หุ่นยนต์เตะฟุตบอล

SOCCER ROBOT

โดย

นายชุติมันต์ ยงประพัฒน์

นายภัทราวุธ จันทร์เสงี่ยม

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

หุ่นยนต์เตะฟุตบอล
SOCCER ROBOT

โดย

นายชุติมันต์ ยงประพัฒน์ เลขประจำตัว 44010118

นายภัทราวุธ จันทร์เสงี่ยม เลขประจำตัว 44010360

อาจารย์ที่ปรึกษา

รศ.ดร.สุรพันธ์ เอื้อไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

ภาควิชาอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

หุ่นยนต์เตะฟุตบอล

SOCCER ROBOT

นายชุติมันต์ ยงประพัฒน์

นายภัทราวุธ จันทร์เสงี่ยม

โครงงานนี้ได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร.สุรพันธ์ เอื้อไพบูลย์)

อาจารย์ที่ปรึกษา

กิตติกรรมประกาศ

โครงงานนี้สำเร็จได้ด้วยดี เนื่องจากได้รับความอนุเคราะห์จากหลายฝ่าย อาทิ อาจารย์ที่ปรึกษา รศ.ดร.สุรพันธ์ เอื้อไพบูลย์ และ ดร.ยุทธนา คิดใจเดียว ที่ให้คำแนะนำที่ดี รวมถึงรุ่นพี่และเพื่อนๆในชุมนุมและในภาควิชา ที่คอยช่วยเหลือและเป็นกำลังใจเรื่อยมา

นายชุติมันต์ ยงประพัฒน์

นายภัทราวุธ จันทร์เสงี่ยม

ผู้จัดทำ

หุ่นยนต์เตะฟุตบอล

นายชุติมันต์ ยงประพัฒน์
นายภัทราวุธ จันทร์เสงี่ยม
รศ.ดร.สุรพันธ์ เอื้อไพบูลย์
อาจารย์ที่ปรึกษา
ภาคเรียนที่ 2 ปีการศึกษา 2547

บทคัดย่อ

วิทยานิพนธ์เล่มนี้นำเสนอการประยุกต์ใช้การประมวลผลภาพดิจิตอลจากกล้องที่ติดบนตัวหุ่นยนต์ ร่วมกับระบบปัญญาประดิษฐ์ ซึ่งใช้การพัฒนาโปรแกรมด้วยภาษา C# บนเครื่อง Pocket PC ทำการตัดสินใจในการส่งคำสั่งควบคุมหุ่นยนต์ไปยังไมโครคอนโทรลเลอร์ ซึ่งทำหน้าที่ควบคุมการทำงานหุ่นยนต์ โดยหุ่นยนต์จะต้องสามารถค้นหาลูกบอล ทำการเลี้ยงลูกบอลเข้าหาเขตประตู และยิงลูกบอลเข้าประตูได้อย่างมีประสิทธิภาพ

Soccer Robot

Mr. Chutiman Yongprapat

Mr.Pattravut Chansakhiam

Assoc. Prof. Dr. Surapan Airphaiboon

Adviser

Academic 2004

Abstract

This thesis presents the application of digital image processing technique and artificial intelligence developed by visual C# programming on pocket pc, which is installed on a soccer robot controlled by microcontrollers. The aim is to control the robot to search the ball from the image captured by a digital camera on the robot, dribble and shoot the ball to the goal efficiently.

สารบัญ

สารบัญรูปภาพ

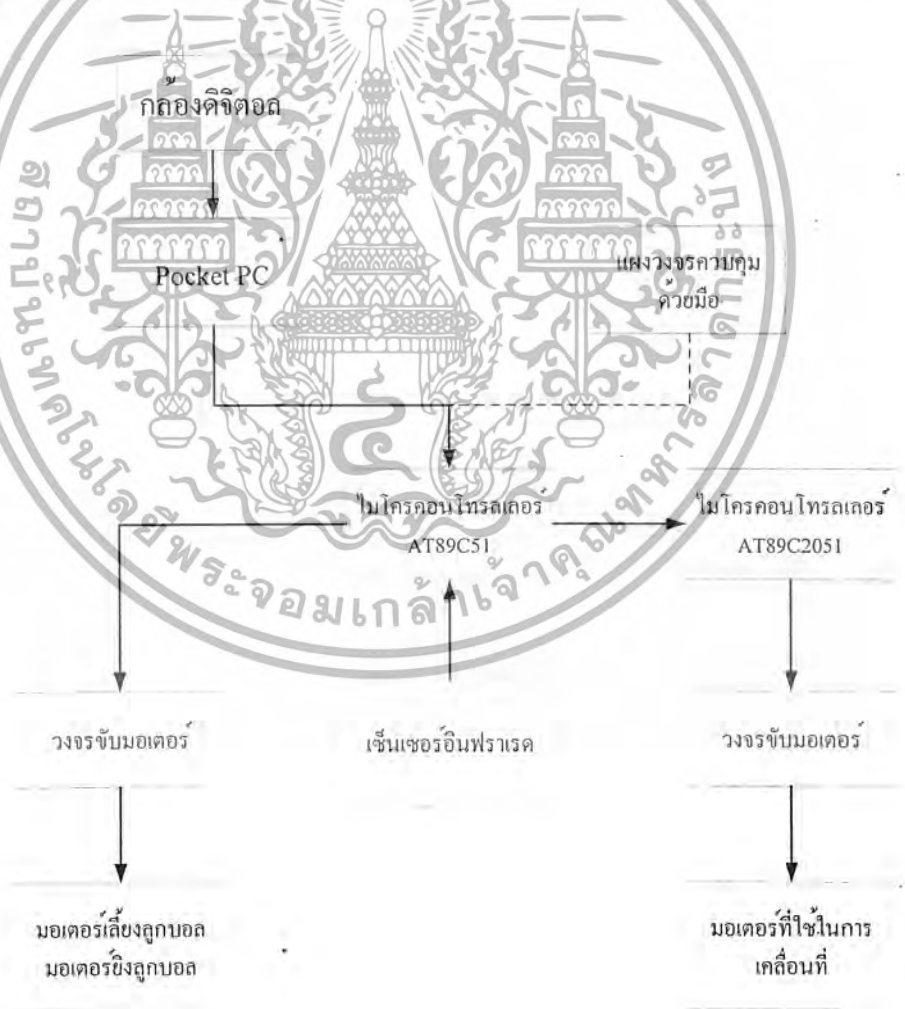
สารบัญรูปภาพ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1
บทนำ

โครงงานนี้เป็นการประยุกต์ใช้เทคนิคการประมวลผลภาพดิจิตอล (Digital Image Processing) จากกล้องที่ติดบนตัวหุ่น ทำการค้นหาตำแหน่งลูกบอล และเขตประตู จากตำแหน่งที่ได้ จะนำไปประมวลผลด้วยปัญญาประดิษฐ์ (Artificial Intelligencer) ซึ่งใช้การพัฒนาโปรแกรมด้วยภาษา C# บนเครื่อง Pocket PC ทำการตัดสินใจในการส่งคำสั่งควบคุมหุ่นยนต์ไปยังไมโครคอนโทรลเลอร์บนตัวหุ่นยนต์ ซึ่งจะไปควบคุมการทำงานของมอเตอร์ที่ใช้ในการเคลื่อนที่ เลี้ยงบอล และ ยิงบอล โดยภาพรวมหุ่นยนต์จะต้องสามารถค้นหาลูกบอล ทำการเลี้ยงลูกบอลเข้าหาเขตประตู และยิงลูกบอลเข้าประตูได้อย่างมีประสิทธิภาพ



รูปที่ 1.1 บล็อกไดอะแกรมของหุ่นยนต์เตะฟุตบอล

จากรูปที่ 1.1 แสดงบล็อกไดอะแกรมของหุ่นยนต์เตะฟุตบอล ซึ่งจะประกอบด้วยส่วนต่างๆ ดังต่อไปนี้

1. กล้องดิจิตอล ใช้กล้องดิจิตอลของ Veo Photo Traveler ซึ่งเป็นกล้องที่ใช้กับ Pocket PC ในการรับภาพ ทำการค้นหาตำแหน่งลูกบอล และเขตประตู

2. Pocket PC ใช้รุ่น iPAQ H3600 ในการประมวลผลภาพดิจิตอลและ นำไปประมวลผลด้วยปัญญาประดิษฐ์ รวมทั้งส่งคำสั่งควบคุมหุ่นยนต์ไปให้ ไมโครคอนโทรลเลอร์ เบอร์ AT89C51 ทางพอร์ตอนุกรม

3. ไมโครคอนโทรลเลอร์ เบอร์ AT89C51 ควบคุมมอเตอร์ที่ใช้เลี้ยงและยิง รวมทั้งส่งข้อมูล ซึ่งเป็นรูปแบบการเคลื่อนที่ไปให้ไมโครคอนโทรลเลอร์ เบอร์ AT89C2051

4. ไมโครคอนโทรลเลอร์ เบอร์ AT89C2051 รับข้อมูลที่ส่งมาจาก AT89C51แล้วสร้างสัญญาณพัลส์ไปควบคุมมอเตอร์ที่ใช้เคลื่อนที่

5. วงจรขับมอเตอร์ ใช้ไอซีขับกระแสเบอร์ L293D

6. มอเตอร์ ใช้มอเตอร์กระแสตรง

จากรูปที่ 1.2 แสดงบล็อกไดอะแกรมของหุ่นยนต์ผู้รักษาประตู ซึ่งจะประกอบด้วยส่วนต่างๆดังต่อไปนี้

1. กล้องWebCam ใช้กล้องWebCam รุ่น Alpha Cam Pro ในการรับภาพ

2.คอมพิวเตอร์ ใช้ในการรับภาพจากกล้องWebCam นำภาพที่ได้มาประมวลผลภาพดิจิตอลเพื่อคำนวณหาพิกัดตำแหน่งของลูกบอลและตำแหน่งที่หุ่นยนต์จะเคลื่อนที่ไป แล้วส่งคำสั่งควบคุมการเคลื่อนที่ไปให้ ไมโครคอนโทรลเลอร์ เบอร์ AT89C2051 ทางพอร์ตอนุกรมบนตัวหุ่น

3. ไมโครคอนโทรลเลอร์ เบอร์ AT89C2051 รับข้อมูลที่ส่งมาจากคอมพิวเตอร์แล้วสร้างสัญญาณพัลส์ไปควบคุมมอเตอร์ที่ใช้เคลื่อนที่

4. วงจรขับมอเตอร์ ใช้ไอซีขับกระแสเบอร์ L293D

5. มอเตอร์ ใช้มอเตอร์กระแสตรง

รูปที่ 1.2 บล็อกไดอะแกรมของหุ่นยนต์ผู้รักษาประตู

วัตถุประสงค์

1. เพื่อศึกษาการประมวลผลภาพดิจิตอล
2. เพื่อศึกษาการพัฒนาโปรแกรมบนเครื่อง Pocket PC
3. เพื่อศึกษาระบบปัญญาประดิษฐ์

ขอบเขตของโครงงาน

1. หุ่นยนต์สามารถนำภาพจากกล้อง มาทำการหาตำแหน่งของลูกบอล และเคลื่อนที่เข้าหาลูกบอลได้

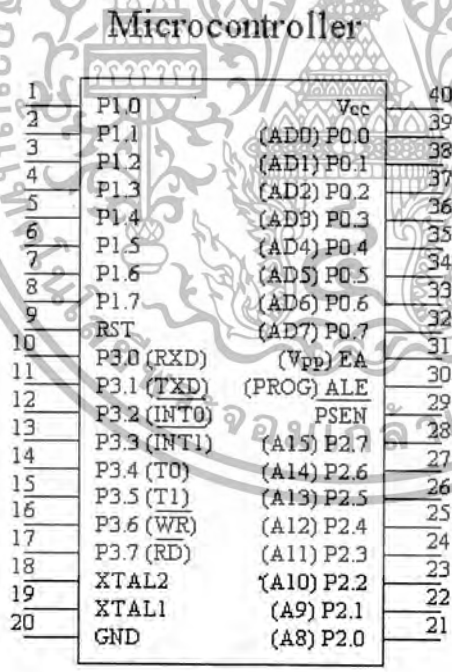
2. เมื่อหุ่นยนต์สามารถครอบครองลูกบอลได้แล้ว จะทำการหาเขตประตู และยิงลูกบอลเข้าประตูได้

3. หุ่นยนต์ผู้รักษาประตูสามารถป้องกันประตูได้

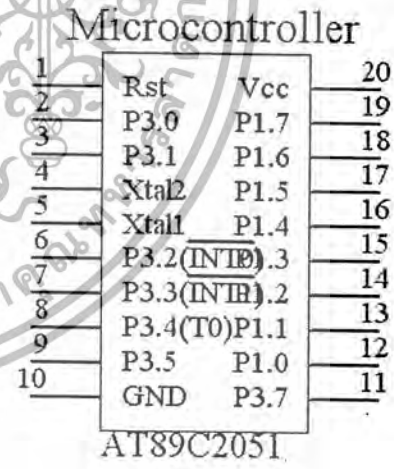
บทที่ 2
ทฤษฎี

2.1 ไมโครคอนโทรลเลอร์

ไมโครคอนโทรลเลอร์(microcontroller) เป็นชื่อของอุปกรณ์อิเล็กทรอนิกส์ที่มีความสามารถมากมาย ไมโครคอนโทรลเลอร์ประกอบไปด้วย หน่วยประมวลผล หน่วยคำนวณทางคณิตศาสตร์และลอจิก วงจรรับสัญญาณอินพุท วงจรขับสัญญาณทางเอาท์พุท หน่วยความจำ และ วงจรกำเนิดสัญญาณนาฬิกา ซึ่งนำไปประยุกต์ใช้งานทางอิเล็กทรอนิกส์ได้มากมาย โดยที่สามารถเลือกจากความต้องการสร้างระบบควบคุมขนาด 8 บิต มีการเชื่อมต่ออุปกรณ์ภายนอกไม่มากนัก โดยในโครงงานนี้ใช้ไมโครคอนโทรลเลอร์ 2 เบอร์ได้แก่ AT89C51 และ AT89C2051 ซึ่งแต่ละเบอร์มีขาแสดงดังรูปที่ 2.1 และรูปที่ 2.2 ตามลำดับ



รูปที่ 2.1 ขาของ AT89C51



รูปที่ 2.2 ขาของ AT89C51

ไมโครคอนโทรลเลอร์ เบอร์ AT89C51 เป็นไอซีที่มี 40 ขา มีพอร์ทอินพุท/เอาท์พุท 4 พอร์ท ส่วนเบอร์ AT89C2051 เป็นไอซีที่มี 20 ขา มีพอร์ทให้เลือกใช้งานเพียง 2 พอร์ท นอกจากนี้ไมโครคอนโทรลเลอร์ เบอร์ AT89C51 ยังมีหน่วยความจำมากกว่าด้วย

2.1.1 คุณสมบัติทางเทคนิคของไมโครคอนโทรลเลอร์ MCS-51

- เป็นไมโครคอนโทรลเลอร์ที่ใช้ซีพียูขนาด 8 บิต
- ภายในมีหน่วยความจำโปรแกรมเป็นแบบแฟลช
- หน่วยความจำข้อมูลพื้นฐานเป็นหน่วยความจำแบบแรม และ บางเบอร์จะมีหน่วยความจำแบบอีอีพรอมเพิ่มเติม
- ขาพอร์ตเป็นแบบ 2 ทิศทาง สามารถใช้งานได้ทั้งอินพุทและเอาท์พุท
- มีวงจรสื่อสารอนุกรมแบบฟูลดูเพล็กซ์
- ไทเมอร์/เคาท์เตอร์ขนาด 16 บิตอย่างน้อย 2 ตัว
- สามารถรองรับแหล่งกำเนิดอินเตอร์รัพต์ได้ 5 หรือ 6 ประเภท
- สามารถขยายหน่วยความจำภายนอกเพิ่มเติมได้สูงสุด 64 กิโลไบต์
- มีวงจรกำเนิดสัญญาณนาฬิกาภายในชิป

2.1.2 การจัดหน่วยความจำของไมโครคอนโทรลเลอร์ MCS-51 แบบแฟลช

ในไมโครคอนโทรลเลอร์มีหน่วยความจำภายในหลักๆอยู่ 2 ส่วน คือ หน่วยความจำโปรแกรมและหน่วยความจำข้อมูล ซึ่งก็มีขนาดและการจัดสรรแตกต่างกันไปในแต่ละเบอร์

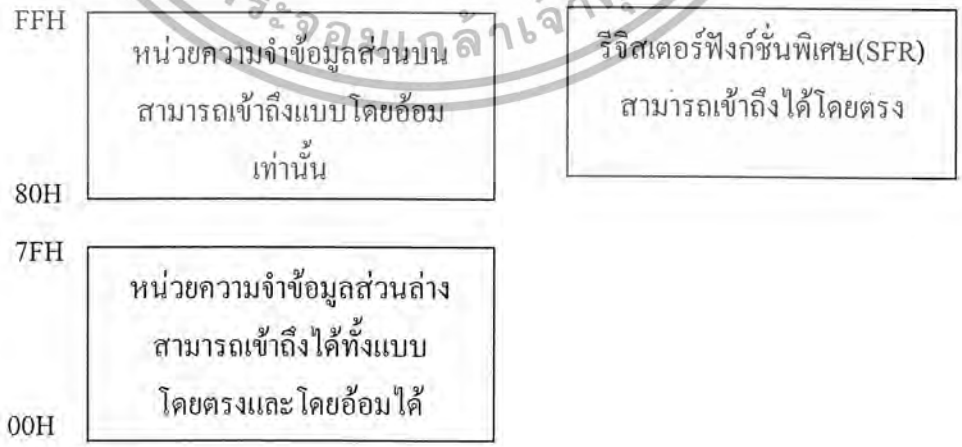
1. หน่วยความจำโปรแกรม ไมโครคอนโทรลเลอร์เบอร์ AT89C51 มีการจัดสรรหน่วยความจำโปรแกรมดังนี้

รูปที่ 2.3 การจัดสรรหน่วยความจำโปรแกรมของAT89C51

ไมโครคอนโทรลเลอร์เบอร์ AT89C2051 มีการจัดสรรหน่วยความจำโปรแกรมดังนี้

รูปที่ 2.4 การจัดสรรหน่วยความจำโปรแกรมของAT89C2051

2. หน่วยความจำข้อมูล ทุกเบอร์จะมีหน่วยความจำข้อมูลภายในเป็นแบบแรม ซึ่งแต่ละเบอร์จะมีขนาดแตกต่างกันไปโดย AT89C51 และ AT89C2051 จะมีขนาด 128 ไบต์ ส่วน AT89C52 จะมีขนาด 256 ไบต์ สำหรับการจัดสรรหน่วยความจำข้อมูลภายในแบ่งเป็น 3 ส่วน ดังนี้



รูปที่2.5 การจัดสรรหน่วยความจำข้อมูลของAT89C51 และ AT89C2051

สำหรับหน่วยความจำข้อมูลภายนอก ไมโครคอนโทรลเลอร์จะติดต่อได้สูงสุด 64 กิโลไบต์ จะเห็นว่าหน่วยความจำข้อมูลส่วนบนและรีจิสเตอร์ฟังก์ชั่นพิเศษมีตำแหน่งทับซ้อนกัน ซึ่งจะมีการติดต่อแตกต่างกัน

2.1.3 การรับส่งข้อมูลผ่านพอร์ตอนุกรม

พอร์ตสื่อสารอนุกรมมีโครงสร้างการทำงานในแบบที่เรียกว่าฟูลดูเพล็กซ์(Full Duplex) คือ สามารถรับและส่งข้อมูลอนุกรมในเวลาเดียวกัน โดยทางด้านส่งใช้ขา TxD (พอร์ต 3.1) ทางด้านรับใช้ขา RxD (พอร์ต 3.0) SBUF ใช้เป็นบัฟเฟอร์สำหรับรับและส่งข้อมูลแบบอนุกรม

พอร์ตสื่อสารอนุกรมสามารถโปรแกรมการทำงานได้หลายโหมดด้วยกันโดยเลือกที่บิต SM1 และ SM0 การทำงานของพอร์ตสื่อสารอนุกรมมี 4 โหมด ดังนี้

1. โหมด 0 : พอร์ตสื่อสารอนุกรม 8 บิต โดยการส่งจะเลื่อนออกทีละบิต โดยส่งบิต D0 ออกไปก่อนทางขา RxD และไม่มีการส่งบิตเริ่มต้น(start bit) แต่จะส่ง shift clock ทางขา TxD (ความเร็ว1/12 เท่าของซีพียู)

2. โหมด 1 : พอร์ตสื่อสารอนุกรม 10 บิต ข้อมูล 8 บิต 1 start bit และ 1 stop bit และสามารถเปลี่ยนแปลงความเร็วในการส่งข้อมูลได้ โดยขึ้นกับบิตSMOD ใน TCON และอัตราโอเวอร์โฟว์ของไทเมอร์1

$$\text{Baud rate mode 1} = \frac{2^{SMOD} \times oscilatorFrequency}{32 \times 12 \times [256 - (TH1)]}$$ โดยใช้ไทเมอร์ 1 (สมการที่2.1)

$$\text{Baud rate mode 1} = \frac{oscilatorFrequency}{32 \times [65536 - (RCAP2H, RCAP2L)]}$$ โดยใช้ไทเมอร์ 2 (สมการที่2.2)

3. โหมด 2 : พอร์ตสื่อสารอนุกรม 11 บิต ข้อมูล 9 บิต 1 start bit และ 1 stop bit โดยบิตข้อมูลที่เพิ่มขึ้นมา 1 บิตนั้นคือบิตพาริตี้(parity bit) ซึ่งเป็นบิตที่ใช้ในการตรวจสอบความถูกต้องของข้อมูลที่ส่ง โดยจะแบ่งออกเป็นการส่งแบบบิตพาริตี้คู่กับบิตพาริตี้คี่

การส่งแบบบิตพาริตี้คู่ คือ จำนวนในข้อมูล 8 บิตมีบิตที่เป็น 1 อยู่เป็นจำนวนคู่ บิต พาริตี้จะมีค่าเป็นลอจิก 0 ถ้าในข้อมูล มีบิตที่เป็น 1 อยู่เป็นจำนวนคี่ บิตพาริตี้จะมีค่าเป็นลอจิก 1

การส่งแบบบิตพาริตี้คี่ คือ ข้อมูล 8 บิตมีจำนวนข้อมูลที่เป็น 1 อยู่เป็นจำนวนคี่ บิตพาริตี้จะมีค่าเป็นลอจิก 0 ถ้ามีข้อมูลที่เป็น 1 อยู่เป็นจำนวนคู่ บิตพาริตี้จะมีค่าเป็นลอจิก 1

ความเร็วในการรับส่งข้อมูลเท่ากับ 1/32 และ 1/64 ของความเร็วสัญญาณนาฬิกาของซีพียู โดยขึ้นกับบิต SMOD ใน TCON

-Baud rate mode 2 = (1/32)x ความถี่ของออสซิเลเตอร์ เมื่อ SMOD = 1 (สมการที่2.3)

-Baud rate mode 2 = (1/64)x ความถี่ของออสซิเลเตอร์ เมื่อ SMOD = 0 (สมการที่2.4)

4. โหมด 3 : พอร์ตสื่อสารแบบ 11 บิต UART โดยส่งข้อมูล 9 บิต 1 start bit และ 1 stop bit เหมือนโหมด 2 แต่อัตราเร็วจะขึ้นอยู่กับบิต SMOD ใน TCON และอัตราโอเวอร์โฟว์ของไทเมอร์1 และไทเมอร์ 2 สำหรับ 8051 หรือ อัตราโอเวอร์โฟว์ของไทเมอร์2 สำหรับ80C154D

$$\text{Baud rate mode 3} = \frac{2^{SMOD} \times oscilatorFrequency}{32 \times 12 \times [256 - (TH1)]}$$ โดยใช้ไทเมอร์ 1 (สมการที่2.5)

$$\text{Baud rate mode 1} = \frac{oscilatorFrequency}{32 \times [65536 - (RCAP2H, RCAP2L)]}$$ โดยใช้ไทเมอร์ 2 (สมการที่2.6)

รีจิสเตอร์ควบคุมการทำงานของพอร์ตอนุกรม หรือ SCON (Serial port Control Register)

SCON เป็นรีจิสเตอร์ที่ใช้ควบคุมการทำงานของพอร์ตอนุกรม ซึ่งประกอบด้วยบิตต่างๆ 8 บิตดังนี้

บิต7	บิต 6	บิต 5	บิต 4	บิต3	บิต 2	บิต 1	บิต 0
SME/FE	SM1	SM2	REN	TB8	RB8	TI	RI

SME/FE (Serial port mode bit 0 /Framing error bit) : ปกติจะใช้ร่วมกับบิต SM1 เพื่อกำหนดโหมดการทำงานของพอร์ตอนุกรม การเข้าถึงบิตนี้จะเกิดขึ้นได้ก็ต่อเมื่อเมื่อการเคลียร์บิต SMOD ซึ่งก็คือ บิต 6 ของรีจิสเตอร์ PCON ในกรณีที่ใช้ความสามารถในการตรวจจับข้อผิดพลาดของเฟรมบิตข้อมูลนี้ บิตนี้จะแจ้งความผิดพลาดโดยจะเซตเป็น 1 ทันทีเมื่อพบว่าไม่สามารถตรวจจับบิตหยุด(stop bit)ของข้อมูลของพอร์ตอนุกรมได้

SM1 (Serial port mode bit 1) ใช้ร่วมกับบิต SM0 ในการกำหนดโหมดการทำงานของพอร์ตอนุกรม ดังรายละเอียดในตารางที่ 2.1

ตารางที่ 2.1 การทำงานของพอร์ตอนุกรม

SM0	SM1	โหมด	รายละเอียด	อัตราบอด(Baud rate)
0	0	0	ชิฟรีจิสเตอร์	ความถี่สัญญาณนาฬิกา/6
0	1	1	UART 8 บิต	ปรับค่าได้
1	0	2	UART 9 บิต	ความถี่สัญญาณนาฬิกา/32
1	1	3	UART 9 บิต	ปรับค่าได้

SM2 (Serial port mode bit 2) ใช้ในเอ็นนาเบิลความสามารถในการรับรู้แอดเดรสในการติดต่ออัตโนมัติ เมื่อมีการเชื่อมต่อไมโครคอนโทรลเลอร์หลายตัวเข้าด้วยกัน โดยความสามารถนี้ใช้ได้ก็ต่อเมื่อทำงานในโหมด 2 หรือ 3

REN (Receive enable bit) ใช้เอ็นนาเบิลความสามารถในการรับข้อมูลของพอร์ตอนุกรม

TB8(Transmit data bit 8) ใช้เก็บบิตที่ 8 หรือบิตที่ 9 ที่ต้องการส่งออกในพอร์ตอนุกรม เมื่อทำงานในโหมด 2 และ โหมด 3

RB8(Receive data bit 8) ใช้เก็บข้อมูลบิต 8 ของข้อมูลที่รับเข้ามาในพอร์ตอนุกรม เมื่อทำงานในโหมด 2 และ โหมด 3

TI (Transmit interrupt flag) แสดงการเกิดอินเตอร์รัพต์จากการส่งข้อมูลออกทางพอร์ตอนุกรม

RI (Receive interrupt flag) บิตแสดงการเกิดอินเตอร์รัพต์จากการรับข้อมูลเข้ามาของพอร์ตอนุกรม

2.2 การพัฒนาโปรแกรมด้วยภาษา C#

2.2.1 Microsoft .NET

Microsoft .NET เป็นแพลตฟอร์มที่บริษัทไมโครซอฟท์ ได้พัฒนาขึ้น เป็นแพลตฟอร์มที่จะผสมผสานคอมพิวเตอร์ ให้เข้ากับอินเทอร์เน็ตได้อย่างสมบูรณ์ที่สุด เป็นระบบที่เอื้อต่อการพัฒนาโปรแกรมยุคใหม่ ทำให้นักพัฒนา สามารถสร้างโปรแกรมที่ทำงานทางด้านอินเทอร์เน็ต ฐานข้อมูลได้ง่ายขึ้น รวมทั้งมีข้อดีในด้านการทำงานข้ามภาษา ซึ่งจะทำให้โปรแกรมไม่ถูกจำกัดด้วยภาษาใดภาษาเดียว หรือระบบใดระบบเดียวอีกต่อไป

2.2.2 ภาษา C#

ภาษา C# เป็นภาษาใหม่ที่เกิดขึ้นจากการพัฒนา Microsoft.NET โดยได้รวบรวมเอาจุดเด่นและความสามารถของภาษาดั้งเดิมหลายภาษา (เช่น VB VC++) เข้าด้วยกัน ลักษณะเฉพาะของภาษา C# มีดังนี้

- เขียนง่าย ด้วยประสิทธิภาพของโปรแกรม Visual Studio .NET และรูปแบบการเขียนโปรแกรมแบบ Visual Programming เช่นเดียวกับภาษา Visual Basic หรือ Delphi
- โครงสร้างภาษาเป็นแบบ Object อย่างสมบูรณ์แบบ
- มีประสิทธิภาพสูง สามารถทำงานภายใต้เทคโนโลยี .NET ได้อย่างสมบูรณ์ รวมทั้งสามารถใช้งาน Base Class ต่างๆ ของ .NET Framework อีกด้วย
- สามารถทำงานระดับล็ก ซึ่งเป็นคุณสมบัติเด่นที่มีอยู่ในภาษา C และ C++
- สามารถทำงานแบบ multi-thread ซึ่งช่วยให้โปรแกรมสามารถทำงานหลายอย่างได้ในเวลาเดียวกัน

จะเห็นว่าภาษา C# เป็นภาษาที่เขียนง่าย เหมือนภาษา Visual Basic แต่สามารถทำงานระดับล็กได้เช่นเดียวกับภาษา C หรือ C++ ประกอบกับการพัฒนาของ Visual Studio .NET ทำให้การพัฒนาโปรแกรมมีความสะดวกมากขึ้น นอกจากนี้ ภาษา C# ยังสามารถใช้พัฒนาโปรแกรมบน Smart Devices (ได้แก่ Pocket PC, โทรศัพท์มือถือ ฯลฯ) ได้อีกด้วย

2.3 Pocket PC

Pocket PC มีองค์ประกอบคล้ายกับพีซี คือ ประกอบด้วยฮาร์ดแวร์ ระบบปฏิบัติการ ซอฟแวร์ และมีอุปกรณ์ต่อพ่วงหรืออุปกรณ์เสริมต่างๆได้อย่างมากมาย

คุณสมบัติพื้นฐานขั้นต่ำทางด้านฮาร์ดแวร์ของเครื่อง PDA ที่จะมาเป็น Pocket PC คือ

1.หน่วยความจำขั้นต่ำรอม (ROM) และแรม (RAM) อย่างละ 16 เมกกะไบต์

2.พอร์ตอินฟราเรด

3.ไมโครโฟนและลำโพง

4. I/O พอร์ตเชื่อมต่อกับพีซีเป็นแบบอนุกรมหรือ USB

5.ความละเอียดของจอภาพ 320 x240 พิกเซล

2.3.1 องค์ประกอบของ Pocket PC

1.ซีพียูหรือโปรเซสเซอร์ ซีพียูของ Pocket PC จะมีหลายตระกูล คือ MIPS,SH3 และ StrongArm(หรือ ARM) โดยมีความเร็วต่ำสุดอยู่ที่ 70 MHz และสูงสุดอยู่ที่ 206 MHz

ข้อเสียของการมีซีพียูหลายตระกูลใน Pocket PC คือ ทำให้ผู้พัฒนาโปรแกรมสำหรับ Pocket PC จะต้องทำการคอมไพล์โปรแกรมของซีพียูแต่ละตระกูลเอาไว้ให้ผู้ใช้เลือกดาวน์โหลดไปติดตั้ง ซึ่งผู้ใช้จะต้องเลือกให้ถูกต้องตามรุ่นที่ใช้อยู่ ดังนั้นตั้งแต่ Pocket PC 2002 เป็นต้นไป ระบบปฏิบัติการ Window CE จึงสนับสนุนเฉพาะซีพียูตระกูล StrongArm ของ Intel เท่านั้น ซึ่งส่งผลดีต่อผู้พัฒนาโปรแกรมและผู้ใช้

2.หน่วยความจำ หน่วยความจำของ Pocket PC แบ่งออกเป็น 2 ส่วน คือ รอม (Read only memory : ROM) และ แรม (Read and Access memory : RAM) ซึ่งหน่วยความจำของรอมและแรมขั้นต่ำคือ 16 เมกกะไบต์

รอม หน่วยความจำส่วนที่เป็นรอมจะใช้ระบบปฏิบัติการ Window CE และ โปรแกรมพื้นฐานของ Pocket PC ซึ่งระบบปฏิบัติการและ โปรแกรมพื้นฐานที่เก็บอยู่ในรอมจะไม่สูญหายไปเมื่อแบตเตอรี่หมด รอมใน Pocket PC แบ่งออกเป็น 2 ประเภท คือ รอมแบบธรรมดา กับ รอมแบบ Flash รอมแบบFlash ผู้ใช้จะสามารถแก้ไขหรืออัพเกรดระบบปฏิบัติการของ Pocket PC ได้

แรม หน่วยความจำส่วนที่เป็นแรม จะใช้เป็นที่เก็บโปรแกรมและซอฟแวร์ต่างๆที่ผู้ใช้ติดตั้งเข้าไป รวมทั้งไฟล์ข้อมูลต่างๆระหว่างการทำงาน ข้อมูลในแรมจะอยู่ได้จะต้องมีไฟเลี้ยงตลอด ดังนั้นเมื่อแบตเตอรี่ของ Pocket PC หมด โปรแกรมและข้อมูลที่อยู่ในแรมจะหายไป

3.แบตเตอรี่ แหล่งจ่ายไฟหลักของ Pocket PC คือ แบตเตอรี่ ซึ่งมีอยู่ 2 ส่วนคือ แบตเตอรี่หลักและแบตเตอรี่สำรอง โดยแบตเตอรี่หลักจะจ่ายไฟเลี้ยงแก่ Pocket PC ในสภาวะการทำงานปกติและเมื่อแบตเตอรี่หลักหมด แบตเตอรี่สำรองก็จะจ่ายไฟเลี้ยงชั่วคราวแทน เพื่อไม่ให้ข้อมูลใน Pocket PC สูญหาย ซึ่งแบตเตอรี่ทั้งสองเป็นแบบชาร์จหรือประจุไฟได้

4.จอภาพ จอภาพของ Pocket PCจะมีความละเอียดขั้นต่ำอยู่ที่ 320 x 240 พิกเซล และเป็นจอสี Pocket PC 2002 ส่วนใหญ่จะมีจอภาพ LCD ที่ใช้เทคโนโลยี Reflective TFT ในบางรุ่นจะมีตัวตรวจจับแสงเพื่อวัดระดับแสงแล้วนำไปปรับความสว่างของจอภาพโดยอัตโนมัติ Pocket PC ส่วนใหญ่ในปัจจุบันสามารถแสดงสีได้ในระดับ 16 บิต หรือ 65,536 สี

5.ปุ่มสำหรับการทำงานต่างๆ บริเวณด้านล่างของ Pocket PC จะมีปุ่มสำหรับกดเพื่อเรียกโปรแกรมต่างๆขึ้นมาทำงานจำนวน 4 ปุ่ม ซึ่งเราสามารถกำหนดได้ว่าเมื่อแต่ละปุ่มถูกกดแล้วจะเรียกโปรแกรมใดขึ้นมาทำงาน ส่วนปุ่มตรงกลางจะทำหน้าที่เหมือนปุ่มลูกศร

6.อุปกรณ์เสริมอื่นๆ Pocket PC สามารถเชื่อมต่อกับอุปกรณ์ภายนอกได้หลายอย่างทั้งที่ออกแบบมาสำหรับ Pocket PC โดยเฉพาะ หรืออุปกรณ์บางอย่างที่กำลังจะกลายเป็นมาตรฐานของอุปกรณ์ดิจิตอล เช่น การ์ด CF , MMC/SDแบบต่างๆคือ หน่วยความจำ โมเด็มและกล้องดิจิตอล

2.4 การประมวลผลภาพดิจิตอล

การประมวลผลภาพดิจิตอล คือการนำภาพดิจิตอล มากระทำการใดๆ เพื่อบรรลุวัตถุประสงค์ คือ เพื่อให้สามารถอธิบายความหมายของภาพได้ดีขึ้น นอกจากนี้ ยังรวมถึง การเก็บบันทึกภาพ(Storage) และการนำภาพออกมาแสดง (Representation) ดังนั้น การประมวลผลภาพ จึงคลอบคลุมเทคนิคการประมวลผลภาพต่างๆ ได้แก่ การปรับปรุงภาพ (Enhancement) การแยกส่วนภาพ (Segmentation) การหาวัตถุในภาพ (Object Recognition) การกำจัดสิ่งรบกวนในภาพ (Restoration) การบีบอัดภาพ (Compression) เป็นต้น

ในโครงงานนี้ ได้นำภาพจากกล้อง ซึ่งเป็นภาพสี มาทำการประมวลผล เพื่อหาตำแหน่งของลูกบอล เทคนิคที่ใช้ในการประมวลผล จึงประกอบไปด้วย การแยกส่วนภาพ เพื่อแยกจุดที่เป็นลูกบอล ออกจากสนามหรือวัตถุอื่น จากนั้นจึงนำส่วนที่แยกได้ มาคำนวณหาตำแหน่งจุดศูนย์กลางของลูกบอล โดยใช้เทคนิคการหาค่าเฉลี่ยของตำแหน่งที่แยกได้ทั้งหมด ดังนี้

2.4.1 การแยกส่วนภาพในพิกัด RGB (Segmentation in RGB Vector Space)

การแยกส่วนภาพในพิกัด RGB (Segmentation in RGB Vector Space) เป็นการแยกส่วนที่มีเฉพาะสีที่เราต้องการออกมาโดยมีองค์ประกอบของสีเป็น RGB จุดประสงค์ของการแยกส่วนคือการแยกสีแต่ละพิกเซลว่ามีสีที่อยู่ในขอบเขตดังกล่าวหรือไม่ โดยขอบเขตดังกล่าวอาจเป็นลูกบาศก์ (Hypercube) ทรงกลม (Sphere) หรือเป็นทรงรี (Ellipse) ก็ได้

ถ้าเรามีเซตของสีของรูปที่เราต้องการ เราจะได้ค่าเฉลี่ยของสีนั้นออกมา ซึ่งเราสามารถเขียนแทนด้วยเวกเตอร์ a ซึ่งมี 3 องค์ประกอบคือ R, G และ B ให้ Z เป็นจุดที่อยู่ในระนาบ RGB เราจะกล่าวว่า Z คล้ายกับ a เมื่อระยะทางระหว่าง Z กับ a น้อยกว่าค่าขีดเริ่ม (Threshold),D0 โดยระยะทางระหว่าง Z กับ a เขียนแทนด้วย

$$D(z,a) = \|z-a\|$$
$$= \left[(z-a)^T (z-a)\right]^{\frac{1}{2}} \qquad \text{(สมการที่ 2.7)}$$
$$= \left[(z-a_R)^2 + (z-a_G)^2 + (z-a_B)^2\right]^{\frac{1}{2}}$$

โดย a_R, a_G, a_B เป็นองค์ประกอบของเวกเตอร์ a ในแกน R ,Gและ B ตามลำดับ ซึ่งสมการที่ 2.7 นี้เป็นการใช้ขอบเขตเป็นทรงกลม ถ้าจุดใดมีค่า $D(z,a) \leq D_0$ จุดนั้นก็จะอยู่ในทรงกลมที่มีรัศมี D_0 ซึ่งเป็นสีเราต้องการ โดยสมการที่ 2.7 นั้นมาจากสมการที่ 2.8

$$D(z,a) = \left[(z-a)^T C^{-1} (z-a)\right]^{\frac{1}{2}} \qquad \text{(สมการที่ 2.8)}$$

เมื่อ C เป็น Covariance matrix ของสีที่เราต้องการแยกส่วนซึ่งถ้าขอบเขตนั้นเป็นทรงรี เราจะได้ว่า C = I และ C^{-1}=C=I ซึ่งเราจะได้เป็นสมการที่ 2.7 นั่นเอง ถ้าเราใช้การประมาณว่าทรงรีนั้นคือทรงสี่เหลี่ยมที่มีจุดศูนย์กลางอยู่ที่ a และมีระยะทางระหว่างแกนองค์ประกอบสีแต่ละแกนเป็นสัดส่วนกับค่าเบี่ยงเบนมาตรฐานของสีนั้น ซึ่งการประมาณโดยใช้ทรงสี่เหลี่ยมจะง่ายกว่าการใช้ทรงรีและทรงกลมมาก

2.4.2 การประมวลผลฮิตโทแกรม

ฮิตโทแกรมของภาพดิจิตอลที่มีระดับสีเทา(gray level)อยู่ในช่วง [0,L-1] คือ ดิสตรีตฟังก์ชัน $h(r_k) = n_k$ เมื่อ r_k คือระดับสีเทาลำดับที่ k และ n_k คือ จำนวนของพิกเซลในภาพที่มีระดับสีเทาเป็น r_k ถ้าเราทำการนอร์มอลไลซ์ฮิตโทแกรมโดยการนำจำนวนพิกเซลทั้งหมดของภาพมาหาร เราจะได้นอร์มอลไลซ์ฮิตโทแกรมเป็น $p(r_k)=n_k/n$ เมื่อ k = 0,1,2,3,…,L-1 เราจะเรียก $p(r_k)$ ว่าเป็นความน่าจะเป็นที่จะมีระดับสีเทาเป็น r_k สังเกตว่าผลรวมของนอร์มอลไลซ์ฮิตโทแกรมจะมีค่าเท่ากับ 1 เสมอ

การทำฮิตโทแกรมเป็นเทคนิคพื้นฐานของการประมวลผลภาพในโดเมนสปาเชียล(Spatial domain) ซึ่งสามารถใช้ในการปรับปรับภาพได้อย่างมีประสิทธิภาพ นอกจากนั้นยังสามารถประยุกต์ใช้ได้กับการบีบอัดภาพและ การแยกส่วนภาพ การทำฮิตโทแกรมนั้นง่ายต่อการคำนวณโดยใช้ซอฟแวร์ ทำให้สามารถใช้เป็นเครื่องมือสำหรับการประมวลผลภาพในเวลาจริงได้

ภาพที่มีสีมืดจะมีองค์ประกอบของอิสโทแกรมหนาแน่นในช่วงระดับสีเทาที่มีความเข้มต่ำ ส่วนที่มีสีสว่างจะมีองค์ประกอบของอิสโทแกรมหนาแน่นในช่วงระดับสีเทาที่มีความเข้มสูง ภาพที่มีคอนทราสต่ำจะมีฮิตโทแกรมอยู่ในช่วงแคบๆและจะอยู่ตรงกลางของระดับสีเทา ส่วนภาพที่มีคอนทราสสูงจะมีฮิตโทแกรมอยู่ในช่วงกว้างตลอดทั้งระดับสีเทา ดังนั้นถ้าเราทราบฮิตโทแกรมของภาพก็จะปรับปรุงภาพให้ดีขึ้นได้

2.4.3 การหาตำแหน่งจุดศูนย์กลางของวัตถุ

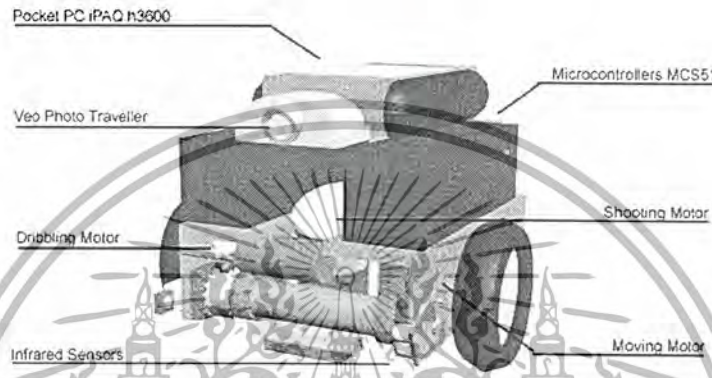
ใช้เทคนิคการนำตำแหน่งของแต่ละจุดที่ถูกพิจารณาว่าเป็นวัตถุจากการแยกส่วนภาพ ตามสมการ

$$x_{av} = \frac{\sum_{i=1}^{N} x_i}{N}$$

$$y_{av} = \frac{\sum_{i=1}^{N} y_i}{N} \qquad \text{(สมการที่ 2.9)}$$

บทที่ 3
การออกแบบฮาร์ดแวร์

3.1 หุ่นยนต์เตะฟุตบอล



รูปที่ 3.1 โครงสร้างของหุ่นยนต์เตะฟุตบอล

โครงสร้างของหุ่นยนต์เตะฟุตบอลแสดงดังรูปที่ 3.1 โดยมีหลักการทำงานคือ นำภาพจากกล้องที่ติดบนตัวหุ่น มาทำการการประมวลผลภาพดิจิตอล (Digital Image Processing) เพื่อค้นหาตำแหน่งลูกบอล และเขตประตู จากตำแหน่งที่ได้ จะนำไปประมวลผลด้วยปัญญาประดิษฐ์ (Artificial Intelligencer) ซึ่งใช้การพัฒนาโปรแกรมด้วยภาษา C# บนเครื่อง Pocket PC ทำการตัดสินใจในการส่งคำสั่งควบคุมหุ่นยนต์ไปยังไมโครคอนโทรลเลอร์บนตัวหุ่นยนต์ ซึ่งจะไปควบคุมการทำงานของมอเตอร์ที่ใช้ในการเคลื่อนที่ เลี้ยงบอล และ ยิงบอล ฮาร์ดแวร์ของหุ่นยนต์เตะฟุตบอลประกอบด้วยส่วนต่างๆดังนี้

3.1.1 กล้องดิจิตอล

รูปที่3.2 กล้องดิจิตอล Veo Photo Traveler for Pocket PC

กล้องที่ใช้ในการจับภาพเป็นกล้องดิจิตอล Veo Photo Traveler สำหรับใช้กับ Pocket PC ดังรูปที่ 3.2 โดยใช้การเชื่อมต่อทางช่องสำหรับเสียบการ์ดหน่วยความจำ CF (Compact Flash) มีส่วนประกอบที่สำคัญดังรูปที่ 3.3

คุณสมบัติที่สำคัญของกล้องดิจิตอล Veo Photo Traveler

1.ขนาดภาพที่สามารถจับได้สูงสุดคือ 640 x 480 พิกเซล

2.ภาพที่จับได้เป็น ชนิด standard JPEG

3.สามารถถ่ายภาพนิ่งและถ่ายวีดีโอได้

รูปที่ 3.3 ส่วนประกอบของกล้องดิจิตอล Veo Photo Traveler

3.1.2 Pocket PC

ในโครงงานนี้เลือกใช้ Pocket PC เป็นส่วนประมวลผล เนื่องจากมีจอภาพที่ใช้แสดงผลภาพในตัว มีขนาดเล็ก สามารถติดตั้งบนตัวหุ่นได้ สามารถเชื่อมต่อกับกล้องดิจิตอลได้ และมีแหล่งจ่ายไฟเป็นแบตเตอรี่อยู่ภายใน ทำให้หุ่นยนต์ สามารถทำงานได้โดยไม่ต้องมีการต่อพ่วง หรือเป็นระบบแยกตัว (Stand alone system)

แต่การใช้ Pocket PC ก็มีข้อจำกัดคือ การมีความเร็วในการประมวลผลต่ำ เมื่อเทียบกับเครื่องพีซี ทำให้เทคนิคการประมวลผลที่เลือกใช้ ต้องคำนึงถึงความเร็วเป็นหลัก

Pocket PC ที่ใช้ เป็น Pocket PC รุ่น iPAQ H3600 ดังรูป 3.4 ซึ่งมีคุณสมบัติดังนี้

1. ซีพียูที่ใช้เป็นซีพียูรุ่น StrongArm ความเร็ว 206 MHz
2. หน่วยความจำรอม (ROM) และแรม (RAM) แบบ Flash อย่างละ 32 เมกกะไบต์
3. จอภาพของ Pocket PC ความละเอียดของจอภาพ 320 x240 พิกเซล โดยเป็นจอสี LCD
4. มีพอร์ตอนุกรม RS 232, IrDA: FIR/SIR และพอร์ต USB
5. มีไมโครโฟนและลำโพง

รูปที่ 3.4 Pocket PC รุ่น iPAQ H3600

พอร์ตอนุกรมของ Pocket PCรุ่น iPAQ H3600 มีทั้งหมด 4 พอร์ต คือ พอร์ต com1, com2, com3 และ com4 ในโครงงานนี้ได้ใช้พอร์ต com1 ซึ่งเป็นพอร์ตอนุกรม RS232 มีการเชื่อมต่ออยู่ที่ cradle connector ที่บริเวณด้านล่างของPocket PC สำหรับ cradle connector นี้มีขาทั้งสิ้น 15 ขาดัง ตารางที่ 3.1 ดังนี้

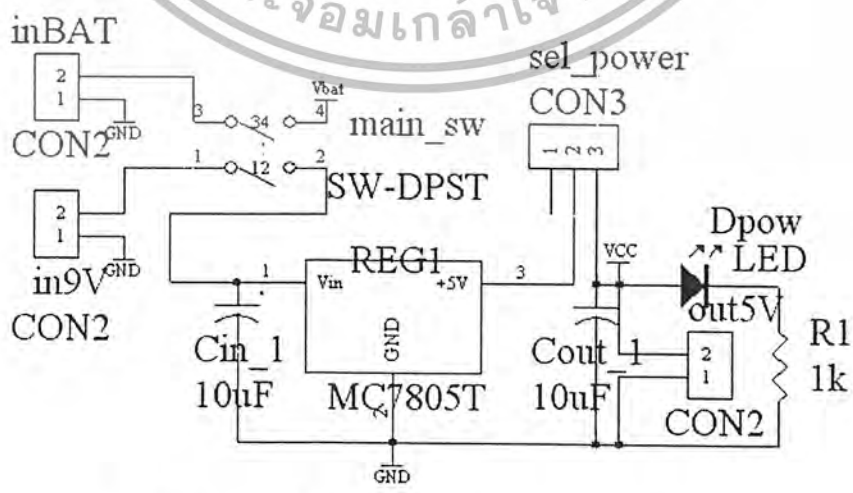
ตารางที่ 3.1 ขาของ cradle connector

ขา	สัญลักษณ์	รายละเอียด
1	V_ADP	AC adapter power in
2	V_ADP	AC adapter power in
3	DTR	RS-232 Data Terminal Ready
4	GND	Power ground

5	CTS	RS-232 Clear To Send
6	RTS	RS-232 Request To Send
7	TXD	RS-232 Transmit Data (from the SA-1110)
8	RXD	RS-232 Receive Data (from the SA-1110)
9	DCD	RS-232 Data Carry Detect
10	GND	Power ground
11	UDC_P	USB positive data signal
12	USB_N	USB negative data signal
13	FG	Frame ground
14	FG	Frame ground
15	FG	Frame ground

โดยการส่งข้อมูลผ่านพอร์ตอนุกรม จะส่งข้อมูลออกมาทางขา 7 (RXD) และ ขา 8 (RXD) ไปยัง MAX232 โดยจะต้องเชื่อมขา3 (DTR) กับขา9 (DCD) และขา5 (CTS) กับ ขา6 (RTS) เข้าด้วยกัน

3.1.3 วงจรแหล่งจ่ายไฟ



รูปที่ 3.5 วงจรแหล่งจ่ายไฟสำหรับหุ่นยนต์เตะฟุตบอล

แหล่งจ่ายไฟในโครงงานนี้มี 2 ส่วนคือ แหล่งจ่ายไฟ 5โวลต์ และแหล่งจ่ายไฟ 24 โวลต์ในส่วนของแหล่งจ่ายไฟ 5โวลต์ ใช้ไอซีเรกกูเลเตอร์เบอร์ 7805 เปลี่ยนแรงดัน 9 โวลต์เป็นแรงดันไฟตรง 5 โวลต์ ซึ่งใช้เป็นไฟเลี้ยงให้กับไอซี MAX232, AT89C51และ AT89C2051

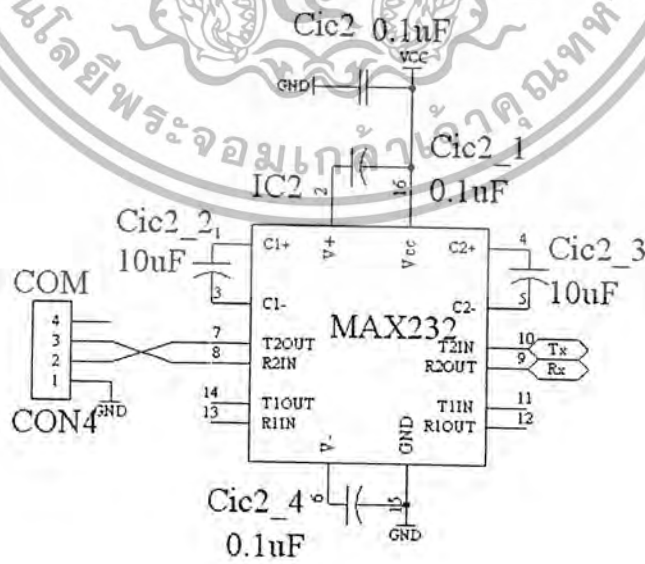
ส่วนแหล่งจ่ายไฟ 24 โวลต์ ใช้ถ่านไฟฉายที่สามารถอัดประจุใหม่ได้ มีแรงดันก้อนละ 1.2 V จำนวน 20 ก้อน เป็นไฟเลี้ยงให้กับมอเตอร์กระแสตรง

การที่ต้องแยกแหล่งจ่ายไฟออกเป็น 2 ส่วน เนื่องจากถ้าใช้แหล่งจ่ายไฟร่วมกันระหว่างมอเตอร์กับไอซีต่างๆในวงจรจะทำให้มีสัญญาณรบกวนที่เกิดจากกระแสกระชากของมอเตอร์ไปรบกวนการทำงานของไอซีต่างๆในวงจรได้

3.1.4 วงจรส่งข้อมูลจาก Pocket PC ไปยังไมโครคอนโทรลเลอร์

การส่งข้อมูลจาก Pocket PC ไปยังไมโครคอนโทรลเลอร์จะส่งข้อมูลผ่านพอร์ตอนุกรม โดยใช้ไอซี MAX232 เป็นตัวรับข้อมูลจาก Pocket PC แล้วส่งข้อมูลไปยังไมโครคอนโทรลเลอร์ AT89C51 ทางขา R2OUT และ T2IN

ไอซี MAX232 เป็นไอซีที่ใช้ในการรับส่งข้อมูล โดยมีความเร็วในการส่งข้อมูล 120 Kbps มีค่าสลูว์เรท (slew rate) ไม่ต่ำกว่า 3V/us



รูปที่ 3.6 วงจรส่งข้อมูลจาก Pocket PC ไปยังไมโครคอนโทรลเลอร์

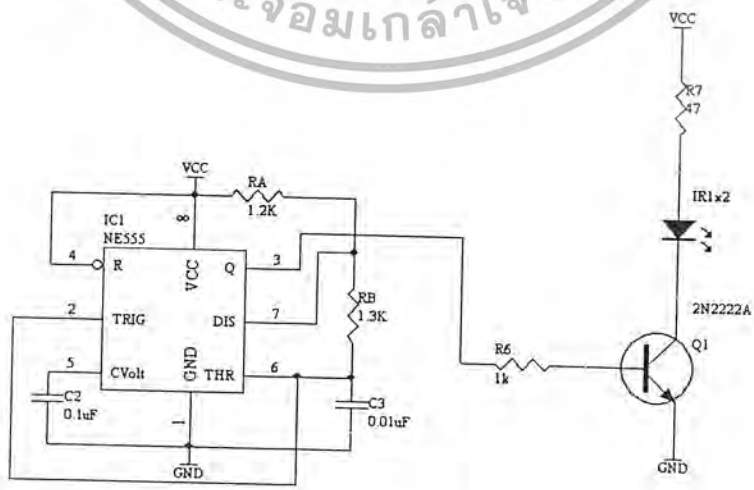
จากรูปที่ 3.6 แสดงวงจรการส่งข้อมูลผ่านพอร์ตอนุกรมโดยใช้ไอซี MAX232 ซึ่งเป็นไอซีที่มีขา ต่างๆเป็นดังตารางที่ 3.2

ตารางที่ 3.2 ขาและหน้าที่การทำงานของไอซี MAX232

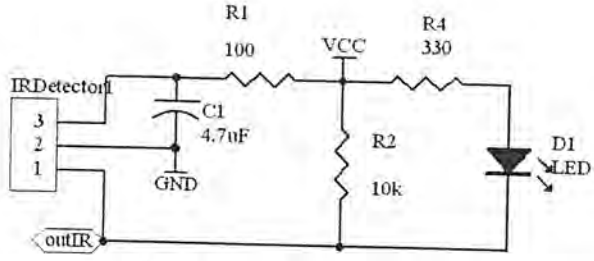
ขา	ชื่อ	รายละเอียด
1,3	C1+,C1-	Terminals for positive charge-pump capacitor
2	V+	+2VCC voltage generated by the charge pump
4,5	C2+,C2-	Terminals for negative charge-pump capacitor
6	V-	-2VCC voltage generated by the charge pump
7,14	T1OUT,T2OUT	RS-232 Driver Outputs
8,13	R1IN,R2IN	RS-232 Receiver Inputs
9,12	R1OUT,R2OUT	RS-232 Receiver Outputs
10,11	T1IN,T2IN	RS-232 Driver Inputs
15	GND	Ground
16	VCC	+4.5V to +5.5V Supply-Voltage Input

3.1.5 วงจรอินฟราเรด

เนื่องจากถ้าลูกบอลอยู่ใกล้หุ่นยนต์เกินไปจะทำให้กล้องดิจิตอลไม่สามารถจับภาพลูกบอลได้ จึงต้องมีวงจรเซ็นเซอร์สำหรับตรวจจับลูกบอลอยู่บริเวณด้านหน้าซึ่งกล้องดิจิตอลจับภาพลูกบอลไม่ได้ โดยเซ็นเซอร์ที่ใช้จะเป็นอินฟราเรดและมีวงจรดังรูปที่ 3.7 และ 3.8



รูปที่ 3.7 วงจรอินฟราเรด (ภาคส่ง)



รูปที่ 3.8 วงจรอินฟราเรด (ภาครับ)

จากรูปที่ 3.7 ซึ่งเป็นวงจรอินฟราเรดภาคส่ง จะเห็นว่ามีวงจร Astable Multivibrator โดยใช้ IC NE555 การทำงานของวงจร คือ เมื่อตัวเก็บประจุ C_3 ได้รับประจุที่มาจากกระแสที่ไหลผ่าน R_A และ R_B จนแรงดันที่ตกคร่อมตัวมันมีค่า 2*VCC/3 วงจรภายในจะทำให้ตัวเก็บประจุคายประจุจนกระทั่งแรงดันตกคร่อม C_3 เหลือ VCC/3 โดยการคายประจุผ่าน R_B เข้าไปยังวงจรภายในแล้ว C_3 จะเริ่มประจุใหม่ แรงดันที่ตกคร่อม C_3 จึงแกว่งอยู่ระหว่าง VCC/3 กับ 2* VCC/3 เป็นผลทำให้แรงดันเอาท์พุทเกิดการสวิตซ์อยู่ระหว่าง 2 ระดับ คือ VCC กับ กราวนด์ โดยความถี่ที่ได้จากวงจรนี้หาได้จากสมการ 3.1 ส่วนค่าดิวตี้ไซเคิล (D) หาได้จากสมการ 3.2

$$f = \frac{1.44}{(R_A + 2R_B)C_3} \qquad \text{(สมการ 3.1)}$$

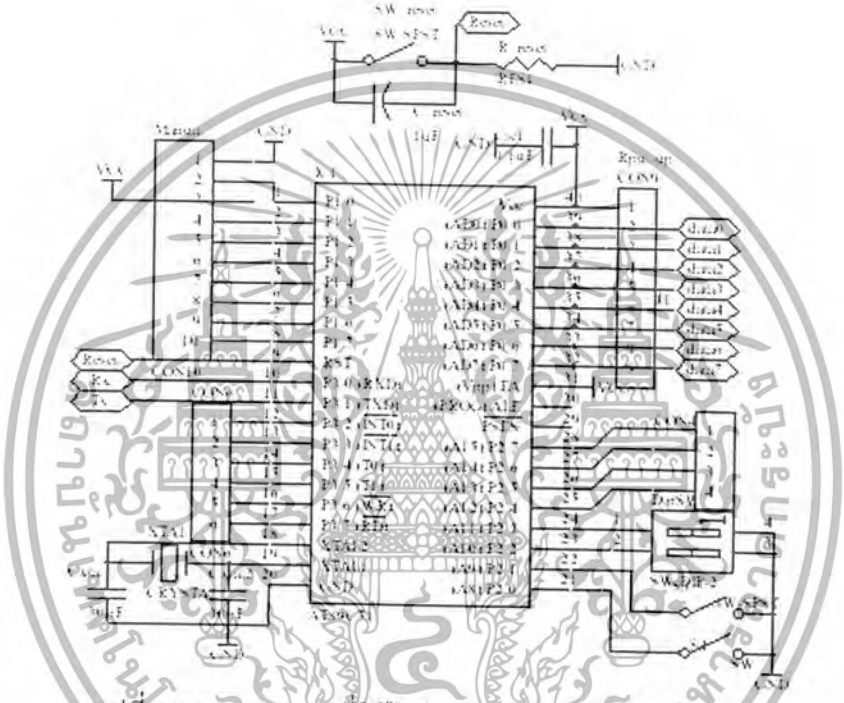
$$D = \frac{R_B}{R_A + 2R_B} \qquad \text{(สมการ 3.2)}$$

เนื่องจากอินฟราเรดตัวรับซึ่งใช้เบอร์ TSOP4838 สามารถรับสัญญาณอินฟราเรดที่ความถี่ 38kHz ดังนั้นภาคส่งจึงต้องส่งสัญญาณที่ความถี่เดียวกันนี้ โดยค่าความต้านทานที่ใช้ได้แก่ R_A = 1.2k, R_B = 1.3k และ C_3= 0.01uF จะสามารถผลิตความถี่ประมาณ 38kHz ค่าดิวตี้ไซเคิล 34.21 % จากนั้นเอาท์พุทจาก NE555 นำไปขับกระแสให้กับ IR LED โดยใช้ทรานซิสเตอร์ชนิด NPN เบอร์ 2N2222

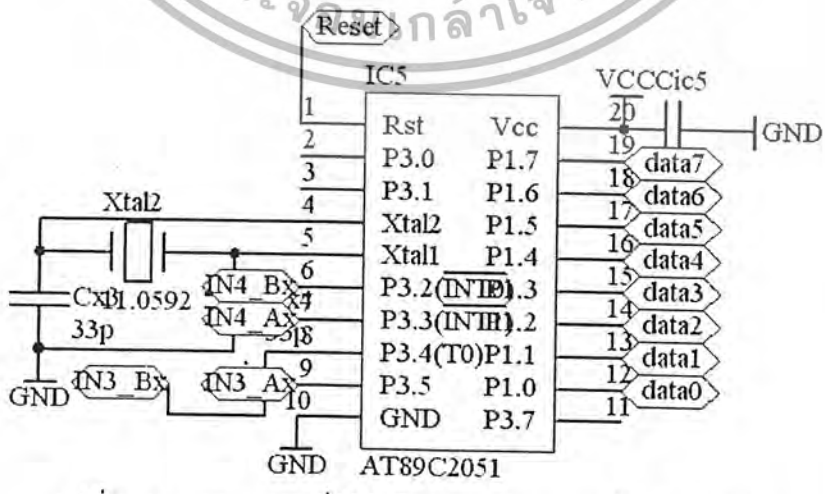
การที่ต้องใช้ตัวรับที่รับสัญญาณที่ความถี่เฉพาะ เพื่อไม่ให้ตัวรับได้รับผลจากคลื่นอินฟราเรดที่มีอยู่ในแสงแดด ทำให้สามารถตรวจสอบลูกบอลด้านหน้าได้อย่างมีประสิทธิภาพ

3.1.6 วงจรควบคุม

วงจรควบคุมของหุ่นยนต์เตะฟุตบอลจะใช้ไมโครคอนโทรลเลอร์ AT89C51 ควบคุมมอเตอร์ที่ใช้ในการเคลื่อนที่ และใช้ไมโครคอนโทรลเลอร์ AT89C2051 ควบคุมมอเตอร์ที่ใช้ในการเลี้ยงและยิงลูกบอล วงจรสำหรับไมโครคอนโทรลเลอร์ AT89C51และไมโครคอนโทรลเลอร์ AT89C2051 เป็นดังรูปที่ 3.9 และ 3.10 ตามลำดับ



รูปที่ 3.9 วงจรควบคุมที่ใช้ไมโครคอนโทรลเลอร์ AT89C51



รูปที่ 3.10 วงจรควบคุมที่ใช้ไมโครคอนโทรลเลอร์ AT89C2051

3.1.7 วงจรขับมอเตอร์

วงจรขับมอเตอร์ ในโครงงานนี้ได้ใช้ IC L293D 2 ตัว ในการขับมอเตอร์กระแสตรง 4 ตัว โดยไอซี L293D 1 ตัวใช้ขับมอเตอร์กระแสตรง 2 ตัว

L293D เป็นไอซีสำหรับขับกระแสที่สามารถจ่ายกระแสได้สูง โดยมี 4 Channel สำหรับการขับแบบทิศทางเดียว หรือใช้เป็น 2 Channel สำหรับการขับแบบ H-Bridge

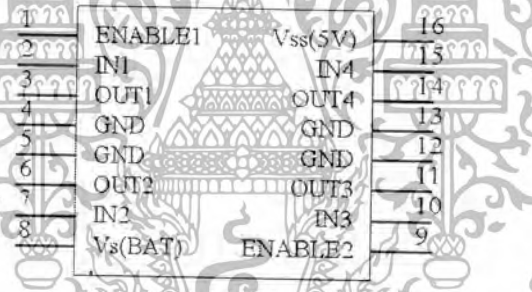
คุณสมบัติของ L293D

1.จ่ายกระแสได้พร้อมกัน 600mA ต่อ 1 CHANNEL

2.กระแสสูงสุดที่แต่ละ CHANNEL สามารถจ่ายได้ คือ 1.2A

3.สามารถป้องกันสัญญาณรบกวนได้ดี

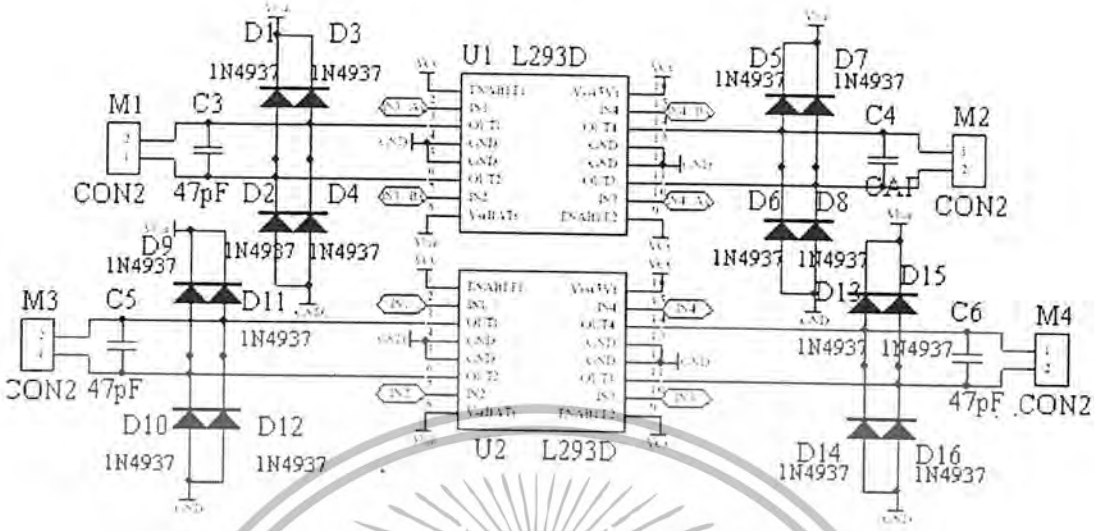
4.มี clamping diode ภายใน



รูปที่ 3.11 ขาของไอซี L293D

จากรูปที่ 3.11 แสดงขาของไอซี L293D

- ขา 2,7,10 และ 15 เป็นอินพุท1, อินพุท2, อินพุท3, อินพุท4 ตามลำดับ
- ขา 3, 6,11และ 14 เป็นเอาท์พุท1, เอาท์พุท2, เอาท์พุท3, เอาท์พุท4 ตามลำดับ
- ขา 1 เป็นขา enable ควบคุมเอาท์พุท1 และเอาท์พุท 2 ซึ่งจะactive high
- ขา 9 เป็นขา enable ควบคุมเอาท์พุท3 และเอาท์พุท 4 ซึ่งจะactive high

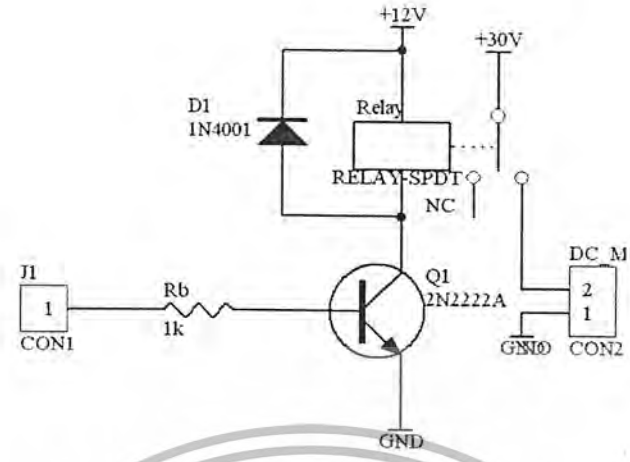


รูปที่ 3.12 วงจรควบคุมมอเตอร์สำหรับหุ่นเตะฟุตบอล

จากรูปที่ 3.12 แสดงวงจรขับมอเตอร์ โดยอินพุทของวงจรมี 4 อินพุทซึ่งมาจากไมโครคอนโทรลเลอร์ เอาท์พุทคือ มอเตอร์ M1 - M4 โดยจะมีตัวเก็บประจุแบบเซรามิก ต่อคร่อมมอเตอร์แต่ละตัวไว้ เพื่อช่วยลดผล transient response ของมอเตอร์ ซึ่งจะทำให้มอเตอร์เปลี่ยนจากหมุนด้านหนึ่งไปเป็นหมุนอีกด้านหนึ่งได้ดีขึ้น ส่วนไดโอดเบอร์ 1N4937 ซึ่งเป็นแบบ Fast Recovery Rectifier ที่ต่ออยู่ระหว่างมอเตอร์กับไฟเลี้ยงและมอเตอร์กับกราวด์ ทำหน้าที่ป้องกันกระแสไหลย้อนกลับ

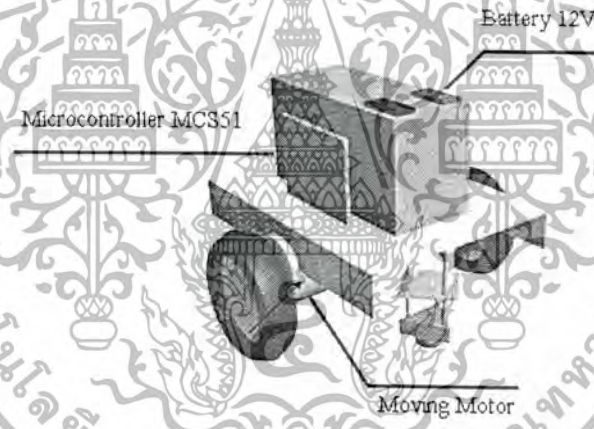
3.1.8 วงจรรีเลย์

วงจรรีเลย์ใช้เป็นวงจรสำหรับการขับมอเตอร์กระแสตรงเพื่อให้ได้กระแสจากแหล่งจ่ายเต็มที่ จากรูปที่ 3.13 แสดงวงจรรีเลย์ที่ใช้ วงจรรีเลย์ใช้ไฟเลี้ยง 12 โวลต์ซึ่งได้จากไอซีเรกกูเลเตอร์เบอร์ LM7812T แปลงแรงดันไฟตรง 24 โวลต์จากแบตเตอรี่ เป็นแรงดันไฟตรง 12 โวลต์ การทำงานของวงจรคือ เมื่อทรานซิสเตอร์ Q1 ไม่นำกระแสจะทำให้หน้าสัมผัส ของรีเลย์อยู่ที่ตำแหน่ง NC(Normal Close) แต่เมื่อทรานซิสเตอร์ Q1 นำกระแสจะทำให้หน้าสัมผัส ของรีเลย์เลื่อนไปอยู่ที่ตำแหน่ง NO(Normal Open) ซึ่งจะเป็นการขับมอเตอร์กระแสตรง สำหรับไดโอด 1N4001 ที่ต่อคร่อมรีเลย์ใช้เพื่อป้องกันกระแสไหลย้อนกลับ



รูปที่ 3.13 วงจรรีเลย์

3.2 หุ่นยนต์ผู้รักษาประตู



รูปที่ 3.14 โครงสร้างของหุ่นยนต์ผู้รักษาประตู

โครงสร้างของหุ่นยนต์ผู้รักษาประตูแสดงดังรูปที่ 3.14 โดยมีหลักการทำงานคือ นำภาพที่ได้จากกล้องWebCam มาทำการประมวลผลภาพดิจิตอลด้วยคอมพิวเตอร์ เพื่อค้นหาตำแหน่งลูกบอล จากนั้นใช้ระบบปัญญาประดิษฐ์ในการหาตำแหน่งที่หุ่นยนต์สามารถเคลื่อนที่ไปป้องกันประตูได้ แล้วส่งคำสั่งควบคุมไปยังไมโครคอนโทรลเลอร์บนหุ่นยนต์ทางพอร์ตอนุกรม ซึ่งจะควบคุมมอเตอร์ที่ใช้ในการเคลื่อนที่ของหุ่นยนต์ ฮาร์ดแวร์ของหุ่นยนต์ผู้รักษาประตูประกอบด้วยส่วนต่างๆดังนี้

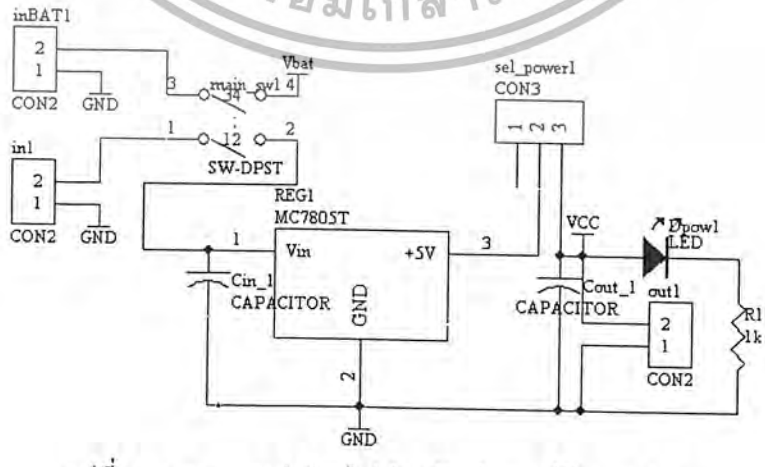
3.2.1 กล้องWebCam

กล้องWebCam ซึ่งติดตั้งไว้ที่เหนือพื้นสนาม เป็นกล้องที่ใช้จับภาพลูกบอล นำภาพที่ได้มาทำการประมวลผลภาพดิจิตอลเพื่อหาตำแหน่งของลูกบอล หุ่นยนต์ผู้รักษาประตูจะรู้ตำแหน่งของลูกบอลทำให้สามารถป้องกันประตูได้

รูปที่ 3.15 กล้อง WebCam รุ่น AlphaCam pro

กล้องWebCam ที่ใช้เป็นกล้องรุ่น AlphaCam pro ดังรูป 3.15 ซึ่งมีคุณสมบัติคือ ความละเอียดสูงสุด 640x480 พิกเซล สามารถถ่ายภาพนิ่ง, ภาพเคลื่อนไหวและถ่ายวีดีโอได้ โดยใช้การเชื่อมต่อทางพอร์ต USB

3.2.2 วงจรแหล่งจ่ายไฟ



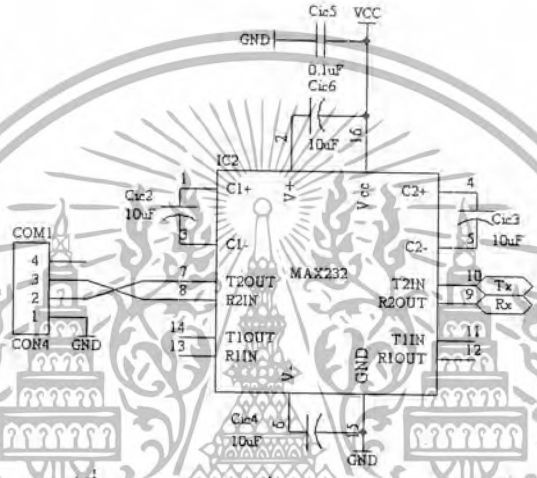
รูปที่ 3.16 วงจรแหล่งจ่ายไฟสำหรับหุ่นยนต์ผู้รักษาประตู

วงจรแหล่งจ่ายไฟดังรูปที่ 3.14 แบ่งออกเป็น 2 ส่วนคือ

1) แหล่งจ่ายไฟตรง 12 โวลต์ ใช้แบตเตอรี่ 12 โวลต์ เพื่อเป็นไฟเลี้ยงให้มอเตอร์

2) แหล่งจ่ายไฟตรง 5 โวลต์ ใช้ไอซีเรกกูเลเตอร์ 7805 แปลงแรงดันไฟตรง 12 โวลต์ เป็นแรงดันไฟตรง 5 โวลต์

3.2.3 วงจรรับข้อมูลจากคอมพิวเตอร์

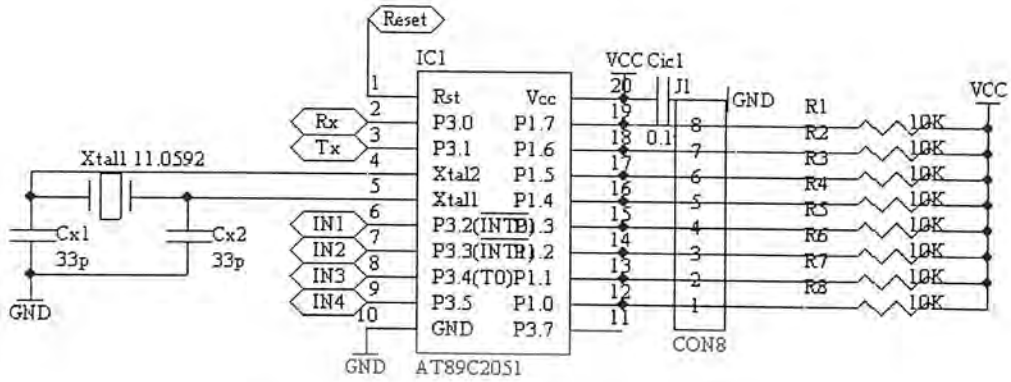


รูปที่ 3.17 วงจรรับข้อมูลจากคอมพิวเตอร์

วงจรรับข้อมูลจากคอมพิวเตอร์ดังรูป 3.17 จะใช้ไอซี MAX 232 ในการรับข้อมูลซึ่งส่งมาทางพอร์ตอนุกรม โดยรับข้อมูลเข้าทางขา T2OUT และ R2IN และส่งข้อมูลที่ได้ไปยังวงจรควบคุมทางขา Rx และ Tx

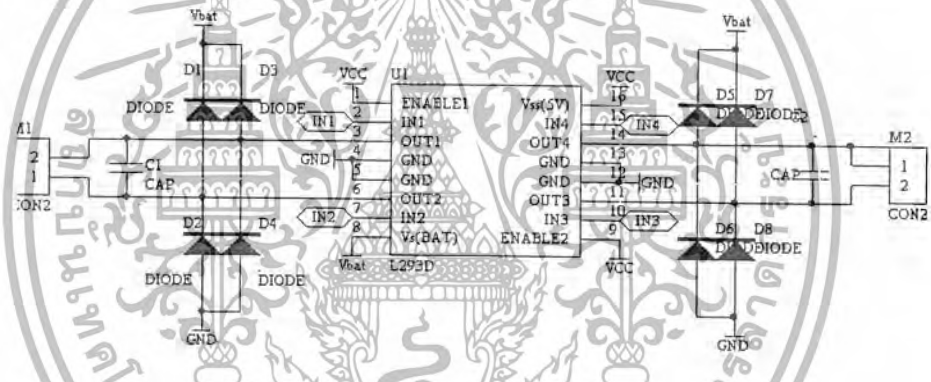
3.2.4 วงจรควบคุม

วงจรควบคุมดังรูป 3.18 ใช้ไมโครคอนโทรลเลอร์เบอร์ AT89C2051 ในการประมวลผลสัญญาณที่ส่งมาจากไอซี MAX232 เพื่อนำไปควบคุมมอเตอร์ที่ใช้ในการเคลื่อนที่



รูปที่ 3.18 วงจรควบคุมสำหรับหุ่นยนต์ผู้รักษาประตู

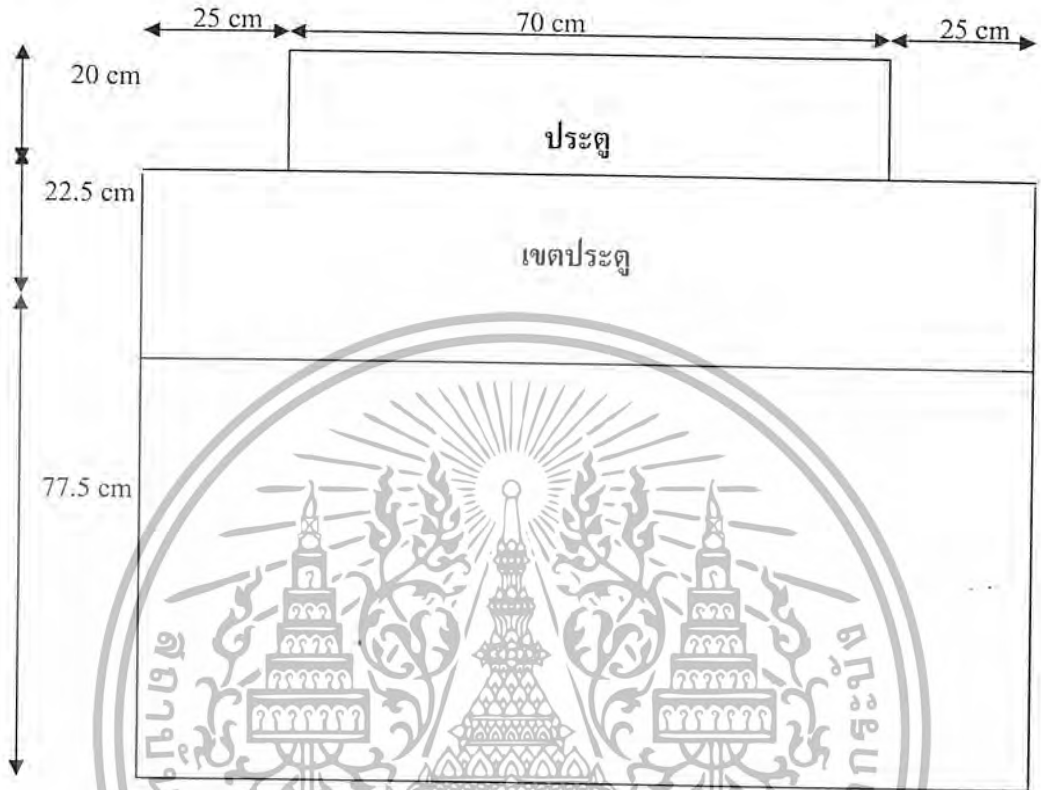
3.2.5 วงจรขับมอเตอร์



รูปที่ 3.19 วงจรขับมอเตอร์สำหรับหุ่นยนต์ผู้รักษาประตู

วงจรขับมอเตอร์ดังรูป 3.19 ใช้ไอซี L293D ซึ่งจ่ายกระแสได้สูงขับมอเตอร์กระแสตรง 2 ตัวที่ใช้ในการเคลื่อนที่

3.3 รูปแบบสนาม



รูปที่ 3.20 สนามที่ใช้ในโครงงาน

สนามที่ใช้ในโครงงานนี้มีรูปแบบดังรูปที่ 3.20 โดยมีรายละเอียดต่างๆดังนี้

ขนาด

- สนามเป็นรูปสี่เหลี่ยมผืนผ้า ขนาด กว้าง 100 ซม. ยาว 120 ซม.

ผิวสนาม

- ผิวสนามเป็นพื้นแข็ง เรียบ และ มีสีเขียว

กำแพง เส้น และ จุดต่างๆบนสนาม

- กำแพงสนาม เป็นสีขาว สูง 10 ซม.

- กำแพงสนามมีสี่ด้าน ด้านที่ยาวกว่าสองด้านเรียกว่า กำแพงสนามด้านข้าง ส่วนอีกสองด้านที่เหลือ เรียกว่า กำแพงสนามด้านหลัง

- เส้นทุกเส้นบนสนามเป็น เส้นสีขาว กว้าง 1 ซม.

เขตผู้รักษาประตู

- กำหนดโดยเส้นด้านหน้าของประตู เส้นซึ่งขนานกับปากประตูอยู่ห่างจากปากประตู 22.5 ซม.

ประตู

- เป็นสี่เหลี่ยมผืนผ้าโดยห่างจากกำแพงสนามด้านข้าง 25 ซม. โดยประตูมีความกว้าง 70 ซม. สูง 10 ซม. และลึกเข้าไป 20 ซม.

- ประตูมีสีเหลือง

- มีคานซึ่งอยู่สูงจากพื้นสนาม 10 ซม. (วัดจากระดับคานบน) โดยมีความสูง 1 นิ้ว และมีสีขาว

บทที่ 4
การออกแบบโปรแกรม

ในโครงงานนี้ใช้ภาษา C# ในการเขียนโปรแกรมเนื่องจากเป็นภาษาที่เขียนง่าย และสามารถพัฒนาบน Pocket PC ได้ การออกแบบโปรแกรมแบ่งออกเป็นส่วนต่างๆดังนี้

4.1 การออกแบบโปรแกรมสำหรับหุ่นยนต์เตะฟุตบอล

4.1.1 การจับภาพจากกล้อง

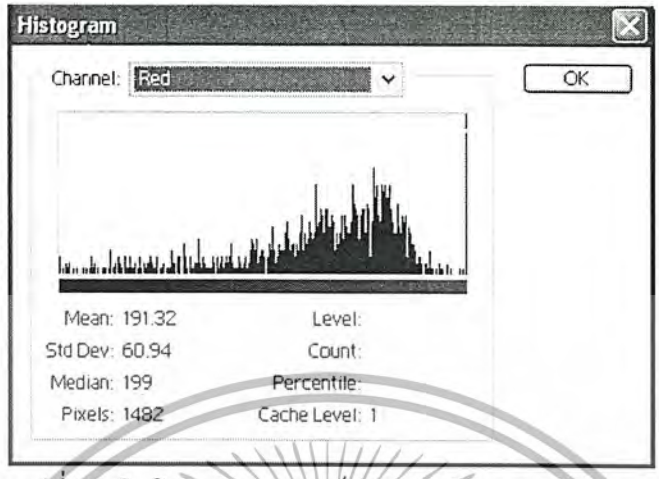
จากคุณสมบัติของกล้อง สามารถจับภาพด้วยความละเอียด 3 ระดับ คือ 640x480, 320x240 และ 160x120 แต่เนื่องจากข้อจำกัดทางด้านความเร็วในการประมวลผลของ Pocket PC จึงเลือกใช้ที่ความละเอียดต่ำสุด คือ 160x120 เนื่องจากเป็นการนำภาพมาหาตำแหน่งของวัตถุ ความละเอียดของภาพจึงไม่มีความจำเป็นมากนัก

ในการนำภาพมาประมวลผลนั้น จะใช้ฟังก์ชัน Preview ของกล้องในการจับภาพอย่างต่อเนื่อง โดยกล้องจะส่งข้อความ (window message) ที่มีชื่อว่า WM_PREV_END มายังเครื่อง Pocket PC เมื่อจับภาพได้แต่ละเฟรม ดังนั้นจึงใช้การตรวจสอบข้อความดังกล่าว ในการนำภาพมาประมวลผล

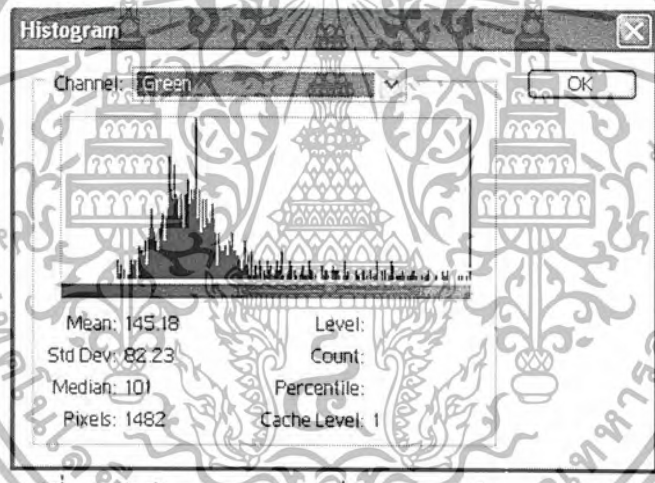
4.1.2 การประมวลผลภาพ

ในโครงงานนี้ ได้ทำการทดลองหาช่วงสี เพื่อใช้ในการแยกส่วนภาพ โดยช่วงสีที่ใช้ในการแยกลูกบอลออกจากสนาม ได้มาจากการนำภาพตัวอย่างมาทำฮิสโทแกรม (Histogram) ขององค์ประกอบสีทั้งสาม คือ R, G และ B ดังนี้

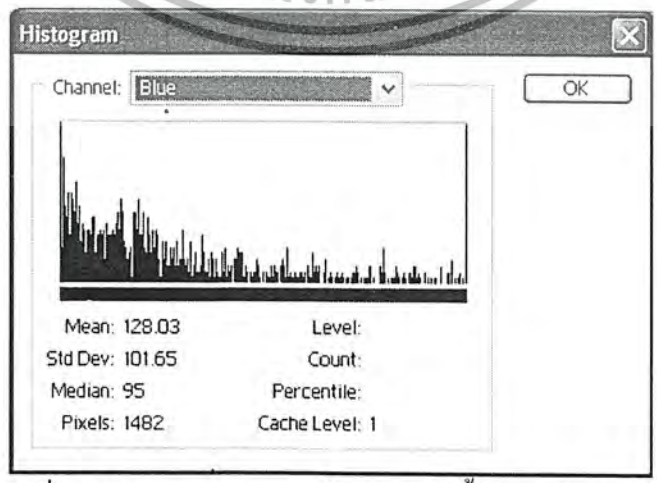
รูปที่ 4.1 ภาพตัวอย่างจากกล้อง



รูปที่ 4.2 ฮิสโทแกรมขององค์ประกอบสีแดงของลูกบอล



รูปที่ 4.3 ฮิสโทแกรมขององค์ประกอบสีเขียวของลูกบอล



รูปที่ 4.4 ฮิสโทแกรมขององค์ประกอบสีน้ำเงินของลูกบอล

จากรูปที่ 4.1 เป็นภาพที่ได้จากกล้อง นำมาตัดเอาเฉพาะบริเวณที่เป็นลูกบอล แล้วนำมาทำฮิสโทแกรมขององค์ประกอบสีทั้งสามด้วยโปรแกรม Photoshop ได้ดังรูปที่ 4.2 – รูปที่ 4.4

จากฮิสโทแกรมที่ได้ สามารถนำมากำหนดช่วงสีคงที่ สำหรับใช้ในการแยกลูกบอล ได้แต่เนื่องจากความเข้มของแสงขึ้นกับระยะห่างของวัตถุจากกล้อง ดังนั้นการแยกช่วงสีด้วยช่วงคงที่อย่างเดียวไม่สามารถแยกวัตถุได้ทุกระยะ จากการทดลองนำภาพที่ถ่ายได้จากกล้องที่ระยะต่างๆกันจากวัตถุพบว่าสามารถกำหนดช่วงในการแยกสีคือ สีแดงต้องมากกว่าสีเขียว และสีเขียวต้องมากกว่าสีน้ำเงิน หรือ

$$R > G > B$$

หลังจากนั้นเมื่อแยกช่วงสีแล้ว จึงนำจุดที่แยกได้ มาหาค่าเฉลี่ยของตำแหน่ง ตามสมการที่ 2.9 แล้วนำตำแหน่งดังกล่าว ไปประมวลผลในส่วนปัญญาประดิษฐ์ต่อไป

ในส่วนของการแยกช่วงสีเพื่อค้นหาประตู ก็ใช้หลักการเดียวกันโดยประตูซึ่งมีสีเหลืองสามารถแยกได้ด้วยช่วงสีดังนี้

$$G - R > 10$$

$$G - B > 10$$

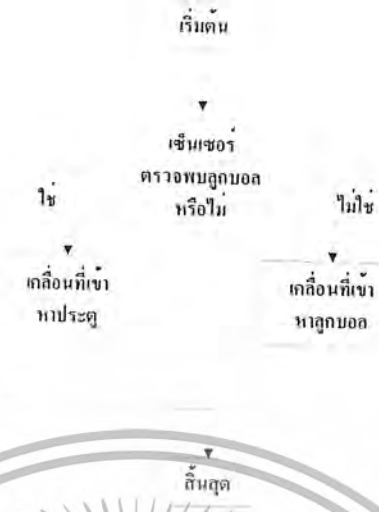
$$|R - B| < 20$$

$$G > 60$$

การหาตำแหน่งของประตู ใช้การนำพิกัดในแนวแกน X ซึ่งอยู่ซ้ายสุดและขวาสุดมาหาค่าเฉลี่ยกัน ส่วนค่าพิกัดในแนวแกน Y ใช้พิกัดซึ่งอยู่ต่ำสุด ทั้งนี้เนื่องจากขอบเขตประตูมีขนาดใหญ่ การนำทุกพิกัดที่แยกได้มาหาค่าเฉลี่ยทำให้มีการคำนวณมาก อีกทั้งไม่ต้องการความแม่นยำมากนัก

4.1.3 ปัญญาประดิษฐ์

การทำงานของหุ่นยนต์เริ่มจากตรวจสอบว่าเซ็นเซอร์อินฟราเรดพบลูกบอลหรือไม่ ถ้าตรวจเจอแสดงว่าหุ่นยนต์ครองบอลอยู่ ให้เคลื่อนที่เข้าหาประตูเพื่อยิงประตู ในกรณีที่ตรวจหาลูกบอลบนภาพไม่พบ ได้ออกแบบให้หุ่นยนต์ค้นหาลูกบอล ดังโฟลว์ชาร์ตรูปที่ 4.5

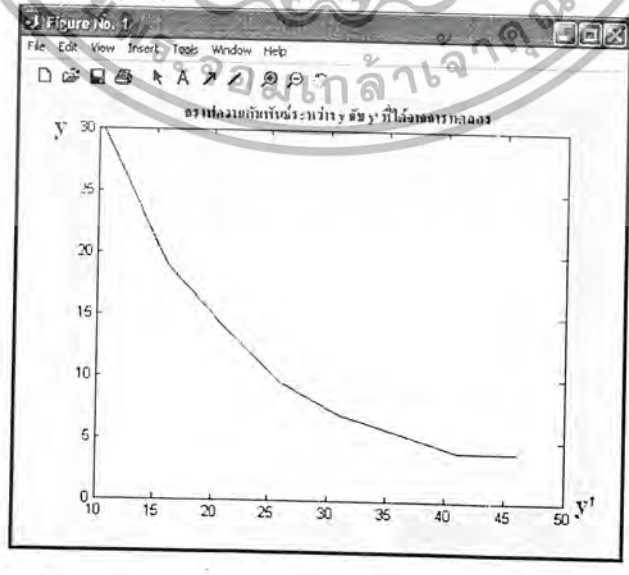


รูปที่ 4.5 โฟลว์ชาร์ตแสดงการประมวลผลของส่วนปัญญาประดิษฐ์

4.1.3.1 การเคลื่อนที่เข้าหาลูกบอล

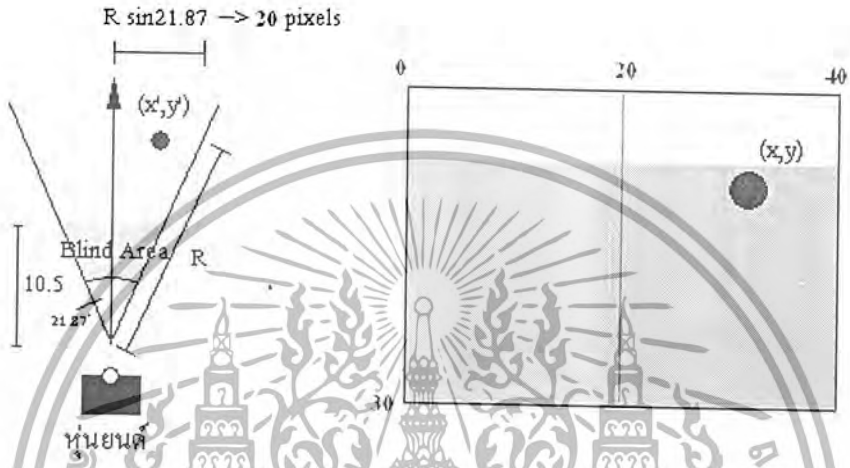
เป็นส่วนที่ทำหน้าที่ตัดสินใจว่าจะให้หุ่นเคลื่อนที่ไปในทิศทางใด ด้วยความเร็วเท่าไร โดยความเร็วของล้อแต่ละข้าง มี 8 ระดับ ทิศทางและความเร็วที่จะทำการเคลื่อนที่นั้น ขึ้นอยู่กับตำแหน่งของลูกบอล

จากตำแหน่งบนภาพ (x,y) นำมาคำนวณหาตำแหน่งของวัตถุจริง (x',y') โดยใช้การทดลองวางลูกบอลที่ตำแหน่งต่างๆ แล้วจับภาพหาตำแหน่งบนภาพ ได้กราฟดังรูปที่ 4.6 โดยมีแกนนอนเป็นระยะห่างของวัตถุจากกล้อง(y') แกนตั้งเป็นตำแหน่งบนภาพ(y)



รูปที่ 4.6 ผลการทดลองหาความสัมพันธ์ระหว่าง y กับ y'

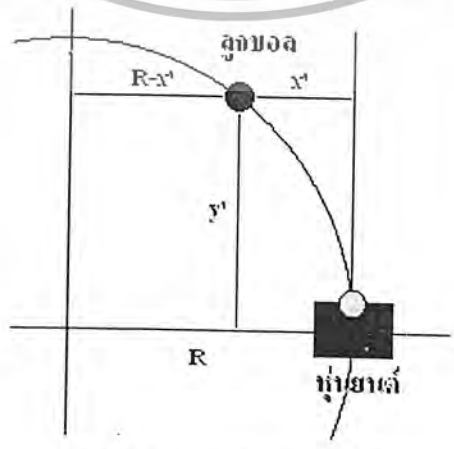
จากนั้นเมื่อทำการทดลองวัดมุมที่กล้องสามารถมองเห็นได้ โดยการวางลูกบอลให้ปรากฏอยู่ที่ขอบของภาพ พบว่ามุมการมองเห็นของกล้องมีค่าประมาณ 21.87 องศา จากแนวตั้งฉาก นำมาคำนวณหาความสัมพันธ์ของ x กับ x' โดยการเทียบบัญญัติไตรยางศ์ตามรูปที่ 4.7 (ถือว่ามุมมีค่าน้อย จึงมีความสัมพันธ์เป็นเชิงเส้น) จะได้ดังสมการที่ 4.1



รูปที่ 4.7 ความสัมพันธ์ของตำแหน่งจริงกับตำแหน่งบนภาพ

$$x' = \frac{y'(x-20)\times \tan 21.87^{\circ}}{20} = \frac{y'(x-20)}{50} \qquad \text{(สมการที่ 4.1)}$$

เมื่อได้ตำแหน่งจริงมาแล้ว นำมาคำนวณหาระดับความเร็วที่เหมาะสมในการเคลื่อนที่ โดยกำหนดให้หุ่นยนต์เคลื่อนที่ในแนววงกลม ทำการหารัศมีความโค้งของการเคลื่อนที่จากตำแหน่งของหุ่นยนต์ ไปยังตำแหน่งของลูกบอล ดังแสดงในรูปที่ 4.8



รูปที่ 4.8 การหารัศมีความโค้ง

จากรูปที่ 4.8 จะได้ว่า

$$R^2 = (R - x')^2 + y'^2$$

ดังนั้นจะได้สมการที่ 4.2 คือ

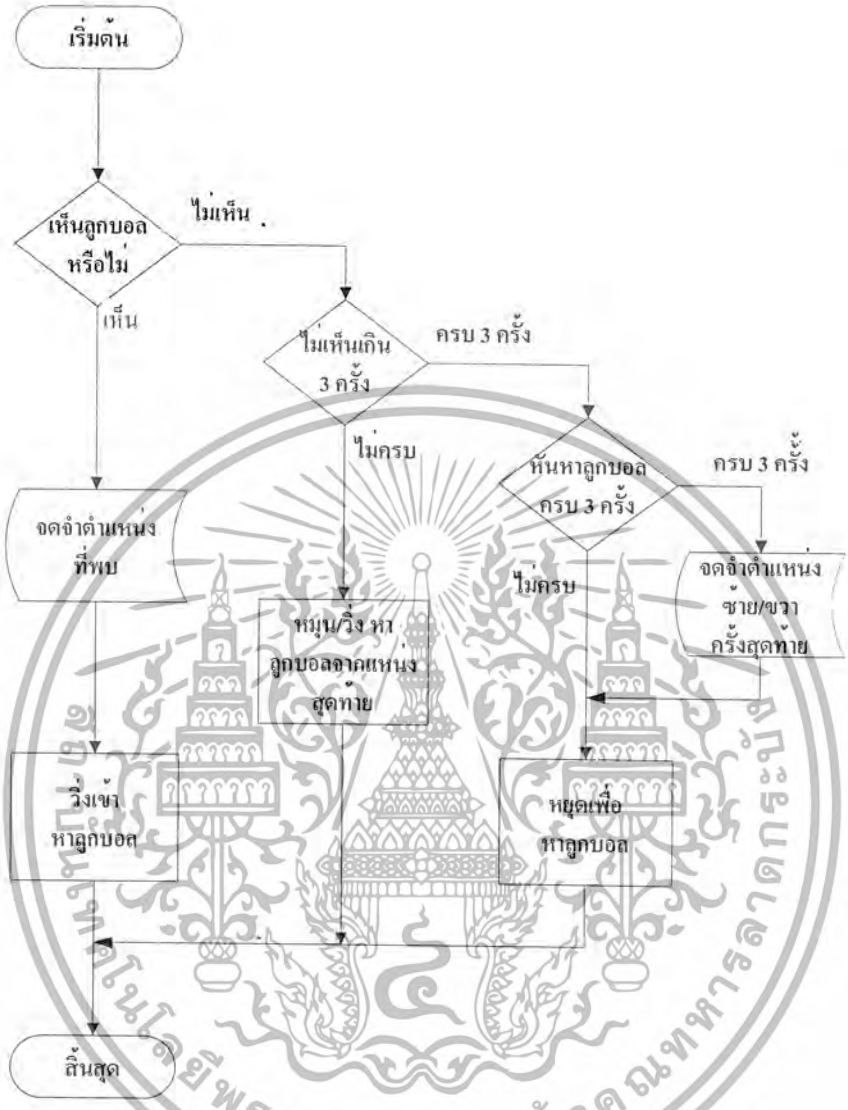
$$R = \frac{x'^2 + y'^2}{2x'} \qquad \text{(สมการที่ 4.2)}$$

จากรัศมีความโค้งที่คำนวณได้ นำไปเทียบกับตารางที่ 4.1 ซึ่งแสดงรัศมีความโค้งของการเคลื่อนที่ที่รูปแบบการเคลื่อนที่ต่างๆซึ่งได้จากการทดลอง โดยการเลือกใช้รูปแบบการเคลื่อนที่ จะต้องเลือกรูปแบบที่มีค่ามากกว่ารัศมีความโค้งที่ต้องการ

ตารางที่ 4.1 รัศมีความโค้งการเคลื่อนที่ของรูปแบบการเคลื่อนที่ต่างๆ

รูปแบบการเคลื่อนที่	รัศมีความโค้ง (cm)
0xF8	12
0xF9	18
0xFA	32
0xFB	45
0xFC	70
0xFD	104
0xFE	324
0xFF	อนันต์

ในกรณีที่ตรวจหาลูกบอลบนภาพไม่พบ ได้ออกแบบให้หุ่นยนต์ค้นหาลูกบอล โดยจะให้หุ่นยนต์จดจำตำแหน่งของลูกบอลกรณีที่พบลูกบอลไว้ ถ้าไม่พบลูกบอลก็จะนำตำแหน่งที่จดจำนี้มาพิจารณาว่าลูกบอลเคลื่อนที่ไปทางใด ซึ่งตำแหน่งได้กำหนดให้มี 4 แบบ คือ หน้าใกล้หน้าไกล ซ้าย และขวา เช่นถ้าตำแหน่งสุดท้ายที่พบลูกบอลเป็นตำแหน่งซ้าย ก็จะให้หุ่นยนต์หมุนไปทางซ้าย และเมื่อหมุนไปแล้วจะให้หยุดเพื่อตรวจหาลูกบอลอีกครั้ง โฟลว์ชาร์ตแสดงการค้นหาลูกบอลแสดงในรูปที่ 4.9



รูปที่ 4.9 โฟลว์ชาร์ตแสดงการค้นหาลูกบอล

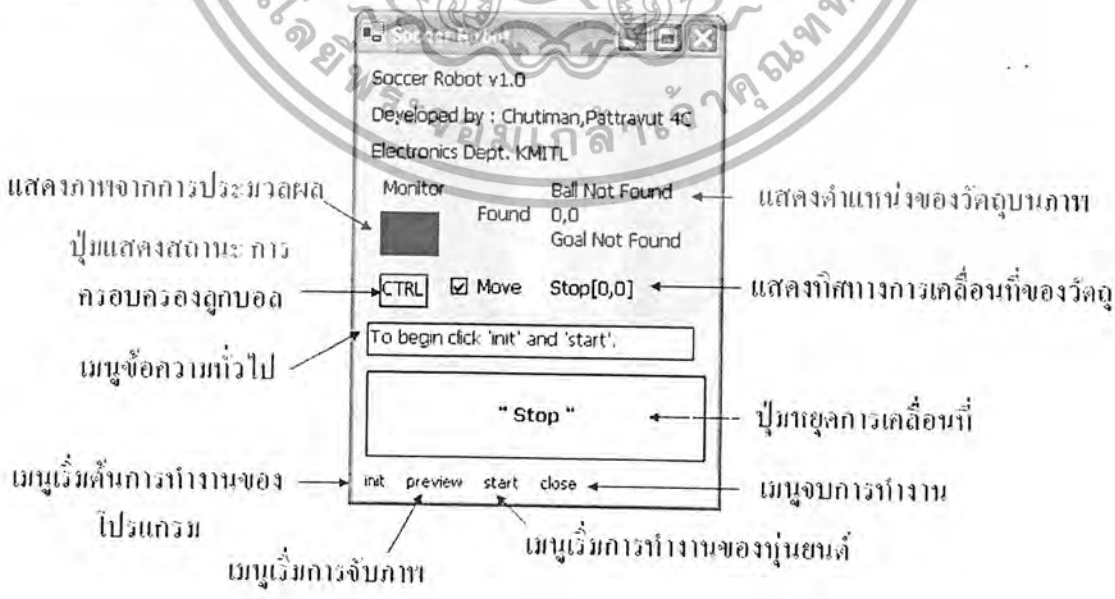
4.1.3.2 การค้นหาประตูและการยิงประตู

เมื่อหุ่นยนต์เข้าหาลูกบอลได้แล้วหุ่นยนต์จะทำการค้นหาประตู โดยการตรวจสอบสีเหลืองซึ่งเป็นสีของประตู เมื่อพบแล้วก็วิ่งเข้าหาประตูและยิงลูกบอลเข้าประตูต่อไป ดังแสดงในโฟลว์ชาร์ตรูปที่ 4.10

รูปที่ 4.10 โฟลว์ชาร์ตแสดงการค้นหาประตูและยิงประตู

4.1.4 การออกแบบหน้าจอของโปรแกรม

หน้าจอของโปรแกรม แสดงดังรูปที่ 4.11 ประกอบด้วยกล่องรูปภาพ แสดงการประมวลผลภาพจากกล้อง กล่องข้อความแสดงตำแหน่งของลูกบอล ระยะห่างและการเคลื่อนที่ และปุ่มควบคุมการทำงานต่างๆ



รูปที่ 4.11 หน้าจอของโปรแกรมบน Pocket PC

4.1.4 การควบคุมหุ่นยนต์

4.1.4.1 การส่งคำสั่งควบคุมจาก Pocket PC

แบ่งออกเป็นสองส่วน คือ การส่งคำสั่งควบคุมเคลื่อนที่ และคำสั่งอื่นๆ ได้แก่ คำสั่งให้ยิงลูกบอล และคำสั่งให้เลี้ยงลูกบอล สำหรับรูปแบบการเคลื่อนที่ จะกำหนดทิศทาง และระดับความเร็วของมอเตอร์แต่ละข้าง มีลักษณะดังนี้

มอเตอร์ด้านซ้าย				มอเตอร์ด้านขวา			
บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
ทิศทาง	ระดับความเร็ว			ทิศทาง	ระดับความเร็ว		

โดย ทิศทาง เป็น 1 หมายถึง เดินหน้า
ทิศทาง เป็น 0 หมายถึง ถอยหลัง

เนื่องจากรูปแบบการเคลื่อนที่ดังกล่าว จะเห็นว่าสามารถมีค่าตั้งแต่ 0x00 จนถึง 0xFF ซึ่งข้อมูลดังกล่าวอาจซ้ำกับคำสั่งอื่นๆได้ ดังนั้น จึงต้องมีการส่งรหัสไป 1 ไบต์ เพื่อบอกว่าจะมีการส่งรูปแบบการเคลื่อนที่ก่อน โดยจะต้องไม่ซ้ำกับคำสั่งอื่นๆ ดังนั้น จึงกำหนดให้ มีการส่งค่า 0x00 ก่อน แล้วจึงตามด้วยรูปแบบการเคลื่อนที่ ส่วนการสั่งให้ทำงานอย่างอื่น จะส่งในไบต์เดียว และมี 4 บิตบนเป็น 1111 รูปแบบการส่งคำสั่งจาก Pocket PC ไปยังไมโครคอนโทรลเลอร์ AT89C51 เป็นดังนี้

คำสั่งควบคุมเคลื่อนที่ : 0x00 + รูปแบบการเคลื่อนที่

คำสั่งอื่นๆ :
- 0xF8 สั่งให้ยิงลูกบอล
- 0xF4 สั่งให้เลี้ยงลูกบอล
- 0xF0 สั่งให้หยุดการเลี้ยงลูกบอล

4.1.4.2 การรับคำสั่งของไมโครคอนโทรลเลอร์

เนื่องจากการส่งข้อมูล มีขนาดของข้อมูลมากกว่า 1 ไบต์ ในการรับคำสั่งของ AT89C51 จึงต้องมีการรับข้อมูลแบบเป็น state โดยออกแบบให้มี flow chart ของโปรแกรมส่วนการตอบสนองอินเตอร์รัพท์ ดังนี้

รูปที่ 4.12โฟลว์ชาร์ตของโปรแกรมบน AT89C51 ส่วนตอบสนองอินเตอร์รัพต์

จากคำสั่งควบคุมการยิงและเลี้ยงลูกบอล AT89C51 จะทำการส่งลอจิก 0 หรือ 1 ไปยังวงจรขับมอเตอร์โดยตรง ส่วนคำสั่งควบคุมการเคลื่อนที่ AT89C2051 จะทำหน้าที่ควบคุมมอเตอร์ที่ใช้ในการเคลื่อนที่ได้แก่ เดินหน้า ถอยหลัง เลี้ยวซ้าย เลี้ยวขวา และหยุด ซึ่ง AT89C2051 จะรับรูปแบบการเคลื่อนที่จากพอร์ต 0 ของ AT89C51 ซึ่งใช้รูปแบบเดียวกับที่ส่งมาจาก Pocket PC

4.1.4.3 การสร้างสัญญาณพัลส์ควบคุมการเคลื่อนที่

จากรูปแบบการเคลื่อนที่ ที่รับมาทางพอร์ต 1 ของ AT89C2051 จะนำไปสร้างสัญญาณพัลส์ ที่มีความถี่คงที่ แต่มีความกว้างของพัลส์ ตามระดับความเร็ว เนื่องจากระดับความเร็วของมอเตอร์แต่ละข้างมีขนาด 3 บิต จึงสามารถปรับความกว้างของพัลส์ (Pulse width modulated) ได้เป็นระดับความเร็วต่างๆกัน 8 ระดับสำหรับการเคลื่อนที่ โดยกำหนดให้แต่ละระดับ มีค่าดิวตี้ไซเคิลดังตารางที่ 4.2

ตารางที่ 4.2 ดิวตี้ไซเคิลของพัลส์ควบคุมมอเตอร์ที่ระดับความเร็วต่างๆ

ระดับความเร็ว	ดิวตี้ไซเคิล (%)
0	0
1	57.16
2	64.30
3	71.44
4	78.58
5	85.72
6	92.86
7	100

โฟลว์ชาร์ตที่ใช้ในการในการสร้างสัญญาณพัลส์ควบคุมการเคลื่อนที่ แสดงดังรูปที่ 4.13 โดยในการเก็บค่าดิวตี้ไซเคิล ถ้าค่าที่ได้รับมาน้อยกว่า 0 จะทำการบวก 8 เพิ่มเข้าไป เพื่อให้ระดับความเร็วตั้งแต่ 1 – 7 มีดิวตี้ไซเคิลมากกว่า 50 เปอร์เซ็นต์ ทั้งนี้เนื่องจากการขับมอเตอร์โดยใช้ดิวตี้ไซเคิลต่ำเกินไป (ต่ำกว่า 50 เปอร์เซ็นต์) จะทำให้มอเตอร์มีกำลังต่ำ ไม่สามารถขับเคลื่อนหุ่นยนต์ได้

รูปที่ 4.13โฟลว์ชาร์ตการสร้างสัญญาณพัลส์ควบคุมการเคลื่อนที่

4.2 การออกแบบโปรแกรมสำหรับหุ่นยนต์ผู้รักษาประตู

4.2.1 การจับภาพจากกล้องWebCam

ใช้หลักการเดียวกับกล้องที่ใช้กับ Pocket PC คือ เมื่อกล้องจับภาพได้แต่ละเฟรม ก็จะเรียกใช้งานฟังก์ชัน CaptureDone การประมวลผลภาพและการคำนวณการเคลื่อนที่ของหุ่นยนต์ต่างๆ ก็จะทำในฟังก์ชันนี้ โดยภาพที่ได้จากกล้องมีขนาด 320x240 พิกเซล นำมาย่อลง 4 เท่า จะเหลือภาพที่นำมาประมวลผลมีขนาด 160x120 พิกเซล

4.2.2 การประมวลผลภาพ

ใช้หลักการเดียวกับการประมวลผลใน Pocket PC คือใช้การแยกช่วงสีโดยสีของลูกบอล กับหุ่นผู้รักษาประตูจะมีสีต่างกัน ลูกบอลที่ใช้มีสีส้ม สำหรับลูกบอลกำหนดช่วงในการแยกคือ

$$R > 1.5\,G$$

$$G > B$$

$$R > 150$$

ส่วนหุ่นยนต์ผู้รักษาประตูซึ่งจะมีกล่องสีดำครอบและมีวงกลมสีน้ำเงินอยู่ด้านบน กำหนดช่วงสีดังนี้คือ

$$|G - R| < 15$$

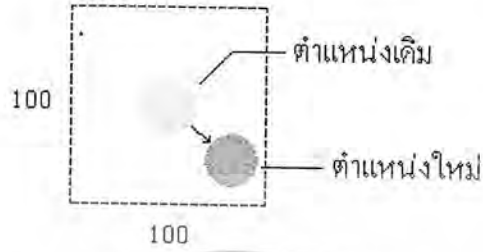
$$B > 1.2\,G$$

$$B > 70$$

หลังจากทำการแยกสีได้แล้วจึงทำการหาตำแหน่งจุดศูนย์กลางภาพ โดยใช้สมการเช่นเดียวกับการประมวลผลใน Pocket PC

ในโครงงานนี้ได้แบ่งสนามออกเป็น 3 ส่วน ด้วยการกำหนดเขตในโปรแกรมคือ สนาม (Field) เขตผู้ประตู (GK Area) และประตู (Goal) การค้นหาลูกบอลจะทำการค้นหาในเขตผู้รักษาประตูก่อน ถ้าไม่พบจึงจะทำการค้นหาในเขตสนาม ส่วนการค้นหาหุ่นยนต์ผู้รักษาประตูนั้นจะค้นหาในเขตผู้รักษาประตูเท่านั้น และการค้นหาวัตถุ (ลูกบอล และหุ่นยนต์) จะทำในบริเวณเดิม

ที่เคยพบวัตถุก่อน โดยกำหนดให้ทำการค้นหาในสี่เหลี่ยมขนาด 100*100 ที่มีจุดศูนย์กลางเป็นตำแหน่งเดิมที่พบวัตถุดังแสดงในรูปที่ 4.14 ทั้งนี้เพื่อสามารถประมวลผลได้อย่างรวดเร็ว

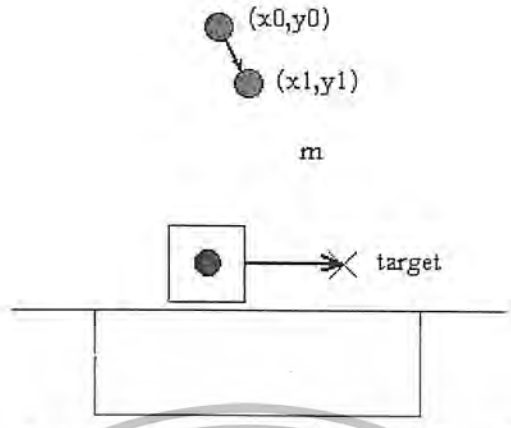


รูปที่ 4.14 การค้นหาวัตถุ(ลูกบอล หรือ หุ่นยนต์)

4.2.3 ปัญญาประดิษฐ์

หุ่นยนต์ผู้รักษาประตูมีหน้าที่ในการป้องกันประตูซึ่งจะเคลื่อนที่อยู่ภายในบริเวณเขตประตูเท่านั้น โดยจะทำการคำนวณหาพิกัดที่ลูกบอลจะเคลื่อนที่มายังประตูแล้วเคลื่อนที่ไปยังพิกัดนั้น ถ้าหากพิกัดนั้นอยู่นอกประตู หุ่นยนต์ผู้รักษาประตูจะเคลื่อนที่ไปยังขอบของประตู่ ซึ่งการเคลื่อนที่ของหุ่นจะเคลื่อนที่ในแนวแกน x เพียงอย่างเดียว รูปที่ 4.15 แสดงโฟลว์ชาร์ตการตัดสินใจ และรูปที่ 4.16 แสดงวิธีการคำนวณหาเป้าหมายการเคลื่อนที่ของหุ่นยนต์ผู้รักษาประตู

รูปที่ 4.15 โฟลว์ชาร์ตการตัดสินใจของหุ่นยนต์ผู้รักษาประตู



รูปที่ 4.16 การคำนวณหาเป้าหมายการเคลื่อนที่ของหุ่นยนต์ผู้รักษาประตู

การหาพิกัดของลูกบอลที่เคลื่อนที่มายังประตูสามารถคำนวณ ทำได้โดยการหาสมการเส้นตรงตามแนวการเคลื่อนที่ของลูกบอลดังนี้

จากสมการเส้นตรง

$$y = m*x + c$$

หาความชัน m จากตำแหน่งที่เปลี่ยนไปของลูกบอล กำหนดให้เป็นจาก (x0,y0) ไปยัง (x1,y1) จะได้

$$m = (y_1-y_0) / (x_1-x_0) \qquad \text{สมการที่ 4.3}$$

ค่าคงที่ c หาได้โดยแทนค่า x1 และ y1 ในสมการเส้นตรง จะได้

$$c = y_1 - m*x_1 \qquad \text{สมการที่ 4.4}$$

จากนั้นแทนค่า y เป็นตำแหน่งของหุ่นผู้รักษาประตู จะได้ตำแหน่งในแกน x คือเป้าหมายที่จะต้องเคลื่อนที่ไป (target) คือ

$$x = (y - c) / m \qquad \text{สมการที่ 4.5}$$

หลังจากคำนวณเป้าหมายการเคลื่อนที่ได้แล้ว ก็จะนำระยะห่างจากตำแหน่งปัจจุบันกับเป้าหมาย มากำหนดความเร็ว โดยใช้ความเร็ว 8 ระดับ โดยหาระดับความเร็วได้ดังนี้

ระดับความเร็ว = 8 * ระยะห่าง / ความกว้างของปากประตู

จากวิธีการดังกล่าวจะเห็นว่ายิ่งระยะห่างมากก็จะวิ่งด้วยความเร็วสูง แต่เมื่อเข้าใกล้เป้าหมายมากขึ้น จะวิ่งด้วยความเร็วลดลง ทั้งนี้เพื่อลดการกระชากของมอเตอร์ ซึ่งจะส่งผลให้การเคลื่อนที่เอียงได้ง่ายขึ้น

4.2.4 การออกแบบหน้าจอแสดงผล

รูปที่ 4.17 หน้าจอแสดงผลบนคอมพิวเตอร์

หน้าจอแสดงผลแสดงดังรูป 4.17 ประกอบด้วยภาพที่ได้จากกล้อง ภาพผลการแยกวัตถุ ตำแหน่งของวัตถุต่างๆ รวมทั้งขอบเขตของเขตในสนามทั้ง 3 ส่วน และเมนูควบคุมการทำงาน

4.2.4 การควบคุมหุ่นยนต์

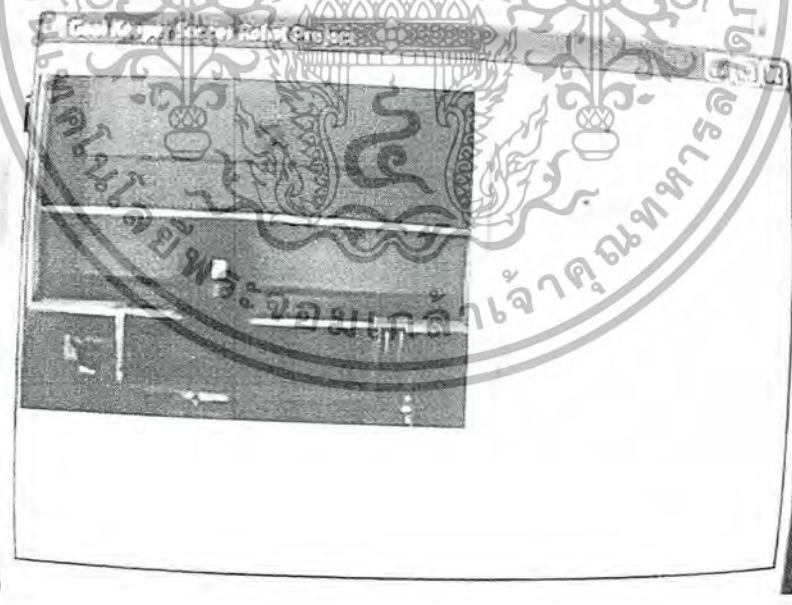
ใช้ระบบการควบคุมเช่นเดียวกับหุ่นยนต์เตะฟุตบอล แต่การส่งข้อมูล ใช้การส่งจากคอมพิวเตอร์ ไปยัง AT89C2051 โดยตรง

บทที่ 5
การทดลอง

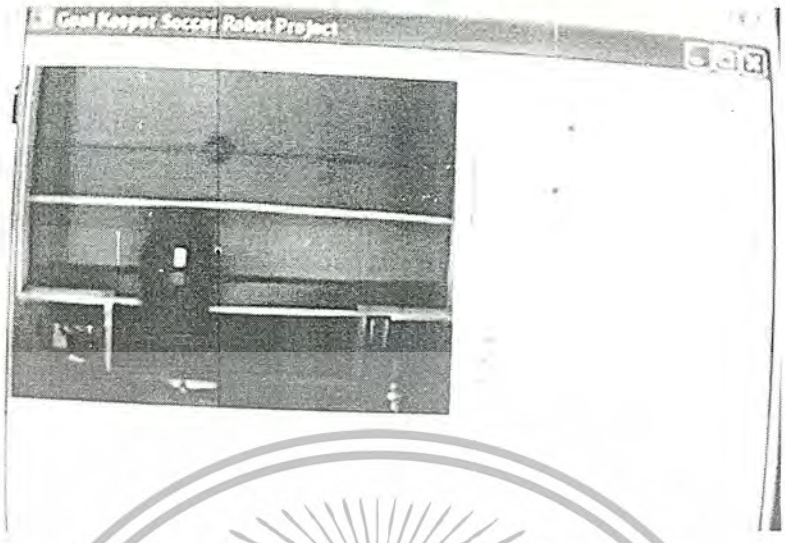
5.1 การทดลองส่วนโปรแกรม

ในส่วนของหุ่นยนต์เตะฟุตบอล จากการทดลองนำภาพมาค้นหาตำแหน่งของลูกบอล พบว่า สามารถทำการแยกส่วนภาพได้ดีเมื่อความสว่างคงที่ แต่เมื่อแสงภายนอกน้อยเกินไป หรือมากเกินไป การแยกภาพจะลดประสิทธิภาพลงไป สามารถแก้ไขได้โดยการควบคุมความสว่างให้คงที่ด้วยการติดหลอดไฟไว้เหนือสนาม หรืออาจใช้การเปลี่ยนแปลงช่วงตามสภาพแสงก็จะทำให้ทำการประมวลผลได้ดีขึ้น

ในส่วนของหุ่นยนต์ผู้รักษาประตู จากการทดลอง เมื่อทำให้ลูกบอลเคลื่อนที่เข้าหาประตูในทิศทางต่างๆกันพบว่า หุ่นยนต์ผู้รักษาประตูสามารถเคลื่อนที่ไปป้องกันประตูในตำแหน่งที่ถูกต้องตามที่ออกแบบไว้ แสดงดังรูปที่ 5.1- 5.3 แต่ในบางครั้งการเคลื่อนที่ของหุ่นยนต์เอียงไปจากแนวแกนX



รูปที่ 5.1การทดลองหุ่นยนต์ผู้รักษาประตู 1



รูปที่ 5.2การทดลองหุ่นยนต์ผู้รักษาประตู 2

รูปที่ 5.3การทดลองหุ่นยนต์ผู้รักษาประตู 3

5.2 การทดลองส่วนวงจร

จากการทดลองส่วนของวงจรควบคุม ซึ่งรับข้อมูลอนุกรมมาจาก Pocket PC หรือ คอมพิวเตอร์ พบว่าสามารถรับคำสั่งได้เป็นอย่างดี และ AT89C51 สามารถ ส่งรูปแบบการเคลื่อนที่ให้แก่ AT89C2051 ได้อย่างถูกต้อง ในส่วนของการควบคุมความเร็วมอเตอร์ สามารถปรับระดับได้ 8 ระดับ เมื่อทำการทดลองวัดความเร็วในการเคลื่อนที่ของหุ่นยนต์เตะฟุตบอล ที่ระดับต่างๆ พบว่า

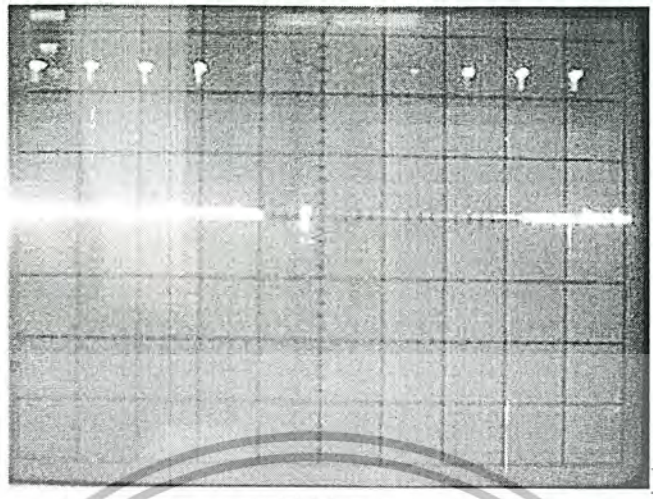
มีความเร็วในหน่วย เซนติเมตรต่อวินาทีดังตาราง 5.1 สำหรับความเร็วหุ่นยนต์ผู้รักษาประตู สามารถปรับระดับได้ 6 ระดับดังตารางที่ 5.2

ตารางที่ 5.1 ความเร็วของหุ่นยนต์เตะฟุตบอลที่ระดับความเร็วต่างๆ

ระดับความเร็ว	ความเร็ว (cm/s)
0	0
1	11.46
2	13.26
3	14.50
4	15.48
5	17.27
6	19.67
7	22.97

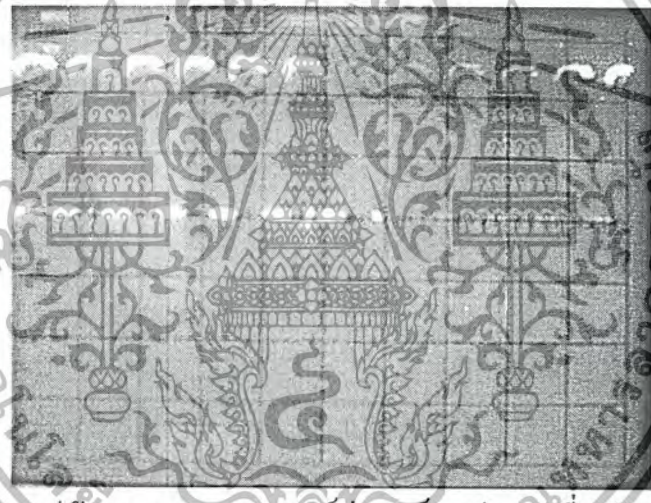
ตารางที่ 5.2 ความเร็วของหุ่นยนต์ผู้รักษาประตูที่ระดับความเร็วต่างๆ

ระดับความเร็ว	ความเร็ว (cm/s)
2	20.68
3	28.80
4	35.26
5	55.00
6	58.51
7	70.51

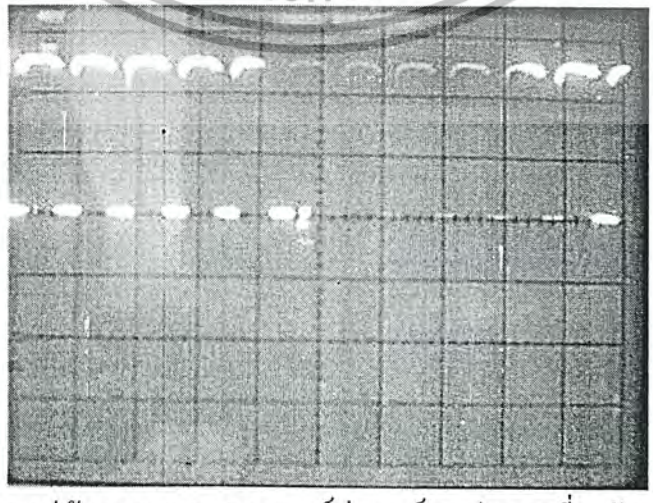


]

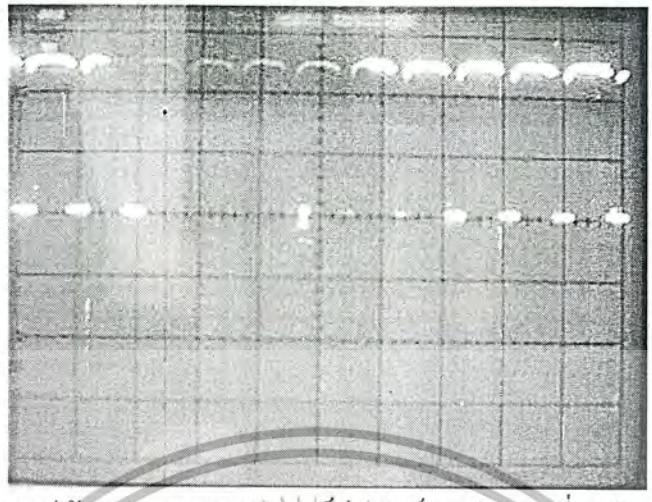
รูปที่ 5.4 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 0



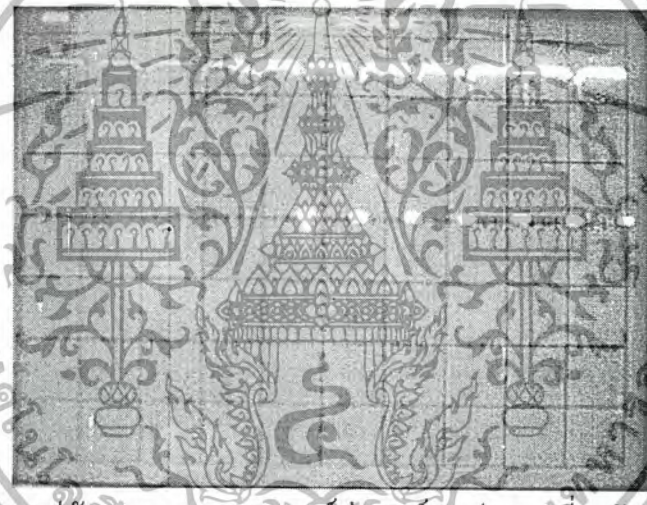
รูปที่ 5.5 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 1



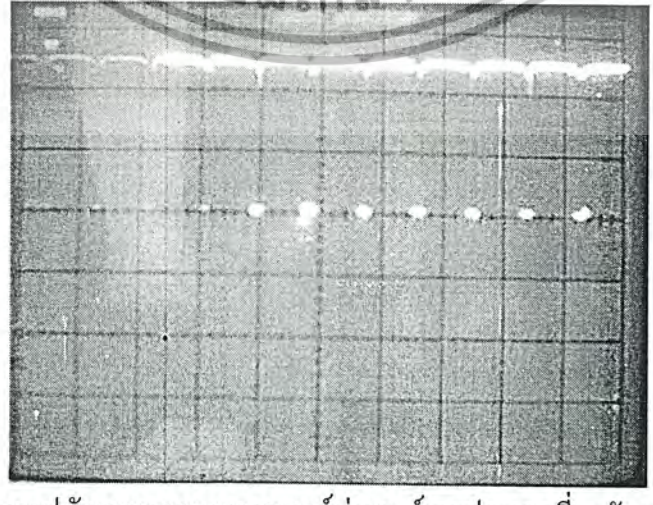
รูปที่ 5.6 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 2



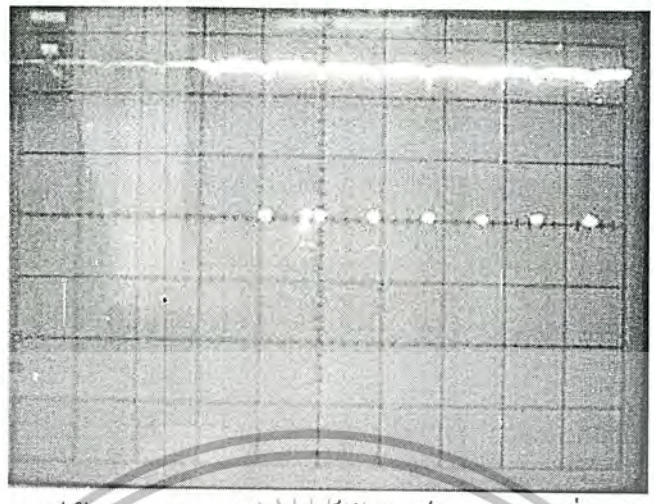
รูปที่ 5.7 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 3



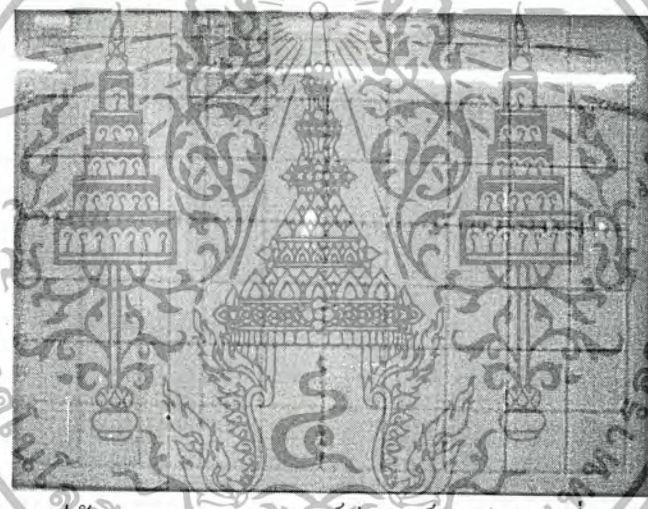
รูปที่ 5.8 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 4



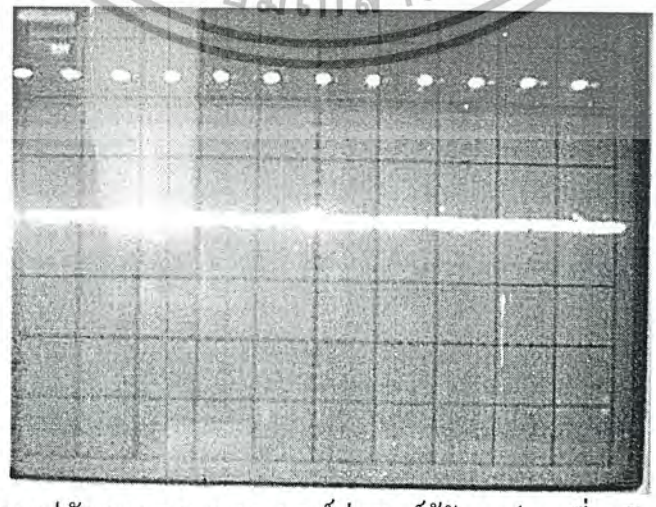
รูปที่ 5.9 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 5



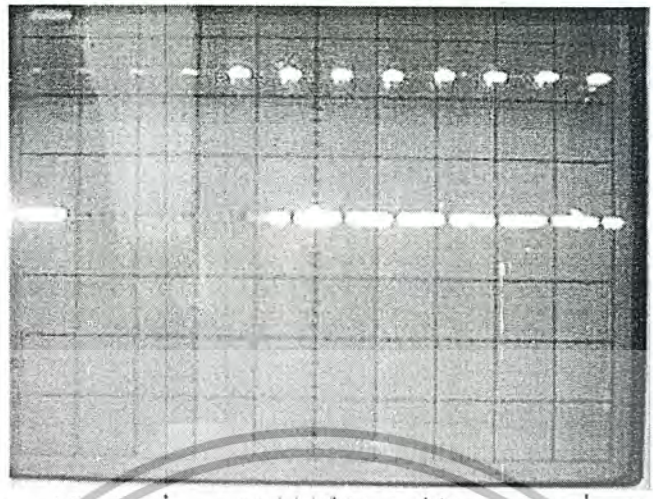
รูปที่ 5.10 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 6



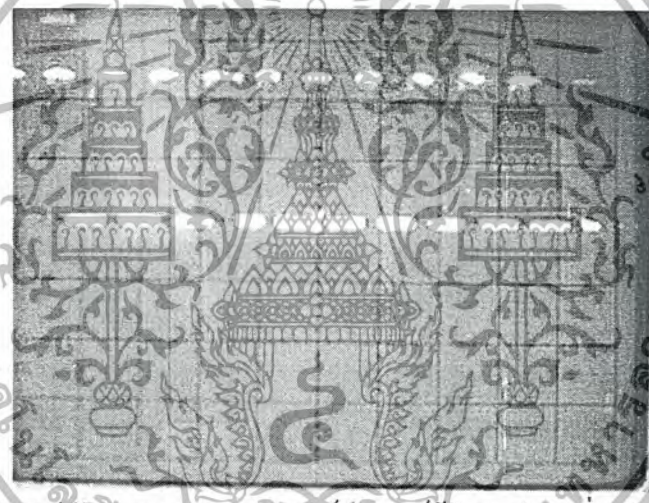
รูปที่ 5.11 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์เตะฟุตบอลที่ระดับความเร็ว 7



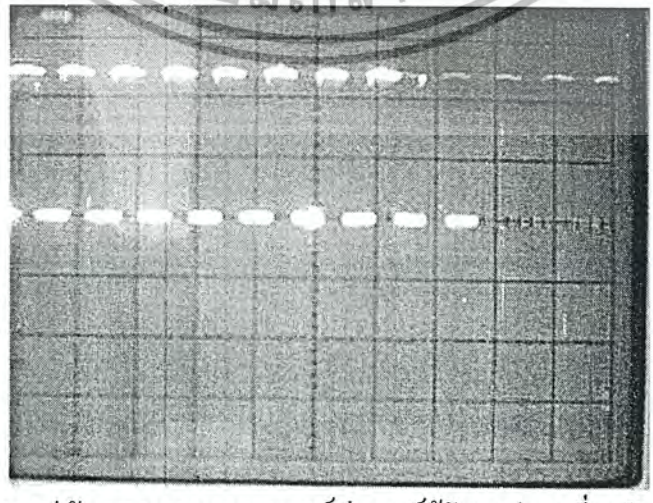
รูปที่ 5.12 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 0



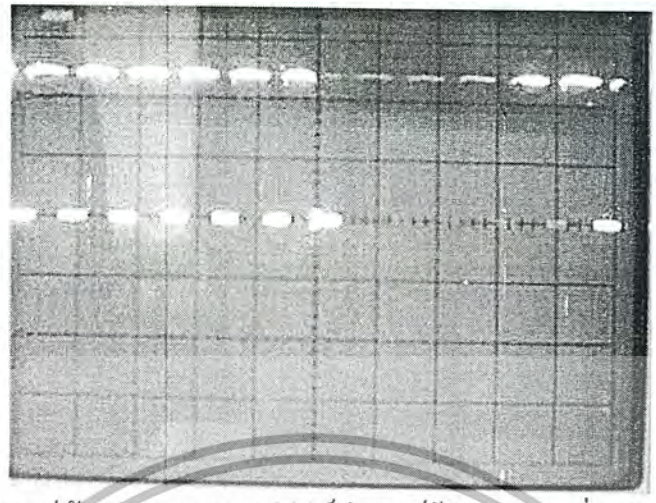
รูปที่ 5.13 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 1



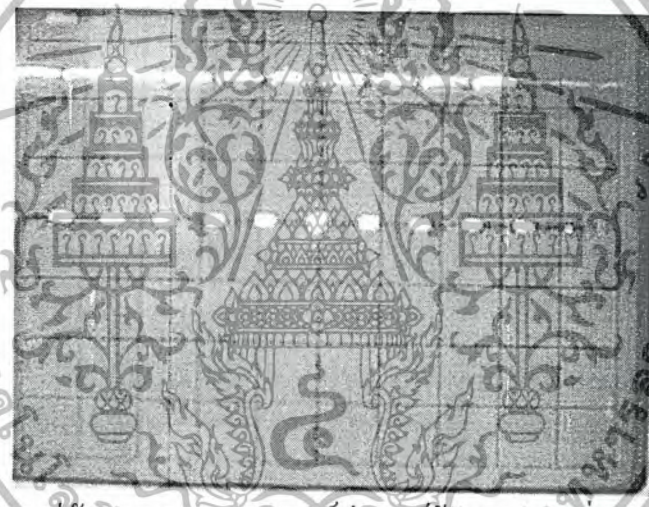
รูปที่ 5.14 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 2



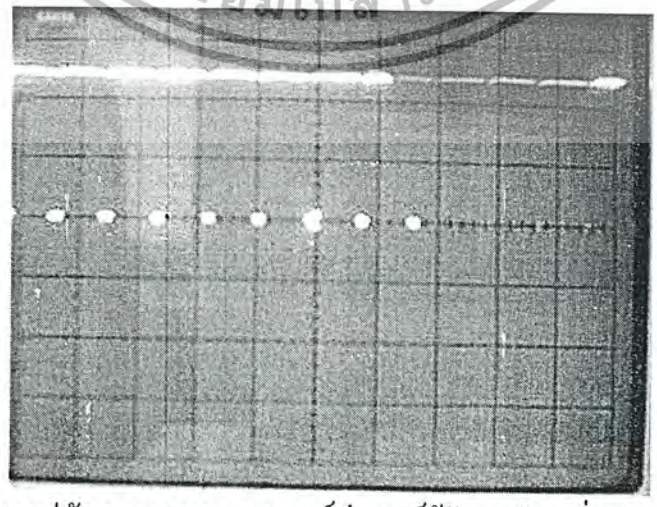
รูปที่ 5.15 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 3



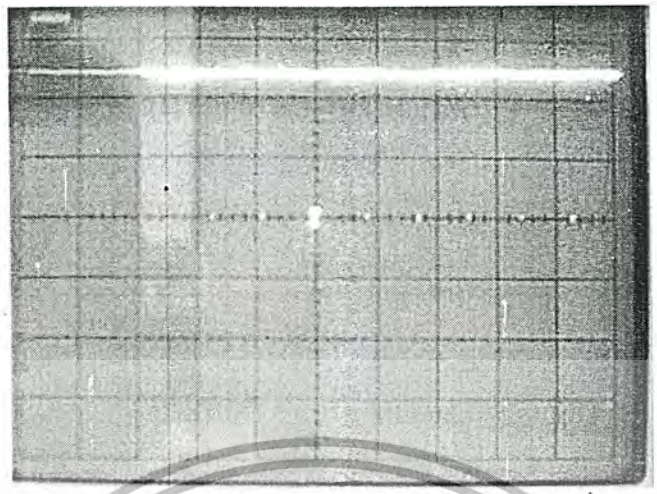
รูปที่ 5.16 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 4



รูปที่ 5.17 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 5



รูปที่ 5.18 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 6



รูปที่ 5.19 กราฟสัญญาณควบคุมมอเตอร์หุ่นยนต์ผู้รักษาประตูที่ระดับความเร็ว 7

บทที่ 6
สรุปและวิเคราะห์ผลการดำเนินงาน

6.1 สรุปผลการทดลอง

สำหรับหุ่นยนต์เตะฟุตบอล

1. สามารถค้นหาลูกบอลได้ เมื่อมีแสงสว่างเพียงพอ
2. สามารถวิ่งเข้าหาลูกบอลเป็นแนวโค้งได้
3. สามารถเลี้ยงลูกบอลได้ ถ้าระดับแรงดันไฟที่จ่ายสูงพอ
4. สามารถยิงประตูเมื่อเข้าใกล้ประตูได้
5. ความเร็วในการประมวลผลประมาณ 8.2 เฟรมต่อวินาที

สำหรับหุ่นยนต์ผู้รักษาประตู

1. สามารถค้นหาลูกบอลได้ เมื่อมีแสงสว่างเพียงพอ
2. สามารถเคลื่อนที่ไปยังเป้าหมายได้
3. สามารถป้องกันประตูได้ ถ้าลูกบอลเคลื่อนที่ไม่เร็วนัก

6.2 ปัญหาที่พบระหว่างดำเนินงาน

1. การออกแบบโครงสร้างของหุ่นยนต์เป็นไปด้วยความล่าช้า เนื่องจากไม่สามารถหาวัสดุและมอเตอร์ที่เหมาะสมได้

2. กล้องที่ใช้เป็นกล้องแบบติดตั้งบนเครื่อง Pocket PC ทำให้ไม่สามารถปรับมุมมองของกล้องได้มากนัก จึงไม่สามารถมองเห็นลูกบอลที่อยู่ใกล้หุ่นยนต์มากๆ ได้

3. ภาษา C# เป็นภาษาใหม่ จึงยังไม่แพร่หลายนัก อีกทั้งเป็นการพัฒนาโปรแกรมบน Pocket PC จึงมีการใช้งานบางอย่างไม่เหมือนบนพีซี นอกจากนี้ ยังไม่มีแหล่งความรู้ทางด้านนี้ในประเทศไทยมากนัก

4. ปัญหาเกี่ยวกับมอเตอร์และล้อของหุ่นผู้รักษาประตู ทำให้หุ่นยนต์เคลื่อนที่เอียงไปจากที่กำหนด และวิ่งออกนอกเขตผู้รักษาประตูได้

6.3 แนวทางการพัฒนาต่อ

1. ปรับปรุงระบบการเคลื่อนที่ เช่น ใช้การเคลื่อนที่แบบ Omni Direction จะทำให้สามารถควบคุมการเคลื่อนที่ให้มีประสิทธิภาพดียิ่งขึ้น

2. พัฒนาระบบปัญญาประดิษฐ์ เพื่อใช้ในการตัดสินใจได้อย่างฉลาดขึ้น

3. ปรับปรุงเทคนิคที่ใช้ในการประมวลผล เพื่อให้มีประสิทธิภาพมากขึ้น และสามารถประมวลผลได้เร็วขึ้น

4.เพิ่มจำนวนหุ่นยนต์ให้มีการแข่งเป็นทีม

เอกสารอ้างอิง

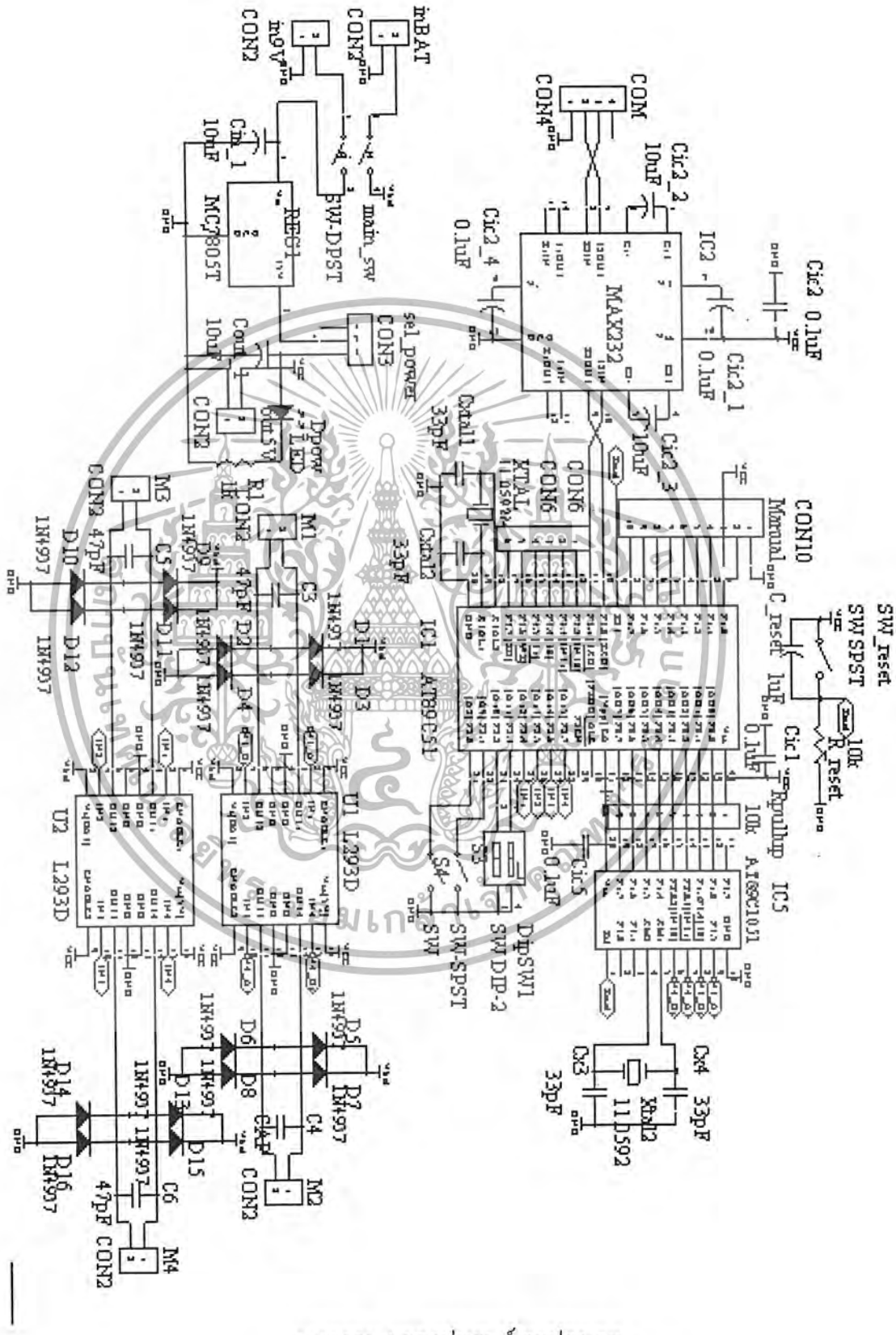
1. นิรันด์ ประวิทย์ธนา, "เก่ง C# ให้ครบสูตร" , บ. วิตตี้กรุ๊ป จำกัด, 492 หน้า, 2545

2. ศุภชัย สมพานิช, "คู่มือการเขียนโปรแกรมและใช้งาน Visual C#.NET ฉบับสมบูรณ์" , อินโฟเพรส, 596 หน้า, 2546

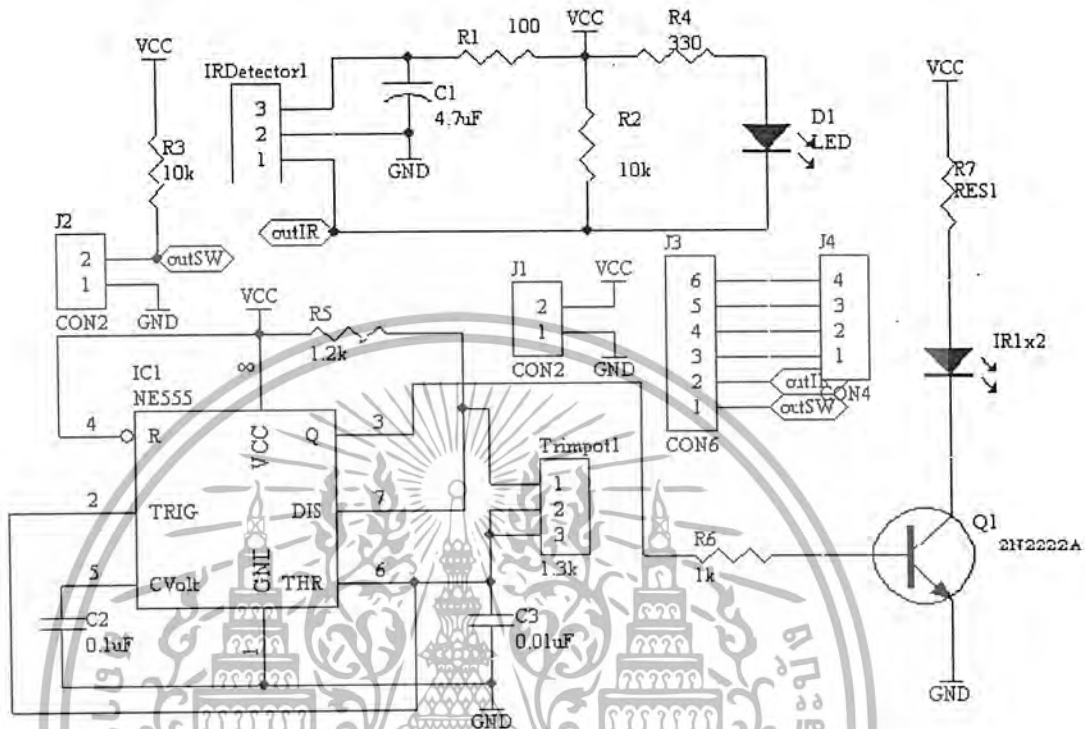
3. ทรงเกียรติ ภาวดี, "คู่มือการใช้ Pocket PC" , ซีเอ็ดยูเคชั่น, 384 หน้า, 2545

4. Rafael C. Gonzalez, Richard E. Woods, "Digital Image Processing" , PEARSON EDUCATION, 793 p., 2002

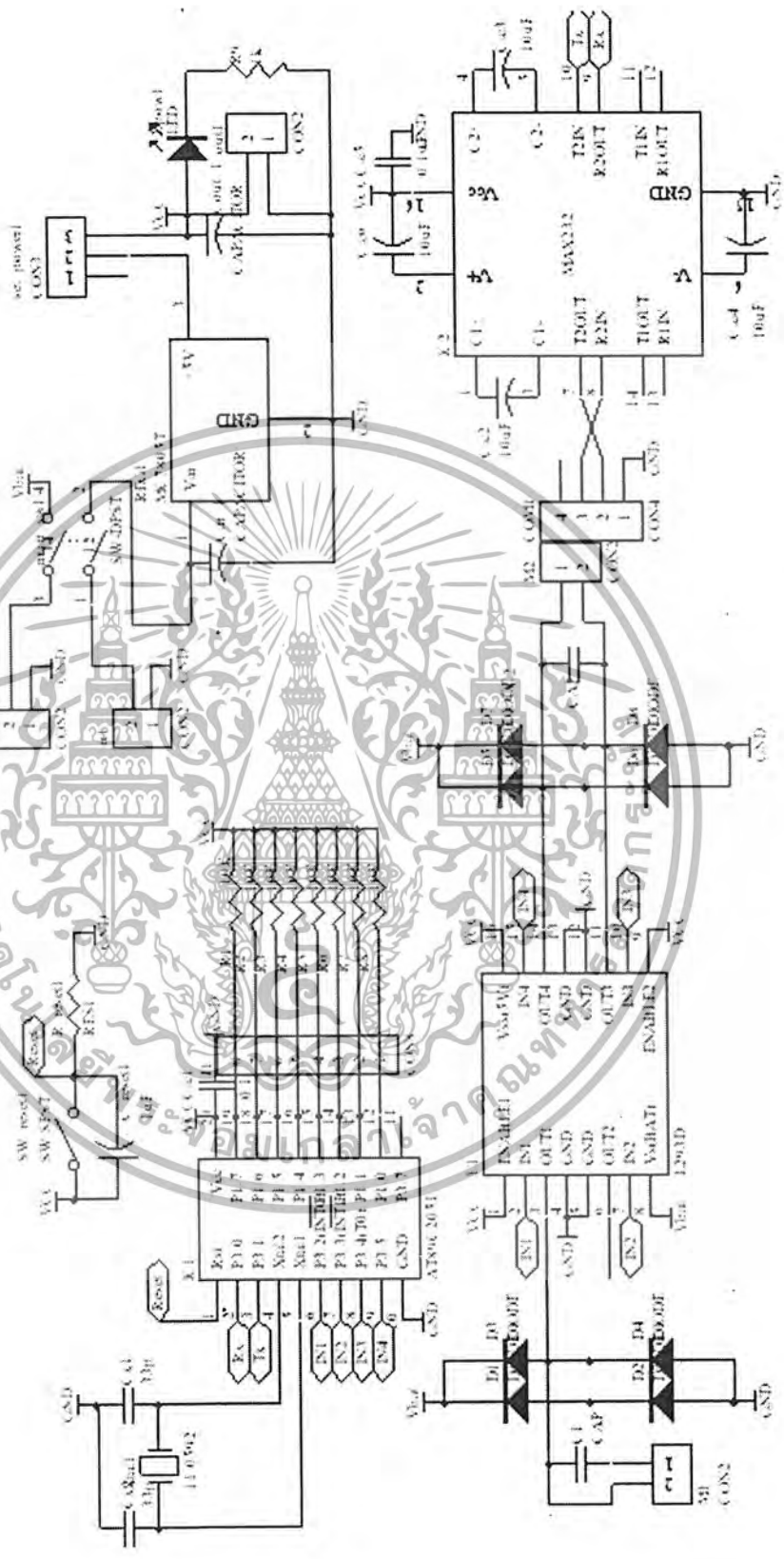
ภาคผนวก ก รูปโครงงานหุ่นยนต์เตะฟุตบอล



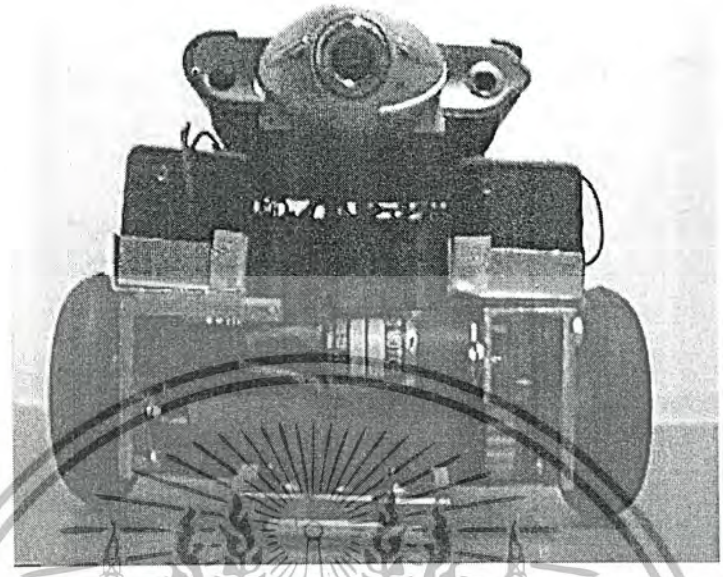
วงจรรวม 1 ของหุ่นยนต์เตะฟุตบอล



วงจรรวม 2 ของหุ่นยนต์เตะฟุตบอล

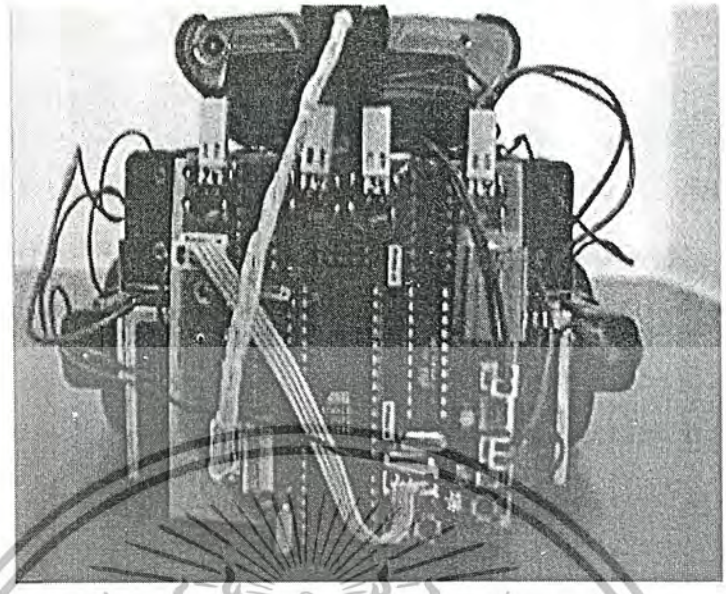


วงจรของหุ่นยนต์ผู้รักษาประตู

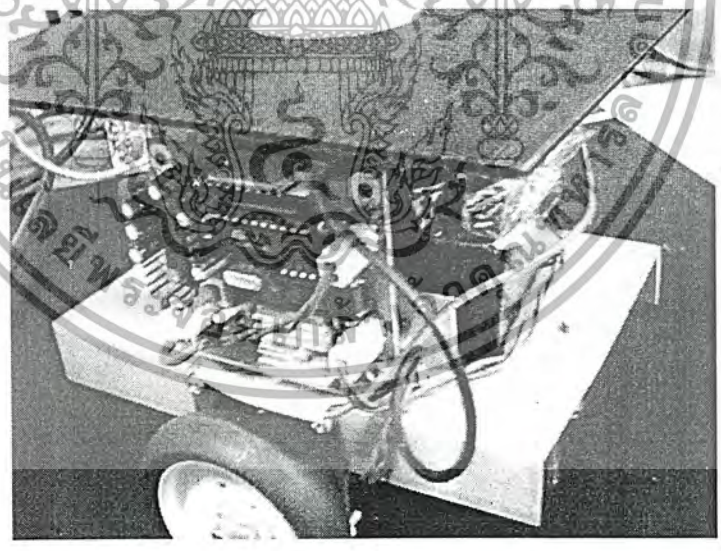


รูปหุ่นยนต์เตะฟุตบอล ด้านหน้า

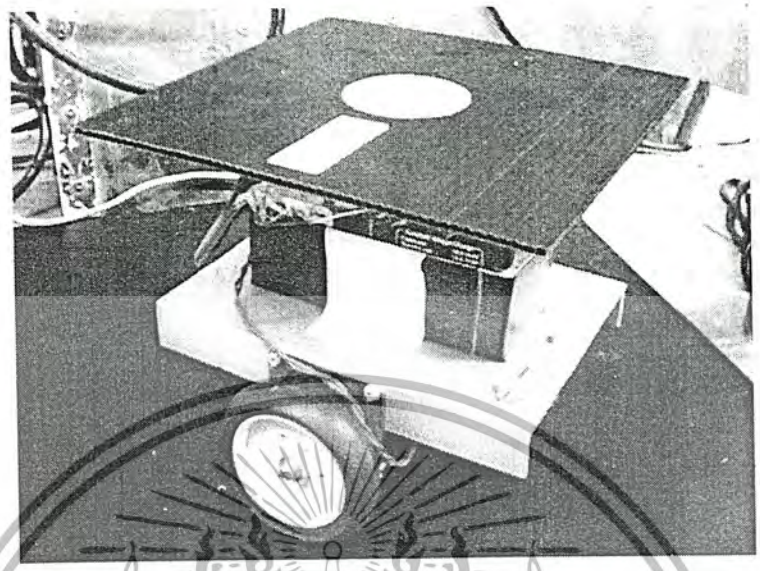
รูปหุ่นยนต์เตะฟุตบอล ด้านบน



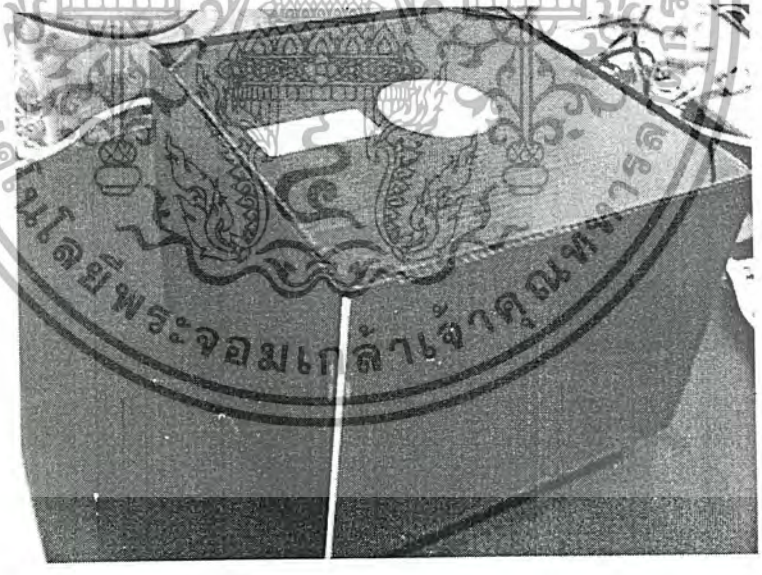
รูปหุ่นยนต์เตะฟุตบอล ด้านหลัง



รูปหุ่นยนต์ผู้รักษาประตู ด้านหลัง



รูปหุ่นยนต์ผู้รักษาประตู ด้านหน้า



รูปหุ่นยนต์ผู้รักษาประตูเมื่อถูกครอบด้วยกล่องดำ

ภาคผนวก ข ส่วนโปรแกรม

```
//main.c
#include <AT89X51.H>
#include <intrins.h>

#define move_dat 0x0f
#define func_dat 0x0f0
// Selector port
sbit reset1 = P2^0 ;
sbit reset2 = P2^1 ;
sbit sel_1  = P2^2 ;
sbit sel_2  = P2^3 ;
// Manual port 51
sbit shoot  = P1^7;
sbit back   = P1^5;
sbit right  = P1^3;
sbit left   = P1^1;
sbit forward= P1^0;
sbit roller = P1^6;
sbit F1     = P1^4;
sbit F2     = P1^2;
// Output port motor
sbit m3a    = P2^4;
sbit m3b    = P2^5;
sbit m4a    = P2^6;
sbit m4b    = P2^7;

bit state;

//    x  x x x       x   x x x
//    motor left       motor right
// direct speed     direct speed
//  0=clockwise  000=min  111=max

char new_cod = 0;
void init_serial()
{
        TMOD = 0x20;
        TH1 = 0xFD;                 //9600 bps
        SCON = 0x50;
        IE = 0x90;                  //enable serial interrupt
```

```
        TR1 = 1;
        RI = 0;
}

void init_motor()
{
        m3a = 0;
        m3b = 0;
        m4a = 0;
        m4b = 0;
}

void init_2051()
{
        P0 = 0x00;
}

void shoot_fn(bit x)
{
        if (x) m3a = 1;
        else m3a = 0;
}

void dribbling_fn(bit x)
{
        if (x) m4a = 1;
        else m4a = 0;
}
void func_do(unsigned char dat)
{
        bit x;
        dat = dat&0x0f;
        if (dat&0x08 == 0x08) x = 1; else x = 0;
        shoot_fn(x);
        if (dat&0x04 == 0x04) x = 1; else x = 0;
        dribbling_fn(x);
}

void serial_do(unsigned char dat)
{
```

```
        if (state == 0)
        {
                if (dat == 0x0f) state = 1;
                else func_do(dat);
        }
        else
        {
                P0 = dat;
                state = 0;
        }
}

void serial() interrupt 4
{
        unsigned char dat;
        if (RI == 1)
        {
                dat = SBUF;
                RI = 0;
                serial_do(dat);
        }
}
void Auto()
{
        init_serial();
}

void manual()
{
        unsigned char dat;
        if (back    == 1) dat = 0x77;    else//backward
        if (right   == 1) dat = 0x7f;    else//right
        if (left    == 1) dat = 0x0f7;   else//left
        if (forward == 1) dat = 0x0ff;   else//forward
                dat = 0x00;
        P0 = dat;
        shoot_fn(shoot);       //shoot
        dribbling_fn(roller);//roller
}
```

```
void main()
{
        init_motor();
        init_2051();
        while (1)
        {
                if (sel_1 == 0) manual();
                else Auto();
        }
}
```

```
//pwm.c
#include <AT892051.H>
#include <intrins.h>
sbit m1a = P3^4;                    //m1 = right motor
sbit m1b = P3^5;
sbit m2a = P3^3;                    //m2 = left motor
sbit m2b = P3^2;
sbit di1 = P1^7;
sbit di2 = P1^3;
//              P1
//              di1[1] duty[3] di2[1] duty[3]
unsigned char duty[4];

void delay(unsigned char time)
{
        unsigned char i,j;
        for (i = 0; i < time; i++)
        {
            for (j = 0; j < 10; j++) _nop_();
        }
}

void clr_motor()
{
        m1a = 0;
        m1b = 0;
        m2a = 0;
        m2b = 0;
}
void in()
{
        unsigned char speed;
//      motor1
        speed = _cror_(P1&0x70,4);
        if (speed != 0)
        {
                if (di1 == 0)
                {
                        duty[0] = speed+8;
                        duty[1] = 0;
```

```
                }
                else
                {
                        duty[1] = speed+8;
                        duty[0] = 0;
                }
        }
        else
        {
                duty[0] = 0;
                duty[1] = 0;
        }
//      motor2
        speed = P1&0x07;
        if (speed != 0)
        {
                if (di2 == 0)
                {
                        duty[2] = speed-8;
                        duty[3] = 0;
                }
                else
                {
                        duty[3] = speed-8;
                        duty[2] = 0;
                }
        }
        else
        {
                duty[2] = 0;
                duty[3] = 0;
        }
}

void main()
{
        char i;
        unsigned char dat = 0x00;
        while (1)
        {
```

```
        clr_motor();
        for (i=15;i>=0;i--)
        {
                if (i == duty[0]) m1a = 1;
                if (i == duty[1]) m1b = 1;
                if (i == duty[2]) m2a = 1;
                if (i == duty[3]) m2b = 1;
                delay(10);
        }
        if (P1 != dat)
        {
                in();
                dat = P1;
        }
}
```

```
using System;
using System.Drawing;
using System.Collections;
using System.Windows.Forms;
using System.Data;

namespace SoccerBot2
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		private static System.Drawing.Bitmap bmp/*,binary*/;
		private static ImageProcessing imagep = new ImageProcessing();
		private static bool camopened;
		private static Robot robot = new Robot();
		private System.Windows.Forms.PictureBox disPicbox;
		private System.Windows.Forms.MenuItem startMenu;
		private System.Windows.Forms.MenuItem stopMenu;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.TextBox mesTextbox;
		private System.Windows.Forms.PictureBox binPicbox;
		private System.Windows.Forms.MainMenu mainMenu1;

		public Form1()
		{
			InitializeComponent();
			camopened = false;
		}
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		protected override void Dispose( bool disposing )
		{
			base.Dispose( disposing );
		}
		#region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
```

```
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        this.mainMenu1 = new System.Windows.Forms.MainMenu();
        this.startMenu = new System.Windows.Forms.MenuItem();
        this.stopMenu = new System.Windows.Forms.MenuItem();
        this.disPicbox = new System.Windows.Forms.PictureBox();
        this.label1 = new System.Windows.Forms.Label();
        this.mesTextbox = new System.Windows.Forms.TextBox();
        this.binPicbox = new System.Windows.Forms.PictureBox();
        //
        // mainMenu1
        //
        this.mainMenu1.MenuItems.Add(this.startMenu);
        this.mainMenu1.MenuItems.Add(this.stopMenu);
        //
        // startMenu
        //
        this.startMenu.Text = "Start";
        this.startMenu.Click += new System.EventHandler(this.startMenu_Click);
        //
        // stopMenu
        //
        this.stopMenu.Text = "Stop";
        this.stopMenu.Click += new System.EventHandler(this.stopMenu_Click);
        //
        // disPicbox
        //
        this.disPicbox.Location = new System.Drawing.Point(48, 0);
        this.disPicbox.Size = new System.Drawing.Size(160, 120);
        //
        // label1
        //
        this.label1.Location = new System.Drawing.Point(8, 240);
        this.label1.Size = new System.Drawing.Size(100, 16);
        this.label1.Text = "Message:";
        //
        // mesTextbox
        //
```

```
            this.mesTextbox.BackColor = System.Drawing.SystemColors.Window;
            this.mesTextbox.Location = new System.Drawing.Point(72, 240);
            this.mesTextbox.ReadOnly = true;
            this.mesTextbox.Size = new System.Drawing.Size(168, 22);
            this.mesTextbox.Text = "Click \'Start\' to begin";
            //
            // binPicbox
            //
            this.binPicbox.Location = new System.Drawing.Point(104, 160);
            this.binPicbox.Size = new System.Drawing.Size(40, 30);
            //
            // Form1
            //
            this.BackColor = System.Drawing.Color.Aqua;
            this.Controls.Add(this.binPicbox);
            this.Controls.Add(this.mesTextbox);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.disPicbox);
            this.Menu = this.mainMenu1;
            this.Text = "SoccerBot Final";
            this.Load += new System.EventHandler(this.Form1_Load);
        }
        #endregion

        /// <summary>
        /// The main entry point for the application.
        /// </summary>

        static void Main()
        {
            Application.Run(new Form1());
        }

        private void Form1_Load(object sender, System.EventArgs e)
        {

        }
```

```
private void startMenu_Click(object sender, System.EventArgs e)
{
        if (!camopened)
        {
            if (imagep.OpenCam())
            {
                mesTextbox.Text = "Camera Opened";
                camopened = true;
                robot.Connect();
            }
            else mesTextbox.Text = "Can't Open Camera";
        }
        else mesTextbox.Text = "Camera Already Opened";
        loop();
}

private void loop()
{
        imagep.Capture();
        bmp = new Bitmap("test.jpg");
        disPicbox.Image = bmp;
        imagep.BmpToBin();
        binPicbox.Image = imagep.BinToBmp();
        //binPicbox.Image = imagep.BallSegment();
}

private void stopMenu_Click(object sender, System.EventArgs e)
{
        imagep.CloseCam();
        mesTextbox.Text = "Camera Closed";
        robot.Disconnect();
        camopened = false;
}

}
}
```

```
using System;
using eOpenNETCF.IO.Serial;

namespace SoccerBot2
{
    /// <summary>
    /// Summary description for Serial.
    /// </summary>
    public class SerialClass
    {
        public static Port port;
        private DetailedPortSettings portSettings;
        public SerialClass()
        {
            Init();
        }
        ~SerialClass()
        {
            ClosePort();
        }
        private void Init()
        {
            portSettings = new HandshakeNone();
            port = new Port("COM1:", portSettings);
            port.PortName = "COM1:";
            port.Settings.BaudRate = BaudRates.CBR_9600;
            port.Settings.Parity = Parity.none;
            port.Settings.StopBits = StopBits.one;
            port.RThreshold = 1;  // get an event for every 1 byte received
            port.InputLen = 1;                  // calling Input will read 1 byte
            port.SThreshold = 1;  // send 1 byte at a time
//          port.DataReceived +=new Port.CommEvent(Received);
//          this.Closed+=new EventHandler(frmMain_Closed);
        }

        public void OpenPort()
        {
            if(!port.IsOpen) port.Open();
```

```
        }

        public void ClosePort()
        {
            if(port.IsOpen) port.Close();
        }

/*      private void Received()
        {
            // since RThreshold = 1, we get an event for every character
            byte[] inputData = new byte[1];

            // read the character
            inputData = port.Input;

            // display as text
            Encoding enc = Encoding.ASCII;
            displayString = enc.GetString(inputData, 0, inputData.Length);
            try
            {
                this.Invoke(new EventHandler(SetText));
            }
            catch(Exception e)
            {
                Console.WriteLine(e.Message);
            }
        }
*/
        public void Send(byte data)
        {
            byte[] outputData = new byte[1];
            outputData[0] = data;
            port.Output = outputData;
        }
    }
}
```

```
using System;

namespace SoccerBot2
{
        /// <summary>
        /// Summary description for Robot.
        /// </summary>
        public class Robot
        {
                private static SerialClass com1 = new SerialClass();
                public void Connect()
                {
                        com1.OpenPort();
                }

                public void Disconnect()
                {
                        com1.ClosePort();
                }

                public void StartDribbling()
                {
                        com1.Send(0xf4);
                }

                public void StopDribbling()
                {
                        com1.Send(0xf0);
                }

                public void Shoot()             //Stop dribbling automatically
                {
                        com1.Send(0xf8);
                }
                public void StopMove()
                {
                        com1.Send(0x0f);
                        com1.Send(0x00);
                }
                public void Move(Dir dir,byte left_motor_speed,byte right_motor_speed)
```

```
    {
        byte data = 0x00;
        switch (dir)
        {
            case Dir.FORWARD  : data |= 0x88; break;
            case Dir.RIGHT    : data |= 0x08; break;
            case Dir.LEFT     : data |= 0x80; break;
            //case Dir.BACKWARD : data unchange
        }
        for (int i = 0; i <= 15; i++) data += left_motor_speed;
        data += right_motor_speed;
        SendSerial(data);
    }

    private void SendSerial(byte data)
    {
        com1.Send(0x0D);
        com1.Send(data);
    }
}
public enum Dir                    //Direction
{
    FORWARD = 1,
    RIGHT,
    LEFT,
    BACKWARD
};
{
```

```
using System;
using System.Drawing;

namespace SoccerBot2
{
	/// <summary>
	/// Summary description for ImageProcessing.
	/// </summary>
	public class ImageProcessing
	{
		private System.Drawing.Bitmap bmp;
		public int count;
		private VeoCam cam1 = new VeoCam();
		private int width = 40,height = 30;
		private bool[,] binary = new bool[40,30];
		public bool OpenCam()
		{
			return cam1.VeoCamOpen();
		}
		public bool CloseCam()
		{
			return cam1.VeoCamClose();
		}

		public void Capture()
		{
			cam1.VeoCamSnapshot("test.jpg",CamResolution.RES_160);
			bmp = new Bitmap("test.jpg");
		}

/*		public Bitmap BallSegment()
		{
			int i,j;
			Bitmap result = new Bitmap(40,30);
			bool inhypecube;
			byte[] min = {100,0,0};
			byte[] max = {255,80,80};
			for (i = 0; i < 40; i++)
				for (j = 0; j < 30; j++)
```

```
                {
                    byte R,G,B;
                    Color color = bmp.GetPixel(4*i,4*j);
                    R = color.R;
                    G = color.G;
                    B = color.B;
                    inhypecube =
(((R>=min[0])&&(R<=max[0]))&&((G>=min[1])&&(G<=max[1]))&&((B>=min[2])&&(B<=max[2])));
                    if (inhypecube) result.SetPixel(i,j,Color.OrangeRed);
                    else result.SetPixel(i,j,Color.Black);
                }
            return result;
        }
*/
        public void BmpToBin()
        {
            int i,j;
            byte[] min = {20,0,0};
            byte[] max = {255,40,40};
            count = 0;
            for (i = 0; i < width; i++)
                for (j = 0; j < height; j++)
                {
                    byte R,G,B;
                    Color color = bmp.GetPixel(4*i,4*j);
                    R = color.R;
                    G = color.G;
                    B = color.B;
                    binary[i,j] = false;
                    if (R>=min[0])
                        if (R<=max[0])
                            if (G>=min[1])
                                if (G<=max[1])
                                    if (B>=min[2])
                                        if (B<=max[2])
                                        {

                        binary[i,j] = true;

                        count++;
```

```
}

public Bitmap BinToBmp()
{
        int i,j;
        Bitmap bmp = new Bitmap(width,height);
        for (i = 0; i < width; i++)
                for (j = 0; j < height; j++)
                {
                        if (binary[i,j]) bmp.SetPixel(i,j,Color.OrangeRed);
                        //else bmp.SetPixel(i,j,Color.Black);
                }
        return bmp;
}
}
```

```
using System;
using System.Drawing;
using System.Runtime.InteropServices;
using System.Windows.Forms;

namespace SoccerBot2
{
    /// <summary>
    /// Summary description for VeoCam.
    /// </summary>
    public class VeoCam
    {
        private IntPtr m_hCam;

        public bool VeoCamOpen()
        {
            CAM_ERRORS err = CamOpen(out m_hCam);
            return err == CAM_ERRORS.E_NOERR;          //true for no error
        }

        public bool VeoCamClose()
        {
            CAM_ERRORS err = CamClose(m_hCam);
            return err == CAM_ERRORS.E_NOERR;          //true for no error
        }

        public bool VeoCamSnapshot(string filename, CamResolution res)
        {
            CAM_ERRORS err = CamStartSnapshot(m_hCam, filename, res);
            return err == CAM_ERRORS.E_NOERR;          //true for no error
        }

        const int WM_USER           = 0x400;    //  Message from device driver
after one preview frame is available
        const int WM_PREV_END       = WM_USER+10;    //  Message from
device driver after one preview frame is available
        const int WM_PREV_ERR       = WM_USER+11;    // Not used
        const int WM_SNAP_END       = WM_USER+12;    // Not used
        const int WM_SNAP_ERR       = WM_USER+13;
```

```
        [DllImport("veocamapi.dll")]
        static public extern CAM_ERRORS CamOpen(out IntPtr phCam);

        [DllImport("veocamapi.dll")]
        static public extern CAM_ERRORS CamClose(IntPtr hCam);

//      [DllImport("veocamapi.dll")]
//      static public extern CAM_ERRORS CamStartPreview(IntPtr hCam, IntPtr previewhwnd);

//      [DllImport("veocamapi.dll")]
//      static public extern CAM_ERRORS CamPreviewDatatoRGB24(IntPtr hCam, CamResolution resolution,
byte[] buffprev, int length, byte[] buffrgb24);

//      [DllImport("veocamapi.dll")]
//      static public extern CAM_ERRORS CamPreviewData(IntPtr hCam, byte[] buff, out int length);

//      [DllImport("veocamapi.dll")]
//      static public extern CAM_ERRORS CamPreviewDatatoJPG(IntPtr hCam, CamResolution resolution, byte[]
buffprev, int length, byte[] buffjpg, ref int jpglength);

//      [DllImport("veocamapi.dll")]
//      static public extern CAM_ERRORS CamStopPreview(IntPtr hCam);

        [DllImport("veocamapi.dll")]
        static public extern CAM_ERRORS CamStartSnapshot(IntPtr hCam, string filename, CamResolution
resolution);

//      [DllImport("veocamapi.dll")]
//      static public extern CAM_ERRORS CamSetPreviewResolution(IntPtr hCam, CamResolution resolution);

    }

    public enum CAM_ERRORS
    {
        E_NOERR =0,             //No error
        E_NOCAM,                //No Camera
        E_NOMEMORY,             //No enough memory
        E_ERRPARAMETER,         //Input parameter is wrong
        E_NOSUPPORTTED,         //This function is not supported in current driver
```

```
        E_DECODE,                   //Error during decode
        E_OTHERERR                  //Other unknown error
    };

    public enum WhiteBalance
    {
        ENV_AWB         =1,             //Auto white balance
        ENV_FLUORESCENT,            //Fluorescent light
        ENV_INCANDESCENT,           //Incandescent light
        ENV_OUTDOOR                 //Outdoor light
    };

    public enum CamResolution
    {
        RES_640 =1,                 //VGA  640x480 Resolution, only supported in snapshot
        RES_320,                    //QVGA  320x240 Resolution, supported in snapshot and
preview
        RES_160                     //QQVGA 160x120  Resolution, supported in
snapshot and preview
    };

}
```

```
using System;
using System.Drawing;

namespace GoalKeeper
{
        public abstract class Obj
        {
                public abstract Color GetColor();
                public bool found = false;
                public Point position = new Point(0,0);
                public abstract bool IsObj(Color color);
        }

        public class Ball : Obj
        {
                public override Color GetColor()
                {
                        return (Color.OrangeRed);
                }
                public override bool IsObj(Color color)
                {
                        int r,g,b;
                        bool result = false;
                        r = color.R;
                        g = color.G;
                        b = color.B;
                        if (r>110)
                                if (r<250)
                                        if (g>40)
                                                if (g<100)
                                                        //if (b>0)
                                                        if (b<100)
                                                                if (r-g>20)
//                                                                      if
(r-b>20)
result = true;
                        return (result);
                }
        }

        public class GK : Obj
        {
                public override Color GetColor()
                {
                        return (Color.Blue);
                }
                public override bool IsObj(Color color)
                {
                        int r,g,b;
                        bool result = false;
                        r = color.R;
                        g = color.G;
                        b = color.B;
                        if (r>40)
                                if (r<140)
                                        if (g>40)
                                                if (g<140)
                                                        if (b>120)
                                                                if (b<200)
                                                                        if (b>r)
if (b>g)
true;                                                                   result =
                        return (result);
                }
        }
```

```
public class GKDir : Obj
{
        public override Color GetColor()
        {
                return (Color.White);
        }
        public override bool IsObj(Color color)
        {
                int r,g,b;
                bool result = false;
                r = color.R;
                g = color.G;
                b = color.B;
                if (r>200)
//                      if (r<120)
//                              if (g>200)
//                                      if (g<100)
//                                              if (b>200)
                                                      if (b<200)
true;                                                         result =
                return (result);
        }

}
```

```
sing System;
using System.Drawing;
namespace GoalKeeper
{
        /// <summary>
        /// Summary description for GKRobot.
        /// </summary>
        public class GKRobot
        {
                public bool hold = false;
                private byte old_speed = 0;
                private bool old_dir = false;
                private int division = 0;
                public Point targetout = new Point(0,0);
                public string Say = "";
                public Point area_start,area_stop;
                private Point center = new Point(0,0);
                private bool firstfound = false;
                private Point goal_start,goal_stop;
                private Point old_ballpos = new Point(0,0);
                public void Init(Point area_startp, Point area_stopp, Point
goal_startp, Point goal_stopp)
                {
                        area_start = area_startp;
                        area_stop = area_stopp;
                        goal_start = goal_startp;
                        goal_stop = goal_stopp;
                        center.X = (goal_stop.X+goal_start.X)/2;
                        center.Y = (area_stop.Y+area_stop.Y)/2;
                        division = (goal_stop.X-goal_start.X)/8;
                }

                /// <summary>
                /// Calculate distance between old position and new position
approximately.
                /// </summary>
                /// <param name="oldpos"></param>
                /// <param name="newpos"></param>
                /// <returns></returns>
                private int Distance(Point oldpos,Point newpos)
                {
                        int disX,disY;
                        disX = Math.Abs(newpos.X-oldpos.X);
                        disY = Math.Abs(newpos.Y-oldpos.Y);
                        if (disX > disY) return (disX);
                        else return (disY);
                }

                private bool IsInArea(Point objpoint, Point startp, Point stopp)
                {
                        bool result = false;
                        if ((objpoint.Y > startp.Y)&&(objpoint.Y < stopp.Y))
                                if ((objpoint.X > startp.X)&&(objpoint.X <
stopp.X))
                                        result = true;
                        return (result);
                }

                public void Think(bool ballfound,Point gkpos, Point gkdir, Point
ballpos)
                {
                        bool goto_center = true;
                        if (ballfound)
                        {
                                if (!firstfound)
                                {
                                        firstfound = true;
                                        old_ballpos = ballpos;
```

```
                        GK1
                    }
                    else
                    {
                        if (Distance(old_ballpos,ballpos) > 3)
//Ball is moving
                        {
//Ball is moving to the goal.
                            if (ballpos.Y > old_ballpos.Y)
                            {
                                goto_center = false;
                                int x;
                                float m = 100F;
old_ballpos.X) < 3) //Ball move straightly.
                                if (Math.Abs(ballpos.X -
                                {
                                    x = ballpos.X;
                                }
                                else
                                {
(float)(ballpos.Y-old_ballpos.Y)/(ballpos.X-old_ballpos.X); //m = (y1-y0)/(x1-x0)
                                    m =
0.7)
                                    if (Math.Abs(m) >
                                    {
ballpos.Y-(m*ballpos.X); //c = y1-m*x1
                                        float c =
Convert.ToInt32((gkpos.Y-c)/m); //x = (y-c)/m
                                        x =
goal_stop.X) x = goal_stop.X;
                                        if (x >
(x < goal_start.X) x = goal_start.X;
                                        else if
                                    }
                                    else
                                    {
0) x = goal_stop.X;
                                        if (m >
goal_start.X;
                                        else x =
                                    }
                                }
Point(x,gkpos.Y);
                                Point target = new
MoveTo(target,gkpos,gkdir);
                            }
                            old_ballpos = ballpos;
                        }
                        else //Ball halt
                        {
                            if
(IsInArea(ballpos,area_start,area_stop))
                            {
                                goto_center = false;
Point(ballpos.X,gkpos.Y);
                                Point target = new
goal_start.X) target.X = goal_start.X;
                                if (target.X <
goal_stop.X) target.X = goal_stop.X;
                                else if (target.X >
MoveTo(target,gkpos,gkdir);
                            }
                            old_ballpos = ballpos;
                        }
                    }
                }
                else
```

```
                {
                        ballfound = false;
                }
                if (goto_center)
                {
                        MoveTo(center,gkpos,gkdir);
                }
        }

        public void Stop()
        {
                SerialClass.serial.SendByte(0x00);
        }

        private void MoveTo(Point target, Point nowpos, Point dir)
        {
                const int adj_dis = 0;
                targetout = target;
                Say = "Stop!!!";
                if (target.X >= nowpos.X)//Right
                {
                        byte speed =
Convert.ToByte((target.X-nowpos.X+adj_dis)/division);
                        Move(true,speed);
                        if (speed > 0) Say = "Move Right";
                }
                else if (nowpos.X > target.X)//Left
                {
                        byte speed =
Convert.ToByte((nowpos.X-target.X+adj_dis)/division);
                        Move(false,speed);
                        if (speed > 0) Say = "Move Left";
                }
        }

        private void Move(bool left,byte speed)
        {
                byte data = 0x00;
                int i;
                if (speed == 0x01) speed = 0x02;        //Ignore speed1
                if (old_dir != left) speed = 0;
//              else speed = Convert.ToByte((old_speed+speed+1)/2);
                for (i = 0;i < 17; i++)
                        data += speed;
                if (!left) data |= 0x88;
                if (hold) SerialClass.serial.SendByte(0x00);
                else SerialClass.serial.SendByte(data);
                old_dir = left;
                old_speed = speed;
        }
    }
}
```

GKSerial

```
using System;
using JH.CommBase;

namespace GoalKeeper
{
        public class SerialClass : CommBase
        {
                public static SerialClass serial = new SerialClass();
                public static CommBaseSettings settings = new CommBaseSettings();

                protected override CommBaseSettings CommSettings()
                {
                        return settings;
                }
/*              protected override void OnRxChar(byte data)
                {
                        form.RxDo(data);
                }
                public void SendChar(char data)
                {
                        Send((byte) data);
                }

                public void SendByte(byte data)
                {
                        Send(data);
                }

                public void SendMes(byte[] data)
                {
                        Send(data);
                }
        }
}
```

GKForm1

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using DirectX.Capture;
using JH.CommBase;

namespace GoalKeeper
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		private GKRobot gkrobot = new GKRobot();
		private Ball ball = new Ball();
		private GK gk = new GK();
		private GKDir gkdir = new GKDir();
//		private Point ball = new Point(0,0);
		private Bitmap bmp;
//		private bool ball_found = false;
		private Point field_start,field_stop;
		private Point goal_start,goal_stop;
		private Point area_start,area_stop;
		private bool opened = false;
		private System.Windows.Forms.Timer timer = new Timer();
		private int frame_count = 0;
		private System.Windows.Forms.Panel videoPanel;
		private Capture capture = null;
		private Filters filters = new Filters();
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.MainMenu mainMenu1;
		private System.Windows.Forms.Label label2;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.Label fpsLabel;
		private System.Windows.Forms.PictureBox dispPicbox;
		private System.Windows.Forms.MenuItem camMenu;
		private System.Windows.Forms.MenuItem prevMenu;
		private System.Windows.Forms.MenuItem comMenu;
		private System.Windows.Forms.MenuItem selportMenu;
		private System.Windows.Forms.MenuItem com1Menu;
		private System.Windows.Forms.MenuItem com2Menu;
		private System.Windows.Forms.MenuItem openportMenu;
		private System.Windows.Forms.MenuItem exitMenu;
		private System.Windows.Forms.MenuItem seldevMenu;
		private System.Windows.Forms.MenuItem startMenu;
		private System.Windows.Forms.MenuItem setareaMenu;
		private System.Windows.Forms.Label fieldLabel2;
		private System.Windows.Forms.Label goalLabel2;
		private System.Windows.Forms.Label areaLabel2;
		private System.Windows.Forms.Label fieldLabel1;
		private System.Windows.Forms.Label goalLabel1;
		private System.Windows.Forms.Label areaLabel1;
		private System.Windows.Forms.CheckBox disEn;
		private System.Windows.Forms.Label areaLabel3;
		private System.Windows.Forms.Label goalLabel3;
		private System.Windows.Forms.Label fieldLabel3;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.Label label5;
		private System.Windows.Forms.Label label6;
		private System.Windows.Forms.Label ballLabel1;
		private System.Windows.Forms.Label ballLabel2;
		private System.Windows.Forms.Label gkLabel1;
		private System.Windows.Forms.Label gkLabel2;
		private System.Windows.Forms.Label gksayLabel;
		private System.Windows.Forms.Label targetLabel;
```

```
private System.Windows.Forms.CheckBox holdEn;
private System.Windows.Forms.Label movetoLabel;
private System.Windows.Forms.Label dirLabel;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();
        timer.Interval = 1000;
        this.timer.Tick += new EventHandler(timer_Tick);
        this.KeyPreview = true;
        this.KeyDown += new KeyEventHandler(Form1_KeyDown);
        this.KeyUp += new KeyEventHandler(Form1_KeyUp);
        try { updateMenu(); }
        catch {}
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
        if( disposing )
        {
            if (opened) gkrobot.Stop();
            SerialClass.serial.Close();
            if (components != null)
            {
                components.Dispose();
            }
        }
        base.Dispose( disposing );
}

#region Windows Form Designer generated code
/// <summary>
/// Required method for Designer support - do not modify
/// the contents of this method with the code editor.
/// </summary>
private void InitializeComponent()
{
        this.videoPanel = new System.Windows.Forms.Panel();
        this.label1 = new System.Windows.Forms.Label();
        this.dispPicbox = new System.Windows.Forms.PictureBox();
        this.mainMenu1 = new System.Windows.Forms.MainMenu();
        this.camMenu = new System.Windows.Forms.MenuItem();
        this.seldevMenu = new System.Windows.Forms.MenuItem();
        this.prevMenu = new System.Windows.Forms.MenuItem();
        this.comMenu = new System.Windows.Forms.MenuItem();
        this.selportMenu = new System.Windows.Forms.MenuItem();
        this.com1Menu = new System.Windows.Forms.MenuItem();
        this.com2Menu = new System.Windows.Forms.MenuItem();
        this.openportMenu = new System.Windows.Forms.MenuItem();
        this.setareaMenu = new System.Windows.Forms.MenuItem();
        this.startMenu = new System.Windows.Forms.MenuItem();
        this.exitMenu = new System.Windows.Forms.MenuItem();
        this.label2 = new System.Windows.Forms.Label();
        this.label3 = new System.Windows.Forms.Label();
        this.fpsLabel = new System.Windows.Forms.Label();
        this.fieldLabel2 = new System.Windows.Forms.Label();
        this.goalLabel2 = new System.Windows.Forms.Label();
        this.areaLabel2 = new System.Windows.Forms.Label();
```

```
            this.fieldLabel1 = new System.Windows.Forms.Label();
            this.goalLabel1 = new System.Windows.Forms.Label();
            this.areaLabel1 = new System.Windows.Forms.Label();
            this.disEn = new System.Windows.Forms.CheckBox();
            this.areaLabel3 = new System.Windows.Forms.Label();
            this.goalLabel3 = new System.Windows.Forms.Label();
            this.fieldLabel3 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.ballLabel1 = new System.Windows.Forms.Label();
            this.ballLabel2 = new System.Windows.Forms.Label();
            this.gkLabel1 = new System.Windows.Forms.Label();
            this.gkLabel2 = new System.Windows.Forms.Label();
            this.gksayLabel = new System.Windows.Forms.Label();
            this.targetLabel = new System.Windows.Forms.Label();
            this.holdEn = new System.Windows.Forms.CheckBox();
            this.movetoLabel = new System.Windows.Forms.Label();
            this.dirLabel = new System.Windows.Forms.Label();
            this.SuspendLayout();
            //
            // videoPanel
            //
            this.videoPanel.BackColor =
System.Drawing.SystemColors.Highlight;
            this.videoPanel.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.videoPanel.Location = new System.Drawing.Point(0,
0);
            this.videoPanel.Name = "videoPanel";
            this.videoPanel.Size = new System.Drawing.Size(320, 240);
            this.videoPanel.TabIndex = 0;
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(8, 248);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(152, 16);
            this.label1.TabIndex = 1;
            this.label1.Text = "Image From Camera at 30fps";
            //
            // dispPicbox
            //
            this.dispPicbox.BackColor =
System.Drawing.Color.LimeGreen;
            this.dispPicbox.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D;
            this.dispPicbox.Location = new System.Drawing.Point(328,
0);
            this.dispPicbox.Name = "dispPicbox";
            this.dispPicbox.Size = new System.Drawing.Size(160, 120);
            this.dispPicbox.TabIndex = 2;
            this.dispPicbox.TabStop = false;
            //
            // mainMenu1
            //
            this.mainMenu1.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {

this.camMenu,

this.comMenu,

this.setareaMenu,
```

```
this.startMenu,

this.exitMenu});
                        //
                        // camMenu
                        //
                        this.camMenu.Index = 0;
                        this.camMenu.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {

this.seldevMenu,

this.prevMenu});
                        this.camMenu.Text = "Camera";
                        //
                        // seldevMenu
                        //
                        this.seldevMenu.Index = 0;
                        this.seldevMenu.Text = "Select Device";
                        this.seldevMenu.Click += new
System.EventHandler(this.seldevMenu_Click);
                        //
                        // prevMenu
                        //
                        this.prevMenu.Enabled = false;
                        this.prevMenu.Index = 1;
                        this.prevMenu.Text = "Preview";
                        this.prevMenu.Click += new
System.EventHandler(this.prevMenu_Click);
                        //
                        // comMenu
                        //
                        this.comMenu.Index = 1;
                        this.comMenu.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {

this.selportMenu,

this.openportMenu});
                        this.comMenu.Text = "COM Port";
                        //
                        // selportMenu
                        //
                        this.selportMenu.Index = 0;
                        this.selportMenu.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {

this.com1Menu,

this.com2Menu});
                        this.selportMenu.Text = "Select Port";
                        //
                        // com1Menu
                        //
                        this.com1Menu.Checked = true;
                        this.com1Menu.Index = 0;
                        this.com1Menu.Text = "COM1";
                        this.com1Menu.Click += new
System.EventHandler(this.com1Menu_Click);
                        //
```

```
                        // com2Menu
                        //
                        this.com2Menu.Index = 1;
                        this.com2Menu.Text = "COM2";
                        this.com2Menu.Click += new
System.EventHandler(this.com2Menu_Click);
                        //
                        // openportMenu
                        //
                        this.openportMenu.Index = 1;
                        this.openportMenu.Text = "Open Port";
                        this.openportMenu.Click += new
System.EventHandler(this.openportMenu_Click);
                        //
                        // setareaMenu
                        //
                        this.setareaMenu.Index = 2;
                        this.setareaMenu.Text = "Set Area";
                        this.setareaMenu.Click += new
System.EventHandler(this.setareaMenu_Click);
                        //
                        // startMenu
                        //
                        this.startMenu.Index = 3;
                        this.startMenu.Text = "Start";
                        this.startMenu.Click += new
System.EventHandler(this.startMenu_Click);
                        //
                        // exitMenu
                        //
                        this.exitMenu.Index = 4;
                        this.exitMenu.Text = "Exit";
                        this.exitMenu.Click += new
System.EventHandler(this.exitMenu_Click);
                        //
                        // label2
                        //
                        this.label2.Location = new System.Drawing.Point(328,
144);
                        this.label2.Name = "label2";
                        this.label2.Size = new System.Drawing.Size(100, 16);
                        this.label2.TabIndex = 3;
                        this.label2.Text = "Processed Image";
                        //
                        // label3
                        //
                        this.label3.Location = new System.Drawing.Point(328,
160);
                        this.label3.Name = "label3";
                        this.label3.Size = new System.Drawing.Size(72, 16);
                        this.label3.TabIndex = 4;
                        this.label3.Text = "Frame Rate :";
                        //
                        // fpsLabel
                        //
                        this.fpsLabel.Location = new System.Drawing.Point(392,
160);
                        this.fpsLabel.Name = "fpsLabel";
                        this.fpsLabel.Size = new System.Drawing.Size(24, 16);
                        this.fpsLabel.TabIndex = 5;
                        this.fpsLabel.Text = "0";
                        this.fpsLabel.TextAlign =
System.Drawing.ContentAlignment.TopRight;
                        //
                        // fieldLabel2
                        //
                        this.fieldLabel2.Location = new System.Drawing.Point(64,
272);
```

```
            this.fieldLabel2.Name = "fieldLabel2";
            this.fieldLabel2.Size = new System.Drawing.Size(88, 16);
            this.fieldLabel2.TabIndex = 7;
            //
            // goalLabel2
            //
            this.goalLabel2.Location = new System.Drawing.Point(64,
288);
            this.goalLabel2.Name = "goalLabel2";
            this.goalLabel2.Size = new System.Drawing.Size(88, 16);
            this.goalLabel2.TabIndex = 8;
            //
            // areaLabel2
            //
            this.areaLabel2.Location = new System.Drawing.Point(64,
304);
            this.areaLabel2.Name = "areaLabel2";
            this.areaLabel2.Size = new System.Drawing.Size(88, 16);
            this.areaLabel2.TabIndex = 9;
            //
            // fieldLabel1
            //
            this.fieldLabel1.Location = new System.Drawing.Point(8,
272);
            this.fieldLabel1.Name = "fieldLabel1";
            this.fieldLabel1.Size = new System.Drawing.Size(48, 16);
            this.fieldLabel1.TabIndex = 10;
            this.fieldLabel1.Text = "&Field:";
            //
            // goalLabel1
            //
            this.goalLabel1.Location = new System.Drawing.Point(8,
288);
            this.goalLabel1.Name = "goalLabel1";
            this.goalLabel1.Size = new System.Drawing.Size(48, 16);
            this.goalLabel1.TabIndex = 11;
            this.goalLabel1.Text = "&Goal:";
            //
            // areaLabel1
            //
            this.areaLabel1.Location = new System.Drawing.Point(8,
304);
            this.areaLabel1.Name = "areaLabel1";
            this.areaLabel1.Size = new System.Drawing.Size(56, 16);
            this.areaLabel1.TabIndex = 12;
            this.areaLabel1.Text = "GK &Area:";
            //
            // disEn
            //
            this.disEn.Checked = true;
            this.disEn.CheckState =
System.Windows.Forms.CheckState.Checked;
            this.disEn.Location = new System.Drawing.Point(328, 128);
            this.disEn.Name = "disEn";
            this.disEn.Size = new System.Drawing.Size(104, 16);
            this.disEn.TabIndex = 13;
            this.disEn.Text = "Display Enable";
            //
            // areaLabel3
            //
            this.areaLabel3.Location = new System.Drawing.Point(176,
304);
            this.areaLabel3.Name = "areaLabel3";
            this.areaLabel3.Size = new System.Drawing.Size(88, 16);
            this.areaLabel3.TabIndex = 16;
            //
            // goalLabel3
            //
```

```
288);
                this.goalLabel3.Location = new System.Drawing.Point(176,
                this.goalLabel3.Name = "goalLabel3";
                this.goalLabel3.Size = new System.Drawing.Size(88, 16);
                this.goalLabel3.TabIndex = 15;
                //
                // fieldLabel3
                //
272);
                this.fieldLabel3.Location = new System.Drawing.Point(176,
                this.fieldLabel3.Name = "fieldLabel3";
                this.fieldLabel3.Size = new System.Drawing.Size(88, 16);
                this.fieldLabel3.TabIndex = 14;
                //
                // label4
                //
272);
                this.label4.Location = new System.Drawing.Point(160,
                this.label4.Name = "label4";
                this.label4.Size = new System.Drawing.Size(16, 16);
                this.label4.TabIndex = 17;
                this.label4.Text = "->";
                //
                // label5
                //
288);
                this.label5.Location = new System.Drawing.Point(160,
                this.label5.Name = "label5";
                this.label5.Size = new System.Drawing.Size(16, 16);
                this.label5.TabIndex = 18;
                this.label5.Text = "->";
                //
                // label6
                //
304);
                this.label6.Location = new System.Drawing.Point(160,
                this.label6.Name = "label6";
                this.label6.Size = new System.Drawing.Size(16, 16);
                this.label6.TabIndex = 19;
                this.label6.Text = "->";
                //
                // ballLabel1
                //
176);
                this.ballLabel1.Location = new System.Drawing.Point(328,
                this.ballLabel1.Name = "ballLabel1";
                this.ballLabel1.Size = new System.Drawing.Size(32, 16);
                this.ballLabel1.TabIndex = 20;
                this.ballLabel1.Text = "Ball :";
                //
                // ballLabel2
                //
176);
                this.ballLabel2.Location = new System.Drawing.Point(384,
                this.ballLabel2.Name = "ballLabel2";
                this.ballLabel2.Size = new System.Drawing.Size(128, 16);
                this.ballLabel2.TabIndex = 21;
                //
                // gkLabel1
                //
192);
                this.gkLabel1.Location = new System.Drawing.Point(328,
                this.gkLabel1.Name = "gkLabel1";
                this.gkLabel1.Size = new System.Drawing.Size(24, 16);
                this.gkLabel1.TabIndex = 22;
                this.gkLabel1.Text = "GK:";
                //
                // gkLabel2
```

```
            //
            this.gkLabel2.Location = new System.Drawing.Point(384,
192);
            this.gkLabel2.Name = "gkLabel2";
            this.gkLabel2.Size = new System.Drawing.Size(128, 16);
            this.gkLabel2.TabIndex = 23;
            //
            // gksayLabel
            //
            this.gksayLabel.ForeColor =
System.Drawing.Color.OrangeRed;
            this.gksayLabel.Location = new System.Drawing.Point(384,
224);
            this.gksayLabel.Name = "gksayLabel";
            this.gksayLabel.Size = new System.Drawing.Size(128, 16);
            this.gksayLabel.TabIndex = 24;
            this.gksayLabel.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter;
            //
            // targetLabel
            //
            this.targetLabel.Location = new System.Drawing.Point(384,
208);
            this.targetLabel.Name = "targetLabel";
            this.targetLabel.Size = new System.Drawing.Size(128, 16);
            this.targetLabel.TabIndex = 25;
            //
            // holdEn
            //
            this.holdEn.Location = new System.Drawing.Point(328,
248);
            this.holdEn.Name = "holdEn";
            this.holdEn.TabIndex = 26;
            this.holdEn.Text = "Hold";
            this.holdEn.CheckedChanged += new
System.EventHandler(this.holdEn_CheckedChanged);
            //
            // movetoLabel
            //
            this.movetoLabel.Location = new System.Drawing.Point(328,
208);
            this.movetoLabel.Name = "movetoLabel";
            this.movetoLabel.Size = new System.Drawing.Size(56, 16);
            this.movetoLabel.TabIndex = 27;
            this.movetoLabel.Text = "Move To:";
            //
            // dirLabel
            //
            this.dirLabel.Location = new System.Drawing.Point(328,
224);
            this.dirLabel.Name = "dirLabel";
            this.dirLabel.Size = new System.Drawing.Size(32, 16);
            this.dirLabel.TabIndex = 28;
            this.dirLabel.Text = "Dir:";
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(560, 342);
            this.Controls.Add(this.dirLabel);
            this.Controls.Add(this.movetoLabel);
            this.Controls.Add(this.holdEn);
            this.Controls.Add(this.targetLabel);
            this.Controls.Add(this.gksayLabel);
            this.Controls.Add(this.gkLabel2);
            this.Controls.Add(this.gkLabel1);
            this.Controls.Add(this.ballLabel2);
            this.Controls.Add(this.ballLabel1);
```

```
            this.Controls.Add(this.label6);
            this.Controls.Add(this.label5);
            this.Controls.Add(this.label4);
            this.Controls.Add(this.areaLabel3);
            this.Controls.Add(this.goalLabel3);
            this.Controls.Add(this.fieldLabel3);
            this.Controls.Add(this.disEn);
            this.Controls.Add(this.areaLabel1);
            this.Controls.Add(this.goalLabel1);
            this.Controls.Add(this.fieldLabel1);
            this.Controls.Add(this.areaLabel2);
            this.Controls.Add(this.goalLabel2);
            this.Controls.Add(this.fieldLabel2);
            this.Controls.Add(this.fpsLabel);
            this.Controls.Add(this.label3);
            this.Controls.Add(this.label2);
            this.Controls.Add(this.dispPicbox);
            this.Controls.Add(this.label1);
            this.Controls.Add(this.videoPanel);
            this.Menu = this.mainMenu1;
            this.Name = "Form1";
            this.Text = "Goal Keeper Soccer Robot Project";
            this.ResumeLayout(false);

        }
        #endregion

        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.Run(new Form1());
        }

        private void seldevMenu_Click(object sender, System.EventArgs e)
        {
            try
            {
                // Get current devices and dispose of capture
object
                // because the video and audio device can only be
changed
                // by creating a new Capture object.
                Filter videoDevice = null;
                Filter audioDevice = null;
                if ( capture != null )
                {
                    videoDevice = capture.VideoDevice;
                    capture.Dispose();
                    capture = null;
                }

                // Get new video device
                MenuItem m = sender as MenuItem;
                videoDevice = ( m.Index>0 ?
filters.VideoInputDevices[m.Index-1] : null );

                // Create capture object
                if ( videoDevice != null )
                {
                    capture = new Capture( videoDevice,
audioDevice );
                }

                // Update the menu
                updateMenu();
```

```
                }
                catch (Exception ex)
                {
                        MessageBox.Show( "Video device not
supported.\n\n" + ex.Message + "\n\n" + ex.ToString() );
                }
        }
        private void updateMenu()
        {
                MenuItem m;
                Filter f;
                Control oldPreviewWindow = null;

                // Disable preview to avoid additional flashes (optional)
                if ( capture != null )
                {
                        oldPreviewWindow = capture.PreviewWindow;
                        capture.PreviewWindow = null;
                }

                // Load video devices
                Filter videoDevice = null;
                if ( capture != null )
                        videoDevice = capture.VideoDevice;
                seldevMenu.MenuItems.Clear();
                m = new MenuItem( "(None)", new
EventHandler(seldevMenu_Click) );
                m.Checked = ( videoDevice == null );
                seldevMenu.MenuItems.Add( m );
                for ( int c = 0; c < filters.VideoInputDevices.Count; c++
)
                {
                        f = filters.VideoInputDevices[c];
                        m = new MenuItem( f.Name, new
EventHandler(seldevMenu_Click) );
                        m.Checked = ( videoDevice == f );
                        seldevMenu.MenuItems.Add( m );
                }
                seldevMenu.Enabled = ( filters.VideoInputDevices.Count >
0 );

                // Check Preview menu option
                prevMenu.Checked = ( oldPreviewWindow != null );
                prevMenu.Enabled = ( capture != null );

                // Reenable preview if it was enabled before
                if ( capture != null )
                        capture.PreviewWindow = oldPreviewWindow;
        }

        private void prevMenu_Click(object sender, System.EventArgs e)
        {
                try
                {
                        if ( capture.PreviewWindow == null )
                        {
                                capture.PreviewWindow = videoPanel;
                                prevMenu.Checked = true;
                        }
                        else
                        {
                                capture.PreviewWindow = null;
                                prevMenu.Checked = false;
                        }
                }
                catch (Exception ex)
                {
                        MessageBox.Show( "Unable to enable/disable
```

```
preview. Please submit a bug report.\n\n" + ex.Message + "\n\n" + ex.ToString()
);
                }
        }

        private bool Search(Obj obj,Bitmap e,Point pstart,Point pstop)
        {
                int i,j;
                bool result = false;
                if (obj.found)
                {
                        if ((obj.position.X-50) > pstart.X) pstart.X =
obj.position.X-50;
                        if ((obj.position.Y-50) > pstart.Y) pstart.Y =
obj.position.Y-50;
                        if ((obj.position.X+50) < pstop.X) pstop.X =
obj.position.X+50;
                        if ((obj.position.Y+50) < pstop.Y) pstop.Y =
obj.position.Y+50;
                }
                int xcount = 0,ycount = 0,number = 0;
                for (i = pstart.X ; i < pstop.X; i += 2)
                        for (j = pstart.Y; j < pstop.Y; j += 2)
                        {
                                Color color = e.GetPixel(i,j);
                                if (obj.IsObj(color))
                                {
                                        if (disEn.Checked)
bmp.SetPixel(i/2,j/2,obj.GetColor());
                                        number++;
                                        xcount += i;
                                        ycount += j;
                                }
                        }
                if (number > 5)
                {
                        obj.position.X = xcount/number;
                        obj.position.Y = ycount/number;
                        result = true;
                }
                return (result);
        }

        private void CaptureDone(System.Drawing.Bitmap e)
        {
                frame_count++;
                bmp = new Bitmap(160,120);
                ball.found = Search(ball,e,area_start,area_stop);
                if (!ball.found) ball.found =
Search(ball,e,field_start,field_stop);
                gk.found = Search(gk,e,area_start,area_stop);
                if (gk.found)
                {
                        Point gkbox_start,gkbox_stop;
                        gkbox_start = new
Point(gk.position.X-20,gk.position.Y-20);
                        gkbox_stop = new
Point(gk.position.X+20,gk.position.Y+20);
                        gkdir.found =
Search(gkdir,e,gkbox_start,gkbox_stop);

gkrobot.Think(ball.found,gk.position,gkdir.position,ball.position);
                        gksayLabel.Text = gkrobot.Say;
                        targetLabel.Text = gkrobot.targetout.ToString();
                        gkLabel2.Text = gk.position.ToString();
                }
                else
```

```
                {
                        gkrobot.Stop();
                        gkLabel2.Text = "Not found";
                }
                if (ball.found) ballLabel2.Text =
ball.position.ToString();
                else ballLabel2.Text = "Not found";
                if (disEn.Checked) dispPicbox.Image = bmp;
        }

        private void timer_Tick(object sender, EventArgs e)
        {
                fpsLabel.Text = frame_count.ToString();
                frame_count = 0;
        }

        private void exitMenu_Click(object sender, System.EventArgs e)
        {
                SerialClass.serial.Close();
                if ( capture != null ) capture.Stop();
                Application.Exit();
        }

        private void com1Menu_Click(object sender, System.EventArgs e)
        {
                com1Menu.Checked = true;
                com2Menu.Checked = false;
        }

        private void com2Menu_Click(object sender, System.EventArgs e)
        {
                com2Menu.Checked = true;
                com1Menu.Checked = false;
        }

        private void openportMenu_Click(object sender, System.EventArgs
e)
        {
                if (!opened)
                {
                        if (com1Menu.Checked)
SerialClass.settings.SetStandard("COM1:",9600,CommBase.Handshake.none);
                        else
SerialClass.settings.SetStandard("COM2:",9600,CommBase.Handshake.none);
                        if (SerialClass.serial.Open())
                        {
                                opened = true;
                                openportMenu.Text = "Close Port";
                                selportMenu.Enabled = false;
                        }
                        else MessageBox.Show("Can not Open Port");
                }
                else
                {
                        SerialClass.serial.Close();
                        opened = false;
                        openportMenu.Text = "Open Port";
                        selportMenu.Enabled = true;
                }
        }

        private void startMenu_Click(object sender, System.EventArgs e)
        {
                if (!prevMenu.Checked)
                        MessageBox.Show("Please Start Preview");
                else if (!opened)
```

```
                        MessageBox.Show("Please Open COM Port");
                    else if ((fieldLabel2.Text == "")||(goalLabel2.Text ==
"")||(areaLabel2.Text == "")
                                ||(fieldLabel3.Text ==
"")||(goalLabel3.Text == "")||(areaLabel3.Text == ""))
                        MessageBox.Show("Please Set Area");
                    else
                    {
//                      capMenu.Checked = true;
                        startMenu.Enabled = false;
                        timer.Enabled = true;
                        capture.FrameEvent2 += new
Capture.HeFrame(CaptureDone);
                        capture.GrapImg();
//
gkrobot.Init(area_start,area_stop,goal_start,goal_stop);
                    }
                }

                private void Form1_KeyDown(object sender, KeyEventArgs e)
                {
                    if (((e.KeyCode == Keys.F)||(e.KeyCode ==
Keys.G)||(e.KeyCode == Keys.A))&&((e.Control)||(e.Alt)))
                    {
                        Point point = new Point(0,0);
                        point.X =
Form1.MousePosition.X-this.Location.X-videoPanel.Location.X-6;
                        point.Y =
Form1.MousePosition.Y-this.Location.Y-videoPanel.Location.Y-52;
                        if (((point.X >= 0)&&(point.X < 320))&&((point.Y
>= 0)&&(point.Y < 240)))
                        {
                            if (e.Control)
                            {
                                switch (e.KeyCode)
                                {
                                    case Keys.F : field_start
= point; fieldLabel2.Text = point.ToString(); break;
                                    case Keys.G : goal_start
= point; goalLabel2.Text = point.ToString(); break;
                                    case Keys.A : area_start
= point; areaLabel2.Text = point.ToString(); break;
                                }
                            }
                            else
                            {
                                switch (e.KeyCode)
                                {
                                    case Keys.F : field_stop
= point; fieldLabel3.Text = point.ToString(); break;
                                    case Keys.G : goal_stop =
point; goalLabel3.Text = point.ToString(); break;
                                    case Keys.A : area_stop =
point; areaLabel3.Text = point.ToString(); break;
                                }
                            }
                        }

gkrobot.Init(area_start,area_stop,goal_start,goal_stop);
                    }
                    else if (e.KeyCode == Keys.H)
                    {
                        holdEn.Checked = true;
                        gkrobot.hold = true;
                    }
                }

                private void setareaMenu_Click(object sender, System.EventArgs e)
```

```
        {
            MessageBox.Show("Move mouse on the start and stop point
and then press Ctrl/ALT + F/G/A.\n\n"
Ctrl/Alt + F   for   Field\n"                                    +"
Ctrl/Alt + G   for   Goal\n"                                     +"
Ctrl/Alt + A   for   GK Area","To set area");                    +"
        }

        private void holdEn_CheckedChanged(object sender,
System.EventArgs e)
        {
            gkrobot.hold = holdEn.Checked;
        }

        private void Form1_KeyUp(object sender, KeyEventArgs e)
        {
            if (e.KeyCode == Keys.H)
            {
                holdEn.Checked = false;
                gkrobot.hold = false;
            }
        }
    }
}
```

ภาคผนวก ค Datasheet

19-0175; Rev 3; 5/95

MAXIM

±15kV ESD-Protected, +5V RS-232 Transceivers

General Description

The MAX202E–MAX213E, MAX232E/MAX241E line drivers/receivers are designed for RS-232 and V.28 communications in harsh environments. Each transmitter output and receiver input is protected against ±15kV electrostatic discharge (ESD) shocks, without latchup. The various combinations of features are outlined in the *Selection Guide*. The drivers and receivers for all ten devices meet all EIA/TIA-232E and CCITT V.28 specifications at data rates up to 120kbps, when loaded in accordance with the EIA/TIA-232E specification.

The MAX211E/MAX213E/MAX241E are available in 28-pin SO packages, as well as a 28-pin SSOP that uses 60% less board space. The MAX202E/MAX232E come in 16-pin narrow SO, wide SO, and DIP packages. The MAX203E comes in a 20-pin DIP/SO package, and needs no external charge-pump capacitors. The MAX205E comes in a 24-pin wide DIP package, and also eliminates external charge-pump capacitors. The MAX206E/MAX207E/MAX208E come in 24-pin SO, SSOP, and narrow DIP packages. The MAX232E/MAX241E operate with four 1µF capacitors, while the MAX202E/MAX206E/MAX207E/MAX208E/MAX211E/MAX213E operate with four 0.1µF capacitors, further reducing cost and board space.

Applications

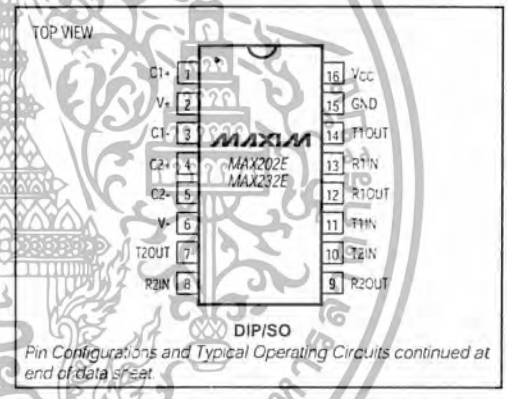
Notebook, Subnotebook, and Palmtop Computers
Battery-Powered Equipment
Hand-Held Equipment

Ordering Information appears at end of data sheet.

Features

- ESD Protection for RS-232 I/O Pins:
  ±15kV—Human Body Model
  ±8kV—IEC1000-4-2, Contact Discharge
  ±15kV—IEC1000-4-2, Air-Gap Discharge
- Latchup Free (unlike bipolar equivalents)
- Guaranteed 120kbps Data Rate—LapLink™ Compatible
- Guaranteed 3V/µs Min Slew Rate
- Operate from a Single +5V Power Supply

Pin Configurations



Pin Configurations and Typical Operating Circuits continued at end of data sheet.

Selection Guide

PART	No. of RS-232 DRIVERS	No. of RS-232 RECEIVERS	RECEIVERS ACTIVE IN SHUTDOWN	No. of EXTERNAL CAPACITORS	LOW-POWER SHUTDOWN	TTL THREE-STATE
MAX202E	2	2	0	4 (0.1µF)	No	No
MAX203E	2	2	0	None	No	No
MAX205E	5	5	0	None	Yes	Yes
MAX206E	4	3	0	4 (0.1µF)	Yes	Yes
MAX207E	5	3	0	4 (0.1µF)	No	No
MAX208E	4	4	0	4 (0.1µF)	No	No
MAX211E	4	5	0	4 (0.1µF)	Yes	Yes
MAX213E	4	5	2	4 (0.1µF)	Yes	Yes
MAX232E	2	2	0	4 (1µF)	No	No
MAX241E	4	5	0	4 (1µF)	Yes	Yes

LapLink is a registered trademark of Traveling Software, Inc.

ABSOLUTE MAXIMUM RATINGS

VCC ..... -0.3V to +6V
V+ ..... (VCC - 0.3V) to +14V
V- ..... -14V to +0.3V
Input Voltages
T_IN ..... -0.3V to (V+ + 0.3V)
R_IN ..... ±30V
Output Voltages
T_OUT ..... (V- - 0.3V) to (V+ + 0.3V)
R_OUT ..... -0.3V to (VCC + 0.3V)
Short-Circuit Duration, T_OUT ..... Continuous
Continuous Power Dissipation (TA = +70°C)
16-Pin Plastic DIP (derate 10.53mW/°C above +70°C) 842mW
16-Pin Narrow SO (derate 8.70mW/°C above +70°C) 696mW
16-Pin Wide SO (derate 9.52mW/°C above +70°C) 762mW
20-Pin Plastic DIP (derate 11.11mW/°C above +70°C) 889mW
20-Pin SO (derate 10.00mW/°C above +70°C) 800mW
24-Pin Narrow Plastic DIP
(derate 13.33mW/°C above +70°C) 1.07W
24-Pin Wide Plastic DIP
(derate 14.29mW/°C above +70°C) 1.14W
24-Pin SO (derate 11.76mW/°C above +70°C) 941mW
24-Pin SSOP (derate 8.00mW/°C above +70°C) 640mW
28-Pin SO (derate 12.50mW/°C above +70°C) 1W
28-Pin SSOP (derate 9.52mW/°C above +70°C) 762mW
Operating Temperature Ranges
MAX2__EC_ _ ..... 0°C to +70°C
MAX2__EE_ _ ..... -40°C to +85°C
Storage Temperature Range ..... -65°C to +165°C
Lead Temperature (soldering, 10sec) ..... +300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(VCC = +5V ±10% for MAX202E/206E/208E/211E/213E/232E/241E, VCC = +5V ±5% for MAX203E/205E/207E; C1–C4 = 0.1µF for MAX202E/206E/207E/208E/211E/213E; C1–C4 = 1µF for MAX232E/241E; TA = TMIN to TMAX; unless otherwise noted. Typical values are at TA = +25°C.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
DC CHARACTERISTICS							
VCC Supply Current	ICC	No load, TA = +25°C	MAX202E/203E		8	15	mA
			MAX205E–208E		11	20	
			MAX211E/213E		14	20	
			MAX232E		5	10	
			MAX241E		7	15	
Shutdown Supply Current		TA = +25°C, Figure 1	MAX205E/206E		1	10	µA
			MAX211E/241E		1	10	
			MAX213E		15	50	
LOGIC							
Input Pull-Up Current		T_IN = 0V (MAX205E–208E/211E/213E/241E)			15	200	µA
Input Leakage Current		T_IN = 0V to VCC (MAX202E/203E/232E)				±10	µA
Input Threshold Low	VIL	T_IN, EN, SHDN (MAX213E) or EN, SHDN (MAX205E–208E/211E/241E)				0.8	V
Input Threshold High	VIH	T_IN		2.0			V
		EN, SHDN (MAX213E) or EN, SHDN (MAX205E–208E/211E/241E)		2.4			
Output Voltage Low	VOL	R_OUT; IOUT = 3.2mA (MAX202E/203E/232E) or IOUT = 1.6mA (MAX205E/208E/211E/213E/241E)				0.4	V
Output Voltage High	VOH	R_OUT; IOUT = -1.0mA		3.5	VCC - 0.4		V
Output Leakage Current		EN = VCC, EN = 0V, 0V ≤ ROUT ≤ VCC, MAX205E–208E/211E/213E/241E outputs disabled			±0.05	±10	µA

ELECTRICAL CHARACTERISTICS (continued)

(V_{CC} = +5V ±10% for MAX202E/206E/208E/211E/213E/232E/241E; V_{CC} = +5V ±5% for MAX203E/205E/207E; C1–C4 = 0.1µF for MAX202E/206E/207E/208E/211E/213E; C1–C4 = 1µF for MAX232E/241E; T_A = T_{MIN} to T_{MAX}; unless otherwise noted. Typical values are at T_A = +25°C.)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
EIA/TIA-232E RECEIVER INPUTS							
Input Voltage Range				-30		30	V
Input Threshold Low		T_A = +25°C, V_{CC} = 5V	All parts, normal operation	0.8	1.2		V
			MAX213E, $\overline{SHDN}$ = 0V, EN = V_{CC}	0.6	1.5		
Input Threshold High		T_A = +25°C, V_{CC} = 5V	All parts, normal operation		1.7	2.4	V
			MAX213E (R4, R5), $\overline{SHDN}$ = 0V, EN = V_{CC}		1.5	2.4	
Input Hysteresis		V_{CC} = 5V, no hysteresis in shutdown		0.2	0.5	1.0	V
Input Resistance		T_A = +25°C, V_{CC} = 5V		3	5	7	kΩ
EIA/TIA-232E TRANSMITTER OUTPUTS							
Output Voltage Swing		All drivers loaded with 3kΩ to ground (Note 1)		±5	±9		V
Output Resistance		V_{CC} = V+ = V- = 0V, V_{OUT} = ±2V		300			Ω
Output Short-Circuit Current					±10	±60	mA
TIMING CHARACTERISTICS							
Maximum Data Rate		R_L = 3kΩ to 7kΩ, C_L = 50pF to 1000pF, one transmitter switching		120			kbps
Receiver Propagation Delay	t_{PLHR}, t_{PHLR}	C_L = 150pF	All parts, normal operation		0.5	10	µs
			MAX213E (R4, R5), $\overline{SHDN}$ = 0V, EN = V_{CC}		4	40	
Receiver Output Enable Time		MAX205E/206E/211E/213E/241E normal operation, Figure 2			600		ns
Receiver Output Disable Time		MAX205E/206E/211E/213E/241E normal operation, Figure 2			200		ns
Transmitter Propagation Delay	t_{PLHT}, t_{PHLT}	R_L = 3kΩ, C_L = 2500pF, all transmitters loaded			2		µs
Transition-Region Slew Rate		T_A = +25°C, V_{CC} = 5V, R_L = 3kΩ to 7kΩ, C_L = 50pF to 1000pF, measured from -3V to +3V or +3V to -3V, Figure 3		3	6	30	V/µs
ESD PERFORMANCE: TRANSMITTER OUTPUTS, RECEIVER INPUTS							
ESD-Protection Voltage		Human Body Model			±15		kV
		IEC1000-4-2, Contact Discharge			±8		
		IEC1000-4-2, Air-Gap Discharge			±15		

Note 1: MAX211EE_ _ tested with V_{CC} = +5V ±5%.

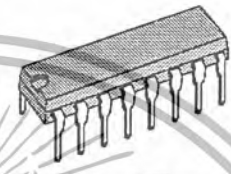
SGS-THOMSON MICROELECTRONICS

L293D

PUSH-PULL FOUR CHANNEL DRIVER WITH DIODES

PRELIMINARY DATA

- 600mA. OUTPUT CURRENT CAPABILITY PER CHANNEL
- 1.2A PEAK OUTPUT CURRENT (NON REPETITIVE) PER CHANNEL
- ENABLE FACILITY
- OVERTEMPERATURE PROTECTION
- LOGICAL "0" INPUT VOLTAGE UP TO 1.5v (HIGH NOISE IMMUNITY
- INTERNAL CLAMPS DIODES



Powerdip
12+2+2

ORDERING NUMBER : L293D

DESCRIPTION

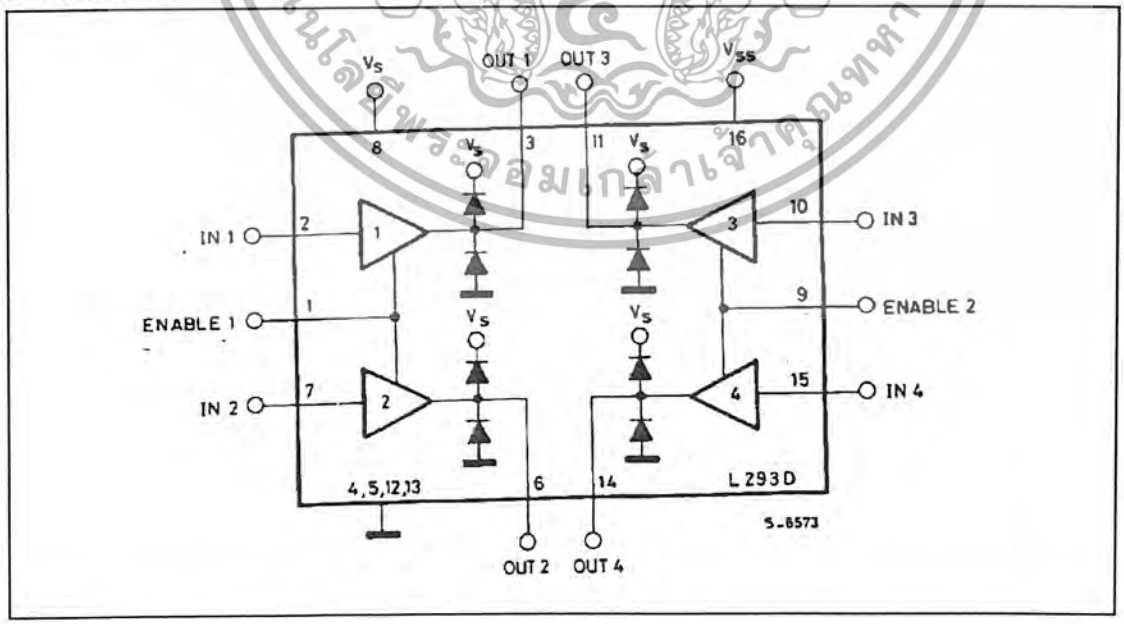
The L293D is a monolithic integrated high voltage, high current four channel vriver designed to accept standard DTL or TTL logic levels and drive inductive loads (such as relays solenoides, DC and stepping motors) and switching power transistors.

To simplify use as two bridges is pair of channels is equiped with an enable input. A separate supply innput is provited formthe logic, allowing operation at a low voltage and internal clamp diodes are included.

This device is suitable for use in switching applications at frequencies up to 5 KHz.

The L293D is assembled in a 16 lead plastic packege which has 4 center pins connected together and used for heatsinking.

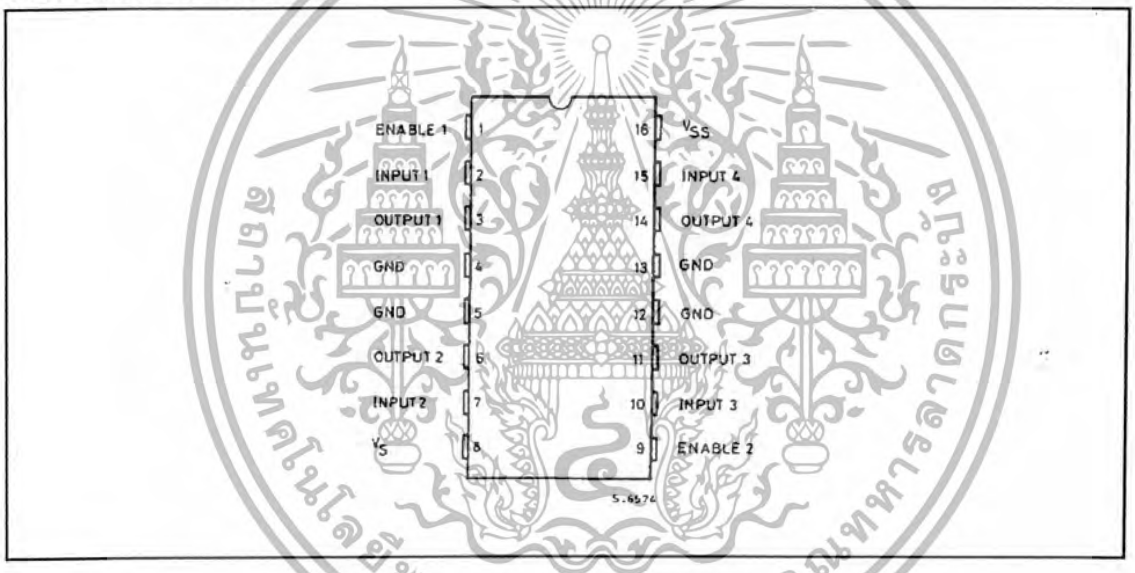
BLOCK DIAGRAM



ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_S	Supply Voltage	36	V
V_{SS}	Logic Supply voltage	36	V
V_i	Input voltage	7	V
V_{en}	Enable voltage	7	V
I_o	Peak output current (100µs non repetitive)	1.2	A
P_{tot}	Total power dissipation at $T_{ground\text{-}pins}$ = 80°C	5	W
T_{stg}, Tj	Storage and junction temperature	-40 to 150	°C

CONNECTION DIAGRAM



THERMAL DATA

Symbol	Parameter		Value	Unit
Rth-j-case	Thermal resistance junction-case	max	14	°C/W
Rth j-case	Thermal resistance junction-ambient	max	80	°C/W

ELECTRICAL CHARACTERISTICS (For each channel, V_S = 24V, V_{SS} = 5V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test conditions	Min.	Typ.	Max.	Unit
V_s	Supply voltage		V_{ss}		36	V
V_{ss}	Logic supply voltage (pin 16)		4.5		36	V
I_s	Total quiescent supply current (pin 8)	V_i = L I_o = 0 V_{en} = H		2	6	mA
		V_i = H I_o = 0 V_{en} = H		16	24	
		V_{en} = L			4	
I_{ss}	Total quiescent logic supply current (pin 16)	V_i = L I_o = 0 V_{en} = H		44	60	mA
		V_i = H I_o = 0 V_{en} = H		16	22	
		V_{en} = L		16	24	
V_{IL}	Input low voltage (pin 2, 7, 10, 15)		-0.3		1.5	V
V_{IH}	Input high voltage (pin 2, 7, 10, 15)	$V_{ss} \leq 7V$	2.3		V_{ss}	V
		$V_{ss} > 7V$	2.3		7	
I_{IL}	Low voltage input current (pin 2, 7, 10, 15)	V_{IL} = 1.5V			-10	µA
I_{IH}	High voltage input current (pin 2, 7, 10, 15)	$2.3 \leq V_{IH} \leq V_{ss}$ -0.6V		30	100	µA
V_{enL}	Enable low voltage (pin 1, 9)		-0.3		1.5	V
V_{enH}	Enable high voltage (pin 1, 9)	$V_{ss} \leq 7V$	2.3		V_{ss}	V
		V_{ss} 7V	2.3		7	
I_{enL}	Low voltage enable current (pin 1, 9)	V_{enL} = 1.5V		-30	-100	µA
i_{enH}	High voltage enable current (pin 1, 9)	$2.3V \leq V_{enH} \leq V_{ss}$ -0.6V			± 10	µA
V_{CEsatH}	Source output saturation voltage (pin 3, 6, 11, 14)	I_o = -0.6A		1.4	1.8	V
V_{CEsatL}	Sink output saturation voltage (pins 3, 6, 11, 14)	I_o +0.6A			1.2	1.8
V_F	Clamp diode forward voltage	I_o = 600 mA		1.3		V
t_r	Rise time (*)	0.1 to 0.9 V_o		250		ns
t_f	Fall time (*)	0.9 to 0.1 V_o		250		ns
t_{on}	Turn-on delay (*)	0.5 V_i to 0.5 V_o		750		ns
t_{off}	Turn-off delay (*)	0.5 V_i to 0.5 V_o		200		ns

(*) See fig.1

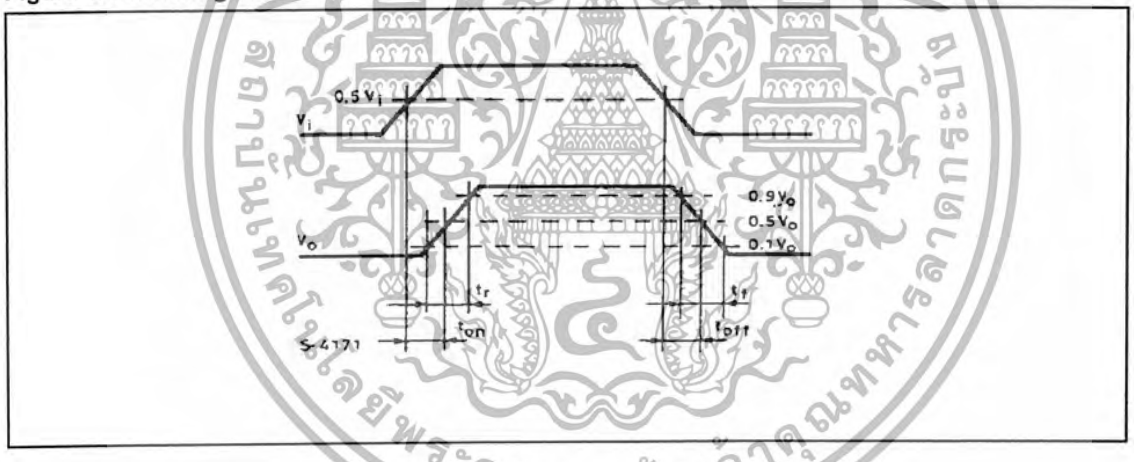
TRUTH TABLE (One channel)

INPUT	ENABLE (*)	OUTPUT
H	H	H
L	H	L
H	L	Z
L	L	Z

Z = High output impedance

(*) Relative to the considered channel

Figure 1. Switching Times



Features

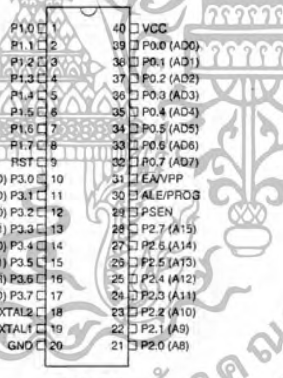
- Compatible with MCS-51™ Products
- 4K Bytes of In-System Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial Channel
- Low-power Idle and Power-down Modes

Description

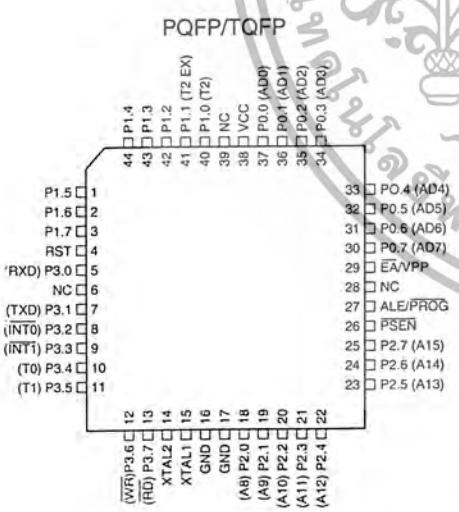
The AT89C51 is a low-power, high-performance CMOS 8-bit microcomputer with 4K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set and pinout. The on-chip Flash allows the program memory to be reprogrammed in-system or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C51 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

Pin Configurations

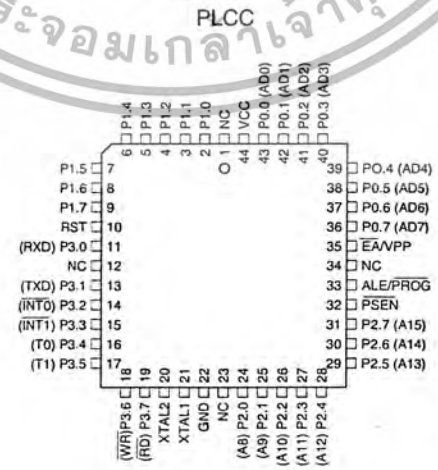
PDIP



PQFP/TQFP



PLCC

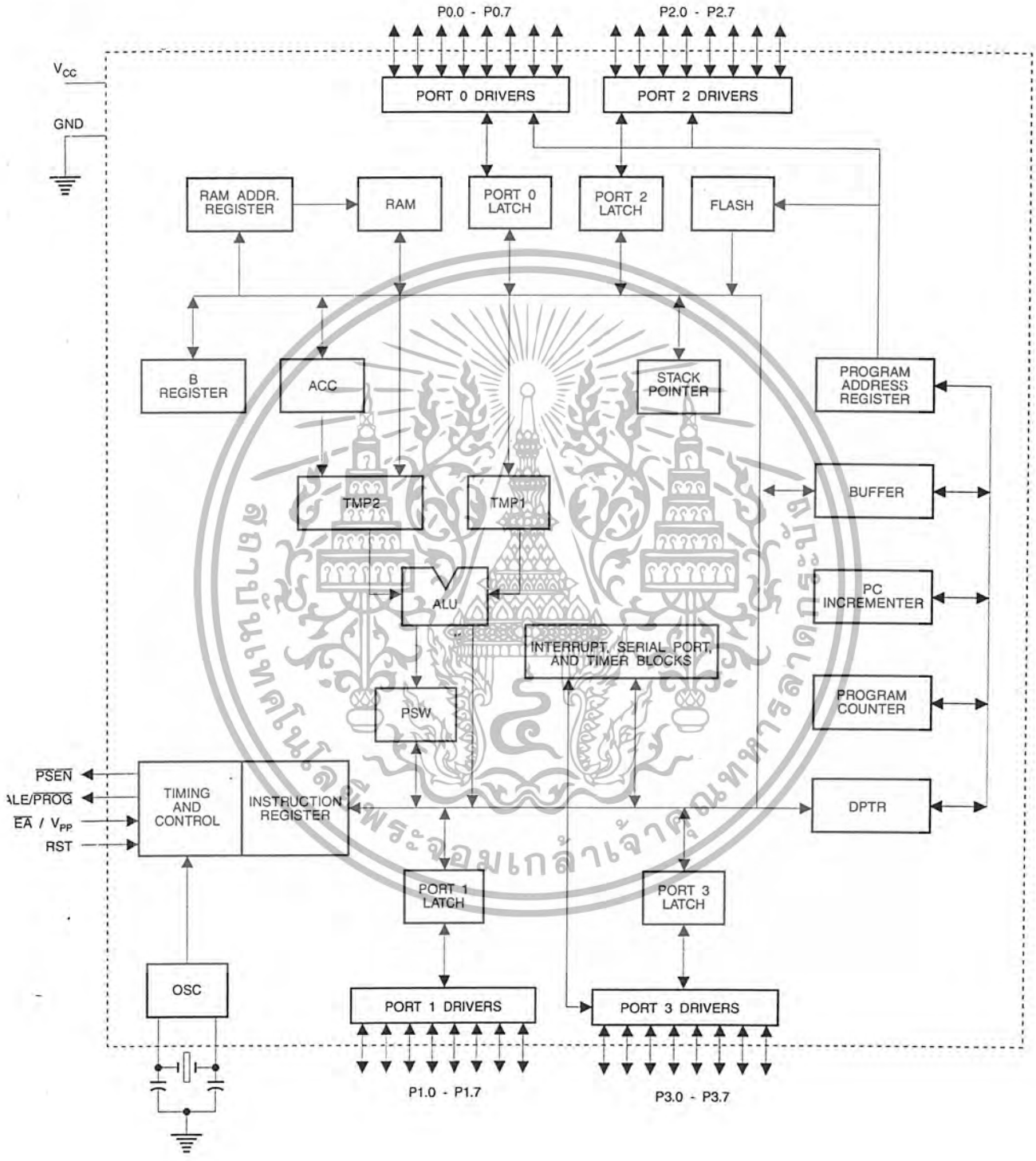


8-bit Microcontroller with 4K Bytes Flash

AT89C51

Rev. 0265G–02/00

Block Diagram



Features

- Compatible with MCS-51™ Products
- 2K Bytes of Reprogrammable Flash Memory
  - Endurance: 1,000 Write/Erase Cycles
- 2.7V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Two-level Program Memory Lock
- 128 x 8-bit Internal RAM
- 15 Programmable I/O Lines
- Two 16-bit Timer/Counters
- Six Interrupt Sources
- Programmable Serial UART Channel
- Direct LED Drive Outputs
- On-chip Analog Comparator
- Low-power Idle and Power-down Modes

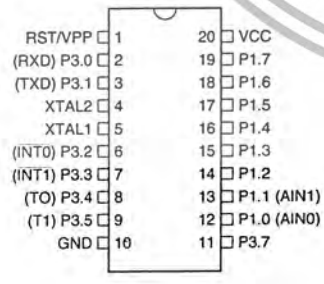
Description

The AT89C2051 is a low-voltage, high-performance CMOS 8-bit microcomputer with 2K bytes of Flash programmable and erasable read only memory (PEROM). The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard MCS-51 instruction set. By combining a versatile 8-bit CPU with Flash on a monolithic chip, the Atmel AT89C2051 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89C2051 provides the following standard features: 2K bytes of Flash, 128 bytes of RAM, 15 I/O lines, two 16-bit timer/counters, a five vector two-level interrupt architecture, a full duplex serial port, a precision analog comparator, on-chip oscillator and clock circuitry. In addition, the AT89C2051 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port and interrupt system to continue functioning. The power-down mode saves the RAM contents but freezes the oscillator disabling all other chip functions until the next hardware reset.

Pin Configuration

PDIP/SOIC

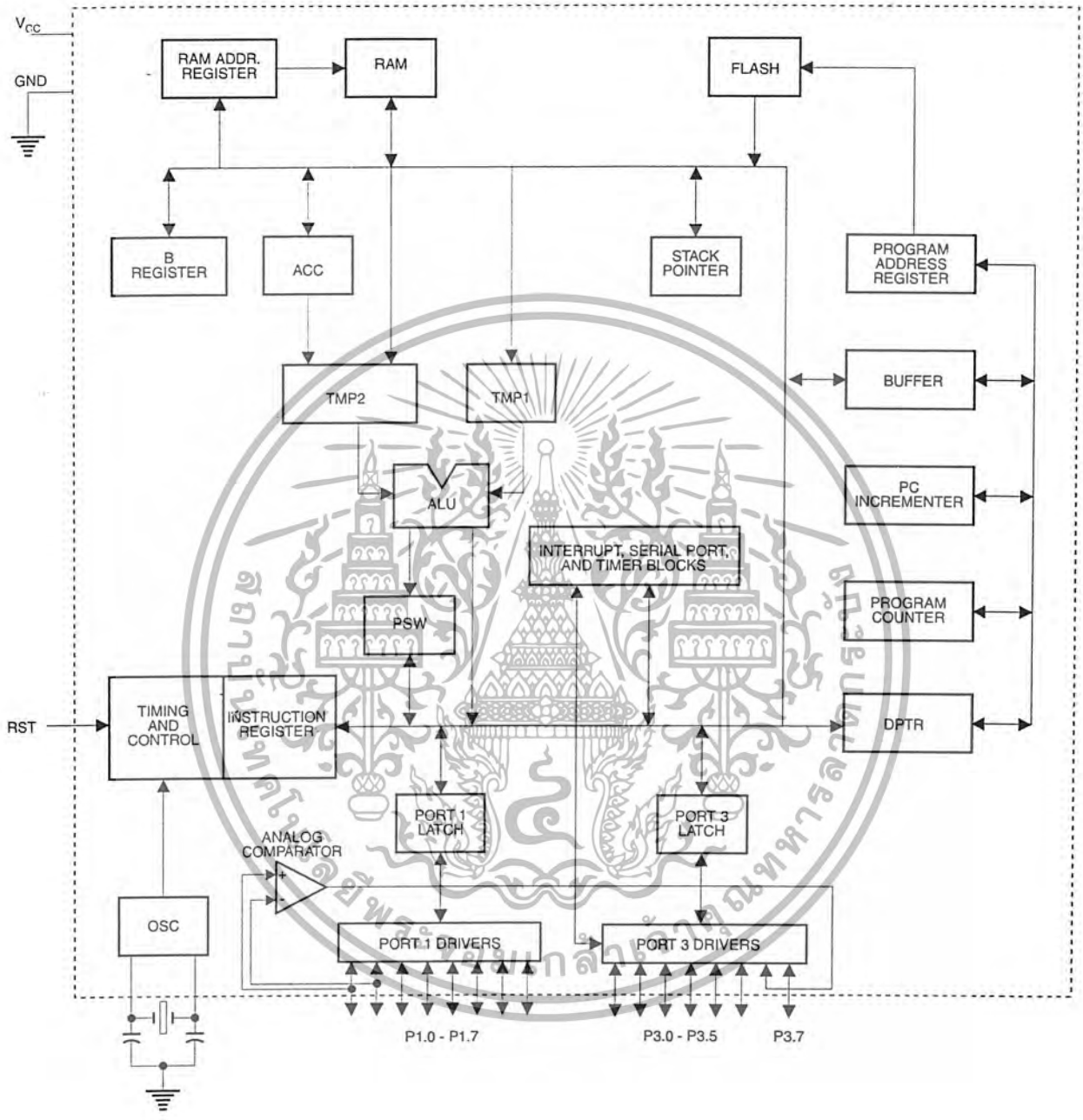


8-bit Microcontroller with 2K Bytes Flash

AT89C2051

Rev. 0368E–02/00

Block Diagram



COMPAQ

Compaq iPAQ H3600 Hardware Design Specification - Version 0.2f

Copyright Notice

Overview

The iPAQ H3600 is a small form-factor, multimedia-centric PDA with versatile expansion capabilities. color reflective TFT LCD, stereo audio output to 3.5mm phone jack and an expansion pack conne mono microphone and speaker, a high performance/low power SA-1110 (206 MHz) CPU, up to 3: SDRAM, up to 32 megabytes of flash ROM, touch panel input, function/application buttons/switches, 232C serial port, a USB client port, a notification/battery charger LED, and an expansion pack interfa

The iPAQ H3600 main unit consists of main board, switch board, color LCD and touch panel modu battery pack.

Document Conventions

The names of signals that are active low will have a '#' suffix.

1. Main board

The main board of the iPAQ H3600 consists of the SA-1110 CPU, flash ROM, SDRAM, serial port (client port, FIR, touch panel interface, stereo audio codec, audio in/out circuit, microcontroller, anc pack interface and connector.

1.1. SA-1110 CPU

The SA-1110 was chosen for its low power, high performance and high integration of peripherals. Th the following features:

- StrongARM core: 150 Dhrystone 2.1 MIPS @ 133 MHz, 235 Dhrystone 2.1 MIPS @ 206 MHz
- Low power (Normal mode)
  - < 240mW @ 1.55V core/133 MHz(TBC)
  - < 400mW @ 1.75V core/206 MHz(TBC)
- Integrated clock generation
  - Internal phase-locked loop (PLL), 3.6864-MHz oscillator, 32.768-kHz oscillator
- Power-management features
  - normal (full-on) mode, idle (power-down) mode, sleep (power-down) mode
- 32-way set-associative caches
- 16 kilobyte instruction cache, 8 kilobyte write-back data cache
  - 32-entry MMUs
  - read/write buffer

Additional features built into SA-1110 chipset

- Memory controller supporting ROM, synchronous mask ROM (SMROM), flash, DRAM, synchron (SDRAM), SRAM, and SRAM-like variable latency I/O
- LCD controller
  - 1-, 2-, or 4-bit gray-scale levels, 8-, 12-, or 16-bit color levels
- Serial communications module supporting
  - SDLC, 230-Kbps UART
- Touch-screen, audio, telecom port
- Infrared data (IrDA) serial port
  - 115 Kbps, 4 Mbps
- Six-channel DMA controller
- Integrated two-slot PCMCIA controller
- Twenty-eight general-purpose I/O ports
- Real-time clock with interrupt capability
- On-chip oscillators for clock sources
- Interrupt controller
- Power-management features
- Normal (full-on) mode
- Idle (power-down) mode
- Sleep (power-down) mode
- Four general-purpose interruptible timers
- 12-Mbps USB device controller
- Synchronous serial port (UCB1100, UCB1200, SPI, TI, Wire)
- 256 mini-ball grid array (mBGA)

1.1.1. Clock generators

SA-1110 uses only two crystals, 32.768 KHz and 3.6864 MHz, to generate all frequency needed.

Please check SA-1110 Developer's Manual section 8.3 and Appendix B and C on the requirement crystals. The frequency column using 3.579545 MHz crystal is for reference only, the iPAQ H3600 dc crystal in its design.

The core frequency can be programmed to the values in table below.

CCF<4..0>	Core Clock Frequency w/3.6864 MHz X'tal	Core Clock Frequency w/3.579545 MHz X'tal

Related Links

- Press Release
- Support
- Warranty

msrp **$79.99** U.S.

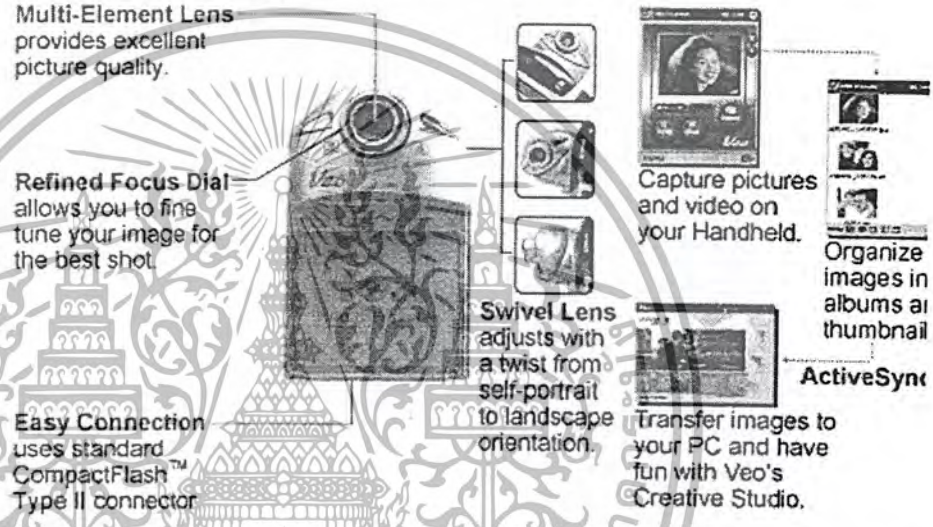
Product Overview | Features | Specifications | Minimum Systems Requirements | Pocke
Download Brochure (PDF) | Larger Product View | Larger Product View 2 | Buy Now

Turn your PDA into a digital camera!

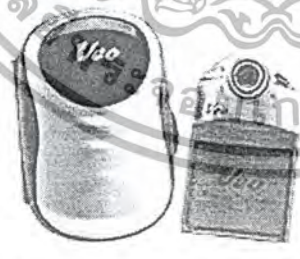
- Plugs right in to your expansion slot
- Capture photos and videos anywhere
- Lightweight and rugged

Plug the Photo Traveler into your PocketPC's CF memory card slot and you're ready to take snapshots with your handheld! Full color preview lets you frame the photo and instantly see the result. The swivel lens allows you to easily capture both self-portrait and traditional forward-facing photos. An adjustable focus lens ensures crisp, clear photos both close up and far away. Use ActiveSync to download images to your PC and enjoy the benefits of the image editing features of Veo Creative Studio.

Product Overview:



Features:



Rugged Protective Travel Case
You'll never have to leave your camera behind. Bring it anywhere, knowing that it's protected in its custom padded case.

Full Color Preview
Allows you to frame your image on your handheld screen in real time.

Thumbnail Viewe
Quickly page throu photo album to finc picture you're looki

Specifications:

- 640x480 digital resolution
- Takes still photos and video clips
- Adjustable self timer

Minimum System Requirements:

- Microsoft 98/ME/2000/XP with
- 166 MHz (or higher)

- Captures standard JPEG pictures
- Automatic exposure, white balance and color control
- CF Expansion Slot
- Swivel lens with adjustable focus
- Compact, lightweight design (weighs < 1 oz)

- 200 MB Available Hard Disk Spa
- 32 MB RAM or Better
- Pocket PC Handheld with Active Cradle
- 800 x 600 Display with 16-Bit Co
- 4x CD-ROM or better

Pocket PC:

- iPAQ 36xx, 37xx, 38xx, 39xx (wi card extension pack)
- iPAQ h2210, h2215 (Requires la Windows Mobile 2003 software Pocket PCs (Pocket PC 2003) d from our webpage.
- Hp5550, HP5150 and HP3970.
- Toshiba e570, e740
- Dell Axim X5

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

LM555/NE555/SA555

Single Timer

Features

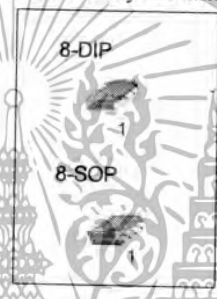
- High Current Drive Capability (200mA)
- Adjustable Duty Cycle
- Temperature Stability of 0.005%/°C
- Timing From μSec to Hours
- Turn off Time Less Than 2μSec

Applications

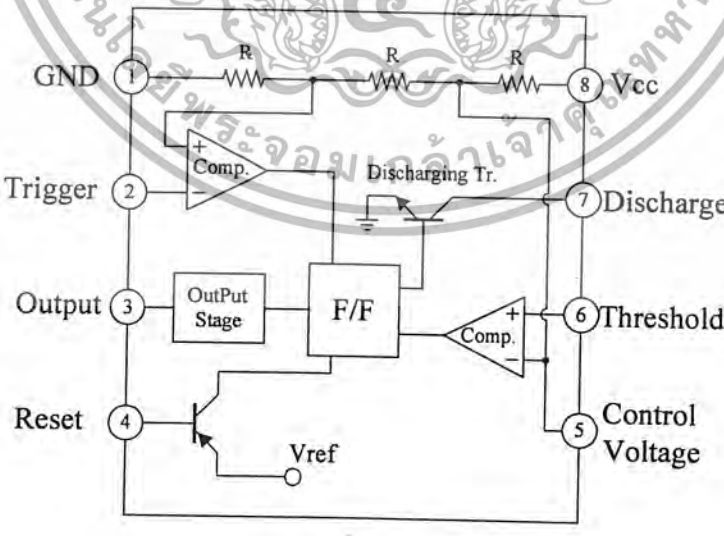
- Precision Timing
- Pulse Generation
- Time Delay Generation
- Sequential Timing

Description

The LM555/NE555/SA555 is a highly stable controller capable of producing accurate timing pulses. With a monostable operation, the time delay is controlled by one external resistor and one capacitor. With an astable operation, the frequency and duty cycle are accurately controlled by two external resistors and one capacitor.



Internal Block Diagram



Absolute Maximum Ratings (T_A = 25°C)

Parameter	Symbol	Value	Unit
Supply Voltage	V_{CC}	16	V
Lead Temperature (Soldering 10sec)	T_{LEAD}	300	°C
Power Dissipation	P_D	600	mW
Operating Temperature Range LM555/NE555 SA555	T_{OPR}	0 ~ +70 -40 ~ +85	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics

(T_A = 25°C, V_{CC} = 5 ~ 15V, unless otherwise specified)

Parameter	Symbol	Conditions	Min.	Typ.	Max.	Unit
Supply Voltage	V_{CC}	-	4.5	-	16	V
Supply Current (Low Stable) (Note1)	I_{CC}	V_{CC} = 5V, R_L = ∞	-	3	6	mA
		V_{CC} = 15V, R_L = ∞	-	7.5	15	mA
Timing Error (Monostable) Initial Accuracy (Note2) Drift with Temperature (Note4) Drift with Supply Voltage (Note4)	ACCUR $\Delta t/\Delta T$ $\Delta t/\Delta V_{CC}$	R_A = 1kΩ to100kΩ C = 0.1μF	-	1.0 50 0.1	3.0 0.5	% ppm/°C %/V
Timing Error (Astable) Intial Accuracy (Note2) Drift with Temperature (Note4) Drift with Supply Voltage (Note4)	ACCUR $\Delta t/\Delta T$ $\Delta t/\Delta V_{CC}$	R_A = 1kΩ to 100kΩ C = 0.1μF	-	2.25 150 0.3	-	% ppm/°C %/V
Control Voltage	V_C	V_{CC} = 15V	9.0	10.0	11.0	V
		V_{CC} = 5V	2.6	3.33	4.0	V
Threshold Voltage	V_{TH}	V_{CC} = 15V	-	10.0	-	V
		V_{CC} = 5V	-	3.33	-	V
Threshold Current (Note3)	I_{TH}	-	-	0.1	0.25	μA
Trigger Voltage	V_{TR}	V_{CC} = 5V	1.1	1.67	2.2	V
		V_{CC} = 15V	4.5	5	5.6	V
Trigger Current	I_{TR}	V_{TR} = 0V		0.01	2.0	μA
Reset Voltage	V_{RST}	-	0.4	0.7	1.0	V
Reset Current	I_{RST}	-		0.1	0.4	mA
Low Output Voltage	V_{OL}	V_{CC} = 15V I_{SINK} = 10mA I_{SINK} = 50mA	-	0.06 0.3	0.25 0.75	V V
		V_{CC} = 5V I_{SINK} = 5mA	-	0.05	0.35	V
High Output Voltage	V_{OH}	V_{CC} = 15V I_{SOURCE} = 200mA I_{SOURCE} = 100mA	12.75	12.5 13.3	-	V V
		V_{CC} = 5V I_{SOURCE} = 100mA	2.75	3.3	-	V
Rise Time of Output (Note4)	t_R	-	-	100	-	ns
Fall Time of Output (Note4)	t_F	-	-	100	-	ns
Discharge Leakage Current	I_{LKG}	-	-	20	100	nA

Notes:

1. When the output is high, the supply current is typically 1mA less than at V_{CC} = 5V.
2. Tested at V_{CC} = 5.0V and V_{CC} = 15V.
3. This will determine the maximum value of $R_A + R_B$ for 15V operation, the max. total R = 20MΩ, and for 5V operation, the max. total R = 6.7MΩ.
4. These parameters, although guaranteed, are not 100% tested in production.

Application Information

Table 1 below is the basic operating table of 555 timer:

Table 1. Basic Operating Table

Threshold Voltage (Vth)(PIN 6)	Trigger Voltage (Vtr)(PIN 2)	Reset(PIN 4)	Output(PIN 3)	Discharging Tr. (PIN 7)
Don't care	Don't care	Low	Low	ON
Vth > 2Vcc / 3	Vth > 2Vcc / 3	High	Low	ON
Vcc / 3 < Vth < 2 Vcc / 3	Vcc / 3 < Vth < 2 Vcc / 3	High	-	-
Vth < Vcc / 3	Vth < Vcc / 3	High	High	OFF

When the low signal input is applied to the reset terminal, the timer output remains low regardless of the threshold voltage or the trigger voltage. Only when the high signal is applied to the reset terminal, the timer's output changes according to threshold voltage and trigger voltage.

When the threshold voltage exceeds 2/3 of the supply voltage while the timer output is high, the timer's internal discharge Tr. turns on, lowering the threshold voltage to below 1/3 of the supply voltage. During this time, the timer output is maintained low. Later, if a low signal is applied to the trigger voltage so that it becomes 1/3 of the supply voltage, the timer's internal discharge Tr. turns off, increasing the threshold voltage and driving the timer output again at high.

1. Monostable Operation

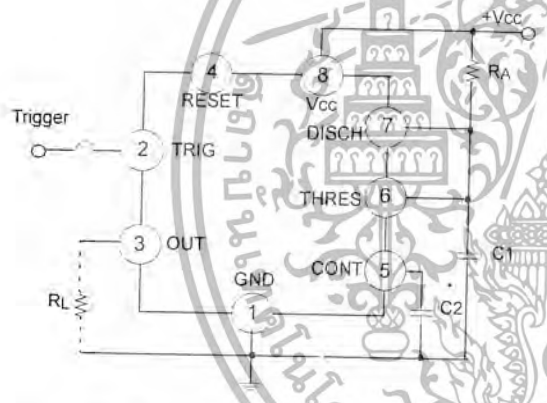


Figure 1. Monoatable Circuit

Figure 2. Resistance and Capacitance vs. Time delay(t_d)

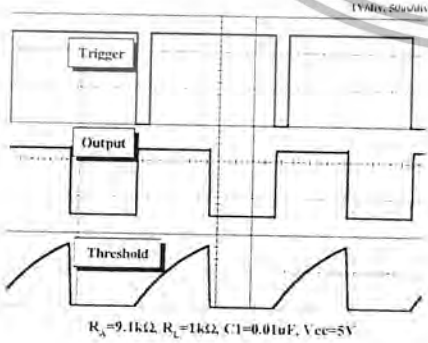


Figure 3. Waveforms of Monostable Operation

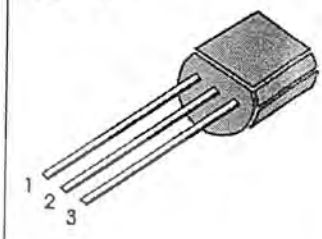
2N2222A

NPN SILICON TRANSISTOR

■ Description
- General Purpose Application
- Switching Transistor

■ Features
- Low Leakage Current: I_{CBO}=10nA(Max.) [V_{CB}=60V, I_E=0mA]
- Low Saturation Voltage: $V_{CE(sat)}$=0.4V(Max.) [I_C=150mA, I_B=15mA]
- Large Collector Current (I_{Cmax}=600mA)
- Complementary Pair with 2N2907A

TO-92



1.Emitter 2.Base 3.Collector

■ ABSOLUTE MAXIMUM RATINGS (T_A=25℃)

Characteristic	Symbol	Limit	Unit
Collector-Base Voltage	V_{CBO}	75	V
Collector-Emitter Voltage	V_{CEO}	40	V
Emitter-Base Voltage	V_{EBO}	6	V
Collector Current	I_C	600	mA
Collector Dissipation	P_C	625	mW
Junction Temperature	T_J	150	℃
Storage Temperature	T_{STG}	-55 ~ 150	℃

■ ELECTRICAL CHARACTERISTICS (T_A=25℃)

Characteristic	Symbol	Test Condition	Min.	Max.	Unit
Collector-Emitter Breakdown Voltage	BV_{CEO}	I_C=10mA	40		V
Collector-Base Breakdown Voltage	BV_{CBO}	I_C=10uA	75		V
Emitter-Base Breakdown Voltage	BV_{EBO}	I_E=10uA	6		V
Collector Cut-Off Current	I_{CBO}	V_{CB}=60V		10	nA
DC Current Gain	h_{FE}	V_{CE}=10V, I_C=10mA	75		
Collector-Emitter Saturation Voltage	$V_{CE(sat)}$	I_C=150mA, I_B=15mA		0.3	V
Current Gain-Bandwidth Product	f_T	V_{CE}=20V, I_C=20mA, f=100MHz	250		MHz
Collector Output Capacitance	C_{OB}	V_{CB}=10V, f=1MHz		8.0	pF

Previous return next