230510_summary

0️⃣ todo

데이터테이블 O
하드웨어 O

1️⃣ 리듬캣치

Data Table

	리시브	빠따	주먹	짱	번트	검	스파이크
소지			1	1	1	0	0
약지			1	1	1	0	0
중지			1	1	0	0	0
검지			1	1	0	0	0
엄지			1	0	0	1	0
손날		1					
금성구	1						
엄지			1				

	기본값	힘줬어		리시브	빠따	주먹	짱	번트	검	스파이크
소지	138	191		206	202	202	202	177	123	127
약지	142	177		209	234	234	234	202	142	145
중지										
검지	170	260		164	241	234	241	145	170	164
엄지	117	209		159	174	177	106	100	213	95
손날					2200					
금성구				2000						
엄지										

mpu6050_read.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <arpa/inet.h>
#include <sys/socket.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
#include <fcntl.h>
#include <stdint.h>
#include <time.h>
#include "mpu6050_reg.h"

#define SPI_CHANNEL 0
#define SPI_SPEED 1000000
#define F1_LIMIT 155
#define F2_LIMIT 170
#define F4_LIMIT 195
#define F5_LIMIT 137
#define FORCE_LIMIT 1000

void error_handling(const char *message);

int read_mcp3008_adc(unsigned char adcChannel)
{
    unsigned char buff[3];
    int adcValue = 0;

    buff[0] = 0x06 | ((adcChannel & 0x07) >> 2);
    buff[1] = ((adcChannel & 0x07) << 6);
    buff[2] = 0x00;

    wiringPiSPIDataRW(SPI_CHANNEL, buff, 3);

    adcValue = ((buff[1] & 0x0F) << 8) | buff[2];

    return adcValue;
}

char* hand_mode(int f0,int f1,int f2,int f3, int f4, int f5, int f6, int f7){
        // receive(1)
  if (f6 > FORCE_LIMIT) {
    return "recv";
  }
  // bat(2)
  if (f0 > FORCE_LIMIT) {
    return "bat";
  }
  // rock(3)
  if (f1 > F1_LIMIT && f2 > F2_LIMIT && f4 > F4_LIMIT && f5 > F5_LIMIT && f7 > FORCE_LIMIT) {
    return "rock";
  }
  // jjang(4)
  if (f1 > F1_LIMIT && f2 > F2_LIMIT && f4 > F4_LIMIT && f5 < F5_LIMIT) {
    return "jjang";
  }
  // bunt(5)
  if (f1 > F1_LIMIT && f2 > F2_LIMIT && f4 < F4_LIMIT && f5 < F5_LIMIT) {
    return "bunt";
  }
  // sword(6)
  if (f1 < F1_LIMIT && f2 < F2_LIMIT && f4 < F4_LIMIT && f5 > F5_LIMIT) {
    return "sword";
  }
  // spike(7)
  if (f1 < F1_LIMIT && f2 < F2_LIMIT && f4 < F4_LIMIT && f5 < F5_LIMIT) {
    return "spike";
  }
  else {
    return "hu gu deong !";
  }
}

int main(int argc, char *argv[])
{
    // Initialize wiringPi and SPI
    if (wiringPiSetup() == -1 || wiringPiSPISetup(SPI_CHANNEL, SPI_SPEED) == -1)
    {
        printf("Failed to setup wiringPi or SPI\n");
        return -1;
    }

    // MPU6050 initialization
    int mpu6050_fd;
    char mpu6050_data[14];
```

```
    int16_t ax, ay, az, gx, gy, gz;
    struct timespec sleep_time;

    mpu6050_fd = open("/dev/mpu6050_device", O_RDONLY);
    if (mpu6050_fd < 0) {
        perror("Failed to open mpu6050_device");
        return 1;
    }

    sleep_time.tv_sec = 0;
    sleep_time.tv_nsec = 100000000; // 100 ms

    // socket error
    int serv_sock=socket(PF_INET, SOCK_STREAM, 0);
    if (serv_sock == -1)
        error_handling("socket() error");

    // ip configuration
    struct sockaddr_in serv_addr;
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    serv_addr.sin_port = htons(30020);

    // bind error
    if (bind(serv_sock, (struct sockaddr*) &serv_addr, sizeof(serv_addr)) == -1 )
        error_handling("bind() error");

    // listen error
    if (listen(serv_sock, 5) == -1)
        error_handling("listen() error");

    // accept error
    socklen_t clnt_addr_size = sizeof(serv_addr);
    int clnt_sock=accept(serv_sock, (struct sockaddr*)&serv_addr, &clnt_addr_size);
    if (clnt_sock == -1)
        error_handling("accept() error");

    char message[] = "Hello UE!";
    write(clnt_sock, message, sizeof(message));
    char buffer[10] = "10";
    char send_buffer[10] = "10";
    char* pSend_buffer;
    char end_buffer[10] = "00";
    int b = 0;
    int flexValue0 = 0;
    int flexValue1 = 0;
    int flexValue2 = 0;
    int flexValue3 = 0;
    int flexValue4 = 0;
    int forceValue5 = 0;
    int forceValue6 = 0;
    int forceValue7 = 0;

    while(1) {
        flexValue0 = read_mcp3008_adc(0); // flex_pinky
        flexValue1 = read_mcp3008_adc(1); // flex_ring
        flexValue2 = read_mcp3008_adc(2); // flex_middle
        flexValue3 = read_mcp3008_adc(3); // flex_index
        flexValue4 = read_mcp3008_adc(4); // flex_thumb
        forceValue5 = read_mcp3008_adc(5); // force_thumb
        forceValue6 = read_mcp3008_adc(6); // force_palm
        forceValue7 = read_mcp3008_adc(7); // force_knife

        if (read(mpu6050_fd, mpu6050_data, 14) != 14) {
            perror("Failed to read data");
            close(mpu6050_fd);
            return 1;
        }

        ax = (int16_t)((mpu6050_data[0] << 8) | mpu6050_data[1]);
        ay = (int16_t)((mpu6050_data[2] << 8) | mpu6050_data[3]);
        az = (int16_t)((mpu6050_data[4] << 8) | mpu6050_data[5]);
        gx = (int16_t)((mpu6050_data[8] << 8) | mpu6050_data[9]);
        gy = (int16_t)((mpu6050_data[10] << 8) | mpu6050_data[11]);
        gz = (int16_t)((mpu6050_data[12] << 8) | mpu6050_data[13]);
        pSend_buffer = hand_mode(flexValue0, flexValue1, flexValue2, flexValue3, flexValue4, forceValue5, forceValue6, forceValue7);
            printf("Acceleration: X = %d, Y = %d, Z = %d\n", ax, ay, az);
            printf("Gyroscope: X = %d, Y = %d, Z = %d\n", gx, gy, gz);
      printf("Hand_Mode\n0 : %d\n1 : %d\n2 : %d\n3 : %d\n4 : %d\n5 : %d\n6 : %d\n7 : %d\n", flexValue0, flexValue1, flexValue2, flexVa
            write(clnt_sock, pSend_buffer, strlen(pSend_buffer));
            printf("%s\n", pSend_buffer);
            b++;
        /*if(b > 10){
            write(clnt_sock, end_buffer, strlen(end_buffer));
            break;
        }*/
```

```
            usleep(250000);
    }
    close(clnt_sock);
    close(serv_sock);
    close(mpu6050_fd);
    return 0;
}
void error_handling(const char *message)
{
    fputs(message, stderr);
    fputc('\n', stderr);
    exit(1);
}
```