IoT Cat Toy System: Code Overview Sheet

Project Summary: This project is an interactive IoT cat toy system powered by three ESP32s: Base, Sensor, and Trigger. The Sensor ESP uses an ultrasonic sensor to detect paw movement and a photoresistor (LDR) to check ambient lighting conditions. When the light is low and a paw is detected, it signals the Base ESP. The Base ESP then activates a DC motor to rotate the turret and a servo to move a laser pointer vertically. The Trigger ESP contains a manual override button that only activates if the photoresistor detects a low-light environment.

Main Components & GPIO Mapping:

Component	Description	GPIO Pin(s)	ESP Role
Ultrasonic Sensor	Detects paw movement	Trig: GPIO32 Echo: GPIO14	Sensor
DC Motor + Encoder	Rotates turret base	IN1: GPIO25, IN2: GPIO26	Base
	Position feedback via encoder	ENC_A: GPIO33, ENC_B: GPIO27	Base
Servo Motor	Moves laser up/down	Signal: GPIO12	Base
LED	System status indicator	GPIO13	Base
Photoresistor (LDR)	Senses ambient light level (voltage divider)	A0: GPIO36	Sensor
Button	Manual trigger input	GPIO5 (example)	Trigger

Mosfet STP16NF06L and Mosfet bs170

Sender Code Explanation

```
from machine import ADC, Pin, time_pulse_us
from time import sleep_ms
import network
import espnow
```

- This sets up modules for reading sensor input, setting up network capabilities, communicating with ESP-NOW protocol, and handling timing.

```
# ———————— GPIO SETUP ————————
TRIG = Pin(32, Pin.OUT, value=0)
ECHO = Pin(14, Pin.IN)
ldr = ADC(Pin(34))
```

- TRIG is the variable used to trigger the ultrasonic sensor
- ECHO is the variable that receives the echo signal from ultrasonic sensor
- Ldr is the analoug input pin connected to the light dependent resistor (LDR)

```
# ———————— LDR SETUP ————————
ldr.atten(ADC.ATTN_11DB)
LIGHT_THRESHOLD = 1000
```

- This basically sets a threshold to determine whether it's "light" or "dark".

```
# ———————— ESP-NOW SETUP ————————
sta = network.WLAN(network.STA_IF)
sta.active(True)
sta.disconnect()

e = espnow.ESPNow()
e.active(True)


peer = b'\x10\x06\x1c\x0c\xe1\x44' # reciever MAC Addresss
e.add_peer(peer)
```

- This code initializes the Wi-Fi interface that prepares the ultrasonic sensor and LDR to sense distance and light.
- Sets up ESP-NOW protocol to send data to another ESP32.

```
# ———————— DISTANCE FUNCTION ————————
def distance_cm():
    TRIG.off(); sleep_ms(2)
    TRIG.on();  sleep_ms(10)
    TRIG.off()
    try:
        us = time_pulse_us(ECHO, 1, 30_000)
        return round((us / 2) / 29.1, 2)
    except OSError:
```

```
        return -1
```
- Sends a 10ms trigger pulse to the ultrasonic sensor
- Converts time to distance in centimeters.

```
# ———————— DISTANCE FILTERING ————————
def stable_distance():
    readings = []
    for _ in range(5):
        d = distance_cm()
        if 0 < d < 100:
            readings.append(d)
        sleep_ms(10)
    if readings:
        readings.sort()
        return readings[len(readings) // 2]
    else:
        return -1
```
- Takes 5 distance reading and returns the median value if the readings are valid.

```
# ———————— PAW DETECTION ————————
def paw_detected():
    print("Paw Detected!")
    e.send(peer, "PAW")
```
-

```
# ———————— WAIT FOR LIGHTS OFF ————————
while True:
    light = ldr.read()
    print(f"LDR value: {light}", end='  ')
    if light > LIGHT_THRESHOLD:
        print("Lights are ON — turn off lights to start play.")
    else:
        print("Lights OFF — starting play!")
        break
    sleep_ms(500)


# ———————— MAIN LOOP ————————

while True:
    # Wait for lights OFF to start play
    while True:
        light = ldr.read()
        print(f"LDR value: {light}", end='  ')
```

```
    if light > LIGHT_THRESHOLD:
        print("Lights are ON — waiting...")
    else:
        print("Lights OFF — starting play!")
        break
    sleep_ms(500)

# Set new ultrasonic baseline before play begins
baseline_cm = stable_distance()
print(f"Baseline distance: {baseline_cm:.2f} cm")


while True:
    # Check if lights turned back on mid-play
    light = ldr.read()
    if light > LIGHT_THRESHOLD:
        print("Lights turned ON — stopping play.")
        break  # Exit play mode

    d = stable_distance()
    if d == -1:
        print("Sensor error or out of range")
        continue

    delta = baseline_cm - d
    print(f"dist={d:.2f} cm  Δ={delta:.2f} cm")

    if delta > 5:
        confirm = 0
        for _ in range(8):
            dd = stable_distance()
            if dd != -1 and (baseline_cm - dd > 5):
                confirm += 1
            sleep_ms(50)
        if confirm >= 6:
            paw_detected()
            sleep_ms(2000)
        else:
            print("False alarm")
            sleep_ms(200)
    else:
        baseline_cm = 0.9 * baseline_cm + 0.1 * d

    sleep_ms(50)
```