Universität Bremen

MaMBA Projekt: Erfassung von Gaskonzentrationen in einem extraterrestrischen Habitat

Softwareentwurf zur Auslese von Peripheriegeräten

1 Ausgangssituation und Aufgabenstellung

Es soll eine Software implementiert werden, welche die von verschiedenen Peripheriegeräten erfassten Daten abfragt, verarbeitet und an ein Netzwerk übergibt. Hier wird nur der konkrete Entwurf einer Software erläutert, welche diese Daten erfasst und versendet. Der gesamte Prozess der Messung dieser Daten, sowie weitere Kommunikation im Netzwerk wird hier nicht betrachtet. Als gegeben anzunehmen sind also Peripheriegeräte und ein Netzwerk, wobei die Verbindung zu beiden nicht als konstant oder fehlerfrei beziehungsweise gegen Abbrüche gesichert angenommen werden kann. Ausfälle der Verbindung zum Netzwerk, sowie zu einzelnen oder mehreren Peripheriegeräten müssen berücksichtigt und behandelt werden.

Ausgeführt wird die Software auf einem Mikrocontroller, welcher mit WLAN in das zur Verfügung stehende Netzwerk eingebunden ist, genutzt wird die Programmiersprache *Micropython*. Alle Peripheriegeräte die über eine I^2C Schnittstelle verfügen werden mit Hilfe dieser an den Mikrocontroller angebunden, welcher hierbei als Master agiert, alle Peripheriegeräte agieren als Slaves. Anders angebundene Peripheriegeräte benötigen je einen Anschluss an den Mikrocontroller, funktionieren aber aus Sicht der Software analog zu den Peripheriegeräten mit I^2C Schnittstelle. Alle auszuführenden Aufgaben müssen in der Software auf dem Mikrocontroller enthalten sein, konkret sind die folgenden Aufgaben zu bewerkstelligen:

1. Verbinden mit dem gegebenen WLAN Netzwerk
2. Verbinden mit dem MQTT Broker
3. Absetzen von Aufträgen zum Messen an die Peripheriegeräte
4. Auslesen von den angeforderten Messdaten
5. Senden der ausgelesenen Daten mit Hilfe des MQTT Protokolls an das gegebene Netzwerk

6. Behandlung von Fehlerfällen

Der Ausfall des Mikrocontrollers, beziehungsweise des Steuerprogramms, also ein Stopp der Ausführung darf nur dann auftreten, wenn dem Mikrocontroller die Betriebsspannung entzogen wird. In allen anderen Fällen muss das System weiter funktionieren.

2 Softwareentwurf

Folgend sollen nun die im vorigen Abschnitt aufgeführten Punkte bezüglich der zu bewerkstelligen Aufgaben konkretisiert und ausgeführt werden, beginnend mit dem ersten Punkt.

2.1 Booten, Verbindungen (1. und 2.)

Zu Beginn, also während des Bootvorgangs des Mikrocontrollers, muss eine Verbindung mit dem Netzwerk über eine WLAN Verbindung hergestellt werden. Des Weiteren muss der Mikrocontroller auf eine lange Betriebszeit vorbereitet werden, entsprechende Einstellungen müssen also getätigt werden. Die Einstellungen sind weitestgehend hardwarespezifische (ESP32) oder softwarespezifische (Micropython) Einstellungen, weniger allgemeiner Natur. Das Aktivieren des *Garbage Collectors (GC)* ist hierbei die wichtigste Aufgabe, so ist gewährleistet, dass keine Speicherüberläufe auftreten. Im späteren Verlauf des Steuerprogramms muss der GC mehrfach aufgerufen werden, um die Speichernutzung zu optimieren. Andere konkrete Aufgaben sind das Einrichten von peripherieabhängigen Fehlermeldungen, zur besseren Behandlung von Fehlern.
Nun kann versucht werden die Verbindung zum MQTT Broker herzustellen. Hierfür muss allerdings eine WLAN Verbindung bestehen, es muss also eine Variable existieren, welche den aktuellen Zustand der WLAN Verbindung hält. Bei gescheiterter WLAN Verbindung bricht ebenfalls die Verbindung zum MQTT Broker ab.

2.2 Messen, Auslesen und Senden (3., 4. und 5.)

Softwareseitig müssen alle Peripheriegeräte als Objekt initialisiert werden. Dafür stehen verschiedene Bibliotheken bereit, diese werden später genauer erläutert. Eine solche Initialisierung bedarf eines gemeinsamen I^2C Objekts, da der Mikrocontroller, auf dem die Software ausgeführt wird, als Master agieren soll (es kann hier angenommen werden, dass alle Peripheriegeräte unterschiedliche Adressen haben), Peripheriegeräte ohne I^2C Schnittstelle werden entsprechend anders initialisiert. Zwar ist das genaue Prinzip der Messung und die exakten Peripheriegeräte nicht relevant, eine Betrachtung verschiedener Fälle ist aus Sicht des Steuerprogramms jedoch sehr wichtig. In einer sich immer wiederholenden Schleife muss das Steuerprogramm drei verschiedene Arten von Peripheriegeräten

ansprechen:

- Peripheriegeräte mit I²C und Registern
- Peripheriegeräte mit I²C und ohne Register, solche die nur einen ROM besitzen
- Peripheriegeräte ohne I²C und ohne Register und ohne ROM

Je nach konkretem Anwendungsfall beziehungsweise konkreter Auswahl von Peripheriegeräten kann diese Liste beliebig erweitert werden, hier genügt aber die Behandlung der drei genannten Fälle. Im ersten Fall muss das Steuerprogramm einen Befehl absetzten, welcher eine Messung veranlasst. Diese wird dann in Registern gespeichert und kann bei belieben ausgelesen werden, die Rückgabe des auslesenden Befehls ist also der Messwert. Im zweiten Fall messen die Peripheriegeräte konstant, nach dem der I²C Bus initialisiert wurde. Das Steuerprogramm fragt also immer nur den aktuellen Wert im ROM des Peripheriegerätes ab. Im letzten Fall muss ein Befehl zum Start einer Messung gesendet werden. Es wird eine Messung ausgeführt und danach müssen die einzelnen Messwerte ausgelesen werden (unter Umständen mit separaten Befehlen, wenn mehrere Werte gemessen werden).

Das exakte Ansprechverhalten der Peripheriegeräte ist in den dazugehörigen Bibliotheken definiert und implementierungsabhängig. Wie genau in den jeweiligen Bibliotheken dann die digitalen Werte in für Menschen lesbare Dezimalwerte gewandelt wird ist abhängig von dem Peripheriegerät und vom Hersteller meist definiert. Wenn nun die ausgelesenen Daten vorliegen, so müssen diese versendet werden. Dafür wird eine gesonderte Funktion definiert. Diese sendet gegebene Daten an ein gegebenes Thema des MQTT Brokers.

2.3 Fehlerfälle (6.)

Nun sollen die wichtigsten Fehlerfälle behandelt werden, beziehungsweise solche die von dem Steuerprogramm auf dem Mikrocontroller abgefangen und behandelt werden können. Ein Peripheriegeräte kann ausfallen, weil beispielsweise die Hardware defekt ist, die Leitung zum Mikrocontroller defekt ist, auf Grund von Querempfindlichkeiten oder anderen Gründen. Diese Fehler kann das Steuerprogramm nicht beheben, ein erneutes Abfragen von Messwerten (in den ersten beiden Fällen) nach dem obigen Muster wirft Fehlermeldung, welche, wenn sie behandelt werden, das gesamte Programm abstürzen lassen. Solch ein Fall darf nicht eintreten, es muss also in jeder Iteration der vorig erwähnte Schleife mit einem solchen Fehler gerechnet werden und das Abfangen der entsprechenden Fehlermeldung muss implementiert sein. In der nächsten Iteration muss wieder versucht werden eine Verbindung zu dem vorig nicht erreichbaren/ausgefallenem Peripheriegerät hergestellt zu werden. Bei nicht Erfolg muss in jeder folgenden Iteration ein neuer Versuch unternommen werden, so kann ein defektes Peripheriegerät problemlos ausgetauscht werden, ohne

die Funktion des Gesamtsystems zu gefährden.

Wenn die Daten nun per WLAN an den MQTT Broker gesendet werden sollen und die betreffende Variable keinen (validen) Wert hält, muss hier beachtet werden, ob das Peripheriegerät angemeldet ist. Dies kann über das Setzen einer Variable geschehen. Wenn keine Anmeldung existiert, werden an das entsprechende Thema auch keine Daten gesendet. Während des Sendens der Daten kann es zu einem Ausfall des WLANs kommen. In diesem Fall muss auch dieser Fehler abgefangen werden. Nach Abfangen des Fehlers sollte dann sofort versucht werden die Verbindung zum WLAN wieder herzustellen. Es ist nicht ausreichend wenn dieser Versuch erst in der nächsten Iteration des Steuerprogramms unternommen wird, da in der verstrichenen Zeit bereits kritische Gaskonzentrationen von der Peripherie gemessen werden können.
Wenn die Verbindung zum WLAN abbricht, so bricht die Verbindung zum MQTT Broker ebenfalls ab. Es muss also nicht nur direkt versucht werden die Verbindung zum WLAN wieder herzustellen, sondern auch die Verbindung zum MQTT Broker muss sofort wieder versucht werden herzustellen.

3 Bibliotheken

Betrachtet werden hier die Bibliotheken oder auch Treiber der verwendeten Peripheriegeräte. Zum einen sind hier der O_2 und der CO Sensor, die Luft- und Feuchtigkeitssensoren, der Drucksensor und der CO_2 Sensor zu erwähnen. Im Folgenden wird auf die Art der Bibliothek, nicht auf die konkrete Implementierung eingegangen.

3.1 O_2 und CO Sensor

Diese Sensoren stellen nur ein ROM und eine I^2C Schnittstelle bereit. Die Messung wird mit der von Master generierten Startbedingung gestartet, danach kann dann immer mit einer lesenden Abfrage für die Adresse des Sensors ein aktueller Messwert ausgelesen werden. Dieser ist in Steps; mit einem vom Hersteller angegebenen Koeffizienten kann dieser Wert in einen Dezimalwert umgerechnet werden.

3.2 Luft- und Feuchtigkeitssensoren

Für diese Sensoren existiert bereits ein Treiber in Micropython, die *DHT22* Bibliothek. In der implementierte Bibliothek werden die Methoden der *DHT22* Bibliothek untergebracht, um eine besserer Lesbarkeit im Steuerprogramm zu ermöglichen. Die Luft- und Feuchtigkeitssensoren verfügen über keine I^2C Schnittstelle und lesen Werte bei gesendeten Befehl via dem *One-Wire-Protocoll* aus.

3.3 Drucksensor und CO_2 Sensor

Für diese Sensoren werden proprietäre Treiber verwendet, eine genauer Erläuterung kann diesen oder den Datenblättern der Sensoren entnommen werden. Beide verfügen über Register und eine I^2C Schnittstelle, es können also konkrete Befehle, beispielsweise das Ändern des Messintervalls, entgegengenommen werden und hardwareseitig umgesetzt werden.