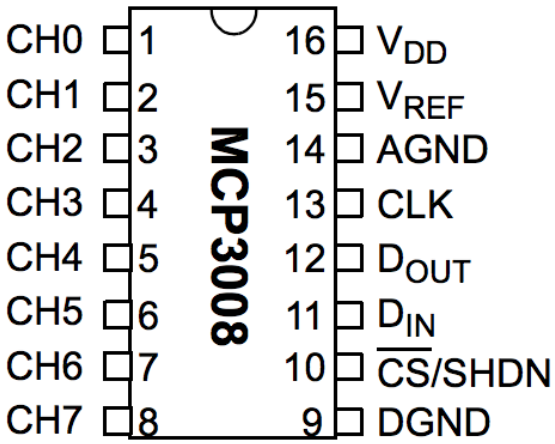
230425_summary

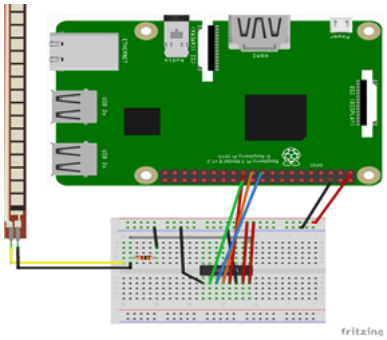
0️⃣ todo

꼬맨틀 o
꼬들 o
몽골뱅기 o
취특 o
flex 1/3 o

1️⃣ Flex Sensor & RPi

준비





```
# gcc 설치
$ sudo apt-get install gcc

# WiringPi 설치
$ git clone https://github.com/WiringPi/WiringPi.git
$ cd WiringPi/
$ ./build

# MQTTClient.h 헤더 파일 설치
$ sudo apt-get install libpaho-mqtt-dev

# MQTTClient 라이브러리 설치확인 + 설치
$ sudo apt list libmqttclient*
$ sudo apt install libmqttclient-dev

# 빌드
$ gcc test.c -lwiringPi -lMQTTClient -o test
$ gcc test.c -lwiringPi -lMQTTClient -L/usr/local/lib -o test

# SPI on 하기
$ sudo nano /boot/config.txt
중간에 주석 부분 풀어주기 (아래 사진 참고)
```

```
# Uncomment some or all of these to enable the optional hardware interfaces
dtparam=i2c_arm=on
dtparam=i2s=on
dtparam=spi=on
```

test.c

```
#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;
}
```

2 3 4 5 6 7 8 9 10