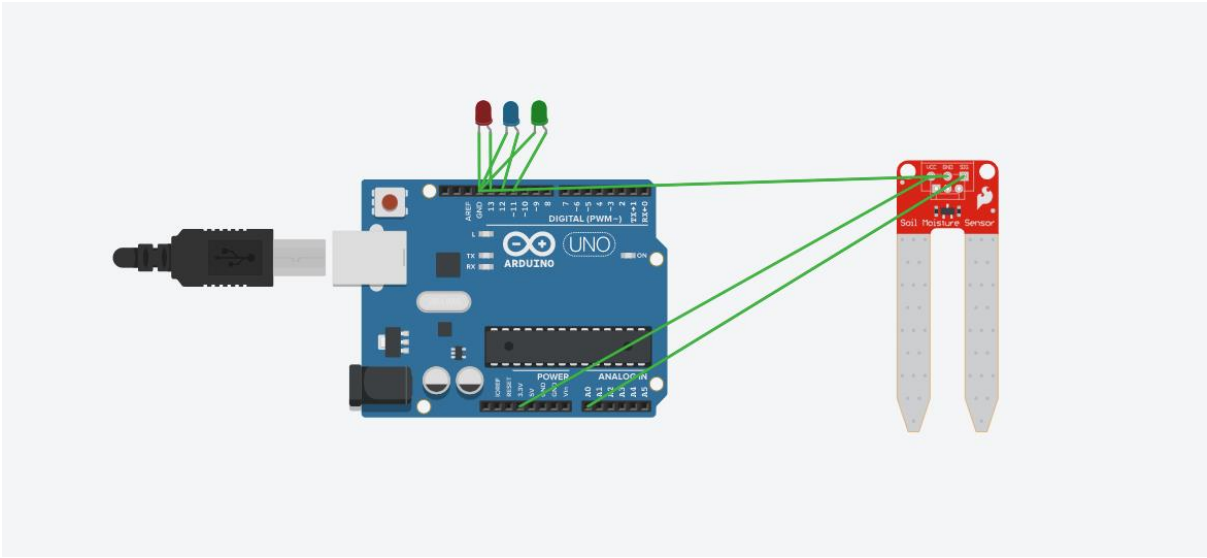
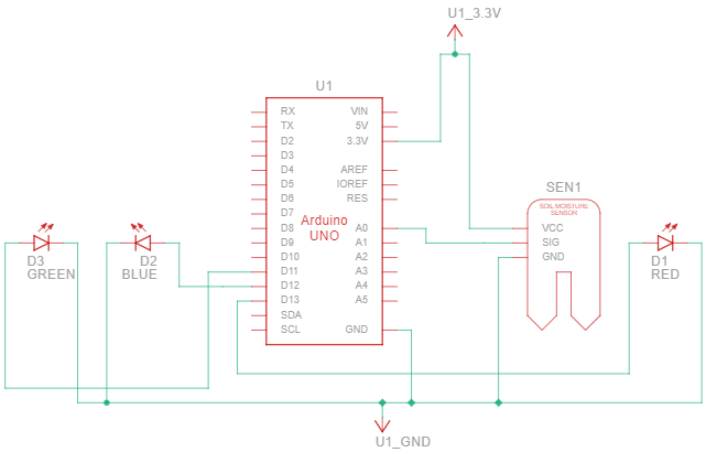
EXP 1: MONITOR THE SOIL MOISTURE LEVEL USING RASPBERRY PI



Name	Quantity	Component
SEN1	1	Soil Moisture Sensor
D1	1	Red LED
D2	1	Blue LED
D3	1	Green LED
U1	1	Arduino Uno R3



CODE:

```
int moistureValue;

float moisturePercentage;
```

```
void setup()
{
Serial.begin(9600);
pinMode(13, OUTPUT);
pinMode(12, OUTPUT);
pinMode(11, OUTPUT);
}

void loop()
{
moistureValue = analogRead(A0);
moisturePercentage = ((moistureValue/539.00) * 100);
Serial.print("Moisture Value: ");
Serial.print(moistureValue);
Serial.print("\nMoisture Percentage: ");
Serial.print(moisturePercentage);
if (moisturePercentage < 15.00)
{
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
if ((moisturePercentage < 70.00) && (moisturePercentage >= 15.00))
{
digitalWrite(12, HIGH);
```

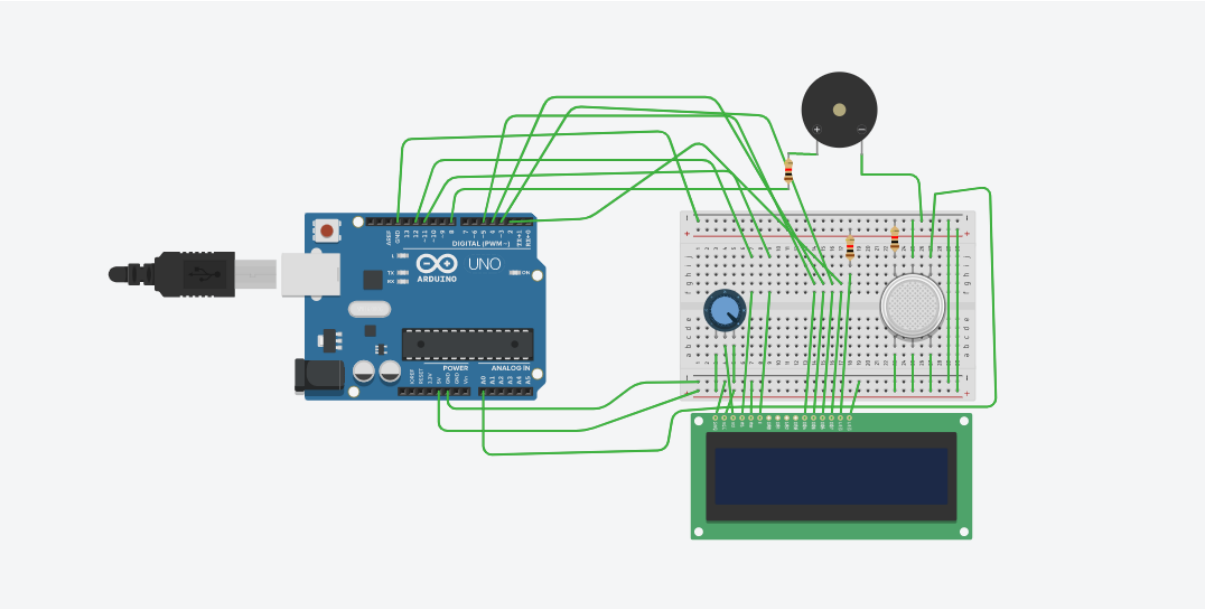
```
delay(1000);
digitalWrite(12, LOW);
delay(1000);
}
if ((moisturePercentage <= 100.00) && (moisturePercentage >= 70.00))
{
digitalWrite(11, HIGH);
delay(1000);
digitalWrite(11, LOW);
delay(1000);
}
Serial.print("\n*\n");
delay(1000);
}
```

OUTPUT:

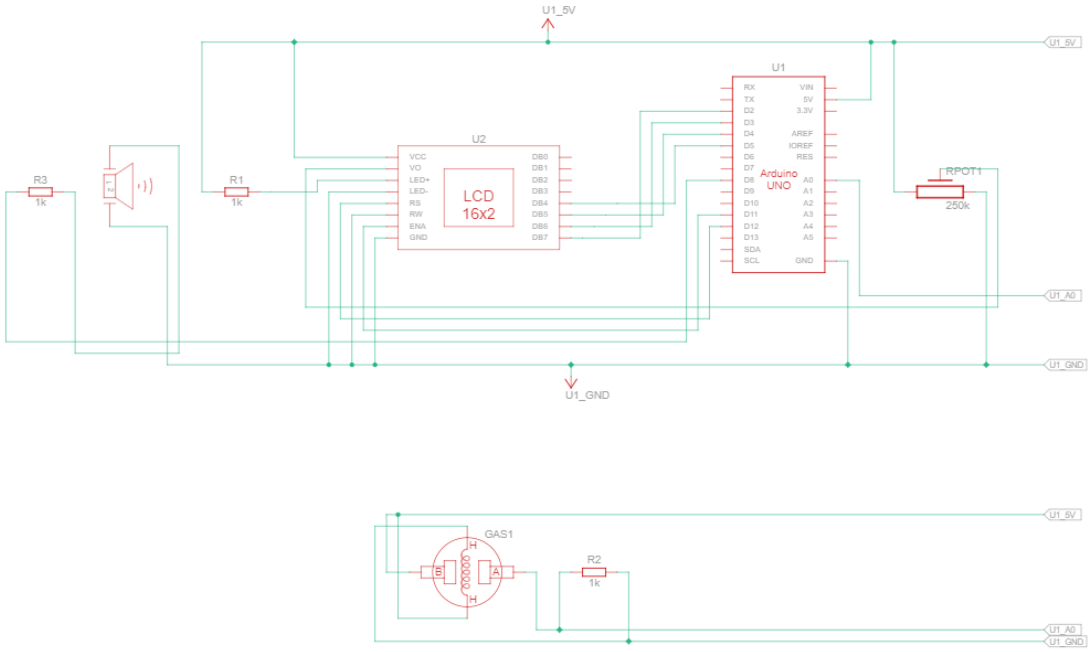
MOISTURE VALUE: 324

MOISTURE PERCENTAGE: 60.11

EXP 2: MONITOR THE SURROUNDING AIR QUALITY



Name	Quantity	Component
U1	1	Arduino Uno R3
Rpot1	1	250 kΩ Potentiometer
R1 R2 R3	3	1 kΩ Resistor
PIEZO1	1	Piezo
U2	1	LCD 16 x 2
GAS1	1	Gas Sensor



CODE:

```
#include <LiquidCrystal.h>
```

```
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
int pin = 8;
int analogPin = A0;
int sensorValue = 0;
void setup()
{
  pinMode(analogPin, INPUT);
  pinMode(pin, OUTPUT);
  lcd.begin(16, 2);
  lcd.print("what is the air");
  lcd.print("Quality Today");
  Serial.begin(9600);
  lcd.display();
}
void loop()
{
  delay(100);
  sensorValue = analogRead(analogPin);
  Serial.print("air Quality in RPM=");
  Serial.println(sensorValue);
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Air Quality:");
  lcd.print(sensorValue);
  if (sensorValue<=500)
  {
```

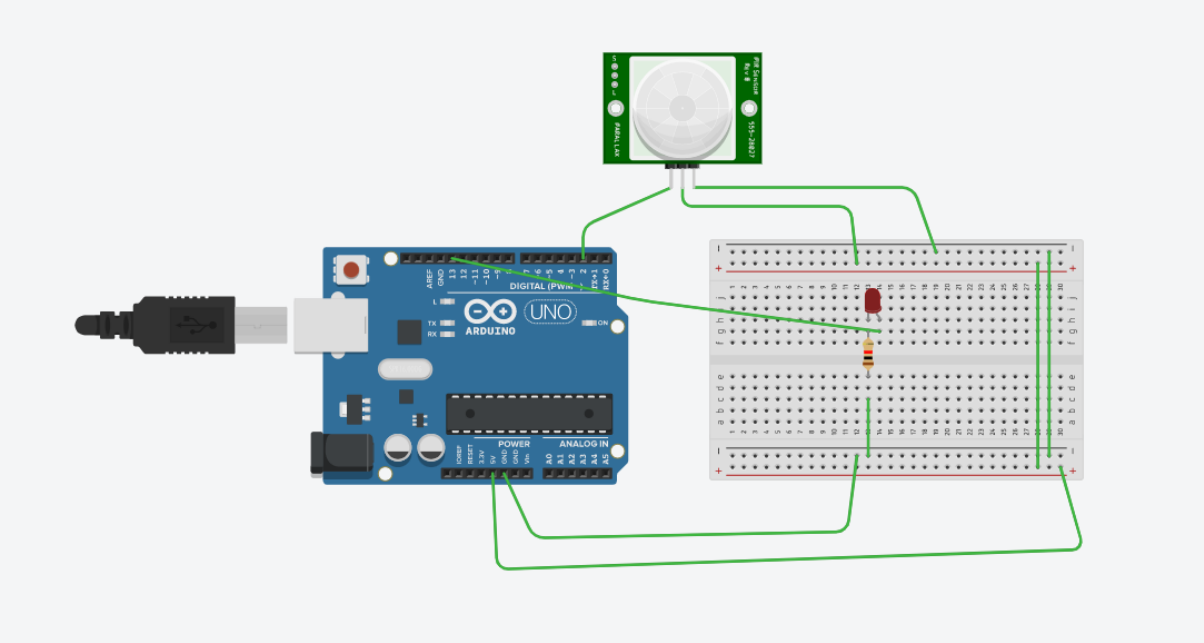
```
  Serial.print("Fresh air");
  Serial.print("\r\n");
  lcd.setCursor(0,1);
  lcd.print("Fresh Air");
 }
 else if(sensorValue>=500 && sensorValue<=600)
 {
  Serial.print("Poor Air");
  Serial.print("\r\n");
  lcd.setCursor(0,1);
  lcd.print("Poor Air");
 }
 else if (sensorValue>=650)
 {
  Serial.print("Very Poor Air");
  Serial.print("\r\n");
  lcd.setCursor(0,1);
  lcd.print("Very Poor Air");
 }
 if (sensorValue > 650) {
  digitalWrite(pin, HIGH);
 } else {
  digitalWrite(pin, LOW);
 }
}
```

OUTPUT:

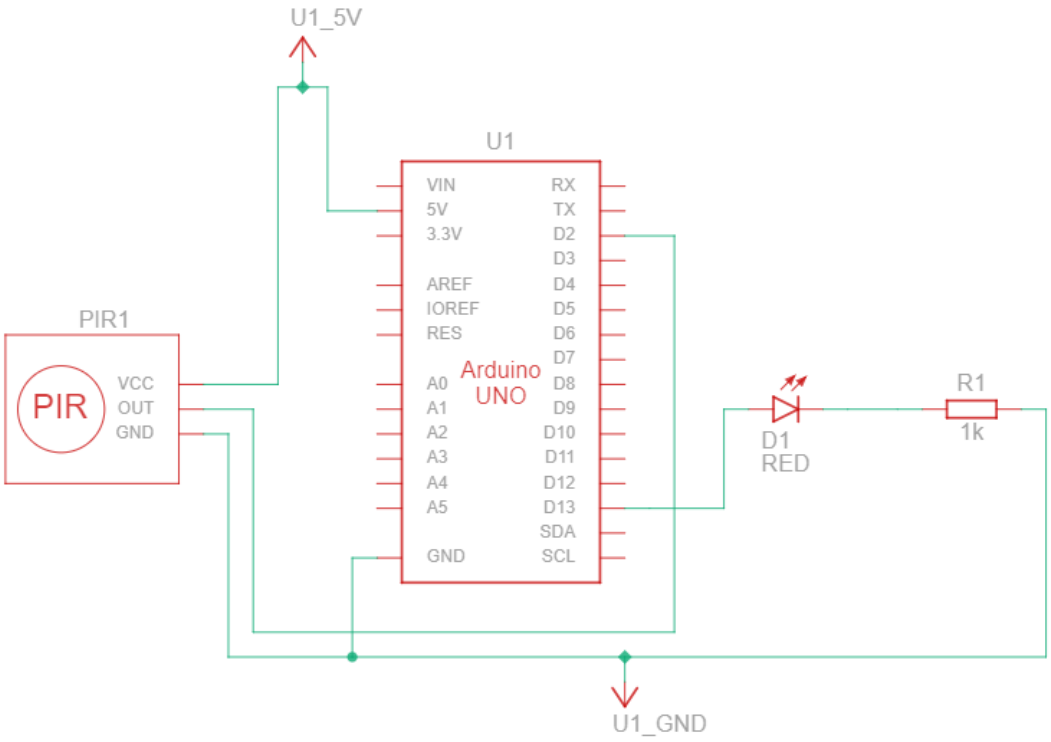
Air Quality in RPM = 101

Fresh Air

EXP 3: INTERFACE PIR SENSOR WITH ESP8266 WIFI MODULE ON ARDUINO PLATFORM



Name	Quantity	Component
U1	1	Arduino Uno R3
PIR1	1	-15.438599043954525 , -195.17093550710402 , -333.42392256112953 PIR Sensor
R1	1	1 kΩ Resistor
D1	1	Red LED



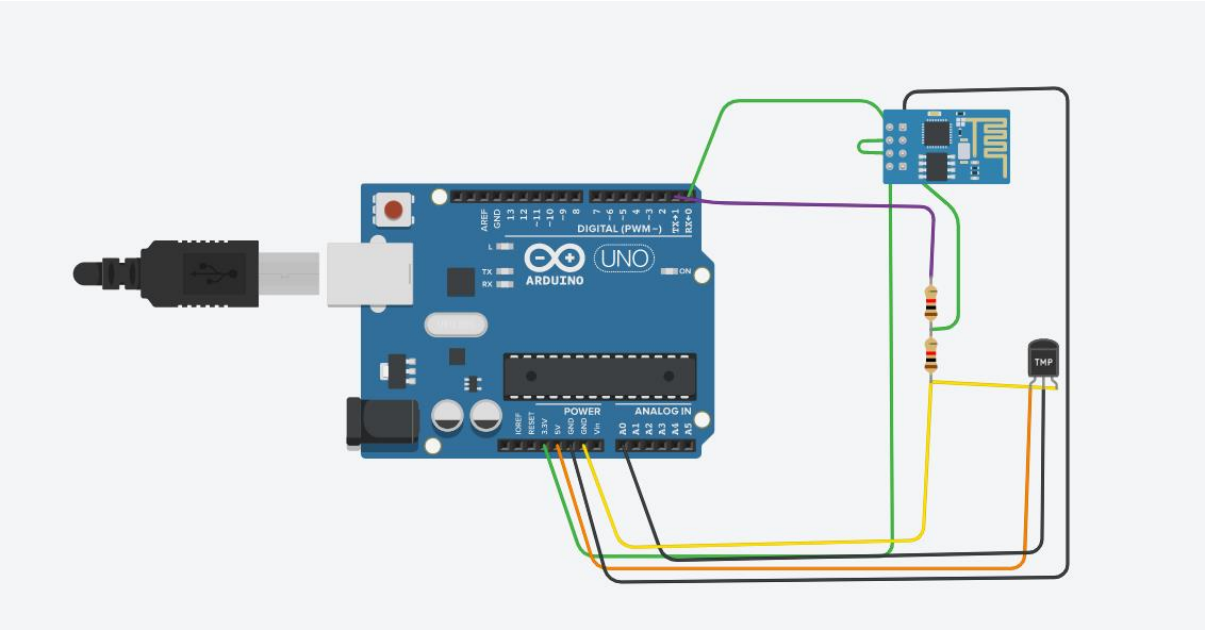
CODE:

```
int pirsensor = 0;
void setup()
{
 pinMode(13, OUTPUT);
 pinMode(2, INPUT);
}
void loop()
{
 pirsensor = digitalRead(2);
 if(pirsensor==HIGH)
 {
 digitalWrite(13, HIGH);
 }
```

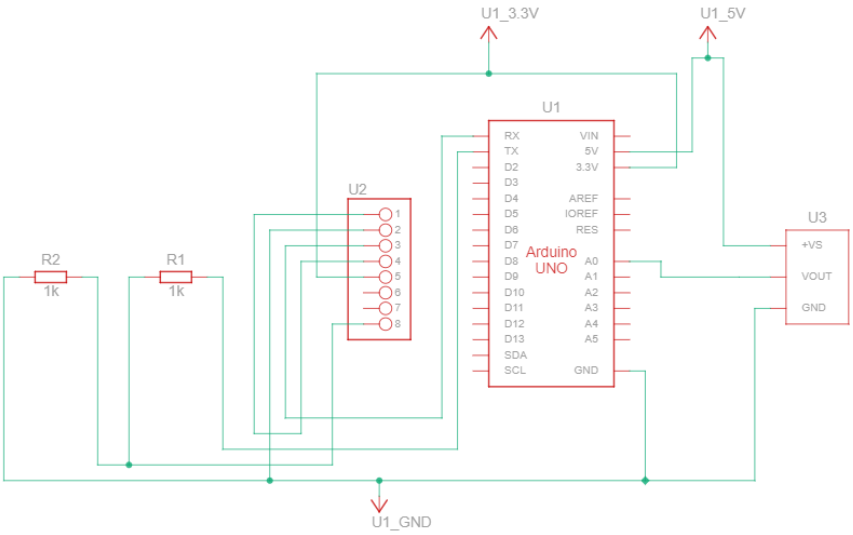
```
else
{
digitalWrite(13, LOW);
}
delay(10);

}
```

EXP 4: POSTING TEMPERATURE & HUMIDITY OF ENVIRONMENT TO THING SPEAK CLOUD USING NODE MCU



Name	Quantity	Component
U1	1	Arduino Uno R3
U2	1	Wifi Module (ESP8266)
R1 R2	2	1 kΩ Resistor
U3	1	Temperature Sensor [TMP36]



CODE:

```
float val, voltage, temp;
String ssid = "Simulator Wifi";
String password = "";
String host = "api.thingspeak.com";

const int httpPort = 80;
String url = "/update?api_key=4J2ET3G2MFQRYELM&field1=";
void setupESP8266(void)
{
Serial.begin(115200);
Serial.println("AT");
delay(10);
if (Serial.find("OK"))
Serial.println("ESP8266 OK!!!");
Serial.println("AT+CWJAP=\"" + ssid + "\",\"" + password + "\"");
delay(10);
```

```
if (Serial.find("OK"))
Serial.println("Connected to WiFi!!!");
Serial.println("AT+CIPSTART=\"TCP\",\"" + host + "\"," + httpPort);
delay(50);
if (Serial.find("OK"))
Serial.println("ESP8266 Connected to server!!!");
}
void anydata(void)
{
val=analogRead(A0);
voltage=val*0.0048828125;
temp = (voltage - 0.5) * 100.0;
String httpPacket = "GET " + url + String(temp) + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";
int length = httpPacket.length();
Serial.print("AT+CIPSEND=");
Serial.println(length);
delay(10);
Serial.print(httpPacket);

delay(10);
if (Serial.find("SEND OK\r\n"))
Serial.println("ESP8266 sends data to the server");
}
void setup()
{
pinMode(A0, INPUT);
```

```
setupESP8266();

}

void loop()

{

anydata();

delay(4000);

}
```

EXP 7: Forward kinematics and validate using robo analyser

```
import numpy as np

# Define the joint angles in radians
theta1 = np.pi/6
theta2 = np.pi/4
theta3 = np.pi/3
theta4 = np.pi/2

# Define the lengths of the links
L1 = 2
L2 = 1.5
L3 = 1
L4 = 0.5

# Calculate the end effector position
x = L1*np.cos(theta1) + L2*np.cos(theta1+theta2) +
L3*np.cos(theta1+theta2+theta3) + L4*np.cos(theta1+theta2+theta3+theta4)
y = L1*np.sin(theta1) + L2*np.sin(theta1+theta2) + L3*np.sin(theta1+theta2+theta3)
+ L4*np.sin(theta1+theta2+theta3+theta4)

print(f"The end effector position is ({x}, {y})")
```

Output:

```
The end effector position is (1.0596192034428373, 2.8024421300268756)
```

EXP 8: Inverse kinematics of real robot and validate using robo analyser

```
import numpy as np

# Define the end effector position
x = 1.5
y = 0.5
```

```
# Define the link lengths
L1 = 2
L2 = 1.5

# Calculate the joint angles using inverse kinematics
theta2 = np.arccos((x**2 + y**2 - L1**2 - L2**2) / (2 * L1 * L2))
theta1 = np.arctan2(y, x) - np.arctan2(L2 * np.sin(theta2), L1 + L2 * np.cos(theta2))

print(f"The joint angles are ({theta1}, {theta2}) in radians")
```

Output:

The joint angles are (-0.5121611931861503, 2.2459278597319283) in radians

EXP 9: Use of source computer vision programming tool opencv

image and video processing functions, such as image filtering, edge detection, and object detection. These functions can be used to perform various image and video processing tasks, such as image enhancement, motion detection, and face recognition.

Robotics: OpenCV can be used to develop computer vision algorithms for robots, such as object detection, tracking, and recognition. These algorithms can be used to help robots navigate in a complex environment, identify objects, and interact with the world around them.

Augmented reality: OpenCV can be used to create augmented reality applications that superimpose digital content over real-world images or videos. This can be used for a variety of applications, such as games, advertising, and education.

Machine learning: OpenCV provides support for machine learning algorithms, such as support vector machines (SVMs), neural networks, and decision trees. These algorithms can be used to develop computer vision applications that can learn and improve over time.

Medical imaging: OpenCV can be used to develop medical imaging applications, such as X-ray image analysis and medical diagnosis. It provides developers with a range of image processing functions that can be used to enhance medical images and extract useful information from them.

Overall, OpenCV is a versatile and powerful computer vision programming tool that can be used for a wide range of applications. Its open-source nature makes it accessible to developers all over the world, and its extensive documentation and community support make it easy to learn and use.

EXP 10: Image processing using opencv

```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
img = cv2.imread('jawlines.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5, 5), 0)
edges = cv2.Canny(blur, 50, 150)
kernel = np.ones((5,5),np.uint8)
dilated = cv2.dilate(edges, kernel, iterations = 1)
cv2_imshow(img)
cv2_imshow(gray)
cv2_imshow(blur)
cv2_imshow(edges)
cv2_imshow(dilated)
cv2.waitKey(0)
cv2.destroyAllWindows()
```