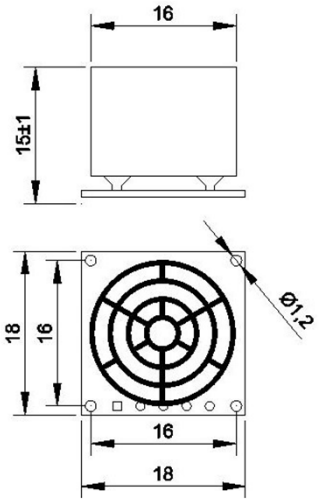
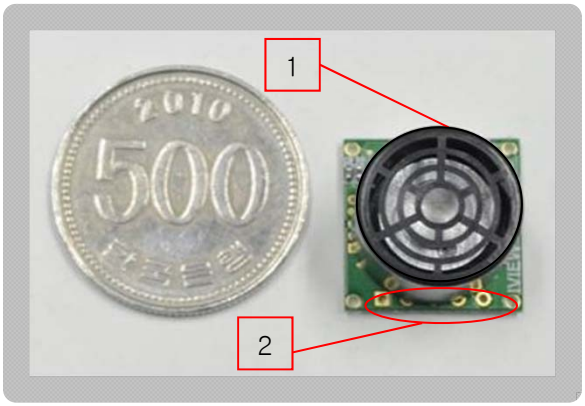
IVIEW Co., LTD

IUM–D16

IUM–D16 은 초소형 싱글 타입 초음파 센서 모듈입니다. 자동으로 온도 보상회로를 초음파의 속도 보정으로 보다 정확한 거리측정이 가능합니다. 초고가 부품사용과 디지털처리방식으로 부품을 현저히 줄여 불량 가능성을 최소화 하였습니다. 다양한 모드지원으로 아두이노 등 다양한 어플리케이션에 적용 가능합니다.

Mechanical Dimension





1 Ultrasonic Transmitter & Receiver 2 In-Out Port

Description

최소 27cm ~ 최대 6.5M 에 이르는 거리의 측정이 가능하고 최초로 초음파의 온도에 따른 속도 변화량까지 보정이 가능합니다. 현 아두이노 관련 초음파 모듈과 완벽 호환하고 간단한 커멘드로 모드변환이 이뤄집니다.
다양한 모드에서 정밀측정과 움직이는 물체감지와 같은 기능을 구현할 수 있습니다.
신뢰성이 우수한 부품사용으로 높은 퀄리티를 자랑합니다.
커멘드 변경으로 송신용, 수신용, 송수신용으로 사용할 수 있고 최대 15개 까지 데이지체인으로 연결하여 다양한 array 구성이 가능합니다.

Electrical/Physical Characteristics

Category	Item	Specification	Unit	Conditions
Electrical	Input Voltage	3.3 ~ 12	V	@DC Typ. 5V
	Current consumption	Typ. 8	mA	@Max. (13mA : Pulse 50ms)
	Frequency	40	kHz	
	Output	Trig out & UART		@ Trig. Pulse Width 10us @3.3~5V(UART)
Physical	Measuring Distance	0.27~6	m	@Trigger, Free Run Mode @Object Detector Mode(6m/6.5m)
	Beam width	70±15	°	
	Dimension	18X18X15	mm^3	

초소형 싱글타입 초음파 거리측정/물체감지 모듈(UART 방식)

Model: IUM-D16

❖ 특징

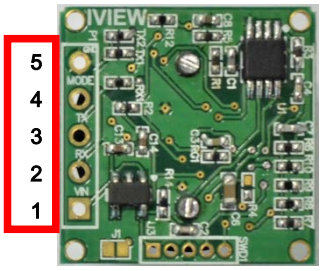
- 초소형 PCB 에 **싱글타입** 초음파 측정모듈 IUM-D16 은 음파를 이용한 비 접촉 방식으로 온도보상과 오류보정을 통해 최대거리 600cm 까지 거리측정이 가능하며, 4 가지의 동작모드가 있어 사용하기에 더욱 편리한 환경을 제공한다.
- 초음파 **모듈은 3 가지 형태의 모듈기능**을 커맨드에 의해 사용 가능하다.
  - **송신기능**: BURST 출력 커맨드를 사용하여 40KHz BURST 펄스를 출력한다.
  - **수신기능**: 송신모듈에서 BURST 펄스를 보내면, 반사된 에코신호를 받을 수 있는 수신모듈 기능이다.
  - **송수신 겸용**: 기존 방식으로 자체에서 발사된 BURST 펄스와 반사된 에코신호를 수신한다.
- 복잡한 아날로그 회로를 디지털화하여 외부부품을 최소화했으며, 많은 부품으로 인한 불량요인을 사전에 제거했고, 입력전압은 DC3.3 ~ 12V 까지 사용할 수 있어서, 사용자의 실수로 과전압을 공급해서 파괴되는 문제를 보완했다.
- UART 통신은 8BIT, 1STOP 및 패리티 비트가 없는 표준 TTL 레벨 UART 형식이며, 전원공급 시 전송속도는 항상 9600Baud 로 사용되고, 원하는 경우 19200,38400,115200baud 전송속도를 커맨드로 바꿀 수 있다.
- 최대 15 개의 모듈에 각각의 ID 를부여해서 RX/TX 통신포트 공통라인에 함께 연결할 수 있고 각종제어용 마이컴, **아두이노, 초음파 응용제품** 등에서 초음파센서 모듈로 사용하기에 편리한 구조로 되어있다.
- 4 가지의 동작모드를 사용자 용도에 맞게 선택하여 사용 가능하며 설정 값은 메모리에 저장된다.
  - **External Trigger**: 외부 Trigger 에 의한 계측방식이다. (출하 시 기본 모드이다)
  - **Command Mode**: UART 를 통하여 계측명령을 전송해야만 계측정보 등을 통보 받는 방식이며, 다수의 초음파센서 모듈을 접속하여 사용시 유용한 모드이다.
    (최대 15 개의 모듈을 RX/TX 공통으로 접속한다. TX 단자는 OPEN DRAIN 방식으로 Wired OR 가 가능하다)
  - **Free Run**: 0.5Sec 주기로 자체 Trigger 에 의한 계측방식으로 계측정보를 주기적으로 얻고자 할 때 유용한 모드이다.
  - **Object Detector**: 물체감지 기능으로 정해진 범위 내 물체나 사람이 근접 시 Trig/Out 단자를 통한 경보출력과 동시에 근접거리 정보를 UART 통신을 통해 전달된다.
  - ★ 다수의 모듈을 RX/TX 라인 공통으로 사용하기 위해서는 **Command Mode** 를 사용해야 한다.
  - ★ Trigger, 커맨드방식인 경우 측정주기는 최소 100ms 이상을 권장한다. (내부계측시간: ~70ms)

❖ 사양

계측정보 출력방법		Trig/Out: 물체감지 경보출력 (High 출력)
		TX: UART (3.3 ~ 5V 대응)
초음파 사용주파수		40KHz
Trigger 펄스 폭		10μs
사용전압		DC 3.3 ~ 12V (권장전압: 5V)
최대측정 거리	일반모드	600cm 이내
	물체감지	650cm 이내
최소측정 거리		27cm
지향성		70±15°
소비전류		8mA(Typical)~ 13mA(50ms 계측구간)
크기		18x18x15mm

IUM-D16

● 커넥터 연결방법



커넥터 접속단자	
1	DC+3.3 ~ 12V 전원 입력단자
2	RX(In): 커맨드입력
3	TX(Out): 정보출력
4	Trig/Out: 입출력 겸용 트리거 방식: 입력 펄스(10us) 물체감지기능: 경보출력 단자
5	GND: 공통 GND

녹색 LED:
1. 전원 공급 시 LED 표시는 ID 수 만큼 점멸한다.
2. 음파펄스 송신, 에코신호 수신, 저장된 계측 정보 요구 시 및 물체감지 모드에서 점멸한다.

❖ 통신 프로토콜

● 커맨드 표 (HEX 표시)

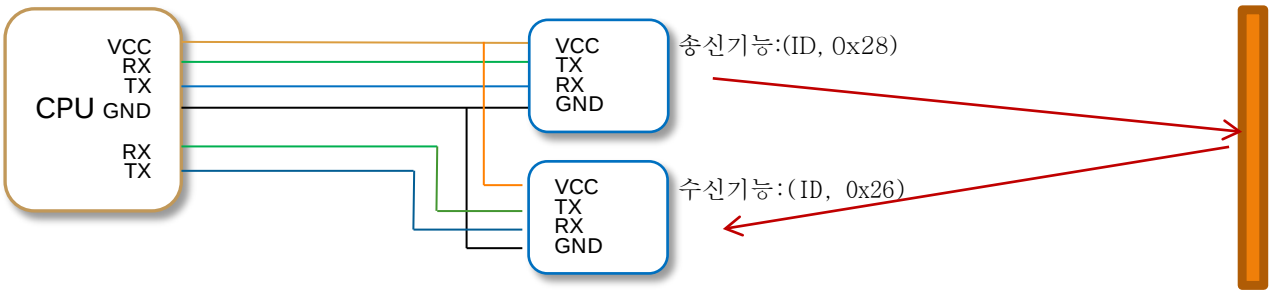
커맨드	ID:0 전체 명령수신	설명(동작/응답)	
0x20	가능	단위: cm	음파펄스를 출력하고 에코신호를 계측한다. 측정값은 TX 로 보내지 않는다.
0x21	가능	단위: in	
0x22	X	단위: cm	음파펄스를 출력하고 에코신호를 계측한 후 측정값을 TX 로 보낸다. (자동응답)
0x23	X	단위: in	
0x24	가능	단위: cm	**수신**: 음파펄스는 보내지 않고 에코신호를 계측한다. 측정값은 TX 로 보내지 않는다.
0x25	가능	단위: in	
0x26	X	단위: cm	**수신**: 음파펄스는 보내지 않고, 다른 초음파 모듈에서 보내는 펄스의 에코신호를 계측한 후 측정값을 TX 로 보낸다. (펄스와의 동기는 사용자가 맞춰야 한다.)
0x27	X	단위: in	
0x28	X	**송신**: 40KHz 음파펄스를 출력한다. 계측은 하지 않고 음파만 출력된다. *송신모듈 기능	
0x29	공통버스 송수신자동	단위: cm	**송신, 수신(자동)**: 공통버스에서 ID 로 지정된 모듈은 40KHz 음파펄스를 출력하고, 다른 초음파 모듈(들)은 즉시 자동 계측하여 값을 저장한 후 요청 시 출력한다.
0x2A		단위: in	
0x2B	X	계측 후 저장된 거리데이터를 갖고 온다. 가장 최근에 계측된 정보를 TX 를 통해 보낸다.	
0x2D	X	F/W 버전을 갖고 온다. "(V 1.0)" ~ "(V99.9)" 등이 출력된다.	
0x30	가능	모드설정 저장됨 ACK 출력	전원 공급 후 이 명령을 통해 외부트리거 모드로 설정된다. (출하 시 기본모드)
0x31	가능		전원 공급 후 이 명령을 통해 커맨드 모드로 설정한다. *1:N 통신 가능
0x32	가능		전원 공급 후 이 명령을 통해 Free Run(자동트리거) 모드로 설정한다.
0x33	가능		전원 공급 후 이 명령을 통해 물체감지 경보기능으로 설정된다.
0x35	X	현재 설정된 모드 커맨드 값이 리턴 된다. "(MODE0)" ~ "(MODE3)" *0x30 은 '0' 이 된다. 모드 설정명령을 보낼 때도 모드변경 후 ACK 신호로 설정된 모드 커맨드 값이 리턴 된다.	
0x37	가능	통신속도 설정	UART 통신속도를 19200 으로 변경한다. 전원 공급 시 통신속도는 항상 9600 이다. UART 통신속도 변경 시는 초기에 변경한 후 사용한다.
0x38	가능		UART 통신속도를 38400 으로 변경한다.
0x39	가능		UART 통신속도를 115200 으로 변경한다.
0xAA 0x55	X	ID 변경명령 ID= 1~15 저장됨 ACK 출력	모듈의 ID(1~15)값을 변경할 때 사용된다. ID: 0 으로는 변경할 수 없다. **(예)모듈의 ID: 1 를 ID: 5 로 바꿀 때 명령: "0xAA, 0x55, 1, 5" 4 바이트**를 전송하면 OLD ID: 1 를 NEW ID: 5 로 변경할 수 있다. ***디폴트 ID 값은 1 이다.** ID 변경 후 ACK 신호로 새로 변경된 ID 값이 리턴 된다. "(ID:XX)"

☞ 모듈의 명령체계는 (ID, Command)로 2 바이트 구조이다. 단, ID 변경 명령만 4 바이트로 구성된다.
변경된 모듈의 **ID 값**과 **동작모드**는 내부 메모리에 저장되고 전원을 꺼도 지워지지 않는다.

1. 모듈에 명령을 보낼 때는 각각의 모듈 ID 를 지정하고 명령을 보낸다.
모듈전체에 보내는 명령인 경우, ID 값을 0 으로 보낸다. ID 를 0 으로 보내는 경우는 복수로 접속된 모듈들의 설정과 동작 등을 일괄변경 하는 경우이다.

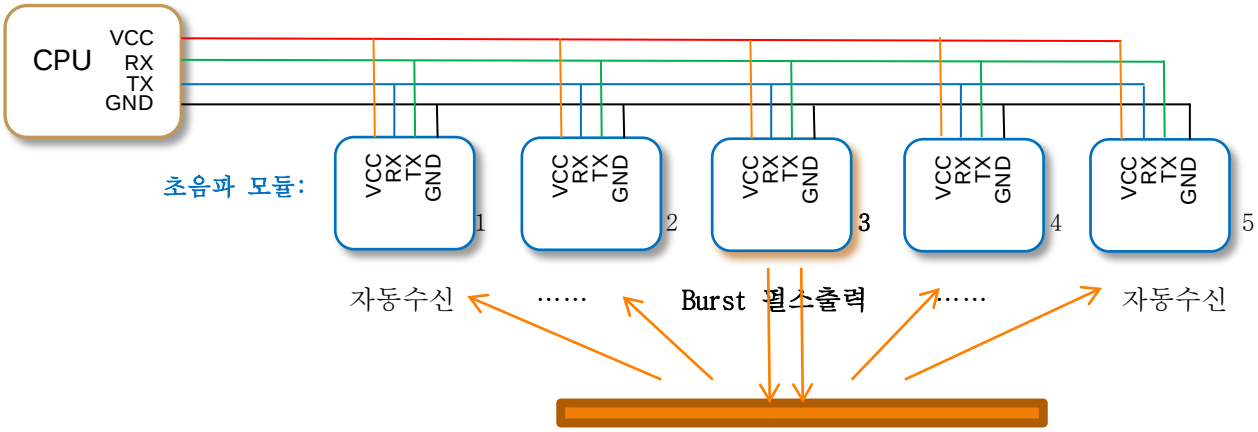
2. ID 변경 후 전원을 새로 공급하면 LED 표시는 ID 수 만큼 점멸한다.
3. 명령 0x20, 0x21 은 계측은 하지만 계측정보를 TX 로 보내지 않고 저장한 후, 0x2B 명령을 받으면 그 때 최근 계측정보를 TX 로 보낸다.
4. 명령 0x22, 0x23 은 자동으로 계측을 진행하고, 계측이 끝난 즉시 계측결과를 TX 로 출력한다.
5. 명령 0x26, 0x27 은 수신모듈로 사용하기 위한 목적으로 BURST 펄스는 출력하지 않고 계측한다.
   BURST 펄스는 송신용 모듈에서 명령 0x28 를 사용해야 하며 송수신 모듈간의 동기는 제어용 마이컴에서 맞춰야 한다.

☞ CPU 에서 Dual UART 를 사용하여 송수신하는 방법 (펌웨어에서 송수신간 동기를 맞춰야 한다.)



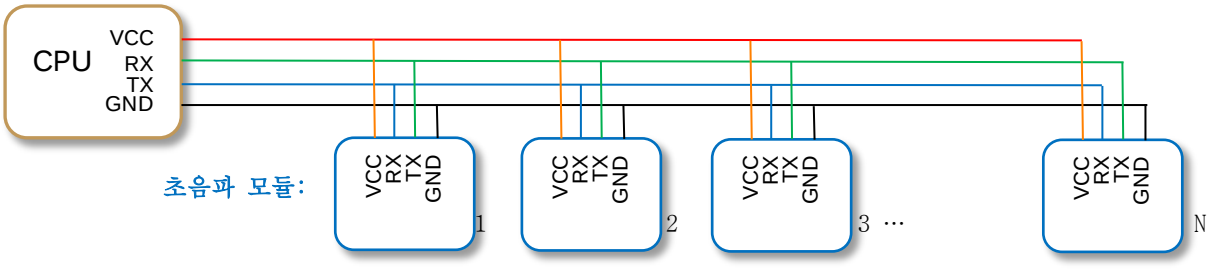
6. 명령(0x29, 0x2A)는 공통버스에서 ID 값과 같은 모듈은(송신용으로 지정) Burst Pulse 를 출력하고 나머지 모듈(들)은 즉시 이 펄스의 에코 신호를 수신하여 계측한 후 저장한다.
   즉, 같은 역할을 하는 수신명령(ID, 0x26(0x27)) 2 바이트와 계측 값 요청명령(ID, 0x2B) 2 바이트(총 4 바이트) 명령을 ID 포함 2 바이트 명령으로 가능하고 공통버스에서는 다른 모듈에서 보낸 Burst 펄스의 에코신호를 수신하여 전체가 한꺼번에 계측할 수 있는 편리함과 송수신간의 동기오차를 줄여 주는 효과도 있다.

☞ CPU 에서 UART 를 공통버스로 구성하여 송수신하는 방법 (모듈간 자동 송수신 및 자동 동기기능)
ID3 을 송신용으로 지정하고 사용한 경우설명: TX_OUT(0x03, 0x29)을 모듈에 보내면 ID3 은 송신 펄스를 출력하고, 나머지 모듈들(1,2,4,5 번)은 다른 명령을 보내지 않아도 자동으로 에코신호를 수신하고 계측 값을 저장한다.
각각의 모듈에서 저장된 계측 값을 갖고 오기 위해선 TX_OUT(ID, 0x2B)명령을 보내서 계측정보를 갖고 온다.



★송수신 기능을 분리하고 사용하는 경우 모듈의 위치, 방향 등에서 계측된 값은 많은 편차가 생길 수 있다.

7. 최대 15 개의 모듈을 사용하려면 **커맨드 모드**를 사용해야 하며, UART RX/TX 를 공통 버스로 구성해야 한다.



★공통버스를 구성하는 경우 케이블 길이는 가능하면 짧게 해야 한다. (4 심 AWG24 번 케이블 사용시 최대 10m 이내)

● 데이터 출력 (ASCII Code): 7Byte

0	1	2	3	4	5	6
STX 거리: '[' 물체: '{' 기타: '('	거리 데이터 (정수부: 3Byte)			점	소수부	ETX 거리: ']' 물체: '}' 기타: ')'
거리데이터 출력 보기: 380.7Cm						
'['	'3'	'8'	'0'	'.'	'7'	']'
거리데이터 출력 보기: 030.5Cm, 002.0Cm *앞자리가 '0' 인 경우 스페이스 문자로 채워짐						
'['	' '	'3'	'0'	'.'	'5'	']'
'['	' '	' '	'2'	'.'	'0'	']'
물체감지 모드에서 기준거리 출력 보기: 420.5Cm 인경우 STX '{' , ETX '}' 로 구분된다						
'{'	'4'	'2'	'0'	'.'	'5'	'}'
물체감지 모드에서 기준거리 설정 시 계측범위를 벗어나면 "{SPACE}" 가 출력됨						
'{'	'S'	'P'	'A'	'C'	'E'	'}'
물체감지 모드에서 기준거리 설정 시 계측범위가 너무 가까우면 "{CLOSE}" 가 출력됨						
'{'	'C'	'L'	'O'	'S'	'E'	'}'
물체감지 모드에서 기준거리 계측 완료 시 "{START}" 가 출력됨						
'{'	'S'	'T'	'A'	'R'	'T'	'}'
물체감지 모드에서 물체 근접 시 근접거리가 출력 됨: 220.3Cm "{220.3}"						
'{'	'2'	'2'	'0'	'.'	'3'	'}'
초음파 모듈의 펌웨어 버전 출력 예: STX '(' , ETX ')' 로 구분되고 "(V 1.0)" ~ "(V99.9)" 를 출력						
'('	'V'	' '	'1'	'.'	'0'	')'
모드변경 명령이 성공하면 ACK 를 출력한다: "(MODE#)" (#=변경된 모드 번호: 0~3), '0..3' = 0x30~0x33 단, 모드변경 후 응답은 항상 ID 번호가 1 인 모듈만 응답한다. (다수의 모듈이 연결된 경우도 동일하다.)						
'('	'M'	'O'	'D'	'E'	'#'	')'
명령(0xAA,0x55,ID,XX)로 ID 변경시 성공하면 ACK 를 출력한다: "(ID:XX)" (XX=새로 변경된 ID 값)						
'('	'I'	'D'	':'	'X'	'X'	')'

✶ 목표물이 계측범위를 벗어난 경우에 UART 거리데이터는 "[999.0]"가 출력된다.

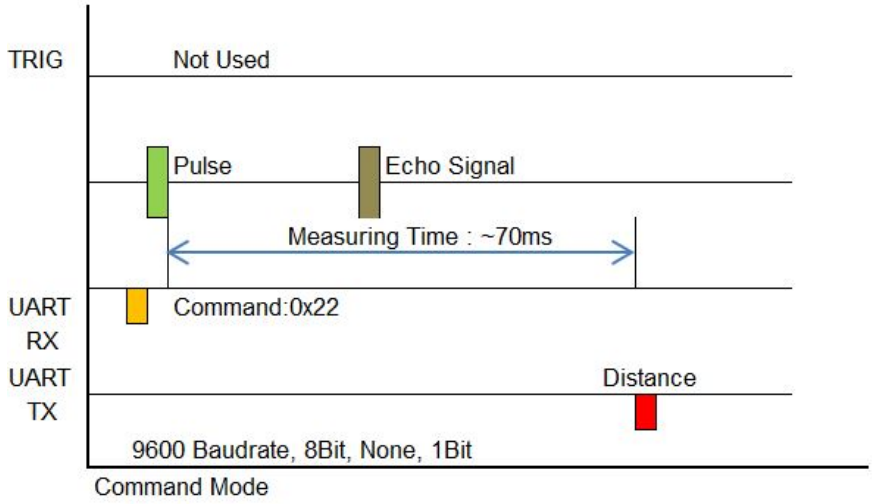
✶ 물체감지 기능에서 기준거리 측정 시 목표물이 너무 가까우면 UART 거리데이터는 "{CLOSE}"가 출력된다.

✶ 싱글타입 초음파센서모듈은 송신 및 수신기능으로 단일센서를 사용하기 때문에 최소 범위는 송신과 수신을 2개의 센서로 따로 구성하여 사용하는 센서보다 최소거리가 높고, 최소거리 이내에서의 계측정보는 부정확하기 때문에 사용하지 않는다.

❖ 동작 모드 별 Timing Chart

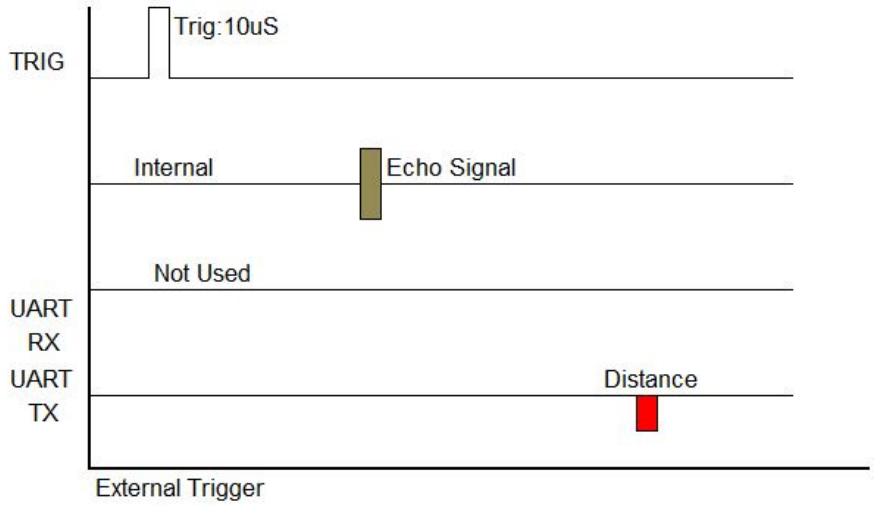
● **커맨드모드**

UART 통신을 통해 계측명령 0x22(0x23)을 보내면 계측된 정보가 출력된다.
(기타 명령은 **커맨드 표** 참조)



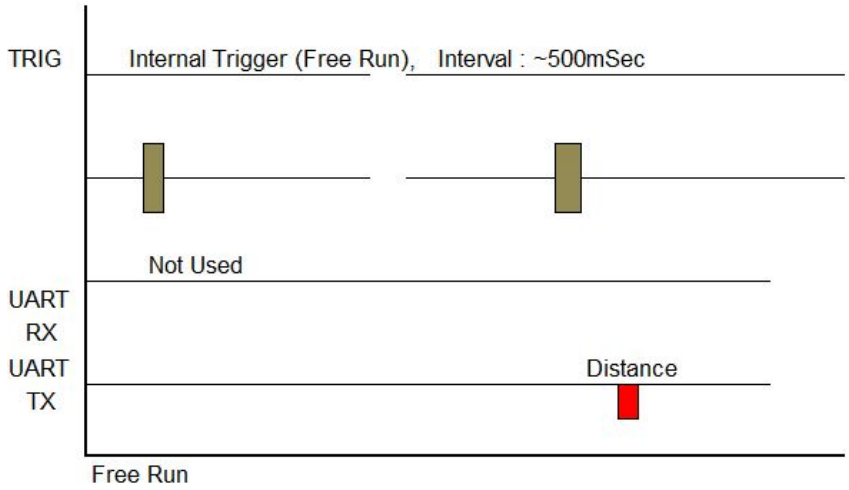
● **외부 트리거모드**

Trigger입력단자에 High Pulse로 트리거를 하면 계측을 시작하고, 계측된 정보는 UART통신을 통해 장애물과의 거리정보를 출력한다.



● Free Run 모드

자체 Trigger 를 발생하며 약 0.5 초 주기로 계측된 정보가 UART 로 전송된다.
Free Run 모드에서는 근접된 장애물에서 순간적으로 반사되는 짧은 초음파 에코 등의 오류를 보정하고 출력된다.



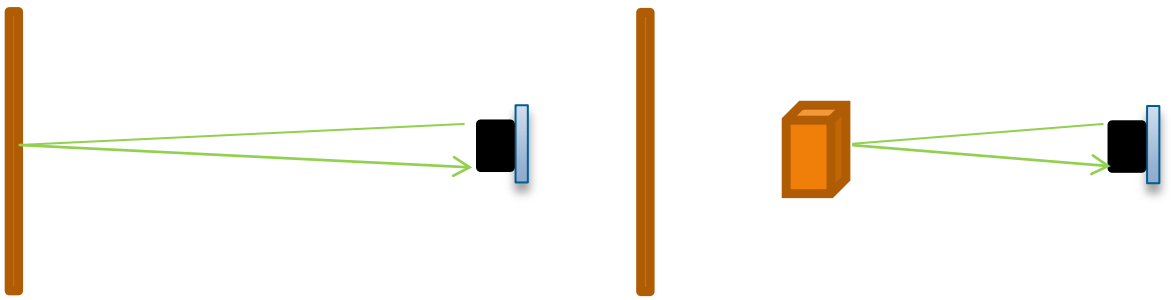
● Object Detector 모드

물체감지 기능으로 정해진 범위 안에 사람이나 물건이 근접하면 OUT(4 핀)단자에는 경보펄스가 High 로 (최소 0.5 초 이상지속)출력되고, UART 를 통해서 목표물과의 거리가 출력된다.

✶ 사용방법

모드를 설정한 후 정해진 장소에 설치하고 전원을 넣으면 10~20 초 동안 LED 가 약 400ms 주기로 점멸하면서 기준거리를 측정하여 저장하고, UART 통신을 통해 검출된 기준 거리가 출력된다.
물체감지 시작을 알리는 "{START}" 가 출력되고, LED 는 항상 켜진다.
이 후부터는 물체가 근접하면 OUT 단자는 최소 500ms 이상 'High' 를 유지하고, LED 는 200ms 주기로 점멸하게 된다.
UART 통신을 통해 근접된 거리가 출력된다. (물체가 계속 검출되면 LED 점멸과 출력은 계속 됨)

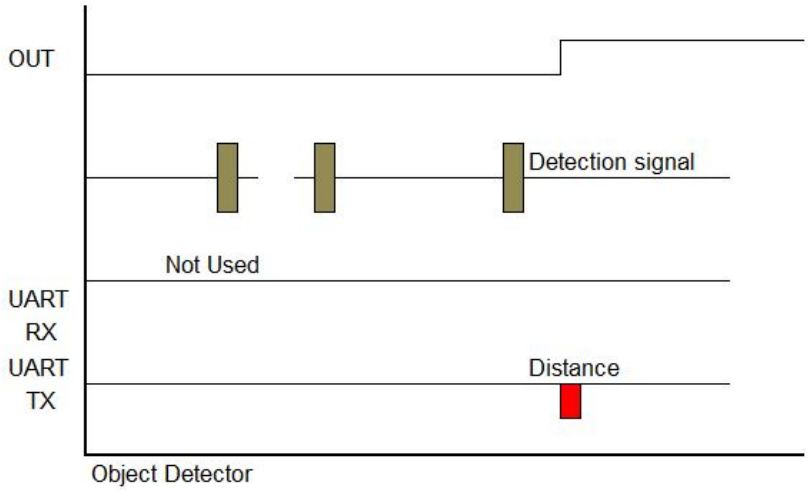
✶ TIP: 센서와의 거리가 650Cm 이상 되는 넓은 장소에 설치하면 기준거리 측정 시 UART 를 통해 거리 정보 "{SPACE}"가 나오고 끝나게 된다. (측정범위 벗어남)
이 경우 물체가 조금이라도 근접하면 경보가 발생하므로 물체감지 기능으로 사용하기에는 더욱 좋다.



기준거리 측정시간 10 초
*목표물과의 거리가 가깝거나 거리정보가 계속 변하면 안정될 때까지 측정은 계속된다.

1. 기준거리 측정

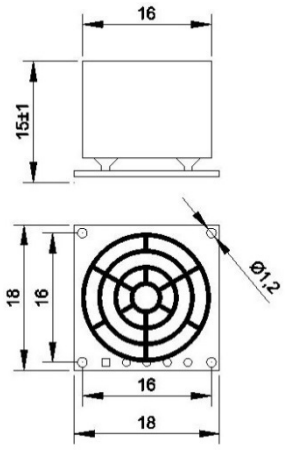
2. 물체 근접 시 경보 발생



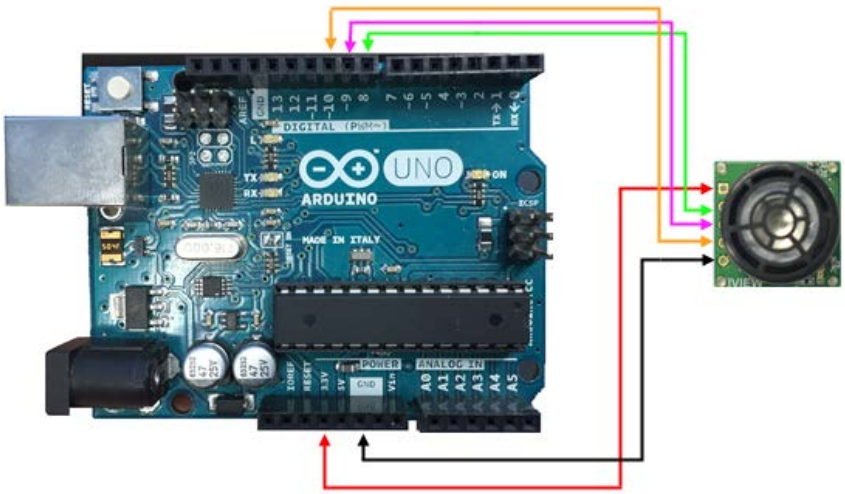
2. 물체감지 기능 전체 Timing Chart

❖ 모듈 사이즈

- Mechanical Dimension



❖ 아두이노와 아이뷰 초음파모듈 결선도 (예)



❖ 아두이노 프로그램 예문

```
/*====================
IUM-D10,D16  Arduino  : 접속 단자 설명
Vin        -> 3.3V     : 전원 (3.3V 또는 5V 를 사용)
RX         -> 8(TX)    : Software Serial 통신 TX 로 지정
TX         -> 9(RX)    : Software Serial 통신 RX 로 지정
I/O        -> 10       : 외부 트리거 단자 또는 물체감지 입력단자로 지정
Ground     -> GND      : GND
====================*/
#include <SoftwareSerial.h>     // 소프트웨어 시리얼을 사용하기 위해 헤더를 포함한다.

#define TrigPin 10              // H/W Define
// 예문에 사용된 커맨드
#define IUM_MODE_TRIG       0x30  // '1' 외부 트리거 모드
#define IUM_MODE_COM        0x31  // '2' UART 커맨드에 의한 동작모드
#define IUM_MODE_AUTO       0x32  // '3' Free RUN
#define IUM_MODE_OBJECT     0x33  // '4' 물체감지 모드
#define IUM_GET_RLRANGE_CM  0x22  // 음파펄스를 출력하고 에코신호를 계측한 후 측정값을 TX 로 보낸다

SoftwareSerial IUM_Serial(9, 8);  // Software Serial 을 사용하여 초음파 모듈을 제어한다.

// 초음파 모듈의 ID 값은 디폴트 1 로 지정되어있다.
// 전역변수 선언
byte ID=1;
char Buff[10];
// 현재의 ID 를 새로운 ID 로 변경하는 명령전송
void SendNewID(unsigned char old, unsigned char New)
{
  IUM_Serial.write(0xAA);     // 첫 번째 ID 변경명령 0xAA 전송
  IUM_Serial.write(0x55);     // 두 번째 연속된 ID 변경명령 0x55 전송
  IUM_Serial.write(old);      // 바꾸려는 모듈의 현재 ID 번호를 전송
  IUM_Serial.write(New);      // 바꾸려는 모듈의 새로운 ID 번호를 전송하면 변경이 완료된다.
}

// IUM 초음파 모듈에 명령을 전송한다.
void SendCmd(unsigned char address, unsigned char cmd)
{
  IUM_Serial.write(address);  // 명령을 전송하려면 모듈의 ID 를 먼저 보낸 후
  IUM_Serial.write(cmd);      // 커맨드를 해당모듈에 전송한다.
}

// 초음파모듈에서 ASCII 문자열 7 바이트를 읽어오는 함수
char GetSevenByte(void)
{
  if(IUM_Serial.available() >= 7) // ID 값 7 바이트를 Rx 단자를 통해서 받았으면...
  {
    // ASCII 문자 7 바이트를 버퍼에 저장한다.
    for(byte n=0; n<7; n++) Buff[n] = IUM_Serial.read();
    Buff[7] = 0;    // 디버그 창에 인쇄하기 위해 문자열을 0 으로 마감한다.
    return 1;
  }
  else return 0;
}
```

```
//=============================================================================================================
//  External Trigger MODE 프로그램 예문
//=============================================================================================================
void setup()
{
  Serial.begin (9600);        // 디버그용 시리얼 포트의 통신속도를 9600Baud 로 지정한다.
  IUM_Serial.begin (9600);    // 전원 공급 시 모듈의 통신속도는 항상 9600Baud 로 사용된다.
  pinMode(TrigPin, OUTPUT);   // TRIG PIN 출력 설정
  digitalWrite(TrigPin, LOW); // TRIG PIN 초기화

  SendCmd(1, IUM_MODE_TRIG);  // ID 1 번 동작모드를 외부 트리거 모드로 변경한다.
  delay(100);
  // 모드변경에 성공하면 모듈에서 ACK 가 출력된다. "(MODE0)" ~ "(MODE3)" *모드 값 0x30 은 ASCII 로 '0' 이된다.
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
      Serial.println(Buff);   // 성공적으로 모드가 변경된 경우 "(MODE0)" 이 출력된다.
}

void loop()
{
  digitalWrite(TrigPin, HIGH);
  delayMicroseconds(10);      // TRIG PIN 을 10us 동안만 High 를 만든다. 측정시작
  digitalWrite(TrigPin, LOW);
  delay(100);                 // 계측이 끝날 때까지 기다린다.
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
      Serial.println(Buff);   // 계측정보를 디버그 창에 인쇄한다.
}

//=============================================================================================================
//  COMMAND MODE 프로그램 예문
//=============================================================================================================
void setup()
{
  Serial.begin (9600);        // 디버그용 시리얼 포트의 통신속도를 9600Baud 로 지정한다.
  IUM_Serial.begin (9600);    // 전원 공급 시 모듈의 통신속도는 항상 9600Baud 로 사용된다.
  pinMode(TrigPin, INPUT);    // TRIG PIN 입력 설정: 외부트리거 모드가 아니면 입력으로 설정한다.
  digitalWrite(TrigPin, LOW); // TRIG PIN 초기화

  // UART RX/TX 를 통신용 공통버스로 구성하여 15 개의 모듈을 연결하는 경우, ID 를 0 으로 하여 전송한다.
  // ID 를 0 으로 하여 이 명령을 전송하면 현재 연결된 모든 모듈들은 같은 모드로 변경된다.
  ID=0;
  // UART RX/TX 를 통신용 공통버스로 구성하여 각각의 모드를 변경하는 경우는 ID 값을 지정하고 변경한다.
  //ID=X;                     // ID 지정
  SendCmd(ID, IUM_MODE_COM);  // 동작모드를 UART 커맨드에 의한 동작모드로 변경한다.
  delay(100);
  // 모드변경에 성공하면 모듈에서 ACK 가 출력된다. "(MODE0)" ~ "(MODE3)" *모드 값 0x31 은 ASCII 로 '1' 이된다.
  // 단, ACK 출력은 ID 가 1 인 모듈만 응답한다. 복수의 모듈을 연결한 경우도 동일하다.
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
      Serial.println(Buff);   // 성공적으로 모드가 변경된 경우 "(MODE1)" 이 출력된다.
}

void loop()
{
  float distance;
```

```
  // 공통버스에 연결된 15 개의 모듈에서 계측정보를 순차적으로 갖고 오는 방법
  for(ID=1; ID<=15; ID++)
  {
    SendCmd(ID, IUM_GET_RLRANGE_CM);  // 음파펄스를 출력하고 에코신호를 계측한 후 측정값을 TX 로 보낸다.
    delay(100);                       // 측정이 완료될 때까지 기다린다.
    if(GetSevenByte()==1)             // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
    {
      Serial.print(Buff);             // 받아온 계측정보를 디버그 창에 표시한다.
      Serial.print(" - ID : ");
      Serial.println(ID, DEC);        // ID 값도 "계측정보" 에 이어서 10 진수로 표시한다.
      // [123.4] 거리데이터가 수신된 경우, '[ '와 ']' 를 제거한 순수 거리데이터 '123.4' 만 추출한다.
      byte dist[5];
      for(byte n=0; n<5; n++) dist[n] = Buff[n+1];
      distance = atof(dist);          // Ascii 데이터가 Float 로 변환된 거리데이터 = 123.4cm
    }
  }
}

//=============================================================================================================
//  Free RUN - AUTO MODE 프로그램 예문
//=============================================================================================================
void setup()
{
  Serial.begin (9600);          // 디버그용 시리얼 포트의 통신속도를 9600Baud 로 지정한다.
  IUM_Serial.begin (9600);      // 전원 공급 시 모듈의 통신속도는 항상 9600Baud 로 사용된다.
  pinMode(TrigPin, INPUT);      // TRIG PIN 입력 설정: 외부트리거 모드가 아니면 입력으로 설정한다.
  digitalWrite(TrigPin, LOW);   // TRIG PIN 초기화

  SendCmd(1, IUM_MODE_AUTO);    // 동작모드를 Free RUN 모드로 변경한다.
  delay(100);
  // 모드변경에 성공하면 모듈에서 ACK 가 출력된다. "(MODE0)" ~ "(MODE3)" *모드 값 0x32 은 ASCII 로 '2' 가된다.
  if(GetSevenByte()==1)         // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
      Serial.println(Buff);     // 성공적으로 모드가 변경된 경우 "(MODE2)" 이 출력된다.
}

void loop()
{
  if(GetSevenByte()==1)         // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
      Serial.println(Buff);     // 계측정보를 디버그 창에 인쇄한다.
}

//=============================================================================================================
//  Object Detector MODE 프로그램 예문
//=============================================================================================================
void setup()
{
  Serial.begin (9600);          // 디버그용 시리얼 포트의 통신속도를 9600Baud 로 지정한다.
  IUM_Serial.begin (9600);      // 전원 공급 시 모듈의 통신속도는 항상 9600Baud 로 사용된다.
  pinMode(TrigPin, INPUT);      // TRIG PIN 입력 설정: 물체근접 시 모듈에서 "HIGH" 신호가 출력된다.
  digitalWrite(TrigPin, LOW); // TRIG PIN 초기화

  SendCmd(1, IUM_MODE_OBJECT);
  delay(100);
```

```
  // 모드변경에 성공하면 모듈에서 ACK 가 출력된다. "(MODE0)" ~ "(MODE3)" *모드 값 0x33 은 ASCII 로 '3' 이된다.
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
     Serial.println(Buff);    // 성공적으로 모드가 변경된 경우 "(MODE3)" 이 출력된다.
}

void loop()
{
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
     Serial.println(Buff);    // 초음파 센서와 물체와의 근접거리 정보를 디버그 창에 인쇄한다.
}

//=============================================================================================
//  초음파 모듈의 ID 변경 방법
//=============================================================================================
void setup()
{
  Serial.begin (9600);        // 디버그용 시리얼 포트의 통신속도를 9600Baud 로 지정한다.
  IUM_Serial.begin (9600);    // 전원 공급 시 모듈의 통신속도는 항상 9600Baud 로 사용된다.

  ID = 1;                     // 현재 모듈의 ID 번호 값 지정
  // ID 값을 변경하기 위해서는 커맨드 모드로 변경하고 진행한다.
  SendCmd(ID, IUM_MODE_COM);  // 동작모드를 UART 커맨드에 의한 동작모드로 변경한다.
  delay(100);
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
     Serial.println(Buff);    // 성공적으로 모드가 변경된 경우 "(MODE1)" 이 출력된다.
  delay(10);

  SendNewID(1, 2);            // 현재의 ID: 1 번을 ID: 2 번으로 변경한다.
  delay(300);

  // ID 변경에 성공하면 ACK 를 출력한다 (ID:XX)" (XX=새로 변경된 ID 값)
  if(GetSevenByte()==1)       // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
     Serial.println(Buff);    // 디버그 창에 새로 변경된 ID 값 "(ID:02)" 가 출력된다.
}

void loop()
{
  // 변경된 ID 로 COMMAND MODE 프로그램을 진행한다.
  float distance;

  ID=2;                             // ID 가 2 번으로 변경됐다.
  SendCmd(ID, IUM_GET_RLRANGE_CM);  // 음파펄스를 출력하고 에코신호를 계측한 후 측정값을 TX 로 보낸다.
  delay(100);                       // 측정이 완료될 때까지 기다린다.

  if(GetSevenByte()==1)             // 만약 7 바이트 이상을 Rx 단자를 통해서 받았으면 Buff 에 저장하고 갖고 온다
  {
    Serial.println(Buff);             // 받아온 계측정보를 디버그 창에 표시한다.
    // [123.4] 거리데이터가 수신된 경우, '[ '와 ']' 를 제거한 순수 거리데이터 '123.4' 만 추출한다.
    byte dist[5];
    for(byte n=0; n<5; n++) dist[n] = Buff[n+1];
    distance = atof(dist);          // Ascii 데이터가 Float 로 변환된 거리데이터 = 123.4cm
  }
}
```