แบบจำลองที่จอดรถอัตโนมัติแบบโรโตรี่โดยใช้ไมโครคอนโทรลเลอร์ควบคุม

AUTOMATED PARKING ROTARY MODEL USING MICROCONTROLLER

นายกิตติวาท พุทธิมณี

นายณัฐพล พรายวัน

นายปิติภัทร ตันเขมจารี

โครงงานวิศวกรรมนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมไฟฟ้า

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ปีการศึกษา 2559

แบบจำลองที่จอดรถอัตโนมัติแบบโรโตรี่โดยใช้ไมโครคอนโทรลเลอร์ควบคุม

AUTOMATED PARKING ROTARY MODEL USING MICROCONTROLLER

นายกิตติวาท พุทธิมณี
นายณัฐพล พรายวัน
นายปิติภัทร ตันเขมจารี

โครงงานวิศวกรรมนี้ เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ
ปีการศึกษา 2559

โครงงานวิศวกรรม

เรื่อง

แบบจำลองที่จอดรถอัตโนมัติแบบโรตารี่โดยใช้ไมโครคอนโทรลเลอร์ควบคุม

ของ

นายกิตติวาท พุทธิมณี

นายณัฐพล พรายวัน

นายปิติภัทร ตันเขมจารี

ได้รับอนุมัติจากคณะวิศวกรรมศาสตร์ให้นับเป็นส่วนหนึ่งของการศึกษาตามหลักสูตร

วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมไฟฟ้า

ของมหาวิทยาลัยศรีนครินทรวิโรฒ

.....คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.เวคิน ปิยรัตน์)

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ปฐมทัศน์ จิระเดชะ)

.....กรรมการ

(อาจารย์ ดร.ธนาธิป สุ่มอิ่ม)

.....กรรมการ

(อาจารย์ ดร.คมกฤษ ประเสริฐวงษ์)

แบบจำลองที่จอดรถอัตโนมัติแบบโรตารี่โดยใช้ไมโครคอนโทรลเลอร์ควบคุม
ปีการศึกษา 2559

โดย

นายกิตติวาท พุทธิมณี
นายณัฐพล พรายวัน
นายปิติภัทร ตันเขมจารี

อาจารย์ที่ปรึกษา

อาจารย์ ดร.คมกฤษ ประเสริฐวงษ์

บทคัดย่อ

โครงการวิศวกรรมฉบับนี้ เป็นรูปแบบการจำลองการทำงานที่จอดรถอัตโนมัติแบบโรตารี่ (rotary parking system) โดยใช้ไมโครคอนโทรลเลอร์ในการควบคุมการทำงานของที่จอดรถแบบอัตโนมัติ ซึ่งแบบจำลองนี้สามารถบรรจุรถได้ทั้งหมด 6 คัน โดยใช้ NFC (Near Field Communication) ในการระบุรหัสผู้ใช้งาน โดยใช้การเขียนโปรแกรมลงบนไมโครคอนโทรลเลอร์จะทำหน้าที่ประมวลผลการทำงาน โดยการควบคุมตำแหน่งของกระเช้า จะรับค่าจากเซนเซอร์นำไปประมวลผลในการควบคุมตำแหน่งกระเช้า โดยใช้มอเตอร์ดีซีเกียร์ ในการเคลื่อนที่ตำแหน่งกระเช้า จึงเกิดเป็นแนวความคิดที่จะมาประยุกต์ใช้ในพื้นที่ที่จำกัด การทำงานทั้งทางด้านควบคุมและการบังคับนั้นจะใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงาน ซึ่งจะรับข้อมูลรหัสผู้ใช้จาก NFC และไปสั่งการดีซีมอเตอร์เกียร์ในการเคลื่อนที่ของกระเช้า และระบุตำแหน่งของกระเช้าโดยการรับค่าจากเซนเซอร์ ซึ่งข้อมูลจะไปแสดงผลบนหน้าจอ LCD

คำสำคัญ: การควบคุมแบบพีดับเบิลยูเอ็ม ระบบควบคุมการเคลื่อนที่แบบเพิ่ม ระบบควบคุมกำลังบิด ระบบชี้เฉพาะอัตโนมัติ ระบบแฮนด์ดาวน์โพลิ่ง แรงเคลื่อนเหนี่ยวนำป้อนกลับ

AUTOMATED PARKING ROTARY MODEL USING MICROCONTROLLER

Academic Year 2016

By

Mr. Kittiwart Puttimanee

Mr. Nattaphon Paywan

Mr. Pitipat Tankhamjaree

Advisor

Mr. Komkrit Prasertwong

Abstract

The Project is a present the rotary car parking model which controlled by Arduino microcontroller. The model can parking maximum six cars. The RFID module with Near Field Communication (NFC) is used to specify the User ID. The DC motor with gear used to drive the car parking system. The optical sensor is used to detect position of the basket. When the car is incoming, the nearest empty basket will come receive the car and the driver will get the User ID. This User ID will be used to identify where the car parking which display on the LCD screen. The optical sensor detects the basket and sends the pulse signal to microcontroller to count the pulse signal to identify the position of the car.

Keywords: Pulse Width Modulation, Incremental Motion Control System, Automatic Identification, Hands Down Polling System, Back Electromotive Force

กิตติกรรมประกาศ

โครงงานวิศกรรมศาสตร์นี้ จะสำเร็จลุล่วงลงได้ทางคณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์ ดร. คมกฤษ ประเสริฐวงษ์ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงานวิศวกรรมที่ให้คำปรึกษาและคำแนะนำต่างๆ จนทำให้โครงงานวิศวกรรมสำเร็จลุล่วงไปด้วยดีและขอบพระคุณอาจารย์ประจำภาควิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ทุกท่านที่ให้ความรู้และข้อเสนอแนะ รวมทั้งเป็นกรรมการในการตรวจสอบและตัดสินในโครงงานวิศวกรรมนี้ ทำให้โครงงานวิศวกรรมนี้มีคุณภาพและสมบูรณ์ได้ รวมทั้งเจ้าหน้าที่ทุกท่านในภาควิชาวิศวกรรมไฟฟ้าที่ได้ให้ความสะดวกแก่คณะผู้จัดทำ

ขอขอบพระคุณ บิดา มารดา และอาจารย์ที่ให้คำอบรมสั่งสอนให้ความรู้ตลอดจนการสนับสนุนคณะผู้จัดทำสามารถจัดทำโครงงานวิศวกรรมนี้ได้และขอบคุณทุกคนที่ให้กำลังใจเสมอมา

คณะผู้จัดทำโครงงาน

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

รายการสัญลักษณ์

สัญลักษณ์	คำอธิบาย	หน่วย
E	แรงเคลื่อนไฟฟ้า	โวลต์
F	แรง	นิวตัน
g	ความโน้มถ่วงพื้นผิว	เมตรต่อวินาทีกำลังสอง
I	กระแสไฟฟ้า	แอมแปร์
m	มวล	กรัม
N	ความเร็วรอบ	รอบต่อนาที
P	กำลังไฟฟ้า	วัตต์
R	ความต้านทานทางไฟฟ้า	โอห์ม
r	ขนาดของรัศมี	เมตร
T	แรงบิด	นิวตัน-เมตร
V	แรงดันไฟฟ้า	โวลต์

ประมวลคำย่อ

คำย่อ	คำอธิบาย
C + +	C Programming Language
DC	Direct Current
EEPROM	Electrical Erasable Programmable Read Only Memory
I / O	Input / Output
LCD	Liquid Crystal Display
LED	Light Emitting Diode
NFC	Near Field Communication
PWM	Pulse Width Modulation
RFID	Radio Frequency Identification
RISC	Reduced Instruction Set Computer
RPM	Revolutions Per Minute
SRAM	Static RAM

บทที่ 1
บทนำ

1.1 ความเป็นมาของโครงงาน

ในปัจจุบันความต้องการในการใช้ลานจอดรถยนต์ในที่สาธารณะหรือสถานที่ที่มีคนพลุกพล่าน มีความต้องการอย่างมาก ซึ่งเป็นที่ทราบกันดีว่าปัญหาเหล่านั้นเกิดจากจำนวนรถรถยนต์ที่เพิ่มมากขึ้นเรื่อยๆ ซึ่งปัจจุบันจำนวนรถยนต์เพิ่มขึ้นเป็นจำนวนมาก ในขณะที่จอดรถมีพื้นที่เท่าเดิม ปัญหารถติดในปัจจุบันส่วนหนึ่งมาจาก การที่รถจอดที่ไม่เป็นระเบียบ เช่น จอดรถในบริเวณที่ห้ามจอด จอดรถซ้อนคันกัน เป็นต้น ซึ่งการที่จำนวนรถมีมากขึ้นเพราะ การซื้อขายรถในปัจจุบันมีราคาถูก และรถบางคันไม่มีที่จอดที่แน่นอน

ในกรุงเทพมหานครมีประชากรจำนวนมาก จึงทำให้ที่จอดรถไม่เพียงพอต่อจำนวนรถที่มีอยู่ในกรุงเทพ ในปัจจุบันกรุงเทพมหานครมีบริการที่จอดรถแบบลานปกติซึ่งเป็นที่ราบและกว้างซึ่งจะต้องใช้พื้นที่จำนวนมากและจอดรถได้ไม่กี่คัน และเป็นแบบอาคารที่ไว้สำหรับจอดรถ ซึ่งหากเป็นอาคารใหญ่และจะต้องมีทางให้รถวิ่งที่ขนาดกว้างเพียงพอเพื่อให้รถยนต์ถอยเข้าออก และมี Ramp ทางลาดยกระดับที่ต้องกว้างเพื่อให้รถเข้าโค้งได้พอดี ซึ่งพื้นที่ส่วนนี้ใช้พื้นที่จำนวนมาก ซึ่งพื้นที่เศรษฐกิจสำคัญในกรุงเทพมหานคร ที่มีราคาสูงเพราะฉะนั้นจะต้องใช้พื้นที่ขนาดเล็กให้มีประโยชน์สูงสุด

จากสาเหตุดังกล่าวกลุ่มกระผมจึงเล็งเห็นประโยชน์ของที่จอดรถอัตโนมัติ ที่จะมีประโยชน์ในอนาคตอันใกล้ จึงมีแนวคิดที่จะสร้างแบบจำลองที่จอดรถอัตโนมัติแบบ Rotary โดยใช้ไมโครคอนโทรลเลอร์เป็นตัวควบคุมการทำงาน เพื่อเป็นแบบจำลองที่อาจจะนำไปสร้างจริง เพื่อในพื้นที่ที่มีประชากรหนาแน่นมีที่จอดรถที่เพียงพอสะดวกสบาย และมีที่จอดรถที่เป็นหลักแหล่งแน่นนอน ไม่ต้องจอดรถริมถนนซึ่งเป็นส่วนหนึ่งของปัญหาจราจรติดขัด และปัญหาเกี่ยวกับข้อกฎหมายจราจรเกี่ยวกับการจอดรถ

1.2 วัตถุประสงค์

1.2.1 เพื่อออกแบบและสร้างแบบจำลองที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี ที่ใช้ Microprocessor ชนิดArduino UNO R3 เป็นตัวควบคุมระบบการทำงาน

1.2.2 เพื่อศึกษาการเขียนโปรแกรมที่ใช้ควบคุมไมโครคอนโทรลเลอร์ ในการสั่งการทำงานของระบบที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี

1.2.3 เพื่อศึกษาการควบคุมมอเตอร์กระแสตรงโดยใช้ไมโครคอนโทรลเลอร์

1.2.4 เพื่อศึกษาการควบคุมการทำงานของระบบเชื่อมต่อระยะสั้น (NFC)

1.3 ขอบเขตโครงงานวิศวกรรม

1.3.1 ออกแบบระบบควบคุมการนำรถเข้าออกของที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี โดยใช้ไมโครคอนโทรลเลอร์ Arduino UNO R3 เป็นตัวควบคุมการทำงาน

1.3.2 แบบจำลองที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี มีจำนวนกระเช้าจอดรถ 6 กระเช้า

1.3.3 เป็นแบบจำลองการเรียกและเก็บรถทีละ 1 คัน

1.3.4 กระเช้าในแบบจำลองจะเคลื่อนที่ในทิศทางเดียวเท่านั้น

1.3.5 ระบบสามารถจะตรวจจับตำแหน่งกระเช้าโดยใช้เซนเซอร์แสง 1 ตัว ที่ติดตั้งอยู่ที่ใต้ตำแหน่งกระเช้านำรถเข้าและออก โดยระยะห่างเท่ากับ 2 เซนติเมตร

1.3.6 ในจอ LCD สามารถแสดงสถานะการจอดรถในแต่ละกระเช้า โดยสถานะ 1 จะมีรถจอดอยู่ และ สถานะ 0 จะไม่มีรถจอดอยู่

1.4 ประโยชน์ที่จะได้รับ

1.4.1 สามารถสร้างโปรแกรมที่ใช้จำลองการควบคุมระบบจัดเก็บรถอัตโนมัติได้

1.4.2 ได้ทราบถึงการทำงานของระบบควบคุมที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี

1.4.3 ได้ชุดทดสอบคุณสมบัติการทำงานของที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี ที่มี Microprocessor เป็นระบบควบคุม

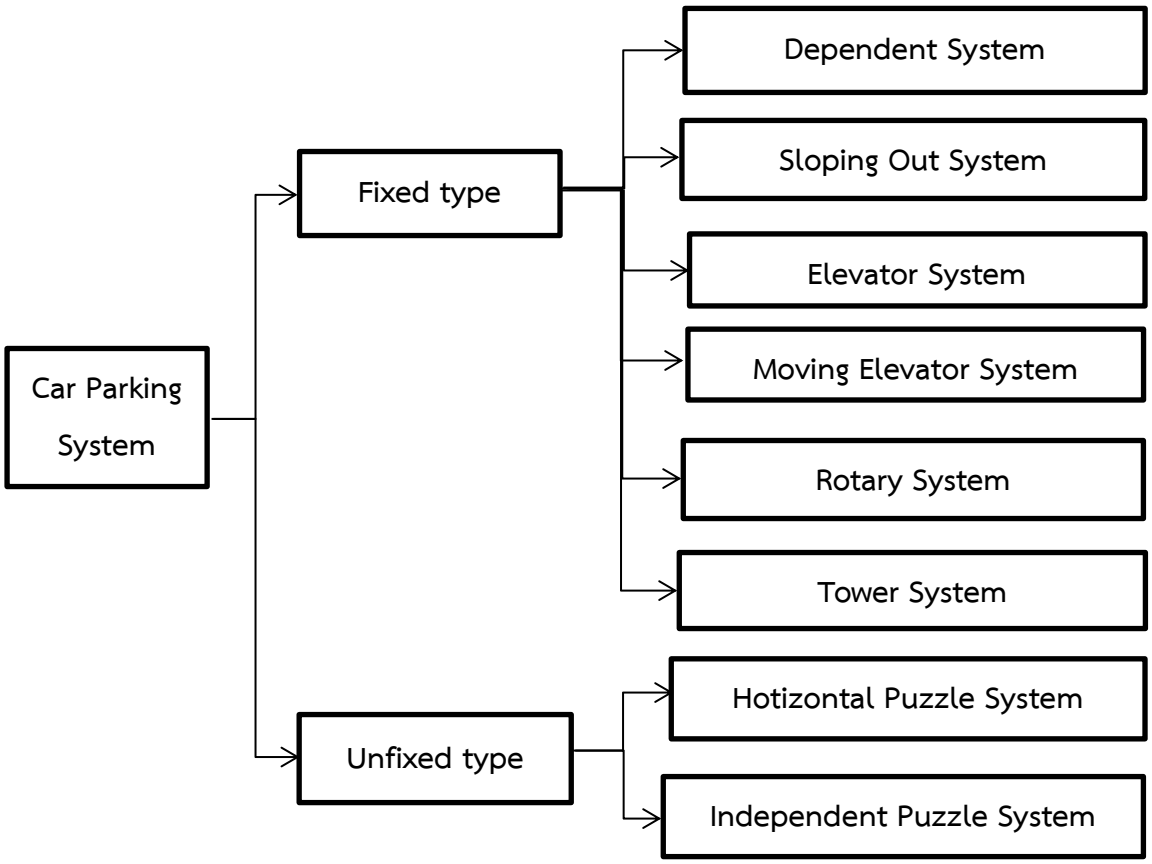
1.4.4 แบบจำลองสามารถนำไปประยุกต์ใช้งานในด้านอุตสาหกรรมได้

1.4.5 ทำให้เกิดความสะดวกสบายในการจอดรถมากยิ่งขึ้น

บทที่ 2
ทฤษฎีและผลงานวิจัยที่เกี่ยวข้อง

2.1 รูปแบบของระบบจอดรถอัตโนมัติ

ปัจจุบันระบบจอดรถอัตโนมัติมีการแข่งขันและพัฒนาขึ้นเรื่อยๆ จึงทำให้เกิดรูปแบบที่แตกต่างกันออกไปตามแต่ความต้องการของผู้ออกแบบ เพื่อตอบสนองความต้องการที่แตกต่างกันออกไป ไม่ว่าจะเป็น พื้นที่ ความสะดวกรวดเร็ว ความปลอดภัย ทำให้มีรูปแบบที่จอดรถออกมามากมาย จึงได้ทำการสรุปและได้แบ่งประเภทของระบบจอดรถอัตโนมัติออกเป็น 2 ประเภทใหญ่ๆได้คือ แบบตายตัว (Fixed Type) และ แบบไม่ตายตัว (Unfixed Type) ดังรูปที่ 2.1



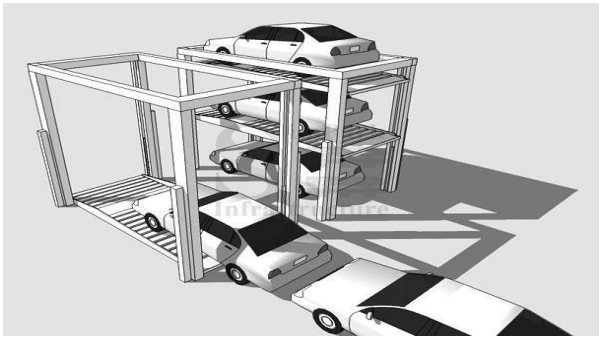
รูปที่ 2.1 รูปแบบของระบบจอดรถอัตโนมัติ

2.1.1 แบบตายตัว (Fixed Type)

เป็นแบบที่มีความนิยมและความซับซ้อนไม่มาก โดยวิธีการนำรถไปวางตำแหน่งที่ว่างและระบบภายในไม่ทำให้รถที่จอดอยู่เดิมเปลี่ยนตำแหน่ง แบ่งเป็นรูปแบบโครงสร้างได้ดังต่อไปนี้

2.1.1.1 Dependent System

เป็นแบบที่ง่ายที่สุดโดยมีหลักการง่ายๆ คือนำรถเข้ามาจอดแล้วยกรถขึ้นไปไว้ด้านบน มีข้อเสีย เมื่อจอดรถเต็มแล้ว คันที่อยู่ด้านบนจะไม่สามารถนำรถออกมาได้ ต้องให้รถด้านล่างออกมาก่อนจึงจะออกมาได้

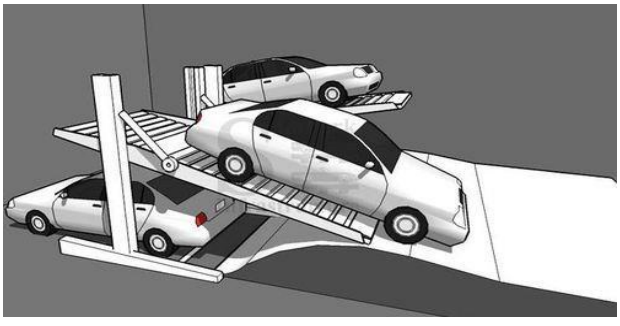


รูปที่ 2.2 Dependent System

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Dependent-jack-system2.jpg

2.1.1.2 Sloping Out System

มีความคล้ายคลึงกับแบบแรกแต่ดัดแปลงพื้นที่ให้เป็นทางลดเพื่อที่เมื่อรถคันชั้นบนหรือชั้นล่างจะออกจะสามารถออกได้โดยไม่ต้องอาศัยกัน

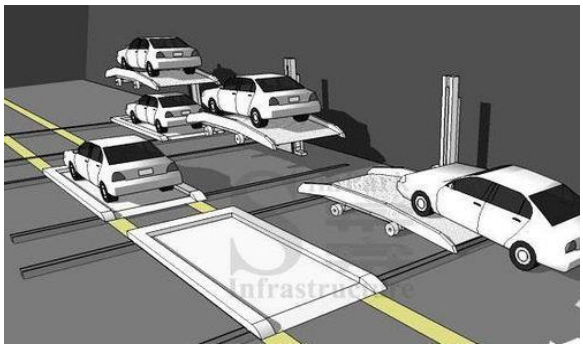


รูปที่ 2.3 Sloping Out System

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Slopping-out-System.jpg

2.1.1.3 Elevator System หรือแบบ Lift

แบบนี้มีความคล่องตัวสูงใช้พื้นที่ในการก่อสร้างน้อยเมื่อต้องการที่จอดรถมากๆจะทำโดยการสร้างให้สูงๆ หรือลงใต้ดินหลายๆชั้นนิยมมากในตึกสำนักงานขนาดกลาง

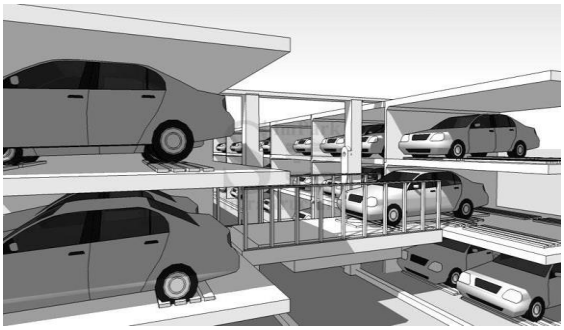


รูปที่ 2.4 Elevator System หรือแบบ Lift

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Lift-and-slideout-system.jpg

2.1.1.4 Moving Elevator System

จะคล้ายกับ Elevator System ต่างกันตรงที่ตัวพารถสามารถเคลื่อนที่ไปในแนวลึกได้จึงทำให้แบบนี้สามารถจอดรถได้จำนวนมากและยังมีความซับซ้อนไม่มาก

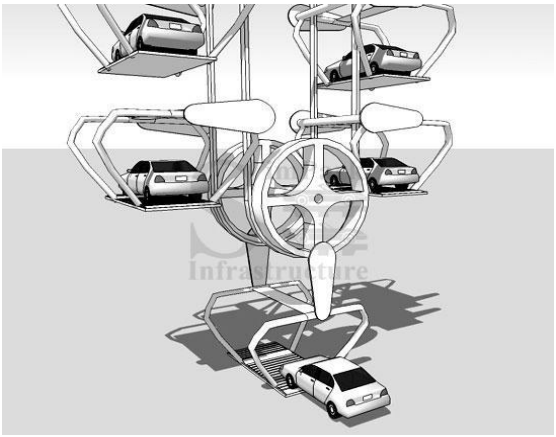


รูปที่ 2.5 Moving Elevator System

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Moving-elevator-system.jpg

2.1.1.5 Rotary System

ทำงานเหมือนชิงช้าสวรรค์ทำให้รถเข้าไปอยู่ในตัวกระเช้าแล้วหมุนไปขึ้นไว้ด้านบนเมื่อต้องการนำรถออกก็จะหมุนมายังจุดรับรถ

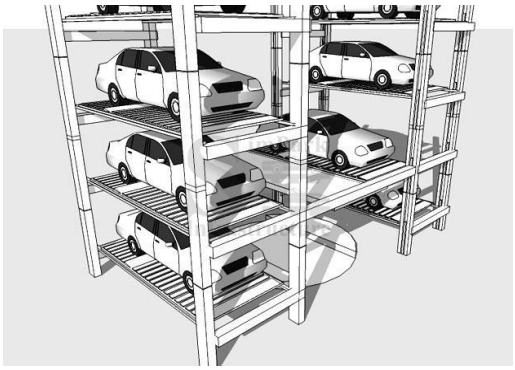


รูปที่ 2.6 Rotary System

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Rotary-system.jpg

2.1.1.6 Tower System

สำหรับพื้นที่ที่เป็นจัตุรัส โดยสร้างตึกเป็นรูปทรงกระบอกแล้วมีแกนกลางเป็นตัวพารถไปจอดยังตำแหน่งที่ต้องการ เมื่อต้องการนำรถออกก็ให้ตัวพารถไปรับรถยังตำแหน่งที่รถจอด



รูปที่ 2.7 Tower System

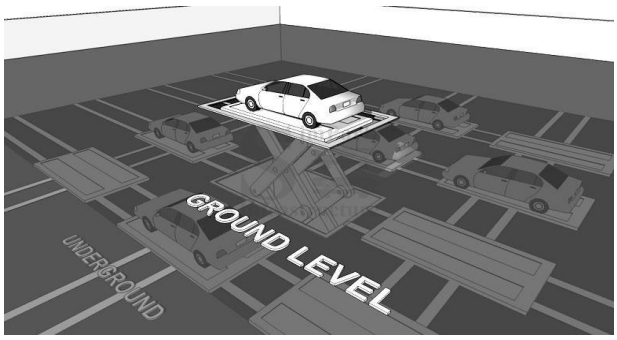
ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Tower-parking-system2.jpg

2.1.2 แบบไม่ตายตัว (Unfixed Type)

เป็นแบบที่ใช้พื้นที่ได้คุ้มค่าที่สุดมีความซับซ้อน แต่ละตำแหน่งที่รถจอดสามารถเคลื่อนที่เปลี่ยนตำแหน่งได้ทุกทิศทาง โดยรถที่นำเข้ามาจอดจะไม่อยู่ตำแหน่งเดิมตลอดเพราะต้องสลับกันไปมาเหมือนเกมส์เรียงภาพ แบ่งประเภทโครงสร้างได้ดังต่อไปนี้

2.1.2.1 Horizontal Puzzle System

เป็นเคลื่อนย้ายตามแนวนอนเมื่อรถเข้ามาจอดระบบจะมีพาเลทมารับแล้วนำไปยังตำแหน่งที่ว่าง เมื่อจะนำรถออกระบบจะสลับเปลี่ยนไปมาเพื่อหาช่องที่รถจะออกทุกตำแหน่งสามารถเคลื่อนที่ได้อย่างอิสระ

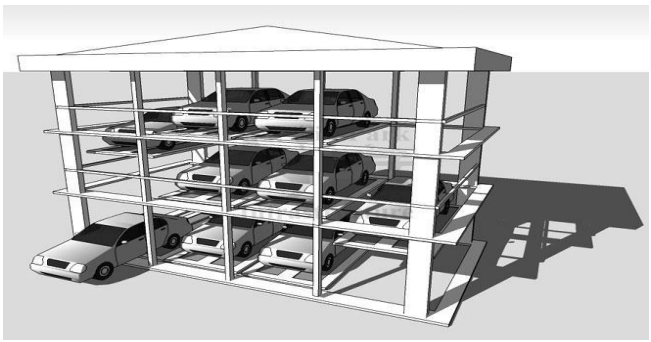


รูปที่ 2.8 Horizontal Puzzle System

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Horizonal-puzzle-system.jpg

2.1.2.2 Independent Puzzle System

เป็นแบบเคลื่อนย้ายตามแนวตั้งทุกตำแหน่งสามารถเคลื่อนที่ได้อย่างอิสระเช่นกัน



รูปที่ 2.9 Independent Puzzle System

ที่มา: http://simparkinfrastructure.com/inc/wp-content/productpics/Independent-puzzle-system.jpg

2.2 มอเตอร์กระแสตรง

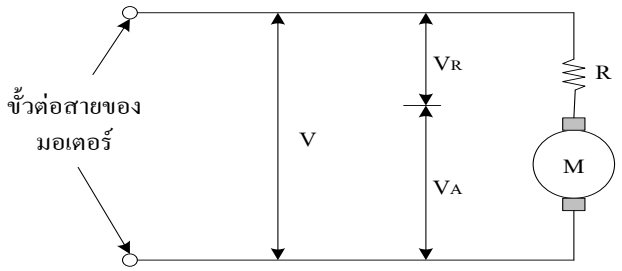
2.2.1 หลักการทำงานของมอเตอร์กระแสตรง

เมื่อมีการผ่านกระแสไฟฟ้าเข้าไปยังขดลวดในสนามแม่เหล็กจะทำให้เกิดแรงแม่เหล็กซึ่งมีสัดส่วนของแรงขึ้นกับกระแสแรงของสนามแม่เหล็ก โดยแรงจะเกิดขึ้นเป็นมุมฉากกับกระแสและสนามแม่เหล็ก ขณะที่ทิศทางของแรงกลับตรงกันข้ามกัน ถ้าหากกระแสของสนามแม่เหล็กไหลย้อนกลับจะทำให้เกิดการเปลี่ยนแปลงของกระแส และ สนามแม่เหล็กเป็นผลทำให้ทิศทางของแรงเปลี่ยนไป ด้วยคุณสมบัตินี้ทำให้มอเตอร์กระแสตรงกลับทิศทางการหมุนได้

สนามแม่เหล็กของมอเตอร์ส่วนหนึ่งเกิดขึ้นจากแม่เหล็กถาวรซึ่งจะถูกยึดติดกับแผ่นเหล็กหรือ เหล็กกล้า โดยปกติส่วนนี้จะเป็นส่วนที่ยึดอยู่กับที่ และ ขดลวดเหนี่ยวนำจะพันอยู่กับส่วนที่เป็นแกนหมุนของมอเตอร์

2.2.2 คุณสมบัติของมอเตอร์กระแสตรง

ในการอธิบายคุณสมบัติของมอเตอร์กระแสตรงให้ละเอียดนั้นต้องพิจารณาแรงดันที่ป้อนและความต้านทานของโรเตอร์ด้วย วงจรภายในของมอเตอร์เขียนได้ดังรูปที่ 2.10



รูปที่ 2.10 วงจรภายในของมอเตอร์กระแสตรง

ที่มา: http://www.rmutphysics.com/charud/scibook/electromagnetic1/magnetic/pic/problem/3-41.gif

โดยสมมติให้ทุ่นโรเตอร์ไม่มีความต้านทานอยู่เลย อนุกรมกับความต้านทานซึ่งในที่นี้ก็คือความต้านทานของขดลวดนั่นเอง แรงดันที่ขั้วต่อสายของมอเตอร์ก็คือผลบวกระหว่างแรงดันที่ทุ่นโรเตอร์ (V_A) และ แรงดันตกคร่อมความต้านทานขดลวด (V_R)

แรงดัน V_A ถูกเรียกว่า แรงเคลื่อนเหนี่ยวนำป้อนกลับ (BACK EMF) ซึ่งเกิดขึ้นในโรเตอร์ขณะที่หมุนแรงดันที่เกิดขึ้นนี้เป็นไปตากกฎของการเหนี่ยวนำแม่เหล็กไฟฟ้าจากการเคลื่อนที่ของตัวนำในสนามแม่เหล็ก สัมพันธ์กับแรงเคลื่อนเหนี่ยวนำแม่เหล็ก และ ความเร็วในการเคลื่อนที่ของตัวนำ แรงดันที่เกิดขึ้นจะมีขั้วตรงกันข้ามกับแรงดันที่ป้อนให้กับมอเตอร์ และ แปรผันตรงกับความเร็วในการหมุน ผลบวก

ของแรงดันที่ทุ่นโรเตอร์ (V_A) และแรงดันตกคร่อมขดลวด (V_R) ต้องเท่ากับแรงดันที่ป้อนให้กับมอเตอร์ (V)

$$V = V_A + V_B \quad (V) \tag{2.1}$$

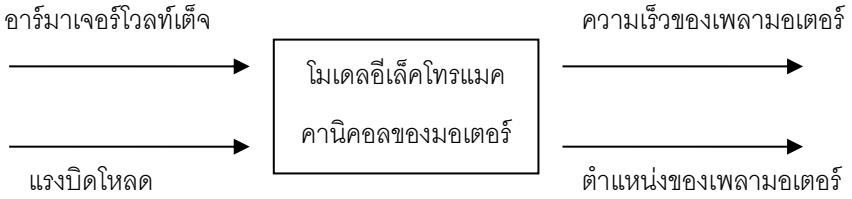
เมื่อพิจารณาตั้งแต่มอเตอร์หยุดนิ่ง ความเร็วมีค่าเป็นศูนย์ ดังนั้น $V_A = 0$, $V_R = V$ กระแสที่ไหลในมอเตอร์หาได้จาก

$$I = \frac{V_R}{R} \quad (A) \tag{2.2}$$

เมื่อมอเตอร์เริ่มหมุนจะมีความเร็ว และ V_A เพิ่มขึ้นเป็นเส้นตรงตามความเร็ว V_R ซึ่งมีค่าเท่ากับความแตกต่างระหว่าง V_A และ V จะเริ่มลดลงกระแส I ก็จะเริ่มลดลงเช่นกันขณะที่มอเตอร์ยังมีความเร่งอยู่ ความเร็วจะเพิ่มขึ้น แรงบิดจะลดลงจนกว่าจะถึงจุดซึ่งแรงบิดจองมอเตอร์รับภาระโหลดได้สมดุลพอดี ขณะที่มอเตอร์ไม่มีโหลด และ หมุนอย่างอิสระจะมีเพียงค่าความฝืดของแบริ่ง และ แรงต้านอากาศทำให้ V_A เกือบเท่ากับค่า V

2.2.3 โมเดลคณิตศาสตร์ของดีซีมอเตอร์

ดีซีมอเตอร์ที่ใช้ร่วมกับดีซีแอมปลิไฟล์ทั้งในระบบการบังคับตำแหน่งและการบังคับความเร็วมักจะได้รับการประยุกต์ใช้เป็นส่วนประกอบสร้างกำลังงานในระบบการนำร่องและระบบบังคับต่างๆ และเนื่องจากวิทยาการเกี่ยวกับสารแม่เหล็กและการขยายด้วยโซลิคสเตททำให้ดีซีมอเตอร์แบบแม่เหล็กถาวรได้รับความนิยมใช้เป็นส่วนประกอบการขับเคลื่อนในระบบการบังคับแบบปิดลูฟต่างๆ มากขึ้น การออกแบบและการชดเชยระบบดังกล่าวได้อย่างเหมาะสมจะต้องใช้โมเดลทางคณิตศาสตร์ของส่วนประกอบทั้งหมดในระบบ



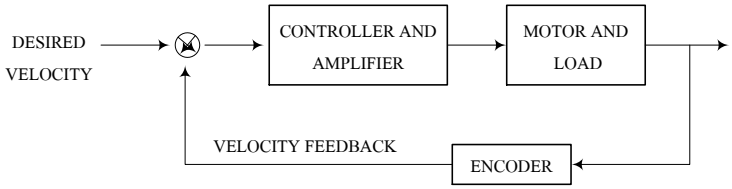
รูปที่ 2.11 แสดงอินพุตและเอาต์พุตของโมเดลทางคณิตศาสตร์ของมอเตอร์

ที่มา: http://www.rmutphysics.com/charud/scibook/electromagnetic1/magnetic/pic/problem/3-41.gif

2.2.4 หลักการควบคุมความเร็ว และ ตำแหน่ง

รูปที่ 2.12 คือ บล็อกไดอะแกรมของระบบควบคุมความเร็วในระบบกลไกแบบเซอร์โวนี้ มอเตอร์ถูกเรียกว่า มอเตอร์แบบเซอร์โว (servo motor) เครื่องวัดรอบในลูปแบบป้อนกลับจะวัดความเร็วของมอเตอร์แบบเซอร์โว และ ส่งป้อนกลับมาในรูปของสัญญาณไฟฟ้า (แรงดัน หรือ กรพะแส) ซึ่ง แปลตามความเร็วเพลาของมอเตอร์ ในที่นี้ลูปป้อนกลับจะทำให้ความเร็วเอาท์พุทของมอเตอร์มีค่าคงที่มากขึ้น

ระบบแบบ closed loop ถูกใช้รวมอยู่ในระบบเครื่องมือ เครื่องจักร หรือ สว่านกำลังเพื่อชดเชยความแตกต่างของโหลดหรือวัสดุที่ถูกตัดหรือถูกเจาะ วัสดุเนื้อแข็งจะหน่วงความเร็วของสว่าน หรือเครื่องมือที่ใช้ในการตัด และ สว่านจะมีความเร็วเพิ่มขึ้นในวัสดุเนื้ออ่อน



รูปที่ 2.12 ระบบควบคุมความเร็ว ที่ประกอบด้วยลูปการควบคุมป้อนกลับเพียงลูปเดียวเหมาะสำหรับอุปกรณ์สว่านไฟฟ้าเคลื่อนที่

ที่มา: http://www.rmutphysics.com/charud/scibook/electromagnetic1/magnetic/pic/problem/3-41.gif

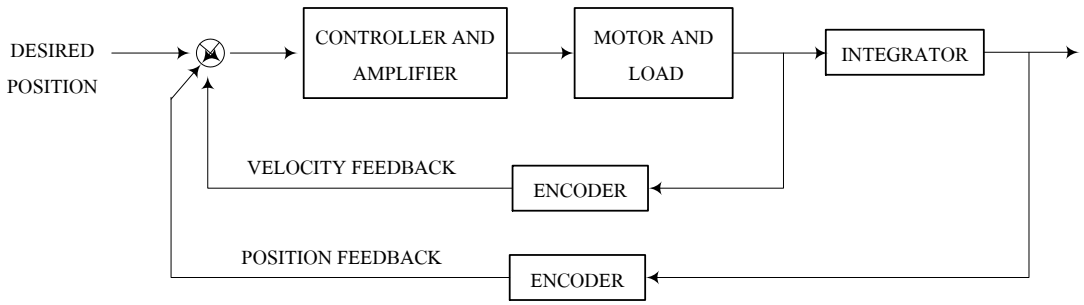
ในลูปป้อนกลับความเร็วของมอเตอร์ ที่ประกอบด้วยเครื่องวัดความเร็ว ความเร็วของเครื่องมือจะยังคงมีค่าคงที่ เนื่องจากเมื่อเครื่องมือที่ใช้ในการตัดมีความเร็วลดลง สัญญาณป้อนกลับจะ

ควบคุมมอเตอร์ให้เพิ่มความเร็วขึ้น ในขณะเดียวกัน เมื่อเครื่องมือตัดชิ้นงานที่เป็นวัสดุอ่อน ลูปป้อนกลับจะป้องกันไม่ให้มอเตอร์เร่งความเร็วเกินขนาด

อย่างไรก็ตาม ถ้าการประยุกต์ใช้งานมีความต้องการ วงจรควบคุมเพิ่มเติมสามารถถูกเพิ่มขึ้นมาเพื่อทำให้มอเตอร์มีความเร็วค่อย ๆ เพิ่มขึ้น และ ค่อย ๆ ลดลงจนกระทั่งหยุด กราฟของความเร็วเขียนได้เป็นรูปของสามเหลี่ยมที่ประกอบด้วยส่วนที่ลาดขึ้น และ ส่วนที่ลาดลง หรือ อาจเป็นรูปแทรปซอยด์ (Trapezoid) ซึ่งแบ่งเป็น 3 ส่วนคือ ส่วนเพิ่มความเร็วขึ้น (Ramp up) ส่วนความเร็วคงที่ในช่วงเวลาหนึ่ง และ ส่วนความเร็วลดลง

รูปที่ 2.13 คือ บล็อกไดอะแกรมของระบบควบคุมตำแหน่งนอกจากเครื่องวัดความเร็วในลูปป้อนกลับความเร็วแล้ว ระบบกลไกแบบเซอร์โวนี้จะมีลูปป้อนกลับการกำหนดตำแหน่ง

ตัวเซนเซอร์ในการตรวจจับตำแหน่ง และ ความเร็ว จะรู้ว่าเมื่อใดที่เพลาของมอเตอร์เซอร์โวอยู่ในตำแหน่งมุมที่ต้องการโดยการนับสัญญาณพัลส์ และ เปรียบเทียบพัลส์กับสัญญาณอินพุท ก่อนที่จะหยุดเพลาเมื่อนับได้เท่ากัน ซึ่งก็คือ บล็อกที่ชื่อตัวอินทิเกรท (Integrator) ในรูปที่ 5 โดยทั่วไป จะเป็นวงจรอิเลคทรอนิคส์ควบคุมตำแหน่งของเพลา ตัวเซนเซอร์ความเร็วในลูปป้อนกลับของระบบควบคุมตำแหน่ง ช่วยทำให้ระบบเกิดเสถียรภาพขึ้น



รูปที่ 2.13 ระบบควบคุมตำแหน่ง ประกอบด้วยลูปการควบคุมตำแหน่งป้อนกลับ และ ลูปควบคุมความเร็วป้อนกลับ

ที่มา: http://www.rmutphysics.com/charud/scibook/electromagnetic1/magnetic/pic/problem/3-41.gif

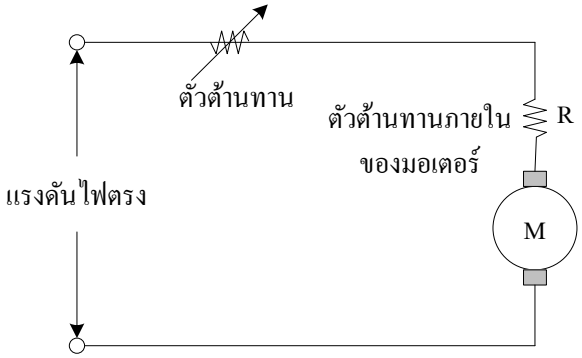
ในระบบควบคุมกำลังบิด (Torque control system) กำลังบิดของมอเตอร์เซอร์โวจะถูกรักษาให้มีค่าคงที่ เนื่องจากแรงบิดของมอเตอร์แปรตามกระแสของมอเตอร์ ดังนั้น กระแสที่ป้อนให้กับมอเตอร์ เพื่อรักษาค่าแรงบิดให้คงที่เอาไว้ วิธีนี้สามารถทำได้ด้วยวงจรที่ทำการเปรียบเทียบกระแสเอาท์พุทของมอเตอร์กับกระแสอินพุทของมอเตอร์ และ ขยายผลต่างเพื่อใช้เป็นวงจรควบคุมแรงบิดป้อนกลับ (Torque control feedback circuit)

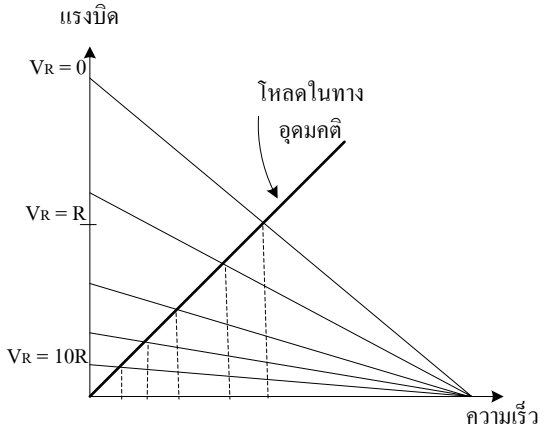
ระบบควบคุมการเคลื่อนที่แบบเพิ่ม (Incremental motion control system) ทำหน้าที่สับเปลี่ยนโหมดควบคุมจากโหมดหนึ่งไปเป็นอีกโหมดหนึ่ง เพื่อให้เกิดสมรรถนะการใช้งานที่ต้องการ ตัวอย่างเช่น การควบคุมความเร็วและตำแหน่ง ทำหน้าที่ควบคุมให้ได้ความเร็วที่ต้องการ แต่ก็สามารถปรับเปลี่ยนเป็นการควบคุมตำแหน่ง เพื่อหยุดเพลาให้ได้ถูกต้องแม่นยำขึ้น

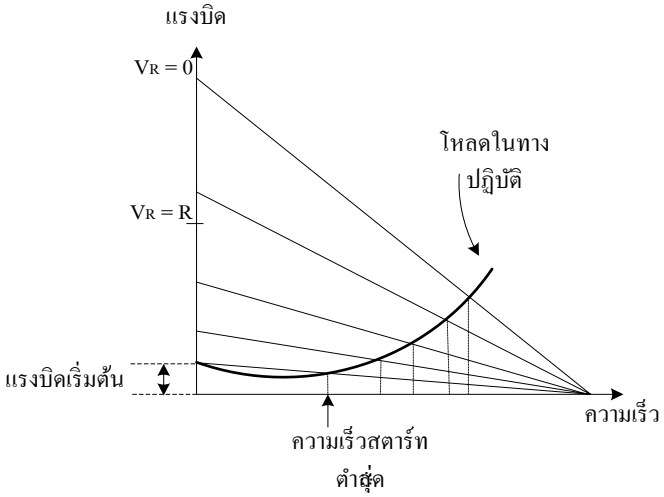
2.2.5 การควบคุมความเร็วของมอเตอร์ขั้นพื้นฐาน

2.2.5.1 การควบคุมด้วยตัวต้านทานที่ปรับค่าได้

เป็นรูปแบบพื้นฐานที่สุดของการควบคุมมอเตอร์คือ ใช้ตัวต้านทานปรับค่าได้อนุกรมกับมอเตอร์ โดยตัวต้านทานที่ปรับค่าได้จะเป็นตัวกำหนดความเร็วในการหมุนของมอเตอร์ การบังคับแบบนี้ไม่มีประสิทธิภาพเพราะกำลังไฟสูญเสียไปในตัวความต้านทาน มักนิยมใช้กับมอเตอร์ตัวเล็กๆ การบังคับแบบนี้ให้คุณสมบัติการสตาร์ทดี(ให้แรงบิดสูงที่ความเร็วต่ำ) แต่จะให้ความเร็วสูงมากเมื่อมอเตอร์อยู่ในภาวะที่มีโหลดน้อยๆ ดังนั้นการบังคับแบบนี้มีประโยชน์เฉพาะภาวะที่แรงต้านคงที่ เช่น การบังคับความเร็วของเครื่องจักรเย็บผ้า เป็นต้น





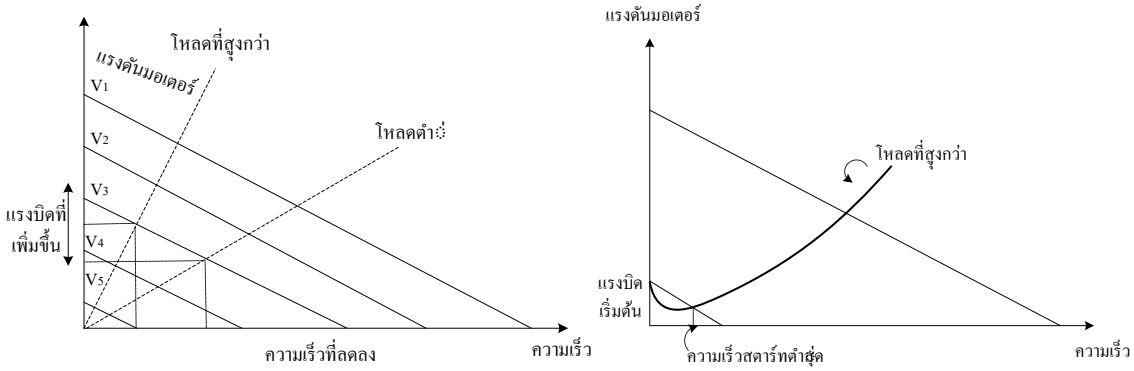


รูปที่ 2.14 วงจรควบคุมความเร็วของมอเตอร์กระแสตรงแบบใช้ตัวต้านทานอนุกรมและกราฟแสดงคุณสมบัติ

ที่มา: http://www.rmutphysics.com/charud/scibook/electromagnetic1/magnetic/pic/problem/3-41.gif

2.5.5.2 การควบคุมด้วยวิธีเปลี่ยนค่าแรงดัน

วิธีการนี้ดีกว่าวิธีการแรกแต่จะซับซ้อนกว่าต้องใช้อุปกรณ์อิเล็กทรอนิคส์ที่ทีอัตราขยายกำลังสูง และ มอเตอร์จะถูกป้อนด้วยแรงดันที่เปลี่ยนแปลงค่าได้ จากแหล่งจ่ายที่มีอิมพีแดนซ์ต่ำ ข้อดีของการควบคุมวิธีนี้คือ ถ้าความเร็วลดลงจากผลของแรงบิด แรงดันที่ป้อนให้กับมอเตอร์จะเพิ่มขึ้นเพื่อรักษาระดับความเร็ว ส่วนข้อเสียจากการควบคุมวิธีนี้คือ เมื่อมอเตอร์มีความเร็วต่ำแรงดันที่ป้อนให้กับมอเตอร์จะมีค่าต่ำเช่นกัน



รูปที่ 2.15 การควบคุมความเร็วโดยเปลี่ยนค่าแรงดัน

ที่มา: http://www.rmutphysics.com/charud/scibook/electromagnetic1/magnetic/pic/problem/3-41.gif

2.5.5.3 การควบคุมด้วยตัวต้านทานที่ปรับค่าได้

การควบคุมแบบนี้สามารถขับดีซีมอเตอร์ได้ความเร็ว 10 : 1 และให้การเร็คกูเลทที่ดีกว่ากระแสถูกปล่อยให้ฟิลด์คงที่ ผลของคุณสมบัติ ความเร็วและแรงบิดได้รับการปรับปรุงดีขึ้นกว่าการบังคับด้วยความต้านทานที่ปรับค่าได้ และให้การเร็คกูเลทความเร็วคงที่ได้ดีขึ้นตลอดช่วงความเร็วที่กว้างกว่า

2.5.5.4 การควบคุมแบบ PWM (Pulse Width Modulation)

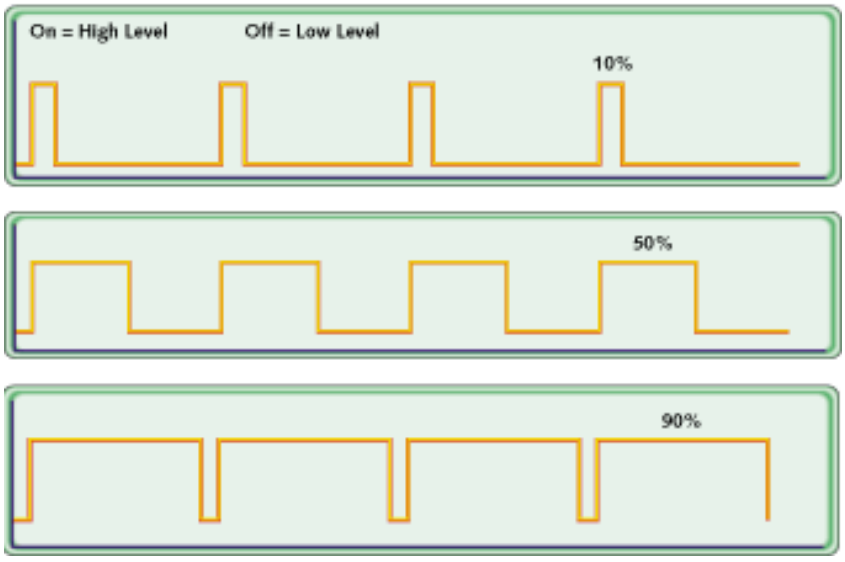
Pulse width modulation (PWM) คือ เทคนิคสำหรับควบคุมวงจรทางด้านฮาร์ดแวร์ โดยใช้สัญญาณเอาท์พุตแบบดิจิตอลของไมโครโปรเซสเซอร์ควบคุมการทำงานของสัญญาณ PWM ดังรูปที่ 2.16 แสดงสัญญาณ PWM ที่แตกต่างกัน 3 สัญญาณ

- โดย 10a แสดงสัญญาณ PWM ที่ 10% duty cycle คือ สัญญาณในการออนจะเป็น 10% ของคาบสัญญาณ และ จะออฟเป็น 90% ของคาบสัญญาณ

- โดย 10b แสดงสัญญาณ PWM ที่ 50% duty cycle คือ สัญญาณในการออนจะเป็น 10% ของคาบสัญญาณ และ จะออฟเป็น 50% ของคาบสัญญาณ

- โดย 10c แสดงสัญญาณ PWM ที่ 90% duty cycle คือ สัญญาณในการออนจะเป็น 10% ของคาบสัญญาณ และ จะออฟเป็น 10% ของคาบสัญญาณ

เช่น ถ้า Power Supply มี 9V และ duty cycle เป็น 10% จะได้เอาท์พุต 0.9V

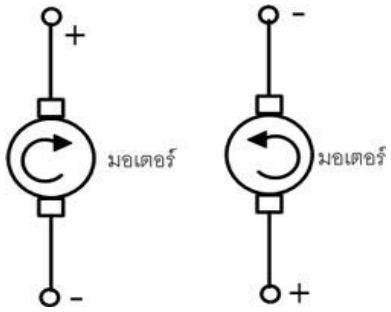


รูปที่ 2.16 แสดงสัญญาณ PWM ซึ่งแสดงค่า duty cycles ที่ต่างๆกัน

ที่มา: http://tutorial.cytron.com.my/wp-content/uploads/2012/01/PWM-signals-of-varying-duty-cycles_thumb1.gif

2.3 ชุดไดร์มอเตอร์

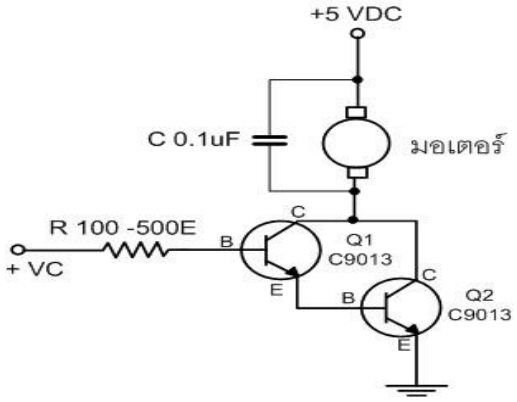
ในการกลับทางหมุนมอเตอร์นั้นจะสามารถทำได้โดยการสลับขั้วของแหล่งจ่ายเพียงเท่านั้น มอเตอร์ก็จะหมุนกลับทาง



รูปที่ 2.17 การกลับทางหมุนมอเตอร์

ที่มา: http://www.semi-shop.com/knowledge/103/103_01.jpg

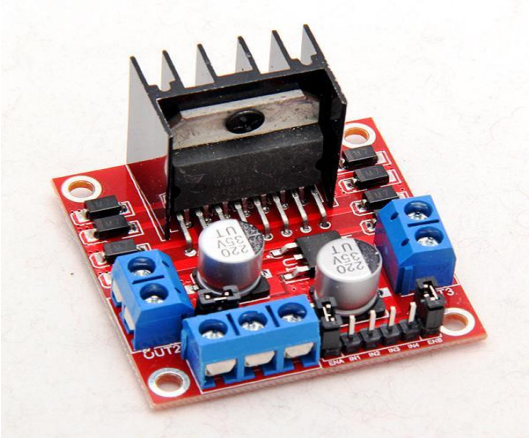
วงจรขับมอเตอร์พื้นฐาน เป็นวงจรที่ใช้ในการขับมอเตอร์นั้นมีหลายวงจรด้วยกัน ตั้งแต่วงจรพื้นฐานทั่วไป ไปถึงกลับทางหมุน และควบคุมความเร็วได้ โดยวงจรพื้นฐานนี้จะใช้ทรานซิสเตอร์เป็นตัวขับ



รูปที่ 2.18 วงจรขับมอเตอร์พื้นฐาน แบบดาลิงตัน

ที่มา: http://www.semi-shop.com/knowledge/102/102_06.jpg

วงจรขับกลับทางหมุนสำหรับมอเตอร์กระแสตรง ซึ่งขึ้นอยู่กับการจ่ายแรงดันขั้วบวกหรือลบ สำหรับในที่นี้เลือกใช้ L298N ต้องใช้แหล่งจ่าย DC ในช่วง 9 ถึง +12 V กระแสโหลดสูงสุดอยู่ที่ 2 แอมป์ สำหรับ L298N เป็นชุดขับมอเตอร์ชนิด H-Bridge ซึ่งส่วนใหญ่จะถูกนำไปใช้ในการควบคุมทิศทาง และความเร็วของมอเตอร์ ซึ่งสามารถควบคุมมอเตอร์ได้ทั้งหมด 2 Channel



รูปที่ 2.19 ชุดไดร์มอเตอร์ L298N

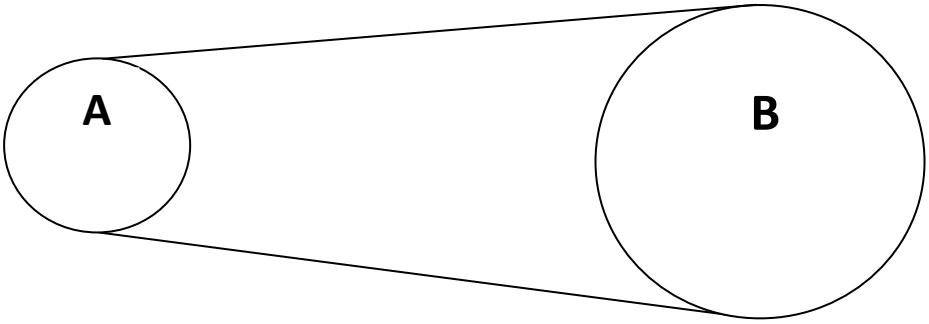
ที่มา: https://electrosome.com/wp-content/uploads/2014/10/L298N-Motor-Driver-Module.jpg

ในการต่อกับ Arduino นั้น ขา IN1,IN2,IN3 และ IN4 นั้น สามารถต่อกับพอร์ต Digital ใดๆก็ได้ เนื่องจาก 4 ขานี้ จะใช้ในการควบคุมสัญญาณลอจิกบอกทิศทางให้กับมอเตอร์ ส่วน ENA และ ENB นั้น จำเป็นที่จะต้องต่อกับพอร์ต Digital ที่รองรับ PWM เนื่องจากจะต้องใช้สัญญาณ PWM ในการควบคุมความเร็วของมอเตอร์

2.4 เฟืองและโซ่

เฟืองและโซ่มีความแข็งแรงและมีความแน่นอนในการควบคุมเพลาในการเคลื่อนที่ตามเส้นทางที่ต้องการหรือเมื่อจะเปลี่ยนทิศทางการเคลื่อนที่ ซึ่งระบบเฟืองและโซ่จะพบได้ในจักรยาน จักรยานยนต์ และรถยนต์ เพราะเฟืองจะถูกยึดกับโซ่ให้เคลื่อนที่ไปด้วยกัน โครงสร้างของเฟืองและโซ่ทำจากโลหะที่มีความแข็งแรงและในการผลิตเฟืองถ้าทำไม่ได้มาตรฐานก็จะทำให้โซ่ทำงานได้ไม่ดี

ในการเปลี่ยนความเร็วระหว่างเฟือง 2 ตัวจะขึ้นอยู่กับจำนวนของฟันเฟืองแต่ละตัว เมื่อเฟืองตัวที่ 1 มีกำลังเต็มที่ในการหมุนมันจะดึงโซ่โดยใช้ซี่ฟันเฟือง และโซ่ก็จะดึงเฟืองอีกตัวหนึ่งตามจำนวนซี่ฟันเฟือง ในตัวอย่าง จะเป็นการอธิบายกำลังของเฟือง เฟือง A มีฟันเฟือง 10 ซี่ ที่ต่อกับเฟือง B มีจำนวน 40 ซี่ เมื่อเฟือง A เคลื่อนที่ไป 10 ซี่ฟันเฟืองจะทำให้โซ่เคลื่อนที่ไป 10 ข้อและเฟือง B ที่มี 40 ซี่ฟันเฟืองจะเคลื่อนที่ไปได้ ¼ ของวงกลม ความเร็วจะลดลง 0.25 และกำลังขับจะเพิ่มเป็น 4 เท่า



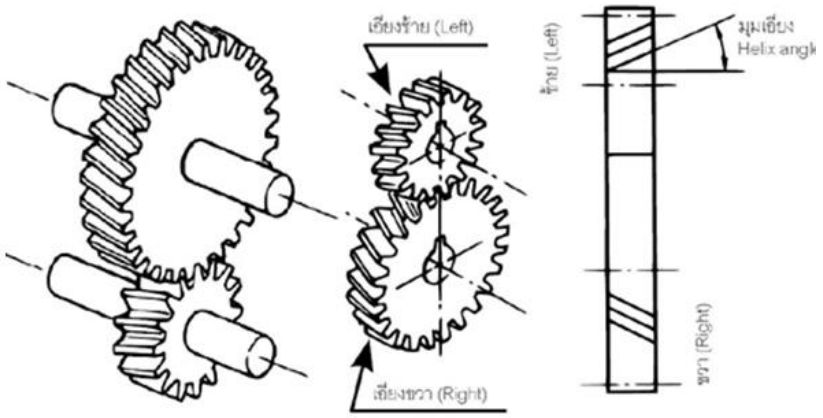
รูปที่ 2.20 เฟืองและโซ่

ที่มา: http://i01.i.aliimg.com/img/pb/002/081/043/1043081002_871.jpg

ปัญหาที่พบจากเฟืองและโซ่ ถ้าต้องการให้ความเร็วลดลง จะต้องมีโซ่พ่วงกลุ่มเฟืองเข้าด้วยกัน และจะทำให้เกิดความเร็วลดลงและไม่มีประสิทธิภาพ ไม่มีขั้นตอนการลดเพียงขั้นตอนเดียวในอัตราส่วน 5 ต่อ 1 ที่จะขัดขวางการทำงานของระบบ

2.5 เฟืองทด

เฟืองทดพบได้ในระบบขับเคลื่อนรถยนต์ นาฬิกาโบราณ และระบบการป้อนกระดาษของเครื่องพิมพ์ เฟืองทดมีการทำงานบนหลักการเดียวกับเฟืองและโซ่ แต่ไม่มีโซ่ เนื่องจากเฟืองทดต้องการความทนทานแทนที่จะใช้โซ่ขนาดใหญ่ถ่ายแรงที่ไม่เป็นเส้นตรง วิธีการยึดของเฟืองโดยตรงแสดงได้ดังรูป



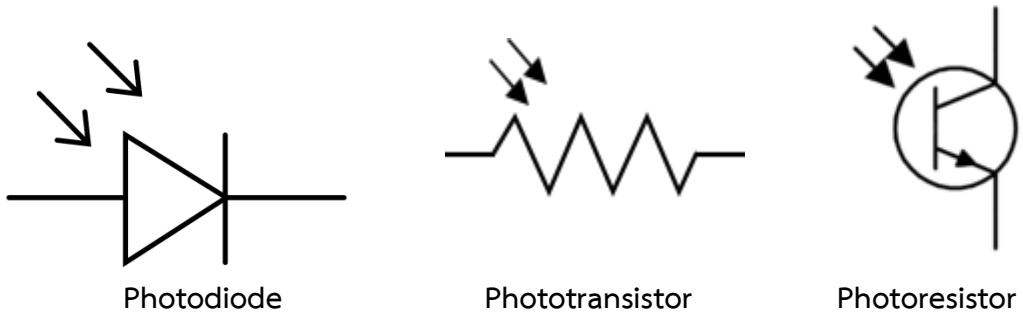
รูปที่ 2.21 เฟืองทด

ที่มา: https://sites.google.com/site/krrmwithikarphlitt/_/rsrc/1468857859658/neuxha-sara/bth-thi-7-kar-phlit-feuxng/12.png

นอกจากนี้ ถ้ามีร้านอุปกรณ์ที่น่าเชื่อถือ หรือชุดเลโก้ที่ดี ควรจะซื้อเฟืองทดสำเร็จรูป ความจริงแล้ว ชนิดอัตราส่วนที่ต้องการจากรายการที่พอดีกับเฟืองทดและพอดีกับแกนของมอเตอร์ซึ่งจะขายเป็นชุดโดยเรียกว่าเฟืองหัวมอเตอร์

2.6 การตรวจจับแสง

แสงเป็นสิ่งจำเป็นของละอองใน silligon และถูกนำไปใช้โดยตรงใน Photovoltaic (Solar Cell) ซึ่งจะมีผลที่ได้จะขึ้นอยู่กับการตอบสนองของแสง (เหมือนแสงจากแบตเตอรี่) หรือในทางอ้อมจะตรวจสอบเหมือน Photodiode หรือ Phototransistor แสงเปลี่ยนความต้านทานของโลหะใน Photoregistor จากความต้านทานสูง (100 K – 1 M) ในความมืดไปยังความต้านทานต่ำในแสงสว่าง เซนเซอร์แสงจะปรากฏในรูป



รูปที่ 2.22 เซนเซอร์แสง
ที่มา: http://www.technologystudent.com/images5/ldrmtr1.gif

2.6.1 เซนเซอร์ชนิดใช้แสง (Optical Sensor)

ในงานบางลักษณะที่ต้องการการตรวจจับชิ้นงานที่มีระยะห่างงจากตัวเซนเซอร์ค่อนข้างมาก ซึ่งอาจมีค่าเป็นเมตร ควรใช้เซนเซอร์ชนิดใช้แสงเป็นตัวตรวจจับ ส่วนเซนเซอร์แบบเหนี่ยวนำและชนิดเก็บประจุไม่สามารถใช้ได้ เพราะมีระยะการตรวจจับสูงสุดประมาณ 50 มิลลิเมตรหรืออย่างมากไม่เกิน 100 มิลลิเมตร หากต้องการระยะตรวจจับมากกว่านี้ก็จะทำให้มีขนาดที่ใหญ่ เพื่อทำให้สนามแม่เหล็กและสนามไฟฟ้ามีค่าสูงขึ้นซึ่งอาจจะมีผลกระทบกับระบบอื่นๆได้ นอกจากเซนเซอร์แบบใช้แสงจะมีคุณลักษณะเด่นในเรื่องของระยะการตรวจจับที่ไกลแล้ว ยังมีข้อดีอีกหลายอย่าง คือ สามารถตรวจจับวัตถุได้เกือบทุกประเภท ความเร็วในการตรวจจับสูง มีรุ่นที่สามารถแยกความแตกต่างของสีได้ นอกจากนี้ยังสามารถตรวจจับวัตถุในบริเวณที่เซนเซอร์แบบทั่วไปไม่สามารถใช้งานได้ เช่น บริเวณที่มีเนื้อที่ติดตั้งจำกัด บริเวณที่มีอุณหภูมิสูง ชิ้นงานที่มีขนาดเล็กๆได้

2.6.2 ชนิดของตัวรับแสงและส่งแสง

เซนเซอร์ชนิดใช้แสงประกอบด้วยส่วนที่สำคัญอยู่ 2 ส่วนด้วยกันคือตัวส่งแสงและตัวรับแสง ลักษณะของการตรวจจับนั้นเกิดขึ้นจากการที่ลำแสงตัวส่งส่งไปสะท้อนกับวัตถุ หรือถูกขวางกั้นด้วยวัตถุ และมีผลให้ตัวรับแสงรู้สภาวะที่เกิดขึ้น โดยจะเปลี่นแปลงสภาวะของสัญญาณเอาต์พุตเพื่อนำไปใช้ ตัวรับแสงจะใช้อุปกรณ์ที่เรียกว่า โฟโต้ไดโอด (Photo Diode)หรือ โฟโต้ทรานซิสเตอร์ (Photo Transistor) ทำหน้าที่เป็นตัวรับแสง และแปลงเปลี่ยนสัญญาณทางไฟฟ้า ซึ่งตัวกำเนิดแสง มีหลายประเภทด้วยกัน คือ

2.6.2.1 หลอดแบบมีไส้

เป็นเซนเซอร์รุ่นเก่าที่เคยใช้กัน มีข้อเสียตรงที่ขาดง่ายและมีขนาดค่อนข้างใหญ่ใช้พลังงานมาก ปัจจุบันก็ยังพอมีใช้อยู่ โดยใช้กับงานเฉพาะอย่าง

2.6.2.2 หลอด LED (Light Emitting Diode)

เป็นอุปกรณ์ที่มีขาดเล็ก มีความทนทานสูงนิยมใช้กันมากในเซนเซอร์รุ่นใหม่

2.6.2.3 LED แบบแสงอินฟราเรด (Infrared)

จะเป็นแสงอินฟราเรด ที่มีความยาวคลื่นอยู่ในช่วง 910 ถึง 950 นาโนเมตร มองด้วยตาเปล่าไม่เห็น ให้ความเข้มของแสงสูงจึงส่งไปได้เป็นระยะทางไกลและสามารถส่งทะลุวัตถุบางชนิดได้ แต่ไม่สามารถแยกแยะความแตกต่างของสีได้

2.6.2.4 LED แบบแสงสีแดง

เป็นแสงที่ตามองเห็น มีความยาวคลื่นประมาณ 660 นาโนเมตร ให้ความเข้มของแสงปานกลาง เซนเซอร์ที่ใช้แสงสีแดง จะมีระยะการตรวจจับใกล้ แต่สามารถตรวจจับเครื่องหมาย (mark) สีดำ น้ำเงิน หรือ เขียว บนพื้นสีขาวได้

2.6.2.5 LED แบบแสงสีเขียว

เป็นแสงที่ตามองเห็น มีความยาวคลื่นประมาณ 560 นาโนเมตร ให้ความเข้มของแสงต่ำ เซนเซอร์ใช้แสงสีเขียว จะมีระยะการตรวจจับใกล้ แต่สามารถตรวจจับเครื่องหมาย (mark) สีแดง บนพื้นสีขาวได้

2.6.2.6 LED แบบแสงเลเซอร์ (Light Amplification by Stimulated Emission of Radiation)

เป็นแสงที่ถูกขยายโดยการกระตุ้นให้แผ่รังสีออก โดยปกติแล้วแสงที่เราเห็นทั่วๆไป ไม่ว่าจะเป็นแสงจากหลอดไฟ แสงจากดวงอาทิตย์ จะเป็นแสงที่กระจายออกมารอบจุดกำเนิด มีหลายความถี่หรือหลายสี แต่เลเซอร์จะมีคุณสมบัติให้แสงสีเดียว มีสีเดียวและเฟสเดียว จึงทำให้แสงเลเซอร์เป็นแสงที่มีความเข้มสูง จะมีความถี่คลื่นอยู่ระหว่าง 0.01 มิลลิเมตร หรืออยู่ในความถี่ช่วงประมาณ 10^{13} ถึง 10^{15} เฮิรตซ์ ปกติทั่วไปแล้วจะเป็นลำแสงสีแดง

2.6.3 เทคนิคในการรับส่งแสง

2.6.3.1 วิธีการรับส่งแบบทั่วไป

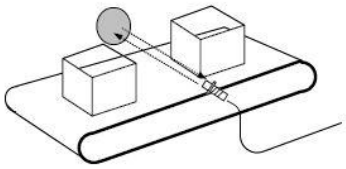
เป็นวิธีที่ตัวส่งแสงส่งลำแสงไปอย่างต่อเนื่องเป็นปกติ เหมือนกับแสงตามธรรมชาติ วิธีนี้ระยะการตรวจจับจะไม่ไกลนักและอาจจะถูกแสงจากภายนอกรบกวนได้ง่าย

2.6.3.2 วิธีการรับส่งแบบพัลซ์

เป็นวิธีที่ตัวส่งแสง จะส่งลำแสงเป็นจังหวะที่สม่ำเสมอด้วยอัตราความถี่ที่สูง และที่ส่วนของตัวรับก็จะถูกออกแบบมาสำหรับรับสัญญาณแสงนี้โดยเฉพาะ ด้วยวิธีนี้จะทำให้ระยะการตรวจจับทำได้ไกลและต้านทานต่อแสงรบกวนจากภายนอก

2.6.3.3 เซนเซอร์แบบลำแสงสะท้อน (Retro-Reflective Sensor)

เซนเซอร์ประเภทนี้จะรวมตัวส่งและตัวรับสัญญาณแสงไว้ในตัวเดียวกัน และใช้แผ่นสะท้อนแสง (reflector) สะท้อนแสงกลับ

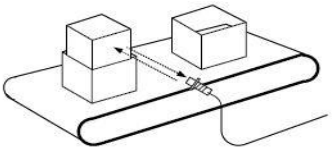


รูปที่ 2.23 ลักษณะเซนเซอร์แบบลำแสงสะท้อนกลับ

ที่มา: http://www.foodnetworksolution.com/uploaded/Measurement%20book/2.8.JPG

2.6.3.4 เซนเซอร์แบบตรวจจับโดยตรง (Diffuse-Reflective Sensor)

เซนเซอร์ประเภทนี้ ตัวส่งและตัวรับสัญญาณแสงจะอยู่ภายในตัวเดียวกัน แล้วใช้วัตถุหรือชิ้นงานที่เป็นตัวสะท้อนกลับ



รูปที่ 2.24 ลักษณะเซนเซอร์แบบตรวจจับโดยตรง

ที่มา: http://www.foodnetworksolution.com/uploaded/Measurement%20book/2.8.JPG

ดังที่ทราบเซนเซอร์แบบนี้ ระยะทาวที่ใช้ในการตรวจจับจะขึ้นอยู่กับตัวคุณสมบัติของตัวชิ้นงาน และตารางต่อไปนี้จะเป็นตัวอย่างของตัวคูณประกอบแฟลกเตอร์ของเซนเซอร์แบบตรวจจับโดยตรงในการตรวจจับชิ้นงานประเภทต่างๆ

2.7 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์(Microcontroller) มาจากคำ 2 คำ คำหนึ่งคือ ไมโคร(Micro) หมายถึง ขนาดเล็ก และคำว่า คอนโทรลเลอร์(controller) หมายถึงตัวควบคุมหรืออุปกรณ์ควบคุม ดังนั้น ไมโครคอนโทรลเลอร์ จึงหมายถึงอุปกรณ์ควบคุมขนาดเล็ก แต่ในตัวอุปกรณ์ควบคุมขนาดเล็กนี้ได้บรรจุความสามารถที่คล้ายคลึงกับระบบคอมพิวเตอร์ โดยมีโครงสร้างใกล้เคียงกับคอมพิวเตอร์ คือ ภายในประกอบด้วยหน่วยรับข้อมูลและโปรแกรม หน่วยประมวลผล หน่วยความจำ หน่วยแสดงผล ซึ่งส่วนประกอบเหล่านี้มีความสมบูรณ์ในตัวของมันเอง ทำให้มีขนาดเล็ก และสามารถเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ต่างๆ ที่เชื่อมต่อกับตัวมัน ง่ายต่อการนำไปประยุกต์ใช้งาน

Arduino เป็นโครงการพัฒนาไมโครคอนโทรเลอร์ของตระกูล AVR แบบ Open Source(ของฟรี) โดยนิยมใช้ AVR ในตระกูล ATMEGA88 ATMEGA168 ATMEGA328 โดยมีตัวถังแบบ DIP ทำให้ง่ายต่อการพัฒนา Arduino คือเครื่องมือที่จะทำให้คอมพิวเตอร์สามารถรับสัญญาณจากภายนอกและส่งสัญญาณไปควบคุมอุปกรณ์ภายนอกได้อย่างมีประสิทธิภาพมากกว่าใช้เครื่องพีซีตั้งโต๊ะตัวบอร์ดออกแบบจากไมโครคอมพิวเตอร์ชิพเดี่ยว และมีโปรแกรมพัฒนาสำหรับเขียนโปรแกรมให้บอร์ดทำงาน Arduino สามารถประยุกต์ทำเครื่องใช้อัจฉริยะ รับสัญญาณจากสวิทซ์ หรือ เซนเซอร์และควบคุม หลอดไฟมอเตอร์ หรืออุปกรณ์อื่นๆ โปรเจค Arduino เป็นได้ทั้งแบบทำงานอิสระ หรือทำงานติดต่อกับโปรแกรมที่ทำงานบนเครื่องพีซี ตัวบอร์ดสามารถประกอบขึ้นใช้เอง หรือจะซื้อสำเร็จที่มีขาย ส่วนโปรแกรมพัฒนา Arduino สามารถดาวน์โหลดได้ฟรี

- ราคา Arduino บอร์ดไม่แพงเมื่อเทียบดับบอร์ดอื่น บอร์ด Arduino ที่ราถูกสุดสามารถทำใช้เองได้หรือซื้อสำเร็จด้วยเงินไม่เกิน 30 $

- ทำงานได้หลายเพลตฟอร์ม โปรแกรมพัฒนา Arduino ทำงานได้ทั้งบนวินโดวส์ Macintosh OSX และบนลินุกซ์ ในขณะที่บอร์ดอื่นทำงานได้เฉพาะวินโดวส์

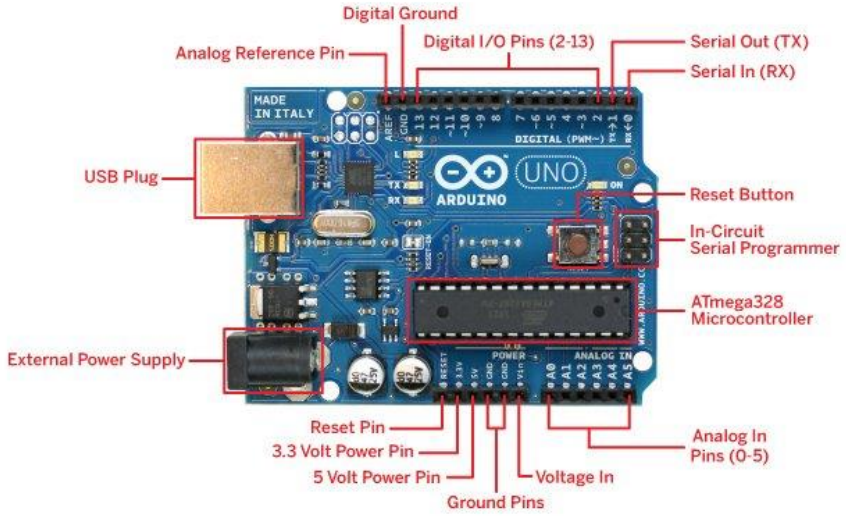
- ใช้งานง่าย มีโปรแกรมพัฒนาที่ไม่ซับซ้อน โปรแกรม Arduino ใช้งานง่ายสำหรับมือใหม่ และมีความสามารถครบตามความต้องการของนักพัฒนามืออาชีพ

- เปิดเผยซอร์สโค้ดและนำไปพัฒนาต่อยอดได้ โปรแกรม Arduino ตีพิมพ์แบบเปิดเผยซอร์สโค้ดและสามารถเพิ่มเติมความสามารถผ่าน C++ library ถ้าคุณต้องการศึกษาให้ลึกซึ้ง คุณสามารถข้ามไปเล่น AVR C ซึ่งเป็นต้นฉบับของ Arduino, และความสามารถเพิ่มเติม AVR C โค้ดได้โดยตรงถ้าคุณต้องการ

- เปิดเผยวงจร และนำไปพัฒนาขยาย hardware ได้ Arduino ใช้ไมโครคอนโทรลเลอร์ของ Atmel เบอร์ ATMEGA8 และ ATMEGA168 วงจรขับของบอร์ดตีพิมพ์แบบเปิดเผยวงจรภายใต้ Creative Commons License คุณสามารถนำไปดัดแปลงต่อขยายและเพิ่มประสิทธิภาพ เพื่อศึกษาการทำงานของมันได้ฟรี

2.7.1 คุณสมบัติของบอร์ดโดยทั่วไป

- Atmega ตระกูล mx8 (168 หรือ 328)รับที่ความเร็ว 16.00 MHz
- บนบอร์ดจะมีสอง LED ใช้แสดง power และ ที่ pin 13 สำหรับทดสอบ และอีกสอง LED ที่แสดงผล TX RX
- Pin บนบอร์ดเป็นแบบ มาตรฐานตามบอร์ดของ Arduino ประกอบไปด้วย
- Digital 0 thru 13
- Analog 0 thru 5
- A Ref, 5V Ground Vin and Reset
- 6-pin standard ICSP header
- Autu-reset capabitity



รูปที่ 2.25 Arduino board
ที่มา: https://j.lnwfile.com/z0wxio.jpg

Arduino เป็น platform ของ I/O บอร์ดอย่างง่ายๆที่มี I/O ขั้นพื้นฐานที่พอเพียงกับการใช้งานและการเรียนรู้โดยตัวบอร์ดจะมาพร้อมกับชุดคำสั่งที่ใช้ควบคุม port I/O ไม่ว่าจะเป็น port digital port analog PWM และ Serial port ซึ่ง Arduino นั้นเป็นเครื่องมือที่จะทำให้คอมพิวเตอร์สามารถรับสัญญาณจากภายนอกและส่งสัญญาณไปควบคุมอุปกรณ์ภายนอกได้อย่างมีประสิทธิภาพมากกว่าใช้เครื่องพีซีตั้งโต๊ะตัวบอร์ดออกแบบจากไมโครคอมพิวเตอร์ชิพเดี่ยวและมีโปรแกรมพัฒนาสำหรับเขียนโปรแกรมให้บอร์ดทำงาน Arduino สามารถประยุกต์ทำเครื่องใช้อัจฉริยะรับสัญญาณจากสวิทซ์หรือเซนเซอร์ และควบคุมหลอดไฟ มอเตอร์ หรือ อุปกรณ์อื่นๆโปรเจค Arduino เป็นได้ทั้งแบบทำงานอิสระ

หรือทำงานติดต่อกับโปรแกรมที่ทำงานบนเครื่องพีซีตัวบอร์ดสามารถประกอบขึ้นใช้เองหรือจะซื้อสำเร็จที่มีขายส่วนโปรแกรมพัฒนา Arduino สามารถดาวน์โหลดได้ฟรี

Arduino เป็นภาษาอิตาลีซึ่งใช้เป็นชื่อของโครงการพัฒนาไมโครคอนโทรลเลอร์ตระกูล AVR แบบ Open Source ที่ได้รับการปรับปรุงมาจากโครงการพัฒนา Open Source ของ AVR อีกโครงการหนึ่งที่ชื่อว่า “Wiring” แต่เรื่องจากโครงการของ “Wiring” เลือกใช้ AVR เบอร์ ATmega128 ซึ่งเป็นไมโครคอนโทรลเลอร์ที่มีจำนวนของหน่วยความจำและ I/O ค่อนข้างมากและที่สำคัญ Atmega128 เป็นชิพที่มีตัวถังแบบ SMD จึงทำให้เป็นอุปสรรคสำหรับผู้เริ่มต้นในการสร้างบอร์ดและต่อวงจรขึ้นมาใช้งานกันเองและบอร์ดจะมีขนาดค่อนข้างใหญ่ซึ่งอาจดูว่าเกินความจำเป็นสำหรับผู้เริ่มต้นจึงไม่ค่อยได้รับความนิยมเท่าที่ควรแต่หลังจากที่ทีมงาน Arduino พัฒนา Source Code ของ “Wiring” มาพัฒนาปรับปรุงใหม่โดยให้สามารถใช้งานกับไมโครคอนโทรลเลอร์ขนาดเล็กเช่น Mega8 Mega168 Mega328 ได้จึงทำให้ระบบวงจรของบอร์ดมีขนาดเล็กลงกว่า “Wiring” มากและยังใช้อุปกรณ์น้อยชิ้นทำให้ง่ายต่อการต่อวงจรใช้งานกันเองและยังประหยัดต้นทุนในการสร้างบอร์ดไปได้มากด้วยเหตุนี้เองที่ทำให้ “Arduino” ได้รับความนิยมจากผู้ใช้งานทั่วโลกเป็นอย่างมากในระยะเวลาอันรวดเร็ว

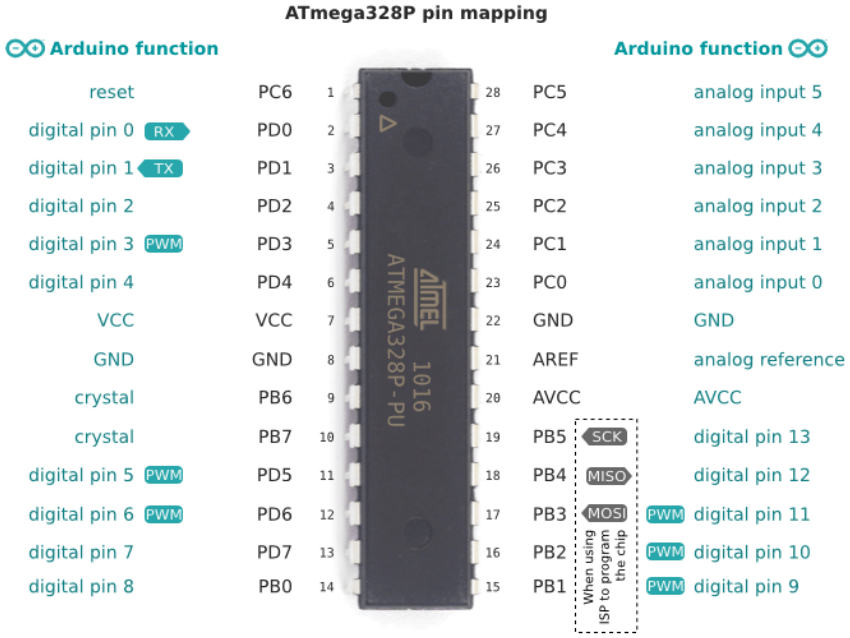
Arduino มีจุดเด่นในเรื่องของความง่ายต่อการเรียนรู้และใช้งานเนื่องจากมีการออกแบบคำสั่งต่างๆขึ้นมาสนับสนุนการใช้งานด้วยรูปแบบที่ง่ายไม่ซับซ้อนในตลาดไมโครคอนโทรลเลอร์มีตัวเลือกมากมายเช่น Parallaxe Basic Stamp Netmedia’s BX-24 Pidgets MIT’s Handyboard และอีกหลายเจ้าที่มีคุณสมบัติใกล้เคียงกันคือทำโปรเจคให้ใช้งานง่ายและเน้นการใช้งานโปรแกรมไมโครคอรโทรลเลอร์เป็นหลัก

Arduino เป็นบอร์ดไมโครคอนโทรลเลอร์โดยใช้ AVR ขนาดเล็กเป็นตัวประมวลผลและสั่งงานเหมาะสำหรับนำไปใช้ในการศึกษาเรียนรู้ระบบไมโครคอนโทรลเลอร์และนำไปประยุกต์ใช้งานเกี่ยวกับการควบคุมอุปกรณ์ Input/Output ต่างๆได้มากมายทั้งในแบบที่เป็นการทำงานตัวเดียวอิสระหรือเชื่อมต่อสั่งงานร่วมกับอุปกรณ์อื่นๆเช่นคอมพิวเตอร์ PC ทั้งนี้ก็เนื่องมาจากว่า Arduino สนับสนุนการเชื่อมต่อกับอุปกรณ์ Input/Output ต่างๆได้มากมายทั้งแบบ Digital และ Analog เช่น การรับค่าจากสวิตซ์หรืออุปกรณ์ตรวจจับ (Sensor) แบบต่างๆรวมไปถึงการควบคุมอุปกรณ์ Output ต่างๆตั้งแต่ LED หลอดไฟ มอเตอร์ รีเลย์ โดยระบบฮาร์ดแวร์ของ Arduino สามารถสร้างและประกอบขึ้นใช้งานได้เองในกรณีที่ผู้ใช้พอมีความรู้ด้านอิเล็กทรอนิกส์อยู่บ้างหรือสามารถซื้อแผงวงจรสำเร็จรูปที่มีการผลิตออกจำหน่ายกันในราคาที่ไม่แพงอีกทั้งยังเผยแพร่ Source Code และตัวอย่างต่างๆให้ผู้ใช้นำไปใช้งานหรือพัฒนาดัดแปลงต่อยอดได้โดยไม่เสียค่าใช้จ่ายใดๆ

ส่วนภาษาในการเขียนโปรแกรมลงบน Arduino นั้นจะใช้ภาษา C++ ซึ่งเป็นรูปแบบของโปรแกรมภาษาซีประยุกต์แบบหนึ่งที่มีโครงสร้างของตัวภาษาโดยรวมใกล้เคียงกันกับภาษาซีมาตรฐาน (ANSI-C) อื่นๆเพียงแต่ได้มีการปรับปรุงรูปแบบในการเขียนโปรแกรมบางส่วนที่ผิดเพี้ยนไปจาก ANSI-C

เล็กน้อยเพื่อช่วยลดความยุ่งยากในการเขียนโปรแกรมและให้ผู้เขียนโปรแกรมสามารถเขียนโปรแกรมได้ง่ายและสะดวกมากขึ้นกว่าการเขียนภาษาซีตามแบบมาตรฐานของ ANSI-C โดยตรง

ซึ่งจากการที่ได้ทำการศึกษาค้นคว้าทดลองการใช้งานภาซีของ Arduino มาในระยะเวลาหนึ่งพบว่าในความจริงแล้ว Arduino นั้นไม่ใช่ C-Compiler โดยตรงแต่ Arduino จะมีลักษระการทำงานเช่นเดียวกับ Text Editor เป็นฉากหน้าในการติดต่อสื่อสารกับผู้ใช้เท่านั้นส่วนเบื้องหลังจริงๆนั้น Arduino จะไปเรียกใช้ตัวแปลภาษาซีและ Utility อื่นๆที่ใช้เป็นเครื่องมือพัฒนาโปรแกรมของไมโครคอนโทรลเลอร์ตระกูล AVR อีกทีหนึ่งโดย Arduino จะเลือกใช้ C-Compiler ของ “GNU AVR-GCC Toolchain” ร่วมกับ Library Function ของ “avr-libc” ส่วน Utility ที่ใช้ในการ Upload Code ให้กับ AVR นั้นก็จะใช้ของ “AVRDude” ดังนั้นผู้ที่เขียนภาษาซีของ AVR เป็นอยู่แล้วและต้องการประยุกต์ใช้งาน Arduino ให้ได้ประสิทธิภาพการทำงานมากยิ่งขึ้นไปอีกก็สามารถศึกษาข้อกำหนดและหน้าที่ในการใช้งาน Libraly และคำสั่งอื่นๆที่บรรจุไว้ใน Libraly ต่างๆทั้งจากของ “GNU AVR-GCC Toolchain” และ “avr-libc” เพิ่มเติมอีกเพื่อใช้เป็นแนวทางในการปรับปรุงและประยุกต์ใช้งาน Arduino ในรูปแบบที่สลับซับซ้อนมากๆขึ้นไปอีก Chip และ IC ภายใน Arduino Board ที่สำคัญ

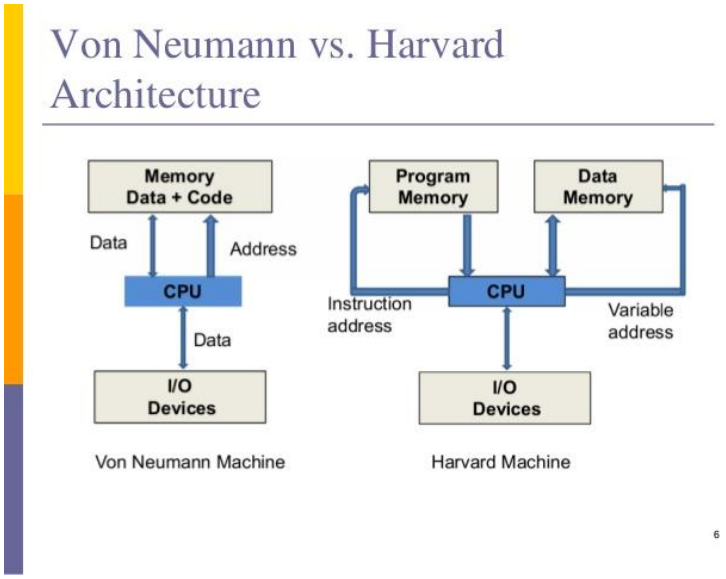


รูปที่ 2.26 ซีพียู ATMEGA328P ขนาด 28 ขา

ที่มา: http://www.openfog.net/images/ATmega328P_A.png

ตัวบอร์ด Arduino ที่ใช้ในโปรเจคนี้จะกล่าวถึงสถาปัตยกรรมของ AVR ขนาด 8 บิตโดยในสถาปัตยกรรม AVR ซึ่งออกแบบโดย ATMEL เมื่อปี 1996 เป็นซีพียูแบบ RISC (Reduced Instruction Set Computer) มีสถาปัตยกรรมการต่อหน่วยความจำแบบ Harvard ซึ่งแยกหน่วยความจำโปรแกรม

และหน่วยความจำข้อมูลออกจากกันโดยเด็ดขาดดังแสดงในรูปที่ 2.25 โดยใช้หน่วยความจำแบบแฟลชสำหรับเป็นหน่วยความจำโปรแกรมและใช้หน่วยความจำแบบ SRAm สำหรับหน่วยความจำข้อมูลและนอกจากนี้ยังมีหน่วยความจำแบบ EEPROM ซึ่งสามารถเก็บข้อมูลเอาไว้ได้โดยไม่จำเป็นต้องมีไฟเลี้ยงอีกด้วย



รูปที่ 2.27 เปรียบเทียบการจัดการหน่วยความจำของสถาปัตยกรรมแบบ Von-Neumann และ Harvard
ที่มา: http://spiroprojects.com/webadmin/uploads/von.jpg

จากรูปที่ 2.26 จะเห็นว่าโพรเซสเซอร์ที่ใช้สถาปัตยกรรมแบบ Harvard จะแยกหน่วยความจำสำหรับเก็บข้อมูลออกจากโปรแกรมอย่างชัดเจนสถาปัตยกรรม AVR และ MCS-51 จะใช้รูปแปบบนี้ในการจัดการหน่วยความจำส่วนสถาปัตยกรรมแบบ Von-neumann การตัดสินใจว่าจะเก็บโปรแกรมหรือข้อมูลจะแบ่งเก็บอย่างไรจะทำได้อย่างอิสระโดยขึ้นอยู่กับโปรแกรมเมอร์หรืออาจจะเป็นระบบปฏิบัติการเป็นผู้ดำเนินการให้ลักษณะเด่นของสถาปัตยกรรม AVR คือคำสั่งส่วนใหญ่สามารถทำงานได้เสร็จภายใน 1 clock cycle ตัวซีพียู AVR ขนาด 8 บิตจะแบ่งออกเป็นประเภทการใช้งานได้ 5 กลุ่ม ได้แก่

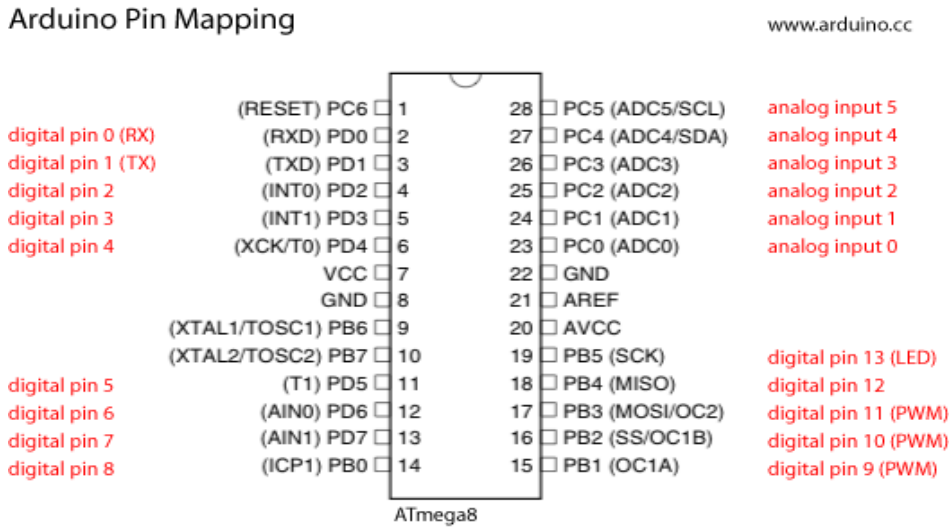
- Tiny AVR เป็นซีพียูในรุ่นเล็กซึ่งต้องการความเล็กกะทัดรัดของวงจรโดยเหมาะสมกับระบบควบคุมขนาดเล็กๆที่ต้องการหน่วยความจำและวงจรสนับสนุนไม่มากนักซีพียูในรุ่นนี้จะมีราคาถูกกว่ากลุ่มอื่น

- Mega AVR จะมีชื่ออีกอย่างว่า ATmega โดยมีวงจรสนับสนุนภายในเพิ่มเติมตลอดจนเพิ่มขนาดของหน่วยความจำให้ใช้งานมากกว่าตระกูล Tiny เหมาะกับงานควบคุมทั่วๆไป

- XMEGA เพิ่มความละเอียดของวงจร A/D จากปกติมีความละเอียด 10 บิตในรุ่นเล็กกว่าเป็น 12 บิตและวงจร DMA controller ซึ่งช่วยลดภาระของซีพียูในการควบคุมการรับส่งข้อมูลระหว่างอุปกรณ์ I/O กับหน่วยความจำ

- FPSLIC (AVR core with FPGA) สำหรับงานที่ต้องการควบคุมที่ต้องการความยืดหยุ่นในขั้นตอนการออกแบบและพัฒนาโดยผู้ออกแบบสามารถออกแบบวงจรในระดับฮาร์ดแวร์เพิ่มเติมด้วยภาษาบรรยายฮาร์ดแวร์ (HDL: Hardware Description Language) เช่นภาษา VHDL หรือภาษา Verilog และให้วงจรที่ออกแบบทำงานร่วมกับซีพียู AVR core

- Application Specific AVR เป็นซีพียูที่ออกแบบมาโดยเพิ่มวงจรควบคุมเฉพาะด้านเข้าไปซึ่งไม่พบในซีพียูกลุ่มอื่นๆเช่นวงจร USB controller หรือวงจร CAN bus เป็นต้น



รูปที่ 2.28 แสดงขา ของ Arduino Board

ที่มา: https://www.arduino.cc/en/uploads/Hacking/Arduino-To-Atmega8-Pins.png

ซีพียู AVR มีให้เลือกใช้งานหลายเบอร์แต่ละเบอร์จะมีขนาดราคาความสามารถและขนาดหน่วยความจำตลอดจนถึงวงจรสนับสนุนภายในที่แตกต่างกันออกไปในโครงงานนี้จะเลือกให้ซีพียูรุ่น ATmega328P ซึ่งมีคุณสมบัติดังนี้

- หน่วยความจำโปรแกรมแบบ FLASH ขนาด 32 กิโลไบต์
- หน่วยความจำข้อมูลแบบ SRAM ขนาด 2 กิโลไบต์
- หน่วยความจำข้อมูลแบบ EEPROM ขนาด 1 กิโลไบต์

- สนับสนุนการเชื่อมต่อบน L2C bus
- พอร์ตอินพุตเอาต์พุตจำนวน 23 บิต
- วงจรสื่อสารอนุกรม
- วงจรนับ/จับเวลาขนาด 8 บิต จำนวน 2 ตัว
- ความถี่ใช้งานสูงสุด 20 MHz

2.8 เทคโนโลยีอาร์เอฟไอดี (RFID Technology)

อาร์เอฟไอดี (RFID) นั้นย่อมาจากคำว่า Radio Frequency Identification หรือระบบชี้เฉพาะอัตโนมัติ (Automatic Identification) แบบไร้สาย (Wireless) เป็นระบบบุกเอกลักษณ์ของวัสดุด้วยคลื่นความถีวิทยุ โดยจุดเด่นของระบบอาร์เอฟไอดีจะอยู่ที่การอ่านข้อมูลแท็ก (Tag) ได้หลายๆแท็กแบบไร้สัมผัส (Contactless) และสามารถที่จะอ่านค่าได้แม้ในสภาพที่ทัศนวิสัยไม่ดี ทนต่อความเปียกชื้น แรงสั่นสะเทือน การกระทบกระแทก และสามารถอ่านค่าได้ด้วยความเร็วสูง โดยข้อมูลจะถูกเก็บไว้ในไมโครชิป (Microchip) ที่อยู่ในแท็ก ในปัจจุบันได้มีการนำเทคโนโลยีอาร์เอฟไอดีไปประยุกต์ใช้งานด้านอื่นๆ นอกจากการนำมาใช้ในระบบบาร์โค้ด (Barcode) แบบเดิม เช่น การใช้งานในบัตรชนิดต่างๆ อย่างเช่น บัตรสำหรับเข้าออกตามหอพัก เป็นต้น หรืออาจจะเป็นแคปซูล (Capsule) ขนาดเล็กฝังอยู่ในตัวสัตว์เพื่อบันทึกประวัติต่างๆเป็นต้น

ระบบอาร์เอฟไอดีจะมีองค์ประกอบหลักอยู่ 2 ส่วน โดยส่วนแรกคือทรานสปอนเดอร์หรือแท็ก (Transponder / Tag) ที่ใช้ติดกับวัตถุต่างๆที่ต้องการ โดยแท็กจะทำหน้าที่บันทึกข้อมูลเกี่ยวกับวัสดุชิ้นนั้นๆไว้ และส่วนสองคือ เครื่องสำหรับอ่านหรือเขียนข้อมูลภายในแท็กด้วยคลื่นความถี่วิทยุ (Interrogator / Reader) โดยการทำงานของเครื่องอ่านจะทำหน้าที่จ่ายกำลังงานในรูปคลื่นความถี่วิทยุให้กับตัวบัตร ยังผลให้วงจรอิเล็กทรอนิกส์ (Electronics) ภายในสามารถส่งข้อมูลจำเพาะที่แสดงถึงเอกลักษณ์ (Identity) ของแต่ละบัตรกลับมาประมวลผลตัวอ่านได้

2.8.1 แท็ก (Tag)

แท็ก มาจากคำว่าทรานสมิตเตอร์ (Transmitter) ผสมกับคำว่าเรสปอนเดอร์ (Responder) โดยโครสร้างภายในของแท็กจะประกอบด้วย 2 ส่วนใหญ่ ได้แก่ ขดลวดเล็กซึ่งทำหน้าที่เป็นสายอากาศ (Antenna) และไมโครชิปซึ่งขดลวดขนาดเล็กที่ทำหน้าที่เป็นสายอากาศนั้นจะใช้สำหรับส่งสัญญาณคลื่นความถี่วิทยุและสร้างพลังงานป้อนให้กับส่วนของไมโครชิป ที่ทำหน้าที่ในการเก็บข้อมูลต่างๆของวัตถุนั้นๆ โดยทั่วไปตัวแท็กอาจจะอยู่ในรูปแบบที่เป็นได้ทั้ง กระดาษ แผ่นฟิลม์ พลาสติก ที่มีขนาดและรูปร่างแตกต่างกันออกไป ทั้งนี้ขึ้นอยู่กับวัสดุที่จะนำมาแท็กไปติดไว้และมีได้หลายรูปแบบ เช่น ขนาดเท่าบัตรเครดิต เหรียญ กระดุม ฉลากสินค้า แคปซูล เป็นต้น

2.8.1.1 ประเภทของแท็ก

โดยหลักการนั้นสามารถแบ่งประเภทของแท็กที่มีการใช้งานออกเป็น 2 ชนิดใหญ่ๆ โดยแต่ละชนิดจะมีความแตกต่างกันในแง่ของการใช้งาน ราคา โครงสร้างและหลักการทำงาน ซึ่งสามารถแบ่งแยกออกได้ดังนี้

- แท็กชนิดของแอคทีฟ (active Tag) แท็กชนิดนี้จะต้องอาศัยแหล่งไฟจากแบตเตอรี่ (Battery) ภายนอกเพื่อจ่ายพลังงานให้วงจรภายในนั้นทำงาน เราจะสามารถทั้งอ่านและเขียนข้อมูลลงในแท็กชนิดนี้ได้ และการที่ต้องใช้แบตเตอรี่จึงทำให้แท็กชนิดนี้มีอายุการใช้งานจำกัดตามอายุของแบตเตอรี่ เมื่อแบตเตอรี่หมดก็ต้องนำแท็กไปทิ้ง ไม่สามารถนำกลับมาใช้งานใหม่ได้ แท็กชนิดนี้สามารถมีหน่วยความจำภายในขนาดใหญ่ได้ถึง 1 เมกกะไบต์ (Megabytes) และสามารถอ่านข้อมูลได้ระยะไกลสุดประมาณ 10 เมตร ซึ่งไกลกว่าชนิดพาสซีฟ (passive) อีกทั้งยังมีกำลังส่งสูงและยังสามารถทำงานในบริเวณที่มีสัญญาณรบกวนได้อีกด้วย

- ชนิดพาสซิฟ (Passive Tag) แท็กชนิดนี้ไม่จำเป็นต้องอาศัยแหล่งจ่ายไฟจากภายนอกใดๆ เพราะภายในแท็กจะมีวงจรกำเนิดไฟฟ้าเหนี่ยวนำขนาดเล็กเป็นแหล่งจ่ายไฟในตัว ทำให้การอ่านข้อมูลนั้นทำได้ไม่ไกลมากนัก ระยะอ่านสูงสุดประมาณ 1 เมตร โดยจะขึ้นอยู่กับความแรงของเครื่องส่งและคลื่นความถี่วิทยุที่ใช้ โดยปกติแล้วแท็กชนิดนี้มักมีหน่วยความจำขนาดเล็ก โดยทั่วไปประมาณ 16 ถึง 1024 ไบต์ (bytes) ราคาต่ำกว่าและมีอายุการใช้งานไม่จำกัด แต่ข้อเสียก็คือ ตัวอ่านข้อมูลจะต้องมีความไวสูง และมีปัญหาเมื่อนำมาใช้งานในสิ่งแวดล้อมที่มีสัญญาณแม่เหล็กไฟฟ้าลบกวนอีกด้วย

2.8.2 เครื่องอ่านหรือเขียน (Reader)

หน้าที่ของเครื่องอ่านก็คือการเชื่อมต่อหรือเขียนข้อมูลลงในแท็กด้วยสัญญาณความถี่วิทยุ นอกจากนี้ตัวอ่านข้อมูลที่ดีต้องมีความสามารถในการป้องกันการอ่านข้อมูลซ้ำเช่น ในกรณีที่แท็กถูกวางทิ้งอยู่ในบริเวณสนามแม่เหล็กไฟฟ้าที่ตัวอ่านข้อมูลสร้างขึ้นหรืออยู่ในระยะการรับส่งข้อมูล ก็อาจทำให้ตัวอ่านข้อมูลทำการรับหรืออ่านข้อมูลจากแท็กซ้ำอยู่เรื่อยๆไม่สิ้นสุด ดังนั้นตัวอ่านข้อมูลที่ดีต้องมีระบบป้องกันเหตุการณ์เช่นนี้ ที่เรียกว่าระบบ “แฮนด์ดาวน์โพลิ่ง (Hands Down Polling)” โดยตัวอ่านข้อมูลจะสั่งให้แท็กหยุดการส่งข้อมูลในกรณีที่เกิดเหตุการณ์ดังกล่าว หรืออาจมีบางกรณีที่มีแท็กหลายๆแท็กอยู่ในบริเวณสนามแม่เหล็กไฟฟ้าพร้อมกัน หรือที่เรียกว่า “แบชลีดดิ่ง (Batch Reading)” ตัวอ่านข้อมูลควรมีความสามารถที่จะจัดลำดับการอ่านแท็กทีละตัวได้ ซึ่งการชี้เฉพาะระบุตัวแท็กนั้นเป็นระบบอัตโนมัติ (Automatic Identification) โดยภายในเครื่องอ่านจะประกอบด้วยส่วนประกอบหลักดังนี้

- และส่งสัญญาณวิทยุ (Transceiver)
- ภาคสร้างสัญญาณพาหะ (Carrier)
- ขดลวดที่ทำหน้าที่เป็นสายอากาศ
- วงจรจูนสัญญาณ (Tuner)
- หน่วยประมวลผลข้อมูล (Processing Unit)

2.8.3 หลักการเบื้องต้นของอาร์เอฟไอดี

2.8.3.1 เครื่องอ่านอาร์เอฟไอดี จะปล่อยคลื่นแม่เหล็กไฟฟ้าออกมาตลอดเวลาและคอยตรวจจับว่ามีแท็กในบริเวณสนามแม่เหล็กไฟฟ้าหรือไม่ หรืออีกนัยหนึ่งก็คือคอยตรวจจับว่ามีการมอดูเลต (Modulation) สัญญาณเกิดขึ้นหรือไม่

2.8.3.2 เมื่อมีแท็กเข้ามาอยู่ในระยะบริเวณสนามแม่เหล็กไฟฟ้า แท็กจะได้รับพลังงานไฟฟ้าที่เกิดจากการเหนี่ยวนำของคลื่นแม่เหล็กไฟฟ้า เพื่อให้แท็กเริ่มทำงานและจะส่งข้อมูลในหน่วยความจำที่ผ่านการมอดูเลตกับคลื่นพาหะแล้วออกมาทางสายอากาศที่อยู่ภายในแท็ก

2.8.3.3 คลื่นพาหะที่ถูกส่งออกมาจากแท็ก จะเกิดการเปลี่ยนแปลงแอมพลิจูด (Amplitude) ความถี่ หรือเฟสขึ้นอยู่กับวิธีการมอดูเลต

2.8.3.4 เครื่องอ่านอาร์เอฟไอดีจะตรวจจับความเปลี่ยนแปลงของคลื่นพาหะแล้วแปลงออกมาเป็นข้อมูล จากนั้นจะทำการถอดรหัสเพื่อนำข้อมูลไปใช้งานต่อไป

บทที่ 3
ขั้นตอนการดำเนินการ

โครงงานนี้เป็นการทำชุดการทดลองที่จอดรถอัตโนมัติโดยใช้โปรแกรม Arduino IDE ในการควบคุมโดยทำการค้นคว้ารูปแบบสถานที่จอดรถอัตโนมัติแบบ Rotary ถึงชุดขับเคลื่อนระบบขับเคลื่อนขั้นตอนและวิธีการนำรถไปเก็บจากจุดรับรถไปยังสถานที่เก็บรวบรวมไปถึงวิธีการนำไปเก็บของระบบจอดรถอัตโนมัติรูปแบบ Rotary แล้วนำไมโครคอนโทรลเลอร์มาประยุกต์ใช้ในการควบคุมชุดจำลองที่จอดรถอัตโนมัติ

3.1 แผนการดำเนินงาน

ตารางที่ 3.1 แผนการดำเนินงานของโครงการ

การดำเนินงาน	**ระยะเวลาดำเนินงาน (เดือนที่)**								
	1	2	3	4	5	6	7	8	9
1.ศึกษาการทำงานของระบบจอดรถอัตโนมัติในรูปแบบโรตารี่	▬▬▬	▬▬							
2.ศึกษาการใช้งานบอร์ด Arduino ร่วมกับ โปรแกรม Arduino IDE					▬▬▬	▬▬▬	▬▬▬	▬	
3.ออกแบบฮาร์ดแวร์		▬▬▬	▬▬▬	▬▬▬	▬				
4.ออกแบบซอฟต์แวร์		▬▬▬	▬▬▬	▬▬▬	▬▬▬	▬▬▬	▬▬▬	▬▬	
5.ทดสอบแบบจำลอง							▬▬▬	▬▬▬	
6.จัดทำปริญญานิพนธ์			▬▬▬	▬▬▬	▬▬▬	▬▬▬	▬▬▬	▬▬▬	

▬▬▬ แสดงแผนการดำเนินงาน

3.1.1 วิธีการดำเนินงาน

3.1.1.1 ศึกษาการค้นคว้าและรวบรวมข้อมูลเกี่ยวกับที่จอดรถอัตโนมัติแบบหมุนเป็นวงรี

3.1.1.2 ศึกษาการทำงานและวิธีการใช้งานของระบบอัตโนมัติแบบหมุนเป็นวงรี

3.1.1.3 ศึกษาการใช้งานโปรแกรม Arduino IDE

3.1.1.4 ออกแบบฮาร์ดแวร์

3.1.1.5 จัดทำวงจรและติดตั้งอุปกรณ์ที่มีใช้ในการควบคุม

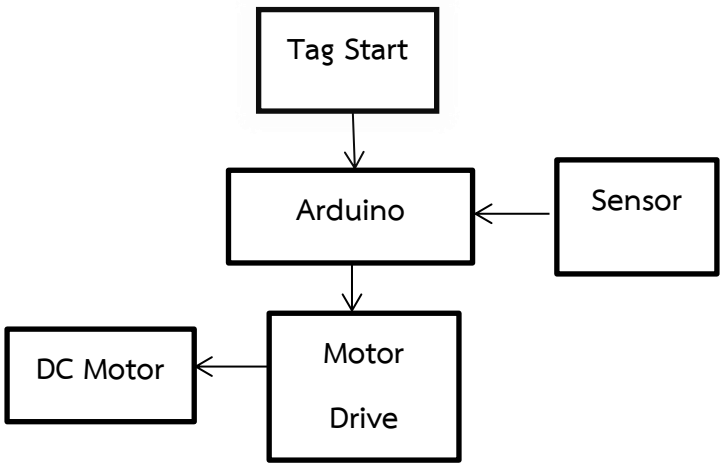
3.1.1.6 ออกแบบซอฟต์แวร์ เขียนโปรแกรมที่ใช้ในการควบคุม

3.1.1.7 ทดสอบการใช้งานของระบบ แก้ไขข้อบกพร่อง

3.1.1.8 สรุปผลการทดลอง

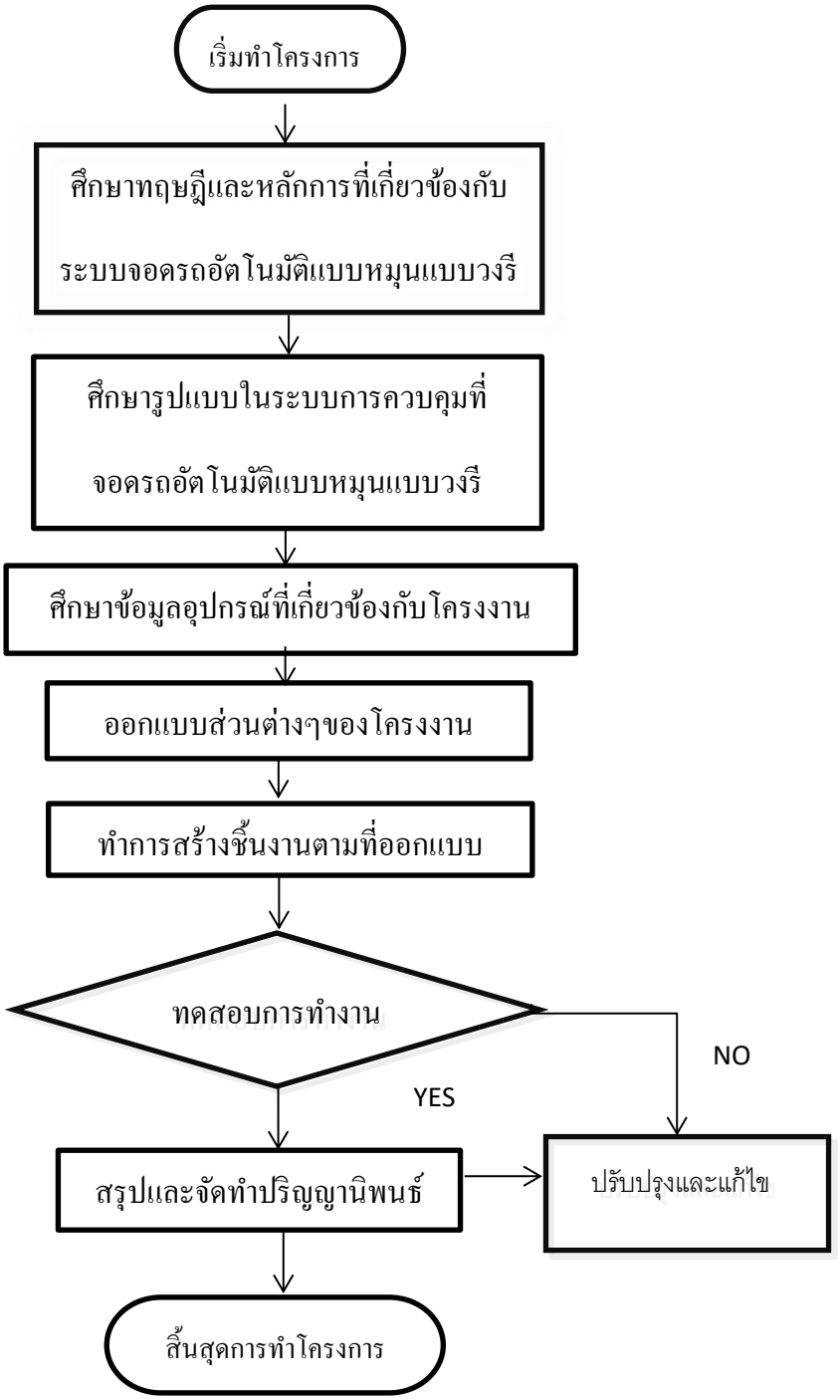
3.2 การออกแบบชุดจำลองระบบจอดรถอัตโนมัติ

การออกแบบการทำงานชุดจำลองระบบจอดรถอัตโนมัติ อุปกรณ์หลักที่ใช้ในการออกแบบคือ DC Motor โดยที่ใช้ Arduino UNO R3 เป็นตัวสั่งการ เพื่อไปควบคุมการทำงานของด้านเอาท์พุท คือ DC Motor มอเตอร์จะทำให้ระบบอัตโนมัตินั้นทำงานตามเงื่อนไขที่เราเขียนโปรแกรมลงในโปรแกรม Arduino IDE ไว้



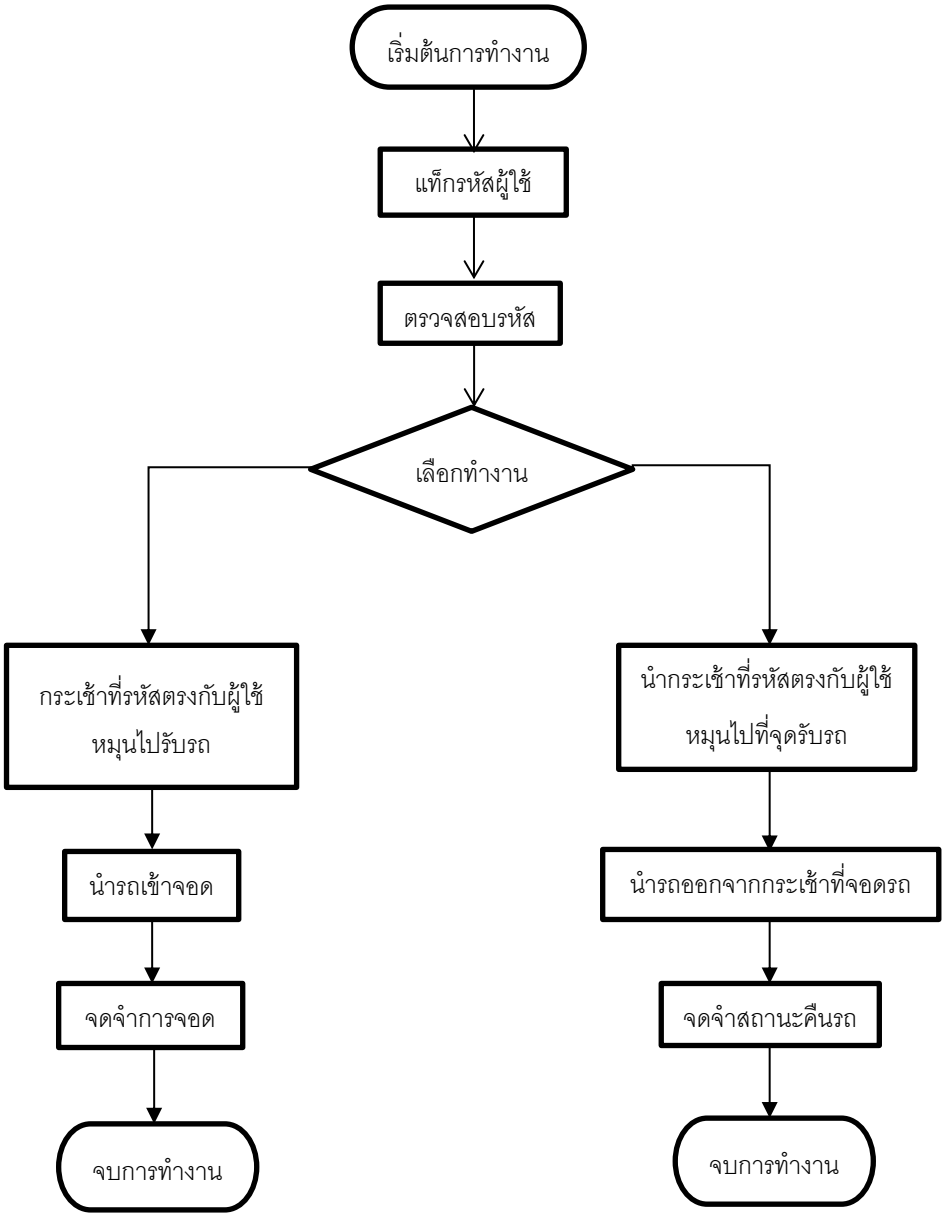
รูปที่ 3.1 การออกแบบการทำงานชุดจำลองระบบจอดรถอัตโนมัติแบบ Rotary

3.2.1 แผนการดำเนินงานของโครงการ



รูปที่ 3.2 แผนการดำเนินงานทั้งหมด

3.2.2 ระบบการทำงานที่จอดรถอัตโนมัติ



รูปที่ 3.3 การทำงานทั้งหมด

3.2.4 อุปกรณ์ที่เลือกใช้

3.2.4.1 มอเตอร์ในส่วนชุดรับรถขึ้น –ลง

ขนาดของมอเตอร์ที่จะนำมาใช้ในการหมุนเพื่อขับโหลด และการเลือกความเร็วที่ต้องการในการหมุนรับรถ จากการหาน้ำหนักของฐานชุดหมุนรับรถจำลองหนัก 0.2 กิโลกรัมต่อ 1 กระเช้า ออกแบบสามารถรับน้ำหนักโหลดได้ 0.3 กิโลกรัมต่อ 1 กระเช้า รวมน้ำหนักดังนั้นถ้า 6 กระเช้ารวมน้ำหนักของโหลดจะได้ 1.8 กิโลกรัม ดุมเฟืองหนัก 2 กิโลกรัม เพลาหนัก 3 กิโลกรัม โซ่หนัก 3 กิโลกรัม เพราะฉะนั้น กระเช้า เพลา ดุมเฟือง โซ่ หนักรวม 9.8 กิโลกรัม เส้นผ่านศูนย์กลางของเพลา โซ่ เท่ากับ 2 เซนติเมตร รัศมี เท่ากับ 1 เซนติเมตร

- การคำนวณหาแรงบิด (Torque) ของมอเตอร์ในส่วนชุดหมุนรับรถขึ้น – ลง

จากสมการ $$T = F \times r \qquad (3.1)$$

$$F = m \times g \qquad (3.2)$$

$$F = 9.8\ Kg \times 9.81\ \ m/s^2$$
$$= 96.138\ \ N \bullet m$$

เมื่อ $$r = 0.01\ \ m$$
$$T = 96.138 \times 0.01$$
$$= 0.96138\ \ N \bullet m$$

∴ แรงบิดมอเตอร์ในส่วนชุดหมุนรับรถขึ้น – ลง เท่ากับ 0.96138 นิวตันเมตร

เมื่อชุดรับรถมีโซ่ 48 ข้อ และเฟืองมี 38 ฟันหมุนจากกระเช้าแรกถึงกระเช้าสุดท้ายต้องการใช้เวลา 7 วินาที โดย 1 กระเช้า ใช้เวลาหมุน 1.16 วินาที จะได้

$$\frac{48}{38} = 1.26315 \quad \textit{รอบ}$$
$$= \frac{1.26315}{7} \times 60$$
$$N = 10.82706 \quad \text{รอบ/นาที}$$

∴ จะได้รอบของมอเตอร์เท่ากับ 10.82706 รอบ/นาที แทนค่าด้วย N

การคำนวณหากำลังไฟฟ้าของมอเตอร์

จากสมการ $$P = 2\pi \times T \times N \quad (3.3)$$

$$T = 0.96138 \ N \cdot m$$
$$N = 10.82706 \quad \text{รอบ/นาที}$$
$$P = 2\pi \times 0.96138 \times 10.82706$$
$$= 65.42749$$

วัตต์การคำนวณหากระแสไฟฟ้าของมอเตอร์ (ต้องการใช้แรงจ่าย 12 V)

จากสมการ $$I = \frac{P}{E} \quad (3.4)$$

$$I = \frac{65.42749 \ W}{12 \ V}$$
$$= 5.45229 \ A$$

∴ กระแสไฟฟ้าของมอเตอร์ในส่วนชุดรับรถ เท่ากับ 5.45229 แอมแปร์

จากผลการคำนวณน้ำหนักที่มอเตอร์ต้องรับภาระคือ 9.8 กิโลกรัม จึงเลือกใช้ DC มอเตอร์12 โวลต์ 2 แอมแปร์ 110 รอบ/นาที ที่มีความสามารถรับโหลดได้ 60 กิโลกรัม

รูปที่ 3.4 มอเตอร์เกียร์ที่เลือกใช้

ที่มา: http://www.robotpark.com.tr/image/cache/data/PRO/93262/93262_01-700x700.jpg

เมื่อความเร็วรอบมอเตอร์ 110 รอบ/นาที จะได้เวลาของการหมุนกระเช้าครบ 1 รอบ

- การคำนวณหาเวลาที่กระเช้าหมุนครบ 1 รอบ

$$= \frac{\text{จำนวณรอบมอเตอร์ } \mathbf{1} \text{ รอบการทำงาน} \times \text{เวลา}}{\text{ความเร็วรอบ}} \qquad (3.5)$$

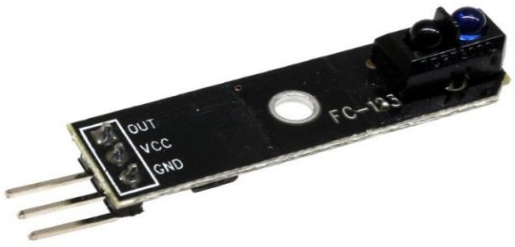
$$= \frac{1.26315 \times 60}{2.5}$$

$$= 50.525664 \quad \text{วินาที}$$

∴ ที่ความเร็วมอเตอร์ 2.5 รอบ/นาที จะใช้เวลา 50.53 วินาที ในการหมุนครบ 1 รอบ

3.2.4.2 เซนเซอร์ชนิดใช้แสง (Optical Sensor)

เลือกใช้เซนเซอร์ชนิดใช้แสงเพื่อใช้ตรวจจับวัตถุที่เป็นโลหะและอโลหะได้ การประยุกต์ใช้งานใช้ตรวจจับตำแหน่งแทนตัวลิมิตสวิตช์ ที่เลือกใช้เนื่องจากเป็นเซนเซอร์ที่มีความเล็กกะทัดรัด เหมาะสมกับชิ้นงาน เป็นการประหยัดพื้นที่ในการติดตั้ง

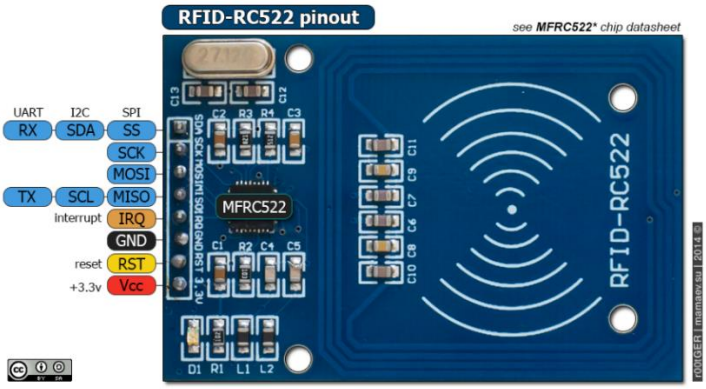


รูปที่ 3.5 track trace module infrared tracking probe 5000 tracking tracing sensor black and white line detection

ที่มา: http://g01.a.alicdn.com/kf/HTB1ZU4pKXXXXXa5XpXXq6xXFXXXP/Robot-1-Channel-font-b-Tracing-b-font-font-b-Tracking-b-font-Sensor-Module-Smart.jpgon

3.2.4.2 NFC (Near Field Communication)

เลือกใช้อาร์เอฟไอดี เพื่อใช้ยืนยันรหัสผู้ใช้งาน โดยการแท็กบัตรเพื่อเรียกกระเช้ามายังจุดรับรถ ที่เลือกใช้เนื่องจากเป็นระบบไร้สายที่มีความเหมาะสมกับชิ้นงานและง่ายต่อการใช้งาน



รูปที่ 3.6 RFID module RC522

ที่มา: http://img.olx.com.br/images/13/131716017415567.jpg

3.3 ขั้นตอนการดำเนินงาน

ขั้นตอนการสร้างชุดจำลองนี้จะแบ่งเป็น 2 ส่วน คือ ส่วนของโครงสร้างและส่วนของโปรแกรม โดยมีวิธีการดำเนินงานดังต่อไปนี้

3.3.1 ส่วนโครงสร้าง

หลังจากที่ได้ออกแบบโครงสร้างในโปรแกรม Solid Works แล้วจึงทำการจัดทำโครงสร้างขึ้นดังนี้

3.3.1.1 ตัดแท่งเหล็กทำโครงที่จอดรถขนาดความสูง 70.5 เซนติเมตร กว้าง 30 เซนติเมตร ขนาดเหล็ก 3 × 3 เซนติเมตร

3.3.1.2 เชื่อมเหล็กทำโครงที่จอดรถ จำนวน 2 อัน

3.3.1.3 นำถาดมาเชื่อมด้านข้าง 2 จุดและนำโครงมาประกอบเข้าด้วยกัน

3.3.1.4 นำน็อตมาเชื่อมกับโครงกระเช้า และนำแกนเหล็กขนาดยาว 21 เซนติเมตรมายึดติด

กับน็อต

3.3.1.5 นำดุมเฟืองไปกลึงเพื่อใส่ลูกปืนที่มีขนาดความโตเท่ากับเพลาซึ่งมีขนาดเส้นผ่านศูนย์กลางขนาด 2 เซนติเมตร และหาซื้อตุ๊กตาที่มีขนาดท่ากับเพลา

3.3.1.6 นำตุ๊กตา เพลา ดุมเฟืองมาประกอบใส่โซ่ทั้ง 2 ข้าง พร้อมติดแกนเพื่อไว้แขวนกระเช้าแล้วนำกระเช้ามาแขวน

3.3.1.7 นำกระเช้ามาประกอบจนครบ 6 กระเช้า และนำมอเตอร์มาติดตั้งพร้อมใส่เฟืองและโซ่เพื่อขับเพลาของทุกกระเช้า พ่นสีใหม่เพื่อความสวยงาม

3.3.1.8 ต่อวงจรควบคุมเข้ากับมอเตอร์ โดยการต่อแหล่งจ่ายไฟขนาด 12 V เข้าบอร์ดชุดขับมอเตอร์ และต่อไมโครคอนโทรลเลอร์เข้ากับชุดขับมอเตอร์เพื่อใช้ควบคุมมอเตอร์ให้หมุน แล้วใช้เซนเซอร์เป็นอินพุตเข้าไมโครคอนโทรลเลอร์เซนเซอร์

3.4 วิธีการทดสอบ

3.4.1 ทดสอบมอเตอร์เกียร์ 110 rpm โดยการป้อนแรงดันไฟฟ้าขนาด 12 โวลต์ เข้าที่มอเตอร์โดยผ่านชุด ไดร์มอเตอร์ เพื่อทดสอบการหมุนของมอเตอร์

3.4.2 ทดสอบ Optical Sensor เซนเซอร์ที่ใช้ติดตั้งจะมี 1 ตัว จะติดตั้งไว้ข้างจุดรับกระเช้ารถเบอร์ 1 โดยใช้นับกระเช้ารับรถ

3.4.3 ทดสอบความเร็วของกระเช้า โดยใช้ ไมโครคอนโทรเลอร์ ควบคุมการทำงาน

3.4.4 ทดสอบเริ่มต้นการทำงาน โดยการแท็กบัตรรหัสผู้ใช้ เพื่อสั่งการมอเตอร์ให้หมุนนำกระเช้ามายังรับรถ

3.4.5 ทดสอบนำรถเข้าจอดทำการทดสอบว่าเมื่อมีกระเช้าว่าง รถจะเข้ามาจอดได้หรือไม่

3.4.6 ทดสอบนำรถออก เมื่อมีข้อมูลส่งมาให้นำรถออก รถจะสามารถออกตรงตามเงื่อนไขได้หรือไม่

3.4.7 ทดสอบเมื่อที่จอดรถเต็ม ทำการทดสอบว่าจะสามารถเข้าไปจอดได้อีกหรือไม่

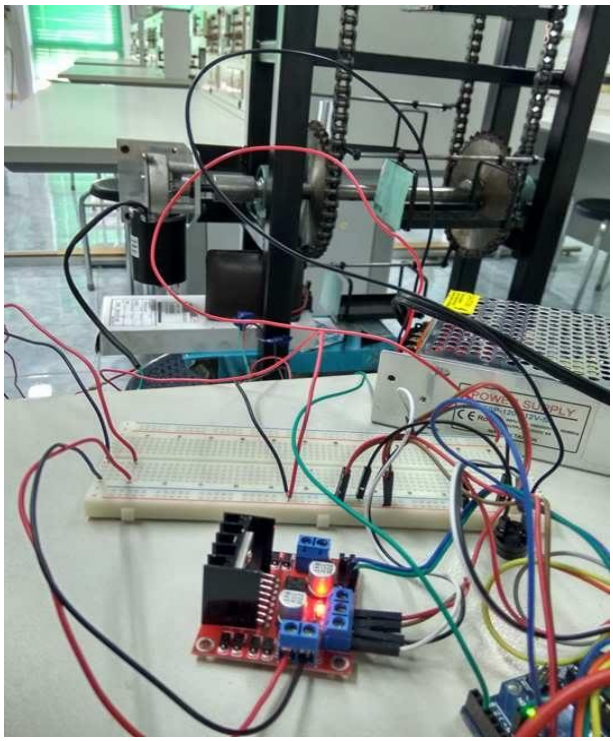
3.4.8 ทดสอบการใช้พลังงานของมอเตอร์

บทที่ 4
ผลการทดสอบ

ในการศึกษาระบบจอดรถอัตโนมัติ โดยใช้ ไมโครคอนโทลเลอร์ มาควบคุมการทำงานของกลไกต่างๆในระบบ โดยบันทึกข้อมูลที่ทำการทดลองได้ มาเปรียบเทียบดูความคลาดเคลื่อนของตำแหน่งจอดรถในแต่ละครั้ง และดูค่ากระแสไฟฟ้า แรงดันไฟฟ้าและกำลังไฟฟ้า เพื่อหาสาเหตุของปัญหาที่เกิดขึ้นซึ่งอาจเกิดได้ทั้งที่ตัวโครงสร้างหรือที่ตัวโปรแกรมเอง

4.1 การทดสอบมอเตอร์และชุดขับมอเตอร์

ดีซีมอเตอร์ ทำงานที่แรงดัน 12 V 110 rpm ทดสอบโดยใช้ ไมโครคอนโทลเลอร์ มาควบคุมการทำงาน ทดสอบแรงดันของการหมุนผ่านชุดไดร์ควบคุมมอเตอร์ขณะทำงานในกระเช้าของที่จอดรถอัตโนมัติดังรูป

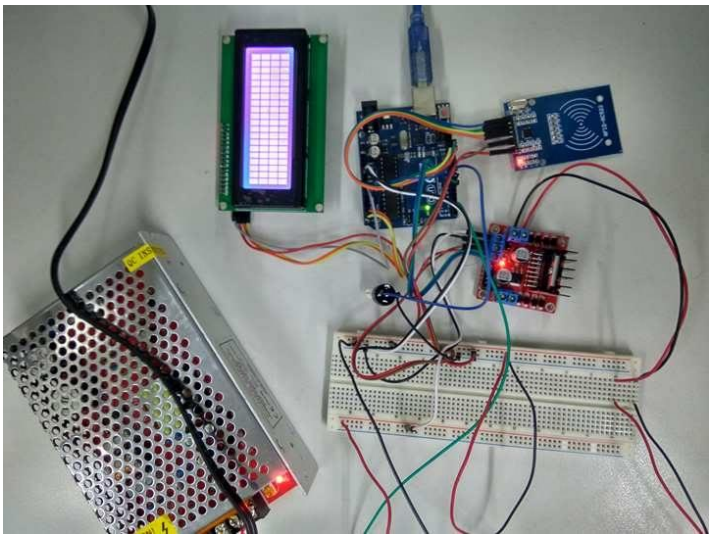


รูปที่ 4.1 วงจรมอเตอร์

ผลการทดสอบแรงดันที่ได้คือ ทดสอบการหมุนได้โดยป้อนแรงดันที่ +12 V ผ่านชุดไดร์มอเตอร์ มอเตอร์สามารถหมุนปกติ

4.2 การทดสอบความเร็วในการนำรถเข้าและนำรถออกโดยใช้ Microcontroller

การทดสอบจับเวลาในการนำรถเข้าและนำรถออก โดยจะใช้ Microcontroller ทดสอบการหมุนทางเดียวของมอเตอร์ จะเป็นการจับเวลาการทำงานในส่วนของการนำรถเข้าและการนำรถออก โดยแท็กบัตรผู้ใช้ผ่าน NFC เพื่อเรียกกระเช้าที่จะนำรถเข้าหรือนำรถออก



รูปที่ 4.2 วงจรการควบคุมที่จอดรถอัตโนมัติโดยใช้ Microcontroller

ตารางที่ 4.1 เวลาเฉลี่ยของการนำรถเข้าจอดแบบทางเดียวโดยใช้ Microcontroller

ตำแหน่งนำรถเข้า (จุดเริ่มต้นที่กระเช้าที่1)	**มอเตอร์หมุนทางเดียว**	
	เวลา (sec)	**กำลังไฟฟ้า (W)**
1	0.00	0,00
2	1.44	2.92
3	2.27	2.99
4	3.32	2.67
5	4.85	2.86
6	5.93	2.90

ผลการทดลองเวลาในเคลื่อนที่ของแต่ละกระเช้าเฉลี่ยเท่ากับ 2.96 และมีค่ากำลังไฟฟ้าเฉลี่ยเท่ากับ 2.39

4.3 การทดสอบเซนเซอร์

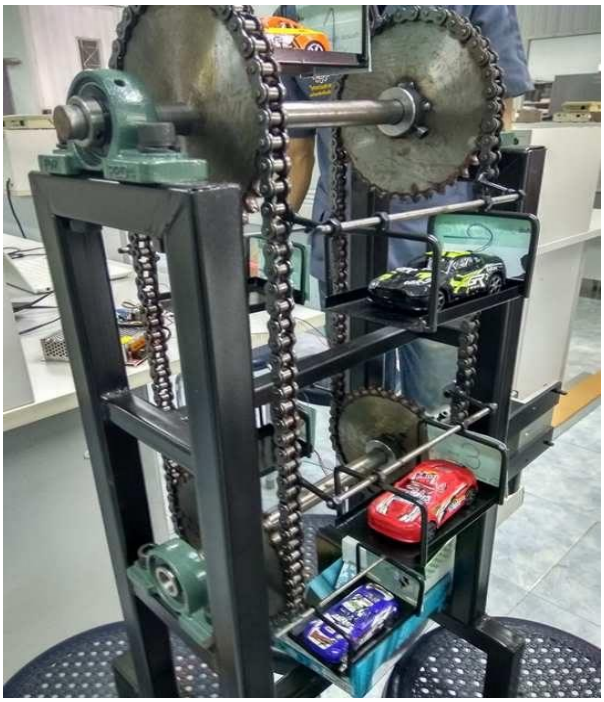
ทดสอบเซนเซอร์ว่ามีการนับกระเช้าได้ถูกต้องหรือไม่

รูปที่ 4.3 การทดสอบการตรวจจับกระเช้าของเซนเซอร์

กระเช้ามีการตรวจนับกระเช้าได้ถูกต้อง และเนื่องจากไมโครคอนโทรลเลอร์ไม่สามารถรับแรงดัน 12 V จากพาวเวอร์ซัพพลายได้ จึงมีการลดแรงดันจากวงจรให้เหลือ 5 V โดยใช้แหล่งจ่ายจากคอมพิวเตอร์แทน

4.4 การทดสอบเวลาของการนำรถเข้าและออก

การทดสอบจับเวลาในการนำรถเข้าและนำรถออกแต่ละครั้งจะเป็นการจับเวลาการทำงานในส่วนของการนำรถไปจอดยังตำแหน่งกระเช้า และการจับเวลาการนำรถออกที่จุดรับรถทั้งหมด 6 ตำแหน่ง โดยการวัดแรงดันของแต่ละกระเช้าผ่านโวลมิเตอร์



รูปที่ 4.4 การทดสอบแรงดันและคาบเวลาของกระเช้าที่ 3 ไปกระเช้าที่ 4

จากรูปที่ 4.4 จะเห็นว่าการเคลื่อนที่ของกระเช้ามีทิศทางที่ทวนเข็มนาฬิกา

ทำการทดสอบจับเวลาทั้งหมด 10 ครั้ง เพื่อนำมาหาค่าเวลาเฉลี่ยในการจอดรถในแต่ละตำแหน่ง และทำการวิเคราะห์หาสาเหตุของปัญหาที่เกิดขึ้นในการทดสอบ โดยจะทำการบันทึกค่าลงในตารางที่ 4.3 และตารางที่ 4.5 ดังต่อไปนี้

ตารางที่ 4.2 เวลานำรถเข้าจอด (จุดเริ่มต้นที่กระเช้าที่ 1 เสมอ)

ครั้งที่	นำรถจากตำแหน่งที่ 1 ไปจอดตามหมายเลขกระเช้า (วินาที)					
	1	2	3	4	5	6
1	0	1.32	2.03	3.32	4.59	6.01
2	0	1.26	2.61	3.77	4.92	5.63
3	0	1.72	2.61	3.60	4.69	5.57
4	0	1.01	2.30	3.30	5.02	6.23
5	0	1.29	2.50	3.46	4.41	5.53
6	0	1.12	2.33	3.21	4.99	5.89

ตารางที่ 4.2 เวลานำรถเข้าจอด (จุดเริ่มต้นที่กระเช้าที่ 1 เสมอ) (ต่อ)

ครั้งที่	นำรถจากตำแหน่งที่ 1 ไปจอดตามหมายเลขกระเช้า (วินาที)					
	1	2	3	4	5	6
7	0	1.33	2.45	3.57	4.55	5.90
8	0	1.21	2.35	3.60	4.90	5.55
9	0	1.10	2.01	3.33	4.89	5.70
10	0	1.05	2.68	3.49	4.56	5.81
เฉลี่ย	0	1.24	2.39	3.47	4.75	5.78

ตารางที่ 4.3 เวลาเฉลี่ยของการเข้าจอดรถต่อ 1 กระเช้า

กระเช้า	เวลา (วินาที)
1-2	1.24
2-3	1.15
3-4	1.08
4-5	1.28
5-6	1.03
เฉลี่ย	1.16

จากการทดสอบเวลาโดยเฉลี่ยของการเคลื่อนที่ของกระเช้าหมุนรับรถ 1 กระเช้านั้นจะใช้เวลาเฉลี่ยอยู่ที่ 1.16 วินาที

ตารางที่ 4.4 เวลานำรถออก (จุดเริ่มต้นที่กระเช้าที่ 1 เสมอ)

ครั้งที่	นำรถจากตำแหน่งที่ 1 ไปจอดตามหมายเลขกระเช้า (วินาที)					
	1	2	3	4	5	6
1	0	1.02	2.09	3.20	4.42	5.55
2	0	1.45	2.44	3.45	4.67	5.58

ตารางที่ 4.4 เวลานำรถออก (จุดเริ่มต้นที่กระเช้าที่ 1 เสมอ) (ต่อ)

ครั้งที่	นำรถจากตำแหน่งที่ 1 ไปจอดตามหมายเลขกระเช้า (วินาที)					
	1	2	3	4	5	6
3	0	1.32	2.60	3.72	4.88	6.01
4	0	1.22	2.51	3.41	4.59	6.00
5	0	1.25	2.54	3.30	4.82	5.81
6	0	1.34	2.08	3.66	4.77	5.98
7	0	1.08	2.28	3.64	4.76	5.76
8	0	1.12	2.14	3.21	4.47	5.60
9	0	1.27	2.21	3.33	4.76	5.58
10	0	1.02	2.41	3.52	4.43	5.55
เฉลี่ย	0	1.21	2.33	3.44	4.66	5.74

ตารางที่ 4.5 เวลาเฉลี่ยของการนำรถออกต่อ 1 กระเช้า

กระเช้า	เวลา (วินาที)
1-2	1.21
2-3	1.12
3-4	1.11
4-5	1.22
5-6	1.08
เฉลี่ย	1.15

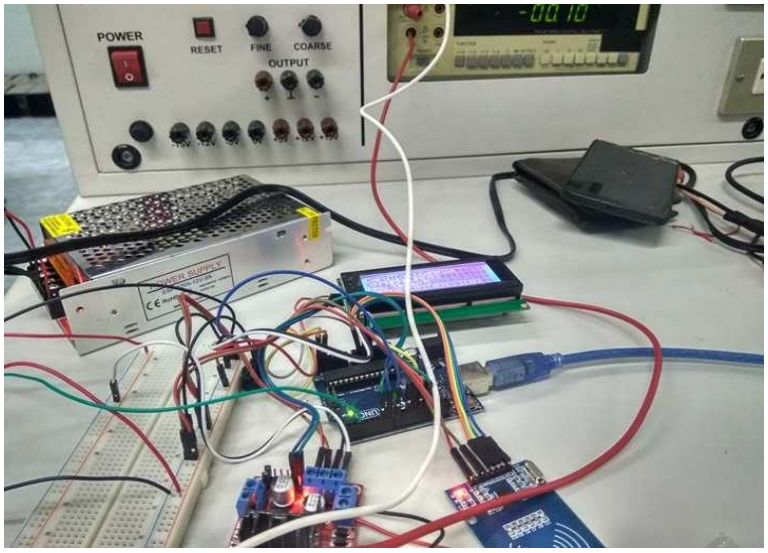
จากการทดสอบเวลาโดยเฉลี่ยของการเคลื่อนที่ของกระเช้าหมุนนำรถออก 1 กระเช้านั้นจะใช้เวลาเฉลี่ยอยู่ที่ 1.15 วินาที

จากการทดสอบจับเวลาในการรับและส่งรถในตารางที่ 4.3 และตารางที่ 4.5 จะนำเวลาเฉลี่ยทั้งหมดของการนำรถไปจอดกับเวลาเฉลี่ยทั้งหมดของการนำรถมาคืน เพื่อนำมาเปรียบเทียบว่าเวลาใกล้เคียงกันหรือไม่ดังนี้

ผลการทดลองเวลาเฉลี่ยในการนำรถเข้าจอดเท่ากับ 1.16 และเวลาเฉลี่ยในการนำรถออกเท่ากับ 1.15 จะเห็นได้ว่ามีเวลาเฉลี่ยที่ใกล้เคียงกันมาก เนื่องจากการนำรถเข้าและจอดมีรูปแบบการเรียกกระเช้าที่เหมือนกันต่างกันเพียงนำรถเข้าหรือนำรถออก

4.5 ผลการทดสอบกระแสและแรงดันของชุดรับรถที่เคลื่อนที่ในแนวตั้ง

การทดสอบการวัดค่ากระแสและแรงดันของชุดรับรถที่เคลื่อนที่ในแนวตั้ง คือการเคลื่อนที่ขึ้นและลงของชุดกระเช้ารับรถ ตามความสูงทั้งหมด 65 เซนติเมตร ทำการทดสอบทั้งหมด 10 ครั้ง โดยวัดค่ากระแสและค่าแรงดันในการทำงานในแต่ละครั้งของชุดรับรถที่เคลื่อนที่ เพื่อนำมาหาค่าเฉลี่ยของกระแสและแรงดันที่เกิดขึ้น และทำการวิเคราะห์สาเหตุของปัญหาที่เกิดขึ้น โดยใช้โหลดที่มีน้ำหนัก 1.8 กรัม ทั้งหมด 6 กระเช้า



รูปที่ 4.5 การทดสอบวัดค่ากระแสและแรงดัน

ตารางที่ 4.6 ทดสอบวัดค่ากระแสและแรงดันของชุดรับรถที่เคลื่อนที่ในแนวตั้ง

ครั้งที่	กระแส (A)		แรงดัน (V)		กำลังไฟฟ้า (W)	
	มีโหลด	ไม่มีโหลด	มีโหลด	ไม่มีโหลด	มีโหลด	ไม่มีโหลด
1	1.12	1.47	3.61	3.84	4.04	5.64
2	1.23	1.01	3.62	3.81	4.45	3.86
3	1.02	1.48	3.78	3.92	3.86	5.80
4	1.34	1.33	4.09	3.99	5.48	5.31
5	1.22	1.11	4.12	4.21	5.03	4.67

ตารางที่ 4.6 ทดสอบวัดค่ากระแสและแรงดันของชุดรับรถที่เคลื่อนที่ในแนวตั้ง (ต่อ)

ครั้งที่	กระแส (A)		แรงดัน (V)		กำลังไฟฟ้า (W)	
	มีโหลด	ไม่มีโหลด	มีโหลด	ไม่มีโหลด	มีโหลด	ไม่มีโหลด
6	0.95	1.45	4.02	4.08	3.82	5.92
7	1.04	1.55	3.89	4.01	4.04	6.21
8	0.98	1.03	3.76	4.14	3.68	4.26
9	0.99	1.25	3.91	3.97	3.87	4.96
10	1.09	1.24	3.99	3.90	4.35	4.84
เฉลี่ย	1.10	1.29	3.88	3.99	4.26	5.15

จากการทดสอบวัดค่ากระแสและแรงดันของชุดรับรถที่เคลื่อนที่ในแนวตั้ง นำผลที่ได้จากตารางที่ 4.6 ได้ค่ากระแสเฉลี่ยตั้งแต่ 1.10 ถึง 1.29 แอมป์ ได้ค่าแรงดันเฉลี่ยตั้งแต่ 3.88 ถึง 3.99 โวลต์ ได้นำมาหาค่ากำลังไฟฟ้าเฉลี่ยตั้งแต่ 4.26 ถึง 5.15 วัตต์ ซึ่งมอเตอร์ดีซีเกียร์ 110 rpm สามารถรับภาระได้ เนื่องจากมอเตอร์ตัวนี้มีพิกัดกระแส 2 แอมป์ พิกัดแรงดัน 12 โวลต์ พิกัดกำลังไฟฟ้า 120 วัตต์ สาเหตุที่การวัดค่ากระแสกับแรงดันได้ค่าไม่คงที่ มีสาเหตุมาจากส่วนที่มีการเสียดสีมากคือ เกิดจากชุดเฟือง ทำให้เกิดความไม่เสถียรของโครงสร้างจึงส่งผลให้กระแสและแรงดันที่วัดค่าได้ไม่คงที่

4.6 การทดสอบความคลาดเคลื่อน

การทดสอบความคลาดเคลื่อนของตำแหน่งจอดรถ จะแบ่งการทดสอบออกเป็น 2 แบบ คือ การทดสอบการสุ่มนำรถเข้าและการทดสอบการสุ่มนำรถออกทั้งหมด 10 ครั้ง เมื่อทดสอบครบทุกตำแหน่งถือเป็น 1 ครั้งและจะนำผลที่ได้มาหาค่าเปอร์เซ็นต์ความคลาดเคลื่อน

ตารางที่ 4.7 การทดสอบความคลาดเคลื่อนของการนำรถเข้าแบบสุ่ม

ครั้งที่	กระเช้าปัจจุบัน (มีรถจอดอยู่)	นำรถเข้า	ผลที่ได้
1	1,5,6	2	กระเช้าที่ 1,2,5,6 มีรถจอดอยู่
2	2,4,5	1,3	กระเช้าที่ 1,2,3,4,5 มีรถจอดอยู่
3	1,6	2	กระเช้าที่ 1,2,6 มีรถจอดอยู่

ตารางที่ 4.7 การทดสอบความคลาดเคลื่อนของการนำรถเข้าแบบสุ่ม (ต่อ)

ครั้งที่	กระเช้าปัจจุบัน (มีรถจอดอยู่)	นำรถเข้า	ผลที่ได้
4	2,4,5,6	1,3	ทุกกระเช้ามีรถจอดอยู่
5	1,2,4,5,6	3	ทุกกระเช้ามีรถจอดอยู่
6	2,5,6	1	กระเช้าที่ 1,2,5,6 มีรถจอดอยู่
7	0	3,4,5	กระเช้าที่ 3,4,5 มีรถจอดอยู่
8	3,4,5	1,2	กระเช้าที่ 1,2,3,4,5 มีรถจอดอยู่
9	1,4,6	2,3	กระเช้าที่ 1,2,3,4,6 มีรถจอดอยู่
10	0	2	กระเช้าที่ 2 มีรถจอดอยู่
4	2,4,5,6	1,3	ทุกกระเช้ามีรถจอดอยู่

ตารางที่ 4.8 การทดสอบความคลาดเคลื่อนของการนำรถออกแบบสุ่ม

ครั้งที่	กระเช้าปัจจุบัน (มีรถจอดอยู่)	นำรถออก	ผลที่ได้
1	1,5,6	5,6	กระเช้าที่ 1 มีรถจอดอยู่
2	2,4,5	4	กระเช้าที่ 2,5 มีรถจอดอยู่
3	1,6	1,6	ทุกกระเช้าว่าง
4	2,4,5,6	2,5,6	กระเช้าที่ 4 มีรถจอดอยู่
5	1,2,4,5,6	2,4	กระเช้าที่ 1,5,6 มีรถจอดอยู่
6	2,5,6	2,5,6	ทุกกระเช้าว่าง
7	1,2	2	กระเช้าที่ 1 มีรถจอดอยู่
8	1,2,3,4,5,6	1,2,3,4,5,6	ทุกกระเช้าว่าง
9	2,3,4,5,6	2,3	กระเช้าที่ 4,5,6 มีรถจอดอยู่
10	1,2,3,4	1,3,4	กระเช้าที่ 2 มีรถจอดอยู่

จากการทดลองใช้ไมโครคอนโทรลเลอร์ประมวลผลจากการทำงานตั้งแต่เริ่มต้น ได้ทำการทดสอบการนำรถเข้าและการนำรถออกแบบสุ่มทั้งหมด 10 ครั้ง พบว่าระบบมีการทำงานที่ถูกต้องตรงตามตำแหน่งที่คาดไว้ดังตารางที่ 4.7 และตารางที่ 4.8

4.7 การทดสอบอื่นๆ

การทดสอบโปรแกรม จะทำการทดสอบเป็นตำแหน่งโดยนำรถไปจอดที่จุดรับรถจากนั้นระบบจะนำรถไปจอดยังตำแหน่งที่ต้องการ เมื่อต้องการนำรถคืนระบบก็จะนำรถมาคืนที่จุดรับรถเสมอ โดยจะทำการทดสอบ 6 ตำแหน่งดังนี้

ตารางที่ 4.9 การทดสอบการทำงานอื่นๆของระบบ

การทดสอบ	**ผลการทดสอบ**
ทดสอบเริ่มต้นการทำงาน - จ่ายไฟเข้าสู่ระบบ	ระบบมีการส่งสถานะขึ้นที่ LED เพื่อรอคำสั่งรหัสผู้ใช้งาน
ทดสอบนำรถคันแรกเข้าจอด - แท็กบัตรผู้ใช้ที่ต้องการ	ระบบจะมีการตรวจสอบรหัสบัตรผู้ใช้แล้วเคลื่อนที่กระเช้าที่ตรงกับรหัสผู้ใช้มายังจุดเริ่มต้น
ทดสอบเมื่อที่จอดรถเต็ม - นำรถเข้าจอดจนครบ 6 กระเช้า	ระบบมีการแจ้งสถานะที่ LED ว่าเต็ม

บทที่ 5
บทสรุป

5.1 คำนำ

โครงงานวิศวกรรมนี้ มีจุดประสงค์เพื่อออกแบบรูปแบบการจำลองที่จอดรถอัตโนมัติโดยอาศัยหลักการทำงานของไมโครคอนโทรเลอร์ เพื่อใช้ในการประมวลผลการสั่งการมอเตอร์ ให้หมุนกระเช้าเพื่อมารับรถที่จะนำมาจอดและใช้ระบบ อาร์เอฟไอดี (RFID) ในการรวบตัวตนรหัสผู้ใช้งาน ซึ่งเป็นระบบไร้สายที่เหมาะสมกับชิ้นงานและง่านต่อการใช้งาน ทางคณะผู้จัดทำได้ทำการสรุปผลการทดลองและรวบรวมปัญหาต่างๆที่เกิดขึ้นระหว่างการทดลองเพื่อให้ผู้ที่สนใจเกี่ยวกับโครงงานนี้นำไปประยุกต์ใช้และพัฒนาต่อไป

5.2 สรุป

จากผลการทดลองทำการควบคุมการทำงานของแบบจำลองที่จอดรถอัตโนมัติแบบโรตารี่ กระเช้าแต่ละกระเช้ามีการเคลื่อนที่ห่างกันประมาณ 1 วินาที และมีการเคลื่อนที่วนลูปในทิศทางทวนเข้มนาฬิกา และเมื่อทำการทดสอบการจดจำตำแหน่งเข้าออกของรถในแต่ละกระเช้าในแบบสุ่มพบว่าระบบมีการจดจำได้โดยในแต่ละสถานะจะระบุขึ้นที่จอ LED โดยจากผลการทดลองแล้วจึงสรุปได้ว่าโครงการนี้สำเร็จได้ลุล่วงตามวัตถุประสงค์และขอบเขตที่กำหนดไว้

5.3 ปัญหาและข้อเสนอแนะ

ในขณะที่ทำการทดลองคณะผู้จัดทำได้พบปัญหา ระหว่างทำการทดลองซึ่งคณะผู้จัดทำได้ทำการสรุปปัญหาและข้อเสนอแนะดังนี้

ในส่วนของไมโครคอนโทรเลอร์ อาร์เอฟไอดี ต้องเสียเวลานานมากในการศึกษาข้อมูลเนื่องจากต้องศึกษาการทำงานของระบบต่างๆให้มีความเข้าใจอย่างลึกซึ้งและต้องเขียนโปรแกรมการทำงานให้มีความสัมพันธ์กัน ดังนั้นจึงเสนอแก่ผู้ที่จะนำไปศึกษาต่อ ควรที่จะศึกษาเนื้อหาเกี่ยวกับ Arduino ให้ดีก่อนหรือควรจะมีพื้นฐานที่ค่อนข้างดี จึงจะช่วยลดระยะเวลาการทำงานให้สั้นลง

ในส่วนของการออกแบบโครงของรูปแบบการจำลองที่จอดรถอัตโนมัติต้องออกแบบให้มีสัดส่วนที่พอดีกับจำนวนกระเช้าและความสูงความกว้างที่เหมาะกับขนาดของเฟืองและมอเตอร์ที่เหมาะสมขนาดของโหลดที่เป็นภาระของมอเตอร์ซึ่งถ้าโหลดมีขนาดที่ใหญ่ไปจะทำให้มอเตอร์อุณภูมิสูงและขับโหลดได้ช้า

เอกสารอ้างอิง

โชคทวี องค์เจริญสุข. (2556). *ความหมายของ RFID คืออะไร. องครักษ์ มหาวิทยาลัยศรีนครินทรวิโรฒ.*

ธีรวัฒน์ ประมวนผล. (2557). *การประยุกต์ใช้งานไมโครคอนโทรลเลอร์. องครักษ์ มหาวิทยาลัยศรีนครินทรวิโรฒ.*

http://www.lampangtc.ac.th/mnfile/branch5/file/knowledge/RFID.pdf

online: 18 Jan. 2017

https://inovancetech.com/ann.html

online: 5 Feb. 2017

http://www.w3ii.com/th/5g/5g_quick_guide.html

online: 1 Mar. 2017

https://www.techtalkthai.com/gigabit-wifi-802-11ac-5-things/

online: 15 Mar. 2017

https://www.blognone.com/node/42428

online: 20 Mar. 2017

ภาคผนวก

ภาคผนวก ก
(คำสั่งภาษาซี)

```
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
LiquidCrystal_I2C        lcd(0x3F,2,1,0,4,5,6,7);
#include <SPI.h>
#include <MFRC522.h>

  int buttonPin=2;
  int buttonState;
  int lastButtonState;

#define RST_PIN          9
#define SS_PIN           10
int pos = 0;
MFRC522 mfrc522(SS_PIN, RST_PIN);

String cardID;
int count;
int count_active;
unsigned long previousMillis = 0;
const long interval = 1000;

int motor_speed=255;
int en[2]={0,7};

int card_check;
```

```
char* id_card[7]={"AAA","8697a324","568b8324","068da724","859d53a8","16649f24","96418424"};
int park_use[7];
int state=0;

void setup() {

  Serial.begin(9600);
  lcd.begin (20,4);
  lcd.setBacklightPin(3,POSITIVE);
  lcd.setBacklight(HIGH);
    SPI.begin();
    mfrc522.PCD_Init();
    pinMode(2,INPUT);
    pinMode(6,OUTPUT);
    pinMode(7,OUTPUT);
}

void loop() {
  buttonState = digitalRead(buttonPin);
          if ( ! mfrc522.PICC_IsNewCardPresent()) {

  buttonState = digitalRead(buttonPin);
      if (buttonState != lastButtonState) {
    if (buttonState == HIGH) {
      Serial.println("Button1");
      state+=1;
            if(state>6){
       state=1;
      }
```

```
      if(state==card_check){
       drive("stop");
       if(park_use[state]==0){
         park_use[state]=1;
       }else{
         park_use[state]=0;
       }
      }
    }
    lastButtonState = buttonState;
  }
lcd.setCursor(0,0);
lcd.print("CardID:");
lcd.print(cardID);
lcd.setCursor(0,1);
lcd.print(">:");
lcd.setCursor(0,2);
lcd.print("Now Open:");
lcd.print(state);

lcd.setCursor(0,3);
lcd.print("["+String(park_use[1])+"]["+String(park_use[2])+"]["+String(park_use[3])+"]["+String(park
_use[4])+"]["+String(park_use[5])+"]["+String(park_use[6])+"]");

   unsigned long currentMillis = millis();
if (currentMillis - previousMillis >= interval) {
  previousMillis = currentMillis;
          if(count==3){
          cardID="";
```

```
          digitalWrite(2,LOW);
          count_active=0;
          count=0;
          card_check=0;
        }
    if(count_active==1){
         count+=1;
    }
  }
                    return;
          }

          // Select one of the cards
          if ( ! mfrc522.PICC_ReadCardSerial()) {
                    return;
          }else{
    tone(8,2000,200);
}
  card_check=0;
  cardID = "";
for (byte i = 0; i < mfrc522.uid.size; i++)

{
    cardID += String(mfrc522.uid.uidByte[i] < 0x15 ? "0" : "");
    cardID += String(mfrc522.uid.uidByte[i], HEX);
  }
  for(int i=1;i<=6;i++){
  if(cardID==id_card[i])
  card_check=i;
```

```
  drive("forward");
}
  if(card_check>=1){
   // count_active=1;
   // digitalWrite(2,HIGH);
    lcd.setCursor(0,1);
    lcd.print(">:PARKNUM["+String(card_check)+"]");
  }
}
void drive(String cmd){
  if(cmd=="stop"){
          analogWrite(6,0);
          digitalWrite(en[1], LOW);
        //  digitalWrite(en[2], LOW);
  }else if(cmd=="forward"){
          analogWrite(6,motor_speed);
          digitalWrite(en[1], HIGH);
  }
}
```

ภาคผนวก ข

(Data sheet)

1. Using a MFRC522 reader to read and write MIFARE RFID cards on ARDUINO through the MFRC522 library BY COOQROBOT.

Mario Capurso (m.capurso@libero.it)

MIFARE is the NXP Semiconductors-owned trademark of a series of chips widely used in contactless smart cards and proximity cards. According to the producers, billions of smart card chips and many millions of reader modules have been sold. [1]

The technology is owned by NXP Semiconductors (spin off from Philips Electronics in 2006) with headquarters in Eindhoven, Netherlands, and main business sites in Nijmegen, Netherlands, and Hamburg, Germany.

The MIFARE name covers proprietary technologies based upon various levels of the ISO/IEC 14443 Type A 13.56 MHz contactless smart card standard. The MIFARE name (derived from the term MIkron FARE Collection System) covers seven different kinds of contactless cards.

MIFARE Classic employs a proprietary protocol compliant to parts (but not all) of ISO/IEC 14443-3 Type A, with an NXP proprietary security protocol for authentication and ciphering.

The MIFARE Classic card is fundamentally just a memory storage device, where the memory is divided into segments and blocks with simple security mechanisms for access control. They are ASIC-based and have limited computational power. Thanks to their reliability and low cost, those cards are widely used for electronic wallet, access control, corporate ID cards, transportation or stadium ticketing.

The MIFARE Classic 1K offers 1024 bytes of data storage, split into 16 sectors; each sector is protected by two different keys, called A and B. Each key can be programmed to allow operations such as reading, writing, increasing valueblocks, etc. MIFARE Classic 4K offers 4096 bytes split into forty sectors, of which 32 are same size as in the 1K with eight more that are quadruple size sectors. MIFARE Classic mini offers 320 bytes split into five sectors.

For each of these card types, 16 bytes per sector are reserved for the keys and access conditions and can not normally be used for user data. Also, the very first 16 bytes contain the serial number of the card and certain other manufacturer data and are read only. That brings the net storage capacity of these cards down to 752 bytes for MIFARE Classic 1k, 3440 bytes for MIFARE Classic 4k, and 224 bytes for Mini. It uses an NXP proprietary security protocol (Crypto-1) for authentication and ciphering.

The encryption used by the MIFARE Classic card uses a 48 bit key. [18]

A presentation by Henryk Plötz and Karsten Nohl[19] at the Chaos Communication
Congress in December 2007 described a partial reverse-engineering of the algorithm used in the MIFARE Classic chip. Abstract and slides[20] are available online. A paper that describes the process of reverse engineering this chip was published at the August 2008 USENIX security conference.[21]

In March 2008 the Digital Security[22] research group of the Radboud University Nijmegen made public that they performed a complete reverse-engineering and were able to clone and manipulate the contents of an OV-Chipkaart which is a MIFARE Classic card.[23] For demonstration they used the Proxmark device, a 125 kHz / 13.56 MHz research instrument.[24] The schematics and software are released under the free GNU General Public License by Jonathan Westhues in 2007. They demonstrate it is even possible to perform card-only attacks using just an ordinary stock-commercial NFC reader in combination with the libnfc library.

In April 2009 new and better card-only attack on MIFARE Classic has been found. It was first announced at the Rump session of Eurocrypt 2009.[35] This attack was presented at SECRYPT 2009.[36] The full description of this latest and fastest attack to date can also be found in the IACR preprint archive.[37] The new attack improves by a factor of more than 10 all previous card-only attacks on MIFARE Classic, has instant running time, and it does not require a costly precomputation.

The new attack allows to recover the secret key of any sector of MIFARE Classic card via wireless interaction, within about 300 queries to the card. It can then be combined with the nested authentication attack in the Nijmegen Oakland paper to recover subsequent keys almost instantly. Both attacks combined and with the right hardware equipment such as Proxmark3, one should be able to clone any MIFARE Classic card in not more than 10 seconds. This is much faster than previously thought.

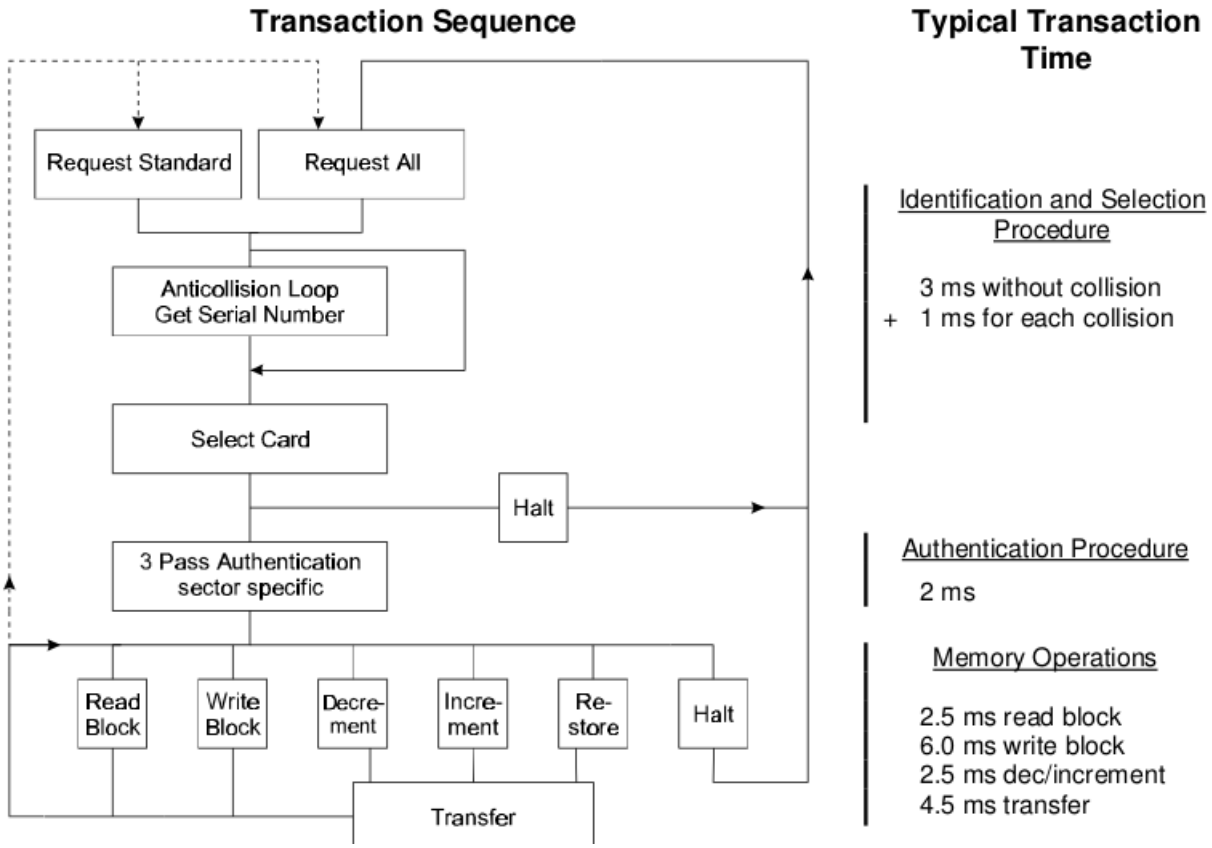
The MFRC522 is a highly integrated reader/writer IC for contactless communication at 13.56 MHz. The MFRC522 reader supports ISO/IEC 14443 A/MIFARE mode.
The MFRC522's internal transmitter is able to drive a reader/writer antenna designed to communicate with ISO/IEC 14443 A/MIFARE cards and transponders without additional active circuitry. The receiver module provides a robust and efficient implementation for demodulating and decoding signals from ISO/IEC 14443 A/MIFARE compatible cards and transponders. The digital module manages the complete ISO/IEC 14443 A framing and error detection (parity and CRC) functionality.

The MFRC522 supports all variants of the MIFARE Mini, MIFARE 1K, MIFARE 4K, MIFARE Ultralight, MIFARE DESFire EV1 and MIFARE Plus RF identification protocols. To aid readability throughout this data sheet, the MIFARE Mini,

MIFARE 1K, MIFARE 4K, MIFARE Ultralight, MIFARE DESFire EV1 and MIFARE Plus products and protocols have the generic name MIFARE.

The following host interfaces are provided:

- Serial Peripheral Interface (SPI)
- Serial UART (similar to RS232 with voltage levels dependant on pin voltage supply)
- I2C-bus interface



Answer to Request :

With the Answer to Request sequence the MIFARE® RWD (Read Write Device) requests all MIFARE® cards in the antenna field. When a card is in the operating range of a RWD, the RWD continues communication with the appropriate protocol.

Anticollision loop

In the Anticollision loop the serial number of the card is read. If there are several cards in the operating range of a RWD they can be distinguished by their different serial numbers and one can be selected (Select card) for further transactions. The unselected cards return to the standby mode and wait for a new Answer to Request and Anticollision loop.

Select Card

With the Select Card command the RWD selects one individual card for further authentication and memory related operations. The card returns the Answer to Select (ATS) code, which determines the individual type of the selected card.

Access Specification

After identification and selection of one card the RWD specifies the memory location of the following access.

Three Pass Authentication

The appropriate access key for the previously specified access is used for 3 Pass Authentication. Any communication after authentication is

automatically encrypted at the sender and decrypted by the receiver.

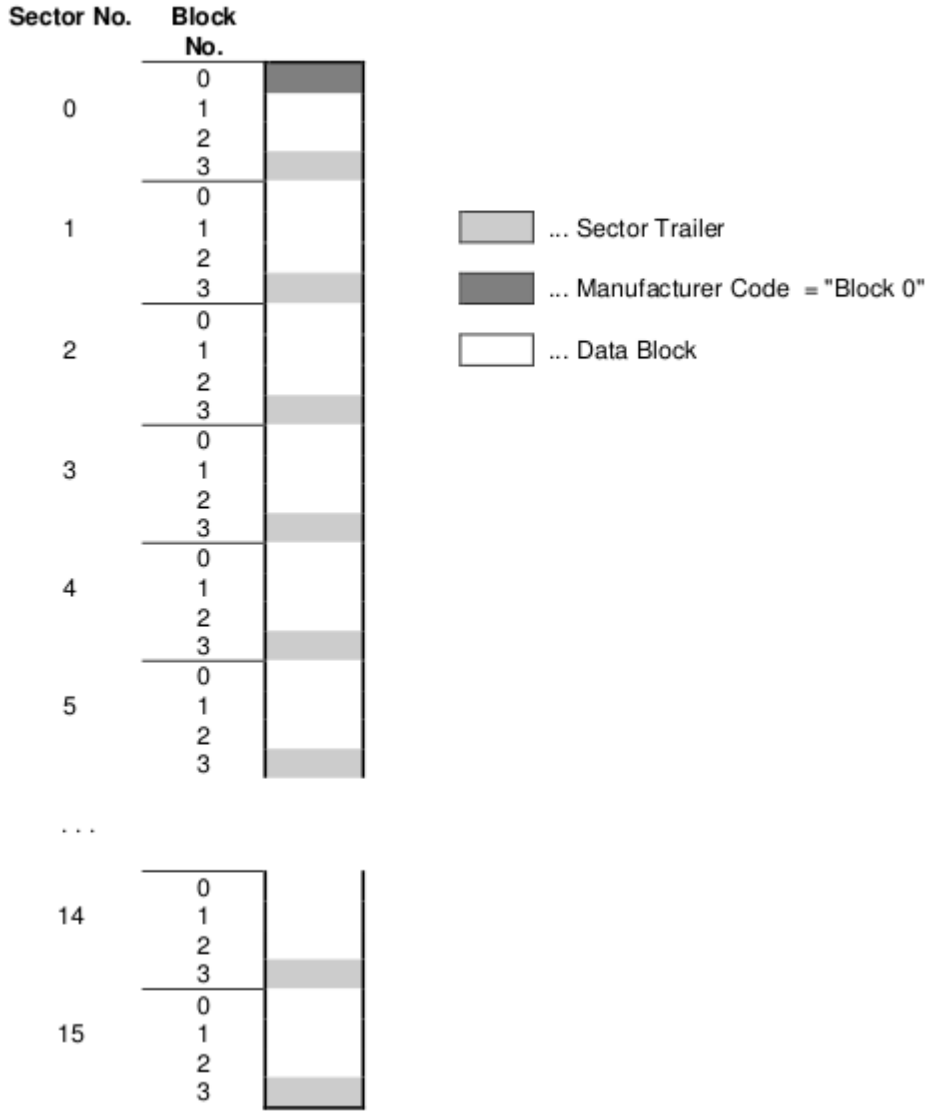
Read/Write

After authentication any of the following operations may be performed:

READ	reads one block
WRITE	writes one block
DECREMENT	decrements the contents of one block and stores the result in the data-register
INCREMENT	increments the contents of one block and stores the result in the data-register
TRANSFER	writes the contents of the data-register to one block
RESTORE	stores the contents of one block in the data-register

The MF1ICS50 IC of a Mifare Classic has integrated a 8192 Bit EEPROM which is split into 16 sectors with 4 blocks. One block consists of 16 bytes (1 Byte = 8 Bit).

Memory Organisation:



Manufacturer Code (Block 0 of Sector 0)

The first block of the memory is reserved for manufacturer data like 32 bit serial number. This is a read only block. In many documents it is named "Block 0".

Data Block (Block 0 to 3 except "Block 0")

Access conditions for the Data Blocks are defined in the Sector Trailers. According to these conditions data can be read, written, incremented, decremented, transferred or restored either with Key A, Key B or never.

2.6.1 Sector Trailer (Block 3):

Bit No. Byte No.	7 6 5 4 3 2 1 0	X ... Sector No. (0 to 15) Y ... Block No. (0 to 3)
0 1 2 3 4 5	KEYSECXA	
6 7 8 9	ACCESS CONDITIONS	
10 11 12 13 14 15	KEYSECXB (optional)	

bit#	7	6	5	4	3	2	1	0
→	C2X3_b	C2X2_b	C2X1_b	C2X0_b	C1X3_b	C1X2_b	C1X1_b	C1X0_b
	C1X3	C1X2	C1X1	C1X0	C3X3_b	C3X2_b	C3X1_b	C3X0_b
	C3X3	C3X2	C3X1	C3X0	C2X3	C2X2	C2X1	C2X0
→	BX7	BX6	BX5	BX4	BX3	BX2	BX1	BX0

_b stands for inversion; e.g. C1X0_b=INV(C1X0)

The fourth block of any sector is the Sector Trailer. The Sector Trailer contains access Key A (KEYSECXA) an optional Key B (KEYSECXB) and the access conditions for the four blocks of that sector. If Key B is not needed, the last 6 Bytes of block 3 can be used as data bytes. The corresponding access condition settings are marked grey below.
C1XY to C3XY which are stored twice for safety reasons define the access condition independently for the sector's four blocks. The last byte of the access conditions may be used to store some specific application data (e.g. location of the write backup block).

- Access condition for the Sector Trailer (Y = 3)

			KEYSECXA		ACCESS COND.		KEYSECXB	
C1X3	C2X3	C3X3	read	write	read	write	read	write
0	0	0	never	key A	key A	never	key A	key A
0	1	0	never	never	key A	never	key A	never
1	0	0	never	key B	key A|B	never	never	key B
1	1	0	never	never	key A|B	never	never	never
0	0	1	never	key A	key A	key A	key A	key A
0	1	1	never	key B	key A|B	key B	never	key B
1	0	1	never	never	key A|B	key B	never	never
1	1	1	never	never	key A|B	never	never	never

incr, decr, transfer, restore : never

NOTE: Key A|B means key A or key B;
If key B may be read (all grey marked lines) the memory space for Key B is used for data storage and it shall not be used for authentication because all further memory access operations will fail.

Since the transport access conditions (after chip manufacturing) equal to 001, new cards must not be authenticated with Key B !

- Access condition for Data Blocks (Y = 0 to 2)

C1XY	C2XY	C3XY	read	write	incr	decr, transfer, restore
0	0	0	keyA|B[1]	key A|B[1]	key A|B[1]	key A|B[1]
0	1	0	keyA|B[1]	never	never	never
1	0	0	keyA|B[1]	key B[1]	never	never
1	1	0	keyA|B[1]	key B[1]	key B[1]	key A|B[1]
0	0	1	keyA|B[1]	never	never	key A|B[1]
0	1	1	key B[1]	key B[1]	never	never
1	0	1	key B[1]	never	never	never
1	1	1	never	never	never	never

The process of decrement and increment of a block's data is performed and controlled by the Card-IC.

- Transport code

For transportation, KEYSECXA and the access conditions are predefined by the manufacturer as follows:

C1X0, C2X0, C3X0 = 0 0 0 block 0 (data block)
C1X1, C2X1, C3X1 = 0 0 0 block 1 (data block)
C1X2, C2X2, C3X2 = 0 0 0 block 2 (data block)
C1X3, C2X3, C3X3 = 0 0 1 block 3 (Sector Trailer)

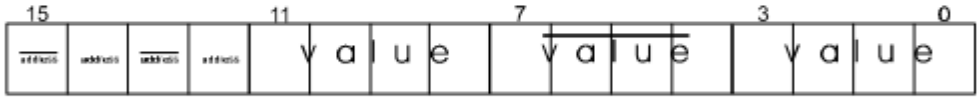
KEYSECXA .secret key, known only by
the manufacturer and system integrator

If Key B may be read in the corresponding Sector Trailer it cannot serve for authentication (all grey marked lines in previous table). Consequences: If the RWD tries to authenticate any block of a sector with key B using grey marked
access conditions, the card will refuse any subsequent memory access after authentication

In the MF1ICS50 IC two types of Data Blocks are used:

a) read/write blocks
are used to read and write general 16 bytes of data.

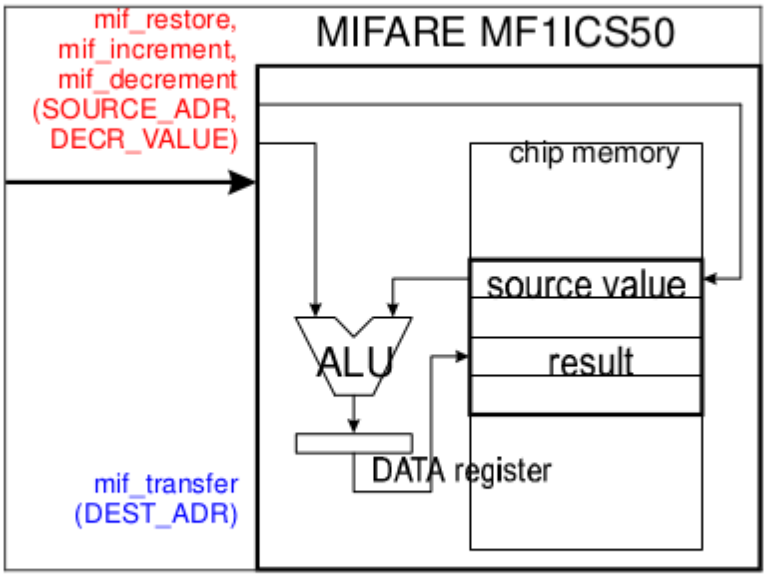
b) value blocks
are used for electronic purse functions (read, increment, decrement, transfer, restore). The maximum size of a value is 4 byte including sign bit, even when a complete 16 byte block has to be reserved. To provide error detection and correction capability, any value is stored 3 times into one value block. The remaining 4 bytes are reserved to some extent for check bits.



value: 32 bit signed 2th complement format stored 3 times
(the consistency of the 3 occurrences of the value is internally checked before the chip can perform any calculation)

address:8 bit arbitrary address byte stored 4 times
(this byte is not internally interpreted)

A value blocks is first time generated by a WRITE instruction to the desired address. The value may be used for subsequent DECREMENT / INCREMENT / RESTORE instructions.



The result of a calculation instruction is temporally stored in a buffer register. For updating the memory with the calculation result the TRANSFER instruction has to be issued. The chip refuses calculations if any error in the block format could be detected.

The described memory organization makes it possible to appoint different sectors to different applications and to prevent data corruption by using application specific secret keys. Keys can only be altered by a RWD which has stored the actual Key A or Key B if this is allowed according to access conditions. Otherwise the actual key cannot be changed anymore.

Before the execution of a command the correct format of the Access Conditions is checked by the Card-IC. Thus, when programming the Sector Trailer the card needs to be fixed within the operating range of a RWD's antenna to prevent interruption of the write operation because any unsuccessful write operation may lead to blocking the whole sector.

2.7 Memory contents after IC test

2.7.1 Block 0 (manufacturer block):

byte															byte
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Serial number				CB	manufacturer data										

CB: "serial number check byte" CB = byte 0 ^ byte 1 ^ byte 2 ^ byte 3 (^ ... XOR)

2.7.2 Data Blocks:

Data blocks: contain variable data.
(blocks 1,2 / 4,5,6 / 8,9,10 / 12,13,14 / 16,17,18 / 20,21,22 / 24,25,26 / 28,29,30 / 32,33,34 / 36,37,38 / 40,41,42 / 44,45,46 / 48,49,50 / 52,53,54 / 56,57,58 / 60,61,62)

2.7.3 Sector Trailers:

Note: The initial state of sector trailers after IC test can be modified depending on the personalisation done e.g. at the card manufacturer.

default coding:

byte															byte
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
transport key A						FF	07	80	xx	transport key B					

(blocks 3 / 7 / 11 / 15 / 19 / 23 / 27 / 31 / 35 / 39 / 43 / 47 / 51 / 55 / 59 / 63)
Byte 9 of all sector trailers is not defined. Its memory contents after IC test can vary.

MFRC522.h – A Library to use ARDUINO RFID MODULE KIT 13.56 MHZ BY COOQROBOT.

There are three hardware components involved:

1) The micro controller: An Arduino
2) The PCD (Proximity Coupling Device): NXP MFRC522 Contactless Reader IC
3) The PICC (short for Proximity Integrated Circuit Card): A card or tag using the ISO 14443A interface, eg Mifare or NTAG203.

MIFARE Classic 1K (MF1S503x):

Has 16 sectors * 4 blocks/sector * 16 bytes/block = 1024 bytes. The blocks are numbered 0-63.

Block 3 in each sector is the Sector Trailer.

* Bytes 0-5: Key A
* Bytes 6-8: Access Bits
* Bytes 9: User data
* Bytes 10-15: Key B (or user data)

Block 0 is read only manufacturer data.

To access a block, an authentication using a key from the block's sector must be performed first.

Example: To read from block 10, first authenticate using a key from sector 3 (blocks 8-11).

All keys are set to FFFFFFFFFFFFh at chip delivery.

Warning: Please read section 8.7 "Memory Access". It includes this text: if the PICC detects a format violation the whole sector is irreversibly blocked.

To use a block in "value block" mode (for Increment/Decrement operations) you need to change the sector trailer. Use PICC_SetAccessBits() to calculate the bit patterns.

Commands sent to the PICC.

The commands used by the PCD to manage communication with several PICCs (ISO 14443-3, Type A, section 6.4)

PICC_CMD_REQA = 0x26,
REQuest command, Type A. Invites PICCs in state IDLE to go to READY and prepare for anticollision or selection. 7 bit frame.
PICC_CMD_WUPA = 0x52,
Wake-UP command, Type A. Invites PICCs in state IDLE and HALT to go to READY(*) and prepare for anticollision or selection. 7 bit frame.
PICC_CMD_CT = 0x88,
Cascade Tag. Not really a command, but used during anti collision.
PICC_CMD_SEL_CL1 = 0x93,
Anti collision/Select, Cascade Level 1
PICC_CMD_SEL_CL2 = 0x95,

Anti collision/Select, Cascade Level 2
PICC_CMD_SEL_CL3 = 0x97,
Anti collision/Select, Cascade Level 3
PICC_CMD_HLTA = 0x50,
HaLT command, Type A. Instructs an ACTIVE PICC to go to state HALT.

The commands used for MIFARE Classic
Use PCD_MFAuthent to authenticate access to a sector, then use these commands to read/write/modify the blocks on the sector.
The read/write commands can also be used for MIFARE Ultralight.

PICC_CMD_MF_AUTH_KEY_A = 0x60,
Perform authentication with Key A
PICC_CMD_MF_AUTH_KEY_B = 0x61,
Perform authentication with Key B
PICC_CMD_MF_READ = 0x30,
Reads one 16 byte block from the authenticated sector of the PICC. Also used for MIFARE Ultralight.
PICC_CMD_MF_WRITE = 0xA0,
Writes one 16 byte block to the authenticated sector of the PICC. Called "COMPATIBILITY WRITE" for MIFARE Ultralight.
PICC_CMD_MF_DECREMENT = 0xC0,
Decrements the contents of a block and stores the result in the internal data register.
PICC_CMD_MF_INCREMENT = 0xC1,
Increments the contents of a block and stores the result in the internal data register.
PICC_CMD_MF_RESTORE = 0xC2,
Reads the contents of a block into the internal data register.
PICC_CMD_MF_TRANSFER = 0xB0,
Writes the contents of the internal data register to a block.

List of the functions in the library

Functions for setting up the Arduino

MFRC522(byte chipSelectPin, byte resetPowerDownPin);
void setSPIConfig();

Basic interface functions for communicating with the MFRC522

void PCD_WriteRegister(byte reg, byte value);
void PCD_WriteRegister(byte reg, byte count, byte *values);
byte PCD_ReadRegister(byte reg);
void PCD_ReadRegister(byte reg, byte count, byte *values, byte rxAlign = 0);
void setBitMask(unsigned char reg, unsigned char mask);
void PCD_SetRegisterBitMask(byte reg, byte mask);
void PCD_ClearRegisterBitMask(byte reg, byte mask);

byte PCD_CalculateCRC(byte *data, byte length, byte *result);

Functions for manipulating the MFRC522

void PCD_Init();
void PCD_Reset();
void PCD_AntennaOn();

Functions for communicating with PICCs

byte PCD_TransceiveData(byte *sendData, byte sendLen, byte *backData, byte *backLen, byte *validBits = NULL, byte rxAlign = 0, bool checkCRC = false);

byte PCD_CommunicateWithPICC(byte command, byte waitIRq, byte *sendData, byte sendLen, byte *backData = NULL, byte *backLen = NULL, byte *validBits = NULL, byte rxAlign = 0, bool checkCRC = false);

byte PICC_RequestA(byte *bufferATQA, byte *bufferSize);
byte PICC_WakeupA(byte *bufferATQA, byte *bufferSize);
byte PICC_REQA_or_WUPA(byte command, byte *bufferATQA, byte *bufferSize);
byte PICC_Select(Uid *uid, byte validBits = 0);
byte PICC_HaltA();

Functions for communicating with MIFARE PICCs

byte PCD_Authenticate(byte command, byte blockAddr, MIFARE_Key *key, Uid *uid);
void PCD_StopCrypto1();
byte MIFARE_Read(byte blockAddr, byte *buffer, byte *bufferSize);
byte MIFARE_Write(byte blockAddr, byte *buffer, byte bufferSize);
byte MIFARE_Decrement(byte blockAddr, long delta);
byte MIFARE_Increment(byte blockAddr, long delta);
byte MIFARE_Restore(byte blockAddr);
byte MIFARE_Transfer(byte blockAddr);
byte MIFARE_Ultralight_Write(byte page, byte *buffer, byte bufferSize);

Support functions

byte PCD_MIFARE_Transceive(byte *sendData, byte sendLen, bool acceptTimeout = false);
const char *GetStatusCodeName(byte code);
byte PICC_GetType(byte sak);
const char *PICC_GetTypeName(byte type);
void PICC_DumpToSerial(Uid *uid);
void PICC_DumpMifareClassicToSerial(Uid *uid, byte piccType, MIFARE_Key *key);
void PICC_DumpMifareClassicSectorToSerial(Uid *uid, MIFARE_Key *key, byte sector);
void PICC_DumpMifareUltralightToSerial();
void MIFARE_SetAccessBits(byte *accessBitBuffer, byte g0, byte g1, byte g2, byte g3);
Convenience functions - does not add extra functionality

bool PICC_IsNewCardPresent();
bool PICC_ReadCardSerial();

Detailed documentation – enum and structures

PICC types we can detect. Remember to update PICC_GetTypeName() if you add more.

enum PICC_Type

Return codes from the functions in this class. Remember to update GetStatusCodeName() if you add more.

enum StatusCode

A struct used for passing the UID of a PICC.

typedef struct {

byte	size;	Number of bytes in the UID. 4, 7 or 10.
byte	uidByte[10];	
byte	sak;	The SAK (Select acknowledge) byte returned from the PICC after successful selection.

} Uid

A struct used for passing a MIFARE Crypto1 key

typedef struct {
byte keyByte[MF_KEY_SIZE];
} MIFARE_Key;

Example:
// Prepare key - all keys are set to FFFFFFFFFFFFh at chip delivery from the factory.
MFRC522::MIFARE_Key key;
for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;

Member variables

Uid uid;	Used by PICC_ReadCardSerial().

Detailed documentation – functions

+Create object instance
MFRC522(
byte chipSelectPin, Arduino pin used for SPI chip select
byte resetPowerDownPin Arduino pin used for SPI reset
);
Example:

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10    //Arduino Uno
#define RST_PIN 9
MFRC522 mfrc522(SS_PIN, RST_PIN);        // Create MFRC522 instance.
```

+Initializes the MFRC522 chip.

```
void MFRC522::PCD_Init()
Example:
void setup() {
        Serial.begin(9600);        // Init serial communications with PC
        SPI.begin();               // Init SPI bus
        mfrc522.PCD_Init();        // Init MFRC522 card
}
```

+Set SPI bus to work with MFRC522 chip.

Please call this function if you have changed the SPI config since the MFRC522 constructor was run.

```
void MFRC522::setSPIConfig()
```

+Performs a soft reset on the MFRC522 chip and waits for it to be ready again.

```
void MFRC522::PCD_Reset()
```

+Turns the antenna on by enabling pins TX1 and TX2.
After a reset these pins are disabled.

```
void MFRC522::PCD_AntennaOn()
```

+Transmits a REQuest command, Type A. Invites PICCs in state IDLE to go to READY and prepare for anticollision or selection. 7 bit frame.
Beware: When two PICCs are in the field at the same time I often get STATUS_TIMEOUT - probably due do bad antenna design.
return STATUS_OK on success, STATUS_??? otherwise.

```
byte MFRC522::PICC_RequestA(
byte *bufferATQA,        The buffer to store the ATQA (Answer to request) in
byte *bufferSize         Buffer size, at least two bytes. Also number of bytes returned if STATUS_OK.
```

+Transmits a Wake-UP command, Type A. Invites PICCs in state IDLE and HALT to go to READY(*) and prepare for anticollision or selection. 7 bit frame.
Beware: When two PICCs are in the field at the same time I often get STATUS_TIMEOUT - probably due do bad antenna design.
return STATUS_OK on success, STATUS_??? otherwise.

```
byte MFRC522::PICC_WakeupA(
byte *bufferATQA,        The buffer to store the ATQA (Answer to request) in
byte *bufferSize         Buffer size, at least two bytes. Also number of bytes returned if STATUS_OK.
```

+Transmits SELECT/ANTICOLLISION commands to select a single PICC.
Before calling this function the PICCs must be placed in the READY(*) state by calling PICC_RequestA() or PICC_WakeupA().

On success:

-The chosen PICC is in state ACTIVE(*) and all other PICCs have returned to state IDLE/HALT. (Figure 7 of the ISO/IEC 14443-3 draft.)
-The UID size and value of the chosen PICC is returned in *uid along with the SAK.

A PICC UID consists of 4, 7 or 10 bytes.
Only 4 bytes can be specified in a SELECT command, so for the longer UIDs two or three iterations are used:

UID size	Number of UID bytes	Cascade levels	Example of PICC
single	4	1	MIFARE Classic
double	7	2	MIFARE Ultralight
triple	10	3	Not currently in use?

return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::PICC_Select(
Uid *uid, Pointer to Uid struct. Normally output, but can also
be used to supply a known UID.

byte validBits The number of known UID bits supplied in *uid.

Normally 0. If set you must also supply uid->size.

Description of buffer structure:

Byte 0: SEL Indicates the Cascade Level: PICC_CMD_SEL_CL1, PICC_CMD_SEL_CL2 or PICC_CMD_SEL_CL3
Byte 1: NVB Number of Valid Bits (in complete command, not just the UID): High nibble: complete bytes, Low nibble: Extra bits.
Byte 2: UID-data or CT See explanation below. CT means Cascade Tag.
Byte 3: UID-data
Byte 4: UID-data
Byte 5: UID-data
Byte 6: BCC Block Check Character - XOR of bytes 2-5
Byte 7: CRC_A
Byte 8: CRC_A

The BCC and CRC_A is only transmitted if we know all the UID bits of the current Cascade Level.
Description of bytes 2-5: (Section 6.5.4 of the ISO/IEC 14443-3 draft: UID contents and cascade levels)

UID size	Cascade level	Byte2	Byte3	Byte4	Byte5		
4 bytes	1		uid0	uid1	uid2	uid3	
7 bytes	1		CT	uid0	uid1	uid2	
		2		uid3	uid4	uid5	uid6
10 bytes	1		CT	uid0	uid1	uid2	
		2		CT	uid3	uid4	uid5

+Instructs a PICC in state ACTIVE(*) to go to state HALT.
return STATUS_OK on success, STATUS_??? otherwise.
byte PICC_HaltA();
Example:
// Halt PICC
mfrc522.PICC_HaltA();

+Executes the MFRC522 MFAuthent command.
This command manages MIFARE authentication to enable a secure communication to any MIFARE Mini, MIFARE 1K and MIFARE 4K card.
The authentication is described in the MFRC522 datasheet section 10.3.1.9 and http://www.nxp.com/documents/data_sheet/MF1S503x.pdf section 10.1. for use with MIFARE Classic PICCs.
The PICC must be selected - ie in state ACTIVE(*) - before calling this function.
Remember to call PCD_StopCrypto1() at the end of communication with the authenticated PICC - otherwise no new communications can start.
All keys are set to FFFFFFFFFFFFh at chip delivery.
return STATUS_OK on success, STATUS_??? otherwise. Probably STATUS_TIMEOUT if you supply the wrong key.
byte MFRC522::PCD_Authenticate(
byte command, PICC_CMD_MF_AUTH_KEY_A or PICC_CMD_MF_AUTH_KEY_B
byte blockAddr, The block number. See numbering in the comments in
  the .h file.
MIFARE_Key *key, Pointer to the Crypto1 key to use (6 bytes)
Uid *uid Pointer to Uid struct. The first 4 bytes of the UID
  is used.
)
Example:
MFRC522::MIFARE_Key key;
for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;
byte trailerBlock   = 7;
byte status;
if (! mfrc522.PICC_ReadCardSerial()) return;
status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, trailerBlock, &key, &(mfrc522.uid));
if (status != MFRC522::STATUS_OK) {
  Serial.print("PCD_Authenticate() failed: ");
  Serial.println(mfrc522.GetStatusCodeName(status));
  return;
}

+Used to exit the PCD from its authenticated state.
Remember to call PCD_StopCrypto1() at the end of communication with the authenticated PICC - otherwise no new communications can start.
void MFRC522::PCD_StopCrypto1()
Example:
// Stop encryption on PCD
mfrc522.PCD_StopCrypto1();

+Reads 16 bytes (+ 2 bytes CRC_A) from the active PICC.

For MIFARE Classic the sector containing the block must be authenticated before calling this function.
For MIFARE Ultralight only addresses 00h to 0Fh are decoded.
 * The MF0ICU1 returns a NAK for higher addresses.
 * The MF0ICU1 responds to the READ command by sending 16 bytes starting from
 the page address defined by the command argument.
 * For example; if blockAddr is 03h then pages 03h, 04h, 05h, 06h are returned.
 * A roll-back is implemented: If blockAddr is 0Eh, then the contents of pages 0Eh, 0Fh, 00h and 01h are returned.
The buffer must be at least 18 bytes because a CRC_A is also returned.
Checks the CRC_A before returning STATUS_OK.
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_Read(
byte blockAddr, MIFARE Classic: The block (0-0xff) number.
MIFARE Ultralight: The first page to return data from.
byte *buffer, The buffer to store the data in

byte *bufferSize Buffer size, at least 18 bytes. Also number of bytes returned
if STATUS_OK.
Example:
byte valueBlockA = 4; byte buffer[18]; byte size = sizeof(buffer);
byte status = mfrc522.MIFARE_Read(valueBlockA, buffer, &size);

+Writes 16 bytes to the active PICC.
For MIFARE Classic the sector containing the block must be authenticated before calling this function.
For MIFARE Ultralight the operation is called "COMPATIBILITY WRITE".
Even though 16 bytes are transferred to the Ultralight PICC, only the least significant 4 bytes (bytes 0 to 3) are written to the specified address. It is recommended to set the remaining bytes 04h to 0Fh to all logic 0.
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_Write(
byte blockAddr,
MIFARE Classic: The block (0-0xff) number.
MIFARE Ultralight: The page (2-15) to write to.
byte *buffer, The 16 bytes to write to the PICC
byte bufferSize Buffer size, must be at least 16 bytes. Exactly 16 bytes are
written.
Example:
byte valueBlockA = 4;
byte value1Block[] = { 1,2,3,4, 5,6,7,8, 9,10,255,12, 13,14,15,16};
status = mfrc522.MIFARE_Write(valueBlockA, value1Block, 16);
if (status != MFRC522::STATUS_OK) {
 Serial.print("MIFARE_Write() failed: ");
 Serial.println(mfrc522.GetStatusCodeName(status));
}

+MIFARE Decrement subtracts the delta from the value of the addressed block, and stores the result in a volatile memory.
For MIFARE Classic only. The sector containing the block must be authenticated before calling this function.
Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001].

Use MIFARE_Transfer() to store the result in a block.
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_Decrement(
byte blockAddr, The block (0-0xff) number.
long delta This number is subtracted from the value of block blockAddr.

+MIFARE Increment adds the delta to the value of the addressed block, and stores the result in a volatile memory.
For MIFARE Classic only. The sector containing the block must be authenticated before calling this function.
Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001].
Use MIFARE_Transfer() to store the result in a block.
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_Increment(
byte blockAddr, The block (0-0xff) number.
long delta This number is added to the value of block blockAddr.
Example:

```
// Add 1 to the value of valueBlockA and store the result in valueBlockA.
byte valueBlockA = 5;
Serial.print("Adding 1 to value of block "); Serial.println(valueBlockA);
byte status = mfrc522.MIFARE_Increment(valueBlockA, 1);
if (status != MFRC522::STATUS_OK) {
        Serial.print("MIFARE_Increment() failed: ");
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
}
status = mfrc522.MIFARE_Transfer(valueBlockA);
if (status != MFRC522::STATUS_OK) {
        Serial.print("MIFARE_Transfer() failed: ");
        Serial.println(mfrc522.GetStatusCodeName(status));
        return;
}
```

+MIFARE Restore copies the value of the addressed block into a volatile memory.
For MIFARE Classic only. The sector containing the block must be authenticated before calling this function.
Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001].
Use MIFARE_Transfer() to store the result in a block.
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_Restore(
byte blockAddr The block (0-0xff) number.
The datasheet describes Restore as a two step operation, but does not explain what data to transfer in step 2.Doing only a single step does not work, so I chose to transfer 0L in step two.

+Helper function for the two-step MIFARE Classic protocol operations Decrement, Increment and Restore.
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_TwoStepHelper(
byte command, The command to use

byte blockAddr, The block (0-0xff) number.
long data The data to transfer in step 2
)

+MIFARE Transfer writes the value stored in the volatile memory into one MIFARE Classic block.
For MIFARE Classic only. The sector containing the block must be authenticated before calling this function.
Only for blocks in "value block" mode, ie with access bits [C1 C2 C3] = [110] or [001].
return STATUS_OK on success, STATUS_??? otherwise.
byte MFRC522::MIFARE_Transfer(
byte blockAddr The block (0-0xff) number.

+Returns a string pointer to a status code name.
const char *MFRC522::GetStatusCodeName(
byte code One of the StatusCode enums.

code values	return values
STATUS_OK:	"Success."
STATUS_ERROR:	"Error in communication."
STATUS_COLLISION:	"Collision detected."
STATUS_TIMEOUT:	"Timeout in communication."
STATUS_NO_ROOM:	"A buffer is not big enough."
STATUS_INTERNAL_ERROR:	"Internal error in the code. Should not happen."
STATUS_INVALID:	"Invalid argument."
STATUS_CRC_WRONG:	"The CRC_A does not match."
STATUS_MIFARE_NACK:	"A MIFARE PICC responded with NAK."
default:	"Unknown error"

Example:
Serial.println(mfrc522.GetStatusCodeName(status));

+Translates the SAK (Select Acknowledge) to a PICC type.
return PICC_Type
byte MFRC522::PICC_GetType(
byte sak The SAK byte returned from PICC_Select().

sak	return value
sak & 0x04	PICC_TYPE_NOT_COMPLETE
0x09	PICC_TYPE_MIFARE_MINI
0x08	PICC_TYPE_MIFARE_1K
0x18	PICC_TYPE_MIFARE_4K
0x00	PICC_TYPE_MIFARE_UL
0x10 or 0x11	PICC_TYPE_MIFARE_PLUS
0x01	PICC_TYPE_TNP3XXX
sak & 0x20	PICC_TYPE_ISO_14443_4
sak & 0x40	PICC_TYPE_ISO_18092

else PICC_TYPE_UNKNOWN

+Returns a string pointer to the PICC type name.
const char *MFRC522::PICC_GetTypeName(
byte piccType One of the PICC_Type enums.

piccType return value
PICC_TYPE_ISO_14443_4 "PICC compliant with ISO/IEC 14443-4"
PICC_TYPE_ISO_18092: "PICC compliant with ISO/IEC 18092 (NFC)"
PICC_TYPE_MIFARE_MINI "MIFARE Mini, 320 bytes"
PICC_TYPE_MIFARE_1K "MIFARE 1KB"
PICC_TYPE_MIFARE_4K "MIFARE 4KB"
PICC_TYPE_MIFARE_UL "MIFARE Ultralight or Ultralight C"
PICC_TYPE_MIFARE_PLUS "MIFARE Plus"

PICC_TYPE_TNP3XXX "MIFARE TNP3XXX"
PICC_TYPE_NOT_COMPLETE "SAK indicates UID is not complete."
PICC_TYPE_UNKNOWN "Unknown type”
Example:
Serial.println(mfrc522.PICC_GetTypeName(piccType));

+Dumps debug info about the selected PICC to Serial.
On success the PICC is halted after dumping the data.
For MIFARE Classic the factory default key of 0xFFFFFFFFFFFF is tried.
void MFRC522::PICC_DumpToSerial(
Uid *uid Pointer to Uid struct returned from a successful PICC_Select().
Example:
mfrc522.PICC_DumpToSerial(&(mfrc522.uid));

+Dumps memory contents of a sector of a MIFARE Classic PICC.
Uses PCD_Authenticate(), MIFARE_Read() and PCD_StopCrypto1.
Always uses PICC_CMD_MF_AUTH_KEY_A because only Key A can always read the sector trailer access bits.
void MFRC522::PICC_DumpMifareClassicSectorToSerial(
Uid *uid, Pointer to Uid struct returned from a successful PICC_Select()
MIFARE_Key *key, Key A for the sector.
byte sector The sector to dump, 0..39.

+Calculates the bit pattern needed for the specified access bits. In the [C1 C2 C3] tuples C1 is MSB (=4) and C3 is LSB (=1).
void MFRC522::MIFARE_SetAccessBits(
byte *accessBitBuffer, Pointer to byte 6, 7 and 8 in the sector trailer.
Bytes [0..2] will be set.

byte g0, Access bits [C1 C2 C3] for block 0
(for sectors 0-31) or blocks 0-4 (for sectors 32-39)

byte g1, Access bits C1 C2 C3] for block 1 (for sectors 0-31)

or blocks 5-9 (for sectors 32-39)

byte g2, Access bits C1 C2 C3] for block 2 (for sectors 0-31)
or blocks 10-14 (for sectors 32-39)

byte g3 Access bits C1 C2 C3] for the sector trailer, block 3
(for sectors 0-31) or block 15 (for sectors 32-39)

The access bits are stored in a peculiar fashion.
There are four groups:

g[3] Access bits for the sector trailer, block 3 (for sectors 0-31) or block 15 (for sectors 32-39)

g[2] Access bits for block 2 (for sectors 0-31) or blocks 10-14 (for sectors 32-39)
g[1] Access bits for block 1 (for sectors 0-31) or blocks 5-9 (for sectors 32-39)
g[0] Access bits for block 0 (for sectors 0-31) or blocks 0-4 (for sectors 32-39)

Each group has access bits [C1 C2 C3]. In this code C1 is MSB and C3 is LSB.
The four CX bits are stored together in a nibble cx and an inverted nibble cx_.
Example:
// Sector trailer that defines blocks 5 and 6 as Value Blocks and enables key B.
byte trailerBuffer[] = {255,255,255,255,255,255,0,0,0,0, 255,255,255,255,255,255};
// Keep default keys.
// g1=6(i.e.110) => block 5 value block. Key B write&increment, A or B decrement.
// g2=6 => Same thing for block 6.
// g3=3 => Key B must be used to modify the Sector Trailer. Key B becomes valid.
mfrc522.MIFARE_SetAccessBits(&trailerBuffer[6], 0, 6, 6, 3);

+Check if a card is present
bool MFRC522::PICC_IsNewCardPresent()
Returns true if a PICC responds to PICC_CMD_REQA.
Only "new" cards in state IDLE are invited. Sleeping cards in state HALT are ignored.
Example:
// Look for new cards
if (! mfrc522.PICC_IsNewCardPresent()) return;

+Read Card Serial
bool MFRC522::PICC_ReadCardSerial()
Simple wrapper around PICC_Select.
Returns true if a UID could be read.The read UID is available in the class variable uid.
Remember to call PICC_IsNewCardPresent(), PICC_RequestA() or PICC_WakeupA() first.
return true if one selected
Now a card is selected. The UID and SAK is in mfrc522.uid.
Example:
Serial.print("Card UID:");
for (byte i = 0; i < mfrc522.uid.size; i++) {

```
    Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
    Serial.print(mfrc522.uid.uidByte[i], HEX);
}
// Dump PICC type
byte piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.print("PICC type: ");
Serial.println(mfrc522.PICC_GetTypeName(piccType));
if ( piccType != MFRC522::PICC_TYPE_MIFARE_MINI
    &&  piccType != MFRC522::PICC_TYPE_MIFARE_1K
    &&      piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println("This sample only works with MIFARE Classic cards.");
    return;
}
```

How to convert a normal block into a value block

Usually a normal block has 000 access bits, while a value block has 110 access bits

Example:

```
byte valueBlockA        = 5;
byte valueBlockB        = 6;
byte trailerBlock       = 7;
// We need a sector trailer with blocks 5 and 6 as Value Blocks and enables key B.
byte trailerBuffer[] = { 255,255,255,255,255,255,
                                                0,0,0,0,255,255,255,255,255,255}; // Keep default keys.
// g1=6 => block 5 as value block. Key B to write&increment, A or B for decrement.
// g2=6 => Same thing for block 6.
// g3=3 => Key B must be used to modify the Sector Trailer. Key B becomes valid.
mfrc522.MIFARE_SetAccessBits(&trailerBuffer[6], 0, 6, 6, 3);
byte status = mfrc522.MIFARE_Write(trailerBlock, trailerBuffer, 16);
if (status != MFRC522::STATUS_OK) {
            Serial.print("MIFARE_Write() failed: ");
            Serial.println(mfrc522.GetStatusCodeName(status));
            return;
}
```

How to setup a Value Block with a value set to zero

Example:

```
byte  blockAddr=5;
byte valueBlock[] = {0,0,0,0, 255,255,255,255, 0,0,0,0,
            blockAddr,~blockAddr,blockAddr,~blockAddr };
byte status = mfrc522.MIFARE_Write(blockAddr, valueBlock, 16);
if (status != MFRC522::STATUS_OK) {
            Serial.print("MIFARE_Write() failed: ");
            Serial.println(mfrc522.GetStatusCodeName(status));
}
```

ประวัติย่อผู้ทำโครงงาน

ประวัติย่อผู้ทำโครงงาน

ชื่อ ชื่อสกุล	นายกิตติวาท พุทธิมณี
วันเดือนปีเกิด	10 พฤษภาคม 2538
สถานที่เกิด	อำเภอเมือง จังหวัดสุรินทร์
สถานที่อยู่ปัจจุบัน	22 ถ.สืบสหการ ต.ในเมือง อ.เมือง จ.สุรินทร์ 32000
หมายเลขโทรศัพท์ติดต่อ	091-340-6692

ประวัติการศึกษา

พ.ศ. 2556	มัธยมศึกษาปีที่ 6 จากโรงเรียนสุรวิทยาคาร
พ.ศ. 2560	กำลังศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมไฟฟ้า คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ประวัติย่อผู้ทำโครงงาน

ชื่อ ชื่อสกุล	นายณัฐพล พรายวัน
วันเดือนปีเกิด	29 ธันวาคม 2537
สถานที่เกิด	อำเภอบางใหญ่ จังหวัดนนทบุรี
สถานที่อยู่ปัจจุบัน	55/1 ม.10 ต.บางแม่นาง อ.บางใหญ่
	จ.นนทบุรี 11140
หมายเลขโทรศัพท์ติดต่อ	086-907-4766

ประวัติการศึกษา

พ.ศ. 2556	มัธยมศึกษาปีที่ 6
	จากโรงเรียนมัธยมบางบัวทอง
พ.ศ. 2560	กำลังศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมไฟฟ้า
	คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ

ประวัติย่อผู้ทำโครงงาน

ชื่อ ชื่อสกุล นายปิติภัทร ตันเขมจารี

วันเดือนปีเกิด 31 พฤกษภาคม 2537

สถานที่เกิด อำเภอเมือง จังหวัดนครศรีธรรมราช

สถานที่อยู่ปัจจุบัน 1080/3 ต.คลัง อ.เมือง
จ.นครศรีธรรมราช 80000

หมายเลขโทรศัพท์ติดต่อ 088-762-0505

ประวัติการศึกษา

พ.ศ. 2556 มัธยมศึกษาปีที่ 6
จากโรงเรียนเบญจมราชูทิศ

พ.ศ. 2560 กำลังศึกษาระดับปริญญาตรี สาขาวิชาวิศวกรรมไฟฟ้า
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ