รายงานความก้าวหน้า 242-402 โครงงานวิศวกรรมคอมพิวเตอร์ 2 ครั้งที่ 2/2565

หุ่นยนต์เคลื่อนที่ควบคุมด้วยระบบ GPS

GPS Mobile Robot

นายปวเรศ ปิ่นแก้ว

รหัสนักศึกษา 6010110680

อาจารย์ที่ปรึกษาโครงงาน

.....

(ผศ.ดร.ธเนศ เคารพาพงศ์)

รายงานความก้าวหน้าโครงงานวิศวกรรมคอมพิวเตอร์นี้เป็นส่วนหนึ่งของการศึกษา

ตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์

5 ตุลาคม 2565

ชื่อโครงงาน	หุ่นยนต์เคลื่อนที่ควบคุมด้วยระบบ GPS		
ผู้จัดทำ	นายปวเรศ ปิ่นแก้ว	**รหัสนักศึกษา**	6010110680
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ปีการศึกษา	2565		

อาจารย์ที่ปรึกษาโครงงาน

.....

(ผศ.ดร.ธเนศ เคารพาพงศ์)

คณะกรรมการสอบ

.....

(ดร.สมชัย หลิมศิโรรัตน์)

.....

(ดร.อนันต์ ชกสุริวงค์)

.....

(ผศ.ดร.นิคม สุวรรณวร)

.....

(ผศ.ดร.ธเนศ เคารพาพงศ์)

โครงงานนี้เป็นส่วนหนึ่งของรายวิชาโครงงานวิศวกรรมคอมพิวเตอร์ 1 และ 2 ตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยสงขลานครินทร์

.....

(รศ.ดร.พิชญา ตัณฑัยย์)

รักษาการแทนหัวหน้าสาขาวิชา

วิศวกรรมคอมพิวเตอร์

หนังสือรับรองความเป็นเอกลักษณ์

ข้าพเจ้าผู้ลงนามท้ายนี้ ขอรับรองว่ารายงานฉบับนี้เป็นรายงานที่มีความเป็นเอกลักษณ์ โดยที่ข้าพเจ้ามิได้มีการคัดลอกมาจากที่ใด เนื้อหาในรายงานทั้งหมดถูกรวบรวมจากการพัฒนาในขั้นตอนต่าง ๆ ของการจัดทำโครงงาน หากส่วนใดที่จำเป็นต้องนำข้อความจากผลงานของบุคคลอื่นใดที่ไม่ใช่ตัวข้าพเจ้า ข้าพเจ้าได้ทำการอ้างอิงถึงเอกสารเหล่านั้นไว้อย่างเหมาะสม และขอรับรองว่ารายงานฉบับนี้ไม่เคยเสนอต่อสถาบันใดมาก่อน

ผู้จัดทำ

.....

(นายปวเรศ ปิ่นแก้ว)

ชื่อโครงงาน	หุ่นยนต์เคลื่อนที่ควบคุมด้วยระบบ GPS		
ผู้จัดทำ	นายปวเรศ ปิ่นแก้ว	**รหัสนักศึกษา**	6010110680
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ปีการศึกษา	2565		

บทคัดย่อ

เนื้อความบทคัดย่อ

โครงงานหุ่นยนต์เคลื่อนที่ควบคุมด้วยระบบ GPS เป็นโครงงานที่พัฒนาขึ้นเพื่อนำระบบระบุตำแหน่งบนโลก(GPS) มาประยุกต์ใช้กับหุ่นยนต์ โดยใช้การระบุพิกัดของ GPS มาใช้ในการกำหนดพิกัดที่ต้องการให้หุ่นยนต์เคลื่อนที่ไป เพื่อนำไปประยุกต์ในการใช้งานต่างๆภายนอกอาคาร เช่น การส่งพัสดุหรือการสำรวจพื้นที่ โดยจะใช้หุ่นยนต์เคลื่อนที่อย่างง่าย ที่มีบอร์ด Arduino เป็นตัวหลักในการควบคุม และสั่งการเซ็นเซอร์ต่างๆของหุ่นยนต์

คำสำคัญ ระบบระบุตำแหน่งบนโลก(GPS) มุมทิศ Haversine formular Azimuth formular

Project	GPS Mobile Robot		
Author	Mr.Pawaret Pinkaew	**Student ID**	6010110680
Major Program	Computer Engineering		
Academic Year	2022		

Abstract

Description

GPS Mobile Robot is a project developed to apply global positioning system (GPS) to robots, using GPS coordinates to determine the target location that you want the robot to move. To be used in various applications outside the building such as parcel delivery or surveying the area, using a simple mobile robot that has an Arduino board as the main control. and command various sensors of the robot

Keyword Global Positioning System (GPS) Azimuth angle Haversine formular Azimuth formular

กิตติกรรมประกาศ

โครงงานนี้สำเร็จลุล่วงได้ด้วยความกรุณาจาก ผศ.ดร.ธเนศ เคารพาพงศ์ อาจารย์ที่ปรึกษาโครงงานที่ได้ให้คำเสนอแนะ แนวคิด ตลอดจนการแก้ไขข้อบกพร่องต่าง ๆ มาโดยตลอดจนโครงงานเล่มนี้เสร็จสมบูรณ์ ผู้จัดทำจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอบพระคุณ ผศ.ดร.นิคม สุวรรณวร ดร.อนันท์ ชกสุริวงค์ และ ดร.สมชัย หลิมศิโรรัตน์ ที่ให้คำแนะนำและข้อคิดต่าง ๆ ในการจัดทำโครงงานฉบับนี้

ขอบพระคุณ คุณอนันต์ นิลโกสีย์ ที่ให้คำแนะนำในการใช้งานเครื่องมือช่าง การเลือกวัสดุและอุปกรณ์ที่ใช้งานในโครงงานและออำนวยความสะดวกในการใช้งานเครื่องมือ

ขอบพระคุณคุณวิมล คำจันทร์ ที่อำนวยความสะดวกในการลงทะเบียนเรียนรายวิชาโครงงาน

ขอบพระคุณผู้ปกครอง ที่ให้การสนับสนุน และขอขอบคุณพี่ๆรวมทั้งเพื่อน ๆ ที่ให้ความช่วยเหลือและให้คำปรึกษาแนวทางในกำรทำโครงงานนี้

ปวเรศ ปิ่นแก้ว

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

ปัจจุบันหุ่นยนต์เริ่มเข้ามามีบทบาทในกิจกรรมต่างๆของมนุษย์มากขึ้น จึงมีการพัฒนาอย่างรวดเร็วและ กว้างขวาง เพื่อช่วยแบ่งเบาภาระการทำงานหรือเพิ่มประสิทธิภาพในการทำงาน เช่น หุ่นยนต์บริการในร้านอาหาร BellaBot [1] ทำหน้าที่ให้บริการต้อนรับลูกค้า รับออร์เดอร์ และเสิร์ฟอาหาร ซึ่งหุ่นยนต์ตัวนี้มีการใช้ AI ในการ ควบคุมเซ็นเซอร์หลายๆตัวเช่น การตรวจสอบการเคลื่อนไหว การระบุตำแหน่งภายในร้าน หากกล่าวถึงการระบุตำแหน่งแล้ว เทคโนโลยีที่มีความสามารถในการระบุตำแหน่งได้อย่างแม่นยำในปัจจุบันก็ คือระบบ Global Positioning System (GPS) ซึ่งสามารถนำมาประยุกต์ใช้เพื่อให้เกิดประโยชน์มากมายไม่ว่าจะ เป็น การทำแผนที่ ระบบนำร่อง หรือการติดตามยานพาหนะ อีกทั้งยังเป็นระบบที่ทุกคนสามารถเข้าถึงและใช้งาน ได้ง่าย ผู้จัดทำจึงมีความสนใจที่จะนำระบบ GPS มาประยุกต์ใช้กับหุ่นยนต์ โดยใช้การระบุพิกัดของระบบ GPS มา ใช้ในการควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์เพื่อนำไปประยุกต์ในการใช้งานต่างๆภายนอกอาคาร เช่น การส่งพัสดุหรือการสำรวจพื้นที่ โดยจะใช้หุ่นยนต์เคลื่อนที่อย่างง่าย ที่มีบอร์ด Arduino เป็นตัวหลักในการควบคุม และสั่งการเซ็นเซอร์ต่างๆของหุ่นยนต์

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างหุ่นยนต์ที่สามารถเคลื่อนที่ได้อัตโนมัติโดยใช้การระบุพิกัดด้วย GPS เป็นตัวควบคุม
2. เพื่อสร้างโปรแกรมในการกำหนดตำแหน่งเพื่อให้หุ่นยนต์เคลื่อนที่

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถสร้างหุ่นยนต์ที่สามารถเคลื่อนที่ได้อัตโนมัติตามตำแหน่งที่ระบุด้วยพิกัดจาก GPS ได้
2. สามารถสร้างโปรแกรมสำหรับการระบุตำแหน่งที่ต้องการให้หุ่นยนต์เคลื่อนที่ไปได้
3. สามารถนำการระบุพิกัด GPS ไปใช้ในการควบคุมทิศทางการเคลื่อนที่ของหุ่นยนต์ได้

1.4 ขอบเขตของโครงงาน

1. หุ่นยนต์สามารถเคลื่อนที่ได้ตามตำแหน่งที่ระบุบน GPS
2. หุ่นยนต์สามารถหลบสิ่งกีดขวางขนาดเล็กที่หยุดนิ่งได้
3. สามารถคำนวณระยะทางและความเร็วในการเคลื่อนที่ศึกของหุ่นได้

1.5แผนการดำเนินงาน

ขั้นตอนการดำเนินการ	Project Preparation (2/63)					Project I (1/64)					Project II (1/65)				
	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
1.ศึกษาหัวข้อโครงงาน															
2.ศึกษางานวิจัยและเอกสารที่เกี่ยวข้องกับโครงงาน															
3.ศึกษาระบบ GPS และระบบควบคุมที่ใช้ GPS															
4.ศึกษาการทำงานของอุปกรณ์และเซ็นเซอร์ต่างๆ ที่ใช้ในโครงงาน															
5.ศึกษาการเขียนโปรแกรมสำหรับควบคุม microcontroller															
6.ออกแบบหุ่นยนต์และระบบการทำงาน															
7.ออกแบบโปรแกรมและเขียนโปรแกรมสำหรับการทำงานของ Microcontroller															
8.พัฒนาหุ่นยนต์และเขียนโปรแกรมสำหรับควบคุม Microcontroller															
9.ทดลองและวัดผลการทดลอง															
10.สรุปผลการทดลอง															
11.จัดทำเล่มโครงงาน															

ตารางที่ 1 แผนการดำเนินงาน

บทที่ 2
ทฤษฎีและหลักการ

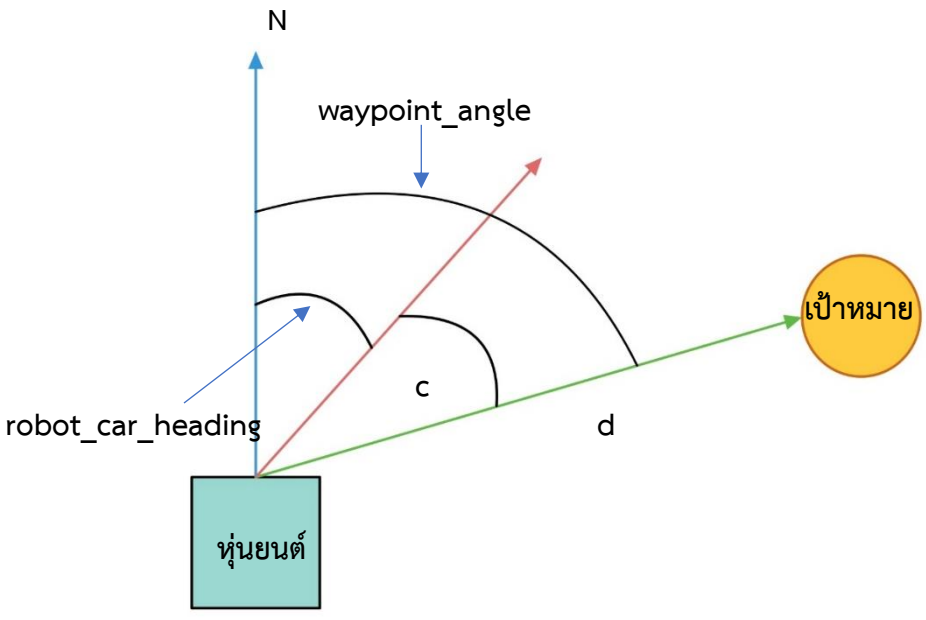
2.1 GPS

GPS ย่อมาจาก Global Positioning System หรือระบบระบุตำแหน่งบนพื้นโลก โดยอาศัยดาวเทียม ซึ่งโคจรอยู่รอบโลก ดาวเทียมดวงหนึ่งโคจรรอบโลก 1 รอบ ใช้เวลา 12 ชั่วโมง ระบบ GPS ทำงานโดยการรับสัญญาณจากดาวเทียม ซึ่งประกอบไปด้วยประกอบไปด้วยข้อมูลที่ระบุตำแหน่งและเวลาขณะส่งสัญญาณ เครื่องรับสัญญาณดาวเทียมจะคำนวนระยะทางระหว่างดาวเทียมและเครื่องรับสัญญาณ เพื่อความแม่นยำในการคำนวณตำแหน่ง ต้องใช้ดาวเทียมอย่างน้อย 4 ดวง นอกจากนี้ความแม่นยำของการระบุตำแหน่งนั้นขึ้นอยู่กับตำแหน่งของดาวเทียมแต่ละดวง ความแปรปรวนของชั้นบรรยากาศ และสิ่งแวดล้อมในบริเวณรับสัญญาณ [2][3]

2.2 ส่วนการควบคุมการนำทาง

2.2.1 Navigation vector

เพื่อให้หุ่นยนต์สามารถเคลื่อนที่ไปยังพิกัดที่กำหนดได้ จะใช้ระยะทาง และมุมที่ต้องเคลื่อนที่ในการนำทาง และเพื่อให้รู้ค่าดังกล่าว จะใช้ค่าพารามิเตอร์ 3 ตัวคือ พิกัดเป้าหมาย พิกัดของหุ่นยนต์ และทิศทางของหัวหุ่นยนต์ที่สัมพันธ์กับเหนือ(robot_car_heading) ซึ่งได้จากโมดูลเข็มทิศและโมดูล GPS



รูปที่ 1 การคำนวณทิศทางและมุมสำหรับการนำทาง

ในคำนวณระยะทางระหว่างพิกัดที่หุ่นอยู่ ณ ปัจจุบันกับพิกัดจุดหมาย ใช้พิกัดละติจูด ลองจิจูด ของตำแหน่ง 2 ตำแหน่งในการคำนวณ ใช้ Haversine formular เป็นสมการในการค้นหา จากสมการ

$\Delta\phi = \phi_2 - \phi_1$

$\Delta\lambda = \lambda_2 - \lambda_1$

$$a = sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1) \cdot \cos(\phi_2) \cdot sin^2\left(\frac{\Delta\lambda}{2}\right)$$

$$d = R \cdot 2 \cdot arctan2\left(\sqrt{a}, \sqrt{1-a}\right)$$

เมื่อ $\emptyset_1$ คือละติจูดของหุ่นยนต์

$\emptyset_2$ คือละติจูดของเป้าหมาย

λ_1 คือลองจิจูดของหุ่นยนต์

λ_2 คือลองจิจูดของเป้าหมาย

R คือรัศมีของโลก (6317 km.)

d คือระยะทางระหว่าง GPS 2 จุด

สำหรับการหามุมที่รถต้องหมุนไป(มุม c) เพื่อให้หันไปตรงกับพิกัดเป้าหมายหาได้จากสมการ

c = waypoint_angle – robot_car_heading

robot_car_heading เป็นมุมตามเข็มนาฬิกาที่หุ่นยนต์ทำกับทิศเหนือหาได้จากการใช้โมดูลเข็มทิศ

waypoint_angle เป็นมุมตามเข็มนาฬิกาที่เป้าทำกับทิศเหนือหาได้จาก Azimuth formular จากสมการ

$$c = \text{arctan2}(\sin(\Delta\lambda)\cos(\phi_2), \cos(\phi_1)\sin(\phi_2) - \sin(\phi_1)\cos(\phi_2)\cos(\Delta\lambda))$$

ค่าที่ได้จากสมการมีทั้งบวกและลบขึ้นอยู่กับทิศของเป้าหมาย หากอยู่ฝั่งทิศเหนือจะมีค่าเป็นบวก หากอยู่ฝั่งทิศใต้จะมีค่าเป็นลบ จึงต้องนำมาบวกด้วย 360 เพื่อให้ได้มุมตามเข็มนาฬิกาที่เป้าทำกับทิศเหนือ

2.2.2 Navigation control

ในการควบคุมสำหรับการนำทางนั้น ก่อนอื่นต้องลดความคาดเคลื่อนของทิศทางการมุ่งหน้าไปยังพิกัดเป้าหมาย(มุม c) โดยหันหุ่นยนต์ไปทางขวาหากค่าความคาดเคลื่อนเป็นบวก และหันไปทางซ้ายหากค่าความคาดเคลื่อนเป็นลบ ผู้จัดทำโครงงานได้กำหนดช่วงของความคาดเคลื่อนไว้ที่ ±10 องศา

เนื่องจากเป็นเรื่องยากที่จะลดค่าความคาดเคลื่อนให้เหลือศูนย์ ในขณะเดียวกันก็ต้องลดความคาดเคลื่อนของระยะทางโดยการให้หุ่นยนต์เคลื่อนที่ไปข้างหน้าในทิศทางที่ใกล้เคียงกับพิกัดเป้าหมาย แต่การเคลื่อนที่ไปข้างหน้านั้นจะทำให้ค่าความคาดเคลื่อนของทิศทางการเคลื่อนที่ทีค่าเพิ่มขึ้นทั้งในทางบวกและทางลบ จึงต้องมีการทำซ้ำไปเรื่อยๆในการคำนวณ navigation vector และการควบคุมการนำทางเพื่อลดค่าความคาดเคลื่อนทั้งสองให้เหลือน้อยที่สุดจนกว่าหุ่นยนต์จะเคลื่อนที่ไปยังพิกัดเป้าหมายได้ โดยเมื่อหุ่นยนต์เคลื่อนที่ไปอยู่ในรัศมีประมาณ 1-3 เมตรของพิกัดเป้าหมายจะถือว่าสามารถเคลื่อนที่ไปยังพิกัดเป้าหมายได้สำเร็จ

สำหรับเส้นทางในการเคลื่อนที่ของหุ่นยนต์นั้น จะเป็นเส้นทางตรงไปยังพิกัดเป้าหมาย คือใช้ตำแหน่ง ณ ปัจจุบันของตัวหุ่นยนต์เป็นจุดอ้างอิงในการหาเส้นทางในการเคลื่อนที่ ดังนั้นเมื่อหุ่นยนต์มีการเลี้ยวหรือเคลื่อนที่ออกนอกทิศทางของการเคลื่อนที่ เส้นทางการเคลื่อนที่จะเปลี่ยนไปเสมอ แต่ทุกเส้นทางจะเป็นระยะกระจัดจากหุ่นยนต์ถึงพิกัดเป้าหมาย

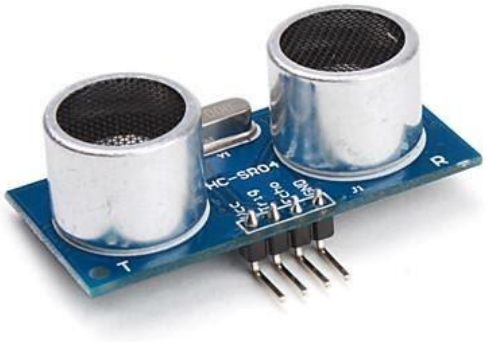
2.3 อุปกรณ์

2.3.1 เครื่องรับสัญญาณ GPS GY-NEO6MV2

โมดูล GPS GY-NEO6MV2 เป็นโมดูล U-blox รุ่น NEO-6M สามารถเชื่อมต่อได้กับไมโครคอนโทรเลอร์หลายหลายประเภทไม่ว่าจะเป็น Arduino, NodeMCU, Raspberry Pi, ESP32 ผ่านทาง Serial UART ความเร็วที่ 9600 (สามารถอัพได้) และตำแหน่งอัพเดทตลอดทุกๆ 1 วินาที สามารถตั้งค่าให้เร็วกว่านี้ได้ การทำงานเมื่อตัวโมดูลจับสัญญาณได้จะขึ้นไฟสีเขียวกระพริบ ตัวโมดูลมีแบตเตอรี่เก็บตำแหน่งล่าสุดและคอนฟิกต่างๆ โดยจะมีความแม่นยำของพิกัดอยู่ที่ 2.5 CEP หรือก็คือภายในรัศมี 2.5 เมตร [4]

รูปที่ 2 เครื่องรับสัญญาณ GPS GY-NEO6MV2 และเสาอากาศ
ที่มา http://www.iot.codemobiles.com/product/296/gy-neo6mv2-gps-module-neo6mv2-with-gps-antenna

2.3.1 Ultrasonic Sensor HC-SR04



รูปที่ 3 Ultrasonic Sensor HC-SR04
ที่มา https://www.arduitronics.com/product/20/ultrasonic-sensor-module-hc-sr04-5v

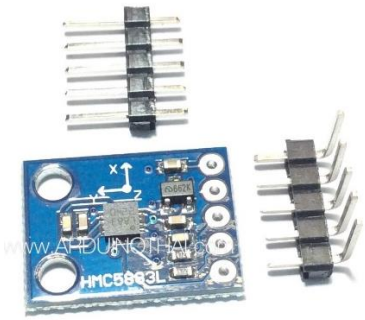
เป็นโมดูลที่ใช้หาระยะห่างระหว่างวัตถุกับเซ็นเซอร์ โดยส่งสัญญาณอัลตร้าโซนิค ความถี่ 40 kHz ไปที่วัตถุที่ต้องการวัดและรับสัญญาณที่สะท้อนกลับมาพร้อมทั้งจับเวลาเพื่อนำมาใช้ในการคำนวณระยะทาง [5]

2.3.1 Shinano Kenshi DCG-5216-038

รูปที่ 4 Shinano Kenshi DCG-5216-038

เป็นมอเตอร์กระแสตรง รองรับแรงดันสูงสุดที่ 24 โวลต์ รองรับกระแสที่ 0.34 แอมป์ สูงสุดที่ 4 แอมป์โดยที่ไม่มีโหลด ความเร็วรอบอยู่ที่ 670 rpm และกำลัง 50 วัตต์

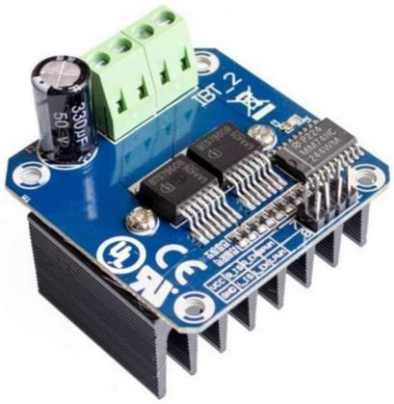
2.3.1 GY-273 3-axis Compass Module (HMC5883L)



รูปที่ 5 GY-273 3-axis Compass Module (HMC5883L)
ที่มา https://www.analogread.com/product/446/gy-273-3-axis-compass-module-hmc5883l

เป็นโมดูลเข็มทิศ ใช้ชิป HMC5883L วัดสนามแม่เหล็ก 3 แกน สามารถประยุกต์ใช้งานเป็นเข็มทิศได้ [6] มีความแม่นยำของเข็มทิศที่ 1-2 องศา [7]

2.3.2 BTS7960 H-Bridge DC Motor Drive (6-27V 47A Max) Module



รูปที่ 6 Motor Drive Module BTS7960

ที่มา https://www.arduino4.com/product/844/btn7960-bts7960-43a-current-limiting-high-power-h-bridge-dc-motor-drive-module-โมดูขับ-motor-dc\

ใช้สำหรับขับดีซีมอเตอร์ที่ต้องการกระแสสูงๆ ใช้สัญญาณ PWM ในการควบคุมความเร็ว รองรับความเร็ว ของ PWM ได้ถึง 25 KHz สามารถควบคุมมอเตอร์ได้ 1 ตัว และควบคุมหมุนซ้าย ขวา (กลับทาง)ได้ แรงดันไฟเลี้ยงมอเตอร์ : 6-27Vdc กระแสเอาต์พุตสูงสูง: 47A Max (กำหนดจากสเปคของ BTS7960) ในทางปฏิบัติควรใช้กระแสไม่เกิน 20A เพื่อความปลอดภัย [8]

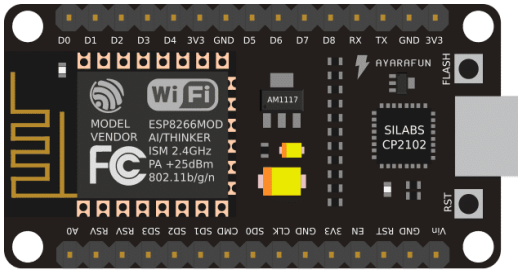
2.3.3 Arduino Mega 2560

รูปที่ 7 Arduino Mega 2560

ที่มา http://www.lungmaker.com/arduino-mega-2560-การใช้งาน/

Arduino MEGA คือบอร์ดรุ่นใหญ่ในกลุ่มบอร์ด Arduino โดยใช้ Atmega2560 เป็นไมโครคอนโทรลเลอร์หลัก โดย Arduino MEGA มี Digital Pins ขา input/output digital จำนวน 54 ขา (เป็น PWM ได้ 15 ขา) มี Analog Input 16 ขา Serial UART 4 ชุด I2C 1 ชุด SPI 1 ชุด และขาแหล่งจ่ายไฟ 5V จำนวน 3 ขา สามารถเขียนโปรแกรมบน Arduino IDE และโปรแกรมผ่าน USB [9]

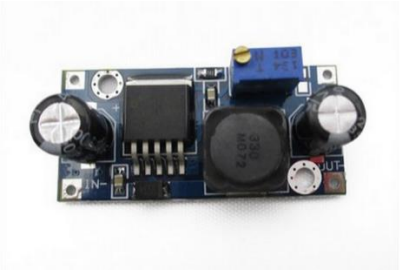
2.3.4 NodeMCU ESP8266



รูปที่ 8 NodeMCU ESP8266
ที่มา https://www.allnewstep.com/article/30/nodemcu-esp8266-esp8285-arduino-1-esp8266-คือ

เป็นชื่อเรียกของชิฟของโมดูล ESP8266 สำหรับติดต่อสื่อสารบนมาตรฐาน WiFi ทำงานที่แรงดันไฟฟ้า 3.0-3.6V ทำงานใช้กระแสโดยเฉลี่ย 80mA ภายในมี Low power MCU 32bit ทำให้เราเขียนโปรแกรมสั่งงานโดยใช้ Arduino IDE ได้ มีวงจร analog digital converter ทำให้สามารถอ่านค่าจาก analog ได้ความละเอียด 10bit [10]

2.3.5 Step down



รูปที่ 9 Step down

ที่มา https://www.analogread.com/category/101/ระบบไฟเลี้ยง /step-down-โมดูลลดแรงดัน

DC-to-DC Step Down LM2596 Module เป็นโมดูลแปลงไฟ DC Step Down (แปลงไฟลง) สามารถปรับค่าได้ตั้งแต่ 1.25-35V โดยปรับค่าที่ Potentiometer โมดูลสามารถจ่ากระแสได้สูงสุด 3A [11]

2.4 เทคโนโลยีที่ใช้

2.4.1 Arduino IDE

Arduino IDE หรือชื่อเต็มคือ Arduino Integrated Development Environment เป็นชุดเครื่องมือสำหรับการเขียนโปรแกรมควบคุม Microcontroller เช่น Text editor สำหรับการเขียน Code ตัว Compiler และอัปโหลดโปรแกรมลงบอร์ด Arduino [12]

แนวคิดการใช้งานโปรแกรม Arduino IDE

1. เขียนโปรแกรมด้วยภาษา C/C++
2. คอมไพล์โปรแกรมภาษา C/C++ เป็นภาษาสำหรับ Microcontroller และบันทึกเป็น Hex file
3. อัปโหลด Intel Hex File ลงบน Microcontroller ผ่านสาย USB

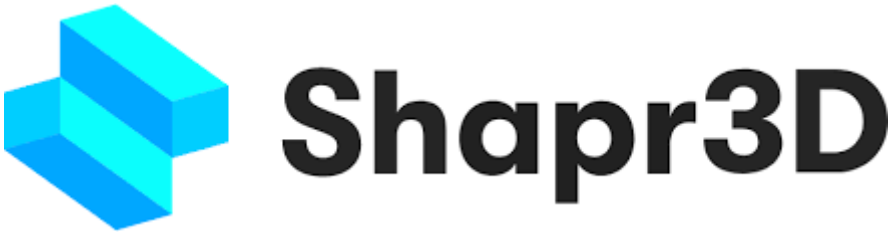
2.4.2 Blynk

รูปที่ 10 Blynk Application

ที่มา https://blynk.io/

Blynk เป็นแพลตฟอร์มแอฟพลิเคชั่นที่ช่วยให้สามารถสร้างอินเตอร์เฟซสำหรับควบคุมและตรวจสอบฮาร์ดแวร์โปรเจคได้จากอุปกรณ์ทั้ง IOS และ Android ได้อย่างรวดเร็ว [13] โดยผู้ใช้สามารถลากและวางเครื่องมือ (widgets) ที่มีให้เลือกใช้อย่างมากมาย Blynk นั้นถูกออกแบบมาสำหรับ Internet Of Things ทำให้สามารถควบคุมฮาร์ดแวร์ได้จากระยะไกล สามารถแสดงข้อมูลเซ็นเซอร์ จัดเก็บข้อมูล แสดงภาพ และทำสิ่งอื่นๆ ได้มากมาย รองรับอุปกรณ์ฮาร์ดแวร์ที่หลากหลาย ไม่ว่าจะเป็น Arduino NodeMCU หรือ Raspberry pi สามารถใช้งานได้ฟรี แต่จะจำกัดเครื่องมือหากต้องการใช้เครื่องมือนอกเหนือจากที่ให้บริการฟรี จะมีค่าใช้จ่ายเพิ่มเติม

2.4.3 Shapr3D



รูปที่ 11 Shapr3D

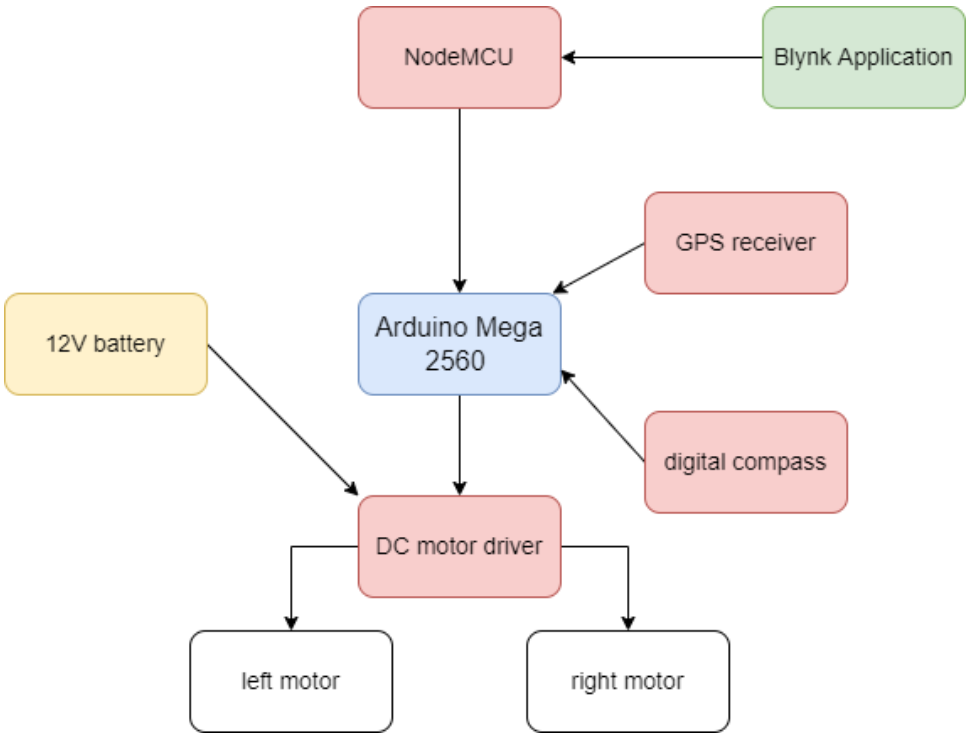
ที่มา https://www.shapr3d.com/

Shapr3D เป็นเครื่องมือการสร้างแบบจำลอง 3 มิติสำหรับการออกแบบทางกลไก สามารถใช้งานได้ทั้ง IOS Android และ Windows บนอุปกรณ์ที่รองรับ

บทที่ 3
รายละเอียดการดำเนินงาน

3.1 ภาพรวมของโครงงาน

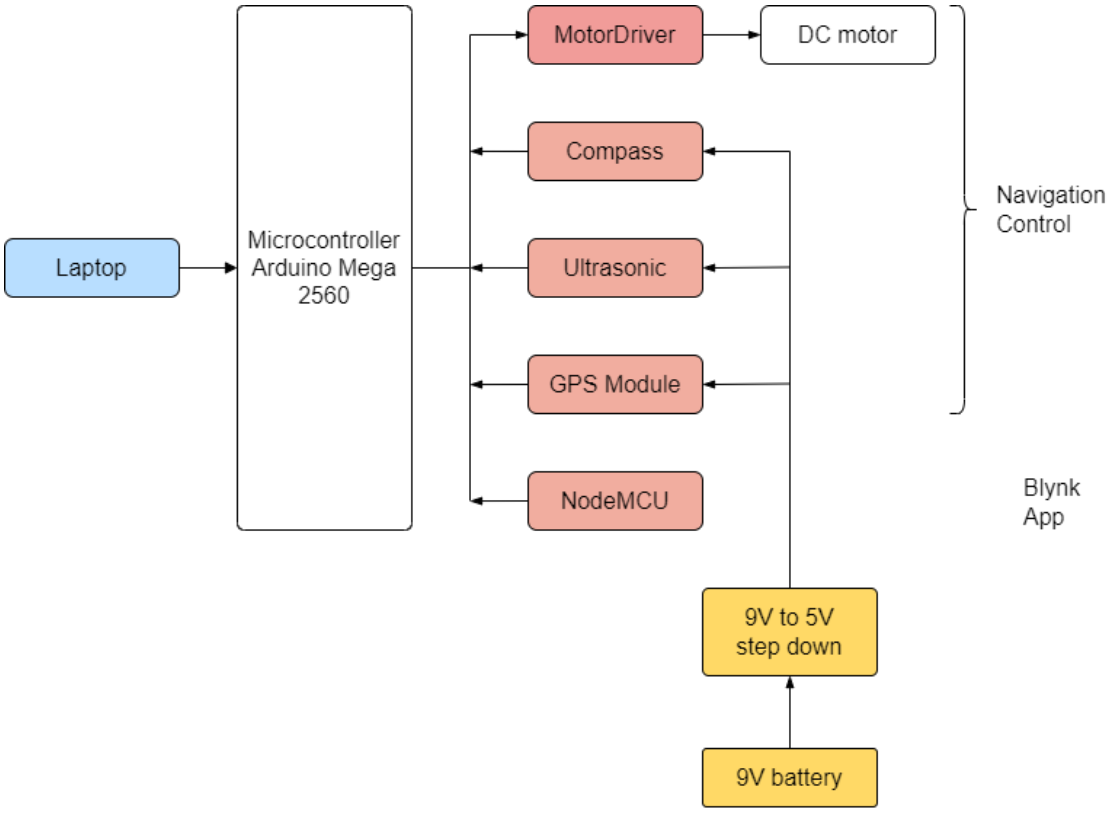
โครงสร้างของโครงงานนี้ประกอบด้วย 2 ส่วน คือ ส่วนที่ 1 ส่วนของฮาร์ดแวร์ ประกอบด้วย โครงสร้างของตัวหุ่นยนต์ Microcontroller การติดตั้งอุปกรณ์ควบคุมและเซ็นเซอร์ได้แก่ เครื่องรับสัญญาณ GPS, NodeMCU ESP8266, Ultrasonic Sensor และ Compass module ส่วนที่ 2 ส่วนของซอฟต์แวร์ ประกอบด้วย แอปพลิเคชันสำหรับกำหนดพิกัดให้กับหุ่นยนต์ โปรแกรมที่เขียนลงบนบอร์ด Arduino สำหรับควบคุมการทำงานของหุ่นยนต์ ได้แก่ การรับค่าจากเซ็นเซอร์ การคำนวณทิศทางหรือองศาการเคลื่อนที่ของหุ่นยนต์ การควบคุมการหมุนของล้อ การป้อนพิกัด GPS ให้แก่หุ่นยนต์ และการรับส่งข้อมูลจากหุ่นยนต์



รูปที่ 12 ภาพรวมของโครงงาน

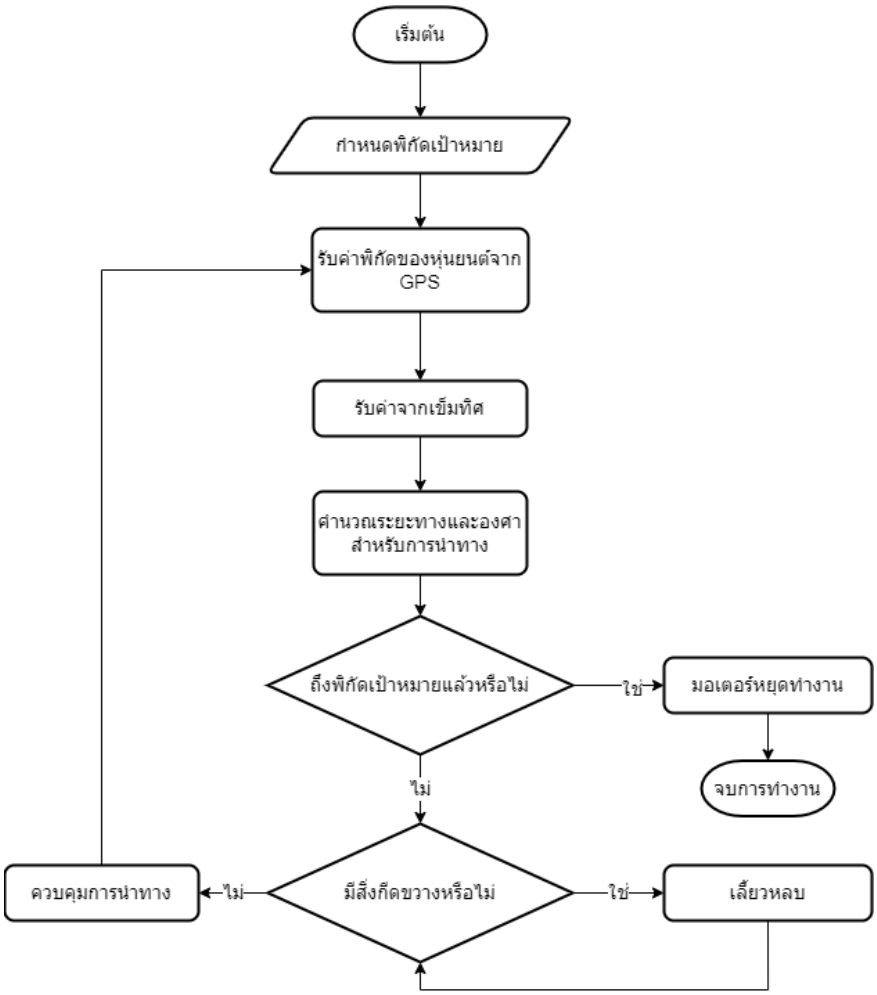
3.2 รายละเอียดของฮาร์ดแวร์

Block Diagram โครงสร้างของหุ่นยนต์



รูปที่ 13 ส่วนประกอบและการเชื่อมต่อของหุ่น

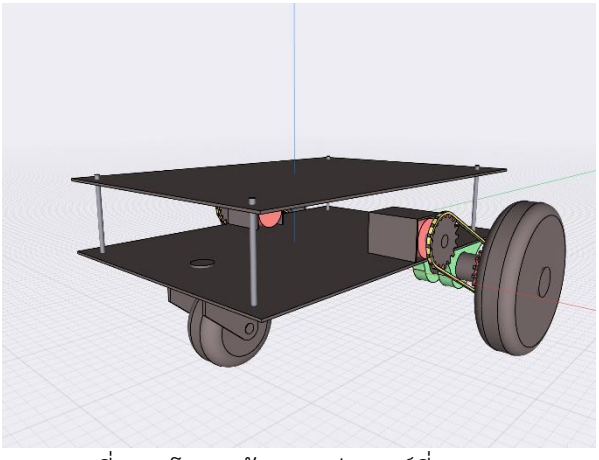
3.3การทำงานของระบบ



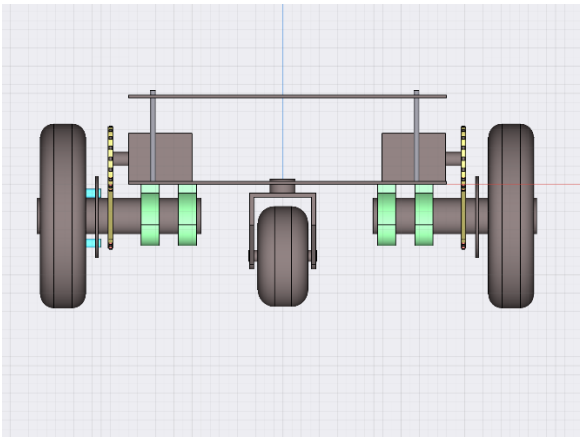
รูปที่ 14 การทำงานของระบบ

3.4 ออกแบบหุ่นยนต์

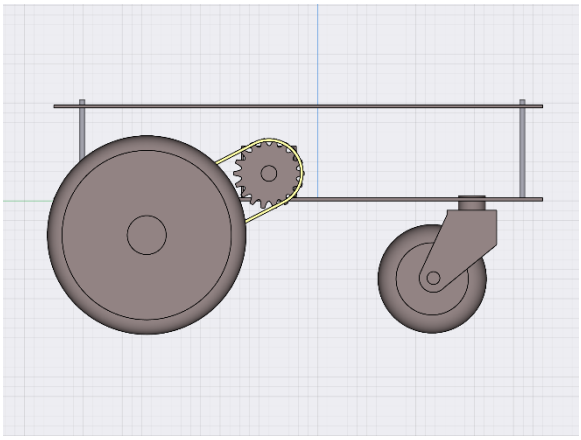
ออกแบบหุ่นยนต์เคลื่อนที่ใหม่ให้มีขนาดที่ใหญ่ขึ้นเพื่อให้มีความเหมาะสมสำหรับการใช้งานภายนอกอาคาร โดยตัวหุ่นจะมีขนาด 35 x 50 เซนติเมตร ใช้ล้อยางตันที่มีขนาดเส้นผ่านศูนย์กลาง 8 นิ้ว ใช้มอเตอร์ SHINANO KENSHI DCG-5216-038 ที่มีสามารถขับแรงดันได้สูงสุด 24vในการขับเคลื่อนล้อ มีล้อหมุนอิสระข้างหน้า 1 ล้อเพื่อค้ำโครงสร้างของตัวหุ่น



รูปที่ 15 โครงสร้างของหุ่นยนต์ที่ออกแบบ



รูปที่ 16 มุมมองด้านหน้าของหุ่นยนต์



รูปที่ 17 มุมมองด้านข้างของหุ่นยนต์

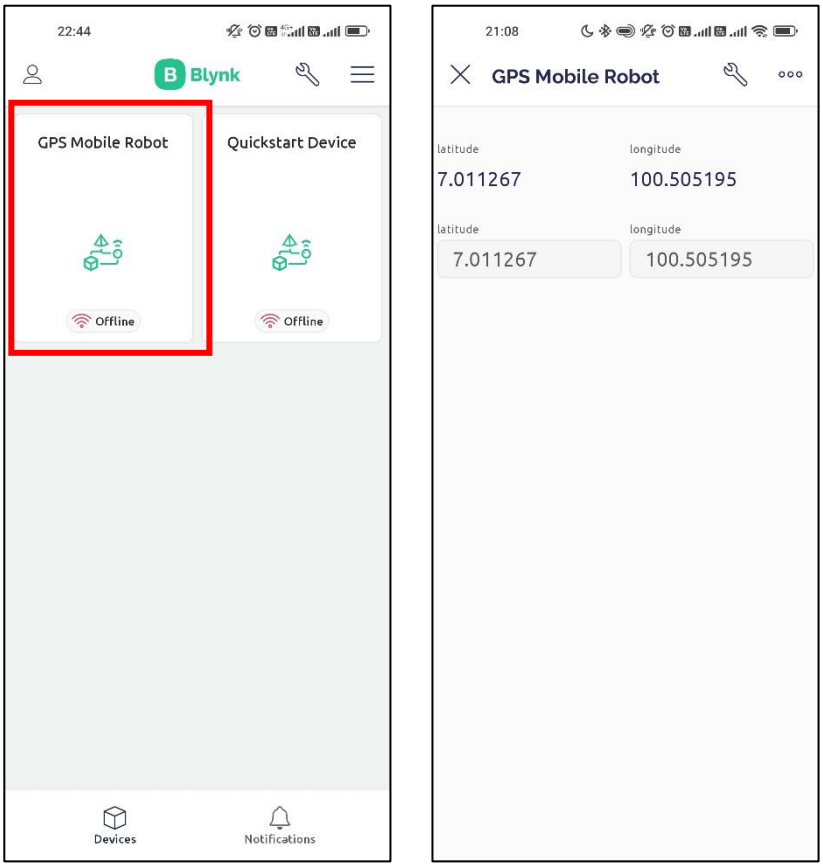
รูปที่ 18 หุ่นยนต์ที่สร้างเสร็จ

บทที่ 4
ความก้าวหน้าการดำเนินงาน

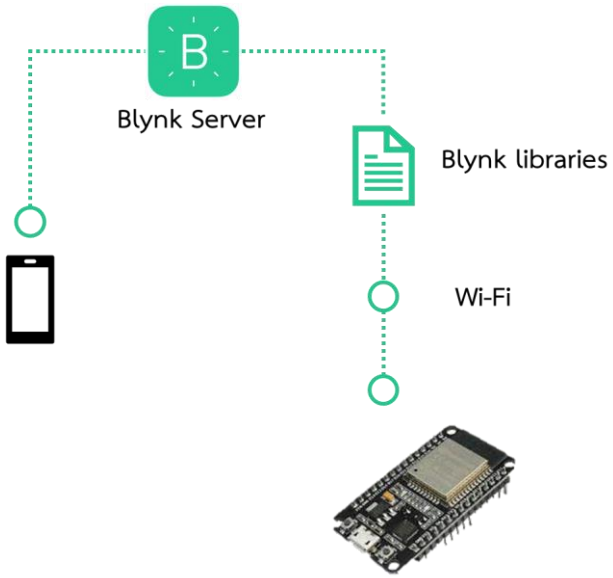
4.1 ความก้าวหน้า 1 พัฒนาอินเตอร์เฟซสำหรับการกำหนดพิกัดเป้าหมายให้หุ่นยนต์

4.1.1 รายละเอียดการพัฒนา

พัฒนาอินเตอร์เฟซสำหรับการกำหนดพิกัดเป้าหมายให้หุ่นยนต์ โดยใช้ Blynk Application ในหน้าอินเตอร์ใช้ widgets ของ Blynk 2 อย่างประกอบด้วย text input 2 ช่องสำหรับป้อนพิกัดละติจูดและลองจิจูด Value Display สำหรับแสดงค่าพิกัดละติจูดและลองจิจูดที่ป้อนลงไป



รูปที่ 19 หน้าอินเตอร์เฟซสำหรับการกำหนดพิกัด



รูปที่ 20 การเชื่อมต่ออุปกรณ์กับ Blynk

4.1.2 อุปสรรคในการพัฒนา

1. wifi ของมหาวิทยาลัยที่เป็นแบบ 802.1x ทำให้ค่อนข้างลำบากต่อการเชื่อมต่อกับเครือข่ายอินเตอร์เน็ต
2. datastrem ตัวแปรประเภท double ของ blynk มีข้อจำกัดในการป้อนตำแหน่งทศนิยมได้เพียง 5 หลักเท่านั้น แต่พิกัดละติจูดและลองจิจูดที่ใช้ในโครงงานนนี้ใช้ทศนิยม 5 ตำแหน่ง

4.1.3 แนวทางการแก้ปัญหา

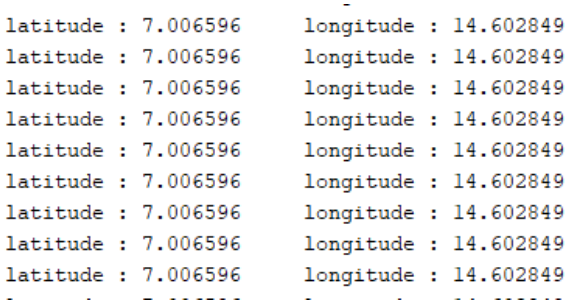
1. เชื่อมต่อกับเครือข่ายอินเตอร์เน็ตโดยใช้ mobile hotspot
2. เปลี่ยนประเภทของตัวแปรที่ใช้ในการป้อนข้อมูลใน Blynk มาเป็น string จากนั้นแปลงเป็น float แล้วจึงส่งข้อมูลไปยัง Arduino

4.2 ความก้าวหน้า 2 การส่งข้อมูลระหว่าง Arduino และ Nodemcu

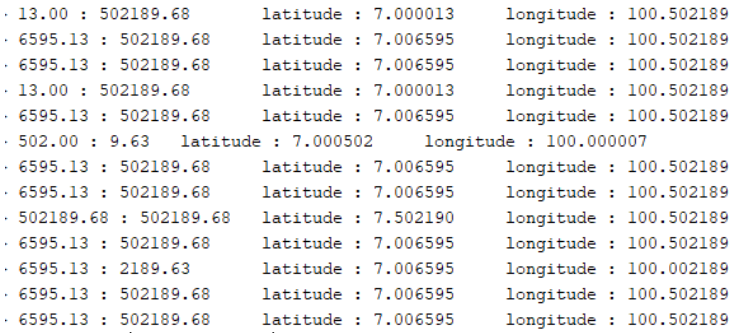
4.2.1 รายละเอียดการพัฒนา

เชื่อมต่อ Arduino mega 2560 พิน 14(TX) และ 15(RX) เข้ากับ Nodemcu ขาD2(RX) และ D1(TX) ตามลำดับ การรับส่งข้อมูลจะใช้การสื่อสารแบบอนุกรม(Serial) โดยที่ Nodemcu จะเป็นตัวส่งข้อมูลพิกัดที่ป้อนผ่าน Blynk application มายัง Arduino

เนื่องจากข้อจำกัดของตัวแปรประเภท float ที่ใช้ในการเก็บค่าพิกัดละติจูดและลองจิจูด ในเรื่องของขนาดของข้อมูลและความแม่นยำของตำแหน่งทศนิยม ก่อนการส่งข้อมูลจึงคูณค่าพิกัดด้วย 1000000 เพื่อให้ตำแหน่งทศนิยมที่ใช้มาเป็นจำนวณเต็ม จากนั้นเมื่อ Arduino รับข้อมูลมา จะนำข้อมูลนั้นมาหารด้วย 100000 เพื่อให้เหลือทศนิยมตำแหน่งที่ต้องการ

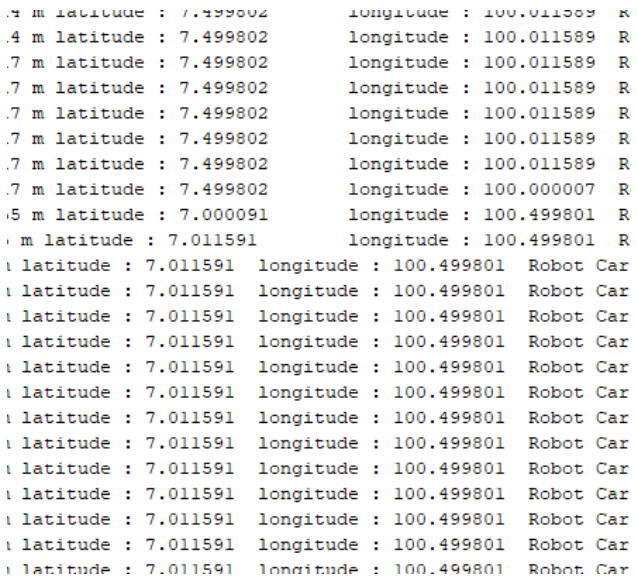


รูปที่ 21 ข้อมูลที่ยังไม่ได้คูณด้วย 1000000 ก่อนการส่ง

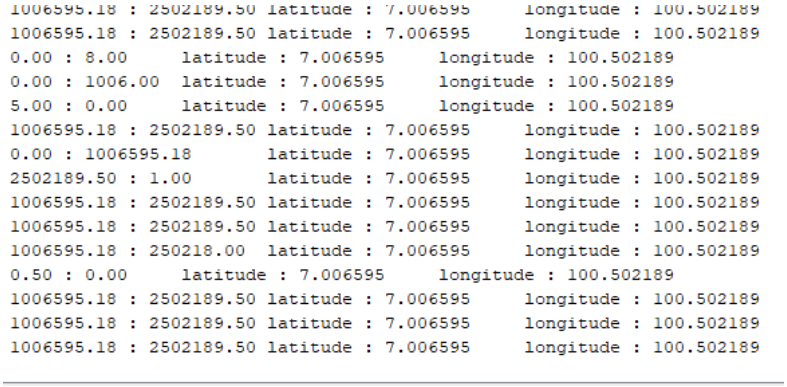


รูปที่ 22 ข้อมูลที่คูณด้วย 1000000 ก่อนการส่งแล้ว

ในการรับส่งข้อมูลนั้น เมื่อรับส่งข้อมูลไปได้ระยะเวลาหนึ่ง พิกัดละติจูดและลองติจูดที่ได้รับจะสลับตำแหน่งกันและค่าพิกัดคาดเคลื่อนไปเยอะมาก จึงต้องใส่ prefix ก่อนการส่งข้อมูล เพื่อใช้ในการตรวจสอบว่า ข้อมูลที่ได้รับมานั้นเป็นพิกัดละติจูดหรือลองจิจูด



รูปที่ 23 การรับส่งค่าพิกัดก่อนใส่ prefix



รูปที่ 24 การรับส่งพิกัดหลังการใส่ prefix

หลังจากใส่ prefix ในการส่งข้อมูลแล้ว เมื่อตรวจสอบแล้วว่าข้อมูลที่รับมาถูกต้อง จะอัพเดทข้อมูลพิกัด หากข้อมูลที่รับมาไม่ถูกต้อง ก็จะไม่มีการเก็บข้อมูลในตัวแปรนั้น

4.2.2 อุปสรรคในการพัฒนา

1. ข้อจำกัดของตัวแปรประเภท float ที่ใช้ในการเก็บค่าในเรื่องของขนาดข้อมูลและความแม่นยำของตำแหน่งทศนิยม
2. พิกัดละติจูดและลองติจูดที่ได้รับจะสลับตำแหน่งกันและค่าพิกัดคาดเคลื่อน

4.2.3 แนวทางการแก้ปัญหา

1. ก่อนการส่งข้อมูลจึงคูณค่าพิกัดด้วย 1000000 เพื่อให้ตำแหน่งทศนิยมที่ใช้มาเป็นจำนวณเต็ม จากนั้นเมื่อ Arduino รับข้อมูลมา จะนำข้อมูลนั้นมาหารด้วย 100000 เพื่อให้เหลือทศนิยมตำแหน่งที่ต้องการ
2. ใส่ prefix ก่อนการส่งข้อมูล เพื่อใช้ในการตรวจสอบว่า ข้อมูลที่ได้รับมานั้นเป็นพิกัดละติจูดหรือลองจิจูด

4.3 ความก้าวหน้า 3 หาค่าความคาดเคลื่อนของพิกัด

4.3.1 รายละเอียดการพัฒนา

ทดสอบโดยการให้หุ่นยนต์เคลื่อนที่ไปยังพิกัดที่กำหนด โดยใช้พิกัดเดิมทุกครั้งในการเคลื่อนที่ บันทึกระยะห่างระหว่างพิกัดเป้าหมายกับพิกัดที่หุ่นยนต์เคลื่อนที่ไปถึง หากอยู่ในระยะ 3 เมตรถือว่าถึงพิกัดเป้าหมายแล้ว (ค่าความแม่นยำของ GPS อยู่ในระยะ 2.5 เมตร)

```
15:22:47.178 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011293 RobotCarLongitude= 100.505187 heading: 185 distance : 2.99 m
15:22:47.300 ->         Brake
15:22:48.170 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011292 RobotCarLongitude= 100.505187 heading: 183 distance : 2.90 m
15:22:48.301 ->         Brake
15:22:49.171 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011291 RobotCarLongitude= 100.505187 heading: 183 distance : 2.81 m
15:22:49.318 ->         Brake
15:43:02.177 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011284 RobotCarLongitude= 100.505187 heading: 169 distance : 2.04 m
15:43:02.301 ->         Brake
15:43:03.170 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011284 RobotCarLongitude= 100.505187 heading: 169 distance : 2.04 m
15:43:03.300 ->         Brake
15:43:04.185 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011285 RobotCarLongitude= 100.505187 heading: 169 distance : 2.09 m
15:43:04.301 ->         Brake
15:52:23.167 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011293 RobotCarLongitude= 100.505187 heading: 178 distance : 2.95 m
15:52:23.303 ->         Brake
15:52:24.183 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011292 RobotCarLongitude= 100.505187 heading: 171 distance : 2.85 m
15:52:24.318 ->         Brake
15:52:25.199 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011291 RobotCarLongitude= 100.505187 heading: 171 distance : 2.81 m
15:52:25.300 ->         Brake
16:09:53.190 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011287 RobotCarLongitude= 100.505187 heading: 178 distance : 2.35 m
16:09:53.307 ->         Brake
16:09:54.177 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011288 RobotCarLongitude= 100.505187 heading: 179 distance : 2.45 m
16:09:54.306 ->         Brake
16:09:55.172 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011289 RobotCarLongitude= 100.505187 heading: 178 distance : 2.54 m
16:09:55.301 ->         Brake
11:16:22.415 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011280 RobotCarLongitude= 100.505180 heading: 131 distance : 2.07 m
11:16:22.529 ->         Brake
11:16:23.420 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011279 RobotCarLongitude= 100.505187 heading: 132 distance : 1.53 m
11:16:23.562 ->         Brake
11:16:24.407 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011279 RobotCarLongitude= 100.505187 heading: 132 distance : 1.45 m
11:16:24.527 ->         Brake
11:32:47.418 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011288 RobotCarLongitude= 100.505180 heading: 101 distance : 2.77 m
11:32:47.536 ->         Brake
11:32:48.417 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011288 RobotCarLongitude= 100.505180 heading: 112 distance : 2.77 m
11:32:48.518 ->         Brake
11:32:49.406 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011289 RobotCarLongitude= 100.505180 heading: 107 distance : 2.81 m
11:32:49.518 ->         Brake
11:45:38.425 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011287 RobotCarLongitude= 100.505180 heading: 113 distance : 2.61 m
11:45:38.508 ->         Brake
11:45:39.405 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011287 RobotCarLongitude= 100.505180 heading: 146 distance : 2.61 m
11:45:39.506 ->         Brake
11:45:40.397 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011287 RobotCarLongitude= 100.505180 heading: 145 distance : 2.69 m
11:45:40.555 ->         Brake
11:57:52.404 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011280 RobotCarLongitude= 100.505180 heading: 102 distance : 2.07 m
11:57:52.535 ->         Brake
11:58:02.390 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011276 RobotCarLongitude= 100.505180 heading: 102 distance : 1.78 m
11:58:02.540 ->         Brake
11:58:03.423 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011276 RobotCarLongitude= 100.505180 heading: 102 distance : 1.78 m
11:58:03.505 ->         Brake
12:10:57.393 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011289 RobotCarLongitude= 100.505187 heading: 136 distance : 2.49 m
12:10:57.535 ->         Brake
12:10:58.396 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011285 RobotCarLongitude= 100.505187 heading: 123 distance : 2.13 m
12:10:58.501 ->         Brake
12:10:59.408 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011285 RobotCarLongitude= 100.505187 heading: 121 distance : 2.13 m
12:10:59.504 ->         Brake
12:23:42.386 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011284 RobotCarLongitude= 100.505180 heading: 173 distance : 2.42 m
12:23:42.551 ->         Brake
12:23:43.391 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011284 RobotCarLongitude= 100.505180 heading: 174 distance : 2.39 m
12:23:43.502 ->         Brake
12:23:44.415 -> lat: 7.011267 long: 100.505195  RobotCarLatitude= 7.011283 RobotCarLongitude= 100.505180 heading: 175 distance : 2.28 m
12:23:44.515 ->         Brake
```

รูปที่ 25 ผลจากการทดลอง 10 ครั้ง

จากการทดลอง ระยะห่างเฉลี่ยระหว่างพิกัดเป้าหมายกับพิกัดที่หุ่นยนต์เคลื่อนที่ไปถึงทั้ง 10 ครั้ง อยู่ที่ 2.390 เมตร

4.3.2 อุปสรรคในการพัฒนา

ในครั้งแรก ได้กำหนดให้ความคาดเคลื่อนของพิกัดที่หุ่นยนต์เคลื่อนที่ไปถึงเป็น 1 เมตรเพื่อจะให้เข้าใกล้กับพิกัดเป้าหมายมากที่สุด ปรากฏว่า เมื่อหุ่นยนต์เคลื่อนที่ไปถึงพิกัดเป้าหมายแล้วช่วงเวลาหนึ่ง ก็จะหาเส้นทางในการเคลื่อนที่ไปยังพิกัดเป้าหมายใหม่อีก เนื่องจากค่าความแม่นยำของ GPS module นั้นอยู่ที่ 2.5 เมตร

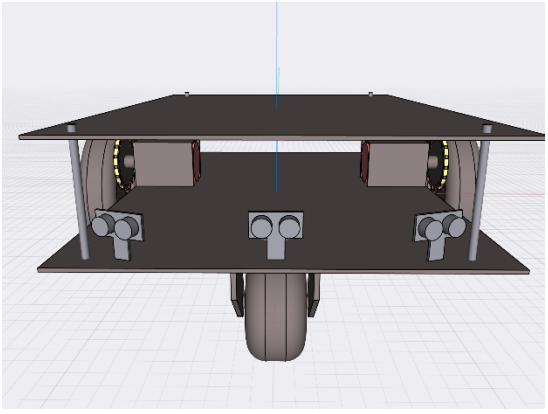
4.3.3 แนวทางการแก้ปัญหา

กำหนดให้ความคาดเคลื่อนของพิกัดที่หุ่นยนต์เคลื่อนที่ไปถึงเป็น 3 เมตร เพื่อให้อยู่ในระยะของพิกัด GPS ที่ทีความแม่นยำภายในระยะ 2.5 เมตร

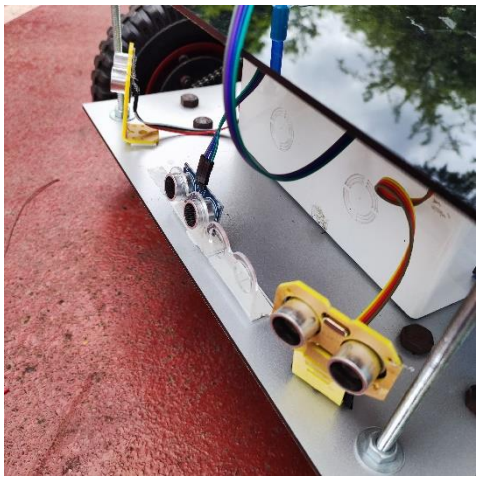
4.4 ความก้วหน้า 4 พัฒนาโปรแกรมเพื่อให้หุ่นยนต์สามารถหลบสิ่งกีดขวางได้

4.4.1 รายละเอียดการพัฒนา

ใช้ ultrasonic sensor 3 ตัวติดตั้งไว้มุมซ้าย ขวา และตรงกลางบริเวณด้านหน้าของหุ่นยนต์ เพื่อใช้ในการตรวจจับวัตถุ หากเซ็นเซอร์ตัวใดตัวหนึ่งตรวจจับเจอวัตถุในระยะ 50 เซนติเมตร จะควบคุมให้หุ่นยนต์เลี้ยวหลบสิ่งกีดขวา ทิศทางของการเลี้ยวนั้นจะขึ้นอยู่กับว่าเซ็นเซอร์ตัวใดตรวจจับเจอวัตถุ หากเซ็นเซอร์ทางขวาตรวจเจอก็จะแล้วซ้าย หากเซ็นเซอร์ตรงกลางหรือทางซ้ายตรวจเจอก็จะเลี้ยวขวา



รูปที่ 26 ภาพจำลองการติดตั้ง ultrasonic sensor



รูปที่ 27 การติดตั้ง ultrasonic sensor

4.4.2 อุปสรรคในการพัฒนา

เนื่องจาก GPS module อัพเดทข้อมูลพิกัดช้ากว่าการตรวจจับวัตถุของ ultrasonic sensor ทำให้การตรวจจับวัตถุช้ากว่าที่ควรจะเป็น เพราะยังติดการทำงานของ GPS module ส่งผลให้หุ่นยนต์หลบสิ่งกีดขวางได้บางครั้ง คือในกรณีที่การอัพเดทพิกัดและตรวจจับเจอวัตถุพอดี

4.4.3 แนวทางการแก้ปัญหา

ใช้ interrupt แทนการใช้ if else เช็คเงื่อนไขการทำงานของ ultrasonic sensor ในการตรวจจับวัตถุ หากเซ็นเซอร์มีการเปลี่ยนแปลง จะขัดจังหวะการทำงานของโปรแกรม ไปทำตามคำสั่งในฟังก์ชันของ interrupt service routine แทน และหากมีวัตถุอยู่ในระยะที่กำหนด จะควบคุมให้หุ่นยนต์เลี้ยวหลบสิ่งกีดขวาง

บทที่ 5
สรุป

5.1 สรุปผลการดำเนินการ

1. พัฒนาอินเตอร์เฟซสำหรับการกำหนดพิกัดเป้าหมายให้หุ่นยนต์
2. การส่งข้อมูลระหว่าง Arduino และ Nodemcu
3. หาค่าความคาดเคลื่อนของพิกัด

5.2 ปัญหาและอุปสรรค

1. Wi-Fi ของมหาวิทยาลัยที่เป็นแบบ 802.1x ทำให้ค่อนข้างลำบากต่อการเชื่อมต่อgกับเครือข่ายอินเตอร์เน็ต
2. datastrem ตัวแปรประเภท double ของ blynk มีข้อจำกัดในการป้อนตำแหน่งทศนิยมได้เพียง 5 หลักเท่านั้น แต่พิกัดละติจูดและลองจิจูดที่ใช้ในโครงงานนนี้ใช้ทศนิยม 5 ตำแหน่ง
3. ข้อจำกัดของตัวแปรประเภท float ที่ใช้ในการเก็บค่าในเรื่องของขนาดข้อมูลและความแม่นยำของตำแหน่งทศนิยม
4. พิกัดละติจูดและลองติจูดที่ได้รับจะสลับตำแหน่งกันและค่าพิกัดคาดเคลื่อน
5. เมื่อหุ่นยนต์เคลื่อนที่ไปถึงพิกัดเป้าหมายแล้วช่วงเวลาหนึ่ง ก็จะหาเส้นทางในการเคลื่อนที่ไปยังพิกัดเป้าหมายใหม่อีก
6. GPS module อัพเดทข้อมูลพิกัดช้ากว่าการตรวจจับวัตถุของ ultrasonic sensor

บรรณานุกรม

[1] Sunrobotics. 2020. BellaBot หุ่นยนต์บริการ (Service Robot) หุ่นยนต์แมวน้อยเสิร์ฟอาหาร ถูกออกแบบมาให้เหมือนแมวน้อยน่ารัก cute สุดๆ. สืบค้นเมื่อ 9 มกราคม 2564, จาก https://www.sunrobotics.co.th/th/articles/202731-bellabot-หุ่นยนต์บริการ-(-service-robot-)-หุ่นยนต์แมวน้อยเสิร์ฟอาหาร-ถูกออกแบบมาให้เหมือนแมวน้อยน่ารัก-cute-สุดๆ

[2] global5thailand. 2006. ความรู้ทั่วไปเกี่ยวกับ GPS. สืบค้นเมื่อ 9 มกราคม 2564, จาก https://global5thailand.com/thai/gps.htm

[3] Garmin ประเทศไทย. (ม.ป.ป). GPS คืออะไร. สืบค้นเมื่อ 9 มกราคม 2564, จาก https://www.garmin.com/th-TH/aboutgps/

[4] iot.codemobile. 2561. GY-NEO6MV2 GPS module NEO6MV2 with GPS Antenna. สืบค้นเมื่อ 15 ก.พ. 2564, จาก http://www.iot.codemobiles.com/product/296/gy-neo6mv2-gps-module-neo6mv2-with-gps-antenna

[5] apiset. 2557. Ultrasonic Sensor Module (HC-SR04) 5V สืบค้นเมื่อ 15 ก.พ. 2564, จาก https://www.arduitronics.com/product/20/ultrasonic-sensor-module-hc-sr04-5v

[6] Analogread. 2564. GY-273 3-axis Compass Module (HMC5883L) สืบค้นเมื่อ 28 ก.พ. 2564, จาก https://www.analogread.com/product/446/gy-273-3-axis-compass-module-hmc5883l

[7] Honeywell. 2563. 3-Axis Digital Compass IC HMC5883L สืบค้นเมื่อ 27 สิงหาคม 2565, จาก https://cdn-shop.adafruit.com/datasheets/HMC5883L_3-Axis_Digital_Compass_IC.pdf

[8] Arduino4. 2564. BTN7960 BTS7960 43A Current Limiting High-Power H-Bridge DC Motor Drive Module โมดูลขับ Motor DC สืบค้นเมื่อ 28 ก.พ. 2564, จาก https://www.arduino4.com/product/844/btn7960-bts7960-43a-current-limiting-high-power-h-bridge-dc-motor-drive-module-โมดูขับ-motor-dc

[9] LungMaker. 2563. การใช้งานบอร์ด Arduino MEGA 2560 เบื้องต้น. สืบค้นเมื่อ 15 ก.พ. 2564, จาก http://www.lungmaker.com/arduino-mega-2560-การใช้งาน/

[10] เจ้าของร้าน. 2561. NodeMCU ESP8266 / ESP8285 Arduino #1 ESP8266 คือ. สืบค้นเมื่อ 15 ก.พ. 2564, จาก https://www.allnewstep.com/article/30/nodemcu-esp8266-esp8285-arduino-1-esp8266-คือ

[11]Analogread, มปป. DC-to-DC Step Down LM2596 Module (3A) โมดูลลดแรงดันไฟฟ้า จ่ายกระแสได้สูงสุด 3A สืบค้นเมื่อ 24 กันยายน 2565, จาก

https://www.analogread.com/product/254/dc-to-dc-step-down-lm2596-module-3a-โมดูลลดแรงดันไฟฟ้า-จ่ายกระแสได้สูงสุด-3a-สต็อกไทยส่งไว

[12] Myarduino. 2563. บทความ Arduino เบื้องต้นแบบละเอียด. สืบค้นเมื่อ 9 มกราคม 2564, จาก https://www.myarduino.net/article

[13] Ermi Media's, Syufrijal and Muhammad Rif'an. 2019. Internet of Things (IoT): BLYNK Framework for Smart Home สืบค้นเมื่อ 17 กันยายน 2565, จาก https://knepublishing.com/index.php/Kne-Social/article/view/4128/8495#info