# Raspberry Pi 5 code explanation

```python
import os
import time
import json
import socket
import numpy as np
import logging
from datetime import datetime
from gpiozero import Button
from picamera import PiCamera
import tflite_runtime.interpreter as tflite
from requests import post, get, ConnectionError
```

These import all the necessary libraries:

- **os, time, json** → file and time handling
- **numpy** → for image data processing
- **logging** → for log output
- **gpiozero** → easy GPIO control (Button)
- **picamera** → to control the Raspberry Pi Camera
- **tflite_runtime** → lightweight TensorFlow Lite interpreter
- **requests** → for HTTP requests

```python
# --- CONFIGURATION ---
BUTTON_PIN = 17
MODEL_PATH = 'model.tflite'
SERVER_URL = 'http://your_server_address/api/upload'
IMAGE_DIR = '/home/pi/images/'
SD_CARD_PATH = '/media/sdcard/'
LOG_DIR = '/home/pi/logs/'
CLASS_LABELS = ['Fine sand', 'Medium sand', 'Coarse sand', 'Granule']
```

These define the:

- GPIO pin for button
- Model path
- Server endpoint for upload
- Folder paths for image/log/data storage
- List of class labels that match your model output

```python
# --- LOGGING SETUP ---
os.makedirs(LOG_DIR, exist_ok=True)
logging.basicConfig(
    level=logging.INFO,
    format="%(asctime)s [%(levelname)s] %(message)s",
    handlers=[
        logging.FileHandler(os.path.join(LOG_DIR, "system.log")),
        logging.StreamHandler()
    ]
)
logger = logging.getLogger(__name__)
```

This:

- Creates `/home/pi/logs/` if not already there.
- Writes all logs to `system.log`.
- Also prints logs live to console if SSH'd in.

```python
# --- HARDWARE INITIALIZATION ---
logger.info("Initializing hardware...")

try:
    os.makedirs(IMAGE_DIR, exist_ok=True)
    os.makedirs(SD_CARD_PATH, exist_ok=True)

    capture_button = Button(BUTTON_PIN)
    camera = PiCamera()
    camera.resolution = (224, 224)
    logger.info("✅ Camera and button initialized.")
except Exception as e:
    camera = None
    capture_button = None
    logger.error(f"⚠️ Hardware initialization issue: {e}")
```

- Creates folders for images and SD card data
- Initializes a button at GPIO 17
- Initializes PiCamera at resolution **224×224** (to match model)
- If anything fails (e.g. camera not found), sets `camera=None` so later code doesn't crash.

```python
# --- MODEL INITIALIZATION ---
try:
    interpreter = tflite.Interpreter(model_path=MODEL_PATH)
    interpreter.allocate_tensors()
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()
    logger.info("✅ TensorFlow Lite model loaded successfully.")
except Exception as e:
    interpreter = None
    logger.error(f"❌ Failed to load TensorFlow Lite model: {e}")
```

This loads your `model.tflite` file and prepares it for inference.
If it fails, logs an error and skips classification later.

```python
# --- UTILITY FUNCTIONS ---
> def get_gps_coordinates():…

> def check_internet():…

> def process_image(image_path):…

> def send_to_server(payload):…

> def store_data_locally(payload):…
```

- get_gps_coordinates():Tries to read latitude/longitude from a GPS module via `gpsd`.
- check_internet():Simple test — if it can fetch Google, internet is available.
- process_image(image_path):Loads the saved image, preprocesses for model.
- send_to_server(payload):Sends JSON payload to your `SERVER_URL`.
- `store_data_locally(payload)`:Saves data to `/media/sdcard/data_YYYYMMDD_HHMMSS.json`.

```python
# --- MAIN LOGIC ---
> def on_button_press():…
```

Main Logic (Button Press):

When button is pressed:

1. **Capture image** with timestamp name.
2. **Fetch GPS** coordinates.
3. **Run model** on captured image.

4. **Prepare payload** dict.
5. **Check internet**.

```python
# --- ENTRY POINT ---
if __name__ == "__main__":
    logger.info("✅ System ready. Press the button to capture data.")
    try:
        if capture_button:
            capture_button.when_pressed = on_button_press
            from signal import pause
            pause()
        else:
            logger.error("⚠️ No button detected. Exiting.")
    except KeyboardInterrupt:
        logger.info("🛑 Program terminated by user.")
    finally:
        if camera:
            camera.close()
```

This block:

- Waits indefinitely for button press (`pause()` keeps script alive).
- Executes `on_button_press()` each time.
- Closes camera safely on exit.