Code Dokumentation

# Einführung

Das vorliegende Skript dient der Steuerung eines Kamerasystems auf einem Raspberry Pi. Es verwendet GPIO-Pins zur Steuerung von Motoren, um Kameras zu positionieren und Bilder zu erfassen. Anschließend werden die Bilder analysiert, um Zentren von Objekten zu bestimmen, die Positionen zu berechnen und die Ergebnisse in einem Word-Dokument zu speichern.

# Funktionen im Code

## 1. initialize\_gpio()

Diese Funktion setzt die GPIO-Pins für die Steuerung von Motoren und Kameras. Sie sorgt dafür, dass keine Warnungen ausgegeben werden und stellt den GPIO-Modus auf BOARD, um die Pins auf dem Raspberry Pi zu definieren.

## 2. select\_camera()

Mit dieser Funktion wird eine Kamera ausgewählt und der entsprechende I2C-Befehl an den Adapter gesendet. Es wird auch der GPIO-Ausgang auf Basis des Kameraindex eingestellt, um die Kamera zu aktivieren.

## 3. initialize\_and\_set\_camera()

Diese Funktion initialisiert eine Kamera und setzt deren Steuerungsparameter, wie Belichtung, Weißabgleich und Verstärkung. Die Kamera wird dann gestartet, um Bilder aufzunehmen.

## 4. initialize\_and\_capture\_images()

Diese Funktion nimmt zwei Bilder mit den angegebenen Kameras auf. Sie initialisiert zuerst die Kameras und wartet kurz, bevor sie ein Bild von jeder Kamera aufnimmt. Danach werden die Bilder gestoppt.

## 5. process\_image()

Diese Funktion führt die Bildverarbeitung durch. Sie wählt die Kamera aus, nimmt ein Bild auf, berechnet den Mittelpunkt des Objekts und speichert die Koordinaten. Anschließend wird der Motor bewegt, um eine neue Position einzunehmen, basierend auf den berechneten Werten.

## 6. calculate\_centroid()

Hier wird die Berechnung des Schwerpunkts eines Objekts durchgeführt. Mit dem Otsu-Verfahren wird der Bildschwellenwert berechnet, und der Mittelpunkt des größten Objekts wird ermittelt. Der Mittelpunkt wird zur weiteren Verarbeitung gespeichert.

## 7. append\_to\_coordinates()

Diese Funktion fügt die berechneten Koordinaten der x- und y-Achsen-Listen hinzu. Sie wird verwendet, um die Positionen während der Bildverarbeitung zu speichern.

## 8. move\_to\_pos()

Mit dieser Funktion wird der Motor zu einer neuen Position bewegt. Es wird ein Datenpaket an den Motorcontroller gesendet, um die neue Position zu erreichen.

## 9. current\_pos()

Diese Funktion liest die aktuelle Position des Motors aus. Sie sendet eine Abfrage an den Motorcontroller und erwartet eine Antwort, die die aktuelle Position enthält.

## 10. startposition()

Die Funktion setzt den Motor auf eine Startposition, indem sie die Motoren auf vorgegebene Werte bewegt und sicherstellt, dass alle Achsen korrekt positioniert sind, bevor die Messungen beginnen.

## 11. endposition()

Diese Funktion bewegt die Motoren zum Ende der Messung, um die Positionen zu sichern.

## 12. position\_y()

Die Funktion setzt den Motor auf die Y-Achse-Startposition zurück, um die Messungen zu beenden.

## 13. int\_to\_bytes()

Diese Funktion konvertiert eine Ganzzahl in ein Byte-Array, das dann an den Motorcontroller gesendet werden kann.

## 14. showResults()

Die Funktion berechnet die Ausgleichsgeraden für die Positionen und berechnet den Fehler. Die Ergebnisse werden dann in einem Word-Dokument gespeichert, um eine Dokumentation der Messungen zu erstellen.

# Ergebnisse

Das Skript berechnet für jede Messung die Fehler der X- und Y-Koordinaten und speichert die Ergebnisse in einem Word-Dokument. Die Werte werden als Mittelwert der Abweichungen der beiden Kameras berechnet.