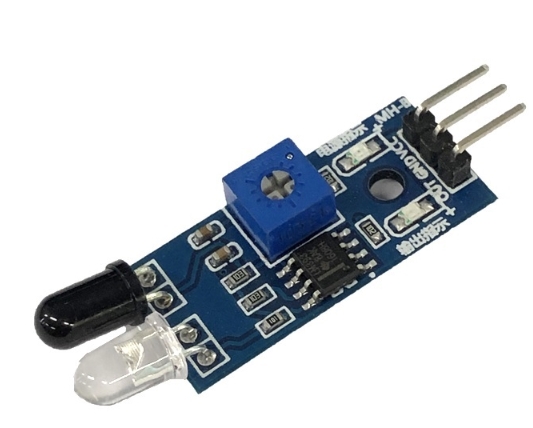
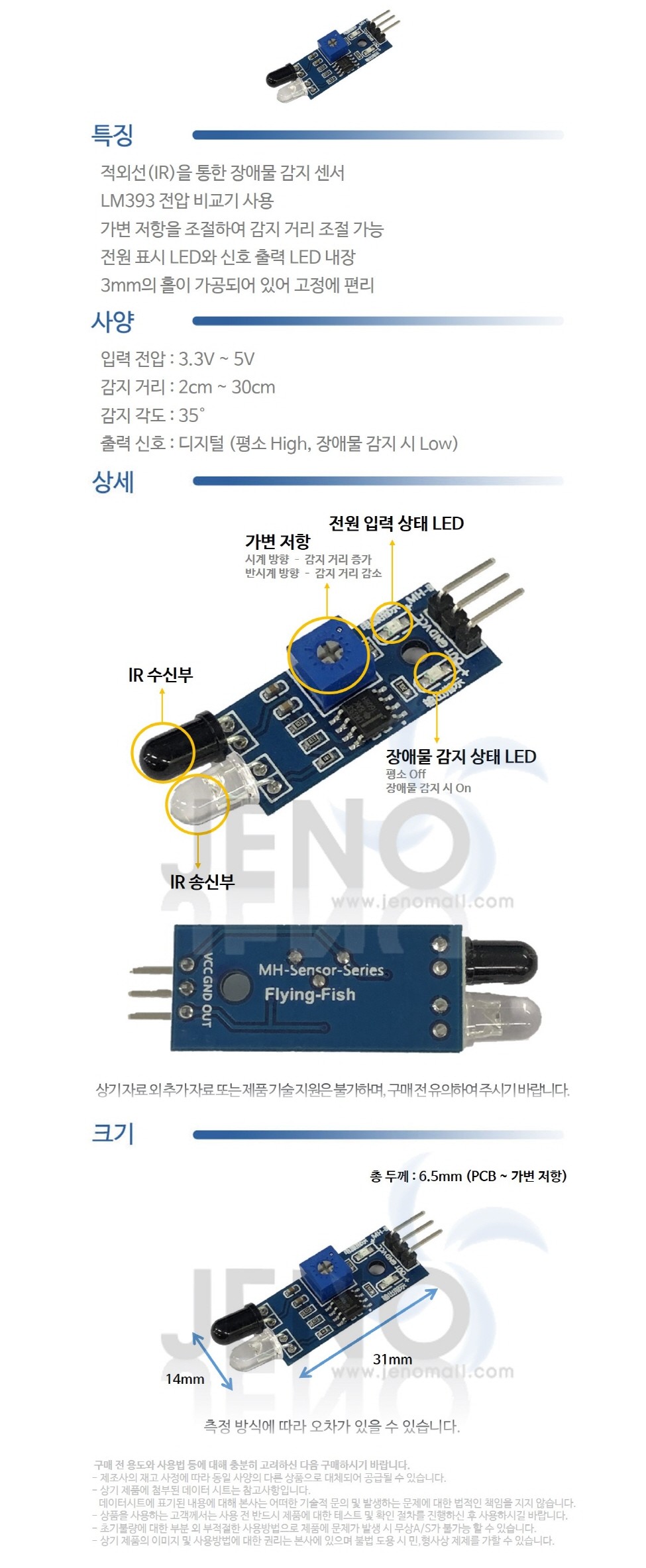
**5조 MDP프로젝트 적외선센서(IR센서)에 대한 정보**



적외선(IR)을 통한 장애물 감지센서, LM393 전압 비교기 사용





stm32로 IR센서를 동작하는 방법

적외선 센서에서 값을 읽어서 일정 거리 이하로 잡히면 (=다트가 벽에 붙었다면) 일정 점수를 변수에 저장.

적외선 센서에있는 vcc와 gnd는 브레드 보드로 한데묶어 연결한다. (vcc는 5V 를 사용.) out핀은 모두 PB0,1,2,4,5,6,7,8,9,10을 사용한다

- 핀 설정

ADC1에 IN1 채널을 활성화 한다.

https://www.os4all.com/88 - 참조

https://www.mathaelectronics.com/interfacing-ir-sensor-with-stm32-discovery-board/ - 참조2 ( 똑같은 센서 )

예제 코드 )

#include *"stm32f4xx.h"*

#include *"stm32f4\_discovery.h"*

GPIO\_InitTypeDef GPIO\_InitLed;

EXTI\_InitTypeDef EXTI\_InitSensor;

NVIC\_InitTypeDef NVIC\_InitSensor;

void Delay(\_\_IO uint32\_t nCount)

{

**while**(nCount--)

{

}

}

int main(void)

{

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOD, ENABLE);

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);

RCC\_AHB2PeriphClockCmd(RCC\_APB2Periph\_SYSCFG, ENABLE);

GPIO\_InitLed.GPIO\_Pin = GPIO\_Pin\_0 | GPIO\_Pin\_1;

GPIO\_InitLed.GPIO\_Mode = GPIO\_Mode\_OUT;

GPIO\_InitLed.GPIO\_OType = GPIO\_OType\_PP;

GPIO\_InitLed.GPIO\_PuPd = GPIO\_PuPd\_NOPULL;

GPIO\_InitLed.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOD, &GPIO\_InitLed);

GPIO\_InitLed.GPIO\_Pin = GPIO\_Pin\_1;

GPIO\_InitLed.GPIO\_Mode = GPIO\_Mode\_IN;

GPIO\_InitLed.GPIO\_Mode = GPIO\_OType\_PP;

GPIO\_InitLed.GPIO\_PuPd = GPIO\_PuPd\_DOWN;

GPIO\_InitLed.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_Init(GPIOA, &GPIO\_InitLed);

SYSCFG\_EXTILineConfig(EXTI\_PortSourceGPIOA, EXTI\_PinSource1);

EXTI\_InitSensor.EXTI\_Line = EXTI\_Line1;

EXTI\_InitSensor.EXTI\_Mode = EXTI\_Mode\_Interrupt;

EXTI\_InitSensor.EXTI\_Trigger = EXTI\_Trigger\_Rising;

EXTI\_InitSensor.EXTI\_LineCmd = ENABLE;

EXTI\_Init(&EXTI\_InitSensor);

NVIC\_PriorityGroupConfig(NVIC\_PriorityGroup\_1);

NVIC\_InitSensor.NVIC\_IRQChannel = EXTI1\_IRQn;

NVIC\_InitSensor.NVIC\_IRQChannelPreemptionPriority = 0;

NVIC\_InitSensor.NVIC\_IRQChannelSubPriority = 0;

NVIC\_InitSensor.NVIC\_IRQChannelCmd = ENABLE;

NVIC\_Init(&NVIC\_InitSensor);

**while** (1)

{

GPIO\_WriteBit(GPIOD, GPIO\_Pin\_0, Bit\_SET);

Delay(16800000);

GPIO\_WriteBit(GPIOD, GPIO\_Pin\_0, Bit\_RESET);

Delay(16800000);

}

}

void EXTI1\_IRQHandler(void)

{

**if**(EXTI\_GetITStatus(EXTI\_Line1) != RESET)

{

GPIO\_WriteBit(GPIOD, GPIO\_Pin\_1, Bit\_SET);

Delay(33600000);

GPIO\_WriteBit(GPIOD, GPIO\_Pin\_1, Bit\_RESET);

EXTI\_ClearITPendingBit(EXTI\_Line1);

}

}

- 이 코드는 STM32F4 마이크로컨트롤러를 사용하여 LED 및 외부 인터럽트 센서를 제어하는 C 프로그램입니다. 아래는 코드의 주요 부분을 해석한 것입니다:

1. 코드는 STM32F4 라이브러리와 관련 헤더 파일을 포함합니다.

2. GPIO\_InitTypeDef, EXTI\_InitTypeDef, 및 NVIC\_InitTypeDef 구조체를 사용하여 GPIO 초기화, 외부 인터럽트 초기화 및 NVIC 초기화를 위한 구조체를 정의합니다.

3. Delay 함수는 주어진 시간 동안 루프를 실행하여 딜레이를 생성하는 데 사용됩니다.

4. main 함수는 프로그램의 진입점입니다. 주요 단계는 다음과 같습니다:

GPIOD와 GPIOA의 클럭을 활성화합니다.

SYSCFG 및 EXTI를 위한 클럭을 활성화합니다.

GPIOD의 0 및 1번 핀을 출력 모드로 설정하여 두 개의 LED를 초기화합니다.

GPIOA의 1번 핀을 입력 모드로 설정하여 외부 센서를 초기화합니다.

EXTI1의 인터럽트 라인을 GPIOA의 1번 핀에 매핑합니다.

EXTI1을 인터럽트 모드로 설정하고, 상승 엣지 트리거를 사용하여 활성화합니다.

NVIC 그룹 우선순위를 그룹 1로 설정하고, EXTI1 인터럽트를 활성화합니다.

무한 루프 내에서 LED를 깜박이는 코드를 실행합니다.

EXTI1\_IRQHandler 함수는 EXTI1의 인터럽트 핸들러입니다. EXTI1 인터럽트가 발생하면 호출됩니다. 인터럽트가 발생하면 1번 핀의 LED를 켜고 약 3.36초 동안 대기한 다음 끕니다. 그런 다음 EXTI 인터럽트를 지웁니다.

코드에서 주의할 점:

GPIO\_InitLed.GPIO\_Mode 설정 중에 중복된 GPIO\_Mode 설정이 있습니다. 두 번째 설정은 잘못되었습니다. GPIO\_Mode 설정을 수정해야 합니다.

EXTI1\_IRQHandler 함수에서는 딜레이 함수가 호출되기 때문에 인터럽트가 발생할 때마다 딜레이가 실행됩니다. 이러한 딜레이가 인터럽트 핸들러 내에서 사용되는 것은 권장되지 않으며, 인터럽트 핸들러는 가능한 한 빨리 실행되어야 합니다. 딜레이를 인터럽트 핸들러 외부로 이동시키는 것이 좋습니다.