เครื่องปล่อยสินค้าอัจฉริยะ

VENDING MACHINE

พชกรณ์ รุ่งโรจน์สุวรรณ

พชร ภูตระกูล

พลาพันธ์ วันดี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

เครื่องปล่อยสินค้าอัจฉริยะ

VENDING MACHINE

พชกรณ์ รุ่งโรจน์สุวรรณ

พชร ภูตระกูล

พลาพันธ์ วันดี

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมระบบควบคุม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2556

VENDING MACHINE

PATCHAKORN RUNGROJSUWAN
PACHARA PHUTRAKOON
PHALAPAN WANDEE

THIS THESIS IS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ENGINEERING IN CONTROL ENGINEERING
FACULTY OF ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2013

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมการวัดและควบคุม คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เครื่องปล่อยสินค้าอัจฉริยะ

VENDING MACHINE

ผู้จัดทำ นายพชกรณ์ รุ่งโรจน์สุวรรณ 53011042

นายพชร ภูตระกูล 53011044

นายพลาพันธ์ วันดี 53011076

.....อาจารย์ที่ปรึกษา

(ดร.รัชนี กุลยานนท์)

เครื่องปล่อยสินค้าอัจฉริยะ

โดย
นายพชกรณ์ รุ่งโรจน์สุวรรณ 53011042
นายพชร ภูตระกูล 53011044
นายพลาพันธ์ วันดี 53011076

อาจารย์ที่ปรึกษา
ดร.รัชนี กุลยานนท์

ปีการศึกษา 2556

บทคัดย่อ

โครงงานฉบับนี้เป็นการจัดทำเครื่องปล่อยสินค้าอัตโนมัติ โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวประมวลผลและจัดเก็บข้อมูล เพื่อให้เครื่องปล่อยสินค้าได้ตรงตามความต้องการของผู้ใช้ อีกทั้งยังเป็นการศึกษาการใช้งานของไมโครคอนโทรลเลอร์ และบอร์ดขับมอเตอร์ไฟฟ้ากระแสตรง สำหรับในโครงงานนี้จะนำเอามอเตอร์ไฟฟ้ากระแสตรงมาประยุกต์ใช้กับเครื่องปล่อยสินค้าอัตโนมัติ โดยจะใช้มอเตอร์ไฟฟ้ากระแสตรงเป็นตัวดันให้สินค้าถูกปล่อยตามความต้องการของผู้ใช้ ซึ่งจะสั่งผ่านรีโมทคอนโทรลไปยังไมโครคอนโทรลเลอร์โดยผู้ใช้ เพื่อประมวลผลแล้วส่งสัญญาณควบคุมไปยังบอร์ดขับมอเตอร์ไฟฟ้ากระแสตรง ทำให้มอเตอร์ไฟฟ้ากระแสตรงดันสินค้าได้ตรงตามความต้องการ จากการทดลองและการออกแบบโครงสร้างเครื่องปล่อยสินค้า จะพบว่าโครงงานนี้สามารถสั่งให้มอเตอร์ไฟฟ้ากระแสตรงดันสินค้าให้ตรงตามความต้องการของผู้ใช้ได้

VENDING MACHINE

By

Mr.Patchakorn Rungrojsuwan 53011042

Mr.Pachara Phutrakoon 53011044

Mr.Phalapan Wandee 53011076

Advisor

Dr.Rutchanee Gullayanon

Academic Year 2013

ABSTRACT

This project is to make an automatic vending machine by using a micro controller as the data processing and collecting in order to command the machine to release product correctly for user's demand .Besides, it is the study of how to use a micro controller and DC motor drives. For this project, We bring DC motor drives to apply with the automatic vending machine. DC motor drives are the main factor which pushes a product to release for meeting the user's demand. It is ordered through remove control towards a micro controller by the user so as to process the data and sent the control signal to DC motor drives. This method makes DC motor drives can push the correct product. Form this experiment and vending machine structure design, we discover that this project can command DC motor drives in pushing the right product for the user's demand.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ ประสบความสำเร็จลุล่วงได้ดี เนื่องด้วยคำปรึกษา แนะนำ ความช่วยเหลือเป็นอย่างดีจาก ดร.รัชนี กุลยานนท์ ซึ่งเป็นอาจารย์ควบคุมปริญญานิพนธ์ ผู้จัดทำรู้สึกซาบซึ้งและขอกราบขอบพระคุณท่านเป็นอย่างสูง

ขอขอบพระคุณคณาจารย์ทุกท่านที่ให้ความรู้ความเข้าใจในเนื้อหาวิชาตั้งแต่เริ่มเข้ารับการศึกษา เพื่อนำความรู้ที่ได้มาประยุกต์ใช้ในการทำปริญญานิพนธ์ฉบับนี้

ขอขอบพระคุณพี่ๆ ทุกท่านและเพื่อนๆ ทุกคนที่ให้คำแนะนำและคำปรึกษา ตลอดจนช่วยเหลือในการทำปริญญานิพนธ์ฉบับนี้

สุดท้ายนี้ขอกราบขอบพระคุณบิดา มารดา ของผู้จัดทำ ผู้มีพระคุณสูงสุด ผู้ให้โอกาสในการศึกษา ตลอดจนให้คำปรึกษาและความช่วยเหลือด้านต่างๆ และเป็นกำลังใจในการทำปริญญานิพนธ์ฉบับนี้ให้สำเร็จลุล่วงไปด้วยดี

ผู้จัดทำหวังเป็นอย่างยิ่งว่าปริญญานิพนธ์ฉบับนี้จะเป็นประโยชน์ต่อผู้ที่สนใจ และหากเกิดข้อผิดพลาดประการใด ทางคณะผู้จัดทำต้องขออภัยมา ณ โอกาสนี้ด้วย

ผู้จัดทำ
พชกรณ์ รุ่งโรจน์สุวรรณ
พชร ภูตระกูล
พลาพันธ์ วันดี

สารบัญ

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญ(ต่อ)

สารบัญรูป

สารบัญรูป(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

ปัญหาจากการลำเลียงสินค้าในโกดังเกิดจากหลายสาเหตุ ทั้งความล่าช้าความไม่เป็นระเบียบและการขโมยสินค้า ซึ่งในปัจจุบันทางโรงงานใช้วิธีให้พนักงานเดินเข้าไปหยิบสินค้าตามใบสั่งซื้อ โดยปัญหาที่พบ คือการหยิบสินค้าไม่ตรงตามใบสั่งซื้อมีความล่าช้าในการหาสินค้า สิ่งเหล่านี้เป็นปัญหาที่เกิดขึ้นจากตัวบุคคลก่อให้เกิดความเสียหายทั้งในแง่ชื่อเสียงบริษัทและเวลาการทำงาน เราจึงได้เล็งเห็นถึงวิธีการแก้ปัญหาโดยเราจะออกแบบเครื่องปล่อยสินค้าอัตโนมัติที่มีความแม่นยำสูง โดยเราจะเริ่มศึกษาและคิดออกแบบกลไกการทำงานของเครื่องจักร รวมไปถึงรูปแบบโครงสร้างของตัวปล่อยสินค้า เพื่อให้ระบบการปล่อยสินค้ามีประสิทธิภาพมากที่สุด

1.2 วัตถุประสงค์ในการทำปริญญานิพนธ์

1. เพื่อช่วยลดข้อผิดพลาดในการจ่ายสินค้า
2. ลดการขโมยสินค้า
3. เพิ่มความเป็นระเบียบในการเก็บสินค้า

1.3 ขอบเขตของโครงงาน

1. สามารถลดพื้นที่ในการจัดเก็บสินค้า
2. สามารถสั่ง Arduino ให้ควบคุมกระบอกสูบไฟฟ้าตามที่ต้องการได้
3. สามารถตรวจเช็ค และปล่อยสินค้าได้ตามต้องการ
4 สามารถประยุกต์ใช้ได้ในโรงงานต่างๆ

1.4 ขั้นตอนการศึกษาและจัดทำโครงงาน

โครงงานนี้เป็นการออกแบบโครงสร้างของรางปล่อยสินค้า และสามารถควบคุมโดยรีโมทคอนโทรล โดยโครงงานนี้แบ่งออกเป็น 2 ส่วน คือ

1. ส่วนฮาร์ดแวร์ ออกแบบโครงสร้างรางปล่อยสินค้าให้สามารถปล่อยสินค้าจากรางได้และสามารถเก็บสินค้าได้มากที่สุด

2. ส่วนซอฟแวร์ ออกแบบโปรแกรมให้สามารถควบคุมมอเตอร์และเซนเซอร์ตามคำสั่งได้

1.5 รายละเอียดของโครงงาน

เนื้อหาที่จะกล่าวในปริญญานิพนธ์ฉบับนี้ประกอบด้วย

บทที่ 1 บทนำ กล่าวถึงความเป็นมาของโครงงาน วัตถุประสงค์ของโครงงาน ขั้นตอนการศึกษาและการจัดทำโครงงาน พร้อมทั้งรายละเอียดของโครงงานแต่ละบท

บทที่ 2 ทฤษฎีและความรู้เกี่ยวกับพร็อกซิมิตี้เซนเซอร์ มอเตอร์ไฟฟ้า และไมโครคอนโทรลเลอร์ โดยนำเสนอถึงหลักการและทฤษฎีที่เกี่ยวข้องในการจัดทำโครงงาน

บทที่ 3 การออกแบบ SolidWorks และโครงสร้างของระบบ อธิบายภาพรวมทั้งหมดของระบบ ขั้นตอนการออกแบบตัวต้นแบบ รวมไปถึงการเลือกใช้อุปกรณ์

บทที่ 4 การทดลองและผลการทดลอง การสั่งสินค้าในแต่ละราง และคำสั่งรีเซ็ท

บทที่ 5 บทวิจารณ์และสรุป เป็นการสรุปผลการดำเนินงาน ปัญหาที่เกิดขึ้น และแนวทางการปรับปรุงพัฒนาโครงงานนี้ต่อไป

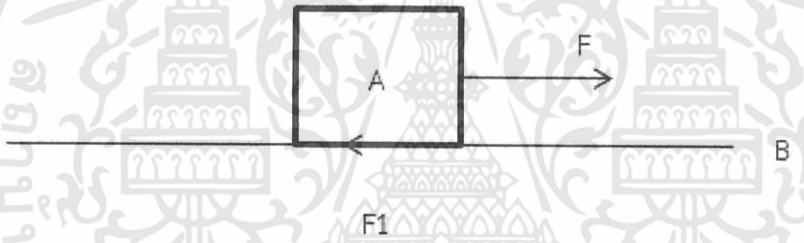
บทที่ 2
ทฤษฎีหรือหลักการ

2.1 ความรู้เบื้องต้นเกี่ยวกับแรงเสียดทาน

2.1.1 แรงเสียดทาน (Friction)

เป็นแรงที่เกิดขึ้นเมื่อวัตถุหนึ่งพยายามเคลื่อนที่ หรือกำลังเคลื่อนที่ไปบนผิวของอีกวัตถุ เนื่องจากมีแรงมากระทำ มีลักษณะที่สำคัญดังนี้

- เกิดขึ้นระหว่างผิวสัมผัสของวัตถุ
- มีทิศทางตรงกันข้ามกับทิศทางที่วัตถุเคลื่อนที่ หรือตรงข้ามทิศทางของแรงที่พยายามทำให้วัตถุเคลื่อนที่ ถ้าวาง A อยู่บนวัตถุ B ออกแรง F ลากวัตถุ วัตถุ A จะเคลื่อนที่หรือไม่ก็ตาม จะมีแรงเสียดทานเกิดขึ้นระหว่างผิวของ A และ B แรงเสียดทานมีทิศทางตรงกันข้ามกับแรง F ที่พยายามต่อต้านการเคลื่อนที่ของ A ดังรูปที่ 2.1



รูปที่ 2.1 การเคลื่อนที่บนพื้นราบ

2.1.2 ประเภทของแรงเสียดทาน

แรงเสียดทานมี 2 ประเภท คือ

2.1.2.1 แรงเสียดทานสถิต (Static Friction)

คือ แรงเสียดทานที่เกิดขึ้นระหว่างผิวสัมผัสของวัตถุ ในสภาวะที่วัตถุได้รับแรงกระทำแล้วอยู่นิ่ง

2.1.2.2 แรงเสียดทานจลน์ (Kinetic Friction)

คือ แรงเสียดทานที่เกิดขึ้นระหว่างผิวสัมผัสของวัตถุ ในสภาวะที่วัตถุได้รับแรงกระทำแล้วเกิดการเคลื่อนที่ด้วยความเร็วคงที่

2.1.3 สมบัติของแรงเสียดทาน

- แรงเสียดทานมีค่าเป็นศูนย์ เมื่อวัตถุไม่มีแรงภายนอกมากระทำ

- ขณะที่มีแรงภายนอกมากระทำต่อวัตถุ และวัตถุยังไม่เคลื่อนที่ แรงเสียดทานที่เกิดขึ้นมีขนาดต่างๆ กัน ตามขนาดของแรงที่มากระทำ และแรงเสียดทานที่มีค่ามากที่สุดคือ แรงเสียดทานสถิต เป็นแรงเสียดทานที่เกิดขึ้นเมื่อวัตถุเริ่มเคลื่อนที่

- แรงเสียดทานมีทิศทางตรงกันข้ามกับการเคลื่อนที่ของวัตถุ

- แรงเสียดทานสถิตมีค่าสูงกว่าแรงเสียดทานจลน์เล็กน้อย

- แรงเสียดทานจะมีค่ามากหรือน้อยขึ้นอยู่กับลักษณะของผิวสัมผัส ผิวสัมผัสหยาบหรือขรุขระจะมีแรงเสียดทานมากกว่าผิวเรียบและลื่น

- แรงเสียดทานขึ้นอยู่กับน้ำหนักหรือแรงกดของวัตถุที่กดลงบนพื้น ถ้าน้ำหนักหรือแรงกดมากแรงเสียดทานก็จะมากขึ้นด้วย

- แรงเสียดทานไม่ขึ้นอยู่กับขนาดหรือพื้นที่ของผิวสัมผัส

แรงเสียดทานนั้นจะขึ้นอยู่กับพื้นผิวของวัตถุแต่ละชนิด ซึ่งจะค่าสัมประสิทธิ์ความเสียดทานแตกต่างกันออกไปดังตารางที่ 2.1

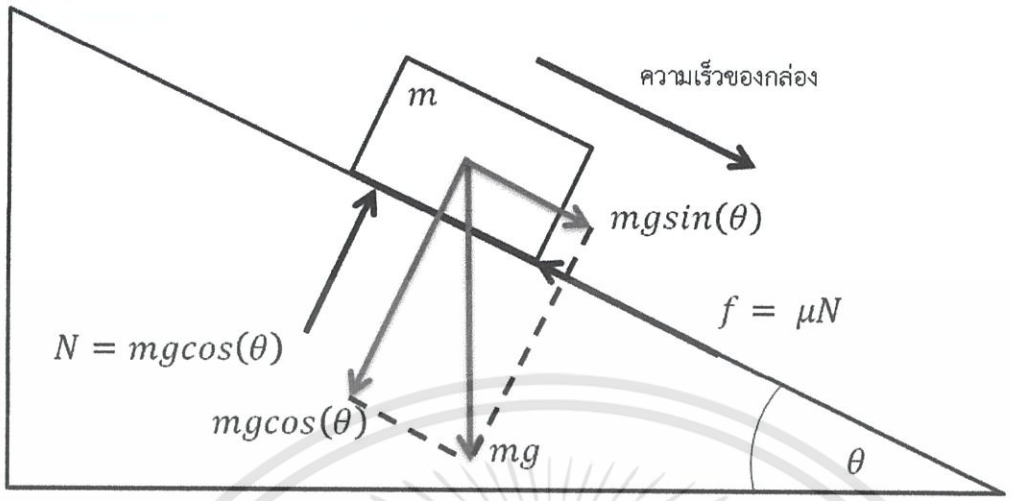
ตารางที่ 2.1 ตารางแสดงค่าสัมประสิทธิ์ความเสียดทานสถิต

วัสดุ	ค่าสัมประสิทธิ์ความเสียดทานสถิต(μ_{static})
ทองแดง	0.44
เหล็ก	0.45
ยาง	0.70
กระดาษ	0.84

2.1.4 การคำนวณหาค่าสัมประสิทธิ์ความเสียดทานพื้นเอียง

หากทราบค่าสัมประสิทธิ์ความเสียดทานจะสามารถคำนวณหามุม (θ) ได้ เมื่อ $mgsin(\theta) \leq f$ แสดงว่าวัตถุจะหยุดนิ่ง และเมื่อ $mgsin(\theta) > f$ วัตถุจะเคลื่อนที่ลงด้วยความเร็ว ดังรูปที่ 2.2

กำหนดให้ m คือ มวลของวัตถุ f คือ แรงเสียดทาน μ คือ ค่าสัมประสิทธิ์ความเสียดทาน



รูปที่ 2.2 การเคลื่อนที่บนพื้นเอียง

2.2 ความรู้เบื้องต้นเกี่ยวกับทฤษฎี Finite State Machine Design

Finite State Machine Design คือ วงจรเชิงลำดับ ซึ่งออกแบบเป็นสถานการณ์ทำงานของวงจรออกเป็นหลายๆ สถานะ แต่ละสถานะจะมีลอจิกการทำงานที่ต่างกัน เพื่อเกิดค่าเอาต์พุตและค่าสถานะถัดไป มีสัญญาณสถานะที่กำหนดว่าสถานะปัจจุบันเป็นสถานะไหน สัญญาณของสถานะจะถูกเก็บไว้ในรีจิสเตอร์ ดังนั้นสถานะจะสามารถเปลี่ยนแปลงได้ที่ขอบขาของ Clock เท่านั้น โดยจะแบ่งการออกแบบเป็น 2 แบบ

- แบบมัวร์เอาต์พุตนั้นจะเป็นฟังก์ชันของสถานะเพียงอย่างเดียว กล่าวคือเอาต์พุตเปลี่ยนแปลงตามจังหวะของ Clock เท่านั้น คือแต่ละสถานะที่ค่าของเอาต์พุตที่กำหนดแน่นอนเอาต์พุตจะเปลี่ยนก็ต่อเมื่อสถานะเปลี่ยน

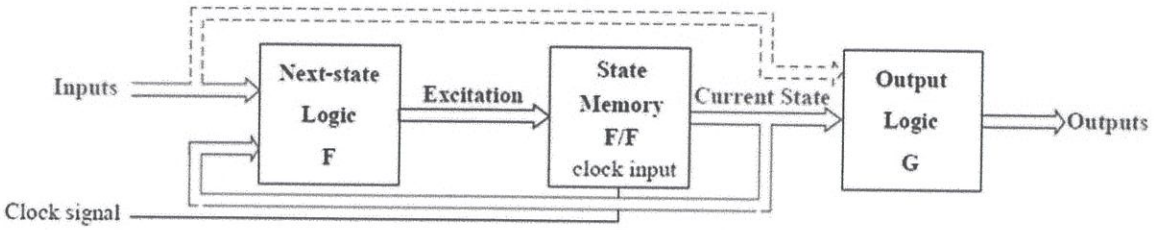
- แบบเมลลี่เอาต์พุตนั้นจะเป็นฟังก์ชันของสถานะ และอินพุตของเครื่องสถานะ กล่าวคือเอาต์พุตไม่จำเป็นต้องเปลี่ยนตามจังหวะของ Clock โดยเมื่ออินพุตเปลี่ยนเอาต์พุตจะเปลี่ยนทันที

โครงสร้างของเครื่องสถานะจำกัดโดยทั่วไปจะประกอบด้วย 3 ส่วนหลัก ดังรูปที่ 2.3

- **หน่วยความจำสถานะ (State Memory)** เป็นฟลิปฟล็อปสำหรับจดจำสถานะการทำงานของวงจรโดยฟลิปฟล็อป n ตัว ใช้เป็นตัวแปรสถานะได้ n ตัว

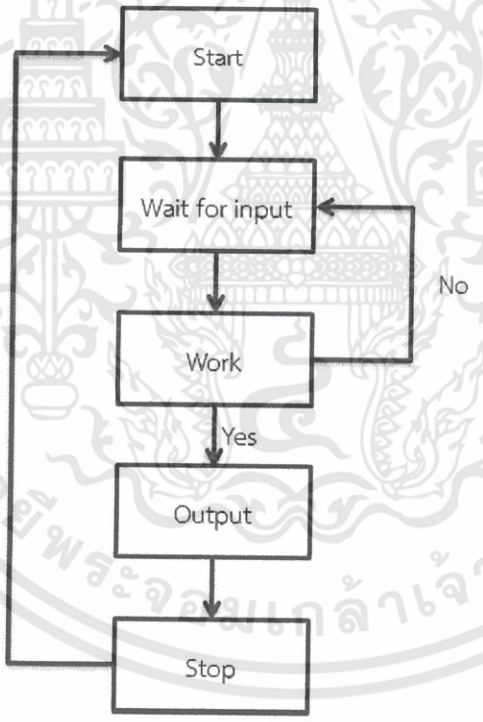
- **วงจรตรรกะกำหนดสถานะถัดไป (Next State Logic Circuit)** เป็นวงจรเชิงหมู่สร้างสัญญาณกระตุ้น (Excitation) ป้อนเข้าหน่วยความจำสถานะ เพื่อใช้กำหนดค่าของสถานะถัดไป โดยรับสัญญาณอินพุตจากตัวแปรอินพุตของเครื่องจักรนี้ และจากสถานะปัจจุบันที่ป้อนกลับมาจากเอาต์พุตของฟลิปฟล็อปที่ใช้เป็นหน่วยความจำสถานะ

- **วงจรตรรกะเอาต์พุต (Output Logic Circuit)** เป็นวงจรเชิงหมู่สำหรับสร้างสัญญาณเอาต์พุตของเครื่องสถานะจำกัด โดยอาจเป็นฟังก์ชันของสถานะ (แบบมัวร์) หรือเป็นทั้งฟังก์ชันของสถานะและอินพุตของเครื่องสถานะจำกัด (แบบเมลลี่)



รูปที่ 2.3 โครงสร้างของเครื่องสถานะจำกัดโดยทั่วไป

จากที่ได้กล่าวมาข้างต้น จะยกตัวอย่างของ State Machine ดังรูปที่ 2.4 สามารถอธิบายได้ว่า เมื่อไม่มีอินพุตเข้ามาระบบก็จะกลับไปเริ่มที่ Wait For Input แต่เมื่อมีอินพุตเข้ามาระบบก็จะเริ่มการทำงาน ระบบนี้จะหยุดทำงานเมื่อ Output นั้นมีค่าเท่ากับ Input ที่ใส่เข้าไป และระบบจะหยุดทำงานแล้วกลับมาจุดเริ่มต้นอีกครั้ง



รูปที่ 2.4 ตัวอย่าง State Machine

2.3 ความรู้เบื้องต้นเกี่ยวกับพร็อกซิมิตี้เซนเซอร์

2.3.1 พร็อกซิมิตี้เซนเซอร์

พร็อกซิมิตี้เซนเซอร์ (Proximity Sensor) หรือพร็อกซิมิตี้สวิตช์ (Proximity Switch) คือ เซนเซอร์ชนิดหนึ่งที่สามารถทำงานโดยไม่ต้องสัมผัสกับชิ้นงานหรือวัตถุภายนอก โดยลักษณะของการทำงานอาจจะส่งหรือรับพลังงานรูปแบบใดรูปแบบหนึ่งดังต่อไปนี้คือ สนามแม่เหล็ก สนามไฟฟ้า แสง เสียง และสัญญาณลม ส่วนการนำเซนเซอร์ประเภทนี้ไปใช้งานนั้น ส่วนใหญ่จะใช้กับงานตรวจจับตำแหน่ง ระดับ ขนาด และรูปร่าง ซึ่งโดยปกติแล้วจะนำมาใช้แทนลิมิตสวิตช์ (Limit Switch) เนื่องด้วยสาเหตุของอายุการใช้งานและความเร็วในการตรวจจับวัตถุเป้าหมาย ทำได้ดีกว่าอุปกรณ์ประเภทสวิตช์ซึ่งอาศัยหน้าสัมผัสทางกล

2.3.2 ประเภทของพร็อกซิมิตี้เซนเซอร์

2.3.2.1 เซนเซอร์แบบเหนี่ยวนำ (Inductive Sensor)

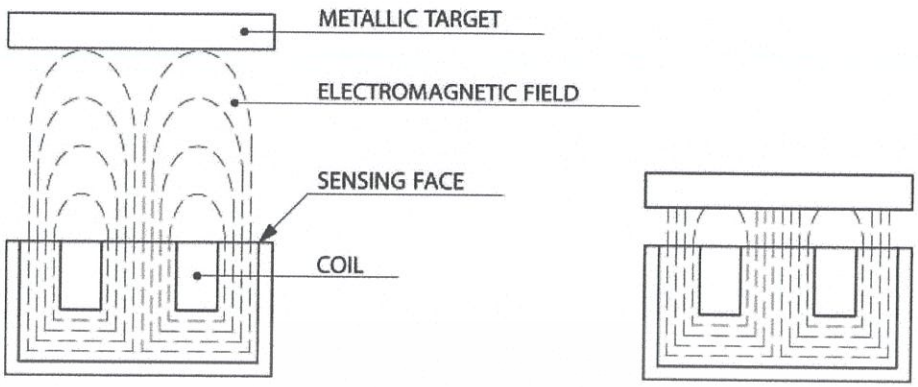
เป็นเซนเซอร์ที่ทำงานโดยอาศัยหลักการเปลี่ยนแปลงค่าความเหนี่ยวนำของขดลวด ซึ่งการเปลี่ยนแปลงดังกล่าวจะมีผลต่อชิ้นงานหรือวัตถุที่เป็นโลหะเท่านั้น หรือเรียกกันทางภาษาเทคนิคว่า “อินดั๊กตีฟเซนเซอร์” ข้อเด่นของเซนเซอร์ชนิดนี้ คือ ทนทานและสามารถทำงานได้ในช่วงอุณหภูมิที่กว้าง (Wide Temperature Ranges) สามารถทำงานในสภาวะที่มีการรบกวนทางแสง (Optical) และเสียง (Acoustic) ซึ่งเทียบเท่ากับชนิดเก็บประจุ

2.3.2.2 เซนเซอร์ชนิดเก็บประจุ (Capacitive Sensor)

เซนเซอร์ประเภทนี้มีโครงสร้างทั้งภายนอกและภายในคล้ายกับแบบเหนี่ยวนำ การเปลี่ยนแปลงของความจุ ซึ่งเนื่องมาจากการเคลื่อนที่ของวัตถุชนิดหนึ่งเข้ามาใกล้สนามไฟฟ้าของคาปาซิเตอร์ เซนเซอร์ชนิดนี้สามารถตรวจจับอุปกรณ์ที่ไม่ได้เป็นโลหะได้ และเป็นโลหะได้

2.3.3 หลักการทำงานของเซนเซอร์แบบเหนี่ยวนำ

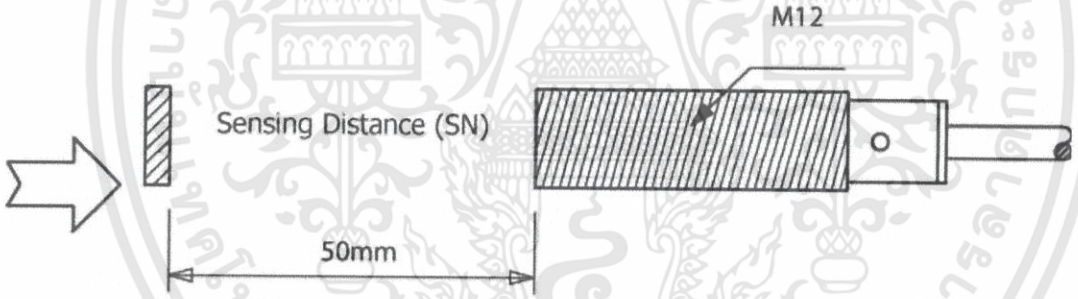
บริเวณส่วนหัวของเซนเซอร์จะมีสนามแม่เหล็กซึ่งมีความถี่สูง โดยได้รับสัญญาณมาจากวงจรกำเนิดความถี่ ในกรณีที่มีวัตถุหรือชิ้นงานที่เป็นโลหะเข้ามาอยู่ในบริเวณที่สนามแม่เหล็ก สามารถส่งไปถึงจะทำให้เกิดการเปลี่ยนแปลงค่าความเหนี่ยวนำ จากเหตุการณ์ที่เกิดขึ้นทำให้เกิดการหน่วงออสซิลเลท (Oscillate) ลดลงไป หรือบางทีอาจถึงจุดที่หยุดการออสซิลเลท และเมื่อนำเอาวัตถุนั้นออกจากบริเวณตรวจจับ วงจรกำเนิดคลื่นความถี่ก็เริ่มต้นการออสซิลเลทใหม่อีกครั้งหนึ่ง สภาวะดังกล่าวในข้างต้นจะถูกแยกแยะได้ด้วยวงจรอิเล็กทรอนิกส์ที่อยู่ภายใน หลังจากนั้นก็จะส่งผลไปยังเอาต์พุตว่าให้ทำงานหรือไม่ทำงาน โดยทั้งนี้จะขึ้นอยู่กับชนิดของเอาต์พุตว่าเป็นแบบดังรูปที่ 2.5



รูปที่ 2.5 การทำงานของเซนเซอร์แบบเหนี่ยวนำ

2.3.3.1 Sensing Distance (SN)

ระยะที่ตัวเซนเซอร์สามารถตรวจวัตถุได้ซึ่งจะขึ้นอยู่กับชนิด ขนาดของวัตถุ และเส้นผ่าศูนย์กลางของเซนเซอร์ ซึ่งโดยปกติแล้ว ถ้าเส้นผ่าศูนย์การของตัวเซนเซอร์ใหญ่ก็ยิ่งทำให้ระยะการตรวจจับได้ไกลดังรูปที่ 2.6



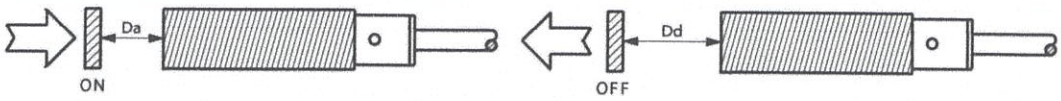
รูปที่ 2.6 ระยะการตรวจจับวัตถุ Sensing Distance

2.3.3.2 Target Material Factor

เป็นค่า Factor โดยประมาณของวัตถุแต่ละชนิด ใช้สัมประสิทธิ์คูณกับค่า Sensing Distance เพื่อให้ได้ค่าระยะการตรวจจับที่แน่นอนยิ่งขึ้น เมื่อใช้ Inductive Sensor ในการตรวจจับวัตถุชนิดนั้นๆ

2.3.3.3 Hysteresis

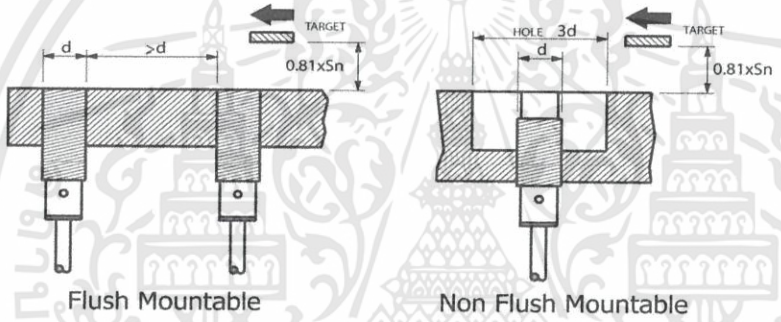
เป็นช่วงหรือย่านที่ตัว Sensor จะให้สถานะของ Output เป็น On หรือ Off ซึ่งโดยปกติแล้วในการออกแบบเครื่องจักรต่างๆ ต้องคำนึงถึงค่านี้ด้วยเพื่อให้มั่นใจได้ว่าตัว Sensor ของเราที่ติดตั้งไปแล้วนั้นจะสามารถทำงานได้อย่างถูกต้องและแน่นอนตลอดเวลาดังรูปที่ 2.7



รูปที่ 2.7 Hysteresis

2.3.3.4 Mountable

เป็นรูปแบบในการติดตั้งตัวเซนเซอร์ ซึ่งโดยปกติแล้วตัวเซนเซอร์ทั้ง Inductive และ Capacitive จะมีรูปแบบในการติดตั้งอยู่ 2 ชนิด คือแบบ Flush Mount และ Non Flush Mount โดยมีลักษณะในการติดตั้งที่แตกต่างกันดังรูปที่ 2.8 ถ้ามีการติดตั้งที่ผิดวิธีก็อาจจะทำให้การทำงานของตัวเซนเซอร์ผิดพลาดได้



รูปที่ 2.8 Mountable

2.4 ความรู้เบื้องต้นเกี่ยวกับมอเตอร์

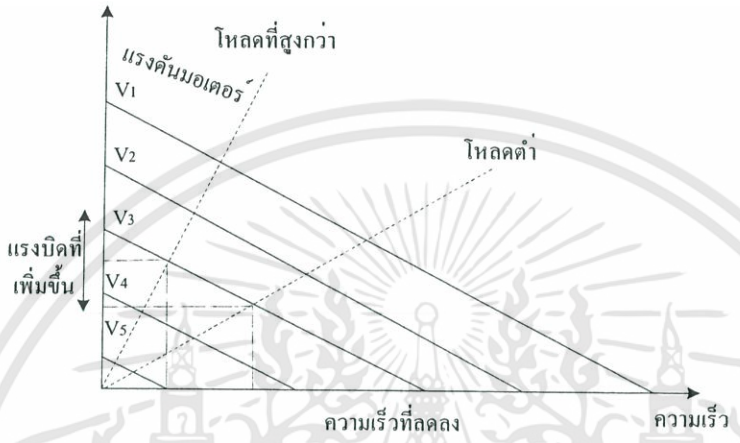
2.4.1 หลักการทำงานของมอเตอร์กระแสตรง

เมื่อมีการผ่านกระแสไฟฟ้าเข้าไปยังขดลวดในสนามแม่เหล็ก จะทำให้เกิดแรงแม่เหล็กซึ่งมีสัดส่วนของแรงขึ้นกับกระแสแรงของสนามแม่เหล็ก โดยแรงจะเกิดขึ้นเป็นมุมฉากกับกระแสและสนามแม่เหล็ก ขณะที่ทิศทางของแรงกลับตรงกันข้ามกัน ถ้าหากกระแสของสนามแม่เหล็กไหลย้อนกลับจะทำให้เกิดการเปลี่ยนแปลงของกระแส และสนามแม่เหล็กเป็นผลทำให้ทิศทางของแรงเปลี่ยนไป ด้วยคุณสมบัตินี้ทำให้มอเตอร์กระแสตรงกลับทิศทางการหมุนได้

สนามแม่เหล็กของมอเตอร์ส่วนหนึ่งเกิดขึ้นจากแม่เหล็กถาวร ซึ่งจะถูกยึดติดกับแผ่นเหล็กหรือเหล็กกล้า โดยปกติส่วนนี้จะเป็นส่วนที่ยึดอยู่กับที่ และขดลวดเหนี่ยวนำจะพันอยู่กับส่วนที่เป็นแกนหมุนของมอเตอร์

2.4.2การควบคุมด้วยวิธีเปลี่ยนค่าแรงดัน

วิธีการนี้ต้องใช้อุปกรณ์อิเล็กทรอนิคส์ที่มีอัตราขยายกำลังสูง และมอเตอร์จะถูกป้อนด้วยแรงดันที่เปลี่ยนแปลงค่าได้ จากแหล่งจ่ายที่มีอิมพีแดนซ์ต่ำ ข้อดีของการควบคุมวิธีนี้คือ ถ้าความเร็วลดลงจากผลของแรงบิด แรงดันที่ป้อนให้กับมอเตอร์จะเพิ่มขึ้นเพื่อรักษาระดับความเร็วดังรูปที่ 2.9 ส่วนข้อเสียจากการควบคุมวิธีนี้คือ เมื่อมอเตอร์มีความเร็วต่ำแรงดันที่ป้อนให้กับมอเตอร์จะมีค่าต่ำเช่นกัน



รูปที่ 2.9กราฟการควบคุมด้วยวิธีเปลี่ยนค่าแรงดัน

2.4.3 การควบคุมด้วยตัวต้านทานที่ปรับค่าได้

การควบคุมแบบนี้สามารถขับดีซีมอเตอร์ได้ความเร็ว 10 ต่อ 1 และให้การเร็คกูเลทที่ดีกว่ากระแสถูกปล่อยให้ฟิลด์คงที่ ผลของคุณสมบัติความเร็วและแรงบิดได้รับการปรับปรุงดีขึ้นกว่าการบังคับด้วยความต้านทานที่ปรับค่าได้ และให้การเร็คกูเลทความเร็วคงที่ได้ดีขึ้นตลอดช่วงความเร็วที่กว้างกว่า

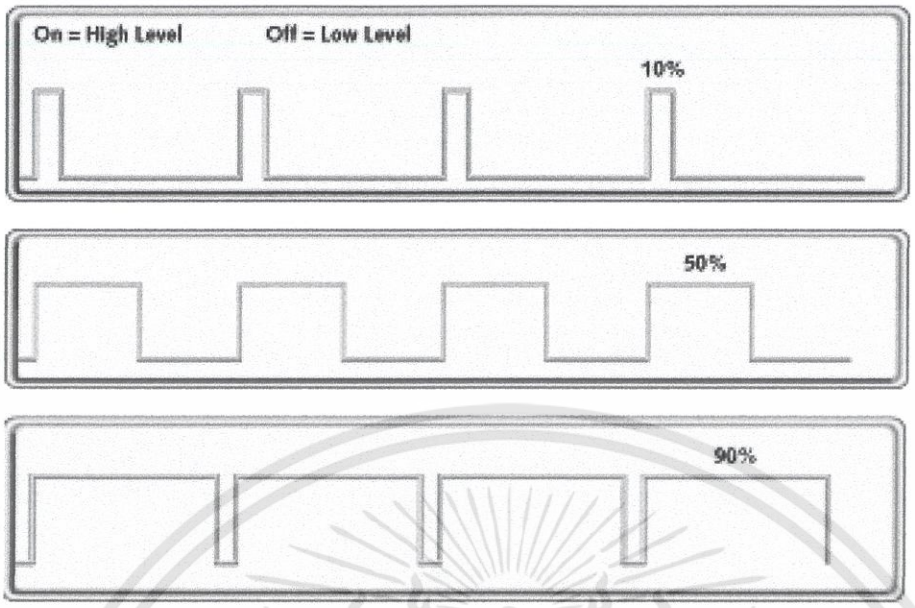
2.4.4 การควบคุมแบบ PWM (Pulse Width Modulation)

Pulse Width Modulation (PWM) คือ เทคนิคสำหรับควบคุมวงจรทางด้านฮาร์ดแวร์โดยใช้สัญญาณเอาต์พุตแบบดิจิตอลของไมโครโปรเซสเซอร์ควบคุมการทำงานของสัญญาณ PWM

รูปที่ 2.10 แสดงสัญญาณ PWM ที่แตกต่างกัน 3 สัญญาณ

- โดยรูป ก แสดงสัญญาณ PWM ที่ 10% Duty Cycle คือสัญญาณในการออนจะเป็น 10% ของคาบสัญญาณ และจะออฟเป็น 90% ของคาบสัญญาณ
- โดยรูป ข แสดงสัญญาณ PWM ที่ 50% Duty Cycle คือ สัญญาณในการออนจะเป็น 10% ของคาบสัญญาณ และจะออฟเป็น 50% ของคาบสัญญาณ
- โดยรูป ค แสดงสัญญาณ PWM ที่ 90% Duty Cycle คือ สัญญาณในการออนจะเป็น 10% ของคาบสัญญาณ และจะออฟเป็น 10% ของคาบสัญญาณ

เช่น ถ้า Power Supply มี 12V และ Duty Cycle เป็น 10% จะได้เอาต์พุต 1.2V



รูปที่ 2.10 แสดงสัญญาณ PWM ซึ่งแสดงค่า Duty Cycles ที่ต่างๆ กัน

2.5 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์ คือ อุปกรณ์ควบคุมขนาดเล็ก ซึ่งบรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ โดยในไมโครคอนโทรลเลอร์ได้รวมเอาซีพียู หน่วยความจำ และพอร์ต ซึ่งเป็นส่วนประกอบหลักสำคัญของระบบคอมพิวเตอร์เข้าไว้ด้วยกัน โดยทำการบรรจุเข้าไว้ในตัวถังเดียวกัน

2.5.1 โครงสร้างทั่วไป

โครงสร้างโดยทั่วไปของไมโครคอนโทรลเลอร์นั้น สามารถแบ่งออกมาได้เป็น 5 ส่วนใหญ่ ๆ ดังต่อไปนี้

2.5.1.1 หน่วยประมวลผลกลางหรือซีพียู (CPU : Central Processing Unit)

ทำหน้าที่เป็นศูนย์กลางควบคุมการทำงานของระบบคอมพิวเตอร์ทั้งหมด โดยนำข้อมูลจากอุปกรณ์รับข้อมูลมาทำงาน ประมวลผลข้อมูลตามคำสั่งของโปรแกรม และส่งผลลัพธ์ออกไปหน่วยแสดงผล

2.5.1.2 หน่วยความจำ (Memory)

สามารถแบ่งออกเป็นส่วนคือ หน่วยความจำที่มีไว้สำหรับเก็บโปรแกรมหลัก (Program Memory) เปรียบเสมือนฮาร์ดดิสก์ของเครื่องคอมพิวเตอร์ตั้งโต๊ะ คือข้อมูลใดๆ ที่ถูกเก็บไว้ในนี้จะไม่สูญหายไปแม้ไม่มีไฟเลี้ยง อีกส่วนหนึ่งคือหน่วยความจำข้อมูล (Data Memory) ใช้เป็นเหมือนกระดาษทดในการคำนวณของซีพียู และเป็นที่พักข้อมูลชั่วคราวขณะทำงาน แต่หากไม่มีไฟเลี้ยงข้อมูลก็จะหายไปคล้ายกับหน่วยความแรม (Ram) ในเครื่องคอมพิวเตอร์ทั่วๆ ไป แต่สำหรับไมโครคอนโทรลเลอร์สมัยใหม่ หน่วยความจำข้อมูลจะมีทั้งที่เป็นหน่วยความจำแรม ซึ่งข้อมูลจะ

หายไปเมื่อไม่มีไฟเลี้ยง และเป็นอีอีพรอม (EEPROM : Erasable Electrically Read-Only Memory) ซึ่งสามารถเก็บข้อมูลได้แม้ไม่มีไฟเลี้ยง

2.5.1.3 ส่วนติดต่อกับอุปกรณ์ภายนอกหรือพอร์ต

พอร์ต (Port) มี 2 ลักษณะคือ พอร์ตอินพุต (Input Port) และพอร์ตส่งสัญญาณหรือพอร์ตเอาต์พุต (Output Port) ส่วนนี้จะใช้ในการเชื่อมต่อกับอุปกรณ์ภายนอก ถือว่าเป็นส่วนที่สำคัญมาก ใช้ร่วมกันระหว่างพอร์ตอินพุต เพื่อรับสัญญาณอาจจะด้วยการกดสวิตช์ เพื่อนำไปประมวลผลและส่งไปพอร์ตเอาต์พุตเพื่อแสดงผลเช่น การติดสว่างของหลอดไฟ เป็นต้น

2.5.1.4 ช่องทางเดินของสัญญาณ

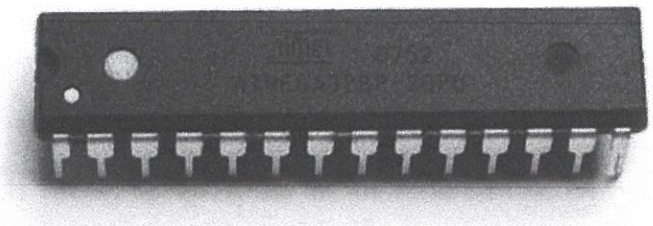
บัส (Bus) คือเส้นทางการแลกเปลี่ยนสัญญาณข้อมูลระหว่าง ซีพียู หน่วยความจำและพอร์ต เป็นลักษณะของสายสัญญาณ จำนวนมากอยู่ภายในตัวไมโครคอนโทรลเลอร์ โดยแบ่งเป็นบัสข้อมูล (Data Bus), บัสแอดเดรส (Address Bus) และบัสควบคุม (Control Bus)

2.5.1.5 วงจรกำเนิดสัญญาณนาฬิกา

นับเป็นส่วนประกอบที่สำคัญมากอีกส่วนหนึ่ง เนื่องจากการทำงานที่เกิดขึ้นในตัวไมโครคอนโทรลเลอร์ จะขึ้นอยู่กับการกำหนดจังหวะ หากสัญญาณนาฬิกามีความถี่สูง จังหวะการทำงานก็จะสามารถทำได้ถี่ขึ้น ส่งผลให้ไมโครคอนโทรลเลอร์ตัวนั้นมีความเร็วในการประมวลผลสูงตามไปด้วย

2.5.2 ไมโครคอนโทรลเลอร์ AVR

ไมโครคอนโทรลเลอร์ AVR เป็นไอซีไมโครคอนโทรลเลอร์ของบริษัท Atmel มีสถาปัตยกรรมภายในเป็นแบบ RISC (Reduced Instruction Set Computer) โดยใช้สัญญาณนาฬิกาเพียง 1 ลูกในการปฏิบัติงานใน 1 คำสั่ง (Instructions in a Single Clock Cycle) เป็นไมโครคอนโทรลเลอร์ที่มีประสิทธิภาพสูง แบ่งได้หลายอนุกรม และในแต่ละอนุกรมยังแบ่งออกได้หลายเบอร์เพื่อรองรับความต้องการที่แตกต่างของผู้ใช้งาน โดยในโครงงานนี้ได้เลือกใช้ AVR เบอร์ ATmega328 ซึ่งไมโครคอนโทรลเลอร์ชนิดนี้สามารถใช้ภาษาซีในการเขียนโปรแกรมได้



รูปที่ 2.11 AVR เบอร์ ATmega328

2.5.2.1 คุณสมบัติไมโครคอนโทรลเลอร์ AVR ATMEGA328

- ไมโครคอนโทรลเลอร์ขนาด 8 บิต ประสิทธิภาพสูง แต่ใช้พลังงานต่ำในตระกูล AVR
- ซีพียู (CPU) เป็นสถาปัตยกรรมแบบ RISC
- มีชุดคำสั่งใช้งาน 131 คำสั่ง และส่วนใหญ่คำสั่งจะใช้สัญญาณนาฬิกาเพียง 1 ลูกในการประมวลผล
- มีรีจิสเตอร์สำหรับใช้งานทั่วไปขนาด 8 บิต จำนวน 32 ตัว
- ทำงานได้สูงสุดที่ 20 ล้านคำสั่งต่อวินาที เมื่อใช้สัญญาณนาฬิกา 20 เมกะเฮิร์ซ (MHz)

2.5.2.2 หน่วยความจำ

- หน่วยความจำ Rom แบบ Flash ขนาด 32 กิโลไบต์ (KB)
- หน่วยความจำข้อมูลแบบ EEPROM (มีโหมดป้องกันหน่วยความจำ) ขนาด 1 กิโลไบต์
- หน่วยความจำข้อมูลแบบ SRAM ขนาด 2 กิโลไบต์
- สามารถเขียนและลบได้ถึง 100,000 ครั้ง
- เก็บข้อมูลได้กว่า 20 ปี ที่อุณหภูมิ 85 องศาเซลเซียส และกว่า 100 ปีที่อุณหภูมิ 25 องศาเซลเซียส

2.5.2.3 คุณสมบัติการเชื่อมต่อกับอุปกรณ์ภายนอก

- มีตัวตั้งเวลาและตัวนับขนาด 8 บิต จำนวน 2 ตัวที่สามารถแยกโหมดการทำงานจากกันได้ 2 โหมดคือ Prescaler และ Compare
- มีตัวตั้งเวลาและตัวนับขนาด 16 บิต จำนวน 1 ตัว ที่แยกโหมดการทำงานได้ 3 โหมดคือ Prescaler Compare และ Capture
- มีตัวนับเวลาจริง (Real Time Counter) ที่แยกวงจรกำหนดความถี่ได้
- โมดูลสร้างสัญญาณ PWM (Pulse Width Modulator) มีจำนวน 6 ช่องสัญญาณ
- โมดูลแปลงสัญญาณแอนะล็อกเป็นดิจิตอลขนาด 10 บิต
- โมดูลเชื่อมต่ออุปกรณ์อนุกรมแบบ SPI ได้ทั้งการเป็นมาสเตอร์ (Master) และสเลฟ (Slave)

- โมดูลเชื่อมต่ออุปกรณ์อนุกรมแบบลักษณะเป็นสายสัญญาณ 2 เส้น แบบส่งข้อมูลเรียงไบต์ (Byte Oriented)

- โมดูลเปรียบเทียบแรงดันแอนะล็อก (Analog Comparator)

- มีตัวตั้งเวลาแบบวอตช์ด็อกที่สามารถกำหนดการทำงานได้ โดยสามารถแยกสัญญาณนาฬิกาได้จากตัวชิพ

- การรองรับการอินเตอร์รัปต์และการเวก-อัพ (Wake Up) เมื่อมีการเปลี่ยนแปลงเกิดขึ้นกับขาของชิพ

2.5.2.4 คุณสมบัติพิเศษ

- มีระบบการรีเซต (บราวน์เอาต์ดีเท็กชั่น : Brown-Out Detection) เมื่อตรวจพบการทำงานผิดปกติ

- มีการตอบสนองแหล่งกำเนิดอินเตอร์รัปต์ทั้งภายในและภายนอก (External and Internal Interrupt sources)

- มีโหมดการทำงานสลีป 6 แบบคือ Idle, ADC Noise Reduction, Power-Save, Power-Down, Standby และ Extended Standby

2.5.2.5 I/O และตัวถัง

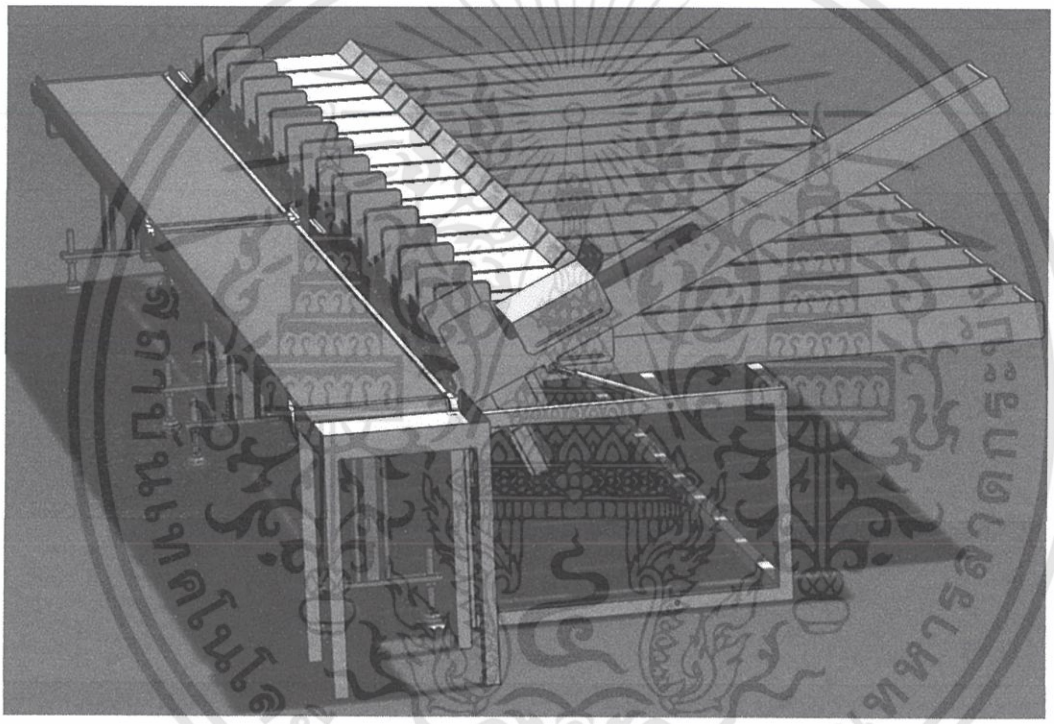
- มีขาของ I/O ที่สามารถกำหนดการทำงานได้ 23 ขา

- 28-pin PDIP, 32-lead TQFP, 28-pad QFN/MLF and 32-pad QFN/MLF

บทที่ 3
การคำนวณและการสร้าง

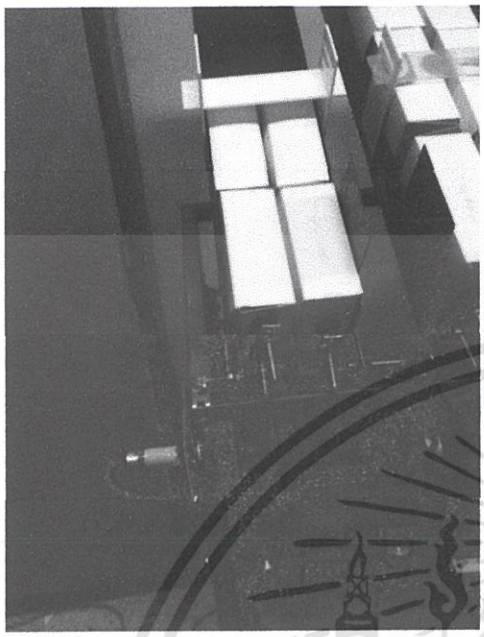
3.1 การออกแบบเครื่องต้นแบบ

โครงงานนี้ในการออกแบบด้านฮาร์ดแวร์จะออกแบบจากโปรแกรม SolidWorks โดยเราคำนึงถึงการไหลสินค้าโดยอัตโนมัติ และแยกชนิดของสินค้าได้อย่างเป็นระเบียบและหมวดหมู่ ซึ่งมีหลังคาสำหรับป้องกันการตกของสินค้า ดังนั้นเราจึงออกแบบลักษณะโครงสร้างดังรูปที่ 3.1

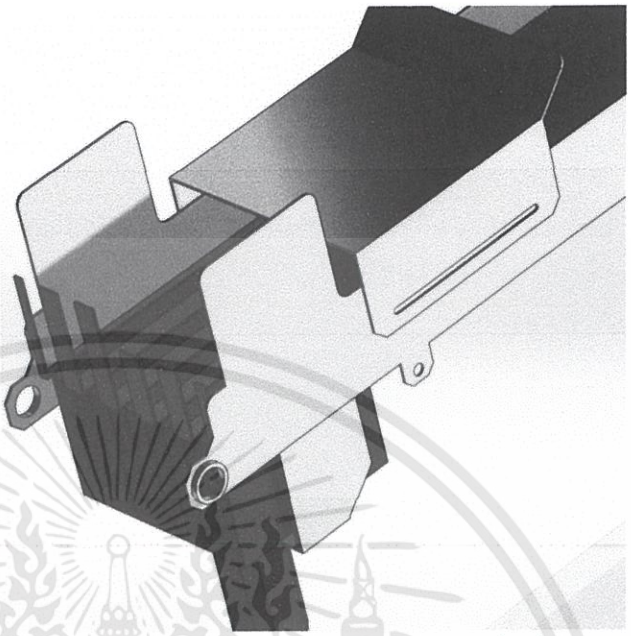


รูปที่ 3.1 ลักษณะโครงสร้างการปล่อยสินค้า

สินค้าที่อยู่ภายในรางเก็บสินค้าจะมีลักษณะเป็นกล่องสี่เหลี่ยม โดยบรรจุได้สองแถวต่อหนึ่งราง ซึ่งโครงงานนี้จะออกแบบการปล่อยสินค้าอยู่สามระดับ โดยระดับแรกคือ ระดับเริ่มต้นที่ยังไม่มีการกดสั่งชนิดสินค้าดังรูปที่ 3.2(ก) เป็นโครงสร้างตัวต้นแบบและรูปที่ 3.2(ข) เป็นการออกแบบด้วยโปรแกรม SolidWorks ระดับที่สองคือ ระดับที่ได้รับคำสั่งการปล่อยสินค้าชิ้นที่หนึ่งดังรูปที่ 3.3(ก) เป็นโครงสร้างตัวต้นแบบและรูปที่ 3.3(ข) เป็นการออกแบบด้วยโปรแกรม SolidWorks ซึ่งในระดับนี้เป็นการปล่อยสินค้าชิ้นที่ 1 โดยสินค้าที่อยู่ฝั่งเหล็กกั้นด้านต่ำจะถูกกระบอกสูบไฟฟ้าดันตกลงมาเป็นชิ้นแรก ระดับสามคือ ระดับที่ได้รับคำสั่งการปล่อยสินค้าชิ้นที่สองดังรูปที่ 3.4(ก) เป็นโครงสร้างตัวต้นแบบและรูปที่ 3.3(ข) เป็นการออกแบบด้วยโปรแกรม SolidWorks ซึ่งในระดับนี้เป็นการปล่อยสินค้าชิ้นที่ 2 โดยสินค้าที่อยู่ฝั่งเหล็กกั้นด้านสูงจะถูกกระบอกไฟฟ้าดันตกลงมาเป็นชิ้นที่ 2



รูปที่ 3.2(ก) การปล่อยสินค้าระดับที่หนึ่ง

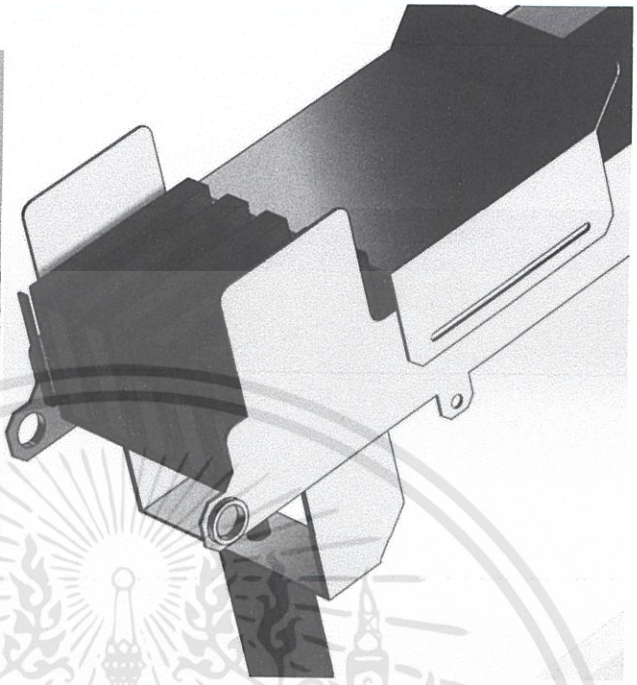


รูปที่ 3.2(ข) การปล่อยสินค้าระดับที่หนึ่ง

รูปที่ 3.3(ก) การปล่อยสินค้าระดับที่สอง

รูปที่ 3.3(ข) การปล่อยสินค้าระดับที่สอง

รูปที่ 3.4(ก) การปล่อยสินค้าระดับที่สาม



รูปที่ 3.4(ข) การปล่อยสินค้าระดับที่สาม

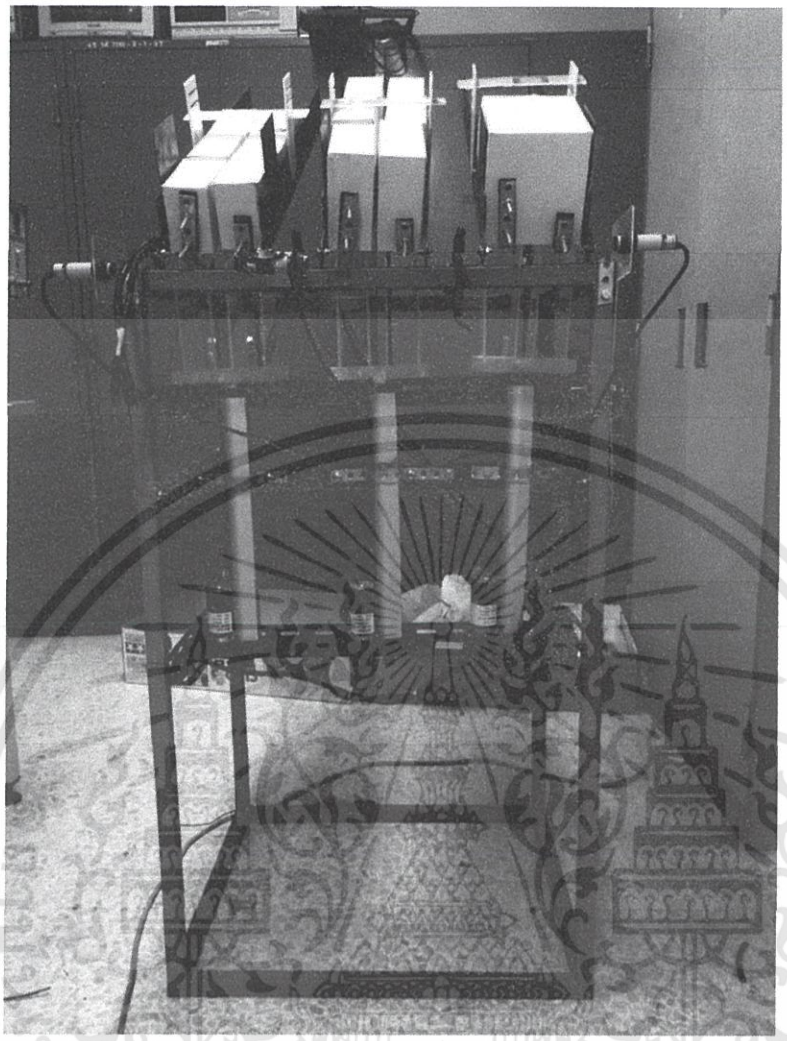
3.2 การสร้างเครื่องต้นแบบ

3.2.1 โครงสร้าง

ทำมาจากเหล็ก โดยมีสัดส่วน 60x50x100 cm. เนื่องจากโครงงานนี้ต้องการให้มีความแข็งแรง จึงเลือกใช้วัสดุที่ทำจากเหล็กโดยสามารถเจาะเพื่อเพิ่มโครงสร้างอื่นๆ ได้อย่างอิสระ

3.2.2 รางจัดเก็บสินค้า

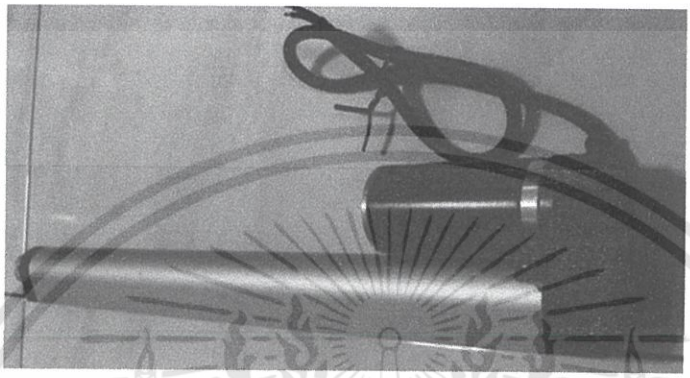
ทำมาจากเหล็ก โดยมีสัดส่วน 70x13x8 cm. มีลักษณะจัดเก็บสินค้าได้ 2 แถว การออกแบบโครงสร้างในโครงงานนี้จะเป็นตัวต้นแบบซึ่งออกแบบรางจัดเก็บออกเป็น 3 รางจัดเก็บสินค้าได้ 3 ชนิดดังรูปที่ 3.5



รูปที่ 3.5 โครงสร้างตัวปล่อยสินค้า

3.2.3 กระบอกสูบไฟฟ้า

ในการผลักสินค้าให้ตกลงจากรางสินค้านั้นต้องใช้มอเตอร์ที่สามารถดันขึ้น-ลง และสามารถจ่ายไฟกระแสตรงได้ จึงเลือกใช้กระบอกสูบไฟฟ้า12Vdc โดยจะใช้การจ่ายไฟให้กับขั้วบวก-ลบ ของมอเตอร์เป็นการควบคุมให้กระบอกสูบไฟฟ้าเคลื่อนที่ขึ้น-ลง และจะมีความเร็วคงที่ 42 mm/s



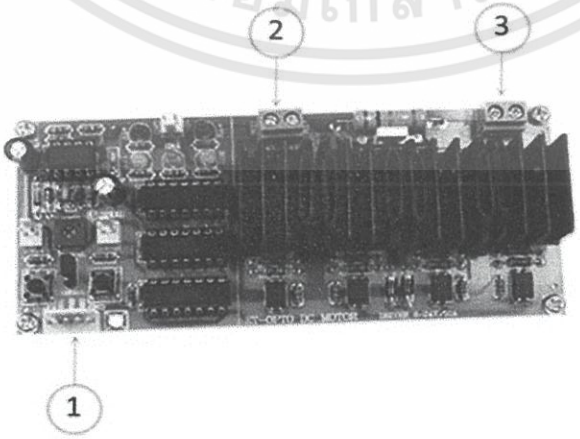
รูปที่ 3.6 กระบอกสูบไฟฟ้า

3.2.4 ET-OPTO-DC MOTOR

โครงงานนี้จะทำการสั่งดีซีมอเตอร์ให้ทำงาน ดังนั้นจึงจะต้องเลือกใช้ ET-OPTO-DC MOTOR เนื่องจากอุปกรณ์นี้สามารถเป็นบอร์ดใช้งานอิสระ หรือต่อเข้ากับบอร์ดไมโครคอนโทรลเลอร์ต่างๆ ใช้ควบคุมการทำงานของตัวดีซีมอเตอร์ให้หมุน ซ้าย, ขวา และควบคุมความเร็วของตัวดีซีมอเตอร์ออกแบบใช้กับดีซีมอเตอร์กระแสสูงๆ ได้

3.2.4.1 รายละเอียดที่ใช้ของ ET-OPTO-DC MOTOR

- หมายเลข 1 มีขั้ว 5 pin สำหรับต่อกับไมโครคอนโทรลเลอร์ เพื่อใช้ส่งสัญญาณจากภายนอกมาควบคุมความเร็วการหมุนของดีซีมอเตอร์ ได้โดยตรงและเป็นไฟเลี้ยง 5Vdc ให้กับไอซี หมายเลข 2 แหล่งจ่ายไฟ 12Vdc สำหรับเลี้ยงดีซีมอเตอร์โดยตรง หมายเลข 3 พอร์ตต่อเข้ากับขั้วของดีซีมอเตอร์



รูปที่ 3.7 ET-OPTO-DC MOTOR

3.2.5 พร็อกซิมิตี้เซนเซอร์

เซนเซอร์นี้ทำงานโดยอาศัยหลักการเปลี่ยนแปลงค่าความเหนี่ยวนำของขดลวด สามารถทำงานโดยไม่ต้องสัมผัสกับชิ้นงานหรือวัตถุภายนอกซึ่งมีระยะตรวจจับอยู่ที่ 0 – 40 เซนติเมตร โดยโครงงานนี้จะใช้ระยะตรวจจับไกลสุดอยู่ที่ 30 เซนติเมตร ถ้าเซนเซอร์ตรวจจับวัตถุได้จะให้แรงดันเอาต์พุต 3Vdc

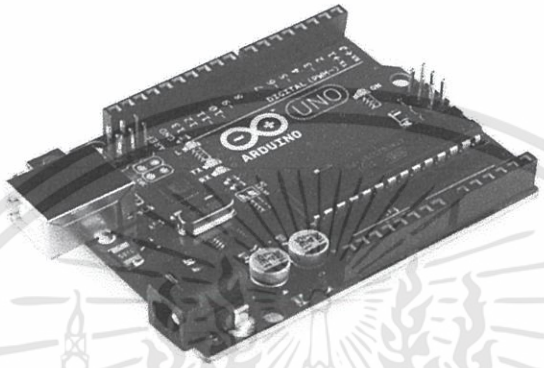
ตารางที่ 3.1 แสดงผลของการตรวจจับวัตถุที่ระยะ 30 เซนติเมตร

หมายเลขเซนเซอร์ / ระยะวัตถุ(cm)	1	2	3
0	3Vdc	3Vdc	3Vdc
10	3Vdc	3Vdc	3Vdc
20	3Vdc	3Vdc	3Vdc
30	3Vdc	3Vdc	3Vdc
40	0Vdc	0Vdc	0Vdc

รูปที่ 3.8 พร็อกซิมิตี้เซนเซอร์

3.2.6 ไมโครคอนโทรลเลอร์

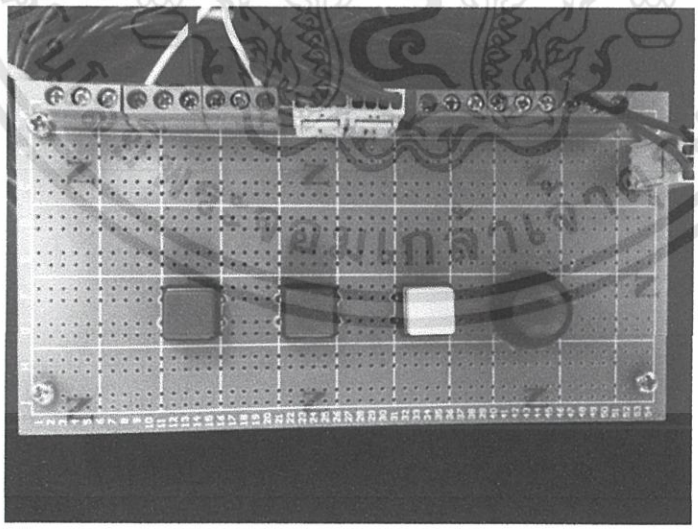
ในการส่งข้อมูลจะต้องมีอุปกรณ์ที่คอยจัดเก็บข้อมูลที่ส่งผ่านจากรีโมทคอนโทรล เพื่อทำการประมวลผลสั่งให้มอเตอร์และเซนเซอร์ทำงาน สำหรับโครงงานนี้เลือกใช้บอร์ด Arduino Uno R3 ซึ่งมีไมโครคอนโทรลเลอร์ ATmage328 เป็นตัวประมวลผล



รูปที่ 3.9 Arduino Uno R3

3.2.7 รีโมทคอนโทรล

ในการสั่งสินค้าจึงจำเป็นต้องมีชุดรีโมทคอนโทรลในการป้อนคำสั่งให้กับบอร์ด Arduino Uno R3 สำหรับโครงงานนี้ใช้ชุดรีโมทคอนโทรลที่ได้สร้างขึ้นมาเองดังรูปที่ 3.10

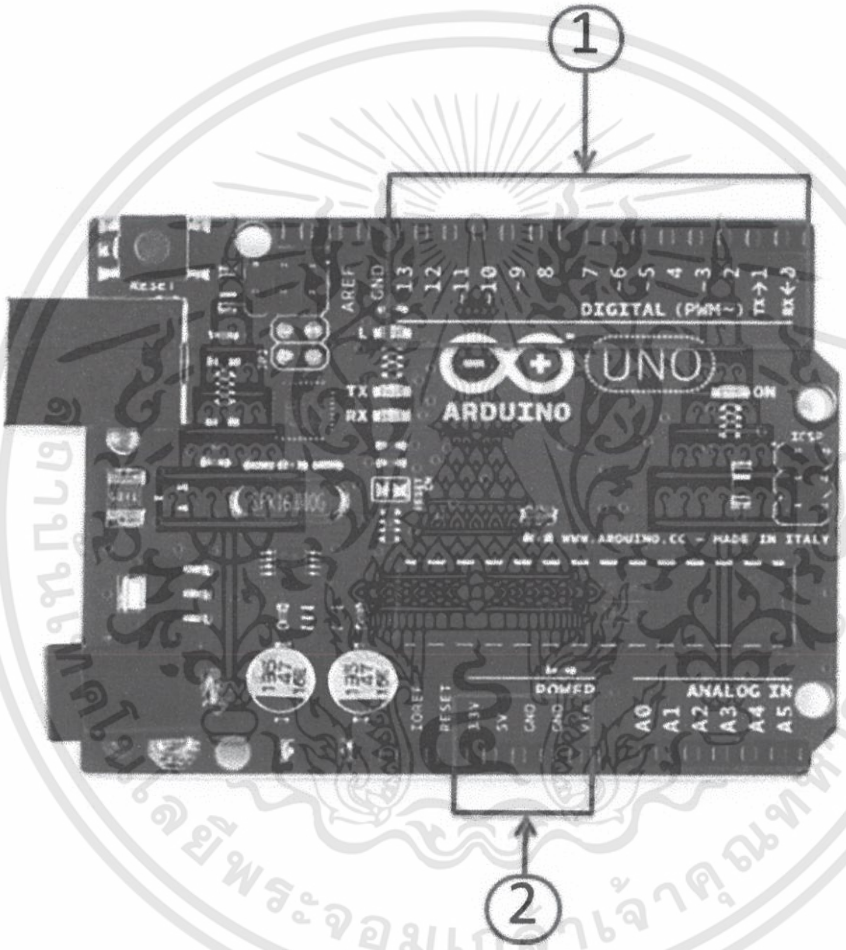


รูปที่ 3.10 รีโมทคอนโทรล

3.2.8 การเชื่อมต่อไมโครคอนโทรลเลอร์, รีโมทคอนโทรล และ ET-OPTO-DC MOTOR

ในโครงงานนี้การต่อวงจรจะประกอบไปด้วยอินพุต ตัวประมวลผลและเอาต์พุตโดยตัวประมวลผลคือ ตัวส่งสัญญาณควบคุมในรูปแบบดิจิตอล

- หมายเลข 1 เป็น I/O ดิจิตอลพอร์ตทำหน้าที่รับส่งสัญญาณกับรีโมทคอนโทรลโดยจะมีพอร์ต D0-D13 ที่เป็น I/O
- หมายเลข 2 ทำหน้าที่จ่าย 5Vdc ให้กับรีโมทคอนโทรล

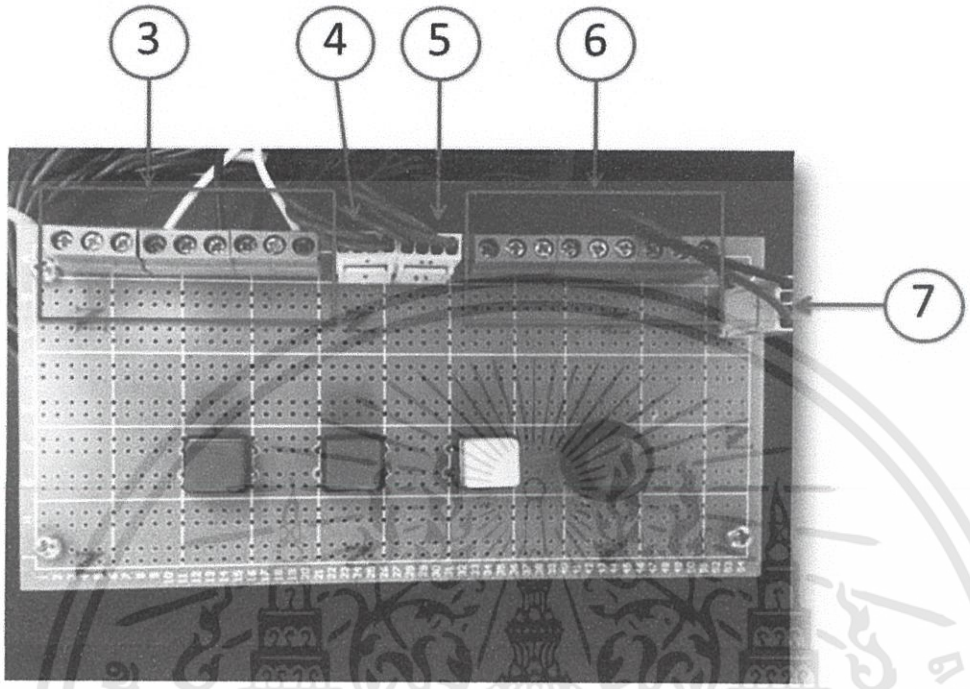


รูปที่ 3.11 ไมโครคอนโทรลเลอร์

อินพุตคือ การป้อนข้อมูลบนรีโมทคอนโทรลไปยังบอร์ดไมโครคอนโทรลเลอร์ดังรูปที่ 3.11

- หมายเลข 3 เป็นตัวรับสัญญาณควบคุมของกระบอกสูบไฟฟ้าจากบอร์ดไมโครคอนโทรลเลอร์และส่งสัญญาณไปยัง ET-OPTO-DC MOTOR
- หมายเลข 4 เป็นตัวส่งสัญญาณเอาต์พุตของเซนเซอร์ส่งไปยังบอร์ดไมโครคอนโทรลเลอร์ โดยมีแรงดันไฟฟ้า 3Vdc
- หมายเลข 5 เป็นตัวส่งสัญญาณของรีโมทคอนโทรลไปยังบอร์ดไมโครคอนโทรลเลอร์
- หมายเลข 6 เป็นตัวจ่ายไฟให้กับเซนเซอร์และรับสัญญาณเอาต์พุตจากเซนเซอร์

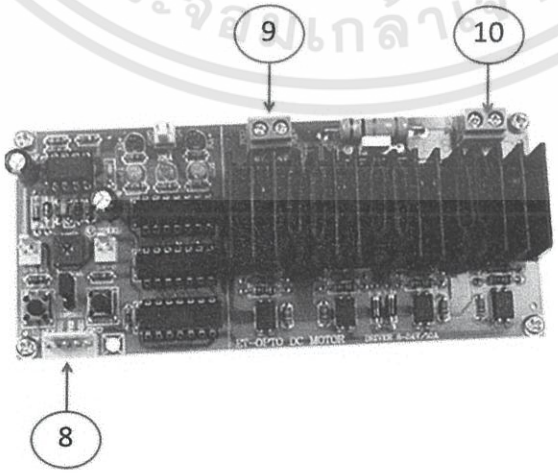
- หมายเลข 7 เป็นตัวรับสัญญาณไฟ 5Vdc จากบอร์ดไมโครคอนโทรลเลอร์



รูปที่ 3.12 รีโมทคอนโทรล

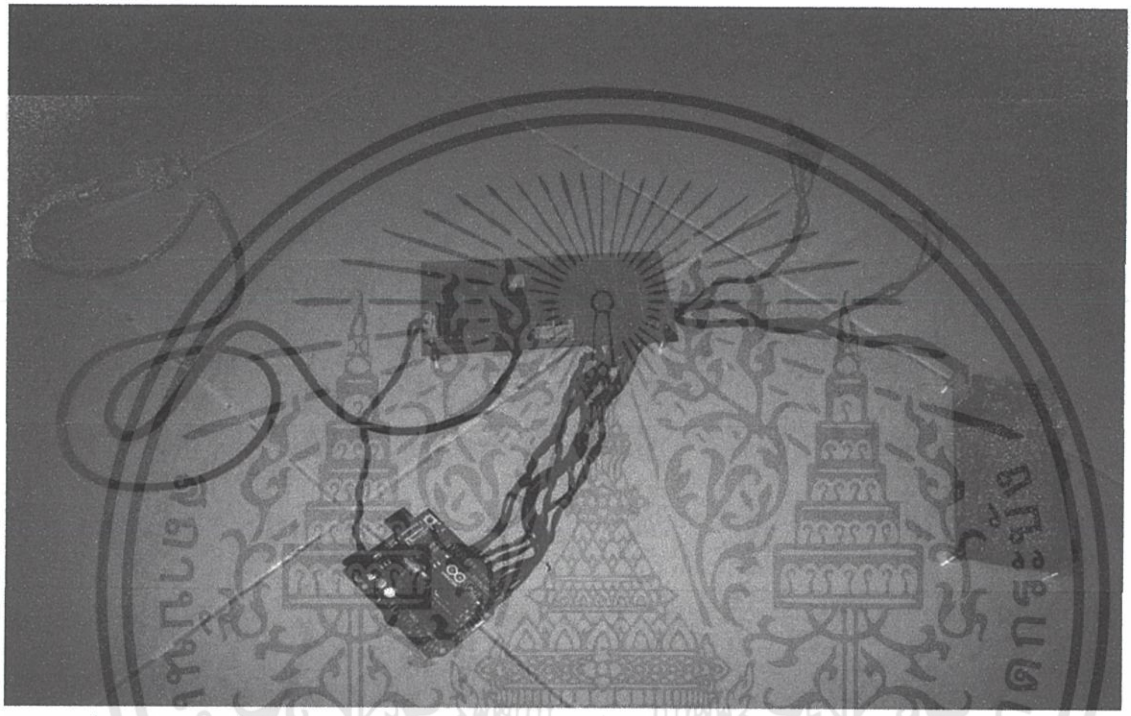
เอาต์พุตคือ การส่งสัญญาณจากรีโมทคอนโทรลไปยัง ET-OPTO-DC MOTOR เพื่อสั่งให้กระบอกสูบไฟฟ้าทำงานตามการประมวลผลดังรูปที่ 3.13

- หมายเลข 8 เป็นตัวรับสัญญาณควบคุมจากรีโมทคอนโทรลโดยจะจ่ายไฟ 5Vdc ให้กับ IC
- หมายเลข 9 เป็นตัวรับสัญญาณไฟจากภายนอกเพื่อจ่ายให้กับกระบอกสูบไฟฟ้า โดยใช้แรงดันไฟฟ้า 12Vdc
- หมายเลข 10 เป็นตัวจ่ายไฟให้กับกระบอกสูบไฟฟ้า



รูปที่ 3.13 ET-OPTO-DC MOTOR

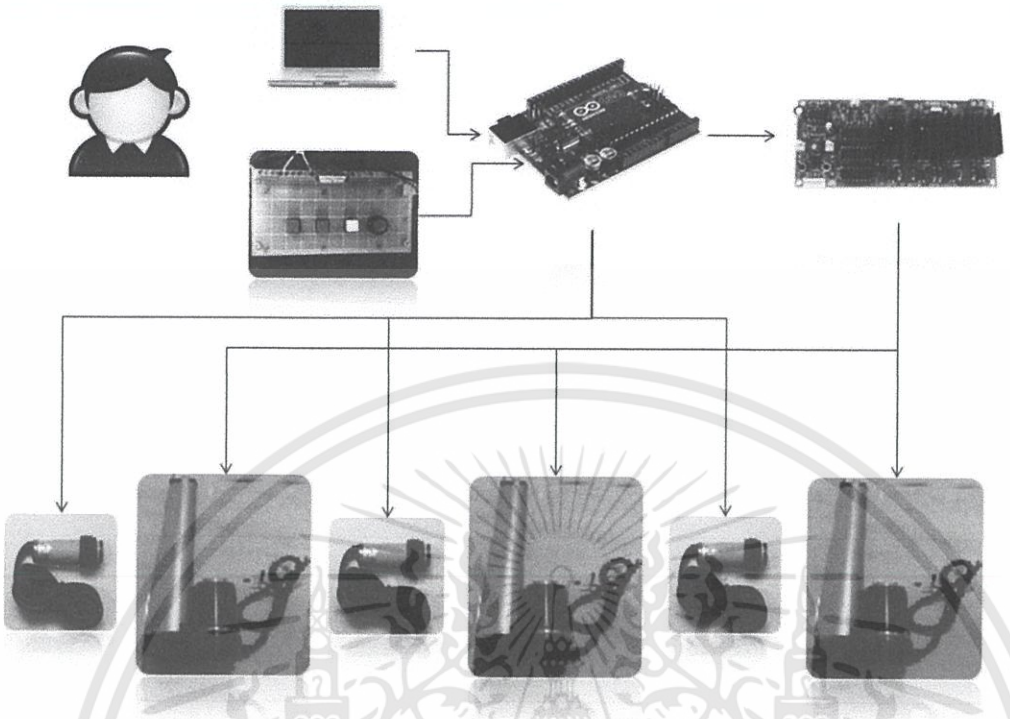
การที่จะปล่อยสินค้าตามการป้อนข้อมูลบนรีโมทคอนโทรล เกิดจากการทำงานของกระบอกสูบไฟฟ้า และมีบอร์ดไมโครคอนโทรลเลอร์เป็นตัวประมวลผล แต่สัญญาณที่ออกมาจากบอร์ดไมโครคอนโทรลเลอร์นั้น ไม่เพียงพอที่จะทำให้กระบอกสูบไฟฟ้าทำงานได้ จึงต้องมี ET-OPTO-DC MOTOR เป็นตัวขับให้กระบอกสูบไฟฟ้าทำงานได้



รูปที่ 3.14 การเชื่อมต่อไมโครคอนโทรลเลอร์, รีโมทคอนโทรล และ ET-OPTO-DC MOTOR

3.3 โครงสร้างของระบบ

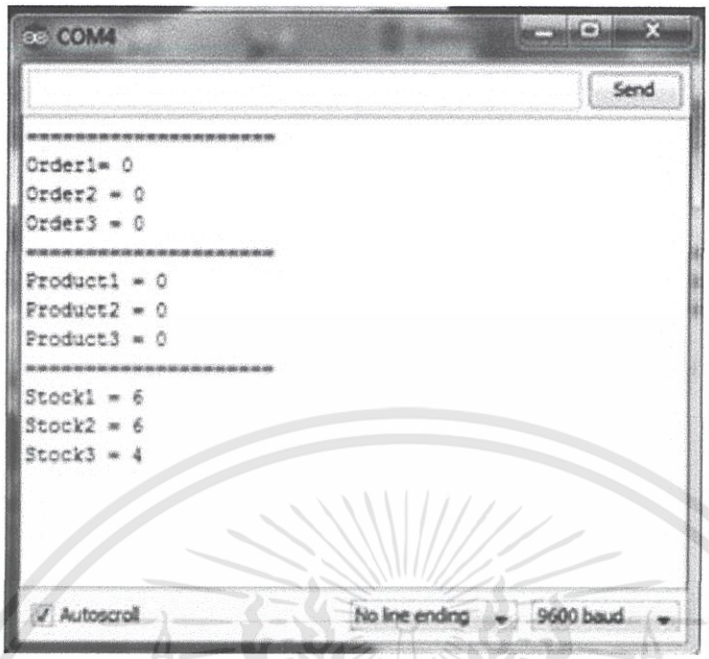
โครงสร้างโดยรวมของโครงงานแสดงดังรูปที่ 3.15 ซึ่งจะเริ่มจากผู้ใช้งานกดเลือกจำนวนและชนิดสินค้าที่ต้องการสั่งจากรีโมทคอนโทรล ข้อมูลจะถูกส่งไปยังไมโครคอนโทรลเลอร์ โดยมีคอมพิวเตอร์โน้ตบุ๊กเป็นจอแสดงผลสถานะการทำงาน และมี DC MOTOR DRIVER เป็นตัวขับมอเตอร์ให้หมุนขึ้น-ลงตามคำสั่ง เพื่อดันสินค้าให้ได้ตรงตามชนิดและจำนวนจากที่ผู้ใช้ได้ทำการสั่งไว้ และมีเซนเซอร์ตรวจสอบสินค้าว่าปล่อยได้ครบตามชนิดและจำนวนที่ได้ทำการสั่งไว้หรือไม่



รูปที่ 3.15 โครงสร้างโดยรวมของระบบ

3.3.1 จอแสดงผล

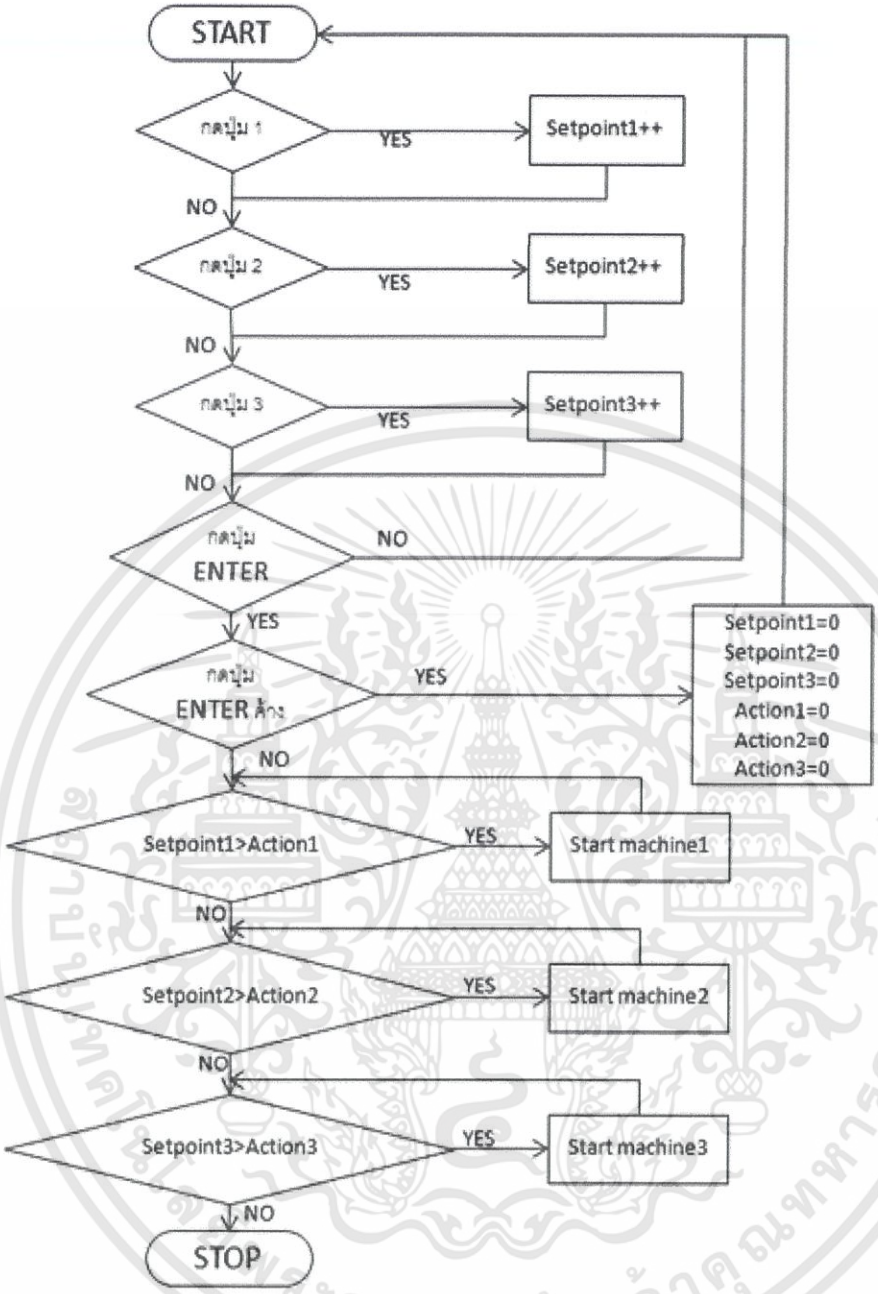
โดยจอแสดงผลของโครงงานนี้ จะใช้การเขียนโปรแกรมลงในบอร์ดคอนโทรลเลอร์เพื่อให้สามารถแสดงสถานการณ์ทำงานว่าปล่อยสินค้าครบตามจำนวนการสั่งหรือยัง และยังสามารถบอกได้ว่าขณะนี้มีจำนวนสินค้าแต่ละชนิดมีอยู่ในรางเท่าไหร่ รวมไปถึงแสดงการแจ้งเตือนให้เติมสินค้าในแต่ละรางในกรณีที่สินค้ามีอยู่ในรางเพียง 2 ชิ้น โดยในโครงงานนี้คอมพิวเตอร์โน๊ตบุ๊คเป็นจอแสดงผลดังกล่าวดังรูปที่ 3.16



รูปที่ 3.16 จอแสดงผล

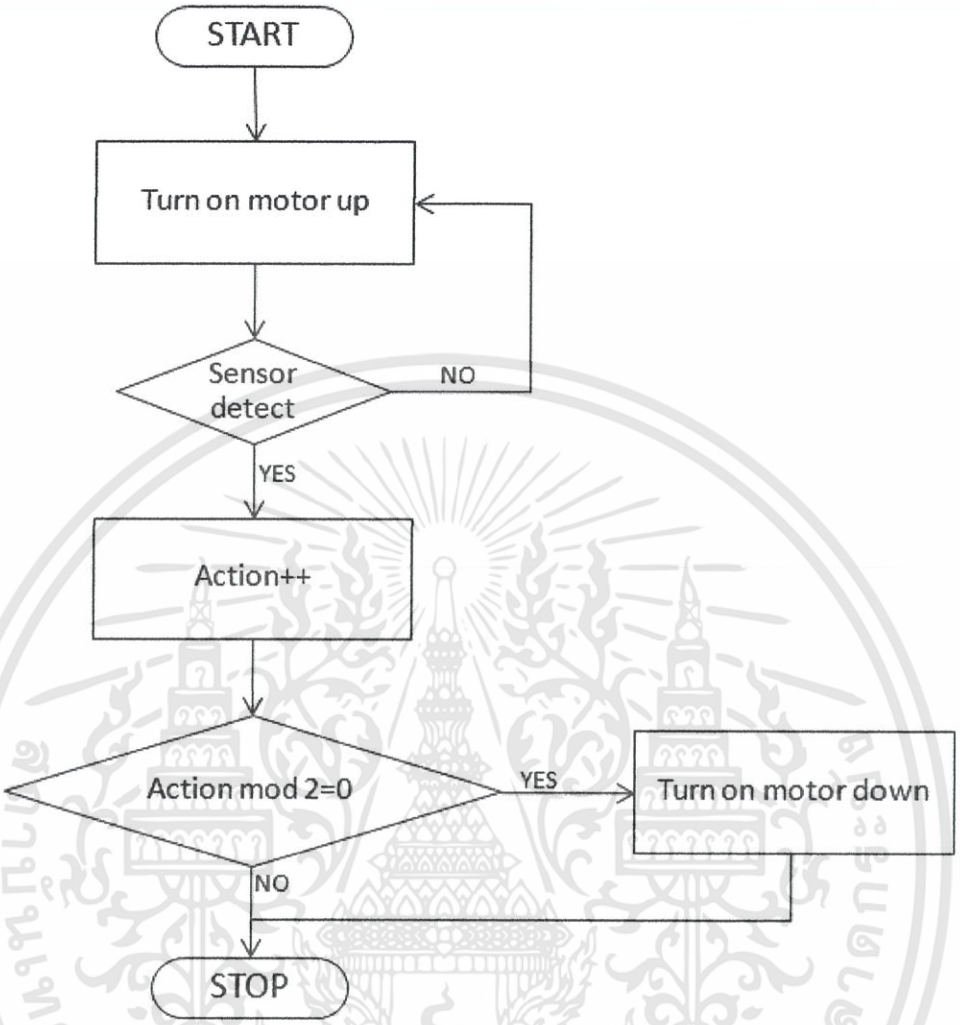
3.3.2 การออกแบบด้านซอฟแวร์

โครงงานนี้เป็นการออกแบบโปรแกรมควบคุม การปล่อยสินค้าให้ปล่อยสินค้าตรงตามความต้องการของลูกค้า โดยสามารถอธิบายการทำงานของระบบดังรูปที่ 3.17 จากแผนผังจะเป็นขั้นตอนการทำงานของเครื่องปล่อยสินค้า เมื่อมีการกดปุ่มเลือกชนิดสินค้าทั้ง 3 ปุ่ม ค่าเซ็ตพอยท์ของแต่ละชนิดสินค้าจะเพิ่มขึ้นตามจำนวนครั้งที่กด หลังจากที่เลือกชนิดและจำนวนสินค้าเสร็จให้กดปุ่มเอ็นเธอร์ แต่ในกรณีที่ต้องยกเลิกการสั่งสินค้าให้กดปุ่มเอ็นเธอร์ค้างระบบจะทำการรีเซ็ตกลับสู่ค่าเริ่มต้น ในกรณีที่ยอมรับการเลือกนั้นให้กดปุ่มเอ็นเธอร์ ระบบจะเปรียบเทียบค่าระหว่างเซตพอยต์กับจำนวนสินค้าที่ได้ทำการปล่อยไปแล้ว ซึ่งจำนวนสินค้านั้นจะมาจากการตรวจจับของเซนเซอร์ ถ้าหากค่าเซ็ตพอยท์ยังมีค่ามากกว่าจำนวนสินค้า ระบบก็จะสั่งให้เครื่องปล่อยสินค้าไปเรื่อยๆ ระบบจะทำการเปรียบเทียบค่าเซ็ตพอยท์กับจำนวนสินค้าที่ปล่อยไปแล้วไปจนกว่าจะมีค่าเท่ากัน ระบบจะทำการเช็คในรางถัดไป โดยจะมีขั้นตอนเหมือนกับที่ได้กล่าวมา และทำการเช็คเปรียบเทียบจนถึงรางสุดท้าย ระบบจึงจะสั่งให้หยุดการทำงาน



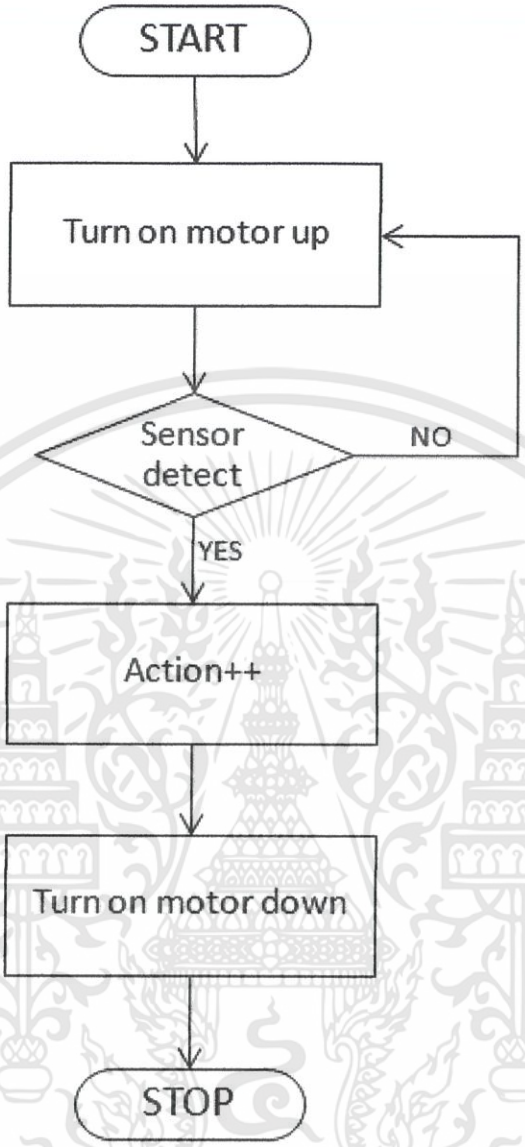
รูปที่ 3.17 การทำงานของระบบ

การทำงานของระบบในรางที่ 1 และ 2 เริ่มต้นจากการกดปุ่มเลือกชนิดสินค้าทั้ง 3 ปุ่มส่งสัญญาณไปยังบอร์ดไมโครคอนโทรลเลอร์เพื่อสั่งให้มอเตอร์ของกระบอกสูบไฟฟ้าเริ่มทำงาน โดยกระบอกสูบไฟฟ้าจะทำงานไปเรื่อยๆ จนกว่าพร็อกซิมิตี้เซนเซอร์ตรวจพบวัตถุที่ผ่านตกลงมา ซึ่งระบบจะมีการตั้งค่าให้กระบอกสูบไฟฟ้ากลับสู่จุดเริ่มต้นใหม่ โดยเมื่อค่าจำนวนสินค้าที่ได้ทำการปล่อยไปแล้วหารสองลงตัวและระบบจะสั่งให้หยุดการทำงานดังรูปที่ 3.18



รูปที่ 3.18 การทำงานของระบบในรางที่ 1 และ 2

การทำงานของระบบในรางที่ 3 จะมีลักษณะคล้ายกันกับการทำงานของระบบในรางที่ 1 และ 2 เริ่มต้นจากการกดปุ่มเลือกชนิดสินค้าทั้ง 3 ปุ่มส่งสัญญาณไปยังบอร์ดไมโครคอนโทรลเลอร์ เพื่อสั่งให้มอเตอร์ของกระบอกสูบไฟฟ้าเริ่มทำงาน โดยกระบอกสูบไฟฟ้าจะทำงานไปเรื่อยๆ จนกว่าพร็อกซิมิตี้เซนเซอร์ตรวจพบวัตถุที่ผ่านตกลงมา ซึ่งในรางที่ 3 พอเวลาพร็อกซิมิตี้เซนเซอร์ตรวจพบวัตถุที่ผ่านตกลงมา ระบบจะสั่งให้กระบอกสูบไฟฟ้ากลับมายังจุดเริ่มต้นและสั่งให้หยุดการทำงานดังรูปที่ 3.19



รูปที่ 3.19 การทำงานของระบบในรางที่ 3

บทที่ 4
การทดลองและผลการทดลอง

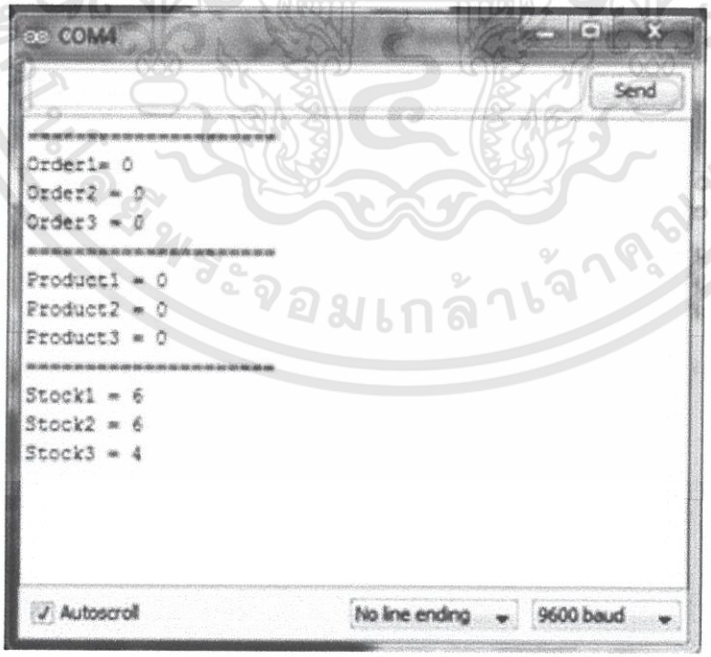
ในโครงงานนี้จะเป็นการทดลองการปล่อยสินค้าโดยจะมีสินค้าอยู่ด้วยทั้งหมด 3 ชนิด ซึ่งลักษณะของการปล่อยสินค้าจะแบ่งเป็น 2 ลักษณะ

4.1 สั่งสินค้าในรางที่ 1 และ 2

ขั้นตอนการทำงานของการปล่อยสินค้าของรางที่ 1 และ 2 มีลักษณะการทำงานเหมือนกัน โดยขั้นตอนแรกของการทำงานคือ การกดเลือกชนิดสินค้า (Order) ระบบจะสั่งให้กระบอกสูบไฟฟ้าดันสินค้าให้ตกลงมาตามจำนวนที่เลือก โดยจำนวนสินค้าสามารถดูได้จากหน้าจอแสดงผล

4.1.1 ขั้นตอนเริ่มต้น

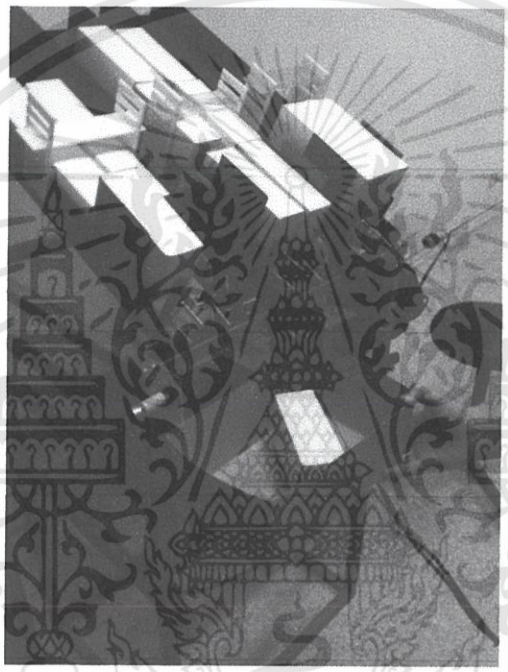
ยังไม่มีการกดสั่งออเดอร์สินค้า นั่นหมายความว่าคลังสินค้า (Stock) ยังคงเต็มอยู่เท่าเดิม คือ Stock1 = 6, Stock2 = 6, Stock3 = 4 โดยค่าออเดอร์ (Order) และค่าโปรดักส์ (Product) ยังคงเป็นศูนย์ ซึ่งค่าออเดอร์ (Order) คือ ค่าการกดปุ่มเลือกจำนวนชนิดสินค้าจากรีโมทคอนโทรล ค่าโปรดักส์ (Product) คือ การนับจำนวนสินค้าที่ถูกปล่อยออกมา โดยในโครงงานนี้จะใช้โฟโตอิเล็กทริกเซนเซอร์ เป็นตัวรับและส่งสัญญาณไปยังไมโครคอนโทรลเลอร์ เมื่อมีสินค้าตกผ่านเซนเซอร์และจะแสดงค่าบนหน้าจอแสดงผลดังรูปที่ 4.1



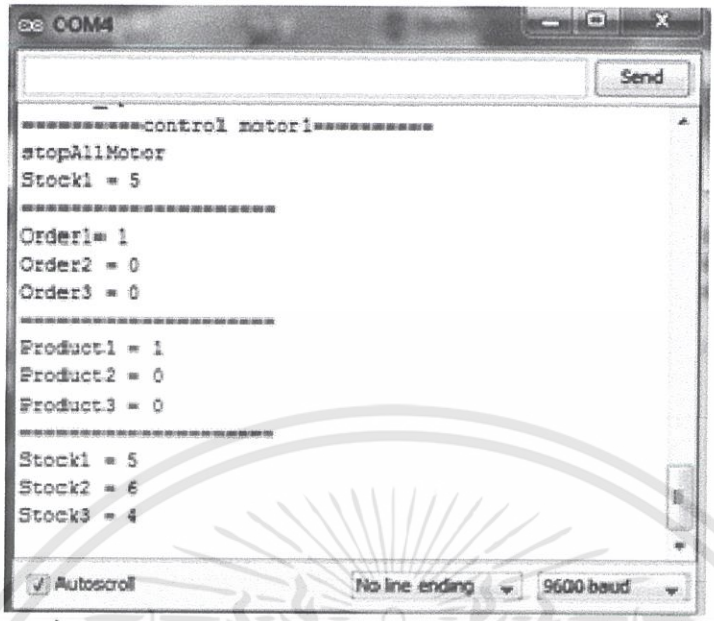
รูปที่ 4.1 หน้าจอแสดงผลกรณียังไม่มีการกดสั่งออเดอร์

4.1.2 ขั้นตอนการสั่งสินค้าในรางที่ 1 จำนวน 1 ชิ้น

ให้กดปุ่มสีฟ้า 1 ครั้ง และกดปุ่มดำ-แดงเป็นการกดตกลง แสดงว่าเป็นการสั่งสินค้าในรางที่ 1 เป็นจำนวน 1 ชิ้น กระบอกสูบไฟฟ้าจะยืดตัวออกทำให้สินค้าที่อยู่ฝั่งขวาของรางที่ 1 ได้ถูกปล่อยออกมาจากราง โดยมีพร็อกซิมิตี้เซนเซอร์จะเป็นตัวตรวจจับสินค้าที่ตกลงมา และทำให้กระบอกสูบไฟฟ้าหยุดทำงานดังรูปที่ 4.2 นั่นหมายความว่าคลังสินค้า (Stock) จะมีค่าเป็น Stock1 = 6, Stock2 = 6, Stock3 = 4 และค่า Product1 = 1, Product2 = 0, Product3 = 0 โดยจะแสดงค่าบนหน้าจอแสดงผลดังรูปที่ 4.3

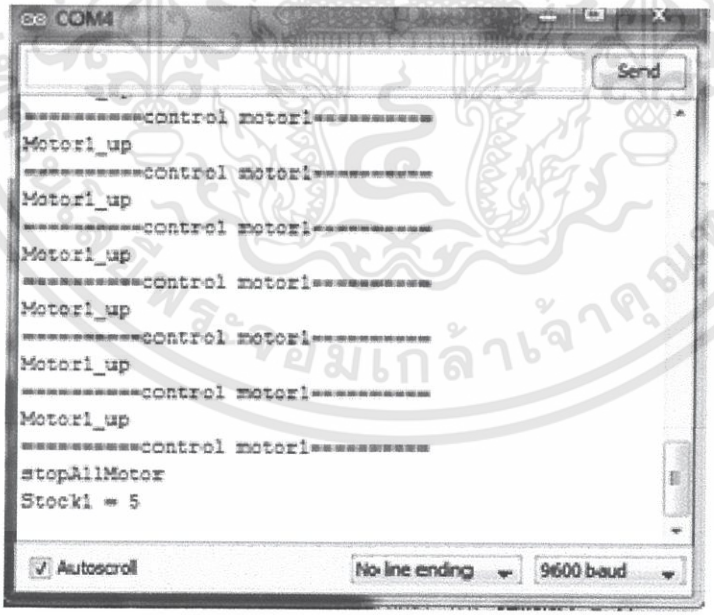


รูปที่ 4.2 สินค้าด้านขวาถูกปล่อย



รูปที่ 4.3 หน้าจอแสดงผลการสั่งสินค้าในรางที่ 1 จำนวน 1 ชิ้น

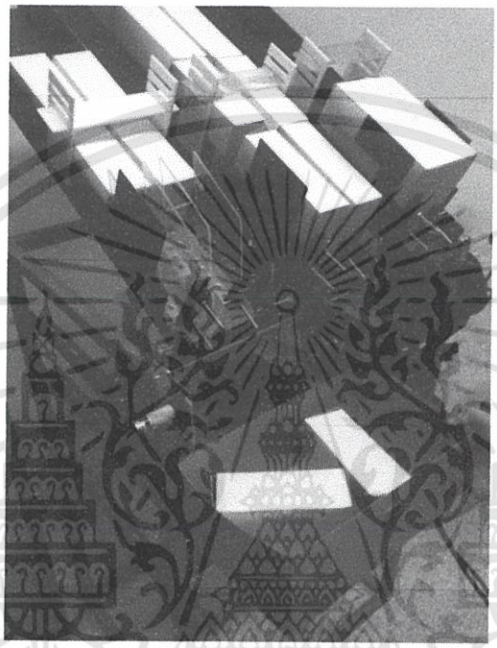
ในขณะที่มีการปล่อยสินค้า กระบอกสูบไฟฟ้าจะทำงานไปเรื่อยๆ จนกว่าดันสินค้าให้ตกลงมาผ่านโฟโตอิเล็กทริกเซนเซอร์กระบอกสูบไฟฟ้าจะหยุดทำงาน ซึ่งสามารถดูการทำงานในการปล่อยสินค้าได้ผ่านหน้าจอแสดงผล โดยเมื่อมีสินค้าตกผ่านโฟโตอิเล็กทริกเซนเซอร์ หน้าจอแสดงผลจะระบุให้กระบอกสูบไฟฟ้าหยุดทำงาน (Stopallmotor) ดังรูปที่ 4.4



รูปที่ 4.4 หน้าจอแสดงผลการทำงานของกระบอกสูบไฟฟ้าเมื่อมีการสั่งสินค้าชิ้นที่ 1

4.1.3 ขั้นตอนการสั่งสินค้าในรางที่ 1 จำนวน 2 ชิ้น

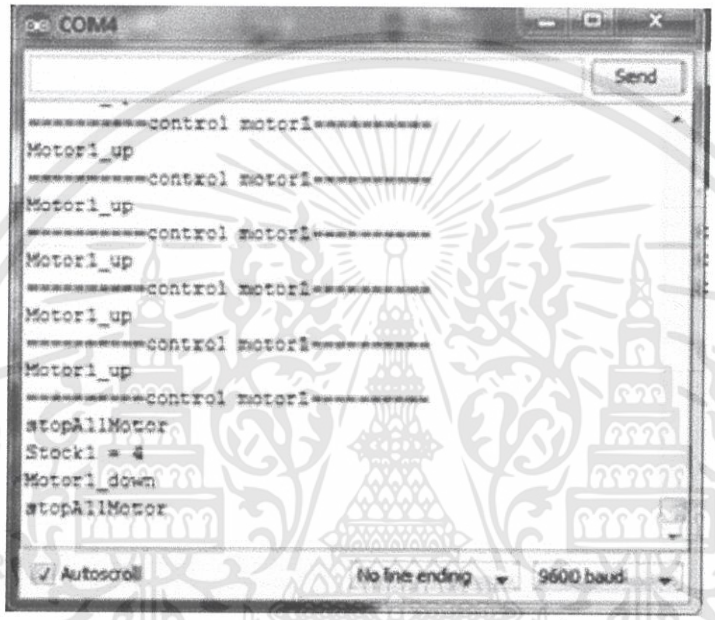
ให้กดปุ่มสีฟ้าอีก 1 ครั้ง และกดปุ่มดำ-แดงเป็นการสั่งสินค้าในรางที่ 1 อีก 1 ชิ้น กระบอกสูบไฟฟ้ายืดตัวออกจากตำแหน่งเดิมที่มีการปล่อยสินค้าชิ้นแรก ทำให้สินค้าด้านซ้ายของรางที่ 1 ถูกปล่อยออกมาจากรางเป็นชิ้นที่ 2 โดยพร็อกซิมิตี้เซนเซอร์จะเป็นตัวตรวจจับสินค้าที่ตกลงมาดังรูปที่ 4.5 และทำให้กระบอกสูบไฟฟ้าหดตัวกลับมายังจุดเริ่มต้นดังรูปที่ 4.6



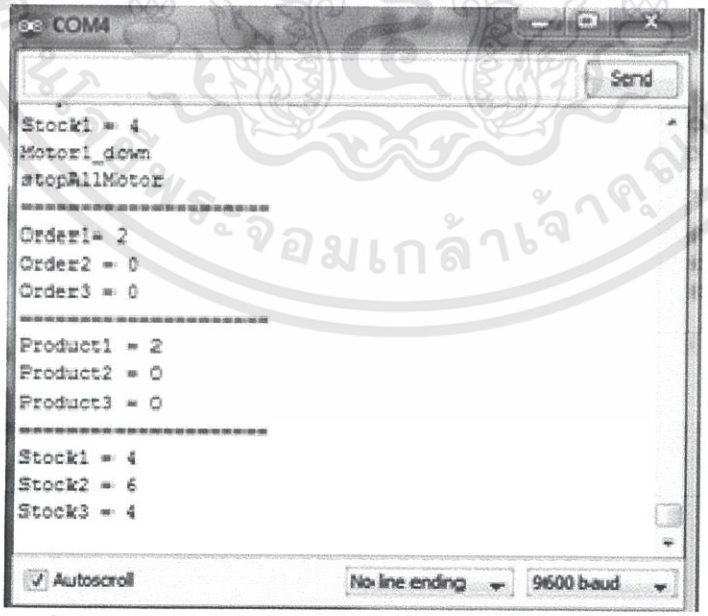
รูปที่ 4.5 สินค้าด้านซ้ายถูกปล่อยเป็นชิ้นที่ 2

รูปที่ 4.6 กระบอกสูบไฟฟ้ากลับมาจุดเริ่มต้น

ในการปล่อยสินค้าชิ้นที่ 2 กระบอกสูบไฟฟ้าจะยืดตัวออกจากตำแหน่งเดิมที่มีการปล่อยสินค้าชิ้นแรก โดยกระบอกสูบไฟฟ้าจะทำงานไปเรื่อยๆ จนกว่าดันสินค้าให้ตกลงมาผ่านโฟโตอิเล็กทริกเซนเซอร์กระบอกสูบไฟฟ้าจะหยุดทำงาน (Stopallmotor) และหดตัวกลับมายังจุดเริ่มต้น (motor1_down) ดังรูปที่ 4.7 นั่นหมายความว่าคลังสินค้า (Stock) จะมีค่าเป็น Stock1 = 4, Stock2 = 6, Stock3 = 4 และค่า Product1 = 2, Product2 = 0, Product3 = 0 โดยจะแสดงค่าบนหน้าจอแสดงผลดังรูปที่ 4.8



รูปที่ 4.7 หน้าจอแสดงผลการทำงานของกระบอกสูบไฟฟ้าเมื่อมีการสั่งสินค้าชิ้นที่ 2



รูปที่ 4.8 หน้าจอแสดงผลการสั่งสินค้าในรางที่ 1 จำนวน 2 ชิ้น

ขั้นตอนการทำงานของการปล่อยสินค้าของรางที่ 1 และ 2 มีลักษณะการทำงานเหมือนกัน คือ กระบอกสูบไฟฟ้าจะทำงานเป็น 3 ระดับได้แก่ ระดับแรก เป็นระดับเริ่มต้นที่ยังไม่มีการกดเลือกสินค้า ระดับที่สอง เป็นระดับที่มีการปล่อยสินค้าชิ้นที่ 1 และระดับสุดท้าย เป็นระดับที่มีการปล่อยสินค้าชิ้นที่ 2 โดยเมื่อปล่อยสินค้าชิ้นที่ 2 ตกผ่านโฟโตอิเล็กทริกเซนเซอร์ ระบบจะสั่งให้กระบอกสูบไฟฟ้าหดตัวกลับมายังจุดเริ่มต้น

ตารางที่ 4.1 แสดงผลการทดลองปล่อยสินค้าของรางที่ 1

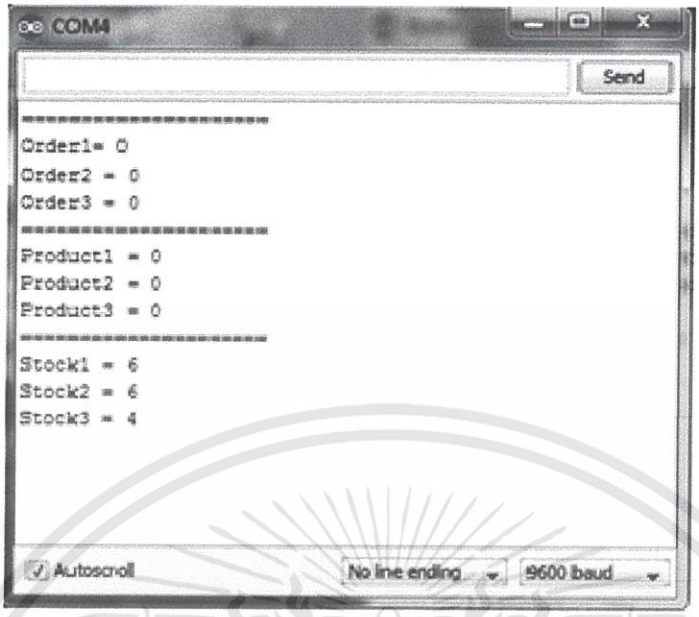
จำนวนสินค้า(ชิ้น) \ การทดลอง(ครั้ง)	1	2	3	4	5
1	√	√	√	√	√
2	√	√	√	X	√
3	√	√	√	√	√
4	√	√	√	√	√
5	√	X	√	√	√
6	√	√	√	√	√

สรุป

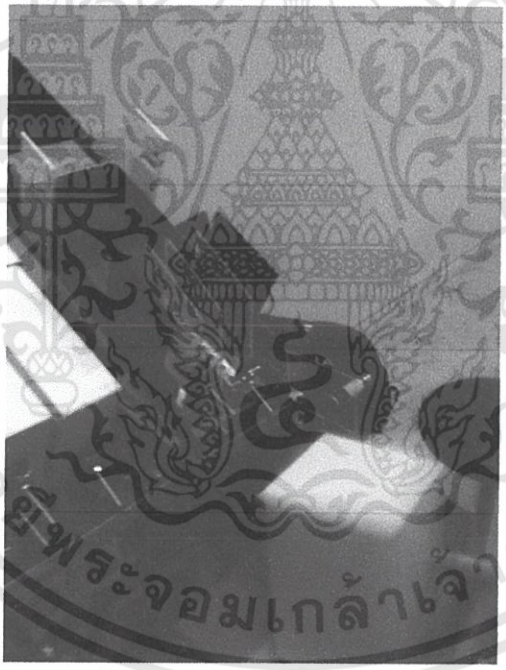
จากการทดลองของการปล่อยสินค้าของรางที่ 1 จะเห็นว่าผลการทดลองของครั้งที่ 2 และ 4 เกิดความผิดพลาดในการปล่อยสินค้า 5 ชิ้น และ 2 ชิ้นตามลำดับ โดยข้อผิดพลาดที่เกิดขึ้นคือ สินค้าที่ถูกปล่อยไม่ตกผ่านพร็อกซิมิตี้เซนเซอร์ทำให้ระบบไม่สามารถทำงานต่อได้ แต่ข้อผิดพลาดดังกล่าวเกิดขึ้นเพียง 6.67% ของการทดลองทั้งหมด

4.2 สั่งสินค้าในรางที่ 3

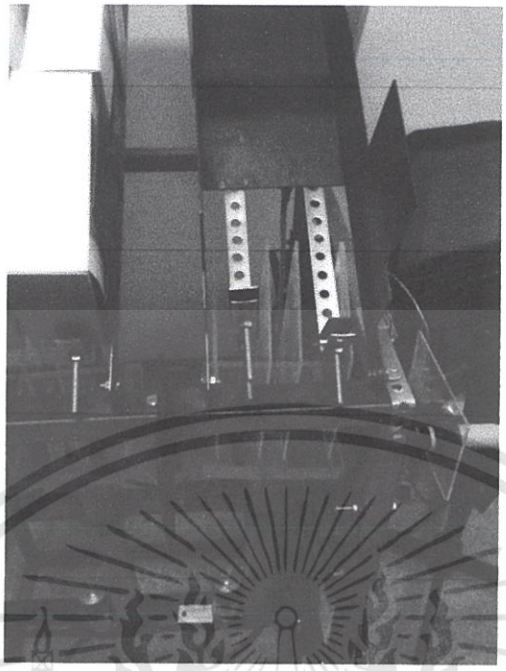
ในขั้นตอนเริ่มต้น ยังไม่มีการกดสั่งออเดอร์สินค้า คลังสินค้า (Stock) ยังคงเต็มอยู่เท่าเดิม คือ Stock1 = 6, Stock2 = 6, Stock3 = 4 ดังรูปที่ 4.9 วิธีการทำงานของการปล่อยสินค้ารางที่ 3 กดปุ่มสีเหลือง 1 ครั้ง และกดปุ่มดำ-แดงเป็นการสั่งสินค้าในรางที่ 3 เป็นจำนวน 1 ชิ้น ซึ่งการทำงานกระบอกสูบไฟฟ้าจะทำการยืดตัวออกทำให้สินค้าถูกปล่อยออกมาจากรางดังรูปที่ 4.10 โดยเมื่อสินค้าตกผ่านพร็อกซิมิตี้เซนเซอร์ ระบบจะสั่งให้กระบอกสูบไฟฟ้าหดตัวกลับไปยังจุดเริ่มต้นดังรูปที่ 4.11 และดูการทำงานกระบอกสูบไฟฟ้าของรางที่ 3 ได้จากหน้าจอแสดงผลดังรูปที่ 4.12 นั่นหมายความว่าคลังสินค้า (Stock) จะมีค่าเป็น Stock1 = 6, Stock2 = 6, Stock3 = 3 และค่า Product1 = 0, Product2 = 0, Product3 = 1 โดยจะแสดงค่าบนหน้าจอแสดงผลดังรูปที่ 4.13



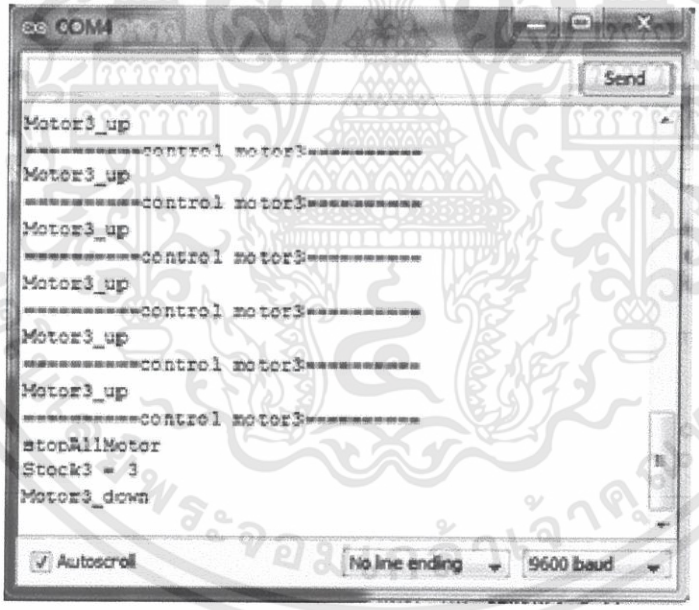
รูปที่ 4.9 หน้าจอแสดงผลกรณียังไม่มีการกดสั่งออเดอร์



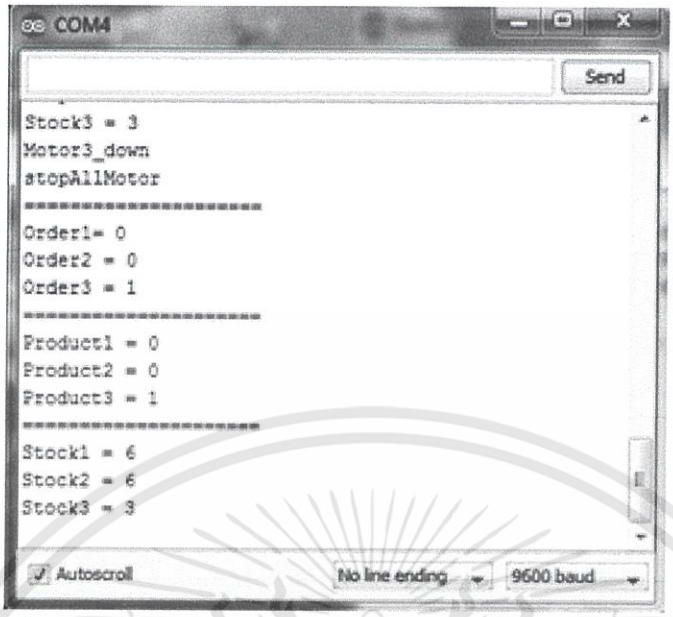
รูปที่ 4.10 สินค้ารางที่ 3 ถูกปล่อย



รูปที่ 4.11 กระบอกสูบไฟฟ้ากลับมาจุดเริ่มต้น



รูปที่ 4.12 หน้าจอแสดงผลการทำงานกระบอกสูบไฟฟ้าของรางที่ 3 จำนวน 1 ชิ้น



รูปที่ 4.13 หน้าจอแสดงผลการสั่งสินค้าในรางที่ 3 จำนวน 1 ชิ้น

ตารางที่ 4.2 แสดงผลการทดลองปล่อยสินค้าของรางที่ 3

การทดลอง(ครั้ง) / จำนวนสินค้า(ชิ้น)	1	2	3	4	5
1	√	√	√	√	√
2	√	√	√	√	√
3	√	√	√	√	√

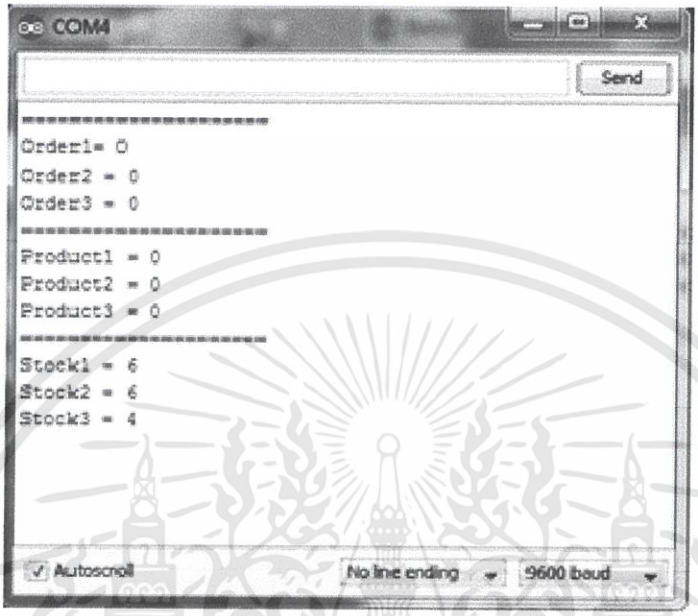
สรุป

จากการทดลองของการปล่อยสินค้าของรางที่ 3 จะเห็นว่าสินค้าได้ถูกปล่อยตรงตามต้องการในทุกกรณีโดยไม่มีข้อผิดพลาด แสดงให้เห็นว่าการทดลองการปล่อยสินค้ารางที่ 3 ทำงานได้ถูกต้อง 100%

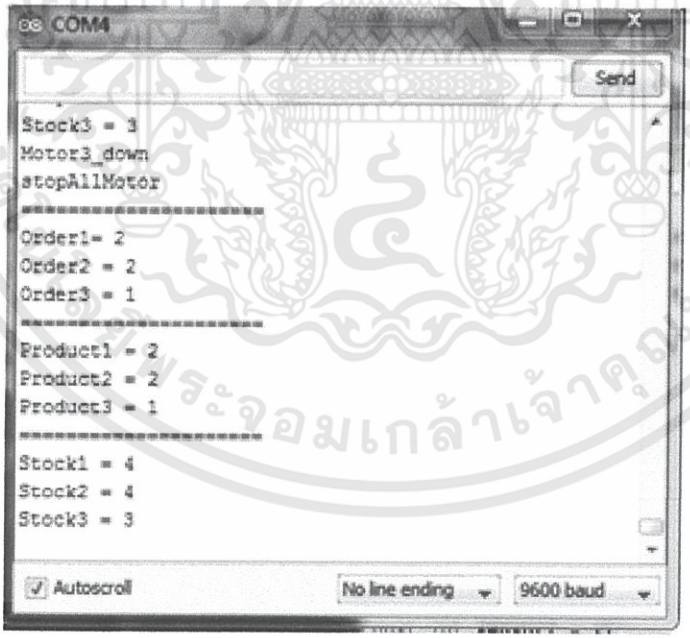
4.3 สั่งสินค้าในรางที่ 1, 2 และ 3 พร้อมกัน

ในขั้นตอนเริ่มต้น ยังไม่มีการกดสั่งออเดอร์สินค้า คลังสินค้า (Stock) ยังคงเต็มอยู่เท่าเดิม คือ Stock1 = 6, Stock2 = 6, Stock3 = 4 ดังรูปที่ 4.14 กดปุ่มสีฟ้า 2 ครั้ง สีแดง 2 ครั้งและเหลือง 1 ครั้ง โดยรางที่ 1 และ 2 จะทำงานตามข้อ 4.1.3 โดยรางที่ 2 จะเริ่มทำงานหลังจากที่กระบอกสูบไฟฟ้าของรางที่ 1 หยุดทำงานเป็นเวลา 2 วินาที และรางที่ 3 ก็จะทำงานตามข้อ 4.2 โดยจะเริ่มทำงานหลังจากที่กระบอกสูบไฟฟ้าของรางที่ 2 หยุดทำงานเป็นเวลา 2 วินาที นั่นหมายความว่า

คลังสินค้า (Stock) จะมีค่าเป็น Stock1 = 4, Stock2 = 4, Stock3 = 3 และค่า Product1 = 2, Product2 = 2, Product3 = 1 โดยจะแสดงค่าบนหน้าจอแสดงผลดังรูปที่ 4.15



รูปที่ 4.14 หน้าจอแสดงผลกรณียังไม่มีการกดสั่งออเดอร์



รูปที่ 4.15 หน้าจอแสดงผลกรณีการสั่งสินค้าในรางที่ 1 จำนวน 2 ชิ้น รางที่ 2 จำนวน 2 ชิ้น และรางที่ 3 จำนวน 1 ชิ้น

ตารางที่ 4.3 แสดงผลการทดลองปล่อยสินค้าของรางที่ 1, 2 และ 3 พร้อมกัน

การทดลอง(ครั้ง) / จำนวนสินค้า(ชิ้น)			1	2	3	4
รางที่1	รางที่2	รางที่3				
2	2	2	√	√	√	√
1	2	2	√	√	√	√
3	1	1	√	X	√	√
2	2	1	√	√	√	√

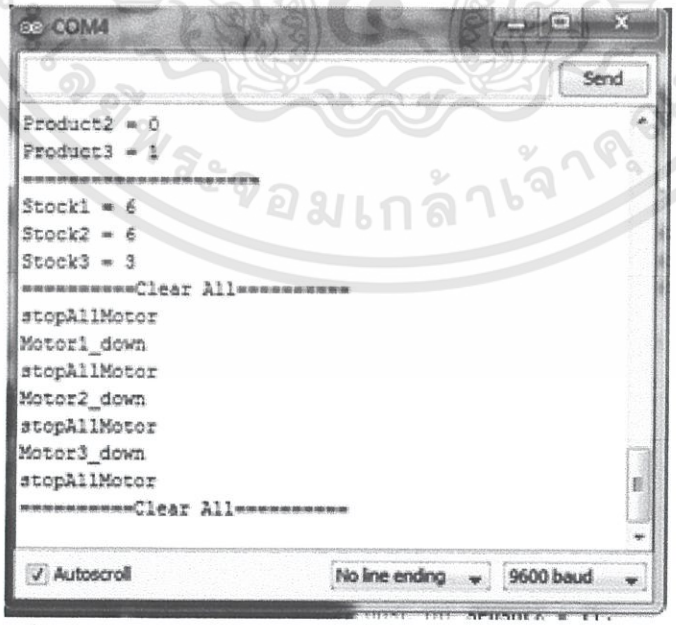
สรุป

จากการทดลองของการปล่อยสินค้าของรางที่ 1, 2 และ3 จะเห็นว่าส่วนใหญ่ระบบจะปล่อยสินค้าได้ตรงตามต้องการทั้ง 3 ราง แต่จะมีข้อผิดพลาดในการทดลองครั้งที่ 2 โดยรางที่ 2 สินค้าที่ถูกปล่อยไม่ตกผ่านพร็อกซิมิตี้เซนเซอร์ทำให้ระบบไม่สามารถทำงานต่อได้ แต่ข้อผิดพลาดดังกล่าวเกิดขึ้นเพียง 6.25% ของการทดลองทั้งหมด

4.4 คำสั่งรีเซ็ต

เมื่อต้องการยกเลิกการสั่งสินค้าสามารถกดปุ่มดำ-แดง ค้างไว้ประมาณ 3 วินาทีดังรูปที่ 4.16 จะแสดงข้อความ “==========Clear All==========” ขึ้นที่มอนิเตอร์ในโปรแกรม Arduino ทำให้ข้อมูลคำสั่งเริ่มต้นใหม่ดังรูปที่ 4.17 ในกรณีที่ต้องการหยุดการทำงานของกระบอกสูบไฟฟ้าที่กำลังยืดตัวออกก็สามารถกดปุ่มดำ-แดง ค้างไว้ประมาณ 3 วินาที ได้เช่นกัน

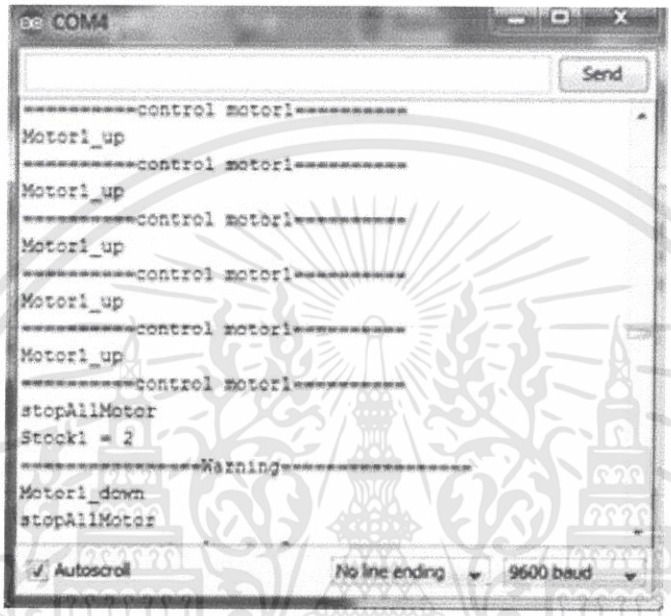
รูปที่ 4.16 กดปุ่มดำ-แดง ค้างไว้ประมาณ 3 วินาที เมื่อต้องการยกเลิกการสั่งสินค้า



รูปที่ 4.17 หน้าจอแสดงผลกรณีเมื่อต้องการยกเลิกการสั่งสินค้า

4.5 ข้อความแจ้งเตือนผู้ใช้งาน

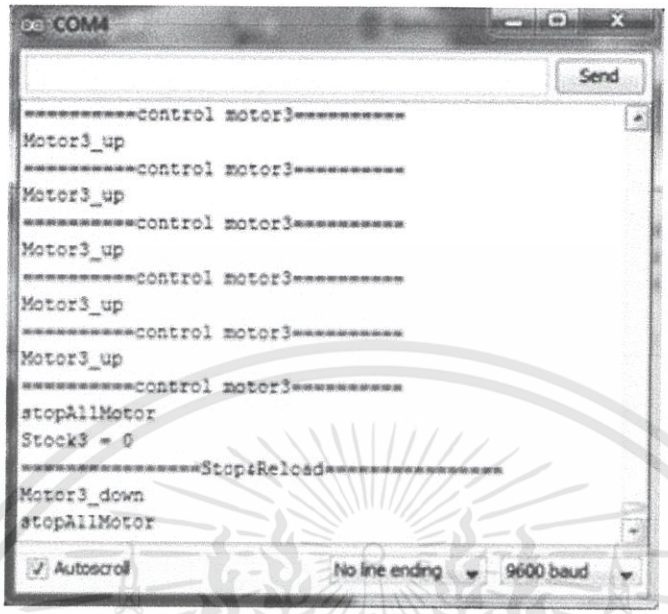
เมื่อมีจำนวนสินค้าในรางจัดเก็บสินค้าเหลืออยู่ 2 ชิ้น ระบบจะแสดงข้อความ “========== warning==========” ขึ้นที่จอมอนิเตอร์ในโปรแกรม Arduino ดังรูปที่ 4.18 เพื่อเป็นการแจ้งเตือนให้ทราบในการเติมสินค้า



รูปที่ 4.18 หน้าจอแสดงผลกรณีเมื่อมีจำนวนสินค้าในรางจัดเก็บเหลือ 2 ชิ้น

4.5.1 กรณีที่สินค้าหมด

ระบบจะแสดงข้อความ “========stop&reload==========” ขึ้นที่จอมอนิเตอร์ในโปรแกรม Arduino ดังรูปที่ 4.19 เป็นการแจ้งให้ทราบถึงการเติมสินค้า



รูปที่ 4.19 หน้าจอแสดงผลกรณีเมื่อสินค้าหมด

สรุป

จากการทดสอบการปล่อยสินค้าทั้ง 3 ราง จะเห็นว่ามีลักษณะการปล่อยสินค้าอยู่ 2 ลักษณะ โดยในผลการทดลองบางครั้งอาจจะเกิดข้อผิดพลาดในการทดลอง คิดเป็นร้อยละ 4.67% ซึ่งคิดจากการทดลองทั้งหมดในกรณีต่างๆ รวม 61 การทดลอง ซึ่งถือว่าอยู่ในเกณฑ์ที่ยอมรับได้

บทที่ 5
บทวิจารณ์และสรุป

5.1 วิจารณ์ผลการทดลอง

จากการทดลองจากเห็นว่าสินค้าได้ถูกปล่อยตามต้องการ และจอแสดงผลได้ผลตอบสนองที่ถูกต้องและรวดเร็ว แต่ในบางครั้งจะมีปัญหากับจอแสดงผลคือ สัญญาณที่ส่งมาจากบอร์ดคอนโทรลเลอร์เกิดขาดหายทำให้ไม่สามารถรู้ได้ว่าสถานะของระบบเป็นเช่นไร แต่ระบบจะสามารถทำงานได้อย่างต่อเนื่อง และเมื่อลองสั่งสินค้าเป็นจำนวนมากขึ้น และซับซ้อนขึ้นระบบก็สามารถทำงานได้ตามปกติ

5.2 สรุปผลการทดลอง

ในส่วนของการทดลองแรก จะเป็นการทดลองการปล่อยสินค้า ซึ่งพบว่าการปล่อยสินค้าเป็นไปตามที่คาดการณ์ไว้คือ สินค้าได้ตกผ่านเซนเซอร์ตรงจุดทำให้เซนเซอร์สามารถตรวจจับวัตถุได้ ระบบจึงทำงานได้โดยไม่มีข้อผิดพลาด แต่ในบางครั้งสินค้าที่ถูกปล่อยออกมา ไม่ผ่านเซนเซอร์ทำให้ระบบเกิดความผิดพลาดไปบ้าง

ส่วนที่ 2 เป็นการทดลองโปรแกรมควบคุม ซึ่งจากการทดลองก็เห็นว่าไม่ว่าจะสั่งสินค้าในกรณีใดๆ ระบบสามารถทำงานได้ตามต้องการได้ทุกกรณี และมีการทดลองผลตอบสนองของจอแสดงผลซึ่งก็ได้ผลตอบสนองที่ถูกต้องตามความจริง ทั้งสถานะของระบบ จำนวนสินค้าและคำแจ้งเตือนต่างๆ

5.3 สรุปผลการดำเนินงาน

1. สินค้าถูกปล่อยออกมาได้สมบูรณ์ในทุกกรณี คิดเป็นร้อยละ 95.33 ของการทดลองทั้งหมด
2. โปรแกรมควบคุมสามารถรองรับการสั่งสินค้าได้ทุกกรณี
3. จอแสดงผลสามารถให้ผลตอบสนองได้ถูกต้อง

5.4 ปัญหาที่เกิดขึ้น

1. การประกอบและติดตั้งอุปกรณ์
2. การเขียนโปรแกรมควบคุมทำได้ยาก เนื่องจากมีพื้นฐานในการเขียนน้อย
3. เนื่องจากจอแสดงผลของโครงงานนี้ใช้คอมพิวเตอร์โน๊ตบุ๊ก จึงไม่ค่อยสะดวกในการเชื่อมต่อระหว่างตัวเครื่องและคอมพิวเตอร์โน๊ตบุ๊ก
4. สินค้าไม่ตกผ่านเซนเซอร์ในบางกรณีคิดเป็นร้อยละ 4.67 ของการทดลองทั้งหมด

5.5 แนวทางการแก้ไขปัญหา

1. ศึกษาการใช้งานของโปรแกรมที่ใช้ในการทำโครงงานเพิ่มเติม
2. ศึกษาการออกแบบเครื่องจักรประเภททำงานอัตโนมัติเพิ่มเติม
3 ศึกษาการออกแบบตำแหน่งการติดตั้งและการเลือกใช้ของเซนเซอร์

5.6 แนวทางปรับปรุงและพัฒนาโครงงาน

1. สามารถนำโครงงานนี้ไปประยุกต์ใช้กับการรับส่งข้อมูลอย่างอื่นได้
2. สามารถทำงานได้อย่างอิสระ ไม่ต้องใช้คอมพิวเตอร์โน๊ตบุ๊กช่วย
3. สามารถรองรับสินค้าได้จำนวนมากและทุกขนาด
4. เพิ่มจำนวนรางให้มากขึ้น

เอกสารอ้างอิง

[1] **"Arduino Uno."** [Online]. Available : http://arduino.cc/en/Main/arduinoBoardUno #.UwNM3ZXNtY0. 2013

[2] **"การใช้งาน AVR ATMEGA 328 สร้างพอร์ต I/O ขนาด8 บิตให้Arduino."** [Online] Available :http://www.electoday.com/index.php?topic=3958.0

[3] "**พร็อกซิมิตี้เซนเซอร์**" [Online]. Available :http://dk.coe.psu.ac.th/assign/proxim/

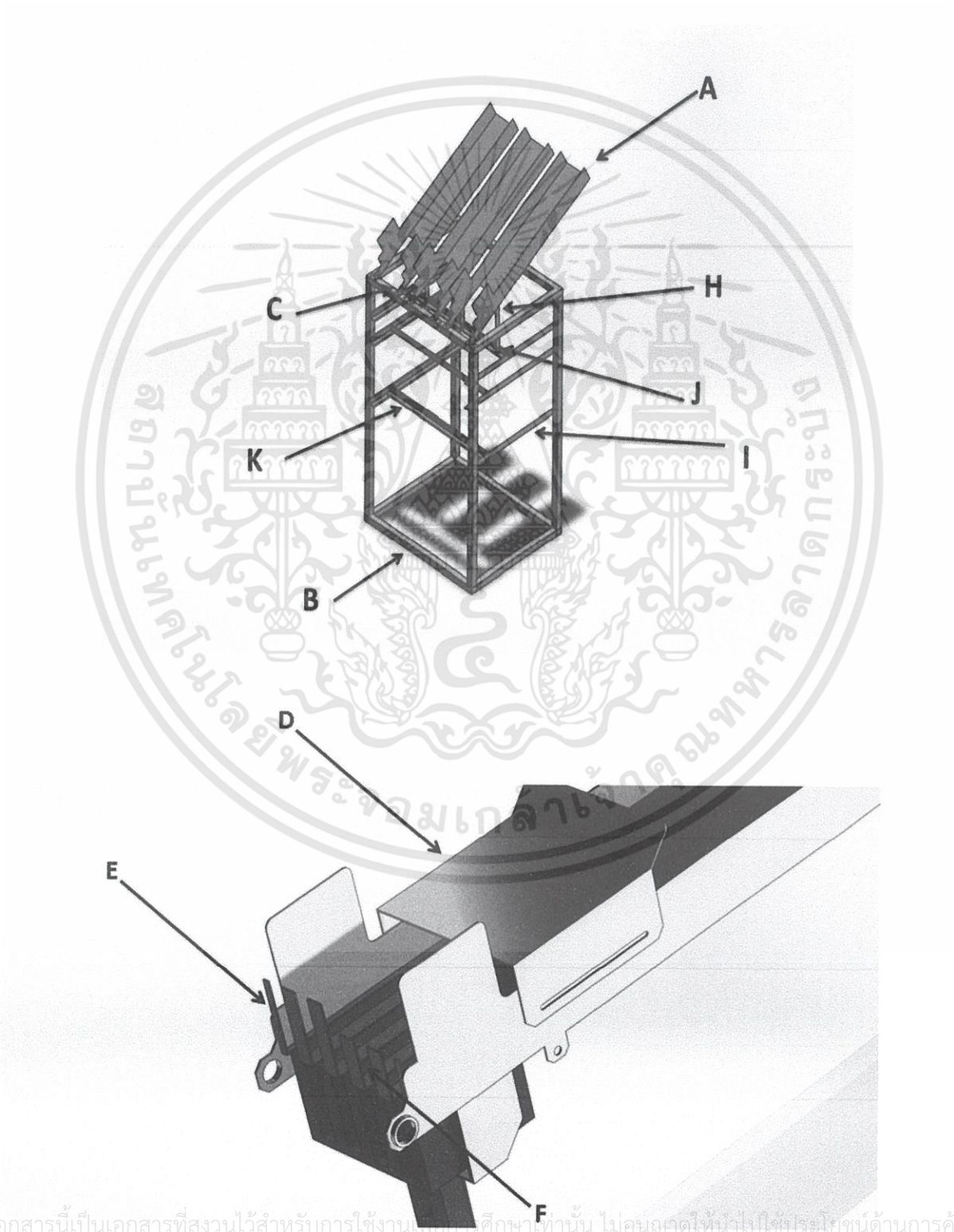
[4] **"ET-OPTO-DC-MOTOR."** [Online]. Available :http://mechanitop.blogpot.com/2010 12/inverted-pendulum-control.html

[5] **"เริ่มต้นการเขียนโปรแกรม"** [Online]. Available :http://www.iprogrammer.info/books/ reviews/5-c/5886-beginning-c-for-arduino.html

ภาคผนวก

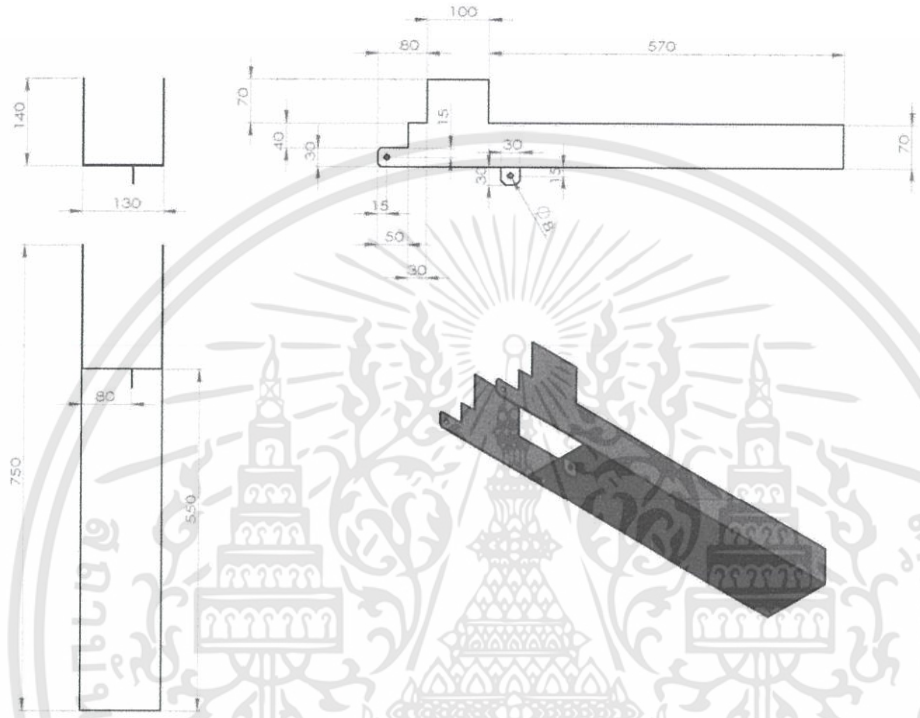
ภาคผนวก ก
ชิ้นส่วนของโครงสร้าง

รูปร่างใน Solidwork



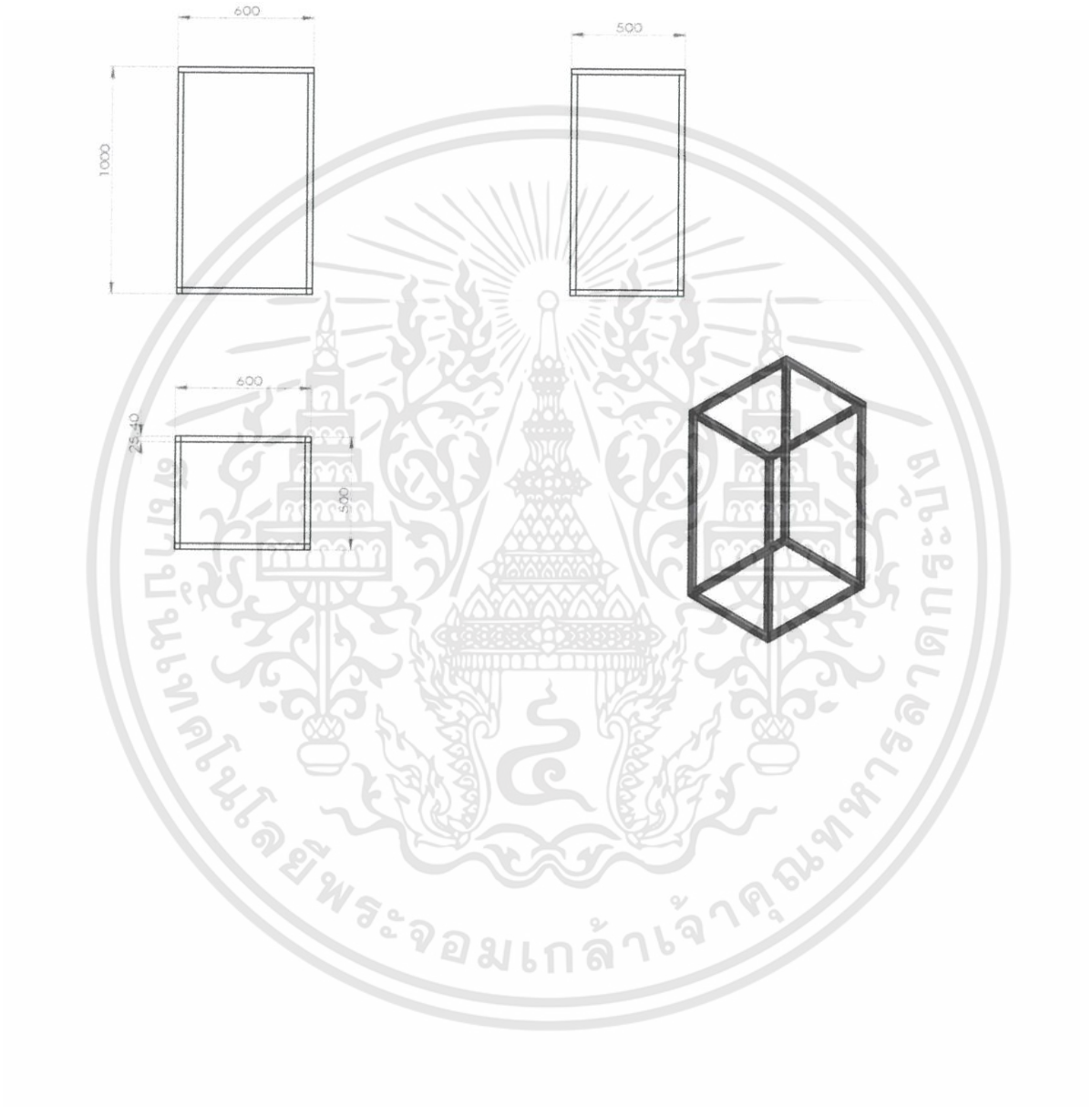
ก.1 รางมี 3 ราง (A)

ขนาดเป็นมิลลิเมตร



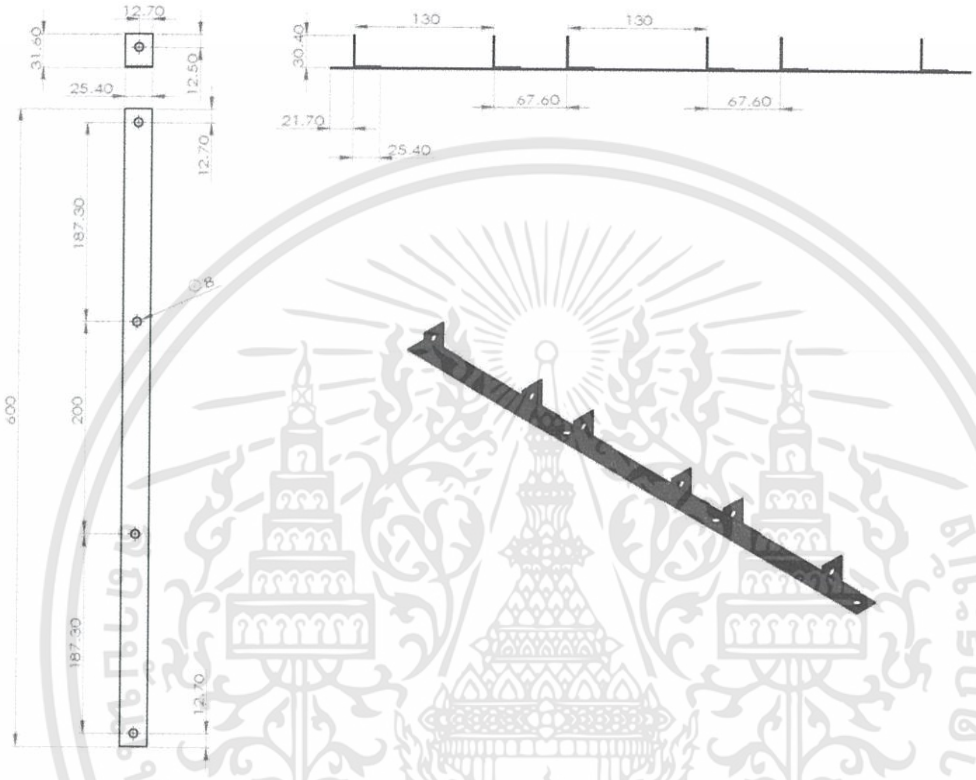
ก.2 ฐานรองราง (B)

หน่วยเป็นมิลลิเมตร



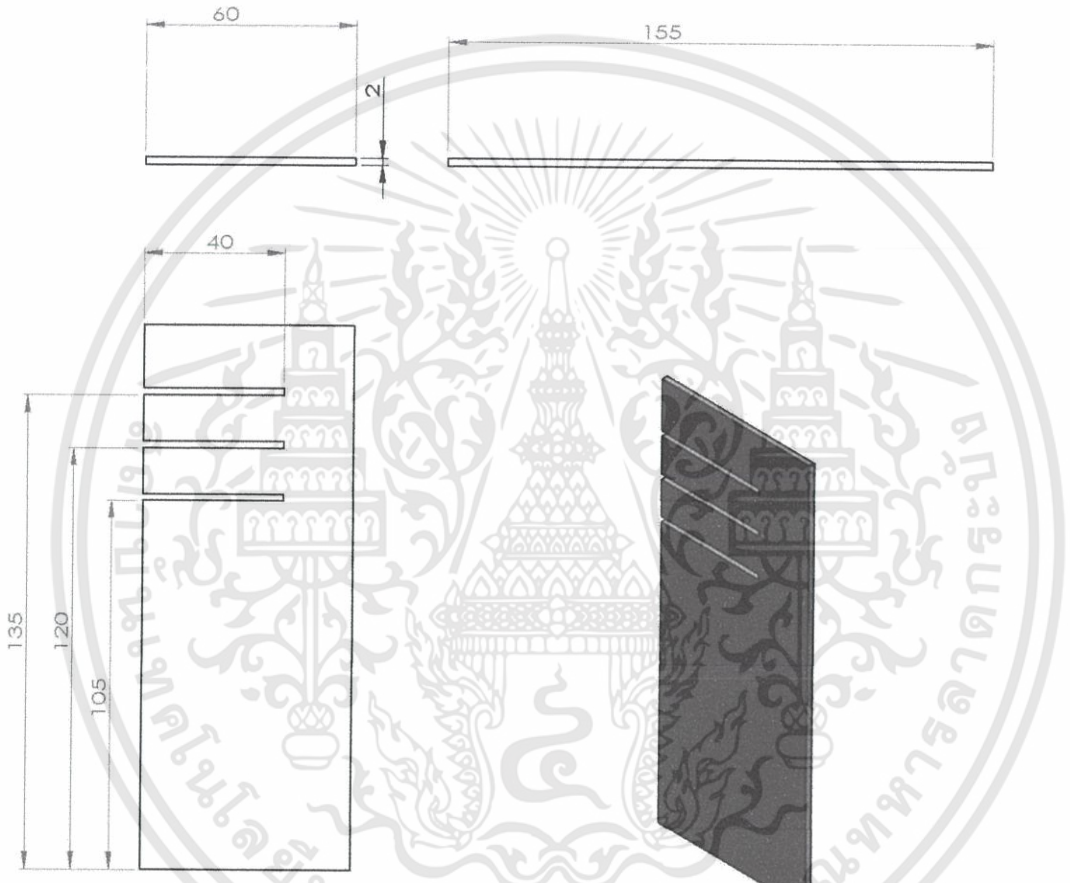
ก.3 ตัวยึดราง (C)

หน่วยเป็นมิลลิเมตร



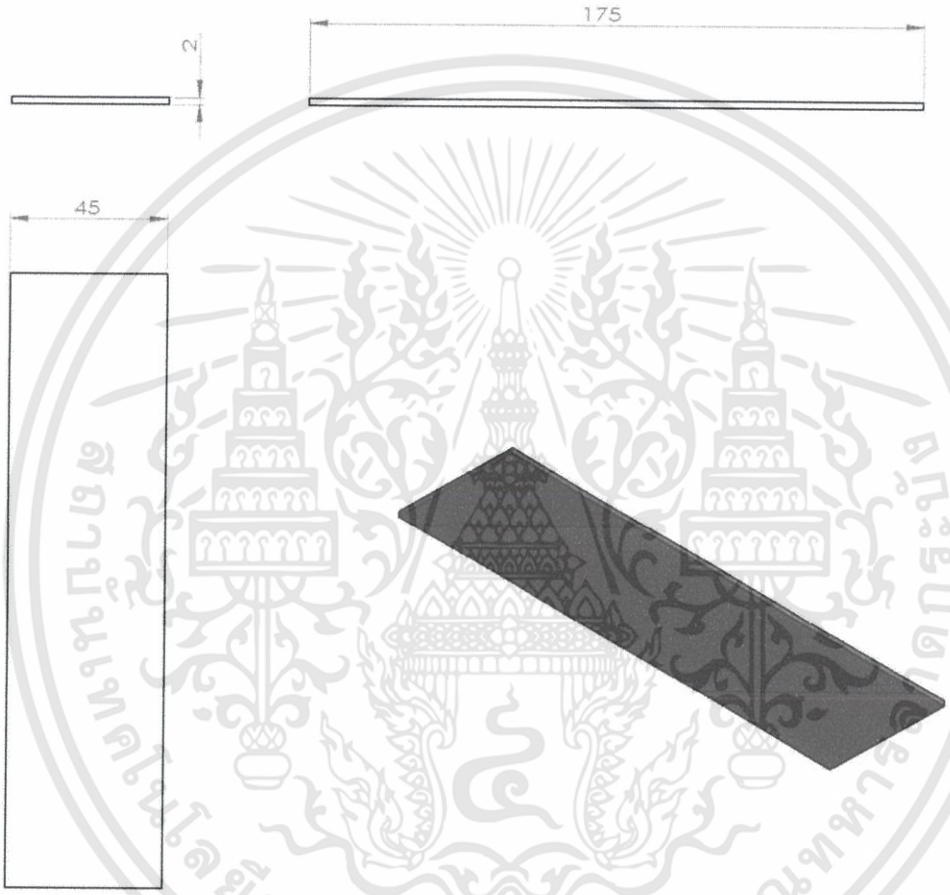
ก.4 ตัวปรับระดับหลังคากั้นสินค้า 6 ชิ้น (D)

หน่วยเป็นมิลลิเมตร



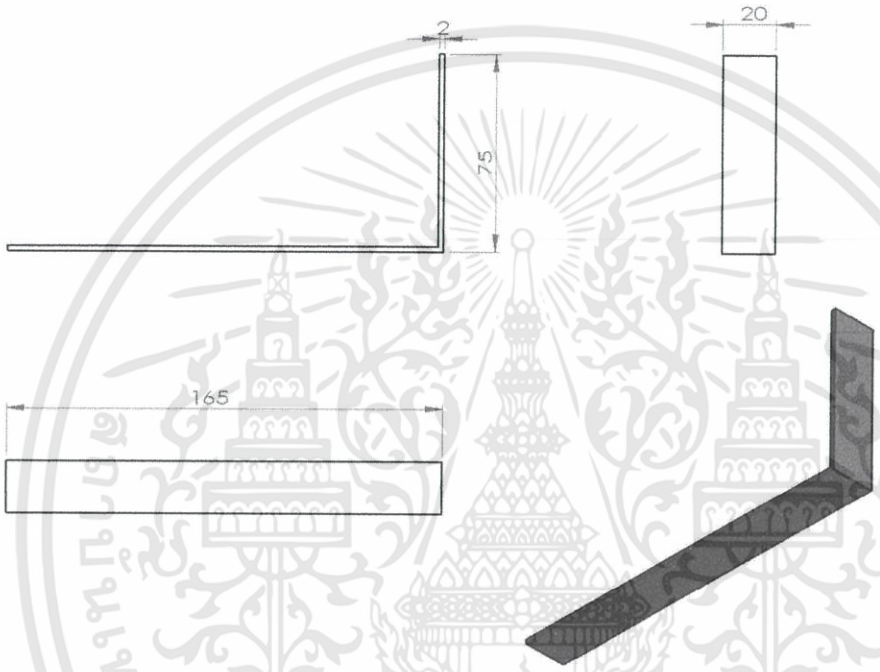
ก.5 หลังคากั้นสินค้าตก 3 ชิ้น (D)

หน่วยเป็นมิลลิเมตร



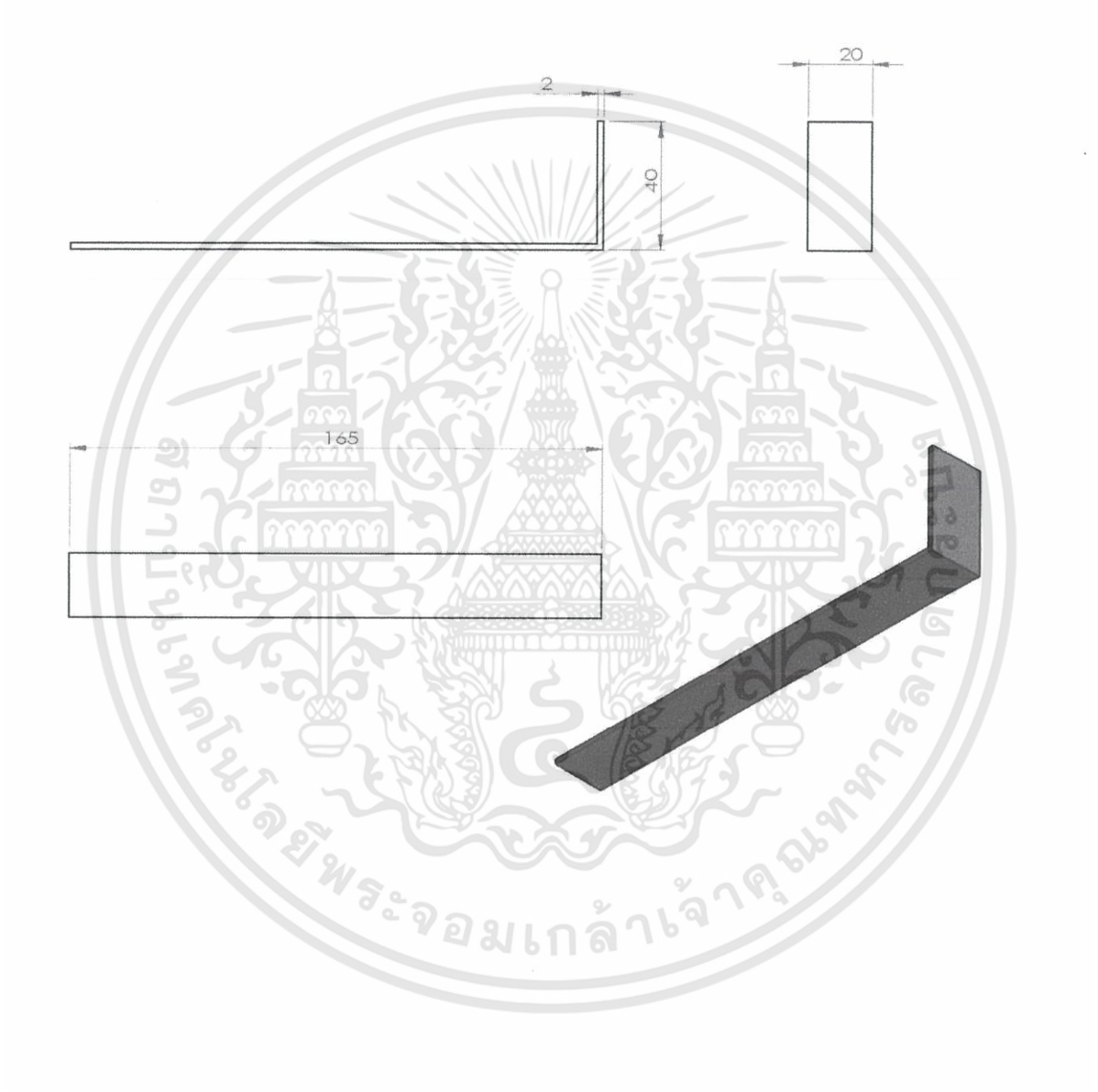
ก.6 ตัวกั้นสินค้าระดับสูง 3 ชั้น (E)

หน่วยเป็นมิลลิเมตร



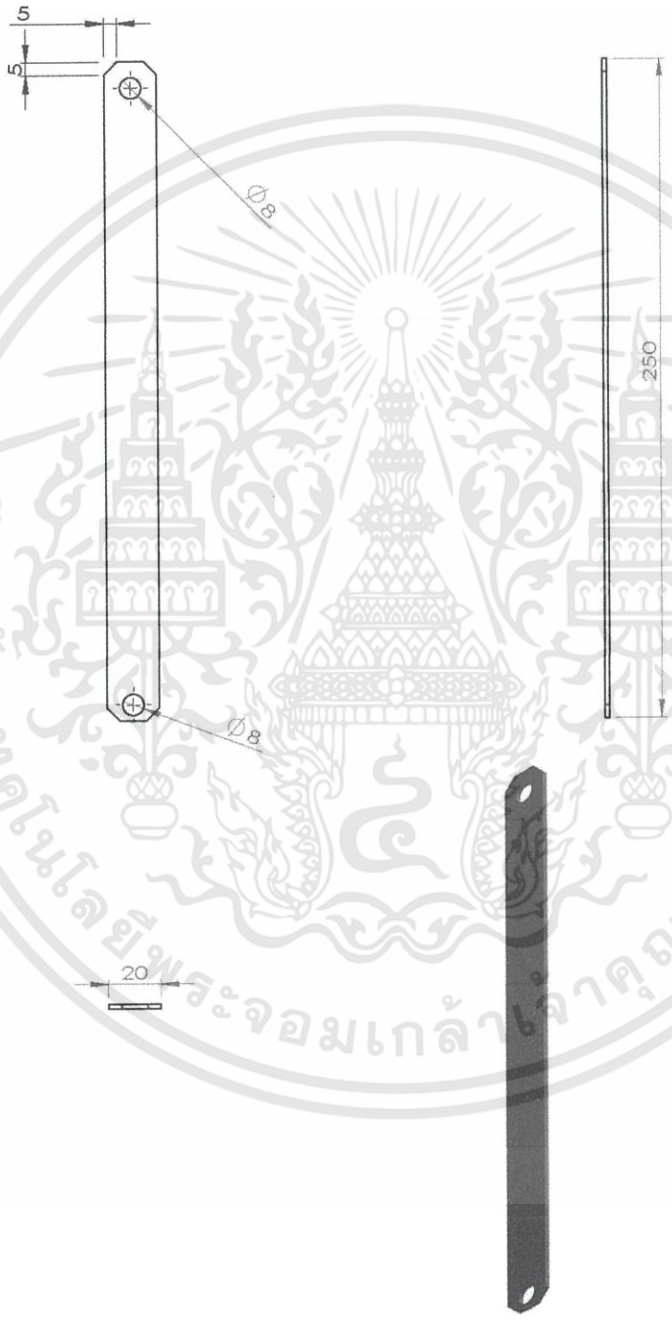
ก.7 ตัวกั้นสินค้าระดับต่ำ 3 ชิ้น (F)

หน่วยเป็นมิลลิเมตร



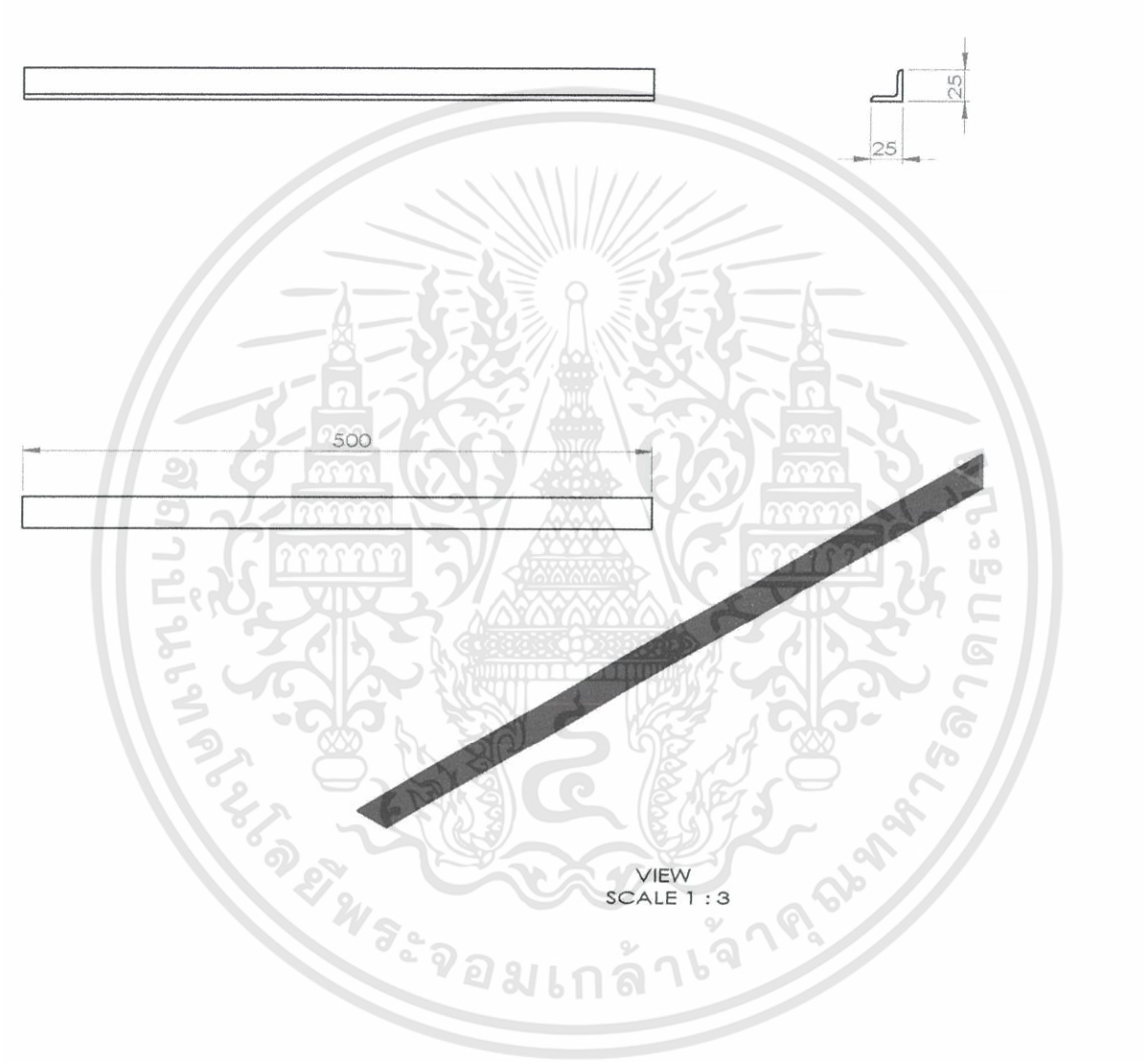
ก.8 ตัวยกราง 3 ชิ้น (H)

หน่วยเป็นมิลลิเมตร



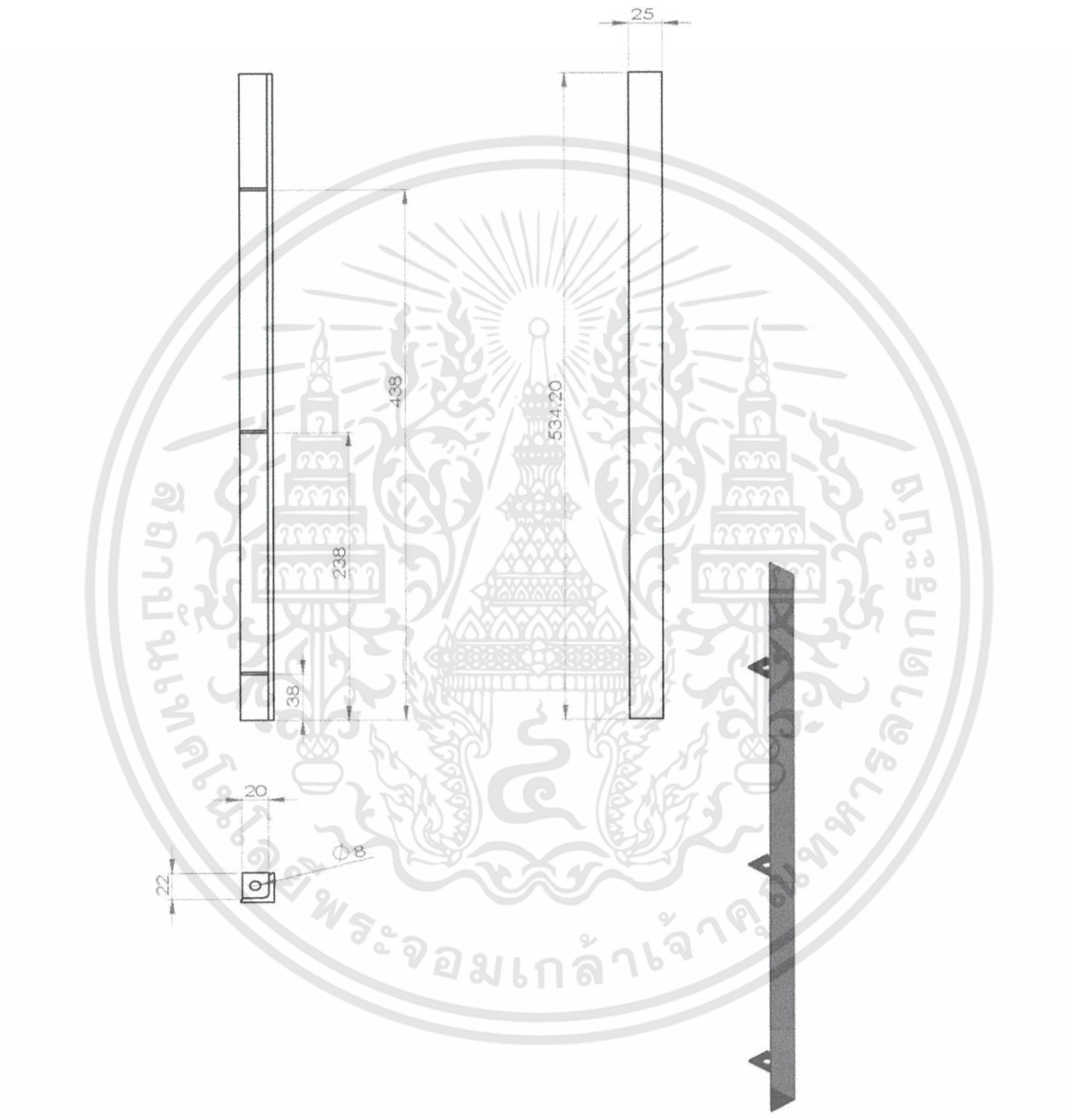
ก.9 ฐานด้านนอก 6 ชิ้น (I)

หน่วยเป็นมิลลิเมตร



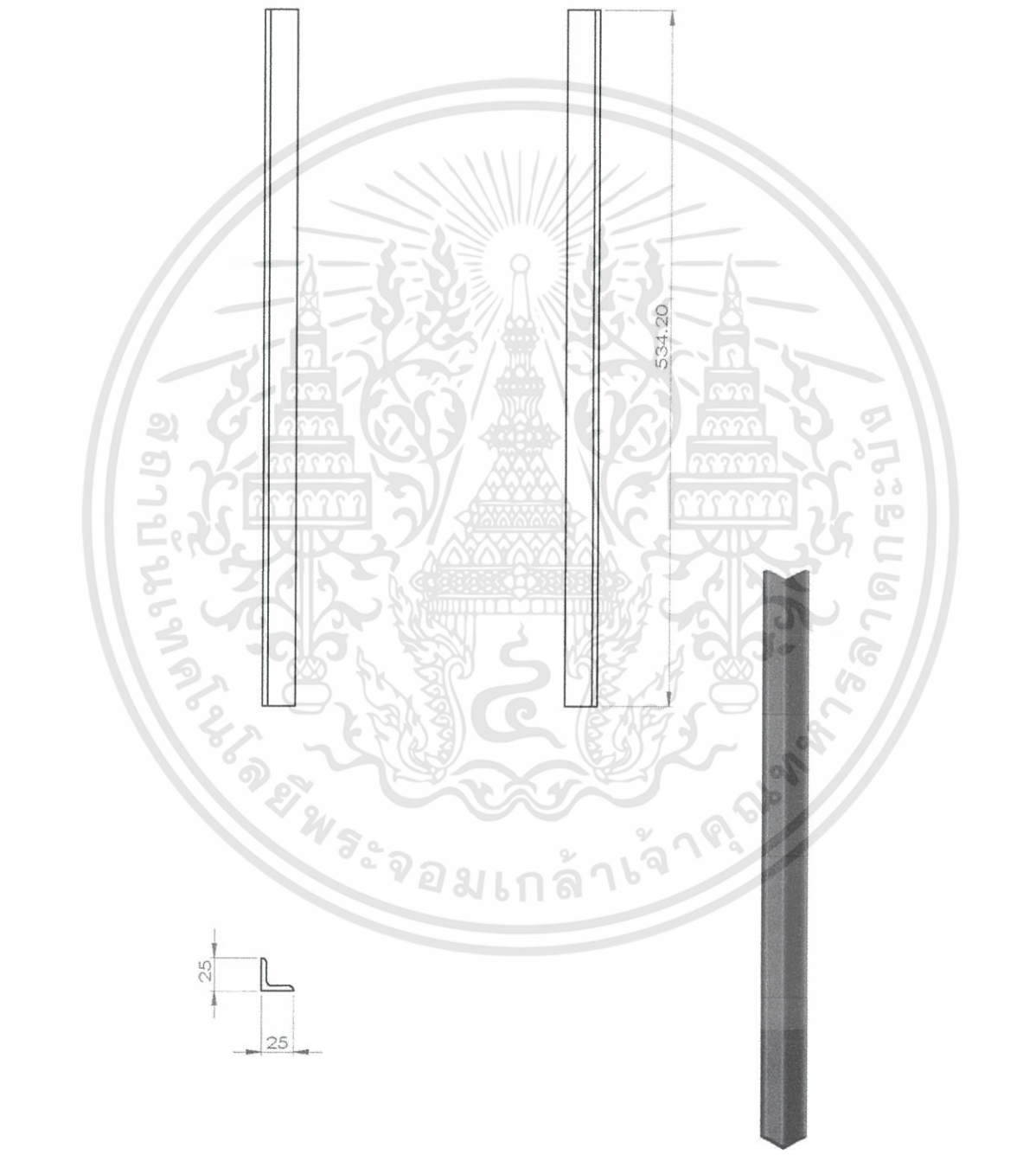
ก.10 ฐานยึด (J)

หน่วยเป็นมิลลิเมตร



ก.11 ฐานยึด 2 ชิ้น (K)

หน่วยเป็นมิลลิเมตร



ภาคผนวก ข
โปรแกรมควบคุม

ข.1 กำหนดตัวแปรและหมายเลขพิน

```
const int timeDelayDownMotor1 = 7000;   //millisec
const int timeDelayDownMotor2 = 7000;   //millisec
const int timeDelayDownMotor3 = 9000;   //millisec
const int btnProduct1 = 0;
const int btnProduct2 = 13;
const int btnProduct3 = 2;
const int btnEnter = 3;
const int mortor1_1= 5;
const int mortor1_2= 4;
const int mortor2_1= 6;
const int mortor2_2= 7;
const int mortor3_1= 8;
const int mortor3_2= 9;
const int sensor1 = 10;
const int sensor2 = 11;
const int sensor3 = 12;

int btnProduct1_state = 0;
int btnProduct2_state = 0;
int btnProduct3_state = 0;
int btnEnter_state = 0;
int sensor1_state = 0;
int sensor2_state = 0;
int sensor3_state = 0;
int clearReq=0;
int checkTimeEnter;
int SetpointProduct1=0;
int SetpointProduct2=0;
int SetpointProduct3=0;
int actionProduct1=0;
int actionProduct2=0;
int actionProduct3=0;
```

```
int pressEnter=0;
int stepProcess=0;
int stock1=6;
int stock2=6;
int stock3=4;
```

ข.2 กำหนดอินพุต เอาต์พุตและการรับค่าจากอุปกรณ์

```
void setup() {

  Serial.begin(9600);
  pinMode(btnProduct1, INPUT);
  pinMode(btnProduct2, INPUT);
  pinMode(btnProduct3, INPUT);
  pinMode(btnEnter, INPUT);
  pinMode(mortor1_1, OUTPUT);
  pinMode(mortor1_2, OUTPUT);
  pinMode(mortor2_1, OUTPUT);
  pinMode(mortor2_2, OUTPUT);
  pinMode(mortor3_1, OUTPUT);
  pinMode(mortor3_2, OUTPUT);
  pinMode(sensor1, INPUT);
  pinMode(sensor2, INPUT);
  pinMode(sensor3, INPUT);
}

void loop()
{
  //Serial.println("====================");
  btnProduct1_state = digitalRead(btnProduct1);

  btnProduct2_state = digitalRead(btnProduct2);
  btnProduct3_state = digitalRead(btnProduct3);
  sensor1_state = digitalRead(sensor1);
  sensor2_state = digitalRead(sensor2);
  sensor3_state = digitalRead(sensor3);
  readEnterBtn();
```

```
if (btnProduct1_state == LOW)
 {
   SetpointProduct1++;
   delay(300);
 }
 if (btnProduct2_state == LOW)
 {
   SetpointProduct2++;
   delay(300);
 }
 if (btnProduct3_state == LOW)
 {
   SetpointProduct3++;
   delay(300);
 }
 //Serial.println(btnEnter_state);
 //Serial.println(pressEnter);
 //Serial.println(stepProcess);
```

ข.3 โปรแกรมสั่งเริ่มทำงาน

```
if (btnEnter_state == HIGH && pressEnter==1)
 {
   Serial.println("====================");
   Serial.print("Order1= ");
   Serial.println(SetpointProduct1);
   Serial.print("Order2 = ");

   Serial.println(SetpointProduct2);
   Serial.print("Order3 = ");
   Serial.println(SetpointProduct3);
  Serial.println("====================");
   Serial.print("Product1 = ");
   Serial.println(actionProduct1);
   Serial.print("Product2 = ");
   Serial.println(actionProduct2);
   Serial.print("Product3 = ");
   Serial.println(actionProduct3);
   Serial.println("====================");
```

```
    Serial.print("Stock1 = ");
    Serial.println(stock1);
     Serial.print("Stock2 = ");
    Serial.println(stock2);
     Serial.print("Stock3 = ");
    Serial.println(stock3);
    pressEnter=0;
    if(stepProcess==0)
    {
      stepProcess = 1;
    }
    delay(800);
  }
  controlProduct1();
  controlProduct2();
  controlProduct3();
  }
```

ข.4 โปรแกรมควบคุมสินค้าชนิดที่ 1

```
void controlProduct1()
{
  int i;

  if(SetpointProduct1>actionProduct1 && stepProcess==1)

  {
    Serial.println("==========control motor1==========");

    if(sensor1_state==LOW)
    {
      stopAllMotor();
      actionProduct1++;
      stock1--;
      Serial.print("Stock1 = ");
    Serial.println(stock1);
      if(stock1==2)
```

```
      {Serial.println("================Warning================");}
      if(stock1==0)
        {Serial.println("================Stop&Reload================");}
    delay(2000);
    if(actionProduct1 % 2 ==0)
    {
      motor1_down();
      delay(timeDelayDownMotor1);
      stopAllMotor();
      delay(1000);
    }
   }
   else
   {
     motor1_up();

   }
  }
  if(SetpointProduct1==actionProduct1 && stepProcess==1)
  {
    stepProcess=2;
  }
}
```

ข.5 โปรแกรมควบคุมสินค้าชนิดที่ 2

```
void controlProduct2()
{
  int i;

  if(SetpointProduct2>actionProduct2 && stepProcess==2)

  {
    Serial.println("==========control motor2==========");

    if(sensor2_state==LOW)
    {
      stopAllMotor();
      actionProduct2++;
```

```
    stock2--;
    Serial.print("Stock2 = ");
  Serial.println(stock2);
    if(stock2==2)
      {Serial.println("================Warning================");}
      if(stock2==0)
        {Serial.println("================Stop&Reload================");}
    delay(2000);
    if(actionProduct2 % 2 ==0)
    {
      motor2_down();
      delay(timeDelayDownMotor2);
      stopAllMotor();
      delay(1000);
    }
  }
  else
  {
    motor2_up();

  }
 }
 if(SetpointProduct2==actionProduct2 && stepProcess==2)
 {
  stepProcess=3;
 }
}
```

ข.6 โปรแกรมควบคุมสินค้าชนิดที่ 3

```
void controlProduct3()
{
 int i;
 if(SetpointProduct3>actionProduct3 && stepProcess==3)
 {
  Serial.println("==========control motor3==========");
  if(sensor3_state==LOW)
  {
   stopAllMotor();
```

```
      actionProduct3++;
      stock3--;
      Serial.print("Stock3 = ");
      Serial.println(stock3);
      if(stock3==2)
        {Serial.println("================Warning================");}

        if(stock3==0)
          {Serial.println("================Stop&Reload================");}
      delay(2000);
        motor3_down();
        delay(timeDelayDownMotor3);
        stopAllMotor();
        delay(1000);
    }
    else
    {
      motor3_up();
    }
  }
  if(SetpointProduct3==actionProduct3 && stepProcess==3)
  {
    stepProcess=0;
  }
}
```

ข.7 โปรแกรมกดปุ่มรีเซ็ต

```
void readEnterBtn()
{
  btnEnter_state = digitalRead(btnEnter);
  if(btnEnter_state == LOW)
  {
    pressEnter=1;
    checkTimeEnter++;
    delay(100);
    if(checkTimeEnter > 10)
    {
      SetpointProduct1=0;
```

```
    SetpointProduct2=0;
    SetpointProduct3=0;

    actionProduct1=0;
    actionProduct2=0;
    actionProduct3=0;
    stepProcess = 0;
    pressEnter=0;
    Serial.println("==========Clear All==========");
    stopAllMotor();
     delay(2000);
    motor1_down();
      delay(timeDelayDownMotor1);
      stopAllMotor();
    motor2_down();
      delay(timeDelayDownMotor2);
      stopAllMotor();
    motor3_down();
      delay(timeDelayDownMotor3);
      stopAllMotor();
    Serial.println("==========Clear All==========");
   }
  }
  else
  {
    checkTimeEnter=0;
  }
}
```

ข.8 โปรแกรมสั่งให้มอเตอร์หยุด

```
void stopAllMotor()
{
  Serial.println("stopAllMotor");
   digitalWrite(mortor1_1,LOW);
   digitalWrite(mortor1_2,LOW);
   digitalWrite(mortor2_1,LOW);

   digitalWrite(mortor2_2,LOW);
   digitalWrite(mortor3_1,LOW);
```

```
  digitalWrite(mortor3_2,LOW);
}
```

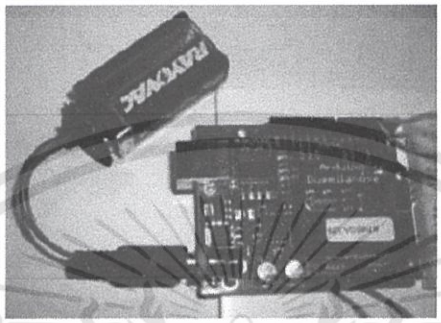
ข.9 โปรแกรมควบคุมมอเตอร์

```
void motor1_up()
{
  Serial.println("Motor1_up");
  digitalWrite(mortor1_1,HIGH);
  digitalWrite(mortor1_2,LOW);
}
void motor1_down()
{
  Serial.println("Motor1_down");
  digitalWrite(mortor1_1,LOW);
  digitalWrite(mortor1_2,HIGH);
}
void motor2_up()
{
  Serial.println("Motor2_up");
  digitalWrite(mortor2_1,HIGH);
  digitalWrite(mortor2_2,LOW);
}
void motor2_down()
{
  Serial.println("Motor2_down");
  digitalWrite(mortor2_1,LOW);
  digitalWrite(mortor2_2,HIGH);
}
void motor3_up()
{
  Serial.println("Motor3_up");

  digitalWrite(mortor3_1,HIGH);
  digitalWrite(mortor3_2,LOW);
}
void motor3_down()
{
  Serial.println("Motor3_down");
```

```
  digitalWrite(mortor3_1,LOW);
  digitalWrite(mortor3_2,HIGH);
}
```

ภาคผนวก ค

เอกสารคู่มือ



Warning: Watch the polarity as you connect your battery to the snap as reverse orientation could blow out your board.

Disconnect your Arduino from the computer. Connect a 9 V battery to the Arduino power jack using the battery snap adapter. Confirm that the blinking program runs. This shows that you can power the Arduino from a battery and that the program you download runs without needing a connection to the host PC

1.5 Moving On

Connect your Arduino to the computer with the USB cable. You do not need the battery for now. The green PWR LED will light. If there was already a program burned into the Arduino, it will run.

Warning: Do not put your board down on a conductive surface; you will short out the pins on the back!

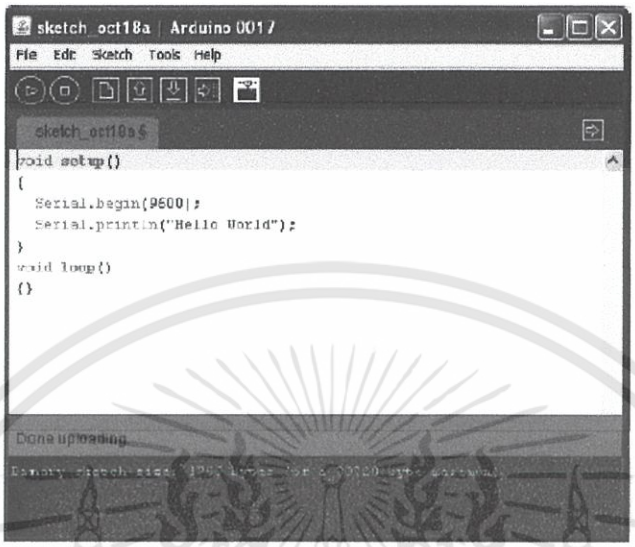
Start the Arduino development environment. In Arduino-speak, programs are called "sketches", but here we will just call them programs.

In the editing window that comes up, enter the following program, paying attention to where semi-colons appear at the end of command lines.

```
void setup()
{
  Serial.begin(9600);
  Serial.println("Hello World");
}
void loop()
{}
```

Your window will look something like this

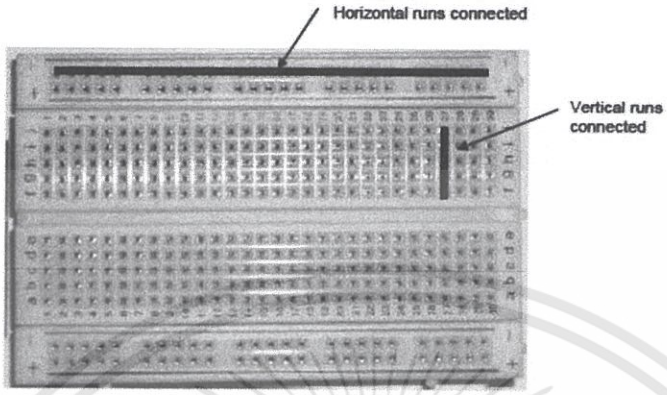
3



Click the Upload button  or Ctrl-U to compile the program and load on the Arduino board.

Click the Serial Monitor button  If all has gone well, the monitor window will show your message and look something like this

Congratulations; you have created and run your first Arduino program!



Warning: Only use solid wire on the breadboard. Strands of stranded wire can break off and fill the holes permanently.

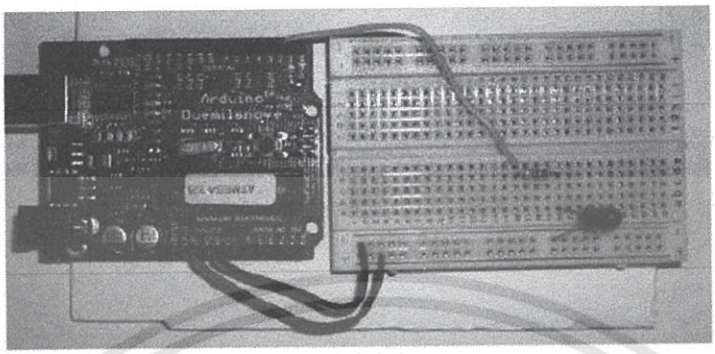
Hint: Trim wires and component leads so that wires and components lie close to the board.

To keep the Arduino board and breadboard together, you can secure both to a piece of fom-core, cardboard or wood using double-stick foam tape or other means.

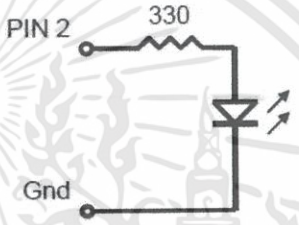
2 Flashing an LED

Light emitting diodes (LED's) are handy for checking out what the Arduino can do.. For this task, you need an LED, a 330 ohm resistor, and some short pieces of 22 or 24 g wire. The figure to the right is a sketch of an LED and its symbol used in electronic schematics

Using 22 g solid wire, connect the 5V power pin on the Arduino to the bottom red power bus on the breadboard and the Gnd pin on the Arduino to the bottom blue power buss on the breadboard. Connect the notched or flat side of the LED (the notch or flat is on the rim that surrounds the LED base; look carefully because it can be hard to find) to the Gnd bus and the other side to a free hole in main area of the breadboard. Place the resistor so that one end is in the same column as the LED and the other end is in a free column. From that column, connect a wire to digital pin 2 on the Arduino board. Your setup will look something like this



To test whether the LED works, temporarily disconnect the wire from pin 2 on the Arduino board and touch to the 5V power bus. The LED should light up. If not, try changing the orientation of the LED. Place the wire back in pin 2.
On the LED, current runs from the anode (+) to the cathode (-) which is marked by the notch. The circuit you just wired up is represented in schematic form in the figure to the right.



Create and run this Arduino program

```
void setup()
{
  pinMode(2,OUTPUT);

  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,LOW);
}

void loop()
{}
```

Did the LED light up for one second? Push the Arduino reset button to run the program again.

Now try this program, which will flash the LED at 1.0 Hz. Everything after the // on a line is a comment, as is the text between '/*' and '*/' at the top. It is always good to add comments to a program.

```
/*-----------------------------
Blinking LED, 1.0 Hz on pin 2
-----------------------------*/
```

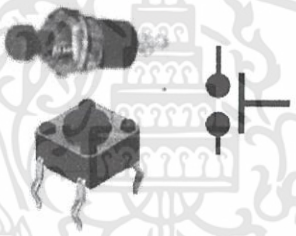
```
void setup()            // one-time actions
{
  pinMode(2,OUTPUT);    // define pin 2 as an output
}

void loop()              // loop forever
{
  digitalWrite(2,HIGH); // pin 2 high (LED on)
  delay(500);           // wait 500 ms
  digitalWrite(2,LOW);  // pin 2 low (LED off)
  delay(500);           // wait 500 ms
}
```

The pinMode command sets the LED pin to be an output. The first digitalWrite command says to set pin 2 of the Arduino to HIGH, or +5 volts. This sends current from the pin, through the resistor, through the LED (which lights it) and to ground. The delay(500) command waits for 500 msec. The second digitalWrite command sets pin 2 to LOW or 0 V stopping the current thereby turning the LED off. Code within the brackets defining the loop() function is repeated forever, which is why the LED blinks.

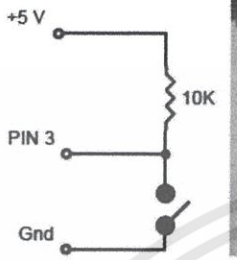
This exercise shows how the Arduino can control the outside world. With proper interface circuitry the same code can turn on and off motors, relays, solenoids, electromagnets, pneumatic valves or any other on-off type device.

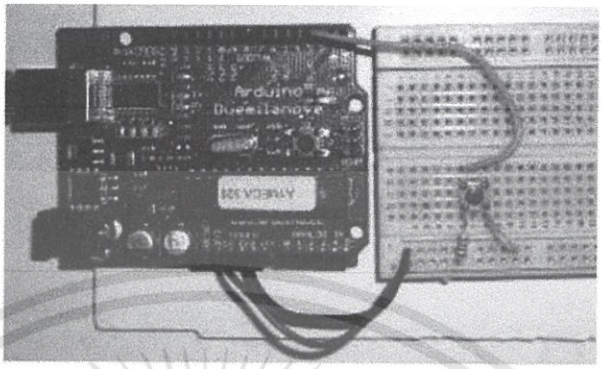
3 Reading a switch

The LED exercise shows how the Arduino can control the outside world. Many applications require reading the state of sensors, including switches. The figure to the right shows a picture of a pushbutton switch and its schematic symbol. Note that the symbol represents a switch whose contacts are normally open, but then are shorted when the button is pushed. If you have a switch, use the continuity (beeper) function of a digital multi-meter (DMM) to understand when the leads are open and when they are connected as the button is pushed.



For this exercise, the Arduino will read the state of a normally-open push button switch and display the results on the PC using the serial.println() command. You will need a switch, a 10 kohm resistor and some pieces of 22 g hookup wire. If you don't have a switch, substitute two wires and manually connect their free ends to simulate a switch closure. The figure below shows the schematic for the circuit on the left and a realization on the right.





Create and run this Arduino program

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println(digitalRead(3));
  delay(250);
}
```

Open the Serial Monitor window. When the switch is open, you should see a train of 1's on the screen. When closed, the 1's change to 0's. On the hardware side, when the switch is open, no current flows through the resistor. When no current flows through a resistor, there is no voltage drop across the resistor, which means the voltage on each side is the same. In your circuit, when the switch is open, pin 3 is at 5 volts which the computer reads as a 1 state. When the switch is closed, pin 3 is directly connected to ground, which is at 0 volts. The computer reads this as a 0 state.

Now try this program which is an example of how you can have the computer sit and wait for a sensor to change state.

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  while (digitalRead(3) == HIGH)
    ;
  Serial.println("Somebody closed the switch!");
```

```
  while (digitalRead(3) == LOW)
    ;
  Serial.println("The switch is now open!");
}
```

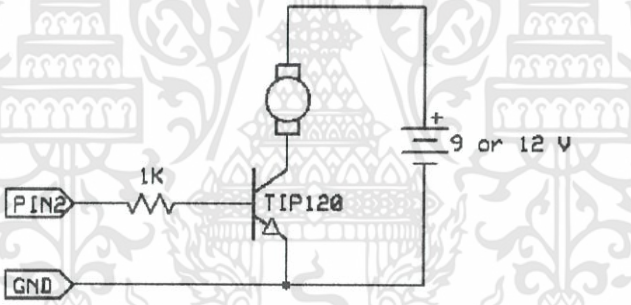
Watch the activity in the Serial Monitor window as you press and release the switch.

4 Controlling a Small DC Motor

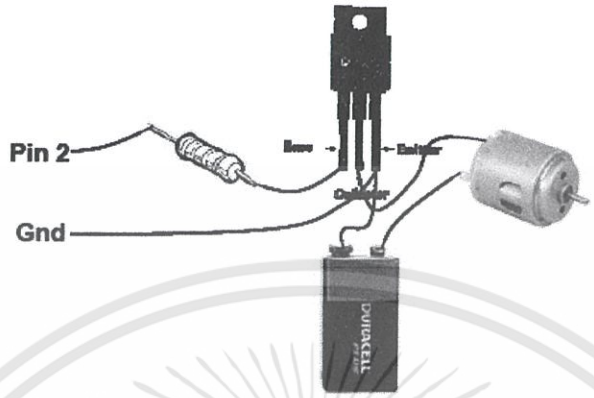
The Arduino can control a small DC motor through a transistor switch. You will need a TIP120 transistor, a 1K resistor a 9V battery with battery snap and a motor.

The TIP120 pins look like this and on a schematic the pins are like this

Here is the schematic diagram for how to connect the motor



And here is a pictorial diagram for how to connect the components. The connections can be soldered or they can be made through a solderless breadboard.



Pin 2 can be any digital I/O pin on your Arduino. Connect the minus of the battery to the emitter of the transistor (E pin) and also connect the emitter of the transistor to Gnd on the Arduino board.

To check if things are working, take a jumper wire and short the collector to the emitter pins of the transistor. The motor should turn on. Next, disconnect the 1K resistor from pin 2 and jumper it to +5V. The motor should turn on. Put the resistor back into pin 2 and run the following test program:

```
void setup()
{
  pinMode(2,OUTPUT);
  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,LOW);
}

void loop()
{}
```

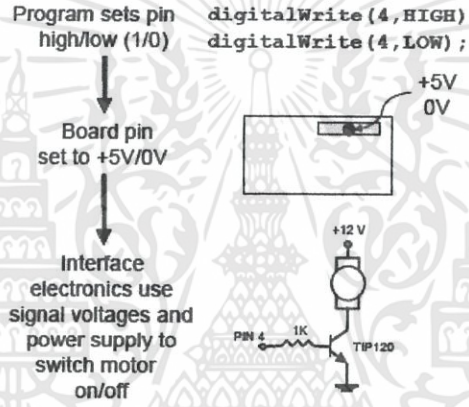
The motor should turn on for 1 second.

5 Arduino Hardware

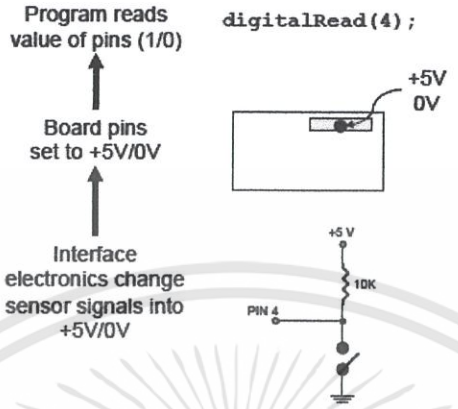
The power of the Arduino is not its ability to crunch code, but rather its ability to interact with the outside world through its input-output (I/O) pins. The Arduino has 14 digital I/O pins labeled 0 to 13 that can be used to turn motors and lights on and off and read the state of switches.

Each digital pin can sink or source about 40 mA of current. This is more than adequate for interfacing to most devices, but does mean that interface circuits are needed to control devices other than simple LED's. In other words, you cannot run a motor directly using the current available from an Arduino pin, but rather must have the pin drive an interface circuit that in turn drives the motor. A later section of this document shows how to interface to a small motor.

To interact with the outside world, the program sets digital pins to a high or low value using C code instructions, which corresponds to +5 V or 0 V at the pin. The pin is connected to external interface electronics and then to the device being switched on and off. The sequence of events is shown in this figure.



To determine the state of switches and other sensors, the Arduino is able to read the voltage value applied to its pins as a binary number. The interface circuitry translates the sensor signal into a 0 or +5 V signal applied to the digital I/O pin. Through a program command, the Ardiomp interrogates the state of the pin. If the pin is at 0 V, the program will read it as a 0 or LOW. If it is at +5 V, the program will read it as a 1 or HIGH. If more than +5 V is applied, you may blow out your board, so be careful. The sequence of events to read a pin is shown in this figure.



Interacting with the world has two sides. First, the designer must create electronic interface circuits that allow motors and other devices to be controlled by a low (1-10 mA) current signal that switches between 0 and 5 V, and other circuits that convert sensor readings into a switched 0 or 5 V signal. Second, the designer must write a program using the set of Arduino commands that set and read the I/O pins. Examples of both can be found in the Arduino resources section of the ME2011 web site.

When reading inputs, pins must have either 0 or 5V applied. If a pin is left open or "floating", it will read random voltages and cause erratic results. This is why switches always have a 10K pull up resistor connected when interfacing to an Arduino pin.

Note: The reason to avoid using pins 0 and 1 is because those pins are used for the serial communications between the Arduino and the host computer.

The Arduino also has six analog input pins for reading continuous voltages in the range of 0 to 5 V from sensors such as potentiometers.

6 Programming Concepts

This chapter covers some basic concepts of computer programming, going under the assumption that the reader is a complete novice.

A computer program is a sequence of step-by-step instructions for the computer to follow. The computer will do exactly what you tell it to do, no more no less. The computer only knows what's in the program, not what you intended. Thus the origin of the phrase, "Garbage in, garbage out".

7.4 Math

The Arduino can do standard mathematical operations. While floating point (e.g. 23.2) numbers are allowed if declared as floats, operations on floats are very slow so integer variables and integer math is recommended. If you have byte variables, no number, nor the result of any math operation can fall outside the range of 0 to 255. You can divide numbers, but the result will be truncated (not rounded) to the nearest integer. Thus in integer arithmetic, 17/3 = 5, and not 5.666 and not 6. Math operations are performed strictly in a left-to-right order. You can add parenthesis to group operations.

The table below shows some of the valid math operators. Full details of their use can be found in the Arduino Language Reference.

Symbol	Description
+	addition
-	subtraction
*	multiplication
/	division
%	modulus (division remainder)
<<	left bit shift
>>	right bit shift
&	bitwise AND
|	bitwise OR

8 The Simple Commands

This section covers the small set of commands you need to make the Arduino do something useful. These commands appear in order of priority. You can make a great machine using only digital read, digital write and delay commands. Learning all the commands here will take you to the next level.

If you need more, consult the Arduino language reference page at *h*
http://arduino.cc/en/Reference/HomePage.

pinMode

This command, which goes in the setup() function, is used to set the direction of a digital I/O pin. Set the pin to OUTPUT if the pin is driving and LED, motor or other device. Set the pin to INPUT if the pin is reading a switch or other sensor. On power up or reset, all pins default to inputs. This example sets pin 2 to an output and pin 3 to an input.

```
void setup()
{
  pinMode(2,OUTPUT);
  pinMode(3,INPUT);
}
void loop() {}
```

Serial.print

The Serial.print command lets you see what's going on inside the Arduino from your computer. For example, you can see the result of a math operation to determine if you are getting the right number. Or, you can see the state of a digital input pin to see if the Arduino is a sensor or switch properly. When your interface circuits or program does not seem to be working, use the Serial.print command to shed a little light on the situation. For this command to show anything, you need to have the Arduino connected to the host computer with the USB cable.

For the command to work, the command Serial.begin(9600) must be placed in the setup() function. After the program is uploaded, you must open the Serial Monitor window to see the response.

There are two forms of the print command. Serial.print() prints on the same line while Serial.println() starts the print on a new line.

Here is a brief program to check if your board is alive and connected to the PC

```
void setup()
{
  Serial.begin(9600);
  Serial.println("Hello World");
}
void loop() {}
```

Here is a program that loops in place, displaying the value of an I/O pin. This is useful for checking the state of sensors or switches and to see if the Arduino is reading the sensor properly. Try it out on your Arduino. After uploading the program, use a jumper wire to alternately connect pin 2 to +5V and to Gnd.

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.println(digitalRead(2));
  delay(100);
}
```

If you wanted to see the states of pins 2 and 3 at the same time, you can chain a few print commands, noting that the last command is a println to start a new line.

```
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  Serial.print("pin 2 = ");
  Serial.print(digitalRead(2));
  Serial.print("   pin 3 = ");
  Serial.println(digitalRead(3));
```

```
  delay(100);
}
```

You may have noticed when trying this out that if you leave one of the pins disconnected, its state follows the other. This is because a pin left floating has an undefined value and will wander from high to low. So, use two jumper wires when trying out this example.

Here's one that checks the value of a variable after an addition. Because the calculation is done just once, all the code is in the setup() function. The Serial.flush()

```
int i,j,k;
void setup()
{
  Serial.begin(9600);
  i=21;
  j=20;
  k=i+j;
  Serial.flush();
  Serial.print(k);
}
void loop() {}
```

digitalWrite

This command sets an I/O pin high (+5V) or low (0V) and is the workhorse for commanding the outside world of lights, motors, and anything else interfaced to your board. Use the pinMode() command in the setup() function to set the pin to an output.

```
digitalWrite(2,HIGH);  // sets pin 2 to +5 volts
digitalWrite(2,LOW);   // sets pin 2 to zero volts
```

delay

Delay pauses the program for a specified number of milliseconds. Since most interactions with the world involve timing, this is an essential instruction. The delay can be for 0 to 4,294,967,295 msec. This code snippet turn on pin 2 for 1 second.

```
digitalWrite(2,HIGH);  // pin 2 high (LED on)
delay(1000);           // wait 500 ms
digitalWrite(2,LOW);   // pin 2 low (LED off)
```

if

This is the basic conditional branch instruction that allows your program to do two different things depending on whether a specified condition is true or false.

Here is one way to have your program wait in place until a switch is closed. Connect a switch to pin 3 as shown in Section 3. Upload this program then try closing the switch

```
void setup()
```

```
{
  Serial.begin(9600);
}

void loop()
{
  if (digitalRead(3) == LOW) {
      Serial.println("Somebody closed the switch!");
  }
}
```

The if line reads the state of pin 3. If it is high, which it will be for this circuit when the switch is open, the code jumps over the Serial.println command and will repeat the loop. When you close the switch, 0V is applied to pin 3 and its state is now LOW. This means the if condition is true so this time around the code between the braces is executed and the message is printed
The syntax for the if statement is

```
if (condition) {
   //commands
}
```

If the condition is true, the program will execute the commands between the braces. If the condition is not true, the program will skip to the statement following the braces.

The condition compares one thing to another. In the example above, the state of pin 1 was compared to LOW with ==, the equality condition. Other conditional operators are != (not equal to), > (greater than), < (less than), >= (greater than or equal to), and <= (less than or equal to).

You can have the program branch depending on the value of a variable. For example, this program will print the value of i only when it is less than 30.

```
int i;

void setup()
{
  Serial.begin(9600);
  i=0;
}

void loop()
{
  i=i+1;
  if (i<30) {
      Serial.println(i);
  }
}
```

for

Arduino Programming Basics

Command	Description
`pinMode(n,INPUT)`	Set pin *n* to act as an input. One-time command at top of program.
`pinMode(n,OUTPUT)`	Set pin *n* to act as an output
`digitalWrite(n,HIGH)`	Set pin *n* to 5V
`digitalWrite(n,LOW)`	Set pin *n* to 0V
`delay(x)`	Pause program for *x* millisec, *x* = 0 to 65,535
`tone(n,f,d)`	Play tone of frequency *f* Hz for *d* millisec on speaker attached to pin *n*
`for()`	Loop. Example: for (i=0;i<3;i++){} Do the instructions enclosed by {} three times
`if (expr) {}`	Conditional branch. If *expr* true, do instructions enclosed by {}
`while (expr) {}`	While *expr* is true, repeat instructions in {} indefinitely

For more commands see the ME2011 "Arduino Microcontroller Guide" and the Language Reference section of the arduino web site.

Instructions in the setup() function are executed once. Those in the loop() function are executed indefinitely.

Examples

1. Turn on LED connected to Pin 2 for 1 s.

```
void setup() {
  pinMode(2,OUTPUT);
  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,LOW);
}
void loop()
{}
```

2. Flash LED connected to Pin 2 at 1 Hz forever.

```
void setup()  {
  pinMode(2,OUTPUT);
}
void loop()   {
  digitalWrite(2,HIGH);
  delay(500);
  digitalWrite(2,LOW);
  delay(500);
}
```

3. Turn on motor connected to Pin 4 for 1 s.

```
void setup() {
  pinMode(4,OUTPUT);
  digitalWrite(4,HIGH);
  delay(1000);
  digitalWrite(4,LOW);
}
void loop()
{}
```

4. Play 440 hz tone for one second on speaker connected to pin 5. Delay is needed because the program does not wait for the tone() command to finish but rather immediately goes to the command following tone().

```
void setup() {
  pinMode(5,OUTPUT);
  tone(5,440,1000);
  delay(1100);
}
void loop()
{}
```

5. LED is on Pin 2 and switch is on Pin 6. Turns on the LED for one sec when switch is pressed.

```
void setup() {
  pinMode(2,OUTPUT);
  pinMode(6,INPUT);
  while (digitalRead(6) == HIGH)
    ;
  digitalWrite(2,HIGH);
  delay(1000);
  digitalWrite(2,LOW);

}
void loop()
{}
```