

## Tutorial Visión Artificial para controlar Arduino

LabTec

Aquí tienen un tutorial creado por el LabTec para trabajar con el código que identifica los dedos de una mano y controla una placa Arduino usando Python, OpenCV, MediaPipe y pyFirmata.

El trabajo se divide en dos momentos, uno para preparar el entorno en Python, y otro dedicado a preparar el circuito en Arduino.

### Paso 1: Preparar el entorno

#### 1. Instala Python:

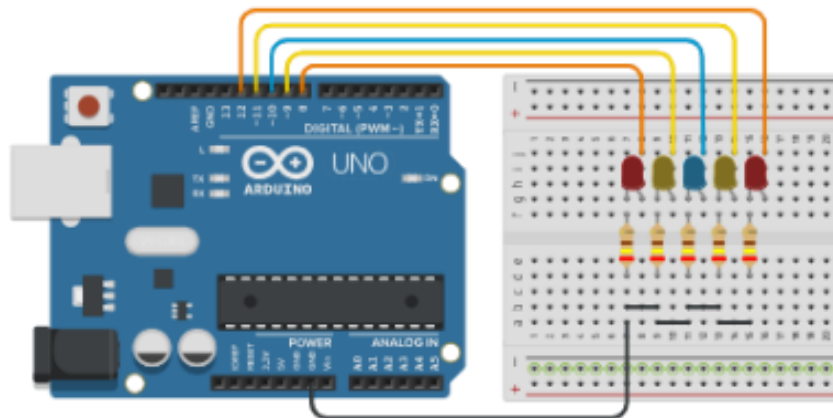
- Con Python 3.10.11 MediaPipe anda bien. Se han reportado fallas de MediaPipe con versiones superiores de Python. Instalar desde [Python Release Python 3.10.11 | Python.org](https://www.python.org/downloads/release/python-31011/)

#### 2. Instalar las bibliotecas necesarias en Python:

- `pip install opencv-python mediapipe numpy pyfirmata`

#### 2. Configurar Arduino:

- Arma el circuito que se muestra en la imagen. También lo puedes ver en el siguiente link <https://www.tinkercad.com/things/bEn7D7bxxMv-vamanoarduino>



- Conecta tu placa Arduino a tu computadora.
- Abre el IDE de Arduino y carga el ejemplo StandardFirmata en la placa. Ruta: Examples -> Firmata -> StandardFirmata

Nota el puerto COM al que está conectada la placa (por ejemplo, COM3. Pero podría ser otro, identifique adecuadamente el puerto COM al que se conecta Arduino).

## Paso 2: Escribir el código

### 1. Importar las bibliotecas necesarias:

```
import cv2
import mediapipe as mp
import numpy as np
from math import acos, degrees
from pyfirmata import Arduino
```

### 2. Inicializar MediaPipe y la cámara:

```
mp_drawing = mp.solutions.drawing_utils
mp_hands = mp.solutions.hands
cap = cv2.VideoCapture(0)
board = Arduino("COM3")
```

### 3. Definir funciones auxiliares:

```
def palm_centroid(coordinates_list):
    coordinates = np.array(coordinates_list)
    centroid = np.mean(coordinates, axis=0)
    centroid = int(centroid[0]), int(centroid[1])
    return centroid
```

### 4. Configurar puntos de referencia para los dedos:

```
thumb_points = [1, 2, 4]
palm_points = [0, 1, 2, 5, 9, 13, 17]
fingertips_points = [8, 12, 16, 20]
finger_base_points = [6, 10, 14, 18]
```

### 5. Procesar los fotogramas de la cámara:

```
with mp_hands.Hands(
    model_complexity=1,
    max_num_hands=1,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as hands:

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Ignorar fotograma vacío")
            break

        frame = cv2.flip(frame, 1)
        height, width, _ = frame.shape
        results = hands.process(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        thickness = [1, 1, 1, 1, 1]

        if results.multi_hand_landmarks:
            coordinates_thumb = []
```

```

coordinates_palm = []
coordinates_ft = []
coordinates_fb = []

for hand_landmarks in results.multi_hand_landmarks:
    for index in thumb_points:
        x = int(hand_landmarks.landmark[index].x * width)
        y = int(hand_landmarks.landmark[index].y * height)
        coordinates_thumb.append([x, y])

    for index in palm_points:
        x = int(hand_landmarks.landmark[index].x * width)
        y = int(hand_landmarks.landmark[index].y * height)
        coordinates_palm.append([x, y])

    for index in fingertips_points:
        x = int(hand_landmarks.landmark[index].x * width)
        y = int(hand_landmarks.landmark[index].y * height)
        coordinates_ft.append([x, y])

    for index in finger_base_points:
        x = int(hand_landmarks.landmark[index].x * width)
        y = int(hand_landmarks.landmark[index].y * height)
        coordinates_fb.append([x, y])

# Calcular el ángulo del pulgar
p1 = np.array(coordinates_thumb[0])
p2 = np.array(coordinates_thumb[1])
p3 = np.array(coordinates_thumb[2])

l1 = np.linalg.norm(p2 - p3)
l2 = np.linalg.norm(p1 - p3)
l3 = np.linalg.norm(p1 - p2)

angle = degrees(acos((l1**2 + l3**2 - l2**2) / (2 * l1 * l3)))
thumb_finger = angle > 160

# Calcular la posición de los otros dedos
nx, ny = palm_centroid(coordinates_palm)
cv2.circle(frame, (nx, ny), 3, (255, 0, 0), 2)
coordinates_centroid = np.array([nx, ny])
coordinates_ft = np.array(coordinates_ft)
coordinates_fb = np.array(coordinates_fb)

d_centrid_ft = np.linalg.norm(coordinates_centroid - coordinates_ft, axis=1)
d_centrid_fb = np.linalg.norm(coordinates_centroid - coordinates_fb, axis=1)
dif = d_centrid_ft - d_centrid_fb
fingers = dif > 0

```

```

    fingers = np.append(thumb_finger, fingers)

    for i, finger in enumerate(fingers):
        if finger:
            thickness[i] = 0

    mp_drawing.draw_landmarks(frame, hand_landmarks,
mp_hands.HAND_CONNECTIONS)

    # Controlar Arduino
    board.digital[8].write(thickness[0] == 0)
    board.digital[9].write(thickness[1] == 0)
    board.digital[10].write(thickness[2] == 0)
    board.digital[11].write(thickness[3] == 0)
    board.digital[12].write(thickness[4] == 0)

    else:
        for pin in range(8, 13):
            board.digital[pin].write(0)

    cv2.imshow("Frame", frame)

    if cv2.waitKey(1) & 0xFF == 27:
        break

cap.release()
cv2.destroyAllWindows()

```

### **Paso 3: Ejecutar el código**

1. Guarda el código en un archivo, por ejemplo, hand\_detection\_arduino.py.
2. Ejecuta el archivo desde la terminal: python hand\_detection\_arduino.py

¡Espero que este tutorial te sea útil! Si tienes alguna pregunta, no dudes en escribirnos a [labtec@umce.cl](mailto:labtec@umce.cl)