บทที่ 3
วิธีการดำเนินงาน

เนื้อหาในบทนี้กล่าวถึงเรื่องการเริ่มการศึกษาข้อมูลเกี่ยวกับการสร้างโครงสร้างตัวรถและการออกแบบวงจรควบคุมซึ่งมีรายละเอียดดังต่อไปนี้

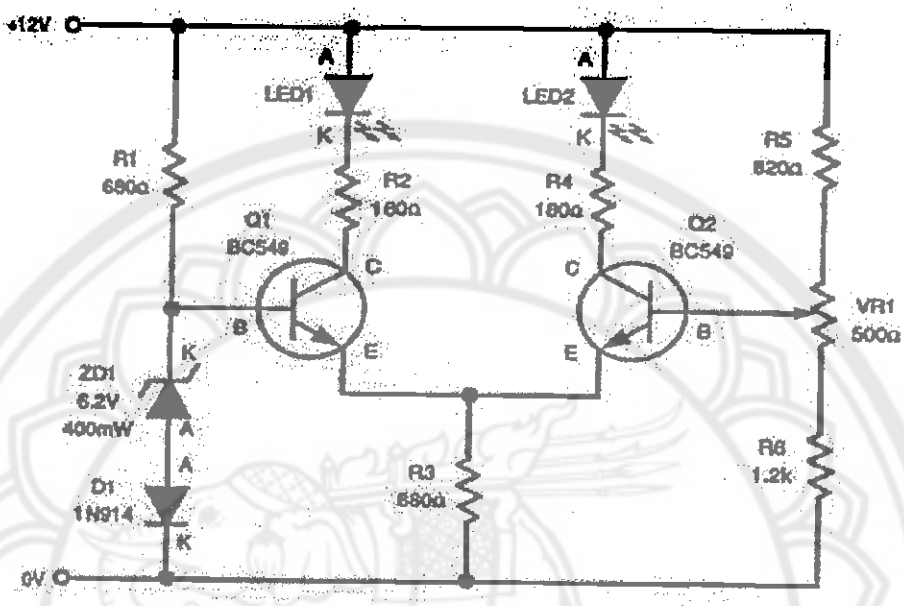
<u>ระยะเวลาตั้งแต่เดือนมีนาคม – กันยายน 2543 โดยมีเนื้อหาดังนี้</u>

- ศึกษาวงจรตรวจสอบแรงดันแบตเตอรี่
- ศึกษาพาวเวอร์ซัพพลายที่ใช้ในการจ่ายให้บอร์ดอื่นๆ
- ศึกษาการใช้งานของบอร์ดไมโครคอนโทรลเลอร์ และเลือกโปรแกรมที่ใช้
- ศึกษาการใช้ไดรเวอร์ (Driver) ในการขับมอเตอร์
- ศึกษาวงจร A/D เพื่อนำมาใช้ในการสั่งงานบอร์ดไมโครคอนโทรลเลอร์
- ศึกษาเกี่ยวกับรอมอีมูเลเตอร์ (ROM Emulator) เพื่อเชื่อมต่อกับ PC ในการเขียนโปรแกรม
- วงจรบัฟเฟอร์
- ระบบเบรค
- แผนภาพระบบโดยรวมของรถไฟฟ้าคนพิการ
- โครงสร้างรถไฟฟ้าคนพิการ

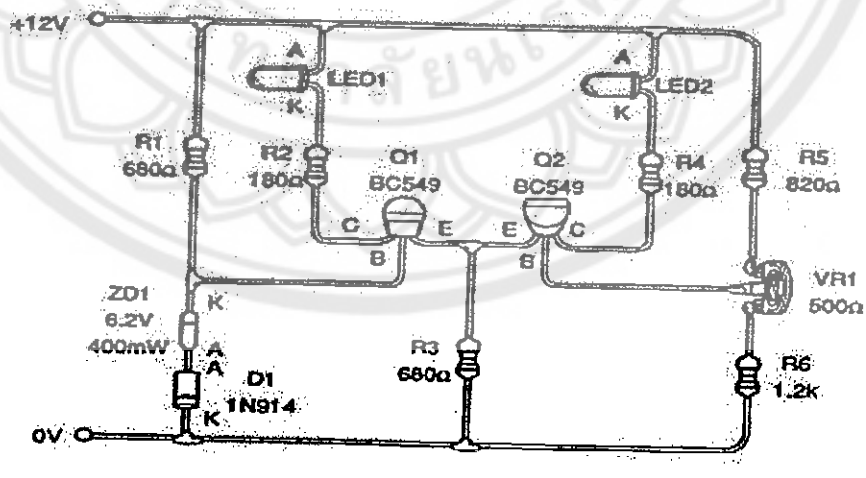
3.1 ศึกษาวงจรตรวจสอบแรงดันแบตเตอรี่

แบตเตอรี่ที่ใช้กับรถไฟฟ้าคนพิการมี 4 ลูก ยี่ห้อ GS BATTERTY ได้ถูกแบ่งการจ่ายไฟดังนี้

- **24 V ใช้แบตตอรี่ 12 V 2 ลูกจ่ายไฟให้มอเตอร์ 2 ตัว**
- **12 V ใช้แบตตอรี่ 12 V 1 ลูกจ่ายไฟให้เพาเวอร์ซัพพลาย**
- **12 V ใช้แบตตอรี่ 12 V 1 ลูกจ่ายไฟให้พัดลมระบายความร้อน**



รูปที่ 3.1 วงจรตรวจสอบแรงดันแบตเตอรี่



รูปที่ 3.2 ลักษณะการต่อวงจรตรวจสอบแรงดันแบตเตอรี่

วงจรตรวจสอบแรงดันแบตเตอรี่ที่แสดงในรูปที่ 3.1 ใช้ตรวจสอบแรงดันในช่วง 11.5 โวลท์ ถึง 14.0 โวลท์ โดยการปรับเฉพาะการทำงาน Q2 ที่ VR1 ก็คือเป็นการปรับไบแอสให้กับทรานซิสเตอร์ Q2 นั่นเอง ในขณะที่แบตเตอรี่ยังมีไฟเต็มอยู่ กระแสจะไหลไปทาง R5 และ VR1 ก่อนเข้าขาเบสของ Q2 เป็นผลให้ Q2 อยู่ในสภาวะนำกระแส LED2 จึงสว่างขึ้นมาได้

ถ้าแบตเตอรี่อ่อนกำลังลงจะทำให้ Q2 หยุดนำกระแส เป็นผลให้ Q1 นำกระแสแทน LED1 จึงสว่างขึ้นมาแทนที่ สำหรับ ZD1 กับ D1 ที่ต่ออยู่ในวงจรเป็นตัวกำหนดแรงดันไบแอสให้แก่ Q1 และในขณะที่วงจรนี้ทำงาน จะกินกระแสประมาณ 20 – 30 mA เท่านั้น

3.2 ศึกษาพาวเวอร์ซัพพลายที่ใช้ในการจ่ายให้บอร์ดอื่นๆ

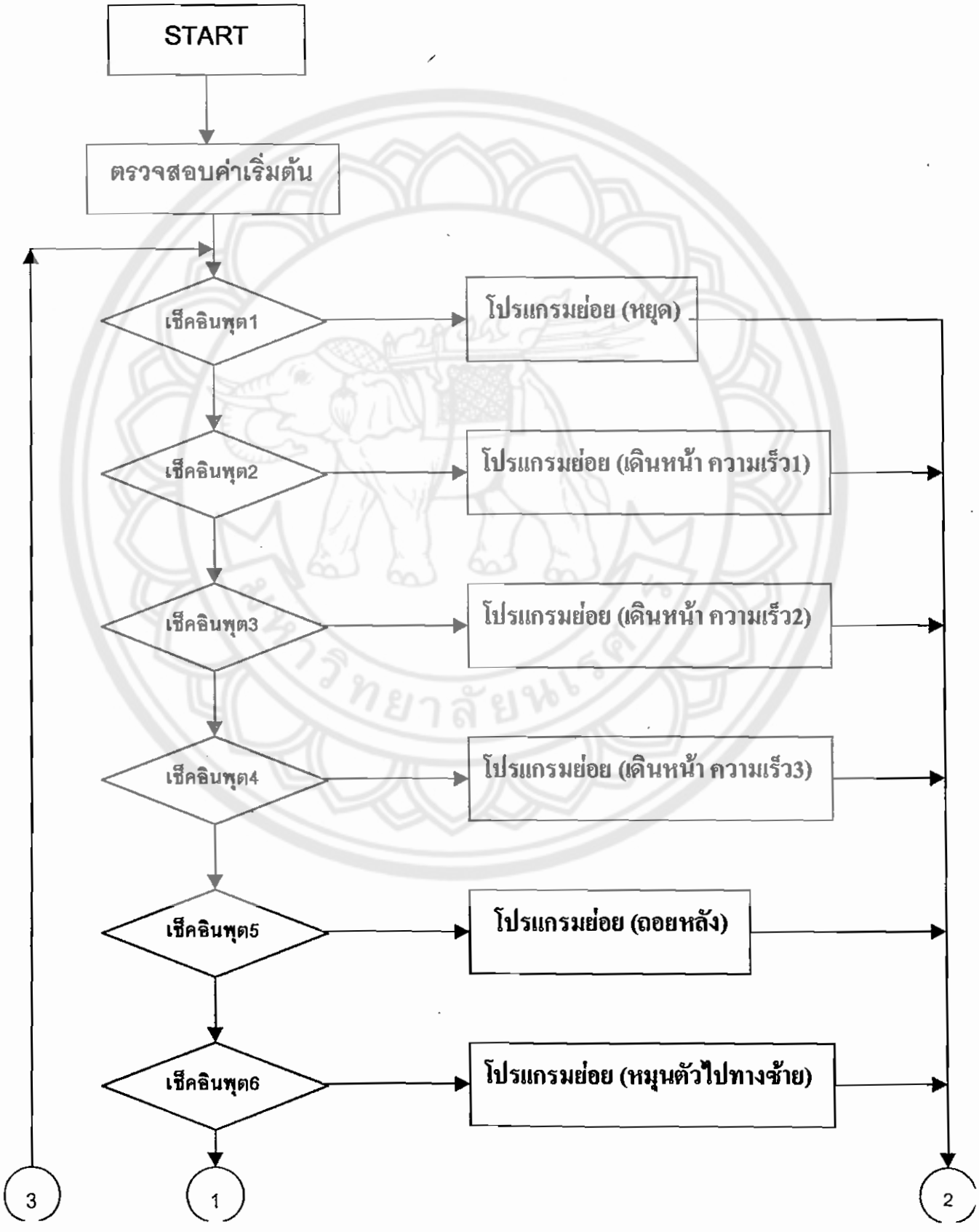
พาวเวอร์ซัพพลายเป็นตัวแปลงไฟจาก 12 V เป็น 5 V เพื่อจ่ายไฟให้กับบอร์ดวงจรดังต่อไปนี้

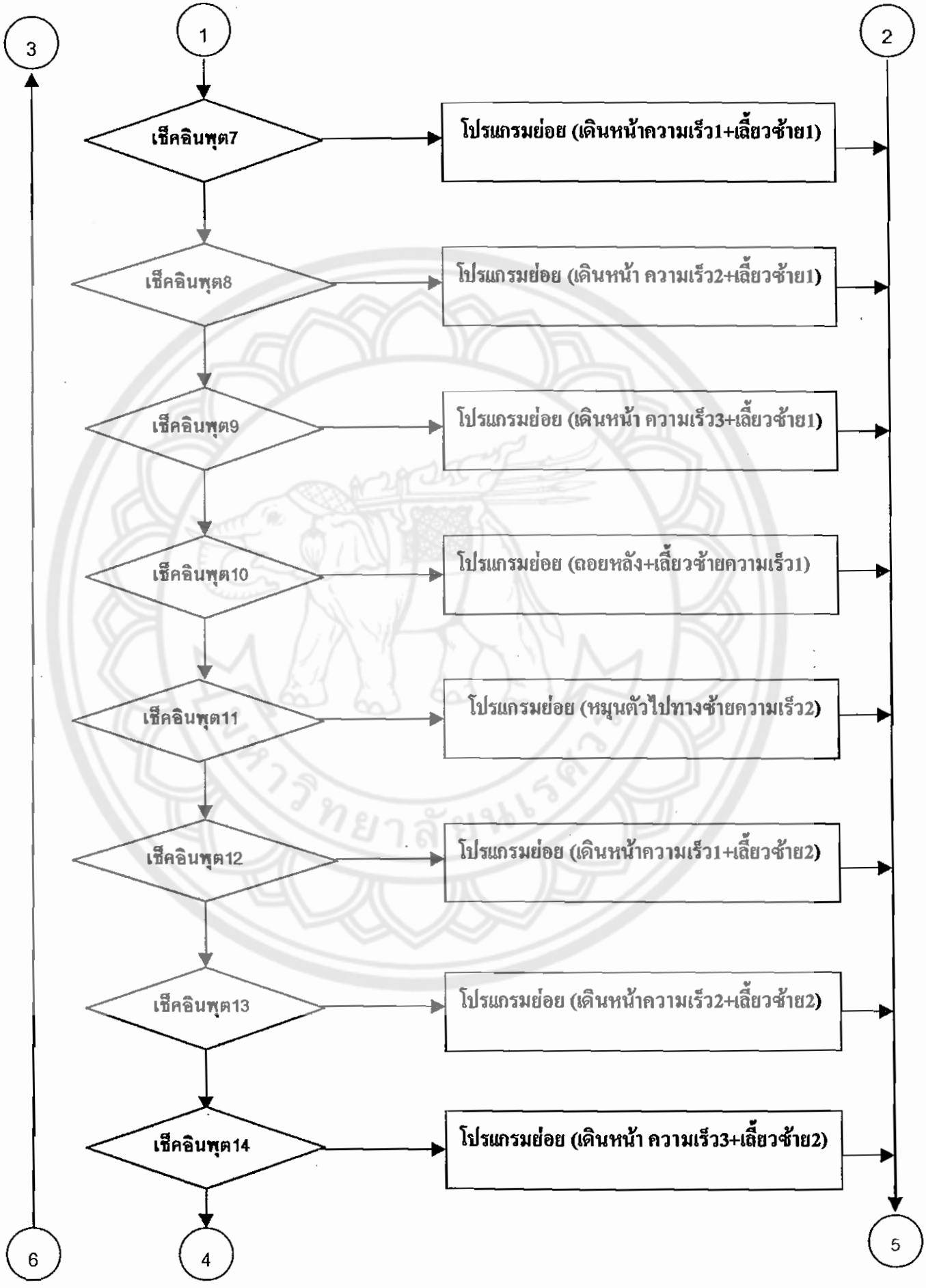
- วงจรไดรเวอร์
- วงจรคอนโทรลเลอร์
- วงจร A/D Converter
- วงจรบัฟเฟอร์
- วงจรเบรค

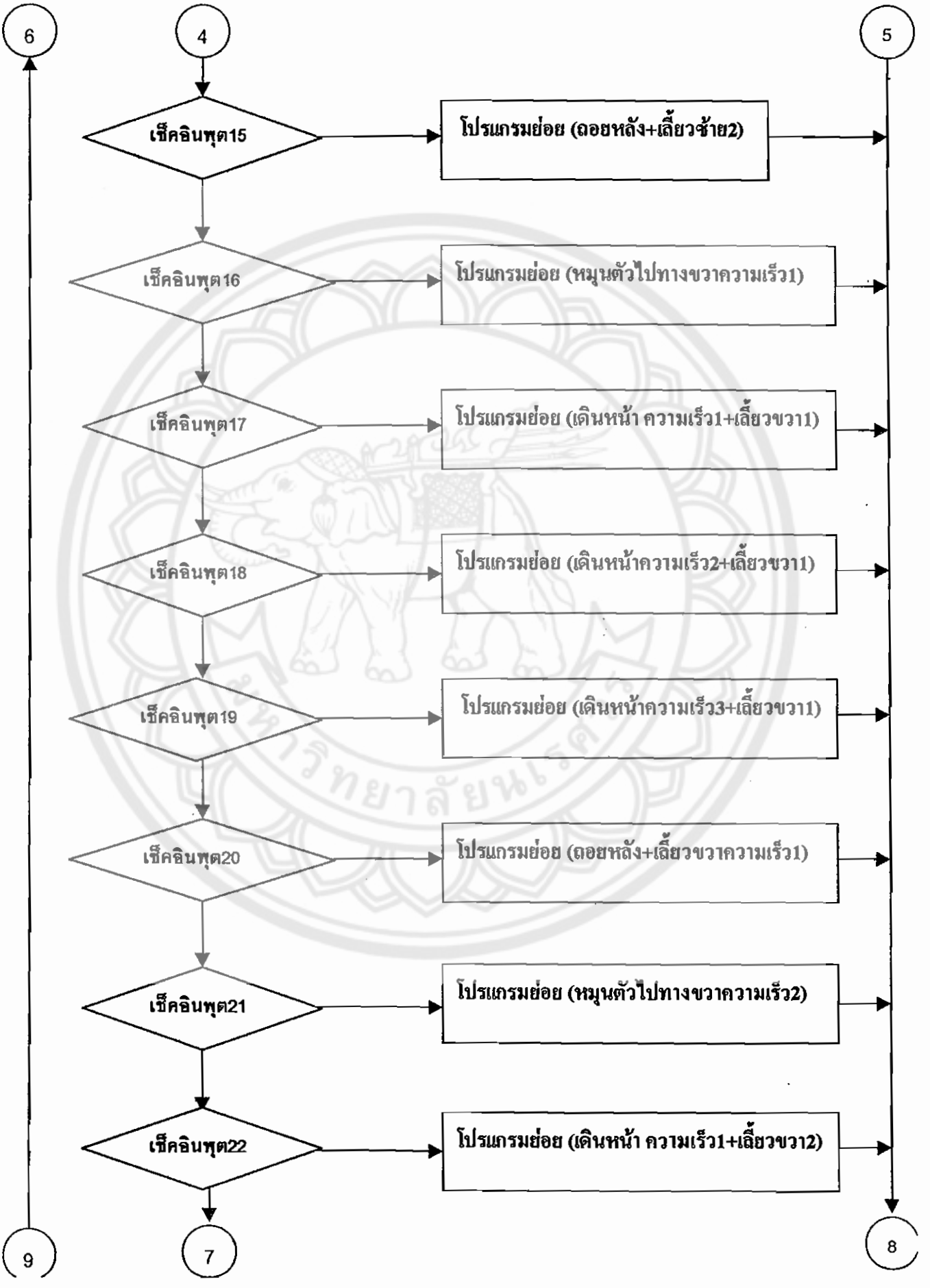
3.3 ศึกษาแล้วเลือกบอร์ดไมโครคอนโทรลเลอร์ และเลือกโปรแกรมที่ใช้

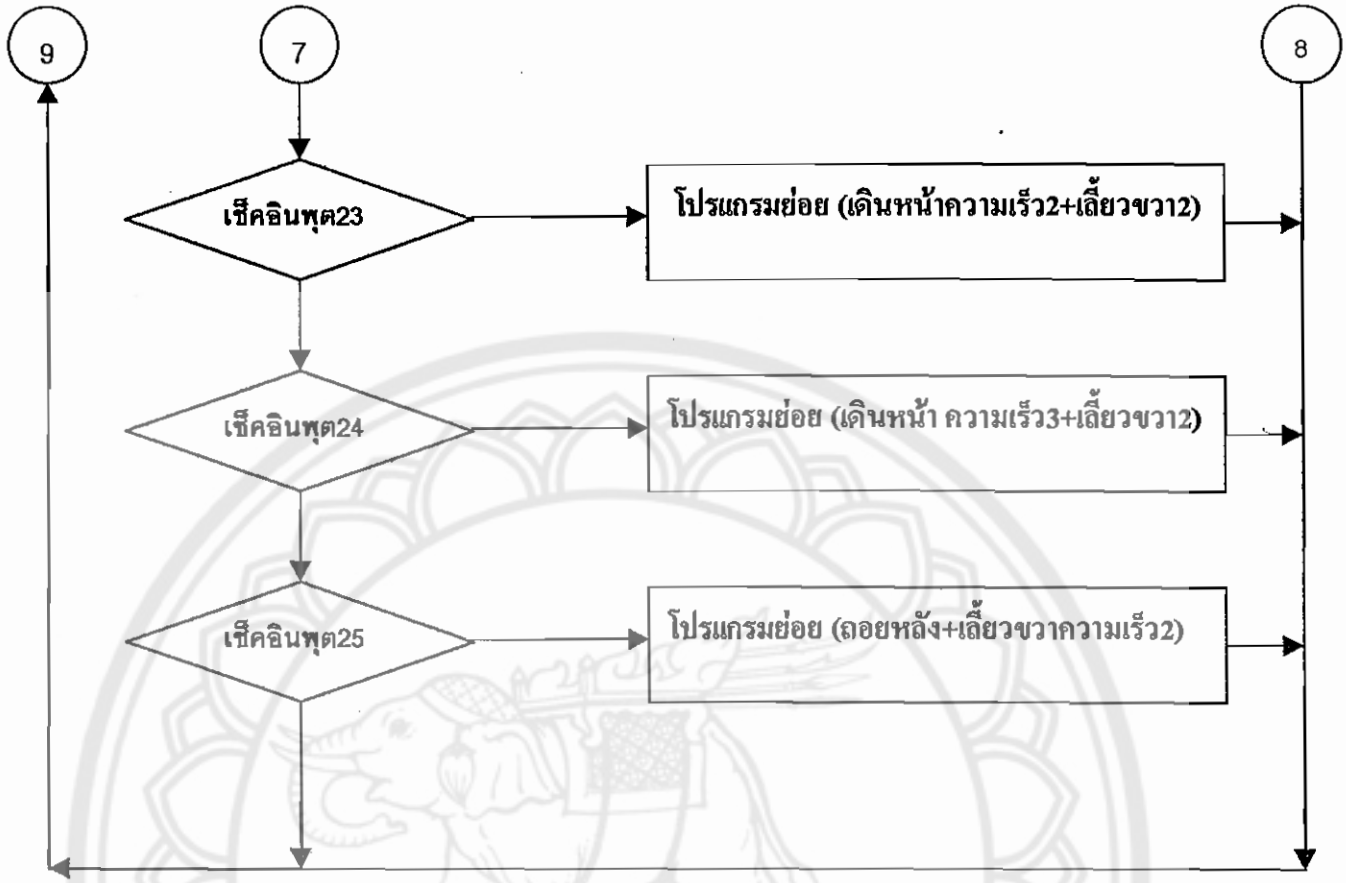
ในการเลือกใช้ชุดไมโครคอนโทรลเลอร์ เราเลือกใช้ชุดไมโครคอนโทรลเลอร์ของบริษัท ETT ซึ่งมี IC 8255 เป็นอินพุตเอาท์พุต (I/O) อยู่บนบอร์ดเลย ทำให้เลือกใช้พอร์ทอินพุตเอาท์พุตได้ถึง 3 พอร์ท ด้านโปรแกรมที่ใช้เขียนเราใช้ภาษา C ในการเขียนโปรแกรม แล้วใช้โปรแกรมบริฟ (Brief) เป็นตัวอิดิเตอร์ (Iditor) และใช้ C51 เป็นตัวคอมไพเลอร์ (Compiler)

แผนผังการเขียนโปรแกรม (Flow Chart)




1
2
3
เช็คอินพุต7
โปรแกรมย่อย (เดินหน้าความเร็ว1+เลี้ยวซ้าย1)
เช็คอินพุต8
โปรแกรมย่อย (เดินหน้า ความเร็ว2+เลี้ยวซ้าย1)
เช็คอินพุต9
โปรแกรมย่อย (เดินหน้า ความเร็ว3+เลี้ยวซ้าย1)
เช็คอินพุต10
โปรแกรมย่อย (ถอยหลัง+เลี้ยวซ้ายความเร็ว1)
เช็คอินพุต11
โปรแกรมย่อย (หมุนตัวไปทางซ้ายความเร็ว2)
เช็คอินพุต12
โปรแกรมย่อย (เดินหน้าความเร็ว1+เลี้ยวซ้าย2)
เช็คอินพุต13
โปรแกรมย่อย (เดินหน้าความเร็ว2+เลี้ยวซ้าย2)
เช็คอินพุต14
โปรแกรมย่อย (เดินหน้า ความเร็ว3+เลี้ยวซ้าย2)
6
4
5


6
4
5
เช็คอินพุต15
โปรแกรมย่อย (ถอยหลัง+เลี้ยวซ้าย2)
เช็คอินพุต16
โปรแกรมย่อย (หมุนตัวไปทางขวาความเร็ว1)
เช็คอินพุต17
โปรแกรมย่อย (เดินหน้า ความเร็ว1+เลี้ยวขวา1)
เช็คอินพุต18
โปรแกรมย่อย (เดินหน้าความเร็ว2+เลี้ยวขวา1)
เช็คอินพุต19
โปรแกรมย่อย (เดินหน้าความเร็ว3+เลี้ยวขวา1)
เช็คอินพุต20
โปรแกรมย่อย (ถอยหลัง+เลี้ยวขวาความเร็ว1)
เช็คอินพุต21
โปรแกรมย่อย (หมุนตัวไปทางขวาความเร็ว2)
เช็คอินพุต22
โปรแกรมย่อย (เดินหน้า ความเร็ว1+เลี้ยวขวา2)
9
7
8



รูปที่ 3.3 แผนผังการเขียนโปรแกรม

โปรแกรมที่ใช้งานจริงในการเขียนระบบควบคุมการขับเคลื่อนของรถไฟฟ้าคนพิการ มีดังต่อไปนี้

```
#include <reg51.h>
#include <string.h>
#include <absacc.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>

#define Pa    XBYTE [0xE0E0]
#define Pb    XBYTE [0xE0E1]
#define Pc    XBYTE [0xE0E2]
#define Pcon  XBYTE [0XE0E3]

xdata unsigned int quantum;
xdata unsigned int duty1,duty2,period;
xdata unsigned char mPa;

void delay(unsigned int time)
{
  unsigned int i;
  for(i=0;i < time;i++)
      {
      }
}
void drive_dir1(unsigned char n_dir)
{
  if (n_dir ==1)
      {
      mPa=mPa | 0x01;
```

```
        }
    else
        {
        mPa=mPa & 0xfe;
        }

    Pa=mPa;

}
void drive_dir2(unsigned char n_dir)
{
    if (n_dir ==1)
        {
        mPa=mPa | 0x08;
        }
    else
        {
        mPa=mPa & 0xf7;
        }
    Pa=mPa;

}

void drive_pwm1(unsigned char n_pwm)
{
    if (n_pwm ==1)
        {
        mPa=mPa | 0x02;
        }
    else
```

```
        {
        mPa=mPa & 0xfd;
        }
    Pa=mPa;
}
void drive_pwm2(unsigned char n_pwm)
{
    if (n_pwm ==1)
        {
        mPa=mPa | 0x10;
        }
    else
        {
        mPa=mPa & 0xef;
        }
    Pa=mPa;
}
void drive_en1(unsigned char n_en)
{
    if (n_en ==1)
        {
        mPa=mPa | 0x04;
        }
    else
        {
        mPa=mPa & 0xfb;
        }
    Pa=mPa;
}
```

```
void drive_en2(unsigned char n_en)
{
  if (n_en ==1)
     {
     mPa=mPa | 0x20;
     }
  else
     {
     mPa=mPa & 0xdf;
     }
  Pa=mPa;
}

void init_timer(void)
{
  TMOD=0x01;
  TH0=0xff;
  TL0=0x80;
  ET0=1;
  TR0=1;
  EA=1;
}
void int_quantum(void) interrupt 1
{
//Interrupt Timer0
  if (quantum<=duty1)
     {
     drive_pwm1(0);
     }
  else
```

```
        {
        drive_pwm1(1);
        }
    if (quantum<=duty2)
        {
        drive_pwm2(0);
        }
    else
        {
        drive_pwm2(1);
        }

    if (quantum>=period)
        {
        quantum=0;
        }

    quantum=quantum+1;
    TH0 = 0xff;
    TL0 = 0x80;
}
void main(void)
 {
   xdata unsigned int i;
      delay(10000);
    Pcon=0x8E;
    init_timer();
    period=100;
   duty1=0;
   duty2=0;
```

```
 drive_en1(1);
 drive_en2(1);
while(1)
  {
  if(((Pb>=0x48)&&(Pb<=0x6F))&&((Pc>=0x28)&&(Pc<=0x47))) // 1stop
    {
    duty1=00;
    duty2=00;
    drive_en1(1);
    drive_en2(1);
delay(1000);

    }
  if(((Pb>=0x30)&&(Pb<=0x47))&&((Pc>=0x28)&&(Pc<=0x47))) // 2 front1
    {
     duty1=60;
     duty2=60;
     drive_en1(0);
     drive_en2(0);
     drive_dir1(1);
     drive_dir2(1);
     }
  if(((Pb>=0x10)&&(Pb<=0x2F))&&((Pc>=0x28)&&(Pc<=0x47))) // 3 front2
     {
     duty1=85;
     duty2=85;
     drive_en1(0);
     drive_en2(0);
     drive_dir1(1);
     drive_dir2(1);
```

```
  }
if(((Pb>=0x00)&&(Pb<=0x0F))&&((Pc>=0x28)&&(Pc<=0x47))) // 4 front3
  {
  duty1=100;
  duty2=100;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(1);
  drive_dir2(1);
  }
if(((Pb>=0x70)&&(Pb<=0x80))&&((Pc>=0x28)&&(Pc<=0x47))) // 5 back
   {
   duty1=70;
   duty2=70;
   drive_en1(0);
   drive_en2(0);
   drive_dir1(0);
   drive_dir2(0);
   }
 if(((Pb>=0x48)&&(Pb<=0x6F))&&((Pc>=0x10)&&(Pc<=0x27))) // 6 stop+L1
   {
   duty2=55;
   duty1=55;
   drive_en1(0);
   drive_en2(0);
   drive_dir2(1);
   drive_dir1(0);
   }
if(((Pb>=0x30)&&(Pb<=0x47))&&((Pc>=0x10)&&(Pc<=0x27))) // 7 front1+L1
  {
```

```
duty1=15;
duty2=55;
drive_en1(0);
drive_en2(0);
drive_dir1(1);
drive_dir2(1);
}
if(((Pb>=0x10)&&(Pb<=0x2F))&&((Pc>=0x10)&&(Pc<=0x27))) // 8 front2+L1
  {
  duty1=15;
  duty2=65;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(1);
  drive_dir2(1);
  }
if(((Pb>=0x00)&&(Pb<=0x0F))&&((Pc>=0x10)&&(Pc<=0x27))) // 9 front3+L1
  {
  duty1=15;
  duty2=75;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(1);
  drive_dir2(1);
  }
if(((Pb>=0x70)&&(Pb<=0x80))&&((Pc>=0x10)&&(Pc<=0x27))) // 10 back+L1
  {
  duty1=15;
  duty2=65;
  drive_en1(0);
```

```
drive_en2(0);
drive_dir1(0);
drive_dir2(0);
 }
if(((Pb>=0x48)&&(Pb<=0x6F))&&((Pc>=0x48)&&(Pc<=0x6F))) // 11 stop+R1
 {
 duty2=55;
 duty1=55;
 drive_en1(0);
 drive_en2(0);
 drive_dir2(0);
 drive_dir1(1);
 }
if(((Pb>=0x30)&&(Pb<=0x47))&&((Pc>=0x48)&&(Pc<=0x6F))) // 12 front1+R1
 {
 duty2=15;
 duty1=55;
 drive_en1(0);
 drive_en2(0);
 drive_dir1(1);
 drive_dir2(1);
 }
if(((Pb>=0x10)&&(Pb<=0x2F))&&((Pc>=0x48)&&(Pc<=0x6F))) // 13 front2+R1
 {
 duty2=15;
 duty1=65;
 drive_en1(0);
 drive_en2(0);
 drive_dir1(1);
 drive_dir2(1);
```

```
  }
if(((Pb>=0x00)&&(Pb<=0x0F))&&((Pc>=0x48)&&(Pc<=0x6F))) // 14 front3+R1
  {
  duty2=15;
  duty1=75;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(1);
  drive_dir2(1);
  }
 if(((Pb>=0x70)&&(Pb<=0x80))&&((Pc>=0x48)&&(Pc<=0x6F))) // 15 back+R1
  {
  duty2=15;
  duty1=65;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(0);
  drive_dir2(0);
  }
 if(((Pb>=0x48)&&(Pb<=0x6F))&&((Pc>=0x00)&&(Pc<=0x0F))) // 16 stop+L2
  {
  duty2=80;
  duty1=80;
  drive_en1(0);
  drive_en2(0);
  drive_dir2(1);
  drive_dir1(0);
  }
 if(((Pb>=0x30)&&(Pb<=0x47))&&((Pc>=0x00)&&(Pc<=0x0F))) // 17 front1+L2
  {
```

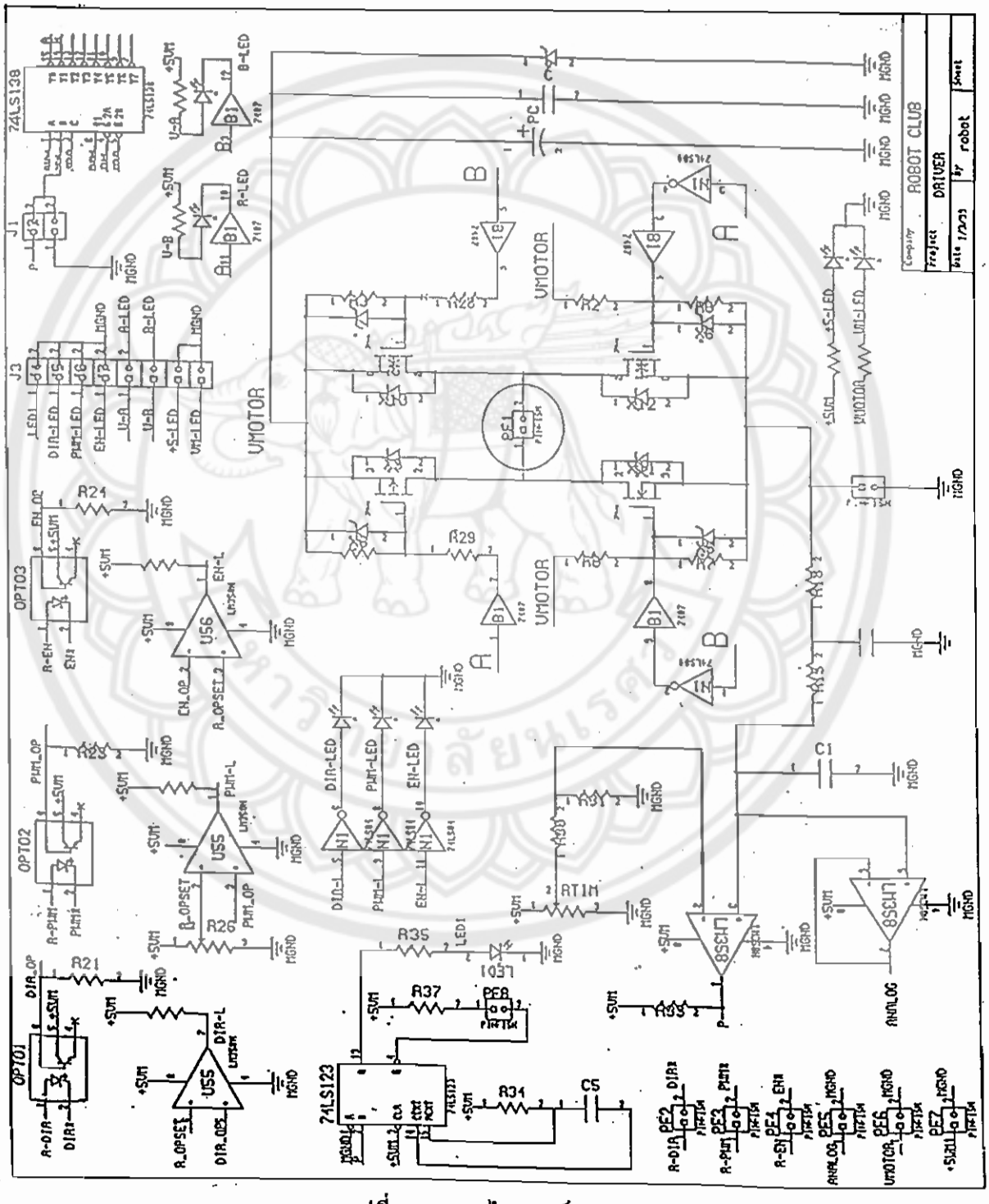
```
duty1=25;
duty2=80;
drive_en1(0);
drive_en2(0);
drive_dir1(1);
drive_dir2(1);
}
if(((Pb>=0x10)&&(Pb<=0x2F))&&((Pc>=0x00)&&(Pc<=0x0F))) // 18 front2+L2
{
duty1=25;
duty2=90;
drive_en1(0);
drive_en2(0);
drive_dir1(1);
drive_dir2(1);
}
if(((Pb>=0x00)&&(Pb<=0x0F))&&((Pc>=0x00)&&(Pc<=0x0F))) // 19 front3+L2
{
duty1=25;
duty2=100;
drive_en1(0);
drive_en2(0);
drive_dir1(1);
drive_dir2(1);
}
if(((Pb>=0x70)&&(Pb<=0x80))&&((Pc>=0x00)&&(Pc<=0x0F))) // 20 back+L2
{
duty1=25;
duty2=90;
drive_en1(0);
```

```
drive_en2(0);
drive_dir1(0);
drive_dir2(0);
 }
if(((Pb>=0x48)&&(Pb<=0x6F))&&((Pc>=0x70)&&(Pc<=0x80))) // 21 stop+R2
 {
 duty2=80;
 duty1=80;
 drive_en1(0);
 drive_en2(0);
 drive_dir2(0);
 drive_dir1(1);
 }
if(((Pb>=0x30)&&(Pb<=0x47))&&((Pc>=0x70)&&(Pc<=0x80))) // 22 front1+R2
 {
 duty2=25;
 duty1=80;
 drive_en1(0);
 drive_en2(0);
 drive_dir1(1);
 drive_dir2(1);
 }
if(((Pb>=0x10)&&(Pb<=0x2F))&&((Pc>=0x70)&&(Pc<=0x80))) // 23 front2+R2
 {
 duty2=25;
 duty1=90;
 drive_en1(0);
 drive_en2(0);
 drive_dir1(1);
 drive_dir2(1);
```

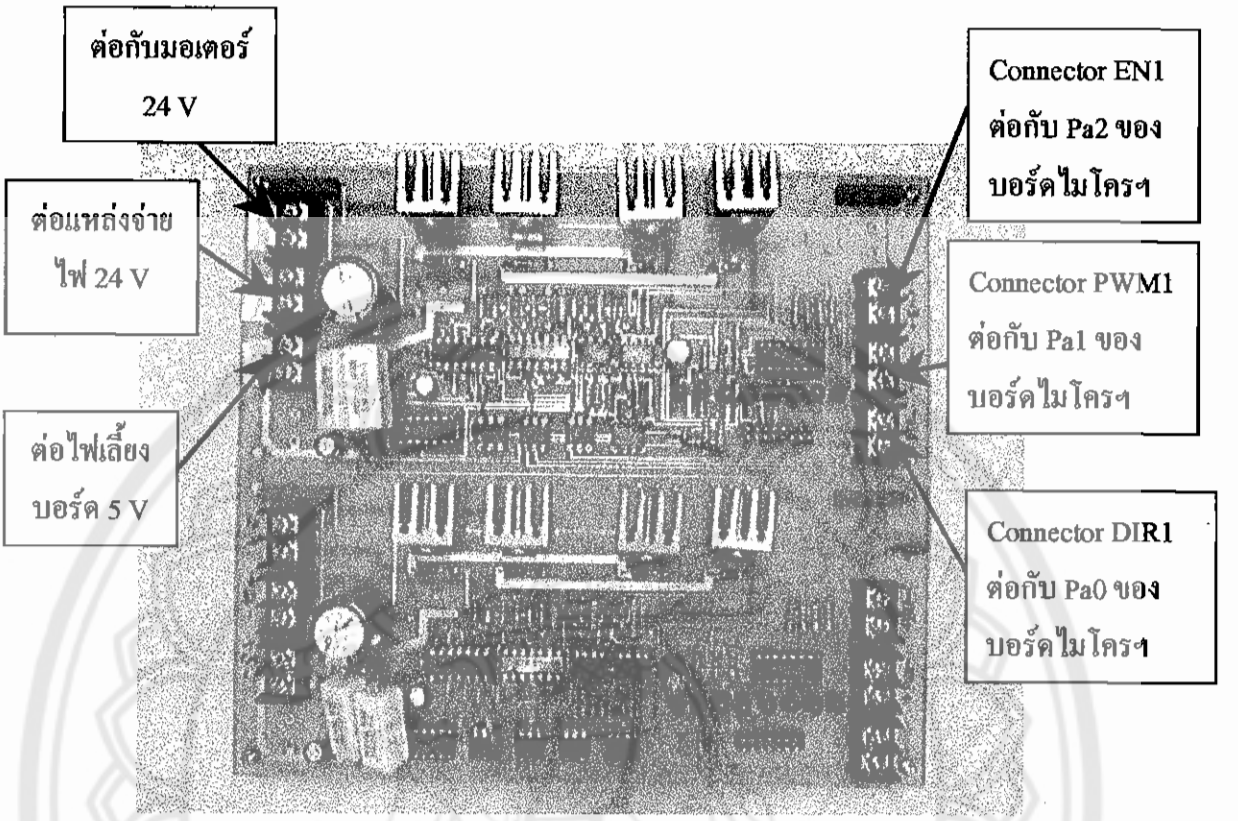
```
  }
if(((Pb>=0x00)&&(Pb<=0x0F))&&((Pc>=0x70)&&(Pc<=0x80))) // 24 front3+R2
  {
  duty2=25;
  duty1=100;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(1);
  drive_dir2(1);
  }
if(((Pb>=0x70)&&(Pb<=0x80))&&((Pc>=0x70)&&(Pc<=0x80))) // 25 back+R2
  {
  duty2=25;
  duty1=90;
  drive_en1(0);
  drive_en2(0);
  drive_dir1(0);
  drive_dir2(0);
  }

 }//end while(1)
}//end main
```

3.4 ศึกษาการใช้ไดรเวอร์ในการขับมอเตอร์

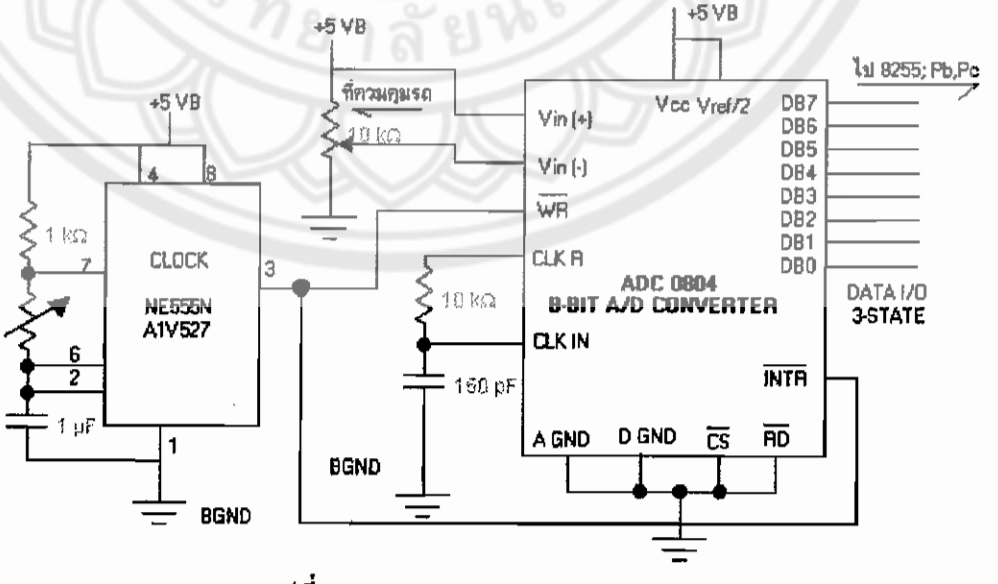


รูปที่ 3.4 วงจรไดรเวอร์



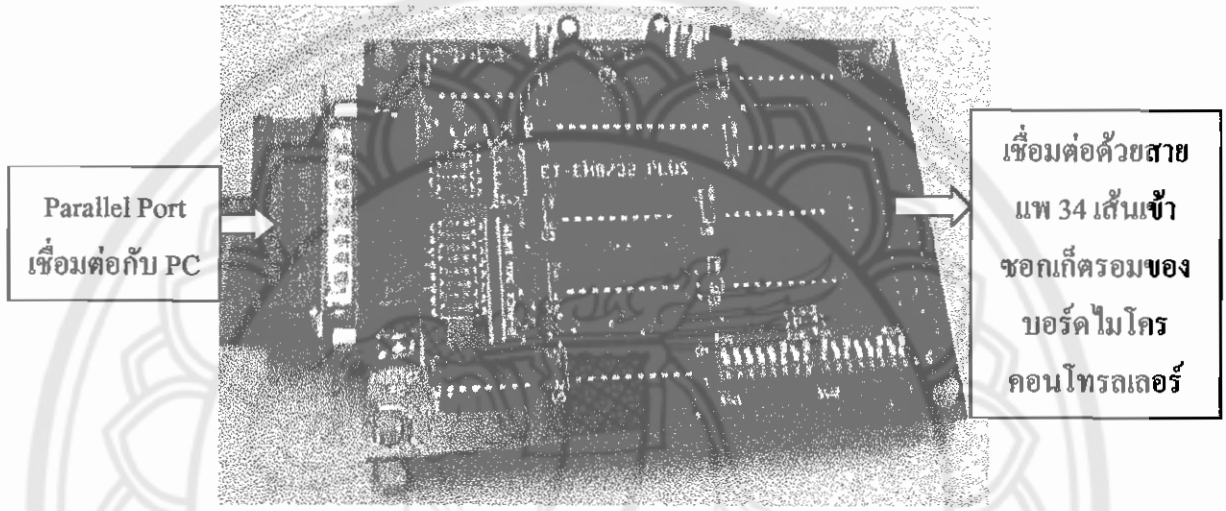
รูปที่ 3.5 บอร์ดไดรเวอร์

3.5 ศึกษาวงจร A/D เพื่อนำมาใช้ในการสั่งงานบอร์ดไมโครคอนโทรลเลอร์



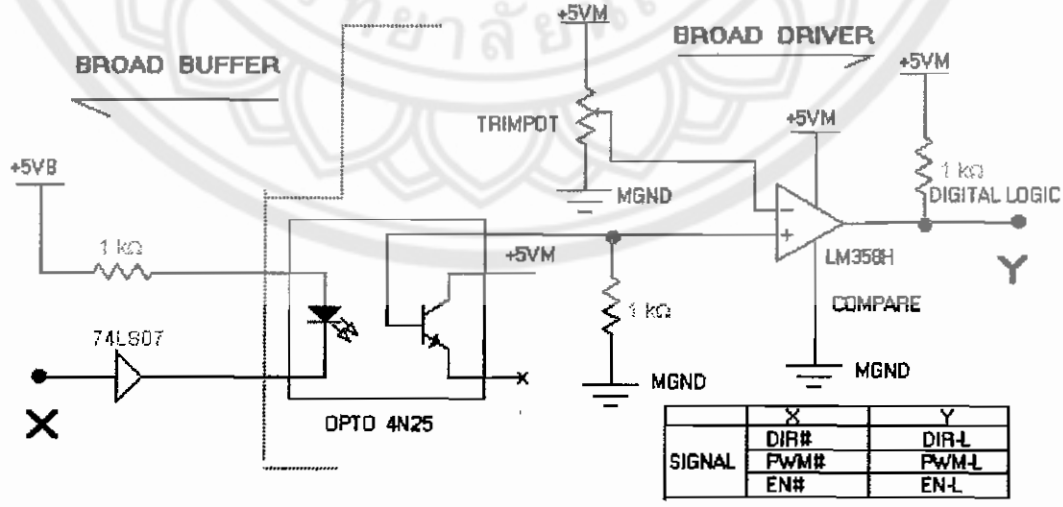
รูปที่ 3.6 วงจร A/D Converter

3.6 ศึกษาเกี่ยวกับรอมอีมูเลเตอร์ (ROM Emulator) เพื่อเชื่อมต่อกับ PC ในการเขียนโปรแกรม



รูปที่ 3.7 บอร์ดรอมอีมูเลเตอร์

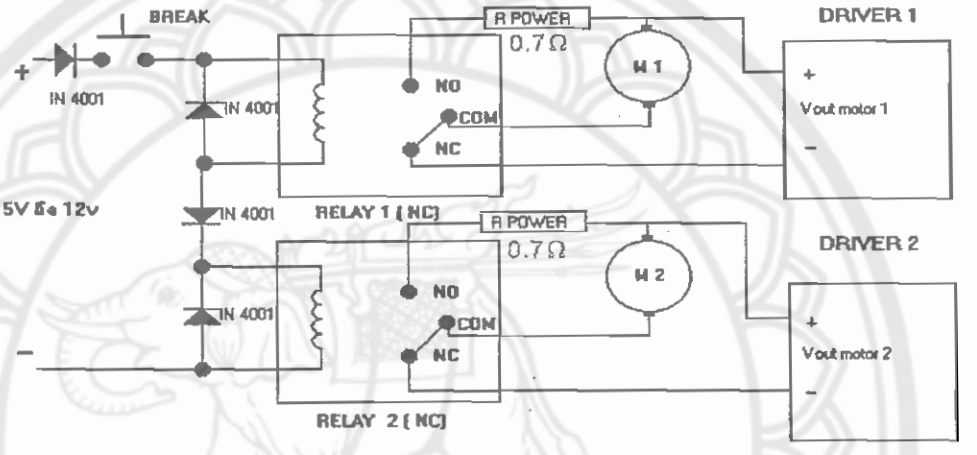
3.7 วงจรบัฟเฟอร์ และการใช้งาน



รูปที่ 3.8 วงจรบัฟเฟอร์

การทำงานของวงจรบัฟเฟอร์ คือ เมื่อมีสัญญาณเอาท์พุตออกจากไมโครคอนโทรลเลอร์ โดยการส่งผ่านสายแพ หรือสายไฟที่จุด X โดยในสายดังกล่าวจะมีค่าความต้านทานในสายทำให้แรงดันตกลง อาจทำให้เกิดความผิดพลาดในการส่งข้อมูลได้ ดังนั้นเราจึงนำวงจรบัฟเฟอร์มาใช้เพื่อยกระดับแรงดันสัญญาณที่อ่อนให้แรงขึ้น เพื่อแก้ปัญหาการส่งข้อมูลผิดพลาด

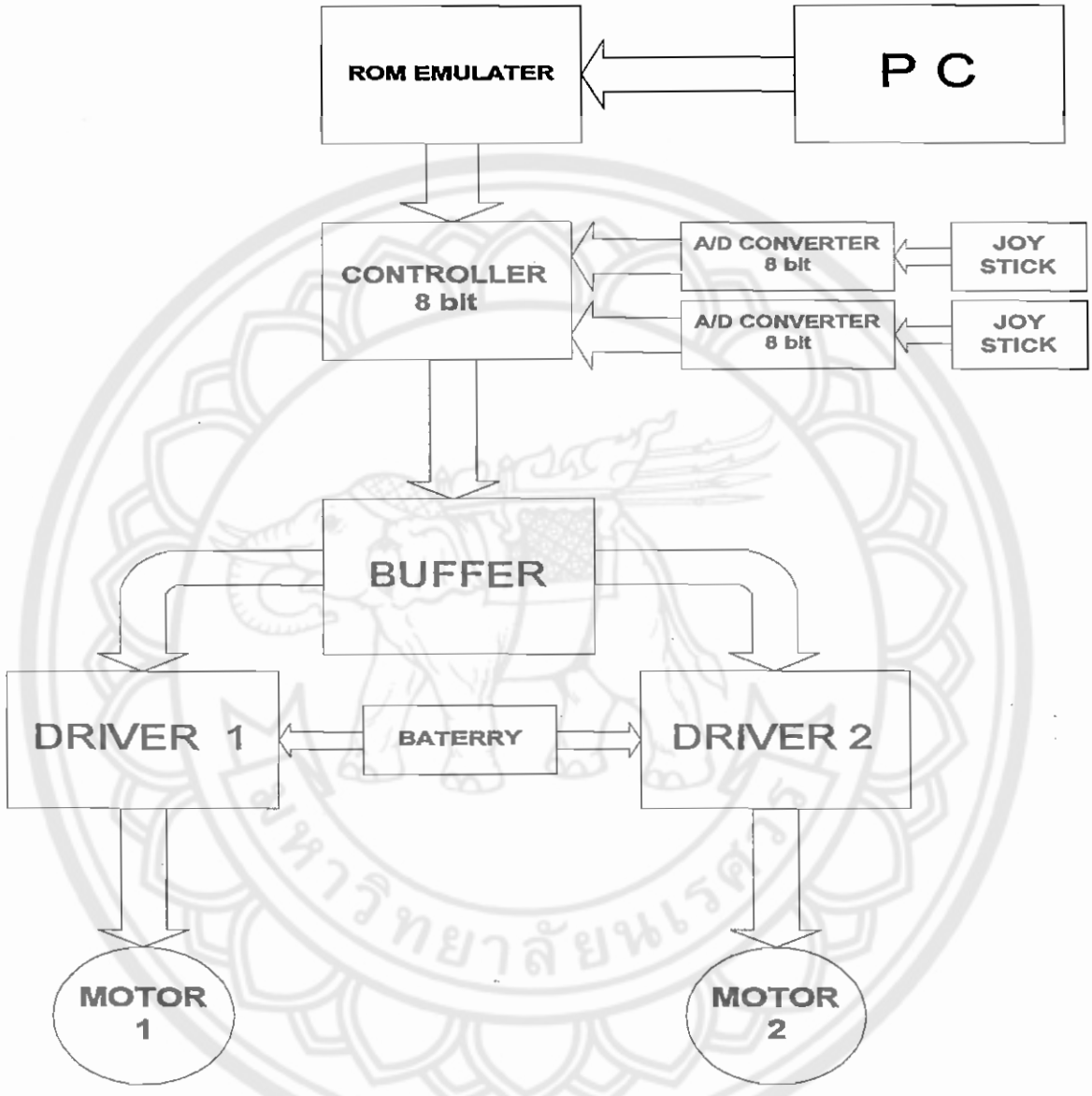
3.8 ระบบเบรค



รูปที่ 3.9 วงจรเบรค

ระบบเบรคที่เราใช้เป็นระบบเบรคทางไฟฟ้าซึ่งเป็นการสั่งงานด้วยการกดสวิทช์เบรคซึ่งจะอยู่ที่ชุดปุ่มควบคุมทางด้านซ้ายมือ โดยใช้หลักการการจ่ายกระแสคืนของมอเตอร์ขณะมีแรงเฉื่อยเมื่อหยุดการจ่ายไฟ โดยจะนำกระแสส่วนนี้มาลัดวงจรใส่ตัวต้านทานค่า 0.7 โอห์ม ซึ่งจะทำให้เกิดการหยุดหมุนของมอเตอร์อย่างรวดเร็วกินเวลาเพียงเสี้ยววินาที และเมื่อมีการกดสวิทช์เบรคจะเป็นการตัดวงจรควบคุมการขับเคลื่อนด้วย เพื่อป้องกันความเสียหายของวงจรควบคุมเนื่องจากกระแสกระชากของการเบรค

3.9 แผนภาพระบบโดยรวมของรถไฟฟ้าคนพิการ



รูปที่ 3.10 บล๊อกไดอะแกรมของรถคนพิการไฟฟ้า

บล๊อกไดอะแกรมของการทำงานเริ่มจาก ไมโครคอนโทรลเลอร์รับอินพุตจากจอยสติ๊กเข้ามาเพื่อประมวลผลตามโปรแกรมที่เขียนไว้ แล้วส่งออกเอาท์พุตเพื่อขับบอร์ดไดรเวอร์ เพื่อสั่งให้มอเตอร์ในระบบการเคลื่อนที่ของตัวรถทำงาน

3.10 มอเตอร์

มอเตอร์ที่ใช้เป็นมอเตอร์กระแสตรง ชนิดแม่เหล็กถาวร มีพิกัดต่าง ๆ ดังต่อไปนี้

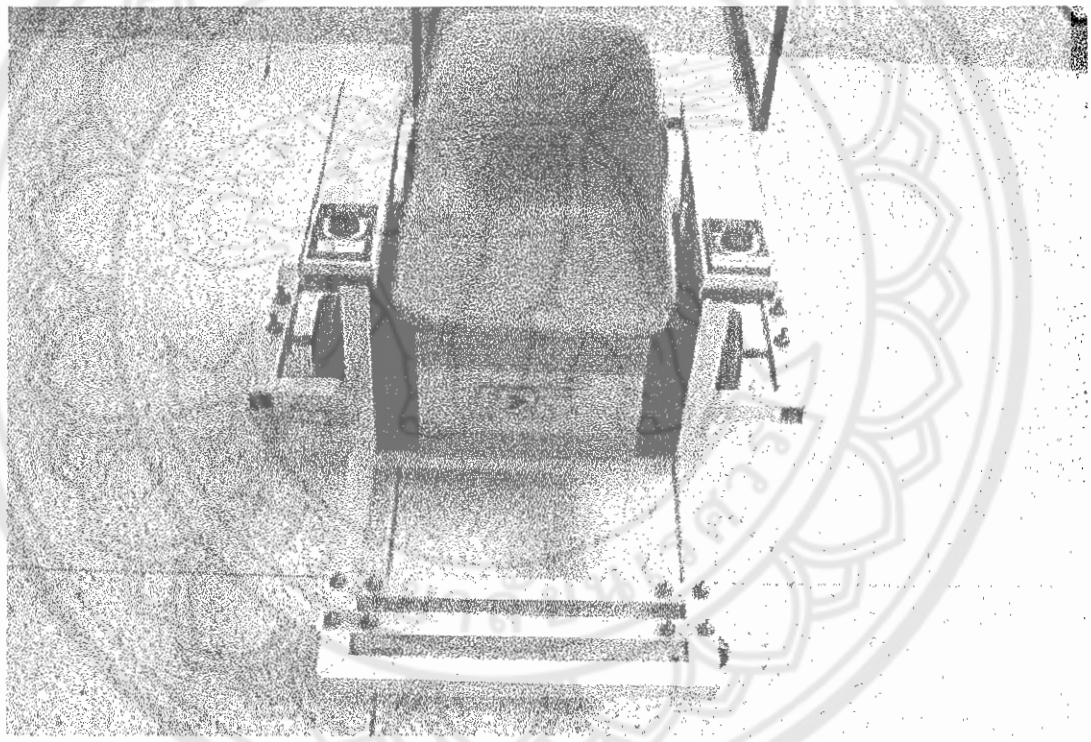
3.9.1 ความเร็วรอบสูงสุดที่ไร้โหลด 175 rpm และมีการทดรอบในตัวมอเตอร์

3.9.2 แรงดันพิกัดสูงสุด 24 โวลท์ กระแสตรง

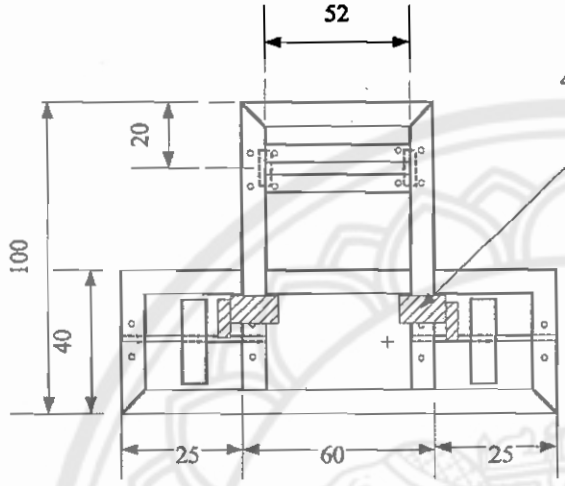
3.9.3 กินกระแสสูงสุดขณะเริ่มเดิน 8 แอมแปร์

3.9.4 กินกระแสเฉลี่ยที่ความเร็วรอบสูงสุด 2 แอมแปร์

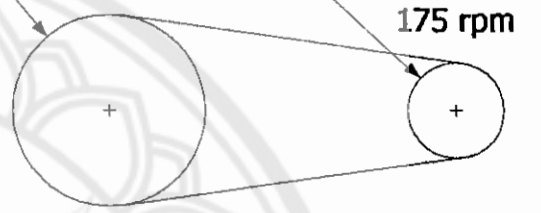
3.11 โครงสร้างรถไฟฟ้าคนพิการ และทิศทางการขับเคลื่อน



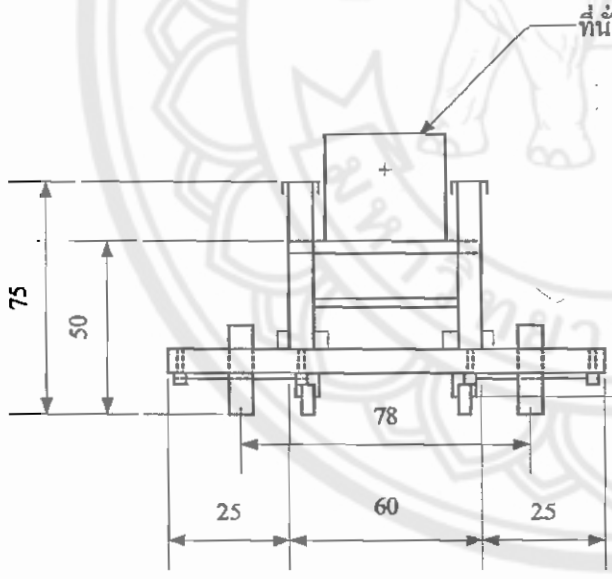
รูปที่ 3.11 โครงสร้างตัวรถคนพิการไฟฟ้า



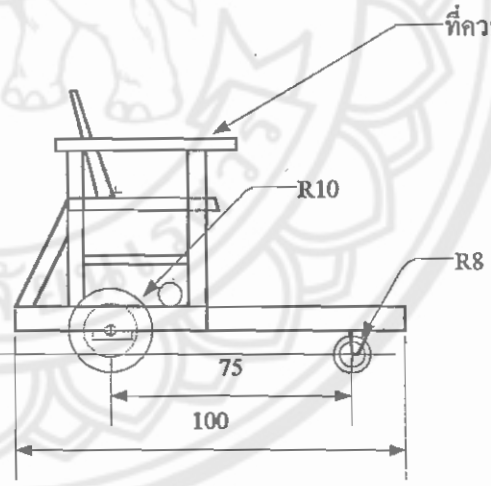
รูปที่ 3.12(ก) ด้านบน



รูปที่ 3.12(ข) ระบบเกียร์อัตราทด 2.353 : 1

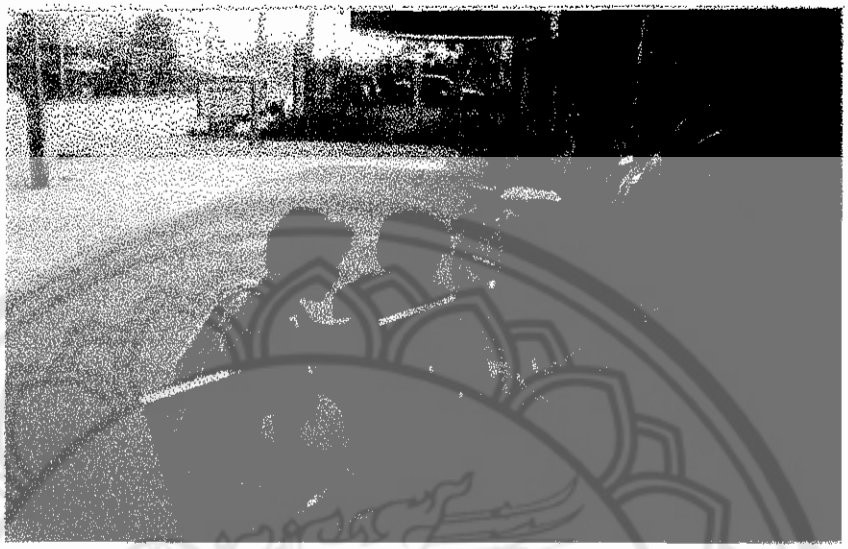


รูปที่ 3.12(ค) ด้านหน้า

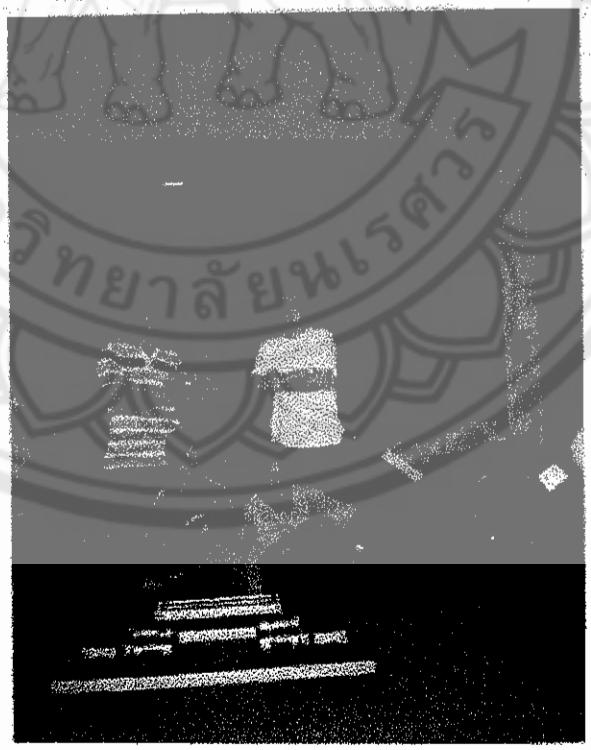


รูปที่ 3.12(ง) ด้านข้าง

รูปที่ 3.12 โครงสร้างตัวรถทางเทคนิค (หน่วยความยาวเป็นเซนติเมตร)



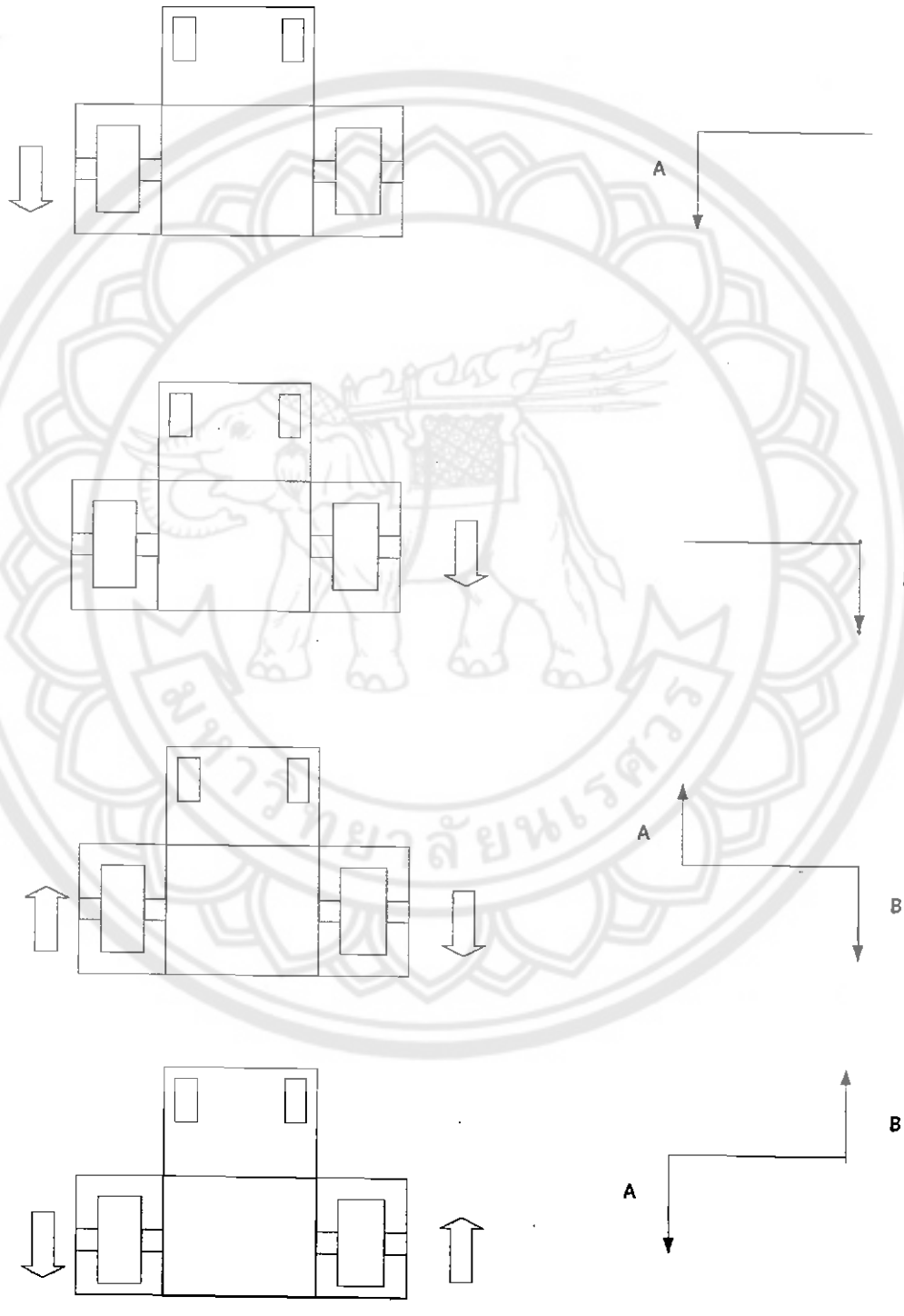
รูปที่ 3.13 การประกอบโครงสร้างตัวรถ

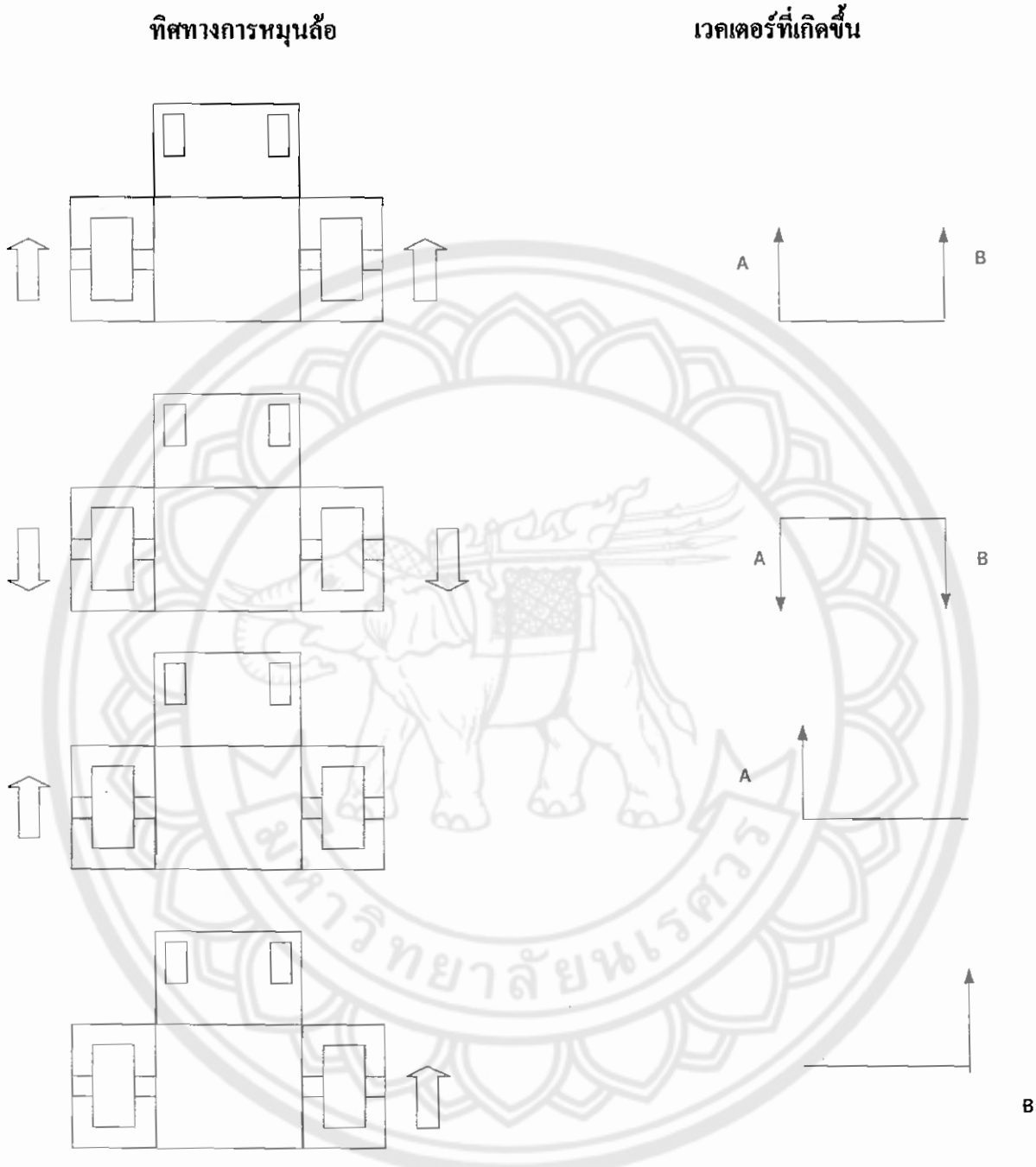


รูปที่ 3.14 ช่วงล่างของรถไฟฟ้าและคณะผู้จัดทำ

<u>การเคลื่อนที่ และการเลี้ยวในทิศทางต่าง ๆ</u>

ทิศทางการหมุนล้อ **เวคเตอร์ที่เกิดขึ้น**





รูปที่ 3.15 การเคลื่อนที่และการเลี้ยวในทิศทางต่าง ๆ