## Inleiding

## De regels toegepast op de code

### Regel 1

Restrict all code to vert simple control flow constructs.

Hierbij worden de voorbeelden gegeven: Goto statements, setjmp longjm constructs of (inderecte) recursie. De huidige code op zowel de computer, arduino en de raspberry pi gebruiken geen goto, setjmp of longjmp statements. Er worden ook geen functies de zichzelf aanroepen of functies bovenliggende functies weer opnieuw aanroepen gebruikt. In de code hebben meerdere functies wel meerdere returns. Dit kan de control flow iets moeilijker maken om te lezen. Dit gebeurt meerdere malen voor errors wat als een uitzondering wordt genoemd in 10 rules for developing safety-critical code (Holzmann 2006). Zie bijvoorbeeld httpmessage() in abnf.c hier wordt op regel 52 gecontroleerd of de startline correct is. Zo niet wordt er direct een false terug gegeven.

### Regel 2

Give all loops a fixed upper bound.   
Omdit te controleren zijn alle while, for en do-while loops gecontrolleerd in de code.   
Do-while loops zijn niet gebruikt in de code.  
While loops worden op meerdere plekken gebruikt.

1. In laptop.py wordt een while true loop gebruikt. Deze loop heeft geen fixed upperbound en zal altijd blijven draaien. Deze keuzen is gemaakt omdat dit programma tot dat je hem handmatig sluit altijd moet blijven herhalen. In 10 rules for developing safety-critical code word hier ook een uitzondering voor gemaakt en moet juist worden aangegeven dat deze loop nooit kan eindigen. Deze loop kan nooit eindigen en voldoet dus aan de regels.
2. In abnf.c, cbuffer.c en token.c worden while loops gebruikt om door tokens en buffers te lopen van onbepaalde groten. Deze loops kunnen in theorie eeuwig blijven door gaan als de lijst waar door heen geittereerd wordt niet eindigt. Dit zou kunnen worden opgelost door een counter toe te voegen die na een bepaald limiet de while loop termineerd. Dit is niet gedaan en voldoet dan ook niet aan de regels.

For loops worden op veel plekken gebruikt.

1. In runPi.py de code op de raspberry pi worden 3 for loops gebruikt zonder expliciete upper bound. Deze loops kunnen in theorie eeuwig blijven draaien en zouden met een extra check een upperbound kunnen krijgen. Dit is niet geimplementeerd en voldoet dus niet aan de regel.
2. In abnf.c en Arduino\_server.ino worden meerdere for loops gebruikt met een upperbound. Deze for loops voldoen dan ook aan de regels.

Regel 2 wordt niet volledig gevolgd in de huidige implemtatie. Dit is redelijk makkelijk op te lossen door extra checks toe te voegen die de loops termineren na een bepaald aantal itteraties.

### Regel 3

Do not use dynamic memory allocation after initialization.

Deze regel wordt niet gevolgd in deze implementatie. De circulaire buffer gebruikt malloc om zichzelf op de correcte grote te innitializeren nadat het programma is begonnen met draaien. Ook in calculations.c wordt malloc gebruikt voor de zelfde reden.   
Calculations.c kan relatief makkelijk worden omgezet naar een static memory allocation. De huidige implementatie en eisen voor de circulaire buffer maken het lastiger om het geheugen van de circulaire buffer statisch te bepalen omdat het niet bepaald is wat de maximale grote van de circulaire buffers mag zijn.

### Regel 4

No function should be longer than what can be printed on a single sheet of paper.

De meeste functies binnen de huidige implementatie vallen makkelijk binnen dit limiet. Echter zijn er een paar functies die veel groter zijn dan dit limiet.

handleRequest in cserver.c  
De handlerequest functie bestaat uit een paar lijnen die ervoor zorgen dat de buffers en calculaties worden aangemaakt. De rest van de functie bestaat uit een grote switch case die de berichten van de ABNF parser afhandelen. Deze switch case kan worden opgedeeld in aparte fucnties om zo aan deze eis te voldoen.

loop() in Arduino\_server.ino  
De loop functie is erg groot geworden door de switch case die de requests van de server afhandeld. Deze switch case zou in zijn eigen functie kunnen worden gestopt om zo de loop functie Kleiner te maken. Er is voor gekozen om dit niet te doen omdat de impact op de leesbaarheid minimaal zou zijn.

//TODO Dit stuk herschrijven en afmaken

### Regel 5

The code’s assertion density should average to minimally two assertions per function.

Asserts worden niet gebruikt in de code. De code voldoet dan ook niet aan deze regel. De code qualiteit zou significant kunnen worden verbeterd door het gebruik van asserts.

### Regel 6

Declare all data objects at the smallest possible level of scope.

### Regel 7

Each calling function must check the return value of nonvoid functions, and each called function must check the validity of all parameters provided by the caller.

### Regel 8

The use of the preprocessor must be limited to the inclusion of header files and simple macro definitions

### Regel 9

The use of pointers must be restricted.

### Regel 10

All code must be compiled, from the first day of development, with all compiler warnings enabled at the most pedantic setting available.