ตู้กดน้ำอัตโนมัติจ่ายเงินด้วยระบบพร้อมเพย์

Prompt pay-operated drinking water vending machine

กฤตภาส เพ็ชรธนากิจ
Krittapas Phetthanakit

กษิเดช ควรวิไลย
Kasided Kuanwilai

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2561

ตู้กดน้ำอัตโนมัติจ่ายเงินด้วยระบบพร้อมเพย์

Prompt pay-operated drinking water vending machine

โดย

กฤตภาส เพ็ชรธนากิจ

กษิเดช ควรวิไลย

อาจารย์ที่ปรึกษา

อาจารย์ชินภัทร นันทจิวากรชัย

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต
ภาควิชาวิศวกรรมอิเล็กทรอนิกส์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ.2561

ปริญญานิพนธ์ปีการศึกษา 2561

ภาควิชา วิศวกรรมอิเล็กทรอนิกส์

คณะ วิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง ตู้กดน้ำอัตโนมัติจ่ายเงินด้วยระบบพร้อมเพย์

Prompt pay-operated drinking water vending machine

ผู้จัดทำ นายกฤตภาส เพ็ชรธนากิจ รหัสนักศึกษา 58010031

นายกษิเดช ควรวิไลย รหัสนักศึกษา 58010057

ปริญญานิพนธ์นี้ผ่านการตรวจสอบโดยอาจารย์ที่ปรึกษาแล้ว

(อาจารย์ ชินภัทร นันทจิวากรชัย)

อาจารย์ที่ปรึกษา

หัวข้อปริญญานิพนธ์	ตู้กดน้ำอัตโนมัติจ่ายเงินด้วยระบบพร้อมเพย์	
นักศึกษา	นายกฤตภาส เพ็ชรธนากิจ	รหัสนักศึกษา 58010031
	นายกษิเดช ควรวิไลย	รหัสนักศึกษา 58010057
ปริญญา	วิศวกรรมศาสตร์บัณฑิต	
ภาควิชา	วิศวกรรมอิเล็กทรอนิกส์	
ปีการศึกษา	2561	
อาจารย์ที่ปรึกษาปริญญานิพนธ์	อาจารย์ชินภัทร นันทจิวากรชัย	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้จัดทำขึ้นโดยมีวัตถุประสงค์เพื่ออธิบายวิธีการศึกษาขั้นตอนวิธีการออกแบบและสร้าง ระบบตู้กดน้ำจำลองชำระเงินด้วยระบบพร้อมเพย์ โดยภายในตู้กดน้ำจำลองประกอบด้วยสามส่วนหลักๆ คือ 1.ระบบชำระเงินด้วยพร้อมเพย์ 2.ระบบควบคุมปริมาณน้ำในถังพักน้ำ 3.ระบบจำหน่ายน้ำ ตู้กดน้ำจำลองเมื่อได้รับคำสั่งจากไมโครคอนโทรเลอร์ของระบบจะต้องสามารถจ่ายน้ำในอัตราแลกเปลี่ยนหนึ่งบาทต่อน้ำหนึ่งลิตร โดยการควบคุมการจ่ายน้ำคำนวนจากอัตราที่ปั๊มน้ำสามารถสูบน้ำออกได้ โดยความคลาดเคลื่อนในการจ่ายน้ำไม่เกินร้อยละ 10 จากอัตราแลกเปลี่ยนที่กำหนดไว้

Thesis Title	Prompt pay-operated drinking water vending machine	
Student	Mr.Krittapas Phetthanakit	Student ID 58010031
	Mr.Kasided Kuanwilai	Student ID 58010057
Degree	Bachelor of Engineering	
Program	Electronics Engineering	
Year	2018	
Thesis Advisor	Mr. Chinnapat Nantajiwakornchai	

Abstract

This thesis is study and develop application and model of drinking water vending machine for Prompt pay-operated drinking water vending machine. The vending machine must run by real time operation and support Prompt pay. System in drinking water vending machine divided to three part. 1) Payment system 2) Water-level control system 3) Water distribute system. Exchange rate is one bath per one liter. By controlling the water distribution, calculated from the rate at which the pump can pump water out. Error in the water supply not exceeding 10 percent from the specified exchange rate.

กิตติกรรมประกาศ

ปริญญานิพนธ์นี้จะไม่สามารถเกิดขึ้นได้เลยหากปราศจากคำชี้แนะจากท่านอาจารย์ที่ปรึกษา อาจารย์ชินภัทร นันทจิวากรชัย ในการให้คำแนะนำต่างๆ ชี้แนะถึงจุดบกพร่องและแนวทางในการพัฒนาชิ้นงานในตลอดระยะเวลาที่ผู้จัดทำได้ศึกษาโครงงานชิ้นนี้ โดยคณะผู้จัดทำได้ขอขอบพระคุณเป็นอย่างสูง นอกจากนี่คณะผู้จัดทำยังได้รับกำลังใจจากครอบครัวในการทำโครงงานชิ้นนี้ให้สำเร็จลุล่วงไปได้ด้วยดี

กฤตภาส เพ็ชรธนากิจ

กษิเดช ควรวิไลย

สารบัญ

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูปภาพ

สารบัญรูปภาพ(ต่อ)

รูปภาพที่ หน้า

บทที่ 1

บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ในปัจจุบันระบบการชำระเงินทางอิเล็กทรอนิกส์ และอินเตอร์เน็ตแห่งสรรพสิ่งเป็นสิ่งที่เริ่มเข้ามามีบทบาทในชีวิตประจำวันมากขึ้น ดังนั้นผู้จัดทำจึงคิดที่จะประยุกต์นำระบบชำระเงินทางอิเล็กทรอนิกส์มาประยุกต์ใช้กับอุปกรณ์ต่างๆโดยในที่นี้เราสนใจที่จะประยุกต์ใช้ระบบชำระเงินพร้อมเพย์เข้ากับตู้กดน้ำอัตโนมัติ

1.2 วัตถุประสงค์ของการศึกษา

1.2.1 เพื่อประยุกต์ใช้ระบบชำระเงินทางอิเล็กทรอนิกส์ พร้อมเพย์ เข้ากับระบบจำลองตู้กดน้ำอัตโนมัติ

1.2.2 เพื่อศึกษาการการจ่ายเงินด้วย QR code

1.2.3 เพื่อพัฒนาแอปพลิเคชั่นสำหรับระบบจำลองตู้กดน้ำอัตโนมัติ

1.2.4 เพื่อพัฒนาระบบสั่งการระบบจำลองตู้กดน้ำอัตโนมัติ

1.2.6 เพื่อศึกษาการทดลองความคลาดเคลื่อนในการจ่ายน้ำ เมื่อให้ปั๊มน้ำทำงานด้วยเวลาคงที่

1.2.7 เพื่อศึกษาการสร้าง และแก้ไขปัญหาตู้กดน้ำจำลอง

1.3 สมมุติฐานการศึกษา

แอปพลิเคชั่นสามารถ รับชำระเงินทางอิเล็กทรอนิกส์ด้วยระบบพร้อมเพย์ได้ เมื่อระบบตู้กดน้ำได้รับยืนยันการชำระเงินจากระบบแล้ว จะทำการเก็บข้อมูลยอดชำระเงินมาคำนวนหาปริมาณน้ำที่ต้องจ่ายออก และระกว่างจำหน่ายสามารถพักการจ่ายน้ำได้ โดยปริมาณน้ำที่จำหน่ายออกต้องผิดพลาดไม่เกินร้อยละ 10 จากที่ตั้งเกณฑ์ไว้ นอกจากนี้ภายในตู้กดน้ำจำลองจะต้องมีระบบที่สามารถรักษาระดับน้ำในถังพักน้ำที่กรองมาแล้ว โดยเมื่อระดับน้ำในถังพักต่ำกว่าระดับที่กำหนดจะต้องเปิดโซลินอยด์วาวล์หรือปั๊มน้ำเพื่อรักษาระดับน้ำในถังพักได้

1.4 ขอบเขตของการศึกษา

1.4.1 ตู้กดน้ำจำลองสามารถรับชำระเงินด้วยระบบพร้อมเพย์ได้

1.4.2 การทำงานแบบตามเวลาจริง เมื่อมีการชำระเงิน ระบบตู้กดน้ำต้องสามารถจำหน่ายสินค้าออกไปได้ทันที

1.4.3 ระบบตู้กดน้ำจำลองประกอบด้วยสามส่วนที่ศึกษาคือ ระบบชำระเงิน ระบบจ่ายน้ำ ระบบรักษาระดับน้ำในถังพัก

1.4.4 ในการศึกษาวิจัยนี้ไม่รวมถึงคุณภาพของน้ำ จึงทำการตัดระบบกรองน้ำผ่านไส้กรองและระบบกรองน้ำแบบรีเวอร์สออสโมซิส (Reverse Osmosis)

1.4.5 กำหนดให้อัตราแลกเปลี่ยน 1บาท เท่ากับน้ำ1ลิตร

1.4.6 ตู้กดน้ำจำลองมีความคลาดเคลื่อนในการจ่ายน้ำไม่เกินร้อยละ 10 จากเกณฑ์ที่กำหนด

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ความรู้ความเข้าใจในการพัฒนาแอปพลิเคชั่นบนโทรศัพท์มือถือระบบปฏิบัติการแอนดรอยด์

1.5.2 ความรู้ในเรื่องระบบ API

1.5.3 ความรู้ในการพัฒนาระบบชำระเงินด้วยพร้อมเพย์

1.5.4 ความรู้ในการเขียนโปรแกรมเพื่อประมวลข้อมูลจาก API

1.5.3 ความรู้ในการแก้ไขปัญหาเฉพาะหน้าที่เกิดขึ้นระหว่างทำการทดลอง

บทที่ 2

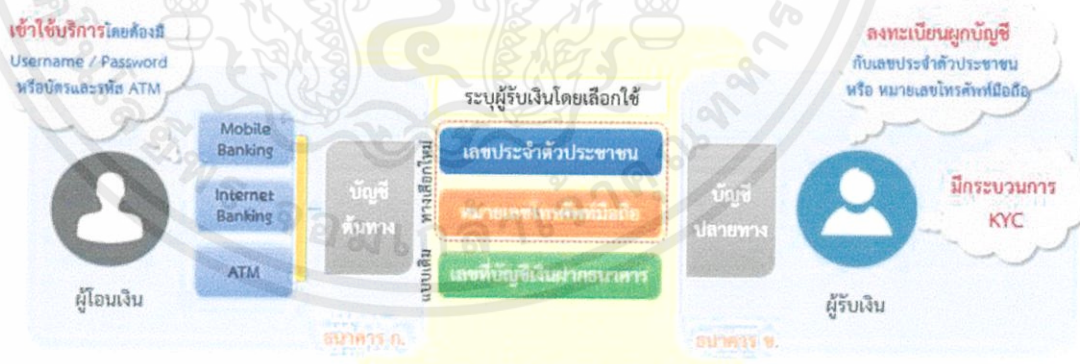
หลักการและทฤษฎี

ในโครงงานนี้ประกอบด้วย ระบบชำระเงินด้วยพร้อมเพย์ และส่วนการแสดงผลและแอปพลิเคชั่น และส่วนควบคุมและประมวลผล ดังนั้นจึงต้องทำความเข้าใจเกี่ยวกับคุณสมบัติและการทำงานต่างๆดังนี้

2.1 พร้อมเพย์ (Prompt Pay)

พร้อมเพย์ เป็นบริการทางเลือกใหม่ให้ประชาชน ธุรกิจ และหน่วยงานต่างๆ ใช้ในการโอนเงินและรับเงิน เป็นบริการเพิ่มจากการโอนเงินแบบเดิม ที่ให้ประชาชนเลือกใช้ได้และทำให้ผู้ใช้มีความสะดวกมากขึ้น เพราะระบบพร้อมเพย์จะใช้เลขประจำตัวประชาชน หรือ หมายเลขโทรศัพท์มือถือของผู้รับเงินแทนได้ ทำให้สะดวกและง่ายต่อการจดจำ จากเดิมที่ต้องรู้เลขที่บัญชีเงินฝากธนาคารจึงจะโอนเงินให้ได้

พร้อมเพย์จึงเป็นบริการทางเลือกให้กับประชาชน โดยทุกคนไม่จำเป็นต้องมาลงทะเบียน แต่ผู้ที่มีการโอนเงินรับเงินบ่อย ๆ ควรมาลงทะเบียนใช้พร้อมเพย์ เพราะจะได้รับประโยชน์จากค่าธรรมเนียมที่ถูกลงมาก



รูปที่ 2.1 แสดงการทำงานของระบบพร้อมเพย์

(ที่มา : ธนาคารแห่งประเทศไทย, (2560), พร้อมเพย์ พร้อมใช้)

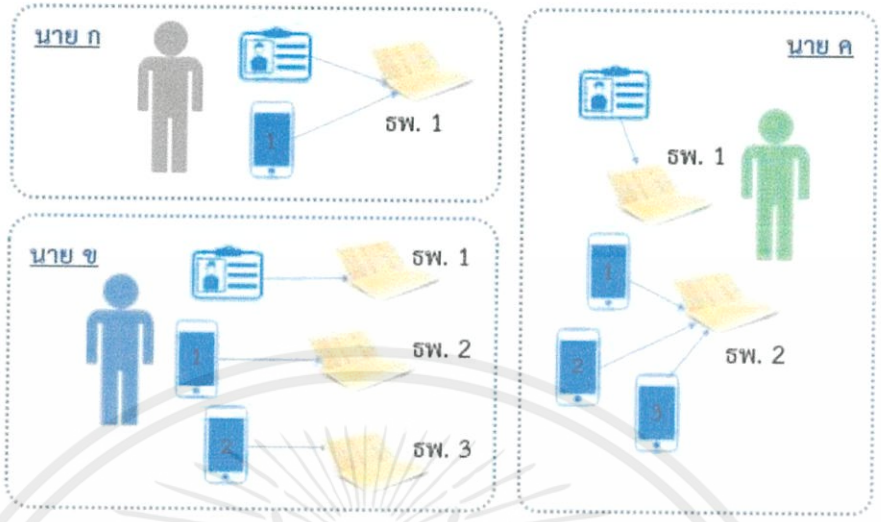
2.1.1 ทำไมต้องมีพร้อมเพย์

เพื่อเพิ่มทางเลือกและความสะดวกในการโอนและรับเงินให้ประชาชน และช่วยลดความเสี่ยงในการถือเงินสดที่อาจสูญหาย ลดภาระและต้นทุนในการบริหารจัดการและพิมพ์ธนบัตรของประเทศ ในระยะต่อไปรัฐบาลจะดูแลประชาชนด้านสวัสดิการโดยจ่ายเงินผ่านช่องทางพร้อมเพย์ด้วย ทำให้ประชาชนได้รับเงินรวดเร็วและทั่วถึง

โดยประชาชนมีทางเลือกในการใช้บริการโอนเงินมากขึ้นและสามารถใช้พร้อมเพย์ในชีวิตประจำวัน ไม่ว่าจะเป็นการโอนเงินค่าเล่าเรียน ค่าบริการต่าง ๆ หรือการจ่ายค่าอาหาร และที่สำคัญด้วยค่าบริการที่ถูกลงกว่าบริการโอนเงินแบบเดิม โดยเฉพาะการโอนเงินครั้งละไม่เกิน 5,000 บาท จะไม่ต้องเสียค่าธรรมเนียมใด ๆ ไม่ว่าจะโอนไปยังธนาคารไหน หรือ ณ จุดใดในประเทศ ถ้าโอนจำนวนมากขึ้นก็มีค่าธรรมเนียมเรียกเก็บบ้าง แต่มีเพดานไม่เกิน 10 บาทต่อรายการ

2.1.2 การดำเนินการเพื่อใช้พร้อมเพย์

ผู้ใช้ต้องมาลงทะเบียนเพื่อผูกบัญชีที่มีอยู่กับธนาคารนั้นเข้ากับเลขประจำตัวประชาชน หรือหมายเลขโทรศัพท์มือถือ โดยธนาคารจะมีการตรวจสอบเอกสารหลักฐานต่าง ๆ ว่าเป็นเจ้าของตัวจริง ไม่มีการปลอมแปลง เพื่อดูแลความปลอดภัยให้ประชาชน และหลังจากลงทะเบียนแล้วธนาคารจะมีการแจ้งยืนยันว่าผ่านการลงทะเบียนหรือไม่ ทุกธนาคารจะเปิดให้ลงทะเบียน ตั้งแต่วันที่ 15 กรกฎาคม 2559 เป็นต้นไป ผ่าน Mobile Banking, Internet Banking ตู้ ATM และสาขาธนาคาร โดยประชาชนสามารถมาลงทะเบียนเมื่อไหร่ก็ได้ ตามความสะดวกและความพร้อมของประชาชน ซึ่งธนาคารบางแห่งที่มีความพร้อมได้เปิดลงทะเบียน ตั้งแต่ 1 กรกฎาคม 2559สำหรับผู้ที่มีบัญชีเงินฝากธนาคารหลายบัญชี ไม่จำเป็นต้องมาลงทะเบียนพร้อมเพย์ทุกบัญชี แต่อาจเลือกบัญชีใดบัญชีหนึ่งไว้สำหรับรับเงินด้วยพร้อมเพย์ก็ได้ (ตัวอย่างกรณีนาย ก) หรือในกรณีที่อยากใช้พร้อมเพย์กับหลายบัญชีก็สามารถเลือกผูกเลขประจำตัวประชาชนกับบัญชีที่ 1 และผูกเบอร์โทรศัพท์มือถือกับบัญชีที่ 2 ก็ได้ (ตัวอย่างกรณีนาย ข และนาย ค) และต้องระวังว่าหมายเลขเดียวกันผูกซ้ำเกินกว่า 1 บัญชีไม่ได้



รูปที่ 2.2 แสดงการผูกบัญชีธนาคารเข้ากับระบบพร้อมเพย์
(ที่มา : ธนาคารแห่งประเทศไทย, (2560), พร้อมเพย์ พร้อมใช้)

เริ่มให้บริการโอนเงินตั้งแต่วันที่ 27 มกราคม 2560 โดยจะโอนเงินผ่านพร้อมเพย์ได้ตามช่องทางต่าง ๆ ของธนาคาร คือ Mobile Banking, Internet Banking และ ตู้ ATM ซึ่งลูกค้าต้องมีรหัส Username / Password หรือมีบัตรและรหัส ATM จึงจะสามารถใช้บริการได้ ระยะเวลาตั้งแต่ลงทะเบียนจนถึงเริ่มให้บริการรับ-โอนเงินระบบพร้อมเพย์จะเป็นช่วงเวลาที่ให้ข้อมูลความรู้แก่ประชาชนเพื่อทำความรู้จัก ศึกษาและเข้าใจวิธีการใช้งานพร้อมเพย์ที่ถูกต้อง รวมทั้งการรักษาความปลอดภัยในการใช้บริการ เพื่อให้มีความพร้อมในการใช้บริการอย่างวางใจและปลอดภัย

2.1.3 ความปลอดภัยของระบบพร้อมเพย์

ระบบพร้อมเพย์มีการดูแลความปลอดภัยตั้งแต่ขั้นตอนการลงทะเบียนที่รัดกุม การพัฒนาระบบกลางที่มั่นคงปลอดภัย และการใช้งานของประชาชนผู้โอนเงินอย่างถูกต้อง

ลงทะเบียนรัดกุม : ธนาคารจะมีการตรวจสอบตัวตนของลูกค้าและความเป็นเจ้าของหมายเลขโทรศัพท์มือถืออย่างรัดกุม นอกจากนี้ ธปท. ได้กำชับให้ธนาคารปฏิบัติตามแนวทางที่กำหนดในการรับลงทะเบียนพร้อมเพย์ ซึ่งได้ออกไปเมื่อวันที่ 29 มิ.ย. 59 เพื่อให้การลงทะเบียนมีความปลอดภัย เป็นมาตรฐานและสร้างความเชื่อมั่นให้กับประชาชน รวมทั้ง ธปท. มีการตรวจสอบขั้นตอนและการควบคุมดูแลระบบการลงทะเบียนพร้อมเพย์เป็นการเฉพาะ เพื่อให้มั่นใจในความมั่นคงปลอดภัยและความพร้อมใช้งาน

ระบบกลางมั่นคงปลอดภัย : พร้อมเพย์เป็นระบบที่พัฒนาเพิ่มจากระบบโอนเงินที่ใช้อยู่ปัจจุบัน จึงมีความปลอดภัยไม่ด้อยกว่าบริการโอนเงินในปัจจุบัน

ธนาคารกับผู้ให้บริการระบบกลางพร้อมเพย์ของประเทศ คือ บริษัท NITMX จึงเป็นระบบปิดที่มีการดูแลรักษาความปลอดภัยตามมาตรฐานสากล คนภายนอกไม่สามารถต่อเข้ากับระบบนี้ผ่านช่องทาง Internet ทั่วไปได้ และ ธปท. ได้ติดตามดูแลการพัฒนาระบบด้านความมั่นคงปลอดภัยด้วย นอกจากนี้ ได้มีการเตรียมการรองรับในด้านความปลอดภัย รวมถึงการดูแลระบบสารสนเทศ (IT) ในเรื่องมาตรฐานความปลอดภัย มาตรฐานความถูกต้อง ความพร้อมใช้ของระบบ การดูแลการเข้าถึงข้อมูล การสำรองข้อมูล และการมีแผนรองรับกรณีฉุกเฉินระบบกลางของพร้อมเพย์ ได้ถูกออกแบบและพัฒนาโดยบริษัทที่เป็นผู้เชี่ยวชาญด้านระบบการชำระเงิน และเป็นผู้พัฒนาระบบการชำระเงินที่ใช้ในประเทศต่าง ๆ โดยมีมาตรฐานความปลอดภัยสูงระดับสากล และยังมีระบบติดตามและป้องกันการทุจริตเพื่อเพิ่มความปลอดภัยด้วย

ทั้งนี้ ผู้ให้บริการระบบกลาง ได้ให้บริการในระบบที่มีมาตรฐานความปลอดภัยด้านเทคโนโลยีและสารสนเทศ ตามมาตรฐาน ISO-27001 ซึ่งเป็นที่ยอมรับในระบบสากล และมีการตรวจสอบประเมินความปลอดภัยจากหน่วยงานภายนอกที่ได้รับการรับรองอีกชั้นหนึ่ง

ประชาชนโอนเงินอย่างถูกต้อง : นอกเหนือจากความปลอดภัยของระบบกลางแล้ว ได้มีการออกแบบความปลอดภัยในส่วนของกระบวนการเข้าใช้งาน กล่าวคือ ผู้ใช้บริการที่ทุกธนาคาร จะต้องมีการใส่เลขรหัสหรือ Password และมีการยืนยันรายการก่อนทำการโอนทุกครั้ง[1]

2.2 คิวอาร์ โค้ด (QR CODE)

รหัสคิวอาร์ หรือคิวอาร์โค้ด (QR Code : Quick Response Code) คือ บาร์โค้ดสองมิติ (Two-Dimensional Bar Code) ชนิดหนึ่ง ที่ถูกพัฒนามาจากบาร์โค้ด (Bar code) ภายใต้แนวคิดเพื่อให้บาร์โค้ดอ่านง่ายและเร็วต่อการตอบสนอง (Quick response) QR Code ถูกพัฒนาขึ้นในปี พ.ศ.2537 โดยบริษัทเดนโซ เวฟ (Denso Wave Incorporated) ประเทศญี่ปุ่น ซึ่งเป็นบริษัทในเครือของโตโยต้า และได้จดทะเบียนลิขสิทธิ์ชื่อ "QR Code" ที่ประเทศญี่ปุ่น และทั่วโลก วัตถุประสงค์หลักในการพัฒนา QR Code คือ เพื่อบริหารจัดการและตรวจสอบข้อมูลชิ้นส่วนอะไหล่ยานพาหนะในกระบวนการผลิต หลังจากนั้นบริษัทเดนโซ เวฟ จึงได้นาเทคโนโลยีนาเสนอต่อสาธารณชน เพราะเล็งเห็นประโยชน์ของ QR Code ที่สามารถเก็บข้อมูลได้หลายประเภทและเก็บข้อมูลได้มากกว่า bar code

รูปที่ 2.3 QR Code (คิวอาร์โค้ด)

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ. (2560). ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

QR Code ประกอบด้วยชิ้นส่วนโมดูลรูปสี่เหลี่ยมสีขาว-ดำ เรียงตัวกัน ในสัณฐานสี่เหลี่ยม สามารถอ่านด้วยการสแกน QR Code ผ่านอุปกรณ์เครื่องอ่าน QR Code หรือผ่านโทรศัพท์สมาร์ทโฟน (Smart phone) ที่มีกล้อง และได้ติดตั้งแอพพลิเคชันสำหรับการถอดรหัส QR Code โดยข้อมูลที่ถูกแปลงเป็นรหัสและถูกจัดเก็บหรือบันทึกอยู่ในสัญลักษณ์ QR Code จะเป็นข้อมูลชนิดตัวอักษร (Characters) หรือตัวเลข (Numeric) ซึ่งสามารถประยุกต์ใช้เพื่อเก็บข้อมูลได้หลากหลาย เช่น เก็บข้อมูลแหล่งของเว็บไซต์ เบอร์โทรศัพท์ ข้อความ และข้อมูลที่เป็นตัวอักษรอื่นๆ ได้หลายรูปแบบขึ้นอยู่กับการประยุกต์ใช้งาน เป็นต้น

2.2.1 ความจุ และเวอร์ชันของ QR Code

QR Code แบ่งเป็นเวอร์ชัน ตั้งแต่ เวอร์ชัน 1 จนถึง เวอร์ชัน 40 ซึ่งแต่ละเวอร์ชันมีความแตกต่างกันในการกำหนดค่าของโมดูล Module configuration) โดยโมดูล คือ จุดสีขาว และสีดา ที่ประกอบกันเป็นสัญลักษณ์ QR Code การกำหนดค่าของโมดูลเป็นการอ้างถึงจานวนของโมดูลที่สามารถบรรจุอยู่ใน QR Code ในแต่ละด้าน ยกตัวอย่าง เช่น เวอร์ชัน 1 (21 x 21 โมดูล) ขยายไปถึงเวอร์ชัน 40 (177 x 177 โมดูล) ซึ่งหมายเลขเวอร์ชันที่สูงขึ้นแต่ละระดับหมายถึงการเพิ่มจานวนโมดูลเข้าไปในแต่ละด้านจานวน 4 โมดูลจากเวอร์ชันก่อนหน้า

รูปที่ 2.4 การเพิ่มจานวนโมดูลในแต่ละเวอร์ชันของ QR Code

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ. (2560). ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

ซึ่งแต่ละเวอร์ชันของ QR Code มีความแตกต่างกัน ในด้านของขนาดความจุข้อมูล ชนิดของข้อมูลที่เก็บ เช่น ชนิดข้อมูลตัวอักษร (Characters) หรือชนิดข้อมูลตัวเลข (Numeric) เป็นต้น และระดับความสามารถในการแก้ไขข้อผิดพลาดและคืนค่าข้อมูล (Error correction level)
ในแง่ของความจุ ถ้าต้องการเก็บข้อมูลปริมาณมากไว้ใน QR Code จะทาให้จำนวนโมดูลที่ประกอบกันเป็นสัญลักษณ์ QR Code มีจานวนมากขึ้น และส่งผลให้ขนาดของสัญลักษณ์ QR Code มีขนาดใหญ่ขึ้นตามไปด้วย

2.2.2 การเลือกใช้ QR Code เวอร์ชัน ให้เหมาะสมกับข้อมูล

การเก็บข้อมูลไว้ใน QR Code ป็นสิ่งที่ต้องคำนึง ถึงอย่างมาก เพราะขนาดของสัญลักษณ์ QR Code จะแปรผันตามขนาดของข้อมูล หากข้อมูลมีขนาดใหญ่ สัญลักษณ์ QR Code จะมีขนาดใหญ่ ในทางกลับกัน หากข้อมูลมีขนาดเล็ก ขนาดของสัญลักษณ์ก็จะเล็กตามไปด้วย แต่ในทางปฏิบัติหากเลือกเวอร์ชันของ QR Code ไม่เหมาะสมกับขนาดของข้อมูล เช่น ต้องการเก็บข้อมูลขนาดเล็ก แต่เลือกใช้ QR Code เวอร์ชันสูงเกินไป จะทำให้ขนาดของสัญลักษณ์ QR Code มีขนาดใหญ่เกินความจำเป็น อาจก่อให้เกิดขนาดรูปทรงที่เทอะทะไม่เหมาะสมกับพื้นที่ที่จะนำไปใช้งาน อีกทั้งอาจจะเกิดข้อผิดพลาดในการอ่านสัญลักษณ์ได้

ตัวอย่างการเลือกใช้เวอร์ชันของ QR Code ให้เหมาะสมกับขนาดข้อมูล

ในกรณีที่ต้องการเก็บข้อมูลชนิดตัวเลข 100 ดิจิต โดยพิจารณาจากตาราง จะมีขั้นตอนดังต่อไปนี้
1) เลือกชนิดข้อมูล (ในกรณีนี้ คือ ตัวเลข (Numeral))
2) เลือกระดับการตรวจสอบข้อผิดพลาด เช่น L M Q และ H (ในกรณีนี้คือระดับ M)
3) ค้นหาข้อมูลในตาราง ที่ใกล้เคียงกับจานวน 100 ในคอลัมน์ Numeric และแถวที่ M จะเกิดจุดตัดในแถวของ QR Code เวอร์ชัน
ดังนั้นเวอร์ชัน QR Code ที่เหมาะสมในการเก็บข้อมูลชนิดตัวเลข 100 ดิจิต คือ เวอร์ชัน 3 (29 x 29 โมดูล)

Version	Modules	ECC Level	Data bits (mixed)	Numeric	Alpha numeric	Binary	Kanji
1	21x21	L	152	41	25	17	10
		M	128	34	20	14	8
		Q	104	27	16	11	7
		H	72	17	10	7	4
2	25x25	L	272	77	47	32	20
		M	224	63	38	26	16
		Q	176	48	29	20	12
		H	128	34	20	14	8
3	29x29	L	440	127	77	53	32
		M	352	101	61	42	26
		Q	272	77	47	32	20
		H	208	58	35	24	15

รูปที่ 2.5 แสดงการเพิ่มจำนวนโมดูลในแต่ละเวอร์ชันของ QR Code

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

ชนิดข้อมูล	ความจุ
ตัวเลขอย่างเดียว	มากสุด 7,089 ดิจิต
ตัวอักษร ผสม ตัวเลข	มากสุด 4,296 ตัวอักษร
ไบนารี (8 บิต)	มากสุด 2,953 ไบต์
คันจิ/คะนะ	มากสุด 1,817 ตัวอักษร

รูปที่ 2.6 แสดงความจุข้อมูลของรหัสคิวอาร์

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

2.2.3 ความสามารถในการแก้ไขข้อผิดพลาดและคืนค่าข้อมูล

QR Code มีความสามารถในการแก้ไขข้อผิดพลาดและคืนค่าข้อมูลที่ถูกต้องได้ ในกรณีที่สัญลักษณ์ QR Code มีคราบสกปรก หรือเกิดความเสียหาย ความสามารถดังกล่าวได้นำ Reed-Solomon Code ซึ่งเป็นหลักการทางคณิตศาสตร์ที่ถูกคิดค้นขึ้นมา เพื่อแก้ไขข้อผิดพลาดในการอ่านข้อมูลจากแผ่นซีดีเพลง และใช้ในการป้องกันแก้ไขสัญญาณรบกวนการสื่อสารผ่านดาวเทียมอีกด้วย ซึ่งเป็นหลักการที่ใกล้เคียงกัน ทำให้ผู้พัฒนานำมาประยุกต์ใช้กับการแก้ไขข้อผิดพลาดและคืนค่าข้อมูลใน QR Code ความสามารถในการแก้ไขข้อผิดพลาด แบ่งออกเป็น 4 ระดับ คือ ระดับ L M H

และ Q ซึ่งมีร้อยละในการคืนค่าข้อมูลที่แตกต่างกัน โดยผู้ใช้สามารถเลือกตามความเหมาะสมของสภาพการใช้งาน การเพิ่มระดับจะเป็นการเพิ่มความสามารถในการแก้ไขข้อผิดพลาด แต่ก็เป็นการเพิ่มจำนวนโมดูลสำหรับการแก้ไขข้อผิดพลาดให้แก่ QR Code ส่งผลทาให้ความจุในการเก็บข้อมูลน้อยลง และขนาดของสัญลักษณ์ก็จะใหญ่ขึ้นตามไปอีกด้วย ปัจจัยต่างๆ ในการเลือกระดับความสามารถในการแก้ไขข้อผิดพลาด เช่น สภาพแวดล้อมการใช้งานและขนาดของสัญลักษณ์ QR Code ยกตัวอย่างเช่น การเลือกระดับ Q (ร้อยละ 25) หรือ H (ร้อยละ30) เหมาะสาหรับสภาพแวดล้อมในโรงงานที่ QR Code อาจมีสิ่งสกปรกไปจับที่พื้นผิวได้ง่าย ในขณะที่ระดับ L เหมาะสาหรับการใช้งานในสภาพแวดล้อมที่สะอาด อีกทั้งยังบันทึกข้อมูลได้มากที่สุด อีกด้วย การใช้งานทั่วไป นิยมเลือกระดับ M (ร้อยละ15) เพราะสัญลักษณ์จะมีขนาดไม่ใหญ่มาก สามารถบันทึกข้อมูลได้มาก และมีระดับความสามารถในการแก้ไขข้อผิดพลาดและคืนค่าได้ในระดับดี

ระดับ	ร้อยละของการคืนค่าข้อมูล
L	ประมาณ 7%
M	ประมาณ 15%
Q	ประมาณ 25%
H	ประมาณ 30%

รูปที่ 2.7 แสดงระดับความสามารถในการแก้ไขข้อผิดพลาดและร้อยละของการคืนค่าข้อมูล
(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

2.2.3 QR Code ประเภทต่างๆ

1.) QR Code โมเดล 1 และ โมเดล 2

โมเดล 1 เป็น QR Code แบบดั้งเดิม มีขนาดใหญ่ ที่สุดเท่ากับ QR Code เวอร์ชัน 14 (73 x 73 โมดูล) สามารถเก็บข้อมูลชนิดตัวเลข ได้สูงสุด 1,167 ดิจิต

รูปที่ 2.8 แสดงรูปQR Code โมเดล 1
(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

โมเดล 2 เป็นการปรับปรุงโครงสร้างมาจาก โมเดล 1 มีขนาดใหญ่ที่สุดเท่ากับ QR Code เวอร์ชัน 40 (177 x 177 โมดูล) สามารถเก็บข้อมูลชนิดตัวเลข ได้สูงสุด 7,089 ดิจิต ซึ่งเป็นโมเดลที่นิยมใช้ในปัจจุบัน

รูปที่ 2.9 แสดงรูปQR Code โมเดล 2

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

2.) ไมโคร QR Code

QR Code ขนาดเล็ก ซึ่งใช้ตาแหน่งในการตรวจสอบรูปแบบ (Position detection pattern) เพียง 1ตำแหน่ง (จากปกติใช้ 3 ตำแหน่ง) เพื่อให้ขนาดของ ไมโคร QR Code มีขนาดเล็กลงกว่า QR Code แบบปกติ ไมโคร QR Code แบ่งออกเป็น 4 เวอร์ชัน ประกอบไปด้วย M1 (11 โมดูล) M2 (13 โมดูล) M3 (15 โมดูล) และ M4 (17 โมดูล) ซึ่งสามารถบันทึกข้อมูลชนิดตัวเลข ได้สูงสุด 35 ดิจิต

Symbol version	Number of modules	Error correction level	Numeric	Alpha numeric	Binary	Kanji
M1	11	-	5	-	-	-
M2	13	L	10	6	-	-
		M	8	5	-	-
M3	15	L	23	14	9	6
		M	18	11	7	4
M4	17	L	35	21	15	9
		M	30	18	13	8
		Q	21	13	9	5

รูปที่ 2.10 ความจุของ ไมโคร QR Code

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

รูปที่ 2.11 แสดงรูปไมโคร QR Code

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

3.) iQR Code

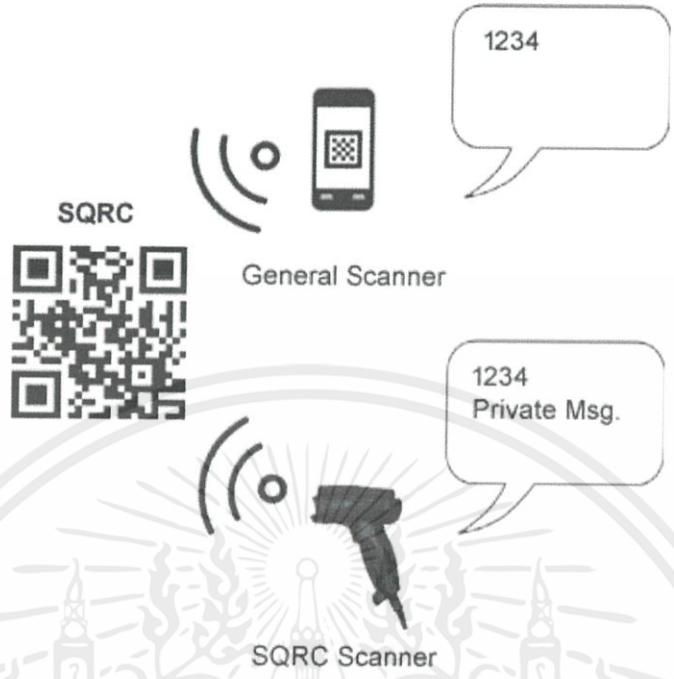
เรียกว่าเป็น QR Code แบบโค้ด เมตริกซ์ 2 มิติ (Matrix-type 2D code) ถูกออกแบบมาเพื่อความง่ายในการอ่าน และมีขนาดเล็ก โดยมี 2 รูปทรง คือ แบบสี่เหลี่ยม และแบบสี่เหลี่ยมผืนผ้า คุณสมบัติในด้านการเก็บข้อมูล เมื่อเปรียบเทียบ iQR Code กับ QR Code ปกติ ที่มีขนาดเท่ากัน พบว่า iQR Code สามารถเก็บข้อมูลได้มากกว่าร้อยละ 80 และด้านการเก็บข้อมูลในปริมาณข้อมูลที่เท่ากัน iQR Code จะมีขนาดเล็กกว่า QR Code ปกติถึงร้อยละ 30 รูปทรงของ iQR Code ที่นิยมนาไปใช้ คือ รูปทรงสี่เหลี่ยมผืนผ้า เพราะสามารถนำไปติดกับป้ายกำกับสินค้าที่มีขนาดเล็ก ซึ่งส่วนมากจะอยู่ในรูปทรงสี่เหลี่ยมผืนผ้าเช่นเดียวกัน ทำให้เกิดความสะดวกในการนำไปใช้งานมากกว่า QR Code รูปทรงสี่เหลี่ยมจัตุรัสที่มีขนาดใหญ่ และใช้พื้นที่ในการพิมพ์ลงบนป้ายกากับสินค้ามากกว่า

รูปที่ 2.12 แสดงรูป iQR Code

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

4.) SQRC

มาจาก Secret-function-equipped QR Code เป็น QR Code ที่ออกแบบมาเพื่อให้มีความสามารถในซ่อนข้อมูล หรือเก็บข้อมูลที่มีความเป็นส่วนตัว ข้อมูลที่เป็นความลับได้ ซึ่งถ้าต้องการที่จะอ่านข้อมูลที่ถูกซ่อนไว้ ต้องใช้เครื่องอ่านพิเศษที่พัฒนาสำหรับ SQRC โดยเฉพาะ จึงจะสามารถอ่านข้อมูลที่ถูกเก็บไว้ในส่วนของความเป็นส่วนตัวได้ ถ้าใช้อุปกรณ์อ่านทั่วไป ก็ยังสามารถอ่านค่าได้แต่จะปรากฏเฉพาะข้อมูลในส่วนที่กาหนดให้เปิดเผยได้เท่านั้น ซึ่งการมองด้วยตาจะไม่สามารถแยกความแตกต่างระหว่าง SQRC กับ QR Code ได้เลย



รูปที่ 2.13 แสดงการอ่านข้อมูลจาก SQRC

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

5.) เฟรมคิวอาร์ (Frame QR)

เป็น QR Code ที่มีส่วนของพื้นที่ว่างตรงกลาง (Canvas area) สาหรับการนารูปต่างๆ เข้ามาใส่ได้ โดยที่ไม่มีผลกระทบต่อข้อมูลที่เก็บใน QR Code เหมาะสาหรับการนาไปใช้เกี่ยวกับการนาเสนอสินค้า ข้อกาหนดในการใช้งานของสินค้า หรืออื่นๆ ตามความต้องการของผู้ใช้

รูปที่ 2.14 แสดงรูปเฟรมคิวอาร์ (Frame QR)

(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

6.) โลโก้คิว (LogoQ)

คือ รูปลักษณ์ใหม่ของ QR Code ที่นาเอารูปภาพที่มีสี ผนวกเข้ากับ QR Code ปกติที่มีสีขาว-ดา โดยสร้างจาก LogoQ algorithm เพื่อให้ได้ LogoQ

รูปที่ 2.15 แสดงรูปการสร้าง LogoQ จาก LogoQ algorithm
(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

7.) QR Code Color multiplexing

เป็นการเพิ่มปริมาณความจุในการเก็บข้อมูลของ QR Code โดยใช้สีอื่นๆ นอกเหนือจากสีขาวและดา ด้วยเทคนิคมัลติเพล็กซ์ชิง นักวิจัยผู้คิดค้นวิธีการนี้ ได้มองเห็นถึงข้อจากัดในการเก็บข้อมูลของ QR Code แบบปกติที่มีเฉพาะสีขาวและดา ซึ่งสีที่มีนัยสาคัญต่อการอ่านข้อมูล คือสีดา ดังนั้น นักวิจัยจึงได้เสนอแนวคิดในการเก็บข้อมูลโดยใช้สีในการเพิ่มปริมาณความจุข้อมูล ข้อมูลจะถูกเก็บแยกออกจากกัน โดยการใช้แม่สี CMYK ซึ่งประกอบไปด้วยสี 4 สี คือ ฟ้า (C : Cyan) ม่วงแดง (M : Magenta) เหลือง (Y : Yellow) และดา (K : Black) และใช้วิธีการอ่านข้อมูลที่เก็บไว้ในแต่ละสี ทาให้สามารถเพิ่มปริมาณความจุข้อมูลได้มากขึ้น[2]

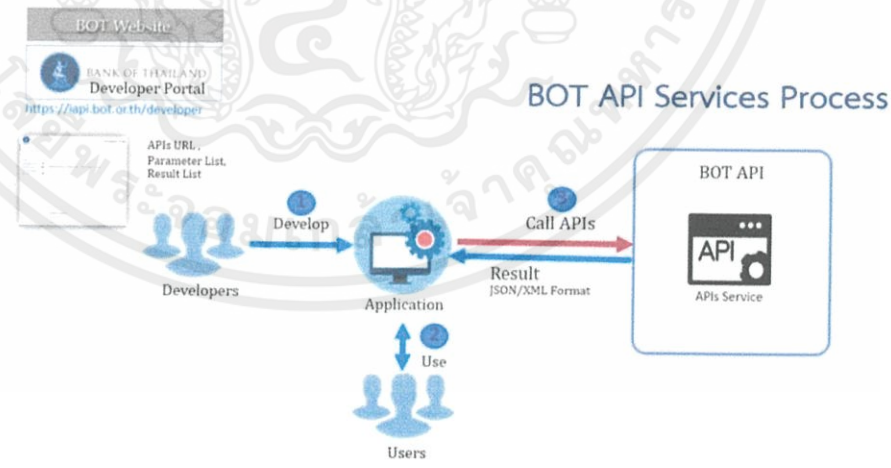
รูปที่ 2.16 แสดงรูป QR Code Color multiplexin
(ที่มา : ณัฐวุฒิ บุญโรจน์วงค์&กชกร พระพรตระการ, (2560), ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code)

2.2.4 ประโยชน์ของคิวอาร์โค้ด

เราสามารถนำคิวอาร์โค้ดมาประยุกต์ใช้งานได้หลากลายรูปแบบ เช่น การเผยแพร่ข้อมูลบนตัวผลิตภัณฑ์ การแสดงข้อมูลช่องทางการติดต่อไม่ว่าจะเป็น เบอร์โทรศัพท์ ชื่อเว็บไซต์หรือ URL ที่ยาวและจดจำได้ยากคิวอาร์โค้ดจึงเป็นช่องทางหนึ่งที่จะช่วยเพื่อความสะดวกให้กับเรามากยิ่งขึ้น เนื่องจากเรางานสามารถนำมือถือที่มีกล้องถ่ายภาพ หรือสมาร์ทโฟน ที่ติดตั้งแอพสำหรับถอดรหัส *QR Code* นำมาสแกนได้เลย จึงหมดปัญหากับการที่จะต้องมานั่งพิมพ์ URL ยาวๆ และจดจำยาก เพียงแค่เรานำโทรศัพท์มาสแกนคิวอาร์โค้ดก็สามารถเข้าชมเว็บไซต์ได้โดยไม่ต้องนั่งพิมพ์ให้เสียเวลา เพราะทุกวันนี้คนส่วนใหญ่นิยมใช้สมาร์ทโฟนกันมากยิ่งขึ้น จึงไม่ใช้ปัญหาสำหรับผู้ที่จะนำ *QR Code* มาประยุกต์ใช้กับงานหรือธุรกิจของตัวเอง[3]

2.3 API (Application Programming Interface)

API หรือ Application Programming Interface เป็นบริการช่องทางการเชื่อมต่อเพื่อแลกเปลี่ยนข้อมูลจากระบบหนึ่งไปสู่ระบบอื่นๆ ที่ สะดวก รวดเร็ว ปลอดภัย หน้าที่หลักของ API คือคอยรับคำสั่งจากฝั่ง Client ซึ่งก็คือ Application ต่างๆ เช่น Web App., Mobile App., Desktop App. เป็นต้น เมื่อฝั่ง Client ส่งคำสั่ง จะเรียกว่าการ Request จากนั้น ตัว API จะรับคำสั่งดังกล่าวนำไปประมวลผล และสรุปเป็นข้อมูลที่ตรงกับ Request และส่งข้อมูลเหล่านั้นกลับไปที่ฝั่ง Client หรือ Application เพื่อนำไปใช้งานต่อไป **[4]**



รูปที่ 2.17 แสดงตัวการทำงานของ BOT APIของธนาคารแห่งประเทศไทย

(ที่มา : ธนาคารแห่งประเทศไทย, (2560), API)

2.4 ปั๊มน้ำ (Water Pump)

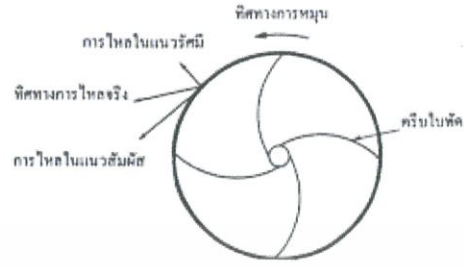
ปั๊มทำหน้าที่ในการสูบของเหลวจากจุดที่มีเฮดกดดันต่ำ(Low pressure head) โดยส่งของเหลวดังกล่าวไปตามระบบท่อ ด้วยเฮดที่ความกดดันสูงกว่าเดิม(High pressure head) การที่จะทำให้ของเหลวไหลจากจุดที่มีความกดดันเฮดต่ำกว่าไปยังจุดที่มีเฮดกดดันสูงกว่านั้น จะต้องใช้ใบพัดของปั๊มทำหน้าที่ในการถ่ายถอดพลังงานกลให้แก่ของไหลนั้นๆ เพื่อที่จะทำให้ของไหลมีพลังงานที่จะใช้ขับเคลื่อนตัวเอง โดยสามารถเอาชนะความต้านทานต่อการไหลภายในระบบนั้น ปั๊มจะสูบของเหลวจากด้านดูด (Suction) และส่งออกไปยังด้านส่ง(Delivery) โดยรับพลังงานจากอุปกรณ์เครื่องต้นกำลัง อาทิ เครื่องยนต์ มอเตอร์ไฟฟ้า เป็นต้นโดยสามารถจำแนกปั๊มตามลักษณะการทำงานได้เป็น 4 ลักษณะ

1) แบบแรงเหวี่ยงหนีศูนย์กลาง (Centrifugal pump)
2) แบบโรตารี่ (ROTARY PUMPS)
3) ปั๊มเลื่อนชักหรือแบบลูกสูบ (Reciprocating pump)
4) แบบพิเศษ (SPECIALIZED PUMPS)

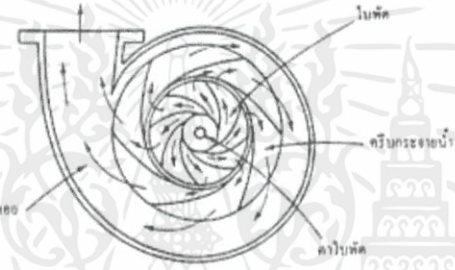
2.4.1 แบบแรงเหวี่ยงหนีศูนย์กลาง (Centrifugal pump)

ปั๊มประเภทนี้นิยมใช้อย่างแพร่หลายในการสูบน้ำ นม สารหล่อลื่น สารละลายเคมี วัสดุทางการเกษตรที่ใช้ในการแปรรูป เป็นต้น มีประสิทธิภาพในการสูบสูงถึง 90 % และยังให้ทำงานที่ระดับความดันสูงได้ ชิ้นส่วนที่หมุนอยู่ภายในเรือนปั๊มเรียกว่าโรเตอร์ (rotor) หรือใบพัด (Impeller) จะเป็นตัวทำให้เกิดการขับดันของไหล ตัวแพร่กระจายน้ำ (Diffuser) เป็นส่วนที่อยู่กับที่ ทำหน้าที่ในการเปลี่ยนเฮดความเร็ว (Velocity head) เป็นความดันสถิตย์ (Static pressure) ของไหลที่ถูกสูบจะไหลผ่านเข้าสู่ช่องทางเข้าซึ่งขนานกับแกนเพลาแล้วถูกเหวี่ยงออกไปตามแนวรัศมีของใบพัดหรือโรเตอร์

กลไกการส่งผ่านพลังงานในโรเตอร์หรือใบพัด เป็นผลจากการเปลี่ยนแปลงโมเมนตัมของของไหลก่อให้เกิดความแตกต่างความดันภายในระบบทำให้เกิดการไหลในแนวเส้นรอบวง (Tangential flow) เป็นผลให้เกิดแรงเหวี่ยงหนีศูนย์กลาง (Centrifugal force) ทำให้เกิดการไหลจากจุดศูนย์กลางของใบพัดของใบพัดออกไปสู่แนวเส้นรอบวงทุกทิศทางออกไปทางท่อส่ง ดังนั้น ของไหลที่ถูกขับดันออกมาก็จะมีทิศทางการไหลที่เกิดจากผลรวมของแรงทั้งสอง ดังรูป



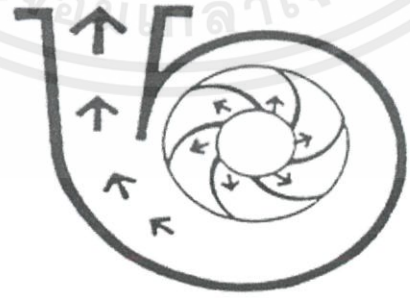
รูปที่ 2.18 แสดงทิศทางการไหลของของไหลขณะผ่านออกจากใบพัดของ Centrifugal pump
(ที่มา : ienergyguru, (2558), ปั๊ม(Pump))



รูปที่ 2.19 แสดงลักษณะทั่วๆไปของ Centrifugal pump
(ที่มา : ienergyguru, (2558), ปั๊ม(Pump))

1) ปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Volute

เป็นปั๊มประเภทแรงดันต่ำ ให้ความดันด้านปล่อยน้อยกว่า 30 เมตรของน้ำ ครีบใบพัดจะหมุนและเหวี่ยงของไหลออกไปสู่ Volute ดังรูป



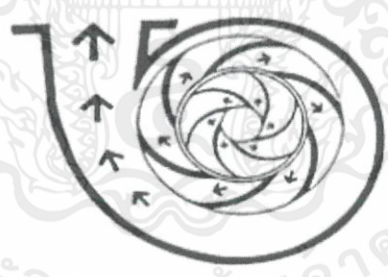
รูปที่ 2.20 แสดงเครื่องสูบแรงเหวี่ยงหนีศูนย์กลางแบบ Volute
(ที่มา : ienergyguru, (2558), ปั๊ม(Pump))

ในกรณีที่ใบพัดหมุนจะเกิดแรงในแนวรัศมีขึ้น ซึ่งจะมีผลกระทำต่อเพลาของใบพัด อาจทำให้เพลาได้รับความเสียหายได้ จึงออกแบบให้เครื่องสูบมีช่องเพิ่มขึ้นเป็นสองช่อง (Double volute) ดังรูป

รูปที่ 2.21 แสดงปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Double volute
(ที่มา : ienergyguru, (2558), ปั๊ม(Pump))

2) ปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Diffuser

เป็นปั๊มประเภทแรงดันปานกลาง (สูงกว่าแบบ Volute) มีลักษณะเหมือนกับปั๊มแบบ Volute แต่จะมีแผ่นกระจายของไหล (Guide vane) ติดอยู่รอบๆเรือนของปั๊มและยังทำหน้าที่ควบคุมทิศทางการไหลของของไหล เพื่อที่จะทำให้เกิดความดันที่สูงขึ้น



รูปที่ 2.22 แสดงปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Diffuser
(ที่มา : ienergyguru, (2558), ปั๊ม(Pump))

3) ปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Regenerative Turbine

เป็นปั๊มประเภทแรงดันสูง ภายในมีชุดใบพัดหลายใบติดอยู่บนเพลาเดียวกัน ใบพัด 1 ชุด เรียกว่า 1 สเตจ ของไหลที่ถูกสูบเมื่อไหลออกมาจากสเตจที่หนึ่งก็จะถูกส่งไปยังสเตจต่อๆไป ทำให้ของไหลมีความดันสูงขึ้น

4) ปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Axial flow

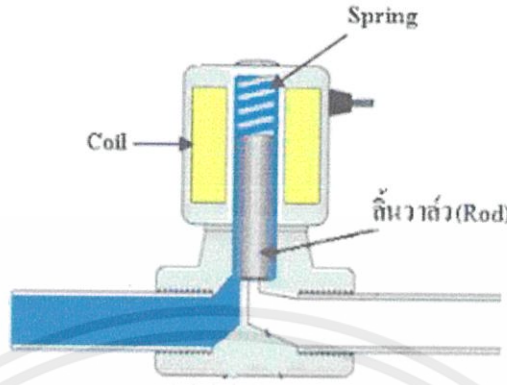
ปั๊มแบบนี้ของไหลจะไหลในแนวแกนเพลา สามารถใช้ได้กับของไหลที่มีสารแขวนลอยปะปนมาด้วย นิยมใช้มากในโรงงานอุตสาหกรรมต่างๆ ซึ่งต้องการเฮดความดันต่ำๆ แต่มีอัตราการไหลสูง

5) ปั๊มแรงเหวี่ยงหนีศูนย์กลางแบบ Mixed flow

ปั๊มแบบนี้จะทำให้การไหล ทั้งในแนวแกนและในแนวรัศมีของใบพัด ซึ่งจะทำให้เกิดแรงในแนวรัศมีและแรงในแนวแกนขึ้น ซึ่งจะช่วยในการขับดันของไหล นิยมใช้กับงานที่ต้องการเฮดความดันต่ำๆ แต่มีอัตราการไหลสูง **[5]**

2.5 โซลินอยด์วาล์ว(Solenoid Valve)

โซลินอยด์ (Solenoid) เป็นอุปกรณ์แม่เหล็กไฟฟ้าชนิดหนึ่ง ที่มีหลักการทำงานคล้ายกับรีเลย์ (Relay) ภายในโครงสร้างของโซลินอยด์จะประกอบด้วยขดลวดที่พันอยู่รอบแท่งเหล็กที่ภายในประกอบด้วยแม่เหล็กชุดบนกับชุดล่าง เมื่อมีกระแสไฟฟ้าไหลผ่านขดลวดที่พันรอบแท่งเหล็ก ทำให้แท่งเหล็กชุดล่างมีอำนาจแม่เหล็กดึงแท่งเหล็กชุดบนลงมาสัมผัสกันทำให้ครบวงจรทำงาน เมื่อวงจรถูกตัดกระแสไฟฟ้าทำให้แท่งเหล็กส่วนล่างหมดอำนาจแม่เหล็ก สปริงก็จะดันแท่งเหล็กส่วนบนกลับสู่ตำแหน่งปกติ จากหลักการดังกล่าวของโซลินอยด์ก็จะนำมาใช้ในการเลื่อนลิ้นวาล์วของระบบนิวแมติกส์ การปิด-เปิดการจ่ายน้ำหรือของเหลวอื่นๆ โครงสร้างของ Solenoid โดยทั่วไปแบ่งออกเป็น 2 ชนิดคือ เลื่อนวาล์วด้วยโซลินอยด์วาล์วกลับด้วยสปริง (Single Solenoid Valve) และเลื่อนวาล์วด้วยโซลินอยด์วาล์วกลับด้วยโซลินอยด์วาล์ว (Double Solenoid Valve) ในที่นี้ใช้แบบ เลื่อนวาล์วด้วยโซลินอยด์วาล์วกลับด้วยสปริง (Single Solenoid Valve) **[6]**

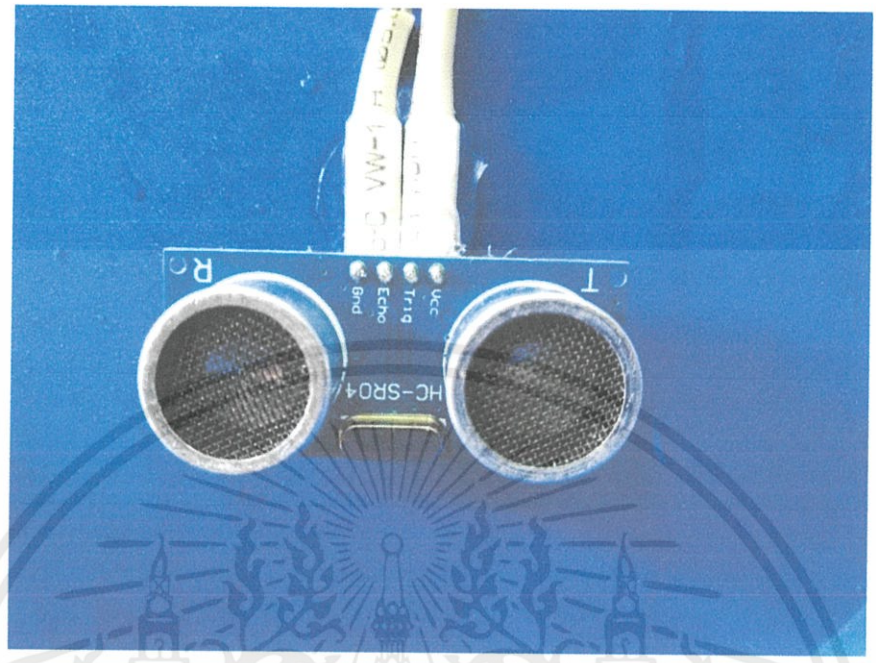


รูปที่ 2.23 แสดงโครงสร้างของโซลินอยด์วาล์ว (Solenoid valve)
(ที่มา : สงกรานต์ สว่างวัล, (2556), โครงงานเครื่องรดน้ำต้นไม้อัตโนมัติ)

2.6 อัลตราโซนิคเซนเซอร์ (Ultrasonic sensor)

เซนเซอร์ชนิดใช้เสียง หรือเซนเซอร์ชนิดอัลตราโซนิก (ultrasonic sensor) เป็นเซนเซอร์ (sensor) ที่ทำงานโดยอาศัยคลื่นเสียงที่มีความถี่สูงกว่า 20 กิโลเฮิร์ต (kHz) ซึ่งเป็นคลื่นในย่านที่มนุษย์ไม่สามารถได้ยินเสียง เซนเซอร์ชนิดอัลตราโซนิกทำงานโดยอาศัยการกระจาย หรือการเคลื่อนที่ของคลื่นเสียงไปกระทบกับพื้นผิวของตัวกลาง ซึ่งอาจเป็นของแข็งหรือของเหลว บางส่วนของคลื่นเสียงจะแทรกผ่านเข้าไปในตัวกลางนั้น และส่วนใหญ่ของคลื่นความถี่สูงนี้จะสะท้อนกลับเรียกว่า "Echo" โดยช่วงเวลาของการสะท้อนกลับของคลื่นเสียงเป็นสัดส่วนโดยตรงกับระยะห่างระหว่างวัตถุกับเซนเซอร์

โดยทั่วไปนิยมใช้อัลตราโซนิกเซนเซอร์สำหรับการวัดระยะทาง (distance measurement) ของวัตถุหรือการวัดระดับ (level measurement) ของเหลว สามารถใช้งานกับวัตถุทั้งชนิดโลหะและอโลหะทุกเฉดสี โปร่งใส โปร่งแสงหรือทึบแสง ตรวจจับวัตถุได้หลายขนาด ไม่เหมาะกับวัตถุที่มีคุณสมบัติการยืดหยุ่นหรือคุณสมบัติการดูดซับเสียง เช่น ผ้า ฝุ่นผง โฟมหรือฟองน้ำ ซึ่งจะดูดซับคลื่นเสียงไม่ให้สะท้อนกลับมายังตัวรับสัญญาณ และเนื่องจากลักษณะการสะท้อนกลับของเสียงขึ้นอยู่กับมุมตกกระทบที่ทำให้เสียงกระจายไปในทิศทางต่าง ๆ จึงไม่เหมาะกับวัตถุที่มีลักษณะเป็นก้อนๆไม่สม่ำเสมอ ผลที่ได้จากการสะท้อนกลับของคลื่นอัลตราโซนิกที่ใช้กับวัตถุลักษณะนี้จะมีความเที่ยงตรง (precision) ต่ำ สำหรับวัตถุที่มีผิวเรียบคลื่นเสียงที่มาตกกระทบส่วนใหญ่จะสะท้อนออกจากพื้นผิวนั้นอย่างมีระเบียบ ค่าความเที่ยงตรงที่ได้จากการวัดจะมีค่าสูงมากกว่า โดยตำแหน่งของเซนเซอร์ที่ตั้งฉากกับพื้นผิวของวัตถุจะให้ประสิทธิภาพในการสะท้อนคลื่นกลับมายังตัวรับมากที่สุด



รูปที่ 2.24 แสดงอัลตราดซนิคเซนเซอร์รุ่น HC-SR04

ในสภาวะแวดล้อมที่มีฝุ่นละอองหรือมีไอน้ำในอากาศ เสียงอาจถูกดูดซึมไปบ้างและสูญเสียพลังงานไปในรูปของพลังงานความร้อน อย่างไรก็ตาม เมื่อเปรียบเทียบกับเซนเซอร์ชนิดแสง (optical sensor/photo sensor) เซนเซอร์ชนิดนี้ได้รับผลกระทบจากละอองไอน้ำที่น้อยกว่า เมื่อพิจารณาอุณหภูมิที่พื้นผิวของวัตถุ พบว่าวัตถุที่มีอุณหภูมิ (temperature) สูงจะทำให้เกิดความผิดเพี้ยนของการวัดขึ้น โดยทำให้ระยะการตรวจจับสั้นลง ผลที่ได้จะไม่แน่นอน เนื่องจากเสียงที่เดินทางผ่านอากาศที่มีอุณหภูมิสูงมีความเร็วสูงกว่าเสียงที่เดินทางผ่านอากาศที่มีอุณหภูมิต่ำกว่า

การติดตั้งเซนเซอร์ชนิดใช้เสียงตั้งแต่ 2 ตัวขึ้นไป ต้องระวังการสอดแทรกหรือการกวนกันของคลื่นเสียงความถี่สูงที่เกิดขึ้นจากเซนเซอร์แต่ละตัว โดยระยะห่างระหว่างตัวเซนเซอร์พิจารณาจากรัศมีของการแผ่กระจายคลื่นความถี่ที่ส่งออกไป และในการติดตั้งเซนเซอร์ต้องระวังมุมอับที่สัญญาณเสียงไม่สามารถเดินทางผ่านไปได้ หรือเรียกว่า บริเวณ "blind zone หรือ dead zone" **[7]**

2.6.1 การคำนวนระยะทางจากอัลตราโซนิคเซนเซอร์

ในการประยุกต์ใช้อัลตราโซนิคเซนเซอร์เราสามารถหาระยะทางได้จากสมการดังต่อไปนี้

$$d = \frac{v \times t}{2} \qquad (2.1)$$

โดยให้ d คือระยะทางระหว่างเซนเซอร์กับวัตถุ

v คือ ความเร็วของคลื่นเสียงในอากาศ ในที่นี้กำหนดให้มีความเร็ว 340m/s

t คือ ระยะเวลาตั้งแต่เซนเซอร์ปล่อยสัญญาณเสียงจนกระทั่งเซนเซอร์ตรวจจับเสียงสะท้อนได้

บทที่ 3

วิธีดำเนินการวิจัย

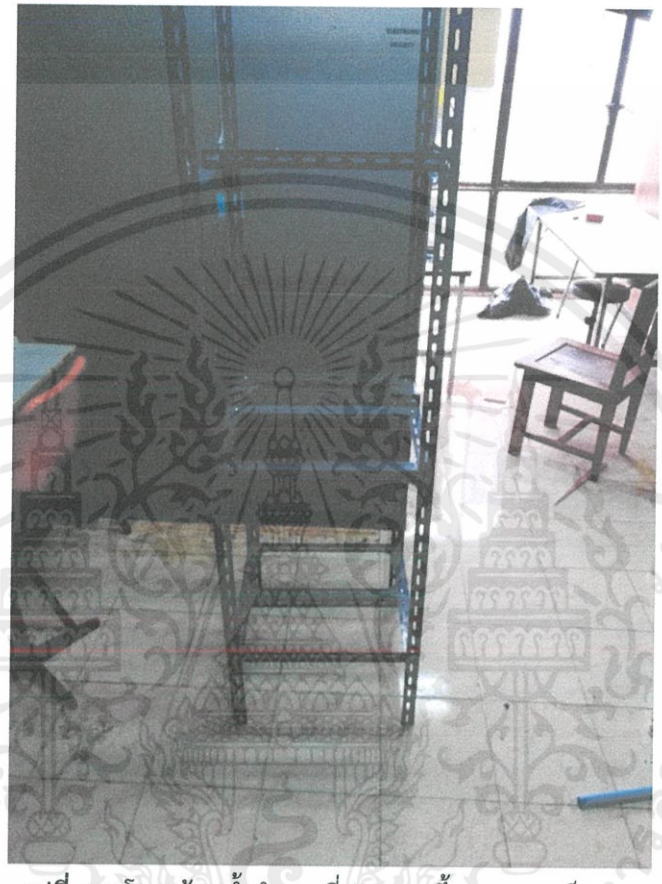
ในการทำชิ้นงานขึ้นมา ผู้จัดทำได้ใช้ไมโครคอนโทรลเลอร์ Wemos UNO+WiFi R3 และสร้างแอปพลิเคชั่นเพื่อแสดง ผลหน้าต่างแสดงผล และ ใช้ API เพื่อเป็นสื่อกลางในการประมวลผลระหว่างเซิพเวอร์และแอปพลิเคชั่น โดยตู้กดน้ำจำลองมีวิธีการดำเนินการดังนี้

3.1 อุปกรณ์ที่ใช้ในการทดลอง

3.1.1	Wemos UNO+WiFi R3 Microcontroller	1 ชิ้น
3.1.2	โทรศัพท์มือถือที่รันด้วยระบบปฏิบัติการแอนดรอย์	1 เครื่อง
3.1.3	Adapter 12V 5A	1 ชิ้น
3.1.4	Switching power supply 12V 3A	1 ชิ้น
3.1.5	เหล็กฉาก ความหนา 1.8 มิลิเมตร ยาว 2 เมตร	4 ชิ้น
3.1.6	เหล็กฉาก ความหนา 1.8 มิลิเมตร ยาว 0.5 เมตร	18 ชิ้น
3.1.7	เหล็กฉาก ความหนา 1.8 มิลิเมตร ยาว 0.25 เมตร	2 ชิ้น
3.1.8	Water pump 12v 240L/H	2 ตัว
3.1.9	Solenoid valve 12V 0.02Mpa-0.8Mpa	1 ตัว
3.1.10	Ultra-sonic sensor HC-SR04	1 ตัว
3.1.11	IRF520N Relay Driver Module	1 ตัว
3.1.12	แผ่นไม้ขนาด 50x50 เซนติเมตร	1 แผ่น
3.1.13	แผ่นไม้ขนาด 50x25 เซนติเมตร	1 แผ่น
3.1.14	ท่อยางเส้นผ่านศูนย์กลาง 5/16” ยาว 3 เมตร	2 ท่อ
3.1.15	ท่อน้ำดื่มเส้นผ่านศูนย์กลาง 1/2”	
3.1.16	สายไฟ หน้าตัดทองแดง 2.5 มิลิเมตร	
3.1.17	สายแพร	
3.1.18	ฟิวเจอร์บอร์ด	

3.2 ขั้นตอนในการประกอบตู้กดน้ำจำลอง

1. นำเหล็กฉากมาประกอบเป็นโครงตู้ โดยให้มีขนาด กว้าง 50 ซม. ยาว 50 ซม. สูง 200 ซม.



รูปที่ 3.1 โครงตู้กดน้ำจำลองที่ประกอบขึ้นมาจากเหล็กฉาก

2. นำแผ่นไม้ มาตัดให้ได้ขนาด 50x25 ซม. และ 50x50 ซม.

3. นำแผ่นไม้ที่ตัดมาพ่นแลกเกอร์เพื่อป้องกันความชื้น

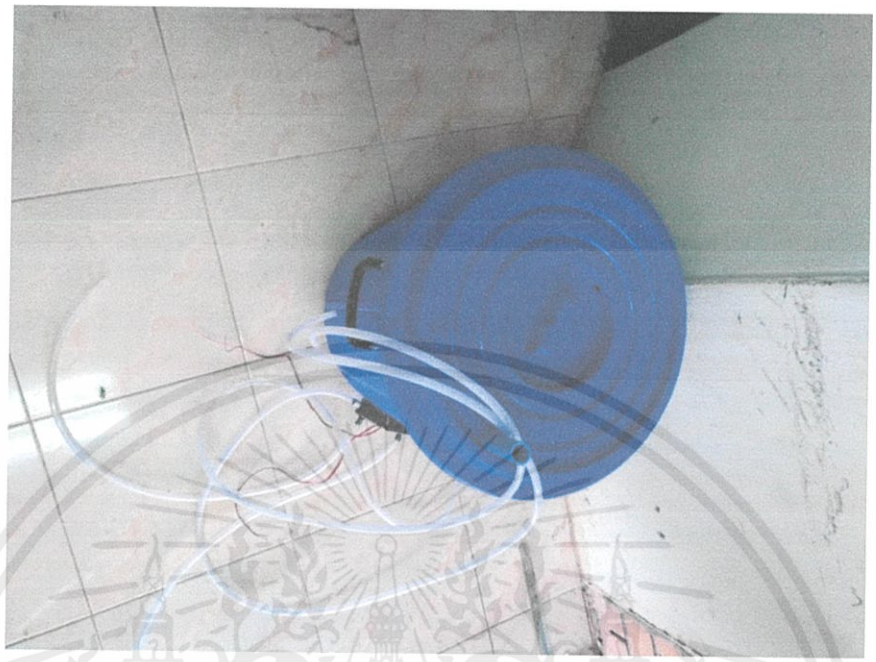
4 นำถังพักน้ำมาเจาะรู สำหรับร้อยสายยางและต่อกับท่อน้ำดื่ม

5. ทำการร้อยสายยางเพื่อเชื่อมต่อกับปั๊มน้ำภายนอก และต่อท่อน้ำดื่มเพื่อเชื่อมต่อกับวาวล์ในขั้นตอนต่อไป

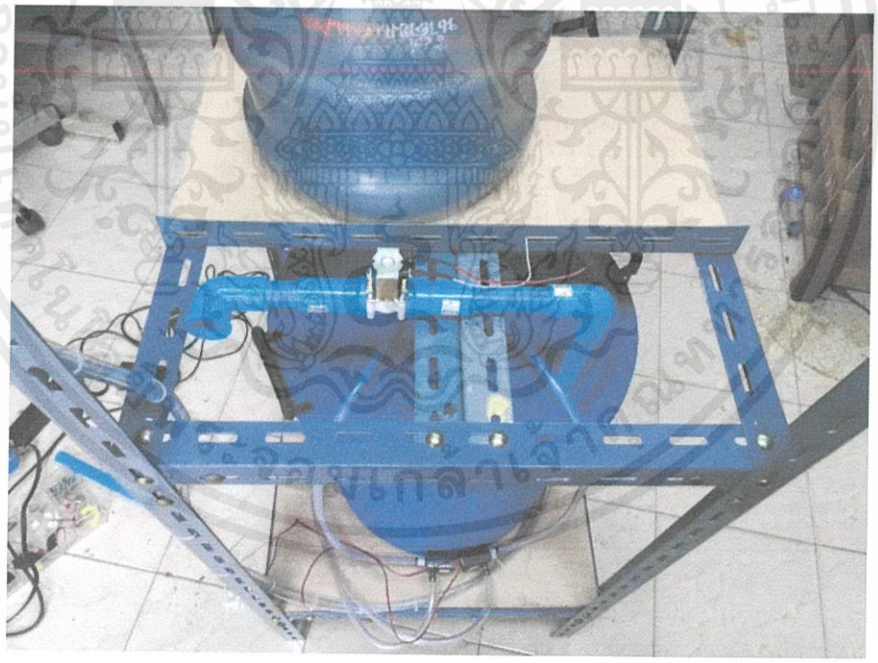
6. นำปั๊มน้ำมาติดตั้งบริเวณด้ายข้างถังพักน้ำ ทำการยึดติดให้เรียบร้อย

7. นำอัลตราโซนิคมาติดตั้งใต้ฝาถังพักน้ำ ปิดฝาถังพักประกอบให้เรียบร้อย

8. นำโซลินอยด์วาวล์มาประกอบกับท่อน้ำดื่มที่เข้าถังพักน้ำ



รูปที่ 3.2 ถังพักน้ำที่เจาะรูและร้อยสายยางกับต่อท่อเรียบร้อยแล้ว



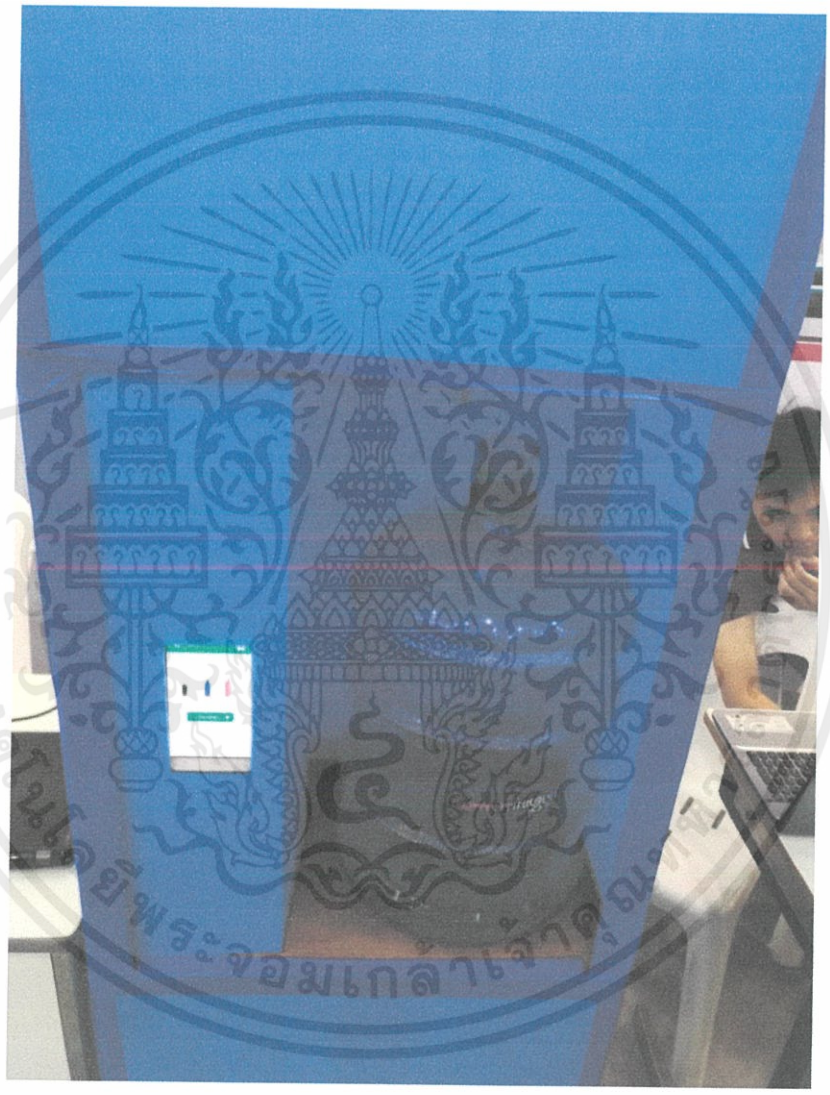
รูปที่ 3.3 ตู้กดน้ำจำลองขณะกำลังดำเนินการเดินสายไฟ

9. นำแผ่นไม้วางบนโครงเหล็กที่ทำไว้ ติดตั้งถังพักน้ำเข้ากับโครงตู้

10. ทำการเดินสายไฟระบบปั๊มน้ำ เซนเซอร์ และโซลินอยด์วาล์วให้เรียบร้อย

11. ทำการเดินสายยางสำหรับจำหน่ายน้ำออก ยึดติดให้เรียบร้อย

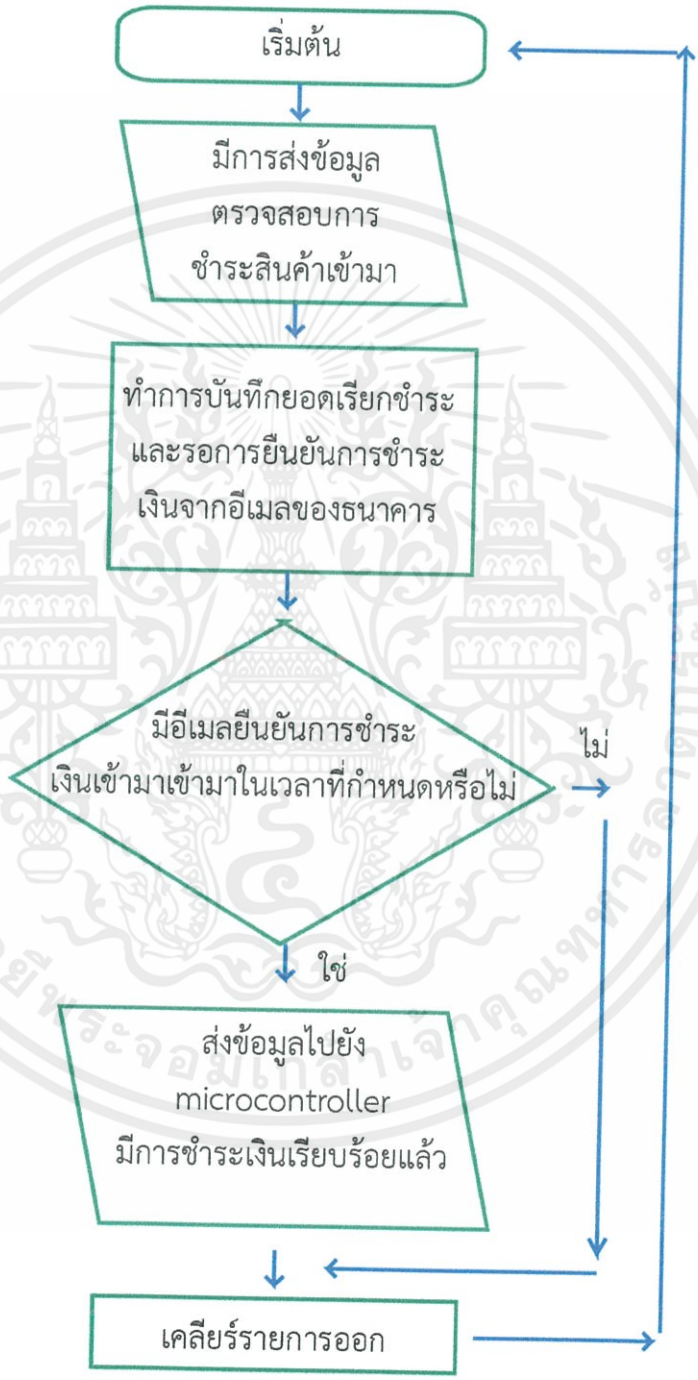
12. ตัดฟิวเจอร์บอร์ดปิดบริเวณด้านข้างให้เรียบร้อย



รูปที่ 3.4 ตู้กดน้ำจำลอง

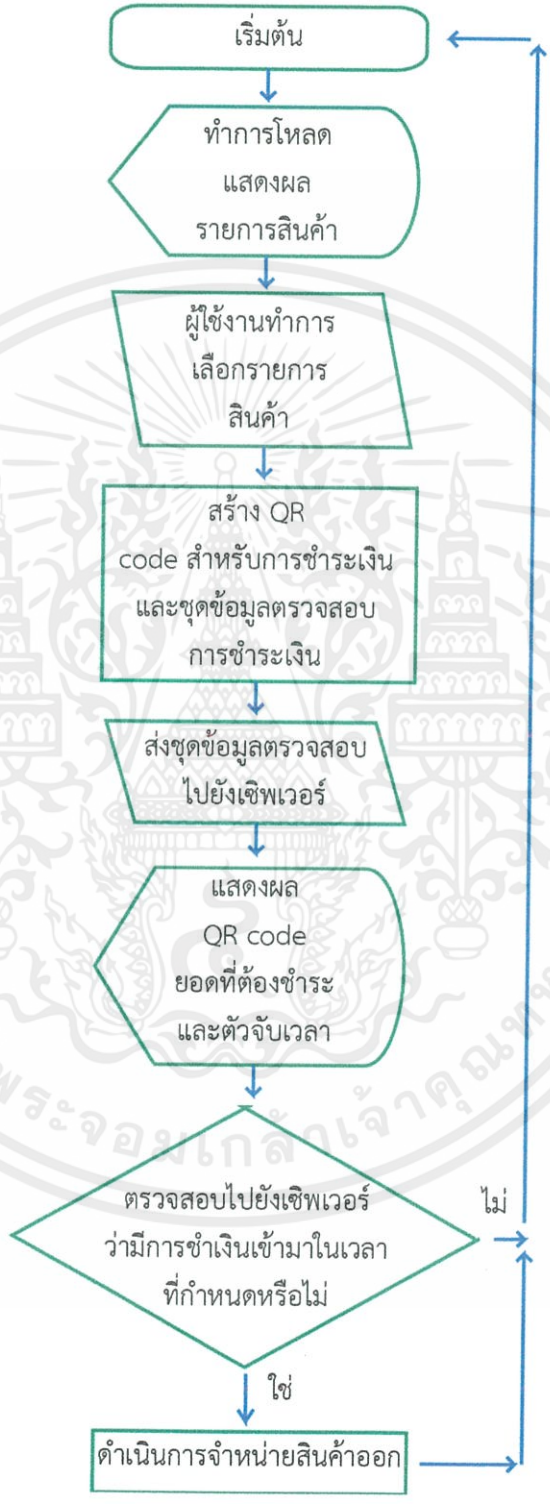
3.2 ลำดับขั้นตอนในการทำงานของโปรแกรม

3.2.1 ลำดับขั้นตอนการทำงานของโปรแกรมของเซิฟเวอร์



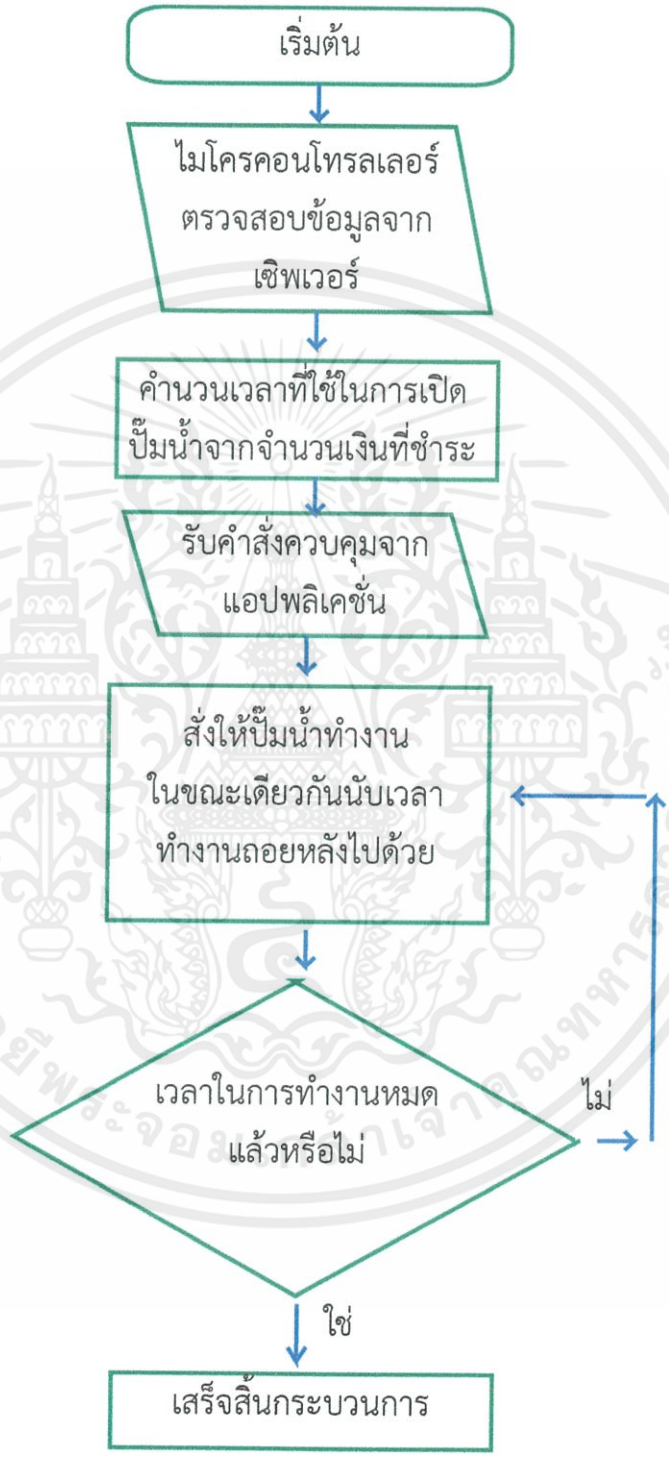
รูปที่ 3.5 แสดงลำดับขั้นตอนการทำงานของโปรแกรมของเซิฟเวอร์

3.2.2 ลำดับขั้นตอนการทำงานของโปรแกรมส่วนแอปพลิเคชั่นและการชำระเงิน



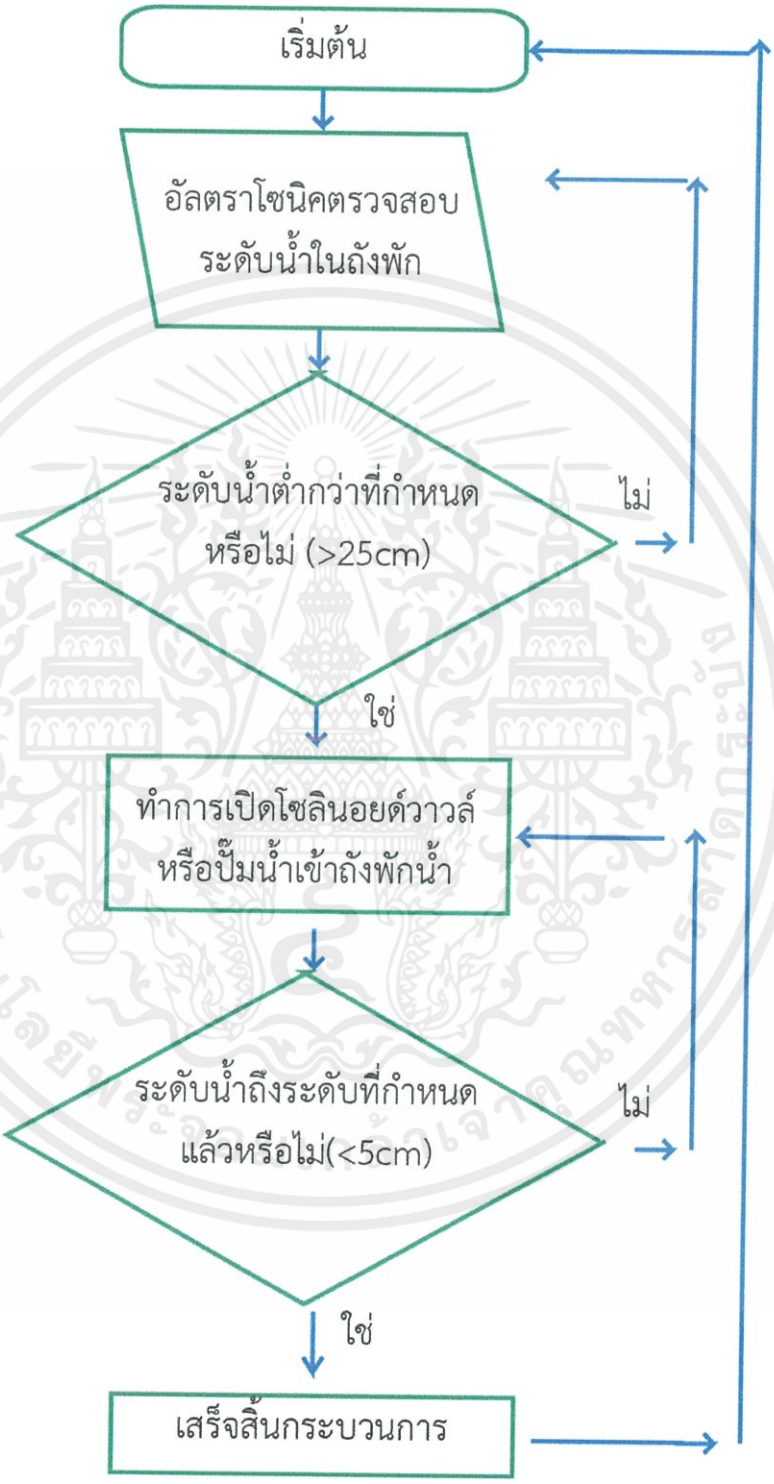
รูปที่ 3.6 แสดงลำดับขั้นตอนการทำงานของโปรแกรมส่วนแอปพลิเคชั่นและการชำระเงิน

3.2.3 ลำดับขั้นตอนการทำงานของโปรแกรมส่วนการจำหน่ายน้ำ



รูปที่ 3.7 แสดงลำดับขั้นตอนการทำงานของโปรแกรมส่วนการจำหน่ายน้ำ

3.2.4 ลำดับขั้นตอนการทำงานของโปรแกรมส่วนรักษาระดับน้ำในถังพัก

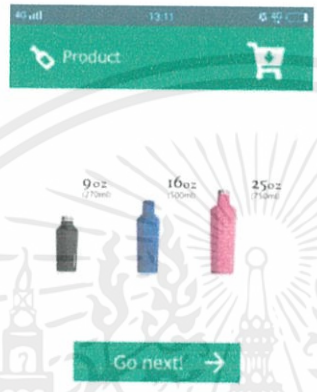


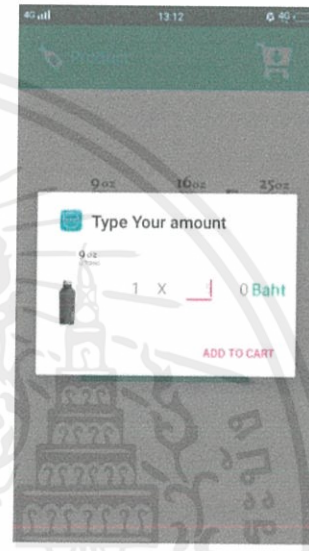
รูปที่ 3.8 แสดงลำดับขั้นตอนการทำงานของโปรแกรมส่วนรักษาระดับน้ำในถังพัก

บทที่ 4

การทดลองและผลการทดลอง

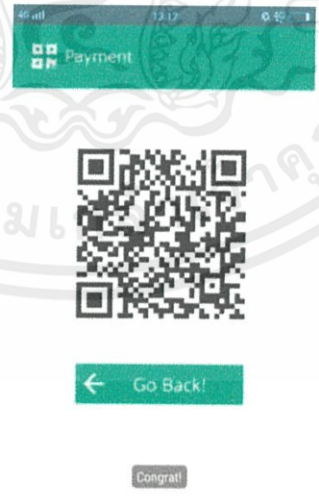
4.1 การแสดงผลติดต่อผู้ใช้





รูปที่ 4.1 (ซ้าย) แสดงผลส่วนติดต่อผู้ใช้ในการเลือกชนิดสินค้า

รูปที่ 4.2 (ขวา) แสดงผลส่วนติดต่อผู้ใช้ในถึงจำนวนเงินที่ต้องชำระ



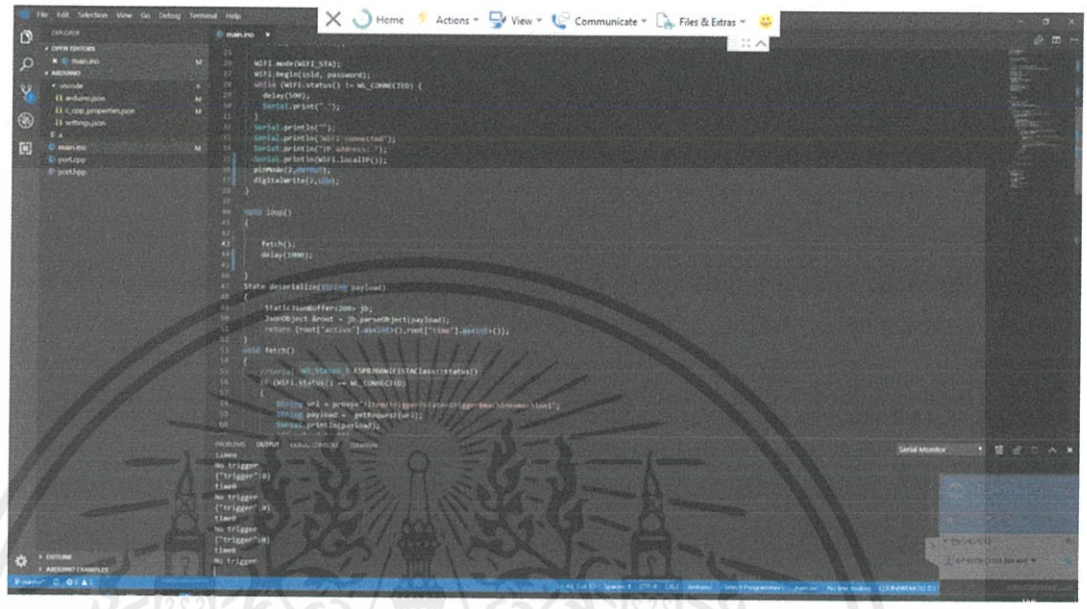
รูปที่ 4.3 แสดงผลส่วนคิวอาร์โค้ดในการสแกนเพื่อชำระเงิน

4.2 การทดลองชำระเงิน

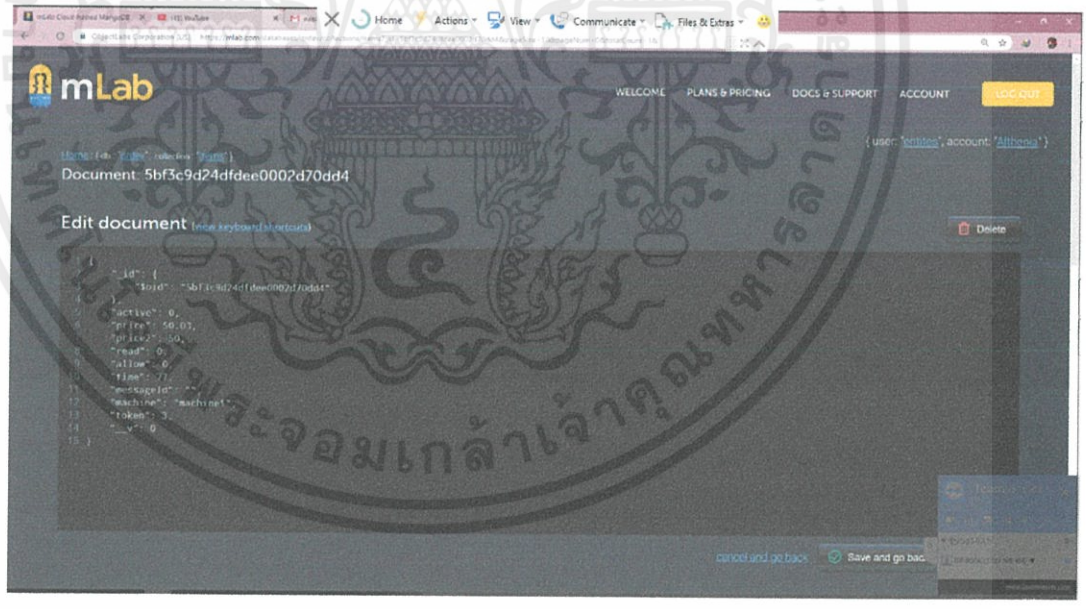
รูปที่ 4.4 แสดงการใช้แอปพลิเคชั่น KMA ในการสแกนเพื่อชำระเงิน

รูปที่ 4.5 แสดงการยืนยันการชำระเงิน

4.3 การประมวลฝั่งเซิพเวอร์

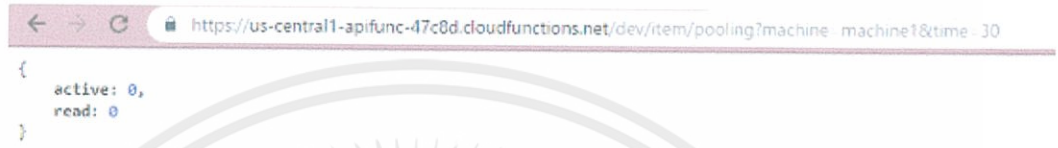


รูปที่ 4.6 แสดงการทำงานของเซิพเวอร์ในการตรวจสอบข้อมูลรายการ

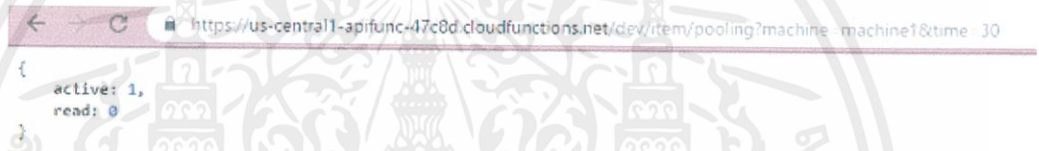


รูปที่ 4.7 แสดงการงานของเซิพเวอร์ในการเก็บข้อมูลของรายการเรียกชำระเงิน

รูปที่ 4.8 แสดงการสร้างรหัสคิวอาร์สำหรับชำระเงินและสถานะในการชำระเงิน



รูปที่ 4.9 แสดงสถานะในการชำระเงิน และสถานะการอ่านข้อมูลของไมโครคอนโทรลเลอร์



รูปที่ 4.10 แสดงสถานะในการชำระเงิน และสถานะการอ่านข้อมูลของไมโครคอนโทรลเลอร์โดยขณะนี้มีการชำระเงินเข้ามาแล้ว

โดยเมื่อมีการทำรายการเซิพเวอร์จะทำการสร้างรายการสำหรับเรียกชำระเงิน ตามรูปที่ 4.7 โดยActive คือสถานะการชำระเงิน Allow คือการตรวจสอบอีเมลของเซิพเวอร์ หากเป็นจำนวนเงินที่ถูกต้องจะทำให้สถานะ(flag) เปลี่ยนเป็น “1” แล้วค่าของ Active จะเป็น “1” Price คือจำนวนเงินที่เรียกชำระจากรหัสคิวอาร์time คือระยะเวลาที่เหลือในการทำรายการ โดยเซิพเวอร์จะทำรายการให้แอปพลิเคชั่นเรียกอ่าน โดยสถานะที่แอปพลิเคชั่นจะอ่านได้ตามรูปที่ 4.8 4.9 และ 4.10 โดยในรูปแรกยังไม่มีสถานะการชำระเงิน และมีรหัสคิวอาร์สำหรับเรียกชำระเงินให้แอปพลิเคชั่นแสดงผล ต่อมา มีสถานะของการชำระเงินและการอ่านข้อมูลของไมโครคอนโทรลเลอร์ โดยเมื่อมีการชำระเงินตามรูปที่ 4.10 ภายในเวลาที่กำหนดและไมโครคอนโทรลเลอร์ทำการอ่านข้อมูลแล้ว (“read 1”) คอนโทรลเลอร์จำดำเนินโปรแกรมเพื่อทำการจำหน่ายสินค้าออกไป หากแต่ถ้าไม่มีการชำระเงินภายในเวลาที่กำหนด (“active 0” “read 1”) จะมีการส่งคำสั่งเพื่อลบรายการออกจากระบบต่อไป

4.4 ผลการทดลองปริมาณน้ำที่จำหน่าย

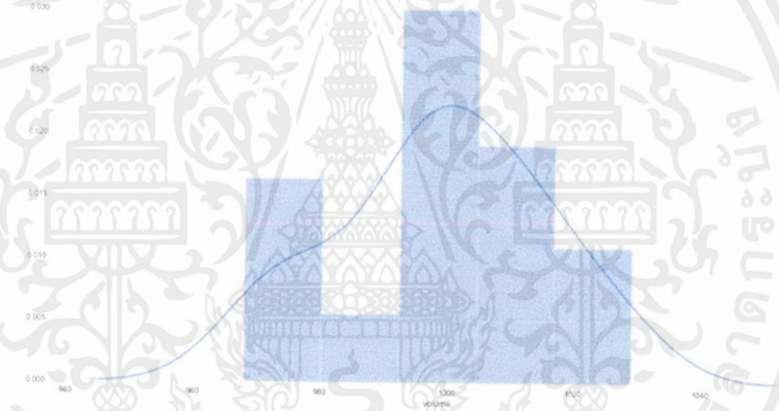
การทดลองปริมาตรน้ำที่จำหน่ายคำนวนจากน้ำหนักน้ำรวมถึงน้ำหนักขวดลบด้วยน้ำหนักขวดเปล่า โดยกำหนดให้น้ำมีความหนาแน่นประมาณ 1 กรัม/ลบ.ซม โดยผลการทดลองเป็นไปตามตารางที่4.1

ครั้งที่	น้ำหนัก	น้ำหนักขวดเปล่า	ปริมาตร	ค่าความผิดพลาดจากร้อยละที่กำหนด
1	1041	40	1001	0.1
2	1064	40	1024	2.4
3	1041	40	1001	0.1
4	1055	40	1015	1.5
5	1032	40	992	0.8
6	1043	40	1003	0.3
7	1048	40	1008	0.8
8	1051	40	1011	1.1
9	1035	40	995	0.5
10	1026	40	986	1.4
11	1046	40	1006	0.6
12	1041	40	1001	0.1
13	1049	40	1009	0.9
14	1031	40	991	0.9
15	1034	40	994	0.6
16	1046	40	1006	0.6
17	1053	40	1013	1.3
18	1037	40	997	0.3
19	1042	40	1002	0.2
20	1041	40	1001	0.1
21	1037	40	997	0.3
22	1050	40	1010	1
23	1033	40	993	0.7

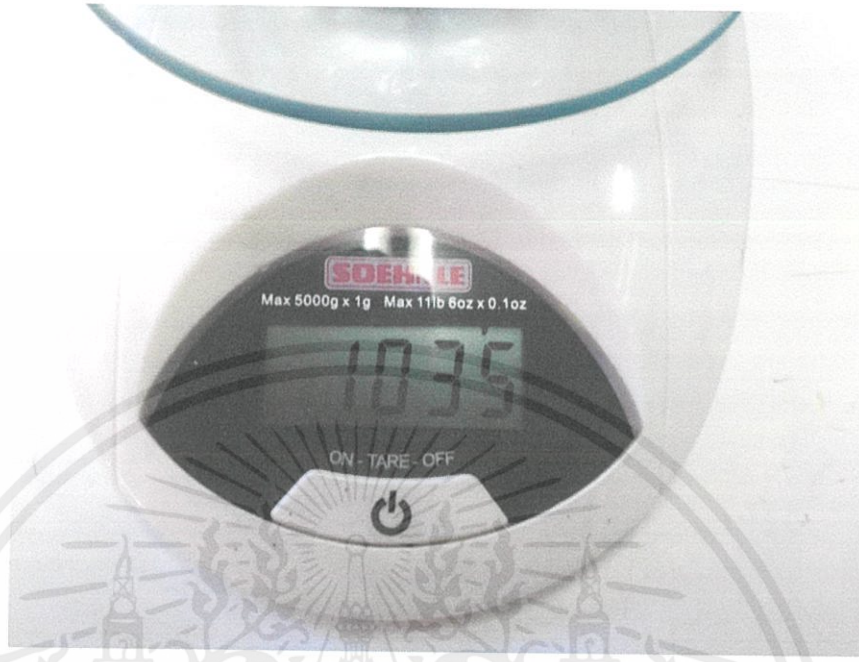
ตารางที่ 4.1 แสดงผลการทดลองวัดปริมาตรน้ำจากการชั่งน้ำหนัก

24	1036	40	996	0.4
25	1042	40	1002	0.2
26	1053	40	1013	1.3
27	1041	40	1001	0.1
28	1037	40	997	0.3
29	1029	40	989	1.1
30	1042	40	1002	0.2

ตารางที่ 4.1(ต่อ) แสดงผลการทดลองวัดปริมาตรน้ำจากการชั่งน้ำหนัก



รูปที่ 4.11 กราฟการกระจายของปริมาตรน้ำ



รูปที่ 4.12 แสดงถึงการชั่งน้ำหนักปริมาตรน้ำ

รูปที่ 4.12 แสดงถึงการชั่งน้ำหนักขวดเปล่า

บทที่ 5

สรุปผลการวิจัยและข้อเสนอแนะ

5.1 สรุปผลการทดลอง

จากการทดลอง ผลนั้นเป็นไปตามที่ตั้งไว้ตามวัตถุประสงค์คือแอปพลิเคชั่นสามารถสร้างรหัสคิวอาร์โค้ดเพื่อให้แอปพลิเคชั่นโมบายแบ้งกิ้งนั้นสามารถสแกนรหัสเพื่อทำการชำระเงินในระบบพร้อมเพย์ได้โดยจากผลการทดลองตามตารางที่ 4.1 ความผิดพลาดในการจ่ายน้ำไม่เกินร้อยละ 10 จากที่ตั้งเป้าหมายไว้

5.2 วิจารณ์ผลการทดลอง

การทดลองนี้มีข้อสังเกตและปัญหามากมายหลายอย่างจากที่ตั้งไว้ตอนแรก ได้แก่ ปัญหาเมื่อมีจำนวนตู้จำหน่ายสินค้าเพิ่มขึ้นจะทำให้ระบบทราบได้อย่างไรว่ามีการสั่งซื้อสินค้าจากตู้ไหน โดยเราได้ทดลองแก้ปัญหานี้ด้วยการเติมเศษสตางค์แบบสุ่มเพื่อให้ทราบว่าเป็นตู้จำหน่ายสินค้าตู้ไหนปัญหาข้อต่อมาเนื่องด้วยเป็นAPI ที่เขียนเองจึงอาจจะยังมีปัญหาในความล่าช้า

5.3 ข้อเสนอแนะ

จากการทดลองนี้เป็นไปได้ที่เราจะสามารถนำระบบชำระเงินพร้อมเพย์ไปประยุกต์กับตู้ขายสินค้าและบริการอัตโนมัติต่างๆ ได้อีกเช่น เครื่องซักผ้าหยอดเหรียญ ตู้ขายสินค้าอัตโนมัติ เป็นต้น นอกจากนี้โครงงานนนี้เป็นการศึกษาทดลองขนาดเล็กในการแก้ไขปัญหาในกรณีที่มีตู้กดน้ำมากกว่าหนึ่งตู้นั้นยังใช้วิธีการในการปัดเศษสตางค์ โดยในอนาคตหากธนาคารมีการพัฒนา API ให้ผู้พัฒนาสามารถตรวจสอบข้อมูลการชำระเงินจากธนาคารได้ จะทำปัญหาในการใช้เศษสตางค์เพื่อตรวจสอบการชำระเงินนั้นไม่จำเป็นต่อไป

เอกสารอ้างอิง

[1] ธนาคารแห่งประเทศไทย. (2560). **พร้อมเพย์คืออะไร.** ค้นข้อมูลจาก https://www.bot.or.th/Thai/PaymentSystems/PSServices/PromptPay/Pages/default.aspx.

[2] ณัฐวุฒิ บุญโรจน์วงค์, &กชกร พระพรตระการ. (2560). **ความหลากหลายของคิวอาร์โค้ด A Variety of QR Code.** (ม.ป.ท.).

[3] กระทรวงศึกษาธิการ. สำนักงานปลัดกระทรวงศึกษาธิการ. (2561). **QR Code คืออะไร?**. ค้นข้อมูลจาก http://www.ops.moe.go.th/ops2017/สาระน่ารู้/1877-qr-code-คืออะไร.

[4] ธนาคารแห่งประเทศไทย. (2560). **API : Application Programming Interface.** ค้นข้อมูลจาก https://www.bot.or.th/Thai/Statistics/EconomicAndFinancial/Pages/API.aspx.

[5] iEnergyGuru. (2558). **ปั๊ม(PUMP).** ค้นข้อมูลจาก https://ienergyguru.com/2015/09/pump/.

[6] พิมพ์เพ็ญ พรเฉลิมพงศ์ และ นวภัทรา หนูนาค. (ม.ป.ป). **Ultrasonic sensor / เซนเซอร์ชนิดใช้เสียง หรือเซนเซอร์ชนิดอัลตราโซนิก.** ค้นข้อมูลจาก http://www.foodnetworksolution.com/wiki/word/4348/.

[7] สงกรานต์ สว่างวัล. (2556) . **โครงงานเครื่องรถน้ำต้นไม้อัตโนมัติ.** ค้นข้อมูลจาก http://aimagin.com/blog/โครงงานเครื่องรดน้ำต้นไม้/?lang=th.

ภาคผนวก

ภาคผนวก ก.

โค้ดโปรแกรมที่ใช้ในโครงงานทั้งหมด

1. Backend

1.1 index

```
require("dotenv").config()
const express = require("express"),
      bodyParser = require("body-parser"),
      errorHandler = require("./handler/error.handler"),
      productRouter = require("./routes/product.route"),
      authRouter = require("./routes/auth.route"),
      functions = require("firebase-functions"),
      firebase = require("firebase-admin");

const firebaseApp = firebase.initializeApp(functions.config().firebase);

var app = express();
var PORT = process.env.MYPORT || 8080;
app.use(bodyParser.json());
app.use("/product",productRouter)
app.use("/token",authRouter)
app.use((req,res,next)=>{
        let err = new Error("Not Found");
        err.status = 404;
        return next(err);
});
app.use(errorHandler);
//app.listen(PORT,process.env.IP,()=>{
        //console.log("server started!")
//})
exports.api = functions.https.onRequest(app);
```

1.2 handler

1.2.1 auth.hander

```
const jwt = require("jsonwebtoken")
function getToken(req,res,next)
{
        let token = jwt.sign({},process.env.SECRET_KEY);
        return res.status(200).json({
                token
        })
}
function validate(req,res,next)
{
        let {token} = req.query;
        jwt.verify(token,process.env.SECRET_KEY,function(err,token){
                if(err) return res.status(200).json({
                        message:"token not valid",
                        status:403
                });
                console.log(req.query);
                return next();
        })
}
module.exports = {getToken,validate}
```

1.2.2 error.handler

```
function errorHandler(err,req,res,next)
{
        return res.status(err.status||500)
                .json(err.message || "sth went wrong from server side!");
}
module.exports = errorHandler;
```

1.2.3 mqtt.handler

```
const mqtt = require("mqtt")
const db = require("../models");
class MqttHandler{
  constructor(host,username,password) {
    this.mqttClient = null;
    this.host = host
    this.username = username
    this.password = password
  }
  connect() {
    this.mqttClient = mqtt.connect(this.host, {  username:this.username,password:
this.password });
    this.mqttClient.subscribe("machine,delete",{qos:0});
    this.mqttClient.on('error', (err) => {
      console.log(err);
      this.mqttClient.end();
    });
```

```
    this.mqttClient.on('connect', () => {
      console.log(`mqtt client connected`);
    });
    this.mqttClient.on('close', () => {
      console.log(`mqtt client disconnected`);
    });
  this.mqttClient.on("message",(topic,message)=>{

          db.product.findOne({machine:message.toString()})
                .then(data=>{
                      data.remove()
                })
                .catch(err=>{
                      console.log("no document to delete");
                })
  })
  }
}
module.exports = MqttHandler;
```

1.2.4 product.handler

```
const db = require("../models"),
      qr = require("qrcode"),
      generatePayload = require("promptpay-qr");
function createProduct(req,res,next)
{
        let {price,machine,flag} = req.query;
        db.product.find({})
              .then(data=>{
```

```
                let tokens = data.map(d=>d.token);
                let token = generateToken(tokens);
                db.product.create({price,machine,token,flag})
                        .then(data=>{
                                let {active,read,mPrice} = data;
                                let number = "095-934-4539";
                                let payload =
generatePayload(number,{amount:mPrice});
                                qr.toDataURL(payload,function(err,url){
                                        res.status(200).json({
                                                active,
                                                read,
                                                url
                                        })
                                })
                        })
        })
                .catch(err=>{
                        next(err)
                })

}
function payment(req,res,next)
{
        let {price} = req.query;
        db.product.findOne({mPrice:price})
                .then(data=>{
                        let {machine,price} = data;
                        db.mqtt.mqttClient.publish(machine,price.toString())
```

```
                    res.status(200).json({
                        log:"valid payment"
                    }
                    )
                })
                .catch(err=>{
                    next(err);
                })
}
function removeProduct(req,res,next)
{
    let {machine} = req.query;
    db.product.findOne({machine})
        .then(data=>{
            data.remove()
            return res.status(200).json({
                log:"removed"
            })
        })
        .catch(err=>{
            return res.status(200).json({
                log:"item not found"
            })
        })
}
function generateToken(list)
{
    let tokens  = [];
    for(let token=0; token<10;token++)
    {
```

```
            if(token in list) continue;
            tokens.push(token);
        }
        let length = tokens.length;
        return tokens[Math.floor(Math.random()*length)];
}
module.exports = {createProduct,payment,removeProduct}
```

1.3 models

1.3.1 index

```
const mongoose = require("mongoose");
    productModel = require("./product.model"),
    mqttHandler = require("../handler/mqtt.handler");

mongoose.Promise = Promise;
var dbHost = "mongodb://entites:123zxc@ds147985.mlab.com:47985/mqtt_qrcode"
var mqtt = new
mqttHandler("tcp://m16.cloudmqtt.com:17516","djmvfzne","5rE2BX7ctJKi")
mqtt.connect()
mongoose.connect(dbHost);
module.exports.product = productModel;
module.exports.mqtt = mqtt;
```

1.3.2 product.model

```
const mongoose = require("mongoose");
const productSchema = new mongoose.Schema({
        price:{
                type:Number,
                default:0
        },
        mPrice:{
                type:Number,
                default:0
        },
        token:{
                type:Number,
                default:0
        },
        machine:{
                type:String,
        },
        flag:{
                type:Number,
                require:true
        }
})
productSchema.pre("save",function(){
        let token = Number(`0.0${this.token}`);
        if(this.flag == 1)
        {
                this.mPrice = this.price - token
        }
        else
```

```
        {
                this.mPrice = this.price + token
        }
        console.log(this.mPrice)
})
const productModel = mongoose.model("product",productSchema);
module.exports = productModel;
```

1.4 routes

1.4.1 auth.route

```
const express = require("express"),
    {getToken} = require("../handler/auth.hander");
var router = express.Router();
router.get("/gettoken",getToken);
module.exports = router
```

1.4.2 product.route

```
const express = require("express"),
    {createProduct,removeProduct,payment} = require("../handler/product.handler.js"),
    {validate} = require("../handler/auth.hander");
var router = express.Router()
router.get("/create",createProduct)
router.get("/remove",removeProduct)
router.get("/paied",validate,payment)
module.exports = router
```

2. Backend app

2.1 app

```
require("dotenv").config();
const express = require("express"),
      bodyParser = require("body-parser"),
      errorHandler = require("./handler/error.handler"),
      qrRoutes = require("./routes/qrcode"),
      cors =require("cors"),
      auth = require("./routes/auth"),
      classifier = require("./routes/classifier");
const app = express();
app.use(cors());
app.use(bodyParser.json());
app.use("/qrcode",qrRoutes);
app.use("/auth",auth);
app.use("/classifier",classifier);
app.use((req,res,next)=>{
    const error = new Error("Not found Page");
    error.status = 404;
    next(error);
})
app.use(errorHandler);
app.listen(process.env.MYPORT,process.env.IP,()=>{
    console.log("server started!");
})
```

2.2 handler

2.2.1 auth.handler

```
const db = require("../model"),
      jwt = require("jsonwebtoken");
async function signupHandler(req,res,next)
{
    try {
        let user = await db.user.create(req.body)
        let {name,id} = user;
        let token = jwt.sign({
            name,
            id,
        },process.env.SECRET_KEY);
        return res.status(200).json({
                name,
                id,
                token
            })
    } catch (err) {
        if(err.code === 11000)
            err.message = "Sorry , that username was taked"
        return next({
            status:401,
            message:err.message || "username/password invalid"
        })
    }
}
async function signinHandler(req,res,next){
```

```
    try {
        let user = await db.user.findOne({name:req.body.name});
        let {name,id} = user;
        let isMatch = await user.comparePassword(req.body.password);
        console.log("is passwordMath: ",isMatch);
        if(isMatch)
        {
            let token = jwt.sign({
                name,
                id
            },process.env.SECRET_KEY);
            return res.status(200).json({name,id,token})
        }
        else
        {
            return next({
                message:"username/password invalid",
                status:401
            })
        }
    } catch (err) {
        return next({
            message:"username/password invalid",
            status:401
        })
    }


}
module.exports = {signinHandler,signupHandler};
```

2.2.2 classifier.handler

```
const vision = require('@google-cloud/vision');
function classifierTest(req,res,next)
{
    let result = [];
    let {imagepath} = req.body;
    const client = new vision.ImageAnnotatorClient();
    client
    .labelDetection(imagepath)
    .then(results => {
        const labels = results[0].labelAnnotations;
        labels.forEach(label => {
            result.push({description:label["description"],score:label["score"]})
        });
        res.status(200).json({imagepath,result:[...result]});
    })
    .catch(err => {
        next({message:"sth went wrong",status:500});
    })
}
module.exports = classifierTest;
```

2.2.3 error.handler

```
function errorHandler(err,req,res,next)
{
    err = {
        ...err,
        message: err.message || "something wentwrong",
    }
    res.status(err.status||500).json(err);
}
module.exports = errorHandler;
```

2.2.4 qrcode.handler

```
const db  = require("../model");
function createQrCodeHandler(req,res,next){
    let {qrcode ,price,imagepath,name}= req.body;
    db.qr.create({name,qrcode,price,imagepath})
    .then(data=>{
        return res.status(200).json(data);
    })
    .catch(err=>{
        err.message = "cant create item";
        err.status = 400;
        return next(err);
    })
}
function fetchQrCodeHandler(req,res,next){
    db.qr.find({})
        .then(qrcode=>{
            if(qrcode && qrcode.length>0)
```

```
            {
                return res.status(200).json(qrcode);
            }
            else
            {
                throw Error("not found")
            }
        })
        .catch(err=>{
            return next({...err,status:400});
        })
}
function deleteQrCodeHandler(req,res,next){
    console.log("deleting",req.params.id);
    db.qr.findByIdAndRemove(req.params.id)
        .then(()=>{
            return  res.status(200).json({id:req.params.id});
        })
        .catch(()=>{
            return  next({message:"cant find item",status:400});
        })
}
function editQrCodeHandler(req,res,next)
{
    db.qr.findByIdAndUpdate(req.params.id,req.body ,{new:true})
        .then(item=>{
            console.log("============Good==============");
            console.log(item);
            return res.status(200).json(item);
        })
```

```
        .catch(()=>{
            console.log("============Error==============");
            return next({message:"cant edit item",status:400});
        })
}
module.exports =
{createQrCodeHandler,fetchQrCodeHandler,deleteQrCodeHandler,editQrCodeHandler};
```

2.3 middleware

2.3.1 verify.req

```
require("dotenv").load();
const jwt = require("jsonwebtoken");
function loginRequired(req,res,next)
{
    const token = req.headers.authorization.split(" ")[1];
    jwt.verify(token,process.env.SECRET_KEY,(err,decoded)=>{
        if(decoded)
            next();
        else
        {
            next({status:401,message:"Please Log in first"})
        }
    })
}
module.exports = loginRequired;
```

2.4 model

2.4.1 index

```
const mongoose = require("mongoose"),
      qrCode = require("./qrcode.model"),
      user = require("./user.model");
mongoose.connect("mongodb://althenia:Q901lune@ds149960.mlab.com:49960/alproje
ct")
mongoose.Promise = Promise;
module.exports.qr = qrCode;
module.exports.user = user
```

2.4.2 qrcode.model

```
const mongoose = require("mongoose");
const qrSchema = new mongoose.Schema({
  name:{
    type:String,
    required:true
  },
  imagepath:{
    type:String,
    required:true,
  },
    qrcode:{
      type:String,
      required:true,
    },
```

```
    price:{
        type:String,
        required:true,
      },
})
const qrModel = mongoose.model("qr",qrSchema);
module.exports = qrModel;
```

2.4.3 user.model

```
const mongoose = require("mongoose"),
      bcrypt = require("bcrypt");
const userSchema = new mongoose.Schema({
    name:{
        type:String,
        required:true,
        unique:true
    },
    password:{
        type:String,
        required:true
    }

})
userSchema.pre("save", async function(next){
    try {
        if(!this.isModified("password"))
        {
            return next();
        }
```

```
        let hashPassword = await bcrypt.hash(this.password,10);
        this.password = hashPassword;
        return next();
    } catch (err) {
        return next(err);
    }
})

userSchema.methods.comparePassword = async function(candicatePassword,next)
{
    try {
        console.log(this.password,typeof(this.password));
        console.log("form comparepassword method:
",candicatePassword,typeof(candicatePassword));
        let isMatch = await bcrypt.compare(candicatePassword,this.password);
        return isMatch
    } catch (err) {
        return next(err);
    }
}

const User = mongoose.model("user",userSchema);
module.exports = User;
```

2.5 routes

2.5.1 auth

```
const express = require('express'),
    authHandler = require("../handler/auth.handler");
const router = express.Router();
router.post("/signup",authHandler.signupHandler);
router.post("/signin",authHandler.signinHandler);
module.exports = router;
```

2.5.2 classifier

```
const express = require("express"),
    classifier = require("../handler/classifier.handler");
const router = express.Router();
router.post("/",classifier);
module.exports = router;
```

2.5.3 qrcode

```
const express = require("express"),
    db  = require("../model"),
    logginRequired = require("../middleware/verify.req"),
    qrcodeHandler = require("../handler/qrcode.handler");
const router = express.Router();
router.post("/create",logginRequired,qrcodeHandler.createQrCodeHandler);
router.get("/fetch",qrcodeHandler.fetchQrCodeHandler);
router.delete("/:id",qrcodeHandler.deleteQrCodeHandler);
router.put("/:id/edit",qrcodeHandler.editQrCodeHandler);
module.exports = router;
```

3. Client side

3.1 bottle.kt

```
package com.example.client_side
import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.SeekBar
import android.widget.TextView
import android.widget.Toast
import kotlinx.android.synthetic.main.activity_qrcode_payment.*
import org.apache.log4j.chainsaw.Main
import org.eclipse.paho.android.service.MqttAndroidClient
import org.eclipse.paho.client.mqttv3.*
class bottle : AppCompatActivity() {
    var seekBar:SeekBar? = null
    var client:MqttAndroidClient? = null
    var value:TextView? = null
    var state:Int = 0
    fun connect():Unit
    {
        var clientId:String = MqttClient.generateClientId()
        client = MqttAndroidClient(this,"tcp://0.0.0.0:1883",clientId)
        Log.i("myqtt","connection starting")
        Log.i("mymqtt",client.toString())
        var options:MqttConnectOptions = MqttConnectOptions()
        options.mqttVersion = MqttConnectOptions.MQTT_VERSION_3_1
        options.isCleanSession = false
```

```
options.isAutomaticReconnect = true
var token:IMqttToken? = client!!.connect(options)
token!!.actionCallback = object :IMqttActionListener{
    override fun onSuccess(asyncActionToken: IMqttToken?) {
        Log.i("mymqtt","Connected Success")
        subscribe("bottle")
        client!!.setCallback(object :MqttCallback{
            override fun messageArrived(topic: String?, message: MqttMessage?) {
                if(topic!!.equals("bottle"))
                {
                    var payload = message!!.toString().toFloat()
                    Log.i("value",payload.toString())
                    seekBar!!.progress = convertLogic(payload)
                    reScale(convertLogic(payload))
                    if(payload <= 0)
                    {
                        val intent = Intent(this@bottle,MainActivity::class.java)
                        startActivity(intent)
                        finish()
                        client!!.close()
                        client!!.disconnect()

                    }
                }
            }

            override fun connectionLost(cause: Throwable?) {
            }

            override fun deliveryComplete(token: IMqttDeliveryToken?) {
```

```
                }

            })

        }

        override fun onFailure(asyncActionToken: IMqttToken?, exception:
Throwable?) {
        }

    }
  }
  fun paied(v:View?)
  {
    var message = MqttMessage()
    when(state)
    {
      0 -> {
        message.payload = "1".toByteArray()
        state = 1
      }
      1 -> {
        message.payload = "0".toByteArray()
        state = 0
      }
    }
    client!!.publish("currentValue",message)
  }
  fun disconnect():Unit
  {
```

```
        var disconToken:IMqttToken = client!!.disconnect()
        disconToken.actionCallback = object:IMqttActionListener{
            override fun onSuccess(asyncActionToken: IMqttToken?) {
                Log.i("mymqtt","disconnect was successfully")
            }
            override fun onFailure(asyncActionToken: IMqttToken?, exception:
Throwable?) {
                Log.i("mymqtt","sth went wrong \t"+exception.toString())
            }
        }
    }
    fun subscribe(topic:String):Unit
    {
        var qos:Int = 1
        var subToken:IMqttToken = client!!.subscribe(topic,qos)
        subToken.actionCallback = object :IMqttActionListener{
            override fun onSuccess(asyncActionToken: IMqttToken?) {
            }
            override fun onFailure(asyncActionToken: IMqttToken?, exception:
Throwable?) {
            }

        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_bottle)
        value = findViewById(R.id.textView) as TextView
        seekBar = findViewById(R.id.seekBar) as SeekBar
```

```
        seekBar!!.max = convertLogic(qrcode_payment.absPrice.toFloat())
        seekBar!!.progress = convertLogic(qrcode_payment.absPrice.toFloat())
        reScale(convertLogic(qrcode_payment.absPrice.toFloat()))
        connect()
    }
    fun convertLogic(value:Float):Int
    {
        var float = value*100
        return float.toInt()
    }
    fun reScale(scaledValue:Int):Unit
    {
        var rescaledValue:Float = (scaledValue.toFloat()/100)
       value!!.setText(String.format("%.2f",rescaledValue))
    }
}
```

3.2 cartlist.kt

```
package com.example.client_side
import android.app.Activity
import android.app.NativeActivity
import android.content.Context
import android.content.DialogInterface
import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.support.v7.app.AlertDialog
import android.os.Bundle
import android.text.Editable
import android.text.InputType
```

```
import android.text.Layout
import android.text.TextWatcher
import android.util.Log
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.*
class cartlist : AppCompatActivity() {
    companion object {
        var totalPriceView:TextView? = null
    }

    fun goBackListener(v:View?) : Unit
    {
        var intent = Intent(this,MainActivity::class.java)
        intent.setFlags(Intent.FLAG_ACTIVITY_REORDER_TO_FRONT)
        startActivity(intent)
    }
    fun goShop(v:View?) : Unit {
        if((MainActivity.order.size > 0) && (MainActivity.totalPrice > MainActivity.check))
        {
            var intent = Intent(this,Paied::class.java)
            startActivity(intent)
            finish()


        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_cartlist)
```

```
        var listview = findViewById<ListView>(R.id.cartlistview)
        var customAdapter = CustomAdaper()
        listview.adapter = customAdapter
        var count = MainActivity.order.size
        var myshopAlert = findViewById<ImageView>(R.id.myshopAlert)
        if (count > 0)
        {
            myshopAlert.visibility = View.VISIBLE
            MainActivity.updateImageView(this,myshopAlert,count,"alert")
        }
        else if(count ==0)
        {
            myshopAlert.visibility = View.INVISIBLE
        }
    }

    class CustomAdaper : BaseAdapter() {
        override fun getView(position: Int, convertView: View?, parent: ViewGroup?): View
{
            var context = parent!!.context
            var inflater = LayoutInflater.from(context)
            if(position == MainActivity.order.size)
            {
                var totalPriceLayout = inflater.inflate(R.layout.total_price,null)
                var price = totalPriceLayout.findViewById<TextView>(R.id.price)
                price.text = MainActivity.totalPrice.toString()
                totalPriceView =  price
                return totalPriceLayout

            }
```

```
            var customLayout = inflater.inflate(R.layout.custom_layout,null)
            var i = MainActivity.order.get(position)
            var totalprice = MainActivity.shoppingList[i][1]
            var amount = MainActivity.shoppingList[i][0]
            var price = MainActivity.bottleProfiles[i][1].toInt()

            var customLayoutEditText =
customLayout.findViewById<EditText>(R.id.editText)
            var customLayoutImageView =
customLayout.findViewById<ImageView>(R.id.imageView3)
            var customLayoutTotalBaht =
customLayout.findViewById<TextView>(R.id.baht)
            var customLayoutPrice =
customLayout.findViewById<TextView>(R.id.textView2)
            customLayoutEditText.tag = i.toString()
            customLayoutPrice.text = price.toString()
            customLayoutEditText.setText(amount.toString())
            customLayoutTotalBaht.setText(totalprice.toString())
            customLayoutEditText.isTextSelectable.not()
            customLayoutEditText.setRawInputType(InputType.TYPE_NULL)

           customLayoutEditText.setOnClickListener { v:View?->
              var innerLayoutcustomLayoyt = inflater.inflate(R.layout.custom_layout,null)
              var innerLayoutcustomLayoutBottleView =
innerLayoutcustomLayoyt.findViewById<ImageView>(R.id.imageView3)
              var innerLayoutcustomLayoutEditText  =
innerLayoutcustomLayoyt.findViewById<EditText>(R.id.editText)
              var innerLayoutcustomLayoutTextView  =
innerLayoutcustomLayoyt.findViewById<TextView>(R.id.baht)
```

```
            var innerLayoutcustomLayoutPrice      =
innerLayoutcustomLayoyt.findViewById<TextView>(R.id.textView2)
            var bottlePrice = MainActivity.bottleProfiles[i][1].toInt()
            innerLayoutcustomLayoutPrice.text = bottlePrice.toString()
            innerLayoutcustomLayoutTextView.text = totalprice.toString()
            innerLayoutcustomLayoutEditText.setText(amount.toString())

MainActivity.updateImageView(context,innerLayoutcustomLayoutBottleView,i,"bottle")

innerLayoutcustomLayoutEditText.addTextChangedListener(object:TextWatcher{
              override fun afterTextChanged(s: Editable?) {
              }

              override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int,
after: Int) {
              }

              override fun onTextChanged(s: CharSequence?, start: Int, before: Int,
count: Int) {
                if(innerLayoutcustomLayoutEditText.text.toString().isEmpty())
                {
                  innerLayoutcustomLayoutTextView.text = (0).toString()
                  return
                }

                var amount = innerLayoutcustomLayoutEditText.text.toString().toInt()
                innerLayoutcustomLayoutTextView.text = (amount *
bottlePrice).toString()

            }
```

```
            })
            AlertDialog.Builder(context)
                .setTitle("Change amount")
                .setView(innerLayoutcustomLayoyt)
                .setPositiveButton("Edit amount",object:DialogInterface.OnClickListener{
                    override fun onClick(dialog: DialogInterface?, which: Int) {
                        var outerAmount = customLayoutEditText.text.toString()
                        var innerAmount =
innerLayoutcustomLayoutEditText.text.toString()
                        if(innerAmount.equals(outerAmount))
                        {
                          return
                        }
                        else
                        {
                         var tag = customLayoutEditText.tag.toString().toInt()
                         var newTotalPrice = 0
                         MainActivity.shoppingList[tag][0] = innerAmount.toInt()
                         MainActivity.shoppingList[tag][1] =  (innerAmount.toInt() *
bottlePrice)
                         for(item in MainActivity.shoppingList)
                         {
                           newTotalPrice += item[1]
                            customLayoutEditText.setText(innerAmount)

customLayoutTotalBaht.setText((innerAmount.toInt()*bottlePrice).toString())
                        }
                        MainActivity.totalPrice = newTotalPrice
                        totalPriceView!!.text = newTotalPrice.toString()
```

```
                }
            }

        })
        .setNegativeButton("Delete item",object:DialogInterface.OnClickListener{
            override fun onClick(dialog: DialogInterface?, which: Int) {
                var tag:Int = customLayoutEditText.tag.toString().toInt()
                MainActivity.shoppingList[tag][0] = 0
                MainActivity.shoppingList[tag][1] = 0
                var newTotalPrice = 0
                for(item in MainActivity.shoppingList)
                {
                  newTotalPrice += item[1]
                }
                totalPriceView!!.text = newTotalPrice.toString()
                MainActivity.totalPrice = newTotalPrice
                customLayout.visibility = View.GONE
                MainActivity.order.remove(tag)
                var intent = Intent(context,MainActivity::class.java)
                context.startActivity(intent)

            }

        })
        .show()

}


MainActivity.updateImageView(context,customLayoutImageView,i,"bottle")
```

```
            return customLayout



        }

        override fun getItem(position: Int): Any {
            TODO("not implemented") //To change body of created functions use File |
Settings | File Templates.
        }

        override fun getItemId(position: Int): Long {
            TODO("not implemented") //To change body of created functions use File |
Settings | File Templates.
        }

        override fun getCount(): Int {
            return MainActivity.Companion.order.size + 1
        }
    }
}
```

3.3 MainActivity.kt

```
package com.example.client_side
import android.app.Service
import android.content.Context
import android.content.DialogInterface
import android.content.Intent
import android.content.pm.PackageManager
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.os.CountDownTimer
import android.os.Environment
import android.support.v4.app.ActivityCompat
import android.support.v4.content.ContextCompat
import android.support.v7.app.AlertDialog
import android.text.Editable
import android.text.TextWatcher
import android.util.Log
import android.view.View
import android.view.inputmethod.InputMethodManager
import android.widget.*
import io.moquette.BrokerConstants
import io.moquette.server.config.MemoryConfig
import java.io.File
import java.util.*
import kotlin.concurrent.thread
class MainActivity : AppCompatActivity(),View.OnClickListener {

    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out
String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
```

```
        if(grantResults.size > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED &&
            grantResults[1] == PackageManager.PERMISSION_GRANTED)
        {
            startMqttServer()
        }
    }
    fun startMqttServer() : Unit
    {

        var server = io.moquette.server.Server()
        try {
            var serverProperties = Properties()

serverProperties.setProperty(BrokerConstants.PORT_PROPERTY_NAME,BrokerConstants.
PORT.toString())

serverProperties.setProperty(BrokerConstants.HOST_PROPERTY_NAME,BrokerConstants.
HOST)

serverProperties.setProperty(BrokerConstants.PERSISTENT_STORE_PROPERTY_NAME,
                Environment.getExternalStorageDirectory().absolutePath + File.separator +
BrokerConstants.DEFAULT_MOQUETTE_STORE_MAP_DB_FILENAME
            )
            var memoryConfig = MemoryConfig(serverProperties)
            initFiles()
            server.startServer(memoryConfig)
            Log.i("mqttserver","server started!")
        } catch (e: Exception) {
            e.printStackTrace()
```

```
        }
    }

    fun initFiles() : Unit
    {

        var path:String = Environment.getExternalStorageDirectory().absolutePath
        var directory = File(path)
        var files =  directory.listFiles()
        for(file in files)
        {
            if(file.name.contains("moquette_store"))
            {
                var willBeDeletedFile =
File(Environment.getExternalStorageDirectory().absolutePath + File.separator +
file.name)
                if(willBeDeletedFile.delete())
                {
                    Log.i("mqttserver",String.format("%s was deleted",file.name))
                }
            }
        }

    }



    override fun onClick(v: View) {
        if(v.id in arrayOf(R.id.bottle1,R.id.bottle2,R.id.bottle3))
        {
```

```
        var customLayoyt = layoutInflater.inflate(R.layout.custom_layout,null)
        var customLayoutBottleView =
customLayoyt.findViewById<ImageView>(R.id.imageView3)
        var customLayoutEditText =
customLayoyt.findViewById<EditText>(R.id.editText)
        var customLayoutTextView = customLayoyt.findViewById<TextView>(R.id.baht)
        var customLayoutPrice =
customLayoyt.findViewById<TextView>(R.id.textView2)

        var bottle = v as ImageButton
        var bottleTag = bottle.tag.toString().toInt()
        var bottlePrice = bottleProfiles[bottleTag][1].toInt()
        customLayoutPrice.text = (bottlePrice).toString()
        customLayoutEditText.addTextChangedListener(object:TextWatcher{
            override fun beforeTextChanged(s: CharSequence?, start: Int, count: Int,
after: Int) {
            }

            override fun afterTextChanged(s: Editable?) {
            }

            override fun onTextChanged(s: CharSequence?, start: Int, before: Int, count:
Int) {

                if(customLayoutEditText.text.toString().isEmpty())
                {
                    customLayoutTextView.text = (0).toString()
                    return
                }
```

```
            var amount = customLayoutEditText.text.toString().toInt()
            customLayoutTextView.text = (amount * bottlePrice).toString()
        }


    })
    MainActivity.updateImageView(this,customLayoutBottleView,bottleTag,"bottle")
    var context:Context = this
    AlertDialog.Builder(this)
        .setIcon(android.R.drawable.sym_def_app_icon)
        .setTitle("Type Your amount")
        .setView(customLayoyt)
        .setPositiveButton("Add to Cart",object:DialogInterface.OnClickListener{
            override fun onClick(dialog: DialogInterface?, which: Int) {
                var amount = customLayoutEditText.text.toString().toInt()
                if(amount>0)
                {
                    var price = shoppingList[bottleTag][1]
                    var bottleAmount = shoppingList[bottleTag][0]
                    bottleAmount += amount
                    var currentPrice = amount*bottlePrice
                    price += currentPrice
                    MainActivity.totalPrice += currentPrice
                    customLayoutTextView.setText(currentPrice.toString())
                    shoppingList[bottleTag][1] = price
                    shoppingList[bottleTag][0] = bottleAmount
                    if(!(bottleTag in order))
                    {
                        order.add(bottleTag)
                    }
```

```
                }

                var myshopAlert = findViewById<ImageView>(R.id.myshopAlert)
                var totalsize = order.size
                if(totalsize>0)
                {
                    myshopAlert.visibility = View.VISIBLE
                    MainActivity.updateImageView(context,myshopAlert,totalsize,"alert")

                }
                else if(totalsize == 0)
                {
                    myshopAlert.visibility = View.INVISIBLE
                }

            }
        })
        .show()
}
 if(order.size > 0)
 {
    if(v.id == R.id.imageView)
    {
        Log.i("current", shoppingList.toString())
        var intent = Intent(this,cartlist::class.java)
        startActivity(intent)
        finish()
    }
    else if((v.id == R.id.goShop) && (totalPrice >  check))
    {
```

```
            Log.i("current", order.toString())
            Log.i("current", totalPrice.toString())
            var intent = Intent(this,Paied::class.java)
            startActivity(intent)
            finish()

        }

    }
}
companion object {
    var shoppingList = arrayOf(arrayOf(0,0),arrayOf(0,0),arrayOf(0,0))
    var firebaseroot:String = "http://us-central1-apifunc-47c8d.cloudfunctions.net/api"
    var order = ArrayList<Int>()
    var bottleProfiles = arrayOf(arrayOf("9 Oz","1"),arrayOf("10 Oz","5"),arrayOf("15
Oz","10"))
    var totalPrice:Int = 0
    var threadInit:Boolean = false
    var check:Int = 0
    fun
updateImageView(context:Context,imageView:ImageView,flag:Int,mode:String):Unit
    {
        if(mode.equals("bottle"))
        {

            when(flag)
            {
                0 ->
imageView.setImageDrawable(context.getDrawable(R.drawable.bottle1))
```

```
                1 ->
imageView.setImageDrawable(context.getDrawable(R.drawable.bottle2))
                2 ->
imageView.setImageDrawable(context.getDrawable(R.drawable.bottle3))

            }

        }
        else if(mode.equals("alert"))
        {
            when(flag)
            {
                1 ->
imageView.setImageDrawable(context.getDrawable(R.drawable.aleat1))
                2 ->
imageView.setImageDrawable(context.getDrawable(R.drawable.aleat2))
                3 ->
imageView.setImageDrawable(context.getDrawable(R.drawable.aleat3))
            }


        }
    }


}
 var count = 0
 override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_main)
        var shoppingList = findViewById<ImageView>(R.id.myshopAlert)
        var order_size = MainActivity.order.size
        if (order_size > 0) {
            shoppingList.visibility = View.VISIBLE
            MainActivity.updateImageView(this, shoppingList, order_size, "alert")
        } else {
            shoppingList.visibility = View.INVISIBLE
        }


            if(threadInit == false)
            {
                thread (start=true) {

Log.i("mqttserver",Environment.getExternalStorageDirectory().absolutePath)

Log.i("mqttserver",BrokerConstants.DEFAULT_MOQUETTE_STORE_MAP_DB_FILENAME)

if(ContextCompat.checkSelfPermission(this,android.Manifest.permission.WRITE_EXTERN
AL_STORAGE) == PackageManager.PERMISSION_GRANTED &&

ContextCompat.checkSelfPermission(this,android.Manifest.permission.READ_EXTERNAL_
STORAGE) == PackageManager.PERMISSION_GRANTED)
                {
                    startMqttServer()
                }
                else
                {
```

```
                    var permission =
arrayOf<String>(android.Manifest.permission.READ_EXTERNAL_STORAGE,android.Manife
st.permission.WRITE_EXTERNAL_STORAGE)
                    ActivityCompat.requestPermissions(this,permission,1)
                }
                threadInit = true
            }



        }


    }
}
```

3.4 Paied.kt

```
package com.example.client_side
import android.content.Intent
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.TextView
import android.widget.Toast
import com.github.kittinunf.fuel.httpGet
import org.json.JSONObject
class Paied : AppCompatActivity() {
    fun onClick(v: View?) {
        Log.i("paiedbug",v!!.id.toString())
        var randomNumber = Math.floor(Math.random()*10)
        var answer:Boolean? = null
        when(v!!.id)
        {
            R.id.yes -> {answer = true}
            R.id.no -> {answer = false}
        }
        var result:Boolean = !((randomNumber>5) xor answer!!)
        number!!.text = randomNumber.toInt().toString()
        number!!.visibility = View.VISIBLE
        if(result)
        {
            makeAPICall(1)
            Toast.makeText(this,"Congrat!",Toast.LENGTH_LONG).show()
        }
        else
```

```
        {
            makeAPICall(0)
            Toast.makeText(this,"sorry,you get penality!",Toast.LENGTH_LONG).show()

        }
    }
    var number:TextView? = null
    fun makeAPICall(flag:Int) :Unit
    {

String.format("%s/product/create?machine=machine1&price=%d&flag=%d",MainActivity
.firebaseroot,MainActivity.totalPrice,flag)
            .httpGet()
            .responseString {req,res,result->
              if(res.statusCode == 200)
              {

                var data:String = result.get()
                var jsonObject = JSONObject(data)
                var url:String = jsonObject.getString("url")
                Log.i("qrcode",url)
                var urlBytes = url.substring(url.indexOf(",")+1)
                var intent = Intent(this,qrcode_payment::class.java)
                intent.putExtra("url",urlBytes)
                startActivity(intent)
                finish()
              }
            }
```

```
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_paied)

        number = findViewById(R.id.displayAnswer) as TextView
    }
}
```

3.5 qrcode_payment.kt

```
package com.example.client_side
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.net.UrlQuerySanitizer
import android.os.AsyncTask
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.os.CountDownTimer
import android.os.Handler
import android.util.Base64
import android.util.Log
import android.view.View
import android.widget.ImageView
import android.widget.Toast
import com.github.kittinunf.fuel.httpGet
import kotlinx.android.synthetic.main.activity_qrcode_payment.*
import kotlinx.android.synthetic.main.total_price.*
import org.eclipse.paho.android.service.MqttAndroidClient
```

```
import org.eclipse.paho.client.mqttv3.*
import org.jetbrains.anko.doAsync
import org.jetbrains.anko.uiThread
import org.json.JSONObject
import java.io.ByteArrayInputStream
import java.io.InputStreamReader
import java.lang.Exception
import java.net.HttpURLConnection
import java.net.URL
import java.util.*
import kotlin.concurrent.thread

class qrcode_payment : AppCompatActivity() {
    companion object {
        var absPrice:Int = 0
    }
    var client:MqttAndroidClient? = null
    fun onGoBack(v:View?) : Unit
    {
        String.format("%s/product/remove?machine=machine1",MainActivity.firebaseroot)
            .httpGet()
            .responseString {req,res,result->
                if(res.statusCode == 200)
                {



                    var data:String = result.get()
                    var log:String = JSONObject(data).getString("log")
                    Log.i("log",log)
                    MainActivity.check = MainActivity.totalPrice
```

```
                var message = MqttMessage()
                message.payload = "machine1".toByteArray()
                client!!.publish("machine,delete",message)
                goToHome("gohome")
            }
        }
    }
    fun goToHome(target:String):Unit
    {
        var intent:Intent? = null
        when(target)
        {
            "gohome" -> {intent = Intent(this,MainActivity::class.java)}
            "done" -> {intent = Intent(this,bottle::class.java)}
        }
        MainActivity.shoppingList = arrayOf(arrayOf(0,0),arrayOf(0,0),arrayOf(0,0))
        MainActivity.order = ArrayList<Int>()
        MainActivity.totalPrice = 0
        client!!.unregisterResources()
        client!!.close()
        client!!.disconnect()
        startActivity(intent)
        finish()
        disconnect()
}
    fun connect():Unit
    {
        var clientId:String = MqttClient.generateClientId()
        client = MqttAndroidClient(this,"tcp://m16.cloudmqtt.com:17516",clientId)
        Log.i("myqtt","connection starting")
```

```
Log.i("mymqtt",client.toString())
var options:MqttConnectOptions = MqttConnectOptions()
options.mqttVersion = MqttConnectOptions.MQTT_VERSION_3_1
options.isCleanSession = false
options.isAutomaticReconnect = true
options.userName ="djmvfzne"
options.password ="5rE2BX7ctJKi".toCharArray()
var token:IMqttToken? = client!!.connect(options)
token!!.actionCallback = object :IMqttActionListener{
    override fun onSuccess(asyncActionToken: IMqttToken?) {
        Log.i("mymqtt","Connected Success")

        subscribe("machine1")
        client!!.setCallback(object :MqttCallback{
            override fun messageArrived(topic: String?, message: MqttMessage?) {
                if(topic!!.equals("machine1"))
                {
                    MainActivity.check = 0
                    Toast.makeText(this@qrcode_payment,"Your payment was
successfully",Toast.LENGTH_LONG).show()
                    Log.i("currentprice",message!!.toString())
                    absPrice = message!!.toString().toInt()
                    goToHome("done")


                }
            }

            override fun connectionLost(cause: Throwable?) {
            }
```

```
                override fun deliveryComplete(token: IMqttDeliveryToken?) {
                }

            })

        }

        override fun onFailure(asyncActionToken: IMqttToken?, exception:
Throwable?) {
        }

    }
}
fun disconnect():Unit
{
    var disconToken:IMqttToken = client!!.disconnect()
    disconToken.actionCallback = object:IMqttActionListener{
        override fun onSuccess(asyncActionToken: IMqttToken?) {
            Log.i("mymqtt","disconnect was successfully")
        }

        override fun onFailure(asyncActionToken: IMqttToken?, exception:
Throwable?) {
            Log.i("mymqtt","sth went wrong \t"+exception.toString())
        }

    }
}
fun subscribe(topic:String):Unit
```

```
    {
        var qos:Int = 1
        var subToken:IMqttToken = client!!.subscribe(topic,qos)
        subToken.actionCallback = object :IMqttActionListener{
            override fun onSuccess(asyncActionToken: IMqttToken?) {


            }


            override fun onFailure(asyncActionToken: IMqttToken?, exception:
Throwable?) {
            }

        }
    }
    override fun onCreate(savedInstanceState: Bundle?) {

        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_qrcode_payment)
        var qrcodeView = findViewById<ImageView>(R.id.qrcode)
        var newIntent = intent
        var urlBytes = newIntent.getStringExtra("url")
        var stream =  ByteArrayInputStream(Base64.decode(urlBytes, Base64.DEFAULT))
        var qrcode = BitmapFactory.decodeStream(stream)
        Log.i("qrcode",qrcode.toString())
        qrcodeView.setImageBitmap(qrcode)
        connect()


    }
}
```

4. Polling SMS

```
package com.example.polling_sms
import android.content.pm.PackageManager
import android.database.Cursor
import android.net.Uri
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.support.v4.app.ActivityCompat
import android.support.v4.content.ContextCompat
import android.util.Log
import android.view.*
import android.widget.*
import com.github.kittinunf.fuel.httpGet
import org.json.JSONObject

class MainActivity : AppCompatActivity() {
    var bodies = ArrayList<Word>();
    var totalEmail:Int = 0;
    var token:String = "";
    var root:String = "http://us-central1-apifunc-47c8d.cloudfunctions.net/api"
    var url:String = String.format("%s/product/paied",root);
    var getToken = String.format("%s/token/gettoken",root);
    override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out
String>, grantResults: IntArray) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if (grantResults.size > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED)
        {
            myRequest(true);
```

```
    }
}
fun checkSMS(cursor:Cursor) : Int
{
    if(cursor.count<0) return -1;
    bodies.clear();
    var count = 0;
    cursor.moveToFirst();
    for(idx in 0..(cursor.count-1))
    {
        var msg:String = cursor.getString(cursor.getColumnIndex("body"));
        var sender:String = cursor.getString(cursor.getColumnIndex("address"));
        if(sender == "KBank") {
            var p = "(เงินเข้า)(\\d?\\d?\\d?\\d\\.\\d\\d)".toRegex()
            val m = p.find(msg)
            if (m != null)
            {
                count += 1
                var (_,text) = m!!.destructured
                bodies.add(Word(text,sender))
            }
        }
        cursor.moveToNext()

    }
    return count

}
fun poolPayment(cursor:Cursor,first:Boolean) : Unit
{
```

```
        if(first)
        {
            totalEmail = checkSMS(cursor)
            return
        }
       var count = checkSMS(cursor)
        if(totalEmail != count)
        {
            totalEmail = count
            var money =   bodies.get(0).mainText
            var paymenturl:String =
String.format("%s?token=%s&price=%s",url,token,money)
              Log.i("token",paymenturl)
              paymenturl
              .httpGet()
              .responseString { request, response, result ->
                 if(response.statusCode == 200)
                 {
                    var results = result.get()
                    var jsonObject = JSONObject(results)
                    var log= jsonObject.getString("log")

                    Log.i("payment_log",log)
                 }
              }


       }
```

```
    }
    fun myRequest(first:Boolean):Unit
    {
        val cursor = getContentResolver().query(Uri.parse("content://sms/inbox"), null,
"address LIKE 'KBank'", null, null);
        if(cursor != null) {
          poolPayment(cursor,first);
        }
    }
    override fun onCreateOptionsMenu(menu: Menu?): Boolean {
        var menuinflater = getMenuInflater();
        menuinflater.inflate(R.menu.menu, menu);
        return super.onCreateOptionsMenu(menu)
    }

    override fun onOptionsItemSelected(item: MenuItem): Boolean {

        when (item.itemId)
        {
            R.id.mytest1->{
                Toast.makeText(this,"hello world",Toast.LENGTH_LONG).show();
                return true;
            }
            else -> return super.onOptionsItemSelected(item);
        }

    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

```
        setContentView(R.layout.activity_main);
        if (ContextCompat.checkSelfPermission(
                this,
                android.Manifest.permission.READ_SMS
            ) != PackageManager.PERMISSION_GRANTED
        ) {
            val mySmsPermission = Array<String>(1, {
android.Manifest.permission.READ_SMS });
            ActivityCompat.requestPermissions(this, mySmsPermission, 1);
        } else {
            myRequest(true);
        }
        var listView = findViewById<ListView>(R.id.listView);
        val arrayAdepter = object:ArrayAdapter<Word>(this,
android.R.layout.simple_list_item_2, android.R.id.text1,bodies ){
            override fun getView(position: Int, convertView: View?, parent: ViewGroup):
View {
                var view = super.getView(position, convertView, parent)
                var text1 = view.findViewById<TextView>(android.R.id.text1);
                var text2 = view.findViewById<TextView>(android.R.id.text2);
                text1.setText(String.format("จำนวนเงิน %s
บาท",bodies.get(position).mainText));
                text2.setText(bodies.get(position).subText);
                return view;
            }
        }

        getToken
         .httpGet()
         .responseString { request, response, result ->
```

```
            if(response.statusCode == 200)
            {
               var results = result.get();
               var jsonObject = JSONObject(results);
                token = jsonObject.getString("token");
                Log.i("token",token);
            }
        }
        listView.adapter = arrayAdepter;
        var handler = Handler();
        var r = object:Runnable {
            override fun run() {
                myRequest(false)
                arrayAdepter.notifyDataSetChanged();
                handler.postDelayed(this,1000);
            }
        }

        handler.postDelayed(r,1000);



    }
}

class Word(var mainText:String,var subText:String)
{}
```

5. Controller

```
#include "Arduino.h"
#include <ESP8266WiFi.h>
#include "ESP8266HTTPClient.h"
#include "PubSubClient.h"
#include "network_handler.hpp"
#include "string.h"

extern float price;
extern int state;
extern int on;
extern PubSubClient client;
extern PubSubClient bottleHandler;
const char *NetworkHandler::bssid="althenia";
const char *NetworkHandler::password="741455rE";
const char *NetworkHandler::mqtt_server="m16.cloudmqtt.com";
const char *NetworkHandler::mqtt_user="djmvfzne";
const char *NetworkHandler::mqtt_password="5rE2BX7ctJKi";
const char *NetworkHandler::proxy="http://us-central1-apifunc-
47c8d.cloudfunctions.net/";

void NetworkHandler::wifi_setup()
{
delay(1000);
Serial.println();
Serial.println("Connecting to : ....");
Serial.println(NetworkHandler::bssid);
WiFi.begin(NetworkHandler::bssid,NetworkHandler::password);
while(WiFi.status() != WL_CONNECTED)
{
```

```
  delay(1000);

}
Serial.println("WiFi connected");
Serial.print("IP address : ");
Serial.println(WiFi.localIP());

}
void NetworkHandler::reconnect(PubSubClient client)
{
 while(!client.connected())
 {
   String clientId = "ESP8266Client-";
   clientId+= String(random(0xffff),HEX);

if(client.connect(clientId.c_str(),NetworkHandler::mqtt_user,NetworkHandler::mqtt_pass
word))
   {
     Serial.println("connected to mqtt borker");
     client.subscribe("machine1");
   }
   else
   {
     Serial.println("failed to connected will try again in 5 sec");
     delay(5000);
   }

 }
}
```

```
void NetworkHandler::callback(char* topic,byte* payload,unsigned int length)
{
  Serial1.println("message is ");
  char str[length];
  for(int i=0;i<length;i++)
  {
    str[i] = ((char)payload[i]);
  }
  String results(str);
  price = (float)results.toInt();

  client.disconnect();
  NetworkHandler::bottleReconnect(bottleHandler);
  state = 1;

}
void NetworkHandler::bottleCallback(char* topic,byte* payload,unsigned int length)
{
 char str[length] ;
 for(int i=0;i<length;i++)
 {
   str[i] = (char)payload[i];
 }
  String parsedString(str);
  on = parsedString.toInt();
  digitalWrite(16,on);
}
void NetworkHandler::bottleReconnect(PubSubClient client)
{
  while(!client.connected())
```

```
  {
    Serial.println("start to connect to local");
    String clientId = "Bottle";
    if(client.connect(clientId.c_str()))
    {
      Serial.println("login to local broker is successful");
      client.subscribe("currentValue");
    }
    else
    {
      Serial.println("failed to connected will try again in 5 sec");
      delay(5000);
    }
  }

}

void ICACHE_RAM_ATTR onTimerISR(){

    digitalWrite(14,!(digitalRead(14)));  //Toggle LED Pin
    timer1_write(25000000);//12us
}
```

ภาคผนวก ข.

Datasheet

Ultrasonic Ranging Module HC - SR04

Product features:

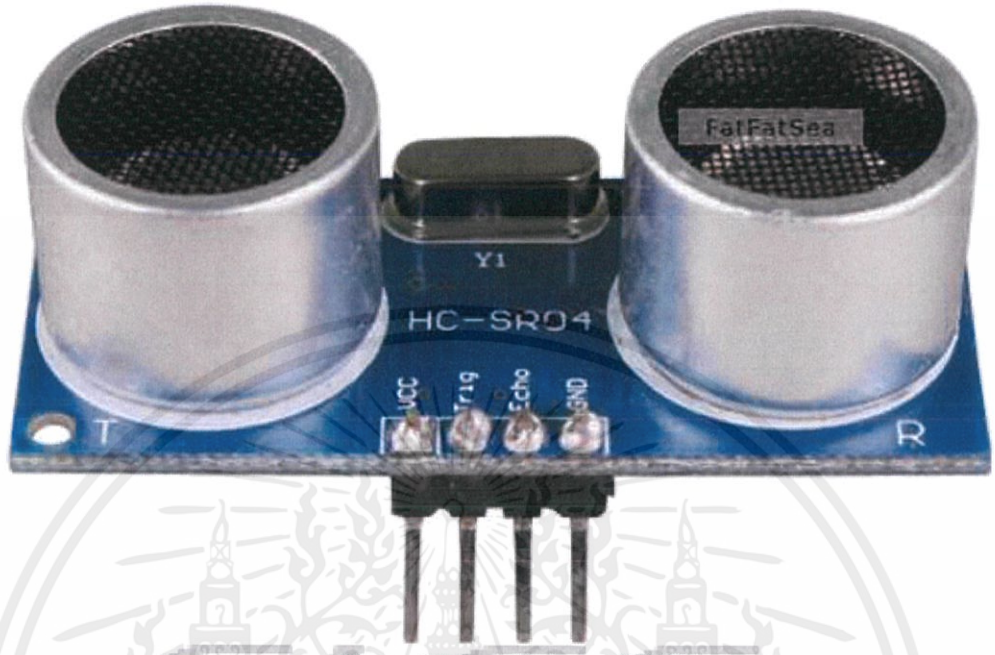
Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

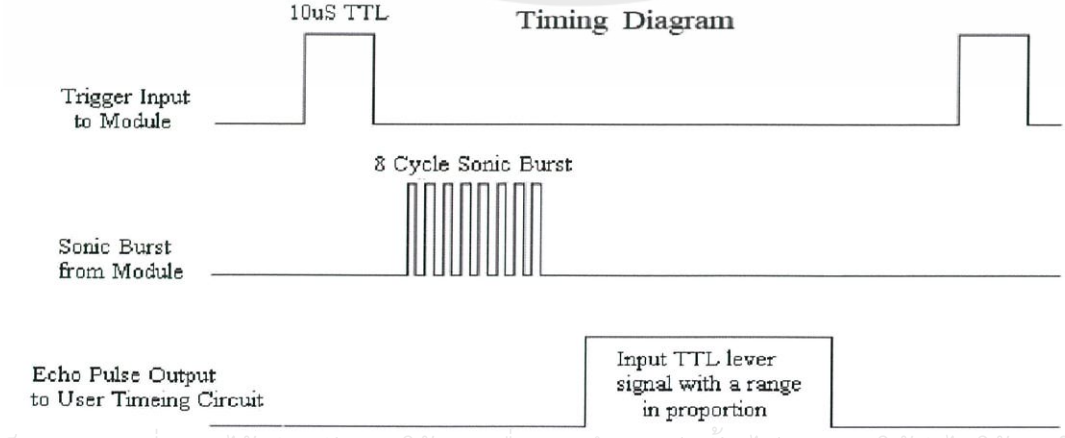
Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**



Vcc Trig Echo GND

Timing diagram

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



Attention:

- The module is not suggested to connect directly to electric, if connected electric, the GND terminal should be connected the module first, otherwise, it will affect the normal work of the module.
- When tested objects, the range of area is not less than 0.5 square meters and the plane requests as smooth as possible, otherwise ,it will affect the results of measuring.

www.Elecfreaks.com