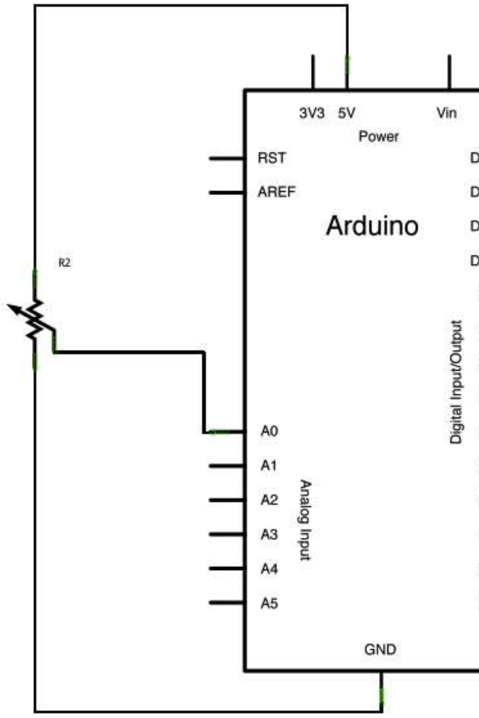
IoT

Arduino-Analog Input/Output

Dr. Sarwan Singh
Deputy Director
NIELIT Chandigarh

Agenda

- Analog Input/Output
- LED fading
- LDR
- PIR
- Practical Example/Experiment



Introduction

- Writes an analog value (PWM wave) to a pin. Can be used to light a LED at varying brightness or drive a motor at various speeds.
- After a call to analogWrite(), the pin will generate a steady square wave of the specified duty cycle until the next call to analogWrite()

int analogRead(pin)

analogWrite(pin, value)

- **pin**: the pin to write to. Allowed data types: int.
- **value**: the duty cycle: between 0 (always off) and 255 (always on). Allowed data types: int

val = analogRead(analogPin); // read the input pin
analogWrite(ledPin, val / 4); // analogRead values go from 0 to 1023, analogWrite values from 0 to 255

Pulse Width Modulation (PWM)

- PWM, is a technique for getting analog results with digital means.
- Digital control is used to create a square wave, a signal switched between on and off. This on-off pattern can simulate voltages in between full on (5 Volts) and off (0 Volts) by changing the portion of the time the signal spends on versus the time that the signal spends off.
- The duration of "on time" is called the pulse width. To get varying analog values, you change, or modulate, that pulse width. If you repeat this on-off pattern fast enough with an LED for example, the result is as if the signal is a steady voltage between 0 and 5v controlling the brightness of the LED

Source: https://www.arduino.cc/en/Tutorial/PWM

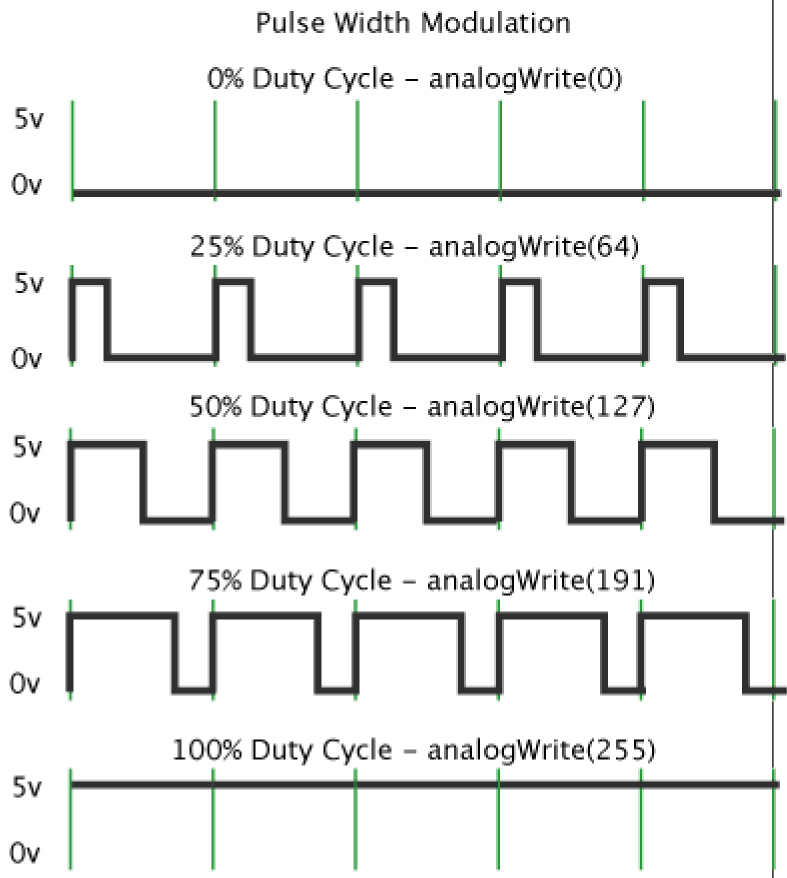
Pulse Width Modulation (PWM)

- The green lines represent a regular time period. (*This duration or period is inverse of PWM frequency*)
- Arduino's PWM frequency at about 500Hz, *the green lines would measure 2 milliseconds each*.
- analogWrite()

  is on a scale of 0 – 255

255 – 100% duty cycle,

127 – 50% duty cycle


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
25% Duty Cycle – analogWrite(64)
50% Duty Cycle – analogWrite(127)
75% Duty Cycle – analogWrite(191)
100% Duty Cycle – analogWrite(255)
5v
0v

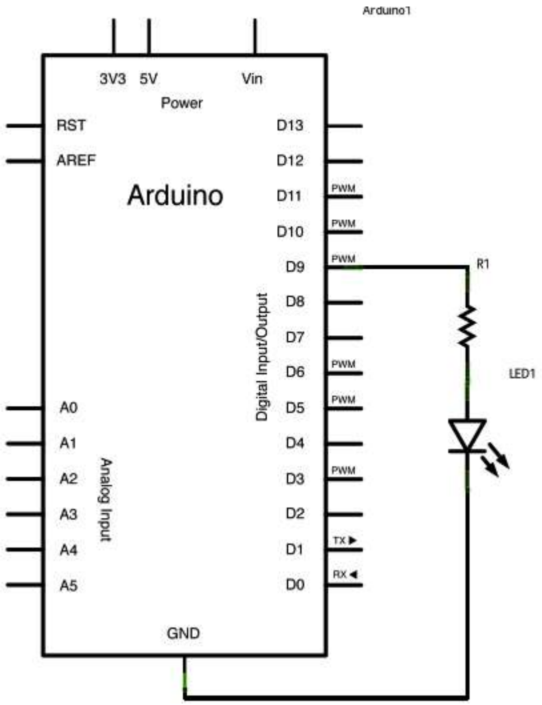
Source: https://www.arduino.cc/en/Tutorial/PWM

LED Fading - PWM

- AnalogWrite uses pulse width modulation (PWM), turning a digital pin on and off very quickly with different ratio between on and off, to create a fading effect.

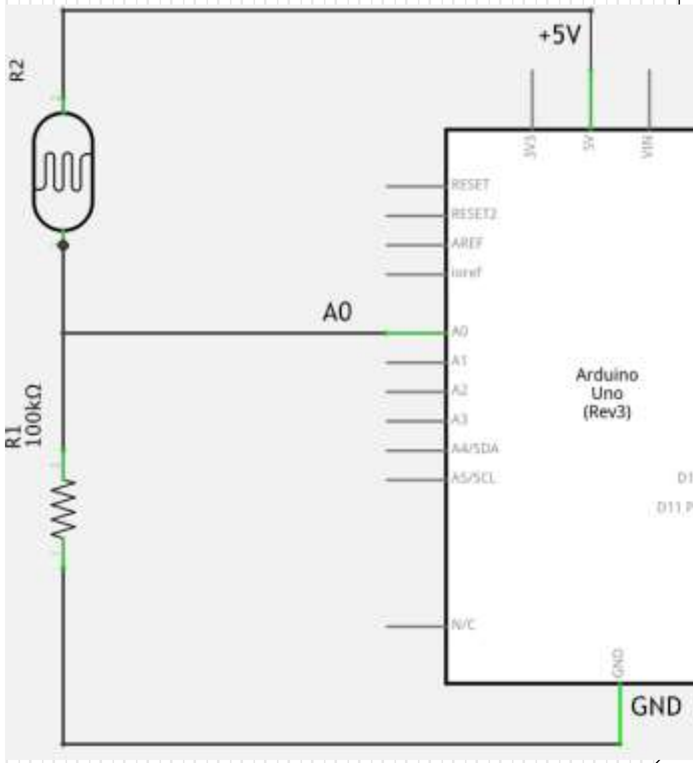
LED Fading - PWM

```
int led = 9;            // the PWM pin the LED is attached to
int brightness = 0;    // how bright the LED is
int fadeAmount = 5;    // how many points to fade the LED by
// the setup routine runs once when you press reset:
void setup() {  // declare pin 9 to be an output:
  pinMode(led, OUTPUT);
}
// the loop routine runs over and over again forever:
void loop() {
  // set the brightness of pin 9:
  analogWrite(led, brightness);
  // change the brightness for next time through the loop:
  brightness = brightness + fadeAmount;
  // reverse the direction of the fading at the ends of the fade:
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  // wait for 30 milliseconds to see the dimming effect
  delay(30);
}
```



LDR Interfacing

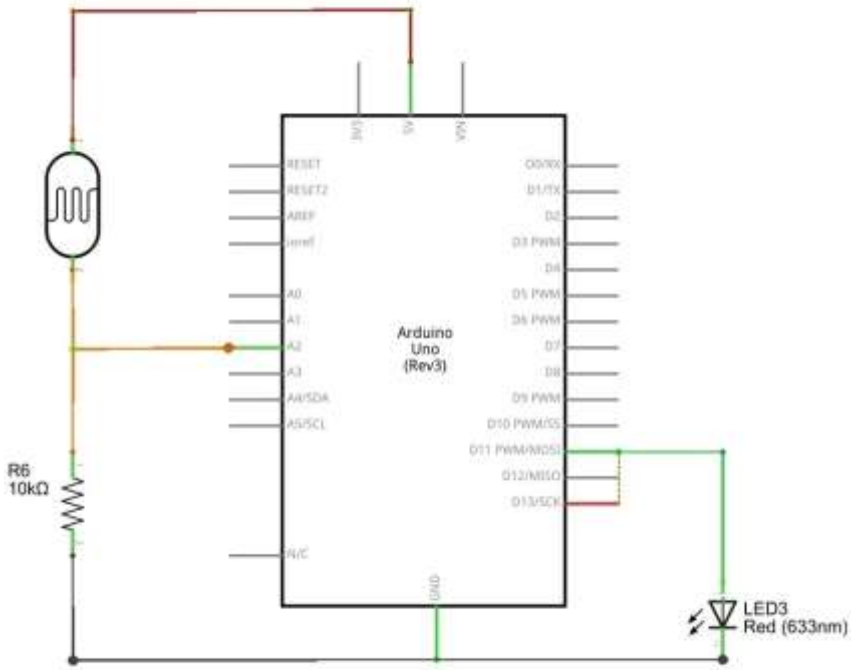
LDR (Light Dependent Resistor, or Photo resistor) is basically a resistor that change resistance depending on light. More light means less resistance. Less light means more resistance.



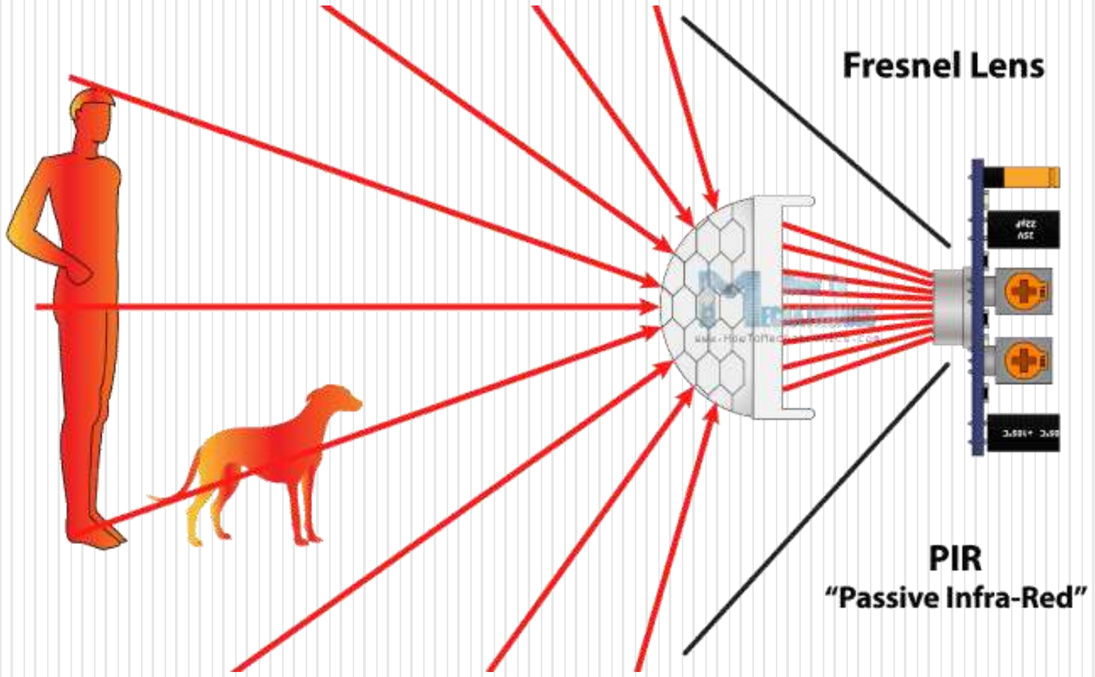
LDR Interfacing

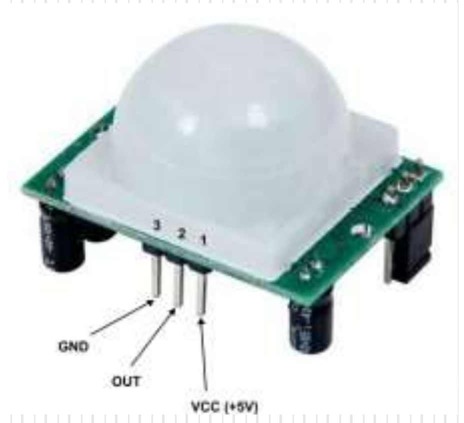
```
#define LDRpin A0 // pin where we connected the LDR and the resistor

int LDRValue = 0;     // result of reading the analog pin

void setup() {
  Serial.begin(9600); // sets serial port for communication
}

void loop() {
  LDRValue = analogRead(LDRpin); // read the value from the LDR
  Serial.println(LDRValue);      // print the value to the serial port
  delay(100);                    // wait a little
}
```

```
int ledPin = 3;
int photocell = 0;

void setup() {
pinMode(ledPin, OUTPUT);
Serial.begin(9600);
}

void loop() {
photocell = (analogRead(0)/4); // Divides
input 0-1023 to resemble to 0-255

analogWrite(ledPin, photocell);
Serial.print(photocell);
// The delay can be change to get the desired
dimming effect
delay(20);
}
```



PIR sensor
Passive Infra Red sensors

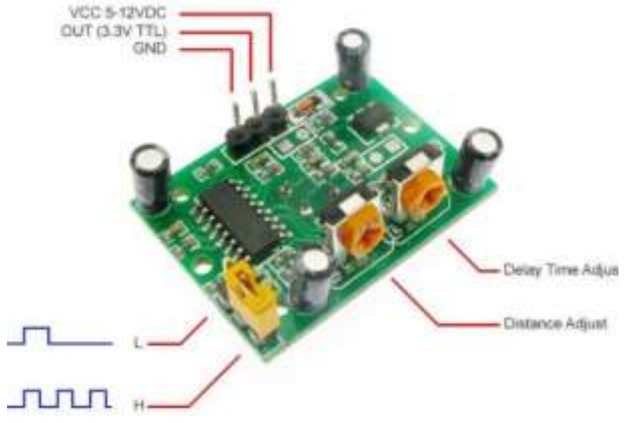


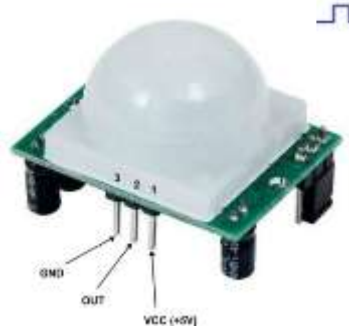


Introduction

PIR sensor is an electronic sensor that measures infrared light radiating from objects in its field of view. Normally this type of sensor would be used as a motion or proximity sensor.

- Quite often they are referred to as:
- PIR Sensor
- Motion Sensor
- Proximity Sensor
- Infrared (motion) Sensor
- Pyroelectric sensor





Working

- The sensor in the PIR detects or “reads” infrared radiation “emitted” from objects all around us.
- Each object with a temperature above absolute zero (-273.15° Celsius, -459.67° Fahrenheit, or 0 Kelvin) will radiate infrared, even us humans, and even though we mere humans cannot see this.
- With special Thermal Infrared Camera’s however, this can be made visible to the human eye.

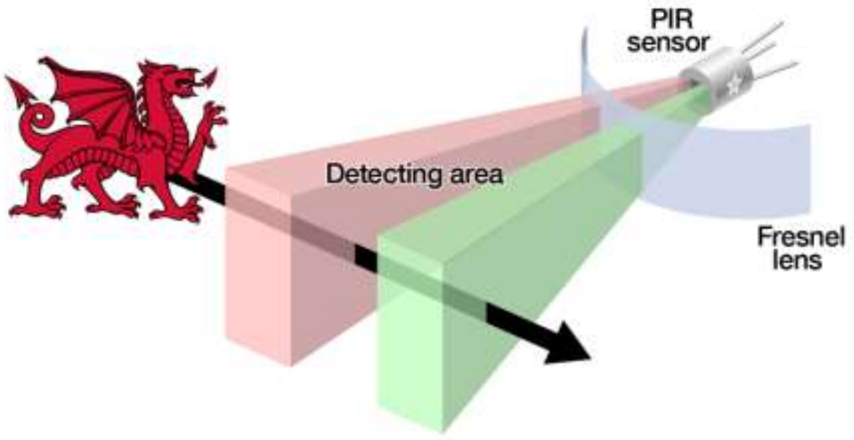
Source : https://www.tweaking4all.com/hardware/pir-sensor/

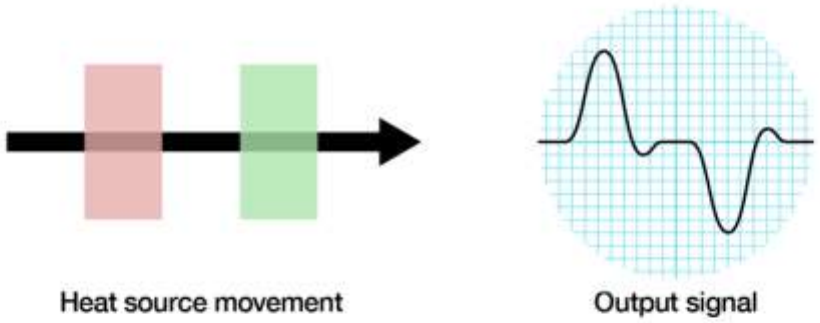
Working

- PIRs are called "passive" since they are not assisted by any "helpers" that for example would send some form or shape of "radiation" or "light" to help detect. It's purely based on what the sensor can pick up out of the environment, what's being emitted by objects.
- PIR's actually only look at the "difference" between two sensor "halves". If the difference is too high then it will trigger – it detects "motion". This is done in a smart way, to avoid false positives caused for example by a brief flash or an increase in room temperature.

Working

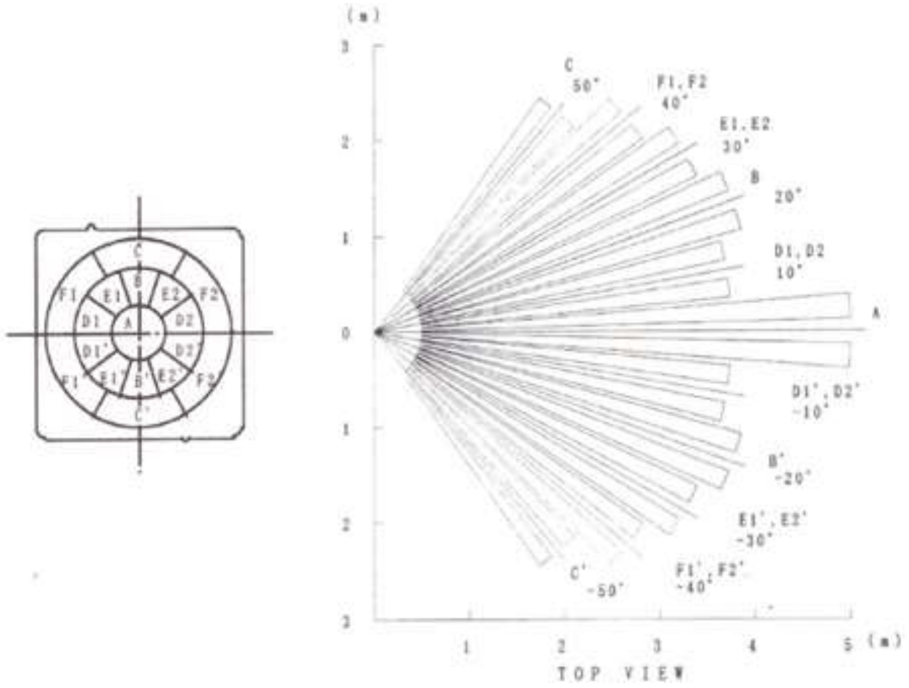
- When the sensor is idle, both slots detect the same amount of IR, the ambient amount radiated from the room or walls or outdoors.
- When a warm body like a human or animal passes by, it first intercepts one half of the PIR sensor, which causes a *positive differential* change between the two halves.





- When the warm body leaves the sensing area, the reverse happens, whereby the sensor generates a negative differential change. These change pulses are what is detected.

Hardware Insight



```
/*    * PIR sensor tester    */

int ledPin = 9;                 // choose the pin for the LED
int inputPin = 2;               // choose the input pin (for PIR sensor)
int pirState = LOW;             // we start, assuming no motion
detected
int val = 0;                    // variable for reading the pin status

void setup() {
  pinMode(ledPin, OUTPUT);      // declare LED as output
  pinMode(inputPin, INPUT);     // declare sensor as input
   Serial.begin(9600);
}
```

```
void loop(){
 val = digitalRead(inputPin);  // read input value
 if (val == HIGH) {           // check if the input is HIGH
     digitalWrite(ledPin, HIGH);  // turn LED ON
     if (pirState == LOW) {
       // we have just turned on
       Serial.println("Motion detected!");
       // We only want to print on the output change, not state
       pirState = HIGH;
     }
 } else {
     digitalWrite(ledPin, LOW); // turn LED OFF
     if (pirState == HIGH){
       // we have just turned of
       Serial.println("Motion ended!");
       // We only want to print on the output change, not state
       pirState = LOW;
     }
 }
}
```