AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: GR Team

Title of WP: WP3 – AI-SAVES@ProjectHub

Activity Reference: 3.2 – Project template for AI-SAVES@ProjectHub

Logo:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Table of Contents **2**

1. Project Title 3

2. Project Category 3

3. Project Description 3

4. Keywords 4

5. Target Audience 4

6. Project Objectives 4

7. Project Timeline 4

8. Resources Provided 5

9. Tools and Technologies Used 5

10. Step-by-Step Implementation Plan 5

11. Evaluation Criteria 9

12. Languages Available 9

13. Contributors 9

14. Feedback and Continuous Improvement 10

Submission Notes **10**

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

AI-SAVES@PROJECTHUB

---

1. Project Title:

Smart Bird AI – Intelligent Bird Detection System

---

2. Project Category:

AI / Computer Vision / Smart Environment

---

3. Project Description:

This project presents a smart AI-powered bird monitoring system that automatically detects and identifies birds using a camera and a Raspberry Pi.
Using computer vision and a lightweight AI model, the system captures images of birds visiting a feeder and classifies them.
The project demonstrates how artificial intelligence can be used for wildlife monitoring, biodiversity education, and environmental STEM learning.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

4. Keywords:

AI, Computer Vision, Bird Detection, Raspberry Pi, Wildlife Monitoring, Smart Environment

---

5. Target Audience:

VET students, secondary school students, STEM clubs, beginner AI developers

---

6. Project Objectives:

- Understand the basics of computer vision and AI image classification.
- Learn how to connect a camera module to a microcomputer.
- Implement an AI model to detect birds.
- Store and analyze observation data.
- Explore applications of AI in environmental monitoring.

---

7. Project Timeline:

Day 1 – Introduction to AI and wildlife monitoring
Day 2 – Hardware setup (Raspberry Pi and camera)
Day 3 – Installing software and AI libraries
Day 4 – Running the bird detection model
Day 5 – Data logging and visualization

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

8. Resources Provided:

- Raspberry Pi
- Raspberry Pi Camera Module
- Bird feeder or outdoor monitoring point
- MicroSD card with Raspberry Pi OS
- Internet connection

---

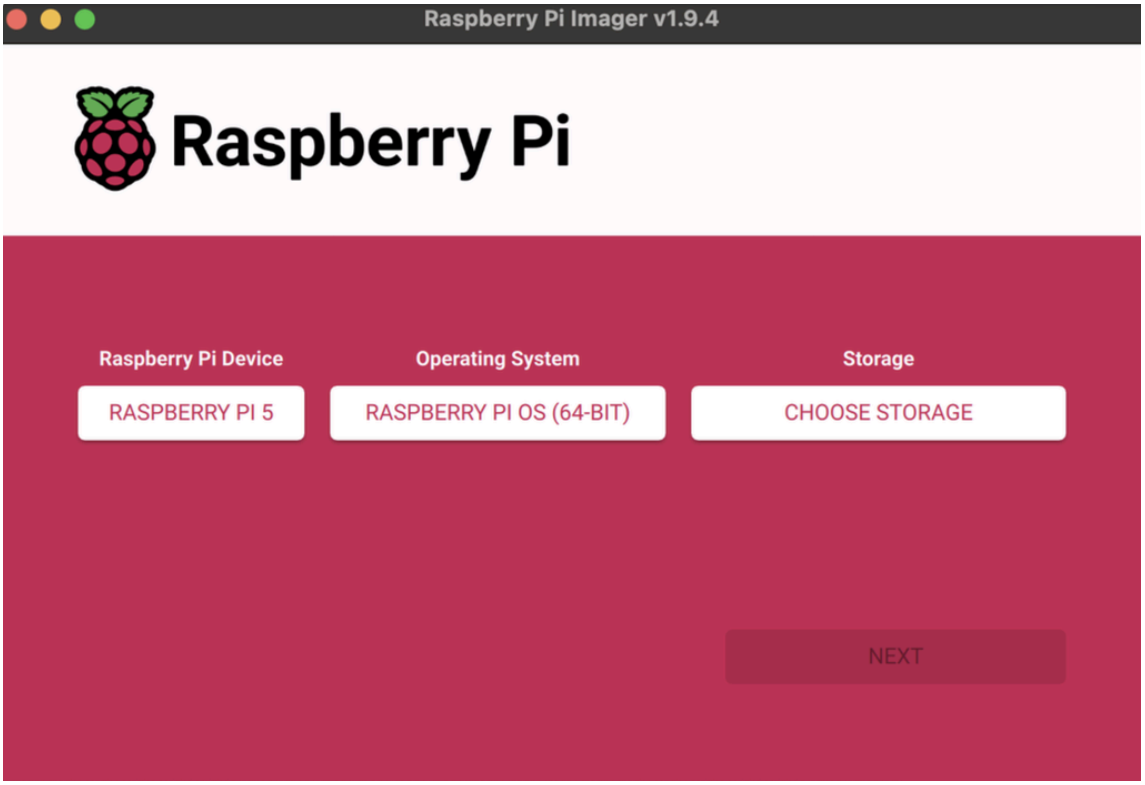
9. Tools and Technologies Used:

- Raspberry Pi
- Python
- OpenCV
- TensorFlow Lite
- Camera Module
- Optional cloud dashboard for observations

---

10. Step-by-Step Implementation Plan:

Step 1: Flashing the MicroSD Card With Raspberry Pi OS



1. To install Raspberry Pi OS, insert your microSD card into a USB adapter and connect it to your

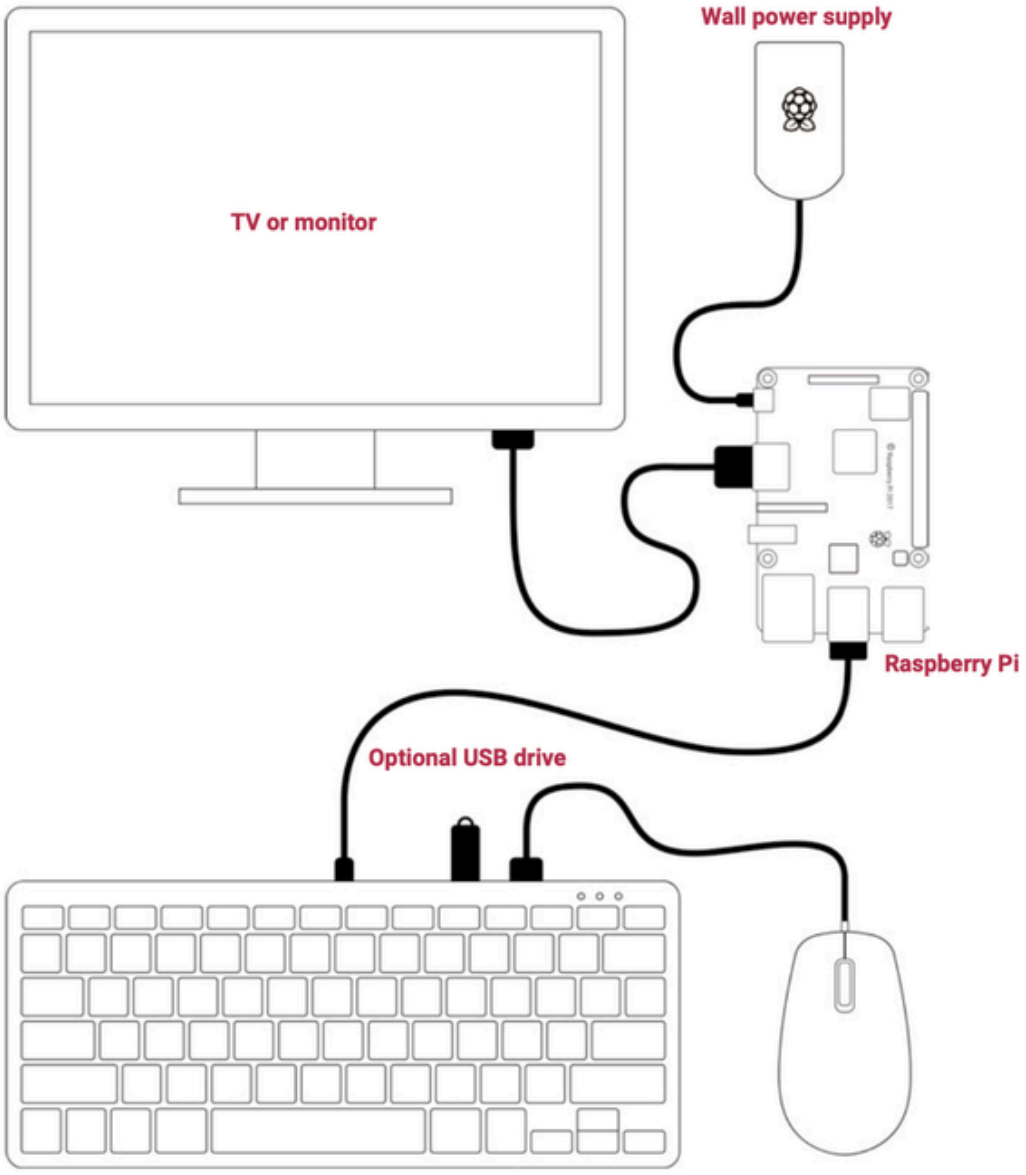
Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

computer.

2. Download and install the [Raspberry Pi Imager] software from the official Raspberry Pi website.
3. Open Imager, and configure it to the settings provided in the image
4. *Write* to flash the OS. This will take around 30 minutes.
5. Once the process is complete, remove the microSD card, insert it into the microSD slot of your Raspberry Pi, and power it on — your Raspberry Pi will boot into the OS.

Step 2: Raspberry Pi Setup



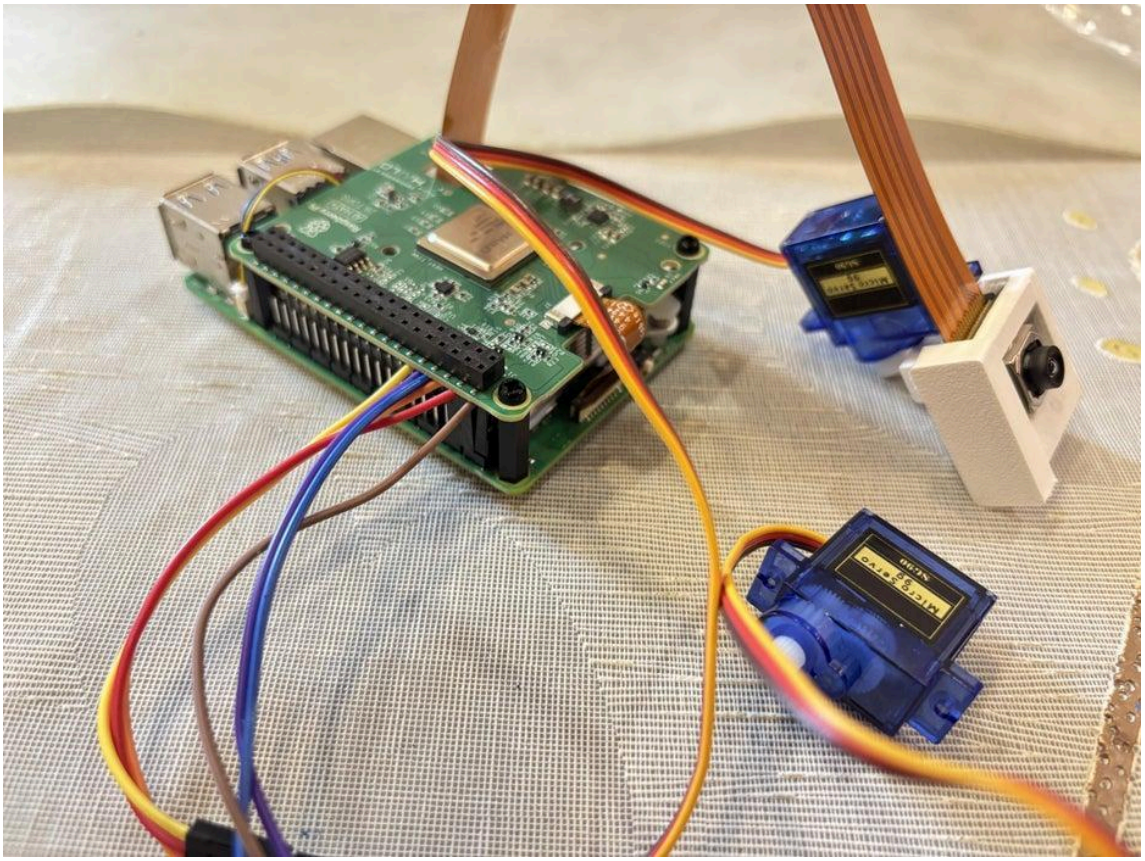
CREDITS: raspberrypi.com/documentation

A Raspberry Pi is essentially a computer with lower specs, but it still functions the same way as a computer. It still requires a monitor/screen and input commands (keyboard & mouse).
To configure this properly, connect both the keyboard and mouse to the Raspberry Pi via USB 3.0 (blue-colored ports). Afterward, insert the Micro-HDMI cable into the Raspberry Pi and the HDMI side into your monitor. Power on the Raspberry Pi, wait for a couple of seconds, and you will arrive at the setup page.
On this page, you will be prompted to enter admin information such as your Raspberry Pi username and password, your Wi-Fi SSID and password, as well as location and language preferences — fill in this information!

Step 3: Wiring the Camera and Mounting the AI Hat Module Onto the Raspberry Pi 5

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



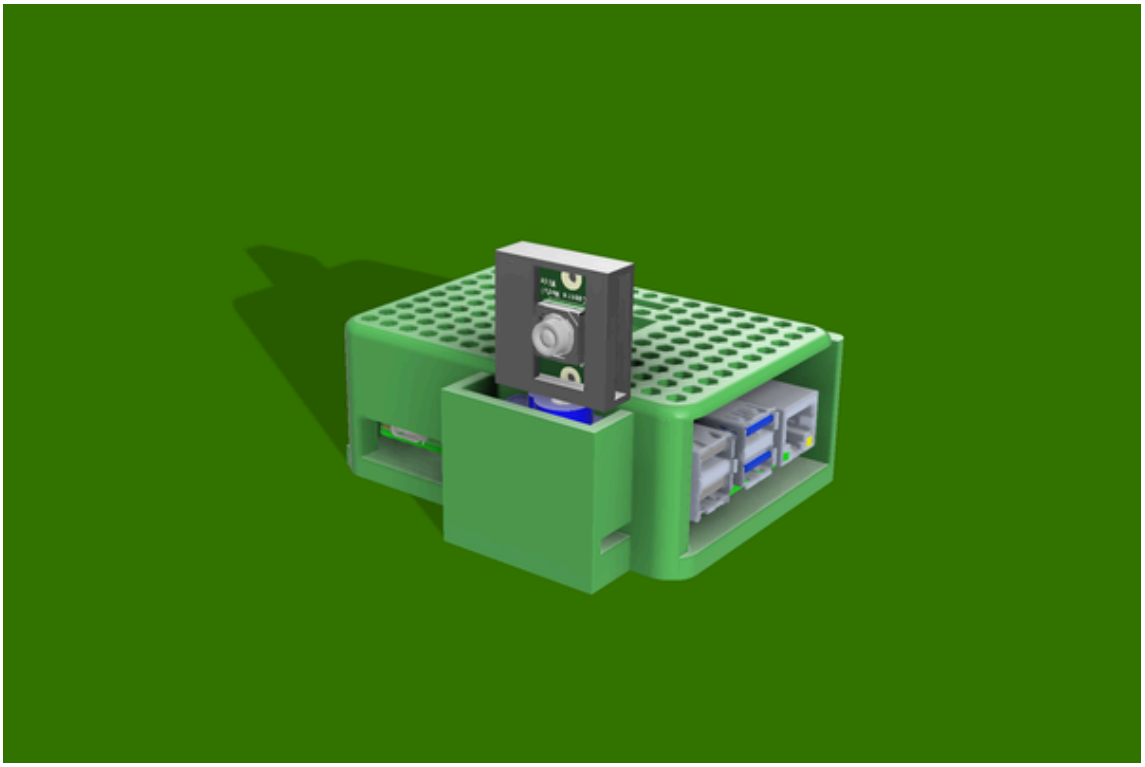
To set up the Raspberry Pi 5 with the camera and AI Hat, start by connecting the camera. Gently lift the black plastic part on the camera port, then insert the small end of the camera cable with the shiny contacts facing the right way. Push the black part back down to lock it in place.
It's better to connect the camera first because it gives you more space and makes things easier. After that, place the AI Hat on top of the Raspberry Pi, ensuring the connectors align.
Use the short screws at the bottom of the Raspberry Pi and the long screws on top to hold the AI Hat. Once the AI Hat is attached, it becomes more difficult to remove or adjust the camera without first removing the Hat.
Ensure that you also wire the servo motor into the Raspberry Pi via the GPIO pins. Please follow the wiring:

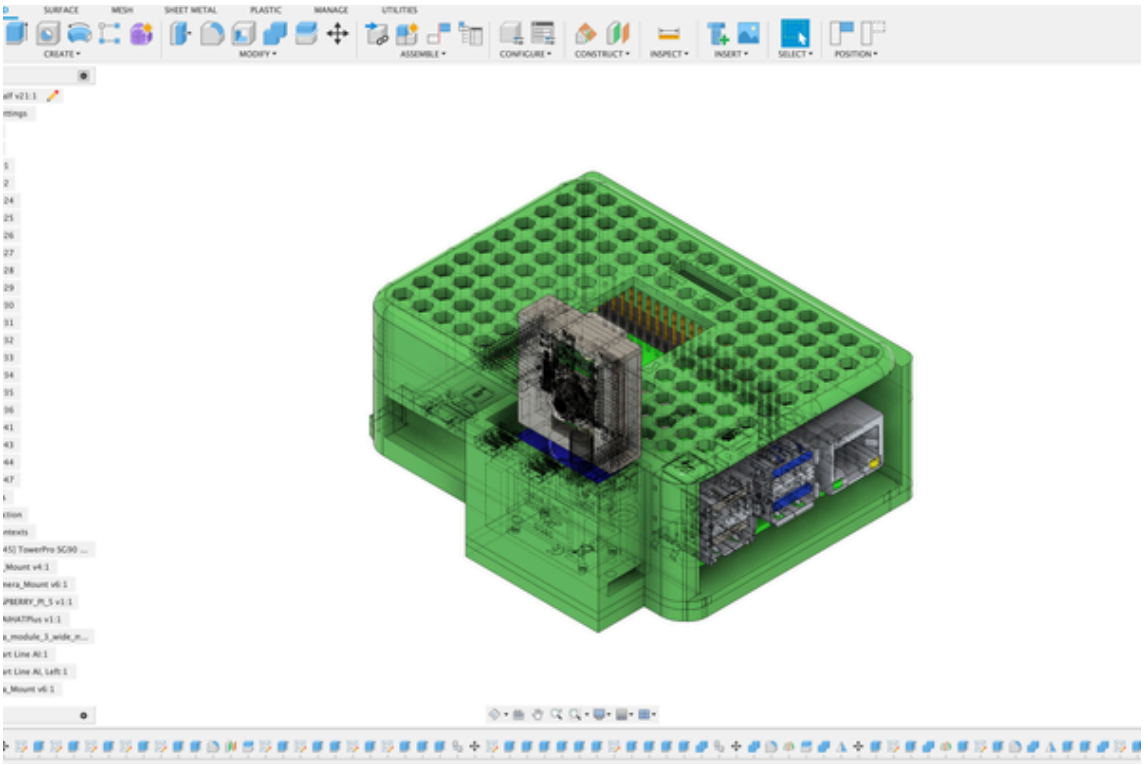
1. Brown Wire ⟶ GND
2. Red Wire ⟶ 5V
3. Yellow Wire ⟶ GPIO 17

Step 4: 3D Render of Casing

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

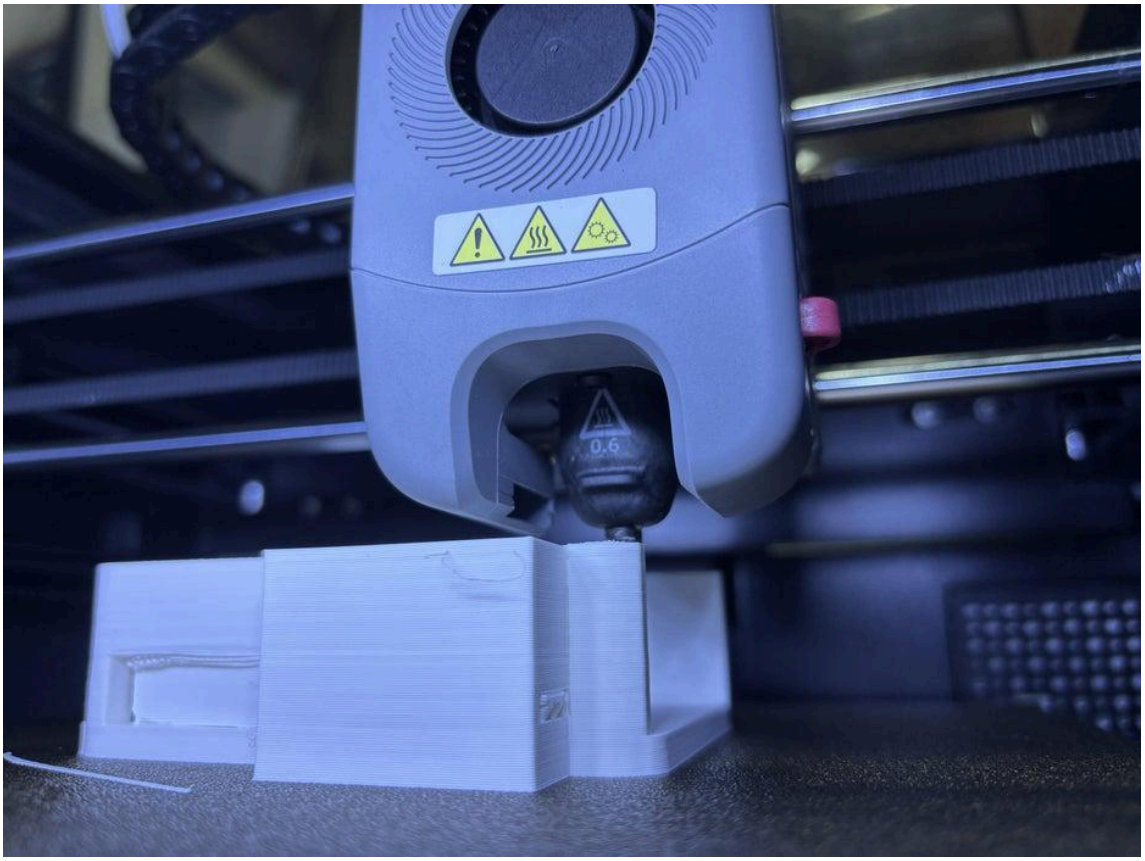




I created this CAD using **Autodesk Fusion 360**. The first image is a render, but the second can help you understand the specific tools I used. I have also used Autodesk to understand the strengths of this model to ensure the production would be sufficient.
To view the 3D render of the casing, please view the above casing. This can also help in visualizing possible changes for your project.
[Smart Line Bird, Version 1 v12.stl]
Step 5: Safety in 3D Printing

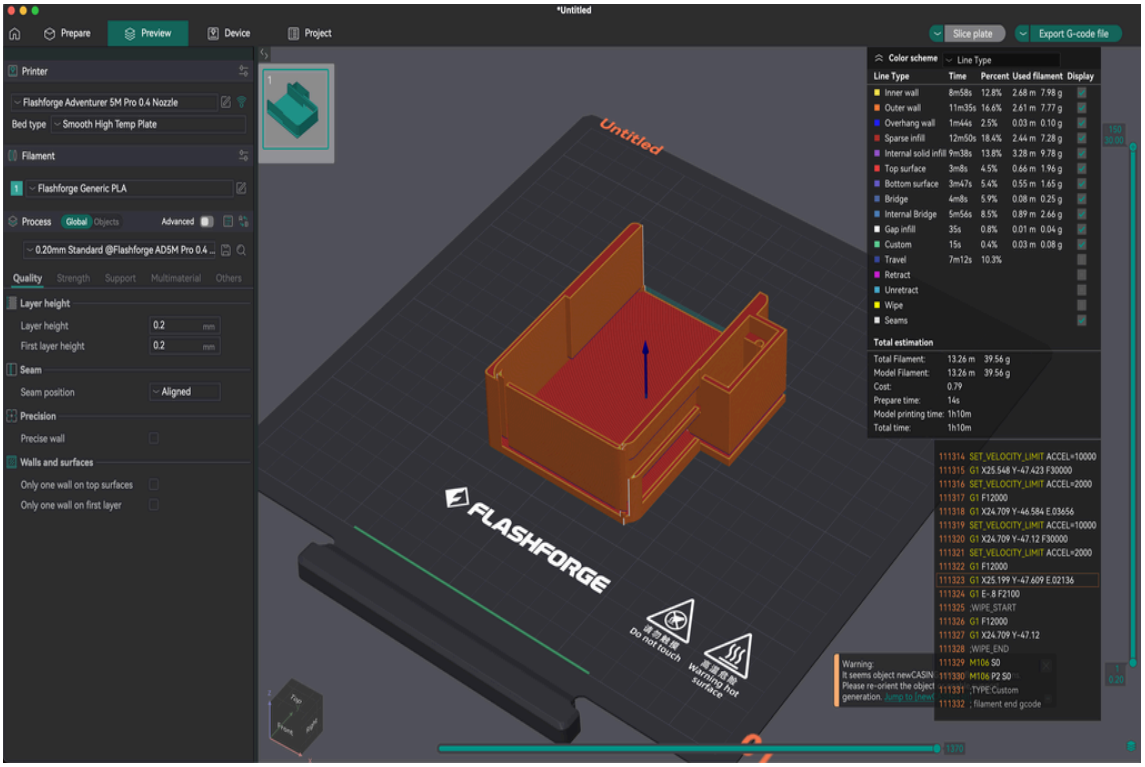
Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



When you are removing your 3D print from the bed post printing, PLEASE ensure that you wait for 5 minutes, allowing the plate to cool down. This will prevent any possible hazards, such as burns.

Step 6: 3D Printing the Casing



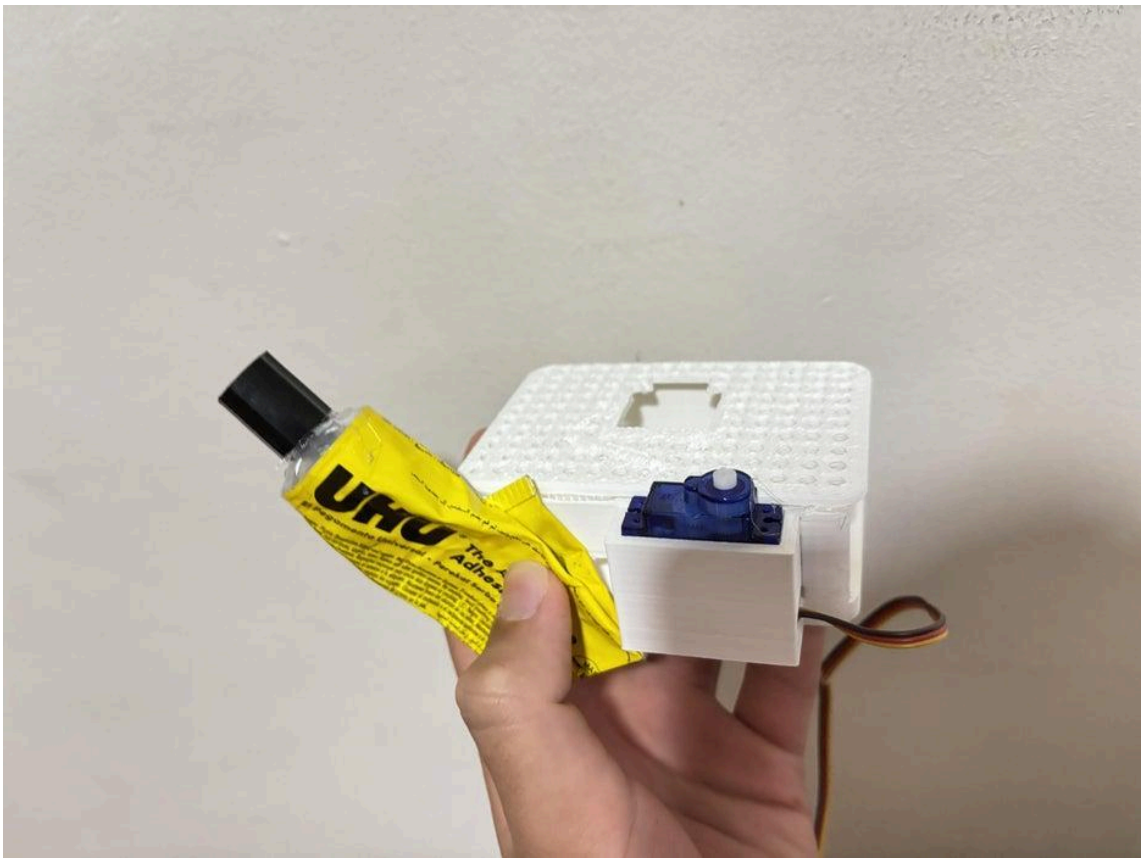
Please download all of the .STL files that I added in the Supporting Files section and place them into your slicer tool, such as Orca or FlashPrint. For this, I would recommend Orca because you can customize the

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

infill of the file, meaning you can set it to 100% in regions around the screw areas and lower elsewhere. I would not recommend setting it to 100% everywhere since this makes the module extremely heavy. Check out the image where I configured this in Orca.
After the print is completed, remember to remove any supports.

Step 7: Gluing



Glue the parts together using UHU adhesive glue. First, start by gluing the top to the bottom of the case. If you have clamps, this would be a good time to use them; otherwise, clips also work great.
Next, glue the casing to the mount using the same adhesive. I didn't use clamps here, but there was a very large surface area for the glue to set firmly. UHU starts to set after 30 minutes, but I left it overnight to ensure it would be strong.

Step 8: Installation

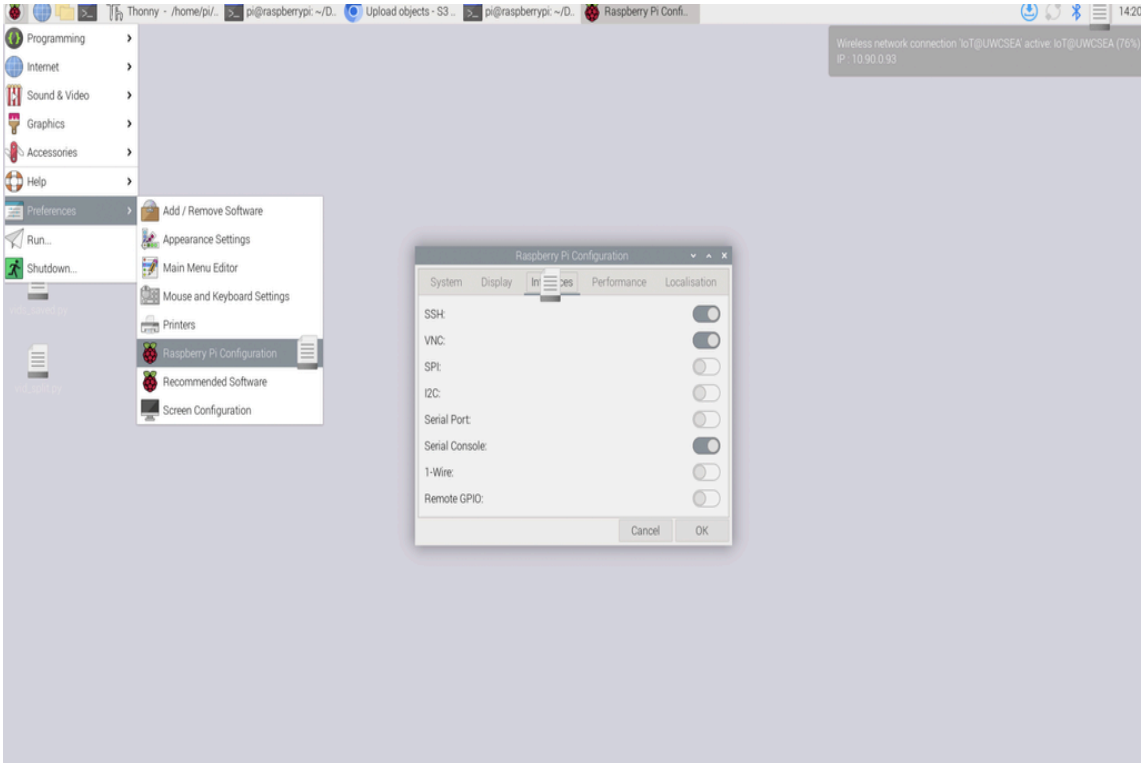
Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



The installation of the module is extremely simple; just place the components in an outdoor environment that is near the field of view!

Step 9: Setting Up VNC and Ensuring Camera Works



Currently, when you connect to your Raspberry Pi, you need to set up your monitor, keyboard, and mouse, which is a huge hassle. Although there are simpler alternatives, such as using a tool called VNC. This allows you to access your Raspberry Pi without a monitor, keyboard, or mouse — all you need is

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

power.
To set this up, click on the Raspberry Pi logo, go into Raspberry Pi Configuration, go to *Interfaces*, and toggle the VNC tab on. After doing this, go to your computer and install RealVNC.
After enabling VNC, you need to find out the IP address of your Raspberry Pi. To do this, you can either look through the Raspberry Pi UI or use an IP scanner. To find it using the UI, move the cursor over the Wi-Fi logo, and the IP address will show. Simply enter those numbers into RealVNC on your computer and type in your username and password. Then you will have complete control through a headless setup. If you wish to use an IP scanner, search for IPs and enter that number into RealVNC.

Step 10: Testing Data With Code
Now for the fun part - code. First, we need to collect the videos that we want to analyse. I recommend around 5 hours of footage for this. Feel free to use the code below that helps to collect data and store it in a local folder called 'videos'.

The videos that you would be filming for this step would just be the outdoor area that you are interested in viewing.

```
from picamera2 import Picamera2
from picamera2.encoders import H264Encoder
from picamera2.outputs import FfmpegOutput
from datetime import datetime, timedelta, time as dtime
import os
import time
import sys

def next_window(now):
today = now.date()
start_today = datetime.combine(today, dtime(12, 20, 0))
end_today = datetime.combine(today, dtime(14, 0, 0))

if now < start_today:
return start_today, end_today
elif now < end_today:
return now, end_today
else:
tomorrow = today + timedelta(days=1)
start_tomorrow = datetime.combine(tomorrow, dtime(12, 20, 0))
end_tomorrow = datetime.combine(tomorrow, dtime(14, 0, 0))
return start_tomorrow, end_tomorrow

def main():
out_dir = os.path.join(os.getcwd(), "videos")
os.makedirs(out_dir, exist_ok=True)

now = datetime.now()
start_dt, end_dt = next_window(now)

stamp = start_dt.strftime("%d%m%Y")
filename = f"{stamp}_video.mp4"
out_path = os.path.join(out_dir, filename)

wait_seconds = (start_dt - datetime.now()).total_seconds()
if wait_seconds > 0:
print(f"waiting until {start_dt.strftime('%Y-%m-%d %H:%M:%S')}...")
try:
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
time.sleep(wait_seconds)
except KeyboardInterrupt:
print("\nInterrupted while waiting. Exiting.")
sys.exit(0)

duration = max(0, (end_dt - datetime.now()).total_seconds())
if duration == 0:
print("[Scheduler] No time left in the window. Exiting.")
return

print(f"Starting capture at {datetime.now().strftime('%H:%M:%S')} "
f"for {int(duration)} seconds (until {end_dt.strftime('%H:%M:%S')}).")
print(f"[Recorder] Output: {out_path}")

picam2 = Picamera2()
video_config = picam2.create_video_configuration(main={"size": (1920, 1080)})
picam2.configure(video_config)

encoder = H264Encoder(bitrate=10_000_000)
output = FfmpegOutput(out_path)

try:
picam2.start_recording(encoder, output)
time.sleep(duration)
except KeyboardInterrupt:
print("\n[Recorder] Interrupted. Stopping recording...")
finally:
try:
picam2.stop_recording()
except Exception:
pass
try:
picam2.close()
except Exception:
pass

print(f"Finished at {datetime.now().strftime('%H:%M:%S')}. Saved: {out_path}")

if __name__ == "__main__":
  main()
```

If you would like to find the code that will send a notification to your email when a bird is detected, please use the following code. Note, you need to replace the vairables with the relevent email address and passwords.

```
import argparse
import os
import smtplib
import ssl
import time
from datetime import datetime
from email.message import EmailMessage
from email.utils import formatdate

import cv2
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
import numpy as np
import pandas as pd
import torch

SMTP_SERVER = "smtp.gmail.com"
SMTP_PORT = 587
GMAIL_SENDER = "youremail@gmail.com"
GMAIL_APP_PASSWORD = "xxxx xxxx xxxx xxxx"
GMAIL_RECIPIENT = "recipient@example.com"

CONFIDENCE_THRESHOLD = 0.35
NMS_IOU_THRESHOLD = 0.45
SNAPSHOT_DIR = "snapshots"
PREVIEW_WINDOW_NAME = "Bird Watch"

def send_email_with_image(subject, body, image_path):
    msg = EmailMessage()
    msg["Subject"] = subject
    msg["From"] = GMAIL_SENDER
    msg["To"] = GMAIL_RECIPIENT
    msg["Date"] = formatdate(localtime=True)
    msg.set_content(body)
    if image_path and os.path.exists(image_path):
        with open(image_path, "rb") as f:
            data = f.read()
        ext = os.path.splitext(image_path)[1].lower()
        maintype, subtype = ("image", "jpeg") if ext in [".jpg", ".jpeg"] else ("image", "png")
        msg.add_attachment(data, maintype=maintype, subtype=subtype,
filename=os.path.basename(image_path))
    context = ssl.create_default_context()
    with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
        server.ehlo()
        server.starttls(context=context)
        server.ehlo()
        server.login(GMAIL_SENDER, GMAIL_APP_PASSWORD.replace(" ", ""))
        server.send_message(msg)

def draw_boxes(frame, detections_df, color=(0, 255, 0)):
    for _, row in detections_df.iterrows():
        if row["name"] != "bird":
            continue
        x1, y1, x2, y2 = int(row["xmin"]), int(row["ymin"]), int(row["xmax"]), int(row["ymax"])
        conf = float(row["confidence"])
        label = f"bird {conf:.2f}"
        cv2.rectangle(frame, (x1, y1), (x2, y2), color, 2)
        cv2.putText(frame, label, (x1, max(0, y1 - 6)), cv2.FONT_HERSHEY_SIMPLEX, 0.6, color, 2,
cv2.LINE_AA)
    return frame

def save_snapshot(frame, folder):
    os.makedirs(folder, exist_ok=True)
    ts = datetime.now().strftime("%Y%m%d_%H%M%S")
    path = os.path.join(folder, f"bird_{ts}.jpg")
    cv2.imwrite(path, frame)
    return path
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
def run(video_path, cooldown_s):
    model = torch.hub.load("ultralytics/yolov5", "yolov5s", pretrained=True)
    model.conf = CONFIDENCE_THRESHOLD
    model.iou = NMS_IOU_THRESHOLD
    model.classes = None
    model.max_det = 300

    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        raise RuntimeError(f"Could not open video file: {video_path}")

    last_email_time = 0.0

    try:
        while True:
            ok, frame = cap.read()
            if not ok:
                break
            results = model(frame, size=640)
            df = results.pandas().xyxy[0]
            bird_df = df[(df["name"] == "bird") & (df["confidence"] >= CONFIDENCE_THRESHOLD)]
            display_frame = frame.copy()
            if not bird_df.empty:
                display_frame = draw_boxes(display_frame, bird_df)
            cv2.imshow(PREVIEW_WINDOW_NAME, display_frame)
            if cv2.waitKey(1) & 0xFF == ord("q"):
                break
            now = time.time()
            if not bird_df.empty and (now - last_email_time >= cooldown_s):
                snapshot_path = save_snapshot(display_frame, SNAPSHOT_DIR)
                count = len(bird_df)
                top_conf = float(bird_df["confidence"].max())
                subject = f"[Bird Alert] {count} bird(s) detected - {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}"
                body = f"Detected {count} bird(s) with top confidence {top_conf:.2f}.\nSource:
{video_path}\nSnapshot: {os.path.basename(snapshot_path)}\nCooldown: {cooldown_s} seconds\n"
                try:
                    send_email_with_image(subject, body, snapshot_path)
                    last_email_time = now
                except Exception as e:
                    print(f"Email send failed: {e}")
    finally:
        cap.release()
        cv2.destroyAllWindows()

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("--video", type=str, default="input.mp4")
    parser.add_argument("--cooldown", type=int, default=180)
    args = parser.parse_args()
    return args.video, args.cooldown

if __name__ == "__main__":
    vid, cd = parse_args()
    run(vid, cd)
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

If you would like to use the code that also provides an interesting graph of the number of birds present at specific times, please use the following code:

```
import argparse
import os
import csv
from datetime import datetime

import cv2
import pandas as pd
import torch
import matplotlib.pyplot as plt

CONFIDENCE_THRESHOLD = 0.35
NMS_IOU_THRESHOLD = 0.45
CSV_FILE = "bird_counts.csv"

def run(video_path):
    model = torch.hub.load("ultralytics/yolov5", "yolov5s", pretrained=True)
    model.conf = CONFIDENCE_THRESHOLD
    model.iou = NMS_IOU_THRESHOLD

    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        raise RuntimeError(f"Could not open video file: {video_path}")

    fps = cap.get(cv2.CAP_PROP_FPS)
    frame_count = 0

    with open(CSV_FILE, mode="w", newline="") as f:
        writer = csv.writer(f)
        writer.writerow(["time_seconds", "bird_count"])

        while True:
            ok, frame = cap.read()
            if not ok:
                break

            results = model(frame, size=640)
            df = results.pandas().xyxy[0]
            bird_df = df[(df["name"] == "bird") & (df["confidence"] >= CONFIDENCE_THRESHOLD)]
            bird_count = len(bird_df)

            time_sec = frame_count / fps
            writer.writerow([round(time_sec, 2), bird_count])

            display_frame = frame.copy()
            for _, row in bird_df.iterrows():
                x1, y1, x2, y2 = int(row["xmin"]), int(row["ymin"]), int(row["xmax"]), int(row["ymax"])
                conf = float(row["confidence"])
                label = f"bird {conf:.2f}"
                cv2.rectangle(display_frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.putText(display_frame, label, (x1, max(0, y1 - 6)),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 255, 0), 2, cv2.LINE_AA)
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
        cv2.imshow("Bird Tracking", display_frame)
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break

        frame_count += 1

    cap.release()
    cv2.destroyAllWindows()

    data = pd.read_csv(CSV_FILE)
    plt.figure(figsize=(10, 5))
    plt.plot(data["time_seconds"], data["bird_count"], marker="o")
    plt.xlabel("Time (seconds)")
    plt.ylabel("Number of birds")
    plt.title("Birds detected over time")
    plt.grid(True)
    plt.savefig("bird_graph.png")
    plt.show()

def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("--video", type=str, default="input.mp4")
    return parser.parse_args().video

if __name__ == "__main__":
    video_file = parse_args()
    run(video_file)
```

Step 11: Running the Program

Running the Program
Running the Program

To run the code, simply write 'python3 file-name', where file-name is the name of your file. In my case, main.py. The analysis you see should be similar to the image attached.

Main aspects to note:

0.XY - This value, where XY is an integer, is the percentage of confidence that the model has in detecting a person.
Please note that this script will also save the data into a CSV file, which we will use in the post-data collection step.

Step 12: Post Data Collection and Analytics

Post Data Collection and Analytics

If you used the last example code, you'll get a graph showing the time of day against the number of birds detected. This is especially valuable for bird watchers, since many species follow predictable flocking patterns. By studying the graph, watchers can return to the same area at similar times the next day with a strong chance of spotting the same birds again.

Step 13: Next Steps

As of now, my project detects when a bird is present, but I am planning to increase its precision in detecting the type of bird. For example, if the user wants to only be notified when a pigeon comes by, my model should ignore any other types of birds.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

11. Evaluation Criteria:

- Successful camera setup
- Correct execution of AI detection model
- Accurate logging of detected birds
- Ability to interpret collected data

---

12. Languages Available:

English

---

13. Contributors:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

17. Feedback and Continuous Improvement:

Participants are encouraged to improve the model by collecting more bird images, training custom datasets, or adding sensors such as motion detectors.

Submission Notes:

Please ensure that all required documents and resources are attached to your submission. The AI-SAVES@ProjectHub review team will evaluate the submission based on the provided information. Once approved, the project will be published on the hub, and partners will be notified.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.