เครื่องคัดแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติ

THE SEMI-AUTOMATIC LIMES COLOR AND SIZE SORTER

ณรงค์ เรืองฤทธิ์

พัชรพล ยาฟอง

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

ปีการศึกษา 2563

ปริญญานิพนธ์ปีการศึกษา 2563
สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

เรื่อง เครื่องคัดแยกสีและขนาดของมะนาวกึ่งอัตโนมัติ

THE SEMI-AUTOMATIC LIMES COLOR AND SIZE SORTER

ผู้จัดทำ

1. นายณรงค์ เรืองฤทธิ์ รหัสนักศึกษา 60511046
2. นายพัชรพล ยาฟอง รหัสนักศึกษา 60511061

.....อาจารย์ที่ปรึกษา
(อาจารย์พิมล ผลพฤกษา)

.....อาจารย์ที่ปรึกษาร่วม
(ผศ.ดร.เกษมสุข เสพศิริสุข)

ชื่อปริญญานิพนธ์	เครื่องคัดแยกสีและขนาดของผลมะนาวกึ่งอัตโนมัติ	
นักศึกษา	นายณรงค์ เรืองฤทธิ์	รหัสนักศึกษา 60511046
	นายพัชรพล ยาฟอง	รหัสนักศึกษา 60511061
อาจารย์ที่ปรึกษา	อาจารย์พิมล ผลพฤกษา	
อาจารย์ที่ปรึกษาร่วม	ผศ.ดร.เกษมสุข เสพศิริสุข	
หลักสูตร	วิศวกรรมศาสตรบัณฑิต	
สาขาวิชา	วิศวกรรมอิเล็กทรอนิกส์	
ปีการศึกษา	2563	

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ได้กล่าวถึงการทำงานของเครื่องคัดแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติ โดยแบ่งเป็น 3 ส่วนสำคัญคือ ส่วนที่หนึ่งการประมวลผลภาพ ส่วนที่สองการคัดแยกสี และส่วนที่สามการคัดแยกขนาด ส่วนที่หนึ่งการประมวลผลภาพ ใช้โอเพ่นซีวีซึ่งเป็นไลบรารีฟังก์ชันในการตรวจจับสี โดยใช้ระบบสีที่คล้ายกับการมองเห็นของมนุษย์คือระบบสีเฮซเอสวี และนำไปเปรียบเทียบความแตกต่างของสี โดยการคัดแยกสีผลมะนาวแบ่งออกเป็น 2 ประเภท คือประเภทที่ 1 คือมะนาวผลสีเขียว และประเภทที่ 2 คือมะนาวผลสีเขียวอมเหลือง และมะนาวผลสีเหลือง ส่วนที่สองการคัดแยกสีผลมะนาว จะอาศัยผลลัพธ์ของการประมวลผลภาพของส่วนที่หนึ่งควบคุมองศาการไหลของผลมะนาว เพื่อคัดแยกมะนาวประเภทที่ 1 และประเภทที่ 2 ออกจากกัน และส่วนที่สามการคัดแยกขนาด เพื่อคัดแยกขนาดออกเป็นเป็น 3 ขนาด คือมะนาวเบอร์ 1 2 และ 3 จากการทดลองพบว่าการประมวลผลภาพของผลมะนาวประเภทที่ 1 และ ประเภทที่ 2 สามารถคัดแยกประเภทของผลมะนาวออกจากกันได้ การทดลองคัดแยกขนาดผลมะนาวพบว่าการคัดแยกขนาดมะนาวเบอร์ 1 มีค่าความผิดพลาดเฉลี่ยที่ 7.33 เปอร์เซ็นต์ มะนาวเบอร์ 2 มีค่าความผิดพลาดเฉลี่ยที่ 7.3 เปอร์เซ็นต์ และมะนาวเบอร์ 3 มีค่าความผิดพลาดเฉลี่ยที่ 2.66 เปอร์เซ็นต์ ระยะเวลาเฉลี่ยในการคัดแยกขนาดและสีของผลมะนาว 40 ลูกคือ 53.2 วินาที มีค่าความผิดพลาดในการคัดแยกเฉลี่ย 10 เปอร์เซ็นต์

Project Title	The semi-automatic limes color and size sorter	
Student	Mr. Narong Ruengrit	ID 60511046
	Mr. Patcharapon Yafong	ID 60511061
Advisor	Mr. Phimon Phonphruksa	
Co-Advisor	Asst.Prof.Dr. Kasemsuk Sepsirisuk	
Education Level	Bachelor of Engineering	
Program in	Electronics Engineering	
Academic Year	2020	

ABSTRACT

This thesis examines the semi-automatic limes color and size sorting machine and discusses the operation of a semi-automatic machine. The study is divided into three significant parts, which include the first part of image processing, the second part of color sorting, and lastly, size sorting operation. Image Processing in the first part uses OpenCV, a color detection function library, to compare color differences by using a color scheme comparable to human vision, the HSV color system, and color sorting, where lemons are divided into two types: Green lime is the first type, while yellow-green lime and yellow lemon are the second. The second part, the color sorting of lemons, relies upon and utilizes the results of the first part's image processing to regulate the degree of the flow of the lemon fruit to distinguish between type 1 and type 2 lemons. In addition, size sorting in the third part, to sort out the sizes into three categories: namely, No. 1, No. 2, and No. 3. According to the results of the experiment, image processing of type 1 and type 2 lemons were able to differentiate the two types of lemons, while the lime size sorting experiment found that lime No. 1 had an error of 7.33 percent, Lime No. 2 had an error of 7.3 percent, and Lime No. 3 had an error of 2.66 percent. The duration for sorting 40 lemons by size and color was 53.2 seconds, with a sorting error less than 10 percent.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้เป็นอย่างดี ด้วยความช่วยเหลือ และการสนับสนุนจากบุคคลหลายๆท่าน ซึ่งผู้เขียนขอขอบคุณทุกๆ ท่านดังต่อไปนี้

ขอขอบพระคุณคุณพ่อและคุณแม่ ผู้ซึ่งคอยให้การอบรมสั่งสอน เลี้ยงดู สนับสนุนการศึกษาตลอดจนให้กำลังใจเสมอมา

ขอขอบพระคุณ อาจารย์พิมล ผลพฤกษา อาจารย์ที่ปรึกษา ผู้ซึ่งให้คำแนะนำต่างๆ และติดตามเกี่ยวกับงานโครงงานตลอดมา ผู้เขียนรู้สึกซาบซึ้งในความเมตตาของท่านจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณ ผศ.ดร.เกษมสุข เสพศิริสุข อาจารย์ที่ปรึกษาร่วมผู้ให้คำปรึกษา และคำแนะนำต่าง ๆ เกี่ยวกับการทำโครงงาน ผู้เขียนรู้สึกซาบซึ้งในความเมตตาของท่านจึงขอกราบขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณคณะอาจารย์ที่เคารพทุกท่าน ที่ให้ความเอาใจใส่แนะนำ คอยช่วยเหลือเสมอมา และต้องขอขอบคุณพี่ๆ เพื่อนๆ น้องๆ ที่คอยช่วยเหลือในการทำโครงงานจนสำเร็จลุล่วงไปได้ด้วยดี

คุณค่า และประโยชน์อันพึงมีจากปริญญานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

ณรงค์ เรืองฤทธิ์

พัชรพล ยาฟอง

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป(ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

ในบทนี้จะกล่าวถึงความเป็นมาและความสำคัญของปัญหา จุดมุ่งหมายและวัตถุประสงค์ ขอบเขตของการทำโครงงาน ประโยชน์หรือผลที่คาดว่าจะได้รับ ขั้นตอนและวิธีการดำเนินงาน ซึ่งมีรายละเอียดดังนี้

1.1 ความเป็นมาและความสำคัญของโครงงาน

มะนาวเป็นพืชเศรษฐกิจที่มีความต้องการบริโภคในแต่ละวันเป็นจำนวนมาก จึงมีเกษตรกรในหลายพื้นที่ยึดอาชีพปลูกมะนาวเป็นอาชีพหลัก ซึ่งการคัดแยกผลมะนาวด้วยแรงงานคนจะต้องอาศัยพนักงานจำนวนมากและใช้เวลาในการคัดแยกเป็นเวลานาน ส่งผลให้ผู้ประกอบการเสียค่าใช้จ่ายในการจ้างแรงงานและได้ผลผลิตไม่เพียงพอต่อความต้องการของผู้บริโภค

ดังนั้นจึงมีแนวคิดพัฒนาเครื่องคัดแยกสีและขนาดของผลมะนาวเพื่อที่จะช่วยลดระยะเวลาในการคัดแยกผลมะนาว ทั้งยังเพิ่มผลผลิตให้เพียงพอต่อความต้องการของตลาด และช่วยลดค่าใช้จ่ายในการจ้างพนักงานคัดแยกสีและขนาดของผลมะนาว

1.2 วัตถุประสงค์ของการทำโครงงาน

1. เพื่อศึกษาและเรียนรู้การคัดแยกสีมะนาวด้วยการประมวลผลภาพ (Image Processing) โดยการใช้ภาษาไพธอนและใช้โอเพ่นซีวีเป็นไลบรารี่ฟังก์ชันในการเขียนโปรแกรมประมวลผลภาพ
2. เพื่อลดทรัพยากรแรงงานคนงานที่ทำหน้าที่คัดแยกสีและขนาดของผลมะนาว
3. เพื่อลดระยะเวลาในการคัดแยกสีและขนาดของมะนาว

1.3 สมมติฐานของการศึกษา

การคัดแยกสีของผลมะนาวใช้วิธีการประมวลผลภาพเพื่อคัดแยกสีของผลมะนาว จากนั้นจึงนำผลลัพธ์ที่ได้จากการประมวลผลภาพไปใช้กำหนดการทำงานของเซอร์โวมอเตอร์ เพื่อใช้คัดแยกผลมะนาวสีเขียวและผลมะนาวสีเหลืองออกจากกัน มะนาวผลสีเขียวจะผ่านไปสู่ขั้นตอนการคัดแยกขนาดโดยคัดมะนาวผลสีเขียวออกเป็น 3 ขนาด โดยขนาดที่ 1 มีเส้นผ่านศูนย์กลาง 4.2 เซนติเมตรขึ้นไป ขนาดที่ 2 มีเส้นผ่านศูนย์กลาง 4.0 ถึง 4.2 เซนติเมตรและ ขนาดที่ 3 มีเส้นผ่านศูนย์กลาง 4.0 ถึง 2.7 เซนติเมตร

1.4 ขอบเขตของโครงงาน

1. สามารถคัดแยกสีของมะนาวผลสีเหลืองและมะนาวผลสีเขียวออกจากกัน
2. สามารถคัดแยกขนาดผลมะนาวสีเขียวได้ตามขนาดมาตรฐานของท้องตลาด
   - มะนาวเบอร์ 1 มีขนาดเส้นผ่านศูนย์กลางมากกว่า 4.2 เซนติเมตร
   - มะนาวเบอร์ 2 มีขนาดเส้นผ่านศูนย์กลาง 4.0 ถึง 4.2 เซนติเมตร
   - มะนาวเบอร์ 3 มีขนาดเส้นผ่านศูนย์กลาง 4.0 ถึง 2.7 เซนติเมตร
3. สามารถคัดด้วยความเร็วไม่ต่ำกว่า 40 ผลต่อนาที

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เครื่องคัดแยกขนาดและสีของผลมะนาวที่ควบคุมด้วยระบบการประมวลผลภาพ
2. สามารถคัดแยกขนาดผลมะนาวได้ตามขนาดมาตรฐานของท้องตลาด
3. ลดต้นทุนในการจ้างพนักงานคัดแยกผลมะนาว
4. ผู้รับซื้อมะนาวมีรายได้เพิ่มขึ้น

1.6 ขั้นตอนการดำเนินงาน

ขั้นตอนการดำเนินงานที่ผู้จัดทำได้วางแผนไว้ 2 ช่วงใหญ่ๆ คือแผนการดำเนินงานในภาคเรียนที่ 1 และภาคเรียนที่ 2 ซึ่งได้แจกแจงรายละเอียดไว้ในตารางที่ 1.1 และตารางที่ 1.2 ดังนี้

ตารางที่ 1.1 ขั้นตอนและวิธีการดำเนินงานภาคเรียนที่ 1

ขั้นตอนการดำเนินงาน	ระยะเวลาการดำเนินงาน																			
	สิงหาคม				กันยายน				ตุลาคม				พฤศจิกายน				ธันวาคม			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1. ศึกษาทฤษฎีและงานวิจัย	↔	↔																		
2. ศึกษาการเขียนโปรแกรมโดยใช้ภาษา Python			↔	↔	↔															
3. ออกแบบโครงสร้างของตัวเครื่อง				↔	↔	↔	↔													
4. จัดทำรายการอุปกรณ์และสั่งซื้ออุปกรณ์						↔	↔	↔	↔											
5. ทำระบบลำเลียงผลมะนาวและคัดแยกสี										↔	↔	↔	↔	↔						
6. ทดลองและวิเคราะห์ผลการทดลอง														↔	↔					
7. รายงานและนำเสนอ																↔	↔	↔	↔	↔

ตารางที่ 1.2 ขั้นตอนและวิธีการดำเนินงานภาคเรียนที่ 2

ขั้นตอนการดำเนินงาน	ระยะเวลาการดำเนินงาน																			
	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. ศึกษาการใช้งานอุปกรณ์	↔																			
2. เขียนโปแกรม			→	→	→	→														
3. ออกแบบโครงสร้างของตัวเครื่อง				→	→	→	→	→												
4. จัดทำโครงสร้างและปรับแก้โครงสร้างของเครื่อง							→	→	→	→	→	→	→							
5. ทำระบบลำเลียงผลมะนาวและคัดแยกสี										→	→	→	→	→						
6. ทดลองและวิเคราะห์ผลการทดลอง															→	→				
7. รายงานและนำเสนอ																	↔			

1.7 โครงสร้างปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้นำเสนอเรื่อง เครื่องคัดแยกสีและของมะนาวกึ่งอัตโนมัติ โดยใช้ราสเบอรี่พายมาควบคุมการทำงานของเครื่อง ซึ่งเป็นการอธิบายขั้นตอนการทำงาน ผลการทดลองสุดท้ายจะเป็นการสรุปผลการทดลองและข้อเสนอแนะ

บทที่ 1 ในบทนี้จะกล่าวถึง ความเป็นมาและความสำคัญของปัญหา จุดมุ่งหมายและวัตถุประสงค์ ขอบเขตของการศึกษา ขั้นตอนการดำเนินงาน และโครงสร้างปริญญานิพนธ์

บทที่ 2 ในบทนี้จะกล่าวถึง ทฤษฎีและหลักการที่เกี่ยวข้องกับโครงงาน ได้แก่ ราสเบอรี่พาย โมดูลกล้อง ภาษาไพทอน รีเลย์ ไลบรารีโอเพ่นซีวี การประมวลผลภาพ มอเตอร์ทดเกียร์ เซ็นเซอร์ตรวจจับวัตถุ บอร์ดขับเซอร์โวมอเตอร์

บทที่ 3 ในบทนี้กล่าวถึง บล็อกไดอะแกรมของระบบ โฟลว์ชาร์ตการทำงานคัดแยกสีของโปรแกรม การออกแบบวงจรที่ใช้งาน การกำหนดค่าเริ่มต้นระบบสีเอชเอสวี (HSV) และการออกแบบเครื่องคัดแยกขนาดและสีของมะนาวกึ่งอัตโนมัติ

บทที่ 4 ในบทนี้จะกล่าวถึง การทดลองและผลการทดลองการคัดแยกสีและขนาดของผลมะนาวโดยใช้การประมวลผลภาพ

บทที่ 5 ในบทนี้จะกล่าวถึง สรุปผลการทดลอง ข้อเสนอแนะ ปัญหา และแนวการแก้ไขของเครื่องคัดแยกสีและขนาดของมะนาวกึ่งอัตโนมัติ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ในบทนี้จะกล่าวถึงทฤษฎีและหลักการของอุปกรณ์ต่างๆ ที่เกี่ยวข้องกับการจัดทำเครื่องคัดแยกสีและขนาดของมะนาวกึ่งอัตโนมัติควบคุมด้วยราสเบอรี่พาย และอุปกรณ์ที่เกี่ยวข้อง

2.1 ผลและดอกมะนาว

มะนาว[1] เป็นพืชพื้นเมืองในภูมิภาคเอเชียตะวันออกเฉียงใต้ ซึ่งนิยมใช้ประโยชน์จากมะนาวโดยน้ำมะนาวให้รสชาติเปรี้ยวสามารถใช้ในการปรุงอาหาร และยังนำมาใช้เป็นเครื่องดื่มได้ ถือเป็นพืชเศรษฐกิจที่มีบทบาทสูง และเป็นที่ต้องการของตลาดตลอดทั้งปี โดยเฉพาะช่วงฤดูแล้งประมาณเดือนมีนาคม-เมษายน มักมีผลผลิตเข้าสู่ตลาดในปริมาณน้อย ทำให้มะนาวช่วงหน้าแล้งมีราคาสูงกว่าปกติ

มะนาว จัดอยู่ในสกุลส้ม โดยลำต้นของมะนาวเป็นไม้พุ่มเตี้ย สูงเต็มที่ราว 5 เมตร ก้านมีหนามเล็กน้อย มักมีใบดก ใบยาวเรียวเล็กน้อย คล้ายใบส้ม ส่วนดอกสีขาวอมเหลืองปกติจะออกดอกตลอดทั้งปี ผลมะนาวมีสีเขียวเมื่อสุกจัดจะเป็นสีเหลือง เปลือกบาง ภายในมีเนื้อแบ่งกลีบๆ ชุ่มน้ำมาก โดยทั่วไปมีขนาดเส้นผ่านศูนย์กลางประมาณ 4 – 4.5 เซนติเมตร แสดงดังรูปที่ 2.1

รูปที่ 2.1 ผลมะนาว

2.1.1 มาตรฐานมะนาว

ได้มีการจัดทำมาตรฐานมะนาวตามมาตรฐานสินค้าเกษตรและอาหารแห่งชาติได้แก่ มาตรฐานมะนาว[2] ตาม มกษ. 27-2560

การแบ่งชั้นคุณภาพ ผลมะนาวตามมาตรฐานสินค้าเกษตรนี้ แบ่งเป็น 3 ชั้นคุณภาพ ดังนี้

2.1.1.1 ชั้นพิเศษ (Extra class) มะนาวในชั้นนี้ต้องมีคุณภาพดีที่สุด มีลักษณะตรงตามพันธุ์ เป็นสีเขียวทั้งผล ไม่มีความผิดปกติ ด้านรูปทรงและสีไม่มีตำหนิที่ผิว ยกเว้นเป็นความผิดปกติหรือตำหนิที่มองเห็นได้ไม่ชัดเจน และไม่มีผลกระทบต่อรูปลักษณ์ทั่วไป และคุณภาพของผลิตผล รวมถึงคุณภาพระหว่าง การเก็บรักษา และการจัดบรรจุในหีบห่อ

2.1.1.2 ชั้นหนึ่ง (Class I) มะนาวในชั้นนี้ต้องมีคุณภาพดี มีลักษณะตรงตามพันธุ์ อาจมีความผิดปกติด้านรูปทรงและสี หรือตำหนิที่ผิวได้เล็กน้อย อย่างไรก็ตาม ความผิดปกติหรือตำหนิดังกล่าวต้องไม่มีผลกระทบ ต่อรูปลักษณ์ทั่วไป และคุณภาพของผลิตผล รวมถึงคุณภาพระหว่างการเก็บรักษา และการจัด บรรจุในหีบห่อ ความผิดปกติหรือตำหนิดังกล่าวที่ยอมให้มีได้ มีดังนี้

1. ความผิดปกติเล็กน้อยด้านรูปทรง
2. ความผิดปกติเล็กน้อยด้านสี เช่น สีผิวผลซีดเนื่องจากเป็นส่วนที่ไม่ได้รับแสงแดด
3. ตำหนิเล็กน้อยที่ผิว ในกรณีที่เป็นแผลต้องเป็นรอยแผลที่สมานแล้ว เช่น รอยขีดข่วน รอยแผลตื้น และร่องรอยการทำลายของแมลง โดยขนาดของตำหนิที่ผิวโดยรวมต้อง ไม่เกิน 5% ของพื้นที่ผิวของผล

2.1.1.3 ชั้นสอง (Class II) มะนาวในชั้นนี้รวมมะนาวที่มีคุณภาพไม่เข้าชั้นที่สูงกว่า แต่มีคุณภาพตามข้อกำหนดขั้นต่ำที่กำหนด อย่างไรก็ตาม มะนาวในชั้นนี้มีความผิดปกติหรือตำหนิได้ หากยังคง ลักษณะที่สำคัญในเรื่องคุณภาพ คุณภาพระหว่างการเก็บรักษา และการจัดบรรจุในหีบห่อ ทั้งนี้ความผิดปกติหรือตำหนิดังกล่าวที่ยอมให้มีได้ มีดังนี้

1. ความผิดปกติด้านรูปทรง
2. ความผิดปกติด้านสีเช่น สีผิวผลซีดเนื่องจากเป็นส่วนที่ไม่ได้รับแสงแดด
3. ตำหนิที่ผิว ในกรณีที่เป็นแผลต้องเป็นรอยแผลที่สมานแล้ว เช่น รอยขีดข่วน รอยแผลตื้น และร่องรอยการทำลายของแมลง โดยขนาดของตำหนิที่ผิวโดยรวมต้องไม่เกิน 10% ของพื้นที่ผิวของผล

2.1.2 เกณฑ์ขนาดของมะนาวพิจารณาเทียบกับเส้นผ่านศูนย์กลาง

เกณฑ์ขนาดของผลมะนาวพิจารณาเทียบกับเส้นผ่านศูนย์กลาง แสดงดังตารางที่ 2.1 ดังนี้

ตารางที่ 2.1 เกณฑ์ขนาดของมะนาวพิจารณาเทียบกับเส้นผ่านศูนย์กลาง

รหัสขนาด	เส้นผ่านศูนย์กลาง (เซนติเมตร)
1	> 4.2
2	4.0-4.2
3	2.7-4.0
4	< 2.7

หมายเหตุ : ใช้ข้อมูลในตารางที่ 2.1 ในการอ้างอิงเพื่อคัดแยกขนาดของมะนาว

2.1.3 เกณฑ์การคัดสีของมะนาว

เกณฑ์การคัดแยกสีมะนาวซึ่งอ้างอิงจาก ความต้องการของตลาด แสดงดังตารางที่ 2.2 ดังนี้

ตารางที่ 2.2 เกณฑ์การคัดแยกสีมะนาว

ผลมะนาว	ระยะผลมะนาว	ลักษณะผลมะนาว
	ผลอ่อน	-มีสีเขียวเข้ม -เปลือกมีลักษณะขรุขระ -น้ำในผลค่อนข้างน้อย
	ผลเริ่มสุก	-มีสีเขียวปนเหลือง -เปลือกมีลักษณะเรียบ
	ผลสุก	-มีสีเหลือง -เปลือกมีลักษณะเรียบ เกลี้ยง -น้ำในผลค่อยข้างมาก

หมายเหตุ : ใช้ข้อมูลในตารางที่ 2.2 ในการอ้างอิงเพื่อคัดแยกสีของมะนาว

2.2 บอร์ดราสเบอร์รี่พาย

ราสเบอร์รี่พาย[3] (Raspberry Pi) คือ บอร์ดที่เปรียบเสมือนคอมพิวเตอร์ขนาดเล็ก มีคุณสมบัติในการเชื่อมต่อกับคีย์บอร์ด เมาส์ และจอมอนิเตอร์ ราสเบอร์รี่พาย สามารถนำมาประยุกต์ใช้งานได้หลายด้าน เช่น ใช้ในการเขียนโปรแกรมควบคุมต่าง ๆ หรือนำมาใช้งานทางด้านอิเล็กทรอนิกส์และยังสามารถใช้เป็นคอมพิวเตอร์ที่มีขนาดเล็ก

บอร์ดราสเบอร์รี่พายสามารถใช้กับระบบปฏิบัติการลินุกซ์ (Linux) โดยรองรับได้หลายระบบ เช่น ราสเบียน (Raspbian) เป็นต้น โดยทำการติดตั้งนเอสดีการ์ด (SD Card) บอร์ดราสเบอร์รี่พายจะมีซีพียู (Central Processing Unit : CPU), จีพียู (Graphics Processing Unit : GPU) และแรม (Random Access Memory : RAM) ภายในชิปเดียวกัน มีจุดเชื่อมต่อกับจีพีไอโอ (General Purpose Input/Output : GPIO) สำหรับนำไปใช้ร่วมกับอุปกรณ์อิเล็กทรอนิกส์อื่นๆ ได้ บอร์ดราสเบอร์รี่พายมีลักษณะ แสดงดังรูปที่ 2.2

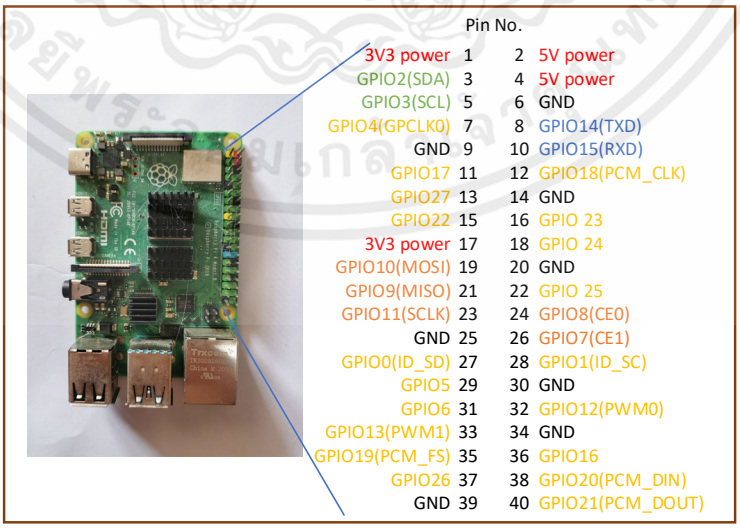
รูปที่ 2.2 บอร์ดราสเบอร์รี่พาย

2.2.1 คุณสมบัติทางเทคนิคของบอร์ด

บอร์ดราสเบอร์รี่พาย ปัจจุบันนี้มี 2 โมเดล คือ โมเดล A และโมเดล B ซึ่งทั้ง 2 โมเดลมีคุณสมบัติทางเทคนิคที่คล้ายคลึงกัน มีเพียงบางส่วนที่แตกต่างกัน

2.2.2 จุดเชื่อมต่อบนบอร์ดราสเบอรี่พาย

ราสเบอร์รี่พายสามารถเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์ได้ผ่านจีพีไอโอ ซึ่งประกอบด้วย ยูอาร์ท (Universal Asynchronous Receiver Transmitter : UART), เอสพีไอ (Serial Peripheral Interface : SPI), พีดับบลิวเอ็ม (Pulse Width Modulation : PWM), ไอสแควซี (Inter-Integrated Circuit : I2C) และอื่นๆ เพื่อใช้ในการควบคุม และสื่อสารกับอุปกรณ์อิเล็กทรอนิกส์ ซึ่งพอร์ตจีพีไอโอ เป็นพอร์ตอินพุตเอาต์พุตสามารถนำไปใช้งานได้ 40 พอร์ต จึงสามารถเชื่อมต่อกับอุปกรณ์อิเล็กทรอนิกส์ได้หลากหลายชนิด และมีพอร์ตสำหรับจ่ายไฟเลี้ยงให้กับอุปกรณ์อิเล็กทรอนิกส์ที่แรงดัน 3V, 5V และกราวด์ ซึ่งได้แสดงดังรูปที่ 2.3



รูปที่ 2.3 จุดเชื่อมต่อบนบอร์ดราสเบอร์รี่พาย

2.3 ภาษาไพธอน

ภาษาไพธอน[4] (Python) เป็นภาษาเขียนโปรแกรมระดับสูงที่ใช้กันอย่างกว้างขวางในการเขียนโปรแกรมสำหรับวัตถุประสงค์ทั่วไป ภาษาไพธอนนั้นสร้างโดย Guido van Rossum และถูกเผยแพร่ครั้งแรกในปี 1991 ภาษาไพธอนนั้นเป็นภาษาแบบอินเทอร์พรีเตอร์ (interpreter) ที่ถูกออกแบบโดยมีปรัญชาที่จะทำให้โค้ดอ่านได้ง่ายขึ้น และโครงสร้างของภาษานั้นจะทำให้โปรแกรมเมอร์สามารถเข้าใจแนวคิดการเขียนโค้ดโดยใช้บรรทัดที่น้อยลงกว่าภาษาอย่างซีพลัสพลัส และภาษาจาวา ซึ่งภาษานั้นถูกกำหนดให้มีโครงสร้างที่ตั้งใจให้การเขียนโค้ดเข้าใจง่ายทั้งในโปรแกรมเล็กไปจนถึงโปรแกรมขนาดใหญ่

ภาษาไพธอนมีคุณสมบัติเป็นภาษาเขียนโปรแกรมแบบไดนามิกส์และมีระบบการจัดการหน่วยความจำอัตโนมัติและสนับสนุนการเขียนโปรแกรมหลายรูปแบบ ที่ประกอบไปด้วย การเขียนโปรแกรมเชิงวัตถุ การเขียนโปรแกรมแบบฟังก์ชัน และการเขียนโปรแกรมแบบขั้นตอน มันมีไลบรารี่ที่ครอบคลุมการทำงานอย่างหลากหลาย ตัวแปรอินเทอร์พรีเตอร์ ของภาษาไพธอนนั้นมีให้ใช้ในหลายระบบปฏิบัติการ ทำให้โค้ดของภาษาไพธอนสามารถรันในระบบต่างๆ ได้อย่างกว้างขวาง มีสัญลักษณ์ของโปรแกรมไพธอน แสดงดังรูปที่ 2.4

รูปที่ 2.4 สัญลักษณ์ของโปรแกรมไพธอน
(ที่มา: http://marcuscode.com/lang/python)

2.4 ไลบรารี่โอเพ่นซีวี

โอเพ่นซีวี[5] (Open source Computer Vision : OpenCV) เป็นไลบรารีฟังก์ชันการเขียนโปรแกรม โดยส่วนใหญ่จะมุ่งเป้าไปที่การแสดงผลด้วยคอมพิวเตอร์แบบเรียลไทม์ (Real-Time Computer Vision) เดิมทีแล้วถูกพัฒนาโดยอินเทล (Intel) แต่ภายหลังได้รับการสนับสนุนโดย Willow Garage โอเพ่นซีวีเป็นไลบรารีแบบข้ามแพลตฟอร์ม และใช้งานได้ฟรีภายใต้ลิขสิทธิ์ของ BSD แบบโอเพ่นซอร์ส (Open-Source BSD License) โดยที่มีสัญลักษณ์ของโอเพ่นซีวี แสดงได้ดังรูปที่ 2.5

รูปที่ 2.5 สัญลักษณ์ของโอเพ่นซีวี
(ที่มา: www.medium.com/@nut.ch40/opencv-คืออะไร-8771e2a4c414)

โอเพ่นซีวี ถูกเขียนขึ้นด้วยภาษาซีพลัสพลัส มีการรองรับภาษาไพธอน จาวา และแมทแลป สำหรับอินเทอร์เฟสเหล่านี้สามารถพบได้ในเอกสารออนไลน์ ซึ่งมีการรวมไว้หลากหลายภาษา ได้รับการพัฒนาเพื่อส่งเสริมการนำมาใช้งานโดยผู้ใช้ที่เพิ่มขึ้น

2.5 ราสเบอรี่พายแคเมอร่าโมดูลเวอชั่น 2

ราสเบอรี่พายแคเมอร่าโมดูลเวอชั่น 2[6] เป็นโมดูลกล้องสำหรับต่อใช้งานร่วมกับบอร์ดราสเบอร์รี่พาย ขนาดความละเอียด 8 ล้านพิกเซล (pixel) สามารถถ่ายวิดีโอ ที่ความละเอียด 1080p, 720p และ 640x480 ด้วยอัตราแสดงผล 30 (1080p), 60 (720p และ 640x480)และ 90 (640x480) เฟรมต่อวินาที ราสเบอร์รี่พายแคเมอร่าโมดูลเวอชั่น 2 แสดงดังรูปที่ 2.6

รูปที่ 2.6 ราสเบอร์รี่พายแคเมอร่าโมดูลเวอชั่น 2

คุณสมบัติ

1. ภาพถ่ายมีความละเอียดสูงถึง 8 ล้านพิกเซล

2. ภาพวิดีโอมีคุณภาพระดับ HD ความคมชัด 1080p, 720p และ 640x480 ด้วยอัตราการแสดงผล 30 (1080p), 60 (720p และ 640x480) และ 90 (640x480) เฟรมต่อวินาที

3. ขนาดของโมดูลกล้อง 25 x 20 x 9 ม.ม. และน้ำหนัก 3 กรัม
4. การเชื่อมต่อกับบอร์ดราสเบอรี่พายด้วยบัสซีเอสไอ (Common System Interface : CSI)
5. ภาพมีความคมชัดสูงเมื่อถ่ายในที่มีแสงเหมาะสมและระยะ 1.5 เมตรขึ้นไป
6. ใช้กระแสไฟฟ้าต่ำเหมาะกับอุปกรณ์กล้องที่ต้องการถ่ายข้อมูลจำนวนมากอย่างรวดเร็ว
7. สายแพ 15 จุดต่อ

2.6 การประมวลผลภาพ

การประมวลผลภาพ[7] เป็นการนำข้อมูลชนิดรูปภาพมาทำการประมวลผลหรือคิดคำนวณด้วยคอมพิวเตอร์ เพื่อให้ได้ผลลัพธ์ของข้อมูลตามที่เราต้องการทั้งในเชิงคุณภาพและปริมาณ โดยแสดงดังรูปที่ 2.7

รูปที่ 2.7 การประมวลผลภาพ

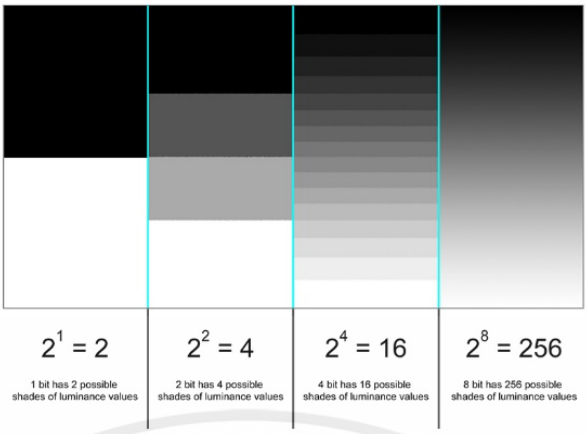
โดยมีขั้นตอนต่างๆ ที่สำคัญ คือ การทำให้ภาพมีความคมชัดมากขึ้น การกำจัดสัญญาณรบกวนออกจากภาพ การแบ่งส่วนของวัตถุที่เราสนใจออกมาจากภาพ เพื่อนำภาพวัตถุที่ได้ไปวิเคราะห์หาข้อมูลเชิงปริมาณ เช่น ขนาด รูปร่าง และทิศทางการเคลื่อนของวัตถุในภาพ จากนั้นเราสามารถนำข้อมูลเชิงปริมาณเหล่านี้ไปวิเคราะห์ และสร้างเป็นระบบ เพื่อใช้ประโยชน์ในงานด้านต่างๆ

2.6.1 ระบบสี

ระบบสี[8] (Color Model) ของภาพจะมีองค์ประกอบของสีหรือเฉดสีที่แตกต่างกัน แบ่งออกเป็น 3 แบบดังนี้

2.6.1.1 ระบบสีเทา

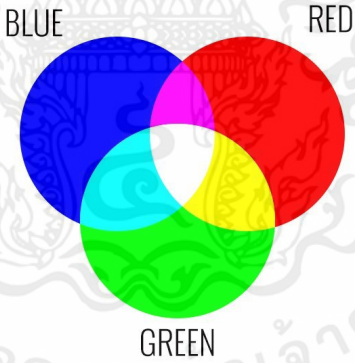
ซึ่งแตกต่างกับภาพขาว-ดำที่มีเพียง 2 สี คือขาวกับดำ สีในระบบสีเทานี้แสดงถึงความเข้มของสี (Intensity) ในระดับต่าง ๆ จำนวนระดับของสีขึ้นอยู่กับขนาดของบิตที่ใช้เก็บค่าสี โดยทั่วไปแล้วจะเก็บข้อมูลสีประเภทนี้ด้วยข้อมูลขนาด 8 บิต หรือ 1 ไบต์ ซึ่งจะให้ความละเอียดของสีที่ 256 เฉดสี โดยสีดำเป็นส่วนที่มีความเข้มของสีน้อย และสีขาวจะมีความเข้มของสีมาก แสดงดังรูปที่ 2.8



รูปที่ 2.8 ระบบสีเทา

(ที่มา : http://dozzdiy.blogspot.com/2014/11/photoshop-cc-photoshop-cc-8-16.html)

2.6.1.2 ระบบสีอาร์จีบี (RGB) เป็นระบบสีที่ประกอบด้วยแม่สีหลักสามสี คือ แดง (Red), เขียว (Green) และน้ำเงิน (Blue) เมื่อนำมาผสมผสานกันทำให้เกิดสีต่างๆ บนจอคอมพิวเตอร์มากถึง 16.7 ล้านสี ซึ่งใกล้เคียงกับสีที่ตาเรามองเห็นปกติ สีที่ได้จากการผสมสีขึ้นอยู่กับความเข้มของสี โดยถ้าสีมีความเข้มมาก เมื่อนำมาผสมกันจะทำให้เกิดเป็นสีขาว แสดงดังรูปที่ 2.9

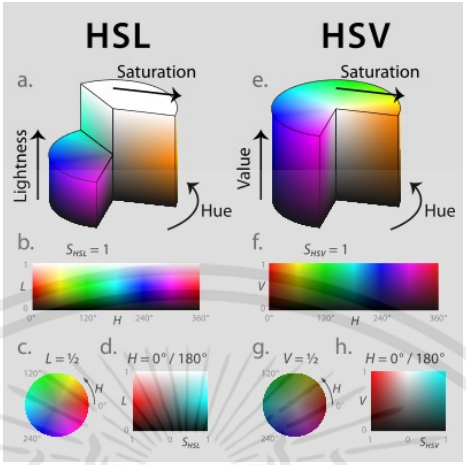


รูปที่ 2.9 ระบบสีอาร์จีบี

(ที่มา : https://www.lalapixthailand.com/2018/12/28/whats-rgb-and-cmyk-color)

2.6.1.3 ระบบสีเอชเอสวี (HSV) เป็นระบบสีแบบการมองเห็นของสายตามนุษย์ ซึ่งแบ่งออกเป็น 3 ส่วน คือ Hue คือ สีต่างๆ ที่สะท้อนออกมาจากวัตถุแล้วเข้าสู่สายตาของเรา ซึ่งมักจะเรียกสีตามชื่อสี เช่น สีเขียว สีเหลือง สีแดง เป็นต้น Saturation คือ ความสดของสี โดยค่าความสดของสีจะเริ่มที่ 0 ถึง 100 ถ้ากำหนด Saturation ที่ 0 สีจะมีความสดน้อย แต่ถ้ากำหนดที่ 100 สีจะมีความสดมาก Brightness คือ ระดับความสว่างของสี โดยค่าความสว่างของสีจะเริ่มที่ 0 ถึง

100 ถ้ากำหนดที่ 0 ความสว่างจะน้อยซึ่งจะเป็นสีดำ แต่ถ้ากำหนดที่ 100 สีจะมีความสว่างมากที่สุด แสดงดังรูปที่ 2.10

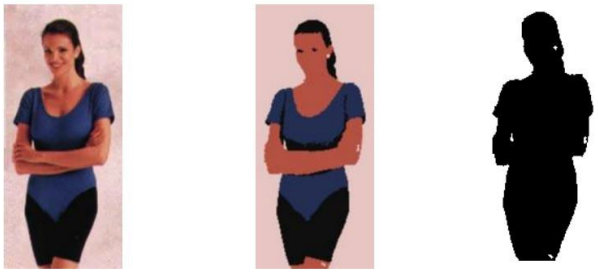


รูปที่ 2.10 ระบบสีเอชเอสวี

(ที่มา : https://medium.com/@kongruksiamza)

2.6.2 การทำเทรสโฮลด์

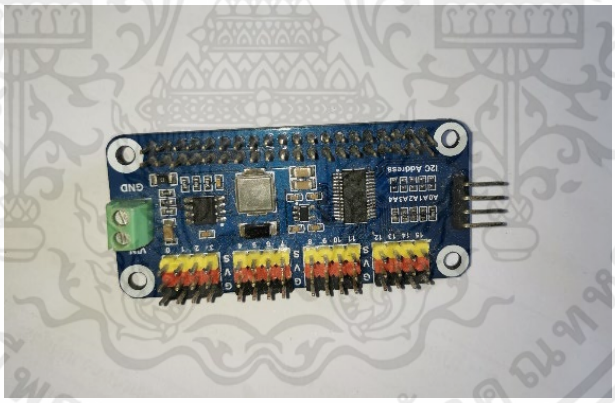
การทำเทรสโฮลด์ (Thresholding) คือ การดึงวัตถุพื้นหน้าออกจากพื้นหลัง เป็นกระบวนการที่ใช้ในการแยกวัตถุ หรือแยกองค์ประกอบต่าง ๆ ออกจากภาพที่ต้องการ วิธีการทำเทรสโฮลด์เป็นการแยกวัตถุจากออกฉากหลังโดยดูจากความเข้มแสงของพิกเซลเป็นหลัก เทคนิคการทำการเทรสโฮลด์ เป็นการพิจารณาว่าจุดใดในภาพควรเป็นจุดขาวหรือเป็นจุดดำ ซึ่งทำได้โดยการเปรียบเทียบระหว่างจุดภาพเริ่มต้นกับค่าคงที่ค่าหนึ่ง เรียกว่าค่าขีดแบ่ง ซึ่งเป็นค่าความเข้มแสงค่าหนึ่งที่ใช้แยกแยะประเภทของจุดภาพ เทคนิคนี้ใช้กันมากในกรณีที่ข้อมูลภาพมีลักษณะแตกต่างกันระหว่างวัตถุและพื้นหลัง ในขั้นตอนนี้ทำการตัดพื้นหลังออกจากพื้นหน้าโดยใช้วิธีเทรสโฮลด์ที่มีระดับความเข้มอยู่ระหว่างกลุ่มทั้งสองของ ฮิสโทรแกรม (Histogram) ซึ่งค่าเทรสโฮลด์ที่ได้จะอยู่ระหว่าง 0-255 เท่านั้น ค่าเทรสโฮลด์จะถูกนำไปเพื่อเปรียบเทียบค่าของแต่ละพิกเซล หากค่า f (x, y) น้อยกว่าค่าเทรสโฮลด์ จุดพิกเซลนั้นจะถูกปรับให้เป็นสีดำหรือส่วนของวัตถุ และหากค่า f (x, y) มากกว่าหรือเท่ากับค่าเทรสโฮลด์ จุดพิกเซลนั้นจะถูกปรับให้เป็นสีขาวหรือส่วนของพื้นหลังแสดงดังรูปที่ 2.11



รูปที่ 2.11 การทำเทรสโฮลด์
(ที่มา : https://nextsoftwares.wordpress.com)

2.7 บอร์ดขับเซอร์โวมอเตอร์

บอร์ดควบคุมเซอร์โวมอเตอร์[9] บอร์ดที่สามารถควบคุมเซอร์โวมอเตอร์ได้ทีละ หลายๆตัวพร้อมกัน เป็นโมดูลขับมอเตอร์เซอโว 16 ช่อง 12 bit แบบพีดับบลิวเอ็ม ติดต่อแบบอินเตอร์เฟส ไอสแควซี ใช้ ไอซีเบอร์ PCA9685 ในการควบคุม ซึ่งสามารถควบคุมเซอโวมอเตอร์ ได้ 16 ตัว โดยการกำหนดตำแหน่ง ของไอสแควซีให้ต่างกัน ถ้าต่อใช้งานโมดูลเดียวจะควบคุมได้ 16 ตัว บอร์ดขับเซอร์โวมอเตอร์ แสดงดังรูปที่ 2.12



รูปที่ 2.12 บอร์ดขับเซอร์โวมอเตอร์

การต่อใช้งาน

1. GND คือ กราวนด์จำเป็นต้องต่อใช้งานกับขากราวนด์ของไมโครคอนโทรลเลอร์ทุกครั้ง

2. VCC คือ เพาเวอร์เชื่อมต่อกับแหล่งจ่ายเพื่อเป็นไฟเลี้ยงให้กับบอร์ดขับเซอร์โวระดับแรงดันแหล่งจ่ายจะอยู่ที่ประมาณ 3-5 โวลต์ นอกจากนี้ยังต่อใช้งานกับขาเอสดีเอ (Serial Data : SDA) และเอสซีแอล(Serial Clock : SCL) โดยมีตัวต้านทาน 10 กิโลโอห์มต่อใช้งานแบบพูลอัพอยู่ด้วย

3. V+ คือ ขาที่ต่อใช้งานกับแหล่งจ่ายเพื่อเป็นแหล่งจ่ายเสริมให้กับเซอร์โวมอเตอร์แหล่งจ่ายที่เหมาะสม คือ 5-6 โวลต์ สามารถรับแรงสูงสุดได้มากกว่า 12 โวลต์ ข้อควรระวังคือ ถ้าต่อใช้งานผิดพลาดและต่อใช้งานขา VCC กับขา V+ ร่วมกันอาจทำให้บอร์ดเสียหายได้

4. เอสซีแอล คือ ขาสัญญาณนาฬิกาต่อใช้งานกับขาเอสซีแอลของไมโครคอนโทรลเลอร์ มีข้อเสียคือต้องพูลอัพกับขา VCC

5. เอสดีเอ คือ ขาเชื่อมต่อสื่อสารข้อมูลซึ่งต่อใช้งานกับขาเอสดีเอของไมโครคอนโทรเลอร์มีข้อเสียคือต้องพูลอัพกับขา VCC

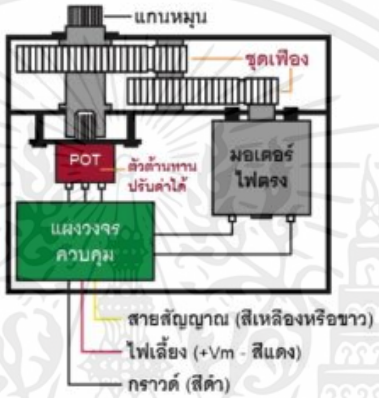
2.8 เซอร์โวมอเตอร์

เซอร์โวมอเตอร์[10] เป็นอุปกรณ์แม่เหล็กไฟฟ้าแบบหนึ่งที่ใช้ในการหมุนตัวขับ (actuator) ไปยังตำแหน่งต่างๆ ด้วยความแม่นยำ โดยใช้สัญญาณพัลส์เพื่อกำหนดตำแหน่งในการหมุน มักนิยมใช้ในรถบังคับวิทยุ เครื่องบินบังคับวิทยุ หรือใช้ควบคุมแขนขาของหุ่นยนต์ ส่วนใหญ่จะรู้จักกันภายใต้ชื่อว่า อาร์ซีเซอร์โวมอเตอร์ โดยคำว่าอาร์ซี (Radio Control : RC) หรือการบังคับด้วยวิทยุ เนื่องจากในยุคแรกๆ ของการพัฒนาเซอร์โวมอเตอร์ จะถูกนำมาใช้ในงานวิทยุบังคับเป็นหลัก ปกติแล้วเซอร์โวมอเตอร์ที่ยังไม่ได้รับการปรับแต่งใดๆ นั้นจะใช้ในการควบคุมตำแหน่งของอุปกรณ์ เช่น การบังคับเลี้ยวของรถบังคับวิทยุ หรือใช้สำหรับปรับหางเสือของเรือหรือ เครื่องบิน ซึ่งงานเหล่านี้ต้องการแรงบิดของมอเตอร์ที่สูงพอสมควร ดังนั้นเซอร์โวมอเตอร์จึงต้องมีอัตราทดที่มากพอ เพื่อให้สามารถรองรับงานดังกล่าวได้ เซอร์โวมอเตอร์มาตรฐานจะมีมุมในการหมุนอยู่ระหว่าง 90 ถึง 180 องศา แล้วแต่ ผู้ผลิต แต่ที่นิยมมากที่สุดคือ 0 ถึง 180 องศา และในบางรุ่นของบางผู้ผลิตจะสามารถดัดแปลง ให้หมุนได้ครบ 360 องศา เซอร์โวมอเตอร์แสดงดังรูปที่ 2.13

รูปที่ 2.13 เซอร์โวมอเตอร์

2.8.1 โครงสร้างของเซอร์โวมอเตอร์

ภายในเซอร์โวมอเตอร์ประกอบด้วย มอเตอร์ไฟตรงขนาดเล็ก,ชุดเฟืองทด, แผงวงจรควบคุม และตัวต้านทานปรับค่าได้ (Potentiometer : POT) โดยแผงวงจรควบคุมจะมีวงจรป้อนกลับ เพื่อให้เซอร์โวมอเตอร์รับรู้ตำแหน่งของตัวเองได้ โดยผู้ใช้งานเพียงส่งสัญญาณพัลส์ออกไปควบคุมเท่านั้น ดังแสดงไดอะแกรมการทำงานของเซอร์โวมอเตอร์ในรูปที่ 1 แกนของมอเตอร์ไฟตรงจะต่อเข้ากับ ชุดเฟืองเพื่อลดความเร็วรอบลงส่งผลให้แรงบิดที่แกนหมุนมากขึ้น ทั้งหมดทำงานร่วมกันภายใต้ ความสัมพันธ์ โดยมีบล็อกไดอะแกรมการทำงานของแผงวงจรควบคุมในเซอร์โวมอเตอร์แสดงดังรูปที่ 2.14



รูปที่ 2.14 ไดอะแกรมการทำงานของเซอร์โวมอเตอร์
(ที่มา: www.inventor.in.th/home/เซอร์โวมอเตอร์)

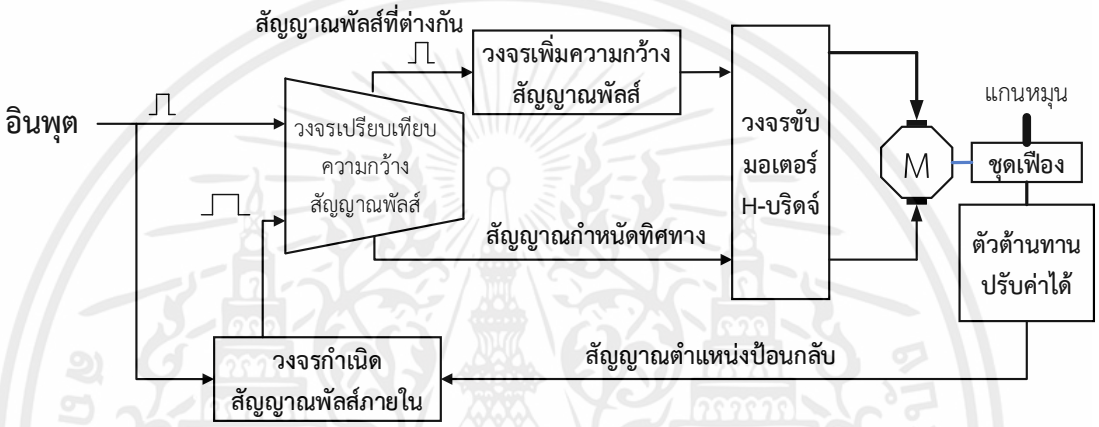
2.8.2 คุณสมบัติทางเทคนิคที่สำคัญของเซอร์โวมอเตอร์

มี 2 ค่าคือ ความเร็ว (speed) และแรงบิดหรือทอร์ค (torque) ความเร็วหมายถึงระยะเวลาที่ทำให้แกนหมุนของมอเตอร์เคลื่อนที่สู่ตำแหน่งมุมที่กำหนด อาทิ เซอร์โวมอเตอร์ ตัวหนึ่งมีความเร็ว 0.15 วินาทีสำหรับ 60 องศา หมายถึงเซอร์โวมอเตอร์ตัวนี้สามารถขับให้แกนหมุนเคลื่อนที่ไปยังตำแหน่งมุม 60 องศาภายในเวลา 0.15 วินาที ส่วนแรงบิดมักจะปรากฏในหน่วยของออนซ์-นิ้ว (ounce-inches : oz-in) หรือ กิโลกรัม-เซนติเมตร (kg-cm) เป็นคุณสมบัติที่จะบอกต่อผู้ใช้งาน ว่าเซอร์โวมอเตอร์ตัวนี้มีแรงในการขับโหลดที่มีน้ำหนักในหน่วยออนซ์ให้สามารถเคลื่อนที่ไปได้ 1 นิ้ว หรือน้ำหนักในหน่วยกิโลกรัมให้เคลื่อนที่ ไปได้ 1 เซนติเมตร (น้ำหนัก 1 ออนซ์เท่ากับ 0.028 กิโลกรัมโดยประมาณ หรือ 1 กิโลกรัม เท่ากับ 35.274 ออนซ์)

ค่าของความเร็วและแรงบิด ต้องสัมพันธ์กับแรงดันไฟเลี้ยงที่จ่ายให้แก่เซอร์โวมอเตอร์ด้วย ซึ่งมักจะแรงดัน 4.8 หรือ 6V นอกจากนั้นยังมีปัจจัยเกี่ยวกับแรง เสียดทานในระบบเฟืองภายในเซอร์โวมอเตอร์ การหล่อลื่นการเชื่อมโยงระหว่างเฟืองต่อเฟืองในชุดเฟืองทด ที่ส่งผลให้ความเร็วและแรงบิดของ เซอร์โวมอเตอร์เปลี่ยนแปลงไปได้

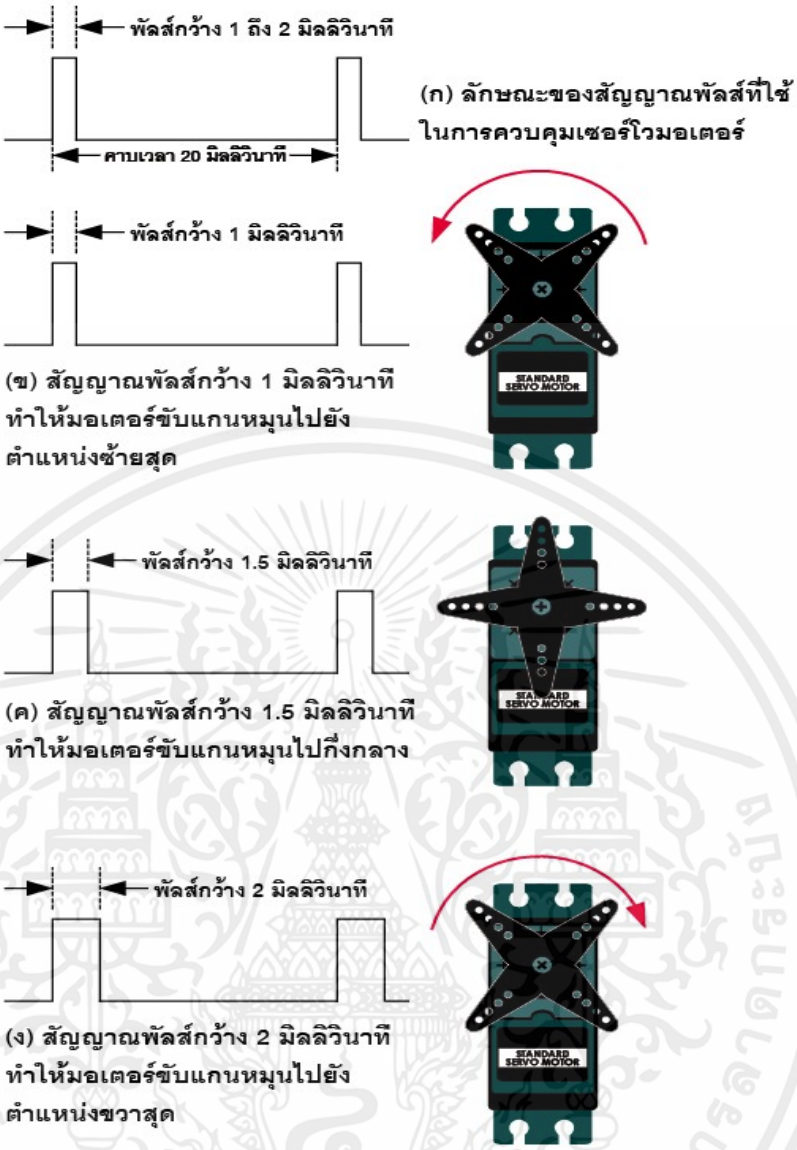
2.8.3 การทำงานของแผงวงจรควบคุมในเซอร์โวมอเตอร์ชนิดอนาล็อก

การหมุนของเซอร์โวมอเตอร์นั้นจะไม่ได้หมุนเป็นอิสระเหมือนมอเตอร์ทั่วไป โดยช่วงระยะการหมุนปกติจะอยู่ระหว่าง 90 ถึง 180 องศา ตำแหน่งการหมุนของแกนมอเตอร์ใน เซอร์โวมอเตอร์นี้สามารถควบคุมได้อย่างแม่นยำ เนื่องจากภายในเซอร์โวมอเตอร์มีวงจรอิเล็กทรอนิกส์ทำหน้าที่ตรวจสอบตำแหน่งของเซอร์โวมอเตอร์อยู่ตลอดเวลา ลักษณะการตรวจสอบจะใช้การป้อนกลับค่าตำแหน่งจากตัวต้านทานปรับค่าได้ แล้วนำค่านี้ไป เปรียบเทียบกับค่าพัลส์ที่ป้อนเข้าทางขาควบคุม ค่าของผลต่างที่ได้จะไปปรับตำแหน่งของมอเตอร์ ค่าผลต่างก็จะได้ตำแหน่งของมอเตอร์ที่แม่นยำ



รูปที่ 2.15 บล็อกไดอะแกรมการทำงานของแผงวงจรควบคุมในเซอร์โวมอเตอร์ชนิดอนาล็อก

จากรูปที่ 2.15 แสดงการทำงานของวงจรควบคุมการทำงานในเซอร์โวมอเตอร์ชนิดอนาล็อก สัญญาณพัลส์ควบคุมที่ส่งเข้ามาทางอินพุต จะถูกส่งไปยังวงจรกำเนิดสัญญาณพัลส์ภายในด้วย โดยมีความกว้างที่เป็น สัดส่วนกับตำแหน่งของแกนหมุนในปัจจุบัน ทั้งสัญญาณพัลส์ที่กำเนิดขึ้นภายในกับสัญญาณพัลส์ควบคุมจะถูกส่งไปยังวงจรเปรียบเทียบเพื่อทำการหักล้างสัญญาณ โดยทิศทางของสัญญาณจะขึ้นอยู่กับว่า ระหว่างสัญญาณพัลส์ควบคุมทางอินพุตกับสัญญาณพัลส์ภายในสัญญาณพัลส์ใดมีความกว้างมากกว่า โดยเอาต์พุตที่ได้เป็นสัญญาณลอจิก “0” หรือ “1” แล้วส่งไปยังวงจรขับมอเตอร์แบบ H-บริดจ์ เพื่อกำหนดทิศทางการหมุน ทางด้านค่าความแตกต่างที่เกิดขึ้นระหว่างพัลส์ทั้งสองสัญญาณจะถูกส่งไปยังวงจรเพิ่มความกว้างพัลส์ เพื่อสร้างสัญญาณพัลส์สำหรับส่งไปขับมอเตอร์ ผ่านวงจรขับมอเตอร์แบบ H-บริดจ์ โดยความแตกต่างของความกว้างพัลส์ 1% ทำให้เกิดสัญญาณพัลส์สำหรับขับมอเตอร์ในระดับ 50% และความเร็วนี้จะลดลงเมื่อแกนหมุนของมอเตอร์เคลื่อนที่เข้าสู่ตำแหน่งที่กำหนด อันเป็นผลมาจากความแตกต่างของความกว้างสัญญาณพัลส์เริ่มลดลง และหยุดลงเมื่อสัญญาณพัลส์ที่นำมาเปรียบเทียบมีค่าความกว้างเท่ากัน ลักษณะของสัญญาณพัลส์ที่ใช้ในการควบคุมเซอร์โวมอเตอร์ แสดงดังรูปที่ 2.16



รูปที่ 2.16 ลักษณะของสัญญาณพัลส์ที่ใช้ในการควบคุมเซอร์โวมอเตอร์
(ที่มา: www.inventor.in.th/home/เซอร์โวมอเตอร์)

2.8.4 รูปแบบสัญญาณที่ใช้ควบคุมเซอร์โวมอเตอร์

การควบคุมเซอร์โวมอเตอร์ทำได้โดยสร้างสัญญาณพัลส์ที่มีคาบเวลา 20 มิลลิวินาที ป้อนให้กับวงจรควบคุมภายในเซอร์โวมอเตอร์ดังรูปที่ 5 แล้วปรับความกว้างของพัลส์ช่วงบวก ที่พัลส์กว้าง 1 มิลลิวินาที มอเตอร์จะหมุนไปตำแหน่งซ้ายมือสุด ถ้าส่งพัลส์กว้าง 1.5 มิลลิวินาที แกนหมุนของมอเตอร์จะเคลื่อนที่ไปยังตำแหน่งกึ่งกลาง และถ้าส่งพัลส์กว้าง 2 มิลลิวินาที แกนหมุนของมอเตอร์จะเคลื่อนที่ไปยังตำแหน่งขวามือสุด การป้อนสัญญาณพัลส์ที่มีคาบเวลาช่วงบวกตั้งแต่ 1.5 ถึง 2 มิลลิวินาทีจะทำให้เซอร์โวมอเตอร์หมุนทวนเข็มนาฬิกา โดยถ้าค่าความกว้างพัลส์ยิ่งห่างจาก 1.5

มิลลิวินาที มากเท่าใด ความเร็วในการหมุนก็จะมากขึ้นเท่านั้น นั่นคือ ความเร็วสูงสุดของการหมุนทวนเข็มนาฬิกาจะเกิดขึ้นเมื่อสัญญาณพัลส์ควบคุมมีความกว้าง 2 มิลลิวินาที การป้อนสัญญาณพัลส์ที่มีคาบเวลาช่วงบวกตั้งแต่ 1 ไปจนถึง 1.5 มิลลิวินาที ทำให้เซอร์โวมอเตอร์หมุนตามเข็มนาฬิกา ซึ่งถ้าค่าความกว้างพัลส์เข้าใกล้ 1 มิลลิวินาทีความเร็วในการหมุนของเซอร์โวมอเตอร์ก็จะมาก นั่นคือ ความเร็วสูงสุดของการหมุนตามเข็มนาฬิกา จะเกิดขึ้นเมื่อสัญญาณพัลส์ควบคุมมีความกว้าง 1 มิลลิวินาที

2.9 มอเตอร์ทดเกียร์

มอเตอร์ทดเกียร์[11] คือ เครื่องมืออุปกรณ์วัตถุไฟฟ้าชนิดหนึ่งที่มักจะใช้กันตามโรงงานหรือมักใช้กันตามกลุ่มวงการอุตสาหกรรม เครื่องมือการใช้งานเหล่านี้ จะทำหน้าที่เปลี่ยนจากพลังงานไฟฟ้าให้เป็นพลังงานกลแทน อุปกรณ์มอเตอร์ทดเกียร์นั้น มีขนาดหลากหลายรูปแบบ มีตั้งแต่ ขนาดเล็ก ขนาดใหญ่ ในแต่ละขนาดจะมีรอบ กำลังที่แตกต่างกันแล้วแต่ตามความต้องการในการใช้งานของแต่ละบุคคล สำหรับลักษณะงานของการใช้มอเตอร์แบบทดเกียร์นั้น ก็มักจะใช้ในลักษณะของการลำเลียงให้เคลื่อนที่ไปอย่างช้า อย่างเช่นพวกบันไดเลื่อน ลิฟต์ โดยมอเตอร์ทดเกียร์ไฟฟ้ากระแสตรงสามารถดูได้ ดังรูปที่ 2.17

รูปที่ 2.17 มอเตอร์ทดเกียร์ไฟฟ้ากระแสตรง

2.9.1 การแบ่งชนิดของมอเตอร์ไฟฟ้ากระแสตรง

มอเตอร์ไฟฟ้ากระแสตรงแบ่งออกได้ดังนี้

1. มอเตอร์แบบอนุกรมหรือเรียกว่าซีรีส์มอเตอร์ (Series Motor)
2. มอเตอร์แบบอนุขนานหรือเรียกว่าชันท์มอเตอร์ (Shunt Motor)
3. มอเตอร์ไฟฟ้าแบบผสมหรือเรียกว่าคอมปาวด์มอเตอร์ (Compound Motor)

มอเตอร์ไฟฟ้ากระแสตรงเป็นต้นกาลังขับเคลื่อนที่สำคัญอย่างหนึ่งในโรงงานอุตสาหกรรมเพราะมีคุณสมบัติที่ดีในด้านการปรับความเร็วได้ตั้งแต่แต่ความเร็วต่าจนถึงสูงสุด นิยมใช้กันมากในโรงงานอุตสาหกรรม เช่นโรงงานทอผ้าโรงงานเส้นใยโพลีเอสเตอร์โรงงานถลุง โลหะหรือให้

เป็นต้นกำลังในการขับเคลื่อนรถไฟฟ้าเป็นต้น ในการศึกษาเกี่ยวกับมอเตอร์ไฟฟ้ากระแสตรงจึงควรรู้จักอุปกรณ์ต่างๆ ของมอเตอร์ไฟฟ้ากระแสตรงและเข้าใจถึงหลักการทางานของมอเตอร์ไฟฟ้ากระแสตรงแบบต่างๆ

2.9.2 หลักการทำงานมอเตอร์ไฟฟ้ากระแสตรง

หลักการของมอเตอร์ไฟฟ้ากระแสตรง เมื่อมีแรงดันกระแสไฟฟ้าตรงเข้าไปในมอเตอร์ ส่วนหนึ่งจะผ่านแปรงถ่านคอมมิวเตเตอร์เข้าไปในขดลวดอาร์มาเจอร์สร้างสนามแม่เหล็กขึ้นและกระแสไฟฟ้าอีกส่วนหนึ่งจะไหลเข้าไปในขดลวดสนามแม่เหล็ก (Field coil) สร้างขั้วเหนือขั้วใต้ในขณะเดียวกันตามคุณสมบัติของเส้นแรงแม่เหล็กจะไม่ตัดกัน ทิศทางตรงข้ามจะหักล้างกันและทิศทางเดียวจะเสริมแรงกันทำให้เกิดแรงบิดในตัวอาร์มาเจอร์ซึ่งวางแกนเพลาและแกนเพลานี้สวมอยู่กับตลับลูกปืนของมอเตอร์ทำให้อาร์มาเจอร์หมุนได้ขณะที่ตัวอาร์มาเจอร์ทาหน้าที่หมุนได้เรียกว่า โรเตอร์ (Rotor) ซึ่งหมายความว่าตัวหมุนการที่อานาจเส้นแรงแม่เหล็กทั้งสองมีปฏิกิริยาต่อกันทำให้ขดลวดอาร์มาเจอร์หรือโรเตอร์หมุนไปนั้นเป็นไปตามกฎมือซ้ายของเฟลมมิ่ง (Fleming left hand rule)

2.10 เซ็นเซอร์ตรวจจับวัตถุ

เซ็นเซอร์ตรวจจับวัตถุ[12] เป็นเซ็นเซอร์ที่ใช้หลักการยิงแสงอินฟาเรดออกไปหากมีวัตถุขวางจะมีการสะท้อนกลับมาเพื่อบอกว่าขณะนี้มีสิ่งกีดขวางหรือวัตถุอยู่หน้าเซนเซอร์หรือไม่ ซึ่งมีระยะทำการอยู่ที่ 3-80 เซนติเมตร โดยใช้ไฟเลี้ยง 5 โวลต์ ซึ่งให้ค่าออกมาเป็นสัญญาณ on-off โดย off คือมีสิ่งกีดขวาง on คือไม่มีสิ่งกีดขวาง ขนาดของ Sensor 17X45 มิลลิเมตร อุณหภูมิในการทำงาน -25-55 องศาเซลเซียส มุมในการวัด 15 องศา ความเร็วในการวัด Response time < 2 มิลลิวินาที เซ็นเซอร์ตรวจจับวัตถุ แสดงดังรูปที่ 2.18

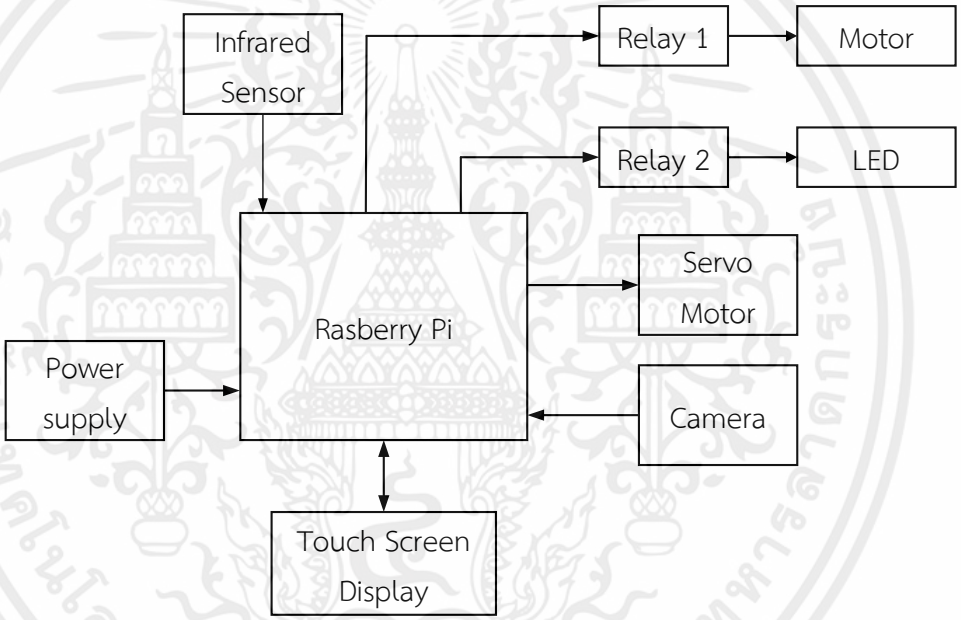
รูปที่ 2.18 เซ็นเซอร์ตรวจจับวัตถุ

บทที่ 3
การออกแบบ

ในบทนี้จะกล่าวถึง การออกแบบเครื่องคัดแยกสีและขนาดของผลมะนาวกึ่งอัตโนมัติที่ผู้จัดทำได้ออกแบบซึ่งประกอบไปด้วย บล็อกไดอะแกรมของการทำงาน แผนผังแสดงการทำงานทั้งหมด การออกแบบวงจร การกำหนดค่าระบบสีเอชเอสวี และการออกแบบเครื่อง

3.1 บล็อกไดอะแกรม

3.1.1 บล็อกไดอะแกรมเครื่องคัดแยกสีและขนาดของผลมะนาว



รูปที่ 3.1 บล็อกไดอะแกรมของเครื่องคัดแยกสีและขนาดของผลมะนาว

จากรูปที่ 3.1 สามารถอธิบายบล็อกไดอะแกรมของเครื่องคัดแยกสีและขนาดของผลมะนาวได้ดังนี้

1. แหล่งจ่ายไฟฟ้ากระแสตรง (Power supply) มีหน้าที่แปลงไฟฟ้ากระแสสลับเป็นไฟฟ้ากระแสตรง เพื่อเป็นแหล่งจ่ายไฟให้กับอุปกรณ์ต่าง ๆ สามารถทำงานได้

2. ราสเบอรี่พาย (Raspberry Pi) เป็นคอมพิวเตอร์ขนาดเล็กใช้ควบคุมการทำงานของอุปกรณ์หรือกระบวนการทำงานต่าง ๆ โดยสามารถรับอินพุตหรือส่งค่าเอาต์พุตได้ผ่านพอร์ตจีพีไอโอ (General Purpose Input/Output : GPIO) เพื่อควบคุมอุปกรณ์ต่างๆ

3. จอสัมผัส (Touch Screen Display) มีหน้าที่ส่งและรับข้อมูลต่าง ๆ จากผู้ใช้งานกับบอร์ดราสเบอรี่พายเป็นอินพุตหรือเอาต์พุต แสดงเป็นแบบกราฟิก รูปภาพ ค่าที่เป็นตัวเลข หรืออื่น ๆ บนหน้าจอ

4. อินฟาเรดเซ็นเซอร์ (Infrared Sensor) คืออุปกรณ์ทำหน้าที่นับจำนวนผลมะนาวแต่ละขนาดเพื่อนำไปแสดงผลบนหน้าจอสัมผัส

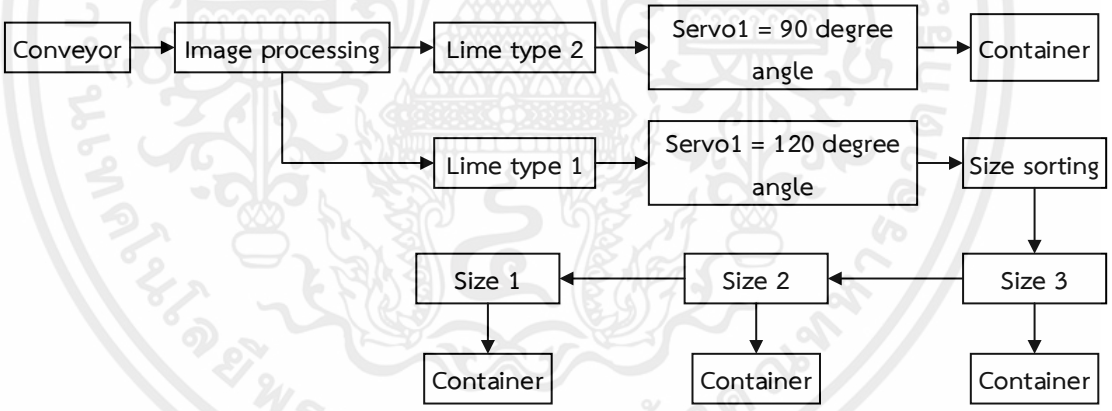
5. มอเตอร์ (Motor) คืออุปกรณ์ที่เปลี่ยนพลังงานไฟฟ้าเป็นพลังงานกลใช้ในการขับเคลื่อนสายพาน

6. แอลอีดี (LED) คืออุปกรณ์ให้ความสว่างกับตัวกล้องเพื่อให้กล้องสามารถตรวจจับจับผลมะนาวได้

7. กล้อง (Camera) มีหน้าที่รับข้อมูลภาพ และส่งไปให้บอร์ดราสเบอรี่พายทำการประมวลผลภาพเพื่อแยกสีของผลมะนาว

8. เซอร์โวมอเตอร์ (Servo Motor) คืออุปกรณ์ที่ใช้ในการคัดแยกสีผลมะนาวจากผลลัพธ์ของการประมวลผลภาพ

3.1.2 บล็อกไดอะแกรมการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาว



รูปที่ 3.2 บล็อกไดอะแกรมการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาว

จากรูปที่ 3.2 สามารถอธิบายบล็อกไดอะแกรมการทำงานโดยรวมของเครื่องคัดแยกสีและขนาดของผลมะนาว ได้ดังนี้

1. ชุดลำเลียง (Conveyor) คือ ชุดลำเลียงผลมะนาวไปยังส่วนการประมวลผลภาพเพื่อคัดแยกสี

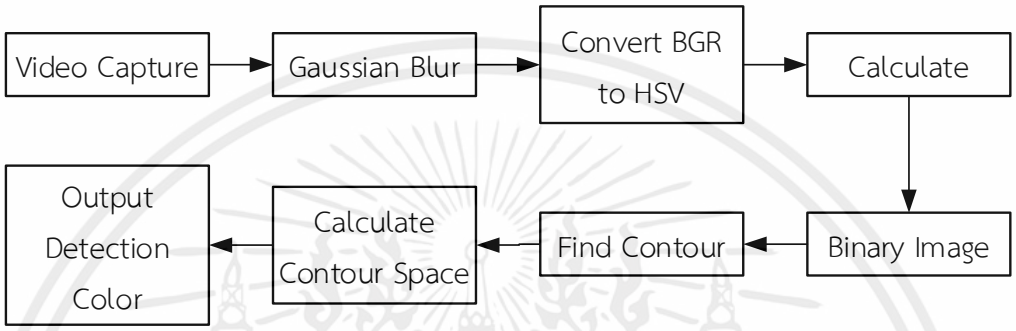
2. การประมวลผลภาพ (Image Processing) คือ ขั้นตอนการนำผลมะนาวมาคัดแยกสีเหลืองและสีเขียวออกจากกัน

3. ส่วนคัดแยกสี คือ ขั้นตอนการควบคุมการกลิ้งของผลมะนาวให้ไปยังทิศทางที่กำหนดไว้

4. ส่วนคัดแยกขนาด (Size sorting) คือ การนำผลมะนาวประเภทที่ 1 ที่ผ่านการประมวลผลภาพมาคัดแยกขนาดโดยแยกออกเป็น 3 ประเภท คือ มะนาวเบอร์ 1 มะนาวเบอร์ 2 และมะนาวเบอร์ 3

5. ภาชนะ (Container) คือ ภาชนะที่จะบรรจุผลมะนาว

3.1.3 บล็อกไดอะแกรมการประมวลผลภาพ



รูปที่ 3.3 บล็อกไดอะแกรมการประมวลผลภาพ

จากรูปที่ 3.3 สามารถอธิบายบล็อกไดอะแกรมการทำงานในส่วนของการประมวลผลภาพของเครื่องคัดแยกสีและขนาดของผลมะนาว ได้ดังนี้

1. จับภาพจากวิดีโอ (Video Capture) คือ การจับภาพหนึ่งภาพจากจากกล้องและนำภาพที่ได้ไปสู่กระบวนการลดสัญญาณรบกวน

2. การกรองสัญญาณรบกวน (Gaussian Blur) คือ การลดความผิดพลาดที่เกิดจากแสงหรือเงา โดยการทำการเบลอภาพเล็กน้อยเพื่อให้โปรแกรมไม่สามารถตรวจจับแสงหรือเงาขนาดเล็กได้

3. การเปลี่ยนแปลงระบบสี (Convert BGR to HSV) คือ การเปลี่ยนแปลงระบบสีจาก BGR ที่อาศัยหลักการผสมระหว่าสีเพื่อให้เกิดสีต่าง ๆ เป็นระบบสีแบบ HSV ซึ่งอาศัยหลักการกำหนดย่านของสีและเพิ่มความเข้มและความจางเพื่อให้ได้ระดับสีทุกระดับตามที่กำหนด

4. การคำนวณหาค่าสีในภาพ (Calculate) คือ การกาหนดช่วงความกว้างของระบบค่าสีเอชเอสวีที่ต้องการ เพื่อประมวลผลภาพ

5. ข้อมูลภาพฐานสอง (Binary Image) คือ ภาพที่มีข้อมูลสีเพียงสองสีคือสีดำและสีขาว

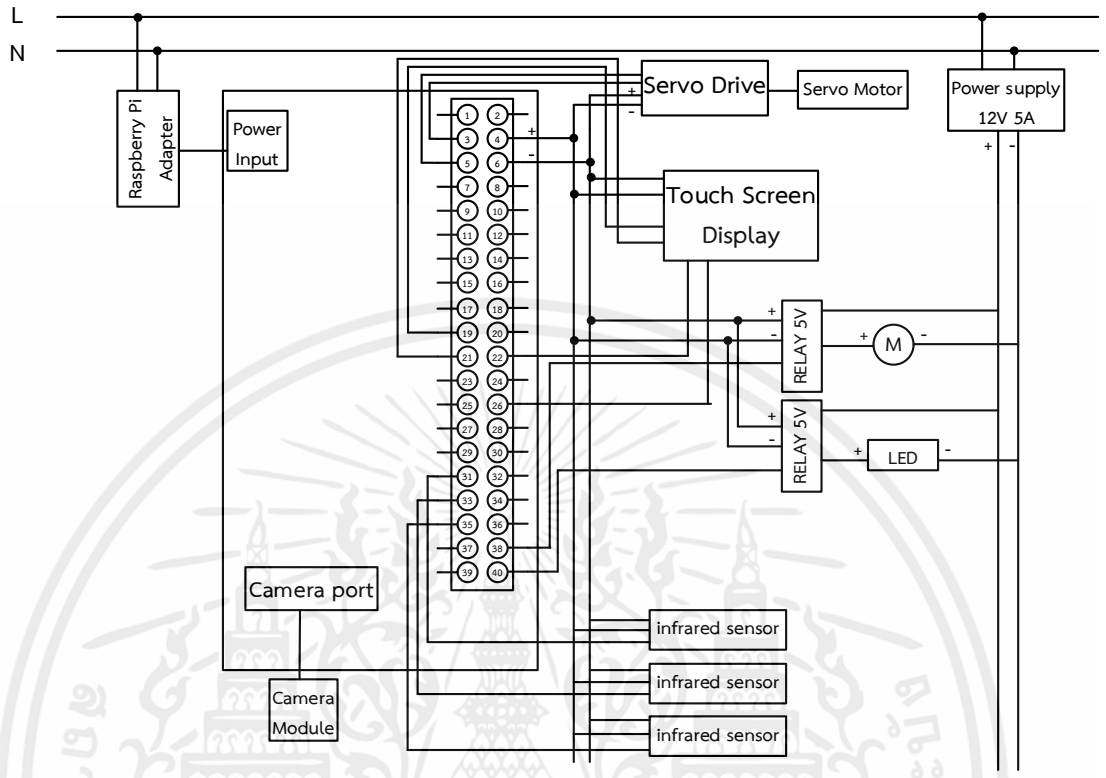
6. การหาค่าสี (Find Contour) คือ การหาพื้นที่สีขาวภายในข้อมูลภาพฐานสอง

7. การคำนวณพื้นที่สี (Calculate Contour Space) คือ การนับจำนวณพื้นที่สีขาวในภาพข้อมูลฐานสอง และเปรียบเทียบกับค่าที่กำหนดไว้

8. การแสดงผลของรูปร่าง (Output Detection Color) คือ เป็นกระบวนการแสดงรูปร่างของวัตถุที่ทำการตรวจจับเสร็จเรียบร้อยแล้ว

3.2 การออกแบบวงจร

3.2.1 การออกแบบวงจรใช้งานร่วมกับราสเบอรี่พาย



รูปที่ 3.4 การต่อพอร์ตจีพีไอโอของราสเบอรี่พายเข้ากับอุปกรณ์

จากรูปที่ 3.4 การต่อพอร์ตจีพีไอโอของราสเบอรี่พายเข้ากับอุปกรณ์ใช้งานโดยใช้งานรับอินพุตและส่งเอาท์พุตเพื่อควบคุมอุปกรณ์ต่างๆ

ซึ่งการเชื่อมต่ออินพุตจะมีดังนี้

-GPIO2(SDA) เชื่อมต่อกับบอร์ดไดร์ฟเซอร์โว

-GPIO6 เชื่อมต่อกับเซ็นเซอร์ตรวจจับวัตถุ

-GPIO13 เชื่อมต่อกับเซ็นเซอร์ตรวจจับวัตถุ

-GPIO19 เชื่อมต่อกับเซ็นเซอร์ตรวจจับวัตถุ

และการเชื่อมต่อเอาต์พุตจะมีดังนี้

-GPIO2(SDA) เชื่อมต่อกับบอร์ดไดร์ฟเซอร์โว

-GPIO3 เชื่อมต่อกับบอร์ดไดร์ฟเซอร์โว

-GPIO10(MOSI) เชื่อมต่อกับจอสัมผัส

-GPIO9(MISO) เชื่อมต่อกับจอสัมผัส

-GPIO11(SCLK) เชื่อมต่อกับจอสัมผัส

-GPIO25(IRQ) เชื่อมต่อกับจอสัมผัส

-GPIO7(CS) เชื่อมต่อกับจอสัมผัส

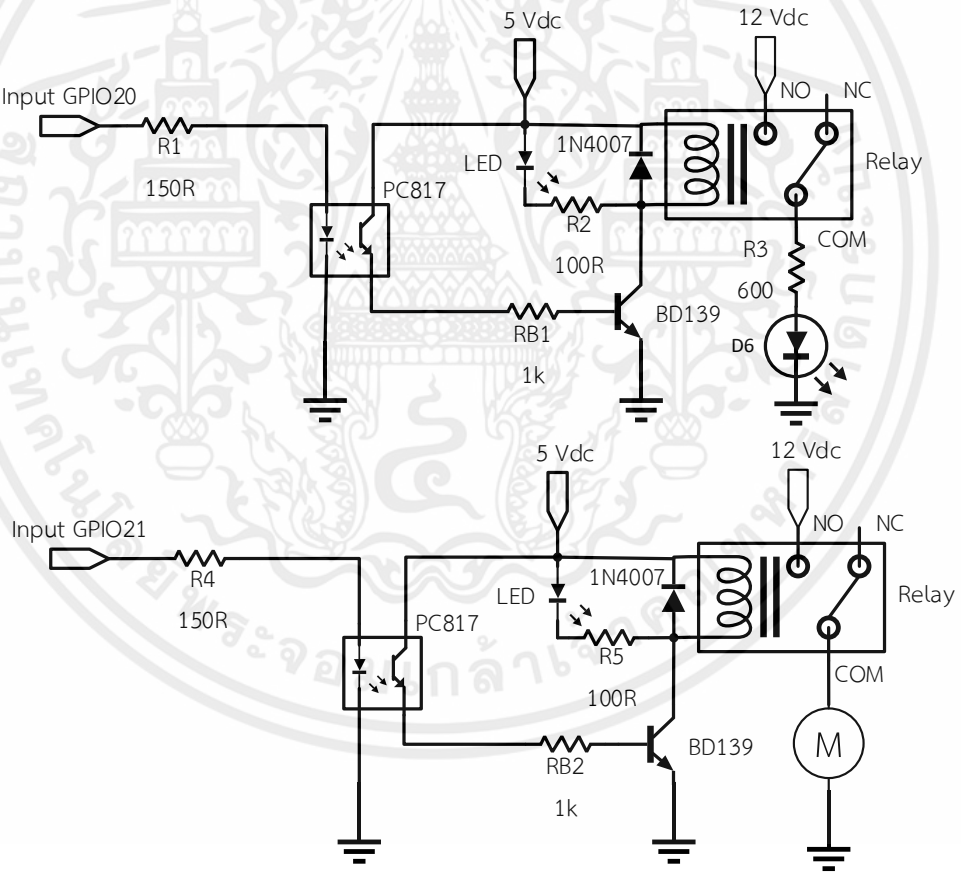
-GPIO20 เชื่อมต่อกับวงจรควบคุมแอลอีดี

-GPIO9 เชื่อมต่อกับวงจรควบคุมมอเตอร์

-GPIO5 เชื่อมต่อกับบอร์ดแปลงระดับแรงดันลอจิก

3.2.2 การออกแบบวงจรควบคุมมอเตอร์และแอลอีดี

การออกแบบวงจรสำหรับใช้ในควบคุมการทำงานของแอลอีดีและมอเตอร์ โดยหลักการการทำงานเมื่อออปโต้คัปเปลอร์ได้รับแรงดันอินพุต 3.3 โวลต์ จากพอร์ทจีพีไอโอของราสเบอร์รี่พาย ทรานซิสเตอร์ภายในออปโต้คัปเปลอร์จะนำกระแสไหลไปไบอัสทรานซิสเตอร์ BD139 เมื่อทรานซิสเตอร์ BD139 ทำงานจะเกิดกระแสไหลครบวงจรทำให้มีกระแสไหลผ่านขดลวดของรีเลย์เกิดพลังงานแม่เหล็กดูดเอาหน้าสัมผัสของรีเลย์ให้เปลี่ยนจากสถานะ NC เป็น NO แสดงดังรูปที่ 3.5



รูปที่ 3.5 การออกแบบวงจรควบคุมมอเตอร์และแอลอีดี

หาแรงดันตกคร่อมรีเลย์ได้จากสมการที่ 3.1

$$V_{cc} = V_{coil} + V_{CE} \quad (3.1)$$
$$V_{coil} = 5 - 0.5$$
$$V_{coil} = 4.5\ V$$

เมื่อ V_{cc} คือ แรงดันไฟฟ้า 5 โวลต์

V_{coil} คือ แรงดันตกคร่อมคอยล์ของรีเลย์

V_{CE} คือ แรงดันที่ตกคร่อมขาคอลเลคเตอร์และอิมิตเตอร์

หากระแส I_{coil} ที่สามารถทำให้รีเลย์ทางานได้ จากสมการที่ 3.2

$$I_{coil} = \frac{V_{coil}}{R_{coil}} \quad (3.2)$$
$$I_{coil} = \frac{4.5V}{70\Omega}$$
$$I_{coil} = 64.29\ mA$$

เมื่อ I_{coil} คือ กระแสที่รีเลย์สามารถทำงานได้

V_{coil} คือ แรงดันตกคร่อมคอยล์ของรีเลย์

R_{coil} คือ ความต้านทานรีเลย์

หาค่าความต้านทาน R2 เพื่อให้แอลอีดีมีกระแสไหลผ่าน 20 mA จากสมการที่ 3.3

$$R_2 = \frac{V_{cc}-V_{LED}}{I_{LED}} \quad (3.3)$$
$$R_2 = \frac{5-3}{20mA}$$
$$R_2 = 100\ \Omega$$

เมื่อ V_{cc} คือ แรงดันไฟฟ้า 5 โวลต์

V_{LED} คือ แรงดันตกคร่อมแอลอีดี

I_{LED} คือ กระแสไหลผ่านแอลอีดี

หาค่ากระแสรวมในวงจรที่ไหลผ่าน Q1 จากสมการที่ 3.4

$$I_{CS} = I_{coil} + I_{LED} \quad (3.4)$$

$$I_{CS} = 20\ mA + 64.29\ mA$$
$$I_{CS} = 84.29\ mA$$

เมื่อ I_{CS} คือ กระแสที่ไหลผ่านทรานซิสเตอร์ขณะอยู่ในโหมด Saturation

I_{coil} คือ กระแสที่ไหลผ่านคอยรีเลย์

I_{LED} คือ กระแสไหลผ่านแอลอีดี

หาค่ากระแส I_B ของทรานซิสเตอร์ จากสมการที่ 3.5

$$ODF = \frac{I_B}{I_{BS}}$$
$$I_{CS} = \beta \times I_{BS} \quad (3.5)$$
$$I_{BS} = \frac{I_C}{\beta}$$
$$I_{BS} = \frac{84.29mA}{40}$$
$$I_{BS} = 2.11\ mA$$
$$I_B = ODF \times I_{BS}$$
$$I_B = 2 \times 2.11\ mA$$
$$I_B = 4.22\ mA$$

เมื่อ β คือ อัตราการขยายของทรานซิสเตอร์

I_B คือ กระแสที่ขาเบสของทรานซิสเตอร์

I_{BS} คือ กระแสที่ขาเบสของทรานซิสเตอร์ขณะอยู่ในโหมด Saturation

ODF คือ ค่า Overdrive factor มีค่าระหว่าง 2 ถึง 5 ในที่นี้เลือกใช้ค่า 2

หาค่าความต้านทาน R_B ของทรานซิสเตอร์ จากสมการที่ 3.6

$$R_B = \frac{V_B - V_{BE}}{I_B} \quad (3.6)$$
$$R_B = \frac{5-1}{6.22\ mA}$$
$$R_B = 947.86\ \Omega$$
$$R_B \approx 1000\ \Omega$$

เมื่อ R_B คือ ค่าความต้านทานที่จำกัดกระแสของทรานซิสเตอร์

V_B คือ แรงดันที่ตกคร่อมตัวต้านทานขาเบสของทรานซิสเตอร์

V_{BE} คือ แรงดันที่ตกคร่อมขาเบสที่และขาอิมิตเตอร์ของทรานซิสเตอร์

หาค่าความต้านทาน R_1 ของออปโต้ เพื่อให้กระแสอินพุตไหลผ่าน 20 mA จากสมการที่ 3.7

$$R_1 = \frac{V_{GPIO}-V_F}{I_F} \quad (3.7)$$

$$R_1 = \frac{3.3-1.2}{20\ mA}$$

$$R_1 = 105\ \Omega$$

$$R_1 \approx 150\ \Omega$$

เมื่อ R_1 คือ ค่าความต้านทานที่จำกัดกระแสของออปโต้คัปเปลอร์

V_{GPIO} คือ แรงดันที่ได้รับจากราสเบอรี่พาย

V_F คือ แรงดันที่ตกคร่อมแอลอีดีภายในออปโต้คัปเปลอร์

3.3 การออกแบบโปรแกรม

3.3.1 การประมวลผลภาพ

3.3.1.1 การเลือกใช้ไลบรารี่โอเพ่นซีวี (OpecCV)

ในการออกแบบการประมวลผลภาพของเครื่องคัดแยกสีและขนาดของผล มะนาวได้เลือกใช้ไลบรารี่โอเพ่นซีวี ซึ่งเป็นไลบรารี่ฟังก์ชันการเขียนโปรแกรม โดยจะมีเป้าหมายไปที่การแสดงผลด้วยคอมพิวเตอร์แบบเรียลไทม์ ซึ่งจะมีคำสั่งย่อยให้จัดการเกี่ยวกับการประมวลผลภาพให้เลือกใช้ เช่น การเบลอภาพ การเปลี่ยนโหมดสี การตรวจจับสี และการคำนวณพิกเซล ฯลฯ

3.3.1.2 การเลือกใช้งานกล้อง

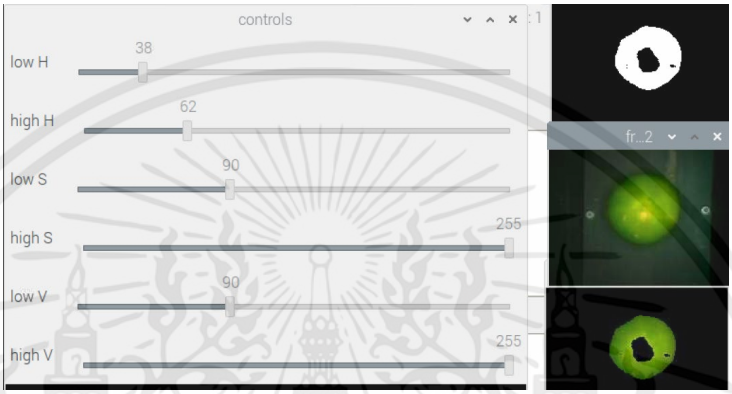
การเลือกใช้งานกล้องในการรับภาพเลือกใช้เป็นราสเบอรี่พายแคเมอร่าโมดูลเวอชั่น 2 สามารถต่อใช้งานร่วมกับบอร์ดราสเบอร์รี่พายได้โดยตรง มีความละเอียด 8 ล้านพิกเซล สามารถถ่ายวิดีโอระดับ HD ที่ความละเอียด 1080p, 720p และ 640x480 ด้วยอัตราแสดงผล 30 (1080p), 60 (720p และ 640x480) และ 90 (640x480) เฟรมต่อวินาที โดยราสเบอร์รี่พายแคเมอร่าโมดูลเวอชั่น 2 แสดงดังรูปที่ 3.6

รูปที่ 3.6 ราสเบอร์รี่พายแคเมอร่าโมดูลเวอชั่น 2

3.3.1.3 การกำหนดค่าเริ่มต้นในระบบสีแบบเอชเอสวี

1. การกำหนดค่าเริ่มต้นในระบบสีเอชเอสวีของมะนาวประเภทที่ 1

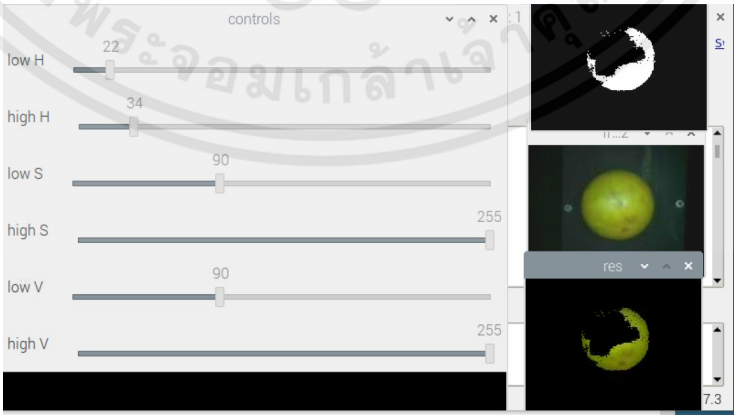
ผลมะนาวประเภทที่ 1 ผลมีสีเขียว การกำหนดค่าสีในระบบสีเอชเอสวีจะต้องกำหนดค่าสีให้ใกล้เคียงกับสีของมะนาวประเภทที่ 1 มากที่สุดเพื่อให้ได้ผลลัพธ์การประมวลผลภาพที่ถูกต้องสมบูรณ์ โดยกำหนดค่าสีต่ำสุดในระบบสีเอชเอสวีเท่ากับ (38, 90, 90) และกำหนดค่าสีสูงสุดในระบบ สีเอชเอสวีเท่ากับ (62, 255, 255) แสดงดังรูปที่ 3.7



รูปที่ 3.7 การกำหนดค่าเริ่มต้นในระบบสีเอชเอสวีของมะนาวประเภทที่ 1

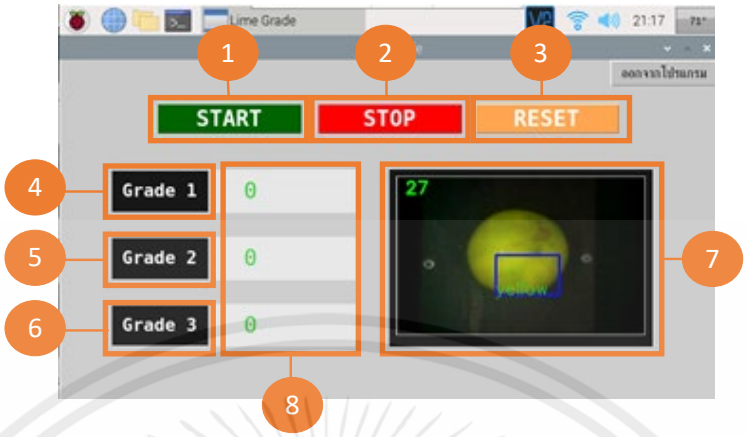
2. การกำหนดค่าเริ่มต้นในระบบสีเอชเอสวีของมะนาวประเภทที่ 2

ผลมะนาวประเภทที่ 2 ผลมีสีเหลือง การกำหนดค่าสีในระบบสีเอชเอสวีจะต้องกำหนดค่าสีให้ใกล้เคียงกับสีของมะนาวประเภทที่ 2 มากที่สุดเพื่อให้ได้ผลลัพธ์การประมวลผลภาพที่ถูกต้องสมบูรณ์ โดยกำหนดค่าสีต่ำสุดในระบบสีเอชเอสวีเท่ากับ (22, 90, 90) และกำหนดค่าสีสูงสุดในระบบ สีเอชเอสวีเท่ากับ (34, 255, 255) แสดงดังรูปที่ 3.8



รูปที่ 3.8 การกำหนดค่าเริ่มต้นในระบบสีเอชเอสวีของมะนาวประเภทที่ 2

3.3.2 การออกแบบหน้าโปรแกรมใช้งานจอสัมผัส



รูปที่ 3.9 หน้าโปรแกรมจียูไอ

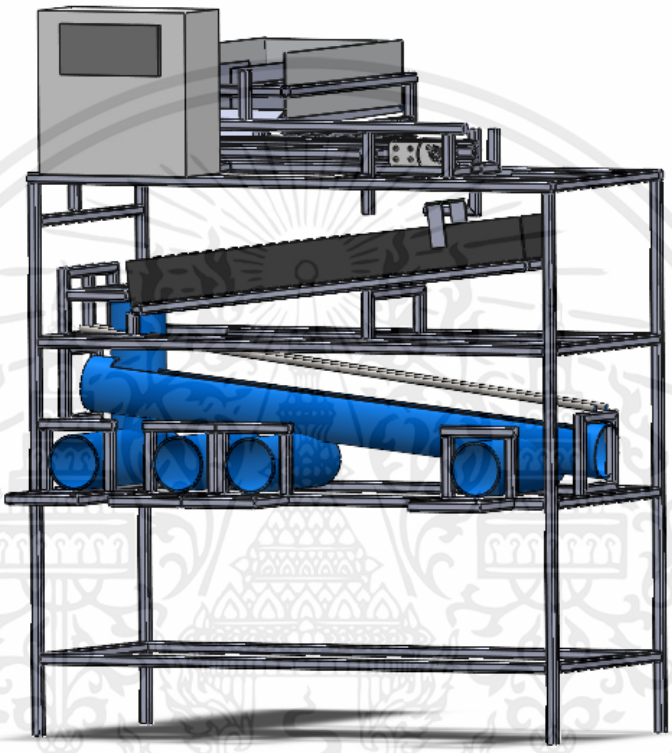
จากรูปที่ 3.9 ออกแบบโดยใช้ไลบรารี่ทีเคอินเตอร์ (Tkinter) ซึ่งเป็นไลบรารี่สำหรับออกแบบหน้าแสดงผลจียูไอ (Graphical User Interface : GUI) สามารถอธิบายการทำงานในแต่ส่วนของหน้าแสดงผลจียูไอของเครื่องคัดแยกสีและขนาดของผลมะนาว ได้ดังนี้

1. START คือ เมื่อสัมผัสที่ START เป็นการเริ่มการทำงานของเครื่องคัดแยกสีและขนาดผลมะนาว
2. STOP คือ เมื่อสัมผัสที่ STOP เป็นการหยุดการทำงานของเครื่องคัดแยกสีและขนาดผลมะนาว
3. RESET คือ เมื่อสัมผัสที่ RESET จะทำการล้างค่าผลรวมจำนวนนับผลมะนาวทุกขนาด
4. Grade1 คือ เมื่อสัมผัสที่ Grade1 จะทำการล้างค่าผลรวมจำนวนนับผลมะนาวขนาดที่ 1
5. Grade2 คือ เมื่อสัมผัสที่ Grade1 จะทำการล้างค่าผลรวมจำนวนนับผลมะนาวขนาดที่ 2
6. Grade3 คือ เมื่อสัมผัสที่ Grade1 จะทำการล้างค่าผลรวมจำนวนนับผลมะนาวขนาดที่ 3
7. หน้าแสดงผลการประมวลผลภาพมะนาวและแสดงผลลัพธ์การประมวลผลภาพระหว่างการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาว
8. แสดงผลรวมจำนวนนับผลมะนาวประเภทที่ 1 ในแต่ละขนาด

3.4 การออกแบบเครื่องคัดแยกสีและขนาดของผลมะนาว

การออกแบบโครงสร้างของเครื่องเครื่องคัดแยกสีและขนาดของผลมะนาวกึ่งอัตโนมัติ นั้นมีความสำคัญมากเท่ากับการออกแบบวงจรและการออกแบบโปรแกรม ซึ่งถ้าหากโครงสร้างไม่มีประสิทธิภาพจะส่งผลให้การทำงานมีความผิดพลาดเกิดขึ้น โดยแบ่งการออกแบบโครงสร้างเป็น 5 ส่วน ซึ่งมีการออกแบบโครงสร้างโดยรวมแสดงดังรูปที่ 3.10 และ 3.11

1. การออกแบบช่องใส่ผลมะนาว
2. การออกแบบโครงสร้างชุดสายพานลำเลียงลำเลียงผลมะนาว
3. การออกแบบโครงสร้างส่วนการคัดแยกสี
4. การออกแบบชุดโครงสร้างส่วนการประมวลผลภาพ
5. การออกแบบชุดคัดแยกขนาดของมะนาวผลสีเขียว

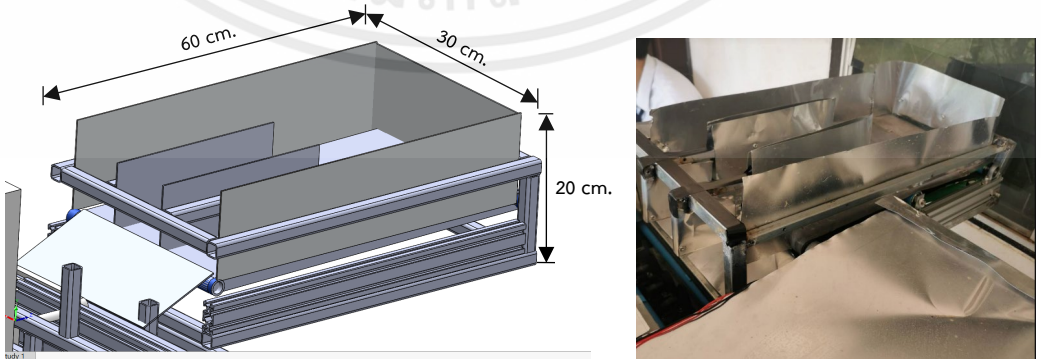


รูปที่ 3.10 การออกแบบโครงสร้างทั้งหมด

รูปที่ 3.11 โครงสร้างทั้งหมด

3.4.1 การออกแบบโครงสร้างช่องใส่ผลมะนาว

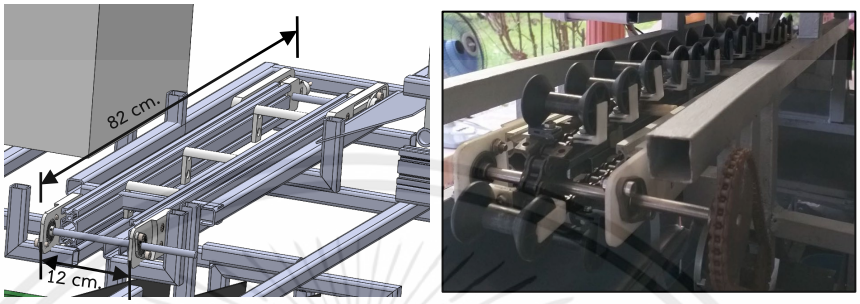
การออกแบบโครงสร้างของช่องใส่ผลมะนาว โดยโครงสร้างทำจากเหล็กกล่องเหล็กกัลวาไนซ์ ขนาด 20X20 มิลลิเมตร และปิดด้วยแผ่นอลูมิเนียม หนา 2 มิลลิเมตร โครงสร้างในส่วนนี้มีขนาดความกว้าง 30 เซนติเมตร ความยาว 60 เซนติเมตร และความสูง 20 เซนติเมตร แสดงดังรูปที่ 3.12



รูปที่ 3.12 การออกแบบโครงสร้างช่องใส่ผลมะนาว

3.4.2 การออกแบบโครงสร้างชุดลำเลียงผลมะนาว

การออกแบบโครงสร้างชุดลำเลียงผลมะนาว จะใช้โครงสร้างที่ทำจากอลูมิเนียมโปรไฟล์ขนาด 20X40 มิลลิเมตร โครงส่วนนี้มีขนาดความกว้าง 12 เซนติเมตร ความยาว 82 เซนติเมตร แสดงดังรูปที่ 3.13



รูปที่ 3.13 การออกแบบโครงสร้างชุดลำเลียงผลมะนาว

3.4.2.1 การคำนวณเฟืองขับชุดลำเลียง

ระบบขับเคลื่อนจะใช้มอเตอร์ไฟฟ้ากระแสตรง 12 โวลต์ รับน้ำหนักได้ 30 กิโลกรัมมีอัตราการหมุนที่ 110 รอบต่อ 1 นาที ในการออกแบบจะมีอัตราการทดรอบเท่ากับ 2 เพื่อลดความเร็วรอบของสายพานลำเลียงให้เหลือประมาณ 55 รอบ ต่อ 1 นาที แสดงดังรูปที่ 3.14 หาอัตราการทดรอบของมอเตอร์จากสมการที่ 3.8

$$I = \frac{Z_2}{Z_1} \tag{3.8}$$

$$I = \frac{38}{16}$$

$$I = 2$$

เมื่อ I คือ อัตราการทดรอบของมอเตอร์

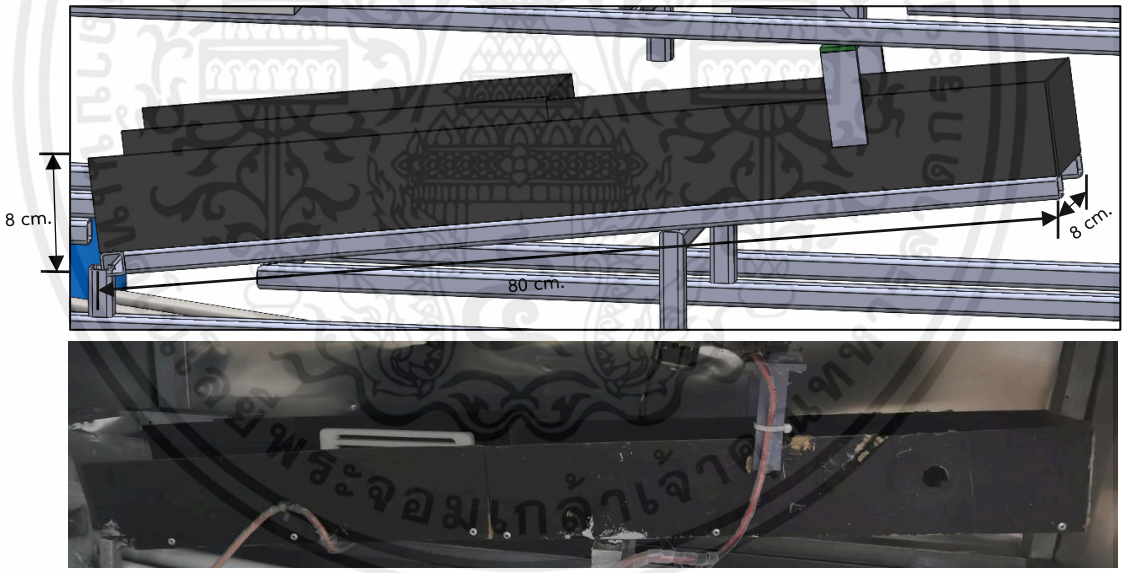
Z_1 คือ เส้นผ่านศูนย์กลางของเฟืองขับ (มิลลิเมตร)

Z_2 คือ เส้นผ่านศูนย์กลางของเฟืองตาม (มิลลิเมตร)

รูปที่ 3.14 เฟืองขับชุดลำเลียง

3.4.3 การออกแบบโครงสร้างส่วนการคัดแยกสีผลมะนาว

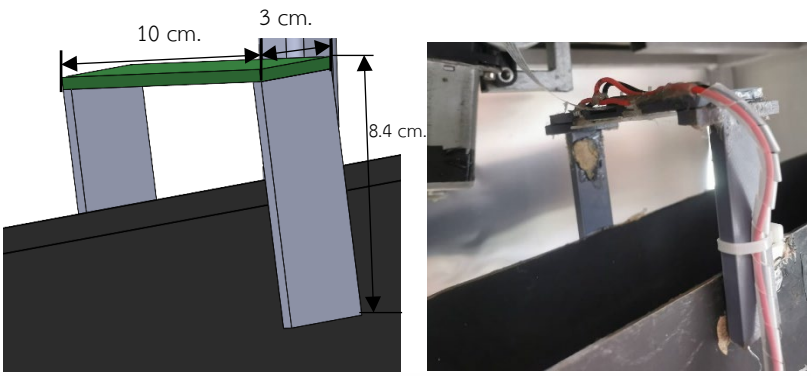
การออกแบบโครงสร้างชุดคัดแยกสีผลมะนาว เหล็กกล่องกัลวไนส์ ขนาด 20X20 มิลลิเมตร และปิดด้วยแผ่นอะคริลิค หนา 3 มิลลิเมตร โครงสร้างในส่วนนี้มีขนาดความกว้าง 8 เซนติเมตร ความยาว 80 เซนติเมตร และความสูง 8 เซนติเมตร แสดงดังรูปที่ 3.15



รูปที่ 3.15 การออกแบบโครงสร้างชุดคัดแยกสีผลมะนาว

3.4.4 การออกแบบโครงสร้างการประมวลผลภาพ

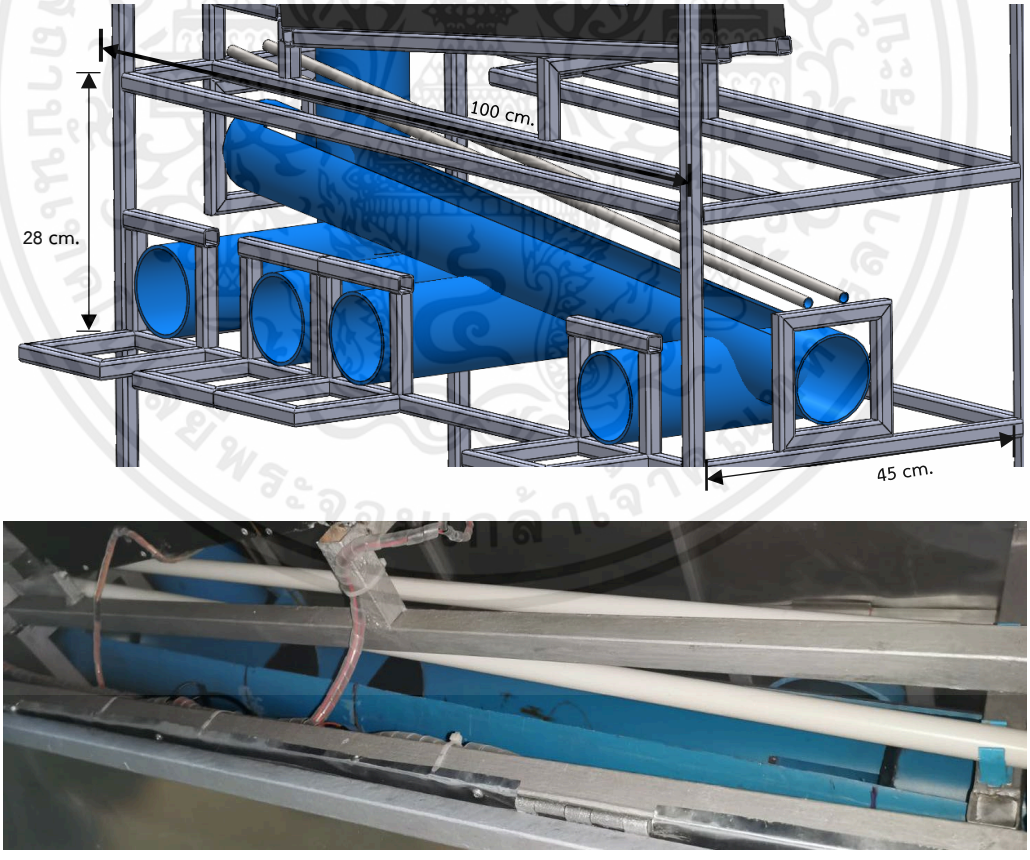
การออกแบบโครงสร้างการประมวลผลภาพทำจากท่อพลาสติกขึ้นรูปโดยใช้เครื่องพิมพ์ 3 มิติ โดยติดตั้งเข้ากับโครงสร้างการคัดแยกสี โครงสร้างในส่วนนี้มีขนาดความกว้าง 3 เซนติเมตร ความยาว 10 เซนติเมตร และความสูง 8.4 เซนติเมตร แสดงดังรูปที่ 3.16



รูปที่ 3.16 การออกแบบโครงสร้างการประมวลผลภาพ

3.4.5 การออกแบบโครงสร้างชุดคัดแยกขนาดผลมะนาว

การออกแบบโครงสร้างชุดคัดแยกขนาดผลมะนาวทำจากท่อ PVC ขนาด ½ นิ้ว วางขนานกันโดยระยะห่างระหว่างข้างบนและข้างล่างไม่เท่ากันเพื่อใช้สำหรับแบ่งขนาดผลมะนาว โครงสร้างในส่วนนี้มีขนาดความกว้าง 45 เซนติเมตร ความยาว 100 เซนติเมตร และความสูง 28 เซนติเมตร แสดงดังรูปที่ 3.17



รูปที่ 3.17 การออกแบบโครงสร้างชุดคัดแยกขนาดผลมะนาว

บทที่ 4
วิธีการและผลการทดลอง

ในบทนี้จะกล่าวถึง การทดลองและบันทึกผลการทดลองเครื่องคัดแยกขนาดและสีขทองผลมะนาวโดยแบ่งการทดลองออกเป็น 2 ส่วน ดังนี้ การทดลองการประมวลผลภาพ และการทดลองการคัดแยกสีของผลมะนาว

4.1 การทดลองการประมวลผลภาพ

เป็นการทดสอบการทำงานในส่วนของการประมวลผลภาพเพื่อคัดแยกสีของผลมะนาว ประเภทที่ 1 คือผลมะนาวสีเขียว และประเภทที่ 2 คือผลมะนาวสีเหลือง

4.1.1 อุปกรณ์ที่ใช้ในการทดลอง

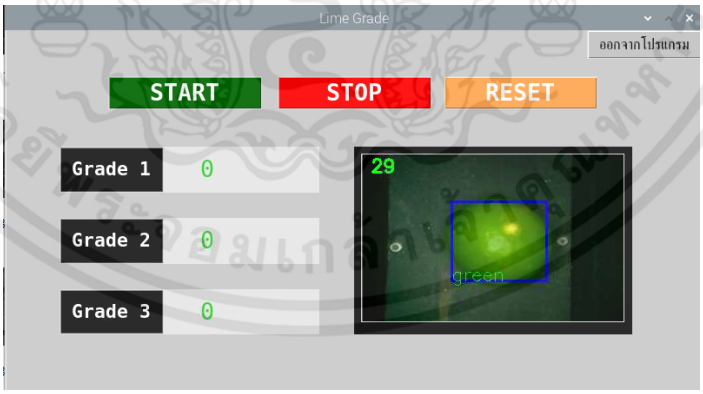
1) เครื่องคัดแยกสีของผลมะนาว

2) ผลมะนาวประเภทที่ 1 จำนวน 10 ผล

3) ผลมะนาวประเภทที่ 2 จำนวน 10 ผล

4.1.2 ขั้นตอนการทดลอง

1) นำผลมะนาวประเภทที่ 1 จำนวน 10 ผล ทำการทดลองประมวลผลภาพผลมะนาวทั้ง 10 ผล และบันทึกผลการทดลองลงในตารางที่ 4.1 โดยหากเป็นผลมะนาวประเภทที่ 1 จะสามารถประมวลผลภาพได้ดังรูปที่ 4.1

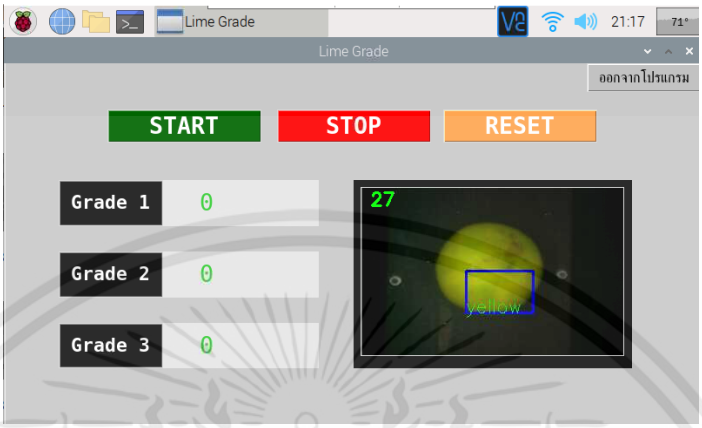


รูปที่ 4.1 การทดลองการประมวลผลภาพผลมะนาวประเภทที่ 1

2) นำผลการทดลองจากตารางที่ 4.1 ไปหาเปอร์เซ็นต์ค่าความผิดพลาด โดยคำนวณจากสมการที่ 4.1

$$\text{เปอร์เซ็นต์ความผิดพลาด} = \left(\frac{\text{ผลรวมของผลมะนาวที่ทดลอง} - \text{ผลรวมของผลมะนาวที่ทดลองผ่าน}}{\text{ผลรวมของผลมะนาวที่ทดลอง}}\right) \times 100 \quad (4.1)$$

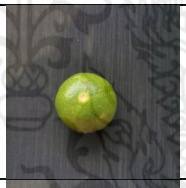
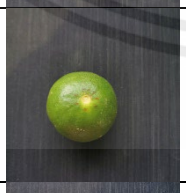
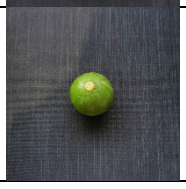
3) นำผลมะนาวประเภทที่ 2 จำนวน 10 ผล ทำการทดลองประมวลผลภาพผลมะนาวทั้ง 10 ผล และบันทึกผลการทดลองลงในตารางที่ 4.1 โดยหากเป็นผลมะนาวประเภทที่ 2 จะสามารถประมวลผลภาพได้ดังรูปที่ 4.2



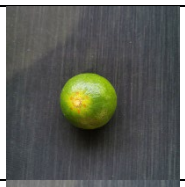
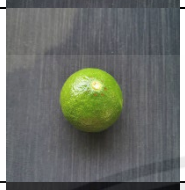
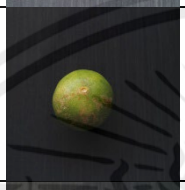
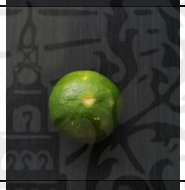
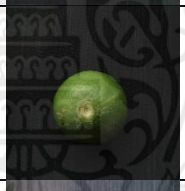
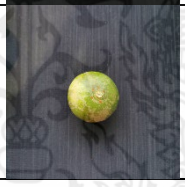
รูปที่ 4.2 การทดลองการประมวลผลภาพผลมะนาวประเภทที่ 2

4) นำผลการทดลองจากตารางที่ 4.2 ไปหาเปอร์เซ็นต์ค่าความผิดพลาด โดยคำนวณจากสมการที่ 4.1

ตารางที่ 4.1 ผลการทดลองการประมวลผลภาพมะนาวประเภทที่ 1

ครั้งที่	ผลมะนาว	การประมวลผลภาพ
1		✓
2		✓
3		✓
4		✓

ตารางที่ 4.1 ผลการทดลองการประมวลผลภาพมะนาวประเภทที่ 1 (ต่อ)

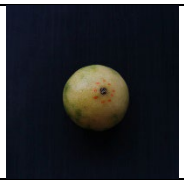
ครั้งที่	ผลมะนาว	การประมวลผลภาพ
5		✓
6		✓
7		✓
8		✓
9		✓
10		✓

หมายเหตุ : ✓ หมายถึง สีจากการประมวลผลภาพตรงกับผลมะนาวที่นำมาทดลอง

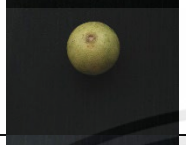
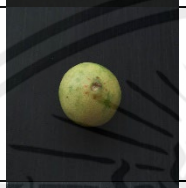
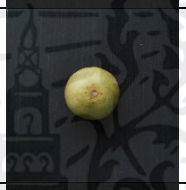
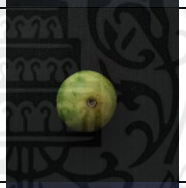
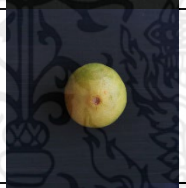
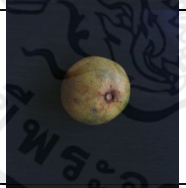
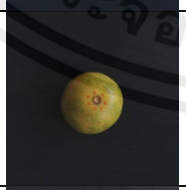
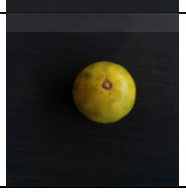
✗ หมายถึง สีจากการประมวลผลภาพไม่ตรงกับผลมะนาวที่นำมาทดลอง

ผลการทดลองจากตารางที่ 4.1 ผลลัพธ์การประมวลผลภาพของผลมะนาวประเภทที่ 1 ทำการทดลองทั้งหมด 10 ผล มีค่าความผิดพลาดเฉลี่ยเท่ากับ 0 เปอร์เซ็นต์

ตารางที่ 4.2 ผลการทดลองการประมวลผลภาพมะนาวประเภทที่ 2

ครั้งที่	ผลมะนาว	การประมวลผลภาพ
1		✓

ตารางที่ 4.2 ผลการทดลองการประมวลผลภาพมะนาวประเภทที่ 2 (ต่อ)

ครั้งที่	ผลมะนาว	การประมวลผลภาพ
2		✓
3		✓
4		✓
5		✓
6		✓
7		✓
8		✓
9		✓
10		✓

หมายเหตุ : ✓ หมายถึง สีจากการประมวลผลภาพตรงกับผลมะนาวที่นำมาทดลอง

✗ หมายถึง สีจากการประมวลผลภาพไม่ตรงกับผลมะนาวที่นำมาทดลอง

ผลการทดลองจากตารางที่ 4.2 ผลลัพธ์การประมวลผลภาพของผลมะนาวประเภทที่ 1 ทำการทดลองทั้งหมด 10 ผล มีค่าความผิดพลาดเฉลี่ยเท่ากับ 0 เปอร์เซ็นต์

4.2 การทดลองการคัดแยกขนาดของผลมะนาว

เป็นการทดลองการทำงานในส่วนของการคัดแยกขนาดผลมะนาวโดยแบ่ง เป็น 3 ขนาด คือ มะนาวเบอร์ 1 2 และ 3

4.2.1 อุปกรณ์ที่ใช้ในการทดลอง

1) เครื่องคัดแยกสีและขนาดของผลมะนาว

2) มะนาวคละขนาด 120 ผล

3) ถุงกระสอบสำหรับใส่มะนาว

4.2.2 ขั้นตอนการทดลอง

1) นำผลมะนาวคละขนาด 90 ผล เทลงในช่องใส่ผลมะนาวเพื่อทำการคัดแยกขนาดผลมะนาว โดยแสดงดังรูปที่ 4.3

รูปที่ 4.3 ผลมะนาวคละขนาด

2) บันทึกจำนวนผลมะนาวที่ผ่านการคัดแยกแต่ละขนาด

3) บันทึกจำนวนผลมะนาวขนาดที่ 1 2 และ 3

4) นำมะนาวขนาดที่ 1 ทำการทดลองคัดแยกอีกครั้งเพื่อหาความผิดพลาดของชุดคัดแยกขนาด แสดงดังรูปที่ 4.4

รูปที่ 4.4 ทดลองแยกมะนาวขนาดที่ 1

5) บันทึกผลการทดลองลงในตารางที่ 4.3

6) นำผลการทดลองจากตารางที่ 4.3 ไปหาเปอร์เซ็นต์ค่าความผิดพลาด โดยคำนวณ

จากสูตร $\left(\frac{\text{จำนวนมะนาวขนาดที่ X ทั้งหมด} - \text{จำนวนผลมะนาวในถุงกระสอบขนาดที่ X}}{\text{จำนวนมะนาวขนาดที่ X ทั้งหมด}}\right) \times 100$ (4.2)

ตารางที่ 4.3 ผลการทดลองคัดแยกขนาดของผลมะนาวขนาดที่ 1

ครั้งที่	มะนาวผลสีเขียว ขนาดที่ 1 (ผล)	มะนาวที่อยู่ในถุง กระสอบขนาดที่ 1 (ผล)	ค่าความผิดพลาด (ผล)	ค่าความผิดพลาด (เปอร์เซ็นต์)
1	30	27	3	10
2	30	26	4	13.3
3	30	30	0	0
4	30	29	1	3.33
5	30	27	3	10

ผลการทดลองจากตารางที่ 4.3 ทำการทดลองทั้งหมด 5 ครั้ง มีค่าความผิดพลาดสูงสุดอยู่ที่การทดลองครั้งที่ 2 คือ 13.3 เปอร์เซ็นต์ และมีความผิดพลาดเฉลี่ยอยู่ที่ 7.33 เปอร์เซ็นต์

7) นำมะนาวขนาดที่ 2 ทำการทดลองคัดแยกอีกครั้งเพื่อหาความผิดพลาดของชุดคัดแยกขนาด แสดงดังรูปที่ 4.5

รูปที่ 4.5 ทดลองแยกมะนาวขนาดที่ 2

8) บันทึกผลการทดลองลงในตารางที่ 4.4

9) นำผลการทดลองจากตารางที่ 4.4 ไปหาเปอร์เซ็นต์ค่าความผิดพลาด โดยคำนวณจากสมการที่ 4.2

ตารางที่ 4.4 ผลการทดลองคัดแยกขนาดของผลมะนาวขนาดที่ 2

ครั้งที่	มะนาวผลสีเขียว ขนาดที่ 2 (ผล)	มะนาวที่อยู่ในถุง กระสอบขนาดที่ 2 (ผล)	ค่าความผิดพลาด (ผล)	ค่าความผิดพลาด (เปอร์เซ็นต์)
1	30	26	4	13.3
2	30	30	0	0
3	30	27	3	10
4	30	28	2	6.6
5	30	28	2	6.6

ผลการทดลองจากตารางที่ 4.4 ทำการทดลองทั้งหมด 5 ครั้ง มีค่าความผิดพลาดสูงสุดอยู่ที่การทดลองครั้งที่ 1 คือ 13.3 เปอร์เซ็นต์ และมีความผิดพลาดเฉลี่ยอยู่ที่ 7.3 เปอร์เซ็นต์

10) นำมะนาวขนาดที่ 3 ทำการทดลองคัดแยกอีกครั้งเพื่อหาความผิดพลาดของชุดคัดแยกขนาด แสดงดังรูปที่ 4.6

รูปที่ 4.6 ทดลองแยกมะนาวขนาดที่ 3

11) บันทึกผลการทดลองลงในตารางที่ 4.5

12) นำผลการทดลองจากตารางที่ 4.5 ไปหาเปอร์เซ็นต์ค่าความผิดพลาด โดยคำนวณจากสมการที่ 4.2

ตารางที่ 4.5 ผลการทดลองคัดแยกขนาดของผลมะนาวขนาดที่ 3

ครั้งที่	มะนาวผลสีเขียวขนาดที่ 3 (ผล)	มะนาวที่อยู่ในถุงกระสอบขนาดที่ 3 (ผล)	ค่าความผิดพลาด (ผล)	ค่าความผิดพลาด (เปอร์เซ็นต์)
1	30	29	1	3.33
2	30	30	0	0
3	30	28	2	6.66
4	30	30	0	0
5	30	29	1	3.33

ผลการทดลองจากตารางที่ 4.5 ทำการทดลองทั้งหมด 5 ครั้ง มีค่าความผิดพลาดสูงสุดอยู่ที่การทดลองครั้งที่ 3 คือ 6.6 เปอร์เซ็นต์ และมีความผิดพลาดเฉลี่ยอยู่ที่ 2.66 เปอร์เซ็นต์

4.3 การทดลองการทำงานของเครื่องคัดแยกขนาดและสีของผลมะนาว

เป็นการทดลองการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาวแบบครบทุกส่วนเพื่อบันทึกระยะเวลาในการคัดแยกขนาดและสีของผลมะนาวจำนวน 2 กิโลกรัมต่อการทำงาน 1 ครั้ง

4.3.1 อุปกรณ์ที่ใช้ในการทดลอง

1) เครื่องคัดแยกสีและขนาดของผลมะนาว

2) ผลมะนาวแบบคละสีและคละขนาด 2 กิโลกรัม (ประมาณ 40 ผล)

3) นาฬิกาจับเวลา

4) เครื่องชั่งน้ำหนัก

5) ตะกร้าสำหรับใส่ผลมะนาว

4.3.2 ขั้นตอนการทดลอง

1) เปิดการทำงานเครื่องคัดแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติ แสดงดังรูปที่ 4.7

รูปที่ 4.7 เปิดการใช้งานเครื่องคัดแยกสีและขนาดผลมะนาว

2) นำผลมะนาวคละสีและคละขนาด 40 ผล เทลงในช่องใส่ผลมะนาวโดยแสดงดังรูปที่ 4.8

รูปที่ 4.8 นำผลมะนาวคละสีและขนาด 40 ผลเทลงในช่องใส่ผลมะนาว

2) ทำการทดลองการคัดแยกสีและขนาดของผลมะนาว และทำการจับเวลาการทำงานของเครื่องดังรูปที่ 4.9

รูปที่ 4.9 คัดแยกสีและขนาดผลมะนาวพร้อมทำการจับเวลา

3) บันทึกเวลาที่ใช้ในการคัดแยกขนาดและสีของผลมะนาวลงในตารางที่ 4.6

4) นำผลการทดลองจากตารางที่ 4.6 ไปหาค่าเฉลี่ยเวลาที่ใช้ในการคัดแยกขนาดและสีของมะนาวโดยคำนวณ

จากสูตร $\left(\frac{\text{ผลรวมของระยะเวลาที่ใช้ในการทดลอง } \mathbf{5} \text{ ครั้ง}}{\text{จำนวนครั้งการทดลอง}}\right)$

ตารางที่ 4.6 ผลการทดลองการทำงานของเครื่องคัดแยกขนาดและสีของผลมะนาว

ครั้งที่	เวลา (วินาที)
1	53
2	54
3	50
4	55
5	54

ผลการทดลองจากตารางที่ 4.6 ทำการทดลอง 5 ครั้ง ระยะเวลาเฉลี่ยที่ใช้ในการคัดแยกขนาดและสีของผลมะนาว 40 ผล คือ 53.2 วินาที

บทที่ 5
บทสรุปและข้อเสนอแนะ

ในบทนี้จะกล่าวถึง บทสรุปในการทดลองจากบทที่ 4 ในหัวข้อ การทดลองการประมวลผลภาพ การทดลองการคัดแยกสีของผลมะนาว ปัญหาและอุปสรรค และแนวทางแก้ไข

5.1 สรุปผลการทดลอง

5.1.1 การทดลองส่วนของการประมวลผลภาพ

จากการทดลองเครื่องคัดแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติในส่วนการประมวลผลภาพ กล้องสามารถตรวจจับมะนาวผลสีเขียวและมะนาวผลสีเหลืองได้

5.1.2 การทดลองการคัดแยกขนาดของผลมะนาว

จากการทดลองเครื่องคัดแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติในส่วนการคัดแยกขนาดผลมะนาว ละขนาดที่ 1, 2 และ3 ขนาดละ 30 ผล โดยทำการทดลองซ้ำ 5 ครั้ง พบว่าการคัดแยกขนาดผลมะนาวขนาดที่ 1 มีความผิดพลาดเฉลี่ยอยู่ที่ 7.33 เปอร์เซ็นต์ ขนาดที่ 2 มีความผิดพลาดเฉลี่ยอยู่ที่ 7.3 เปอร์เซ็นต์ และขนาดที่ 3 มีความผิดพลาดเฉลี่ยอยู่ที่ 2.66 เปอร์เซ็นต์

5.1.3 การทดลองการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาว

จากการทดลองการทำงานโดยรวมของเครื่องคัดแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติใช้เวลาเฉลี่ยในการคัดแยกสีและขนาดของผลมะนาว 40 ผล คือ 53.2 วินาที

5.2 ปัญหาและอุปสรรค

จากการทดลองมีความผิดพลาดเกิดขึ้นที่บริเวณสายพานลำเลียงมีการซ้อนกันของผลมะนาว ทำให้ผลมะนาวที่ซ้อนไม่ผ่านการประมวลผลภาพ

จากการทดลองคัดแยกสีผลมะนาวแบบคละประเภทปัญหาที่พบคือหากมีผลมะนาวลำเลียงบนชุดลำเลียงต่อเนื่องมากเกินไป ส่งผลทำให้การประมวลผลภาพเกิดการขัดข้อง เนื่องจากตรวจจับผลสีเหลืองและผลสีเขียวพร้อมกัน จึงทำให้ผลมะนาวสีเขียวถูกคัดไปยังภาชนะบรรจุของผลมะนาวสีเหลืองด้วย

จากการทดลองควรปรับแก้ความสูงของระยะสายพานลำเลียงกับชุดประมวลผลภาพให้มีความสูงลดลงเนื่องจากผลมะนาวตกกระทบกับชุดประมวลผลภาพค่อนข้างแรง

5.3 แนวทางแก้ไข

เพิ่มชุดแผ่นยางกั้นบริเวณสายพานเพื่อป้องกันการซ้อนทับกันของลูกมะนาวบนสายพานลำเลียง

เพิ่มชุดสายพานลำเลียงไปยังช่องบรรจุผลมะนาวอีก 1 ชุด เพื่อไม่ให้เกิดการติดขัดที่บริเวณช่องใส่ผลมะนาว

ลดระยะห่างระหว่างสายพานลำเลียงและชุดคัดแยกสีให้ลดลงเพื่อลดแรงกระแทกของผลมะนาวกับชุดคัดแยก

เอกสารอ้างอิง

[1] บรรพตี, **คู่มือปลูกมะนาว ฉบับสมบูรณ์**, กรุงเทพ, สนพ. เอ็มไอเอส, 2558.

[2] ราชกิจจานุเบกษาฉบับประกาศและงานทั่วไป, มะนาว 134, กรุงเทพ, สานักงานมาตรฐานสินค้าเกษตรและอาหารแห่งชาติกระทรวงเกษตรและสหกรณ์, 2560

[3] นพ มหิษานนท์, **การติดตั้งและใช้งาน Raspberry Pi**, กรุงเทพ, สนพ. คอร์ฟังก์ชั่น, 2560.

[4] ผศ.ดร. สุชาติ คุ้มมะณี, **Basic Python Coding เรียนง่ายเป็นเร็ว**, กรุงเทพ, สนพ. ไอดีซีพรีเมียร์, 2562.

[5] บุญธรรม ภัทราจารุกุล, **การประมวลผลภาพดิจิตอลเบื้องต้น**, กรุงเทพ, สนพ. ซีเอ็ดยูเคชั่น, 2556.

[6] “โมดูลกล้อง”, URL: https://www.arduitronics.com/product/1169/raspberry-pi-camera-v2, เข้าถึงครั้งสุดท้าย วันที่ 18 พฤศจิกายน 2563

[7] “การประมวลผลภาพ”, URL http://krusuwatcomputer.blogspot.com/p/20_5.html, เข้าถึงครั้งสุดท้ายวันที่ 23 พฤศจิกายน 2563

[8] “ระบบสี”, URL: https://sites.google.com/site/pongnuntipisek/3-rabb-si, เข้าถึงครั้งสุดท้ายวันที่ 23 พฤศจิกายน 2563

[9] “Servo Driver”, URL: https://www.th.cytron.io/p-servo-driver-hat-for-raspberry-pi-16 channel-12-bit-i2c?search, เข้าถึงครั้งสุดท้าย วันที่ 18 พฤศจิกายน 2563

[10] “เซอร์โวมอเตอร์”, URL: https://www.inventor.in.th/home/เซอร์โวมอเตอร์, เข้าถึงครั้งสุดท้าย วันที่ 18 พฤศจิกายน 2563

[11] “มอเตอร์เกียร์”, URL: https://sancarlosantioquia.com/มอเตอร์เกียร์, เข้าถึงครั้งสุดท้ายวันที่ 18 พฤศจิกายน 2563

[12] “เซนเซอร์จับวัตถุ”, URL: http://www.ett.co.th/productSensor/E18-D80NK/E18-D80NK.html, เข้าถึงครั้งสุดท้าย วันที่ 18 พฤศจิกายน 2563

ภาคผนวก ก

โปรแกรมการทำงาน

 # Import ไลบรารีทั้งหมดที่ใช้งาน

```
import threading              # Import ไลบรารี threading
from tkinter import *         # Import * จากไลบรารี tkinter เพื่อออกแบบหน้าต่างจียูไอ
from tkinter import messagebox # Import messagebox จากไลบรารี tkinter
import time                   # Import ไลบรารี time เพื่อใช้สำหรับหน่วงเวลา
import RPi.GPIO as gpio       # Import ไลบรารี GPIO และตั้งชื่อใหม่เป็น gpio
import cv2                    # Import ไลบรารี OpenCV
import numpy as np            # Import ไลบรารี numpy(โมดูลเสริมเกี่ยวกับคณิตศาสตร์) และตั้งชื่อใหม่ว่า np
from PIL import Image, ImageTk  # Import Image และ ImageTk จากไลบรารี PIL เพื่อแสดงหน้าวิดีโอการประมวลผลภาพบนหน้าแสดงผลจียูไอ
import Adafruit_PCA9685     # Import ไลบรารี Adafruit_PCA9685 เพื่อใช้งานบอร์ดขับเซอร์โวมอเตอร์
```

 # กำหนดการใช้งานขา GPIO

```
PIN_3V3 = 5          # ขา GPIO 5 เชื่อมต่อกับวงจรแปลงระดับแรงดันสัญญาณเพื่อใช้เป็นแรงดันอ้างอิง 3.3 โวลต์
PIN_GRADE1 = 13  # ขา GPIO 13 เชื่อมต่อกับอินฟาเรดเซ็นเซอร์เพื่อนับจำนวนมะนาวขนาดที่ 1
PIN_GRADE2 = 19  # ขา GPIO 19 เชื่อมต่อกับอินฟาเรดเซ็นเซอร์เพื่อนับจำนวนมะนาวขนาดที่ 2
PIN_GRADE3 = 26  # ขา GPIO 26 เชื่อมต่อกับอินฟาเรดเซ็นเซอร์เพื่อนับจำนวนมะนาวขนาดที่ 3
PIN_RELAY_MOTOR = 20  # ขา GPIO 20 เชื่อมต่อกับรีเลย์ที่ควบคุมการทำงานของมอเตอร์
PIN_RELAY_LED = 21  # ขา GPIO 26 เชื่อมต่อกับรีเลย์ที่ควบคุมการทำงานของแอลอีดี
```

 # กำหนดโหมดใช้งานขา GPIO

```
gpio.setmode(gpio.BCM)       # ตั้งค่าการอ้างอิงพอร์ตแบบ BCM
gpio.setwarnings(False)      # ปิดการแจ้งเตือน Warning
gpio.setup(PIN_GRADE1, gpio.IN, pull_up_down=gpio.PUD_UP)       # ตั้งค่าให้พอร์ต PIN_GRADE1 เป็น Input แบบ pull up
gpio.setup(PIN_GRADE2, gpio.IN, pull_up_down=gpio.PUD_UP)       # ตั้งค่าให้พอร์ต PIN_GRADE2 เป็น Input แบบ pull up
gpio.setup(PIN_GRADE3, gpio.IN, pull_up_down=gpio.PUD_UP)       # ตั้งค่าให้พอร์ต PIN_GRADE3 เป็น Input แบบ pull up
gpio.setup(PIN_3V3, gpio.OUT)     # ตั้งค่าให้พอร์ต PIN_3V3 เป็น Output
gpio.setup(PIN_RELAY_MOTOR, gpio.OUT# ตั้งค่าให้พอร์ต PIN_RELAY_MOTOR เป็นOutput
gpio.setup(PIN_RELAY_LED, gpio.OUT)    # ตั้งค่าให้พอร์ต PIN_RELAY_LED เป็น Output
```

```
gpio.add_event_detect(PIN_GRADE1, gpio.RISING)    # ตั้งค่าให้พอร์ต PIN_GRADE1 รับสัญญาณขอบขาขึ้น
gpio.add_event_detect(PIN_GRADE2, gpio.RISING)    # ตั้งค่าให้พอร์ต PIN_GRADE2 รับสัญญาณขอบขาขึ้น
gpio.add_event_detect(PIN_GRADE3, gpio.RISING)    # ตั้งค่าให้พอร์ต PIN_GRADE3 รับสัญญาณขอบขาขึ้น
```

 # ตัวแปรชนิด Global เป็นตัวแปรกลางสำหรับใช้ในโปรแกรมทั้งหมด

```
Stop1, Stop2 = 0, 0
Lime_cnt1, Lime_cnt2, Lime_cnt3 = 0, 0, 0
tv_Lime1 = IntVar()          # ตัวแปร tv_Lime1 ใช้สำหรับแสดงจำนวนผลมะนาวขนาดที่ 1 บนหน้าแสดงผลจียูไอ
tv_Lime2 = IntVar()          # ตัวแปร tv_Lime2 ใช้สำหรับแสดงจำนวนผลมะนาวขนาดที่ 2 บนหน้าแสดงผลจียูไอ
tv_Lime3 = IntVar()          # ตัวแปร tv_Lime3 ใช้สำหรับแสดงจำนวนผลมะนาวขนาดที่ 3 บนหน้าแสดงผลจียูไอ
```

 # ฟังก์ชั่นการทำงานของมอเตอร์

```
def motor():
    print('Motor ON..')
    global Stop1, PIN_RELAY_MOTOR, PIN_RELAY_LED  # เรียกใช้ตัวแปรภายนอกฟังก์ชัน
    while True:      # วนรอบการทำงาน
        gpio.output(PIN_RELAY_LED, 1)          # พอร์ต PIN_RELAY_LED มีเอาต์พุตเป็น 1
        gpio.output(PIN_RELAY_MOTOR, 1)        # พอร์ต PIN_RELAY_MOTOR มีเอาต์พุตเป็น 1
        time.sleep(0.2)          # หน่วงเวลา 0.2 วินาที
        if Stop1 == 1:       # ถ้าตัวแปร Stop 1 เป็นจริงให้หยุดการทำงานของมอเตอร์และแอลอีดี
            print("Stop Motor..")
            gpio.output(PIN_RELAY_MOTOR, 0)
            gpio.output(PIN_RELAY_LED, 0)
            time.sleep(0.2)       # หน่วงเวลา 0.2 วินาที
            Stop1 = 0             # ตัวแปร Stop1 มีค่าเท่ากับ 0
            return Stop1          # ส่งค่า Stop1 มีค่าเท่ากับ 0 กลับไปยังตัวแปรนอกฟังก์ชั่น
```

```
# ฟังก์ชั่นการทำงานของเซ็นเซอร์ตรวจจับวัตถุเพื่อนับจำนวนผลมะนาว
def cnt_lime():
    print('Counter ON..')
    global Stop1, PIN_GRADE1, PIN_GRADE2, PIN_GRADE3, Lime_cnt1, Lime_cnt2,
Lime_cnt3           # เรียกใช้ตัวแปรภายนอกฟังก์ชัน
    while True:       # วนรอบการทำงาน
        if gpio.event_detected(PIN_GRADE1): # ถ้าพอร์ต PIN_GRADE1 ได้รับสัญญาณขอบขา
ขึ้น
            Lime_cnt1 += 1      # ตัวแปร Lime_cnt1 มีค่าเพิ่มขึ้นครั้งละ 1
            tv_Lime1.set(Lime_cnt1)       # ตั้งค่าให้ตัวแปร tv_Lime1 มีค่าเท่ากับ Lime_cnt1
            time.sleep(0.2)       # หน่วงเวลา 0.2 วินาที
        if gpio.event_detected(PIN_GRADE2): # ถ้าพอร์ต PIN_GRADE2 ได้รับสัญญาณขอบขา
ขึ้น
            Lime_cnt2 += 1      # ตัวแปร Lime_cnt2 มีค่าเพิ่มขึ้นครั้งละ 1
            tv_Lime2.set(Lime_cnt2)       # ตั้งค่าให้ตัวแปร tv_Lime2 มีค่าเท่ากับ Lime_cnt2
            time.sleep(0.2)       # หน่วงเวลา 0.2 วินาที
        if gpio.event_detected(PIN_GRADE3): # ถ้าพอร์ต PIN_GRADE3 ได้รับสัญญาณขอบขา
ขึ้น
            Lime_cnt3 += 1      # ตัวแปร Lime_cnt3 มีค่าเพิ่มขึ้นครั้งละ 1
            tv_Lime3.set(Lime_cnt3)       # ตั้งค่าให้ตัวแปร tv_Lime3 มีค่าเท่ากับ Lime_cnt3
            time.sleep(0.2)       # หน่วงเวลา 0.2 วินาที
        else:           # หากไม่เป็นจริง
            time.sleep(0.2)       # หน่วงเวลา 0.2 วินาที
        if Stop1 == 1:     # ถ้าตัวแปร Stop 1 เป็นจริงให้หยุดการนับจำนวนผลมะนาว
            print("Stop cnt..")
        # ฟังก์ชั่นการคัดแยกสีของผลมะนาว
def detect():
    print('Camera ON..')
    global Stop1     # เรียกใช้ตัวแปรภายนอกฟังก์ชัน
    pwm = Adafruit_PCA9685.PCA9685()  # กำหนดให้ pwm เป็นตัวแปรสำหรับการใช้งาน
บอร์ดขับเซอร์โวมอเตอร์
    servo_min = 275 # กำหนดองศาการหมุนเท่ากับ 120 องศา (ค่าความกว้างต่ำสุดของสัญญาณ
พัลส์จากความกว้างทั้งหมด 4096)
```

```
servo_max = 375   # กำหนดองศาการหมุนเท่ากับ 90 องศา (ค่าความกว้างต่ำสุดของสัญญาณพัลส์จากความกว้างทั้งหมด 4096)
servo_mid = 375   # กำหนดองศาการหมุนเท่ากับ 90 องศา (ค่าความกว้างต่ำสุดของสัญญาณพัลส์จากความกว้างทั้งหมด 4096)
```

 # ฟังก์ชั่นการทำงานบอร์ดขับเซอร์โวมอเตอร์

```
def set_servo_pulse(channel, pulse):
    pulse_length = 1000000 # 1,000,000 us per second
    pulse_length //= 60 # 60 Hz
    print ('{0}us per period'.format(pulse_length))
    pulse_length //= 4096 # 12 bits of resolution
    print ('{0}us per bit'.format(pulse_length))
    pulse *= 1000
    pulse //= pulse_length
    pwm.set_pwm(channel, 0, pulse)
```

 # เริ่มต้นการทำงานของโปรแกรม

```
pwm.set_pwm_freq(60)   # ตั้งค่าความถี่การทำงานของเซอร์โวมอเตอร์เท่ากับ 60 Hz
pwm.set_pwm(0, 0, servo_mid)   # ตั้งค่าองศาเริ่มต้นของเซอร์โวที่ 90 องศา
cap = cv2.VideoCapture(0)        # สร้าง Camera Object
pTime = 0       # กำหนดตัวแปร pTime = 0
cTime = 0       # กำหนดตัวแปร cTime = 0
while (True):     # วนรอบการทำงาน
    _, frame = cap.read()   # วนรอบจับภาพ Capture frame by frame (640X480)
    frame = cv2.flip(frame, -1)  # Mirror มุมกล้อง -180 องศา ในแกน X
    frame2 = cv2.resize(frame, None, fx=0.47, fy=0.4, interpolation = cv2.INTER_CUBIC)              # กำหนดความกว้างและความสูงของหน้าต่างแสดงผล
    frame_blur = cv2.GaussianBlur(frame2, (5, 5), -1)     # ลดสัญญาณรบกวนของข้อมูลภาพ
    hsv = cv2.cvtColor(frame2, cv2.COLOR_BGR2HSV)   # เปลี่ยนระบบสี BGR เป็นระบบสี HSV
    rgb = cv2.cvtColor(frame2, cv2.COLOR_BGR2RGB)   # เปลี่ยนระบบสี BGR เป็นระบบสี RGB
```

 # ฟังก์ชั่นการตรวจจับสีและคัดแยกสีผลมะนาวผลสีเขียว

```
    green = cv2.inRange(hsv, lower1, upper1)
```

```
        res_green = cv2.bitwise_and(frame_blur, frame_blur, mask=green)
        contours, hierarchy = cv2.findContours(green, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        for pic, contour in enumerate(contours):
            area_green = cv2.contourArea(contour)
            if area_green > 1100:
                x, y, w, h = cv2.boundingRect(contour)
                img = cv2.rectangle(rgb, (x, y), (x + w, y + h), (0, 0, 255), 2)
                cv2.putText(rgb, "green", (x, y + h),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0))
                print('green = ' + str(area_green))
                pwm.set_pwm(0, 0, servo_max)
            # ฟังก์ชั่นการตรวจจับสีและคัดแยกสีผลมะนาวผลสีเหลือง
        yellow = cv2.inRange(hsv, lower2, upper2)
        res_yellow = cv2.bitwise_and(frame_blur, frame_blur, mask=yellow)
        contours, hierarchy = cv2.findContours(yellow, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        for pic, contour in enumerate(contours):
            area_yellow = cv2.contourArea(contour)
            if area_yellow > 800:
                x, y, w, h = cv2.boundingRect(contour)
                img = cv2.rectangle(rgb, (x, y), (x + w, y + h), (0, 0, 255), 2)
                cv2.putText(rgb, "yellow", (x, y + h),
                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0))
                print('yellow = ' + str(area_yellow))
                pwm.set_pwm(0, 0, servo_min)
            # การเขียนแสดงผลเฟรมเรทของหน้าแสดงผลกล้อง
        cTime = time.time()
        fps = 1/(cTime-pTime)
        pTime = cTime
        cv2.putText(rgb, str(int(fps)), (10,20), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
                (0, 255, 0), 2)
            # การเขียนแสดงผลภาพเก็บไว้ในตัวแปรเพื่อไปแสดงบนหน้าจียูไอ
```

```
        image = Image.fromarray(rgb)
        iago = ImageTk.PhotoImage(image)
        label1.configure(image=iago)
        label1.image = iago
        if Stop1 == 1:     # ถ้าตัวแปร Stop 1 เป็นจริงให้หยุดการประมวลผลภาพ
            print("Stop process..")
            Stop1 = 0      # กำหนดให้ตัวแปร Stop1 เป็น 0
            gpio.cleanup()         # เคลียร์ล้างการตั้งค่าของขา GPIO
            cap.release()          # คืนทรัพยากร
            cv2.destroyAllWindows()     # คืนทรัพยากร
            return Stop1  # ส่งค่าตัวแปร Stop1 ออกนอกฟังก์ชั่น
            # ฟังก์ชั่นการเรียกใช้งานฟังก์ชั่นแบบมัลติเทรด(หลายฟังก์ชั่นพร้อมกัน)
def run():
    st1 = threading.Thread(target=detect)
    st2 = threading.Thread(target=motor)
    st3 = threading.Thread(target=cnt_lime)
    st1.start()
    st2.start()
    st3.start()
            # ฟังก์ชั่นหยุดการทำงานของโปรแกรม
def stop():
    global Stop1, Stop2
    status = messagebox.askyesno(title="Lime Grade", message="ต้องการหยุดการทำงาน
ใช่หรือไม่")
    if status > 0:
        Stop1, Stop2 = 1, 1
        time.sleep(2)
    if Stop1 == 1 or Stop2 == 1:
        Stop1,  Stop2 = 0, 0
        return Stop1, Stop2
            # ฟังก์ชั่นรีเซ็ทจำนวนนับผลมะนาวทั้งหมด
def reset():
    global Lime_cnt1, Lime_cnt2, Lime_cnt3
```

```
    status = messagebox.askyesno(title="Lime Grade", message="ต้องการรีเซ็ททั้งหมดใช่หรือไม่")
    if status > 0:
        print('reset')
        Lime_cnt1, Lime_cnt2, Lime_cnt3 = 0, 0, 0
        tv_Lime1.set(Lime_cnt1)
        tv_Lime2.set(Lime_cnt2)
        tv_Lime3.set(Lime_cnt3)
        return Lime_cnt1, Lime_cnt2, Lime_cnt3
            # ฟังก์ชั่นรีเซ็ทจำนวนนับผลมะนาวขนาดที่ 1
def reset_g1():
    global Lime_cnt1
    status = messagebox.askyesno(title="Lime Grade", message="ต้องการรีเซ็ทใช่หรือไม่")
    if status > 0:
        print('reset_g1')
        Lime_cnt1 = 0
        tv_Lime1.set(Lime_cnt1)
        return Lime_cnt1
            # ฟังก์ชั่นรีเซ็ทจำนวนนับผลมะนาวขนาดที่ 2
def reset_g2():
    global Lime_cnt2
    status = messagebox.askyesno(title="Lime Grade", message="ต้องการรีเซ็ทใช่หรือไม่")
    if status > 0:
        print('reset_g2')
        Lime_cnt2 = 0
        tv_Lime2.set(Lime_cnt2)
        return Lime_cnt2
            # ฟังก์ชั่นรีเซ็ทจำนวนนับผลมะนาวขนาดที่ 3
def reset_g3():
    global Lime_cnt3
    status = messagebox.askyesno(title="Lime Grade", message="ต้องการรีเซ็ทใช่หรือไม่")
    if status > 0:
        print('reset_g3')
```

```
        Lime_cnt3 = 0
        tv_Lime3.set(Lime_cnt3)
        return Lime_cnt3
            # ฟังก์ชั่นใช้งานออกจากโปรแกรม
def btn_exit():
    global Stop1, Stop2
    status = messagebox.askyesno(title="Lime Grade", message="คุณต้องการออกจาก
โปรแกรมใช่หรือไม่")
    if status > 0:
        Stop1, Stop2 = 1, 1
        time.sleep(1)
        sys.exit()
            # กำหนดค่าสีต่ำสุดและสูงสุดของมะนาวผลสีเขียว
lower1 = np.array([35, 100, 100], np.uint8)
upper1 = np.array([67, 255, 255], np.uint8)
            # กำหนดค่าสีต่ำสุดและสูงสุดของมะนาวผลสีเหลือง
lower2 = np.array([21, 65, 65], np.uint8)
upper2 = np.array([31, 255, 255], np.uint8)
            # โค้ดการออกแบบหน้าต่างจียูไอ
root = Tk()
width_value = root.winfo_screenwidth()
height_value = root.winfo_screenheight()
root.geometry("%dx%d+0+0" % (width_value, height_value))
root.title("Lime Grade")
root.configure(background='gray80')
root.resizable(width=False, height=False)
frame = Frame(root, bg='gray80', bd=5)
frame.place(relx=0.5, rely=0.1, relwidth=0.75, relheight=0.1, anchor='n')
button = Button(frame, text='START', font=('consolas', 20, 'bold'), command=run,
fg='white', bg='dark green')
button.place(relx=0.02, relwidth=0.3, relheight=1)
button1 = Button(frame, text='STOP', font=('consolas', 20, 'bold'), command=stop,
fg='white', bg='red')
```

```
button1.place(relx=0.35, relwidth=0.3, relheight=1)
button2 = Button(frame, text='RESET', font=('consolas', 20, 'bold'), command=reset,
fg='white', bg='tan1')
button2.place(relx=0.675, relwidth=0.3, relheight=1)
b1 = Button(text='ออกจากโปรแกรม', font=18, command=btn_exit, bg='gray77')
b1.pack(anchor='e')
lower_frame = Frame(root, bg='gray10', bd=5)
lower_frame.place(relx=0.5, rely=0.28, relwidth=0.40, relheight=0.45)
label1 = Label(lower_frame)
label1.place(x=2.5, y=2.5)
#FrameGradeLime
G1 = Frame(root, bg='gray90', bd=5)
G1.place(relx=0.2, rely=0.28, relwidth=0.25, relheight=0.11)
lg1 = Label(G1, textvariable=tv_Lime1, font=('consolas', 20), bg='gray90', fg= 'lime
green')
lg1.place(relx=0, relwidth=0.7, relheight=1)
G2 = Frame(root, bg='gray90', bd=5)
G2.place(relx=0.2, rely=0.45, relwidth=0.25, relheight=0.11)
lg2 = Label(G2, textvariable=tv_Lime2, font=('consolas', 20), bg='gray90', fg= 'lime
green')
lg2.place(relx=0, relwidth=0.7, relheight=1)
G3 = Frame(root, bg='gray90', bd=5)
G3.place(relx=0.2, rely=0.62, relwidth=0.25, relheight=0.11)
lg3 = Label(G3, textvariable=tv_Lime3, font=('consolas', 20), bg='gray90', fg= 'lime
green')
lg3.place(relx=0, relwidth=0.7, relheight=1)
btn_reset1 = Button(root, text='Grade 1', font=('consolas', 16, 'bold'),
command=reset_g1, fg='white', bg='gray10')
btn_reset1.place(relx=0.076, rely=0.28, relwidth=0.15, relheight=0.11)
btn_reset2 = Button(root, text='Grade 2', font=('consolas', 16, 'bold'),
command=reset_g2, fg='white', bg='gray10')
btn_reset2.place(relx=0.076, rely=0.45, relwidth=0.15, relheight=0.11)
```

```
btn_reset3 = Button(root, text='Grade 3', font=('consolas', 16, 'bold'),
command=reset_g3, fg='white', bg='gray10')
btn_reset3.place(relx=0.076, rely=0.62, relwidth=0.15, relheight=0.11)
root.mainloop()
```

ภาคผนวก ข

คู่มือการใช้งานเครื่องคัดแยกสีและขนาดของผลมะนาว

คู่มือการใช้งานเครื่องแยกสีและขนาดผลมะนาวกึ่งอัตโนมัติ

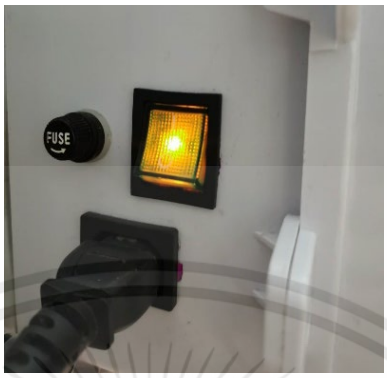
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

ปีการศึกษา 2563

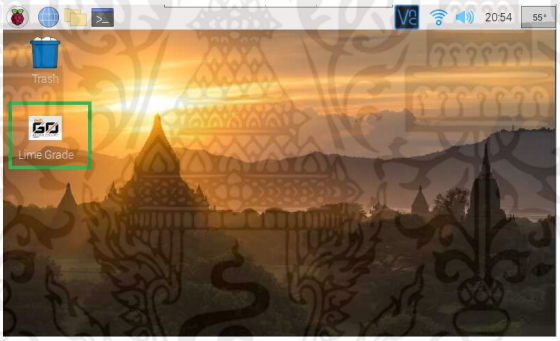
ขั้นตอนการใช้งานเครื่องคัดแยกสีและขนาดของผลมะนาว

1. เปิดสวิตช์การทำงานที่กล่องควบคุม ดังรูปที่ ข.1



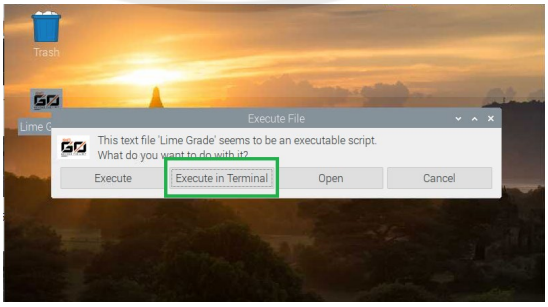
รูปที่ ข.1 สวิตช์ เปิด-ปิด เครื่องคัดแยกสีและขนาดผลมะนาว

2. สำผัสที่ไอคอน Lime Grade ดังรูปที่ ข.2



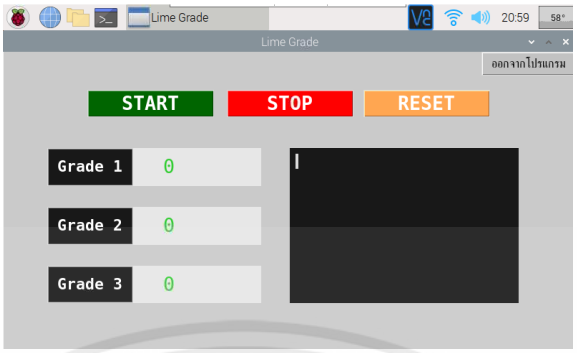
รูปที่ ข.2 โปรแกรมคัดแยกสีและขนาดผลมะนาว

3. เลือก Execute in Terminal ดังรูปที่ ข.3



รูปที่ ข.3 เลือก Execute in Terminal

4. ระบบจะแสดงหน้าต่าง GUI ดังรูปที่ ข.4



รูปที่ ข.4 หน้าต่างแสดงผล GUI

4.1 เมื่อสัมผัสที่ ปุ่ม START สีเขียว เครื่องคัดแยกสีและขนาดผลมะนาวจะเริ่มทำงานและแสดงผลการประมวลผลภาพ

4.2 เมื่อสัมผัสที่ ปุ่ม STOP สีแดง เครื่องคัดแยกสีและขนาดผลมะนาวจะหยุดการทำงาน

4.3 เมื่อสัมผัสที่ ปุ่ม RESET สีส้ม จำนวนนับของผลมะนาวแต่ละขนาดจะถูกกำหนดให้เป็นศูนย์

4.3.1 เมื่อสัมผัสที่ GRADE 1 สีดำ จำนวนนับของผลมะนาวขนาดที่ 1 จะถูกกำหนดให้เป็นศูนย์

4.3.1 เมื่อสัมผัสที่ GRADE 2 สีดำ จำนวนนับของผลมะนาวขนาดที่ 2 จะถูกกำหนดให้เป็นศูนย์

4.3.1 เมื่อสัมผัสที่ GRADE 3 สีดำ จำนวนนับของผลมะนาวขนาดที่ 3 จะถูกกำหนดให้เป็นศูนย์

5. จากข้อ 4.1 เมื่อกดปุ่ม START ให้เทผลมะนาวลงในช่องใส่ผลมะนาว ดังรูปที่ ข.5

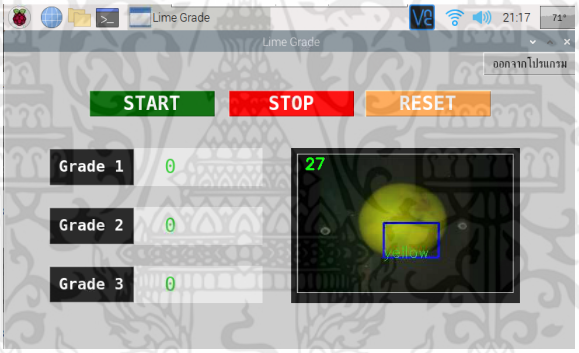
รูปที่ ข.5 เทผลมะนาวลงในช่องใส่ผลมะนาว

6. การประมวลผลภาพ

6.1 การประมวลผลภาพมะนาวประเภทที่ 1 ผลสีเขียว ดังรูปที่ ข.6

รูปที่ ข.6 การประมวลผลภาพมะนาวประเภทที่ 1

6.2 การประมวลผลภาพมะนาวประเภทที่ 2 ผลสีเหลือง ดังรูปที่ ข.7



รูปที่ ข.7 การประมวลผลภาพมะนาวประเภทที่ 2

7. ผลการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาว ดังรูปที่ ข.8

รูปที่ ข.8 ผลการทำงานของเครื่องคัดแยกสีและขนาดของผลมะนาว

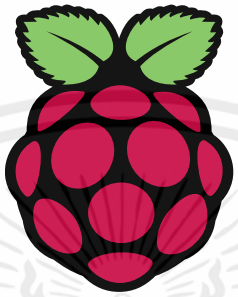
8. หน้าต่าง GUI แสดงจำนวณนับของผลมะนาวแต่ละขนาดหลังจากคัดแยก ดังรูปที่ ข.9

รูปที่ ข.9 แสดงจำนวณนับของผลมะนาวแต่ละขนาดหลังจากคัดแยก

ภาคผนวก ค

คู่มือการใช้งานอุปกรณ์(Datasheet)

DATASHEET



Raspberry Pi 4 Model B

Release 1

June 2019

Table 1: Release History

Release	Date	Description
1	21/06/2019	First release

The latest release of this document can be found at `https://www.raspberrypi.org`

Contents

List of Figures

List of Tables

1 Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with siginficantly enhanced CPU, GPU and I/O performance; all within a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is avaiable with either 1, 2 and 4 Gigabytes of LPDDR4 SDRAM.

2 Features

2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs

2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained
  - Recent Linux kernel support
  - Many drivers upstreamed
  - Stable and well supported userland
  - Availability of GPU functions using standard APIs

3 Mechanical Specification

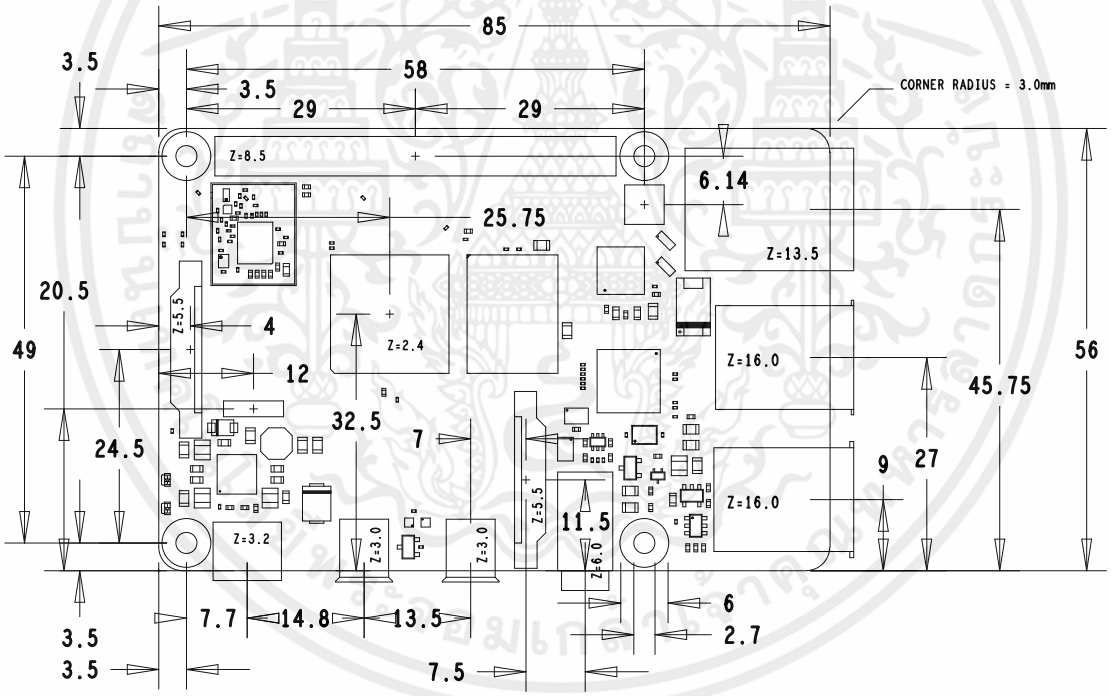


Figure 1: Mechanical Dimensions

4 Electrical Specification

Caution! Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Symbol	Parameter	Minimum	Maximum	Unit
VIN	5V Input Voltage	-0.5	6.0	V

Table 2: Absolute Maximum Ratings

Please note that VDD_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.

Symbol	Parameter	Conditions	Minimum	Typical	Maximum	Unit
V_{IL}	Input low voltage[a]	VDD_IO = 3.3V	-	-	TBD	V
V_{IH}	Input high voltage[a]	VDD_IO = 3.3V	TBD	-	-	V
I_{IL}	Input leakage current	TA = +85°C	-	-	TBD	μA
C_{IN}	Input capacitance	-	-	TBD	-	pF
V_{OL}	Output low voltage[b]	VDD_IO = 3.3V, IOL = -2mA	-	-	TBD	V
V_{OH}	Output high voltage[b]	VDD_IO = 3.3V, IOH = 2mA	TBD	-	-	V
I_{OL}	Output low current[c]	VDD_IO = 3.3V, VO = 0.4V	TBD	-	-	mA
I_{OH}	Output high current[c]	VDD_IO = 3.3V, VO = 2.3V	TBD	-	-	mA
R_{PU}	Pullup resistor	-	TBD	-	TBD	kΩ
R_{PD}	Pulldown resistor	-	TBD	-	TBD	kΩ

[a] Hysteresis enabled
[b] Default drive strength (8mA)
[c] Maximum drive strength (16mA)

Table 3: DC Characteristics

Pin Name	Symbol	Parameter	Minimum	Typical	Maximum	Unit
Digital outputs	t_{rise}	10-90% rise time[a]	-	TBD	-	ns
Digital outputs	t_{fall}	90-10% fall time[a]	-	TBD	-	ns

[a] Default drive strength, CL = 5pF, VDD_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics

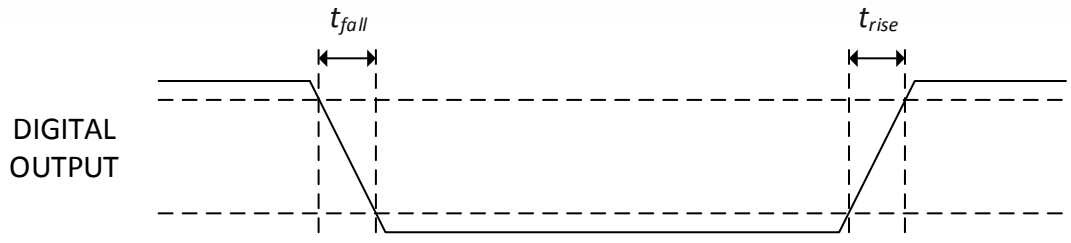


Figure 2: Digital IO Characteristics

4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.

5 Peripherals

5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

5.1.1 GPIO Pin Assignments

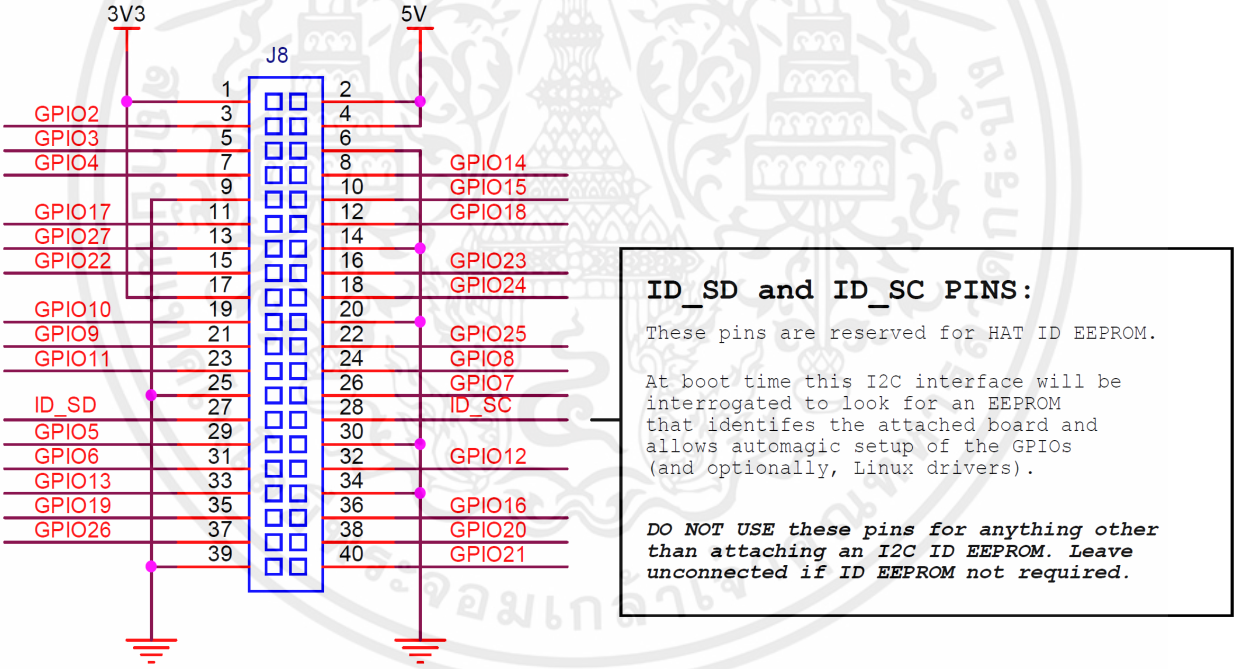


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi4. This gives users much more flexibility when attaching add-on hardware as compared to older models.

5.1.2 GPIO Alternate Functions

GPIO	Default Pull	ALT0	ALT1	ALT2	ALT3	ALT4	ALT5
0	High	SDA0	SA5	PCLK	SPI3_CE0_N	TXD2	SDA6
1	High	SCL0	SA4	DE	SPI3_MISO	RXD2	SCL6
2	High	SDA1	SA3	LCD_VSYNC	SPI3_MOSI	CTS2	SDA3
3	High	SCL1	SA2	LCD_HSYNC	SPI3_SCLK	RTS2	SCL3
4	High	GPCLK0	SA1	DPI_D0	SPI4_CE0_N	TXD3	SDA3
5	High	GPCLK1	SA0	DPI_D1	SPI4_MISO	RXD3	SCL3
6	High	GPCLK2	SOE_N	DPI_D2	SPI4_MOSI	CTS3	SDA4
7	High	SPI0_CE1_N	SWE_N	DPI_D3	SPI4_SCLK	RTS3	SCL4
8	High	SPI0_CE0_N	SD0	DPI_D4	-	TXD4	SDA4
9	Low	SPI0_MISO	SD1	DPI_D5	-	RXD4	SCL4
10	Low	SPI0_MOSI	SD2	DPI_D6	-	CTS4	SDA5
11	Low	SPI0_SCLK	SD3	DPI_D7	-	RTS4	SCL5
12	Low	PWM0	SD4	DPI_D8	SPI5_CE0_N	TXD5	SDA5
13	Low	PWM1	SD5	DPI_D9	SPI5_MISO	RXD5	SCL5
14	Low	TXD0	SD6	DPI_D10	SPI5_MOSI	CTS5	TXD1
15	Low	RXD0	SD7	DPI_D11	SPI5_SCLK	RTS5	RXD1
16	Low	FL0	SD8	DPI_D12	CTS0	SPI1_CE2_N	CTS1
17	Low	FL1	SD9	DPI_D13	RTS0	SPI1_CE1_N	RTS1
18	Low	PCM_CLK	SD10	DPI_D14	SPI6_CE0_N	SPI1_CE0_N	PWM0
19	Low	PCM_FS	SD11	DPI_D15	SPI6_MISO	SPI1_MISO	PWM1
20	Low	PCM_DIN	SD12	DPI_D16	SPI6_MOSI	SPI1_MOSI	GPCLK0
21	Low	PCM_DOUT	SD13	DPI_D17	SPI6_SCLK	SPI1_SCLK	GPCLK1
22	Low	SD0_CLK	SD14	DPI_D18	SD1_CLK	ARM_TRST	SDA6
23	Low	SD0_CMD	SD15	DPI_D19	SD1_CMD	ARM_RTCK	SCL6
24	Low	SD0_DAT0	SD16	DPI_D20	SD1_DAT0	ARM_TDO	SPI3_CE1_N
25	Low	SD0_DAT1	SD17	DPI_D21	SD1_DAT1	ARM_TCK	SPI4_CE1_N
26	Low	SD0_DAT2	TE0	DPI_D22	SD1_DAT2	ARM_TDI	SPI5_CE1_N
27	Low	SD0_DAT3	TE1	DPI_D23	SD1_DAT3	ARM_TMS	SPI6_CE1_N

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the hardware documentation section of the website.

5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which suports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celcius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high termperature at full performance, further cooling may be needed.

6 Availability

Raspberry Pi guarantee availability Pi4B until at least January 2026.

7 Support

For support please see the hardware documentation section of the Raspberry Pi website and post questions to the Raspberry Pi forum.

PCA9685

16-channel, 12-bit PWM Fm+ I²C-bus LED controller

Rev. 3 — 2 September 2010 Product data sheet

1. General description

The PCA9685 is an I²C-bus controlled 16-channel LED controller optimized for LCD Red/Green/Blue/Amber (RGBA) color backlighting applications. Each LED output has its own 12-bit resolution (4096 steps) fixed frequency individual PWM controller that operates at a programmable frequency from a typical of 40 Hz to 1000 Hz with a duty cycle that is adjustable from 0 % to 100 % to allow the LED to be set to a specific brightness value. All outputs are set to the same PWM frequency.

Each LED output can be off or on (no PWM control), or set at its individual PWM controller value. The LED output driver is programmed to be either open-drain with a 25 mA current sink capability at 5 V or totem pole with a 25 mA sink, 10 mA source capability at 5 V. The PCA9685 operates with a supply voltage range of 2.3 V to 5.5 V and the inputs and outputs are 5.5 V tolerant. LEDs can be directly connected to the LED output (up to 25 mA, 5.5 V) or controlled with external drivers and a minimum amount of discrete components for larger current or higher voltage LEDs.

The PCA9685 is in the new Fast-mode Plus (Fm+) family. Fm+ devices offer higher frequency (up to 1 MHz) and more densely populated bus operation (up to 4000 pF).

Although the PCA9635 and PCA9685 have many similar features, the PCA9685 has some unique features that make it more suitable for applications such as LCD backlighting and Ambilight:

- The PCA9685 allows staggered LED output on and off times to minimize current surges. The on and off time delay is independently programmable for each of the 16 channels. This feature is not available in PCA9635.
- The PCA9685 has 4096 steps (12-bit PWM) of individual LED brightness control. The PCA9635 has only 256 steps (8-bit PWM).
- When multiple LED controllers are incorporated in a system, the PWM pulse widths between multiple devices may differ if PCA9635s are used. The PCA9685 has a programmable prescaler to adjust the PWM pulse widths of multiple devices.
- The PCA9685 has an external clock input pin that will accept user-supplied clock (50 MHz max.) in place of the internal 25 MHz oscillator. This feature allows synchronization of multiple devices. The PCA9635 does not have external clock input feature.
- Like the PCA9635, PCA9685 also has a built-in oscillator for the PWM control. However, the frequency used for PWM control in the PCA9685 is adjustable from about 40 Hz to 1000 Hz as compared to the typical 97.6 kHz frequency of the PCA9635. This allows the use of PCA9685 with external power supply controllers. All bits are set at the same frequency.
- The Power-On Reset (POR) default state of LEDn output pins is LOW in the case of PCA9685. It is HIGH for PCA9635.

The active LOW Output Enable input pin ($\overline{OE}$) allows asynchronous control of the LED outputs and can be used to set all the outputs to a defined I²C-bus programmable logic state. The $\overline{OE}$ can also be used to externally 'pulse width modulate' the outputs, which is useful when multiple devices need to be dimmed or blinked together using software control.

Software programmable LED All Call and three Sub Call I²C-bus addresses allow all or defined groups of PCA9685 devices to respond to a common I²C-bus address, allowing for example, all red LEDs to be turned on or off at the same time or marquee chasing effect, thus minimizing I²C-bus commands. Six hardware address pins allow up to 62 devices on the same bus.

The Software Reset (SWRST) General Call allows the master to perform a reset of the PCA9685 through the I²C-bus, identical to the Power-On Reset (POR) that initializes the registers to their default state causing the outputs to be set LOW. This allows an easy and quick way to reconfigure all device registers to the same condition via software.

2. Features and benefits

- 16 LED drivers. Each output programmable at:
  - Off
  - On
  - Programmable LED brightness
  - Programmable LED turn-on time to help reduce EMI
- 1 MHz Fast-mode Plus compatible I²C-bus interface with 30 mA high drive capability on SDA output for driving high capacitive buses
- 4096-step (12-bit) linear programmable brightness per LED output varying from fully off (default) to maximum brightness
- LED output frequency (all LEDs) typically varies from 40 Hz to 1000 Hz (Default of 1Eh in PRE_SCALE register results in a 200 Hz refresh rate with oscillator clock of 25 MHz.)
- Sixteen totem pole outputs (sink 25 mA and source 10 mA at 5 V) with software programmable open-drain LED outputs selection (default at totem pole). No input function.
- Output state change programmable on the Acknowledge or the STOP Command to update outputs byte-by-byte or all at the same time (default to 'Change on STOP').
- Active LOW Output Enable ($\overline{OE}$) input pin. LEDn outputs programmable to logic 1, logic 0 (default at power-up) or 'high-impedance' when $\overline{OE}$ is HIGH.
- 6 hardware address pins allow 62 PCA9685 devices to be connected to the same I²C-bus
- Toggling $\overline{OE}$ allows for hardware LED blinking
- 4 software programmable I²C-bus addresses (one LED All Call address and three LED Sub Call addresses) allow groups of devices to be addressed at the same time in any combination (for example, one register used for 'All Call' so that all the PCA9685s on the I²C-bus can be addressed at the same time and the second register used for three different addresses so that $\frac{1}{3}$ of all devices on the bus can be addressed at the same time in a group). Software enable and disable for these I²C-bus address.
- Software Reset feature (SWRST General Call) allows the device to be reset through the I²C-bus

- 25 MHz typical internal oscillator requires no external components
- External 50 MHz (max.) clock input
- Internal power-on reset
- Noise filter on SDA/SCL inputs
- Edge rate control on outputs
- No output glitches on power-up
- Supports hot insertion
- Low standby current
- Operating power supply voltage range of 2.3 V to 5.5 V
- 5.5 V tolerant inputs
- −40 °C to +85 °C operation
- ESD protection exceeds 2000 V HBM per JESD22-A114, 200 V MM per JESD22-A115 and 1000 V CDM per JESD22-C101
- Latch-up testing is done to JEDEC Standard JESD78 which exceeds 100 mA
- Packages offered: TSSOP28, HVQFN28

3. Applications

- RGB or RGBA LED drivers
- LED status information
- LED displays
- LCD backlights
- Keypad backlights for cellular phones or handheld devices

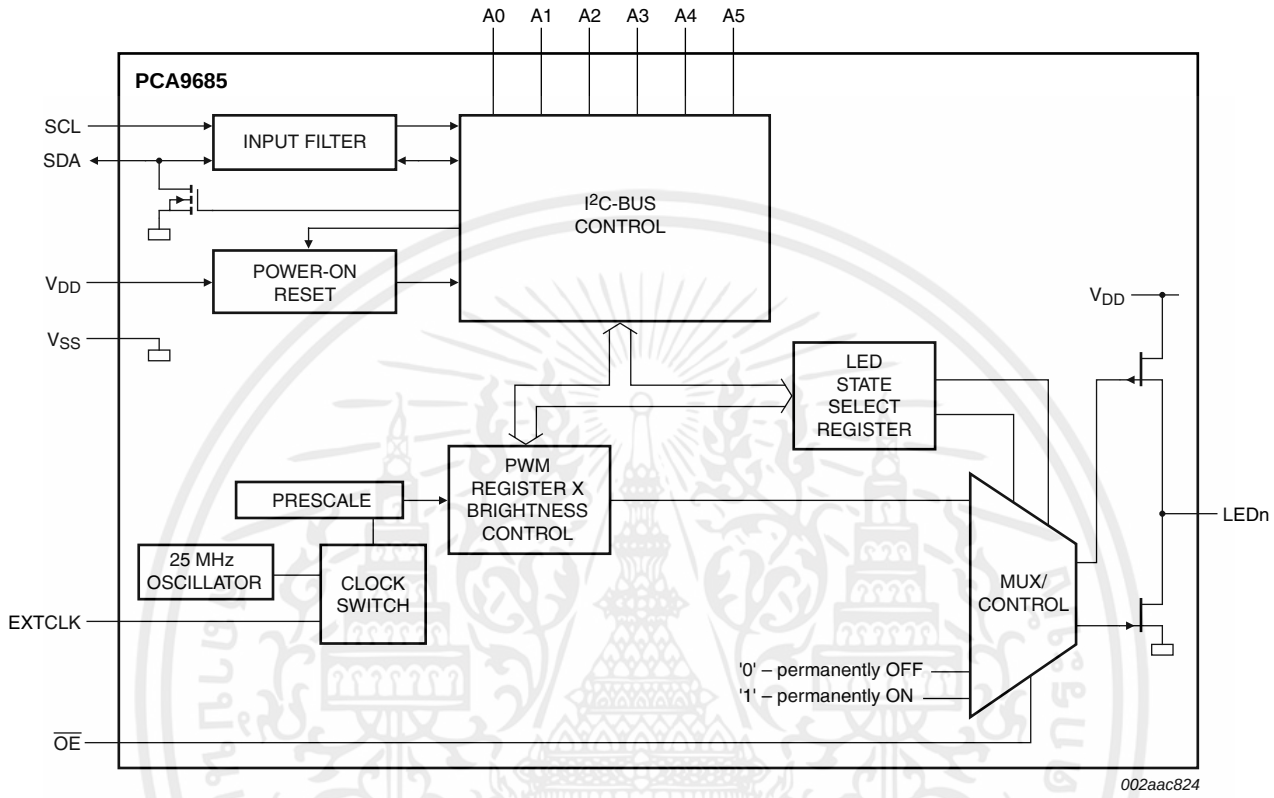
4. Ordering information

Table 1. Ordering information

Type number	Topside mark	Package		
		Name	Description	Version
PCA9685PW	PCA9685PW	TSSOP28	plastic thin shrink small outline package; 28 leads; body width 4.4 mm	SOT361-1
PCA9685PW/Q900[1]	PCA9685PW	TSSOP28	plastic thin shrink small outline package; 28 leads; body width 4.4 mm	SOT361-1
PCA9685BS	P9685	HVQFN28	plastic thermal enhanced very thin quad flat package; no leads; 28 terminals; body 6 × 6 × 0.85 mm	SOT788-1

[1] PCA9685PW/Q900 is AEC-Q100 compliant. Contact **i2c.support@nxp.com** for PPAP.

5. Block diagram



Remark: Only one LED output shown for clarity.

Fig 1. Block diagram of PCA9685

6. Pinning information

6.1 Pinning

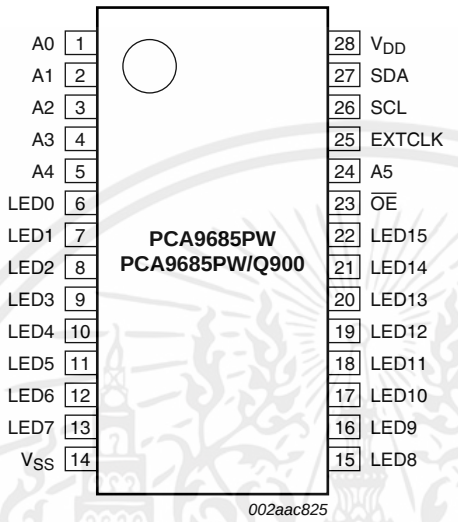


Fig 2. Pin configuration for TSSOP28

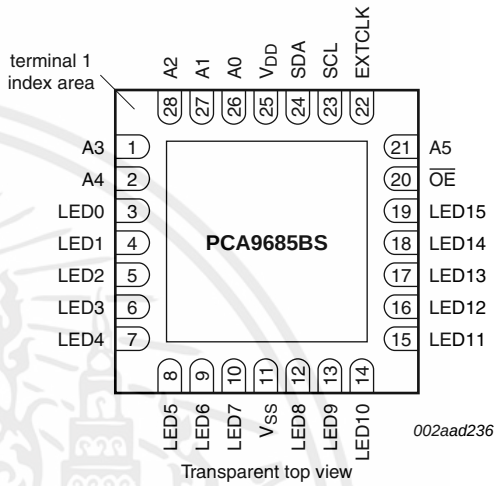


Fig 3. Pin configuration for HVQFN28

6.2 Pin description

Table 2. Pin description

Symbol	Pin		Type	Description
	TSSOP28	HVQFN28		
A0	1	26	I	address input 0
A1	2	27	I	address input 1
A2	3	28	I	address input 2
A3	4	1	I	address input 3
A4	5	2	I	address input 4
LED0	6	3	O	LED driver 0
LED1	7	4	O	LED driver 1
LED2	8	5	O	LED driver 2
LED3	9	6	O	LED driver 3
LED4	10	7	O	LED driver 4
LED5	11	8	O	LED driver 5
LED6	12	9	O	LED driver 6
LED7	13	10	O	LED driver 7
V_{SS}	14	11[1]	power supply	supply ground
LED8	15	12	O	LED driver 8
LED9	16	13	O	LED driver 9
LED10	17	14	O	LED driver 10
LED11	18	15	O	LED driver 11

Table 2. Pin description *...continued*

Symbol	Pin		Type	Description
	TSSOP28	HVQFN28		
LED12	19	16	O	LED driver 12
LED13	20	17	O	LED driver 13
LED14	21	18	O	LED driver 14
LED15	22	19	O	LED driver 15
$\overline{OE}$	23	20	I	active LOW output enable
A5	24	21	I	address input 5
EXTCLK	25	22	I	external clock input[2]
SCL	26	23	I	serial clock line
SDA	27	24	I/O	serial data line
V_{DD}	28	25	power supply	supply voltage

[1] HVQFN28 package die supply ground is connected to both V_{SS} pin and exposed center pad. V_{SS} pin must be connected to supply ground for proper device operation. For enhanced thermal, electrical, and board level performance, the exposed pad needs to be soldered to the board using a corresponding thermal pad on the board and for proper heat conduction through the board, thermal vias need to be incorporated in the PCB in the thermal pad region.

[2] This pin must be grounded when this feature is not used.

7. Functional description

Refer to Figure 1 “Block diagram of PCA9685”.

7.1 Device addresses

Following a START condition, the bus master must output the address of the slave it is accessing.

There are a maximum of 64 possible programmable addresses using the 6 hardware address pins. Two of these addresses, Software Reset and LED All Call, cannot be used because their default power-up state is ON, leaving a maximum of 62 addresses. Using other reserved addresses, as well as any other subcall address, will reduce the total number of possible addresses even further.

7.1.1 Regular I²C-bus slave address

The I²C-bus slave address of the PCA9685 is shown in Figure 4. To conserve power, no internal pull-up resistors are incorporated on the hardware selectable address pins and they must be pulled HIGH or LOW.

Remark: Using reserved I²C-bus addresses will interfere with other devices, but only if the devices are on the bus and/or the bus will be open to other I²C-bus systems at some later date. In a closed system where the designer controls the address assignment these addresses can be used since the PCA9685 treats them like any other address. The LED All Call, Software Reset and PCA9564 or PCA9665 slave address (if on the bus) can never be used for individual device addresses.

- PCA9685 LED All Call address (1110 000) and Software Reset (0000 0110) which are active on start-up

- PCA9564 (0000 000) or PCA9665 (1110 000) slave address which is active on start-up
- 'reserved for future use' I²C-bus addresses (0000 011, 1111 1XX)
- slave devices that use the 10-bit addressing scheme (1111 0XX)
- slave devices that are designed to respond to the General Call address (0000 000) which is used as the software reset address
- High-speed mode (Hs-mode) master code (0000 1XX)

Fig 4. Slave address

The last bit of the address byte defines the operation to be performed. When set to logic 1 a read is selected, while a logic 0 selects a write operation.

7.1.2 LED All Call I²C-bus address

- Default power-up value (ALLCALLADR register): E0h or 1110 000X
- Programmable through I²C-bus (volatile programming)
- At power-up, LED All Call I²C-bus address is enabled. PCA9685 sends an ACK when E0h (R/$\overline{W}$ = 0) or E1h (R/$\overline{W}$ = 1) is sent by the master.

See Section 7.3.7 "ALLCALLADR, LED All Call I²C-bus address" for more detail.

Remark: The default LED All Call I²C-bus address (E0h or 1110 000X) must not be used as a regular I²C-bus slave address since this address is enabled at power-up. All the PCA9685s on the I²C-bus will acknowledge the address if sent by the I²C-bus master.

7.1.3 LED Sub Call I²C-bus addresses

- 3 different I²C-bus addresses can be used
- Default power-up values:
  - SUBADR1 register: E2h or 1110 001X
  - SUBADR2 register: E4h or 1110 010X
  - SUBADR3 register: E8h or 1110 100X
- Programmable through I²C-bus (volatile programming)
- At power-up, Sub Call I²C-bus addresses are disabled. PCA9685 does not send an ACK when E2h (R/$\overline{W}$ = 0) or E3h (R/$\overline{W}$ = 1), E4h (R/$\overline{W}$ = 0) or E5h (R/$\overline{W}$ = 1), or E8h (R/$\overline{W}$ = 0) or E9h (R/$\overline{W}$ = 1) is sent by the master.

See Section 7.3.6 "SUBADR1 to SUBADR3, I²C-bus subaddress 1 to 3" for more detail.

Remark: The default LED Sub Call I²C-bus addresses may be used as regular I²C-bus slave addresses as long as they are disabled.

7.1.4 Software Reset I²C-bus address

The address shown in Figure 5 is used when a reset of the PCA9685 needs to be performed by the master. The Software Reset address (SWRST Call) must be used with R/$\overline{W}$ = logic 0. If R/$\overline{W}$ = logic 1, the PCA9685 does not acknowledge the SWRST. See Section 7.6 “Software reset” for more detail.

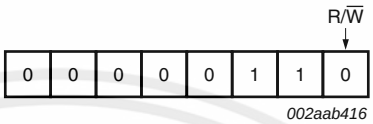


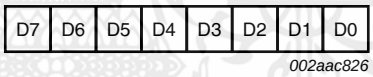
Fig 5. Software Reset address

Remark: The Software Reset I²C-bus address is a reserved address and cannot be used as a regular I²C-bus slave address or as an LED All Call or LED Sub Call address.

7.2 Control register

Following the successful acknowledgement of the slave address, LED All Call address or LED Sub Call address, the bus master will send a byte to the PCA9685, which will be stored in the Control register.

This register is used as a pointer to determine which register will be accessed.



reset state = 00h

Remark: The Control register does not apply to the Software Reset I²C-bus address.

Fig 6. Control register

7.3 Register definitions

Table 3. Register summary

Register # (decimal)	Register # (hex)	D7	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
0	00	0	0	0	0	0	0	0	0	MODE1	read/write	Mode register 1
1	01	0	0	0	0	0	0	0	1	MODE2	read/write	Mode register 2
2	02	0	0	0	0	0	0	1	0	SUBADR1	read/write	I²C-bus subaddress 1
3	03	0	0	0	0	0	0	1	1	SUBADR2	read/write	I²C-bus subaddress 2
4	04	0	0	0	0	0	1	0	0	SUBADR3	read/write	I²C-bus subaddress 3
5	05	0	0	0	0	0	1	0	1	ALLCALLADR	read/write	LED All Call I²C-bus address
6	06	0	0	0	0	0	1	1	0	LED0_ON_L	read/write	LED0 output and brightness control byte 0
7	07	0	0	0	0	0	1	1	1	LED0_ON_H	read/write	LED0 output and brightness control byte 1
8	08	0	0	0	0	1	0	0	0	LED0_OFF_L	read/write	LED0 output and brightness control byte 2
9	09	0	0	0	0	1	0	0	1	LED0_OFF_H	read/write	LED0 output and brightness control byte 3
10	0A	0	0	0	0	1	0	1	0	LED1_ON_L	read/write	LED1 output and brightness control byte 0
11	0B	0	0	0	0	1	0	1	1	LED1_ON_H	read/write	LED1 output and brightness control byte 1
12	0C	0	0	0	0	1	1	0	0	LED1_OFF_L	read/write	LED1 output and brightness control byte 2
13	0D	0	0	0	0	1	1	0	1	LED1_OFF_H	read/write	LED1 output and brightness control byte 3
14	0E	0	0	0	0	1	1	1	0	LED2_ON_L	read/write	LED2 output and brightness control byte 0
15	0F	0	0	0	0	1	1	1	1	LED2_ON_H	read/write	LED2 output and brightness control byte 1
16	10	0	0	0	1	0	0	0	0	LED2_OFF_L	read/write	LED2 output and brightness control byte 2
17	11	0	0	0	1	0	0	0	1	LED2_OFF_H	read/write	LED2 output and brightness control byte 3
18	12	0	0	0	1	0	0	1	0	LED3_ON_L	read/write	LED3 output and brightness control byte 0
19	13	0	0	0	1	0	0	1	1	LED3_ON_H	read/write	LED3 output and brightness control byte 1
20	14	0	0	0	1	0	1	0	0	LED3_OFF_L	read/write	LED3 output and brightness control byte 2
21	15	0	0	0	1	0	1	0	1	LED3_OFF_H	read/write	LED3 output and brightness control byte 3

Table 3. Register summary *...continued*

Register # (decimal)	Register # (hex)	D7	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
22	16	0	0	0	1	0	1	1	0	LED4_ON_L	read/write	LED4 output and brightness control byte 0
23	17	0	0	0	1	0	1	1	1	LED4_ON_H	read/write	LED4 output and brightness control byte 1
24	18	0	0	0	1	1	0	0	0	LED4_OFF_L	read/write	LED4 output and brightness control byte 2
25	19	0	0	0	1	1	0	0	1	LED4_OFF_H	read/write	LED4 output and brightness control byte 3
26	1A	0	0	0	1	1	0	1	0	LED5_ON_L	read/write	LED5 output and brightness control byte 0
27	1B	0	0	0	1	1	0	1	1	LED5_ON_H	read/write	LED5 output and brightness control byte 1
28	1C	0	0	0	1	1	1	0	0	LED5_OFF_L	read/write	LED5 output and brightness control byte 2
29	1D	0	0	0	1	1	1	0	1	LED5_OFF_H	read/write	LED5 output and brightness control byte 3
30	1E	0	0	0	1	1	1	1	0	LED6_ON_L	read/write	LED6 output and brightness control byte 0
31	1F	0	0	0	1	1	1	1	1	LED6_ON_H	read/write	LED6 output and brightness control byte 1
32	20	0	0	1	0	0	0	0	0	LED6_OFF_L	read/write	LED6 output and brightness control byte 2
33	21	0	0	1	0	0	0	0	1	LED6_OFF_H	read/write	LED6 output and brightness control byte 3
34	22	0	0	1	0	0	0	1	0	LED7_ON_L	read/write	LED7 output and brightness control byte 0
35	23	0	0	1	0	0	0	1	1	LED7_ON_H	read/write	LED7 output and brightness control byte 1
36	24	0	0	1	0	0	1	0	0	LED7_OFF_L	read/write	LED7 output and brightness control byte 2
37	25	0	0	1	0	0	1	0	1	LED7_OFF_H	read/write	LED7 output and brightness control byte 3
38	26	0	0	1	0	0	1	1	0	LED8_ON_L	read/write	LED8 output and brightness control byte 0
39	27	0	0	1	0	0	1	1	1	LED8_ON_H	read/write	LED8 output and brightness control byte 1
40	28	0	0	1	0	1	0	0	0	LED8_OFF_L	read/write	LED8 output and brightness control byte 2
41	29	0	0	1	0	1	0	0	1	LED8_OFF_H	read/write	LED8 output and brightness control byte 3

Table 3. Register summary *...continued*

Register # (decimal)	Register # (hex)	D7	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
42	2A	0	0	1	0	1	0	1	0	LED9_ON_L	read/write	LED9 output and brightness control byte 0
43	2B	0	0	1	0	1	0	1	1	LED9_ON_H	read/write	LED9 output and brightness control byte 1
44	2C	0	0	1	0	1	1	0	0	LED9_OFF_L	read/write	LED9 output and brightness control byte 2
45	2D	0	0	1	0	1	1	0	1	LED9_OFF_H	read/write	LED9 output and brightness control byte 3
46	2E	0	0	1	0	1	1	1	0	LED10_ON_L	read/write	LED10 output and brightness control byte 0
47	2F	0	0	1	0	1	1	1	1	LED10_ON_H	read/write	LED10 output and brightness control byte 1
48	30	0	0	1	1	0	0	0	0	LED10_OFF_L	read/write	LED10 output and brightness control byte 2
49	31	0	0	1	1	0	0	0	1	LED10_OFF_H	read/write	LED10 output and brightness control byte 3
50	32	0	0	1	1	0	0	1	0	LED11_ON_L	read/write	LED11 output and brightness control byte 0
51	33	0	0	1	1	0	0	1	1	LED11_ON_H	read/write	LED11 output and brightness control byte 1
52	34	0	0	1	1	0	1	0	0	LED11_OFF_L	read/write	LED11 output and brightness control byte 2
53	35	0	0	1	1	0	1	0	1	LED11_OFF_H	read/write	LED11 output and brightness control byte 3
54	36	0	0	1	1	0	1	1	0	LED12_ON_L	read/write	LED12 output and brightness control byte 0
55	37	0	0	1	1	0	1	1	1	LED12_ON_H	read/write	LED12 output and brightness control byte 1
56	38	0	0	1	1	1	0	0	0	LED12_OFF_L	read/write	LED12 output and brightness control byte 2
57	39	0	0	1	1	1	0	0	1	LED12_OFF_H	read/write	LED12 output and brightness control byte 3
58	3A	0	0	1	1	1	0	1	0	LED13_ON_L	read/write	LED13 output and brightness control byte 0
59	3B	0	0	1	1	1	0	1	1	LED13_ON_H	read/write	LED13 output and brightness control byte 1
60	3C	0	0	1	1	1	1	0	0	LED13_OFF_L	read/write	LED13 output and brightness control byte 2
61	3D	0	0	1	1	1	1	0	1	LED13_OFF_H	read/write	LED13 output and brightness control byte 3

Table 3. Register summary *...continued*

Register # (decimal)	Register # (hex)	D7	D6	D5	D4	D3	D2	D1	D0	Name	Type	Function
62	3E	0	0	1	1	1	1	1	0	LED14_ON_L	read/write	LED14 output and brightness control byte 0
63	3F	0	0	1	1	1	1	1	1	LED14_ON_H	read/write	LED14 output and brightness control byte 1
64	40	0	1	0	0	0	0	0	0	LED14_OFF_L	read/write	LED14 output and brightness control byte 2
65	41	0	1	0	0	0	0	0	1	LED14_OFF_H	read/write	LED14 output and brightness control byte 3
66	42	0	1	0	0	0	0	1	0	LED15_ON_L	read/write	LED15 output and brightness control byte 0
67	43	0	1	0	0	0	0	1	1	LED15_ON_H	read/write	LED15 output and brightness control byte 1
68	44	0	1	0	0	0	1	0	0	LED15_OFF_L	read/write	LED15 output and brightness control byte 2
69	45	0	1	0	0	0	1	0	1	LED15_OFF_H	read/write	LED15 output and brightness control byte 3
...	reserved for future use											
250	FA	1	1	1	1	1	0	1	0	ALL_LED_ON_L	write/read zero	load all the LEDn_ON registers, byte 0
251	FB	1	1	1	1	1	0	1	1	ALL_LED_ON_H	write/read zero	load all the LEDn_ON registers, byte 1
252	FC	1	1	1	1	1	1	0	0	ALL_LED_OFF_L	write/read zero	load all the LEDn_OFF registers, byte 0
253	FD	1	1	1	1	1	1	0	1	ALL_LED_OFF_H	write/read zero	load all the LEDn_OFF registers, byte 1
254	FE	1	1	1	1	1	1	1	0	PRE_SCALE[1]	read/write	prescaler for output frequency
255	FF	1	1	1	1	1	1	1	1	TestMode[2]	read/write	defines the test mode to be entered
...	All further addresses are reserved for future use; reserved addresses will not be acknowledged.											

[1] Writes to PRE_SCALE register are blocked when SLEEP bit is logic 0 (MODE 1).

[2] Reserved. Writes to this register may cause unpredictable results.

Remark: Auto Increment past register 69 will point to MODE1 register (register 0). Auto Increment also works from register 250 to register 254, then rolls over to register 0.

7.3.1 Mode register 1, MODE1

Table 4. MODE1 - Mode register 1 (address 00h) bit description

*Legend: * default value.*

Bit	Symbol	Access	Value	Description
7	RESTART	R		Shows state of RESTART logic. See Section 7.3.1.1 for detail.
		W		User writes logic 1 to this bit to clear it to logic 0. A user write of logic 0 will have no effect. See Section 7.3.1.1 for detail.
			0*	Restart disabled.
			1	Restart enabled.
6	EXTCLK	R/W		To use the EXTCLK pin, this bit must be set by the following sequence: 1. Set the SLEEP bit in MODE1. This turns off the internal oscillator. 2. Write logic 1s to both the SLEEP and EXTCLK bits in MODE1. The switch is now made. The external clock can be active during the switch because the SLEEP bit is set. This bit is a 'sticky bit', that is, it cannot be cleared by writing a logic 0 to it. The EXTCLK bit can **only** be cleared by a power cycle or software reset. EXTCLK range is DC to 50 MHz. $refresh\_rate = \frac{EXTCLK}{4096 \times (prescale + 1)}$
			0*	Use internal clock.
			1	Use EXTCLK pin clock.
5	AI	R/W	0*	Register Auto-Increment disabled[1].
			1	Register Auto-Increment enabled.
4	SLEEP	R/W	0	Normal mode[2].
			1*	Low power mode. Oscillator off[3][4].
3	SUB1	R/W	0*	PCA9685 does not respond to I²C-bus subaddress 1.
			1	PCA9685 responds to I²C-bus subaddress 1.
2	SUB2	R/W	0*	PCA9685 does not respond to I²C-bus subaddress 2.
			1	PCA9685 responds to I²C-bus subaddress 2.
1	SUB3	R/W	0*	PCA9685 does not respond to I²C-bus subaddress 3.
			1	PCA9685 responds to I²C-bus subaddress 3.
0	ALLCALL	R/W	0	PCA9685 does not respond to LED All Call I²C-bus address.
			1*	PCA9685 responds to LED All Call I²C-bus address.

[1] When the Auto Increment flag is set, AI = 1, the Control register is automatically incremented after a read or write. This allows the user to program the registers sequentially.

[2] It takes 500 μs max. for the oscillator to be up and running once SLEEP bit has been set to logic 0. Timings on LEDn outputs are not guaranteed if PWM control registers are accessed within the 500 μs window. There is no start-up delay required when using the EXTCLK pin as the PWM clock.

[3] No PWM control is possible when the oscillator is off.

[4] When the oscillator is off (Sleep mode) the LEDn outputs cannot be turned on, off or dimmed/blinked.

7.3.1.1 Restart mode

If the PCA9685 is operating and the user decides to put the chip to sleep (setting MODE1 bit 4) without stopping any of the PWM channels, the RESTART bit (MODE1 bit 7) will be set to logic 1 at the end of the PWM refresh cycle. The contents of each PWM register are held valid when the clock is off.

To restart all of the previously active PWM channels with a few I²C-bus cycles do the following steps:

1. Read MODE1 register.
2. Check that bit 7 (RESTART) is a logic 1. If it is, clear bit 4 (SLEEP). Allow time for oscillator to stabilize (500 μs).
3. Write logic 1 to bit 7 of MODE1 register. All PWM channels will restart and the RESTART bit will clear.

Remark: The SLEEP bit **must** be logic 0 for at least 500 μs, before a logic 1 is written into the RESTART bit.

Other actions that will clear the RESTART bit are:

1. Power cycle.
2. I²C Software Reset command.
3. If the MODE2 OCH bit is logic 0, write to any PWM register then issue an I²C-bus STOP.
4. If the MODE2 OCH bit is logic 1, write to all four PWM registers in any PWM channel.

Likewise, if the user does an orderly shutdown[1] of all the PWM channels before setting the SLEEP bit, the RESTART bit will be cleared. If this is done the contents of all PWM registers are invalidated and must be reloaded before reuse.

An example of the use of the RESTART bit would be the restoring of a customer's laptop LCD backlight intensity coming out of Standby to the level it was before going into Standby.

1. Two methods can be used to do an orderly shutdown. The fastest is to write a logic 1 to bit 4 in register ALL_LED_OFF_H. The other method is to write logic 1 to bit 4 in each active PWM channel LEDn_OFF_H register.

7.3.2 Mode register 2, MODE2

Table 5. MODE2 - Mode register 2 (address 01h) bit description

*Legend: * default value.*

Bit	Symbol	Access	Value	Description
7 to 5	-	read only	000*	reserved
4	INVRT[1]	R/W	0*	Output logic state not inverted. Value to use when external driver used. Applicable when $\overline{OE}$ = 0.
			1	Output logic state inverted. Value to use when no external driver used. Applicable when $\overline{OE}$ = 0.
3	OCH	R/W	0*	Outputs change on STOP command[2].
			1	Outputs change on ACK[3].
2	OUTDRV[1]	R/W	0	The 16 LEDn outputs are configured with an open-drain structure.
			1*	The 16 LEDn outputs are configured with a totem pole structure.
1 to 0	OUTNE[1:0][4]	R/W	00*	When $\overline{OE}$ = 1 (output drivers not enabled), LEDn = 0.
			01	When $\overline{OE}$ = 1 (output drivers not enabled): LEDn = 1 when OUTDRV = 1 LEDn = high-impedance when OUTDRV = 0 (same as OUTNE[1:0] = 10)
			1X	When $\overline{OE}$ = 1 (output drivers not enabled), LEDn = high-impedance.

[1] See Section 7.7 “Using the PCA9685 with and without external drivers” for more details. Normal LEDs can be driven directly in either mode. Some newer LEDs include integrated Zener diodes to limit voltage transients, reduce EMI, protect the LEDs and these must be driven only in the open-drain mode to prevent overheating the IC.

[2] Change of the outputs at the STOP command allows synchronizing outputs of more than one PCA9685. Applicable to registers from 06h (LED0_ON_L) to 45h (LED15_OFF_H) only. 1 or more registers can be written, in any order, before STOP.

[3] Update on ACK requires **all** 4 PWM channel registers to be loaded before outputs will change on the **last** ACK.

[4] See Section 7.4 “Active LOW output enable input” for more details.

7.3.3 LED output and PWM control

The turn-on time of each LED driver output and the duty cycle of PWM can be controlled independently using the LEDn_ON and LEDn_OFF registers.

There will be two 12-bit registers per LED output. These registers will be programmed by the user. Both registers will hold a value from 0 to 4095. One 12-bit register will hold a value for the ON time and the other 12-bit register will hold the value for the OFF time. The ON and OFF times are compared with the value of a 12-bit counter that will be running continuously from 0000h to 0FFFh (0 to 4095 decimal).

Update on ACK requires all 4 PWM channel registers to be loaded before outputs will change on the last ACK.

The ON time, which is programmable, will be the time the LED output will be asserted and the OFF time, which is also programmable, will be the time when the LED output will be negated. In this way, the phase shift becomes completely programmable. The resolution for the phase shift is $^{1}/_{4096}$ of the target frequency. Table 6 lists these registers.

The following two examples illustrate how to calculate values to be loaded into these registers.

Example 1: (assumes that the LED0 output is used and (delay time) + (PWM duty cycle) ≤ 100 %)

Delay time = 10 %; PWM duty cycle = 20 % (LED on time = 20 %; LED off time = 80 %).

Delay time = 10 % = 409.6 ~ 410 counts = 19Ah.

Since the counter starts at 0 and ends at 4095, we will subtract 1, so delay time = 199h counts.

LED0_ON_H = 1h; LED0_ON_L = 99h

LED on time = 20 % = 819.2 ~ 819 counts.

Off time = 4CCh (decimal 410 + 819 – 1 = 1228)

LED0_OFF_H = 4h; LED0_OFF_L = CCh

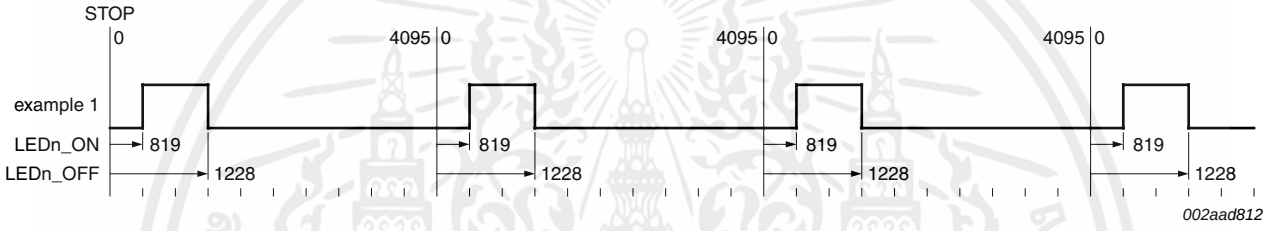


Fig 7. LED output, example 1

Example 2: (assumes that the LED4 output is used and (delay time) + (PWM duty cycle > 100 %)

Delay time = 90 %; PWM duty cycle = 90 % (LED on time = 90 %; LED off time = 10 %).

Delay time = 90 % = 3686.4 ~ 3686 counts – 1 = 3685 = E65h.

LED4_ON_H = Eh; LED4_ON_L = 65h

LED on time = 90 % = 3686 counts.

Since the delay time and LED on period of the duty cycle is greater than 4096 counts, the LEDn_OFF count will occur in the next frame. Therefore, 4096 is subtracted from the LEDn_OFF count to get the correct LEDn_OFF count. See Figure 9, Figure 10 and Figure 11.

Off time = 4CBh (decimal 3685 + 3686 = 7372 – 4096 = 3275)

LED4_OFF_H = 4h; LED4_OFF_L = CBh

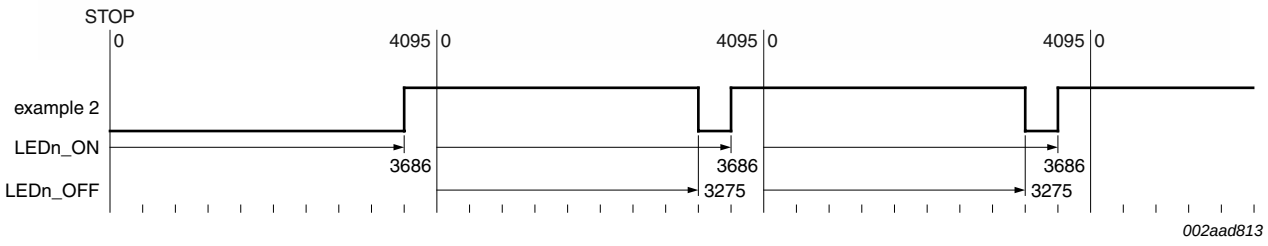
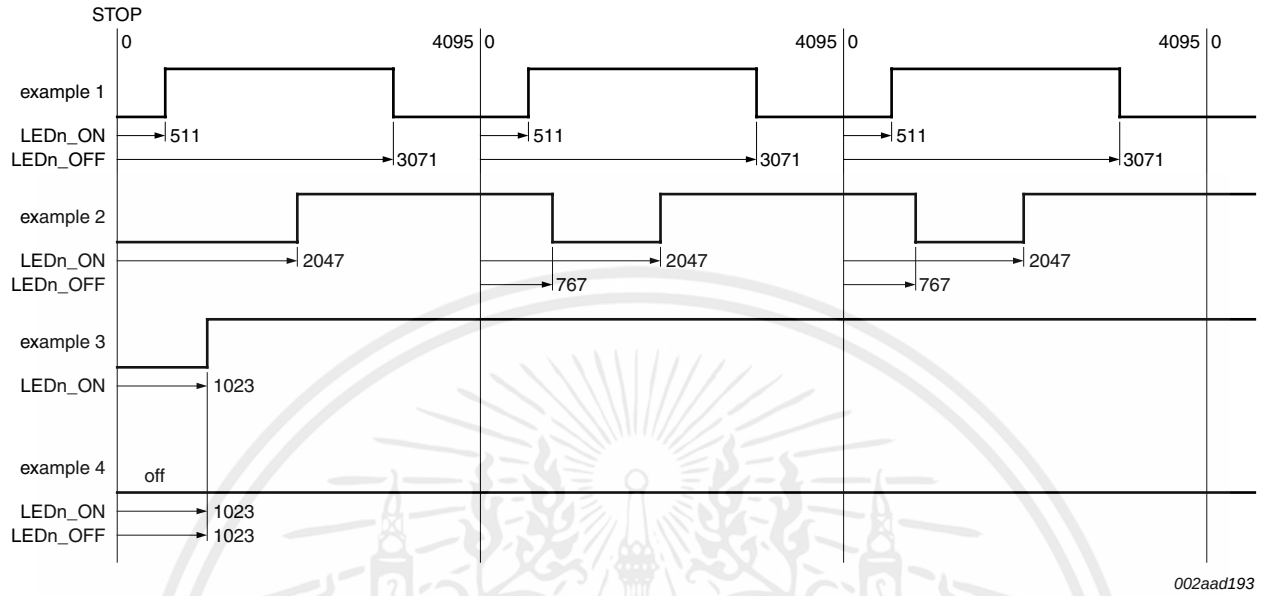


Fig 8. LED output, example 2



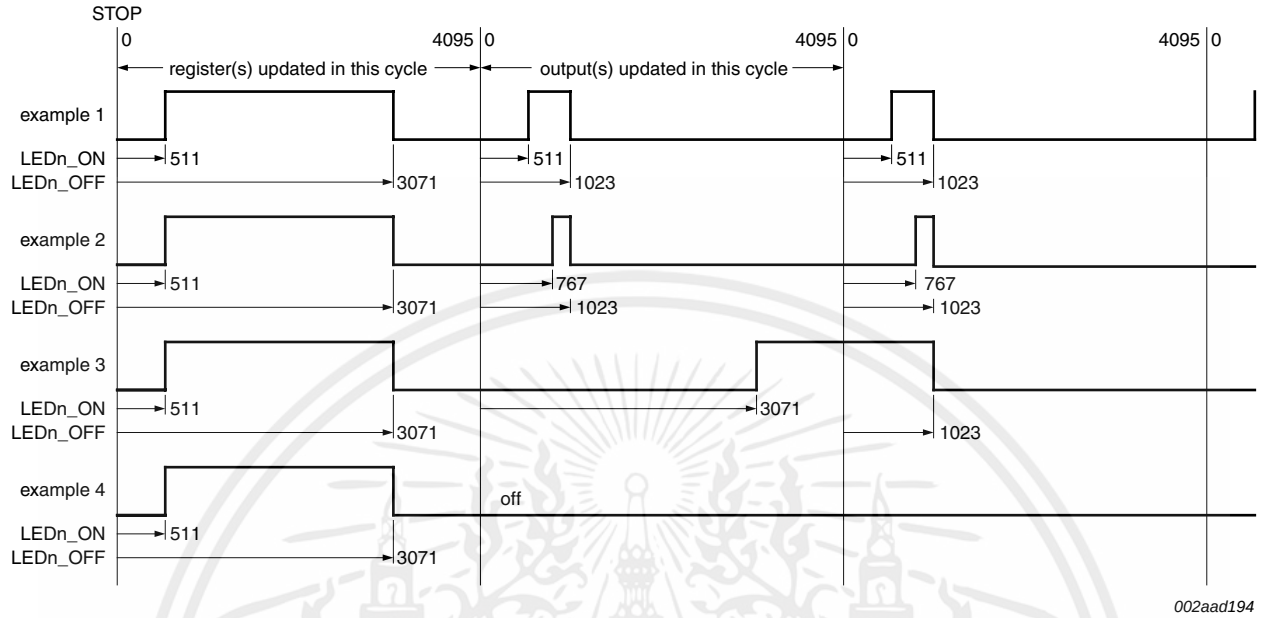
Example 1: LEDn_ON < LEDn_OFF

Example 2: LEDn_ON > LEDn_OFF

Example 3: LEDn_ON[12] = 1; LEDn_ON[11:0] = 1022; LEDn_OFF[12] = 0; LEDn_OFF[11:0] = don't care

Example 4: LEDn_ON[12] = 0; LEDn_OFF[12] = 0; LEDn_ON[11:0] = LEDn_OFF[11:0]

Fig 9. Output example



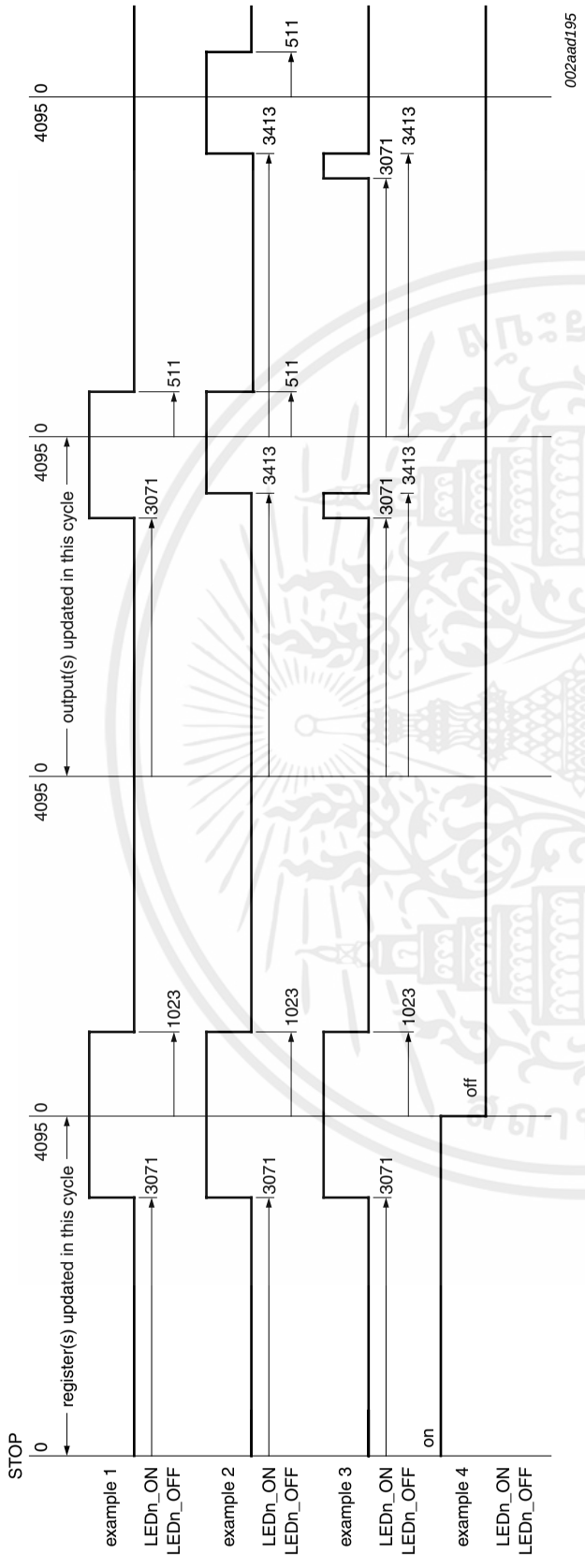
Example 1: LEDn_ON unchanged and LEDn_OFF decreased.

Example 2: LEDn_ON increased and LEDn_OFF decreased.

Example 3: LEDn_ON made > LEDn_OFF.

Example 4: LEDn_OFF[12] set to 1.

Fig 10. Update examples when LEDn_ON < LEDn_OFF



Example 1: LEDn_ON unchanged and LEDn_OFF decreased, but delay still > LEDn_OFF

Example 2: LEDn_ON changed and LEDn_OFF changed, but delay still > LEDn_OFF

Example 3: LEDn_ON unchanged and LEDn_OFF increased where LEDn_ON < LEDn_OFF

Example 4: LEDn_ON[12] = 1 and LEDn_OFF[12] changed from 0 to 1

Fig 11. Update examples when LEDn_ON > LEDn_OFF

Table 6. LED_ON, LED_OFF control registers (address 06h to 45h) bit description

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
06h	LED0_ON_L	7:0	LED0_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED0, 8 LSBs
07h	LED0_ON_H	7:5	reserved	R	000*	non-writable
		4	LED0_ON_H[4]	R/W	0*	LED0 full ON
		3:0	LED0_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED0, 4 MSBs
08h	LED0_OFF_L	7:0	LED0_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED0, 8 LSBs
09h	LED0_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED0_OFF_H[4]	R/W	1*	LED0 full OFF
		3:0	LED0_OFF_H[3:0]	R/W	0000*	
0Ah	LED1_ON_L	7:0	LED1_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED1, 8 LSBs
0Bh	LED1_ON_H	7:5	reserved	R	000*	non-writable
		4	LED1_ON_H[4]	R/W	0*	LED1 full ON
		3:0	LED1_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED1, 4 MSBs
0Ch	LED1_OFF_L	7:0	LED1_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED1, 8 LSBs
0Dh	LED1_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED1_OFF_H[4]	R/W	1*	LED1 full OFF
		3:0	LED1_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED1, 4 MSBs
0Eh	LED2_ON_L	7:0	LED2_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED2, 8 LSBs
0Fh	LED2_ON_H	7:5	reserved	R	000*	non-writable
		4	LED2_ON_H[4]	R/W	0*	LED2 full ON
		3:0	LED2_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED2, 4 MSBs
10h	LED2_OFF_L	7:0	LED2_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED2, 8 LSBs
11h	LED2_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED2_OFF_H[4]	R/W	1*	LED2 full OFF
		3:0	LED2_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED2, 4 MSBs
12h	LED3_ON_L	7:0	LED3_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED3, 8 LSBs
13h	LED3_ON_H	7:5	reserved	R	000*	non-writable
		4	LED3_ON_H[4]	R/W	0*	LED3 full ON
		3:0	LED3_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED3, 4 MSBs
14h	LED3_OFF_L	7:0	LED3_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED3, 8 LSBs
15h	LED3_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED3_OFF_H[4]	R/W	1*	LED3 full OFF
		3:0	LED3_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED3, 4 MSBs
16h	LED4_ON_L	7:0	LED4_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED4, 8 LSBs
17h	LED4_ON_H	7:5	reserved	R	000*	non-writable
		4	LED4_ON_H[4]	R/W	0*	LED4 full ON
		3:0	LED4_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED4, 4 MSBs

Table 6. LED_ON, LED_OFF control registers (address 06h to 45h) bit description *…continued*

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
18h	LED4_OFF_L	7:0	LED4_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED4, 8 LSBs
19h	LED4_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED4_OFF_H[4]	R/W	1*	LED4 full OFF
		3:0	LED4_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED4, 4 MSBs
1Ah	LED5_ON_L	7:0	LED5_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED5, 8 LSBs
1Bh	LED5_ON_H	7:5	reserved	R	000*	non-writable
		4	LED5_ON_H[4]	R/W	0*	LED5 full ON
		3:0	LED5_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED5, 4 MSBs
1Ch	LED5_OFF_L	7:0	LED5_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED5, 8 LSBs
1Dh	LED5_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED5_OFF_H[4]	R/W	1*	LED5 full OFF
		3:0	LED5_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED5, 4 MSBs
1Eh	LED6_ON_L	7:0	LED6_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED6, 8 LSBs
1Fh	LED6_ON_H	7:5	reserved	R	000*	non-writable
		4	LED6_ON_H[4]	R/W	0*	LED6 full ON
		3:0	LED6_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED6, 4 MSBs
20h	LED6_OFF_L	7:0	LED6_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED6, 8 LSBs
21h	LED6_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED6_OFF_H[4]	R/W	1*	LED6 full OFF
		3:0	LED6_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED6, 4 MSBs
22h	LED7_ON_L	7:0	LED7_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED7, 8 LSBs
23h	LED7_ON_H	7:5	reserved	R	000*	non-writable
		4	LED7_ON_H[4]	R/W	0*	LED7 full ON
		3:0	LED7_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED7, 4 MSBs
24h	LED7_OFF_L	7:0	LED7_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED7, 8 LSBs
25h	LED7_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED7_OFF_H[4]	R/W	1*	LED7 full OFF
		3:0	LED7_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED7, 4 MSBs
26h	LED8_ON_L	7:0	LED8_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED8, 8 LSBs
27h	LED8_ON_H	7:5	reserved	R	000*	non-writable
		4	LED8_ON_H[4]	R/W	0*	LED8 full ON
		3:0	LED8_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED8, 4 MSBs
28h	LED8_OFF_L	7:0	LED8_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED8, 8 LSBs
29h	LED8_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED8_OFF_H[4]	R/W	1*	LED8 full OFF
		3:0	LED8_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED8, 4 MSBs

Table 6. LED_ON, LED_OFF control registers (address 06h to 45h) bit description *…continued*

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
2Ah	LED9_ON_L	7:0	LED9_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED9, 8 LSBs
2Bh	LED9_ON_H	7:5	reserved	R	000*	non-writable
		4	LED9_ON_H[4]	R/W	0*	LED9 full ON
		3:0	LED9_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED9, 4 MSBs
2Ch	LED9_OFF_L	7:0	LED9_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED9, 8 LSBs
2Dh	LED9_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED9_OFF_H[4]	R/W	1*	LED9 full OFF
		3:0	LED9_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED9, 4 MSBs
2Eh	LED10_ON_L	7:0	LED10_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED10, 8 LSBs
2Fh	LED10_ON_H	7:5	reserved	R	000*	non-writable
		4	LED10_ON_H[4]	R/W	0*	LED10 full ON
		3:0	LED10_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED10, 4 MSBs
30h	LED10_OFF_L	7:0	LED10_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED10, 8 LSBs
31h	LED10_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED10_OFF_H[4]	R/W	1*	LED10 full OFF
		3:0	LED10_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED10, 4 MSBs
32h	LED11_ON_L	7:0	LED11_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED11, 8 LSBs
33h	LED11_ON_H	7:5	reserved	R	000*	non-writable
		4	LED11_ON_H[4]	R/W	0*	LED11 full ON
		3:0	LED11_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED11, 4 MSBs
34h	LED11_OFF_L	7:0	LED11_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED11, 8 LSBs
35h	LED11_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED11_OFF_H[4]	R/W	1*	LED11 full OFF
		3:0	LED11_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED11, 4 MSBs
36h	LED12_ON_L	7:0	LED12_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED12, 8 LSBs
37h	LED12_ON_H	7:5	reserved	R	000*	non-writable
		4	LED12_ON_H[4]	R/W	0*	LED12 full ON
		3:0	LED12_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED12, 4 MSBs
38h	LED12_OFF_L	7:0	LED12_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED12, 8 LSBs
39h	LED12_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED12_OFF_H[4]	R/W	1*	LED12 full OFF
		3:0	LED12_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED12, 4 MSBs
3Ah	LED13_ON_L	7:0	LED13_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED13, 8 LSBs
3Bh	LED13_ON_H	7:5	reserved	R	000*	non-writable
		4	LED13_ON_H[4]	R/W	0*	LED13 full ON
		3:0	LED13_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED13, 4 MSBs

Table 6. LED_ON, LED_OFF control registers (address 06h to 45h) bit description *…continued*

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
3Ch	LED13_OFF_L	7:0	LED13_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED13, 8 LSBs
3Dh	LED13_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED13_OFF_H[4]	R/W	1*	LED13 full OFF
		3:0	LED13_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED13, 4 MSBs
3Eh	LED14_ON_L	7:0	LED14_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED14, 8 LSBs
3Fh	LED14_ON_H	7:5	reserved	R	000*	non-writable
		4	LED14_ON_H[4]	R/W	0*	LED14 full ON
		3:0	LED14_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED14, 4 MSBs
40h	LED14_OFF_L	7:0	LED14_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED14, 8 LSBs
41h	LED14_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED14_OFF_H[4]	R/W	1*	LED14 full OFF
		3:0	LED14_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED14, 4 MSBs
42h	LED15_ON_L	7:0	LED15_ON_L[7:0]	R/W	0000 0000*	LEDn_ON count for LED15, 8 LSBs
43h	LED15_ON_H	7:5	reserved	R	000*	non-writable
		4	LED15_ON_H[4]	R/W	0*	LED15 full ON
		3:0	LED15_ON_H[3:0]	R/W	0000*	LEDn_ON count for LED15, 4 MSBs
44h	LED15_OFF_L	7:0	LED15_OFF_L[7:0]	R/W	0000 0000*	LEDn_OFF count for LED15, 8 LSBs
45h	LED15_OFF_H	7:5	reserved	R	000*	non-writable
		4	LED15_OFF_H[4]	R/W	1*	LED15 full OFF
		3:0	LED15_OFF_H[3:0]	R/W	0000*	LEDn_OFF count for LED15, 4 MSBs

The LEDn_ON_H output control bit 4, when set to logic 1, causes the output to be always ON. The turning ON of the LED is delayed by the amount in the LEDn_ON registers. LEDn_OFF[11:0] are ignored. When this bit = 0, then the LEDn_ON and LEDn_OFF registers are used according to their normal definition.

The LEDn_OFF_H output control bit 4, when set to logic 1, causes the output to be always OFF. In this case the values in the LEDn_ON registers are ignored.

Remark: When all LED outputs are configured as ‘always OFF’, the prescale counter and all associated PWM cycle timing logic are disabled. If LEDn_ON_H[4] and LEDn_OFF_H[4] are set at the same time, the LEDn_OFF_H[4] function takes precedence.

7.3.4 ALL_LED_ON and ALL_LED_OFF control

The ALL_LED_ON and ALL_LED_OFF registers allow just four I²C-bus write sequences to fill all the ON and OFF registers with the same patterns.

Table 7. ALL_LED_ON and ALL_LED_OFF control registers (address FAh to FEh) bit description

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
FAh	ALL_LED_ON_L	7:0	ALL_LED_ON_L[7:0]	W only	0000 0000*	LEDn_ON count for ALL_LED, 8 MSBs
FBh	ALL_LED_ON_H	7:5	reserved	R	000*	non-writable
		4	ALL_LED_ON_H[4]	W only	1*	ALL_LED full ON
		3:0	ALL_LED_ON_H[3:0]	W only	0000*	LEDn_ON count for ALL_LED, 4 MSBs
FCh	ALL_LED_OFF_L	7:0	ALL_LED_OFF_L[7:0]	W only	0000 0000*	LEDn_OFF count for ALL_LED, 8 MSBs
FDh	ALL_LED_OFF_H	7:5	reserved	R	000*	non-writable
		4	ALL_LED_OFF_H[4]	W only	1*	ALL_LED full OFF
		3:0	ALL_LED_OFF_H[3:0]	W only	0000*	LEDn_OFF count for ALL_LED, 4 MSBs
FEh	PRE_SCALE	7:0	PRE_SCALE[7:0]	R/W	0001 1110*	prescaler to program the output frequency

The LEDn_ON and LEDn_OFF counts can vary from 0 to 4095. The LEDn_ON and LEDn_OFF count registers should never be programmed with the same values.

Because the loading of the LEDn_ON and LEDn_OFF registers is via the I²C-bus, and asynchronous to the internal oscillator, we want to ensure that we do not see any visual artifacts of changing the ON and OFF values. This is achieved by updating the changes at the end of the LOW cycle.

7.3.5 PWM frequency PRE_SCALE

The hardware forces a minimum value that can be loaded into the PRE_SCALE register at '3'. The PRE_SCALE register defines the frequency at which the outputs modulate. The prescale value is determined with the formula shown in Equation 1:

$$prescale\ value = round\left(\frac{osc\_clock}{4096 \times update\_rate}\right) - 1 \quad (1)$$

where the update rate is the output modulation frequency required. For example, for an output frequency of 200 Hz with an oscillator clock frequency of 25 MHz:

$$prescale\ value = round\left(\frac{25\ MHz}{4096 \times 200}\right) - 1 = 30 \quad (2)$$

The PRE_SCALE register can only be set when the SLEEP bit of MODE1 register is set to logic 1.

7.3.6 SUBADR1 to SUBADR3, I²C-bus subaddress 1 to 3

Table 8. SUBADR1 to SUBADR3 - I²C-bus subaddress registers 0 to 3 (address 02h to 04h) bit description

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
02h	SUBADR1	7:1	A1[7:1]	R/W	1110 001*	I²C-bus subaddress 1
		0	A1[0]	R only	0*	reserved
03h	SUBADR2	7:1	A2[7:1]	R/W	1110 010*	I²C-bus subaddress 2
		0	A2[0]	R only	0*	reserved
04h	SUBADR3	7:1	A3[7:1]	R/W	1110 100*	I²C-bus subaddress 3
		0	A3[0]	R only	0*	reserved

Subaddresses are programmable through the I²C-bus. Default power-up values are E2h, E4h, E8h, and the device(s) will not acknowledge these addresses right after power-up (the corresponding SUBx bit in MODE1 register is equal to 0).

Once subaddresses have been programmed to their right values, SUBx bits need to be set to logic 1 in order to have the device acknowledging these addresses (MODE1 register).

Only the 7 MSBs representing the I²C-bus subaddress are valid. The LSB in SUBADRx register is a read-only bit (0).

When SUBx is set to logic 1, the corresponding I²C-bus subaddress can be used during either an I²C-bus read or write sequence.

7.3.7 ALLCALLADR, LED All Call I²C-bus address

Table 9. ALLCALLADR - LED All Call I²C-bus address register (address 05h) bit description

*Legend: * default value.*

Address	Register	Bit	Symbol	Access	Value	Description
05h	ALLCALLADR	7:1	AC[7:1]	R/W	1110 000*	ALLCALL I²C-bus address register
		0	AC[0]	R only	0*	reserved

The LED All Call I²C-bus address allows all the PCA9685s in the bus to be programmed at the same time (ALLCALL bit in register MODE1 must be equal to 1 (power-up default state)). This address is programmable through the I²C-bus and can be used during either an I²C-bus read or write sequence. The register address can also be programmed as a Sub Call.

Only the 7 MSBs representing the All Call I²C-bus address are valid. The LSB in ALLCALLADR register is a read-only bit (0).

If ALLCALL bit = 0, the device does not acknowledge the address programmed in register ALLCALLADR.

7.4 Active LOW output enable input

The active LOW output enable ($\overline{OE}$) pin, allows to enable or disable all the LED outputs at the same time.

- When a LOW level is applied to $\overline{OE}$ pin, all the LED outputs are enabled and follow the output state defined in the LEDn_ON and LEDn_OFF registers with the polarity defined by INVRT bit (MODE2 register).
- When a HIGH level is applied to $\overline{OE}$ pin, all the LED outputs are programmed to the value that is defined by OUTNE[1:0] in the MODE2 register.

Table 10. LED outputs when $\overline{OE}$ = 1

OUTNE1	OUTNE0	LED outputs
0	0	0
0	1	1 if OUTDRV = 1, high-impedance if OUTDRV = 0
1	0	high-impedance
1	1	high-impedance

The $\overline{OE}$ pin can be used as a synchronization signal to switch on/off several PCA9685 devices at the same time. This requires an external clock reference that provides blinking period and the duty cycle.

The $\overline{OE}$ pin can also be used as an external dimming control signal. The frequency of the external clock must be high enough not to be seen by the human eye, and the duty cycle value determines the brightness of the LEDs.

7.5 Power-on reset

When power is applied to V_{DD}, an internal power-on reset holds the PCA9685 in a reset condition until V_{DD} has reached V_{POR}. At this point, the reset condition is released and the PCA9685 registers and I²C-bus state machine are initialized to their default states. Thereafter, V_{DD} must be lowered below 0.2 V to reset the device.

7.6 Software reset

The Software Reset Call (SWRST Call) allows all the devices in the I²C-bus to be reset to the power-up state value through a specific formatted I²C-bus command. To be performed correctly, it implies that the I²C-bus is functional and that there is no device hanging the bus.

The SWRST Call function is defined as the following:

1. A START command is sent by the I²C-bus master.
2. The reserved SWRST I²C-bus address '0000 000' with the R/$\overline{W}$ bit set to '0' (write) is sent by the I²C-bus master.
3. The PCA9685 device(s) acknowledge(s) after seeing the General Call address '0000 0000' (00h) only. If the R/$\overline{W}$ bit is set to '1' (read), no acknowledge is returned to the I²C-bus master.
4. Once the General Call address has been sent and acknowledged, the master sends 1 byte with 1 specific value (SWRST data byte 1):
   a. Byte 1 = 06h: the PCA9685 acknowledges this value only. If byte 1 is not equal to 06h, the PCA9685 does not acknowledge it.

   If more than 1 byte of data is sent, the PCA9685 does not acknowledge any more.
5. Once the correct byte (SWRST data byte 1) has been sent and correctly acknowledged, the master sends a STOP command to end the SWRST Call: the PCA9685 then resets to the default value (power-up value) and is ready to be addressed again within the specified bus free time (t_{BUF}).

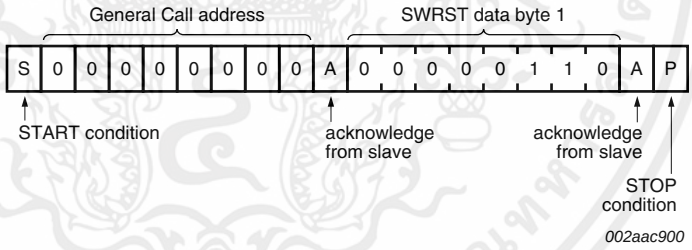


Fig 12. SWRST Call

The I²C-bus master must interpret a non-acknowledge from the PCA9685 (at any time) as a 'SWRST Call Abort'. The PCA9685 does not initiate a reset of its registers. This happens only when the format of the SWRST Call sequence is not correct.

7.7 Using the PCA9685 with and without external drivers

The PCA9685 LED output drivers are 5.5 V only tolerant and can sink up to 25 mA at 5 V.

If the device needs to drive LEDs to a higher voltage and/or higher current, use of an external driver is required.

- INVRT bit (MODE2 register) can be used to keep the LED PWM control firmware the same independently of the type of external driver. This bit allows LED output polarity inversion/non-inversion only when $\overline{OE}$ = 0.
- OUTDRV bit (MODE2 register) allows minimizing the amount of external components required to control the external driver (N-type or P-type device).

Table 11. Use of INVRT and OUTDRV based on connection to the LEDn outputs when $\overline{OE}$ = 0[1]

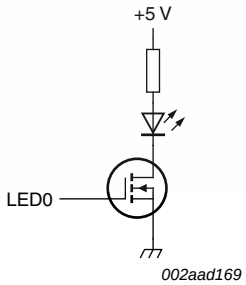
INVRT	OUTDRV	Direct connection to LEDn		External N-type driver		External P-type driver	
		Firmware	External pull-up resistor	Firmware	External pull-up resistor	Firmware	External pull-up resistor
0	0	formulas and LED output state values inverted	LED current limiting R[2]	formulas and LED output state values inverted	required	formulas and LED output state values apply	required
0	1	formulas and LED output state values inverted	LED current limiting R[2]	formulas and LED output state values apply[3]	not required[3]	formulas and LED output state values inverted	not required
1	0	formulas and LED output state values apply[2]	LED current limiting R	formulas and LED output state values apply	required	formulas and LED output state values inverted	required
1	1	formulas and LED output state values apply[2]	LED current limiting R	formulas and LED output state values inverted	not required	formulas and LED output state values apply[4]	not required[4]

[1] When $\overline{OE}$ = 1, LED output state is controlled only by OUTNE[1:0] bits (MODE2 register).

[2] Correct configuration when LEDs directly connected to the LEDn outputs (connection to V_{DD} through current limiting resistor).

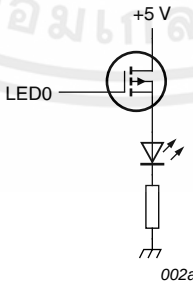
[3] Optimum configuration when external N-type (NPN, NMOS) driver used.

[4] Optimum configuration when external P-type (PNP, PMOS) driver used.



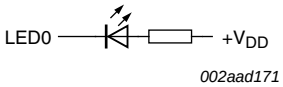
INVRT = 0
OUTDRV = 1

Fig 13. External N-type driver



INVRT = 1
OUTDRV = 1

Fig 14. External P-type driver



INVRT = 1
OUTDRV = 0

Fig 15. Direct LED connection

8. Characteristics of the I²C-bus

The I²C-bus is for 2-way, 2-line communication between different ICs or modules. The two lines are a serial data line (SDA) and a serial clock line (SCL). Both lines must be connected to a positive supply via a pull-up resistor when connected to the output stages of a device. Data transfer may be initiated only when the bus is not busy.

8.1 Bit transfer

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse as changes in the data line at this time will be interpreted as control signals (see Figure 16).

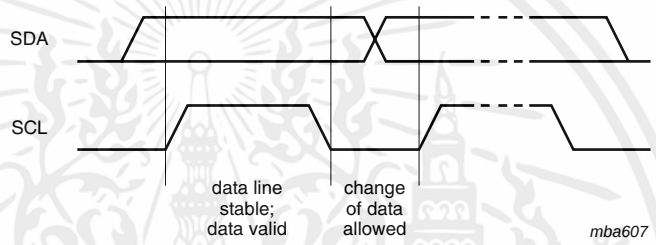


Fig 16. Bit transfer

8.1.1 START and STOP conditions

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line while the clock is HIGH is defined as the START condition (S). A LOW-to-HIGH transition of the data line while the clock is HIGH is defined as the STOP condition (P) (see Figure 17).

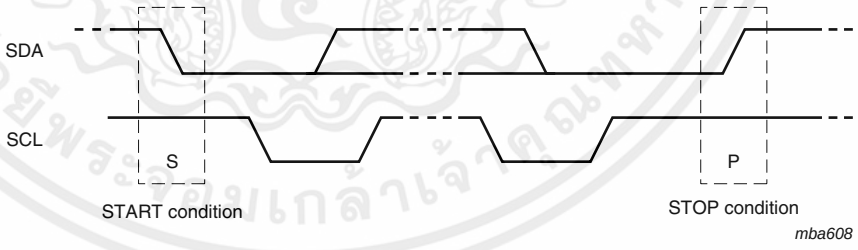


Fig 17. Definition of START and STOP conditions

8.2 System configuration

A device generating a message is a ‘transmitter’; a device receiving is the ‘receiver’. The device that controls the message is the ‘master’ and the devices which are controlled by the master are the ‘slaves’ (see Figure 18).

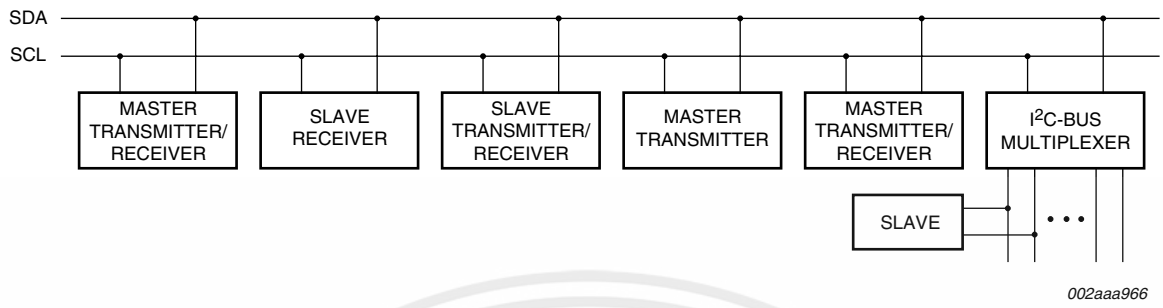


Fig 18. System configuration

8.3 Acknowledge

The number of data bytes transferred between the START and the STOP conditions from transmitter to receiver is not limited. Each byte of eight bits is followed by one acknowledge bit. The acknowledge bit is a HIGH level put on the bus by the transmitter, whereas the master generates an extra acknowledge related clock pulse.

A slave receiver which is addressed must generate an acknowledge after the reception of each byte. Also a master must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, so that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse; set-up time and hold time must be taken into account.

A master receiver must signal an end of data to the transmitter by not generating an acknowledge on the last byte that has been clocked out of the slave. In this event, the transmitter must leave the data line HIGH to enable the master to generate a STOP condition.

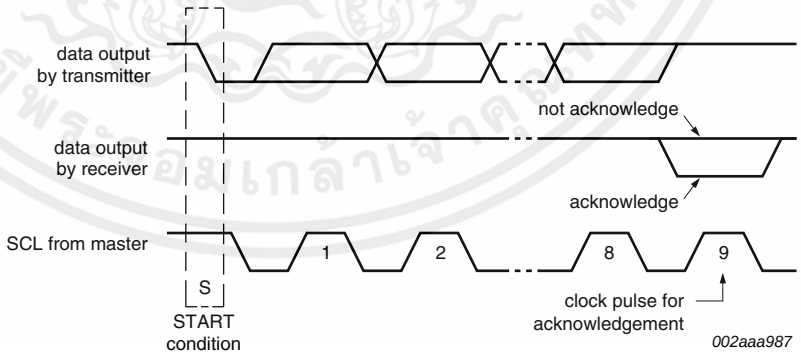
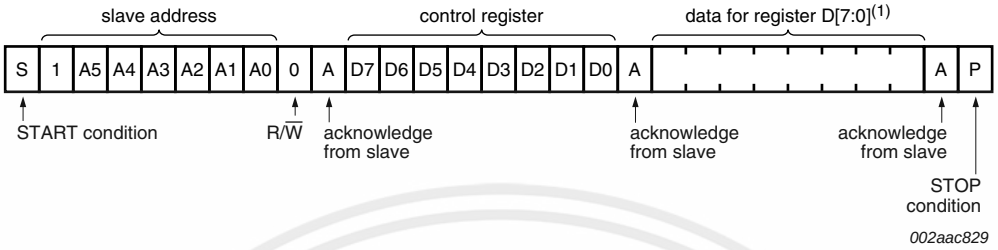


Fig 19. Acknowledgement on the I²C-bus

9. Bus transactions



(1) See Table 3 for register definition.

Fig 20. Write to a specific register

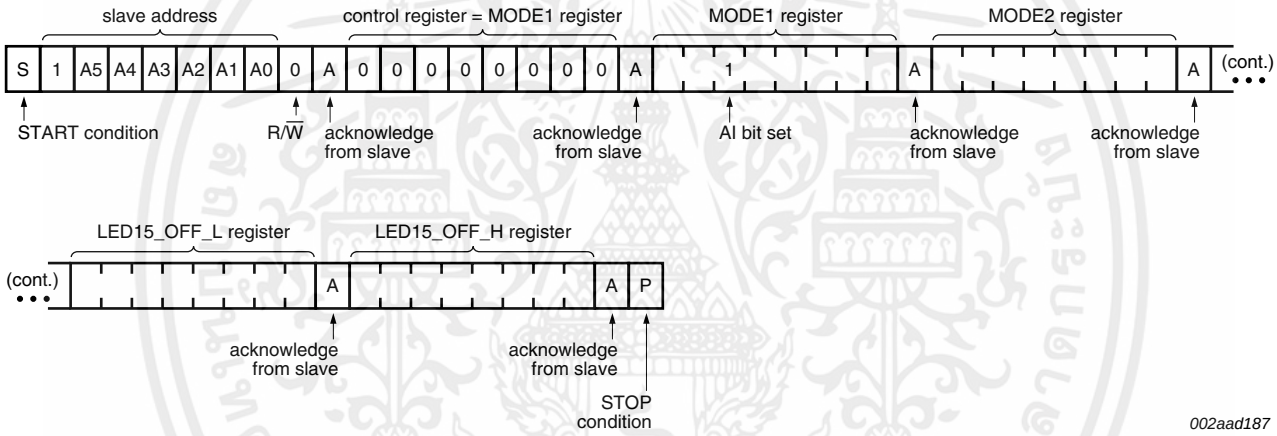


Fig 21. Write to all registers using the Auto-Increment feature; AI initially clear

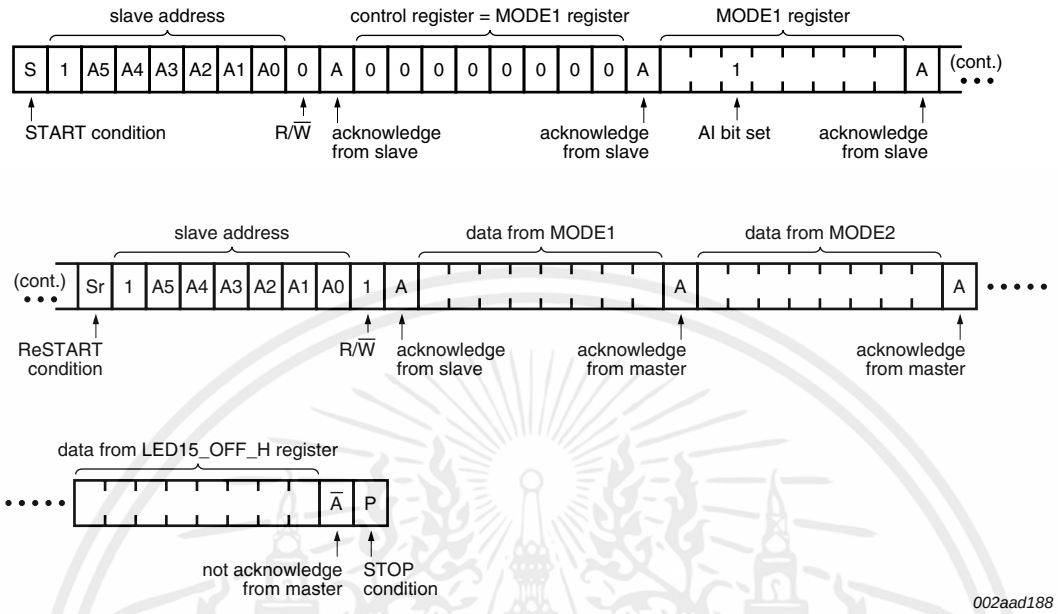


Fig 22. Read all registers using the Auto-Increment feature; AI initially clear

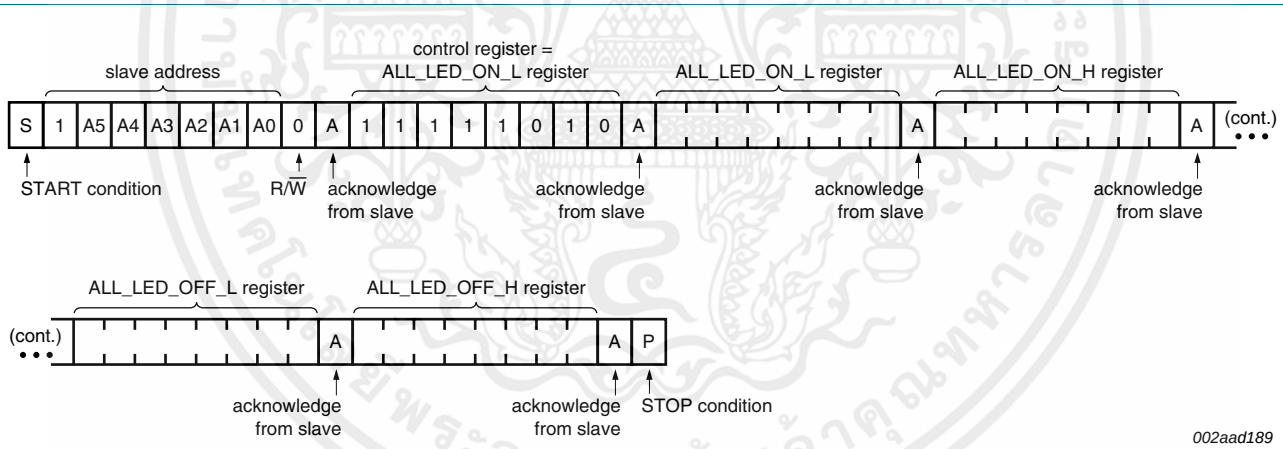


Fig 23. Write to ALL_LED_ON and ALL_LED_OFF registers using the Auto-Increment feature; AI initially set

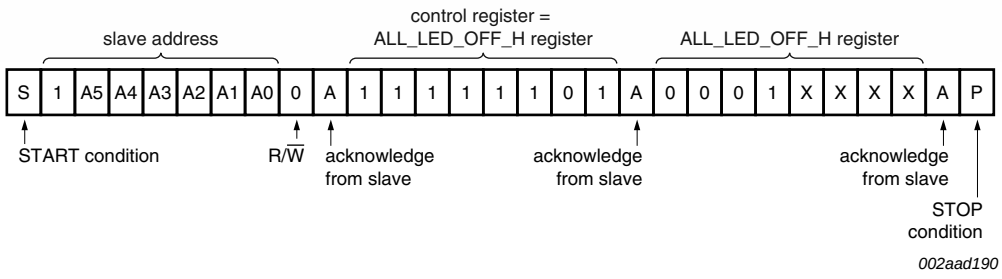
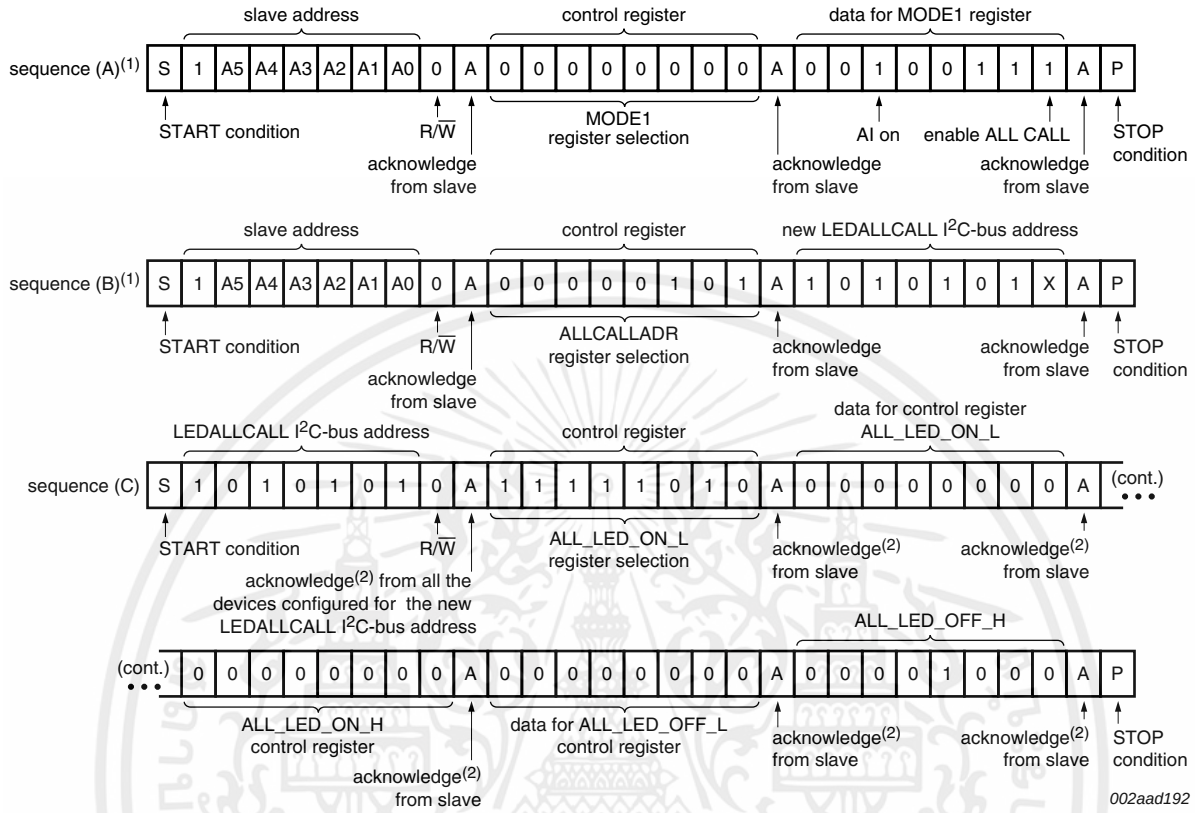


Fig 24. Write to ALL_LED_OFF_H to turn OFF all PWMs

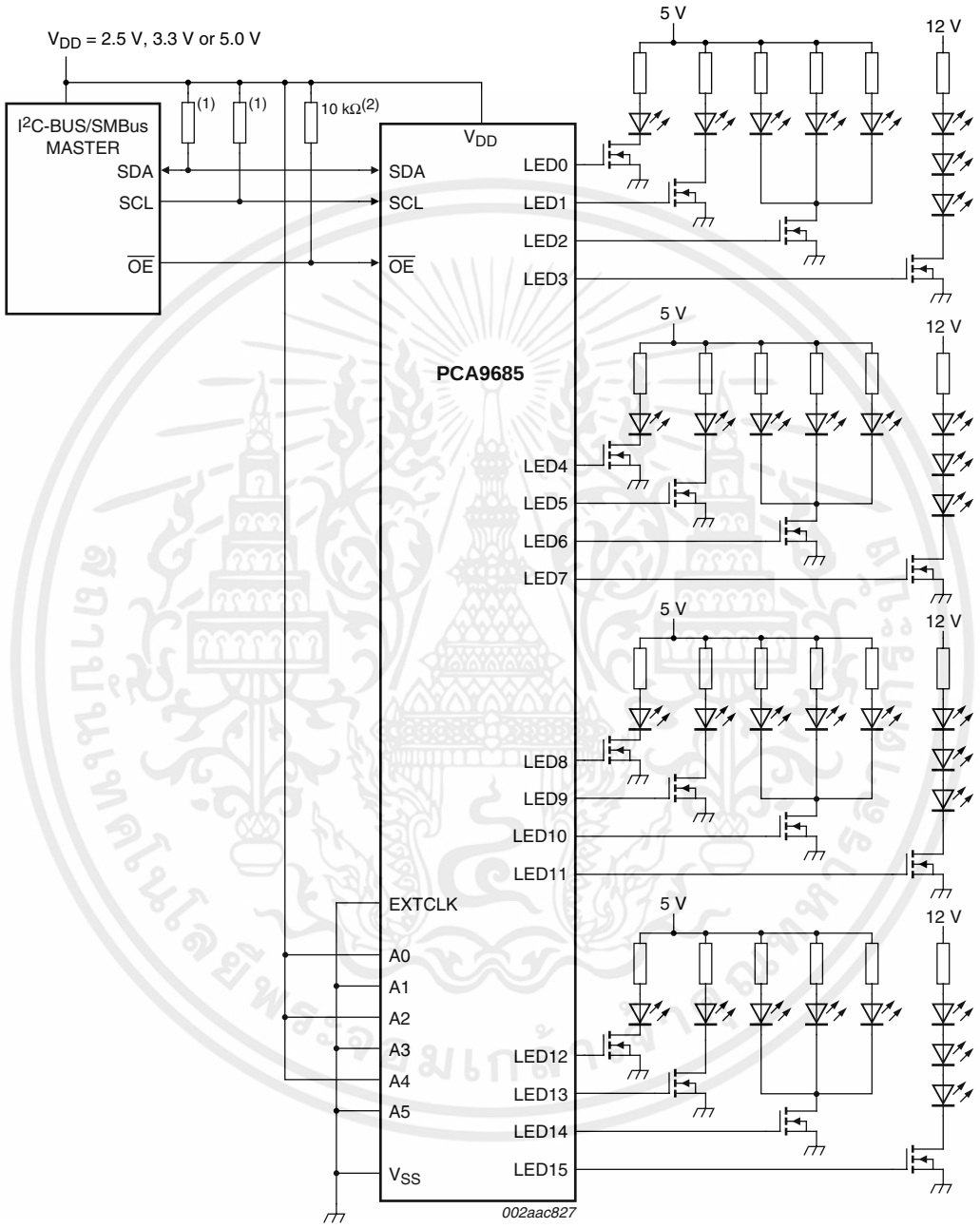


(1) In this example, several PCA9685s are used and the same sequences (A) and (B) above are sent to each of them.

(2) Acknowledge from all the slave devices configured for the new LED All Call I²C-bus address in sequence (B).

Fig 25. LED All Call I²C-bus address programming and LED All Call sequence example

10. Application design-in information



I²C-bus address = 1010 101x.

All 16 of the LEDn outputs configurable as either open-drain or totem pole. Mixing of configuration is not possible.

Remark: Set INVRT = 0, OUTDRV = 1, OUTNE = 01 (MODE2 register bits)

(1) Resistor value should be chosen by referencing section 7 of *UM10204, "I²C-bus specification and user manual"*.

(2) $\overline{OE}$ requires pull-up resistor if control signal from the master is open-drain.

Fig 26. Typical application

Question 1: What kind of edge rate control is there on the outputs?

- The typical edge rates depend on the output configuration, supply voltage, and the applied load. The outputs can be configured as either open-drain NMOS or totem pole outputs. If the customer is using the part to directly drive LEDs, they should be using it in an open-drain NMOS, if they are concerned about the maximum I_{SS} and ground bounce. The edge rate control was designed primarily to slow down the turn-on of the output device; it turns off rather quickly (~1.5 ns). In simulation, the typical turn-on time for the open-drain NMOS was ~14 ns (V_{DD} = 3.6 V; C_L = 50 pF; R_{PU} = 500 Ω).

Question 2: Is ground bounce possible?

- Ground bounce is a possibility, especially if all 16 outputs are changed at full current (25 mA each). There is a fair amount of decoupling capacitance on chip (~50 pF), which is intended to suppress some of the ground bounce. The customer will need to determine if additional decoupling capacitance externally placed as close as physically possible to the device is required.

Question 3: Can I really sink 400 mA through the single ground pin on the package and will this cause any ground bounce problem due to the PWM of the LEDs?

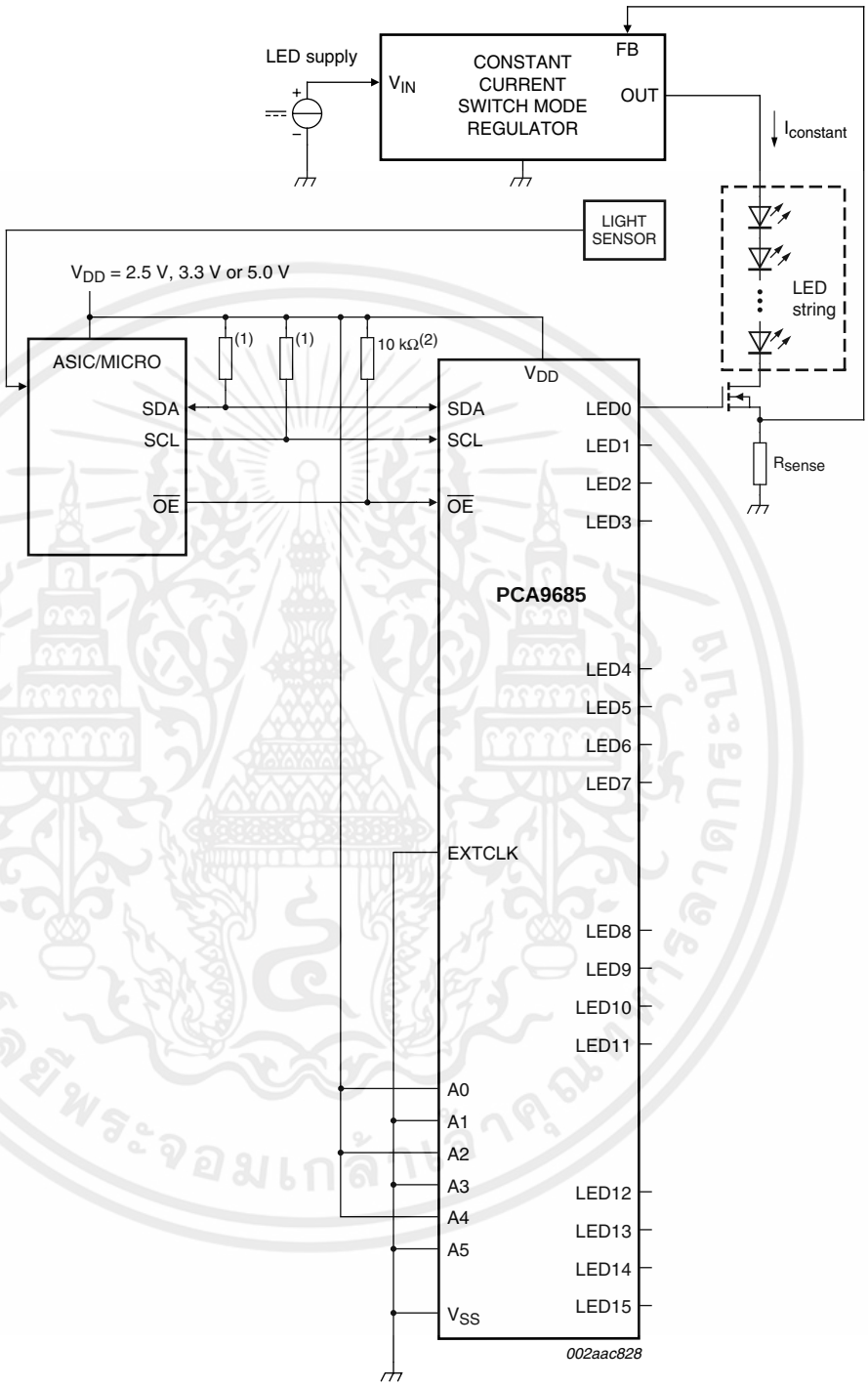
- Yes, you can sink 400 mA through a single ground pin on the **package**. Although the package only has one ground pin, there are two ground pads on the die itself connected to this one pin. Although some ground bounce is likely, it will not disrupt the operation of the part and would be reduced by the external decoupling capacitance.

Question 4: I can't turn the LEDs on or off, but their registers are set properly. Why?

- Check the MODE1 register SLEEP (bit 4) setting. The bit needs to be 0 in order to enable the clocking. If both clock sources (internal osc and EXTCLK) are turned OFF (bit 4 = 1), the LEDs cannot be dimmed or blinked.

Question 5: I'm using LEDs with integrated Zener diodes and the IC is getting very hot. Why?

- The IC outputs can be set to either open-drain or push-pull and default to push-pull outputs. In this application with the Zener diodes, they need to be set to open-drain since in the push-pull architecture there is a low resistance path to GND through the Zener and this is causing the IC to overheat.



I²C-bus address = 1010 101x.

Remark: Set INVRT = 0, OUTDRV = 1, OUTNE = 01 (MODE2 register bits) for this configuration.

(1) Resistor value should be chosen by referencing Section 7 of *UM10204, "I²C-bus specification and user manual"*.

(2) $\overline{OE}$ requires pull-up resistor if control signal from the master is open-drain.

Fig 27. LCD backlighting application

11. Limiting values

Table 12. Limiting values

In accordance with the Absolute Maximum Rating System (IEC 60134).

Symbol	Parameter	Conditions	Min	Max	Unit
V_{DD}	supply voltage		–0.5	+6.0	V
$V_{I/O}$	voltage on an input/output pin		V_{SS} – 0.5	5.5	V
$I_{O(LEDn)}$	output current on pin LEDn		-	25	mA
I_{SS}	ground supply current		-	400	mA
P_{tot}	total power dissipation		-	400	mW
T_{stg}	storage temperature		–65	+150	°C
T_{amb}	ambient temperature	operating	–40	+85	°C

12. Static characteristics

Table 13. Static characteristics

V_{DD} = 2.3 V to 5.5 V; V_{SS} = 0 V; T_{amb} = –40 °C to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions		Min	Typ	Max	Unit
Supply							
V_{DD}	supply voltage			2.3	-	5.5	V
I_{DD}	supply current	operating mode; no load; f_{SCL} = 1 MHz; V_{DD} = 2.3 V to 5.5 V		-	6	10	mA
I_{stb}	standby current	no load; f_{SCL} = 0 Hz; V_I = V_{DD} or V_{SS}; V_{DD} = 2.3 V to 5.5 V		-	2.2	15.5	µA
V_{POR}	power-on reset voltage	no load; V_I = V_{DD} or V_{SS}	[1]	-	1.70	2.0	V
Input SCL; input/output SDA							
V_{IL}	LOW-level input voltage			–0.5	-	+0.3V_{DD}	V
V_{IH}	HIGH-level input voltage			0.7V_{DD}	-	5.5	V
I_{OL}	LOW-level output current	V_{OL} = 0.4 V; V_{DD} = 2.3 V		20	28	-	mA
		V_{OL} = 0.4 V; V_{DD} = 5.0 V		30	40	-	mA
I_L	leakage current	V_I = V_{DD} or V_{SS}		–1	-	+1	µA
C_i	input capacitance	V_I = V_{SS}		-	6	10	pF
LED driver outputs							
I_{OL}	LOW-level output current	V_{OL} = 0.5 V; V_{DD} = 2.3 V to 4.5 V	[2]	12	25	-	mA
$I_{OL(tot)}$	total LOW-level output current	V_{OL} = 0.5 V; V_{DD} = 4.5 V	[2]	-	-	400	mA
I_{OH}	HIGH-level output current	open-drain; V_{OH} = V_{DD}		–10	-	+10	µA
V_{OH}	HIGH-level output voltage	I_{OH} = –10 mA; V_{DD} = 2.3 V		1.6	-	-	V
		I_{OH} = –10 mA; V_{DD} = 3.0 V		2.3	-	-	V
		I_{OH} = –10 mA; V_{DD} = 4.5 V		4.0	-	-	V
I_{OZ}	OFF-state output current	3-state; V_{OH} = V_{DD} or V_{SS}		–10	-	+10	µA
C_o	output capacitance			-	5	8	pF

Table 13. Static characteristics *...continued*

V_{DD} = 2.3 V to 5.5 V; V_{SS} = 0 V; T_{amb} = −40 °C to +85 °C; unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Address inputs; $\overline{OE}$ input; EXTCLK						
V_{IL}	LOW-level input voltage		−0.5	-	+0.3V_{DD}	V
V_{IH}	HIGH-level input voltage		0.7V_{DD}	-	5.5	V
I_{LI}	input leakage current		−1	-	+1	μA
C_i	input capacitance		-	3	5	pF

[1] V_{DD} must be lowered to 0.2 V in order to reset part.

[2] Each bit must be limited to a maximum of 25 mA and the total package limited to 400 mA due to internal busing limits.

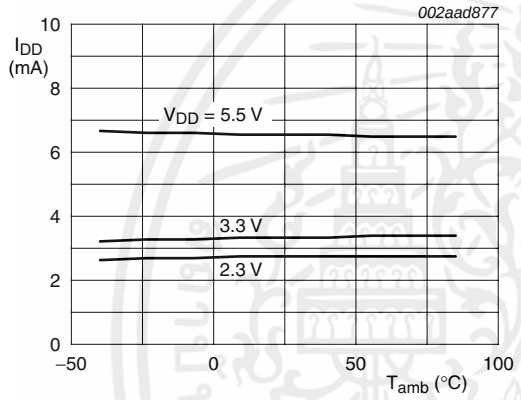


Fig 28. I_{DD} typical values with OSC on and f_{SCL} = 1 MHz versus temperature

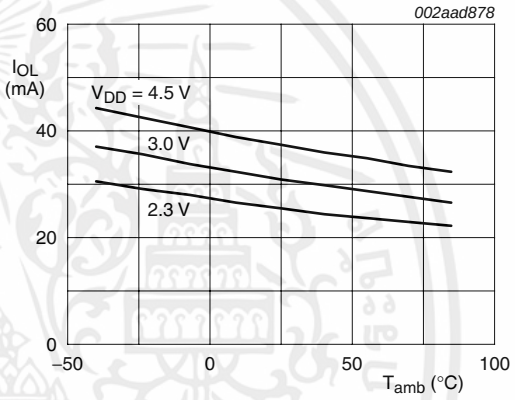


Fig 29. I_{OL} typical drive (LEDn outputs) versus temperature

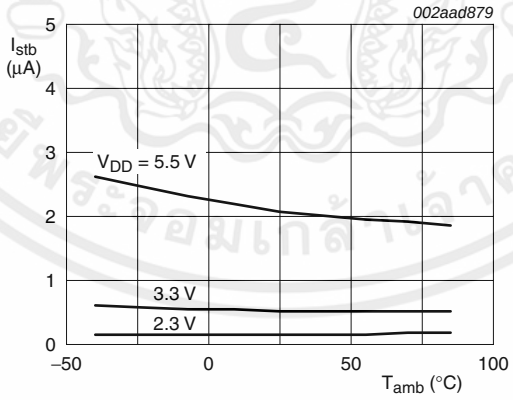


Fig 30. Standby supply current versus temperature

13. Dynamic characteristics

Table 14. Dynamic characteristics

Symbol	Parameter	Conditions	Standard-mode I²C-bus		Fast-mode I²C-bus		Fast-mode Plus I²C-bus		Unit
			Min	Max	Min	Max	Min	Max	
f_{SCL}	SCL clock frequency	[1]	0	100	0	400	0	1000	kHz
f_{EXTCLK}	frequency on pin EXTCLK		DC	50	DC	50	DC	50	MHz
t_{BUF}	bus free time between a STOP and START condition		4.7	-	1.3	-	0.5	-	µs
$t_{HD;STA}$	hold time (repeated) START condition		4.0	-	0.6	-	0.26	-	µs
$t_{SU;STA}$	set-up time for a repeated START condition		4.7	-	0.6	-	0.26	-	µs
$t_{SU;STO}$	set-up time for STOP condition		4.0	-	0.6	-	0.26	-	µs
$t_{HD;DAT}$	data hold time		0	-	0	-	0	-	ns
$t_{VD;ACK}$	data valid acknowledge time	[2]	0.3	3.45	0.1	0.9	0.05	0.45	µs
$t_{VD;DAT}$	data valid time	[3]	0.3	3.45	0.1	0.9	0.05	0.45	µs
$t_{SU;DAT}$	data set-up time		250	-	100	-	50	-	ns
t_{LOW}	LOW period of the SCL clock		4.7	-	1.3	-	0.5	-	µs
t_{HIGH}	HIGH period of the SCL clock		4.0	-	0.6	-	0.26	-	µs
t_f	fall time of both SDA and SCL signals	[4][5]	-	300	$20 + 0.1C_b$ [6]	300	-	120	ns
t_r	rise time of both SDA and SCL signals		-	1000	$20 + 0.1C_b$ [6]	300	-	120	ns
t_{SP}	pulse width of spikes that must be suppressed by the input filter	[7]	-	50	-	50	-	50	ns
t_{PLZ}	LOW to OFF-state propagation delay	$\overline{OE}$ to LEDn; OUTNE[1:0] = 10 or 11 in MODE2 register	-	40	-	40	-	40	ns
t_{PZL}	OFF-state to LOW propagation delay	$\overline{OE}$ to LEDn; OUTNE[1:0] = 10 or 11 in MODE2 register	-	60	-	60	-	60	ns
t_{PHZ}	HIGH to OFF-state propagation delay	$\overline{OE}$ to LEDn; OUTNE[1:0] = 10 or 11 in MODE2 register	-	60	-	60	-	60	ns

Table 14. Dynamic characteristics *…continued*

Symbol	Parameter	Conditions	Standard-mode I²C-bus		Fast-mode I²C-bus		Fast-mode Plus I²C-bus		Unit
			Min	Max	Min	Max	Min	Max	
t_{PZH}	OFF-state to HIGH propagation delay	$\overline{OE}$ to LEDn; OUTNE[1:0] = 10 or 11 in MODE2 register	-	40	-	40	-	40	ns
t_{PLH}	LOW to HIGH propagation delay	$\overline{OE}$ to LEDn; OUTNE[1:0] = 01 in MODE2 register	-	40	-	40	-	40	ns
t_{PHL}	HIGH to LOW propagation delay	$\overline{OE}$ to LEDn; OUTNE[1:0] = 00 in MODE2 register	-	60	-	60	-	60	ns

[1] Minimum SCL clock frequency is limited by the bus time-out feature, which resets the serial bus interface if either SDA or SCL is held LOW for a minimum of 25 ms. Disable bus time-out feature for DC operation.

[2] $t_{VD;ACK}$ = time for Acknowledgement signal from SCL LOW to SDA (out) LOW.

[3] $t_{VD;DAT}$ = minimum time for SDA data out to be valid following SCL LOW.

[4] A master device must internally provide a hold time of at least 300 ns for the SDA signal (refer to the V_{IL} of the SCL signal) in order to bridge the undefined region of SCL's falling edge.

[5] The maximum t_f for the SDA and SCL bus lines is specified at 300 ns. The maximum fall time (t_f) for the SDA output stage is specified at 250 ns. This allows series protection resistors to be connected between the SDA and the SCL pins and the SDA/SCL bus lines without exceeding the maximum specified t_f.

[6] C_b = total capacitance of one bus line in pF.

[7] Input filters on the SDA and SCL inputs suppress noise spikes less than 50 ns.

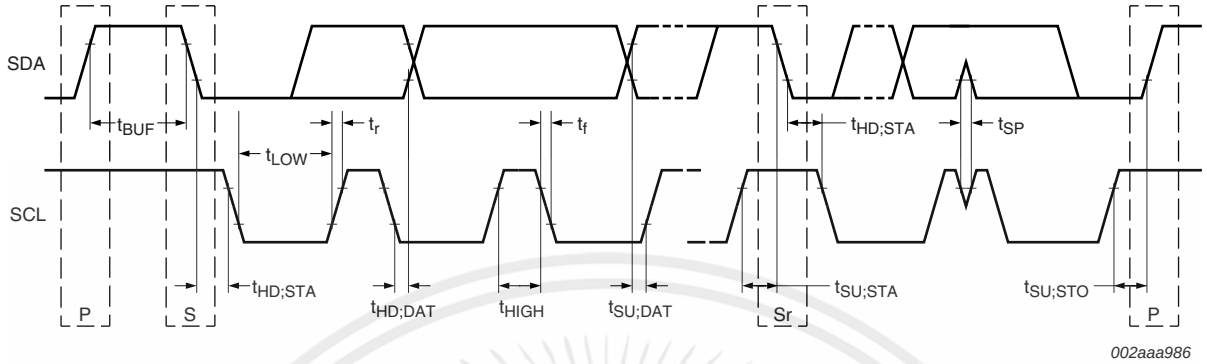
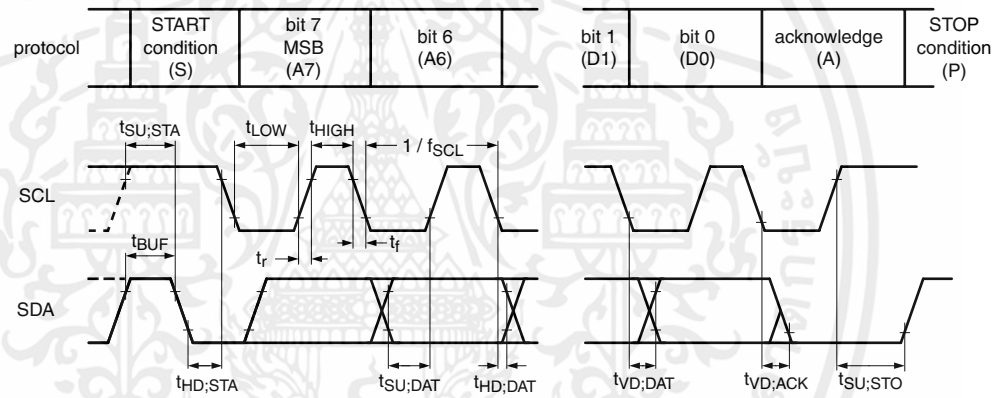


Fig 31. Definition of timing



Rise and fall times refer to V_{IL} and V_{IH}.

Fig 32. I²C-bus timing diagram

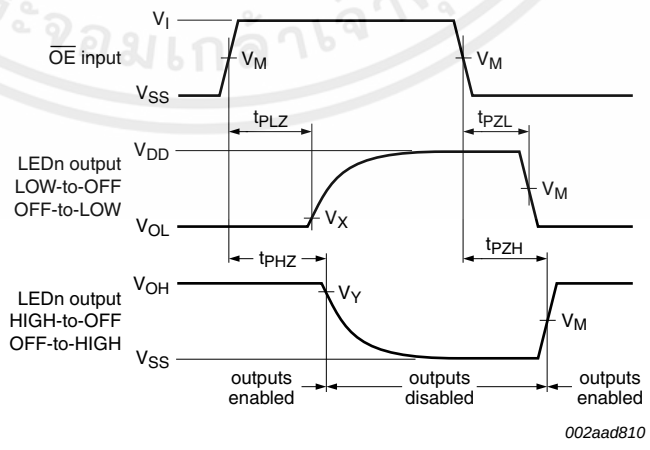
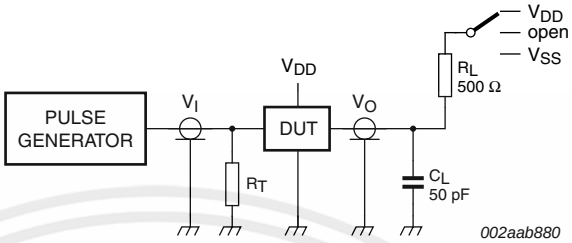


Fig 33. t_{PLZ}, t_{PZL} and t_{PHZ}, t_{PZH} times

14. Test information

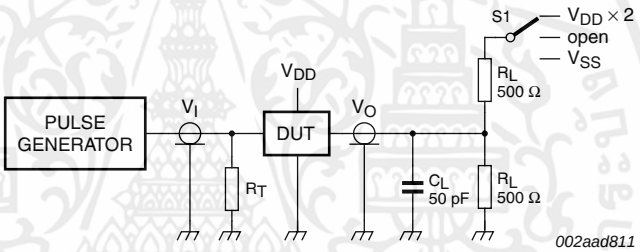


R_L = Load resistor for LEDn.

C_L = Load capacitance includes jig and probe capacitance.

R_T = Termination resistance should be equal to the output impedance Z_o of the pulse generators.

Fig 34. Test circuitry for switching times



R_L = Load resistor for LEDn.

C_L = Load capacitance includes jig and probe capacitance.

R_T = Termination resistance should be equal to the output impedance Z_o of the pulse generators.

Test data are given in Table 15.

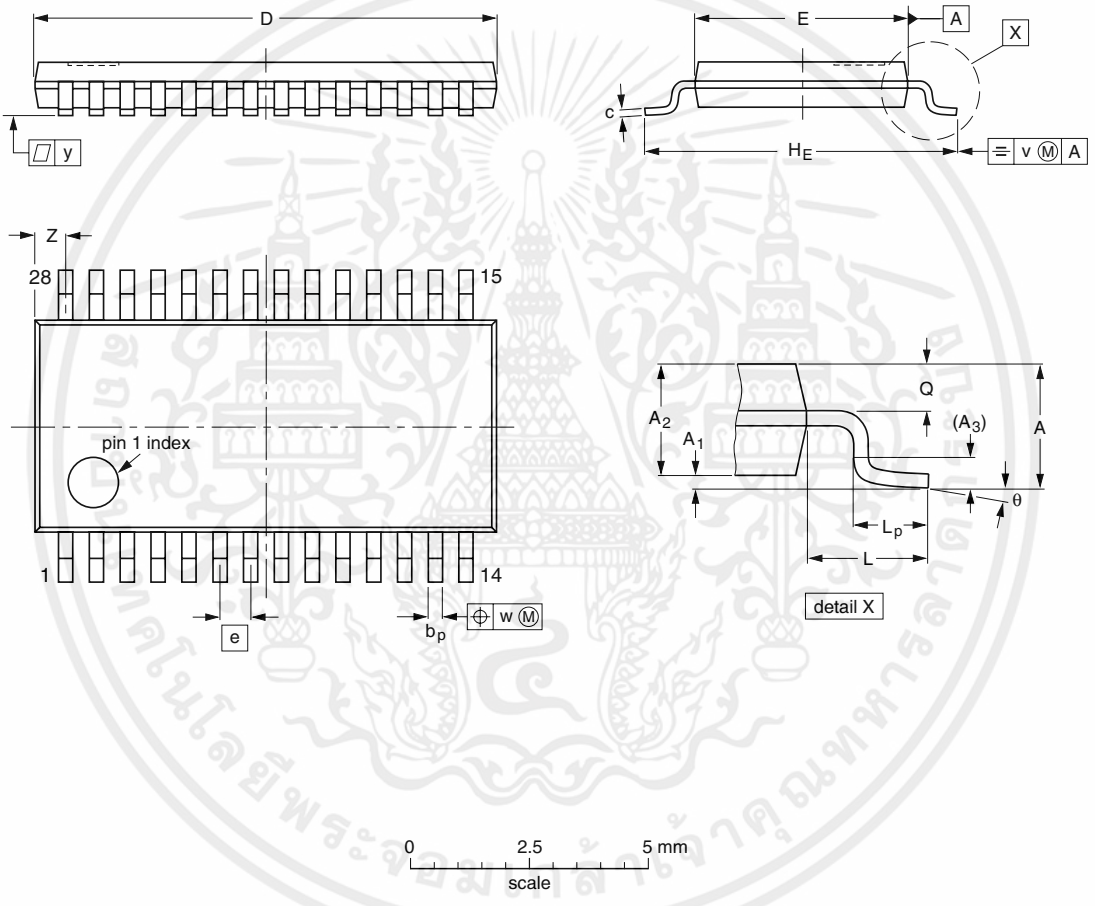
Fig 35. Test circuitry for switching times for enable/disable

Table 15. Test data for enable/disable switching times

Test	Load		Switch
	C_L	R_L	
t_{PD}	50 pF	500 Ω	open
t_{PLZ}, t_{PZL}	50 pF	500 Ω	$V_{DD} \times 2$
t_{PHZ}, t_{PZH}	50 pF	500 Ω	V_{SS}

15. Package outline

TSSOP28: plastic thin shrink small outline package; 28 leads; body width 4.4 mm **SOT361-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	D (1)	E (2)	e	H_E	L	L_p	Q	v	w	y	Z (1)	θ
mm	1.1	0.15 0.05	0.95 0.80	0.25	0.30 0.19	0.2 0.1	9.8 9.6	4.5 4.3	0.65	6.6 6.2	1	0.75 0.50	0.4 0.3	0.2	0.13	0.1	0.8 0.5	8° 0°

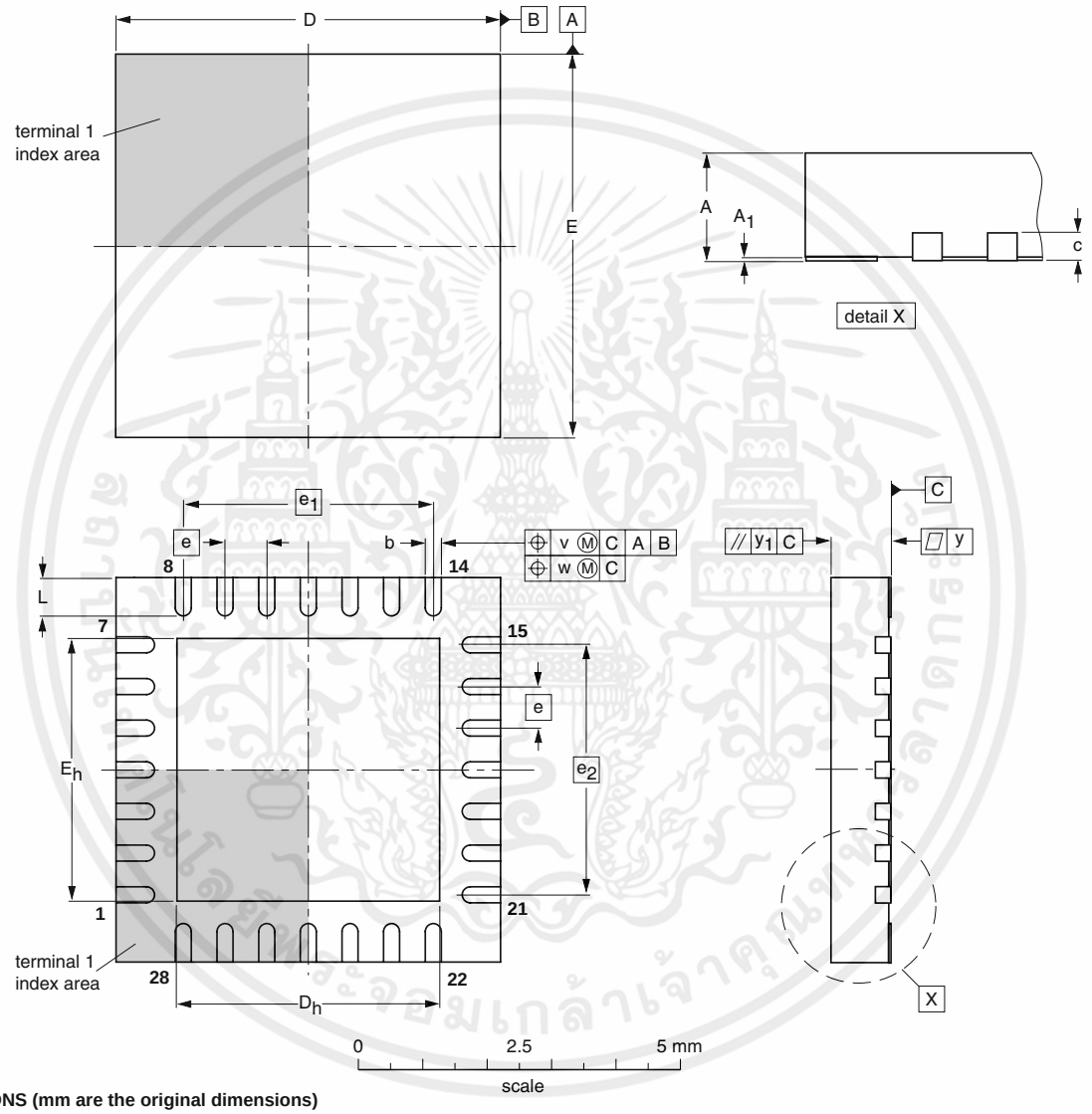
Notes

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.
2. Plastic interlead protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT361-1		MO-153				~~99-12-27~~ 03-02-19

Fig 36. Package outline SOT361-1 (TSSOP28)

HVQFN28: plastic thermal enhanced very thin quad flat package; no leads; 28 terminals; body 6 x 6 x 0.85 mm **SOT788-1**



DIMENSIONS (mm are the original dimensions)

UNIT	A(1) max.	A_1	b	c	D(1)	D_h	E(1)	E_h	e	e_1	e_2	L	v	w	y	y_1
mm	1	0.05 0.00	0.35 0.25	0.2	6.1 5.9	4.25 3.95	6.1 5.9	4.25 3.95	0.65	3.9	3.9	0.75 0.50	0.1	0.05	0.05	0.1

Note

1. Plastic or metal protrusions of 0.075 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	JEITA			
SOT788-1	- - -	MO-220	- - -			02-10-22

Fig 37. Package outline SOT788-1 (HVQFN28)

16. Handling information

All input and output pins are protected against ElectroStatic Discharge (ESD) under normal handling. When handling ensure that the appropriate precautions are taken as described in *JESD625-A* or equivalent standards.

17. Soldering of SMD packages

This text provides a very brief insight into a complex technology. A more in-depth account of soldering ICs can be found in Application Note *AN10365 "Surface mount reflow soldering description"*.

17.1 Introduction to soldering

Soldering is one of the most common methods through which packages are attached to Printed Circuit Boards (PCBs), to form electrical circuits. The soldered joint provides both the mechanical and the electrical connection. There is no single soldering method that is ideal for all IC packages. Wave soldering is often preferred when through-hole and Surface Mount Devices (SMDs) are mixed on one printed wiring board; however, it is not suitable for fine pitch SMDs. Reflow soldering is ideal for the small pitches and high densities that come with increased miniaturization.

17.2 Wave and reflow soldering

Wave soldering is a joining technology in which the joints are made by solder coming from a standing wave of liquid solder. The wave soldering process is suitable for the following:

- Through-hole components
- Leaded or leadless SMDs, which are glued to the surface of the printed circuit board

Not all SMDs can be wave soldered. Packages with solder balls, and some leadless packages which have solder lands underneath the body, cannot be wave soldered. Also, leaded SMDs with leads having a pitch smaller than ~0.6 mm cannot be wave soldered, due to an increased probability of bridging.

The reflow soldering process involves applying solder paste to a board, followed by component placement and exposure to a temperature profile. Leaded packages, packages with solder balls, and leadless packages are all reflow solderable.

Key characteristics in both wave and reflow soldering are:

- Board specifications, including the board finish, solder masks and vias
- Package footprints, including solder thieves and orientation
- The moisture sensitivity level of the packages
- Package placement
- Inspection and repair
- Lead-free soldering versus SnPb soldering

17.3 Wave soldering

Key characteristics in wave soldering are:

- Process issues, such as application of adhesive and flux, clinching of leads, board transport, the solder wave parameters, and the time during which components are exposed to the wave
- Solder bath specifications, including temperature and impurities

17.4 Reflow soldering

Key characteristics in reflow soldering are:

- Lead-free versus SnPb soldering; note that a lead-free reflow process usually leads to higher minimum peak temperatures (see Figure 38) than a SnPb process, thus reducing the process window
- Solder paste printing issues including smearing, release, and adjusting the process window for a mix of large and small components on one board
- Reflow temperature profile; this profile includes preheat, reflow (in which the board is heated to the peak temperature) and cooling down. It is imperative that the peak temperature is high enough for the solder to make reliable solder joints (a solder paste characteristic). In addition, the peak temperature must be low enough that the packages and/or boards are not damaged. The peak temperature of the package depends on package thickness and volume and is classified in accordance with Table 16 and 17

Table 16. SnPb eutectic process (from J-STD-020C)

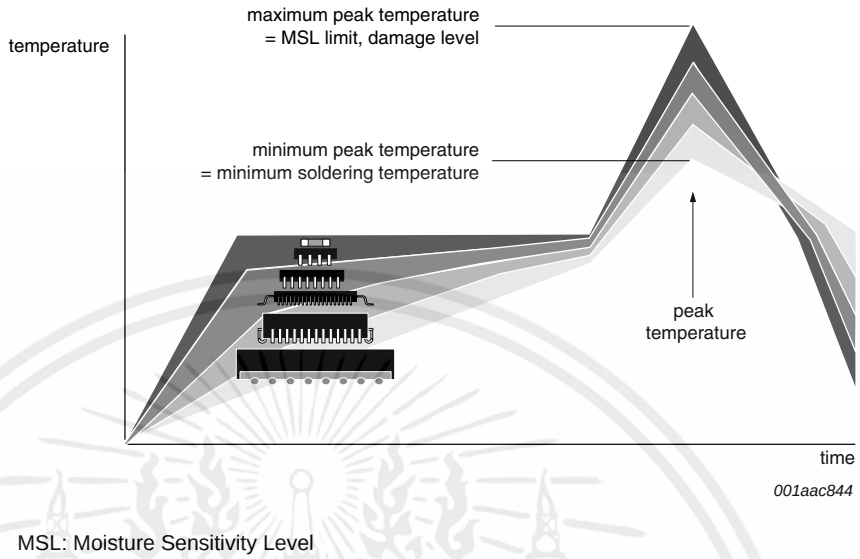
Package thickness (mm)	Package reflow temperature (°C)	
	Volume (mm³)	
	< 350	≥ 350
< 2.5	235	220
≥ 2.5	220	220

Table 17. Lead-free process (from J-STD-020C)

Package thickness (mm)	Package reflow temperature (°C)		
	Volume (mm³)		
	< 350	350 to 2000	> 2000
< 1.6	260	260	260
1.6 to 2.5	260	250	245
> 2.5	250	245	245

Moisture sensitivity precautions, as indicated on the packing, must be respected at all times.

Studies have shown that small packages reach higher temperatures during reflow soldering, see Figure 38.



MSL: Moisture Sensitivity Level

Fig 38. Temperature profiles for large and small components

For further information on temperature profiles, refer to Application Note *AN10365 "Surface mount reflow soldering description"*.

18. Abbreviations

Table 18. Abbreviations

Acronym	Description
CDM	Charged-Device Model
DUT	Device Under Test
EMI	ElectroMagnetic Interference
ESD	ElectroStatic Discharge
HBM	Human Body Model
I²C-bus	Inter-Integrated Circuit bus
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LSB	Least Significant Bit
MM	Machine Model
MSB	Most Significant Bit
NMOS	Negative-channel Metal-Oxide Semiconductor
PCB	Printed-Circuit Board
PMOS	Positive-channel Metal-Oxide Semiconductor
POR	Power-On Reset
PWM	Pulse Width Modulation; Pulse Width Modulator
RGB	Red/Green/Blue
RGBA	Red/Green/Blue/Amber
SMBus	System Management Bus

19. Revision history

Table 19. Revision history

Document ID	Release date	Data sheet status	Change notice	Supersedes
PCA9685 v.3	20100902	Product data sheet	-	PCA9685 v.2
Modifications:	• Table 1 “Ordering information”: Topside mark for PCA9685BS changed from “PCA9685BS” to “P9685”			
PCA9685 v.2	20090716	Product data sheet	-	PCA9685 v.1
PCA9685 v.1	20080724	Product data sheet	-	-

20. Legal information

20.1 Data sheet status

Document status[1][2]	Product status[3]	Definition
Objective [short] data sheet	Development	This document contains data from the objective specification for product development.
Preliminary [short] data sheet	Qualification	This document contains data from the preliminary specification.
Product [short] data sheet	Production	This document contains the product specification.

[1] Please consult the most recently issued document before initiating or completing a design.

[2] The term 'short data sheet' is explained in section "Definitions".

[3] The product status of device(s) described in this document may have changed since this document was published and may differ in case of multiple devices. The latest product status information is available on the Internet at URL http://www.nxp.com.

20.2 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

Short data sheet — A short data sheet is an extract from a full data sheet with the same product type number(s) and title. A short data sheet is intended for quick reference only and should not be relied upon to contain detailed and full information. For detailed and full information see the relevant full data sheet, which is available on request via the local NXP Semiconductors sales office. In case of any inconsistency or conflict with the short data sheet, the full data sheet shall prevail.

Product specification — The information and data provided in a Product data sheet shall define the specification of the product as agreed between NXP Semiconductors and its customer, unless NXP Semiconductors and customer have explicitly agreed otherwise in writing. In no event however, shall an agreement be valid in which the NXP Semiconductors product is deemed to offer functions and qualities beyond those described in the Product data sheet.

20.3 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Limiting values — Stress above one or more limiting values (as defined in the Absolute Maximum Ratings System of IEC 60134) will cause permanent damage to the device. Limiting values are stress ratings only and (proper) operation of the device at these or any other conditions above those given in the Recommended operating conditions section (if present) or the Characteristics sections of this document is not warranted. Constant or repeated exposure to limiting values will permanently and irreversibly affect the quality and reliability of the device.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

No offer to sell or license — Nothing in this document may be interpreted or construed as an offer to sell products that is open for acceptance or the grant, conveyance or implication of any license under any copyrights, patents or other industrial or intellectual property rights.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from national authorities.

Non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

20.4 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

I²C-bus — logo is a trademark of NXP B.V.

21. Contact information

For more information, please visit: **http://www.nxp.com**

For sales office addresses, please send an email to: **salesaddresses@nxp.com**

22. Contents

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 2 September 2010
Document identifier: PCA9685

SONGLE RELAY

松乐继电器 ® SONGLE RELAY	RELAY ISO9002	SRD

1. MAIN FEATURES

- Switching capacity available by 10A in spite of small size design for highdensity P.C. board mounting technique.
- UL,CUL,TUV recognized.
- Selection of plastic material for high temperature and better chemical solution performance.
- Sealed types available.
- Simple relay magnetic circuit to meet low cost of mass production.

2. APPLICATIONS

- Domestic appliance, office machine, audio, equipment, automobile, etc.
  (Remote control TV receiver, monitor display, audio equipment high rushing current use application.)

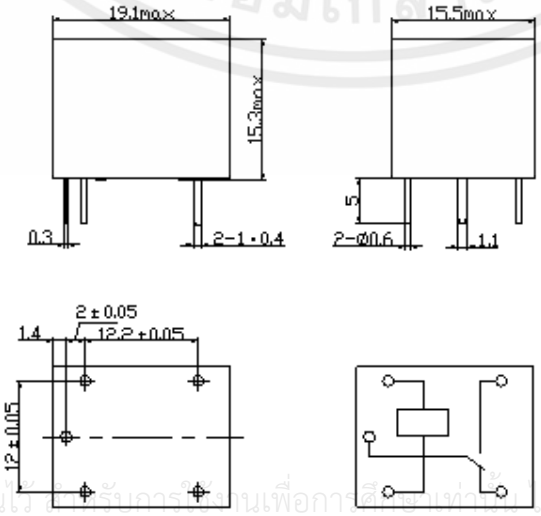
3. ORDERING INFORMATION

SRD	XX VDC	S	L	C
Model of relay	Nominal coil voltage	Structure	Coil sensitivity	Contact form
SRD	03、05、06、09、12、24、48VDC	S:Sealed type	L:0.36W	A:1 form A
				B:1 form B
		F:Flux free type	D:0.45W	C:1 form C

4. RATING

CCC	FILE NUMBER:CH0052885-2000	7A/240VDC
CCC	FILE NUMBER:CH0036746-99	10A/250VDC
UL /CUL	FILE NUMBER: E167996	10A/125VAC 28VDC
TUV	FILE NUMBER: R9933789	10A/240VAC 28VDC

5. DIMENSION(unit:mm) DRILLING(unit:mm) WIRING DIAGRAM



6. COIL DATA CHART (AT20°C)

Coil Sensitivity	Coil Voltage Code	Nominal Voltage (VDC)	Nominal Current (mA)	Coil Resistance (Ω) ±10%	Power Consumption (W)	Pull-In Voltage (VDC)	Drop-Out Voltage (VDC)	Max-Allowable Voltage (VDC)
SRD (High Sensitivity)	03	03	120	25	abt. 0.36W	75%Max.	10% Min.	120%
	05	05	71.4	70				
	06	06	60	100				
	09	09	40	225				
	12	12	30	400				
	24	24	15	1600				
	48	48	7.5	6400				
SRD (Standard)	03	03	150	20	abt. 0.45W	75% Max.	10% Min.	110%
	05	05	89.3	55				
	06	06	75	80				
	09	09	50	180				
	12	12	37.5	320				
	24	24	18.7	1280				
	48	48	10	4500	abt. 0.51W			

7. CONTACT RATING

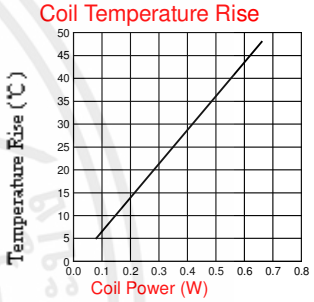
Item \ Type	SRD	
	FORM C	FORM A
Contact Capacity Resistive Load (cosΦ=1)	7A 28VDC 10A 125VAC 7A 240VAC	10A 28VDC 10A 240VAC
Inductive Load (cosΦ=0.4 L/R=7msec)	3A 120VAC 3A 28VDC	5A 120VAC 5A 28VDC
Max. Allowable Voltage	250VAC/110VDC	250VAC/110VDC
Max. Allowable Power Force	800VAC/240W	1200VA/300W
Contact Material	AgCdO	AgCdO

8. PERFORMANCE (at initial value)

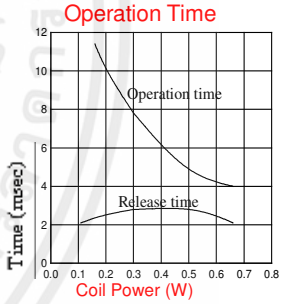
Item \ Type	SRD
Contact Resistance	100mΩ Max.
Operation Time	10msec Max.
Release Time	5msec Max.
Dielectric Strength Between coil & contact Between contacts	 1500VAC 50/60HZ (1 minute) 1000VAC 50/60HZ (1 minute)
Insulation Resistance	100 MΩ Min. (500VDC)
Max. ON/OFF Switching Mechanically Electrically	 300 operation/min 30 operation/min
Ambient Temperature	-25°C to +70°C
Operating Humidity	45 to 85% RH
Vibration Endurance Error Operation	 10 to 55Hz Double Amplitude 1.5mm 10 to 55Hz Double Amplitude 1.5mm
Shock Endurance Error Operation	 100G Min. 10G Min.
Life Expectancy Mechanically Electrically	 10^7 operations. Min. (no load) 10^5 operations. Min. (at rated coil voltage)
Weight	abt. 10grs.

9.REFERENCE DATA

Coil Temperature Rise

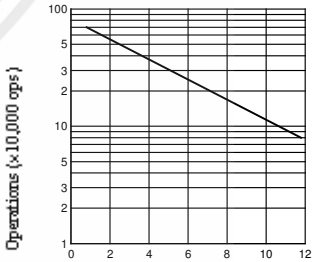


Operation Time

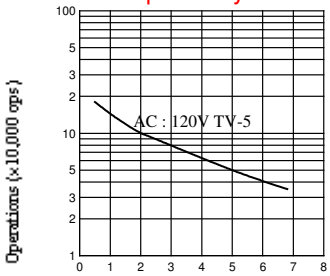


Life Expectancy

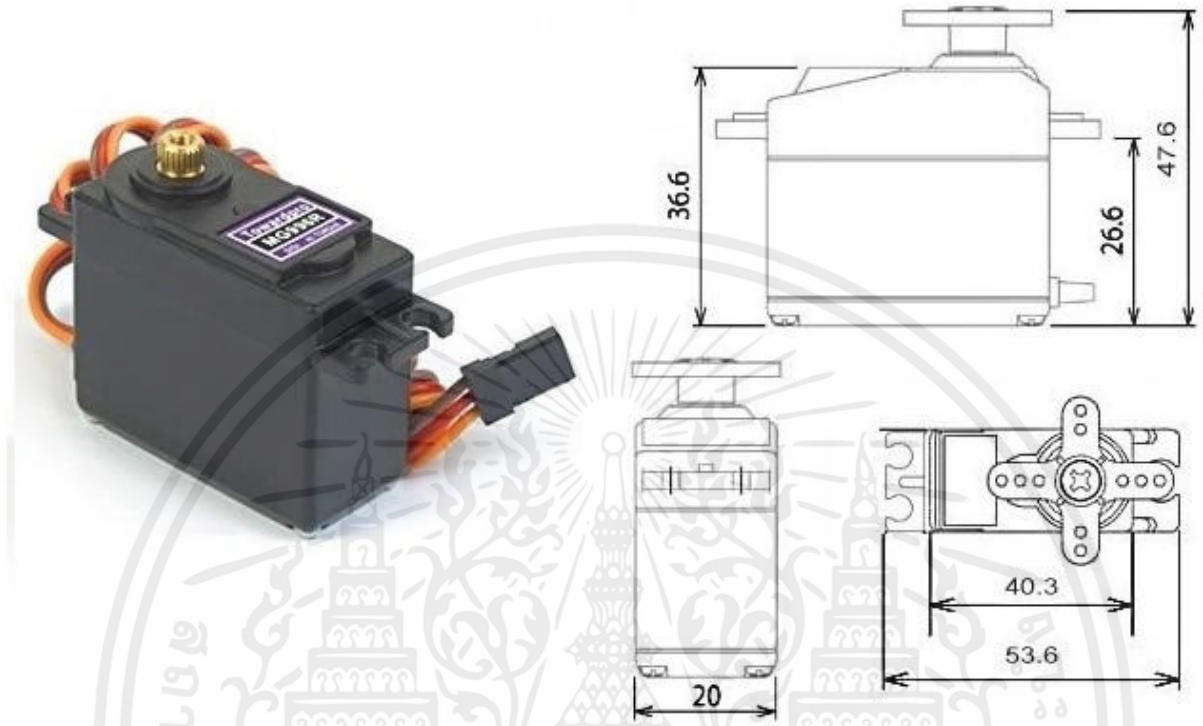
AC120V/DC24V cosΦ=1



Life Expectancy



MG996R High Torque
Metal Gear Dual Ball Bearing Servo



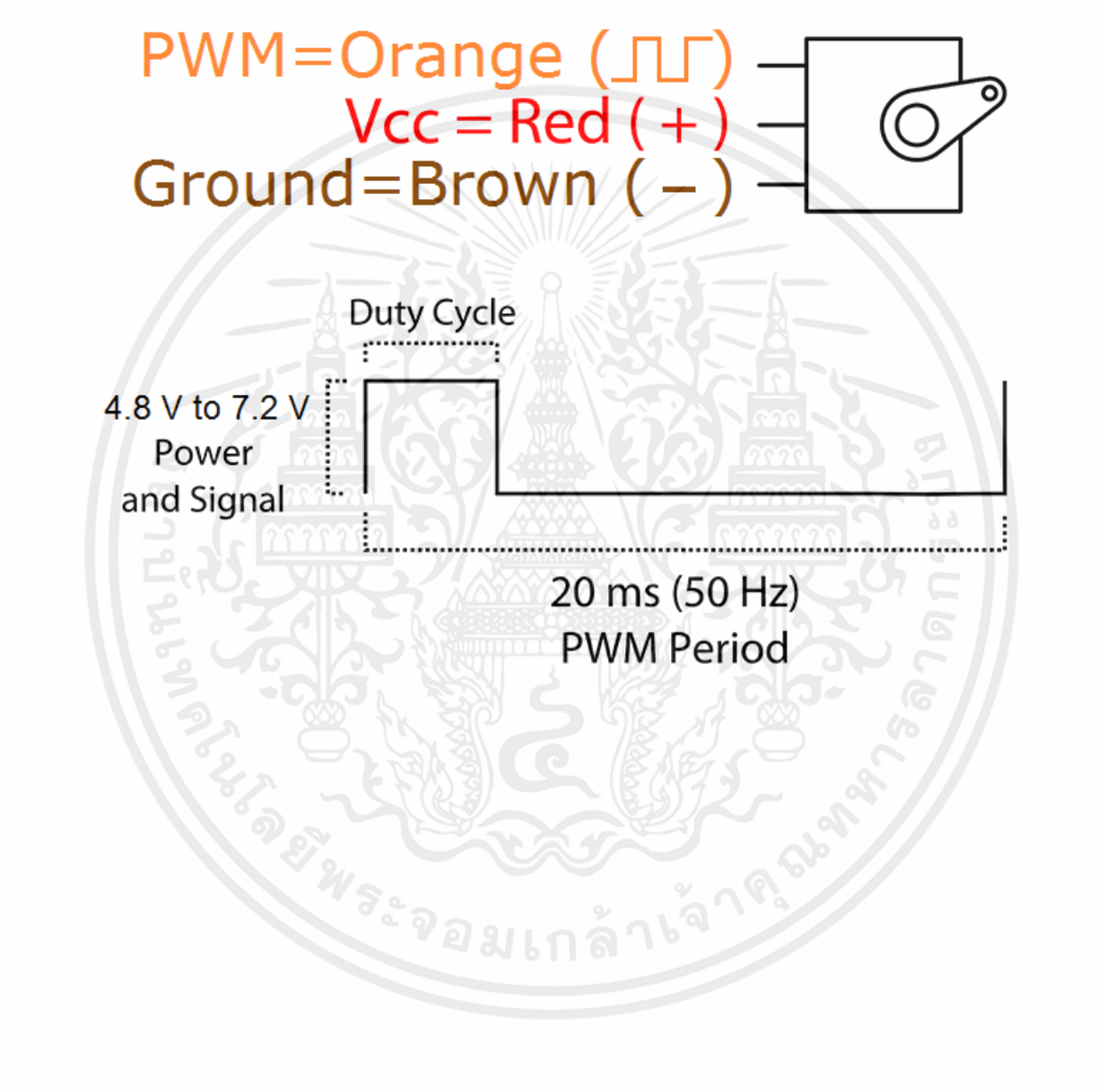
This High-Torque MG996R Digital Servo features metal gearing resulting in extra high 10kg stalling torque in a tiny package. The MG996R is essentially an upgraded version of the famous MG995 servo, and features upgraded shock-proofing and a redesigned PCB and IC control system that make it much more accurate than its predecessor. The gearing and motor have also been upgraded to improve dead bandwith and centering. The unit comes complete with 30cm wire and 3 pin 'S' type female header connector that fits most receivers, including Futaba, JR, GWS, Cirrus, Blue Bird, Blue Arrow, Corona, Berg, Spektrum and Hitec.

This high-torque standard servo can rotate approximately 120 degrees (60 in each direction). You can use any servo code, hardware or library to control these servos, so it's great for beginners who want to make stuff move without building a motor controller with feedback & gear box, especially since it will fit in small places. The MG996R Metal Gear Servo also comes with a selection of arms and hardware to get you set up nice and fast!

Specifications

- Weight: 55 g
- Dimension: 40.7 x 19.7 x 42.9 mm approx.
- Stall torque: 9.4 kgf·cm (4.8 V), 11 kgf·cm (6 V)
- Operating speed: 0.17 s/60º (4.8 V), 0.14 s/60º (6 V)

- Operating voltage: 4.8 V a 7.2 V
- Running Current 500 mA – 900 mA (6V)
- Stall Current 2.5 A (6V)
- Dead band width: 5 µs
- Stable and shock proof double ball bearing design
- Temperature range: 0 ºC – 55 ºC

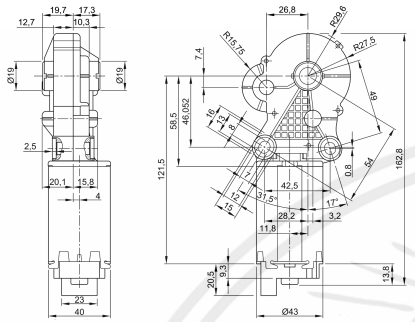


A Nidec Group Company
Nidec Motors & Actuators
All for dreams

Series GMPD
Motor type 405 061

Design Data	
Commutation	Brushed
Direction of rotation	Bi-directional
Bearing type	A:Sleeve - B:Sleeve

0320 BR



Performance data		
Rated voltage [V]	U_N	12
Nominal torque [Nm]	M_N	2.00
No-load speed [min^{-1}]	n_0	21.0
Nominal power [W]	P_N	3.95
Nominal current [A]	I_N	4.0
Nominal force [kN]	F_N	0.00
Duty cycle		s3

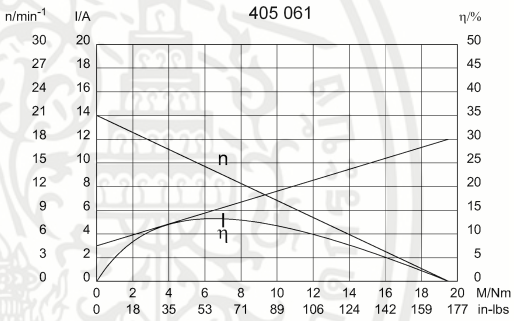
Sensor data	
Pulses	210
Output channels	2

Other data	
Gear ratio	210/1
Gear wheel material	plastic
Suppression components	5,5µH, 47nF
Enclosure class	IP 40
Weight [kg]	0.000

Remarks:

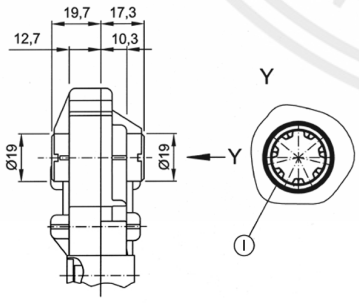
Motor picture

Characteristic curves



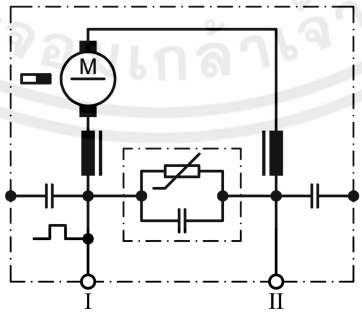
Output shaft drawing (W), Wiring diagrams (S) and Connector layout (K)

W 224

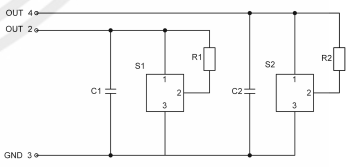


I through Profile KRC 316928 AEIZ 01

S 93

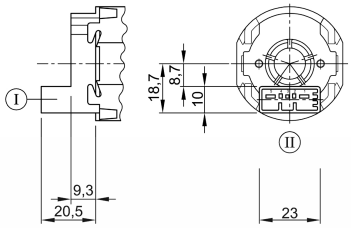


I Terminal 1
II Terminal 5



Output shaft drawing (W), Wiring diagrams (S) and Connector layout (K)

K 220



Connector TYCO C-208-15621 (Z) mating with:
Connector housing 1379217-3 **& cover**
1379218-2
www.tycoelectronics.com

Notes

ประวัติผู้จัดทำ

ชื่อ-นามสกุล	นายณรงค์ เรืองฤทธิ์
วัน เดือน ปี	7 ตุลาคม 2541
ที่อยู่ปัจจุบัน	62 หมู่ 2 ต.กำเนิดนพคุณ อ.บางสะพาน จ.ประจวบคีรีขันธ์ 77140
ประวัติการศึกษา	พ.ศ. 2559 สำเร็จการศึกษา ระดับมัธยมศึกษาตอนปลาย สายวิทย์-คณิต จากโรงเรียนมาบอำมฤตวิทยา จ.ชุมพร
อีเมล์	Email. 60511046@kmitl.ac.th Tel. 093-5946106

ชื่อ-นามสกุล	นายพัชรพล ยาฟอง
วัน เดือน ปี	2 กันยายน 2541
ที่อยู่ปัจจุบัน	14/2 ถ.เชิดบุญชาติ ต.หล่มสัก อ.หล่มสัก จ.เพชรบูรณ์ 67110
ประวัติการศึกษา	พ.ศ. 2559 สำเร็จการศึกษา ระดับมัธยมศึกษาตอนปลาย สายวิทย์-คณิต จากโรงเรียนป่าพะยอมพิทยาคม จ.พัทลุง
อีเมล์	Email. 60511061@kmitl.ac.th Tel. 091-0181861

ใบรับรองรูปเล่มปริญญานิพนธ์

ปริญญานิพนธ์ปีการศึกษา 2563
สาขาวิชาวิศวกรรมศาสตร์ หลักสูตรวิศวกรรมอิเล็กทรอนิกส์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร

ชื่อโครงงาน เครื่องคัดแยกขนาดและสีของมะนาวกึ่งอัตโนมัติ
The semi-automatic limes color and size sorter

ผู้จัดทำ

1.นาย ณรงค์ เรืองฤทธิ์ **รหัสนักศึกษา** 60511046 .

2.นาย พัชรพล ยาฟอง **รหัสนักศึกษา** 60511061 .

ด้วยข้าพเจ้านักศึกษาวิศวกรรมอิเล็กทรอนิกส์ สจล.วิทยาเขตชุมพรเขตรอุดมศักดิ์ จังหวัดชุมพร ได้จัดทำรูปเล่มปริญญานิพนธ์ตามหลักสูตรปริญญาตรี สาขาวิศวกรรมศาสตร์ หลักสูตรวิศวกรรมอิเล็กทรอนิกส์ ซึ่งในการนี้ข้าพเจ้าได้แก้ไขเนื้อหาและจัดทำรูปเล่มตามข้อกำหนดของรูปเล่มปริญญานิพนธ์เรียบร้อยแล้ว จึงขอให้อาจารย์ตรวจสอบ และรับรองความถูกต้องเหมาะสมของปริญญานิพนธ์ในครั้งนี้ด้วย

อาจารย์รับรองรูปเล่มปริญญานิพนธ์

1.อาจารย์ ว่าที่ร้อยตรี ศิลา ศิริมาสกุล ลงชื่อ .

2.อาจารย์ ผศ.ดร.มนตรี ไชยชาญยุทธ์ ลงชื่อ .

3.อาจารย์ที่ปรึกษา อาจารย์พิมล ผลพฤกษา ลงชื่อ .

4.อาจารย์ที่ปรึกษาร่วม ผศ.ดร.เกษมสุข เสพสิริสุข ลงชื่อ .