Checklist Code Review v.1

Datum Review: \_\_29-1-2019\_\_\_

Paraaf Reviewer: \_David\_\_\_\_\_

Datum Response: \_\_\_30-1-2019

Paraaf Developer: \_\_Andre\_\_\_\_

## Naam Developer: \_\_Andre\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Naam Reviewer: \_\_David\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Project / Opdracht: \_\_klokcode project 5/6\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## Programmeertaal: \_\_C++\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| # | Type | Observatie | G | V | M | Commentaar Reviewer | Respons Ontwikkelaar |
| 1 | RUN | De code compileert | X |  |  |  |  |
| 2 | DOC | Er zijn duidelijke commentaarblokken bij iedere functie |  | X |  | Mag wat meer | Moet ik meer eigen maken, voor zowel mezelf als anderen |
| 3 | STY | Indentaties zijn netjes en consequent | X |  |  |  |  |
| 4 | STY | Regellengtes zijn korter dan 80 karakters | X |  |  |  |  |
| 5 | STY | Klassenamen beginnen met een hoofdletter, verder of overal CamelCase of overal Pothole\_case. | X |  |  |  |  |
| 6 | STY | Functienamen en variabelenamen beginnen met een kleine letter, verder of overal camelCase of overal potholeCase | X |  |  |  |  |
| 7 | ARC | Streams en connecties worden netjes afgesloten, bij voorkeur automatisch door een destructor of een “with” statement. |  |  |  | Niet van toepassing |  |
| 8 | STR | Er zijn geen deadlocks in de code | X |  |  |  |  |
| 9 | DOC | Commentaarregels voegen iets toe aan onderhoudbaarheid |  | X |  | Mogen stuk meer comments bij | Moet ik meer eigen maken |
| 10 | STY | De code maakt gebruik van herhalingslussen en arrays om uitschrijven van herhaalde activiteiten te voorkomen. | X |  |  | Eerst had je het niet, maar goed je code verbeterd |  |
| 11 | STY | Elk getal dat meer dan 1 x voorkomt, wordt gedefinieerd en gebruikt als constante of (indien goed gemotiveerd) macro. | X |  |  |  |  |
| 12 | STY | Delays en blocking waits in Arduinocode worden vermeden omdat ze de code unresponsive maken. 1 delay in de main loop is alles wat nodig is. Alle andere vertragingen worden gerealiseerd door 1 x per cyclus de huidige tijd te meten en vervolgens in elke cyclus tijden van events te vergelijken en actie te ondernemen afhankelijk van het de elapsed time sinds een event. | X |  |  | Gewerkt met millis |  |
| 13 | STY | Fysieke entiteiten en hun adressen of I/O pins krijgen betekenisvolle symbolische namen, evt. vergezeld van een nummer indien arrays worden gebruikt. | X |  |  |  |  |
| 14 | STY | De code maakt gebruik van herhalingslussen en arrays om uitschrijven van herhaalde activiteiten te voorkomen. | X |  |  |  |  |
| 15 |  | Code is zo geschreven dat hij in delen kan worden getest. | X |  |  | Elk deel van de klokweergave heeft een eigen functie. |  |
| 16 |  | Commentaar is kort, bondig en helder, bij voorkeur (maar niet verplicht) engelstalig. Het is geen herhaling van wat uit de broncode al duidelijk is. |  |  | X | Er is weinig commentaar toegevoegd |  |
| 17 |  | Commentaar is kort, bondig en helder, bij voorkeur (maar niet verplicht) engelstalig. Het is geen herhaling van wat uit de broncode al duidelijk is. | X |  |  |  |  |

Lijst van Jacques die ingebed kan/moet worden in de tabel:

1. De code maakt gebruik van heldere en consequente benamingen, bij voorkeur (maar niet verplicht) engelstalig:
   1. Zelfstandige naamwoorden voor getal, string, char, samengestelde variabelen, objects en classes: counter, opponentName, separator, memberList, inputDriver, Driver respectively.
   2. Classes beginnen met een hoofdletter. Functions, methods, variables, instances en properties beginnen met een kleine letter.
   3. Werkwoorden + evt. lijdend voorwerp voor functions en methods : multiply, computeAverage, sortByName.
   4. Predicaten voor booleans: isMaximum, hasMoreCharacters, canBeActivated
2. De code maakt gebruik van functies met parameters om duplicatie van vrijwel identieke stukken code te voorkomen.
3. De code maakt gebruik van herhalingslussen en arrays om uitschrijven van herhaalde activiteiten te voorkomen.
4. Functies gebruiken geen globale variabelen maar parameters en return values, tenzij hier een heel goede reden voor bestaat, ter beoordeling van docent
5. Commentaar is kort, bondig en helder, bij voorkeur (maar niet verplicht) engelstalig. Het is geen herhaling van wat uit de broncode al duidelijk is.
6. Grote lappen code zijn gesplitst in modules.
7. Elke module behalve de main module bestaat uit een <moduleName>.h en een <moduleName>.cpp (of .c) file.
8. In de header staan uitsluitend declarations en macro definitions: class declarations, external variables en external functions (prototype) en #define's
9. Elk getal dat meer dan 1 x voorkomt, wordt gedefinieerd en gebruikt als constante of (indien goed gemotiveerd) macro.
10. Fysieke entiteiten en hun adressen of I/O pins krijgen betekenisvolle symbolische namen, evt. vergezeld van een nummer indien arrays worden gebruikt.
11. Alle berekeningen, opslagstructuren etc. worden zero-based geadresseerd, tenzij in een user interface, waarbij omrekening onmiddelijk na invoer of voor uitvoer plaatsvindt.
12. Delays en blocking waits in Arduinocode worden vermeden omdat ze de code unresponsive maken. 1 delay in de main loop is alles wat nodig is. Alle andere vertragingen worden gerealiseerd door 1 x per cyclus de huidige tijd te meten en vervolgens in elke cyclus tijden van events te vergelijken en actie te ondernemen afhankelijk van het de elapsed time sinds een event.
13. Sensors en actuators worden bij voorkeur ingekapseld in objecten.
14. Er wordt gestreefd naar uniforme, dunne, stabiele interfaces voor alle gebruikte classes en modules.
15. Er wordt gestreefd naar uniforme behandeling van uiteenlopende sensors, actuators en, in het algemeen, entities, door gebruik te maken van interface inheritance en polymorphism.
16. Er wordt strak onderscheid gemaakt tussen booleans en integers, tenzij de student er expliciet blijk van geeft dit onderscheid consequent en op goede gronden te laten vallen: if (counter == 0) is altijd goed, if (!counter) alleen als de student er blijk van geeft deze keuze overtuigend te kunnen onderbouwen, ter beoordeling van de docent
17. Constructies zoals if (isLargestVoltage == true) worden vermeden. Gewoon if (isLargestVoltage), geen contaminaties dus.
18. Daar waar niet-triviale ontwerpkeuzen werden gemaakt, blijkt uit het commentaar waarom, niet alleen uit een apart ontwerpdocument (dat onveranderlijk veroudert of zoekraakt).
19. Alle code kan door de student worden uitgelegd, ook van andere teamleden of afkomstig van Internet of uit boeken.
20. Alle code van derden is voorzien van bronvermeldingen in comments.
21. Keuze om al dan niet library code te gebruiken zijn onderbouwd.
22. Coding styles binnen een project mogen verschillen, maar niet binnen 1 sourcefile.
23. "Geen tijd" == "Geen excuus". Deze zaken kosten geen tijd, ze leveren tijd op.
24. Regressietests (unittests) maken deel uit van de opgeleverde code en kunnen tijdens assessments worden gedemonstreerd.
25. Code is zo geschreven dat hij in delen kan worden getest.
26. Modules zijn hierarchisch gerangschikt, waarbij toepassingsspecifieke modules gebruik kunnen maken van algemene modules, maar nooit omgekeerd.
27. Code gegenereerd door tools als Visual Studio, dient of te worden vermeden, of geheel te kunnen worden uitgelegd.
28. Nodeloos gebruik van de STL (incl. String) op eenvoudige single board computers zoals de Arduino Uno wordt vermeden met het oog op geheugengebruik en performance.
29. Class diagrams, state transition diagrams, component diagrams en sequence charts zijn prima, use case diagrams mogen met mate, control flow charts zijn verboden omdat ze gespecialiseerd zijn in het weergeven van spaghetticode, die daaraan z'n naam te danken heeft.