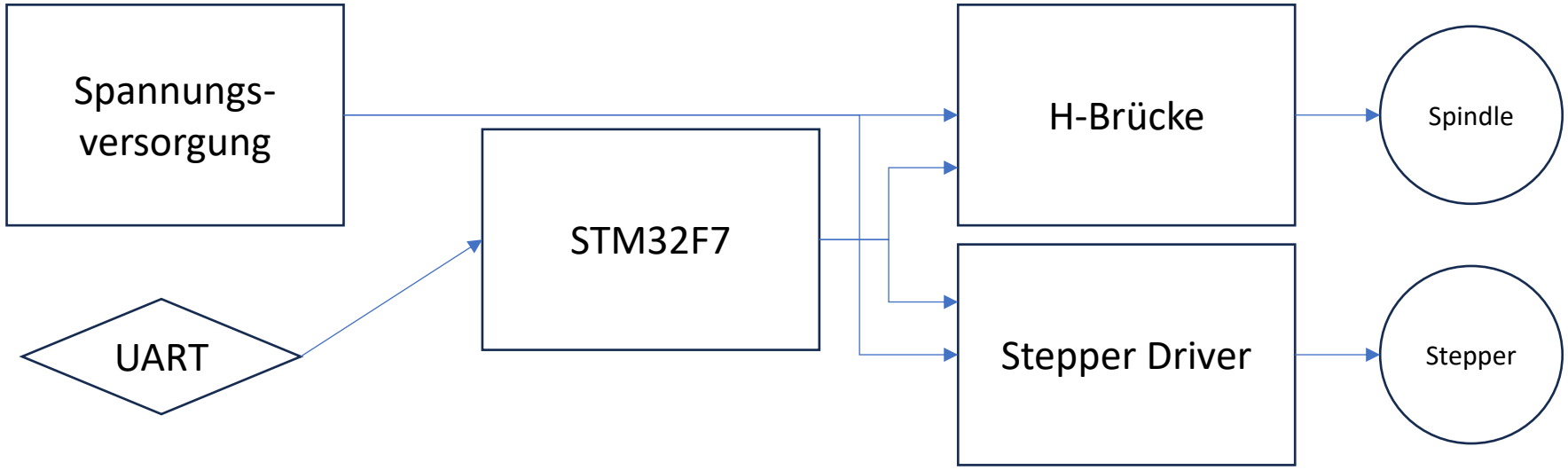
BALLUFF

Einordnung der Aufgabenstellung

Grobe Architektur

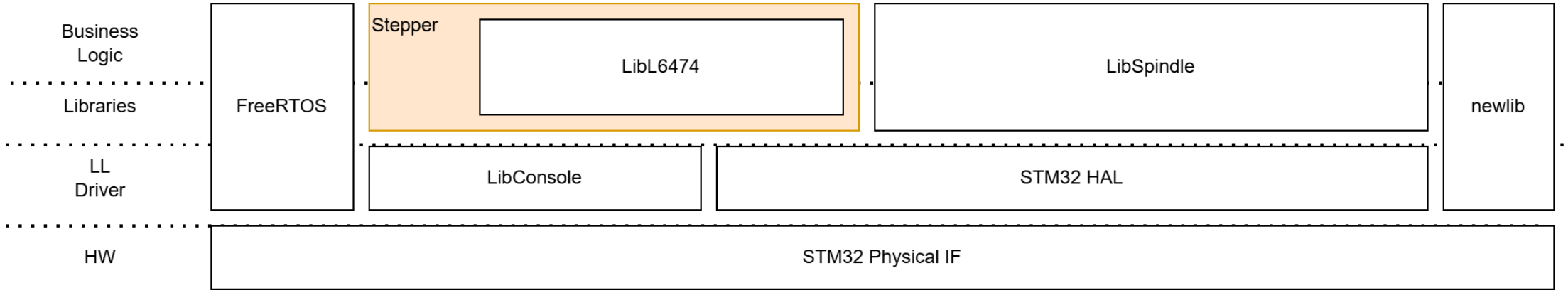
DHBW

Block Diagramm HW



Grobe Einordnung der SW-Architektur

- Der orangene Kasten ist die Primäraufgabe
- Die Libraries müssen eingebunden werden aber sind bereits implementiert



Was wird gestellt, was nicht?

- Alle Bibliotheken werden bereitgestellt
  - LibSpindle
  - LibL6474
  - LibRtosConsole
- Es muss ein Controller für den Stepper integriert werden, der aber LibL6474 einbetten soll, um die Kommunikation zu vereinfachen
- Ein Testaufbau mit Schrittmotor wird pro Gruppe zur Verfügung gestellt, Ein Gesamtaufbau pro Kurs wird ebenfalls für Tests zur Verfügung gestellt.

Sonderfall: HAL autogenerierter Code

- Möglichst schnell in eigene C- und H-Files mit der Business Logik, da die generierten Files sehr stark im LL-Bereich angesiedelt sind
- Die AutoGen-Mimik war historisch nicht immer stabil und je nach Anwendung gab es Code-Verluste
- Das File wird sehr schnell sehr unübersichtlich
- Eine Austauschbarkeit wird erschwert, wenn alles im main.c der HAL implemenitert wird!

Mögliche Lösungsansätze für die PWM-Generierung des Steppers

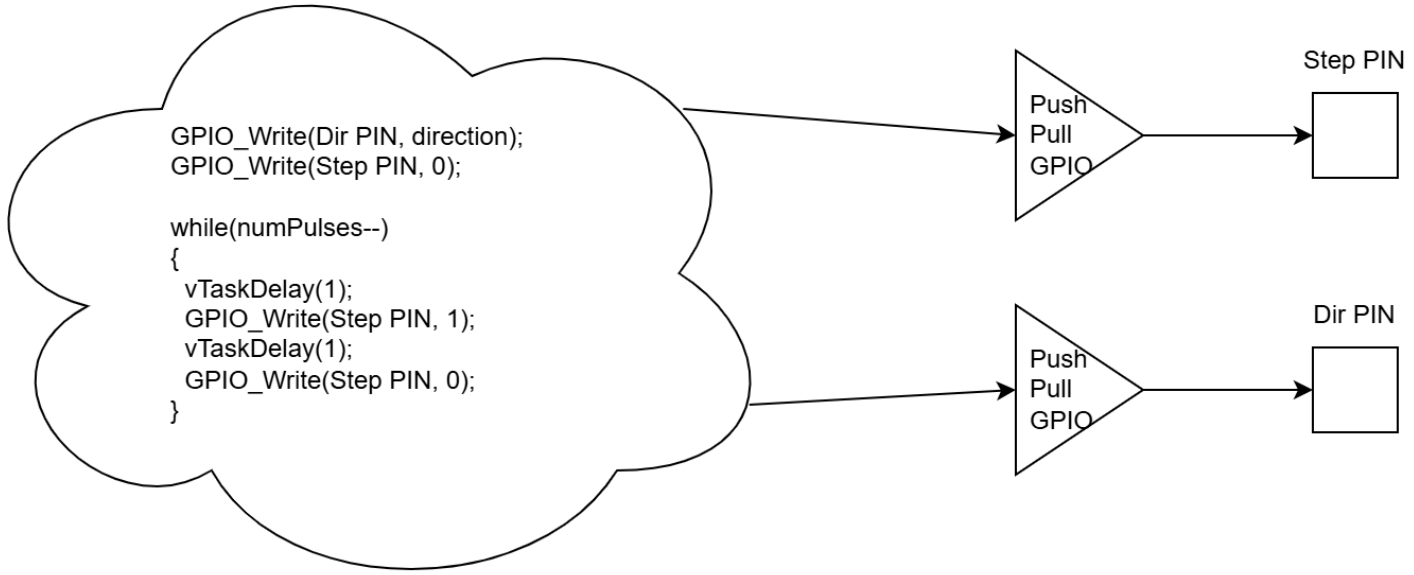
Grobe Skizzierung der Ansätze und ein paar Hinweise

GPIO-Lösung

Easy Going

Direkteste und einfachste Lösung, die keine Asynchronen Vorgänge erlaubt und eine

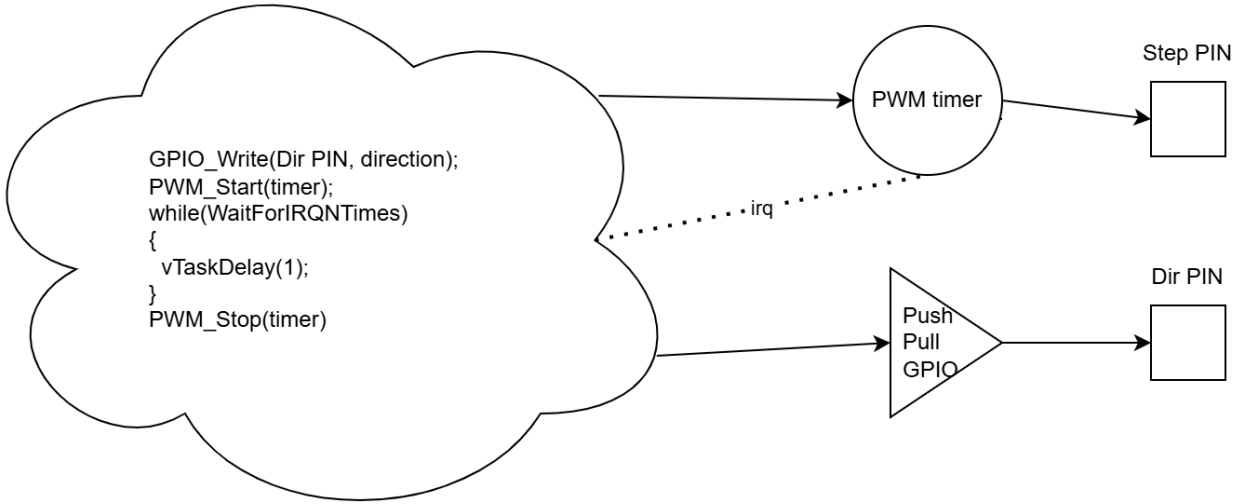
Geschwindigkeitsregelung unmöglich macht. Reicht für eine Minimalvariante der Bewegung aus



PWM-Only-Lösung

Zwischending mit reiner PWM

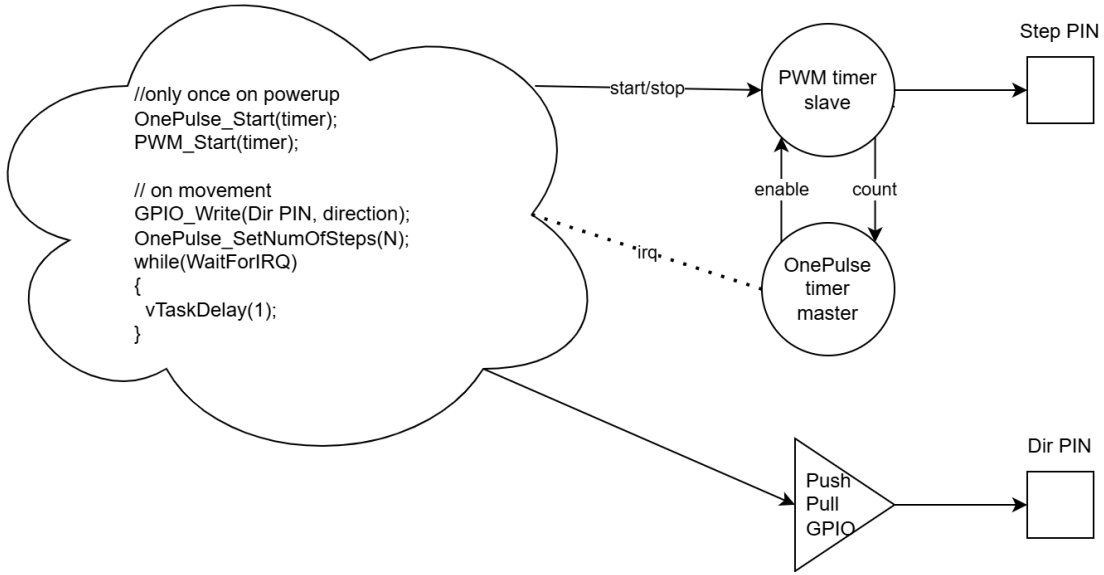
Vertretbare Lösung, die aber etwas aufwendig im Detail sein kann. Erlaubt asynchrone Vorgänge und Geschwindigkeitsregelung aber kann bei der genauen Positionsregelung problematisch werden und ggf. auch je nach Konfiguration einen großen Verwaltungsaufwand mitbringen, um die korrekte Position beizubehalten



All in HW PWM-Lösung

Exakte PWM mit automatischer Zählung durch HW

Sicherste Lösung für die exakte Zahl an Pulsen. Zusätzlich eignet sich diese Lösung perfekt für die Geschwindigkeitsregelung, da lediglich der PWM timer geändert wird. Ein Asynchroner Betrieb ist einfach Möglich, da man lediglich auf 2 Interrupt (eigentlich nur einer im besten Fall) reagieren muss. Der OnePulse Timer schaltet den PWM timer exakt nach der gegebenen Zahl an Pulsen ab!



Anmerkung zu der GPIO-Lösung

- Zeitverhalten:
  - Dies ist die einfachste aller Lösungen, allerdings auch die Ungenauste. Sie limitiert die maximale Pulsfrequenz und kann nur synchron zur Ausführung implementiert werden**.
  - Mit dieser Lösung erzielt man lediglich ~500Hz sodass nur etwa 3% der maximalen Geschwindigkeit gefahren werden kann!

Anmerkung zu den PWM-Lösungen

- Drehzahlsteuerung:
  - Die Berechnung der Taktfrequenz erfolgt in der Regel mit einem Prescaler- und einem Period-Register. Es gibt somit ein Optimierungsproblem mit zwei Größen, um den geeignetsten Arbeitspunkt zu bestimmen.
- PWM-All in HW:
  - Bei der Master-Slave Lösung muss man beachten, dass der OnePulse Timer lediglich 16 Bit Zählen kann, somit ist die Zahl an Pulsen auf 65535 Pulse limitiert. In einem finalen Interrupt muss also geprüft werden, ob damit bereits alle Pulse generiert wurden oder ob man erneut den OnePulse Mode mit den verbleibenden Pulsen startet!