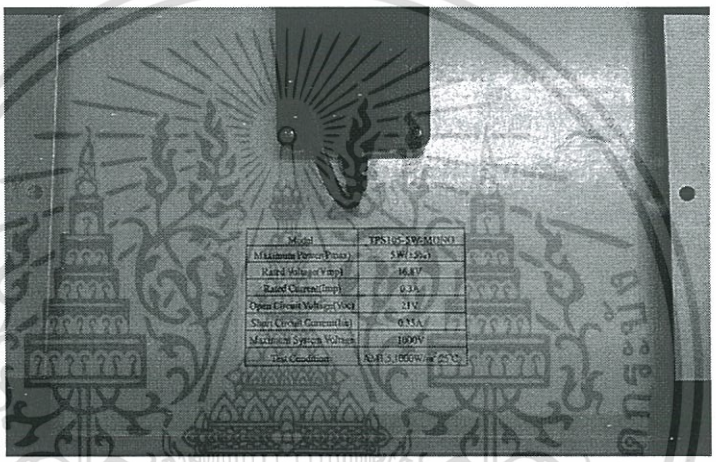
File Edit Format View Help

```
<?php
    include("dbconnect.php");

    $SQL = "INSERT INTO marinemachine (watertemperature(C),Airtemperature(c),Pressure(kpa),Pressure(mmhg),Humidity(%),GPS(degree))
    VALUES ('".$_GET["addwatertemperaturec"]."','".$_GET["addAirtemperaturec"]."','".$_GET["addPressurekpa"]."','".$_GET["addPressuremmhg"]."',
    '".$_GET["addhumidity%"]."','".$_GET["addGPSdegree"]."');

    mysqli_query($dbh,$SQL);
?>



<?
    echo date('d;m;y H:i:s');
    $time =date('d/m/y H:i:s');

    $servername = "localhost";
    $username ="root";
    $password ="12345678";
    $dbname ="marine machine";

    $conn = mysqli_connect($servername,$username,$password,$dbname);
    if(!$conn)
    {echo "fault";
    exit;
    }

    date_default_timezone_set("Asia/bangkok");
    $sql = "INSERT INTO lalo(time,velo,humid,pressureinHg,pressurekpa,
    AirtemperatureC,WatertemperatureC,warning,latitude,longitude,wavenature)
            VALUE ('$time','{$_GET["velo"]}','{$_GET["H"]}','{$_GET["P"]}',
    '{$_GET["K"]}','{$_GET["A"]}','{$_GET["w"]}','{$_GET["X"]}','{$_GET["La"]}',
    '{$_GET["Lo"]}','{$_GET["V"]}')";
    $result = mysqli_query($conn,$sql);

mysqli_close($conn);
?>
```

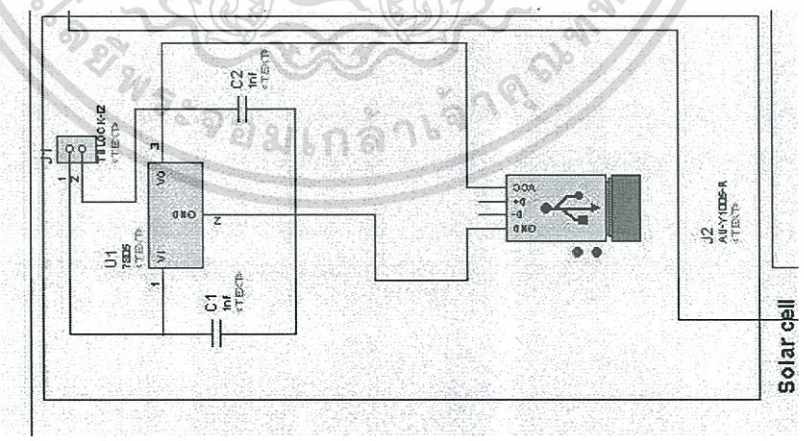
รูปที่ 3.19 แสดงคำสั่งใน add_data.php เพื่อส่งข้อมูลจากตัวอุปกรณ์เข้าตาราง Server

3.1.6 การออกแบบการเชื่อมต่อระหว่าง Arduino กับ Solar cell และ Power bank

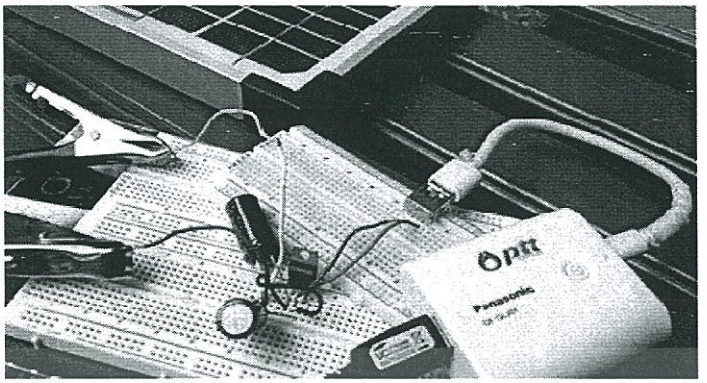
เนื่องจากโซล่าเซลล์ที่ใช้ในโครงงานครั้งนี้จ่ายไฟ 16.8 V และให้ค่าแรงดันไม่คงที่ จึงต้องนำมาแปลงไฟก่อน การแปลงไฟจาก 16.8 V (จาก Solar cell) เป็น 5 V ด้วย LM7805 เป็นการนำพลังงานแสงเป็นกระแสไฟฟ้า โดยใช้ IC เบอร์ 7805 และตัวเก็บประจุขนาด 220 ไมโครฟารัด 16 V ในการแปลงกระแสไฟฟ้า จาก 16 V เป็น 5 V โดย IC 7805 นั้นสามารถใช้ได้ตั้งแต่ 8 - 24 V โดยประมาณ แต่โวลต์ยิ่งสูง ไอซียิ่งร้อนอาจจะทำให้ไอซีไหม้ได้ หลังจากนั้นก็จะเก็บพลังงานที่ได้ไว้ใน Power bank เพื่อใช้ในการทำงานต่อไป ซึ่งทำการต่อวงจรดังรูปที่ 3.21



รูปที่ 3.20 แผง solar cell



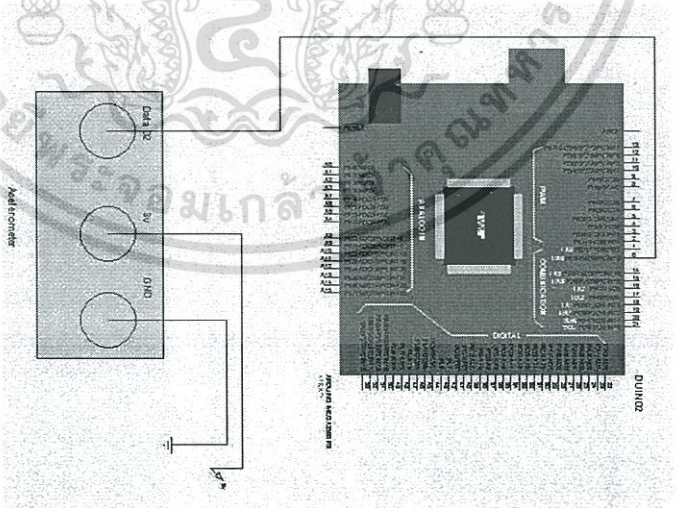
รูปที่ 3.21 วงจรของการเชื่อมต่อ LM7805 , Solar cell และ Power bank ในโปรแกรม Proteus



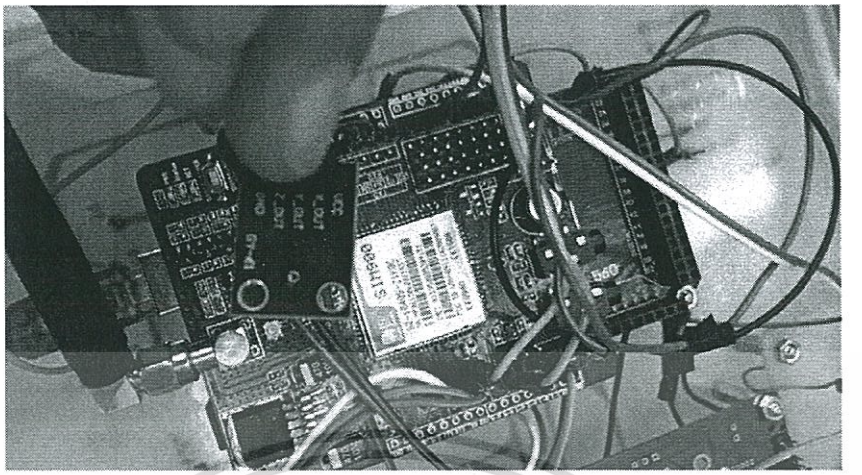
รูปที่ 3.22 วงจรจริงของการเชื่อมต่อ LM7805, Solar cell และ Power bank

3.1.7 การออกแบบการเชื่อมต่อระหว่าง Arduino กับ Accelerometer

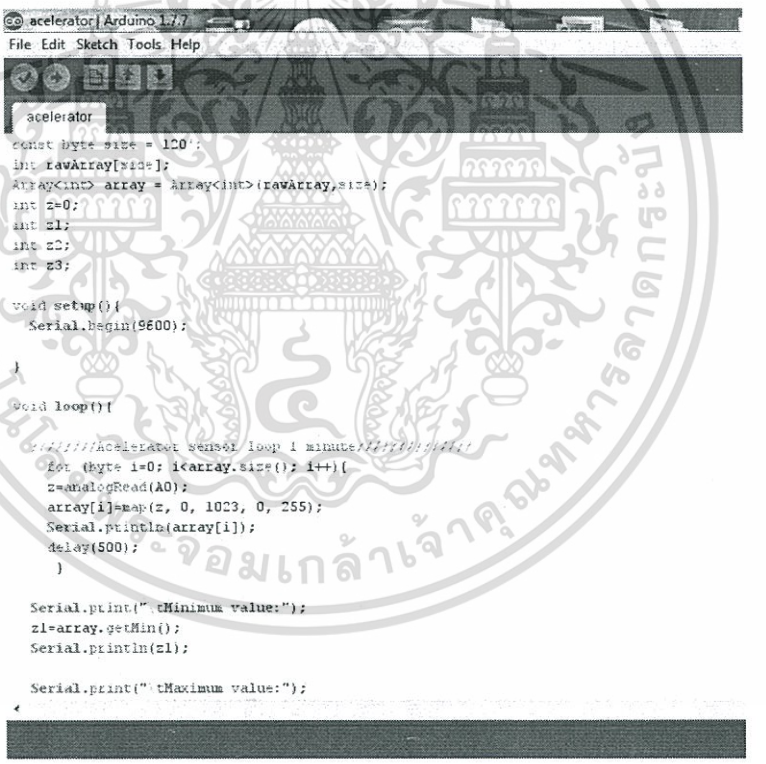
ADXL335 เป็นเซนเซอร์วัดความเร่งแบบ 3 แกน(3-axis Accelerometer) มีขนาดเล็ก ใช้พลังงานต่ำเซนเซอร์วัดความเร่ง 3 แกน สามารถวัดอัตราเร่งคงที่ของแรงโน้มถ่วงของโลก รองรับไฟ 3.3 V ให้ค่าเซนเซอร์แกน x,y,z ออกมาเป็น Analog ทำการต่อกับ Arduino ใช้ขาอะนาล็อก เช่น A0-A5 อ่านค่าเข้ามาได้ ทำการออกแบบการเชื่อมต่อ Arduino กับ Accelerometer ดังรูปที่ 3.23 และทำการต่อวงจรจริงแสดงดังรูปที่ 3.24 ซึ่งทำการเขียนคำสั่งให้ Arduino เพื่อรับค่าจาก Accelerometer โดยคำสั่งที่ใช้แสดงดังรูปที่ 3.25



รูปที่ 3.23 วงจรของการเชื่อมต่อระหว่าง Arduino กับ Accelerometer ในโปรแกรม Proteus



รูปที่ 3.24 วงจรจริงของการเชื่อมต่อระหว่าง Arduino กับ Accelerometer

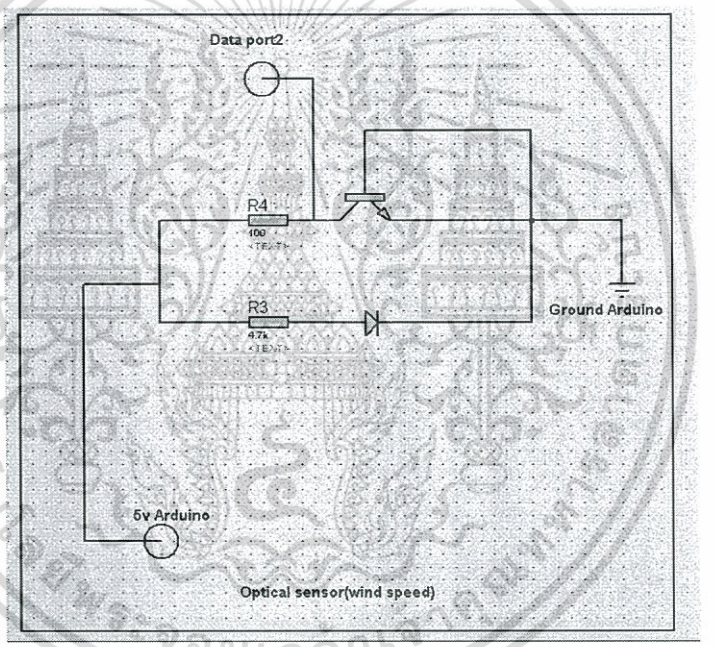


```
acelerator | Arduino 1.7.7
File  Edit  Sketch  Tools  Help

 acelerator

const byte size = 120;
int rawArray[size];
Array<int> array = Array<int>(rawArray,size);
int z=0;
int z1;
int z2;
int z3;

void setup(){
  Serial.begin(9600);

}

void loop(){

  ////////Accelerator sensor loop 1 minute////////////
    for (byte i=0; i<array.size(); i++){
    z=analogRead(A0);
    array[i]=map(z, 0, 1023, 0, 255);
    Serial.println(array[i]);
    delay(500);
     }

  Serial.print("\tMinimum value:");
  z1=array.getMin();
  Serial.println(z1);

  Serial.print("\tMaximum value:");
```

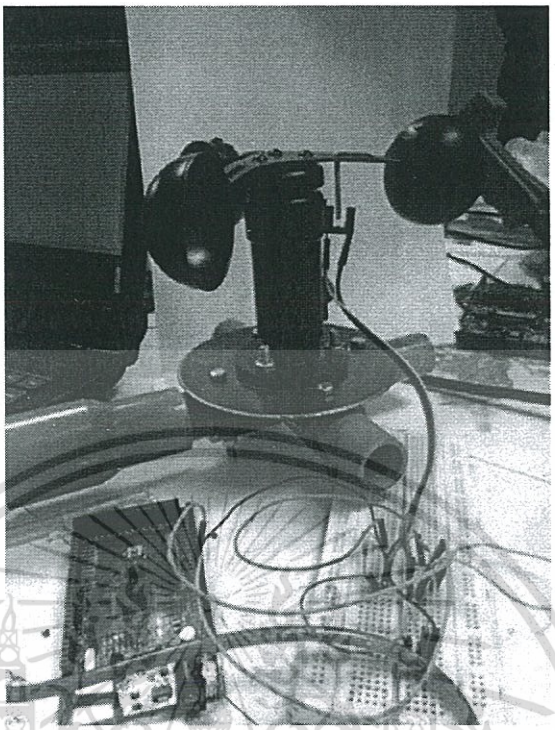
รูปที่ 3.25 คำสั่งโปรแกรม Arduino ที่ใช้ในการรับค่าจาก Accelerometer

3.1.8 การออกแบบการเชื่อมต่อระหว่าง Arduino กับ Optical sensor H42B6

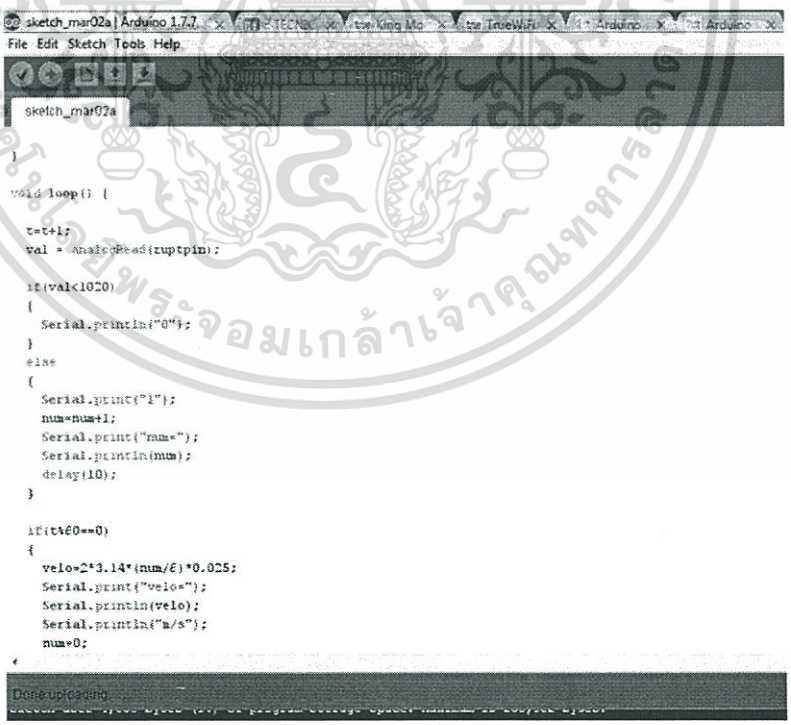
เนื่องจากค่าของเซ็นเซอร์วัดความเร็วลมมีค่าความคลาดเคลื่อนมาก (เอาต์พุตออกเป็นมิลลิแอมป์) ส่งผลให้ใช้ไฟเลี้ยงมากเกินไป จึงได้ทำการแก้ไขโดยการเปลี่ยนเซ็นเซอร์ลมเป็นออปติคอลเซ็นเซอร์ ซึ่งเป็นการส่งผ่านพลังงานสูง อินฟราเรดของโฟโต้ไดโอดและทรานซิสเตอร์มีความไวสูง (ออปติคอลเซ็นเซอร์) เป็นประเภทของเซ็นเซอร์แสง หลักการในการดำเนินงานคือ การแปลงแสงเป็นแรงดัน ออปติคอเซ็นเซอร์ ใช้สำหรับการแปลง หรือ การวัด Photoelectric sensors ทำการออกแบบวงจรดังรูปที่ 3.26 และทำการต่อวงจรจริงดังรูปที่ 3.27 โดยการวัดพลังงานไฟฟ้าและสามารถนำมาใช้ในการวัดระยะทาง วัดค่าความเร็วต่างๆได้ ทำการเขียนคำสั่งดังรูปที่ 3.28 เพื่อทำการรับค่าจาก Optical sensor H42B6



รูปที่ 3.26 วงจรของการเชื่อมต่อระหว่าง Arduino กับ Optical sensor H42B6 ในโปรแกรม Proteus



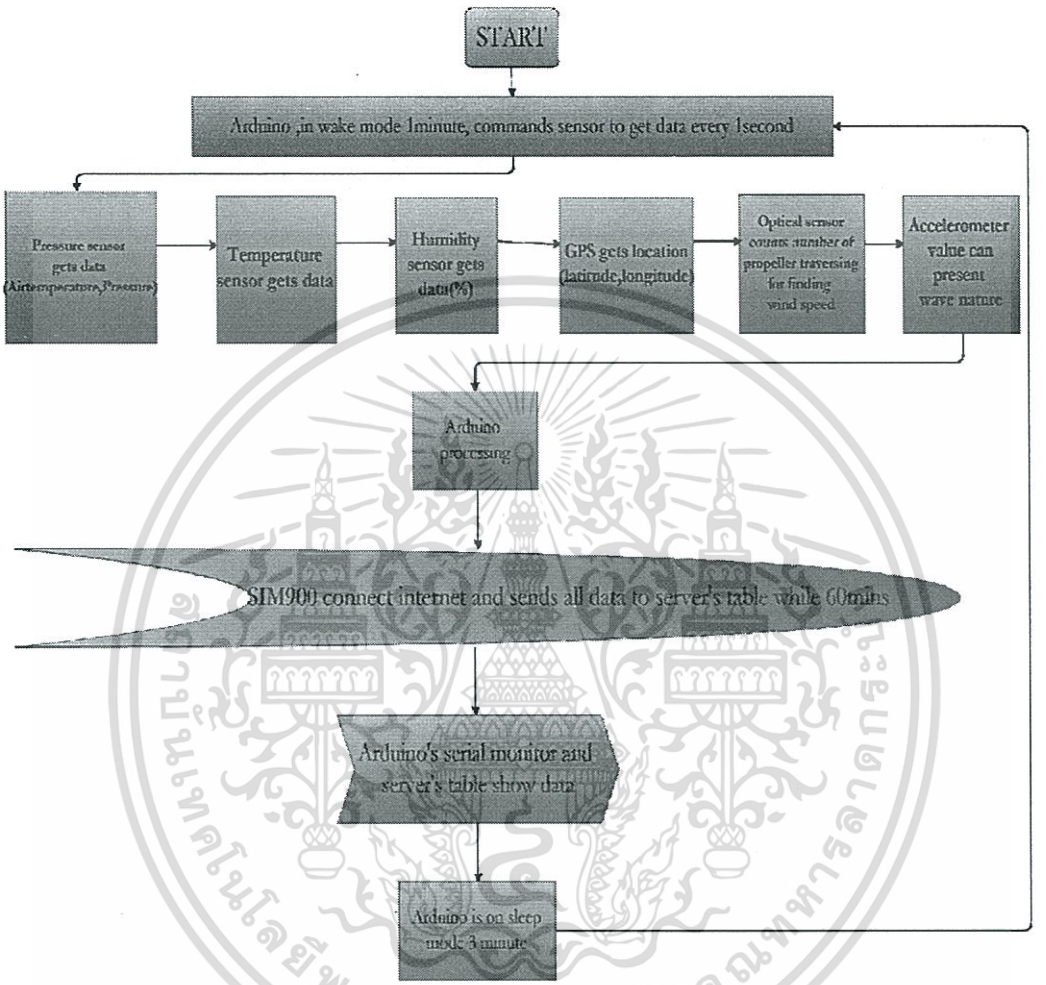
รูปที่ 3.27 วงจรจริงของการเชื่อมต่อระหว่าง Arduino กับ Optical sensor



```
sketch_mar02a | Arduino 1.7.7
File  Edit  Sketch  Tools  Help
sketch_mar02a

}
void loop() {

t=t+1;
val = analogRead(ruptpin);

if(val<1020)
{
  Serial.println("0");
}
else
{
  Serial.print("1");
  num=num+1;
  Serial.print("num=");
  Serial.println(num);
  delay(10);
}

if(t%60==0)
{
  velo=2*3.14*(num/6)*0.025;
  Serial.print("velo=");
  Serial.println(velo);
  Serial.println("m/s");
  num=0;
}
Done uploading.
```

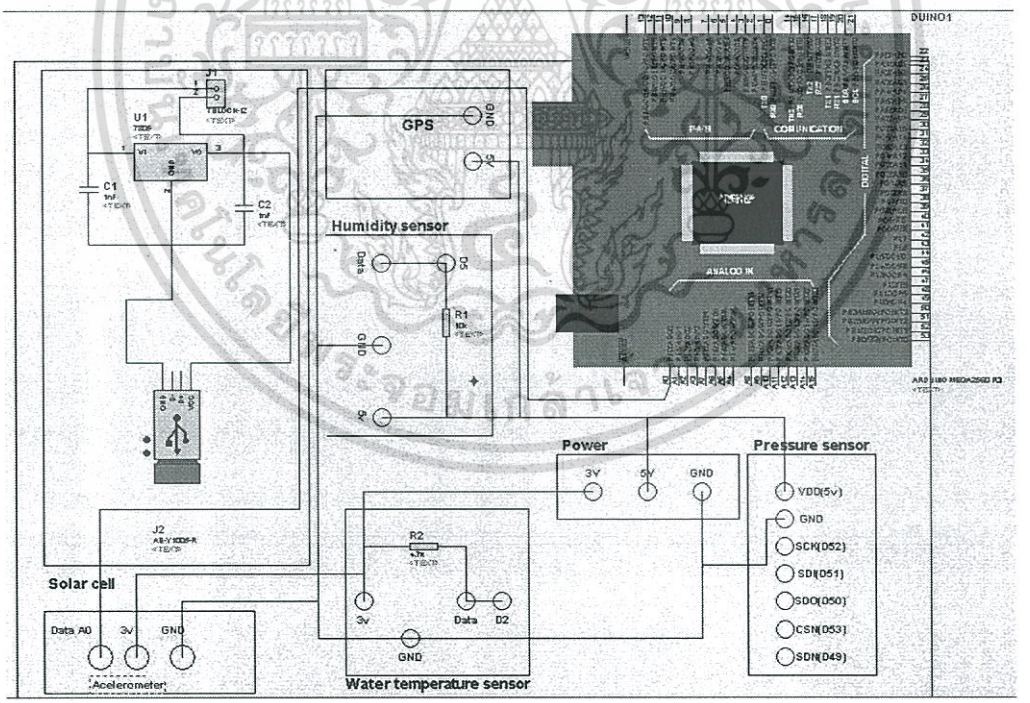
รูปที่ 3.28 คำสั่งโปรแกรม Arduino ในการรับค่าจาก Optical sensor H42B6

3.2 การทำงานของวงจรรวม

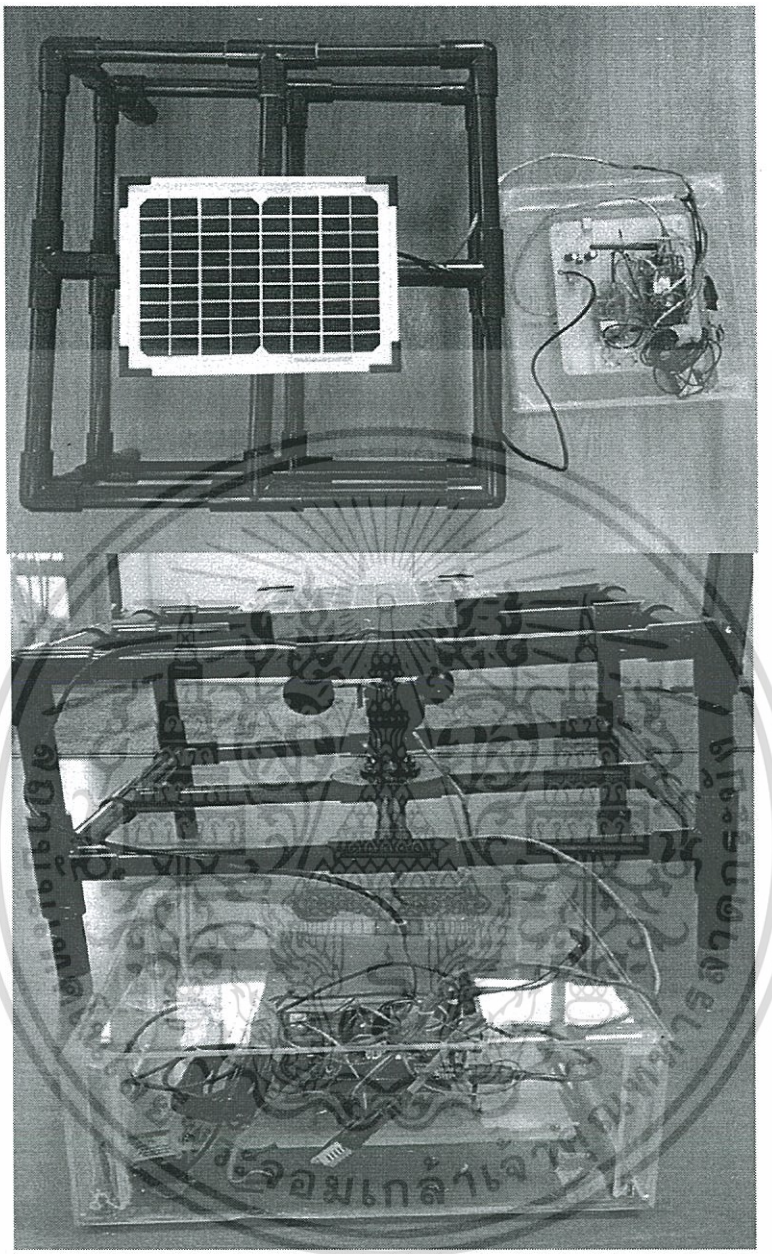


รูปที่ 3.29 Flow chart การทำงานของวงจรรวม

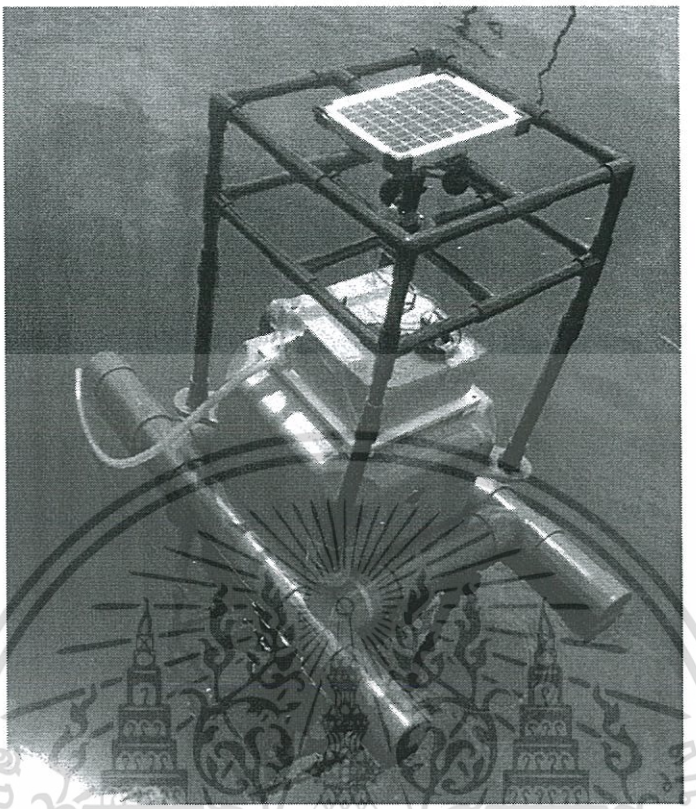
หลักการทำงานเมื่อทำการเปิดเครื่องให้ทำงาน ตัว Arduino จะได้รับไฟเลี้ยงจาก Solar cell และ Power bank โดยที่ตัว Arduino จะจ่ายไฟเลี้ยงให้กับเซนเซอร์ทุกตัว และโมดูล GPS อุปกรณ์ทุกตัวเริ่มทำงานเก็บค่าต่างๆจากสภาพแวดล้อม Arduino จะทำการประมวลผล Pressure sensor จะประกอบไปด้วยค่าอุณหภูมิอากาศ (องศาเซลเซียส) และ ค่าความกดอากาศ (inHg,kpa) Humidity sensor จะเก็บค่าความชื้นในอากาศเป็นเปอร์เซ็นต์ (%) Temperature sensor จะเก็บเฉพาะค่าอุณหภูมิน้ำ โดยแบ่งเป็น ถ้าอุณหภูมิต่ำกว่า -1 องศาเซลเซียส หรือ มากกว่า 30 องศาเซลเซียส จะทำการแจ้งเตือน และโมดูล GPS จะเก็บค่าตำแหน่ง ณ ขณะนั้นให้ถูกต้อง แต่บางครั้ง GPS อาจค้นหาตำแหน่งไม่ได้หรือได้ค่าที่คลาดเคลื่อนไป เนื่องจากมีสิ่งมารบกวนการทำงานของตัวสายอากาศ จึงต้องทำการหาค่าตำแหน่งใหม่ นอกจากนี้ยังมี Anemometer ชนิดลูกถ้วย 3 ลูกที่ติด Optical sensor มาใช้ในการวัด จำนวนรอบที่ตัดผ่านต่อนาที เพื่อนำมาคำนวณความเร็วลม (เมตรต่อวินาที) ส่วนในด้านของคลื่นจะใช้ค่าพฤติกรรมของเซนเซอร์ Accelerometer มาใช้ในการหาว่าคลื่นในช่วงนั้นมีลักษณะสงบ มีคลื่นปานกลางหรือมีคลื่นสูง ค่าจากเซนเซอร์ทุกตัวจะถูกส่งด้วย SIM900 ไปเก็บในตารางของตัว Server ทุกๆ 3 นาที พร้อมทั้งแสดงบน Serial Monitor บนโปรแกรม Arduino ได้ด้วย



รูปที่ 3.30 แสดงวงจรรวมในโปรแกรม Proteus



รูปที่ 3.31 แสดงวงจรรวมจริง



รูปที่ 3.32 แสดงทุ่นขณะทำการทดลองลอยน้ำเพื่อเก็บผลการทดลองจริง

บทที่ 4

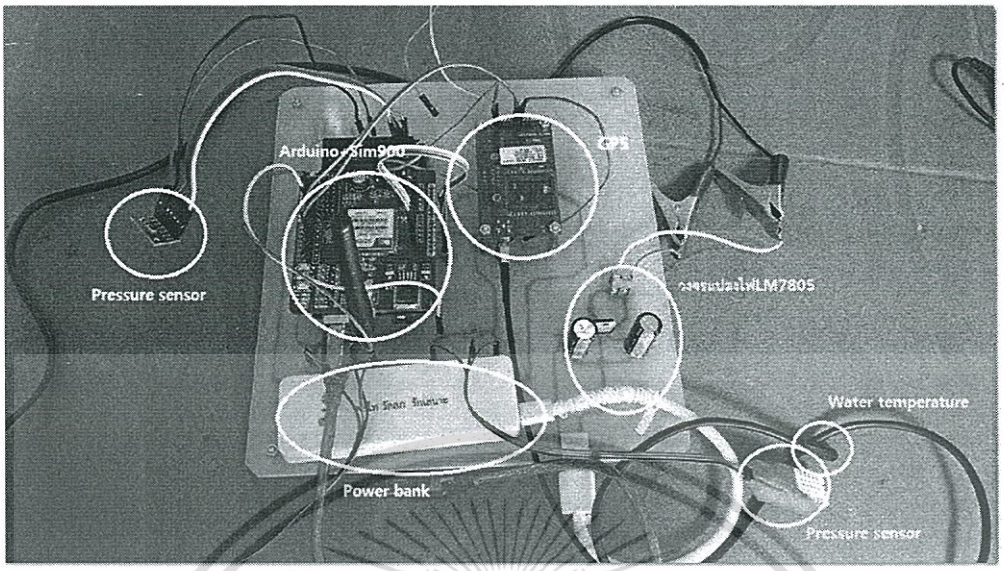
วิธีการทดลองและผลการทดลอง

4.1 วิธีการทดลองและผลการทดลองการเก็บค่าจากตัวเซ็นเซอร์แสดงบน Serial Monitor

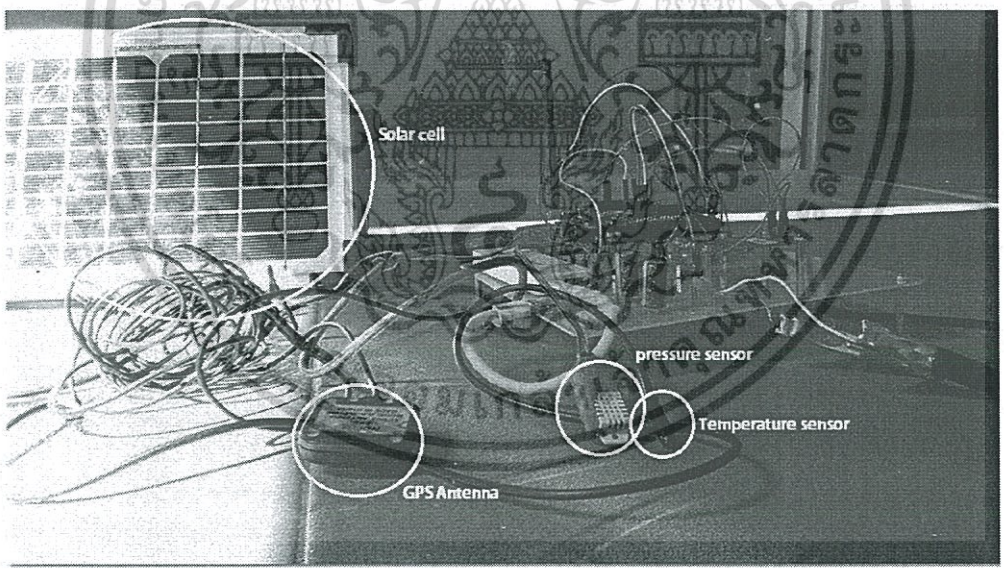
ประกอบด้วยเซนเซอร์ที่ใช้ในการทดลองดังนี้

- Water temperature sensor (bs18B20)
- Pressure sensor (MPL115A1)
- Humid sensor (AM2302)
- Accelerometer (ADXL335)
- Optical sensor (H42B6)
- GPS/Antenna

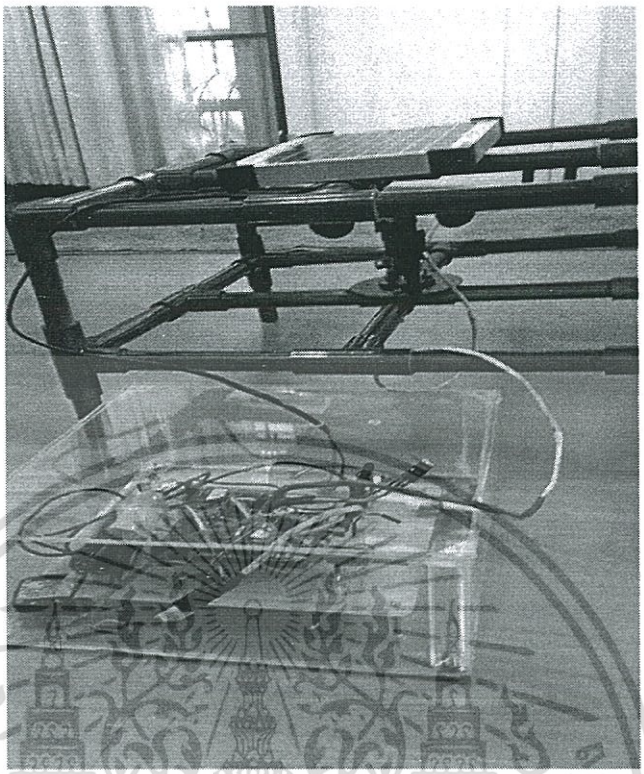
โดยทำการต่ออุปกรณ์ตามที่ได้ทำการออกแบบไว้ พร้อมทั้งดูเอกสารรายละเอียดของตัวอุปกรณ์แต่ละตัวเพิ่มเติมขณะต่ออุปกรณ์ เขียนโปรแกรม Arduino ทดสอบก่อนว่าเซนเซอร์ทั้งหลายสามารถรับค่าและส่งค่ามาประมวลผล พร้อมทั้งแสดงผลบน Serial monitor ได้จากรูปที่ 4.1 และ รูปที่ 4.2 เซนเซอร์แต่ละชนิดที่ใช้บนแผงวงจร โดยใช้แบตเตอรี่สำรองจ่ายไฟเลี้ยงให้กับ Arduino ส่วนในรูปที่ 4.3 แผงวงจรจะถูกบรรจุลงในกล่องอะคริลิกและนำเซนเซอร์บางชนิดออกมาด้านนอก เนื่องจากเซนเซอร์วัดความชื้น เซนเซอร์วัดความดันอากาศ และเซนเซอร์วัดอุณหภูมิน้ำ ต้องสัมผัสกับสภาพแวดล้อมเพื่อเก็บผล ส่วนเสาอากาศของ Sim900, Antenna ของ GPS ก็ต้องนำออกมาไว้ด้านนอกกล่องอุปกรณ์เช่นกัน เพื่อให้ Sim900 เชื่อมต่อสัญญาณโทรศัพท์ได้ง่ายขึ้น และ Antenna ของ GPS สามารถรับสัญญาณได้เช่นกัน ค่าต่างๆที่เซ็นเซอร์รับค่าได้นั้นแสดงผลบน Serial Monitor ดังรูปที่ 4.4 และแสดง Location ละติจูด, ลองจิจูด จาก GPS ดังรูปที่ 4.5



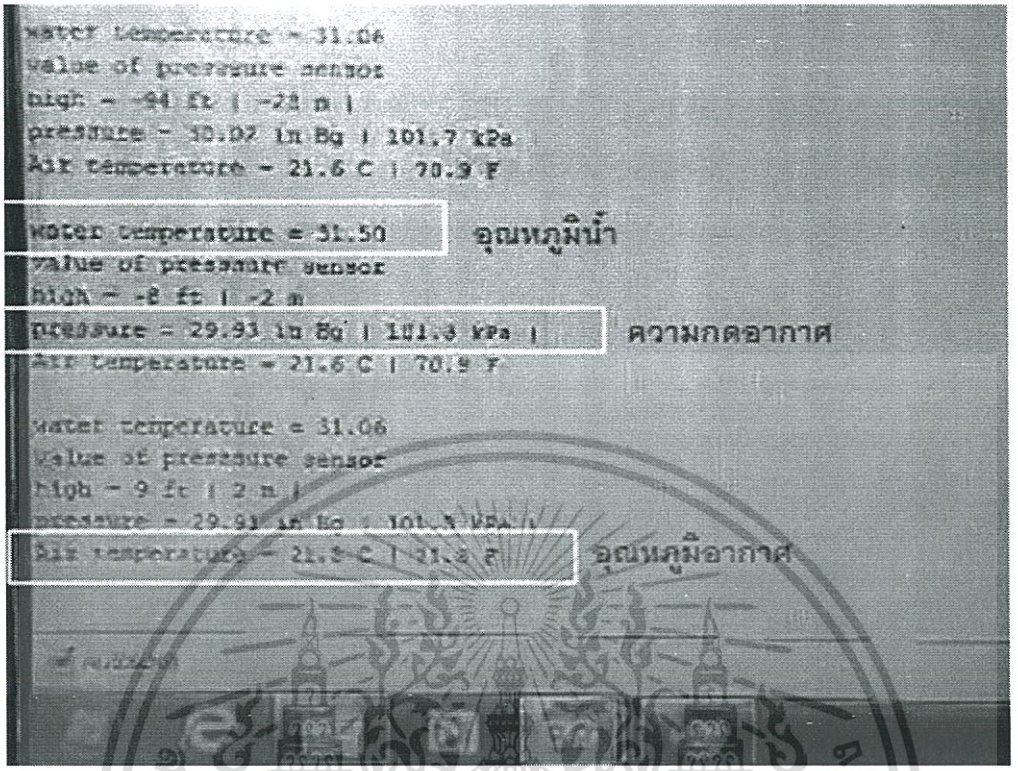
รูปที่ 4.1 แสดงการต่อวงจรรวมของอุปกรณ์ต่างๆ



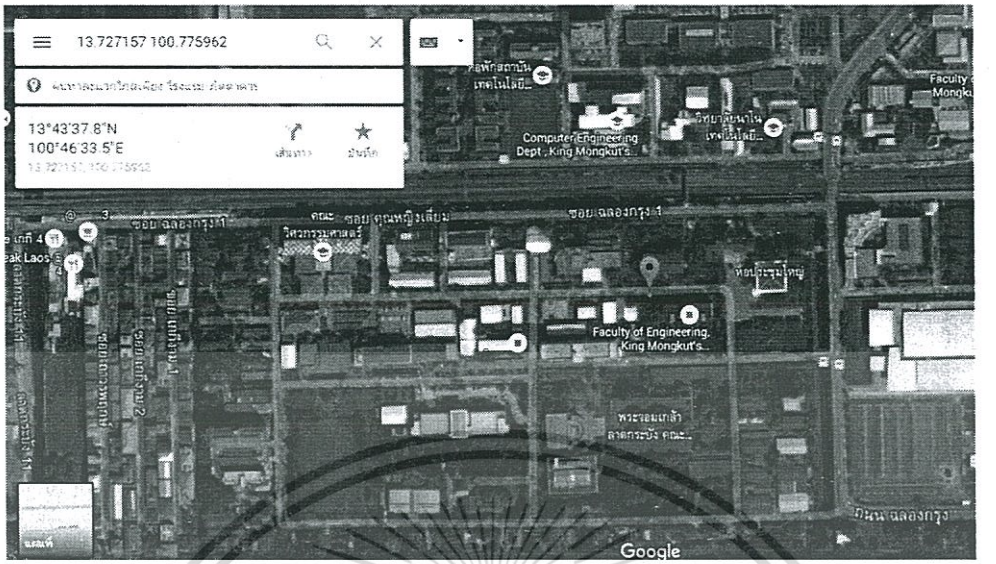
รูปที่ 4.2 แสดงการต่อวงจรรวมของอุปกรณ์ต่างๆ



รูปที่ 4.3 แสดงการต่อวงจรรวมของอุปกรณ์ต่างๆ เมื่อบรรจุใส่บรรจุภัณฑ์



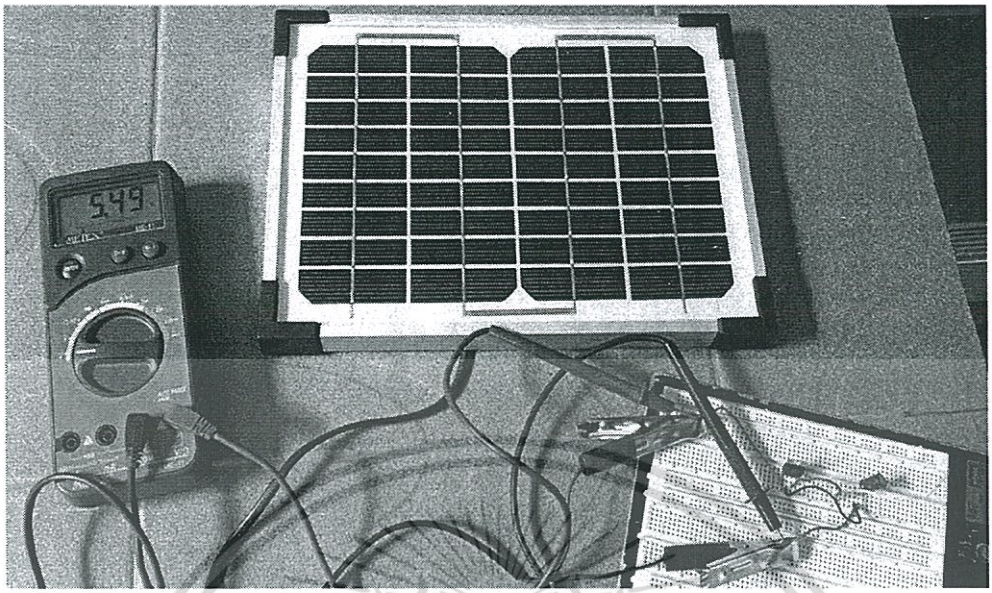
รูปที่ 4.4 แสดงค่าอุณหภูมิน้ำ, ค่าความกดอากาศ, ค่าอุณหภูมิอากาศ, ความเร็วลม, และตำแหน่งจาก GPS บน Serial Monitor



รูปที่ 4.5 แสดง Location ละติจูด, ลองจิจูด จาก GPS

4.2 วิธีการทดลองแหล่งจ่ายพลังงาน

อุปกรณ์จะใช้แหล่งพลังงานจากโซล่าเซลล์เป็นหลัก โดยโซล่าเซลล์จะต่อเข้ากับไอซี 7805 เพื่อแปลงแรงดันของโซล่าเซลล์จาก 16 โวลต์ ออกมาประมาณ 5 - 6 โวลต์ ดังแสดงในรูปที่ 4.6 เพื่อจ่ายให้กับแบตเตอรี่สำรองเพื่อกักเก็บพลังงานไว้ใช้ในตอนกลางคืนและจ่ายให้กับ Arduino นอกจากนี้ยังต้องทำการทดสอบว่าแบตเตอรี่สำรองขนาดเท่าใดมีประสิทธิภาพที่ชาร์จได้เต็มที่ในช่วงที่มีแสง และสามารถจ่ายพลังงานในตอนกลางคืนโดยเพียงพอ จากการทดสอบ หากใช้แบตเตอรี่สำรองขนาดประมาณ 30,000 mA หรือมากกว่า แม้จะไม่สามารถชาร์จได้เต็มที่ในช่วงที่มีแสงมากนัก แต่สามารถให้พลังงานในตอนกลางคืนได้ตลอดช่วง และเพื่อเป็นการประหยัดพลังงาน จึงทำการเก็บข้อมูลและส่งในทุกๆ 3 นาที เท่านั้น การทำงานจึงมีช่วง Sleep/Wake mode



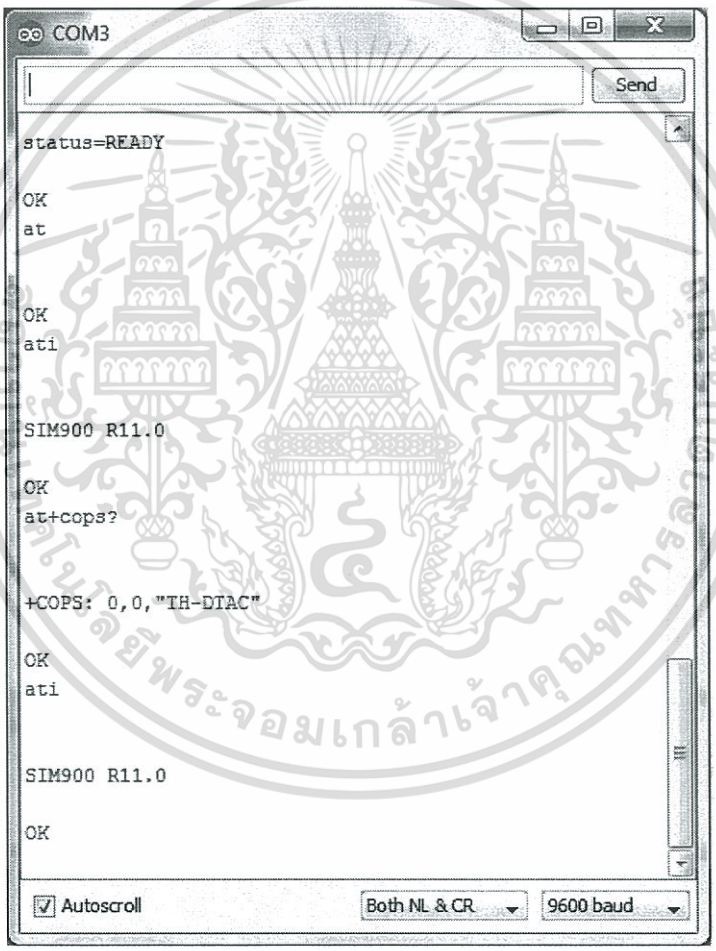
รูปที่ 4.6 แสดงการทดสอบการแปลงแรงดันของโซล่าเซลล์ด้วยไอซี 7805

4.3 วิธีการทดลองและผลการทดลองการในการส่งข้อมูลเข้า Server

เริ่มจากทำการสร้างฐานข้อมูลหลักด้วย phpMyadmin ในส่วนของการสร้างฐานข้อมูลจะมีเรื่องของการเปิดพอร์ต 3306 หรือ MySql และพอร์ต 80 หรือ Apeche สามารถอ่านเพิ่มเติมได้ในบทที่ 3.1.5 การออกแบบการเชื่อมต่อระหว่าง SIM900 GSM/GPRS Module และ Arduino MEGA 2560) ฐานข้อมูล MySQL ในชื่อของ marine machine จะถูกสร้างตารางขึ้น โดยอาจกำหนดวันที่เวลาเป็นหลักในการเข้าของข้อมูล ดังรูปที่ 4.9 เป็นโครงสร้างตาราง ชื่อคอลัมน์และกำหนดชนิดของข้อมูลที่จะเข้ามา ในส่วนของโค้ด HTML ในรูปที่ 4.10 จะถูกจัดเก็บไว้ในโฟลเดอร์ www ของ Appserv เพื่อใช้เชื่อมการส่งค่าระหว่างฐานข้อมูลกับตัวเซนเซอร์ที่ถูกควบคุมด้วยตัว Arduino ควรพยายามกำหนดตัวแปรให้ตรงกันเพื่อไม่ให้เกิดความผิดพลาดในการส่ง

หลังจากนั้นลองทำการส่งข้อมูลด้วย Sim900 พร้อมทั้งใส่ซิมการ์ดโทรศัพท์ของเครือข่ายที่มีคลื่นความถี่ประเภท 900 MHz และใช้คำสั่งในการเชื่อมต่อด้วย AT Commend บนโปรแกรม Arduino (ก่อนการส่งข้อมูลควรทดสอบว่าตัว Sim900 Tx, Rx มีการติดต่อกับตัว Arduino ที่แสดงดังรูป 4.11 และ มีไฟเลี้ยงเพียงพอ หลังจากที่ Sim900 กับ Arduino มีการตอบสนองต่อกัน ให้ทดสอบว่า Sim900 สามารถเชื่อมต่อเครือข่ายโทรศัพท์ได้โดยอาจลองทดสอบกับโค้ดตัวอย่าง GSM_GPRSLibrary AT และ GSM_GPRSLibrary client)

คำสั่งตรวจสอบเครือข่ายที่ SIM900 ใช้อยู่ พิมพ์ AT + COPS? แล้วกด Enter SIM900 จะแสดงเครือข่ายที่กำลังเชื่อมต่อ ดังแสดงในรูปที่ 4.7 ในที่นี้แสดงเครือข่ายของ DTAC และแสดงสถานะเชื่อมต่อกับเครือข่ายได้แล้ว



รูปที่ 4.7 แสดงการเชื่อมต่อระหว่าง Sim900 กับ Arduino ผ่าน Tx, Rx

เมื่อเชื่อมต่อกับเครือข่ายมือถือได้แล้ว ก็สามารถสั่งการมือถือให้โทรออก / รับสาย / วางสายได้ โดยใช้คำสั่ง “adtหมายเลข;” เช่น โทรหมายเลข 0854545454 พิมพ์คำสั่งดังนี้

atd0854545454 ;

OK

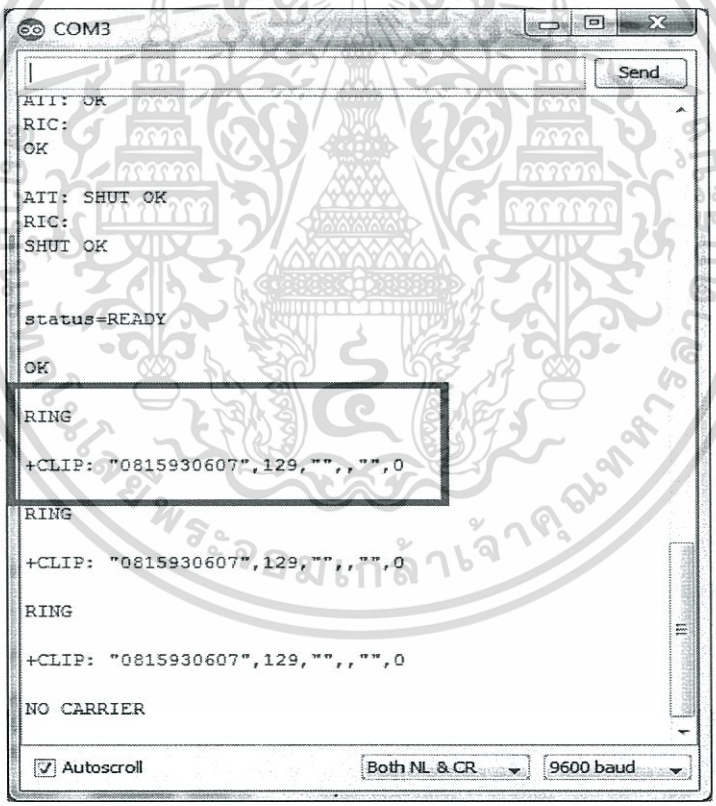
ต่อมาทดลองโทรเข้ามาที่หมายเลขใน SIM900 ก็จะพบว่า มีคำว่า RING และแสดงหมายเลขที่โทรเข้ามาด้วย แสดงดังรูปที่ 4.8 ถ้าต้องการกดวางสายก็พิมพ์ ATH

RING

+CLIP: "0854545454",129,"",,"",0

ATH

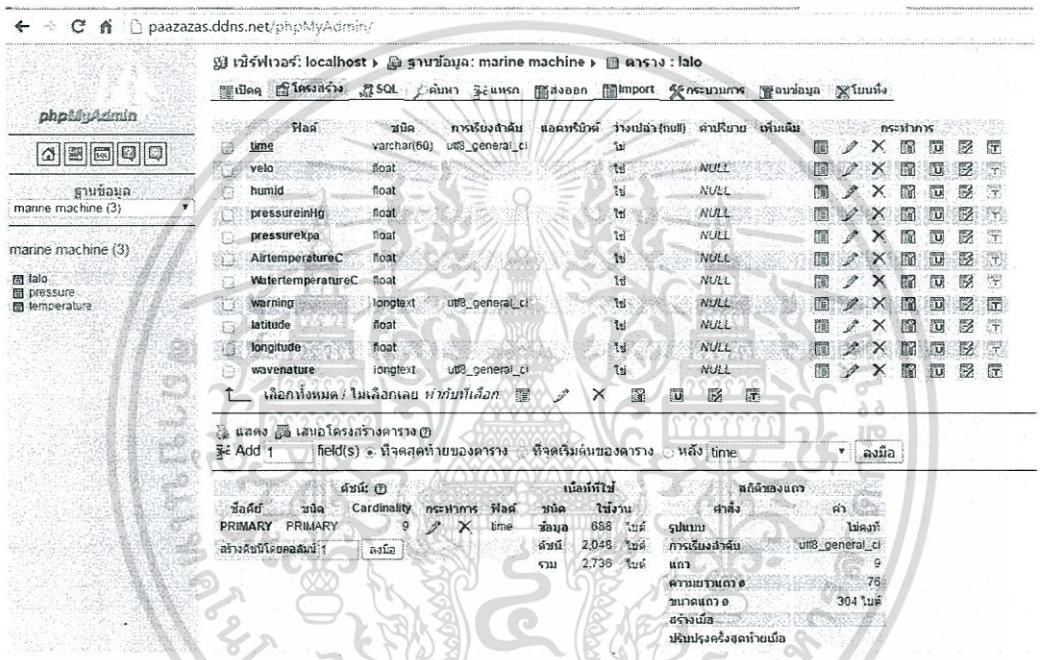
OK



รูปที่ 4.8 แสดงการเชื่อมต่อเครือข่ายของซิมการ์ด กับ Sim900 ทดสอบโดยการโทรออก

ต่อมาเพื่อการส่งค่าข้อมูลเข้าสู่ฐานข้อมูล (ต้องกำหนดค่าตัวแปรในตารางเซิร์ฟเวอร์ คำสั่ง HTML และในคำสั่ง Arduino ให้ตรงกัน) ใช้คำสั่งดังแสดงรูปที่ 4.10

ผลที่ได้ คือ ค่าจากตัวเซนเซอร์จะเข้าสู่ sleep mode จนกว่าจะ 3 นาที Arduino จะทำการเปิดระบบและเซนเซอร์ต่างๆจะทำงานเก็บค่าภายในหนึ่งนาทีโดยเก็บค่า 1 ครั้ง ต่อ 1 วินาที แล้วจึงส่งค่าที่ประมวลผลได้ให้กับ Sim900 เพื่อทำการส่งเข้าฐานข้อมูล ดังรูป 4.9 เมื่อทำการใช้ออสซิลโลสโคปวัดสัญญาณ Tx และ Rx จาก Sim900 ระหว่างที่ทำการรับส่งข้อมูล จะแสดงสัญญาณดังรูปที่ 4.11 และแสดงข้อมูลที่ได้รับในตารางบน Server ในรูปที่ 4.12

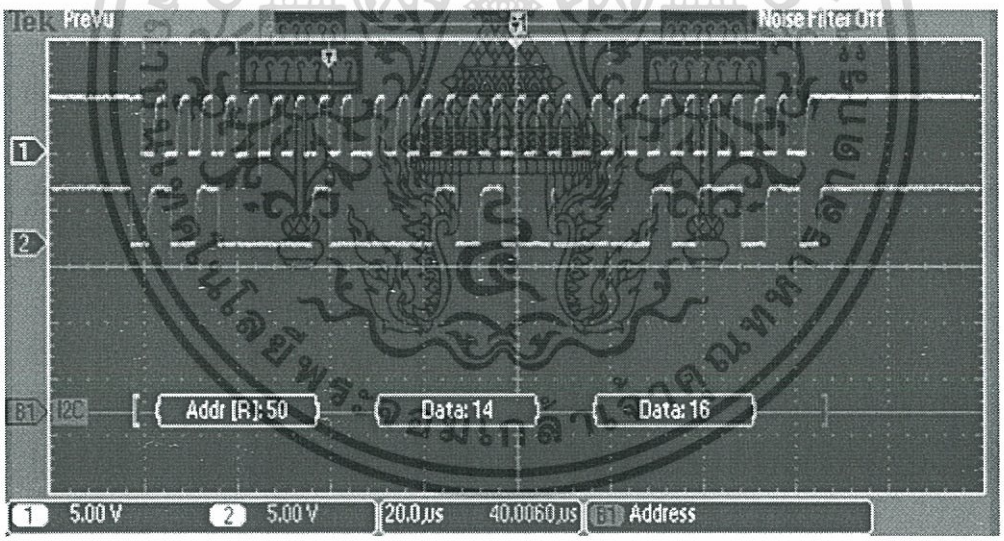


รูปที่ 4.9 โครงสร้างตารางฐานข้อมูล

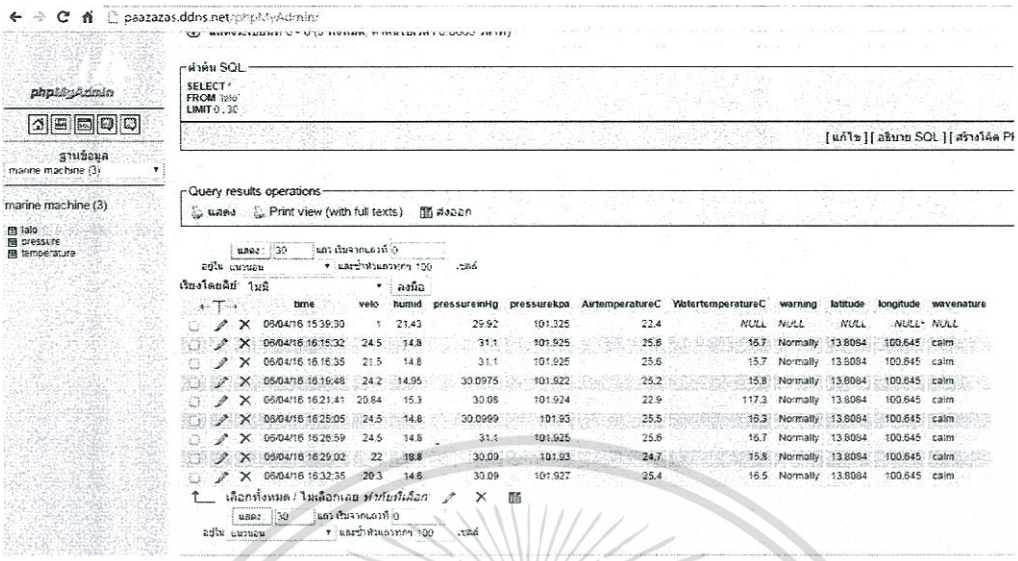
File Edit Format View Help

```
<?
        echo date('d:m:y H:i:s');
        $time =date('d/m/y H:i:s');

        $survername = "localhost";
        $username ="root";
        $password ="12345678";
        $dbname ="marine machine";

        $conn = mysqli_connect($survername,$username,$password,$dbname);
        if(!$conn)
        {echo "fault";
        exit;
        }

        date_default_timezone_set("Asia/bangkok");
        $sql = "INSERT INTO lalo(time,velo,humid,pressureinHg,pressurekpa,
        AirtemperatureC,WatertemperatureC,warning,latitude,longitude,wavenature)
                VALUE ('$time','{$_GET["velo"]}','{$_GET["H"]}','{$_GET["P"]}',
        {$_GET["K"]}','{$_GET["A"]}','{$_GET["W"]}','{$_GET["X"]}','{$_GET["La"]}',
        '{$_GET["Lo"]}','{$_GET["V"]}')";
        $result = mysqli_query($conn,$sql);


mysqli_close($conn);
?>
```

รูปที่ 4.10 คำสั่ง HTML ที่ใช้ในการส่งข้อมูลเข้าฐานข้อมูล



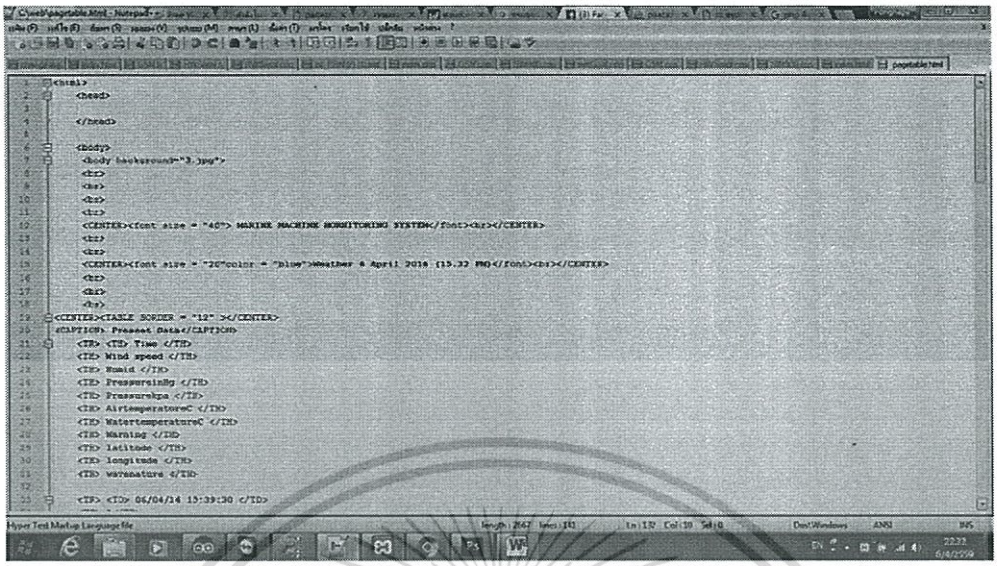
รูปที่ 4.11 สัญญาณจากขา Tx, Rx ของ Sim900



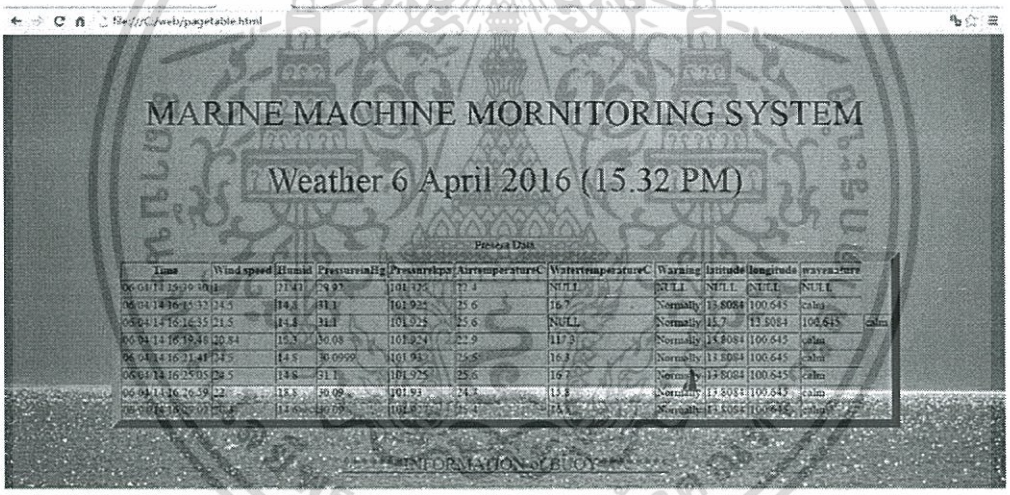
รูปที่ 4.12 ตารางเก็บข้อมูลบน Server

4.4 วิธีการทดลองและผลการทดลองในการแสดงค่าบนหน้าเว็บบราวเซอร์

หลังจากการเก็บผลเข้าสู่ฐานข้อมูลแล้ว การแสดงผลเพื่อให้สามารถดูข้อมูลได้โดยง่าย ดังรูปที่ 4.14 และอัพเดตตลอดเวลา โดยการเขียนหน้าเว็บบราวเซอร์เรียกข้อมูลจากฐานข้อมูลมาแสดงเป็นตารางบนหน้าเว็บ ด้วยการใช้โปรแกรม Notepad++ เขียน HTML ดังรูปที่ 4.13 แล้วฝากเว็บเพจไว้กับ Web server ที่ให้บริการของ freewebHostingrea.com เพื่อให้หน้าเว็บเพจของเราออนไลน์



รูปที่ 4.13 คำสั่ง HTML ที่ใช้สร้างหน้าเว็บเพจ

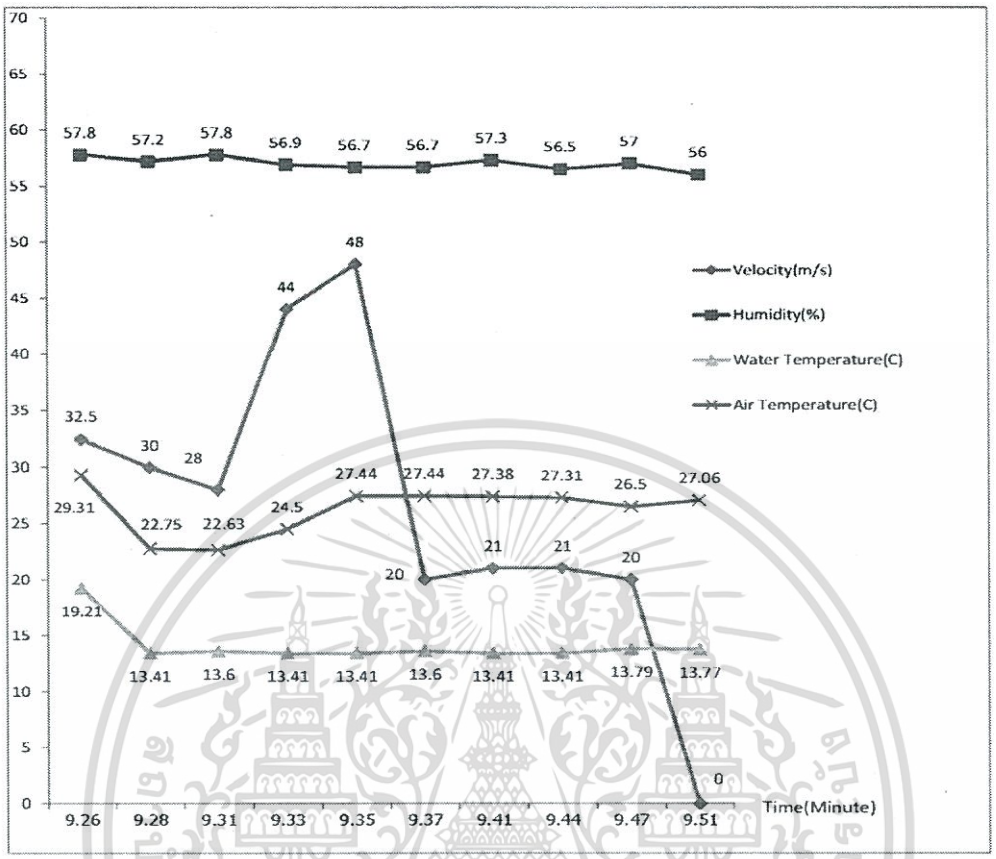


รูปที่ 4.14 แสดงตารางข้อมูลบนเว็บเพจที่เรียกค่ามาจากฐานข้อมูล

นอกจากนี้ข้อมูลที่เรียกค่ามาจากฐานข้อมูลยังสามารถนำมาแสดงผลในรูปแบบกราฟข้อมูล ดังรูปที่ 4.16 โดยใช้คำสั่งสำหรับการเรียกค่ามาจากฐานข้อมูลดังที่แสดงในรูปที่ 4.15

```php
<?php

include ("../jpgraph.php");
include ("../jpgraph_bar.php");
include ("../jpgraph_line.php");

$datay=array(2,3,5,8.5,11.5,6,3);

// Create the graph.
$graph = new Graph(350,300);

$graph->SetScale("textlin");

$graph->SetMarginColor('navy:1.9');
$graph->SetBox();

$graph->title->Set('Bar Pattern');
$graph->title->SetFont(FF_ARIAL,FS_BOLD,20);

$graph->SetTitleBackground('lightblue:1.3',TITLEBKG_STYLE2,TITLEBKG_FRAME_BEVEL);
$graph->SetTitleBackgroundFillStyle(TITLEBKG_FILLSTYLE_HSTRIPED,'lightblue','blue');

// Create a bar pot
$bplot = new BarPlot($datay);
$bplot->SetFillColor('darkorange');
$bplot->SetWidth(0.6);

$bplot->SetPattern(PATTERN_CROSS1,'navy');

$graph->Add($bplot);

$graph->Stroke();
?>
```

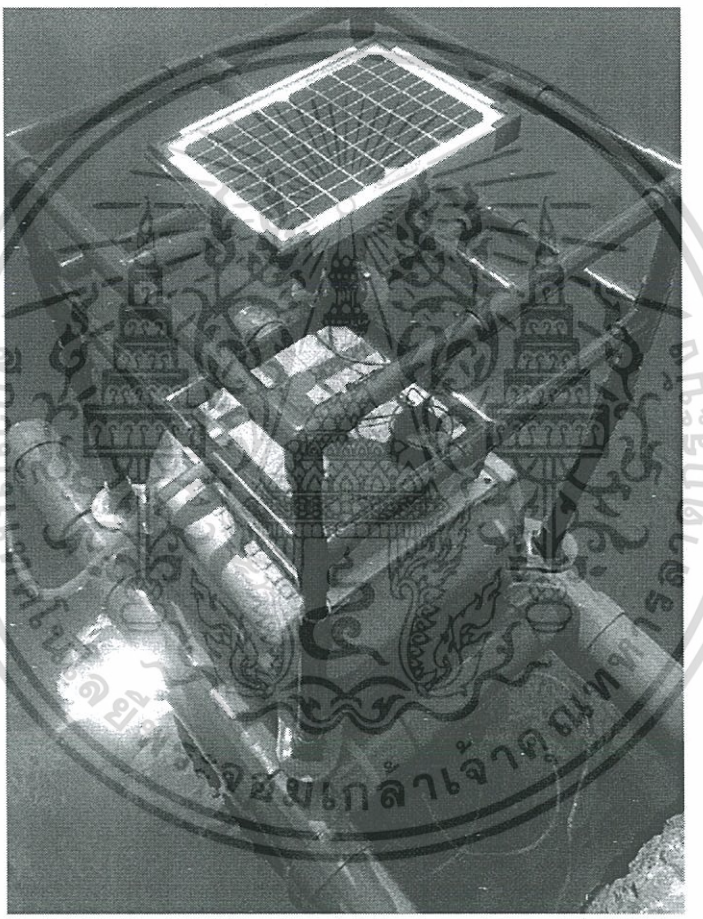
รูปที่ 4.15 คำสั่งที่ใช้สำหรับการเรียกค่ามาจากฐานข้อมูลเพื่อแสดงผลเป็นกราฟข้อมูล



รูปที่ 4.16 กราฟแสดงข้อมูลที่ได้รับจากเซ็นเซอร์

4.5 ผลการทดลองการทำงานในสภาพแวดล้อมจริง

ในส่วนของตัวโครงสร้างที่ออกแบบ จะใช้ท่อพีวีซี เป็นวัสดุหลักในการทำโครงเพื่อติดตัวใบพัดลูกถ้วย และแผงโซล่าเซลล์ไว้ด้านบน ส่วนกล่องวงจร จะติดตั้งไว้กับตัวทุ่นด้านล่าง กล่องวงจรทำจากแผ่นอะคริลิก และจะถูกออกแบบเพื่อให้เซนเซอร์บางชนิดที่ต้องรับสัญญาณหรือต้องสัมผัสกับสภาพแวดล้อมเพื่อเก็บค่า ต้องระวังไม่ให้น้ำเข้าภายในกล่องวงจร อาจใช้เป็นซิลิโคนเคลือบไว้บริเวณที่เป็นรูที่ใช้ให้เซ็นเซอร์ที่ฐาน มีโครงล้อมตัวทุ่นไว้เพื่อไม่ให้ทุ่นพลิกคว่ำและป้องกันการกระแทกจากคลื่น แสดงดังรูปที่ 4.15



รูปที่ 4.17 ชิ้นโครงงาน ขณะเก็บผลการทดลอง

บทที่ 5

สรุปผลและข้อเสนอแนะ

5.1 สรุปผล

Arduino สามารถประมวลผลได้อย่างมีประสิทธิภาพ Pressure sensor, Humidity sensor, Temperature sensor สามารถรับค่าข้อมูลต่างๆได้อย่างถูกต้อง อีกทั้ง GPS จะเก็บค่าตำแหน่ง ณ ขณะนั้นอย่างถูกต้องแต่ในบางครั้ง GPS อาจหาตำแหน่งไม่ได้หรือมีค่าผิดเพี้ยนไป เนื่องจากมีสิ่งรบกวนการทำงานของสายอากาศจึงต้องทำการหาค่าตำแหน่งใหม่ ด้าน Optical sensor ซึ่งใช้ในการวัดจำนวนรอบที่ตัดผ่านสามารถวัดความเร็วลมได้ แต่จะมีความคลาดเคลื่อนเล็กน้อยเนื่องจากแสง ความชื้น และฝุ่น ซึ่งเป็นตัวสิ่งทำให้เกิดการทำงานไม่แม่นยำได้ ในส่วนของ Accelerometer สามารถบอกลักษณะของคลื่นได้ตามความเป็นจริงข้อมูลต่างๆ สามารถส่งผ่าน Sim900 เพื่อนำไปแสดงผลบน Server ได้อย่างถูกต้อง

5.2 ข้อเสนอแนะ

- ควรใช้แบตเตอรี่สำรองที่มีขนาด 30,000 mA ขึ้นไป เพราะจากการทดสอบเก็บพลังงานจากพลังงานแสงอาทิตย์โดยโซล่าร์เซลล์แล้ว ระบบมีประสิทธิภาพของการเก็บพลังงาน โดยคิดเป็น 75% ของแบตเตอรี่สำรองขนาด 30,000 mA พลังงานเพียงพอต่อการทำงาน แต่เนื่องจากบางครั้งอาจเกิดกรณีที่ความเข้มแสงไม่เพียงพอ ทำให้ประสิทธิภาพในการเก็บพลังงานลดน้อยลง อาจสาเหตุเนื่องจาก สภาพแวดล้อม ท้องฟ้า อากาศ พายุ ฝน เป็นต้น

- ควรคำนึงเรื่องการใช้พลังงานของอุปกรณ์แต่ละชนิด ควรเลือกใช้อุปกรณ์ที่ใช้พลังงานต่ำ จะช่วยในการประหยัดพลังงานของระบบเป็นอย่างมาก

- ในภาคการรับส่งข้อมูลแบบไร้สาย ควรเลือกใช้อุปกรณ์ที่รองรับสัญญาณคลื่นได้หลายรูปแบบไม่ว่าจะเป็น GSM, 3G, 4G เพื่อประสิทธิภาพที่ดีในการส่งข้อมูลในอนาคต

- เนื่องจากเซ็นเซอร์และอุปกรณ์ที่ใช้ในโครงงานครั้งนี้บางชนิดโดนน้ำไม่ได้ทำให้ต้องระมัดระวังในการใช้งานเป็นอย่างมาก ควรเลือกใช้เซ็นเซอร์หรืออุปกรณ์ที่สามารถกันน้ำได้แต่อุปกรณ์เหล่านั้นย่อมมีราคาที่สูงมาก

- สามารถนำทุ่นสำรวจข้อมูลสภาพแวดล้อมไปประยุกต์ใช้ในการเก็บข้อมูลสภาพแวดล้อมจากแหล่งน้ำอื่นๆได้นอกเหนือจากทะเล สามารถทำการเพิ่มเซ็นเซอร์ต่างๆ เพื่อใช้วัดค่าอื่นๆนอกเหนือจากนี้ได้อีกมากมาย เช่น เซ็นเซอร์วัดค่าความโปร่งแสงของน้ำ วัดค่าความเค็ม วัดปริมาณออกซิเจนในน้ำ เพื่อเป็นแนวทางในการพัฒนาโครงงานให้ดียิ่งขึ้น

บรรณานุกรม

[1] Chokelive.com."Microcontroller Board ไมโครคอนโทรลเลอร์สร้างโลก."
http://www.chokelive.com/blog/2013/07/micro-controller-application.html

[2] Siriwimon Sunthon. "Arduino Mega 2560."
http://mbeddedweekly.blogspot.com/2014/08/arduino-mega2560.html.

[3] Arduitronics.com."บทความการใช้งานโปรแกรม Arduino IDE."
http://www.arduitronics.com/article/เริ่มต้นใช้งานarduinoIDE.html

[4] เนติพงษ์ ฟองสุวรรณ."การเลือกทำงานตามเงื่อนไข (คำสั่ง IF ELSE SWITCH)."
http://www.natipongkak.blogspot.com/2012/08/if-else-switch.html

[5] พฤธิพงศ์ ชลภัทรสุข."คอมพิวเตอร์คืออะไร."
https://sites.google.com/site/itcomputerthailand.html

[6] บ้านอิเล็กทรอนิกส์."โปรโตบอร์ด ."
http://www.semi-shop.com/knowledge/knowledge.php

[7] Infocellar.com. "Universal Serial Bus."
http://www.infocellar.com/hardware/usb.html

[8] Bizitestequipment.com. "Agilent / HP E3630A - Power Supply."
http://www.bizitestequipment.com/detail.asp?id=E3630A

[9] Khonkaenlink. "แบตเตอรี่สำรอง Power Bank."
http://forum.khonkaenlink.info/index.php?topic=17212607.0

[10] Leotech.co.th."เซลล์แสงอาทิตย์ (Solar Cell)."
http://leotech.co.th/blog/ Solar Cell-solar-cell.html

[11] Sparkfun.com." Temperature Sensor - Waterproof (DS18B20)."
http://www.sparkfun.com/products/11050

[12] Warf.com." MPL115A1 Barometric Pressure Sensor."
http://www.warf.com/view.MPL115A1_BarometricPressureSensor.html

[13] Hobbyist.co.nz." DHT22."
http://www.hobbyist.co.nz/?q=dht-22

[14] Aliexpress.com." Accelerometer-Module."
http://th.aliexpress.com/store/product/GY-61-ADXL335-3-axis-Analog-Output-Accelerometer-Module.html

[15] Wikipedia.org."Photoelectric sensor."
https://th.wikipedia.org/wiki/photoelectric sensor

[16] Globalsources.com."GPS GR-83 Module."
http://www.globalsources.com/si/AS/Holux-Technology/6008801120919/pdtl/GPS-Module/1031230140.htm
[17] Hobbyist.co.nz."SIM900 GSM / GPRS Module."
http://www.hobbyist.co.nz/?q=arduino-gsm-module
[18] Sparkfun.com."LM7805."
https://www.sparkfun.com/datasheets/Components/LM7805.pdf
[19] Xtranetworks.com."บทที่ 1 เริ่มต้นใช้งาน Proteus."
http://www.xtranetworks.com/2012/08/1-proteus.html
[20] Thaidomainregister.com."วิธีติดตั้ง NO-IP และ จดโดเมนเนม."
http://www.thaidomainregister.com/no-ip-signup.php
[21] Appservnetwork.com."ขั้นตอนการติดตั้ง AppServ"
https://www.appservnetwork.com/th
[22] Alldatasheet.com. "MPL115A1"
https://www.alldatasheet.com/datasheet-pdf/310191/MPL115A1.html
[23] Datasheets.maximintegrated.com."DS18b20"
https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf
[24] Propox.com."SIM900 GSM/GPRS Module"
https://www.propox.com/download/docs/SIM900.pdf
[25] Arduino.cc."Arduino Mega."
http://arduino.cc/en/uploads/Main/Arduino_Maga_2560-schematic.pdf

ภาคผนวก ก

“คำสั่ง ที่ใช้เขียนไมโครคอนโทรลเลอร์”

คำสั่ง ที่ใช้เขียนไมโครคอนโทรลเลอร์

/////send by sim900/////

```
#include "SIM900.h"
#include <SoftwareSerial.h>
#include "inetGSM.h"
InetGSM inet;
char msg[50];
int numdata;
char inSerial[50];
int i=0;
boolean started=false;

char s[50];
String S="";

/////result wind speed//////////////////////
unsigned long time;
uint32_t Ts=60000;
int ruptpin = 2;
int val =0;
float velo;
long num =0;

//////////////////Humid sensor//////////////
#include <dht.h>

dht DHT;

#define DHT22_PIN 5

struct
{
    uint32_t total;
    uint32_t ok;
    uint32_t crc_error;
    uint32_t time_out;
    uint32_t connect;
    uint32_t ack_l;
    uint32_t ack_h;
    uint32_t unknown;
} stat = { 0,0,0,0,0,0,0,0};
float H;
```

```
//////////////Pressure sensor//////////////
#include <SPI.h>
#include <OneWire.h>

int DS18B20_Pin = 2;
OneWire ds(DS18B20_Pin);

#define NWS_BARO 29.92

// Pin definitions
#define MPL115A1_ENABLE_PIN 49
#define MPL115A1_SELECT_PIN 53

// Masks for MPL115A1 SPI i/o
#define MPL115A1_READ_MASK  0x80
#define MPL115A1_WRITE_MASK 0x7F

// MPL115A1 register address map
#define PRESH   0x00    // 80
#define PRESL   0x02    // 82
#define TEMPH   0x04    // 84
#define TEMPL   0x06    // 86

#define A0MSB   0x08    // 88
#define A0LSB   0x0A    // 8A
#define B1MSB   0x0C    // 8C
#define B1LSB   0x0E    // 8E
#define B2MSB   0x10    // 90
#define B2LSB   0x12    // 92
#define C12MSB  0x14    // 94
#define C12LSB  0x16    // 96
#define C11MSB  0x18    // 98
#define C11LSB  0x1A    // 9A
#define C22MSB  0x1C    // 9C
#define C22LSB  0x1E    // 9E

// Unit conversion macros
#define FT_TO_M(x) ((long)((x)*(0.3048)))
#define KPA_TO_INHG(x) ((x)*(0.295333727))
#define KPA_TO_MMHG(x) ((x)*(7.50061683))
#define KPA_TO_PSIA(x) ((x)*(0.145037738))
#define KPA_TO_KGCM2(x) ((x)*(0.0102))
#define INHG_TO_PSIA(x) ((x)*(0.49109778))
#define DEGC_TO_DEGF(x) ((x)*(9.0/5.0)+32)
float P;
float K;
float A;
float W;
```

```
String W1;


///////set Acelerator sensor//////////
#include <Array.h>

const byte size = 120';
int rawArray[size];
Array<int> array = Array<int>(rawArray,size);
int z=0;
int z1;
int z2;
int z3;
String Y;

//////set GPS//////////////////////////
#include <NewSoftSerial.h>
#include <TinyGPS.h>
TinyGPS gps;
NewSoftSerial nss(2, 3);

void gpsdump(TinyGPS &gps);
bool feedgps();
void printFloat(double f, int digits = 2);


void setup(){

  Serial.begin(9600);

  ////////send by sim
900////////////////////////////////////
  Serial.println("GSM Shield testing.");
     //Start configuration of shield with baudrate.
     //For http uses is raccomanded to use 4800 or
slower.
     if (gsm.begin(9600)) {
          Serial.println("\nstatus=READY");
          started=true;
     } else Serial.println("\nstatus=IDLE");
     if(started) {
          //GPRS attach, put in order APN, username
and password.
          //If no needed auth let them blank.
          if (inet.attachGPRS("internet", "true",
"true"))
               Serial.println("status=ATTACHED");
          else Serial.println("status=ERROR");
```

```
          delay(1000);
          //Read IP address.
          gsm.SimpleWriteln("AT+CIFSR");
          delay(5000);
          //Read until serial buffer is empty.
          gsm.WhileSimpleRead();
          //TCP Client GET, send a GET request to the
server and
          //save the reply.

           //Print the results.
          Serial.println("\nNumber of data
received:");
          Serial.println(numdata);
          Serial.println("\nData received:");
          Serial.println(msg);

     /////////////// set URL
sending/////////////////////////

S="/getcomplete.php?velo="+String(velo)+"&H="+String(H
)+"&P="+String(P)+"&K="+String(K)+"&A="+String(A)+"&W=
"+String(W)+"&X="+X+"&La"+L+"&Lo"+l+"&V"+V;
     S.toCharArray(s, 200);
     numdata=inet.httpGET("paazazas.ddns.net", 80, s,
msg, 50);
     numdata=inet.httpGET("161.246.18.217", 80, s,
msg, 50);
     }

   ////////////////////set pressure
sensor////////////////////
    SPI.begin();

    pinMode(MPL115A1_SELECT_PIN, OUTPUT);
    pinMode(MPL115A1_ENABLE_PIN, OUTPUT);

    digitalWrite(MPL115A1_ENABLE_PIN, LOW);
    digitalWrite(MPL115A1_SELECT_PIN, HIGH);


 ////////////set GPS//////////////////

  nss.begin(4800);

}
void loop(){
  time = millis();
```

```
  val = analogRead(ruptpin);
  if(val>1020)
  {
    num=num+1;
    delay(10);
  }
  if(time>Ts)
{
  Ts=time+60000;
  velo=2*3.14*(num/60)*0.075;
    Serial.print("num=");
    Serial.println(num);
    Serial.print("velo=");
    Serial.print(velo);
    Serial.println(" m/s");

    //////////humid sensor//////////
    uint32_t start = micros();
    int chk = DHT.read22(DHT22_PIN);
    uint32_t stop = micros();

    stat.total++;
    switch (chk)
    {
    case DHTLIB_OK:
        stat.ok++;
        break;
    case DHTLIB_ERROR_CHECKSUM:
        stat.crc_error++;
        Serial.print("Checksum error,\t");
        break;
    case DHTLIB_ERROR_TIMEOUT:
        stat.time_out++;
        Serial.print("Time out error,\t");
        break;
    case DHTLIB_ERROR_CONNECT:
        stat.connect++;
        Serial.print("Connect error,\t");
        break;
    case DHTLIB_ERROR_ACK_L:
        stat.ack_l++;
        Serial.print("Ack Low error,\t");
        break;
    case DHTLIB_ERROR_ACK_H:
        stat.ack_h++;
        Serial.print("Ack High error,\t");
        break;
    default:
```

```
    stat.unknown++;
    Serial.print("Unknown error,\t");
    break;
}
// DISPLAY DATA
H=DHT.humidity;
Serial.println(H);

//////////////pressure sensor//////////////////
float pressure_pKa = 0;
float temperature_c= 0;
float W = getTemp();

 //แจ้งเตือนอุณหภูมิเปลี่ยนแปลง
if (W>30){
X="H%over%water";
Serial.println("H over water");
}
if(W<-1){
X="L%over%water";
Serial.println("L over water");
}



// wake the MPL115A1
digitalWrite(MPL115A1_ENABLE_PIN, HIGH);
delay(20);  // give the chip a few ms to wake up

pressure_pKa = calculatePressurekPa();
temperature_c = calculateTemperatureC();

digitalWrite(MPL115A1_ENABLE_PIN, LOW);


Serial.print(P);
Serial.print(" in Hg | ");
Serial.print(K);
Serial.print(" kPa | ");

////////water temp codition////////
Serial.print(A);
Serial.println("C | ");
Serial.print("water temperature = ");
Serial.print(W);
Serial.println("C");
```

```
    P=KPA_TO_INHG(pressure_pKa);
    K=pressure_pKa;
    A=temperature_c;

  ////////Acelerator sensor loop 1
minute//////////////
    for (byte i=0; i<array.size(); i++){
    z=analogRead(A0);
    array[i]=map(z, 0, 1023, 0, 255);
    Serial.println(array[i]);
    delay(500);
     }

  Serial.print("\tMinimum value:");
  z1=array.getMin();
  Serial.println(z1);

  Serial.print("\tMaximum value:");
  z2=array.getMax();
  Serial.println(z2);

  Serial.print("\tAverage value:");
  z3=array.getAverage();
  Serial.println(z3);

  delay(1);

    for (byte i=0; i<array.size(); i++){
    z=analogRead(A0);
    array[i]=map(z, 0, 1023, 0, 255);
    Serial.println(array[i]);
    delay(500);
     }

  Serial.print("\tMinimum value:");
  z11=array.getMin();
  Serial.println(z1);

  Serial.print("\tMaximum value:");
  z22=array.getMax();
  Serial.println(z2);

  Serial.print("\tAverage value:");
  z4=array.getAverage();
  Serial.println(z4);


  delay(1);
```

```
  D = z4-z3;
  Serial.println(D);

  if(D>=z2){
  V="Rough/High";
  Serial.println("Rough/High");
  }
  if(D<(0.5*z2))
  {
   V="Calm";
    Serial.println("Calm");
  }
  if (z3 < D <(0.75*z2))
  {
    V="Moderate";
   Serial.println("Moderate");
   }

//////////////////////GPS//////////////////////////////
//////////////////////////////////////////////////////
////////
 long lat, lon;
  float flat, flon;
  unsigned long age, date, time, chars;
  int year;
  byte month, day, hour, minute, second, hundredths;
  unsigned short sentences, failed;

  gps.get_position(&lat, &lon, &age);
  Serial.print("Lat/Long(10^-5 deg): ");
Serial.print(lat); Serial.print(", ");
Serial.print(lon);
  Serial.print(" Fix age: "); Serial.print(age);
Serial.println("ms.");

  feedgps(); // If we don't feed the gps during this
long routine, we may drop characters and get checksum
errors

  gps.f_get_position(&flat, &flon, &age);
  Serial.print("Lat/Long(float): "); printFloat(flat,
5); Serial.print(", "); printFloat(flon, 5);
  Serial.print(" Fix age: "); Serial.print(age);
Serial.println("ms.");

  feedgps();

  gps.get_datetime(&date, &time, &age);
```

```
  Serial.print("Date(ddmmyy): "); Serial.print(date);
Serial.print(" Time(hhmmsscc): "); Serial.print(time);
  Serial.print(" Fix age: "); Serial.print(age);
Serial.println("ms.");

  feedgps();

  gps.crack_datetime(&year, &month, &day, &hour,
&minute, &second, &hundredths, &age);
  Serial.print("Date: ");
Serial.print(static_cast<int>(month));
Serial.print("/");
Serial.print(static_cast<int>(day));
Serial.print("/"); Serial.print(year);
  Serial.print("  Time: ");
Serial.print(static_cast<int>(hour));
Serial.print(":");
Serial.print(static_cast<int>(minute));
Serial.print(":");
Serial.print(static_cast<int>(second));
Serial.print(".");
Serial.print(static_cast<int>(hundredths));
  Serial.print("  Fix age: "); Serial.print(age);
Serial.println("ms.");

  feedgps();

  Serial.print("Alt(cm): ");
Serial.print(gps.altitude()); Serial.print("
Course(10^-2 deg): "); Serial.print(gps.course());
Serial.print(" Speed(10^-2 knots): ");
Serial.println(gps.speed());
  Serial.print("Alt(float): ");
printFloat(gps.f_altitude()); Serial.print("
Course(float): "); printFloat(gps.f_course());
Serial.println();
  Serial.print("Speed(knots): ");
printFloat(gps.f_speed_knots()); Serial.print(" (mph):
");  printFloat(gps.f_speed_mph());
  Serial.print(" (mps): ");
printFloat(gps.f_speed_mps()); Serial.print(" (kmph):
"); printFloat(gps.f_speed_kmph()); Serial.println();

  feedgps();

  gps.stats(&chars, &sentences, &failed);
  Serial.print("Stats: characters: ");
Serial.print(chars); Serial.print(" sentences: ");
```

```
Serial.print(sentences); Serial.print(" failed
checksum: "); Serial.println(failed);



S="/getcomplete.php?velo="+String(velo)+"&H="+String(H
)+"&P="+String(P)+"&K="+String(K)+"&A="+String(A)+"&W=
"+String(W)+"&X="+X+"&La"+L+"&Lo"+l+"&V"+V;
     S.toCharArray(s, 200);
     numdata=inet.httpGET("paazazas.ddns.net", 80, s,
msg, 50);
     numdata=inet.httpGET("161.246.18.217", 80, s,
msg, 50);
          //Print the results.
     Serial.println("\nNumber of data received:");
     Serial.println(numdata);
     Serial.println("\nData received:");
     Serial.println(msg);
      delay(2000);

    num=0;
    delay(10);


   //check again///
          //GPRS attach, put in order APN, username
and password.
          //If no needed auth let them blank.
          if (inet.attachGPRS("internet", "true",
"true"))
               Serial.println("status=ATTACHED");
          else Serial.println("status=ERROR");
          delay(1000);
          //Read IP address.
          gsm.SimpleWriteln("AT+CIFSR");
          delay(5000);
          //Read until serial buffer is empty.
          gsm.WhileSimpleRead();
          //TCP Client GET, send a GET request to the
server and
          //save the reply.


}

}
```

```
void serialhwread()
{
     i=0;
     if (Serial.available() > 0) {
          while (Serial.available() > 0) {
               inSerial[i]=(Serial.read());
               delay(10);
               i++;
          }
          inSerial[i]='\0';
          if(!strcmp(inSerial,"/END")) {
               Serial.println("_");
               inSerial[0]=0x1a;
               inSerial[1]='\0';
               gsm.SimpleWriteln(inSerial);
          }
          //Send a saved AT command using serial port.
          if(!strcmp(inSerial,"TEST")) {
               Serial.println("SIGNAL QUALITY");
               gsm.SimpleWriteln("AT+CSQ");
          }
          //Read last message saved.
          if(!strcmp(inSerial,"MSG")) {
               Serial.println(msg);
          } else {
               Serial.println(inSerial);
               gsm.SimpleWriteln(inSerial);
          }
          inSerial[0]='\0';
     }
}
void serialswread()
{
     gsm.SimpleRead();
}


float getTemp(){
 //returns the temperature from one DS18S20 in DEG
Celsius

 byte data[12];
 byte addr[8];

 if ( !ds.search(addr)) {
   //no more sensors on chain, reset search
```

```
    ds.reset_search();
    return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
    Serial.println("CRC is not valid!");
    return -1000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28) {
    Serial.print("Device is not recognized");
    return -1000;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1); // start conversion, with parasite
power on at the end

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE); // Read Scratchpad


  for (int i = 0; i < 9; i++) { // we need 9 bytes
   data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];

  float tempRead = ((MSB << 8) | LSB); //using two's
compliment
  float TemperatureSum = tempRead / 16;

  return TemperatureSum;
}

float calculateTemperatureC() {

    unsigned int uiTadc;
    unsigned char uiTH, uiTL;

    unsigned int temperature_counts = 0;
```

```
    writeRegister(0x22, 0x00);  // Start temperature
conversion
    delay(2);                   // Max wait time is
0.7ms, typ 0.6ms

    // Read pressure
    uiTH = readRegister(TEMPH);
    uiTL = readRegister(TEMPL);

    uiTadc = (unsigned int) uiTH << 8;
    uiTadc += (unsigned int) uiTL & 0x00FF;

    // Temperature is a 10bit value
    uiTadc = uiTadc >> 6;

    // -5.35 counts per ยฐC, 472 counts is 25ยฐC
    return 25 + (uiTadc - 472) / -5.35;
}


float calculatePressurekPa() {

    // See Freescale document AN3785 for detailed
explanation
    // of this implementation.

    signed char sia0MSB, sia0LSB;
    signed char sib1MSB, sib1LSB;
    signed char sib2MSB, sib2LSB;
    signed char sic12MSB, sic12LSB;
    signed char sic11MSB, sic11LSB;
    signed char sic22MSB, sic22LSB;
    signed int sia0, sib1, sib2, sic12, sic11, sic22,
siPcomp;
    float decPcomp;
    signed long lt1, lt2, lt3, si_c11x1, si_a11,
si_c12x2;
    signed long si_a1, si_c22x2, si_a2, si_a1x1,
si_y1, si_a2x2;
    unsigned int uiPadc, uiTadc;
    unsigned char uiPH, uiPL, uiTH, uiTL;

    writeRegister(0x24, 0x00);      // Start Both
Conversions
    //writeRegister(0x20, 0x00);    // Start Pressure
Conversion
    //writeRegister(0x22, 0x00);    // Start
temperature conversion
```

```
    delay(2);                          // Max wait time
is 1ms, typ 0.8ms

    // Read pressure
    uiPH = readRegister(PRESH);
    uiPL = readRegister(PRESL);
    uiTH = readRegister(TEMPH);
    uiTL = readRegister(TEMPL);

    uiPadc = (unsigned int) uiPH << 8;
    uiPadc += (unsigned int) uiPL & 0x00FF;
    uiTadc = (unsigned int) uiTH << 8;
    uiTadc += (unsigned int) uiTL & 0x00FF;

    // Placing Coefficients into 16-bit Variables
    // a0
    sia0MSB = readRegister(A0MSB);
    sia0LSB = readRegister(A0LSB);
    sia0 = (signed int) sia0MSB << 8;
    sia0 += (signed int) sia0LSB & 0x00FF;

    // b1
    sib1MSB = readRegister(B1MSB);
    sib1LSB = readRegister(B1LSB);
    sib1 = (signed int) sib1MSB << 8;
    sib1 += (signed int) sib1LSB & 0x00FF;

    // b2
    sib2MSB = readRegister(B2MSB);
    sib2LSB = readRegister(B2LSB);
    sib2 = (signed int) sib2MSB << 8;
    sib2 += (signed int) sib2LSB & 0x00FF;

    // c12
    sic12MSB = readRegister(C12MSB);
    sic12LSB = readRegister(C12LSB);
    sic12 = (signed int) sic12MSB << 8;
    sic12 += (signed int) sic12LSB & 0x00FF;

    // c11
    sic11MSB = readRegister(C11MSB);
    sic11LSB = readRegister(C11LSB);
    sic11 = (signed int) sic11MSB << 8;
    sic11 += (signed int) sic11LSB & 0x00FF;

    // c22
    sic22MSB = readRegister(C22MSB);
    sic22LSB = readRegister(C22LSB);
```

```
sic22 = (signed int) sic22MSB << 8;
sic22 += (signed int) sic22LSB & 0x00FF;

// Coefficient 9 equation compensation
uiPadc = uiPadc >> 6;
uiTadc = uiTadc >> 6;

// Step 1 c11x1 = c11 * Padc
lt1 = (signed long) sic11;
lt2 = (signed long) uiPadc;
lt3 = lt1*lt2;
si_c11x1 = (signed long) lt3;

// Step 2 a11 = b1 + c11x1
lt1 = ((signed long)sib1)<<14;
lt2 = (signed long) si_c11x1;
lt3 = lt1 + lt2;
si_a11 = (signed long)(lt3>>14);

// Step 3 c12x2 = c12 * Tadc
lt1 = (signed long) sic12;
lt2 = (signed long) uiTadc;
lt3 = lt1*lt2;
si_c12x2 = (signed long)lt3;

// Step 4 a1 = a11 + c12x2
lt1 = ((signed long)si_a11<<11);
lt2 = (signed long)si_c12x2;
lt3 = lt1 + lt2;
si_a1 = (signed long) lt3>>11;

// Step 5 c22x2 = c22*Tadc
lt1 = (signed long)sic22;
lt2 = (signed long)uiTadc;
lt3 = lt1 * lt2;
si_c22x2 = (signed long)(lt3);

// Step 6 a2 = b2 + c22x2
lt1 = ((signed long)sib2<<15);
lt2 = ((signed long)si_c22x2>1);
lt3 = lt1+lt2;
si_a2 = ((signed long)lt3>>16);

// Step 7 a1x1 = a1 * Padc
lt1 = (signed long)si_a1;
lt2 = (signed long)uiPadc;
lt3 = lt1*lt2;
si_a1x1 = (signed long)(lt3);
```

```
    // Step 8 y1 = a0 + a1x1
    lt1 = ((signed long)sia0<<10);
    lt2 = (signed long)si_a1x1;
    lt3 = lt1+lt2;
    si_y1 = ((signed long)lt3>>10);

    // Step 9 a2x2 = a2 * Tadc
    lt1 = (signed long)si_a2;
    lt2 = (signed long)uiTadc;
    lt3 = lt1*lt2;
    si_a2x2 = (signed long)(lt3);

    // Step 10 pComp = y1 + a2x2
    lt1 = ((signed long)si_y1<<10);
    lt2 = (signed long)si_a2x2;
    lt3 = lt1+lt2;

    // Fixed point result with rounding
    //siPcomp = ((signed int)lt3>>13);
    siPcomp = lt3/8192;

    // decPcomp is defined as a floating point number
    // Conversion to decimal value from 1023 ADC count
value
    // ADC counts are 0 to 1023, pressure is 50 to
115kPa respectively
    decPcomp = ((65.0/1023.0)*siPcomp)+50;

    return decPcomp;
}


unsigned int readRegister(byte thisRegister) {

    byte result = 0;

    // select the MPL115A1
    digitalWrite(MPL115A1_SELECT_PIN, LOW);

    // send the request
    SPI.transfer(thisRegister | MPL115A1_READ_MASK);
    result = SPI.transfer(0x00);

    // deselect the MPL115A1
    digitalWrite(MPL115A1_SELECT_PIN, HIGH);
```

```
return result;
}


void writeRegister(byte thisRegister, byte thisValue)
{

    // select the MPL115A1
    digitalWrite(MPL115A1_SELECT_PIN, LOW);

    // send the request
    SPI.transfer(thisRegister & MPL115A1_WRITE_MASK);
    SPI.transfer(thisValue);

    // deselect the MPL115A1
    digitalWrite(MPL115A1_SELECT_PIN, HIGH);
}

bool feedgps()
{
  while (nss.available())
  {
    if (gps.encode(nss.read()))
      return true;
  }
  return false;
}
```

ภาคผนวก ข

“คำสั่งที่ใช้สร้างหน้าเว็บเพจ HTML เพื่อเรียกแสดงข้อมูล”

คำสั่งที่ใช้สร้างหน้าเว็บเพจ HTML

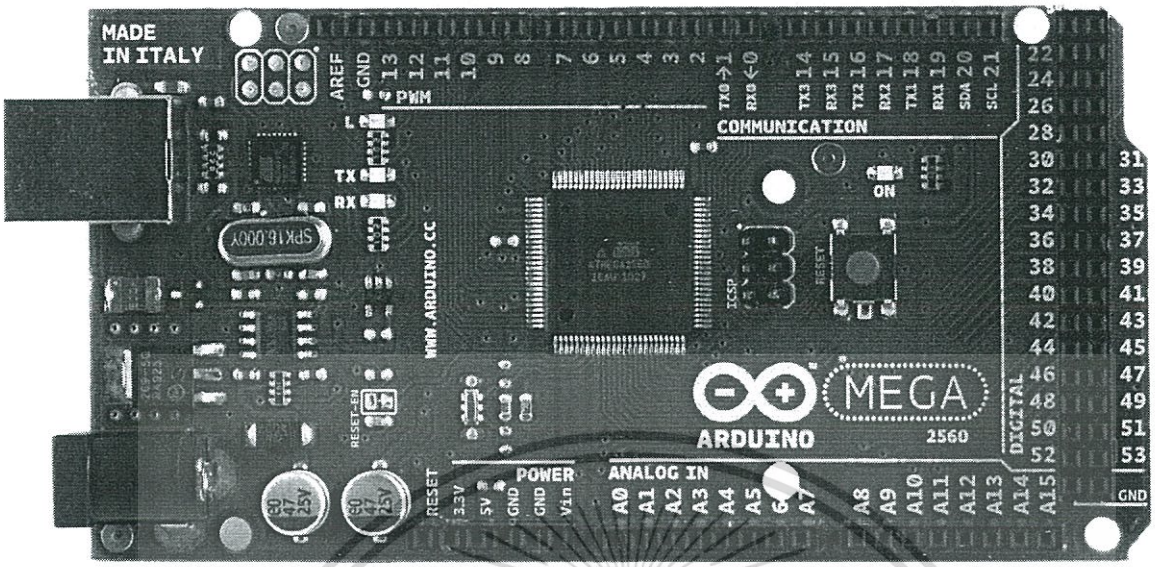
PHP code getconnect.php

```
<?
 echo date('d:m:y H:i:s');
 $time =date('d/m/y H:i:s');

 $survername = "localhost";
 $username ="root";
 $password ="1234";
 $dbname ="marine machine";

 $conn =
mysqli_connect($survername,$username,$password,$dbname
);
 if(!$conn)
 {echo "fault";
 exit;
 }

 date_default_timezone_set("Asia/bangkok");
 $sql = "INSERT INTO
lalo(time,velo,humid,pressureinHg,pressurekpa,Airtempe
ratureC,WatertemperatureC,warning,latitude,longitude,w
avenature)
      VALUE
('$time','{$_GET["velo"]}','{$_GET["H"]}','{$_GET["P"]
}','{$_GET["K"]}','{$_GET["A"]}','{$_GET["W"]}','{$_GE
T["X"]}','{$_GET["La"]}','{$_GET["Lo"]}','{$_GET["V"]}
')";
 $result = mysqli_query($conn,$sql);


mysqli_close($conn);
?>
```

ภาคผนวก ค

“DATASHEET”

Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

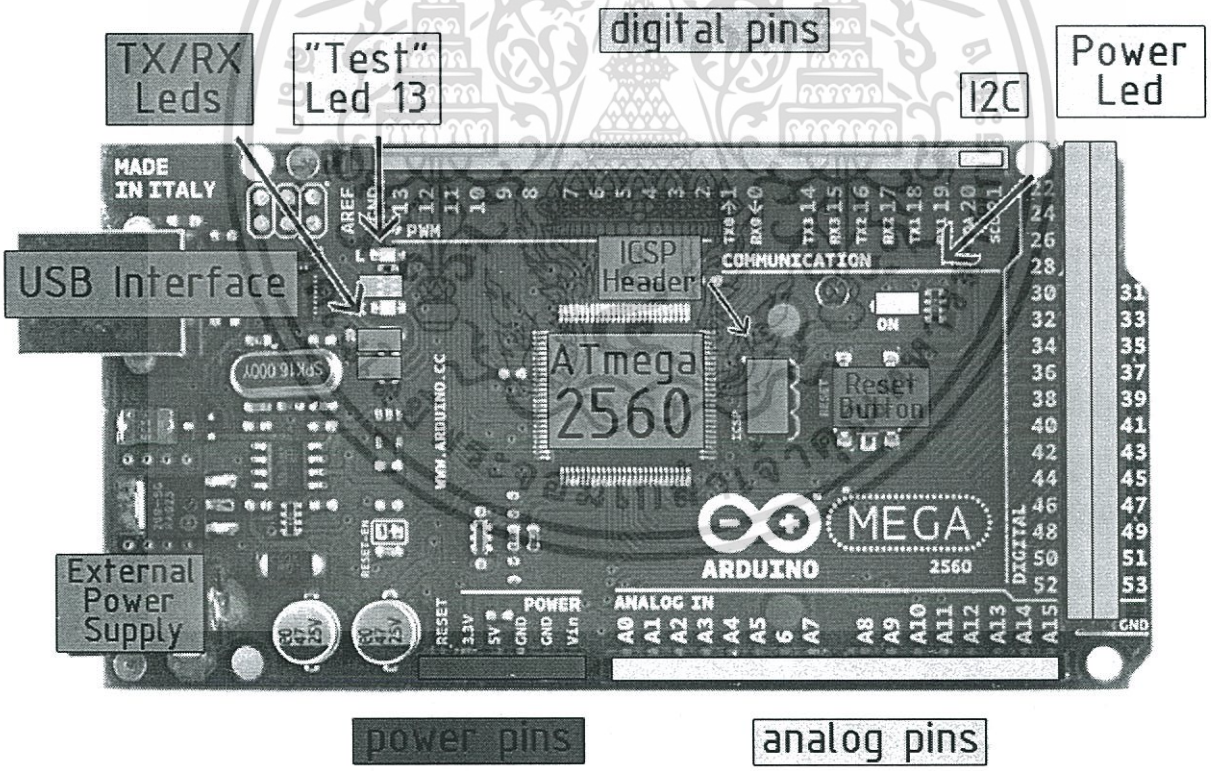
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- **I^2C: 20 (SDA) and 21 (SCL).** Support I^2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I^2C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

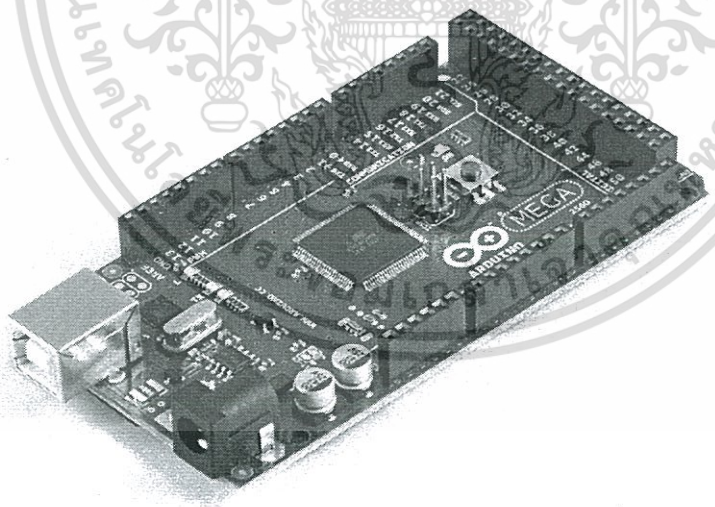
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.



Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega2560 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Mega2560 is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Mega2560. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Mega contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Mega has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics and Shield Compatibility

The maximum length and width of the Mega PCB are 4 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

The Mega is designed to be compatible with most shields designed for the Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega and Duemilanove / Diecimila. **Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).**

How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

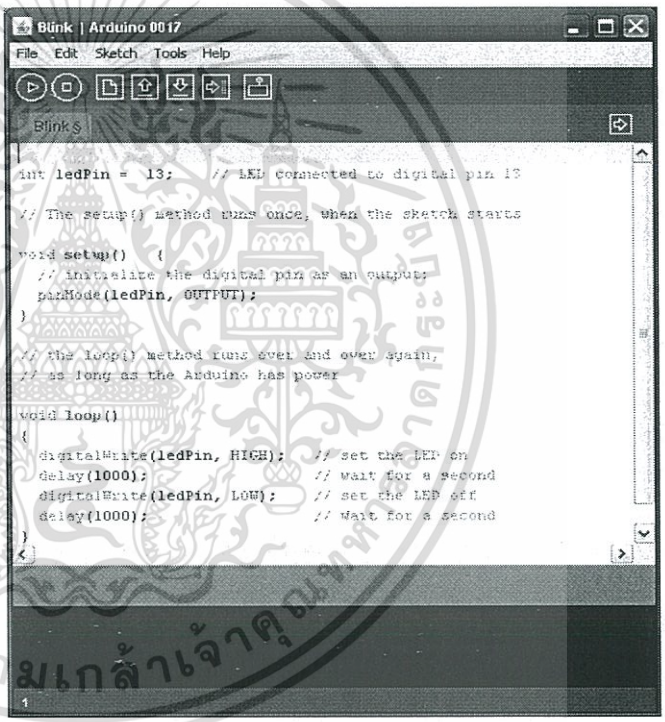
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

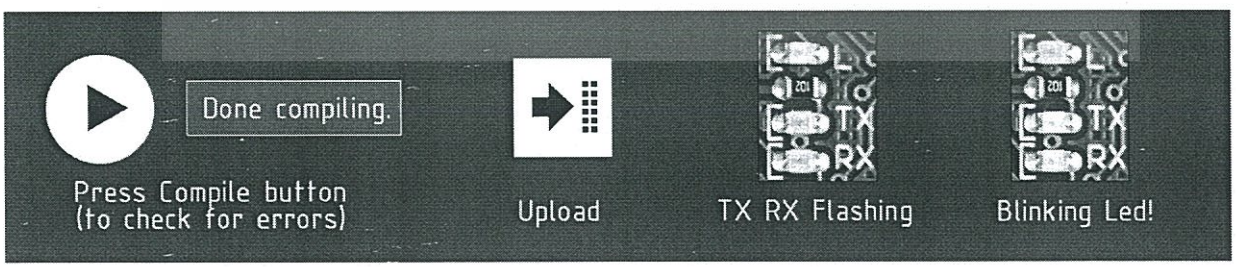
File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

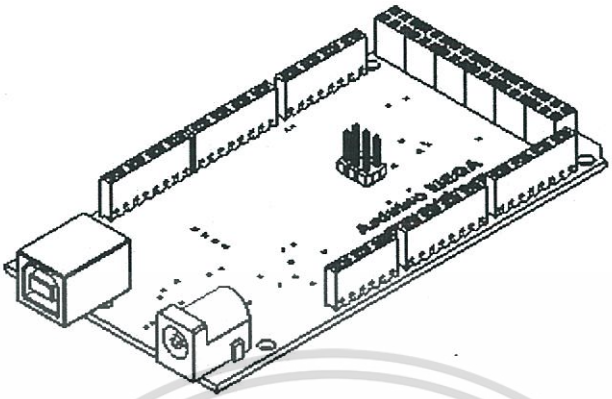
Once you have your skecth you'll see something very close to the screenshot on the right.

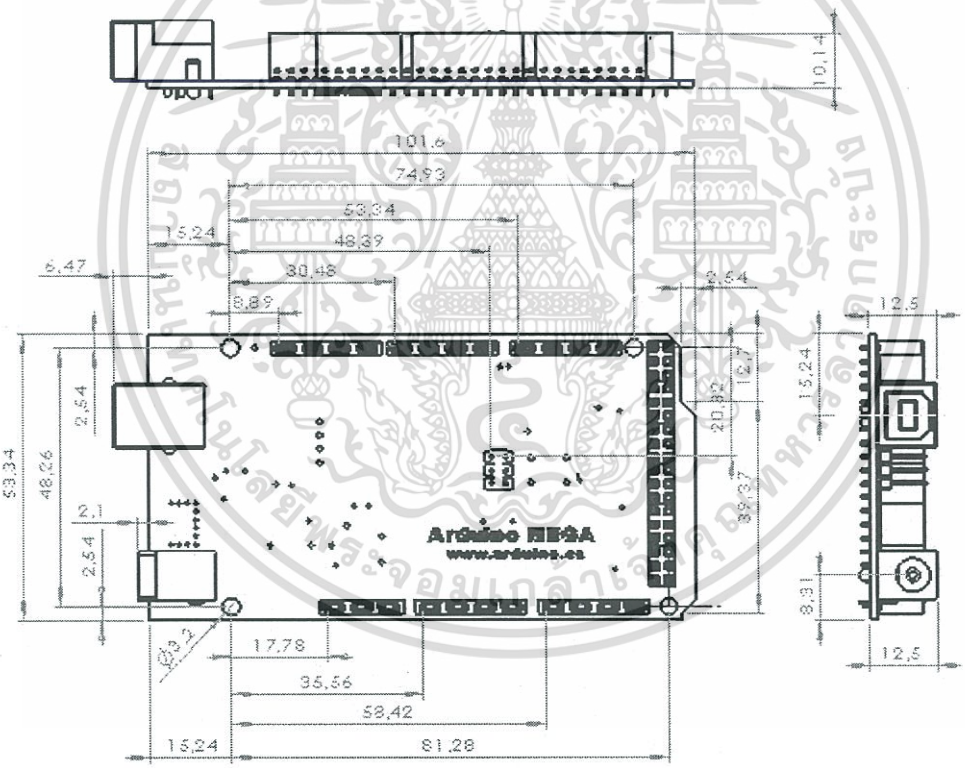
In **Tools>Board** select MEGA

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.









Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

SIM900
GSM/GPRS Module

The SIM900 is a complete Quad-band GSM/GPRS solution in a SMT module which can be embedded in the customer applications.

Featuring an industry-standard interface, the SIM900 delivers GSM/GPRS 850/900/1800/1900MHz performance for voice, SMS, Data, and Fax in a small form factor and with low power consumption. With a tiny configuration of 24mm x 24mm x 3 mm, SIM900 can fit almost all the space requirements in your M2M application, especially for slim and compact demand of design.

- SIM900 is designed with a very powerful single-chip processor integrating AMR926EJ-S core
- Quad - band GSM/GPRS module with a size of 24mmx24mmx3mm
- SMT type suit for customer application
- An embedded Powerful TCP/IP protocol stack
- Based upon mature and field-proven platform, backed up by our support service, from definition to design and production

General featrues

- Quad-Band 850/ 900/ 1800/ 1900 MHz
- GPRS multi-slot class 10/8
- GPRS mobile station class B
- Compliant to GSM phase 2/2+
  – Class 4 (2 W @850/ 900 MHz)
  – Class 1 (1 W @ 1800/1900MHz)
- Dimensions: 24* 24 * 3 mm
- Weight: 3.4g
- Control via AT commands (GSM 07.07 ,07.05 and SIMCOM enhanced AT Commands)
- SIM application toolkit
- Supply voltage range 3.4 ... 4.5 V
- Low power consumption
- Operation temperature: -30 °C to +80 °C

Specifications for fax

- Group 3, class 1

Specifications for data

- GPRS class 10: max. 85.6 kbps (downlink)
- PBCCH support
- Coding schemes CS 1, 2, 3, 4
- CSD up to 14.4 kbps
- USSD
- Non transparent mode
- PPP-stack

Specifications for SMS via GSM / GPRS

- Point-to-point MO and MT
- SMS cell broadcast
- Text and PDU mode

Drivers

- MUX Driver

Specifications for voice

- Tricodec
  – Half rate (HR)
  – Full rate (FR)
  – Enhanced Full rate (EFR)
- Hands-free operation （Echo suppression）
- AMR
  Half Rate(HR)
  Full Rate(FR)

Interfaces

- Interface to external SIM 3V/ 1.8V
- analog audio interface
- RTC backup
- SPI interface
- Serial interface
- Antenna pad
- I2C
- GPIO
- PWM
- ADC

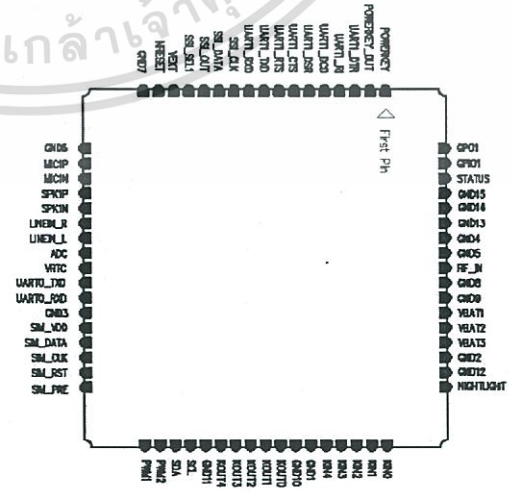
Compatibility

- AT cellular command interface

Approvals（in planning）

- CE
- FCC
- ROHS
- PTCRB
- GCF
- AT&T
- IC
- TA

Pin Assignment



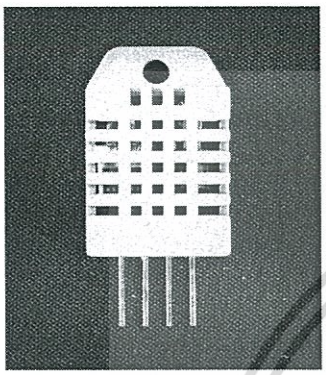
More about SIM900 module, Please contact: Tel:+86 21 32523300
Fax:+86 21 32523301
Email:simcom@sim.com

Aosong(Guangzhou) Electronics Co.,Ltd

Tell: +86-020-36380552, +86-020-36042809 Fax: +86-020-36380562
http://www.aosong.com
Email: thomasliu198518@yahoo.com.cn sales@aosong.com
Address: No.56, Renhe Road, Renhe Town, Baiyun District, Guangzhou, China

Digital-output relative humidity & temperature sensor/module

AM2303



Capacitive-type humidity and temperature module/sensor

1. Feature & Application:

* Full range temperature compensated * Relative humidity and temperature measurement
* Calibrated digital signal *Outstanding long-term stability *Extra components not needed
* Long transmission distance * Low power consumption *4 pins packaged and fully interchangeable

2. Description:

AM2303 output calibrated digital signal. It utilizes exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability.Its sensing elements is connected with 8-bit single-chip computer.

Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.
Small size & low consumption & long transmission distance(20m) enable AM2303 to be suited in all kinds of harsh application occasions.

Single-row packaged with four pins, making the connection very convenient.

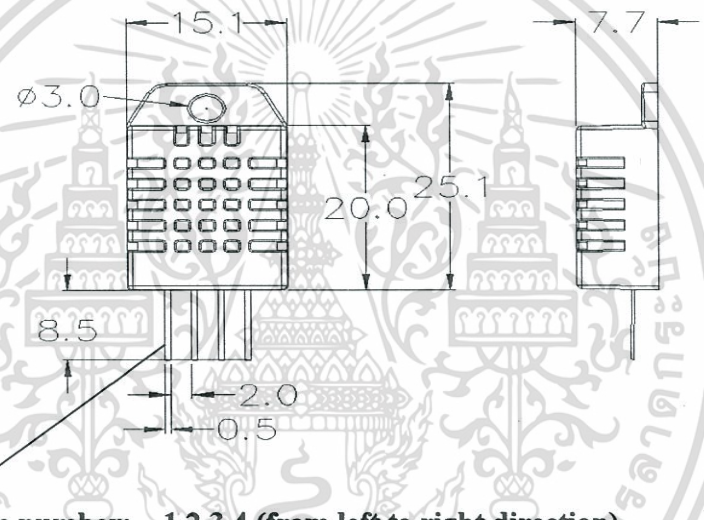
3. Technical Specification:

Model	AM2303
Power supply	3.3-6V DC
Output signal	digital signal via single-bus
Sensing element	Polymer humidity capacitor & DS18B20 for detecting temperature
Measuring range	humidity 0-100%RH; temperature -40~125Celsius

Tell: +86-020-36380552, +86-020-36042809 Fax: +86-020-36380562
http://www.aosong.com
Email: thomasliu198518@yahoo.com.cn sales@aosong.com
Address: No.56, Renhe Road, Renhe Town, Baiyun District, Guangzhou, China

Accuracy	humidity +-2%RH(Max +-5%RH);	temperature +-0.2Celsius
Resolution or sensitivity	humidity 0.1%RH;	temperature 0.1Celsius
Repeatability	humidity +-1%RH;	temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH	
Long-term Stability	+-0.5%RH/year	
Sensing period	Average: 2s	
Interchangeability	fully interchangeable	

4. Dimensions: (unit----mm)



Pin sequence number: 1 2 3 4 (from left to right direction).

Pin	Function
1	VDD---power supply
2	DATA--signal
3	NULL
4	GND

5. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-6V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

Single-bus data is used for communication between MCU and AM2303, it costs 5mS for single time communication.

DALLAS SEMICONDUCTOR

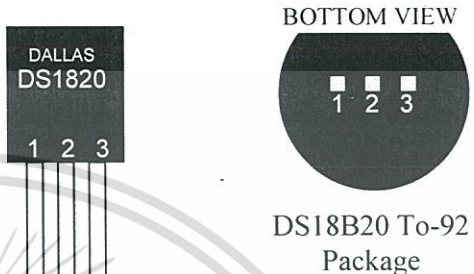
DS18B20
Programmable Resolution 1-Wire® Digital Thermometer

www.dalsemi.com

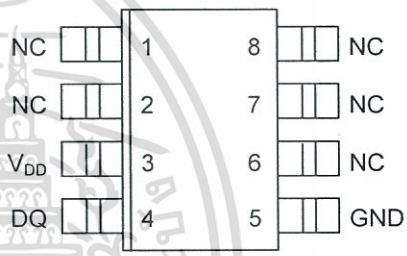
FEATURES

- Unique 1-Wire interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Zero standby power required
- Measures temperatures from -55°C to +125°C. Fahrenheit equivalent is -67°F to +257°F
- ±0.5°C accuracy from -10°C to +85°C
- Thermometer resolution is programmable from 9 to 12 bits
- Converts 12-bit temperature to digital word in 750 ms (max.)
- User-definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

PIN ASSIGNMENT



DS18B20 To-92 Package



DS18B20Z
8-Pin SOIC (150 mil)

PIN DESCRIPTION

GND - Ground
DQ - Data In/Out
V_{DD} - Power Supply Voltage
NC - No Connect

DESCRIPTION

The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device.

Information is sent to/from the DS18B20 over a 1-Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1-Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.

DETAILED PIN DESCRIPTION Table 1

PIN 8PIN SOIC	PIN TO92	SYMBOL	DESCRIPTION
5	1	GND	Ground.
4	2	DQ	**Data Input/Output pin.** For 1-Wire operation: Open drain. (See "Parasite Power" section.)
3	3	V_{DD}	**Optional V_{DD} pin.** See "Parasite Power" section for details of connection. V_{DD} must be grounded for operation in parasite power mode.

DS18B20Z (8-pin SOIC): All pins not specified in this table are not to be connected.

OVERVIEW

The block diagram of Figure 1 shows the major components of the DS18B20. The DS18B20 has four main data components: 1) 64-bit lasered ROM, 2) temperature sensor, 3) nonvolatile temperature alarm triggers TH and TL, and 4) a configuration register. The device derives its power from the 1-Wire communication line by storing energy on an internal capacitor during periods of time when the signal line is high and continues to operate off this power source during the low times of the 1-Wire line until it returns high to replenish the parasite (capacitor) supply. As an alternative, the DS18B20 may also be powered from an external 3 volt - 5.5 volt supply.

Communication to the DS18B20 is via a 1-Wire port. With the 1-Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. These commands operate on the 64-bit lasered ROM portion of each device and can single out a specific device if many are present on the 1-Wire line as well as indicate to the bus master how many and what types of devices are present. After a ROM function sequence has been successfully executed, the memory and control functions are accessible and the master may then provide any one of the six memory and control function commands.

One control function command instructs the DS18B20 to perform a temperature measurement. The result of this measurement will be placed in the DS18B20's scratch-pad memory, and may be read by issuing a memory function command which reads the contents of the scratchpad memory. The temperature alarm triggers TH and TL consist of 1 byte EEPROM each. If the alarm search command is not applied to the DS18B20, these registers may be used as general purpose user memory. The scratchpad also contains a configuration byte to set the desired resolution of the temperature to digital conversion. Writing TH, TL, and the configuration byte is done using a memory function command. Read access to these registers is through the scratchpad. All data is read and written least significant bit first.

HOLUX *Technology,Inc.* 長天科技股份有限公司
1F, No.30, R&D Rd. II, Hsinchu City 300, 新竹市研發二路30號 1F Tel:886 3 668 7000
Science-based Industrial Park, Taiwan. E-Mail:info@holux.com.tw Web: www.holux.com Fax:886 3 668 7111

GR-83 GPS Receiver

■ Features

- Use Holux high performance GPS module.
- Independent operation board. Direct communication interface with computer.
- 20 parallel satellite-tracking channels for fast acquisition and reacquisition.
- High speed signal acquisition.
- Built-in WAAS/EGNOS Demodulator.
- Support U.S. Coast Guard DGPS beacon signal.
- Low power consumption with Advanced Trickle-Power and Push-To-Fix mode.
- Optional Rechargeable battery for memory and RTC backup and for fast Time to First Fix (TTFF).
- Support NMEA0183 v2.2 data protocol and SiRF binary code.
- Enhanced algorithms provide superior navigation performance in urban, canyon and foliage environments.
- For Car Navigation , Marine Navigation, Fleet Management, AVL and Location-Based Services , Auto Pilot ,Personal Navigation or touring devices, Tracking devices/systems and Mapping devices application.
- Ver. A compatible to Holux's GR-88 module on top side, GR-86 module on bottom side.
- Ver. B compatible to Holux's GR-88 module on top side, GR-89 module on bottom side.

■ Specifications

Snap Start	< 3 sec (at < 25 minutes off period).
Hot Start	≦1 sec(typ).
Warm Start	≦38 sec(typ).
Cold Start	≦42 sec(typ).
Satellite Reacquisition Time Accuracy	100 ms
Channels	20
Position Accuracy	25 m CEP without SA
Receiver	L1, C/A code
Protocol	NMEA V2.2, 4800, 8, N, 1 or SiRF Binary
Maximum Altitude	< 60,000 feet

HOLUX *Technology,Inc.* 長天科技股份有限公司
1F, No.30, R&D Rd. II, Hsinchu City 300, 新竹市研發二路30號 1F Tel:886 3 668 7000
Science-based Industrial Park, Taiwan. E-Mail:info@holux.com.tw Web: www.holux.com Fax:886 3 668 7111

Maximum Velocity	< 1,000 knots
Max. Update Rate	1 Hz
RF Connector	MMCX, MCX or SMA
Dimension	40.6(W) x 71.1(L) x 13.5(H) (mm) (height of connector included)
Weight	11 g
Firmware Upgrade	Flash EPROM field programming software available
Time Mark	Output 1 pulse/sec, aligned with GPS time +-0.1 usec
Operating Temperature	-10 to +60 degree C
Storage Temperature	-20 to +70 degree C
Operating Humidity	5% to 95%, no condensing

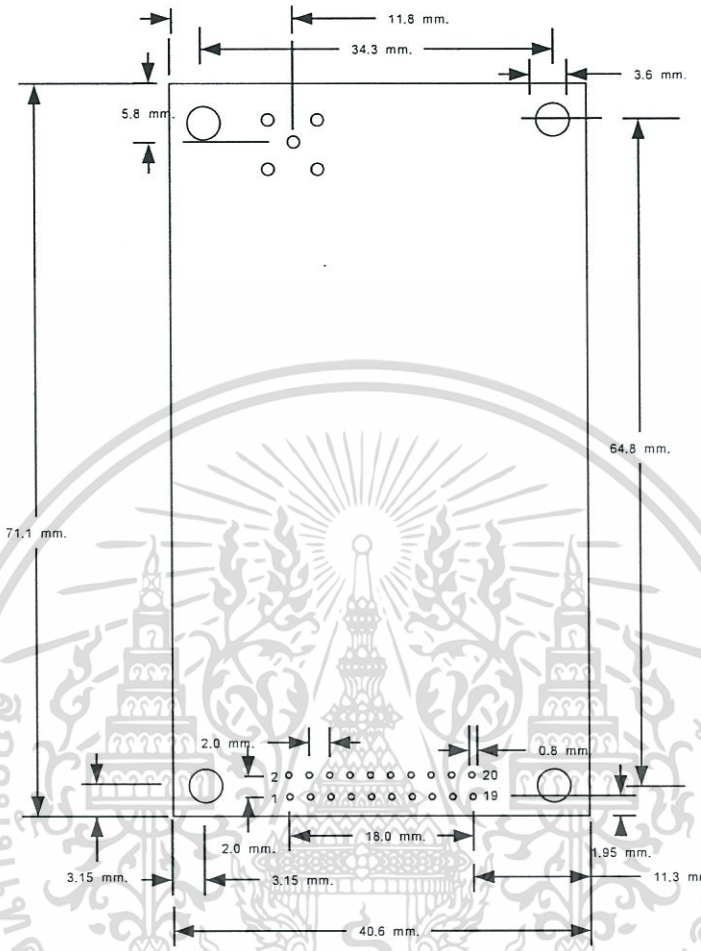
■ Electrical specifications :

Less than 75 mA without antenna

■ Electrical Output specification :

- Interface
  1、 RS-232
  2、 CMOS 3V
  3、 5 GPIO
- NMEA output protocol:
  Baud rate: 4800 bps
  Data bit: 8
  Parity: N
  Stop bit: 1
  Output format: GGA, GSA, GSV, MC.(VTG , GLL, RMS option)
- **Output terminal and definition**
  20-pin header (2.0 mm pitch)
  DB9 female, RS-232 output (optional for TEST board)

HOLUX *Technology,Inc.* 長天科技股份有限公司
1F, No.30, R&D Rd. II, Hsinchu City 300, 新竹市研發二路30號 1F Tel:886 3 668 7000
Science-based Industrial Park, Taiwan. E-Mail:info@holux.com.tw Web: www.holux.com Fax:886 3 668 7111



J4, 20-pin connector (2 mm pitch) pin definition

Pin	Pin Name	Function description
1	**NC**	No function
2	***VCC_5V***	+3.3 ~ +5V DC power input for 5 model
	NC	No function for -3 model
3	***NC***	No function
4	***NC***	No function for -5 model
	VCC_3V	regulated 3V power input for -3 model
5	***nRESET***	Reset input, active low
6	***GPIO1***	General purpose I/O pin
7	***GPIO2***	General purpose I/O pin
8	***GPIO3***	General purpose I/O pin
9	***GPIO4***	General purpose I/O pin, LED flash when tracking
10	***GND***	Ground
11	***TXA***	Serial Data output A
12	***RXA***	Serial Data input A

13	***GND***	Ground
14	***TXB***	Serial Data output B
15	***RXB***	Serial Data input B
16	***GND***	Ground
17	***GPIO5***	General purpose I/O pin, BOOT_SEL, Reserved for re-programming flash.
18	***GND***	Ground
19	***TIMEMARK***	1 PPS time mark output
20	***NC***	No function

J7, 3-pin connector (2.54 mm pitch) pin definition. Available in test board.

Pin	Pin Name	Function description
1	***TXDA***	Serial Data output A, RS-232
2	***RXDA***	Serial Data input A, RS-232
3	***GND***	Ground

J8, 3-pin connector (2.54 mm pitch) pin definition. Available in test board.

Pin	Pin Name	Function description
1	***TXDB***	Serial Data output B, RS-232
2	***RXDB***	Serial Data input B, RS-232
3	***GND***	Ground

J10, 2-pin connector (2.54 mm pitch)

BOOT Selection. Short these two pins at power on stage will set CPU into boot mode. Used for flash program update.

J13, DB9 female connector pin definition. Available in test board.

Pin	Pin Name	Function description
1	***NC***	No function
2	***TXA***	RS-232 signal serial data output
3	***RXA***	RS-232 signal serial data input
4	***NC***	No function
5	***GND***	Ground
6	***NC***	No function
7	***NC***	No function
8	***NC***	No function

Freescale Semiconductor

MPL115A1
Rev 4, 03/2010

Miniature SPI Digital Barometer

MPL115A1
50 to 115 kPa

The MPL115A1 is an absolute pressure sensor with digital output for low cost applications. A miniature 5 x 3 x 1.2 mm LGA package ideally suits it for portable electronics and space constrained applications. Low current consumptions of 5 µA during Active mode and 1 µA during Shutdown (Sleep) mode target battery and other low-power applications. A wide operating temperature range from -40°C to +105°C fits demanding environmental requirements.

MPL115A1 employs a MEMS pressure sensor with a conditioning IC to provide accurate pressure measurement from 50 to 115 kPa. An integrated ADC provides digitized temperature and pressure sensor outputs via a SPI port. Calibration Data is stored in internal ROM. Utilizing raw sensor output, the host microcontroller executes a compensation algorithm to render *Compensated Absolute Pressure* with 1 kPa accuracy.

The MPL115A1 pressure sensor's small form factor, low power capability, precision, and digital output optimize it for barometric measurement applications.

Features

- Digitized pressure and temperature information together with programmed calibration coefficients for host micro use.
- Factory Calibrated
- 50 kPa to 115 kPa Absolute Pressure
- 1 kPa Accuracy
- 2.375 V to 5.5 V Supply
- Integrated ADC
- SPI Interface
- Monotonic Pressure and Temperature Data Outputs
- Surface Mount RoHS Compliant Package

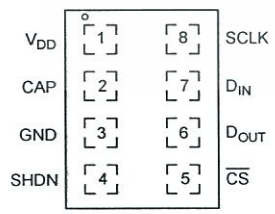
Application Examples

- Barometry (portable and desk-top)
- Altimeters
- Weather Stations
- Hard Disk-Drives (HDD)
- Industrial Equipment
- Health Monitoring
- Air Control Systems

ORDERING INFORMATION									
Device Name	Package Options	Case No.	# of Ports			Pressure Type			Digital Interface
			None	Single	Dual	Gauge	Differential	Absolute	
MPL115A1T1	Tape & Reel	2015	•					•	SPI

LGA PACKAGE

MPL115A1
5.0 mm X 3.0 mm X 1.2 mm MAX



PIN CONNECTIONS

Pin Description

PIN	NAME	FUNCTION
1	VDD	VDD Power Supply Connection.
2	CAP	External Capacitor
3	GND	Ground
4	SHDN	Shutdown (Sleep): Connect to GND to disable the device. When in shut down the part draws < 1µA supply current
5	$\overline{CS}$	$\overline{CS}$: SPI Chip Select line.
6	DOUT	DOUT: Serial data output.
7	DIN	DIN: Serial data input.
8	SCLK	SPI Serial Clock Input.

freescale™ semiconductor

Maximum Ratings

Voltage (with respect to GND unless otherwise noted)

V_{DD} ..... -0.3 V to +5.5 V

SCLK, $\overline{CS}$, D_{IN}, D_{OUT} ..... -0.3 V to V_{DD}+0.3 V

Operating Temperature Range ..... -40°C to +105°C

Storage Temperature Range ..... -40°C to +125°C

Overpressure ..... 1000 kPa

Operating Characteristics

(V_{DD} = 2.375 V to 5.5 V, T_A = -40°C to +105°C, unless otherwise noted. Typical values are at V_{DD} = 3.3 V, T_A = +25°C.

Ref	Parameters	Symbol	Conditions	Min	Typ	Max	Units
1	Operating Supply Voltage	V_{DD}		2.375	3.3	5.5	V
2	Supply Current	I_{DD}	Shutdown (SHDN = GND)	—	—	1	µA
			Standby	—	3.5	10	µA
			Average – at one measurement per second	—	5	—	µA
Pressure Sensor							
3	Range			50	—	115	kPa
4	Resolution			—	0.15	—	kPa
5	Accuracy		-20°C to 85°C	—	±1	—	kPa
6	Conversion Time (*Start Pressure Convert*)	tcp	Time between start convert command and data available in the Pressure register	—	0.6	0.7	ms
Temperature Sensor							
7	Range			-40	—	105	°C
8	Conversion Time (*Start Temperature Convert*)	tct	Time between start convert command and data available in the Temperature register	—	0.6	0.7	ms
9	Conversion Time (*Start Both Convert*)	tcb	Time between start convert command and data available in the Pressure and Temperature registers	—	0.8	1	ms
10	Resolution		Temperature ADC is 472 counts at 25°C	—	-5.35	—	counts/°C
SPI Inputs: SCLK, $\overline{CS}$, D_{IN}							
11	SCLK Clock Frequency	f_{SCLK}	(1)	—	—	8	MHz
12	Low Level Input Voltage	VIL		—	—	$0.3V_{DD}$	V
13	High Level Input Voltage	VIH		$0.7V_{DD}$	—	—	V
SPI Outputs: D_{OUT}							
14	Low Level Output Voltage	VOL1	At 3 mA sink current	0	—	0.4	V
		VOL2	At 6 mA sink current	0	—	0.6	
15	High Level Output Voltage	VOH1	At 3 mA source current	V_{DD} – 0.4 V	—	—	V

1. Nominal maximum SPI clock frequency.

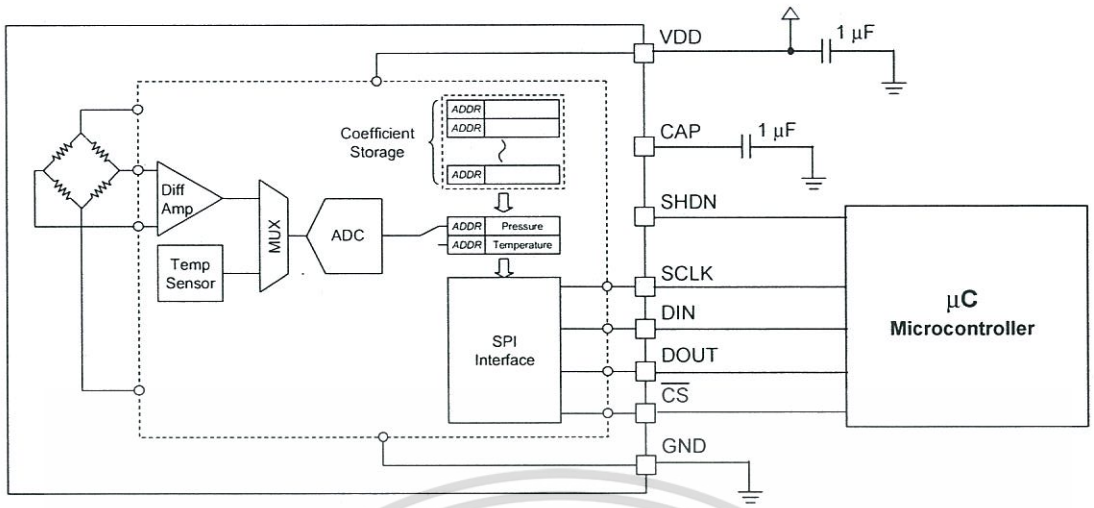


Figure 1. Uncompensated Pressure Sensor Schematic

Compensation

The pressure compensation for MPL115A1 is based on a 2-dimensional, second order polynomial based on Microsystems mst_trimlib library.

The 10-bit compensated pressure output, Pcomp, is calculated as follows:

$$Pcomp = a0 + (b1 + c11*Padc + c12*Tadc) * Padc + (b2 + c22*Tadc) * Tadc$$

Where:

Padc is the 10-bit pressure output of the MPL115A1 ADC,

Tadc is the 10-bit temperature output of the MPL115A1 ADC,

a0 is the pressure offset coefficient,

b1 is the pressure sensitivity coefficient,

c11 is the pressure linearity (2nd order) coefficient,

c12 is the coefficient for temperature sensitivity coefficient (TCS),

b2 is the 1st order temperature offset coefficient (TCO),

c22 is the 2nd order temperature offset coefficient.

Ideally, Pcomp will produce a value of 0 with an input pressure of 50 kPa and will produce a full-scale value of 1023 with an input pressure of 115 kPa.

Coefficient Bit-Width Specs

The table below specifies the initial coefficient bit-width specs for the compensation algorithm.

10-bit Output: Compensation Coefficient Specs							Total Coeff. Bits
	a0	b1	b2	c12	c11	c22	
Total Bits	16	16	16	14	11	11	84
Sign Bits	1	1	1	1	1	1	
Integer Bits	12	2	1	0	0	0	
Fractional Bits	3	13	14	13	11	10	
dec pt zero pad	—	—	—	9	11	15	

Example Binary Format Definitions:

1. Sign = 0, Integer Bits = 8, Fractional Bits = 4 : Coeff = S I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0 . F_3 F_2 F_1 F_0
2. Sign = 1, Integer Bits = 4, Fractional Bits = 7 : Coeff = S I_3 I_2 I_1 I_0 . F_6 F_5 F_4 F_3 F_2 F_1 F_0
3. Sign = 0, Integer Bits = 0, Fractional Bits = 6, dec pt zero pad = 2 : Coeff = S 0 . 0 0 F_5 F_4 F_3 F_2 F_1 F_0
4. Sign = 0, Integer Bits = 0, Fractional Bits = 5, dec pt zero pad = 3 : Coeff = S 0 . 0 0 0 F_4 F_3 F_2 F_1 F_0

NOTE: Negative coefficients (Sign = 1) are coded in 2's complement notation.

Coefficient Address Map

Address	Coefficient
$04	a0 MS Byte
$05	a0 LS Byte
$06	b1 MS Byte
$07	b1 LS Byte
$08	b2 MS Byte
$09	b2 LS Byte
$0A	c12 MS Byte
$0B	c12 LS Byte
$0C	c11 MS Byte
$0D	c11 LS Byte
$0E	c22 MS Byte
$0F	c22 LS Byte

For coefficients with less than 16 bits, the lower lsbs are zero. For example, c11 is 11 bits and is stored into 2 bytes as follows:

c11 MS byte = c11[10:3] = [$c11_{b10}$, $c11_{b9}$, $c11_{b8}$, $c11_{b7}$, $c11_{b6}$, $c11_{b5}$, $c11_{b4}$, $c11_{b3}$]

NOTE: c11 LS byte = c11[2:0] & "00000" = [$c11_{b2}$, $c11_{b1}$, $c11_{b0}$, 0 , 0 , 0 , 0 , 0]

Solder Recommendations

1. Use SAC solder alloy (i.e., Sn-Ag-Cu) with a melting point of about 217°C. It is recommended to use SAC305 (i.e., Sn-3.0 wt.% Ag-0.5 wt.% Cu).
2. Reflow
   - Ramp up rate: 2 to 3 C/s.
   - Preheat flat (soak): 110 to 130s.
   - Reflow peak temperature: 250°C to 260°C (depends on exact SAC alloy composition).
   - Time above 217°C: 40 to 90s (depends on board type, thermal mass of the board/quantities in the reflow).
   - Ramp down: 5 to 6 C/s.
   - Using an inert reflow environment (with O2 level about 5 to 15 ppm).

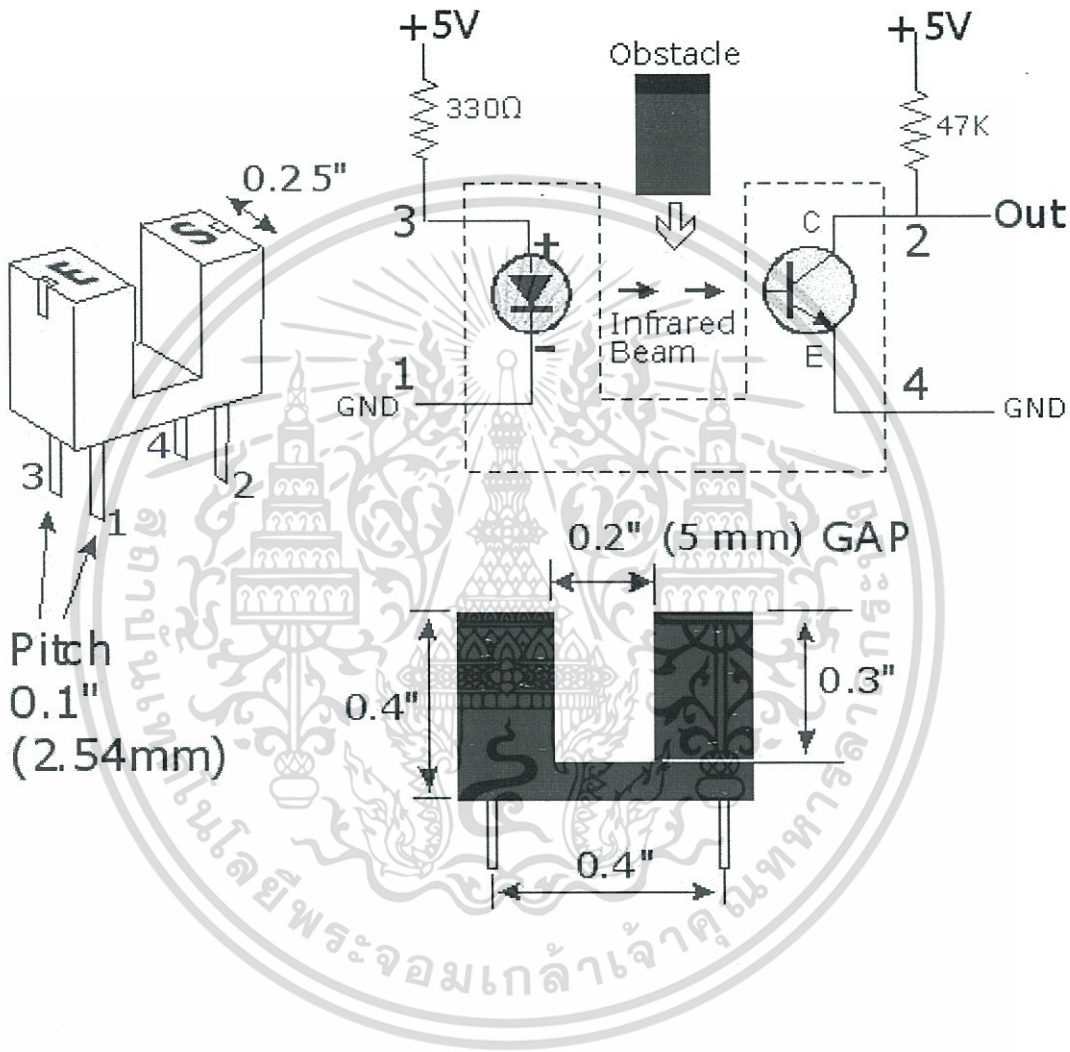
NOTE: The stress level and signal offset of the device also depends on the board type, board core material, board thickness and metal finishing of the board.

Handling Recommendations

It is recommended to handle the MPL115A Pressure Sensor with a vacuum pick and place tool. Sharp objects utilized to move the MPL115A Pressure Sensor increase the possibility of damage via a foreign object/tool into the small exposed port.

The sensor die is sensitive to light exposure. Direct light exposure through the port hole can lead to varied accuracy of pressure measurement. Avoid such exposure to the port during normal operation.

OPTICAL SENSOR



Small, Low Power, 3-Axis ±3 *g* Accelerometer

ADXL335

FEATURES

- 3-axis sensing
- Small, low profile package
  - 4 mm × 4 mm × 1.45 mm LFCSP
- Low power : 350 µA (typical)
- Single-supply operation: 1.8 V to 3.6 V
- 10,000 *g* shock survival
- Excellent temperature stability
- BW adjustment with a single capacitor per axis
- RoHS/WEEE lead-free compliant

APPLICATIONS

- Cost sensitive, low power, motion- and tilt-sensing applications
  - Mobile devices
  - Gaming systems
  - Disk drive protection
  - Image stabilization
  - Sports and health devices

GENERAL DESCRIPTION

The ADXL335 is a small, thin, low power, complete 3-axis accelerometer with signal conditioned voltage outputs. The product measures acceleration with a minimum full-scale range of ±3 *g*. It can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion, shock, or vibration.

The user selects the bandwidth of the accelerometer using the C_X, C_Y, and C_Z capacitors at the X_{OUT}, Y_{OUT}, and Z_{OUT} pins. Bandwidths can be selected to suit the application, with a range of 0.5 Hz to 1600 Hz for the X and Y axes, and a range of 0.5 Hz to 550 Hz for the Z axis.

The ADXL335 is available in a small, low profile, 4 mm × 4 mm × 1.45 mm, 16-lead, plastic lead frame chip scale package (LFCSP_LQ).

FUNCTIONAL BLOCK DIAGRAM

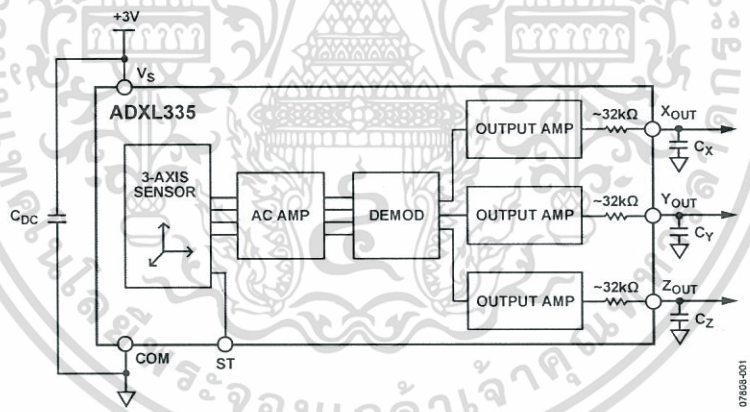


Figure 1.

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781.329.4700 www.analog.com
Fax: 781.461.3113

SPECIFICATIONS

T_A = 25°C, V_S = 3 V, C_X = C_Y = C_Z = 0.1 µF, acceleration = 0 *g*, unless otherwise noted. All minimum and maximum specifications are guaranteed. Typical specifications are not guaranteed.

Table 1.

Parameter	Conditions	Min	Typ	Max	Unit
SENSOR INPUT	Each axis				
Measurement Range		±3	±3.6		*g*
Nonlinearity	% of full scale		±0.3		%
Package Alignment Error			±1		Degrees
Interaxis Alignment Error			±0.1		Degrees
Cross-Axis Sensitivity[1]			±1		%
SENSITIVITY (RATIOMETRIC)[2]	Each axis				
Sensitivity at X_{OUT}, Y_{OUT}, Z_{OUT}	V_S = 3 V	270	300	330	mV/*g*
Sensitivity Change Due to Temperature[3]	V_S = 3 V		±0.01		%/°C
ZERO *g* BIAS LEVEL (RATIOMETRIC)					
0 *g* Voltage at X_{OUT}, Y_{OUT}	V_S = 3 V	1.35	1.5	1.65	V
0 *g* Voltage at Z_{OUT}	V_S = 3 V	1.2	1.5	1.8	V
0 *g* Offset vs. Temperature			±1		mg/°C
NOISE PERFORMANCE					
Noise Density X_{OUT}, Y_{OUT}			150		µg/√Hz rms
Noise Density Z_{OUT}			300		µg/√Hz rms
FREQUENCY RESPONSE[4]					
Bandwidth X_{OUT}, Y_{OUT}[5]	No external filter		1600		Hz
Bandwidth Z_{OUT}[5]	No external filter		550		Hz
R_{FILT} Tolerance			32 ± 15%		kΩ
Sensor Resonant Frequency			5.5		kHz
SELF-TEST[6]					
Logic Input Low			+0.6		V
Logic Input High			+2.4		V
ST Actuation Current			+60		µA
Output Change at X_{OUT}	Self-Test 0 to Self-Test 1	−150	−325	−600	mV
Output Change at Y_{OUT}	Self-Test 0 to Self-Test 1	+150	+325	+600	mV
Output Change at Z_{OUT}	Self-Test 0 to Self-Test 1	+150	+550	+1000	mV
OUTPUT AMPLIFIER					
Output Swing Low	No load		0.1		V
Output Swing High	No load		2.8		V
POWER SUPPLY					
Operating Voltage Range		1.8		3.6	V
Supply Current	V_S = 3 V		350		µA
Turn-On Time[7]	No external filter		1		ms
TEMPERATURE					
Operating Temperature Range		−40		+85	°C

[1] Defined as coupling between any two axes.
[2] Sensitivity is essentially ratiometric to V_S.
[3] Defined as the output change from ambient-to-maximum temperature or ambient-to-minimum temperature.
[4] Actual frequency response controlled by user-supplied external filter capacitors (C_X, C_Y, C_Z).
[5] Bandwidth with external capacitors = 1/(2 × π × 32 kΩ × C). For C_X, C_Y = 0.003 µF, bandwidth = 1.6 kHz. For C_Z = 0.01 µF, bandwidth = 500 Hz. For C_X, C_Y, C_Z = 10 µF, bandwidth = 0.5 Hz.
[6] Self-test response changes cubically with V_S.
[7] Turn-on time is dependent on C_X, C_Y, C_Z and is approximately 160 × C_X or C_Y or C_Z + 1 ms, where C_X, C_Y, C_Z are in microfarads (µF).

ABSOLUTE MAXIMUM RATINGS

Table 2.

Parameter	Rating
Acceleration (Any Axis, Unpowered)	10,000 *g*
Acceleration (Any Axis, Powered)	10,000 *g*
V_S	−0.3 V to +3.6 V
All Other Pins	(COM − 0.3 V) to (V_S + 0.3 V)
Output Short-Circuit Duration (Any Pin to Common)	Indefinite
Temperature Range (Powered)	−55°C to +125°C
Temperature Range (Storage)	−65°C to +150°C

Stresses above those listed under Absolute Maximum Ratings may cause permanent damage to the device. This is a stress rating only; functional operation of the device at these or any other conditions above those indicated in the operational section of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ESD CAUTION

ESD (electrostatic discharge) sensitive device. Charged devices and circuit boards can discharge without detection. Although this product features patented or proprietary protection circuitry, damage may occur on devices subjected to high energy ESD. Therefore, proper ESD precautions should be taken to avoid performance degradation or loss of functionality.

PIN CONFIGURATION AND FUNCTION DESCRIPTIONS

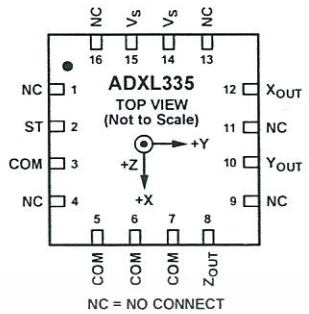


Figure 2. Pin Configuration

Table 3. Pin Function Descriptions

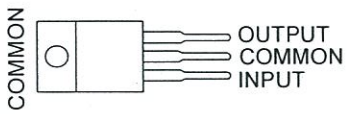
Pin No.	Mnemonic	Description
1	NC	No Connect[1].
2	ST	Self-Test.
3	COM	Common.
4	NC	No Connect[1].
5	COM	Common.
6	COM	Common.
7	COM	Common.
8	Z_{OUT}	Z Channel Output.
9	NC	No Connect[1].
10	Y_{OUT}	Y Channel Output.
11	NC	No Connect[1].
12	X_{OUT}	X Channel Output.
13	NC	No Connect[1].
14	V_S	Supply Voltage (1.8 V to 3.6 V).
15	V_S	Supply Voltage (1.8 V to 3.6 V).
16	NC	No Connect[1].
EP	Exposed Pad	Not internally connected. Solder for mechanical integrity.

[1]NC pins are not internally connected and can be tied to COM pins, unless otherwise noted.

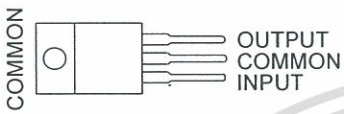
SLVS056J – MAY 1976 – REVISED MAY 2003

- 3-Terminal Regulators
- Output Current up to 1.5 A
- Internal Thermal-Overload Protection
- High Power-Dissipation Capability
- Internal Short-Circuit Current Limiting
- Output Transistor Safe-Area Compensation

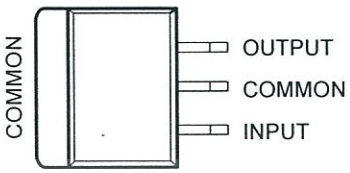
KC (TO-220) PACKAGE
(TOP VIEW)



KCS (TO-220) PACKAGE
(TOP VIEW)



KTE PACKAGE
(TOP VIEW)



description/ordering information

This series of fixed-voltage integrated-circuit voltage regulators is designed for a wide range of applications. These applications include on-card regulation for elimination of noise and distribution problems associated with single-point regulation. Each of these regulators can deliver up to 1.5 A of output current. The internal current-limiting and thermal-shutdown features of these regulators essentially make them immune to overload. In addition to use as fixed-voltage regulators, these devices can be used with external components to obtain adjustable output voltages and currents, and also can be used as the power-pass element in precision regulators.

ORDERING INFORMATION

T_J	$V_{O(NOM)}$ (V)	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 125°C	5	POWER-FLEX (KTE)	Reel of 2000	μA7805CKTER	μA7805C
		TO-220 (KC)	Tube of 50	μA7805CKC	μA7805C
		TO-220, short shoulder (KCS)	Tube of 20	μA7805CKCS	
	8	POWER-FLEX (KTE)	Reel of 2000	μA7808CKTER	μA7808C
		TO-220 (KC)	Tube of 50	μA7808CKC	μA7808C
		TO-220, short shoulder (KCS)	Tube of 20	μA7808CKCS	
	10	POWER-FLEX (KTE)	Reel of 2000	μA7810CKTER	μA7810C
		TO-220 (KC)	Tube of 50	μA7810CKC	μA7810C
	12	POWER-FLEX (KTE)	Reel of 2000	μA7812CKTER	μA7812C
		TO-220 (KC)	Tube of 50	μA7812CKC	μA7812C
		TO-220, short shoulder (KCS)	Tube of 20	μA7812CKCS	
	15	POWER-FLEX (KTE)	Reel of 2000	μA7815CKTER	μA7815C
		TO-220 (KC)	Tube of 50	μA7815CKC	μA7815C
		TO-220, short shoulder (KCS)	Tube of 20	μA7815CKCS	
	24	POWER-FLEX (KTE)	Reel of 2000	μA7824CKTER	μA7824C
		TO-220 (KC)	Tube of 50	μA7824CKC	μA7824C

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265