Erarbeitungs-/Reflexionsphase

*Einleitung*

In der zweiten Phase der Entwicklung des Projekts, welches die serielle Kommunikation zwischen einer PC-Anwendung und einem Mikrocontroller implementiert, wurden die in der Konzeptionsphase geplanten Schritte umgesetzt. Die Erarbeitungs-/Reflexionsphase beinhaltet die Entwicklung und Testung aller Module sowie die Überprüfung der Interaktion zwischen den Komponenten. Ziel war es, eine funktionale erste Version der Software bereitzustellen, die alle Grundfunktionen korrekt ausführt.

*Umsetzung der PC-Anwendung*

1. **Verbindungsmanagement:**
   * Die PC-Anwendung wurde in C++ unter Verwendung des Qt-Frameworks entwickelt.
   * Ein Hauptmerkmal ist die Verbindung zu einem Mikrocontroller über eine serielle Schnittstelle. Ein Button ermöglicht die Verbindung und Trennung.
   * Es wurde ein Mechanismus implementiert, der die Verbindung alle 100 Millisekunden überprüft. Der Status wird durch eine LED-Anzeige (rot=nicht verbunden/grün=verbunden) visualisiert.
2. **Eingabevalidierung:**
   * Die Funktion check\_input() überprüft die Benutzereingaben. Erlaubte Formate sind z. B. a+b, a-b, a\*b und a/b.
   * Negative Zahlen und Dezimalwerte werden unterstützt. Es wird überprüft, ob durch null geteilt wird.
3. **Kommunikation und Interaktion:**
   * Die eingegebene Rechenoperation wird nach Überprüfung auf Richtigkeit an den Mikrocontroller gesendet.
   * Die Rückmeldung des Mikrocontrollers wird in einem Textfeld angezeigt.
   * Ein Button ermöglicht das Speichern aller Interaktionen in einer Textdatei.
4. **Fehlerbehandlung:**
   * Fehlermeldungen, wie bei Verbindungsverlust, werden in einem separaten Logfeld und als Dialog angezeigt. Eine Variable wurde implementiert, um sicherzustellen, dass Dialoge bei Verbindungsverlust nur einmal erscheinen.

*Umsetzung der Mikrocontroller-Anwendung*

1. **Initialisierung:**
   * Konfiguration der seriellen Schnittstelle zur Kommunikation mit der PC-Anwendung.
2. **Eingangsverarbeitung:**
   * Die empfangenen Daten werden in drei Variablen (Zahl1, Operator, Zahl2) aufgeteilt.
   * Bei Dezimalzahlen wird das Komma in einen Punkt umgewandelt.
3. **Berechnungen:**
   * Die arithmetische Operation wird basierend auf dem Operator ausgeführt.
   * Division durch null wird abgefangen und entsprechend behandelt.
4. **Ausgabe:**
   * Das Ergebnis wird als Zeichenkette formatiert und an die PC-Anwendung zurückgesendet.

*Codequalität und Testung*

* **Kommentierung:** Der gesamte Code wurde mit klaren Kommentaren versehen, um die Verständlichkeit zu erhöhen.
* **Lesbarkeit:** Einheitliche Namenskonventionen und einheitliches Codelayout wurden eingehalten.
* **Testen:**
  + Jedes Modul wurde einzeln getestet.
  + Die Interaktion zwischen PC- und Mikrocontroller-Anwendung wurde durch verschiedene Szenarien überprüft (z. B. fehlerhafte Eingaben, Verbindungsverlust, gültige Berechnungen).
  + Das System wurde erfolgreich auf die Behandlung von Sonderfällen geprüft.

*Herausforderungen und Lösungen*

1. **Verbindungsverlust erkennen:**
   * Herausforderung: Verbindungsverluste korrekt zu erkennen und zu melden.
   * Lösung: Ein Hintergrundthread überprüft den Verbindungsstatus. Ein Meldungsdialog erscheint nur einmal pro Verbindungsverlust.
2. **Flexibilität bei Endzeichen:**
   * Herausforderung: Unterschiedliche Endzeichen (\n, \r, \r\n) zu unterstützen.
   * Lösung: Eine Konfigurationsoption wurde implementiert, mit der Benutzer das Endzeichen auswählen können.
3. **Eingabeverarbeitung:**
   * Herausforderung: Benutzerfreundlichkeit und Fehlertoleranz bei Eingaben sicherzustellen.
   * Lösung: Die Funktion check\_input() wurde robust gestaltet, um alle gängigen Eingabeformate zu akzeptieren und Benutzer bei Fehlern zu informieren.

*Nächste Schritte*

* Optimierung und Erweiterung der Software in der Finalisierungsphase.
* Berücksichtigung von Feedback durch Tests und Tutorien.
* Erstellung eines Abstracts und einer abschließenden Zusammenfassung.
* Behebung kleinerer verbleibender Fehler und Verbesserung der Benutzerfreundlichkeit.