

# Ejercicios Sigue-Persona para la plataforma académica Robotics Academy, usando un robot real y simulado en ROS2.

**Trabajo Fin de Grado (2021/2022)**

Alumno: Carlos Caminero Abad  
Tutor: Jose María Cañas Plaza

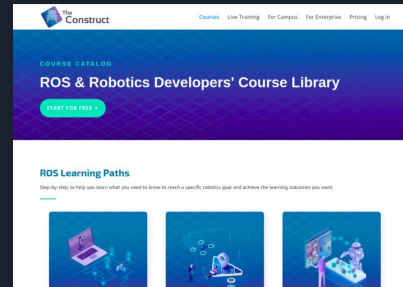


# ÍNDICE

1. Introducción
2. Motivación y Objetivos
3. Herramientas utilizadas
4. Soporte del TurtleBot2 en ROS2 Foxy
5. Ejercicios Sigue-Persona en Robotics Academy
  - a. *Plugin* de Teleoperación
  - b. HAL API - ROS *topics*
  - c. Plantillas web
6. Soluciones de referencia
  - a. Algoritmo de seguimiento visual (Tracking)
  - b. Campo de Fuerzas virtuales (VFF)
  - c. Máquina de Estados Finitos
  - d. Validación Experimental Sigue-Persona simulado
  - e. Validación Experimental Sigue-Persona real
7. Conclusiones

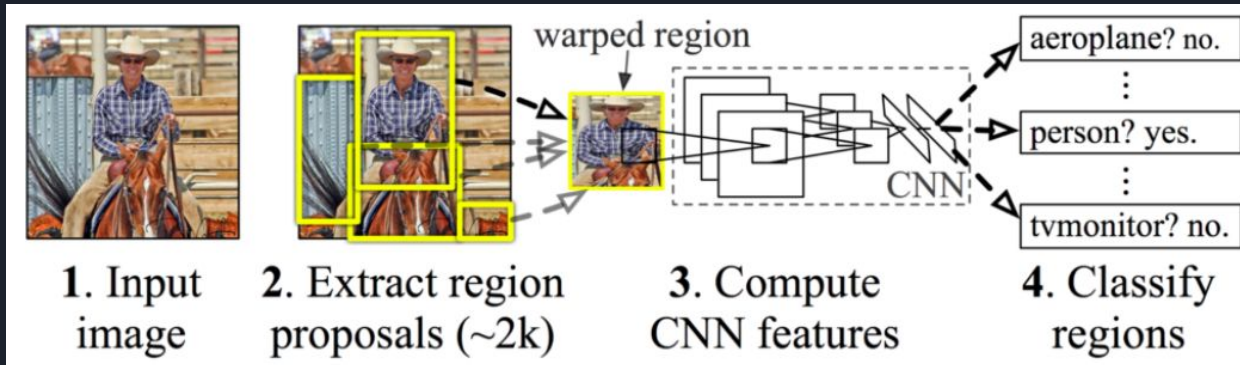
# INTRODUCCIÓN

- La robótica aporta cada vez más servicios útiles a la sociedad.
- Es importante la formación profesional del ingeniero robótico.
- Robótica Educativa
  - The Construct
  - GIRS de la ETSIT (URJC)
  - Robotics Academy y Unibotics



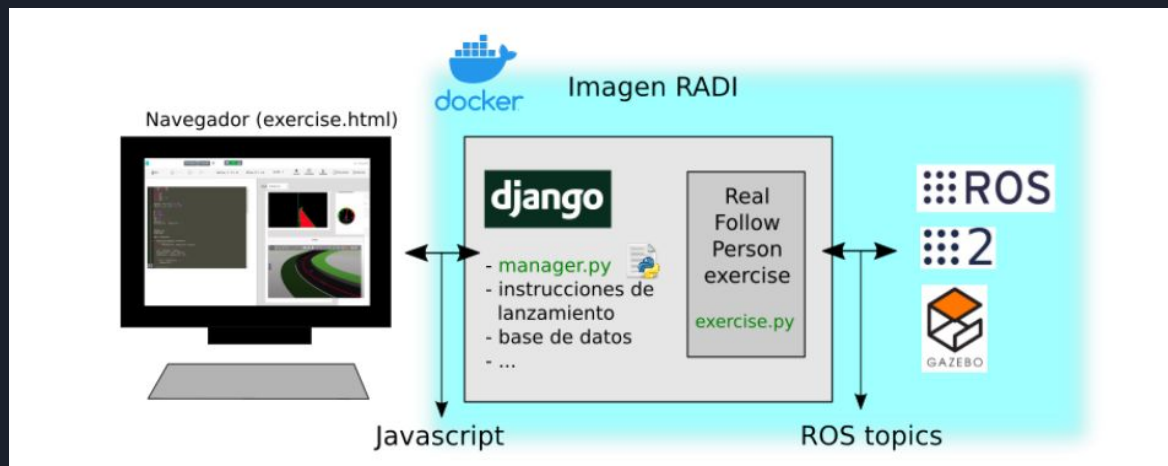
# MOTIVACIÓN

- Deep Learning abre un abanico de aplicaciones robóticas.
- Redes Neuronales Convolucionales basadas en Regiones (R-CNN).
- De la clasificación a la detección de objetos.
- RNA óptima para sistemas computacionales de bajo rendimiento (SSD Inception V2)



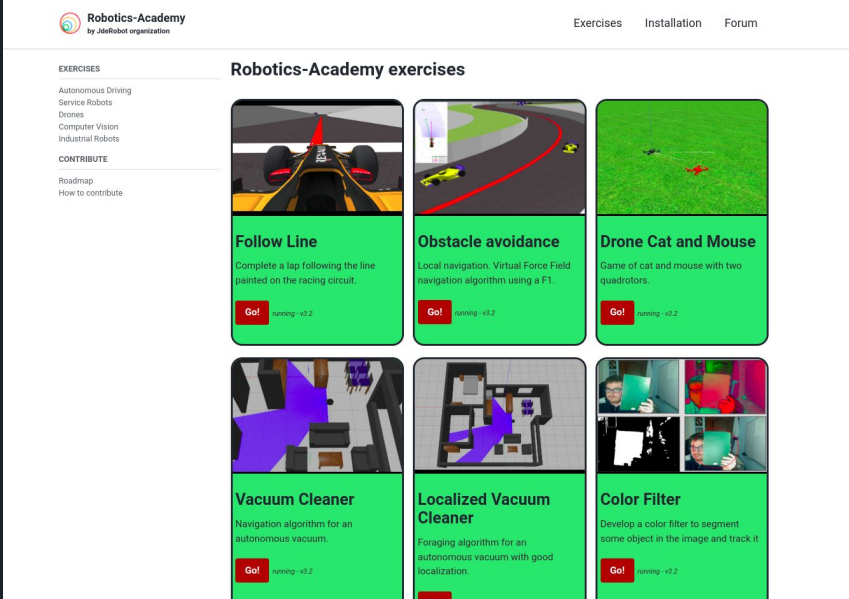
# MOTIVACIÓN

- Programación de un robot real + capa de abstracción (HAL y GUI).



# OBJETIVOS

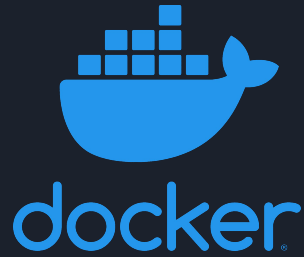
- Dos ejercicios para Robotics Academy.
- Soporte del TurtleBot2 simulado y real (ROS2 Foxy).
- Los alumnos programan un algoritmo de seguimiento con Deep Learning.



The screenshot displays the Robotics-Academy website interface. At the top, the logo "Robotics-Academy by JdeRobot organization" is on the left, and navigation links "Exercises", "Installation", and "Forum" are on the right. A left sidebar contains links for "EXERCISES" (Autonomous Driving, Service Robots, Drones, Computer Vision, Industrial Robots) and "CONTRIBUTE" (Roadmap, How to contribute). The main content area is titled "Robotics-Academy exercises" and features a grid of six exercise cards, each with a thumbnail image, a title, a description, and a "Go!" button with a version number.

Exercise Title	Description	Version
Follow Line	Complete a lap following the line painted on the racing circuit.	running - v3.2
Obstacle avoidance	Local navigation, Virtual Force Field navigation algorithm using a F1.	running - v3.2
Drone Cat and Mouse	Game of cat and mouse with two quadrotors.	running - v3.2
Vacuum Cleaner	Navigation algorithm for an autonomous vacuum.	running - v3.2
Localized Vacuum Cleaner	Foraging algorithm for an autonomous vacuum with good localization.	running - v3.2
Color Filter	Develop a color filter to segment some object in the image and track it	running - v3.2

# HERRAMIENTAS UTILIZADAS



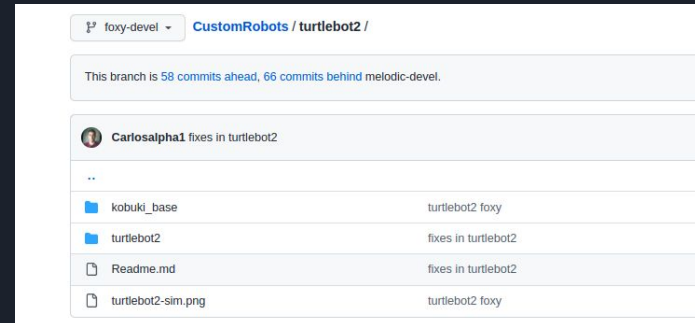
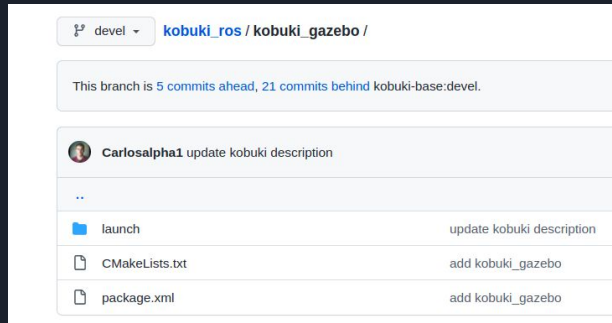
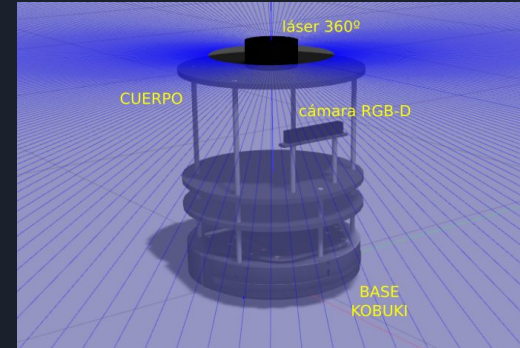
URDF

Xacro



# SOPORTE DEL TURTLEBOT2 EN ROS2 FOXY

- Drivers de la base Kobuki real en ROS2 (más *wrappers*).
- ¿y para la simulación?
- Diseño del TurtleBot2 en Gazebo más simulación de los sensores y actuadores.





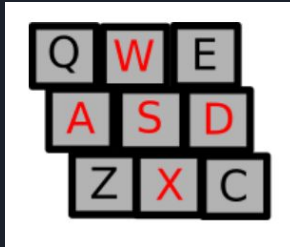
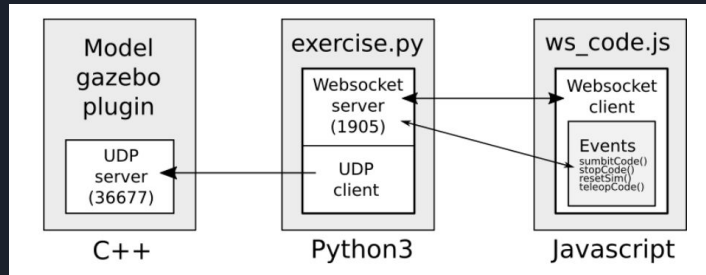
# EJERCICIOS SIGUE-PERSONA EN ROBOTICS ACADEMY

- Desarrollo del frontend de cada ejercicio
  - plantillas web.
- Desarrollo del entorno simulado
  - *plugin* (teleoperación - C++) e integración hospital AWS.
- Creación de *Launchers ROS2*.
- Comunicación con ROS *topics*
  - Plantillas python (HAL/GUI e *interfaces*) adaptadas a ROS2.



# EJERCICIOS SIGUE-PERSONA EN ROBOTICS ACADEMY. Plugin de Teleoperación

- Controlar una persona simulada



ws\_code.js

```
// Key events to move the model
window.onkeydown = (e) => {
  if (activate_teleop) {
    key_pressed = e.key;
    websocket_code.send("#key_"+key_pressed);
  }
}
```

exercise.py

```
if (message[:4] == "#key"):
    mode = message[message.find('_')+1:]
    if mode == "w":
        self.model_client.sendto(str.encode("UVF"), self.model_address)
    elif mode == "s":
        self.model_client.sendto(str.encode("UVB"), self.model_address)
```

person.cpp

```
void ServerThreadLoop(void)
{
    char msg[3];

    while (true) {
        recv_message(this->sockfd, this->addr, &msg, sizeof msg);
```

# EJERCICIOS SIGUE-PERSONA EN ROBOTICS ACADEMY. HAL API - ROS topics

- setV() y setW() :
  - /cmd\_vel
  - /commands/velocity
- getLaserData():
  - /scan
- getImage():
  - /depth\_camera/image\_raw
  - /image\_raw
- getPose3d():
  - /odom
- getBoundingBoxes():
  - Llama a SSD Inception

```
class HAL:

    def __init__(self):
        # init node
        rclpy.init()

        self.motors = PublisherMotors("cmd_vel", 4, 0.3)
        self.laser = ListenerLaser("scan")
        self.camera = ListenerCamera("/depth_camera/image_raw")
        self.odometry = ListenerPose3d("/odom")

        self.listener_executor = MultiThreadedExecutor(num_threads=4)
        self.listener_executor.add_node(self.laser)
        self.listener_executor.add_node(self.camera)
        self.listener_executor.add_node(self.odometry)

        self.net = NeuralNetwork()

        # Update thread
        self.thread = ThreadHAL(self.listener_executor)

    def start_thread(self):
        self.thread.start()

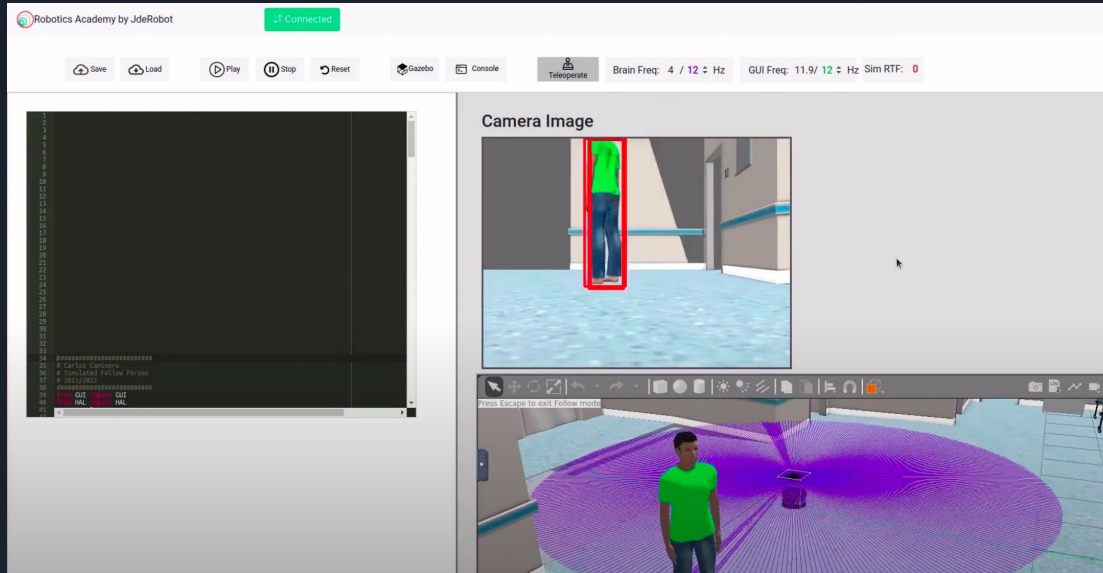
    def setV(self, velocity):
        self.motors.sendV(velocity)

    def setW(self, velocity):
        self.motors.sendW(velocity)

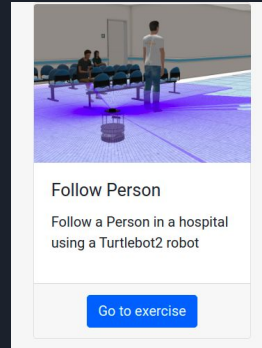
    def getLaserData(self):
        values = self.laser.getLaserData().values
        return values[90:0:-1] + values[0:1] + values[360:270:-1]

    def getImage(self):
        return self.camera.getImage().data
```

# Plantilla web Sigue-Persona simulado



- editor texto
- botón teleoperación
- *canvas* imagen
- ventana simulador
- ventana terminal




# Plantilla web Sigue-Persona Real

Robotics Academy by JdeRobot Connected

Save Load Play Stop Reset Console Brain Freq: 8.5 / 12 Hz GUI Freq: 12/ 12 Hz

```
1 #####
2 # Carlos Cantares
3 # Simulated Follow Person
4 # 2021/2021
5 #####
6 from GUI import GUI
7 from HAL import HAL
8 import cv2
9 import math
10
11 SEARCH_PERSON = 0
12 FOLLOW_PERSON = 1
13
14 state = SEARCH_PERSON
15
16 class BoundingBoxObject:
17
18     def __init__(self, bounding_box):
19         self.bounding_box = bounding_box
20         self.centroid = centroid_bounding_box(bounding_box)
21         self.area = area_bounding_box(bounding_box)
22
23
24 class Tracker:
25
26     def __init__(self, obj=None):
27         self.obj = None
28         self.limit = 50
29
30     def setObjective(self, obj):
31         self.obj = obj
32
33     def getObjective(self):
34         return self.obj
35
36     def getObjectiveFromGet(self, objlist):
37
38
39
40
41
```

Camera Image



- editor texto
- *canvas* imagen
- ventana terminal



## Real Follow Person

Follow Person exercise  
where the user will control a  
real Turtlebot2

[Go to exercise](#)

# SOLUCIONES DE REFERENCIA

- Dos soluciones de referencia.
- 3 pasos: *tracking*, VFF y Máquina de Estados.
- Diferente comportamiento en real y en simulado.

## Variantes alternativas:

- Control de velocidad basado en casos (franjas).
- Control de velocidad basado en un PID.
- Detección por color de ropa.

```
elif state == FOLLOW_PERSON:
    if len(filter3) > 0:
        centroids = []
        objects = []

        for bbox in filter3:
            objects.append(BoundingBoxObject(bbox))
            draw_bounding_box(img, bbox, color=(0, 255, 0), thickness=3)

        object2follow, index = tracker.getObjectiveFromSet(objects)
        if index >= 0:
            tracking_failure_cont = 0
            centroid = object2follow.centroid
            last_centroid = centroid

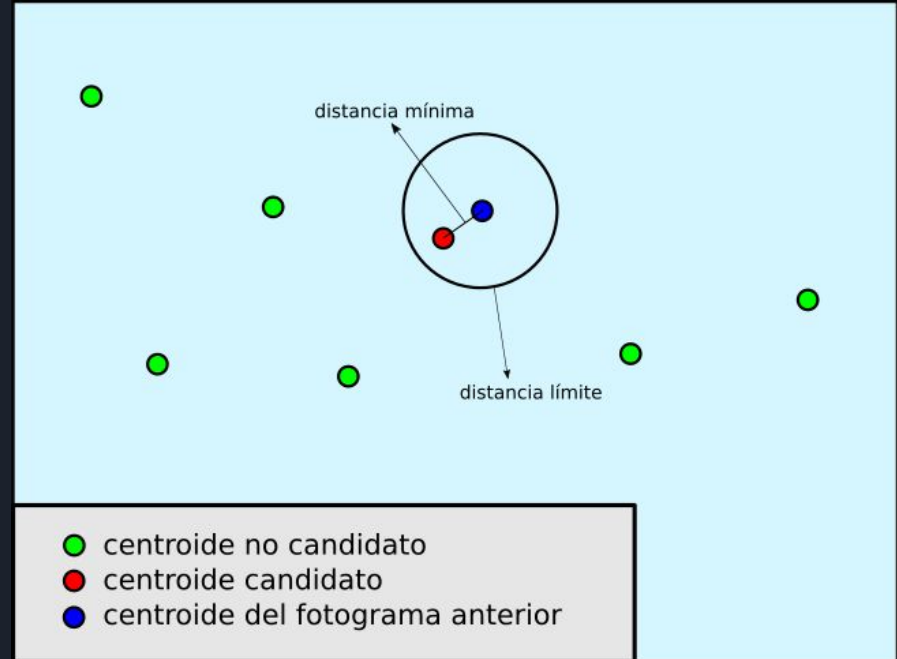
            # VFF Algorithm
            tx, ty = attraction_vector([centroid[0], width])
            rx, ry = repulsion_vector(parse_laser_data(HAL.getLaserData()))
            fx = alfa * tx + beta * rx # final vector (fx)
            fy = alfa * ty + beta * ry # final vector (fy)

            if object2follow.area < 16000:
                linear_vel = 0.2
            elif object2follow.area < 45000:
                linear_vel = 0.1
            else:
                linear_vel = 0.0
            draw_bounding_box(img, filter3[index], color=(0, 0, 255), thickness=3)
        else:
            tracking_failure_cont += 1
            linear_vel = 0.1

    HAL.setW(-fx)
    HAL.setV(linear_vel)
```

# Algoritmo de seguimiento visual (tracking)

- Uso de R-CNN
- Filtros: *score*, clase, área.
- Clase Tracker (Actualiza objetivo)
- Tolerancia a la pérdida de detecciones en fotogramas.

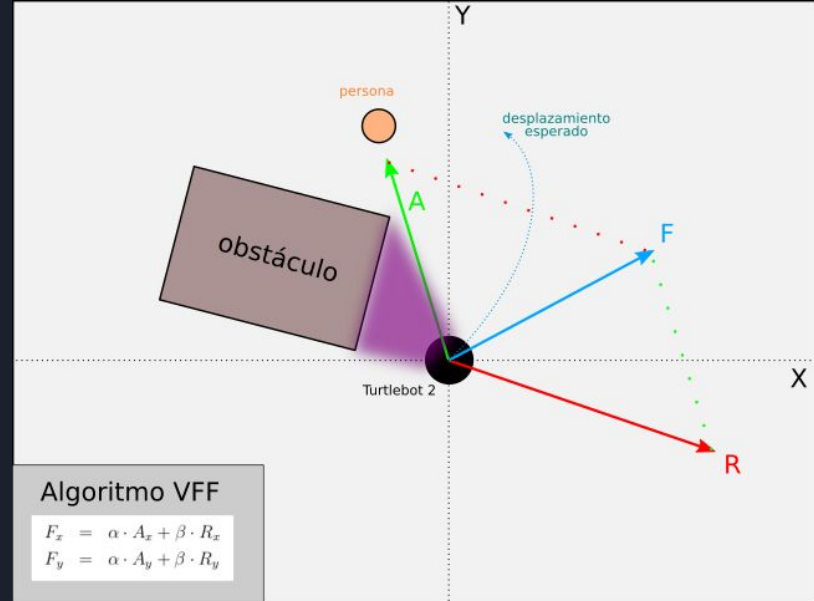


# Campo de Fuerzas Virtuales (VFF)

- Movimiento = suma vectorial:
  - Vector A -> objetivo (FOV)
  - Vector R -> Repulsión del láser.
- Pesos de los vectores:
  - $\alpha$  - Vector de Atracción.
  - $\beta$  - Vector de Repulsión.

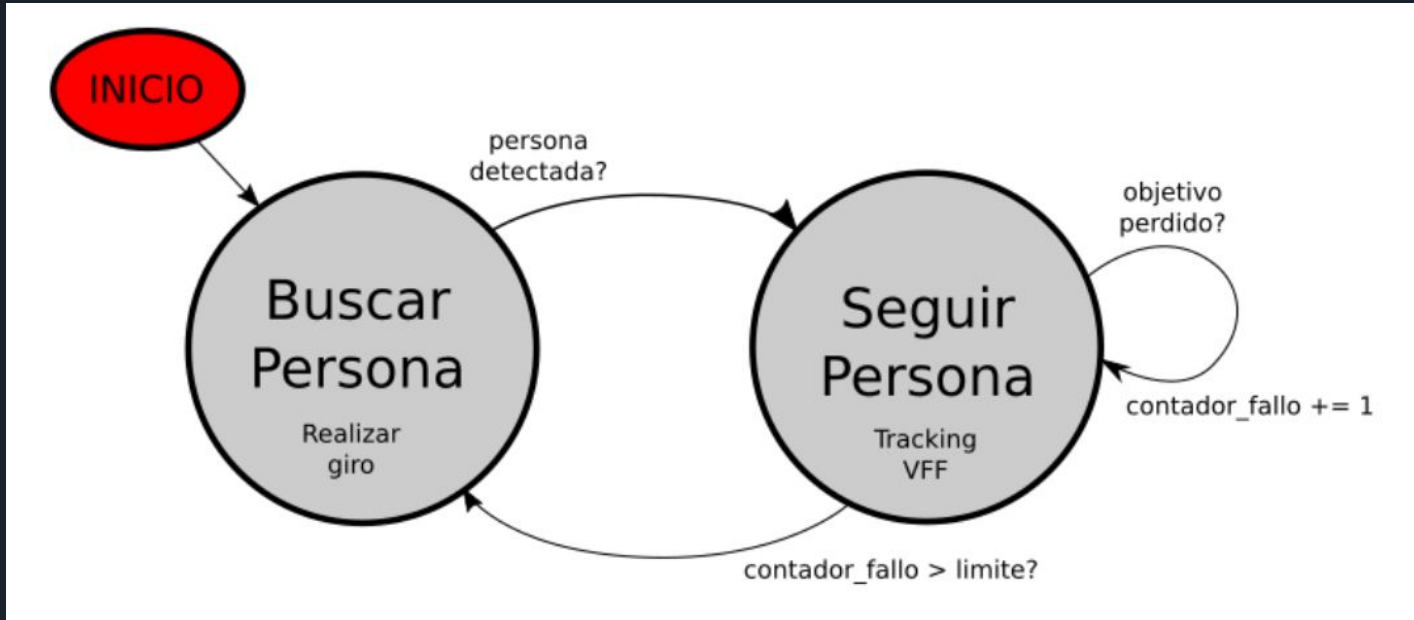
$$F_x = \alpha \cdot A_x + \beta \cdot R_x$$

$$F_y = \alpha \cdot A_y + \beta \cdot R_y$$





# Máquina de Estados Finitos (FSM)



# Validación experimental Sigue-Persona Simulado

## **Simulated Follow Person Robotics Academy**



[https://www.youtube.com/  
watch?v=fDAU465eVxQ](https://www.youtube.com/watch?v=fDAU465eVxQ)

# Validación experimental Sigue-Persona Real

**Real Follow Person  
Robotics Academy**



<https://www.youtube.com/watch?v=54Jb4KJwyDM>



# CONCLUSIONES. Aportaciones

- TurtleBot2 real y simulado en ROS2 Foxy.
- Dos nuevos ejercicios para Robotics Academy.
- Nuevo escenario simulado y plugin de teleoperación.
- Un robot real en un ejercicio de RA.
- Soluciones de referencia.
- Red Neuronal SSD Inception en RADI.



# CONCLUSIONES. Líneas Futuras

- Nuevos ejercicios con el TurtleBot2
- Soporte de cámaras RGB-D.



Fin

¿PREGUNTAS?  
¡MUCHAS GRACIAS!

