소스코드

AVR

※ 이동형 로봇팔

* DcPid.c

```c
#include <iom128.h>
#include <ina90.h>
#include "Header/DcPid.h"
#include "Header/i2c_k.h"
#include "Header/dynamixel_k.h"
unsigned char PortCtrl = 0x00;
unsigned int SecFlag = 0, tim_count = 0;
unsigned int LeftObjectSpeed, RightObjectSpeed, LeftPresentSpeed, RightPresentSpeed;
unsigned int DutyA, DutyB;
unsigned int LeftEncoder = 0, RightEncoder = 0;
int RightOutput, LeftOutput, LeftBeforeError, RightBeforeError;
int LeftErrorSum = 0, RightErrorSum = 0;
int LeftError, RightError;
void StartDc(void)
{
    if(twi_ch[2] == 1)//drive mode
    {
        BasicRobotArm();
        PortCtrl = 0x00;
        if(twi_ch[0] > 5)
        {
            LeftObjectSpeed = twi_ch[0] - 2;
            PortCtrl |= 0x01;
        }
        else if (twi_ch[0] < 5)
        {
            LeftObjectSpeed = 8 - twi_ch[0];
            PortCtrl |= 0x02;
        }
        else
        {
            LeftObjectSpeed = 0;
            PortCtrl |= 0x01;
        }

        if(twi_ch[1] > 5)
        {
            RightObjectSpeed = twi_ch[1] - 2;
```

```
            PortCtrl |= 0x04;
        }
        else if(twi_ch[1] < 5)
        {
            RightObjectSpeed = 8 - twi_ch[1];
            PortCtrl |= 0x08;
        }
        else
        {
            RightObjectSpeed = 0;
            PortCtrl |= 0x04;
        }
    }
    else // mission perform mode
    {
        twi_ch[0] = 5;
        twi_ch[1] = 5;
        PortCtrl = 0x00;
    }

    if(SecFlag == 1)
    {
        __disable_interrupt();
        LeftPresentSpeed = (LeftEncoder*312)/1000;
        LeftError = LeftObjectSpeed - LeftPresentSpeed;

        if(LeftError > 0)
        {
            /*LeftOutput = (58*LeftError)/100 + (500*(LeftError-LeftBeforeError))/100 + (10*Left
            LeftOutput = (L_K_P*LeftError + L_K_D*(LeftError-LeftBeforeError)
+ L_K_I*LeftErrorSum)/SCALING_FACTOR;
            DutyA += LeftOutput;
            if(DutyA > 1000)
                DutyA = 1000;
        }
        else if( LeftError < 0)
        {
            DutyA -= 5;
            if(DutyA < 1)
                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;
```

```
        RightEncoder = 0;

        SecFlag = 0;
        __enable_interrupt();
    }
}
void Tim1PwmInit(void)
{
    TCCR1A = 0x00;
    TCCR1B = 0x00;
    TCCR1A |= (1<<WGM10) | (1<<WGM11) | (1<<COM1A1) | (1<<COM1B1) ;
//10bit, TOP = 0x3ff, fast PWM & inverting PWM
    TCCR1B |= (1<<WGM12) | (1<<CS12) | (1<<CS10);
//prescaler 1024
}
void ExtInterruptInit(void)
{
    EIMSK = 0x00;
    EICRA = 0x00;
    EICRB = 0x00;
    EIFR = 0x00;

    EIMSK |= (1<<INT4) | (1<<INT6);                                  //INT4,6
    EICRB |= (1<<ISC41)  | (1<<ISC61) | (1<<ISC40)| (1<<ISC60);
//rising edge interrupt
    EIFR |= (1<<INTF4) | (1<<INTF6);
}
void Tim3OvfInit(void)
{
    TCCR3A = 0x00;
    TCCR3B = 0x00;
    TCCR3C = 0x00;

    ETIMSK = 0x00;
    ETIFR = 0x00;

    TCCR3B |= (1<<CS32);    //prescaler 256
    TCNT3 = 0xffff-625;        //65535 - 625

    ETIMSK |= (1<<TOIE3);    //timer3 overflow interrupt
    ETIFR |= (1<<TOV3);
}
```

```
#pragma vector = INT4_vect
__interrupt void INT4_interrupt(void)
{
    __disable_interrupt();
    LeftEncoder++;
    __enable_interrupt();
}
#pragma vector = INT6_vect
__interrupt void INT6_interrupt(void)
{
    __disable_interrupt();
    RightEncoder++;
    __enable_interrupt();
}
#pragma vector=TIMER3_OVF_vect
__interrupt void TIMER3_OVF_interrupt(void) // 50ms
{
    __disable_interrupt();
    TCNT3 = 0xffff-625;
    tim_count++;
    if(tim_count == 5)
    {
        SecFlag = 1;
        tim_count = 0;
    }
    __enable_interrupt();
}
```

* dynamixel_k.c

```
#include <iom128.h>
#include <ina90.h>
#include "Header/dynamixel_k.h"
#include "Header/uart_k.h"
#include "Header/i2c_k.h"
void StartRobotArm(void)
{
    unsigned int i;
    if(twi_ch[2] == 2)
```

```
    {
        trans0_ch('S');

        for(i=0; i<13; i++)
            uart_arr[i]=receive0_ch();

        UartCheckSum = uart_check_sum(uart_arr,13);

        if( uart_arr[12] == UartCheckSum )
        {
            arm_control_ang_ve(start_arm);
            check_sum(start_arm,start_arm_length);
            for(i=0; i<start_arm_length; i++)
            trans1_ch( start_arm[i] );
        }
    }
}
void BasicRobotArm(void)
{
    unsigned int i;
    check_sum(basic_arr,start_arm_length);
    for(i=0; i<start_arm_length; i++)
        trans1_ch( basic_arr[i] );
}
void dec_hex_convert(int dec, unsigned char* low_temp, unsigned char* high_temp )
{
    *high_temp = 0x00;
    *low_temp = 0x00;

    *high_temp = (unsigned char)0xff&(dec/255);
    *low_temp = (unsigned char)0xff&(dec%255);
}
void check_sum(unsigned char * check_arr,int arr_length)//디지털모터의
//checksum계산하는 함수
{
    int i=0;
    unsigned char data=0;
    unsigned char check=0;

    for(i=2;i<(arr_length-1);i++)//check sum을 구하기 위한 for문
        data += check_arr[i];

```

```
    check = ~data;
    *(check_arr+(arr_length-1))=check;
}
unsigned char start_arm[33] = {
        //각도 인덱스 (8,9) (13,14)(18,19) (23,24)
        //속도 인덱스 (10,11) (15,16) (20,21) (25,26)
        0xff,
        0xff,
        0xfe,
        0x1d,////((L+1)*n)+4,n은 모터개수
        0x83,//sync write 명령어

        0x1e,//control table의 명령어
        0x04,//쓰고자하는 L의 길이

        0x11,//아이디 17
        0x00,//L의 시작
        0x00,
        0x00,
        0x00,//L의 끝

        0x03,//아이디 3
        0x00,
        0x00,
        0x00,
        0x00,

        0x06,//아이디 6
        0x00,
        0x00,
        0x00,
        0x00,

        0x07,//아이디 7
        0x00,
        0x00,
        0x00,
        0x00,

        0x13,//아이디 19
        0x00,
        0x00,
```

```
        0x00,
        0x00,
        //check sum = ~(ID + Length + Instruction + parameter1 + ~ + Parameter N)
        0x00
};
unsigned char basic_arr[33] = {
        //각도 인덱스 (8,9) (13,14)(18,19) (23,24)
        //속도 인덱스 (10,11) (15,16) (20,21) (25,26)
        0xff,
        0xff,
        0xfe,
        0x1d,//((L+1)*n)+4,n은 모터개수
        0x83,//sync write 명령어

        0x1e,//control table의 명령어
        0x04,//쓰고자하는 L의 길이

        0x11,//아이디 17
        0xff,//L의 시작
        0x01,
        0x65,
        0x00,//L의 끝

        0x03,//아이디 3
        0xff,
        0x01,
        0x65,
        0x00,

        0x06,//아이디 6
        0xff,
        0x01,
        0x65,
        0x00,

        0x07,//아이디 7
        0xff,
        0x01,
        0x65,
        0x00,

        0x13,//아이디 19
```

```
        0xff,
        0x01,
        0x65,
        0x00,
        //check sum = ~(ID + Length + Instruction + parameter1 + ~ + Parameter N)
        0x00
};
unsigned int start_arm_length = (sizeof(start_arm)/sizeof(unsigned char));
```

※ 모형 로봇 팔

* adc_k.c

```
#include <iom128.h>
#include <ina90.h>
#include "Header/adc_k.h"
#include "Header/delay_k.h"
unsigned int adc_count = 2;
unsigned int MuxCount = 0;
unsigned char adc_arr[13] = {0xff,0xff,0,0,0,0,0,0,0,0,0,0,0xff};
void StartArmModel(void)
{
    ADMUX = (1<<REFS0) | (MuxCount << MUX0);
    delay_us(10);
    ADCSRA |= (1<<ADSC);
}
void EndArmModel(void)
{
    MuxCount++;
    if(MuxCount == 5)
    {
        MuxCount = 0;
        adc_count = 2;
    }
}
void KongAdcInit(void)
{
    ADMUX = 0x00;
    ADCSRA = 0x00;
    ADCSRA |= (1<<ADEN) | (1<<ADIE) | (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0) ;
}
#pragma vector = ADC_vect
```

```
__interrupt void ADC_conversion_done(void)
{
    __disable_interrupt();

    adc_arr[adc_count] = ADCL;
    adc_arr[adc_count+1] = ADCH;
    adc_count += 2;
    __enable_interrupt();
}
```

* eeprom.c

```
#include <iom128.h>
#include <ina90.h>
#include "Header/eeprom.h"
unsigned int eeprom_write_i=11;
unsigned int eeprom_read_i=11;
unsigned char EEPROM_read(unsigned int uiAddress)
{
    while(EECR & (1<<EEWE));    /* Wait for completion of previous write */
    EEAR = uiAddress;           /* Set up address register */
    EECR |= (1<<EERE);           /* Start eeprom read by writing EERE */
    return EEDR;               /* Return data from data register */
}
void EEPROM_write(unsigned int uiAddress, unsigned char ucData)
{
    while(EECR & (1<<EEWE));    /* Wait for completion of previous write */
    EEAR = uiAddress;           /* Set up address and data registers */
    EEDR = ucData;
    EECR |= (1<<EEMWE);           /* Write logical one to EEMWE */
    EECR |= (1<<EEWE);           /* Start eeprom write by setting EEWE */
}
```

4.2.2 Linux

* RobotArm.c

```
/*
 *-------------------------*
 * FILE : RobotArm.c   *
 * AUTH : Kong, Young Hoon *
 *-------------------------*
 */
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <math.h>
#include <linux/videodev2.h>
#include <arpa/inet.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <pthread.h>
#include <wiringPi.h>
#include <wiringPiI2C.h>
#include "RobotArm.h"
#define DEBUG 1
void YUVtoRGB(unsigned char *ImageBuffer,unsigned char *Y,
unsigned char *Cb, unsigned char *Cr);
void WebCamInit();
void tcp_init();
void udp_init();
void twi_init();
int main (void)
{
    thstat_udpsrv = pthread_create(&thid_udpsrv, NULL , thread_udpsrv , NULL);
       thstat_tcpsrv = pthread_create(&thid_tcpsrv, NULL, thread_tcpsrv, NULL);
    #ifdef DEBUG
    if((thstat_udpsrv < 0)||(thstat_tcpsrv<0))
    {
        printf("Udpsrv or tcpsrv Thread Create Failure..\n");
        exit(1);
    }
    #endif
    thstat_udpsrv = pthread_join(thid_udpsrv, NULL);
       thstat_tcpsrv = pthread_join(thid_tcpsrv, NULL);
    #ifdef DEBUG
    if((thstat_udpsrv != 0) || (thstat_tcpsrv != 0))
    {
        printf("Udpsrv or tcpsrv Thread join Failure..\n");
        exit(1);
```

```
    }
    #endif
    return 0;
}
void * thread_tcpsrv(void * arg)
{
    int i,j,ret;
    unsigned char * bmp_buffer;
    unsigned char * image_buffer;
    YUVBUFFERS yuv_buffer;

    WebCamInit();
    bmp_buffer =  malloc(WIDTH * HEIGHT * DEPTH);
    memset(bmp_buffer, 0, WIDTH*HEIGHT*DEPTH);

    image_buffer = mmap( NULL, buf.length, PROT_READ | PROT_WRITE ,
MAP_SHARED, cam_fd, buf.m.offset);
    #ifdef DEBUG
    if(MAP_FAILED == image_buffer)
    {
        printf("memory map fail\n");
        return 0;
    }
    #endif
    tcp_init();
    while(1)
    {
        ret = ioctl(cam_fd,VIDIOC_QBUF,&buf);
        #ifdef DEBUG
        if(ret<0)
        {
            printf("VIDIOC_QBUF fail\n");
            exit(1);
        }
        #endif
        ret = ioctl(cam_fd, VIDIOC_DQBUF,&buf);
        #ifdef DEBUG
        if(ret<0)
        {
            printf("Retrieving Frame\n");
            exit(1);
        }
```

```
        #endif
        for(i=0; i< (WIDTH*HEIGHT*2); i+=4)
        {
            yuv_buffer.y[i/2]    = image_buffer[i];
            yuv_buffer.cb[i/4]    = image_buffer[i+1];
            yuv_buffer.y[(i/2)+1]    = image_buffer[i+2];
            yuv_buffer.cr[i/4]    = image_buffer[i+3];
        }
        YUVtoRGB(bmp_buffer,yuv_buffer.y,yuv_buffer.cb, yuv_buffer.cr);
        write(tcp_clnt_sock,bmp_buffer,BMPMAXBUFF);
    }
    munmap(image_buffer,buf.length);
    free(bmp_buffer);
    close (cam_fd);
    close(tcp_clnt_sock);
    close(tcp_serv_sock);
}
void * thread_udpsrv(void * arg)
{
    int strlen;
    twi_init();
    udp_init();
    udp_clnt_adr_size=sizeof(udp_clnt_adr);
    while(1)
    {
        strlen = recvfrom(udp_serv_sock,UdpTwiArr, 3, 0,
 (struct sockaddr*)&udp_clnt_adr, &udp_clnt_adr_size);
        UdpTwiFlag = strlen;
        #ifdef DEBUG
        if(UdpTwiFlag != 0)
        {
            printf("udp receive data : %d ,%d, %d \n", (int)UdpTwiArr[0],
 (int)UdpTwiArr[1], (int)UdpTwiArr[2] );
        }
        wiringPiI2CWrite(twi_fd, UdpTwiArr[0]);
        usleep(10);
        wiringPiI2CWrite(twi_fd, UdpTwiArr[1]);
        usleep(10);
        wiringPiI2CWrite(twi_fd, UdpTwiArr[2]);
        usleep(10);
        #endif
    }
```

```
    close(udp_serv_sock);
}
void YUVtoRGB(unsigned char *ImageBuffer,unsigned char *Y, unsigned char *Cb,
unsigned char *Cr)
{
    int i,j;
    unsigned char R[WIDTH*HEIGHT], G[WIDTH*HEIGHT], B[WIDTH*HEIGHT];
    for(i=0; i<WIDTH*HEIGHT; i++)
    {
        R[i] =  CLIP( ( 298*(Y[i]-16) + 409*(Cb[i/2]-128) + 128 ) >> 8 );
        G[i] =  CLIP( ( 298*(Y[i]-16) - 100*(Cr[i/2]-128) -
208*(Cb[i/2]-128) + 128 ) >> 8 );
        B[i] =  CLIP( ( 298*(Y[i]-16) + 516*(Cr[i/2]-128)+ 128 ) >> 8 );
    }
    for(i=0, j=0; i<(WIDTH*HEIGHT*3); i+=3, j++) // combine rgb bmp 만들기 전
    {
        ImageBuffer[i]     = R[j];
        ImageBuffer[i+1] = G[j];
        ImageBuffer[i+2] = B[j];
    }
}
void WebCamInit()
{
    int ret;
    cam_fd = open("/dev/video0",O_RDWR);
    #ifdef DEBUG
    if( cam_fd < 0 )
    {
        printf("cam open fail\n");
        exit(1);
    }
    #endif
    ret = ioctl( cam_fd, VIDIOC_QUERYCAP, &cap );

    #ifdef DEBUG
    if( ret < 0 )
    {
        printf("vidioc_querycap fail\n");
        exit(1);
    }
    #endif
    memset(&fmt, 0, sizeof(fmt));
```

```
    fmt.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    fmt.fmt.pix.width = WIDTH;
    fmt.fmt.pix.height = HEIGHT;
    fmt.fmt.pix.pixelformat = V4L2_PIX_FMT_YUYV;
    fmt.fmt.pix.field = V4L2_FIELD_INTERLACED;
    ret = ioctl( cam_fd, VIDIOC_S_FMT, &fmt);
    #ifdef DEBUG
    if(ret < 0)
    {
        printf("vidio_s_fmt fail\n");
        exit(1);
    }
    #endif
    memset( &req_buf, 0, sizeof(req_buf));
    req_buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    req_buf.memory = V4L2_MEMORY_MMAP;
    req_buf.count = 1;
    ret=ioctl(cam_fd,VIDIOC_REQBUFS,&req_buf);
    #ifdef DEBUG
    if( ret < 0 )
    {
        printf("video capture or mmap fail or VIDIOC_REQBUFS fail\n");
        exit(1);
    }
    #endif
    memset(&buf, 0 ,sizeof(buf));
    buf.type = V4L2_BUF_TYPE_VIDEO_CAPTURE;
    buf.memory = V4L2_MEMORY_MMAP;
    buf.index = 0;
    ret = ioctl(cam_fd,VIDIOC_QUERYBUF,&buf);
    #ifdef DEBUG
    if( ret < 0 )
    {
        printf("querying buffer");
        exit(1);
    }
    #endif
    ret = ioctl ( cam_fd, VIDIOC_STREAMON,&buf.type);
    #ifdef DEBUG
    if( ret < 0 )
    {
        printf(" capture error");
```

```
        exit(1);
    }
    printf("Cam init success\n");
    #endif
}
void tcp_init()
{
    int ret;
    tcp_serv_sock=socket(PF_INET,SOCK_STREAM,0);
    #ifdef DEBUG
    if(tcp_serv_sock==-1)
    {
        printf("Socket Error...\n");
        exit(1);
    }
    #endif
    memset(&tcp_serv_adr,0,sizeof(tcp_serv_adr));
    tcp_serv_adr.sin_family=AF_INET;
    tcp_serv_adr.sin_addr.s_addr=htonl(INADDR_ANY);
    tcp_serv_adr.sin_port=htons(PORT);
    ret=bind(tcp_serv_sock,(struct sockaddr*)&tcp_serv_adr,sizeof(tcp_serv_adr));

    #ifdef DEBUG
    if(ret == -1)
    {
        printf("bind error");
        exit(1);
    }
    #endif
    ret=listen(tcp_serv_sock,5);
    #ifdef DEBUG
    if(ret== -1)
    {
        printf("listen error");
        exit(1);
    }
    #endif
    tcp_clnt_adr_size=sizeof(tcp_clnt_adr);
    tcp_clnt_sock=accept(tcp_serv_sock,
(struct sockaddr*)&tcp_clnt_adr,&tcp_clnt_adr_size);
    #ifdef DEBUG
    if(tcp_clnt_sock == -1)
```

```
    {
        printf("accept error");
        exit(1);
    }
    printf("tcp server init success\n");
    #endif
}
void udp_init()
{
    int ret;
    udp_serv_sock=socket(PF_INET, SOCK_DGRAM, 0);
    #ifdef DEBUG
    if(udp_serv_sock==-1)
    {
        printf("UDP socket creation error");
        exit(1);
    }
    #endif
    memset(&udp_serv_adr, 0, sizeof(udp_serv_adr));
    udp_serv_adr.sin_family=AF_INET;
    udp_serv_adr.sin_addr.s_addr=htonl(INADDR_ANY);
    udp_serv_adr.sin_port=htons(PORT);

    ret = bind(udp_serv_sock, (struct sockaddr*)&udp_serv_adr,
sizeof(udp_serv_adr));
    #ifdef DEBUG
    if(ret == -1)
    {
        printf("udp bind error");
        exit(1);
    }
    printf("udp server init success\n");
    #endif
}
void twi_init()
{
    twi_fd = wiringPiI2CSetup(twi_slave_addr);
    #ifdef DEBUG
    if( twi_fd == -1 )
    {
        printf("twi setup fail\n");
            exit(1);
```

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

4.2.3 WinForm

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
namespace RobotArm
{
    public partial class Form1 : Form
    {
        byte mode_flag = 1;
        byte[] BmpBuffer = new byte[230400];
        int CamMaxBufSize = 230400;
        uint cam_flag = 1;
        uint wheel_flag = 0, sendto_flag = 0;
        uint t1_flag = 0, t2_flag = 0;
        string raspberry_ip = "192.168.10.12";
        string pc_ip = "192.168.10.11";
        Bitmap bmp = new Bitmap(width: 320, height: 240,
format: System.Drawing.Imaging.PixelFormat.Format24bppRgb);
        Thread t1, t2;
        public Form1()
        {
            InitializeComponent();
        }
        private void Form1_FormClosing(object sender, FormClosingEventArgs e)
        {
            if (MessageBox.Show("정말 종료하시겠습니까?", "질문",
 MessageBoxButtons.YesNo) == DialogResult.No)
            {
                e.Cancel = true;
```

```
            }
            else
            {
                if (t1_flag == 1)
                {
                    t1.Abort();
                }
                if (t2_flag == 1)
                {
                    t2.Abort();
                }
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            t1 = new Thread(new ThreadStart(TcpCamThread));
            t1.Start();
            t1_flag = 1;
        }
        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            cam_flag = ~cam_flag;
        }
        private void TcpCamThread()
        {
            int length, byteLength = 0;
            NetworkStream stream = null;
            TcpClient client = null;
            IPEndPoint clientAddress = null;
            IPEndPoint serverAddress = null;
            try
            {
                clientAddress = new IPEndPoint(address: IPAddress.Parse(pc_ip),
port: 10000);
                serverAddress = new IPEndPoint(address: IPAddress.Parse(raspberry_ip),
port: 10000);
                client = new TcpClient(clientAddress);
                client.Connect(serverAddress);
                stream = client.GetStream();
                while (true)
                {
                    if (cam_flag == 1)
```

```
                {
                    while ((length = stream.Read(buffer: BmpBuffer,
offset: byteLength, size: CamMaxBufSize - byteLength)) != 0)
                    {
                        byteLength += length;
                    }
                    if (byteLength == 230400)
                    {
                        BmpInit();
                        pictureBox1.Image = bmp;
                        byteLength = 0;
                    }
                }
            }
            catch (SocketException e)
            {
                Console.WriteLine(e);
            }
            finally
            {
                stream.Close();
                client.Close();
            }
        }
        private void BmpInit()
        {
            System.Drawing.Imaging.BitmapData bmpData =
                bmp.LockBits(new Rectangle(0, 0, bmp.Width, bmp.Height),
 System.Drawing.Imaging.ImageLockMode.ReadOnly, bmp.PixelFormat);
            IntPtr ptr = bmpData.Scan0;
            System.Runtime.InteropServices.Marshal.Copy(source: BmpBuffer,
startIndex: 0, destination: ptr, length: 320 * 240 * 3);
            bmp.UnlockBits(bmpData);
        }
        private void button2_Click(object sender, EventArgs e)
        {
            t2 = new Thread(new ThreadStart(UdpModeWheelCtrl));
            t2.Start();
            t2_flag = 1;
        }
        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
```

```
                            wheel_mode_arr[0] = 5;
                            wheel_mode_arr[1] = 5;
                            udpSocket.SendTo(wheel_mode_arr, remoteEP);
                            sendto_flag = 1;
                            break;
                        }//switch문의 끝
                    }//if문의 끝
                }//while문의 끝
            }
            catch (SocketException e)
            {
                Console.WriteLine(e);
            }
            finally
            {
                udpSocket.Close();
            }
        }
        private void button3_Click(object sender, EventArgs e)//전진
        {
            sendto_flag = 0;
            wheel_flag = 1;
        }
        private void button5_Click(object sender, EventArgs e)//후진
        {
            sendto_flag = 0;
            wheel_flag = 2;
        }
        private void button6_Click(object sender, EventArgs e)//좌회전
        {
            sendto_flag = 0;
            wheel_flag = 3;
        }
        private void button4_Click(object sender, EventArgs e)//우회전
        {
            sendto_flag = 0;
            wheel_flag = 4;
        }
        private void button7_Click(object sender, EventArgs e)//정지
        {
            sendto_flag = 0;
            wheel_flag = 5;
```

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