Smart Home for Pet

รายวิชา 242-402 Computer Engineering Project II

ภาคการศึกษา 2/2561

รายชื่อผู้จัดทำ

นายสหรัฐ จันทร์ทิพย์ รหัสนักศึกษา 5835512024

อาจารย์ที่ปรึกษา ดร.นพพณ เลิศชูวงศา

อาจารย์ที่ปรึกษาร่วม อ.คมสันต์ กาญจนสิทธิ์

อาจารย์ที่ปรึกษาร่วม อ.พัชรี เทพนิมิตร

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยสงขลานครินทร์

ชื่อโครงงาน Smart Home for Pet

ผู้จัดทำ นายสหรัฐ จันทร์ทิพย์ 5835512024

ภาควิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2561

อาจารย์ที่ปรึกษาโครงงาน

.....

(ดร.นพพณ เลิศชูวงศา)

คณะกรรมการสอบ

....

(ดร.นพพณ เลิศชูวงศา) (อ.คมสันต์ กาญจนสิทธิ์) (อ.พัชรี เทพนิมิตร)

โครงงานนี้เป็นส่วนหนึ่งของรายวิชา Computer Engineering Project I-II ตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์

.....

(อ.พัชรี เทพนิมิต)

ผู้จัดการหลักสูตร
ภาควิชาวิศวกรรมคอมพิวเตอร์

หนังสือรับรองความเป็นเอกลักษณ์

ผู้จัดทำที่ได้ลงนามท้ายนี้ ขอรับรองว่ารายงานฉบับนี้เป็นรายงานที่มีความเป็นเอกลักษณ์ โดยที่ผู้จัดทำไม่ได้มีการคัดลอกมาจากที่ใดเลย เนื้อหาทั้งหมดถูกรวบรวมจากการพัฒนาในขั้นตอนต่าง ๆ ของการจัดทำโครงงาน หากมีส่วนใดที่จำเป็นต้องนำเอาข้อความจากผลงานของผู้อื่น หรือบุคคลอื่นใดที่ไม่ใช่ตัวข้าพเจ้า ข้าพเจ้าได้ทำอ้างอิงถึงเอกสารเหล่านั้นไว้อย่างเหมาะสม และขอรับรองว่ารายงานฉบับนี้ไม่เคยเสนอต่อสถาบันใดมาก่อน

ผู้จัดทำ

.....

(นายสหรัฐ จันทร์ทิพย์)

ชื่อโครงงาน บ้านสัตว์เลี้ยงอัจฉริยะ

ผู้จัดทำ นายสหรัฐ จันทร์ทิพย์ 5835512024

ภาควิชา วิศวกรรมคอมพิวเตอร์

ปีการศึกษา 2561

บทคัดย่อ

โครงงานนี้เป็นการออกแบบที่อยู่อาศัยของสัตว์เลี้ยงเพื่อให้ผู้เลี้ยงสามารถดูแลสัตว์เลี้ยงได้แม้จะไม่ได้อยู่ที่บ้านโดย ให้อาหารแก่สัตว์เลี้ยงได้เพียงแค่ตั้งเวลาที่ต้องการจะให้ การเติมน้ำในถาดน้ำของสัตว์เลี้ยงแบบอัตโนมัติผ่านตัววัดระดับน้ำ รวมไปถึงการบอกอุณหภูมิและความชื้นภายในบ้านสัตว์เลี้ยงถ้ามีการผิดแปลกอุณหภูมิร้อนเกินไปจะทำการสั่งให้พัดลมปรับอากาศทำงานตามที่ตั้งไว้เพื่อให้อุณหภูมิภายในบ้านสัตว์เลี้ยงเหมาะสมอยู่ตลอดเวลา โดยการดูผ่าน website โดยจะมีการแสดงเป็นตารางและกราฟ ทั้งหมดนี้จะเป็นการตอบโจทย์การเข้ามาช่วยเหลือในเหตุการณ์ที่ผู้เลี้ยงไม่มีเวลาแต่จำเป็นต้องเลี้ยงสัตว์เลี้ยงนั้นเอง

คำสำคัญ: website และ automatic

Project Title Smart Home for Pet

Author Mr.Saharat Chanthip 5835512024

Department Computer Engineering

Academic Year 2018

Abstract

This project is a design of a pet residence so that the shepherd can take care of the pet even if not at home. Give food to pets just by setting the time they want to give. Automatic filling of pet water trays through the water level meter Including telling the temperature and humidity inside the pet house, if there is an abnormal temperature, too hot, will order the air-conditioning fan to work as set, so that the temperature inside the pet house is always appropriate by viewing through the website Are displayed as tables and graphs All of this will be a response to help in the event that the shepherd has no time but needs to feed that pet.

Keywords: website and automatic

คำนำหรือกิตติกรรมประกาศ

ขอขอบคุณข้อมูลความรู้จากสื่อการเรียนออนไลน์ต่างๆ ที่ให้ความรู้เกี่ยวกับการทำงานของตัว Hardware แต่ละชนิดว่ามีการทำงานเป็นอย่างไรบ้าง Data Sheet ต่างๆ และ การใช้งานเบื้องต้น รวมไปถึง ในส่วนสื่อออนไลน์ต่างๆ ที่ให้ความรู้เกี่ยวกับ Software ที่ใช้กับตัวอุปกรณ์ Hardware และจะขาดไม่ได้เลย อาจารย์ที่ปรึกษาที่คอยให้คำแนะนำทั้งด้านความรู้ ข้อมูล และการค้นคว้าต่างๆ ดร.นพพน เลิศชูวงศา ขอขอบคุณเป็นอย่างสูง

นายสหรัฐ จันทร์ทิพย์

ผู้จัดทำ

22 กุมภาพันธ์ 2561

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

สารบัญรูปภาพ(ต่อ)

สารบัญตาราง

สารบัญคำย่อ

IDE	Integrated Development Environment
OS	Operating System
RAM	Random Access Memory
DC	Direct Current
AC	Alternating Current
RTC	Real Time Clock
IoT	Internet of Thing

บทที่ 1 บทนำ

1.1 ความเป็นมา

เนื่องจากในปัจจุบันผู้คนส่วนใหญ่นิยมเลี้ยงสัตว์เลี้ยงกันเป็นจำนวนมากไม่ว่าจะเป็น สุนัข แมว กระต่าย หรือสัตว์ตัวเล็กๆน่ารักต่างๆ จึงได้ทำการคิดค้นบ้านสัตว์เลี้ยงอัจฉริยะขึ้นมาเพื่อตอบโจทย์สำหรับผู้ที่ต้องการจะเลี้ยงสัตว์แต่ไม่มีเวลาดูแลได้โดยจะเป็นระบบที่อำนวยความสะดวกของผู้เลี้ยงให้สามารถเลี้ยงสัตว์เลี้ยงของตัวเองได้แม้ไม่ได้อยู่ที่บ้านก็ตาม

1.2 วัตถุประสงค์ของโครงงาน

- เพื่อช่วยเหลือผู้ที่ต้องการเลี้ยงสัตว์แต่ไม่มีเวลาที่จะดูแลการให้น้ำให้อาหารแกสัตว์เลี้ยงของท่าน
- แบ่งเบาภาระของท่านโดยเราสามารถเลี้ยงสัตว์ได้แม้ไม่ต้องอยู่ที่บ้าน

1.3 ขอบเขตของโครงงาน

- สามารถกำหนดเวลาการให้เวลาอาหารสัตว์เลี้ยงได้ตามที่เจ้าของต้องการ โดยสามารถตั้งเวลาเครื่องให้อาหารสัตว์เลี้ยงได้ตามที่เจ้าของต้องการ
- สามารถอ่านค่าอุณหภูมิภายในบ้านสัตว์เลี้ยงได้เพื่อแจ้งให้ผู้เลี้ยงทราบ และทำการปรับอุณหภูมิให้เหมาะสมอยู่เสมอ
- สามารถเติมน้ำลงไปในถาดน้ำของสัตว์เลี้ยงได้เมื่อน้ำในถาดเลี้ยงเหลือน้อยกว่าค่าที่ตั้งเอาไว้

1.4 ขั้นตอนในการดำเนินงาน

- Prepare Project
    - ศึกษาการทำงานของอุปกรณ์ต่างๆที่ต้องนำมาใช้
    - ศึกษาการเขียน Web Site ที่ใช้ทำการร่วมกับอุปกรณ์ Hardware
    - ศึกษาการเชื่อมต่อและการส่งข้อมูลระหว่าง Website กับอุปกรณ์ Hardware
    - เขียนรูปเล่มโครงงาน
    - วางรูปแบบแผนงานและโครงสร้างของตัวโครงงาน
- Project 1
    - เริ่มลงมือทำตัว Hardware ต่างๆที่ต้องใช้ภายใน
    - เขียน Code เพื่อให้อุปกรณ์ Hardware ทำงานได้อย่างสมบูรณ์และถูกต้อง

- ศึกษา Hardware และ Software ที่เกี่ยวข้องกับตัว Hardware เพิ่มเติมเพื่อนำความรู้มาใช้กับตัว Project หลักให้มากขึ้น
- แก้ไขปัญหาที่เกิดขึ้นกับตัว Project ให้สามารถทำงานได้เหมือนเดิม
- ทำการ upload ข้อมูลขึ้นตัว Server เพื่อนำไปทำการใช้ข้อมูลเพื่อแจ้งผ่าน website

- Project 2
  - ทำการเชื่อมต่อข้อมูลและแสดงข้อมูลผ่านหน้า website ที่เราต้องการให้แสดง
  - ทำการรวมระบบแต่ละส่วนให้กลายเป็นหนึ่งเดียวกันให้กลายเป็นระบบๆเดียว
  - ทำระบบเครื่องให้น้ำอัตโนมัติของสัตว์เลี้ยงได้เสร็จสิ้น
  - ปรับปรุงแก้ไขปัญหาที่เกิดขึ้นให้ดีกว่าเดิมมากยิ่งขึ้น

1.5 ประโยชน์ที่คาดว่าจะได้รับ

- ช่วยแก้ปัญหาเวลาเจ้าของไม่มีเวลาดูแลสัตว์เลี้ยงได้
- สามารถแก้ไขปัญหาที่จะเกิดขึ้นได้ในกรณีที่อาหารไม่มี หรือ น้ำไม่เพียงพอในถาดอาหารหรือน้ำ
- สามารถปรับอุณหภูมิของบ้านสัตว์เลี้ยงให้อุณหภูมิที่พอเหมาะต่อสัตว์เลี้ยงตลอดเวลา

1.6 สถานที่ทำโครงงาน

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตภูเก็ต

1.7 เครื่องมือที่ใช้ในการพัฒนา

Hardware

- Computer intel Core i7 /Ram 8GB
- สายไฟ
- บอร์ด Anduino UNO
- Sensor สะท้อนแสง
- RTC DS3231
- Servo Motor
- บอร์ด NodeMCU
- DHT22

- พัดลม
- Motor Drive L298N
- Relay
- Non-Contact Water

Software

- Git hub
- Anduino IDE
- Firebase

บทที่ 2 ความรู้พื้นฐาน

2.1 Overview

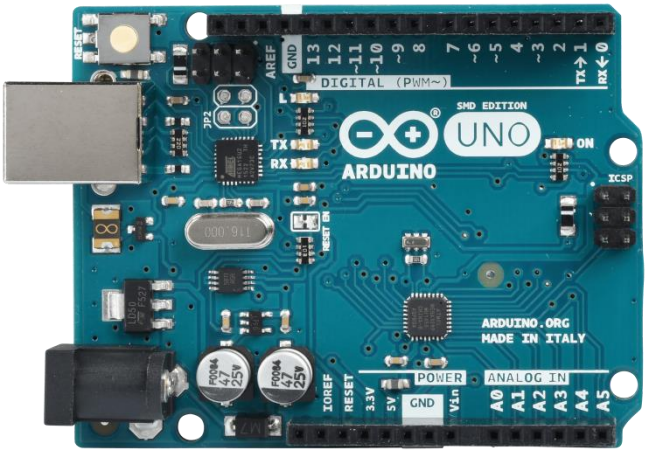
ในส่วนของตัวโครงงานนั้นจะประกอบไปด้วยการนำความรู้ทางด้านบอร์ด arduino เพื่อนำมาประยุกต์ใช้กับตัวให้อาหารและน้ำอัตโนมัติ และตัวปรับอุณหภูมิภายในบ้านสัตว์เลี้ยง เพื่อให้สัตว์เลี้ยงได้รับความสะดวกสบายมากยิ่งขึ้น

2.1.1 Arduino

โดย Arduino ที่เลือกใช้จะเป็นบอร์ด Arduino UNO ซึ่งเอามาประยุกต์ใช้ไปทำอุปกรณ์ต่างๆ ที่จะนำมาใช้ได้เป็นอย่างดีซึ่งข้อมูลของบอร์ดจะเป็นดังนี้

Arduino อ่านว่า (อา-ดู-อิ-โน่ หรือ อาดุยโน่) เป็นบอร์ดไมโครคอนโทรเลอร์ตระกูล AVR ที่มีการพัฒนาแบบ Open Source คือมีการเปิดเผยข้อมูลทั้งด้าน Hardware และ Software ตัว บอร์ด Arduino ถูกออกแบบมาให้ใช้งานได้ง่าย ดังนั้นจึงเหมาะสำหรับผู้เริ่มต้นศึกษา ทั้งนี้ผู้ใช้งานยังสามารถดัดแปลงเพิ่มเติม พัฒนาต่อยอดทั้งตัวบอร์ด หรือโปรแกรมต่อได้อีกด้วย

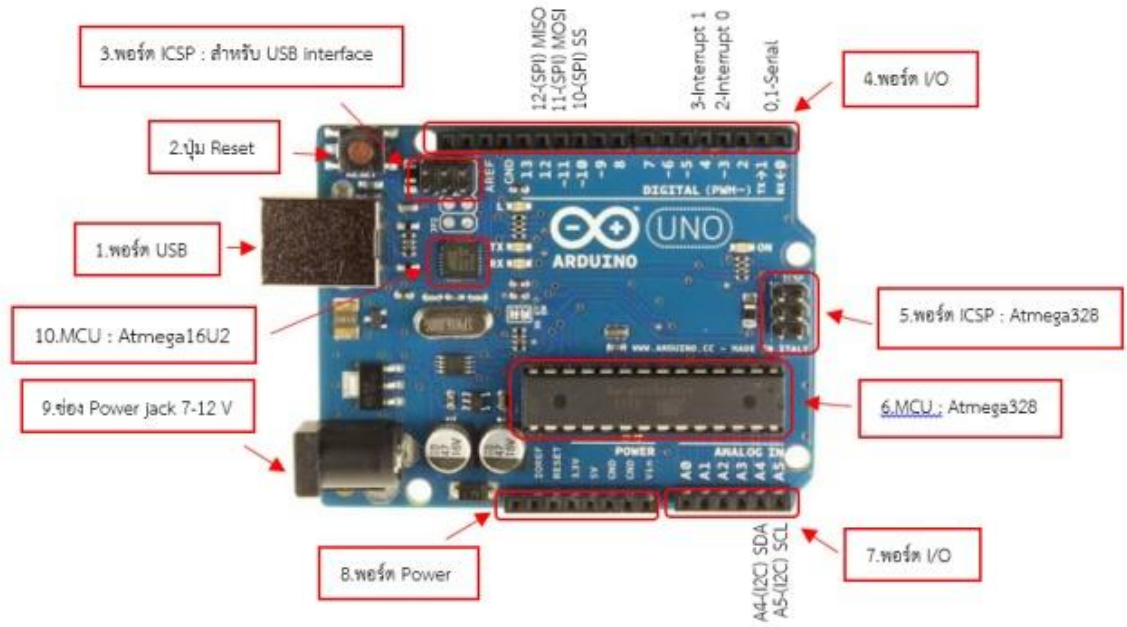
ความง่ายของบอร์ด Arduino ในการต่ออุปกรณ์เสริมต่างๆ คือผู้ใช้งานสามารถต่อวงจรอิเล็กทรอนิคส์จากภายนอกแล้วเชื่อมต่อเข้ามาที่ขา I/O ของบอร์ด หรือเพื่อความสะดวกสามารถเลือกต่อกับบอร์ดเสริม (Arduino Shield) ประเภทต่างๆ เช่น Arduino XBee Shield, Arduino Music Shield, Arduino Relay Shield, Arduino Wireless Shield, Arduino GPRS Shield เป็นต้น มาเสียบกับบอร์ดบนบอร์ด Arduino แล้วเขียนโปรแกรมพัฒนาต่อได้เลย



รูปที่ 2-1 บอร์ด Arduino UNO

จุดเด่นที่ทำให้บอร์ด Arduino เป็นที่นิยม

- ง่ายต่อการพัฒนา มีรูปแบบคำสั่งพื้นฐาน ไม่ซับซ้อนเหมาะสำหรับผู้เริ่มต้น
- มี Arduino Community กลุ่มคนที่ร่วมกันพัฒนาที่แข็งแรง
- Open Hardware ทำให้ผู้ใช้สามารถนำบอร์ดไปต่อยอดใช้งานได้หลายด้าน
- ราคาไม่แพง
- Cross Platform สามารถพัฒนาโปรแกรมบน OS ใดก็ได้



รูปที่ 2-2 ส่วนต่างๆของบอร์ด Arduino UNO

Layout & Pin out Arduino Board (Model: Arduino UNO R3)

1. USB Port: ใช้สำหรับต่อกับ Computer เพื่ออับโหลดโปรแกรมเข้า MCU และจ่ายไฟให้กับบอร์ด
2. Reset Button: เป็นปุ่ม Reset ใช้กดเมื่อต้องการให้ MCU เริ่มการทำงานใหม่
3. ICSP Port ของ Atmega16U2 เป็นพอร์ตที่ใช้โปรแกรม Visual Com port บน Atmega16U2
4. I/O Port Digital I/O ตั้งแต่ขา D0 ถึง D13 นอกจากนี้ บาง Pin จะทำหน้าที่อื่นๆ เพิ่มเติมด้วย เช่น Pin0,1 เป็นขา Tx,Rx Serial, Pin3,5,6,9,10 และ 11 เป็นขา PWM
5. ICSP Port: Atmega328 เป็นพอร์ตที่ใช้โปรแกรม Bootloader
6. MCU: Atmega328 เป็น MCU ที่ใช้บนบอร์ด Arduino
7. I/O Port: นอกจากจะเป็น Digital I/O แล้ว ยังเปลี่ยนเป็น ช่องรับสัญญาณอนาล็อก ตั้งแต่ขา A0-A5
8. Power Port: ไฟเลี้ยงของบอร์ดเมื่อต้องการจ่ายไฟให้กับวงจรภายนอก ประกอบด้วยขาไฟเลี้ยง +3.3 V, +5V, GND, Vin

9. Power Jack: รับไฟจาก Adapter โดยที่แรงดันอยู่ระหว่าง 7-12 V
10. MCU ของ Atmega16U2 เป็น MCU ที่ทำหน้าที่เป็น USB to Serial โดย Atmega328 จะติดต่อกับ Computer ผ่าน Atmega16U2

การเข้าใช้งานหรือเขียนคำสั่งให้บอร์ด Arduino ทำงานนั้นเราจะต้องมีตัว Software ชื่อ ArduinoIDE เพื่อเป็นตัวในการเขียนและป้อนคำสั่งที่เราต้องการลงไปในบอร์ด Arduino โดยสามารถโหลดได้จากเว็บหลักของ Arduino เลยคือ https://www.arduino.cc/en/main/software

2.1.2 Motor Drive L298N สำหรับขับ Motor

หลักการทำงาน

วงจร H-Bridge ของ L298N จะขับกระแสเข้ามอเตอร์ ตามขั้วที่กำหนดด้วยลอจิกเพื่อควบคุมทิศทาง ส่วนความเร็วของมอเตอร์นั้นจะถูกควบคุมด้วย สัญญาณ (PWM Pulse Width Modulation)

PWM หมายถึง การควบคุมช่วงจังหวะการทำงานของอิเล็กตรอน ลองจินตนาการถึงแปรงขดลวดในมอเตอร์เป็นระหัดวิดน้ำและอิเล็กตรอนเป็นน้ำที่ตกลงมาจากระหัดวิดน้ำ

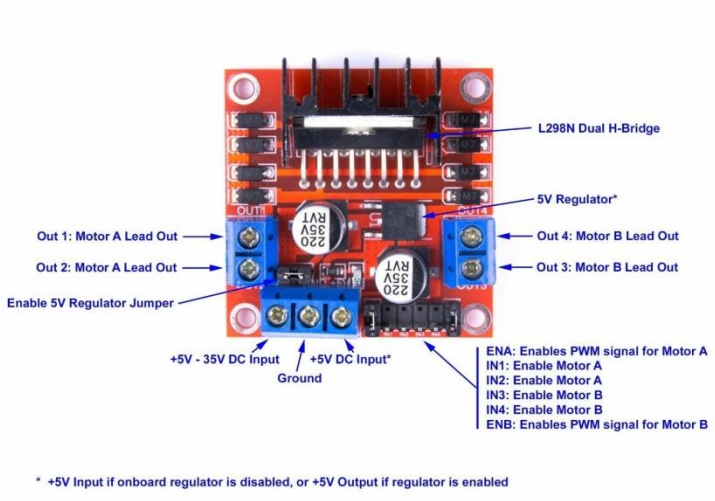
ค่าแรงดันไฟฟ้าก็คล้ายกับกระแสน้ำที่ไหลผ่านระหัดวิดน้ำด้วยความเร็วคงที่ ยิ่งกระแสน้ำไหลเร็วเท่าไรก็จะหมายความว่าแรงดันไฟฟ้ายิ่งสูงขึ้น แต่มอเตอร์มีอัตราความเร็วคงที่และสามารถเสียหายได้หากมีแรงดันไฟฟ้าสูงไหลผ่านหรือหยุดทันทีเพื่อที่จะหยุดมอเตอร์ ดังนั้น PWM คล้ายกับการควบคุมระหัดวิดน้ำให้ตักน้ำในจังหวะคงที่ที่กระแสน้ำคงที่ ยิ่งระหัดวิดน้ำหมุนเร็วเท่าไรช่วงของ pulse ก็จะยาวขึ้น ในทางกลับกันถ้าระหัดวิดน้ำหมุนช้าช่วงของ pulse จะสั้นลง ดังนั้นเพื่อยืดอายุการใช้งานของมอเตอร์จึงควรที่จะควบคุมมอเตอร์ด้วย PWM

พิจารณาโครงสร้างโค้ดคร่าวๆของ Arduino

โค้ดการทำงานของ Arduino มีการพัฒนาขึ้นเรื่อยๆ แต่ยังไม่มีสร้าง library ที่เกี่ยวกับตัว L298N Dual H-Bridge เพื่อควบคุมมอเตอร์ดังนั้นผู้ใช้งานจึงต้องประกาศพินเพื่อใช้งานขึ้นมาเอง สามารถใช้โค้ด int dir(number)Pin(letter) ต่อเข้ากับพินดิจิตอลที่เลือกใช้ แค่นี้ก็สามารถทำงานได้อย่างถูกต้องและช่วยให้ตัว L298N Dual H-Bridge ควบคุมมอเตอร์ได้อย่างอเนกประสงค์ถ้าบอร์ด Arduino ที่เลือกใช้งานมีพินหลายตัว และถ้าต้องการปรับความเร็วของมอเตอร์ด้วย PWM สามารถใช้คำสั่ง int speedPin(letter) แล้วต่อเข้ากับพินที่เลือกใช้ หากต้องการวิธีลัดเพื่อใช้งาน PWM อย่างรวดเร็วสามารถเลือกพินใช้งานได้ตามรายการด้านล่าง

AT MEGA –PWM 2-13 และ 44-46 ตั้งค่าเอาท์พุทของ PWM ให้เป็น 8 bit ด้วยฟังก์ชั่น analogWrite()

UNO-PWM 3, 5, 6, 9, 10 และ 11 ตั้งค่าเอาท์พุทของ PWM ให้เป็น 8 bit ด้วยฟังก์ชั่น analogWrite()



รูปที่ 2-3 Pin ต่างๆของ L298N

รายละเอียดของบอร์ด

- Out 1: ช่องต่อขั้วไฟของมอเตอร์ A
- Out 2: ช่องต่อขั้วไฟของมอเตอร์ A
- Out 3: ช่องต่อขั้วไฟของมอเตอร์ B
- Out 4: ช่องต่อขั้วไฟของมอเตอร์ B
- 12V: ช่องจ่ายไฟเลี้ยงมอเตอร์ 12V (ต่อได้ตั้งแต่ 5V ถึง 35V)
- GND: ช่องต่อไฟลบ (Ground)
- 5V: ช่องจ่ายไฟเลี้ยงมอเตอร์ 5V (หากมีการต่อไฟเลี้ยงที่ช่อง 12V แล้ว
- ช่องนี้จะทำหน้าที่จ่ายไฟออก เป็น 5V Output
- สามารถต่อไฟจากช่องนี้ไปเลี้ยงบอร์ด Arduino ได้
- ENA: ช่องต่อสัญญาณ PWM สำหรับมอเตอร์ A
- IN1: ช่องต่อสัญญาณลอจิกเพื่อควบคุมทิศทางของมอเตอร์ A
- IN2: ช่องต่อสัญญาณลอจิกเพื่อควบคุมทิศทางของมอเตอร์ A
- IN3: ช่องต่อสัญญาณลอจิกเพื่อควบคุมทิศทางของมอเตอร์ B
- IN4: ช่องต่อสัญญาณลอจิกเพื่อควบคุมทิศทางของมอเตอร์ B
- ENB: ช่องต่อสัญญาณ PWM สำหรับมอเตอร์ B

Specification

- Dual H bridge Drive Chip : L298N
- แรงดันสัญญาณลอจิก : 5V Drive voltage: 5V-35V
- กระแสของสัญญาณลอจิก : 0-36mA
- กระแสขับมอเตอร์ : สูงสุดที่ 2A (เมื่อใช้มอเตอร์เดียว)
- กำลังไฟฟ้าสูงสุด : 25W
- ขนาด : 43 x 43 x 26 มิลลิเมตร
- น้ำหนัก : 26 กรัม

*มี Power Supply 5V ในตัว สามารถจ่ายไฟออกจากช่อง 5V (เพื่อจ่ายให้บอร์ด Arduino) ได้ เมื่อต่อไฟเลี้ยงเข้าที่ช่อง 12V

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Power Supply	50	V
V_{SS}	Logic Supply Voltage	7	V
V_I,V_{en}	Input and Enable Voltage	–0.3 to 7	V
I_O	Peak Output Current (each Channel) – Non Repetitive (t = 100µs) –Repetitive (80% on –20% off; t_{on} = 10ms) –DC Operation	 3 2.5 2	 A A A
V_{sens}	Sensing Voltage	–1 to 2.3	V
P_{tot}	Total Power Dissipation (T_{case} = 75°C)	25	W
T_{op}	Junction Operating Temperature	–25 to 130	°C
T_{stg}, T_j	Storage and Junction Temperature	–40 to 150	°C

รูปที่ 2-4 คุณสมบัติที่รับได้ของ L298N

2.1.3 Real Time Clock DS3231

DS3231 module เป็นโมดูนาฬิกาแบบเวลาจริง RTC(Real Time Clock) ที่มีความถูกต้องแม่นยำสูงเพราะข้างในมีวงจรวัดอุณหภูมิ เพื่อนำอุณหภูมิจากสภาพแวดล้อมมาคำนวณชดเชยความถี่ของ Crystal ที่ถูกรบกวนจากอุณหภูมิภายนอก มาพร้อมแบตเตอร์รี่ ใช้งานได้แม้ไม่มีแหล่งจ่ายไฟจากภายนอก สามารถตั้งค่า วัน เวลา ได้อย่างง่าย มี Library มาพร้อมใช้งาน สามารถเลือกแสดงผลเวลาแบบ 24 ชั่วโมง หรือแบบ 12 ชั่วโมงก็ได้ นอกจากจะแสดงวันและเวลาได้อย่างแม่นยำแล้ว โมดูลนี้ยังสามารถ แสดงอุณหภูมิภายนอกได้ เป็นเหมือนนาฬิกาดิจิตอลที่บอกอุณหภูมิได้ด้วย

ข้อมูลเชิงเทคนิคที่สำคัญของไอซี DS3231

- ใช้แรงดันไฟเลี้ยง (VCC) ในช่วง +2.5V .. +5.5V (+3.3V typ.)
- ใช้แบตเตอรี่สำรองได้ แรงดันในช่วง (VBAT) +2.5V .. +5.5V (+3V typ.)
- ใช้พลังงานต่ำ (Low-Power Consumption) ดังนั้นเมื่อปิดแรงดันไฟเลี้ยง VCC
- สามารถทำงานต่อเนื่องได้โดยใช้แรงดันไฟเลี้ยง VBAT ได้โดยอัตโนมัติ
- ใช้ตัวถังแบบ SO (Small Outline) จำนวน 16 ขา
- เชื่อมต่อแบบบัส I2C (สัญญาณ SDA และ SCL) และใช้ความเร็วได้ถึง 400kHz
- ภายในมีวงจรสร้างสัญญาณ clock (crystal oscillator) ความถี่ 32kHz
- มีความแม่นยำ (Accuracy) ±2ppm ในช่วงอุณหภูมิ 0°C..+40°C และ ±3.5ppm สำหรับ -40°C..+85°C
- สามารถตั้งค่าการแจ้งเตือนหรือ Alarm เลือกได้จาก 2 ชุด และสร้างสัญญาณ Interrupt
- สามารถเลือกสร้างสัญญาณเอาต์พุตได้ (Programmable Square-Wave Output) ที่ขา #INT/SQW
- สามารถวัดค่าอุณหภูมิได้ ให้ข้อมูล Digital แบบ 10 บิต (2's complement) ความละเอียด 0.25°C
- แต่มีความแม่นยำ ±3°C

ในการเขียนโปรแกรมเพื่อสื่อสารข้อมูลกับไอซี DS3231 จะต้องส่งผ่านบัส I2C และเขียนหรืออ่านข้อมูลจากรีจิสเตอร์ที่อยู่ภายใน (อ้างอิงจาก datasheet)

ADDRESS	BIT 7 MSB	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0 LSB	FUNCTION	RANGE
00H	0	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Hours	1–12 + $\overline{AM}$/PM 00–23
03H	0	0	0	0	0	Day			Day	1–7
04H	0	0	10 Date		Date				Date	00–31
05H	Century	0	0	10 Month	Month				Month/ Century	01–12 + Century
06H	10 Year				Year				Year	00–99
07H	A1M1	10 Seconds			Seconds				Alarm 1 Seconds	00–59
08H	A1M2	10 Minutes			Minutes				Alarm 1 Minutes	00–59
09H	A1M3	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Alarm 1 Hours	1–12 + $\overline{AM}$/PM 00–23
0AH	A1M4	DY/$\overline{DT}$	10 Date		Day				Alarm 1 Day	1–7
					Date				Alarm 1 Date	1–31
0BH	A2M2	10 Minutes			Minutes				Alarm 2 Minutes	00–59
0CH	A2M3	12/$\overline{24}$	$\overline{AM}$/PM 10 Hour	10 Hour	Hour				Alarm 2 Hours	1–12 + $\overline{AM}$/PM 00–23
0DH	A2M4	DY/$\overline{DT}$	10 Date		Day				Alarm 2 Day	1–7
					Date				Alarm 2 Date	1–31
0EH	$\overline{EOSC}$	BBSQW	CONV	RS2	RS1	INTCN	A2IE	A1IE	Control	—
0FH	OSF	0	0	0	EN32kHz	BSY	A2F	A1F	Control/Status	—
10H	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	Aging Offset	—
11H	SIGN	DATA	DATA	DATA	DATA	DATA	DATA	DATA	MSB of Temp	—
12H	DATA	DATA	0	0	0	0	0	0	LSB of Temp	—

Note: *Unless otherwise specified, the registers' state is not defined when power is first applied.*

รูปที่ 2-5 ตารางแสดงรายการรีจิสเตอร์ที่เกี่ยวข้องกับการเก็บข้อมูลวันและเวลา รวมถึงการตั้ง alarm

ขออธิบายเป็นตัวอย่างดังนี้

- การเก็บข้อมูลเช่น วินาที (Seconds) ที่มีค่าอยู่ในช่วง 0..59 ในรีจิสเตอร์ที่ตำแหน่ง 00h จะเก็บโดยแบ่งเป็นสองส่วนๆละ 4 บิต แบบ BCD เช่น ถ้าวินาทีเท่ากับ 39 จะเก็บค่าเป็น 0011 1001 (เลขฐานสอง)
- การเก็บข้อมูลสำหรับชั่วโมง (Hour) จะมีบิต 12/24 (12-hour or 24-hour mode selection) เพื่อเลือกว่าจะเป็นในรูปแบบใดระหว่าง 12 ชั่วโมง (0..11) หรือ 24 ชั่วโมง (0..23) ถ้าเก็บแบบ 12 ชั่วโมง จะมีบิต AM/PM (ก่อนหรือหลังเที่ยงวัน) แต่ถ้าเก็บค่าแบบ 24 ชั่วโมง จะไม่มีบิต AM/PM เป็นต้น
- รีจิสเตอร์ที่ตำแหน่ง 03h จะเป็นค่าวันของสัปดาห์ (day of week) ซึ่งมีค่าอยู่ระหว่าง 1 ถึง 7 โดยที่ วันอาทิตย์ (Sunday) = 1, วันอังคาร (Tuesday) = 2, เป็นอย่างนี้ไปจนถึงวันเสาร์ ในขณะที่รีจิสเตอร์ที่ตำแหน่ง 04h จะเป็นค่าวันของเดือน (day of month) ซึ่งมีค่าอยู่ระหว่าง 1 ถึง 31
- การตั้งเวลาแจ้งเตือนหรือเวลาปลุก (Alarm) มี 2 ชุด (Alarm 1 และ Alarm 2) Alarm 1 จะตั้งเวลาแจ้งเตือนได้ในระดับวินาที แต่ Alarm 2 จะได้ในระดับนาที และการตั้งเวลาแจ้งเตือน จะต้องระบุค่าเวลาในอนาคต ลงในรีจิสเตอร์ที่เกี่ยวข้อง (ตามที่อยู่ 07h..0Ah สำหรับ Alarm 1 และ 0Bh..0Dh สำหรับ Alarm 2) เพื่อใช้ในการเปรียบเทียบ ถ้าตรงกัน จะสร้าง Alarm แจ้งเตือนได้
- การตั้งเวลาแจ้งเตือน จะต้องเซตบิต A1E และ A2E (Alarm Enable) ในรีจิสเตอร์ Control สำหรับ Alarm 1 และ Alarm 2 ที่เกี่ยวข้อง เมื่อเกิด Alarm ไอซี DS3231 จะสร้างสัญญาณ Interrupt ที่ขา #INT/SQW (ทำงานแบบ Active-low) และบิต A1F หรือ A2F จะถูกเซตเป็น 1 ในรีจิสเตอร์ Status เมื่อเกิด Alarm 1 หรือ Alarm 2
- ถ้าตั้งเวลาแจ้งเตือนเป็นระยะๆ และขา #INT/SQW เป็นเอาต์พุต เพื่อสร้างสัญญาณปลุกวงจรหรือระบบอื่น เช่น ไมโครคอนโทรลเลอร์ ก็ช่วยในการประหยัดการใช้พลังงานได้ อย่างในกรณีที่ใช้แบตเตอรี่เป็นแหล่งพลังงาน
- ถ้าไม่ใช้ Alarm ก็สามารถใช้ขา #INT/SQW สร้างสัญญาณเอาต์พุตรูปคลื่นสี่เหลี่ยมได้ (Square Wave) โดยโปรแกรมเลือกความถี่ได้จาก 4 ค่า คือ 1Hz, 1000Hz, 4096Hz, 8192Hz

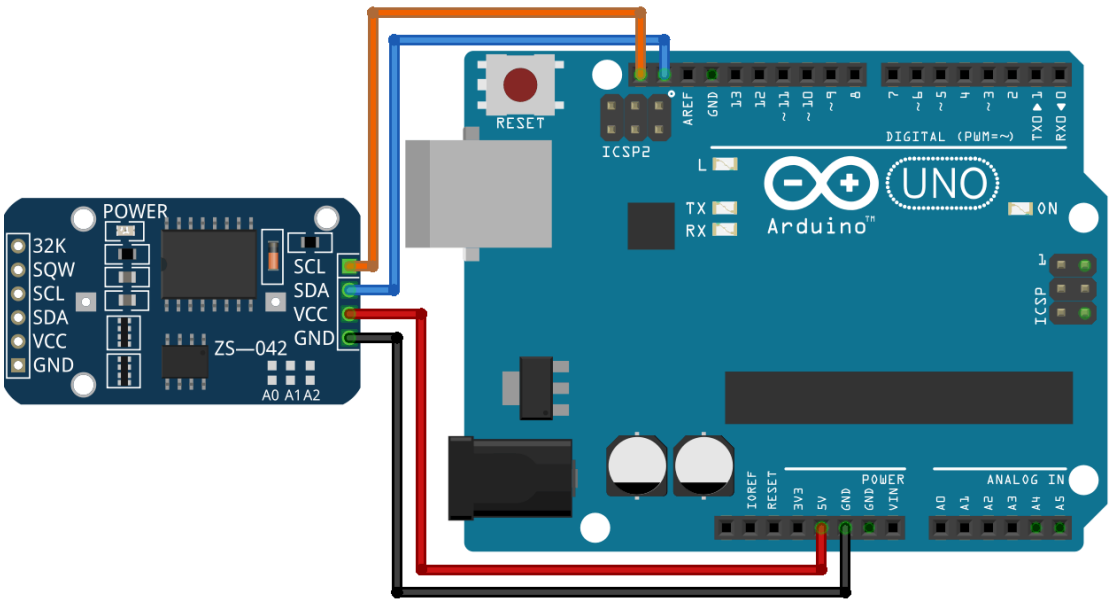
DY/$\overline{DT}$	ALARM 1 REGISTER MASK BITS (BIT 7)				ALARM RATE
	A1M4	A1M3	A1M2	A1M1	
X	1	1	1	1	Alarm once per second
X	1	1	1	0	Alarm when seconds match
X	1	1	0	0	Alarm when minutes and seconds match
X	1	0	0	0	Alarm when hours, minutes, and seconds match
0	0	0	0	0	Alarm when date, hours, minutes, and seconds match
1	0	0	0	0	Alarm when day, hours, minutes, and seconds match

DY/$\overline{DT}$	ALARM 2 REGISTER MASK BITS (BIT 7)			ALARM RATE
	A2M4	A2M3	A2M2	
X	1	1	1	Alarm once per minute (00 seconds of every minute)
X	1	1	0	Alarm when minutes match
X	1	0	0	Alarm when hours and minutes match
0	0	0	0	Alarm when date, hours, and minutes match
1	0	0	0	Alarm when day, hours, and minutes match

รูปที่ 2-6 ตารางแสดงการกำหนดค่าบิตในรีจิสเตอร์ที่เกี่ยวข้องกับการตั้งเวลาแจ้งเตือน (Alarm)

รูปที่ 2-8 RTC DS3231 ด้านแผงวงจร

รูปที่ 2-7 RTC DS3231 ด้านที่ใส่ถ่านนาฬิกา



รูปที่ 2-9 แผนภาพการต่อวงจร RTC DS3231 เข้ากับ Arduino

2.1.4 Servo Motor MG90s

เป็นมอเตอร์ที่มีการควบคุมการเคลื่อนที่ของมัน (State) ไม่ว่าจะเป็นระยะ ความเร็ว มุมการหมุน โดยใช้การควบคุมแบบป้อนกลับ (Feedback control) ซึ่งข้อแตกต่างนี้ชัดเจนระหว่าง Servo motor กับ Servo motor

Stepper motor จะไม่มีการป้อนกลับ แต่ควบคุมการเคลื่อนที่แบบระบบเปิด โดยการส่งสัญญาณไปที่ตัว Stator ให้เคลื่อนที่ไปเป็น step ที่ตายตัว เช่น ทีละ 1.5 องศา เป็นต้น (ความละเอียดของการเคลื่อนที่ขึ้นกับคุณสมบัติของ Stepper motor และเทคนิคการควบคุม) ในขณะที่ Servo motor ต้องการสัญญาณป้อนกลับเพื่อใช้ในการประเมินตำแหน่ง หรือ ความเร็ว หรือ State อื่นๆ เพื่อไปประมวลการเคลื่อนไหวที่เหมาะสมเทียบกับตำแหน่งที่ผู้ใช้ระบุ โดยอาจจะใช้การควบคุมแบบปิดได้หลายๆ แบบ แล้วแต่ผู้ผลิต

รูปที่ 2-10 แผนภาพ Servo MG90s

รูปที่ 2-11 Servo MG90s

Servo motor ที่มีขายในปัจจุบันก็มีหลายแบบมากๆ เช่น แบบที่ประกอบด้วยมอเตอร์กระแสตรง (DC motor) แบบที่ใช้มอเตอร์กระแสสลับ (AC motor) ก็มี แบบมอเตอร์ไร้แปรงถ่านก็มี แถมมีหลากหลายวิธีการควบคุมด้วย

ข้อดีของ Servo motor

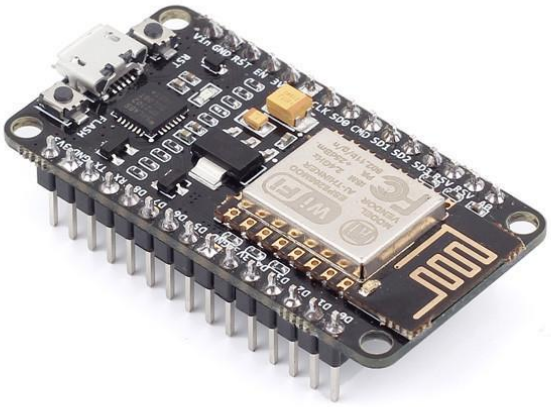
- สามารถให้ค่าทอร์กที่สูง
- สามารถเคลื่อนที่ความเร็วสูง
- ใช้งานกับการควบคุมความเร็วได้ดี
- มีหลากหลายขนาดให้เลือก (มากกว่า Stepper motor)
- เงียบ

ข้อเสีย Servo motor

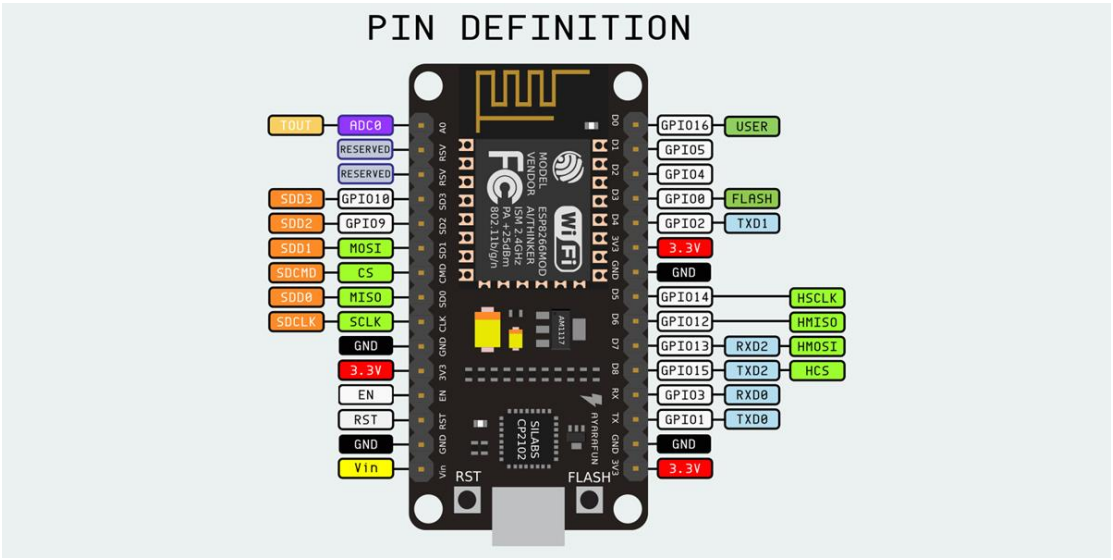
- แพงว่า Stepper motor
- ไม่สามารถทำงานโดยการควบคุมแบบเปิด
- ต้องมีการปรับค่าในการควบคุม
- ในกรณีที่ใช้ DC motor ต้องมีการบำรุงรักษา เนื่องจากแปรงถ่านอาจสึก

2.1.5 NodeMCU

NodeMCU คือ แพลตฟอร์มหนึ่งที่ใช้ช่วยในการสร้าง Project Internet of Things (IoT) ที่ประกอบ ไปด้วย Development Kit (ตัวบอร์ด) และ Firmware (Software บนบอร์ด) ที่เป็น open source สามารถเขียน โปรแกรมด้วยภาษา Lau ได้ ทำให้ใช้งานได้ง่ายขึ้น มาพร้อมกับโมดูล WiFi (ESP8266) ซึ่งเป็นหัวใจสำคัญในการใช้เชื่อมต่อกับอินเตอร์เน็ตนั่นเอง ตัวโมดูล ESP8266 นั้นมีอยู่ด้วยกันหลายรุ่นมาก ตั้งแต่เวอร์ชันแรก ที่เป็น ESP-01 ไล่ไปเรื่อยๆจนปัจจุบันมีถึง ESP-12 แล้ว และที่ฝังอยู่ใน NodeMCU version แรกนั้นก็เป็น ESP-12 แต่ใน version2 นั้นจะใช้เป็น ESP-12E แทน ซึ่งการใช้งานโดยรวมก็ไม่แตกต่างกันมากนัก NodeMCU นั้นมีลักษณะคล้ายกับ Arduino ตรงที่มีพอร์ต Input Output built inมาในตัว สามารถเขียน โปรแกรมคอนโทรลอุปกรณ์ I/O ได้โดยไม่ต้องผ่านอุปกรณ์อื่นๆ และเมื่อไม่นานมานี้ก็มีนักพัฒนาที่ สามารถทา ให้ Arduino IDE ใช้งานร่วมกับ Node MCU ได้จึงทำ ให้ใช้ภาษา C/C++ ในการเขียนโปรแกรม ได้ ทำให้ราสามารถใช้งานมันได้หลากหลายมากยิ่งขึ้น NodeMCUตัวนี้สามารถทำอะไรได้หลายอย่างมาก โดยเฉพาะเรื่องที่เกี่ยวข้องกับ IoT ไม่ว่าจะเป็นการทำ Web Server ขนาดเล็ก การควบคุมการเปิดปิดไฟผ่าน WiFi และอื่นๆอีกมากมาย



รูปที่ 2-13 NodeMCU ESP-12E



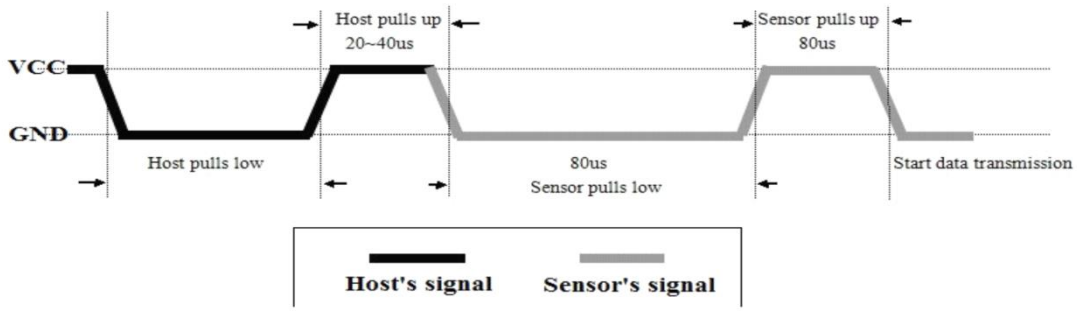
รูปที่ 2-12 ขา Pin ของ NodeMCU

- ชุดพัฒนานี้ based on module WiFi ที่ชื่อ ESP8266
- มี GPIO PWM, I2C, 1-Wire และ ADC รวมมาอยู่บนบอร์ดเดียว
- มี USB-TTL มาในตัว ไม่ต้องซื้อแยกเหมือนกับการใช้ ESP8266 ปกติ ทำให้ใช้งานได้สะดวกขึ้น
- มีขา GPIO 10 ขา ทุกๆขาสามารถเป็น PWM, I2C และ 1-wire ได้
- มี PCB antenna สำหรับรับส่งสัญญาณไร้สาย
- ใช้ Connector แบบ micro-USB สำหรับจ่ายแรงดันไฟเลี้ยงหรือเท่ากับ +5V และสำหรับดาวน์โหลด Frameware

2.1.6 Sensor ตรวจจับความชื้น และ อุณหภูมิ

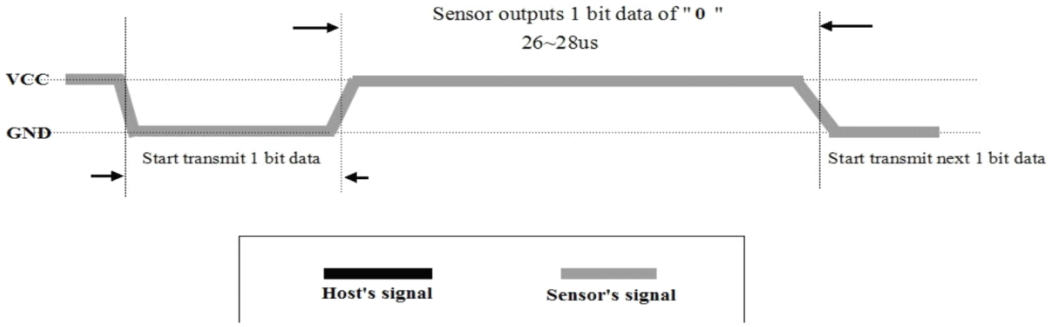
เป็น sensor ที่ใช้สำหรับตรวจจับอุณหภูมิภายในกรงสัตว์เลี้ยงเพื่อเช็คว่าสัตว์เลี้ยงได้รับอุณหภูมิที่ถูกต้องและสามารถนำมาดัดแปลงเพื่อเป็นตัวตรวจจับว่ามีสิ่งสกปรกอยู่ภายในกรงไหม และถ้ามีความชื้นถึงจุดก็จะทำการแจ้งเตือนว่าควรทำความสะอาดภายในที่อยู่อาศัยได้

โดยจะเลือกใช้เป็นตัว DHT22 เนื่องจากสามารถนำมาใช้และประยุกต์เข้ากับบอร์ด Arduino ได้ โดยข้อมูลของตัว DHT22 เป็นดังนี้

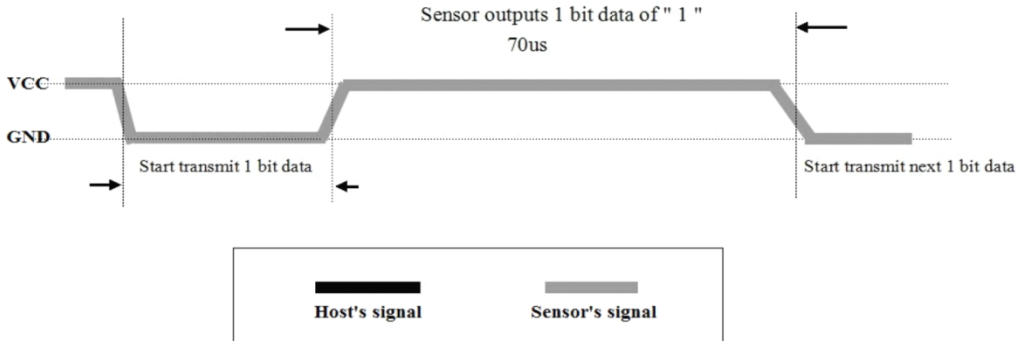


รูปที่ 2-14 กราฟการทำงานของ DHT22 1

เริ่มจาก MCU จะส่งสัญญาณ pull down voltage ไปยัง DHT11/22 โดย ถ้าเป็น DHT 11 จะใช้เวลาส่ง down voltage อย่างต่ำ 18 ms แต่ถ้าเป็น DHT22 จะใช้เวลาอย่างต่ำ 1 ms และ MCU จะ pull up voltage เพื่อรอการตอบสนองจาก DHT ประมาณ 20-40 us หลังจากนั้น DHT จะส่งสัญญาณ pull down voltage เวลา 80 us เป็นการตอบสนองไปยัง MCU แล้ว DHT ก็จะ pull up voltage เพื่อเตรียมส่งข้อมูล โดยในการส่งข้อมูลแต่ละบิต DHT จะมีการ pull down voltage 50 us



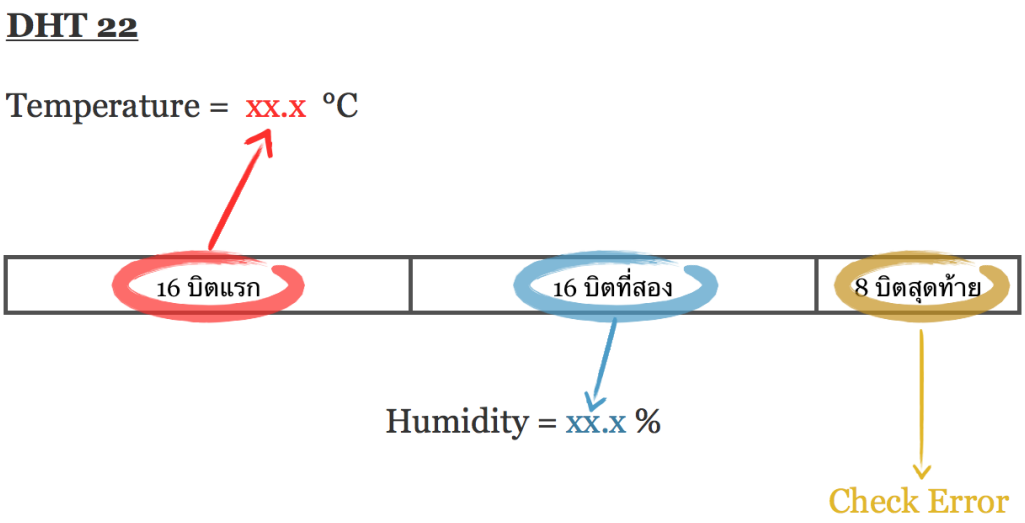
รูปที่ 2-15 กราฟการทำงานของ DHT22 2



รูปที่ 2-16 กราฟการทำงานของ DHT22 3

หลังจาก DHT มีการ pull down voltage 50 us เพื่อเป็นการบอก MCU ว่าจะส่งข้อมูล 1 บิต โดยการส่งบิตค่า “0” DHT จะทำการส่งสัญญาณ pull up voltage 26-28 us และ ส่งบิตค่า “1” DHT จะทำการส่งสัญญาณ pull up voltage 70 us

โดยการส่งข้อมูลของ DHT11 คือ จะส่งทั้งหมด 40 บิต โดยจะแบ่งเป็น 5 ส่วน ส่วนละ 8 บิต ซึ่ง 8 บิตแรกจะเป็นค่าหน้าทศนิยมของอุณหภูมิ, 8บิตที่สองเป็นค่าหลังทศนิยมของอุณหภูมิ, 8บิตที่สามจะเป็นค่าหน้าทศนิยมของความชื้น, 8บิตที่สี่เป็นค่าหลังทศนิยมของความชื้น และ 8บิตสุดท้ายคือเป็นค่าที่ตรวจสอบว่าข้อมูล error หรือไม่

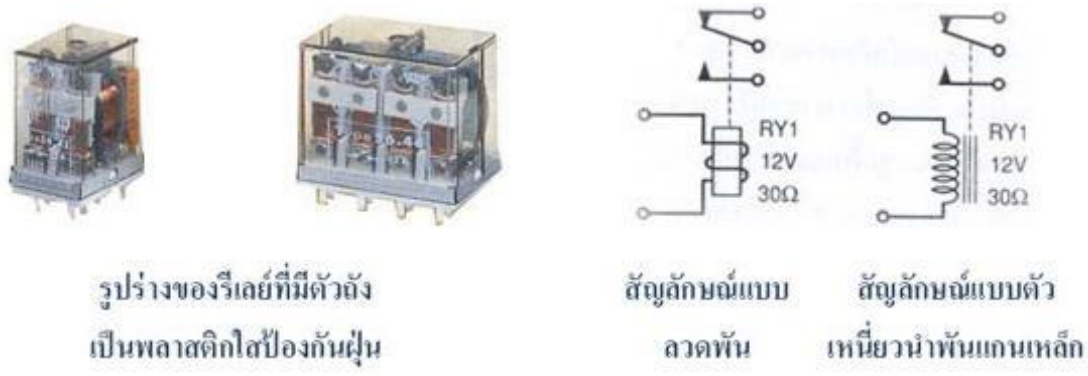


รูปที่ 2-17 การอ่าค่าของ DHT22

การส่งข้อมูลของ DHT22 คือ จะส่งทั้งหมด 40 บิต โดยจะแบ่งเป็น 3 ส่วน สองส่วนแรกส่วนละ 16 บิต และส่วนสุดท้าย 8 บิต ซึ่ง 16บิตแรกและ 16บิตที่สอง หมายถึงค่าอุณหภูมิและค่าความชื้นตามลำดับ ที่รวมทั้งค่าหน้าและหลังทศนิยม โดย ตัวเลขหลักหน่วยจะหมายถึงตัวหลังทศนิยม และ 8บิตสุดท้ายคือเป็นค่าที่ตรวจสอบว่าข้อมูล error หรือไม่

2.1.7 รีเลย์ (Relay)

เป็นอุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าให้เป็นพลังงานแม่เหล็ก เพื่อใช้ในการดึงดูดหน้าสัมผัสของคอนแทคให้เปลี่ยนสภาวะ โดยการป้อนกระแสไฟฟ้าให้กับขดลวด เพื่อทำการปิดหรือเปิดหน้าสัมผัสคล้ายกับสวิตช์อิเล็กทรอนิกส์ ซึ่งเราสามารถนำรีเลย์ไปประยุกต์ใช้ ในการควบคุมวงจรต่าง ๆ ในงานช่างอิเล็กทรอนิกส์มากมาย



รูปที่ 2-18 ลักษณะของ Relay

ส่วนประกอบของรีเลย์

1. ส่วนของขดลวด (coil) เหนี่ยวนำกระแสต่ำ ทำหน้าที่สร้างสนามแม่เหล็กไฟฟ้าให้แกนโลหะไปกระทุ้งให้หน้าสัมผัสต่อกัน ทำงานโดยการรับแรงดันจากภายนอกต่อคร่อมที่ขดลวดเหนี่ยวนำนี้ เมื่อขดลวดได้รับแรงดัน(ค่าแรงดันที่รีเลย์ต้องการขึ้นกับชนิดและรุ่นตามที่ผู้ผลิตกำหนด) จะเกิดสนามแม่เหล็กไฟฟ้าทำให้แกนโลหะด้านในไปกระทุ้งให้แผ่นหน้าสัมผัสต่อกัน

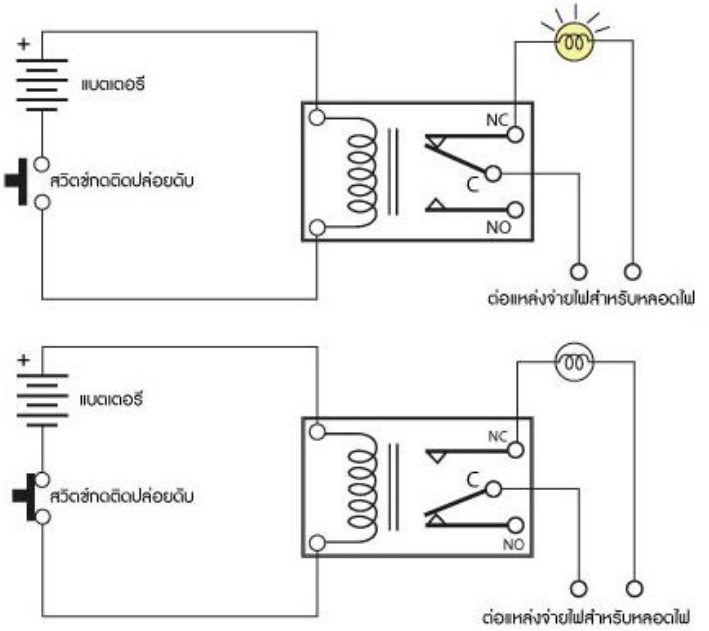
2. ส่วนของหน้าสัมผัส (contact) ทำหน้าที่เหมือนสวิตช์จ่ายกระแสไฟให้กับอุปกรณ์ที่เราต้องการ

จุดต่อใช้งานมาตรฐาน

จุดต่อ NC ย่อมาจาก normal close หมายความว่าปกติปิด หรือ หากยังไม่จ่ายไฟให้ขดลวดเหนี่ยวนำหน้าสัมผัสจะติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการให้ทำงานตลอดเวลาเช่น

จุดต่อ NO ย่อมาจาก normal open หมายความว่าปกติเปิด หรือหากยังไม่จ่ายไฟให้ขดลวดเหนี่ยวนำหน้าสัมผัสจะไม่ติดกัน โดยทั่วไปเรามักต่อจุดนี้เข้ากับอุปกรณ์หรือเครื่องใช้ไฟฟ้าที่ต้องการควบคุมการเปิดปิดเช่นโคมไฟสนามเหนือหน้าบ้าน

จุดต่อ C ย่อมากจาก common คือจุดร่วมที่ต่อมาจากแหล่งจ่ายไฟ



รูปที่ 2-19 ภายในการทำงานของ Relay

ข้อคำถึงในการใช้งานรีเลย์ทั่วไป

1. แรงดันใช้งาน หรือแรงดันที่ทำให้รีเลย์ทำงานได้ หากเราดูที่ตัวรีเลย์จะระบุค่า แรงดันใช้งานไว้ (หากใช้ในงานอิเล็กทรอนิกส์ ส่วนมากจะใช้แรงดันกระแสตรงในการใช้งาน) เช่น 12VDC คือต้องใช้แรงดันที่ 12 VDC เท่านั้นหากใช้มากกว่านี้ ขดลวดภายใน ตัวรีเลย์อาจจะขาดได้ หรือหากใช้แรงดันต่ำกว่ามาก รีเลย์จะไม่ทำงาน ส่วนในการต่อวงจรนั้นสามารถต่อขั้วใด เพราะตัวรีเลย์ จะไม่ระบุขั้วต่อไว้ (นอกจากชนิดพิเศษ)

2. การใช้งานกระแสผ่านหน้าสัมผัส ซึ่งที่ตัวรีเลย์จะระบุไว้ เช่น 10A 220AC คือ หน้าสัมผัสของรีเลย์นั้นสามารถทนกระแสได้ 10 แอมแปร์ที่ 220VAC แต่การใช้ก็ควรจะใช้งานที่ระดับกระแสต่ำกว่านี้จะเป็นการดีกว่า เพราะถ้ากระแสมากหน้าสัมผัส ของรีเลย์จะละลายเสียหายได

3. จำนานหน้าสัมผัสการใช้งาน ควรดูว่ารีเลย์นั้นมีหน้าสัมผัสให้ใช้งานกี่อัน และมีขั้วคอมมอนด้วยหรือไม่

2.1.8 Contact Non-Water

เซนเซอร์สำหรับวัดระดับน้ำ แบบไร้สัมผัส นำไปวางในจุดที่ต้องการวัด เช่นวางตรงระดับของถังน้ำที่ต้องการวัด เมื่อน้ำสูงถึงระดับที่เซนเซอร์อยู่ ก็จะตรวจจับได้ ไฟสีแดงบนเซนเซอร์ติด ให้สัญญาณเอาต์พุตออกมาเป็นค่า 0 หรือ 1 สามารถวัดทะลุถังน้ำได้หนาสูงสุดถึง 13mm ใช้ไฟเลี้ยงได้ในช่วงกว้าง 5-24V ใช้งานได้โดยตรง ไม่ต้องมีบอร์ดต่อเพิ่ม รองรับ Arduino , ESP8266 , ESP32 และ PLC ลักษณะจะเป็นไปตามรูปที่ 2 – 20 และรูปที่ 2 - 21

รูปที่ 2-20 ลักษณะ sensor

รูปที่ 2-21 ขนาดและความยาว

Specifications:

- Model: XKC-Y25-V
- Output: NPN
- Input Voltage (InVCC): DC 5-24V
- Current: 5mA
- Output Voltage (high level): In VCC
- Output Voltage (low level): 0V
- Output Current: 1-100mA
- Response Time: 500mS
- Operating Temperature: 0 - 105℃
- Induction Thickness (sensitivity): 0 - 13 mm
- Communication: RS485
- Humidity: 5% - 100%
- Material: ABS
- Ingress Protection: IP67

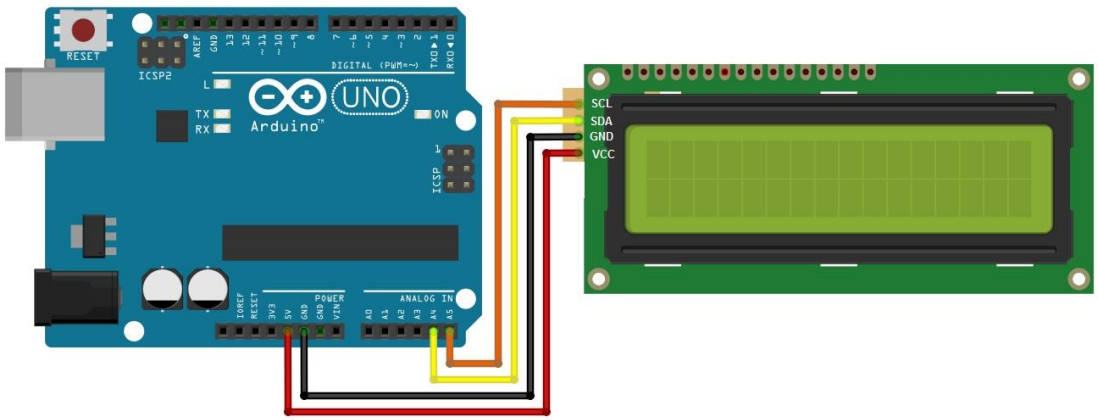
รูปที่ 2-22 spec และ สายสัญญาณ

2.1.9 LCD แบบ I2C

ถ้าเคยมีปัญหากับการต่อสายไฟหลายเส้น กับจอ LCD ทำให้สับสน หรือทำให้ขา Arduino ไม่พอใช้งาน การใช้โมดูล I2C LCD ตัวนี้เป็นตัวเลือกที่แนะนำ เพราะสามารถเชื่อมต่อกับหน้าจอ LCD รุ่น 16x2 หรือรุ่น 20x4 โดยใช้สายไฟเพียง 2 เส้น สะดวก ง่าย ช่วยให้เหลือขา Arduino ไว้ใช้งานที่สำคัญอย่างอื่น ใช้งานง่ายมี library มาตรฐานพร้อมใช้งาน

ตัวนี้รวมเอา 1602 LCD(สีเหลือง) กับ I2C LCD มาให้พร้อมใช้งาน ช่วยให้การติดต่อ Arduino กับ LCD เป็นเรื่องง่าย โดยตัว LCD ที่เลือกใช้จะเป็นแบบจอสีเขียวดังรูปที่ 2 - 23 และการเชื่อมต่อ LCD เข้ากับ บอร์ด Arduino ดังรูปที่ 2 – 24

รูปที่ 2-23 ตัวอย่าง LCD



รูปที่ 2-24 ตัวอย่างการต่อ LCD เข้าบอร์ด Arduino

บทที่ 3 รายละเอียดการทำงาน

3.1 System Specification

- ระบบให้อาหารอัตโนมัติสามารถให้อาหารสัตว์เลี้ยงได้แม้ไม่อยู่ที่บ้าน โดยใช้ระบบตั้งเวลาเพื่อให้สามารถกำหนดการให้อาหารของตัวอุปกรณ์ได้
- ระบบปรับอุณหภูมิสามารถอ่านค่าอุณหภูมิและความชื้นภายในบ้านสัตว์เลี้ยงได้
- ระบบปรับอุณหภูมิสามารถทำการแสดงผลผ่านทางหน้าจอ LCD ที่ตัวเครื่องโดยจะมีการแสดงผลอุณหภูมิและความชื้น
- ระบบปรับอุณหภูมิสามารถสั่งให้พัดลมปรับอุณหภูมิทำงานได้เมื่ออุณหภูมิเกินที่กำหนดเพื่อทำให้อุณหภูมิอยู่ในที่พอเหมาะตลอดเวลา โดยอุณหภูมิที่ตั้งไว้เมื่ออุณหภูมิมากกว่า 30 องศาให้พัดลมทำงานเต็มกำลัง แต่ถ้าน้อยกว่า 30 และไม่เกิน 25 องศาให้พัดลมทำงาน 50 % แต่เมื่ออุณหภูมิน้อยกว่า 25 องศาก็ให้พัดลมหยุดทำงาน
- ระบบปรับอุณหภูมิสามารถอ่านค่าอุณหภูมิผ่านตัว website ได้ โดยจะมีการแสดงเป็นกราฟและค่าในตารางที่อ่านได้ต่างๆ รวมไปถึงสถานการณ์ทำงานของพัดลมปรับอุณหภูมิ

1. ระบบเครื่องให้น้ำอัตโนมัติจะทำการเติมน้ำภายในถาดสัตว์เลี้ยงเมื่อน้ำในถาดลดน้อยกว่าที่กำหนด

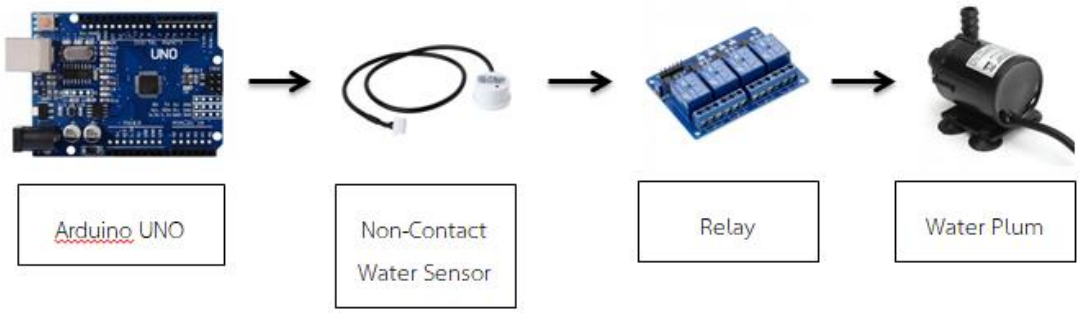
3.2 System Architecture

3.2.1 ระบบเครื่องให้อาหารอัตโนมัติ

รูปที่ 3-1 System Architecture ระบบให้อาหารอัตโนมัติ

ทำงานให้ Servo ที่ทำการติดตั้งถาดใส่อาหารได้ทำการหมุนไปทางขวาเป็นมุม 90องศา เพื่อเทอาหารลงไปในช่องอาหารที่เตรียมไว้ได้โดยผู้ใช้สามารถตั้งค่าเวลาในการเทถาดอาหารได้ตามเวลาจริง เพราะว่า ได้นำ RTC DS3231 ที่เป็นตัว Real Time Clock มานับเวลาให้เสมือนเวลาจริงเพื่อให้สามารถตั้งเวลาได้ตามที่ผู้ใช้ต้องการ โดยบอร์ด Arduino UNO R3 จะติด ESP8266 เข้าไปด้วยเพื่อให้สามารถเชื่อมต่อ Wifi และส่งข้อมูลให้กับตัว NodeMCU ที่จะใช้ส่งข้อมูลขึ้น website

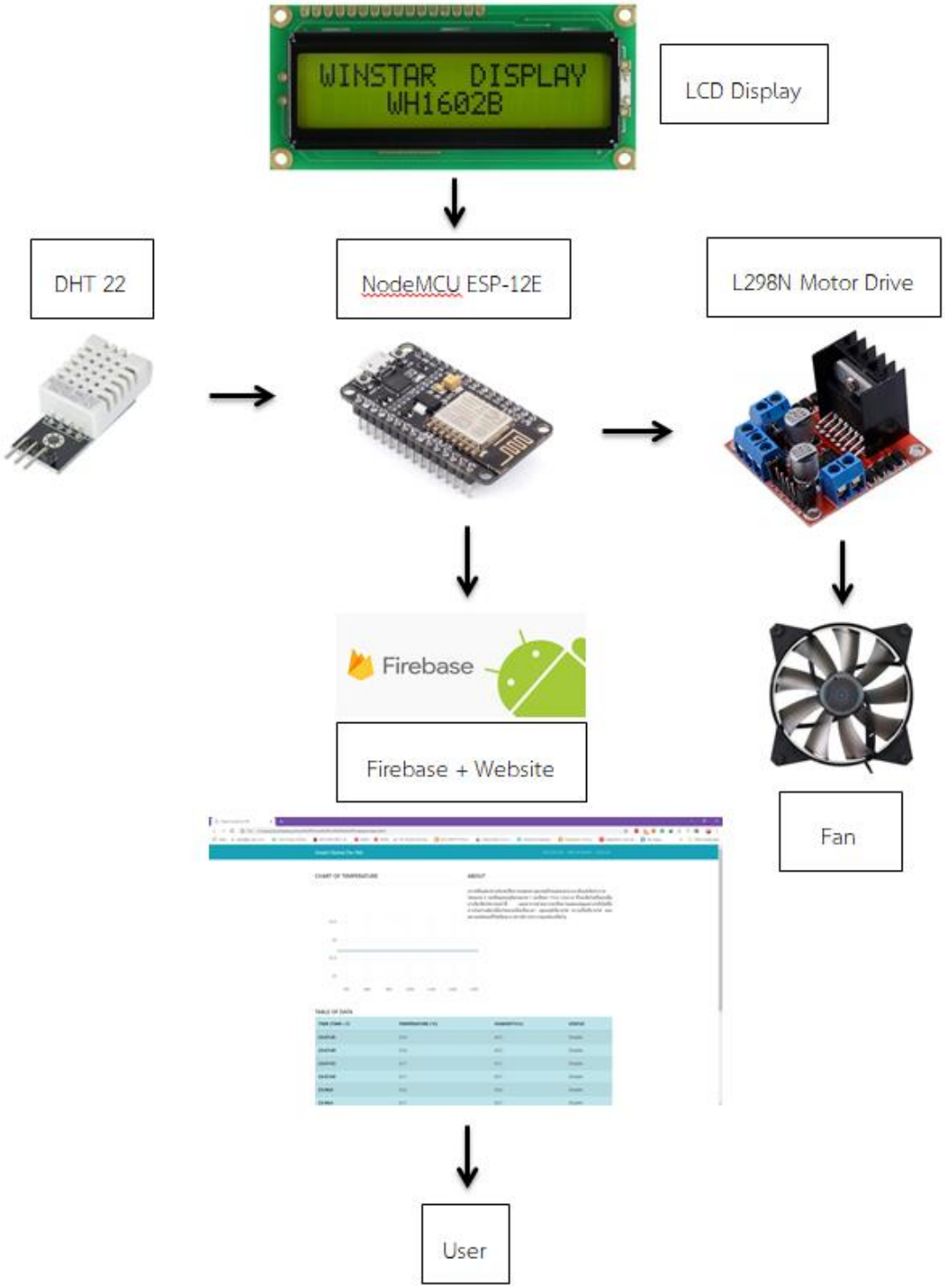
3.2.2 ระบบเครื่องให้น้ำอัตโนมัติ



รูปที่ 3-2 System Architecture ระบบให้น้ำอัตโนมัติ

ระบบเครื่องให้น้ำอัตโนมัติจะเป็นการเอา Arduino UNO R3 มาเป็นตัวควบคุมการทำงานโดยจะมีการใช้ Sensor ตัววัดระดับน้ำเป็นตัววัดว่าในถาดให้น้ำมีน้ำเหลือไหมถ้าไม่มีก็จะสั่งให้ปั้มน้ำมาใส่ถาดให้น้ำของสัตว์เลี้ยง

3.2.3 ระบบปรับอุณหภูมิภายในอัตโนมัติ

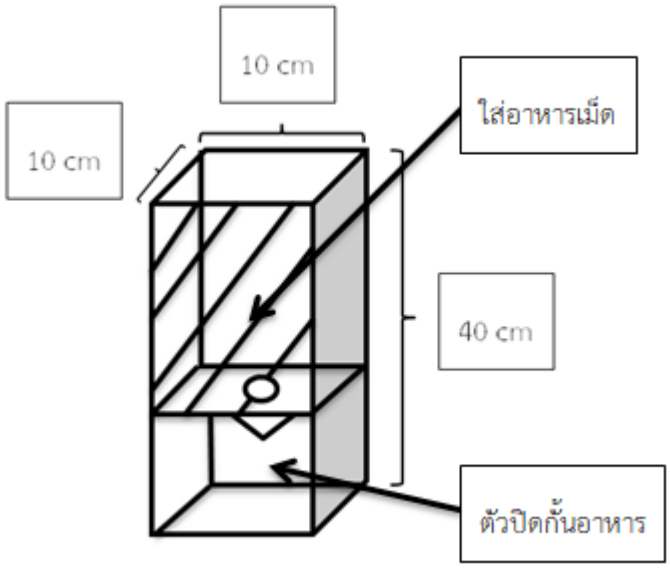


รูปที่ 3-3 System Architecture ระบบปรับอุณหภูมิภายในอัตโนมัติ

ระบบเครื่องปรับอุณหภูมิภายในบ้านสัตว์เลี้ยงอัตโนมัติจะใช้เป็น NodeMCU ESP-12E มาเป็นตัวกลางในการรับค่าจาก DHT22 และจะเป็นตัวสั่งการให้พัดลมทำงานเมื่อค่าอุณหภูมิที่อ่านได้จาก DHT22 นั้นมีค่ามากกว่าหรือเท่ากับ 30 องศาเซลเซียส จะเป็นการสั่งให้พัดลมทำงาน แต่ถ้าอุณหภูมิที่อ่านได้น้อยกว่า 25 องศาเซลเซียส ก็จะสั่งให้พัดลมหยุดทำงานโดยผ่านตัวขับ Motor L298N เพื่อให้พัดลมสามารถปรับอุณหภูมิได้อย่างเต็มที่ และให้แสดงค่าผ่านจอ LCD และทำการส่งค่าการทำงานต่างๆ และที่อ่านค่าได้ขึ้น Firebase แล้วจึงให้แสดงผ่านหน้า website ดังรูปที่ 3 – 3

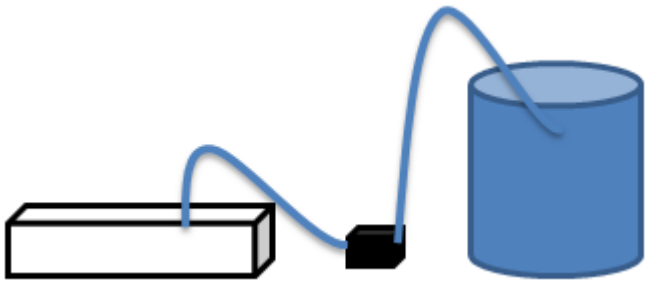
3.3 System Design

3.3.1 ระบบเครื่องให้อาหารอัตโนมัติ



รูปที่ 3-4 ภาพร่างระบบให้อาหารอัตโนมัติ

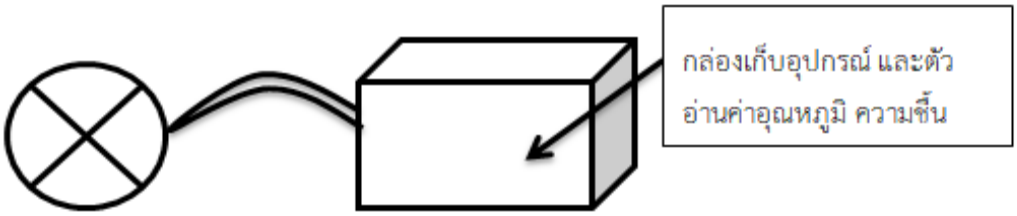
3.3.2 ระบบเครื่องให้น้ำอัตโนมัติ



รูปที่ 3-5 ภาพร่างระบบให้น้ำอัตโนมัติ

โดยระบบนี้จะอยู่ที่ฝั่งหนึ่งโดยจะตั้งตรงข้ามกับส่วนให้อาหารและสัตว์เลี้ยงเพื่อให้อากาศที่ได้รับการปรับอุณหภูมิโดยใช้พัดลมได้มีการถ่ายเทอากาศให้ไหลเวียนในบริเวณนั้นไม่ให้ร้อนเกินไป

3.3.3 ระบบปรับอุณหภูมิภายในอัตโนมัติ



รูปที่ 3-6 ภาพร่างระบบปรับอุณหภูมิภายในบ้านสัตว์เลี้ยง

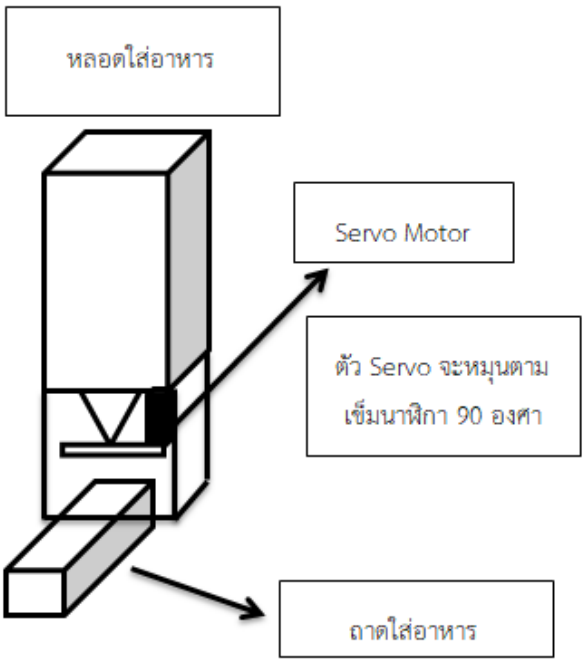
โดยระบบนี้จะอยู่ที่ฝั่งหนึ่งโดยจะตั้งตรงข้ามกับส่วนให้อาหารและสัตว์เลี้ยงเพื่อให้อากาศที่ได้รับการปรับอุณหภูมิโดยใช้พัดลมได้มีการถ่ายเทอากาศให้ไหลเวียนในบริเวณนั้นไม่ให้ร้อนเกินไป

3.4 System Implementation

3.4.1 ระบบให้อาหารสัตว์เลี้ยงอัตโนมัติ

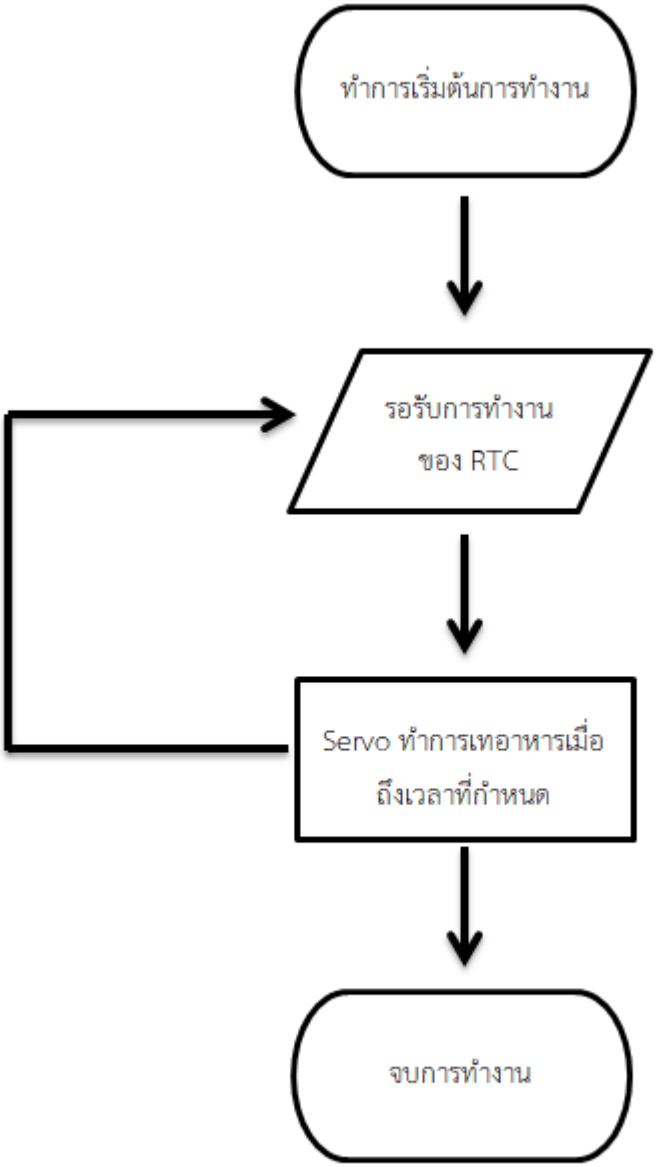
จะมีหลักการทำงานหลักๆอยู่ 2 อย่างได้แก่

1. ตัว Servo Motor ที่ใช้เป็นตัวควบคุมการหมุนของที่ใส่อาหารให้มีการให้อาหารอัตโนมัติใช้เป็นตัวหมุนเพื่อให้อาหารได้หมุนลงมายังถาดใส่อาหารของสัตว์เลี้ยง (อาหารเม็ด)



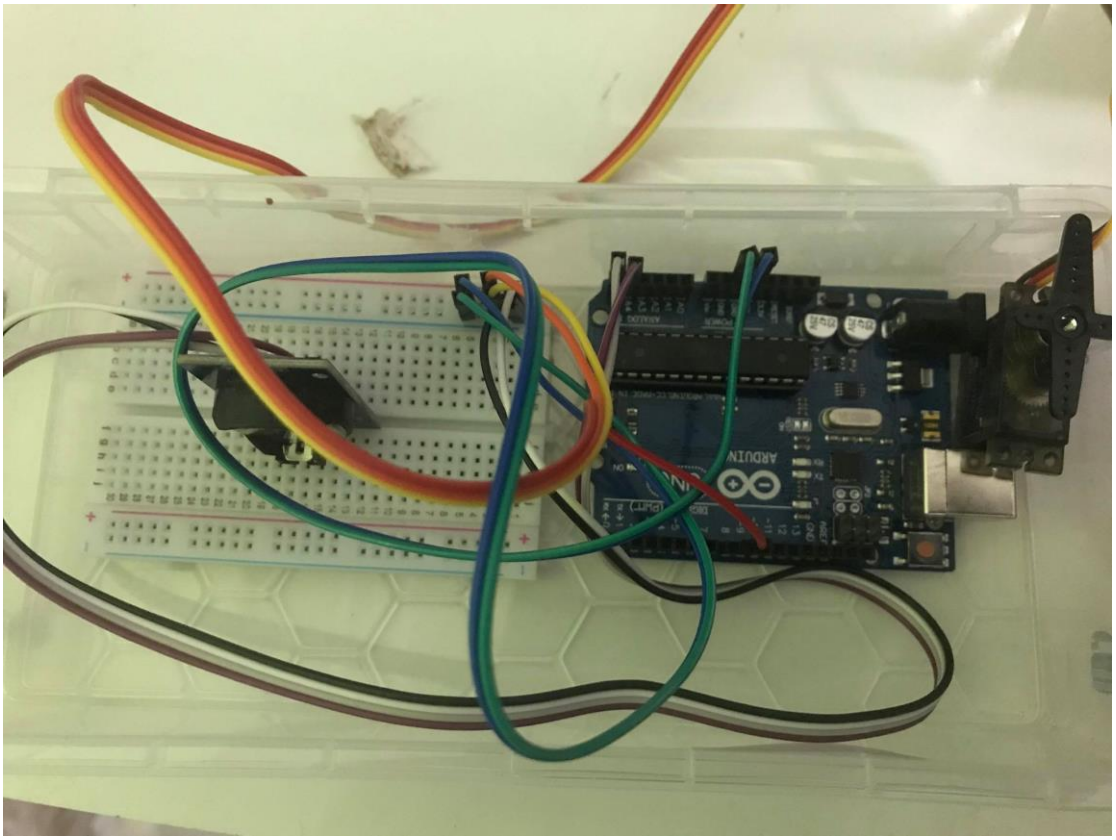
รูปที่ 3-7 เครื่องให้อาหารอัตโนมัติ

2. Flowchart การทำงานของระบบเครื่องให้อาหารอัตโนมัติ



รูปที่ 3-8 Flowchart ระบบเครื่องให้อาหารอัตโนมัติ

3. ตัวบอร์ด Arduino จะเป็นตัวป้อนคำสั่งการทำงานของที่ได้จากการเขียน Code ไปให้ตัว Servo และให้ตัว Real Time Clock (RTC) เป็นตัวกำหนดเวลาให้ Servo ทำงานเมื่อถึงเวลากำหนด โดยเราสามารถกำหนดเวลาได้ผ่านตัว Code โดยสามารถกำหนดเป็นหลักชั่วโมง และนาทีได้ การต่อตามรูปที่ 3 - 4



รูปที่ 3-9 วงจรระบบควบคุมให้อาหารอัตโนมัติ

ขา 9	ต่อเข้ากับขาสัญญาณของ Servo
ขา A4	ต่อเข้ากับขา SDA ของตัว RTC DS3231
ขา A5	ต่อเข้ากับขา SCL ของตัว RTC DS3231

ตารางที่ 3-1 ตารางบอกขาที่ใช้เชื่อมต่อระบบเครื่องให้อาหารอัตโนมัติ

4. ฟังก์ชันที่ใช้ในการทำงานมี 2 อย่างหลักๆ คือ Servo และ RTC DS3231.
    a. Servo จะมีการทำงานโดยการรับค่าจากตัว Code มาแล้วทำการหมุนองศาตามที่กำหนดไว้โดยตัวกำหนดจะอยู่ในเงื่อนไขการหมุนในฟังก์ชัน

```
 if (RTC.checkIfAlarm(1)) {
   Serial.println("Feed Food");
   feederOpen();
   delay(150);
   feederClose();  }
else
   myservo.write(145);
```

```
void feederClose() {
    myservo.write(145);
}
void feederOpen(){
    myservo.write(180);
}
```

จาก Code ข้างต้นจะเป็นตัวกำหนดการหมุนของตัว Servo ว่าให้ต้องหมุนกี่องศาโดยสามารถทำได้ตั้งแต่ 0 – 180 หรือถ้าต้องการทำให้ถึง 360 ก็สามารถทำได้โดยการแก้ไข Code ให้เหมาะสม

b. RTC DS3231 จะเป็นตัวกำหนดการตัวให้ Servo ต้องทำงานในเวลานั้นๆโดยตัว DS3231 จะทำการนับเวลาเทียบเท่าเวลาปัจจุบันไปเรื่อยๆเสมือนนาฬิกา โดยจะมีฟังก์ชันที่เมื่อเวลานับถึงที่กำหนดจะทำให้เข้าเงื่อนไขที่ตัว Servo จะทำงานต่อตามฟังก์ชันด้านบน

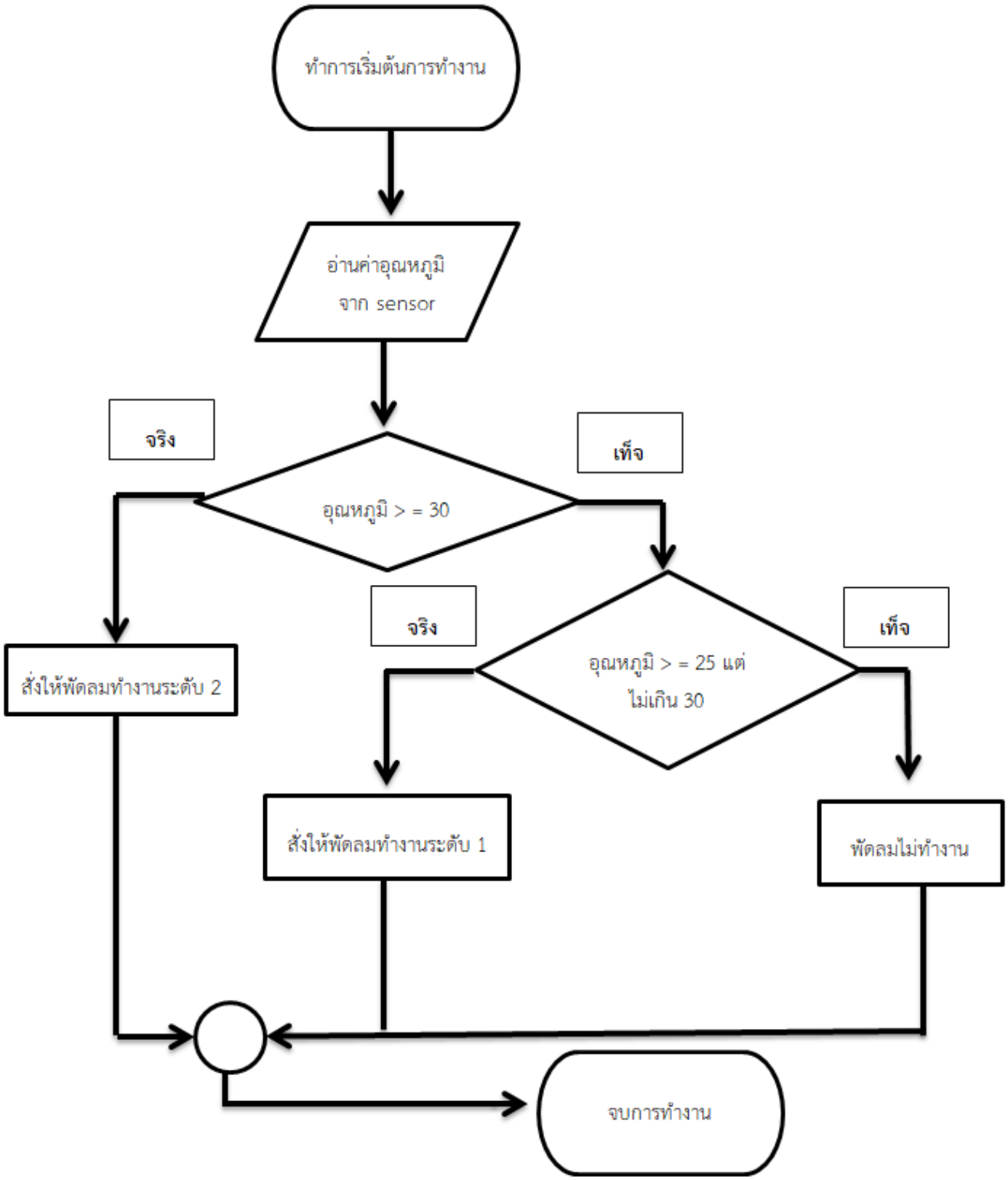
```
RTC.begin();

RTC.adjust(DateTime(__DATE__, __TIME__));

if (! RTC.isrunning()) {

..... }

DateTime now = RTC.now();

RTC.setAlarm1Simple(23, 9);

if (RTC.checkIfAlarm(1)) {

.......  }
```

จาก Code ข้างต้นจะมีส่วนที่ทำให้ฟังก์ชัน RTC ทำงานก็คือตัว RTC.begin คือจะเป็นการทำให้ตัวเวลาของ RTC นั้นเริ่มนับเวลาตามที่ตั้งค่าไว้ และให้ทำการอ่านค่าจะค่า DATE TIME ที่อยู่ใน Library ของ DS3231 เพื่อทำการรันค่าเวลาให้เท่ากับเวลาปัจจุบันที่สุด และมีฟังก์ชันเงื่อนไข โดยตัว RTC.setAlarm1Simple(23,9) จะเป็นตัวกำหนดให้เวลาไหนที่ต้องการให้ตั้งค่าเตือน โดยในฟังก์ชันจะเป็น 23.09 น. จะทำการแจ้งเตือน และจะทำการเข้าสู่ฟังก์ชัน RTC.checkAlarm จะมีค่าเป็น 1 เมื่อ RTC.setAlarm1Simple(23, 9) ทำงานและจะเข้าเงื่อนไขไปทำในฟังก์ชันนั้นๆ

3.4.2 ระบบปรับอุณหภูมิภายในอัตโนมัติ

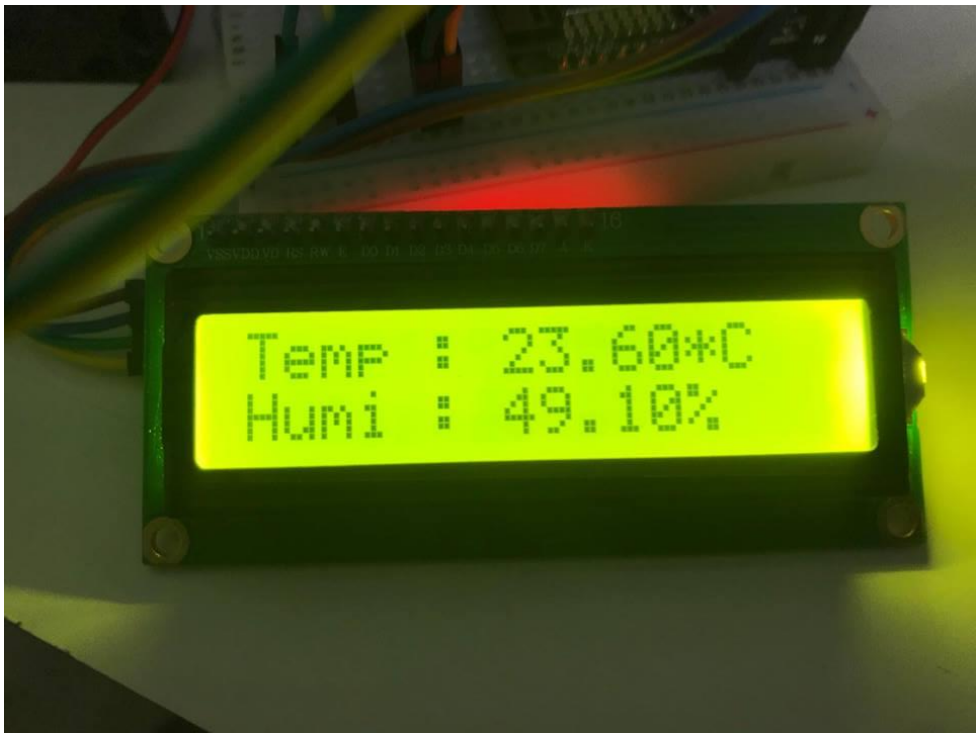
3.4.2.1 ส่วนการทำงานการสั่งการพัดลม

1. Flowchart การทำงานของระบบปรับอุณหภูมิภายในอัตโนมัติ



รูปที่ 3-10 Flowchart ระบบปรับอุณหภูมิภายในอัตโนมัติ

2. ในส่วนของตัวเครื่องนี้จะเป็นการใช้ DHT 22 ในการอ่านค่าของอุณหภูมิภายในบ้านสัตว์เลี้ยง ว่ามีอุณหภูมิเท่าไร และแจ้งให้ผู้ใช้ทราบผ่านหน้าจอ LCD โดยจะบอกทั้งความชื้นและตัวอุณหภูมิตามรูปที่ 3 – 5



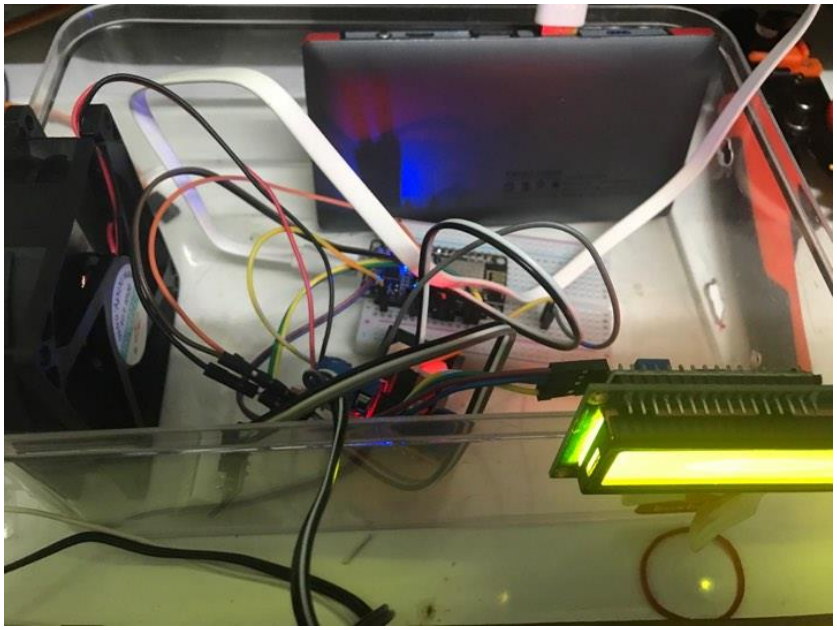
รูปที่ 3-11 จอ LCD แสดงผลอุณหภูมิ และ ความชื้น

```
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C lcd(0x27, 16, 2);
void Setup{
  Wire.begin(D2,D1);
// สั่งให้ LCD ทำงาน
  lcd.begin();
}
```

```
void loop{
    lcd.setCursor(0,0);
    lcd.print("Temp : ");
    lcd.print(t);
    lcd.print("*C");
} // กำหนดสิ่งที่จะแสดง
ให้ออกผ่านทาง LCD ที่กำหนดไว้
```

- ในส่วนของ Code จะเป็นการนำค่าที่อ่านได้จากตัววัดอุณหภูมิ DHT 22 ที่เก็บไว้ในตัวแปรต่างๆ เอาออกมาแสดงผ่านจอ LCD เพื่อให้สะดวกต่อการดูค่าอุณหภูมิและความชื้น ณ ขณะนั้น
- แต่การจะใช้ LCD ได้นั้นต้องมีการ Import Library Liquid Crystal I2C มาก่อน เพื่อให้สามารถใช้งานและเขียนค่าให้แสดงตามที่เราต้องการได้

3. ในส่วนการสั่งการพัดลมให้ทำงานโดยการทำงานของพัดลมนั้นจะทำงานก็ต่อเมื่อตัวอ่านค่าอุณหภูมิมีค่ามากกว่า 30 องศาเซลเซียส ก็จะสั่งการให้ค่าที่ต่อกับพัดลมโดยใช้คำสั่ง motor.speed(255) และจะสั่งให้พัดลมทำงานเต็มกำลังแต่ถ้าไม่ถึง 30 แต่มากกว่า 25 ก็จะสั่งให้ทำงานแค่ครึ่งเดียว



รูปที่ 3-12 ตัวระบบของปรับอุณหภูมิภายในอัตโนมัติ

ขา D1	ต่อเข้ากับขาสัญญาณ SCL ของหน้าจอ LCD แบบ I2C
ขา D2	ต่อเข้ากับขาสัญญาณ SDA ของหน้าจอ LCD แบบ I2C
ขา D4	ต่อเข้ากับขาสัญญาณของ DHT22
ขา D6	ต่อเข้ากับขาสัญญาณ IN1 ของ Motor Drive L298N
ขา D7	ต่อเข้ากับขาสัญญาณ IN2 ของ Motor Drive L298N

ตารางที่ 3-2 ตารางบอกขาที่ใช้เชื่อมต่อระบบปรับอุณหภูมิภายในอัตโนมัติ

```
  // เงื่อนไขในการให้พัดลมทำงาน
  if(t >= 30){
    motor.setSpeed(255);
  }else if( t >= 25 && t < 30){
    motor.setSpeed(128);
  }else{
    motor.stop();
}
```

```
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);
```

- โดยตัวแปร t จะเป็นตัวแปรที่เก็บค่า temperature ที่อ่านได้จาก dht22 โดยค่าที่อ่านได้จะเก็บไว้เป็นตัวแปร float และนำไปเปรียบเทียบในเงื่อนไขว่ามีค่าอุณหภูมิที่มากกว่า 30 องศาเซลเซียสหรือไม่ ถ้ามากกว่าก็ให้เป็นพัดลมทำงานโดยใช้คำสั่ง motor.speed(255) ถ้าน้อยกว่า 30 แต่มากกว่า 25 จะใช้คำสั่งเป็น motor.speed(128) จะเป็นตัวกำหนดให้ตัวพัดลมทำงาน แต่ถ้าเป็นอุณหภูมิน้อยกว่า 25 จะใช้คำสั่ง motor.stop() เพื่อสั่งให้พัดลมหยุด

3.4.2.2 ส่วนการทำงานการส่งค่าขึ้น Firebase

1. ต้องทำการตั้งค่าต่างๆที่ใช้ในการเชื่อมต่อกับ Firebase โดยจะมี 2 สิ่งหลักๆ ที่ใช้ในการทำให้ NodeMCU สามารถเชื่อมต่อเข้ากับ Database Realtime ของ Firebase ได้ คือ Firebase_Host และ Firebase_Auth ดังรูปที่ 3 - 7 โดยทั้ง 2 ค่านี้สามารถดูได้จาก Project ที่เราสร้างไว้ใน Firebase ซึ่งจะมีเอกลักษณ์เฉพาะของแต่ละ Firebase ไป

```
// Config Firebase
#define FIREBASE_HOST "testdht22-e02fb.firebaseio.com"
#define FIREBASE_AUTH "HO1M8U4d8C                    "
```

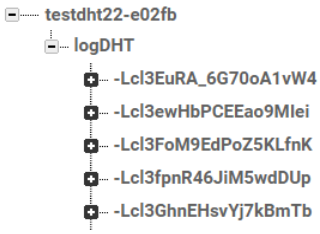
รูปที่ 3-13 ตัวอย่างการกำหนดค่าเพื่อให้สามารถใช้คำสั่ง Firebase.begin ได้

2. เมื่อกำหนดค่าทั้ง 2 ไว้ภายใน Code เรียบร้อยแล้วก็จะสามารถใช้คำสั่ง Firebase.begin เพื่อทำการส่งค่า Firebase Host และ Firebase Auth ไปเพื่อให้ NodeMCU สามารถติดต่อกับ Database Realtime ของ Firebase ได้ ซึ่งคำสั่ง Firebase.begin จะทำการเรียกใช้ภายในฟังก์ชัน void setup()
   ตัวอย่างคำสั่ง Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

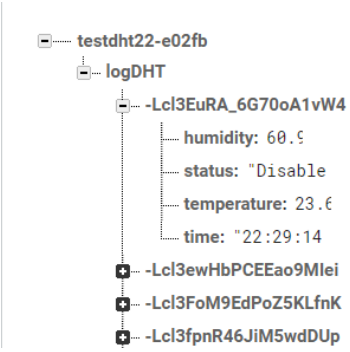
3. ในการส่งค่าที่อ่านได้จาก sensor เราจะใช้เป็นคำสั่ง Firebase.pushFloat() โดยการใช้คำสั่งนี้จะเป็นการส่งค่า root ที่มีการเก็บค่าตัวแปร 4 ตัวแปรไว้ในตัวแปร root นี้ซึ่งจะค่า t ที่ได้จากคำสั่ง dht.readTemperature(), h ที่ได้รับจากคำสั่ง dht.readHumidity(), time ที่ได้จากการเรียกใช้งานฟังก์ชัน NowString ที่แสดงค่าเวลา ณ ปัจจุบัน และตัวแปร Status ที่ได้จากการเรียกใช้งานฟังก์ชัน FanStatus(t,fs) โดยสิ่งที่จะแสดงบนหน้าจอ Port ในตัวโปรแกรมจะเป็นรหัส Token ต่างๆที่จะเป็น Key ที่บันทึกค่าต่างๆของความชื้นและอุณหภูมิที่อ่านได้จากตัว sensor โดยจะบันทึกไว้ตามรูปที่ 3 – 8 และจะเก็บแยกค่าต่างๆไว้ตามรูปที่ 3 – 9 ที่จะเป็นการบันทึกค่าอุณหภูมิและความชื้นที่อ่านได้

```
StaticJsonBuffer<200> jsonBuffer;
  JsonObject& root = jsonBuffer.createObject();
  root["temperature"] = t;
  root["humidity"] = h;
  root["time"] = NowString();
  root["status"] = FanStatus(t,fs);
```

```
float t = dht.readTemperature();
  String name = Firebase.pushFloat("logDHT", root);
  Delay(100);
  if (Firebase.failed()) {
.... เช็คว่าสามารถเชื่อมต่อ Firebase ได้ไหม }
```



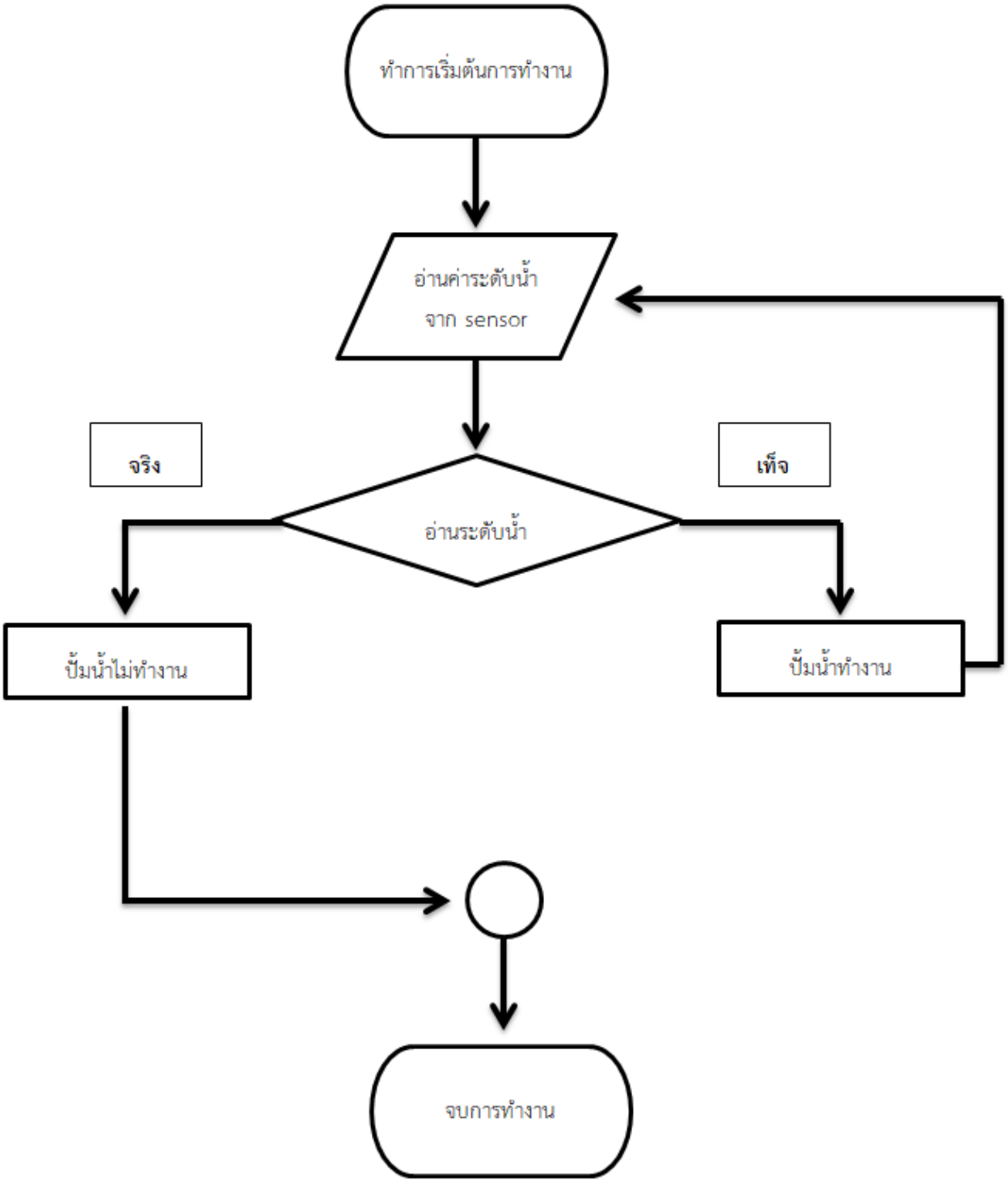
รูปที่ 3-14 รูปแบบการเก็บค่าใน Firebase



รูปที่ 3-15 ค่าต่างๆที่ส่งมาจาก NodeMCU

3.4.3 ระบบเครื่องให้น้ำอัตโนมัติ

1. Flowchart แสดงการทำงานของระบบให้น้ำอัตโนมัติ



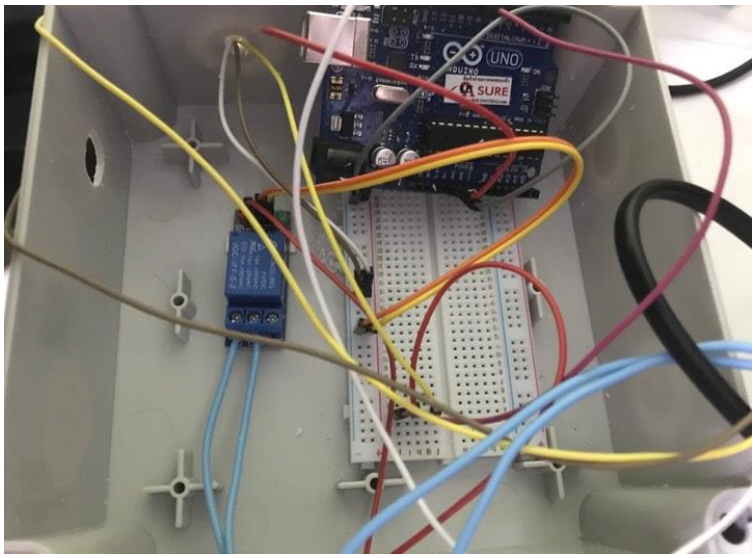
รูปที่ 3-16 Flowchart ระบบเครื่องให้น้ำอัตโนมัติ

2. จะเป็นการทำงานโดยใช้ตัว Sensor ตรวจจับระดับของน้ำแบบไม่ต้องจุ่มลงไปในน้ำเป็นตัวตรวจจับว่าน้ำในถาดน้ำของสัตว์เลี้ยงน้อยกว่าระดับที่กำหนดหรือไม่

   - โดยตัว Sensor จะมีการส่งค่า High หรือ Low กลับมาให้กับตัวบอร์ดที่เชื่อมต่อโดยตัว Sensor จะส่งค่า High มาให้ก็ต่อมือ Sensor แปะแล้วเจอน้ำ แต่ถ้า Sensor ตรวจจับไม่เจอน้ำก็จะคืนค่า Low กลับมาให้กับโปรแกรม
   - โดยเราสามารถเขียนโปรแกรมแสดงเงื่อนไขการทำงานของอุปกรณ์นี้ได้จากการที่ Sensor ส่งค่า High Low กลับมาตามตัวอย่าง Code ข้างต้น

```
If(digitalRead(9) == 1)
        Serial.println("Water Detect");
else
        Serial.println("No Water");
```

3. เมื่อเราสามารถใช้ค่าที่อ่านได้จาก Sensor ได้แล้วก็ให้ทำการเขียนเงื่อนไขเพิ่มเติมเข้าไปคือ การสั่งให้ตัวปั๊มน้ำทำงานโดยตัวปั๊มน้ำจะทำงานเมื่อ Sensor ไม่สามารถตรวจจับน้ำภายในถาดได้และในที่นี้ก็จะหมายถึงน้ำในถาดหมด ก็จะทำการสั่งให้ Relay ทำงานไปต่อระบบไฟให้ตัวปั๊มน้ำทำงานได้ และเมื่อ Sensor ส่งค่า High มาก็สั่งให้ Relay ตัดเพราะว่าน้ำถึงระดับที่กำหนดไว้แล้วตามรูปที่ 3 -10 เพื่อให้วงจรทำงานได้สมบูรณ์ขึ้น



รูปที่ 3-17 ตัวระบบของเครื่องให้น้ำอัตโนมัติ

ขา 9	ต่อเข้ากับขาสัญญาณของ Relay
ขาสัญญาณ Relay	ต่อเข้ากับขาสัญญาณของ Non-Contact Water

ตารางที่ 3-3 ตารางบอกขาที่ใช้เชื่อมต่อระบบปรับอุณหภูมิภายในอัตโนมัติ

3.5 แผนการดำเนินงาน

ตารางที่ 3-4 ตารางงาน Project Prepare

การดำเนินงาน / ระยะเวลา	ปี พ.ศ.2561																			
	มกราคม				กุมภาพันธ์				มีนาคม				เมษายน				พฤษภาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
วางแผนงานที่จะต้องทำในแต่ช่วงเวลาควรทำอะไร																				
ออกแบบรูปแบบที่อยู่และจัดวางสิ่งต่างๆ ที่ต้องการจะติดตั้ง																				
ศึกษาโปรแกรมเพื่อทำความเข้าใจและศึกษาการจัดวางโปรแกรมที่ใช้เข้ากับส่วนของ Hardware																				
ทำการสร้างเครื่องให้อัตโนมัติที่ต้องใช้และทำการตั้งค่า sensor ต่างๆที่จะใช้และทำการตรวจสอบว่าทำงานได้อย่างถูกต้องหรือไม่																				
ทดสอบโปรแกรมเพื่อแก้ไขและปรับปรุง																				

ตารางที่ 3-5 ตารางงาน Project 1

การดำเนินงาน / ระยะเวลา	ปี พ.ศ.2561																			
	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน				ธันวาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
วางแผนงานที่จะต้องทำในแต่ช่วงเวลาควรทำอะไร																				
ทำการสั่งซื้อของเพิ่มเติมจากสิ่งที่ขาดหายไป																				
ทำเครื่องให้อาหารอัตโนมัติให้สำเร็จเสร็จสิ้น โดยใช้สิ่งของที่สั่งมาเพิ่มเติมทำให้ตัวเครื่องให้อาหารสมบูรณ์มากยิ่งขึ้น																				
แก้ไขปัญหาต่างๆเกี่ยวกับสิ่งที่เกิดขึ้นในตัวเครื่องให้อาหารอัตโนมัติ และเตรียมตัวสำหรับการนำเสนอ Project																				
ทำการศึกษาการทำงานของ Firebase ที่นำมาใช้เป็นฐานข้อมูล																				
ทำการ upload ข้อมูลที่ได้จากการอ่านค่าของตัว Sensor ขึ้น Server																				

ตารางที่ 3-6 ตารางงาน Project 2

การดำเนินงาน / ระยะเวลา	ปี พ.ศ.2562																			
	มกราคม				กุมภาพันธ์				มีนาคม				เมษายน				พฤษภาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
วางแผนงานที่จะต้องทำในแต่ช่วงเวลาควรทำอะไร																				
ทำการสั่งซื้อของเพิ่มเติมจากสิ่งที่ขาดหายไป																				
ทำการเชื่อมต่อข้อมูลทีอ่านได้จากตัว Sensor ต่างๆให้เก็บข้อมูลไว้ใน Firebase																				
แก้ไขปัญหาต่างๆเกี่ยวกับการรวมระบบแต่ละส่วนของ Projectให้เป็นระบบเดียวกันและเตรียมตัวสำหรับการนำเสนอ Project																				
ทำระบบเครื่องให้น้ำอัตโนมัติให้เสร็จสิ้นและนำมาเข้ารวมกับระบบ																				
ทำการแก้ไขครั้งสุดท้ายและทำการเตรียมพร้อมเพื่อนำเสนอในส่วนของ Project																				

บทที่ 4 การทดลอง

4.1 การทดลองที่ 1 การตรวจจับสิ่งที่อยู่ในภาชนะเป็นของเหลว

4.1.1 วัตถุประสงค์

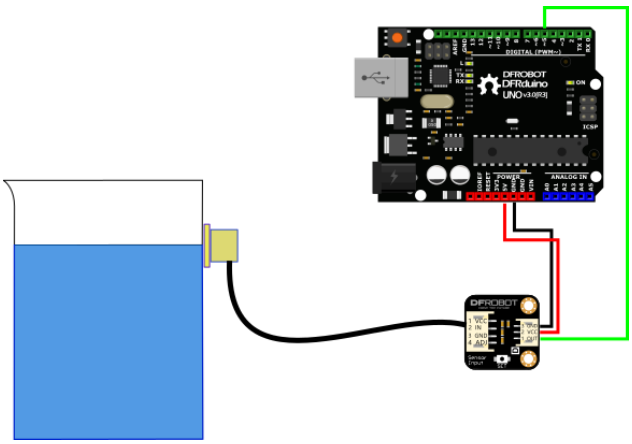
- เพื่อทำการทดลองให้สามารถอ่านค่าได้ว่าสิ่งที่อยู่ภายในภาชนะสัตว์เลี้ยงเป็นของเหลวเพื่อจะได้นำไปใช้ในการวัดระดับน้ำในการพัฒนาต่อไป

4.1.2 อุปกรณ์

- Non-contact liquid water
- Board Arduino
- สายไฟ
- สาย USB เพื่อใส่ Code เข้าไปในบอร์ด

4.1.3 วิธีการทดลอง

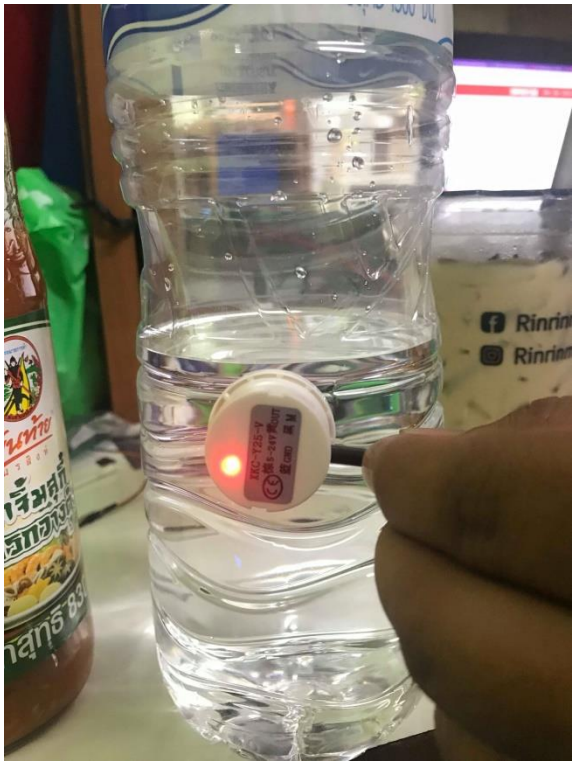
- ต่ออุปกรณ์ให้ถูกต้องตามหลักการตามรูปที่ 4 - 1



รูปที่ 4-1 การต่ออุปกรณ์เพื่อใช้งาน

- เมื่อทำการต่อเสร็จสิ้นจะสามารถใช้งานได้ทันทีโดยเราจะสามารถนำไปแตะที่ภาชนะต่างๆเพื่อตรวจสอบภายในภาชนะเป็นของเหลวหรือไม่
- โดยจะมีการติดดับของ LED สีแดงโดยถ้าภายในวัตถุมีของเหลว LED ก็จะติดแต่ถ้าไม่มีก็จะไม่มีการติดของ LED ตามรูปที่ 4 – 2 และรูปที่ 4 - 3

รูปที่ 4-3 เมื่อไม่มีของเหลวอยู่ตรงระดับ Sensor



รูปที่ 4-2 เมื่อมีของเหลวอยู่ตรงระดับ Sensor

4.1.4 ผลการทดลอง

- จากการทดลองจะเห็นได้ว่าเราจะสามารถใช้เป็นตัวเช็คระดับน้ำได้ว่าในระดับนี้มีน้ำอยู่หรือไม่
- จากการทดลองสามารถเช็คได้ในภาชนะหลายๆแบบเช่น ในขวดสุกี้ที่เป็นแก้ว หรือ จะเป็นแก้วพลาสติกที่ใส่น้ำชาไว้ ก็สามารถบอกได้ว่าตรงที่ Sensor ติดอยู่มีของเหลวหรือไม่ ดังรูปที่ 4 – 4 4 – 5 และ 4 – 6
- แต่ถ้าหากเราไปเอาตั้งไว้บนอุปกรณ์ที่เป็นสัญญาณแม่เหล็กไฟฟ้าเช่น โทรศัพท์ โน้ตบุ้ค ก็จะแจ้งเตือนที่ไฟ LED เช่นกัน

รูปที่ 4-6 ขวดน้ำจิ้มสุกี้

รูปที่ 4-5 แก้วน้ำพลาสติกมีน้ำ

รูปที่ 4-4 แก้วน้ำพลาสติกไม่มีน้ำ

4.1.5 สรุปผลการทดลอง

- ผลการทดลองที่ได้ทำให้เราเข้าใจได้ว่า เราสามารถนำการทดลองตรงนี้ไปดัดแปลงไปใช้เป็นตัววัดระดับของของเหลวภายในภาชนะได้โดยสามารถนำไปใช้กับการเติมน้ำภายถาดน้ำของสัตว์เลี้ยงที่เป็นส่วนของระบบให้น้ำอัตโนมัติต่อไปได้

4.2 การทดลองที่ 2 DS3231 Real Time Clock

4.2.1 วัตถุประสงค์

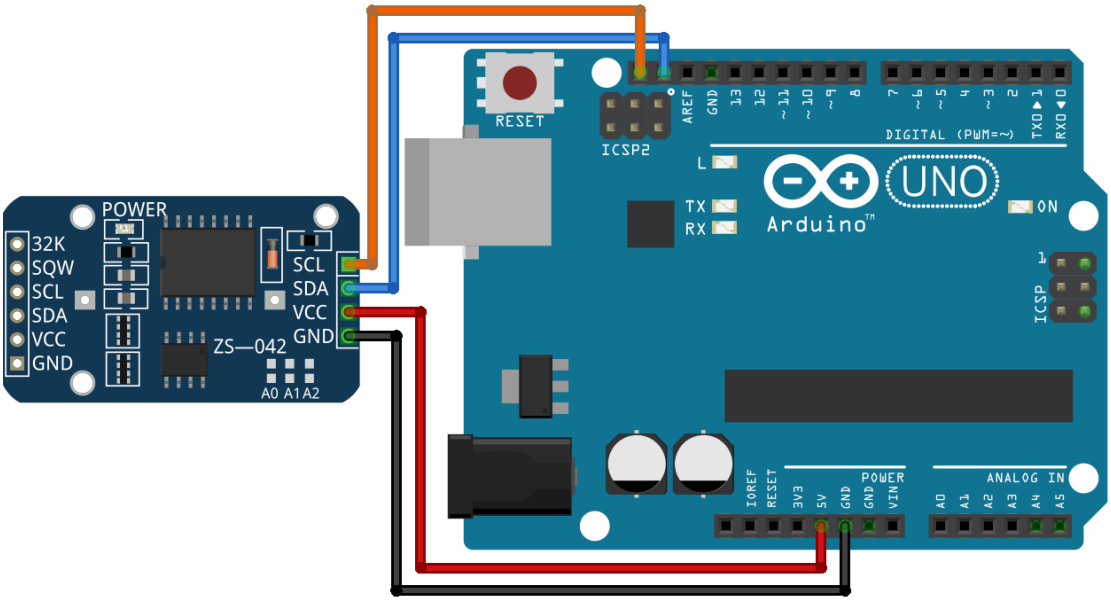
- เพื่อทำการเช็คว่าเวลานั้นวิ่งได้ถูกต้องกับที่ต้องการหรือไหมและสามารถตั้งเงื่อนไขการตั้งเตือนเมื่อถึงเวลาที่เราต้องการให้มันแจ้งเตือนเพื่อนำไปใช้กับตัวอุปกรณ์ให้อาหารอัตโนมัติ
- ทำการเช็คว่าสามารถตรวจสอบวัดอุณหภูมิได้ไหมจากตัว DS3231 ได้หรือไม่

4.2.2 อุปกรณ์

- RTC DS3231
- Board Arduino
- สายไฟ
- สาย USB เพื่อใส่ Code เข้าไปในบอร์ด

4.2.3 วิธีการทดลอง

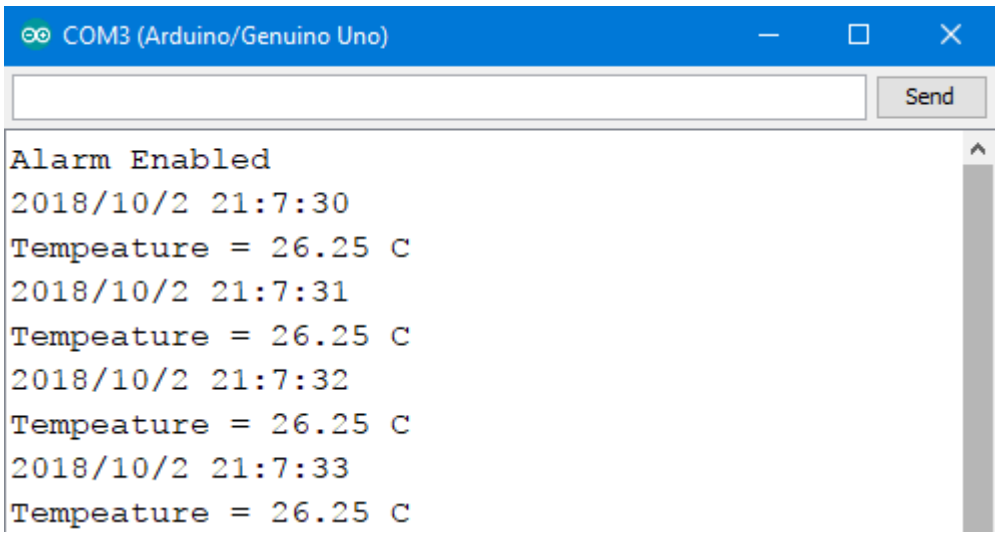
- ต่ออุปกรณ์ให้ถูกต้องตามหลักการดังรูปที่ 4 -7



รูปที่ 4-7 การต่อวงจร RTC DS3231

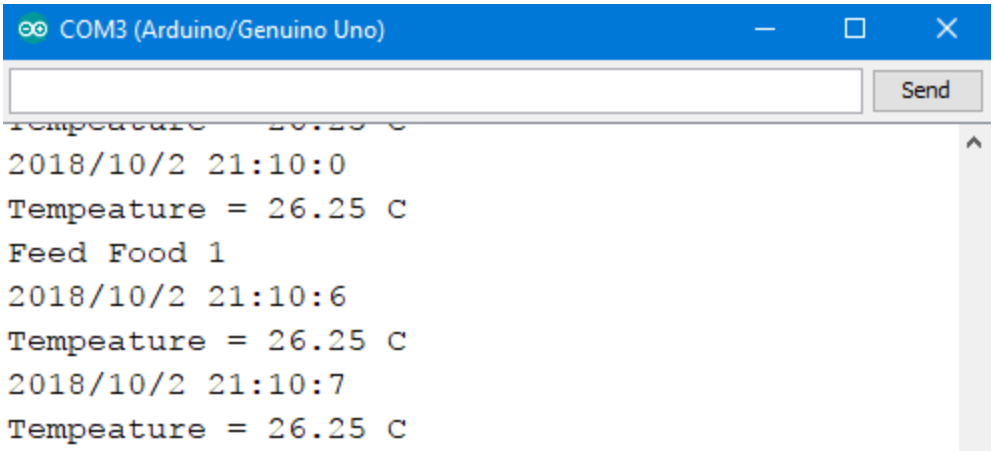
- เมื่อต่อเสร็จทำการป้อน Code ในส่วนของการทดลองใช้ตัว RTC DS3231 ที่อยู่ในส่วนของภาคผนวก

- ทำการอ่านค่าผ่าน Serial Port ที่ได้ทำการส่งค่าให้อ่านโดยจะอ่านค่าเวลาและค่าอุณหภูมิได้ตามรูปภาพด้านล่าง
    - เมื่อทำการกดดูค่าผ่าน Serial Port จะแสดงค่าออกมาโดยเริ่มคำแรกทุกครั้งที่เปิด Serial Port ว่า Alarm Enabled เพื่อแจ้งให้ทราบว่าเปิดใช้งานแล้ว และจะทำการแจ้งวันที่ เวลาและ อุณหภูมิ ณ ปัจจุบันตอนนั้นให้ทราบผ่าน Serial Port



รูปที่ 4-8 เมื่อกดเริ่มอ่านค่าที่ Serial Port

- เมื่อถึงเวลาที่กำหนดให้แจ้งเตือนก็จะแสดงข้อความว่า Feed Food ให้เห็นผ่าน Serial Port



รูปที่ 4-9 เมื่อเวลาวิ่งไปถึงเวลาที่เรากำหนดเพื่อให้แจ้งเตือน

- โดยในรูปที่ 4 – 9 จะเป็นการแจ้งเตือนเมื่อถึงเวลา 21.10 เป็นเวลาแจ้งให้ทราบ

4.2.4 ผลการทดลอง

- ผลการทดลองที่เห็นจะทำให้เราสามารถทราบค่าวันที่ และ เวลา ณ ปัจจุบันที่เราได้ทำการตั้งค่าให้ตรงหรือใกล้เคียงกับเวลาปัจจุบันมากที่สุด และทำการแจ้งอุณหภูมิที่ตัว DS3231 จับค่าได้ ณ ปัจจุบัน

4.2.5 สรุปผลการทดลอง

- จากการทดลองทำให้เราทราบว่าเราสามารถนำการจับเวลาของตัว RTC DS3231 เอามาใช้ประโยชน์ได้ในการเป็นตัวกำหนดเวลาที่จะให้อาหารของเครื่องให้อาหารอัตโนมัติ และยังสามารถอ่านค่าอุณหภูมิได้ในตัวเดียวกันทำให้ประหยัดต้นทุนการใช้งานได้

4.3 การทดลองที่ 3 ใช้ DHT22 เพื่อควบคุมการทำงานของพัดลม

4.3.1 วัตถุประสงค์

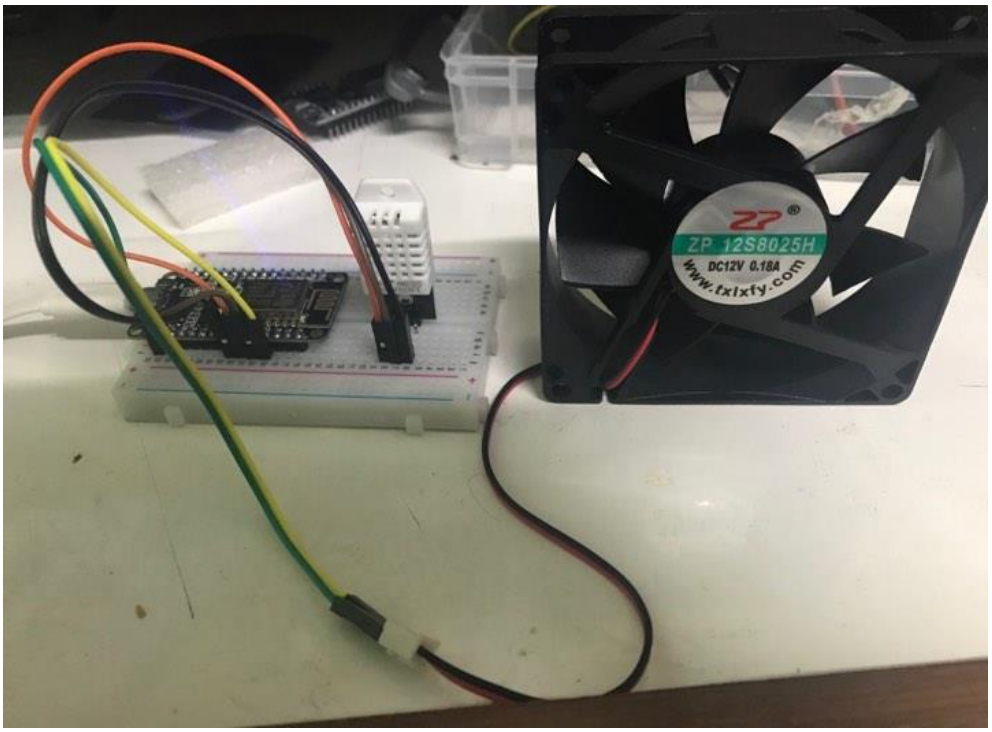
- เพื่อทำการตรวจสอบว่าสามารถอ่านค่าอุณหภูมิและความชื้นได้อย่างถูกต้อง
- เพื่อให้สามารถสั่งให้พัดลมทำงานได้เมื่ออุณหภูมิถึงจุดที่กำหนด

4.3.2 อุปกรณ์

- DHT22
- NodeMCU
- สายไฟ
- สาย Micro USB
- พัดลมคอมพิวเตอร์

4.3.3 วิธีการทดลอง

- ทำการต่อตามในรูปที่ 4 – 10 เพื่อทำการทดลอง



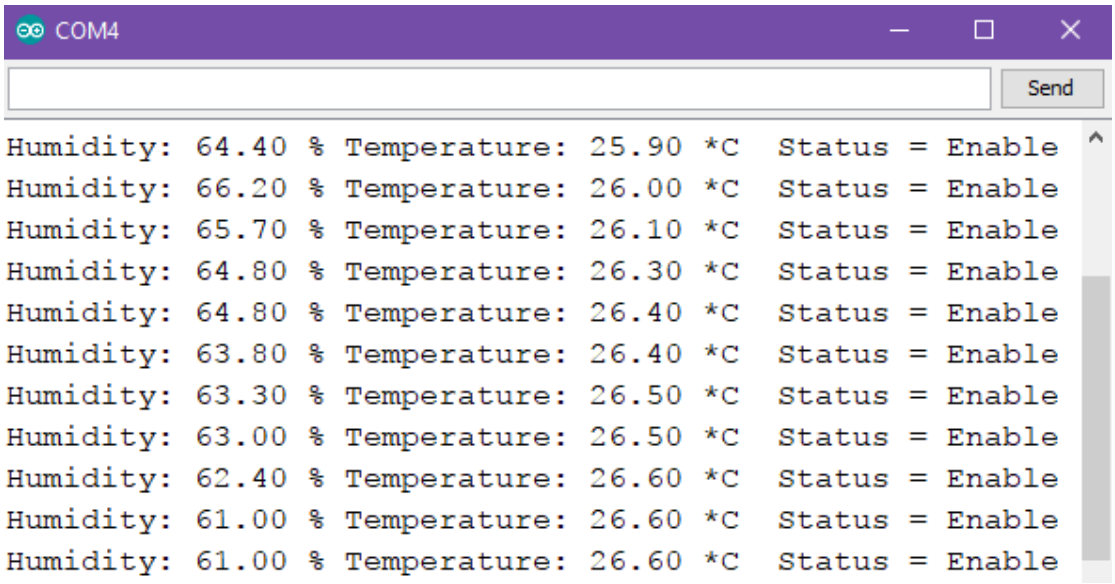
รูปที่ 4-10 ภาพการต่อวงจรของ DHT22 + พัดลม

- ทำการป้อน Code เพื่อทำการทดลองเกี่ยวกับการอ่านค่าของ DHT 22 และสร้างเงื่อนไขในการสั่งให้พัดลมทำงานดังรูปที่ 4 - 11

```
// Test Fan ( T = 31 *c )
  if(t >= 24){
    digitalWrite(0,HIGH);
    Status = "Enable";
  }
  else{
    digitalWrite(0,LOW);
    Status = "Disable";
  }
```

รูปที่ 4-11 Code ที่ใช้ในการเช็คเงื่อนไขพัดลม

- เมื่อลองทำการรัน Code ที่ตัวอย่างเข้าไปโดยเพิ่มเงื่อนไขการทำงานของพัดลมเข้าไปก็จะได้ตามรูปที่ 4 – 12



รูปที่ 4-12 ผลรันเมื่อทำการรัน Code

- โดยภายในภาพที่ 4 – 12 Status จะเป็นการบอกว่าพัดลมกำลังทำงานอยู่หรือไม่โดยถ้าทำงานจะแสดงเป็น Enable แต่ถ้าไม่ทำงานจะแสดงเป็น Disable

4.3.4 ผลการทดลอง

- จากผลการทดลองที่ได้เราจะสามารถทราบค่าอุณหภูมิ ความชื้น และสถานะของพัดลมได้ว่ามีการทำงานอยู่หรือไม่ โดยอุณหภูมิที่แสดงจะเป็นการแสดงอุณหภูมิในหน่วยองศาเซลเซียส และความชื้นจะแจ้งเป็นแบบเปอร์เซ็นที่ได้มากจาสูตรคำนวณ

4.3.5 สรุปผลการทดลอง

- จากการทดลองทำให้เราสามารถใช้ sensor ของ DHT22 เพื่ออ่านค่าอุณหภูมิและความชื้นได้ และได้นำอุณหภูมิที่อ่านได้ไปใช้เป็นเงื่อนไขในการให้พัดลมทำงาน เพื่อใช้ในการปรับอุณหภูมิภายในบ้านสัตว์เลี้ยงได้

4.4 การทดลองที่ 4 ทำการส่งค่าขึ้น Firebase

4.4.1 วัตถุประสงค์

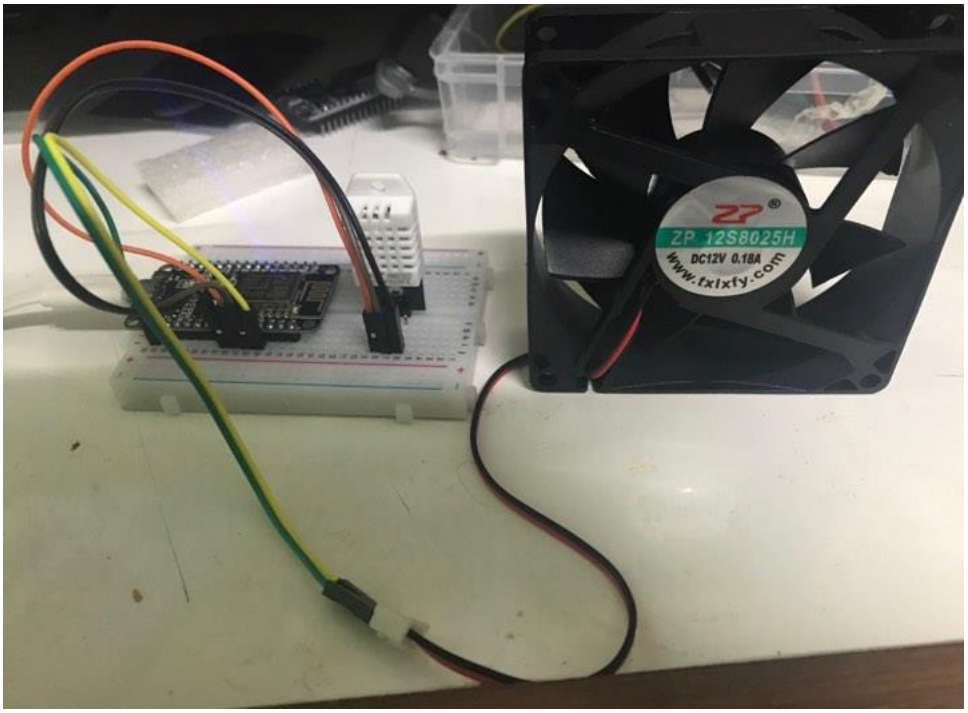
- เพื่อทำการทดสอบว่าสามารถนำค่าที่อ่านได้จาก DHT22 ส่งขึ้น Firebase ได้อย่างถูกต้อง

4.4.2 อุปกรณ์

- DHT22
- NodeMCU
- สายไฟ
- สาย Micro USB
- พัดลมคอมพิวเตอร์

4.4.3 วิธีการทดลอง

- ทำการต่อตามรูปที่ 4- 13 เพื่อทำการทดลองทดสอบการทำงาน



รูปที่ 4-13 ภาพการต่อวงจรการทดลอง

- ทำการตั้งค่าในโปรแกรมเพื่อให้สามารถเชื่อมต่อกับ Firebase ได้โดยการใช้ Firebase_Host โดยเอามาจากใน Database ของ Firebase ที่ได้ทำการสร้างเตรียมไว้ และ Firebase_Auth โดยเอามาจาก Secret Key ของ Firebase ที่เราได้ทำการสร้างไว้ มาใส่ในโปรแกรมดังรูปที่ 4 – 14

```
// Config Firebase
#define FIREBASE_HOST "testdht22-e02fb.firebaseio.com"
#define FIREBASE_AUTH "HO1M8U4d8[redacted]J"
```

รูปที่ 4-14 การตั้งค่าการเชื่อมต่อกับ Firebase

- ทำการใช้คำสั่งดังรูปภาพที่ 4 – 15 เพื่อทำการสร้าง Object ที่ใช้สำหรับส่งไปให้ Firebase โดยจะทำการเก็บค่า 4 ค่าเพื่อไว้ใช้สำหรับส่งค่าไปให้ Firebase 4 ตัวแปร

```
StaticJsonBuffer<200> jsonBuffer;
JsonObject& root = jsonBuffer.createObject();
root["temperature"] = t;
root["humidity"] = h;
root["time"] = NowString();
root["status"] = FanStatus(t,fs);
```

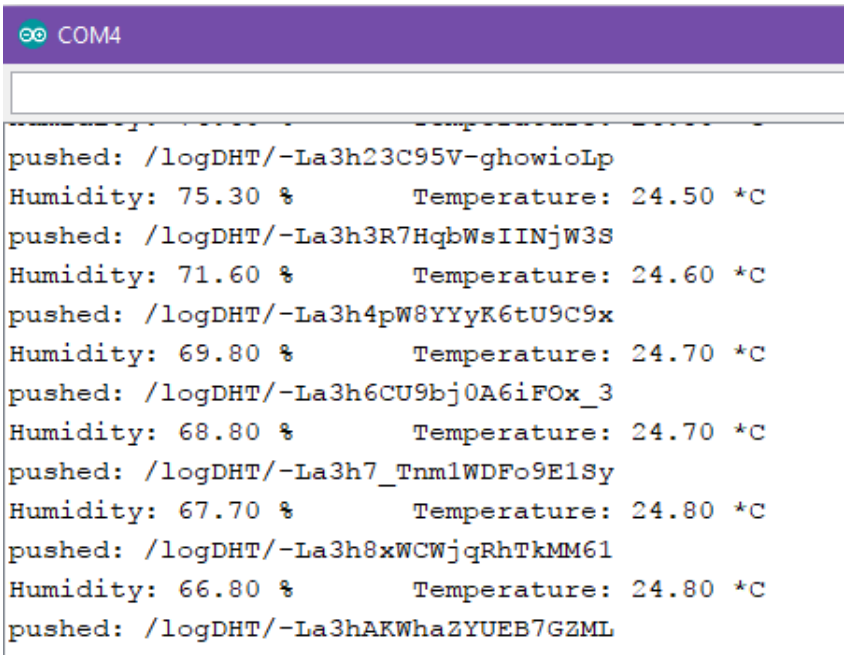
รูปที่ 4-15 คำสั่งในการสร้าง Object และใส่ตัวแปรเข้าไป

- ทำการใช้คำสั่ง Firebase.push(“logDHT”, root) ดังรูปที่ 4 - 16 เพื่อทำการส่งค่า Object ที่ได้สร้างเก็บไว้ในขั้นตอนก่อนหน้านี้เพื่อให้ Firebase ได้ทำการเก็บค่าเหล่านี้ไว้ข้างใน Database

```
String name = Firebase.push("logDHT", root);
```

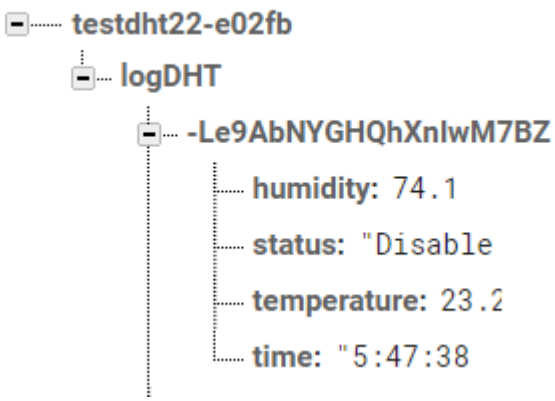
รูปที่ 4-16 คำสั่งที่ใช้ส่งค่าขึ้น Firebase

- หลังจากทดลองส่งค่าใน Serial Port จะมีการแสดงค่าดังรูปที่ 4 – 17 โดยจะมีการแสดงค่าความชื้น อุณหภูมิ และ Key log ที่ถูกบันทึกไว้ใน Firebase โดยเราสามารถเปิดได้ตาม Key log ที่บันทึกไว้



รูปที่ 4-17 ผลที่ออกผ่าน Serial Port

- ค่าที่อยู่ใน Firebase จะมีการแก็บค่า Key ไว้และใน Key จะมีตัวแปรต่างๆที่เราส่งมาจากตัวโปรแกรมมาเก้บไว้ใน Firebase ดังรูปที่ 4 – 18



รูปที่ 4-18 ค่าต่างๆที่เก็บไว้ใน Firebase

4.4.4 ผลการทดลอง

- จากการทดลองเราสามารถส่งค่าต่างๆที่เราอ่านค่าได้จาก DHT22 ให้ไปเก็บไว้ในฐานข้อมูลของ Firebase ได้โดยจะเป็นการส่งแบบ Object โดยเก็บตัวแปรทั้งหมดไว้ในตัวแปรเดียวแล้วค่อยส่งขึ้นไปทำให้เราสามารถเก็บค่าต่างๆได้โดยการส่งเพียงครั้งเดียว

4.4.5 สรุปผลการทดลอง

- จากผลการทดลองและการรันต่างๆทำให้เราสามารถเก็บค่าข้อมูลทั้งหมดที่เราต้องการให้ไปไว้อยู่ใน Firebase ได้ตามที่เรากำหนดโดยอาจจะนำการทดลองนี้ไปใช้ร่วมกันการส่งค่าอย่างอื่นได้ซึ่งทำให้เราสามารถดึงค่าข้อมูลต่างๆที่อยู่ในฐานข้อมูลนี้ไปแสดงบนหน้า website ได้สะดวกมากยิ่งขึ้น

บทที่ 5 สรุปผล

5.1 ผลการดำเนินงาน

- Prepare Project
    - ศึกษาการทำงานของอุปกรณ์ต่างๆที่ต้องนำมาใช้
    - ศึกษาการเขียน application ที่ใช้ทำการร่วมกับอุปกรณ์ Hardware
    - ศึกษาการเชื่อมต่อและการส่งข้อมูลระหว่าง Application กับอุปกรณ์ Hardware
    - เขียนรูปเล่มโครงงาน
    - วางรูปแบบแผนงานและโครงสร้างของตัวโครงงาน
- Project 1
    - ทำการออกแบบและสร้างเครื่องอุปกรณ์ต่างๆที่ใช้ใน Project
    - แก้ไขปัญหาต่างๆที่เกิดขึ้นเกี่ยวกับตัวองค์ประกอบของ Hardware
    - จัดเตรียมการนำเสนอ Project ให้เตรียมพร้อมมากยิ่งขึ้น
    - วางแผนการจัดการการทำงานในส่วนของ Project 1
    - ทำการ upload ข้อมูลขึ้นตัว Server
- Project 2
    - ทำการรวมระบบต่างๆให้เป็นส่วนเดียวกัน
    - แก้ไขปัญหาที่ผิดพลาดให้ประสบผลสำเร็จ
    - ทำการแก้ไขตัวกล่องให้อาหารอัตโนมัติที่มีปัญหาให้แล้วเสร็จดียิ่งขึ้น
    - ทำการเพิ่มระบบต่างๆที่วางแผนไว้ให้มารวมอยู่ในระบบ
    - ทำการเชื่อมต่อ Firebase เข้ากับ ระบบของตัว Project ให้มีการดึงข้อมูลมาแสดงผลได้ในทุกส่วนของระบบ

5.2 สรุปผล

- ทดลองต่อ Hardware เบื้องต้น ทำให้สามารถควบคุมการทำงานของตัว Servo Motor ให้ทำงานได้ตามที่เราต้องการซึ่งทำให้นำไปใช้ต่อกับตัวเครื่องให้อาหารอัตโนมัติต่อได้
- ทำการสร้างอุปกรณ์ Hardware ที่ใช้ในการทำงานของ Project โดยเน้นไปที่ส่วนของ Hardware เป็นหลักเพื่อให้การทำงานเป็นไปได้อย่างดีไม่มีข้อผิดพลาด

- จากการทดลองการใช้ Sensor ตรวจจับการระดับของน้ำภายในภาชนะสามารถนำมาใช้ทำระบบเครื่องเติมน้ำอัตโนมัติได้โดยการใช้ Sensor ที่อ่านค่ามาเป็นตัวตัดการทำงานของ Relay ที่ใช้ควบคุมปั้มน้ำสำหรับมาเติมน้ำ
- ทำให้สามารถออกแบบระบบควบคุมอุณหภูมิภายในบ้านสัตว์เลี้ยงได้โดยการใช้การอ่านค่าของ DHT22 แล้วนำข้อมูลเก็บไว้ใน Firebase และเรียกใช้ข้อมูลผ่าน Firebase แล้วนำมาแสดงผ่านทางหน้า website ให้สามารถรับชมข้อมูลได้อย่างง่าย

5.3 ปัญหาและอุปสรรค

- ผู้พัฒนาไม่มีความรู้ ความสามารถในการทำความเข้าใจตัว Software และ Hardware ของตัว Arduino ได้ดีพอจึงทำให้เกิดปัญหาในการสั่งให้ตัว Hardware ทำงานหรือการเชื่อมต่อวงจรของตัว Hardware เพื่อรับค่าจาก Software ยังทำได้ไม่เต็มที่เท่าที่ควร
- ไม่สามารถใช้ Library ได้อย่างที่ต้องการต้องทำการปรับเปลี่ยนแก้ไขเพื่อให้สามรถใช้ได้กับการทำงานของตัวอุปกรณ์ Hardware และยังไม่สามารถทำให้การทำงานของอุปกรณ์แต่ละส่วนทำงานร่วมกันได้ดีเท่าที่ควร
- Library ที่เอามาใช้งานไม่สามารถใช้งานได้กับตัว Hardware ที่มีจึงต้องทำการเปลี่ยน Library ใหม่เพื่อให้เข้ากับอุปกรณ์ Hardware นั้นๆ เพราะในบาง Library จะเป็นตัวกำหนดให้ใช้เฉพาะแค่สถาปัตยกรรมนั้นๆ ไม่สามารถใช้กับ สถาปัตยกรรมอื่นๆ ได้
- พบเจอปัญหาในการนำระบบต่างๆเข้ารวบรวมเข้ากับ Firebase เนื่องจากไม่สามารถทำการเชื่อมต่อให้เป็นแบบ Database Realtime ได้ในตอนแรก จึงต้องดูในส่วนของ Library ให้ดีและในส่วนของllkkการดึง Firebase Host และ รหัสยืนยันของ Database Firebase มาใช้ให้ถูกต้อง
- ปริมาณอาหารที่ไหลลงมาทางช่องผ่านอาหารของตัวระบบเครื่องให้อาหารอัตโนมัติเกิดการติดขัดขึ้นเนื่องจากปริมาณอาหารที่หล่นลงมามีมากเกินไปทำให้เกิดการติดขัดไม่มีการหล่นของอาหารต้องมีการขยับตัวกล่องให้อาหารไหลลงมาที่ลิ้นกั้นอาหารเป็นปกติ

5.4 ข้อเสนอแนะ / แนวทางการพัฒนาต่อ

- ทำการศึกษา และทำความเข้าใจการทำงานของตัว Software และ Hardware ของ Arduino มาให้ดีเพื่อป้องกันไม่ให้เกิดปัญหาในการทำงาน
- สามารถปรับเปลี่ยนวัสดุที่ใช้มาเป็นอุปกรณ์ต่างๆให้มีความแข็งแรงทดทานได้เพื่อให้ระบบมีความทนทานในการใช้งานของตัวอุปกรณ์ต่างๆ
- สามารถใช้ KeyPad เพื่อทำการกำหนดเวลาการให้อาหารภายนอกได้โดยไม่ต้องเข้าโปรแกรมเพื่อกำหนดเวลา และสามารถเปลี่ยน Servo เป็นตัวที่ใหญ่ขึ้นหมุนได้ 180 องศาเพื่อให้มีความมั่นคงยิ่งขึ้น

- สามารถเปลี่ยน Database ของ Firebase เป็น Database แบบอื่นๆได้ที่มีความเสถียรมากกว่าการใช้ Firebase เช่น Mysql เป็นต้น
- สามารถนำบอร์ด Microcontroller ตัวอื่นๆมาเป็นแกนกลางในการทำงานของระบบต่างๆได้โดยการให้ระบบอื่นๆถูกเรียกใช้ผ่าน Microcontroller หลักเพื่อให้มีความดูแลและเช็คความพร้อมของระบบได้และสะดวกในการจัดการระบบ
- สามารถนำระบบอื่นๆเข้ามาเพิ่มเติมเข้ากับระบบนี้ได้เช่น การตรวจจับว่าสัตว์เลี้ยงยังอยู่ภายในบ้านสัตว์เลี้ยงหรือไม่ได้โดยการใช้ Image Processing วิเคราะห์ว่าสัตว์เลี้ยงยังอยู่ในห้องที่เลี้ยงอยู่เป็นต้น

บรรณานุกรม

ข้อมูลเกี่ยวกับ RTC

http://cpre.kmutnb.ac.th/esl/learning/index.php?article=ds3231-i2c-rtc

อัพเดท 09 ตุลาคม 2014

https://www.arduinoall.com/article/26/ds3231-สอน-วิธี-ใช้งาน-arduino-ds3231-at24c32-โมดูลนาฬิกา-ใช้ได้ใน-3-นาที

อัพเดท 07 ธันวาคม 2014

ข้อมูลเกี่ยวกับ Servo

https://www.thaieasyelec.com/article-wiki/review-product-article/บทความตัวอย่างการควบคุม-rc-servo-motor-ด้วย-arduino.html

อัพเดท 22 กันยายน 2560

ข้อมูลเกี่ยวกับ L298N motor drive

http://gamerbloggerport.blogspot.com/2017/09/l298n-motor-drive.html

อัพเดท 9 ตุลาคม 2559

ข้อมูลเกี่ยวกับ Arduino UNO R3

https://www.thaieasyelec.com/products/development-boards/arduino/official-boards-made-in-italy/arduino-uno-r3-detail.html

https://www.arduinoall.com/product/16/arduino-uno-r3-พร้อมสาย-usb-คอร์สเรียน-arduino-starter-ออนไลน์

https://www.thaieasyelec.com/article-wiki/basic-electronics/บทความ-arduino-คืออะไร-เริ่มต้นใช้งาน-arduino.html

อัพเดท 28 กรกฎาคม 2557

ข้อมูลเกี่ยวกับ Relay

http://www.psptech.co.th/รีเลย์relayคืออะไร-15696.page

ภาคผนวก

ก) Source Code ระบบเครื่องให้อาหารอัตโนมัติ

```
// Date, Time and Alarm functions using a DS3231 RTC connected via I2C and Wire lib

#include "Wire.h"
#include "SPI.h"  // not used here, but needed to prevent a RTClib compile error
#include "RTClib.h"
#include "Servo.h"

RTC_DS3231 RTC;
Servo myservo;

// stop the food from flowing
void feederClose() {
  myservo.write(145);
}

// release a ration of food
void feederOpen() {
  myservo.write(180);
}
void setup () {

  myservo.attach(9);

  Serial.begin(9600);
  Wire.begin();
  RTC.begin();
  RTC.adjust(DateTime(__DATE__, __TIME__));

  if (! RTC.isrunning()) {
    Serial.println("RTC is NOT running!");
    // following line sets the RTC to the date & time this sketch was compiled
    RTC.adjust(DateTime(__DATE__, __TIME__));  }
```

```
DateTime now = RTC.now();
// ตั้งเวลา ในตัวอย่างนี้ เซตค่าเป็นเวลา 23:09 ถ้าถึงเวลานี้จะให้ทำงานที่ฟังก์ชัน
  RTC.setAlarm1Simple(16,16);
   if (RTC.checkIfAlarm(1)) {
   Serial.println("Alarm Triggered");
   }

  RTC.turnOnAlarm(1);

  if (RTC.checkAlarmEnabled(1)) {
    Serial.println("Alarm Enabled");
  }}
void loop () {

  DateTime now = RTC.now();
  Serial.print(now.year(), DEC);
  Serial.print('/');
  Serial.print(now.month(), DEC);
  Serial.print('/');
  Serial.print(now.day(), DEC);
  Serial.print(' ');
  Serial.print(now.hour(), DEC);
  Serial.print(':');
  Serial.print(now.minute(), DEC);
  Serial.print(':');
  Serial.println(now.second(), DEC);

  if (RTC.checkIfAlarm(1)) {
    Serial.println("Feed Food");
    feederOpen();
    delay(5000);
    feederClose();  }
  else
    myservo.write(145);
  delay(1000);
}
```

ข) Source Code ระบบปรับอุณหภูมิภายในบ้านสัตว์เลี้ยง

```
#include <ESP8266WiFi.h>
#include <FirebaseArduino.h>
#include <DHT.h>
#include <time.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>
#include <L298N.h>

// Config Firebase
#define FIREBASE_HOST "testdht22-e02fb.firebaseio.com"
#define FIREBASE_AUTH "HO1M8U4d8Ojdr9uRqDByRVfcxA3FiHgf9debbXNJ"

// Config connect WiFi
#define WIFI_SSID "game"
#define WIFI_PASSWORD "123456789"

// Config DHT
#define DHTPIN D4
#define DHTTYPE DHT22

// Config Fan
#define EN D0
#define IN1 D6
#define IN2 D7

// Config time
int timezone = 7;

char ntp_server1[20] = "ntp.ku.ac.th";
char ntp_server2[20] = "fw.eng.ku.ac.th";
char ntp_server3[20] = "time.uni.net.th";
int dst = 0;
```

```
DHT dht(DHTPIN, DHTTYPE);
LiquidCrystal_I2C lcd(0x27, 16, 2);
L298N motor(EN, IN1, IN2);

void setup() {

  pinMode(DHTPIN, INPUT);

  Serial.begin(9600);

  Wire.begin(D2,D1);
  lcd.begin();

  WiFi.mode(WIFI_STA);
  // connect to wifi.
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("connecting");

  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("connected: ");
  Serial.println(WiFi.localIP());

  configTime(timezone * 3600, dst, ntp_server1, ntp_server2, ntp_server3);
  Serial.println("Waiting for time");
  while (!time(nullptr)) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.println("Now: " + NowString());
```

```
Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);

  dht.begin();
  motor.setSpeed(80); // an integer between 0 and 255
}

void loop() {
   delay(100);
  // Read temp & Humidity for DHT22
  float h = dht.readHumidity();
  float t = dht.readTemperature();
  motor.forward();
  String fs = "Prepare Fan";
  fs = FanStatus(t,fs);

  delay(100);

  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    delay(500);

    lcd.clear();
    lcd.setCursor(0,0);
    lcd.print("DHT Failed");
  }
  else{
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.print(" *C ");
  Serial.print(FanStatus(t,fs));
  Serial.println();
```

```
lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temp : ");
  lcd.print(t);
  lcd.print("*C");
  lcd.setCursor(0,1);
  lcd.print("Humi : ");
  lcd.print(h);
  lcd.print("%");

  delay(2000);

  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Status: ");
  lcd.setCursor(0,1);
  lcd.print(fs);

  StaticJsonBuffer<200> jsonBuffer;
  JsonObject& root = jsonBuffer.createObject();
  root["temperature"] = t;
  root["humidity"] = h;
  root["time"] = NowString();
  root["status"] = FanStatus(t,fs);

delay(2500);

  if(WiFi.status() != WL_CONNECTED) {
     Serial.println("WiFi Disconnect");
     lcd.clear();
     lcd.setCursor(0,0);
     lcd.print("WiFi Failed");
     WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
     Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
     delay(500);      }
```

```
else{
    String name = Firebase.push("logDHT", root);
    delay(100);
    // handle error
    if(Firebase.success())    {
      Serial.print("pushed: /logDHT/");
      Serial.println(name);    }
    if (Firebase.failed()) {
      Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
      delay(100);
      return ;    }}
  delay(2500);  }}

String NowString() {
  time_t now = time(nullptr);
  struct tm* newtime = localtime(&now);
  String tmpNow = "";
  tmpNow += String(newtime->tm_hour);
  tmpNow += ":";
  tmpNow += String(newtime->tm_min);
  tmpNow += ":";
  tmpNow += String(newtime->tm_sec);
  return tmpNow;}

String FanStatus(float t,String fs){
  if(t >= 30){
    motor.setSpeed(255);
    fs = "Enable 2";  }
  else if(t >= 25 && t < 30){
    motor.setSpeed(128);
    fs = "Enable 1";  }
  else{
    motor.stop();
    fs = "Disable";  }
  return fs; }
```

ค) Source Code ระบบเครื่องให้น้ำอัตโนมัติ

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

void loop() {
  if (digitalRead(9) == 1) {
    Serial.println("Water Detect");
  } else {
    Serial.println("No Water");
  }
  delay(1000);
}
```

ง) Library ที่ใช้ในแต่ละการทำงาน

- มีการเรียกใช้ Library L298N.h โดยมีการใช้คำสั่ง
  - L298N motor(EN, IN1, IN2);
  - motor.forward();
  - motor.stop();
  - motor.setSpeed(x); // x = 0 – 255
- มีการเรียกใช้ Library FirebaseArduino.h โดยมีการใช้คำสั่ง
  - FIREBASE_HOST // เอามาจากชื่อ Http: บนหัวข้อ Database
  - FIREBASE_AUTH // เอามาจาก Secret Key ทีอยู่ใน Firebase
  - Firebase.begin(FIREBASE_HOST, FIREBASE_AUTH);
  - Firebase.push();
  - Firebase.success();
  - Firebase.failed()
- มีการเรียกใช้ Library ESP8266WiFi.h โดยมีการใช้คำสั่ง
  - WIFI_SSID // ชื่อ WiFi ที่ต้องทำการเชื่อมต่อ
  - WiFi_PASSWORD // Password WiFi ที่ต้องการเชื่อมต่อ
  - WiFi.mode(WIFI_STA);
  - WIFI.begin(WIFI_SSID,WIFI_PASSWORD);
  - WiFi.status();
  - WL_CONNECTED

- WiFi.localIP()

- มีการเรียกใช้ Library DHT.h โดยมีการใช้คำสั่ง
  - DHTPIN // เพื่อกำหนดขาการส่งสัญญาณ
  - DHTTYPE // เพื่อกำหนดประเภท DHT ที่ใช้
  - Dht.begin
  - Dht.readHumidity();
  - Dht.readTempearature();
- มีการเรียกใช้ Library LiquidCrystal_I2C.h โดยมีการใช้คำสั่ง
  - Lcd.begin();
  - Lcd.clear();
  - Lcd.setCursor(0,0);
  - Lcd.print(t);
- มีการเรียกใช้ Library Wire.h โดยมีการใช้คำสั่ง
  - Wire.begin(D2,D1); // กำหนดขา I2C
- มีการเรียกใช้ Library time.h โดยมีการใช้คำสั่ง
  - timezone = 7;
  - ntp_server1[20] = "ntp.ku.ac.th";
  - ntp_server2[20] = "fw.eng.ku.ac.th";
  - ntp_server3[20] = "time.uni.net.th";
  - configTime(timezone * 3600, dst, ntp_server1, ntp_server2, ntp_server3);
- มีการเรียกใช้ Library Servo.h โดยมีการใช้คำสั่ง
  - Servo myservo;
  - Myservo.write(x);
  - Myservo.attach(9);
- มีการเรียกใช้ Library SPI.h โดยมีการใช้คำสั่ง
  - ใช้แก้ปัญหาในกรณีที่ไม่สามารถ compile Library RTC ได้
- มีการเรียกใช้ Library RTClib.h โดยมีการใช้คำสั่ง
  - RTC_DS3231 RTC;
  - RTC.begin();
  - RTC.adjust(DateTime(__DATE__, __TIME__));
  - RTC.isrunning()
  - RTC.now();
  - RTC.setAlram1Simple
  - RTC.checkIfAlarm(1)

- RTC.turnOnAlarm(1);
- RTC.checkAlarmEnable(1)