Programmablauf

Der Programmablauf wird in diesem Abschnitt Schritt für Schritt erklärt und anhand geeigneter Diagramme dargestellt. Zu einer genauen Übersicht des Ablaufes ist der Flowchart des Programmes (MainFlowchart.png – Anhang – Figure 3) hilfreich. Nun eine chronologische Erklärung der Programmteile.

doaV2

Nach dem Programmaufruf wird die DOA Konfiguration, bestehend aus vorberechneten Werten, ausgelesen. Dies wird durch die doaV2 Klasse mithilfe der loadPrecalculatedValues Methode ermöglicht (s. Figure 1).

Hier wird zunächst die Konfigurationsdatei geöffnet. Nun werden für alle möglichen Mikrofonpaare alle vorberechneten Laufzeiten aller möglichen Winkel eingelesen und in das Array precalculatedCompareValues abgespeichert. Zuletzt wird der Zugriff auf die Datei gestoppt. Die DOA Klasse beinhaltet auch die Berechnung des Ursprungsortes eines Geräusches, auf die im späteren Programmverlauf weiter eingegangen wird.

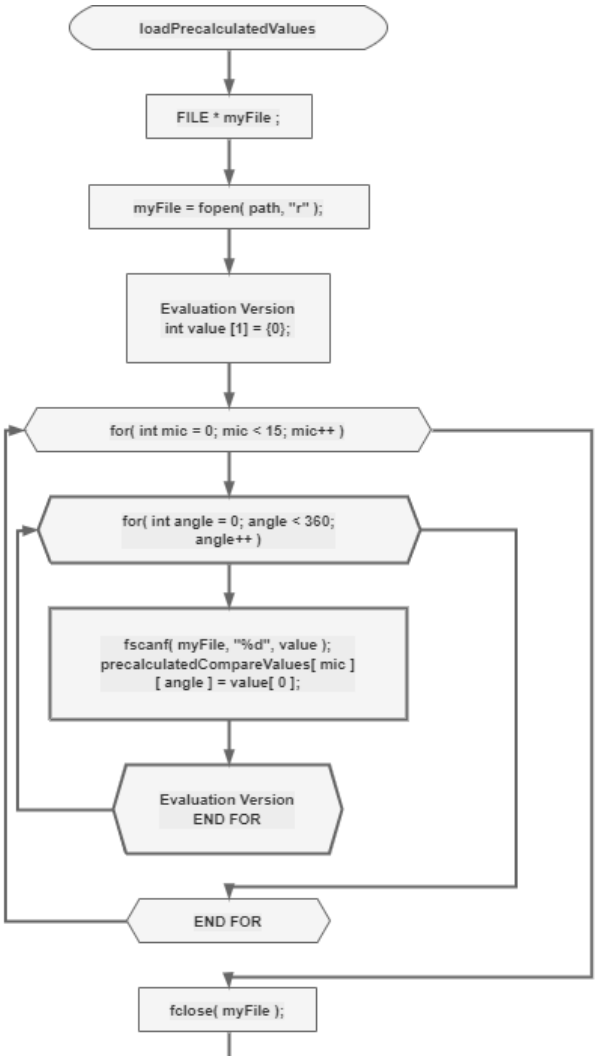

Figure 1 - Flowchart der doaV2 Klasse

create_pipe

Nach dem Laden der Laufzeiten wird mithilfe der create_pipe Methode ein FIFO angelegt, mit dem die Echtzeitmessungen zur Datenverarbeitung weitergegeben werden.

updateValues

Diese Methode stellt sicher, dass der Daten Buffer nach jedem FIFO Zyklus neu gefüllt ist. Die Methode bestimmt zuerst das Maximum an gemessenen Werten aus einem Zyklus und schreibt sie dann mithilfe der *fastAddValue* Methode im Signed UInt32 Format in den Speicher.

processFFT

Eine einfache „Fast Fourier Transformation“, die ein Array an komplexen Gleitkommazahlen ausgibt. Mithilfe dieser Fouriertransformation ist es möglich, die Intensität der Geräusche von der zwischengespeicherten Aufnahme herauszufinden, um diese zu vergleichen.

getDataForSoundDetect

Im Programmablauf folgt diese einfache Getter-Methode, um die gepufferten Daten aus dem FIFO zur Weiterverarbeitung in ein Array zu schieben.

checkForBang

Die checkForBang Methode ist eine der wichtigsten im gesamten Programmablauf und sitzt direkt in einer Bedingung. Wenn durch diese Methode ein lautes Signal bestätigt wird, fährt das Programm in der Verarbeitung dieses Signales fort.

Wie in Figure 2 zu erkennen, wird anfangs mithilfe der calcEnergy Methode aus derselben Klasse der Unterschied zwischen den Energieniveaus von Geräuschen ausgerechnet. Dies geschieht durch die schon erwähnte FFT. Wenn nun die errechnete Differenz den Wert 600000 überschreitet, kann davon ausgegangen werden, dass ein lautes und beobachtungswertes Event stattgefunden hat, da eine sehr hohe Energiedifferenz zu vorherigen Werten gemessen wurde.

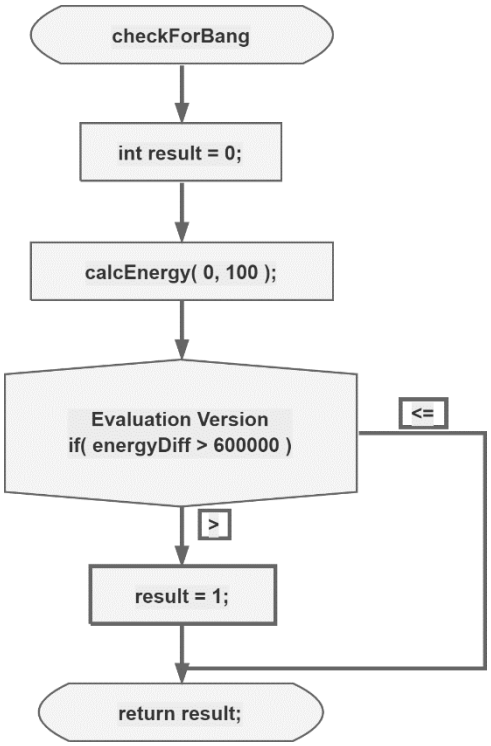

Figure 2 – Flowchart der checkForBang Methode

getAngle

Diese Methode ist nun dafür verantwortlich, dem Geräuschevent eine eindeutige Richtung zuzuordnen, um eine Beweisführung zu ermöglichen. Sie ruft zwei weitere wichtige Methoden auf – xcorr und compareValues.

Mithilfe der vorberechneten Korrelationswerte für das Mikrofonarray kann die xcorr Methode verwendet werden, um die Korrelation zweier Signale zu finden.

Mit compareValues werden die Korrelationen verglichen und mit einer einfachen Sortierung die stärksten Übereinstimmungen gefunden, um daraus eine Richtung zu erschließen.

Mehr dazu im Abschnitt des Protokolls, dass auf die Berechnung des Winkels näher eingeht.

cameraControl

Die Klasse cameraControl benutzt nun, wenn sich der errechnete Winkel im richtigen Sichtfeld der Kamera befindet, zwei Methoden, um ein Foto zu erstellen und dies so zu bearbeiten, das der Bereich der Schallquelle sichtbar wird. Mithilfe der einfachen Schnittstelle der RaspiCam und dazugehörigen Bibliotheken wird ein Bild im RGB-Format von der Kamera angefordert und abgespeichert. Nun wird mithilfe des errechneten Ergebnisses aus getAngle und OpenCV das Bild bearbeitet, indem durch mehrere else if abfragen zum richtigen Bildsegment iteriert wird und dann dieses Segment mit einem Rechteck hervorgehoben wird.

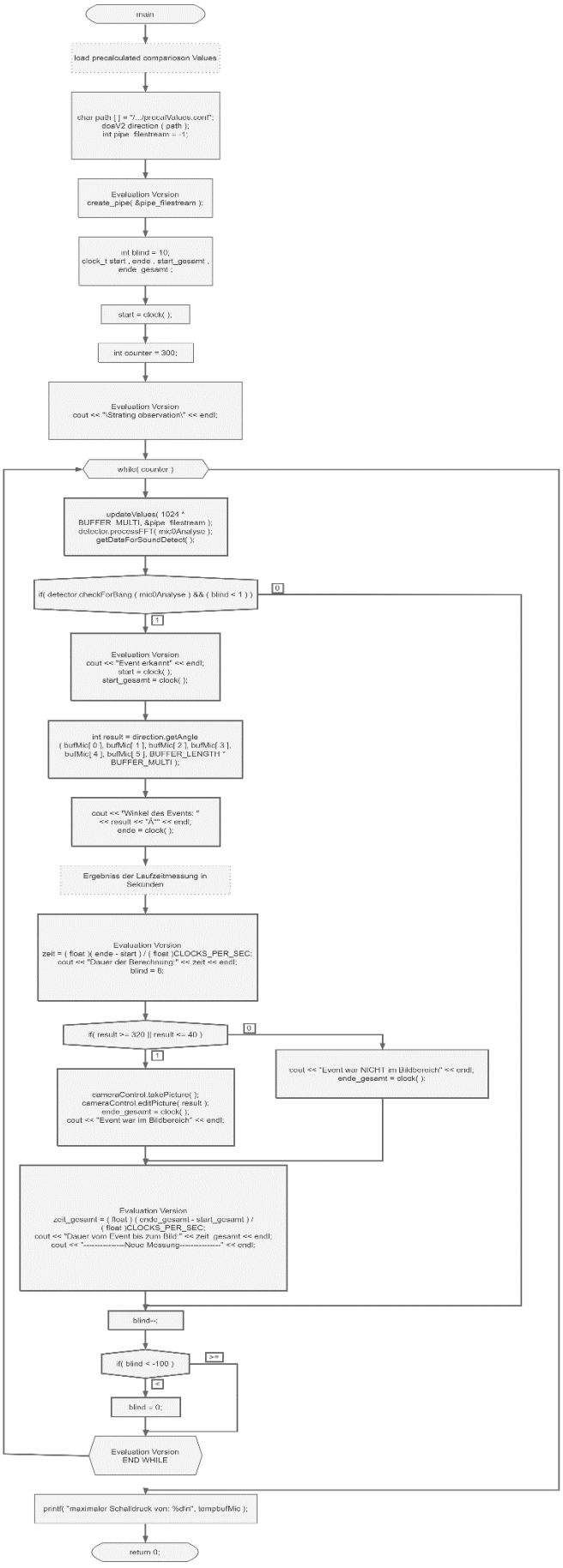

Figure 3 - Flowchart der Main