ระบบควบคุมการเลือกชั้นลิฟต์โดยใช้อาร์เอฟไอดี

โดย

วรุตม์ บุญเลี่ยม

พรประสิทธิ์ บุญทอง

อาทิตย์ อยู่เย็น

สนับสนุนงบประมาณโดย

มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

ประจำปีงบประมาณ 2557

The evaluator access control using RFID

By

WAROOT BOONLIAM

PORNPRASIT BOONTONG

ARTHIT YOOYAN

Granted by

Rajamangala University of Technology Rattanakosin

Fiscal year 2014

กิตติกรรมประกาศ

วิจัยเรื่องระบบควบคุมการเลือกชั้นลิฟต์โดยใช้อาร์เอฟไอดี นี้สำเร็จลุล่วงได้ ทั้งนี้ได้รับความช่วยเหลือจาก นายศรายุทธ นันทสมบูรณ ประกอบธุรกิจส่วนตัวด้านลิฟต์ ซึ่งเป็นที่ปรึกษาของงานวิจัย ที่ได้เสียสละเวลาให้คำปรึกษา และแนะนำวิธีต่างๆ เพื่อให้โครงการสำเร็จลุล่วงไปได้ด้วยดี

สุดท้ายนี้ขอขอบพระคุณ มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์ วิทยาเขตวังไกลกังวล ที่สนับสนุนงบประมาณในการทำวิจัยจนเสร็จสมบูรณ์ได้ในครั้งนี้

นายวรุตม์ บุญเลี่ยม และคณะ

พฤศจิกายน 2557

บทคัดย่อ

รหัสโครงการ : Inno 003/2557

ชื่อโครงการ : ระบบควบคุมการเลือกชั้นลิฟต์โดยใช้อาร์เอฟไอดี

ชื่อนักวิจัย : นายวรุตม์ บุญเลี่ยม, นายพรประสิทธิ์ บุญทอง, นายอาทิตย์ อยู่เย็น

งานวิจัยนี้มีวัตถุประสงค์เพื่อศึกษาและนำเทคโนโลยี อาร์เอฟไอดี และบอร์ด Arduino Mega 2560 มาประยุกต์ใช้ควบคุมการเลือกชั้นลิฟต์ ด้วยบัตร อาร์เอฟไอดี ตามชั้นต่างๆ ของอาคารที่ได้กำหนดสิทธิ์ให้กับบัตร โดยระบบดังกล่าวสามารถกำหนดสิทธิ์ให้บัตรในการเลือกชั้นลิฟต์ที่ต้องการนั้นๆได้โดยเฉพาะ เพื่อความเป็นส่วนตัวและความสะดวกสำหรับผู้ใช้งาน ซึ่งในระบบดังกล่าวนี้ เหมาะสำหรับอาคารขนาดกลาง และขนาดใหญ่ ซึ่งสามารถทำเป็นระบบที่เพิ่มความเป็นส่วนตัว และความปลอดภัยให้กับผู้ใช้งาน งานวิจัยนี้เป็นการนำ ตัวอ่านอาร์เอฟไอดี มาเป็นตัวกำหนดสิทธิ์การเลือกชั้นต่างๆที่กำหนดไว้ เพื่อความปลอดภัยในระบบชั้นต่างๆหรือตัวอาคาร โดยมีตัวบอร์ด Arduino Mega 2560 คอยเก็บข้อมูลของบัตร อาร์เอฟไอดี และสิทธิในการเข้าถึงชั้นลิฟต์ เพื่อสั่งให้ขึ้นไปยังชั้นที่บัตร อาร์เอฟไอดี กำหนดสิทธิ์ไว้

E-mail Address : waroot.boo@rmutr.ac.th

ระยะเวลาโครงการ : ตุลาคม 2556 – กันยายน 2557

Abstract

Code of project : Inno 003/2557

Project name : The evaluator access control using RFID

Researcher name : Waroot Boonliam, Pornprasit Boontong, Arthit Yooyen

The RFID project selection controls the elevator. Aims to study the technology and RFID Arduino Mega 2560 board used elevator control with RFID card of different floor of the building is set right on the card. The system can use the card to set up the right to be particular. For privacy and convenience for users. In such a system, this Ideal for medium and large buildings, which can be added to a system. And security for users. This thesis is the introduction of RFID Tag is set to a different class defined. Safety systems in the building or floor. Arduino Mega 2560 board with the help of RFID card data to order up to level RFID cards assigned to them

E-mail Address : waroot.boo@rmutr.ac.th

Period of Project : October 2013 – September 2014

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

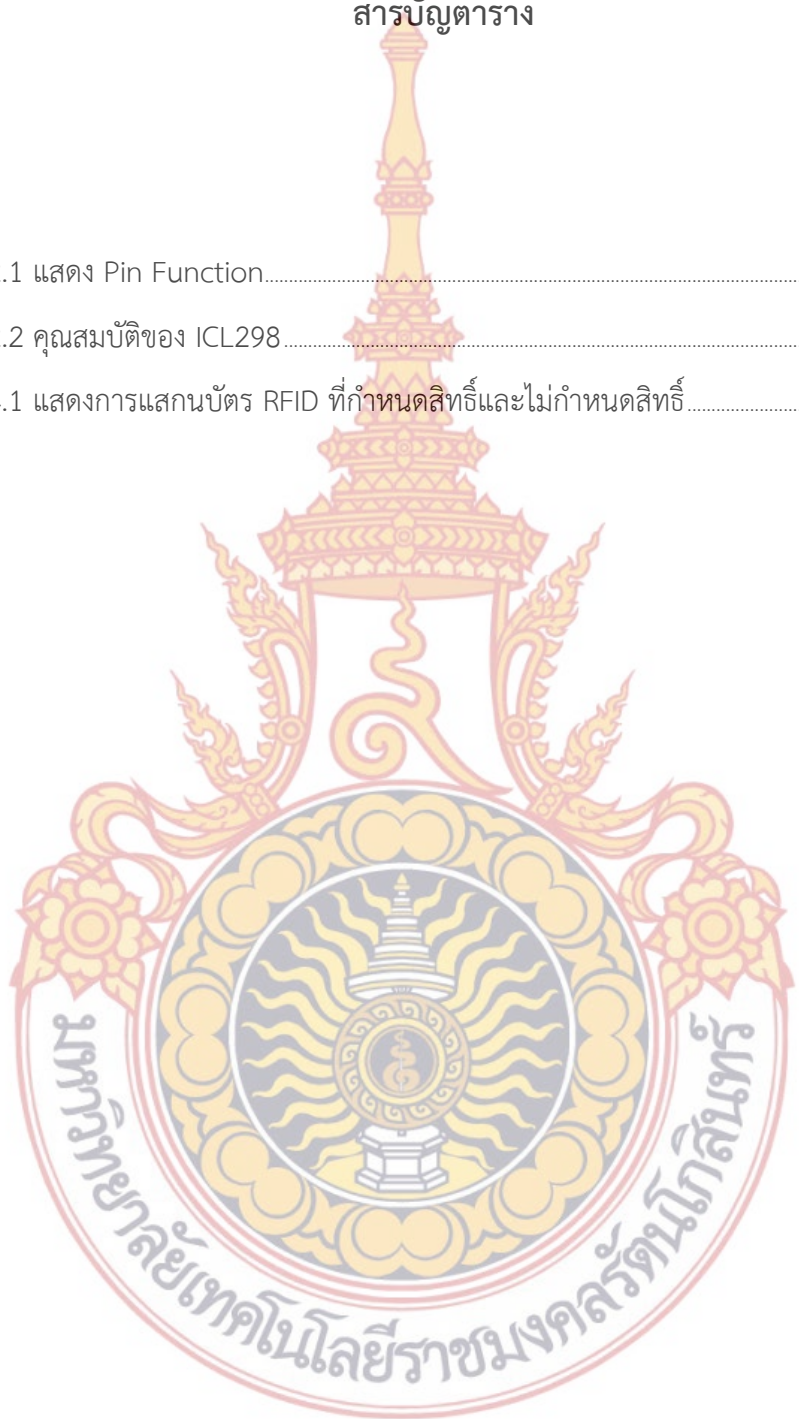
ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญตาราง

ตารางที่ **หน้า**



บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

ปัจจุบันเทคโนโลยี RFID (Radio Frequency Identification) เป็นเทคโนโลยีที่กำลังได้รับความสนใจอย่างมากทั่วโลกในขณะนี้ ด้วยเชื่อว่าจะเป็นเทคโนโลยีที่ส่งผลต่อการดำเนินชีวิตประจำวันและการดำเนินธุรกิจ เพราะเทคโนโลยีดังกล่าวสามารถนำไปประยุกต์ใช้งานได้หลากหลาย ไม่ว่าจะเป็นในระบบค้าปลีก ค้าส่ง การผลิต จนกระทั่งการบริหารจัดการ Supply Chain และระบบ Logistic ตลอดจนระบบรักษาความปลอดภัย (Security & Access Control)

การนำเทคโนโลยี RFID มาใช้งานใน Application ที่เป็น Access Control นั้น เป็นการนำไปใช้กับการควบคุมการเข้า-ออกอาคาร สถานที่ต่างๆ และยังสามารถนำไปประยุกต์ใช้กับการตรวจสอบเวลาทำงานของฝ่ายทรัพยากรบุคคลขององค์กร รวมทั้ง Log In เข้าใช้เครื่องคอมพิวเตอร์ ทั้งยังมีการนำเทคโนโลยี RFID มาใช้ในย่านความถี่ที่แตกต่างกันออกไปตามลักษณะของการนำมาสนับสนุนการใช้งานใน Application ในด้านต่างๆ

งานวิจัยนี้จึงนำความสามารถของ RFID มาประยุกต์ใช้กับการควบคุมการเลือกชั้นลิฟต์เพื่อควบคุมการเข้าถึงตามชั้นต่างๆ ของอาคารโดยใช้ไมโครคอนโทรลเลอร์ในการโปรแกรมควบคุม โดยมี RFID มาเป็นตัวกำหนดให้กับผู้ใช้ ซึ่ง RFID ดังกล่าว จะมีการกำหนดระดับสิทธิ์ในการใช้ลิฟต์ในแต่ละชั้น ซึ่งสามารถทำเป็นระบบที่เพิ่มความเป็นส่วนตัว และความปลอดภัยให้กับผู้ใช้งาน

1.2 วัตถุประสงค์ของโครงการวิจัย

1.2.1 ได้ระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID

1.3 ขอบเขตของโครงการวิจัย

1.3.1 สามารถใช้ RFID ควบคุมการใช้งานลิฟต์ได้

1.3.2 สามารถเก็บข้อมูลการใช้งานได้

1.3.3 สามารถแสดงข้อมูลประวัติการใช้งานได้

1.4 ขั้นตอนดำเนินการ

1.4.1 ศึกษาทฤษฎีที่เกี่ยวกับหลักการทำงานของ RFID และการควบคุมการทำงานของลิฟต์

1.4.2 ออกแบบระบบควบคุม และโปรแกรมบนไมโครคอนโทรลเลอร์

1.4.3 พัฒนา-ปรับปรุงแก้ไขให้ทำงานได้มีประสิทธิภาพมากยิ่งขึ้น

1.4.4 ทดสอบการทำงานของระบบควบคุมการเลือกชั้นลิฟต์

1.4.5 สรุปผลการทดสอบและจัดทำเอกสาร รวบรวมรายละเอียดของโครงการทั้งหมด

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 สามารถป้องกันบุคคลภายนอกที่ไม่ได้รับอนุญาตในการใช้ลิฟต์

1.5.2 สามารถเพิ่มความปลอดภัยภายในอาคารได้

บทที่ 2
ทฤษฎีและหลักการ

ในบทนี้คณะผู้จัดทำปริญญานิพนธ์จะกล่าวถึงทฤษฎีที่สัมพันธ์กับปริญญานิพนธ์ ซึ่งเป็นทฤษฎีที่ใช้ประกอบเพื่อให้การศึกษาจัดทำปริญญานิพนธ์ของคณะผู้จัดทำลุล่วงไปได้ด้วยดี ซึ่งทฤษฎีที่กล่าวถึงจะเป็นแนวทางในการนำไปทดสอบการทำงานของระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID

2.1 ทฤษฎี RFID

อุปกรณ์ RFID RFID ย่อมาจากคำว่า Radio Frequency Identification เป็นระบบฉลากที่ได้ถูกพัฒนามาตั้งแต่ปี ค.ศ. 1980 โดยที่อุปกรณ์ RFID ที่มีการประดิษฐ์ขึ้นใช้งานเป็นครั้งแรกนั้นเป็นผลงานของ Leon Theremin ซึ่งสร้างให้กับรัฐบาลของประเทศรัสเซียในปี ค.ศ. 1945 ซึ่งอุปกรณ์ที่สร้างขึ้นมาในเวลานั้นทำหน้าที่เป็นเครื่องมือดักจับสัญญาณ ไม่ได้ทำหน้าที่เป็นตัวระบุเอกลักษณ์อย่างที่ใช้งานกันอยู่ในปัจจุบัน

RFID ในปัจจุบันมีลักษณะเป็นป้ายอิเล็กทรอนิกส์ (RFID Tag) ที่สามารถอ่านค่าได้โดยผ่านคลื่นวิทยุจากระยะห่าง เพื่อตรวจ ติดตามและบันทึกข้อมูลที่ติดอยู่กับป้าย ซึ่งนำไปฝังไว้ในหรือติดอยู่กับวัตถุต่างๆเช่น ผลิตภัณฑ์ กล่อง หรือสิ่งของใดๆ สามารถติดตามข้อมูลของวัตถุ 1 ชิ้นว่า คืออะไร ผลิตที่ไหน ใครเป็นผู้ผลิต ผลิตอย่างไร ผลิตวันไหน และเมื่อไร ประกอบไปด้วยชิ้นส่วนกี่ชิ้น และแต่ละชิ้นมาจากที่ไหน รวมทั้งตำแหน่งที่ตั้งของวัตถุนั้น ๆ ในปัจจุบันว่าอยู่ส่วนใดในโลก โดยไม่จำเป็นต้องอาศัยการสัมผัส (Contact-Less) หรือต้องเห็นวัตถุนั้นๆ ก่อน ทำงานโดยใช้เครื่องอ่านที่สื่อสารกับป้ายด้วยคลื่นวิทยุในการอ่านและเขียนข้อมูล

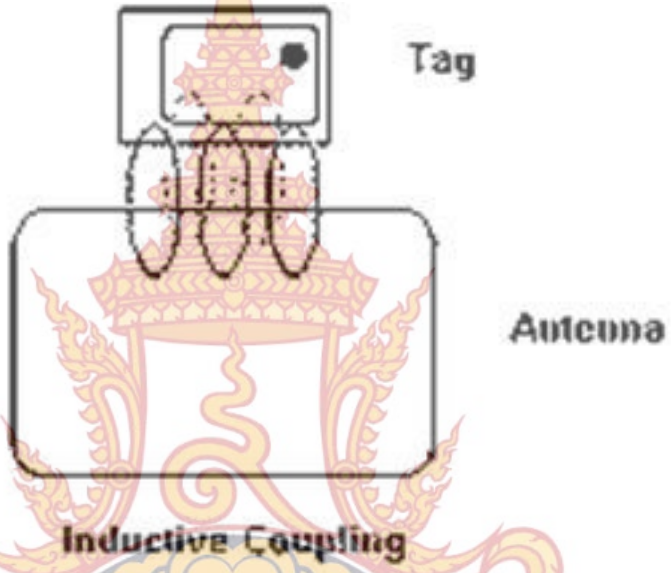
2.1.1 หลักการทำงานเบื้องต้นของระบบ RFID

2.1.1.1 ตัวอ่านข้อมูลจะปล่อยคลื่นแม่เหล็กไฟฟ้าออกมาตลอดเวลา และคอยตรวจจับว่ามีแท็กส์เข้ามาอยู่ในบริเวณสนามแม่เหล็กไฟฟ้าหรือไม่ หรืออีกนัยหนึ่งก็คือการคอยตรวจจับว่ามีการมอดูเลตสัญญาณเกิดขึ้นหรือไม่

2.1.1.2 มื่อมีแท็กส์เข้ามาอยู่ในบริเวณสนามแม่เหล็กไฟฟ้า แท็กส์จะได้รับพลังงานไฟฟ้าที่เกิดจากการเหนี่ยวนำของคลื่นแม่เหล็กไฟฟ้าเพื่อให้แท็กส์เริ่มทำงาน และจะส่งข้อมูลในหน่วยความจำที่ผ่านการมอดูเลตกับคลื่นพาหะแล้วออกมาทางสายอากาศที่อยู่ภายในแท็กส์

2.1.1.3 คลื่นพาหะที่ถูกส่งออกมาจากแท็กส์จะเกิดการเปลี่ยนแปลงแอมปลิจูด , ความถี่ หรือเฟส ขึ้นอยู่กับวิธีการมอดูเลต

2.1.1.4 ตัวอ่านข้อมูลจะตรวจจับความเปลี่ยนแปลงของคลื่นพาหะแปลงออกมาเป็นข้อมูลแล้วทำการถอดรหัสเพื่อนำข้อมูลไปใช้งานต่อไป



ภาพที่ 2.1 การสื่อสารระหว่างแท็กส์และตัวรับข้อมูล

2.1.2 ส่วนประกอบของ RFID

ระบบอาร์เอฟไอดี ประกอบด้วย ส่วนทรานสปอนเดอร์ หรือ ป้าย ส่วนนี้จะประกอบไปด้วย สายอากาศและไมโครชิปที่มีการบันทึกข้อมูล ส่วนที่สองคือ เครื่องอ่าน/เขียนข้อมูล ด้วนคลื่นวิทยุซึ่งแตกต่างกับระบบรหัสแท่ง ซึ่งอ่านด้วยแสงเลเซอร์ ข้อเสียของการอ่านด้วยเลเซอร์คือจะต้องไม่มีสิ่งกีดขวางหรือต้องอยู่ในแนวเส้นตรงกับลำแสงที่ยิงจากเครื่องสแกนและสามารถอ่านได้ทีละรหัสในระยะใกล้ๆ แต่ระบบอาร์เอฟไอดี มีความแตกต่างคือ สามารถอ่านรหัสจากป้ายได้โดยไม่ต้องเห็นป้าย หรืออยู่ในวัตถุ และไม่จำเป็นต้องอยู่ในแนวเดียวกับคลื่น เพียงแต่อยู่ในระยะที่สามารถอ่านป้ายได้ และการอ่านป้ายในระบบอาร์เอฟไอดี ยังสามารถอ่านได้ทีละหลายๆป้ายในเวลาเดียวกัน ส่วนที่สามคือ ระบบประยุกต์การใช้งาน ทั้งนี้รวมถึงระบบฮาร์ตแวร์และซอฟต์แวร์ประยุกต์ใช้งาน หรือระบบฐานข้อมูล

2.1.2.1 ป้าย (Tag/Transponders)

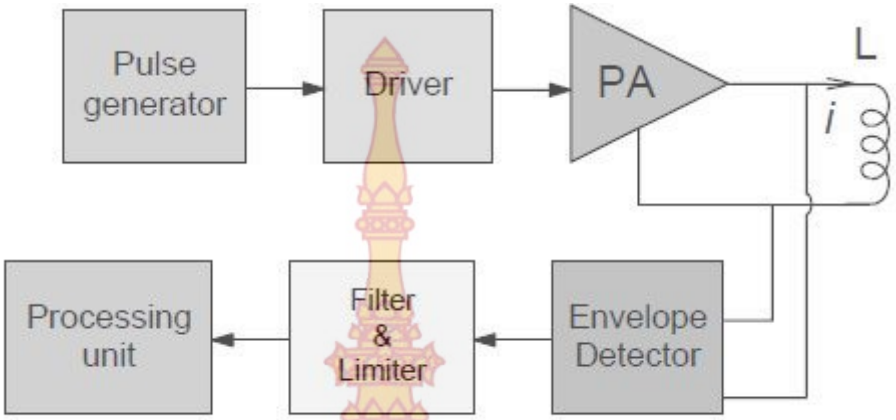
ภายในประกอบไปด้วย 2 ส่วนใหญ่ๆ คือ ส่วนของไมโครชิป มีหน้าที่เก็บข้อมูล และขดลวดขนาดเล็กซึ่งทำหน้าที่เป็นสายอากาศ สำหรับส่งสัญญาณคลื่นความถี่วิทยุและสร้างพลังงานป้อนให้ไมโครชิป ป้ายจะอยู่ในรูปแบบกระดาษ แผ่นฟิลม์ พลาสติก มีขนาดรูปร่างต่างๆกันไป และมีหลายรูปแบบเช่น บัตรเครดิต เหรียญ กระดุม ฉลาก สินค้า แคปซูล เป็นต้น สามารถแบ่งป้ายเป็น 2 ชนิด คือ ป้ายแบบแพสทีฟ และแบบแอ็กทีฟ โดยแต่ละชนิดแตกต่างกันตามจุดประสงค์การใช้งาน หรือสามารถแบ่งชนิดของป้ายตามความถี่การใช้งาน

2.1.2.2 Passive Tags

ป้ายชนิดนี้ทำงานโดยไม่ต้องอาศัยแหล่งจ่ายไฟจากภายนอก เนื่องจากภายในป้ายจะมีวงจรกำเนิดไฟฟ้าเหนี่ยวนำขนาดเล็ก ทำหน้าที่เป็นแหล่งจ่ายไฟในตัว ซึ่งทำให้ป้ายชนิดนี้อ่านข้อมูลได้ในระยะใกล้เท่านั้น ระยะอ่านประมาณ 1 เมตร ขึ้นอยู่กับความถี่ที่ใช้งานละกำลังของเครื่องส่ง หน่วยความภายในป้ายชนิดนี้จะมีขนาดเล็ก ประมาณ 10-1k byte ทำให้มีข้อดีคือขนาดเล็กน้ำหนักเบา และราคาถูก

2.1.2.3 เครื่องอ่าน/เขียนข้อมูล

เครื่องอ่าน/เขียนข้อมูล ทำหน้าเชื่อมต่อกับป้ายด้วยคลื่นวิทยุ ซึ่งภายในประกอบไปด้วยเสาอากาศ ทำด้วยขดลวดทองแดง ทำหน้าที่เป็นตัวรับ/ส่งสัญญาณ ภาครับ/ภาคส่งสัญญาณวิทยุ และวงจรควบคุมการอ่าน/เขียนข้อมูลซึ่งมักเป็นวงจรไมโครคอนโทรลเลอร์ และส่วนติดต่อกับคอมพิวเตอร์ดังภาพที่ 2.2



ภาพที่ 2.2 โครงสร้างการทำภายในเครื่องอ่านอาร์เอฟไอดี

2.1.3 ย่านความถี่ใช้งานในระบบ RFID

ความถี่	ช่วงความถี่	ช่วงความยาวคลื่น	ความถี่ ISM	ระยะการอ่าน (ป้ายแพสซิฟ)
ความถี่ต่ำ (LF)	30 – 300 kHz	10 km – 1 km	125 – 135 kHz	< 50 cm
ความถี่สูง (HF)	3 – 30 MHz	100 m – 10 m	6.78, 8.11, 13.56, และ 27.12 MHz	< 3 m
ความถี่สูงยิ่ง (UHF)	300 – 3000 MHz	1 m – 10 cm	433, 869, 915 MHz	< 9 m
ความถี่ไมโครเวฟ	1 – 300 GHz	30 cm – 1 mm	2.44, 5.89 GHz	> 10 m

ภาพที่ 2.3 แสดงภาพย่านความถี่ที่ใช้งานในระบบ RFID

ป้าย RFID และเครื่องอ่านใช้ความถี่วิทยุ ในการติดต่อสื่อสารซึ่งเรียกความถี่นั้นว่า ความถี่ ใช้งาน (operating frequency) ความถี่วิทยุเป็นคลื่นแม่เหล็กไฟฟ้าที่เป็นส่วนหนึ่งของสเปกตรัมความถี่แม่เหล็ก ไฟฟ้าซึ่งเรียกว่าสเปกตรัมความถี่วิทยุ (radio frequency spectrum) เนื่องจากระบบ RFID สร้างและส่งคลื่นแม่เหล็กไฟฟ้าซึ่งอยู่ในสเปกตรัมความถี่วิทยุดังนั้นจึงต้องมีระบบจัดสรรคลื่นความถี่วิทยุสำหรับใช้ในงานประยุกต์ ต่างๆ เช่นวิทยุ ,โทรทัศน์, และโทรศัพท์มือถือเพราะฉะนั้นจึงมีความจำเป็นอย่างยิ่งที่จะต้องกำหนดความถี่ที่ใช้งานของระบบ RFID เพื่อไม่ไปรบกวนการทำงานของระบบอื่นๆ

2.2 Arduino

Arduino คือ ไมโครคอนโทรลเลอร์ ชนิดหนึ่ง ซึ่งเป็นแบบที่เรียกว่า Open Hardware กล่าวคือ Arduino อุปกรณ์ที่มีแบบส่วนประกอบเป็นมาตรฐานที่เปิดเผย หมายความว่า เราสามารถทำเองโดยใช้แบบที่มีการเปิดเผยทั่วไปก็ได้ หรือสามารถซื้อหาได้ง่าย มีราคาถูก มีซอฟต์แวร์ให้ใช้งานฟรี สามารถนำไปใช้งานทั่วไปหรือแบบธุรกิจได้โดยไม่ต้องเสียค่าลิขสิทธิ์ เป็นรูปแบบที่มีข้อมูลมากที่สุดบนอินเทอร์เน็ต การพัฒนาก็ง่าย เพราะมีตัวอย่างมากมาย และไม่ต้องเขียนโปรแกรมในรูปแบบ Low Level หมายความว่า เราสามารถใช้คำสั่งเขียนโปรแกรมได้เสมือนโปรแกรมภาษาชั้นสูงทั่วไป

2.2.1 โครงสร้างโปรแกรมของArduno

ในการเขียนโปรแกรมสำหรับโมดูล Freeduino จะต้องเขียนโปรแกรมโดยใช้ภาษาของ Arduino (Arduino Programming Language) ซึ่งตัวภาษาของ Arduino เองก็นำเอาโอเพ่นซอร์สโปรเจคชื่อ Wiring มาพัฒนาต่อภาษาของ Arduino แบ่งออกเป็น 2 ส่วนหลักคือ

2.2.1.1 โครงสร้างภาษา (Structure) ตัวแปรและค่าคงที่

2.2.1.2 ฟังก์ชั่น (Function)

ภาษาของ Arduino จะอ้างอิงตามภาษา C/C++ ซึ่งการเขียนโปรแกรมสำหรับ Arduino ก็คือการเขียนโปรแกรมภาษาซีโดยเรียกใช้ฟังก์ชั่นและไลบรารี่ที่ทาง Arduino ได้เตรียมไว้ให้แล้วซึ่งสะดวกและทำให้ผู้ที่ไม่มีความรู้ด้านไมโครคอนโทรลเลอร์อย่างลึกซึ้งสามารถเขียนโปรแกรมสั่งงานได้

โปรแกรมของ Arduino แบ่งได้เป็นสองส่วนคือ void setup () และ void loop () โดยฟังก์ชั่น setup () เมื่อโปรแกรมทำงานจะทำคำสั่งของฟังก์ชั่นนี้เพียงครั้งเดียว ใช้ในการกำหนดค่าเริ่มต้นของการทำงาน ส่วนฟังก์ชั่น loop () เป็นส่วนทำงาน โปรแกรมจะทำคำสั่งในฟังก์ชั่นนี้ต่อเนื่องกันตลอดเวลา โดยปกติใช้กำหนดโหมดการทำงานของขาต่าง ๆ กำหนดการสื่อสารแบบอนุกรม ฯลฯ ส่วนของ loop () เป็นโค้ดโปรแกรมที่ทำงาน เช่น อ่านค่าอินพุต ประมวลผล สั่งงานเอาต์พุต ฯลฯ โดยกำหนดค่าเริ่มต้น เช่น ตัวแปร จะต้องเขียนที่ส่วนหัวของโปรแกรม ก่อนถึงตัวฟังก์ชั่น นอกจากนั้นยังต้องคำนึงถึง ตัวพิมพ์เล็ก-ใหญ่ ของตัวแปรและชื่อฟังก์ชั่นให้ถูกต้อง

2.2.2 ส่วนของฟังก์ชั่น setup ()

ฟังก์ชั่นนี้จะเขียนที่ส่วนต้นของโปรแกรม ทำงานเมื่อโปรแกรมเริ่มต้นเพียงครั้งเดียว ใช้เพื่อกำหนดค่าของตัวแปร โหมดการทำงานของขาต่าง ๆ เริ่มต้นเรียกใช้ไลบรารี่ ฯลฯเช่น

```
int  buttonPin  =  3;
void setup()
{
      beginSerial(9600);
      pinMode(buttonPin, INPUT);
}
void loop()
{
      if(digitalRead(buttonPin == HIGH)
      serialWrite('H');
      else
      serialWrite('L');
      delay(1000);
}
```

ภาพที่ 2.4 แสดงการใช้ฟังก์ชั่น setup ()

2.2.3 ส่วนของฟังก์ชั่น loop ()

หลักจากที่เขียนฟังก์ชั่น setup () ที่กำหนดค่าเริ่มต้นของโปรแกรมแล้ว ส่วนถัดมาคือฟังก์ชั่น loop () ซึ่งมีการทำงานตรงตามชื่อ คือ จะทำงานตามฟังก์ชั่นนี้วนต่อเนื่องตลอดเวลา ภายในฟังก์ชั่นนี้จะมีโปรแกรมของผู้ใช้ เพื่อรับค่าจากพอร์ต ประมวลผล แล้วสั่งเอาต์พุตออกขาต่าง ๆ เพื่อควบคุมการทำงานของบอร์ด

```
int  buttonPin  =  3;
// setup initializes serial and the button pin
void setup()
{
      beginSerial(9600);
      pinMode(buttonPin, INPUT);
}

//loop checks the button pin each time
//and will send serial if it is pressed

Void loop()
{
      if(digitalRead(buttonPin == HIGH)
      serialWrite('H');
      else
      serialWrite('L');
      delay(1000);
}
```

ภาพที่ 2.5 แสดงการใช้คำสั่ง loop ()

2.2.4 คำสั่งควบคุมการทำงาน

2.2.4.1 คำสั่ง if

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรม เช่น ถ้าอินพุตมีค่ามากกว่าค่าที่กำหนดไว้จะให้ทำอะไร โดยมีรูปแบบการเขียน ดังนี้

```
if(someVariable > 200)
      {
      // do something here
      }
```

ตัวโปรแกรม จะทดสอบว่า ถ้าตัวแปร someVariable มีค่ามากกว่า 50 หรือไม่ ถ้าใช่ให้ทำอะไร ถ้าไม่ใช่ให้ข้ามการ
ทำงานส่วนนี้ การทำงานของคำสั่งจะทดสอบเงื่อนไข ที่เขียนในเครื่องหมายวงเล็บ ถ้าเงื่อนไขเป็นจริง ทำตามคำสั่งที่เขียนในวงเล็บปีกกา ถ้าเงื่อนไขเป็นเท็จ ข้ามการทำงานส่วนนี้ไป

```
if(pinFiveInput < 200)
{
      //   do Thing A
}
else
{
     // do Thing B
}
```

ภาพที่ 2.6 แสดงเงื่อนไขที่อยู่ภายในวงเล็บที่ต้องใช้ตัวกระทำเปรียบเทียบต่าง ๆ

ในการเปรียบเทียบตัวแปรให้ใช้ตัวกระทำ == (เช่น if (x == 10) ห้ามเขียนผิดเป็น = (เช่น if (x = 0)) คำสั่งที่เขียนผิดในแบบที่สองนี้ ทำให้ผลการทดสอบเป็นจริงเสมอ และเมื่อผ่านคำสั่งนี้แล้ว x มีค่าเท่ากับ 10 ทำให้การทำงานของโปรแกรมผิดเพี้ยนไป ไม่เป็นไปตามที่กำหนด
เราสามารถใช้คำสั่ง if ในคำสั่งควบคุมการแยกเส้นทางของโปรแกรม โดยใช้คำสั่ง if .. else

2.2.4.2 คำสั่ง if...else

ใช้ทดสอบเพื่อกำหนดเงื่อนไขการทำงานของโปรแกรมได้มาก กว่าคำสั่ง if ธรรมดา โดยสามารถกำหนดได้ว่า ถ้าเงื่อนไขเป็นจริงให้ทำอะไร ถ้าเงื่อนไขเป็นเท็จให้ทำอะไร เช่น ถ้าค่าอินพุตอนาลอกที่อ่านได้น้อยกว่า 500 ให้ทำอะไร ถ้าค่ามากกว่าหรือเท่ากับ 500 ให้ทำอีกอย่าง สามารถเขียนคำสั่งได้ดังนี้

```
if(pinFiveInput < 500)
{
      //  do Thing A
}

else if (pinFiveInput <= 1000)
{
      // do Thing B
}
else if
{
      // do Thing C
}
```

ภาพที่ 2.7 แสดงอินพุตที่อ่านได้ 500 ให้ทำอะไร ถ้ามากกว่าให้ทำอีกอย่าง

หลังคำสั่ง else สามารถตามด้วยคำสั่ง if สำหรับการทดสอบอื่น ๆ ทำให้รูปแบบคำสั่งกลายเป็น if....else....if เป็นการทดสอบ อื่น ๆ เมื่อเป็นจริงให้ทำตามที่ต้องการ ดังตัวอย่างต่อไปนี้

```
x == y ( x เท่ากับ y)
x != y ( x ไม่เท่ากับ y)
x < y ( x น้อยกว่า y)
x > y ( x มากกว่า y)
x <= y ( x น้อยกว่าหรือเท่ากับ y)
x >= y ( x มากกว่าหรือเท่ากับ y)
```

ภาพที่ 2.8 หลังคำสั่ง else สามารถตามด้วยคำสั่ง if ได้ไม่จำกัดจำนวน

หลังคำสั่ง else สามารถตามด้วยคำสั่ง if ได้ไม่จำกัดจำนวน (สามารถใช้คำสั่ง switch case แทนคำสั่ง if...else...if) สำหรับการทดสอบเงื่อนไขจำนวนมาก ๆ ได้ เมื่อใช้คำสั่ง if..else แล้วต้องกำหนดด้วยว่าถ้าทดสอบไม่ตรงกับเงื่อนไขใด ๆเลยให้ทำอะไรโดยให้กำหนดที่คำสั่ง else ตัวสุดท้าย

2.2.4.3 คำสั่ง for ()

คำสั่งนี้ใช้เพื่อสั่งให้คำสั่งที่อยู่ภายในวงเล็บปีกกาหลัง for มีการทำงานซ้ำกันตามจำนวนรอบที่ต้องการ คำสั่งนี้มีประโยชน์มากสำหรับการทำงานใด ๆ ที่ต้องทำซ้ำกันและทราบจำนวนรอบของการทำซ้ำที่แน่นอน มักใช้คู่กับตัวแปรอะเรย์ในการเก็บสะสมค่าที่อ่านได้จากขาอินพุตแอนาลอกหลาย ๆ ขาที่มีหมายเลขขาต่อเนื่องกัน

รูปแบบของคำสั่ง for () แบ่งได้ 3 ส่วนดังนี้

```
for (initialization; codition; increment)
}
// statement (s);
}
```

เริ่มต้นด้วย initialization ใช้กำหนดค่าเริ่มต้นของตัวแปรควบคุมการวนรอบ ในการทำงานแต่ละรอบจะทดสอบcondition ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่งในวงเล็บปีกกา แล้วมาเพิ่มหรือลดค่าตัวแปรตามที่สั่งใน increment แล้วทดสอบเงื่อนไขอีก ทำซ้ำกว่าเงื่อนไขเป็นเท็จ

```
for(int i=1; i<=8; i++)
{
//statement using the value i;
}
```

คำสั่ง for ของภาษาซีจะยืดหยุ่นกว่าคำสั่ง for ของภาษาคอมพิวเตอร์อื่น ๆ โดยสามารถละเว้นบางส่วนหรือทั้งสามส่วน

ของคำสั่ง for ได้ อย่างไรก็ตามยังคงต้องมีเซมิโคลอน เราสามารถนำคำสั่งภาษาซีที่มีตัวแปรที่ไม่เกี่ยวข้องมาเขียนในส่วนของinitialization , condition และ increment ของคำสั่ง for ได้

2.2.4.4 คำสั่ง switch-case

ใช้ทดสอบเงื่อนไขเพื่อกำหนดการทำงานของโปรแกรม ถ้าตัวแปรที่ทดสอบตรงกับเงื่อนไขใดก็ให้ทำงานตามที่กำหนดไว้

พารามิเตอร์

var ตัวแปรที่ต้องการทดสอบว่าตรงกับเงื่อนไขใด

default ถ้าไม่ตรงกับเงื่อนไขใด ๆ เลยให้ทำคำสั่งต่อท้ายนี้

break เป็นส่วนสำคัญมากให้เขียนต่อท้าย case ต่างๆเมื่อพบเงื่อนไขนั้นแล้วให้ทำตามคำสั่งต่างๆแล้วให้หยุดการทำงานของคำสั่ง switch-case ถ้าลืมเขียน break เมื่อพบเงื่อนไขทำตามเงื่อนไขแล้ว โปรแกรมจะทำงานตามเงื่อนไขต่อไปเรื่อยๆจนกว่าจะพบคำสั่ง break

```
switch (var)
{
        case 1:
                // do something when var ==1
        break;
        case 2:
                // do something when var ==2
        break;
        default:
                // if nothing else matches, do the default
 }
```

ภาพที่ 2.9 แสดงคำสั่งโดยการใช้ var,default,sbreak

2.2.4.5 คำสั่ง while

เป็นคำสั่งวนรอบ โดยจะทำคำสั่งที่เขียนไว้ในวงเล็บปีกกาอย่างต่อเนื่อง จนกว่าเงื่อนที่เขียนไว้ในวงเล็บของคำสั่ง while( ) จะเป็นเท็จ คำสั่งที่ให้ทำซ้ำจะต้องมีการเปลี่ยนแปลงค่าตัวแปรที่ใช้ทดสอบ เช่นมีการเพิ่มค่าตัวแปรหรือมีเงื่อนไขภายนอกเช่น อ่านค่าจากเซ็นเซอร์ได้

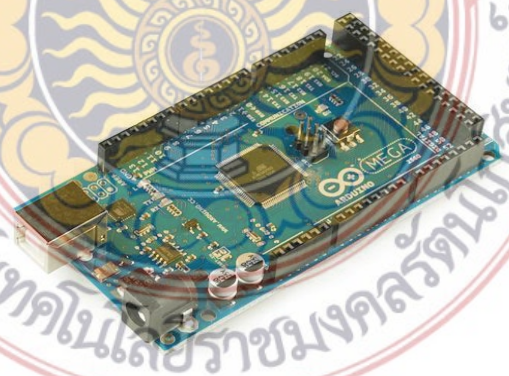
เรียบร้อยแล้วให้หยุดการอ่านค่า มิฉะนั้นเงื่อนไขในวงเล็บของ while( ) เป็นจริงตลอดเวลาทำให้คำสั่ง while ทำงานวนรอบไปเรื่อยๆไม่รู้จบ

```
var   = 0;
while (var <= 7)
{
       // do something repetitive 200 times
       var++;
}
```

ภาพที่ 2.10 แสดงการใช้คำสั่ง while ซึ่งทำงานวนรอบไปเรื่อยๆ

2.2.4.6 Arduino Mega 2560 R3

Arduino Mega 2560 R3 เป็นบอร์ด Arduino ที่ออกแบบมาสำหรับงานที่ต้องใช้ I/O มากกว่า Arduino Uno R3 เช่น งานที่ต้องการรับสัญญาณจาก Sensor หรือควบคุมมอเตอร์ Servo หลายๆ ตัว ทำให้ Pin I/O ของบอร์ด Arduino Uno R3 ไม่สามารถรองรับได้ ทั้งนี้บอร์ด Mega 2560 R3 ยังมีความหน่วยความจำแบบ Flash มากกว่า Arduino Uno R3 ทำให้สามารถเขียนโค้ดโปรแกรมเข้าไปได้มากกว่า ในความเร็วของ MCU ที่เท่ากัน

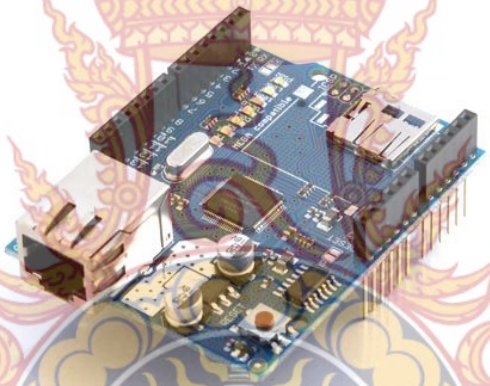


ภาพที่ 2.11 แสดงบอร์ด Arduino Mega 2560 R3

2.2.4.7 Arduino Ethernet Shield

ETHERNET SHIELD จะใช้ประกอบกับ ARDUNIO UNO เพื่อที่จะทำให้สามารถติดต่อกับระบบเครือข่ายได้ โดยใช้ Ethernet Library ซึ่งเวอร์ชั่นล่าสุดจะมี ช่องอ่าน Micro SD Card ติดมาด้วย สามารถใช้กับ SD Library ของ ARDUINO

การเชื่อมต่อกับ ETHERNET SHIELD นี้จะใช้สาย RJ45 อาจจะใช้ CAT5 หรือ CAT6 โดยสามารถใช้ DHCP ได้ แต่การเชื่อมต่อระหว่างสองจุดยังต้องใช้สาย Cross Over อยู่ เพราะไม่มีวงจร Cross Over ภายใน ความเร็วในการสื่อสารของบอร์ดนี้จะอยู่ที่ 10/50 Mbps หรือ 10/100 Mbps แล้วแต่แหล่งผลิต



ภาพที่ 2.12 แสดงบอร์ด Arduino Ethernet Shield

ในส่วนของ ETHERNET SHIELD นี้มีโมดูลที่สามารถรองรับ Power over Ethernet (PoE) ซึ่งสามารถใช้แหล่งจ่ายไฟของบอร์ดได้เลย นอกจากนี้ยังมีไฟแสดงผลซึ่งสามารถอธิบายได้ดังนี้

PWR	ไฟแสดงสัญญาณ Power
LINK	ไฟแสดงสถานะการอัพโหลดและดาวโหลดข้อมูลผ่านเครือข่าย
FULLD	ไฟแสดงสถานะของการเชื่อมต่อแบบ Full Duplex
100M	ไฟแสดงถานะเมื่อมีการเชื่อมต่อเครือข่ายได้ถึง 100 Mbps

RX	ไฟแสดงสถานะเมื่อ ETHERNET SHIELD มีการรับข้อมูล
TX	ไฟแสดงสถานะเมื่อ ETHERNET SHIELD เมื่อมีการส่งข้อมูล
COLL	ไฟแสดงสถานะเมื่อมี IP ชนกันของเครือข่าย

2.3 ความเป็นมาของลิฟต์

ตึกสูงเสียดฟ้าต่างๆ คงไม่สามารถเกิดขึ้นได้ถ้าไม่มีการคิดค้นและการพัฒนาระบบลิฟท์ ในอาคารก่อนหน้านั้นอาคาร 5-6 ชั้นถือได้ว่าสูงที่สุดแล้วสำหรับอาคารก่ออิฐฉาบปูนทั้งหลาย โดยที่จริงแล้วเทคโนโลยีอาจจะเพียงพอที่จะสร้างอาคารได้สูงกว่านั้นแต่ปัญหาของการสัญจรทางตั้งหรือการขึ้นลงอาคารเป็นข้อจำกัดทันทีที่ระบบลิฟท์ถูกคิดค้นขึ้นมาอาคารก็เริ่มพัฒนาความสูงอย่างรวดเร็วโครงสร้างเหล็กและการพัฒนาระบบลิฟท์จากไฮดรอลิกมาเป็นลิฟท์ไฟฟ้าทำให้อาคารสูงขึ้นอย่างมากมายในช่วงคริสศตวรรษที่ 19-20 ในระยะเริ่มต้นลิฟต์ได้ใช้แรงจากคนหรือสัตว์ในยุคต่อมาเริ่มพัฒนาการใช้แรงกังหันน้ำในช่วงศตวรรษที่ 3 งานก่อสร้างปิรามิดที่อียิปต์เป็นตัวอย่างหนึ่งในการใช้ลิฟต์จากแรงงานของทาสทั้งหลายในการยกรูปแบบของลิฟต์สมัยใหม่เกิดขึ้นจริงๆ ในปี 1743 ในฝรั่งเศสสำหรับพระราชวังแวร์ซายส์ของพระเจ้าหลุยส์ที่ 14 มีความเชื่อว่าพระองค์ทรงใช้ลิฟต์สำหรับนางสนมต่างๆ อย่างลับๆ ลิฟต์ได้พัฒนาการใช้เครื่องจักรกลในศตวรรษที่ 19 ซึ่งมีลักษณะที่คล้ายคลึงกับระบบปัจจุบัน โดย Elisha Otis ประดิษฐ์ลิฟต์ที่มีระบบความปลอดภัยครั้งแรกในราวปี 1850ตัวลิฟต์จะเคลื่อนที่ขึ้นลงโดยระบบไฮดรอลิกหรือระบบตุ้มน้ำหนักมีรางรับด้านข้าง ซึ่งติดตั้งตัวหนีบในกรณีฉุกเฉิน ถ้าลิฟท์มีการเคลื่อนที่เร็วเกินที่กำหนดระบบความปลอดภัยจะทำงานโดยการชลอและหยุดลิฟท์ ทำให้เกิดความปลอดภัยในการใช้มากขึ้น

2.3.1 การใช้รอกยกของในยุคกลาง

Elisha Otis เป็นสุดยอดช่างเครื่องแห่งบริษัท Bedstead Manufacturing Company ในเมือง Yonkers แห่งมลรัฐ New York ลิฟต์ที่เผื่อระบบความปลอดภัยที่เขาผลิตดังที่กล่าวมาแล้วนั้น ถูกใช้ในการติดตั้งในลิฟต์สำหรับยกของในปี 1852ต่อจากนั้นเขาก็มาเริ่มตั้งบริษัท E.G. Otis และขายลิฟต์ของเขาโดยจะเป็นลักษณะของลิฟต์ขนของ จนกระทั่งปี 1857 บริษัท Otis จึงได้มีโอกาสได้ติดตั้งลิฟต์สำหรับคนโดยสารเป็นครั้งแรกของโลก ที่ร้าน E.V.Haughwout & Company ในมหานครนิวยอร์ค

2.3.2 E.V.Haughwout & Company

ในปี 1870 Lord และ Taylor ได้ใช้ลิฟต์ในร้านในนิวยอร์ลิฟต์สำหรับคนโดยสารถูกนำมาใช้ในอาคารสำนักงานเมื่อปี 1871 มาถึงปี 1873 ลิฟต์ของบริษัท Otis ก็ถูกติดตั้งสำหรับอาคารสำนักงาโรงแรมและร้านค้าถึง 2,000 แห่ง ทั่วอเมริกาทั้งนี้ลิฟต์ทั้งหมดที่ติดตั้งในยุคนั้นใช้พลังงานจากเครื่องจักรไอน้ำในช่วงราวปี 1870 ลิฟต์ในระบบไฮดรอลิกใช้ระบบแรงดันของน้ำแทนระบบเครื่องไอน้ำได้ถูกคิดค้นขึ้นซึ่งกลายมาเป็นระบบที่มีความปลอดภัยสูงและประหยัดกว่าแบบเดิมห้องโดยสารถูกออกแบบให้ยึดติดอยู่กับลูกสูบซึ่งเคลื่อนที่ขึ้นลงในปล่องลิฟท์ควบคุมโดยระบบวาล์วของไฮดรอลิก อาคารหลังแรกที่ใช้ลิฟต์ระบบนี้คือ อาคารเลขที่ 155 ถนนบรอดเวย์ในนิวยอร์ค ในปี 1878 ลิฟต์นี้สามารถขนส่งผู้โดยสารได้สูงถึง 111 ฟุต ทำให้ในช่วงปีทศวรรษที่ 1870 ลิฟต์ไฮดรอลิกสามารถขนส่งผู้โดยสารได้สูงกว่าระบบเดิมโดยอยู่ระหว่างความสูงประมาณ 9-10 ชั้น ลิฟต์ในระบบไฮดรอลิกยังคงใช้ในอาคารไม่สูงมากในปัจจุบัน

ในช่วงทศวรรษที่ 1880 สถาปนิกในนิวยอร์คและชิคาโกถือได้ว่าเป็นผู้บุกเบิกในการระบบโครงสร้างเฟรมเหล็กก่อนหน้านี้ความสูงของอาคารถูกจำกัดโดยความหนาของผนังรับน้ำหนักของอาคารก่ออิฐฉาบปูน อาคาร Home Insurance ในเมืองชิคาโกตึกระฟ้าหลังแรกของโลกถูกสร้างขึ้นในปี1889 ก็ยังใช้ลิฟต์ไฮดรอลิกอยู่จนกระทั่งปี1889 ลิฟต์ตัวแรกที่ใช้พลังงานไฟฟ้าในการขับเคลื่อนมอเตอร์จึงเกิดขึ้นด้วยการผสมผสานของเทคโนโลยีในการออกแบบอาคารและระบบลิฟต์ใหม่ๆ อาคารจึงมีความสูงมากขึ้นโดยเฉพาะในช่วงทศวรรษ1890 ในปี1902 ลิฟต์ Otisได้ถูกใช้ในอาคาร Flatiron ในเมืองนิวยอร์คถือว่าเป็นอาคารยุคแรกๆ ที่มีความสูงถึง 285 ฟุต ที่มีลิฟต์ในที่สุดลิฟต์ในระบบ Gearless Traction ก็ถูกคิดค้นขึ้นและกลายมาเป็นระบบหลักของลิฟต์ตราบเท่าปัจจุบัน โดยถูกใช้ในอาคาร Beaver ในนิวยอร์ค เป็นแห่งแรก ทำให้ความเร็วของลิฟต์รวดเร็วขึ้นกว่าแต่ก่อนอย่างมากมายและสามารถใช้ได้ในอาคารไม่ว่าจะสูงแค่ไหน ระบบจะประกอบด้วยเคเบิลที่ใช้ดึงตัวห้องโดยสารขึ้นและใช้ตุ้มน้ำหนักในการถ่วง ซึ่งจะช่วยลดการใช้พลังงานที่ใช้ในการเคลื่อนที่ของห้องโดยสาร ลิฟท์ระบบนี้จะมีความนุ่มนวลในการเคลื่อนที่มากที่สุดเมื่อเปรียบเทียบกับระบบอื่นๆ หลังจากระบบนี้ถูกนำมาใช้ ความนิยมของลิฟท์ไฮดรอลิกก็เริ่มลดลงจนกระทั่งถูกใช้ในอาคารความสูงประมาณ 5-6 ชั้น เท่านั้นการออกแบบลิฟต์ให้กับอาคาร Woolworth ซึ่งสูงถึง 792 ฟุต ทำให้เกิดการคำนึงถึงการจำกัดความเร็วของลิฟต์โดยไม่ให้เกิน 700 ฟุตต่อนาที

2.3.3 อาคาร Woolworth นิวยอร์ค

ในปี 1924 ได้มีการติดตั้งลิฟต์ที่ใช้ระบบสัญญาณควบคุม (Signal Control System) ขึ้นเป็นครั้งแรกในอาคาร Standard Oil ที่นิวยอร์คแม้ว่าผู้โดยสารจะยังคงต้องเปิดปิดประตูเอง แต่ก็ได้เริ่มเอาการใช้สัญญาณเรียกลิฟท์และการออกแบบให้ลิฟท์ที่ใกล้ที่สุดไปหาสัญญาณเรียกได้ ซึ่งนับเป็นการพัฒนาการอีกขั้นหนึ่งในช่วงปี 1930 การออกแบบลิฟต์มีแนวความคิดที่จะลดช่องลิฟต์ เพื่อให้มีพื้นที่การใช้งานของอาคารมากขึ้นและเพื่อเป็นการประหยัดการก่อสร้างอาคารสูง ในช่วงต้นทศวรรษที่ 1930 อาคารสูงที่ในโลกได้ถูกก่อสร้างขึ้นในนิวยอร์คไม่ว่าจะเป็นEmpire State Building, Chrysler Building, Bank of Manhattan, และ 60 Wall Street Tower อาคารขนาด 40-60 ชั้นเกิดขึ้นหลายอาคารทั่วประเทศการจำกัดความเร็วของลิฟต์ได้ถูกกำหนดใหม่อีกครั้งโดยที่ตึก Empire State ซึ่งมีลิฟท์มากถึง 73 ตัว มีความเร็วของลิฟต์ 1200 ฟุตต่อนาที

ในปี 1948 Otis ได้แนะนำผลิตภัณฑ์ใหม่ซึ่งใช้ระบบควบคุมที่ชื่อว่า “autotronic” อันเป็นคำผสมระหว่าง automatic และ electronic ลิฟต์จะขึ้นลงตามคำสั่งจากปุ่มเรียกตัวลิฟต์ถูกควบคุมโดยระบบไฟฟ้าซึ่งถูกพัฒนาไปอีกระดับหนึ่ง ในช่วงกลางทศวรรษที่ 1960 เริ่มมีการแบ่งโซนการใช้ลิฟต์ถูกออกแบบให้มีโซนด่วนเพื่อให้บริการเฉพาะชั้นล่างหรือช่วงโซนบนหรือทุกชั้น ด้วยการออกแบบให้เป็นโซนอย่างนี้ได้พัฒนาใช้มาจนปัจจุบันในปี 1967 ตึกคู่ World Trade Center ซึ่งมีความสูง 110 ชั้นได้ถูกสร้างขึ้น ใช้ลิฟต์ทั้งหมด 255 ตัว เพื่อให้บริการพนักงานในตึกประมาณ 50,000 คน และนักท่องเที่ยวอีก 80,000 คนต่อวัน CN Tower ซึ่งเป็นหอชมวิวที่สูงที่สุดในโลกตั้งอยู่เมืองโตรอนโตสร้างในปี 1975 ลิฟต์ผนังกระจกได้ถูกออกแบบให้ผู้โดยสารสามารถชมวิวไปด้วยได้ ลิฟต์ขึ้นไปถึงส่วนร้านอาหารและส่วนชมวิวซึ่งอยู่สูง 1126 ฟุตจากพื้น

ในปี 1982 Otis ได้ออกแบบให้ลิฟต์มีเกียร์เพื่อปรับความเร็วของลิฟต์ได้ เพื่อให้เหมาะสมกับน้ำหนักบรรทุก มอเตอร์รุ่นใหม่ก็ให้การขับเคลื่อนที่นุ่มนวลกว่าก่อนเก่า ในปี 1996 Otis ได้ออกลิฟต์ระบบใหม่ที่ชื่อว่า Odyssey ซึ่งถือว่าเป็นการออกแบบเพื่ออนาคต ผสมผสานการสัญจรของอาคารเพื่อให้สามารถออกแบบอาคารให้มีความสูงมากๆ และกว้างมากๆ ได้ โดยลิฟต์สามารถเคลื่อนที่ทั้งแนวราบและแนวนอน

2.3.4 ชนิดของลิฟต์

ระบบเครื่องกลในพัฒนาการของลิฟต์ได้เปลี่ยนรูปโฉมอย่างมากมายในช่วงปี 1900s เมื่อไฟฟ้าถูกนำมาใช้ในอาคาร ระบบลิฟต์แบบ Tractionและ Hydraulic กลายมาเป็นลิฟต์ชนิดหลักของอาคารสูงไป

2.3.4.1 Traction Elevator

เป็นลิฟต์ที่ใช้ในอาคารตั้งแต่ระดับความสูงกลางๆไปถึงสูงมากใช้ระบบการลากดึงโดยรอกและใช้ตุ้มน้ำหนักเป็นตัวถ่วงน้ำหนักระบบความปลอดภัยสำหรับลิฟต์ประเภทนี้มีหลายทาง เช่น การมีเคเบิลในการดึงหลายๆ เส้นเบรคอัตโนมัติในกรณีฉุกเฉิน เป็นต้น ห้องเครื่องที่ใช้สำหรับการดึงลิฟต์ขึ้นลงจะอยู่ส่วนบนสุดของช่องลิฟต์แยกออกเป็นแบบมีเกียร์ (Geared Traction Elevator) และไม่มีเกียร์ (Gearless Traction Elevator)โดยแบบเกียร์จะใช้มอเตอร์ความเร็วสูงในการขับเคลื่อนลูกรอกในขณะที่แบบไม่มีเกียร์จะใช้มอเตอร์ความเร็วต่ำ

2.3.4.2 Hydraulic Elevator

นิยมใช้ในอาคารไม่กี่ชั้นในความเร็วที่ต่ำโดยห้องลิฟต์จะตั้งอยู่บนท่อ Hydraulic ที่ถูกดันให้สูงขึ้นเมื่ออัดน้ำมันเข้าสู่กระบอก Hydraulic และลดต่ำลงเมื่อดูดน้ำมันออก Hydraulic Elevator จะเคลื่อนตัวช้ากว่า Traction Elevator โดย Hydraulic Elevator จะมีความเร็วประมาณ 30-50 m/m ขณะที่ Traction Elevator สามารถเร็วได้มากกว่า 70 m/m จนถึง 300 m/m โดยทั่วไปแล้ว Hydraulic Elevator เหมาะสำหรับอาคารไม่สูง เช่น อพาร์ตเมนต์เล็กๆ ซึ่งสามารถประหยัดค่าใช้จ่ายลงได้ 15% เมื่อเทียบกับ Traction Elevator นอกจากนี้ Hydraulic Elevator มีความต้องการพื้นที่สำหรับห้องเครื่องน้อยกว่าเพราะ Traction Elevator ต้องการพื้นที่สำหรับรางและตุ้มน้ำ

2.4 การออกแบบทางสถาปัตยกรรม

การสัญจรทางตั้งของอาคารสูงขึ้นอยู่กับระบบลิฟต์เป็นหลัก การเลือกระบบลิฟต์จะต้องเริ่มพร้อมๆ ไปกับการออกแบบทางสถาปัตยกรรม อาจจะกล่าวได้ว่าพัฒนาการของอาคารเป็นไปพร้อมๆ กับพัฒนาการของลิฟต์ การออกแบบระบบลิฟต์มีความเกี่ยวพันกับการใช้งานอาคาร จำนวนผู้ใช้และพื้นที่ที่ต้องการใช้ลิฟต์โดยทั่วไปจำนวนลิฟต์จะมีประมาณ 1 ตัวต่อพื้นที่ 4645 ตร.ม. (50,000 sq.ft.) สำหรับอาคารสำนักงาน อาคารที่พักอาศัยจะมีการประมาณการจำนวนลิฟต์ประมาณ 100-200 ห้องต่อ ลิฟต์ 1 ตัว ในการคำนวณหาจำนวนที่แท้จริงของลิฟต์จะต้องลง

ในรายละเอียดของความหนาแน่น ของผู้ใช้ ความ สามารถในการรองรับในชั่วโมงเร่งด่วน รวมไปถึงการพิจารณาช่วงเวลาในการรอลิฟท์ชนิดของลิฟต์อัตราความเร็วและความสามารถในการรองรับในอาคารสูงชั้นมากขึ้นลิฟต์ก็จะต้องรองรับคนมากขึ้นการออกแบบให้ลิฟต์แต่ละชุดรองรับผู้ใช้ในแต่ละโซนจะทำให้ความจุของลิฟต์แต่ละตัวไม่ใหญ่เกินไปโดยปกติลิฟต์แต่ละชุดจะให้บริการสำหรับความสูงประมาณ 12-15 ชั้น ชุดแรกอาจจะรับส่งผู้โดยสารในช่วงชั้นล่าอีกชุดสำหรับช่วงกลางและชุดสุดท้ายสำหรับช่วงบนของอาคารเพราะฉะนั้นแต่ละชุดจะต้องมีโถงลิฟท์เพื่อรองรับผู้โดยสารโดยทั่วไปโซนสูงสุดจะถูกวางไว้ช่องกลางของแกนลิฟท์เพื่อให้โครงสร้างมีความสมดุลที่สุดเมื่อโซนอื่นถูกตัดออกและสามารถจัดเป็นพื้นที่ให้เช่าได้ง่ายอย่างไรก็ตามเทศบัญญัติระบุให้มีลิฟต์อย่างน้อยหนึ่งตัวที่สามารถจอดได้ทุกชั้นซึ่งส่วนใหญ่ลิฟท์บรรทุกของจะถูกออกแบบให้สามารถจอดได้ทุกชั้นและถูกใช้ในกรณีฉุกเฉินหรือเพลิงไหม้ด้วยมีคำแนะนำให้มีลิฟต์แบบนี้อย่างน้อยสองตัวเผื่อว่าตัวใดตัวหนึ่งมีปัญหาหรือต้องการการบำรุงรักษาจะได้มีตัวสำรองใช้งานได้

ระบบโถงลอยฟ้า (Sky Lobby) เป็นอีกตัวอย่างหนึ่งที่น่าสนใจโดยเฉพาะในอาคารแบบผสานประโยชน์ใช้สอย (Mixed-Use Building) โดยมีลิฟต์ทั่วไปจากชั้นล่างส่งผู้โดยสารไปยังส่วนโถงลอยฟ้าก่อนที่จะแยกย้ายไปใช้ลิฟต์ในแต่ส่วน (Local Elevator) แนวความคิดนี้จะช่วยลดความยุ่งยากในการจัดการและพื้นที่สำหรับส่วนโถงล่างของอาคาร โดยทั่วไปผังพื้นของอาคารสูงจะประกอบไปด้วยส่วนรอบนอกส่วนพื้นที่ภายในและส่วนแกนสัญจรทางตั้ง ส่วนแกนสัญจรนี้อาจจะแบ่งได้เป็นหลายรูปแบบ ส่วนใหญ่จะเป็นแบบรวมอยู่กลางอาคาร (Central-core Plan) และแยกส่วน (Split-core Plan) แบบรวมอยู่กลางอาคารจะเหมาะ สำหรับผังพื้นรูปสี่เหลี่ยมผืนผ้าเมื่อความลึกของอาคารถูกจำกัดโดยที่ตั้งหรือการออกแบบในขณะที่แบบแยกส่วนอาจจะเหมาะสำหรับผังพื้นที่เป็นจตุรัสมากกว่า

2.4.1 ตำแหน่งแกนสัญจร

ส่วนแกนสัญจรที่รวมอยู่กลางอาคารนอกจากจะประกอบด้วยลิฟท์ แล้วยังมีห้องช่องท่อสำหรับงานเครื่องกล (Mechanical Shafts) ท่อสำหรับงานสุขาภิบาล ห้องระบบไฟฟ้าโทรคมนาคม หรืออาจจะใช้เป็นห้องน้ำ ส่วนให้เช่าสำหรับพื้นที่เพิ่มขึ้นจากการแบ่งโซนลิฟท์ ส่วนแกนสัญจรนี้ควรออก แบบให้ตรงกันทุกๆชั้นเพื่อง่ายต่อการจ่ายงานระบบและไม่สิ้นเปลืองในการเปลี่ยนแนวท่อ ส่วนบันได ต้องออกแบบให้อยู่ห่างจากกันเพื่อควบคุมการหนีให้ทั่วถึงที่สุด

2.5 โครงสร้าง

แกนสัญจรถูกใช้เป็นส่วนประกอบหลักในการรับแรงทางแนวนอนของอาคารสูงในหลายๆ อาคารเพราะส่วนแกนนี้เป็นส่วนประกอบของอาคารที่สูงต่อเนื่องตลอดความสูง ส่วนใหญ่ จะถูกออกแบบให้เป็นคอนกรีตในลักษณะของ Shear Wall แต่ส่วนแกนคอนกรีตนี้อาจจะเสียความแข็งแรงโดยการเจาะช่องสำหรับช่วงเปิดของประตูลิฟต์ช่องท่อหรืออื่นๆ การเจาะเหล่านี้ไม่ควรมาก เกิน 30% และควรวางตำแหน่งให้เหมาะไม่ใกล้กันเกินไปโดยสรุปคือส่วนแกนควรสามารถคงความแข็งแรงในรูปแบบของโครงสร้างแบบท่อ (Tube Structure) เพื่อช่วยรับแรงทางแนวนอน (Lateral Load) ของอาคารโดยรวมอย่างไรก็ตามส่วนแกนสัญจรนี้ยังคงต้องทำหน้าที่แทนเสาในการรับแรงทางตั้งหรือน้ำหนักอาคารด้วยการใช้แกนสัญจรในระบบโครงสร้างอาคารสูงอาจจะแยกได้เป็น

2.5.1 Single Core Structure

ในอาคารที่ส่วนแกนสัญจรกลางเป็นส่วนโครงสร้างหลักของอาคาร ส่วนแกนนี้จึงต้องรับแรงกด(Compression) ได้ดีด้วย คอนกรีตเสริมเหล็กจึงเป็นวัสดุที่นิยมที่สุด

2.5.2 Core with Hinged Frame

ในกรณีนี้ส่วนแกนสัญจรทำหน้าที่ช่วยรับแรงกระทำทางแนวนอนขณะที่ส่วนโครงเฟรมของอาคารทำหน้าที่รับแรงกระทำทางแนวตั้งไปสู่ฐานรากโครงสร้างรอบนอกจึงสามารถออกแบบให้ไม่ซับ ซ้อนและเบาได้ ในอาคารที่ความสูงประมาณ 20 ชั้น แกนคอนกรีตส่วนกลางโดยทั่วไปจะเพียงพอ สำหรับการรับแรงกระทำทางแนวนอน

2.5.3 Core with Rigid Frame

ในกรณีนี้ส่วนแกนกลางและส่วนโครงเฟรมรอบนอกต่างก็ช่วยในการรับแรงกระทำทางแนวนอน

2.5.4 Core with Outriggers and Belt Truss

เป็นการใช้โครงทรัสเสริมความแข็งแรงให้แก่เสารอบนอกโดยเชื่อมต่อไปยังแกนสัญจรกลางทำให้การรับแรงเป็นอันหนึ่งอันเดียวกันการใช้แกนสัญจรกลางได้ถูกท้าทายโดยแนวความคิดใหม่ในช่วงปี 1960 สถาปนิกหลายๆ คนพยายามผสมผสานระบบแกนสัญจรกับระบบโครงสร้างสะพานอาคารสูงหลายๆ อาคารสูงได้ออก แบบให้ให้มีส่วนแกนสัญจรมากกว่าหนึ่งแห่ง

2.6 กฎหมาย

สำหรับในประเทศไทย อาคารสูง จะต้องมีบันไดหนีไฟไม่น้อยกว่า 2 บันได อยู่ห่างไม่เกิน 60 เมตร โดยวัดจากแนวทางเดินและต้องแสดงการคำนวณระบบบันไดหนีไฟให้เห็นว่าสามารถใช้ลำเลียงบุคคลทั้งหมดในอาคาร ออกนอกอาคารได้ภายในเวลา 1 ชั่วโมง ตัวบันไดต้องทำด้วยวัสดุทนไฟและ ไม่ผุกร่อน ห้ามสร้างบันไดหนีไฟเป็นบันไดเวียนหรือบันไดแนวดิ่ง (บันไดลิง) บันไดหนีไฟที่อยู่ภายใน อาคารต้องมีอากาศถ่ายเทจากภายนอกอาคารได้ หรือมีระบบอัดลมภายในช่องบันไดหนีไฟที่ทำงานโดยอัตโนมัติเมื่อเกิดเพลิงไหม้ บันไดหนีไฟที่อยู่ภายในอาคารต้องมีผนังกันไฟที่เป็นผนังทึบก่อด้วยอิฐธรรมดาหนาไม่น้อยกว่า 18 ซม. และไม่มีช่องที่ทำให้ไฟหรือควันผ่านไปได้หรือคอนกรีตเสริมเหล็ก หนาไม่น้อยกว่า 12 ซม. โดยรอบบันไดยกเว้นช่องระบายอากาศ

2.6.1 สำหรับระบบลิฟต์แล้วมีข้อกำหนดต่างๆ

2.6.1.1 ลิฟต์โดยสารและลิฟต์ดับเพลิงแต่ละชุดที่ใช้กับอาคารสูงให้มีขนาดมวลบรรทุกไม่น้อยกว่า 630 กิโลกรัม

2.6.1.2 ลิฟต์ดับเพลิงต้องจอดได้ทุกชั้นของอาคาร และต้องมีระบบควบคุมพิเศษสำหรับพนักงานดับเพลิงใช้ขณะเกิดเพลิงไหม้โดยเฉพาะ

2.6.1.3 บริเวณห้องโถงหน้าลิฟต์ดับเพลิงทุกชั้นต้องติดตั้งตู้สายฉีดน้ำดับเพลิงหรือหัวต่อสายฉีดน้ำดับเพลิงและอุปกรณ์ดับเพลิงอื่นๆ

2.6.1.4 ห้องโถงหน้าลิฟต์ดับเพลิงทุกชั้นต้องมีผนังหรือประตูที่ทำด้วยวัตถุทนไฟปิดกั้นมิให้เปลวไฟหรือควันเข้าไปได้มีหน้าต่างเปิดออกสู่ภายนอกอาคารโดยตรหรือมีระบบอัดลมภายในห้องโถงหน้าลิฟต์ดับเพลิงที่มีความดันลมขณะใช้งานไม่น้อยกว่า 3.86 เมกะปาสคาลและทำงานอัตโนมัติเมื่อเกิดเพลิงไหม้

2.6.1.5 ระยะเวลาในการเคลื่อนที่อย่างต่อเนื่องของลิฟต์ดับเพลิงระหว่างชั้นล่างสุดกับชั้นสูงสุดของอาคารต้องไม่เกิน 1 นาที

2.6.1.6 ในเวลาปกติลิฟต์ดับเพลิงสามารถใช้เป็นลิฟต์โดยสารได้

2.6.1.7 ในปล่องลิฟต์ห้ามติดตั้งท่อสายไฟฟ้าท่อส่งน้ำท่อระบายน้ำและอุปกรณ์ต่างๆเว้นแต่เป็นส่วนประกอบของลิฟต์หรือจำเป็นสำหรับการทำงานและการดูแลรักษาลิฟต์

2.6.1.8 ระบบและอุปกรณ์การทำงานที่ให้ความปลอดภัยด้านสวัสดิภาพและสุขภาพของผู้โดยสาร

2.6.1.9 ต้องมีระบบการทำงานที่จะทำให้ลิฟต์เคลื่อนมาหยุดตรงที่จอดชั้นระดับดินและประตูลิฟต์ต้องเปิดโดยอัตโนมัติเมื่อไฟฟ้าดับ

2.6.1.10 ต้องมีสัญญาณเตือนและลิฟต์ต้องไม่เคลื่อนที่เมื่อบรรทุกเกินพิกัด

2.6.1.11 ต้องมีอุปกรณ์ที่จะหยุดลิฟต์ได้ในระยะที่กำหนดโดยอัตโนมัติเมื่อตัวลิฟต์มีความเร็วเกินพิกัด

2.6.1.12 ต้องมีระบบป้องกันประตูลิฟต์หนีบผู้โดยสาร

2.6.1.13 ลิฟต์ต้องไม่เคลื่อนที่เมื่อประตูลิฟต์ปิดไม่สนิท

2.6.1.14 ประตูลิฟต์ต้องไม่เปิดขณะลิฟต์เคลื่อนที่หรือหยุดไม่ตรงที่จอด

2.6.1.15 ต้องมีระบบการติดต่อกับภายนอกห้องและสัญญาณแจ้งเหตุขัดข้อง

2.6.1.16 ต้องมีระบบแสงสว่างฉุกเฉินในห้องลิฟต์และหน้าชั้นที่จอด

2.6.1.17 ต้องมีระบบการระบายอากาศในห้องลิฟต์ตามที่กำหนด

2.6.1.18 จัดให้มีคำแนะนำอธิบายการใช้ การขอความช่วยเหลือ การให้ความช่วยเหลือ และข้อห้ามใช้ลิฟต์

2.6.1.19 การใช้ลิฟต์และการขอความช่วยเหลือให้ติดไว้ในห้องลิฟต์

2.6.1.20 การให้ความช่วยเหลือให้ติดไว้ในห้องจักรกลและห้องผู้ดูแลลิฟต์

2.6.1.21 ให้ติดข้อห้ามใช้ลิฟต์ไว้ที่ข้างประตูลิฟต์ด้านนอกทุกชั้น

2.7 ขับเคลื่อนรีเจนเนอเรทีฟในลิฟต์

ลิฟต์เป็นพาหนะที่ใช้ในการขนส่งในแนวดิ่งด้วยเหตุนี้ลิฟต์จึงเป็นพาหนะที่ใช้ภายในสิ่งปลูกสร้างที่มีความสูง ไม่ว่าจะเป็นในอาคารสำนักงานห้างสรรพสินค้าโรงงานหอสูงเป็นต้นและโดยเหตุที่ลิฟต์มีการใช้ในอาคารสำนักงานจำนวนมากและเป็นอุปกรณ์หรือเครื่องจักรกลหนึ่งที่ใช้พลังงานไฟฟ้าในระบบไฟฟ้ากำลังเป็นอันดับที่ 3 รองจากระบบปรับอากาศและระบบไฟฟ้าแสงสว่างดังนั้นการพัฒนาเทคโนโลยีการใช้พลังงานของลิฟต์ให้มีประสิทธิภาพจึงเป็นการพัฒนาที่สำคัญต่อการอนุรักษ์พลังงาน

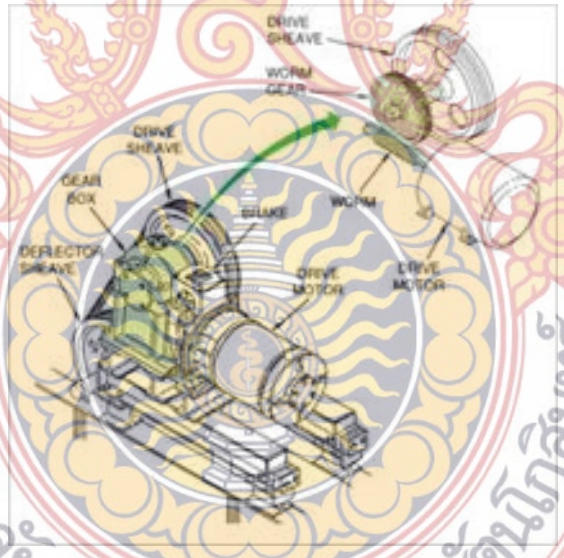
2.7.1 ลิฟต์ที่ใช้ระบบไฮดรอลิกมีอุปกรณ์ต้นกำลังและอุปกรณ์ส่งกำลังดังต่อไปนี้

2.7.1.1 มอเตอร์เหนี่ยวนำ (Induction Machine) ที่ใช้ในการส่งน้ำมันไฮดรอลิกไปยังอุปกรณ์ไฮดรอลิก

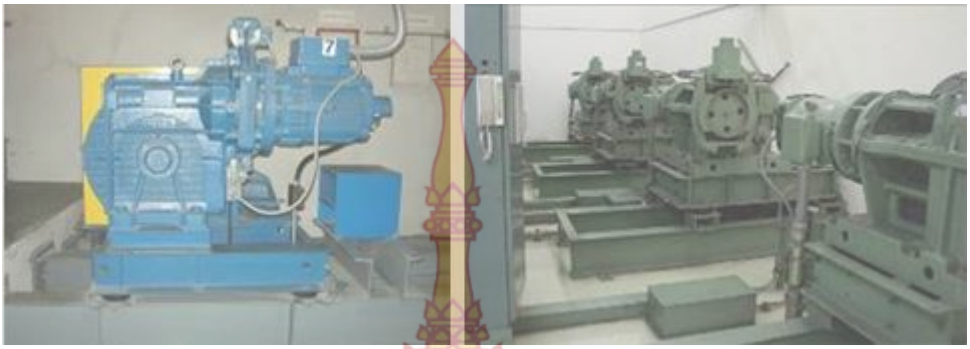
2.7.1.2 ระบบไฮดรอลิกซึ่งประกอบไปด้วยชุดวาล์วและมอเตอร์ไฮดรอลิก (Hydraulic Gear Motor)โดยเหตุที่ระบบไฮดรอลิกมีการส่งผ่านกำลังที่ไม่มีประสิทธิภาพเมื่อเทียบกับการส่งผ่านกำลังไปยังเฟืองทดโดยตรงทำให้ปัจจุบันลิฟต์ที่ใช้ระบบไฮดรอลิกแทบไม่มีการใช้งานอีกแล้วลิฟต์ที่ใช้เฟืองทด (Geared Machine) ซึ่งปัจจุบันนับว่าเป็นประเภทลิฟต์ที่มีการใช้งานเป็นส่วนใหญ่ในประเทศไทยจะมีอุปกรณ์ต้นกำลังและอุปกรณ์ส่งกำลังดังต่อไปนี้

2.7.1.3 มอเตอร์เหนี่ยวนำซึ่งจะเป็นมอเตอร์ที่มีแรงบิดขณะเริ่มเดินเครื่องสูง (High Starting Torque)

2.7.1.4 ชุดเฟืองทด (Gear Box) ซึ่งนิยมใช้ Worm Gearเพื่อทดความเร็วรอบลงจากความเร็วรอบของมอเตอร์



ภาพที่ 2.13 ลักษณะและองค์ประกอบลิฟต์ประเภทใช้เฟืองทด



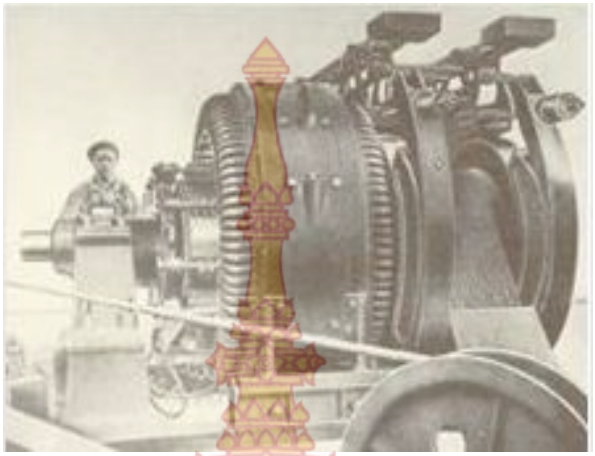
ภาพที่ 2.14 ตัวอย่างระบบต้นกำลังของลิฟต์ของอาคารสำนักงาน

ลิฟต์ที่ไม่ใช้เฟืองทด (Gearless Machine) ซึ่งเป็นลิฟต์ที่มีการใช้งานเป็นส่วนใหญ่ในต่างประเทศและกำลังได้รับความนิยมเพิ่มมากขึ้นในประเทศไทยมีอุปกรณ์ต้นกำลังและอุปกรณ์ส่งกำลังดังต่อไปนี้

มอเตอร์เหนี่ยวนำ (Induction Motor) หรืออาจเป็นมอเตอร์ซิงโครนัสชนิดที่ใช้แม่เหล็กถาวรในชุดของโรเตอร์(Permanent Magnet Synchronous Motor, PMSM)

โดยเหตุที่ลิฟต์ชนิดนี้ไม่มีเฟืองทดความเร็วรอบของมอเตอร์ระบบจ่ายไฟฟ้ากำลังให้กับมอเตอร์จะต้องเป็นระบบที่สามารถปรับแรงดันและความถี่ของแรงดันไฟฟ้าให้เหมาะสมกับจุดทำงานของมอเตอร์โดยทั่วไปจะเรียกระบบจ่ายไฟฟ้ากำลังชนิดนี้ว่า VVVF หรือ VariableVoltage Variable Frequencyสำหรับลิฟต์ที่ไม่ใช้เฟืองทดที่ใช้มอเตอร์ PMSM จะต้องมีตัวต้านทานที่ใช้การระบายความร้อนในการเบรก (Dynamic Brake Resistor) เนื่องจากพลังงานกลในโรเตอร์ของมอเตอร์ซิงโครนัสจะถูกแปลงเป็นพลังงานไฟฟ้า ซึ่งเมื่อจ่ายพลังงานไฟฟ้าให้กับตัวต้านทานจะทำให้พลังงานไฟฟ้าถูกแปลงไปเป็นพลังงานความร้อนทำให้พลังงานกลที่สะสมอยู่ในโรเตอร์หมดไปและหยุดหมุนในจุดที่ลิฟต์จอดได้พอดี

โดยทั่วไปลิฟต์ที่ไม่ใช้เฟืองทดสามารถลดการสูญเสียพลังงานที่ใช้ในการส่งผ่านเฟืองทดอีกทั้งลิฟต์ที่ใช้เฟืองทดซึ่งมอเตอร์ต้องหมุนด้วยความเร็วรอบที่สูงทำให้เกิดการสูญเสียพลังงานอันเนื่องจากความเสียดทานระหว่างผิวสัมผัส (Friction Loss) และความเสียดทานลมระหว่างโรเตอร์และสเตเตอร์ (Windage Loss)



ภาพที่ 2.15 Gearless Machine ที่ใช้งานตั้งแต่ พ.ศ.2446 ที่อาคาร Beaver ใน New Yoke

ภาพที่ 2.16 มอเตอร์หมุนด้วยความเร็ว 375 รอบต่อวินาที

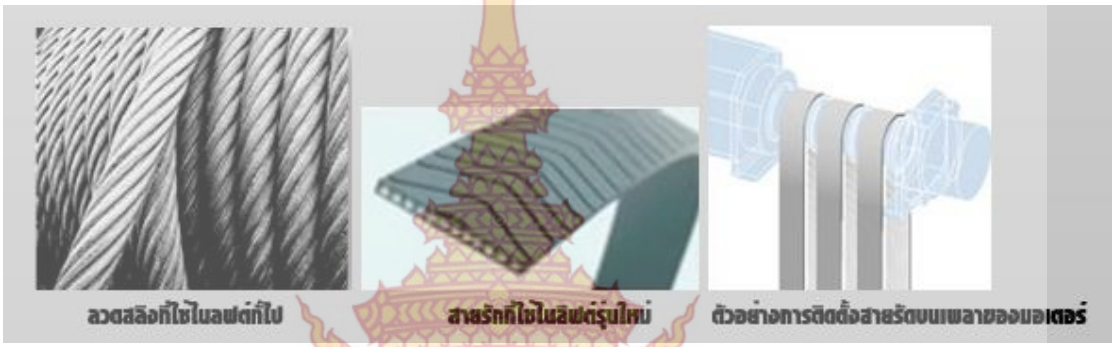
ใช้โพลียูเรเทนเคลือบลวดเหล็กเพื่อลดการสัมผัสหรือการเสียดสีกันโดยตรงระหว่างดุมล้อหมุน (Main Sheave) กับตัวลวดซึ่งต่างจากสลิงที่โลหะสัมผัสกับดุมล้อหมุนทำให้เกิดการสึกหรอของสลิงและดุมล้อดังนั้นเพื่อป้องกันการสึกหรอจึงต้องใช้น้ำมันหล่อลื่นระหว่างผิวสัมผัส

สายรัดประกอบด้วยลวดเหล็กเล็กๆที่ทนแรงดึงสูงประมาณ 600 เส้นเรียงตัวกันอยู่ในแนวระนาบ

ลดข้อจำกัดในการเลือกใช้ดุมล้อที่มีขนาดเล็กลงเนื่องจากขนาดหน้าตัดของสายลัดโพลียูเรเทนที่มีขนาดเล็กกว่าสลิงแบบเดิมทำให้สามารถลดรัศมีความโค้งของดุมล้อลงได้

สายรัดมีขนาดเบากว่าสลิงทั่วไป 20% ทำให้อุปกรณ์ต้นกำลังทำงานน้อยลงและมีขนาดเล็กลง (Compact Prime Mover) ตามน้ำหนักที่ลดลง

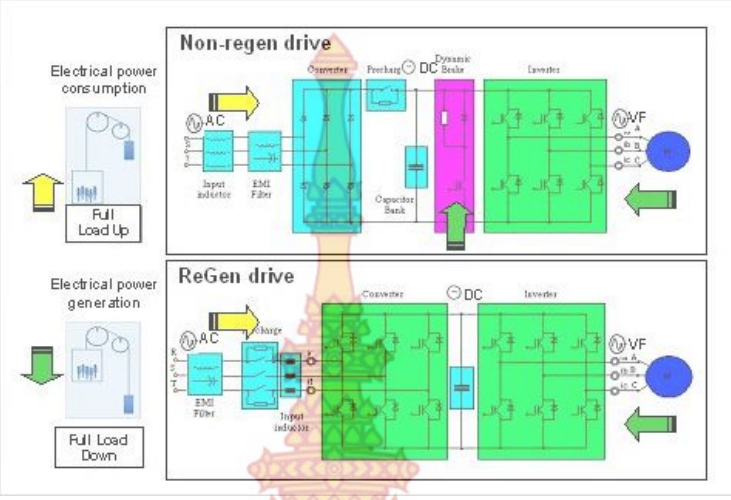
อายุการใช้งานนานกว่าสลิงทั่วไป 2-3 เท่า



ภาพที่ 2.17 สลิงและสายรัดแบบใหม่ที่ใช้กับระบบลิฟต์อนุรักษ์พลังงานในระบบลิฟต์

2.7.2 เทคโนโลยีอนุรักษ์พลังงานในระบบลิฟต์

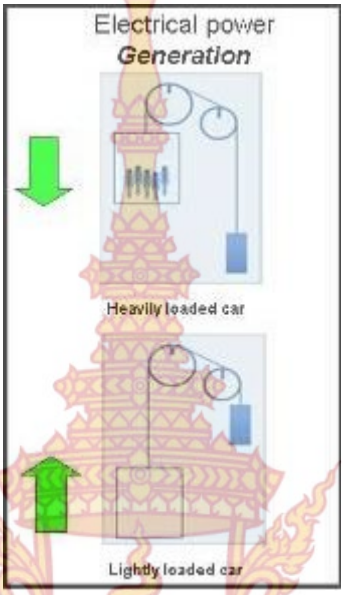
การพัฒนาเทคโนโลยีด้านการอนุรักษ์พลังงานที่เกี่ยวข้องกับระบบลิฟต์ ยังรวมถึงการพัฒนาอุปกรณ์ควบคุมการทำงานของลิฟต์แบบวงรอบปิด (Closed Loop Control) เพื่อใช้ในการควบคุมการทำงานของมอเตอร์ให้ทำงานด้วยความเร็วในการเคลื่อนที่ที่สัมพันธ์กับตำแหน่งจุดจอดของลิฟต์ทำให้ลดการสูญเสียพลังงานที่ต้องใช้ในตัวต้านทานที่ใช้ในการเบรกและนอกจากนี้ยังได้นำพลังงานศักย์ที่สะสมอยู่ในน้ำหนักบรรทุกกลับมาใช้ในการผลิตกระแสไฟฟ้าป้อนกลับเข้าสู่ระบบไฟฟ้ากำลังของลิฟต์และรวมถึงระบบไฟฟ้าของอาคารสำนักงานรวมทั้งลดความร้อนที่เกิดขึ้นจากการเบรคลิฟต์ด้วยการแปลงพลังงานศักย์ที่ได้กลับไปเป็นพลังงานไฟฟ้าแทนที่จะทำให้เกิดพลังงานความร้อนที่ระบบเบรคซึ่งจะช่วยลดความต้องการใช้พลังงานในการระบายความร้อนที่ชุดเบรคในห้องลิฟต์ ได้ด้วย ระบบดังกล่าวสามารถเรียกว่าระบบ Regenerative Drive หรือเรียกสั้นๆว่า Regen Drive



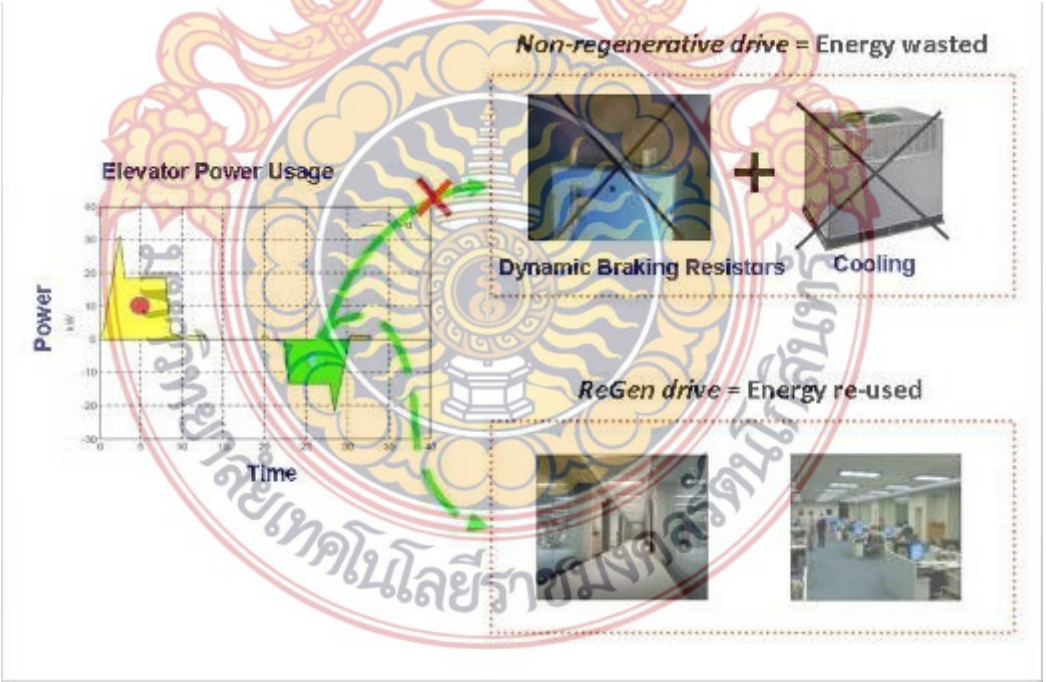
ภาพที่ 2.18 หลักการระบบ Regenerative Drive ของลิฟต์

ตัวอย่างกราฟแสดงกำลังไฟฟ้าที่ใช้ในระบบ Regenerative Drive พบว่าในช่วงที่มีการใช้พลังงานจากลิฟต์ จะใช้พลังงานไฟฟ้าประมาณ 160 กิโลวัตต์วินาที ในขณะที่ลิฟต์จ่ายกำลังไฟฟ้ากลับเข้าสู่ระบบไฟฟ้ากำลัง (พื้นที่กราฟสีเขียว) จะใช้พลังงานไฟฟ้าประมาณ 100 กิโลวัตต์วินาที

ภาพที่ 2.19 การเคลื่อนที่ของลิฟต์ที่ใช้พลังงานไฟฟ้า



ภาพที่ 2.20 การเคลื่อนที่ของลิฟต์ที่สามารถผลิตไฟฟ้าได้



ภาพที่ 2.21 Regenerative Drive สามารถเปลี่ยนพลังงานสูญเสียมาเป็นพลังงานที่นำไปใช้ได้

2.8 การบำรุงรักษาลิฟต์

2.8.1 การเช็คการทำงานของลิฟต์ก่อนการใช้งานประจำวัน

2.8.1.1 ตรวจปุ่มกดทำงานถูกต้อง

2.8.1.2 ตรวจเช็คแผงควบคุม (Switch Box) จะต้องล็อคอยู่ตลอดเวลา

2.8.1.3 ตรวจเช็คแสงสว่างและพัดลมระบายอากาศภายในห้องโดยสาร

2.8.1.4 ตรวจเช็คการทำงานของ Safety shoes กับ Door Sensor

2.8.1.5 ตรวจเช็คธรณีประตูจะต้องไม่มีเศษวัสดุ

2.8.1.6 ตรวจเช็คการทำงานของโทรศัพท์

2.8.1.7 ทดลองลิฟต์วิ่งขึ้น-ลง ว่าเรียบร้อยดีไม่มีเสียงและไม่สั่น

2.8.1.8 ตรวจดูกุญแจเปิดประตูลิฟต์

2.8.2 การบำรุงรักษาลิฟต์ทุกระยะ 1 เดือน

2.8.2.1 ตรวจเช็คการทำงานของวงจรเซฟตี้ทั้งหมด

2.8.2.2 ตรวจเช็คสวิทซ์หน้าคอนแทคและกลไกของดอร์ล็อค

2.8.2.3 ตรวจเช็คระดับชั้น (การจอดเสมอระดับขั้นหรือไม่)

2.8.2.4 ตรวจการทำงานของชุดเซฟตี้ชูส์/ไลท์เรย์

2.8.2.5 ตรวจระบบไฟแสงสว่างฉุกเฉิน/กระดิ่ง อินเตอร์คอมฯ/แบตเตอรี่

2.8.2.6 ตรวจผ้าเบรคและระยะการทำงานของเบรค

2.8.2.7 ตรวจสัญญาณบอกชั้นทิศทางการขึ้นลงและสัญญาณเสียงแจ้งเตือน

2.8.2.8 ตรวจการทำงานของปุ่มกดหน้าชั้นสัญญาณบอกชั้น

2.8.2.9 ตรวจเช็คอุณหภูมิมอเตอร์และพัดลมระบายอากาศ

2.8.2.10 ตรวจเช็คและทดสอบการทำงานของชุดกัฟเวอเนอร์โดยวิธี manual

2.8.3 การบำรุงรักษาทุกระยะ 3 เดือน

2.8.3.1 ตรวจเช็คสภาพการทำงานของหน้าคอนแทครีเลย์

2.8.3.2 ตรวจเช็คทำความสะอาดแผงวงจรไฟฟ้า/ขั้วแบตเตอรี่

2.8.3.3 ตรวจเช็คขั้นตอนการทำงานของระบบทั้งหมด

2.8.3.4 ตรวจเช็คการทำงานระบบแสงสว่างฉุกเฉิน

2.8.3.5 ตรวจเช็คชูส์ประตูโรลเล่อร์ประตู

2.8.3.6 ตรวจเช็คทำความสะอาดราง/รอกแขวนประตูและหล่อลื่นระบบประตู

2.8.3.7 ตรวจเช็คสภาพการสึกหรอและการทำงานชุดกัฟเวอเนอร์

2.8.4 การบำรุงรักษาทุกระยะ 6 เดือน

2.8.4.1 ตรวจเช็คปรับตั้งลิมิตสวิทซ์

2.8.4.2 ตรวจเช็คไฟแสงสว่างในช่องลิฟต์/บนหลังคาตัวลิฟต์

2.8.4.3 ตรวจเช็คระดับน้ำมันของบัฟเฟอร์

2.8.4.4 ตรวจเช็คสภาพของฉนวนที่สายเทรเวลลิ่งเคเบิล

2.8.4.5 ตรวจเช็คสภาพและความดึงของลวดสลิงกัฟเวอเนอร์

2.8.4.6 ทดสอบการทำงานของชุดป้องกันมอเตอร์

2.8.4.7 ขันตรวจเทอร์มินอลของมอเตอร์ทุกตัว

2.8.5 การบำรุงรักษาทุกระยะ 12 เดือน

2.8.5.1 ตรวจเช็คการทำงานของโอเวอร์โหลดรีเลย์

2.8.5.2 ตรวจเช็คสกรูจุดต่อสาย/ระดับแรงดันไฟฟ้าภายในตู้คอนโทรลไฟฟ้า

2.8.5.3 ตรวจทำความสะอาดรางตัวลิฟต์และรางตุ้มน้ำหนัก

2.8.5.4 ตรวจเช็คทำความสะอาดรอกขับ

2.8.5.6 ตรวจเช็คขนาดเส้นผ่าศูนย์กลางของลวดสลึงขับลิฟต์

2.8.5.7 ตรวจเช็คระดับน้ำมันเกียร์และเปลี่ยนถ่ายตามระยะเวลาที่กำหนด

2.8.5.8 ขันตรวจความแน่นของน็อตยึดต่าง ๆ

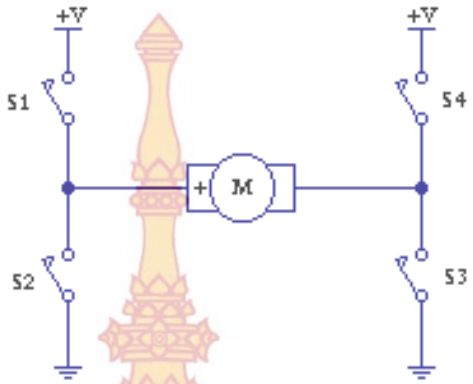
2.8.5.9 ตรวจเช็คมอเตอร์พัดลมระบายความร้อน และปริมาณแรงลม

2.8.5.10 ถอดทำความสะอาดฟิวส์/เซอร์กิตเบรดเกอร์

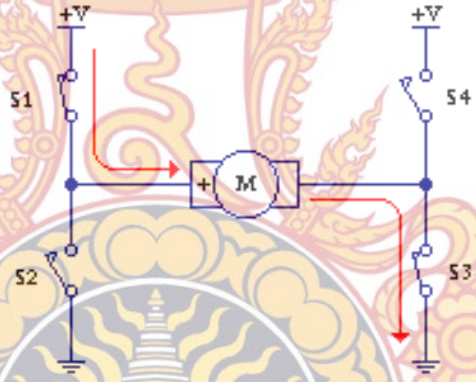
2.9 หลักการทำงานของระบบมอเตอร์ไฟฟ้ากระแสตรง(DC motor)

2.9.1 หลักการทำงานของระบบมอเตอร์ไฟฟ้ากระแสตรง(DC motor)

ในการควบคุมทิศทางการหมุน DC motor นั้นผู้จัดทำสามารถควบคุมได้โดยการควบคุมการจ่ายไฟให้กับ DC motor ในที่นี้จะพูดถึงการทำงานของ H-Bridge Switching เป็นวงจรที่ควบคุมการจ่ายไฟให้กับ DC motor ด้วยการ switching เพื่อเปลี่ยนทิศทางของกระแสที่ไปเลี้ยง DC motor ซึ่งจะมีผลให้ DC motor สามารถหมุนกลับทางได้ซึ่งวงจรจะประกอบไปด้วย switch 4 ตัวที่ต่อดังภาพต่อไปนี้

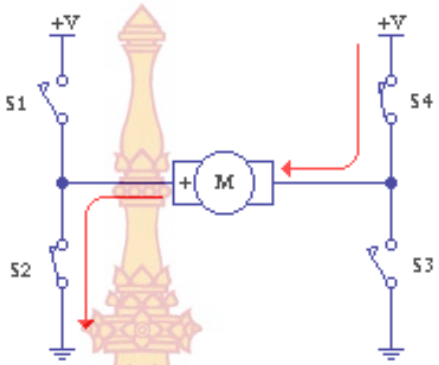


ภาพที่ 2.22 แสดงรูปวงจรของ H-Bridge Switching



ภาพที่ 2.23 การไดร์ฟมอเตอร์ในทิศตามเข็มนาฬิกา

จากภาพที่ 2.5 จะเห็นว่ายังไม่มีการจ่ายไฟให้กับ DC motor จะยังไม่มีการหมุน แต่เมื่อมีการจ่ายไฟดังรูปด้านล่าง DC motor จะหมุนในลักษณะตามเข็มนาฬิกาดังภาพที่ 2.23



ภาพที่ 2.24 การไดร์ฟมอเตอร์ในทิศทวนเข็มนาฬิกา

แต่เมื่อใดก็ตามที่มีการ switch ดังภาพที่2.23ด้านล่างจะเกิดการป้อนกระแสกลับทางให้กับ DC motor ทำให้ DC motor หมุนในทิศทางทวนเข็มนาฬิกา

2.10 IC Drive motor IC-L298

วงจรที่ใช้ Drive DC motor เลือกใช้ IC เบอร์ L298 เพื่อใช้ขับมอเตอร์ของแขนกล

ภาพที่ 2.25 แสดงรูปไอซีไดร์ฟ L298

2.10.1 คุณสมบัติทั่วไปของ ไอซีไดร์ฟ L298

2.10.1.1 ทำงานที่แรงดันสูงสุดได้ถึง 46 V

2.10.1.2 สามารถทำงานที่กระแสสูงสุดได้ถึง 4 A

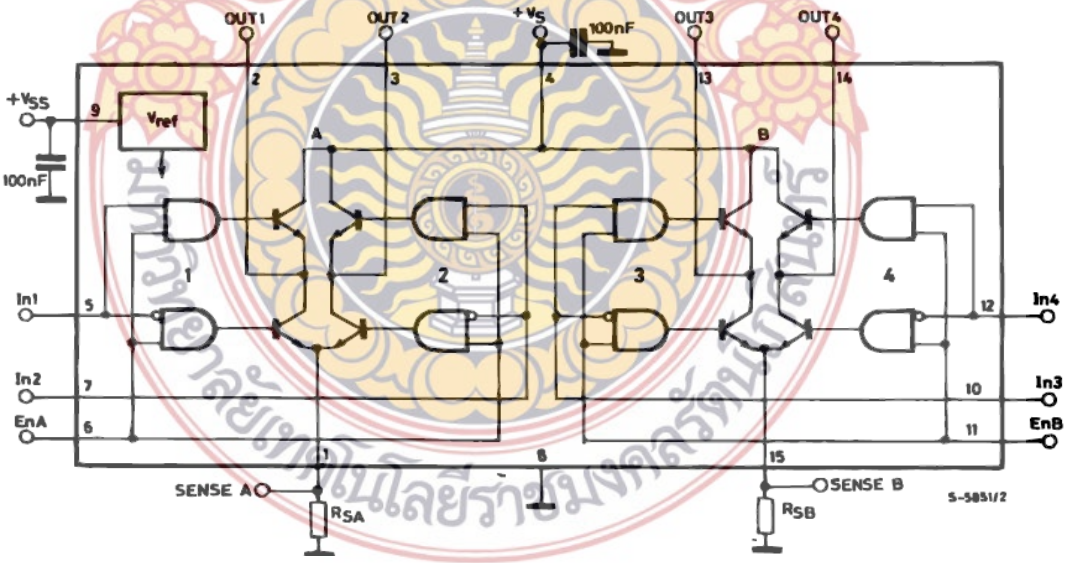
2.10.1.3 แรงดันอิ่มตัวต่ำ

2.10.1.4 ป้องกัน Overtemperature

2.10.1.5 Logic เป็น “ 0 ” ที่แรงดันต่ำกว่า 1.5 V (ป้องสัญญาณรบกวนที่ amplitude สูงๆได้)

ถ้าเปรียบ Controller เป็นดังสมองมนุษย์ IC L298ก็เป็นเหมือนประสาทที่อยู่ตามส่วนต่างๆของร่างกายที่จะนำสัญญาณจากสมองมนุษย์มาขับให้แขนทำงานเนื่องจาก IC Controller ไม่สามารถที่ขับกระแสได้สูงพอที่จะใช้ขับ DC motor จึงได้นำเอา IC L298 มาใช้ในการขับ DC motor โดยรับคำสั่งจาก Controller

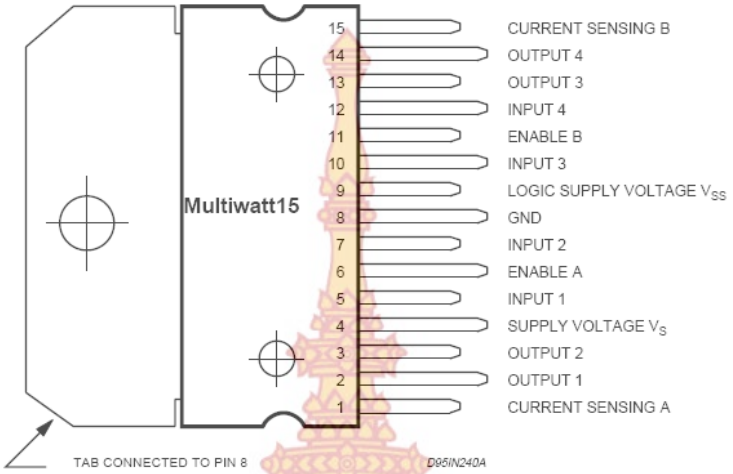
ลักษณะการทำงานของ L298 จะเหมือนแนนเกตต่ออยู่กับทรานซิสเตอร์โดยไอซีนี้จะทำหน้าที่เป็นตัวขับกระแสและเป็นสวิตช์เปิดปิดให้มอเตอร์ทำงานหรือหยุดทำงานอินพุตที่ได้รับจะต่างกัน เป็น 5 V กับ 0 V ซึ่งจะทำให้มอเตอร์ทำงานทันที การออกแบบภาครับของวงจรนี้คือ จะใช้บัฟเฟอร์เป็นตัวทำให้แรงดันมีความสมดุล ซึ่งจะรับมาจากตัวไมโครคอนโทรลเลอร์นั้นเอง



ภาพที่ 2.26 แสดง Block Diagram L298

ตารางที่ 2.1 แสดง Pin Function

MW.15	PowerSO	Name	Function
1;15	2,19	Sense A; Sense B	เชื่อมต่อกับตัวต้านทานเพื่อระหว่าง pin และ ground ควบคุมกระแสของโหลด(load)
2;3	4,5	Out 1; Out 2	Outputs ของ Bridge A; กระแสไหลไปยังโหลดที่ต่ออยู่ระหว่าง 2 pin สังเกตที่ pin1
4	6	V_S	แหล่งจ่ายแรงดันสำหรับ Output ไม่มีความเหนี่ยวนำ มีตัวเก็บประจุมากกว่า 100nF ต่ออยู่ระหว่าง pin และ ground
5;7	7,9	Input 1; Input 2	TTL รวมกับ Inputs ของ Bridge A.
6;11	8,14	Sense A; Sense B	TTL รวมกับ Enable Input: the L state disables the bridge A(enable A) and/or the bridge B (enable B).
8	1,10,11,20	GND	Ground.
9	12	VSS	แหล่งจ่ายแรงดันสำหรับ Logic Blocks. A มีตัวเก็บประจุมากกว่า 100nFต่ออยู่ระหว่าง pin นี้และ ground
10;12	13;15	Input 3; Input 4	TTL ร่วมกับ Inputs ของ Bridge B.
13;14	16;17	Out 3; Out 4	Outputs ของ Bridge B. กระแสไหลไปยัง Load ที่ต่ออยู่ระหว่าง 2 pin นี้สังเกตที่ pin15
-	3;18	N.C.	ไม่มีการเชื่อมต่อ



ภาพที่ 2.27 แสดง Pin Diagram L298

ตารางที่ 2.2 คุณสมบัติของ ICL298

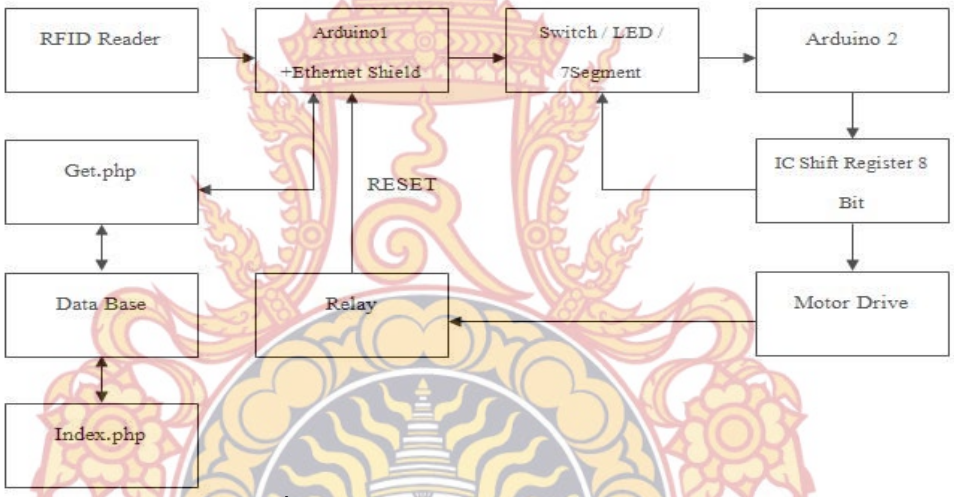
Symbol	Parameter	Value	Unit
V_S	พิกัดแรงดัน	50	V
V_{SS}	พิกัดแรงดันสัญญาณ	7	V
V_I , V_{en}	สัญญาณ Input และ Enable	-0.3 to 7	V
I_O	พิกัดกระแส Output		
	– Non Repetitive (t = 100ms)	3	A
	–Repetitive (80% on –20% off; ton = 10ms)	2.5	A
	–DC Operation	2	A
V_{sens}	แรงดัน Sensing	-1 to 2.3	V
P_{tot}	ผลรวมพลังงาน Dissipation ที่ 75°C	25	W
T_{op}	อุณหภูมิในช่วงที่สามารถทำงานได้	-25 to 130	°C
T_{stg} ,T_j	อุณหภูมิที่ใช้เก็บ	-40 to 150	°C

บทที่ 3
วิธีการดำเนินงาน

วิธีการดำเนินงานการสร้างระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID ขึ้นมานั้น ได้ศึกษารายละเอียดและลักษณะหน้าที่ของงาน ทฤษฎีต่าง ๆ ที่เกี่ยวข้อง สรุปขั้นตอนการทำงานได้ดังนี้

3.1 ขั้นตอนการดำเนินงาน

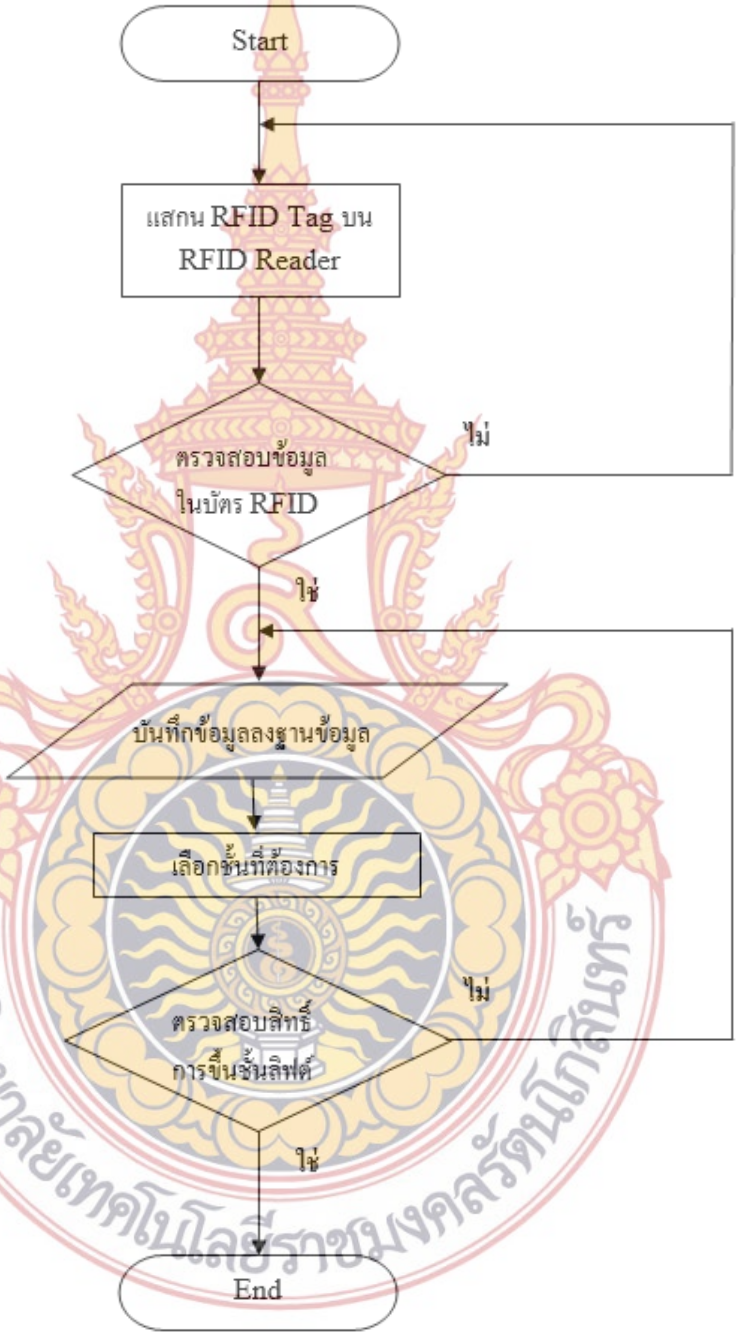
3.1.1 โครงสร้างของบล็อกไดอะแกรม ระบบควบคุมการเลือกชั้นลิฟต์ ด้วย RFID



ภาพที่ 3.1 โครงสร้างของบล็อกไดอะแกรม

จากภาพที่ 3.1 จะแสดงให้เห็นถึงการทำงานของระบบควบคุมการเลือกชั้นลิฟต์ ด้วย RFID ที่เริ่มทำงานตั้งแต่รับข้อมูลจากTag RFID จากนั้น Arduino1 จะทำการประมวณผลและทำการเชื่อมต่อไปยัง Arduino Ethernet Shield โดยใช้ไฟล์ GET.php ในการติดต่อและบันทึกข้อมูลลงใน DATA BASE เพื่อนำไปใช้แสดงในหน้าเว็บ INDEX.php ในขณะเดียวกันจะส่งค่าที่ได้ไปยังสวิทช์กด โดยจะแสดงผลออกทาง LED เมื่อไฟ LED ติดสว่างแล้ว ก็จะสามารถกดได้แค่ปุ่มที่มีไฟแสดงอยู่ เมื่อกดปุ่มแล้ว Arduino 2 จะส่งค่าไปยัง 74HC595 แสดงปุ่มนั้นๆที่ได้ทำการกดเลือกไป และจะแสดงผลเป็นตัวเลขนับจำนวนชั้น ด้วย 7 Segment 74HC595 ยังทำหน้าที่ส่งค่าไปยัง Motor Drive เพื่อขับ มอเตอร์ ขึ้น/ลง และ RESET ข้อมูล Tag ที่ได้จำไว้ตอนเริ่มต้นด้วย

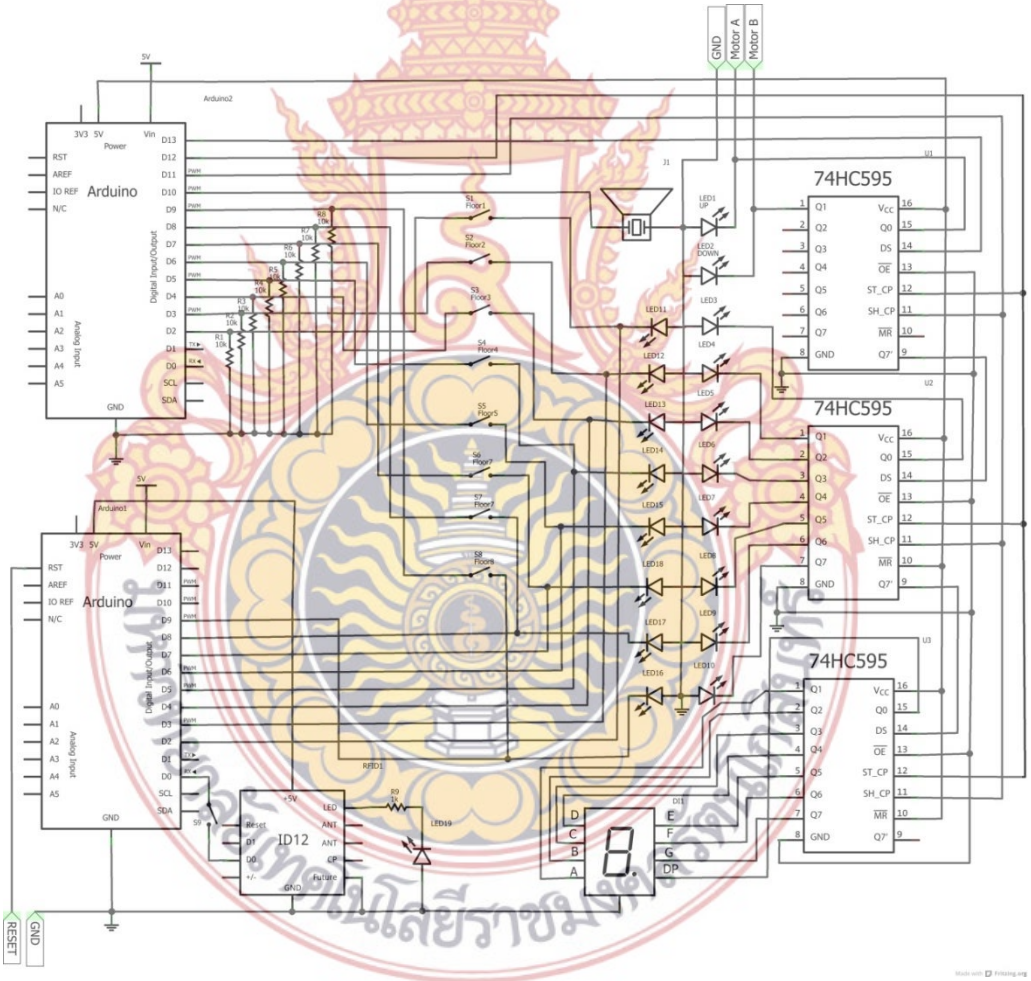
3.2 Flow Chart แสดงการทำงานของระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID



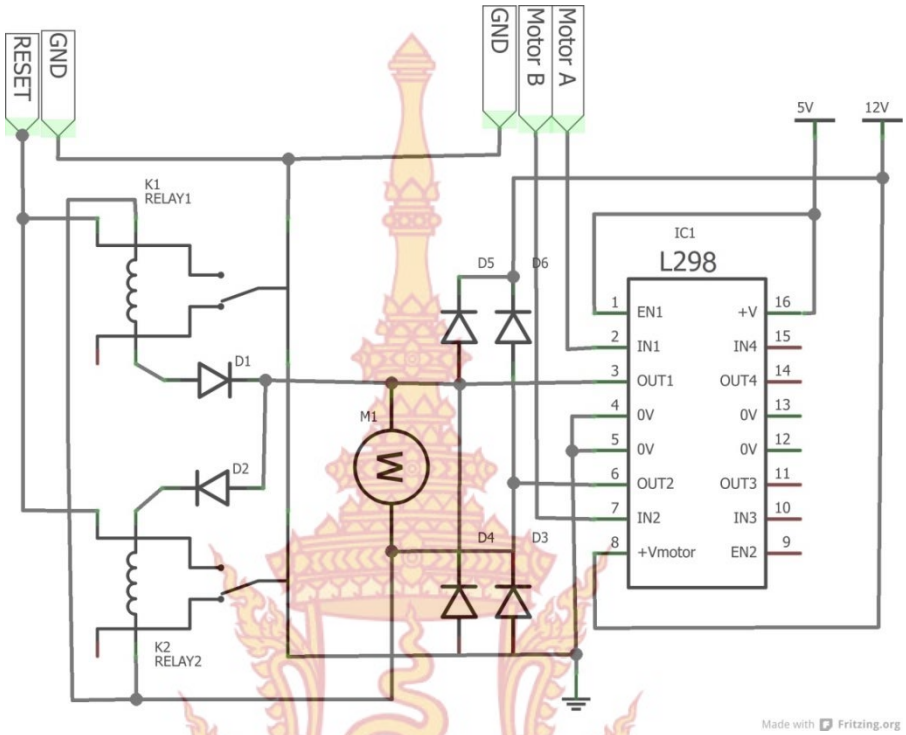
ภาพที่ 3.2 Flow Chart แสดงการทำงานของระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID

จากภาพที่ 3.2 แสดง Flow Chart แสดงการทำงานของระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID ในส่วนของ Flow Chart จะเริ่มต้นด้วยการแสกน Tag RFID บน RIFD Reader เพื่อตรวจสิทธิ์บัตรว่าบัตรสามารถขึ้น-ลงชั้นไหนได้บ้าง ถ้าบัตรไม่มีข้อมูลก็กลับไปแสกนบัตรใหม่แต่ถ้าตรวจสอบบัตรแล้วสามารถเลือกชั้นลิฟต์ได้ก็เลือกชั้นที่ต้องการจะขึ้น แต่ถ้าเลือกชั้นที่ไม่ถูกต้องกับที่ Tag RFID เก็บข้อมูลบัตรไว้ก็กลับไปเลือกชั้นใหม่ ถ้าเลือกชั้นที่บัตรกำหนดสิทธิ์ไว้ลิฟต์ก็จะทำงานขึ้นไปยังชั้นที่บัตรกำหนดสิทธิ์ไว้ และบันทึกข้อมูลการใช้งานลงในฐานข้อมูล

3.3 หลักการทำงานของวงจร



ภาพที่ 3.3 แสดงภาพการต่อวงจรในการทำงานทั้งหมด



ภาพที่ 3.4 แสดงวงจรขับมอเตอร์ด้วย IC L298

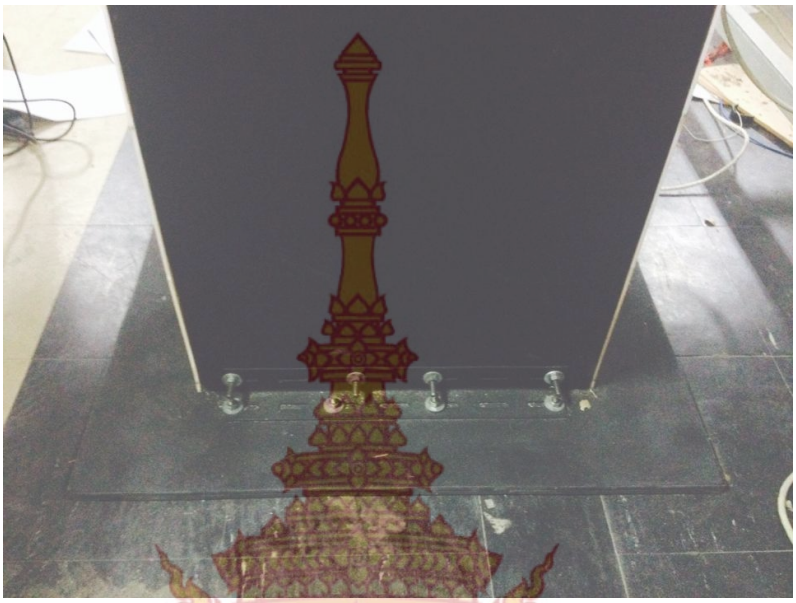
3.4 หลักการทำงานของวงจรทั้งหมด

เมื่อทำการใช้ Tag แตะลงบน Reader ID12 โปรแกรมจาก Arduino ตัวที่1 จะประมวลผลว่า Tag ที่นำมาแตะนั้นสามารถไปยังชั้นไหนได้บ้าง โดยArduino ตัวที่1 จะประกอบกับ Arduino Ethernet Shield ซึ่งทำหน้าที่เชื่อมต่อฐานข้อมูล จากนั้นไฟ LED จะติดสว่างก็ต่อเมื่อ Tag ได้มีการกำหนดสิทธิ์ชั้นของลิฟต์ เอาไว้ในโปรแกรมของ Arduino เมื่อ ไฟตามชั้นที่สามารถไปได้ติดสว่าง Arduino ตัวที่ 2 ที่รอคำสั่งจากปุ่มกด ซึ่งจะใช้กระแสไฟจาก Arduino ตัวที่1 ในการรับคำสั่งไปยังชั้นต่างๆ เมื่อมีการกดปุ่มแล้ว Arduino ตัวที่ 2 จะตรวจสอบว่าได้กดไปยังชั้นใด เมื่อกดปุ่มแล้วจะมีเสียง Buzzer ส่งเสียงBeep ตามชั้น และจะมีเสียง Beep 2 ครั้งก็ต่อเมื่อ ได้ไปยังชั้นที่กดแล้ว IC Chip Register74HC595 ทำหน้าที่แสดงผลสถานะต่างๆ โดยมี IC 1 74HC595 จะแสดงผลสถานะลิฟต์ ขึ้น/ลง และส่งไปยังภาคขับมอเตอร์ดึงลิฟต์ ขึ้น/ลง ตามไปด้วย ซึ่งจะมี IC L298 เป็นภาคขับมอเตอร์ 12 โวลต์ และในส่วนของ Relay 1 , Relay 2 จะทำหน้าที่ Reset บอร์ด Arduino ตัวที่ 1 ที่ได้จำค่าจาก Tag ไว้ ก็ต่อเมื่อ ได้มีการกดลิฟต์แล้ว IC 2 74HC595 จะแสดงสถานะ เมื่อกดปุ่มตามชั้นต่างๆIC 3 74HC595 จะแสดงสถานะตัวเลขของชั้น ด้วย 7Segment

3.5 การออกแบบและประกอบวงจรและตัวลิฟต์

ภาพที่ 3.5 แสดงการจำลองลิฟต์ที่ใช้งาน

จากภาพที่ 3.5 แสดงภาพการจำลองลิฟต์ที่ประกอบขึ้นเพื่อมาใช้ในการทดลองการใช้งาน

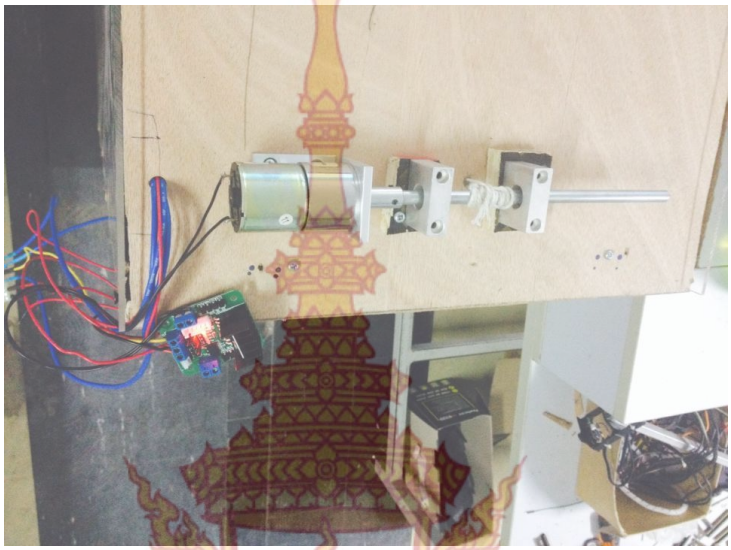


ภาพที่ 3.6 แสดงฐานตัวลิฟต์

จากภาพที่ 3.6 เป็นฐานตัวลิฟต์เพื่อไว้ยึดตัวลิฟต์และให้สามารถขึ้น-ลงได้

ภาพที่ 3.7 แสดงตัวโครงลิฟต์

จากภาพที่ 3.7 เป็นการนำตัวลิฟต์ประกอบกับตัวฐานลิฟต์และประกอบกับตัวโครงลิฟต์

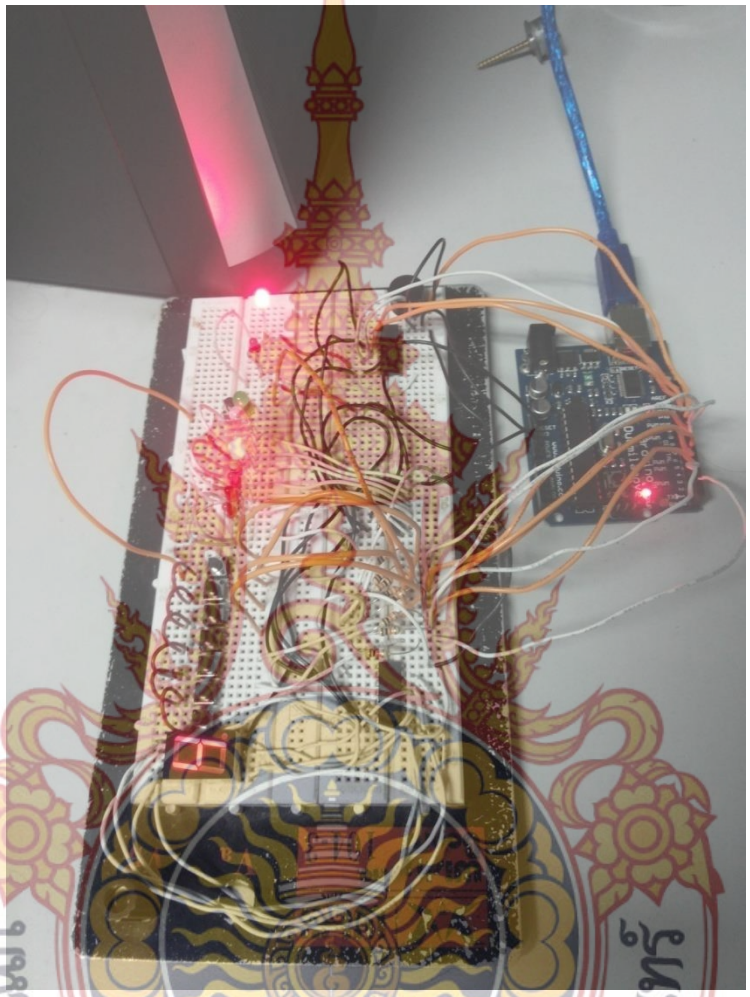


ภาพที่ 3.8 แสดงการติดตั้งภาคขับมอเตอร์

จากภาพที่ 3.8 เป็นการติดตั้งภาคขับมอเตอร์เพื่อควบคุมการขึ้นลงของชั้นลิฟต์

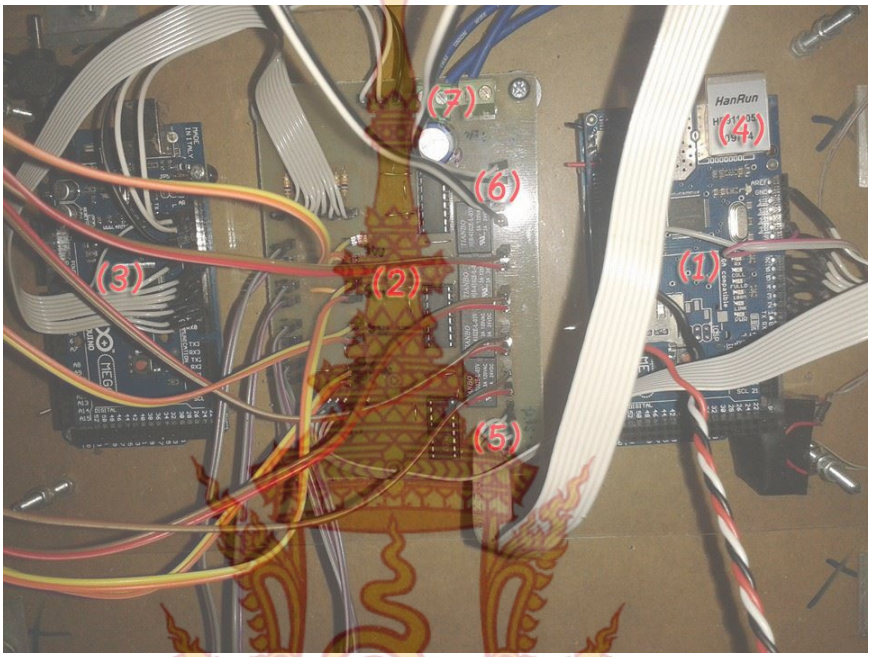
ภาพที่ 3.9 แสดงการต่อบอร์ดแสดง LED กับ วงจรควบคุมแสดงผลไฟ

จากภาพที่ 3.9 แสดงปุ่มกดซึ่งเป็นชนิดเดียวกันกับใช้งานในลิฟต์จริง



ภาพที่ 3.10 แสดงการทดลองคอนโทรลชั้นลิฟต์

จากภาพที่ 3.10 แสดงการเขียนโปรแกรมและทดลองการขึ้น-ลง ของลิฟต์



ภาพที่ 3.11 แสดงการประกอบวงจรต่าง ๆ เข้าด้วยกัน

จากภาพที่ 3.11 เป็นการนำอุปกรณ์ต่างๆ ประกอบเข้าด้วยกัน โดยอธิบายได้ดังนี้

หมายเลข (1) คือบอร์ด Arduino Mega 6410 ประกอบกับ Arduino Ethernet Shield

หมายเลข (2) คือจงจรของลิฟต์จำลอง โดยที่ หมายเลข (1) จะทำหน้าที่ควบคุมการทำงานของ สวิตซ์ ในบอร์ดนี้

หมายเลข (3) คือบอร์ด Arduino Mega 6410 ตัวที่ 2 ทำหน้าที่เป็นบอร์ดควบคุมการทำงานโดยรวมของลิฟต์จำลอง

หมายเลข (4) คือส่วนที่จะต่อสาย RJ45

หมายเลข (5) คือส่วนที่ต่อกับ 7Segment เพื่อแสดงชั้นลิฟต์

หมายเลข (6) คือส่วนที่ต่อกับ LED สถานะ ขึ้น-ลง ของลิฟต์

หมายเลข (7) คือส่วนไฟเลี้ยงบอร์ดและสวิตซ์ โดยใช้ 5 โวลต์ และ 24 โวลต์

ภาพที่ 3.12 แสดงลิฟต์ที่เสร็จสมบูรณ์

จากภาพที่ 3.12 เป็นตัวลิฟต์ที่ประกอบเสร็จสมบูรณ์และพร้อมใช้ทดลองงาน

บทที่ 4
การทดลองและผลการทดลอง

ในบทนี้จะเป็นขั้นตอนและการทดสอบระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID ที่สร้างขึ้นมาและเขียนโปรแกรมขึ้นมานั้น สามารถทำงานได้ตามที่คาดหวังไว้หรือไม่เพียงใดซึ่งเราจะเน้นที่การกำหนดสิทธิ์การเลือกชั้นลิฟต์ที่ถูกต้องรวมถึงการทำงานของการขึ้น-ลงของลิฟต์ ขั้นตอนการทดลองและสรุปผลดังนี้

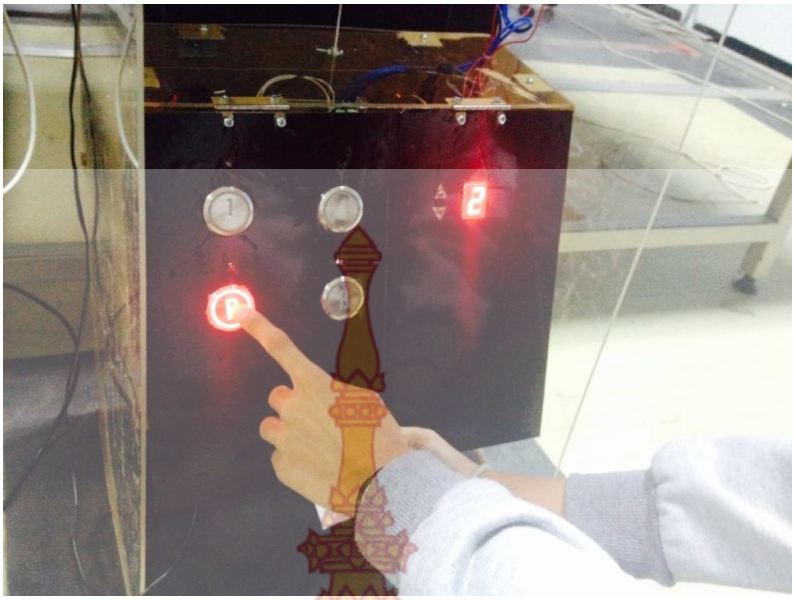
4.1 การทดลองการเลือกชั้นลิฟต์โดยการกำหนดสิทธิ์ในบัตร RFID

4.2 สรุปผลการทดลองในการทดลองการขึ้นชั้นลิฟต์

4.1 ขั้นตอนการทดลองการใช้งาน

ภาพที่ 4.1 แสดงการแสกนบัตร RIFD ที่มีการกำหนดสิทธิ์

จากภาพที่ 4.1 หลอด LED แสดงการสแกนบัตร RFID ที่มีการกำหนดสิทธิ์ชั้นที่ 1,2,3

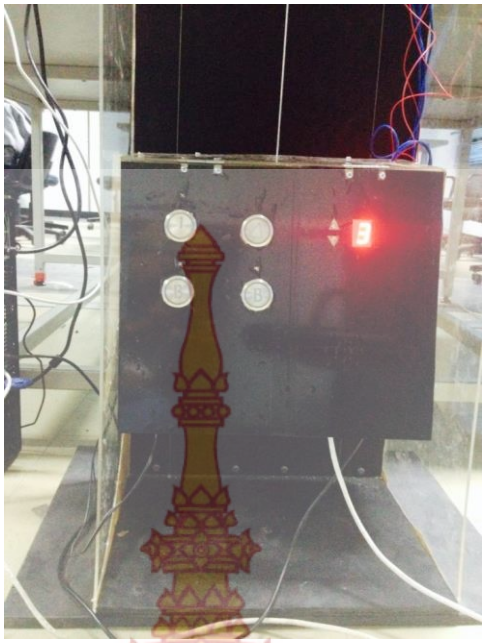


ภาพที่ 4.2 แสดงการเลือกชั้นลิฟต์

จากภาพที่ 4.2 แสดงการกดเลือกชั้นลิฟต์ชั้นที่ 3

ภาพที่ 4.3 แสดงการทำงานของลิฟต์

จากภาพที่ 4.3 หลอด LED แสดงสถานการณ์ทำงานของการขึ้นชั้นลิฟต์ไปชั้นที่ 3

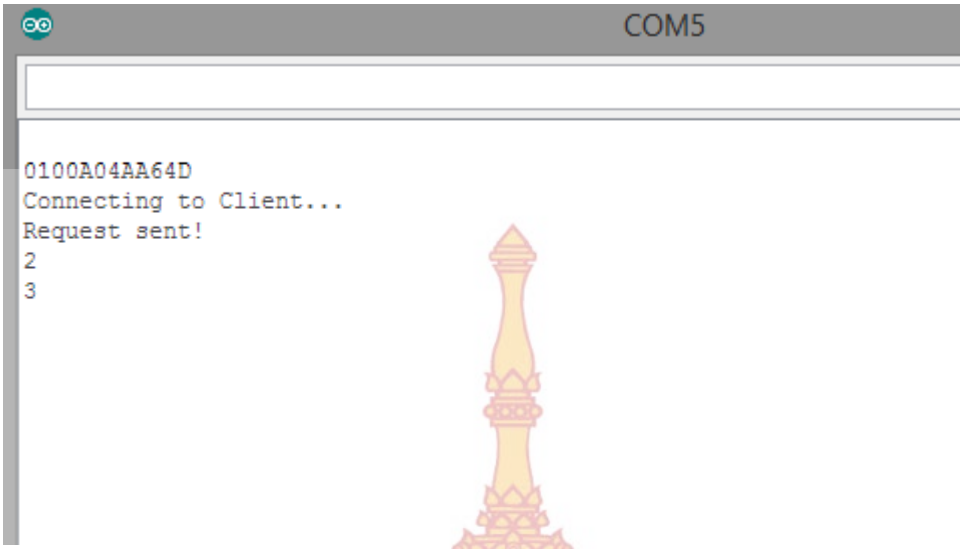


ภาพที่ 4.4 แสดงการขึ้นชั้นลิฟต์

จากภาพที่ 4.4 แสดงสถานการณ์ขึ้นชั้นลิฟต์จากชั้นที่ 1 ไปยังชั้นที่ 3

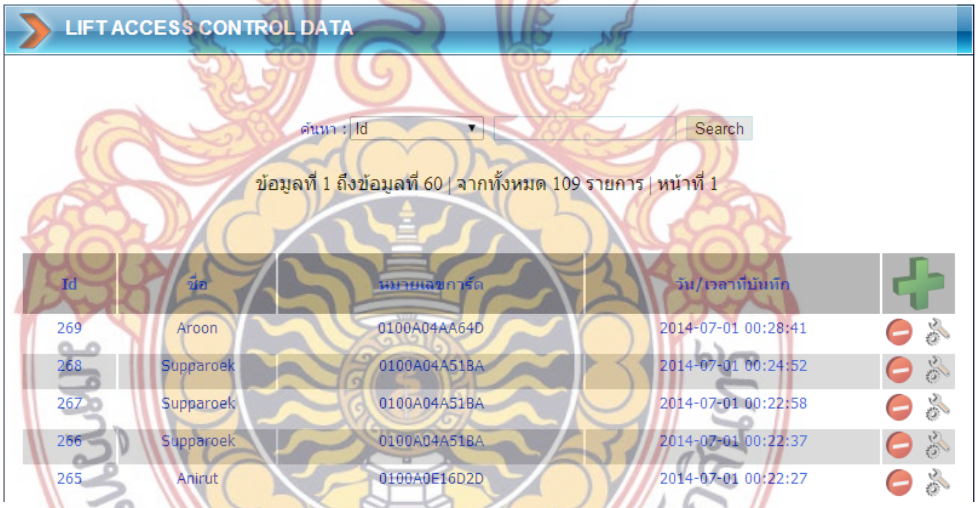
ภาพที่ 4.5 แสดงการสแกนบัตรRIFD ที่ไม่มีการกำหนดสิทธิ์

จากภาพที่ 4.5 เมื่อนำบัตรRFID ที่ไม่มีการกำหนดสิทธิ์มาแสกน หลอดไฟ LED จะไม่แสดงสถานะการขึ้นชั้นลิฟต์



ภาพที่ 4.6 แสดงการติดต่อฐานข้อมูล

จากภาพที่ 4.6 เมื่อนำบัตรRFID ที่มีการกำหนดสิทธิ์ไว้ไมโครคอลโทรลเลอร์จะทำการส่งข้อมูลการใช้งานไปบันทึกยังฐานข้อมูล



ภาพที่ 4.7 แสดงข้อมูลการใช้งานที่บันทึกไว้

จากภาพที่ 4.6 เมื่อมีการบันทึกข้อมูลลงฐานข้อมูลจะสามารถดูข้อมูลที่บันทึกได้โดยผ่านเว็บเบราส์เซอร์

ตารางที่ 4.1 แสดงการแสกนบัตร RIFD ที่กำหนดสิทธิ์และไม่กำหนดสิทธิ์

	RFID Reader อ่านค่าถูกต้อง (100)	LED แสดงชั้น ถูกต้อง (100)	สวิตช์ทำงาน (100)	บันทึกข้อมูลลง ฐานข้อมูล (100)
บัตรกำหนดสิทธิ์ชั้นที่ 1,3	100	100	98	100
บัตรกำหนดสิทธิ์ชั้นที่ 1,2,4	100	100	99	100
บัตรกำหนดสิทธิ์ชั้นที่ 1,2,3,4	100	100	100	100
บัตรที่ไม่ได้กำหนดสิทธิ์	100	100	100	100

จากตารางที่ 4.1 แสดงผลการทดลองแสกนบัตร RFID ทีมีการกำหนดสิทธิ์และไม่กำหนดสิทธิ์ จำนวนอย่างละ 100 ครั้ง ผลสรุป RFID Reader อ่านค่าถูกต้อง คิดเป็น 100% LED แสดงชั้นถูกต้อง คิดเป็น 100% สวิตช์ทำงานถูกต้อง 99.25% และ บันทึกข้อมูลลงฐานข้อมูลถูกต้อง คิดเป็น 100%

4.2 สรุปผลการทดลอง

ผลการทดลองจะเห็นได้ว่าผลที่ออกมาเป็นที่น่าพอใจ การทำงานของระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID สามารถควบคุมการทำงานได้ตามที่ต้องการและมีโอกาสที่จะผิดพลาดน้อยมาก บัตรRFID ที่กำหนดสิทธิ์ไว้แล้วสแกนบน RFID Reader หลอดไฟ LED ก็ขึ้น สถานะการเลือกชั้น ส่วน RFID ที่ไม่ได้กำหนดสิทธิ์ไว้แล้วสแกนบน RFID Reader หลอดไฟ LED จะไม่ขึ้นสถานะการเลือกชั้น

บทที่ 5
สรุปผล อภิปรายผลและข้อเสนอแนะ

ในการทำปริญญานิพนธ์ระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID เราสามารถที่จะสรุปผลของการทำงาน และปัญหาต่างๆ ที่เกิดขึ้นระหว่างการทำโครงงานปริญญานิพนธ์รวมถึงการแก้ไขและข้อเสนอแนะ เพื่อที่จะได้นำโครงงานปริญญานิพนธ์นี้ไปพัฒนาต่อในอนาคต

5.1 สรุปผล

ปริญญานิพนธ์นี้เป็นการระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID สามารถนำไปใช้งานในอาคารหรือในชั้นที่ต้องการรักษาความปลอดภัยในระดับต่างๆได้ เนื่องจากระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID สามารถกำหนดสิทธิ์ให้กับบุคคลดังกล่าวว่ามีสิทธิ์ขึ้นชั้นลิฟต์ในระดับชั้นไหนบ้าง

ผลของการทำโครงงานระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID ได้ทำการค้นคว้าและออกแบบที่น่าพอใจและบรรลุวัตถุประสงค์ คือ สามารถกำหนดสิทธิ์การขึ้นชั้นลิฟต์ได้ และทำให้ผู้จัดทำปริญญานิพนธ์มีความรู้ การเขียนโปรแกรม Arduino, ภาษา C และ C++ ได้เป็นอย่างดี

5.2 ปัญหาที่พบ

5.2.1 เนื่องจาก ปุ่มกดเลือกชั้นลิฟต์และสาย ไม่มีขายตามท้องตลาด

5.2.2 เนื่องจากวงจรต้องการใช้สวิตซ์ลิฟต์จำเป็นต้องใช้ไฟในหลายๆ แรงดันทำให้ต้องใช้งานแหล่งจ่ายที่มีหลายๆ แรงดันไฟ

5.2.3 เนื่องจากต้องการความสมจริงเลยออกแบบให้อุปกรณ์ส่วนใหญ่อยู่ในกล่องลิฟต์ในขณะเดียวกันสายไฟใช้งานต้องอยู่ด้านนอกเลยพบปัญหาในการจัดสาย

5.3 แนวทางการแก้ไข

5.3.1 ทำการหาสสวิตซ์ลิฟต์ที่ใช้งานแล้วมาดัดแปลงสายใช้งานเพื่อนใช้ทดสอบ

5.3.2 ใช้พาวเวอร์ซัพพลายสวิทซ์เพื่อให้ได้ซึ่งแรงดันหลายๆระดับ

5.4 ข้อเสนอแนะ

ข้อเสนอแนะของการทำปริญญานิพนธ์ระบบควบคุมการเลือกชั้นลิฟต์ด้วย RFID มีข้อเสนอแนะและนำไปพัฒนาได้ดัง

5.4.1 สามารถนำไปพัฒนาระบบให้เสถียรยิ่งขึ้น

5.4.2 สามารถนำไปพัฒนาโดยใช้ไมโครคอลโทรลเลอร์ตัวอื่นๆเพื่อการแสดงผลที่ดีกว่า

5.4.3 สามารถนำไปประยุกต์ใช้กับระบบอื่นๆ เช่น เปิดปิดรั้ว – ประตู หรือ ระบบเช็คและบันทึกการเข้าเรียนของนักศึกษา

บรรณานุกรม

ทรงศักดิ์ รวิรังสรรค์**. ข้อกำหนดและกฎหมายในการออกแบบอาคาร**. ซีเอ็ดยูเคชั่น จำกัด (มหาชน), 2544.

สถาปัตยกรรมของอาคารสูง. 2548. [ออนไลน์]. เข้าถึงได้จาก http://ekchanakijelevator.com/ articlesm**.**. (สืบค้นเมื่อ 10 กุมภาพันธ์ 2557)**.**

การขนส่งแนวตั้งลิฟต์และบันไดเลื่อน. 2548. [ออนไลน์]. เข้าถึงได้จาก http://ekchanakijelevator.com/article. Lesmenub5htm. (สืบค้นเมื่อ 20 มีนาคม 2556).

โครงสร้างโปรแกรมของ Arduino. (ม.ป.ป.). [ออนไลน์]. เข้าถึงได้จาก http://elec.cmtc.ac.th/ 2011/attachments/article/314. (สืบค้นเมื่อ 13 ธันวาคม 2556).

พื้นฐานไมโครคอนโทรลเลอร์ด้วยArduino. (ม.ม.ป.). [ออนไลน์]. เข้าถึงได้จาก http://www.thainetbeans.com /arduino/start/start.php.สืบค้นเมื่อ15 มกราคม 2556.

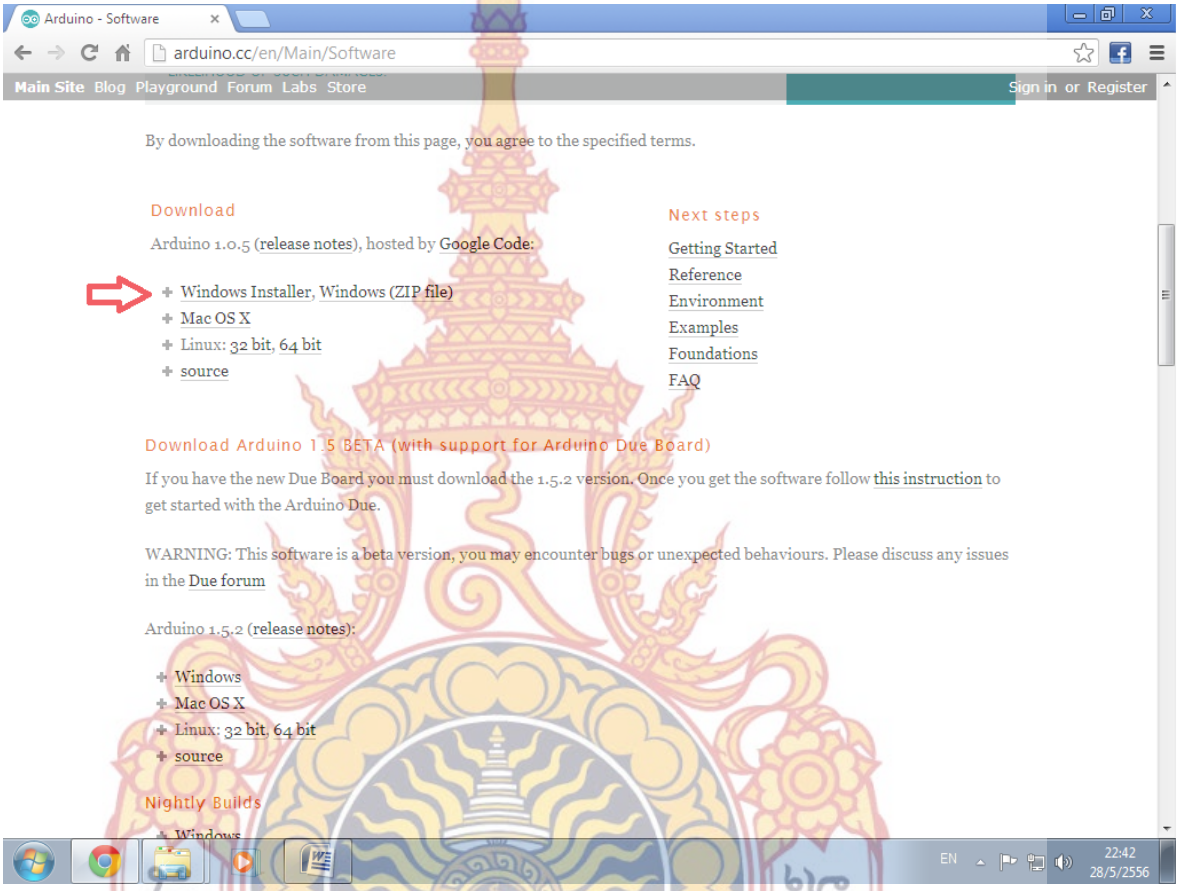
เริ่มต้นกับ Arduino ติดตั้งใช้งานบน Windows 7. (ม.ม.ป.). [ออนไลน์]. เข้าถึงได้จาก http://codesanook.com/article/details. (สืบค้นเมื่อ 17 มกราคม 2556).

ภาคผนวก ก.

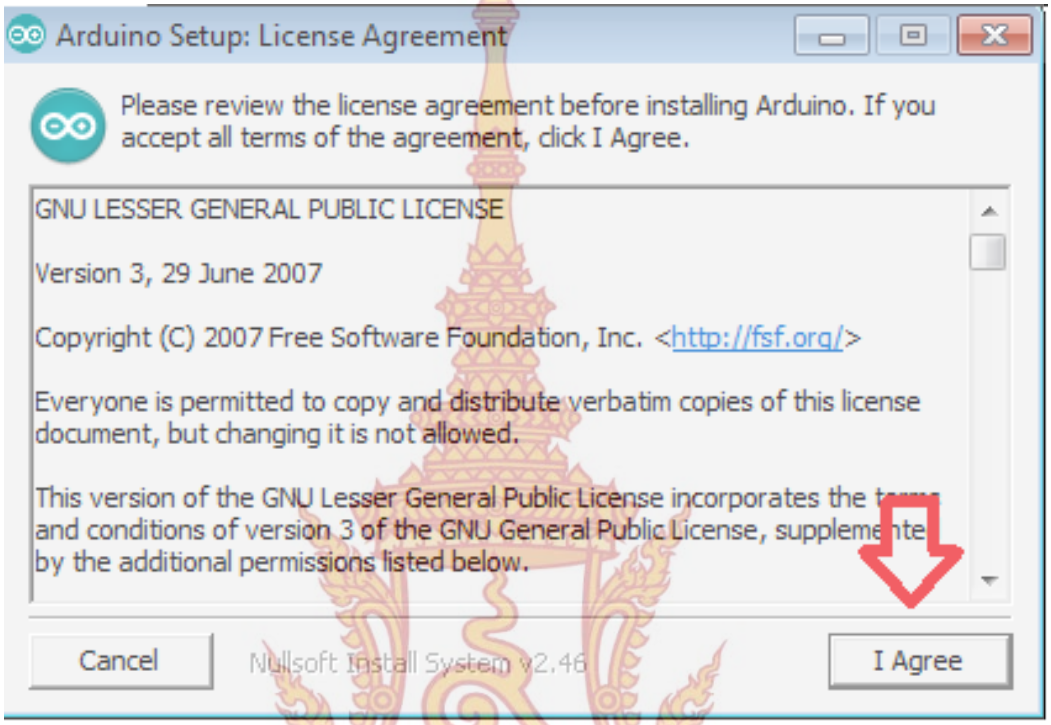
การติดตั้งโปรแกรม Arduino

การติดตั้งโปรแกรม Arduino

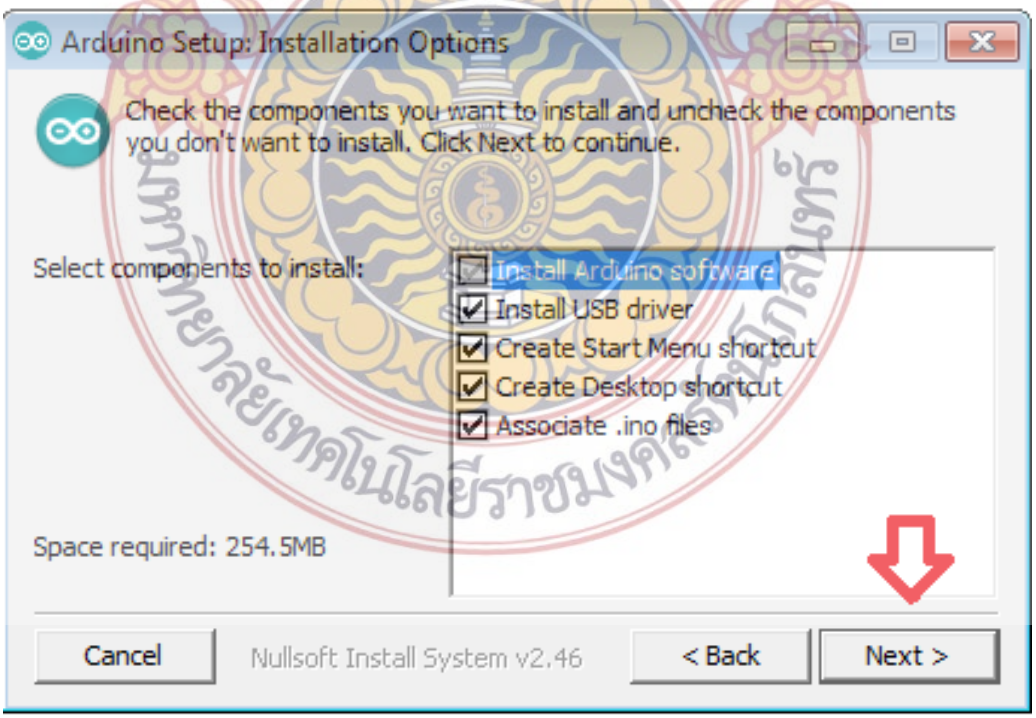
1. ไปยังเว็บไซต์ http://arduino.cc/en/Main/Softwareเลือกดาวน์โหลดที่Windows Installer



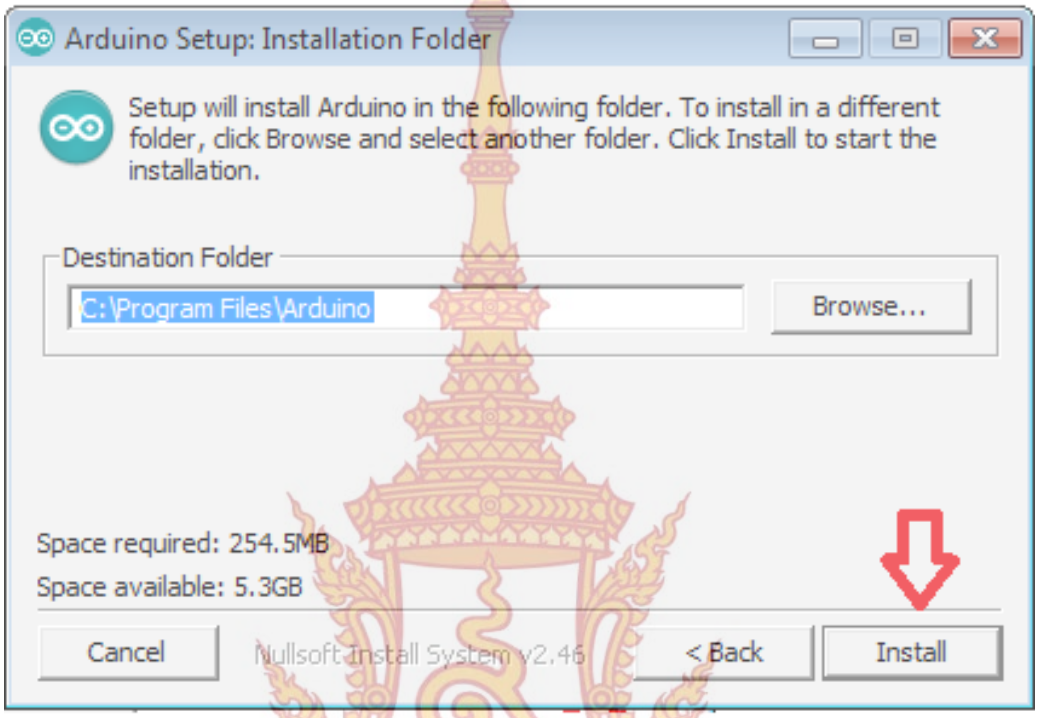
2. เมื่อดาวน์โหลดไฟล์มาแล้วทำการเปิดโปรแกรมและติดตั้งโดยคลิ้ก I Agree



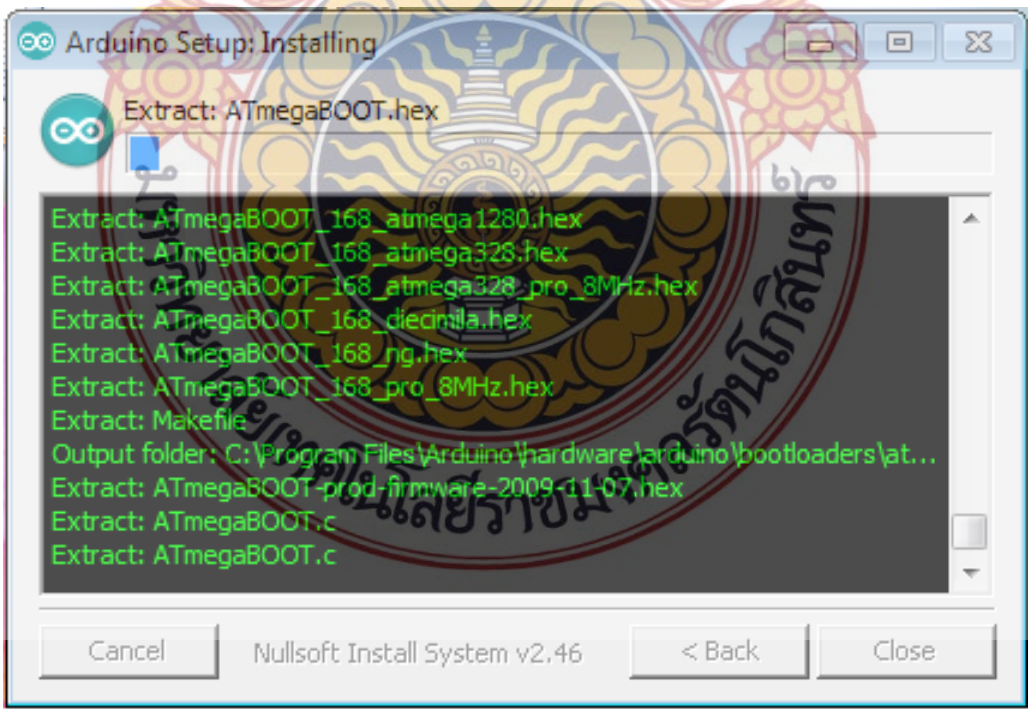
3. คลิ้ก Next >



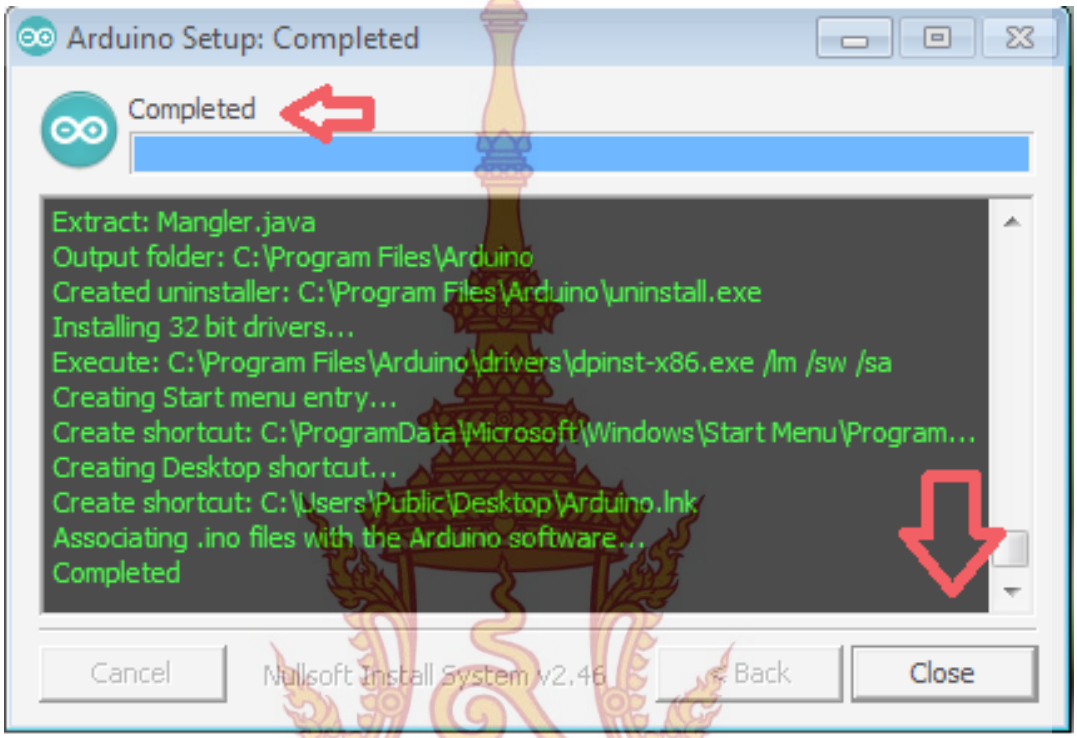
4. คลิ้ก Install



5. รอจนกว่าโปรแกรมทำการติดตั้งเสร็จสิ้น



6. เมื่อติดตั้งเสร็จแล้ว จะแสดง Completed คลิ้ก Close เป็นการเสร็จสิ้นการติดตั้งโปรแกรม

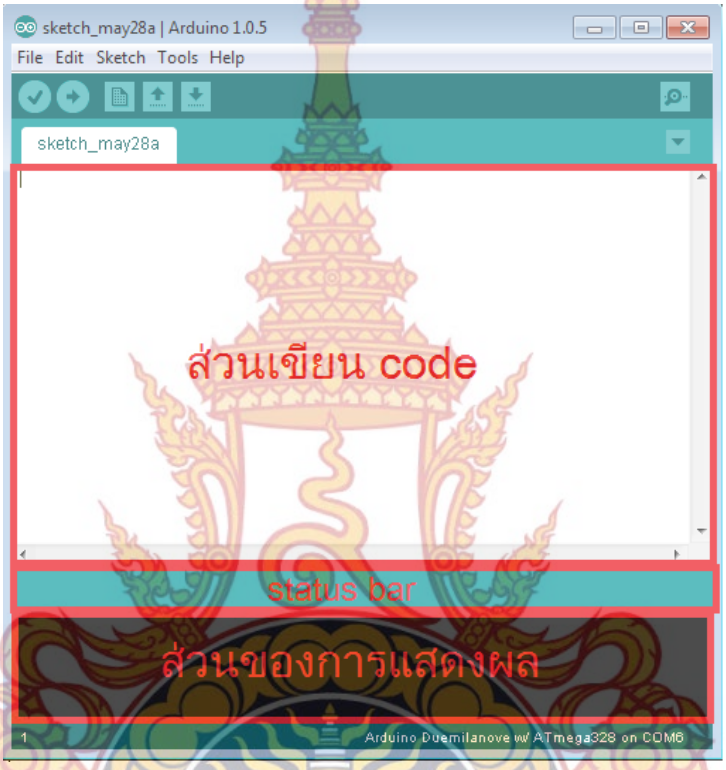


ภาคผนวก ข.

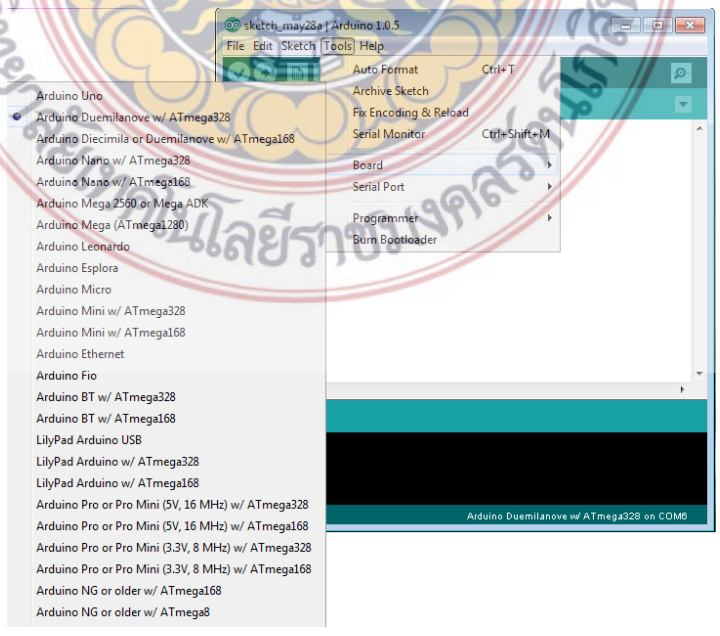
การใช้งาน โปรแกรม Arduino

การใช้งาน โปรแกรม Arduino

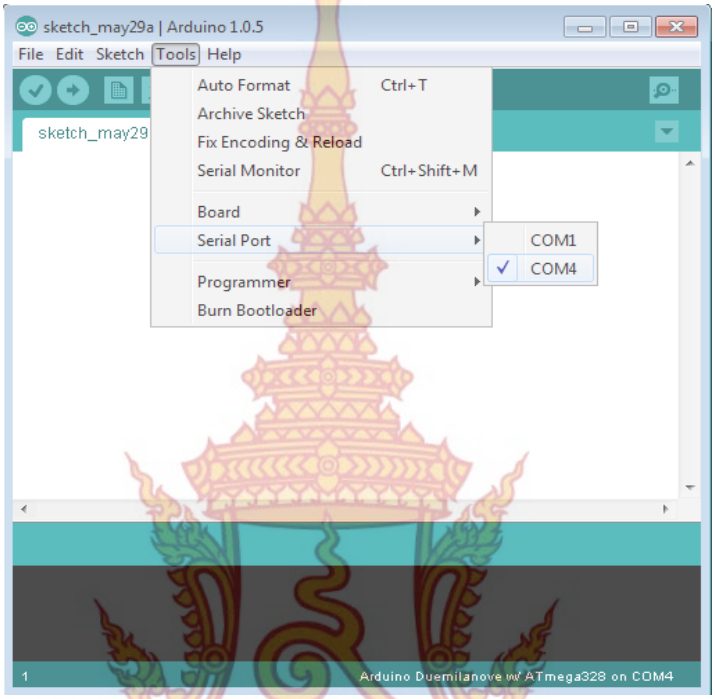
1. ลักษณะของโปรแกรม และส่วนต่างๆ



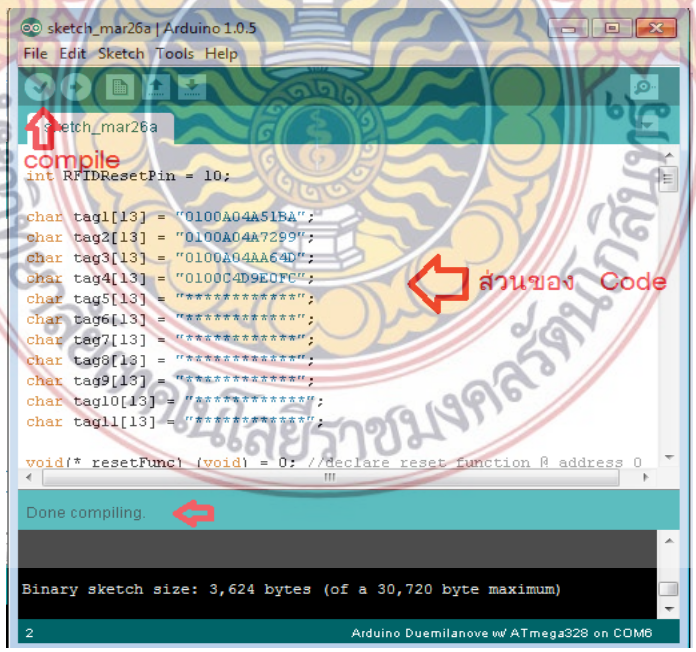
2. การเริ่มต้นใช้งาน จะต้องเลือก บอร์ด ให้ตรงกับรุ่นที่ใช้ ในที่นี้ ใช้บอร์ดรุ่น Arduino Duemilanove ATmega328



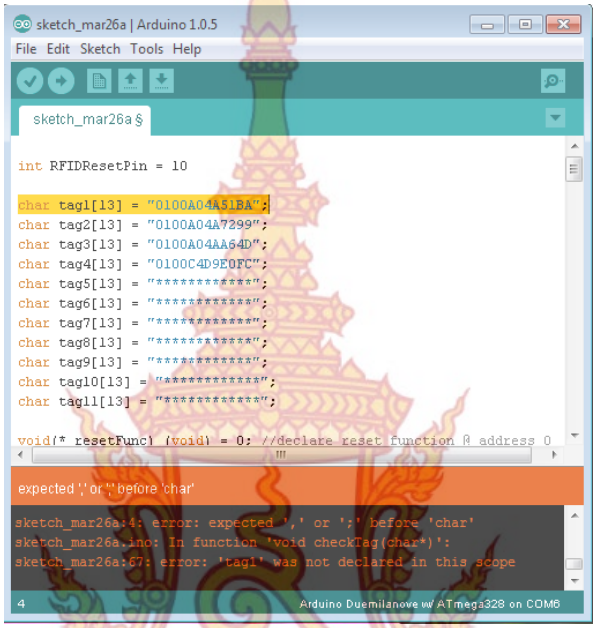
3. จากนั้นให้เลือก Serial Port ให้ตรงกับ Port ที่ได้เชื่อมต่อกับบอร์ด



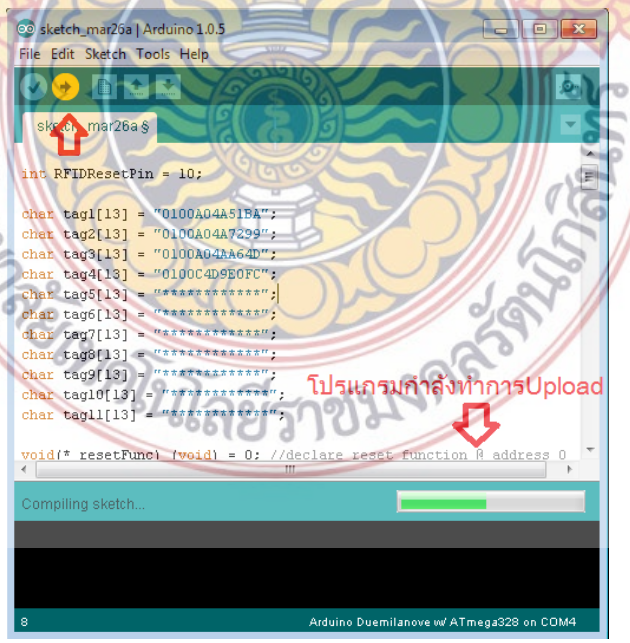
4. เมื่อทำการเขียน Code โปรแกรมเสร็จสิ้น จะต้องทำการคลิ้กปุ่ม compileจากนั้นโปรแกรมจะแสดง Done compiling แสดงว่าโปรแกรมที่เขียนไม่มีการ Error เกิดขึ้น



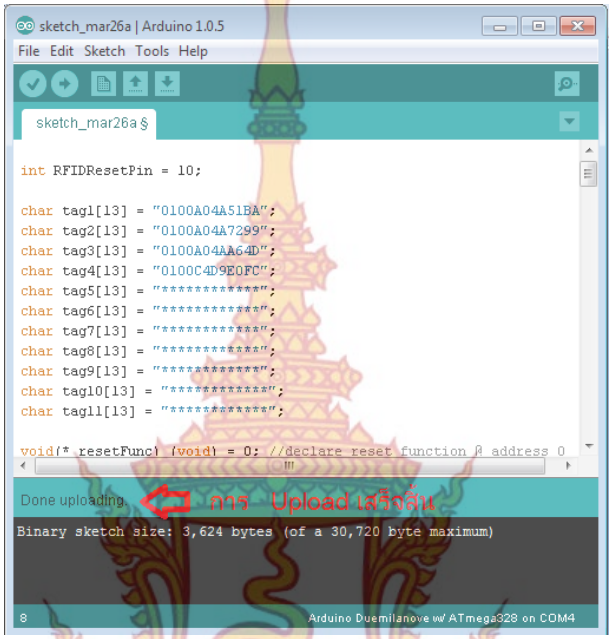
5. แสดงการ Error ของ Code ที่เขียน โปรแกรมจะแสดงออกมาว่าส่วนไหนคือส่วนที่ต้องทำการแก้ไข



6. เมื่อทุกอย่างพร้อมแล้ว ก็ให้ทำการ upload โปรแกรมที่เขียนลงบนบอร์ด Arduino โดยคลิ้กที่ปุ่ม ดังรูป



7. แสดงเมื่อ Upload สำเร็จ



8. เมื่อ Upload เสร็จสิ้น สามารถดูการทำงานของโปรแกรมที่ได้ออกแบบไว้ได้โดย คลิ้กที่ Serial Monitor

ภาคผนวก ค.

Control Code

Control Code

```
//CONSTANTS
// Arduino data pins ---------------------------------------------------
// toggle is hooked up to ANALOG IN 0
#define MODETOGGLE  0
// the rest is hooked up to digital in.
#define BUTTON1     2
#define BUTTON2     3
#define BUTTON3     4
#define BUTTON4     5
#define BUTTON5     6
#define BUTTON6     7
#define BUTTON7     8
#define BUTTON8     9
#define BUZZER      10
//Pin connected to ST_CP of all 74HC595 shift registers
#define LATCHPIN    11
//Pin connected to SH_CP of all 74HC595 shift registers
#define CLOCKPIN    12
////Pin connected to DS of first 74HC595 shift register
#define DATAPIN     13
// Byte shift positions of various LEDs ------------------------------------
#define UPARROW_SHIFT 0
#define DOWNARROW_SHIFT 1
#define REDFLASHLED_SHIFT 2
#define BLUEFLASHLED_SHIFT 3
// other constants ---------------------------------------------------------
```

```
// debug related
#define ERROR 100
#define WARN 200
#define INFO 300
#define DEBUG 400
#define TRACE 500
#define ELEVATOR_MODE 0
#define WACKAMOLE_MODE 1
// analog level sent to piezo at floor arrival beep
#define FLOORBUZZLEVEL  150
// # of millis we buzz for a floor
#define FLOORBUZZMILLIS  300
// how long between the parts of the door swish animation
#define DOOR_SWISH_MILLIS 95
// how long to wait before running swish
#define PRE_SWISH_DELAY_MILLIS 250
// how long to wait after running swish
#define POST_SWISH_DELAY_MILLIS 300
// how long it takes to move from one floor to another
#define FLOOR_TRAVELTIME_MILLIS 1700
#define NUM_FLOOR_BUTTONS  8
#define HALF_NUM_FLOOR_BUTTONS  4
// the number of bytes needed to control all our 595 shift regs.
#define NUM_SHIFT_BYTES   3
// the number of phases for the 'door swipe' effect on the seven segment display
#define SWIPE_PHASE_COUNT  4
// stuff used by the startup routine
#define STARTUP_DELAY_MILLIS 750
```

```
#define STARTUP_BUZZ_MILLIS 250
#define STARTUP_BUZZ_ITERS 2
#define STARTUP_ITER_BUZZLEVEL 100
// siren stuff
#define SIREN_START_LEVEL 70
#define SIREN_STOP_LEVEL 200
#define SIREN_STEP_AMOUNT 10
#define SIREN_STEP_MILLIS 50
#define SIREN_SLEEP_BETW 200

// ---------------------------------------------------------------------
// binary representation of the bit patterns needed to display 0 .. 9 on the 7 segment display
// when it is attached to a 595 shift register IC.
// ASSUMES A attached to pin 1, B attached to pin 2, and so on (pin 0 attached to DP)
//   so the bits that matter are A-G (for any standard 7 segment display): GFEDCBA-
// 7 seg display led pattern:
//   -A-
//  F   B
//   -G-
//  E   C
//   -D-    DP
//                          GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-  GFEDCBA-
```

```
byte BINARY_NUMBER_PATTERNS[10]= {B01111110, B00001100, B10110110,
B10011110, B11001100, B11011010, B11111010, B00001110, B11111110,
B11011110};
// 7 segment effects, round and round ------------------------------
byte SEVEN_SEGMENT_ROUND_N_ROUND[] =        { B00000010, B00000100,
B00001000, B00010000, B00100000, B01000000 };
// 7 segment effects, side to side ---------------------------------
byte SEVEN_SEGMENT_LEFT_TO_RIGHT_MASK[SWIPE_PHASE_COUNT] = {
B10011110, B00001100, B00000000, B00000001  };
byte SEVEN_SEGMENT_RIGHT_TO_LEFT_MASK[SWIPE_PHASE_COUNT] = {
B00000001, B00001100, B10011110, B11111110  };
// -----------------------------------------------------------------------
// Control Vars
int LOG_LEVEL = INFO;
// direction of elevator (-1=down, 0=dormant, +1=up)
int _currentDirection = 0;
// floor the elevator is on (1 - 8)
int _currentFloor = 1;
// when we should get to the next floor
unsigned long _nextFloorArrivalTime = 0;
// if we're buzzing, this is when we stop
unsigned long _currentBuzzStopTime = 0;
// the previous values of buttons 1-8 [indexes 0-7, respectively]; Values are 0 or 1
int _prevFloorButtonValues[NUM_FLOOR_BUTTONS] = {0,0,0,0,0,0,0,0};
int _floorButtonInputPins[NUM_FLOOR_BUTTONS] = {BUTTON1, BUTTON2,
BUTTON3, BUTTON4, BUTTON5, BUTTON6, BUTTON7, BUTTON8};
// current values of other LED's
int _currentUpArrowLedVal = LOW;
```

```
int _currentDownArrowLedVal = LOW;
int _currentRedFlashLedVal = LOW;
int _currentBlueFlashLedVal = LOW;
// this is what we want the current value of the 7 segment display to be.
//  see the comments near the constant BINARY_NUMBER_PATTERNS for the
value
//  of each bit.
byte _currentSevenSegByte = B00000000;
// mode of 0 is elevator mode, nonzero is whackamole (tbd)
byte _mode = ELEVATOR_MODE;
boolean _modeChangeFlag = false;
// flag that tells us that the last loop finished a floor buzz.
boolean _doneFloorBuzzEventFlag = false;

// ----------------------------------------------------------------------
//  Startup
// ----------------------------------------------------------------------

void setup() {
 log(INFO, "setup()", "Starting up...");
 log(INFO, "setup()", true);
 log(INFO, "setup()", 99);
 log(INFO, "       high: ", HIGH);
 log(INFO, "       low: ", LOW);

 //  pinMode(MODETOGGLE, INPUT);
 for(int i=0; i<NUM_FLOOR_BUTTONS; i++) {
 pinMode(_floorButtonInputPins[i], INPUT);
```

```
 }
 randomSeed(analogRead(0));

 pinMode(BUZZER, OUTPUT);
 pinMode(LATCHPIN, OUTPUT);
 pinMode(CLOCKPIN, OUTPUT);
 pinMode(DATAPIN, OUTPUT);
 Serial.begin(9600);

 initState();
 // todo: run a diagnostic routine to excercise all components
}

void initState() {
 _currentDirection = 0;
 _currentFloor = 1;
 _nextFloorArrivalTime = 0;
 _currentBuzzStopTime = 0;
 _currentUpArrowLedVal  = LOW;
 _currentDownArrowLedVal = LOW;
 _currentRedFlashLedVal = LOW;
 _currentBlueFlashLedVal = LOW;
 _currentSevenSegByte = B00000000;
 _modeChangeFlag = false;
 _doneFloorBuzzEventFlag = false;
 setAllFloorButtonLEDs(0);
 setCurrentSevenSegmentToDecimalNumber(_currentFloor);
}
```

```
// ---------------------------------------------------------------------
//  Main Loop
// ---------------------------------------------------------------------
void loop() {
 // runStartupRoutine();
 // delay(3000);
 loopAction();

 // test the 7segment
 //testSevenSegment();
}

/* run the main loop depending on what mode the toggle button is in.
 * There is only one mode implemented (elevator simulation) at this time.
 */
void loopAction() {
 log(TRACE, "TOP of loopAction, mode: ", _mode);
 // was a button pushed? if so, update state
 checkForAnyButtonPresses();
 // now, everything else is mode dependent, so breakout.
 if(_mode==ELEVATOR_MODE) elevatorModeLoopAction();
 else // it's wackamole
 wackamoleLoopAction();
 // update all display LEDs controlled thru shift regs.
 log(TRACE, "*updateShiftRegs ", "");
 updateShiftRegisters();
}
```

```
/* Placeholder for TBD mode -
 *  This mode would emulate a wackamole type carnival game with the floor buttons.
 */
void wackamoleLoopAction() {
 // todo: implement wackamole loop
 elevatorModeLoopAction();
}

/* Control loop for the elevator mode.  Each iteration checks for conditions and reacts to them.
 * It is really a problem that could use an event driven model solution.  Since we don't have that
 * we just loop over and look for conditions in a big if .. then .. else ..
 */
void elevatorModeLoopAction() {
 log(TRACE, "elevatorModeLoopAction  ----- fl. ", _currentFloor);
 // have we arrived at a floor yet (assuming we're moving, that is)?
 if(isFloorReached()) { // floor arrival ----------------------------------------
 log(TRACE, "*floor reached ", "");
 int newFloor = _currentFloor + _currentDirection;
 // update _currentFloor, re-init _nextFloorArrivalTime, starts buzz
 processFloorArrivalEvent(newFloor);
 } else if(isFloorBuzzing()) { // floor buzz in progress ------------------------------
 log(TRACE, "*floor buzzing ", "");
 // clear _currentBuzzStopTime and stop buzzing
 _doneFloorBuzzEventFlag = processInFloorBuzzEvent();
```

```
} else if (_doneFloorBuzzEventFlag) {   // floor buzz over ----------------------------
log(TRACE, "*doneFloorBuzzEvent triggered ", "");
_doneFloorBuzzEventFlag = false;
// are we stopping on this floor???
if(isStoppingOnFloor(_currentFloor)) {
log(TRACE, "*stoppingOnFloor: ", _currentFloor);
// ok, turn off button light, open/close door
processStopAtFloorEvent(_currentFloor);
}
// should this be broken out of the parent conditional?  should it happen this cycle, or wait for the next?
//    figure out directional logic:(figure out if we're still moving, in what direction, etc.)
if(isAFloorRequestedInDirection(_currentDirection)) {
log(TRACE, "*floorRequestedInDirection, curFl: ", _currentFloor);
// start countdown to next floor,
processElevatorMovementStartingEvent();
} else if(isAFloorRequestedInDirection(getOppositeDirection(_currentDirection))) {
log(TRACE, "*floorRequestedInOPPOSITEDirection, curFl: ", _currentFloor);
// this resets _currentDirection and the up/down LEDs
processDirectionMovementChangeEvent(getOppositeDirection(_currentDirection));
// start countdown to next floor,
processElevatorMovementStartingEvent();
} else {
log(TRACE, "*NOT MOVING ANYMORE, curFl: ", _currentFloor);
// we're officially not moving
// this resets _currentDirection and the up/down LEDs
processDirectionMovementChangeEvent(0);
```

```
 }
 } else if(_currentDirection==0) {   // elevator is IDLE ------------------------------
 log(TRACE, "*ELEVATOR idle, curFl: ", _currentFloor);
 if(isAFloorRequestedInDirection(1)) {
 log(TRACE, "*Floor REQUESTED in DIRECTION: ", "up");
 processDirectionMovementChangeEvent(1);
 processElevatorMovementStartingEvent();
 } else if(isAFloorRequestedInDirection(-1)) {
 log(TRACE, "*Floor REQUESTED in DIRECTION: ", "down");
 processDirectionMovementChangeEvent(-1);
 processElevatorMovementStartingEvent();
 }
 }
 log(TRACE, "*THE END elevatorModeLoopAction() --------", "");
}

// ----------------------------------------------------------------------
//  Startup Routine
// ----------------------------------------------------------------------
// run a sequence of display actions, used to verify hardware's working and
//  offer a bit of LED eye candy.
void runStartupRoutine() {
 // display 0 ------------------------------------------------------
 setCurrentSevenSegmentToDecimalNumber(0);
 updateShiftRegisters();
 delay(STARTUP_DELAY_MILLIS);
 // foreach floor, 1..8, display number, beep, light floor button -----------
 setUpArrowLEDCurrentState(HIGH);
```

```
for(_currentFloor=1; _currentFloor<=NUM_FLOOR_BUTTONS; _currentFloor++) {
setAllFloorButtonLEDs(0);
setFloorButtonLED(_currentFloor, 1); // turn on the current floor
setCurrentSevenSegmentToDecimalNumber(_currentFloor);
if(_currentFloor==NUM_FLOOR_BUTTONS) setUpArrowLEDCurrentState(LOW);
updateShiftRegisters();
runBeep(FLOORBUZZLEVEL, STARTUP_BUZZ_MILLIS, STARTUP_DELAY_MILLIS);
}
// beep, beep ----------
runStartupBeepBeep();
// run swipe ----------
processDoorOpenCloseEvent();
// beep, beep ----------
runStartupBeepBeep();
// now back go down ----------
setDownArrowLEDCurrentState(HIGH);
for(_currentFloor=NUM_FLOOR_BUTTONS; _currentFloor>0; _currentFloor--) {
setAllFloorButtonLEDs(0);
setFloorButtonLED(_currentFloor, 1); // turn on the current floor
setCurrentSevenSegmentToDecimalNumber(_currentFloor);
if(_currentFloor==1) setDownArrowLEDCurrentState(LOW);
updateShiftRegisters();
runBeep(FLOORBUZZLEVEL, STARTUP_BUZZ_MILLIS, STARTUP_DELAY_MILLIS);
}
// beep, beep ----------
runStartupBeepBeep();
// flash alarm and beep ----------
setRedFlashLEDCurrentState(HIGH);
```

```
 setBlueFlashLEDCurrentState(HIGH);
 updateShiftRegisters();
 runSirenSound(5);
 // lights out. sleep.
 initState();
 updateShiftRegisters();
 delay(300);
}

void runStartupBeepBeep() {
 for(int i=0; i<STARTUP_BUZZ_ITERS; i++)
 runBeep(STARTUP_ITER_BUZZLEVEL, STARTUP_BUZZ_MILLIS,
STARTUP_DELAY_MILLIS);
}

void runSirenSound(int pNumberSirens) {
 for(int i=0; i<pNumberSirens; i++) {
 int curLevel = SIREN_START_LEVEL;
 while(curLevel<SIREN_STOP_LEVEL) {
 analogWrite(BUZZER, curLevel);
 delay(SIREN_STEP_MILLIS);
 curLevel += SIREN_STEP_AMOUNT;
 }
 analogWrite(BUZZER, 0);
 delay(SIREN_SLEEP_BETW);
 }
}
```

```
// run the buzzer at analog level pBuzzLevel for pBuzzDurationMillis.
//  then turn it off and delay an additional pPostBuzzMillis
void runBeep(int pBuzzLevel, int pBuzzDurationMillis, int pPostBuzzMillis) {
 analogWrite(BUZZER, pBuzzLevel);
 delay(pBuzzDurationMillis);
 analogWrite(BUZZER, 0);
 if(pPostBuzzMillis>0) delay(pPostBuzzMillis);
}

// ----------------------------------------------------------------------
//  Event Processing Functions
// ----------------------------------------------------------------------

// called when it is determined that we are in a floor buzz.
// Figures out if we have to stop buzzing.  If so, turn off up/down LED
// returns true if buzz is DONE.
boolean processInFloorBuzzEvent() {
 unsigned long now = millis();
 boolean buzzDone  = false;
 if(isFloorBuzzing()) {
 if(_currentBuzzStopTime<now) {
 _currentBuzzStopTime=0;
 // we should stop buzzing
 analogWrite(BUZZER, 0);
 buzzDone=true;
 }
 }
 return buzzDone;
```

```
}

// called when it is determined that we have reached a floor
// kick off the beep, reset the floor arrival timer, update the floor number
void processFloorArrivalEvent(int pFloorNumber) {
 unsigned long now      = millis();
 _nextFloorArrivalTime = 0;
 _currentFloor         = pFloorNumber;
 _currentBuzzStopTime  = now + FLOORBUZZMILLIS;
 analogWrite(BUZZER, FLOORBUZZLEVEL);
 setCurrentSevenSegmentToDecimalNumber(_currentFloor);
}

// Called when we are to stop at a requested floor.
// we must turn off button light and 'open/close' the door
void processStopAtFloorEvent(int pFloorNumber) {
 checkForAnyButtonPresses();
 delay(PRE_SWISH_DELAY_MILLIS);
 checkForAnyButtonPresses();
 processDoorOpenCloseEvent();
 checkForAnyButtonPresses();
 setFloorButtonLED(pFloorNumber, LOW);
 delay(POST_SWISH_DELAY_MILLIS);
 checkForAnyButtonPresses();
}

// called when we arrive at a floor that has been selected for stopping.
//  This is called AFTER the audio effect is over.
```

```
//  Run a quickie visual effect - turn on the 7seg's decimal point, then run a
//  right to left swipe 'animation', leaving everything on.  This is followed
//  by a left to right swipe leaving the current floor number.
//  At the same time, simulate a swoosh sound effect on the piezo.
void processDoorOpenCloseEvent() {
 // start off with the floor number and turn on the decimal point
 setCurrentSevenSegmentToDecimalNumber(_currentFloor);
 // setCurrentSevenSegmentDecimalPoint(true);
 // update the 7 seg, buttons, etc.
 updateShiftRegisters();
 delay(DOOR_SWISH_MILLIS);
 checkForAnyButtonPresses();
 // let's still respond to button presses while we're running the animation
 checkForAnyButtonPresses();
 // now step through the LEFT to RIGHT mask
 for(int i;i<SWIPE_PHASE_COUNT; i++) {
 _currentSevenSegByte = _currentSevenSegByte &
SEVEN_SEGMENT_LEFT_TO_RIGHT_MASK[i];
 analogWrite(BUZZER, 40*i);
 updateShiftRegisters();
 delay(DOOR_SWISH_MILLIS);
 checkForAnyButtonPresses();
 }
 // now step through the RIGHT to LEFT mask
 for(int i;i<SWIPE_PHASE_COUNT; i++) {
 // in right to left, we always start with the actual number
 setCurrentSevenSegmentToDecimalNumber(_currentFloor);
 setCurrentSevenSegmentDecimalPoint(true);
```

```
 _currentSevenSegByte = _currentSevenSegByte &
SEVEN_SEGMENT_RIGHT_TO_LEFT_MASK[i];
 analogWrite(BUZZER, ((40*SWIPE_PHASE_COUNT)/i));
 updateShiftRegisters();
 delay(DOOR_SWISH_MILLIS);
 checkForAnyButtonPresses();
 }
 analogWrite(BUZZER, 0);
 // now return back to the number w/out the decimal point
 setCurrentSevenSegmentToDecimalNumber(_currentFloor);
 updateShiftRegisters();
}

// called when the elevator changes direction (up, then down) or stops moving
//  because the top/bottom floor has been reached or no further floors have
//  been requested in the current direction.
// pDirection: >0 when new direction is up, <0 when down, =0 when stop
void processDirectionMovementChangeEvent(int pDirection) {
 int priorDirection = _currentDirection;
 _currentDirection  = pDirection;
 // reinitialize directional LED state vars
 setUpArrowLEDCurrentState(LOW);
 setDownArrowLEDCurrentState(LOW);
 // now set accordingly
 if(_currentDirection>0)      setUpArrowLEDCurrentState(HIGH);
 else if(_currentDirection<0) setDownArrowLEDCurrentState(HIGH);
}
```

```
// start countdown to next floor,
void processElevatorMovementStartingEvent() {
 unsigned long now      = millis();
 _nextFloorArrivalTime = now + FLOOR_TRAVELTIME_MILLIS;
}
// called when we determine that a floor button was just pushed.
void processFloorButtonPushEvent(int pFloorNumber) {
 // update state var
 setFloorButtonLED(pFloorNumber, HIGH);
 // todo: do we need to do anything else?
 log(DEBUG, "EVENT: floorbtnPush fired, floor: ", pFloorNumber);

//-------------------------------------------------------------

}
int nextFloorRoundRound(int pFloorNum, boolean pClockwise) {
 if(pClockwise) {
 switch(pFloorNum) {
 case 1 : return 3;
 case 2 : return 1;
 case 3 : return 5;
 case 4 : return 2;
 case 5 : return 7;
 case 6 : return 4;
 case 7 : return 8;
 case 8 : return 6;
 default : return 1;
 }
```

```
 } else { // counter clockwise
 switch(pFloorNum) {
 case 1 : return 2;
 case 2 : return 4;
 case 3 : return 1;
 case 4 : return 6;
 case 5 : return 3;
 case 6 : return 8;
 case 7 : return 5;
 case 8 : return 7;
 default : return 1;
 }
 }
}
void adjustFreakoutBuzzer(int pNum) {
 int HI = 180;
 int LO = 100;
 int lvl = (pNum % 2) == 0 ? LO : HI;
 analogWrite(BUZZER, lvl);
}

// -----------------------------------------------------------------------
//  Condition Evaluation Functions
// -----------------------------------------------------------------------

boolean isFloorButtonJustPushed(int pFloorNumber) {
 int index     = pFloorNumber-1;
 int inputPin = _floorButtonInputPins[index];
```

```
 int oldValue = _prevFloorButtonValues[index];
 log(TRACE, "isFloorButtonJustPushed? fl#: ", pFloorNumber);
 return (isButtonJustPushed(inputPin, oldValue));
}

boolean isButtonJustPushed(int pInputPin, int pOldValue) {
 log(TRACE, "  pOldValue: ", pOldValue);
 return ( isButtonPushed(pInputPin) && (pOldValue==LOW) );
}

boolean isButtonPushed(int pInputPin) {
 int val = digitalRead(pInputPin);
 boolean retval = (val==HIGH);
 log(TRACE, "  isButtonPushed on input pin: ", pInputPin);
 log(TRACE, "     is this pin high?: ", retval);
 return retval;
}

boolean isFloorBuzzing() {
 return (_currentBuzzStopTime>0);
}
// returns true if we're moving and we _just_ reached a floor
// reaching a floor means that the _nextFloorArrivalTime has been reached.
boolean isFloorReached() {
 unsigned long now = millis();
 return ( (_nextFloorArrivalTime>0) &&  (_nextFloorArrivalTime<now) ) ;
}
// are we supposed to be stopping on pFloorNum?
```

```
//  pFloorNum starts at 1
boolean isStoppingOnFloor(int pFloorNum) {
 boolean stopping = false;
 if( (pFloorNum>0) && (pFloorNum<=NUM_FLOOR_BUTTONS) ) {
 stopping = (_prevFloorButtonValues[pFloorNum-1] != 0);
 }
 return stopping;
}

// given the current direction and floor, is there another stop in this direction?
// pDirection is either -1 (down), +1(up) or 0(stationary)
boolean isAFloorRequestedInDirection(int pDirection) {
 boolean floorRequested = false;
 int nextFloor          = _currentFloor + pDirection;
 if(pDirection!=0) {
 while( !floorRequested && (nextFloor>0) && (nextFloor<=NUM_FLOOR_BUTTONS)
) {
 log(TRACE, "   is floor requested?: ", nextFloor);
 floorRequested = isStoppingOnFloor(nextFloor);
 nextFloor     += pDirection;
 }
 log(DEBUG, "floorRequested: ", floorRequested);
 if(floorRequested) log(INFO, "  nextFloor: ", nextFloor);
 }
 return floorRequested;
}

// ----------------------------------------------------------------------------
```

```
//  State Setting Functions
// --------------------------------------------------------------------

// set our internal representation of what the Up Arrow LED should be.
// pVal: either HIGH or LOW
void setUpArrowLEDCurrentState(int pVal) {
 _currentUpArrowLedVal = pVal;
}

// set our internal representation of what the Down Arrow LED should be.
// pVal: either HIGH or LOW
void setDownArrowLEDCurrentState(int pVal) {
 _currentDownArrowLedVal = pVal;
}

// set our internal representation of what the Red Flash LED should be.
// pVal: either HIGH or LOW
void setRedFlashLEDCurrentState(int pVal) {
 _currentRedFlashLedVal = pVal;
}

// set our internal representation of what the Blue Flash LED should be.
// pVal: either HIGH or LOW
void setBlueFlashLEDCurrentState(int pVal) {
 _currentBlueFlashLedVal = pVal;
}

// set our internal representation of what pFloorNum's LED should be
```

```
// pVal: either HIGH or LOW
// pFloorNum: the floor number from 1 to 8 (NUM_FLOOR_BUTTONS)
void setFloorButtonLED(int pFloorNum, int pVal) {
 if(pVal==LOW) log(DEBUG, "turning off button for floor: ", pFloorNum);
 _prevFloorButtonValues[pFloorNum-1] = pVal;
}

void setAllFloorButtonLEDs(int pVal) {
 for(int j=1; j<=NUM_FLOOR_BUTTONS; j++) setFloorButtonLED(j, pVal);
}

// set the internal state's working variable for the pattern of what
//  should be displayed on the 7segment display to the given decimal integer.
void setCurrentSevenSegmentToDecimalNumber(int pNumForDisplay) {
 _currentSevenSegByte = BINARY_NUMBER_PATTERNS[pNumForDisplay];
 log(DEBUG, "set 7seg decimal: ", pNumForDisplay);
 log(TRACE, "  -->7seg pattern: ", (int)_currentSevenSegByte, BIN);
}

// turns on or off the decimal point
void setCurrentSevenSegmentDecimalPoint(boolean pShowDP) {
 log(TRACE, "set 7seg DP?: ", pShowDP);
 log(TRACE, "       7seg pattern BEFORE: ", (int)_currentSevenSegByte, BIN);
 if(pShowDP)  _currentSevenSegByte = _currentSevenSegByte | 1 ;
 else         _currentSevenSegByte = _currentSevenSegByte & B11111110 ;
 log(DEBUG, "  -->7seg pattern AFTER : ", (int)_currentSevenSegByte, BIN);
}
```

```
// ---------------------------------------------------------------------
//  Input Processors
// ---------------------------------------------------------------------

// checks for any button presses of any sort
void checkForAnyButtonPresses() {
 checkForFloorButtonPresses();
 // todo: check the other buttons and switches
 // read toggle button - MODETOGGLE input
 checkForModeButtonPress();
}

// TODO: since we have only one mode, this does nothing right now.
void checkForModeButtonPress() {
 //  int newMode = digitalRead(MODETOGGLE);
 /*  int newMode = analogRead(MODETOGGLE);
 if( (_mode!=ELEVATOR_MODE) && (newMode==ELEVATOR_MODE) ) {
 // we went from whackamole to elevator mode
 processModeChange(ELEVATOR_MODE);
 } else if( (_mode==ELEVATOR_MODE) && (newMode!=ELEVATOR_MODE) ) {
 // we went from elevator mode to whackamole mode
 processModeChange(WACKAMOLE_MODE);
 }
 */
}

void processModeChange(int pNewMode) {
 _mode            = pNewMode;
```

```
 _modeChangeFlag = true;
 // todo: do whatever else it takes to do a modechange.
}

// check to see if any buttons are being pushed right now
// if so, update the internal state and call the button event handler
//  does NOT update shift regs
void checkForFloorButtonPresses() {
 for(int i=1; i<=NUM_FLOOR_BUTTONS; i++) {
 if((i!=_currentFloor) && isFloorButtonJustPushed(i)) {
 // this button was just pressed, and wasn't in the pressed state before
 processFloorButtonPushEvent(i);
 }
 }
}

// ------------------------------------------------------------------------
//  Miscellaneous Functions
// ------------------------------------------------------------------------

// directions are either +1(up), -1(down) or 0(stationary)
// pDirection should be either +1 or -1
// returns an integer representing the opposite direction of pDirection -1->+1, +1-
>-1, 0->0
int getOppositeDirection(int pDirection) {
 return pDirection*=-1;
}
```

```
void log(int pLogLevel, char* pPrefix, char* pMsg)   {  if(logPrePayload(pLogLevel, pPrefix))   Serial.println(pMsg);  }
void log(int pLogLevel, char* pPrefix, int pMsg)     {  log(pLogLevel, pPrefix, pMsg, DEC);  }
void log(int pLogLevel, char* pPrefix, int pMsg, int FORMAT)     {  if(logPrePayload(pLogLevel, pPrefix))   Serial.println(pMsg, FORMAT);  }
void log(int pLogLevel, char* pPrefix, boolean pMsg) {  if(logPrePayload(pLogLevel, pPrefix))   {Serial.println(pMsg?"TRUE":"FALSE");  } }

// internal function - return true if logLevel says we should be logging this.
boolean logPrePayload(int pLogLevel, char* pPrefix) {
 boolean shouldLog = (pLogLevel<=LOG_LEVEL);
 if(shouldLog) {
 Serial.print(millis());
 Serial.print(" ");
 if(pPrefix) {
 Serial.print(pPrefix);
 Serial.print(" ");
 }
 }
 return shouldLog;
}

// ---------------------------------------------------------------------------
//  Shift Register Related Functions
// ---------------------------------------------------------------------------
// some code dealing with the 595 adapted from
http://arduino.cc/en/Tutorial/ShftOut11
```

```
//
// ARDUINO PINS:
// ------------        +-------+
// DATAPIN(13)-------->| 595 Q0|----> * UP Led
// CLOCKPIN(12)----+-->| #1  Q1|----> * DOWN Led                ie, sending
shiftOutByte a byte of 00110000
// LATCHPIN(11)-+----->|     Q2|----> * RED flashing Led          powers the RED and
BLUE flashing LEDs only.
//              | |   |     Q3|----> * BLUE flashing Led
//              | |   |     Q4|----> X
//              | |   |     Q5|----> X
//              | |   |     Q6|----> X
//              | |   |     Q7|----> X
//              | |   +-------+
//              | |       | (data)
//              | |       |
//              | |   +-------+
//              | |   | 595 Q0|----> O Button LED floor L
//              | +-->| #2  Q1|----> O Button LED floor 2      ie, sending shiftOutByte a
byte of 00001001
//              +---|-->|     Q2|----> O Button LED floor 3          turns on floor #4 and
L's LED.
//              | |   |     Q3|----> O Button LED floor 4
//              | |   |     Q4|----> O Button LED floor 5
//              | |   |     Q5|----> O Button LED floor 6
//              | |   |     Q6|----> O Button LED floor 7
//              | |   |     Q7|----> O Button LED floor 8
//              | |   +-------+
```

```
//              | |       | (data)
//              | |       |
//              | |   +-------+     +--------------------+
//              | |   | 595 Q0|---->|(DP) pin DP   *7seg* |     ie, sending shiftOutByte a
byte of 10110110
//              |  +-->| #3  Q1|---->|(A) pin A        ___  |        displays the number '2'
//              +------>|     Q2|---->|(B) pin B       |   ||      see descr. for
BINARY_NUMBER_PATTERNS above
//                      |     Q3|---->|(C) pin C       |   ||    Seven Segment display layout:
//                      |     Q4|---->|(D) pin D        ---  |         -A-
//                      |     Q5|---->|(E) pin E       |   ||        F   B
//                      |     Q6|---->|(F) pin F       |   ||       -G-
//                      |     Q7|---->|(G) pin G        ---  |        E   C
//                      +-------+     +--------------------+         -D-
//
// So, blasting out a full set of data to the entire sequence of 595's would
involve sending the 7seg's byte first,
//      then the button LED array's byte, and finally the up/down/flashing LEDs
byte

// Based on the internal state of a pile of variables, construct the 3 bytes that we
send
//  to the shift registers.
// Based on the assumption that the 595 shift registers are
//  hooked up as depicted above, it assumes that:
//  * the first byte we shift is for the seven segment display (hooked
//      up last in the chain)
//  * the second byte is for the button LEDs
```

```
//  * the third, and final byte is for the remaing LEDs
void updateShiftRegisters() {
 byte bytes[3] = {0,0,0};
 // 1) Seven Segment display
 //    easy, this is calculated for us elsewhere
 bytes[0] = _currentSevenSegByte;
 // 2) Button LEDs
 //    Use _prevFloorButtonValues array, which should now be set for what we
want to happen
 //    Q7 maps to the most significant (left most) bit, down to Q0 mapped to the
LSB (right most)
 for(int i=0; i<8; i++) {
 // bytes is initialized to 0, so we can ignore anybutton that's off
 if(_prevFloorButtonValues[i]!=0) {
 // ie, if i=2 and _prevFloorButtonValues[i]=1, we will or together bytes[1] and
00000100
 //       that will turn on floor 3's button led.
 byte currentMask = 1 << i;
 bytes[1] = bytes[1] | currentMask;
 }
 }
 // 3) other LEDs
if(_currentUpArrowLedVal==HIGH)   bytes[2] = bytes[2] | (1 << UPARROW_SHIFT);
 if(_currentDownArrowLedVal==HIGH) bytes[2] = bytes[2] | (1 <<
DOWNARROW_SHIFT);
 if(_currentRedFlashLedVal==HIGH)  bytes[2] = bytes[2] | (1 <<
REDFLASHLED_SHIFT);
```

```
 if(_currentBlueFlashLedVal==HIGH)  bytes[2] = bytes[2] | (1 <<
BLUEFLASHLED_SHIFT);

 log(DEBUG, "updateShiftRegisters ", "----------------------");
 log(DEBUG, "   byte 0 (7seg)    : ", (int)bytes[0], BIN);
 log(DEBUG, "   byte 1 (fl btns): ", (int)bytes[1], BIN);
 log(DEBUG, "   byte 2 (LEDs)    : ", (int)bytes[2], BIN);
 log(DEBUG, "/updateShiftRegisters ", "----------------------");
 // Now DO IT.
 shiftBytes(DATAPIN, CLOCKPIN, LATCHPIN, 3, bytes);
}

// shift out pNumBytes number of bytes to our 595s.  We're going to start with
the FIRST
//  byte and move forward.
// the pLatchPin is set low before all shifting, and finally set high when shifting is
done (causing the
//    shift register to affect the changes.
void shiftBytes(int pDataPin, int pClockPin, int pLatchPin, int pNumBytes, byte
*pDataOut) {
 // latchPin is low while sending data to chip.
 digitalWrite(pLatchPin, 0);
 // pinMode(pClockPin, OUTPUT);
 // pinMode(pDataPin, OUTPUT);

 //clear everything out just in case to
 //prepare shift register for bit shifting
 digitalWrite(pDataPin, 0);
```

```
 digitalWrite(pClockPin, 0);

 //internal function setup
 int i=0;
 byte *current_byte;
 for(i=0; i<pNumBytes; i++) {
 current_byte = pDataOut + i;
 shiftOutByte(pDataPin, pClockPin, *current_byte);
 }
 //stop shifting
 digitalWrite(pClockPin, 0);
 // set the latch pin high to signal chip that it no longer needs to listen for
information
 digitalWrite(pLatchPin, 1);
}

// shift out a single byte.  Start with the most significant bit (which maps to Q7),
//  all the way down to the least significant bit (which maps to Q0)
void shiftOutByte(int pDataPin, int pClockPin, byte pDataOut) {
 int i=0;
 int pinState;

 //for each bit in the byte pDataOut
 //NOTICE THAT WE ARE COUNTING DOWN in our for loop
 //This means that %00000001 or "1" will go through such
 //that it will be pin Q0 that lights.
 for (i=7; i>=0; i--)  {
 digitalWrite(pClockPin, 0);
```

```
 //if the value passed to pDataOut and a bitmask result
 // true then set pinState to HIGH, else LOW
 // ie, if we are at i=6 and our value is
 // %11010100 it would the code compares it to %01000000
 // and proceeds to set pinState to 1.
 if ( pDataOut & (1<<i) ) {
 pinState= HIGH;
 } else {
 pinState= LOW;
 }

 //Sets the pin to HIGH or LOW depending on pinState
 digitalWrite(pDataPin, pinState);
 //register shifts bits on upstroke of clock pin
 digitalWrite(pClockPin, 1);
 //zero the data pin after shift to prevent bleed through
 digitalWrite(pDataPin, 0);
 }
}

// ---------------------------------------------------------------------
// TEST ROUTINES
// ---------------------------------------------------------------------

// display 0
// when button pushed, display that button's floor num on 7seg and beep
// run this in the loop and look at the Serial output
void testSevenSegment() {
```

```
//  LOG_LEVEL=DEBUG;
log(INFO, "       high: ", HIGH);
log(INFO, "       low: ", LOW);
// display 0 and dec pt. ----------------------------------------------------
setCurrentSevenSegmentToDecimalNumber(0);
setCurrentSevenSegmentDecimalPoint(true);
updateShiftRegisters();
runBeep(FLOORBUZZLEVEL, STARTUP_BUZZ_MILLIS, STARTUP_DELAY_MILLIS);
delay(STARTUP_DELAY_MILLIS);

while(true) {
updateShiftRegisters();
checkForFloorButtonPresses();
for(int i=0; i<NUM_FLOOR_BUTTONS; i++) {
if(_prevFloorButtonValues[i]>0) {
// set 7seg to that floor's button value
setCurrentSevenSegmentToDecimalNumber(i+1);
updateShiftRegisters();
runBeep(FLOORBUZZLEVEL, STARTUP_BUZZ_MILLIS, STARTUP_DELAY_MILLIS);
// test the swipe
_currentFloor = i+1;
processDoorOpenCloseEvent();
// turn off that floor
_prevFloorButtonValues[i]=0;
updateShiftRegisters();
}
}
delay(100);
```

```
  }
}
```


มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์

ภาคผนวก ง.

RFID CODE

RFID CODE

```
int RFIDResetPin = 10;
char tag1[13] = "0100A04A51BA";
char tag2[13] = "0100A04A7299";
char tag3[13] = "0100A04AA64D";
char tag4[13] = "0100C4D9E0FC";
char tag5[13] = "************";
char tag6[13] = "************";
char tag7[13] = "************";
char tag8[13] = "************";
char tag9[13] = "************";
char tag10[13] = "************";
char tag11[13] = "************";

void(* resetFunc) (void) = 0; //declare reset function @ address 0

void setup(){
  Serial.begin(9600);
  pinMode(RFIDResetPin, OUTPUT);
  digitalWrite(RFIDResetPin, HIGH);
  //ONLY NEEDED IF CONTROLING THESE PINS - EG. LEDs
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
```

```
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);

}

void loop(){
  char tagString[13];
  int index = 0;
  boolean reading = false;

  while(Serial.available()){

    int readByte = Serial.read(); //read next available byte

    if(readByte == 2) reading = true; //begining of tag
    if(readByte == 3) reading = false; //end of tag

    if(reading && readByte != 2 && readByte != 10 && readByte != 13){
      //store the tag
      tagString[index] = readByte;
      index ++;
    }
  }

  checkTag(tagString); //Check if it is a match
  clearTag(tagString); //Clear the char of all value
  resetReader(); //eset the RFID reader
```

```
}
void checkTag(char tag[]){
////////////////////////////////////
//   REGISTER TAG NUMBER USER
////////////////////////////////////9

  if(strlen(tag) == 0) return; //empty, no need to contunue

  if(compareTag(tag, tag1)){ // register tag 1
    lightLED(2); //floor 1
    lightLED(4); //floor 3
    lightLED(6); //floor 5

  }else if(compareTag(tag, tag2)){ // register tag 2
    lightLED(2); //floor 1
    lightLED(3); //floor 2
    lightLED(5); //floor 4

  }else if(compareTag(tag, tag3)){ // register tag 3
    lightLED(2); //floor 1
    lightLED(9); //floor 8
  }else if(compareTag(tag, tag4)){ // register tag 4
    lightLED(2); //floor 1
    lightLED(3); //floor 2
    lightLED(4); //floor 3
    lightLED(5); //floor 4
    lightLED(6); //floor 5
    lightLED(7); //floor 6
```

```
    lightLED(8); //floor 7
    lightLED(9); //floor 8
  }else{
    Serial.println(tag); //read out any unknown tag
  }
}
void lightLED(int pin){
////////////////////////////////////
//Turn on LED on pin "pin" for 250ms
////////////////////////////////////
  Serial.println(pin);

  digitalWrite(pin, HIGH);
  if(pin = HIGH);
  //delay(55);
  //digitalWrite(pin, LOW);
}
void resetReader(){
////////////////////////////////////
//Reset the RFID reader to read again.
////////////////////////////////////
  digitalWrite(RFIDResetPin, LOW);
  digitalWrite(RFIDResetPin, HIGH);
  delay(150);
}

void clearTag(char one[]){
////////////////////////////////////
```

```
//clear the char array by filling with null - ASCII 0
//Will think same tag has been read otherwise
////////////////////////////////////
  for(int i = 0; i < strlen(one); i++){
    one[i] = 0;
  }
}
boolean compareTag(char one[], char two[]){
////////////////////////////////////
//compare two value to see if same,
//strcmp not working 100% so we do this
////////////////////////////////////
  if(strlen(one) == 0) return false; //empty
  for(int i = 0; i < 12; i++){
    if(one[i] != two[i]) return false;
  }
  return true; //no mismatches
}
```

ประวัติผู้วิจัย

ประวัติผู้วิจัย

1. ชื่อ - นามสกุล (ภาษาไทย) นายวรุตม์ บุญเลี่ยม

   ชื่อ - นามสกุล (ภาษาอังกฤษ) Mr. Waroot Boonliam

2. เลขหมายบัตรประจำตัวประชาชน 3 779800044 09 2

3. ตำแหน่งปัจจุบัน พนักงานมหาวิทยาลัย (อาจารย์)

4. หน่วยงานและสถานที่อยู่ที่ติดต่อได้สะดวก พร้อมหมายเลขโทรศัพท์ที่ทำงาน โทรศัพท์มือถือ โทรสาร และไปรษณีย์อิเล็กทรอนิกส์ (e-mail)

คณะอุตสาหกรรมและเทคโนโลยี มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์ ถ. เพชรเกษม ต.หนองแก อ.หัวหิน จ.ประจวบคีรีขันธ์ 77110 โทรศัพท์ 0-3261-8500 ต่อ 4033 โทรสาร 0-3261-8570 E-nail k_waroot@yahoo.com ,waroot.boo@rmutr.ac.th

5. ประวัติการศึกษา

ปีที่จบ	ระดับปริญญา	ชื่อย่อปริญญา	สาขาวิชา	สถาบัน	ประเทศ
2552	ปริญญาโท	วท.ม.	เทคโนโลยีคอมพิวเตอร์เพื่อการศึกษา	มหาวิทยาลัยราชภัฏเพชรบุรี	ไทย
2543	ปริญญาตรี	วศ.บ.	วิศวกรรมคอมพิวเตอร์	มหาวิทยาลัยเทคโนโลยีมหานคร	ไทย

6. สาขาวิชาการที่มีความชำนาญพิเศษ (แตกต่างจากวุฒิการศึกษา) ระบุสาขาวิชาการ

   - ไมโครคอนโทรลเลอร์

7. ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยทั้งภายในและภายนอกประเทศ โดยระบุสถานภาพในการทำการวิจัยว่าเป็นผู้อำนวยการแผนงานวิจัย หัวหน้าโครงการวิจัย หรือผู้ร่วมวิจัย ในแต่ละข้อเสนอการวิจัย

ประวัติผู้วิจัย

1. ชื่อ - นามสกุล (ภาษาไทย) นายพรประสิทธิ์ บุญทอง
   ชื่อ - นามสกุล (ภาษาอังกฤษ) Mr. Pornprasit Boontong
2. เลขหมายบัตรประจำตัวประชาชน 3 8102 00065 561
3. ตำแหน่งปัจจุบัน อาจารย์ 7
4. หน่วยงานและสถานที่อยู่ที่ติดต่อได้สะดวก พร้อมหมายเลขโทรศัพท์ที่ทำงาน โทรศัพท์มือถือ โทรสาร และไปรษณีย์อิเล็กทรอนิกส์ (e-mail)

   คณะอุตสาหกรรมและเทคโนโลยี มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์ ถ. เพชรเกษม ต.หนองแก อ.หัวหิน จ.ประจวบคีรีขันธ์ 77110 โทรศัพท์ 0-3261-8500 ต่อ 4033 โทรสาร 0-3261-8570 E-nail pornprasit@rmutr.ac.th

5. ประวัติการศึกษา

ปีที่จบ	ระดับปริญญา	ชื่อย่อปริญญา	สาขาวิชา	สถาบัน	ประเทศ
2550	ปริญญาโท	วศ.ม.	วิศวกรรมคอมพิวเตอร์	สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง	ไทย
2539	ปริญญาตรี	วศ.บ.	วิศวกรรมคอมพิวเตอร์	สถาบันเทคโนโลยีราชมงคล	ไทย

6. สาขาวิชาการที่มีความชำนาญพิเศษ (แตกต่างจากวุฒิการศึกษา) ระบุสาขาวิชาการ
   - ระบบเครือข่าย
   - ระบบฐานข้อมูล

7. ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยทั้งภายในและภายนอกประเทศ โดยระบุสถานภาพในการทำการวิจัยว่าเป็นผู้อำนวยการแผนงานวิจัย หัวหน้าโครงการวิจัย หรือผู้ร่วมวิจัยในแต่ละข้อเสนอการวิจัย

ประวัติผู้วิจัย

1. ชื่อ - นามสกุล (ภาษาไทย) นายอาทิตย์ อยู่เย็น
   ชื่อ - นามสกุล (ภาษาอังกฤษ) Mr. Arthit Yooyen
2. เลขหมายบัตรประจำตัวประชาชน 3 770100018 38 4
3. ตำแหน่งปัจจุบัน พนักงานมหาวิทยาลัย (อาจารย์)
4. หน่วยงานและสถานที่อยู่ที่ติดต่อได้สะดวก พร้อมหมายเลขโทรศัพท์ที่ทำงาน โทรศัพท์มือถือ โทรสาร และไปรษณีย์อิเล็กทรอนิกส์ (e-mail)

คณะอุตสาหกรรมและเทคโนโลยี มหาวิทยาลัยเทคโนโลยีราชมงคลรัตนโกสินทร์ ถ. เพชรเกษม ต.หนองแก อ.หัวหิน จ.ประจวบคีรีขันธ์ 77110 โทรศัพท์ 0-3261-8500 ต่อ 4033 โทรสาร 0-3261-8570 E-nail thit19@hotmail.com, thit19@gmail.com

5. ประวัติการศึกษา

ปีที่จบ	ระดับปริญญา	ชื่อย่อปริญญา	สาขาวิชา	สถาบัน	ประเทศ
2554	ปริญญาโท	วท.ม.	เทคโนโลยีสารสนเทศ	มหาวิทยาลัยเทคโนโลยีมหานคร	ไทย
2547	ปริญญาตรี	อส.บ.	เทคโนโลยีคอมพิวเตอร์	สถาบันเทคโนโลยีราชมงคล วข.วังไกลกังวล	ไทย

6. สาขาวิชาการที่มีความชำนาญพิเศษ (แตกต่างจากวุฒิการศึกษา) ระบุสาขาวิชาการ
   - ไมโครคอนโทรลเลอร์

7. ประสบการณ์ที่เกี่ยวข้องกับการบริหารงานวิจัยทั้งภายในและภายนอกประเทศ โดยระบุสถานภาพในการทำการวิจัยว่าเป็นผู้อำนวยการแผนงานวิจัย หัวหน้าโครงการวิจัย หรือผู้ร่วมวิจัยในแต่ละข้อเสนอการวิจัย