230426_summary

0️⃣ todo

몽골계약서 o
몽골가는편 x
몽골오는편 x
flex센서회로구성 o
felx센서값확인 o
스트레칭 o
건싸피제출 o
최종발표PPT x
산출물틀작성 x
사람인이력서 o
자소설채용공고 o
카카오브레인패스파인더 x
SK퓨쳐탤런트 o

1️⃣ flex sensor

test0.c

- gcc test0.c -o test0 -lwiringPi -lpaho-mqtt3c

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <MQTTClient.h>

#define CHANNEL 0
#define MCP3008_SPICHANNEL 0
#define MCP3008_SPEED 1000000
#define MAX_PAYLOAD_SIZE 100

char* broker_address = "127.0.0.1";
char* client_id = "ClientPublisher";
int count = 0;

int read_adc(int adc_channel) {
    unsigned char buffer[3];
    buffer[0] = 1;
    buffer[1] = (8 + adc_channel) << 4;
    buffer[2] = 0;
    wiringPiSPIDataRW(MCP3008_SPICHANNEL, buffer, 3);
    int adc_value = ((buffer[1] & 3) << 8) + buffer[2];
    return adc_value;
}

int main() {
    // Initialize WiringPi library and SPI communication
    wiringPiSetup();
    wiringPiSPISetup(MCP3008_SPICHANNEL, MCP3008_SPEED);

    // Initialize MQTT client
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    MQTTClient_deliveryToken token;

    int rc;
    if ((rc = MQTTClient_create(&client, broker_address, client_id, MQTTCLIENT_PERSISTENCE_NONE, NULL)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to create MQTT client, return code %d\n", rc);
        return EXIT_FAILURE;
    }

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect to MQTT broker, return code %d\n", rc);
        return EXIT_FAILURE;
    }
```

```
    // Continuously read ADC values and publish messages
    while(1) {
        // Get current time
        time_t current_time = time(NULL);
        struct tm* time_info = localtime(&current_time);

        // Read ADC value
        int sensor_value = read_adc(CHANNEL);
        printf("sensor : %d\n", sensor_value);

        usleep(300000);

        // If the sensor value is below 50, increment the counter and publish message
        if (sensor_value < 50) {
            count++;
            char payload[MAX_PAYLOAD_SIZE];
            snprintf(payload, MAX_PAYLOAD_SIZE, "Refrigerator use count = %d\n %d/%d %d:%d:%d", count, time_info->tm_mon, time_info->t
            printf("%s\n", payload);
            pubmsg.payload = payload;
            pubmsg.payloadlen = strlen(payload);
            pubmsg.qos = 0;
            pubmsg.retained = 0;
            MQTTClient_publishMessage(client, "bme", &pubmsg, &token);
            MQTTClient_waitForCompletion(client, token, 2000);
            usleep(2000000);
        }
    }

    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
    return EXIT_SUCCESS;
}
```

test1.c

- gcc test1.c -o test1 -lwiringPi

```
#include <stdio.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>

#define SPI_CHANNEL 0
#define SPI_SPEED 1000000

int main(void)
{
    int spi_result;
    unsigned char spi_data[3];

    if(wiringPiSetup() == -1)
    {
        printf("wiringPiSetup failed.\n");
        return -1;
    }

    if(wiringPiSPISetup(SPI_CHANNEL, SPI_SPEED) == -1)
    {
        printf("wiringPiSPISetup failed.\n");
        return -1;
    }

    while(1)
    {
        spi_data[0] = 0b00000001;
        spi_data[1] = 0b10000000;
        spi_data[2] = 0;

        spi_result = wiringPiSPIDataRW(SPI_CHANNEL, spi_data, 3);

        if(spi_result == -1)
        {
            printf("wiringPiSPIDataRW failed.\n");
            return -1;
        }

        int adc_result = ((spi_data[1] & 3) << 8) + spi_data[2];

        printf("Flex sensor value: %d\n", adc_result);

        delay(500);
    }
```

```
    return 0;
}
```

test2.c

- gcc test2.c -o test2 -lwiringPi -lpaho-mqtt3c

```
#include <stdio.h>
#include <stdlib.h>
#include <wiringPi.h>
#include <wiringPiSPI.h>
#include <unistd.h>
#include <time.h>
#include <string.h>
#include <MQTTClient.h>

#define CHANNEL 0
#define MCP3008_SPICHANNEL 0
#define MCP3008_SPEED 1000000
#define MAX_PAYLOAD_SIZE 100

char* broker_address = "127.0.0.1";
char* client_id = "ClientPublisher";
int count = 0;

int read_adc(int adc_channel) {
    unsigned char buffer[3];
    buffer[0] = 1;
    buffer[1] = (8 + adc_channel) << 4;
    buffer[2] = 0;
    wiringPiSPIDataRW(MCP3008_SPICHANNEL, buffer, 3);
    int adc_value = ((buffer[1] & 3) << 8) + buffer[2];
    return adc_value;
}

int main() {
    // Initialize WiringPi library and SPI communication
    wiringPiSetup();
    wiringPiSPISetup(MCP3008_SPICHANNEL, MCP3008_SPEED);

    // Initialize MQTT client
    MQTTClient client;
    MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;
    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    MQTTClient_deliveryToken token;

    int rc;
    if ((rc = MQTTClient_create(&client, broker_address, client_id, MQTTCLIENT_PERSISTENCE_NONE, NULL)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to create MQTT client, return code %d\n", rc);
        return EXIT_FAILURE;
    }

    if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
        printf("Failed to connect to MQTT broker, return code %d\n", rc);
        return EXIT_FAILURE;
    }

    // Continuously read ADC values and publish messages
    while(1) {
        // Get current time
        time_t current_time = time(NULL);
        struct tm* time_info = localtime(&current_time);

        // Read ADC value
        int sensor_value = read_adc(CHANNEL);
        printf("sensor value : %d\n", sensor_value);

        usleep(600000);
    }

    MQTTClient_disconnect(client, 10000);
    MQTTClient_destroy(&client);
    return EXIT_SUCCESS;
}
```