

เครื่องซักรองเท้ากีฬาแบบอัตโนมัติโดยใช้การประมวลผลภาพ

Automatic Sport Shoes Washing Machine using by Image Processing

นางสาวณัฐธิชา พันธ์ธง รหัสนักศึกษา 5711110405

ปริญญาบัตรนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมเมคคาทรอนิกส์ สถาบันนวัตกรรมมหานคร

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีมหานคร

ปีการศึกษา 2561

บทคัดย่อ

ปริญานินพนธ์นำเสนอเกี่ยวกับเครื่องซักรองเท้ากีฬาแบบอัตโนมัติโดยใช้การประมวลผลภาพ โดยมีวัตถุประสงค์หลักในทำปริญานินพนธ์รังนี้เพื่อแก้ปัญหาความยุ่งยากในการซักและการทำให้แห้งรองเท้ากีฬาแห้ง ในเครื่องซักรองเท้ากีฬาแบบอัตโนมัติโดยใช้การประมวลผลภาพ จะมี 2 ระบบ คือ ระบบอัตโนมัติและระบบควบคุมด้วยมือ ซึ่งในระบบอัตโนมัติจะทำงานตั้งแต่การซัก ล้าง ไปจนถึงการทำให้แห้งโดยใช้การประมวลผลภาพ ในส่วนของระบบควบคุมด้วยมือจะเลือกรอบการซักหรือการทำให้แห้งโดยการตั้งเวลาในการทำงาน พบร่วมค่า SSIM ที่ควรนำมาใช้ในการประมวลผลภาพ คือ 0.25 และระบบทำให้แห้งด้วยระบบควบคุมอุณหภูมิของลมตั้งแต่ 40 - 55 องศาเซลเซียส มีค่าความละเอียดครั้งละ 5 องศาเซลเซียส และมีค่าความผิดพลาดไม่เกิน ± 2 องศาเซลเซียส เมื่อระบบทำให้แห้งทำงาน เปอร์เซ็นต์ค่าความชื้นจะลดลงเรื่อยๆ ตามเวลาที่เพิ่มมากขึ้น และในแต่ละช่วงอุณหภูมิจะมีเปอร์เซ็นต์ค่าความชื้นที่ลดลงต่างกัน

Abstract

This thesis proposes about Automatic Sport Shoes Washing Machine using by Image Processing. The main objective of this thesis is for solving the problem of washing and drying sport shoes. This machine is consisting of 2 systems as the automation and manual systems. First, the automation system operates from washing step to drying step using by image processing. Second, the manual system, can select the washing step or drying step by setting the working time. The SSIM value used in image processing is 0.25 and the system makes the shoes dry with the temperature of the wind from 40 to 55 ° C with a resolution of 5 ° C and error values are not exceeded ± 2 ° C when the system dries. Moisture content decreases with increasing time. In each temperature range, the percentage of moisture content decreased.

กิตติกรรมประกาศ

ปริญญาอุดมศึกษาบัณฑิตที่ได้ด้วยความอนุเคราะห์ และความกรุณาจากบุคคลที่ทางผู้จัดทำต้องขอ
กราบขอบพระคุณ คือ บิดา มารดา ของผู้จัดทำทั้งสองที่เคยอบรมสั่งสอนตลอดมา พี่สาวผู้ที่ให้ทุนสนับสนุน
การศึกษา ขอขอบพระคุณนายปารเมศ ลอดทอง นายณัฐกร ชีวีพัฒนา นายบุญอนันต์ เชื่อมคำ นายธนบดี โชติช่วง
นายสินภาส สุขดี และเพื่อน ๆ สาขาวิชาวิศวกรรมระบบวัดคุณ ที่ช่วยเหลือและให้ยืมอุปกรณ์ที่จำเป็นในการทำโปร
เจค และขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ชนม์รัตน์ ตติยะวนันท์ อาจารย์ที่ปรึกษาโครงงาน ท่านได้สละ
เวลาให้กำปรึกษาในเรื่องต่างๆ และยังติดตามสอบถามถึงความคืบหน้าของโครงงานอยู่เป็นระยะ ผู้จัดทำจึง
ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.ชนม์รัตน์ ตติยะวนันท์ เป็นอย่างสูงที่คอยให้คำแนะนำและตรวจสอบความ
ถูกต้องและความเหมาะสมโดยละเอียดจนกระทั้งปริญญาอุดมศึกษาบัณฑิตที่แล้วเสร็จและมีความสมบูรณ์

นางสาวณัฐริชา พันธ์รง

ผู้จัดทำ

สารบัญ

หน้า

บทคัดย่อ	I
กิตติกรรมประกาศ	II
สารบัญ	III
สารบัญตาราง	V
สารบัญรูป	VI
บทที่ 1 บทนำ	1
1.1 วัตถุประสงค์	1
1.2 ขอบเขตการดำเนินงาน	1
1.3 ขั้นตอนการดำเนินงาน	1
บทที่ 2 ทฤษฎีที่เกี่ยวข้อง	4
2.1 การประมวลผลภาพ	4
2.2 ไลบรารีสำหรับใช้งานเรื่องการประมวลผลภาพ	6
2.3 ภาษา Python	7
2.4 MSE และ SSIM	8
2.5 ตัวควบคุม Raspberry Pi	11
2.6 แผ่นอะคริลิก	11
2.7 โมเตอร์กระแสตรงแบบแม่เหล็กถาวร	12
2.8 เซนเซอร์ระดับน้ำ หรือลูกกลอย	13
2.9 โซลินอยด์วาวล์	15
2.10 เซนเซอร์ตรวจวัดอุณหภูมิและความชื้น	18
2.11 ชนและสังเคราะห์	21
2.12 ลดนิโตรם	23
2.13 พัดลมแบบหมุนแรงเหวี่ยง	25
2.14 ระบบลมร้อน	25
บทที่ 3 การออกแบบ	28
3.1 การออกแบบระบบการทำงาน	28
3.2 การออกแบบโครงสร้าง	32
3.3 การออกแบบวงจรไฟฟ้า	34
บทที่ 4 การทดลอง และผลการทดลอง	36
4.1 การทดลองการใช้งานถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน	36
4.2 การทดลองจับเวลาการประมวลผลภาพในค่าพิเศษของภาพที่แตกต่างกันและแสดงค่า MSE และค่า SSIM	37
4.3 การทดลองตรวจสอบค่าของอุณหภูมิที่เพิ่มขึ้นเมื่อเทียบกับเวลาระหว่างเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer	40

สารบัญ (ต่อ)

หน้า

4.4 การทดลองระบบทำให้แห้งในอุณหภูมิตั้งแต่ 40 - 55 องศา มีค่าความละเอียดครั้งละ 5 องศาเซลเซียส ในเวลา 0 – 20 นาที มีค่าความละเอียดครั้งละ 5 นาที	41
4.5 การทดลองระบบซัก ในเวลา 1-8 นาที มีค่าความละเอียดครั้งละ 1 นาที	43
บทที่ 5 สรุปผลการทดลอง	46
5.1 สรุปผลการทดลอง	46
5.2 ปัญหาที่เกิดขึ้น	46
5.3 แนวทางการพัฒนา	46
เอกสารอ้างอิง	47
ภาคผนวก ก. รายงานเสนอขออนุมัติโครงการงานวิศวกรรม	ก-1
ภาคผนวก ข. การออกแบบจาร	ข-1
ภาคผนวก ค. Data Sheets	ค-1
ภาคผนวก ง. โปรแกรมควบคุม	ง-1
ภาคผนวก จ. รูปเครื่องซักรองเท้ากีฬาแบบอัตโนมัติโดยใช้การประมวลผลภาพ	จ-1

สารบัญตาราง

หน้า

ตารางที่ 2.1 รายชื่อสีและค่าที่แทนค่าสีต่าง ๆ	4
ตารางที่ 2.2 การเปรียบเทียบค่า bit per pixel	6
ตารางที่ 4.1 ผลการทดลองการใช้งานถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน	36
ตารางที่ 4.2 ผลการทดลองจับเวลาการประมวลผลภาพในค่าพิกเซลของรูปภาพที่ต่างกันและแสดงค่า MSE และค่า SSIM	38
ตารางที่ 4.3 ผลการทดลองค่าของอุณหภูมิที่เพิ่มขึ้นเมื่อเทียบกับเวลาระหว่างเซ็นเซอร์ DHT22 และ เครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer	40
ตารางที่ 4.4 ผลการทดลองทดลองระบบทำให้แห้งในอุณหภูมิตั้งแต่ 40 - 55 องศา มีค่าความลับเอี้ยด ครั้งละ 5 องศาเซลเซียส ใน เวลา 0 – 20 นาที มีค่าความลับเอี้ยดครั้งละ 5 นาที	41
ตารางที่ 4.5 ผลการทดลองระบบซัก ในเวลา 1-8 นาที มีค่าความลับเอี้ยดครั้งละ 1 นาที	43

สารบัญรูป

หน้า

รูปที่ 1.1 การทำงานของโครงงานวิศวกรรม	2
รูปที่ 2.1 ระดับสีของระบบสีเทาตามขนาดข้อมูลที่เก็บค่าสี	5
รูปที่ 2.2 ภาพแสดงจุด Pixel ภายในภาพ	6
รูปที่ 2.3 Minkowski Metric สำหรับการคำนวณ MSE	8
รูปที่ 2.4 หลักการของรูปแบบการใช้งานร่วมกัน	9
รูปที่ 2.5 ส่วนประกอบของ Raspberry Pi	11
รูปที่ 2.6 ตัวอย่างแผ่นอะคริลิค	12
รูปที่ 2.7 วงจรโมเตอร์กระแสตรงแบบแม่เหล็กถาวร	12
รูปที่ 2.8 กราฟคุณลักษณะของโมเตอร์กระแสตรงแบบแม่เหล็กถาวร	13
รูปที่ 2.9 การอ่านระดับของลูกloy แต่ละแบบ	13
รูปที่ 2.10 การวัดระดับชนิดลูกloy	14
รูปที่ 2.11 การวัดระดับแบบจุดด้วยลูกloy ที่ติดตั้งด้านบนภาชนะ	14
รูปที่ 2.12 การวัดระดับแบบจุดด้วยลูกloy ที่ติดตั้งด้านข้างภาชนะ	14
รูปที่ 2.13 เครื่องมือวัดระดับของเหลวชนิดลูกloy และแม่เหล็ก	15
รูปที่ 2.14 ระบบเปิดปิดโดยตรง	16
รูปที่ 2.15 ระบบเปิดปิดทางอ้อม	17
รูปที่ 2.16 ระบบลูกผู้ชาย	17
รูปที่ 2.17 วิธีคำนวณหาค่าความชื้นสัมพัทธ์	18
รูปที่ 2.18 การส่งสัญญาณ pull down voltage ไปยัง DHT11/22	18
รูปที่ 2.19 MCU ส่งสัญญาณ pull up เพื่อรอการตอบสนองจาก DHT	18
รูปที่ 2.20 DHT ส่งสัญญาณ pull up เป็นการตอบสนองไปยัง MCU	19
รูปที่ 2.21 รายละเอียดของสัญญาณ pull down ที่ได้จาก DHT11	19
รูปที่ 2.22 รายละเอียดของสัญญาณ pull down ที่ได้จาก DHT22	19
รูปที่ 2.23 อุปกรณ์ตรวจวัดความชื้นและอุณหภูมิ	20
รูปที่ 2.24 Level Shifter Module 5V to 3V	20
รูปที่ 2.25 ความต้านทาน 1kΩ	20
รูปที่ 2.26 โครงสร้างภายใน DHT11/22	20
รูปที่ 2.27 Schematic Diagram ของ DHT11/22	21
รูปที่ 2.28 เปรียบเทียบความแตกต่างระหว่าง DHT11 และ DHT22	21
รูปที่ 2.29 ลักษณะของขน贲รประสังเคราะห์ชนิดโพลีโพลีลีน	22
รูปที่ 2.30 ลักษณะของขน贲รประสังเคราะห์ชนิดชนิดโพลีเอสเทอเร	22
รูปที่ 2.32 ลักษณะของขน贲รประสังเคราะห์ชนิดพีทีเอฟอี	23
รูปที่ 2.33 ลักษณะของลวดนิโกร姆	23
รูปที่ 2.34 แสดงการให้ผลของอากาศผ่านตัวพัดลมแบบหมุนเหวี่ยง	25

สารบัญรูป (ต่อ)

หน้า

รูปที่ 2.35 ลักษณะของอีตเตอร์คอยล์	26
รูปที่ 2.36 ลักษณะของเทอร์โมสตัท	26
รูปที่ 2.37 วงจรทำความร้อนของชด漉ด	26
รูปที่ 3.1 แผนผังภาพการทำงานของระบบ	28
รูปที่ 3.2 แผนผังภาพการทำงานของระบบแบบ Manual	29
รูปที่ 3.3 แผนผังภาพการทำงานของระบบแบบ Automatic	30
รูปที่ 3.4 แผนผังภาพการทำงานของระบบซักและล้างรองเท้ากีฬา	31
รูปที่ 3.5 แผนผังภาพการทำงานของระบบการทำให้แห้ง	32
รูปที่ 3.6 ภาพองค์ประกอบโดยรวมโครงสร้างเครื่องซักรองเท้ากีฬา	33
รูปที่ 3.7 ภาพลักษณะของตัวเครื่องซักรองเท้ากีฬา	34
รูปที่ 3.8 วงจรควบคุม Solenoid valve	34
รูปที่ 3.9 วงจรควบคุม Motor สำหรับหมุนรองเท้า	35
รูปที่ 3.10 วงจรควบคุมอีตเตอร์	35
รูปที่ 4.1 การทำงานของการใช้งานถ่ายภาพนิ่ง	36
รูปที่ 4.2 การทำงานในการประมวลผลภาพ	38
รูปที่ 4.3 ผลการทดลองค่าของอุณหภูมิระหว่างเซ็นเซอร์ DHT22 และ Fluke 52 K/J Thermometer	40
รูปที่ 4.4 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 40 องศาเซลเซียสในเวลา 0 – 20 นาที	42
รูปที่ 4.5 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 45 องศาเซลเซียสในเวลา 0 – 20 นาที	42
รูปที่ 4.6 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 50 องศาเซลเซียสในเวลา 0 – 20 นาที	42
รูปที่ 4.7 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 55 องศาเซลเซียสในเวลา 0 – 20 นาที	43

บทที่ 1

บทนำ

การซักรองเท้ากีฬาจะประสบความยุ่งยากในการซักและการทำให้ร่องเท้ากีฬาแห้ง เพราะส่วนมากต้องใช้มือในการซักทำความสะอาดรองเท้ากีฬา ซึ่งใช้เวลานานพอกสมควรในการซักทำความสะอาดแต่ละคู่ การซักทำความสะอาดนั้นต้องมีการใช้ผงซักฟอก เพื่อให้รองเท้ากีฬามีความสะอาดมากขึ้นซึ่งผงซักฟอกจะมีฤทธิ์เป็นเบสที่สามารถกัดกร่อนมือได้ อาจทำให้เกิดอาการแพ้ได้สำหรับบางคน วิธีแก้ส่วนใหญ่ คือการเอารองเท้ากีฬาทำความสะอาดด้วยเครื่องซักผ้า เพื่อที่จะให้สัมผัสผงซักฟอกน้อยที่สุดและประหยัดเวลาในการซักทำความสะอาด ซึ่งอาจจะส่งผลให้รองเท้ามีความเสียหายได้ เมื่อจากการกระแทกกับขอบถังซัก ในส่วนของการทำให้ร่องเท้ากีฬาแห้งส่วนใหญ่จะใช้แสงแดดจากดวงอาทิตย์ในการทำให้ร่องเท้ากีฬาแห้ง ซึ่งบางวันอาจจะไม่มีแสงแดด ทำให้รองเท้าไม่แห้งและก่อให้เกิดกลิ่นอับชื้นจากการของเท้ากีฬา ปัจจุบันยังไม่ค่อยมีเครื่องที่ใช้สำหรับซักรองเท้ากีฬาและทำให้ร่องเท้าแห้งได้

จึงมีแนวคิดที่จะสร้างเครื่องซักรองเท้ากีฬา โดยใช้การประมวลผลภาพ ซึ่งใช้คอนโทรลเลอร์ในการประมวลผล เพื่ออำนวยความสะดวกต่อผู้ใช้งานในการซักรองเท้ากีฬาและการทำให้ร่องเท้ากีฬาแห้ง

การทำโครงงานครั้งนี้เป็นการนำปัญหาการส่วนตัวและของบุคคลทั่วไปที่ใช้รองเท้ากีฬานิดผ้าที่ต้องการซักรองเท้ากีฬาที่มีความสกปรกให้สะอาดมากยิ่งขึ้น

1.1 วัตถุประสงค์

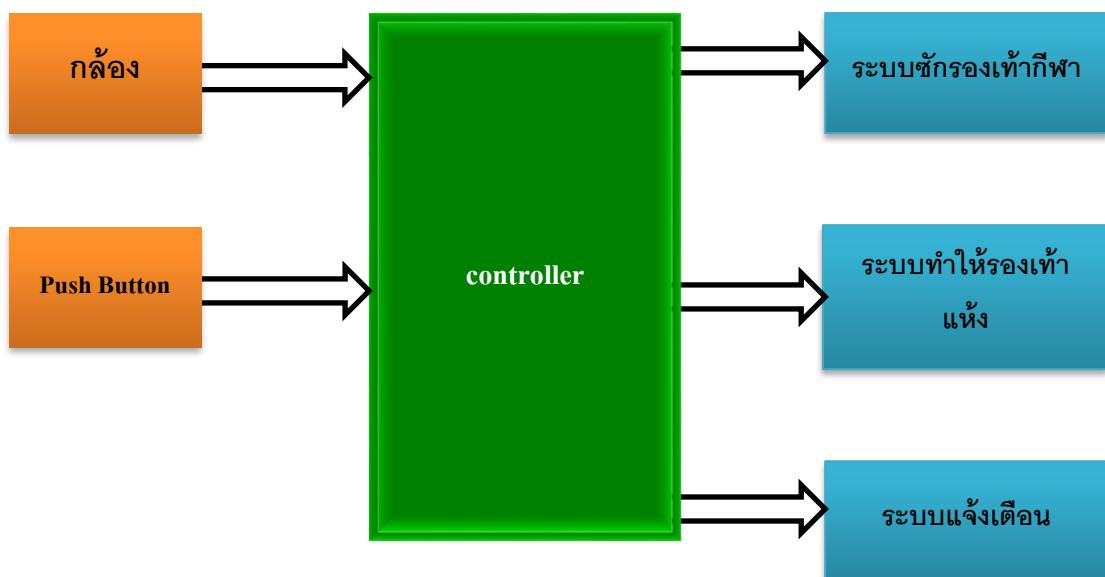
- เพื่อตอบสนองความต้องการของบุคคลทั่วไปในการซักรองเท้ากีฬา
- เพื่อศึกษาการทำงานของระบบการทำให้ร่องเท้าแห้ง
- เพื่อเป็นการสร้างเครื่องซักรองเท้า
- เพื่อลดความเสียหายของรองเท้าที่เกิดจากการซักให้น้อยที่สุด
- เพื่อแก้ปัญหาการซักรองเท้าและก็ตากองเท้าในวันที่ไม่มีแสงอาทิตย์

1.2 ขอบเขตการดำเนินงาน

- โครงงานนี้มีขนาดไม่เกิน 800x800x1000 มิลลิเมตร
- สามารถซักรองเท้ากีฬาได้ครั้งละ 1 คู่
- สามารถรองรับรองเท้ากีฬา ขนาดตั้งแต่ 36 - 42 ไซซ์ตามยูโรป
- สามารถซักและล้างรองเท้ากีฬาแบบผ้าเท่านั้น
- มีระบบทำให้ร่องเท้าแห้งด้วยระบบควบคุมอุณหภูมิของลมตั้งแต่ 40 - 55 องศาเซลเซียส มีค่าความ לחายด์ครั้งละ 5 องศาเซลเซียส และมีค่าความผิดพลาดไม่เกิน ± 2 องศาเซลเซียส ณ ขณะอ่าน
- สามารถใช้การประมวลผลภาพในการตรวจสอบและแสดงผลความสะอาดของรองเท้าที่ปะอะเบื้อนได้
- มีระบบแจ้งเตือน เช่น แจ้งเตือนเมื่อน้ำล้นถังที่เกิดจากความผิดพลาดของการทำงาน, แจ้งเตือนเมื่อระบบทำงานเสร็จเรียบร้อย
- สามารถสั่งงานเฉพาะทำให้แห้งได้ เพื่อเพิ่มเวลาในการณ์ที่ไม่แห้ง

1.3 ขั้นตอนการดำเนินการ

- ศึกษาหาข้อมูลเกี่ยวกับเครื่องซักผ้าต่างๆ
- ออกแบบเครื่องซักผ้าร่องเท้ากีฬา
- ศึกษาหาวัสดุที่เหมาะสมในการสร้างโครงสร้าง
- ศึกษาระบบทำให้ร่องเท้าแห้ง
- ทำการประกอบเขื่อมต่อระบบ
- ศึกษาการเขียนโปรแกรมในมircrocontroller
- เขียนโปรแกรมในmicrocontroller
- ทดลองการทำงานของตัวเครื่อง
- จัดทำปริญญา屁น์



รูปที่ 1.1 การทำงานของโครงงานวิศวกรรม

จากรูปที่ 1.1 จะเห็นได้ว่าการทำงานจะใช้มircrocontrollerเป็นตัวประมวลผลหลักในการรับค่าอินพุต ซึ่งตัวอินพุต คือการสั่งงานด้วยการกดปุ่มและรับค่ารูปภาพมาจากการกล้อง และนำค่าอินพุตมาประมวลผลภาพและสั่งการให้ค่าเอาต์พุตที่ได้ไปยังมีคอนิกส์เครื่องซักรองเท้ากีฬา โดยจะมีอุปกรณ์ประกอบไปด้วย แปรผูกรถลึงสำหรับซักรองเท้า 4 อัน มอเตอร์เกียร์กระ斯特รองจำนวน 1 ตัว วาล์วน้ำโซลินอยด์จำนวน 2 ตัว และเซนเซอร์ลูกloyด์น้ำล้น 3 ตัว ซึ่งค่าเอาต์พุตจะมีการการทำงานด้วยกัน 3 ระบบ คือ ระบบซักรองเท้ากีฬา ระบบทำให้ร่องเท้ากีฬาแห้ง และระบบแจ้งเตือน เช่น แจ้งเตือนเมื่อน้ำล้นถังที่เกิดจากความผิดพลาดของการทำงาน แจ้งเตือนเมื่อระบบทำงานเสร็จเรียบร้อย

1.3.1 โครงการนวัตกรรม 1

- 1.3.1.1 ศึกษาเกี่ยวกับระบบการทำงานของเครื่องซักผ้า
- 1.3.1.2 ศึกษาหาวัสดุที่เหมาะสมในการสร้างโครงสร้าง
- 1.3.1.3 ศึกษาเกี่ยวกับระบบทำให้รองเท้าแห้ง
- 1.3.1.4 ออกแบบชิ้นงานและจัดทำโครงการนวัตกรรม 1
- 1.3.1.5 ทำการทดลองระบบต่าง ๆ
- 1.3.1.6 จัดทำรายงานโครงการนวัตกรรม 1

1.3.2 โครงการนวัตกรรม 2

- 1.3.2.1 จัดทำโครงการนวัตกรรม 2
- 1.3.2.2 ออกแบบระบบไฟฟ้า
- 1.3.2.3 ทำการเชื่อมต่อระบบห้องหมวด
- 1.3.2.4 เลือนโปรแกรมไมโครคอนโทรลเลอร์และการแสดงผลทั้งหมด
- 1.3.2.5 ทำการทดลองระบบต่าง ๆ
- 1.3.2.6 จัดทำปริญญานิพนธ์

บทที่ 2

ทฤษฎีที่เกี่ยวข้อง

2.1 การประมวลผลภาพ

การประมวลผลภาพ (Image Processing) เพื่อที่จะได้ค่ามาวิเคราะห์นั้นประกอบด้วยหลายตัวแปร หรือ หลายปัจจัยด้วย โดยที่มีความสำคัญมากที่สุดก็คงหนีไม่พ้นเรื่องของสี ที่จะเป็นตัวบ่งบอกความแตกต่างระหว่างพื้นหลัง และตัววัตถุหรือรูปทรงต่าง ๆ (Contour) และอันดับต่ำมาคือหน่วยของภาพ หรือพิกเซลที่มีผลลัพธ์เข่นกัน ในเรื่อง ของความละเอียด ซึ่งยิ่งละเอียดมากก็จะสามารถแยกส่วนของข้อมูลภาพได้ถูกต้องแม่นยำยิ่งขึ้น และแสดงก็มี ความสำคัญไม่แพ้กัน ต่อให้ภาพมีความละเอียดสูงแต่ถ้าแสงน้อยเกินไปอาจจะทำให้วิเคราะห์ไม่ตรงกับความจริงเลย ดังนั้น ปัจจัยพื้นฐานที่จะได้ข้อมูลจากภาพเพื่อนำไปวิเคราะห์ได้อย่างแม่นยำนั้นได้แก่ สี ปริมาณของพิกเซล และ แสง

2.1.1 ระบบสี (Color System)

เป็นรูปแบบที่เราใช้ในการนิยาม หรือแทนค่าพื้นที่บนภาพ ซึ่งปกติแล้วจะนิยามเป็นในเชิง คณิตศาสตร์ จะประกอบไปด้วยค่าของสี 3 หรือ 4 ค่า โดยระบบสีแบ่งออกได้ดังนี้

2.1.1.1 ระบบสี RGB

ตารางที่ 2.1 ตารางรายชื่อสีและค่าที่แทนค่าสีต่าง ๆ

Color	Name	Hex Code #RRGGBB	Decimal Code (R,G,B)
Black	Black	#000000	(0,0,0)
White	White	#FFFFFF	(255,255,255)
Red	Red	#FF0000	(255,0,0)
Lime	Lime	#00FF00	(0,255,0)
Blue	Blue	#0000FF	(0,0,255)
Yellow	Yellow	#FFFF00	(255,255,0)
Cyan / Aqua	Cyan / Aqua	#00FFFF	(0,255,255)
Magenta / Fuchsia	Magenta / Fuchsia	#FF00FF	(255,0,255)
Silver	Silver	#C0C0C0	(192,192,192)
Gray	Gray	#808080	(128,128,128)
Maroon	Maroon	#800000	(128,0,0)
Olive	Olive	#808000	(128,128,0)
Green	Green	#008000	(0,128,0)
Purple	Purple	#800080	(128,0,128)
Teal	Teal	#008080	(0,128,128)
Navy	Navy	#000080	(0,0,128)

จากตารางที่ 2.1 จะประกอบไปด้วยแม่สีทั้งหมด 3 สี ได้แก่ สีแดง สีเขียว และสีน้ำเงิน โดยการผสมสีในระบบนี้เป็นลักษณะของแสง ซึ่งแสงมีลักษณะเป็นคลื่นเมื่อได้ที่แสงมา ขอนทับกัน จะทำให้เกิดการรวมตัวของความยาวคลื่นดังนั้นจะทำให้เกิดแสงสีต่าง ๆ ใช้ในการ แสดงผล ซึ่งอุปกรณ์ที่ใช้ในการแสดงผลหรือ output ที่ได้ทางจะแสดงบนอุปกรณ์ที่เป็นโทรศัพท์

คอมพิวเตอร์ หรือ smart phone ส่วน input ของการรับค่าสีในระบบ RGB เช่น กล้องวิดีโอหรือ webcam เป็นต้น ซึ่งระบบสี RGB จะใช้ในการวิเคราะห์ในระบบ โดยจะเห็นได้จากตารางเปรียบเทียบค่าสีในมาตรฐานต่าง ๆ ได้

2.1.1.2 ระบบสี RYB

ระบบสีนี้ ประกอบไปด้วยแม่สีทั้ง 3 ได้แก่ สีแดง สีเหลือง และสีน้ำเงิน เป็นมาตรฐานที่ใช้ในการผสมสีในงานศิลปะ เช่น การวาดรูป

2.1.1.3 ระบบสี CMYK

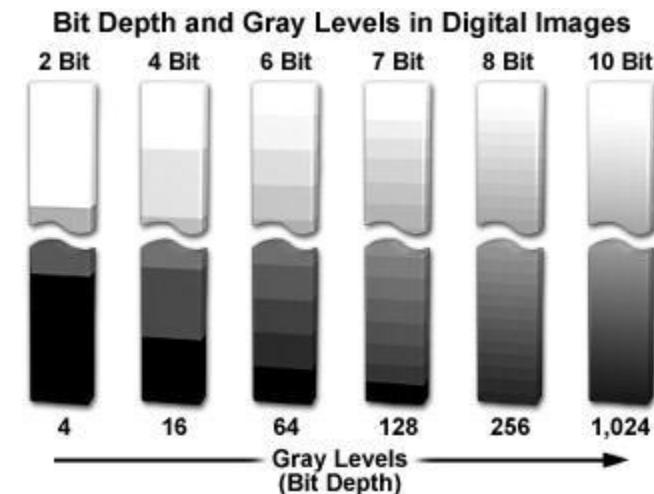
ระบบสีนี้คือการผสมสี 4 สี ได้แก่ สีน้ำเงินอมเขียว สีแดงอมม่วง สีเหลือง สีดำ ซึ่งใช้ในงานพิมพ์สี ซึ่งจะสังเกตได้จาก หมึกพิมพ์ของเครื่องปรินต์ ink jet ทั่วไปจะมี 4 สี

2.1.1.4 ระบบสี HSV และ HSL

เป็นระบบสีที่ปรับปรุงระบบสี RGB เพื่อให้ได้การแสดงผลสีของรูปทรงต่าง ๆ ได้อย่างมีคุณภาพ ใช้งานด้าน คอมพิวเตอร์กราฟิก เพื่อความสมจริงของแสงและเงา

2.1.1.5 ระบบสีเทา

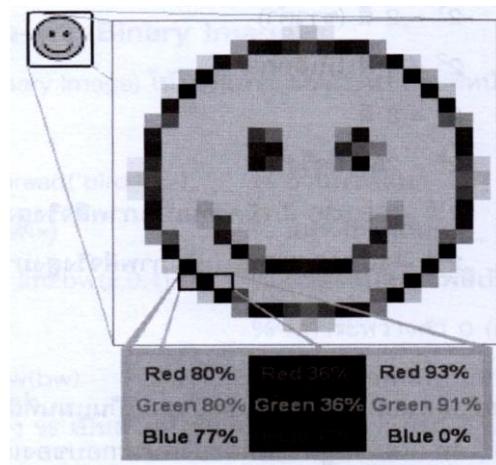
เป็นช่วงของเฉดสีเทา ซึ่งแตกต่างกับภาพขาว-ดำที่มีเพียง 2 สี คือขาวกับดำ สีในระดับสีเท่านี้ แสดงถึงความเข้มของสี ในระดับต่าง ๆ โดยสีดำเป็นส่วนที่มีความเข้มของสีน้อย และสีขาวจะมีความเข้มของสีมาก จำนวนระดับของสีขึ้นอยู่กับขนาดของบิตที่ใช้เก็บค่าสี โดยทั่วไปแล้วจะเก็บข้อมูลสีประเภทนี้ด้วยข้อมูลขนาด 8 บิต หรือ 1 ไบต์ ซึ่งจะให้ความละเอียดของสีที่ 256 เฉดสี ดังรูปที่ 2.1



รูปที่ 2.1 ระดับสีของระบบสีเทาตามขนาดข้อมูลที่เก็บค่าสี

2.1.2 พิกเซล (Pixel)

โดยคำว่า Pixel มาจากการผสมคำระหว่าง Picture กับ Element ทำให้ได้เป็นความหมายว่า องค์ประกอบของภาพ หรือหน่วยของภาพ โดยความหนาแน่นของ Pixel ในภาพนั้นจะเป็นตัวบ่งบอกความละเอียดของภาพนั้น เช่น ภาพที่มีขนาด Resolution เท่ากับ 2272*1704 จะมีค่าเท่ากับ 3,871,488 พิกเซล หรือประมาณ 4MP โดยพิกเซลเป็นหน่วยที่เล็กที่สุดของภาพและแต่ละพิกเซลมีที่อยู่ หรือตำแหน่งที่อยู่ทางจากแกน x, y หรือความกว้างและความยาวตาม resolution ของภาพนั้น แต่ละพิกเซลจะเก็บค่าของสี และเมื่อพิกเซลแต่ละพิกเซลรวมตัวกันมากพอ มันจะทำให้เกิดเป็นภาพได้ สังเกตได้จากรูปที่ 2.2



รูปที่ 2.2 ตัวอย่างแสดงจุด Pixel ภายในภาพ

บิตต่อพิกเซลเป็นการนิยามให้ในแต่ละพิกเซลแทนค่าสีได้ตามจำนวนของ เช่น จากสมการ
เปรียบเทียบในตารางด้านล่างบ่งบอกถึงปริมาณสีที่ความสามารถแทนค่าได้ในแต่ละพิกเซล โดยยิ่งมี
จำนวนบิตเทียบก็ยิ่งสามารถแทนค่าสีได้酵ะตามไปด้วยแล้วจะทำให้ภาพมีความสมจริงยิ่งขึ้น จะเห็นได้
ดังตารางที่ 2.2 ที่เปรียบเทียบค่าในขนาด bit per pixel ในขนาดที่ต่างกันออกไป

ตารางที่ 2.2 การเปรียบเทียบค่า bit per pixel

1 bpp	$2^1 = 2$ สี (ขาวดำ)
2 bpp	$2^2 = 4$ สี
3 bpp	$2^3 = 8$ สี
8 bpp	$2^8 = 256$ สี
16 bpp	$2^{16} = 65,536$ สี
24 bpp	$2^{24} = 16,800,000$ สี

2.2 ไลบรารีสำหรับใช้งานเรื่องการประมวลผลภาพ

OpenCV (Open Source Computer Vision Library) เป็นไลบรารีสำหรับใช้งานเรื่องการประมวลผลภาพ (Image Processing) และคอมพิวเตอร์ วิทัศน์ (Computer vision) ซึ่งความสามารถของ OpenCV ได้แก่ การทำภาพเบลอ การหา threshold การหา Histogram ของภาพ เป็นต้น แต่ความสามารถโดยส่วนใหญ่มักใช้ค้นหาของของภาพ การตรวจสอบการเคลื่อนไหว และการแบ่งภาพออกเป็นส่วน (Image segmentation)

นอกจากนี้ OpenCV สามารถจัดการกับข้อมูลแบบบิวติโลได้ด้วย เนื่องจาก OpenCV เป็นชุดคำสั่งที่ไม่ได้เป็นตัวโปรแกรม เมื่อต้องการเรียกใช้งานซึ่งต้องเขียนโปรแกรมเพื่อเรียกชุดคำสั่งเหล่านั้น ซึ่งภาษาที่นิยมได้แก่ ภาษา C, ภาษา C++ และภาษา Python ซึ่ง OpenCV จะประกอบด้วยสองส่วน คือ ส่วนโครงสร้างข้อมูล (Data Structure) ที่ใช้ในการเก็บข้อมูลต่าง ๆ โดยเฉพาะการประมวลทางรูปภาพ สำหรับใน OpenCV จะประกอบด้วย ไลบรารีอยู่ 4 ส่วน ได้แก่

2.2.1 ไลบรารี CXCORE เป็นพังก์ชันเบื้องต้นที่ใช้จัดการเกี่ยวกับจุด ขนาด อาร์เรย์ หน่วยความจำ คำสั่งในการวาดภาพ การประมวลตัวแปรภาพ เป็นต้น ตัวอย่างคำสั่งในการประมวลรูปภาพคือ IplImage, CvMat , CvMatND

2.2.2 ไลบรารี CV ใช้ในการประมวลผลและการวิเคราะห์รูปภาพ พังก์ชันส่วนใหญ่จะทำงานกับจุดภาพที่เป็นอาร์เรย์สองมิติ หรือที่เรียกว่าภาพนั่นเอง เช่น การหาขอบหรือมุน การทำฮิสโตแกรม (Histogram) เป็นต้น

2.2.3 ไลบรารี Machine Learning เป็นไลบรารีที่รวมคลาสและฟังก์ชันทางสถิติ (Statistical) การแยกคลาสและการแบ่งกลุ่มข้อมูล (Clustering)

2.2.4 ไลบรารี HighGUI เป็นไลบรารีที่ใช้ในการดึงภาพ การบันทึกภาพ การติดต่อกับกล้องวิดีโอ การเปลี่ยนแปลงขนาดและเคลื่อนย้ายหน้าต่าง รวมไปถึงการตรวจสอบเมาร์ และแป้นพิมพ์

2.3 ภาษาไพธอน (Python)

ภาษาไพธอน (Python) คือชื่อภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่ง ซึ่งถูกพัฒนาขึ้นมาโดยไม่มีผู้ติดกับแพลตฟอร์ม กล่าวคือสามารถรันภาษา Python ได้ทั้งบนระบบ Unix, Linux, Windows NT, Windows 2000, Windows XP หรือแม้แต่ระบบ FreeBSD อีกอย่างหนึ่งภาษาตัวนี้เป็น OpenSource เมนูออนไลน์ PHP ทำให้ทุกคนสามารถที่จะนำ Python มาพัฒนาโปรแกรมของเราได้พร้อมๆ โดยไม่ต้องเสียค่าใช้จ่าย และความเป็น Open Source ทำให้มีคนเข้ามาช่วยกันพัฒนาให้ Python มีความสามารถสูงขึ้น และใช้งานได้ครบคุมกับทุกักษณะงาน

จุดเด่นของภาษาไพธอน

ภาษาไพธอน พัฒนาโดยชาวเยอรมันชื่อนาย Guido van Rossum ซึ่งได้ออกแบบมาเพื่อให้ทำงานได้กับ Web Application ที่ลักษณะคล้ายกับภาษา Perl, PHP, JAVA และ ASP เนื่องจากภาษาไพธอน เป็นภาษาที่ใหม่จึงมีคุณสมบัติที่ดีดังต่อไปนี้

- สามารถใช้ได้ทุกแพลตฟอร์ม กล่าวคือ สามารถทำงานได้ทุกๆ CPU หลายๆ ระบบปฏิบัติการ เพียงแต่ผู้เขียนโปรแกรมเขียนจากแพลตฟอร์มใด ๆ แล้วนำโปรแกรมที่ได้ไปให้ทำงานต่างแพลตฟอร์มกันได้
- ไม่ต้องเสียค่าใช้จ่ายในการจัดซื้อโปรแกรมต้นฉบับ โดยปกติแล้วโปรแกรมภาษาทั่ว ๆ ไป จะต้องจัดซื้อโปรแกรมต้นฉบับเพื่อนำมาติดตั้งในราคาที่แพงมาก แต่โปรแกรมภาษาไพธอน สามารถดาวน์โหลดจาก www.python.org ได้โดยตรง แล้วนำมาติดตั้งและศึกษาการใช้ด้วยตนเอง เพราะเป็นโปรแกรมประเภท Open Source
- ภาษาไพธอนได้นำเอาข้อดีของโปรแกรมในอดีตเข้ามาไว้ด้วยกัน เช่น ภาษา C, C++, Java และ Perl เป็นต้น
- มีความปลอดภัยสูง เนื่องจากภาษาไพธอนทำงานอยู่ด้าน Server เป็นหลัก เมื่อมีการร้องขอจากเครื่อง Client จะประมวลผลที่เครื่อง Server ทำให้ผู้ใช้ทั่วไปไม่สามารถเข้าถึงเครื่อง Server ได้โดยตรงจึงมีความปลอดภัยสูงกว่า
- ใช้ในการพัฒนา Web Service ซึ่งในปัจจุบันการพัฒนาซอฟต์แวร์ได้เน้นที่มีการแลกเปลี่ยนข้อมูลซึ่กันและกันทั้งในองค์กรเดียวกันหรือแม้แต่ต่างองค์กรกัน ทำให้เกิดความสะดวกสบาย ไม่ต้องใช้ซอฟต์แวร์อื่น ๆ มาแปลงข้อมูลเพื่อให้เข้ากันได้อีกต่อไปเรียนรู้ได้เร็วกว่าโปรแกรมภาษาอื่น ๆ เพราะมีโครงสร้างภาษาที่ไม่ซับซ้อน ซึ่งโครงสร้างภาษาคล้ายคลึงกับภาษา C ถ้าโปรแกรมเมอร์ที่เคยใช้ภาษา C มา ก่อนจะทำให้เรียนรู้ได้เร็วขึ้น นอกจากนี้การเขียนโปรแกรมด้วยภาษาไพธอนจะมีความกระชับและสั้นกว่าภาษา C

2.4 MSE และ SSIM

2.4.1 ค่าผิดพลาดกำลังสองเฉลี่ย (MEAN SQUARE ERROR - MSE)

วิธีดังเดิมและง่ายในการวัดกำลังของสัญญาณผิดพลาดในภาพทดสอบ คำนวณความแตกต่างระหว่างความผิดพลาดของภาพต้นฉบับและภาพทดสอบ

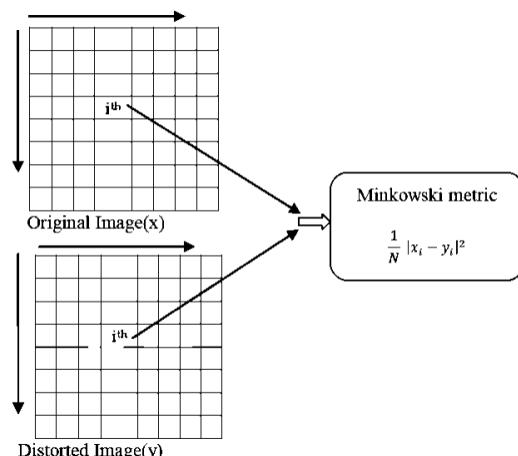
สองสัญญาณจะถูกเปรียบเทียบเป็นพิกเซลตามพิกเซลจากซ้ายไปขวาและจากบนลงล่างผ่านแถวและคอลัมน์ จากนั้นคำนวณหาค่าเฉลี่ยความแตกต่างระหว่างความผิดพลาดของภาพต้นฉบับและภาพทดสอบ ถ้า x และ y เป็นภาพสีเทาสองภาพที่ไม่เป็นค่าลบ MSE คำนวณโดยใช้สมการที่ 2.1

$$MSE = \frac{1}{N} \sum_{i=1}^N |x_i - y_i|^2 \quad (2.1)$$

ค่า Peak signal to noise ratio (PSNR) จะคำนวณดังสมการที่ 2.2

$$PSNR = 10 \log_{10} \left(\frac{m^2}{MSE} \right) \text{ db.} \quad (2.2)$$

โดยที่ m เป็นระดับสีเทาสูงสุดที่ 8 บิต / พิกเซลของภาพ ($m = 100$ ถือว่าเป็นอย่างอื่น $m = 255$ สำหรับ 8 บิต / พิกเซล)



รูปที่ 2.3 Minkowski Metric สำหรับการคำนวณ MSE

จากรูปที่ 2.3 จะเห็นได้ว่าสองภาพจะถูกเปรียบเทียบเป็นพิกเซลตามพิกเซลจากซ้ายไปขวาและจากบนลงล่างผ่านแถวและคอลัมน์ จากนั้นคำนวณหาค่าเฉลี่ยความแตกต่างระหว่างความผิดพลาดของภาพต้นฉบับและภาพทดสอบ

คุณสมบัติเด่นของ MSE

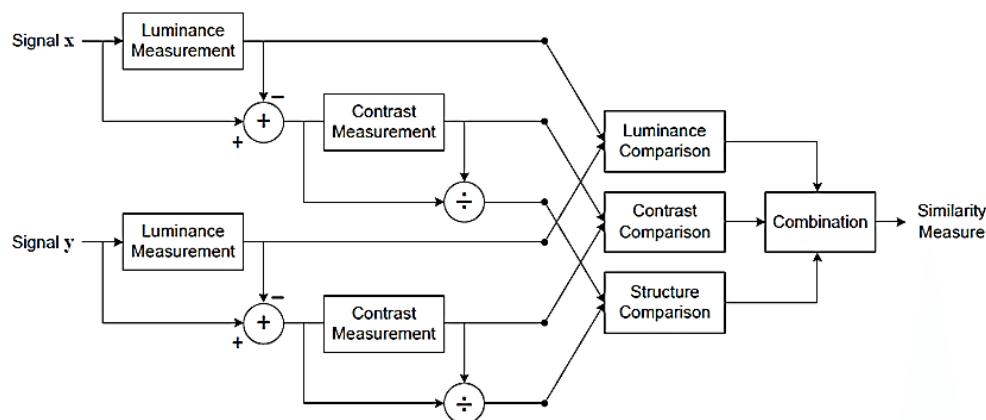
- ง่ายและคำนวณได้เร็ว
- คำนวณข้อผิดพลาดสแควร์เป็นอิสระจากกลุ่มตัวอย่างอื่น ๆ
- $MSE \geq 0$
- $MSE = 0$ ถ้าหากว่าสัญญาณเดิม (x) = สัญญาณทดสอบ (y)
- ล้ำความหมายทางภาษาภาพ

ค่า MSE มีขนาดเล็กหรือเท่ากับศูนย์ระบุค่าความผิดเพี้ยนต่ำสุดหรือศูนย์ MSE ใช้กันอย่างแพร่หลายในหลากหลายรูปแบบของการประมวลผลสัญญาณ เช่น การลดสัญญาณรบกวน การฟื้นฟูการจำแนก การฟื้นฟู และการออกแบบตัวกรอง

ข้อเสียของ MSE

- ข้อเสียเปรียบที่สำคัญของ MSE มีความสัมพันธ์กับการรับรู้ของมนุษย์ในระบบภาพ
- ตามระบบภาพ HVS (Human visual system) ของมนุษย์การแสดงข้อผิดพลาดจะสัมพันธ์กับการสูญเสียคุณภาพ แต่ MSE ไม่ตรงกันทั้งหมดนี้ เพราะการบิดเบือนบางอย่างไม่สามารถมองเห็นได้ชัดเจนและบางส่วนมืออยู่ แต่ไม่รบกวนคุณภาพของภาพ
- ภาพทั้งหมดที่มี MSE เท่ากันไม่ได้หมายความว่าทุกภาพมีการบิดเบือนหรือสูญเสียรบกวนเดียวกัน

2.4.2 รูปแบบการใช้งานร่วมกัน (Structural Similarity Index Matrix - SSIM)



รูปที่ 2.4 หลักการของรูปแบบการใช้งานร่วมกัน

จากรูปที่ 2.4 หลักการของรูปแบบการใช้งานร่วมกัน (Structural Similarity Index Matrix - SSIM) จะเห็นว่าโดยการเริ่มจากการวัดค่าความสว่างของภาพและวัดค่าความแตกต่างในแต่พิกเซลของภาพ แล้วจึงนำมาเปรียบเทียบ 3 อย่าง คือ ค่าความสว่าง ความแตกต่าง และโครงสร้างของภาพ เมื่อทำการเปรียบเทียบทั้ง 3 อย่างมาร่วมกันก็จะได้ค่า SSIM ออกมา

วิธีการใหม่เพื่อแก้ไขปัญหานี้ซึ่งจะให้แนวทางที่เป็นอิสระจากสภาพปัญหาการมองเห็น วัดคุณลักษณะเพื่อดึงข้อมูลโครงสร้างจากภาพ เมทริกซ์ด้านความคล้ายคลึงโครงสร้าง (Structural Similarity Index Matrix - SSIM) และพารามิเตอร์ 3 อย่าง เช่น ค่าความส่องสว่าง ความคมชัด และโครงสร้าง ซึ่งเป็นอิสระจากกันและมีโครงสร้างของภาพที่สูง ถ้าพิจารณาสองภาพที่ไม่ใช่เชิงลับ x และ y โดยที่ x เป็นสัญญาณที่ไม่ต่อเนื่องและ y เป็นสัญญาณผิดเพี้ยน จากสมการที่ 2.3

$$SSIM(x, y) = f[l(x, y), c(x, y), s(x, y)] \quad (2.3)$$

ความสว่าง $l(x, y)$ ดังสมการที่ 2.4

$$l(x, y) = \frac{(2\mu_x\mu_y + C_1)}{(\mu_x^2 + \mu_y^2 + C_1)} \quad (2.4)$$

โดยที่ความเข้มของสัญญาณเริ่มต้น $\mu_x = \frac{1}{N} \sum_{i=1}^N x_i$ สัญญาณที่บิดเบี้ยวหมายถึงความเข้ม $\mu_y = \frac{1}{N} \sum_{i=1}^N y_i$ และ C_1 เป็นค่าคงที่เพื่อหลีกเลี่ยงความไม่แน่นอนเมื่อ $(\mu_x^2 + \mu_y^2)$ อยู่ใกล้กับศูนย์มากและเท่ากับ $(K_1 l)^2$ l คือช่วงไดนามิกของค่าพิกเซล (255 สำหรับภาพขนาด 8 บิตต่อเทา) และ $K_1 \ll l$ ตามกฎของ Weber ขนาดของการเปลี่ยนแปลงการส่องสว่างที่เห็นได้ชัด $Δl$ จะประมาณสัดส่วนกับความ

ส่วนพื้นหลัง I สำหรับค่าความสว่างที่หลากหลาย ให้ R มีการเปลี่ยนแปลงเทียบกับความสว่างพื้นหลังเรา จะเขียนความสว่างของสัญญาณที่บิดเบี้ยวดังสมการที่ 2.5

$$\mu_y = (1 + R)\mu_x \quad (2.5)$$

สมการที่ 2.5 แทนในสมการข้างต้นสมการ 2.4 จะได้สมการที่ 2.6

$$l(x, y) = \frac{2(1+R)}{1+(1+R^2+\frac{C_1}{\mu_x^2})} \quad (2.6)$$

ถ้าสมมติว่า C_1 มีขนาดเล็กพอเมื่อเปรียบเทียบกับ μ_x^2 แล้วสมการข้างต้นเป็นเพียงฟังก์ชัน R เท่านั้นและสอดคล้องกับกฎของ weber หากความคงซัด $c(x, y)$ ได้จากสมการที่ 2.7

$$c(x, y) = \frac{(2\sigma_x\sigma_y+C_2)}{(\sigma_x^2+\sigma_y^2+C_2)} \quad (2.7)$$

โดยที่ σ_x ส่วนเบี่ยงเบนมาตรฐานเป็นค่าประมาณความคงซัดของสัญญาณโดยการลบความเข้มเฉลี่ยจากสัญญาณดังสมการที่ 2.8

$$\sigma_x = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \quad (2.8)$$

$C_2 = (K_2 l)^2$ และ $K_2 \ll I$ เป็นจำนวนน้อย มีการเปลี่ยนแปลงความคงซัดเท่ากัน $\Delta\sigma = \sigma_y - \sigma_x$ ซึ่งไม่สำคัญต่อความเปรียบต่างของฐานสูง σ_x มากกว่าค่าคงที่ของฐานต่ำพิสูจน์คุณลักษณะการปิดกั้นความคงซัดในระบบภาพของมนุษย์ (HVS) โครงสร้างคำนวนหลังจากหักค่าความสว่างและค่าความแปรปรวนของค่าความแปรปรวนแล้วเราเชื่อมโยงกับเวกเตอร์สองหน่วยที่กำหนดโดย $\frac{(x-\mu_x)}{\sigma_x}$ และ $\frac{(y-\mu_y)}{\sigma_y}$ โครงสร้างของ $s(x, y)$ คำนวนดังสมการที่ 2.9

$$s(x, y) = \frac{\sigma_{xy}+C_3}{\sigma_x\sigma_y+C_3} \quad (2.9)$$

ที่ C_3 มีค่าคงที่น้อยและสมการที่ 2.10

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y) \quad (2.10)$$

ดังนั้นผล $SSIM(x, y)$ จะได้ดังสมการที่ 2.11

$$SSIM(x, y) = \frac{(2\mu_x\mu_y+C_1)(2\sigma_x\sigma_y+C_2)}{(\mu_x^2+\mu_y^2+C_1)(\sigma_x^2+\sigma_y^2+C_2)} \quad (2.11)$$

ที่ทั้งสามพารามิเตอร์มีโครงสร้างสูงและค่อนข้างอิสระ ค่าเฉลี่ย $SSIM(MSSIM)$ ถูกคำนวนเพื่อประเมินคุณภาพภาพโดยรวมเป็นสมการที่ 2.12

$$MSSIM(x, y) = \frac{1}{M} \sum_{j=1}^M SSIM(x_j, y_j) \quad (2.12)$$

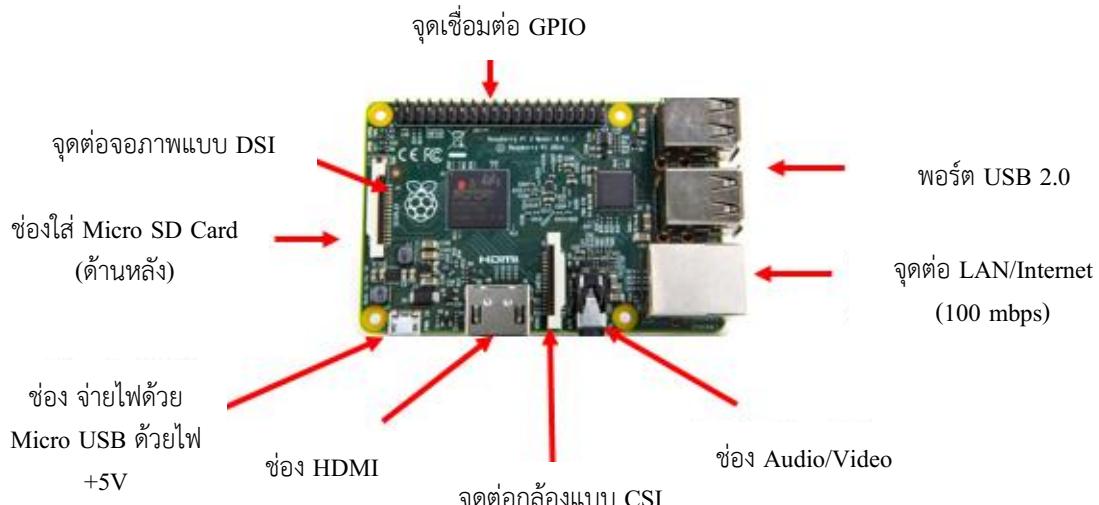
คุณสมบัติของ $SSIM$ มีดังนี้

- สมมาตร $S(x, y) = S(y, x)$
- ค่าสูงสุดที่ไม่ใช้ $S(x, y) = 1$ ถ้า $x = y$
- ขอบเขต $S(x, y) \leq 1$

2.5 ตัวควบคุม Raspberry Pi

ตัวควบคุม Raspberry Pi คือ บอร์ดคอมพิวเตอร์ขนาดเล็กที่สามารถเชื่อมต่อกับจอmonitor คีย์บอร์ด และเมาส์ได้ สามารถนำมาประยุกต์ใช้ในการทำโครงการทางด้านอิเล็กทรอนิกส์ การเขียนโปรแกรม หรือเป็นเครื่องคอมพิวเตอร์ตั้งโต๊ะขนาดเล็ก ไม่ว่าจะเป็นการทำงาน Spreadsheet Word Processing ท่องอินเทอร์เน็ต ส่งอีเมล หรือเล่นเกมส์ อีกทั้งยังสามารถเล่นไฟล์วีดีโอด้วยความละเอียดสูง (High-Definition) ได้

บอร์ด Raspberry Pi รองรับระบบปฏิบัติการลินุกซ์ (Linux Operating System) ได้หลายระบบ เช่น Raspbian (Debian) Pidora (Fedora) และ Arch Linux เป็นต้น โดยติดตั้งบน SD Card บอร์ด Raspberry Pi นี้ถูกออกแบบมาให้มี CPU GPU และ RAM อยู่ภายในชิปเดียวgan มีจุดเชื่อมต่อ GPIO ให้ผู้ใช้สามารถนำไปใช้ร่วมกับอุปกรณ์อิเล็กทรอนิกส์อื่น ๆ ดังรูปที่ 2.5



รูปที่ 2.5 ส่วนประกอบของตัวควบคุม Raspberry Pi

2.6 แผ่นอะคริลิก

อะคริลิก หรืออะคริลิเครชัน (Acrylic Resins) เป็นพอลิเมอร์ และโคโพลิเมอร์ที่เตรียมได้จากการอะคริลิก และอนุพันธ์ของกรดอะคริลิก และเอสเทอร์ของกรดอะคริลิก โดยใช้สารตั้งต้น ได้แก่ Methyl Acrylate, Ethyl Acrylate และ Methyl Methacrylate ผลิตออกมานเป็นอะคริลิกที่นิยมใช้มากคือ Polymethyl Methacrylate (PMMA)

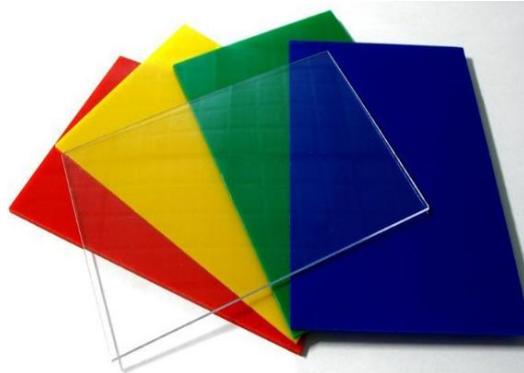
2.6.1 ประเภทของอะคริลิก

2.6.1.1 อะคริลิกของแข็ง เป็นท่อรูโมพลาสติกเรซินที่อยู่ในรูปของแข็ง อาจเป็นเม็ดอะคริลิก หรือขี้นรูปเป็นแผ่น เช่น แผ่นอะคริลิกหรือพลาสติกอะคริลิก ดังรูปที่ 2.6 ถือเป็นโอลิเมอร์ของเมทاكրิเลตอะสเตอร์ หรือโคโพลิเมอร์ของเมทاكริเลต ได้แก่

- เมทิลเมทاكริเลต
- เอทิลเมทاكริเลต
- นอร์มอล-บิวทิวเมทاكริเลต
- ไอโซบิวทิวเมทاكริเลต
- 酇ิลเมทاكริเลต
- เมทิลอะคริเลต
- นอร์มอล-บิวทิวอะคริเลต

2.6.1.2 อะคริลิกเหลว เป็นอะคริลิกที่อยู่ในรูปสารละลายที่ผลิตได้จากการกระบวนการพอลิเมอร์ไรเซชันแบบสารละลาย

2.6.1.3 อะคริลิกอีมัลชัน ใช้เป็นส่วนผสมของสีทาบ้าน มักผลิตในลักษณะของลาเท็กซ์

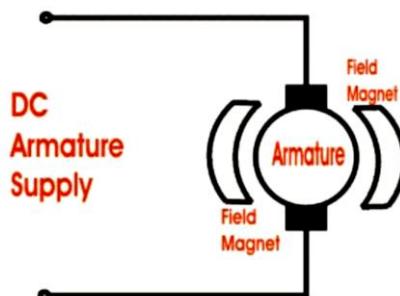


รูปที่ 2.6 ตัวอย่างแผ่นอะคริลิก

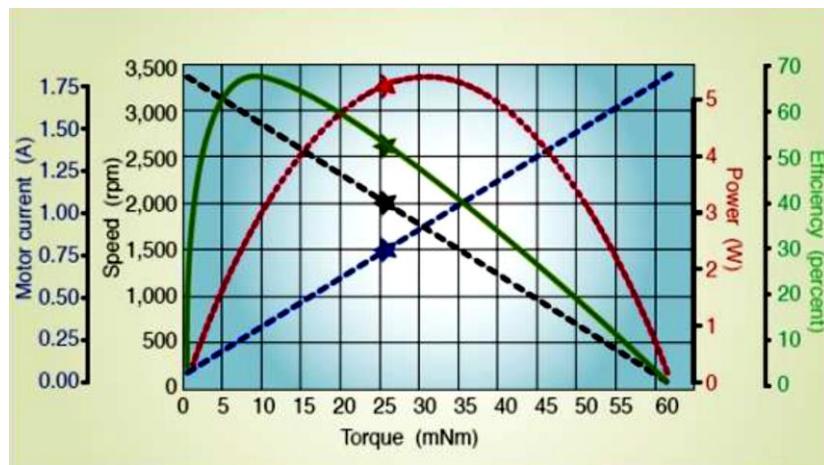
2.7 มอเตอร์กระแสตรงแบบแม่เหล็กถาวร

มอเตอร์ไฟฟ้ากระแสตรงแบบแม่เหล็กถาวร โดยส่วนที่อยู่กับที่ของมอเตอร์แบบนี้จะเป็นชื่อสามมิติ ซึ่งทางด้วยแม่เหล็กถาวรติดอยู่ที่โครงของมอเตอร์ อาจจะมี 1 คู่ หรือมากกว่าก็ได้ แม่เหล็กถาวรเป็นส่วนสำคัญสร้างขึ้นจากเซรามิก อัลนิโค หรือแร่ธาตุพิเศษอื่น ๆ ที่อยู่บนพื้นโลก และปัจจุบันได้มีการพัฒนาสร้างวัสดุที่เป็นแม่เหล็กถาวรแบบใหม่ คือ ชามาเรียมโคบล็อก และนีโอไดเมียม-เหล็ก-ไบرون ตามมอเตอร์ถูกสร้างให้อยู่ภายในโครงอันเดียวกัน มีขนาดกะทัดรัด และมีประสิทธิภาพสูง สามารถส่งส่วนที่หมุนจะเป็นคลื่นอาร์เมเจอร์พันด้วยชุดลวดทองแดงปลายด้านหนึ่งต่อเข้ากับคอมมิวเตเตอร์ หลักการทำงานเกิดจากพลาร์มของเส้นแรงแม่เหล็กของแม่เหล็กถาวรที่สเตเตอร์กับเส้นแรงแม่เหล็กของชุดลวดอาร์เมเจอร์ จึงทำให้เกิดแรงบิด และการหมุนที่ตัวอาร์เมเจอร์ขึ้น คุณลักษณะทั่วไปของแรงบิดและความเร็วรอบจะเป็นเส้นตรง แรงบิดและความเร็วรอบที่เกิดขึ้นจะถูกควบคุมโดยการปรับแรงดันของอาร์เมเจอร์ ดังแสดงในรูปที่ 2.7 – 2.8

- ข้อดีของ ขั้วแม่เหล็กถาวรที่ทำด้วยอัลนิโคคือเมื่อมอเตอร์ทำงานมีอุณหภูมิสูงจะมีปัญหาน้อยที่สุด
- ข้อเสียคือ เมื่อไม่มีการควบคุมชุดลวดสนามแม่เหล็กจึงทำให้คุณสมบัติพิเศษของความเร็วรอบกับแรงบิดจะขาดหายไป และถ้ามอเตอร์ชนิดนี้ทำงานในสภาพที่โหลดไม่เกินพิกัด ความเป็นแม่เหล็กในช่องอากาศจะยังคงตัวอยู่



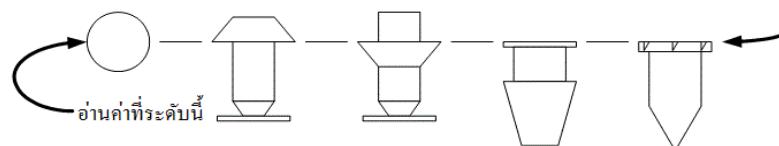
รูปที่ 2.7 วงจร มอเตอร์กระแสตรงแบบแม่เหล็กถาวร



รูปที่ 2.8 กราฟคุณลักษณะมอเตอร์กระแสตรงแบบแม่เหล็กการ

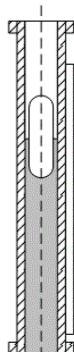
2.8 เชนเชอร์ระดับน้ำ หรือลูกกลอย (Float Switch)

ลูกกลอย (Float Switch) เป็นอุปกรณ์ใช้สำหรับการวัดระดับ (level measurement) ของเหลว อาศัยหลักการลอยตัวของลูกกลอยบนของเหลว โดยน้ำหนักของลูกกลอยที่กระทำกับแรงโน้มถ่วงโลกมีค่าเท่ากับน้ำหนักของเหลวที่มีปริมาตรเท่ากับปริมาตรของลูกกลอยส่วนที่จมอยู่ในของเหลว ลูกกลอยที่ใช้คร้มีรูปร่างและขนาดที่ออกแบบให้รับแรงลอยตัวได้มากโดยมีพื้นที่ผิวน้อย และความมีขนาดที่เหมาะสมเพื่อความไว (sensitivity) ใน การวัดนั่นคือ ส่วนที่จมอยู่ในของเหลวครั้งหนึ่งของปริมาตรทั้งหมดของลูกกลอย โดยทั่วไปรูปร่าง มาตรฐานของลูกกลอยเป็นทรงกลมหรือทรงกระบอก โดยขนาดเส้นผ่านศูนย์กลางขึ้นอยู่กับคุณลักษณะและสมบัติของเหลวที่ต้องการวัดระดับ เช่น ลูกกลอยที่มีขนาดใหญ่เหมาะสมสำหรับการวัดระดับของเหลวที่มีความหนาแน่นต่ำ และในทางกลับกันการวัดระดับของเหลวที่มีความหนาแน่นสูงควรใช้ลูกกลอยที่มีขนาดเล็ก โดยทั่วไปขนาดเส้นผ่านศูนย์กลางของลูกกลอยอยู่ในช่วง 75 mm ถึง 175 mm ตัวอย่างรูปร่างของลูกกลอยแต่ละแบบและการอ่านค่าระดับแสดงดังรูปที่ 2.9



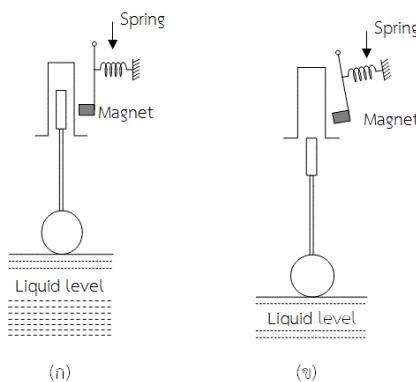
รูปที่ 2.9 การอ่านระดับของลูกกลอยแต่ละแบบ

ลูกกลอยเป็นอุปกรณ์วัดระดับที่นิยมใช้ เนื่องจากมีโครงสร้างที่ง่าย ใช้งานง่าย สามารถใช้งานภายใต้สภาพ อุณหภูมิ (temperature) และความดันสูง (pressure) ได้ สะดวกต่อการปรับเทียบ (calibration) และมีความเที่ยงตรง (precision) สูง สามารถวัดระดับได้ทั้งแบบจุดและแบบต่อเนื่องโดยขึ้นอยู่กับลักษณะการอุปกรณ์ และการเลือกใช้ อุปกรณ์เพิ่มเติม ตัวอย่างการวัดระดับด้วยลูกกลอยอย่างง่ายที่สุดแสดงดังรูปที่ 2.10 เป็นการวัดระดับโดยตรง ผู้วัดสามารถอ่านค่าระดับของเหลวได้โดยตรงจากตำแหน่งของลูกกลอยที่เปลี่ยนแปลงตามระดับความสูงของเหลว ซึ่ง เป็นการวัดเพื่อติดตามกระบวนการและการปฏิบัติงาน (monitoring processes and operations)



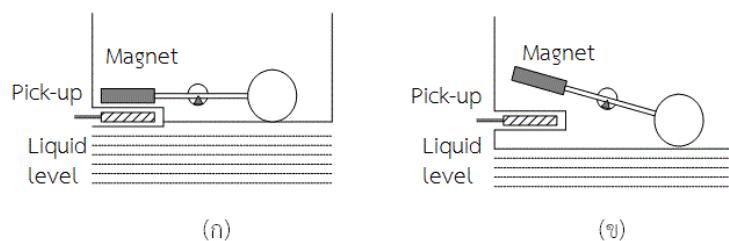
รูปที่ 2.10 การวัดระดับชนิดกลาโวย

สำหรับการวัดระดับโดยมีวัตถุประสงค์เพื่อการควบคุม (control processes and operations) ทำได้โดยการติดตั้งทรานสิเดวเซอร์ (transducer) หรืออุปกรณ์อื่นเพิ่ม เพื่อนำสัญญาณไฟฟ้าทางด้านເອົາພຸດທີ່ໄດ້ອ່ານເກັບເຄື່ອງควบคุม การติดตั้งเครื่องมือวัดระดับสามารถติดตั้งได้ทั้งด้านบนหรือด้านข้างของภาชนะขึ้นอยู่กับลักษณะการใช้งาน ตัวอย่างเช่น การใช้ถุงกลอยวัดระดับแบบจุดเพื่อควบคุมระดับน้ำภายในถัง ทำได้โดยการติดตั้งถุงกลอยทางด้านบน ของภาชนะ และทำงานร่วมกับสวิตช์ตัดต่อ แสดงดังรูปที่ 2.11 (ก และ ข) โดยการยืดหรือหดตัวของสปริงแสดงถึง การเปลี่ยนแปลงสถานะทางวงจรไฟฟ้า ซึ่งสามารถนำหลักการดังกล่าวไปใช้ควบคุมการทำงานของปั๊มน้ำได้



รูปที่ 2.11 การวัดระดับแบบจุดด้วยกล้องที่ติดตั้งด้านบนภายนอก

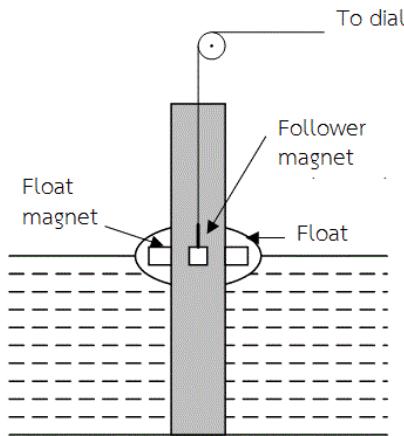
สำหรับการวัดระดับแบบจุดทำได้โดยการติดตั้งลูกกลอยบริเวณด้านข้างภาชนะ ดังรูปที่ 2.12 มีลักษณะการทำงานเช่นเดียวกับตัวอย่างในรูปที่ 2.11



รูปที่ 2.12 การวัดระดับแบบจุดด้วยลูกกลอยที่ติดตั้งด้านข้างภาชนะ

นอกจากนี้อุปกรณ์วัดระดับชนิดลูกloyยังสามารถใช้วัดระดับของเหลวแบบต่อเนื่องได้เช่นกัน โดยการติดตั้งอุปกรณ์ติดตามการเคลื่อนที่ของลูกloyซึ่งอาจใช้หลักทางกล เช่น การติดตั้งเข็มชี้วัด หรือการติดตั้งลูกloy

ร่วมกับทรานส์ดิวเซอร์ (transducer) วัดระดับและการเคลื่อนที่ เช่น การใช้ไฟฟ้าติดอยู่ (potentiometer) โดยการต่อ ก้านของลูกloy เข้ากับไวเปอร์ (wiper) ของไฟฟ้าติดอยู่ หรือการอาศัยสมบัติแม่เหล็กเป็นแรงดึงดูด ดังรูปที่ 2.13 เมื่อลูกloy แม่เหล็กเคลื่อนที่ขึ้น/ลง แรงดึงดูดของแม่เหล็กจะดึงให้แม่เหล็กที่ติดกับรอกเคลื่อนที่ตามลูกloy ด้วย โดยการเคลื่อนที่ของลูกloy เปลี่ยนแปลงตามระดับความสูงของของเหลว อย่างไรก็ตาม ลูกloy ที่ใช้สมบัติสารแม่เหล็กไม่เหมาะสมสำหรับการทำงานภายใต้อุณหภูมิ (temperature) สูง เนื่องจากสารแม่เหล็กเสื่อมสภาพเร็วทำให้มีอายุการใช้งานสั้น



รูปที่ 2.13 เครื่องมือวัดระดับของเหลวชนิดลูกloy และแม่เหล็ก

2.9 โซลินอยด์วาล์ว (Solenoid Valve)

โซลินอยด์วาล์ว (Solenoid Valve) คือ วาล์วที่ทำงานด้วยไฟฟ้ามีทั้งชนิด 2/2, 3/2, 4/2, 5/2 และ 5/3 ในบทความนี้จะได้กล่าวถึงเฉพาะวาล์วชนิด 2/2 ซึ่งใช้ควบคุมการเปิดปิดของเหลวและก๊าซเท่านั้น ส่วนวาล์วชนิด 3/2, 4/2, 5/2 และ 5/3 ซึ่งส่วนใหญ่ใช้กับระบบนิวแมติกและระบบไฮดรอลิก

เมื่อกล่าวถึงชนิดของวาล์วเป็นตัวเลขเช่น 2/2, 4/2 หรือ 5/2 นั้น ตัวเลขหน้าบวกถึงจำนวนทางเข้าออกของวาล์วนั้น ๆ ว่ามีกี่ทางหรือมีกี่รู (port) ส่วนตัวเลขที่ตามหลังเครื่องหมายทับ (/) นั้นบวกถึงจำนวนสถานะหรือจำนวนตำแหน่ง (position) ของวาล์ว เช่น วาล์ว 2/2 ก็คือวาล์วที่มี 2 ทาง และมี 2 สถานะ คือปิดและเปิด ส่วน วาล์ว 5/2 ก็คือวาล์วที่มี 5 ทาง และมี 2 สถานะ เป็นต้น

2.9.1 การทำงานของโซลินอยด์วาล์ว 2/2

2.9.1.1 ระบบเปิดปิดโดยตรง (Direct Acting หรือ Direct Operated)

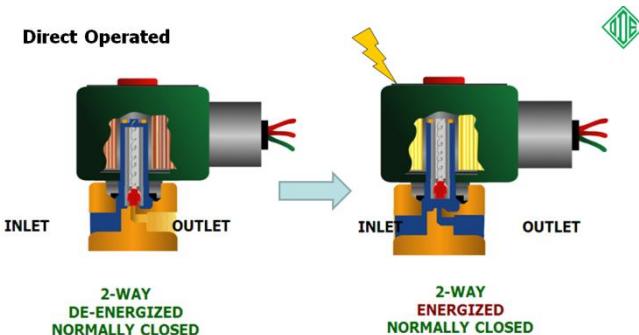
ระบบเปิดปิดโดยตรงโซลินอยด์วาล์ว 2 ทางแบบปกติปิด (N/C) ที่มีระบบการทำงานแบบเปิดปิดโดยตรงนั้นมีทางเข้าหนึ่งทางและทางออกหนึ่งทางทั้งสอง ชั้น มีชีลอยู่ปลายด้านล่างทำหน้าที่เปิดและปิดรูทางผ่าน (orifice) ของของเหลวเมื่อจ่ายไฟฟ้าเข้าหรือตัดไฟฟ้าออกจากคอยล์ดังรูปที่ 2.14

ข้อควรระวังในการใช้วาล์วที่ทำงานด้วยระบบบันช์ คือ เมื่อมีการเพิ่มความดัน (pressure) ของของเหลวในระบบจะทำให้ต้องใช้แรงมากขึ้นในการเปิดวาล์ว หากความดันของของเหลวสูงกว่าที่กำลังของคอยล์จะเปิดวาล์วได้ วาล์วนั้นก็จะไม่ทำงานถึงแม้จะมีการจ่ายไฟฟ้าแล้วก็ตาม

หลักการ - วาล์วเปิด-ปิดโดยอาศัยแรงจาก Coil และสปริงเพียงอย่างเดียว

ข้อดี - ไม่จำเป็นต้องอาศัยความดันของของเหลวในการช่วยเปิด-ปิด

ข้อจำกัด - มักจะใช้กับวาล์วที่มีขนาดไม่ใหญ่นักส่วนมากจะอยู่ที่ขนาด 1/8"-1/4"



รูปที่ 2.14 ระบบเปิดปิดโดยตรง

2.9.1.2 ระบบเปิดปิดทางอ้อม (Indirect Acting หรือ Pilot Operated)

ระบบเปิดปิดทางอ้อม (pilot control) โซลินอยด์瓦ล์ว 2 ทางแบบปกติปิด (N/C) ที่มีระบบการทำงานแบบเปิดปิดทางอ้อมนั้น มีทางเข้าหนึ่งทาง และ ทางออกหนึ่งทาง รูทางผ่านหลัก (main orifice) ซึ่งอยู่ในตัววาล์วนั้นเปิดได้ด้วยวิธีการทำให้ความดันที่กระทำต่อพื้นผิวด้านบน และด้านล่างของแผ่นไดอะแฟร์ม (diaphragm) เกิดการเสียสมดุล ดังรูปที่ 2.15

ในขณะที่ยังไม่มีไฟฟ้าจ่ายไปยังคอยล์ของไฟ จะมีความดันสูงไปทั้งในช่องบนซึ่งมีพื้นที่ผิวเต็มพื้นที่ของแผ่นไดอะแฟร์ม และในขณะเดียวกันก็มีความดันสูงไปที่พื้นผิวด้านล่าง แต่สูงไปเฉพาะพื้นที่ผิวรอบ ๆ รูทางผ่านเท่านั้น

ซึ่งเป็นพื้นที่ที่น้อยกว่าด้านบน เมื่อต้องการให้วาล์วเปิด โดยการป้อนไฟเข้าที่คอยล์ทุน (plunger) ของโซลินอยด์วาล์วตัวช่วยจะยกเปิดและระบายน้ำของไฟลูซึ่งอยู่ด้านบนของไดอะแฟร์มที่ทางรู (orifice) ย่อของโซลินอยด์วาล์วตัวช่วยยังส่งผลให้เกิดการเสียสมดุลของแผ่นไดอะแฟร์ม เกิดการเคลื่อนที่เปิดรูทางผ่านหลักให้ของไฟลูไหลผ่านไปได้

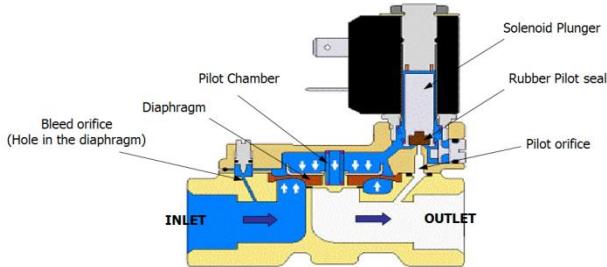
ข้อควรระวังในการใช้วาล์วที่ทำงานด้วยระบบบันห์ คือ ความดันของขาเข้าและขาออกจำต้องมีความแตกต่างกันในค่าหนึ่งตามที่กำหนดของผู้ผลิต (minimum differential pressure) เพื่อทำให้วาล์วทำงานอย่างถูกต้อง จะเห็นได้ว่าวาล์วที่ทำงานในระบบเปิดปิดทางอ้อมนี้ก็ต้องอาศัยตัวโซลินอยด์วาล์วที่ทำงานด้วยระบบเปิดปิดโดยตรงมาเป็นตัวช่วยเพื่อให้ทำงาน ดังนั้น จึงต้องคำนึงถึงความดันสูงสุดและกำลังของคอยล์ที่ใช้เปิด มีฉะนั้นวาล์วอาจไม่ทำงานถึงแม้ว่าจะมีการจ่ายไฟฟ้าแล้วก็ตาม และเพื่อให้วาล์วระบบบันห์ทำงานได้อย่างถูกต้องและหลีกเลี่ยงการสึกหรอย่างรวดเร็วของแผ่นไดอะแฟร์ม

ควรออกแบบการใช้งานโดยคำนึงถึงค่า K_v (อัตราการไหลผ่านวาล์วที่ความดันต่างศักย์ 1 bar) ของตอนที่วาล์วจะปิดว่ามีอัตราการไหลในขณะนั้นไม่เกินค่า K_v ด้วยเหตุผลดังกล่าวหากความดันของขาเข้าในขณะที่วาล์วปิดอยู่สูงกว่า 1 bar ต้องไม่ปล่อยให้ของไฟลูไหลออกทางขาออกโดยอิสระ (free outlet) จะต้องมีการจำกัดอัตราการไหลของขาออกเพื่อรักษาให้ความต่างศักย์ของความดันขาเข้าและขาออกไม่เกิน 1 bar มีฉะนั้นแผ่นไดอะแฟร์มจะเกิดการกระแทกกับปากทางผ่านหลักอย่างรุนแรงเมื่อปิดวาล์ว ทำให้แผ่นไดอะแฟร์มสึกหรอและเสียหายอย่างรวดเร็ว

หลักการ - วาล์วเปิด-ปิดโดยอาศัยหลักการความต่างของความดัน กล่าวคือ มีการจ่ายไฟเข้าคอยล์เพื่อให้เกิดการ Pilot ของของเหลวที่อยู่ด้านบนของแผ่นไดอะแฟร์ม ซึ่งจะทำให้เกิดความแตกต่างระหว่างความดันด้านบนและด้านล่าง

- ไดอะแฟร์มกับความดันของของเหลวที่เหลือข้ามมา จึงทำให้แผ่นไดอะแฟร์มยกขึ้นซึ่งจะทำให้เกิดการเปิด-ปิดของวาล์ว
- ข้อดี**
- โครงสร้างแบบนี้จะใช้กับวาล์วที่มีขนาด $3/8"$ ขึ้นไป โดยขณะที่คุณลักษณะเด่นที่สำคัญคือไม่จำเป็นต้องมีขนาดใหญ่ (เพราะคุณลักษณะเด่นที่เพียงแค่เปิดปิด) จึงทำให้ราคาถูกและเป็นที่นิยมใช้
 - ข้อจำกัด** - เนื่องจากต้องอาศัยความดันของของเหลวในการช่วยเปิด-ปิด ดังนั้นจึงไม่สามารถนำไปใช้กับงานที่มีความต่างของความดันต่ำได้

Pilot Operated



รูปที่ 2.15 ระบบเปิดปิดทางอ้อม

2.9.1.3 ระบบลูกผสม (Combined Acting หรือ Combine Operated)

โฉลกนอยด์วาล์ว 2 ทางชนิดปิดติด (N/C) ที่มีระบบการทำงานแบบลูกผสมนั้นมีทางเข้าหนึ่งทางและทางออกหนึ่งทาง การเปิดปิดผ่านหลัก (orifice) ซึ่งอยู่ภายในตัววาล์วนั้นเป็นการผสมผสานทั้งการทำให้ความดันของพื้นที่ด้านบนและด้านล่างของแผ่นไดอะแฟร์มเสียสมดุล บวกกับแรงที่ทุ่น (plunger) ของโฉลกนอยด์ตัวช่วยออกแรงยกแผ่นไดอะแฟร์มโดยตรงด้วยการทำงานหลัก ๆ ของแผ่นไดอะแฟร์มก็เหมือนกับระบบเปิดปิดทางอ้อมจะต่างก็ตรงที่ว่าแม้จะมีความดันขาเข้าเพียงน้อยนิดวาล์วก็สามารถเปิดได้ด้วยแรงยกของทุ่น (plunger) ดังรูปที่ 2.16

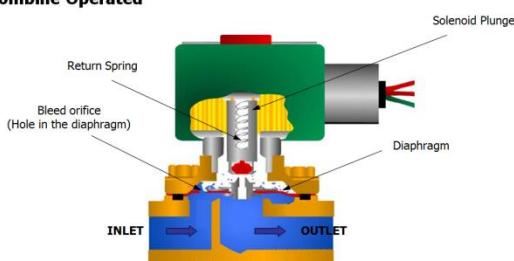
ข้อควรระวังในการใช้วาล์วนี้คือ นอกจากข้อยกเว้นที่วาล์วนี้ไม่จำเป็นต้องมีความต่างศักย์ของความดันระหว่างขาเข้าและขาออกก็เปิดปิดได้แล้ว ข้อควรระวังอีก 1 คือ เมื่อมีน้ำท่วมในห้องที่ต้องการจะต้องทำการซ่อมทุกประการ

หลักการ - วาล์วเปิด-ปิดโดยอาศัยแรงจากหัว Coil และ Mechanic ภายใน

ข้อดี - ใช้กับวาล์วที่มีขนาด $3/8"$ ขึ้นไปและของเหลวมีความต่างของความดันต่ำ ๆ

ข้อจำกัด - ราคากลางๆ กว่า Pilot Operated เพราะขนาดของ Coil จะต้องใหญ่กว่า

Combine Operated



รูปที่ 2.16 ระบบลูกผสม

2.10 เชนเซอร์ตรวจอุณหภูมิและความชื้น

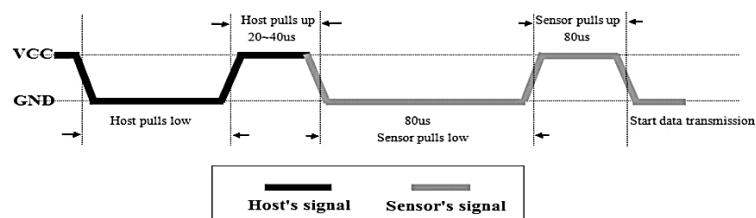
อุณหภูมิ คือปริมาณความร้อน ถ้าอากาศร้อน ปริมาณความร้อนก็จะมาก แต่ถ้าอากาศเย็น ปริมาณความร้อนก็จะน้อย

ความชื้น คือปริมาณไอน้ำที่มีอยู่ในอากาศบริเวณใดบริเวณหนึ่ง ซึ่งมีสัดส่วนที่แตกต่างกันไปในแต่ละท้องที่ ถ้าอากาศมีความชื้นต่ำ น้ำก็จะระเหยได้มาก แต่ถ้าอากาศมีความชื้นสูง น้ำก็จะระเหยได้น้อย โดยความชื้นนั้นมีหลายประเภท ได้แก่ ความชื้นสัมบูรณ์, ความชื้นจำเพาะ และความชื้นสัมพัทธ์ แต่เนื่องจาก DHT 11/22 Relative Humidity and Temperature Sensor เป็นชนเซอร์ที่สามารถวัดได้แค่ ความชื้นสัมพัทธ์ ดังนั้นความชื้นที่เราระรู้จักต่อไปคือ “ความชื้นสัมพัทธ์” ความชื้นสัมพัทธ์ คืออัตราส่วนของปริมาณไอน้ำในอากาศต่อปริมาณไอน้ำที่ทำให้อากาศอิ่มตัว(อากาศอิ่มตัว คืออากาศที่มีไอน้ำอยู่เต็มที่และไม่สามารถรับเพิ่มได้อีกแล้ว ณ อุณหภูมินั้น) ดังรูปที่ 2.17

$$\text{ความชื้นสัมพัทธ์} = \left(\frac{\text{ปริมาณไอน้ำที่อยู่ในอากาศ}}{\text{ปริมาณไอน้ำที่ทำให้อากาศอิ่มตัว}} \right) \times 100\%$$

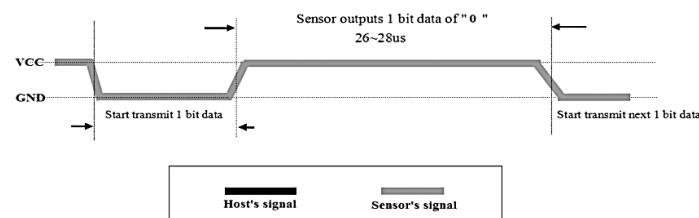
รูปที่ 2.17 วิธีคำนวณหาค่าความชื้นสัมพัทธ์

หลักการทำงาน

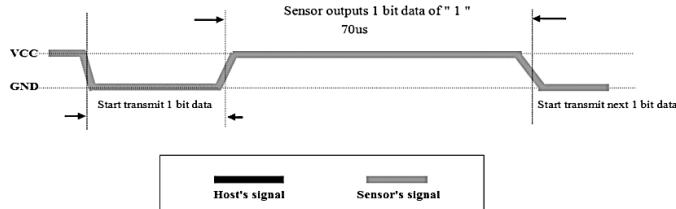


รูปที่ 2.18 การส่งสัญญาณ pull down voltage ไปยัง DHT11/22

จากรูปที่ 2.18 รีบจาก MCU จะส่งสัญญาณ pull down voltage ไปยัง DHT11/22 โดย ถ้าเป็น DHT 11 จะใช้เวลาส่ง down voltage อย่างต่ำ 18 ms แต่ถ้าเป็น DHT22 จะใช้เวลาอย่างต่ำ 1 ms และ MCU จะ pull up voltage เพื่อรอการตอบสนองจาก DHT ประมาณ 20-40 us ดังรูปที่ 2.19 หลังจากนั้น DHT จะส่งสัญญาณ pull down voltage 80 us เป็นการตอบสนองไปยัง MCU ดังรูปที่ 2.20 และ DHT ก็จะ pull up voltage เพื่อเตรียมส่งข้อมูล โดยในการส่งข้อมูลแต่ละบิต DHT จะมีการ pull down voltage 50 us



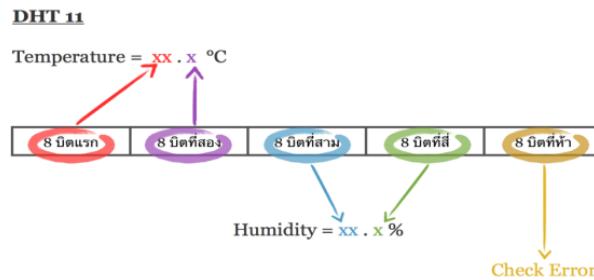
รูปที่ 2.19 MCU ส่งสัญญาณ pull up เพื่อรอการตอบสนองจาก DHT



รูปที่ 2.20 DHT ส่งสัญญาณ pull up เป็นการตอบสนองไปยัง MCU

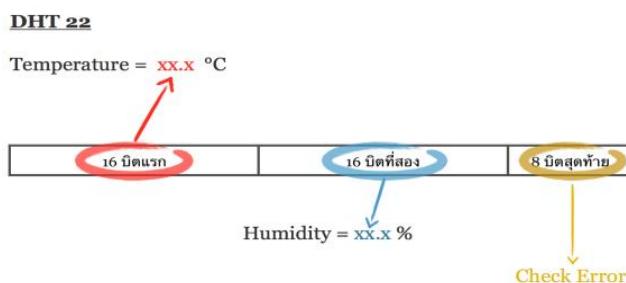
หลังจาก DHT มีการ pull down voltage 50 us เพื่อเป็นการบอก MCU ว่าจะส่งข้อมูล 1 บิต โดยการส่งบิตค่า “0” DHT จะทำการส่งสัญญาณ pull up voltage 26-28 us และ ส่งบิตค่า “1” DHT จะทำการส่งสัญญาณ pull up voltage 70 us

โดยการส่งข้อมูลของ DHT11 คือ จะส่งทั้งหมด 40 บิต โดยจะแบ่งเป็น 5 ส่วน ส่วนละ 8 บิต ซึ่ง 8 บิตแรกจะเป็นค่าหน้าทศนิยมของอุณหภูมิ, 8บิตที่สองเป็นค่าหลังทศนิยมของอุณหภูมิ, 8 บิตที่สาม จะเป็นค่าหน้าทศนิยมของความชื้น, 8 บิตที่สี่เป็นค่าหลังทศนิยมของความชื้น และ 8 บิตสุดท้ายคือเป็นค่าที่ตรวจสอบว่าข้อมูล error ดังรูปที่ 2.21



รูปที่ 2.21 รายละเอียดของสัญญาณ pull down ที่ได้จาก DHT11

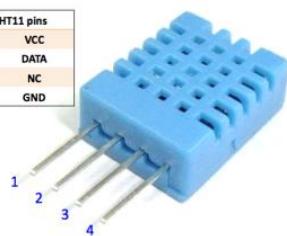
การส่งข้อมูลของ DHT22 คือ จะส่งทั้งหมด 40 บิต โดยจะแบ่งเป็น 3 ส่วน ส่วนแรกส่วนละ 16 บิต และส่วนสุดท้าย 8 บิต ซึ่ง 16 บิตแรกและ 16 บิตที่สอง หมายถึงค่าอุณหภูมิและค่าความชื้น ตามลำดับ ที่รวมทั้งค่าหน้าและหลังทศนิยม โดย ตัวเลขหลักหน่วยจะหมายถึงตัวหลังทศนิยม และ 8 บิตสุดท้ายคือเป็นค่าที่ตรวจสอบว่าข้อมูล error ดังรูปที่ 2.22



รูปที่ 2.22 รายละเอียดของสัญญาณ pull down ที่ได้จาก DHT22

อุปกรณ์ในการต่อ DHT11 และ DHT22 เป็นไปดังรูปที่ 2.23 – 2.25

DHT11 pins	
1	VCC
2	DATA
3	NC
4	GND



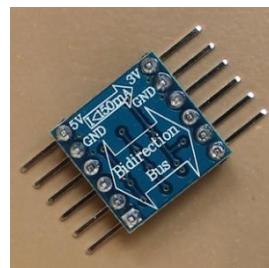
(ก) DHT 11

DHT22 pins	
1	VCC
2	DATA
3	NC
4	GND



(ก) DHT 22

รูปที่ 2.23 อุปกรณ์ตรวจความชื้นและอุณหภูมิ



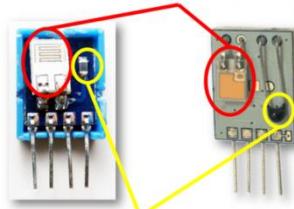
รูปที่ 2.24 Level Shifter Module 5V to 3V



รูปที่ 2.25 ความต้านทาน 1kΩ

โครงสร้าง DHT11/22 ดังรูปที่ 2.26

Resistive Humidity Sensing Component



NTC Temperature Sensor Thermistor

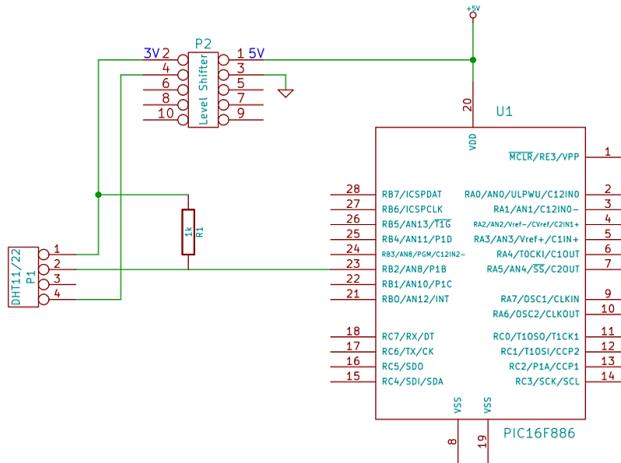
รูปที่ 2.26 โครงสร้าง DHT11/22

- Resistive Humidity Sensing Component : เชนเซอร์ความชื้นที่จะวัดการเปลี่ยนแปลงอัมพีเดนซ์ไฟฟ้าของตัวกลางดูดความชื้น การทำงานของเซนเซอร์คือดูดซับไอน้ำและไอออนที่แตกตัวเป็นผลให้ค่าความนำไฟฟ้าของตัวกลางเพิ่มขึ้น โดยช่วงเวลาการตอบสนองของเซนเซอร์อยู่ในช่วง 10 ถึง 30 วินาที

- NTC Temperature Sensor Thermistor : เป็นเซนเซอร์ที่ความต้านทานลดลงเมื่ออุณหภูมิเพิ่มขึ้น แต่มีการเปลี่ยนแปลงความต้านทานสูงมาก ตัวอย่างเช่น ที่อุณหภูมิ 0 °C NTC มีความต้านทาน

10k Ω แต่ที่อุณหภูมิ 100 °C NTC จะมีความต้านทานลดลงเหลือเพียง 200 Ω เท่านั้น ด้วยความไวต่อการเปลี่ยนแปลงมาก เทอร์มิสเตอร์แบบนี้จึงเหมาะสมกับงานที่ต้องการวัดความแตกต่างของอุณหภูมิที่ชัดเจน แต่เทอร์มิสเตอร์มีคุณสมบัติไม่เป็นเชิงเส้น ดังนั้นช่วงอุณหภูมิที่ใช้งานจึงจำกัดอยู่ในช่วงแคบ ๆ เป็นช่วง ๆ ไป เช่น ช่วง 50-150 °C หรือ 150-250 °C เป็นต้น

Schematic Diagram ของวงจร DHT11/22



รูปที่ 2.27 Schematic Diagram ของ DHT11/22

จากรูปที่ 2.27 จะเห็นว่า มีการให้ Level Shifter Module 5V to 3V เนื่องจาก DHT11/22 ใช้กับไฟ 5V ไม่ได้ ต้องไปใช้กับ 3 V แทน

เปรียบเทียบความแตกต่างระหว่าง DHT11 และ DHT22 ดังรูปที่ 2.28

ความแตกต่าง	DHT11	DHT22
ขอบเขตและความคลื่อนในการวัดความชื้นสัมพath์	ความชื้น : 20 - 80 % ความคลาดเคลื่อน : 5 %	ความชื้น : 0 - 100 % ความคลาดเคลื่อน : 5 %
ขอบเขตและความคลื่อนในการวัดอุณหภูมิ	อุณหภูมิ : 0 - 50 °C ความคลาดเคลื่อน : ± 2 °C	อุณหภูมิ : -40 - 125 °C ความคลาดเคลื่อน : ± 0.5 °C
ความกว้างใน การประมวลผล	1 Hz (1 ครั้งต่อวินาที)	0.5 Hz (2 ครั้งต่อวินาที)
ขนาด	กว้าง x ยาว x สูง : 5.5 mm x 12 mm x 15.5 mm	กว้าง x ยาว x สูง : 7.7 mm x 15.1 mm x 25 mm
ราคา	ประมาณ 100 บาท	ประมาณ 200 บาท

รูปที่ 2.28 เปรียบเทียบความแตกต่างระหว่าง DHT11 และ DHT22

2.11 งานแปรรูปสังเคราะห์

ชนในล่อน มีคุณสมบัติ ที่ดีในการ ป้องกันการขูดขีด มีความแข็งแรงสูง ยึดหยุ่นและป้องกันสารเคมีได้เป็นอย่างดี. ทำให้เกิดความเมื่อยล้าขึ้นที่ขันแปรงได้ง่ายเนื่องจากต้องรับน้ำหนัก ที่มากขึ้น. ชนในล่อนมีคุณสมบัติที่ดีมาก ในการทนการเสียรูปที่เกิดขึ้นจากความร้อน, ชนในล่อนมีแนวโน้มที่ สามารถใช้งานขณะที่มีความร้อนสูงได้ดี ดังรูปที่ 2.29

คุณลักษณะเฉพาะเด่นของขันในล่อน คือ มีทั้งขนาด และ ชนหายักษ์ สามารถเลือกขนาดขันแปรรูป หรือสี ได้มีตั้งแต่ขนาดเล็ก 0.08 - จนถึง 2.3 มิลลิเมตรขันในล่อน 6 : ขันแปรรูปคุณภาพ ราคาไม่แพง สามารถใช้ได้กับหลายลักษณะงาน แบ่งลักษณะการใช้งานตามความต้องการได้ 3 ประเภท อาทิ

ขันในล่อน 6.6 : ขันแปรรูปคุณภาพสูง สามารถใช้ได้ทั้งลักษณะงานที่เป็นยีก หรือ แห้ง มีความยึดหยุ่นตัวสูง ป้องกันการกดกร่อนได้ดี ทนความร้อนได้ดีกว่าในล่อนเกรด 6

ขันในล่อน 6.12 รุ่นชนิดพิเศษ : คุณภาพดีมาก มีความยึดหยุ่นตัว ดีมากกว่าเมื่อเทียบกับในล่อน 6.6 ป้องกันการกดกร่อน ไม่อุ้มน้ำสามารถใช้งานกับสภาพแวดล้อมและแห้งได้เป็นอย่างดี ในกรณี ที่ต้องใช้งานเปยกันนี้เป็นส่วนที่ดีที่สุดสำหรับในล่อน

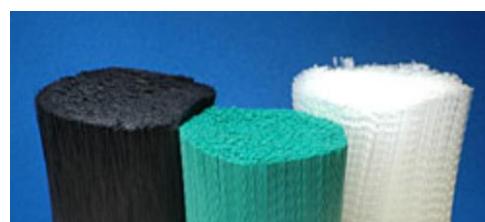
ขันในล่อนที่ป้องกันไฟฟ้าสถิตย์ : เป็นขันที่ออกแบบมาพิเศษเพื่อลดหรือป้องกันไฟฟ้าสถิตย์ ขันในล่อนที่ป้องกันไฟฟ้าสถิตย์ ออกแบบมาสำหรับหลากหลายลักษณะงาน สามารถป้องกันไฟฟ้าสถิตย์หรือลดไฟฟ้าสถิตย์ได้ทันที



รูปที่ 2.29 ลักษณะของขันแปรรูปสังเคราะห์

- ขันโพลีโพลีลีน

ขันโพลีโพลีลีน (PP) ดังรูปที่ 2.30 ขันแปรรูปชนิดนี้เหมาะสมสำหรับงานหลากหลายแบบ สามารถใช้งานในสภาพแวดล้อมที่ต้องการได้เป็นอย่างดี ใช้กับของเหลว เช่น น้ำมัน สารทำละลาย เคมี และเป็นชนิดที่สามารถทนต่อกรดและด่างได้เป็นอย่างดี มีความยึดหยุ่นตัวสูง เมื่อเทียบกับขันแปรรูปชนิดอื่น ๆ มีให้เลือกทั้งที่เป็นขันตรงหรือขันหยัก และมีขนาดเส้นผ่าศูนย์กลางตั้งแต่ 0.006" ถึง 0.060" สามารถระบุแบบหรือขนาดที่ใหญ่ขึ้นได้



รูปที่ 2.30 ลักษณะของขันแปรรูปสังเคราะห์ชนิดโพลีโพลีลีน

- ขันโพลีเอสเตอร์

ขันโพลีเอสเตอร์ ดังรูปที่ 2.31 ออกแบบมาเพื่อแทนที่ขันแปรรูปในล่อน 6.6 และ 6.12 เนื่องจากมีราคาถูกกว่า สามารถใช้งานได้หลากหลาย ถูกพัฒนาให้มีความสามารถในการทนต่อการชืดชิดมากกว่า ขันพีพี แต่น้อยกว่าในล่อน มีความสามารถในการยึดหยุ่นตัวได้ดี ทนต่อสารทำละลาย ป้องกันการเกิดออกซิเดชันที่อุณหภูมิสูง ๆ ป้องกันแสงแดดขณะใช้งานกลางแจ้งได้ดี ความสามารถในการอุ้มน้ำต่ำ สามารถใช้กับงานน้ำจีมากกว่าในล่อน มีให้เลือกทั้งที่เป็น ขันตรงหรือขันหยัก และมีขนาดเส้นผ่าศูนย์กลางตั้งแต่ 0.006" ถึง 0.075"



รูปที่ 2.31 ลักษณะของขันแปรสังเคราะห์ชนิดขันโพลีเอสเทอร์

- ขันพีทีเอฟอี

ขันพีทีเอฟอี (Teflon) : เป็นวัสดุเดียวกันกับเทฟล่อน ใช้สำหรับงานที่มีความร้อนสูง ๆ สามารถใช้งานได้ต่อกับสภาพแวดล้อมและด่างมาก ๆ มีความสามารถในการด้านทนความร้อน ต่อเนื่องถึง 500 องศาฟาเรนไฮต์ ขันเทฟล่อน มีขนาดเส้นผ่าศูนย์กลาง $0.08"$, $0.011"$, $0.016"$ และ $0.035"$ ดังรูปที่ 2.32



รูปที่ 2.32 ลักษณะของขันแปรสังเคราะห์ชนิดพีทีเอฟอี

2.12 ลวดนิโคราม



รูปที่ 2.33 ลักษณะของลวดนิโคราม

จากรูปที่ 2.33 คือลวดทนความร้อน ลวดนิโคราม คือ ลวดที่มีความต้านทานสูง เมื่อป้อนแรงดันไฟฟ้าเข้าไปทำให้เกิดร้อนแดง เปลี่ยนพลางงานไฟฟ้าให้เป็นพลางงานความร้อน มีหลายแบบหลายชนิดแบ่งตามลักษณะลวด เช่น แบบกลม แบบแบน และอุณหภูมิการใช้งาน มี yanai ให้เลือกใช้ 100 องศา ถึงประมาณ 1400 องศา หรือแบ่งตามส่วนผสมของลวด เช่นเหล็ก นิกелиโครเมียม ส่วนใช้ในการผลิต ฮิตเตอร์ให้ความร้อน ชุดลวดความร้อน ลวด ni800.6 ลวดนิโคราม nichrome wire ลวดฮีตเตอร์ ni80 nichrome80 ni80wire

ลวดนิโคราม nichrome wire ลวดฮีตเตอร์ ni80 nichrome80 ni80wire แบ่งตามส่วนผสม

- ลวดฮีตเตอร์ที่มีส่วนผสม เหล็ก เป็นส่วนผสมหลักของ FeCrAl มีคุณสมบัติทนอุณหภูมิสูง ได้ถึงประมาณ 1400 องศาเซลเซียล ไม่ทนต่อความชื้นและสารเคมีมากนัก มีความยืดหยุ่นตัวต่ำ หากไม่ใช้ความร้อนข้าช่วยจะทำให้แตกหักง่าย(กรณีลวดขนาดใหญ่)ทำให้ตัดขึ้นรูปได้ค่อนข้าง

ยก เกิดสภาวะสนามแม่เหล็กเมื่อมีการจ่ายแรงดันไฟฟ้าเข้าไป การเก็บรักษาครัวไว้ในที่แห้ง และต้องห้ามมิดชิด มิฉะนั้นอาจเกิดสนิมได้ ตัวอย่างการประยุกต์ใช้งาน เตาเผาอุตสาหกรรม ฮีตเตอร์แบบต่าง ๆ เป็นต้น

- 乩อดี้ตเตอร์ที่มีส่วนผสม นิกели เป็นส่วนผสมหลักของ NiCr มีคุณสมบัติทนอุณหภูมิได้สูง คือ ไม่เกิน 1300 องศาเซลเซียล ขึ้นอยู่กับปริมาณ สารนิกелиที่ผสม อาจมีถึง Ni 80% ทนทานต่อ ความชื้นและสารเคมี มีความยืดหยุ่นสูงทำให้ตัดขึ้นรูปได้ง่ายไม่เกิดสภาวะสนามแม่เหล็กเมื่อมี การจ่ายแรงดันไฟฟ้าเข้าไป การเก็บรักษาครัวไว้ในที่แห้งและต้องห้ามมิดชิด มิฉะนั้นอาจเกิด สนิมได้ ขาด漉ดความร้อน 乩อดni800.6 ตัวอย่างการประยุกต์ใช้งาน เช่น เตาเผาที่สารเคมี ขาด漉ดสปริงขนาดใหญ่ เป็นต้น

乩อนิโครม nichrome wire 乩อดี้ตเตอร์ ni80 nichrome80 ni80wire แบ่งตามลักษณะภายนอก

- 乩อดี้ตเตอร์ลักษณะ แบบ Ribbon heater คือ乩อดี้ตเตอร์ที่มีลักษณะ แบบ การใช้งานทั่วไป จะเป็น 乩อดซีลิง 乩อดตัดวงพลาสติก หรือแปรรูปเป็นฮีตเตอร์ความร้อนประเภทอื่น ขนาดมัก กำหนดจากหน้าตัดยาวคือ(กว้าง x หนา) หน่วยเป็นมิลลิเมตร มีทั้งแบบเหล็กและนิกелиให้ เลือกใช้งาน อาจเป็นแบบเรียบหรือมีหลายขนาดให้เลือกใช้งาน เช่น 2x0.14, 3x0.2, 4x0.2, 6x0.15, 10x0.2 หรือขนาดอื่นที่นอกจากระบุก็ได้ การเก็บรักษาครัวไว้ในที่แห้งและต้อง ห้ามมิดชิด มิฉะนั้นอาจเกิดสนิมได้

- 乩อดี้ตเตอร์ลักษณะ nichrome wire กลม Round heater คือ乩อดี้ตเตอร์ที่มีลักษณะ กลม การใช้งานทั่วไปจะเป็น 乩อดตัดพลาสติก คอยล์สปริง หรือแปรรูปเป็นฮีตเตอร์ความร้อน ประเภทอื่น ขนาดมักกำหนดจาก เส้นผ่าศูนย์กลาง หน่วยเป็นมิลลิเมตร มีทั้งแบบเหล็กและนิ กелиให้เลือกใช้งานมีหลายขนาดให้เลือกใช้งาน เช่น ตั้งแต่ 0.2 มิลลิเมตร - 10 มิลลิเมตร หรือ ขนาดอื่นที่นอกจากระบุก็ได้ การเก็บรักษาครัวไว้ในที่แห้งและต้องห้ามมิดชิด มิฉะนั้นอาจเกิด สนิมได้ ขาด漉ดความร้อน 乩อดni800.6

หลักการทำงาน漉ดความร้อน

nichrome wire หลักการทำงาน漉ดความร้อน คือการเปลี่ยน พลังงานไฟฟ้า เป็นพลังงานความร้อน เมื่อมีการการป้อนแรงดันไฟฟ้าผ่าน乩อดความร้อน 乩อดความร้อนซึ่งเป็น乩อดที่มีความ ต้านสูงจะชักดูดแรงดันไฟฟ้าใน乩อดความร้อน,ทำให้เกิดแรงเสียดทานและสามารถให้ความร้อนได้ ปริมาณ ความร้อนขึ้นอยู่กับ แรงดันไฟฟ้า และความต้านทานภายใน乩อดความร้อน

การประยุกต์ใช้งาน

การประยุกต์ใช้งาน乩อดความร้อน สามารถประยุกต์การใช้งานได้หลายรูปแบบ เช่น การนำ乩อด 乩อดไปใช้งานโดยตรง เครื่องตัดโฟม เครื่องซีล อาจนำไปตัดของให้เป็นรูปทรงที่ต้องการหรือแบบการยึดตรง ๆ ก็ได้ อีกแบบคือ การนำไปพันเป็น乩อดความร้อนหรือ "ขาดสปริง" คือการนำ乩อดขนาดต่าง ๆ พันเป็น สปริงเพื่อให้ได้ความต้านทานที่สูงขึ้นและความร้อนที่มากขึ้นด้วย เช่น 乩อดความร้อนในเตาเผา 乩อด สำหรับดัดوصلคิลิค 乩อดที่ใช้ในเตาไฟฟ้า 乩อดที่ใช้เป็นโหลดความต้านทาน หรือ乩อดที่นำไปผลิตเป็นฮีต เตอร์ทำความสะอาดร้อนแบบต่าง ๆ ทั้งนี้ขึ้นอยู่กับความต้องการในการใช้งานเป็นหลักด้วย

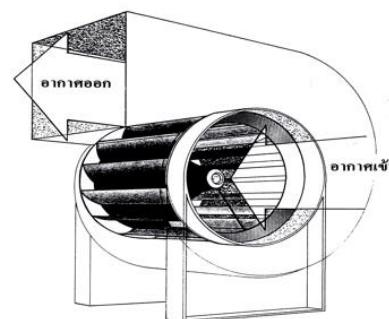
2.13 พัดลมแบบหมุนแรงเหวี่ยง (Centrifugal flow or radial fans)

พัดลมแบบแรงเหวี่ยงหรือพัดลมซึ่งมีการไหลของอากาศในแนวรัศมี ดังรูปที่ 2.34 จะประกอบด้วยใบพัดหมุนอยู่ภายในตัวเรือนของพัดลม (Fan house) ชุดใบพัดจะประกอบด้วยแผ่นใบเล็ก ๆ ประกอบเข้าด้วยกันเป็นลักษณะกล้อง ความดันของอากาศจะถูกทำให้มีค่าสูงขึ้นภายในตัวเรือนของพัดลม ซึ่งสามารถเพิ่มค่าให้สูงขึ้นได้ด้วยการเพิ่มขนาดความยาวของใบพัด ซึ่งจะทำให้แรงเหวี่ยงหนึ่งศูนย์กลางภายในระบบมีค่ามากขึ้น อากาศจะไหลผ่านเข้าไปในท่อทางเข้าโดยมีทิศทางวนกับแกนของใบพัด และไหลออกในทิศทางตั้งฉากกับแกนของเพลาใบพัดในท่อทางออก พัดลมประเภทนี้จำแนกตามลักษณะรูปร่างของใบพัดเป็น 3 แบบ คือ

2.13.1 แบบใบพัดตรง (Straight blade หรือ Radial fans) พัดลมชนิดนี้มีจำนวนใบน้อยที่สุด ประมาณ 6 ถึง 20 ใบ และใบพัดจะอยู่ในระนาบราศมีจากเพลา ใบพัดหมุนด้วยความเร็วรอบอย่างต่อเนื่อง ประมาณ 500-3000 รอบ/นาที ดังนั้นจึงเหมาะสมกับงานที่ต้องการปริมาณการไหลน้อย ๆ และมีค่าความดันของอากาศสูง ๆ

2.13.2 แบบใบพัดโค้งไปข้างหน้า (Forward curved blade fans) พัดลมชนิดนี้จะมีใบพัดโค้งไปข้างหน้า ในทิศทางเดียวกับการหมุนชุดใบพัดจะมีจำนวนแผ่นใบพัดประมาณ 20 – 60 ใบ ชุดใบพัดจะมีลักษณะคล้ายกับกรงกระรอก (Squirrel cage) เพลาใบพัดจะมีขนาดเล็กหมุนด้วยความเร็วรอบที่สูงกว่าพัดลมชนิดใบพัดตรง การทำงานของพัดลมชนิดนี้มีเสียงเบาที่สุด มีข้อเสียคือมีโอกาสที่มอเตอร์จะทำงานเกินกำลังและมีช่วงการทำงานของพัดลมที่ไม่เสถียร ดังนั้นจึงมีควรใช้กับงานหรือระบบที่มีอัตราการไหลของอากาศเปลี่ยนแปลงตลอดเวลา พัดลมชนิดนี้จะให้ค่าความดันลมและอัตราการไหลของอากาศสูงที่สุด

2.13.3 แบบใบพัดโค้งไปข้างหลัง (Backward curved blade fans) พัดลมชนิดนี้จะมีใบพัดเอียงไปข้างหลัง ในทิศทางตรงกันข้ามกับทิศทางการหมุนของใบพัด จะมีจำนวนใบพัดประมาณ 10 – 50 ใบ และเป็นพัดลมที่มีความเร็วรอบสูง ไม่ก่อให้เกิดเสียงดังเกินครัว ไม่มีลักษณะที่มอเตอร์จะทำงานเกินกำลัง และไม่มีช่วงการทำงานที่ไม่เสถียร เหมาะที่จะใช้งานระยะไกลอากาศและอากาศที่ใช้ต้องสะอาดด้วยเนื่องจากสามารถที่จะควบคุมความดันและปริมาณลมได้ง่าย พัดลมชนิดนี้จะมีราคาสูงกว่าชนิดอื่น ๆ เมื่อเทียบขนาดเท่ากัน

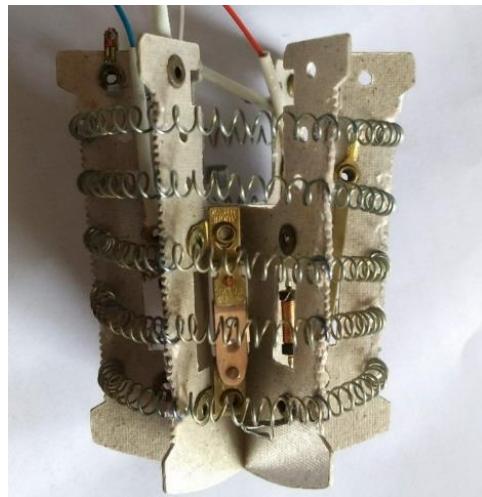


รูปที่ 2.34 แสดงการไหลของอากาศผ่านตัวพัดลมแบบหมุนเหวี่ยง

2.14 ระบบลมร้อน

ส่วนประกอบความร้อนของระบบลมร้อนเป็นกรอบที่ทำจากวัสดุที่ไม่ติดไฟกับชุดวงหารอยเส้นที่ทำจากลวด nichrome (เกลี่ย) ที่พันไว้กับแผ่นไม้ไผ่ ก้า เรียกว่า อีตตเตอร์คอยล์ ดังรูปที่ 2.35 ชุดวงหารอยของส่วนประกอบระบบลมร้อนต้องถูกเปิดด้วยลมเย็นโดยเฉพาะอย่างยิ่งเมื่อทำงานที่กำลังไฟสูงสุด หากมีเหตุผลการไหลของอากาศไม่เพียงพอ (ตัวอย่างเช่นมอเตอร์ผิดพลาด) เพื่อความปลอดภัยและเพื่อหลีกเลี่ยงความเสียหายของอุปกรณ์ทำความ

ร้อน ควรจะปิดโดยอัตโนมัติ การปิดเครื่องซุกเฉินนี้เกิดขึ้นเนื่องจากการมีอุปกรณ์เสริมที่มีความร้อนสูงซึ่งจะมีการป้องกันอยู่ภายในเครื่องทำความร้อน



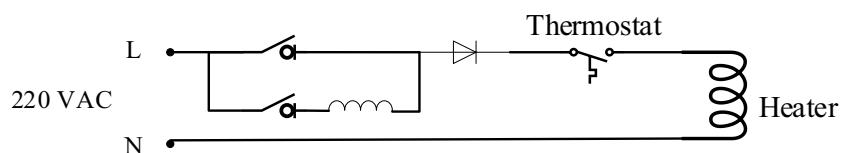
รูปที่ 2.35 ลักษณะของฮีตเตอร์คอยล์

จากรูปที่ 2.35 จะเห็นได้ว่าในฮีตเตอร์คอยล์จะมีเทอร์โมสตัท ดังรูปที่ 2.36 จะติดตั้งอยู่ภายในตัวทำความร้อนอยู่ใกล้กับทางออกของอากาศร้อนที่ติดต่อจะถูกเปิดด้วยลมเย็น เมื่ออากาศออกซึ่งสัมผัสจะแยกจากกันและกันโดยอุปทานของระบบลมร้อนจะเปิดออก หลังจากผ่านไปสองสามนาทีเมื่อแผ่น bimetallic ระบายความร้อนจะปิดลงอีครั้งและระบบลมร้อนจะเปิดขึ้นอีครั้ง



รูปที่ 2.36 ลักษณะของเทอร์โมสตัท

วงจรทำความร้อนของชุดลาด



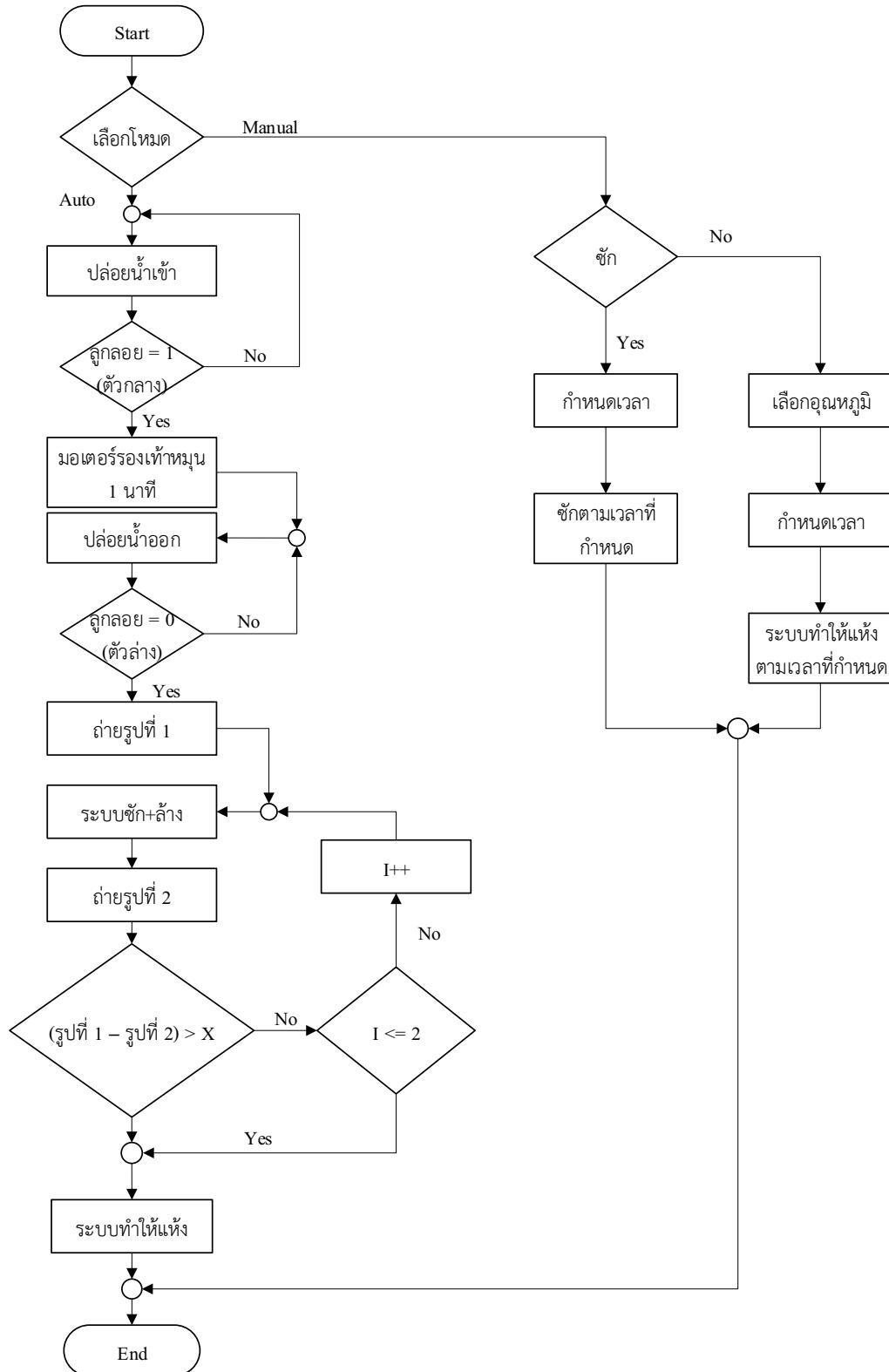
รูปที่ 2.37 วงจรทำความร้อนของชุดลาด

จากรูปที่ 2.37 ภายในวงจรทำความร้อนของชุดລາດจะมีສວິຕ່ຽງທີ່ໃຊ້ສໍາຫຼັບເລືອກຮະດັບຄວາມຮອນ ໄດ້ໂດຍ
ປັບກັນໄມ້ໃຫ້ກະແສ້ໄລຍ້ອນກລັບແລະເຫຼວຮົມສັກທີ່ໃຊ້ຕ້າງຈາກເມື່ອຂຸດລາດມີຄວາມຮອນສູງຈົນເກີນໄປ ໄພທີ່ໃຫ້ໃນງຈຣົກ້ອ
ໄພ 220 VAC

บทที่ 3

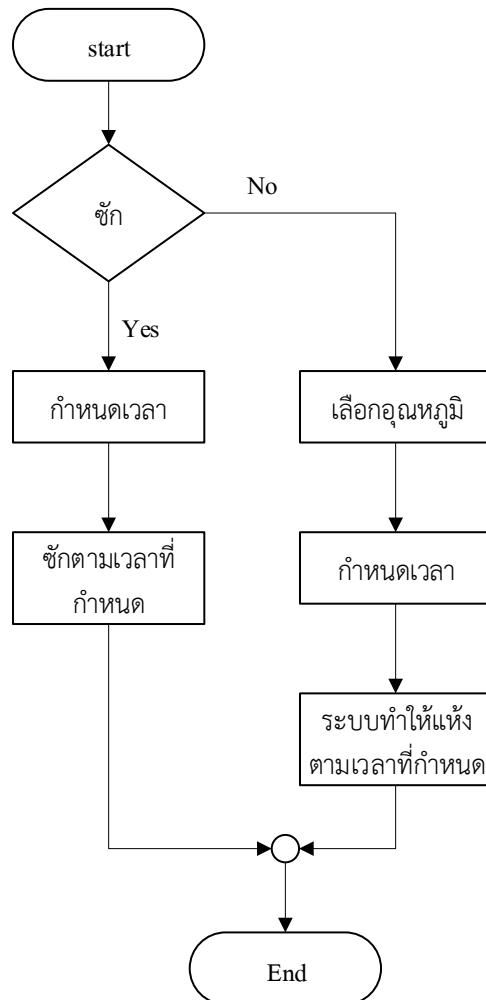
การออกรูปแบบ

3.1 การออกรูปแบบระบบการทำงาน



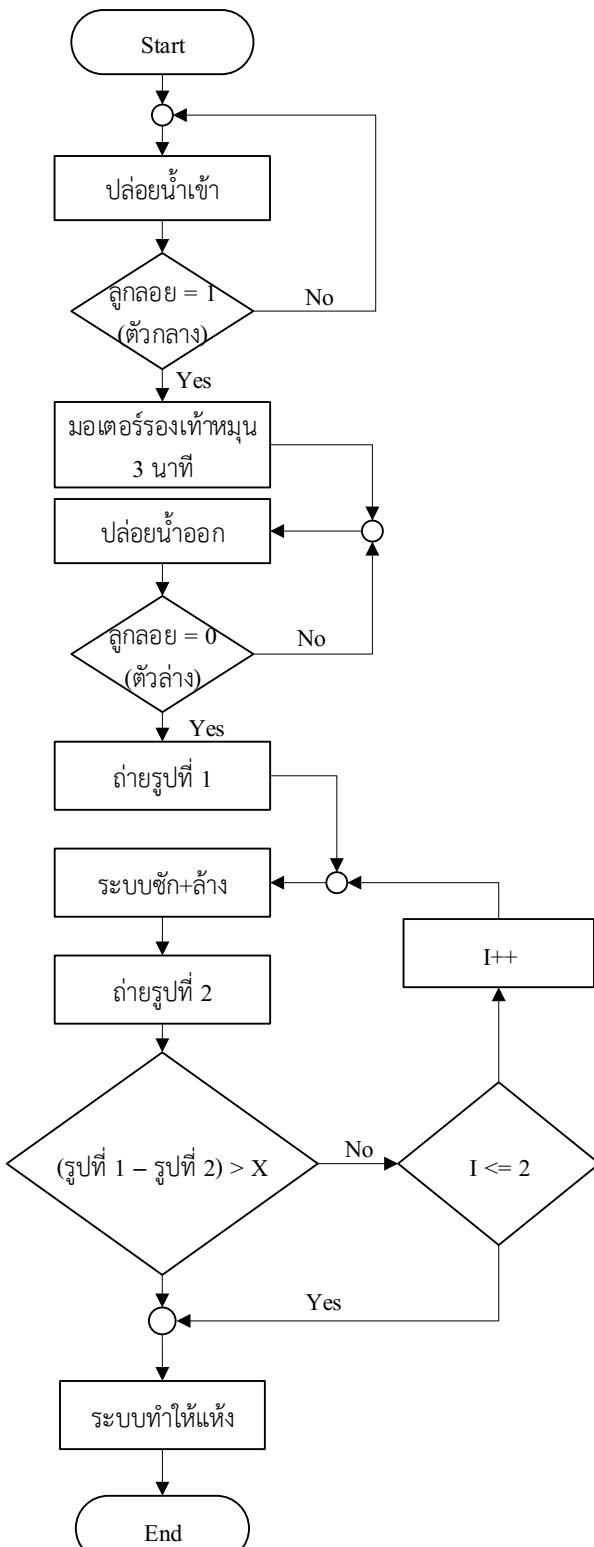
รูปที่ 3.1 แผนผังภารกิจการทำงานของระบบ

จากรูปที่ 3.1 อธิบายระบบของเครื่องซักรองเท้ากีฬา ซึ่งในเครื่องซักรองเท้ากีฬาจะระบบในการทำงาน 2 แบบ คือ แบบอัตโนมัติและแบบควบคุมด้วยมือ ซึ่งการทำงานในระบบแบบอัตโนมัติ จะประกอบไปด้วย การถ่ายรูป ระบบซักและระบบทำให้แห้ง โดยควบคุมผ่านบอร์ด Raspberry pi และการทำงานในระบบแบบควบคุมด้วยมือ จะเลือกระหว่างการซักแบบกำหนดเวลาในการทำงานหรือการทำให้แห้งแบบกำหนดเวลาในการทำงาน ซึ่งถ้าเลือกการทำให้แห้ง จะต้องเลือกอุณหภูมิที่จะใช้ในการทำให้แห้งก่อน ถึงจะสามารถกำหนดเวลาในการทำงานได้



รูปที่ 3.2 แผนผังภาพการทำงานของระบบแบบ Manual

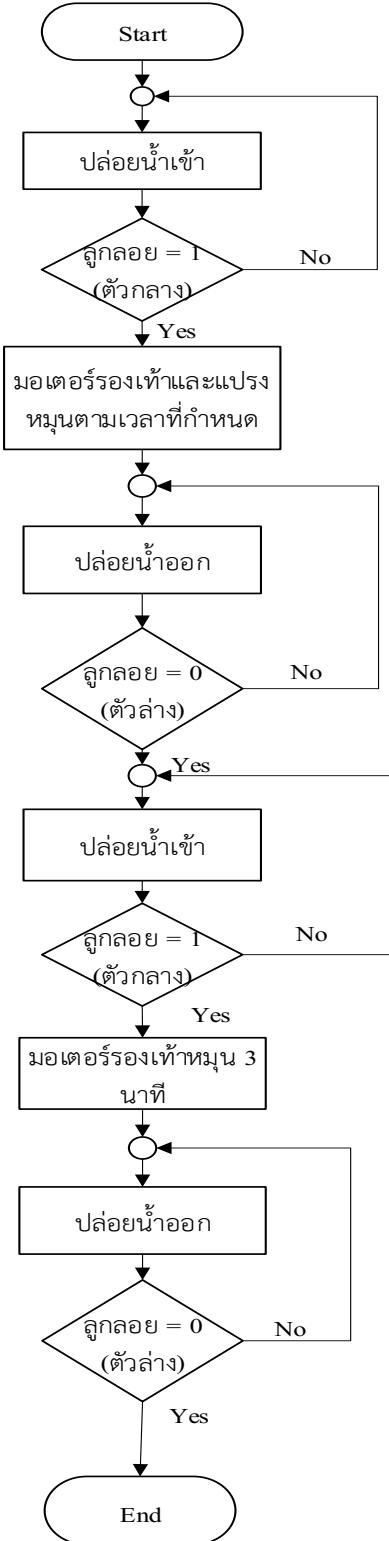
จากรูปที่ 3.2 อธิบายระบบของเครื่องซักรองเท้ากีฬาแบบควบคุมด้วยมือ จะเลือกระหว่างระบบซักแบบกำหนดเวลาหรือการทำให้แห้งแบบกำหนดเวลา ถ้าเลือกระบบการทำให้แห้งจะต้องเลือกอุณหภูมิที่จะใช้ในการทำให้แห้งตั้งแต่อุณหภูมิ 40 – 55 องศาเซลเซียส โดยมีค่าความลับเอียดครั้งละ 5 องศาเซลเซียส ถึงจะสามารถกำหนดเวลาในการทำงานได้



รูปที่ 3.3 แผนผังภารกิจการทำงานของระบบแบบ Automatic

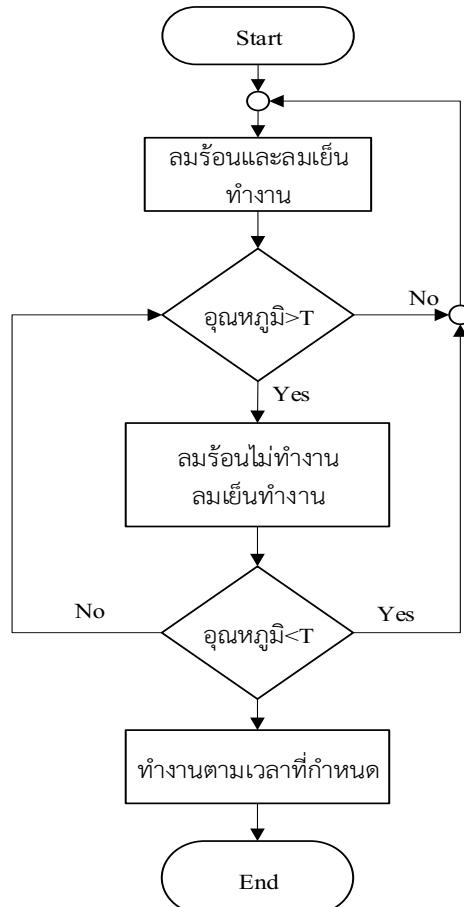
จากรูปที่ 3.3 อธิบายระบบของเครื่องซักรองเท้ากีฬาแบบอัตโนมัติ โดยระบบเมื่อเริ่มการทำงานจะทำการปล่อยน้ำเข้า และให้มอเตอร์รองเท้าหมุน 1 นาที เพื่อให้รองเท้าโดนน้ำก่อน จากนั้นก็ทำการปล่อยน้ำออกแล้วจึงถ่ายรูปที่ 1 เก็บไว้เพื่อนำไปใช้ในการประมวลผลภาพ จากนั้นก็จะเข้าสู่ระบบในการซักรองเท้ากีฬาแบบอัตโนมัติ เมื่อระบบชักทำการซักและล้างรองเท้ากีฬาเสร็จ ระบบก็จะทำการถ่ายรูปที่ 2 และนำค่าของภาพที่ 1 และภาพที่ 2

มาลับซึ่งต้องมีค่ามากกว่า X (X คือค่า SSIM ที่กำหนดไว้) ถ้าเป็นจริงระบบทำให้แห้งจะเริ่มทำงานอัตโนมัติ เมื่อระบบทำให้แห้งทำงานเสร็จแล้วเครื่องจะหยุดทำงานทั้งหมด แต่ถ้าไม่เป็นจริงเครื่องจะซักและล้างรองเท้ากีฬาอีกรอบแล้วทำการถ่ายรูปที่ 2 ใหม่ แล้วถ้ายังไม่เป็นไปตามเงื่อนไขที่กำหนดเครื่องจะทำงานซักและล้างรองเท้ากีฬา และทำการถ่ายรูปที่ 2 ไปเรื่อยๆ แต่ถ้าจำนวนรอบ (J) ครบ 2 รอบ เครื่องจะทำงานระบบทำให้แห้งโดยอัตโนมัติ และจบการทำงาน



รูปที่ 3.4 แผนผังภาพการทำงานของระบบซักและล้างรองเท้ากีฬา

จากรูปที่ 3.4 อธิบายระบบซักและล้างรองเท้ากีฬา ในขั้นตอนแรกจะเป็นระบบการซัก ซึ่งจะเริ่มจากการปล่อยน้ำเข้าเครื่อง ถ้าลูกloyตัวกลางมีค่าเท่ากับ 1 มอเตอร์ที่ใช้ควบคุมแปรรูปและมอเตอร์ที่ใช้ในการควบคุมรองเท้าจะทำการหมุนตามเวลาที่กำหนดไว้ หลังจากนั้นทำการปล่อยน้ำที่ใช้ในการซักออกจากเครื่อง ต่อไปคือการกระบวนการล้างรองเท้ากีฬา โดยการปล่อยน้ำเข้าเครื่อง แต่ในกระบวนการล้างจะให้ลูกloyตัวบนในการเข้ามาเมื่อค่าเท่ากับ 1 หรือไม่ ถ้าใช่ให้มอเตอร์ที่ใช้ในการควบคุมรองเท้า 1 นาที แล้วจึงปล่อยน้ำออก ในระบบการล้างเราจะทำ 2 รอบ



รูปที่ 3.5 แผนผังภาพการทำงานของระบบการทำให้แห้ง

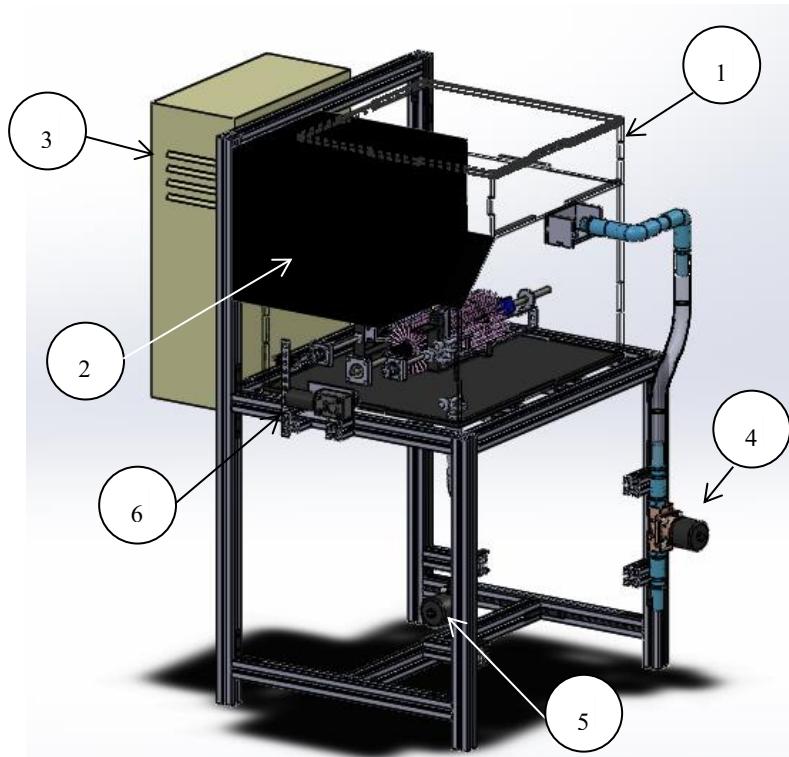
จากรูปที่ 3.5 อธิบายระบบการทำให้แห้ง ในตอนแรกลmur้อนและลmurเย็นทำงานพร้อมกัน ถ้าอุณหภูมิมากกว่า T (T คืออุณหภูมิที่เรากำหนดในการทำให้แห้ง) ลmur้อนจะหยุดทำงาน จะมีแค่ลmurเย็นที่ทำงาน เมื่ออุณหภูมิน้อยกว่าอุณหภูมิที่เรากำหนด ลmur้อนจะกลับมาทำงานพร้อมกับลmurเย็น โดยระบบจะทำงานตามเวลาที่กำหนด เอาไว้

3.2 การออกแบบโครงสร้าง

โครงสร้างดังรูปที่ 3.6 และรูปที่ 3.7 ประกอบดังนี้

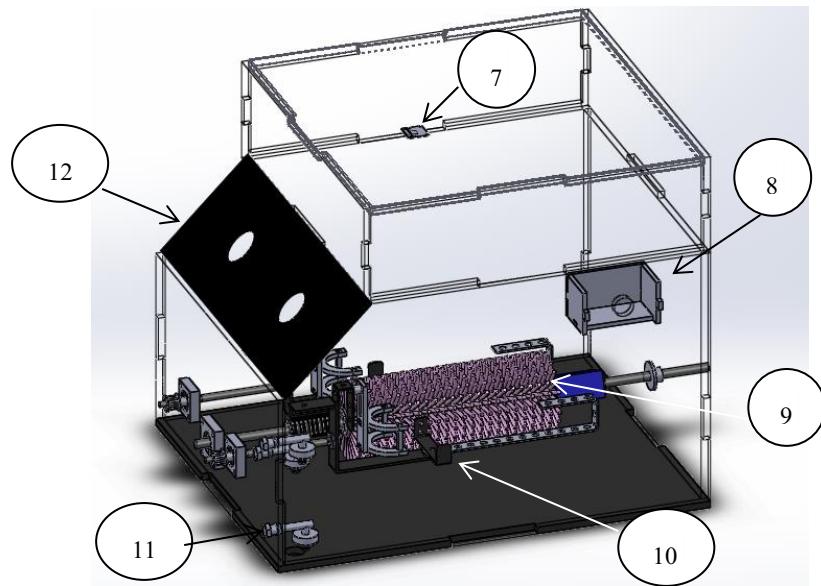
- หมายเลข 1 ตัวเครื่องซักรองเท้ากีฬา
- หมายเลข 2 กล่องระบบทำให้แห้ง
- หมายเลข 3 ตู้คอนโทรล
- หมายเลข 4 ตำแหน่งโซลินอยด์ฝังน้ำเข้า
- หมายเลข 5 ตำแหน่งโซลินอยด์ฝังน้ำออก

- หมายเลขอ 6 มอเตอร์ที่ใช้ในการหมุนร่องเท้า
- หมายเลขอ 7 กล้องสำหรับ Raspberry Pi
- หมายเลขอ 8 สำหรับใส่ผงซักฟอก
- หมายเลขอ 9 แปรง
- หมายเลขอ 10 ที่จับยึดร่องเท้า
- หมายเลขอ 11 เชนเชอร์ลูกกลอย
- หมายเลขอ 12 ช่องสำหรับกล่องระบบทำให้แห้ง



รูปที่ 3.6 ภาพองค์ประกอบโดยรวมโครงสร้างเครื่องซักรองเท้ากีฬา

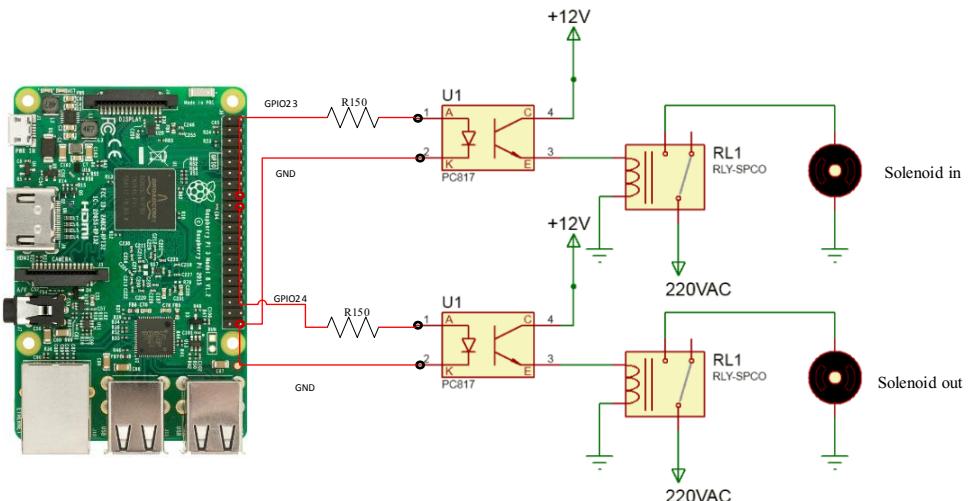
จากรูปที่ 3.6 เป็นรูปองค์ประกอบโดยรวมของเครื่องซักรองเท้ากีฬาโดย จะประกอบด้วย 3 ส่วนหลักๆคือ ตัวเครื่องซักรองเท้ากีฬา กล่องระบบทำให้แห้ง และตู้คอนโทรล ในส่วนของเครื่องซักรองเท้ามีขนาด กว้างxยาวxสูง คือ 400x500x425 มิลลิเมตร ตัวเครื่องซักรองเท้ากีฬาและกล่องระบบทำให้แห้ง จะใช้อะคริลิกในการทำโครงสร้าง ภายในกล่องระบบทำให้แห้งจะประกอบไปด้วยบอร์ดคอนโทรลเลอร์ วงจรรีเลย์ สวิตซ์เพาเวอร์ซัพพลาย และเบรกเกอร์ ส่วนภายในกล่องระบบทำให้แห้งจะประกอบไปด้วยระบบลมร้อนและระบบลมเย็น



รูปที่ 3.7 ภาพลักษณะของตัวเครื่องซักรองเท้ากีฬา

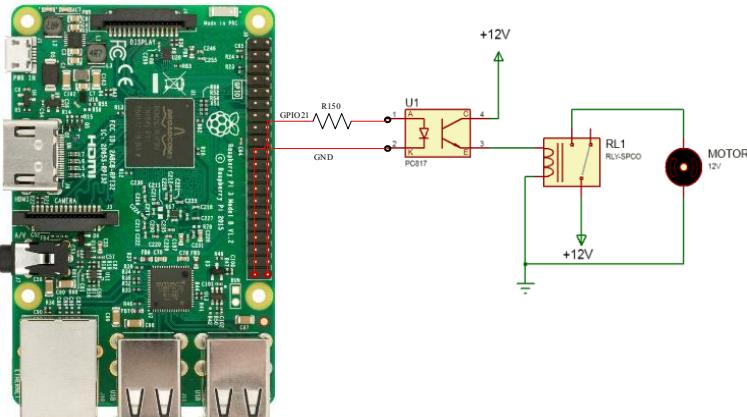
จากรูปที่ 3.7 จะเห็นได้ว่าหมายเลข 7 กล้องสำหรับ Raspberry Pi จะติดไว้ตรงฝาที่ใช้เปิด-ปิดได้ หมายเลข 8 จะมีช่องให้น้ำไหลเข้า เพื่อที่จะได้ผสมกับผงซักฟอก หมายเลข 9 คือแปรรูปที่ใช้สำหรับขัดรองเท้า ซึ่งทำมาจากชนในล่อน หมายเลข 10 คือที่จับยึดรองเท้า เพื่อให้รองเท้าอยู่กับที่และสามารถหมุนได้ 360 องศา หมายเลข 11 เชนเซอร์ลูกloy ใช้วัดระดับน้ำที่เข้ามาในตัวเครื่องและสั่งงานให้โซลินอยด์ทำงานหรือหยุดทำงาน หมายเลข 12 ช่องสำหรับกล่องระบบทำให้แห้ง เพื่อให้มั่นรอนสามารถเข้ามาในตัวเครื่องซักรองเท้าได้

3.3 การออกแบบวงจรไฟฟ้า



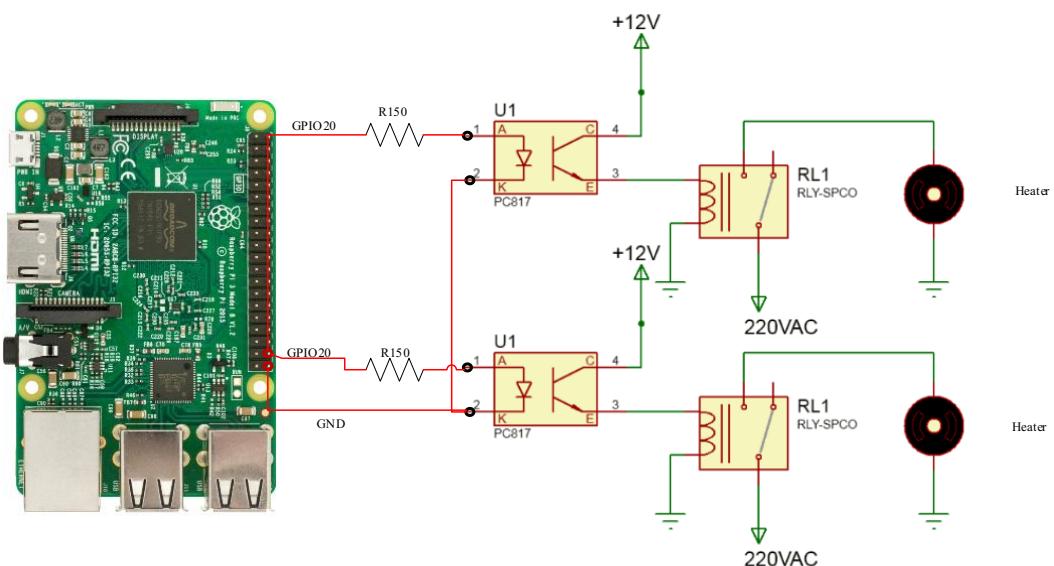
รูปที่ 3.8 วงจรควบคุม Solenoid valve

จากรูปที่ 3.8 เป็นวงจรที่ใช้ควบคุมการเปิด-ปิด solenoid valve 2 ตัว คือสำหรับทางน้ำเข้าและทางน้ำออก สั่งงานจาก Raspberry pi ผ่านทางขา GPIO ที่ 23 ควบคุมโซลินอยด์ทางน้ำเข้า และขา GPIO ที่ 24 ควบคุมโซลินอยด์ทางน้ำออก และขา GND จะไม่ต่อร่วมกัน เพราะ GND ที่เข้าเรียลย์ เป็น DC ส่วน GND เข้าโซลินอยด์ จากรูปจะใช้ opto isolate เบอร์ PC817 ในการควบคุมผ่านแสง เพื่อแยกระหว่างระบบคอนโทรล และระบบเมคานิกส์ เมื่อเกิดความเสียหายขึ้น จะทำให้ระบบหงั้ง ไม่เกิดความเสียหาย ไฟที่ใช้ 12V ได้มาจากสวิตซ์เพาเวอร์ซัพพลาย ส่วนไฟ 220VAC ได้มาจากการไฟบ้าน



รูปที่ 3.9 วงจรควบคุม Motor สำหรับหมุนรองเท้า

จากรูปที่ 3.9 เป็นวงจรที่ใช้ควบคุมมอเตอร์ที่ใช้ในการหมุนรองเท้า สั่งงานจาก raspberry pi ผ่านทางขา GPIO ที่ 21 จะสามารถหมุนมอเตอร์ได้แค่ 1 ทิศทาง จากระบบที่ใช้ opto isolate เบอร์ PC817 ในการควบคุม ผ่านแสง เพื่อแยกระหว่างระบบ control และระบบ mechanic เมื่อเกิดความเสียหายขึ้น จะทำให้ระบบทั้ง 2 ไม่เกิด ความเสียหาย ไฟที่ใช้ 12V ได้มาจาก switching supply



รูปที่ 3.10 วงจรควบคุมอีตเตอร์

จากรูปที่ 3.10 เป็นวงจรที่ใช้ควบคุมการเปิด-ปิดอีตเตอร์ 2 ตัว จะสั่งงานจาก Raspberry pi ผ่านทางขา GPIO ที่ 20 สำหรับอีตเตอร์ที่ทำงานไปพร้อมกัน 2 ตัว จากระบบที่ใช้ opto isolate เบอร์ PC817 ในการควบคุมผ่าน แสง เพื่อแยกระหว่างระบบคอนโทรล และระบบเมคานิกส์ เมื่อเกิดความเสียหายขึ้น จะทำให้ระบบทั้ง 2 ไม่เกิด ความเสียหาย ไฟที่ใช้ 12V ได้มาจากสวิตซ์เพาเวอร์ซัพพลาย ส่วนไฟ 220VAC ได้มาจากไฟบ้าน

บทที่ 4

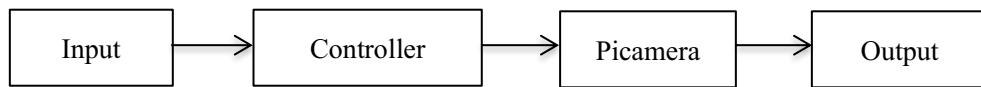
การทดลองและผลการทดลอง

4.1 การทดลองการใช้งานถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน

4.1.1 ขั้นตอนการทดลอง

- กำหนดระยะห่างที่ใช้ในการถ่ายภาพนิ่งในระยะห่างตั้งแต่ 30-50 เซนติเมตร มีค่าความละเอียดครึ่งละ 10 เซนติเมตร
- ถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน
- ทำการบันทึกผลลงตารางที่ 4.1

4.1.2 ลักษณะการทำงานของการถ่ายภาพนิ่ง



รูปที่ 4.1 การทำงานของการใช้งานถ่ายภาพนิ่ง

4.1.3 ผลการทดลองการใช้งานถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน

ตารางที่ 4.1 ผลการทดลองการใช้งานถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน

ระยะห่าง	บันทึกผล
30 เซนติเมตร	
40 เซนติเมตร	

ตารางที่ 4.1 ผลการทดลองการใช้งานถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน (ต่อ)

ระยะห่าง	บันทึกผล
50 เซนติเมตร	

จากตารางที่ 4.1 จะเห็นได้ว่าเวลาในการถ่ายภาพนิ่งในระยะห่างที่แตกต่างกัน ระยะห่าง 30 เซนติเมตร จะสังเกตจากถ่ายภาพนิ่งได้ว่าภาพไม่สามารถมองเห็นรองเท้าได้ทั้งหมด ระยะห่าง 40 เซนติเมตร จะสังเกตจากถ่ายภาพนิ่งได้ว่าภาพสามารถมองเห็นรองเท้าได้ทั้งหมดและพื้นที่ในภาพส่วนมากเป็นรูปร่องเท้า ทั้งหมด ระยะห่าง 50 เซนติเมตร จะสังเกตจากถ่ายภาพนิ่งได้ว่าภาพสามารถมองเห็นรองเท้าได้ทั้งหมด แต่ส่วนของภาพว่างจะนิ่งไปทำสามารถมองเห็นส่วนต่าง ๆ มากขึ้น ซึ่งทำให้พื้นที่ในภาพโพกสรองเท้าน้อย

สรุปได้ว่า ระยะที่ควรใช้ในการถ่ายภาพนิ่ง คือ ระยะห่าง 40 เซนติเมตร เพราะสามารถถ่ายภาพนิ่งที่มองเห็นรองเท้าได้ทั้งหมดและพื้นที่ในภาพส่วนมากเป็นรูปร่องเท้า

4.2 การทดลองจับเวลาการประมวลผลภาพในค่าพิกเซลของภาพที่แตกต่างกันและแสดงค่า MSE และค่า SSIM

4.2.1 ขั้นตอนการทดลอง

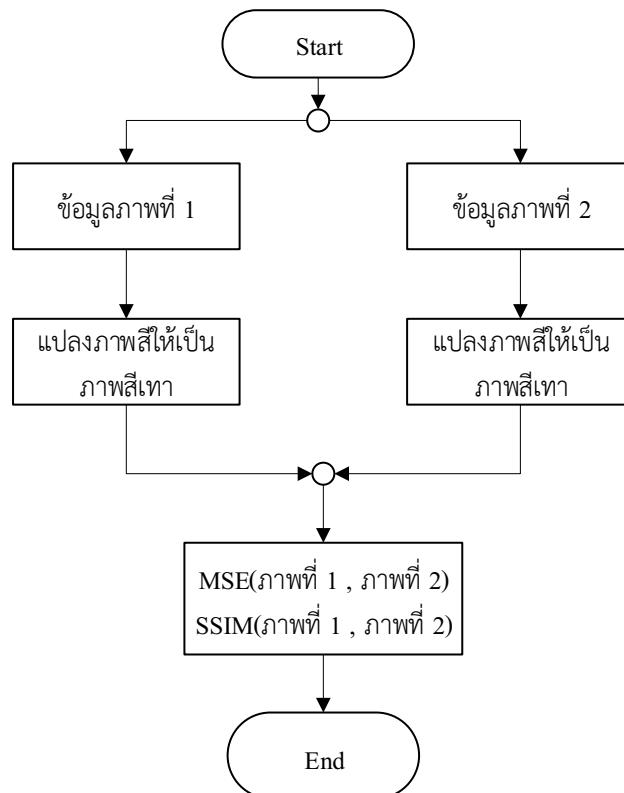
- ถ่ายรูปที่มีค่าพิกเซลตั้งแต่ 400x400 พิกเซล ถึง 1000x1000 พิกเซล มีค่าความละเอียดครึ่งละ 100x100 พิกเซล
- นำภาพที่ได้มาประมวลผลผ่านโปรแกรม python
- จับเวลาในการประมวลผลในค่าพิกเซลของภาพที่ต่างกัน
- ทำการบันทึกผลลงตารางที่ 4.2

4.2.2 ผลการทดลองจับเวลาการประมวลผลภาพในค่าพิกเซลของภาพที่ต่างกันและแสดงค่า MSE และค่า SSIM

โดยคำนวณจากสมการที่ 2.1 และ 2.11 ดังนี้

$$MSE = \frac{1}{N} \sum_{i=1}^N |x_i - y_i|^2$$

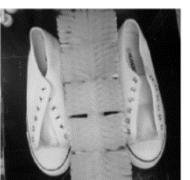
$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C1)(2\sigma_x\sigma_y + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)}$$



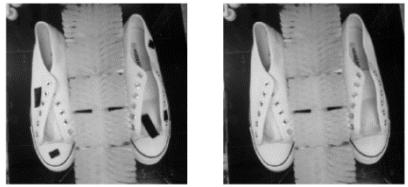
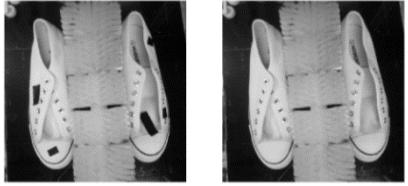
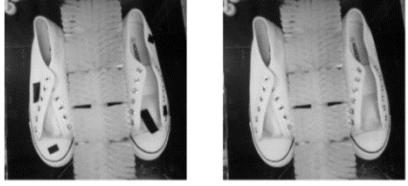
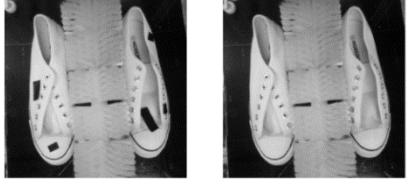
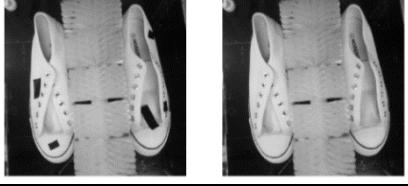
รูปที่ 4.2 การทำงานในการประมวลผลภาพ

จากรูปที่ 4.2 คือในการประมวลผลภาพจะนำข้อมูลของภาพที่ 1 และภาพที่ 2 มาเปลี่ยนให้เป็นภาพสีเทา ก่อนจึงจะนำไปประมวลผลหาค่า MSE และ SSIM

ตารางที่ 4.2 ผลการทดลองจับเวลาการประมวลผลภาพในค่าพิเศษของรูปภาพที่ต่างกันและแสดงค่า MSE และค่า SSIM

ค่าพิเศษของรูปภาพ	เวลาในการประมวลผล (วินาที)	ค่า MSE และค่า SSIM
400x400	04.66	MSE: 888.62, SSIM: 0.70  
500x500	04.76	MSE: 885.71, SSIM: 0.71  

ตารางที่ 4.2 ผลการทดลองจับเวลาการประมวลผลภาพในค่าพิกเซลของรูปภาพที่ต่างกันและแสดงค่า MSE และค่า SSIM (ต่อ)

ค่าพิกเซลของรูปภาพ	เวลาในการประมวลผล (วินาที)	ค่า MSE และค่า SSIM
600x600	04.92	MSE: 885.12, SSIM: 0.72 
700x700	05.31	MSE: 858.10, SSIM: 0.74 
800x800	05.96	MSE: 842.51, SSIM: 0.74 
900x900	06.10	MSE: 845.13, SSIM: 0.74 
1000x1000	06.33	MSE: 632.67, SSIM: 0.79 

จากตารางที่ 4.2 จะเห็นได้ว่าเวลาในการประมวลผลของค่าพิกเซลของรูปภาพที่ต่างกัน ยิ่งรูปภาพมีค่าพิกเซลที่มากขึ้น เวลาที่ใช้ในการประมวลผลก็จะมากขึ้นเช่นกัน และค่า SSIM จะเพิ่มขึ้นตามค่าพิกเซลเช่นกัน แต่ค่า MSE จะลดลง

สรุปได้ว่า เวลาในการประมาณผลและค่า SSIM และผันตกับค่าพิกเซล ส่วนค่า MSE แปรผันกับค่าพิกเซล

4.3 การทดลองตรวจสอบค่าของอุณหภูมิที่เพิ่มขึ้นเมื่อเทียบกับเวลาระหว่างเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer

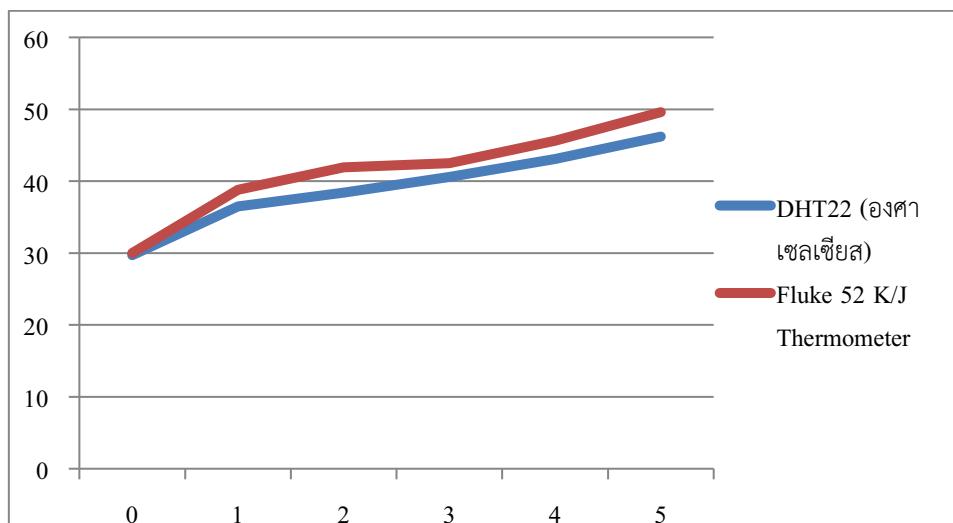
4.3.1 ขั้นตอนการทดลอง

- ตั้งเวลาระบบทำให้แห้ง แบบไม่ควบคุมอุณหภูมิ ตั้งแต่เวลา 0 – 5 นาที มีค่าความละเอียดครั้งละ 1 นาที
- สังเกตค่าของอุณหภูมิที่เพิ่มขึ้นในแต่ละครั้งเมื่อเทียบกับเวลา ระหว่างเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer
- ทำการบันทึกผลการทดลองลงตารางที่ 4.3

ตารางที่ 4.3 ผลการทดลองค่าของอุณหภูมิที่เพิ่มขึ้นเมื่อเทียบกับเวลาระหว่างเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer

เวลา (นาที)	DHT22 (องศาเซลเซียส)	Fluke 52 K/J Thermometer (องศาเซลเซียส)	ผลต่างที่วัดได้
0	29.7	30.0	0.3
1	36.5	38.8	2.3
2	38.4	41.9	3.5
3	40.6	42.5	1.9
4	43.1	45.6	2.5
5	46.2	49.6	3.4

จากตารางที่ 4.3 จะเห็นได้ว่าอุณหภูมิตั้งแต่ระบบทำให้แห้งขึ้นไม่เริ่มการทำงาน ค่าของเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer ที่วัดได้มีความแตกต่างกัน และเมื่อเริ่มทำงานระบบทำให้แห้ง ค่าที่วัดได้มีค่าสูงขึ้น แต่มีบ้างช่วงที่ค่าความแตกต่างลดลง ดังรูปที่ 4.3 จากราฟจะเห็นได้ว่าอุณหภูมิที่วัดด้วย DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer ในเวลานาทีที่ 0 ใกล้เคียงกันจนเส้นเกือบจะทับกัน และในเวลาที่เพิ่มขึ้นราบทั้งสองเส้นมีระยะห่างกันไม่เกิน ± 5 องศาเซลเซียส



รูปที่ 4.3 ผลการทดลองค่าของอุณหภูมิระหว่างเซนเซอร์ DHT22 และ Fluke 52 K/J Thermometer

สรุปได้ว่า ค่าของเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer ที่วัดได้มีความแตกต่างกันทุกนาทีที่ระบบทำให้แห้งทำงาน

4.4 การทดลองระบบทำให้แห้งในอุณหภูมิตั้งแต่ 40 - 55 องศา มีค่าความละเอียดครั้งละ 5 องศาเซลเซียส ในเวลา 0 – 20 นาที มีค่าความละเอียดครั้งละ 5 นาที

4.4.1 ขั้นตอนการทดลอง

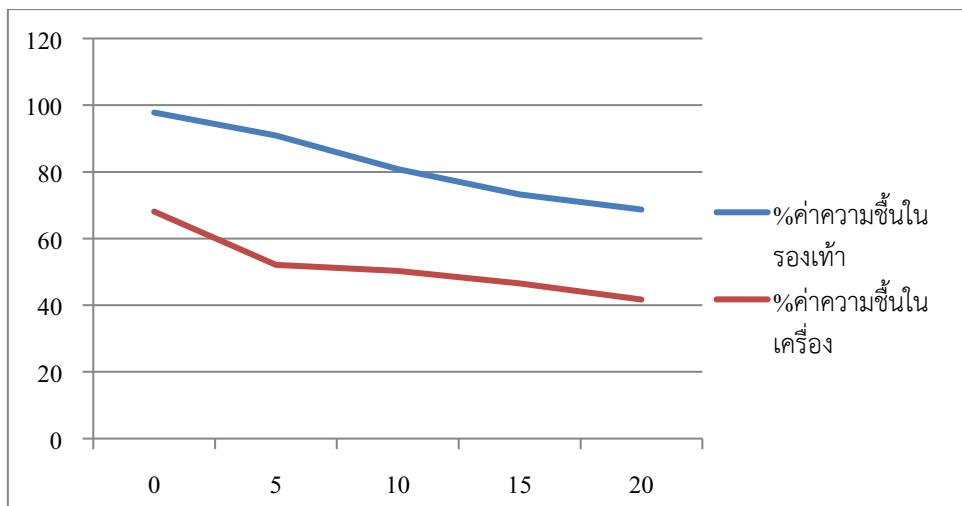
- ตั้งอุณหภูมิตามที่ต้องการทำการทำทดลองตั้งแต่ 40 - 55 องศา มีค่าความละเอียดครั้งละ 5 องศา
- ตั้งเวลาระบบทำให้แห้ง ตั้งแต่เวลา 0 – 20 นาที มีค่าความละเอียดครั้งละ 5 นาที
- ใช้เซนเซอร์ DHT22 ในการวัดค่าความชื้นภายในรองเท้า
- ทำการบันทึกผลลงตารางที่ 4.4

ตารางที่ 4.4 ผลการทดลองระบบทำให้แห้งในอุณหภูมิตั้งแต่ 40 - 55 องศาเซลเซียส มีค่าความละเอียดครั้งละ 5 องศาเซลเซียส ในเวลา 0 – 20 นาที มีค่าความละเอียดครั้งละ 5 นาที

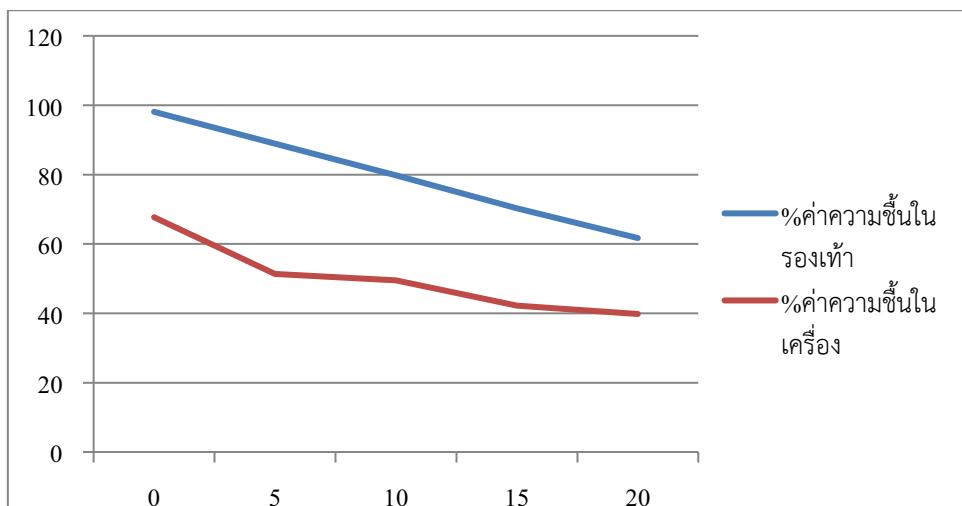
อุณหภูมิ (องศาเซลเซียส)	เวลา (นาที)	%ค่าความชื้นในรองเท้า	%ค่าความชื้นในเครื่อง
40	0	97.8	68.1
	5	90.9	52.1
	10	80.8	50.3
	15	73.2	46.5
	20	68.7	41.7
45	0	98.1	67.7
	5	88.9	51.3
	10	79.8	49.5
	15	70.3	42.2
	20	61.7	39.8
50	0	96.7	68.4
	5	84.5	48.7
	10	72.4	43.2
	15	65.8	38.7
	20	53.7	32.3
55	0	98.3	69.1
	5	81.2	45.9
	10	69.7	40.7
	15	57.9	35.3
	20	45.1	29.6

จากตารางที่ 4.4 ในแต่ละช่วงเวลาและอุณหภูมิจะเห็นได้ว่า ยิ่งเวลาเพิ่มมากขึ้นค่าของ %ค่าความชื้นจะมีค่าที่ลดลงเรื่อย ๆ และในแต่ละช่วงอุณหภูมิจะมีค่าของ %ค่าความชื้นที่แตกต่างกันในช่วงเวลากันที่เท่ากัน ยิ่งอุณหภูมิสูงค่าของ %ค่าความชื้นก็จะยิ่งลดลงเร็วมากขึ้น %ค่าความชื้นภายในรองเท้าและ%ค่าความชื้นภายในเครื่องซึกรองเท้าก็มีค่าไม่เท่ากัน %ค่าความชื้นภายในรองเท้ามีค่ามากกว่า%ค่าความชื้นภายในเครื่องซึกรองเท้า

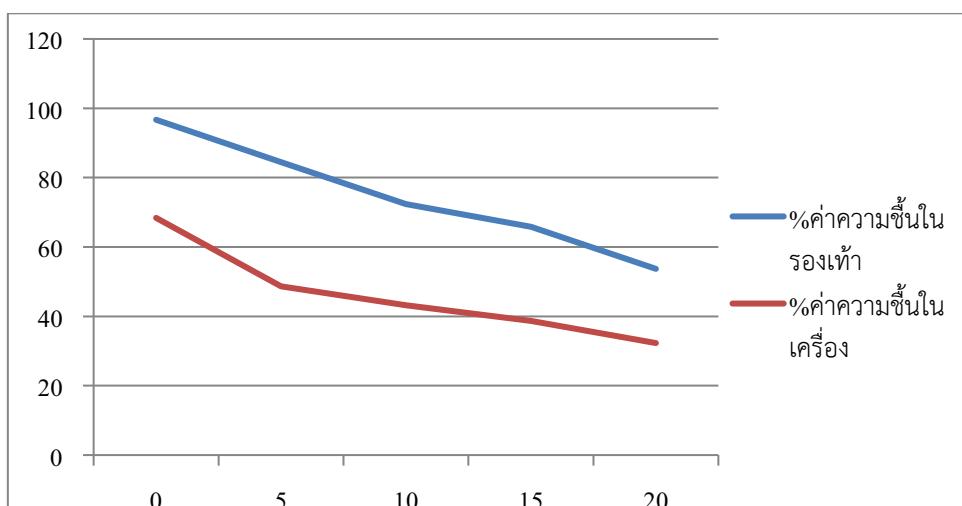
กีฬา เพราะปริมาณไอน้ำที่มีอยู่ในอากาศภายในเครื่องซึ่งกรองเท้ากีฬาในขณะนั้น ต่อปริมาณไอน้ำที่มีอยู่ภายในรองเท้ามีน้อยกว่า



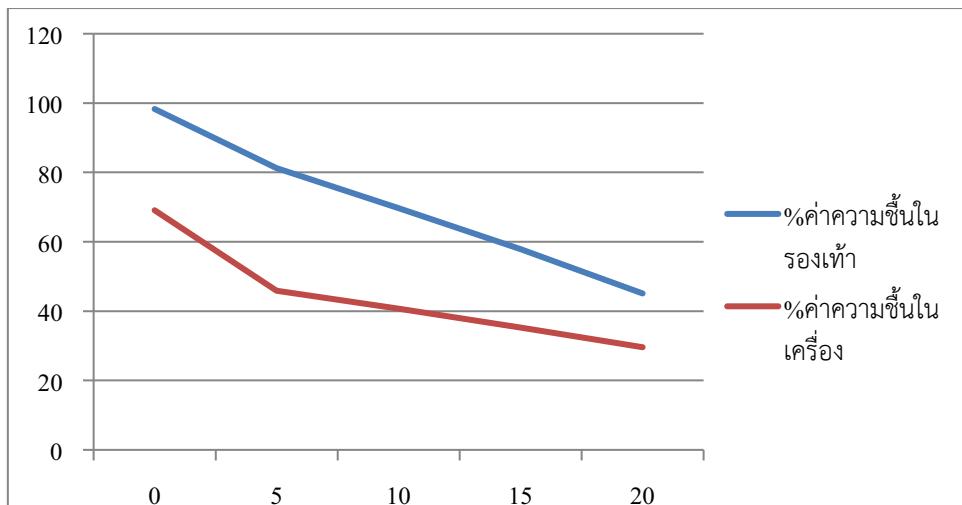
รูปที่ 4.4 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 40 องศาเซลเซียสในเวลา 0 – 20 นาที



รูปที่ 4.5 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 45 องศาเซลเซียสในเวลา 0 – 20 นาที



รูปที่ 4.6 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 50 องศาเซลเซียสในเวลา 0 – 20 นาที



รูปที่ 4.7 ผลการทดลองระบบทำให้แห้งในอุณหภูมิ 55 องศาเซลเซียสในเวลา 0 – 20 นาที

จากรูปที่ 4.4 – 4.7 จะเห็นได้ว่ากราฟของเปอร์เซ็นต์ค่าความชี้นภายในรองเท้าและเปอร์เซ็นต์ค่าความชี้นภายในเครื่องมีความแตกต่างกันมาก เนื่องจากในรองเท้ามีความชี้นเพราะโน่นน้ำ แต่ภายในเครื่องจะมีความแห้งของอากาศ ทำให้ภายในเครื่องมีเปอร์เซ็นต์ค่าความชี้นน้อยกว่า ระบบทำให้แห้งยิ่งมีอุณหภูมิสูงมากขึ้นก็จะยิ่งทำให้เปอร์เซ็นต์ค่าความชี้นลดลงเร็วขึ้นเมื่อเทียบกับเวลา

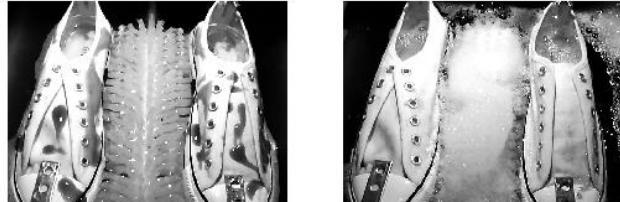
สรุป เวลาและอุณหภูมิแปลงผันกับค่าของ % ค่าความชี้น % ค่าความชี้นภายในรองเท้ามีค่ามากกว่า % ค่าความชี้นภายในเครื่องซึ่งรองเท้ากีฬา

4.5 การทดลองระบบชัก ในเวลา 1-8 นาที มีค่าความลักษณะเอี้ยดครั้งละ 1 นาที

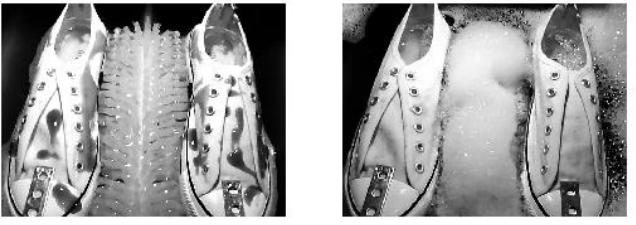
4.5.1 ขั้นตอนการทดลอง

- นำรองเท้าใส่ลงไปในเครื่องซักรองเท้ากีฬา
- เลือกรอบการทำงานเป็นระบบชักและตั้งเวลาในการทำงานตั้งแต่ 1-8 นาที มีค่าความลักษณะเอี้ยดครั้งละ 1 นาที
- ทำการบันทึกผลลัพธาระบบที่ 4.5

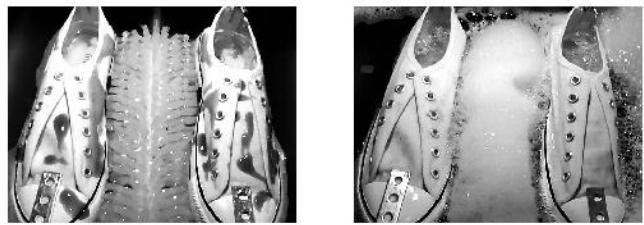
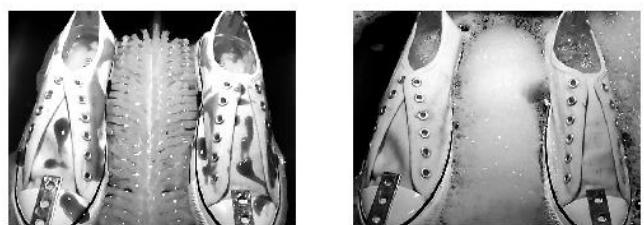
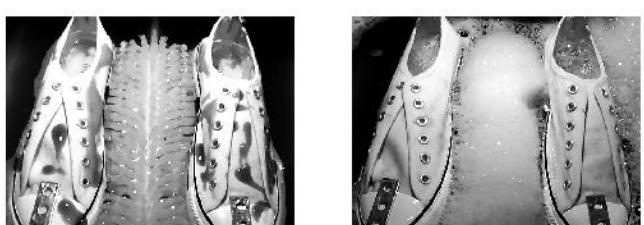
ตารางที่ 4.5 ผลการทดลองระบบชัก ในเวลา 1-8 นาที มีค่าความลักษณะเอี้ยดครั้งละ 1 นาที

เวลา (นาที)	ค่า MSE และ ค่า SSIM
1	MSE: 4863.43, SSIM: 0.28 

ตารางที่ 4.5 ผลการทดลองระบบชัก ในเวลา 1-8 นาที มีค่าความละเอียดครั้งละ 1 นาที (ต่อ)

	MSE: 5532.45, SSIM: 0.28
2	
3	MSE: 5888.07, SSIM: 0.26
4	MSE: 6406.42, SSIM: 0.25
5	MSE: 6116.23, SSIM: 0.24

ตารางที่ 4.5 ผลการทดลองระบบชัก ในเวลา 1-8 นาที มีค่าความละเอียดครั้งละ 1 นาที (ต่อ)

	MSE: 6093.44, SSIM: 0.23
6	
7	
8	

จากตารางที่ 4.5 จะเห็นได้ว่า ค่าของ MSE และ SSIM จะลดลงเรื่อยๆ ตามเวลาที่ชัก เมื่อชักผ่านไป 1 นาที จะเห็นได้ว่าค่าของ MSE มีค่ามากกว่า 4500 และ ค่า SSIM มีค่าน้อยกว่า 0.3 เมื่อชักไปนานๆ จะเห็นได้ว่า ค่าของของ MSE เริ่มมีการเปลี่ยนแปลงเล็กน้อย และ ค่า SSIM มีค่าคงที่ที่ 0.23

สรุปได้ว่า ยิ่งชักในเวลาที่มากขึ้น ค่าของ MSE มีค่ามากขึ้นเรื่อยๆ และ ค่า SSIM มีค่าน้อยลง จะหยุดนิ่ง ที่ค่า SSIM เท่ากับ 0.23

ดังนั้น ค่าของ SSIM ที่เหมาะสมแก่การประมาณผลคือค่าที่น้อยกว่า 0.25

บทที่ 5

สรุปผลการทดลอง

5.1 สรุปผลการทดลอง

จากการทดลองที่ 4.1 จากรезультатการทดลองพบว่า เมื่อระยะห่างของภาพนิ่งที่เกิดขึ้น ส่งผลให้ภาพที่มีความห่างนั้นมีความสว่าง ความมืดที่แตกต่างกัน ทำให้การวิเคราะห์ครบสกปรกในระยะที่แตกต่างกันทำได้ยากขึ้น

จากการทดลองที่ 4.2 จากรезультатการทดลองพบว่าเมื่อพิกเซลของภาพที่มีขนาดเล็ก ทำให้การประมวลผลภาพได้เวลาที่ร่วดเร็ว ในขณะที่ภาพที่มีพิกเซลที่ใหญ่ขึ้นส่งผลให้การประมวลผลภาพช้าลง

จากการทดลองที่ 4.3 จากรезультатการทดลองพบว่าค่าของเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer ที่วัดได้มีความแตกต่างกันทุกนาทีที่ระบบทำให้แห้งทำงาน

จากการทดลองที่ 4.4 จากรезультатการทดลองพบว่าเมื่อระบบทำให้แห้งทำงาน %ค่าความชื้นจะลดลงเรื่อย ๆ ตามเวลาที่เพิ่มมากขึ้น และในแต่ละช่วงอุณหภูมิจะมี%ค่าความชื้นที่ลดลงต่างกัน ยิ่งอุณหภูมิสูงยิ่ง%ค่าความชื้นจะลดลงมากขึ้น %ค่าความชื้นภายในรองเท้ามีค่ามากกว่า%ค่าความชื้นภายในเครื่องซักอบรีดรองเท้ากีฬา

จากการทดลองที่ 4.5 จากรезультатการทดลองพบว่าอย่างซักในเวลาที่มากขึ้น ค่าของ MSE มีค่ามากขึ้นเรื่อย ๆ และ ค่า SSIM มีค่าน้อยลง จะหยุดนิ่งที่ค่า SSIM เท่ากับ 0.23

จากการทดลองระบบ พบร้าในการประมวลผลภาพ ระยะห่างของภาพนิ่งกับกล้องควรเลือกระยะห่างที่ 40 เซนติเมตรเป็นระยะที่เหมาะสมในการนำมาประมวลผลภาพ และเลือกค่าพิกเซลของภาพ 500x500 เพื่อการประมวลผลที่รวดเร็ว ค่าของเซนเซอร์ DHT22 และเครื่องวัดอุณหภูมิ Fluke 52 K/J Thermometer ที่วัดได้มีผลต่าง ± 5 องศา เมื่อระบบทำให้แห้งทำงาน %ค่าความชื้นจะลดลงชี้น้อยกว่ากับเวลาที่ใช้ในการทำงานและอุณหภูมิที่ใช้ ยิ่งเวลาในการซักมากขึ้นก็จะยิ่งทำให้ค่าของ MSE และ SSIM มีการเปลี่ยนแปลงตลอด

5.2 ปัญหาที่เกิดขึ้น

จากการทดลองที่ 4.2 หากทำการบันทึกภาพต่างสถานที่ ต่างเวลากัน ส่งผลให้การวิเคราะห์ของภาพแตกต่างกัน ก่อให้การประมวลภาพในหน่วยความจำที่ค่าที่แตกต่างกัน

จากการทดลองที่ 4.3 ในบางช่วงเวลาเซนเซอร์ DHT22 ไม่สามารถเชื่อมต่อกับบอร์ดคอนโทรลได้ ทำให้การวัดอุณหภูมิเกิดความผิดพลาด

จากการทดลองที่ 4.4 ในบางช่วงเวลาเซนเซอร์ DHT22 ไม่สามารถเชื่อมต่อกับบอร์ดคอนโทรลได้ ทำให้การวัดอุณหภูมิเกิดความผิดพลาด เลยทำให้การควบคุมอุณหภูมิอาจเกินความผิดพลาดในบางช่วงเวลา

จากการทดลองที่ 4.5 ในการปล่อยน้ำออกมัจจะเกิดปัญหาการอุดตันที่โซลินอยด์ ทำให้น้ำไหลออกช้า

5.3 แนวทางการพัฒนา

- ปรับเปลี่ยนทางน้ำเข้า-ออกจากโซลินอยด์เป็นไฟฟ้าแทน ซึ่งจะทำให้น้ำไหลเร็วขึ้น
- การประมวลผลภาพควรໂฟก์ส่องเท้าให้มากกว่านี้ เพื่อให้การประมวลผลภาพมีประสิทธิภาพมากขึ้น

เอกสารอ้างอิง

- [1] Grayson, J. E., “Python and Tkinter Programming”. the United States of America, Manning Publications, January 2000.
- [2] ทวีรัตน์ นวลช่วย. (2016). Internet of Things (IoT) ตอนที่ 5. สืบค้นวันที่ 8 สิงหาคม พ.ศ.2561 .
สืบค้นจาก: <http://taweerat.blogspot.com/2016/06/internet-of-things-iot-5.html>
- [3] манพ ตันตะบันฑิตย์, “วัสดุวิศวกรรม”, สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2545

ภาคผนวก ก.

รายงานเสนอขออนุมัติโครงการนิเทศกรรม

รายงานเสนออนุมัติโครงการวิศวกรรม

เครื่องซักรองเท้ากีฬาแบบอัตโนมัติโดยใช้การประมวลผลภาพ

Automatic Sport Shoes Washing Machine using by Image Processing

เสนอต่อ

รองคณบดีฝ่ายสถาบันนวัตกรรมมหาวิทยาลัยเทคโนโลยีมหานคร

คณบดีวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีมหานคร

เสนอโดย

นายสิณภัส สุขดี รหัสนักศึกษา 5711110149

นักศึกษาสาขาวิชาวิศวกรรมเมchatronics หลักสูตรวิศวกรรมศาสตรบัณฑิต

นางสาวณัฐธิชา พันธ์รง รหัสนักศึกษา 5711110405

นักศึกษาสาขาวิชาวิศวกรรมเมchatronics หลักสูตรวิศวกรรมศาสตรบัณฑิต

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ ดร.ชนม์รัตน์ ตติยะวนันท์

สถาบันนวัตกรรมมหาวิทยาลัยเทคโนโลยีมหานคร

คณบดีวิศวกรรมศาสตร์ มหาวิทยาลัยเทคโนโลยีมหานคร

เริ่มโครงการวันที่ 19 สิงหาคม พ.ศ. 2560

ระยะเวลา 1 ปี

บทนำ

การซักกรองเท้ากีฬาจะประสบความยุ่งยากในการซักและการทำให้ร่องเท้าแห้ง เพราะส่วนมากต้องใช้มือในการซักทำความสะอาดรองเท้ากีฬา ซึ่งใช้เวลานานพอกสมควรในการซักทำความสะอาดแต่ละคู่ การซักทำความสะอาดนั้นต้องมีการใช้ผงซักฟอก เพื่อให้ร่องเท้ากีฬามีความสะอาดมากขึ้นซึ่งผงซักฟอกจะมีฤทธิ์เป็นเบสที่สามารถกัดกร่อนมือได้ อาจทำให้เกิดอาการแพ้ได้สำหรับบางคน วิธีแก้ส่วนใหญ่ คือการเอารองเท้ากีฬาทำความสะอาดด้วยเครื่องซักผ้า เพื่อที่จะให้สัมผัสผงซักฟอกน้อยที่สุดและประหยัดเวลาในการซักทำความสะอาด ซึ่งอาจจะส่งผลให้รองเท้ามีความเสียหายได้ เนื่องจากการกระแทกกับขอบลังซัก ในส่วนของการทำให้ร่องเท้ากีฬาแห้งส่วนใหญ่จะใช้แสงแดดจากดวงอาทิตย์ในการทำให้ร่องเท้ากีฬาแห้ง ซึ่งบางวันอาจจะไม่มีแสงแดด ทำให้ร่องเท้าไม่แห้งและก่อให้เกิดกลิ่นอับชื้นจากการร่องเท้ากีฬา ปัจจุบันยังไม่ค่อยมีเครื่องที่ใช้สำหรับซักกรองเท้ากีฬาและทำให้ร่องเท้าแห้งได้

จึงมีแนวคิดที่จะสร้างเครื่องซักกรองเท้ากีฬา โดยใช้การประมวลผลภาพ ซึ่งใช้ค้อนไทรอลเลอร์ในการประมวลผล เพื่ออำนวยความสะดวกต่อผู้ใช้งานในการซักกรองเท้ากีฬาและการทำให้ร่องเท้ากีฬาแห้ง

การทำโครงงานครั้งนี้เป็นการนำปัญหาการส่วนตัวของบุคคลทั่วไปที่ใช้รองเท้ากีฬานิดหน้าที่ต้องการซักกรองเท้ากีฬาที่มีความสกปรกให้สะอาดมากยิ่งขึ้น

1. วัตถุประสงค์

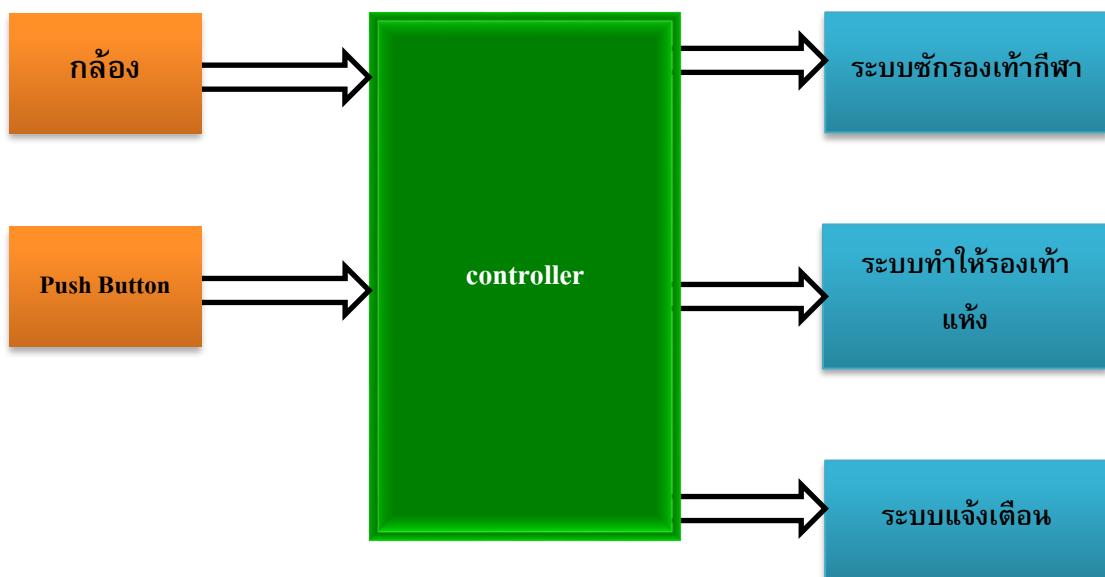
- เพื่อตอบสนองความต้องการของบุคคลทั่วไปในการซักกรองเท้ากีฬา
- เพื่อศึกษาการทำงานของระบบการทำให้ร่องเท้าแห้ง
- เพื่อเป็นการสร้างเครื่องซักกรองเท้า
- เพื่อลดความเสียหายของรองเท้าที่เกิดจากการซักให้น้อยที่สุด
- เพื่อแก้ปัญหาการซักกรองเท้าและกีตากองเท้าในวันที่ไม่มีแสงอาทิตย์

2. ขอบเขตการดำเนินงาน

- โครงงานนี้มีขนาดไม่เกิน 800x800x1000 mm
- สามารถซักกรองเท้ากีฬาได้ครั้งละ 1 คู่
- สามารถรับรองเท้ากีฬา ขนาดตั้งแต่ 36 - 42 ไซซ์ตาม Europe
- สามารถซักและล้างรองเท้ากีฬาแบบผ้าเท่านั้น
- มีระบบทำให้ร่องเท้าแห้งด้วยระบบควบคุมอุณหภูมิของลมตั้งแต่ 40 - 55 องศา มีค่าความละเอียดครั้งละ 5 องศา และมีค่าความผิดพลาดไม่เกิน ± 2 องศา ณ ขณะอ่าน
- สามารถใช้การประมวลผลภาพในการตรวจสอบและแสดงผลความสะอาดของรองเท้าที่ประposeเป็นได้
- มีระบบแจ้งเตือน เช่น แจ้งเตือนเมื่อน้ำล้นถังที่เกิดจากความผิดพลาดของการทำงาน, แจ้งเตือนเมื่อระบบทำงานเสร็จเรียบร้อย
- สามารถสั่งงานเฉพาะทำให้แห้งได้ เพื่อเพิ่มเวลาในการณ์ที่ไม่แห้ง

3. ขั้นตอนการดำเนินการ

- ศึกษาหาข้อมูลเกี่ยวกับเครื่องซักต่างๆ
- ออกแบบเครื่องซักรองเท้ากีฬา
- ศึกษาหาวัสดุที่เหมาะสมในการสร้างโครงสร้าง
- ศึกษาระบบทามให้รองเท้าแห้ง
- ทำการประกอบเขื่อมต่อระบบ
- ศึกษาการเขียนโปรแกรมในมircrocontroller
- เขียนโปรแกรมในmicrocontroller
- ทดลองการทำงานของตัวเครื่อง
- จัดทำปริญญาพนธ์



รูปที่ 3.1 การทำงานของโครงงานวิศวกรรม

จากรูปที่ 3.1 จะเห็นได้ว่าการทำงานจะใช้มircrocontrollerเป็นตัวประมวลผลหลักในการรับค่าอินพุต ซึ่งตัวอินพุต คือการสั่งงานด้วยการกดปุ่มและรับค่ารูปภาพมาจากกล้อง และนำค่าอินพุตมาประมวลผลภาพและสั่ง การให้ค่าเอาร์พุตที่ได้ไปยังเมมโมนิกส์เครื่องซักรองเท้ากีฬา โดยจะมีอุปกรณ์ประกอบไปด้วย มอเตอร์สเต็ปปิ้ง 3 ตัว, แพร่ลูกกลิ้งสำหรับซักรองเท้า 2 อัน, มอเตอร์กระแสตรง 2 ตัว, วาล์วน้ำโซลินอยด์จำนวน 2 ตัว และเซนเซอร์ ลูกลอยวัดน้ำล้น 2 ตัว ซึ่งค่าเอาร์พุตจะมีการการทำงานด้วยกัน 3 ระบบ คือ ระบบซักรองเท้ากีฬา ,ระบบทำให้ ร่องเท้ากีฬาแห้ง และระบบแจ้งเตือน เช่น แจ้งเตือนเมื่อน้ำล้นถังที่เกิดจากความผิดพลาดของการทำงาน, แจ้งเตือน เมื่อ ระบบทำงานเสร็จเรียบร้อย

3.1 โครงการวิศวกรรม 1

- ศึกษาเกี่ยวกับระบบการทำงานของเครื่องซักผ้า
- ศึกษาハウสดูที่เหมาะสมในการสร้างโครงสร้าง
- ศึกษาเกี่ยวกับระบบทำให้ร่องเท้าแห้ง
- ออกแบบชิ้นงานและจัดทำโครงการวิศวกรรม 1
- ทำการทดลองระบบต่างๆ
- จัดทำรายงานโครงการวิศวกรรม 1

3.2 โครงการวิศวกรรม 2

- จัดทำโครงการวิศวกรรม 2
- ออกแบบระบบไฟฟ้า
- ทำการเชื่อมต่อระบบทั้งหมด
- เขียนโปรแกรมไมโครคอนโทรลเลอร์และการแสดงผลทั้งหมด
- ทำการทดลองระบบต่างๆ
- จัดทำปริญญา呢พนธ์

ตารางที่ 3.1 การดำเนินงานโครงการวิศวกรรม 1

แผนงานในแต่ละสัปดาห์	สิงหาคม				กันยายน				ตุลาคม				พฤษจิกายน			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. ศึกษาเกี่ยวกับระบบการทำงานของเครื่องซักผ้า																
2. ศึกษาハウสดูที่เหมาะสมในการสร้างโครงสร้าง																
3. ศึกษาเกี่ยวกับระบบทำให้ร่องเท้าแห้ง																
4. ออกแบบชิ้นงานและจัดทำโครงการวิศวกรรม 1																
5. ทำการทดลองระบบต่างๆ																
6. จัดทำรายงานโครงการวิศวกรรม 1																

ตารางที่ 3.2 การดำเนินงานโครงการวิศวกรรม 2

แผนงานในแต่ละสัปดาห์	มกราคม				กุมภาพันธ์				มีนาคม				เมษายน				พฤษภาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. จัดทำโครงการวิศวกรรม 2																				
2. ออกแบบระบบไฟฟ้า																				
3. ทำการเชื่อมต่อระบบทั้งหมด																				
4. เขียนโปรแกรมไมโครคอนโทรลเลอร์และการแสดงผลทั้งหมด																				
5. ทำการทดลองระบบต่างๆ																				
6. จัดทำปริญญา呢พนธ์																				

4. เอกสารอ้างอิง

- [1] ณรงค์ ตันชีวะวงศ์, “เมคคาทรอนิกส์เบื้องต้น”, สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2551
- [2] พร้อมเดช หล่อวิจิตร, “คู่มือเขียนแอพ Android ฉบับสมบูรณ์”, บริษัท โปรดิวชั่น จำกัด, 2556
- [3] ประจิน พลังสันติกล, “พื้นฐานภาษา C สำหรับ Android”, บริษัท แอพซอฟต์เทค จำกัด, 2553
- [4] นานพ ตันตะระบันฑิตย์, “วัสดุวิศวกรรม”, สำนักพิมพ์สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น), 2545

5. บุคลากร

จำนวนนักศึกษา 2 คน ได้แก่

นายสิมภัส สุขดี รหัสนักศึกษา 5711110149
นางสาวณัฐธิชา พันธ์รง รหัสนักศึกษา 5711110405

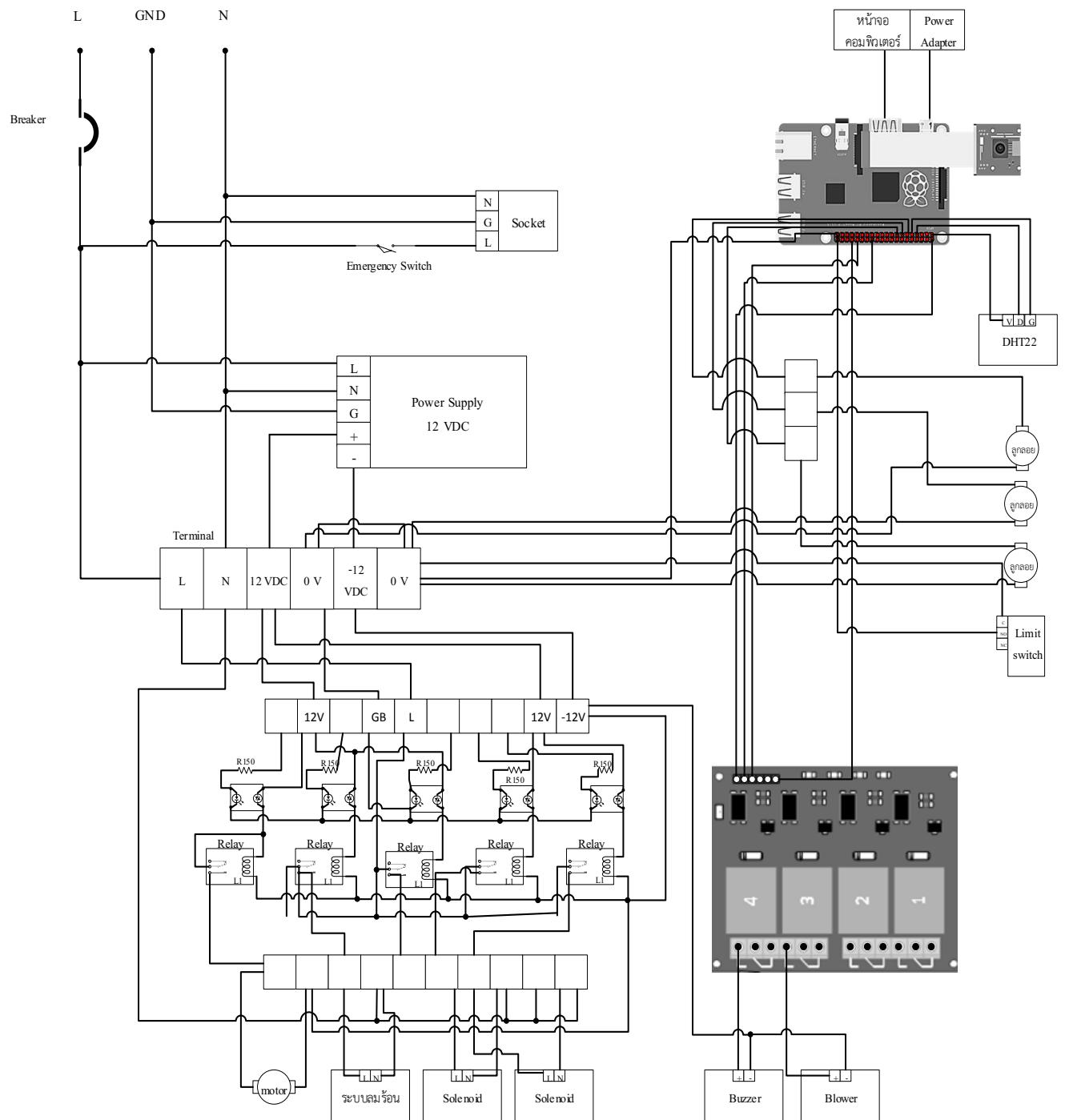
6. งบประมาณ

รายการ	งบประมาณ (บาท)
1. คอนโดมิเนียม	3,000.00
2. มอเตอร์ไซด์	1,000.00
3. มอเตอร์ไซด์รถแต่ง	1,000.00
4. แพร่ลูกกลิ้งสำหรับซักรองเท้า	1,000.00
5. ชุดโครงสร้าง	5,000.00
6. วาร์วน้ำโซลินอยด์	2,000.00
7. เชนเชอร์ลูกloyวัดน้ำล้าน	2,000.00
8. กล้อง	3,000.00
9. อุปกรณ์อื่นๆ	10,000.00
รวมทั้งสิ้น	28,000.00

ภาคผนวก ๊.

การออกแบบวงจร

220 VAC



รูปที่ ข-1 วงจรรวมของระบบชุดควบคุม

ການຜົນງານ ດ

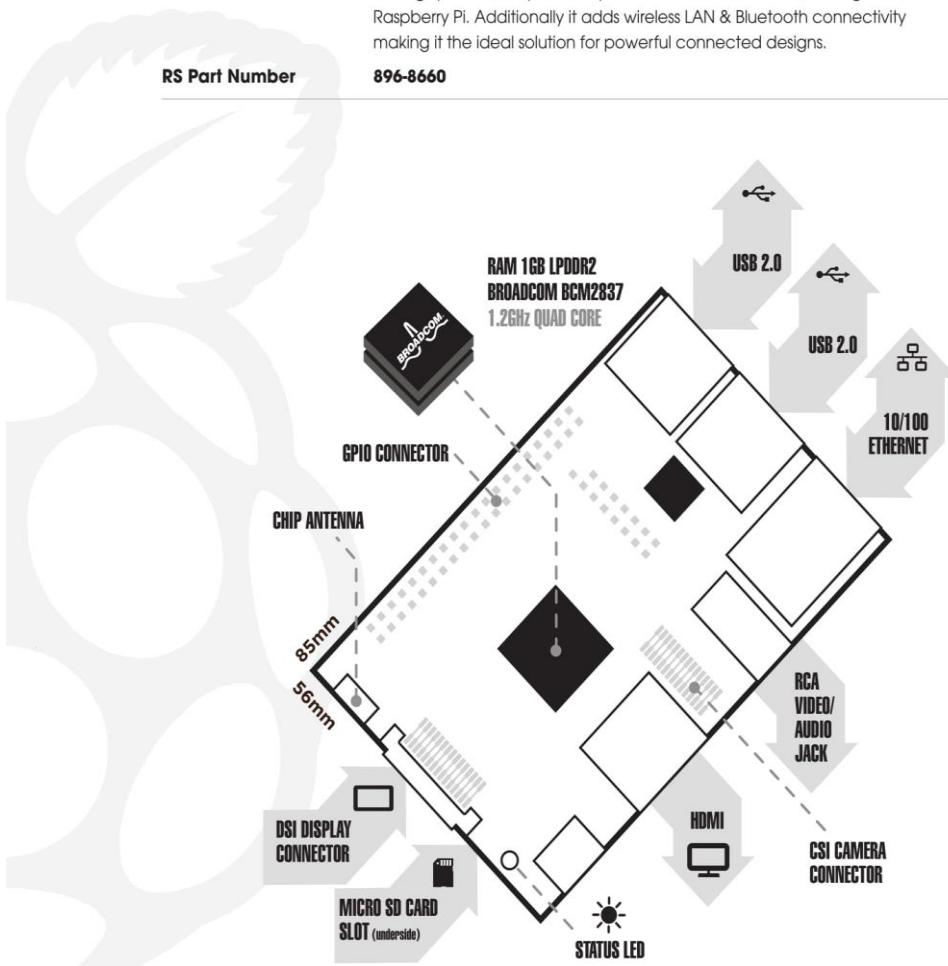
Data Sheets

Raspberry Pi 3



Raspberry Pi 3 Model B

Product Name	Raspberry Pi 3
Product Description	The Raspberry Pi 3 Model B is the third generation Raspberry Pi. This powerful credit-card sized single board computer can be used for many applications and supersedes the original Raspberry Pi Model B+ and Raspberry Pi 2 Model B. Whilst maintaining the popular board format the Raspberry Pi 3 Model B brings you a more powerful processor, 10x faster than the first generation Raspberry Pi. Additionally it adds wireless LAN & Bluetooth connectivity making it the ideal solution for powerful connected designs.
RS Part Number	896-8660





Raspberry Pi 3 Model B

Specifications

Processor	Broadcom BCM2837 chipset. 1.2GHz Quad-Core ARM Cortex-A53 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
GPU	Dual Core VideoCore IV® Multimedia Co-Processor. Provides Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
Memory	1GB LPDDR2
Operating System	Boots from Micro SD card, running a version of the Linux operating system or Windows 10 IoT
Dimensions	85 x 56 x 17mm
Power	Micro USB socket 5V1, 2.5A

Connectors:

Ethernet	10/100 BaseT Ethernet socket
Video Output	HDMI (rev 1.3 & 1.4) Composite RCA (PAL and NTSC)
Audio Output	Audio Output 3.5mm jack, HDMI USB 4 x USB 2.0 Connector
GPIO Connector	40-pin 2.54 mm (100 mil) expansion header: 2x20 strip Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines
Camera Connector	15-pin MIPI Camera Serial Interface (CSI-2)
Display Connector	Display Serial Interface (DSI) 15 way flat flex cable connector with two data lanes and a clock lane
Memory Card Slot	Push/pull Micro SDIO

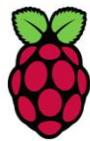
Key Benefits

- Low cost
- 10x faster processing
- Consistent board format
- Added connectivity

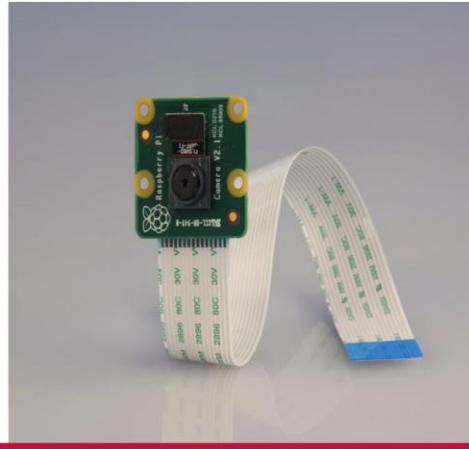
Key Applications

- Low cost PC/tablet/laptop
- Media centre
- Industrial/Home automation
- Print server
- Web camera
- Wireless access point
- Environmental sensing/monitoring (e.g. weather station)
- IoT applications
- Robotics
- Server/cloud server
- Security monitoring
- Gaming

Raspberry Pi Camera Module



Raspberry Pi



Camera Module

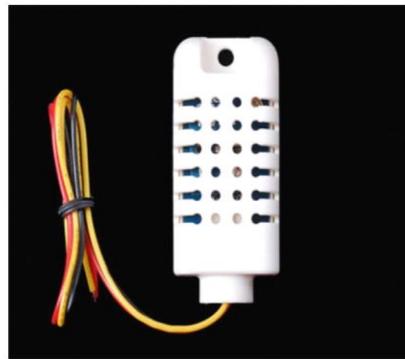
Product Name	Raspberry Pi Camera Module
Product Description	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
RS Part Number	913-2664
Specifications	
Image Sensor	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
Resolution	8-megapixel
Still picture resolution	3280 x 2464
Max image transfer rate	1080p: 30fps (encode and decode) 720p: 60fps
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
Image control functions	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
Temp range	Operating: -20° to 60° Stable image: -20° to 60°
Lens size	1/4"
Dimensions	23.86 x 25 x 9mm
Weight	3g

DHT22

Your specialist in innovating humidity & temperature sensors



Standard AM2302/DHT22



AM2302/DHT22 with big case and wires

Digital relative humidity & temperature sensor AM2302/DHT22

1. Feature & Application:

- *High precision
- *Capacitive type
- *Full range temperature compensated
- *Relative humidity and temperature measurement
- *Calibrated digital signal
- *Outstanding long-term stability
- *Extra components not needed
- *Long transmission distance, up to 100 meters
- *Low power consumption
- *4 pins packaged and fully interchangeable

2. Description:

AM2302 output calibrated digital signal. It applies exclusive digital-signal-collecting-technique and humidity sensing technology, assuring its reliability and stability. Its sensing elements are connected with 8-bit single-chip computer.

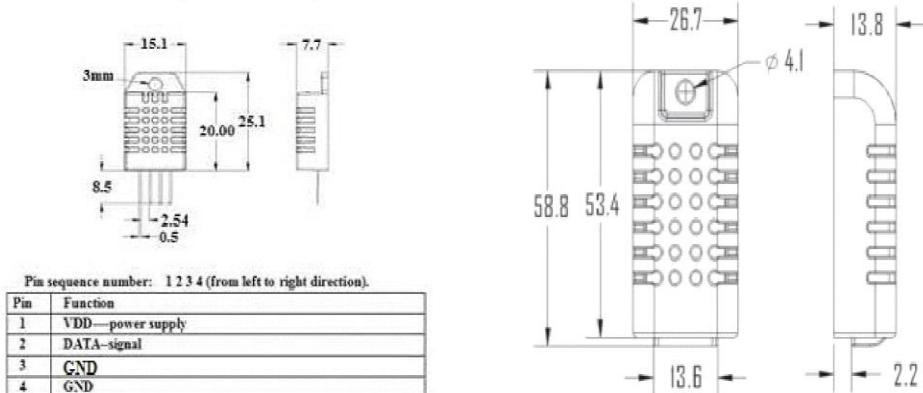
Every sensor of this model is temperature compensated and calibrated in accurate calibration chamber and the calibration-coefficient is saved in type of programme in OTP memory, when the sensor is detecting, it will cite coefficient from memory.

Small size & low consumption & long transmission distance(100m) enable AM2302 to be suited in all kinds of harsh application occasions. Single-row packaged with four pins, making the connection very convenient.

3. Technical Specification:

Model	AM2302
Power supply	3.3-5.5V DC
Output signal	digital signal via 1-wire bus
Sensing element	Polymer humidity capacitor
Operating range	humidity 0-100%RH; temperature -40~80Celsius
Accuracy	humidity +2%RH (Max +-5%RH); temperature +-0.5Celsius
Resolution or sensitivity	humidity 0.1%RH; temperature 0.1Celsius
Repeatability	humidity +-1%RH; temperature +-0.2Celsius
Humidity hysteresis	+-0.3%RH
Long-term Stability	+-0.5%RH/year
Interchangeability	fully interchangeable

4. Dimensions: (unit---mm)



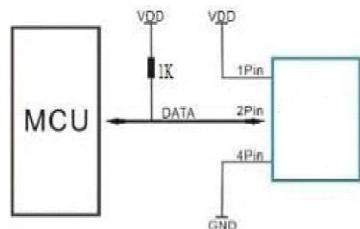
Standard AM2302's dimensions as above

Big case's dimensions as above

Red wire—power supply, Black wire—GND

Yellow wire—Data output

5. Electrical connection diagram:



6. Operating specifications:

(1) Power and Pins

Power's voltage should be 3.3-5.5V DC. When power is supplied to sensor, don't send any instruction to the sensor within one second to pass unstable status. One capacitor valued 100nF can be added between VDD and GND for wave filtering.

(2) Communication and signal

1-wire bus is used for communication between MCU and AM2302. (Our 1-wire bus is specially designed, it's different from Maxim/Dallas 1-wire bus, so it's incompatible with Dallas 1-wire bus.)

Illustration of our 1-wire bus:

DATA=16 bits RH data+16 bits Temperature data+8 bits check-sum

Example: MCU has received 40 bits data from AM2302 as

0000 0010 1000 1100 0000 0001 0101 1111 1110 1110

16 bits RH data 16 bits T data check sum

Here we convert 16 bits RH data from binary system to decimal system,

0000 0010 1000 1100 → 652

Binary system Decimal system

RH=652/10=65.2%RH

Here we convert 16 bits T data from binary system to decimal system,

0000 0001 0101 1111 → 351

Binary system Decimal system

T=351/10=35.1°C

When highest bit of temperature is 1, it means the temperature is below 0 degree Celsius.

Example: 1000 0000 0110 0101, T= minus 10.1°C

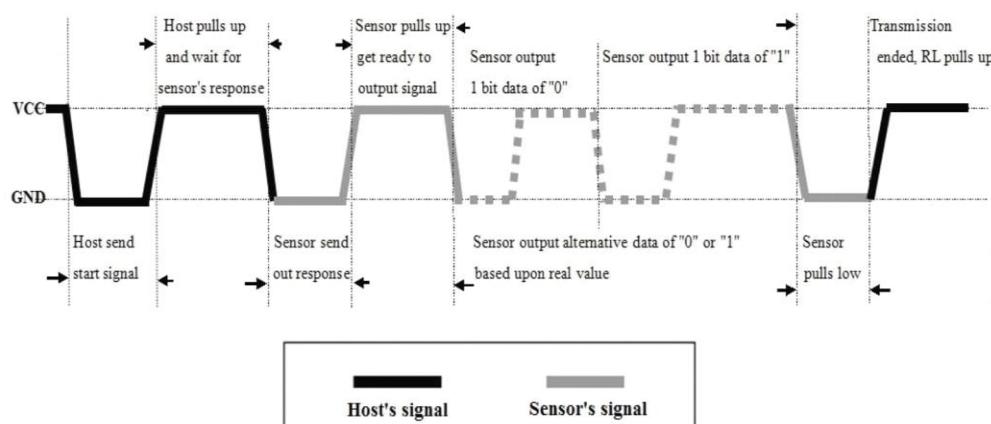
16 bits T data

Sum=0000 0010+1000 1100+0000 0001+0101 1111=1110 1110

Check-sum=the last 8 bits of Sum=1110 1110

When MCU send start signal, AM2302 change from standby-status to running-status. When MCU finishes sending the start signal, AM2302 will send response signal of 40-bit data that reflect the relative humidity and temperature to MCU. Without start signal from MCU, AM2302 will not give response signal to MCU. One start signal for one response data from AM2302 that reflect the relative humidity and temperature. AM2302 will change to standby status when data collecting finished if it don't receive start signal from MCU again.

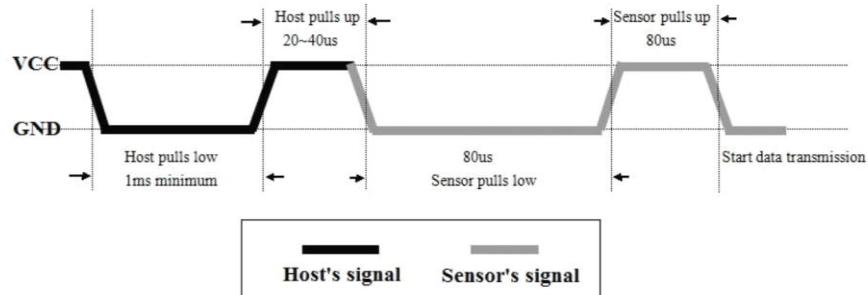
See below figure for overall communication process, **the interval of whole process must beyond 2 seconds**.



- 1) Step 1: MCU send out start signal to AM2302 and AM2302 send response signal to MCU

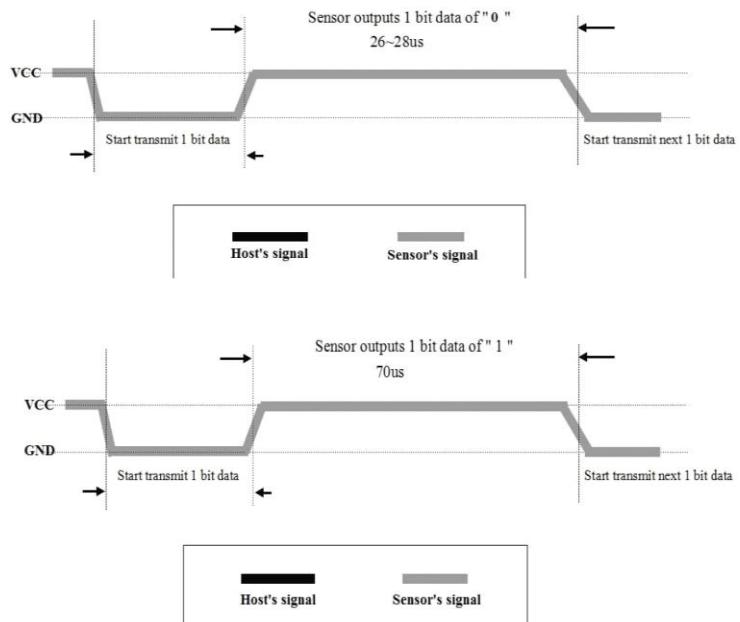
Data-bus's free status is high voltage level. When communication between MCU and AM2302 begins, MCU will pull low data-bus and this process must beyond at least 1~10ms to ensure AM2302 could detect MCU's signal, then MCU will pulls up and wait 20-40us for AM2302's response.

When AM2302 detect the start signal, AM2302 will pull low the bus 80us as response signal, then AM2302 pulls up 80us for preparation to send data. See below figure:



2). Step 2: AM2302 send data to MCU

When AM2302 is sending data to MCU, every bit's transmission begin with low-voltage-level that last 50us, the following high-voltage-level signal's length decide the bit is "1" or "0". See below figures:



Attention:

If signal from AM2302 is always high-voltage-level, it means AM2302 is not working properly, please check the electrical connection status.

7. Electrical Characteristics:

Items	Condition	Min	Typical	Max	Unit
Power supply	DC	3.3	5	6	V
Current supply	Measuring	1		1.5	mA
	Stand-by	40	Null	50	uA
Collecting period	Second		2		Second

8. Attentions of application:

(1) Operating and storage conditions

We don't recommend the applying RH-range beyond the range stated in this specification. The AM2302 sensor can recover after working in abnormal operating condition to calibrated status, but will accelerate sensors' aging.

(2) Attentions to chemical materials

Vapor from chemical materials may interfere AM2302's sensitive-elements and debase AM2302's sensitivity.

(3) Disposal when (1) & (2) happens

Step one: Keep the AM2302 sensor at condition of Temperature 50~60Celsius, humidity <10%RH for 2 hours;

Step two: After step one, keep the AM2302 sensor at condition of Temperature 20~30Celsius, humidity >70%RH for 5 hours.

(4) Attention to temperature's affection

Relative humidity strongly depend on temperature, that is why we use temperature compensation technology to ensure accurate measurement of RH. But it's still be much better to keep the sensor at same temperature when sensing.

AM2302 should be mounted at the place as far as possible from parts that may cause change to temperature.

(5) Attentions to light

Long time exposure to strong light and ultraviolet may debase AM2302's performance.

(6) Attentions to connection wires

The connection wires' quality will effect communication's quality and distance, high quality shielding-wire is recommended.

(7) Other attentions

* Welding temperature should be bellow 260Celsius.

* Avoid using the sensor under dew condition.

* Don't use this product in safety or emergency stop devices or any other occasion that failure of AM2302 may cause personal injury.

Solenoid Valve

2W Brass Series 2-Way Direct Acting Solenoid Valve Normally Closed

● The Professional Solenoid Valves Manufactory

2W 160 15 AC220V V

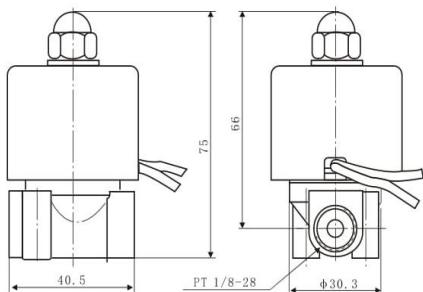
2 Way
Direct
Acting Orifice Pipe Size Available
Voltage
AC:110V
AC:220V
DC :24V
Contact the
Factory
for Others
Blank:NBR
V:For High Temp.



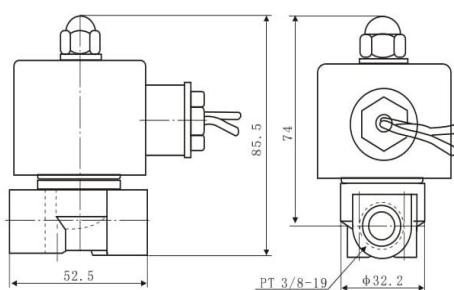
Specifications

Model code	2W025-06	2W025-08	2W040-10	2W160-10	2W160-15	2W200-20	2W250-25	2W350-35	2W400-40	2W500-50
Symbol										
Fluid Media	Air, Water, Oil, Gas									
Operating Mode	Direct Acting									
Type	Normally Closed									
Orifice	2.5	4	16	20	25	35	40	50		
Cv Factor	0.23	0.6	4.8	7.6	12	24	29	48		
Pipe Size	1/8"	1/4"	3/8"	3/8"	1/2"	3/4"	1"	1 1/4"	1 1/2"	2"
Viscosity	Under 20CTS									
Operating Pressure	Water:0~7 Air:0~7 Oil:0~5									
Fluids Temp.	-5~80°C									
Available Voltage	± 10%									
Body Material	Brass									
Seals Material	NBR or VITON									

Construction Dimensions Chart



2W025-06

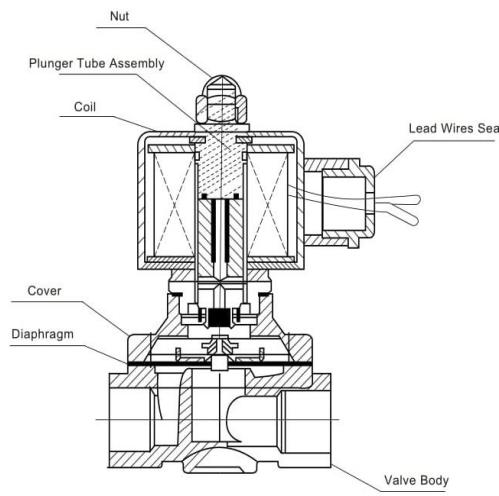
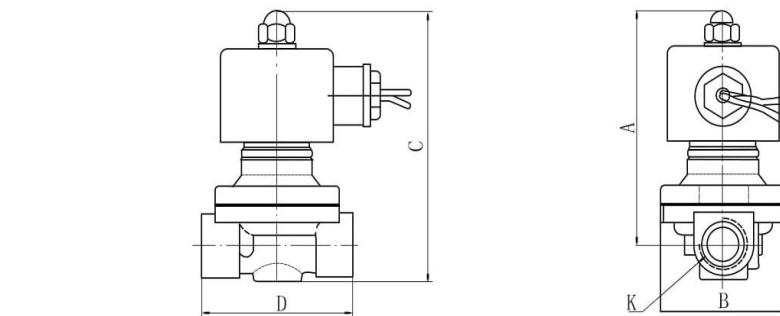


2W040-10

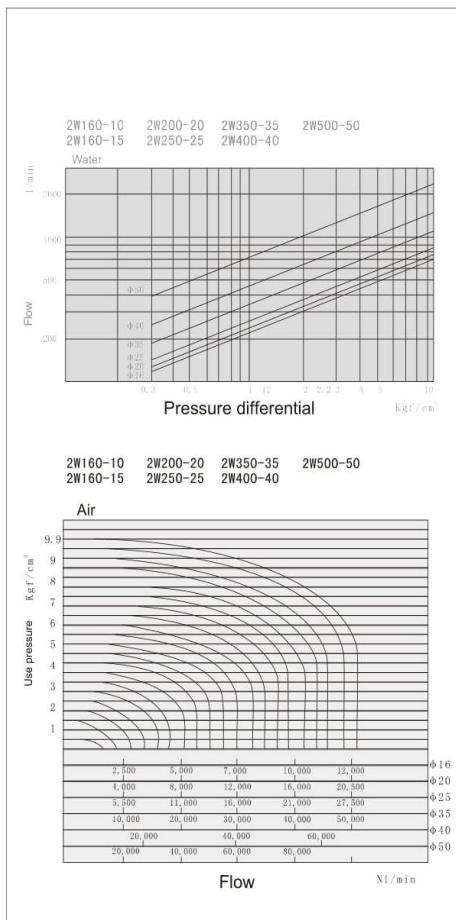
2W Brass Series 2-Way Direct Acting Solenoid Valve Normally Closed

●The Professional Solenoid Valves Manufactory

Construction Dimensions Chart



Fluid Flow Chart



Technical Parameter

Model code	A	B	C	D	K
2W160-10	101.5	57	117	69	PT 3/8"
2W160-15	101.5	57	117	69	PT 1/2"
2W260-20	107	57	123.5	73	PT 3/4"
2W260-25	111.5	73.5	134.5	99	PT 1"
2W360-35	142	95	172	112	PT1 1/4"
2W460-40	142	95	172	123	PT1 1/2"
2W560-50	172	123	209	168	PT2"

2W Stainless Steel Series 2-Way Direct Acting Solenoid Valve Normally Closed

● The Professional Solenoid Valves Manufactory

Technical Parameter

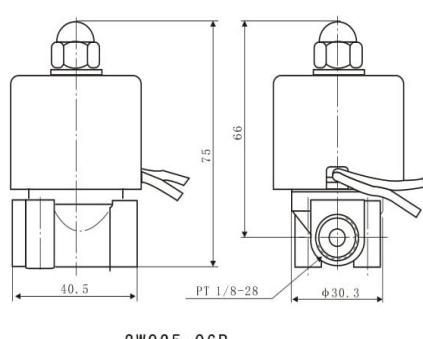
Model code	L	H	Pipe Size
2W025-06B	40.5	75	PT1/8"
2W025-08B	40.5	75	PT1/4"
2W040-10B	52.5	85.5	PT3/8"
2W160-10B	69	117	PT3/8"
2W160-15B	69	117	PT1/2"
2W200-20B	73	123.5	PT3/4"
2W250-25B	99	134.5	PT1"
2W350-35B	112	172	PT1 1/4"
2W400-40B	123	172	PT1 1/2"
2W500-50B	168	209	PT2"
2W250-25FB	135	160	Flange Connection 4 Holes
2W350-35FB	152	212	Flange Connection 4 Holes
2W400-40FB	152	215	Flange Connection 4 Holes
2W500-50FB	200	252	Flange Connection 4 Holes



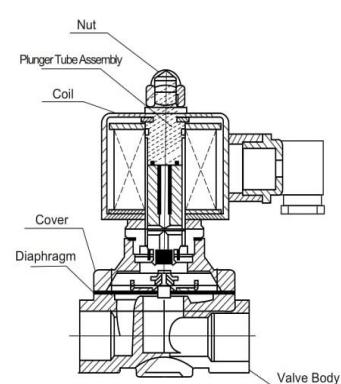
Specifications

Model code	2W025-06B	2W025-08B	2W040-10B	2W160-10B	2W160-15B	2W200-20B	2W250-25B	2W350-35B	2W400-40B	2W500-50B
Symbol										
Fluid Media						Air, Water, Oil, Gas				
Operating Mode						Direct Acting				
Type						Normally Closed				
Orifice	2.5	4	16	20	25	35	40	50		
Cv Factor	0.23	0.6	4.8	7.6	12	24	29	48		
Pipe Size	1/8"	1/4"	3/8"	3/8"	1/2"	3/4"	1"	1 1/4"	1 1/2"	2"
Viscosity						Under 20CTS				
Operating Pressure				Water:0~7	Air:0~70	Oil:0~5				
Fluids Temp.						-5~80°C				
Available Voltage				± 10%	AC220V	110V	DC24V	Contact the factory for Others		
Body Material						S.S.304 (S.S.316 Special Made)				
Seals Material						NBR or VITON				

Construction External Dimensions Chart



2W025-06B



2W Brass Series 2-Way Direct Acting Solenoid Valve Normally Open

●The Professional Solenoid Valves Manufactory

Characteristics

Normally open, open when de-energized
 Closed when energized
 Body material: forged brass
 They are capable of operating at zero differential pressure
 Available voltage:
 Voltage tolerance:



Inapplicable Fluids

Fluids that will turn to liquid after being heated and become solid after being cooled
 Strong corrosive fluids
 Fluids that have kinematic viscosity over 50CST

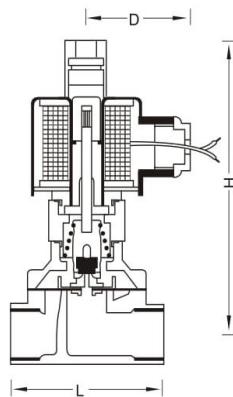
Attention

Make sure the pipe is clean before installing
 Pls fix a Y frame filter in front of the solenoid valve, for longer life-span

Technical Parameter

Model Code	L	H	Pipe Size
2W025-06H	40.5	110	1/8"
2W025-08H	40.5	110	1/4"
2W040-10H	52.5	115	3/8"
2W160-10H	69	135	3/8"
2W160-15H	69	135	1/2"
2W200-20H	73	142	3/4"
2W250-25H	99	150	1"
2W350-35H	112	186	PT1 1/4"
2W400-40H	123	197	PT1 1/2"
2W500-50H	168	225	PT2"

Construction Dimensions Chart



Specifications

Model Code	2W025-06H	2W025-08H	2W040-10H	2W160-10H	2W160-15H	2W200-20H	2W250-25H	2W350-35H	2W400-40H	2W500-50H
Fluid Media	Air, Water, Oil, Gas									
Operating Mode	Direct Acting									
Type	Normally Open									
Orifice	2.5	4	16	20	25	35	40	50		
Cv Rating	0.23	0.6	4.8	7.6	12	24	29	48		
Pipe Size	1/8"	1/4"	3/8"	3/8"	1/2"	3/4"	1"	1 1/4"	1 1/2"	2"
Viscosity	20 CTS以下									
Operating Pressure	Water:0~7 Air:0~7 Oil:0~5									
Fluids Temp.	-5~80°C									
Available Voltage	± 10%									
Body Material	Forged Brass									
Seals Material	NBR or VITON									

2W Stainless Steel Series 2-Way Direct Acting Solenoid Valve Normally Open

● The Professional Solenoid Valves Manufactory

Characteristics

Normally open, open when de-energized,
closed when energized
Body material: forged brass
They are capable of operating at zero differential pressure
Available voltage:
Voltage tolerance:

Inapplicable Fluids

Fluids that will turn to liquid after being heated and
become solid after being cooled
Strong corrosive fluids
Fluids that have kinematic viscosity over 50CST



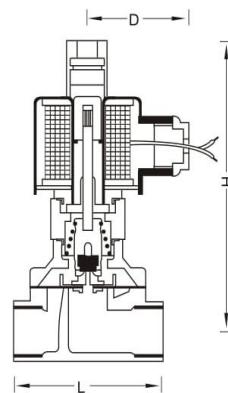
Attention

Make sure the pipe is clean before installing
Pls fix a Y frame filter in front of the solenoid valve,
for longer life-span

Technical Parameter

Model Code	L mm	H mm	Pipe Size
2W025-06BH	40.5	110	1/8"
2W025-08BH	40.5	110	1/4"
2W040-10BH	52.5	115	3/8"
2W160-10BH	69	135	3/8"
2W160-15BH	69	135	1/2"
2W200-20BH	73	142	3/4"
2W250-25BH	99	150	1"
2W350-35BH	112	186	PT1 1/4"
2W400-40BH	123	197	PT1 1/2"
2W500-50BH	168	225	PT2"
2W250-25FBH	135	160	Flange Connection 4 Holes
2W350-35FBH	152	230	Flange Connection 4 Holes
2W400-40FBH	152	240	Flange Connection 4 Holes
2W500-50FBH	200	270	Flange Connection 4 Holes

Construction Dimensions Chart



Specifications

Model Code	2W025-06BH	2W025-08BH	2W040-10BH	2W160-10BH	2W160-15BH	2W200-20BH	2W250-25BH	2W350-35BH	2W400-40BH	2W500-50BH
Fluid Media	Air, Water, Oil, Gas									
Operating Mode	Direct Acting									
Type	Normally Open									
Orifice	2.5	4	16	20	25	35	40	50		
Cv Rating	0.23	0.6	4.8	7.6	12	24	29	48		
Pipe Size	1/8"	1/4"	3/8"	3/8"	1/2"	3/4"	1"	1 1/4"	1 1/2"	2"
Viscosity	Under 20CTS									
Operating Pressure	Water:0~7 Air:0~7 Oil:0~5									
Fluids Temp.	-5~80°C									
Available Voltage	± 10%									
Body Material	S.S.304									
Seals Material	NBR or VITON									

การประมวลผลภาพด้วย MSE และ SSIM

MSE Vs SSIM

¹Swati A.Gandhi,²C.V.kulkarni

Abstract—IQA plays important role in digital image processing. It can be used to improve pictorial information or processing of data for transmission and representation. It can be done using FR,RR and NR methods depending on the availability of original and test image. In this paper concentrating on FR IQA methods using SSIM.

Index Terms— Digital image processing, Image Quality Assessment (IQA),Mean Square Error (MSE),Structural Similarity Index Matrices (SSIM). .

1.INTRODUCTION :



Fig.1 Basic Digital Image Processing System.

Digital image processing methods are used for two basic principal applications, such as first for improvement of pictorial information for human perception and second for processing of image data for storage, transmission and representation for autonomous machine perception.

An image may be defined as two dimensional function $f(x,y)$ where x and y are spatial coordinates and amplitude of f at any pair of coordinate (x,y) is called the intensity or gray level of image at that point when x , y and amplitude value of f are all finite value, is called as digital image. Digital image composed of finite number of elements where each have particular location and value called pixels. As shown in fig.(1), overall digital image processing is divided into three levels such as low level, medium level and high level of processing.

Low level processing involves source and transmitter. Source is nothing but image, audio or video .here we consider digital image only. At the transmitter side, preprocessing is done after sensing the object. Preprocessing involves primary operations such as noise reduction, contrast enhancement and image sharpening.

Medium level processing is done at communication channel.

- *1Swati.Ajay.Gandhi is currently pursuing masters degree program in electronics and telecommunication engineering inMIT COE, Universit of Pune, Pune,India, E-mail: swati.gandhi25@yahoo.com*
- *2C.V.Kulkarni,Department of E&TC,MIT COE,Pune,India, E-mail: cvk1971@gmail.com*

At this level, segmentation, storage and transmission is done on input image. Hence output from medium level processing is nothing but extracted features such as edges, lines, curves and identity of individual object. Finally higher level processing is on receiver side. Receiver reconstruct the information and on that basis reorganization or perception is done.

During all this ,image quality which is received at the end side get degraded. Human being are the good observer who can analyse image quality using principle of perception of vision. Image quality can be defined in terms of image fidelity that is perfect image. Image quality is degree to which image satisfy the naturalness and usefulness of image. Hence, image quality assessment plays important role in variety of applications. It can be used to in acquisition and display system to monitor image quality. It can be used as benchmarking in different compression algorithm.

2 IMAGE QUALITY ASSESSMENT METHODS:

Image quality assessment methods (IQA) can be widely done using two methods namely subjective method and objective method as shown in fig.(2). In subjective analysis, group of observers are present to analyse the perceived image quality. Then overall quality score is calculated using mean

opinion score(MOS) and differential mean opinion score (DMOS) is calculated. On that basis, quantitative analysis of image is done. Some criteria are consider during analysis such as viewing distance, angle, period, background and knowledge about image processing. Though it is more accurate for evaluating quality of perceived image still it is complicated and time consuming.

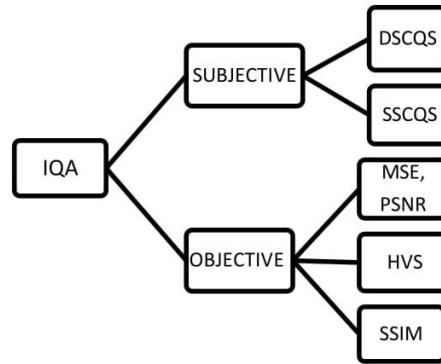


Fig.2: Image Quality Assessment Methods

In objective analysis is done for automatically evaluate the quality matrices using mathematical calculation. Depending on availability of reference image, it can be classified into full reference FR, reduced reference RR and no/blind reference NR IQA.

- Full Reference (FR) IQA-Both test and original images are present.
- No Reference(NR) IQA-Only test image is present.
- Reduced Reference(RR) IQA-Partial information is available.(A set of extracted features made available.)

Most o f IQA methods are derived using full reference method. Here ,SSIM based full reference IQA is explained

3.MEAN SQUARE ERROR (MSE):

Traditional and simple method for measuring the energy of error signal in test image. square of difference between error of original and test /distorted image is calculated.

Two signals are compared pixel by pixel from left to right and top to bottom through a row and column. Then calculate by averaging square of difference between error of original and test /distorted image.

If x and y are two non negative gray scale images,

the1n MSE is calculated using

$$MSE = \frac{1}{N} \sum_{i=1}^N |x_i - y_i|^2 \quad (1)$$

Peak signal to noise ratio is calculated as follow,

$$PSNR = 10 \log_{10} \left(\frac{m^2}{MSE} \right) \text{ db.} \quad (2)$$

where m are maximum gray levels of 8 bits/pixel of image.(Here, m=100 considered otherwise m=255 for 8 bits/pixel).

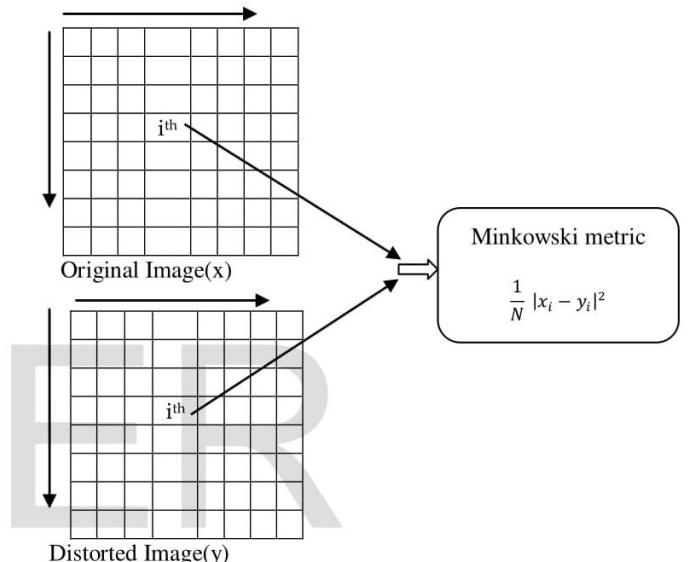


Fig.3: Minkowski Metric For calculating MSE

Two signals are compared pixel by pixel from left to right and top to bottom through a row and column. Then calculate by averaging square of difference between error of original and test /distorted image.

3.1 PROPERTIES:

- [1] simple and parameter independent.
- [2] Square error calculated ,is independent of other sample.
- [3] $MSE \geq 0$.
- [4] $MSE = 0$ iff original signal(x)=test signal(y).
- [5] Clear physical meaning.

MSE value is small or equal to zero, indicate minimum or zero distortion.MSE widely used in variety of signal processing applications such as denoising, reconstruction, classification, restoration, filter designing.

3.2 DISADVANTAGES:

- Major disadvantage of MSE is poorly correlate with human perception of visual system.
- According to human visual system HVS, error visibility is correlate with loss of quality. But ,MSE totally mismatch with this because some distortions are not clearly visible and some are present but not disturbing the image quality.
- All images with equal MSE, does not mean that all contain same distortions or noise.

4. STRUCTURAL SIMILARITY INDEX MATRICES(SSIM):

A new approach proposed to overcome this problem which provide solutions which are independent of visibility conditions and threshold problems. Main aim to extract structural information from image. Structural Similarity Index Matrix (SSIM) separate out the three parameter such as luminance, contrast and structure which are independent of each other and are highly structured.

If consider two non negative images x and y where x is original discrete signal and y is distorted discrete signal, then

$$\text{SSIM}(x,y)=f[l(x,y),c(x,y),s(x,y)] \quad (3)$$

luminance $l(x,y)$ is given by

$$l(x,y)=\frac{(2\mu_x\mu_y+C1)}{(\mu_x^2+\mu_y^2+C1)} \quad (4)$$

where original signal mean intensity $\mu_x=\frac{1}{N}\sum_{i=1}^Nx_i$, distorted signal mean intensity $\mu_y=\frac{1}{N}\sum_{i=1}^Ny_i$ and C1 is constant added to avoid instability when $(\mu_x^2+\mu_y^2)$ is very close to zero and equal to $(K_1l)^2$. l is dynamic range of pixel values (255 for 8 bit gray scale image) and $K_1 \ll 1$.

According to Webbers law, the magnitude of just noticeable luminance change Δl is approximately proportional to background luminance I for wide range of luminance values. Let R is change relative to background luminance, we rewrite luminance of distorted signal as

$$\mu_y = (1 + R) \mu_x. \quad (5)$$

Substituting in above equation(4),

$$l(x,y)=\frac{2(1+R)}{1+(1+R^2+\frac{C1}{\mu_x^2})} \quad (6)$$

If we assume C1 is small enough compare to μ_x^2 ,then above equation is function of only R and consistent with webbers law.

Contrast $c(x,y)$ is given by

$$c(x,y)=\frac{(2\sigma_x\sigma_y+C2)}{(\sigma_x^2+\sigma_y^2+C2)} \quad (7)$$

where σ_x , standard deviation as an estimate of signal contrast by subtracting mean intensity from signal.

$$\sigma_x = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)^2 \quad (8)$$

$$C2=(K_2l)2 \text{ and } K_2 \ll 1 \text{ as small count.}$$

With same amount of contrast change $\Delta\sigma = \sigma_y - \sigma_x$, this is less sensitive to high base contrast σ_x than low base contrast proves contrast masking feature in human visual system(HVS).

Structure is calculated after subtracting luminance and variance normalization, we associate with two unit vectors given by $\frac{(x-\mu_x)}{\sigma_x}$ and $\frac{(y-\mu_y)}{\sigma_y}$.Structure $s(x,y)$ is calculated as follows:

$$s(x,y)=\frac{\sigma_{xy}+C3}{\sigma_x\sigma_y+C3} \quad (9)$$

where C3 is small constant and

$$\sigma_{xy} = \frac{1}{N-1} \sum_{i=1}^N (x_i - \mu_x)(y_i - \mu_y) \quad (10)$$

Hence, resultant SSIM(x,y) index is given by

$$\text{SSIM}(x,y)=\frac{(2\mu_x\mu_y+C1)(2\sigma_x\sigma_y+C2)}{(\mu_x^2+\mu_y^2+C1)(\sigma_x^2+\sigma_y^2+C2)} \quad (11)$$

where all three parameters are highly structured and relatively independent.

Mean SSIM (MSSIM) is calculated to evaluate overall image quality is given by

$$\text{MSSIM}(x,y)=\frac{1}{M} \sum_{j=1}^M \text{SSIM}(x_j,y_j) \quad (11)$$

4.1 PROPERTIES OF SSIM:

- Symmetry , $S(x,y)=S(y,x)$
- Unique maximum, $S(x,y)=1$ iff $x=y$
- Boundedness, $S(x,y)\leq 1$

5. EXPERIMENTAL RESULTS:

A original image of harbour is taken, which is distorted with 4 different types of distortion

such as blurring, gaussian noise, JPG and JPG2K . All distorted images set with equal value of MSE. SSIM and error map is calculated for these distorted image along with original image. It proves that the matrices calculated is independent of visibility of error.

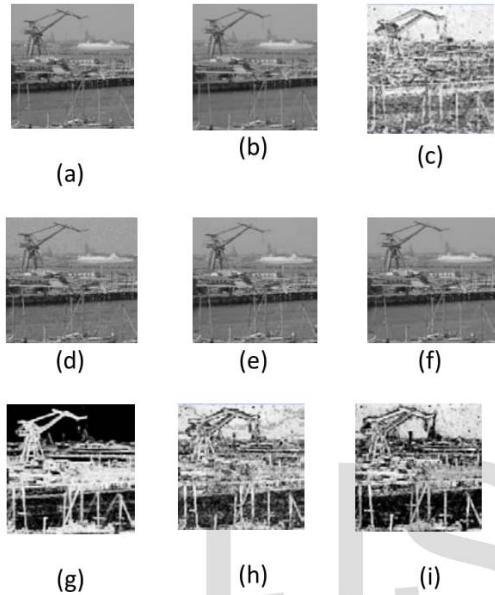


Fig. 4.Comparision of "harbor" images with different types of distortions and their SSIM error map.
 a)Original image, MSE=0;PSNR=INF;MSSIM=1;b)Blurr image;MSE=83.61;PSNR=27.93; MSSIM=0.909; d) Noisy image;MSE=87.23;PSNR=28.68; MSSIM=0.734; e)JPEG image;MSE=83.58;PSNR=28.90; MSSIM=0.877; f) JPG2 image;MSE=85.88;PSNR=28.79; MSSIM=0.834; c),g),h) and i)SSIM error map of Blurr image, Noisy image,JPG image and JPG2 repectively.

CONCLUSION:

Most of digital image processing application where IQA plays important role. SSIM totally depends on structural features which are extracted from image. Luminance, contrast and structure ,all parameters are independent. SSIM error map shows area which is more

affected by noise. Hence it is easy to reconstruct the distorted image .

REFERENCES

- [1] K. Moorthy, Z. Wang and A. C. Bovik, "Visual perception and quality assessment," in Optical and Digital Image Processing (Cristobal, Schelkens and Thienpont, eds.), Wiley publisher, 2010.
- [2] K. Seshadrinathan, T. N. Pappas, R. J. Safranek, J. Chen, Z. Wang, H. R. Sheikh and A. C. Bovik, "Image quality assessment," in Essential Guide to Image Processing, Elsevier, 2009.
- [3] Z. Wang, A. C. Bovik and H. R. Sheikh, "Structural similarity based image quality assessment," in Digital Video Image Quality and Perceptual Coding (H. R. Wu, and K. R. Rao, eds.), Marcel Dekker Series in Signal Processing and Communications, Nov. 2005.
- [4] Z. Wang, A. C. Bovik and E. P. Simoncelli, "Structural approaches to image quality assessment," in Handbook of Image and Video Processing (Al Bovik, eds.), second edition, Academic Press, May 2005.
- [5] Z.Wang,A.C.Bovik,H.R.Shekikh, and E.P.Simoncelli,"Image quality assessment:From error visibility to structural similarity,"IEEE transactions on Image Processing,vol.13,no.4,Apr.
- [6] Z.Wang,A.C.Bovik and L.Lu,"Why is image quality assessment so difficult?"IEEE International 2004Conference on Acoustics,speech and Signal Processing,May 2002.
- [7] Z.Wang, "Application of Objective Image Quality Assessment Methods,"IEEE SIGNAL PROCESSING MAGAZINE,pp-137-142,Nov. 2011.
- [8] Kim-Han Thung,Paramesran Raveendran,"A Survey Of Image Quality Measures,"TECHPOS,International Conference,pp-1-4,Dec.2009
- [9] LIVE, <http://live.ece.utexas.edu>
- [10] "Z.Wang Plublication Home Page", https://ece.uwaterloo.ca/~z70wang/publications.htm#journal_papers

ภาคผนวก ง.

โปรแกรมควบคุม

ໂປຣແກຣມສໍາໜັບການປະກາດໄລບຣາຣີແລະອິນພູຕ-ເອາດີພູດຂອງໂປຣແກຣມ

```
# import the necessary packages
import Tkinter as tk
from Tkinter import *
import sys
from time import sleep
from skimage.measure import compare_ssim as ssim
import matplotlib.pyplot as plt
import numpy as np
import cv2
import picamera
import Adafruit_DHT
import RPi.GPIO as GPIO

#####
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

sensor = Adafruit_DHT.DHT22
pin = 4

#####
# input #####
GPIO.setup(17, GPIO.IN, pull_up_down = GPIO.PUD_UP) #bottom
GPIO.setup(27, GPIO.IN, pull_up_down = GPIO.PUD_UP) #top
GPIO.setup(22, GPIO.IN, pull_up_down = GPIO.PUD_UP) #mid
GPIO.setup(26, GPIO.IN, pull_up_down = GPIO.PUD_UP) #limit switch
#####
# output #####
GPIO.setup(23, GPIO.OUT)      #solenoid_in
GPIO.output(23,0)
GPIO.setup(24, GPIO.OUT)      #solenoid_out
GPIO.output(24,0)
GPIO.setup(7, GPIO.OUT)       #Buzzer
GPIO.output(7,GPIO.HIGH)
GPIO.setup(4, GPIO.OUT)       #DHT22
GPIO.output(4,0)
GPIO.setup(20, GPIO.OUT)      #Dry_Hot
GPIO.output(20,0)
GPIO.setup(21, GPIO.OUT)      #motor_shoes
GPIO.output(21,0)
GPIO.setup(12, GPIO.OUT)      #Blower
GPIO.output(12,GPIO.HIGH)
```

ໂປຣແກຣມສໍາຫັບການປະໜວລຜລກພາພ

```
def mse(imageA, imageB):
    # the 'Mean Squared Error' between the two images is the
    # sum of the squared difference between the two images;
    # NOTE: the two images must have the same dimension
    err = np.sum((imageB.astype("float") - imageA.astype("float")) ** 2)
    err /= float(imageB.shape[0]*imageB.shape[1])
    # return the MSE, the lower the error, the more "similar"
    # the two images are
    return err

def compare_images(imageA, imageB, title):
    # compute the mean squared error and structural similarity
    # index for the images
    m = mse(imageA, imageB)
    s = ssim(imageA, imageB)

    if s > 0.8 :
        self.after(1000, self.run1) #back to clean system
    else:
        self.after(1000, self.Dry40) #out of system

    # setup the figure
    fig = plt.figure(title)
    plt.suptitle("MSE: %.2f, SSIM: %.2f" % (m, s))

    # show first image
    ax = fig.add_subplot(1, 2, 1)
    plt.imshow(imageA, cmap = plt.cm.gray)
    plt.axis("off")

    # show the second image
    ax = fig.add_subplot(1, 2, 2)
    plt.imshow(imageB, cmap = plt.cm.gray)
    plt.axis("off")

    # show the images
    plt.show()

def compare(self):
    before = cv2.imread("image1.jpg")
    after = cv2.imread("image2.jpg")
```

```

# convert the images to grayscale
before = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
after = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

# show the figure
plt.show()

# compare the images
compare_images(before, after, 'Before vs. After')

```

ໂປຣແກຣມສໍາຮັບຄ່າຍກາພນີ້ງ

```

def camera1(self):
    while True:
        if GPIO.input(26) == 0:
            GPIO.output(21,0)
            print("About to take a picture.")
            with picamera.PiCamera() as camera:
                camera.resolution = (500,500)
                camera framerate = 24
                sleep(3)
                camera.capture("/home/pi/image1.jpg")
            print("Picture taken.")
            break
        else:
            GPIO.output(21,1)
            self.after(2000, self.showIm1)

def camera2(self):
    while True:
        if GPIO.input(26)==0:
            GPIO.output(21,0)
            print("About to take a picture.")
            with picamera.PiCamera() as camera:
                self.camera.resolution = (500,500)
                camera framerate = 24
                sleep(3)
                self.camera.capture("/home/pi/image2.jpg")
            print("Picture taken.")
            break
        else:
            GPIO.output(21,1)
            self.after(2000, self.showIm2)

def camera2_2(self):

```

```

while True:
    if GPIO.input(26)==0:
        GPIO.output(21,0)
        print("About to take a picture.")
        with picamera.PiCamera() as camera:
            self.camera.resolution = (500,500)
            camera.framerate = 24
            sleep(3)
            self.camera.capture("/home/pi/image2.jpg")
            print("Picture taken.")
            break
    else:
        GPIO.output(21,1)
        self.after(2000, self.showIm2_2)

```

ໂປຣແກຣມສໍາຫຼັບໂຈ່ງກາພນິຈ

```

def showIm1(self):
    img = cv2.imread('image1.jpg') #show image
    cv2.imshow('image',img)
    cv2.waitKey(5000)
    cv2.destroyAllWindows()
    self.after(100, self.solenoid_in)

def showIm2(self):
    img = cv2.imread('image2.jpg') #show image
    cv2.imshow('image',img)
    cv2.waitKey(5000)
    cv2.destroyAllWindows()
    self.after(100, self.compare)

def showIm2_2(self):
    img = cv2.imread('image2.jpg') #show image
    cv2.imshow('image',img)
    cv2.waitKey(5000)
    cv2.destroyAllWindows()
    self.after(100, self.compare1)

```

ໂປຣແກຣມສໍາหารັບຕຽບຈຳກວດຕັບນ້ຳເຂົາ-ນ້ຳອອກຕ້າຍລຸກລອຍ

```
def solenoid_in(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break
        elif GPIO.input(27)==0: #top
            GPIO.output(23,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)
                sleep(.5)
                GPIO.output(7,GPIO.HIGH)
                sleep(.5)
                x += 1
            break
        self.after_cancel(self.solenoid_in)
    else:
        print("NOT FULL")
        GPIO.output(23,1)
    self.after(100, self.clean)

def solenoid_out(self):
    GPIO.output(24,1)
    while True:
        if GPIO.input(17) == 1: #bottom
            GPIO.output(24,0)
            break
        elif GPIO.input(22) == 0: #mid
            GPIO.output(24,1)
        else:
            GPIO.output(24,1)
    self.after(100, self.solenoid_in1)
```

ໂປຣແກຣມສໍາຫັບການອອກແບບ GUI

```
class SampleApp(tk.Tk):

    def __init__(self):
        tk.Tk.__init__(self)

        self._frame = None
        self.switch_frame(StartPage)

    def switch_frame(self, frame_class):
        """Destroys current frame and replaces it with a new one."""
        new_frame = frame_class(self)
        if self._frame is not None:
            self._frame.destroy()
        self._frame = new_frame
        self._frame.pack()

class StartPage(tk.Frame):

    def __init__(self, master):
        tk.Frame.__init__(self, master)

        start_label = tk.Label(self, text="Please Choose Mode",fg="#000",font="times 30 bold",width=30,height=2)
        page_1_button = tk.Button(self, text="AUTO MODE",fg="#FFF",bg="black",font="times 24 bold",height=2,
                                command=lambda: master.switch_frame(PageOne))
        page_2_button = tk.Button(self, text="MANUAL MODE",fg="#FFF",bg="black",font="times 24 bold",height=2,
                                command=lambda: master.switch_frame(PageTwo))
        quit_button = tk.Button(self, text="Quit",fg="#FFF",bg="black",font="times 24 bold",height=2,
                               command=self.quit)

        start_label.pack(side="top", fill="x", pady=10)
        page_1_button.pack(side="left",fill="x", pady=20)
        page_2_button.pack(side="right",fill="x", pady=20)
        quit_button.pack(side="bottom",fill="x", pady=20)

    def quit(self):
        self.master.destroy()

#####
#####Auto Mode#####
class PageOne(tk.Frame):

    def __init__(self, master):
        tk.Frame.__init__(self, master)
```

```

page_1_label = tk.Label(self, text="AUTO MODE",fg="#000",font="times 25
bold",width=30,height=2)
start_button = tk.Button(self, text="START",fg="#000",bg="red",font="times 24
bold",width=15,height=1,
command=lambda: self.update("START"))
start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
command=lambda: master.switch_frame(StartPage))
page_1_label.pack(side="top", fill="x", pady=5)
start_button.pack(side="top", anchor=N , fill="x", pady=30)
start2_button.pack(side="right")

#####
Manual Mode#####
class PageTwo(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        page_2_label = tk.Label(self, text="MANUAL MODE",fg="#000",font="times 25
bold",width=30,height=2)
        start_button = tk.Button(self, text="WASH",fg="#000",bg="cyan",font="times 24
bold",width=15,height=2,
command=lambda: master.switch_frame(PageWash))
        start1_button = tk.Button(self, text="DRY",fg="#000",bg="gray",font="times 24
bold",width=15,height=2,
command=lambda: master.switch_frame(PageDry))
        start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
command=lambda: master.switch_frame(StartPage))
        page_2_label.pack(side="top", fill="x", pady=10)
        start_button.pack(side="left", anchor=N , fill="x", pady=30)
        start1_button.pack(side="top", fill="x", pady=30)
        start2_button.pack(side="right")

class PageWash(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key",font="times 30",width=15, validatecommand=(vcmd, "%P"))
        page_2_label = tk.Label(self, text="TIMES (min)",fg="#000",font="times 25 bold",width=30,height=2)
        start_button = tk.Button(self, text="START",bg="RED",font="times 20 bold",width=11,

```

```

        command=lambda: self.update("START"))

reset_button = tk.Button(self, text="Reset",bg="white",font="times 20 bold",width=11,
                        command=lambda: self.update("reset"))

start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
                        command=lambda: master.switch_frame(StartPage))

page_2_label.pack(side="top", fill="x", pady=10)
self.entry.pack(side="left",anchor=N,fill="x")
start_button.pack(side="left", anchor=N)
reset_button.pack(side="top",anchor=W)
start2_button.pack(side="right",fill="x",pady=30)
GPIO.cleanup()

class PageDry(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        page_2_label = tk.Label(self, text="TEMPERATURE",fg="#000",font="times 25
bold",width=30,height=2)
        start_button = tk.Button(self, text="40 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
                                command=lambda: master.switch_frame(PageDry40))
        start1_button = tk.Button(self, text="45 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
                                command=lambda: master.switch_frame(PageDry45))
        start2_button = tk.Button(self, text="50 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
                                command=lambda: master.switch_frame(PageDry50))
        start3_button = tk.Button(self, text="55 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
                                command=lambda: master.switch_frame(PageDry55))
        start4_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
                                command=lambda: master.switch_frame(StartPage))

        page_2_label.pack(side="top", fill="x", pady=10)
        start_button.pack(side="left", anchor=N , fill="x", pady=30)
        start1_button.pack(side="left", anchor=N , fill="x", pady=30)
        start2_button.pack(side="left", anchor=N , fill="x", pady=30)
        start3_button.pack(side="top", fill="x", pady=30)
        start4_button.pack(side="right")

class PageDry40(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

```

```

self.total_label_text = IntVar()
self.total_label = Label(master, textvariable=self.total_label_text)

vcmd = master.register(self.validate) # we have to wrap the command
self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, '%P'))
page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                        command=lambda: self.update("START"))
reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,
                        command=lambda: self.update("reset"))
start2_button = tk.Button(self, text="HOME", fg="#FFF", bg="black", font="times 24 bold", width=10,
                         command=lambda: master.switch_frame(StartPage))
page_2_label.pack(side="top", fill="x", pady=10)
self.entry.pack(side="left", anchor=N, fill="x")
start_button.pack(side="left", anchor=N)
reset_button.pack(side="top", anchor=W)
start2_button.pack(side="right", fill="x", pady=30)

class PageDry45(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, '%P'))
        page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
        start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                                command=lambda: self.update("START"))
        reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,
                                command=lambda: self.update("reset"))
        start2_button = tk.Button(self, text="HOME", fg="#FFF", bg="black", font="times 24 bold", width=10,
                                 command=lambda: master.switch_frame(StartPage))
        page_2_label.pack(side="top", fill="x", pady=10)
        self.entry.pack(side="left", anchor=N, fill="x")
        start_button.pack(side="left", anchor=N)
        reset_button.pack(side="top", anchor=W)
        start2_button.pack(side="right", fill="x", pady=30)

class PageDry50(tk.Frame):

```

```

def __init__(self, master):
    tk.Frame.__init__(self, master)

    self.entered_number = 0

    self.total_label_text = IntVar()
    self.total_label = Label(master, textvariable=self.total_label_text)

    vcmd = master.register(self.validate) # we have to wrap the command
    self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, "%P"))
    page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
    start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                           command=lambda: self.update("START"))
    reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,
                           command=lambda: self.update("reset"))
    start2_button = tk.Button(self, text="HOME", fg="#FFF", bg="black", font="times 24 bold", width=10,
                           command=lambda: master.switch_frame(StartPage))
    page_2_label.pack(side="top", fill="x", pady=10)
    self.entry.pack(side="left", anchor=N, fill="x")
    start_button.pack(side="left", anchor=N)
    reset_button.pack(side="top", anchor=W)
    start2_button.pack(side="right", fill="x", pady=30)

class PageDry55(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, "%P"))
        page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
        start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                               command=lambda: self.update("START"))
        reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,
                               command=lambda: self.update("reset"))
        start2_button = tk.Button(self, text="HOME", fg="#FFF", bg="black", font="times 24 bold", width=10,
                               command=lambda: master.switch_frame(StartPage))
        page_2_label.pack(side="top", fill="x", pady=10)
        self.entry.pack(side="left", anchor=N, fill="x")
        start_button.pack(side="left", anchor=N)

```

```
reset_button.pack(side="top",anchor=W)
start2_button.pack(side="right",fill="x",pady=30)
```

ໂປຣແກຣມລຳຫຽວບະບົບທຳໃຫ້ແໜ່ງ

```
def Dry40(self):
    x=0
    while(x<300):
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        if humidity is not None and temperature is not None:
            print ('Temp={0:0.1f}*C  Humidity={1:0.1f}%.format(temperature, humidity))
            sleep(3)
            t1=temperature
            temp1=t1
            if t1 < 40 :
                GPIO.output(20,1)
                GPIO.output(12,GPIO.LOW)
            elif t1 > 40:
                GPIO.output(20,0)
                GPIO.output(12,GPIO.LOW)
            else:
                GPIO.output(20,0)
                GPIO.output(12,GPIO.HIGH)
        else:
            print ('Failed to get reading. Try again!')
            self.master.update()
            x += 1
    else:
        GPIO.output(20,0)
        GPIO.output(12,GPIO.HIGH)
        GPIO.cleanup()
        self.after(1000, self.finish)
```

ໂປຣແກຣມຮັງມ່ານທີ່ໜຸດ

```

# import the necessary packages
import Tkinter as tk
from Tkinter import *
import sys
from time import sleep
from skimage.measure import compare_ssim as ssim
import matplotlib.pyplot as plt
import numpy as np
import cv2
import picamera
import Adafruit_DHT
import RPi.GPIO as GPIO

#####
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

sensor = Adafruit_DHT.DHT22
pin = 4

#####
GPIO.setup(17, GPIO.IN, pull_up_down = GPIO.PUD_UP) #bottom
GPIO.setup(27, GPIO.IN, pull_up_down = GPIO.PUD_UP) #top
GPIO.setup(22, GPIO.IN, pull_up_down = GPIO.PUD_UP) #mid
GPIO.setup(26, GPIO.IN, pull_up_down = GPIO.PUD_UP) #limit switch
#####
GPIO.setup(23, GPIO.OUT)      #solenoid_in
GPIO.output(23,0)
GPIO.setup(24, GPIO.OUT) #solenoid_out
GPIO.output(24,0)
GPIO.setup(7, GPIO.OUT) #Buzzer
GPIO.output(7,GPIO.HIGH)
GPIO.setup(4, GPIO.OUT) #DHT22
GPIO.output(4,0)
GPIO.setup(20, GPIO.OUT)      #Dry_Hot
GPIO.output(20,0)
GPIO.setup(21, GPIO.OUT)      #motor_shoes
GPIO.output(21,0)
GPIO.setup(12, GPIO.OUT)      #Blower
GPIO.output(12,GPIO.HIGH)

class SampleApp(tk.Tk):
    def __init__(self):

```

```

tk.Tk.__init__(self)

    self._frame = None
    self.switch_frame(StartPage)


def switch_frame(self, frame_class):
    """Destroys current frame and replaces it with a new one."""
    new_frame = frame_class(self)
    if self._frame is not None:
        self._frame.destroy()
    self._frame = new_frame
    self._frame.pack()

class StartPage(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        start_label = tk.Label(self, text="Please Choose Mode",fg="#000",font="times 30 bold",width=30,height=2)
        page_1_button = tk.Button(self, text="AUTO MODE",fg="#FFF",bg="black",font="times 24 bold",height=2,
                                command=lambda: master.switch_frame(PageOne))
        page_2_button = tk.Button(self, text="MANUAL MODE",fg="#FFF",bg="black",font="times 24 bold",height=2,
                                command=lambda: master.switch_frame(PageTwo))
        quit_button = tk.Button(self, text="Quit",fg="#FFF",bg="black",font="times 24 bold",height=2,
                               command=self.quit)

        start_label.pack(side="top", fill="x", pady=10)
        page_1_button.pack(side="left",fill="x", pady=20)
        page_2_button.pack(side="right",fill="x", pady=20)
        quit_button.pack(side="bottom",fill="x", pady=20)

    def quit(self):
        self.master.destroy()

#####
class PageOne(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        page_1_label = tk.Label(self, text="AUTO MODE",fg="#000",font="times 25 bold",width=30,height=2)

```

```

start_button = tk.Button(self, text="START",fg="#000",bg="red",font="times 24 bold",width=15,height=1,
command=lambda: self.update("START"))

start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
command=lambda: master.switch_frame(StartPage))

page_1_label.pack(side="top", fill="x", pady=5)
start_button.pack(side="top", anchor=N , fill="x", pady=30)
start2_button.pack(side="right")

def update(self, method):
    if method == "START":
        self.after(100,self.run)
    else:
        self.after_cancel(self.run)

#####
#####AUTO1#####
def run(self):
    print("START")
    self.after(500, self.camera1)

def camera1(self):
    while True:
        if GPIO.input(26) == 0:
            GPIO.output(21,0)
            print("About to take a picture.")
            with picamera.PiCamera() as camera:
                camera.resolution = (500,500)
                camera.framerate = 24
                sleep(3)
                camera.capture("/home/pi/image1.jpg")
            print("Picture taken.")
            break
        else:
            GPIO.output(21,1)
    self.after(2000, self.showIm1)

def solenoid_in(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break

```

```

        elif GPIO.input(27)==0: #top
            GPIO.output(23,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)
                sleep(.5)
                GPIO.output(7,GPIO.HIGH)
                sleep(.5)
                x += 1
            break
        self.after_cancel(self.solenoid_in)
    else:
        print("NOT FULL")
        GPIO.output(23,1)
        self.after(100, self.clean)

def clean(self):
    GPIO.output(21,1)
    self.after(180000, self.solenoid_out)

def solenoid_out(self):
    GPIO.output(24,1)
    while True:
        if GPIO.input(17) == 1: #bottom
            GPIO.output(24,0)
            break
        elif GPIO.input(22) == 0: #mid
            GPIO.output(24,1)
        else:
            GPIO.output(24,1)
    self.after(100, self.solenoid_in1)

def solenoid_in1(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break
        elif GPIO.input(27)==0: #top
            GPIO.output(23,0)
            x=0
            while(x<5):

```

```

        GPIO.output(7,GPIO.LOW)
        sleep(.5)
        GPIO.output(7,GPIO.HIGH)
        sleep(.5)
        x += 1
        break
    self.after_cancel(self.solenoid_in1)
else:
    print("NOT FULL")
    GPIO.output(23,1)
self.after(100, self.clean_w)

def clean_w(self):
    GPIO.output(21,1)
    self.after(180000, self.solenoid_out1)

def solenoid_out1(self):
    GPIO.output(24,1)
    while True:
        if GPIO.input(17) == 1:    #bottom
            GPIO.output(24,0)
            break
        elif GPIO.input(22) == 0:  #mid
            GPIO.output(24,1)
        else:
            GPIO.output(24,1)
    self.after(100, self.camera2)

def camera2(self):
    while True:
        if GPIO.input(26)==0:
            GPIO.output(21,0)
            print("About to take a picture.")
            with picamera.PiCamera() as camera:
                self.camera.resolution = (500,500)
                camera.framerate = 24
                sleep(3)
                self.camera.capture("/home/pi/image2.jpg")
                print("Picture taken.")
                break
        else:
            GPIO.output(21,1)
    self.after(2000, self.showIm2)

```

```

#####
#####compare#####
def mse(imageA, imageB):
    # the 'Mean Squared Error' between the two images is the
    # sum of the squared difference between the two images;
    # NOTE: the two images must have the same dimension
    err = np.sum((imageB.astype("float") - imageA.astype("float")) ** 2)
    err /= float(imageB.shape[0]*imageB.shape[1])
    # return the MSE, the lower the error, the more "similar"
    # the two images are
    return err

def compare_images(imageA, imageB, title):
    # compute the mean squared error and structural similarity
    # index for the images
    m = mse(imageA, imageB)
    s = ssim(imageA, imageB)

    if s > 0.8 :
        self.after(1000, self.run1) #back to clean system
    else:
        self.after(1000, self.Dry40) #out of system

    # setup the figure
    fig = plt.figure(title)
    plt.suptitle("MSE: %.2f, SSIM: %.2f" % (m, s))

    # show first image
    ax = fig.add_subplot(1, 2, 1)
    plt.imshow(imageA, cmap = plt.cm.gray)
    plt.axis("off")

    # show the second image
    ax = fig.add_subplot(1, 2, 2)
    plt.imshow(imageB, cmap = plt.cm.gray)
    plt.axis("off")

    # show the images
    plt.show()

def compare_images1(imageA, imageB, title):
    # compute the mean squared error and structural similarity
    # index for the images
    m = mse(imageA, imageB)
```

```

s = ssim(imageA, imageB)

if s > 0.8 :
    self.after(1000,self.Dry40) #back to clean system
else:
    self.after(1000,self.Dry40) #out of system

    # setup the figure
fig = plt.figure(title)
plt.suptitle("MSE: %.2f, SSIM: %.2f" % (m, s))

    # show first image
ax = fig.add_subplot(1, 2, 1)
plt.imshow(imageA, cmap = plt.cm.gray)
plt.axis("off")

    # show the second image
ax = fig.add_subplot(1, 2, 2)
plt.imshow(imageB, cmap = plt.cm.gray)
plt.axis("off")

    # show the images
plt.show()

def compare(self):
    before = cv2.imread("image1.jpg")
    after = cv2.imread("image2.jpg")

    # convert the images to grayscale
    before = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
    after = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

    # show the figure
    plt.show()

    # compare the images
    compare_images(before, after, 'Before vs. After')

def compare1(self):
    before = cv2.imread("image1.jpg")
    after = cv2.imread("image2.jpg")
    # convert the images to grayscale

```

```

before = cv2.cvtColor(before, cv2.COLOR_BGR2GRAY)
after = cv2.cvtColor(after, cv2.COLOR_BGR2GRAY)

# show the figure
plt.show()

# compare the images
compare_images1(before, after, 'Before vs. After')

#####
#####AUTO2#####
def run1(self):
    print("START")
    self.after(500, self.solenoid_in_2)

def solenoid_in_2(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break
        elif GPIO.input(27)==0: #top
            GPIO.output(23,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)
                sleep(.5)
                GPIO.output(7,GPIO.HIGH)
                sleep(.5)
                x += 1
            self.after_cancel(self.solenoid_in_2)
        else:
            print("NOT FULL")
            GPIO.output(23,1)
            self.after(100, self.clean_2)

def clean_2(self):
    GPIO.output(21,1)
    self.after(120000, self.solenoid_out)

def solenoid_out_2(self):
    GPIO.output(24,1)
    while True:

```

```

if GPIO.input(17) == 1:    #bottom
    GPIO.output(24,0)
    break
elif GPIO.input(22) == 0:  #mid
    GPIO.output(24,1)
else:
    GPIO.output(24,1)
self.after(100, self.solenoid_in1_2)

def solenoid_in1_2(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break
        elif GPIO.input(27)==0: #top
            GPIO.output(23,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)
                sleep(.5)
                GPIO.output(7,GPIO.HIGH)
                sleep(.5)
                x += 1
            self.after_cancel(self.solenoid_in1_2)
        else:
            print("NOT FULL")
            GPIO.output(23,1)
    self.after(100, self.clean_w_2)

def clean_w_2(self):
    GPIO.output(21,1)
    self.after(120000, self.solenoid_out1_2)

def solenoid_out1_2(self):
    GPIO.output(24,1)
    while True:
        if GPIO.input(17) == 1:    #bottom
            GPIO.output(24,0)
            break
        elif GPIO.input(22) == 0:  #mid
            GPIO.output(24,1)

```

```

        else:
            GPIO.output(24,1)
            self.after(100, self.camera2_2)

def camera2_2(self):
    while True:
        if GPIO.input(26)==0:
            GPIO.output(21,0)
            print("About to take a picture.")
            with picamera.PiCamera() as camera:
                self.camera.resolution = (500,500)
                camera.framerate = 24
                sleep(3)
                self.camera.capture("/home/pi/image2.jpg")
                print("Picture taken.")
                break
        else:
            GPIO.output(21,1)
            self.after(2000, self.showIm2_2)

#####
#Finish#####
def quit(self):
    GPIO.cleanup()
    self.master.destroy()

def finish(self):
    x=0
    while(x<5):
        GPIO.output(7,GPIO.LOW)
        sleep(.5)
        GPIO.output(7,GPIO.HIGH)
        sleep(.5)
        x += 1
    self.after_cancel(self.quit)
    self.master.switch_frame(StartPage)

def showIm1(self):
    img = cv2.imread('image1.jpg') #show image
    cv2.imshow('image',img)
    cv2.waitKey(5000)
    cv2.destroyAllWindows()
    self.after(100, self.solenoid_in)

def showIm2(self):

```

```

        img = cv2.imread('image2.jpg') #show image
        cv2.imshow('image',img)
        cv2.waitKey(5000)
        cv2.destroyAllWindows()
        self.after(100, self.compare)

def showIm2_2(self):
    img = cv2.imread('image2.jpg') #show image
    cv2.imshow('image',img)
    cv2.waitKey(5000)
    cv2.destroyAllWindows()
    self.after(100, self.compare1)

#####
#DRY#####
def Dry40(self):
    x=0
    while(x<300):
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        if humidity is not None and temperature is not None:
            print ('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
            sleep(3)
            t1=temperature
            temp1=t1
            if t1 < 40 :
                GPIO.output(20,1)
                GPIO.output(12,GPIO.LOW)
            elif t1 > 40:
                GPIO.output(20,0)
                GPIO.output(12,GPIO.LOW)
            else:
                GPIO.output(20,0)
                GPIO.output(12,GPIO.HIGH)
        else:
            print ('Failed to get reading. Try again!')
            self.master.update()
            x += 1
    else:
        GPIO.output(20,0)
        GPIO.output(12,GPIO.HIGH)
        GPIO.cleanup()
        self.after(1000, self.finish)

#####
class PageTwo(tk.Frame):

```

```

def __init__(self, master):
    tk.Frame.__init__(self, master)

    page_2_label = tk.Label(self, text="MANUAL MODE",fg="#000",font="times 25 bold",width=30,height=2)
    start_button = tk.Button(self, text="WASH",fg="#000",bg="cyan",font="times 24 bold",width=15,height=2,
                           command=lambda: master.switch_frame(PageWash))
    start1_button = tk.Button(self, text="DRY",fg="#000",bg="gray",font="times 24 bold",width=15,height=2,
                           command=lambda: master.switch_frame(PageDry))
    start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
                           command=lambda: master.switch_frame(StartPage))

    page_2_label.pack(side="top", fill="x", pady=10)
    start_button.pack(side="left", anchor=N , fill="x", pady=30)
    start1_button.pack(side="top", fill="x", pady=30)
    start2_button.pack(side="right")

class PageWash(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key",font="times 30",width=15, validatecommand=(vcmd, "%P"))
        page_2_label = tk.Label(self, text="TIMES (min)",fg="#000",font="times 25 bold",width=30,height=2)
        start_button = tk.Button(self, text="START",bg="RED",font="times 20 bold",width=11,
                               command=lambda: self.update("START"))
        reset_button = tk.Button(self, text="Reset",bg="white",font="times 20 bold",width=11,
                               command=lambda: self.update("reset"))
        start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
                               command=lambda: master.switch_frame(StartPage))

        page_2_label.pack(side="top", fill="x", pady=10)
        self.entry.pack(side="left",anchor=N,fill="x")
        start_button.pack(side="left", anchor=N)
        reset_button.pack(side="top",anchor=W)
        start2_button.pack(side="right",fill="x",pady=30)
        GPIO.cleanup()

    def validate(self, new_text):

```

```

if not new_text: # the field is being cleared
    self.entered_number = 0
    return True

try:
    self.entered_number = int(new_text)
    return True
except ValueError:
    return False

def update(self, method):
    if method == "START":
        t = abs(int(self.entered_number))
        ti = (t*60000)+60000
        if t == 0:
            self.after_cancel(self.run)
            self.after_cancel(self.clean)
            self.after_cancel(self.solenoid_out)
        else:
            self.after(t, self.run)
            #self.after(100, self.clean)
            self.after(ti, self.solenoid_out)
        else:
            t = 0
            self.after_cancel(self.run)
            self.entry.delete(0, END)
#####
#####MANAUL#####
def run(self):
    print("START")
    self.after(100, self.solenoid_in)

def solenoid_in(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break
        elif GPIO.input(27)==0: #top
            GPIO.output(23,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)

```

```

        sleep(.5)
        GPIO.output(7,GPIO.HIGH)
        sleep(.5)
        x += 1
        break
    self.after_cancel(self.solenoid_in)
else:
    print("NOT FULL")
    GPIO.output(23,1)
self.after(100, self.clean)

def clean(self):
    GPIO.output(21,1)

def solenoid_out(self):
    while True:
        if GPIO.input(17) == 1:    #bottom
            GPIO.output(24,0)
            break
        elif GPIO.input(22) == 0:  #mid
            GPIO.output(24,1)
        else:
            GPIO.output(24,1)
    self.after(100, self.solenoid_in1)

def solenoid_in1(self):
    while True:
        if GPIO.input(17)==1:
            GPIO.output(23,1)
        elif GPIO.input(22)==0:
            GPIO.output(23,0)
            print("FULL")
            break
        elif GPIO.input(27)==0:  #top
            GPIO.output(23,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)
                sleep(.5)
                GPIO.output(7,GPIO.HIGH)
                sleep(.5)
                x += 1
            self.after_cancel(self.solenoid_in1)
        else:

```

```

        print("NOT FULL")
        GPIO.output(23,1)
        self.after(100, self.clean_w)

    def clean_w(self):
        GPIO.output(21,1)
        self.after(120000, self.solenoid_out1)

    def solenoid_out1(self):
        GPIO.output(24,1)
        GPIO.output(24,1)
        while True:
            if GPIO.input(17) == 1:      #bottom
                GPIO.output(24,0)
                break
            elif GPIO.input(22) == 0:   #mid
                GPIO.output(24,1)
            else:
                GPIO.output(24,1)
        self.after(100, self.finish)
#####Finish#####
    def quit(self):
        GPIO.cleanup()
        self.master.destroy()

    def finish(self):
        if GPIO.input(26) == 0:
            GPIO.output(21,0)
            x=0
            while(x<5):
                GPIO.output(7,GPIO.LOW)
                sleep(.5)
                GPIO.output(7,GPIO.HIGH)
                sleep(.5)
                x += 1
            else:
                self.after_cancel(self.finish)
                self.master.switch_frame(StartPage)
            else:
                GPIO.output(21,1)
######
class PageDry(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

```

```

page_2_label = tk.Label(self, text="TEMPERATURE",fg="#000",font="times 25
bold",width=30,height=2)
start_button = tk.Button(self, text="40 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
command=lambda: master.switch_frame(PageDry40))
start1_button = tk.Button(self, text="45 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
command=lambda: master.switch_frame(PageDry45))
start2_button = tk.Button(self, text="50 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
command=lambda: master.switch_frame(PageDry50))
start3_button = tk.Button(self, text="55 C",fg="#000",bg="gray",font="times 24
bold",width=10,height=2,
command=lambda: master.switch_frame(PageDry55))
start4_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
command=lambda: master.switch_frame(StartPage))
page_2_label.pack(side="top", fill="x", pady=10)
start_button.pack(side="left", anchor=N , fill="x", pady=30)
start1_button.pack(side="left", anchor=N , fill="x", pady=30)
start2_button.pack(side="left", anchor=N , fill="x", pady=30)
start3_button.pack(side="top", fill="x", pady=30)
start4_button.pack(side="right")

class PageDry40(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key",font="times 30",width=15, validatecommand=(vcmd, "%P"))
        page_2_label = tk.Label(self, text="TIMES (min)",fg="#000",font="times 25 bold",width=30,height=2)
        start_button = tk.Button(self, text="START",bg="RED",font="times 20 bold",width=11,
                               command=lambda: self.update("START"))
        reset_button = tk.Button(self, text="Reset",bg="white",font="times 20 bold",width=11,
                               command=lambda: self.update("reset"))
        start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
                               command=lambda: master.switch_frame(StartPage))
        page_2_label.pack(side="top", fill="x", pady=10)
        self.entry.pack(side="left",anchor=N,fill="x")

```

```

start_button.pack(side="left", anchor=N)
reset_button.pack(side="top", anchor=W)
start2_button.pack(side="right", fill="x", pady=30)

def validate(self, new_text):
    if not new_text: # the field is being cleared
        self.entered_number = 0
    return True

try:
    self.entered_number = int(new_text)
    return True
except ValueError:
    return False

def update(self, method):
    if method == "START":
        ti = abs(int(self.entered_number))*20
        x=0
        while(x<ti):
            humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
            if humidity is not None and temperature is not None:
                print ('Temp={0:0.1f}*C  Humidity={1:0.1f}%'.format(temperature, humidity))
                sleep(3)
                t1=temperature
                temp1=t1
                if t1 < 40.0 :
                    GPIO.output(20,1)
                    GPIO.output(12,GPIO.LOW)
                elif t1 > 40.0:
                    GPIO.output(20,0)
                    GPIO.output(12,GPIO.LOW)
                else:
                    GPIO.output(20,0)
                    GPIO.output(12,GPIO.LOW)
            else:
                print ('Failed to get reading. Try again!')
                self.master.update()
                x += 1
        else:
            GPIO.output(20,0)
            GPIO.output(12,GPIO.HIGH)
            GPIO.cleanup()
            self.after_cancel(self.update)

```

```

        else: # reset
            ti = 0
            self.entry.delete(0, END)

    class PageDry45(tk.Frame):
        def __init__(self, master):
            tk.Frame.__init__(self, master)

            self.entered_number = 0

            self.total_label_text = IntVar()
            self.total_label = Label(master, textvariable=self.total_label_text)

            vcmd = master.register(self.validate) # we have to wrap the command
            self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, "%P"))
            page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
            start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                                    command=lambda: self.update("START"))
            reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,
                                    command=lambda: self.update("reset"))
            start2_button = tk.Button(self, text="HOME", fg="#FFF", bg="black", font="times 24 bold", width=10,
                                    command=lambda: master.switch_frame(StartPage))
            page_2_label.pack(side="top", fill="x", pady=10)
            self.entry.pack(side="left", anchor=N, fill="x")
            start_button.pack(side="left", anchor=N)
            reset_button.pack(side="top", anchor=W)
            start2_button.pack(side="right", fill="x", pady=30)

        def validate(self, new_text):
            if not new_text: # the field is being cleared
                self.entered_number = 0
            return True

        try:
            self.entered_number = int(new_text)
            return True
        except ValueError:
            return False

    def update(self, method):
        if method == "START":
            ti = abs(int(self.entered_number))*20
            x=0
            while(x<ti):

```

```

humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
if humidity is not None and temperature is not None:
    print ('Temp={0:0.1f}*C  Humidity={1:0.1f}%.format(temperature, humidity))
    sleep(3)
    t1=temperature
    temp1=t1
    if t1 < 45.0 :
        GPIO.output(20,1)
        GPIO.output(12,GPIO.LOW)
    elif t1 > 45.0:
        GPIO.output(20,0)
        GPIO.output(12,GPIO.LOW)
    else:
        GPIO.output(20,0)
        GPIO.output(12,GPIO.LOW)
    else:
        print ('Failed to get reading. Try again!')
        self.master.update()
        x += 1
    else:
        GPIO.output(20,0)
        GPIO.output(12,GPIO.HIGH)
        GPIO.cleanup()
        self.after_cancel(self.update)
else: # reset
    ti = 0
self.entry.delete(0, END)

```

```

class PageDry50(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, "%P"))
        page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
        start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                               command=lambda: self.update("START"))
        reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,

```

```

        command=lambda: self.update("reset"))

start2_button = tk.Button(self, text="HOME",fg="#FFF",bg="black",font="times 24 bold",width=10,
                        command=lambda: master.switch_frame(StartPage))

page_2_label.pack(side="top", fill="x", pady=10)
self.entry.pack(side="left",anchor=N,fill="x")
start_button.pack(side="left", anchor=N)
reset_button.pack(side="top",anchor=W)
start2_button.pack(side="right",fill="x",pady=30)

def validate(self, new_text):
    if not new_text: # the field is being cleared
        self.entered_number = 0
    return True

try:
    self.entered_number = int(new_text)
    return True
except ValueError:
    return False

def update(self, method):
    if method == "START":
        ti = abs(int(self.entered_number))*20
        x=0
        while(x<ti):
            humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
            if humidity is not None and temperature is not None:
                print ('Temp={0:0.1f}*C  Humidity={1:0.1f}%.format(temperature, humidity))
                sleep(3)
                t1=temperature
                temp1=t1
                if t1 < 50.0 :
                    GPIO.output(20,1)
                    GPIO.output(12,GPIO.LOW)
                elif t1 > 50.0:
                    GPIO.output(20,0)
                    GPIO.output(12,GPIO.LOW)
                else:
                    GPIO.output(20,0)
                    GPIO.output(12,GPIO.LOW)
            else:
                print ('Failed to get reading. Try again!')
                self.master.update()
                x += 1

```

```

        else:
            GPIO.output(20,0)
            GPIO.output(12,GPIO.HIGH)
            GPIO.cleanup()
            self.after_cancel(self.update)

    else: # reset
        ti = 0
        self.entry.delete(0, END)

class PageDry55(tk.Frame):
    def __init__(self, master):
        tk.Frame.__init__(self, master)

        self.entered_number = 0

        self.total_label_text = IntVar()
        self.total_label = Label(master, textvariable=self.total_label_text)

        vcmd = master.register(self.validate) # we have to wrap the command
        self.entry = tk.Entry(self, validate="key", font="times 30", width=15, validatecommand=(vcmd, "%P"))
        page_2_label = tk.Label(self, text="TIMES (min)", fg="#000", font="times 25 bold", width=30, height=2)
        start_button = tk.Button(self, text="START", bg="RED", font="times 20 bold", width=11,
                               command=lambda: self.update("START"))
        reset_button = tk.Button(self, text="Reset", bg="white", font="times 20 bold", width=11,
                               command=lambda: self.update("reset"))
        start2_button = tk.Button(self, text="HOME", fg="#FFF", bg="black", font="times 24 bold", width=10,
                               command=lambda: master.switch_frame(StartPage))
        page_2_label.pack(side="top", fill="x", pady=10)
        self.entry.pack(side="left", anchor=N, fill="x")
        start_button.pack(side="left", anchor=N)
        reset_button.pack(side="top", anchor=W)
        start2_button.pack(side="right", fill="x", pady=30)

    def validate(self, new_text):
        if not new_text: # the field is being cleared
            self.entered_number = 0
        return True

    try:
        self.entered_number = int(new_text)
        return True
    except ValueError:
        return False

```

```

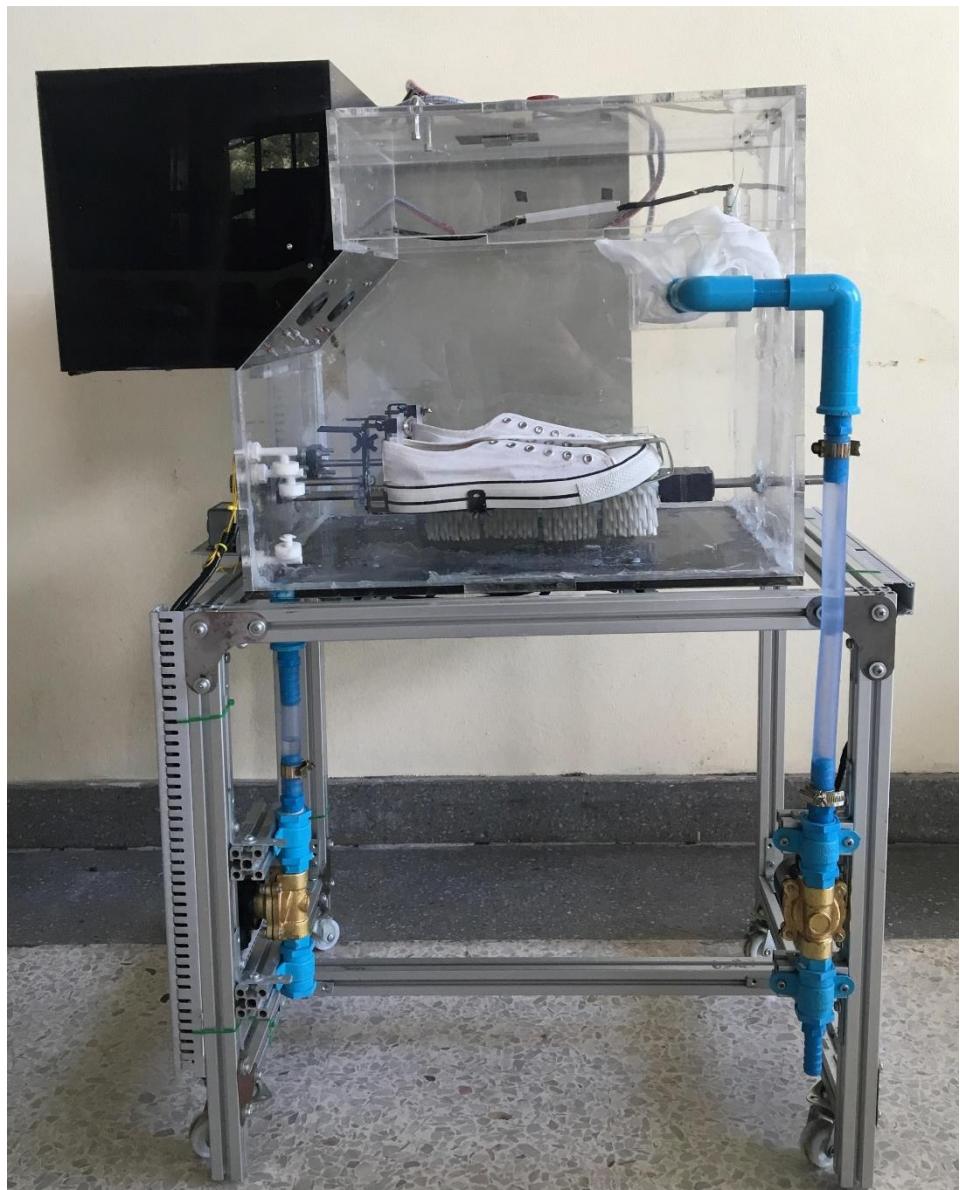
def update(self, method):
    if method == "START":
        ti = abs(int(self.entered_number))*20
        x=0
        while(x<ti):
            humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
            if humidity is not None and temperature is not None:
                print ('Temp={0:0.1f}*C  Humidity={1:0.1f}%.format(temperature, humidity))
                sleep(3)
                t1=temperature
                temp1=t1
                if t1 < 55.0 :
                    GPIO.output(20,1)
                    GPIO.output(12,GPIO.LOW)
                elif t1 > 55.0:
                    GPIO.output(20,0)
                    GPIO.output(12,GPIO.LOW)
                else:
                    GPIO.output(20,0)
                    GPIO.output(12,GPIO.LOW)
            else:
                print ('Failed to get reading. Try again!')
                self.master.update()
                x += 1
        else:
            GPIO.output(20,0)
            GPIO.output(12,GPIO.HIGH)
            GPIO.cleanup()
            self.after_cancel(self.update)
    else: # reset
        ti = 0
        self.entry.delete(0, END)

if __name__ == "__main__":
    app = SampleApp()
    app.mainloop()
    GPIO.cleanup()

```

ภาคผนวก จ.

รูปเครื่องซักรองเท้ากีฬาแบบอัตโนมัติโดยใช้การประมวลผลภาพ



รูปที่ จ-1 รูปด้านหน้าของเครื่องซักรองเท้ากีฬาแบบอัตโนมัติ



รูปที่ จ-2 รูปด้านข้างของเครื่องซักรองเท้ากีฬาแบบอัตโนมัติผ่านมอเตอร์สำหรับหมุนรองเท้า



รูปที่ จ-3 รูปด้านข้างของเครื่องซักรองเท้ากีฬาแบบอัตโนมัติผึ่งระบบยล莫อก



รูปที่ จ-4 รูปด้านหลังของเครื่องซักรองเท้ากีฬาแบบอัตโนมัติผู้ดูแลควบคุม



รูปที่ จ-5 รูปองค์ประกอบทั้งหมดของเครื่องซักรองเท้ากีฬาแบบอัตโนมัติ