# Main Program (main.py) — Detailed Documentation

main.py contains the main execution loop for the autonomous robot, orchestrating image processing, hardware control, stripe counting, PID steering, and turn handling.  
This file is the entry point for your robot’s run.

## Overview of Main Blocks

### 1. **Startup & Initialization**

* **Button Press to Start**  
  Waits for the user to press a physical button (GPIO 26, active LOW) before starting. This ensures safe, controlled startup.
* **Camera Initialization**  
  Sets up the USB camera at 1280x720 resolution, 30 FPS.  
  Checks if the camera opens correctly; exits if not.

### 2. **Region of Interest (ROI) Setup**

Defines 5 ROIs (ROI1 to ROI5) as rectangles [x1, y1, x2, y2]: - **ROI1**: Bottom left — left wall detection - **ROI2**: Bottom right — right wall detection - **ROI3**: Center — stripe detection - **ROI4**: Far left — used for orange turn logic - **ROI5**: Far right — used for blue turn logic

**Purpose:**  
ROIs allow focused contour detection, increasing accuracy and reducing computational load.

### 3. **Control Variables**

* **PID parameters:**
  + kp (proportional gain): Controls how strongly the robot reacts to wall area difference.
  + kd (derivative gain): Dampens oscillations based on change in area difference.
* **Steering constants:**
  + straightConst: Angle for straight driving.
  + sharpLeft, sharpRight, maxLeft, maxRight: Boundaries for sharp turns.
* **Speed:**
  + Constant value (speed = 1500) for motor PWM.
* **Course color detection:**
  + activeColor: Set to "orange" or "blue" on first detected stripe.
  + colorCount: Tracks the number of stripes detected.
  + debounceSec: Ensures stripes aren’t double-counted.

### 4. **Main Loop — Frame-by-Frame Workflow**

#### a. **Image Capture and Preprocessing**

* Captures a frame.
* Converts frame to LAB color space (more robust to lighting).
* Applies Gaussian blur to reduce noise.

#### b. **Contour Detection**

* Uses functions from functions.py:
  + find\_contours() called for each relevant ROI and color mask.
  + max\_contour() retrieves area and position for the largest contour.

#### c. **Course Color Selection & Stripe Counting**

* If activeColor is not set, assigns based on first detected stripe.
* Stripe count (colorCount) incremented only if sufficient time has passed since last detection (debounceSec logic).

#### d. **Stopping Logic**

* When 12 stripes detected, robot waits a set period (e.g., 4 seconds) then calls stop\_car() and exits the loop.

#### e. **Turn Handling**

* **Blue Course:**  
  If left wall is lost in ROI4, rotates left (steers to minimum angle) until left wall is detected in ROI1.
* **Orange Course:**  
  If right wall is lost in ROI5, rotates right (steers to maximum angle) until right wall is detected in ROI2.
* **Purpose:**  
  Allows robot to reliably handle sharp turns and intersections.

#### f. **PID Steering Control**

* Calculates aDiff = rightArea - leftArea (difference in wall detection).
* **Angle calculation:**  
  angle = straightConst + aDiff \* kp + (aDiff - prevDiff) \* kd
  + If more right wall detected (positive aDiff), robot steers left.
  + If more left wall detected (negative aDiff), robot steers right.
* **Sharp Turn Correction:**  
  If wall area drops below threshold, robot applies hard turn until wall reacquired.

#### g. **Debug Display**

* If debug is True, overlays ROIs on live camera feed and prints key variables.
* Exits if user presses ‘q’.

### 5. **Cleanup**

* Releases the camera.
* Closes all OpenCV windows.
* Calls stop\_car() to halt motors and center steering.

## How Angle Calculation and PID Steering Work

* **Wall following:**
  + The robot aims to keep equal wall area in ROI1 and ROI2.
  + PID control smooths and stabilizes steering, reducing oscillations and keeping the robot centered.
* **Turn handling:**
  + When a wall is lost in turn ROIs, robot rotates in place until the wall is found in main ROI.
  + This avoids getting stuck or missing sharp corners.

## Key Functions Used in main.py

* find\_contours(img\_lab, mask, ROI): Finds contours for a color in a region.
* max\_contour(contours, ROI): Gets largest detected area and position.
* write(angle) & multi\_write([angle, speed]): Set steering and speed together.
* stop\_car(): Safely stops robot.

## Example Control Flow

1. Start robot with button press.
2. Begin capturing frames, process each for stripes and walls.
3. Use PID to steer between walls.
4. Detect colored stripes and count.
5. Handle turns using dedicated ROIs.
6. Stop after completing course.

## Troubleshooting

* If camera doesn’t open, check USB connection or camera index.
* If robot doesn’t move or steer correctly, verify wiring and GPIO pin assignments.
* Use debug mode for live feedback.

## References

* [OpenCV LAB Color Processing](https://docs.opencv.org/)
* [RPi.GPIO Library](https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/)
* [L298N Motor Driver Guide](https://www.electronicwings.com/nodemcu/l298n-motor-driver)