บรรณานุกรม

[1] Iris F.A. Vis, "Survey of research in the design and control of automated guided vehicle systems", European Journal of Operational Research 170, pp. 677-709, 2006.

[2] Ronald J. Mantel and Henri R.A. Landeweerd, "Design and operational control of an AGV system", International Journal of Production Economics 41, pp 257-266, 1995.

[3] Kyung Sup Kim and Byung Do Chung, "Design for a tandem AGV system with two-load AGVs", Computers & Industrial Engineering 53, pp 247-251, 2007.

[4] Wooyeon Yu and Pius J. Egbelu, "Design of a Variable Path Tandem Layout for Automated Guided Vehicle Systems", Journal of Manufacturing Systems, Vol. 20/No. 5, 2001.

[5] Götting, H.H., 2000. Automation and Steering of Vehicles in ports. Port Technology International 10, 101–111.

[6] Egbelu, P.J., "Concurrent specification of unit load sizes and automated guided vehicle fleet size in manufacturing system", International Journal of Production Economics 29, 49–64, 1993.

[7] Moon, S.W., Hwang, H., "Determination of unit load sizes of AGV in multi-product multi-line assembly production systems", International Journal of Production Research 37 (15), 3565–3581, 1999.

[8] Ganesharajah, T., Sriskandarajah, C., "Survey of scheduling research in AGV-served manufacturing systems", Advances in Instrumentation and Control, Proceedings of the International Conference and Exhibit 50(1), 87–94, 1995.

[9] King, R.E., Wilson, C., "A review of automated guided vehicle systems design and scheduling", Production Planning and Control 2 (1), 44–51, 1991.

ภาคผนวก

โปรแกรมควบคุมการทำงาน

```
#include<16F877A.h>
#fuses hs,nowdt,noprotect,nolvp
#use delay(clock=8000000)
#use rs232(baud=9600,xmit=PIN_C6,rcv=PIN_C7)

#define sensor_right  pin_A2
#define sensor_center pin_A1
#define sensor_left   pin_A0
#define sensor_bar    pin_A3
#define led1 PIN_E0
#define led2 PIN_E1
#define led3 PIN_E2
#define sw_1 PIN_D7
#define sw_2 PIN_D6
#define sw_3 PIN_D5
#define sw_4 PIN_D4
#define motor_R_EN pin_B4
#define motor_L_D  pin_B5
#define motor_R_D  pin_B6
#define motor_L_EN pin_B7
#define motor_L_C  pin_C1
#define motor_R_C  pin_C2
#define min_pwm 0
#define max_pwm 255

int16 seconds = 0;
int int_count=0;
int tts=0;
#INT_TIMER0
void clock_isr()
{
    if(int_count++>38)
 {
      ++seconds;
      int_count = 0;
   }
}
```

```
void forward(unsigned char speedR,unsigned char speedL);
void right(unsigned char speedR,unsigned char speedL);
void left(unsigned char speedR,unsigned char speedL);
void backward(unsigned char speedR,unsigned char speedL);
void stop(void);

/************************************
* Forward
************************************/
void forward(unsigned char speedR,unsigned char speedL)
{
   set_pwm1_duty(min_pwm);
   output_bit(motor_R_EN,1);
   output_bit(motor_R_C,1);
   output_bit(motor_R_D,0);

   set_pwm2_duty(min_pwm);
   output_bit(motor_L_EN,1);
   output_bit(motor_L_C,1);
   output_bit(motor_L_D,0);

   set_pwm1_duty(speedR);
   set_pwm2_duty(speedL);
}
/*********************************
* Stop
*********************************/
void stop(void)
{
   set_pwm1_duty(min_pwm);
   output_bit(motor_R_EN,0);
   output_bit(motor_R_C,0);
   output_bit(motor_R_D,0);

   set_pwm2_duty(min_pwm);
   output_bit(motor_L_EN,0);
```

```
   output_bit(motor_L_C,0);
   output_bit(motor_L_D,0);
   set_pwm1_duty(0);
   set_pwm2_duty(0);
}
/*********************************
* main program
*********************************/
void main(void) {
   int16 start=0;
   int   get_line=0;
   int   x=0;
   int check=0;
   int y=0 ;

   enable_interrupts(GLOBAL);
   enable_interrupts(INT_TIMER0);

   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
   set_timer0(0);

   set_tris_A(0b11111111);
   set_tris_B(0b00000000);
   set_tris_C(0b00000000);
   Set_tris_E(0b00000000);
   Set_tris_D(0b11111111);

   setup_timer_2(T2_DIV_BY_16,255,1);
   setup_ccp1(ccp_pwm);
   setup_ccp2(ccp_pwm);

   stop();
   while(TRUE){
      while(x==0){
         if(input(sw_1)==0)
         {
            x=1;
```

```
        if (x==y)
        {
        x=0;
         }

        output_bit(led1,1);
        output_bit(led2,0);
        output_bit(led3,0);
    }
    else if(input(sw_2)==0){
        x=2;
        if (x==y)
        {
        x=0;
         }
        output_bit(led1,0);
        output_bit(led2,1);
        output_bit(led3,0);
    }
    else if(input(sw_3)==0){
        x=3;
        if (x==y)
        {
        x=0;
         }
        output_bit(led1,0);
        output_bit(led2,0);
        output_bit(led3,1);
    }
    else if(input(sw_4)==0){
        x=4;
        if (x==y)
        {
        x=0;
         }
        output_bit(led1,1);
        output_bit(led2,1);
```

```
        output_bit(led3,1);
    }
    else
    {
        x=0;
    }
}

while (TRUE) {
    if(start!=0)
    {
        if(seconds-start>1)
        {
if(x==get_line)
          {
  stop();
            y= get_line;
            get_line=0;
            start=0;
            x=0;
            break;
}
          get_line=0;
          start=0;
        }
    }

    if(input(sensor_bar)==1)
    {
        if(check==0)
        {
          start=seconds;
      get_line++;
          check=1;
        }
     }
    if(input(sensor_bar)!=1)
```

```
          {
                check=0;
          }

if((input(sensor_center)==1)&&(input(sensor_left)==0)&&(input(
sensor_right)==0))
          {
             forward(255,255);
             delay_ms(5);
          }
         else
if((input(sensor_left)==1)&&(input(sensor_right)==0))
          {
             forward(200,120);
             delay_ms(5);
          }

          else
if((input(sensor_left)==0)&&(input(sensor_right)==1))
          {
             forward(120,200);
             delay_ms(5);
          }
          else
          {
             forward(75,75);
             delay_ms(5);
          }
       }
    }
}
```