Firmware Design

Beschrijving voor de Firmware van de SenseBox

Luc Appelman (640696@student.inholland.nl)

28 juni 2022

Inhoudsopgave

1 Inleiding

In dit document worden de relaties van de klassen beschreven voor de SenseBox. De klassen zullen op een *higher level* worden besproken om het document kort te houden.

Het doel van de SenseBox is om iedere minuut data van verschillende sensoren uit te lezen en deze te versturen naar een IoT (Internet of Things) Platform waarop de data uitgelezen kan worden en opgeslagen wordt in een lokale database. Om de gemeten data te versturen wordt een WiFi verbinding gebruikt.

Voor het project wordt er gebruik gemaakt van een RTOS (Real Time Operating System), ESP-IDF. De UML diagrammen zijn ook met de bestaande structuur van dit OS in gedachten gemaakt.

1.1 Schrijfwijze

Om de resulterende code zo overzichtelijk mogelijk te houden hebben we als team een aantal regels ingesteld hoe we variabelen, klassen etc. kunnen onderscheiden. Om eventuele verwarring te voorkomen, voor bijvoorbeeld een volgend team dat aan het project gaat werken, hebben we besloten om deze regels ook toe te passen in het UML ontwerp van de code.

- In klassen krijgen alle member functies en properties een *m* als voorvoegsel. Als het een private member of property is krijgt deze *m_* als voervoegsel. Andere voervoegsels komen na de *m* (maar voor het liggende streepje als het om een private property gaat).
- Iedere variabele start met een hoofdletter, tenzij er geen voorvoegsel is.
- Argument van een functie: een argument start altijd met een *a*.
- Pointer: de variabele krijgt een *p* als voervoegsel.
- Reference: de variabele krijgt een *r* als voorvoegsel.
- Klasse: de variabele krijgt een hoofdletter *C* als voorvoegsel.
- Enum/Struct/Datatype: Gebruikt pascal casing

2 Het opstart proces van de SenseBox

Het systeem kent vier statussen: Init, Active, SoftReset en Stopped. Alleen wanneer de applicatie aan het opstarten is bevindt het systeem zich in de Init status. Na volledig opgestart te zijn gaat het systeem naar de Active status. De andere twee statussen worden in het volgende onderdeel uitgelegd.

Het opstarten van de SenseBox gaat in meerdere stappen die één voor één uitgevoerd worden. Het hele process voor het opstarten staat in het bestand *app.cpp*.

1. De SD kaart initialiseren
2. Het log bestand wordt geïnitialiseerd

3. De klasse voor het configuratie bestand wordt geïnitialiseerd en het config bestand wordt uitgelezen van de SD kaart
4. Er wordt verbinding gemaakt met het WiFi netwerk dat ingesteld staat op de SD kaart
5. De NTP functionaliteit van ESP-IDF wordt geïnitialiseerd om de tijd op te halen voor de logs
6. Er wordt verbinding gemaakt met de MQTT server op de virtual machine.
   (a) Er wordt verbinding gemaakt met het provision account, en er wordt een provisioning request gemaakt
   (b) Er komt een response van de server met de status van het request, als dit met succes gelukt is wordt er opnieuw verbinding gemaakt met de inloggegevens die bij het apparaat hoort
7. Alle sensoren worden geïnitialiseerd
8. De callbacks voor WiFi (dis)connect worden ingesteld
9. De timer voor het versturen van de metingen wordt gestart

2.1 Soft start

Als er een onherstelbare fout in de SenseBox start het systeem opnieuw op, dit wordt gedaan middels een soft start functie. De soft start functionaliteit zorgt er voor dat het alle onderdelen van het systeem gedeïnitialiseerd worden. Dit gaat in de omgekeerde volgorde van het opstarten van het systeem. Hierna wordt de functie aangeroepen waarmee het systeem opstart. Terwijl dit wordt uitgevoerd staat de applicatie dan ook in de SoftStart status.

De soft start functie wordt ook gebruikt wanneer de applicatie tijdens de Init status in fout status gaat. Echter zal de applicatie niet opnieuw de soft start uitvoeren als er een fout ontstaat tijdens het initialiseren en de applicatie zich al in de SoftStart status bevindt. De logs worden dan wel nog uitgeprint, maar hierna zal de applicatie zich in de Stopped status bevinden. De applicatie wordt dan wel nog gedeïnitialiseerd, als er nog fouten optreden tijdens het deïnitialiseren zal het systeem per direct stil gezet worden.

3 Relaties tussen de klassen

3.1 Sensoren

Iedere sensor heeft zijn eigen klasse waar de implementatie wordt voor een specifieke sensor, de klasse krijgt de zelfde naam als het modelnaam van de sensor. De implementatie van een sensor dient afgeleid te zijn van de CSensor klasse. Deze klasse bevat de basis functionaliteit die iedere sensor ten minste moet hebben. De member functies worden aangeroepen vanuit de CSensorManager klasse. Met deze klasse worden alle toegevoegde sensoren bijgehouden en kunnen de resultaten van de metingen opgehaald worden. Op deze manier is het erg makkelijk om nieuwe sensoren toe te voegen zonder dat er wijzigingen gemaakt hoeven te worden aan de rest van de code.

Iedere sensor heeft zijn eigen meet interval, hiervoor wordt een timer gestart tijdens het aanmaken van de processor. Hoe vaak een sensor meet dient in de m_InitCallback() functie van de sensor gedefineerd te worden.

Iedere keer dat er een meting gemaakt wordt wordt het resultaat hiervan opgeteld bij m_MeasurementTotal. Wanneer de resultaten opgevraagd worden aan de SensorManager zal hiervan het gemiddelde berekent worden.

3.2 WiFi en MQTT

Voor een verbinding met de MQTT server is het belangrijk dat de SenseBox altijd verbonden is met het internet. Wanneer de verbinding wegvalt/herstelt zijn er daarom callbacks gemaakt die er voor zorgen dat de MQTT verbinding ook verbroken of opnieuw gestart wordt.

3.3 Timers

De timers van ESP-IDF zorgen er voor dat functies met een precies interval of vertraging uitgevoerd worden. Tijdens het soft starten is het belangrijk dat deze gestopt en opgeruimd worden zodat deze niet nog uitgevoerd worden tijdens het soft starten. Met deze reden zijn er twee klassen aangemaakt, CTimer en CTimers.

CTimers is de eigenaar val alle timers die aangemaakt worden. CTimer is een abstractie laag, dit is het object dat alle informatie van een aangemaakte timer bevat. De CTimer klasse bevat ook een statische functie waarmee een timer aangemaakt kan worden, de functie geeft alleen de pointer van de timer terug zodat verdere bewerkingen op de nieuw aangemaakte timer gedaan kunnen worden.

Als er beschikking is tot de CTimer pointer kunnen alle bewerkingen hiermee gedaan worden. Met de CTimers klasse kan er alleen gecheckt worden of een timer bestaat aan de hand van de naam en kan een timer verwijderd worden.

3.4 File en Dir

Met de CDir en CFile klassen kan er met het filesysteem gewerkt worden. Met CDir kan een map geopend worden, waarna bijvoorbeeld alle bestanden uit de map weergegeven kunnen worden. Maar er kan ook een bestand geopend worden in de map.

Een bestand kan in één van de drie modussen geopend worden: Read, Write en Append. Read spreekt voor zich, maar bij write wordt de gehele inhoud van het bestand vervangen met de tekst in het argument van de functie. Bij Append wordt tekst in het argument toegevoegd aan het einde van het bestand.

Om de performance van de Append modus te verbeteren is er een queue aangemaakt (deze zit standaard in FreeRTOS en is iets aangepast bij ESP-IDF). Het schrijven naar de SD kaart wordt gedaan met een interval en wordt direct uitgevoerd als de queue al vol zit.

Het (de)initialiseren en de status van de SD kaart is ook toegevoegd als statische functies aan de CFile klasse.

3.5 Log en LogScope

Voor het loggen van de applicatie zijn er twee klassen toegevoegd, CLog en CLogScope. De taak van de CLog klasse is om alle logs in het zelfde format naar de stdout en de SD kaart te schrijven. Denk hierbij het formatten van de tijd maar ook de kleuren waarin het log wordt uitgeprint.

De functie van CLogScope is om log aan te maken met een tag, dit zodat we kunnen zien welk onderdeel van de applicatie iets logt. Maak een variabele aan met de naam *logger*, vervolgens kan deze variabele door het hele bestand gebruikt worden om logs met de zelfde tag te maken. De klasse heeft vier functies: mInfo(), mWarn(), mError() en mDebug(). Iedere functie heeft de zelfde argumenten als een normale printf(), maar zorgt er ook voor dat de regel ook met de juiste loglevel wordt uitgeprint.

3.6 Tijd en Log

De CTime klasse initialiseert de NTP functionaliteit ingebouwd in ESP-IDF. Hierdoor wordt de tijd regelmatig opgehaald bij een NTP server en is de tijd altijd nauwkeurig. Als de tijd eenmaal is opgehaald bij de NTP server kan de CLog klasse de tijd opvragen.

4 Aanbevelingen

Config als attributen opsturen

Momenteel worden de attributen van Thingsboard niet goed gebruikt. Attributen zou informatie moeten zijn die normaal niet/weinig veranderd. Daarom is het niet nuttig om de eerste telemetrie data op te sturen. Wat wel nuttig zou zijn, zijn de offsets en factors voor de sensoren. Vertaal in C++ het toml object ”calibratie”naar JSON en stuur dat op wanneer het systeem op start.

USB OTG

Het is niet makkelijk om bij de SD kaart te komen om de logs uit te lezen, het kan daarom handig zijn om de SenseBox te programmeren met USB OTG functionaliteit en de SD kaart hierop weer te geven. Echter zal het niet echt veilig zijn voor productie, maar tijdens testen kan dit prima.

SD kaart unmounten

Tijdens het soft starten zou eigenlijk de SD kaart moeten unmounten van het systeem. Het is nog niet gelukt om dit voor elkaar te krijgen, de functie die dit zou moeten doen volgens ESP-IDF staat wel al in de CFile klasse.

File niet sluiten na write in Append modus

Momenteel is het nodig om nadat de queue in de Append modus geleegd is het bestand direct te sluiten. Dit zou niet nodig moeten zijn, maar door een (tot nu toe) onverklaarbare reden werkt anders het loggen naar de SD kaart niet na het soft starten. Als bovenstaande punt lukt ga ik er vanuit dat dit probleem ook direct verholpen is.

Houdt er rekening mee dat het niet verstandig om bestanden onbeperkt open te laten staan. In commit 91dda72 staat hiervoor nog een methode waarbij het bestand na de ingesteld tijd automatisch sluit door de timer.

Het is niet mogelijk om meerdere keer het zelfde bestand te openen

Dit is nog slechts een assumptie en niet getest, maar waarschijnlijk zal er tegen problemen aangelopen worden wanneer er meerdere keren het zelfde bestand in de Append modus geopend wordt. Dit komt doordat CTimers de laatste zestien karakters van het pad gebruikt als key in de map.

WiFi en MQTT op lange termijn testen

Met de WiFi en MQTT kunnen nog wat problemen voortdoen. Er is nog één WiFi event waarvan ESP-IDF slecht opnieuw verbindt met het netwerk, dit event heet *disassoc*.