Universidad Zaragoza
1542

Trabajo Fin de Grado

Prototipo de control de fauna de pequeño tamaño basado en Raspberry Pi y el uso de identificación RFID

Small wildlife control prototype based on Raspberry Pi and the use of RFID identification

Autor/es

Pablo Miguel Cuadrado García

Director/es

Ana María López Torres

Facultad / Escuela

Escuela Universitaria Politécnica de Teruel

Año

2022

Autor: Pablo Miguel Cuadrado García

Resumen:

En el presente trabajo de fin de grado se desarrolla una aplicación para la identificación de aves mediante el uso de un ordenador monoplaca, potente, compacto y de bajo coste, denominado Raspberry Pi 3 B+. A partir de este dispositivo, se emplea un lector de identificación por radiofrecuencia como método para detectar e identificar el pájaro que llega a la estación, al que previamente se le ha dotado de una etiqueta que reconoce dicho dispositivo. Una vez el sensor ha leído dicha etiqueta, se toma una fotografía del pájaro, formando de esta manera un perfil individual que diferencia y caracteriza al pájaro de forma inequívoca. La información sobre el pájaro identificado se guarda en una base de datos para su seguimiento posterior.

Agradecimientos:

Me gustaría agradecer a mi directora del trabajo de fin de grado Ana María López Torres por toda su ayuda, comprensión y paciencia durante el desarrollo.

Agradecer a mi padre por sus consejos de carpintería para la construcción de la estación.

Agradecer a mi hermana y a mi madre por su cariño.

Finalmente, agradecer a todo el personal de la EUPT por todos estos años de aprendizaje.

Índice.

Figuras.

Tablas.

1. Introducción

La importancia de las aves para el ecosistema es crucial. Desde funciones como la eliminación de carroña o de insectos considerados como plagas, hasta la polinización de plantas o la expulsión de semillas en las deposiciones para el crecimiento de nuevas plantas, estos vertebrados son capaces de abarcar un gran espectro de actividades beneficiosas para todo el planeta Tierra, y por lo tanto, para el ser humano.

Hoy en día en España, existen un total de 21 especies de aves en peligro de extinción. Desde contactos eléctricos directos con las líneas de alta tensión, hasta depredadores o el propio crecimiento de las ciudades, son causas de la reducción de individuos de estas especies. Es de una dificultad notoria, la aplicación de medidas para su protección sobre muchos de estos factores. Sin embargo, y gracias a la aplicación que va a desarrollarse en el presente documento, será posible mantener localizadas a estas aves, permitiendo de esta forma, rastrear alguna de las fuentes de peligro para ellas, y en el caso de que sea posible, actuar sobre ella y eliminarla.

Una de las principales ventajas que nos brinda la electrónica hoy en día, es la portabilidad y el pequeño tamaño de sus componentes. Hace 20 años, no existía la posibilidad de acceder a un ordenador potente, de bajo coste y sobre todo que tuviese un reducido tamaño. Sin embargo, en el año 2022, la disponibilidad de dispositivos monoplaca como Raspberry Pi 3 B+, nos va a dar la herramienta necesaria para emplazar una estación de localización con identificación por radiofrecuencia en una cantidad elevada de lugares, ya sea en nuestros bosques o ciudades. Este hecho, unido a la cobertura global de internet que se promete, gracias a proyectos como Starlink [4], permite prácticamente la monitorización de datos y el seguimiento en tiempo real. Además, existe la posibilidad de disponer de una alimentación autónoma que permite el funcionamiento en los lugares remotos en los que se instalará la estación, consistente en una pequeña placa solar y una batería.

Sin embargo, el uso de la tecnología RFID como método de rastreo e identificación en el campo de la ornitología no es algo nuevo. David N.Bonter et al [5] enumeran algunas aplicaciones en el campo de la ornitología "*Entre los primeros y quizás los más productivos usos de la tecnología RFID para la investigación en ornitología se encuentra un proyecto concebido por Peter Becker y sus colegas trabajando con Charrán Común (Sterna hirundo) cerca de Wilhelmshaven, Alemania.*" (Traducción propia).

2. Marco teórico

2.1. ¿Qué es la tecnología RFID?

La tecnología RFID o identificación por radiofrecuencia (Radio Frequency Identification) es un sistema que permite almacenar y leer datos de unas etiquetas o "tag". Estas etiquetas, poseen un código único, que permite identificarlas de forma inequívoca. Dichas etiquetas tienen integrada una antena propia que les permite comunicarse mediante campos magnéticos o electromagnéticos.

2.2. ¿Cómo funciona?

El proceso para la lectura de etiquetas pasivas, que al fin y al cabo son las utilizadas en esta aplicación, se esquematiza en la figura 1:

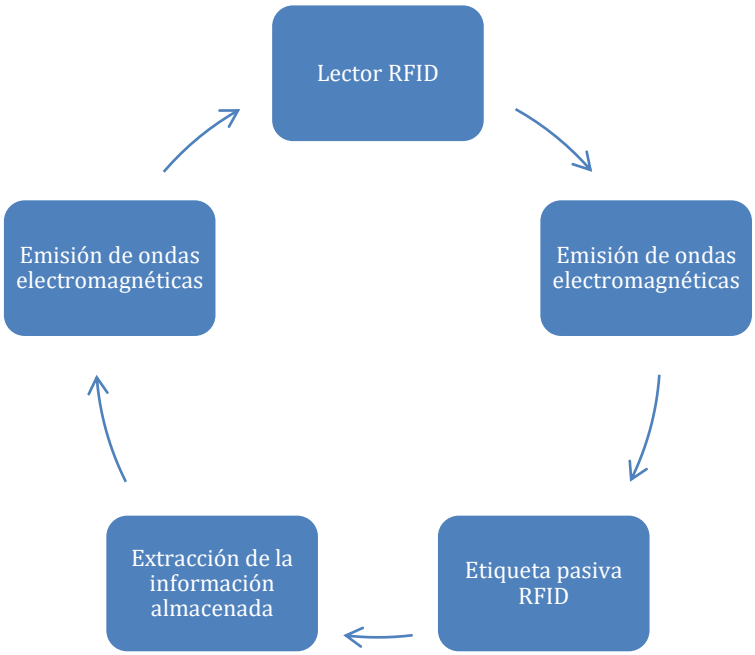

Figura 1. Proceso de lectura de una etiqueta RFID pasiva.

En primer lugar, y para poner en contexto, se deben conocer los componentes que forman un sistema RFID:

1. **Etiqueta RFID:** Está compuesta por una antena y un microchip electrónico. El microchip es el encargado de almacenar los datos (con una capacidad de hasta varios millares de bytes). La antena es la encargada de emitir o recibir las ondas electromagnéticas que transportan los datos. Existen tanto etiquetas de solo lectura como etiquetas de lectura y escritura. Dentro de las etiquetas, hay que realizar una distinción importante entre los siguientes tipos:

- **Etiquetas pasivas**: No poseen fuente de alimentación propia, por lo tanto la única forma de comunicarse la obtienen de la energía de las ondas proveniente del lector.
- **Etiquetas activas**: Poseen su propia alimentación. De esta manera pueden enviar señales de manera autónoma e independiente del lector.

2. **Lector RFID**: Se compone de una antena, un transceptor y un módulo de control. La antena es la encargada de enviar y recibir ondas. El transceptor permite la transmisión y recepción. Por último, el módulo de control permite controlar el tráfico de datos y el reloj del sistema.
3. **Sistema de procesamiento de datos**: Es un microchip que permite procesar y en este caso enviar los datos a través de los pines hasta la Raspberry Pi o cualquier otro sistema.

Para frecuencias de operación entre 100 kHz y 30 MHz se utiliza un sistema para la comunicación llamado acoplamiento inductivo.
Este es el caso del sensor RC522 (lector RFID utilizado para implementar la aplicación), con una frecuencia de operación de 13.56 MHz. El fenómeno del acoplamiento inductivo entre el lector y la etiqueta pasiva se explicaría de la siguiente forma:

La antena del lector RFID genera un campo electromagnético fuerte y de alta frecuencia. Una pequeña parte de este campo llega hasta la etiqueta, penetrando en la bobina que forma la antena. Por inductancia, se genera un voltaje en la bobina que funciona como antena en la etiqueta. Tras un proceso de rectificación mediante una etapa previa, este voltaje ya estabilizado, funciona como alimentación para el microchip de la etiqueta que contiene los datos de la misma. Gracias a un condensador colocado en paralelo con la antena del lector (la parte inductiva) se consigue un circuito resonante, con una frecuencia de resonancia que es la frecuencia de transmisión de dicho lector. En la etiqueta existe el mismo circuito resonante formado por la inductancia de la antena y otro condensador en paralelo que permite sintonizar con la misma frecuencia del lector.

Es decir, para una operación básica de lectura el lector RFID generaría ondas electromagnéticas de forma periódica. En el momento en el que una etiqueta entra en el rango de estas ondas, se produciría el acoplamiento inductivo, permitiendo de esta forma la transmisión de datos de la etiqueta al lector o viceversa.

2.3. Frecuencias de operación

Existen tres rangos de frecuencias de operación para los lectores comerciales:

- Baja frecuencia (Low frequency): Frecuencias de operación menores de 400 kHz. Las más comunes se encuentran entre 125 kHz y 134,2 kHz.

- Alta frecuencia (High frequency): Frecuencias de operación entre 3 y 30 MHz. Los más comunes son el de 13,56 MHz, como es el caso del RC522.

- Ultra alta frecuencia (Ultra high frequency): Frecuencias de operación entre 300 MHz y 5,8 GHz.

2.4. Ventajas y desventajas

En el siguiente punto se van a desarrollar las ventajas y desventajas que entraña la elección de una frecuencia de operación.

Ventajas para baja frecuencia.

- Funcionan de forma fiable en condiciones ambientales desfavorables (humedad alta, colisiones con otros objetos, demasiado polvo en el aire etc).
- No se necesita tener línea de visión entre la etiqueta y el lector. Es decir, puede leerse a través de cualquier sólido no metálico (obviamente dentro del rango de distancia).
- La lectura se puede realizar de forma omnidireccional, permitiendo una orientación relativa flexible de la etiqueta o el lector.
- Ideales para trabajar en entornos con interferencias electromagnéticas altas.

Desventajas para baja frecuencia.

- Rango de lectura relativamente bajo.
- Velocidad de transmisión de datos lenta.

Ventajas para alta frecuencia.

- Se puede conseguir un mayor rango de lectura en zonas con poca o ninguna interferencia electromagnética (hasta unas decenas de centímetros).
- Velocidad de transmisión de datos rápida.

Desventajas para alta frecuencia.

- Gran sensibilidad a cualquier interferencia electromagnética.
- La señal está mucho más enfocada, por lo tanto se pierde esa omnidireccionalidad, teniendo que orientar la etiqueta de forma correcta.
- No permite la comunicación a través de líquidos.

Ventajas para ultra alta frecuencia.

- Rango de lectura aún más elevado en zonas con poca o ninguna interferencia electromagnética (del orden de metros).
- Velocidad de transmisión de datos aún más elevada.

Desventajas para ultra alta frecuencia.

- Mayor sensibilidad a interferencias electromagnéticas.
- Mayor enfoque de la señal, teniendo que orientar la etiqueta de forma correcta.
- No permite la comunicación a través de líquidos.
- Necesidad de un sistema anticolisión.

2.5. Aplicaciones de la tecnología RFID

Los usos de la tecnología RFID son muy variados y están aplicados en una gran cantidad de industrias y servicios. Algunas aplicaciones son las siguientes:

- Seguimiento de mascotas y ganado: Por ejemplo, los chips que colocados a los perros en el veterinario para su identificación son etiquetas RFID.
- Gestión del inventario: Un tag RFID en forma de pegatina colocada en un objeto del inventario permite identificarlo, contarlo y localizarlo.
- Control de acceso: Se equipa a tarjetas de identificación con etiquetas RFID, pudiendo de esta forma, mediante un lector identificar y permitir el acceso de personas.
- Envíos: Permite tomar la lectura de una etiqueta RFID en un objeto recibido, teniendo así constancia de su llegada al destino.
- Cuidado de la salud: Facilitando el triaje y la identificación del historial del paciente.
- Seguridad anti robo: Dotando a los objetos de las tiendas con etiquetas RFID y lectores RFID a la salida.
- Pagos con tarjeta de crédito “tap-and-go”: Se puede dotar a las tarjetas de crédito de un tag RFID y usar un lector RFID para realizar los pagos.

3. Hardware

En el siguiente punto se realiza un breve bosquejo de los elementos hardware que componen el proyecto, tanto su coste como sus especificaciones, incidiendo de manera más detallada en aquellos componentes considerados más relevantes.

3.1. Listado

Componente	Precio unitario	Cantidad	Especificaciones	Coste total
Computador monoplaca	50 €	1	Raspberry Pi 3 B+	50 €
Módulo lector RFID	10 €	1	RC522 de 13.56 MHz	10 €
Cámara	28.90 €	1	Raspberry Pi Pi camera module V2 8MP	28.90 €
Cables	0.04 €	7	Female to female cables	0.28 €
Monitor	65 €	1	Conexión USB	65 €
Teclado	15 €	1	Conexión USB	15 €
Ratón	10 €	1	Conexión USB	10 €
Cables	3 €	1	HDMI	3 €
Cables	1 €	1	Ethernet	1 €
Tarjeta micro SD	10.95 €	1	32 GB de memoria	10.95 €
Estructura de madera	20 €	1	Tablones de aglomerado, tornillos y cola de madera.	20 €

Tabla 1.Listado de componentes.

En los siguientes dos apartados, se proveerá de algunos detalles de aquellos componentes que se consideran más importantes en el proyecto.

Nota: Todos los precios escritos en la Tabla 1 son válidos y actualizados a junio de 2022.

3.2. Raspberry Pi 3 B+. Especificaciones técnicas

Este computador monoplaca (figura 2) fue lanzado en marzo de 2018. La principal ventaja respecto de su antecesor fue la inclusión de un nuevo procesador más potente de 2.4 GHz. Alguna de sus características técnicas más reseñables son las siguientes:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- RAM: 1GB LPDDR2 SDRAM.
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE.
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps).
- GPIO de 40 pines.
- Entrada HDMI.
- 4 puertos USB 2.0.
- Puerto CSI para conectar una cámara.

- Puerto DSI para conectar una pantalla táctil.
- Salida de audio estéreo y vídeo compuesto.
- Entrada para tarjeta Micro-SD.
- Power-over-Ethernet (PoE).

Figura 2. Raspberry Pi 3 B+.

3.3. Módulo lector RFID RC 522. Especificaciones técnicas.

El módulo RC522 (figura 3) es un Lector-Grabador RFID de 13.56 Mhz. Realiza la comunicación mediante SPI (Serial Peripheral Interface), lo que lo hace ideal para trabajar con casi todos los microcontroladores del mercado. El rango de detección es de aproximadamente 5-7 cm. Algunas de sus características técnicas más reseñables son las siguientes:

- Voltaje de Operación: 3.3V DC.
- Corriente de Operación: 13-26mA/3.3V DC.
- Corriente de Standby: 10-13mA/3.3V DC.
- Corriente de Sleep: <80uA.
- Corriente pico: <30mA.
- Frecuencia de operación: 13.56 MHz.
- Transferencia de datos: Max. 10Mbit/s.
- Dimensiones RFID-RC522: 40 mm x 60 mm.
- Dimensiones Tarjeta: 85 mm x 54 mm.
- Temperatura de funcionamiento: -20 a 80 grados centígrados.
- Temperatura de almacenamiento: -40 a 85 grados centígrados.
- Humedad relativa: 5% hasta 95 %.
- La tasa de transmisión por defecto: 9600bps, velocidad de transferencia máxima: 1228800bps.

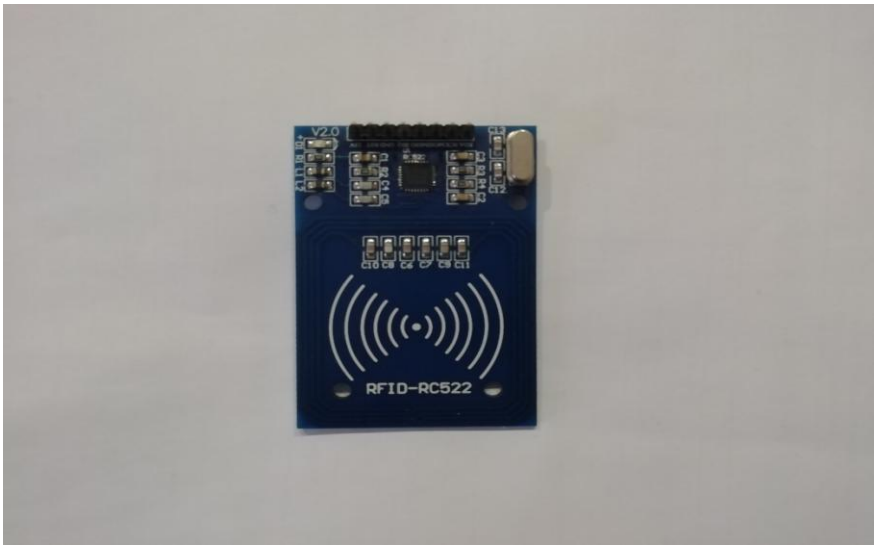

Figura 3.Lector RFID-RC522.

3.4. Conexión de los componentes.

- Conexión entre la placa y el ordenador portátil.

En el desarrollo de esta aplicación se ha utilizado un ordenador portátil como monitor para la Raspberry Pi 3 B+ (Figura 2), debido a que no se disponía de un monitor independiente. Esta conexión se realiza con cable Ethernet (Figura 4). Sin embargo, la forma más barata y práctica sería conectando un monitor aparte vía conexión HDMI.

Figura 4.Conexión placa-PC vía Ethernet.

- Conexión entre la placa y los periféricos.

En la figura 5 se muestra la conexión entre la fuente de alimentación, el ratón USB, el teclado USB y la placa.

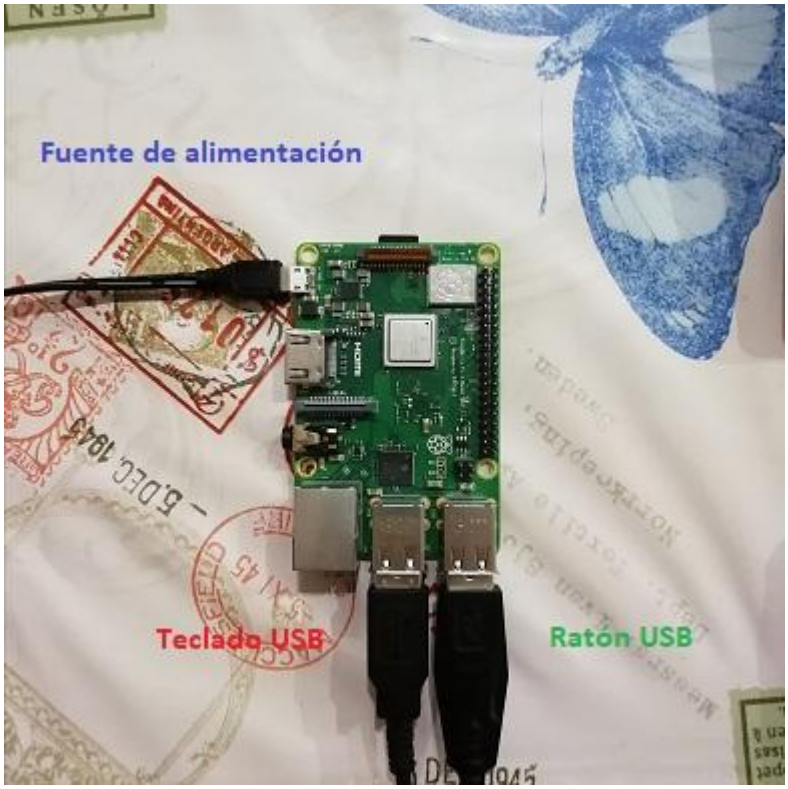

Figura 5.Conexión periféricos- placa.

- Conexión entre la placa y el lector RFID.

En la tabla 2 se muestran las conexiones entre pines de la placa y la antena lector RFID. En la figura 6 se muestra la conexión física.

Número de PIN de la placa	Conexión en la antena lector RFID
PIN 1	3.3 V
PIN 6	GND
PIN 19	MOSI
PIN 21	MISO
PIN 22	RST
PIN 23	SCK
PIN 24	SDA

Tabla 2.Conexiones placa-antena lector RFID.

Figura 6. Conexión placa-RFID.

- Conexión entre la placa y la cámara.

Dado que la cámara está fabricada por la misma empresa que realiza las placas Raspberry Pi la conexión será de lo más sencilla. Levantando cuidadosamente un zócalo situado en la placa, se introduce el conector de la cámara, como se muestra en la figura 7.

Figura 7. Conexión placa-cámara.

- Conexión de todo el sistema.

En la figura 8 se muestran conectados al mismo tiempo todos los elementos hardware que componen la aplicación.

Figura 8. Conexión completa.

- Montaje completo.

En la figura 9 se muestra la estación como tal, en la que va acoplado el sensor RC522, la cámara y la Raspberry Pi 3 B+. Se ha decidido realizar mediante la compra de tableros y un pequeño trabajo de carpintería un prototipo de caseta para acoplar el sensor, la cámara y la Raspberry Pi 3 B+ para darle cierto realismo.
El trabajo de carpintería consistió en:

1. Toma de medidas y corte de las piezas mediante sierra de calar.
2. Realización de los siguientes agujeros: Agujeros para ventilación, cable de alimentación, cable de Ethernet, conexión del lector RFID y para la cámara.
3. Montaje mediante cola plástica de madera de las diferentes partes.
4. Realización de medición, agujeros y avellanado para tornillos.
5. Acople de tornillos y clavos.
6. Eliminación de aristas vivas y sellado de juntas.

Figura 9. Montaje completo.

4. Software.

En este punto se realiza la definición y explicación de la aplicación a nivel informático.

4.1. Funcionamiento básico del programa.

La aplicación se divide en **dos partes**, cada una caracterizada por una función principal:

La **primera parte** es la identificación del animal mediante la lectura de la etiqueta RFID. Dentro de esta parte, el funcionamiento es el siguiente:

1. El lector RFID se encuentra leyendo, a la espera de que una etiqueta se posicione dentro de su rango de lectura.
2. Una vez una etiqueta ha entrado en el rango de lectura, se registran sus datos (identificador y nombre del ave), además del momento de llegada y se toma una foto del pájaro. Toda esta información, se guarda en una base de datos de forma ordenada y correcta.
3. Para evitar leer la misma etiqueta varias veces en ese espacio temporal, el lector esperará un tiempo variable fijado por el usuario evitando así una excesiva carga de información redundante en la base de datos. Este tiempo variable se fijará en función de la experiencia, es decir mediante la observación de estadísticas de tiempo que dicha especie de pájaro pasa delante del lector.
4. Tras un número variable de lecturas introducidas en la base de datos, fijado por el usuario, esta base de datos se enviará por correo electrónico y se borrará para evitar que la memoria de la Raspberry Pi se llene demasiado.

La **segunda parte** contiene la función orientada a la interacción con el usuario, es decir, la escritura de las etiquetas y la lectura de información de la base de datos mediante una interfaz gráfica. El funcionamiento no sigue una secuencia, ya que depende de los deseos del usuario. Permite realizar las siguientes acciones mediante botones:

- Escritura de la etiqueta.
- Contar el número de avistamientos de un tipo de pájaro que se han producido y visualizar su última fotografía.
- Salida del programa.

Se describirá esta segunda parte de forma más intensa en el apartado 6 de la memoria.

4.2. Programas utilizados.

Para el desarrollo y funcionamiento de la aplicación se utilizan los siguientes programas listados en esta tabla.

Tipo de programa	Programa específico
Sistema operativo	Raspbian GNU/Linux 11.0
Python IDE	Geany 1.37.1
Visualizador de bases de datos	DB Browser for SQLite 3.12.2
Terminal de comandos Linux	LX Terminal
Python IDE	Spyder 4.1.5

Tabla 3.Lista de programas utilizados.

4.3. Librerías y módulos utilizados.

En este apartado se hace mención a aquellas librerías que deben importarse para hacer funcionar la aplicación.

- Librería “RPi.GPIO”: Se trata de una librería esencial. Contiene todas las funciones que nos permiten utilizar los pines entrada/salida de propósito general (GPIO).
- Librería “tkinter”: Es la librería estándar de Python para la implementación de una interfaz gráfica.
- Librería “mfrc522”: Contiene las funciones que permiten la puesta en marcha, lectura y escritura con el lector RFID RC522.
- Módulo “time”: Permite usar herramientas relacionadas con la medición del tiempo.
- Módulo “datetime”: Permite usar herramientas relacionadas con la medición del tiempo, pero a diferencia del módulo “time” está más orientado a la obtención de fechas.
- Módulo “sqlite3”: Permite la creación y gestión de bases de datos.
- Librería “picamera”: Permite usar la cámara de la Raspberry Pi de forma sencilla e intuitiva.
- Librería “smtplib”: Contiene las funciones que permiten enviar correos electrónicos mediante el protocolo SMTP (protocolo de transferencia de correo simple).
- Librería “email”: Utilizado para la organización a la hora de mandar correos electrónicos.
- Librería “PIL”: Utilizada para el procesado de imágenes a la hora de mostrar por pantalla en la interfaz gráfica.

4.4. Archivos que componen la aplicación.

En este apartado se van a mostrar (Tabla 4) aquellos archivos de extensión python que en su conjunto forman la aplicación.

Nombre del archivo	Importancia	Función
Reading_RFID.py	Principal	Lectura de la etiqueta RFID, toma de fotografía y gestión de datos.
Writing_RFID.py	Principal	Escritura de la etiqueta RFID, extracción de datos y gestión de la interfaz gráfica.
MandarEmail.py	Secundaria	Enviar email al destinatario elegido con un mensaje y la base de datos adjunta.
Camera.py	Secundaria	Utilizar la cámara conectada a la Raspberry Pi para fotografiar el pájaro.

Tabla 4.Archivos que componen la aplicación.

4.5. Base de datos.

La base de datos denominada "Entrada_Aves.db" está constituida por la tabla "Llegadas" (Tabla 5) .Esta tabla contiene las siguientes columnas:

codigo	identificador	tipo	fecha	foto

Tabla 5. Columnas de la tabla Llegadas.

Más concretamente se definen de la siguiente forma:

- codigo: De tipo entero, correspondiente a la variable global que numera las filas.
- identificador: De tipo entero, es el código único que identifica al tag RFID.
- tipo: Columna en la que se introduce el texto grabado en el tag RFID, en este caso el nombre del ave.
- fecha: De tipo texto, contiene la información del momento exacto en el que el pájaro fue identificado por el lector RFID.
- foto: De tipo binario, contiene la información binaria de la imagen del pájaro fotografiada por la cámara.

5. Funcionamiento de la aplicación.

En este punto se explican los dos archivos principales con extensión Python que definen la aplicación. Debe mencionarse que las capturas de pantalla del código están realizadas en el Python IDE spyder únicamente por razones de visualización, sin embargo la totalidad del código ha sido desarrollado en el IDE Python Thonny.

Reading_RFID.py

En la figura 10, se cargan todos aquellos módulos o librerías que contienen las funciones explicadas en el apartado 4.2.

```
5   import RPi.GPIO as GPIO
6   from mfrc522 import SimpleMFRC522
7   import time
8   from datetime import datetime
9   import sqlite3
10  from MandarEmail import *
11  from Camera import Tomar_foto
```

Figura 10. Importado Reading_RFID.py

En la figura 11, se inicializan las variables globales. La variable código, es aquella que contabiliza las filas introducidas en la tabla "Llegadas" de la base de datos. La variable "time_between_arrivals" permite controlar cada cuanto tiempo leemos un posible tag RFID presente en el rango de lectura.

```
13  codigo=0
14  time_between_arrivals=20
```

Figura 11. Inicialización de variables globales.

En la figura 12, se define la función que permite pasar los archivos a binario, en este caso es un requisito para poder introducir las imágenes fotografiadas como parte de la base de datos. El argumento es un string con el nombre del archivo, incluyendo su extensión.

```
16  def convertToBinaryData(filename):
17
18      with open(filename, 'rb') as file:
19          blobData = file.read()
20      return blobData
```

Figura 12. Función de conversión de datos.

En la figura 13, se define la función que permite borrar las filas de la tabla "Llegadas" en la base de datos. Se utiliza para liberar espacio interno de la Raspberry Pi, por supuesto una vez se ha mandado por correo para evitar la pérdida de datos.

```
22      def Borrar_todas_filas():
23
24          sql = 'DELETE FROM Llegadas'
25          cur = conexion.cursor()
26          cur.execute(sql)
27          conexion.commit()
```

Figura 13. Borrado de filas.

En la figura 14, se define la función que permite la creación y/o apertura de la base de datos "Entrada_Aves.db" y la tabla "Llegadas" que contiene los datos introducidos. Sus elementos se encuentran definidos en el punto 4.5.

```
29      def Crear_Abrir_DB():
30
31          global conexion
32          conexion=sqlite3.connect("Entrada_Aves.db")
33          conexion.execute("""create table if not exists Llegadas (
34                                  codigo integer,
35                                  identificador integer,
36                                  tipo text,
37                                  fecha text,
38                                  foto blob
39
40                          )""")
```

Figura 14. Creación y/o apertura de la tabla en la base de datos.

En la figura 15, se define la función que permite introducir valores en cada columna de la correspondiente nueva fila. Estos valores son variables y ordenados en el orden en el que fueron creadas las columnas (Figura 14).

```
42      def Introducir_filas():
43
44          conexion.execute("insert into Llegadas values (?,?,?,?,?)", (codigo,identificador,tipo,fecha,foto))
45          conexion.commit()
```

Figura 15. Función para introducir filas en la tabla Llegadas.

En la figura 16, se define la función que permite mostrar por consola las filas de la tabla "Llegadas" en la base de datos "Entradas_Aves.db". Debe mencionarse que las fotos no se muestran, ya que al ser convertidas a binario solo veríamos una lista de unos y ceros, que no nos proporciona información útil.

```
47      def Mostrar_filas():
48
49          cursor=conexion.execute("select codigo,identificador,tipo,fecha from Llegadas")
50          for fila in cursor:
51              print(fila)
```

Figura 16. Función para mostrar filas de la tabla Llegadas.

En la figura 17, se define la primera parte de la función principal. En dicha parte se crea el objeto "reader" que permite realizar la lectura del tag RFID. Cada vez que se realiza una lectura, la variable global "codigo" se actualiza, de manera que sabemos el número de lecturas que se han realizado. Posteriormente, se lee la etiqueta RFID, guardando los datos, además del momento de la lectura y la toma de la fotografía.

```
53  def LeerTag():
54
55      global codigo
56      print("Leyendo")
57      reader = SimpleMFRC522()
58      codigo=codigo+1
59
60      try:
61
62          global identificador
63          identificador, nombre = reader.read()
64          now=datetime.now()
65          dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
66          print(identificador)
67          print(nombre)
68          print(dt_string)
69          Tomar_foto(identificador)
70
```

Figura 17. Función para leer tag parte 1.

En la figura 18, se guardan las fotos con el nombre apropiado dependiendo del identificador que porte el pájaro. Posteriormente se vuelca toda la información en la tabla "Llegadas" de la base de datos "Entrada_Aves.db". Por último se ponen a cero los valores de los pines y se espera un tiempo prefijado por el usuario para realizar otra lectura. Si el número de entradas en la tabla "Llegadas" es de 10, se manda por correo electrónico y se borra la tabla. Se reinicia la variable que cuenta el número de entradas.

```
71          if identificador==564391352100:
72
73              fotoB=convertToBinaryData("imagen_buitre.jpg")
74              print("Pájaro fotografiado")
75
76          if identificador==83483035521:
77
78              fotoB=convertToBinaryData("imagen_aguila.jpg")
79              print("Pájaro fotografiado")
80
81          Crear_Abrir_DB()
82          conexion.execute("insert into Llegadas(codigo,identificador,tipo,fecha,foto)
83          conexion.commit()
84          conexion.close()
85
86      finally:
87
88          GPIO.cleanup()
89          time.sleep(time_between_arrivals)
90          print (codigo)
91
92          if codigo==10:
93
94              Email()
95              Borrar_todas_filas()
96              codigo=0
```

Figura 18. Función para leer tag parte 2.

En la figura 19, se ejecuta el bucle principal del programa. Mediante una interrupción por teclado (combinación de teclas ctrl+c) se da por finalizado el programa.

```
try:

    while True:

        LeerTag()

except KeyboardInterrupt:
    print("Terminando de leer")
    pass
```

Figura 19. Bucle principal del programa.

En la figura 20, se esquematiza mediante un diagrama de flujo el funcionamiento del código explicado anteriormente.

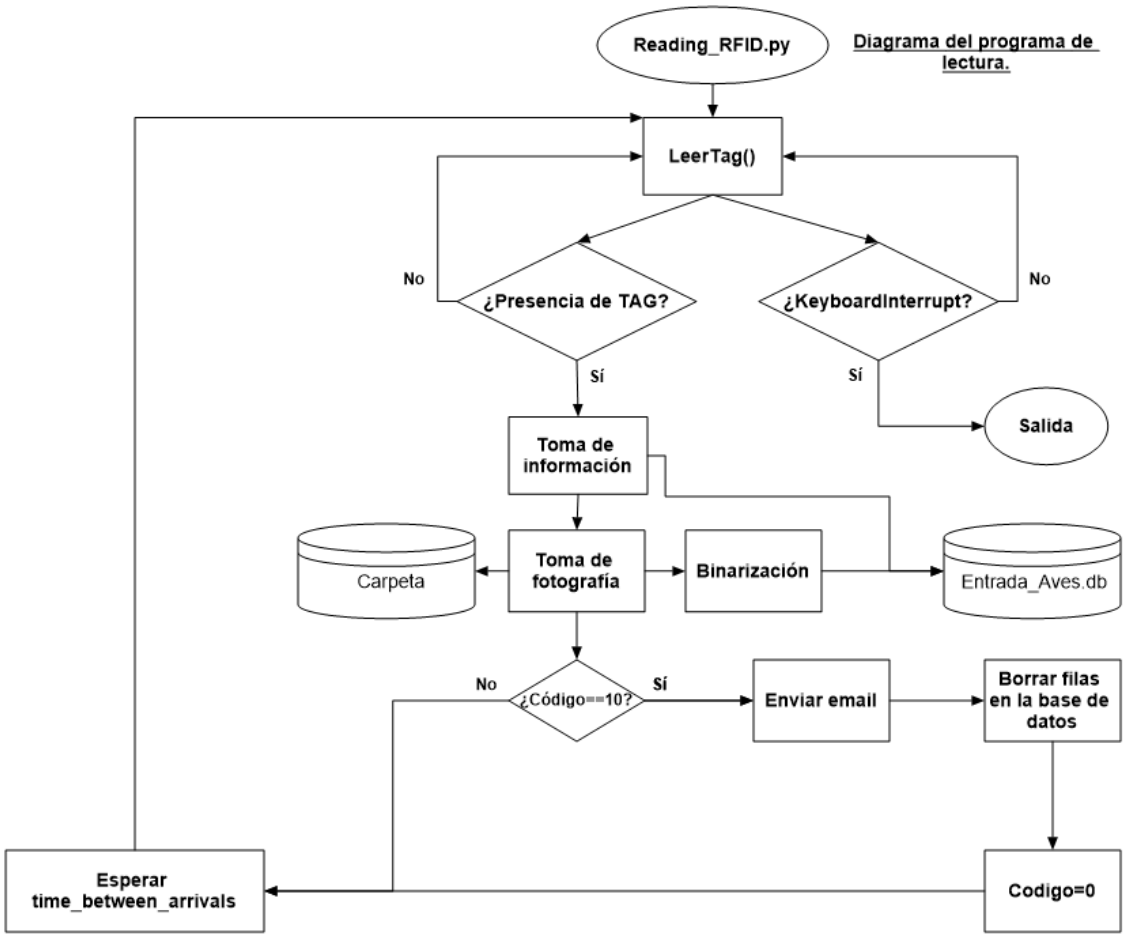

Figura 20.Diagrama de flujo Reading_RFID.py.

Writing_RFID.py

Como ya se mencionaba en el apartado 4.1, este programa realiza tres funciones. La primera función realiza la escritura de las etiquetas (figura 22). La segunda función cuenta el número de avistamientos que se han producido de cada pájaro que porta su identificador y muestra su última fotografía en una ventana (figura 23, figura 24 y figura 25). La última función cierra la interfaz gráfica (figura 27).

En la figura 21, se cargan todos aquellos módulos o librerías que contienen las funciones explicadas en el apartado 4.2.

```
5   import tkinter as tk
6   import RPi.GPIO as GPIO
7   from mfrc522 import SimpleMFRC522
8   import sqlite3
9   import time
10  from PIL import Image, ImageTk
```

Figura 21. Importado Writing_RFID.py.

En la figura 22, se define la función que permite escribir en el tag RFID. Para ello se define el objeto "reader" que contiene la información que lo permite. El texto necesario para escribir se obtiene del objeto "text" obtenido de un cuadro en el que se inserta. Finalmente se ponen a cero los pines utilizados.

```
10  def InsertarTag():
11
12      reader = SimpleMFRC522()
13
14      try:
15
16          print("Introduzca el nombre del ave que portará el identificador")
17          text=entry.get()
18          reader.write(text)
19          print("Nombre introducido satisfactoriamente.")
20
21      finally:
22
23          GPIO.cleanup()
```

Figura 22. Escribir tag.

En la figura 23, se define la primera parte de la función Datos, que cuenta el tipo de pájaros que se han identificado. Para ello utilizamos el número de identificador del tag RFID. De esta forma podemos identificar el número de pájaros de cada tipo y el total como la suma de los diversos pájaros.

```
28  def Datos():
29
30      Abrir_imagen()
31      con=sqlite3.connect("Entrada_Aves.db")
32      cur=con.cursor()
33      sqlite_select_query = """SELECT identificador from Llegadas where identificador=564391352100"""
34      cur.execute(sqlite_select_query)
35      records = cur.fetchall()
36      print("Numero total de buitres avistados: ", len(records))
37      sqlite_select_query2 = """SELECT identificador from Llegadas where identificador=83483035521"""
38      cur.execute(sqlite_select_query2)
39      records2 = cur.fetchall()
40      print("Numero total de aguilas avistadas: ", len(records2))
41      global recuentotal
42      recuentotal=records+records2
43      print("Numero total de aves avistadas: ", len(recuentotal))
```

Figura 23.Función datos parte 1.

En la figura 24, se define la segunda parte de la función datos. En esta se crea una pantalla secundaria que permite mostrar información relativa al número de avistamientos.

```
    top3=tk.Toplevel(window)
    top3.geometry("1280x400")
    top3.mainloop
    label4=tk.Label(top3,
                    text="Numero total de buitres avistados:"+str(len(records)),
                    borderwidth=4,
                    relief="ridge",
                    foreground="white",
                    background="black",
                    width=1280,
                    height=5
                    )
    label4.pack()
    label5=tk.Label(top3,
                    text="Numero total de aguilas avistadas:"+str(len(records2)),
                    borderwidth=4,
                    relief="ridge",
                    foreground="white",
                    background="black",
                    width=1280,
                    height=5
                    )
    label5.pack()
    label6=tk.Label(top3,
                    text="Numero total de aves avistadas:"+str(len(recuentotal)),
                    borderwidth=4,
                    relief="ridge",
                    foreground="white",
                    background="black",
                    width=1280,
                    height=200
                    )
    label6.pack()

```

Figura 24. Función datos parte 2.

En la figura 25, se define la función que permite crear dos ventanas emergentes para mostrar las últimas imágenes de los pájaros.

```
def Abrir_imagen():

    top=tk.Toplevel(window)
    top.geometry("1280x400")
    top.mainloop
    top2=tk.Toplevel(window)
    top2.geometry("1280x400")
    top2.mainloop
    image1 = Image.open("imagen_buitre.jpg
    test = ImageTk.PhotoImage(image1)

    label2 = tk.Label(top,image=test)
    label2.image = test

    # Posiciona imagen
    label2.place(x=0,y=0)
    image2 = Image.open("imagen_aguila.jpg
    test2 = ImageTk.PhotoImage(image2)

    label3 = tk.Label(top2,image=test2)
    label3.image = test2

    # Posiciona imagen
    label3.place(x=0,y=0)
```

Figura 25. Función para abrir imágenes.

En la figura 26, se define la primera parte de la interfaz gráfica principal. Primero, se crea la ventana “window”, con nombre “Menu principal. Despúes se le da un tamaño elegido. Se crea una etiqueta denominada “label” que da un mensaje de bienvenida. Posteriormente, se crea una entrada de texto. Por último se crea un botón “button”, que permite la interacción del usuario mediante el ratón. Tanto la etiqueta como el botón tienen diversas opciones de estilo.

```
43  window = tk.Tk(className='Menu principal')
44  #Tamaño de la ventana
45  window.geometry("1280x400")
46  #Creación de una etiqueta de texto que da la bienvenida
47  label = tk.Label(
48      text="Bienvenido al menú principal",
49      borderwidth=4,
50      relief="ridge",
51      foreground="white",
52      background="black",
53      width=1280,
54      height=5
55  )
56  label.pack()
57  #Creación de un espacio para entrada de texto
58  entry = tk.Entry()
59  entry.pack()
60  #Creación de un botón que permite insertar texto en el tag
61  button = tk.Button(
62      text="Escribir tag",
63      borderwidth=2,
64      relief="solid",
65      width=1280,
66      height=5,
67      bg="blue",
68      fg="yellow",
69      command=InsertarTag
70  )
71  button.pack()
```

Figura 26. Creación de la interfaz gráfica parte 1.

En la figura 27, se define la segunda parte de la interfaz gráfica. En ella se crean dos botones más “button2” y “button3”. Ambos tienen sus opciones de estilo propias y permiten la interacción del usuario. Por último se llama al bucle principal de la interfaz “window.mainloop()”, que permite ejecutar de forma cíclica todas las funciones pertenecientes a la interfaz gráfica.

```
137 button2 = tk.Button(
138     text="Obtener datos e imágenes",
139     borderwidth=2,
140     relief="solid",
141     width=1280,
142     height=5,
143     bg="green",
144     fg="black",
145     command=Datos
146 )
147 button2.pack()
148 #Creación de un botón que permite salir del programa
149 button3 = tk.Button(
150     text="Salir del programa",
151     borderwidth=2,
152     relief="solid",
153     width=1280,
154     height=5,
155     bg="red",
156     fg="black",
157     command=window.destroy
158 )
159 button3.pack()
160
161 #Bucle principal de la ventana de la interfaz gráfica
162 window.mainloop()
```

Figura 27.Creación de la interfaz gráfica parte 2.

6. Interacción del usuario con el programa.

El archivo **Writing_RFID.py,** cuyo funcionamiento ya se ha descrito en el apartado, implementa una interfaz gráfica mediante la librería tkinter, pensada para la interacción sencilla del usuario.
Tras ejecutar el archivo desde el terminal de comandos, el usuario visualiza la siguiente pantalla de menú (figura 28), que contiene una etiqueta, una entrada de texto y tres botones.

Figura 28. Menú principal.

El primer botón en la figura 28, denominado “Escribir tag”, permite al usuario introducir texto en la etiqueta. Para ello, debe escribir en el recuadro blanco localizado encima del botón. Una vez ha escrito el texto deseado, deberá pulsar el botón y colocar delante del lector RC522 la etiqueta, completándose así el proceso de inserción de datos de forma satisfactoria.

El segundo botón en la figura 28, denominado “Obtener datos e imágenes”, permite al usuario utilizar parte de la información de la base de datos. Una vez el usuario ha presionado el botón, tres ventanas emergentes se abrirán. La primera y la segunda muestran las últimas imágenes de las dos etiquetas con las que se ha elaborado la aplicación (figuras 29 y 30). La tercera ventana emergente contiene información del número de veces que se ha identificado cada etiqueta y del total (figura 31).

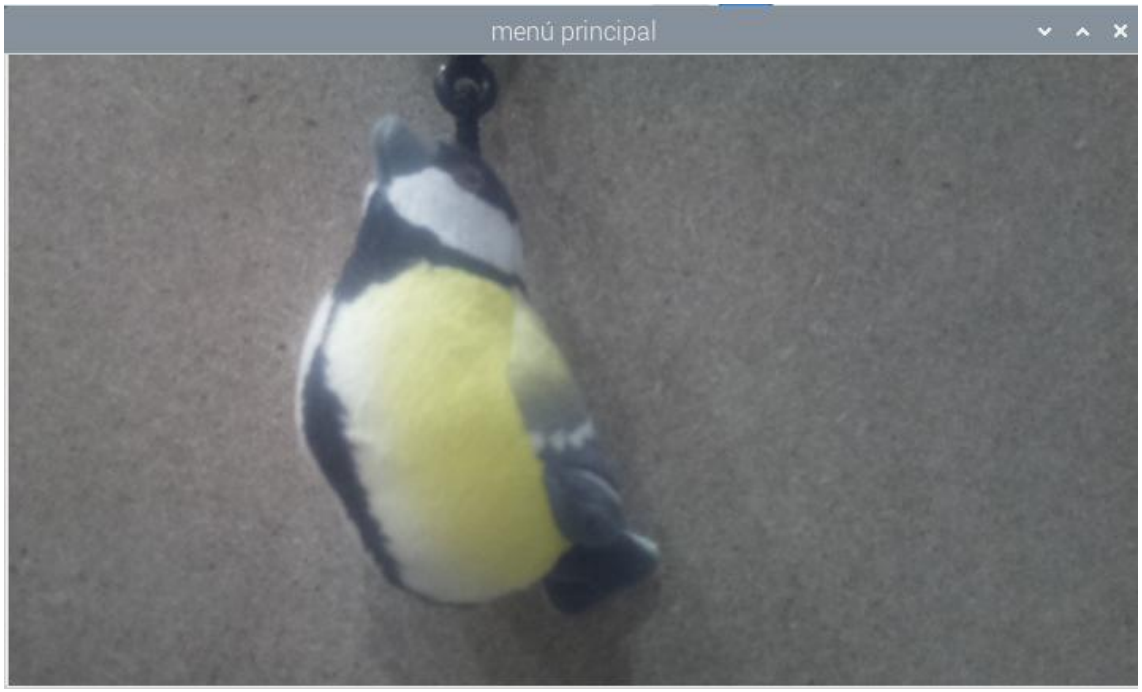

Figura 29. Foto de pájaro 1.

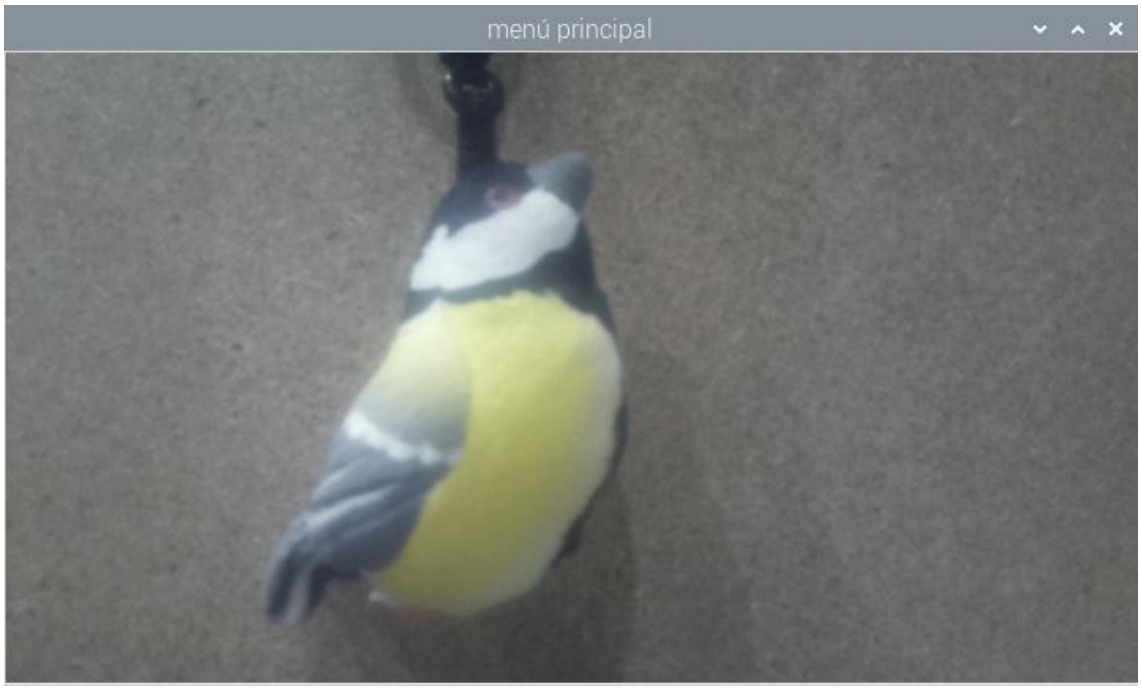

Figura 30. Foto de pájaro 2.

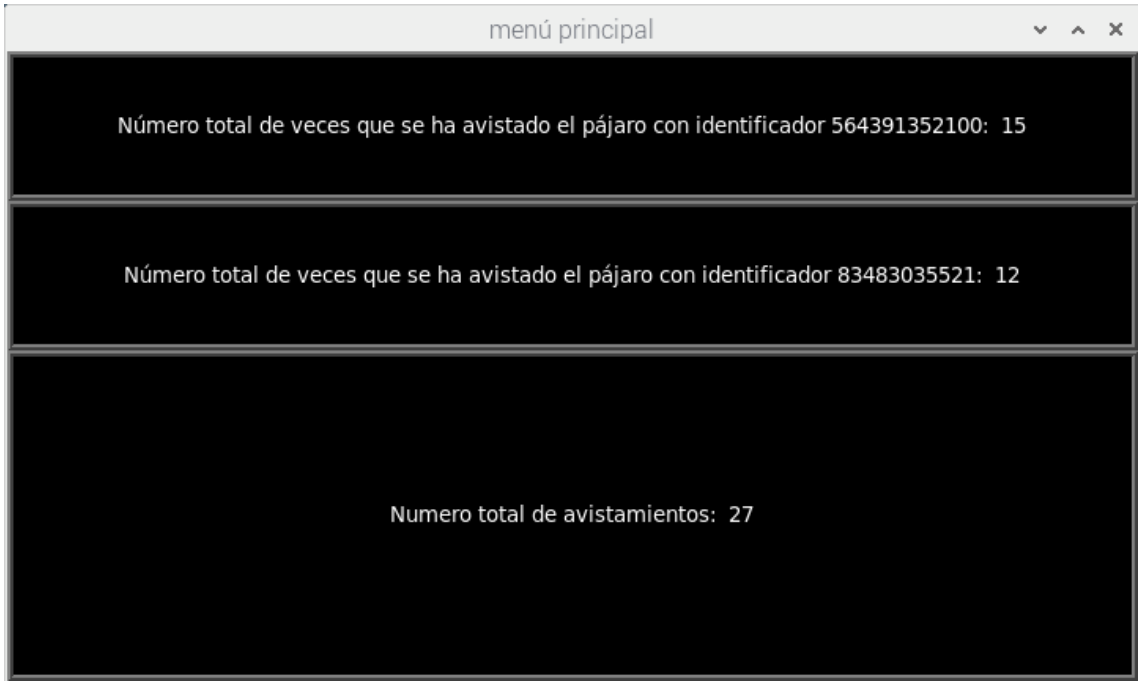

Figura 31. Ventana con datos.

7. Problemas durante el desarrollo.

La realización de este apartado tiene como objetivo, aparte de enriquecer el trabajo, la posibilidad de que sirva como ayuda para futuros estudiantes que realicen trabajos de índole parecida, que puedan tener problemas identificados aquí.

Uno de los problemas que he encontrado, y que además es común a todos los trabajos con sensores, es la **dificultad para detectar el origen de un funcionamiento incorrecto o el no funcionamiento**. Los sensores comerciales y más populares, como es el caso del RC522 en el presente trabajo, son como "cajas negras". Esto quiere decir, que en el momento en que el sensor tiene un comportamiento defectuoso, se puede realizar una cantidad de medidas reducidas para identificar su origen. Algunas **soluciones** recomendadas gracias a la experiencia propia serían las siguientes:

1. Comprobar todas las conexiones de los cables. Tanto las del ordenador monoplaca como la salida del sensor. Desconectar y conectar de nuevo.
2. Medir la continuidad de los cables utilizados mediante un polímetro, detectando de esta forma posibles cortes.
3. Medir la tensión entre la línea de alimentación y tierra del sensor. Una vez se esté ejecutando medir la tensión entre la línea de datos y la tierra del sensor, para comprobar si cambia con la acción que deba realizar.
4. Revisar las librerías y el código utilizados. Cabe destacar la importancia de constatar la versión del lenguaje de programación en la que fue creado, para encontrar posibles incompatibilidades entre el código propio y el de las librerías utilizadas.
5. Comprar un nuevo sensor.

Otro de los problemas que he encontrado, en este caso en el trabajo con Raspberry Pi (seguramente sea extrapolable a otros ordenadores monoplaca), es la **pérdida de datos de la tarjeta micro SD** que contiene el sistema operativo y toda la información que guardemos en su memoria. Este problema viene originado por la desconexión directa e inmediata de la fuente de alimentación, o de problemas de estabilidad en la tensión suministrada. En este caso son dos **soluciones** que deben aplicarse juntas:

- Comprobar de forma periódica la tensión suministrada por la fuente de alimentación. Tiene que ser la necesaria y con un valor estable. Se recomienda el uso de un osciloscopio para ello.
- Antes de retirar la alimentación, ejecutar en el terminal el siguiente comando: "**sudo shutdown –h now**". Una vez ejecutado, visualizaremos el led verde de la placa llamado "ACT". En el momento en el que este led deje de emitir luz, será segura la desconexión de la alimentación.

Otro de los problemas que he encontrado, específico del trabajo con Raspberry Pi, es la **incapacidad para correr el sistema operativo**. Este problema se presenta de forma que el led rojo denominado "PWR" está encendido tras alimentar la placa, pero el led verde denominado "ACT" no se enciende. Este parpadeo del led verde tras conectar a la alimentación la placa, indica que el sistema operativo instalado está ejecutándose. Por lo tanto, si esto no ocurre, el usuario será incapaz de visualizar el escritorio y de interactuar con los archivos .Alguna de las **soluciones** son las siguientes:

- Formatear la tarjeta micro SD y reinstalar el sistema operativo.
- Comprar una nueva tarjeta micro SD e instalar el sistema operativo.
- Existen varios tipos de programas gratuitos para formatear las tarjetas micro SD. Para la instalación del sistema operativo Raspbian se recomienda su propio programa, llamado "Raspberry Pi Imager".
- Comprar un nuevo ordenador monoplaca.

Por último, y un **problema** común a todos aquellos proyectos que se basen en el uso de diversos componentes electrónicos, es la posibilidad de adquirirlos, debido a la crisis mundial de escasez de semiconductores que se está dando actualmente.

8. Futuro de la aplicación.

Dado que este prototipo es de gran sencillez, es necesario hacer un bosquejo de la hoja de ruta que podría o debería seguir la aplicación para conseguir un funcionamiento que proporcione características adicionales. Para ello se va a hacer el análisis desde las siguientes dos perspectivas:

- **Hardware.**

La alimentación eléctrica del sistema debe de ser tal que permita la autonomía de la estación en los lugares remotos en los que va a ser colocada. De esta manera, aprovechando la pendiente del tejado, puede colocarse una pequeña placa solar, junto con una batería que suministre de la energía eléctrica necesaria a lo largo de todo el día. Es importante hacer hincapié en la necesidad de una buena etapa de filtrado, ya que la Raspberry Pi 3 B+ necesita de una alimentación constante y estable para su correcto funcionamiento.

Cambio de los sensores por unos más adecuados para la aplicación. Aunque en este trabajo se ha utilizado una antena lector RFID (RC522) de alta frecuencia, que ha permitido poner en funcionamiento el prototipo, para una aplicación con animales en la que las etiquetas deben inocularse dentro de la piel debe utilizarse uno de baja frecuencia, para que estas etiquetas puedan leerse a través de la piel. Además, esa capacidad en la baja frecuencia para atravesar sólidos no metálicos puede permitir encapsular la antena lector RFID, protegiéndola de colisiones accidentales y humedad. Otro punto a tener en cuenta sería el cambio de la cámara por otra que tenga un cableado más flexible, permitiendo posicionar la cámara de manera que las fotografías no estén desenfocadas y el ángulo sea adecuado.

- **Software.**

Obtención de una mayor cantidad de datos. Sería muy interesante poder generar gráficas con librerías Python como “Data Visualization”. Estas gráficas podrían representar el número de avistamientos a lo largo del tiempo, de esta manera se obtendría información visual de la localización aproximada del pájaro en cierto período, permitiendo así generar hipótesis sobre migraciones o períodos de reproducción.

Guardado y actualización de la base de datos en la nube. Gracias a esto ya no sería necesario mandar por correo electrónico la base de datos y eliminarla para liberar espacio de la Raspberry Pi 3 B+.

Aunque ya ha sido mencionado en la introducción de este mismo trabajo, **deberá realizarse la habilitación de internet** en la estación para poder comunicarse con las centrales que se ocupen de su gestión.

Aplicación de algoritmos de visión por computador para el procesado de imagen y la obtención de información. De esta manera, no solo se tomarán fotografías, sino que estás serán de mayor calidad, permitiendo la extracción de características para su posterior estudio (heridas en el pájaro, patrones extraños etc).

9. Resultados y conclusiones.

Finalmente, como una necesidad apremiante, me gustaría sincerarme con los **resultados** obtenidos.

Por una parte, resaltando los **elementos positivos**, se ha conseguido poner en funcionamiento la antena lector RFID, que es la tecnología en la que se basa la aplicación, realizando gracias a esta la **identificación de aves** que portan una etiqueta. Tanto la **lectura** como la **escritura** de estas etiquetas han sido realizadas con éxito. Además, estando en un plano más secundario, se ha puesto en funcionamiento la **cámara**, obteniendo fotografías que enriquecen el contenido de la base de datos creada.

La creación de una **interfaz gráfica**, aunque es tremendamente simple, genera un valor de interacción con el usuario, que da a la aplicación un enfoque interactivo que debería apreciarse.

Por último, la capacidad de comunicarse mediante el uso de internet con el exterior en forma de **correo electrónico**, me parece una parte importante que permite una realimentación sobre el estado de la estación, que aunque no es en tiempo real, se realiza en un período de tiempo corto relativo al tipo de aplicación.

Por otra parte, resaltando los **elementos negativos**, la **elección de la antena lector RFID** podría ser algo a tener en cuenta. Como ya se ha hablado en el apartado 8 de este mismo trabajo, la antena lector RFID más adecuada no habría sido la RC522, que es de alta frecuencia, sino otra de baja frecuencia, mucho mejor dada la naturaleza del proyecto. De todas formas la antena lector RC522 tiene un buen funcionamiento en presencia de humedad, que es un factor bastante característico de los lugares donde se va a operar. Además gracias a la existencia de librerías, este sensor ha sido puesto en funcionamiento, haciendo la aplicación posible.

También, y aunque quizás no se considere una competencia importante en el presente grado, **el código de programación** que implementa la aplicación debería ser mucho más genérico y eficiente.

Como **conclusión**, me gustaría realizar una reflexión sobre como las **competencias aprendidas** durante el estudio del presente grado, me han permitido plantar cara de la forma más objetiva a los problemas acaecidos durante el desarrollo del proyecto (ver apartado 7).

En un principio y dado que **Python** es un lenguaje informático en el que apenas tenemos tiempo para profundizar durante el grado, tuve dificultades a la hora de entender el código de algunas librerías. Sin embargo, gracias a la capacidad de **resolver problemas y la visión crítica** que se nos inculca durante estos cuatro años, fui capaz mediante la bibliografía de la que disponía de sacar adelante este proyecto. Esto lo considero muy importante y digno de mención, ya que en el día de mañana, a la hora de desarrollar mi actividad laboral en la ingeniería, lo que más me va ayudar son estas capacidades, dado que cada problema por afrontar es un mundo nuevo que requiere de herramientas básicas prácticas y teóricas.

También quiero resaltar el valor extra que me han aportado algunas asignaturas como **electrotecnia, fundamentos de electrónica y electrónica analógica** para resolver, o por lo menos para proporcionarme las herramientas necesarias en el trabajo práctico con sensores.

Por supuesto, y para concluir, la asignatura que me dio la motivación para realizar este trabajo de fin de grado fue **sistemas electrónicos programables**, cuyo estudio me proporcionó bastante satisfacción.

10. Bibliografía.

Nota: Todas las páginas web proporcionadas en la bibliografía son válidas y accesibles en junio de 2022.

Introducción:

[1] https://www.nacsinfo.com/fauna/fauna-autoctona/aves/
[2] https://cirhe.com/las-aves-en-el-medioambiente/
[3] https://cirhe.com/aves-en-peligro-de-extincion-en-espana/
[4] https://www.starlink.com/
[5] David N.Bonter, Eli S.Bridge. "*Applications of radio frequency identification (RFID) in ornithological research: a review*". 23 de febrero de 2011.

Marco teórico:

[6] Klaus Finkenzeller, "*RFID handbook. Fundamentals and applications in contactless smart cards, radio frequency identification and near-field communication, third edition*". 2010, John Wiley & Sons, Ltd.
[7] https://www.trovan.com/sp/RFID-FAQ/tipos-de-RFID

Hardware:

[8] https://pimylifeup.com/raspberry-pi-rfid-rc522/
[9] https://www.etechnophiles.com/raspberry-pi-3-b-pinout-with-gpio-functions-schematic-and-specs-in-detail/
[10] https://beebom.com/how-use-windows-laptop-as-monitor-raspberry-pi/

Software:

[11] David Beazley, Brian K. Jones, "*Python Cookbook THIRD EDITION*". Mayo de 2013 O´Reilly Media, Inc.
[12] https://www.raspberrypi.com/software/operating-systems/
[13] https://www.geany.org/
[14] https://sqlitebrowser.org/
[15] https://www.spyder-ide.org/

Funcionamiento de la aplicación:

[16] https://docs.python.org/3/library/sqlite3.html
[17] https://realpython.com/python-gui-tkinter/

[18] https://bc-robotics.com/tutorials/sending-email-using-python-raspberry-pi/
[19] https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/2
[20] https://www.nchsoftware.com/chart/es/index.html

Problemas durante el desarrollo:

[21] https://www.fororaspberry.es/viewtopic.php?t=5532
[22] https://forums.raspberrypi.com/viewtopic.php?t=202219

Anexo I
Código Camera.py

```
8   #Cargado de las funciones necesarias para el funcionamiento óptimo
9   from picamera import PiCamera
10  from time import sleep
11
12  #Función a la que se le pasa como argumento el identificador leído en la etiqueta
13  def Tomar_foto(identificador):
14
15  #Se crea el objeto camera y se gira 180 grados ya que está instalada al revés
16      camera=PiCamera()
17      camera.rotation=180
18      camera.start_preview()
19      sleep(5)
20  #Nombre de guardado según identificador
21      if identificador==564391352100:
22
23          camera.capture('/home/pi/pi-rfid/imagen_buitre.jpg')
24
25      if identificador==83483035521:
26
27          camera.capture('/home/pi/pi-rfid/imagen_aguila.jpg')
28  #Se cierra la cámara
29      camera.stop_preview()
30      camera.close()
```

Figura 32. Código Camera.py.

Anexo II
Código MandarEmail.py

```python
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email import encoders
#Funcion que permite enviar email
def Email():
    mail_content = '''Se están enviando satisfactoriamente los datos de 10 avistamientos.
    '''
    #Dirección email y contraseña
    sender_address = '1998razorsedge@gmail.com'
    sender_pass = '************'
    receiver_address = '1998razorsedge@gmail.com'
    #Setup de MIME
    message = MIMEMultipart()
    message['From'] = sender_address
    message['To'] = receiver_address
    message['Subject'] = 'Base de datos aves.'
    #Linea de asunto
    #Cuerpo y adjuntos del email
    message.attach(MIMEText(mail_content, 'plain'))
    attach_file_name = 'Entrada_Aves.db'
    attach_file = open(attach_file_name, 'rb') # Abre el archivo en modo binario
    payload = MIMEBase('application', 'octate-stream')
    payload.set_payload((attach_file).read())
    encoders.encode_base64(payload) #Encode del adjuntado

    payload.add_header('Content-Decomposition', 'attachment', filename=attach_file_name)
    message.attach(payload)
    #Crea sesión SMTP para mandar el email
    session = smtplib.SMTP('smtp.gmail.com', 587) #Uso del puerto gmail
    session.starttls() #Habilitar seguridad
    session.login(sender_address, sender_pass) #Nos metemos en el correo con las credenciales
    text = message.as_string()
    session.sendmail(sender_address, receiver_address, text)
    session.quit()
    print('Email enviado.')
```

Figura 33. Código MandarEmail.py