โครงงาน 814/2562 (วศบ.อุตสาหการ)

การพัฒนาระบบตรวจสอบตำแหน่งการวางชิ้นงานบนสายพาน

นายจิรกร ศรีโปธา รหัสนักศึกษา 590612045

นายศุภกิจ เหล่าสามารถ รหัสนักศึกษา 590612095

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมอุตสาหการ

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

ปีการศึกษา 2562

หัวข้อโครงงาน	การพัฒนาระบบตรวจสอบตำแหน่งการวางชิ้นงานบนสายพาน
โดย	นายจิรกร ศรีโปธา รหัสนักศึกษา 590612045
	นายศุภกิจ เหล่าสามารถ รหัสนักศึกษา 590612095
ภาควิชา	วิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
อาจารย์ที่ปรึกษา	ผศ.ดร.อรรถพล สมุทคุปติ์
ปีการศึกษา	2562

ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ อนุมัติให้นับโครงงานนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

กรรมการสอบโครงงาน

..... ประธานกรรมการ

(ผศ.ดร.อรรถพล สมุทคุปติ์)

..... กรรมการ

(รศ.ดร.นิวิท เจริญใจ)

..... กรรมการ

(รศ.ดร.คมกฤต เล็กสกุล)

กิตติกรรมประกาศ

โครงงานวิจัยนี้สามารถสำเร็จไปได้ด้วยดี โดยได้รับคำปรึกษาจาก ผศ.ดร.อรรถพล สมุทคุปติ์ อาจารย์ที่ปรึกษาโครงงาน ซึ่งเป็นผู้มอบความรู้ คำแนะนำ แนวทางการศึกษา ตลอดจนให้ความกรุณาในการตรวจทานแก้ไขโครงงานวิจัยนี้จนทำให้ประสบความสำเร็จอย่างสมบูรณ์ ทางผู้ทำโครงงานจึงต้องขอขอบพระคุณเป็นอย่างสูงมา ณ โอกาสนี้

นอกจากนี้ต้องขอขอบพระคุณ คณาจารย์ในภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ทุกท่าน โดยเฉพาะ รศ.ดร.คมกฤต เล็กสกุล รศ.ดร.นิวิท เจริญใจ ผศ.ดร.กรกฎ ใยบัวเทศ ทิพยาวงศ์ และผศ.ดร.วรพจน์ เสรีรัฐ ที่ได้ให้ความรู้จากการเรียนคำแนะนำในการแก้ปัญหา และแนวทางในการทำโครงงาน แก่ผู้วิจัยและตลอดจนบุคลากร เจ้าหน้าที่ ในภาควิชาวิศวกรรมอุตสาหการทุกท่านที่ให้คำแนะนำ เสนอแนวทาง และให้ความช่วยเหลือในการทำโครงงานด้วยดีตลอดมา

ขอขอบพระคุณ คุณวุฒินันท์ อินทยศ คุณนรินทร์ จักร์ปั่นและคุณณัฐวุฒิ รินโน ครูห้องปฏิบัติการภาควิชาอุตสาหการ คณะวิศวกรรรมศาสตร์ มหาวิทยาลับเชียงใหม่ ที่คอยช่วยเหลือและให้ความรู้ด้านการขึ้นรูปชิ้นงาน ตลอดจนให้คำแนะนำแก้ไข ซึ่งทำให้โครงงานนี้ประสบความสำเร็จสมบูรณ์

ขอขอบพระคุณ อ.ดร.เกษมสิทธิ์ ตียพันธ์ อาจารย์ภาควิชาคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ ที่ให้แนะนำชี้แนะการเลือกใช้บอร์ดและเซนเซอร์ ที่ใช้ง่ายและไม่ซับซ้อนสำหรับมือใหม่

ขอขอบพระคุณ ผศ.มานะ แซ่ด่าน อาจารย์ภาควิชาเครื่องกล คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ ที่ให้คำปรึกษาวิธีการส่งข้อมูลจากเครื่องจักรขึ้นระบบอินเตอร์เน็ต

ขอขอบพระคุณ อ.ดร.สำราญ หม่อมพกุล อาจารย์ภาควิชาภาษาอังกฤษ คณะมนุษยศาสตร์ มหาวิทยาลัยเชียงใหม่ ที่ให้คำปรึกษาการแปลภาษาอังกฤษ และเขียนภาษาอังกฤษ ให้ถูกหลัก

ขอขอบพระคุณ นายภาคภูมิ แสงตัน รุ่นน้องภาควิชาอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่ ที่ให้คำปรึกษาการทำโครงงาน

ขอขอบพระคุณ บิดา มารดา ญาติพี่น้อง ตลอดจนรุ่นพี่ รุ่นน้อง และเพื่อนๆ ในภาควิชาวิศวกรรมอุตสาหการทุกท่าน ที่ได้เป็นกำลังใจ และให้การสนับสนุนในด้านการศึกษา และการดำเนินชีวิตตลอดมา

ขอขอบพระคุณ บริษัท นอร์ธเทิร์น อาร์ต โปรดักท์ จำกัด ที่สละเวลาช่วยจัดหาอุปกรณ์ในการทำโครงงานและราคาที่เหมาะสมกับการทำวิจัยในครั้งนี้

สุดท้ายนี้ทางผู้วิจัยทำหวังเป็นอย่างยิ่งว่า ความรู้จากโครงงานเล่มนี้จะสามารถเป็นประโยชน์ต่อผู้ที่สนใจในด้านการออกแบบเครื่องจักรอัตโนมัติ หากมีส่วนใดที่บกพร่องหรือมีความผิดพลาดประการใด ทางผู้จัดทำต้องขออภัยเป็นอย่างสูง และขอน้อมรับคำเสนอแนะอันเป็นประโยชน์ทุกประการ

จิรกร ศรีโปธา
ศุภกิจ เหล่าสามารถ

หัวข้อโครงงาน	การพัฒนาระบบตรวจสอบตำแหน่งการวางชิ้นงานบนสายพาน
โดย	นายจิรกร ศรีโปธา รหัสนักศึกษา 590612045
	นายศุภกิจ เหล่าสามารถ รหัสนักศึกษา 590612095
ภาควิชา	วิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่
อาจารย์ที่ปรึกษา	ผศ.ดร.อรรถพล สมุทคุปติ์
ปีการศึกษา	2562

บทคัดย่อ

งานวิจัยนี้มุ่งเน้นศึกษาการจำลองตรวจสอบความผิดพลาดการวางตำแหน่งของกล่องบรรจุภัณฑ์ 5x7x5 เซนติเมตร โดยใช้แขนกล Dobot ในการวางตำแหน่งกล่องบรรจุภัณฑ์ บนสายพานในกระบวนการผลิต โดยจะมีการเก็บค่าผ่านราสเบอร์รี่ พาย (Raspberry Pi) ที่เชื่อมต่อกับตัวเซนเซอร์ (Sensor) โดยจะมีเซนเซอร์อินฟาเรด (Sensor Infrared) อยู่บริเวณทางซ้าย และทางด้านขวาของสายพานเป็นตัวตรวจสอบการวางตำแหน่งของแขนกล Dobot ตัวที่ 1 และคัลเลอร์ดีเทคเตอร์เซนเซอร์ (Color Detector Sensor), โฟโตอิเล็คทริกเซนเซอร์ (Photo Electric Sensor) จะตรวจสอบการปิดฝากล่องให้สนิทของแขนกล Dobot ตัวที่ 2 หากเซนเซอร์ (Sensor) ตรวจจับได้ว่าไม่เป็นไปตามเงื่อนไขก็จะถูกปัดลงกล่อง เพื่อแสดงว่าการวางตำแหน่งบริเวณนั้นไม่เป็นไปตามเงื่อนไขที่ผู้ผลิตได้กำหนดขึ้นมา ซึ่งการศึกษานี้จะทำให้ผู้ผลิตได้ทราบถึงบริเวณที่เกิดปัญหาขึ้นเป็นส่วนใหญ่ทำให้ผู้ผลิตแก้ไขปัญหาได้ตรงจุด และประหยัดเวลาในการหาสาเหตุและบริเวณที่เกิดปัญหาขึ้น

โดยจะมีเซนเซอร์อินฟาเรด (Sensor Infrared) ในการตรวจจับว่า กล่องบรรจุภัณฑ์ ได้อยู่ตำแหน่งตรงกลางของสายพานหรือไม่ ถ้าไม่แขนกล Dobot จะทำการปัดลงสู่กล่องที่ 1 ถ้าใช่สายพานก็จะทำงานต่อ แล้วจะมีโฟโตอิเล็คทริกเซนเซอร์ (Photo Electric Sensor) สั่งให้สายพานหยุดเมื่อกล่องบรรจุภัณฑ์ได้มาถึงตำแหน่งนั้น แขนกล Dobot ตัวที่ 2 จะทำการปิดฝากล่องบรรจุภัณฑ์จากนั้นคัลเลอร์ดีเทคเตอร์เซนเซอร์ (Color Detector Sensor) จะตรวจสอบว่าการปิดฝากล่องบรรจุภัณฑ์ได้เข้าตำแหน่งที่ต้องการหรือไม่ ถ้าไม่แขนกล Dobot จะทำการปัดลงสู่กล่องที่ 2 ถ้าใช่สายพานจะทำงานต่อจนสิ้นสุดกระบวนการ

Project Title	Development of Part Placement Checking System on Conveyer Belt
Name	Mr. Jirakorn Sripota Code 590612045
	Mr. Supakit Laosamart Code 590612095
Department	Industrial Engineering, Faculty of Engineering, Chiang Mai University
Project Advisor	Assistant Professor Dr. Uttapol Smutkupt
Academic Year	2019

ABSTRACT

The purpose of this project is to create an automatic machine that uses a detect system that sends Realtime data to a robot automation system by Raspberry Pi to pack box. At present, the manufacturing industry has properly used technology to improve manufacturing processes in most companies. The researcher has recognized the importance of technology as it can be uses as a guideline to improve the production processes.

Research methodology involves learning knowledge of IoT and Robotic system, designing function of machine and IoT, programing Robot and IoT, drawing machines in program computer, and assembling machinery and testing to meet objectives. From testing with the packaging box and detect product in conveyor, it was found that the sensor can detect and reject 20time of position box in a robot automation system for Realtime.

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญภาพ (ต่อ)

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มาของปัญหาที่ทำโครงงาน

ในปัจจุบันโรงงานอุตสาหกรรมได้เล็งเห็นถึงความสำคัญของระบบอัตโนมัติซึ่งระบบอัตโนมัตินั้นสามารถเพิ่มกาลังการผลิตของบริษัทซึ่งแขนกล เป็นส่วนหนึ่งของระบบอัตโนมัติที่สามารถควบคุมและช่วยลดความเสียหายจากการทำงานโดยแรงงานมนุษย์อีกทั้งการทำงานที่สะดวก ง่าย และหุ่นยนต์ยังช่วยปฏิบัติงาน เพื่อความรวดเร็ว แม่นยำ สามารถบริหารจัดการให้เกิดประโยชน์สูงสุด การใช้คนทำงานน้อยลง ซึ่งช่วยลดต้นทุนในการบริหารจัดการ และมีมาตรฐานการควบคุมกระบวนการผลิตอย่างเข้มงวดทุกขั้นตอน รวมถึงการดูแลด้านสิ่งแวดล้อม โดยสามารถลดการใช้พลังงานความร้อน ความเย็น แสงสว่าง และไฟฟ้าจากเครื่องจักรที่ใช้ในกระบวนการผลิตรวม ยังสามารถผลิตได้แบบต่อเนื่องจนกว่าเครื่องจะมีปัญหาหรือจนกว่าจะมีการหยุดจ่ายพลังงาน (ไฟฟ้า) ลดปัญหาการขาดหรือลาของพนักงาน โดยการนำคอมพิวเตอร์เข้ามาช่วยควบคุม หรือการทำงานในทุกขั้นตอนอย่างไรก็ตามแขนกลอาจหยิบ หรือวางชิ้นส่วนของผลิตภัณฑ์ผิดตำแหน่งตำแหน่ง ซึ่งทำให้เกิดของเสีย เพราะฉะนั้นจึงได้มีระบบเซนเซอร์ในการตรวจสอบ เพื่อบ่งบอกว่ามีความผิดปกติของตำแหน่งชิ้นงานมาแล้ว จากนั้นระบบจะทำการแก้ไขและเก็บข้อมูลผ่านระบบ Internet of Things (IoT) จะเข้าไปช่วยแก้ปัญหาในขบวนการควบคุมต่างๆ ที่มีผลกระทบต่อระบบการผลิตอาหารในโรงงานอุตสาหกรรม เนื่องจากระบบเซนเซอร์จะมีการติดตามตลอดเวลา และรายงานผลแบบออนไลน์ตลอด 24 ชั่วโมง ส่งผลให้กระบวนการผลิตสามารถทำงานได้เต็มประสิทธิภาพมากยิ่งขึ้น ลดการสูญเสียค่าใช้จ่าย ลดความเสียหายของผลิตภัณฑ์ หรือสินค้า และยังสามารถลดการใช้แรงงานคนจัดการคลังสินค้า โดยระบบดังกล่าวจะช่วยลดต้นทุนการสูญเสีย และเพิ่มจำนวนการผลิตได้มากกว่าที่ผ่านมา การใช้งานแขนกลสามารถส่งเสริมความปลอดภัยของกระบวนการผลิตได้เป็นอย่างดีด้วยระบบเซนเซอร์ตรวจจับการทำงาน ทดแทนแรงงานมนุษย์ในการปฏิบัติงานที่มีความเสี่ยงต่อสุขภาพ และความปลอดภัยของผู้ปฏิบัติงานในพื้นที่ โดยเฉพาะการปฏิบัติงานที่มีความเสี่ยงต่อความสูญเสีย

เป็นหลักไม่ว่าจะเป็นการยกของหนัก การขนย้ายวัสดุในพื้นที่จำกัด หรือการทำงานในพื้นที่ที่มีสภาพแวดล้อมอันตราย ทำให้ระบบออโตเมชัน และหุ่นยนต์ถูกนำมาใช้กันอย่างแพร่หลาย เพื่อจะแสดงผลข้อมูลไปให้พนักงานที่เกี่ยวข้องในบริษัทได้เห็น เพื่อที่จะได้ดำเดินการแก้ไขต่อไป

1.2 วัตถุประสงค์

1.2.1 เพื่อประยุกต์ใช้เซนเซอร์ในการตรวจสอบตำแหน่งของชิ้นงาน

1.2.2 เพื่อสร้างระบบเตือนโดยผ่านระบบ Internet of Things (IoT) ให้แสดงข้อมูลความผิดปกติ

1.3 ขอบเขตการศึกษา

1.3.1 ศึกษากระบวนการหยิบกล่องว่างบนสายพาน

1.3.2 ศึกษาความผิดปกติตำแหน่งการว่างกล่อง

1.3.3 การประยุกต์ใช้ระบบ Internet of Things (IoT) ผ่านราสเบอร์รี่ พาย (Raspberry Pi)

1.3.4 การแสดงผลข้อมูลผ่านทางคอมพิวเตอร์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ความสะดวก และความแม่นยำในการเก็บข้อมูล

1.4.2 สามารถเข้าถึงข้อมูลได้ง่ายโดยผ่านระบบ Internet of Things (IoT)

1.4.3 สามารถนำไปประยุกต์ใช้กับระบบอื่น

บทที่ 2

หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 หลักการและทฤษฎีของราสเบอร์รี่พาย

2.1.1 ราสเบอร์รี่ พาย (Raspberry Pi)

ราสเบอร์รี่ พาย (Raspberry Pi) ดังภาพ 2.1 เป็นคอมพิวเตอร์ขนาดเล็กพัฒนาขึ้น เพื่อนำพลังด้านดิจิตอลเข้าสู่ผู้ใช้งานทั่วโลก ดังนั้นผู้ใช้งานสามารถทำความเข้าใจและสร้างโลกดิจิทัลเพิ่มขึ้นได้โดยง่าย สามารถแก้ปัญหาต่างๆ ซึ่งราสเบอร์รี่ พาย (Raspberry Pi) เป็นคอมพิวเตอร์ที่มีประสิทธิภาพสูง ราคาประหยัด และมีประสิทธิภาพสูงที่ผู้คนใช้เพื่อเรียนรู้ในการแก้ปัญหา ซึ่งสามารถประหยัดค่าใช้จ่ายในการเขียนโปรแกรม

ราสเบอร์รี่ พาย (Raspberry Pi) สามารถเชื่อมต่อระบบเครือข่ายแบบใช้สาย หรือไร้สายได้ ทำให้กลายเป็นอุปกรณ์ Internet of Things (IoT) โดยสมบูรณ์ สามารถนำไปประยุกต์ใช้ เพื่อเชื่อมต่อกับตัวตรวจจับเซนเซอร์ (Sensor) ในการเก็บข้อมูลตามต้องการ โดยระบบปฏิบัติการที่ใช้นั้น คือ ราสเบียน (Raspbian) ซึ่งเป็นระบบปฏิบัติการลินุกซ์เป็นฐานถูกปรับแต่งมาใช้กับ ราสเบอร์รี่พาย (Raspberry Pi) โดยเฉพาะระบบปฏิบัติการ ติดตั้งผ่าน Micro SD Card สามารถตั้งค่าเป็นเครื่องแม่ข่ายและใช้งานบริการต่าง ๆ เช่น Web Server FTP Server เป็นต้น

- CPU: Quad-core 1.2 GHz ARM Cortex-A53 แบบ 64 bits
- GPU: Broadcom Video Core IV @ 400 MHz
- Memory ขนาด 1 GB (LPDDR2-900 SDRAM)
- หน่วยความจุแบบ MicroSD
- 4 USB ports
- 1 Ethernet port
- 802.11n Wireless LAN
- Bluetooth 4.0
- รองรับ HDMI/Composite ผ่านทาง RCA Jack
- GPIO 40 pins

ภาพ 2.1 บอร์ดราสเบอร์รี่ พาย (Raspberry Pi)

ที่มา : https://www.scimath.org.

โปรแกรม Qt ที่ใช้ในการเขียนโค้ด (code) ในราสเบอร์รี่ พาย (Raspberry Pi) เป็นเครื่องมือในการสร้างแอปพลิเคชั่น และ GUI ซึ่งสามารถทำงานบน Desktop PC Smart Phone และ Embedded System สามารถทำงานได้หลายระบบปฏิบัติการ (OS) หรือ เรียกว่า Cross - platform แปลความได้ว่า เมื่อเรามีโปรแกรมที่ทำงานบนระบบปฏิบัติการ (OS) หนึ่ง เราไม่จำเป็นต้องเขียนโปรแกรมใหม่ สามารถนำโปรแกรมไปคัมไพล์ (Compile) เพื่อให้สามารถทำงานบนระบบปฏิบัติการ (OS) อื่นได้โดยไม่ต้องแก้โปรแกรมเลย เมื่อโปรแกรมทำงาน หน้าตาของมันจะเปลี่ยนไปตามสิ่งแวดล้อมของระบบปฏิบัติการ (OS) นั้นๆ โดยอัตโนมัติ ดังภาพ 2.2

ภาพ 2.2 โปรแกรม Qt

ที่มา : https://www.thaieasyelec.com

การเขียน GUI ให้กับแอพพลิเคชัน หรือระบบต่าง ๆ ในปัจจุบัน มีทางเลือกหลายทางในการพัฒนา มีเครื่องมือหลายตัวให้เลือกใช้ เช่น VC# และ VB .NET บน WIN CE เป็นต้น แต่สำหรับ Qt แล้ว ถือว่าเป็นอีกทางเลือกหนึ่งที่น่าสนใจ เพราะ สามารถเลือกใช้ API Library ต่างๆ ซึ่งมีมากมายได้

เช่นกัน (เปรียบเทียบได้กับ MSDN จาก Microsoft แต่เป็น Opensource ไม่จำเป็นต้องเริ่มตั้งต้นใหม่ ซึ่งเสียเวลาโดยเปล่าประโยชน์

Qt จะมี API และ Library ต่าง ๆ ที่เขียนด้วยภาษา C++ ทำให้สามารถพัฒนาแอพพลิเคชั่นแบบ GUI และยังสนับสนุนการพัฒนาทั้ง C++ Java Python Perl Pascal และ PHP

การติดตั้ง Qt และ QtCreator IDE บนราสเบอร์รี่ พาย (Raspberry Pi)

- เปิด Terminal หรือ Command-Line ใน ราสเบอร์รี่ พาย (Raspberry Pi)
- รันคำสั่งต่อไปนี้เพื่ออัพเดท Core Library ของ Pi ก่อน “sudo apt-get update”
- รันคำสั่งนี้เพื่อติดตั้ง Qt SDK ก่อน “sudo apt-get install qt5-default”
- รันคำสั่งนี้เพื่อติดตั้ง QtCreator “sudo apt-get install qtcreator”
- รันคำสั่งนี้เพื่อติดตั้ง g++ “sudo apt-get install build-essential g++”
- รันคำสั่งนี้เพื่อติดตั้ง Library สำหรับสื่อสารกับ Serial Port “sudo apt-get install libqt5serialport5-dev”
- เมื่อลงเสร็จแล้ว ให้รันคำสั่ง “qtcreator -noload Welcome” เพื่อเปิด Qt Creator ขึ้นมาเพื่อพร้อมเขียนโปรแกรม

2.1.2 หลักการทำงานของอุปกรณ์

◆ จุดเชื่อมต่อ GPIO

- เชื่อมต่อกับบอร์ด ราสเบอร์รี่ พาย (Raspberry Pi) ผ่านทางคอนเน็กเตอร์ GPIO 40 ขา ดังภาพ 2.3
- มีจุดต่อพอร์ต GPIO 40 ขา ครบทุกขาเป็นแบบ IDC ตัวผู้ระยะห่างระหว่างขา 2.54 มิลลิเมตร
- มีจุดต่อแบบ JST 2 มิลลิเมตร 3 ขา จำนวน 10 ขา ประกอบด้วย GPIO2/SDA, 3/SCL, 4/1-Wire, 5, 14/TxD, 15/RxD, 24, 25, 26 และ 27
- มีสวิตช์กดติดปล่อยดับพร้อมใช้งานต่อกับขา GPIO13
- มีสวิตช์แบบจอยสติ๊ก 4 ทิศทาง และปุ่มกดกึ่งกลาง ต่อกับขา GPIO6, 17, 18, 22 และ 23
- มีจอแสดงผล OLED 0.96 นิ้ว ความละเอียด 128x64 จุด ใช้การเชื่อมต่อแบบบัส I2C
- มี LED 3 สี RGB แบบอนุกรม และโปรแกรมได้ เบอร์ WS2812B ต่อกับขาพอร์ต GPIO12

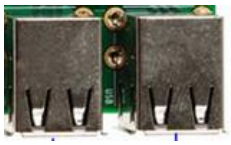
- ขนาด 65 x 56 มิลลิเมตร โดยออกแบบรูปร่างของแผ่นวงจรพิมพ์ ให้มีขนาดระยะห่างของรูยึดช่องว่างต่างๆ เป็นไปตามข้อกำหนดของการออกแบบบอร์ด HAT ของพื้นฐานราสเบอร์รี่ พาย (Raspberry Pi Foundation)

ภาพ 2.3 จุดเชื่อมต่อ GPIO

ที่มา : http://www.mindphp.com

◆ จุดเชื่อมต่อ Port USB

ใช้สำหรับหรือช่องทางในการเชื่อมต่อและสื่อสารระหว่างคอมพิวเตอร์กับอุปกรณ์ภายนอกอื่นๆ ดังภาพ 2.4



ภาพ 2.4 จุดเชื่อมต่อ Port USB

ที่มา : http://www.mindphp.com

◆ ช่อง Audio/Video

Audio Output อยู่ 2 ช่อง ด้วยกันคือผ่านทางช่อง HDMI และแจ๊ค 3.5 มิลลิเมตร ที่เป็นทั้ง Stereo Audio และ Composite Video โดยบอร์ด Raspberry Pi จะเลือกให้โดยอัตโนมัติว่าจะใช้งานช่องไหน ดังภาพ 2.5

ภาพ 2.5 ช่อง Audio / Video

ที่มา : http://www.mindphp.com

◆ จุดเชื่อมต่อ LAN / Internet

ในการเชื่อมต่อ Internet ดังภาพ 2.6

ภาพ 2.6 จุดเชื่อมต่อ LAN / Internet

ที่มา : http://www.mindphp.com

◆ จุดต่อกล้องแบบ Camera Serial interface (CSI)

พอร์ต Camera Serial Interface (CSI) ใช้สำหรับเชื่อมต่อโมดูลกล้อง ดังภาพ 2.7

ภาพ 2.7 จุดต่อกล้องแบบ Camera Serial interface (CSI)

ที่มา : http://www.mindphp.com

◆ จุดเชื่อมต่อ HDMI

ใช้สำหรับเชื่อมต่อกับจอแสดงผล หากเลือกใช้จอมอนิเตอร์ (Monitor) ที่ไม่มีพอร์ต HDMI รองรับต้องใช้ตัวแปลง HDMI to VGA ด้วย หรือเชื่อมต่อสายวีดีโอ RCA ก็ได้เช่นเดียวกัน (เลือกอย่างใดอย่างหนึ่ง) ดังภาพ 2.8

ภาพ 2.8 จุดเชื่อมต่อ HDMI

ที่มา : http://www.mindphp.com

◆ ช่องจ่ายไฟด้วย Micro USB Power

ทำหน้าที่เพื่อจ่ายไฟเลี้ยงให้กับวงจร สามารถเลือกใช้แหล่งจ่ายไฟจากพอร์ต USB ของเครื่องคอมพิวเตอร์ได้ ดังภาพ 2.9

ภาพ 2.9 ช่องจ่ายไฟด้วย Micro USB Power

ที่มา : http://www.mindphp.com

◆ ช่องเสียบ SD card

ตัวนี้จะอยู่ด้านหลังของบอร์ดใช้สำหรับติดตั้งระบบปฏิบัติการ Linux ต้องมีความจุมากกว่า 2 กิกะไบต์ ขึ้นไป แต่แนะนำให้ใช้ ขนาด 4 กิกะไบต์ หรือมากกว่า ดังภาพ 2.10

ภาพ 2.10 ช่องเสียบ SD card

ที่มา : http://www.mindphp.com

◆ พอร์ต Display Serial Interface (DSI)

ใช้สำหรับต่อจอแสดงผล เช่น จอแสดงผลแบบ TFT Touch Screen ดังภาพ 2.11

ภาพ 2.11 พอร์ต Display Serial Interface (DSI)

ที่มา : http://www.mindphp.com

การเขียนโปรแกรมภาษา ไพธอน (Python) สามารถนำไปประยุกต์ใช้งานกับบอร์ดไมโครคอนโทรเลอร์ทั่วไป โดยสามารถติดตั้งอุปกรณ์อื่น เช่น ระบบเปิด/ปิดไฟอัตโนมัติ อุปกรณ์วัดความเอียง ระบบตรวจสอบอุณหภูมิห้องแบบเรียลไทม์ รถยนต์บังคับสำหรับงานด้านต่าง ๆ กังหันลม เครื่องชงชา ระบบควบคุมไฟจราจร ระบบวัดระยะทาง ระบบแจ้งเตือนต่าง ๆ ระบบวัดความเข้มแสง เครื่องนับแต้ม ระบบจัดเก็บข้อมูลจากตัวตรวจจับต่าง ๆ ลงฐานข้อมูล MySQL เป็นต้น แต่จำเป็นต้องมีการซื้อตัวตรวจจับเซนเซอร์ (Sensor) เพิ่มเติมเหมือนกับการใช้งานด้วย อาดุอีโน่ แพลตฟอร์ม (Arduino Platform)

2.2 หลักการและทฤษฎีของเซ็นเซอร์

2.2.1 โฟโต้เซ็นเซอร์ประกอบด้วย 2 ส่วน ดังภาพ 2.12 ได้แก่

- ภาคส่งแสง (Emitter หรือ Transmitter)

- Pulse Modulator คือวงจรอิเล็กทรอนิกส์ที่สร้างพลัส์ซึ่งความถี่ของพลัส์นี้จะเป็นความถี่ของแสงที่จะถูกส่งออกไป

- Amplifier ทำหน้าที่ขยายสัญญาพัลส์ไหม้โวลต์เตจสูงขึ้น

- Opto-diode ทำหน้าที่เปลี่ยนสัญญาณไฟฟ้าที่ได้ให้เป็นแสง ซึ่งมีองค์ประกอบ 2 ส่วน คือแสงอินฟราเรด และแสงที่มนุษย์มองเห็น ซึ่งส่วนใหญ่เป็นแสงสีแดง รองลงมีคือสีเขียว

- เลนส์ (Lense) ทำหน้าที่รวมแสงแล้วส่งออกไป

- ภาครับแสง (Reciever)

- Pre-Amplifier ทำหน้าที่ขยายโวลต์เตจที่รับมาจาก Photo Transister ให้สูงขึ้น

- Synchronizer ทำหน้าที่เปรียบเทียบความถี่ของแสงที่รับมาจาก Pulse Modulator ว่าตรงกัน หรือไม่ หากตรงกันจะส่งเอาพุตออกไป ซึ่งวงจรเหล่านี้ จะช่วยป้อกันแสงรบกวนจากแสงภายนอกทั้งจากแสงแดด และหลอดไฟในห้องทำงาน เพราะความถี่ของแสงที่รบกวนจะไม่ตรงกัน ความถี่ที่ส่งมาจากภาคส่งแสงทำให้สามารถแยกความแตกต่างได้

- Photo Transistor ทำหน้าที่แปลงแสงที่รับเข้ามาให้เป็นสัญญาณไฟฟ้าออกมาเป็นมิลลิโวลต์

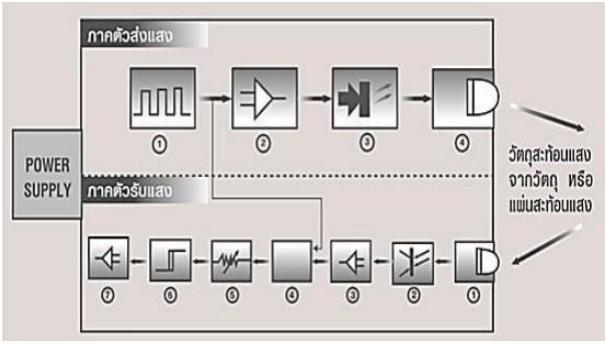
- เลนส์ (Lense) ทำหน้าที่รวมแสงที่เข้ามา

- Sensitivity Adjustment เป็นตัววัดความต้านทานที่ปรับค่าได้เพื่อกำหนดปริมาณแสงที่ได้รับมาว่า ปริมาณเท่าใดจึงจะให้เอาพุตทำงานโดยจะเป็นการปรับค่าโวลต์เตจ เพื่อจะให้วงจรถัดไปคือ Trigger ทำการ ON หรือ OFF

- Trigger คือวงจรที่จะสั่งให้ทำการ ON หรือ OFF จะมีค่า ฮีสเตอร์รีซิส

(Hysterresis) เพื่อป้องกันไม่ให้เอาพุตทำงานบ่อยเกินไป

- Amplifier ทำหน้าที่ขยายสัญญาณให้มีโวลต์เตจสูงขึ้น เพื่อสั่งให้เอาท์พุดทรานซิเตอร์เปลี่ยนสภาวะ

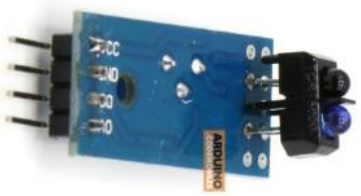


ภาพ 2.12 แสดงองค์ประกอบของโฟโต้สวิตช์ทั้งภาคส่งและภาครับ

ที่มา : http://www.research-system.siam.edu

2.2.2 วงจรตรวจจับวัตถุ

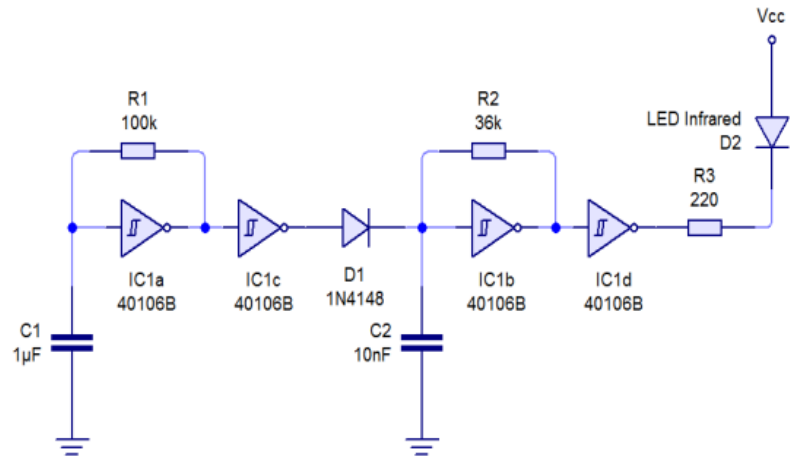
ในวงจรนี้จะมีความพิเศษจากเซ็นเซอร์แบบสะท้อน (Reflectance) ทั่วๆ ไปตรงที่เซ็นเซอร์แบบReflectance จะใช้โฟไดโอดในการรับแสงอินฟาเรด เมื่อมีวัตถุเข้ามาบังแนวแสง ก็จะทำให้ไฟโต้ไดโอดไม่ได้รับแสงปล่อยสถานะทางลอจิกที่เปลี่ยนไป ดังภาพ 2.13



ภาพ 2.13 เซ็นเซอร์ตรวจจับ

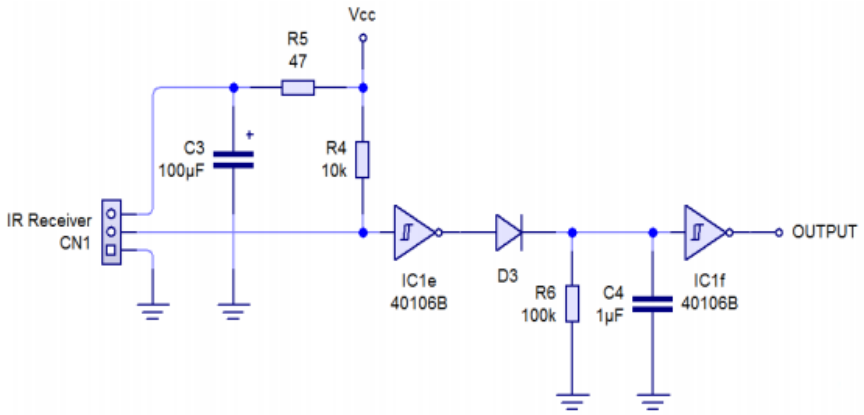
ที่มา : http://www.research-system.siam.edu

2.2.3. รูปวงจรตรวจจับวัตถุความแม่นยำสูงทั้งภาคส่ง และรับ ดังภาพ 2.14 และ2.15



ภาพ 2.14 วงจรภาคส่ง

ที่มา : http://www.research-system.siam.edu



ภาพ 2.15 วงจรภาครับ

ที่มา : http://www.research-system.siam.edu

2.3 หลักการทำงานและระบบของหุ่นยนต์ DOBOT MAGICIAN

หุ่นยนต์ดูบอทเมจิกเชียน (Dobot Magician) ที่นำมาใช้นั้นคือแขนกลหรือหุ่นยนต์ที่สำเร็จรูปมาอยู่แล้ว ไม่จำเป็นที่จะต้องคำนวณค่า coordinate ต่างๆ หรือเขียนโปรแกรมสำหรับการเคลื่อนไหว

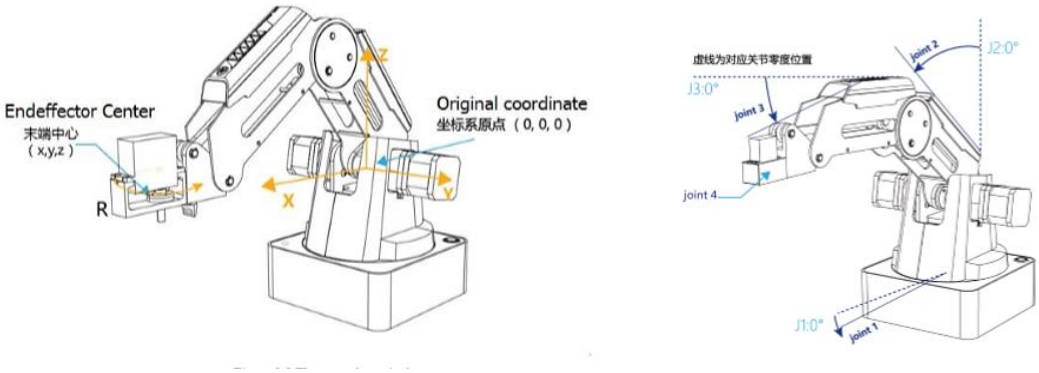
ในแต่ละ จุด (joint) อีกทั้งยังมีการแสดงผลในหน้าจอเป็นตำแหน่งของปลายแขนกล (End Effector) เป็นค่าพิกัด x y z เมื่อมีการหมุนจุด (joint) ต่างๆ ที่ถูกออกแบบมาสำหรับผู้ที่สนใจ และอยากเรียนรู้การควบคุมแขนหุ่นยนต์ ในราคาที่ไม่แพง ซึ่งโรงเรียนและโรงงาน สามารถนำไปประยุกต์ใช้สำหรับสอนมือใหม่ หรือพนักงานให้รู้จัก การควบคุมแขนหุ่นยนต์ให้ทำงานตามที่ต้องการตัวหุ่นดูบอท (Dobot) นั้นสามารถโปรแกรม ให้ทำงานได้หลากหลาย รวมถึงสามารถต่ออุปกรณ์อื่นๆ เพื่อให้สามารถทำงานร่วมกันกับอุปกรณ์ และหุ่นตัวอื่นๆ ได้ สำหรับแขนหุ่นยนต์ดูบอท (Dobot) นั้นมาพร้อมกับโปรแกรมที่เอาไว้สำหรับควบคุม การทำงาน ซึ่งรองรับภาษาในการเขียนโปรแกรมได้หลากหลายภาษา เช่น ภาษา C หรือจะเป็นไพธอน (Python) ก็ได้ รวมไปถึงยังมีโปรแกรมที่เป็นแบบ Block Code สำหรับให้มือใหม่ ที่ต้องการฝึกเขียนโปรแกรมตัวหุ่นยนต์ดูบอทเมจิกเชียน (Dobot Magician) ยังรองรับระบบปฏิบัติการ ROS ซึ่งเป็นระบบหุ่นยนต์แบบโอเพนซอร์ซ (Open source) ที่ได้รับความนิยมเป็นอย่างมาก ดังภาพ 2.16

ภาพ 2.16 Dobot Magician

ที่มา : https://www.siamreprap.com

2.3.1 โครงสร้าง และส่วนประกอบของแขนกล Dobot Structure โครงสร้างของหุ่นยนต์ชนิดนี้จะแบ่งเป็น 2 ประเภท คือส่วนของโครงสร้างการทำงานและโครงสร้างระบบ

◆ โครงสร้างการทำงานนั้นหุ่นยนต์ชนิดนี้จะแบ่งเป็นทั้งหมด 3 แกน การทำงาน และ 4 จุด ต่อการหมุนโดยแกนการทำงานนั้นจะมีแกน X Y Z ตามระนาบมาตรฐานและแต่ละจุดหมุนนั้นเป็น ดังภาพ 2.17



ภาพ 2.17 Dobot Magician axis coordinate system

ที่มา : https://www.dobot.cc

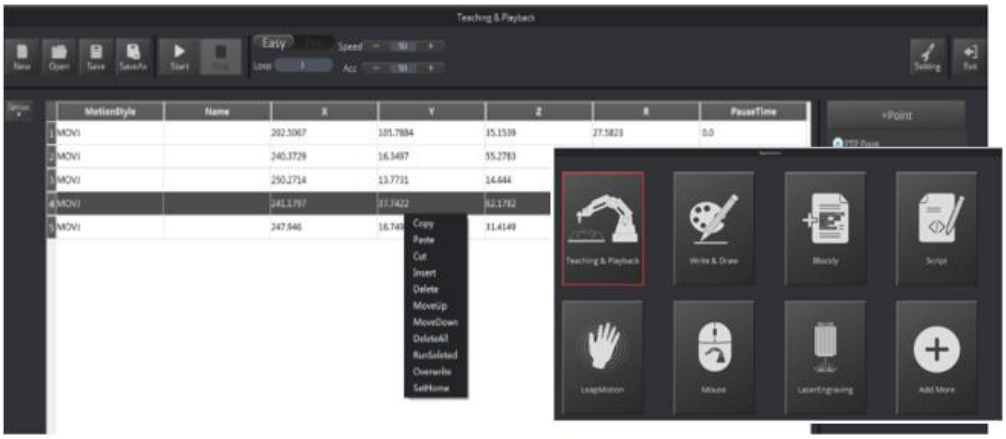
โดยทิศทางการเคลื่อนที่ของ X Y Z นั้นดังภาพ 2.17 นั้นเมื่อปรับ X- หุ่นยนต์จะเลื่อนแขนเข้ามาและเมื่อกด X+หุ่นยนต์จะกางแขนออกไปด้านหน้า Y+ จะหุ่นยนต์จะหันไปทางซ้ายมือและ Y- จะหันไปท้างขวามือ ส่วนแกน Z นั้น Z+ จะทำให้แขนกลยกขึ้นส่วน Z- จะทำให้หุ่นยนต์กดแขนลง ต่อไปเป็นอธิบายของส่วนแกนหมุน (Joint) โดยแต่ละแกนหมุนนั้นจะมีการทำงานที่เหมือนกันคือ ค่าบวกจะทำให้แกนหมุนหมุนตามเข็มนาฬิกาส่วนค่าลบจะทำให้หมุนทวนเข็มนาฬิกา

◆ โครงสร้างของระบบการทำงานในส่วนของระบบการทำงานนั้นจะเป็นการป้อนโปรแกรมเข้าไปผ่านโปรแกรมสำเร็จรูปของตัวหุ่นยนต์เองที่ชื่อว่า Dobot Studio ซึ่งรายละเอียดจะพูดถึงในหัวข้อต่อไป

2.3.2 การโปรแกรม Teaching & Playback เบื้องต้นในหุ่นยนต์ Dobot Magician ดังภาพ 2.18 คือหน้าต่างของโปรแกรมฟังชั่นการทำงานพื้นฐานของโปรแกรม และมีวิธีการโปรแกรมหุ่นยนต์พื้นฐานมีดังต่อไปนี้

- เปิดโปรแกรม Dobot Studio แล้วเชื่อมต่ออุปกรณ์กับคอมพิวเตอร์
- กดปุ่มโฮม (Home) แล้วเข้าโหมด Teaching & Playback

- การจะให้หุ่นยนต์เคลื่อนที่จากที่หนึ่งไปอีกที่หนึ่งต้องกดปุ่มบันทึกตำแหน่งอุปกรณ์ตรงแขนหุ่นยนต์ด้านบน เมื่อกดแล้วจะขึ้นตำแหน่งของอุปกรณ์ตำแหน่งแรกหากต้องการเพิ่มจุดที่สองให้กดอีกครั้งแล้วย้ายตำแหน่งแขนหุ่นยนต์ไปตำแหน่งที่ต้องการแล้วจึงปล่อยโปรแกรมจะกำหนดจุดที่สอง และเมื่อกดเริ่มต้นการทำงาน (Start) หุ่นยนต์จะเริ่มจากตำแหน่งแรกแล้วไปตำแหน่งที่สองต่อกัน



ภาพ 2.18 Dobot Magician programming

ที่มา : https://www.dobot.cc

2.3.3 การเชื่อมต่ออุปกรณ์หุ่นยนต์กับอุปกรณ์เสริมต่างๆ (EIO multiplex function) เมื่อต้องการต่ออุปกรณ์ที่รับอินพุตเข้าระบบเช่นเซ็นเซอร์ต่างๆ ต้องต่อเข้ากับช่อง 4 Pin บนแขน และฐานของหุ่นยนต์เมื่อต่อสายเข้าไปแล้วจากนั้นต้องโปรแกรมฟังชั่นต่างๆ ในโปรแกรม เพื่อควบคุมหุ่นยนต์ ตัวอย่างเช่น การใช้เซ็นเซอร์จับแสง (Photoelectric Sensor) ในการตรวจจับวัตถุที่อยู่บนสายพาน โดยชื่อในช่องต่างๆ จะมีชื่อประจำช่อง และแรงดันไฟฟ้าที่ต่างกันดังนั้นจึงต้องใช้อุปกรณ์ที่เหมาะสมกับแรงดันไฟฟ้า หากใช้ไม่เหมาะสมจะทำให้อุปกรณ์เสียหายได้ ในหุ่นยนต์จะมีช่องสัญญาณพิเศษอยู่อีก 2 ประเภท คือ ช่องที่สามารถต่อ Pulse Width Modulation (PMW) และ Analog to Digital Converter

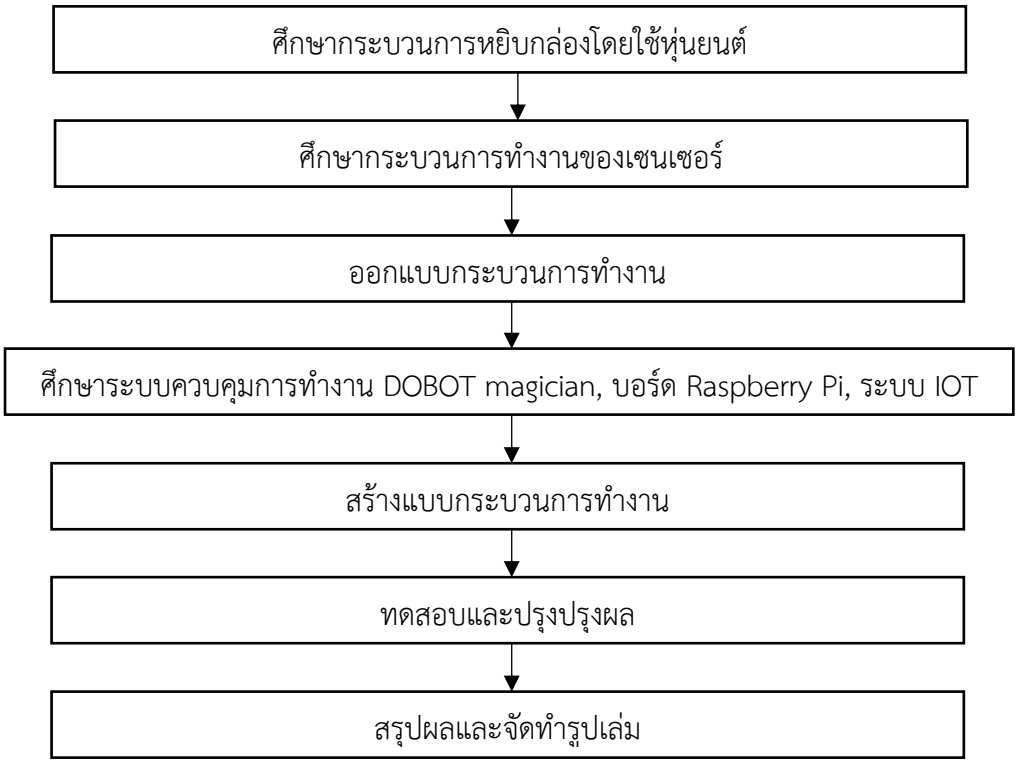
บทที่3

วิธีการดำเนินการ

งานวิจัยนี้มุ่งเน้นเพื่อทำการประยุกต์ใช้ความรู้ทางด้านระบบอัตโนมัติโดยเน้นหุ่นยนต์อุตสาหกรรมเป็นหลักในการสร้างเครื่องจักรอัตโนมัติที่สามารถทดแทนคน และเพิ่มกำลังผลิตได้

3.1 วิธีการและขั้นตอนในการทำงานวิจัย

วิธีการ และขั้นตอนในการทำงานวิจัย ในการสร้างเครื่องจักรอัตโนมัติในการวางชิ้นงานบนสายพาน ดังภาพ 3.1



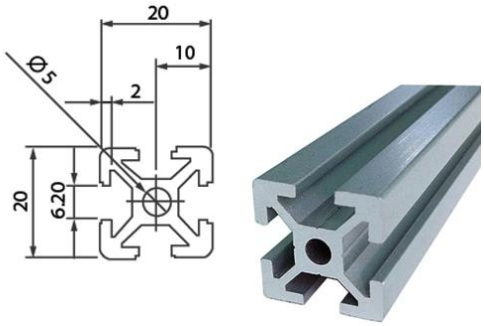
ภาพ 3.1 วิธีและขั้นตอนในการทำวิจัย

3.2 เครื่องมือที่ใช้ในงานวิจัย

3.2.1 โครงสร้างหลัก

◆ แท่งอลูมิเนียมโปรไฟล์ (Aluminum Profile)

สำหรับการขึ้นโครงสร้างหลักที่ใช้วางอุปกรณ์ต่างๆ นั้นจะใช้อลูมิเนียมโปรไฟล์ในการตัดเป็นขนาดต่างๆ แล้วนำมาต่อกัน ดังภาพ 3.2

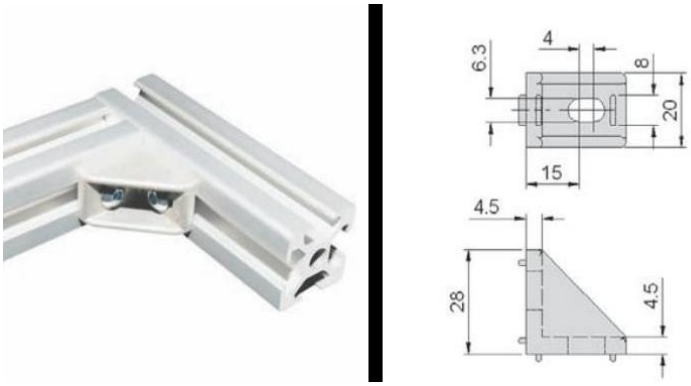


ภาพ 3.2 อลูมิเนียมโปรไฟล์

ที่มา : http:// www.buildingmaterial.co.th

◆ ตัวยึดฉาก (Bracket)

หลังจากที่ตัดอลูมิเนียมโปรไฟล์แล้วนำมาประกอบให้เป็นโครงสร้างต้องใช้ตัว Bracket ในการยึดตามมุมของโปรไฟล์ ดังภาพ 3.3



ภาพ 3.3 ตัวยึดฉาก (Bracket)

ที่มา : http:// www.buildingmaterial.co.th

◆ ทีนัท (T-Nut)

เหมาะกับงานที่ต้องรับโหลดหนัก เพราะตัวใหญ่พื้นที่ผิวสัมผัสสำหรับจับยึดโปรไฟล์มาก ต้องใส่จากปลายร่องโปรไฟล์เท่านั้น ทำให้ต้องวางแผนก่อนประกอบให้ดี หรืองานที่ต้องถอดย้ายบ่อยๆ ดังภาพ 3.4

ภาพ 3.4 ทีนัท (T-Nut)

ที่มา : http://www.dojogarden.com

◆ ฟรีนัท (Free Nut)

ใช้งานสะดวก เพราะใส่ระหว่างร่องได้ เหมาะกับงานที่ต้องการให้น้ำหนักโครงสร้างน้อยที่สุด ใช้กับงานต่อเติมได้ดี หรืองานที่มีการถอดประกอบเป็นประจำ ดังภาพ 3.5

ภาพ 3.5 ฟรีนัท (Free Nut)

ที่มา : http://www.dojogarden.com

◆ ล็อคนัท (Lock Nut)

มีบอลสปริงช่วยล็อกตำแหน่ง ทำให้ง่ายในการประกอบเสาโปรไฟล์แนวต้อง เพราะใช้กำหนดตำแหน่งได้ดี หรือกับงานที่ต้องมีการปรับตำแหน่งบ่อยๆ ดังภาพ 3.6

ภาพ 3.6 ล็อคนัท (Lock Nut)

ที่มา : http://www.dojogarden.com

◆ สกรูหัวจม (Hexagon Socket Head Cap Screws)

เป็นตัวยึดระหว่าง Bracket กับ Nut ดังภาพ 3.7

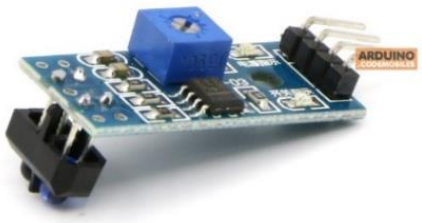
ภาพ 3.7 สกรูหัวจม (Hexagon Socket Head Cap Screws)

ที่มา : https://www.yahatafastenerthai.com

3.2.2 อุปกรณ์ในระบบไฟฟ้า

◆ เซ็นเซอร์ตรวจจับวัตถุสิ่งกีดขวางแบบอินฟาเรด (Infrared IR Obstacle Avoidance Sensor)

ใช้เป็นตัวตรวจจับวัตถุในสายพาน และส่งสัญญาณให้ระบบหุ่นยนต์ และราสเบอร์รี่ พาย (Raspberry Pi) ในการสั่งการ ดังภาพ 3.8



ภาพ 3.8 เซ็นเซอร์ตรวจจับวัตถุสิ่งกีดขวางแบบอินฟาเรด (Infrared IR Obstacle Avoidance Sensor)

ที่มา : https://th.element14.com

◆ โฟโตอิเล็กทริกเซนเซอร์ (Photoelectric sensor)

ใช้เป็นเซ็นเซอร์ตรวจจับวัตถุ เพื่อให้สายพานหยุด ดังภาพ 3.9

ภาพ 3.9 โฟโตอิเล็กทริกเซนเซอร์ (Photoelectric sensor)

ที่มา : https://www.fromfactory.net

◆ คัลเลอร์ดีเทคเตอร์ เซ็นเซอร์ (Color Detector Sensor)

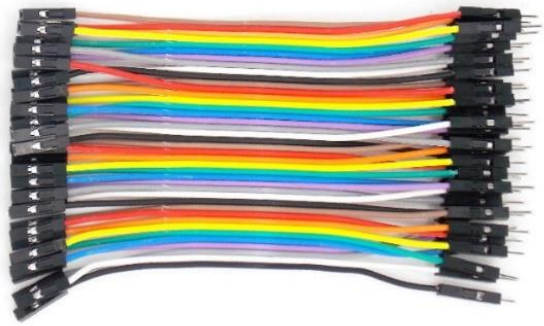
ใช้เป็นเซ็นเซอร์ตรวจจับวัตถุที่เป็นสี (ฝากล่อง) เพื่อแสดงว่ามีการปิดฝากล่อง ดังภาพ 3.10

ภาพ 3.10 คัลเลอร์ดีเทคเตอร์ เซ็นเซอร์ (Color Detector Sensor)

ที่มา : https://www.robotshop.com

◆ สายแพ หรือสายจัมเปอร์ (Jumper)

ใช้เป็นตัวส่งสัญญาณให้ระบบหุ่นยนต์ ระบบเซ็นเซอร์ และราสเบอร์รี่ พาย (Raspberry Pi) ดังภาพ 3.11



ภาพ 3.11 สายแพหรือสายจัมเปอร์ (Jumper)

ที่มา : https://www.mosfex.com

◆ มัลติมิเตอร์ (Multimeter)

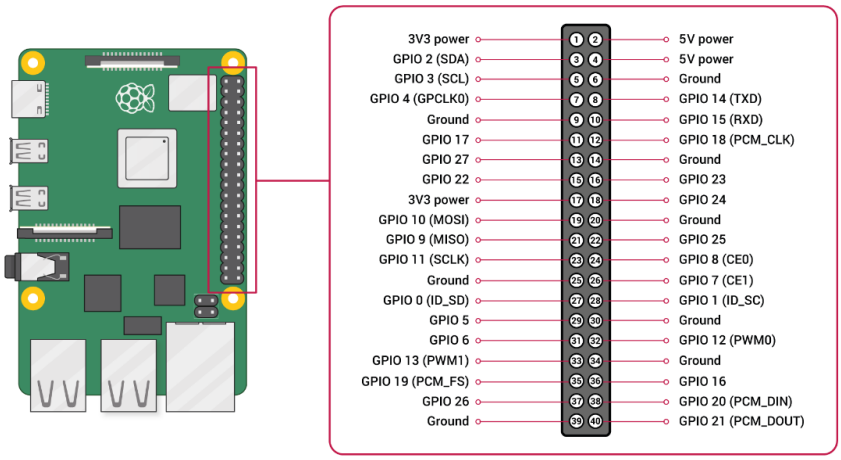
ในการต่อวงจรไฟฟ้านั้นต้องใช้ multimeter เพื่อวัดกระแสไฟฟ้า และตรวจการต่อเนื่องของสายไฟในวงจร ดังภาพ 3.12

ภาพ 3.12 มัลติมิเตอร์ (Multimeter)

ที่มา : https://skupaem-auto.ru

◆ พอร์ตเอนกประสงค์ราสเบอร์รี่ พาย (GPIO Raspberry Pi)

มี GPIO ทั้งหมด 40 Pin และที่ใช้สั่งควบคุมในการทำงานทั้งหมด 17 Pin ในการใช้งาน Pin ต่างๆ โดยสามารถดูการใช้งาน Pin ต่างๆ ได้ ดังภาพ 3.13

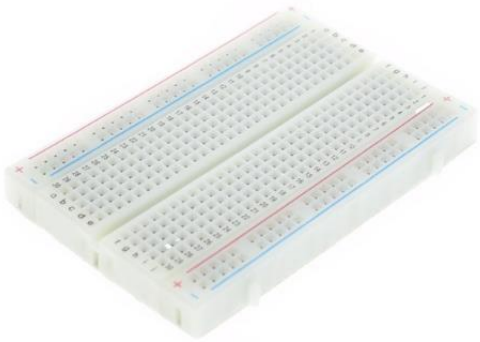


ภาพ 3.13 พอร์ตเอนกประสงค์ราสเบอร์รี่ พาย (GPIO Raspberry Pi)

ที่มา : https://www.raspberrypi.org

◆ เบรดบอร์ด (Breadboard)

เป็นบอร์ดที่ใช้ทดลองวงจรอิเล็กทรอนิกส์ ลักษณะเป็นแผ่นพลาสติกหนาสีขาว บนแผ่นมีรูเรียงกันจำนวนมาก ภายในรูมีตัวนำไฟฟ้าซึ่งเชื่อมต่อกันในรูปแบบที่มีการกำหนดไว้ เวลาทดลองก็เสียบขาของอุปกรณ์อิเล็กทรอนิกส์ลงไปให้ตัวนำภายในเชื่อมวงจรถึงกัน และอาจใช้สายไฟเสียบลงรูเพื่อเชื่อมวงจรไฟฟ้า ดังภาพ 3.14



ภาพ 3.14 เบรดบอร์ด (Breadboard)

ที่มา : http://www.robotwinner.com

3.2.3 อุปกรณ์ที่ใช้ใน Robotic System

◆ Dobot Magician

ใช้ในกระบวนการหยิบว่างกล่องบนสายพาน หยิบฝากล่องมาปิด และเป็นตัวควบคุมความเร็วสายพาน แสดงดังภาพ 3.15

ภาพ 3.15 Dobot Magician

ที่มา : https://www.dobot.cc

◆ สายพาน (Conveyer)

ใช้เป็นตัวเคลื่อนกล่องไปยังตำแหน่งต่างๆ ขนาดของ Conveyer มีหน้ากว้าง 10 เซนติเมตร และความยาวของสายพานนั้นอยู่ที่ 670 มิลลิเมตร ดังภาพ 3.16

ภาพ 3.16 สายพาน (Conveyer)

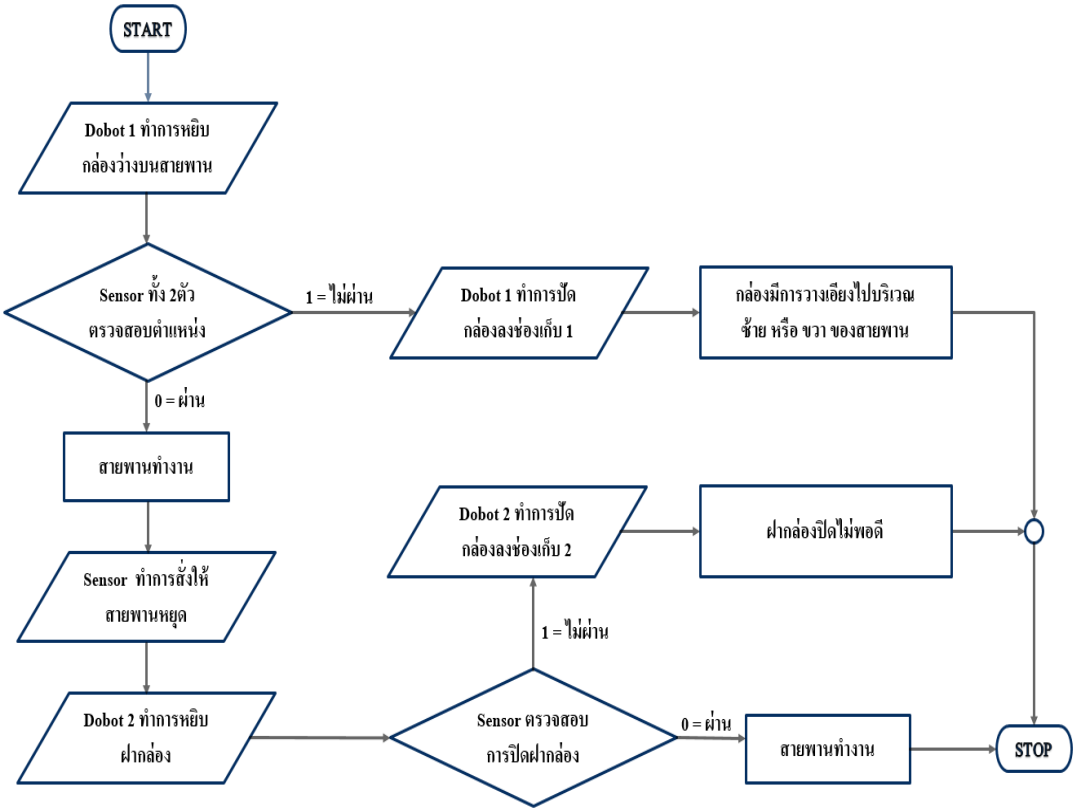
ที่มา : https://www.dobot.cc/

3.3 การศึกษาปัญหา

ปัจจุบันโรงงานอุตสาหกรรมได้เล็งเห็นถึงความสำคัญของระบบอัตโนมัติซึ่งระบบอัตโนมัตินั้นสามารถเพิ่มกำลังการผลิตให้กับบริษัท ซึ่งแขนกลเป็นส่วนหนึ่งของระบบอัตโนมัติที่สามารถควบคุมและช่วยลดความเสียหายจากการทำงานโดยแรงงานมนุษย์อีกทั้งการทำงานที่สะดวก และง่าย สามารถผลิตสินค้าที่ได้มาตรฐาน สามารถทำงานได้ตามเวลาที่กำหนด สามารถควบคุมกระบวนการทำงานได้ทุกขั้นตอนของการผลิต สามารถตรวจสอบความผิดพลาดของกระบวนการทำงานได้ทุกขั้นตอน อย่างไรก็ตามแขนกลอาจหยิบ หรือวางชิ้นส่วนของผลิตภัณฑ์ผิดตำแหน่งตำแหน่ง ซึ่งทำให้เกิดของเสียอันเนื่องมาจากปัจจัยต่างๆ เช่น การติดตั้งแขนกลที่ไม่ได้มาตรฐานของผู้ติดตั้ง ระยะเวลาในการใช้งานมากเกินกว่าที่แขนกลจะรับได้ทำให้เกิดความร้อนสูงภายในวงจรอาจทำให้เกิดความผิดพลาดของระบบได้ เพราะฉะนั้นจึงได้มีระบบเซนเซอร์ในการตรวจสอบ เพื่อบ่งบอกว่ามีความผิดปกติของตำแหน่งชิ้นงานมาแล้ว จากนั้นระบบจะทำการแก้ไข และเก็บข้อมูลผ่านระบบ Internet of Things (IoT) เพื่อจะแสดงผลข้อมูลไปให้พนักงานที่เกี่ยวข้องในบริษัทได้รับรู้ เพื่อที่จะได้ดำเดินการแก้ไขในขั้นต่อไป

3.4 การออกแบบเครื่องจักร

3.4.1 การออกแบบกระบวนการด้วยแผนผังการทำงาน

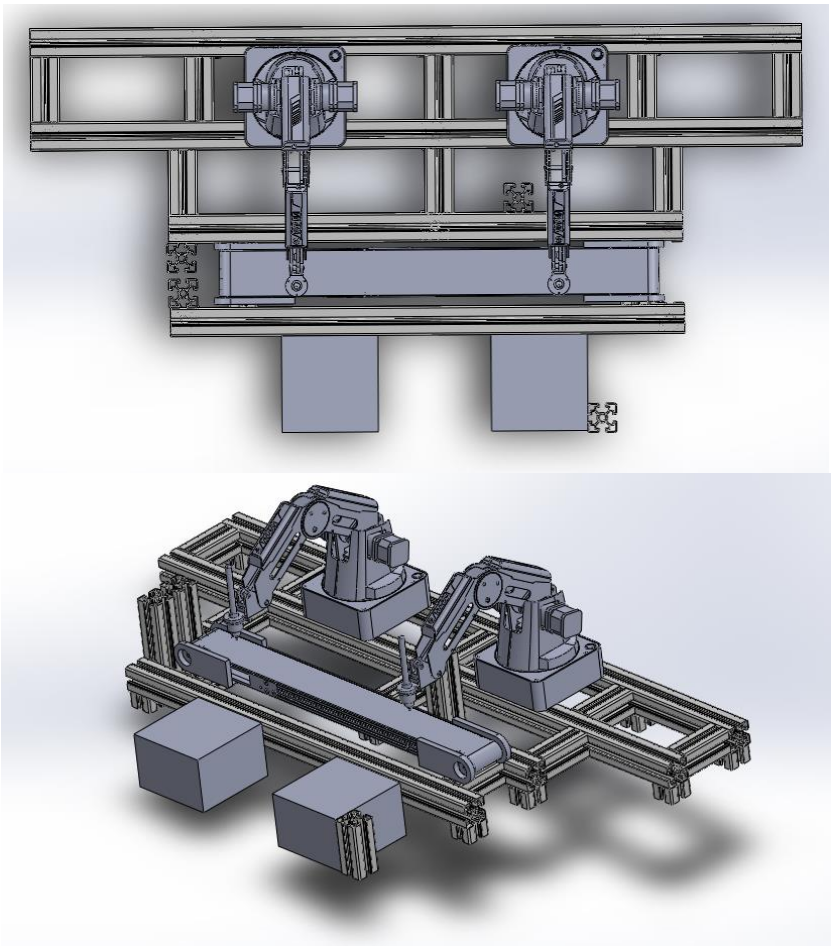


ภาพ 3.17 แผนผังการทำงานของเครื่องจักร

จากภาพ 3.17 คือกระบวนการทำงาน โดยเริ่มจาก Dobot ตัวที่ 1 ทำการหยิบกล่องว่างบนสายพานแล้วจะมีเซ็นเซอร์จำนวน 2 ตัว มาตรวจสอบตำแหน่งว่ากล่องได้วางในตำแหน่งที่ถูกต้องหรือไม่ ถ้าไม่ Dobot ตัวที่ 1 จะทำการปัดกล่องลงช่องเก็บ 1 ถ้าผ่านสายพานก็จะทำงานต่อไป จนกล่องเคลื่อนที่มาพบเซ็นเซอร์ ซึ่งจะทำให้สายพานหยุดจากนั้น Dobot ตัวที่ 2 ทำการหยิบฝากล่องมาปิด แล้วจะมีเซ็นเซอร์ตรวจสอบสี ซึ่งฝากล่องเราจะทาสีไว้ เพื่อจะได้ทราบว่าฝากล่องถูกปิดสนิท ถ้าเซ็นเซอร์ตรวจสอบสีตรวจไม่พบ Dobot ตัวที่ 2 จะทำการปัดกล่องลงช่องเก็บที่ 2 แต่ถ้าเซ็นเซอร์ตรวจสอบสีตรวจพบ สานพานก็จะทำงานต่อ แล้วส่งกล่องที่ทำการปิดฝาไปยังช่องเก็บ โดนทุกกระบวนการจะมีเซ็นเซอร์ทำการเก็บข้อมูลแล้วนำมาแสดงผ่านคอมพิวเตอร์ ว่ามีการผิดพลาดตรงส่วนไหนของระบบ แล้วทำผลมาวิเคราะห์แล้วทำการหาสาเหตุแล้วทำการแก้ไขระบบ

3.4.2 การออกแบบเครื่องจักรด้วยโปรแกรมโซลิดเวิร์ค (Solidwork)

เริ่มทำการสร้างเครื่องจักรนั้นจำเป็นต้องมีการออกแบบเครื่องจักรก่อนเพื่อที่จะสามารถกำหนดลักษณะให้เป็นตามข้อจำกัดต่างๆ และเพื่อไม่ให้เกิดข้อผิดพลาดในการประกอบ การสั่งซื้ออุปกรณ์ หลักการในการออกแบบเครื่องจักรนั้นต้องมีการวัดขนาดของอุปกรณ์ที่ต้องใช้ทั้งหมด เขียนเป็นภาพ 3D และจากนั้นนำมาประกอบกันเป็นเครื่องจักร ประโยชน์ของการออกแบบด้วยโปรแกรมนั้น เราจะสามารถนำไปสั่งซื้ออุปกรณ์โดยคิดเป็น BOQ ได้ และยังสามารถปรับแต่งให้เหมาะสม โดยเครื่องจักรที่ออกแบบนั้นจะแสดงดังภาพ 3.18



ภาพ 3.18 การออกแบบเครื่องจักรด้วยโปรแกรมคอมพิวเตอร์

3.5 การซื้อวัสดุ

ได้เลือกซื้อในบริษัท นอร์ธเทิร์น อาร์ต โปรดักท์ จำกัด เป็นบริษัทแยกส่วนประกอบของเครื่องจักร ซึ่งจะมีราคาที่ถูก สิ่งที่ซื้อมาคือ Aluminium profile Bracket T-Nut Free Nut Lock Nut และHexagon Socket Head Cap Screws ดังภาพ 3.19 และภาพ 3.20

ภาพ 3.19 Aluminium profile Bracket

ภาพ 3.20 T-Nut Free Nut Lock Nut และHexagon Socket Head Cap Screws

3.6 ขั้นตอนการทำและประกอบชิ้นส่วนของกระบวนการ

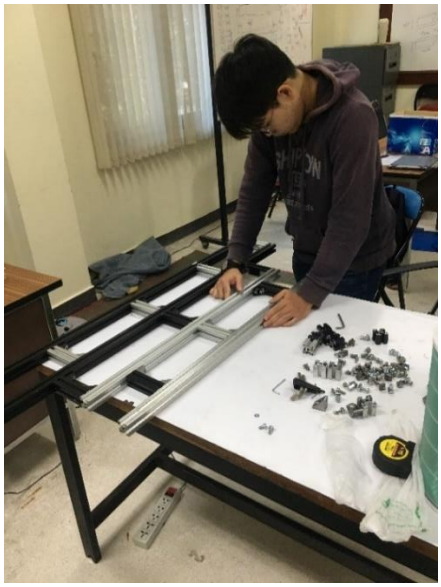
3.6.1 ตัดอลูมิเนียมโปรไฟล์ ดังภาพ 3.21 และภาพ 3.22

- นำอลูมิเนียมโปรไฟล์ตัดให้มีความยาว 120 เซนติเมตร จำนวน 2 เส้น
- นำอลูมิเนียมโปรไฟล์ตัดให้มีความยาว 80 เซนติเมตร จำนวน 2 เส้น
- นำอลูมิเนียมโปรไฟล์ตัดให้มีความยาว 10 เซนติเมตร จำนวน 10 เส้น
- นำอลูมิเนียมโปรไฟล์ตัดให้มีความยาว 3 เซนติเมตร จำนวน 16 ชิ้น

ภาพ 3.21 ตัดอลูมิเนียมโปรไฟล์ด้วยเครื่องตัดโลหะ

ภาพ 3.22 ตัดอลูมิเนียมโปรไฟล์

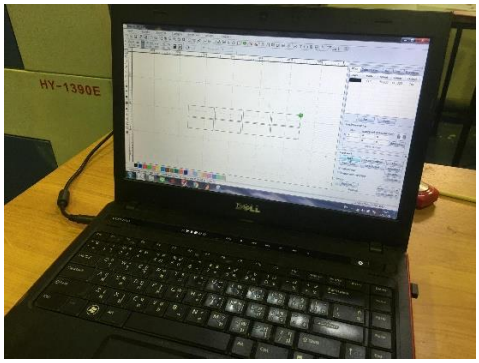
3.6.2 ทำการประกอบเข้าด้วยกัน ดังภาพ 3.23 และภาพ3.24



ภาพ 3.23 การประกอบอลูมิเนียมโปรไฟล์

ภาพ 3.24 ประกอบอลูมิเนียมโปรไฟล์เมื่อเสร็จ

3.6.3 ทำการตัดแผ่นอะคริลิคหนา 3 มิลลิเมตร ดังภาพ 3.25 3.26 และ3.27



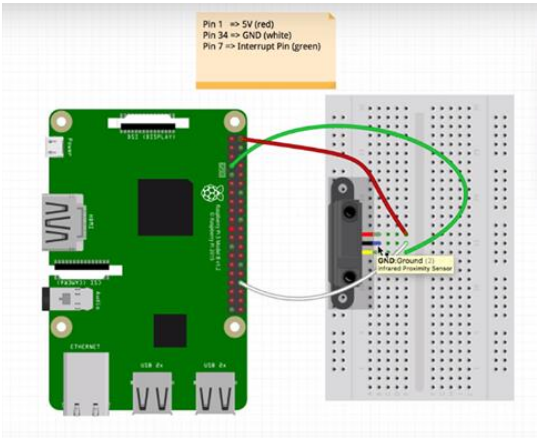
ภาพ 3.25 ออกแบบรูปทรงที่จะตัด

ภาพ 3.26 ทำการตัดแผ่นอะคริลิค

ภาพ 3.27 ประกอบแผ่นอะคริลิคเข้ากับอลูมิเนียมโปรไฟล์

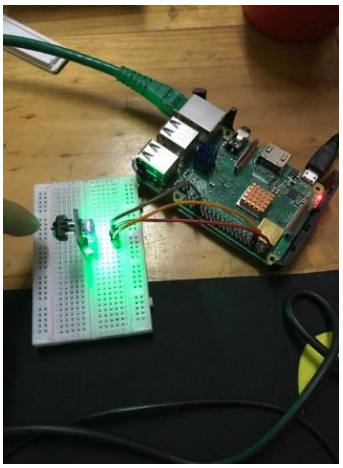
3.7 ขั้นตอนการศึกษา

3.7.1 การจำลองการต่อวงจรเซนเซอร์อินฟาเรด (Infrared Sensor) กับราสเบอร์รี่ พาย (Raspberry Pi) โดยใช้โปรแกรม Fritzing ดังภาพ 3.28



ภาพ 3.28 การจำลองการต่อวงจรเซนเซอร์อินฟาเรด (Infrared Sensor) กับราสเบอร์รี่ พาย (Raspberry Pi) โดยใช้โปรแกรม Fritzing

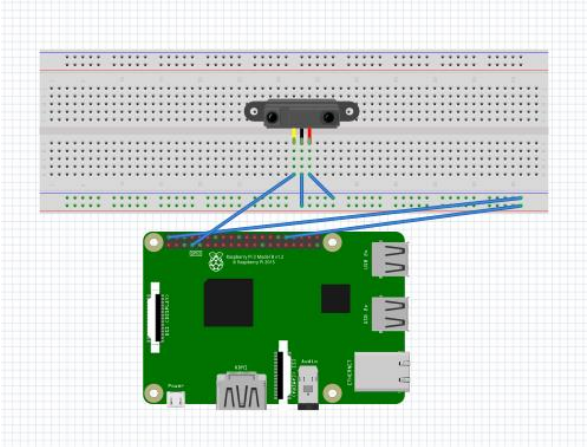
3.7.2 การต่อวงจรเซนเซอร์อินฟาเรด (Infrared Sensor) กับราสเบอร์รี่ พาย (Raspberry Pi) ดังภาพ 3.29



ภาพ 3.29 การต่อวงจรเซนเซอร์อินฟาเรด (Infrared Sensor) กับราสเบอร์รี่ พาย (Raspberry Pi)

3.8 การออกแบบระบบการต่อเซนเซอร์ (Sensor) กับราสเบอร์รี่ พาย (Raspberry Pi)

3.8.1 อุปกรณ์ในระบบวงจร sensor ทั้งหมดจะประกอบไปด้วยเซ็นเซอร์อินฟาเรด (Infrared sensor) 2 ตัว สายไฟจัมป์เปอร์ 9 เส้น เบรดบอร์ด 1 ตัว และราสเบอร์รี่ พาย (Raspberry Pi) 1 ตัว การทำงานจะเริ่มจาก เมื่อกดปุ่มเริ่ม (Run) ในโปรแกรม Qt ในราสเบอร์รี่ พาย (Raspberry Pi) จะสั่งให้ตัวเซ็นเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) ทำการส่งสัญญาณ เป็นแสงอินฟราเรดออกไปกระทบกับกล่องที่ตรวจพบในระยะ จะทำการสะท้อนกลับมายังตัวหลอดโฟโต้ไดโอดที่ทำหน้าที่รับแสงอินฟราเรด จากนั้นจะทำการส่งค่าไปที่โปรแกรม Qt บนราสเบอร์รี่ พาย (Raspberry Pi) ดังภาพ 3.30

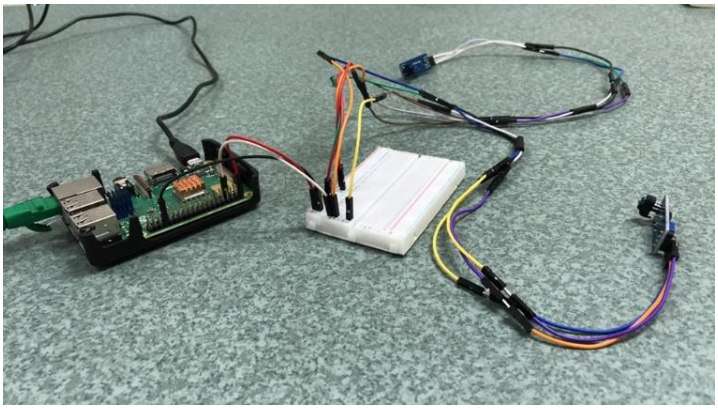


ภาพ 3.30 การจำลองการต่อวงจรเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) กับราสเบอร์รี่ พาย (Raspberry Pi) โดยใช้โปรแกรม Fritzing

3.8.2 ทำการเชื่อมต่อวงจร เซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) กับ ราสเบอร์รี่ พาย(Raspberry Pi)

นำสายจัมเปอร์ เสียบ GPIO ราสเบอร์รี่ พาย เข้ากับ ขาที่ 2 ขา ที่ 34 และขาที่ 7 ลงบนเบรดบอร์ด จากนั้นนำสายจัมเปอร์ต่อที่ขาของเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) ทั้ง 3 ขา ซึ่งจะมีไฟแสดงสถานะอยู่ 2 ดวง จากนั้นนำมาต่อลงบนเบรดบอร์ด โดยขาที่ 2 จะเป็นขาที่จ่ายแรงดันไฟฟ้า 5 โวลท์ ที่ส่งไปยังตัวเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) โดยจะมีการต่อขาที่ 34 คือกาวด์ และขา 7 จะเป็น GPIO4 ที่รับค่า input จาก

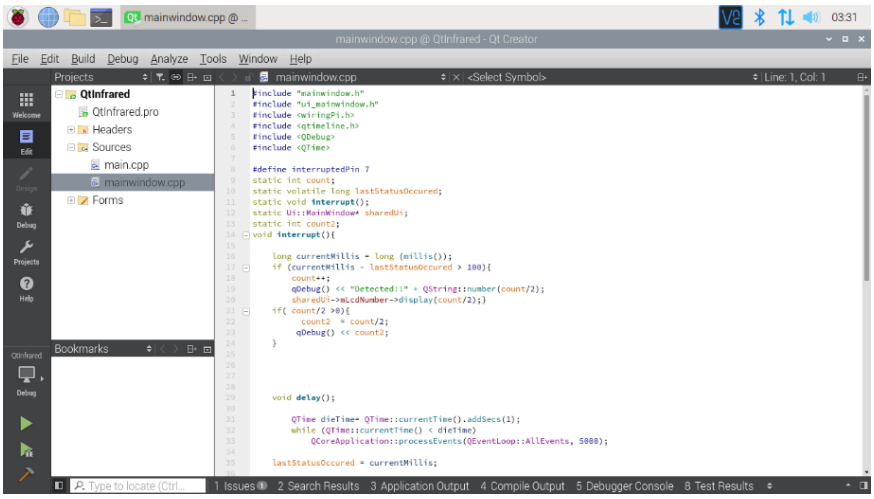
เซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) ทำให้ไฟแสดงสถานะติดขึ้น 1 ดวง เมื่อตรวจพบเจอกล่องไฟเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) ไฟแสดงสถานะจะติดขึ้นอีก 1 ดวง รวมเป็น 2 ดวง เนื่องจากสัญญาณถูกส่งไปยังขาเข้า input GPIO4 เพื่อแสดงผลไปยังตัวโปรแกรม ดังภาพ 3.31



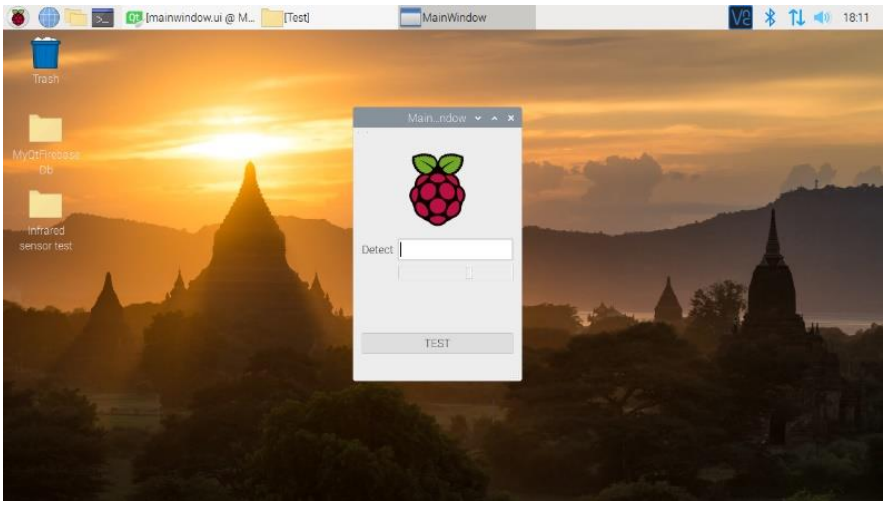
ภาพ 3.31 แสดงภาพการเชื่อมต่อวงจรเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) กับราสเบอร์รี่ พาย (Raspberry Pi)

3.8.3 การโปรแกรม sensor เพื่อตรวจจับกล่องที่ถูกปฏิเสธ (Reject) แล้วส่งข้อมูลแบบตอบสนองทันที (Real Time) ด้วยภาษา C++

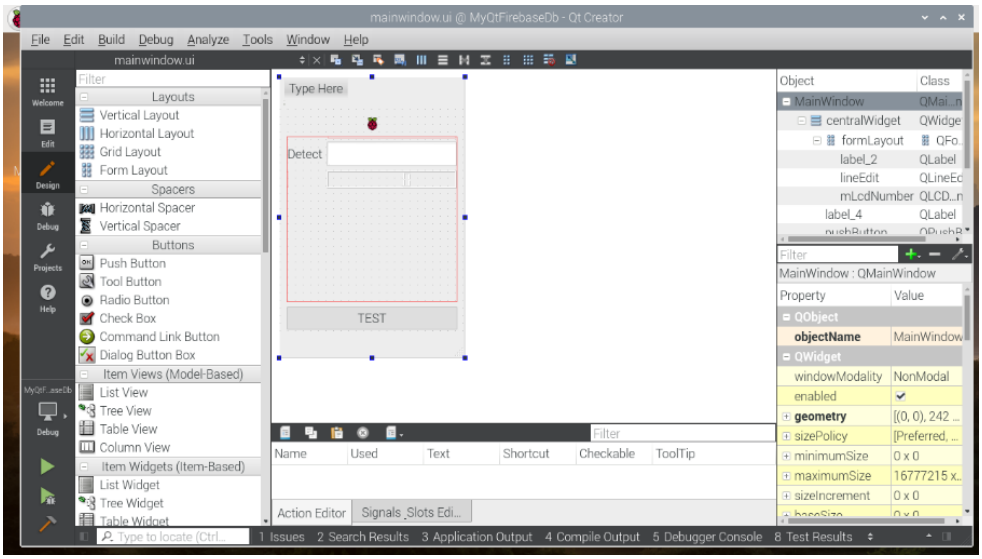
ระบบหุ่นยนต์นั้นจะมีลักษณะการทำงานแบบหยิบกล่อง และปิดฝากล่องหากเกิดการวางกล่องผิดตำแหน่ง หรือการปิดฝากล่องที่ไม่สมบูรณ์เมื่อโฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) และเซนเซอร์สี (Color Sensor) ตรวจจับได้จะถูกทำการปฏิเสธ (Reject) ไปยังกล่องที่อยู่ข้างล่างซึ่งจะมีเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor TCRT5000) ในการตรวจจับซึ่งจะถูกส่งข้อมูลไปยังหน้า ดังภาพ 3.33 GUI คือการติดต่อกับผู้ใช้โดยใช้ภาพสัญลักษณ์ เป็นการออกแบบส่วนของโปรแกรมคอมพิวเตอร์ให้มีการโต้ตอบกับผู้ใช้งาน โดยใช้รูปภาพและสัญลักษณ์เข้ามา แทนที่ผู้ใช้จะพิมพ์คำสั่งต่างๆ ในการทำงาน ดังภาพ 3.34 และยังส่งข้อมูลไปยังฐานข้อมูล (Data Base) เพื่อแสดงผล ดังภาพ 3.35 ไฟร์เบส กูเกิล (Firebase Google) แบบตอบสนองทันที (Real Time) โดยการเขียนภาษา C++ ในโปรแกรม Qt Creator ดังภาพ 3.32



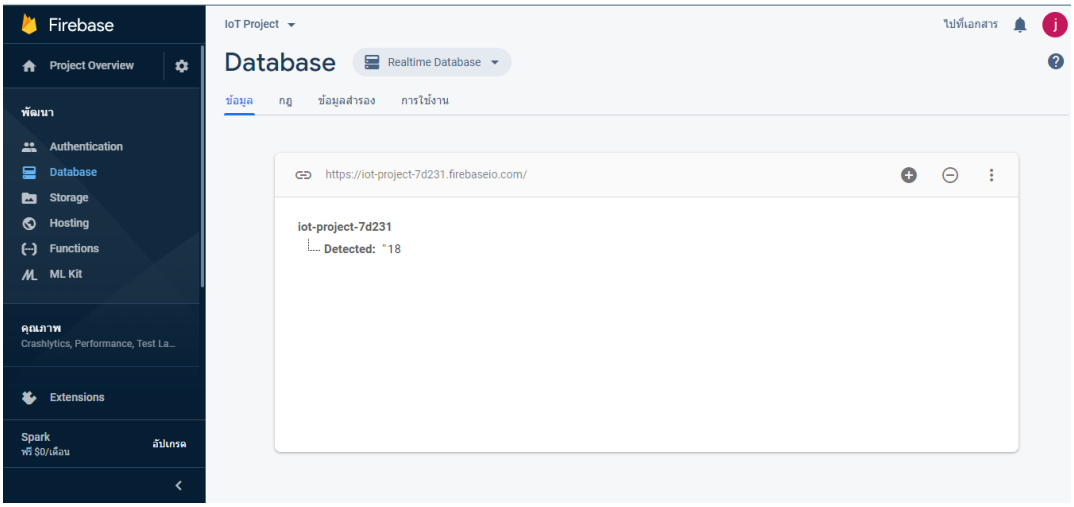
ภาพ 3.32 การโปรแกรมเซนเซอร์ (Sensor) ด้วยโปรแกรม Qt Creator



ภาพ 3.33 Graphical User Interface (GUI)



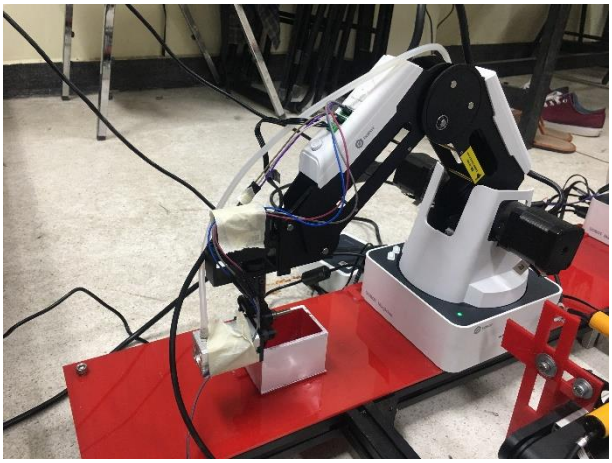
ภาพ 3.34 แสดงการออกแบบ Graphical User Interface



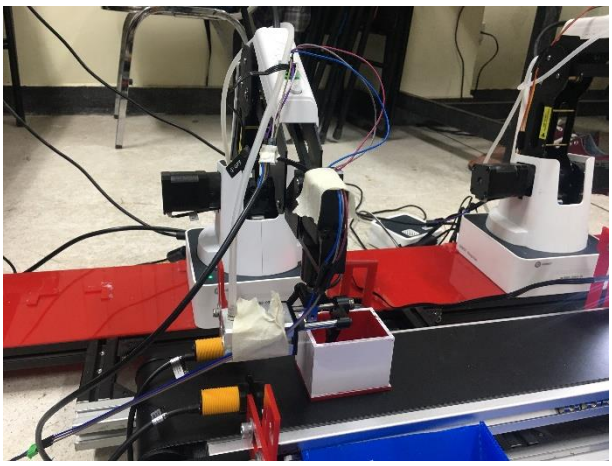
ภาพ 3.35 ไฟร์เบส กูเกิล (Firebase Google)

3.9 การทำงานของ DOBOT

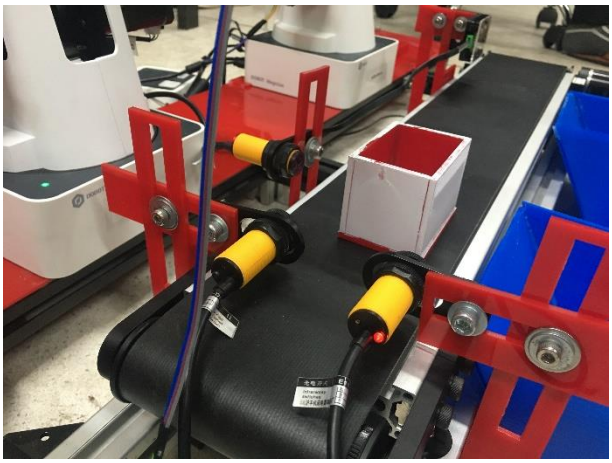
3.8.1 DOBOT ตัวที่ 1 ทำการหยิบกล่องจากตำแหน่งที่ 1 ดังภาพ 3.36 โดยการหยิบกล่องนำไปวางบนสายพานมีโฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) จำนวน 2 ตัว ทีการตรวจจับว่ากล่องวางถูกตำแหน่ง หรือไม่ ดังภาพ 3.37 ถ้าโฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) ตรวจพบ DOBOT ตัวที่ 1 ดังภาพ 3.38 ทำการดันกล่องออกจากสายพาน ดังภาพ 3.39 โค้ดการทำงานของ DOBOT ตัวที่ 1 ดังภาพ 3.40



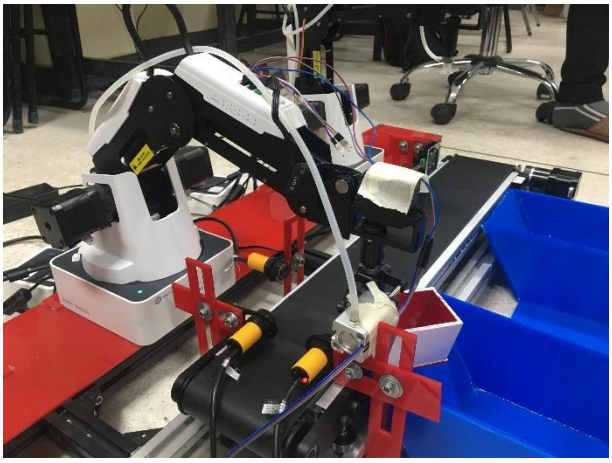
ภาพ 3.36 DOBOT ตัวที่ 1 ทำการหยิบกล่องจากตำแหน่งที่ 1



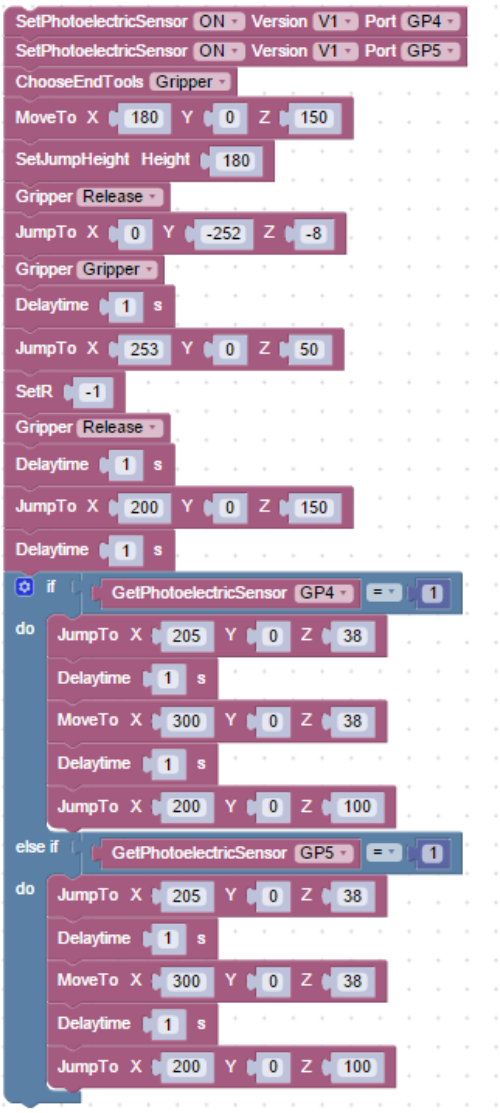
ภาพ 3.37 หยิบกล่องนำไปวางบนสายพาน



ภาพ 3.38 โฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) ตรวจพบ

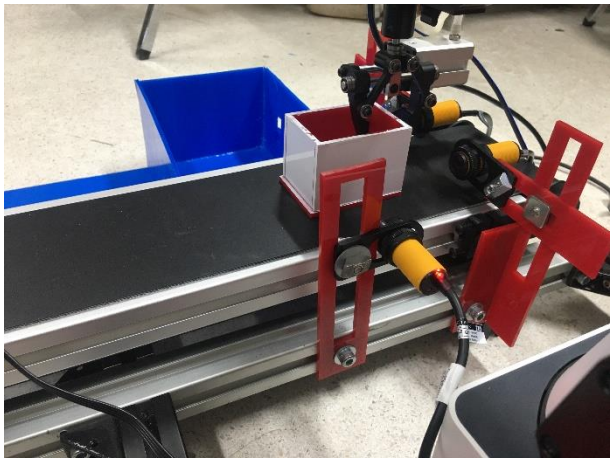


ภาพ 3.39 ทำการดันกล่องออกจากสายพาน

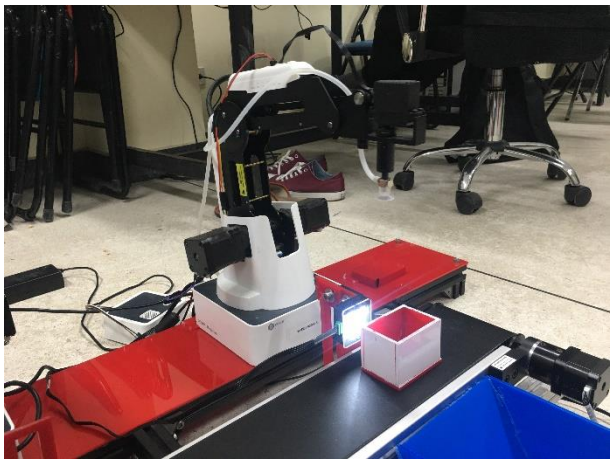


ภาพ 3.40 โค้ด DOBOT ตัวที่ 1

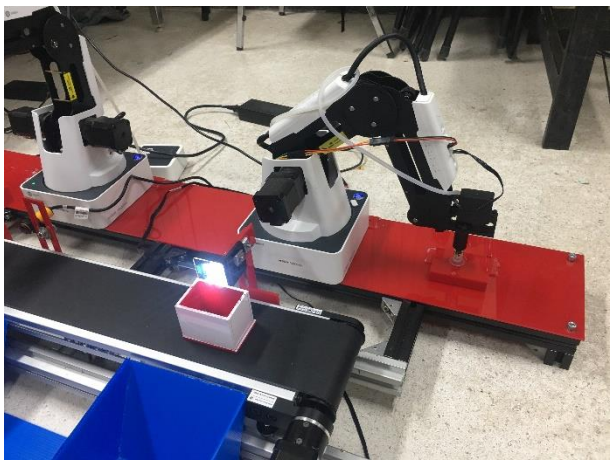
3.8.2 DOBOT ตัวที่ 2 โฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) ทำการตรวจพบวัตถุ ดังภาพ 3.41 สายพานเคลื่อนกล่องมายังตำแหน่งที่ 2 ดังภาพ 3.42 DOBOT ตัวที่ 2 ทำการหยิบฝากล่องมาปิด ดังภาพ 3.43 มีคัลเลอร์ดีเทคเตอร์เซ็นเซอร์ (Color Detector Sensor) ทำการตรวจสอบว่ากล่องถูกปิดฝา ดังภาพ 3.44 โดยตัวกล่องกับฝาจะมีสีที่แตกต่างกัน ถ้าคัลเลอร์ดีเทคเตอร์เซ็นเซอร์ (Color Detector Sensor) ตรวจพบ สีแดงแสดงว่าฝากล่องถูกปิด สายพานจะเคลื่อนที่นำกล่องไปเก็บ ดังภาพ 3.45 แต่ถ้าตรวจไม่พบ DOBOT ตัวที่ 2 ทำการดันกล่องออกจากสายพาน ดังภาพ 3.46 โค้ดการทำงานของ DOBOT ตัวที่ 2 ดังภาพ 3.47



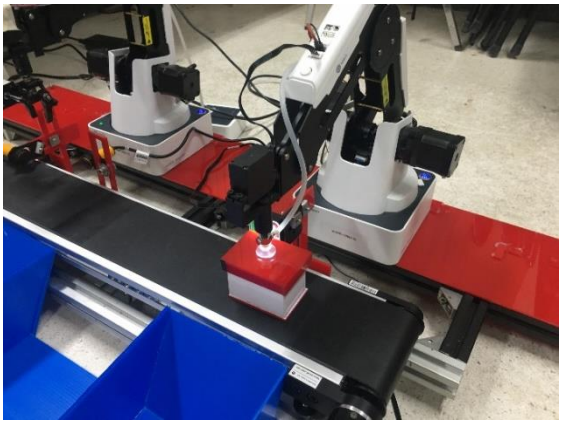
ภาพ 3.41 โฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) ทำการตรวจพบวัตถุ



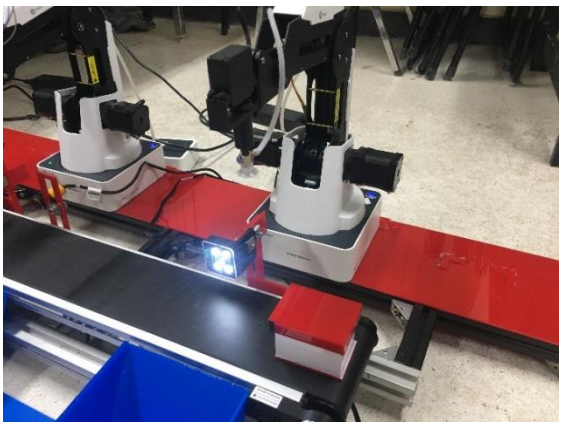
ภาพ 3.42 สายพานเคลื่อนกล่องมายังตำแหน่งที่ 2



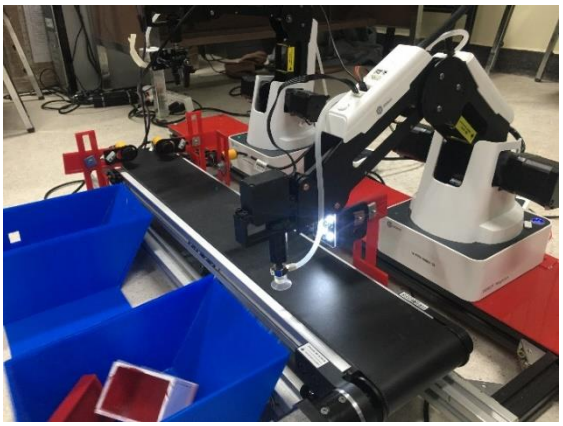
ภาพ 3.43 ทำการหยิบฝากล่องมาปิด



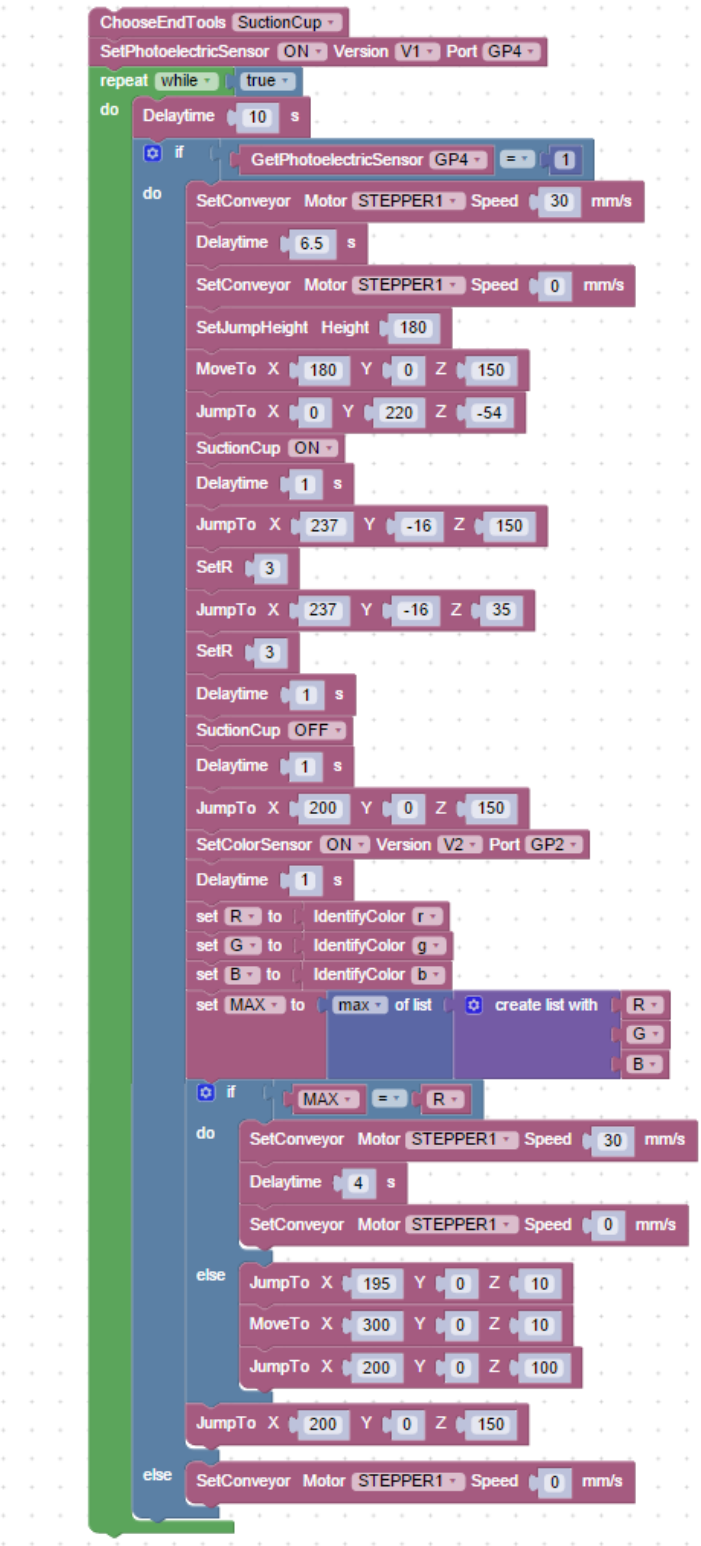
ภาพ 3.44 คัลเลอร์ดีเทคเตอร์เซ็นเซอร์ (Color Detector Sensor) ทำการตรวจสอบว่ากล่องถูกปิดฝา



ภาพ 3.45 คัลเลอร์ดีเทคเตอร์เซ็นเซอร์ (Color Detector Sensor) ตรวจพบสีแดง



ภาพ 3.46 ตรวจไม่พบ DOBOT ตัวที่ 2 ทำการดันกล่องออกจากสายพาน



ภาพ 3.47 โค้ด DOBOT ตัวที่ 2

3.10 การทำงานของราสเบอร์รี่ พาย (Raspberry Pi)

ทำการนับจำนวนของเสียจาก DOBOTทั้ง 2 ตัว โดยตำแหน่งว่างที่ 1 ก็มี เซนเซอร์อินฟาเรด (Infrared Reflectance Sensor) 1 ตัว คอยนับจำนวนของเสีย ดังภาพ 3.48

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "firebase.h"
#include <QDebug>
#include <QJsonDocument>
#include <QJsonValue>
#include <QJsonArray>
#include <QJsonObject>
#include <wiringPi.h>
#include <qtimer.h>
#include <qtimeline.h>
#include <QTime>

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string>
#include <sstream>
#define interruptedPin 7
static int count;
static volatile long lastStatusOccured;
static int countmix;
static Ui::MainWindow* sharedUi;


MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow ){
    {
        count = 0;
        sharedUi = ui;

        ui->setupUi(this);
        wiringPiSetup();
        wiringPiISR(interruptedPin,INT_EDGE_FALLING, &interrupt);

    }

    ui->setupUi(this);
    connect(ui->pushButton, SIGNAL(clicked()), this, SLOT(on_pushButton_clicked()));
    QString url = "https://iot-project-7d231.firebaseio.com/";
        Firebase *myFirebase = new Firebase(url);
        myFirebase->setToken("5AFAtb8Oe9YfBH3KvLnFsuXDLXxifs0QB9YafBQu");
        connect(myFirebase, SIGNAL(eventResponseReady(QString)), this, SLOT(onResponseReady(QString)));
        connect(myFirebase, SIGNAL(eventDataChanged(DataSnapshot*)), this, SLOT(onDataChanged(DataSnapshot*)));

        myFirebase->listenEvents();
        myFirebase->getValue();

        wiringPiSetup();
        QTimer *timer = new QTimer(this);
        connect(timer, SIGNAL(timeout()), this, SLOT(interrupt()));
       this->updateFirebase("Detected", sharedUi->lineEdit->text());


}

//sensor//
void MainWindow::interrupt(){


long currentMillis = long (millis());


if (currentMillis - lastStatusOccured > 100){
    count++;
    qDebug() << "Detected=>>" + QString::number(count/2);
    sharedUi->mLcdNumber->display(count/2);
}
if( count/2 >0){
```

ภาพ 3.48 โค้ดราสเบอร์รี่ พาย (Raspberry Pi)

บทที่ 4

ผลการดำเนินการ

4.1 การทดสอบการทำงานของเครื่องจักร

จุดประสงค์ของการทดสอบเครื่องจักรเพื่อศึกษาการทำงานของ Dobot Magician ว่ามีความผิดปกติในขณะทำงาน หรือไม่ และเซนเซอร์สามารถนับจำนวนของเสียที่เกิดจากการทำงานผิดพลาดของ Dobot Magician หรือไม่ โดยนำเซนเซอร์เข้ามาทำการตรวจสอบในแต่ละกระบวนการทำงาน ในการในการทดสอบได้สร้างกระบวนการทำงานขึ้นมา 1 กระบวนการ เพื่อนำมาใช้ในการทดสอบ จึงได้สร้างกล่องที่มีขนาด 5 X 7 X 5 เซนติเมตร พร้อมฝากล่อง ขั้นตอนในการทดสอบจะเริ่มจาก DOBOT ตัวที่ 1 หยิบกล่องแบบไม่มีฝาปิดวางบนสายพาน เมื่อจบกระบวนการทำงานกล่องจะถูกปิดฝา แล้วทำการนับความผิดพลาดที่เกิดขึ้นในแต่ละกระบวนการ จะอธิบายโดยละเอียดในหัวข้อ 4.2

4.2 การดำเนินการทดสอบเครื่องจักร

4.2.1 DOBOT ตัวที่ 1 ทำการหยิบกล่องฝาเปิดจากตำแหน่งที่กำหนดไว้ นำมาวางบนสายพาน ก็จะมีโฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) 2 ตัว ตรวจจับว่ามีการวางอยู่ตรงกลางสายพาน

4.2.2 DOBOT ตัวที่ 2 ทำการหยิบฝากล่องในตำแหน่งที่กำหนดไว้ นำมาปิดฝากล่อง ตรวจจับว่ามีการปิดฝากล่องพอดี

4.2.3 เซนเซอร์อินฟาเรด (Infrared Reflectance Sensor) ทั้ง 2 ตัว นับจำนวนความผิดพลาดของ DOBOT แล้วความผิดพลาดจะถูกส่งข้อมูล เพื่อนำไปแสดงบนในเว็ป

4.3 การดำเนินการทดสอบ

4.3.1 การตรวจสอบครั้งแรก

ผลการทดสอบ DOBOT ตัวที่ 1 การทำงานได้ปกติ แต่ DOBOT ตัวที่ 2 การทำงานค่อนข้างมีความผิดพลาดเยอะพอสมควร ดังตาราง 4.1

ตาราง 4.1 ตรวจสอบการทำงานของ DOBOT

จำนวนครั้ง	DOBOT ตัวที่ 1		DOBOT ตัวที่ 2		สาเหตุของความผิดพลาด
	ทำงานปกติ	ทำงานผิดปกติ	ทำงานปกติ	ทำงานผิดปกติ	
1	/			/	1. DOBOT มีความคลาดเคลื่อน 2. ผู้ปฏิบัติวางฝาผิดตำแหน่งทำให้ DOBOT หยิบผิดตำแหน่ง 3. ตำแหน่ง Color Detector Sensor อยู่ต่ำเกินไป 4. DOBOT มีการใช้งานอย่างต่อทำให้ตัว DOBOT เกิดความร้อนจึงทำให้เกิดความคลาดเคลื่อน
2	/		/		
3	/			/	
4	/		/		
5	/		/		
6	/		/		
7	/		/		
8	/			/	
9	/			/	
10	/			/	
11	/		/		
12	/		/		
13	/			/	
14	/			/	
15	/		/		
16	/		/		
17	/		/		
18	/			/	
19	/			/	
20	/		/		
รวม	20	0	11	9	

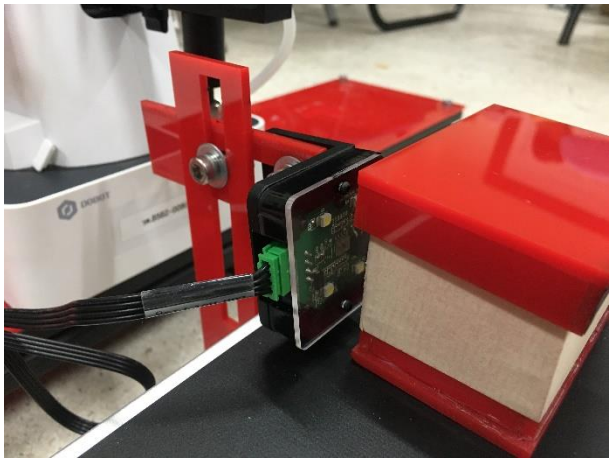
เซนเซอร์อินฟาเรด (Infrared Reflectance Sensor) ทั้ง 2 ตัว ทำงานได้เป็นส่วนใหญ่ปกติแต่ก็ยังมีบ่างส่วนที่ทำงานผิดพลาด ดังตาราง 4.2

ตาราง 4.2 ตรวจสอบการทำงานของเซนเซอร์นับจำนวนของเสีย

จำนวนครั้ง	การตรวจเซนเซอร์แบบ monitor		การตรวจเซนเซอร์แบบ real time		สาเหตุของความผิดพลาด
	ทำงาน	ไม่ทำงาน	ส่งข้อมูล	ไม่ส่งข้อมูล	
1		/		/	1. ตำแหน่งเซนเซอร์อยู่ห่างจากชิ้นงานมากเกินไป 2. ตัวกล่องมีสีทึบมากเกินไปทำให้เซนเซอร์ไม่สามารถตรวจพบ 3. อินเตอร์เน็ตเกิดความขัดข้อง
2	-	-	-	-	
3	/		/		
4	-	-	-	-	
5	-	-	-	-	
6	-	-	-	-	
7	-	-	-	-	
8	/			/	
9	/		/		
10	/		/		
11	-	-	-	-	
12	-	-	-	-	
13		/		/	
14	/		/		
15	-	-	-	-	
16	-	-	-	-	
17	-	-	-	-	
18	/		/		
19	/		/		
20	-	-	-	-	
รวม	7	2	6	3	

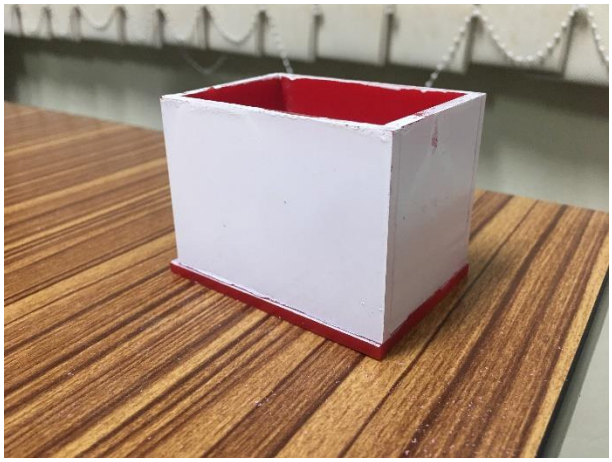
4.3.2 ทำการปรับปรุงเพื่อลดความผิดพลาดของการทำงาน

- การปรับตำแหน่งเซนเซอร์ให้เหมาะสม ดังภาพ 4.1



ภาพ 4.1 ปรับตำแหน่งเซนเซอร์

- พ้นสีกล่องให้เป็นสีขาว ดังภาพ 4.2



ภาพ 4.2 พ้นสีกล่องให้เป็นสีขาว

4.3.3 การตรวจสอบหลังการปรับปรุง

ผลการทดสอบ DOBOT ตัวที่ 1 การทำงานได้ปกติ แต่ DOBOT ตัวที่ 2 การทำงานค่อนข้างมีความผิดพลาดเล็กน้อย ดังตาราง 4.3

ตาราง 4.3 ตรวจสอบการทำงานของ DOBOT หลังการปรับปรุง

จำนวนครั้ง	DOBOT ตัวที่ 1		DOBOT ตัวที่ 2		สาเหตุของความผิดพลาด
	ทำงานปกติ	ทำงานผิดปกติ	ทำงานปกติ	ทำงานผิดปกติ	
1	/			/	
2	/		/		
3	/		/		
4	/		/		
5	/		/		
6	/		/		
7	/		/		
8	/		/		
9	/			/	
10	/		/		
11	/		/		
12	/		/		
13	/		/		
14	/		/		
15	/		/		
16	/		/		
17	/			/	
18	/		/		
19	/		/		
20	/		/		
รวม	20	0	11	9	

เซนเซอร์อินฟาเรด (Infrared Reflectance Sensor) ทั้ง 2 ตัว ทำงานได้เป็นส่วนใหญ่ปกติแต่ก็ยังมีบ่างส่วนที่ทำงานผิดพลาดแต่น้อยมาก ดังตาราง 4.4

ตาราง 4.4 ตรวจสอบการทำงานของเซนเซอร์นับจำนวนของเสียหลังการปรับปรุง

จำนวนครั้ง	การตรวจเซนเซอร์แบบ monitor		การตรวจเซนเซอร์แบบ real time		สาเหตุของความผิดพลาด
	ทำงาน	ไม่ทำงาน	ส่งข้อมูล	ไม่ส่งข้อมูล	
1		/		/	
2	-	-	-	-	
3	-	-	-	-	
4	-	-	-	-	
5	-	-	-	-	
6	-	-	-	-	
7	-	-	-	-	
8	-	-	-	-	
9	/		/		
10	-	-	-	-	
11	-	-	-	-	
12	-	-	-	-	
13	-	-	-	-	
14	-	-	-	-	
15	-	-	-	-	
16	-	-	-	-	
17	/	-	/	-	
18	-	-	-	-	
19	-	-	-	-	
20	-	-	-	-	
รวม	2	1	2	1	

4.4 การทำงานของเครื่องจักร

4.4.1 กระบวนการที่ 1 DOBOT ตัวที่ 1 เคลื่อนที่ไปที่ตำแหน่งปลอดภัย แล้วทำการหยิบกล่องฝาเปิดจากตำแหน่งที่กำหนดไว้ นำมาวางบนสายพาน ก็จะมีโฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) 2 ตัว ตรวจจับว่ามีการวางอยู่ตรงกลางสานพาน ถ้าวางไม่ตรงสายพาน โฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) จะตรวบพบ จากนั้น DOBOT ตัวที่ 1 ทำการดันกล่องช่องเก็บแล้วจะมีเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor) นับจำนวนความผิดพลาดของ DOBOT แล้วความผิดพลาดจะถูกส่งข้อมูล เพื่อนำไปแสดงบนในเว็ป

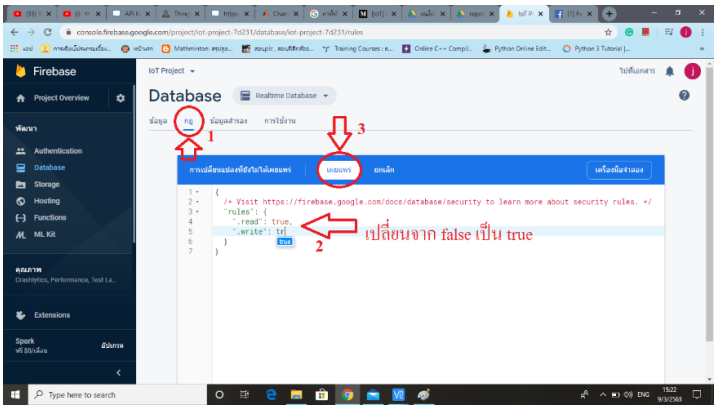
4.4.2 กระบวนการที่ 2 โฟโตอิเล็กทริคเซ็นเซอร์ (Photoelectric Sensor) ตรวจจับว่ามีกล่องถ้าพบกล่อง สายพานจะเคลื่อนที่ไปยังตำแหน่งต่อไป แต่ถ้าไม่พบกล่องกระบวนการจะหยุดทันที

4.4.3 กระบวนการที่ 3 เมื่อกล่องมายังตำแหน่งที่กำหนดไว DOBOT ตัวที่ 2 เคลื่อนที่ไปที่ตำแหน่งปลอดภัย แล้วทำการหยิบฝากล่องในตำแหน่งที่กำหนดไว้ นำมาปิดฝากล่อง

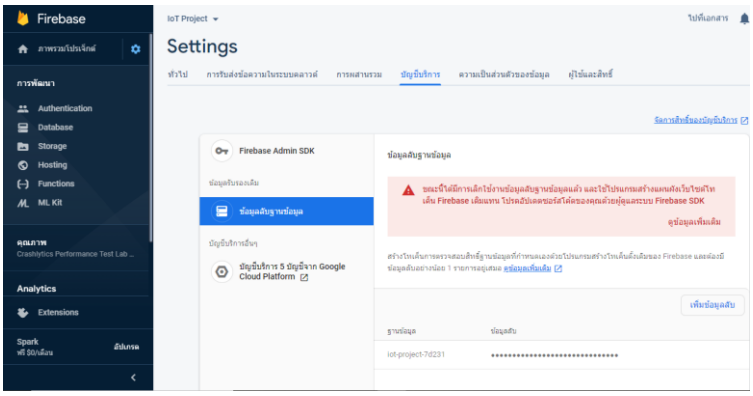
4.4.4 กระบวนการที่ 4 คัลเลอร์ดีเทคเตอร์เซ็นเซอร์ (Color Detector Sensor) ตรวจสอบการปิดกล่อง ถ้าปิดกล่องได้สายพานจะเคลื่อนที่ไปสุดสายพาน แต่ถ้าฝากล่องไม่ถูกปิดในทุกๆ กรณี DOBOT ตัวที่ 2 ทำการดันกล่องช่องเก็บ แล้วจะมีเซนเซอร์อินฟาเรด (Infrared Reflectance Sensor) นับจำนวนความผิดพลาดของ DOBOT แล้วความผิดพลาดจะถูกส่งข้อมูลเพื่อนำไปแสดงบนในเว็ป

4.5 การส่งของข้อมูลจาก Raspberry Pi ไป Firebase Google

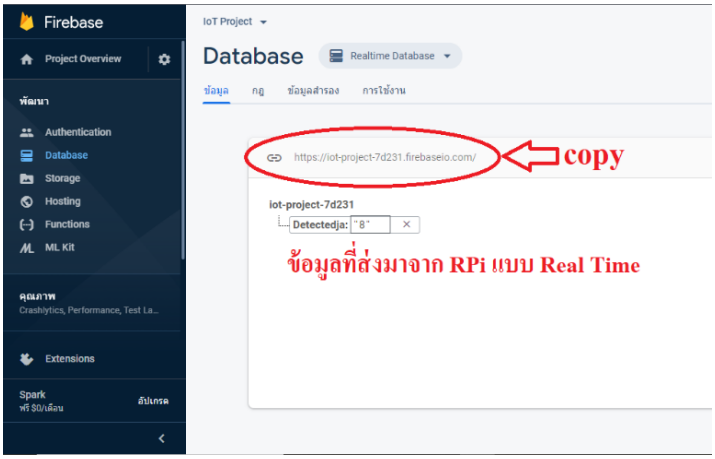
การส่งข้อมูลแบบตอบสนองทันที (Real Time) ลักษณะสำคัญที่สุดของระบบปฏิบัติการเวลาจริงจะต้องตอบสนองโดยทันทีต่อกระบวนการเวลาจริงในไม่ช้าเท่ากระบวนการนั่นต้องการ โดยที่ไฟร์เบส กูเกิล (Firebase Google)จะเป็นตัวรับข้อมูลจากราสเบอร์รี่ พาย (Raspberry Pi) โดยจะต้องทำการสมัครสมาชิกบนเว็บไซต์ไฟร์เบส กูเกิล (Firebase Google)เพื่อใช้งานโดยการที่จะส่งข้อมูลหากันได้นั้นจะเกี่ยวข้องของกฎในฐานข้อมูลแบบทันที (Real Time Data Base) ซึ่งเป็นกฎในการแก้ไขและเข้าถึงข้อมูล โดยจะต้องเปลี่ยน .read และ .write ให้เป็นจริง (True) แล้วคลิกเพื่อเผยแพร่ ดังภาพ 4.3 โดยจะต้องทราบรหัส Token ซึ่งจะเป็นรหัสข้อมูลลับสำหรับเชื่อมต่อระหว่างราสเบอร์รี่ พาย (Raspberry Pi) กับไฟร์เบส กูเกิล (Firebase Google) โดยจะใส่รหัสข้อมูลลับลงไปในการเขียนโปรแกรม Qt Creator ซึ่งรหัสของมูลลัพธ์จะอยู่ที่ข้อมูลลัพธ์ฐานข้อมูล ดังภาพ 4.4 และทำการคัดลอกลิงค์ที่อยู่ในฐานข้อมูล (Data Base) ดังภาพ 4.5 เพื่อนำไปเขียนใน Qt Creator อีกเช่นกัน



ภาพ 4.3 การแก้ไข และเข้าถึงข้อมูลของกฎในฐานข้อมูลแบบทันที (Real Time Data Base)



ภาพ 4.4 รหัสข้อมูลลับ



ภาพ 4.5 ฐานข้อมูล (Data Base)

บทที่ 5

สรุปผลและข้อเสนอแนะ

การวิจัยการประยุกต์ใช้หุ่นยนต์อุตสาหกรรม และระบบ Internet of Things (IoT) นี้ตั้งแต่ต้นจนจบการวิจัย ทางผู้วิจัยได้รับเทคนิคการสร้าง และการออกแบบระบบรวมทั้งความรู้ประสบการณ์ด้านต่างๆ โดยผลการวิจัยอยู่ในระดับที่น่าพอใจเป็นอย่างมาก

5.1 สรุปผลที่ได้จากโครงงานวิจัย

ผู้วิจัยโครงงานได้ทำการวางแผน และลงมือสร้างเครื่องจักรที่สามารถทำการปิดฝากล่องได้อย่างสนิทแบบระบบ Internet of Things (IoT) ก็สามารถส่งข้อมูลแบบตอบสนองทันที (Real Time) และกระบวนการทำงานสามารถทำงานได้ปกติ จากนั้นได้ทำการออกแบบหุ่นยนต์ด้วยโปรแกรมคอมพิวเตอร์เพื่อเป็นแบบในการสร้างเครื่องจักรออกมา เมื่อได้แบบแล้วทางผู้วิจัยทำการสั่งซื้ออุปกรณ์ตามการออกแบบโดยจะแบ่งเป็น 2 ส่วนหลักๆ คือ อุปกรณ์ระบบ Internet of Things (IoT) และอุปกรณ์ทางโครงสร้างหลังจากนั้นได้ประกอบเครื่องจักรขึ้นมา และทำการเริ่มการทำงานขอบระบบครั้งแรก ต่อมาได้เจอปัญหาต่างๆ โดยในการเดินเครื่องครั้งแรกเช่น ขั้นตอนการทำงานที่ไม่เป็นไปตามการออกแบบปัญหาความคลาดเคลื่อนของ DOBOT ในเรื่องของตำแหน่งการหยิบกล่อง และการทำงานของเซนเซอร์ต่างๆ ที่มีการทำงานที่ไม่คงที่ จากนั้นผู้วิจัยได้ทำการปรับปรุง หลังการปรับปรุงผลที่ได้ออกมาพึ่งพอใจมาก กระบวนการสามารถทำงานได้ มีความผิดพลาดในการทำงานน้อยมาก และลงข้อมูลแบบตอบสนองทันที (Real Time) ได้

5.2 ปัญหาที่พบในการดำเนินโครงงานวิจัย

5.2.1 DOBOT มีความคลาดเคลื่อนในการหยิบกล่อง

5.2.2 ผู้ปฏิบัติวางฝาผิดตำแหน่งทำให้ DOBOT หยิบผิดตำแหน่ง

5.2.3 ตำแหน่งคัลเลอร์ดีเทคเตอร์เซ็นเซอร์ (Color Detector Sensor) อยู่ต่ำเกินไป

5.2.4 ตำแหน่งเซนเซอร์อยู่ห่างจากชิ้นงานมากเกินไป

5.2.5 ตัวกล่องมีสีทึบมากเกินไปทำให้เซนเซอร์ไม่สามารถตรวจพบ

5.2.6 อินเตอร์เน็ตเกิดความขัดข้อง

5.2.7 เซนเซอร์เสีย

5.3 ข้อเสนอแนะของโครงงานวิจัย

5.3.1 ในการสร้างเครื่องจักรนั้นต้องใช้ความรู้ในหลายด้านทั้งภาควิชาเครื่องกล เรื่องการออกแบบเครื่องจักรโดยโปรแกรมคอมพิวเตอร์ ภาควิชาไฟฟ้าใน เรื่องการต่อวงจร ภาควิชาอุตสาหการ เรื่องการวางแผนกระบวนการผลิต ภาควิชาคอมพิวเตอร์ เรื่องการเขียนโปรแกรมหุ่นยนต์โดยภาษา Phyton ดังนั้นจึงต้องใช้เวลาในการศึกษา และขอคำปรึกษาจากอาจารย์หรือผู้มีความรู้ เพื่อไม่ให้เกิดปัญหา หรืออุบัติเหตุได้

5.3.2 ควรวางแผนกระบวนการให้รอบครอบ และทำการจำลองสถานการณ์ให้ก่อน เพื่อดูข้อจำกัดในด้านต่างๆ และทำการแก้ไขแบบ

5.3.3 ในการต่อระบบต่างๆ ควรเช็คอุปกรณ์ก่อน ว่าสามารถใช้งานได้หรือไม่

บรรณานุกรม

ชัชชัย คุณบัว. IoT: สถาปัตยกรรมการสื่อสาร Internet of Things. กรุงเทพฯ : ซีเอ็ดยูเคชั่น, 2562.

มาโนชญ์ แสงศิริ. “Raspberry Pi คอมพิวเตอร์ขนาดเล็กสำหรับด้านการศึกษา” [ระบบออไลน์]. แหล่งที่มา https://www.scimath.org/article-technology/item/9104-raspberry-pi?fbclid=IwAR1RUrcxpUHudzuM3Bu5ulyDCnvt-TslUEo5Qt1Cg3id0v9CwxqUh7omDoE (10 September 2019)

สถาพร ลักษณธเจริญ. วิศวกรรมหุ่นยนต์. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย - ญี่ปุ่น), 2548.

อรรณพ เรืองวิเศษ และกฤษดา วิศวธีรานนท์. เปิดโลกหุ่นยนต์ สำหรับนักประดิษฐ์รุ่นใหม่. กรุงเทพฯ : สมาคมส่งเสริมเทคโนโลยี (ไทย -ญี่ปุ่น), 2548.

Unknown. “Dobot magician” [ระบบออไลน์]. แหล่งที่มา https://fra15compro.blogspot.com/?fbclid=IwAR2WRMpiG5f513KX7QI_Gowuuv4bnjDWDHOZCDAEX5NEuKVQvVzXwYWyDU8 (8 September 2019)

ภาคผนวก ก

โคดจากบอร์ดราสเบอร์รี่ พาย (Raspberry Pi)

QtFirebase

```
QT       += core gui sql network
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets
TARGET = MyQtFirebaseDb
TEMPLATE = app
SOURCES += main.cpp\
        mainwindow.cpp \
    json.cpp \
    firebase.cpp \
    datasnapshot.cpp
HEADERS  += mainwindow.h \
    json.h \
    firebase.h \
    datasnapshot.h
FORMS    += mainwindow.ui
LIBS += -L/usr/local/lib -lwiringPi -lwiringPiDev
INCLUDEPATH +=/usr/local/include
RESOURCES += \
    resouresrpi.qrc
```

Headers

Fire datasnapshot.h

```
#ifndef DATASNAPSHOT_H
#define DATASNAPSHOT_H
#include <QObject>
#include <QVariantMap>
class DataSnapshot : public QObject
{
    Q_OBJECT
public:
    explicit DataSnapshot(QObject *parent = 0);
    DataSnapshot(QString);
    QVariantMap getDataMap();
    QString getPath();
    QString getValue();
signals:

public slots:
private:
    QString rawData;
    void parse();
    QVariantMap dataMap;
    QString data_path;
    QString data_value;
    QString trimValue(const QString &line) const;
};
#endif // DATASNAPSHOT_H
```

firebase.h

```
#ifndef FIREBASE_H
#define FIREBASE_H
#include <QObject>
#include <QtNetwork/QNetworkAccessManager>
#include <QtNetwork/QNetworkReply>
#include <QtNetwork/QNetworkRequest>
#include <QUrl>
#include <QUrlQuery>
#include <QDebug>
#include <QtGlobal>
#include <datasnapshot.h>
class Firebase : public QObject
{
    Q_OBJECT
public:
    explicit Firebase(QObject *parent = 0);
    Firebase(QString hostName,QString child);
    Firebase(QString hostName);
    void init();
    void setValue(QString str);
    void getValue();
    void deleteValue();
    void setToken(QString);
    void listenEvents();
    Firebase* child(QString childName);
signals:
    void eventResponseReady(QString);
    void eventDataChanged(DataSnapshot*);
public slots:
    void replyFinished(QNetworkReply*);
```

```
    void onReadyRead(QNetworkReply*);
    void eventFinished();
    void eventReadyRead();
private:
    QString host;
    QString firebaseToken="";
    QNetworkAccessManager *manager;
    QString currentNode;
    QString latestNode;
    QString buildPath(int);
    QString createJson(QString);
    void open(const QUrl &url);
    QByteArray trimValue(const QByteArray &line) const;
};
#endif // FIREBASE_H
```

json.h

```
/**
 * QtJson - A simple class for parsing JSON data into a QVariant hierarchies and vice-versa.
 * Copyright (C) 2011  Eeli Reilin
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
/**
 * \file json.h
 */

#ifndef JSON_H
#define JSON_H
#include <QVariant>
#include <QString>
/**
 * \namespace QtJson
 * \brief A JSON data parser
 *
```

```
 * Json parses a JSON data into a QVariant hierarchy.
 */
namespace QtJson {
    typedef QVariantMap JsonObject;
    typedef QVariantList JsonArray;
    /**
     * Clone a JSON object (makes a deep copy)
     *
     * \param data The JSON object
     */
    QVariant clone(const QVariant &data);

    /**
     * Insert value to JSON object (QVariantMap)
     *
     * \param v The JSON object
     * \param key The key
     * \param value The value
     */
    void insert(QVariant &v, const QString &key, const QVariant &value);
    /**
     * Append value to JSON array (QVariantList)
     *
     * \param v The JSON array
     * \param value The value
     */
    void append(QVariant &v, const QVariant &value);
    /**
     * Parse a JSON string
     *
     * \param json The JSON data
```

```
 */
QVariant parse(const QString &json);
/**
 * Parse a JSON string
 *
 * \param json The JSON data
 * \param success The success of the parsing
 */
QVariant parse(const QString &json, bool &success);
/**
 * This method generates a textual JSON representation
 *
 * \param data The JSON data generated by the parser.
 *
 * \return QByteArray Textual JSON representation in UTF-8
 */
QByteArray serialize(const QVariant &data);
/**
 * This method generates a textual JSON representation
 *
 * \param data The JSON data generated by the parser.
 * \param success The success of the serialization
 *
 * \return QByteArray Textual JSON representation in UTF-8
 */
QByteArray serialize(const QVariant &data, bool &success);
/**
 * This method generates a textual JSON representation
 *
 * \param data The JSON data generated by the parser.
 *
```

```
     * \return QString Textual JSON representation
     */
    QString serializeStr(const QVariant &data);
    /**
     * This method generates a textual JSON representation
     *
     * \param data The JSON data generated by the parser.
     * \param success The success of the serialization
     * \return QString Textual JSON representation
     */
    QString serializeStr(const QVariant &data, bool &success);
    /**
     * This method sets date(time) format to be used for QDateTime::toString
     * If QString is empty, Qt::TextDate is used.
     *
     * \param format The JSON data generated by the parser.
     */
    void setDateTimeFormat(const QString& format);
    void setDateFormat(const QString& format);
    /**
     * This method gets date(time) format to be used for QDateTime::toString
     * If QString is empty, Qt::TextDate is used.
     */
    QString getDateTimeFormat();
    QString getDateFormat();
}
#endif //JSON_H
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <datasnapshot.h>
#include <QTimer>
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    void updateFirebase(QString key, QString value);
private slots:
    void onResponseReady(QString data);
    void onDataChanged(DataSnapshot *data);
    void putSuccessful(QString result);
    void on_pushButton_clicked();
    static void interrupt();
private:
    Ui::MainWindow *ui;
    QPalette* paletteOfLCD;
    QTimer *timer;
};
#endif // MAINWINDOW_H
```

Sources

datasnapshot.cpp

```
#include "datasnapshot.h"
#include <qdebug.h>
#include <qjsondocument.h>
#include <qjsonobject.h>
#include <QVariantMap>
DataSnapshot::DataSnapshot(QObject *parent) : QObject(parent)
{
}
DataSnapshot::DataSnapshot(QString data)
{
    rawData=data;
    parse();
}
void DataSnapshot::parse()
{
    rawData=trimValue(rawData);
    QJsonDocument doc = QJsonDocument::fromJson(rawData.toUtf8());
    QVariantMap eventMap = doc.toVariant().toMap();
    QString path = eventMap["path"].toString();
    QString value = eventMap["data"].toString();
    QVariantMap qDataMap = eventMap["data"].toMap();
    data_path=path;
    data_value=value;
    dataMap=qDataMap;
}
QString DataSnapshot::trimValue(const QString &line) const
{
    QString value;
    int index = line.indexOf(':');
```

```
    if (index > 0)
        value = line.right(line.size() - index  - 1);
    return value.trimmed();
}
QString DataSnapshot::getPath()
{
    return data_path;
}
QString DataSnapshot::getValue()
{
    return data_value;
}
QVariantMap DataSnapshot::getDataMap()
{
    return dataMap;
}
```

firebase.cpp

```
#include "firebase.h"
#include <string.h>
#include <QIODevice>
#include <QBuffer>
#include <QJsonDocument>
#include <datasnapshot.h>
Firebase::Firebase(QObject *parent) :
    QObject(parent)
{
}
void Firebase::init()
{
    manager=new QNetworkAccessManager(this);

connect(manager,SIGNAL(finished(QNetworkReply*)),this,SLOT(replyFinished(QNetwork
Reply*)));
}
Firebase::Firebase(QString hostName)
{
    host=hostName;
    currentNode="";
    init();
}
void Firebase::setToken(QString token)
{
    firebaseToken=token;
}
Firebase::Firebase(QString hostName,QString child)
{
    host=hostName
```

```
            .append(child).append("/");
    latestNode=child;
    init();
}
Firebase* Firebase::child(QString childName)
{
    Firebase *childNode=new Firebase(host,childName);
    childNode->setToken(firebaseToken);
    return childNode;
}
void Firebase::open(const QUrl &url)
{
    QNetworkRequest request(url);
    request.setRawHeader("Accept",
                         "text/event-stream");
    QNetworkReply *_reply = manager->get(request);
    connect(_reply, &QNetworkReply::readyRead, this, &Firebase::eventReadyRead);
    connect(_reply, &QNetworkReply::finished, this, &Firebase::eventFinished);
}
void Firebase::eventFinished()
{
    QNetworkReply *reply = qobject_cast<QNetworkReply*>(sender());
    if (reply)
    {
        QUrl redirectUrl = reply-
>attribute(QNetworkRequest::RedirectionTargetAttribute).toUrl();
        if (!redirectUrl.isEmpty())
        {
            reply->deleteLater();
            open(redirectUrl);
            return;
```

```
        }
        reply->deleteLater();
    }
}
void Firebase::eventReadyRead()
{
    QNetworkReply *reply = qobject_cast<QNetworkReply*>(sender());
    if(reply)
    {
        QByteArray line=reply->readLine();
        if(!line.isEmpty())
        {
            QByteArray eventName=trimValue(line);
            line=reply->readAll();
            if(eventName=="put")
            {
                DataSnapshot *dataSnapshot=new DataSnapshot(line);
                emit eventDataChanged(dataSnapshot);
            }
        }
    }
    reply->readAll();
}
void Firebase::onReadyRead(QNetworkReply *reply)
{
    /*qDebug()<<"incoming data";
    qDebug()<<reply->readAll();*/
}
void Firebase::replyFinished(QNetworkReply *reply)
{
    //qDebug()<<reply->readAll();
```

```
    QString data=QString(reply->readAll());
    emit eventResponseReady(data);
}
void Firebase::setValue(QString strVal)
{
    //Json data creation
    QNetworkRequest request(buildPath(1));
    request.setHeader(QNetworkRequest::ContentTypeHeader,
                      "application/x-www-form-urlencoded");
    QBuffer *buffer=new QBuffer();
    buffer->open((QBuffer::ReadWrite));
    buffer->write(createJson(strVal).toUtf8());
    buffer->seek(0);
    /*
     * To be able to send "PATCH" request sendCustomRequest method is used.
     * sendCustomRequest requires a QIOdevice so QBuffer is used.
     * I had to seek 0 because it starts reading where it paused.
     */
    manager->sendCustomRequest(request,"PATCH",buffer);
    buffer->close();
}
void Firebase::getValue()
{
    QNetworkRequest request(buildPath(0));
    manager->get(request);
}
void Firebase::listenEvents()
{
    open(buildPath(0));
}
void Firebase::deleteValue()
```

```
{
    QNetworkRequest request(buildPath(0));
    manager->deleteResource(request);
}
QString Firebase::createJson(QString str)
{
    QString
data=QString(QString("{").append("\"").append(latestNode).append("\"").append(":").appe
nd("\"").append(str).append("\"").append(QString("}")));
    return data;
}
QString Firebase::buildPath(int mode)
{
    QString destination="";
    if(mode)
    {
        host.replace(QString("/").append(latestNode).append("/"),"");
        destination
                .append(host)
                .append("/.json");
    }
    else
    {
        destination
                .append(host)
                .append(currentNode)
                .append(".json");
    }
    if(!firebaseToken.isEmpty())
        destination.append("?auth=").append(firebaseToken);
    return destination;
```

```
}
QByteArray Firebase::trimValue(const QByteArray &line) const
{
    QByteArray value;
    int index = line.indexOf(':');
    if (index > 0)
        value = line.right(line.size() - index  - 1);
    return value.trimmed();
}
```

json.cpp

```
/**
 * QtJson - A simple class for parsing JSON data into a QVariant hierarchies and vice-versa.
 * Copyright (C) 2011  Eeli Reilin
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */
/**
 * \file json.cpp
 */
#include <QDateTime>
#include <QStringList>
#include "json.h"
namespace QtJson {
    static QString dateFormat, dateTimeFormat;
    static QString sanitizeString(QString str);
    static QByteArray join(const QList<QByteArray> &list, const QByteArray &sep);
    static QVariant parseValue(const QString &json, int &index, bool &success);
    static QVariant parseObject(const QString &json, int &index, bool &success);
```

```
    static QVariant parseArray(const QString &json, int &index, bool &success);
    static QVariant parseString(const QString &json, int &index, bool &success);
    static QVariant parseNumber(const QString &json, int &index);
    static int lastIndexOfNumber(const QString &json, int index);
    static void eatWhitespace(const QString &json, int &index);
    static int lookAhead(const QString &json, int index);
    static int nextToken(const QString &json, int &index);
    template<typename T>
    QByteArray serializeMap(const T &map, bool &success) {
        QByteArray str = "{ ";
        QList<QByteArray> pairs;
        for (typename T::const_iterator it = map.begin(), itend = map.end(); it != itend;
++it) {
            QByteArray serializedValue = serialize(it.value());
            if (serializedValue.isNull()) {
                success = false;
                break;
            }
            pairs << sanitizeString(it.key()).toUtf8() + " : " + serializedValue;
        }
        str += join(pairs, ", ");
        str += " }";
        return str;
    }
    void insert(QVariant &v, const QString &key, const QVariant &value);
    void append(QVariant &v, const QVariant &value);
    template<typename T>
    void cloneMap(QVariant &json, const T &map) {
          for (typename T::const_iterator it = map.begin(), itend = map.end(); it != itend;
++it) {
              insert(json, it.key(), (*it));
```

```
        }
    }

    template<typename T>
    void cloneList(QVariant &json, const T &list) {
        for (typename T::const_iterator it = list.begin(), itend = list.end(); it != itend;
++it) {
            append(json, (*it));
        }
    }
    /**
     * parse
     */
    QVariant parse(const QString &json) {
        bool success = true;
        return parse(json, success);
    }
    /**
     * parse
     */
    QVariant parse(const QString &json, bool &success) {
        success = true;
        // Return an empty QVariant if the JSON data is either null or empty
        if (!json.isNull() || !json.isEmpty()) {
            QString data = json;
            // We'll start from index 0
            int index = 0;
            // Parse the first value
            QVariant value = parseValue(data, index, success);
            // Return the parsed value
            return value;
```

```
    } else {
        // Return the empty QVariant
        return QVariant();
    }
}
/**
 * clone
 */
QVariant clone(const QVariant &data) {
    QVariant v;
    if (data.type() == QVariant::Map) {
        cloneMap(v, data.toMap());
    } else if (data.type() == QVariant::Hash) {
        cloneMap(v, data.toHash());
    } else if (data.type() == QVariant::List) {
        cloneList(v, data.toList());
    } else if (data.type() == QVariant::StringList) {
        cloneList(v, data.toStringList());
    } else {
        v = QVariant(data);
    }
    return v;
}
/**
 * insert value (map case)
 */
void insert(QVariant &v, const QString &key, const QVariant &value) {
    if (!v.canConvert<QVariantMap>()) v = QVariantMap();
    QVariantMap *p = (QVariantMap *)v.data();
    p->insert(key, clone(value));
}
```

```
/**
 * append value (list case)
 */
void append(QVariant &v, const QVariant &value) {
       if (!v.canConvert<QVariantList>()) v = QVariantList();
       QVariantList *p = (QVariantList *)v.data();
       p->append(value);
}
QByteArray serialize(const QVariant &data) {
    bool success = true;
    return serialize(data, success);
}
QByteArray serialize(const QVariant &data, bool &success) {
    QByteArray str;
    success = true;
    if (!data.isValid()) { // invalid or null?
        str = "null";
    } else if ((data.type() == QVariant::List) ||
               (data.type() == QVariant::StringList)) { // variant is a list?
        QList<QByteArray> values;
        const QVariantList list = data.toList();
        Q_FOREACH(const QVariant& v, list) {
            QByteArray serializedValue = serialize(v);
            if (serializedValue.isNull()) {
                success = false;
                break;
            }
            values << serializedValue;
        }
        str = "[ " + join( values, ", " ) + " ]";
    } else if (data.type() == QVariant::Hash) { // variant is a hash?
```

```
        str = serializeMap<>(data.toHash(), success);
    } else if (data.type() == QVariant::Map) { // variant is a map?
        str = serializeMap<>(data.toMap(), success);
    } else if ((data.type() == QVariant::String) ||
               (data.type() == QVariant::ByteArray)) {// a string or a byte array?
        str = sanitizeString(data.toString()).toUtf8();
    } else if (data.type() == QVariant::Double) { // double?
        double value = data.toDouble(&success);
        if (success) {
            str = QByteArray::number(value, 'g');
            if (!str.contains(".") && ! str.contains("e")) {
                str += ".0";
            }
        }
    } else if (data.type() == QVariant::Bool) { // boolean value?
        str = data.toBool() ? "true" : "false";
    } else if (data.type() == QVariant::ULongLong) { // large unsigned number?
        str = QByteArray::number(data.value<qulonglong>());
    } else if (data.canConvert<qlonglong>()) { // any signed number?
        str = QByteArray::number(data.value<qlonglong>());
    } else if (data.canConvert<long>()) { //TODO: this code is never executed
because all smaller types can be converted to qlonglong
        str = QString::number(data.value<long>()).toUtf8();
    } else if (data.type() == QVariant::DateTime) { // datetime value?
        str = sanitizeString(dateTimeFormat.isEmpty()
                             ? data.toDateTime().toString()
                             : data.toDateTime().toString(dateTimeFormat)).toUtf8();
    } else if (data.type() == QVariant::Date) { // date value?
        str = sanitizeString(dateTimeFormat.isEmpty()
                             ? data.toDate().toString()
                             : data.toDate().toString(dateFormat)).toUtf8();
```

```
    } else if (data.canConvert<QString>()) { // can value be converted to string?
        // this will catch QUrl, ... (all other types which can be converted to string)
        str = sanitizeString(data.toString()).toUtf8();
    } else {
        success = false;
    }
    if (success) {
        return str;
    }
    return QByteArray();
}
QString serializeStr(const QVariant &data) {
    return QString::fromUtf8(serialize(data));
}
QString serializeStr(const QVariant &data, bool &success) {
    return QString::fromUtf8(serialize(data, success));
}
/**
 * \enum JsonToken
 */
enum JsonToken {
    JsonTokenNone = 0,
    JsonTokenCurlyOpen = 1,
    JsonTokenCurlyClose = 2,
    JsonTokenSquaredOpen = 3,
    JsonTokenSquaredClose = 4,
    JsonTokenColon = 5,
    JsonTokenComma = 6,
    JsonTokenString = 7,
    JsonTokenNumber = 8,
    JsonTokenTrue = 9,
```

```
    JsonTokenFalse = 10,
    JsonTokenNull = 11
};
static QString sanitizeString(QString str) {
    str.replace(QLatin1String("\\"), QLatin1String("\\\\"));
    str.replace(QLatin1String("\""), QLatin1String("\\\""));
    str.replace(QLatin1String("\b"), QLatin1String("\\b"));
    str.replace(QLatin1String("\f"), QLatin1String("\\f"));
    str.replace(QLatin1String("\n"), QLatin1String("\\n"));
    str.replace(QLatin1String("\r"), QLatin1String("\\r"));
    str.replace(QLatin1String("\t"), QLatin1String("\\t"));
    return QString(QLatin1String("\"%1\"")).arg(str);
}
static QByteArray join(const QList<QByteArray> &list, const QByteArray &sep) {
    QByteArray res;
    Q_FOREACH(const QByteArray &i, list) {
        if (!res.isEmpty()) {
            res += sep;
        }
        res += i;
    }
    return res;
}
/**
 * parseValue
 */
static QVariant parseValue(const QString &json, int &index, bool &success) {
    // Determine what kind of data we should parse by
    // checking out the upcoming token
    switch(lookAhead(json, index)) {
        case JsonTokenString:
```

```
            return parseString(json, index, success);
        case JsonTokenNumber:
            return parseNumber(json, index);
        case JsonTokenCurlyOpen:
            return parseObject(json, index, success);
        case JsonTokenSquaredOpen:
            return parseArray(json, index, success);
        case JsonTokenTrue:
            nextToken(json, index);
            return QVariant(true);
        case JsonTokenFalse:
            nextToken(json, index);
            return QVariant(false);
        case JsonTokenNull:
            nextToken(json, index);
            return QVariant();
        case JsonTokenNone:
            break;
    }
    // If there were no tokens, flag the failure and return an empty QVariant
    success = false;
    return QVariant();
}
/**
 * parseObject
 */
static QVariant parseObject(const QString &json, int &index, bool &success) {
    QVariantMap map;
    int token;
    // Get rid of the whitespace and increment index
    nextToken(json, index);
```

```
// Loop through all of the key/value pairs of the object
bool done = false;
while (!done) {
    // Get the upcoming token
    token = lookAhead(json, index);
    if (token == JsonTokenNone) {
        success = false;
        return QVariantMap();
    } else if (token == JsonTokenComma) {
        nextToken(json, index);
    } else if (token == JsonTokenCurlyClose) {
        nextToken(json, index);
        return map;
    } else {
        // Parse the key/value pair's name
        QString name = parseString(json, index, success).toString();
        if (!success) {
            return QVariantMap();
        }
        // Get the next token
        token = nextToken(json, index);
        // If the next token is not a colon, flag the failure
        // return an empty QVariant
        if (token != JsonTokenColon) {
            success = false;
            return QVariant(QVariantMap());
        }
        // Parse the key/value pair's value
        QVariant value = parseValue(json, index, success);
        if (!success) {
            return QVariantMap();
```

```
            }
            // Assign the value to the key in the map
            map[name] = value;
        }
    }
    // Return the map successfully
    return QVariant(map);
}
/**
 * parseArray
 */
static QVariant parseArray(const QString &json, int &index, bool &success) {
    QVariantList list;
    nextToken(json, index);
    bool done = false;
    while(!done) {
        int token = lookAhead(json, index);
        if (token == JsonTokenNone) {
            success = false;
            return QVariantList();
        } else if (token == JsonTokenComma) {
            nextToken(json, index);
        } else if (token == JsonTokenSquaredClose) {
            nextToken(json, index);
            break;
        } else {
            QVariant value = parseValue(json, index, success);
            if (!success) {
                return QVariantList();
            }
            list.push_back(value);
```

```
        }
    }
    return QVariant(list);
}
/**
 * parseString
 */
static QVariant parseString(const QString &json, int &index, bool &success) {
    QString s;
    QChar c;
    eatWhitespace(json, index);
    c = json[index++];
    bool complete = false;
    while(!complete) {
        if (index == json.size()) {
            break;
        }
        c = json[index++];
        if (c == '\"') {
            complete = true;
            break;
        } else if (c == '\\') {
            if (index == json.size()) {
                break;
            }
            c = json[index++];
            if (c == '\"') {
                s.append('\"');
            } else if (c == '\\') {
                s.append('\\');
            } else if (c == '/') {
```

```
                s.append('/');
            } else if (c == 'b') {
                s.append('\b');
            } else if (c == 'f') {
                s.append('\f');
            } else if (c == 'n') {
                s.append('\n');
            } else if (c == 'r') {
                s.append('\r');
            } else if (c == 't') {
                s.append('\t');
            } else if (c == 'u') {
                int remainingLength = json.size() - index;
                if (remainingLength >= 4) {
                    QString unicodeStr = json.mid(index, 4);
                    int symbol = unicodeStr.toInt(0, 16);
                    s.append(QChar(symbol));
                    index += 4;
                } else {
                    break;
                }
            }
        } else {
            s.append(c);
        }
    }
    if (!complete) {
        success = false;
        return QVariant();
    }
    return QVariant(s);
```

```
}
/**
 * parseNumber
 */
static QVariant parseNumber(const QString &json, int &index) {
    eatWhitespace(json, index);
    int lastIndex = lastIndexOfNumber(json, index);
    int charLength = (lastIndex - index) + 1;
    QString numberStr;
    numberStr = json.mid(index, charLength);
    index = lastIndex + 1;
    bool ok;
    if (numberStr.contains('.')) {
        return QVariant(numberStr.toDouble(NULL));
    } else if (numberStr.startsWith('-')) {
        int i = numberStr.toInt(&ok);
        if (!ok) {
            qlonglong ll = numberStr.toLongLong(&ok);
            return ok ? ll : QVariant(numberStr);
        }
        return i;
    } else {
        uint u = numberStr.toUInt(&ok);
        if (!ok) {
            qulonglong ull = numberStr.toULongLong(&ok);
            return ok ? ull : QVariant(numberStr);
        }
        return u;
    }
}
/**
```

```
 * lastIndexOfNumber
 */
static int lastIndexOfNumber(const QString &json, int index) {
    int lastIndex;
    for(lastIndex = index; lastIndex < json.size(); lastIndex++) {
        if (QString("0123456789+-.eE").indexOf(json[lastIndex]) == -1) {
            break;
        }
    }
    return lastIndex -1;
}
/**
 * eatWhitespace
 */
static void eatWhitespace(const QString &json, int &index) {
    for(; index < json.size(); index++) {
        if (QString(" \t\n\r").indexOf(json[index]) == -1) {
            break;
        }
    }
}
/**
 * lookAhead
 */
static int lookAhead(const QString &json, int index) {
    int saveIndex = index;
    return nextToken(json, saveIndex);
}
/**
 * nextToken
 */
```

```
static int nextToken(const QString &json, int &index) {
    eatWhitespace(json, index);
    if (index == json.size()) {
        return JsonTokenNone;
    }
    QChar c = json[index];
    index++;
    switch(c.toLatin1()) {
        case '{': return JsonTokenCurlyOpen;
        case '}': return JsonTokenCurlyClose;
        case '[': return JsonTokenSquaredOpen;
        case ']': return JsonTokenSquaredClose;
        case ',': return JsonTokenComma;
        case '"': return JsonTokenString;
        case '0': case '1': case '2': case '3': case '4':
        case '5': case '6': case '7': case '8': case '9':
        case '-': return JsonTokenNumber;
        case ':': return JsonTokenColon;
    }
    index--; // ^ WTF?
    int remainingLength = json.size() - index;
    // True
    if (remainingLength >= 4) {
        if (json[index] == 't' && json[index + 1] == 'r' &&
            json[index + 2] == 'u' && json[index + 3] == 'e') {
            index += 4;
            return JsonTokenTrue;
        }
    }
    // False
    if (remainingLength >= 5) {
```

```
            if (json[index] == 'f' && json[index + 1] == 'a' &&
                json[index + 2] == 'l' && json[index + 3] == 's' &&
                json[index + 4] == 'e') {
                index += 5;
                return JsonTokenFalse;
            }
        }

        // Null
        if (remainingLength >= 4) {
            if (json[index] == 'n' && json[index + 1] == 'u' &&
                json[index + 2] == 'l' && json[index + 3] == 'l') {
                index += 4;
                return JsonTokenNull;
            }
        }
        return JsonTokenNone;
    }
    void setDateTimeFormat(const QString &format) {
        dateTimeFormat = format;
    }
    void setDateFormat(const QString &format) {
        dateFormat = format;
    }
    QString getDateTimeFormat() {
        return dateTimeFormat;
    }
    QString getDateFormat() {
        return dateFormat;
    }
} //end namespace
```

main.cpp

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "firebase.h"
#include <QDebug>
#include <QJsonDocument>
#include <QJsonValue>
#include <QJsonArray>
#include <QJsonObject>
#include <wiringPi.h>
#include <qtimer.h>
#include <qtimeline.h>
#include <QTime>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <string>
#include <sstream>
#define interruptedPin 7
static int count;
static volatile long lastStatusOccured;
static int countmix;
static Ui::MainWindow* sharedUi;
MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow ){
    {
        count = 0;
        sharedUi = ui;
        ui->setupUi(this);
        wiringPiSetup();
```

```
        wiringPiISR(interruptedPin,INT_EDGE_FALLING, &interrupt);
    }
    ui->setupUi(this);
    connect(ui->pushButton, SIGNAL(clicked()), this, SLOT(on_pushButton_clicked()));
    QString url = "https://iot-project-7d231.firebaseio.com/";
        Firebase *myFirebase = new Firebase(url);
        myFirebase->setToken("5AFAtb8Oe9YfBH3KvLnFsuXDLXxifs0QB9YafBQu");
        connect(myFirebase, SIGNAL(eventResponseReady(QString)), this,
SLOT(onResponseReady(QString)));
        connect(myFirebase, SIGNAL(eventDataChanged(DataSnapshot*)), this,
SLOT(onDataChanged(DataSnapshot*)));
        myFirebase->listenEvents();
        myFirebase->getValue();
        wiringPiSetup();
        QTimer *timer = new QTimer(this);
        connect(timer, SIGNAL(timeout()), this, SLOT(interrupt()));
       this->updateFirebase("Detected", sharedUi->lineEdit->text());
}
//sensor//
void MainWindow::interrupt(){
long currentMillis = long (millis());
if (currentMillis - lastStatusOccured > 100){
    count++;
    qDebug() << "Detected=>>" + QString::number(count/2);
    sharedUi->mLcdNumber->display(count/2);
}
if( count/2 >0){
     countmix  = count/2;
     qDebug() << countmix;
     sharedUi->lineEdit ->setText(QString::number(countmix));
             }
```

```
void delay();
     QTime dieTime= QTime::currentTime().addSecs(1);
     while (QTime::currentTime() < dieTime)
        QCoreApplication::processEvents(QEventLoop::AllEvents, 5000);
        lastStatusOccured = currentMillis;
}
//sensor//
void MainWindow::onResponseReady(QString data){
    qDebug() << "Response:" << data;
    QJsonObject jsonObj = QJsonDocument::fromJson(data.toUtf8()).object();
    qDebug() << "IR: " << jsonObj["IRSensorStatus"].toDouble();
}
void MainWindow::onDataChanged(DataSnapshot *data){
    QString path = data->getPath();
    qDebug() << "str path " << path;
    if (path == "/Counter"){
        qDebug () << "Counter: " <<  data->getValue();
    }else if (path == "/Data"){
        qDebug () << "Data: " <<  data->getValue();
    }
}
void MainWindow::putSuccessful(QString result){
     this->updateFirebase("Detected", sharedUi->lineEdit->text());
}
void MainWindow::on_pushButton_clicked(){
    QString url = "https://iot-project-7d231.firebaseio.com/";
    Firebase *myFirebase = new Firebase(url);
    Firebase* firebaseValue = myFirebase->child("Detectedja");
connect(firebaseValue,SIGNAL(eventResponseReady(QString)),this,SLOT(putSuccessful(
QString)));
    firebaseValue->setValue("0");
```

```
}
void MainWindow::updateFirebase(const QString key , const QString value){
    QString url = "https://iot-project-7d231.firebaseio.com/";
    Firebase *myFirebase = new Firebase(url);
    Firebase* firebaseValue = myFirebase->child(key);

connect(firebaseValue,SIGNAL(eventResponseReady(QString)),this,SLOT(putSuccessful(
QString)));
    firebaseValue->setValue(value);
}
MainWindow::~MainWindow(){
    delete ui;
}
```

ประวัติผู้เขียน

ชื่อ นายจิรกร ศรีโปธา รหัสนักศึกษา 590612045

วัน เดือน ปี เกิด 27 พฤษภาคม 2536

ภูมิลำเนา 113/5 หมู่ 6 ตำบลเหมืองง่า อำเภอเมือง จังหวัดลำพูน 51000

ประวัติการศึกษา

- สำเร็จการศึกษาจากชั้นมัธยมตอนต้นจากโรงเรียนจักรคำคณาทร ลำพูน
- สำเร็จการศึกษาจากชั้นมัธยมตอนปลายจากโรงเรียนจักรคำคณาทร ลำพูน
- ปัจจุบันกำลังศึกษาอยู่ที่ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่

ชื่อ นายศุภกิจ เหล่าสามารถ รหัสนักศึกษา 590612095

วัน เดือน ปี เกิด 10 มิถุนายน 2535

ภูมิลำเนา 276 หมู่ 5 ตำบลนางแล อำเภอเมือง จังหวัดเชียงราย 57100

ประวัติการศึกษา

- สำเร็จการศึกษาจากชั้นมัธยมตอนต้นจากโรงเรียนเม็งรายมหาราชวิทยาคม เชียงราย
- สำเร็จการศึกษาจากชั้นมัธยมตอนปลายจากโรงเรียนเม็งรายมหาราชวิทยาคม เชียงราย
- ปัจจุบันกำลังศึกษาอยู่ที่ภาควิชาวิศวกรรมอุตสาหการ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเชียงใหม่