Ingeniería en Inteligencia Artificial
Sem: 2024-2, 4BV1, Proyecto

Implementación de un Alimentador Automático Basado en Visión Artificial para Gatos Utilizando Raspberry Pi y Arduino

Docente: Hernández Cruz Macario
Unidad de aprendizaje: Fundamentos de Inteligencia Artificial

Fecha: 2024-06-29

Alumno(s):
Crombie Esquinca Alexander Iain
Alvarado Sandoval Juan Manuel
Luna Gonzalez Gabriel Alexis

Contents

1 Resumen

Este proyecto describe el desarrollo y la implementación de un alimentador automático para gatos, basado en un sistema de visión artificial. El sistema se acciona de dos maneras: de forma automática y manual, y está diseñado para mejorar la alimentación de mascotas mediante el uso de tecnologías modernas como la Raspberry Pi y Arduino. Nombre del Paper: "Automatic Cat Feeder System Using Artificial Vision with Raspberry Pi and Arduino"

2 Introducción

El cuidado y la alimentación de las mascotas, en particular de los gatos, es una responsabilidad que requiere tiempo y atención constante. Sin embargo, en el ritmo de vida moderno, es común que los dueños no siempre dispongan del tiempo necesario para alimentar a sus gatos de manera regular. Para abordar esta necesidad, el presente proyecto introduce el desarrollo e implementación de un alimentador automático basado en visión artificial, utilizando tecnologías como la Raspberry Pi y Arduino.

Este sistema innovador pretende mejorar la alimentación de los gatos mediante un enfoque automatizado que combina la detección visual con la capacidad de dispensar alimentos de manera precisa y en intervalos específicos. La integración de la visión artificial permite la identificación del gato en tiempo real, garantizando que la comida sea dispensada sólo cuando es necesario, mientras que el uso de Raspberry Pi y Arduino proporciona una plataforma robusta y flexible para la gestión del sistema.

En este trabajo se detallan las etapas de diseño, implementación y prueba de un alimentador automático que opera en dos modos: automático, mediante la detección del gato y el control del horario de alimentación, y manual, a través de una aplicación web que permite al usuario dispensar comida a su conveniencia. La comunicación serial entre la Raspberry Pi y el Arduino, junto con el control preciso de un servomotor, constituye la base técnica del sistema, destacando su capacidad para integrarse de manera efectiva en el entorno doméstico.

2.1 Antecedentes del problema

La tenencia de mascotas, especialmente gatos, ha crecido significativamente en los últimos años. Con ello, también ha aumentado la necesidad de proporcionarles cuidados adecuados, que incluyen una alimentación regular y controlada. La rutina de alimentación puede volverse un desafío para los dueños que tienen horarios irregulares o están ausentes durante el día. Esto puede llevar a una alimentación inconsistente, exceso de comida, o, por el contrario, a la falta de alimento, lo que impacta negativamente en la salud de las mascotas.

Tradicionalmente, se han utilizado alimentadores manuales o de temporizador mecánico. Sin embargo, estos dispositivos tienen limitaciones en cuanto a flexibilidad y no pueden adaptarse a situaciones imprevistas o cambios en la rutina de alimentación. Además, la supervisión remota es prácticamente inexistente en estos modelos. Con el avance de la tecnología, especialmente en áreas como la visión artificial y la electrónica accesible como Raspberry Pi y Arduino, se abre la oportunidad para crear soluciones más sofisticadas que puedan automatizar y optimizar este proceso, proporcionando una mejor

calidad de vida tanto para las mascotas como para sus dueños.

2.2 Objetivos del proyecto

Objetivo General: Desarrollar e implementar un alimentador automático para gatos basado en un sistema de visión artificial que utilice una Raspberry Pi y Arduino, capaz de operar en modo automático y manual para asegurar una alimentación adecuada y oportuna de las mascotas.

Objetivos Específicos:

Diseñar un sistema de detección basado en visión artificial que identifique la presencia de gatos en el área de alimentación. Desarrollar una aplicación web que permita el accionamiento manual del alimentador y la visualización de imágenes en tiempo casi real (al ser que nada más se actualizá la imágen cada 1s aproximadamente) del área de alimentación. Implementar un mecanismo de control de un servomotor mediante Arduino, que permita la dispensación precisa de la comida. Integrar la Raspberry Pi con el sistema de detección y el mecanismo de control para operar en modo automático y manual, con comunicación serial entre los componentes.

2.3 Justificación y relevancia del proyecto

El proyecto de implementación de un alimentador automático para gatos es relevante por varias razones:

Mejora en la Calidad de Vida de las Mascotas: Un sistema automatizado asegura una alimentación regular y controlada, lo que es crucial para la salud y el bienestar de las mascotas. Evita problemas como la sobrealimentación o la falta de alimento cuando el dueño no está presente.

Conveniencia para los Dueños: Facilita la vida de los dueños de mascotas al permitir la supervisión y el control remoto de la alimentación, lo cual es especialmente útil para personas con horarios irregulares o que deben ausentarse con frecuencia.

Innovación Tecnológica: El proyecto explora el uso de tecnologías emergentes como la visión artificial y la integración de sistemas basados en Raspberry Pi y Arduino, lo que representa una innovación significativa en el ámbito de los dispositivos para mascotas.

Contribución al Campo de la Robótica y la Domótica: Los conocimientos adquiridos y las soluciones desarrolladas pueden aplicarse a otros proyectos de robótica y domótica, extendiendo el impacto más allá de la alimentación de mascotas.

Accesibilidad y Coste: Al utilizar componentes relativamente económicos y de fácil acceso como la Raspberry Pi y Arduino, el proyecto ofrece una

solución potencialmente más asequible que los sistemas comerciales existentes, ampliando su accesibilidad a un público más amplio.

3 Estado del arte

La alimentación automática de mascotas ha sido un tema de interés en la comunidad tecnológica y científica, y se han propuesto diversas soluciones que van desde mecanismos simples hasta sistemas avanzados que incorporan IoT (Internet de las Cosas) y tecnologías de inteligencia artificial.

Alimentadores Automáticos Comerciales:

PetSafe y SureFeed son ejemplos de alimentadores comerciales que utilizan temporizadores o mecanismos de identificación por microchip. Aunque eficaces, su funcionalidad se limita a horarios preestablecidos o identificación de mascotas por cercanía.

Proyectos Basados en IoT:

Furbo y Petnet han desarrollado dispositivos que permiten la alimentación remota a través de aplicaciones móviles. Estos sistemas ofrecen características como la programación remota y notificaciones, pero dependen en gran medida de la conectividad a Internet. Visión Artificial en Dispositivos de Mascotas:

Investigaciones como las de Wang et al. (2021) han explorado la aplicación de visión artificial para la identificación y comportamiento de animales domésticos, aunque estos estudios se centran más en la vigilancia y seguridad que en la alimentación. Integración de Raspberry Pi y Arduino:

Proyectos de código abierto y estudios académicos han demostrado la viabilidad de integrar Raspberry Pi con Arduino para diversas aplicaciones de automatización. Por ejemplo, la integración de sistemas para la automatización del hogar y el control de dispositivos electrodomésticos proporciona un marco sobre el cual se puede construir para este proyecto. El presente proyecto se diferencia de los anteriores al combinar la detección basada en visión artificial con un mecanismo de alimentación accionado por un servomotor, gestionado a través de la comunicación entre Raspberry Pi y Arduino, proporcionando una solución más adaptable y flexible.

4 Metodología

Recolección de Requisitos:

Sistema de Detección: Implementar y entrenar un modelo de visión artificial utilizando bibliotecas como OpenCV y TensorFlow en la Raspberry Pi para detectar la presencia del gato. Control del Servomotor: Programar el Arduino para controlar el servomotor encargado de la dispensación de la comida, configurando la conexión serial para recibir comandos desde la Raspberry Pi. Desarrollo de la Aplicación Web:

Crear una interfaz web que permita el accionamiento manual del alimentador y la visualización de imágenes capturadas en tiempo real. Utilizar frameworks como Flask o Django para el backend y React o Vue.js para el frontend. Integración de Sistemas:

Establecer la comunicación serial entre la Raspberry Pi y el Arduino. Integrar el sistema de detección con el control del servomotor, de manera que las señales de detección automaticen la dispensación de alimento.

Pruebas y Validación:

Realizar pruebas de detección con gatos reales para ajustar el umbral de confianza del sistema de visión artificial. Validar la funcionalidad del modo automático y manual en diversos escenarios, incluyendo interrupciones de señal y variaciones en el entorno de detección.

Documentación y Evaluación:

Documentar el diseño, desarrollo y pruebas del sistema. Evaluar el desempeño del sistema en términos de precisión de detección y fiabilidad en la dispensación de alimento, ajustando parámetros según sea necesario. Esta metodología asegura un enfoque estructurado para el desarrollo del alimentador automático, abarcando desde la conceptualización hasta la validación y documentación, garantizando la implementación exitosa del proyecto.

5 Descripción del Funcionamiento del Sistema

El sistema de alimentador automático desarrollado en este proyecto se basa en la detección de gatos a través de una aplicación de visión artificial, utilizando una Raspberry Pi Zero 2W y una cámara web. El alimentador puede activarse de dos formas distintas:

- **Detección del Gato:**:El sistema utiliza una cámara web conectada a la Raspberry Pi para capturar imágenes del entorno cada segundo. Estas imágenes se procesan para detectar la presencia de un gato con una confianza mayor al 50
- **Verificación del Intervalo de Alimentación:**:Además de la detección del gato, el sistema verifica que esté activo el token correspondiente al intervalo o horario de alimentación establecido.
- **Accionamiento del Alimentador:**:Si ambas condiciones anteriores se cumplen, el sistema envía una señal a través de una conexión serial desde la computadora a un Arduino. Este Arduino controla un servomotor que gira 45° durante 2 segundos, permitiendo la dispensación de la comida, y luego regresa a su posición original.
- **Modo Manual:**:
- **Accionamiento Manual:**:El alimentador puede ser accionado manualmente presionando un botón en una aplicación web. Al presionar este botón, se envía una señal similar a la del modo automático para activar el servomotor y dispensar la comida.
- **Detalles Técnicos:** Conexión Serial: La señal de activación del alimentador se envía desde la computadora al Arduino mediante una conexión serial. El mensaje específico activa el mecanismo que controla el servomotor. Raspberry Pi y Visión Artificial: Una Raspberry Pi Zero 2W, conectada a una cámara web, captura imágenes del entorno cada segundo. Estas imágenes se transmiten a la computadora mediante el protocolo SSH, donde son procesadas para la detección del gato. Las imágenes más recientes se muestran en la aplicación web y se actualizan continuamente. Al cerrar el programa, todas las imágenes capturadas se eliminan para conservar el espacio de almacenamiento. Y también durante el mismo procesamiento, existe la variable de retención, que específica cuantas imágenes capturadas el dispositivo servidor va a mantener.
- **Conexión Serial**:La señal de activación del alimentador se envía desde la computadora al Arduino mediante una conexión serial. El mensaje específico activa el mecanismo que controla el servomotor.

- **Raspberry Pi y Visión Artificial**:Una Raspberry Pi Zero 2W, conectada a una cámara web, captura imágenes del entorno cada segundo. Estas imágenes se transmiten a la computadora mediante el protocolo SSH, donde son procesadas para la detección del gato. Las imágenes más recientes se muestran en la aplicación web y se actualizan continuamente. Al cerrar el programa, todas las imágenes capturadas se eliminan para conservar el espacio de almacenamiento.

Este sistema combina tecnologías de visión artificial, comunicación serial y control de servomotores para proporcionar una solución efectiva y moderna a la alimentación automática de mascotas.

6 Algoritmos

Uno de los algoritmos principales que ocupamos en el desarrollo del proyecto, es el protocolo de conexión y comunicación de ssh [1]. En nuestro caso, ocupamos una implementación nativa en python del protocolo con la libraría paramiko.

Posteriormente, para la detección de la imágen, utilizamos una red neuronal convoulcional pre entrenada para la detección de gatos, que se observa su implementación en la siguiente figura 1.

```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
input_tensor = tf.convert_to_tensor(frame_rgb)
input_tensor = input_tensor[tf.newaxis, ...]

detections = model(input_tensor)
detection_scores = detections['detection_scores'][0].numpy()
detection_classes = detections['detection_classes'][0].numpy().astype(np.int32)
detection_boxes = detections['detection_boxes'][0].numpy()
```

Figure 1: Código para la carga de la imágen al modelo

Posteriormente, después de ser cargado, realizamos la comparación de los datos obtenidos, y el diccionario de objetos [2]. Para finalmente devolver si se considera presente el objeto de gato o no. 2

```
frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
input_tensor = tf.convert_to_tensor(frame_rgb)
input_tensor = input_tensor[tf.newaxis, ...]

detections = model(input_tensor)
detection_scores = detections['detection_scores'][0].numpy()
detection_classes = detections['detection_classes'][0].numpy().astype(np.int32)
detection_boxes = detections['detection_boxes'][0].numpy()
```

Figure 2: Clasificación de la imágen para observar si contiene un gato

7 Implementación

7.1 Det.py

```python
from flask import Flask, send_from_directory, render_template_string, jsonify, request
import os
from glob import glob
from datetime import datetime, timedelta
import time as t
import json
import threading
import serial
import jwt as pyjwt  # Import with alias
import cv2
import numpy as np
import tensorflow as tf
```

• **Flask:** Framework web utilizado para manejar las rutas y la lógica del servidor.

- **os:** Proporciona funciones para interactuar con el sistema operativo.
- **glob:** Permite buscar archivos en el sistema utilizando patrones.
- **datetime, timedelta:** Para manejar y operar con fechas y tiempos.
- **time as t:** Se usa para retrasos y esperas.
- **json:** Para trabajar con datos en formato JSON.
- **threading:** Para ejecutar funciones en segundo plano.
- **serial:** Para comunicarse con dispositivos a través de puertos seriales (e.g., Arduino).
- **jwt (pyjwt):** Para manejar tokens de autenticación JWT.
- **cv2 (OpenCV):** Para procesar y analizar imágenes.
- **numpy:** Para operaciones numéricas.
- **tensorflow:** Para cargar y utilizar el modelo de detección de objetos.

```python
app = Flask(__name__)

# RUTA DEL DIRECTORIO
IMAGE_FOLDER = r"/home/juan/detector/img"
MODEL_PATH = "/home/juan/models/ssd_mobilenet_v2_fpnlite_320x320/ssd_mobilenet_v2_fpnlite_320x320_coco17_tpu-8/saved_model"
(constant) SECRET_KEY: Literal['your_secret_key']
SECRET_KEY = 'your_secret_key'

# Intervalos de alimentación
FEEDING_INTERVALS = []  # Se agregarán dinámicamente
```

- **app:** Instancia de Flask.
- **IMAGE FOLDER:** Directorio donde se almacenan las imágenes.
- **MODEL PATH:** Ruta al modelo de detección de objetos guardado.
- **SECRET KEY:** Clave secreta para firmar los tokens JWT.

• **FEEDING INTERVALS:** Lista para almacenar intervalos de alimentación.

```
# Cargar modelo de detección
model = None
def load_model(model_path):
    print("Cargando el modelo...")
    global model
    model = tf.saved_model.load(model_path)
    print("Modelo cargado.")
```

load model: Función que carga el modelo de detección de objetos desde la ruta especificada.

```
def get_latest_image():
    """Devuelve la imagen más reciente en la carpeta."""
    images = glob(os.path.join(IMAGE_FOLDER, '*'))
    if not images:
        return None
    latest_image = max(images, key=os.path.getctime)
    return latest_image
```

get latest image: Devuelve la imagen más reciente en el directorio de imágenes.

```python
def detect_cat_in_image(image_path):
    """Detecta si hay un gato en la imagen especificada."""
    frame = cv2.imread(image_path)
    if frame is None:
        return False

    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    input_tensor = tf.convert_to_tensor(frame_rgb)
    input_tensor = input_tensor[tf.newaxis, ...]

    detections = model(input_tensor)
    detection_scores = detections['detection_scores'][0].numpy()
    detection_classes = detections['detection_classes'][0].numpy().astype(np.int32)
    detection_boxes = detections['detection_boxes'][0].numpy()

    max_score = 0
    for i in range(detection_scores.shape[0]):
        if (detection_scores[i] > 0.5 and detection_classes[i] == 17):
            if detection_scores[i] > max_score:
                max_score = detection_scores[i]

    return max_score > 0.5
```

detect cat in image: Detecta la presencia de un gato en la imagen utilizando el modelo cargado. Retorna True si se encuentra un gato, basado en la clase 17 (gato) de COCO dataset.

```python
@app.route('/latest_image_path')
def latest_image_path():
    """Devuelve la ruta de la imagen más reciente."""
    latest_image = get_latest_image()
    if latest_image:
        return jsonify({'image_path': os.path.basename(latest_image)})
    return jsonify({'image_path': None}), 404
```

/image: Sirve la imagen más reciente almacenada en el directorio.

```python
@app.route('/latest_image_path')
def latest_image_path():
    """Devuelve la ruta de la imagen más reciente."""
    latest_image = get_latest_image()
    if latest_image:
        return jsonify({'image_path': os.path.basename(latest_image)})
    return jsonify({'image_path': None}), 404
```

/latest image path: Devuelve la ruta de la imagen más reciente como JSON.

```python
@app.route('/')
def index():
    """Renderiza la página HTML principal."""
    token = generate_token()
    return render_template_string('''
    <!DOCTYPE html>
    <html lang="es">
    <head>
```

/: Renderiza la página HTML principal con un token generado.

```python
@app.route('/shutdown', methods=['POST'])
def shutdown():
    """Endpoint para cerrar el programa."""
    os._exit(0)
```

/shutdown: Cierra la aplicación.

```python
def activate_feeder():
    """Lógica para activar el alimentador."""
    latest_image = get_latest_image()
    if latest_image and detect_cat_in_image(latest_image):
        try:
            arduino = serial.Serial('/dev/ttyACM0', 9600, timeout=1)
            t.sleep(2)  # Espera a que la conexión serial se establezca
            arduino.write(b'M')  # Comando para alimentar
            arduino.close()
            return True
        except serial.SerialException as e:
            print(f"No se pudo abrir el puerto serial: {e}")
            return False
    else:
        print("No se detectó un gato en la imagen más reciente.")
        return False
```

activate feeder: Activa el alimentador si se detecta un gato en la imagen más reciente.

```python
@app.route('/feed', methods=['POST'])
def feed():
    """Endpoint para alimentar manualmente al gato."""
    success = activate_feeder()
    if success:
        return '', 200
    else:
        return '', 500
```

/feed: Endpoint para activar manualmente el alimentador.

```python
@app.route('/feeding_intervals', methods=['POST'])
def update_feeding_intervals():
    """Endpoint para actualizar los intervalos de alimentación."""
    global FEEDING_INTERVALS
    data = request.get_json()
    FEEDING_INTERVALS = data.get('feeding_intervals', FEEDING_INTERVALS)
    return '', 200
```

/feeding intervals: Actualiza los intervalos de alimentación recibidos en el cuerpo de la solicitud.

```python
def generate_token():
    """Genera un token JWT."""
    payload = {
        'exp': datetime.utcnow() + timedelta(minutes=30),
        'iat': datetime.utcnow(),
    }
    return pyjwt.encode(payload, SECRET_KEY, algorithm='HS256')
```

/token: Devuelve un token JWT.

```python
def generate_token():
    """Genera un token JWT."""
    payload = {
        'exp': datetime.utcnow() + timedelta(minutes=30),
        'iat': datetime.utcnow(),
    }
    return pyjwt.encode(payload, SECRET_KEY, algorithm='HS256')
```

generate token: Genera un token JWT válido por 30 minutos.

```python
def validate_token(token):
    """Valida un token JWT."""
    try:
        pyjwt.decode(token, SECRET_KEY, algorithms=['HS256'])
        return True
    except pyjwt.ExpiredSignatureError:
        return False
    except pyjwt.InvalidTokenError:
        return False
```

validate token: Valida un token JWT.

```python
def schedule_feeding():
    """Programa la alimentación automática."""
    while True:
        now = datetime.now()
        for interval in FEEDING_INTERVALS:
            start, end = interval.split(' a ')
            start_time = datetime.strptime(start, '%H:%M').time()
            end_time = datetime.strptime(end, '%H:%M').time()
            if start_time <= now.time() <= end_time and detect_cat_in_image(get_latest_image()):
                activate_feeder()
                t.sleep(60)  # Evita que se active varias veces
        t.sleep(1)
```

schedule feeding: Programa la alimentación basada en los intervalos definidos, activando el alimentador si se detecta un gato.

```python
if __name__ == '__main__':
    load_model(MODEL_PATH)
    threading.Thread(target=schedule_feeding, daemon=True).start()
    app.run(debug=True)
```

main: Carga el modelo, inicia un hilo para la alimentación automática, y corre la aplicación Flask en modo de depuración.

Flujo del Código

Configuración Inicial: Se definen rutas y se inicializa Flask.

Carga de Modelo: Se carga un modelo de detección de objetos de TensorFlow.

Detección y Manejo de Imágenes: Se define la lógica para obtener y procesar la imagen más reciente.

Rutas de API: Varias rutas de Flask manejan solicitudes relacionadas con imágenes, intervalos de alimentación, y activación del alimentador.

Generación de Tokens: Se manejan tokens JWT para la autenticación.

Alimentación Automática: Se define un hilo para manejar la alimentación automática basada en los intervalos definidos.

Ejecución: La aplicación Flask se inicia, cargando el modelo y comenzando el monitoreo de intervalos de alimentación.

Cada función está cuidadosamente definida para manejar un aspecto específico del proyecto, desde la detección de gatos en imágenes hasta la comunicación con el hardware del alimentador y la interfaz web para gestionar la configuración y monitoreo.

8 Resultados

8.1 Análisis de resultados

En cuánto a resultados preliminares, se observa que efectivamente el modelo funcionó con el gato de prueba en la mayoría de los casos. Existían instancias donde por el movimiento del sujeto la camara tomaba una foto borrosa, que causaba errores a la hora de pasar la foto por el modelo.

En cuanto a valores esperados del sistema. Cuando se encontraba dentro del intervalo de tiempo, y estaba la presciencia del gato, se lograba proporcionar la comida de manera adecuada.

8.2 Contraste con los objetivos

En términos de presentación física, no habíamos realizado ningún objetivo, pero de igual manera, a contnuación se encuentra una imágen del dispositvo. El producto final se puede observar en la figura 3.

Figure 3: Figura que muestra el alimentador de alimentos

A continuación encontramos una figura de la interfáz gráfica 3

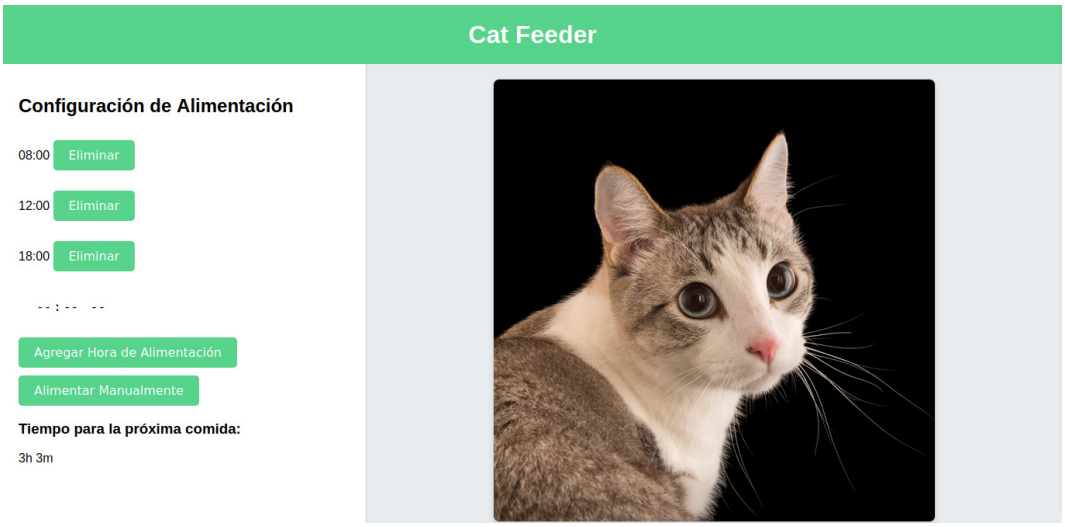

Figure 4: Una figura de la interfáz gráfica

Después en la figura 5 encontramos la camera, y en la figura 6 encontramos el raspberry pi que tomo el rol principal de tomar y mandar las fotos a la aplicación, para que este dispositivo no tenga que estar cercano a la computadora.

Figure 5: Foto de la camera

Figure 6: Foto de la raspberry 2w

En cuanto a otros objetivos. se considera que todos los objetivos tanto generales como específicos se cumplieron con el desarrollo del proyecto.

8.3 Liga del video de demostración

Presione aqui

9 Conclusión

El desarrollo del alimentador automático basado en visión artificial para gatos, utilizando Raspberry Pi y Arduino, ha demostrado ser una solución eficaz para mejorar la gestión de la alimentación de mascotas. A través de la implementación de un sistema que combina la detección visual con un control preciso de la dispensación de alimentos, se ha logrado:

- **Automatización Eficiente de la Alimentación**: El sistema puede identificar de manera precisa la presencia del gato utilizando visión artificial, lo que permite la dispensación de comida únicamente cuando es necesario, minimizando el desperdicio de alimentos y asegurando que el gato sea alimentado adecuadamente.

- **Versatilidad de Operación**: La capacidad de operar en modo automático y manual ofrece flexibilidad a los dueños de mascotas, permitiéndoles confiar en el sistema para la alimentación regular o intervenir manualmente cuando sea necesario.

- **Integración Sencilla de Tecnología**: La combinación de Raspberry Pi y Arduino ha proporcionado una plataforma robusta para la implementación de funciones avanzadas de visión artificial y control de hardware. La comunicación serial ha facilitado la coordinación entre los diferentes componentes del sistema, demostrando su eficacia en aplicaciones de automatización doméstica.

- **Facilidad de Uso y Monitoreo**: La interfaz web permite a los usuarios monitorear y controlar el sistema de alimentación de manera intuitiva, brindando una experiencia de usuario amigable y accesible.

En conclusión, este proyecto ha validado la viabilidad de utilizar tecnologías modernas para abordar desafíos cotidianos en el cuidado de mascotas. El alimentador automático desarrollado no solo mejora la conveniencia para los dueños, sino que también garantiza un cuidado más consistente y eficiente para los gatos, destacándose como una contribución significativa en el ámbito de la automatización y la robótica aplicada a la vida diaria.

Este trabajo sienta las bases para futuras mejoras, como la incorporación de algoritmos más avanzados de detección y la ampliación de las capacidades de la aplicación web, lo que podría llevar a una mayor personalización y funcionalidad en sistemas de alimentación automática para mascotas.

10 Referencias bibliográficas

Furbo. (s.f.). Furbo Dog Camera: Treat Tossing, Full HD Wifi Pet Camera and 2-Way Audio, Designed for Dogs. Recuperado el 26 de junio de 2024, de `https://shopus.furbo.com/`

PetSafe. (s.f.). PetSafe® Official Website. Recuperado el 26 de junio de 2024, de `https://www.petsafe.net/`

Petnet. (s.f.). Smart Pet Products for Smart Pet Parents. Recuperado el 26 de junio de 2024, de `https://www.petnet.io/`

SureFeed. (s.f.). SureFeed Microchip Pet Feeder. Recuperado el 26 de junio de 2024, de `https://www.surepetcare.com/en-us/pet-feeder/microchip-pet-feeder`

Wang, Z., Yu, X., Zhang, W., & Wang, X. (2021). Application of Artificial Intelligence in Intelligent Pet Monitoring and Feeding System. *Procedia Computer Science, 181*, 898-903. `https://doi.org/10.1016/j.procs.2021.01.244`

OpenCV. (s.f.). Open Source Computer Vision Library. Recuperado el 26 de junio de 2024, de `https://opencv.org/`

Raspberry Pi Foundation. (s.f.). Raspberry Pi Documentation. Recuperado el 26 de junio de 2024, de `https://www.raspberrypi.com/documentation/`

Arduino. (s.f.). Arduino Documentation. Recuperado el 26 de junio de 2024, de `https://docs.arduino.cc/`

Flask. (s.f.). Flask Documentation. Recuperado el 26 de junio de 2024, de `https://flask.palletsprojects.com/`

TensorFlow. (s.f.). TensorFlow Documentation. Recuperado el 26 de junio de 2024, de `https://www.tensorflow.org/`

11 Anexo

Tanto la guía de instalación como el manual de usuario se encuentran en el repositorio de git, pero también se integrará como parte de este documento.

Repositorio git

11.1 Guía de instalación

Raspberry Pi zero 2w

- En tu raspbery pi zero 2w, configuralo al internet con acceso ssh.
- Modifica el tamaño del tamaño swap para que sea posible actualizar el sistema

```
sudo nano /etc/dphys-swapfile
sudo dphys-swapfile setup
sudo dphys-swapfile swapon
```

- Actualiza el sistema

```
sudo apt update
sudo apt upgrade
```

- Instala fswebcam

```
sudo apt install fswebcam
```

Servidor o máquina principal

- Clona el repositorio

```
git clone https://github.com/Juan-Alvarado21/Automatic_Cat_Feeder_
cd Automatic_Cat_Feeder_AI
```

- Instala los requerimientos de python

```
pip install -r requirements.txt
```

- Verifica que ssh este habilitado tanto en el raspberry como en tu máquina principal

- Obtener el ip de tu raspberry, esto se puede realizar tanto con páginaas web, o en el caso de que no tengas un monitor, podrás revisar los logs del network manager.

- Modifica los valores del host, usuario y contraseña y descomentalos en "src/main.py", a los datos que colocó para su raspberry pi 2w

```
#Establecer usuario, host y password
#usuario=""
#contrasenia=""
#host="localhost"
```

- Descarga y coloca el modelo "mobile_netv2.caffemodel" en el src. (Es muy pesado y no se pudo inclulir en el repositorio git

- Corre main.py y det.py en dos ventanas distintas de manera simultanea.

11.2 Guía de usuario

- Si se quiere alimentar el gato manualmente, utiliza el botón de "Alimentar Manualmente"

- Si se quiere alimentar automáticamante, agregue los horarios necesarios, y mientras corre el sistema, el gato será alimentado cuando se logra observar, y este en el horario adecuado.