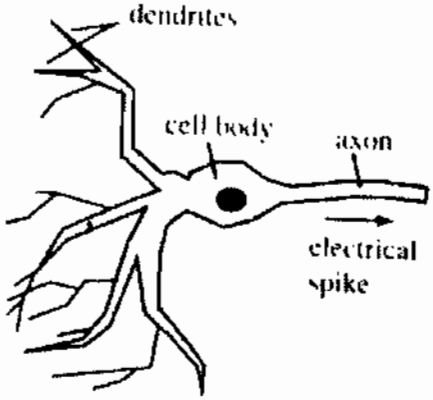
# บทที่ 2
# ทฤษฎีและเครื่องมือที่ใช้

ในบทนี้กล่าวถึงการศึกษาทฤษฎีเกี่ยวกับข่ายงานระบบประสาทและเครื่องมือและอุปกรณ์ที่ใช้ในงานวิจัยโดยแบ่งเนื้อหาออกเป็น 4 ส่วนคือ ส่วนแรกกล่าวถึงข่ายงานระบบประสาทซึ่งอธิบายขั้นตอนการทำงานของข่ายงานระบบประสาทที่สามารถทำให้เกิดการเรียนรู้ในลักษณะของการจำแนกได้ รวมทั้งอธิบายถึงขั้นตอนวิธีการฝึกสอนแบบป้อนกลับที่ใช้ในการฝึกสอนข่ายงานระบบประสาทที่ใช้ในงานวิจัยนี้ ส่วนที่สองกล่าวถึงหลักการเกี่ยวกับการทรงตัวและการเดินของหุ่นยนต์สี่ขา ส่วนที่สามกล่าวถึง Open Dynamics Engine ซึ่งเป็นเครื่องมือช่วยสำหรับการสร้างแบบจำลองทางคอมพิวเตอร์กราฟฟิกส์ที่สามารถจำลองสภาพทางฟิสิกส์ของวัตถุแข็งได้ ในส่วนสุดท้ายกล่าวถึงอุปกรณ์ทางฮาร์ดแวร์ที่ใช้สื่อสารและควบคุมหุ่นยนต์

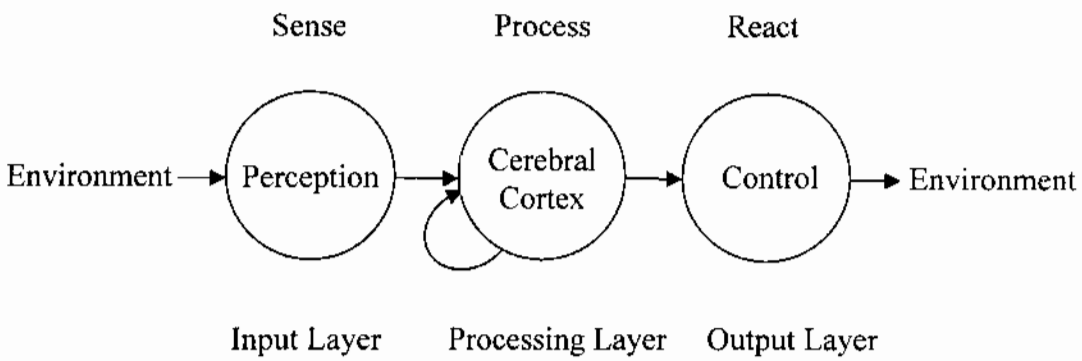
## 2.1 ข่ายงานระบบประสาท (Neural Network)




**ภาพที่ 2-1** เซลล์ประสาท [5]

ข่ายงานระบบประสาท (Neural Network) เป็นการจำลองการทำงานพื้นฐานบางส่วนของสมองมนุษย์ ในสมองประกอบด้วยตัวเซลล์ประสาท (Cell Body : Neuron ) ซึ่งทำหน้าที่เป็นหน่วยประมวลผล (Processing Element) ตัวเซลล์ประสาทเชื่อมต่อกันด้วยแกนประสาทนำออก (Axon) โดยมีใยประสาทนำเข้า (Dendrite) ทำหน้าที่รับข้อมูลเข้าสู่เซลล์ประสาท ในสมองมนุษย์มีเซลล์

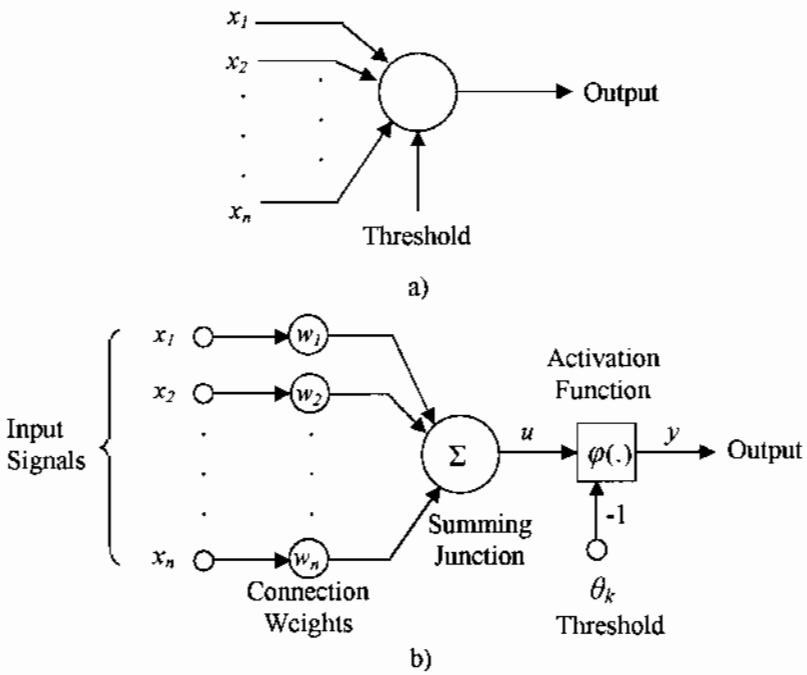
ประสาทประมาณ $10^{11}$ ตัว [5] แต่ละตัวเชื่อมต่อกับเซลล์ประสาทอื่นๆ ประมาณ 1,000 ตัว ยกเว้นในส่วนของ Cerebral Cortex ที่มีจำนวนเซลล์ประสาทมากกว่าบริเวณอื่น [6] โดยใยประสาทนำเข้าทำหน้าที่รับสัญญาณไฟฟ้าเคมีซึ่งส่งมาจากเซลล์ประสาทตัวอื่นที่เชื่อมต่อกัน เมื่อสัญญาณไฟฟ้าเคมีที่รับเข้ามาเกินค่าค่าหนึ่ง เซลล์จะถูกกระตุ้นและส่งสัญญาณไปทางแกนประสาทนำออกไปยังเซลล์อื่น ๆ ต่อไป โครงสร้างของสมองมนุษย์นั้นซับซ้อนมาก แต่สามารถพิจารณาให้ง่ายขึ้นได้โดยมองการทำงานของสมองให้เป็นแบบลำดับชั้น (Layer) โดยชั้นข้อมูลเข้าใช้แทนการรับรู้ของมนุษย์ เช่น ระบบสัมผัสต่าง ๆ ชั้นของการประมวลผลคือการประมวลผลข้อมูลเข้า ส่วนชั้นข้อมูลออกคือพฤติกรรมที่แสดงออกเพื่อตอบสนองกับสิ่งแวดล้อมที่มากระทบ




**ภาพที่ 2-2** รูปแบบการทำงานของสมองอย่างง่าย [6]

2.1.1 เพอร์เซ็บตรอน (Perceptron)

เพอร์เซ็บตรอน คือข่ายงานระบบประสาทรูปแบบหนึ่ง ที่ประกอบด้วยหน่วยประมวลผลเพียงหน่วยเดียวเท่านั้น [5, 6, 7] ข้อมูลนำเข้าของเพอร์เซ็บตรอนจะถูกคูณด้วยค่าน้ำหนักการเชื่อมโยง (Connection Weights) จากนั้นนำมาคำนวณผลรวมที่ได้และหักออกด้วยค่าขีดเริ่มเปลี่ยน (Threshold) แล้วนำผลลัพธ์เข้าสู่ฟังก์ชันกระตุ้น (Activation Function) ซึ่งค่าที่ได้หลังจากผ่านฟังก์ชันกระตุ้นแล้วคือผลลัพธ์ที่ได้จากเพอร์เซ็บตรอน โดยทั่วไปฟังก์ชันกระตุ้นที่นิยมใช้กับเพอร์เซ็บตรอนมักเป็นฟังก์ชันแบบสองขั้ว (Bipolar Function) ที่ให้คำตอบเป็น 2 ค่าคือบวกและลบ (+, -) หรือฟังก์ชันไบนารี (Binary Function) ที่ให้คำตอบเป็น 2 ค่า คือศูนย์และหนึ่ง (0, 1) ในช่วงแรก ๆ ของการวิจัยและพัฒนาทางด้านข่ายงานระบบประสาทนั้น เพอร์เซ็บตรอนเป็นรูปแบบของข่ายงานระบบประสาทที่ง่ายที่สุด ที่สามารถใช้จำแนกชุดข้อมูลที่แยกกันได้เชิงเส้น (Linearly Separable Patterns) อย่างเช่นฟังก์ชันตรรกะพื้นฐาน NOT Gate, AND Gate และ OR Gate สำหรับสัญลักษณ์และรูปแบบการเชื่อมโยงของการทำงานภายในเพอร์เซ็บตรอนแสดงในภาพที่ 2-3




**ภาพที่ 2-3** เพอร์เซ็ปตรอน a) สัญลักษณ์ b) รูปแบบการจัดวาง [5]

จากภาพของเพอร์เซ็ปตรอนมีส่วนประกอบพื้นฐานที่สำคัญ 3 ส่วนคือ

1) ค่าน้ำหนักการเชื่อมโยง (Connection Weights หรือ Synapstic Weights)

2) การรวมค่าน้ำหนักการเชื่อมโยงทั้งหมด (Summing Junction)

3) ฟังก์ชันกระตุ้น (Activation Function)

และสามารถอธิบายการทำงานของเพอร์เซ็ปตรอนได้ดังสมการต่อไปนี้

$$u = \sum_{i-1}^{n} w_i x_i \tag{2-1}$$

$$y = \varphi(u - \theta) \tag{2-2}$$

โดยที่ $x_i$ คือข้อมูลเข้าที่ $i$ ของเพอร์เซ็ปตรอน

$w_i$ คือค่าน้ำหนักการเชื่อมโยงของข้อมูลเข้า $x_i$

$\theta$ คือค่าขีดเริ่มเปลี่ยน

$n$ คือมิติของเวกเตอร์ข้อมูลเข้า

$u$ คือระดับกระตุ้นภายในสุทธิ

$\varphi$ คือฟังก์ชันกระตุ้น

$y$ คือข้อมูลออกของเพอร์เซ็ปตรอน

สมการที่ 2-1 คือการคำนวณผลรวมของข้อมูลเข้าแต่ละตัวของเพอร์เซ็บตรอนที่ถูกคูณด้วยค่าน้ำหนักการเชื่อมโยง ซึ่งได้ผลลัพธ์คือ $u$ ส่วนสมการที่ 2-2 คือการคำนวณผลลัพธ์ที่เป็นคำตอบของเพอร์เซ็บตรอน โดยการนำผลลัพธ์ที่ได้จากสมการที่ 2-1 ลบด้วยค่าขีดเริ่มเปลี่ยน $\theta$ และนำเข้าสู่ฟังก์ชันกระตุ้น $\varphi$

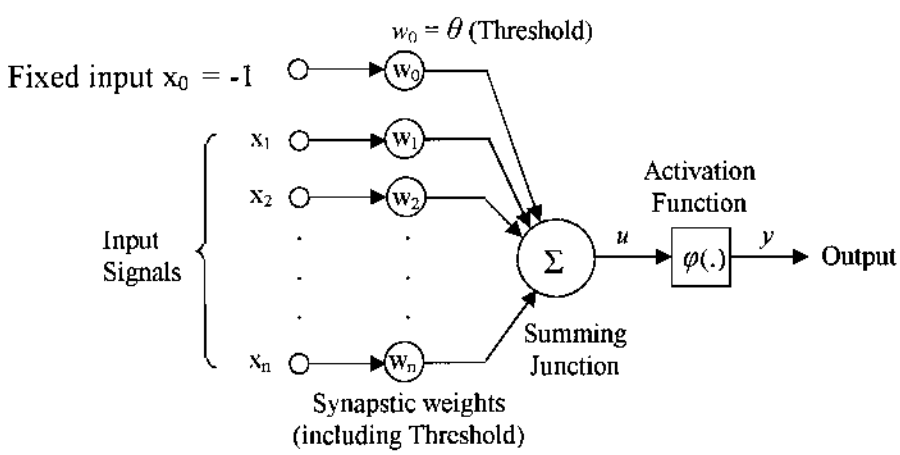
ในกรณีที่ใช้ฟังก์ชันกระตุ้นของเพอร์เซ็บตรอนเป็นฟังก์ชันไบนารี ข้อมูลออกหรือคำตอบจะมีค่าที่เป็นไปได้ 2 ค่า คือ 0 หรือ 1 ตามเงื่อนไขที่กำหนด ดังตัวอย่างในสมการที่ 2-3

$$y = \begin{cases} 0 & \text{if } u >= 0 \\ 1 & \text{if } u < 0 \end{cases} \qquad (2\text{-}3)$$

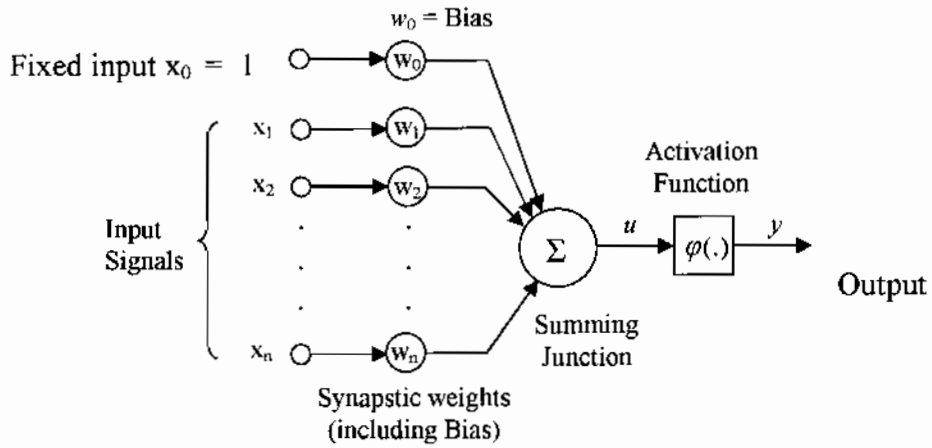
ในกรณีที่ใช้ฟังก์ชันกระตุ้นของเพอร์เซ็บตรอนเป็นฟังก์ชันสองขั้ว ข้อมูลออกหรือคำตอบจะมีค่าที่เป็นไปได้ 2 ค่า คือ 1 หรือ -1 ตามเงื่อนไขที่กำหนด ดังตัวอย่างในสมการที่ 2-4

$$y = \begin{cases} 1 & \text{if } u >= 0 \\ -1 & \text{if } u < 0 \end{cases} \qquad (2\text{-}4)$$

นอกจากรูปแบบของเพอร์เซ็บตรอนจะแสดงได้ดังภาพที่ 2-3 แล้วยังสามารถจัดรูปแบบของเพอร์เซ็บตรอนเป็นแบบอื่นได้อีก เช่นในภาพที่ 2-4 และภาพที่ 2-5 เป็นตัวอย่างอีก 2 แบบของเพอร์เซ็บตรอน โดยต่างกันที่ข้อมูลเข้าที่เพิ่มเข้ามา (Fixed Input $x_0$) ซึ่งเป็นค่าคงที่ และไม่คำนวณโดยทำการลบออกด้วยค่าขีดเริ่มเปลี่ยนหลังจากคำนวณค่าผลลัพธ์แล้ว




**ภาพที่ 2-4** เพอร์เซ็บตรอนแบบลบออกด้วยค่าขีดเริ่มเปลี่ยน




**ภาพที่ 2-5** เพอร์เซ็บตรอนแบบเพิ่มด้วยค่า Bias

รูปแบบของเพอร์เซ็บตรอนในภาพที่ 2-4 ทำให้สมการของเพอร์เซ็บตรอนในส่วนของค่าที่ได้ก่อนเข้าสู่ฟังก์ชันกระตุ้นต่างออกไป ดังแสดงด้านล่างนี้ สมการที่ 2-5 ค่าที่คำนวณได้จะถูกลบออกด้วยค่าขีดเริ่มเปลี่ยน ($\theta$) ส่วนสมการที่ 2-6 ค่าที่คำนวณได้จะถูกเพิ่มเข้าด้วยค่า Bias ($w_0$)

$$u = \sum_{i=1}^{n} w_i x_i \; - \; \theta \tag{2-5}$$

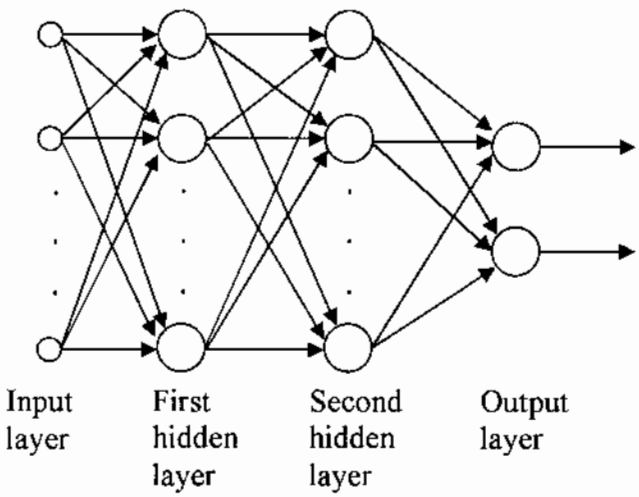
$$u = w_0 + \sum_{i=1}^{n} w_i x_i \tag{2-6}$$

แม้ว่าเพอร์เซ็บตรอนจะมีประสิทธิภาพและสามารถนำมาใช้เรียนรู้ฟังก์ชันทางตรรกะพื้นฐาน เช่น NOT Gate, OR Gate หรือ AND Gate ได้แต่ไม่สามารถเรียนรู้ฟังก์ชันง่าย ๆ อย่าง Exclusive OR (XOR) ได้เนื่องจากคำตอบที่ได้จากฟังก์ชัน XOR กับข้อมูลเข้ามีความสัมพันธ์กันแบบไม่เชิงเส้น

2.1.2 มัลติเลเยอร์เพอร์เซ็บตรอน (Multi-Layer Perceptron)

มัลติเลเยอร์เพอร์เซ็บตรอนเกิดจากการนำเพอร์เซ็บตรอนหลาย ๆ หน่วย มาจัดวางให้อยู่ในรูปของข่ายงานแบบลำดับชั้นดังแสดงในภาพที่ 2-6 ถูกสร้างขึ้นเพื่อแก้ปัญหาการเรียนรู้ฟังก์ชันที่มีความสัมพันธ์แบบไม่เชิงเส้น เช่น ปัญหาของฟังก์ชัน XOR โดยเพอร์เซ็บตรอนในชั้นซ่อนที่หนึ่ง (First Hidden Layer) รับข้อมูลเข้าจากชั้นข้อมูลเข้า (Input Layer) ข้อมูลออกของเพอร์เซ็บตรอนในชั้นซ่อนที่หนึ่ง จะเป็นข้อมูลเข้าสำหรับเพอร์เซ็บตรอนในชั้นซ่อนที่สอง (Second Hidden Layer) ในทำนองเดียวกันข้อมูลออกของเพอร์เซ็บตรอนแต่ละหน่วยในชั้นซ่อนที่สองจะเป็นข้อมูลเข้า

ให้กับเพอร์เซ็บตรอนในชั้นข้อมูลออก (Output Layer) ซึ่งปกติแต่ละเพอร์เซ็บตรอนในแต่ละชั้นจะเชื่อมโยงไปยังทุกเพอร์เซ็บตรอนของชั้นถัดไป (Fully Connected) และไม่มีการเชื่อมต่อข้ามชั้นหรือเชื่อมต่อย้อนกลับ




**ภาพที่ 2-6** แผนภาพของมัลติเลเยอร์เพอร์เซ็บตรอนแบบมี 2 ชั้นซ่อน

โดยปกติมัลติเลเยอร์เพอร์เซ็บตรอนจะไม่ใช้ฟังก์ชันกระตุ้นเป็นฟังก์ชันไบนารีหรือฟังก์ชันสองขั้ว แต่จะใช้ฟังก์ชันเชิงตรรกะ (Logistic Function) หรือฟังก์ชันไฮเพอร์โบลิกแทนเจนต์ (Hyperbolic Tangent Function) โดยสำหรับกรณีที่ฟังก์ชันกระตุ้นเป็นฟังก์ชันเชิงตรรกะข้อมูลออกของเพอร์เซ็บตรอนจะมีค่าดังสมการที่ 2-7

$$y = \varphi(u) = \frac{1}{1+\exp(-u)} \tag{2-7}$$

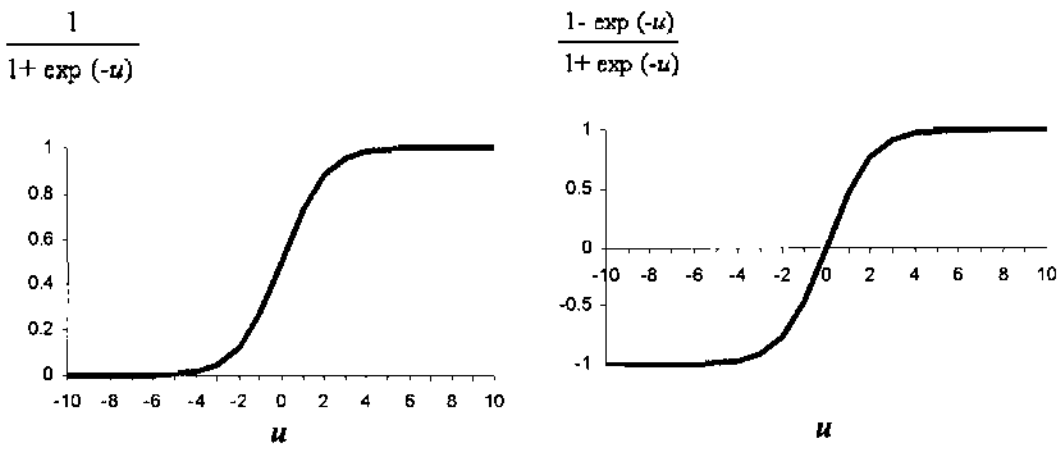
โดยที่ $u$ คือระดับกระตุ้นภายในสุทธิ

$y$ คือข้อมูลออกของเพอร์เซ็บตรอน

ส่วนกรณีที่ฟังก์ชันกระตุ้นเป็นฟังก์ชันไฮเพอร์โบลิกแทนเจนต์ ข้อมูลออกของเพอร์เซ็บตรอนจะมีค่าดังสมการที่ 2-8

$$y = \varphi(u) = \frac{1-\exp(-u)}{1+\exp(-u)} \tag{2-8}$$

จากการใช้ฟังก์ชันเชิงตรรกะเป็นฟังก์ชันกระตุ้น ข้อมูลออกของเพอร์เซ็บตรอนจะมีพิสัยอยู่ในช่วง $0 < y < 1$ ส่วนการใช้ฟังก์ชันไฮเพอร์โบลิกแทนเจนต์ ข้อมูลออกของเพอร์เซ็บตรอนจะมีพิสัยอยู่ในช่วง $-1 < y < 1$




**ภาพที่ 2-7** พิสัยข้อมูลของฟังก์ชันกระตุ้น a) พิสัยข้อมูลของฟังก์ชันเชิงตรรกะ b) พิสัยข้อมูลของฟังก์ชันไฮเพอร์โบลิกแทนเจนต์

เมื่อเปลี่ยนฟังก์ชันกระตุ้นในเพอร์เซ็บตรอนและนำมาประกอบกันเป็นข่ายงานของมัลติเลเยอร์เพอร์เซ็บตรอน ส่งผลให้ประสิทธิภาพในการจำแนกของข่ายงานระบบประสาทขยายขึ้น จากเดิมที่สามารถใช้เพอร์เซ็บตรอนในการจำแนกชุดข้อมูลที่แยกกันได้เชิงเส้นเท่านั้น โดยที่มัลติเลเยอร์เพอร์เซ็บตรอนสามารถสร้างระนาบเกินไม่เชิงเส้น (Non-linear Hyper-plane) เพื่อใช้ในการจำแนกชุดข้อมูลที่มีความสัมพันธ์แบบไม่เชิงเส้น ผลลัพธ์จากเพอร์เซ็บตรอนแต่ละหน่วยของมัลติเลเยอร์เพอร์เซ็บตรอนแต่ละชั้นคำนวณได้จากสมการที่ 2-9

$$y_i^h = \varphi ( w_{i,1}^h y_1^{h-1} + w_{i,2}^h y_2^{h-1} + w_{i,3}^h y_3^{h-1} + \ldots + w_{i,m}^h y_m^{h-1} + \theta )$$

$$y_i^h = \varphi \left( \sum_{j=1}^{m} w_{i,j}^h y_j^{h-1} + \theta \right) \qquad (2\text{-}9)$$

โดยที่ $y_i^h$ คือผลลัพธ์ของเพอร์เซ็บตรอนที่ $i$ ของชั้น $h$

$w_{i,j}^h$ คือค่าน้ำหนักการเชื่อมโยงระหว่างเพอร์เซ็บตรอนที่ $j$ ของชั้น $h$-1 กับเพอร์เซ็บตรอนที่ $i$ ของชั้น $h$

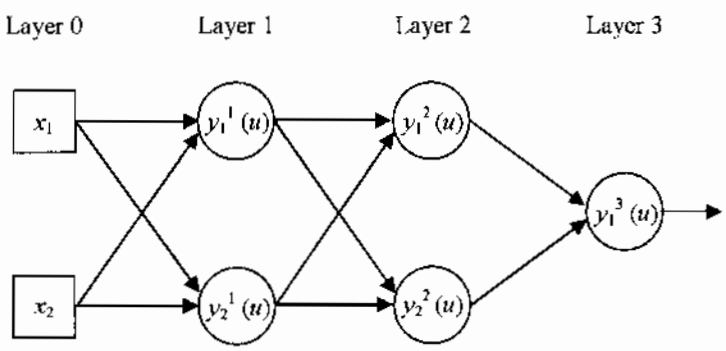
$\theta_i^h$ คือค่า Bias ของเพอร์เซ็บตรอนที่ $i$ ของชั้น $h$

$\varphi$ คือฟังก์ชันกระตุ้นของเพอร์เซ็บตรอน

$m$ คือจำนวนเพอร์เซ็บตรอนในชั้นซ่อนแต่ละชั้น

2.1.3 การฝึกสอนแบบป้อนกลับ (Back-Propagation Training)

การฝึกสอนแบบป้อนกลับเป็นวิธีการสำหรับฝึกสอนข่ายงานระบบประสาทแบบมัลติเลเยอร์เพอร์เซ็ปตรอนที่ได้รับความนิยมอย่างมาก โดยใช้การคำนวณค่าผิดพลาดหรือผลต่างที่เกิดขึ้นระหว่างค่าผลลัพธ์ที่คำนวณได้จริงในชั้นข้อมูลออก (Actual) กับค่าคำตอบที่ถูกต้อง (Desire) แล้วป้อนกลับเข้าไปในข่ายงานเพื่อปรับค่าน้ำหนักการเชื่อมโยง ในการฝึกสอนจำเป็นต้องใช้ชุดข้อมูลสำหรับฝึกสอน (Training data set) ซึ่งประกอบด้วยข้อมูลเข้า (Input Data) และค่าผลลัพธ์ที่ต้องการ (Desire output) การฝึกสอนจะเป็นกระบวนการทำซ้ำ โดยในแต่ละรอบของการทำงานค่าน้ำหนักการเชื่อมโยงระหว่างชั้นจะถูกปรับด้วยค่าที่คำนวณได้จากชุดข้อมูลเข้าชุดใหม่ และก่อนทำตามขั้นตอนวิธีของการฝึกสอนแบบกลับเพื่อทำการปรับค่าน้ำหนักการเชื่อมโยงนั้น จำเป็นต้องกำหนดค่าเริ่มต้นให้กับค่าน้ำหนักเชื่อมโยงทั้งหมดเสียก่อนโดยใช้วิธีการสุ่มค่า ปกติแล้วการสุ่มจะเลือกค่าจากช่วงข้อมูลน้อย ๆ เช่น ในช่วงค่าระหว่าง -0.05 ถึง 0.05 หรือช่วงค่าระหว่าง 0 ถึง 0.05 เป็นต้น หลังจากนั้นจึงเริ่มเข้าสู่กระบวนการฝึกสอน และเพื่ออธิบายการทำงานของขั้นตอนวิธีแพร่กระจายย้อนกลับ จะใช้การอ้างอิงกับข่ายงานระบบประสาทในภาพที่ 2-8 ซึ่งเป็นตัวอย่างข่ายงานระบบประสาทแบบ 3 ชั้น (three layers neural network) ที่มีข้อมูลเข้าจำนวน 2 หน่วยคือ $x_1$ ($y_1^0$) และ $x_2$ ($y_2^0$) ชั้นซ่อนชั้นที่หนึ่งมีหน่วยประมวลผล 2 หน่วยคือ $y_1^1$ และ $y_2^1$ ชั้นซ่อนชั้นที่สองมีหน่วยประมวลผล 2 หน่วยคือ $y_1^2$ และ $y_2^2$ ส่วนชั้นข้อมูลออกมีหน่วยประมวลผล 1 หน่วยคือ $y_1^3$




$x_1 = y_1^0$
$x_2 = y_2^0$

**ภาพที่ 2-8** ข่ายงานระบบประสาทแบบ 3 ชั้น ใช้อธิบายขั้นตอนวิธีการฝึกสอนแบบป้อนกลับ

ขั้นตอนวิธีของการฝึกสอนแบบป้อนกลับ (Back-Propagation Training Algorithm)

ให้ $X = [x_1, x_2, x_3, ..., x_n]$ คือชุดข้อมูลเข้า

$Z = [z_1, z_2, z_3, ..., z_n]$ คือคำตอบที่ถูกต้อง

1. ป้อนชุดข้อมูลสำหรับฝึกสอนเข้าสู่ข่ายงานทีละรายการ
2. คำนวณผลลัพธ์ของข่ายงานระบบประสาทแบบป้อนไปหน้าจากข้อมูลในข้อ 1.
3. คำนวณค่าผิดพลาดที่เกิดขึ้นของเพอร์เซ็บตรอนแต่ละหน่วย โดยเริ่มทำจากชั้นข้อมูลออก ย้อนกลับจนถึงชั้นซ่อนชั้นแรก ตามสมการที่ 2-10 และสมการที่ 2-11

สำหรับชั้นข้อมูลออก

$$\delta_i^n = (z_i - y_i^n) \qquad (2\text{-}11)$$

$n$ คือลำดับที่ของชั้นข้อมูลออก

$\delta_i^n$ คือค่าผิดพลาดของเพอร์เซ็บตรอนที่ $i$ ในชั้นข้อมูลออก $n$

$z_i$ คือข้อมูลที่ถูกต้องสำหรับเพอร์เซ็บตรอนที่ $i$ ของชั้นข้อมูลออก

สำหรับชั้นซ่อน

$$\delta_i^h = \sum_{i=1}^{m} w_{i,j}^h \ \delta_j^{h+1}$$

$w_{i,j}^h$ คือค่าน้ำหนักการเชื่อมโยงระหว่างเพอร์เซ็บตรอนที่ $j$ ของชั้น $h+1$ กับเพอร์เซ็บตรอนที่ $i$ ของชั้นซ่อน $h$

$h$ คือลำดับที่ของชั้นซ่อนจาก 0 ถึง $n$-1

$m$ คือจำนวนเพอร์เซ็บตรอนในชั้นซ่อนแต่ละชั้น

4. ปรับค่าน้ำหนักการเชื่อมโยง โดยใช้ค่าผิดพลาดที่คำนวณได้จากข้อ 3. โดยเริ่มทำการปรับจากชั้นข้อมูลเข้าก่อน ตามสมการต่อไปนี้

$$w'_{i,j}{}^h = w_{i,j}^h + \rho \, \delta_j^{h+1} \ dy_j^{h+1}(u)/du \ y_i^h \qquad (2\text{-}12)$$

$\rho$ คือค่าอัตราการเรียนรู้

$dy_i(u)/du$ คือค่าอนุพันธ์ของฟังก์ชันกระตุ้น

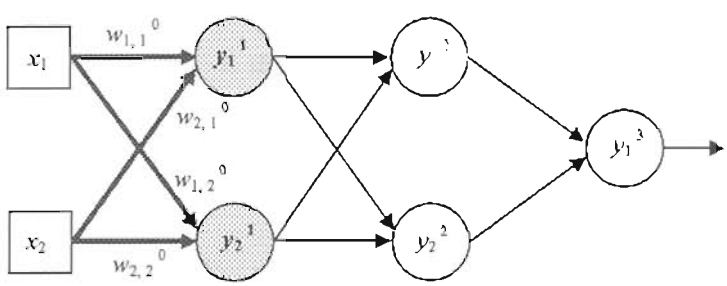
$u$ คือระดับกระตุ้นภายในสุทธิ

การปรับค่าน้ำหนักการเชื่อมโยงจะทำในทุก ๆ รอบที่มีการป้อนข้อมูลฝึกสอนแต่ละรายการเข้าไปในข่ายงาน ขั้นตอนการฝึกสอนนี้จะทำไปจนกว่า Error ของข่ายงานมีค่าต่ำกว่าค่าที่ยอมรับได้หรือจนกว่าจำนวนรอบของการฝึกครบตามจำนวนที่ตั้งไว้ สำหรับการหาค่าอัตราการเรียนรู้ที่เหมาะสมกับปัญหาต่าง ๆ นั้น เป็นสิ่งที่ทำได้ยาก เนื่องจากไม่มีกฎเกณฑ์ตายตัวหรือสูตรทางคณิตศาสตร์กำหนดไว้ ดังนั้นวิธีการทั่วไปในการหาค่าอัตราการเรียนรู้คือการทดลองปรับเปลี่ยนค่าไปเรื่อย ๆ จนกว่าจะได้ค่าที่เหมาะสมกับปัญหานั้นๆ ซึ่งมักจะเริ่มต้นการทดลองโดยใช้ค่าอัตราการเรียนรู้ค่าน้อย ๆ เช่น 0.05 หรือ 0.1 เป็นต้น โดยในส่วนของการปรับเปลี่ยนค่าของอัตราการเรียนรู้พิจารณาจากการลู่เข้าของค่าเฉลี่ยของค่าผิดพลาดยกกำลังสอง (Mean Square Error) โดยถ้าค่าเฉลี่ยของค่าผิดพลาดยกกำลังสองไม่ลู่เข้าสู่ค่าศูนย์ ก็จะทำการลดค่าอัตราการเรียนรู้ให้น้อยลง แต่ถ้าค่าค่าเฉลี่ยของค่าผิดพลาดยกกำลังสองลู่เข้าแต่ช้ามาก ก็จะใช้วิธีปรับค่าอัตราการเรียนรู้ให้มากขึ้น

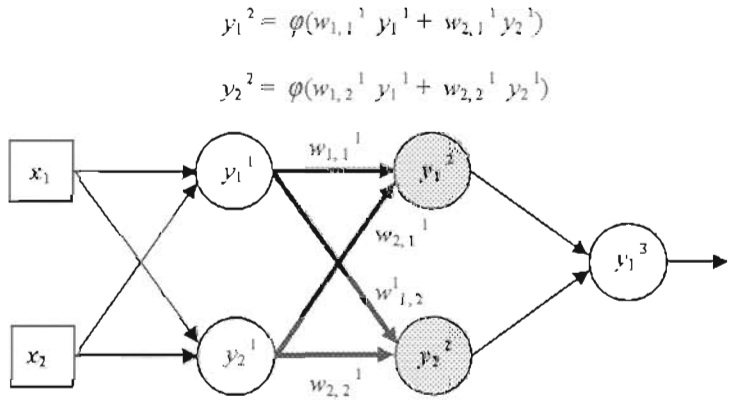
และเพื่อช่วยให้สามารถทำความเข้าใจการทำงานของขั้นตอนวิธีการฝึกสอนแบบป้อนกลับได้ง่ายขึ้น จึงอธิบายโดยใช้ภาพประกอบ โดยในส่วนของการคำนวณผลลัพธ์ของหน่วยประมวลผลแต่ละหน่วยด้วยวิธีการหาผลรวมของข้อมูลเข้าที่คูณด้วยค่าน้ำหนักการเชื่อมโยงแล้วนำเข้าสู่ฟังก์ชันกระตุ้น แสดงดังภาพที่ 2-9 ภาพที่ 2-10 และภาพที่ 2-11 ตามลำดับ

$$y_1^{\,1} = \varphi(w_{1,1}^{\,0} x_1 + w_{2,1}^{\,0} x_2)$$

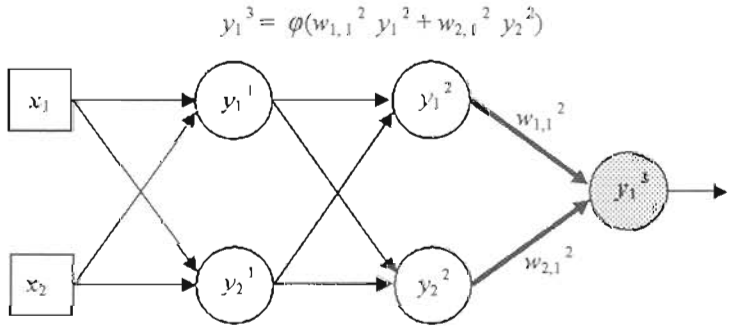
$$y_2^{\,1} = \varphi(w_{1,2}^{\,0} x_1 + w_{2,2}^{\,0} x_2)$$




ภาพที่ 2-9 การคำนวณแบบป้อนไปหน้าจากชั้นข้อมูลเข้าไปยังชั้นซ่อนชั้นที่ 1

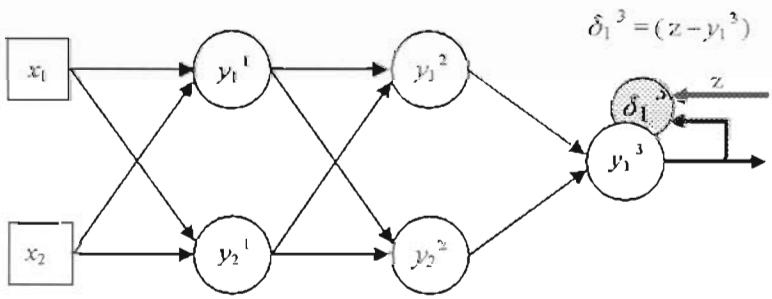



ภาพที่ 2-10 การคำนวณแบบป้อนไปหน้าจากชั้นซ่อนที่ 1 ไปยังชั้นซ่อนชั้นที่ 2




ภาพที่ 2-11 การคำนวณแบบป้อนไปหน้าจากชั้นซ่อนที่ 2 ไปยังชั้นข้อมูลออก

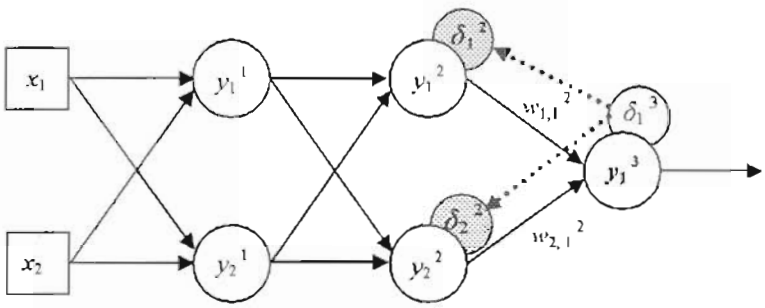
ในส่วนของการคำนวณค่าผิดพลาดสำหรับแต่ละหน่วยประมวลผลนั้น มีการประมวลผล 2 กรณีคือ กรณีของการคำนวณค่าผิดพลาดจากชั้นข้อมูลออก โดยคำนวณค่าผิดพลาดจากการนำค่าผลลัพธ์ที่ถูกต้อง ( $z_i$ ) ลบด้วยค่าผลลัพธ์ที่คำนวณได้ ($y_i$) ดังแสดงในภาพที่ 2-12 ส่วนอีกกรณีหนึ่งคือการคำนวณค่าผิดพลาดของหน่วยประมวลผลในชั้นซ่อน ซึ่งค่าผิดพลาดในกรณีนี้คำนวณได้จากการนำผลรวมของค่าผิดพลาดจากชั้นข้อมูลออกที่คูณด้วยค่าน้ำหนักการเชื่อมโยง ดังแสดงในภาพที่ 2-12 ภาพที่ 2-13 และภาพที่ 2-14 ตามลำดับ




ภาพที่ 2-12 การคำนวณค่าผิดพลาดในชั้นข้อมูลออก

$$\delta_1^{\,2} = w_{1,1}^{\,2}\,\delta_1^{\,3}$$

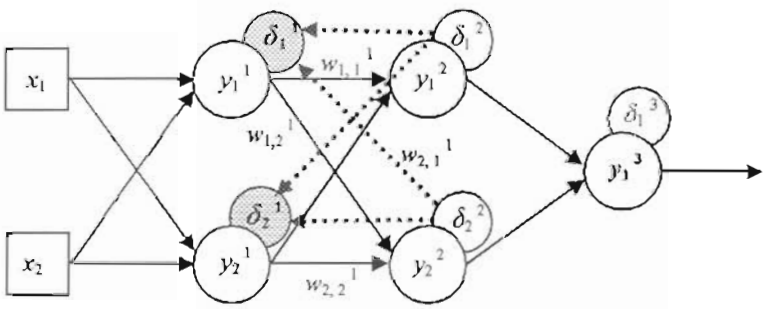
$$\delta_2^{\,2} = w_{2,1}^{\,2}\,\delta_1^{\,3}$$




ภาพที่ 2-13 การคำนวณค่าผิดพลาดในชั้นซ่อนชั้นที่ 2

$$\delta_1^{\,1} = (w_{1,1}^{\,1}\,\delta_1^{\,2} + w_{1,2}^{\,1}\,\delta_2^{\,2})$$

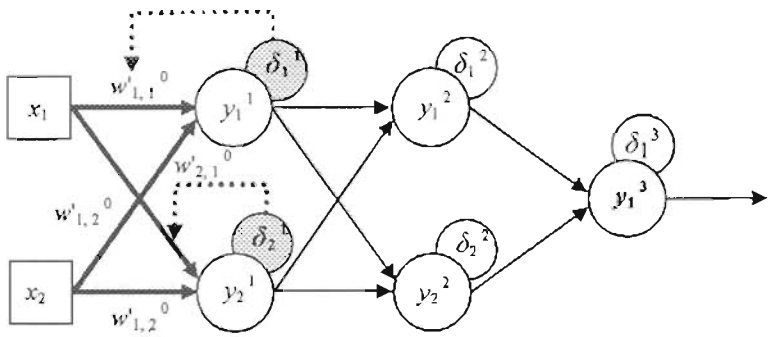
$$\delta_2^{\,1} = (w_{2,1}^{\,1}\,\delta_1^{\,2} + w_{2,2}^{\,1}\,\delta_2^{\,2})$$




ภาพที่ 2-14 การคำนวณค่าผิดพลาดในชั้นซ่อนชั้นที่ 1

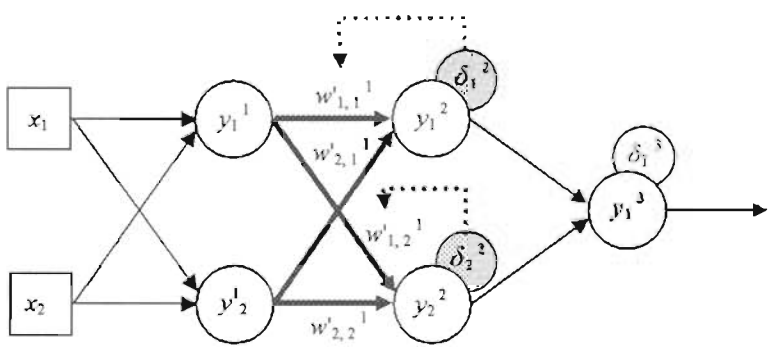
เมื่อคำนวณค่าผิดพลาดของแต่ละหน่วยประมวลผลเรียบร้อยแล้ว ขั้นตอนต่อไปคือการปรับค่าน้ำหนักการเชื่อมโยง โดยการนำค่าน้ำหนักการเชื่อมโยงเดิม ($w_{i,j}$) บวกเพิ่มด้วยค่าการเปลี่ยนแปลงซึ่งคำนวณได้จากการคูณของค่าอัตราการเรียนรู้ (Learning Rate) $\rho$ กับค่าผิดพลาดของหน่วยประมวลผล $\delta_i$ และคูณด้วยค่าอนุพันธ์ของฟังก์ชันกระตุ้น $dy_i(u)/du$ และคูณด้วยข้อมูลเข้าของค่าน้ำหนักการเชื่อมโยงนั้น ๆ ขั้นตอนการคำนวณของการปรับค่าน้ำหนักการเชื่อมโยงแสดงในภาพที่ 2-15 ภาพที่ 2-16 และภาพที่ 2-17 ตามลำดับ

$$
\begin{aligned}
w'_{1,1}{}^{0} &= w_{1,1}{}^{0} + \rho\,\delta_1{}^{1}\,\frac{dy_1{}^{1}(u)}{du}\,x_1 \\
w'_{2,1}{}^{0} &= w_{2,1}{}^{0} + \rho\,\delta_1{}^{1}\,\frac{dy_1{}^{1}(u)}{du}\,x_2 \\
w'_{1,2}{}^{0} &= w_{1,2}{}^{0} + \rho\,\delta_2{}^{1}\,\frac{dy_2{}^{1}(u)}{du}\,x_1 \\
w'_{2,2}{}^{0} &= w_{2,2}{}^{0} + \rho\,\delta_2{}^{1}\,\frac{dy_2{}^{1}(u)}{du}\,x_2
\end{aligned}
$$




ภาพที่ 2-15 การปรับค่าน้ำหนักการเชื่อมโยงในชั้นซ่อนชั้นที่ 1

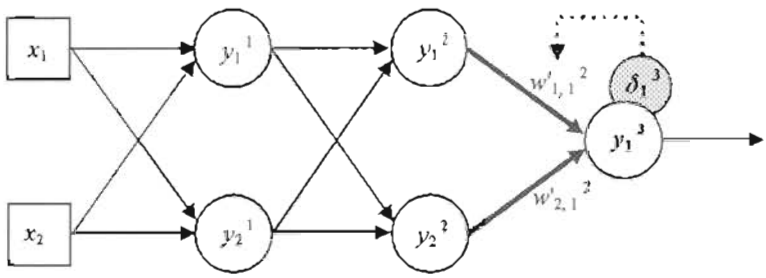
$$
\begin{aligned}
w'_{1,1}{}^{1} &= w_{1,1}{}^{1} + \rho\,\delta_1{}^{2}\,\frac{dy_1{}^{2}(u)}{du}\,y_1{}^{1} \\
w'_{2,1}{}^{1} &= w_{2,1}{}^{1} + \rho\,\delta_2{}^{2}\,\frac{dy_1{}^{2}(u)}{du}\,y_2{}^{1} \\
w'_{1,2}{}^{1} &= w_{1,2}{}^{1} + \rho\,\delta_2{}^{2}\,\frac{dy_2{}^{2}(u)}{du}\,y_1{}^{1} \\
w'_{2,2}{}^{1} &= w_{2,2}{}^{1} + \rho\,\delta_2{}^{2}\,\frac{dy_2{}^{2}(u)}{du}\,y_2{}^{1}
\end{aligned}
$$




ภาพที่ 2-16 การปรับค่าน้ำหนักการเชื่อมโยงในชั้นซ่อนชั้นที่ 2

$$w'_{1,1}{}^{2} = w_{1,1}{}^{2} + \rho\,\delta_1{}^{3}\,\frac{dy_1{}^{3}(u)}{du}\;\; y_1{}^{2}$$

$$w'_{2,1}{}^{2} = w_{2,1}{}^{2} + \rho\,\delta_1{}^{3}\,\frac{dy_1{}^{3}(u)}{du}\;\; y_2{}^{2}$$



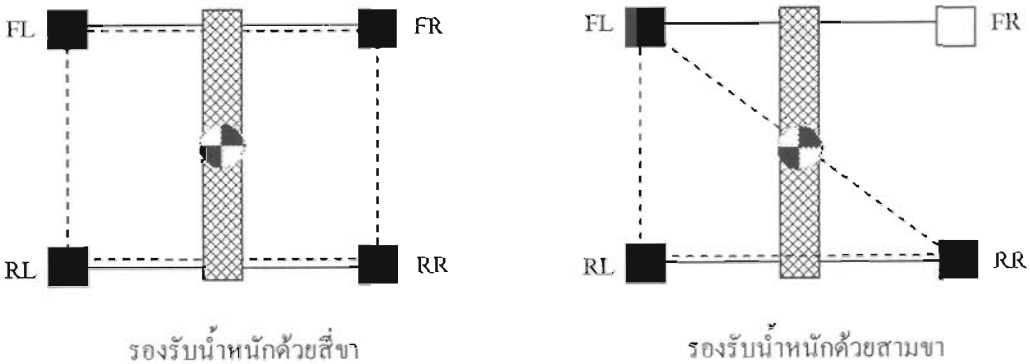

ภาพที่ 2-17 การปรับค่าน้ำหนักการเชื่อมโยงในชั้นข้อมูลออก

แต่ละรอบของการฝึกสอนข่ายงาน ชุดข้อมูลสำหรับฝึกสอนจะถูกป้อนเข้าข่ายงานทีละรายการ และใช้กระบวนการฝึกสอนเพื่อปรับค่าน้ำหนักการเชื่อมโยง ดังนั้นใน 1 รอบการฝึกสอนจึงมีการปรับค่าน้ำหนักการเชื่อมโยงเป็นจำนวนเท่ากับชุดข้อมูลที่ใช้ฝึกสอน

## 2.2 การเดินของหุ่นยนต์สี่ขา

### 2.2.1 บริเวณรองรับน้ำหนัก

การเดินแบบ Wave Gait ใช้สามขารองรับน้ำหนักและใช้หนึ่งขาก้าวเพื่อไปอยู่ในตำแหน่งใหม่ การรองรับน้ำหนักด้วยสามขาอาจส่งผลให้หุ่นยนต์เสียการทรงตัวได้ ถ้าจุดศูนย์กลางมวล (Center of Mass) ของหุ่นยนต์ไม่อยู่ในบริเวณรองรับ (Supportting Area) ดังนั้นเมื่อต้องการยกขาข้างใดขึ้นเพื่อจะทำการก้าวขานั้นจำเป็นต้องคำนึงถึงตำแหน่งของสามขาที่ใช้รองรับน้ำหนักด้วย เนื่องจากรูปสามเหลี่ยมที่เกิดขึ้นสัมพันธ์โดยตรงกับตำแหน่งของขาทั้งสามขาที่ใช้ ดังแสดงในภาพที่ 2-18 โดยภาพด้านซ้ายแสดงให้เห็นถึงการรองรับน้ำหนักด้วยขาทั้ง 4 ข้าง บริเวณรองรับน้ำหนักที่เกิดขึ้นจึงเป็นรูปสี่เหลี่ยมและจุดศูนย์กลางมวลอยู่ภายในบริเวณรองรับน้ำหนัก ภาพด้านขวาแสดงการรองรับน้ำหนักด้วยขา 3 ข้างเกิดเป็นบริเวณรองรับน้ำหนักรูปสามเหลี่ยม ส่วนภาพที่ 2-19 แสดงรูปสามเหลี่ยมของบริเวณรองรับน้ำหนักที่ต่างกัน เมื่อยกขาต่างกัน โดยในภาพด้ายซ้ายคือความต้องการที่จะยกขาหน้าด้านขวา (Front Right : FR) แต่ไม่สามารถยกขึ้นได้เนื่องจากรูปสามเหลี่ยมของบริเวณรองรับน้ำหนักที่เกิดขึ้นไม่ทำให้จุดศูนย์กลางมวลอยู่ภายในได้ ส่วนภาพด้านขวาแม้ว่าตำแหน่งของเท้าทั้ง 4 ข้างจะอยู่ที่ตำแหน่งเดียวกับภาพด้านซ้ายแต่จะสามารถยกขาหลังด้านขวาได้

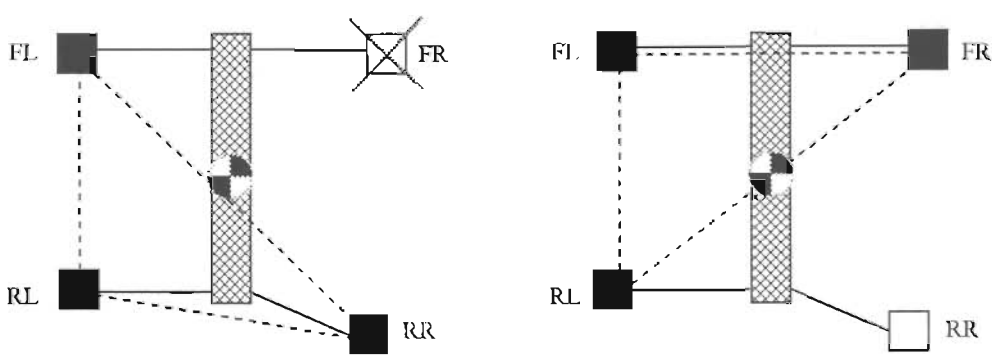



รองรับน้ำหนักด้วยสี่ขา

รองรับน้ำหนักด้วยสามขา

เท้าสัมผัสพื้น
เท้ายกขึ้น
ลำตัว
จุดศูนย์กลางมวล
ขาช่วงบน
แนวบริเวณรองรับน้ำหนัก
FR ขาหน้าข้างขวา Front Right
FL ขาหน้าข้างซ้าย Front Left
RR ขาหลังข้างขวา Rear Right
RL ขาหลังข้างซ้าย Rear Left

ภาพที่ 2-18 ตำแหน่งการวางเท้าและบริเวณรองรับน้ำหนัก




ยกขา FR ไม่ได้เนื่องจากจุดศูนย์กลางมวล อยู่นอกรูปสามเหลี่ยม

ยกขา RR ได้

ภาพที่ 2-19 รูปสามเหลี่ยมของบริเวณรองรับน้ำหนักที่ต่างกัน

2.2.2 ลักษณะการเดิน

สัตว์สี่ขาตามธรรมชาติมีรูปแบบการเดินที่หลากหลาย เนื่องจากองค์ประกอบและอวัยวะที่ใช้สำหรับการเดินและทรงตัวที่มีความเหมาะสมและเอื้อต่อการดำรงชีวิต แต่สำหรับหุ่นยนต์สามารถจำแนกรูปแบบการเดินที่พบมากในการทดลองการเดินของหุ่นยนต์สี่ขาได้ 2 แบบคือ แบบก้าวทีละขา หรือการเดินแบบ Wave Gait และการเดินแบบก้าวทีละ 2 ขา หรือการเดินแบบ Trot Gait การเดินแบบ Wave Gait เป็นการเดินแบบสมดุลสถิต (Static Walking) โดยจะยกขาเพื่อก้าวเดินครั้งละ 1 ข้าง การเดินแบบนี้ทำให้เกิดบริเวณรองรับน้ำหนักขึ้น จึงทำให้สามารถจัดตำแหน่งจุดศูนย์กลางมวลให้อยู่ภายในบริเวณรองรับน้ำหนักได้ตลอดการเดิน การเดินแบบนี้ทำให้หุ่นยนต์เดินได้ช้าแต่สามารถหยุดเดินที่สถานะใดก็ได้โดยไม่เสียการทรงตัว ส่วนการเดินแบบ Trot Gait เป็นการเดินแบบสมดุลพลวัตร (Dynamic Walking) โดยจะยกขาเพื่อก้าวเดินครั้งละ 2 ข้าง ที่อยู่ในแนวทแยงมุม ทำให้มีขาเพียง 2 ข้างที่ใช้รองรับน้ำหนักซึ่งไม่สามารถทำให้เกิดบริเวณรองรับน้ำหนักได้ หุ่นยนต์จึงต้องก้าวขาที่ยกอย่างรวดเร็วเพื่อก้าวไปยังตำแหน่งที่ต้องการก่อนที่จะเสียการทรงตัว การเดินแบบนี้ทำให้เดินได้เร็วกว่าแบบ Wave Gait แต่ไม่สามารถหยุดที่สถานะใดในระหว่างการยกขาและการก้าวขาได้ เนื่องจากเป็นการเดินที่ไม่มีบริเวณรองรับน้ำหนัก

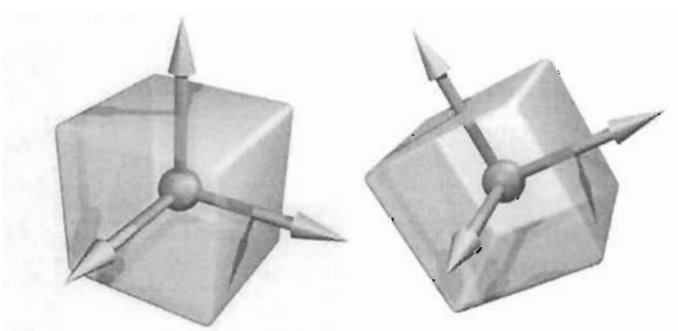
## 2.3 Open Dynamics Engine

Open Dynamics Engine (ODE) พัฒนาโดย Russell Smith เป็นเครื่องมือสำหรับช่วยสร้างระบบจำลองที่เกี่ยวข้องกับผลกระทบเชิงพลวัตรหรือกฎทางฟิสิกส์ โดยเป็น Library เสริมเข้ามาช่วยในการพัฒนาโปรแกรม เช่น ภาษา C/C++ Python หรือ Ruby ซึ่ง ODE พัฒนาด้วยภาษา C++ บนพื้นฐานของ OpenGL ได้รับการยอมรับว่ามีมาตรฐาน นำไปใช้งานได้โดยไม่มีค่าลิขสิทธิ์ และเปิดเผยรหัสโปรแกรม อีกทั้งทำงานได้โดยไม่ขึ้นกับแพลทฟอร์ม ดังนั้นจึงสามารถนำระบบที่จำลองไปทำงานได้บนระบบปฏิบัติการต่างๆ เช่น ลินุกซ์ วินโดวส์ หรือ Mac OS [8] ตัวอย่างลักษณะของงานที่นำ ODE ไปใช้ เช่น การจำลองการเคลื่อนไหวของวัตถุแข็งที่เชื่อมต่อกัน (Articulated Rigid Body) การจำลองยานพาหนะแบบล้อเลื่อนที่เคลื่อนที่ไปบนพื้น (Ground Vehicles) การจำลองลักษณะของสิ่งมีชีวิตที่มีขา (Legged Creatures) รวมทั้งจำลองการเคลื่อนไหวของวัตถุในสิ่งแวดล้อมเสมือนจริง ODE เป็น Library ที่ทำงานได้อย่างรวดเร็ว ยืดหยุ่นและทนทาน ถูกออกแบบมาเพื่อใช้งานในระบบ Interactive หรือสามารถติดต่อกับส่วนของผู้ใช้งานได้แบบทันทีทันใด (Real Time) จึงเหมาะสำหรับการนำมาใช้จำลองระบบที่วัตถุเคลื่อนที่ไปในสิ่งแวดล้อมที่เปลี่ยนแปลงได้ เนื่องจาก ODE มีความเร็ว ความทนทาน และเสถียร อีกทั้งผู้ใช้สามารถเปลี่ยนแปลงโครงสร้าง

ของวัตถุในระบบที่จำลองได้แม้ในขณะที่ระบบกำลังทำงาน และมีส่วนของการตรวจสอบการสัมผัสหรือชนกันของวัตถุ ซึ่งเป็นฟังก์ชันพื้นฐานที่สามารถนำไปใช้ ได้อย่างมีประสิทธิภาพ

ในระบบหุ่นยนต์จะนำคุณสมบัติของ ODE มาช่วยในการสร้างแบบจำลองของหุ่นยนต์ เพื่อใช้ในการวิจัยแทนการใช้หุ่นยนต์จริง เนื่องจาก ODE ได้รับการยอมรับว่าเป็น Library ที่มีมาตรฐาน ดังนั้นจึงพบว่ามีงานวิจัย โปรแกรมประยุกต์ รวมทั้งเกมส์จำนวนมากที่ใช้ความสามารถของ ODE ทดสอบการทำงานของโปรแกรมหรืออัลกอริทึมที่ออกแบบ [8, 10, 11] ส่วนประกอบพื้นฐานของ ODE ประกอบด้วย World, Space, Body และ Geom (Geometry) โดยในส่วนของ World จะควบคุมปัจจัยที่เกี่ยวข้องกับระบบที่จำลอง เช่น ค่าแรงโน้มถ่วง (Gravity) มวล (Mass) ส่วนใหญ่แล้วในระบบที่จำลองจะมีเพียง 1 World เท่านั้น แต่ก็สามารถกำหนดให้มีมากกว่า 1 ได้ แต่วัตถุที่อยู่ต่าง World จะไม่สามารถติดต่อกันได้ ส่วนของ Space จะจัดการเกี่ยวกับการชนกันของวัตถุ ส่วนของ Body หมายถึง รูปทรงต่าง ๆ ที่ประกอบเป็นวัตถุ ในระบบ โดยสามารถกำหนดน้ำหนัก ขนาด และตำแหน่งได้ ส่วนของ Geom คือ ค่าตำแหน่งและมิติของวัตถุต่าง ๆ ในระบบใช้เพื่อตรวจสอบการชนหรือสัมผัส (Collision) ของวัตถุ


**ภาพที่ 2-20** ส่วนประกอบพื้นฐานของการใช้งาน ODE



**ภาพที่ 2-21** ตัวอย่างวัตถุแข็งและแกนสามมิติ

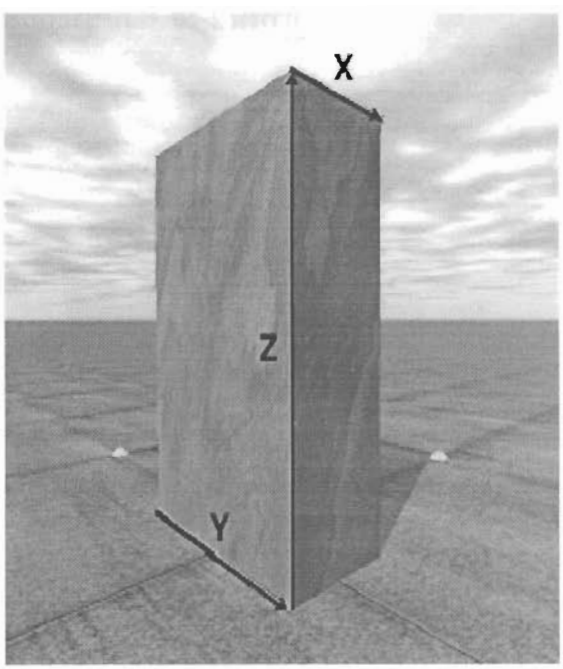
2.3.1วัตถุแข็ง (Rigid Body)

วัตถุแข็งคือวัตถุที่มีคุณสมบัติของลักษณะภายนอกที่แข็ง ไม่เปลี่ยนแปลงรูปร่าง ตัวอย่างของวัตถุแข็งที่สร้างด้วย ODE แสดงในภาพที่ 2-21 ในทางคอมพิวเตอร์กราฟฟิกส์การจำลองวัตถุ

แข็งจะเกี่ยวข้องกับการกำหนดและควบคุมค่าคุณลักษณะของวัตถุแข็ง ซึ่งแบ่งได้เป็น 2 ประเภท คือ คุณลักษณะที่เปลี่ยนแปลงตามเวลาในขณะทำการจำลองมี 4 ค่า คือ 1) จุดอ้างอิง (Point of Reference ) เป็นค่าที่ใช้อ้างอิงตำแหน่งของวัตถุในระบบแกน 3 มิติ คือจะเป็นค่าตำแหน่งบนแกน x, y และ z โดยจุดอ้างอิงนี้เป็นจุดเดียวกับตำแหน่งของจุดศูนย์กลางมวลของวัตถุ เวกเตอร์ตำแหน่งของจุดอ้างอิงเขียนแทนด้วย (x, y, z) 2) ความเร็วเชิงเส้น (Linear Velocity) เป็นค่าความเร็วเชิงเส้นของวัตถุแข็ง โดยกำหนดไว้ที่จุดอ้างอิงตำแหน่งวัตถุ เวกเตอร์ของความเร็วเชิงเส้นเขียนแทนด้วย (vx, vy, vz) 3) ทิศทางของวัตถุ (Orientation of a Body) คือทิศทางของวัตถุที่เปลี่ยนแปลงในขณะการจำลอง ซึ่งการเปลี่ยนแปลงทิศทางของวัตถุทำได้ 2 แบบ คือแบบแรกใช้ Quaternion Vector ที่เขียนแทนด้วย (qs, qx, qy, qz) ส่วนแบบที่ 2 คือใช้เมตริกซ์การหมุน (Rotation Matrix) ขนาด 3x3 คูณกับเวกเตอร์ตำแหน่งของจุดอ้างอิง 4) ความเร็วเชิงมุม (Angular Velocity) เป็นค่าทิศทางของวัตถุในแต่ละแกนที่เปลี่ยนแปลงในขณะทำการจำลอง เวกเตอร์ของความเร็วเชิงมุมเขียนแทนด้วย (wx, wy, wz) ส่วนคุณลักษณะของวัตถุแข็งที่ไม่เปลี่ยนแปลงตามเวลามี 3 ค่า คือ 1) น้ำหนักของวัตถุ (Mass of the Body) 2) ตำแหน่งของจุดศูนย์กลางมวล (Position of the Center of Mass) ที่จะเป็นจุดเดียวกับจุดอ้างอิงของวัตถุเสมอ 3) Inertia Matrix เป็นเมตริกซ์ขนาด 3x3 ใช้กำหนดการกระจายน้ำหนักของวัตถุออกไปรอบ ๆ จุดศูนย์กลางมวล

**Creating Simple Object**

```
// width, length, and height of the box
dReal x_size = 0.3;
dReal y_size = 0.6;
dReal z_size = 0.9;
Object* obj = newObject();
// create the box's body
dMass m;
dMassSetBoxTotal
(&m,1,x_size,y_size,z_size);
obj->body = dBodyCreate(world);
dBodySetMass (obj->body,&m);
dBodySetPosition (obj-
>body,0,0,z_size/2.0);
// create a geom and attach it.
dGeomID g = dCreateBox
(space,x_size,y_size,z_size);
addGeom(obj, g, 1,1,0); // make it
yellow (1.1,0)
```

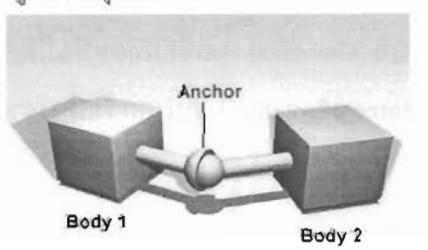



ภาพที่ 2-22 โปรแกรมตัวอย่างสำหรับสร้างวัตถุแข็งด้วย ODE

2.3.2 จุดเชื่อมต่อและการบังคับ (Joints and constraints)

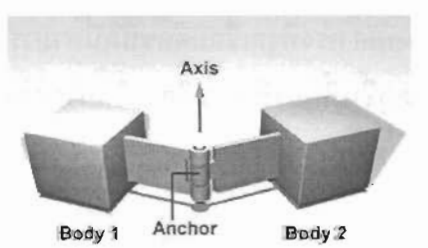
จุดเชื่อมต่อที่พบเห็นในชีวิตประจำวันส่วนมากจะเป็นจุดเชื่อมต่อแบบบานพับ (Hinge) เช่น ที่ใช้ยึดประตู หน้าต่าง เป็นต้น ใน ODE จุดเชื่อมต่อคือสิ่งสำคัญที่ใช้เชื่อมต่อวัตถุ 2 ชิ้นเข้าด้วยกัน ซึ่งจะทำให้วัตถุทั้ง 2 ชิ้นที่นำมาเชื่อมต่อกันนั้นมีความสัมพันธ์และสามารถบังคับควบคุมกันได้ โดยค่าที่วัตถุหนึ่งบังคับไปยังวัตถุที่เชื่อมต่อกันคือ ตำแหน่งและทิศทาง ใน ODE จุดเชื่อมต่อและการบังคับจะถูกใช้รวมกันและจำแนกออกเป็น 5 ชนิดได้แก่

2.3.2.1 จุดเชื่อมต่อแบบ Ball and Socket จุดเชื่อมต่อชนิดนี้ที่วัตถุชิ้นแรกจะเป็นส่วนของจุดเชื่อมต่อที่เป็น Socket ส่วนวัตถุชิ้นที่ 2 จะเป็นลักษณะของ Ball คล้ายกับข้อต่อหัวไหล่ของมนุษย์ ซึ่งจุดเชื่อมต่อแบบ Ball and Socket นี้จะบังคับให้วัตถุชิ้นที่ 2 สามารถเปลี่ยนตำแหน่งและทิศทางเป็นรูปกรวยที่มีฐานติดอยู่กับวัตถุชิ้นแรก



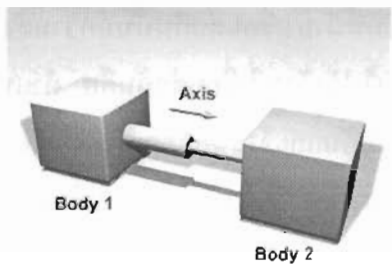

**ภาพที่ 2-23** ตัวอย่างจุดเชื่อมต่อแบบ Ball and Socket

2.3.2.2 จุดเชื่อมต่อแบบ Hinge เมื่อนำวัตถุสองชิ้นมาเชื่อมต่อกันด้วยจุดเชื่อมต่อชนิดนี้แล้วจะทำให้วัตถุทั้งสองชิ้นใช้แกนหมุนร่วมกันเพียงแกนเดียว และวัตถุทั้งสองชิ้นสามารถเคลื่อนที่ในลักษณะของการหมุนได้รอบแกนที่ใช้ร่วมกันเท่านั้น คล้ายกับข้อต่อส่วนข้อศอกของมนุษย์



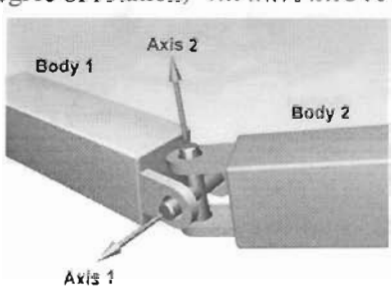

**ภาพที่ 2-24** ตัวอย่างจุดเชื่อมต่อแบบ Hinge

2.3.2.3 จุดเชื่อมต่อแบบ Slider ลักษณะของจุดเชื่อมต่อชนิดนี้ วัตถุสองชิ้นที่เชื่อมต่อกันสามารถเปลี่ยนตำแหน่งเข้าหา หรือออกห่างจากกันได้ในลักษณะของการเลื่อนตามทิศทางของแกนที่กำหนดไว้



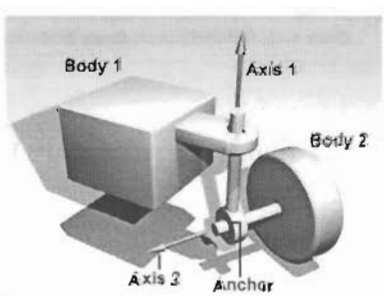

**ภาพที่ 2-25** ตัวอย่างจุดเชื่อมต่อแบบ Slider

2.3.2.4 จุดเชื่อมต่อแบบ Universal จุดเชื่อมต่อชนิดนี้คล้ายกับจุดเชื่อมต่อแบบ Ball and socket แต่จะมีองศาการหมุน (degree of rotation) ที่มากกว่าเนื่องจากมีแกนที่ใช้จำนวน 2 แกน




**ภาพที่ 2-26** ตัวอย่างจุดเชื่อมต่อแบบ Universal

2.3.2.5 จุดเชื่อมต่อแบบ Hinge-2 จุดเชื่อมต่อชนิดนี้ใช้จุดเชื่อมต่อแบบ Hinge จำนวน 2 จุดมาเชื่อมเข้าด้วยกันแบบอนุกรมทำให้เกิดเป็นจุดเชื่อมต่อแบบพิเศษที่มีแกนจำนวน 2 แกน แต่จะต่างจากจุดเชื่อมต่อแบบ Universal ตรงที่จุดเชื่อมต่อชนิดนี้สามารถกำหนดให้จุดเชื่อมต่อจุดที่ 2 อยู่ห่างจากจุดแรกได้ จึงทำให้วัตถุชิ้นที่สองมีความอิสระในการหมุนมากกว่าจุดเชื่อมต่อแบบ Universal



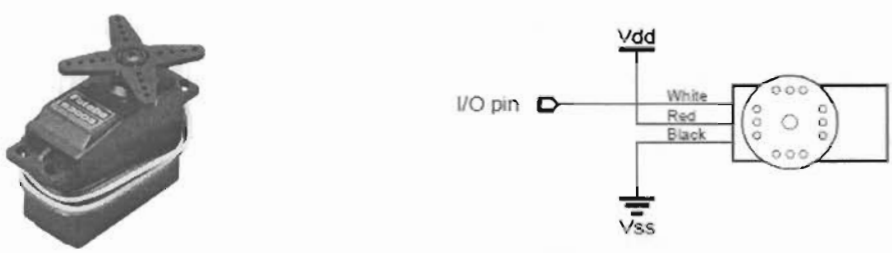

**ภาพที่ 2-27** ตัวอย่างจุดเชื่อมต่อแบบ Hing-2

ในการจำลองระบบที่ประกอบด้วยวัตถุเชื่อมต่อกัน ส่วนประกอบที่สำคัญคือ วัตถุแข็งรูปทรงต่างๆ เช่น กล่องสี่เหลี่ยม ทรงกลมกระบอก หรือรูปทรงอื่นที่กำหนดขึ้นเองตามความต้องการของผู้ใช้และกำหนด Geom ให้กับวัตถุที่สร้างขึ้นเพื่อใช้ตรวจสอบการชนของวัตถุในระบบ จากนั้นเชื่อมต่อวัตถุเหล่านั้นด้วยจุดเชื่อมต่อชนิดที่เหมาะสม เพื่อทำให้การกระทำกับวัตถุหนึ่งส่งผลกระทบกับวัตถุที่เชื่อมต่อด้วย Joint ตามประเภทและการกำหนดตำแหน่งของการเชื่อมต่อ ส่วนกระบวนการในการจำลองการเคลื่อนไหววัตถุแข็งตามค่าเวลาที่เปลี่ยนไปนั้นคือการทำ Integration โดยแต่ครั้งของการทำ Integration ค่าสถานะของวัตถุจะถูกปรับใหม่และนำไปแสดงในค่าเวลาถัดไปทำให้เกิดเป็นภาพเคลื่อนไหว

## 2.4 อุปกรณ์ทางฮาร์ดแวร์

### 2.4.1 เซอร์โวมอเตอร์ (Servo Motor)

เป็นมอเตอร์ที่มีระบบเฟืองทดรอบอยู่ภายใน เพื่อให้มีแรงบิดสูงขึ้น และเป็นมอเตอร์ที่มีการจำกัดองศาการหมุน ไม่สามารถหมุนครบรอบเหมือนมอเตอร์ประเภทอื่น เซอร์โวมอเตอร์จะจำกัดการหมุนไว้ที่ 90 องศา หรือ 180 องศาขึ้นกับเซอร์โวมอเตอร์แต่ละรุ่น เซอร์โวมอเตอร์ยังมีลักษณะเด่นอีกอย่างที่แตกต่างจากมอเตอร์ทั่วไปคือจะพยายามรักษาตำแหน่งองศาของมุมที่กำหนดไว้ ในภาพที่ 2-28 ทางด้านซ้ายแสดงภาพถ่ายของเซอร์โวมอเตอร์ขนาดเล็กของ Futaba รุ่น S3003 และในด้านขวาแสดงรายละเอียดของสายที่ใช้เชื่อมต่อ



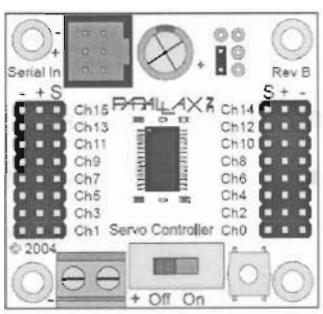

**ภาพที่ 2-28** เซอร์โวมอเตอร์

เซอร์โวมอเตอร์จะมีสายเชื่อมต่อด้วยกันสามเส้นโดยเส้นที่ 1 (Black) เป็นสายขั้วลบ เส้นที่ 2 (Red) เป็นสายขั้วบวก เส้นที่ 3 (White) เป็นสายสัญญาณควบคุม วิธีการควบคุมจะใช้วิธีการส่งสัญญาณพัลส์ (pulse) ที่มีจำนวน 60 ลูกคลื่นในหนึ่งวินาทีตลอดการทำงานเพื่อเป็นคำสั่งให้เซอร์โวมอเตอร์ทำงาน และสามารถกำหนดตำแหน่งการหมุนของเซอร์โวมอเตอร์ได้จากพัลส์ที่ส่งออกไปมีความกว้างอยู่ระหว่าง 1 มิลลิวินาทีจนถึง 2 มิลลิวินาที ถ้าเซอร์โวมอเตอร์เป็นแบบหมุนได้ 90 องศา ความกว้างพัลส์ที่ 1 มิลลิวินาทีจะหมุนไปที่ตำแหน่ง -45 องศา ตำแหน่งความกว้างพัลส์ที่ 1.5

มิลลิวินาทีจะหมุนไปที่ตำแหน่ง 0 องศา ความกว้างพัลส์ที่ 2 มิลลิวินาทีจะหมุนไปที่ตำแหน่ง +45 องศา ส่วนเซอร์โวมอเตอร์แบบหมุนได้ 180 องศา ความกว้างพัลส์ที่ 1 มิลลิวินาทีจะหมุนไปที่ตำแหน่ง -90 องศา ความกว้างพัลส์ที่ 1.5 มิลลิวินาทีจะหมุนไปที่ตำแหน่ง 0 องศา และความกว้างพัลส์ที่ 2 มิลลิวินาทีจะหมุนไปที่ตำแหน่ง +90 องศา

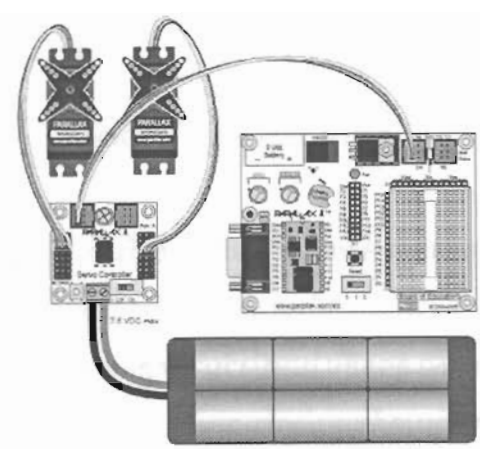
2.4.2 แผงวงจรควบคุมเซอร์โวมอเตอร์ (Servo Controller Board) [9]

แผงวงจรควบคุมเซอร์โวมอเตอร์ที่ใช้ในการทดลองทำหน้าที่ควบคุมเซอร์โวมอเตอร์ทุกตัวที่ใช้ขับเคลื่อนหุ่นยนต์ โดยใช้แผงวงจรที่ชื่อ Parallax Servo Controller (PSC) ซึ่งผลิตจากบริษัท Parallax Inc. สามารถควบคุมเซอร์โวมอเตอร์ได้ 16 ตัวพร้อมกัน และสามารถรายงานค่าองศาการหมุนของเซอร์โวมอเตอร์แต่ละตัวได้




**ภาพที่ 2-29** แผงวงจรควบคุมเซอร์โวมอเตอร์ Parallax Servo Controller [9]

แผงวงจรควบคุมเซอร์โวมอเตอร์ สามารถควบคุมเซอร์โวมอเตอร์ให้รักษาค่าตำแหน่งไว้ได้หลังจากส่งคำสั่งควบคุมไปแล้ว ซึ่งช่วยให้เซอร์โวมอเตอร์ไม่หมุนหรือเปลี่ยนตำแหน่งถ้าไม่มีการสั่งงาน โดยมีรูปแบบการเชื่อมต่อเพื่อใช้งานดังแสดงในภาพที่ 2-30



**ภาพที่ 2-30** การเชื่อมต่อและใช้งานแผงวงจรควบคุมเซอร์โวมอเตอร์ [9]

การใช้งานแผงวงจรควบคุมเซอร์โวมอเตอร์ ต้องส่งข้อความแบบอนุกรมจากอุปกรณ์ที่เชื่อมต่ออยู่กับแผงวงจรควบคุม ซึ่งอาจจะเป็นเครื่องคอมพิวเตอร์โดยผ่านทางพอร์ตอนุกรมหรือใช้ไมโครคอนโทรลเลอร์โดยผ่านทาง Output Port การส่งข้อความทั้งสองแบบใช้มาตรฐาน RS-232 สำหรับคำสั่งที่สำคัญในการใช้งานมี 2 คำสั่งคือ คำสั่งควบคุมให้เซอร์โวมอเตอร์หมุนไปยังตำแหน่งที่ต้องการ และคำสั่งสำหรับให้เซอร์โวมอเตอร์รายงานค่าตำแหน่งปัจจุบัน

รูปแบบคำสั่งในการควบคุมให้มอเตอร์หมุนไปยังตำแหน่งที่ต้องการ

Syntax : "!SC" C R pw.LOWBYTE, pw.HIGHBYTE, $0D

Reply : none

!SC คือข้อความที่ส่งไปยังแผงวงจรควบคุมเซอร์โวมอเตอร์เพื่อให้รู้ว่าเป็นการสั่งให้หมุนเซอร์โวมอเตอร์ C คือพารามิเตอร์ สำหรับระบุหมายเลขเซอร์โวมอเตอร์ที่จะควบคุม มีค่าระหว่าง 0-31 นั่นคือสามารถควบคุมเซอร์โวมอเตอร์ได้ 32 ตัวพร้อมกัน (ในกรณีที่ใช้แผงวงจรควบคุมเซอร์โวมอเตอร์ 2 แผง) R คือพารามิเตอร์ สำหรับระบุค่าความเร็วในการหมุนของเซอร์โวมอเตอร์ มีค่าระหว่าง 0 – 63 โดย 0 หมายถึงหมุนด้วยความเร็วสูงสุด 63 หมายถึงความเร็วน้อยที่สุด ส่วน p เป็นค่าพารามิเตอร์สำหรับระบุตำแหน่งที่จะให้เซอร์โวมอเตอร์หมุน มีค่าระหว่าง 250 – 1250 แทนการหมุนไปยังตำแหน่ง 0 – 180 องศา ส่วน $0D เป็นส่วนปิดข้อความเพื่อให้รู้ว่าข้อความจบเพียงเท่านี้ คำสั่งนี้ไม่มีการคืนค่า

รูปแบบคำสั่งในการร้องขอให้มีการรายงานค่าตำแหน่งของเซอร์โวมอเตอร์

Syntax : "!SCRSP" x $0D

Reply : x y z

!SCRSP คือข้อความที่ส่งไปยังแผงวงจรควบคุมเซอร์โวมอเตอร์เพื่อให้รู้ว่าเป็นการร้องขอให้รายงานค่าตำแหน่งของเซอร์โวมอเตอร์ x คือหมายเลขเซอร์โวมอเตอร์ที่ต้องการทราบค่าตำแหน่งสำหรับค่าที่คืนกลับมาคือ x แทนหมายเลขเซอร์โวมอเตอร์ที่รายงานค่าตำแหน่งกลับมา y และ z เป็นค่าตำแหน่งของเซอร์โวมอเตอร์