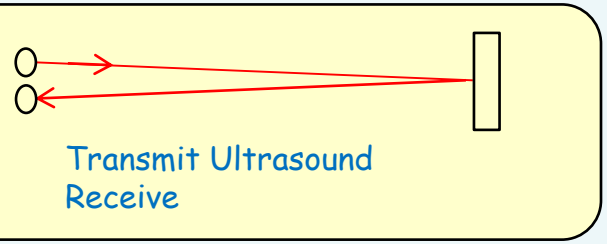
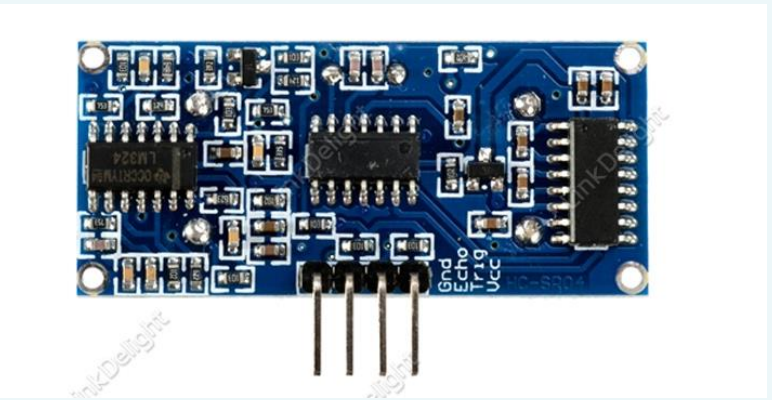
Internet of Things
class 5

Digital Sensors: Ultrasonic / PIR Sensor
Interrupt

Ultrasonic Sensor

- HC-SR04P (초음파센서)
  - Measurable Range: 2cm~400cm
  - Basic principle of work:
    - Using IO trigger for at least 10us high level signal
    - Module automatically sends eight 40kHz and detect whether there is a pulse signal back
    - If the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning
  - Test distance = (high level time X velocity of sound (340M/S) / 2





Ultrasonic Sensor

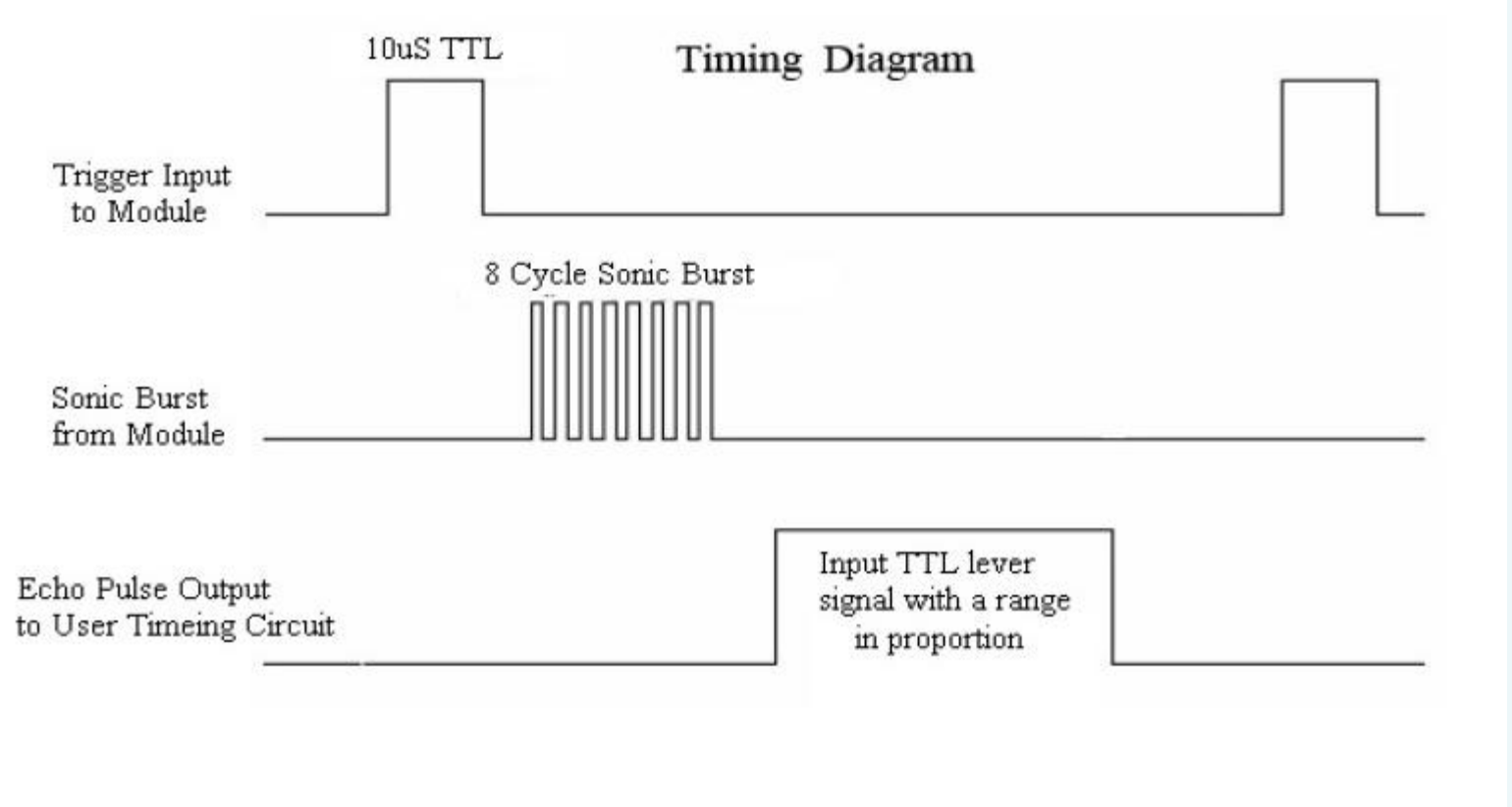
- HC-SR04
  - Electronic Parameters

Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm

- HC-SR04P.. DC 3.3V 용

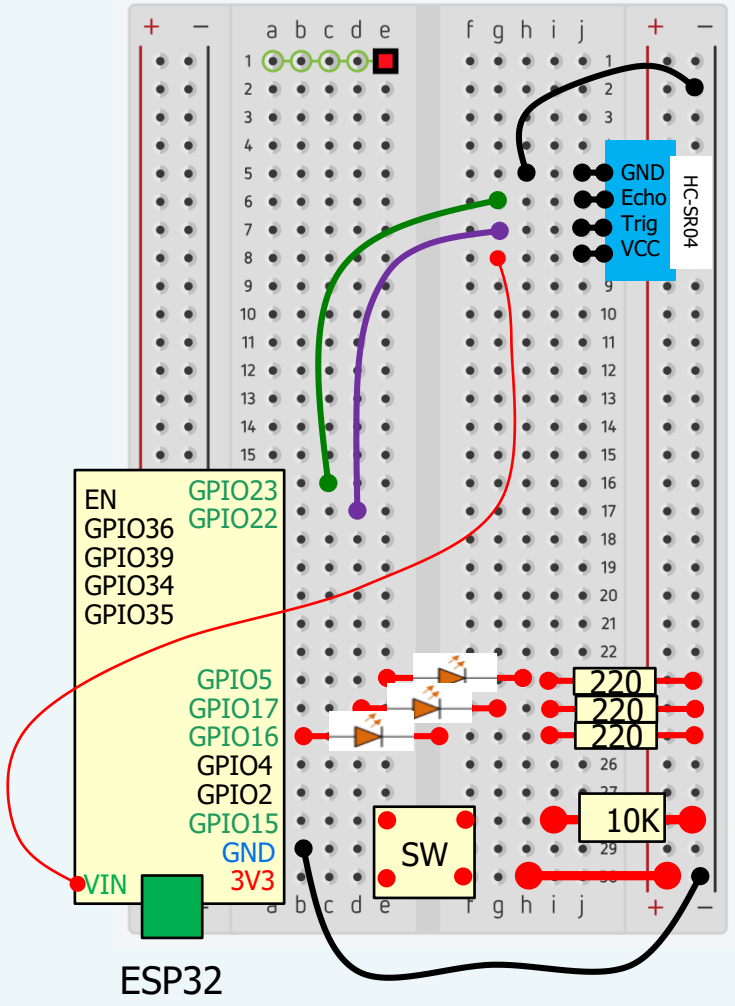
Ultrasonic Sensor

- HC-SR04(P)
  - Timing Diagram

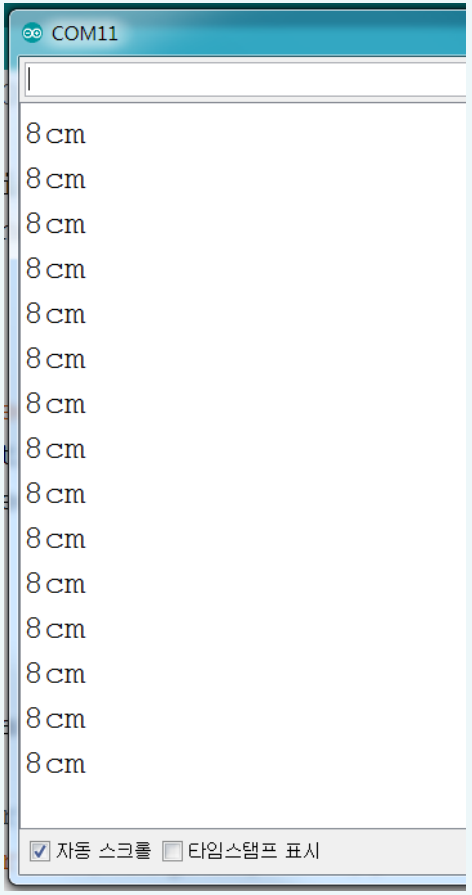


Ultrasonic Sensor

<Task05-1> ** HC-SR04(P)의 Echo, Trig Pin을 GPIO23, 22에 연결 **



HC-SR04는 VCC가 5V이므로 VIN Pin에 연결
HC-SR04P는 VCC가 3.3V이므로 +3.3V에 연결



C.B.Choi gen1223@kau.ac.kr

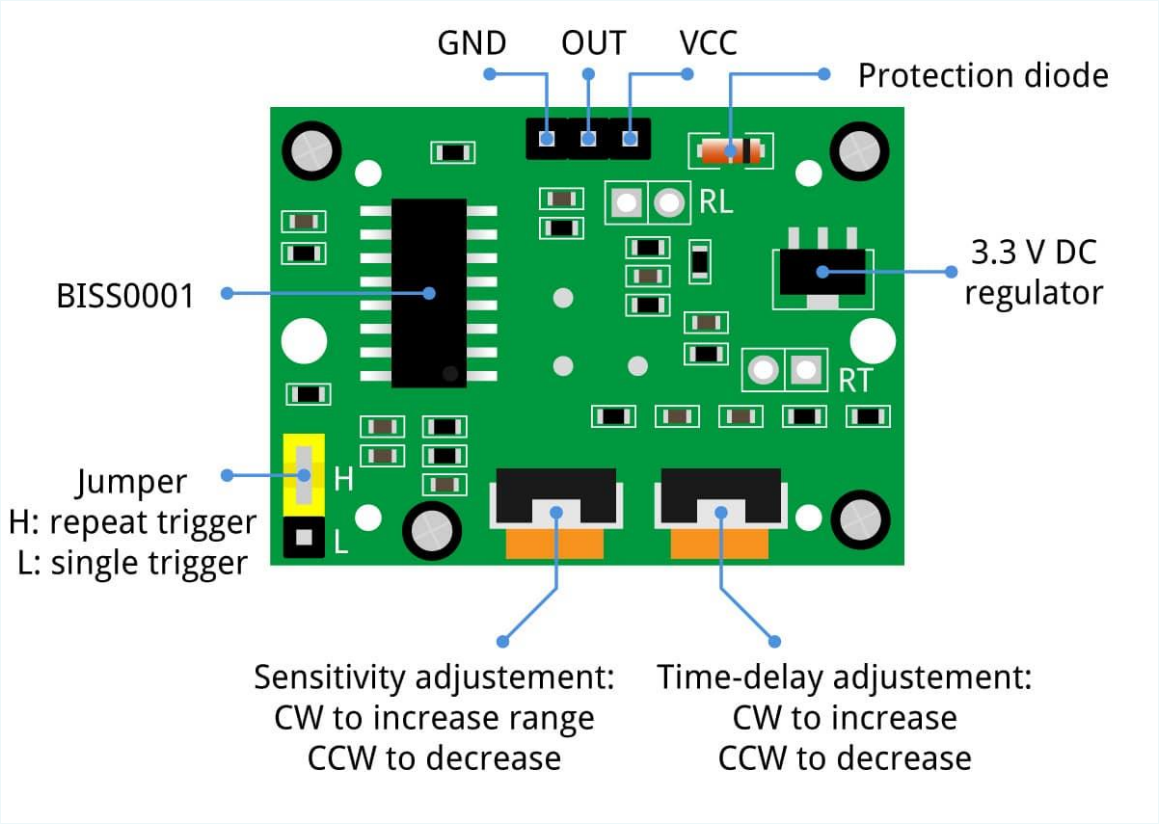
Ultrasonic Sensor

<Task05-1>

```
const int trigPin = 22;
const int echoPin = 23;

void setup()
{
    Serial.begin(115200);
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
}

void loop()
{
    long duration, distance;

    //Triggering by 10us pulse
    digitalWrite(trigPin, LOW);      // trig low for 2us
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);     // trig high for 10us
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    //getting duration for echo pulse
    duration = pulseIn(echoPin, HIGH);

    //sound speed = 340 m/s = 34/1000 cm/us
    //distance = duration * 34/1000 * 1/2
    distance = duration * 17 / 1000;

    Serial.print(distance);
    Serial.print("cm");
    Serial.println();

    delay(100);
}
```

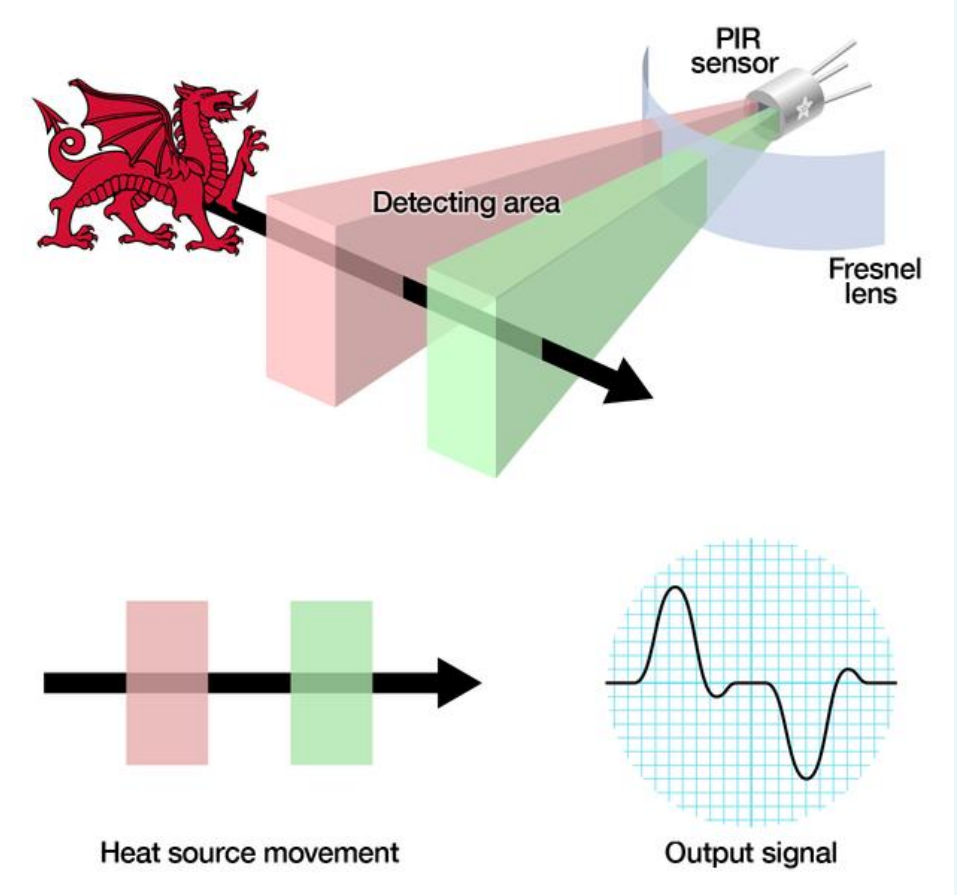
PIR Motion Sensor - Module

- Passive Infrared Sensor: HC-SR501 (인체감지 센서)
  - Design



PIR Motion Sensor - Operations

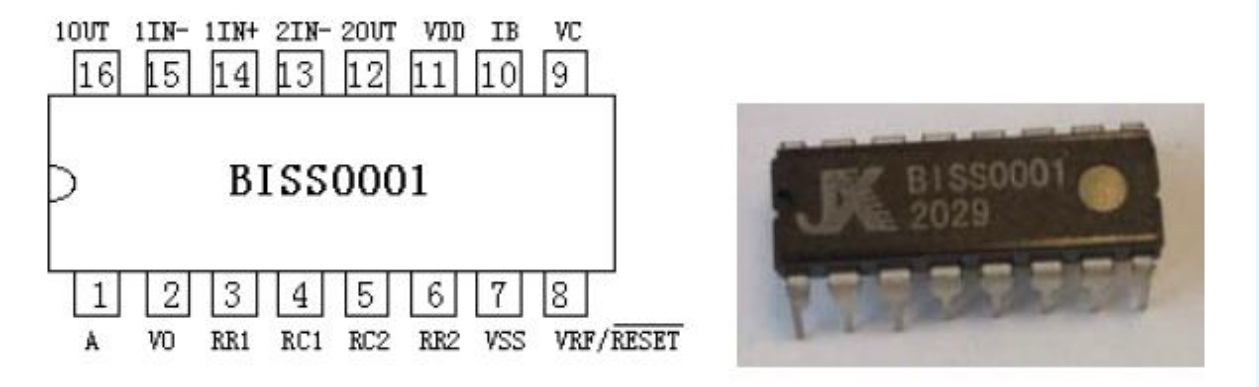
- Passive Infrared Sensor: HC-SR501
  - Operation



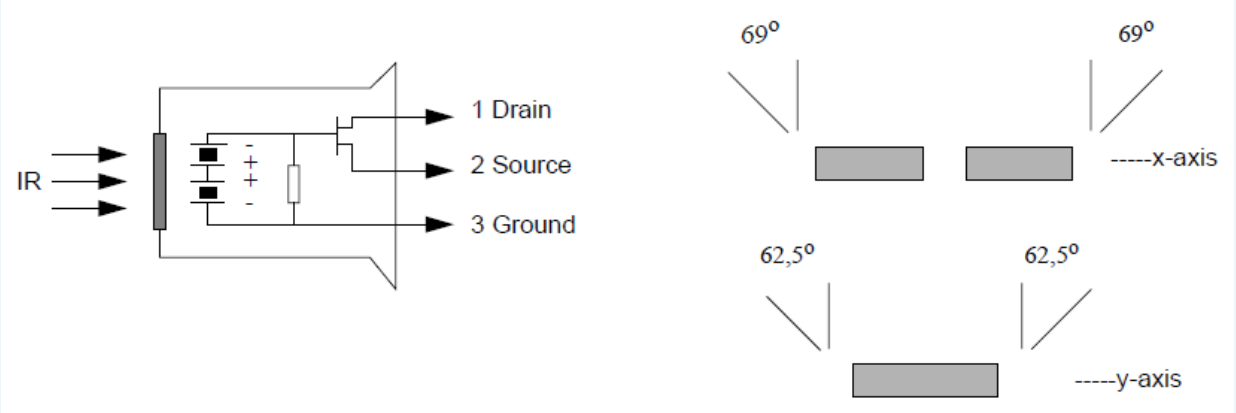
*soure: www.adafruit.com

PIR Motion Sensor – Motion Detector IC

- Passive Infrared Sensor: HC-SR501
  - Micro Power PIR Motion Detector IC



  - Pyroelectric Passive Infrared Sensor



PIR Motion Sensor - Specification

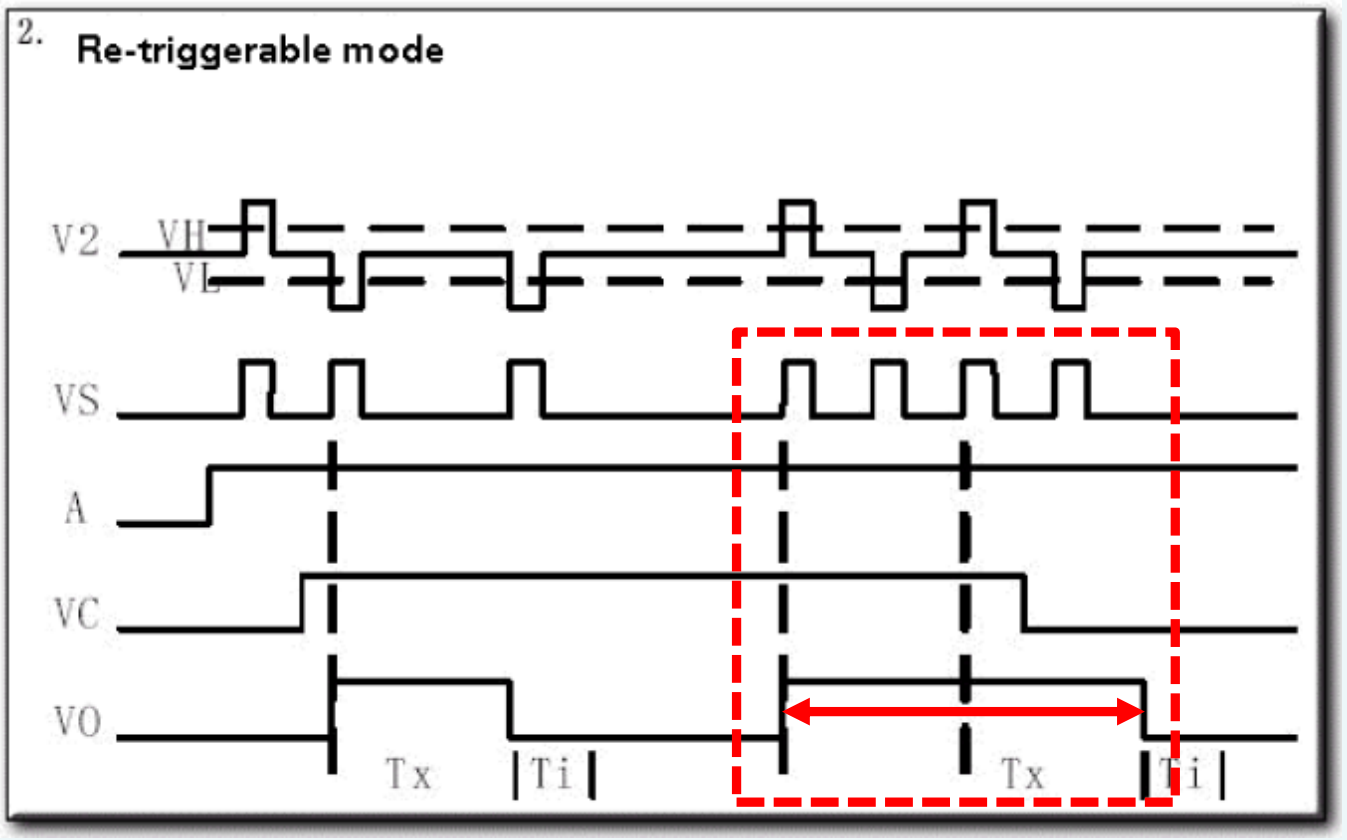
- HC-SR501 Specification
  - Voltage: 5V ~ 20V
  - Power Consumption: 65mA
  - TTL output: 3.3V, 0V
  - Delay time: Adjustable (0.3 ~ 5 min)
  - Lock time: 0.2 sec
  - Trigger methods: L-disable repeat trigger, H-enable repeat trigger (default)
  - Sensing range: less than 120 degree, within 7 meters
  - Temperature: -15 ~ +70
  - Dimension: 32 x 24 mm

PIR Motion Sensor - Instructions

- HC-SR501 Instructions for Use
  - Sensor module is powered up after a minute, in this initialization time intervals during this module will output 0~3 times, a minute later enters the standby state
  - Should try to avoid the lights and other sources of interference close direct module surface of the lens, in order to avoid the introduction of interference siganl malfunction; environment should avoid the wind flow, the wind will cause interference on the sensor
  - Sensor module with dual probe, the probe window is rectangular, dual (A B) in both ends of the longitudinal direction
  - The dual direction of sensor should be installed parallel as far as possible in inline with human movement. In order to increase the sensor angle range, the module using a circular lens also makes the probe surrounded induction, but the left and right sides still up and down in both directions sensing range, sensitivity, still need to try to install the above requirements

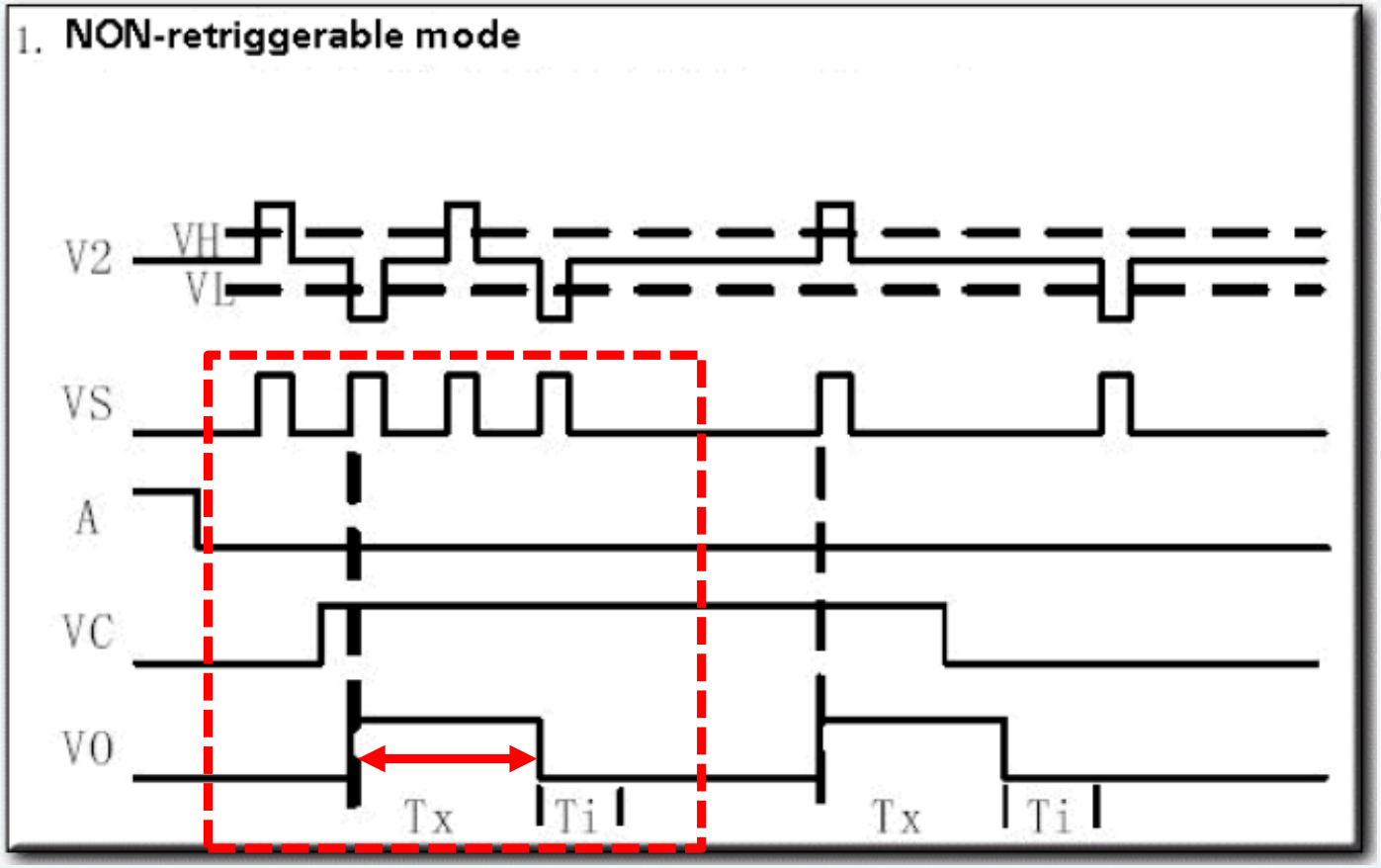
PIR Motion Sensor – Operation Mode

- Operation Mode
  - Retrigerrable vs. Non-Retrigerrable



PIR Motion Sensor – Operation Mode

- Operation Mode
  - Retrigerrable vs. Non-Retrigerrable

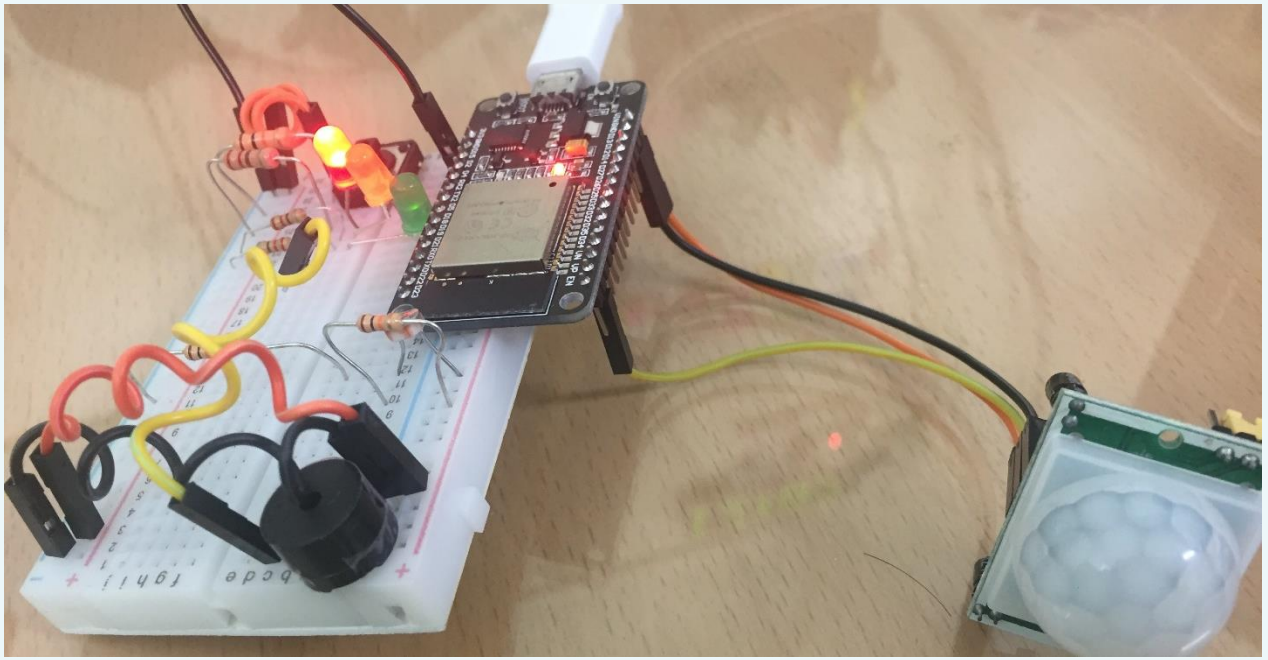


PIR Motion Sensor: Interrupts and Timers

- Example: when motion is detected, the ESP32 starts a timer and turns an LED on for some seconds.
- Two major concepts:

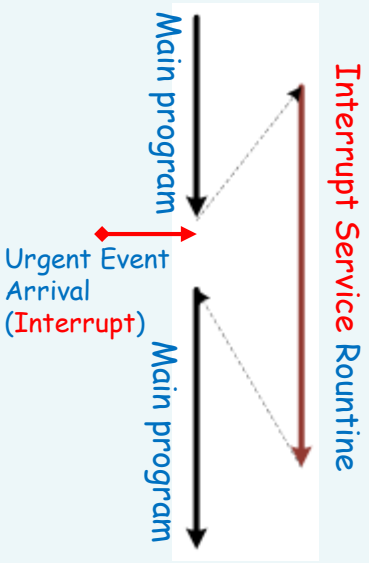
  1) Interrupts

  2) Timers



Interrupts in Arduino IDE

- Why Interrupt? Interrupt vs Polling
  - Interrupt is used to trigger an event
  - Don't need to constantly check the status
- When a change is detected..
  - an event is triggered (a function is called)



- Setting Interrupts..

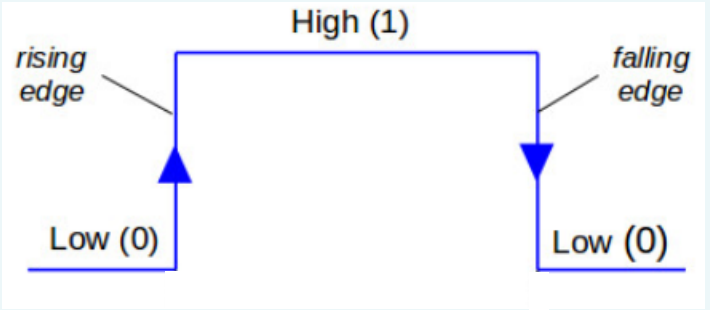
attachInterrupt(digitalPinToInterrupt(GPIO), function, mode)

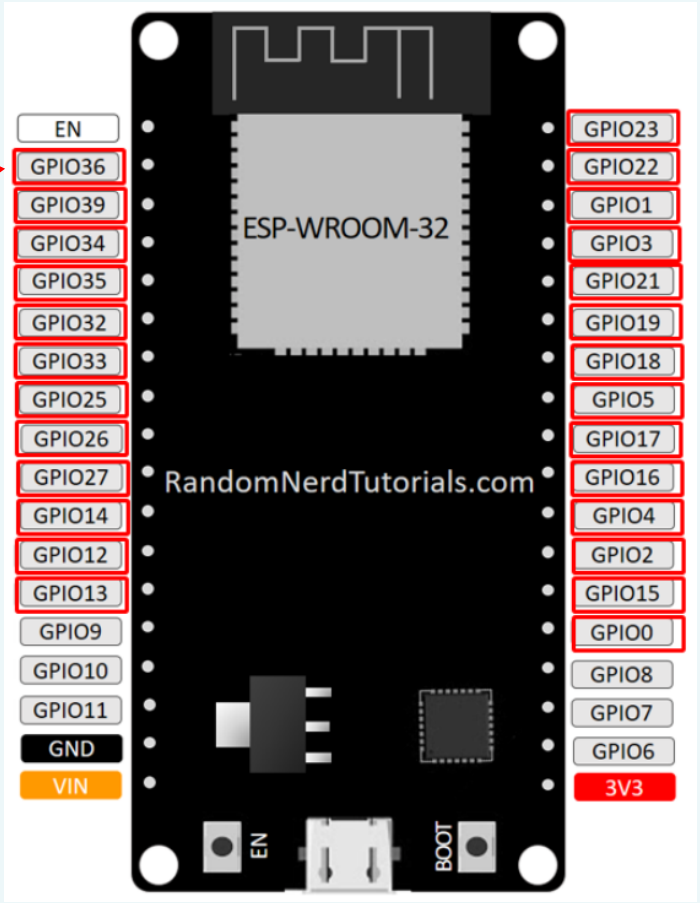
  - digitalPinToInterrupt(GPIO): assigning GPIO for interrupt
  - function: Name of the Function to be executed
  - mode: Operation mode

Interrupts in Arduino IDE

- GPIO pins for Interrupts
  - For example

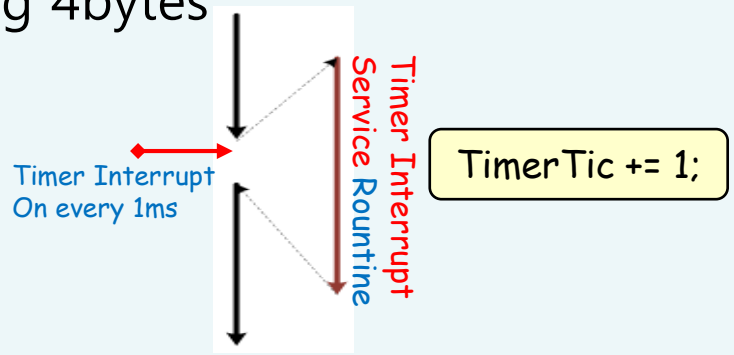
  digitalPinToInterrupt(**36**)
- Mode: when to trigger
  - LOW: when the pin is LOW
  - HIGH: when the pin is HIGH
  - CHANGE: when changes value
  - FALLING: from HIGH to LOW
  - RISING: LOW to HIGH





Needs for Timers

- delay(**time in milliseconds**) for waiting..
- But, delay() is a blocking function
  - Blocking function cannot do anything else until it is completed.
- Cannot use delay() in Multi-tasking Programs
  - Use timers instead
  - millis() function returns Timer-tick value..
  - the number of milliseconds that have passed since the power ON
  - about 50 days for expiring 4bytes
  - about 213 billion days for 8bytes value



C.B.Choi gen1223@kau.ac.kr

Blinking an LED with millis()

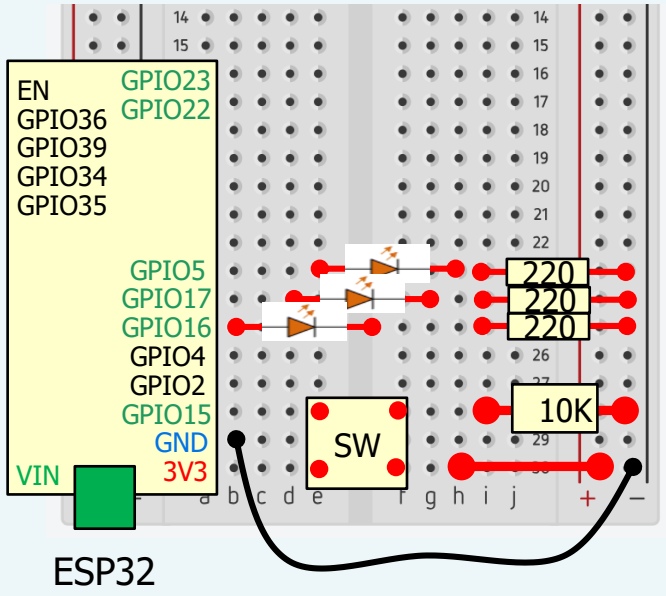
<Task05-2> Write code turning on LED without delay() function

```
const int ledPin = 16;                  // the number of the LED pin
int ledState = LOW;                     // ledState used to set the LED
// Generally, you should use "unsigned long" for variables that hold time
// The value will quickly become too large for an int to store
unsigned long previousMillis = 0; // will store last time LED was updated
const long interval = 1000; // interval at which to blink (milliseconds)

void setup() {
      // set the digital pin as output:
      pinMode(ledPin, OUTPUT);
}

void loop() {
// here is where you'd put code that needs to be running all the time.
      unsigned long currentMillis = millis();
      if (currentMillis - previousMillis >= interval) {
      // save the last time you blinked the LED
            previousMillis = currentMillis;
            if (ledState == LOW) ledState = HIGH;
            else ledState = LOW;
      }
      // set the LED with the ledState of the variable:
      digitalWrite(ledPin, ledState);
}
```

Blinking an LED with millis()

<Task05-2>

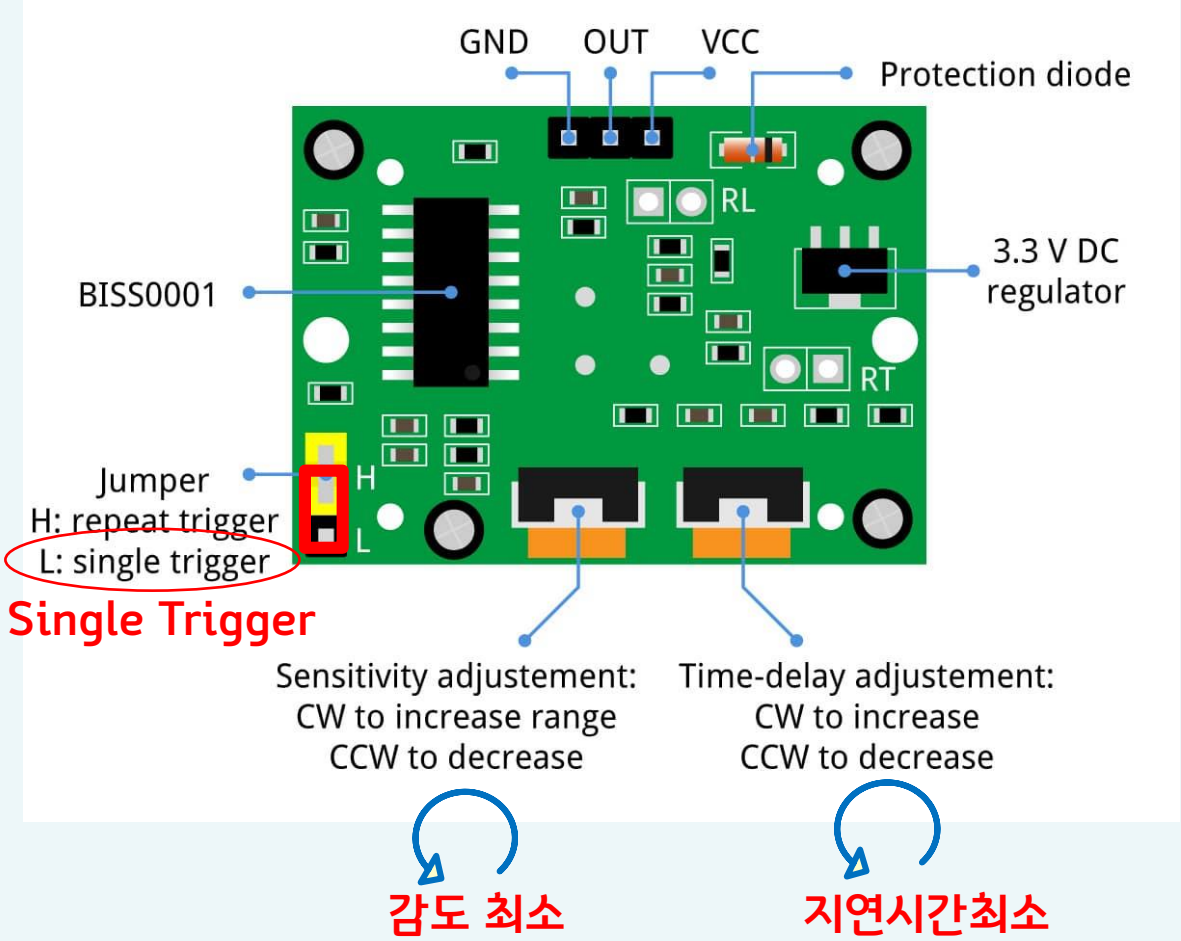
- millis() is used instead of delay()
- if (currentMillis - previousMillis >= interval)
  - Timer values gap is tested if it is bigger than interval
  - that is, elapsed time after changing LED
- Other codes can be added without blocking in loop()



PIR Motion Sensor - Module

<Task05-3> Motion Detector.. 인체가 감지되면 LED-ON

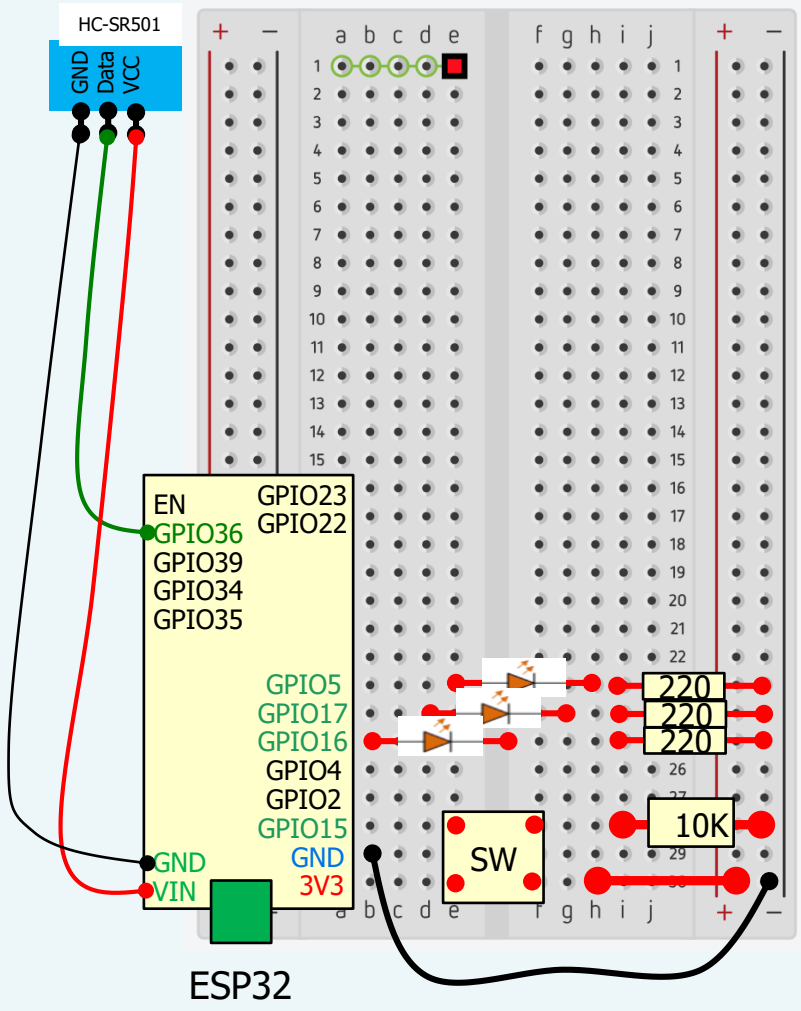
- Setting Sensitivity, Time-delay, Trigger Mode for HC-SR501



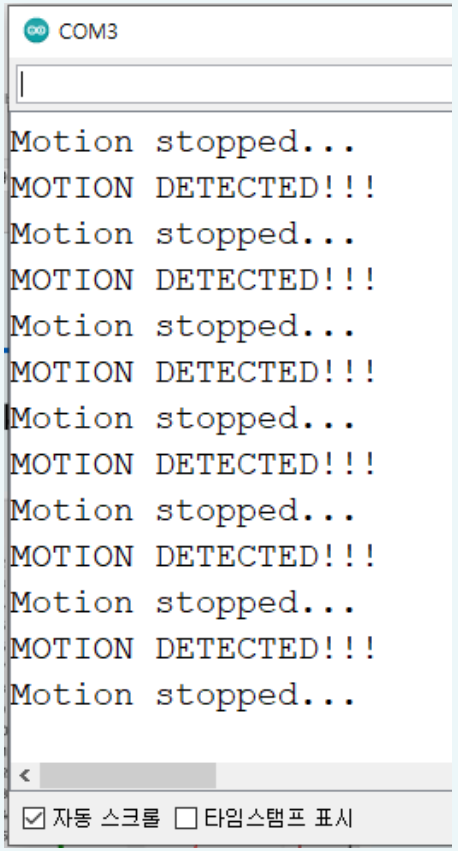
C.B.Choi gen1223@kau.ac.kr

ESP32 with PIR Motion Sensor

<Task05-3> ** HC-SR501의 DataOut Pin을 GPIO36에 연결 **



HC-SR501은 VCC가 5V이므로 VIN Pin에 연결



C.B.Choi gen1223@kau.ac.kr

ESP32 with PIR Motion Sensor

<Task05-3>

```
#define timeSeconds 3

// Set GPIOs for LED and PIR Motion Sensor
const int led = 16;
const int motionSensor = 36;

// Timer: Auxiliary variables
unsigned long now = millis();
unsigned long lastTrigger = 0;
boolean startTimer = false;

// Checks if motion was detected, sets LED HIGH and starts a timer
void IRAM_ATTR detectsMovement() {
      Serial.println("MOTION DETECTED!!!");
      digitalWrite(led, HIGH);
      startTimer = true;
      lastTrigger = millis();
}
```

- Interrupt Handler: detectsMovement()
- IRAM_ATTR: put the code in RAM

ESP32 with PIR Motion Sensor

```
void setup() {
// Serial port for debugging purposes
      Serial.begin(115200);
// PIR Motion Sensor mode INPUT_PULLUP
      pinMode(motionSensor, INPUT_PULLUP);
// Set motionSensor pin as interrupt, assign interrupt function and set RISING mode
      attachInterrupt(digitalPinToInterrupt(motionSensor), detectsMovement, RISING);
// Set LED to LOW
      pinMode(led, OUTPUT);
      digitalWrite(led, LOW);
}

void loop() {
// Current time
      now = millis();
      // Turn off the LED after the number of seconds defined in the timeSeconds variable
      if (startTimer && (now - lastTrigger > (timeSeconds*1000))) {
            Serial.println("Motion stopped...");
            digitalWrite(led, LOW);
            startTimer = false;
      }
}
```

- PIR sensor Pin Mode: INPUT_PULLUP.. Internally pulled-up