Project Baby-reanimatie pop

ESE

Auteurs: Len Verploegen – 647316
Thomas IJsseldijk – 673923

Semester: 3

Datum: 6-1-2023

Begeleider: Johan Korten

Inhoud

1 Inleiding

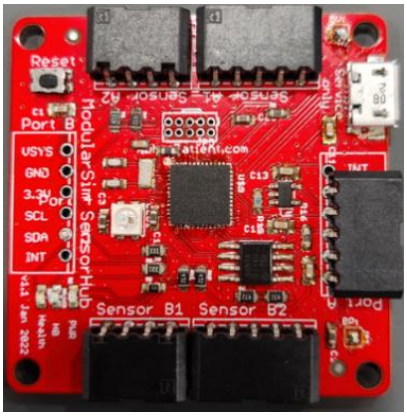
In semester 3 van de opleidingen WTB, IPO en ESE gaan vijf studenten aan de slag met het verbeteren van een basic life supportbabyreanimatiepop. Dit is het ESE-gedeelte van het tutorial-stijl verslag. Het eerste deel van het project hebben we besteed aan het uitzoeken van de sensoren en daarna het testen van de bestaande code. Vaak werkte de code niet en moest ik deze eerst aanpassen/oplossen. Hierdoor was er minder tijd over om een werkende pop op te leveren met alle werkende sensoren erin. Samen met de opdrachtgever hebben we besloten om de pop op te leveren met de 2 belangrijkste sensoren erin. Het eerste gedeelte is het meten van ventilaties doormiddel van een flowsensor, Thomas heeft dit gedeelte opgepakt. Ik ben verdergegaan met de tweede sensor voor het meten van de compressie. Hiervoor gebruik ik een time of flight sensor.

2 Hardware

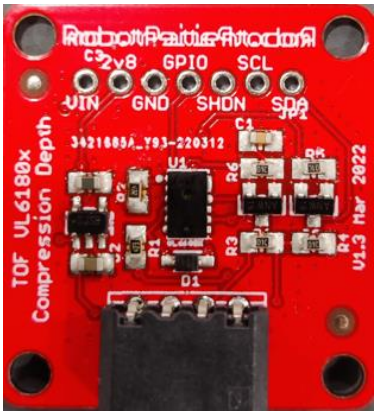
De microcontroller SAMD21 die wij gebruiken als sensor hub(figuur 1), bestaat uit een Adafruit Feather M0 Express microchip. De taal waarop deze microcontroller werkt is Arduino.

Via een 4 polige connector kan er verbinding worden gemaakt met een sensor, Hiervoor wordt I2C gebruikt. De pinnen van deze connector zijn: 3.3V, GND, SDA en SCL.

Op figuur 2 is een pcb-board te zien in het midden zit de Time of flight Sensor. De meetgegevens worden gemaakt met de Time of flight sensor, de gegevens worden daarna doorgestuurd via I2C naar de Sensor hub.



Figuur 1 Sensor hub module



Figuur 2 Time of flight sensor

3 Software

3.1 Visual Studio Code

Als code editor voor het programmeren is gekozen voor Microsoft Visual Studio Code(VScode). VScode is een algemene code editor die voor vele talen geschikt is. Voor dit project is gekozen voor VScode omdat het een flexibele werkomgeving is die aangepast kan worden met extensies. De belangrijkste extensie is de PlatformIO extensie. Andere extensies die handig zijn om te gebruiken zijn bijvoorbeeld intellisense voor C++ en de GitHub extensie.

3.1.1 Visual Studio Code installeren

Visual studio code kan op twee manieren worden geïnstalleerd.

Methode 1: downloaden via de originele website.

Methode 2: downloaden via de Microsoft store.

3.2 PlatformIO

PlatformIO is een crossplatform omgeving voor het ontwikkelen en debuggen van microcontrollers, met een focus op Arduino-achtige apparaten. In dit specifieke project wordt het gebruikt in VScode, maar het kan ook worden gebruikt in programma's zoals Clion, Eclipse, QTcreator en Visual Studio. In vergelijking met de Arduino IDE biedt PlatformIO een aantal voordelen. Ten eerste ondersteunt het zonder extra installaties een breed scala aan chips en development boards. In tegenstelling tot de Arduino IDE, waarbij je vaak een package via de board manager moet downloaden, zijn deze boards vaak al beschikbaar in PlatformIO. Een ander groot voordeel is de beschikbaarheid van automatische code-aanvulling, wat in de Arduino IDE versie 1.0 helemaal niet mogelijk is en in versie 2.0 nog steeds niet optimaal werkt. Bovendien biedt PlatformIO bij veel chips de mogelijkheid om af te wijken van de Arduino-core, wat kan leiden tot efficiëntere code of bijvoorbeeld gemakkelijker gebruik van een RTOS.

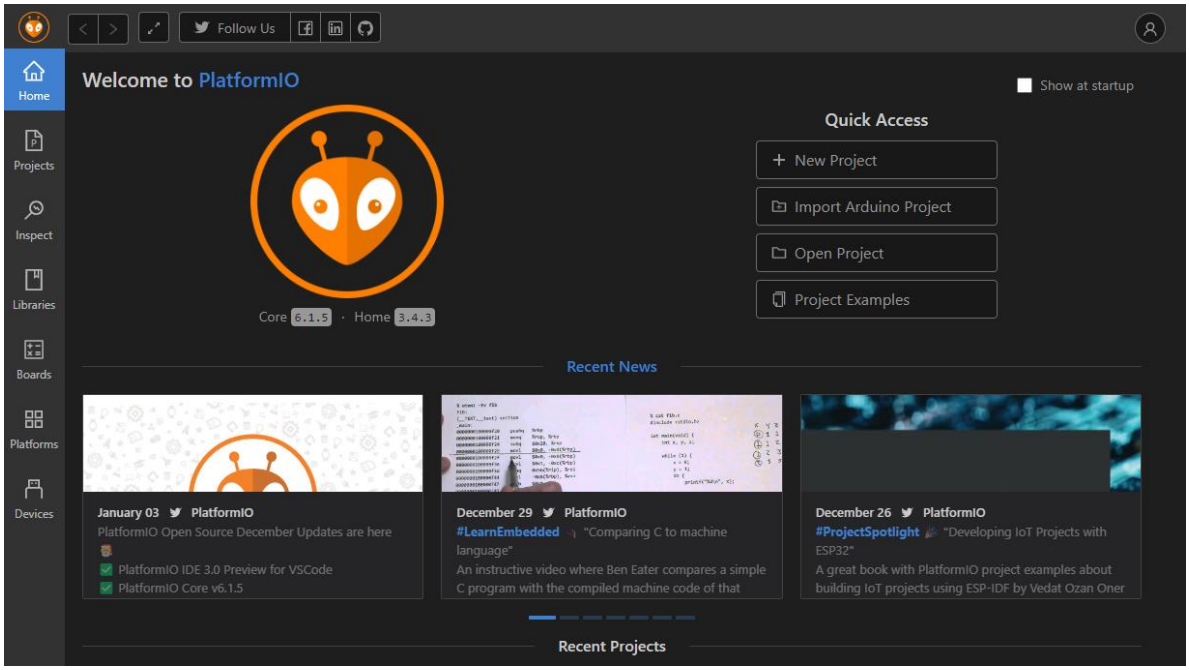
3.2.1 PlatformIO installeren

Om platformIO te gebruiken in VScode moet de extensie geïnstalleerd worden. Dit kan in het extensie venster aan de linker kant van het scherm. In dit extensie venster kan gezocht worden naar PlatformIO IDE.

Deze extensie kan vervolgens geïnstalleerd worden door op de INSTALL knop te klikken.

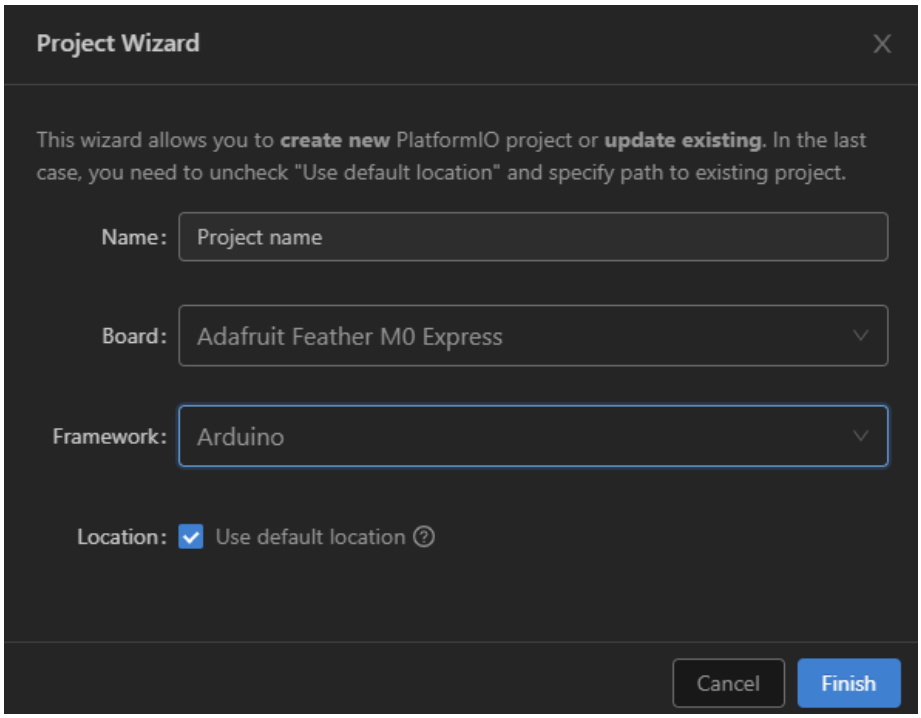
3.2.2 Project Maken

Een project maken kan via het PlatformIO home scherm.



Om een nieuw project aan te maken moet in dit scherm op de New Project knop geklikt worden.

Door hierop te klikken komt de project wizard tevoorschijn.



In deze project wizard kan het type board gekozen worden en het framework.

Voor de SensorHub van de babypop is dit de Adafruit Feather M0 Express.

Voor het MainBoard is dit de Adafruit Feather M4 express.

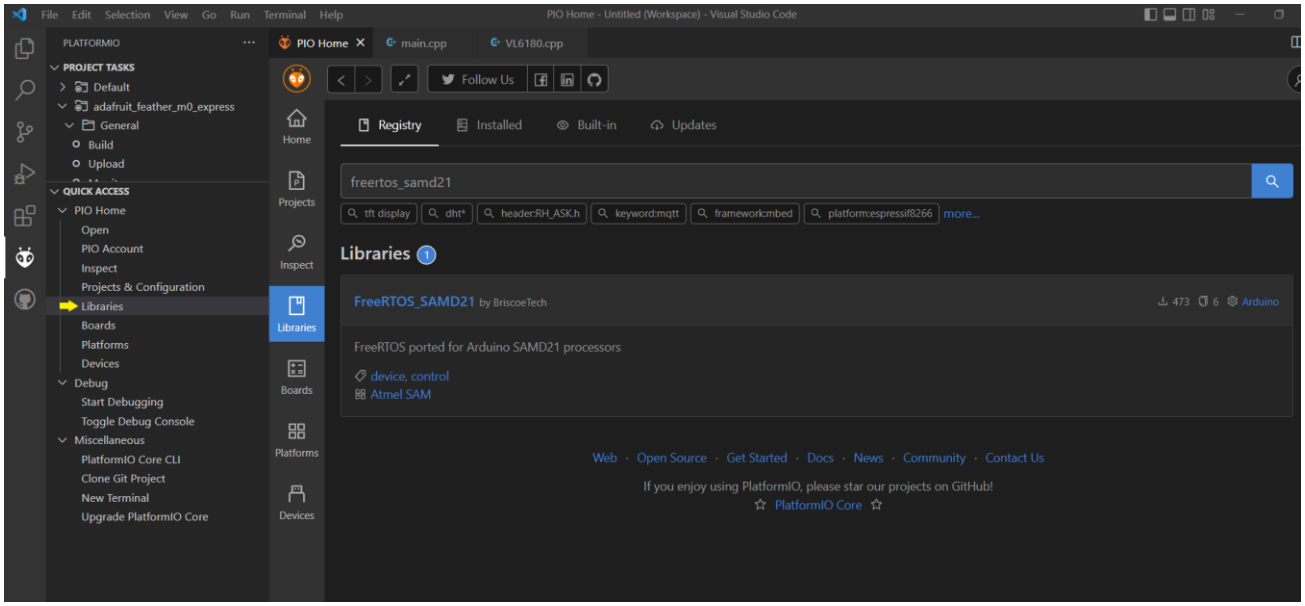
Het frameWork voor zowel de SensorHub als het MainBoard is Arduino.

3.2.3 Gebruik van bibliotheken

Om een bibliotheek te kunnen gebruiken in de code heb je 3 opties.

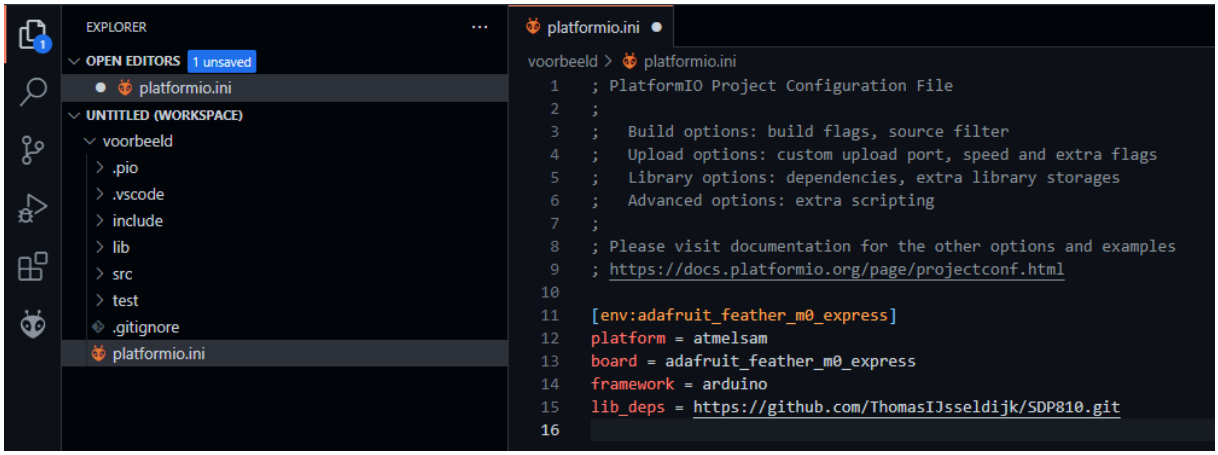
Optie 1, PIO library manager gebruiken:

de eerste manier om een library te gebruiken is door de in PlatformIO ingebouwde library manager te gebruiken. Via het kopje Libraries kun je een library toevoegen. Een voorbeeld hiervan is de freeRTOS_SAMD21 bibliotheek die gebruikt word voor het real time operating system.



Optie 2, dependancy toevoegen in .pio bestand.

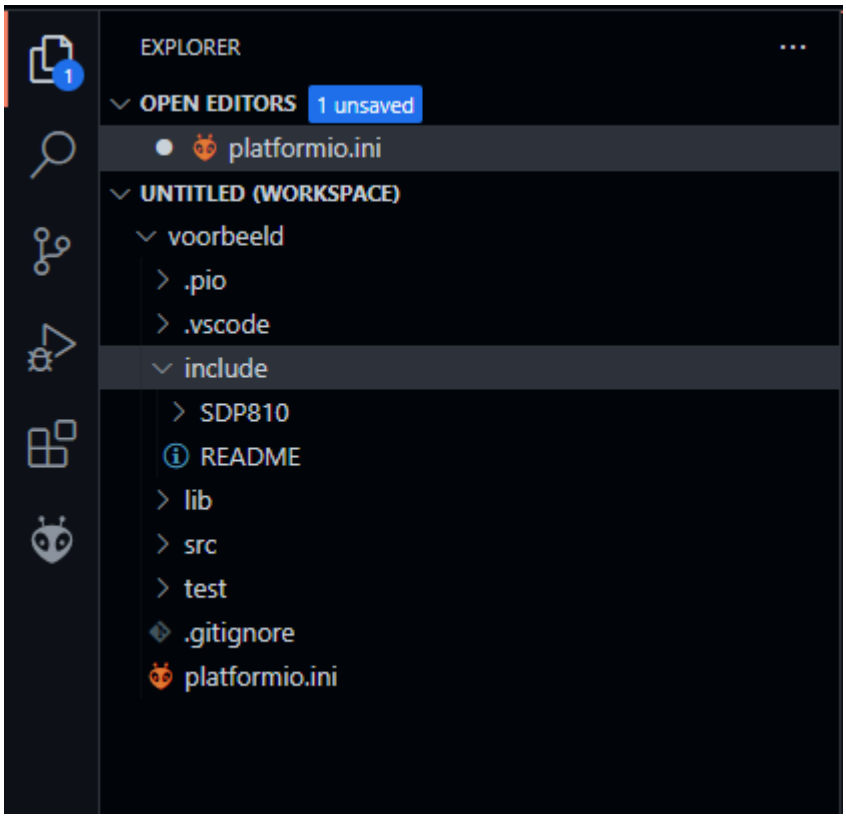
de tweede optie om een bibliotheek toe te voegen is door een github link als dependancy toe te voegen in het .pio bestand. Dit bestand is het configuratie bestand van het project.



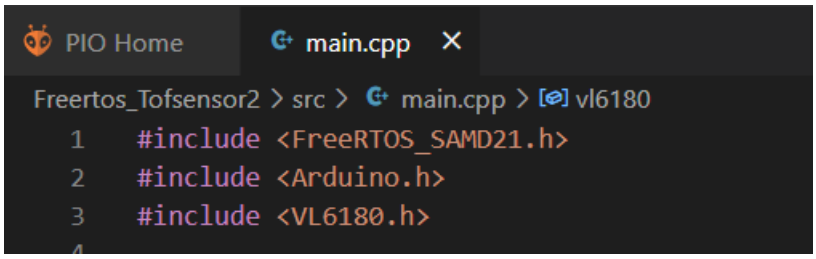
Het toevoegen van een bibliotheek kan in dit bestand doormiddel van de github https url toe te voegen onder het kopje lib_deps. Het kan zijn dat dit kopje nog niet bestaat. Deze moet dan handmatig toegevoegd worden aan het .pio bestand.

Optie 3, bibliotheek in include map zetten.

De derde en laatste optie voor het toevoegen van een bibliotheek is om het te plaatsen in de include map.



Om een bibliotheek te kunnen gebruiken moet na het uitvoeren van optie 1, 2 of 3 de code eerst gecompileerd worden. Daarna kan de biblitheek worden toegevoegd aan het programma door middel van #include <libraryName.h>



4 Code

4.1 SDP810 Differentiaal druk sensor

De SDP810 een differentiaal druk sensor van sensirion die gebruikt kan worden om een drukverschil tot 500 Pa te meten tussen twee slangen. De sensor kan uitgelezen worden via I2C.

4.1.1 Gebruik van de sensor

In de baby pop wordt de SDP810 gebruikt voor zowel het meten van luchtstroom als het meten of er geventileerd wordt. Voor het meten van de luchtstroom wordt gebruik gemaakt van een orifice. Een orifice is een smalle opening in een buis waarmee doormiddel van de wet van Bernoulli druk verschill onstaat. De luchtstroom meting kan worden gebruikt om te kijken of de ventilaties goed uitgevoerd worden. Om te meten of er geventileerd word wordt de sensor zo aangesloten dat een ingang verbonden is met de luchtweg en de andere ingang niet verbonden is. Hierdoor ontstaat er een druk verschil tussen de twee ingangen. Als deze sensor een drukverschil meet weet je dat er geventileerd wordt.

4.1.2 bibliotheek

voor het gebruiken van de SDP810 sensor is er een bibliotheek beschikbaar. Deze bibliotheek kan zowel voor het drukverschil als voor de luchtstroom worden gebruikt.

4.1.2.1 Functies van bibliotheek

void SDP810::begin(robotPatient_Wire *SDP810wire);

deze functie neemt een robotPatient_Wire object. Dit object is een I2C communicatie object wat wordt gebruikt voor het uitlezen van de sensor. Ook zorgt deze functie er voor dat de SDP810 in continuous measurement mode staat.

void SDP810::read();

deze functie leest de sensor uit. De waarde die de sensor terugstuurt wordt opgeslagen binnen de klasse

int16_t SDP810::getRaw();

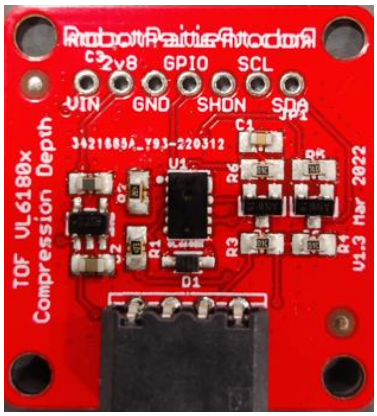
deze functie retourneert de ruwe sensor waarde die opgeslagen is in de klasse als een 16 bit signed integer.

float SDP810::getVolume();

deze functie retourneert het aantal liters per minuut dat door de orifice stroomt als een float. Let op dat deze waarde alleen klopt voor een orifice van 4mm.

4.2 VL6180 TOF sensor

De Time of flight sensor is een sensor die de afstand tot een voorwerp kan meten. Het meetbereik van de sensor ligt tussen ongeveer tussen de 15 en 255 mm. De sensor wordt uitgelezen via I2C.

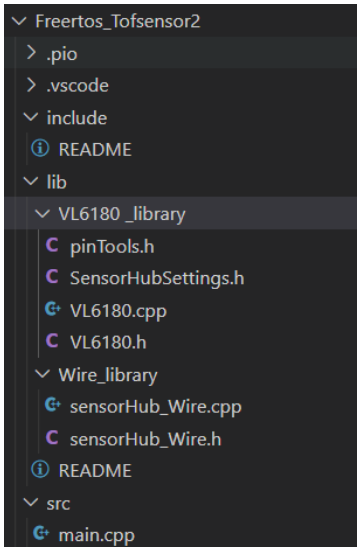


4.2.1 Gebruik van de sensor

Voor het juist kunnen reanimeren is het belangrijk om ook dieper compressies te kunnen geven. De compressiediepte kan gemeten worden, met juiste compressie ongeveer 1/3 van de baby pop. De diepte van de compressie is belangrijk om goed te doseren. Een gewenste diepte van ongeveer 4 cm moet behaald worden, te veel kan pijnlijk zijn voor de baby, te weinig zal niet voldoende zijn het hart volledig te legen. Het plaatsen van een Time of flight sensor in de borst moet inzicht gaan geven in deze diepte om zo een optimaal resultaat te behalen.

4.2.2 bibliotheek/code

Voor de Time of flight sensor heb ik een bibliotheek geschreven. De bibliotheek heet VL6180. In het bestand pinTools.h en SensorHubSettings.h staan i2c pinnen gedefinieerd.



```
6
7   xTaskHandle read_handle, write_handle = NULL;
8
9   void taskRead(void *arg);
10  void taskWrite(void *arg);
11
```

4.2.2.1 Main code

Boven in de main code staat de Taskhandle gedifinieerd Met read_handle en write_handle. Daaronder maak ik de functies aan: void taskread() en void taskwrite(). Dit zijn de FreeRTOS taken.

```
12  void setup()
13  {
14    Serial.begin(115200); // Start Serial at 115200bps
15    vl6180.initWires();
16
17    xTaskCreate(taskRead, "read", 500, NULL, 1, &read_handle);
18    xTaskCreate(taskWrite, "write", 500, NULL, 1, &write_handle);
19    vTaskStartScheduler();
20  }
```

In de void setup zet ik de serial moinitor op 115200bps. Daarna wordt de functie initwires uit klasse vl6180 uitgevoerd. Hierna worden de 2 taken geïnitialiseerd. Het maken van een taak gaat als volgt. Je voert de volgende regel in

xTaskCreate(vTaskCode, "NAME", STACK_SIZE, Parameters, tskIDLE_PRIORITY, &xHandle);.

vervolgens stel je de parameters goed in. De betekenis van de parameters is alsvolgt:
vTaskCode: Pointer naar de taakinvoerfunctie (alleen de naam van de functie die de taak implementeert).

"NAME": Een beschrijvende naam voor de taak. Dit wordt voornamelijk gebruikt om foutopsporing te vergemakkelijken, maar kan ook worden gebruikt om een taakafhandeling te verkrijgen.

STACK_SIZE: Het aantal woorden (geen bytes!) dat moet worden toegewezen voor gebruik als stapel van de taak. Als de stapel bijvoorbeeld 16 bits breed is en usStackDepth 100, dan worden 200 bytes toegewezen voor gebruik als de stapel van de taak. Een ander voorbeeld: als de stapel 32 bits breed is en usStackDepth 400, dan worden 1600 bytes toegewezen voor gebruik als de stapel van de taak.

Parameters: Een waarde die als parameter wordt doorgegeven aan de gemaakte taak. Als Parameters is ingesteld op het adres van een variabele, moet de variabele nog steeds bestaan wanneer de gemaakte taak wordt uitgevoerd - het is dus niet geldig om het adres van een stapelvariabele door te geven.

Priority: De prioriteit waarop de gemaakte taak wordt uitgevoerd.

&xHandle: Wordt gebruikt om een ingang door te geven aan de gemaakte taak vanuit de functie xTaskCreate(). pxCreatedTask is optioneel en kan worden ingesteld op NULL.

Ik stel de parameters voor allebei de functies zo in.
taskRead, “read”, 500, NULL, 1, &read_handle
taskWrite, “write”, 500, NULL, 1, &write_handle
met vtaskschedular start ik de taken, zonder deze regel werkt het programma niet.

```
21
22  void loop()
23  {
24
25  }
26
27  void taskRead(void *arg)
28  {
29    while (1)
30    {
31      vl6180.Distance_Measure();
32      vTaskDelay(100/portTICK_PERIOD_MS);
33    }
34  }
35
36  void taskWrite(void *arg)
37  {
38    while (1)
39    {
40      vl6180.ToggleLED();
41      vTaskDelay(100/portTICK_PERIOD_MS);
42    }
43  }
```

De void loop is tot nu toe leeg omdat de taken in de setup worden geïnitialiseerd en aangestuurd.

Daaronder maak ik de 2 functies van de taken uit de setup. Void taskRead() runt een functie Distance_Measure. In de functie zit nog een functie die waarde van de sensor omzet in de afstand in mm.

In de taskWrite functie draait een functie toggleLED.

4.2.2.2 VL6180.h code

In de H file zitten alle includes en defines van alle variable zoals bijvoorbeeld het i2c addres van de sensor. Daaronder staat een klasse VL6180 met daarin verschillende functies voor het uitvoeren van de code, bijvoorbeeld initWires(void); ToggleLED(void); en Distance_Measure(void);

4.2.2.3 VL6180.cpp code

void VL6180::begin(port channel);
Deze functie zet adress op de goede port.

void VL6180::initWires();
Eerst wordt de wirebackbone goed ingesteld met de juiste sercom door middel van deze functie WireBackbone.begin();
Daarna wordt met deze functie portBackbone.setPinPeripheralStates(); pinPeripheral goed gezet zodat de datapin en clock pin goed staan.

void VL6180::ToggleLED(void);
deze functie laat een ledje knipperen met een frequentie van 500ms.

void VL6180::Distance_Measure(void);
in de deze functie zit een andere functie getDistance. Deze functie zet de gemeten waarden om in een afstand in mm.

4.3 F-ram

Onderstaand figuur toond het break-out bord van het F-ram

Figuur 3 F-ram Break-out bord

Aansluitschema:

- VCC - dit is de voedingspin. Aangezien de chip 3-5 VDC gebruikt, moet u de logische spanning kiezen die u gebruikt. Voor de meeste Arduino's is dat 5V.
- GND - raakvlak voor kracht en logica
- I2C logische pinnen:
- WP - Schrijfbeveiligingspin. Dit wordt gebruikt om schrijfbeveiliging te forceren, zodat u niet naar het FRAM kunt schrijven. Het heeft een interne pulldown. Breng naar een hoge spanning (VCC) om WP in te schakelen
- SCL - I2C-klokpin, sluit aan op de I2C-kloklijn van uw microcontroller.
- SDA - I2C-datapin, sluit aan op de I2C-datalijn van uw microcontroller.

Type nummer senserhub: S2MTA 12140 730

Vanwege weinig tijd kon ik het F-ram niet verder onderzoeken/programmeren. Tot nu toe kon ik hand matig in de void loop waardes naar en F-ram sturen. Deze waarde kon ik via de seriële monitor uitlezen.