|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objecten | Taaksoort | Periode | Deadline | Prioriteit |
| InstelControl | Intern, Demand Driven |  | 30ms | 2 |
| TijdInstelControl | Intern, Demand Driven |  | 30ms | 2 |
| AlarmInstelControl | Intern, Demand Driven |  | 30ms | 2 |
| OperationControl | Intern, Demand Driven | 1000ms | 1000ms | 3 |
| NecReceiver | Intern, Demand Driven |  | 100us | 1 |
| PauseDetector | Intern, Demand Driven |  | 100us | 1 |
| Display | IO, Demand Driven |  | 200ms | 3 |
| Buzzer | IO, Demand Driven |  | 2100ms | 3 |
| IR-Receiver | IO, Demand Driven |  | 100us | 1 |

De eerste versie van de taken van de alarm klok zal bestaan uit de bovenstaande taken. Ik heb vooral de use case van Design Like A Robot als uitgangspunt genomen voor de taken omdat deze voor mij ook goed werkt.

De instelcontrol, tijdinstelcontrol en alarminstelcontrol hebben alle drie een deadline van 30ms omdat ik deze best responsive wil houden maar ook weer niet te snel wil hebben.

De OperationControl heeft een deadline en periode van 1000ms (1s) omdat deze elke seconde de tijd moet updaten en moet controleren of het alarm af zal moeten gaan. Dit gebeurt niet sneller dan een elke seconde vandaar de periode. Door deze lange periode heb ik er ook voor gekozen deze taak een lagere priority te geven omdat deze veel flexibeler is met de deadline. Ten tweede zal de execution time ook veel lager liggen dus of de code na 100ms of na 300ms runt maakt niet uit.

De twee receivers hebben een deadline van 100us, dit komt doordat ik in eerste instantie de voorbeeld STDs als uitgangspunt heb genomen. Later, nadat ik dit heb aangepast naar een interrupt driven methode zal deze deadline lager kunnen zijn maar vind ik de resolutie van 100us nog steeds meer dan voldoende, daarom houd ik deze zo. Vanwege de interrupts heb ik ook geen periode meer nodig omdat ze niet meer elke 100us pollen maar puur via een interrupt demand driven werken.

De display heb ik op 200ms seconde staan. Het display heeft genoeg marge nodig om alles te kunnen laten zien. Hierdoor is ( net zoals design like a robot ) een redelijke marge gekozen voor het display.

Verder heb ik voor de buzzer een deadline van 2.1s omdat ik bij een alarm de buzzer voor 2s wil laten klinken, de extra 100ms heb ik als marge meegenomen, ik vind het niet erg als de buzzer iets langer klinkt want er zal wellicht 2s tussen de toggles zitten maar die kosten ook wellicht nog wat tijd. Ik heb geen periode omdat ik verwacht de PWM pins aan te sturen van de microcontroller of een piezo buzzer te gebruiken. Mocht ik een GPIO gebruiken zonder PWM dan zal ik wellicht een periode van ~500us hebben om de pin te toggelen.

In mijn eerste ontwerp had ik de IR receiver als taak opgenomen omdat dit een boundary device was dat gepolled moest worden. Later ben ik erachter gekomen dat dit niet nodig is omdat de IR receiver alleen door de SignalPause task gebruikt wordt. Hierdoor kon ik control cohesie toepassen en uiteindelijk na het overstappen naar interrupts was deze klasse helemaal niet meer nodig.

Vanuit de use cases is te zien dat InstelControl het hoofd menu bedient en TijdInstelControl en AlarmInstelControl beide submenus zijn. Hierdoor is er sprake van control cohesie waarbij InstelControl ( hoofdmenu ) de master is van de submenus. Vanwege deze cohesie heb ik gekozen deze drie taken samen te voegen in een centrale instel taak.

Voor mij versimpelt dit de taken en vermindert dit ook context switching doordat de taken erg afhankelijk zijn van elkaar. De gebruiker kan immers ook nooit twee instellingen tegelijk aanpassen en moet altijd door het hoofdmenu.

De taken zijn nu als volgt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Objecten | Taaksoort | Periode | Deadline | Prioriteit |
| InstelControl | Intern, Demand Driven |  | 30ms | 2 |
| OperationControl | Intern, Demand Driven | 1000ms | 1000ms | 3 |
| NecReceiver | Intern, Demand Driven |  | 100us | 1 |
| PauseDetector | Intern, Demand Driven |  | 100us | 1 |
| Display | IO, Demand Driven |  | 200ms | 3 |
| Buzzer | IO, Demand Driven |  | 2s | 3 |

Hieruit is de applicatie voor de alarm klok erg versimpelt en zijn de main taken duidelijk te zien. Verder zal dit als het goed is ook geen problemen opleveren doordat wanneer instelling aangepast worden de operations verder gaat als aparte taak.