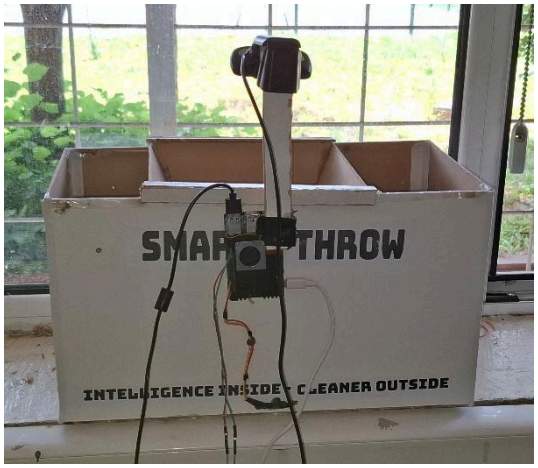
Паметна канта – Интелегенција одвнатре, почисто однадвор

Паметна канта е корпа за отпадоци што работи автоматски, препознавајќи го видот на отпад што се фрла во неа и го сортира во соодветниот оддел. Со овој проект, нашата цел е да го намалиме загадувањето и да го направиме процесот на рециклирање полесен и поефикасен. Кога корисникот фрла отпад во корпата, интегрирана камера го скенира предметот и користејќи претходно обучена база на податоци, го идентификува видот на отпад. Потоа, механизам за движење го пренесува отпадот во соодветниот оддел и го ослободува, по што се враќа во почетната положба.





ЦЕЛИ

• Зголемување на практичното знаење за вештачка интелигенција, компјутерска визија и IoT системи преку практична работа.

• Подобрување на соработката и тимската работа преку заедничка работа низ хардверски, софтверски и дизајнерски улоги.

• Развивање на вештини за решавање проблеми и критичко размислување за време на фазите на изградба, тестирање и решавање проблеми.

• Зајакнување на креативноста и иновативноста преку дизајнирање на уникатно решение за предизвиците од реалниот свет.

• Стекнување искуство во документација и презентација на проекти, корисно за идните академски или професионални активности.

ЦЕЛНО НИВО

• Ученици од средно училиште вклучени во STEM-ориентирани стручни програми

• Ученици специјализирани за автоматизација, роботика или вградени системи

• Технолошки ентузијасти или студентски клубови со основни вештини за кодирање и дизајн на кола

ПОТРЕБНИ ПРЕДЗНАЕЊА:

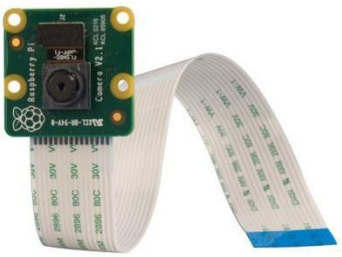
• Основно разбирање на програмирањето во Python

• Запознаеност со електронски компоненти и breadboarding

• Воведно познавање на концепти за вештачка интелигенција или компјутерска визија

• Способност за следење на шеми и поврзување на хардверски модули како серво и сензори

ХАРДВЕР

Материјал	
Raspberry Pi 5	
27W УСБ Тип – Ц Напојување за Raspberry Pi 5	
Жици за поврзување	
Серво мотор DS04-NFC 360° Континуирана ротација	
Raspberry Pi камера со модул 4	

Дрвени плочи за физичката рамка	
Тркала	

Табела 1- Листа на материјали за имплементирање на проектот Паметна канта

СОФТВЕР

• **Оперативен систем:** Raspberry Pi OC (64-битен)

• **Програмска околина:** Python 3 (Thonny или VS Code)

• **Python библиотеки:** opencv-python, ultralytics, torch, gpiozero, lgpio, time, os

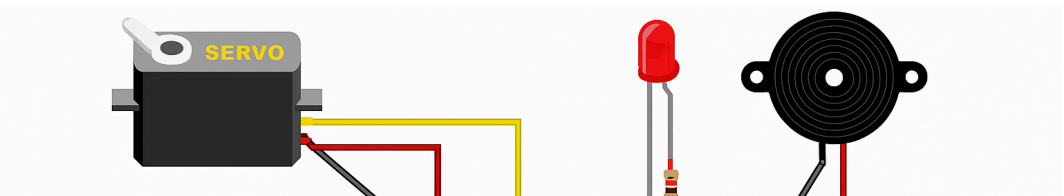
• **Алатки за бази на податоци:** Roboflow (за анотација и експортирање на базата на податоци користена за тренирање 157-2MAT.pt)

• **Платформа за тренирање на модел:** Google Colab (за тренирање и тестирање на YOLOv8 модел)

• **Далечински пристап:** VNC Viewer и SSH (за мониторинг и контрола преку локална мрежа)

КОЛО

Ќе треба да ги поврзете сите компоненти со заземјувањето (GND) и пиновите од 3.3V или 5V за напојување, во зависност од нивните барања. Модулот на камерата се поврзува директно со CSI портот на Raspberry Pi. Континуираните серво мотори се напојуваат преку пиновите од 5V и GND, а нивните сигнални пинови се поврзани со GPIO 17 и GPIO 18. Звучникот е поврзан со GPIO 22, а LED индикаторот е поврзан со GPIO 23 со отпорник за ограничување на струјата. Сите врски се направени безбедно за да се обезбеди стабилно работење, а далечинскиот пристап е овозможен преку VNC или SSH за следење на системот.
Еве го колото за овој проект:



Слика 1 - Коло на проектот Паметна канта

ЧЕКОР-ПО-ЧЕКОР ИНСТРУКЦИИ ЗА SMART THROW

1. Направете го колото

Проверете дали сите електронски компоненти се правилно поврзани:

- Поврзете ги континуираните серво мотори со GPIO 17 и GPIO 18.

- Поврзете го звучниот сигнал со GPIO 22.

- Поврзете ја LED-светилката за статус со GPIO 23 користејќи отпорник за ограничување на струјата.

- Поврзете го модулот 4 на камерата Raspberry Pi со CSI портата за камера.

- Напојувајте ги сите компоненти преку пиновите 5V и GND.

Користете заедничка заземјување за сите компоненти и проверете ги двапати доделувањата на пиновите на GPIO.

2. Подгответе ја софтверската околина

Проверете дали сте го инсталирале следново:

- Raspberry Pi OS (64-битен)

- Python 3

- Потребни библиотеки: opencv-python, ultralytics, pytorch, gpiozero, lgpio, os, time

- Roboflow сметка за управување со податоци

- VNC или SSH овозможен за далечински пристап

Препорачливо е да ја тестирате камерата со помош на основна скрипта пред да го стартувате целиот систем.

3. Стартувајте ја скриптата за детекција на објекти

Користете го следниов Python код за да го стартувате моделот за детекција на објекти и да ги контролирате серво моторите врз основа на препознаениот тип на отпад:

```
1.  #!/home/Desktop/SmartTrash/venv/bin/python
2.  import cv2
3.  from ultralytics import YOLO
4.  from gpiozero import TonalBuzzer, Servo
5.  from gpiozero.pins.lgpio import LGPIOFactory
6.  from gpiozero import Device
7.  from time import sleep
8.  import time
```

```
import threading
import queue
import numpy as np
import os

# --- GPIO and Global Setup ---
try:
    Device.pin_factory = LGPIOFactory()
except Exception as e:
    print(f"Error setting LGPIOFactory: {e}. GPIO operations might fail.")
    print("Ensure pigpiod or lgpiod service is running and gpiozero is configured correctly.")

BUZZER_PIN_NUM = 26
SERVO1_PIN_NUM = 18
SERVO2_PIN_NUM = 19

buzzer = None
servo = None

# Camera settings
CAMERA_INDEX = 0
CAMERA_WIDTH = 640
CAMERA_HEIGHT = 480
CAMERA_FPS = 30

BEEP_FREQUENCY = 800
OBJECT_DETECTED_BEEP_DURATION = 0.2
BEEP_COOLDOWN = 2.0
TARGET_CLASSES = ["limenka", "shishe"]
LATCH_DURATION = 2.0

# --- OpenCV Camera Setup ---
cap = None

def initialize_camera():
    global cap
    cap = cv2.VideoCapture(CAMERA_INDEX)
    if not cap.isOpened():
        print(f"Error: Could not open camera at index {CAMERA_INDEX}")
        return False

    # Set camera properties for low latency
    cap.set(cv2.CAP_PROP_FRAME_WIDTH, CAMERA_WIDTH)
    cap.set(cv2.CAP_PROP_FRAME_HEIGHT, CAMERA_HEIGHT)
    cap.set(cv2.CAP_PROP_FPS, CAMERA_FPS)
    cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)  # Minimize camera buffer

    # Additional low-latency settings (may not work on all cameras)
    try:
        cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G'))  # MJPEG for lower latency
    except:
        pass

    # Verify settings
    actual_width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    actual_height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    actual_fps = cap.get(cv2.CAP_PROP_FPS)

    print(f"Camera initialized:")
    print(f"  Index: {CAMERA_INDEX}")
    print(f"  Resolution: {actual_width}x{actual_height}")
    print(f"  FPS: {actual_fps}")

    return True
```

```
73.
74. # --- Model Configuration ---
75. SPECIFIED_NCNN_MODEL_PATH = "157epoch_ncnn_model"
76. SPECIFIED_BASE_MODEL_PATH = None
77. ncnn_model = None
78.
79. print("--- Model Loading ---")
80. if SPECIFIED_NCNN_MODEL_PATH:
81.     print(f"Attempting to load specified NCNN model from:
    {SPECIFIED_NCNN_MODEL_PATH}")
82.     if os.path.isdir(SPECIFIED_NCNN_MODEL_PATH):
83.         try:
84.             ncnn_model = YOLO(SPECIFIED_NCNN_MODEL_PATH)
85.             print(f"Successfully loaded NCNN model from specified path:
    {SPECIFIED_NCNN_MODEL_PATH}")
86.         except Exception as e:
87.             print(f"Failed to load NCNN model from
    {SPECIFIED_NCNN_MODEL_PATH}: {e}")
88.             print("Will fallback to base model logic if
    SPECIFIED_BASE_MODEL_PATH is set.")
89.             ncnn_model = None
90.     else:
91.         print(f"Specified NCNN model path '{SPECIFIED_NCNN_MODEL_PATH}' is not
    a valid directory.")
92.         print("Will fallback to base model logic if SPECIFIED_BASE_MODEL_PATH
    is set.")
93.
94. if ncnn_model is None:
95.     if SPECIFIED_BASE_MODEL_PATH:
96.         print(f"NCNN model not loaded directly. Using base model path:
    {SPECIFIED_BASE_MODEL_PATH}")
97.         if not os.path.isfile(SPECIFIED_BASE_MODEL_PATH):
98.             print(f"Error: Base model file '{SPECIFIED_BASE_MODEL_PATH}' not
    found.")
99.             cv2.destroyAllWindows()
100.                exit("Fatal: No valid model path provided or base model file
    not found.")
101.            else:
102.                base_model_name_no_ext =
    os.path.splitext(os.path.basename(SPECIFIED_BASE_MODEL_PATH))[0]
103.                derived_ncnn_model_dir = f"{base_model_name_no_ext}_ncnn_model"
104.                print(f"Derived NCNN model directory name:
    {derived_ncnn_model_dir}")
105.                try:
106.                    print(f"Checking for existing NCNN model at:
    {derived_ncnn_model_dir}")
107.                    if os.path.isdir(derived_ncnn_model_dir):
108.                        ncnn_model = YOLO(derived_ncnn_model_dir)
109.                        print(f"Successfully loaded existing NCNN model from
    {derived_ncnn_model_dir}")
110.                    else:
111.                        raise FileNotFoundError
112.                except Exception as e_load_derived:
113.                    print(f"NCNN model not found at '{derived_ncnn_model_dir}'
    or failed to load ({e_load_derived}).")
114.                    print(f"Exporting base model '{SPECIFIED_BASE_MODEL_PATH}'
    to NCNN format...")
115.                    try:
116.                        base_model_for_export = YOLO(SPECIFIED_BASE_MODEL_PATH)
117.                        base_model_for_export.export(format="ncnn", half=False)
118.                        print(f"Export complete. Loading exported NCNN model
    from {derived_ncnn_model_dir}...")
119.                        ncnn_model = YOLO(derived_ncnn_model_dir)
120.                        print(f"Successfully loaded exported NCNN model from
    {derived_ncnn_model_dir}")
121.                    except Exception as e_export_load:
```

```
                        print(f"FATAL error during base model loading, export, or NCNN loading: {e_export_load}")
                        cv2.destroyAllWindows()
                        exit()
        else:
            print("FATAL: Neither SPECIFIED_NCNN_MODEL_PATH nor SPECIFIED_BASE_MODEL_PATH were provided or valid.")
            cv2.destroyAllWindows()
            exit()

    if ncnn_model is None:
        print("FATAL: Could not load any model after all attempts. Exiting.")
        cv2.destroyAllWindows()
        exit()
    print("--- Model loading successful ---")

    # --- Threading Setup ---
    frame_queue = queue.Queue(maxsize=1)  # Reduced buffer size
    results_queue = queue.Queue(maxsize=1)  # Reduced buffer size
    stop_event = threading.Event()

    # --- Thread Functions ---
    def camera_thread_func():
        global stop_event, cap
        print("Camera thread started.")
        while not stop_event.is_set():
            try:
                ret, frame_bgr = cap.read()
                if not ret:
                    print("Camera thread: Failed to read frame. Retrying...")
                    sleep(0.01)  # Reduced sleep time
                    continue

                # Convert BGR to RGB for YOLO processing
                frame_rgb = cv2.cvtColor(frame_bgr, cv2.COLOR_BGR2RGB)

                # Non-blocking put - drop frames if queue is full to reduce latency
                try:
                    frame_queue.put(frame_rgb, block=False)
                except queue.Full:
                    # Drop the frame if queue is full to prevent backlog
                    pass

            except Exception as e:
                print(f"Camera thread error: {e}")
                stop_event.set()
                break
        print("Camera thread finished.")

    def yolo_thread_func():
        global stop_event, ncnn_model
        print("YOLO thread started.")
        while not stop_event.is_set():
            try:
                frame_rgb = frame_queue.get(timeout=0.1)  # Reduced timeout
                results = ncnn_model(frame_rgb, verbose=False, stream=False)

                # Non-blocking put - drop results if queue is full
                try:
                    results_queue.put((frame_rgb, results), block=False)
                except queue.Full:
                    # Drop old results to prevent backlog
                    try:
                        results_queue.get(block=False)  # Remove old result
                        results_queue.put((frame_rgb, results), block=False)
```

```
                    except queue.Empty:
                        pass

                frame_queue.task_done()
            except queue.Empty:
                continue
            except Exception as e:
                print(f"YOLO thread error: {e}")
                stop_event.set()
                break
        print("YOLO thread finished.")

    # --- Servo Control Functions ---
    def servo_stop(s):
        if s:
            s.value = None

    def servo_turn_continuous(s, speed_value):
        """
       Controls continuous servo.
       speed_value: -1.0 (full reverse) to 1.0 (full forward). 0 should be stop.
       """
        if s:
            if -1.0 <= speed_value <= 1.0:
                s.value = speed_value
            else:
                print(f"Invalid servo speed_value: {speed_value}. Must be between -1.0 and 1.0.")

    # --- Main Execution ---
    if __name__ == "__main__":
        main_thread_terminated_normally = False
        try:
            # Initialize camera first
            if not initialize_camera():
                print("Failed to initialize camera. Exiting.")
                exit()

            buzzer = TonalBuzzer(BUZZER_PIN_NUM)
            servo1 = Servo(SERVO1_PIN_NUM)
            servo2 = Servo(SERVO2_PIN_NUM)
            servo_stop(servo1)
            servo_stop(servo2)
            print("Servo initialized and stopped.")

            print("DEBUG: Testing TonalBuzzer immediately after init...")
            try:
                buzzer.play(BEEP_FREQUENCY)
                sleep(OBJECT_DETECTED_BEEP_DURATION)
                buzzer.stop()
                print(f"DEBUG: TonalBuzzer test beep at {BEEP_FREQUENCY}Hz finished. Did you hear it?")
                sleep(0.5)
            except Exception as e:
                print(f"DEBUG: Error during initial TonalBuzzer test: {e}")

            print("Camera ready.")
            sleep(1)

            cam_thread = threading.Thread(target=camera_thread_func, daemon=True)
            proc_thread = threading.Thread(target=yolo_thread_func, daemon=True)

            print("Starting worker threads...")
```

```
            cam_thread.start()
            proc_thread.start()

            prev_time_fps = time.time()
            fps = 0.0
            last_beep_trigger_time = 0
            object_currently_triggering_beep = False
            object_detected_start_time_for_latch = None
            latch_active = False
            active_detected_class_name = ""

            print("Starting detection loop. Press 'q' in OpenCV window to quit.")

            while not stop_event.is_set():
                try:
                    original_frame_rgb, yolo_results_list = results_queue.get(timeout=0.1)  # Reduced timeout
                except queue.Empty:
                    if not cam_thread.is_alive() and cam_thread.ident is not None:
                        print("Error: Camera thread seems to have stopped unexpectedly.")
                        stop_event.set()
                        break
                    if not proc_thread.is_alive() and proc_thread.ident is not None:
                        print("Error: YOLO processing thread seems to have stopped unexpectedly.")
                        stop_event.set()
                        break

                    # Don't show "waiting" frame as frequently to reduce overhead
                    key = cv2.waitKey(1) & 0xFF
                    if key == ord("q"):
                        print("User pressed 'q'.")
                        stop_event.set()
                    continue

                current_time = time.time()
                delta_time_fps = current_time - prev_time_fps
                if delta_time_fps > 0:
                    fps = 1.0 / delta_time_fps
                prev_time_fps = current_time

                found_target_object_this_frame = False
                confidence_of_detected_object = 0.0
                yolo_results = yolo_results_list[0]

                if yolo_results.boxes:
                    for box in yolo_results.boxes:
                        class_id = int(box.cls[0])
                        class_name = ncnn_model.names[class_id]
                        confidence = float(box.conf[0])
                        if class_name in TARGET_CLASSES and confidence > 0.5:
                            found_target_object_this_frame = True
                            confidence_of_detected_object = confidence
                            active_detected_class_name = class_name
                            break

                if found_target_object_this_frame:
                    if not object_currently_triggering_beep:
                        if current_time - last_beep_trigger_time > BEEP_COOLDOWN:
```

```
                            print(f"Object '{active_detected_class_name}' detected! Confidence: {confidence_of_detected_object:.2f}. Activating beep/servo...")
                            try:
                                buzzer.play(BEEP_FREQUENCY)
                            except Exception as e:
                                print(f"Error during beep/servo activation: {e}")
                            finally:
                                servo_stop(servo)
                                if buzzer:
                                    buzzer.stop()
                                print(f"DEBUG: Beep/servo sequence for '{active_detected_class_name}' finished.")
                            last_beep_trigger_time = current_time
                            object_currently_triggering_beep = True
                else:
                    if object_currently_triggering_beep:
                        servo_stop(servo)
                        object_currently_triggering_beep = False

                if found_target_object_this_frame:
                    if object_detected_start_time_for_latch is None:
                        object_detected_start_time_for_latch = current_time
                    elif not latch_active and (current_time - object_detected_start_time_for_latch >= LATCH_DURATION):
                        print(f"LATCH TRIGGERED: '{active_detected_class_name}' continuously detected for {LATCH_DURATION:.1f}s.")
                        latch_active = True
                        if active_detected_class_name == "limenka":
                            servo1.value=-1.0
                            sleep(1.2)
                            servo1.value=None
                            servo2.value=1.0
                            sleep(0.4)
                            servo2.value=None
                            sleep(2.0)
                            servo2.value=0.0
                            sleep(0.8)
                            servo1.value=1.0
                            sleep(0.8)
                            servo2.value=None
                            servo1.value=None
                        elif active_detected_class_name == "shishe":
                            servo1.value=1.0
                            sleep(1.2)
                            servo1.value=None
                            servo2.value=1.0
                            sleep(0.4)
                            servo2.value=None
                            sleep(2.0)
                            servo2.value=0.0
                            sleep(0.8)
                            servo1.value=-1.0
                            sleep(0.8)
                            servo2.value=None
                            servo1.value=None
                else:
                    if object_detected_start_time_for_latch is not None:
                        object_detected_start_time_for_latch = None
                        if latch_active:
                            print("Latch deactivated (target object lost).")
                            latch_active = False

                annotated_frame_rgb = yolo_results.plot()
```

```
                annotated_frame_bgr_display = cv2.cvtColor(annotated_frame_rgb, cv2.COLOR_RGB2BGR)

                cv2.putText(annotated_frame_bgr_display, f"FPS: {fps:.2f}", (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0), 2)
                latch_text = f"Latch: {'ACTIVE' if latch_active else 'Inactive'}"
                latch_color = (0, 255, 0) if latch_active else (0, 0, 255)
                cv2.putText(annotated_frame_bgr_display, latch_text, (10, 60), cv2.FONT_HERSHEY_SIMPLEX, 0.7, latch_color, 2)
                if found_target_object_this_frame:
                    cv2.putText(annotated_frame_bgr_display, f"Detected: {active_detected_class_name}", (10, 90), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255, 100, 0), 2)

                cv2.imshow("Camera", annotated_frame_bgr_display)
                results_queue.task_done()

                key = cv2.waitKey(1) & 0xFF
                if key == ord("q"):
                    print("User pressed 'q'.")
                    stop_event.set()

            main_thread_terminated_normally = True

        except KeyboardInterrupt:
            print("KeyboardInterrupt received. Exiting...")
            stop_event.set()
        except Exception as e:
            print(f"An unexpected error occurred in the main loop: {e}")
            import traceback
            traceback.print_exc()
            stop_event.set()
        finally:
            print("Initiating shutdown sequence...")
            if not stop_event.is_set():
                stop_event.set()

            if 'cam_thread' in locals() and cam_thread.is_alive():
                print("Joining camera thread...")
                cam_thread.join(timeout=2)
                if cam_thread.is_alive():
                    print("Camera thread did not join in time.")
            if 'proc_thread' in locals() and proc_thread.is_alive():
                print("Joining YOLO thread...")
                proc_thread.join(timeout=2)
                if proc_thread.is_alive():
                    print("YOLO thread did not join in time.")

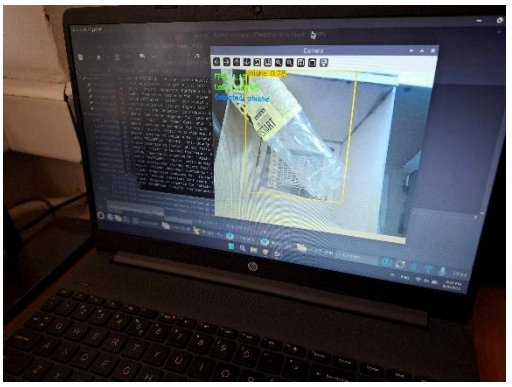
            print("Closing GPIO resources...")
            if buzzer:
                try:
                    buzzer.stop()
                    buzzer.close()
                    print("Buzzer stopped and closed.")
                except Exception as e:
                    print(f"Error closing buzzer: {e}")
            if servo:
                try:
                    servo_stop(servo)
                    servo.close()
                    print("Servo stopped and closed.")
                except Exception as e:
                    print(f"Error closing servo: {e}")

            if cap and cap.isOpened():
```

```
422.                 print("Releasing camera...")
423.                 cap.release()
424.                 print("Camera released.")
425.
426.             cv2.destroyAllWindows()
427.             print("OpenCV windows destroyed.")
428.             print("Program shutdown complete.")
```

4. Набљудувајте ги резултатите





КРИТЕРИУМИ ЗА ЕВАЛУАЦИЈА:

Имплементација на хардвер (40%): Правилно поврзано коло, без лабави жици, правилно поставување на сензорот.

Функционалност на кодот (30%): Скриптата работи без грешки, правилно ги детектира промените на светлината.

Разбирање и објаснување (20%): Способност за објаснување на тоа како функционира системот и неговите апликации во реалниот свет.

Креативност и оптимизација (10%): Предлози за подобрувања, како што е додавање дополнителни функции за автоматизација.

ПРОЦЕНЕТО ВРЕМЕТРАЕЊЕ:

- Датум на почеток: 5 март 2025 година
- Датум на завршување: 20 мај 2025 година

РЕЗУЛТАТИ ОД УЧЕЊЕТО:

• Разбирање како вештачката интелигенција може да се примени за решавање на проблеми со животната средина во реалниот свет

• Развивање вештини за интегрирање на хардвер (сензори, серво) со софтверски решенија базирани на вештачка интелигенција

• Стекнување практично искуство со алатки како Raspberry Pi, Python и Roboflow

• Подобрување на вештините за тимска работа, планирање на проекти и техничка документација

• Научете да оценувате и усовршувате прототипови преку повратни информации и тестирање

1. Спроведување тестови на системот од почеток до крај: Потврдете дека целиот систем - од откривање на отпад до сортирање - работи правилно и сигурно.

ПРЕДИЗВИЦИ И СОВЕТИ ЗА РЕШАВАЊЕ НА ПРОБЛЕМИ:

• Неуспеси во препознавањето на камерата – Тестирајте ја камерата со основни скрипти за да се осигурате дека снима влезни податоци. Повторно калибрирајте го моделот или проверете ги условите на осветлување ако точноста се намали.

• Неисправности на серво моторот – Проверете ја конфигурацијата на GPIO пинот и потврдете го напојувањето; заменете ги моторите ако се прегреат или треперат.

• Неточност на моделот – Користете повеќе анотирани слики во Roboflow и повторно обучете го моделот ако откривањето на објекти не успее или погрешно се класифицира.

• Грешки во софтверот – Проверете ги зависностите од Python и верзиите на библиотеката; користете изјави за печатење или логирање за полесно дебагирање.

• Проблеми со поврзувањето (VNC/SSH) – Проверете дали Raspberry Pi е на точната мрежа и потврдете дека SSH е овозможен во поставките.