การควบคุมการเคลื่อนที่ของรถหุ่นยนต์ด้วยเส้น

ในการศึกษาการทำงานของหุ่นยนต์นั้น หัวใจสำคัญที่สุดอยู่ที่ขั้นตอนของการเขียนโปรแกรมสำหรับควบคุมการทำงานของหุ่นยนต์ให้สามารถทำงานตามเงื่อนไขที่เรากำหนดไว้ได้ ซึ่งการที่จะสามารถสร้างระบบหุ่นยนต์ให้มีความเฉลียวฉลาดได้มากน้อยเพียงใดนั้น ปัจจัยหลักอย่างหนึ่งก็คือ การติดตั้งอุปกรณ์ตรวจจับแบบต่างๆให้กับหุ่นยนต์ เพื่อใช้ในการตรวจจับสภาพแวดล้อมภายนอกรอบๆตัวหุ่นยนต์ อันจะเป็นหนทางเพื่อนำไปสู่การวิเคราะห์และตัดสินใจสั่งงานระบบกลไกต่างๆเพื่อตอบสนองต่อเงื่อนไขต่างๆที่ตรวจจับได้ สำหรับระบบโครงสร้างของรถหุ่นยนต์ ET-ROBOT RD2 ก็เช่นเดียวกัน ในชุดมาตรฐานจะมีการติดตั้งอุปกรณ์สำหรับตรวจจับ Input แบบต่างๆไว้ 2 ส่วนด้วยกัน คือ Switch สำหรับตรวจสอบการชน ด้านหน้าและด้านหลัง และแผงวงจรสำหรับตรวจจับเส้นขนาด 3 จุด สำหรับใช้นำทางให้รถหุ่นยนต์เคลื่อนที่ไปตามแนวเส้น ซึ่งในบทนี้เราจะมาทดลองศึกษาเรียนรู้เกี่ยวกับวิธีการในการตรวจจับเส้นของรถหุ่นยนต์ ET-ROBOT RD2 กัน โดยจะทดลองกำหนดสถานะการณ์ของเส้นในลักษณะต่างๆเพื่อใช้เป็นสนามในการทดสอบประสิทธิภาพในการควบคุมการทำงานของรถหุ่นยนต์ ไม่ว่าจะเป็น การเคลื่อนที่ไปตามแนวเส้นตรงและเส้นโค้ง การเคลื่อนที่ไปตามเส้นตรงและเส้นมุมฉาก การเคลื่อนที่ไปตามแนวเส้นตรงและเส้นสลับฟันปลา หรือ การตรวจจับเส้นเพื่อควบคุมให้รถเคลื่อนที่อยู่ภายในกรอบของเส้นเป็นต้น

สำหรับแนวทางและวิธีการในการเขียนโปรแกรมเพื่อใช้ควบคุมการเคลื่อนที่ของรถหุ่นยนต์ให้เคลื่อนที่ไปตามแนวของเส้นที่กำหนดเอาไว้ ก็เป็นอีกรูปแบบหนึ่งของการเขียนโปรแกรมสำหรับควบคุมการเคลื่อนที่ของรถหุ่นยนต์อย่างมีเงื่อนไข ซึ่งการที่จะสามารถเขียนโปรแกรมเพื่อควบคุมให้รถหุ่นยนต์เคลื่อนที่ไปตามแนวเส้นที่กำหนดไว้นั้น ในอันดับแรกผู้อ่านจะต้องเข้าใจถึงพื้นฐานในการควบคุมการเคลื่อนที่ของรถหุ่นยนต์เสียก่อน ไม่ว่าจะเป็นการเคลื่อนที่ไปข้างหน้า เคลื่อนที่ถอยหลัง การควบคุมให้รถเลี้ยวซ้าย หรือเลี้ยวขวา เป็นต้น ซึ่งในบทนี้จะไม่ขอกล่าวถึงวิธีการในเรื่องเหล่านี้ โดยผู้อ่านสามารถย้อนกลับไปทำความเข้าใจในหัวข้อการทดลองเรื่อง “การควบคุมการเคลื่อนที่ของรถหุ่นยนต์” ได้ และจำเป็นอย่างยิ่งที่ผู้อ้านจะต้องทำความเข้าใจกับวิธีการควบคุมการเคลื่อนที่ของรถหุ่นยนต์ในลักษณะต่างๆดังได้กล่าวมาแล้วข้างต้น ไม่เช่นนั้นแล้วอาจไม่สามารถเข้าใจและทำการทดลองในหัวข้อนี้ได้อย่างมีประสิทธิภาพเท่าที่ควร

สำหรับหลักการและวิธีการในการเขียนโปรแกรมเพื่อใช้ควบคุมการเคลื่อนที่ของตัวรถหุ่นยนต์ให้เคลื่อนที่ไปตามแนวเส้นนั้น ความจริงแล้วจะไม่มีหลักเกณฑ์หรือหลักการใดๆเป็นตัวบังคับว่าจะต้องทำอย่างนั้น อย่างนี้ จึงจะถูกต้อง ถ้าทำอย่างนั้นจะผิดต้องทำอย่างนี้จึงจะถูก ซึ่งวิธีการต่างๆสามารถปรับเปลี่ยนไปได้ตามสถานะการณ์และความเหมาะสม โดยผู้อ่านสามารถคิดค้นและพัฒนาเทคนิควิธีต่างๆขึ้นมาใช้งานเองได้ สำหรับตัวอย่างที่จะแสดงให้เห็นในหัวข้อการทดลองของบทนี้ ถือว่าเป็นเพียงตัวอย่างหรือแนวทางเบื้องต้น ในการเขียนโปรแกรมควบคุมเท่านั้น ซึ่งผู้อ่านอาจใช้เป็นแนวทางในการทดลองเพื่อประกอบความเข้าใจในเบื้องต้น ซึ่งถ้าเห็นว่ามีข้อดีก็สามารถจดจำและนำไปใช้งานในโอกาสต่อไปได้ หรือ

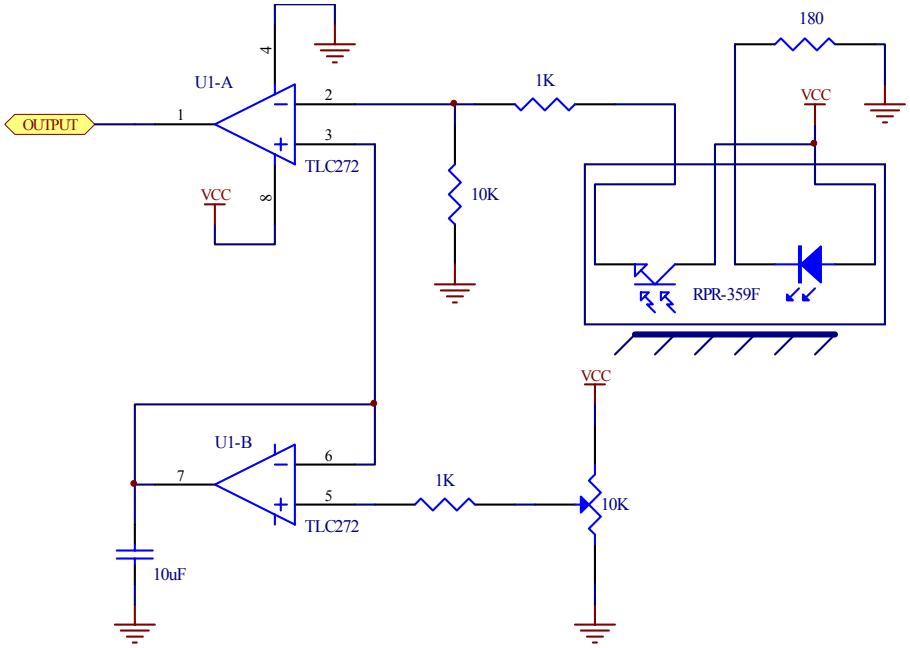
ถ้าผู้อ่านยังเห็นว่าตัวอย่างยังมีข้อจำกัดหรือข้อบกพร่องอยู่ก็อาจนำไปพัฒนาปรับปรุงให้มีความสามารถมากยิ่งขึ้น หรือถ้าหากผู้อ่านสามารถคิดค้นหรือมีวิธีการอื่นๆที่ดีกว่า ก็สามารถพัฒนาเทคนิคเหล่านั้นขึ้นมาใช้งานทดแทนเองก็ได้ไม่ถือว่าผิดอะไร โดยทุกอย่างควรยึดเอาผลการทำงานของโปรแกรมเป็นเครื่องชี้วัด ว่าการทำงานของโปรแกรมสามารถตอบสนองต่อเงื่อนไขที่กำหนดไว้ได้เพียงใดมากกว่า

แต่อย่างไรก็ตามก่อนที่จะเริ่มต้นเข้าสู่การทดลองนั้น ในอันดับแรกเราควรมาทำความเข้าใจกับการทำงานของวงจรที่ใช้ในการตรวจจับเส้นกันเสียก่อนว่า วงจรมีหลักการทำงานเป็นอย่างไร และให้ผลลัพธ์ของการทำงานสัมพันธ์กับลักษณะของเส้นที่ตรวจจับได้อย่างไรบ้าง จากนั้นค่อยไปทำความเข้าใจกับวิธีการเขียนโปรแกรมเพื่อตรวจสอบเงื่อนไขการตรวจจับเส้น เพื่อใช้เป็นทางเลือกในการสั่งงานรถหุ่นยนต์ให้เคลื่อนที่ไปในทิศทางต่างๆในภายหลัง

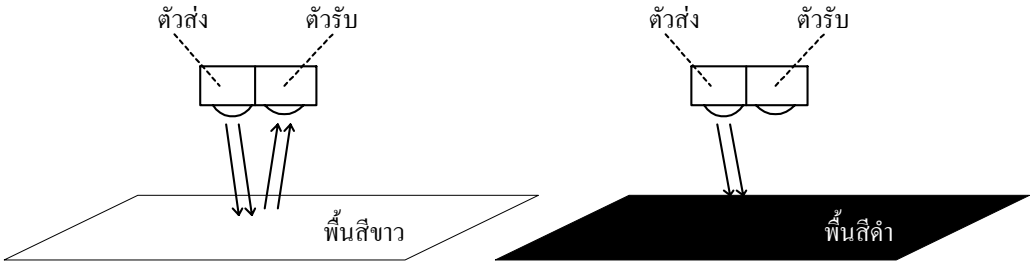
การตรวจจับเส้น (TRACKER SENSOR)

ในการตรวจจับเส้น (TRACKER) นั้น บอร์ด ET-ROBOT RD2 ได้จัดเตรียมขั้วต่อ TRACKER3 ซึ่งเป็นขั้วต่อสำหรับเชื่อมต่อกับบอร์ดตรวจจับเส้นขนาด 3 จุด ของอีทีที รุ่น R-TRACKER3 ไว้ให้ภายในบอร์ดแล้ว โดยบอร์ดตรวจจับเส้น รุ่น R-TRACKER3 นั้น จะให้ผลการทำงานของการตรวจจับเป็นสัญญาณโลจิก กล่าวคือจะให้ผลการทำงานเพียง 2 สภาวะ คือ จริง กับ เท็จ โดยบอร์ด R-TRACKER3 เป็นวงจรตรวจจับเส้น โดยจะใช้หลักการส่งและสะท้อนกลับของแสงอินฟราเรด ซึ่งภายในบอร์ด R-TRACKER3 นั้นจะมีการติดตั้งชุดตรวจจับเส้นแบบอินฟราเรดไว้ 3 จุด ด้วยกัน เพื่อเพิ่มความสามารถในการตรวจจับให้มากขึ้น โดยจะอาศัยหลักการในการสะท้อนกลับของสัญญาณอินฟราเรด ซึ่งคุณสมบัติของแสงอินฟราเรดนี้จะไม่มีการสะท้อนกลับในวัตถุหรือพื้นผิวที่เป็นสีดำ ดังนั้นเราจึงสามารถทำการตรวจสอบหรือแยกความแตกต่างระหว่างวัตถุหรือพื้นผิวที่เป็นสีขาวและสีดำได้ แต่เนื่องจากในสถานที่ต่างๆ นั้นจะมีความเข้มข้นหรือปริมาณของแสงที่ไม่เท่ากัน ซึ่งจะมีผลต่อการทำงานของวงจรอินฟราเรด ดังนั้นเราจึงได้ออก แบบวงจรตรวจจับให้สามารถทำการปรับระดับความไวในการตรวจจับได้ โดยการปรับที่ตัวต้านทานปรับค่าได้แบบเกือกมาค่า 10KOHM เพื่อให้มีความอ่อนตัวเมื่อนำไปใช้ในสถานที่ต่างๆ มากขึ้น

รูปแสดง ลักษณะของแผงวงจรสำหรับตรวจจับเส้น (R-TRACKER3 BOARD)



รูปแสดง วงจรสำหรับตรวจจับเส้นขนาด 1 จุด (TRACKER CIRCUIT)



รูปแสดง ลักษณะการสะท้อนของคลื่นอินฟราเรด

จากรูปแสดงลักษณะการสะท้อนของตัวตรวจจับเส้นโดยใช้ชุดรับส่งอินฟราเรดข้างต้น จะเห็นได้ว่าเมื่อตัวส่งทำการส่งคลื่นอินฟราเรดออกไปกระทบกับวัตถุก็จะมีการสะท้อนกลับของสัญญาณไปยังตัวรับ แต่ขนาดความแรงของสัญญาณที่สะท้อนกลับนั้นจะขึ้นอยู่กับสีพื้นผิวของวัสดุ โดยในพื้นผิวที่มีสีทึบ เช่น สีดำ จะมีการสะท้อนกลับของสัญญาณน้อยมาก ส่วนสีที่มีความสว่าง เช่น สีขาว จะมีการสะท้อนสัญญาณได้ดี จากคุณสมบัติดังกล่าวเราจึงสามารถแยก ความแตกต่างของเส้นได้ สำหรับวงจรของชุดตรวจจับเส้น R-TRACKER3 นั้น จะใช้วงจรเปรียบเทียบแรงดัน (OP-AMP) ในการตรวจจับ โดยการปรับค่าแรงดันอ้างอิงของการตรวจจับไว้ล่วงหน้าโดยใช้ตัวต้านทานปรับค่าได้แบบเกือกม้าเพื่อนำไปเปรียบเทียบกับแรงดันที่ได้จากการสะท้อนกลับของ ตัวรับ อินฟราเรด เมื่อมีการสะท้อนกลับของคลื่นอินฟราเรดจำนวนมากจนได้

ค่าแรงดันเท่ากับแรงดันอ้างอิงที่กำหนดไว้จะได้ผลการเปรียบเทียบเป็นค่าโลจิก “0” แต่ถ้าการสะท้อนกลับมีค่าน้อยจะให้ผลเป็น “1” ซึ่ง บอร์ด ET-ROBOT RD2 จะมีหลอดแสดงผลแบบ LED สำหรับใช้แสดงผลการตรวจจับเส้นเตรียมไว้ให้ด้วย โดยหลอดแสดงผล LED จะติดสว่างเมื่อผลการตรวจจับเส้นได้ผลลัพธ์เป็น “0” โดยผลการตรวจจับจะมีการเปลี่ยนแปลงตามการสะท้อนของสัญญาณอินฟราเรดดังนี้

- เมื่อมีการสะท้อน (พื้นผิวสีขาว) ที่ขาสัญญาณเอาต์พุตจะมี Logic “0” และ LED ติดสว่าง
- เมื่อไม่มีการสะท้อน (พื้นผิวสีดำ) ที่ขาสัญญาณเอาต์พุตจะมี Logic “1” และ LED จะดับ

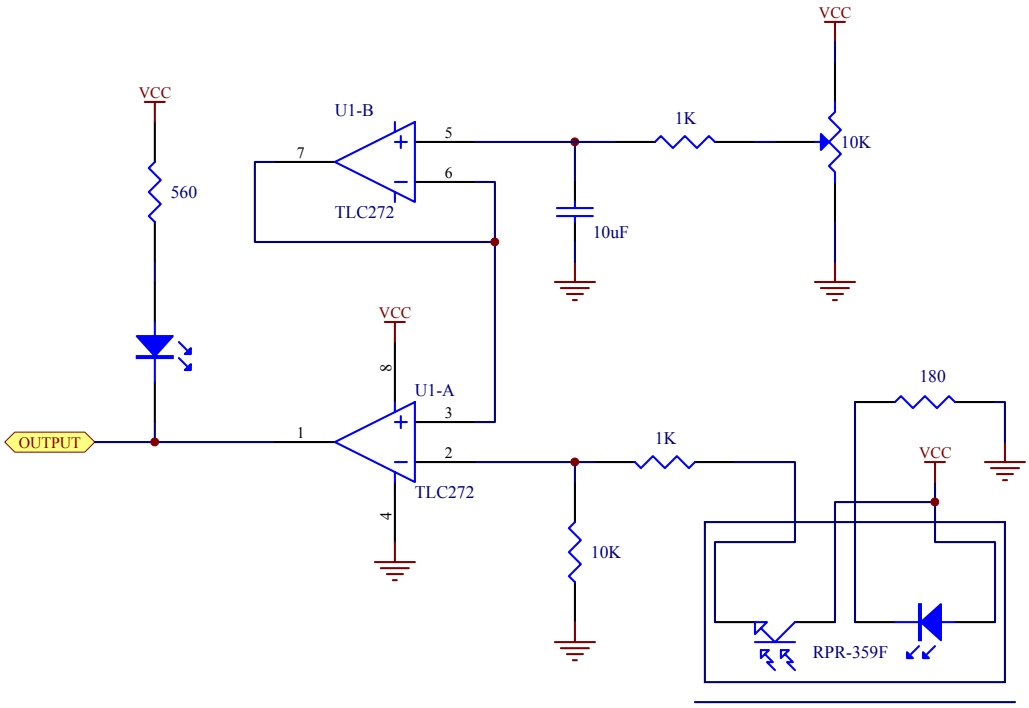
สำหรับการปรับแต่งค่าระดับความเข้มของเส้นในการตรวจจับนั้น จะขึ้นอยู่กับลักษณะความแตกต่างของพื้นและเส้น ซึ่งควรใช้พื้นและเส้นที่มีความแตกต่างกันอย่างชัดเจน เช่น ใช้พื้นสีขาวกับเส้นสีดำ หรือ ใช้พื้นสีดำกับเส้นสีขาว เป็นต้น โดยวิธีการปรับแต่งค่าความไว้ในการตรวจจับเส้นนั้นให้ทำการนำรถหุ่นยนต์ที่ติดตั้งแผงตรวจจับเส้นแรียบร้อยแล้วไปวางทับไว้กับเส้น พร้อมกับปรับค่าตัวต้านทานจนได้จุดที่เกิดความแตกต่างของวงจรตรวจจับพอดี แล้วจึงลองนำ Sensor เคลื่อนที่ผ่านแนวเส้นดู ซึ่งจะต้องเห็นการเปลี่ยนแปลงของ LED ที่แสดงสถานะการตรวจจับเส้นทำงานได้อย่างถูกต้อง เมื่อ Sensor อยู่ในตำแหน่งทับเส้น และ อยู่นอกเส้น ตัวอย่างเช่น ถ้าใช้พื้นสีขาวกับเส้นสีดำ เมื่อ Sensor ทับอยู่บนเส้น LED จะต้องดับ แต่เมื่อ Sensor อยู่บนพื้น LED แสดงสถานะการทำงานของ Sensor จะต้องติดสว่าง เป็นต้น

การตรวจสอบการทำงานของชุดตรวจจับเส้น

สำหรับวงจรตรวจจับเส้น ของบอร์ด R-TRACKER3 นั้น จะเลือกใช้ Sensor เบอร์ RPR-359 ซึ่งเป็น Sensor แบบ อินฟราเรด โดยตัวโมดูล ของ RPR-359 นั้นจะประกอบไปภาคส่งและภาครับคลื่นอินฟราเรด รวมอยู่ในตัวเดียวกัน โดยในด้านของภาคส่งนั้นจะมีลักษณะโครงสร้างเป็น LED แบบอินฟราเรด ซึ่งใช้เป็นตัวส่ง และ ส่วนของภาครับอินฟราเรดนั้น จะมีโครงสร้างเป็นแบบ Transistor โดยลักษณะการทำงานของ Sensor จะใช้หลักการสะท้อนกลับของคลื่นอินฟราเรดที่ส่งออกไป ซึ่งคุณสมบัติของคลื่นอินฟราเรดนั้นจะสามารถสะท้อนได้ดีกับวัตถุที่มีสีอยู่ในโทนสว่าง เช่นสีขาว ส่วนในวัตถุที่มีสีอยู่ในโทนมืด เช่น สีดำ การสะท้อนของคลื่นอินฟราเรดจะอยู่ในระดับต่ำมากหรือแทบไม่มีเลย ซึ่งจากคุณสมบัติเช่นนี้ เราจึงนำเอาตัว Sensor เบอร์ RPR-359 นี้มาประยุกต์ใช้ในการตรวจจับความแตกต่างของเส้น เพื่อใช้กับรถหุ่นยนต์ ET-ROBOT RD2 ได้

ซึ่งตามปรกติแล้วถ้านำ Sensor เบอร์ RPR-359 มาใช้ในการตรวจจับเส้น จะอาศัยหลักการตรวจจับขนาดของแรงดันทางไฟฟ้าที่ได้จากการนำกระแสของทราสซิสเตอร์ โดยระดับแรงดันจะเปลี่ยนแปลงตามความเข้มของคลื่นอินฟราเรดที่สะท้อนกลับมาเข้ายังภาครับ (ขา Base ของทรานซิสเตอร์) ซึ่งถ้าการสะท้อนอยู่ในปริมาณมากทรานซิสเตอร์จะนำกระแสมากจะได้ระดับแรงดันสูง ยิ่งถ้าทรานซิสเตอร์นำกระแสมากจนถึงจุดอิ่มตัวจะได้ค่าแรงดันประมาณเท่ากับแหล่งจ่าย +VCC เลยทีเดียว แต่ถ้าการสะท้อนมี

ปริมาณน้อยระดับแรงดันจะมีค่าต่ำ ซึ่งถ้าน้อยมากจนทรานซิสเตอร์ไม่สามารถนำกระแสได้ก็จะไม่มีแรงดันที่ขา Emitter เลย ซึ่งจะเห็นได้ว่าถ้าใช้เฉพาะตัวโมดูล RPR-359 ในการตรวจจับสีของเส้นเพียงอย่างเดียวนั้น จะต้องใช้วงจร A/D ในการตรวจวัดระดับความแตกต่างของแรงดันจากทรานซิสเตอร์ และต้องเขียนโปรแกรมตรวจสอบค่าที่อ่านได้จาก A/D เพื่อตัดสินใจอีกครั้งหนึ่ง ซึ่งจะทำให้เกิดความยุ่งยากในการเขียนโปรแกรมมากขึ้นตามไปด้วย ดังนั้นทางอีทีที จึงได้ออกแบบวงจรสำหรับทำหน้าที่ เปรียบเทียบแรงดันเพิ่มเติมเข้าไปเพื่อทำการเปรียบเทียบระดับแรงดันที่ได้จากการสะท้อนกลับของคลื่นอินฟราเรดกับค่าแรงดันอ้างอิงที่กำหนดไว้ภายในวงจร โดยผลการเปรียบเทียบจะได้ผลลัพธ์เป็นค่าแบบโลจิก ซึ่งจะง่ายต่อการตรวจสอบมากขึ้น โดยลักษณะการจัดวงจรของ RPR-359 สำหรับตรวจจับเส้นนั้นเป็นดังรูป



รูปแสดง ลักษณะของวงจรสำหรับตรวจจับเส้น

จากวงจร จะเห็นได้ว่า Sensor เบอร์ RPR-359 นั้น จะถูกจัดวงจรให้ทำงานอยู่ตลอดเวลาทั้งส่วนของภาคส่ง(LED) และภาครับ(Transistor) โดยระดับแรงดันที่ขา Emitter ของทรานซิสเตอร์จาก RPR-359 นั้น จะมีค่าสูงถ้ามีการสะท้อนของคลื่นอินฟราเรดกลับมายังภาครับในปริมาณมาก และ ระดับแรงดันจะมีค่าน้อยลงถ้ามีการสะท้อนของคลื่นอินฟราเรดในระดับต่ำๆ ซึ่งเพื่อให้ง่ายต่อการตรวจสอบการทำงานของ Sensor จะเห็นว่าในวงจรจะใช้ OP-AMP ทำหน้าที่เปรียบเทียบแรงดันที่ได้จากขา Emitter ของ Sensor กับระดับแรงดันอ้างอิงที่เรากำหนดไว้จากการปรับตัวต้านทานปรับค่าได้ โดยผลของการเปรียบเทียบจะได้ค่าเป็น “0” เมื่อระดับแรงดันที่มาจากขา Emitter ของ Sensor มีค่ามากกว่าแรงดันอ้างอิงที่ตั้งไว้ โดย LED แสดงผลจะติดสว่าง แต่ถ้าหากว่าการสะท้อนกลับของคลื่นอินฟราเรดมีระดับต่ำมาก จะทำให้

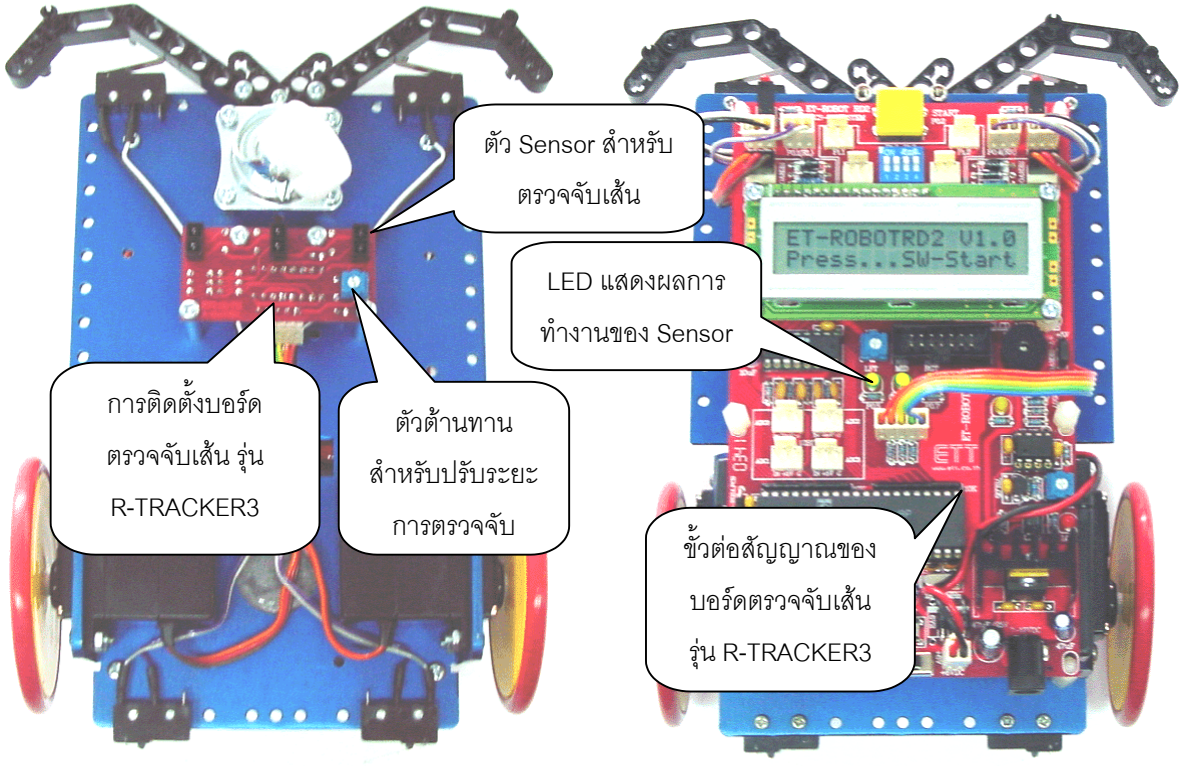
ทรานซิสเตอร์ไม่นำกระแสหรือนำกระแสเพียงเล็กน้อย ซึ่งก็จะส่งผลให้ค่าระดับแรงดันที่ขา Emitter ของทรานซิสเตอร์มีค่าต่ำกว่าค่าแรงดันอ้างอิงที่กำหนดไว้ ดังนั้นผลลัพธ์จากการเปรียบเทียบแรงดันจะได้เป็นโลจิก “1” โดยการแสดงผลการทำงานของหลอด LED จะดับ

ซึ่งจากคุณสมบัติของวงจรตรวจจับเส้นจะเห็นได้ว่า ผลการสะท้อนของคลื่นอินฟราเรดจะขึ้นอยู่กับปัจจัย หลัก 2 ข้อ คือ สีของวัตถุ และ ระยะความห่างของวัตถุ ที่ใช้เป็นตัวสะท้อนคลื่นอินฟราเรด โดยการทำงานของวงจรสามารถทำงานได้ดี ถ้าใช้ในการตรวจจับสีที่มีความแตกต่างกันข้อนข้างมาก เช่นใช้ในการตรวจจับสีของเส้นและสีของพื้นที่มีความแตกต่างกันมากๆ เช่น เส้นสีดำ กับ พื้นสีขาว แต่อย่างไรก็ตามวงจรสามารถปรับค่าแรงดันอ้างอิงในการตรวจจับสีของเส้นได้ เพื่อให้เกิดความเหมาะสมกับสภาพการใช้งานจริงๆ ทั้งนี้ก็เนื่องมาจากว่าในสถานะการณ์จริงนั้น บางครั้งระยะความสูงของ Sensor จากพื้นถึงตัว Sensor รวมทั้งสีของพื้นและเส้นอาจมีความแตกต่างกัน ซึ่งสามารถทำการปรับค่าแรงดันอ้างอิงของการเปรียบเทียบที่เหมาะสมได้ ดังนั้นจึงสามารถดัดแปลงวงจรไปใช้ในการตรวจจับสีอื่นๆที่มีความแตกต่างกันระหว่างสีในโทนมืดกับสีในโทนสว่างได้ เช่น ใช้ในการตรวจจับเส้นสีน้ำเงิน กับพื้นสีเทา เป็นต้น โดยวงจรสำหรับตรวจจับเส้นที่ใช้ในบอร์ด R-TRACKER3 นั้นจะมีอยู่ด้วยกัน 3 จุด เพื่อใช้ตรวจจับเส้นที่อยู่ในตำแหน่ง ซ้าย กลาง และ ขวา ตามลำดับ

โดยในการใช้งานนั้น ตามปรกติจะใช้ Sensor ตัวที่อยู่ในตำแหน่งกลางเป็นตัวตรวจจับเส้นเพื่อบังคับให้รถหุ่นยนต์เคลื่อนที่ไปตามแนวเส้น ส่วน Sensor ตัวที่อยู่ด้านซ้ายและขวา จะใช้สำหรับตรวจจับเส้นเพื่อบังคับทิศทางการเลี้ยวของรถให้อยู่ในเส้น ตัวอย่างเช่น ถ้า Sensor ตัวกลางวางทับอยู่ในแนวเส้นก็บังคับให้รถหุ่นยนต์เคลื่อนที่ไปข้างหน้า แต่ถ้าตรวจพบว่า Sensor ตำแหน่งทางซ้ายวางทับแนวเส้นอยู่ก็แสดงว่ารถหุ่นยนต์เริ่มเอียงไปทางขวาของเส้นแล้ว ก็ต้องสั่งให้รถหุ่นยนต์เลี้ยวกลับมาทางซ้ายเพื่อให้อยู่ในแนวเส้นตามเดิม อย่างนี้เป็นต้น

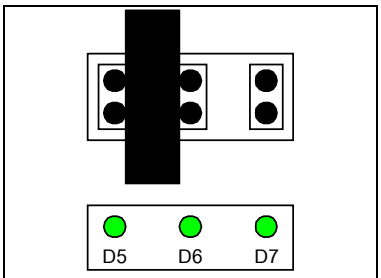
ในการทดสอบการทำงานของชุดตรวจจับเส้น R-TRACKER3 นั้น ผู้ใช้สามารถทดสอบการทำงานในเบื้องต้นได้ โดยการนำวัตถุไปบังตัวตรวจจับเส้น หรือ นำตัวรถหุ่นยนต์ไปวางทับเส้นสีดำที่วางพาดผ่านอยู่บนพื้นสีขาวพร้อมกับสังเกตุการทำงานของ LED แสดงผลของชุด R-TRACKER ที่อยู่บนบอร์ด ET-ROBOT RD2 ซึ่งถ้าทุกอย่างถูกต้องจะพบว่า เมื่อตำแหน่งของตัวตรวจจับวางทับอยู่บนสีขาว LED จะติดสว่าง(“0”) แต่เมื่อตัวตรวจจับวางอยู่บนสีดำ LED จะดับ (“1”) ถ้าผลการทำงานไม่เป็นดังที่กล่าวมาให้ทดลองทำการปรับค่าความต้านทานที่อยู่บนแผงวงจร R-TRACKER3 ใหม่ โดยในการปรับนั้นอาจทดลองใช้นิ้วปิดที่ตำแหน่งของตัวตรวจจับเส้นในระยะห่างประมาณ 1ซม. แล้วจึงทำการปรับค่าตัวต้านทานจนถึงตำแหน่งรอยต่อระหว่างการติดและดับของ LED โดยให้ปรับไปทางตำแหน่งเริ่มต้นของการติดสว่างของ LED เล็กน้อย จากนั้นอาจทดลองเลื่อนมือไปมาเพื่อสังเกตุ การ ติด-ดับ ของ LED ซึ่งถ้าปรับค่าตัวต้านทานอยู่ในตำแหน่งที่เหมาะสมแล้ว เมื่อตำแหน่งของนิ้วมือไปบังตัวตรวจจับ LED จะติดสว่าง และเมื่อเลื่อนนิ้วมือหลบไป LED จะต้องดับ ซึ่งถ้าถูกต้องก็ให้ทดลองนำตัวตรวจจับเส้นไปทดสอบกับเส้นจริงๆซ้ำอีกครั้ง

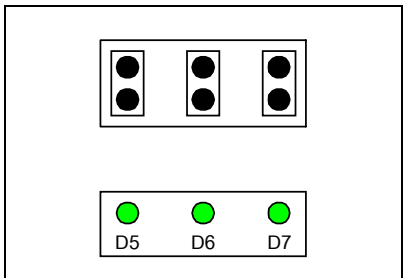
หนึ่งจนได้ผลที่ถูกต้อง แต่ต้องไม่ลืมสังเกตุตำแหน่งการติดและดับของ LED ในตำแหน่ง ซ้าย กลาง และ ขวา ด้วยว่า สามารถติดและดับ ตรงกับตัว Sensor ที่อยู่ในตำแหน่ง ซ้าย กลาง และ ขวาเหมือนกันหรือไม่ เพื่อป้องกันการติดตั้งบอร์ด Sensor สลับตำแหน่งหรือกลับทางระหว่างด้านซ้ายและขวา

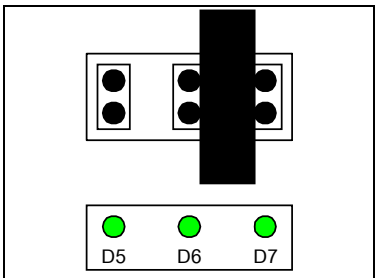


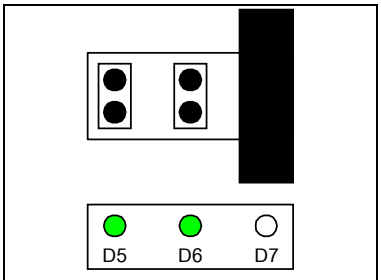
รูปแสดง การติดตั้งใช้งานบอร์ด R-TRACKER3 เพื่อใช้งานกับบอร์ด ET-ROBOT RD2

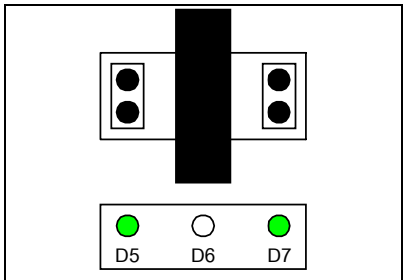
รูปแสดงตัวอย่างสนามสำหรับทดสอบหุ่นยนต์เดินตามเส้น

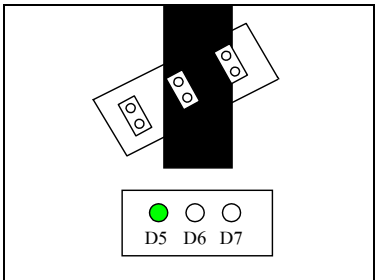


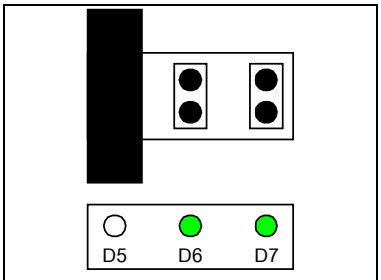


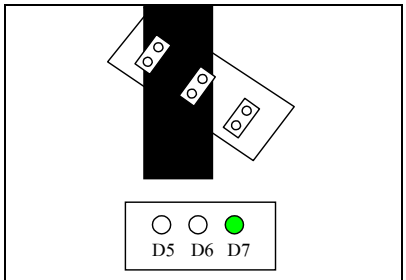


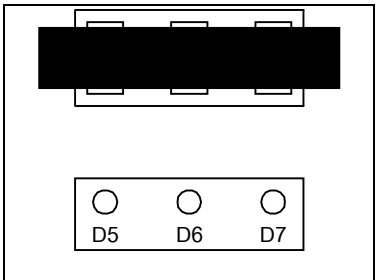












สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“0”	“0”	“0”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“0”	“0”	“0”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“0”	“0”	“0”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“0”	“0”	“1”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“0”	“1”	“0”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“0”	“1”	“1”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“1”	“0”	“0”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“1”	“1”	“0”

สัญญาณที่ได้จาก Sensor		
ซ้าย	กลาง	ขวา
“1”	“1”	“1”

○ = LED ดับที่ขาพอร์ต = 1 ● = LED ติดสว่างที่ขาพอร์ต = 0

ลักษณะ ตัวอย่างการตรวจจับเส้นของรถหุ่นยนต์เมื่ออยู่ในแนวเส้นตรงและเส้นโค้ง

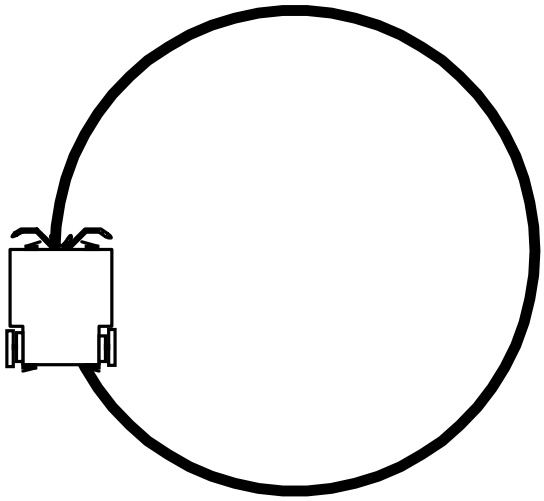
การเขียนโปรแกรมควบคุมให้รถหุ่นยนต์เคลื่อนที่ไปตามเส้นตรงและเส้นโค้ง

ในการเขียนโปรแกรมเพื่อควบคุมให้รถหุ่นยนต์เคลื่อนที่ไปตามแนวเส้นนั้น ลักษณะของเส้นที่เป็นแนวตรงและเส้นโค้ง ถือว่าเป็นสถานะการณ์ที่ง่ายที่สุด โดยสามารถใช้เงื่อนไขง่ายๆในการตรวจสอบและควบคุม สำหรับหลักการในการตรวจสอบ Sensor ที่ใช้ตรวจจับเส้น เพื่อวิเคราะห์เงื่อนไขในการบังคับให้รถหุ่นยนต์หยุดหรือเคลื่อนที่ไปในทิศทางต่างๆนั้นจะอาศัยเงื่อนไขความเป็นไปได้ของผลการตรวจจับเส้นของชุด R-TRACKER3 ดังตัวอย่างต่อไปนี้

สถานะของ Sensor			สถานะการณ์ความเป็นไปได้ของการตรวจจับเส้น (สีดำ)	การแก้ไขหรือสั่งงานเพื่อควบคุมการเคลื่อนที่ให้อยู่ในแนวเส้น
ซ้าย	กลาง	ขวา		
“0”	“0”	“0”	อยู่นอกเส้นหรืออาจอยู่ในแนวเส้น	เดินหน้าต่อไปตามปรกติ
“0”	“0”	“1”	ใกล้จะออกนอกเส้นไปทางซ้าย	เลี้ยวขวากลับเข้าเส้นอย่างเร็ว
“0”	“1”	“0”	อยู่ในแนวเส้นตำแหน่งกึ่งกลางพอดี	เดินหน้าต่อไปตามปรกติ
“0”	“1”	“1”	อยู่ในแนวเส้นแต่เอียงไปทางซ้าย	เลี้ยวขวากลับเข้าเส้นแบบปรกติ
“1”	“0”	“0”	ใกล้จะออกนอกเส้นไปทางขวา	เลี้ยวซ้ายกลับเข้าเส้นอย่างเร็ว
“1”	“0”	“1”	เป็นไปไม่ได้(กรณีเส้นตรงและโค้ง)	เดินหน้าต่อไปตามปรกติ
“1”	“1”	“0”	อยู่ในแนวเส้นแต่เอียงไปทางขวา	เลี้ยวซ้ายกลับเข้าเส้นแบบปรกติ
“1”	“1”	“1”	ทับแนวเส้นหรือพบจุดที่เป็นแยก	เลี้ยวขวาหรือซ้ายเพื่อค้นหาแนวเส้น

ตาราง สรุปความเป็นไปได้ของการตรวจจับเส้น (ในกรณีใช้กับเส้นสีดำ)

หมายเหตุ ค่าสถานะการในตาราง ใช้สำหรับทดสอบการทำงานในกรณีที่สนามเป็นเส้นตรงและเส้นโค้ง โดยใช้กับแนวเส้นที่เป็นสีดำ ซึ่งในกรณีนี้ เมื่อ Sensor ตัวใดทับแนวเส้นอยู่จะให้สถานะเป็นโลจิกเป็น “1”(LED แสดงสถานะจะดับ) แต่ถ้าตำแหน่ง Sensor ไม่อยู่ในแนวเส้นจะให้สถานะเป็น โลจิก “0” (LED แสดงสถานะติดสว่าง)



ภาพแสดงลักษณะตัวอย่างสนามที่จะใช้ในการทดสอบ

สำหรับแนวทางในการเขียนโปรแกรมเพื่อตรวจสอบแนวเส้นเพื่อนำมาวิเคราะห์และสั่งงานให้รถสามารถเคลื่อนที่ไปตามแนวเส้นตามทิศทางที่ต้องการได้นั้น จะใช้หลักการอ่านค่าสถานะของโลจิกที่ได้จากวงจรตรวจจับเส้น (R-TRACKER3) สำหรับบอร์ด ET-ROBOT RD2 นั้นผลการทำงานของวงจรตรวจจับเส้นจะต่อเข้ากับ P0.5-P0.7 ตามลำดับ

- P0.5 = ตัวตรวจจับเส้นตำแหน่งด้านซ้ายของรถ
- P0.6 = ตัวตรวจจับเส้นตำแหน่งกลางของรถ
- P0.7 = ตัวตรวจจับเส้นตำแหน่งด้านขวาของรถ

สำหรับโปรแกรม BASCOM-8051 นั้นสามารถใช้วิธีการอ่านค่าสถานะ Input จากพอร์ต I/O ของ CPU ได้โดยตรง ทั้ง 4 พอร์ต คือ P0,P1,P2 และ P3 ตามลำดับ โดยเมื่อต้องการอ่านค่า Input จากพอร์ตใด ในอันดับแรกจะต้องทำการเขียนค่า “1” ออกไปยังพอร์ตนั้นๆก่อนแล้วจึงอ่านค่าสถานะของพอร์ตกลับมาในภายหลัง โดยในการอ่านค่าหรือเขียนค่าให้กับพอร์ตนั้น ในกรณีของภาษาเบสิก BASCOM-8051 จะใช้เครื่องหมายเท่ากับ (=) ในการรับค่า(Input) และ ให้ค่า(Output) ดังรูปแบบต่อไปนี้

```
Port = Var      'เขียนค่า Output ให้พอร์ต เช่น P0 = &HFF (เขียนค่า “1” ให้พอร์ต P0 ทุกบิต)
Var = Port      'อ่านค่า Input จากพอร์ต เช่น In_Buf = P0 (อ่านค่า P0 มาไว้ในตัวแปร In_Buf)
```

สำหรับแนวคิดในการเขียนโปรแกรมเพื่อสั่งงานให้รถหุ่นยนต์เคลื่อนที่ไปตามแนวเส้นนั้น จะอาศัยหลักการตรวจสอบสถานะทางโลจิกที่ได้จากวงจรตรวจจับเส้น (P0.5-P0.7) โดยถ้าพบว่าตัวตรวจจับตำแหน่งกลางมีค่าเป็น “1” แสดงว่าตำแหน่งเส้นอยู่ตรงกึ่งกลางก็จะสั่งให้รถเคลื่อนที่ไปข้างหน้าตามปรกติ แต่ถ้าพบว่าตัวตรวจจับด้านซ้ายมีค่าเป็น “1” ก็แสดงว่า รถเริ่มเอียงไปทางด้านขวามากแล้วซึ่งกำลังจะหลุดจากแนวเส้นในไม่ช้านี้ ซึ่งกรณีนี้ก็ต้องสั่งให้รถเลี้ยวกลับมาทางด้านซ้ายเพื่อให้รถกลับมาอยู่ในแนวเส้นตามเดิม โดยในกรณีนี้ควรใช้วิธีการสั่งเลี้ยวแบบรวดเร็วเนื่องจากรถใกล้จะหลุดจากแนวเส้นแล้วแล้วหรือถ้าพบว่าตัวตรวจจับด้านซ้ายและตัวตรวจจับตรงกลางเกิดมีค่าเป็น “1” พร้อมกัน ก็แสดงว่ารถเริ่มเอียงไปทางด้านขวานิดหน่อย ซึ่งก็ต้องสั่งให้รถเลี้ยวซ้ายกลับมาหาแนวเส้นเช่นกัน แต่ในกรณีนี้ควรใช้วิธีการสั่งเลี้ยวแบบปรกติแทน เนื่องจากรถออกจากศูนย์กลางของเส้นไปเพียงเล็กน้อยเท่านั้น ซึ่งถ้าสั่งเลี้ยวแบบรวดเร็วอาจทำให้รถเอียงเลยออกไปทางซ้ายแทน

ซึ่งในการวิเคราะห์แนวเส้นนั้น ถ้ามีการสั่งควบคุมการเคลื่อนที่เพื่อตอบสนองต่อสภาะการตรวจจับที่ได้อย่างเหมาะสมแล้วก็จะทำให้การเคลื่อนที่ของรถตามแนวเส้นเป็นไปอย่างราบรื่น แต่ถ้าการสั่งงานเพื่อตอบสนองต่อการตรวจจับไม่เหมาะสมก็จะทำให้การเคลื่อนที่ของตัวรถไม่เป็นธรรมชาติและอาจเกิดความผิดพลาดขึ้นได้ ตัวอย่างเช่นเมื่อพบว่าตัวตรวจจับด้านซ้ายและตัวตรวจจับตรงกลางเกิดมีค่าเป็น “1” พร้อมกัน ก็แสดงว่ารถเริ่มเอียงไปทางด้านขวานิดหน่อยเท่านั้น ซึ่งถ้าสั่งให้รถเลี้ยวซ้ายกลับเข้ามาในแนวเส้นด้วยวิธีการสั่งเลี้ยวแบบรวดเร็วนั้น ก็อาจทำให้รถเลี้ยวเลยศูนย์กลางของเส้นล้ำออกไปทางซ้ายแทน ซึ่งในการตรวจสอบสภานะของตัวตรวจจับเส้นครั้งต่อไปก็อาจพบว่าตัวตรวจจับด้านขวาและตัวตรวจจับตรงกลางเกิดมีค่าเป็น “1” พร้อมกันอีก และเมื่อสั่งให้รถเลี้ยวขวากลับมาหาแนวเส้นด้วยวิธีการเลี้ยวขวาอย่างเร็วอีกก็อาจทำให้รถเกิดการเลี้ยวขวาเกินศูนย์กลางของเส้นล้ำออกไปทางซ้ายอีกอย่างนี้เรื่อยๆ ซึ่งถึงแม้ว่าจะยังคงสามารถควบคุมการเคลื่อนที่ของตัวรถให้อยู่ในแนวเส้นต่อไปได้ แต่ลักษณะการเคลื่อนที่ของตัวรถจะมีลักษณะแบบส่ายไปมาตลอดเวลา หรือในกรณีที่ตรวจสอบแล้วพบว่ารถกำลังจะหลุดแนวเส้นออกไปทางขวา(ตัวตรวจจับซ้ายเป็น “1”) แล้วสั่งเลี้ยวกลับเข้ามายังแนวเส้นด้วยวิธีการเลี้ยวแบบปรกติ ก็อาจทำให้รถไม่สามารถเลี้ยวกลับเข้ามายังแนวศูนย์กลางของเส้นได้ทันซึ่งอาจเกิดการหลุดออกจากแนวเส้นได้ถ้าอยู่ในแนวเส้นที่เป็นทางโค้งมากๆเป็นต้น

```
'/*************************************/;
'/* Program Demo For ET-ROBOT RD2 V1.0 */;
'/* Controller  : P89C51RD2(Philips)   */;
'/* Run X - TAL : 18.432 Mhz           */;
'/*             : X2 Mode(6 Cycle Run) */;
'/* Complier    : BASCOM-51(V2.0.11.0) */;
'/*************************************/;

$regfile = "89c51rd.dat"        'กำหนดให้ใช้งานกับ CPU เบอร์ P89C51RD2(Philips)
$ramstart = 0
$ramsize = 256
$crystal = 36864000             'กำหนดค่า XTAL = 18.432MHz แบบ X2 Mode

Config Timer0=Timer,Gate=Internal,Mode=2  'ให้ Timer0 = 8Bit Auto Reload
Load Timer0 , 240               'กำหนดค่า Reload  = 240x325.52ns = 78.125uS
Disable Timer0                  'ห้ามไม่ให้เกิดการ Interrupt จากการทำงานของ Timer0
Start Timer0                    'เริ่มการทำงานของ Timer0 เพื่อ Trig PCA Counter
Cmod = &B00000100               'ให้ CPS(1:0) = 10 (PCA นับจาก Overflow ของ Timer0)
Ccapm1 = &B01000010             'Set Bit ECOM1,PWM1 (Enable PWM1)
Ccapm2 = &B01000010             'Set Bit ECOM2,PWM2 (Enable PWM2)
Ccap1h = 0                      'กำหนดค่า PWM1 Duty Cycle = 20mS
Ccap2h = 0                      'กำหนดค่า PWM2 Duty Cycle = 20mS
Ccon = &B01000000               'Set Bit CR (เริ่มการทำงานของระบบ PCA Timer)

Declare Sub Robot_forward       'โปรแกรมย่อยสำหรับบังคับให้รถเดินหน้า
Declare Sub Robot_backward      'โปรแกรมย่อยสำหรับบังคับให้รถถอยหลัง
Declare Sub Robot_fast_left     'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว
Declare Sub Robot_slow_left     'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ
Declare Sub Robot_fast_right    'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
Declare Sub Robot_slow_right    'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ
Declare Sub Robot_stop          'โปรแกรมย่อยสำหรับบังคับให้รถหยุด
Declare Sub Beep                'โปรแกรมย่อยสำหรับกำเนิดเสียง Beep

Dim Sensor_new As Byte          'ประกาศตัวแปรเก็บค่า Sensor ปัจจุบันเป็นแบบ Byte
Dim Sensor_old As Byte          'ประกาศตัวแปรเก็บค่า Sensor อ้างอิง(เก่า) แบบ Byte
Sensor_old = &H00               'กำหนดค่าเริ่มต้นให้กับตัวแปรเป็น 00H
```

```
Robot_stop                          'สั่งให้รถหยุดอยู่กับที่ก่อน เพื่อรอการกดสวิตช์ Start
Bitwait P0.2 , Reset                'รอการกดสวิตช์ Start เพื่อเริ่มต้นการทำงาน
Beep                                'ส่งเสียง Beep เพื่อแสดงการเริ่มต้นทำงาน
Do
  P0 = &HFF                         'เขียนค่า “1” ไปยังพอร์ต P0 เพื่อกำหนดเป็น Input
  Sensor_new = P0                   'อ่านค่าสถานะ P0 มาเก็บไว้ยังตัวแปร Sensor_new
  Sensor_new = Sensor_new And &B11100000   'เคลียร์บิตอื่นเป็น “0” ยกเว้นเส้น

  If Sensor_new <> Sensor_old Then 'ถ้า Sensor_new ไม่เท่ากับ Sensor_old
    Sensor_old = Sensor_new         'ให้ Update Sensor_old ด้วย Sensor_new


    Select Case Sensor_new                  'เริ่มวิเคราะห์สถานะเส้นที่ตรวจได้
      Case &B00000000 : Robot_forward       'ยังไม่พบเส้นให้เดินหน้าต่อไป
      Case &B00100000 : Robot_fast_left     'ซ้ายเป็น “1” สั่งเลี้ยวซ้ายอย่างเร็ว
      Case &B01000000 : Robot_forward       'กลางเป็น “1” สั่งเดินหน้า
      Case &B01100000 : Robot_slow_left     'ซ้ายและกลางเป็น 1 สั่งเลี้ยวซ้าย
      Case &B10000000 : Robot_fast_right    'ขวาเป็น “1” สั่งเลี้ยวขวาอย่างเร็ว
      Case &B10100000 : Robot_forward       'ซ้ายและขวาเป็น “1” สั่งเดินหน้า
      Case &B11000000 : Robot_slow_right    'ขวาและกลางเป็น “1” สั่งเลี้ยวขวา
      Case &B11100000 : Robot_fast_right    'ซ้าย,กลาง,ขวาเป็น 1 สั่งเลี้ยวขวา
    End Select                              'สิ้นสุดการวิเคราห์เส้นที่ตรวจจับ
  End If                                    'จบเงื่อนไข IF
Loop                                        'กลับไปเริ่มต้นตรวจเส้นอีก (DO)



'/*****************/;
'/* ROBOT Forward */;
'/*****************/;
'
Sub Robot_forward           ‘จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเคลื่อนที่ไปข้างหน้า
  Ccap1h = 256 - 26         ‘กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 256 - 13         ‘กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                     ‘จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเคลื่อนที่ไปข้างหน้า
```

```
'/******************/;
'/* ROBOT Backward */;
'/******************/;
'
Sub Robot_backward          'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์ถอยหลัง
  Ccap1h = 256 - 13         'กำหนดให้ SERVO1(ซ้าย) = 1mS = ถอยหลัง
  Ccap2h = 256 - 26         'กำหนดให้ SERVO2(ขวา) = 2mS = ถอยหลัง
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์ถอยหลัง


'/************************/;
'/* ROBOT Turn Left(Fast) */;
'/************************/;
'
Sub Robot_fast_left         'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว
  Ccap1h = 256 - 13         'กำหนดให้ SERVO1(ซ้าย) = 1mS = ถอยหลัง
  Ccap2h = 256 - 13         'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว


'/************************/;
'/* ROBOT Turn Left(Slow) */;
'/************************/;
'
Sub Robot_slow_left         'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ
  Ccap1h = 0                'กำหนดให้ SERVO1 (ซ้าย) = "1" = หยุดการเคลื่อนที่
  Ccap2h = 256 - 13         'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ


'/*************************/;
'/* ROBOT Turn Right(Fast) */;
'/*************************/;
'
Sub Robot_fast_right        'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
  Ccap1h = 256 - 26         'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 256 - 26         'กำหนดให้ SERVO2(ขวา) = 2mS = ถอยหลัง
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
```

```
'/*************************/;
'/* ROBOT Turn Right(Slow) */;
'/*************************/;
'
Sub Robot_slow_right      'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ
  Ccap1h = 256 - 26       'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 0              'กำหนดให้ SERVO2 (ขวา) = "1" = หยุดการเคลื่อนที่
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ


'/**************/;
'/* ROBOT Stop */;
'/**************/;
'
Sub Robot_stop            'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์หยุด
  Ccap1h = 0              'กำหนดให้ SERVO1 (ซ้าย) = "1" = หยุดการเคลื่อนที่
  Ccap2h = 0              'กำหนดให้ SERVO2 (ขวา) = "1" = หยุดการเคลื่อนที่
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์หยุด


'/**************/;
'/* Sound Beep */;
'/**************/;
'
Sub Beep                  'จุดเริ่มต้นโปรแกรมย่อยสร้างเสียง Beep
  Sound P1.7 , 100 , 50   'กำเนิดเสียง Beep
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสร้างเสียง Beep
```

สำหรับตัวอย่างโปรแกรมข้างต้นนั้น จะเป็นโปรแกรมสำหรับสั่งให้รถหุ่นยนต์เดินตามเส้น โดยโปรแกรมในตัวอย่างนี้จะสามารถใช้ได้กับลักษณะของเส้นที่เป็นเส้นตรงและเส้นโค้ง โดยใช้กับสนามที่มีเส้นเป็นสีดำ และที่สำคัญเส้นจะต้องต่อกันแบบครบวงจร คือไม่มีการขาดหายของเส้น เนื่องจากโปรแกรมจะไม่ได้ตรวจสอบเงื่อนไขของกรณีเส้นขาดไว้ด้วย

โดยหลักการทำงานของโปรแกรมจะเริ่มจากการสร้างตัวแปรสำหรับเก็บค่าสถานะของ Sensor ขึ้นมาใช้งานจำนวน 2 ตัวแปร ด้วยกัน คือ Sensor_new และ Sensor_old โดย Sensor_new จะใช้สำหรับเก็บค่าของสถานะ Input ที่อ่านได้ในปัจจุบันส่วน Sensor_old จะใช้สำหรับเก็บค่าสถานะของ Sensor เพื่อนำไปใช้เปรียบเทียบกับสถานะใหม่ที่อ่านเข้ามาได้ว่ามีความแตกต่างกันหรือไม่ โดยค่าของ Sensor ที่อ่านได้จากพอร์ต P0 นั้น จะถูกนำไปทำการ AND กับค่า 11100000B เสียก่อน เพื่อเคลียร์ค่าสถานะของ

Input อื่นๆ (ให้ค่าของบิต 0-4 เป็น “0”) ที่ไม่ใช้สถานะของการตรวจจับเส้นมีค่าเป็น “0” ทั้งหมด ซึ่งผลจากการ AND จะทำให้เหลือเพียงค่าสถานะของเส้นอยู่ในตำแหน่งข้อมูลบิตที่ 5-7 ของตัวแปร Sensor_new เท่านั้น

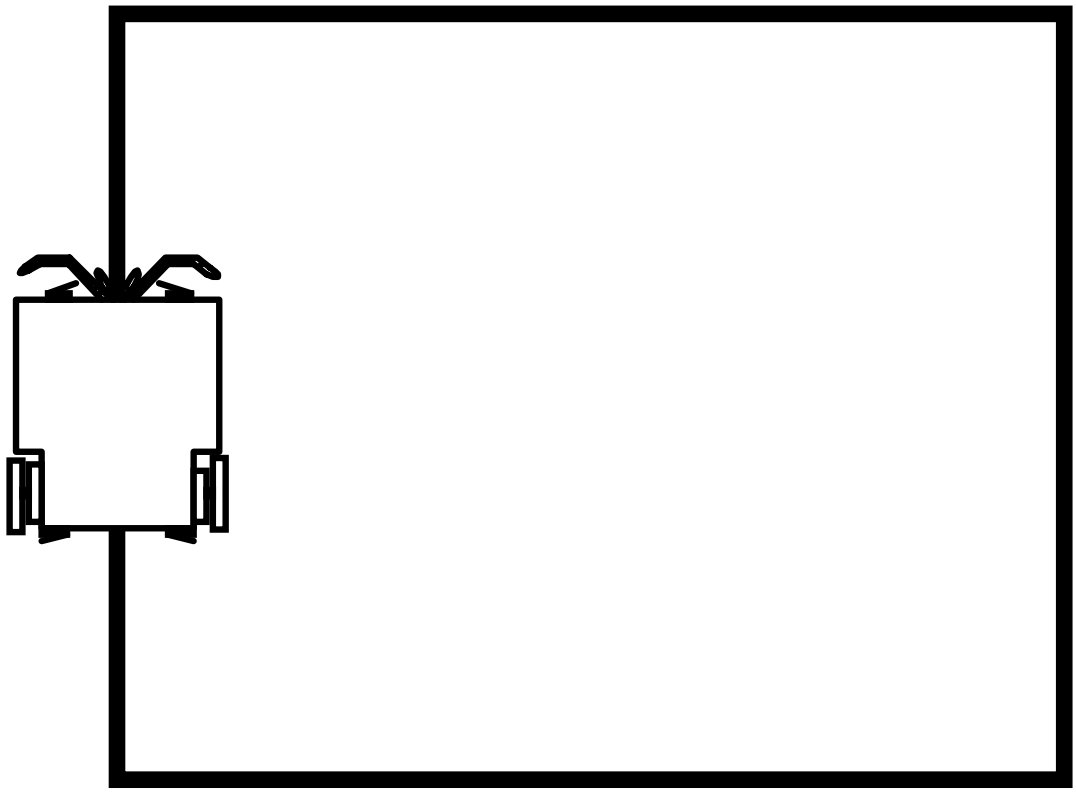
ซึ่งโปรแกรมจะใช้คำสั่งเงื่อนไข IF..THEN ในการตรวจสอบความแตกต่างของสถานะ Input ที่อ่านได้ในปัจจุบัน (Sensor_new) กับค่าสถานะที่อ่านได้ครั้งสุดท้าย (Sensor_old) โดยถ้าตรวจพบว่าค่าของสถานะ Sensor ใหม่และเก่ายังมีค่าเท่ากันอยู่ก็แสดงว่าสถานะของ Sensor ของการตรวจจับเส้นยังไม่มีการเปลี่ยนแปลงใดๆ ก็จะไม่เข้าทำงานในเงื่อนไขของ IF..THEN แต่ถ้าตรวจพบว่าสถานะของ Sensor ใหม่และเก่ามีความแตกต่างกัน ก็จะทำการ Update ค่าของตัวแปร Sensor_old ด้วยค่าใหม่ที่อ่านได้เพื่อเก็บไว้ใช้เป็นค่าเปรียบเทียบในครั้งต่อไป ซึ่งในกรณีนี้ โปรแกรมจะเข้าทำงานในเงื่อนไข IF..THEN โดยจะมีการตรวจสอบสถานะของ Sensor และทำการวิเคราห์ว่าเป็นอย่างไร โดยใช้คำสั่ง Select..Case โดยในโปรแกรมจะเห็นได้ว่ามีการแยกเงื่อนไขการตรวจสอบสถานะของ Sensor_new ที่ใช้ในการตรวจจับเส้นทั้ง 8 เงื่อนไข โดยสถานะของ Sensor ตรวจจับเส้นจะอยู่ที่บิตข้อมูลตำแหน่งที่ 5-7 ของตัวแปร Sensor_new ซึ่งการวิเคราะห์เงื่อนไขของ Sensor พอสรุปได้ดังนี้

- ถ้า Sensor ทั้ง3 จุดเป็น “0” ทั้งหมด แสดงว่าตำแหน่งของตัวรถยังไม่อยู่ในแนวเส้น หรือ อีกกรณีหนึ่งก็คือ ตำแหน่งของเส้นอยู่ในแนวตำแหน่งกึ่งกลางระหว่างช่องว่างของตัว Sensor 2 ชุด ซึ่งอาจเป็นช่องว่างระหว่าง Sensor ตัวซ้าย กับ Sensor ตัวกลาง หรือ Sensor ตัวขวา และ Sensor ตัวกลาง ก็เป็นไปได้เช่นกัน ซึ่งในกรณีนี้ ก็จะต้องสั่งให้รถเคลื่อนที่ต่อไปข้างหน้า
- ถ้า Sensor ตำแหน่งซ้ายสุดซึ่งก็คือ บิตที่5 มีค่าเป็น “1” เพียงบิตเพียงบิตเดียว ส่วนบิตที่ 6 และ 7 เป็น “0” อยู่ ก็แสดงว่า ตำแหน่งของเส้นอยู่ตรงกับตำแหน่ง Sensor ด้านซ้าย ซึ่งลักษณะเช่นนี้ จะหมายถึงว่า รถมีการเคลื่อนที่ออกจากจุดศูนย์กลางของเส้นเอียงออกไปทางขวาจึงทำให้ตำแหน่งของ Sensor ตัวซ้ายเคลื่อนที่ไปอยู่เหนือเส้นพอดี ซึ่งถ้าปล่อยไว้ต่อไปก็จะทำให้รถเคลื่อนที่ออกนอกแนวเส้นไปอย่างแน่นอน ดังนั้น จึงต้องสั่งให้รถเลี้ยวซ้ายกลับมายังแนวเส้นใหม่ โดยการสั่งเลี้ยวจะต้องสั่งให้เลี้ยวแบบรวดเร็วด้วย ไม่เช่นนั้นอาจทำให้รถหลุดจากแนวเส้นออกไปได้
- ถ้า Sensor ตำแหน่งซ้ายและกลาง ซึ่งก็คือ บิตที่5 และ 6 มีค่าเป็น “1” พร้อมกันทั้งคู่ ส่วนบิตที่ 7 เป็น “0” อยู่ ก็แสดงว่า ตำแหน่งของเส้นอยู่ตรงกับตำแหน่ง Sensor ด้านซ้ายและกลาง ซึ่งลักษณะเช่นนี้ จะหมายถึงว่า รถมีการเคลื่อนที่ออกจากจุดศูนย์กลางของเส้นเอียงออกไปทางขวาเพียงเล็กน้อยเท่านั้น ซึ่งเราก็จะทำการสั่งให้รถเลี้ยวซ้ายกลับมาหาจุดศูนย์กลางของเส้นใหม่ โดยการสั่งเลี้ยวจะต้องสั่งให้เลี้ยวแบบปรกติหรือเลี้ยวแบบช้า เพราะถ้าสั่งเลี้ยวแบบเร็วอาจทำให้รถเลี้ยวเกินจุดศูนย์กลางของเส้นเอียงออกไปทางซ้ายได้

- ถ้า Sensor ตำแหน่งขวาสุดซึ่งก็คือ บิตที่7 มีค่าเป็น “1” เพียงบิตเพียงบิตเดียว ส่วนบิตที่ 5 และ 6 เป็น “0” อยู่ ก็แสดงว่า ตำแหน่งของเส้นอยู่ตรงกับตำแหน่ง Sensor ด้านขวา ซึ่งลักษณะเช่นนี้ จะหมายถึงว่า รถมีการเคลื่อนที่ออกจากจุดศูนย์กลางของเส้นเอียงออกไปทางซ้ายจึงทำให้ตำแหน่งของ Sensor ตัวขวาเคลื่อนที่ไปอยู่เหนือเส้นพอดี ซึ่งถ้าปล่อยไว้ต่อไปก็จะทำให้รถเคลื่อนที่ออกนอกแนวเส้นไปอย่างแน่นอน ดังนั้น จึงต้องสั่งให้รถเลี้ยวขวากลับมายังแนวเส้นใหม่ โดยการสั่งเลี้ยวจะต้องสั่งให้เลี้ยวแบบรวดเร็วด้วย ไม่เช่นนั้นอาจทำให้รถหลุดจากแนวเส้นออกไปได้
- ถ้า Sensor ตำแหน่งขวาและกลาง ซึ่งก็คือ บิตที่ 6 และ 7 มีค่าเป็น “1” พร้อมกันทั้งคู่ ส่วนบิตที่ 5 เป็น “0” อยู่ ก็แสดงว่า ตำแหน่งของเส้นอยู่ตรงกับตำแหน่ง Sensor ด้านขวาและกลาง ซึ่งลักษณะเช่นนี้ จะหมายถึงว่า รถมีการเคลื่อนที่ออกจากจุดศูนย์กลางของเส้นเอียงออกไปทางซ้ายเพียงเล็กน้อยเท่านั้น ซึ่งเราก็จะทำการสั่งให้รถเลี้ยวขวากลับมาหาจุดศูนย์กลางของเส้นใหม่ โดยการสั่งเลี้ยวจะต้องสั่งให้เลี้ยวแบบปรกติหรือเลี้ยวแบบช้า เพราะถ้าสั่งเลี้ยวแบบเร็วอาจทำให้รถเลี้ยวเกินจุดศูนย์กลางของเส้นเอียงออกไปทางขวาได้
- ถ้า Sensor ตำแหน่งซ้ายและขวา ซึ่งก็คือ บิตที่5 และ 7 มีค่าเป็น “1” พร้อมกันทั้งคู่ ส่วนบิตที่ 6 เป็น “0” อยู่ ก็แสดงว่า ตำแหน่งของเส้นอยู่ตรงกับตำแหน่ง Sensor ด้านซ้ายและขวา ส่วนตรงกลางไม่มีเส้น ซึ่งลักษณะเช่นนี้ไม่น่าจะเป็นไปได้ในกรณีของสนามที่เป็นเส้นตรงและเส้นโค้ง แต่น่าจะเป็นสนามที่เป็นทางแยกมากกว่า ดังนั้นเราจึงสั่งให้รถเดินหน้าต่อไปตามปรกติ
- ถ้า Sensor ทั้ง3 จุดเป็น “1” ทั้งหมด แสดงว่าตำแหน่งของตัวรถอยู่บนอยู่บนแนวเส้นที่ตั้งฉากกับทิศทางการเคลื่อนที่ของตัวรถอยู่ ซึ่งเราก็จะต้องสั่งให้รถเลี้ยวขวาหรือซ้ายเพื่อให้ทิศทางของรถเคลื่อนที่ไปตามแนวเส้น

การเขียนโปรแกรมควบคุมให้รถเคลื่อนที่ไปตามเส้นตรงและเส้นมุมฉาก

สำหรับตัวอย่างนี้จะเป็นตัวอย่างของสถานะการณ์ที่มีความซับซ้อนมากขึ้น เนื่องจากสภาวะของเส้นที่มีลักษณะเป็นมุมฉากนั้น สถานะของเส้นที่ตรวจพบได้อาจมีการเปลี่ยนแปลงอย่างรวดเร็ว ดังนั้นการที่เราจะสามารถทำการตรวจสอบแนวเส้นจุดที่เป็นมุมหักเหต่างๆได้นั้น เราจำเป็นต้องคิดค้นวิธีการในการตรวจสอบที่สลับซับซ้อนมากขึ้นกว่าเดิม เพราะถ้าใช้การตรวจสอบแนวเส้นแบบตัวอย่างที่ผ่านมานั้น อาจทำให้การเคลื่อนที่ของตัวรถหลุดออกจากแนวเส้นได้ง่าย เนื่องจากการบังคับเลี้ยวของ SERVO MOTOR อาจไม่สามารถตอบสนองต่อการเปลี่ยนแปลงของ Input ที่ตรวจจับได้อย่างทันทีทันใด ซึ่งมักพบว่า เมื่อรถเคลื่อนที่มาถึงตำแหน่งที่เป็นมุมหักเหต่างๆ ไม่ว่าจะเป็นมุมฉาก หรือ มุมสามเหลี่ยม ตัวรถมักจะเคลื่อนที่ผ่านตำแหน่งของเส้นออกไป หรือเรียกว่า การหลุดจากแนวเส้นออกไป เนื่องจากไม่สามารถเลี้ยวตามแนวเส้นได้ทัน



ลักษณะของ สนามทดสอบที่ใช้

สำหรับการเขียนโปรแกรมเพื่อใช้ตรวจสอบเส้น สำหรับควบคุมให้รถหุ่นยนต์เคลื่อนที่ไปตามแนวเส้นนั้น จุดที่ยากที่สุดของการตรวจสอบและควบคุมการเคลื่อนที่ของตัวรถนั้นจะอยู่ตรงจุดที่เป็นมุมหักเหต่างๆ เช่น เส้นมุมฉาก หรือเส้นทะแยงฟันปลา และ จุดมุมของสามเหลี่ยมเป็นต้น

```
'/*************************************/;
'/* Program Demo For ET-ROBOT RD2 V1.0 */;
'/* Controller  : P89C51RD2(Philips)   */;
'/* Run X - TAL : 18.432 Mhz           */;
'/*             : X2 Mode(6 Cycle Run) */;
'/* Complier    : BASCOM-51(V2.0.11.0) */;
'/*************************************/;

$regfile = "89c51rd.dat"        'กำหนดให้ใช้งานกับ CPU เบอร์ P89C51RD2(Philips)
$ramstart = 0
$ramsize = 256
$crystal = 36864000             'กำหนดค่า XTAL = 18.432MHz แบบ X2 Mode

Config Timer0=Timer,Gate=Internal,Mode=2  'ให้ Timer0 = 8Bit Auto Reload
Load Timer0 , 240               'กำหนดค่า Reload  = 240x325.52ns = 78.125uS
Disable Timer0                  'ห้ามไม่ให้เกิดการ Interrupt จากการทำงานของ Timer0
Start Timer0                    'เริ่มการทำงานของ Timer0 เพื่อ Trig PCA Counter
Cmod = &B00000100               'ให้ CPS(1:0) = 10 (PCA นับจาก Overflow ของ Timer0)
Ccapm1 = &B01000010             'Set Bit ECOM1,PWM1 (Enable PWM1)
Ccapm2 = &B01000010             'Set Bit ECOM2,PWM2 (Enable PWM2)
Ccap1h = 0                      'กำหนดค่า PWM1 Duty Cycle = 20mS
Ccap2h = 0                      'กำหนดค่า PWM2 Duty Cycle = 20mS
Ccon = &B01000000               'Set Bit CR (เริ่มการทำงานของระบบ PCA Timer)

Declare Sub Robot_forward       'โปรแกรมย่อยสำหรับบังคับให้รถเดินหน้า
Declare Sub Robot_backward      'โปรแกรมย่อยสำหรับบังคับให้รถถอยหลัง
Declare Sub Robot_fast_left     'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว
Declare Sub Robot_slow_left     'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ
Declare Sub Robot_fast_right    'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
Declare Sub Robot_slow_right    'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ
Declare Sub Robot_stop          'ประกาศโปรแกรมย่อยสำหรับบังคับให้รถหยุด
Declare Sub Robot_exit_line     'โปรแกรมย่อยสำหรับตรวจสอบการหลุดเส้น
```

```
Declare Sub Robot_over_line     'โปรแกรมย่อยสำหรับตรวจสอบการทับเส้น
Declare Sub Beep                'โปรแกรมย่อยสำหรับกำเนิดเสียง Beep

Dim Sensor_new As Byte          'ประกาศตัวแปรเก็บค่า Sensor ปัจจุบันเป็นแบบ Byte
Dim Sensor_old As Byte          'ประกาศตัวแปรเก็บค่า Sensor อ้างอิง(เก่า) แบบ Byte

Sensor_old = &H00               'กำหนดค่าเริ่มต้นให้กับตัวแปรเป็น 00H
Robot_stop                      'สั่งให้รถหยุดอยู่กับที่ก่อน เพื่อรอการกดสวิตช์ Start
Bitwait P0.2 , Reset            'รอการกดสวิตช์ Start เพื่อเริ่มต้นการทำงาน
Beep                            'ส่งเสียง Beep เพื่อแสดงการเริ่มต้นทำงาน

Do
  P0 = &HFF                     'เขียนค่า “1” ไปยังพอร์ต P0 เพื่อกำหนดเป็น Input
  Sensor_new = P0               'อ่านค่าสถานะ P0 มาเก็บไว้ยังตัวแปร Sensor_new
  Sensor_new = Sensor_new And &B11100000   'เคลียร์บิตอื่นเป็น “0” ยกเว้นเส้น

  If Sensor_new <> Sensor_old Then 'ถ้า Sensor_new ไม่เท่ากับ Sensor_old
     Sensor_old = Sensor_new       'ให้ Update Sensor_old ด้วย Sensor_new

     Select Case Sensor_new                'เริ่มต้นการวิเคราะห์เส้นที่ตรวจจับ
      Case &B00000000 : Robot_exit_line    'หลุดเส้น ให้ตรวจสอบการหลุดเส้น
      Case &B00100000 : Robot_fast_left    'ซ้าย สั่งเลี้ยวซ้ายเข้าหาเส้น
      Case &B01000000 : Robot_forward      'กลาง สั่งเดินหน้าต่อไป
      Case &B01100000 : Robot_fast_left    'ซ้าย,กลาง สั่งเลี้ยวซ้ายเข้าหาเส้น
      Case &B10000000 : Robot_fast_right   'ขวา สั่งเลี้ยวขวาเข้าหาเส้น
      Case &B10100000 : Robot_forward      'ซ้าย,ขวา สั่งเดินหน้าต่อไป
      Case &B11000000 : Robot_fast_right   'ขวา,กลาง สั่งเลี้ยวขวาเข้าหาเส้น
      Case &B11100000 : Robot_over_line    'อยู่บนเส้น ตรวจสอบการทับเส้น
    End Select                             'สิ้นสุดการวิเคราห์เส้นที่ตรวจจับได้

  End If                                   'จบเงื่อนไข IF
Loop                                       'กลับไปตรวจสอบเส้นอีก (DO)
```

```
'/******************/;
'/* ROBOT Forward */;
'/******************/;
'
Sub Robot_forward          'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเคลื่อนที่ไปข้างหน้า
  Ccap1h = 256 - 26        'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 256 - 13        'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                    'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเคลื่อนที่ไปข้างหน้า


'/******************/;
'/* ROBOT Backward */;
'/******************/;
'
Sub Robot_backward         'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์ถอยหลัง
  Ccap1h = 256 - 13        'กำหนดให้ SERVO1(ซ้าย) = 1mS = ถอยหลัง
  Ccap2h = 256 - 26        'กำหนดให้ SERVO2(ขวา) = 2mS = ถอยหลัง
End Sub                    'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์ถอยหลัง


'/*************************/;
'/* ROBOT Turn Left(Fast) */;
'/*************************/;
'
Sub Robot_fast_left        'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว
  Ccap1h = 256 - 13        'กำหนดให้ SERVO1(ซ้าย) = 1mS = ถอยหลัง
  Ccap2h = 256 - 13        'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                    'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว


'/*************************/;
'/* ROBOT Turn Left(Slow) */;
'/*************************/;
'
Sub Robot_slow_left        'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ
  Ccap1h = 0               'กำหนดให้ SERVO1 (ซ้าย) = "1" = หยุดการเคลื่อนที่
  Ccap2h = 256 - 13        'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                    'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ
```

```
'/*************************/;
'/* ROBOT Turn Right(Fast) */;
'/*************************/;
'
Sub Robot_fast_right      'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
  Ccap1h = 256 - 26       'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 256 - 26       'กำหนดให้ SERVO2(ขวา) = 2mS = ถอยหลัง
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว


'/*************************/;
'/* ROBOT Turn Right(Slow) */;
'/*************************/;
'
Sub Robot_slow_right      'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ
  Ccap1h = 256 - 26       'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 0              'กำหนดให้ SERVO2 (ขวา) = "1" = หยุดการเคลื่อนที่
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ


'/**************/;
'/* ROBOT Stop */;
'/**************/;
'
Sub Robot_stop            'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์หยุด
  Ccap1h = 0              'กำหนดให้ SERVO1 (ซ้าย) = "1" = หยุดการเคลื่อนที่
  Ccap2h = 0              'กำหนดให้ SERVO2 (ขวา) = "1" = หยุดการเคลื่อนที่
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์หยุด


'/*********************/;
'/* ROBOT Out of Line */;
'/*********************/;
'
Sub Robot_exit_line                   'จุดเริ่มต้นโปรแกรมตรวจสอบการหลุดเส้น
  Select Case Sensor_last             'เริ่มต้นตรวจสอบสถานะเก่าก่อนหลุดเส้น
    Case &B00000000 : Robot_forward       'ไม่พบเส้นอยู่ก่อนแล้วสั่งเดินหน้า
    Case &B00100000 : Robot_fast_left     'Sensor ซ้าย (**อาจเป็นมุมซ้ายr**)
    Case &B01000000 : Robot_fast_right    'Sensor กลาง (**จุดสิ้นสุดของเส้น**)
```

```
    Case &B01100000 : Robot_fast_left    'ซ้าย+กลาง(**อาจเป็นมุมซ้าย**)
    Case &B10000000 : Robot_fast_right   'Sensor ขวา (**อาจเป็นมุมขวา**)
    Case &B10100000 : Robot_backward     'ขวา+ซ้าย (**อาจเป็นแยกตัว Y**)
    Case &B11000000 : Robot_fast_right   'ขวา+กลาง(**อาจเป็นมุมขวา**)
    Case &B11100000 : Robot_backward     'ทั้งหมด (**อาจวิ่งเลยจากทางแยก**)
  End Select                             'จุดสิ้นสุดการหาสาเหตุการหลุดเส้น
End Sub                                  'จุดสิ้นสุดโปรแกรมตรวจการหลุดเส้น


'/*******************/;
'/* ROBOT Over Line */;
'/*******************/;
'
Sub Robot_over_line                      'จุดเริ่มต้นโปรแกรมตรวจการทับเส้น
  Select Case Sensor_last                'เริ่มต้นการตรวจสถานะก่อนทับเส้น
    Case &B00000000 : Robot_fast_right   'ไม่พบ Sensor (**พบเส้นครั้งแรก**)
    Case &B00100000 : Robot_fast_right   'ซ้าย(**อาจเป็นมุมขวา**)
    Case &B01000000 : Beep               'กลาง(**อาจเป็นทางแยกตัว T**)
    Case &B01100000 : Robot_fast_left    'ซ้ายและกลาง(**อาจเป็นมุมซ้าย**)
    Case &B10000000 : Robot_fast_left    'ขวา (**อาจเป็นมุมซ้าย**)
    Case &B10100000 : Robot_forward      'ขวาและซ้าย(**อาจเป็นแยกตัว Y**)
    Case &B11000000 : Robot_fast_right   'ขวาและกลาง (**อาจเป็นมุมขวา**)
    Case &B11100000 : Robot_forward      'ทั้งหมด (**ทับเส้นอยู่ก่อนแล้ว**)
  End Select                             'จุดสิ้นสุดการตรวจสาเหตุการทับเส้น
End Sub                                  'จุดสิ้นสุดโปรแกรมตรวจการทับเส้น


'/**************/;
'/* Sound Beep */;
'/**************/;
'
Sub Beep                    'จุดเริ่มต้นโปรแกรมย่อยสร้างเสียง Beep
  Sound P1.7 , 100 , 50     'กำเนิดเสียง Beep
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสร้างเสียง Beep
```

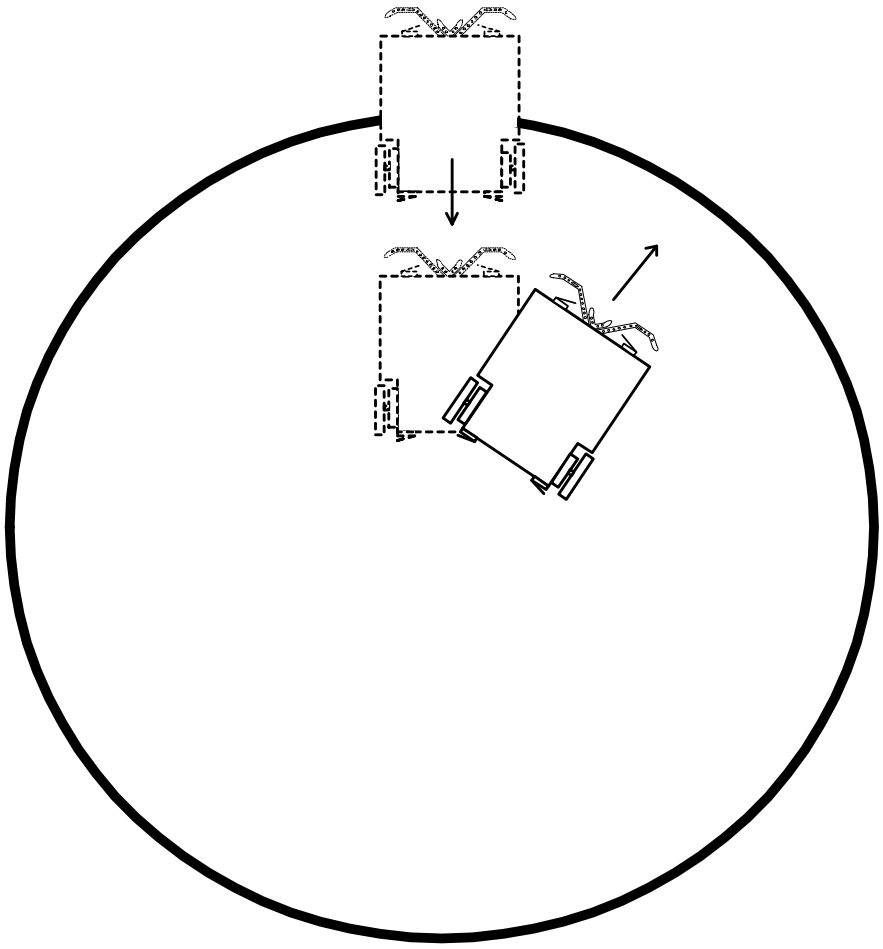
สำหรับหลักการทำงานของตัวอย่างโปรแกรมนี้ จะคล้ายกับตัวอย่างอื่นๆที่ผ่านมา เพียงแต่มีการปรับปรุง เงื่อนไขของการตรวจสอบและวิเคราะห์แนวเส้นเพิ่มเติมขึ้นมาอีก 2 จุด คือ

- มีการตรวจสอบเงื่อนไขเพิ่มเติม เมื่อพบว่ารถหลุดจากแนวเส้นไปแล้ว โดยในกรณีจะกลับไปนำเอาค่าสถานะการทำงานของ Sensor ตรวจจับเส้น ที่บันทึกไว้ได้ก่อนที่จะพบว่าหลุดเส้นเข้ามาตรวจสอบ เพื่อหาความเป็นไปได้ของลักษณะเส้น เพื่อจะได้สั่งให้รถเลี้ยวกลับเข้าไปหาแนวเส้นใหม่อีกครั้งหนึ่ง
- มีการตรวจสอบเงื่อนไขเพิ่มเติม เมื่อพบว่ารถอยู่ในแนวตั้งฉากกับแนวเส้นอยู่ (Sensor ทั้ง 3 ตัวทับเส้นอยู่) ซึ่งในกรณีนี้จะใช้เพื่อวิเคราะห์สภาวะของเส้นในขณะนั้นๆว่าน่าจะมีลักษณะเป็นทางแยกแบบใด

โดยความสามารถของตัวอย่างโปรแกรมนี้ จะสามารถใช้ควบคุมให้รถเคลื่อนที่ไปตามแนวเส้นได้หลายรูปแบบ ไม่ว่าจะเป็นเส้นตรง เส้นโค้ง ที่มีความคดเคี้ยวไปมา เส้นหักมุมแบบมุมฉาก หรือเส้นที่เป็นแนวทะแยงแบบสลับฟันปลา เป็นต้น

การเขียนโปรแกรมควบคุมให้รถหุ่นยนต์เคลื่อนที่อยู่ภายในกรอบของเส้น

สำหรับตัวอย่างนี้ จะเป็นการประยุกต์การใช้งานของวงจรตรวจจับเส้น ซึ่งแทนที่จะใช้สำหรับตรวจจับเส้น เพื่อควบคุมให้รถเคลื่อนที่ไปตามแนวเส้นเหมือนตัวอย่างอื่นๆที่ผ่านมานั้น ในตัวอย่างนี้เราจะมาทดลองเปลี่ยนเงื่อนไขการตรวจจับเส้นกันดู กล่าวคือ เราจะเปลี่ยนวิธีการจากเดิมที่ให้รถเคลื่อนที่ไปตามแนวเส้นที่กำหนดไว้ แต่ในตัวอย่างนี้จะให้รถเคลื่อนที่อยู่ภายในกรอบของเส้น หรือใช้แนวเส้นเป็นรั้วหรือเขตแดนสำหรับควบคุมการเคลื่อนที่ของรถแทน โดยลักษณะของสนามทดสอบนั้น อาจทำเป็นเส้นวงกลมหรือเส้นกรอบสี่เหลี่ยม แล้วนำตัวรถไปปล่อยไว้ภายในกรอบ จากนั้นก็จะปล่อยให้รถเคลื่อนที่ไปเรื่อยๆ โดยเมื่อตรวจพบแนวเส้นก็จะต้องสั่งให้รถถอยกลับมาจากแนวเส้น แล้วเลี้ยวไปในทิศทางอื่นๆแทน ซึ่งลักษณะการทำงานของตัวอย่างนี้ สามารถนำไปดัดแปลง หรือประยุกต์เพื่อใช้ในการแข่งขันรถหุ่นยนต์แบบ SUMO ได้ เพียงแต่ในตัวอย่างนี้เราจะยังไม่มีการเพิ่มการตรวจจับการชน หรือสิ่งกีดขวางอื่นๆเข้าไปด้วย โดยจะแสดงให้เห็นเฉพาะการตรวจจับเส้น เพื่อใช้ควบคุมไม่ให้รถเคลื่อนที่ผ่านแนวเส้นออกไป โดยจะปล่อยให้รถเคลื่อนที่อยู่ภายในวงกรอบของเส้นที่กำหนดไว้เท่านั้น



ภาพ แสดงลักษณะการเคลื่อนที่ของตัวรถในสนามทดสอบ

```
'/*************************************/;
'/* Program Demo For ET-ROBOT RD2 V1.0 */;
'/* Controller  : P89C51RD2(Philips)   */;
'/* Run X - TAL : 18.432 Mhz           */;
'/*             : X2 Mode(6 Cycle Run) */;
'/* Complier    : BASCOM-51(V2.0.11.0) */;
'/*************************************/;

$regfile = "89c51rd.dat"        'กำหนดให้ใช้งานกับ CPU เบอร์ P89C51RD2(Philips)
$ramstart = 0
$ramsize = 256
$crystal = 36864000             'กำหนดค่า XTAL = 18.432MHz แบบ X2 Mode

Config Timer0=Timer,Gate=Internal,Mode=2 ' ให้ Timer0 = 8Bit Auto Reload
Load Timer0 , 240               'กำหนดค่า Reload  = 240x325.52ns = 78.125uS
Disable Timer0                  'ห้ามไม่ให้เกิดการ Interrupt จากการทำงานของ Timer0
Start Timer0                    'เริ่มการทำงานของ Timer0 เพื่อ Trig PCA Counter
Cmod = &B00000100               'ให้ CPS(1:0) = 10 (PCA นับจาก Overflow ของ Timer0)
Ccapm1 = &B01000010             'Set Bit ECOM1,PWM1 (Enable PWM1)
Ccapm2 = &B01000010             'Set Bit ECOM2,PWM2 (Enable PWM2)
Ccap1h = 0                      'กำหนดค่า PWM1 Duty Cycle = 20mS
Ccap2h = 0                      'กำหนดค่า PWM2 Duty Cycle = 20mS
Ccon = &B01000000               'Set Bit CR (เริ่มการทำงานของระบบ PCA Timer)

Declare Sub Robot_forward       'โปรแกรมย่อยสำหรับบังคับให้รถเดินหน้า
Declare Sub Robot_backward      'โปรแกรมย่อยสำหรับบังคับให้รถถอยหลัง
Declare Sub Robot_fast_left     'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว
Declare Sub Robot_slow_left     'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ
Declare Sub Robot_fast_right    'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
Declare Sub Robot_slow_right    'โปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ
Declare Sub Robot_stop          'โปรแกรมย่อยสำหรับบังคับให้รถหยุด
Declare Sub Beep                'โปรแกรมย่อยสำหรับกำเนิดเสียง Beep

Dim Sensor_new As Byte          'ประกาศตัวแปรเก็บค่า Sensor ปัจจุบันเป็นแบบ Byte
Dim Sensor_old As Byte          'ประกาศตัวแปรเก็บค่า Sensor อ้างอิง(เก่า) แบบ Byte
```

```
Sensor_old = &H00                    'กำหนดค่าเริ่มต้นให้กับตัวแปรเป็น 00H
Robot_stop                           'สั่งให้รถหยุดอยู่กับที่ก่อน เพื่อรอการกดสวิตช์ Start
Bitwait P0.2 , Reset                 'รอการกดสวิตช์ Start เพื่อเริ่มต้นการทำงาน
Beep                                 'ส่งเสียง Beep เพื่อแสดงการเริ่มต้นทำงาน

Do
  P0 = &HFF                          'เขียนค่า "1" ไปยังพอร์ต P0 เพื่อกำหนดเป็น Input
  Sensor_new = P0                    'อ่านค่าสถานะพอร์ต P0 มาเก็บไว้ยัง Sensor_new
  Sensor_new = Sensor_new And &B11100000   'เคลียร์บิตอื่นเป็น "0" ยกเว้นเส้น

  If Sensor_new <> Sensor_old Then 'ถ้า Sensor_new ไม่เท่ากับ Sensor_old
     Sensor_old = Sensor_new         'ให้ Update Sensor_old ด้วย Sensor_new
     Select Case Sensor_new                  'เริ่มต้นวิเคราะห์เส้นที่ตรวจจับได้
      Case &B00000000 : Robot_forward        'กรณียังไม่พบเส้นให้เดินหน้าต่อไป
      Case Else                              'ถ้าพบเส้นไม่ว่ากรณีใดให้ทำดังนี้
         Robot_backward                      'ถอยกลับจากแนวเส้น
         Wait 1                              'หน่วงเวลาจนกว่าจะพ้นจากเส้น
         Robot_fast_right                    'สั่งเลี้ยวขวาหลบจากแนวเส้นที่พบ
         Wait 1                              'รอให้ระยะการเลี้ยวหลบพ้นแนวเส้น
         Robot_forward                       'เดินหน้าต่อไปตามปรกติ
      End Select                             'สิ้นสุดการวิเคราห์เส้นที่ตรวจจับได้
  End If                                     'จบเงื่อนไข IF
Loop                                         'กลับไปเริ่มต้นตรวจเส้นอีก (DO)


'/*****************/;
'/* ROBOT Forward */;
'/*****************/;
'
Sub Robot_forward          'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเคลื่อนที่ไปข้างหน้า
  Ccap1h = 256 - 26        'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 256 - 13        'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                    'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเคลื่อนที่ไปข้างหน้า
```

```
'/******************/;
'/* ROBOT Backward */;
'/******************/;
'
Sub Robot_backward          'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์ถอยหลัง
  Ccap1h = 256 - 13         'กำหนดให้ SERVO1(ซ้าย) = 1mS = ถอยหลัง
  Ccap2h = 256 - 26         'กำหนดให้ SERVO2(ขวา) = 2mS = ถอยหลัง
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์ถอยหลัง


'/************************/;
'/* ROBOT Turn Left(Fast) */;
'/************************/;
'
Sub Robot_fast_left         'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว
  Ccap1h = 256 - 13         'กำหนดให้ SERVO1(ซ้าย) = 1mS = ถอยหลัง
  Ccap2h = 256 - 13         'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวซ้ายอย่างเร็ว


'/************************/;
'/* ROBOT Turn Left(Slow) */;
'/************************/;
'
Sub Robot_slow_left         'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์เลี้ยวซ้าย
แบบปรกติ
  Ccap1h = 0                'กำหนดให้ SERVO1 (ซ้าย) = "1" = หยุดการเคลื่อนที่
  Ccap2h = 256 - 13         'กำหนดให้ SERVO2(ขวา) = 1mS = เคลื่อนที่ไปข้างหน้า
End Sub                     'จุดสิ้นสุดโปรแกรมสำหรับบังคับให้รถเลี้ยวซ้ายแบบปรกติ


'/*************************/;
'/* ROBOT Turn Right(Fast) */;
'/*************************/;
'
Sub Robot_fast_right        'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
  Ccap1h = 256 - 26         'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 256 - 26         'กำหนดให้ SERVO2(ขวา) = 2mS = ถอยหลัง
End Sub                     'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาอย่างเร็ว
```

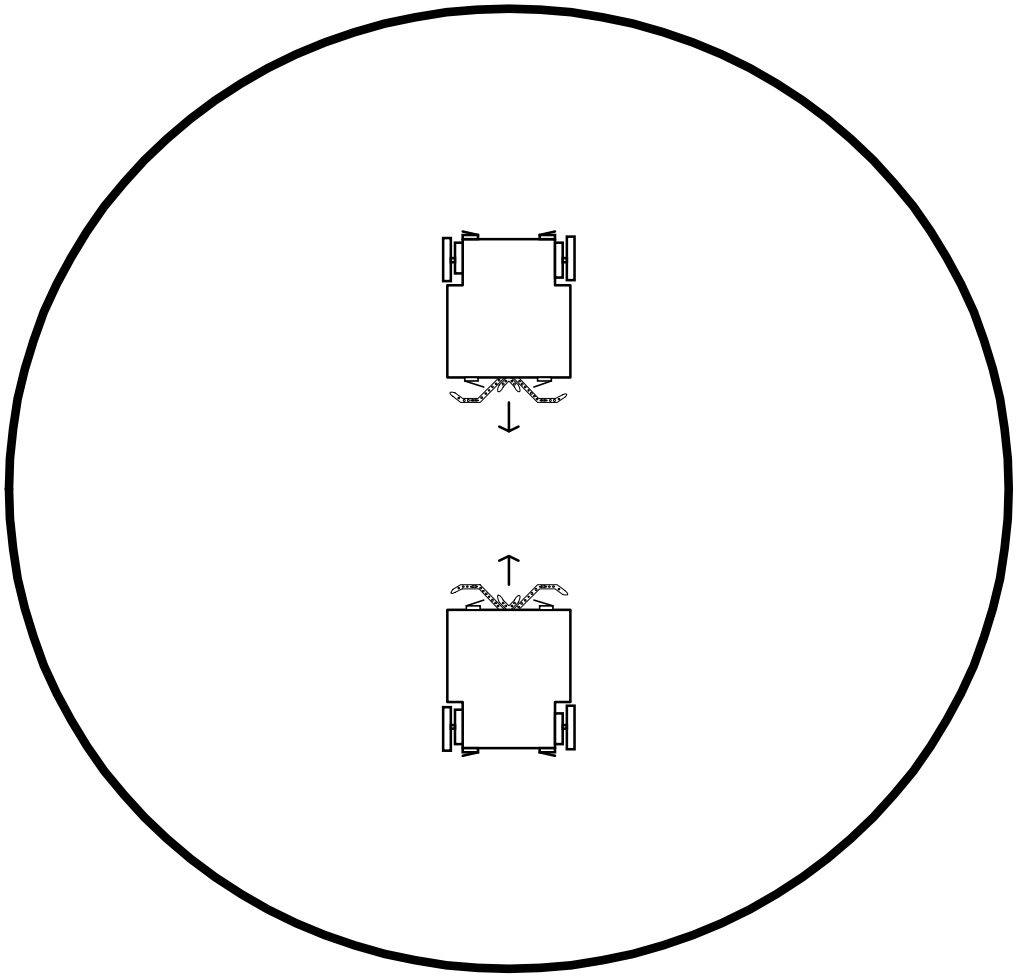
```
'/*************************/;
'/* ROBOT Turn Right(Slow) */;
'/*************************/;
'
Sub Robot_slow_right      'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ
  Ccap1h = 256 - 26       'กำหนดให้ SERVO1(ซ้าย) = 2mS = เคลื่อนที่ไปข้างหน้า
  Ccap2h = 0              'กำหนดให้ SERVO2 (ขวา) = "1" = หยุดการเคลื่อนที่
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถเลี้ยวขวาแบบปรกติ


'/**************/;
'/* ROBOT Stop */;
'/**************/;
'
Sub Robot_stop            'จุดเริ่มต้นโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์หยุด
  Ccap1h = 0              'กำหนดให้ SERVO1 (ซ้าย) = "1" = หยุดการเคลื่อนที่
  Ccap2h = 0              'กำหนดให้ SERVO2 (ขวา) = "1" = หยุดการเคลื่อนที่
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสำหรับบังคับให้รถหุ่นยนต์หยุด


'/**************/;
'/* Sound Beep */;
'/**************/;
'
Sub Beep                  'จุดเริ่มต้นโปรแกรมย่อยสร้างเสียง  Beep
  Sound P1.7 , 100 , 50   'กำเนิดเสียง Beep
End Sub                   'จุดสิ้นสุดโปรแกรมย่อยสร้างเสียง Beep
```

ซึ่งจะเห็นได้ว่า ลักษณะโครงสร้างการทำงานของโปรแกรมจะเหมือนกับโปรแกรมอื่นๆที่ผ่านมา โดยมีจุดการทำงานที่แตกต่างกันเพียงจุดเดียว คือ เงื่อนไขในการวิเคราะห์แนวเส้น โดยในตัวอย่างนี้จะใช้เงื่อนไขการตรวจสอบแนวเส้น เพียง 2 เงื่อนไขเท่ากนั้น กล่าวคือ ถ้าตรวจสอบพบว่ารถยังไม่อยู่ในแนวเส้นก็สั่งให้รถเดินหน้าต่อไปเรื่อยๆ แต่ถ้ามีการตรวจพบแนวเส้น ไม่ว่าจะเป็นกรณีใดๆ (Sensor ตัวใดตัวหนึ่งมีค่าเป็น “1”) ก็จะสั่งให้รถถอยหลังกลับออกมาจากแนวเส้นพร้อมกับหน่วงเวลาสำหรับถอยหลังไว้ระยะหนึ่งเพื่อให้ระยะการถอยของรถพ้นจากแนวเส้นออกมา จากนั้นก็จะสั่งเลี้ยวไปทางขวาพร้อมกับหน่วงเวลาไว้อีกชั่วขณะหนึ่งเพื่อให้ระยะการเลี้ยวของรถหลบพ้นจากแนวเส้นที่ตรวจพบ จากนั้นก็จะสั่งให้รถเดินหน้าไปตามปรกติ ซึ่งเงื่อนไขการตรวจสอบเส้นจะเป็นอย่างนี้ไปตลอด

โดยเมื่อนำรถไปทดสอบการทำงานในสนามก็จะเห็นว่า รถจะมีการเคลื่อนที่ไปได้อย่างอิสระถ้าไม่มีแนวเส้นมาขวางอยู่ แต่เมื่อรถเคลื่อนที่ไปพบแนวเส้นเมื่อใด ก็จะมีการถอยหลังกลับออกมาแล้วเลี้ยวหลบไปทางขวาเสมอ ซึ่งจากแนวคิดนี้ เราสามารถนำตัวอย่างโปรแกรมนี้ไปดัดแปลงเพื่อใช้ในการแข่งขันหุ่นยนต์แบบ SUMO ได้ โดยอาจทำการติดตั้งตัวตรวจจับวัตถุเพิ่มเติมเข้าไป จากนั้นก็เพิ่มการตรวจสอบเงื่อนไขในโปรแกรมให้ทำการค้นหาสิ่งกีดขวางเพื่อจะได้วิ่งชนวัตถุนั้นให้พ้นออกจากแนวเส้นไป โดยต้องควบคุมไม่ให้ตัวรถวิ่งผ่านแนวเส้นออกไปดังนี้เป็นต้น



ตัวอย่าง การประยุกต์โปรแกรมเป็นการแข่งขันหุ่นยนต์แบบ SUMO