VIETNAMESE - GERMAN UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Introduction to Information Technology Project

Student : Le Mai Bao Linh - 10423072

HO CHI MINH CITY, 05/ 2024

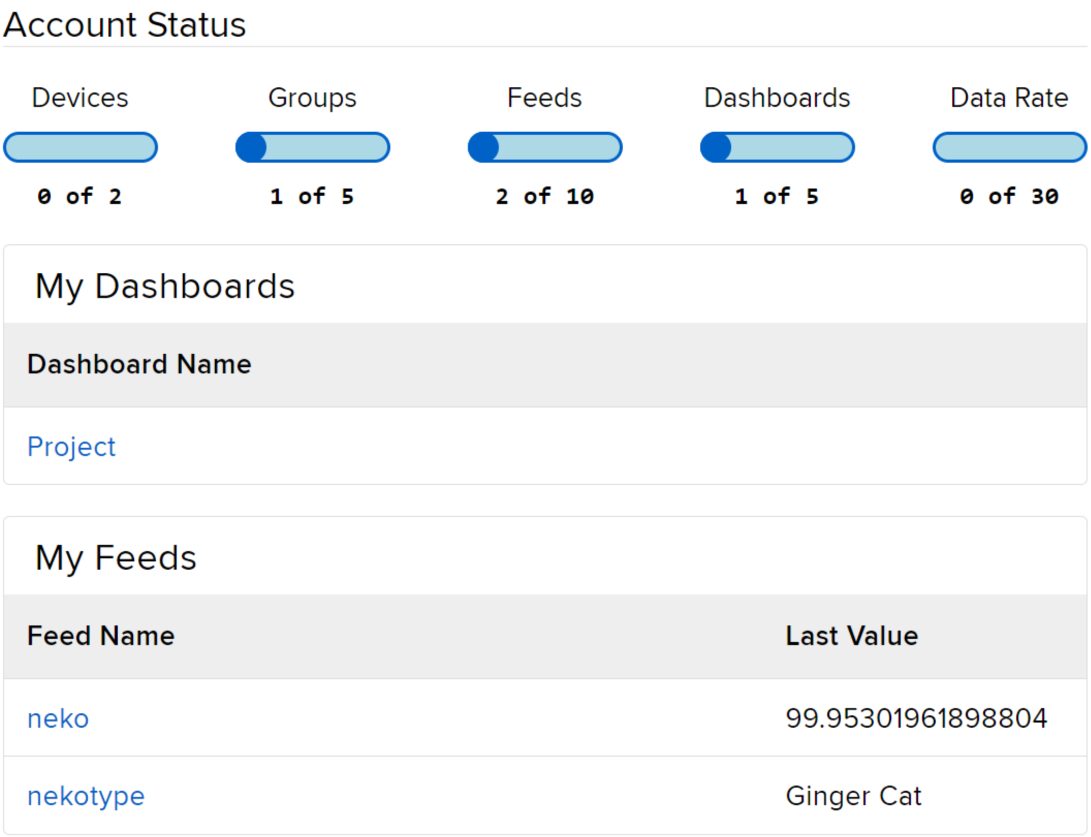
Content

1 Introduction

The project is a part of Information Technology course, aiming to have fundamental knowledge of Artificial Intelligence and the Internet of Things system. It focuses on working with Google Teachable Machine and Adafruit IO to detect the coat colors of cats through images and upload data to an IoT server. This could be applied in animal information management systems for veterinary hospitals or animal shelters.

Google Teachable Machine is a web-based tool developed by Google. It has a user-friendly interface, enabling users to create machine learning models without any data science expertise required. It supports multiple input types, such as images, sounds, and poses, uses them to train AI, and provides real-time feedback. For those features, Google Teachable Machine is a great tool to get used to training AI models.

In this project, the result data from Google Teachable Machine is published to MQTT using Adafuit IO. This is an IOT platform developed by Adafruit Industries, allowing users to display, respond, and interact with the data of a project.



Account Status

Devices	Groups	Feeds	Dashboards	Data Rate
0 of 2	1 of 5	2 of 10	1 of 5	0 of 30

My Dashboards

Dashboard Name
Project

My Feeds

Feed Name	Last Value
neko	99.95301961898804
nekotype	Ginger Cat

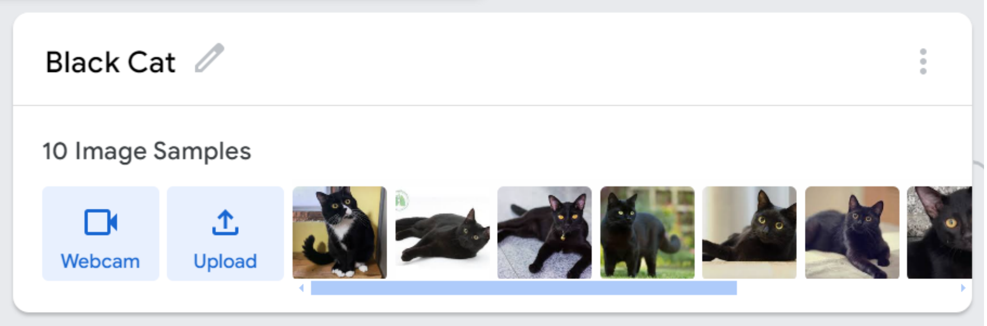
Figure 1: Adafruit Overview page

2 Dataset Collection

Images of cats with 4 different types of coat colors (black, ginger, calico, and white) and backgrounds without cats are collected and uploaded to Google Teachable Machine in 5 classes.

2.1 Black Cat

This class represents black cats with different poses and perspectives.



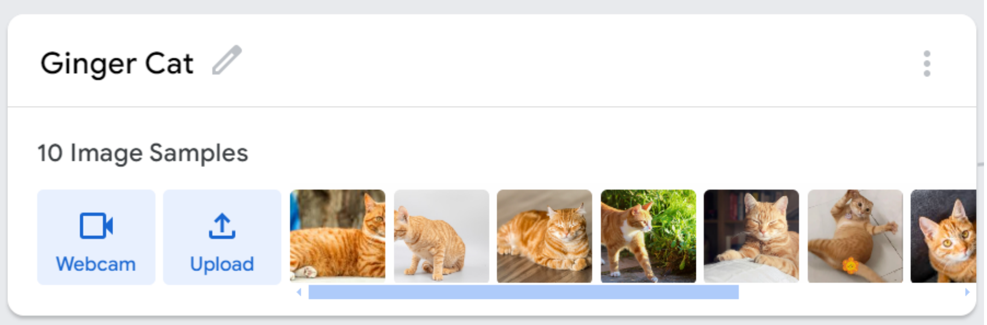
Black Cat

10 Image Samples

Figure 2: Dataset of black cats

2.2 Ginger Cat

A ginger cat is which has orange coat and striped pattern. In this class, ginger cats are represented in different poses and perspectives.



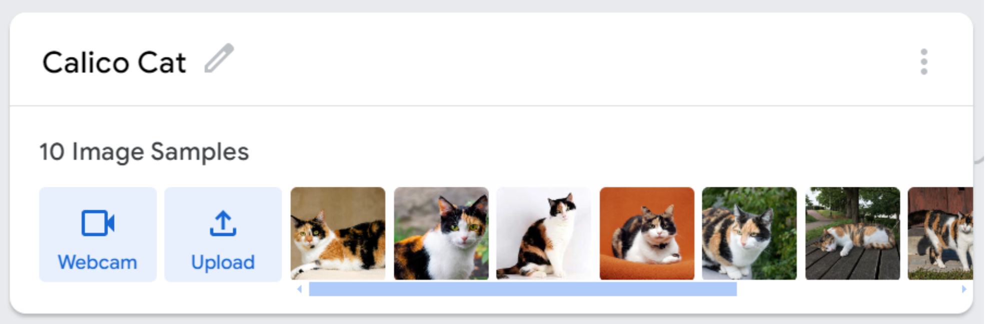
Ginger Cat

10 Image Samples

Figure 3: Dataset of ginger cats

2.3 Calico Cat

A calico cat is which has tri-color coat. They at the most basic are white, black, and orange. This class represents calico cats in different poses and perspectives.

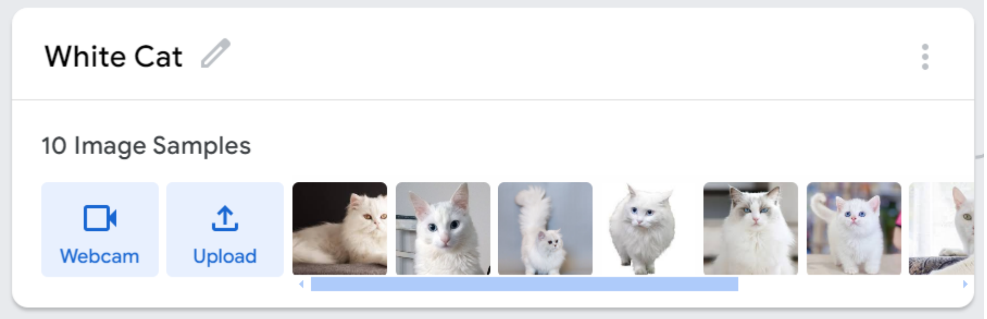

Calico Cat

10 Image Samples

Figure 4: Dataset of calico cats

2.4 White Cat

This class represents white cats in different poses and perspectives.

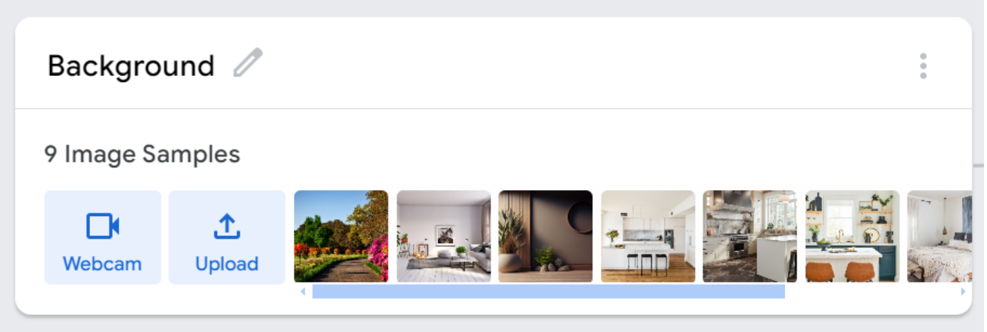

White Cat

10 Image Samples

Figure 5: Dataset of white cats

2.5 Class Background

This class is the images for backgrounds, in which cats do not appear.


Background

9 Image Samples

Figure 6: Dataset of backgrounds

3 Result

After the data is processed, the results will be uploaded to the Adafruit IO Dashboard.

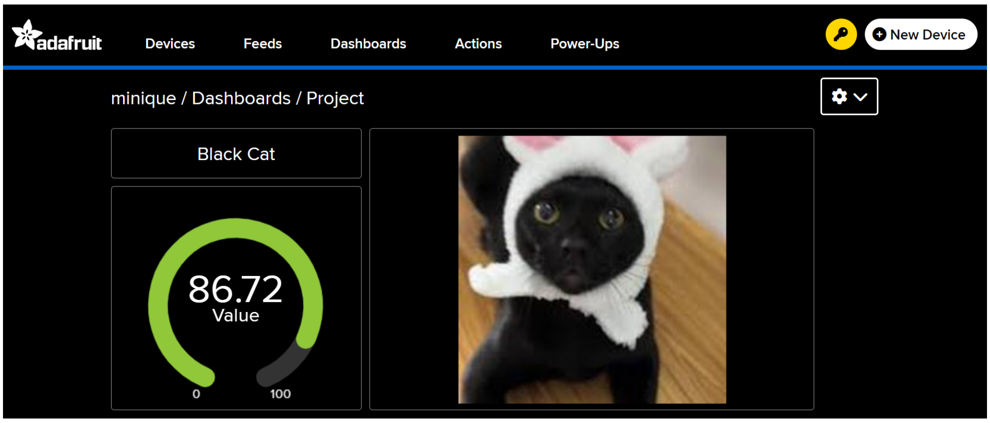


Figure 7: Example of testing black cat

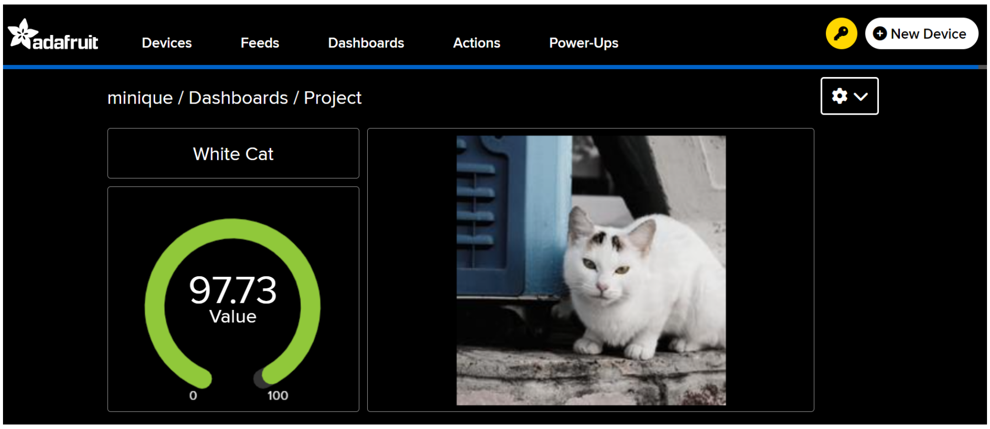


Figure 8: Example of testing white cat

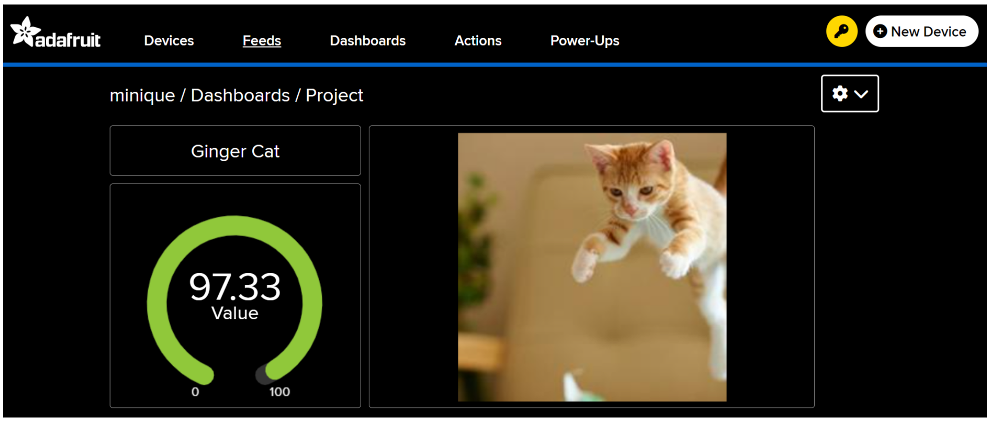


Figure 9: Example of testing ginger cat

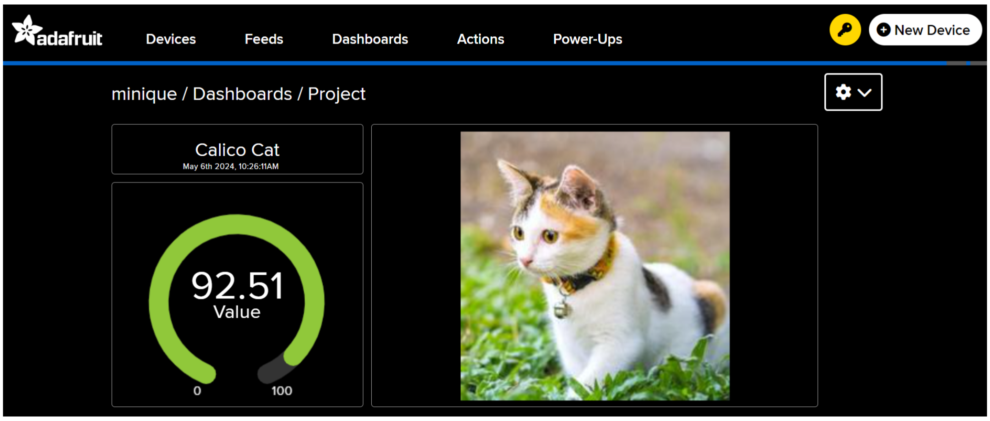


Figure 10: Example of testing calico cat

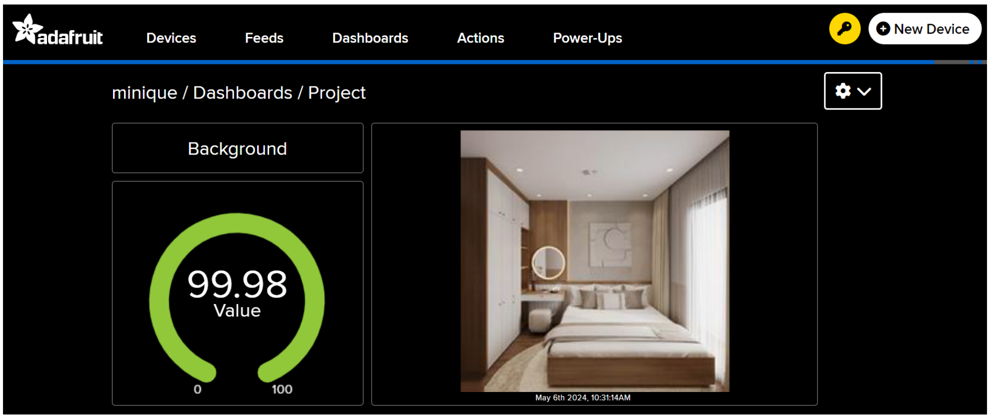


Figure 11: Example of testing background

4 Python Source Code

Listing 1: Example of your Python code

```python
from keras.models import load_model  # TensorFlow is required for Keras to work
from PIL import Image, ImageOps  # Install pillow instead of PIL
import numpy as np
import random
import time
import  sys
from  Adafruit_IO import  MQTTClient
import io
import base64

AIO_FEED_ID = ""
```

```
AIO_USERNAME = "minique"
AIO_KEY = "aio_Hydc43vpIiNzzzlFArPxFNdQ2iwB"

def connected(client):
    print("Connected to the AIO server!!!!")
    client.subscribe(AIO_FEED_ID)

def subscribe(client , userdata , mid , granted_qos):
    print("Subscribed to TOPIC!!!")

def disconnected(client):
    print("Disconnected from the AIO server!!!")
    sys.exit (1)

def message(client , feed_id , payload):
    print("Received: " + payload)

client = MQTTClient(AIO_USERNAME , AIO_KEY)
client.on_connect = connected
client.on_disconnect = disconnected
client.on_message = message
client.on_subscribe = subscribe
client.connect()
client.loop_background()

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = load_model(r"D:\VGU\FY\HK2\IT\Project1\source\converted_keras\↩
    keras_model.h5", compile=False)

# Load the labels
class_names = open(r"D:\VGU\FY\HK2\IT\Project1\source\converted_keras\↩
    labels.txt", "r").readlines()

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image
image = Image.open(r"D:\VGU\FY\HK2\IT\Project1\Yellow\11.jpg").convert↩
    ("RGB")

```

```
# resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)

# turn the image into a numpy array
image_array = np.asarray(image)

# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1

# Load the image into the array
data[0] = normalized_image_array

# Predicts the model
prediction = model.predict(data)
index = np.argmax(prediction)
class_name = class_names[index]
confidence_score = prediction[0][index]

# Convert image to base64
stream = io.BytesIO();
image.save(stream, format="JPEG")
image_uploaded = base64.b64encode(stream.getvalue())

# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)

# Use it in your AI
client.publish("image", image_uploaded)
client.publish("nekotype", class_name[2:])
client.publish("neko", float(confidence_score)*100)
```

5 Extra Features

5.1 Image uploaded to Adafruit IO

Besides the name of the cat and its confidence score, the tested image will be uploaded to the Adafruit IO Dashboard. Data visualized properly enables users to work effectively by providing them with the ability to gain profound insights and make informed decisions.

To add this feature to the project, io and base64 library are used to create an in-memory buffer and convert images to base64 to upload them to Adafruit IO.

Listing 2: Example of your Python code

```
import io
import base64

# Convert image to base64
stream = io.BytesIO();
image.save(stream, format="JPEG")
image_uploaded = base64.b64encode(stream.getvalue())
```

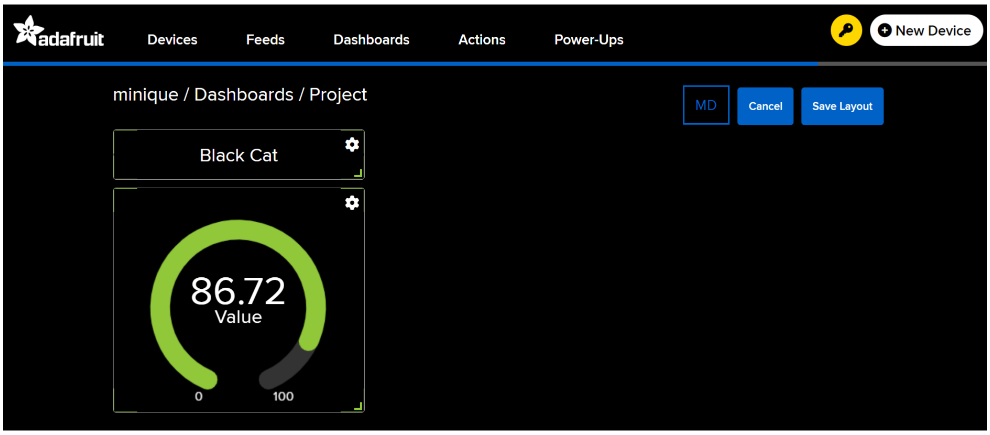


Figure 12: Example of data visualized on Dashboard without Image

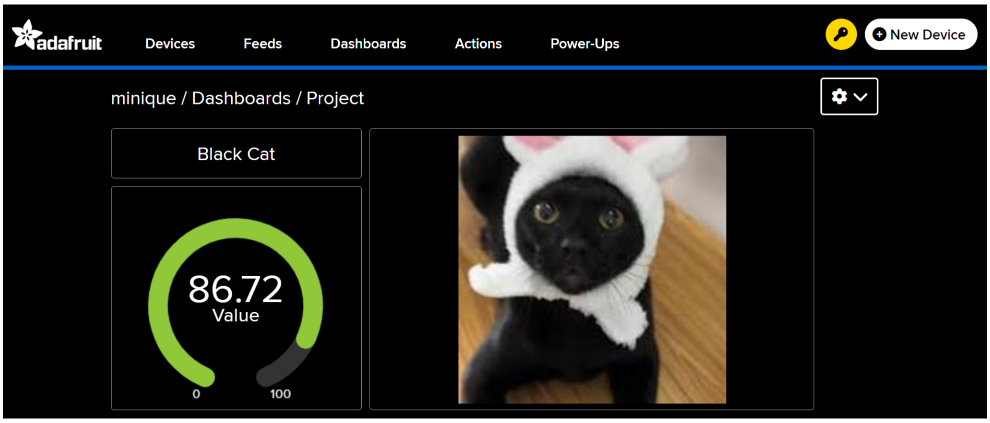


Figure 13: Example of data visualized on Dashboard with image

6 Conclusion

In the course of this project, we learned how to train AI with Google Teachable Machine and connect with the Adafruit IO server to visualize data received from connected devices, providing the foundation for expanding the knowledge of machine learning, framework, and IoT systems. In the rapidly developing era of AI and IoT, the practical experience with these technologies gained from the project is highly valuable for our future education and career, opening up opportunities in various industries.