**Raspberry Pi Machine Learning Wildlife Camera**

This project is based on the MagPi guide [here](https://magpi.raspberrypi.org/articles/wildlife-camera-object-recognition), but I found that guide expected you to know quite a bit already. My guide is an attempt to spell out the MagPi guide in a bit more detail for novices like me. Also, the MagPi guide is a few years old now and I had to correct some of the code to make it work.

**Hardware:**

* [Raspberry Pi 4 model B](https://thepihut.com/collections/raspberry-pi-kits-and-bundles/products/raspberry-pi-starter-kit) (2GB RAM starter set £58)
  + Power supply
  + SD card – at least 16GB
  + Keyboard/mouse/monitor/hdmi cable – not required for operation but required for setup.
  + Raspberry Pi case – I prefer cases with built in fans
* [Raspberry Pi Camera V2](https://thepihut.com/collections/raspberry-pi-camera/products/raspberry-pi-camera-module) (£24)
* [Camera extension cable](https://thepihut.com/collections/raspberry-pi-camera-cables/products/flex-cable-for-raspberry-pi-camera-or-display-24-610mm) (optional) (£2.50)
* [ZeroView camera mount](https://thepihut.com/collections/raspberry-pi-camera/products/zeroview) (optional) (£7)

A picture containing text

Description automatically generatedYou will also want to think about what you want to watch and where the camera is placed. I have a window mounted bird feeder, so I was able to keep the tech inside which saved me from waterproofing it. The [bird feeder cost £9](https://smile.amazon.co.uk/gp/product/B082X4C69X/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1).

I used a RPi 3B+ because that’s what I had at home. Apparently, it’s also possible to use a Pi Zero too. I have one at home, so I’ll have a play around with that and see where we get.

Considering the cost of the RPI 3B+, case, PSU, and SD card; the total project cost for me came in at about £89. I can take about £35 off that if I manage to get it on a Pi Zero in the future.

During the first iteration (pic), I just used a USB power bank to supply power while I tested the code. I bought the 610mm camera cable, I could’ve got away with the 300mm.

If you use a Pi Zero, you can have all your kit nicely mounted on the back of the ZeroView.

**Setup**

Before you start, decide if you want to tweet from your own account or build a new account. I built a new one @WiltshireCam

You’ll also need to apply for a developer account here: apps.twitter.com – it’s very simple, just need to answer a few questions and it’s approved very quickly (a few minutes).

You will need a Google account to access the Application Programming Interfaces (APIs). You can use your own or set up a new one. I used my own.

So, you’ve got your brand-new Raspberry Pi and camera with a fresh Raspian build on it. Follow [this guide](https://thepihut.com/blogs/raspberry-pi-tutorials/16021420-how-to-install-use-the-raspberry-pi-camera) to setup your camera.

If you’re mounting it on a ZeroView like me, [follow this guide](https://cdn.shopify.com/s/files/1/0176/3274/files/ZeroView_User_Guide_1.1.pdf) for installation.

You might also want to drop the monitor and keyboard now so you can carry on configuring your Pi from your laptop. Enable VNC using [this guide](https://www.raspberrypi.org/documentation/remote-access/vnc/). Don’t forget to install it on your laptop/PC/phone too.

**The code**

This section follows the MagPi guide, but I’ve broken it up so it’s easier to read, and I’ve corrected a few errors.

If you didn’t update and upgrade your OS during the camera install, make sure you do it now. Open a terminal window:

sudo apt-get-update

sudo apt-get upgrade

**pi-timolo**

pi-timolo is a picamera application designed for time-lapse photography and motion detection. We will use the motion detection mostly but the time-lapse is useful for setting up your camera.

Install by opening a terminal window and typing (you can just copy this and paste it straight in to a terminal):

cd ~

wget https://raw.github.com/pageauc/pi-timolo/master/source/pi-timolo-install.sh

chmod +x pi-timolo-install.sh

./pi-timolo-install.sh

Once it’s completed, you can test it with:

cd ~ pi-timolo

./pi-timolo.py

This will start the pi-timolo python script and the camera will start logging time-lapse frames at 120 sec intervals in **/home/pi/pi-timolo/media/timelapse** you can change the interval or turn off time-lapse altogether if you wish – I’ll cover that later.

At this point, you should be alerted to any errors such as the camera not being installed correctly, otherwise the script will run and you should see debug info in the Terminal window. Check the pictures by waving your hand in front of the camera, then looking in Pi-timolo > Media Recent > Motion.

You may need to change the image size and orientation of the camera; in the Terminal window, enter:

nano config.py

and edit these variables: imageWidth, imageHeight, and imageRotation. I had to rotate my image by 180 degrees.

While we’re here, if you get a lot of false positives, try changing the motionTrackMinArea and motionTrackTrigLen variables and experiment with the values by increasing to reduce sensitivity. See the [Pi-timolo GitHub repo](https://github.com/pageauc/pi-timolo) for more details. *I haven’t touched this yet. We will only tweet pictures that the Google Vision API labels with “Birds”, so false positives are largely irrelevant and the pi-timolo code deletes old data to create space anyway.*

There’s also going to be some editing of the pi-timolo.py file, so don’t close the Terminal window. Code needs to be added to import some Python libraries (below), and also added [to the function userMotionCodeHere() to check with the Vision API](https://github.com/themagpimag/magpi-issue71/blob/master/WildlifeTrap/listing2.py) (*the code linked here contains errors, I have corrected the code and written it out below*) before tweeting. We can do this now in preparation of setting up our Google and Twitter API. You should still be in the Pi-timolo folder, so type

nano pi-timolo.py

and add the imports at the top of the file with the other imports:

import io

import tweepy

from google.cloud import vision

from google.cloud.vision import types

from google.cloud import storage

Press CTRL+X then Y and RETURN to save.

We’re going to step away from MagPi guide now. Instead of inserting the user code into the pi-timolo.py file, we need to do it in the user\_motion\_code.py file *\*\*You can also edit the .py files in an IDE like Thonny if you prefer. I sometimes find them more visually appealing than nano\*\*:*

nano user\_motion\_code.py

Copy in the following, overwriting everything in there. I’ve corrected the errors that were present in the MagPi code:

"""

This module will be imported into pi-timolo.py and will

execute the userMotionCode function after

motion is detected. The filenamePath will be passed

in case you want to process the file as an attachment

or include in a message, Etc. If you need to import other

python modules they can be added to the top of this

module and used in the userMotionCode.

You can also include other functions within this module

as long as they are directly or indirectly called

within the userMotionCode function since that is

the only function that is called in the pi-timolo.py

program when motion is detected.

For more information see pi-timolo github Wiki

"""

import io

import tweepy

from google.cloud import vision

from google.cloud.vision import types

from google.cloud import storage

#------------------------------------------------------------------------------

def userMotionCode(filenamePath):

"""

Users can put code here that needs to be run

after motion detected and image/video taken

Eg Notify or activate something.

Note all functions and variables will be imported.

pi-timolo.py will execute this function userMotionCode(filename)

in pi-timolo.py per example below

user\_motion\_code.userMotionCode(filename)

"""

# Insert User code Below

print("User Code Executing from userMotionCode function")

print("file path is %s" % filenamePath)

# we need to create an instance of the Google Vision API

client = storage.Client()

# instantiates a client

client = vision.ImageAnnotatorClient()

# loads the image into memory

with io.open(filenamePath, 'rb') as image\_file:

content = image\_file.read()

image = types.Image(content=content)

# performs label detection on the image file

response = client.label\_detection(image=image)

# pass the response into a variable

labels = response.label\_annotations

# we have our labels, now create a string to add to the tweet message

# for debugging - lets see what Google thinks is in the image

print('Labels:')

# add labels to our tweet text

tweetText = "Labels: "

animalInPic = False

for label in labels:

print(label.description)

tweetText = tweetText + " " + label.description

# edit this line to change the animal you want to detect

if "Bird" in tweetText: animalInPic = True

# set up Tweepy

# consumer keys and access tokens, used for authorisation

consumer\_key = 'XXX'

consumer\_secret = 'XXX'

access\_token = 'XXX’

access\_token\_secret = 'XXX'

# authorisation process, using the keys and tokens

auth = tweepy.OAuthHandler(consumer\_key, consumer\_secret)

auth.set\_access\_token(access\_token, access\_token\_secret)

# creation of the actual interface, using authentication

api = tweepy.API(auth)

# send the tweet with photo and message

photo\_path = filenamePath

# only send tweet if it contains a desired animal

if animalInPic:

api.update\_with\_media(photo\_path, status=tweetText)

return

Press CTRL+X then Y and RETURN to save. Next, we’ll set up the APIs.

**Animal detection and tweeting**

We will be using Google Label Detection, which returns a list it associates with the image. First off, you will need to install the Google Cloud Vision libraries on your Raspberry Pi, so type:

pip install --upgrade google-cloud-vision

into your Terminal window. Once finished, run:

pip install google-cloud-storage.

Now you need authorisation, by going to the [Cloud Vision API site](https://cloud.google.com/vision/docs/quickstart) to set up an account. Click on the Manage Resources link and create a new project (you may need to log in or create a Google account).

Go to the API Dashboard and search for and enable the Vision and Storage APIs. Then go to API & Services > Credentials, click on Create Credentials > Service Account Key > New Service Account from the drop-down.

Don’t worry about choosing a Role. Click Create and you’ll be prompted to download a JSON file. You need this as it contains your service account key to allow you to make calls to the API locally. Rename (*I didn’t*) and move the JSON file into your Pi-timolo folder and make a note of the file path.

Next, go back to pi-timolo.py and add the line:

os.environ["GOOGLE\_APPLICATION\_CREDENTIALS"] = "pathtoyour.jsoncredential\_file"

below import os to reference the credentials in your JSON file.

Finally, set up a Twitter account if you haven’t already and install Tweepy by entering:

sudo pip install tweepy

into a Terminal window. Once set up, visit apps.twitter.com and create a new app. Edit app permissions to read and write, then click on Keys and Access Tokens.

Make a note or copy into a notepad or Word.doc the consumer API and secret keys, and the authentication token access token and secret keys.

Go back to edit the user\_motion\_code.py file either in nano or your IDE and paste the relevant keys into the consumer keys and access tokens in place of the ‘XXX’ – I have highlighted the area of code above.

IMPORTANT – if you make any changes to the settings of your Twitter API, you will need to generate new keys and change the code accordingly.

Finally, place your camera in front of your bird feeder and run:

./pi-timolo.py

Any pictures taken of a bird should now be tweeted! If you want to identify a different animal, change the line:

if "Bird" in tweetText: animalInPic = True.

**Fine-tuning**

You can change your time-lapse settings (like turn them off), image dimensions, added text, and motion detection settings by editing the config.py file in the pi-timolo folder. Just open it in your IDE, or nano and play around to get the setup that suits you.

**Testing**

Can’t wait for a bird to pop by? I just searched for a picture of a bird on my phone and held it in front of the camera for a few seconds. Here’s my first tweet:

![Graphical user interface, application, Teams

Description automatically generated]()

As you can see, the Google Vision API generates labels for everything it thinks it can see. If one of those labels is “Bird”, your code will tweet the picture.

**Troubleshooting**

I think I have corrected the code so if you follow mine rather than MagPi’s hopefully yours will work too. That being said, I’ll describe a few of the issues I had and that might give you a steer if you’re having problems.

**Non-ASCII characters.** There seems to be a copy and paste bug around the apostrophe character (‘). I was getting a non-ascii character ' xe2' error. I had to delete and re-write all the apostrophes in the code, even the ones in comments.

NameError: global name 'storage' is not defined - I started testing by just waiving my hand in front to activate the motion detection loops. This should have the effect of taking and image, storing it on Google Storage API and running it through the Vision API to apply labels. On the first test I was getting global name errors on ‘storage’. The MagPi guide only instructed me to import the Google modules into the master pi-timolo.py code – not the user\_motion\_code.py file.

So I double-tapped all the additional import commands into both files and it seemed to fix the issue.

Syntax errors. The MagPi code named the label “bird” but Google calls it “Bird” – case matters. There was also a true/True mix up in the code.

If you simply copy my code, it should work. Although be prepared to change all the apostrophes!

**But what kind of bird is it?**

I’m going to start work on V2.0 which will hopefully build on this and identify species of birds. There are apps out there that can do this so hopefully accessing those APIs wont be too difficult.

Good luck and if you want to try it out and have any questions, please get in touch.

Paul