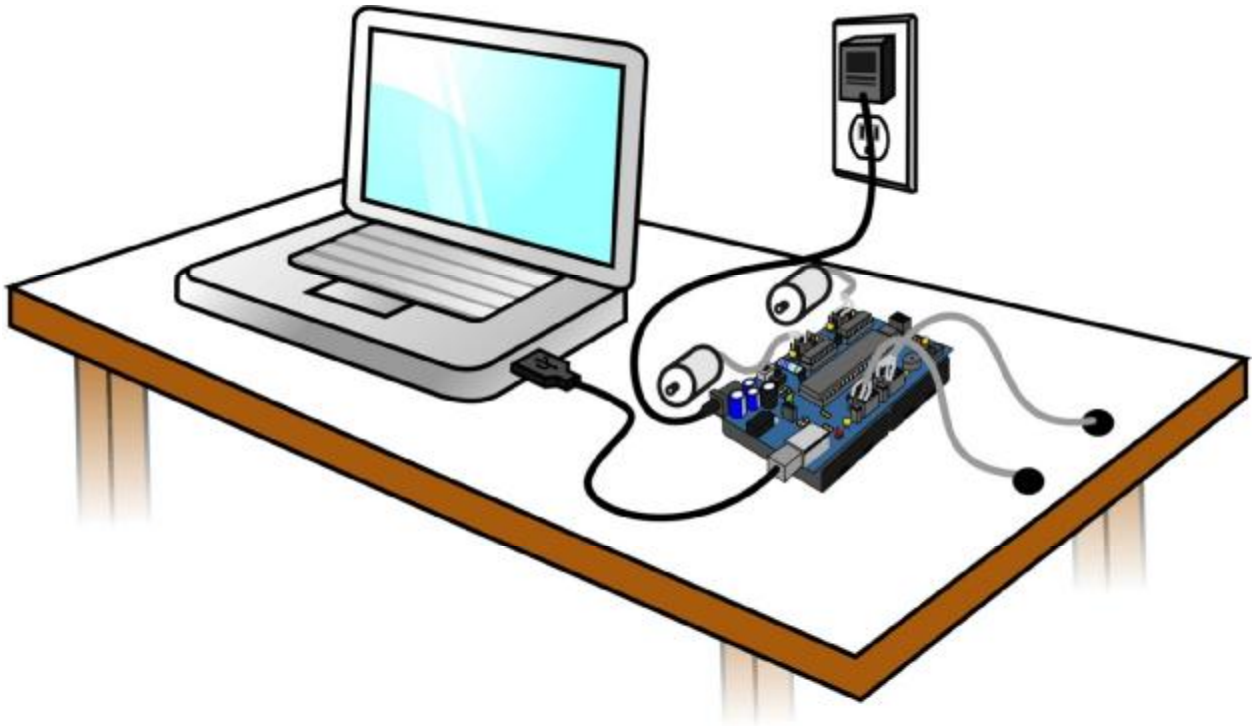
www.gogoboard.org

Science & Robotics for the Inventive Mind



แนะนำการใช้ GoGo Board เบื้องต้น

อาจารย์ ดร. อานันท์ สีห์พิทักษ์เกียรติ

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเชียงใหม่

ชุดหุ่นยนต์ GoGo Board ประกอบด้วย

1. GoGo Board 1 ตัว
2. เซ็นเซอร์ 4 ชนิด อย่างละ 1 ตัว ประกอบด้วย
   2.1. เซ็นเซอร์อุณหภูมิ
   2.2. เซ็นเซอร์แสง
   2.3. เซ็นเซอร์สวิทช์
   2.4. เซ็นเซอร์อินฟราเรด ใช้ตรวจสอบวัตถุโดยไม่ต้องสัมผัส (ใช้ในร่มเท่านั้น)
3. มอเตอร์ 2 ตัว
4. ล้อ 2 ตัว
5. LED 2 ตัว (สีแดงและสีน้ำเงิน)
6. สาย USB 1 เส้น
7. รีโมทควบคุม 1 ตัว
8. คู่มือ

เซ็นเซอร์แสง

เซ็นเซอร์อุณหภูมิ

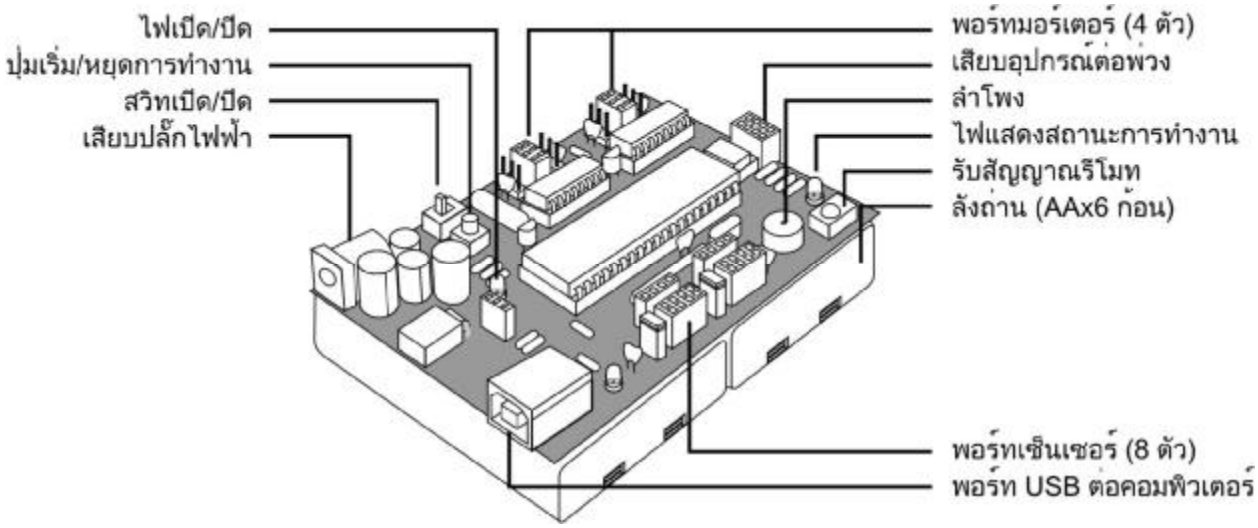
สวิท

เซ็นเซอร์อินฟราเรด

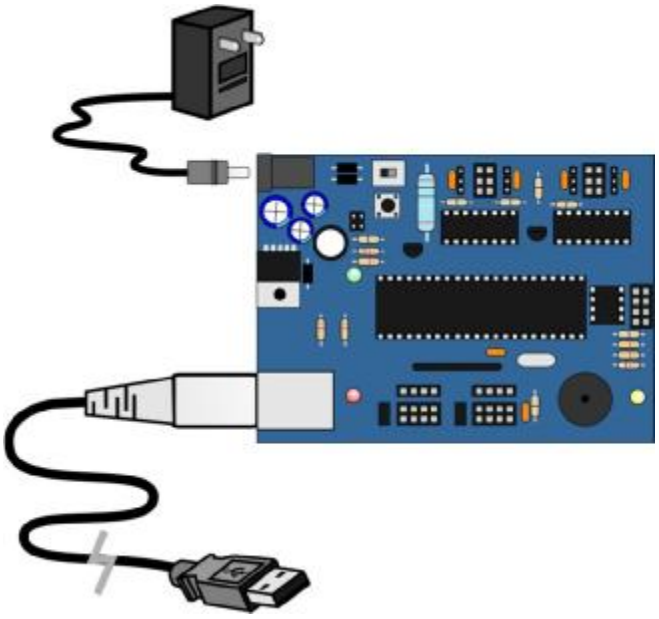
ไฟ LED

มอร์เตอร์

ส่วนประกอบของ GoGo Board



อุปกรณ์ที่จำเป็น



แหล่งจ่ายไฟ

GoGo Board ใช้ไฟได้จากสามแหล่งคือ

- ไฟจากสาย USB
- ถ่าน AA จำนวน 6 ก้อน ซึ่งจะบรรจุอยู่ในลังถ่านด้านใต้ หรือ
- จะเสียบไฟจากหม้อแปลงไฟกระแสตรงที่มีแรงดันระหว่าง 9–12 โวลท์

หมายเหตุ: ไฟจาก USB จะไม่เพียงพอต่อการควบคุมมอเตอร์ ควรใช้แหล่งไฟอื่นเสริมด้วยเสมอ

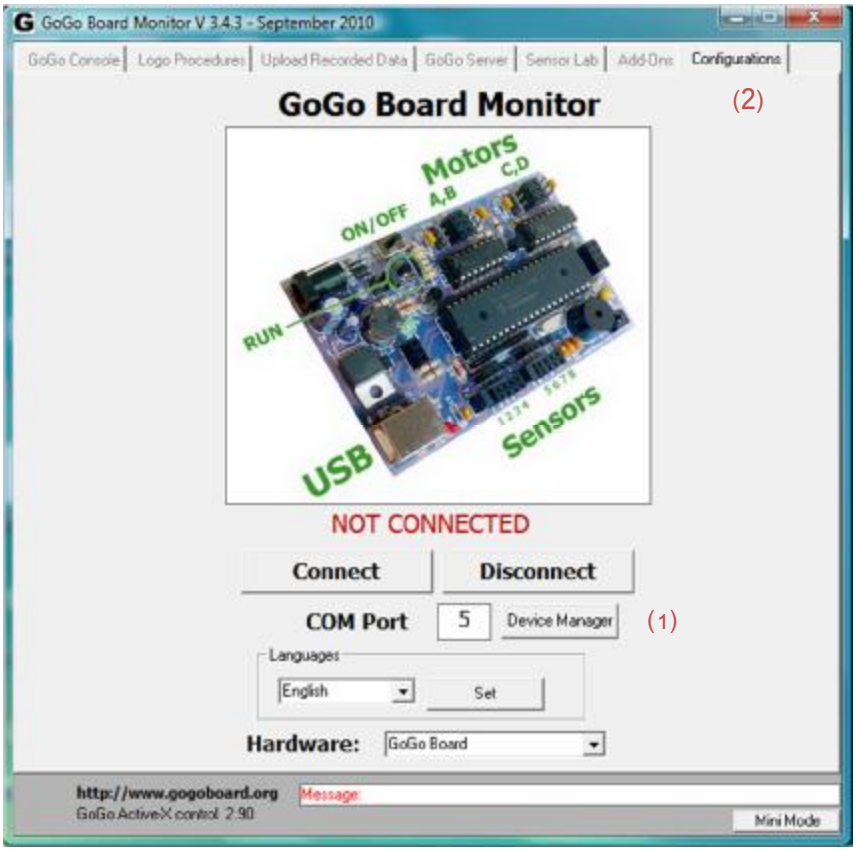
การต่อพ่วงกับคอมพิวเตอร์

GoGo Board ติดต่อกับคอมพิวเตอร์ผ่านทางพอร์ท USB

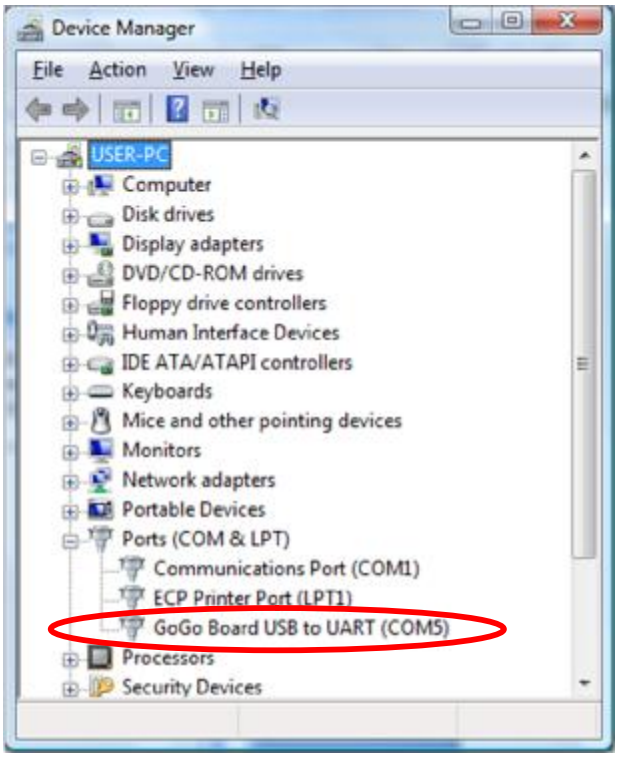
เริ่มต้นใช้งาน GoGo Board

โปรแกรม GoGo Monitor

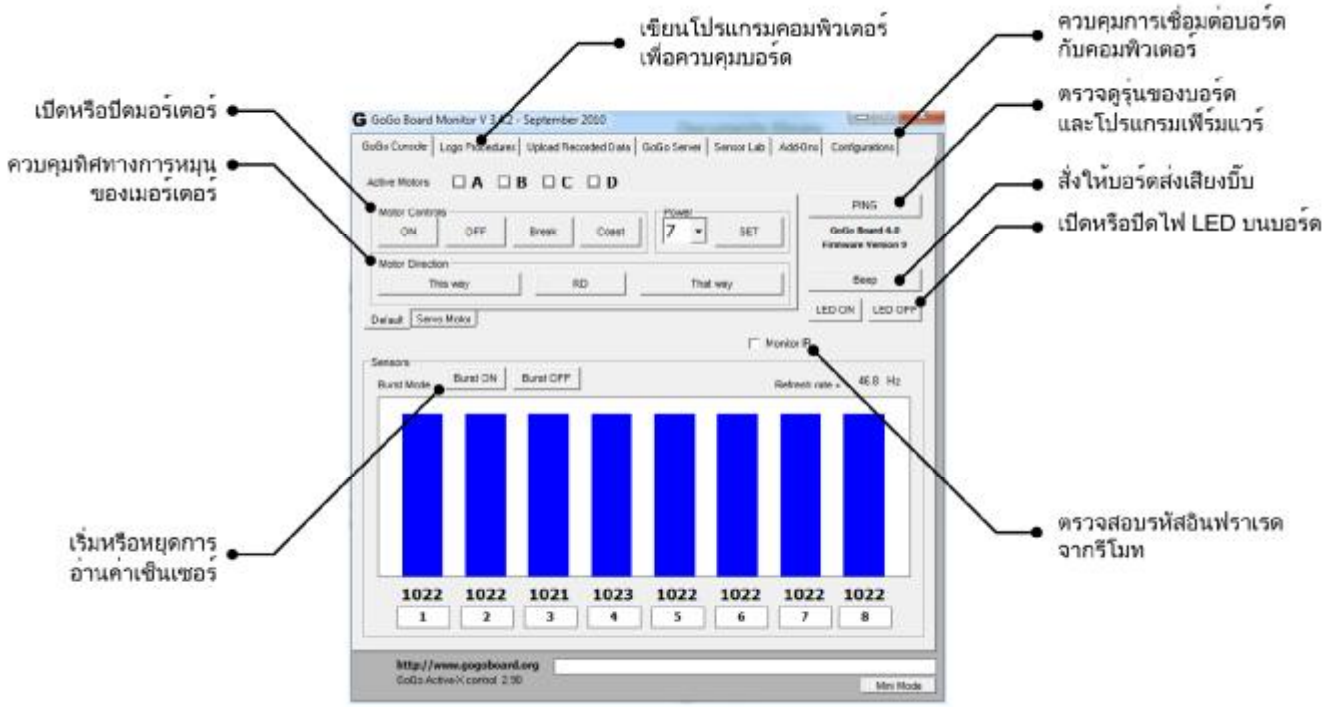
เมื่อเชื่อมต่อ GoGo Board เข้ากับคอมพิวเตอร์ผ่านทางสาย USB แล้ว และเปิดบอร์ด เราสามารถควบคุมการทำงานของ GoGo Board ได้ผ่านทางโปรแกรม GoGo Monitor โดยโปรแกรมนี้สามารถดาวน์โหลดฟรีได้ที่ http://gogoboard.stanford.edu/th/downloads เมื่อติดตั้งแล้วและเรียกใช้งานโปรแกรมจะพบหน้าต่างดังรูปต่อไปนี้



การที่จะเริ่มใช้งาน GoGo Board ได้นั้นเราจะต้องทราบหมายเลขพอร์ทสื่อสาร (COM port) ที่ใช้ก่อนแล้วจึงกดปุ่ม “เชื่อมต่อ” หมายเลขนี้จะมีค่าต่างกันไป หากไม่ทราบว่าจะใช้ค่าใดให้เข้าไปดูหมายเลขพอร์ทสื่อสารของเครื่องที่มีได้ทาง Device Manager ของ Windows โดยคลิกที่ปุ่ม Device Manager (1) ใน tab Configurations (2) จะปรากฏหน้าต่าง Device Manager ให้ดูหมายเลข Com Port จากสัญลักษณ์ 



เมื่อทดลองจนเชื่อมต่อกับ GoGo Board ได้แล้วก็จะเข้าถึงหน้าต่างควบคุมหลักของโปรแกรมดังภาพ



การควบคุมทั่ วไป

- ส่งเสียง – สั่ งให้GoGo Board ส่งเสียงร้องสั้ นๆ
- เปิด LED, ปิด LED – สั่ งเปิด และ ปิด ไฟแสดงสถานการณ์ทำงานบน GoGo Board
- PING – จะตรวจสอบรุ่นของ GoGo Board ที่ใช้งานอยู่

ส่วนควบคุมมอเตอร์

- GoGo Board มีพอร์ทเสียบมอเตอร์ทั้ งหมด4 ช่อง โดยแต่ละช่องมีชื่อว่า A, B, C, D ตามลำดับ หากต้องการควบคุมการทำงานของมอเตอร์ใดก็ให้เลือกมอเตอร์นั้นโดยการทำเครื่องหมายถูกในช่องด้านข้างชื่อของมอเตอร์นั้ นๆ โดยสามารถเลือกมอเตอร์ได้หลายตัวพร้อมกัน เมื่อเลือกมอเตอร์ที่ต้องการแล้วก็สามารถสั่ งงานต่างๆ ได้ดังนี้
  - เปิด, ปิด – สั่ งเปิดและปิดพอร์ทนั้ นๆ
  - เบรก –คือการใส่แรงต้านการหมุนเพื่อให้มอเตอร์อยู่นิ่ง
  - ปล่อย – คือการตัดไฟเฉยๆ มอเตอร์สามารถหมุนฟรีได้
  - กำลัง – คือการตั้ งค่าว่ามอเตอร์จะหมุดด้วยกำลังสูงต่ำเพียงใด โดย 7 คือกำลังสูงสุด
  - ทางนี้, ทางนั้ น– คือการกำหนดทิศทางการหมุนของมอเตอร์
- กลับทิศ– คือการกลับทิศการหมุนของมอเตอร์

เซ็นเซอร์

- GoGo Board มีพอร์ทเซ็นเซอร์ทั้ งหมด8 ช่อง เรียกชื่อว่า เซ็นเซอร์1 ถึง เซ็นเซอร์8 โดย GoGo Board จะส่งค่าของเซ็นเซอร์เหล่านี้มายังโปรแกรม GoGo Monitor อย่างต่อเนื่องทันทีหลังจากกดปุ่ ม"เริ่ม"
- ปกติแล้วหากไม่มีเซ็นเซอร์ต่อเชื่อมอยู่ค่าที่อ่านได้จะมีค่าเท่ากับ1023 ซึ่งเป็นค่าสูงสุด ค่าที่ได้นี้จะเปลี่ยนไปเมื่อมีเซ็นเซอร์เสียบอยู่

แผงควบคุมหลักของโปรแกรม GoGo Monitor แบ่งออกเป็น 3 ส่วนใหญ่ด้วยกันคือ

1. **ส่วนการควบคุมทั่วไป** – ส่วนนี้ใช้ทดสอบการเชื่อมต่อกับ GoGo Board เช่น
   ส่งเสียง – สั่งให้ GoGo Board ส่งเสียงร้องสั้นๆ
   เปิด LED, ปิด LED – สั่งเปิด และ ปิด ไฟแสดงสถานการณ์ทำงานบน GoGo Board
   PING – จะตรวจสอบรุ่นของ GoGo Board ที่ใช้งานอยู่

2. **ส่วนควบคุมมอเตอร์** – GoGo Board มีพอร์ทเสียบมอเตอร์ทั้งหมด 4 ช่อง โดยแต่ละช่องมีชื่อว่า A, B, C, D ตามลำดับ หากต้องการควบคุมการทำงานของมอเตอร์ใดก็ให้เลือกมอเตอร์นั้นโดยการทำเครื่องหมายถูกในช่องด้านข้างชื่อของมอเตอร์นั้นๆ โดยสามารถเลือกมอเตอร์ได้หลายตัวพร้อมกัน เมื่อเลือกมอเตอร์ที่ต้องการแล้วก็สามารถสั่งงานต่างๆ ได้ดังนี้
   เปิด, ปิด – สั่งเปิดและปิดพอร์ทนั้นๆ
   เบรก –คือการใส่แรงต้านการหมุนเพื่อให้มอเตอร์อยู่นิ่ง
   ปล่อย – คือการตัดไฟเฉยๆ มอเตอร์สามารถหมุนฟรีได้
   กำลัง – คือการตั้งค่าว่ามอเตอร์จะหมุดด้วยกำลังสูงต่ฎเพียงใด โดย 7 คือกำลังสูงสุด
   ทางนี้, ทางนั้น – คือการกำหนดทิศทางการหมุนของมอเตอร์
   กลับทิศ – คือการกลับทิศการหมุนของมอเตอร์

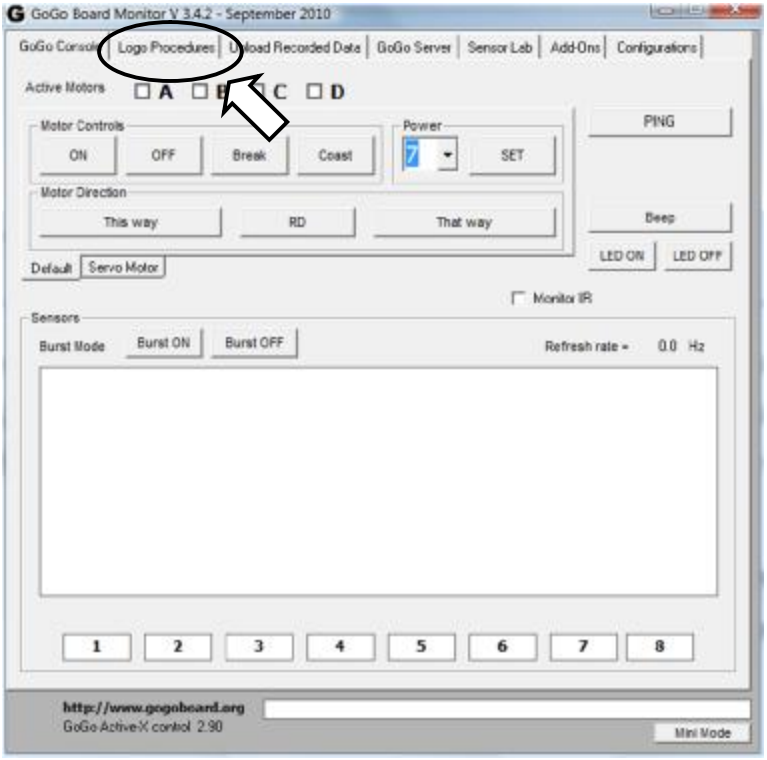
3. **เซ็นเซอร์ –** GoGo Board มีพอร์ทเซ็นเซอร์ทั้งหมด 8 ช่อง เรียกชื่อว่า เซ็นเซอร์ 1 ถึง เซ็นเซอร์ 8 โดย GoGo Board จะส่งค่าของเซ็นเซอร์เหล่านี้มายังโปรแกรม GoGo Monitor อย่างต่อเนื่องทันทีหลังจากกดปุ่ม “เริ่ม”

   ปกติแล้วหากไม่มีเซ็นเซอร์ต่อเชื่อมอยู่ค่าที่อ่านได้จะมีค่าเท่ากับ 1023 ซึ่งเป็นค่าสูงสุด ค่าที่ได้นี้จะเปลี่ยนไปเมื่อมีเซ็นเซอร์เสียบอยู่

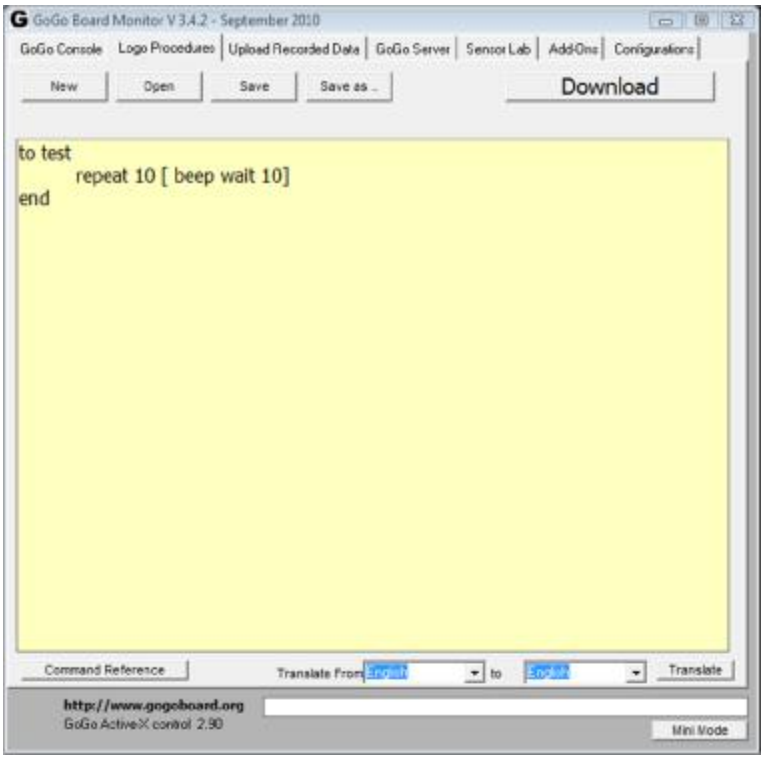
การเขียนโปรแกรมควบคุม GoGo Board

นอกจากการควบคุม GoGo Board โดยตรงแล้ว เรายังสามารถเขียนโปรแกรมเพื่อโหลดลงไปเก็บไว้บน GoGo Board ได้ด้วย วิธีนี้มีข้อดีคือ GoGo Board สามารถทำงานตามโปรแกรมได้โดยไม่ต้องเชื่อมต่ออยู่กับเครื่องคอมพิวเตอร์ การเขียนโปรแกรม GoGo Board จะมีขั้นตอนตามตัวอย่างต่อไปนี้

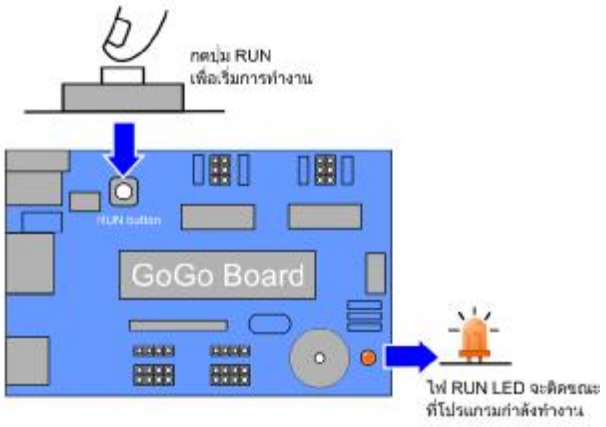
1. จุดประสงค์ของโปรแกรม: สมมุติว่าเราต้องการสั่งให้ GoGo Board ส่งเสียงบี๊บทุกๆ 1 วินาที ทั้งหมด 10 ครั้ง
2. ให้เชื่อมต่อกับ GoGo Board ผ่านทางโปรแกรม GoGo Monitor ตามที่ได้อธิบายไปก่อนหน้านี้
3. ในหน้าหลักของโปรแกรม GoGo Monitor ให้คลิกเลือกแท็บที่ชื่อ Logo Procedures ดังรูป



4. ในหน้า Logo Procedures จะมีช่องว่างให้เขียนโปรแกรมลงไป ซึ่งจากจุดประสงค์ของโปรแกรมในข้อ 1 โปรแกรมจะมีลักษณะดังในภาพต่อไปนี้



5. เมื่อเขียนโปรแกรมเสร็จแล้วให้กดปุ่ม “Download” เพื่อส่งโปรแกรมไปเก็บไว้ใน GoGo Board ถ้าโปรแกรมไม่มีข้อผิดพลาด และการ Download เสร็จสิ้น จะได้ยินเสียงบี๊บจาก GoGo Board 1 ครั้ง
6. การสั่งให้ GoGo Board ทำงานตามโปรแกรมที่ Download ไปจะทำโดยการกดปุ่ม Run บน GoGo Board ซึ่งในกรณีนี้ควรจะได้ยินเสียง GoGo Board บี๊บ 10 ครั้ง โดยในขณะที่โปรแกรมกำลังทำงานจะเห็นว่าไฟ USER LED สีส้มติดอยู่ หากต้องการหยุดการทำงานของโปรแกรมกลางคันให้กดปุ่ม RUN อีกครั้ง



รายการคำสั่งควบคุม GoGo Board โดยโปรแกรม GoGo Monitor

มอเตอร์

GoGo Board มีมอเตอร์สี่ตัวชื่อว่า A, B, C, และ D การใช้งานมอเตอร์นั้นจะเริ่มต้นด้วยการเลือกมอเตอร์ (ใช้คำสั่ง a, b, ab, ฯลฯ) แล้วบอกมันว่าต้องการให้ทำอะไร (เช่น, on (เปิด), off (ปิด), rd (กลับทิศ) ฯลฯ)

`a,`	เลือกสั่งงานมอเตอร์ A
`b,`	เลือกสั่งงานมอเตอร์ B
`c,`	เลือกสั่งงานมอเตอร์ C
`d,`	เลือกสั่งงานมอเตอร์ D

สามารถผสมอักษรเพื่อเลือกมอเตอร์หลายตัว เช่น

`ab,`	เลือกสั่งงานมอเตอร์ AB พร้อมกัน
`acd,`	เลือกสั่งงานมอเตอร์ A C และ D พร้อมกัน
`on`	เปิดมอเตอร์ที่เลือกไว้
`off`	ปิดมอเตอร์ที่เลือกไว้
`onfor` *ระยะเวลา*	เปิดมอเตอร์ไว้เป็นเวลาหนึ่ง, "ระยะเวลา" เป็นตัวกำหนดว่ามอเตอร์จะถูกเปิดไว้เป็นเวลานานเท่าใด หน่วยของเวลาคือ หนึ่งในสิบของวินาที ยกตัวอย่างเช่น onfor 10 จะเปิดมอเตอร์ไว้เป็นเวลาหนึ่งวินาที onfor 100 จะเปิดมอเตอร์ไว้ 10 วินาที
`thisway`	กำหนดทิศทางการหมุนของมอเตอร์ให้เป็น "ทางนี้" (ซึ่งจะเป็นทางไหนนั้นขึ้นอยู่กับการต่อมอเตอร์) ไฟสถานะเมื่อมอเตอร์หมุน "ทางนี้" จะเป็นสีเขียว
`thatway`	กำหนดทิศทางการหมุนของมอเตอร์ให้เป็น "ทางนั้น" (ซึ่งจะเป็นทางตรงกันข้ามกับ "ทางนี้") ไฟสถานะเมื่อมอเตอร์หมุน "ทางนั้น" จะเป็นสีแดง
`rd`	กลับทิศการหมุน ไม่ว่ามอเตอร์จะหมุนทางใด คำสั่งนี้จะกลับทิศการหมุนนั้นให้เป็นทางตรงข้าม
`setpower` *ระดับ*	ตั้งค่ากำลังของมอเตอร์: ค่า "ระดับ" มีช่วงอยู่ระหว่าง 0 (ไม่กำลังเลย มอเตอร์จะหยุด) ไปจนถึง 7 (เต็มกำลัง มอเตอร์จะหมุนเร็วและแรงที่สุด) ค่าระดับกำลังปกติของมอเตอร์คือ 7

เวลา

ด้วยคำสั่ง wait เราสามารถบอกให้ GoGo Board อยู่เฉยๆ ให้เวลาผ่านไประยะหนึ่ง ในขณะที่ GoGo Board อยู่เฉยๆ นี้ มอเตอร์จะถูกเปิดทิ้งไว้ก็ได้ เช่น

```
ab, on wait 20 off
```

จะเปิดมอเตอร์ A และ B ไว้เป็นเวลา 2 วินาที แล้วจึงปิดมอเตอร์ คำสั่งนี้ให้ผลเหมือนกันกับ:

```
ab, onfor 20
```

นอกจากนี้ในตัว GoGo Board ยังมีนาฬิกาที่เดินอยู่ตลอดเวลาแม้ในขณะที่ GoGo Board กำลังทำงานอย่างอื่นอยู่ มีคำสั่งสองคำสั่งที่ใช้งานนาฬิกานี้คือ resett ซึ่งจะทำให้นาฬิกาเริ่มนับจากศูนย์ใหม่ และ timer ซึ่งจะแจ้งค่านาฬิกาปัจจุบัน (เป็นค่าตัวเลขค่าหนึ่ง)

ตารางต่อไปนี้สรุปการใช้งานคำสั่ง wait, timer, และ resett

`wait` *ระยะเวลา*	หยุดอยู่เฉยๆ เป็นเวลาเท่ากับค่า "ระยะเวลา" ที่กำหนด หน่วยของเวลาคือ หนึ่งในสิบวินาที ตัวอย่างเช่น wait 10 จะทำให้ GoGo Board อยู่เฉยๆ เป็นเวลาหนึ่งวินาที
`timer`	แจ้งค่านาฬิกา หน่วยของเวลาจะนับทุกๆ 0.1 วินาที ดังนั้นถ้า timer = 20, แสดงว่านาฬิกาเริ่มนับมาเป็นเวลาสองวินาที
`resett`	บอกให้นาฬิกาเริ่มนับจากศูนย์ใหม่

เซ็นเซอร์

GoGo Board มีเซ็นเซอร์อยู่ 8 พอร์ทด้วยกัน ชื่อว่า sensor1–8 มีอุปกรณ์หลายชนิดที่สามารถใช้งานกับเซ็นเซอร์นี้ได้ เช่น

- อุปกรณ์ที่มีสถานะเปิด/ปิด เช่นปุ่ม หรือ สวิตช์ชนิดต่างๆ
- เซ็นเซอร์ที่เปลี่ยนแปลงค่าความต้านทานของตัวเอง เช่น เซ็นเซอร์แสง หรือ เซ็นเซอร์อุณหภูมิ
- อุปกรณ์อิเลคทรอนิคส์ใดๆ ที่ผลิตความต่างศักดิ์ ระหว่าง 0 ถึง 5 โวลต์

คำสั่งที่ใช้อ่านค่าเซ็นเซอร์มีอยู่สองแบบด้วยกัน แบบแรกเรียกว่า switch ซึ่งจะส่งค่าจริงหรือเท็จกลับมาเท่านั้น (ใช้กับเซ็นเซอร์ที่มีสองสถานะคือเปิดหรือปิด) แบบที่สองคือsensor ซึ่งจะส่งค่าตัวเลขระหว่าง 0 ถึง 1023 ออกมา โดยค่าที่ได้จะขึ้นอยู่กับเซ็นเซอร์ที่ใช้

`Switch1`
`Switch2`
`Switch3`
`Switch4`
`Switch5`
`Switch6`
`Switch7`
`Switch8`
	ถ้าสวิตช์ที่ต่ออยู่กับเซ็นเซอร์ 1–8 **ถูกกด**อยู่ คำสั่งนี้จะรายงานค่า **"จริง"** ออกมา ถ้าไม่เช่นนั้นก็จะรายงานค่า "เท็จ"

`Sensor1`
`Sensor2`
`Sensor3`
`Sensor4`
`Sensor5`
`Sensor6`
`Sensor7`
`Sensor8`
	รายงานค่าเซ็นเซอร์ 1–8 โดยค่านี้จะอยู่ระหว่าง 0 ถึง 1023

คำสั่งโครงสร้าง

ภาษา Logo มีคำสั่งโครงสร้างชุดเล็กๆ ที่มีประโยชน์มาก คำสั่งเหล่านี้ประกอบไปด้วยคำสั่งที่ใช้ในการวนรอบ ทดสอบเงื่อนไข รอแบบมีเงื่อนไข และคำสั่งจบการทำงานของโปรแกรม

ภาพรวม

คำสั่งโครงสร้างที่มีใน ภาษา Logo ถูกสรุปไว้ในตารางต่อไปนี้

`repeat` ***ครั้ง*** [*คำสั่ง*]	วนทำ "คำสั่ง" เป็นจำนวนครั้งเท่ากับ "**ครั้ง**" ค่า "ครั้ง" อาจเป็นค่าคงที่ ค่าจากการคำนวณ หรือตัวแปรก็ได้
`forever` [*คำสั่ง*]	วนทำ "คำสั่ง" ไปแบบไม่มีที่สิ้นสุด

คำสั่งเงื่อนไข

`if` *เงื่อนไข* [*คำสั่ง*]	ถ้า "เงื่อนไข" เป็นจริง จะทำ "คำสั่ง" เงื่อนไขที่มีค่าเป็น 0 ถือว่ามีค่าเป็น **เท็จ** ค่าอื่นที่**ไม่ใช่** 0 จะถือว่ามีค่าเป็น **จริง**
`ifelse` เงื่อนไข [*คำสั่ง–1*] [*คำสั่ง–2*]	ถ้า "เงื่อนไข" เป็นจริง จะทำ "คำสั่ง–1" ถ้าไม่เช่นนั้นจะทำ "คำสั่ง–2"
`waituntil [`*เงื่อนไข*`]`	โปรแกรมจะรอและไม่ทำคำสั่งถัดไปจนกระทั่ง "เงื่อนไข" เป็นจริง โปรดสังเกตว่าเงื่อนไขจะต้องอยู่ในวงเล็บเหลี่ยม ไม่เหมือนกับคำสั่ง if และ ifelse ที่เงื่อนไขไม่ต้องอยู่ในวงเล็บใดๆ
`stop`	หยุดการทำงานของ procedure ปัจจุบัน และ กลับไปทำคำสั่งถัดไปใน procedure แม่ (procedure ที่เรียกใช้งาน procedure ปัจจุบัน)
`output` *ค่า*	หยุดการทำงานของ procedure ปัจจุบัน และ ส่ง "ค่า" กลับไปยัง procedure แม่

ตัวอย่าง

procedure ต่อไปนี้จะทำให้มอเตอร์ A หมุนกลับไปกลับมา 10 ครั้ง

```
to flippy
  repeat 10 [a, onfor 10 rd]
end
```

ตัวอย่างต่อไปนี้แสดงให้เห็นวิธีที่จะทำให้มอเตอร์ A หมุนกลับไปกลับมาเรื่อยๆ ไม่มีวันสิ้นสุด

```
to flippy-forever-1
  forever [a, onfor 10 rd]
end
```

procedure ต่อไปนี้ทำการเปิดมอเตอร์ A แล้วรอจดกว่าสวิทช์ 2 ถูกกด แล้วจึงปิดมอเตอร์

```
to on-wait-off
  a, on
  waituntil [switch2]
  off
end
```

procedure ต่อไปนี้จะทำการอ่านค่าสวิทช์ 2 อยู่เรื่อยๆ ถ้าสวิทช์ถูกกดมอเตอร์ A จะหมุนไป "ทางนี้" แต่ถ้าสวิทช์ไม่ถูกกดมอเตอร์จะหมุนไป "ทางนั้น"

```
to switch-controls-direction
  a, on
  forever [
    ifelse switch2 [thisway][thatway]
  ]
end
```

ระบบตัวเลข

GoGo Board ใช้ระบบตัวเลขขนาด 16 บิท ซึ่งหมายความว่าค่าตัวเลขที่สามารถใช้งานได้จะอยู่ระหว่าง –32768 ถึง +32767

การใช้งานเครื่องหมายทางเลขคณิตจะต้องมีการเว้นวรรคทั้งสองด้านเสมอ นั่นคือการเขียน **3+4** เป็นรูปแบบที่ผิด รูปแบบที่ถูกคือ **3 + 4** (มีวรรคก่อนหน้าและหลังเครื่องหมายบวก)

ภาษา Logo ไม่ได้ใช้ระบบลำดับความสำคัญของเครื่องหมายคณิตศาสตร์ที่เป็นมาตรฐานทั่วไป แต่จะถือเอาตามลำดับการเขียนเรียงจากซ้ายไปขวา ดังนั้น

```
3 + 4 * 5
```

จะมีค่าเท่ากับ 35 เพราะ ภาษา Logo จะทำ 3 + 4 แล้วคูณผลลัพธ์ด้วย 5 (ซึ่งต่างจากมาตรฐานการประมวลผลในภาษาคอมพิวเตอร์ทั่วไป ซึ่งจะถือว่า * สำคัญกว่า + ดังนั้นผลลัพธ์ที่จะได้จะเป็น 4*5 แล้วบวกด้วย 3)

วงเล็บเป็นวิธีการที่ใช้ในการ**กำหนดลำดับก่อนหลัง**ให้กับการคำนวณ เช่น

```
(3 + (4 * 5))
```

ค่าที่ได้คือ 23.

ตารางต่อไปนี้แสดงเครื่องหมายทางคณิตศาสตร์ทั้งหมดที่มีใน ภาษา Logo

`+`	บวก (แบบ infix)
`-`	ลบ (แบบ infix)
`*`	คูณ (แบบ infix)
`/`	หาร (แบบ infix)
`%`	หารเอาเศษ (เช่น 5 % 3 จะเท่ากับ 2)
`and`	ตรรกะ "และ" ใช้ทั้งกับการหาผลลัพธ์ทางตรรกศาสตร์ (จริงหรือเท็จ) และ bitwise operation
`or`	ตรรกะ "หรือ"
`not`	ตรรกะ "ไม่"
`random`	ใช้สุ่มค่าตัวเลข ค่าที่ได้จะอยู่ระหว่าง –32768 ถึง +32768. ถ้าต้องการลดช่วงของค่าลง ให้ใช้การหารเอาเศษ (%) เช่น (random % 100) จะได้ค่าสุ่ม

ตั้งแต่ 0 ถึง 99

Procedures และการรับ-ส่งค่า (input-output)

คำจำกัดความ

การสร้าง procedure จะเริ่มด้วยคำสั่ง to ตามด้วยชื่อ procedure ตามด้วยชุดคำสั่งที่เป็นใจความของ procedure แล้วจบด้วยคำสั่ง end ตัวอย่างต่อไปนี้ เป็นการสร้าง procedure ชื่อว่า flash ซึ่งทำการเปิดและปิดมอเตอร์ A สิบครั้ง

```
to flash
  repeat 10 [a, onfor 5 wait 5]
end
```

การรับค่า (Inputs)

เราสามารถกำหนด Procedures ให้ทำการรับค่าได้ ซึ่งค่าดังกล่าวจะกลายเป็นตัวแปรของ procedure นั้นๆ (local variable) การกำหนดการรับค่าจะทำโดยการใช้เครื่องหมาย colon (:) ตัวอย่างต่อไปนี้เป็นการสร้าง procedure ชื่อ flash ซึ่งมีการรับค่าหนึ่งค่า (ชื่อว่า n) ค่าที่รับเข้ามานี้ถูกใช้ในการกำหนดจำนวนครั้งการวนรอบของคำสั่ง repeat

```
to flash :n
  repeat :n [a, onfor 5 wait 5]
end
```

เมื่อเรียกใช้ procedure นี้จะต้องตามด้วยค่าตัวเลขหนึ่งค่าเสมอ เช่น flash 5, flash 10, flash 20, ฯลฯ

procedure สามารถรับค่ากี่ค่าก็ได้ ภาษา Logo ไม่ได้จำกัดไว้ แต่ในทางปฏิบัติปริมาณหน่วยความจำที่เหลืออยู่ของ GoGo Board จะเป็นตัวจำกัด

การส่งค่า (Outputs)

Procedure สามารถส่งค่ากลับได้โดยใช้คำสั่ง `output` เมื่อ procedure เรียกใช้คำสั่งดังกล่าวแล้ว มันจะจบการทำงานทันที ตัวอย่างต่อไปนี้แสดง procedure ชื่อ detect ซึ่งจะส่งค่า 0, 1, หรือ 2 ขึ้นอยู่กับค่าของเซ็นเซอร์ 1

```
to detect
  make “temp sensor1
  if :temp < 300 [output 1]
  if :temp < 500 [output 2]
  output 3
end
```

ในตัวอย่างนี้มีการสร้างตัวแปรชื่อ temp ซึ่งถูกใช้ในการเก็บค่าของเซ็นเซอร์ 1 ถ้าค่าเซ็นเซอร์นี้น้อยกว่า 300 procedure จะส่งค่า 1 แต่ถ้าค่าเซ็นเซอร์มากกว่า 300 คำสั่งถัดไปจะทำงาน ซึ่งจะทดสอบว่าถ้าค่าดังกล่าวน้อยกว่า 500 procedure จะส่งค่า 2 ท้ายที่สุดถ้าค่าเซ็นเซอร์ไม่น้อยกว่า 500 procedure จะส่งค่า 3

ข้อควรระวัง ถ้าตัดสินใจใช้คำสั่ง output แล้ว จะต้องตรวจสอบให้แน่ใจเสมอว่า procedure นั้นจะมีการส่งค่าไม่ว่าในกรณีใดๆ (นั่นคือ procedure จะส่งค่าบ้างไม่ส่งค่าบ้างไม่ได้) การทำงานของ GoGo Board จะล้มเหลวทันทีถ้า procedure นั้นจบการทำงานโดยไม่มีการส่งค่า

ตัวแปร

การสร้างตัวแปร Global จะทำโดยใช้คำสั่ง **make “*ชื่อตัวแปร*** เช่น

```
Make “cats 0
make “dogs 1
```

จะสร้างตัวแปรสองตัวชื่อ cats และ dogs และกำหนดค่าให้เป็น 0 และ 1 ตามลำดับ

จะสังเกตได้ว่าเวลาเราจะเขียนค่าลงในตัวแปรเราจะใส่เครื่องหมายคำพูดนำหน้าชื่อตัวแปรเสมอ แต่ถ้าต้องการอ่านค่าจากตัวแปรจะใช้เครื่องหมายโคล่อน : แทน เช่น

```
If :cats > 0 [ a, on ]
```

จะทดสอบว่าค่าในตัวแปร cats มากกว่า 0 หรือไม่ ถ้าใช่ก็จะเปิดมอเตอร์ A

```
Make “cats :cats + 1
```

จะเพิ่มค่าตัวแปร cats ขึ้น 1

ตัวแปรจะถูกจัดเก็บในหน่วยความจำ RAM ซึ่งจะต้องมีไฟเลี้ยงอยู่เสมอ ดังนั้นค่าตัวแปรจะสูญหายถ้าปิด GoGo Board

การบันทึกและเรียกคืนข้อมูล (Data Recording and Playback)

ภาษา Logo มี global array ขนาด 8000 ช่อง อยู่หนึ่งตัวซึ่งสามารถใช้งานกับคำสั่งต่อไปนี้ได้

`setdp` ตำแหน่ง ตั้งค่าตัวชี้ตำแหน่ง

`record` ค่า บันทึก "ค่า" ลงไปในตำแหน่งปัจจุบัน และเลื่อนตัวชี้ให้ไปอยู่ในตำแหน่งถัดไป

`recall` เรียกคืนค่าในตำแหน่งปัจจุบัน และเลื่อนตัวชี้ให้ไปอยู่ในตำแหน่งถัดไ

ตัวอย่าง procedure ชื่อ take-data ต่อไปนี้จะบันทึกค่าเซ็นเซอร์ A ทุกๆ หนึ่งวินาที

```
to take-data
  setdp 0
  repeat 1000 [record sensor1 wait 10]
end
```

เมื่อมีการบันทึกข้อมูลแล้ว สามารถใช้ GoGo Monitor ในการดึงข้อมูลมาเก็บไว้ในรูปแบบของไฟล์ CSV (Comma Separated Values) เพื่อนำไปประมวลผลโดยใช้โปรแกรมอื่นๆ เช่น Excel ต่อไป

หมายเหตุ ภาษา Logo ไม่มีการตรวจสอบว่าการบันทึกข้อมูลเกินขอบเขตหรือไม่ (มากกว่า 8000) ซึ่งอาจส่งผลให้เกิดความผิดพลาดกับข้อมูลหรือโปรแกรมส่วนอื่น

การสื่อสารข้อมูลทางช่องสัญญาณอนุกรม (Serial Port)

ภาพรวม

GoGo Board สามารถส่งและรับข้อมูลหากันผ่านทางช่องสัญญาณอนุกรมได้โดยใช้คำสั่ง send และ serial ตามลำดับ คำสั่ง serial จะรายงานค่าที่ได้รับล่าสุด นอกจากนี้ยังมีคำสั่ง newserial? ซึ่งจะรายงานค่าจริงถ้า GoGo Board ได้รับข้อมูลตัวใหม่เข้ามาแต่ยังไม่ได้ถูกนำไปใช้

ลองพิจารณาตัวอย่างต่อไปนี้ procedure ชื่อ sender จะทำการส่งข้อมูลไปยังคอมพิวเตอร์, โดยค่าที่ส่งจะเป็นค่าสุ่มระหว่าง 0 ถึง 2

```
to sender
  forever [
    send random % 3
    beep
    wait 30
]
end
```

คำสั่ง random % 3 จะสร้างค่า 0, 1, หรือ 2 ซึ่งเป็นผลจากการใช้เครื่องหมาย "หารเอาเศษ" ค่าที่ได้จะถูกส่งโดยคำสั่ง send หลังจากนั้น ก็จะส่งเสียง beeps และรอ 3 วินาทีก่อนที่จะวนสร้างและส่งค่าออกไปอีกครั้ง

ในตัวอย่างที่สอง Procedure ชื่อ doit จะทำการรับค่าที่ Computer แล้วจะเปิดมอเตอร์ A, มอเตอร์ B หรือทั้งคู่ ขึ้นอยู่กับค่าที่มันได้รับ

```
to doit
  forever [
    waituntil [newserial?]
    if serial = 0 [a, onfor 10]
    if serial = 1 [b, onfor 10]
    if serial = 2 [ab, onfor 10]
  ]
end
```

หมายเหตุ

GoGo Board ใช้ค่า 128 ถึง 134 สำหรับการทำงานระดับต่ำระหว่าง GoGo Board ดังนั้นพยายามอย่าส่งค่าตัวเลขเหล่านี้ ค่าเหล่านี้อาจทำให้หน่วยความจำของ GoGo Board บางตัว (ตัวที่เปิดอยู่แต่ไม่ได้ทำงานอะไร) ถูกเขียนทับและส่งผลเสียหายได้

ปุ่มบนตัว GoGo Board

เมื่อกดปุ่ม “Run” ในขณะที่ GoGo Board ไม่ได้ทำงานอะไร จะทำให้มันเริ่มต้นประมวลผล procedure แรกที่เขียนไว้ในหน้าจอของ ภาษา Logo

ถ้ากดปุ่ม “Run” ในขณะที่ GoGo Board กำลังทำงานจะส่งผลให้มันหยุดการทำงานปัจจุบันทันที

การรับค่าคำสั่งจากรีโมทอินฟราเรด

GoGo Board สามารถรับคำสั่งจากรีโมท SONY ใดๆ ผ่านทางตัวรับสัญญาณอินฟราเรดบนบอร์ด โดยค่าที่ได้รับนั้นจะเป็นค่ารหัสของปุ่มบนรีโมท ซึ่งไม่จำเป็นต้องตรงกับค่าตัวเลขที่เห็นบนรีโมท เช่น เลข 1 บนรีโมทจะมีรหัส = 128 เป็นต้น

ปุ่มเปิด/ปิด บนรีโมทเป็นปุ่มพิเศษ ถ้า GoGo Board ได้รับค่าปุ่ม เปิด/ปิด นี้มันจะทำการ เริ่ม/หยุด โปรแกรมเหมือนกับการกดปุ่ม Run

คำสั่งในภาษา Logo เกี่ยวกับรีโมทมีดังต่อไปนี้

`newir?` จะมีค่าเป็นจริงถ้า มีการส่งคำสั่งรีโมทมายัง GoGo Board

`ir` จะคืนค่ารหัสของปุ่มรีโมทที่ถูกส่งมา

ตัวอย่างต่อไปนี้จะเลือกทำคำสั่ง beep ถ้ามีการกดปุ่ม “1” บนรีโมท (รหัส = 128) และจะกระพริบ LED ถ้ามีกดปุ่ม “2” (รหัส 129)

```
to testIR
  forever [
    waituntil [newir?]
    if ir = 128 [beep]
    if ir = 129 [ledon wait 10 ledoff]
  ]
end
```

10 ตัวอย่างโปรแกรมควบคุม GoGo Board

ตัวอย่างที่ 1

สั่งให้ GoGo Board ส่งเสียง beep 5 ครั้งในช่วงเวลาห่างกัน 1 วินาที

```
to example1
  repeat 5 [ beep wait 10 ]
end
```

ตัวอย่างที่ 2

สั่งให้มอเตอร์ a หมุนเป็นเวลา 2 วินาที และหมุนกลับทิศเป็นเวลา 2 วินาที

```
to example2
  a, on wait 20
  rd
  wait 20
  off
end
```

ตัวอย่างที่ 3

สั่งให้ไฟ LED หรือมอเตอร์ ที่ต่อกับพอร์ต A ทำงานเป็นเวลา 2 วินาทีและหยุดเป็นเวลา 1 วินาที โดยทำงานซ้ำทั้งหมด 5 ครั้ง

```
to example3
  repeat 5 [a, on wait 20 a, off wait 10]
end
```

ตัวอย่างที่ 4

สั่งให้ ส่งเสียง Beep ทุกครั้งที่เซนเซอร์ 1 มีค่าน้อยกว่า 500

```
to example4
  forever [ if sensor1 < 500 [ beep ] ]
end
```

ตัวอย่างที่ 5

สั่งให้ไฟ LED หรือมอเตอร์ ที่ต่อกับพอร์ต A ทำงานทุกครั้งที่เซนเซอร์ 1 มีค่าน้อยกว่า 500 และ สั่งให้ปิดไฟ LED หรือปิดมอเตอร์ ถ้าเซนเซอร์1มีค่ามากกว่า 500

```
to example5
  forever [ifelse sensor1 < 500 [a, on] [ a, off ] ]
end
```

ตัวอย่างที่ 6

เก็บค่าเซนเซอร์ 1 ทุกๆ 1 วินาที เป็นเวลา 1 นาที

```
to example6
  resetdp
  repeat 60 [ record sensor1 beep wait 10 ]
end
```

ตัวอย่างที่ 7

สั่งให้ไฟ LED หรือมอเตอร์ ที่ต่อกับพอร์ต A ทำงานทุกครั้งที่เซนเซอร์ 1 มีค่าน้อยกว่า 500 และ
สั่งให้ปิดไฟ LED หรือปิดมอเตอร์พอร์ต A ถ้าเซนเซอร์1 มีค่ามากกว่า 500 พร้อมๆ กับ
สั่งให้ไฟ LED หรือมอเตอร์ ที่ต่อกับพอร์ต Bทำงานทุกครั้งที่เซนเซอร์ 2 มีค่าน้อยกว่า 500 และ
สั่งให้ปิดไฟ LED หรือปิดมอเตอร์พอร์ต B ถ้าเซนเซอร์ 2 มีค่ามากกว่า 500

```
forever [
   ifelse sensor1 < 500 [a, on] [a, off]
   ifelse sensor2 < 500 [b, on] [b, off]
  ]
end
```

ตัวอย่างที่ 8

สั่งให้ GoGo Board เปิดมอเตอร์พอร์ท A เมื่อเงื่อนไขทั้งสองต่อไปนี้**เป็นจริงพร้อมกัน**

ถ้าเซ็นเซอร์หมายเลข 1 มีค่าน้อยกว่า 500 และ

ถ้าเซ็นเซอร์หมายเลข 2 มีค่ามากกว่า 500

ถ้าไม่ตรงตามเงื่อนไขให้ปิดมอเตอร์ A

```
to example8
 forever [
        ifelse (sensor1 < 500) and (sensor2 > 500)
        [a, on][a, off]
 ]
end
```

ตัวอย่างที่ 9

สั่งให้ GoGo Board เปิดมอเตอร์พอร์ท A เมื่อเงื่อนไข**อันใดอันหนึ่ง**ต่อไปนี้**เป็นจริง**

ถ้าเซ็นเซอร์หมายเลข 1 มีค่าน้อยกว่า 500 หรือ

ถ้าเซ็นเซอร์หมายเลข 2 มีค่ามากกว่า 500

ถ้าไม่ตรงตามเงื่อนไขใดๆ ข้างต้นให้ปิดมอเตอร์ A

```
to example9
  forever [
         ifelse (sensor1 < 500) or (sensor2 > 500)
         [a, on][a, off]
  ]
end
```

ตัวอย่างที่ 10

สั่งให้ GoGo Board คอยตรวจสอบค่าเซ็นเซอร์หมายเลข 1 เมื่อใดก็ตามที่ค่าเซ็นเซอร์ต่ำกว่า 500 ให้นับค่าโดยเพิ่มค่าของตัวแปรชื่อ count ขึ้น 1 แล้วสั่งให้ GoGo Board ส่งเสียงบี๊บเป็นจำนวนครั้งเท่ากับค่าที่นับได้ในขณะนั้น

```
to example10
  make “count 0
  forever [
         if sensor1 < 500 [
                make “count :count + 1
                repeat :count [beep wait 5 ]
                waituntil [sensor1 > 499]
         ]
  ]
end
```