自動分別自動開閉ゴミ箱の製作

藤井　紬生　谷　奏多
久山　幸起　石原　総二郎

１．　研究概要

自動分別自動開閉ゴミ箱を制作することで、高齢者や障がい者にも使いやすい物にし、自動分別機能でリサイクル効率を上げ、自動開閉機能では、非接触で衛生的なゴミ箱を目指す。また、情報技術科で学習したセンサーやAIの技術を使用し、技術力を向上させることを目的として行った。

２．ゴミ箱の構成

今回製作したゴミ箱は、カメラで読み取った画像をAIで判定して､ペットボトルと缶と燃えるゴミに分別する。ラズベリーパイで、制御を行う。(図１)

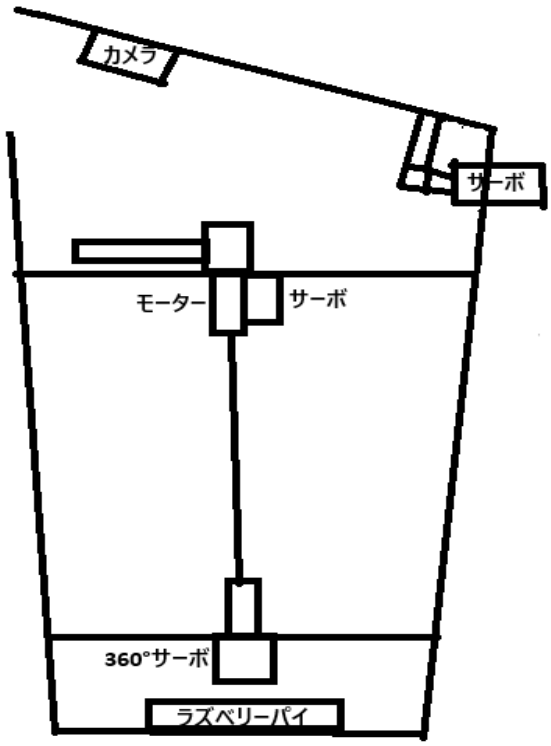

図１　ゴミ箱設計図

３．製作スケジュール

表１　製作スケジュール

４月	テーマ決定
５月	ゴミ箱の設計書の制作
６月	AIの制作
７月	AIを使った分別プログラムの制作
８月	AIの精度、プログラムの改良
９月	AIの精度、プログラムの改良
１０月	自動開閉機能の制作
１１月	ゴミ箱の機構の製作
１２月	分別機能の制作

４．必要部品

（１）自動分別機能

○AI認識（YOLO）側

・PC または Raspberry Pi 4/5
・USBカメラ（Webカメラ）
　YOLO がリアルタイム動作できる画質(720p以上推奨)

○サーボ制御側

・FEETECH 360° サーボ
・FEETECH HLS3606M
・サーボ通信アダプタ
・FE-URT1（USB ⇔ TTL/RS485 シリアル変換アダプタ）
・ サーボ用ケーブル
・FEETECH 5264-3P サーボケーブル

○電源

・6V 2A DC電源アダプタ（サーボ用）
　最低限の配線・部材
・ジャンパー線（オス-メス / メス-メス）

・M3 ビス・ナット（サーボ固定用）
・ブレッドボード

○回転仕切り構造
・回転ディスク（アクリル・プラ板・3D プリント）
・仕分けシュート 3 個（3 方向）
・サーボホーン用ネジ
・サーボ固定ブラケット（アルミ/3D プリント）

○筐体（外装）
・ゴミ箱本体
・木板 / アクリル板 / 3D プリント製フレーム
・ネジ・蝶番
・サーボ固定用ブラケット

○検知系（開閉トリガー）
・超音波距離センサ（HC-SR04）

○駆動系（フタ開閉）
・サーボモーター（SG90）
・サーボホーン（アーム型）。
・6V 安定化 DC-DC ステップダウンコンバータ

ソフトウェア
○YOLO 側
・Python
・Ultralytics YOLOv8/v9
・OpenCV
・NumPy
・Python 3
・GPIO 制御ライブラリ
・time モジュール

○サーボ制御側
・FEETECH シリアルバスサーボ Python ライブラリ
・自作 Python コントロールコード

５．学習環境・プログラム環境

AI モデルを制作するにあたって、Google colab を学習環境として使用した。

図２　Google colab

Raspberry pi で AI モデルを利用するにあたって、Raspberry pi geany をプログラム環境として使用した。

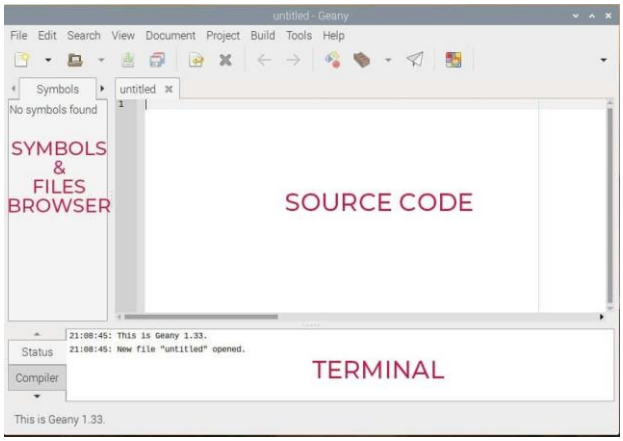

図３　Raspberry pi geany

自動開閉機能を制作するにあたって、Raspberry pi thonny をプログラム環境として使用した。

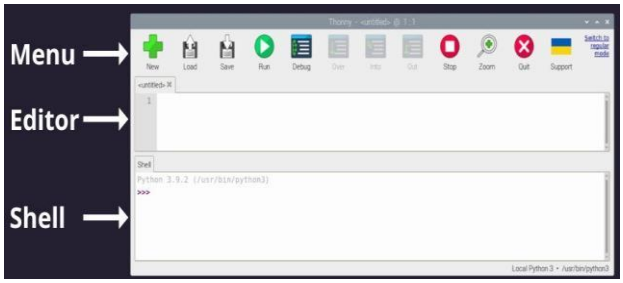

図４　Raspberry pi thonny

６．制作工程

（１）AI の制作

ペットボトルと缶を判別するために、AI を制作した。まず動体認識がどのような原理で動作するのかを調べた。カメラから取得した画像をピクセル単位で解析し、学習データから抽出された特徴と照合することで物体を分類する仕組みであることを理解した。また、物体検出モデルの種類についても調べ、現在幅広く利用されている YOLO 系モデルには、高速処理と軽量性を兼ね備えた YOLOv8 が存在することを知った。特に、Raspberry Pi のような小型コンピュータでも動作可能な「YOLOv8n(ナノモデル)」が適していると判断した。

そこで、缶とペットボトルを自動で識別できる AI を作ることを目標に設定し、まず学習用データセットの準備を行った。

表２　YOLO 判別表

項目	YOLOv8（標準）	YOLOv8n（ナノ）
モデルサイズ	大きい	非常に小さい
パラメータ数	多い	少ない
検出精度	高い	やや低い
推論速度	普通～高速	非常に高速
計算負荷	高い	低い
メモリ使用量	多い	少ない
動作環境	GPU 搭載 PC 向け	Raspberry Pi 等に最適

以下にプログラミングの手順を示す。

手順１：YOLOv8 を使う準備

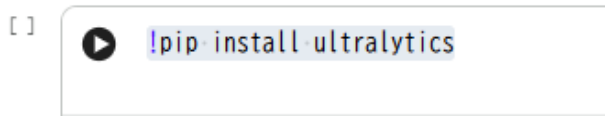

図５　YOLOv8 を使う準備

物体を自動で判別するための手法として、近年多く利用されている物体検出モデル YOLOv8 を使用することにした。YOLOv8 は高速かつ高精度に物体検出が可能であり、リアルタイム処理にも適している。

そこで、Python 環境に Ultralytics ライブラリを導入し、学習および推論を行える環境を構築した。

手順２：学習用データの準備

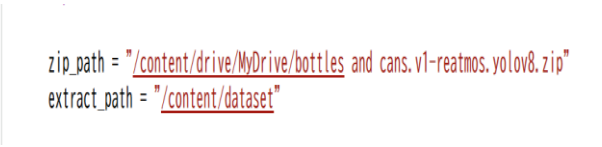

図６　学習用データ準備

ペットボトルと缶を識別する AI を作成するために、それぞれの画像を収集し、合計約 4000 枚の学習用データセットを自作した。

データは zip 形式で保存されていたため、指定したフォルダに展開し、画像データとアノテーション（ラベル）データが正しく含まれていることを確認した。

手順３：データ構成の確認

図７　データ構成の確認

展開後、学習用（train)、検証用（val)、テスト用(test)のフォルダ構成を確認した。

この構成が正しくない場合、学習時にエラーが発生する可能性があるため、画像とラベ

ルファイルが対応するフォルダに配置されているかを慎重に確認した。

手順４：YOLO 用設定ファイルの確認

```
# YAMLファイルを上書き
data = {
    "train": "/content/dataset/train/images",
    "val": "/content/dataset/valid/images",
    "test": "/content/dataset/test/images",
    "nc": 2,
    "names": ["Bottle", "Tin-can"]
}
```

図８　YOLO 用設定ファイルの確認

YOLOv8 がデータセットを正しく読み込むために使用する 設定ファイル（data.yaml）を確認した。このファイルには、学習用画像のパスや検証用画像のパスなどの情報が記述されている。

本研究では、分類対象を 「Bottle（ペットボトル）」と「Tin-can（缶）」の２種類とした。

手順５：AI モデルの選択

```
[ ]
# Colab でファイル編集
!nano /content/dataset/data.yaml
```

図９　AI モデルの選択

YOLOv8 には複数のモデルサイズが存在するが、今回は処理が軽く高速に動作する YOLOv8n を選択した。このモデルは計算量が少なく、CPU 環境や Raspberry Pi のような小型コンピュータでも動作可能である。そのため、実験用途、および実際の装置への組み込みに適したモデルであると判断した。

手順６：学習条件の設定、AI の学習実行

```
model.train(
    data="/content/dataset/data.yaml",
    epochs=3,
    imgsz=320,
    batch=8,
    fraction=0.2,
    cache=True,
    device="cpu"   # ← GPU ではなく CPU を明示
)
```

図 10　学習条件の設定

学習条件として、まず学習回数（エポック数）を 10 回に設定し試したが、時間がかかり学習が終わらず、５回も試したが、時間がかかり学習が終わらなかった。最終的に学習回数を３回に設定することで、モデルが正しく動作するかどうかを確認することができた。

また、画像サイズを 320×320 に設定することで処理負荷を軽減し、学習および推論の高速化を図った。

設定した条件をもとに、YOLOv8 を用いて AI の学習を実行した。

AI は、画像と正解ラベルを照合しながら、ペットボトルと缶の特徴を学習していった。

手順７：学習済みモデルの保存と活用

完成した学習済みモデルをパソコンに保存し、カメラ映像からペットボトルと缶をリアルタイムで認識できるようにした。

このモデルを利用することで、自動分別ゴミ箱の制御システムへ応用できる状態となり、AI とハードウェアを組み合わせた実践的なシステム構築が可能となった。

（２）Raspberry Pi で AI をつかう

手順１：AI 実行環境構築

○Python を最新情報に更新する

```
sudo apt install python3-pip -y
```

○OpenCV のインストール

```
sudo apt update
sudo apt install -y python3-opencv
確認
python3 - << 'EOF'
import cv2
print("OpenCVversion:",cv2._version_)
EOF
```

○カメラを有効化

```
sudo raspi-config
```

○YOLO をインストール

```
pip3 install ultralytics
```

手順２：AI をダウンロード

Pi の Chrome で Google Drive を開いて best.pt をダウンロード。

手順３：ペットボトル、缶、燃えるゴミを判定するプログラムを制作

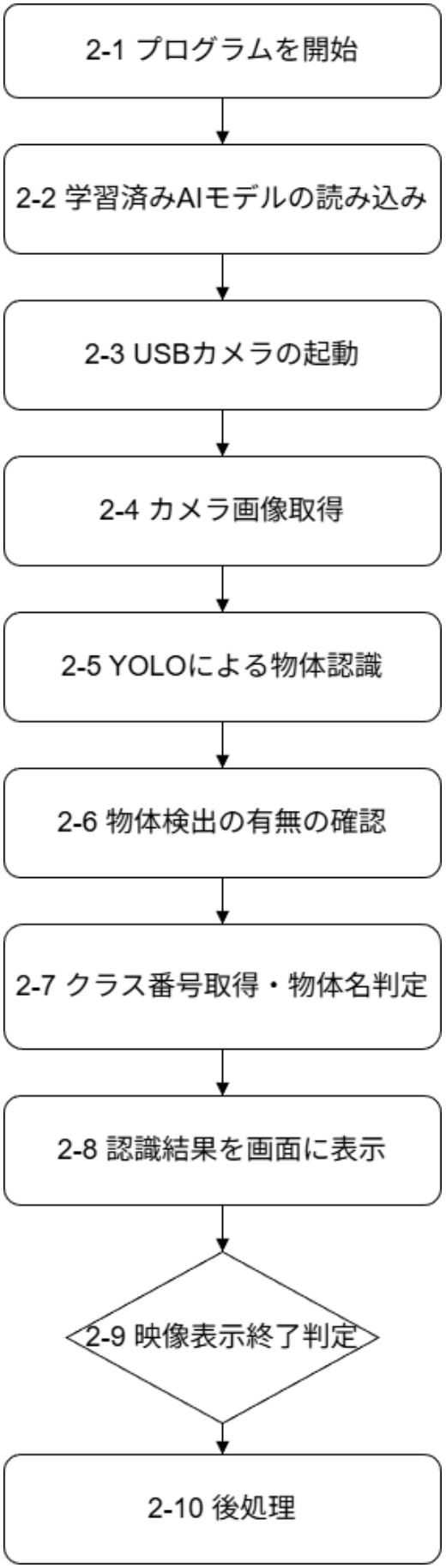

図 11　フローチャート

２－２：学習済み AI モデルの読み込み

- 事前に学習させて作った AI のモデルファイル「best.pt」を読み込んだ。
- このモデルは、ペットボトル（Bottle）と缶（Tin-can）を識別できる物体認識 AI である。
- これにより、カメラ画像を使った判定が可能になった。

２－５：YOLO による物体認識の実行

- 取得した画像を YOLO に渡して推論を行った。
- 画像サイズを 640×640 を 320×320 に縮小することで面積（＝計算量）は ４分の１になり、処理が４倍速くなる。
- Raspberry Pi のため、推論は CPU を使用した。
- 不要なログ表示を省き、動作を軽くした。

２－６：検出結果の確認

- 物体が検出された場合のみ、分類処理を行った。
- 検出された物体のクラス番号を取得した。
- クラス番号から、物体の名前を判定した。

２－７：ゴミの種類を判定

- 検出結果が「bottle」や「pet」の場合は「PET」と判断した。
- 検出結果が「can」の場合は「CAN」と判断した。
- 最初に検出された物体を基準にし、ラベルを確定させた。

判定結果

燃えるゴミ

図 12　燃えるゴミ判定結果

ペットボトル

図 13　ペットボトル判定結果

缶

図 14 缶判定結果

（３）自動開閉機能の制作

①自動開閉機能の処理

- ゴミ箱のフタを手をかざすだけで自動開閉させる。
- 触れずに動作するため衛生的である。
- センサー検知後、一定角度だけサーボを回転させて開閉する。

②配線

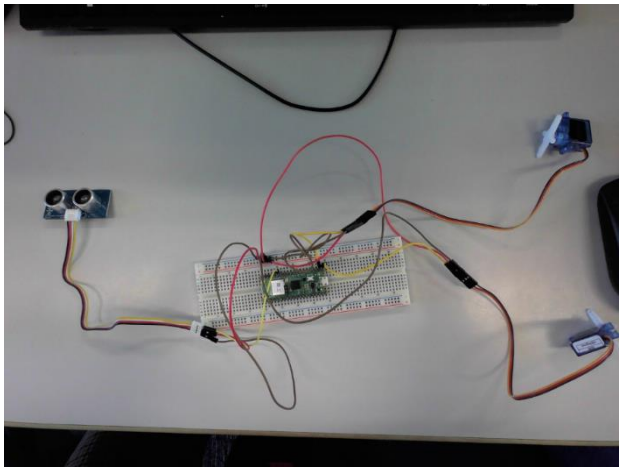

図 15　自動開閉機能の配線

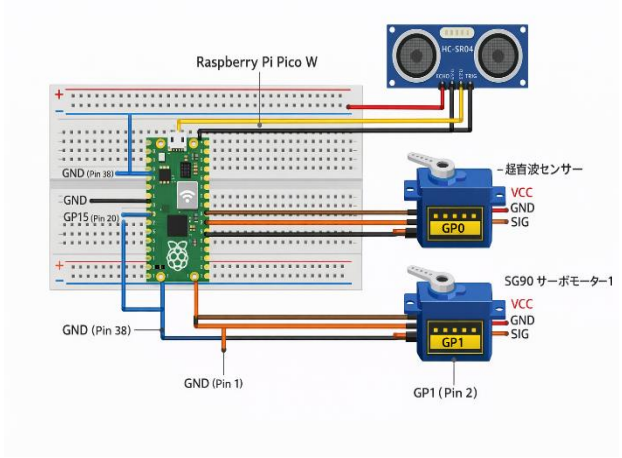

図 16　自動開閉機能の配線図

表３　接続表

デバイス	ピン/ワイヤー色	Pico W 接続先	ブレッドボード接続
HC-SR04	VCC (赤)	–	赤い電源レール (+5V)
HC-SR04	GND (黒)	–	青色の GNDレール
HC-SR04	TRIG (緑)	GP14 (Pin 19)	–
HC-SR04	ECHO (黄)	GP15 (Pin 20)	–
SG90 サーボ1	VCC (赤)	–	赤い電源レール (+5V)
SG90 サーボ1	GND (茶/黒)	–	青色の GNDレール
SG90 サーボ1	SIG (黄/オレンジ)	GP0 (Pin 1)	–
SG90 サーボ2	VCC (赤)	–	赤い電源レール (+5V)
SG90 サーボ2	GND (茶/黒)	–	青色の GNDレール
SG90 サーボ2	SIG (黄/オレンジ)	GP1 (Pin 2)	–

③処理の流れ

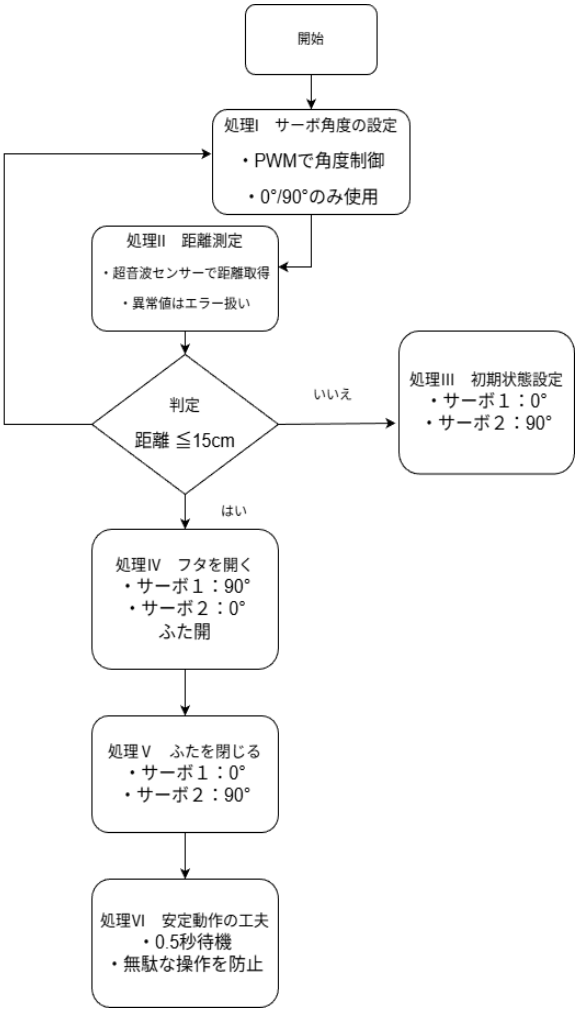

図 17　フローチャート

処理 I ：サーボ角度の設定方法

```
16
17  # --- 定数設定（変更なし） ---
18  DISTANCE_THRESHOLD = 15
19  DUTY_MIN = 2500
20  DUTY_MAX = 7500
21
22  ANGLE_0 = DUTY_MIN
23  ANGLE_90 = 5000
24
```

図 18　サーボ角度

- サーボは PWM（パルス幅）によって角度を制御する仕組みである。
- 0°　の位置はデューティ比　2500　に対応させた。
- 90°　の位置はデューティ比　5000　に対応させた。
- 角度を指定すると、それに対応したデューティ比がサーボに送られるようにした。
- 今回は 0°　と 90°　の 2 種類のみを使用した。

処理 II ：距離測定の仕組み

- 超音波センサーを使って距離を測定した。
- Trig ピンを 10 マイクロ秒だけ High にして超音波を発射した。
- 音波が物体に当たって戻ってくると Echo ピンが High になった。
- Echo　が　High　になっている時間を計測した。
- 計測した時間を使い、音速　343m/s　をもとに距離を計算した。
- 異常な値（0cm 未満や 400cm 超え）はエラーとして扱った。

処理 III ：初期状態の動作

- プログラム開始時、サーボ 1 は 0°　に設定した。
- サーボ 2 は 90°　に設定した。
- 2 つのサーボが逆向きになることで、フタは閉じた状態になった。

処理 IV ：手が近づいたときの動作

```
83  while True:
84      distance = measure_distance()
85
86      if distance > 0:
87          print(f"距離: {distance:.2f} cm")
88
89          if distance <= DISTANCE_THRESHOLD and current_servo1_angle != 90:
90              # 手が近づいた場合（閾値以下）：サーボ1を90度、サーボ2を0度に
91              print("手が近づきました！サーボ1:90度，サーボ2:0度に回転させます。")
92              set_servo_angle(servo1, 90) # 正方向（90度）
93              set_servo_angle(servo2, 0)  # 逆方向（0度）
94              current_servo1_angle = 90
```

図 19　手が近づいたときの動作

- 測定した距離が　15cm　以下になると、手が近づいたと判断した。
- サーボ 1 を 90°　に回転させた。
- サーボ 2 を 0°　に回転させた。
- 2 つのサーボが反対方向に動き、フタが開いた。
- 同じ動作を繰り返さないように、現在の角度を記録した。

処理Ｖ：手が離れたときの動作

```
96          elif distance > (DISTANCE_THRESHOLD + 5) and current_servo1_angle != 0:
97              # 手が遠ざかった場合：サーボ1を0度、サーボ2を90度に
98              print("手が遠ざかりました。サーボ1:0度，サーボ2:90度に戻します。")
99              set_servo_angle(servo1, 0)  # 正方向（0度）
100             set_servo_angle(servo2, 90) # 逆方向（90度）
101             current_servo1_angle = 0
```

図 20　手が離れたときの動作

・ 距離が　20cm　を超えると、手が離れたと判断した。
・ サーボ１を　0°　に戻した。
・ サーボ２を　90°　に戻した。
・ フタは元の位置に戻り、閉じた状態になった。
・ 開く距離と閉じる距離に差をつけることで、誤動作を防いだ。

処理Ⅵ：安定動作の工夫

```
102         else:
103             print("距離測定エラーまたは範囲外です。")
104
105         time.sleep(0.5)
106
```

図 21　安全動作の工夫

・ 距離測定は 0.5 秒ごとに行った。
・ センサーの値が細かく変化しても、サーボが無駄に動かないようにした。
・ エラー時にはメッセージを表示し、安全に動作を続ける構成とした。

（４）ごみ箱の機構製作

①ゴミ箱の中の機構

１２３デザインで製作した。モーターを組み込む機構を底面につけることで、この棒を回してごみを落とす。

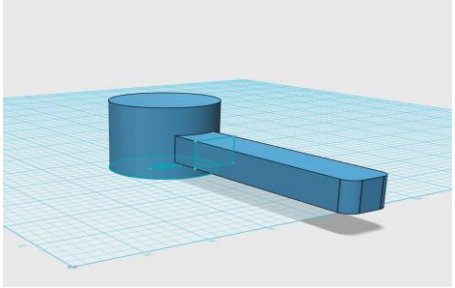

図 22　１２３デザイン

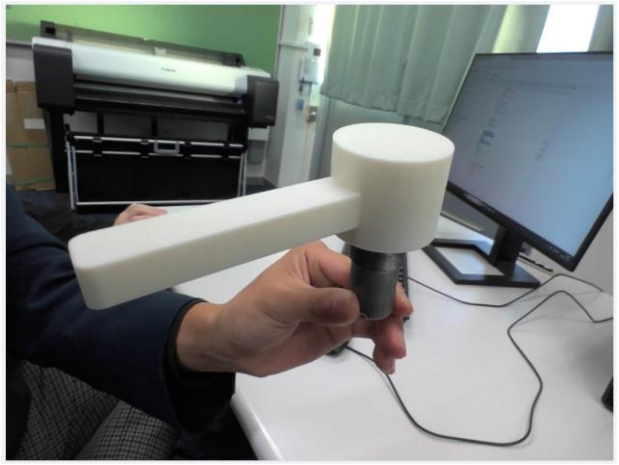

図 23　印刷後

図 24　回転ディスク

回転ディスクを２枚製作した。

それぞれにサーボモーターを取り付け、回転させることで穴の位置が変化し、特定の位置でのみ投入口が現れる仕組みとした。

２枚の回転ディスクを正確に重ねて動作させるため、中心に通す棒を製作した。

この棒により､ディスクの位置ずれを防ぎ、安定した回転が可能となった。

ゴミを分別するための場所を分ける板を製作した。

②サーボモーター取り付け

サーボモーターを取り付ける箱を製作した。

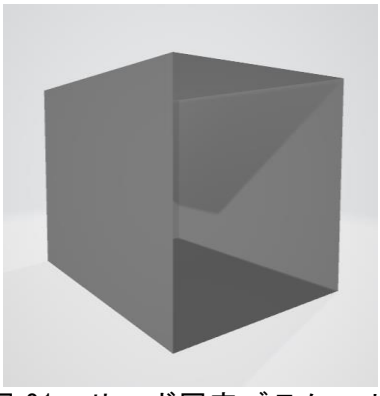

図 21　サーボ固定ブラケット

（５）ゴミ箱の分別プログラム

○システム概要

- YOLO によるリアルタイム物体認識と、360° サーボ（HLS3606M）を組み合わせた自動分別システム。
- サーボの角度制御によって、PET・CAN・燃えるゴミの 3 方向に分別する仕組み。
- 1 軸回転のみで分別できるため、構造がシンプルで高速に動作する。

○分別の仕組み

- YOLO の認識結果をすぐに読み取り、サーボ角度を自動で変更する。
- 分別角度

  0～120° ：燃えるゴミ

  120～240° ：PET ボトル

  240～360° ：アルミ缶
- 複雑な仕切りを使わず、回転動作だけで確実に分別できる。

○サーボ制御の特徴

- HLS3606M は 0～360° の正確な角度制御が可能。
- FE-URT1 を使い、PC や Raspberry Pi から直接デジタル制御できる。
- PWM 制御より安定しており、位置・速度・電流・温度の情報を取得できる。

①プログラム

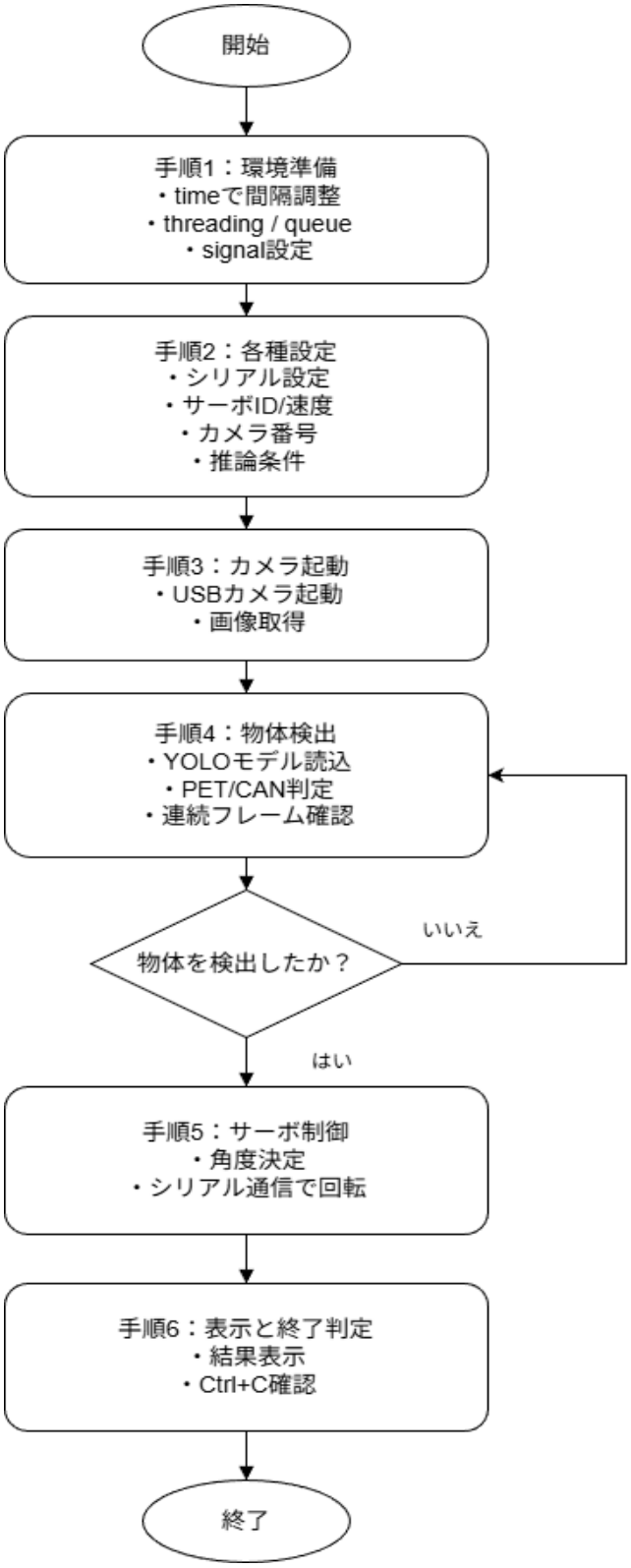

図 22　フローチャート

手順１：環境準備

```
import time
import queue
import threading
import signal
import sys

import cv2
from ultralytics import YOLO
import serial
```

図 23　環境準備

- time を使い、処理間隔を調整した。
- threading・queue を使い、YOLO 処理とサーボ制御を同時に行えるようにした。
- signal を使い、安全にプログラムを終了できるようにした。

手順２：各種設定

```
SERIAL_PORT = "/dev/ttyUSB0"
BAUDRATE = 115200            # HLS3606M は 38400 - 1_000_000bps の幅。115200で試す
SERVO_ID = 1                 # 実際に設定したサーボID
SERVO_SPEED = 200            # 速度パラメータ（サーボ仕様に合わせて調整）

CAM_INDEX = 0                # カメラデバイス
IMG_SIZE = 640               # YOLO の画像サイズ（推論速度と精度のバランス）
SHOW_WINDOW = True           # 推論結果をウィンドウで見るか（True/False）

CONF_THRESHOLD = 0.45        # YOLO 信頼度閾値（0〜1）
CONSECUTIVE_N = 3            # 同一ラベルをNフレーム連続で確認して確定
```

図 24　各種設定

- リアルポートと通信速度を設定し、サーボと接続した。
- サーボ ID と回転速度を指定した。
- 使用するカメラ番号を決めた。
- 推論サイズと信頼度下限を設定し、誤認識を減らした。

手順３：カメラ起動

- USB カメラを起動し、映像を取得した。
- 取得した画像を推論用に使った。

手順４：物体検出

```

MODEL_PATH = "best.pt"          # あなたのモデル

```

図 25　物体検出

- 学習済み YOLO モデルを読み込んだ。
- 画像から PET・CAN を判定した。
- 連続フレームで確認し、誤動作を防いだ。

手順５：サーボ制御

```
# 分別エリアの"中央"角度（各120°セクションの中央を目標に）
# (0-120) burn -> target 60
# (120-240) PET -> target 180
# (240-360) CAN -> target 300
ANGLE_MAP = {
    "burn": 60,
    "pet": 180,
    "can": 300
}

# ラベル名（モデルで使っているラベルに合わせて変更）
LABEL_PET = "PET"
LABEL_CAN = "CAN"
# その他は "burn" 扱い（モデルに 'burn' ラベルがある場合はその文字列に合わせる）

```

図 26　サーボ制御

- 判定結果に応じてサーボ角度を決定した。
- シリアル通信でサーボを回転させた。

手順６：表示と終了

- 認識結果を画面に表示した。
- Ctrl+C で安全に終了するようにした。

７．研究のまとめ

本研究では、物体認識 AI（YOLO）とサーボモーター制御を組み合わせた、自動開閉自動分別ゴミ箱の制作に取り組んだ。Raspberry Pi の環境構築やカメラ設定、サーボの誤作動など多くの技術的課題があり、計画通りには進まなかったが、原因を一つずつ検証し改善を行った。

超音波センサーによる自動開閉機能では、距離測定のばらつきや誤動作が発生し、設置方法や閾値設定の重要性を学んだ。距離測定から判定、サーボ動作までの流れを整理することで、安定した自動開閉動作を実現できたと感じた。

AI 制作では、データ量や学習回数を抑えたことで短時間で動作確認ができた点は良かったが、認識精度や評価の正確さには課題が残った。今後はデータ数や学習回数を増やし、GPU を活用することで、より高精度なモデルに改善できると考えた。

Raspberry Pi 上での AI 実装では、YOLO と OpenCV を用いた物体認識をシンプルな構成

で実現でき、処理の流れを理解しながら実験できた。一方で、処理速度や判定の安定性に限界があり、画像サイズや判定方法の工夫、例外処理の強化が必要であると感じた。

ゴミ箱の機構製作では、柱を軸としたシンプルな構造で仕組みが分かりやすく、3D プリントしやすい設計であったが、強度やサーボ固定方法など実用面での改善点も確認した。一方で、サーボ到達確認、発熱や電源の安定性など､実用化に向けた課題も明確になった。

本研究を通して、AI・電子回路・機構設計を総合的に考える重要性を理解した。今後は精度、安定性、耐久性をさらに向上させ、より実用的な自動開閉自動分別ゴミ箱の完成を目指したいと考えた。

８．参考文献

python でできることを紹介する初心者向け学習入門サイト python アカデミー

ラズパイ(Raspberry Pi)への OpenCV のインストール方法・手順

Zenn

生成 AI×Colab で始める Python 学習 - デジタルネイティブ世代に追いつくための新しい学び方

電気設計人.com

【ラズパイ電子工作】サーボモータを動かす方法(角度指定) | 電気設計人.com

Sozorablog

【初心者向け】ラズベリーパイで超音波距離センサー HC-SR04 を動かしてみよう | sozorablog

Pythonjapan

ゼロからの Python 入門講座 - python.jp