자동 정밀타격 포탑

- 팀장 : 김동혁
- 팀원 : 이동훈
  - 김왕배
  - 정범수

역할 분담(1차 시기)

- 동혁 (팀장) : CAN/NETWORKS SERVER, 기구(몸체) 설계, 통신
- 왕배 : (레일건) 회로제작 및 실험, MCU, FPGA
- 범수 : FPGA(LIDAR, 절대엔코더), MPU (기구 수평)
- 동훈 : (레이저) 제작 및 실험, MCU, 제어기(속도), (+A 기구 설계)

코일건

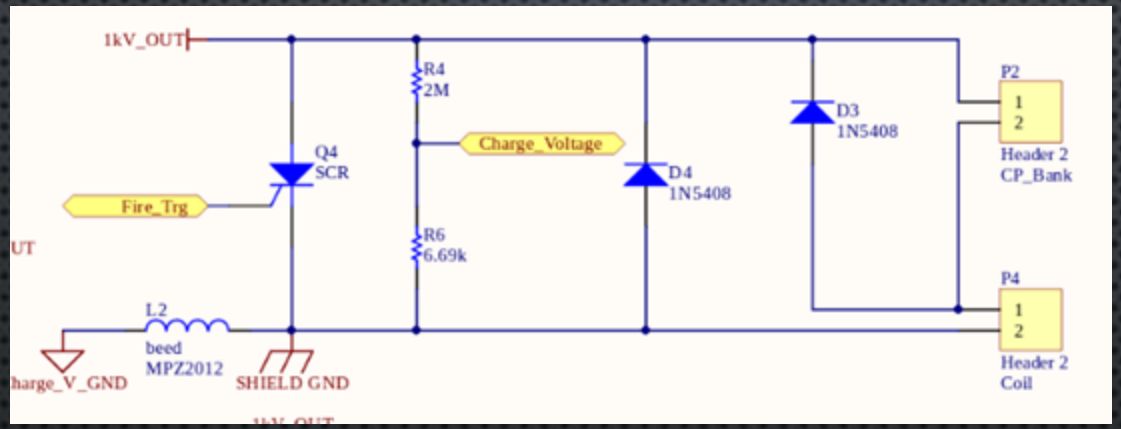
전압 ADC

계산값(adc)	측정값
26	39
38	50
45	57
98	107
133	141.6
220	225
291	294
371	371
525	546
600	583
650	634
700	684

- 문제점 : 실측과 예측값이 다름
  - 발사 전압 700~720이므로
- 약 15정도 더하여 해당구간만 개선

발사 트리거

- GIO핀을 이용하여 SCR GATE에 전압을 걸어 발사



- 최소 WAIT 0X30값이상은 돼야 안정적으로 발사가능

```
gioSetBit(gioPORTA, 7, 1);
wait(0x30);
gioSetBit(gioPORTA, 7, 0);
```

기타

- PWM을 통한 서지개선
  - 부품 배송지연으로 인해 다음주 실행
  - DC컨버터의 NE555가 8K , 80K PWM으로 조절할 예정
  - 개선이 되었는지 확인 방법이 없다
- 발사 불량
  - 지금까지 잘되던 코일건이 갑자기 발사불량상태
  - 원인파악중

기구부

리니어 가이드 부분 조립

- 리니어 가이드 부분 조립
- 결합이 가능한지 여부 확인
- 동작에 걸림이 없는지 확인

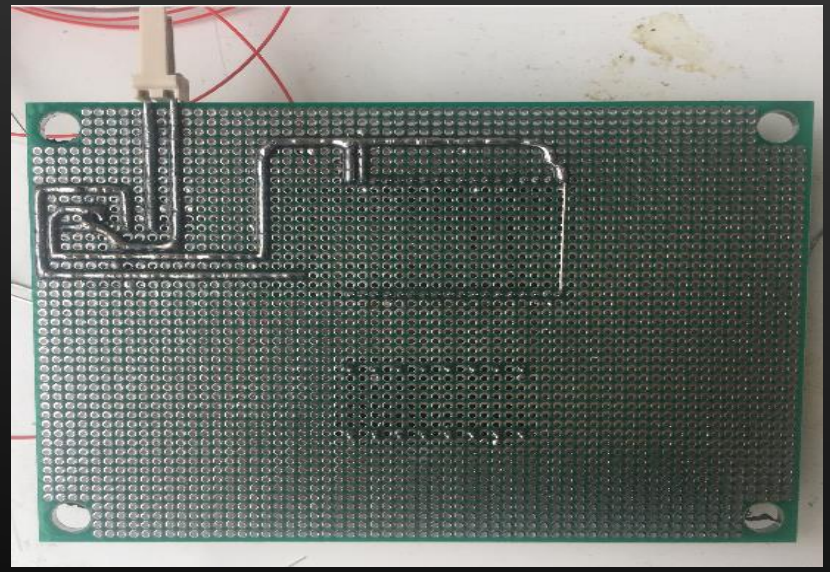
포신 부분 일부 조립

- 차폐부분 조립
- 볼트가 부족하여 전체 조립하지 못함
- LIDAR와 LASER 방열부분, 포신 앞면에 결합 가능 한지 여부 확인

BLUE TOOTH

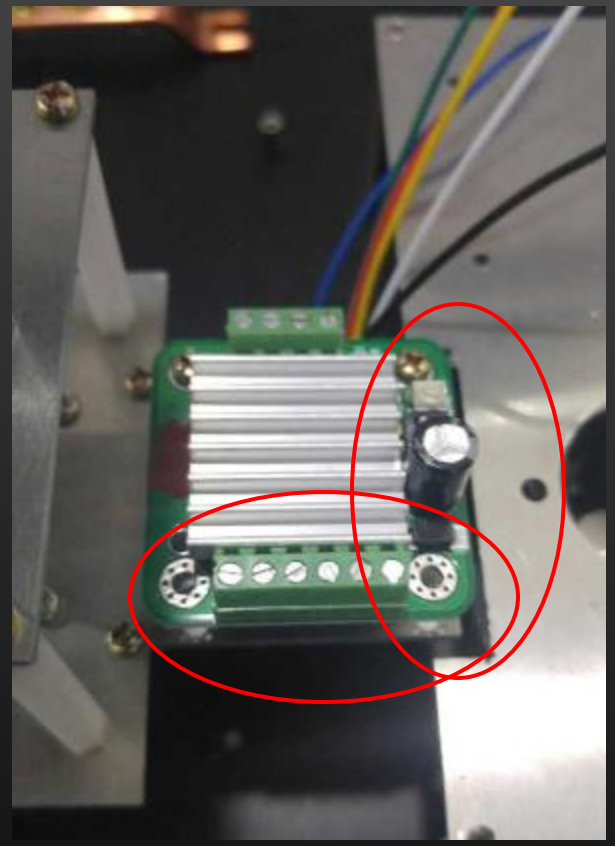
- BLUE TOOTH 회로 납땜

- 회로 납땜 후 동작 테스트



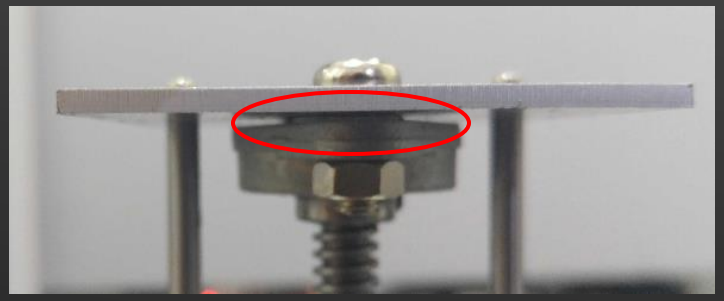
문제 발생 및 수정

- 뒤쪽 AB엔코더 고정부분과 겹치는 부분이 있어 겹치는 부분을 잘라냄
- 구멍 홀이 맞지 않아서 한 쪽 부분만 고정



문제 발생 및 수정

- 스크류가 회전할 때 위쪽 면에 닿아 마찰이 생기는 문제가 발생
- 임시로 얇은 플라스틱에 구멍을 뚫어 와셔 대용으로 사용.



문제 발생 및 수정

- LIDAR 결합시 거리는 문제 발생
- 걸리는 부분을 그라인더로 갈아서 해결함
- 레이저 방열 부분과 결합되는 부분에 나사산을 내주지 않아서 레이저가 들어가지 않음
- 해당 부분을 넓히고 레이저 방열 부분의 나사산을 없애서 해결

DSP

MCU 테스트 코드 작성

- DSP와 UDP 통신을 위해서 MCU 테스트 코드 작성
- MCU코드는 정한 프로토콜을 바탕으로 작성함
- 테스트 코드의 동작은 처음 S7의 데이터를 보내다가 DSP로 부터 G 명령을 받으면 S6 데이터를 보냄.

```
        p = pbuf_alloc(PBUF_TRANSPORT, 3, PBUF_RAM);
        if(start_send_flag)
        {
            udbuf[0] ='s';
            udbuf[1] = (status_flag_test-1);
            udbuf[2] = 0;
            memcpy(p->payload,udbuf, 3);
            udp_sendto(pcb, p, IP_ADDR_BROADCAST, 7777);
        }
        else
        {
            udbuf[0] = 's';
            udbuf[1] = status_flag_test;
            udbuf[2] = 0;
            memcpy(p->payload,udbuf, 3);
            udp_sendto(pcb, p, IP_ADDR_BROADCAST, 7777);
        }
```

```
    if (p != NULL)
    {
        sprintf(vbuf,"UDP on data\n\r");
        char *rx_pk = p->payload;
        if(rx_pk[0] == 's')
        {
#if 1
            setCNT = rx_pk[1] << 24U |
                     rx_pk[2] << 16U |
                     rx_pk[3] << 8U  |
                     rx_pk[4];
#endif
            setDGR = rx_pk[5] << 24U |
                     rx_pk[6] << 16U |
                     rx_pk[7] << 8U  |
                     rx_pk[8];


#if SCI_DEBUG
            sprintf(vbuf,"%d,%d\n\r",setCNT,setDGR);
#endif
        }
        /* MCU가 준비되서 Ready signal을 전송하면 DSP에서 받고 준비되면 'g'를 보내서 MCU 전체 테스크 동작 시작. */
        else if(rx_pk[0] == 'g')
        {
            start_send_flag =1;
            sprintf(vbuf,"start send data to dsp");
        }
```

```
void init_udp(int *sock, sockad_in *server_addr, sockad_in *client_addr,uint16_t port)
{

    read_mcu_state = 0;
    *sock = socket(PF_INET, SOCK_DGRAM,0);

    if(*sock == -1)
    {
        printf("Socket creation failed\r\n");
        exit(1);
    }

    memset(server_addr,0,sizeof(*server_addr));
    server_addr->sin_family = AF_INET; //IPv4 Internet
    server_addr->sin_port = htons(port); //port_num 7777(host to network short(Big Endia
    server_addr->sin_addr.s_addr = htonl(INADDR_ANY); //assign automatically an IP addre

    if(bind(*sock,(sockad *)server_addr,sizeof(*server_addr)) == -1)
    {
        printf("Bind execution error\r\n");
        exit(1);
    }

    usleep(100);
}
```

```
void Udp_Send2MCU(int *sock, sockad_in *client_addr, int ins,int setCnt, int setDeg)
{
    char buff_snd[10] ={0,0,0,0,0,0,0,0,0,0};

    if(ins)
    {
        buff_snd[0] = 's';
        buff_snd[1] = (setCnt & 0xFF000000) >> 24;
        buff_snd[2] = (setCnt & 0x00FF0000) >> 16;
        buff_snd[3] = (setCnt & 0x0000FF00) >> 8;
        buff_snd[4] = setCnt & 0x000000FF;
        buff_snd[5] = (setDeg & 0xFF000000) >> 24;
        buff_snd[6] = (setDeg & 0x00FF0000) >> 16;
        buff_snd[7] = (setDeg & 0x0000FF00) >> 8;
        buff_snd[8] = setDeg & 0x000000FF;
        sendto(*sock, buff_snd, 10, 0, (sockad *)client_addr, sizeof(*client_addr));
    }
    else
    {
        buff_snd[0] = 'g';
        printf("sand_data g\r\n");
        sendto(*sock, buff_snd, 2, 0, (sockad *)client_addr, sizeof(*client_addr));
    }
```

```
void Udp_Receive2MCU(int *sock, sockad_in *client_addr)
{
    int client_addr_size = sizeof(*client_addr);
    char rcv_buf[3] = {0,0,0};
    volatile char mcu_state =0;
#if 1
    recvfrom(*sock, rcv_buf, 3,MSG_DONTWAIT , (sockad *)client_addr, &client_addr_size);
#else
    recvfrom(*sock, rcv_buf, 3, 0 , (sockad *)client_addr, &client_addr_size);
#endif
    if(rcv_buf[0] == 's')
    {
          mcu_state = rcv_buf[1];
        printf("Receive data %c%d\r\n", rcv_buf[0],rcv_buf[1]);
    }

    if(read_mcu_state == 0)
    {
        read_mcu_state = mcu_state;
    }
}
```

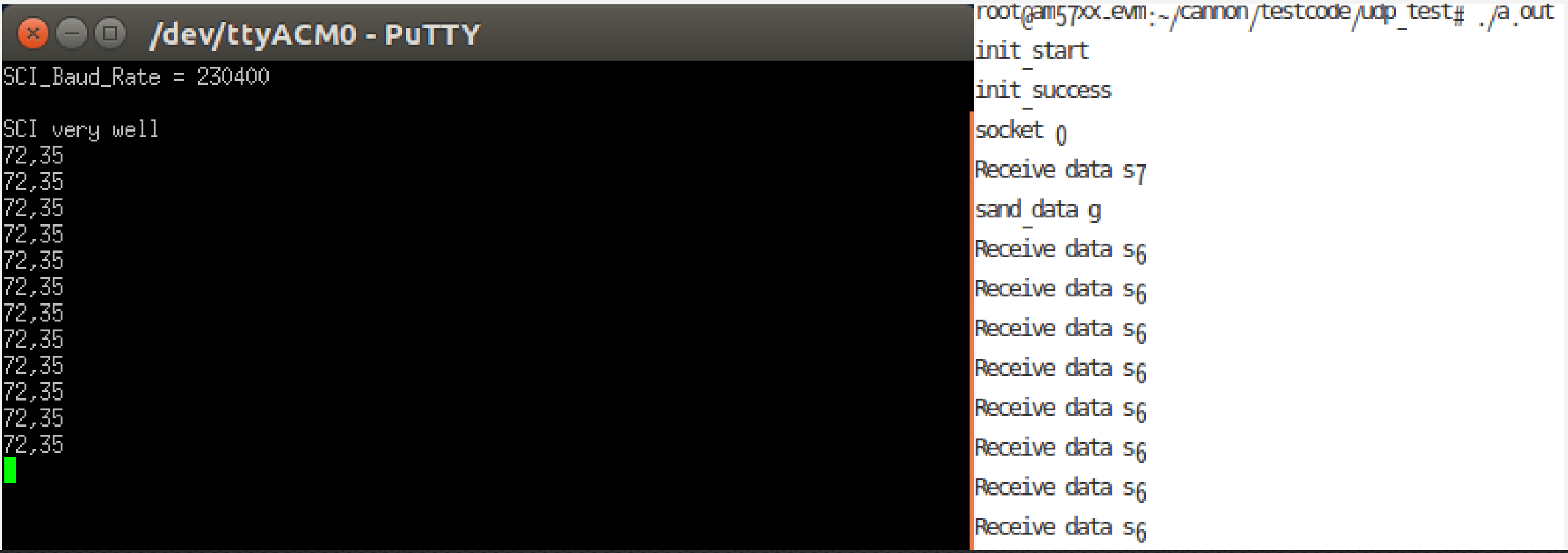
DSP 코드 작성

- UDP_COM.C 파일을 따로 작성하여 UDP 관련 함수를 모아 놓음
- 해당 C파일은 초기화 SAND, RECEIVE 부분으로 나뉘어 있음

DSP MAIN문

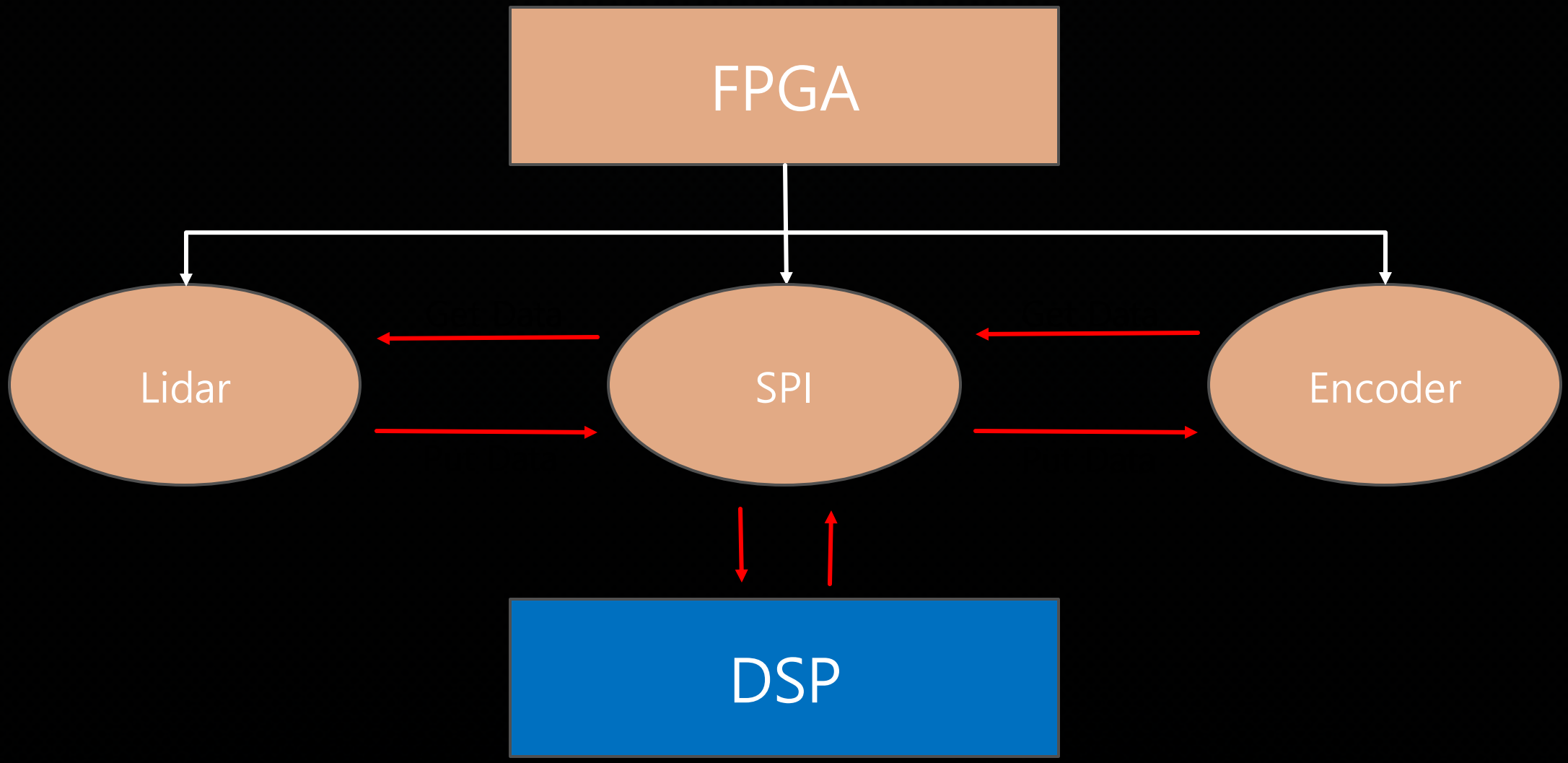
- UDP_COM.H를 불러와서 조건에 따라 UDP함수를 실행
- MCU에서 통신 가능한 상태가 되면 G 명령을 보내어 통신 시작을 알림
- 통신 가능한 상태는 MCU가 보내주는 상태 값으로 확인

```
int setCNT = 72;
int degree = 35;

int main(void)
{
    int sock = -1;
    int first_recieve =1;
    sockad_in   server_addr;
    sockad_in   client_addr;

    printf("init_start\r\n");
    init_udp(&sock, &server_addr, &client_addr, UDP_PORT);
    printf("init_success\r\n");
    printf("socket %d\r\n");

    while(1)
    {
        Udp_Receive2MCU(&sock, &client_addr);

        if(read_mcu_state)
        {
            if(((read_mcu_state & 0X04)>>2) && first_recieve)
            {
                Udp_Send2MCU(&sock, &client_addr,STR_MCU,0,0); //start mcu g instrument
                first_recieve =0;
            }

            Udp_Send2MCU(&sock, &client_addr,SET_MCU,setCNT,degree);
            read_mcu_state = 0;
        }
    }

    return 0;
}
```



DSP와 MCU간 UDP 통신 결과

FPGA


FPGA
Lidar
SPI
Encoder
DSP

진행 상황

TX

Hide completed items | Delete

100%

- ~~Extended ID 송신~~
- ~~DATA 송신~~

Add an item

RX

Hide completed items | Delete

100%

- ~~Extended ID 수신~~
- ~~DATA 수신~~

TXRX

Hide completed items | Delete

100%

- ~~송수신 완료~~

Add an item

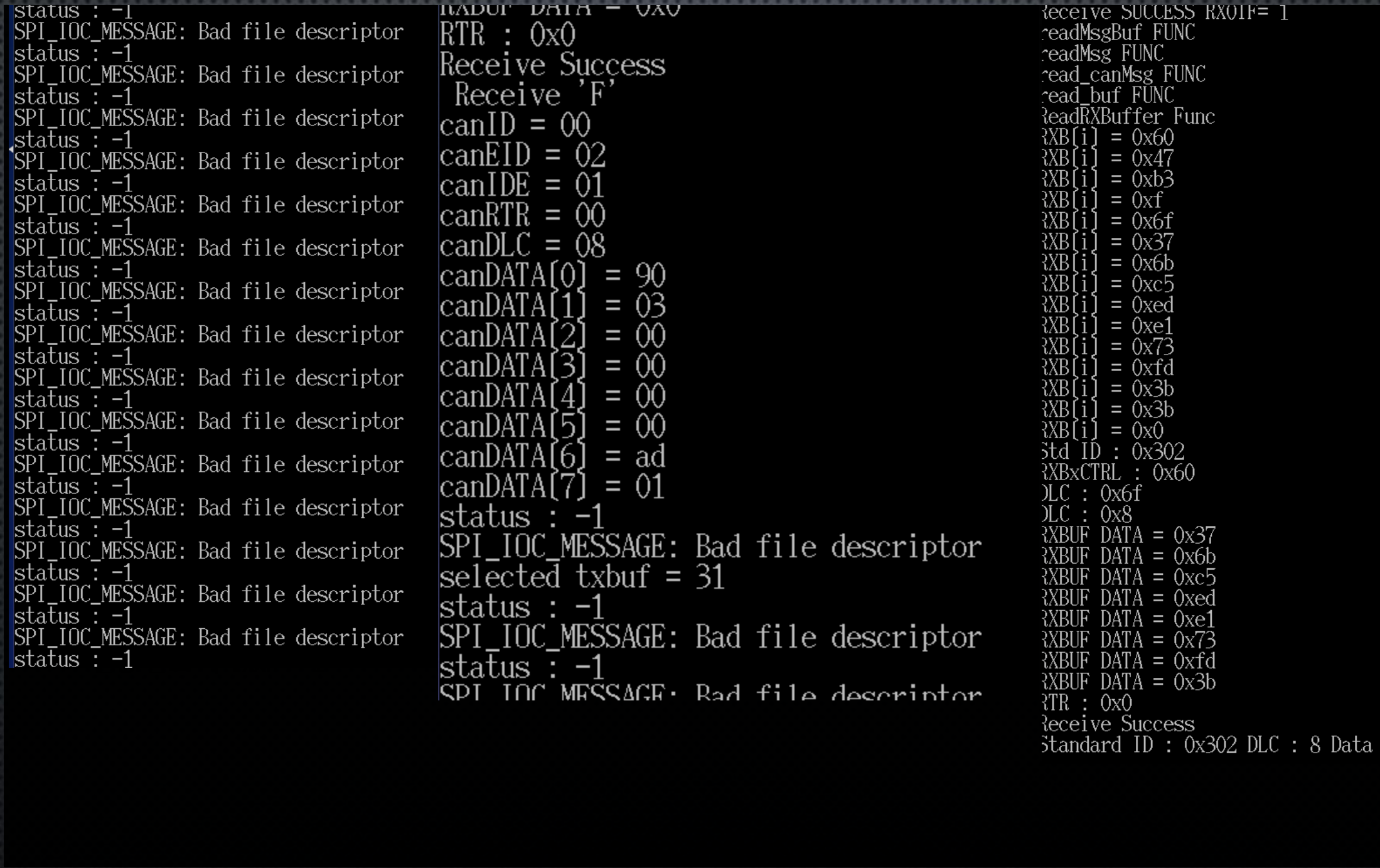
문제점

Hide completed items | Delete

13%

- [ ] SIDH레지스터에 0x60 들어오는 이유 모름
- [ ] TXRX 송수신 시 수신 값 오류
- [ ] FPGA->DSP 데이터 전송 중 DSP 가 송신할 때
- [ ] DSP->FPGA 데이터 전송 중 FPGA 가 송신할 때
- [ ] RX 데이터 값 오류
- [ ] Bad file descriptor 오류
- [x] ~~RX buffer명령 오류 (버퍼길이)~~
- [ ] delay

```
Receive Success
Standard ID : 0x2 DLC : 8 Data
Receive SUCCESS RX0IF= 1
readMsgBuf FUNC
readMsg FUNC
read_canMsg FUNC
read_buf FUNC
RXBUF[0] = 0
RXBUF[1] = 8
RXBUF[2] = 0
RXBUF[3] = 2
RXBUF[4] = 8
RXBUF[5] = 17
RXBUF[6] = 34
RXBUF[7] = 51
RXBUF[8] = 68
RXBUF[9] = 85
RXBUF[10] = 102
RXBUF[11] = 119
RXBUF[12] = 136
RXBUF[13] = 0
RXBUF[14] = 0
Std ID : 0x0
Ext ID : 0x2
RXBxCTRL : 0x60
DLC : 0x8
DLC : 0x8
RXBUF DATA = 0x11
RXBUF DATA = 0x22
RXBUF DATA = 0x33
RXBUF DATA = 0x44
RXBUF DATA = 0x55
RXBUF DATA = 0x66
RXBUF DATA = 0x77
RXBUF DATA = 0x88
RTR : 0x0
Receive Success
Standard ID : 0x2 DLC : 8 Data
```


status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
RTR : 0x0
Receive Success
Receive 'F'
canID = 00
canEID = 02
canIDE = 01
canRTR = 00
canDLC = 08
canDATA[0] = 90
canDATA[1] = 03
canDATA[2] = 00
canDATA[3] = 00
canDATA[4] = 00
canDATA[5] = 00
canDATA[6] = ad
canDATA[7] = 01
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
selected txbuf = 31
status : -1
SPI_IOC_MESSAGE: Bad file descriptor
status : -1
eadMsgBuf FUNC
eadMsg FUNC
ead_canMsg FUNC
ead_buf FUNC
eadRXBuffer Func
XB[i] = 0x60
XB[i] = 0x47
XB[i] = 0xb3
XB[i] = 0xf
XB[i] = 0x6f
XB[i] = 0x37
XB[i] = 0x6b
XB[i] = 0xc5
XB[i] = 0xed
XB[i] = 0xe1
XB[i] = 0x73
XB[i] = 0xfd
XB[i] = 0x3b
XB[i] = 0x3b
XB[i] = 0x0
td ID : 0x302
XBxCTRL : 0x60
LC : 0x6f
LC : 0x8
XBUF DATA = 0x37
XBUF DATA = 0x6b
XBUF DATA = 0xc5
XBUF DATA = 0xed
XBUF DATA = 0xe1
XBUF DATA = 0x73
XBUF DATA = 0xfd
XBUF DATA = 0x3b
TR : 0x0
eceive Success
tandard ID : 0x302 DLC : 8 Data

MCU

이번 주 예정

1. MCU에서 데이터를 요청할 때만 전송 (CAN)
2. 통신 속도 변경
3. 오일러 각도 중 pich or roll 만 전송

표 3-1 MW-AHRS 센서의 오브젝트 요약

Name	Index	Sub-i	Access, Size	Description	Default
ver	1	0	RO, INT32	공급자 ID, 0으로 고정	0
ver	2	0	RO, INT32	제품(AHRS 센서) ID	5001
ver	3	0	RO, INT32	장치 펌웨어 버전	100
ver	4	0	RO, INT32	장치 하드웨어 버전	200
cmd	7	0	WO, INT32	장치에 내려지는 명령 (RS-232 Text 모드에서는 다음 명령 사용 가능: fw, fd, cal, cam, zro, rcd, rst, ver, h, help)	
id	11	0	RW, INT32	장치 ID	1
cb	12	0	RW, INT32	CAN 통신 속도 [Kbps]	1000
sb	13	0	RW, INT32	RS-232 통신 속도 [bps]	115200
gs	15	0	RW, INT32	자이로 센서의 측정 스케일 설정	0 ~ 3
as	16	0	RW, INT32	가속도 센서의 측정 스케일 설정	0 ~ 3
mv	19	0	RW, INT32	자기 센서의 측정값에 대한 분산 설정	0
av	20	0	RW, INT32	가속도 센서의 측정값에 대한 분산 설정	1000
ss	21	0	RW, INT32	RS-232로 동기화 데이터 전송 설정	0
sc	22	0	RW, INT32	CAN으로 동기화 데이터 전송 설정	0
sp	24	0	RW, INT32	동기화 데이터 전송 주기 설정 [ms]	100
st	25	0	RW, INT32	장치에 전원이 투입될 때, RS-232 데이터 전송 타입 결정 (0-Binary, 1-Text)	1
acc	51	1~3	RO, FLOAT	가속도 데이터 전송 (x, y, z) [g]	
gyr	52	1~3	RO, FLOAT	각속도 데이터 전송 ($\omega_x, \omega_y, \omega_z$) [°/s]	
ang	53	1~3	RO, FLOAT	오일러 각도 전송 (ϕ, θ, ψ) [°]	
mag	54	1~3	RO, FLOAT	자기 데이터 전송 (x, y, z) [μT]	
tmp	57	1	RO, FLOAT	온도 전송 [°C]	

AHRSV1 TEST (MCU TO AHRS)

CAN 동작 확인


```
HL_sys_main.c | HL_can.c | trgmsg.c

85
86      enable_interrupt ();
87
88      canInit();
89      printf("can initializing..\n");
90      canEnableErrorNotification(canREG2);
91
92      while(gioGetBit(gioPORTB, 4))
93      {
HL_sys_main.c, line 92 (main + 0x48) [H/W BP]
95      }
96
97      canTransmit(canREG2, canMESSAGE_BOX1, tx_data);
98      wait(1000);
99
100     canTransmit(canREG2, canMESSAGE_BOX1, tx_data2);
101     wait(1000);
102
103     for(;;)
104     {
105         gioToggleBit(gioPORTB, 6);
106 #if 0
107         if(canIsRxMessageArrived(canREG2, canMESSAGE_BOX2))
108         {
109             canGetData(canREG2, canMESSAGE_BOX2, rx_data);
110         }
```

Memor... 0xFFF7DF30

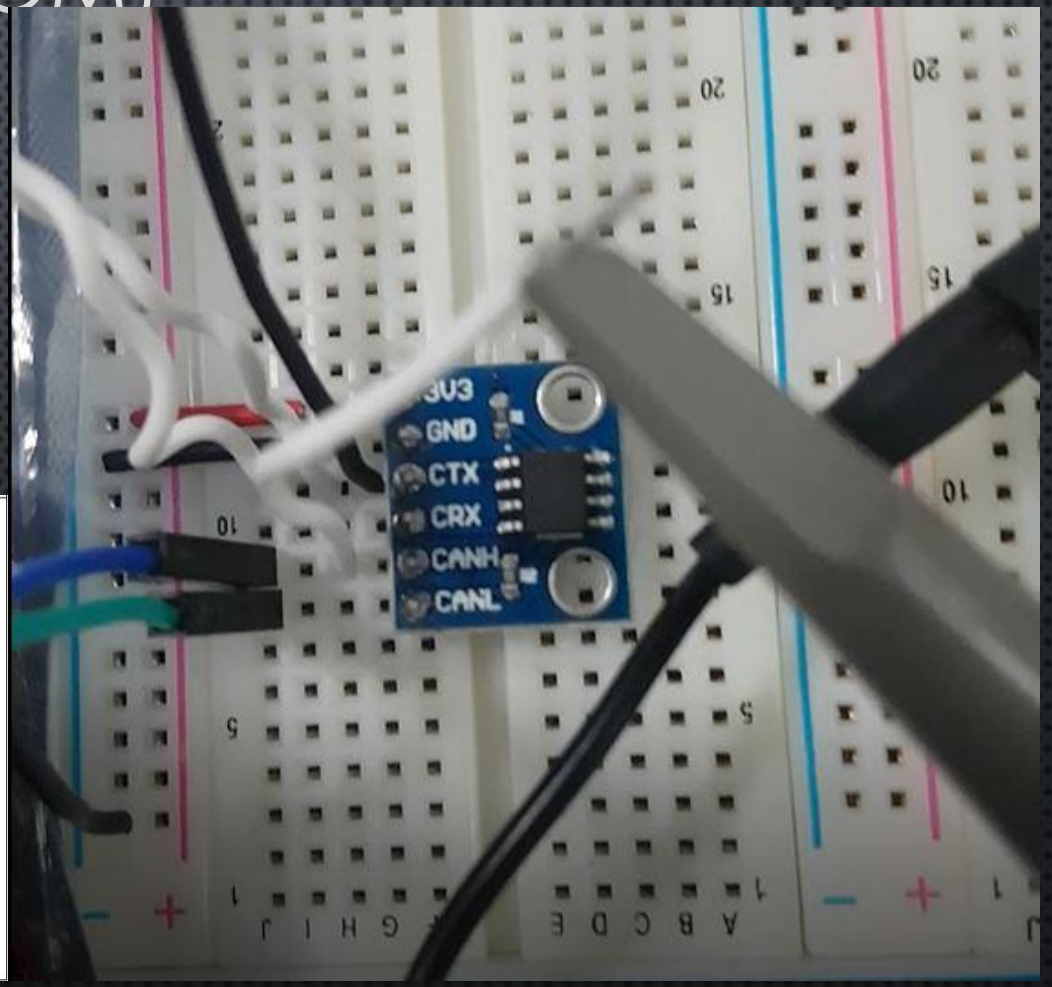
Console

```
test_AHRS:CIO
rx_data : f0    34      0       0       0       0       0       0
rx_data : f0    33      ff      ff      2       0       ea      3
rx_data : f0    34      0       0       0       0       1       0
rx_data : f0    33      ff      ff      2       0       e4      3
rx_data : f0    34      1       0       0       0       ff      ff
rx_data : f0    33      fa      ff      2       0       e4      3
rx_data : f0    34      0       0       0       0       0       0
rx_data : f0    33      ff      ff      0       0       eb      3
rx_data : f0    34      1       0       0       0       0       0
```

CAN TX TEST (MCU TO COM)

원하는 동작 :
MCU에서 Transceiver을 통해 메시지를 전송하면 COM에서 Receive를 해야함.

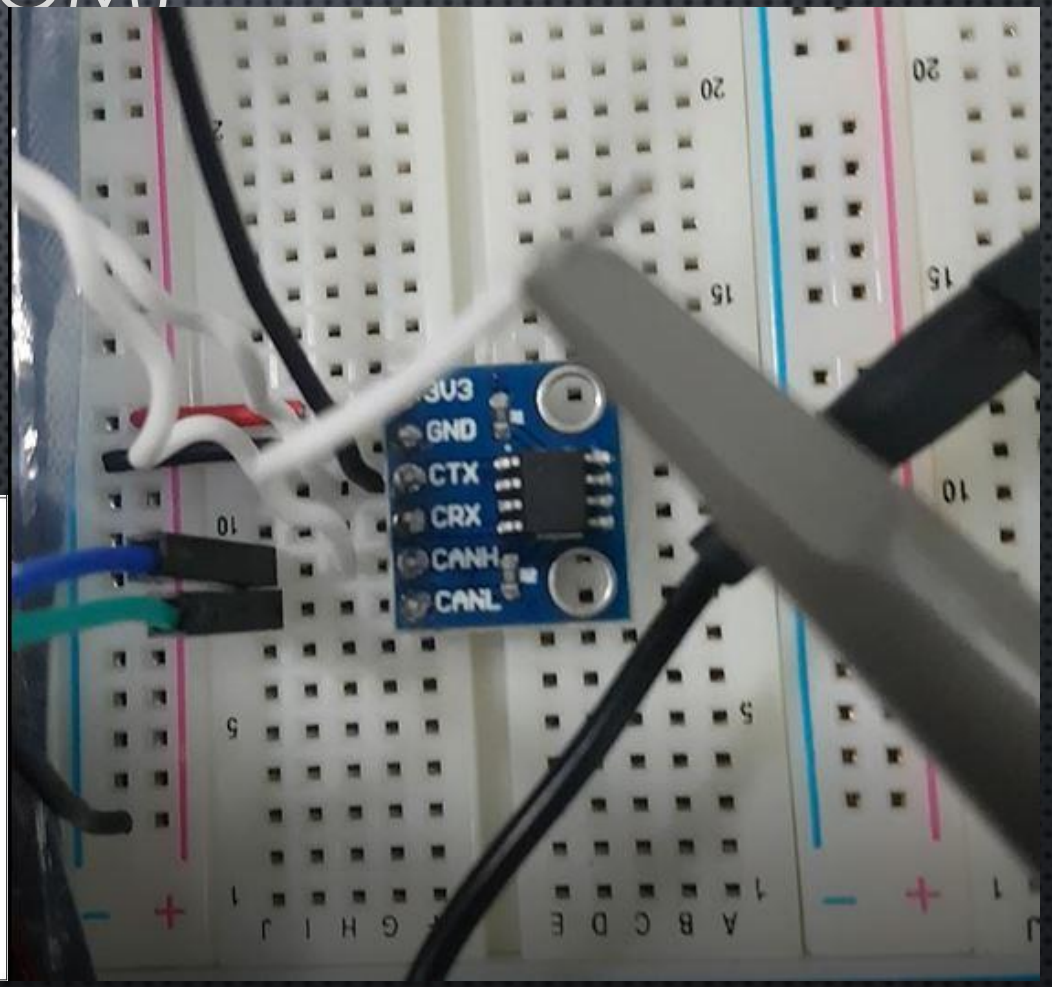
2-0	LEC		Last Error Code
			The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error.
		0	No Error
		1h	Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.
		2h	Form Error: A fixed format part of a received frame has the wrong format.
		3h	Ack Error: The message this CAN Core transmitted was not acknowledged by another node.
		4h	Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
		5h	Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value 0), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		6h	CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).
		7h	No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register reinitializes the LEC bit to 7.



CAN RX TEST (MCU TO COM)

원하는 동작 :
COM에서 보낸 데이터를 MCU에서 Receive해야함.

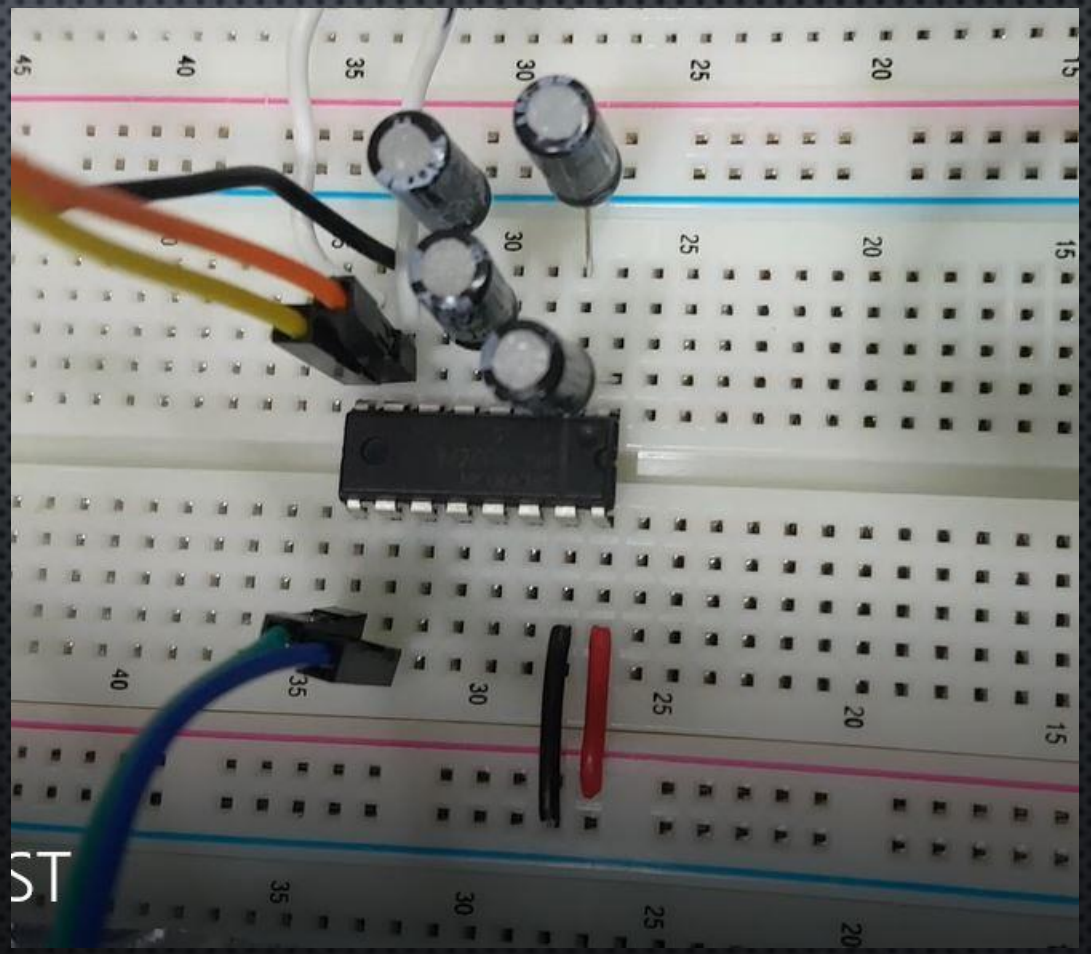
2-0	LEC		Last Error Code The LEC field indicates the type of the last error on the CAN bus. This field will be cleared to 0 when a message has been transferred (reception or transmission) without error.
		0	No Error
		1h	Stuff Error: More than five equal bits in a row have been detected in a part of a received message where this is not allowed.
		2h	Form Error: A fixed format part of a received frame has the wrong format.
		3h	Ack Error: The message this CAN Core transmitted was not acknowledged by another node.
		4h	Bit1 Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value 1), but the monitored bus value was dominant.
		5h	Bit0 Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), the device wanted to send a dominant level (logical value 0), but the monitored bus level was recessive. During Bus-Off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceeding of the Bus-Off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
		6h	CRC Error: In a received message, the CRC check sum was incorrect. (CRC received for an incoming message does not match the calculated CRC for the received data).
		7h	No CAN bus event was detected since the last time when CPU has read the Error and Status Register. Any read access to the Error and Status Register reinitializes the LEC bit to 7.



RS232 TX TEST (MCU TO COM)

원하는 동작 :
MCU에서 보낸 메시지를 COM에서 받음

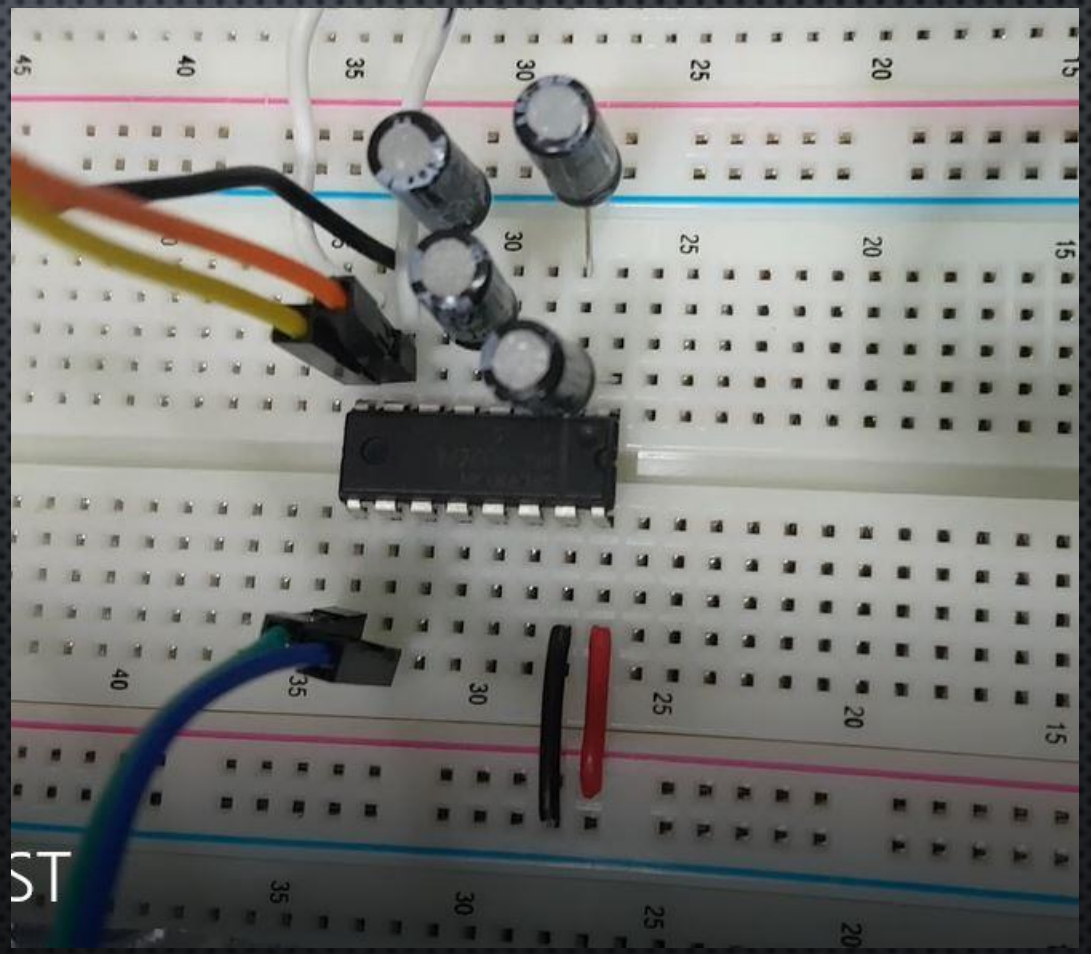
SCI
Baudrate : 115200
Transceiver : Max232



RS232 RX TEST (MCU TO AHRS)

원하는 동작 :
AHRS에서 보낸 데이터를 MCU에서 확인

SCI
Baudrate : 115200
Transceiver : Max232



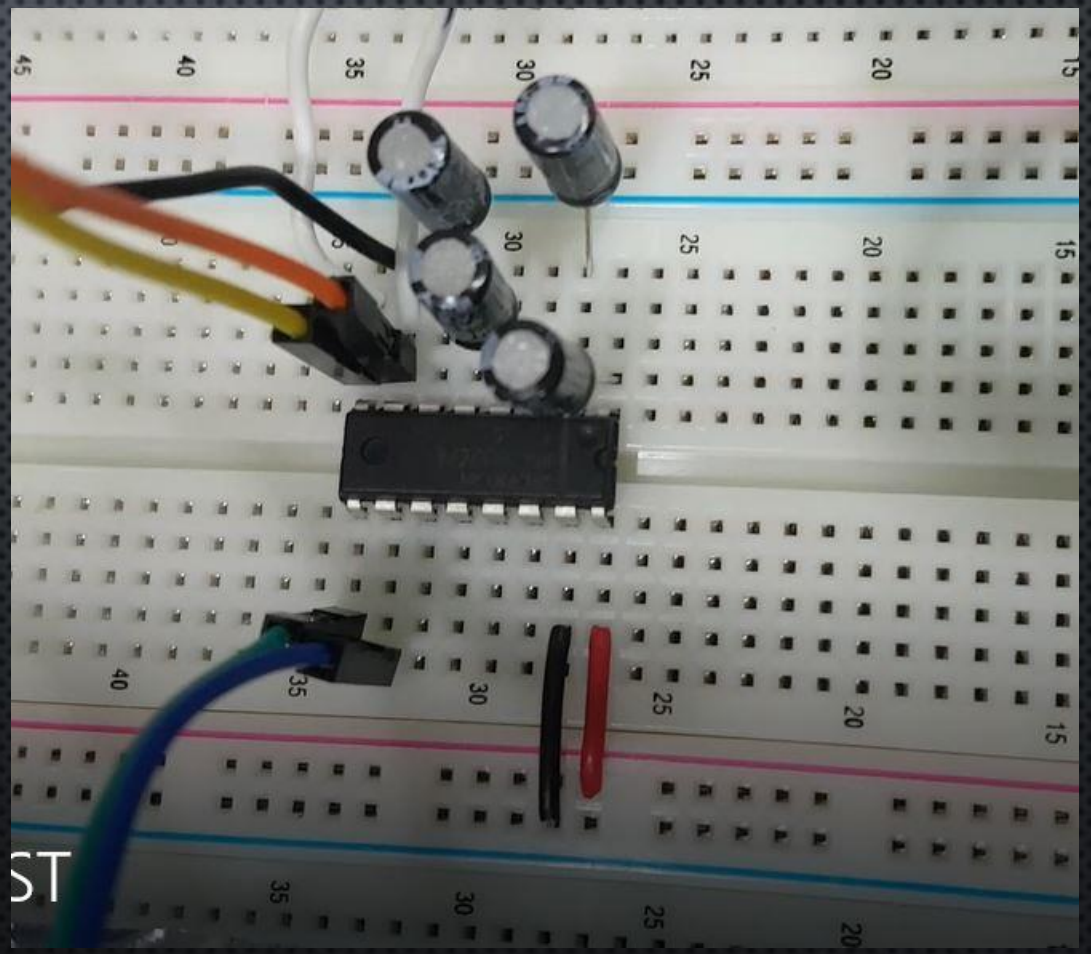
RS232 RX TEST (MCU TO AHRS)

문제점 :

1. MCU 전류 부족 -> DC Jack 해결

2. Max232 IC가 고장남.
   전압값이 승압이 안됨.
   발열이 굉장히 심함.

예상 : Capacitor의 접촉불량, GND의 접촉불량 등 회로 접촉 불량



다음 주 예정

1. MCU에서 데이터를 요청할 때만 전송 (RS-232)
2. 오일러 각도 중 pich or roll 만 전송
3. 통신 회로 구성

표 3-1 MW-AHRS 센서의 오브젝트 요약

Name	Index	Sub-i	Access, Size	Description	Default
ver	1	0	RO, INT32	공급자 ID, 0으로 고정	0
ver	2	0	RO, INT32	제품(AHRS 센서) ID	5001
ver	3	0	RO, INT32	장치 펌웨어 버전	100
ver	4	0	RO, INT32	장치 하드웨어 버전	200
cmd	7	0	WO, INT32	장치에 내려지는 명령 (RS-232 Text 모드에서는 다음 명령 사용 가능: fw, fd, cal, cam, zro, rcd, rst, ver, h, help)	
id	11	0	RW, INT32	장치 ID	1
cb	12	0	RW, INT32	CAN 통신 속도 [Kbps]	1000
sb	13	0	RW, INT32	RS-232 통신 속도 [bps]	115200
gs	15	0	RW, INT32	자이로 센서의 측정 스케일 설정	0 ~ 3
as	16	0	RW, INT32	가속도 센서의 측정 스케일 설정	0 ~ 3
mv	19	0	RW, INT32	자기 센서의 측정값에 대한 분산 설정	0
av	20	0	RW, INT32	가속도 센서의 측정값에 대한 분산 설정	1000
ss	21	0	RW, INT32	RS-232로 동기화 데이터 전송 설정	0
sc	22	0	RW, INT32	CAN으로 동기화 데이터 전송 설정	0
sp	24	0	RW, INT32	동기화 데이터 전송 주기 설정 [ms]	100
st	25	0	RW, INT32	장치에 전원이 투입될 때, RS-232 데이터 전송 타입 결정 (0-Binary, 1-Text)	1
acc	51	1~3	RO, FLOAT	가속도 데이터 전송 (x, y, z) [g]	
gyr	52	1~3	RO, FLOAT	각속도 데이터 전송 ($\omega_x, \omega_y, \omega_z$) [°/s]	
ang	53	1~3	RO, FLOAT	오일러 각도 전송 (ϕ, θ, ψ) [°]	
mag	54	1~3	RO, FLOAT	자기 데이터 전송 (x, y, z) [μT]	
tmp	57	1	RO, FLOAT	온도 전송 [°C]	