# Detailed Documentation for functions.py and main.py

## functions.py — Function-by-Function Explanation

### GPIO and PWM Setup (Top of File)

## **Purpose**: Initialize hardware control for DC motor and steering servo using Raspberry Pi’s GPIO pins and PWM channels.

### 1. set\_motor(speed)

**Purpose**: Controls the direction and speed of the DC motor using the L298N driver. **Parameters**: - speed (int): Range -100 to 100. Positive values move forward, negative backward, zero stops.

**Logic**: - Sets IN1/IN2 pins for direction. - Sets PWM duty cycle for speed (clamped to 0-100).

**When called**: Whenever the robot needs to move forward, backward, or stop.

### 2. set\_servo\_angle(angle)

**Purpose**: Sets the steering angle for the servo motor. **Parameters**: - angle (int): Desired steering angle (clamped between 60 left and 160 right).

**Logic**: - Converts angle to PWM duty cycle. - Updates servo, allows 20ms for movement.

**When called**: On every steering update.

### 3. write(value)

**Purpose**: Unified interface for motor and servo control. **Parameters**: - value (int): If 60-160, sets steering; if 1000-2000, sets motor speed.

**Logic**: - Calls set\_servo\_angle or set\_motor depending on range.

**When called**: For simple control commands (in main loop and maneuvers).

### 4. multi\_write(sequence)

**Purpose**: Executes a list of actions: steering, speed, or sleep. **Parameters**: - sequence (list): List of values; numbers <25 are treated as sleep durations.

**Logic**: - Loops through sequence, calls write or time.sleep.

**When called**: For initialization or multi-step maneuvers.

### 5. stop\_car()

**Purpose**: Stops all robot movement and closes OpenCV windows.

**Logic**: - Sets motor speed to 0. - Centers steering. - Destroys OpenCV windows.

**When called**: On exit or when robot needs to halt.

### 6. move\_backward\_with\_mirror(forward\_angle, duration=1, speed=50)

**Purpose**: Moves robot backward with steering mirrored to last forward angle (for advanced maneuvers). **Parameters**: - forward\_angle (int): Last used steering angle. - duration (float): Time to move backward. - speed (int): Speed for backward movement.

**Logic**: - Computes mirrored angle. - Moves backward for the given duration. - Returns steering to forward angle.

**When called**: For reversing, not used in main loop.

### 7. display\_roi(img, ROIs, color)

**Purpose**: Draws rectangles on the image for visualizing detection zones. **Parameters**: - img (np.ndarray): The image. - ROIs (list): List of [x1, y1, x2, y2] rectangles. - color (tuple): BGR color for drawing.

**Logic**: - Draws lines for each ROI.

**When called**: For debugging with live OpenCV window.

### 8. find\_contours(img\_lab, lab\_range, ROI)

**Purpose**: Finds contours of a specified color in a region of interest. **Parameters**: - img\_lab (np.ndarray): LAB color space image. - lab\_range (list): Lower and upper LAB bounds. - ROI (list): [x1, y1, x2, y2] for cropping.

**Logic**: - Crops the ROI, applies mask for color, erodes/dilates to reduce noise, finds contours.

**When called**: Every frame for stripe/wall detection.

### 9. max\_contour(contours, ROI)

**Purpose**: Finds largest contour in a region, returns area and position. **Parameters**: - contours (list): List of contours. - ROI (list): The detection region.

**Logic**: - Loops through contours, selects the largest by area. - Computes adjusted position for main loop.

**When called**: For PID control and turn detection.

### 10. display\_variables(variables)

**Purpose**: Prints current state variables for debugging in console. **Parameters**: - variables (dict): Name-value pairs.

**Logic**: - Prints variables, then moves cursor up to overwrite next frame.

**When called**: If debug=True, prints every frame.

## main.py — Main Control Loop Block Explanations

### **Startup Block**

* **Waits for button press**: Robot only starts when user presses the physical button (for safety).
* **Initializes camera**: Sets up USB webcam (resolution and FPS).

### **ROI Setup**

* **Defines five ROIs**: Each is a rectangle [x1, y1, x2, y2].
  + ROI1: Bottom left (left wall)
  + ROI2: Bottom right (right wall)
  + ROI3: Center (stripe detection)
  + ROI4: Far left (orange turn detection)
  + ROI5: Far right (blue turn detection)
* **Purpose**: Focuses color and contour detection on relevant areas, improving speed and accuracy.

### **Control Variables**

* **PID gains (kp, kd)**: Tune steering responsiveness.
* **Angle limits**: straightConst, sharpLeft, sharpRight, maxLeft, maxRight define steering bounds.
* **Speed**: Sets constant running speed.
* **Color logic**: Tracks active course color (orange or blue), stripe count, debounce timing.

### **Main Loop**

**Runs every frame:**

#### 1. **Image Acquisition & Processing**

* Captures frame from camera.
* Converts to LAB color space, applies Gaussian blur for noise reduction.

#### 2. **Contour Detection**

* Calls find\_contours for each ROI and color:
  + rBlack in ROI1/ROI2 for walls.
  + rOrange/rBlue in ROI3 for stripes.
* Calls max\_contour to get largest detected area (used for PID and turn logic).

#### 3. **Active Color Selection**

* If activeColor is None, sets to first detected stripe (orange or blue).
* Starts counting stripes only for that color.

#### 4. **Stripe Counting (Debounce)**

* Increments colorCount only if enough time has passed since last detection (prevents double-counting).

#### 5. **Stopping Logic**

* When 12 stripes are counted, sets a timer to stop after a few seconds.
* Calls stop\_car() and halts loop.

#### 6. **Turn Handling**

* **Blue course**: If black wall lost in ROI4 (left), rotates left until wall found in ROI1.
* **Orange course**: If black wall lost in ROI5 (right), rotates right until wall found in ROI2.
* Uses write(angle) for hard steering.

#### 7. **PID Steering Control**

* **Calculates aDiff**: Difference in wall area (right minus left).
  + If right wall more visible, angle increases (steer left).
  + If left wall more visible, angle decreases (steer right).
* **Angle calculation**:
  + angle = straightConst + (aDiff \* kp) + (aDiff - prevDiff) \* kd
  + Ensures robot stays centered between walls.
  + Uses clamping: If wall is lost, applies sharp or maximum turn logic.

#### 8. **Debug Display**

* If debug=True, draws ROIs on image and prints key variables.
* cv2.imshow shows live video, stops on ‘q’ key press.

#### 9. **Cleanup**

* Releases camera, closes windows, stops motors on exit.

## How Angle Calculation and PID Control Work

* **PID wall following**:
  + The robot tries to keep both left and right walls equally detected (i.e., equal area in ROI1/ROI2).
  + If right wall area is greater, robot must steer left to center.
  + If left wall area is greater, robot must steer right.
  + PID logic smooths steering to avoid oscillations and over-corrections.
* **Turn logic**:
  + When a wall is lost in turn ROIs, robot rotates in place in the direction required until wall is reacquired in main wall ROI.
  + This reliably handles sharp corners and intersections.

## Summary

* functions.py provides all hardware and vision utilities.
* main.py orchestrates the robot’s behavior: image analysis, PID steering, turn handling, stripe counting, and stopping.
* ROIs and color masks ensure reliable, efficient vision.
* PID keeps robot centered; turn logic ensures it navigates sharp turns.