

소스코드

AVR

※ 이동형 로봇팔

* DcPid.c

```
1#include <iom128.h>
2#include <ina90.h>
3#include "Header/DcPid.h"
4#include "Header/i2c_k.h"
5#include "Header/dynamixel_k.h"
6unsigned char PortCtrl = 0x00;
7unsigned int SecFlag = 0, tim_count = 0;
8unsigned int LeftObjectSpeed, RightObjectSpeed, LeftPresentSpeed, RightPresentSpeed;
9unsigned int DutyA, DutyB;
10unsigned int LeftEncoder = 0, RightEncoder = 0;
11int RightOutput, LeftOutput, LeftBeforeError, RightBeforeError;
12int LeftErrorSum = 0, RightErrorSum = 0;
13int LeftError, RightError;
14void StartDc(void)
15{
16    if(twi_ch[2] == 1)//drive mode
17    {
18        BasicRobotArm();
19        PortCtrl = 0x00;
20        if(twi_ch[0] > 5)
21        {
22            LeftObjectSpeed = twi_ch[0] - 2;
23            PortCtrl |= 0x01;
24        }
25        else if (twi_ch[0] < 5)
26        {
27            LeftObjectSpeed = 8 - twi_ch[0];
28            PortCtrl |= 0x02;
29        }
30        else
31        {
32            LeftObjectSpeed = 0;
33            PortCtrl |= 0x01;
34        }
35
36        if(twi_ch[1] > 5)
37        {
38            RightObjectSpeed = twi_ch[1] - 2;
```

```

39         PortCtrl |= 0x04;
40     }
41     else if(twi_ch[1] < 5)
42     {
43         RightObjectSpeed = 8 - twi_ch[1];
44         PortCtrl |= 0x08;
45     }
46     else
47     {
48         RightObjectSpeed = 0;
49         PortCtrl |= 0x04;
50     }
51 }
52 else // mission perform mode
53 {
54     twi_ch[0] = 5;
55     twi_ch[1] = 5;
56     PortCtrl = 0x00;
57 }
58
59 if(SecFlag == 1)
60 {
61     __disable_interrupt();
62     LeftPresentSpeed = (LeftEncoder*312)/1000;
63     LeftError = LeftObjectSpeed - LeftPresentSpeed;
64
65     if(LeftError > 0)
66     {
67         /*LeftOutput = (58*LeftError)/100 + (500*(LeftError-LeftBeforeError))/100 + (10*Left
68         LeftOutput = (L_K_P*LeftError + L_K_D*(LeftError-LeftBeforeError)
69 + L_K_I*LeftErrorSum)/SCALING_FACTOR;
70         DutyA += LeftOutput;
71         if(DutyA > 1000)
72         {
73             DutyA = 1000;
74         }
75     else if( LeftError < 0)
76     {
77         DutyA -= 5;
78         if(DutyA < 1)
79         {
80             DutyA = 0;

```

```

81     RightPresentSpeed = (RightEncoder*312)/1000;
82     RightError = RightObjectSpeed - RightPresentSpeed;
83
84     if(RightError > 0)
85     {
86         /*RightOutput = (65*RightError)/100 + (500*(RightError-RightBeforeError))/100 + (125
87         RightOutput = (R_K_P*RightError + R_K_D*(RightError-RightBeforeError)
88 + R_K_I*RightErrorSum)/SCALING_FACTOR ;
89         DutyB += RightOutput;
90         if(DutyB > 1000)
91             DutyB = 1000;
92
93     }
94     else if( RightError < 0)
95     {
96         DutyB -= 5;
97         if(DutyB < 1)
98             DutyB = 0;
99     }
100    if(twi_ch[0] == 5)
101    {
102        LeftError = 0;
103        LeftErrorSum = 0;
104        DutyA = 0;
105    }
106    if(twi_ch[1] == 5)
107    {
108        RightError = 0;
109        RightErrorSum = 0;
110        DutyB = 0;
111    }
112    OCR1A = DutyA;
113    OCR1B = DutyB;
114    PORTA = PortCtrl;
115
116    LeftBeforeError = LeftError;
117    RightBeforeError = RightError;
118
119    LeftErrorSum += LeftError;
120    RightErrorSum += RightError;
121
122    LeftEncoder = 0;

```

```

123     RightEncoder = 0;
124
125     SecFlag = 0;
126     __enable_interrupt();
127 }
128 }
129 void Tim1PwmInit(void)
130 {
131     TCCR1A = 0x00;
132     TCCR1B = 0x00;
133     TCCR1A |= (1<<WGM10) | (1<<WGM11) | (1<<COM1A1) | (1<<COM1B1) ;
134 //10bit, TOP = 0x3ff, fast PWM & inverting PWM
135     TCCR1B |= (1<<WGM12) | (1<<CS12) | (1<<CS10);
136 //prescaler 1024
137 }
138 void ExtInterruptInit(void)
139 {
140     EIMSK = 0x00;
141     EICRA = 0x00;
142     EICRB = 0x00;
143     EIFR = 0x00;
144
145     EIMSK |= (1<<INT4) | (1<<INT6); //INT4,6
146     EICRB |= (1<<ISC41) | (1<<ISC61) | (1<<ISC40) | (1<<ISC60);
147 //rising edge interrupt
148     EIFR |= (1<<INTF4) | (1<<INTF6);
149 }
150 void Tim3OvfInit(void)
151 {
152     TCCR3A = 0x00;
153     TCCR3B = 0x00;
154     TCCR3C = 0x00;
155
156     ETIMSK = 0x00;
157     ETIFR = 0x00;
158
159     TCCR3B |= (1<<CS32); //prescaler 256
160     TCNT3 = 0xffff-625; //65535 - 625
161
162     ETIMSK |= (1<<TOIE3); //timer3 overflow interrupt
163     ETIFR |= (1<<TOV3);
164 }

```

```

165 #pragma vector = INT4_vect
166 __interrupt void INT4_interrupt(void)
167 {
168     __disable_interrupt();
169     LeftEncoder++;
170     __enable_interrupt();
171 }
172 #pragma vector = INT6_vect
173 __interrupt void INT6_interrupt(void)
174 {
175     __disable_interrupt();
176     RightEncoder++;
177     __enable_interrupt();
178 }
179 #pragma vector=TIMER3_OVF_vect
180 __interrupt void TIMER3_OVF_interrupt(void) // 50ms
181 {
182     __disable_interrupt();
183     TCNT3 = 0xffff-625;
184     tim_count++;
185     if(tim_count == 5)
186     {
187         SecFlag = 1;
188         tim_count = 0;
189     }
190     __enable_interrupt();
191 }

```

* dynamixel_k.c

```

1#include <iom128.h>
2#include <ina90.h>
3#include "Header/dynamixel_k.h"
4#include "Header/uart_k.h"
5#include "Header/i2c_k.h"
6void StartRobotArm(void)
7{
8    unsigned int i;
9    if(twi_ch[2] == 2)

```

```

10     {
11         trans0_ch('S');
12
13         for(i=0; i<13; i++)
14             uart_arr[i]=receive0_ch();
15
16         UartCheckSum = uart_check_sum(uart_arr,13);
17
18         if( uart_arr[12] == UartCheckSum )
19             {
20                 arm_control_ang_ve(start_arm);
21                 check_sum(start_arm,start_arm_length);
22                 for(i=0; i<start_arm_length; i++)
23                     trans1_ch( start_arm[i] );
24             }
25     }
26 }
27 void BasicRobotArm(void)
28 {
29     unsigned int i;
30     check_sum(basic_arr,start_arm_length);
31     for(i=0; i<start_arm_length; i++)
32         trans1_ch( basic_arr[i] );
33 }
34 void dec_hex_convert(int dec, unsigned char* low_temp, unsigned char* high_temp )
35 {
36     *high_temp = 0x00;
37     *low_temp = 0x00;
38
39     *high_temp = (unsigned char)0xff&(dec/255);
40     *low_temp = (unsigned char)0xff&(dec%255);
41 }
42 void check_sum(unsigned char * check_arr,int arr_length)//디지털모터의
43 //checksum계산하는 함수
44 {
45     int i=0;
46     unsigned char data=0;
47     unsigned char check=0;
48
49     for(i=2;i<(arr_length-1);i++)//check sum을 구하기 위한 for문
50         data += check_arr[i];
51

```

```

52     check = ~data;
53     *(check_arr+(arr_length-1))=check;
54 }
55 unsigned char start_arm[33] = {
56     //각도 인덱스 (8,9) (13,14)(18,19) (23,24)
57     //속도 인덱스 (10,11) (15,16) (20,21) (25,26)
58     0xff,
59     0xff,
60     0xfe,
61     0x1d,//((L+1)*n)+4,n은 모터개수
62     0x83,//sync write 명령어
63
64     0x1e,//control table의 명령어
65     0x04,//쓰고자하는 L의 길이
66
67     0x11,//아이디 17
68     0x00,//L의 시작
69     0x00,
70     0x00,
71     0x00,//L의 끝
72
73     0x03,//아이디 3
74     0x00,
75     0x00,
76     0x00,
77     0x00,
78
79     0x06,//아이디 6
80     0x00,
81     0x00,
82     0x00,
83     0x00,
84
85     0x07,//아이디 7
86     0x00,
87     0x00,
88     0x00,
89     0x00,
90
91     0x13,//아이디 19
92     0x00,
93     0x00,

```

```

94     0x00,
95     0x00,
96     //check sum = ~(ID + Length + Instruction + parameter1 + ~ + Parameter N)
97     0x00
98 };
99 unsigned char basic_arr[33] = {
100     //각도 인덱스 (8,9) (13,14)(18,19) (23,24)
101     //속도 인덱스 (10,11) (15,16) (20,21) (25,26)
102     0xff,
103     0xff,
104     0xfe,
105     0x1d,/// $((L+1)*n)+4$ ,n은 모터개수
106     0x83,///sync write 명령어
107
108     0x1e,///control table의 명령어
109     0x04,///쓰고자하는 L의 길이
110
111     0x11,///아이디 17
112     0xff,///L의 시작
113     0x01,
114     0x65,
115     0x00,///L의 끝
116
117     0x03,///아이디 3
118     0xff,
119     0x01,
120     0x65,
121     0x00,
122
123     0x06,///아이디 6
124     0xff,
125     0x01,
126     0x65,
127     0x00,
128
129     0x07,///아이디 7
130     0xff,
131     0x01,
132     0x65,
133     0x00,
134
135     0x13,///아이디 19

```

```

136     0xff,
137     0x01,
138     0x65,
139     0x00,
140     //check sum = ~(ID + Length + Instruction + parameter1 + ~ + Parameter N)
141     0x00
142 };
143 unsigned int start_arm_length = (sizeof(start_arm)/sizeof(unsigned char));

```

※ 모형 로봇 팔

* adc_k.c

```

1#include <iom128.h>
2#include <ina90.h>
3#include "Header/adc_k.h"
4#include "Header/delay_k.h"
5unsigned int adc_count = 2;
6unsigned int MuxCount = 0;
7unsigned char adc_arr[13] = {0xff,0xff,0,0,0,0,0,0,0,0,0,0,0xff};
8void StartArmModel(void)
9{
10    ADMUX = (1<<REFS0) | (MuxCount << MUX0);
11    delay_us(10);
12    ADCSRA |= (1<<ADSC);
13}
14void EndArmModel(void)
15{
16    MuxCount++;
17    if(MuxCount == 5)
18    {
19        MuxCount = 0;
20        adc_count = 2;
21    }
22}
23void KongAdcInit(void)
24{
25    ADMUX = 0x00;
26    ADCSRA = 0x00;
27    ADCSRA |= (1<<ADEN) | (1<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0) ;
28}
29#pragma vector = ADC_vect

```

```

30  __interrupt void ADC_conversion_done(void)
31  {
32      __disable_interrupt();
33
34      adc_arr[adc_count] = ADCL;
35      adc_arr[adc_count+1] = ADCH;
36      adc_count += 2;
37      __enable_interrupt();
38  }

```

* eeprom.c

```

1  #include <iom128.h>
2  #include <ina90.h>
3  #include "Header/eeprom.h"
4  unsigned int eeprom_write_i=11;
5  unsigned int eeprom_read_i=11;
6  unsigned char EEPROM_read(unsigned int uiAddress)
7  {
8      while(EECR & (1<<EWE)); /* Wait for completion of previous write */
9      EEAR = uiAddress; /* Set up address register */
10     EECR |= (1<<EERE); /* Start eeprom read by writing EERE */
11     return EEDR; /* Return data from data register */
12 }
13 void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
14 {
15     while(EECR & (1<<EWE)); /* Wait for completion of previous write */
16     EEAR = uiAddress; /* Set up address and data registers */
17     EEDR = ucData;
18     EECR |= (1<<EEMWE); /* Write logical one to EEMWE */
19     EECR |= (1<<EWE); /* Start eeprom write by setting EWE */
20 }

```

4.2.2 Linux

* RobotArm.c

```

1 /*
2  *-----*
3  * FILE : RobotArm.c *
4  * AUTH : Kong, Young Hoon *
5  *-----*
6  */
7 #include <stdio.h>

```

```

8  #include <stdlib.h>
9  #include <errno.h>
10 #include <unistd.h>
11 #include <string.h>
12 #include <fcntl.h>
13 #include <sys/mman.h>
14 #include <sys/types.h>
15 #include <sys/ioctl.h>
16 #include <math.h>
17 #include <linux/videodev2.h>
18 #include <arpa/inet.h>
19 #include <sys/types.h>
20 #include <sys/socket.h>
21 #include <pthread.h>
22 #include <wiringPi.h>
23 #include <wiringPiI2C.h>
24 #include "RobotArm.h"
25 #define DEBUG 1
26 void YUVtoRGB(unsigned char *ImageBuffer, unsigned char *Y,
27 unsigned char *Cb, unsigned char *Cr);
28 void WebCamInit();
29 void tcp_init();
30 void udp_init();
31 void twi_init();
32 int main (void)
33 {
34     thstat_udpsrv = pthread_create(&thid_udpsrv, NULL , thread_udpsrv , NULL);
35     thstat_tcpsrv = pthread_create(&thid_tcpsrv, NULL, thread_tcpsrv, NULL);
36     #ifdef DEBUG
37     if((thstat_udpsrv < 0) || (thstat_tcpsrv < 0))
38     {
39         printf("Udpsrv or tcpsrv Thread Create Failure..\n");
40         exit(1);
41     }
42     #endif
43     thstat_udpsrv = pthread_join(thid_udpsrv, NULL);
44     thstat_tcpsrv = pthread_join(thid_tcpsrv, NULL);
45     #ifdef DEBUG
46     if((thstat_udpsrv != 0) || (thstat_tcpsrv != 0))
47     {
48         printf("Udpsrv or tcpsrv Thread join Failure..\n");
49         exit(1);

```

```

50     }
51     #endif
52     return 0;
53 }
54 void * thread_tcpsrv(void * arg)
55 {
56     int i,j,ret;
57     unsigned char * bmp_buffer;
58     unsigned char * image_buffer;
59     YUVBUFFERS yuv_buffer;
60
61     WebCamInit();
62     bmp_buffer = malloc(WIDTH * HEIGHT * DEPTH);
63     memset(bmp_buffer, 0, WIDTH*HEIGHT*DEPTH);
64
65     image_buffer = mmap( NULL, buf.length, PROT_READ | PROT_WRITE ,
66 MAP_SHARED, cam_fd, buf.m.offset);
67     #ifdef DEBUG
68     if(MAP_FAILED == image_buffer)
69     {
70         printf("memory map fail\n");
71         return 0;
72     }
73     #endif
74     tcp_init();
75     while(1)
76     {
77         ret = ioctl(cam_fd,VIDIOC_QBUF,&buf);
78         #ifdef DEBUG
79         if(ret<0)
80         {
81             printf("VIDIOC_QBUF fail\n");
82             exit(1);
83         }
84         #endif
85         ret = ioctl(cam_fd, VIDIOC_DQBUF,&buf);
86         #ifdef DEBUG
87         if(ret<0)
88         {
89             printf("Retrieving Frame\n");
90             exit(1);
91         }

```

```

92     #endif
93     for(i=0; i< (WIDTH*HEIGHT*2); i+=4)
94     {
95         yuv_buffer.y[i/2]    = image_buffer[i];
96         yuv_buffer.cb[i/4]   = image_buffer[i+1];
97         yuv_buffer.y[(i/2)+1] = image_buffer[i+2];
98         yuv_buffer.cr[i/4]   = image_buffer[i+3];
99     }
100     YUVtoRGB(bmp_buffer,yuv_buffer.y,yuv_buffer.cb, yuv_buffer.cr);
101     write(tcp_clnt_sock,bmp_buffer,BMPMAXBUFF);
102 }
103 munmap(image_buffer,buf.length);
104 free(bmp_buffer);
105 close (cam_fd);
106 close(tcp_clnt_sock);
107 close(tcp_serv_sock);
108 }
109 void * thread_udpsrv(void * arg)
110 {
111     int strlen;
112     twi_init();
113     udp_init();
114     udp_clnt_adr_size=sizeof(udp_clnt_adr);
115     while(1)
116     {
117         strlen = recvfrom(udp_serv_sock,UdpTwiArr, 3, 0,
118 (struct sockaddr*)&udp_clnt_adr, &udp_clnt_adr_size);
119         UdpTwiFlag = strlen;
120         #ifdef DEBUG
121         if(UdpTwiFlag != 0)
122         {
123             printf("udp receive data : %d ,%d, %d \n", (int)UdpTwiArr[0],
124 (int)UdpTwiArr[1], (int)UdpTwiArr[2] );
125         }
126         wiringPiI2CWrite(twi_fd, UdpTwiArr[0]);
127         usleep(10);
128         wiringPiI2CWrite(twi_fd, UdpTwiArr[1]);
129         usleep(10);
130         wiringPiI2CWrite(twi_fd, UdpTwiArr[2]);
131         usleep(10);
132     #endif
133     }

```

```

134     close(udp_serv_sock);
135 }
136 void YUVtoRGB(unsigned char *ImageBuffer, unsigned char *Y, unsigned char *Cb,
137 unsigned char *Cr)
138 {
139     int i,j;
140     unsigned char R[WIDTH*HEIGHT], G[WIDTH*HEIGHT], B[WIDTH*HEIGHT];
141     for(i=0; i<WIDTH*HEIGHT; i++)
142     {
143         R[i] = CLIP( ( 298*(Y[i]-16) + 409*(Cb[i/2]-128) + 128 ) >> 8 );
144         G[i] = CLIP( ( 298*(Y[i]-16) - 100*(Cr[i/2]-128) -
145 208*(Cb[i/2]-128) + 128 ) >> 8 );
146         B[i] = CLIP( ( 298*(Y[i]-16) + 516*(Cr[i/2]-128)+ 128 ) >> 8 );
147     }
148     for(i=0, j=0; i<(WIDTH*HEIGHT*3); i+=3, j++) // combine rgb bmp 만들기 전
149     {
150         ImageBuffer[i] = R[j];
151         ImageBuffer[i+1] = G[j];
152         ImageBuffer[i+2] = B[j];
153     }
154 }
155 void WebCamInit()
156 {
157     int ret;
158     cam_fd = open("/dev/video0",O_RDWR);
159     #ifdef DEBUG
160     if( cam_fd < 0 )
161     {
162         printf("cam open fail\n");
163         exit(1);
164     }
165     #endif
166     ret = ioctl( cam_fd, VIDIOC_QUERYCAP, &cap );
167
168     #ifdef DEBUG
169     if( ret < 0 )
170     {
171         printf("vidioc_querycap fail\n");
172         exit(1);
173     }
174     #endif
175     memset(&fmt, 0, sizeof(fmt));

```

```

176     fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
177     fmt.fmt.pix.width = WIDTH;
178     fmt.fmt.pix.height = HEIGHT;
179     fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUVV;
180     fmt.fmt.pix.field = V4L2_FIELD_INTERLACED;
181     ret = ioctl( cam_fd, VIDIOC_S_FMT, &fmt);
182     #ifdef DEBUG
183     if(ret < 0)
184     {
185         printf("vidio_s_fmt fail\n");
186         exit(1);
187     }
188     #endif
189     memset( &req_buf, 0, sizeof(req_buf));
190     req_buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
191     req_buf.memory = V4L2_MEMORY_MMAP;
192     req_buf.count = 1;
193     ret=ioctl(cam_fd,VIDIOC_REQBUFS,&req_buf);
194     #ifdef DEBUG
195     if( ret < 0 )
196     {
197         printf("video capture or mmap fail or VIDIOC_REQBUFS fail\n");
198         exit(1);
199     }
200     #endif
201     memset(&buf, 0 ,sizeof(buf));
202     buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
203     buf.memory = V4L2_MEMORY_MMAP;
204     buf.index = 0;
205     ret = ioctl(cam_fd,VIDIOC_QUERYBUF,&buf);
206     #ifdef DEBUG
207     if( ret < 0 )
208     {
209         printf("querying buffer");
210         exit(1);
211     }
212     #endif
213     ret = ioctl ( cam_fd, VIDIOC_STREAMON,&buf.type);
214     #ifdef DEBUG
215     if( ret < 0 )
216     {
217         printf(" capture error");

```

```

218     exit(1);
219 }
220 printf("Cam init success\n");
221 #endif
222 }
223 void tcp_init()
224 {
225     int ret;
226     tcp_serv_sock=socket(PF_INET,SOCK_STREAM,0);
227     #ifdef DEBUG
228     if(tcp_serv_sock== -1)
229     {
230         printf("Socket Error...\n");
231         exit(1);
232     }
233     #endif
234     memset(&tcp_serv_adr,0,sizeof(tcp_serv_adr));
235     tcp_serv_adr.sin_family=AF_INET;
236     tcp_serv_adr.sin_addr.s_addr=htonl(INADDR_ANY);
237     tcp_serv_adr.sin_port=htons(PORT);
238     ret=bind(tcp_serv_sock,(struct sockaddr*)&tcp_serv_adr,sizeof(tcp_serv_adr));
239
240     #ifdef DEBUG
241     if(ret == -1)
242     {
243         printf("bind error");
244         exit(1);
245     }
246     #endif
247     ret=listen(tcp_serv_sock,5);
248     #ifdef DEBUG
249     if(ret== -1)
250     {
251         printf("listen error");
252         exit(1);
253     }
254     #endif
255     tcp_clnt_adr_size=sizeof(tcp_clnt_adr);
256     tcp_clnt_sock=accept(tcp_serv_sock,
257 (struct sockaddr*)&tcp_clnt_adr,&tcp_clnt_adr_size);
258     #ifdef DEBUG
259     if(tcp_clnt_sock == -1)

```

```

    {
        printf("accept error");
        exit(1);
260     }
261     printf("tcp server init success\n");
262     #endif
263 }
264 void udp_init()
265 {
266     int ret;
267     udp_serv_sock=socket(PF_INET, SOCK_DGRAM, 0);
268     #ifdef DEBUG
269     if(udp_serv_sock==-1)
270     {
271         printf("UDP socket creation error");
272         exit(1);
273     }
274     #endif
275     memset(&udp_serv_adr, 0, sizeof(udp_serv_adr));
276     udp_serv_adr.sin_family=AF_INET;
277     udp_serv_adr.sin_addr.s_addr=htonl(INADDR_ANY);
278     udp_serv_adr.sin_port=htons(PORT);
279
280     ret = bind(udp_serv_sock, (struct sockaddr*)&udp_serv_adr,
281 sizeof(udp_serv_adr));
282     #ifdef DEBUG
283     if(ret == -1)
284     {
285         printf("udp bind error");
286         exit(1);
287     }
288     printf("udp server init success\n");
289     #endif
290 }
291 void twi_init()
292 {
293     twi_fd = wiringPiI2CSetup(twi_slave_addr);
294     #ifdef DEBUG
295     if( twi_fd == -1 )
296     {
        printf("twi setup fail\n");
        exit(1);
    }

```

```

    }
    printf("twi init success\n");
    #endif
}

```

4.2.3 WinForm

```

1 using System;
2 using System.Collections.Generic;
3 using System.ComponentModel;
4 using System.Data;
5 using System.Drawing;
6 using System.Linq;
7 using System.Text;
8 using System.Windows.Forms;
9 using System.Net;
10 using System.Net.Sockets;
11 using System.Threading;
12 namespace RobotArm
13 {
14     public partial class Form1 : Form
15     {
16         byte mode_flag = 1;
17         byte[] BmpBuffer = new byte[230400];
18         int CamMaxBufSize = 230400;
19         uint cam_flag = 1;
20         uint wheel_flag = 0, sendto_flag = 0;
21         uint t1_flag = 0, t2_flag = 0;
22         string raspberry_ip = "192.168.10.12";
23         string pc_ip = "192.168.10.11";
24         Bitmap bmp = new Bitmap(width: 320, height: 240,
25 format: System.Drawing.Imaging.PixelFormat.Format24bppRgb);
26         Thread t1, t2;
27         public Form1()
28         {
29             InitializeComponent();
30         }
31         private void Form1_FormClosing(object sender, FormClosingEventArgs e)
32         {
33             if (MessageBox.Show("정말 종료하시겠습니까?", "질문",
34 MessageBoxButtons.YesNo) == DialogResult.No)
35             {
36                 e.Cancel = true;

```

```

37     }
38     else
39     {
40         if (t1_flag == 1)
41         {
42             t1.Abort();
43         }
44         if (t2_flag == 1)
45         {
46             t2.Abort();
47         }
48     }
49 }
50 private void button1_Click(object sender, EventArgs e)
51 {
52     t1 = new Thread(new ThreadStart(TcpCamThread));
53     t1.Start();
54     t1_flag = 1;
55 }
56 private void checkBox1_CheckedChanged(object sender, EventArgs e)
57 {
58     cam_flag = ~cam_flag;
59 }
60 private void TcpCamThread()
61 {
62     int length, byteLength = 0;
63     NetworkStream stream = null;
64     TcpClient client = null;
65     IPEndPoint clientAddress = null;
66     IPEndPoint serverAddress = null;
67     try
68     {
69         clientAddress = new IPEndPoint(address: IPAddress.Parse(pc_ip),
70 port: 10000);
71         serverAddress = new IPEndPoint(address: IPAddress.Parse(raspberry_ip),
72 port: 10000);
73         client = new TcpClient(clientAddress);
74         client.Connect(serverAddress);
75         stream = client.GetStream();
76         while (true)
77         {
78             if (cam_flag == 1)

```

```

79         {
80             while ((length = stream.Read(buffer: BmpBuffer,
81 offset: byteLength, size: CamMaxBufSize - byteLength)) != 0)
82             {
83                 byteLength += length;
84             }
85             if (byteLength == 230400)
86             {
87                 BmpInit();
88                 pictureBox1.Image = bmp;
89                 byteLength = 0;
90             }
91         }
92     }
93 }
94 catch (SocketException e)
95 {
96     Console.WriteLine(e);
97 }
98 finally
99 {
100     stream.Close();
101     client.Close();
102 }
103 }
104 private void BmpInit()
105 {
106     System.Drawing.Imaging.BitmapData bmpData =
107         bmp.LockBits(new Rectangle(0, 0, bmp.Width, bmp.Height),
108 System.Drawing.Imaging.ImageLockMode.ReadOnly, bmp.PixelFormat);
109     IntPtr ptr = bmpData.Scan0;
110     System.Runtime.InteropServices.Marshal.Copy(source: BmpBuffer,
111 startIndex: 0, destination: ptr, length: 320 * 240 * 3);
112     bmp.UnlockBits(bmpData);
113 }
114 private void button2_Click(object sender, EventArgs e)
115 {
116     t2 = new Thread(new ThreadStart(UdpModeWheelCtrl));
117     t2.Start();
118     t2_flag = 1;
119 }
120 private void UdpModeWheelCtrl()

```

```

121     {
122         byte[] wheel_mode_arr = new byte[3];
123         wheel_mode_arr[0] = 5;
124         wheel_mode_arr[1] = 5;
125         wheel_mode_arr[2] = 0;
126         Socket udpSocket = null;
127         EndPoint localEP = null;
128         EndPoint remoteEP = null;
129         try
130         {
131             udpSocket = new Socket(AddressFamily.InterNetwork, SocketType.Dgram,
132 ProtocolType.Udp);
133             localEP = new IPEndPoint(address: IPAddress.Any, port: 10000);
134             remoteEP = new IPEndPoint(IPAddress.Parse(raspberry_ip), port: 10000);
135             while (true)
136             {
137                 wheel_mode_arr[2] = mode_flag;
138                 if (sendto_flag != 1)
139                 {
140                     switch (wheel_flag)
141                     {
142                         case 1://전진
143                             wheel_mode_arr[0] += 1;
144                             wheel_mode_arr[1] += 1;
145                             if (wheel_mode_arr[0] > 8)
146                             {
147                                 wheel_mode_arr[0] = 8;
148                             }
149                             if (wheel_mode_arr[1] > 8)
150                             {
151                                 wheel_mode_arr[1] = 8;
152                             }
153                             udpSocket.SendTo(wheel_mode_arr, remoteEP);
154                             sendto_flag = 1;
155                             break;
156                         case 2://후진
157                             wheel_mode_arr[0] -= 1;
158                             wheel_mode_arr[1] -= 1;
159                             if (wheel_mode_arr[0] < 2)
160                             {
161                                 wheel_mode_arr[0] = 2;
162                             }

```

```

163         if (wheel_mode_arr[1] < 2)
164         {
165             wheel_mode_arr[1] = 2;
166         }
167         udpSocket.SendTo(wheel_mode_arr, remoteEP);
168         sendto_flag = 1;
169         break;
170     case 3://좌회전
171         wheel_mode_arr[0] -= 1;
172         wheel_mode_arr[1] += 1;
173         if (wheel_mode_arr[0] < 2)
174         {
175             wheel_mode_arr[0] = 2;
176         }
177         if (wheel_mode_arr[1] > 8)
178         {
179             wheel_mode_arr[1] = 8;
180         }
181         udpSocket.SendTo(wheel_mode_arr, remoteEP);
182         sendto_flag = 1;
183         break;
184     case 4://우회전
185         wheel_mode_arr[0] += 1;
186         wheel_mode_arr[1] -= 1;
187         if (wheel_mode_arr[0] > 8)
188         {
189             wheel_mode_arr[0] = 8;
190         }
191         if (wheel_mode_arr[1] < 2)
192         {
193             wheel_mode_arr[1] = 2;
194         }
195         udpSocket.SendTo(wheel_mode_arr, remoteEP);
196         sendto_flag = 1;
197         break;
198     case 5://정지
199         wheel_mode_arr[0] = 5;
200         wheel_mode_arr[1] = 5;
201         udpSocket.SendTo(wheel_mode_arr, remoteEP);
202         sendto_flag = 1;
203         break;
204     case 6://모드 변경

```

```

205         wheel_mode_arr[0] = 5;
206         wheel_mode_arr[1] = 5;
207         udpSocket.SendTo(wheel_mode_arr, remoteEP);
208         sendto_flag = 1;
209         break;
210     } //switch문의 끝
211     } //if문의 끝
212     } //while문의 끝
213     }
214     catch (SocketException e)
215     {
216         Console.WriteLine(e);
217     }
218     finally
219     {
220         udpSocket.Close();
221     }
222     }
223     private void button3_Click(object sender, EventArgs e) //전진
224     {
225         sendto_flag = 0;
226         wheel_flag = 1;
227     }
228     private void button5_Click(object sender, EventArgs e) //후진
229     {
230         sendto_flag = 0;
231         wheel_flag = 2;
232     }
233     private void button6_Click(object sender, EventArgs e) //좌회전
234     {
235         sendto_flag = 0;
236         wheel_flag = 3;
237     }
238     private void button4_Click(object sender, EventArgs e) //우회전
239     {
240         sendto_flag = 0;
241         wheel_flag = 4;
242     }
243     private void button7_Click(object sender, EventArgs e) //정지
244     {
245         sendto_flag = 0;
246         wheel_flag = 5;

```

```
    }  
    private void button8_Click(object sender, EventArgs e)//모드변경  
    {  
        sendto_flag = 0;  
        wheel_flag = 6;  
    }  
    247  
    248    private void radioButton1_CheckedChanged(object sender, EventArgs e)//주행모드  
    249    {  
    250        mode_flag = 1;  
    251    }  
    252    private void radioButton2_CheckedChanged(object sender, EventArgs e)//임무수행모드  
    253    {  
        mode_flag = 2;  
    }  
    }  
}
```