บทที่ 2
แนวคิดและทฤษฎีที่เกี่ยวข้อง

2.1 การประมวลผลภาพ

การประมวลผลภาพ คือการเอาภาพมาประมวลผล โดยคำนวณด้วยคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์รู้ว่าภาพนั้นคือภาพอะไรใช่สิ่งที่เราต้องการหรือไม่

การมองเห็นของมนุษย์เป็นสิ่งที่สำคัญและเป็นกลไกการรับภาพที่ซับซ้อนอย่างหนึ่ง ซึ่งจะให้ข้อมูลที่มีความจำเป็นสำหรับใช้ในงานง่าย ๆ (ตัวอย่างเช่น การจดจำวัตถุ)และสำหรับงานที่มีความซับซ้อน(ได้แก่ การวางแผน การตัดสินใจ การค้นคว้าทางวิทยาศาสตร์ การพัฒนาทางด้านความคิด) รูปภาพมีบทบาทมากสำหรับองค์กรต่าง ๆ เช่น หนังสือพิมพ์ โทรทัศน์ ภาพยนตร์ซึ่งได้ใช้ภาพ (ภาพนิ่ง ภาพเคลื่อนที่)เป็นสื่อนำเสนอข้อมูลข่าวสารต่าง ๆ สิ่งที่น่าสนใจของข้อมูลที่เกี่ยวกับการมองเห็นหรือข้อมูลภาพนั้นก็คือกระบวนการประมวลผลภาพ (Image Processing) โดยใช้ดิจิตอลคอมพิวเตอร์

Digital image processing จะเกี่ยวกับการแปลงข้อมูลภาพให้อยู่ในรูปแบบข้อมูลดิจิตอล (Digital format) ซึ่งสามารถที่จะนำเอาข้อมูลนี้จัดผ่านกระบวนการต่าง ๆ ด้วยดิจิตอลคอมพิวเตอร์ได้ ในระบบของดิจิตอล อินพุตและเอาต์พุตของระบบจะอยู่ในรูปแบบดิจิตอลเท่านั้น

Digital image analysis จะเกี่ยวกับวิธีการอธิบายและการจดจำข้อมูลภาพดิจิตอล ซึ่งอินพุตของระบบจะเป็นข้อมูลภาพดิจิตอลและเอาพุตจะเป็นเครื่องหมายที่ใช้แทนข้อมูลภาพดิจิตอลเหล่านั้น ในการวิเคราะห์ภาพมีอยู่หลายวิธีด้วยกันที่ได้นำมาจากการทำงานของตามนุษย์(human vision)นั่นก็คืองานทางด้าน Computer Vision เป็นลักษณะเดียวกับ Digital image analysis นั่นเอง การมองเห็นของมนุษย์นับว่าเป็นกระบวนการที่ซับซ้อนซึ่งลักษณะเทคนิคโดยทั่ว ๆ ไปในกระบวนการ Digital image analysis และ Computer Vision จะค่อนข้างซับซ้อนเช่นกัน

การคำนวณนั้นมีหลายวิธี เช่นการนำเอาจุดสีแต่ละ Pixel มาคำนวณหรือการนำหลาย ๆ จุดมารวมกันเป็นบริเวณกว้างขึ้น เช่น การดูPattern ของวัตถุ การวิเคราะห์หารูปร่างของวัตถุ เพื่อหาค่าอะไรบางอย่างที่ช่วยให้เข้าใจได้ว่า ภาพนั้นมีลักษณะอย่างไร

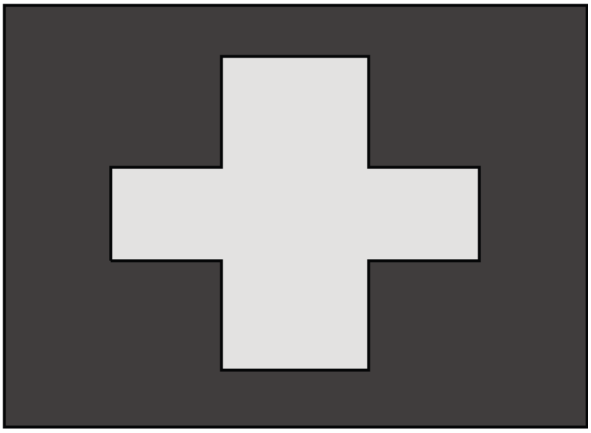
2.1.1 Pixel

Pixel คือ พื้นที่เล็กๆจุดหนึ่งในภาพ โดยในแต่ละจุดนั้นจะมีค่าตัวเลขกำกับ ซึ่งตัวเลข เหล่านี้จะมาจากค่าของแม่สีสามสี R (สีแดง) G (สีเขียว) B (สีฟ้า) ใช้บอกระดับความเข้มของแต่ละ เฉดสี หากมี Pixel หลายๆจุดมาต่อกันจะกลายเป็นภาพซึ่งมีขนาด จำนวน Pixel ด้านกว้าง X จำนวน Pixel ด้านยาว ยกตัวอย่าง เช่น รูปภาพขนาด 800 x 600 pixels หมายความว่ารูปภาพนี้มีความกว้าง 800 pixels และมีความยาว 600 pixels เป็นต้น

2.1.2 ระดับเทา (Gray level)

ระดับเทา (Gray Level) เป็นค่าซึ่งระบุความสว่างหรือความเข้ม ซึ่งมีค่าตั้งแต่ 0-255 (0 คือ ระดับเข้ม 255 คือระดับสว่าง) รวมทั้งพิกัดแนวนอนและแนวตั้ง ซึ่งใช้ระบุตำแหน่งในแถวลำดับ ภาพ (Image Array) เช่นจากรูปตัวอย่างที่ 2.1 และ 2.2 จุดภาพแถวนอนที่ 3 และแนวตั้งที่ 2 ในภาพที่ 2.2 ซึ่งมีค่าระดับเทา 40 วิธีการหาค่าระดับเทา (Gray Level) สามารถทำได้ดังสมการ 2.1

$$\text{Gray Level} = \frac{R+G+B}{3} \qquad (2.1)$$



ภาพที่ 2.1 ภาพที่ใช้ในการทดสอบระดับเทา

40	40	40	40	40	40	40	40	40	40
40	40	40	40	200	200	40	40	40	40
40	40	40	40	200	200	40	40	40	40
40	40	200	200	200	200	200	200	40	40
40	40	40	40	200	200	40	40	40	40
40	40	40	40	200	200	40	40	40	40
40	40	40	40	40	40	40	40	40	40

ภาพที่ 2.2 ระดับเทาของ pixel ในแถวแนวนอนที่ 3 และแถวแนวตั้งที่ 2

2.1.3 การแปลงภาพสีเป็นภาพขาว-ดำ

การแปลงภาพสีให้เป็นภาพขาว-ดำ (Thresholding) เป็นกระบวนการแปลงภาพสีให้มีการ แสดงผลได้แค่ 2 ระดับ คือ ขาว และดำ โดยจะแปลงข้อมูลภาพให้เป็นภาพ binary (Binary Image) มีกระบวนการแปลงภาพที่มีความเข้มหลายระดับ (Multilevel Image) ให้เป็นภาพที่มีความเข้มเพียง 2 ระดับ หรือ 1 บิต (bit) คือ 0 และ 1 โดย 0 แทนด้วยจุดที่มีภาพสีขาว และ 1 แทนด้วยจุดที่มีภาพสี ดำ Thresholding Technique คือการพิจารณาจุด pixel ในภาพว่าจุดใดควรจะเป็นจุดขาวหรือจุดใด ควรจะเป็นจุดที่มีค่าเท่ากับ 1 (จุดดำ) โดยจะทำการเปรียบเทียบค่าของแต่ละpixel (f(x,y)) กับค่าคงที่ ที่เรียกว่า Threshold (Threshold Value) เทคนิคนี้นิยมใช้กันมากในกรณีที่ความแตกต่างระหว่าง วัตถุ (Object) และพื้นหลัง (Background) ค่า pixel ในภาพที่มีค่าน้อยกว่าค่า Threshold จะถูก กำหนดเป็น 1 (จุดดำ) และถ้าค่าของ pixel ใด ๆ ในภาพมีค่ามากกว่าหรือเท่ากับค่า Threshold จะถูก กำหนดให้เป็น 0 (จุดขาว) ในการทำภาพ Binary โดยการทำ Thresholding ให้ได้ภาพดีและคมชัด ต้องเกิดจากการเลือกค่า Threshold ที่ถูกต้องและเหมาะสม ถ้าเลือกค่า Threshold ไม่เหมาะสม เช่น ค่า Threshold ที่มากหรือน้อยจนเกินไป ภาพที่ได้จะขาดความคมชัดหรืออาจทำให้รายละเอียดของ ภาพขาดหายไป หรือภาพที่ได้อาจจะมืดเกินไป หรือสว่างเกินไป หรืออาจจะเป็นภาพที่มีสิ่งรบกวน (Noise) เกิดขึ้น ทำให้ภาพผลลัพธ์ที่ได้ไม่ชัดเจน

2.1.4 ระบบสี RGB

ระบบสี RGB เป็นระบบสีที่เกิดจากการรวมกันของแสงสีแดง เขียวและน้ำเงินโดยมีการรวมกันแบบ Additive ซึ่งโดยปกติจะนำ ไปใช้ในจอภาพแบบ CRT (Cathode ray tube) ในการใช้งานระบบสี RGB ยังมีการสร้างมาตรฐานที่แตกต่างกันออกไปที่นิยมใช้งานได้แต่ RGBCIE และ RGBNTSC

2.1.4.1 ระบบสีแบบ RGB ของ CIE

เป็นระบบสีที่พัฒนาขึ้นโดย CIE (Commission International l ‘Eclairage) ซึ่งอ้างอิงสีด้วยสีแดงที่ 700nm สีเขียวเท่ากับ 546.1 nm และสีน้ำเงิน 435.8 nm

2.1.4.2 ระบบสีแบบ RGB ของ NTSC

เป็นระบบที่พัฒนาโดย NTSC (National Television System Committee) เพื่อใช้สำหรับการแสดงภาพของจอภาพแบบ CRT เป็นมาตรฐานสำหรับผู้ผลิตแบบ CRT ให้มีลักษณะเดียวกัน

2.1.5 ระบบสี HSV

ระบบสี HSV (Hue Saturation Value) เป็นการพิจารณาสีโดยใช้ Hue Saturation และ Valueซึ่ง Hue คือค่าสีของสีหลัก (แดง เขียวและน้ำเงิน) ในทางปฏิบัติจะอยู่ระหว่าง 0 และ 255 ซึ่งถ้า Hue มีค่าเท่ากับ 0 จะแทนสีแดงและเมื่อ Hue มีค่าเพิ่มขึ้นเรื่อย ๆ สีก็จะเปลี่ยนแปลงไปตามสเปกตรัมของสีจนถึง256 จึงจะกลับมาเป็นสีแดงอีกครั้ง ซึ่งสามารถแทนให้อยู่ในรูปขององศาได้

ดังนี้คือ สีแดง = 0 องศา สีเขียวเท่ากับ 120 องศา สีน้ำเงินเท่ากับ 240 องศา Hue สามารถคำนวณได้จากระบบสี RGB ได้ดังนี้

$$red_h = red - min(red,green,blue) \quad (2.2)$$

$$green_h = green - min(red,green,blue) \quad (2.3)$$

$$blue_h = blue - min(red,green,blue) \quad (2.4)$$

จากลักษณะโมเดลของระบบ Hue พบว่าจะมีค่าอย่างน้อยหนึ่งค่าที่จะเท่ากับ 0 แต่ถ้ามีสองค่าเท่ากับ 0 แล้ว hue จะเป็นมุมของสี(ค่าสี)มีค่าเป็นไปตามสี่ที่สามและถ้าทั้งสามสีมีค่าเท่ากับ 0 แล้วจะทำให้ไม่มีค่าของ Hue หรือสีที่ได้จะมีค่าเท่ากับสีขาวนั่นเอง ตัวอย่างเช่น จอภาพขาว-ดำ ถ้าเกิดมีสีใดสีหนึ่งมีค่าเท่ากับ 0 จะทำให้ค่าสีที่ได้เป็นไปตามสีที่เหลือ การให้น้ำหนักในการพิจารณาเมื่อสีแดงมีค่าเท่ากับ 0

$$\frac{(240\times blue_h)+(120\times green_h)}{blue_h+green_h} \quad (2.5)$$

Saturation คือความบริสุทธิ์ของสีซึ่งถ้า Saturation มีค่าเท่ากับ 0 แล้วสีที่ได้จะไม่มี Hue ซึ่งจะเป็นสีขาวล้วน แต่ถ้า Saturation มีค่าเท่ากับ 255 แสดงว่าจะไม่มีแสงสีขาวผสมอยู่เลย

Saturation สามารถคำนวณได้ดังนี้

$$Saturation = \frac{max(red,green,blue)-min(red,green,blue)}{max(red,green,blue)} \quad (2.6)$$

Value คือความสว่างของสี ซึ่งสามารถวัดได้โดยค่าความเข้มของความสว่างของแต่ละสีที่ประกอบกันสามารถคำนวณได้จาก

$$value = max(red,green,blue) \quad (2.7)$$

2.1.6 การตรวจหาตำแหน่งดวงตา

การตรวจหาตำแหน่งบริเวณดวงตาจะแสดงดังภาพที่ 2.3 ซึ่งได้แบ่งออกเป็นสองขั้นตอน ขั้นตอนแรกจะทำการหาบริเวณดวงตาจากค่าข้อมูลสี (Chrominance) และขั้นตอนที่สองเป็นการตรวจหาตำแหน่งบริเวณดวงตาจากค่าความสว่าง (Luminance) จากภาพสี YCbCr โดยตำแหน่ง

บริเวณดวงตาจากค่าข้อมูลสี (Chrominance) ได้จากการสังเกตค่าสูงสุดของค่าข้อมูลสี Cb และค่าต่ำสุด Cr ซึ่งหาได้จากสมการ ดังนี้

$$E_{MC} = \frac{1}{3}\{(C_b^2) + (C_r^{\sim 2}) + (C_b/C_r)\} \quad (2.8)$$

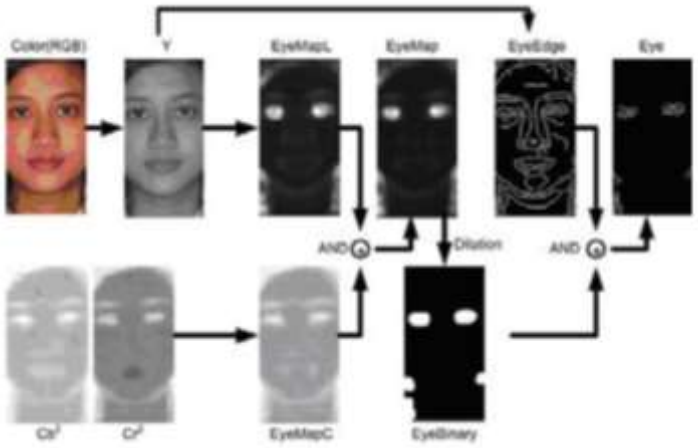
สมการ 2.8 คือการหา EyeMapC ดังภาพที่ 2.3 โดย (C2b),(C~r 2) และ (Cb/Cr) โดยทำการปรับค่าทั้งหมดให้อยู่ ในช่วง (0-255)และ C~r คือ ค่าลบของ (255- Cr) เนื่องจาก ตำแหน่งบริเวณดวงตา จะมีลักษณะของจุดภาพที่สว่างและมืด อยู่ในค่าความสว่าง (Luminance) ซึ่งเป็นคุณสมบัติของตัว ดำเนินการมอร์โฟโลยีค่าระดับเทาโดยการไดเลชันและอีโรชัน ซึ่งในงานวิจัยจะใช้ การไดเลชัน และ อีโรชันค่าระดับเทาร่วมกับ 8 ส่วนประกอบโครงร่างแบบ hemispheric ซึ่งตำแหน่งบริเวณดวงตาที่ตรวจพบจากค่าความสว่าง หาได้จากสมการดังนี้

$$\mathrm{E_{ML}} = \frac{\mathrm{Y(x,y)} \oplus \mathrm{g\sigma(x,y)}}{\mathrm{Y(x,y)} \Theta \mathrm{g\sigma(x,y)} + 1} \quad (2.9)$$

สมการ 2.9 คือการหา EyeMapL ดังภาพที่ 2.3 เมื่อตัวดำเนินการไดเลชันและอีโรชัน ค่าระดับเทา อยู่บน ฟังก์ชันพื้นฐาน f:F $\subset$ R2 -> Rโดยใช้ส่วนประกอบโครงร่างซึ่งสามารถเขียนเป็นฟังก์ชันได้ดังนี้ g:G $\subset$ R2 -> R เพื่อทำให้ภาพตำแหน่งบริเวณดวงตาที่ตรวจพบจากค่าข้อมูลสีมีคุณภาพดีขึ้น จะน้าภาพตำแหน่งบริเวณดวงตาที่ตรวจพบ จาก ค่าข้อมูลสีไปทำการประมวลผลภาพด้วยฮิสโตรแกรมอีควอไลเซชัน เพื่อให้ระดับความเข้มของแสงภาพมีการกระจายอย่างสม่ำเสมอแล้วน้าภาพที่ได้จากการประมวลผลไป AND เข้ากับ ภาพตำแหน่งบริเวณดวงตาที่ตรวจพบจากค่าความสว่าง ซึ่งหาได้จากสมการ ดังนี้

$$\mathrm{EMAP} = (\mathrm{EMC})\mathrm{AND}(\mathrm{EML}) \quad (2.10)$$

สมการ 2.10 คือกระบวนการทำ EyeMap ดังภาพที่ 2.3 ซึ่งภาพผลลัพธ์ที่ได้จากการตรวจหาตำแหน่งบริเวณดวงตา เรายังสามารถทำการปรับค่าไดเลชัน เพื่อเพิ่มความสว่างให้กับ ตำแหน่งบริเวณดวงตา รวมถึงการปรับปรุงภาพโดยใช้เทคนิค การกำหนดค่าเทรสโชล และค่าไบนารี ที่เหมาะสมให้กับภาพ จะทำให้ภาพผลลัพธ์มีประสิทธิภาพมากยิ่งขึ้นเมื่อทำกระบวนการดังกล่าวแล้วจะได้ผลลัพธ์คือภาพ EyeBinary ดังภาพที่ 2.3 จากนั้นนำภาพ EyeEdge และ EyeBinary มาทำกระบวนการ AND ซึ่งจะได้ผลลัพธ์คือขอบของดวงตาเท่านั้น ซึ่งกระบวนการตรวจหาตำแหน่งดวงตาจะแสดงดังภาพที่ 2.3



ภาพที่ 2.3 การตรวจหาดวงตา

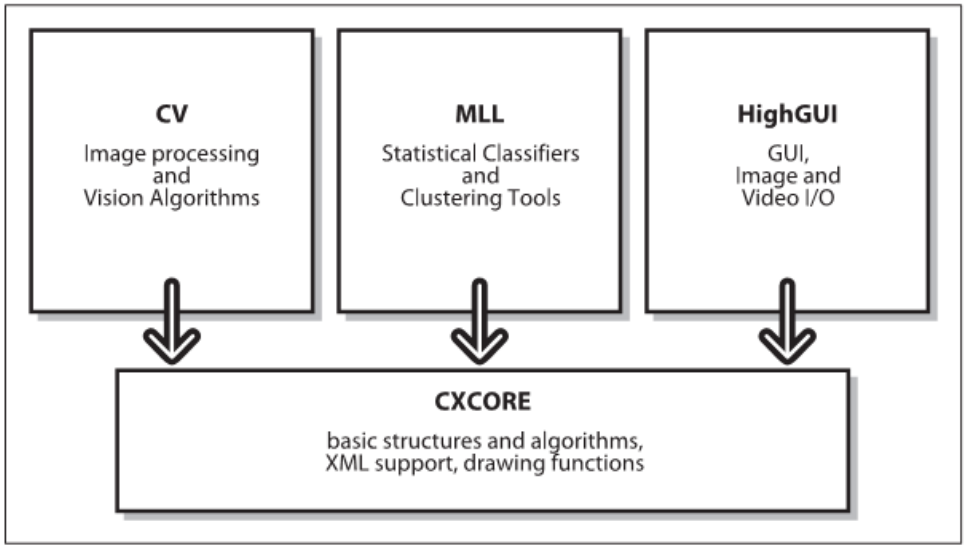
2.2 OpenCV

OpenCV ย่อมาจาก Open Source Computer Vision ซึ่ง OpenCV เป็นไลบรารี่แบบ ครอสแพลตฟอร์ม (Cross-platform) โดยสามารถรันบนระบบปฏิบัติการลินุกซ์ (Linux), วินโดวส์ (Windows) และแมคโอเอสเอ็ก (Mac OS X) OpenCV นั้นถูกออกแบบมา สำหรับการคำนวณที่มีประสิทธิภาพอย่างยิ่งสำหรับแอพพลิเคชั่นที่ทำงานแบบเรียลไทม์ OpenCV นี้ถูกเขียนขึ้นโดยภาษาซีที่ถูกออฟติไมซ์แล้วและสามารถใช้ข้อได้เปรียบจากโปรเซสเซอร์แบบมัลติคอร์

จุดประสงค์หนึ่งของ OpenCV ก็คือการทำให้โครงสร้างพื้นฐานของคอมพิวเตอร์วิชั่น นั้นง่ายต่อการใช้งาน ซึ่งช่วยให้ผู้ใช้สามารถสร้างวิชั่นแอพพลิเคชั่น (Vision application) ที่ค่อนข้างซับซ้อนได้อย่างรวดเร็ว OpenCV ไลบรารี่นั้นประกอบไปด้วยฟังก์ชันมากกว่า 500 ฟังก์ชัน ซึ่งครอบคลุมหลายๆหัวข้อของวิชั่น รวมถึงการตรวจสอบสินค้าจากโรงงาน การประมวลผลภาพในทางการแพทย์ ระบบรักษาความปลอดภัย ส่วนติดต่อกับผู้ใช้ การปรับเทียบกล้องสเตอริโอวิชั่น และการใช้งานกับหุ่นยนต์ และเนื่องจากคอมพิวเตอร์วิชั่น และแมตชีนเลินนิ่ง (Machine Learning) นั้นมักจะต้องทางานไปด้วยกันบ่อยครั้ง OpenCV จึงมีแมตชีนเลินนิ่งไลบรารี่ รวมอยู่ด้วย ซึ่งไลบรารี่ย่อยนี้ เน้นไปที่การทำแพทเทินเรคคอกนิชั่น (Pattern recognition) แบบสถิติ และการทำคลัสเตอริ่ง (Clustering) ซึ่งแมตชีนเลินนิ่งไลบรารี่นั้นสามารถใช้ได้เป็นอย่างดีกับงานที่เกี่ยวกับวิชั่น ซึ่งสิ่งนี้คือหัวใจหลักของ OpenCV และโดยทั่วไปแล้วเพียงพอที่จะใช้ในปัญหาของแมตชีนเลินนิ่งต่างๆได้

2.2.1 โครงสร้างของ OpenCV และองค์ประกอบ

OpenCV นั้นมีโครงสร้างอยู่ห้าส่วนประกอบหลัก สี่ในห้าแสดงดังรูปต่อไปนี้โดยส่วนซีวี (CV) ประกอบไปด้วยการประมวลผลภาพพื้นฐาน และคอมพิวเตอร์วิชั่นอัลกอริทึมระดับสูงกว่า และแอ็มแอล (ML) คือแมตชีนเลินนิ่งไลบรารี่ซึ่งรวมถึงสแตทิสทิคอลคลาสสิไฟลเออร์ (Statistical classifiers) และคลัสเตอริ่งทูลในส่วนของไฮจียูไอ (HighGUI) จะประกอบไปด้วยการทางานกับอินพุต เอาท์พุทและฟังก์ชันสาหรับการจัดเก็บและโหลดวีดีโอและภาพและ ซีเอ็กคอร์ (CXCore) ประกอบไปด้วยโครงสร้างข้อมูล (Data structures) และคอนเท็น (Content) พื้นฐาน ภาพที่ 2.4 แสดงโครงสร้างของ OpenCV



ภาพที่ 2.4 โครงสร้างของ OpenCV

2.2.2 ระบบปฏิบัติการและสถาปัตยกรรมที่รองรับ

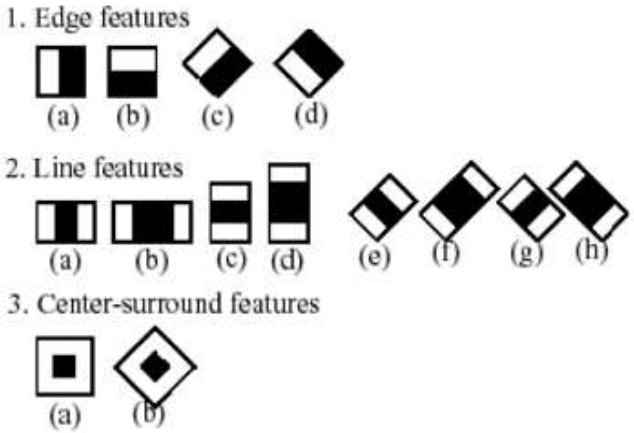
OpenCV นั้นถูกออกแบบมาให้สามารถใช้งานได้ง่าย เดิมทีนั้น OpenCV ถูกเขียนขึ้นและคอมไพล์ผ่าน Borland C++, Microsoft Visual C++ และอินเทลคอมไพล์เลอร์ซึ่งการใช้ภาษาC และC++ ในการโค้ดเป็นมาตรฐานทาให้ OpenCV สามารถสนับสนุนการครอสแพลตฟอร์มได้ง่ายขึ้น รูปต่อไปนี้แสดงถึงแฟลตฟอร์มที่ OpenCV สามารถทำงานได้ เริ่มจากระบบ IA32 บนวินโดวส์ตามด้วยลินุกซ์ บนสถาปัตยกรรมเดียวกัน บนแมคโอเอสเอ็ก นั้นมีความสำคัญตั้งแต่แอปเปิล ได้เริ่มใช้โปรเซสเซอร์ของอินเทลตามด้วยสถาปัตยกรรมแบบ 64 บิต และบนซันฮาร์ดแวร์และบนระบบปฏิบัติการอื่นๆ โดยภาพที่ 2.5 แสดงถึงระบบปฏิบัติการและสถาปัตยกรรมที่ OpenCV สนับสนุน

	IA32	EM64T	IA64	Other (PPC, Sparc)
Windows	✓ (w. IPP; MSVC6, .NET2005+OMP, ICC, GCC, BCC)	✓ (w. IPP; MSVC6+PSDK.NET2005+OMP, PSDK)	± (w. IPP; PSDK, some tests fail)	N/A
Linux	✓ (w. IPP; GCC, BCC)	✓ (w. IPP; GCC, BCC)	✓ (GCC, ICC)	✗
MacOSX	✓ (w. IPP, GCC, native APIs)	? (not tested)	N/A	✓ (iMac G5, GCC, native APIs)
Others (BSD, Solaris…)	✗	✗	✗	Reported to build on UltraSparc Solaris

ภาพที่ 2.5 ระบบปฏิบัติการและสถาปัตยกรรมที่ OpenCV สนับสนุน

2.3 Haar like-feature

Haar like-Features เป็นวิธีการตรวจจับและตีความวัตถุภายในภาพ โดยใช้การสร้างรูปเหลี่ยม (Feature) โดยที่ภาพนี้แสดงถึงผลต่างระหว่างพื้นที่ส่วนสีขาว และส่วนที่เป็นสีดำ ซึ่งรูปเหลี่ยมที่สร้างขึ้นสามารถเปลี่ยนแปลงขนาด และตำแหน่งได้ ใช้สำหรับการตรวจจับลักษณะบนภาพแบบต่าง เช่น เส้นตรง, วงกลม เป็นต้น ภาพที่ 2.6 แสดงรูปแบบของรูปเหลี่ยมสำหรับการตรวจจับลักษณะแบบต่างๆ

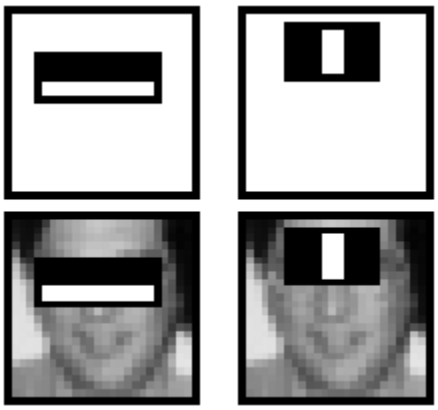


ภาพที่ 2.6 รูปแบบของรูปเหลี่ยมสำหรับการตรวจจับลักษณะแบบต่างๆ

(1) ความสามารถของขอบ

(2) ความสามารถของเส้น

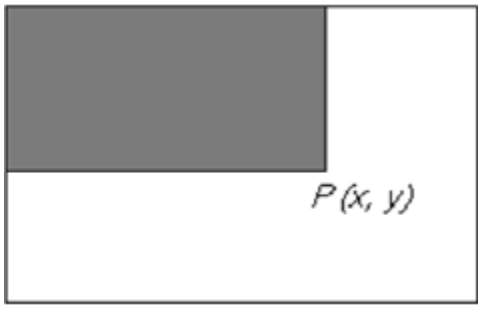
(3) ความสามารถของบริเวณที่ล้อมรอบจุดตรงกลาง



ภาพที่ 2.7 ตัวอย่างการใช้รูปเหลี่ยมตรวจจับลักษณะต่างๆ

การคำนวณค่าของรูปเหลี่ยม (feature) นั้น ใช้หลักการคำนวณแบบ Integral image คือ ผลรวมของค่าในทุกๆ จุดภาพ ที่ตำแหน่ง (x, y) ใดๆ ดังสมการที่ 2.11 ภาพที่ 2.8 แสดงการคำนวณแบบ Integral image

$$P(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \qquad (2.11)$$



ภาพที่ 2.8 การคำนวณแบบ Integral image

ในการทำ Haar like-feature นั้น จำเป็นต้องมีภาพตัวอย่างจำนวนมาก ซึ่งใช้ในการคัดเลือกลักษณะของรูปต้องการตรวจจับและตีความหมาย ซึ่งมีสองลักษณะคือ Positive Image หรือรูปมีวัตถุนั้นๆประกอบอยู่ภายในภาพ และ Negative Image หรือภาพใดๆที่ไม่มีวัตถุที่เราต้องการอยู่ภายในภาพ จากนั้นใช้ขั้นตอนวิธีของ AdaBoost (Adaptive Boost) ซึ่งเป็นกระบวนการหารูปเหลี่ยมที่มีลักษณะใกล้เคียง และแตกต่างกับภาพนำเข้า สำหรับการจัดประเภทของภาพ โดยการถ่วง

น้ำหนักให้ส่วนต่างๆภายในภาพ บนภาพ Positive และภาพ Negative เพื่อใช้หาลักษณะของวัตถุที่ “ใช่” และ “ไม่ใช่” ในลักษณะต่างๆ

2.4 Region of interest (ROI)

Region of interest (ROI) คือบริเวณที่เราสนใจซึ่งอาจจะเป็นบริเวณใดภายในภาพก็ได้ โดยการตีกรอบล้อมรอบบริเวณที่สนใจ ด้วยวงกลม กรอบสี่เหลี่ยม หรือกรอบรูปเหลี่ยมใดๆ เพื่อนำภาพเฉพาะส่วนดังกล่าวมาประมวลผลหรือเปลี่ยนแปลงภาพตามต้องการ โดยไม่มีผลกระทบกับส่วนอื่นๆซึ่งภายในหนึ่งภาพ สามารถกำหนดได้หลายๆ บริเวณที่สนใจ การตัดเฉพาะส่วนของภาพเพื่อนำมาประมวลผลทำให้การประมวลผลภาพเร็วขึ้นเนื่องจาก Resolution ของภาพที่นำมาประมวลผลมีขนาดเล็กลง ซึ่งในการประมวลผลภาพขนาดใหญ่จะทำให้เห็นความแตกต่างของประสิทธิภาพในการประมวลผลมาก ภาพที่ 2.9 แสดงตัวอย่างการกำหนดบริเวณที่สนใจ

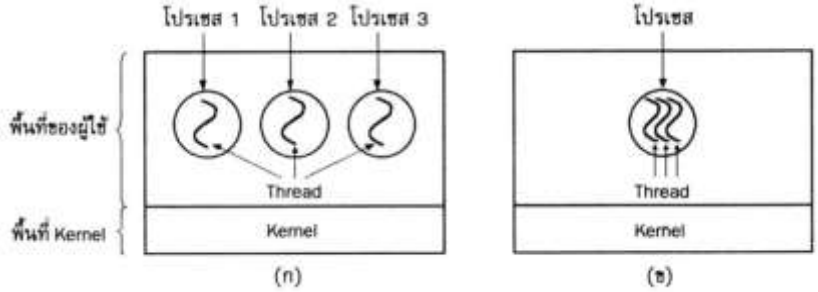
ภาพที่ 2.9 ตัวอย่างการกำหนดบริเวณที่สนใจ

2.5 Circle Detection

Circle Detection คือการตรวจจับวงกลมในรูปภาพเพื่อใช้ในการตรวจหาตาดำ โดยจะใช้ภาพ Grey Scale เพื่อใช้ในการตรวจจับวงกลมโดยการตรวจจับวงกลมจะดูความลึกของสีซึ่งลักษณะของนัยน์ตาอาจมีสีดำหรือสีฟ้าซึ่งเมื่อแปลงเป็น Grey Scale แล้วจะทำให้มีระดับความลึกของสีอยู่ในระดับที่แตกต่างกันกับของตาขาวทำให้สามารถใช้การตรวจหาวงกลมเพื่อหานัยน์ตาได้ โดย EmguCV มี Library ที่สำเร็จรูปในการตรวจหาวงกลมซึ่งสามารถกำหนด Parameter 4 ค่าคือ ค่า Threshold ที่ใช้ในการทำ Canny Edge Detection, ค่า Threshold ที่ใช้ในการทำ Accumulator, ค่าอัตราส่วนของภาพที่จะทำการตรวจหาวงกลม, ค่าระยะของรัศมีในการตรวจหามากสุดและน้อยสุดของวงกลม, และค่าระยะห่างของวงกลมที่ตรวจสอบพบกับวงกลมถัดไป ซึ่งในการตรวจสอบนัยน์ตาจะต้องกำหนดค่าของรัศมีวงกลมให้พอเหมาะจึงจะสามารถตรวจหานัยน์ตาได้อย่างแม่นยำ

2.6 Multithread Processing

Process ดั้งเดิมที่เรียกว่า “Heavy Weight” ที่มีการควบคุมเพียง 1 Thread หมายถึงภายใน 1 Process จะควบคุมงานเพียง 1 งานเท่านั้น แต่ในระบบปฏิบัติการสมัยใหม่ในแต่ละ Processสามารถมีได้หลายThread อาจจะกล่าวได้ว่า Thread คือส่วนประกอบย่อยของ Process นั่นเอง ซึ่งอาจเรียก Thread ว่า “Lightweight Process” ในภาพที่ 2.10 (ก) แสดงโปรเซส 3 Process แต่ละ Process จะมี Address เป็นของตนเอง และควบคุมเพียง 1 งานเท่านั้น แต่ละภาพที่ 2.10 (ข) จะเห็นว่าในแต่ละ Process จะควบคุม 3 Thread โดยใช้ Address เดียวกันอยู่

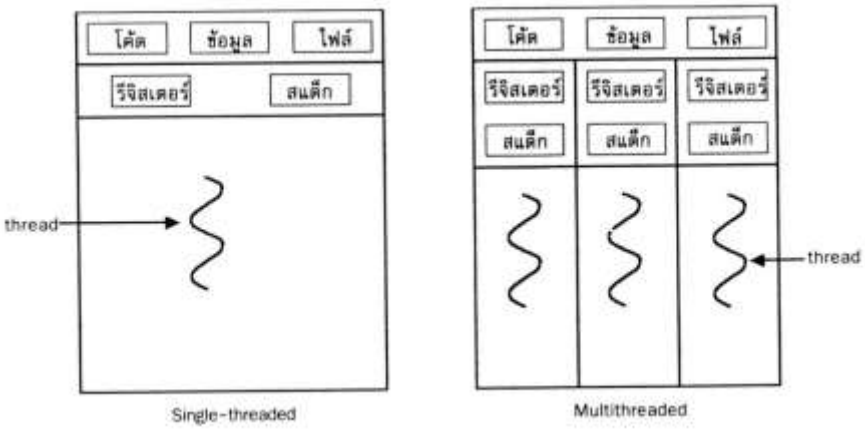


ภาพที่ 2.10 ตัวอย่างการทำงานแบบ Multithread

(ก) การทำงาน 3 Process

(ข) การทำงาน 1 Process

Thread เป็นหน่วยพื้นฐานของการจัดสรรการใช้ประโยชน์ของของซีพียู Process จะประกอบด้วย Thread จะมีการแชร์โค้ด, ข้อมูล และ Resource เช่น ไฟล์, อุปกรณ์ต่างๆเป็นต้น แต่ถ้า Process มีหลาย Thread หรือ Multithread จะทำงานได้หลายงานในเวลาเดียวกัน Process ที่เป็น Single-Threaded และ Multithread ภาพที่ 2.11 แสดงโครงสร้างของ Single-Threaded และ Multithreaded



ภาพที่ 2.11 โครงสร้างของ Single-Threaded และ Multithreaded

Software ปัจจุบันที่ทำงานกับระบบปฏิบัติการสมัยใหม่มีการออกแบบให้รองรับ Multithread โดยแยกออกเป็น Process ที่ควบคุมหลายๆ Thread เช่น Web browser ที่มี Thread หนึ่งในการแสดงรูปภาพหรือเขียนข้อความในขณะที่อีก Thread หนึ่งกำลังดึงข้อมูลจาก Network หรืออย่างในโปรแกรมWord Processing ที่มีหลาย Thread โดยที่ Thread หนึ่งกำลังแสดงภาพกราฟฟิก ในขณะที่ Thread ที่สองกำลังรอรับคำสั่งจากคีย์บอร์ดจากผู้ใช้ ในขณะที่ Threadที่สามกำลังตรวจสอบคำสะกดและไวยกรณ์ในลักษณะ Background เป็นต้น

การที่ระบบปฏิบัติการสนับสนุนระบบ Multithread ทำให้มีข้อได้เปรียบ 4 ข้อหลักๆ ดังนี้

1) การตอบสนองในระบบ Multithread ที่โต้ตอบ Application จะยินยอมให้ Program ยังคงดำเนินต่อไปถึงแม้ว่าจะมีบางส่วนรอการทำงานอยู่ เช่น มีการปฏิบัติที่ยาวนานเนื่องจากการเพิ่มการ โต้ตอบกับผู้ใช้นั่นเอง ยกตัวอย่างเช่น โปรแกรมเว็บบราวเซอร์ยังคงโต้ตอบกับผู้ใช้ได้ ในขณะที่มีหนึ่งที่กำลังโหลดรูปภาพอยู่
2) การแชร์ทรัพยากรของ Multithread จะแชร์หน่วยความจำ และทรัพยากรของ Process อยู่แล้ว ข้อได้เปรียบ ของการแชร์ Code จะทำให้ Application สามารถมีกิจกรรมของ Thread ได้หลายๆกิจกรรมภายใน Address เดียวกัน
3) ความประหยัดของ Multithread การจัดสรรหน่วยความจำและทรัพยากรสำหรับการสร้าง Process มีค่าใช้จ่ายมาก ในทางตรงข้าม เนื่องจาก Thread แชร์ทรัพยากรของ Process ที่มันอาศัยอยู่แล้ว ทำให้เกิดการประหยัดในการสร้าง Thread และทำการ Context switching ของ Thread เป็นการยากที่จะวัดความแตกต่างระหว่างการสร้างและคงสภาพ Process ที่มีมากกว่าThread ได้ แต่โดยปกติแล้วจะใช้เวลาในการสร้างและคงสภาพ Process สูงกว่าเวลาที่ใช้กับ Thread ใน Solaris2 การสร้าง Process จะช้ากว่าการสร้าง thread 30 เท่าและ Context switching Process จะช้ากว่า Thread 5 เท่า
4) การเอื้อประโยชน์ของสถาปัตยกรรม Multiprocessing ช่วยเสริมสถาปัตยกรรม Multiprocessing ให้สูงขึ้น ในขณะที่แต่ละ Thread สามารถประมวลผลขนานกันไปใน Process อื่นได้ใน Process ที่มี Thread เดียวสามารถประมวลผลเพียงซีพียูเดียวเท่านั้น ไม่ว่าจะมีกี่ซีพียูก็ตามระบบ Multithread ในเครื่องที่มีหลายซีพียูจะเพิ่มประสิทธิภาพในการทำงานพร้อม ๆ กันได้มากขึ้น ในสถาปัตยกรรมที่มีซีพียูเดียว ซีพียูจะย้ายแต่ละ Thread ให้เร็วมากขึ้นเพื่อให้ทำงานเสมือนว่าขนานกันอยู่ แต่ในความเป็นจริงจะรันเพียง thread เดียวเท่านั้นในเวลานั้น ๆ

2.7 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP, ทีซีพี) เป็นหนึ่งในโพรโทคอลหลักในเครือข่ายอินเทอร์เน็ต หน้าที่หลักของทีซีพี คือ ควบคุมการรับส่งข้อมูลระหว่าง Host ถึง Host ในเครือข่ายเพื่อใช้แลกเปลี่ยนข้อมูลระหว่างกัน โดยตัวโพรโทคอลจะรับประกันความถูกต้อง และลำดับของข้อมูลที่ส่งผ่านระบบเครือข่าย นอกจากนั้นทีซีพียังช่วยจำแนกข้อมูลให้ส่งผ่านไปยังแอปพลิเคชัน ที่ทำงานอยู่บนโฮสเดียวกันให้ถูกต้องด้วย

งานหลักที่สำคัญของทีซีพีอีกงานหนึ่งคือ เป็นโพรโทคอลที่ขั้นกลางระหว่างแอปพลิเคชันและเครือข่ายไอพี ทำให้แอปพลิเคชันจากโฮสหนึ่ง สามารถส่งข้อมูลออกยังอีกโฮสหนึ่งผ่านเครือข่ายเปรียบเสมือนมีท่อส่งข้อมูลระหว่างกัน

TCP มีลักษณะดังต่อไปนี้

2.7.1 Connection-Oriented

ก่อนส่งข้อมูล จะเกิดสองกระบวนการนี้ใน Application Layer ต้องมีการติดต่อก่อนเชื่อมต่อ การใช้กระบวนการการสร้างการเชื่อมต่อ TCP และการปิดการเชื่อมต่ออย่างเป็นทางการ สำหรับคำแนะนำ TCP มากกว่าเกี่ยวกับกระบวนการการเชื่อมต่อ TCP ดูในบทที่11 โปรโตคอลหน่วยควบคุม “Transmission (TCP) Connections”

2.7.2 การส่งแบบสองทาง (Full duplex)

สำหรับ TCP แต่ละชั้น การเชื่อมต่อ TCP ประกอบด้วยการเชื่อมต่อทางลอจิก ข้อมูลปลายทางและข้อมูลที่เข้ามา จะใช้เทคโนโลยีที่ส่งข้อมูลได้สองทางพร้อมกัน ก็คือข้อมูลสามารถไหลออกมาจากปลายทางและเข้าไปสายส่งข้อมูลที่เข้ามาพร้อมกัน TCP header บรรจุหมายเลขลำดับข้อมูลต้นทางและปลายทางและ acknowledgment ของข้อมูลที่เข้ามา

2.7.3 ความน่าเชื่อถือ(Reliable)

ข้อมูลที่เชื่อมต่อใน TCP จะต้องถูกตามลำดับและมีสัญญาณ positive acknowledgment จากเครื่องรับ ถ้าไม่ใช่ positive acknowledgment TCP ก็จะทำการส่งข้อมูลอีกครั้ง ที่เครื่องรับ จะทำสำเนาส่วนถูกทิ้งและที่มาถึงออกมาจากข้อมูลตามลำดับและเก็บข้อมูลเรียงลำดับไห้ถูกต้อง และมี TCP checksum ใช้ในการตรวจสอบความถูกต้องของข้อมูล

2.7.4 ความต่อเนื่องของข้อมูล(Byte stream)

ข้อมูลที่ส่งออก และรับเข้ามาผ่านสายส่งสัญญาณมีความต่อเนื่องของข้อมูลแต่ละ Byte มาก หมายเลขลำดับและหมายเลข Acknowledgment ในTCP header ที่ถูกกำหนดในรอยต่อของแต่ละ byte

TCP ไม่มีบันทึกหรือข้อความบอกความต่อเนื่องของข้อมูล โปรโตคอล Application Layer ต้องแยกตามความเหมาะสมของข้อมูลขาเข้าแต่ละ byte

2.7.5 ผู้ส่ง-และผู้รับ (Sender- and receiver-side flow control)

เพื่อหลีกเลี่ยงการส่งข้อมูลที่เยอะจนทำให้เกิดการแออัดภายใน เราเตอร์(Router) ของเครือข่าย TCP ดำเนินการส่งควบคุมการไหลของข้อมูลที่ค่อยทำการส่งข้อมูลในแต่ละครั้ง เพื่อหลีกเลี่ยงผู้ส่งส่งข้อมูลที่ไม่ได้รับ บัฟเฟอร์ TCP จะทำกระบวนการ flow control ทางผู้รับแสดงจำนวนไบต์ที่รับจะสามารถรับข้อมูลได้

2.7.6 การแบ่งส่วนของแอพลิเคชั่นเลเยอร์ดาต้า (Segmentation of Application Layer data)

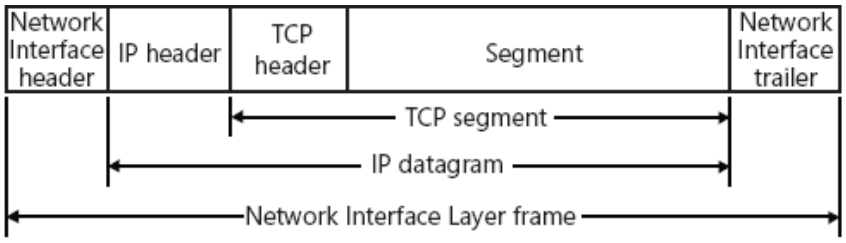
TCP มีการแบ่งข้อมูลที่ได้รับจากชั้น Application คือการแบ่งข้อมูลแต่ละ layer ให้เหมาะสมจนถึงการส่ง IP datagram ภายในเครือข่าย ภายในชั้น TCP แสดงถึงขนาดที่มากที่สุดของแต่ละ segment และในการรับข้อมูลและยังมีการปรับขนาดที่ดีที่สุดเพื่อจะได้ค้นพบเส้นทางที่เหมาะสมในการส่ง

2.7.7 การส่งแบบหนึ่งต่อหนึ่ง (One-to-one delivery)

TCP มีการเชื่อมต่อทางลอจิกแบบจุดต่อจุดระหว่าง โปรโตคอลในชั้น applications layer TCP จะไม่มีการจัดส่งข้อมูลแบบ จุด ต่อ หลายจุด TCP ใช้เมื่อ Applications layer ต้องการส่งข้อมูลที่มีความเชื่อถือได้

2.7.8 การแบ่งย่อยของข้อมูล (The TCP Segment)

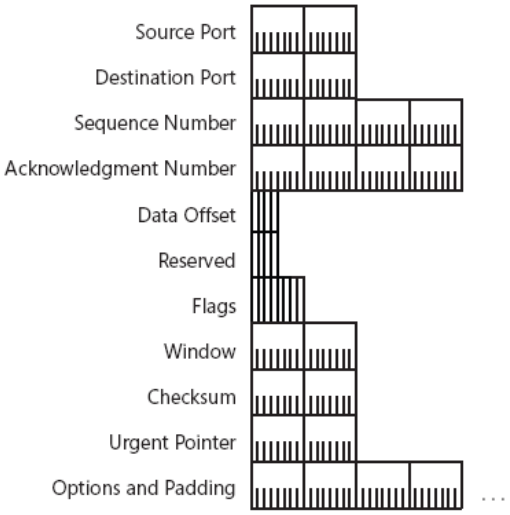
ประกอบด้วย TCP header และการเลือก payload แต่ละ segment ซึ่งมีการระบุไว้ใน IPV6 TCP Segment สามารถส่งข้อมูลได้มากที่สุด 65,495 bytes IP headerขนาดเล็กที่สุด 20 bytes ซึ่งส่งผลให้เกิดการ Encapsulated ที่เหมาะสม ในชั้น Network Interface ส่วนหัวและส่วนท้ายข้องเฟรม แสดงในรูปภาพ ภาพที่ 2.12 แสดงส่วนประกอบของ TCP header



ภาพที่ 2.12 ส่วนประกอบของ TCP header

2.7.9 The TCP Header

ขนาดของ TCP header ประกอบด้วย ส่วนต่างๆที่แสดง ในภาพที่ 2.13 เมื่อ TCP ปัจจุบันของ TCP header มีความยาว 20 ไบท์ ภาพที่ 2.13 แสดงโครงสร้างของ TCP header



ภาพที่ 2.13 โครงสร้างของ TCP header

2.7.9.1 Source Port

ขนาด 2 byte เป็นส่วนที่บอกถึงพอร์ต address ของผู้ส่งซึ่งมีไม่ซ้ำกัน

2.7.9.2 Destination Port

ขนาด 2 byte เป็นส่วนที่ โพรโตคอลชั้น application แสดงถึง ปลายทางในการส่งข้อมูล ซึ่งรวม IP address ใน IP header และ destination port ใน TCP header ซึ่งจะบอกถึงต้นทาง *socket*—a ซึ่งเป็นตำแหน่งที่ส่งจะมีเพียงแอดเดรสเดียวในโลก

2.7.9.3 Sequence Number

4 byte ระบุถึงลำดับที่ทำการส่งไบต์แรกของ segment Sequence Number จะถูต้องค่าไว้เสมอ แม้แต่เวลาไม่มีการส่งข้อมูลใน Segment ในกรณีนี้ Sequence Number จะถูกตั้งไว้ในการส่งข้อมูล byte ต่อไป เมื่อมีการเชื่อมต่อ TCP segments และ Synchronization ค่าแฟลกที่ 1 จะถูกเซทเป็นค่าเริ่มต้นของ Sequence Number ซึ่งแสดงว่าเป็น byte แรกในการส่งข้อมูล

2.7.9.4 Acknowledgment Number

ขนาด 4-byte ซึ่งระบุหมายเลขลำดับของหมายเลขต่อไปที่จะทำการส่งไปให้ผู้รับ หมายเลข acknowledgment จะแจ้ง positive acknowledgment กลับมา แต่ไม่รวมถึงตัวเลข acknowledgment ที่ถูกรับ ตัวเลขacknowledgment มีความสำคัญใน TCP segments ทั้งหมดขึ้นอยู่กับการตั้งค่า flagของacknowledgment

2.7.9.5 Data Offset

ขนาด 4 bit เป็นพื้นที่บอกถึงจุดเริ่มต้นของ TCP แต่ละ segment และบอกถึงขนาดของ TCP Headerและ IP header ว่ามี Header length, Data Offset มีขนาด 32 บิต สำหรับ TCP Header ที่มีขนาดเล็ก (no options) Data Offset เซตไว้ที่ 5 (0x5), เป็นการชี้บอกข้อมูลเริ่มต้น

ใน byte ที่ 20 จากการเริ่มต้นของ TCP Segment กับส่วนของ Data Offset ตั้งค่ามากที่สุดไว้เป็น 15 (0xF), มากกว่า TCP Header รวมถึงในออฟชั่นต่าง ในTCP สามารถยาวได้ 60 ไบต์

2.7.9.6 Reserved

ขนาด 4 bit สงวนไว้ใช้สำหรับอนาคต ผู้ใช้ตั้งค่าบิตนี้ไว้ เป็น 0

2.7.9.7 Flags

ขนาด 8 บิต เป็นพื้นที่บอกถึงแฟลก ทั้ง8 ใน TCP ที่นิยามใน RFCs 793 และ 3168 ในแฟลกทั้ง 8 เรียกว่า CWR (Congestion Window Reduced), ECE (Explicit Congestion Notification [ECN]-Echo), URG (Urgent), ACK, PSH (Push), RST (Reset),SYN and FIN (Finish) ซึ่งจะกล่าวถึงในรายละเอียดในเรื่อง TCP flag

2.7.9.8 Window

ขนาด 2 byte เป็นส่วนที่บอกจำนวนไบต์ที่ส่งเข้ามาในแต่ละพื้นที่โดยการดูwindow size ทุก segment ผู้รับบอกผู้ส่งถึงจำนวนข้อมูลที่สามารถส่งได้และข้อมูลที่รับเสร็จแล้ว ผู้ส่งจะไม่ส่งข้อมูลที่มากกว่าผู้รับสามารถรับข้อมูลได้ ถ้าผู้รับไม่สามารถรับข้อมูลได้มากกว่านี้ก็จะบอกผู้ส่งว่า Window size มีขนาดเท่ากับ 0 ผู้ส่งจะไม่สามารถส่งข้อมูลได้จนกว่าขนาดของ Window size จะมีค่าไม่เท่ากับ 0 byte เครื่องมือที่สำคัญของการบอกว่า window size เป็น0 ก็คือ flow control

2.7.9.9 Checksum

ขนาด 2 byte เป็นบิตที่ให้บริการตรวจสอบข้อผิดพลาดของแต่ละเซกเม็นต์ (TCP header and segment) ค่าของ Checksum ถูกคำนวณเหมือนกับ IP header checksum ทั้งหมดมีขนาด 16 บิต ใน TCP pseudo header, TCP header, TCP segment และถ้าต้องการ padding เพิ่มจะได้ค่าเป็น 0x00 ไบต์ padding ใช้เฉพาะความยาวของ segment ที่มีย่อยมาก ค่าของ Checksum จะถูกตั้งไว้เป็น 0 ขณะที่มีการคำนวณ

2.7.9.10 Urgent Pointer

ขนาด 2 byte เป็นพื้นที่บอกว่าข้อมูลเร่งด่วนที่จะทำการส่งข้อมูล

2.7.9.11 Options

หรือเรียกอีกอย่างว่า TCP Options สามารถเพิ่ม TCP Header แต่ไม่เกิน 4 byte เพื่อบอกว่าขนาดของ TCP Header มีส่วนที่เป็น Data Offset

2.7.9.12 TCP Ports

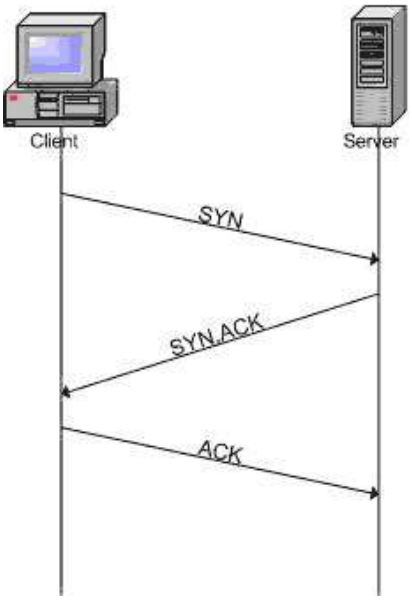
TCP port เป็นพอร์ตที่ให้บริการเชื่อมต่อในการส่งข้อมูล รวมถึงแต่ละพอร์ตของ TCP Segment คือพอร์ตที่ Application Layer ใช้ในการทำกระบวนการต่างจาก segment ที่ทำการส่ง และ พอร์ตเป้าหมายที่ Application Layerใช้ในกระบวนการส่งข้อมูล มีตัวเลขพอร์ตที่

ถูกกำหนดโดย Internet Assigned Numbers Authority (IANA) ถึงโพรโตคอลในชั้น Application Layer

2.7.10 3-Way hand shake

ก่อนที่จะเริ่มต้นการสื่อสาร จะต้องมีการส่งสัญญาณเพื่อบอกโฮสต์อีกฝั่งหนึ่งให้เตรียมตัวติดต่อ ซึ่งกระบวนการที่ใช้มีชื่อเรียกว่า 3-Way Hand Shake ดังแสดงในภาพที่ 2.14 มีขั้นตอนคือ

1) เครื่องไคลเอนต์จะทำการส่งเซกเมนต์ โดยเปิด SYN Flag ระบุหมายเลขพอร์ตที่ต้องการติดต่อบนเซิร์ฟเวอร์และระบุหมายเลข ลำดับของข้อมูล (ISN - Initial Sequence Number)
2) เครื่องเซิร์ฟเวอร์เมื่อได้รับข้อมูลเซกเมนต์จากข้อ 1 ก็จะตอบกลับด้วยการเพิ่มค่า ISN ที่ได้รับขึ้นอีก 1 พร้อมทั้งระบุหมายเลขลำดับ (ISN) ของตนเอง และเปิด SYN กับ ACK Flag
3) ไคล์เอนต์เมื่อได้รับการตอบกลับจากเซิร์ฟเวอร์ตามข้อ 2 ก็จะทำการตอบรับกลับไป โดยการเพิ่มค่า ISN ของเซิร์ฟเวอร์ขึ้นอีก 1 และเปิด ACK Flag เมื่อผ่านการสร้าง connection ทั้ง 3 ขั้นตอนแล้ว ตอนนี้ทั้งไคล์เอนต์ และเซิร์ฟเวอร์เปรียบเสมือนมีการเชื่อมต่อถึงกันแล้ว สถานะของการเชื่อมต่อในขณะนี้เรียกว่า Established



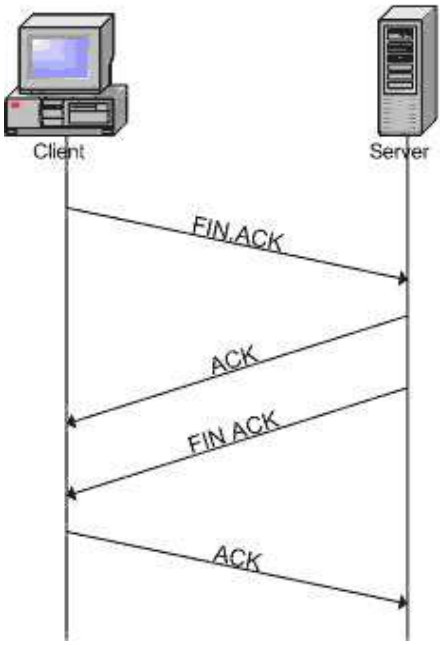
ภาพที่ 2.14 แบบจำลอง 3-Way hand shake

2.7.11 Connection Termination

เมื่อการสื่อสารของทั้งสองฝั่งจบลง และไม่ต้องการรับส่งข้อมูลอีกต่อไป จะต้องทำตามขั้นตอนการยุติการสื่อสารเพื่อให้การสื่อสารจบลงอย่างสมบูรณ์ ซึ่งมีอยู่ 4 ขั้นตอนคือ

1) ไคลเอนต์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังเซิร์ฟเวอร์
2) เซิร์ฟเวอร์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag
3) เซิร์ฟเวอร์ทำการส่ง ISN พร้อมกับ FIN ACK Flag ไปยังไคลเอนต์
4) ไคลเอนต์ทำการตอบรับ ISN และบวกค่า ISN อีก 1 พร้อม ACK Flag

การยุติการเชื่อมต่อ โดยส่ง FIN ACK ออกไปมีความหมายคือ โฮสต์ที่ส่งไม่มีข้อมูลจะส่งไปอีก มิใช่ต้องการปิดการสื่อสารทั้งหมดในทันที ดังนั้นจึงต้องทำทั้งสองทาง การสื่อสารจึงจะยุติลงอย่างสมบูรณ์ ในการใช้งานจริง อาจมีการยุติการสื่อสารเพียงด้านเดียว คือหยุดส่งข้อมูล แต่ยังคงเปิดพอร์ตไว้รอรับข้อมูลจากอีกด้านหนึ่ง ทั้งนี้ขึ้นอยู่กับลักษณะการใช้งาน การปิดพอร์ตสื่อสารเพียงด้านเดียวเช่นนี้ เรียกว่า Half-Close ภาพที่ 2.15 แสดงแบบจำลองการปิดการสื่อสาร



ภาพที่ 2.15 แบบจำลองการปิดการสื่อสาร

2.7.12 Listening

ในการเขียนโปรแกรมเพื่อติดต่อแบบ Socket Programming แบบ TCP นั้น ในส่วนของ TCP Server จะต้องเปิด Port รอรับการเชื่อมต่อเพื่อสร้าง Connection กับ Client โดยการ

เปิด Port เพื่อรอรับการเชื่อมต่อนี้เรียกว่า Listening Port ซึ่ง Listening Port นี้จะต้องเป็นหมายเลข Port ที่ไม่เปลี่ยนแปลงและใช้เพื่อรอรับการเชื่อมต่อจาก Client เท่านั้น

2.7.13 Accept Socket

เมื่อมี Client ติดต่อเข้ามายัง Listening Port เพื่อขอสร้างการเชื่อมต่อใน TCP แล้ว ทาง Server จะทำการ Accept Socket โดยการสร้าง Socket ใหม่เพื่อมารองรับ Connection ที่จะสร้างขึ้นเพื่อเชื่อมต่อกับ Client โดย Accept Socket ที่สร้างขึ้นจะใช้หมายเลข Port ที่ Server generate ขึ้นมาให้

2.8 Microcontroller

Microcontroller คือ อุปกรณ์ควบคุมขนาดเล็ก ซึ่งบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ โดยในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู, หน่วยความจำและพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน

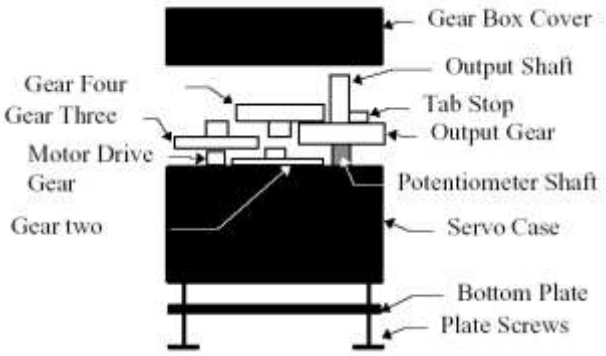
โครงสร้างโดยทั่วไป ของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ๆ ดังต่อไปนี้

1) หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)
2) หน่วยความจำ (Memory) สามารถแบ่งออกเป็น 2 ส่วน คือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะ คือข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกกระดาษทดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยง ข้อมูลก็จะหายไปคล้ายกับหน่วยความแรม (RAM) ในเครื่องคอมพิวเตอร์ทั่วๆ ไป แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรม ซึ่งข้อมูลจะหายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง
3) ส่วนติดต่อกับอุปกรณ์ภายนอก หรือพอร์ต (Port) มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุตเพื่อรับสัญญาณ อาจจะด้วยการกดสวิตช์ เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุตเพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

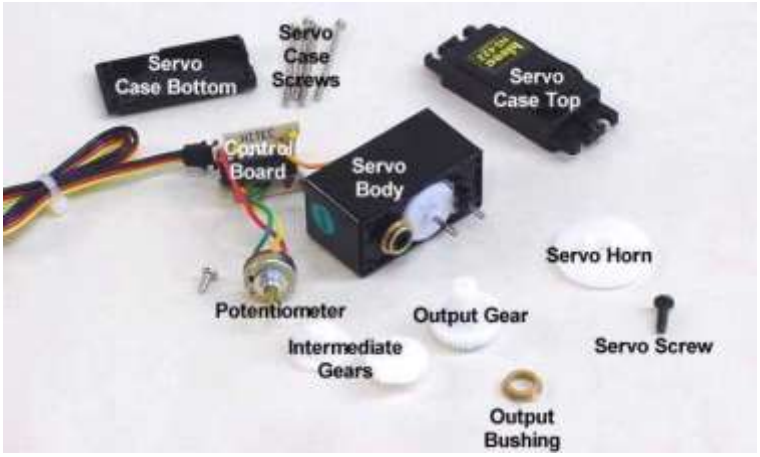
4) ช่องทางเดินของสัญญาณ หรือบัส (BUS) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำและพอร์ต เป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus), บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)
5) วงจรกำเนิดสัญญาณนาฬิกา นับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับการกำหนดจังหวะ หากสัญญาณนาฬิกามีความถี่สูง จังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้นส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้น มีความเร็วในการประมวลผลสูงตามไปด้วย

2.9 Servo Motor

Servo motor คือ มอเตอร์ไฟฟ้ากระแสตรง (DC motor) ที่ถูกประกอบรวมกับชุดเกียร์และส่วนควบคุมต่างๆไว้ในโมดูลเดียวกัน หรือภายในกล่องพลาสติกเดียวกัน โดยมอเตอร์ชนิดนี้จะมีสายต่อใช้งานงานเพียง 3 เส้นเท่านั้น คือ VCC, GND และ สายสัญญาณควบคุม(Control Line) ซึ่งสามารถควบคุมให้มอเตอร์หมุนซ้าย หรือ ขวาได้จากสายสัญญาณเพียงเส้นเดียว โดยสัญญาณที่ใช้ควบคุมนี้จะเป็นสัญญาณ พัลส์วิดมอด (PWM) แบบ TTL Level ระดับแรงดันที่จ่ายให้มอเตอร์นี้จะอยู่ในช่วงประมาณ 4 ถึง 6 โวลต์ ขึ้นอยู่กับคุณสมบัติของมอเตอร์แต่ละตัว ข้อดีของมอเตอร์ชนิดนี้ก็คือ จะมีขนาดเล็กน้ำหนักเบา, ให้แรงบิดสูง, กินพลังงานน้อยและสามารถควบคุม ด้วยแรงดันลอจิกที่เป็น TTL ได้โดยตรงไม่จำเป็นต้องต่อวงจรขับ(Driver) อื่นๆ เพราะ มอเตอร์ชนิดนี้จะมีวงจรควบคุมบรรจุไว้ภายในอยู่แล้ว ซึ่งมอเตอร์ชนิดนี้สามารถควบคุมให้หมุนไปในตำแหน่ง หรือ ทิศทางองศาที่ต้องการได้ โดยอาศัยสัญญาณความกว้างพัลส์ ที่ป้อนให้มอเตอร์ แต่เซอร์โวมอเตอร์นี้จะหมุนได้แค่เพียงในช่วงประมาณ 180º หรือ ครึ่งรอบเท่านั้น หรือ บางรุ่นอาจหมุนได้ถึง 210˚ แต่จะไม่สามารถหมุนเป็นวงรอบได้เนื่องจากโครงสร้างภายในจะประกอบด้วย ตัวต้านทานชนิดปรับค่าได้ (VR) ที่ทำหน้าที่ตรวจสอบตำแหน่งการหมุนของมอเตอร์ และ ตัวต้านทานนี้จะถูกยึดติดกับแกนหมุนของมอเตอร์ ซึ่งจากการที่ตัวต้านทานปรับค่านี้ไม่สามารถหมุนเป็นวงรอบได้ ดังนั้น เซอร์โวมอเตอร์จึงถูกออกแบบให้หมุนได้เพียงแค่ประมาณ 180 องศา หรือ ครึ่งรอบเท่านั้น เพื่อป้องกันความเสียหายที่จะเกิดกับตัวต้านทานปรับค่าได้ แต่ถ้าหากเราต้องการให้มอเตอร์หมุนเป็นวงรอบ (360º) นั้นก็สามารถทำได้ โดยจะต้องทำการปรับแต่ง (Modify) ดัดแปลงชิ้นส่วนบางอย่างของมอเตอร์ ภาพที่ 2.16 แสดงองค์ประกอบของ Servo Motor และภาพที่ 2.17 ส่วนประกอบของ Servo Motor



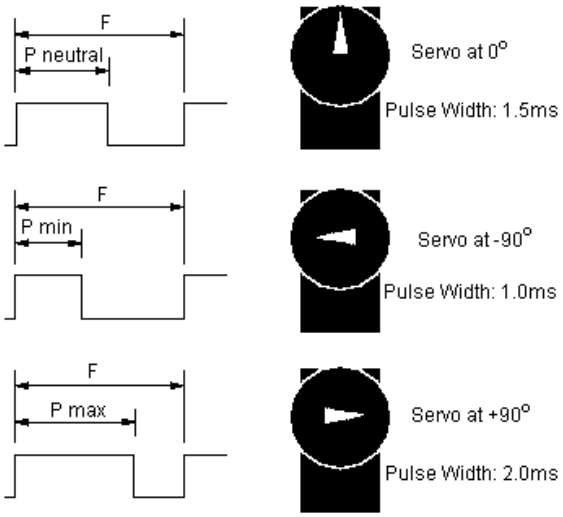
ภาพที่ 2.16 องค์ประกอบของ Servo Motor



ภาพที่ 2.17 ส่วนประกอบของ Servo Motor

การควบคุมการทำงานของ เซอร์โวมอเตอร์ ทำได้โดย การป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ซึ่งตำแหน่งและทิศทางการหมุนของมอเตอร์นี้จะขึ้นอยู่กับขนาดของความกว้างของพัลส์นั้นๆ โดยทั่วไปแล้วความกว้างของสัญญาณพัลส์จะมีจุดให้อ้างอิง 3 จุด ดังภาพที่ 2.18 คือ

- สัญญาณความกว้างพัลส์ขนาด 1.5 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม 0 องศา หรือ จุดกึ่งกลางของมอเตอร์
- สัญญาณความกว้างพัลส์ขนาด 1 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม - 90 องศา หรือในทิศทางทวนเข็มนาฬิกา
- สัญญาณความกว้างพัลส์ขนาด 2 ms จะควบคุมให้เซอร์โวมอเตอร์หมุนไปอยู่ที่ตำแหน่งมุม + 90 องศา หรือในทิศทางตามเข็มนาฬิกา



ภาพที่ 2.18 สัญญาณพัลส์กับตำแหน่งการหมุน

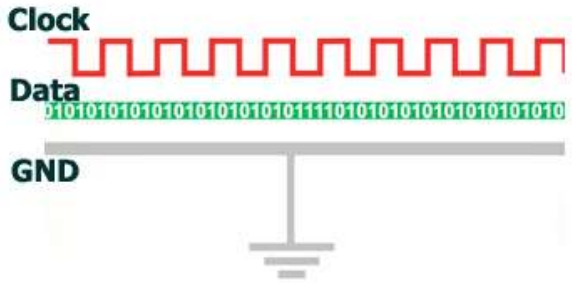
ส่วนการที่จะควบคุมให้มอเตอร์หมุนเป็นมุมอื่นๆ นั้นก็สามารถทำได้โดยการป้อนสัญญาณพัลส์เป็นระดับความกว้างต่างๆ โดยอ้างอิงจากจุด ทั้ง 3 จุดที่กล่าวมานี้ ตัวอย่างเช่น ถ้าต้องการให้มอเตอร์หมุนไปที่มุม - 45 องศา เราก็จะต้องป้อนสัญญาณพัลส์ที่มีความกว้าง 1.25 ms เป็นต้น และสัญญาณพัลส์นี้จะต้องจ่ายให้มอเตอร์ทุกๆ 20 ms (Period) เพื่อรักษาสภาพตำแหน่งของมอเตอร์ไว้ โดยหลักการก็คือ จะอาศัยการเปรียบเทียบช่วงเวลาของความกว้างพัลส์ที่จ่ายให้กับมอเตอร์ทางขาสัญญาณควบคุมกับค่าเวลาของวงจร RC ภายในบอร์ดควบคุมในตัวของมอเตอร์ ซึ่งค่าเวลาของวงจร RC นี้จะมีการเปลี่ยนแปลงตามการหมุนของมอเตอร์ เนื่องจากตัวต้านทานปรับค่าจะถูกยึดติดอยู่กับแกนหมุนของมอเตอร์ ซึ่งการหมุนของมอเตอร์จะทำให้ค่าความต้านทานของตัวต้านทานปรับค่า (VR) เปลี่ยนแปลงไป เป็นผลทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงตามไปด้วย โดยในขณะที่เราป้อนสัญญาณความกว้างพัลส์ให้กับมอเตอร์ทางขาสัญญาณควบคุม สัญญาณนี้จะถูกนำไปเปรียบเทียบกับค่าเวลาของวงจร RC หากค่าทั้ง 2 ไม่เท่ากันมอเตอร์ก็จะหมุนทำให้ค่าเวลาของวงจร RC เปลี่ยนแปลงจนกระทั้งค่าเวลาความกว้างพัลส์ของ วงจร RC เปลี่ยนแปลงจนเท่ากับสัญญาณพัลส์ทางขาควบคุม (Control line) มอเตอร์จึงจะหยุดหมุน

2.10 UART Serial Communication

การสื่อสารแบบอนุกรมจะแบ่งเป็น 2 แบบ คือ

1) การสื่อสารอนุกรมแบบ Synchronous เป็นรูปแบบที่ใช้วิธีส่งข้อมูล โดยใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะ การรับส่งข้อมูล การส่งข้อมูลแบบนี้เป็นการรับส่งที่ค่อนค่างมีคุณภาพ และส่งได้ที่ความเร็วสูง มีโอกาสที่ข้อมูลจะสูญหายระหว่างการส่งน้อย ตัวอย่างการส่งข้อมูลลักษณะนี้เช่น I2C, I2S, SPI

ข้อเสียของการส่งข้อมูลแบบนี้คือ ต้องใช้สายสัญญาณมาก เพราะว่าต้องส่ง Clock ไปด้วย โดยภาพที่ 2.19 การสื่อสารอนุกรมแบบ Synchronous



ภาพที่ 2.19 การสื่อสารอนุกรมแบบ Synchronous

2) การสื่อสารอนุกรมแบบ Asynchronous เป็นการส่งข้อมูลที่ไม่ต้องใช้สัญญาณ Clock มาเป็นตัวกำหนดจังหวะการรับส่งข้อมูลแต่ ใช้วิธีกำหนด รูปแบบ Format การรับส่งข้อมูลขึ้นมาแทน และ อาศัยการกำหนด ความเร็วของการรับ และ ส่ง ที่เท่ากันทั้งฝั่งรับและฝั่งส่ง ข้อดีของการใช้ Asynchronous คือสามารถสื่อสารแบบ Full Duplex รับ และ ส่งได้ในเวลาเดียวกัน แต่ Asynchronous มีโอกาสที่ข้อมูลจะสูญหายขณะรับส่งข้อมูล หรือรับส่งข้อมูลผิดพลาดได้มากกว่าแบบ Synchronous

UART ย่อมาจากคำว่า Universal Asynchronous Receiver Transmitter หมายถึงอุปกรณ์ที่ทำหน้าที่รับและส่งข้อมูลแบบอะซิงโครนัส (Asynchronous) ซึ่งเป็นส่วนหนึ่งในการสื่อสารอนุกรมแบบ Asynchronous

UART (Universal Asynchronous Receiver Transmitter) จะมีรูปแบบการส่งข้อมูล ที่ถูกกำหนดขึ้นมาเพื่อใช้รับส่งข้อมูลแบบ Asynchronous โดยมีรูปแบบดังภาพที่ 2.20

ภาพที่ 2.20 รูปแบบการส่งข้อมูลของ UART

เริ่มต้นจาก Start Bit เป็น Logic 0 จากนั้นจะตามด้วย Data ที่เราส่ง แล้วจะถูกปิดด้วย STOP Bit เป็น Logic 1