Sedimenthoogtemeter

Met WIFI interface

Bachelor in de Elektronica-ICT
Afstudeerrichting Elektronica

Vancampenhout Stef

Academiejaar 2013-2014
Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

VOORWOORD

De sedimenthoogte meter die in deze scriptie beschreven wordt is het onderwerp van mijn stage bij het Nederlands Instituut voor Onderzoek der Zee, gelegen op het eiland Texel in Nederland. Deze stage vormt het laatste onderdeel van de opleiding Elektronica-ICT met afstudeerrichting Elektronica aan de Thomas More hogeschool in Geel.

Bij deze wil ik dan ook een woord van dank richten aan de elektronici van de afdeling Mariene Technologie – Elektronica waaronder: Walther Lenting, Frank van Maarseveen, Martin Laan, Sander Asjes, Ruud Groenewegen en Henk Franken. Zij hebben mij met raad en daad bijgestaan tijdens het doorontwikkelen van de sedimentmeter waardoor dit project mij veel heeft bijgeleerd. Verder richt ik ook een woord van dank aan mijn stagebegeleider Luc Friant, en stage coördinator Ludo Cambré.

Tenslotte wil ik iedereen, in het bijzonder mijn klasgenoten in 3ELO, nog bedanken die de voorbije 3 jaren onvergetelijk hebben gemaakt.

SAMENVATTING

De sediment-hoogtemeter beschreven in dit eindwerk is een doorontwikkeling op vraag van onderzoekers van de afdeling Spatial Ecology te NIOZ Yrseke. Zij willen de sedimenthoogte op de zeebodem meten over een periode van enkele dagen tot enkele maanden. Het ontwikkelde instrument hiervoor betreft een analoge onderwater sediment-hoogtemeter uitgerust met 200 fototransistoren over een afstand van 40cm. Deze worden analoog ingelezen waarna ze worden weggeschreven naar een SD kaart. Mijn bijdrage aan dit project omvat het uitbreiden van de sedimentmeter met als voornaamste uitgangspunt een Wifi interface zodat na een bepaalde meetperiode de SD kaart niet meer opgehaald dient te worden, maar de data via deze Wifi interface met behulp van een tablet pc kan worden opgehaald. Ook zijn er enkele sensoren extra voorzien zoals een accelerometer, hall-sensor, temperatuursensor en IR ontvanger die bijkomende informatie geven over de fysieke factoren waarin de metingen gebeuren. Dit alles moet voldoen aan strikte voorwaarden omtrent energieverbruik, nauwkeurigheid en betrouwbaarheid.

Het ontwikkelproces van de sedimentmeter kent verschillende fasen. Deze zijn het uitwerken van een werkend schema en dit om te zetten naar een pcb, het uitwerken van de bijhorende software, alsook verschillende tests die moeten uitwijzen of de gekozen componenten en circuits voldoen aan de producteisen die vooraf gesteld zijn. Denk hierbij aan test omtrent de absorptie van elektromagnetische golven door (zee)water of het bereik van de Wifi module.

Na afloop van dit 13 weken durende project is gebleken dat de ontworpen hard- en software voldoet aan de verwachtingen. Het gemiddelde stroomverbruik in idle toestand is 100µA, het gemiddelde over een langere meetperiode met één meting per 10 seconden ligt rond de 1mA wat neerkomt op een maximale deployment periode van ±13 maanden; de reproduceerbaarheid en nauwkeurigheid van de metingen blijken ruim te voldoen aan de specificaties. En ook het bereik van de Wifi module met keramische chip-antenne is genoeg voor probleemloos gebruik in het veld.
De volgende stap is een optimalisatie van de hardware, waarna deze in productie kan worden genomen voor een oplage van 75 tot 100 stuks.

INHOUDSOPGAVE

LIJST VAN ILLUSTRATIES

LIJST VAN GEBRUIKTE AFKORTINGEN EN SYMBOLEN

NIOZ	Koninklijk Nederlands Instituut voor Onderzoek der Zee
PCB	Printed Circuit Board
SMT	Surface Mount Technology
SMD	Surface Mount Devices
LDO	Low Drop Out regulator
MCU	MicroController Unit
IC	Integrated Circuit
ASCII	American Standard Code for Information Interchange

INLEIDING

Al bij het begin van de stage voorbereidingen stond vast dat ik deze in het buitenland zou volbrengen. Ook wou ik tijdens dit 13 weken durende project mijn bijdrage leveren aan het wetenschappelijk onderzoek en op deze manier zowel kennis delen als opdoen die de maatschappij in staat stelt om zijn omgeving beter te begrijpen. Tenslotte vond ik het belangrijk dat het onderzoek dat ik hiermee zou steunen inspeelt op mijn persoonlijke interesses. Aangezien ik al van kinds af gefascineerd ben door de rivieren, zeeën, oceanen en alles wat hiermee te maken heeft zou dit ook de bepalende factor zijn bij het kiezen van een stagebedrijf. Met deze criteria in het achterhoofd kwam ik uit bij het Nederlands Instituut voor Onderzoek der Zee (NIOZ). Het project dat ik gedurende deze periode zou volbrengen betreft de sediment-hoogtemeter, deze wordt ingezet op zandbanken in de baai van Cádiz, voor de Spaanse kust om het verloop van de sedimenthoogte te meting in de tijd. Sediment is de onderste laag van de zeebodem die bestaat uit neerdalende resten van in het water levende organismen alsook neerdalende mineralen en stofdeeltjes.

De voornaamste producteisen zijn het opwaarderen van de bestaande sedimentmeter met nieuwe hardware die het voor de onderzoeker gemakkelijker maken om de data op te halen, en het risico op schade voor de hardware te beperken. Hiervoor is een Wifi interface gebruikt waarover u verder in deze scriptie meer informatie van kan vinden. Ook is er een LEDbar voorzien die de fototransistoren aanstraalt wanneer er onvoldoende omgevingslicht voorhanden is om een kwalitatieve meting te kunnen garanderen.

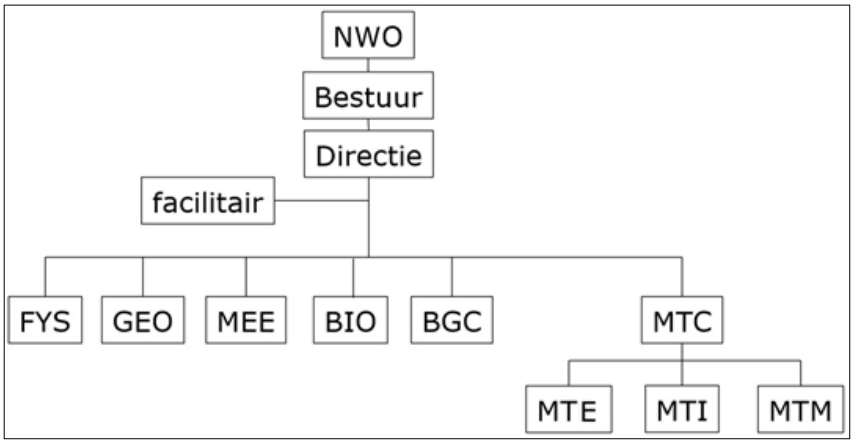
1 TEXEL

Figuur 1: Texel, 't Horntje

Figuur 2: Wadden

Texel is het eerste, en tevens het grootste van de Waddeneilanden voor de kust van Nederland. Het waddengebied is een bijzonder landschap dat door zijn geografische hoogte net onder het zeeniveau bij laagtij op bepaalde plaatsen droog komt te liggen. De zandbanken die zich dan boven het waterpeil bevinden worden wadden genoemd. In dit beschermde natuurgebied vindt men verschillende (bedreigde) planten en diersoorten die op deze wadden gedijen.

1.1 NIOZ



Figuur 3: Organigram NIOZ

Het NIOZ (Koninklijk Nederlands Instituut voor Onderzoek der Zee) bevindt zich in 't Horntje, het meest zuidelijk gelegen gehucht van Texel. De oceanografische onderzoeksafdelingen die zich hier bevinden zijn:

- Biologische Oceanografie (BIO)
- Fysische Oceanografie (FYS)
- Mariene Ecologie (MEE)
- Mariene Geologie en Chemische Oceanografie (GCO)
- Mariene Organische Biochemie. (BGC)

Daarnaast is er de afdeling Mariene Technologie die uit 3 afdelingen bestaat en bovenstaand onderzoek mogelijk maken:

- Marine Technologie Elektronica (MTE)
- Mariene Technologie Instrumentenmakerij (MTI)
- Mariene Technologie Mechanica. (MTM)

Voor dit onderzoek beschikt het NIOZ over een onderzoeksschip, de Pelagia.

Figuur 4: Pelagia

Verspreid over deze afdelingen telt het NIOZ 370 werknemers die er samen voor zorgen dat hun missie kan bereikt worden: het verkrijgen en uitdragen van kennis over de zeeën en oceanen zodat we hiervan een beter beeld krijgen en kunnen werken aan een duurzame toekomst voor onze planeet.

1.1.1 MTE

MTE is de afdeling waar het elektronisch gedeelte van de zeegaande apparatuur wordt ontworpen, getest, en onderhouden. Het bestaat uit een divers team van elektronischi met elk hun eigen vakgebied. Op deze afdeling is dan ook de sedimenthoogtemeter waarover deze scriptie gaat gemaakt. Het ontwerpen van schema's en PCB's (Printed Circuit Boards) gebeurt met het Altium Designer 14 softwarepakket. Voor kleine oplages of prototypes beschikt MTE over een verzameling JBC soldeerstations, een SMT (Surface Mount Technology) pick&place tafel en een SMT soldeeroven.

2 OPDRACHT

De Waddenzee is een complex ecosysteem waarin vele variabelen bijdragen aan het behoud of de teloorgang hiervan. Een van deze variabelen is de slib of sedimenthoogte. Slib is het zompige zand dat ontstaat wanneer bij laagtij een deel van het wad boven het waterpeil komt te staan. De hoogte van dit slib is afhankelijk van de getijdenstromingen, maar kan ook variëren door ingrepen van de mens. Door de sedimenthoogte nauwkeurig te meten en uit te zetten in functie van de tijd kunnen onderzoekers aan het NIOZ patronen ontdekken die een indicatie geven van de toestand in dit gebied.

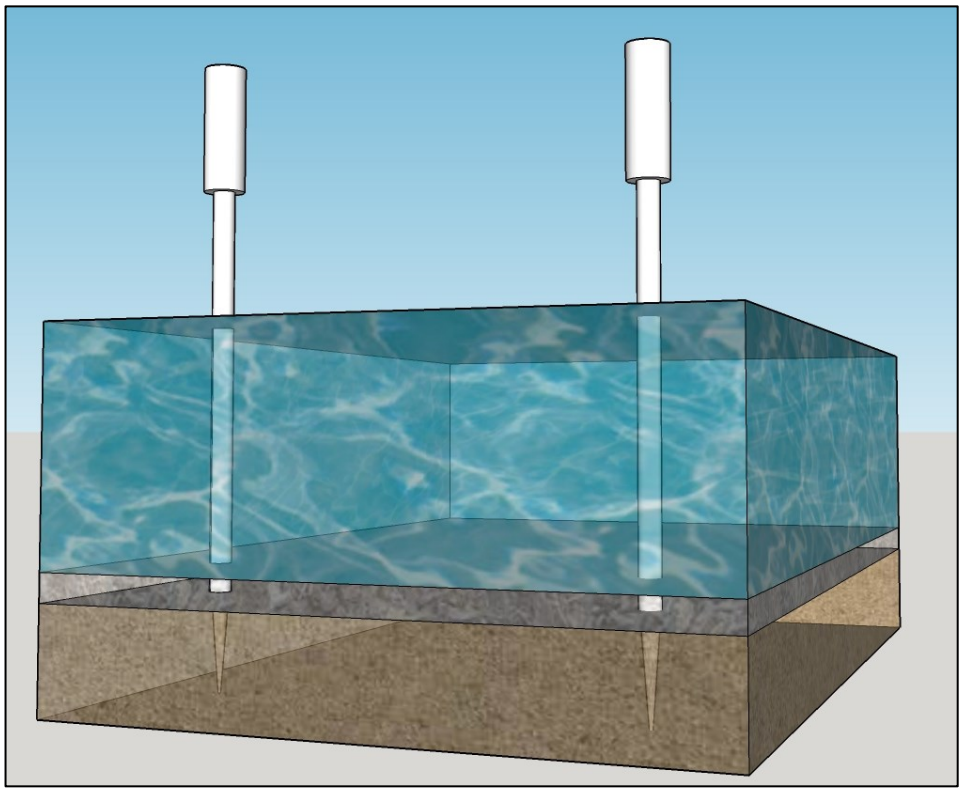
2.1 Doel

De huidige versie van de sedimentmeter is voorzien van een SD kaart om zijn data op te slaan. Dit is onhandig omdat voor het recupereren van de data de sedimentmeter uit de bodem verwijderd en opengemaakt dient te worden. Het openmaken van deze sondes is niet zonder risico aangezien het gebeurt in de nabijheid van zout water. Ook heeft het ophalen van de sedimentmeter invloed op toekomstige metingen, aangezien elk fysisch contact de bodem verstoort. De oplossing hiervoor zit hem in het draadloos versturen van de data. Dit maakt het ophalen van de gegevens efficiënter en de metingen meer betrouwbaar.

Het meetprincipe dat gebruikt wordt bij de sedimentmeter maakt gebruik van 200 fototransistoren die naast elkaar gemonteerd zijn op een lengte van 75cm. Wanneer sediment ophoopt voor de sensoren zijn de lichtmetingen een maat voor de hoogte hiervan. De vorige versie van de sedimentmeter was afhankelijk van het zonlicht om de sensoren te belichten. Hierdoor werd hij bij weinig licht onbetrouwbaar en ’s nachts zelfs onbruikbaar. Om dit probleem op te lossen dient de sedimentmeter uitgerust te worden met een aanlichtbar zodat het zonlicht niet meer van invloed is op de metingen

2.2 Opbouw sedimentmeter

Onderstaande rendering geeft aan hoe de sedimentmeter in gebruik zal worden genomen. De cilinders worden met een metalen pin vastgezet in de bodem, in dit voorbeeld zullen we stellen dat de linkse cilinder de sensor betreft en de rechtse de LEDbar. De middelste laag stelt het sediment voor waarvan we de hoogte willen bepalen. Dit gebeurt door het licht te meten over de lengte van de sensor. In de metingen zullen dan 2 of 3 zones voortkomen die aangeven over welke afstand de sensoren in het sediment, water, en eventueel voor een gedeelte droog staan. Overdag kan deze meting gebeuren met het licht van de zon, maar ’s nachts is dit niet mogelijk. Daarvoor dient de 2^{e} cilinder, deze wordt actief op het moment dat het licht van de zon niet meer voldoende of niet meer aanwezig is. Vanaf dan zal de sedimentmeter een signaal versturen naar de LEDbar waardoor deze gedurende 250ms een IR lichtbalk projecteert op de sensoren.

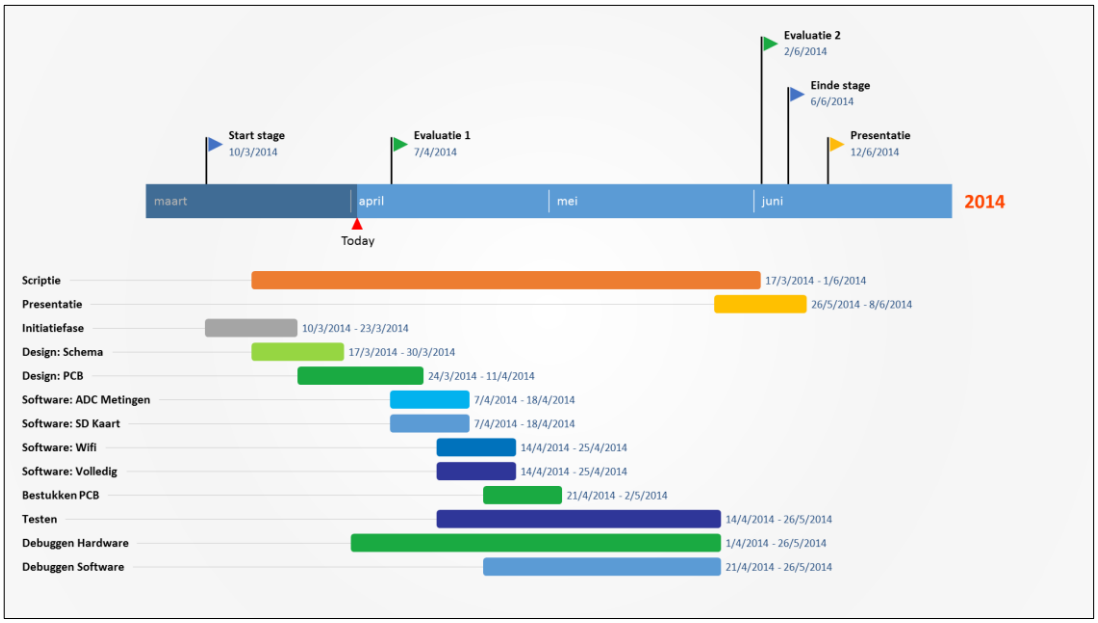


Figuur 5: Opstelling Sedimentmeter

3 PLAN VAN AANPAK

Bij het aanvatten van mijn eindwerkstage bleek al snel dat ik met de op school vergaarde kennis niet zou toekomen. MTE steunt namelijk op het Altium Designer ontwikkelpakket voor het ontwerpen van schema's en PCB's. Ook IAR Development Suite voor de MSP430 microcontroller van Texas Instruments, specifiek uitgekozen omwille van zijn lage stroomverbruik in vergelijking met gelijkaardige controllers was volledig vreemd voor mij. De eerste mijlpaal die behaald moest worden was dus het eigen maken van deze softwarepakketten

Het verloop van dit 13 weken durende project kan u op onderstaande timeline vinden.



Figuur 6: Planning

3.1 Altium Designer DX14 & IAR Development Suite

Altium Designer DX14, opvolger van het welgekende Protel softwarepakket is de Electronic Design Assistant (EDA) bij uitstek voor het ontwikkelen van uitgebreide schema's en printplaten. Het eigen maken van de ontwikkelomgeving is niet vanzelfsprekend, omwille van de uitgebreide mogelijkheden van de gebruikersinterface. Toch loont het de moeite om met dit pakket te leren omgaan, aangezien al deze aanpassingsmogelijkheden het in uitgebreide schema's gemakkelijker maken om efficiënt te werk te gaan.

3.2 Product Requirements

Aangezien de sedimentmeter gedurende lange tijd standalone moet kunnen loggen, is het vanzelfsprekend dat dit enkele specifieke eisen aan het ontwerp stelt. Een voorbeeld hiervan is de gebruiksduur van de sedimentmeter: deze is rechtstreeks afhankelijk van zijn stroomverbruik, dus het is van groot belang om deze tot een minimum te reduceren.

3.2.1 Low Power

Zoals eerder vermeld is het van cruciaal belang om de sedimentmeter zo energiezuinig mogelijk te maken. Om dit te bewerkstelligen zijn er verregaande maatregelen getroffen in het design die wanneer er niet gemeten wordt grote delen van de hardware gaat uitschakelen. Het stand-by stroomverbruik van de hele schakeling wordt op deze manier beperkt tot +-100µA. Het actieve verbruik over langere periode ligt rond de 1mA. De totale meetperiode op 4D cellen van 12000mAh is geschat op 14 maanden.

3.2.2 Accuracy

De nauwkeurigheid van de metingen is van groot belang wanneer de data opgehaald en geanalyseerd wordt. Daarom zijn er 3 meet bereiken voorzien waarop de lichtmetingen kunnen worden uitgevoerd. Dit aangepast meetbereik in combinatie van een hoge resolutie ADC van 14 bit maakt dat de opgeslagen data van voldoende hoge kwaliteit is om nauwkeurige metingen te kunnen verrichten.

3.2.3 Reliability

De sedimentmeter wordt gebruikt in een voor elektronica vijandige omgeving. Het mechanische ontwerp is hierop voorzien, en ook de elektronica is voldoende robuust ontworpen om bedrijfszekerheid op lange termijn te kunnen garanderen.

4 REALISATIE

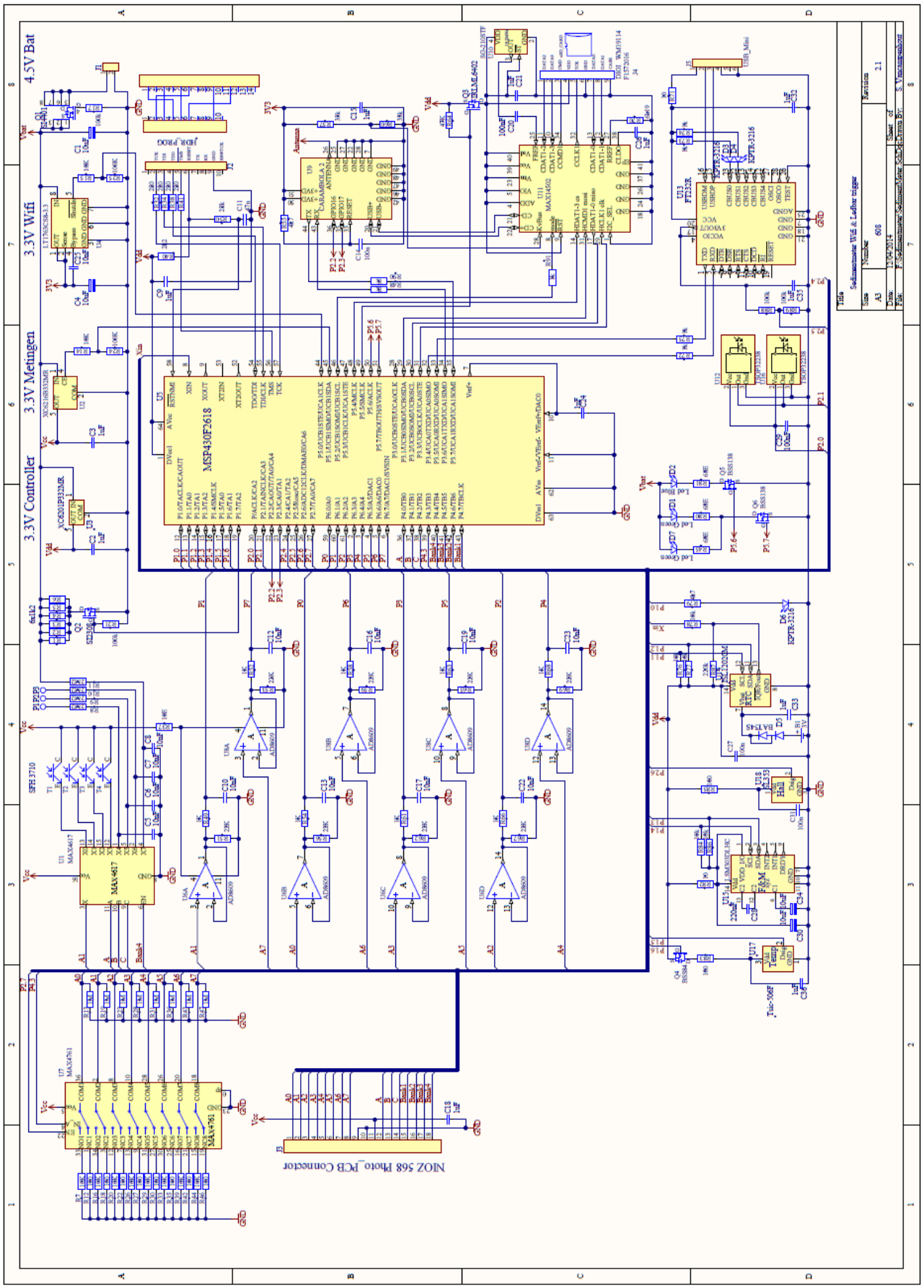
In dit hoofdstuk staat de opbouw en werking van de sedimentmeter uitvoerig beschreven. Bijhorende software code vindt u in de bijlagen op pagina 44.

4.1 Hardware

De sedimentmeter bestaat uit 2 printplaten: een meetprobe waarop de 200 fototransistoren bevestigd zijn, en de controller waar de gemeten data ingelezen en opgeslagen wordt. Deze pcb's zijn verbonden via een 18 polige dual in-line package (DIP) pin header.

4.1.1 Controller Schema

Op de controller PCB zit de MSP430F2618 microcontroller, 3 verschillende 3.3V voedingen, een analoog gedeelte om de data in te lezen, en een digitaal gedeelte waarin de data wordt opgeslagen en later verzonden. Ook is er een hall sensor, een accelerometer en een hoge resolutie temperatuursensor voorzien.



Figuur 7: Schema controller

4.1.1.1 Microcontroller

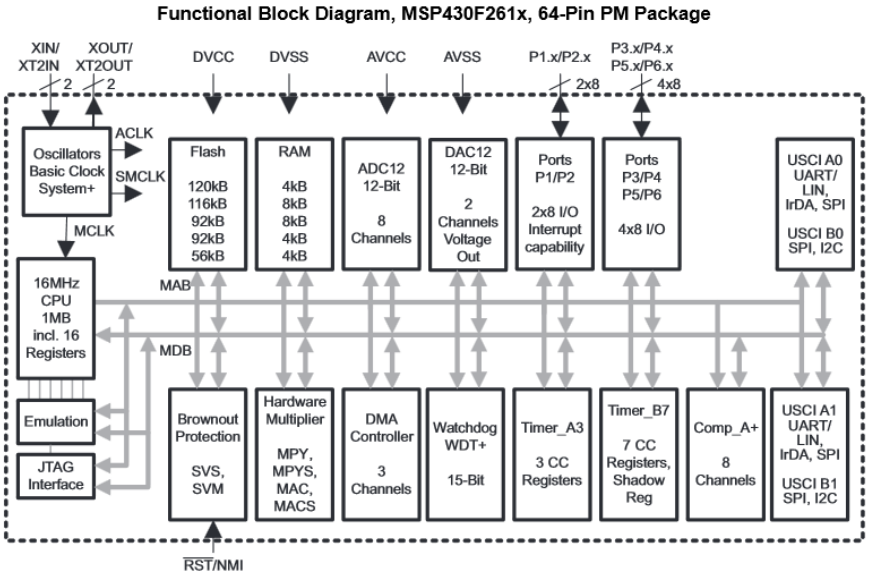
De gebruikte microcontroller (MCU) is een MSP430F2618 van Texas Instruments, deze is gekozen omwille van zijn lage stroomverbruik en nauwkeurige 12 bit AD convertors. Het programmeren van deze controller gebeurt via een JTAG interface die ook gebruikt kan worden voor debugging doeleinden. In onderstaande tabel vindt u enkele key-features die deze MCU een interessante keuze voor dit project maken.

MIXED SIGNAL MICROCONTROLLER

FEATURES

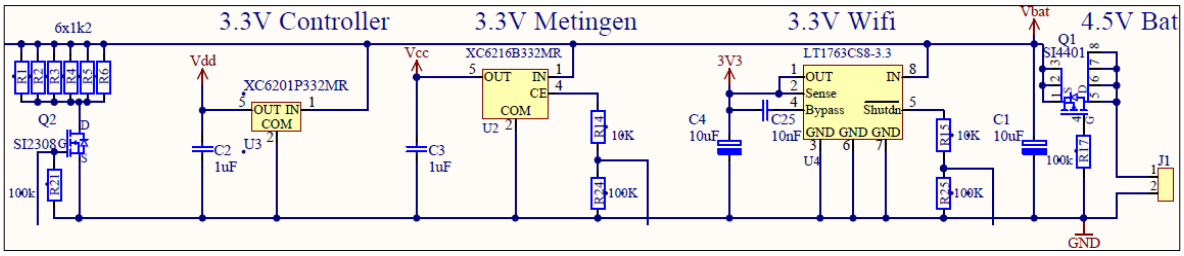
- Low Supply Voltage Range 1.8 V to 3.6 V
- Ultra-Low Power Consumption
  - Active Mode: 365 µA at 1 MHz, 2.2 V
  - Standby Mode (VLO): 0.5 µA
  - Off Mode (RAM Retention): 0.1 µA
- Wake-Up From Standby Mode in Less Than 1 µs
- 16-Bit RISC Architecture, 62.5-ns Instruction Cycle Time
- Three-Channel Internal DMA
- 12-Bit Analog-to-Digital (A/D) Converter With Internal Reference, Sample-and-Hold, and Autoscan Feature
- Dual 12-Bit Digital-to-Analog (D/A) Converters With Synchronization
- 16-Bit Timer_A With Three Capture/Compare Registers
- 16-Bit Timer_B With Seven Capture/Compare-With-Shadow Registers
- On-Chip Comparator
- Four Universal Serial Communication Interfaces (USCIs)
  - USCI_A0 and USCI_A1
    - Enhanced UART Supporting Auto-Baudrate Detection
    - IrDA Encoder and Decoder
    - Synchronous SPI
  - USCI_B0 and USCI_B1
    - I^2C™
    - Synchronous SPI
- Supply Voltage Supervisor/Monitor With Programmable Level Detection
- Brownout Detector
- Bootstrap Loader
- Serial Onboard Programming, No External Programming Voltage Needed, Programmable Code Protection by Security Fuse
- Family Members:
  - MSP430F2416
    - 92KB + 256B Flash Memory
    - 4KB RAM
  - MSP430F2417
    - 92KB + 256B Flash Memory
    - 8KB RAM
  - MSP430F2418
    - 116KB + 256B Flash Memory
    - 8KB RAM
  - MSP430F2419
    - 120KB + 256B Flash Memory
    - 4KB RAM
  - MSP430F2616
    - 92KB + 256B Flash Memory
    - 4KB RAM
  - MSP430F2617
    - 92KB + 256B Flash Memory
    - 8KB RAM
  - MSP430F2618
    - 116KB + 256B Flash Memory
    - 8KB RAM
  - MSP430F2619
    - 120KB + 256B Flash Memory
    - 4KB RAM
- Available in 80-Pin Quad Flat Pack (LQFP), 64-Pin LQFP, and 113-Pin Ball Grid Array (BGA) (See Table 1)
- For Complete Module Descriptions, See the *MSP430x2xx Family User's Guide* (SLAU144)

Figuur 8: MSP430F2618 specs



Figuur 9: Blokschema MSP430F2618

4.1.1.2 Voeding



Figuur 10: Schema voeding

De sedimentmeter heeft 3 verschillende 3.3V voedingen.

- 3.3V Controller Im = 200mA, Iq = 2µA
- 3.3V ADC metingen Im = 200mA, Iq = 5µA, Istby = 0.1µA
- 3.3V Wifi Im = 500mA, Iq = 5mA, Istby = 0.1µA

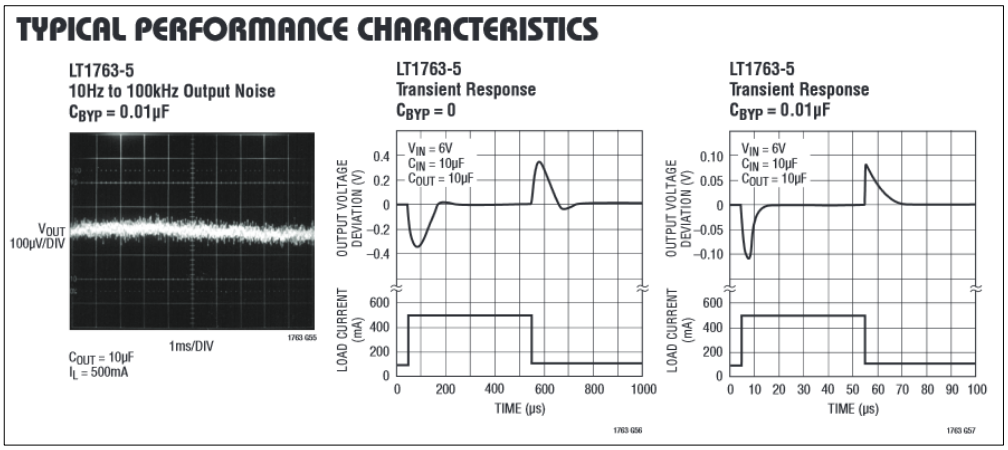
(Im = I max, Iq = I quiescent, Istby = I standby)

Van deze voedingen kunnen er 2 in stand-by gezet worden zodat hun stroomverbruik naar 0.1µA zakt. Dit is van belang omdat deze Low Drop Out (LDO) regulatoren in actieve modus zelf meer stroom verbruiken dan wenselijk is. Door het uitschakelen van de voeding is het ook gemakkelijk om niet-gebruikte hardware volledig uit te schakelen en zo het stroomverbruik tot een minimum te beperken. De enige LDO die nooit in stand-by hoeft te gaan en waar dit ook niet op voorzien is, is de regulator die de voeding voor de MCU voorziet.

De CE/Shutdown ingang van de LDO's is via een weerstandsnetwerk aan de MCU gekoppeld. Gedurende een power-on reset bij het inschakelen van de sedimentmeter zijn de uitgangen van de MCU gedurende korte tijd in tri-state. Door de 100k pull-down weerstanden worden de LDO's in stand-by gehouden om ongewenst inschakelen te voorkomen.

- Q1 is een P-kanaals mosfet die gebruikt wordt als ompoolbeveiliging. In eerste instantie zal er een kleine stroom door de diode lopen waardoor de source positiever wordt t.o.v. de gate die aan de ground zit. Hierdoor komt de mosfet zelf in geleiding waardoor deze met een minimale spanningsval de batterijspanning doorschakelt naar de rest van het ciruit. Bij een verkeerde polarisatie zal er geen diodestroom, en zelfs moest dit zo zijn dan zou Ugs verkeerd gepolariseerd zijn waardoor de mosfet blijft sperren.

- De LT1763CS8 LDO van Linear Technology gaat de 3.3V spanning voorzien voor de Carambola 2 WIFI module. Daarom is het van belang dat deze regulator een heel stabiele uitgangspanning heeft. Door gebruik te maken van een externe bypass condensator kan een low-noise 3.3V spanning bereikt worden.

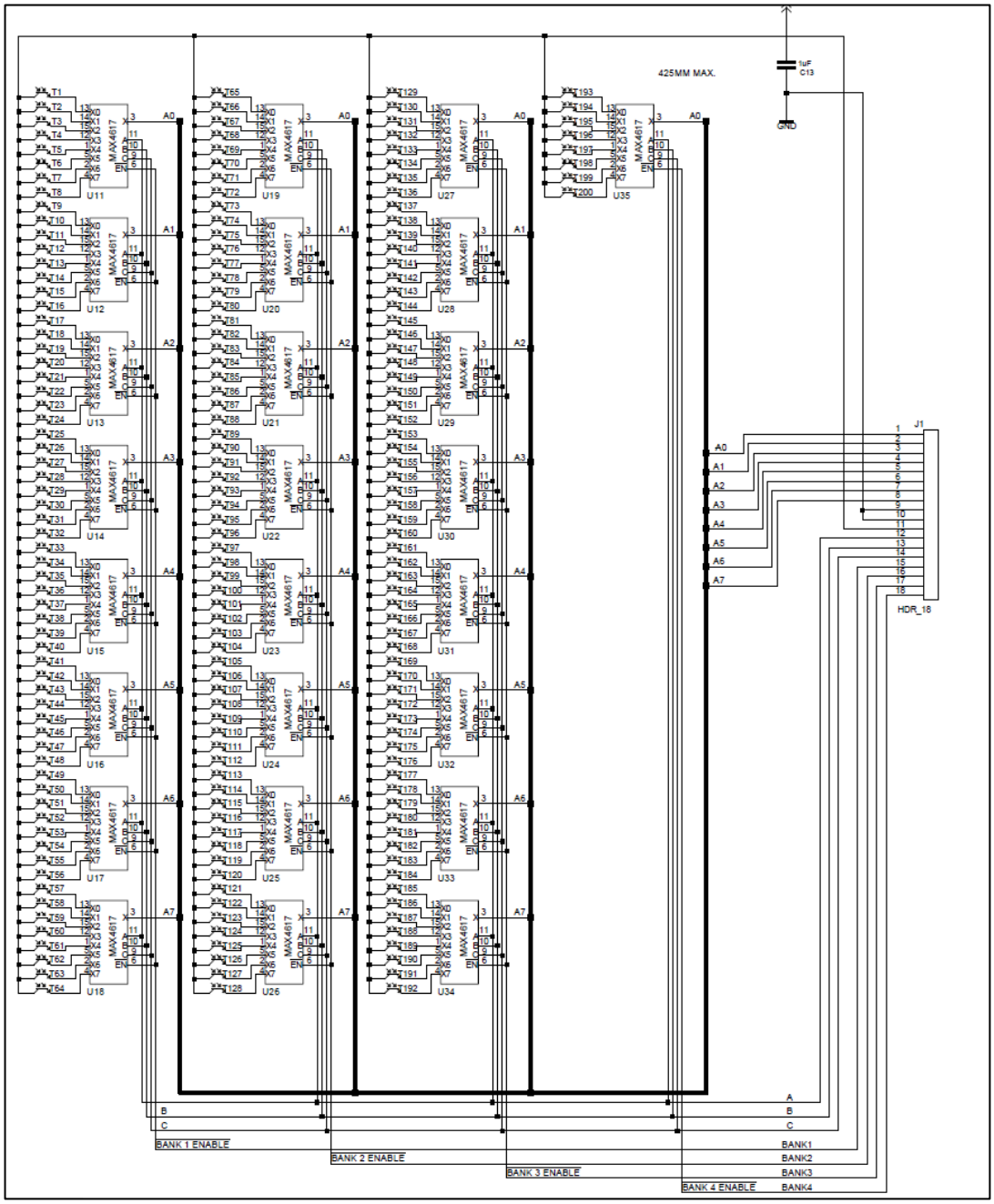


Figuur 11: LT1763 Low Noise Characteristics

- Om het laadniveau van een batterij correct te meten, dient dit te gebeuren onder belasting. Daarom zijn er 6 100Ω weerstanden parallel voorzien die wanneer bijhorende fet is ingeschakeld een batterijstroom van 225mA trekken. Via een vrije poort op een MAX4617 (later meer hierover) kan de batterijspanning ingelezen worden.

4.1.1.3 Analoge multiplexing

De 200 fototransistors worden gemultiplexed ingelezen om het aantal ADC ingangen op de microcontroller beperkt te kunnen houden. Voor deze multiplexing zijn MAX4617 IC's gebruikt. Er zijn 4 banken van deze IC's die via de enable ingang sequentieel ingelezen worden. Via 3 adreslijnen kunnen deze 8 chips ook weer sequentieel geadresseerd worden. Per analoge multiplexer worden 8 fototransistors ingelezen.



Figuur 12: Schema fototransistor PCB

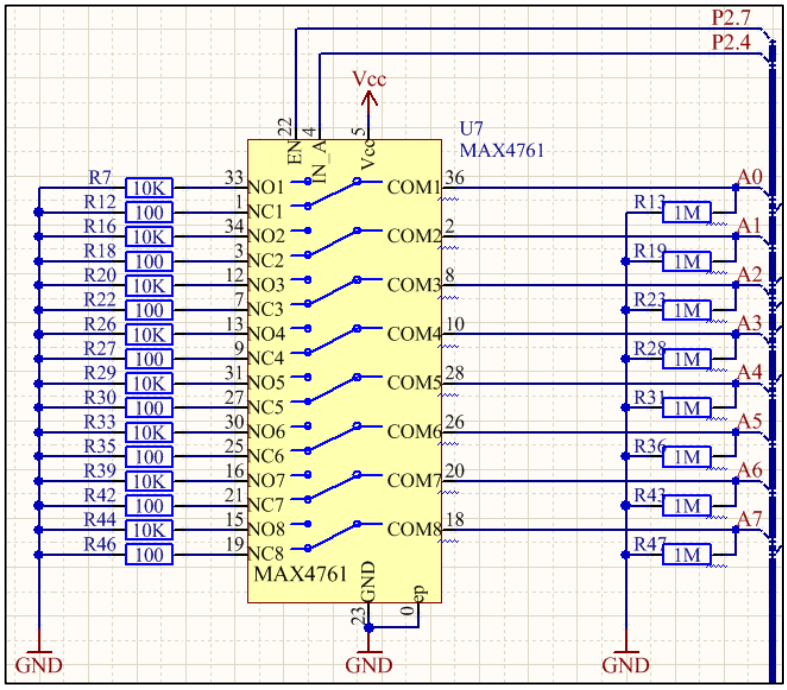
De multiplexing gebeurt in volgende sequentie:

Bank	Adres		A0	A1	A2	A3	A4	A5	A6	A7
1	000		T1	T9	T17	T25	T33	T41	T49	T57
1	001		T2	T10	T18	T26	T34	T42	T50	T58
1	010		T3	T11	T19	T27	T35	T43	T51	T59
1	011		T4	T12	T20	T28	T36	T44	T52	T60

1	100		T5	T13	T21	T29	T37	T45	T53	T61
1	101		T6	T14	T22	T30	T38	T46	T54	T62
1	110		T7	T15	T23	T31	T39	T47	T55	T63
1	111		T8	T16	T24	T32	T40	T48	T56	T64
2	000		T65	T73	T81	T89	T97	T105	T113	T121
2	001		T66	T74	T82	T90	T98	T106	T114	T122
2	010		T67	T75	T83	T91	T99	T107	T115	T123
2	011		T68	T76	T84	T92	T100	T108	T116	T124
2	100		T69	T77	T85	T93	T101	T109	T117	T125
2	101		T70	T78	T86	T94	T102	T110	T118	T126
2	110		T71	T79	T87	T95	T103	T111	T119	T127
2	111		T72	T80	T88	T96	T104	T112	T120	T128
3	000		T129	T137	T145	T153	T161	T169	T177	T185
3	001		T130	T138	T146	T154	T162	T170	T178	T186
3	010		T131	T139	T147	T155	T163	T171	T179	T187
3	011		T132	T140	T148	T156	T164	T172	T180	T188
3	100		T133	T141	T149	T157	T165	T173	T181	T189
3	101		T134	T142	T150	T158	T166	T174	T182	T190
3	110		T135	T143	T151	T159	T167	T175	T183	T191
3	111		T136	T144	T152	T160	T168	T176	T184	T192
4	000		T193	**Tr1***						
4	001		T194	**Tr2***						
4	010		T195	**Tr3***						
4	011		T196	**Tr4***						
4	100		T197	**Tr5***						
4	101		T198	**Tr6***						
4	110		T199	**Tr7***						
4	111		T200	**Tr8***						

***Tr = Referentie fototransistors op de controller PCB**

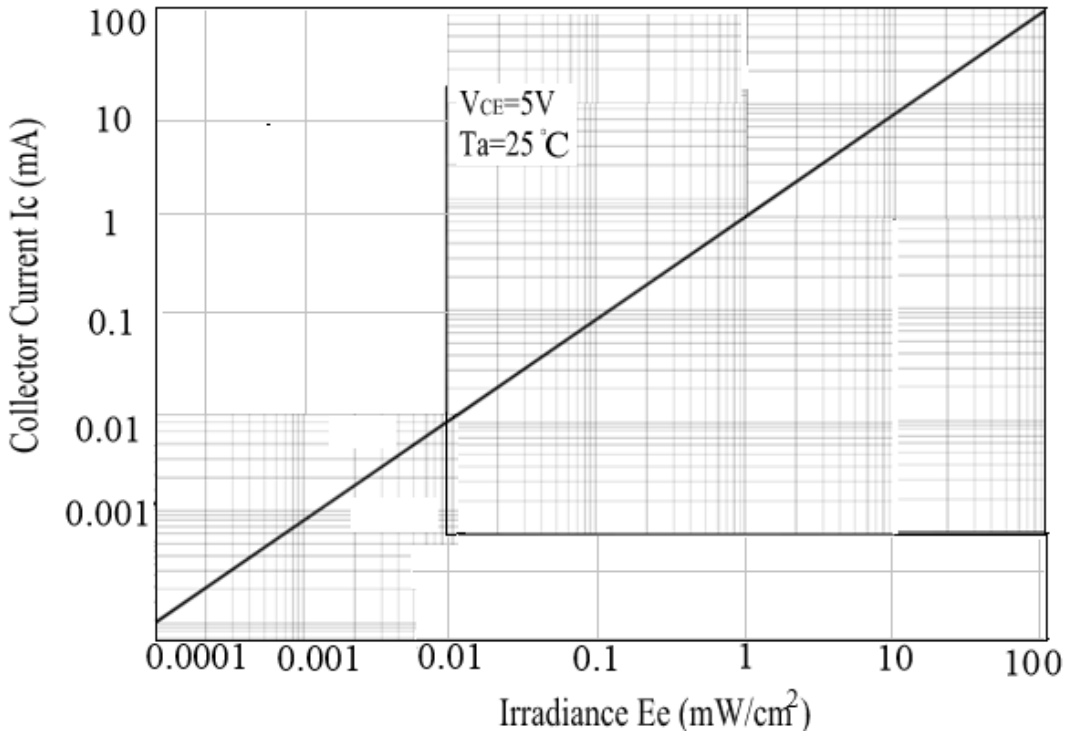
De stroom afkomstig van de fototransistors wordt vervolgens via een MAX4761 analoge switch door een weerstand van 1M, 100K of 100Ω gestuurd waarover de gemeten spanningsval een maat is voor de hoeveelheid licht die de fototransistor bereikt. Op deze manier kan door de grootte van de meetweerstand te veranderen te schaal aangepast worden.



Figuur 13: Schema schaalbereik

Met deze 3 weerstandswaarden en de grafiek van de sperstroom in functie van het opgevangen vermogen op de fototransistor is het mogelijk om de schaalverdeling te berekenen.

ADC bereik = 0V tot 3.3V
R = 1M, 10K, 100Ω

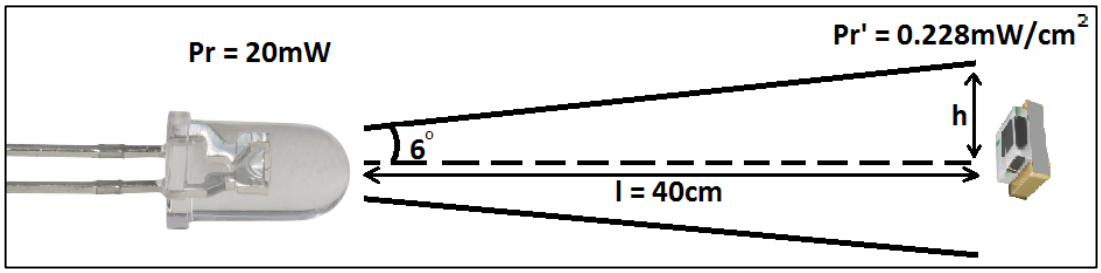


Figuur 14: Grafiek Collectorstroom in functie van ingestraald vermogen

R	Ira (Ur = 3.3V)	Ee(Ira)	Resolutie(12bit)
1M	3.3nA	<100nW/cm²	Δ1 bit = Δ<24.4pW/cm²
10K	330nA	3 µW/cm²	Δ1 bit = Δ 0.73nW/cm²
100Ω	33mA	12 mW/cm²	Δ1 bit = Δ 2.9µW/cm²

Met bovenstaande grafiek, de gegevens van de leds in de lichtbar en de gegevens van de fototransistoren is het mogelijk om de sperstromen door de detectoren te bepalen en bijgevolg ook de spanningen over de meetweerstanden.

Zender: IR led 830nm, Uitgestraalt vermogen = 20mW, hoek = 12º
Ontvanger: SFH3710



Figuur 15: Geometrisch stralingsdiagramma LED

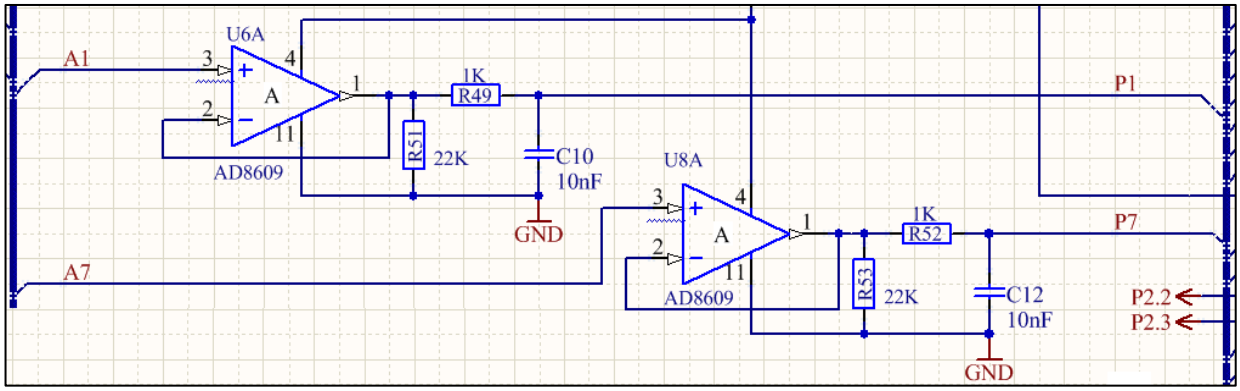
$$Tang(6°) = \frac{h}{40cm} \qquad h = 4.2cm$$

$$A = \pi r^2 = \pi * 4.2cm^2 = 55.42cm^2$$

Uitgestraalt vermogen per cm² op 40cm afstand: 20mW/55.42cm²
= 0.36mW/cm², na inbrengen van de demping (4.8dB/40cm) blijft hier
Pr' = 0.228mW/cm² van over.

Als we deze waarde op de grafiek van de ontvanger leggen, dan valt hieruit af te leiden dat de maximum sperstroom 0.2mA bedraagt. Bepaalde fysische factoren zoals algen en opstuivend zand zullen tot effect hebben dat de gemeten waarde steeds onder deze 0.2mA blijft.

4.1.1.4 Analoge ingangen

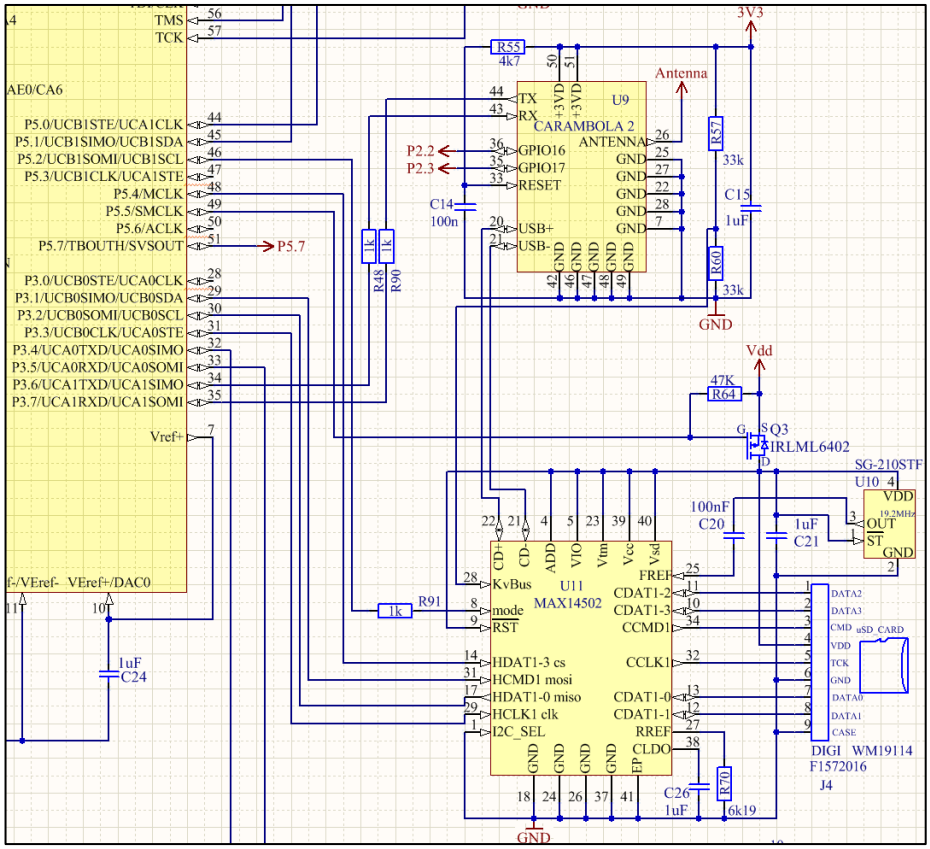


Figuur 16: Schema buffertrap

De gemultiplexte analoge worden eerst gebufferd door AD8609 micropower opamps. Deze hebben een laag eigen verbruik van 50µA en zijn ruisarm waardoor de invloed op de metingen minimaal is. Na elke buffertrap is een RC laagdoorlaatfilter voorzien zodat eventuele spikes uit het signaal weg-gefilterd worden.

$$Fc = \frac{1}{2\pi RC} = \frac{1}{2\pi * 1K\Omega * 10nF} = 15{,}915\text{KHz}$$

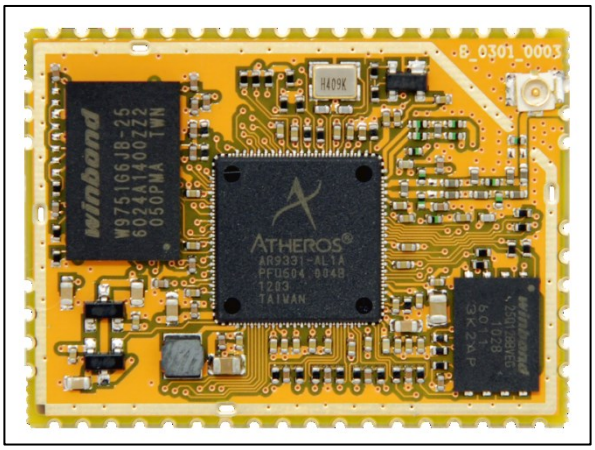
4.1.1.5 Data opslag en transmissie



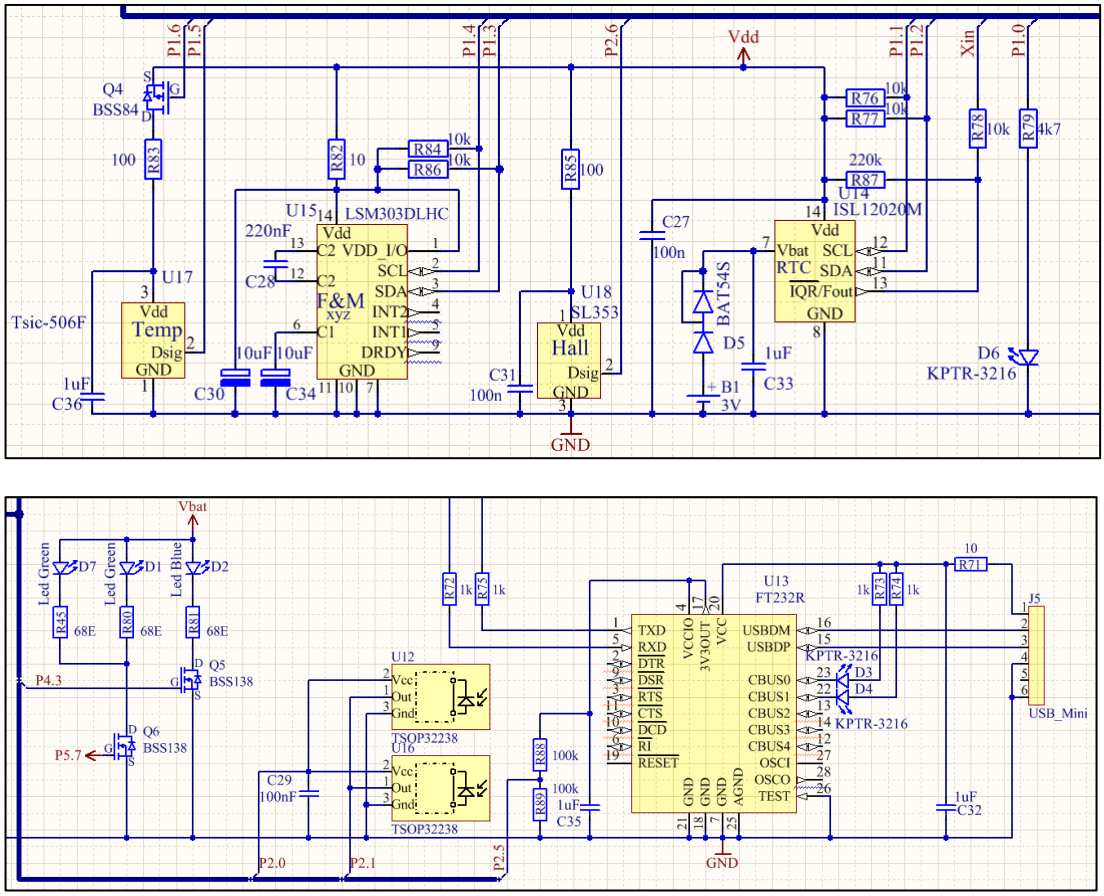
Figuur 17: Schema WIFI, SD kaart

De gemeten data wordt weggeschreven naar een 2GB micro-SD kaart. Per meting worden 200 sensoren 3x over een bereik van 12 bits ingelezen (3 schaal bereiken), wat neerkomt op 7200 bits/meting = 900 Bytes/meting. Op een 2GB geheugenkaart kunnen dus 2 222 222 metingen opgeslagen worden. Als we meten met een interval van 1 minuut komt dit neer op 1543 dagen tot de kaart vol is. Na compressie van de data is deze meetperiode zelfs tot enkele tienvouden van het oorspronkelijke uitbreidbaar.

Het ophalen van de data gebeurt met een wifi interface. Deze wordt verzorgd door de Carambola2 Wifi module welke is uitgerust met een Atheros AR9331 chipset die draait op embedded Linux. De data wordt vanuit de SD kaart opgehaald door de MAX14502, dit is een high-speed SD card reader die de data gaat wegschrijven naar de Carambola2. Deze kaartlezer is voorzien van een eigen klokgenerator op 19.2MHz zodat het versturen van data over Wifi volledig onafhankelijk van de microcontroller kan gebeuren.



4.1.1.6 Digitale periferie



Figuur 18: Schema digitale periferie

Naast het analoge gedeelte en de data opslag/transmissie zijn er ook enkele parallelle systemen die de fysische omstandigheden van de sensor bijhouden.

- TSIC-506F is een 0.1ºK nauwkeurige digitale temperatuursensor die adresseerbaar is met het ZaCwire protocol, soortgelijk aan 1-Wire.

- LSM303DLHC is een I²C 3D accelerometer en kompasmodule. Deze wordt gebruikt om de oriëntatie van de sensor te bepalen uit de opgeslagen data. Deze sensor kan via I²C in sleep mode gebracht worden wat het stroomverbruik verlaagd tot gemiddeld 1µA

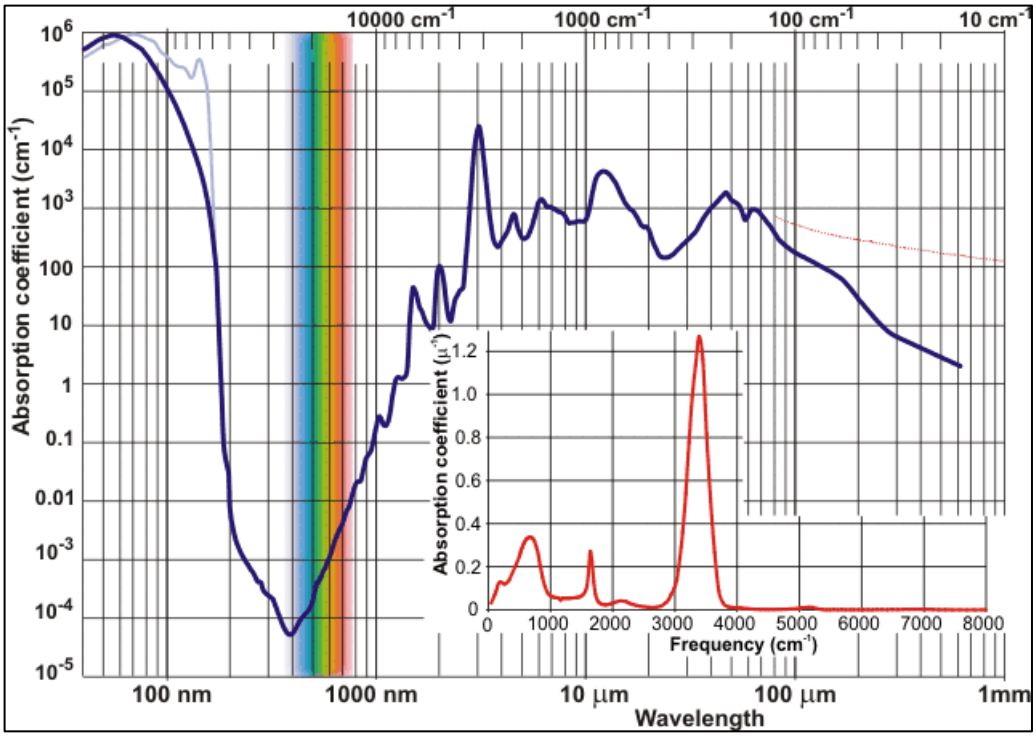
- SL353 is een digitale hall sensor die met zijn push-pull output een interrupt genereert wanneer er een sterk magneetveld aanwezig is. Deze trigger wordt gebruikt om de Carambola2 in te schakelen wanneer de data moet worden overgedragen. Deze sensor doet polling bij een duty cycle van 0.013% wat neerkomt op een gemiddeld stroomverbruik van 1.8µA.

- ISL12020M is een Real Time Clock (RTC) die eveneens via I²C adresseerbaar is. Deze chip houdt de timestamp bij voor de metingen terwijl de MCU in sleep-mode is. Een externe CR2032 batterij dient als backup-power wanneer de hoofdvoeding afgekoppeld wordt.

- Debug-led, voorzien ter indicatie bij deployment of debugging.

- Groene led 550nm, voorzien langs beide zijden van de PCB ter indicatie dat de deployment-mode succesvol is ingezet. Het menselijke oog is het meest gevoelig voor groen licht, een groene led kan dus met minder stroom even fel lijken als eender welke andere kleur.

- Blauwe led op 450nm, wordt gebruikt met een draaggolf. Deze draaggolf wordt opgepikt door de ledbar waardoor deze zijn trigger ontvangt om de led aanlichtbalk te activeren. In onderstaand absorptie diagram van water blijkt dat deze golflengte het minste gedempt wordt in water waardoor dit een evidente keuze is om datacommunicatie op te gaan doen.



Figuur 19: Grafiek absorptiecoöficient in functie van golflengte

- TSOP32238 IR ontvanger, eveneens voorzien langs beide zijden van de PCB. Deze dient eveneens als trigger zodat de Wifi communicatie kan gestart worden met zowel een magneetveld (hall-sensor) als een RC5 afstandsbediening

- FT232 USB com poort interface, deze wordt gebruikt voor debugging doeleinden en prototyping zodat via een terminalprogramma de status van de sedimentmeter kan opgevolgd worden.

4.1.2 Controller PCB

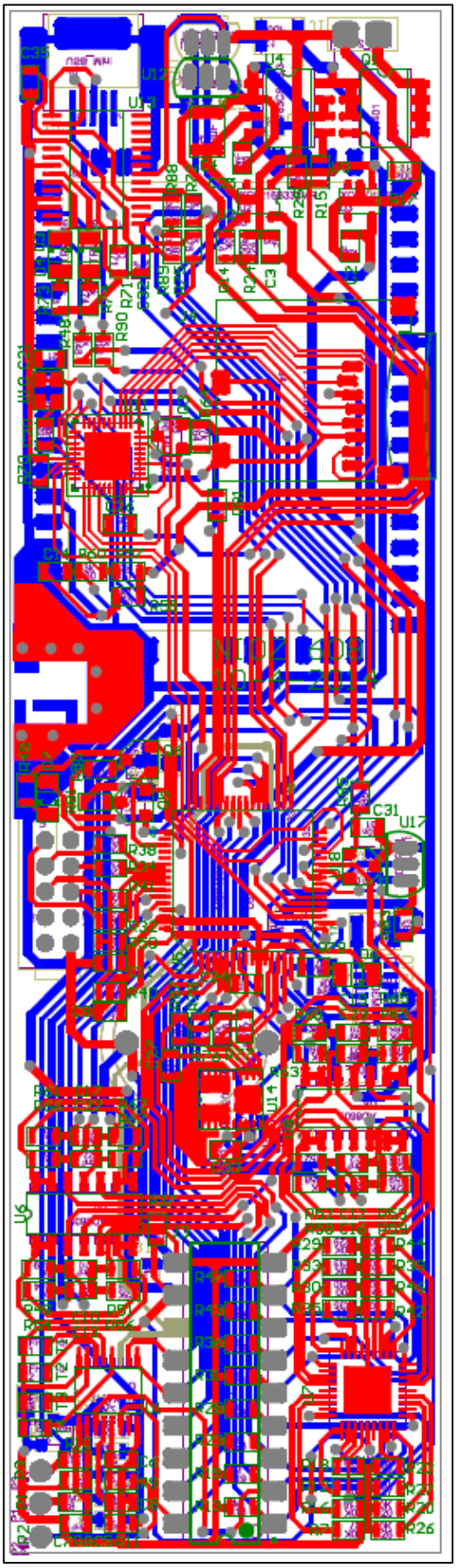
De grootte van de PCB is 33.6*120mm.
Hierop bevinden zich:

- 36 condensatoren
- 7 (light emitting) diodes
- 5 connectoren
- 6 mosfets
- 91 weerstanden
- 4 foto transistors
- 18 Integrated Circuits (IC's)

Gezien de grote hoeveelheid componenten op een beperkte oppervlakte is gekozen voor een bijna volledige SMD print met zo klein mogelijke componenten en een dubbelzijdige routing.

De voedingslijnen zijn 0.6mm breed, dit laat een maximale stroom toe van 700mA bij een temperatuursverhoging van 10ºC.
De meeste datalijnen zijn 0.2mm breed, wat een maximale stroom van 250mA toelaat.

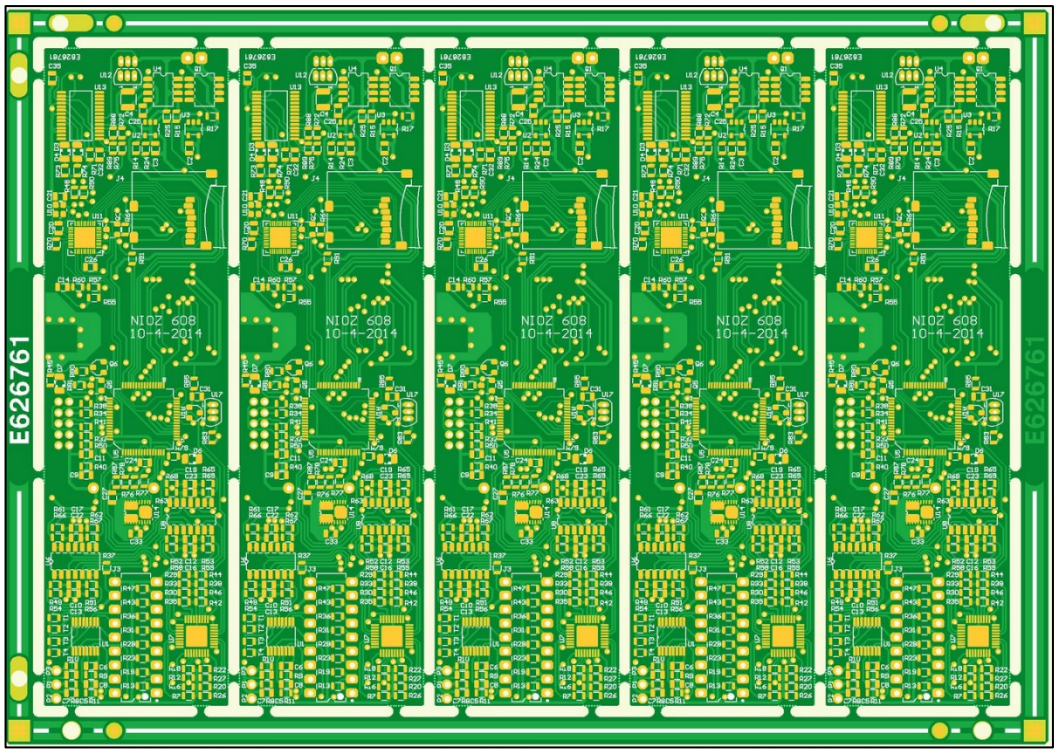
Links in het midden zit een transmissielijn voor de wifi chip antenne. Deze is volledig omsloten door een groundvlak zodat de lijn genormaliseerd blijft op een impedantie van 50Ω en er geen HF storing wordt uitgestraald naar de rest van de PCB.
Om storingen tegen te gaan is ook de klokgenerator voor de MAX14502 voorzien van een omliggend groundvlak.



Figuur 20: PCB Sedimentmeter

4.1.2.1 PCB Productie – Eurocircuits

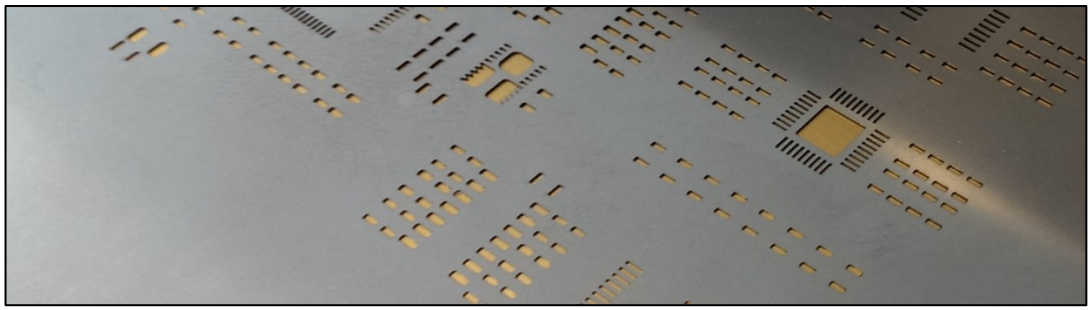
De PCB's voor de prototypes van de sedimentmeter zijn geproduceerd door Eurocircuits. Zij zijn niet de goedkoopste op de markt, maar dit maken ze ruimschoots goed door uitgebreide service, opmerkingen en feedback bij het productieproces. De eerste versie van de vernieuwde sedimentmeter PCB is in 5-voud gepaneliseerd uitgevoerd. Een voorbeeld hiervan ziet u hieronder.



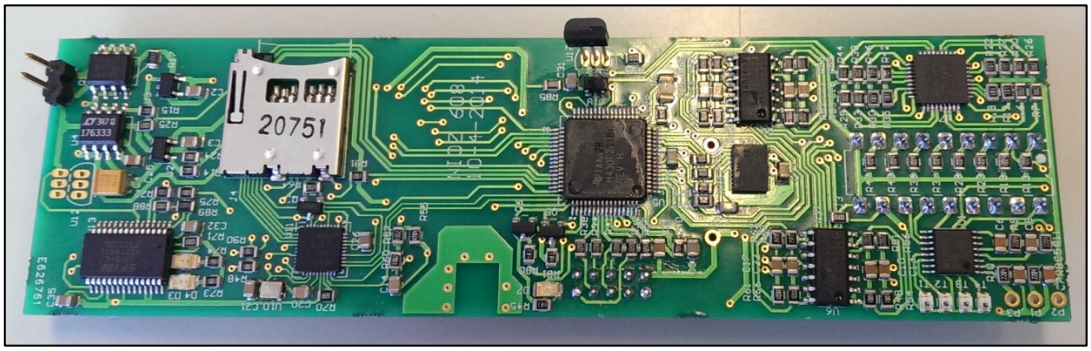
Figuur 21: Rendering PCB door EuroCircuits

4.1.2.2 Assemblage PCB

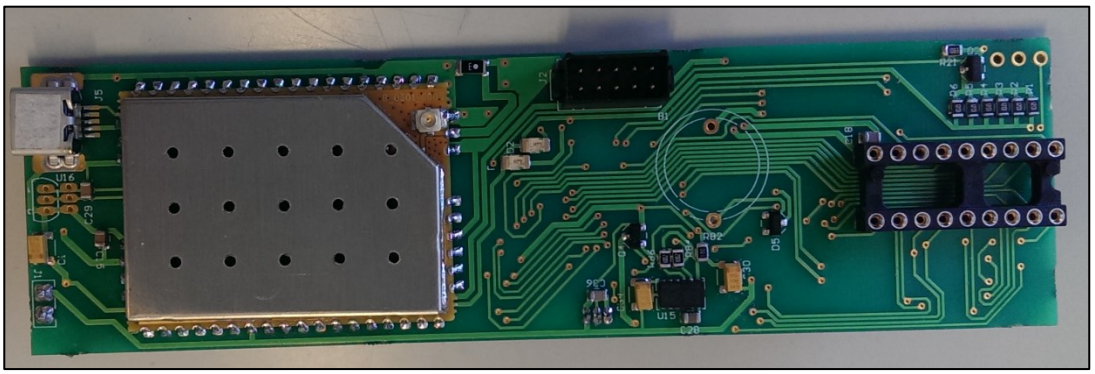
Aangezien de pcb voornamelijk uit SMD componenten bestaat, waarvan 2 TQFN 40 packages met een pitch van 0.4mm is er gebruik gemaakt van een stencil om de soldeerpasta te voorzien. Hiermee kan veel nauwkeuriger gewerkt worden dan met een dispenser waardoor de kans op soldeerbruggen tussen de pads veel kleiner is. De stencil heeft een dikte van 130µm, de kleinste pad heeft een afmeting 0.2mm*0.4mm. Dit resulteert in een volume soldeerpasta van 5,2µm³. De 2^{e} stap is het plaatsen van de componenten, hiervoor is een vacuüm pick&place machine gebruikt, dit maakt het plaatsen van de kleinste componenten een stuk eenvoudiger. Tenslotte is de PCB gesoldeerd in een reflow oven met configureerbaar temperatuurverloop. Het configureren hiervan is nodig gezien de verschillende variabelen van de PCB waaronder: dikte, aantal koperlagen, totale oppervlakte, soort soldeerpasta, etc.



Figuur 22: Soldeerpasta Stencil



Figuur 23: Sedimentmeter PCB Top-Layer



Figuur 24: Sedimentmeter PCB Bottom-Layer

4.2 Software

De source code is geschreven in C-code. Een deel hiervan is overgenomen uit eerdere versies van de sedimentmeter, andere delen zoals de aansturing voor de digitale periferie en de Wifi interface waren niet voorhanden en zijn specifiek geschreven voor de vernieuwde hardware.

Hieronder vind u een beknopte samenvatting met bijhorende flowchart ter verduidelijking van de werking. De werkelijke code hierachter vind u in de bijlagen op bladzijde 44.

4.2.1 Menu

De menustructuur is d.m.v. een structure tabelgewijs opgebouwd en opgeslagen in het eeprom geheugen. Op deze manier kan de menuboom gemakkelijk uitgebreid worden zonder het RAM geheugen van de controller onnodig te belasten.

```
const struct menuitem main_menu[] = {
        { "LList directory",            menu_fs_listdir                 },
        { "VView file contents",        menu_fs_listfile                },
        { "HHexdump file",              menu_fs_hexdump                 },
        { "FFormat SD-card",            menu_fs_format                  },
        { "RReload SD-card",            menu_fs_reload                  },
        { "PParameter menu",            deploy_parameters               },
        { "MMeasurements by labview",   deploy_labview                  },
        { "IPrint version",             print_version                   },
        { "\001Start deployment now",   worker_menu_start               },
        { "\030Expert",                 menu_enter,     menu_expert     },
        { 0 },
};
```

In de linkse kolom staat de ASCII string die via de serieële poort in een terminalprogramma zoals TeraTerm weergegeven wordt. De eerste letter van deze string is de shortcut om dit commando uit te voeren. In de de rechtse kolom staat het commando dat bij deze ASCII string hoort.

```
while (1) {
        if (worker_getstate()) {
                worker_call();
        } else {
                worker_print_info();
                menu_enter(main_menu);
                printf("\n");
        }
}
```

Het uitvoeren van de worker heeft voorrang op de menustructuur. Wanneer dit menu dus wordt afgebroken (bv vanuit een interrupt routine of vanuit het menu zelf) en een end of file genereert dan neemt de worker over en gaat deze zijn opgegeven taak uitvoeren.

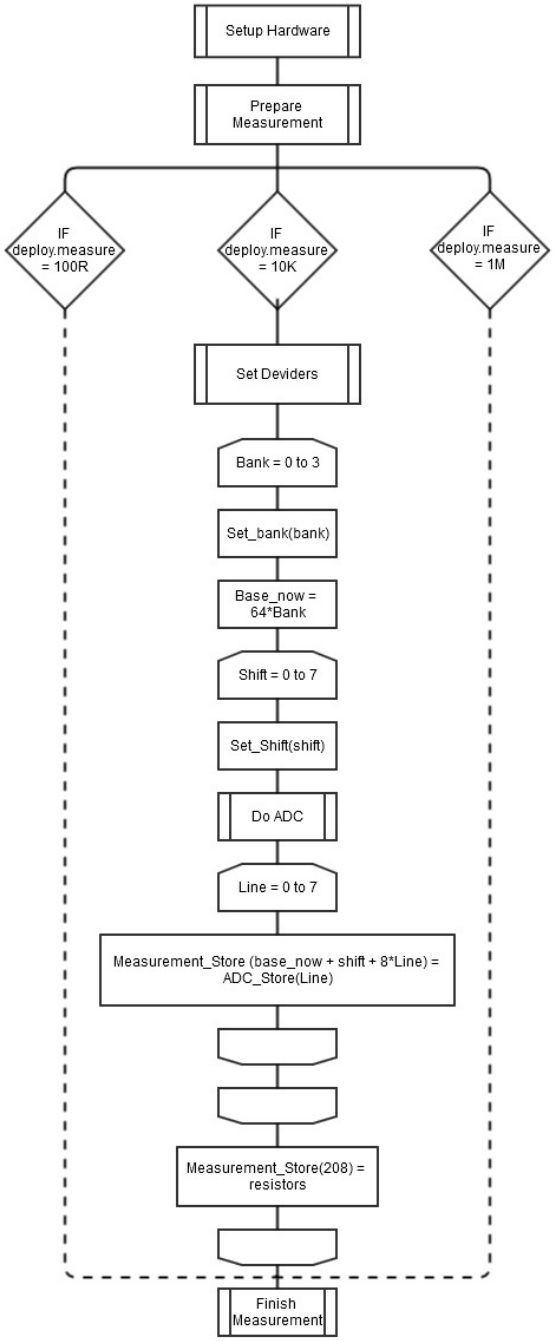
4.2.2 Metingen

Wanneer de worker een meting vraagt is het eerste wat er gebeurt de hardware initialiseren. In deze subroutine worden de tristates van de poorten verbonden met de analoge elektronica aangepast naar hun functie.

Wanneer dit voltooid is wordt de meting voorbereid. In deze subroutine wordt de voeding voor de analoge elektronica aangezet en worden de ADC kanalen geïnitialiseerd en van een referentiespanning voorzien.

Daarna worden alle sensoren 3 keer gesampled. Telkens door een andere shuntweerstand om zo gegevens in 3 schaalbereiken te bekomen. Als eerste worden de divider weerstanden ingeschakeld. Dit zijn de weerstanden die het schaalbereik bepalen (zie blz. 22, figuur 12). Daarna worden de sensoren d.m.v. een nested-for loop alle 200 sequentieel ingelezen en opgeslagen op de SD kaart.
Plaats 201 tot en met 207 zijn voorzien voor referentiesensoren op de controllerprint alsook 3 externe analoge kanalen.
Op plaats 208 van de array wordt vermeld over welke shuntweestand deze waarden gemeten zijn.

Tenslotte wordt de meting beëindigd. Dit omvat het uitschakelen van de voeding voor de sensoren en het herstellen van de IO lijnen in hun oorspronkelijke staat.



Figuur 25: Flowchart Metingen

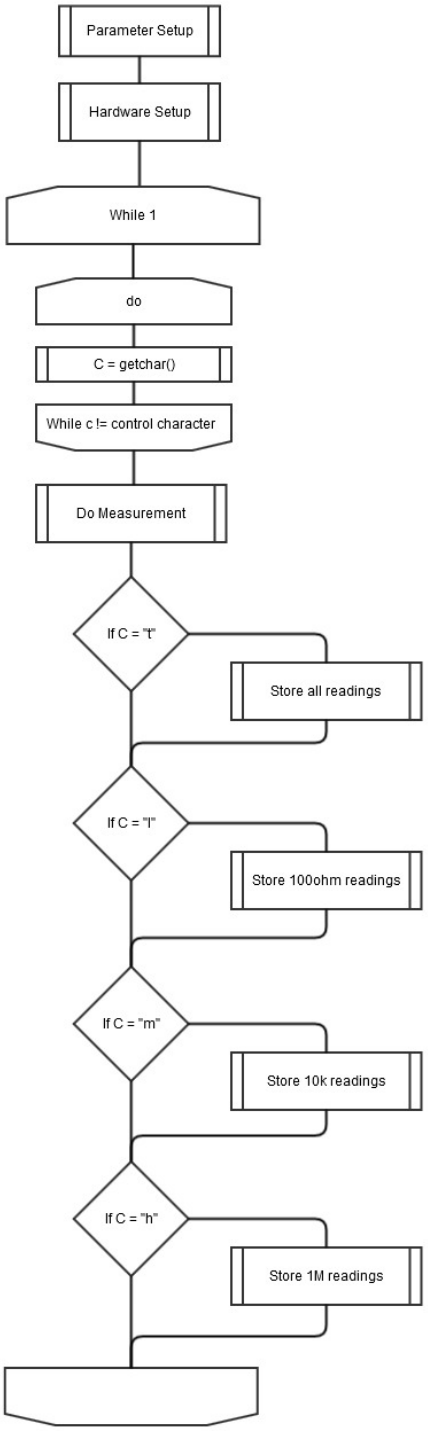
4.2.3 Labview

Vooraleer de sedimentmeters in deployment gaan worden ze via een Labview applicatie getest op een goede werking. Deze verbinding gebeurt net zoals de menustructuur via de RS232 lijn met een FT232 usb com poort emulator IC.
Aangezien zowel het menu als labview dezelfde bus delen kan er maar een van deze 2 in op het zelfde moment in gebruik zijn. Als eerste dient de sedimentmeter in de Labview mode te worden gezet via een terminalprogramma, om daarna de verbinding vrij te geven zodat het Labview programma via deze bus de sensor kan uitlezen.

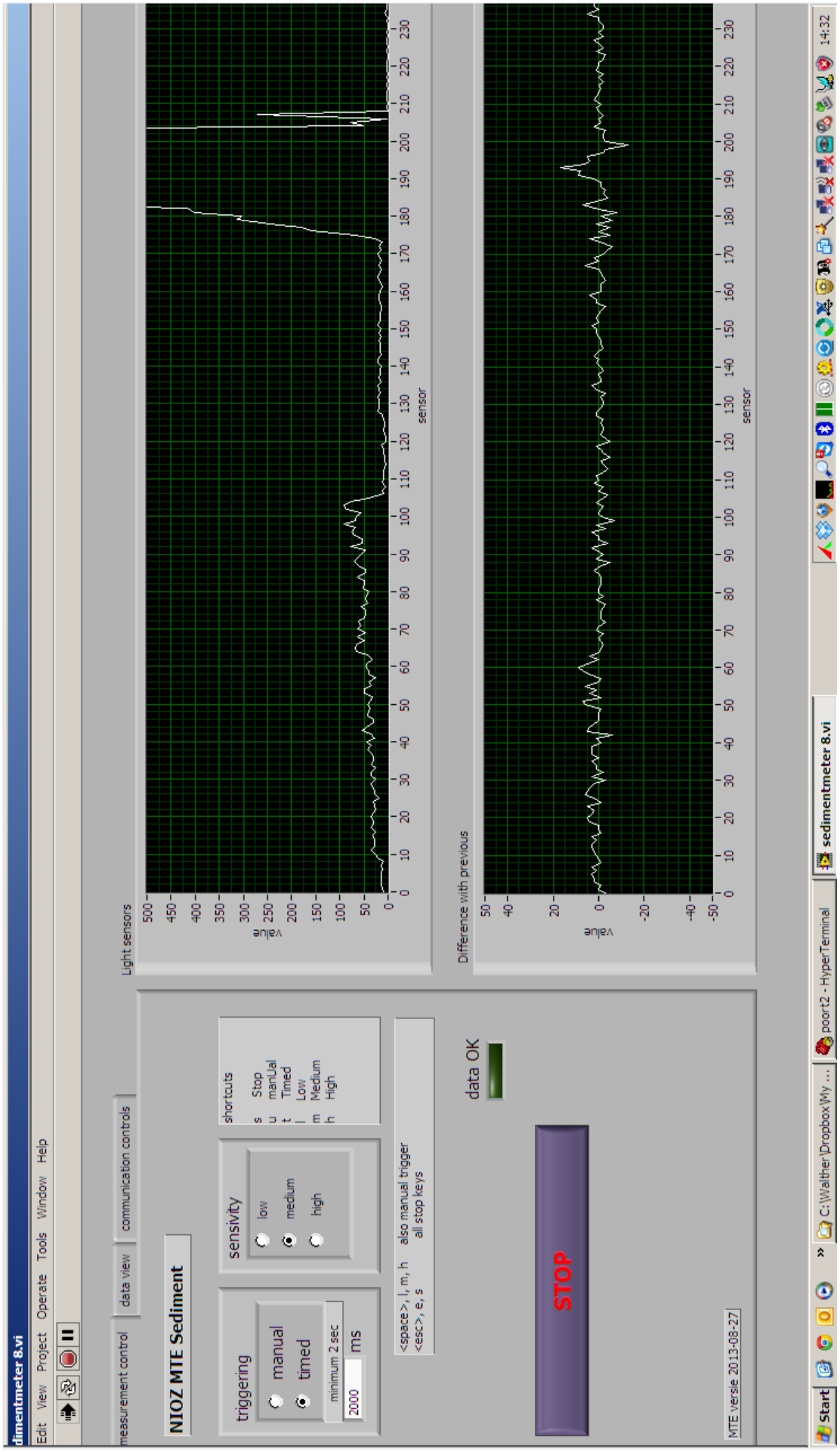
Hierna volgt der hardware setup waar de IO poorten geconfigureerd worden, deze functie wordt ook gebruikt bij configureren van de standalone meting.

Daarna komt het programma in de loop waar telkens karakters worden ingelezen, deze worden door de Labview applicatie op de serieële lijn gezet. Vervolgens worden de metingen uitgevoerd(deze staan verder beschreven op bladzijde 34 punt 4.2.2). Wanneer een controlekarakter t,l,m of h wordt ingelezen gaat een IF structuur bepalen welke actie uitgevoerd wordt. Dit kan het wegschrijven van alle metingen zijn, of slechts een van de 3.

De afbeelding op bladzijde 36 toont hoe de gebruikersinterface van de Labview er uit ziet en welke mogelijkheden deze heeft.



Figuur 26: Flowchart Labview



Figuur 27: Labview User Interface (Walther Lenting, 2014)

4.3 Tests

Om een correcte werking van de sedimentmeter te garanderen is het nodig om bepaalde componenten aan tests te onderwerpen zodat we zeker weten dat alles naar behoren blijft functioneren bij de deployment. Het is niet altijd mogelijk om voort te gaan op de datasheets van deze componenten omdat de fysieke omstandigheden waar deze gaan gebruikt worden niet altijd beschreven zijn.

4.3.1 IR transmissie TSOP32238 door water

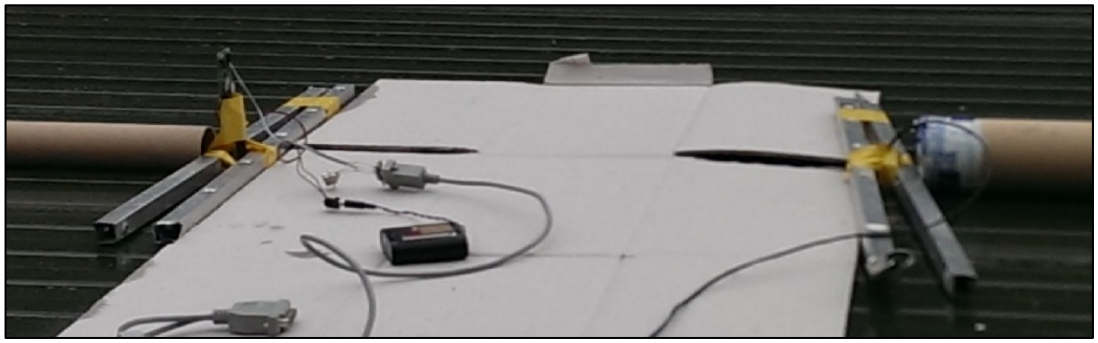
Aangezien de Carambola2 Wifi module 200mA gebruikt wanneer deze ingeschakeld is, is er gekozen om de LT1763 3.3V LDO hiervan uit te schakelen tot er een Wifi verbinding nodig is. Hiervoor is een externe trigger nodig die door de MSP herkend wordt en deze de voeding aan schakelt. Deze trigger kan komen van de hall-sensor door met een magneet langs de sensor te gaan of vanuit een TSOP 32238 IR ontvanger door deze met een IR afstandsbediening aan te stralen. Het probleem van IR transmissie door water is de sterke absorptie van alle golflengtes buiten het zichtbare spectrum (zie blz. 28 figuur 18).

De testopstelling hiervoor was een TSOP 32238 IR ontvanger 20cm onder het water oppervlak op 40cm van een IR zender. Uit onderstaande scoopbeelden blijkt dat op deze afstand de demping van het zeewater geen probleem is voor het ontvangen van dit signaal. Dit komt door de ingebouwde Automatic Gain Control (AGC) in de TSOP die zijn versterking automatisch aanpast aan het ontvangen signaal.

4.3.2 Test 830nm IR leds + fototransistoren door water

Een van de producteisen voor de sedimentmeter opstelling was dat deze ’s nachts niet zou opvallen. Het gebruik van zichtbare blauwgroene leds op een golflengte van 525nm is dus geen optie.(deze golflengte is het minst onderhevig aan demping: zie figuur 18 op blz 28) Als alternatief is gekozen voor leds in het Near-IR spectrum met een bundel van 6°. Als gevolg van deze aanpassing zijn ook de ontvangende fototransistoren aangepast naar types die ook gevoelig zijn bij deze golflengte. Deze vernieuwde opstelling is getest op een goede werking in een afgedekt testbassin op een sensordiepte van 20 tot 40cm, met een zender-ontvanger afstand van 60cm om een meting tijdens de nacht wanneer deze IR ledstraler nodig is te simuleren.

Uit deze test is gebleken dat de nieuwe zender-ontvanger combinatie goed werkt en mogelijks zelfs gevoeliger is dan de vorige versie hiervan.

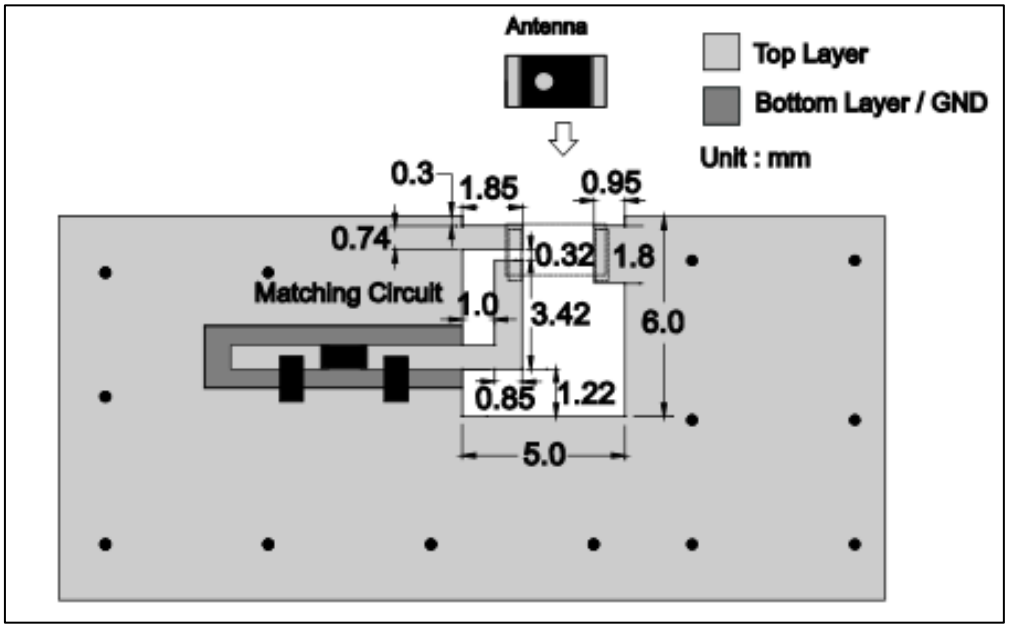


Figuur 28: Opstelling test IR led + fototransistor

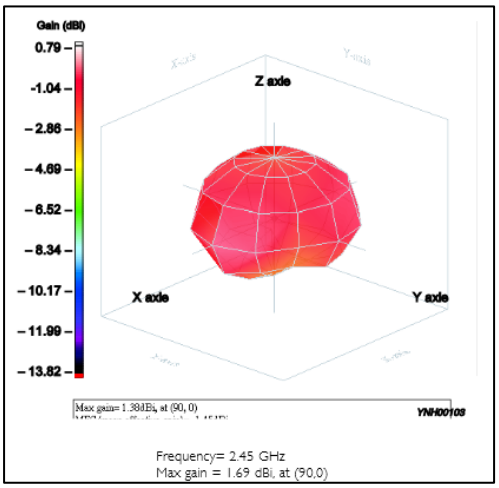
4.3.3 Carambola 2, 2.4GHz chip-antenne signaalsterkte

Een 3e test is pas uitgevoerd na het assembleren van de sedimentmeter. Dit betreft het introduceren van een 2.4GHz chip-antenne die strikte eisen aan het PCB design stelt. De minste onvolkomenheid in het ontwerp hiervan zou zware gevolgen hebben voor de signaalsterkte die met dit design kan gerealiseerd worden.

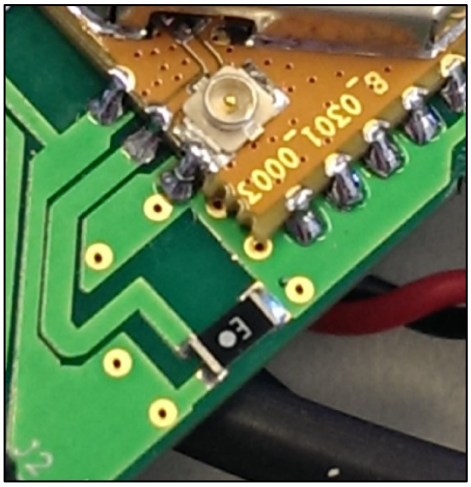
Resultaat: Het chip-antenne design werkt naar behoren, een afstand van ±10m doorheen verschillende muren levert slechts een minimale verzwakking van de signaalsterkte op.



Figuur 29: Design specs chip antenne



Figuur 30: Stralingsdiagramma chip-antenne



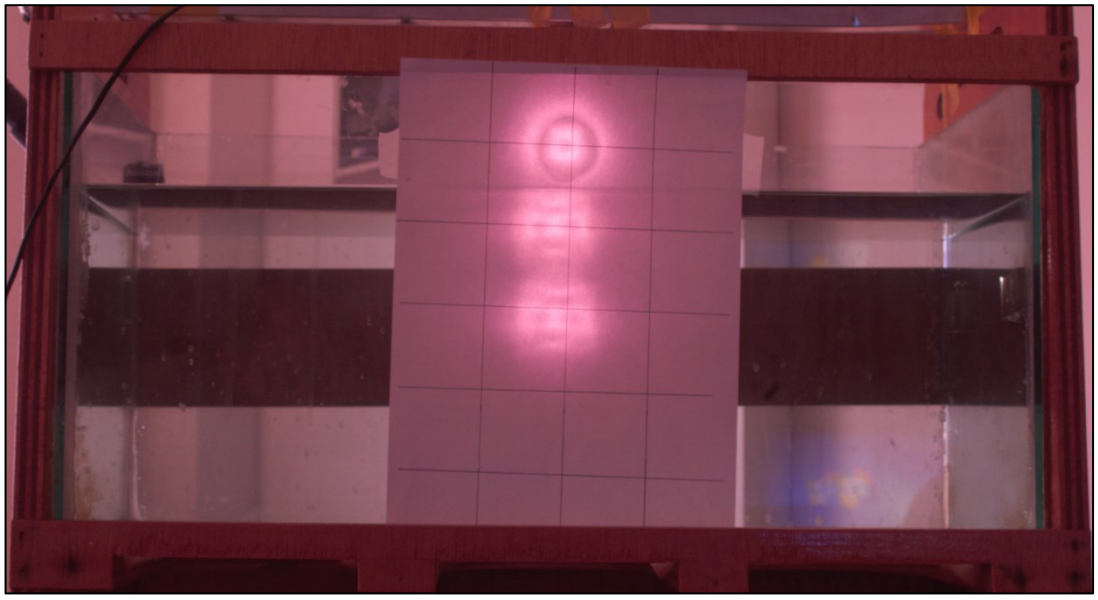
Figuur 31: PCB layout chip-antenne

4.3.4 Stralingspatroon LED bar

Het stralingsbeeld van de lichtbundel die de ledbar projecteert op de fototranistoren is afhankelijk van de individuele stralingshoek van de leds. Ideaal is dit een volledig egale balk met een hoogte van 40cm en een 5 tal cm breed. De testopstelling op onderstaande foto betreft 3 830nm IR leds met een bundel van 6° op een onderlinge afstand van 5cm. De bovenste led bevindt zich boven het wateroppervlak, de overige 2 eronder. Het gebruikte grid is 5cm.

Uit dit stralingsbeeld blijkt dat deze leds niet geschikt zijn aangezien ze niet voldoende overlappen, maar wat belangrijker is: dat ze niet egaal zijn.

Onderstaande foto is gemaakt met een gemodificeerde DSRL camera, met het IR sperfilter over de sensor verwijdert.



Figuur 32: Stralingspatroon 6° IR leds

BESLUIT

Het doel van deze 13 weken stage was het doorontwikkelen van de sedimentmeter. Het doorlopen van dit hele proces omvatte onder andere het bestuderen van de oude hardware, uitzoeken van componenten voor de nieuwe, het tekenen van zowel een schema als de bijhorende pcb, het aanpassen van de software, het bestukken van de pcb en het uitgebreid testen van de hele opstelling. Uit deze tests is gebleken dat de sedimentmeter naar behoren werkt en na enkele kleine aanpassingen was deze klaar om in productie genomen te worden.

Het uitwerken van dit project was dan ook een periode waarin ik veel bijgeleerd heb op vlakken waarvan ik al een bepaalde kennis had, maar ook op andere gebieden waar ik tot geheel nieuwe inzichten ben gekomen. Denk hierbij aan het oplossen van power-on verschijnselen, HF PCB design en het omgaan met parasitaire eigenschappen van componenten. Het was dan ook de uitgelezen kans om de tijdens mijn opleiding vergaarde kennis in de praktijk om te zetten.

LITERATUURLIJST

M. Chaplin. (2014, 18 Mei). *Water Absorption Spectrum* http://www1.lsbu.ac.uk/water/vibrat.html

IAR Systems. (2014, Februari). *IAR for MSP430 Tutorials* *http://supp.iar.com/FilesPublic/UPDINFO/005316/infocenter/tutorials.ENU.html*

Altium. (2012). *Altium Designer: Schematic Capture and PCB Editing Training*

van Maarseveen F. – NIOZ.(2014) *Software Sedimentmeter*

Lenting W. – NIOZ. (2014) *Labview applicatie Sedimentmeter*

Laan M. – NIOZ. (2014) *Hardware oudere type(s) Sedimentmeter*

5 BIJLAGEN

5.1 Bill of Materials

Comment	Designator	Footprint	LibRef	Quantity
3V	B1	3V Bat	3V_bat	1
10uF	C1, C4, C30, C34	CT_B	CT_B	4
1uF	C2, C3, C9, C15, C18, C21, C24, C26, C32,C33, C35, C36	C0805	C1206	12
10nF	C5, C6, C7, C8, C10, C12, C13, C16, C17, C19, C22, C23	C0805	C1206	12
47n	C11	C0805	C0805	1
100n	C14, C27, C31	C0805	C0805	3
100nF	C20, C29	C0805	C1206	2
220nF	C28	C0805	C1206	1
Led Green	D1, D7	LED1206	LED_1206	2
Led Blue	D2	LED1206	LED_1206	1
KPTR-3216	D3, D4, D6	LED1206	LED_1206	3
BAT54S	D5	SOT23D	BAS31	1
HDR_2	J1	HDR2P2.54	HDR_2	1
HDR_PROG	J2	HDR_PROG	HDR_prog	1
DIL18	J3	DIP18	DIL18	1
F1572016	J4	SD-CARD_uSD	SD CARD	1
F1125348	J5	USB mini	HDR_6	1
PIN	P1, P2, P3	pin1	PIN	3
SI4401	Q1	SO8	SI4401	1
SI2308	Q2	SOT23F	SI2308	1
IRLML6402	Q3	SOT23F	IRLML6402	1
BSS84	Q4	SOT23F	BSS84	1
BSS138	Q5, Q6	SOT23F	BSS138	2
1K2	R1, R2, R3, R4, R5, R6	R0805	R0805	6
10K	R7, R14, R15, R16, R20, R26, R29, R33, R39, R44	R0805	R0805	10
2M2	R8, R9, R10, R11	R0805	R0805	4
100	R12, R18, R22, R27, R30, R35, R42, R46, R76, R77, R78, R83, R84, R85, R86	R0805	R0805	15
1M	R13, R19, R23, R28, R31, R36, R43, R47	R0805	R0805	8
100k	R17, R21, R24, R25, R88, R89	R0805	R0805	6
220	R32, R34, R38, R41	R0805	R0805	4
10E	R37	R0805	R0805	1
2k2	R40	R0805	R0805	1
68E	R45, R80, R81	R0805	R0805	3
1k	R48, R49, R52, R54, R58, R61, R63, R66, R68, R72, R73, R74, R75, R90	R0805	R0805	14
22k	R50, R51, R53, R56, R59, R62, R65, R67, R69	R0805	R0805	9
4k7	R55, R79	R0805	R0805	2
33k	R57, R60	R0805	R0805	2
47K	R64	R0805	R0805	1

6k19	R70	R0805	R0805	1
10	R71, R82	R0805	R0805	2
220k	R87	R0805	R0805	1
SFH3710	T1, T2, T3, T4	SFH3710	SFH3710	4
MAX4617	U1	TSOP16	MAX4617	1
XC6216B332MR	U2	SOT23_5	XC6216B332MR	1
XC6201P332MR	U3	SOT23_5	XC6201P332MR	1
LT1763CS8-3.3	U4	SO8	LT1763CS8-3.3	1
MSP430F2618	U5	S-PQFP-64	MSP430F2618_64_1	1
AD8609	U6, U8	SO14	AD8609	2
MAX4761	U7	TQFN36 (6*6)	MAX4761	1
CARAMBOLA 2	U9	CARAMBOLA	CARABBOLA 2	1
SG-210STF	U10	SG210	SG-210STF	1
MAX14502	U11	40TQFP	MAX14502	1
TSOP32238	U12, U16	TO92 - flat back	TSOP32238	2
FT232R	U13	SSOP28	FT232R	1
ISL12020M	U14	L20.5.5x4.0	ISL12020M	1
LSM303DLHC	U15	LGA_14L	LSM303DLHC	1
Tsic-506F	U17	TO92 - flat back	Tsic-506F	1
SL353	U18	SOT23num	SL353	1
ANT3216A063R2400A	Chip Antenna	Chip_Ant	3216	1

5.2 Code

In deze bijlagen vind u de C code, onderverdeeld in verschillende onderdelen.

5.2.1 Main.c

```
#include <stdio.h>
#include <limits.h>

#include "common/termio.h"
#include "common/menuitem.h"
#include "common/stack.h"
#include "common/bits.h"
#include "common/ts.h"
#include "common/delay.h"
#include "common/ts_timer_a.h"
#include "common/isl12020m.h"
#include "common/fat.h"
#include "common/fs.h"
#include "common/sd.h"
#include "common/worker.h"
#include "common/tsic506f.h"
#include "common/lsm303.h"
#include "hardware.h"
#include "rtc.h"
#include "deploy.h"
#include "hall.h"
#include "wifi232.h"

static char rxbuf[80], txbuf[80];

static void dco_init(void)
{
    DCOCTL = CALDCO_8MHZ;
    BCSCTL1 = CALBC1_8MHZ;
    BCSCTL3 = (BCSCTL3 & ~XCAP_3) | LFXT1S_3;   // Digital input at XIN: ACLK from
ISL12020M
}

static void print_version(const struct menuitem *menu)
{
    printf("Version 2013/05/23 (built on %s %s)\n", __DATE__, __TIME__);
}

const struct menuitem main_menu[] = {
    { "LList directory",        menu_fs_listdir             },
    { "VView file contents",    menu_fs_listfile            },
    { "HHexdump file",      menu_fs_hexdump             },
    { "FFormat SD-card",        menu_fs_format              },
    { "RReload SD-card",        menu_fs_reload              },
    { "PParameter menu",        deploy_parameters           },
    { "MMeasurements by labview",   deploy_labview              },
    { "IPrint version",     print_version               },
    { "\001Start deployment now",   worker_menu_start           },
    { "\030Expert",         menu_enter, menu_expert     },
    { 0 },
};

ISR(PORT1_VECTOR)
{
    tsic506f_p1_interrupt();
}

ISR(PORT2_VECTOR)
{
    hall_p2_interrupt();
    wifi232_p2_interrupt();
}

void main(void)
{
    WDTCTL = WDTPW + WDTHOLD;
    stack_monitor();
    do_eint();
    dco_init();
```

```
    WHITE_LED_OFF;
    OUTPUT(WHITE_LED);
    SENSOR_POWER_OFF;
    OUTPUT(SENSOR_POWER);
    IRBLUE_LED_OFF;
    OUTPUT(IRBLUE_LED);

    ts_timer_a();
    ts_waitms(50);          /* ACLK startup, required before delay timing */
    delay_init();
    sd_init();
    termio_init(&uart0, rxbuf, sizeof (rxbuf), txbuf, sizeof (txbuf));
    termio_on();
    printf("\n\n");
    delay_print();
    isl12020m_init(&isl12020m_phy);
    isl12020m_restore(&isl12020m_phy);
    fs_init();
    if (!lsm303_config(&lsm303_phy, LSM303_CONFIG_ALL))
        printf("LSM303: error=%04x\n", lsm303_errorbits);
    lsm303_suspend(&lsm303_phy);
    tsic506f_init();
    worker_init();
    deploy_init();
    wifi232_init();
    hall_init();

    while (1) {
        if (worker_getstate()) {
            worker_call();
        } else {
            worker_print_info();
            menu_enter(main_menu);
            printf("\n");
        }
    }
}
```

5.2.2 Deploy.c

```
#include <stdio.h>
#include <string.h>
#include <limits.h>

#include "common/termio.h"
#include "common/menu.h"
#include "common/ts.h"
#include "common/ts_timer_a.h"
#include "common/file.h"
#include "common/fat.h"
#include "common/isl12020m.h"
#include "common/param.h"
#include "common/flash.h"
#include "common/worker.h"
#include "common/tsic506f.h"
#include "common/lsm303.h"
#include "hardware.h"
#include "measure.h"
#include "hall.h"
#include "wifi232.h"

#include "deploy.h"
struct deploy deploy;

#define PARAMETERS ts_param, worker_param, deploy_param, wifi232_param

static const struct param deploy_param[] = {
    { .name = "NID",                .type = PT_UINT16,  .ptr = &deploy.id, .max = 999
},
    { .name = "Iinterval",              .type = PT_TIMEDIFF,    .ptr =
&deploy.interval, .min = 2000 },
    { .name = "2100 Ohm measurement",       .type = PT_BOOL,    .ptr =
&deploy.measure_100 },
    { .name = "410k Ohm measurement",       .type = PT_BOOL,    .ptr =
&deploy.measure_10k },
    { .name = "61M Ohm measurement",        .type = PT_BOOL,    .ptr =
&deploy.measure_1M },
    { .name = "LLog data in file",          .type = PT_BOOL,    .ptr =
&deploy.file_logging },
    { .name = "LRS232 logging (0=off, 1=txt, 2=bin",.type = PT_UINT8,   .ptr =
&deploy.rs232_logging, .min = 0, .max = 2 },
    { 0 }
};

static struct flash flash_id = { .data = &deploy.id, .size = sizeof (deploy.id),
.offset = 4 };
static struct time  timestamp;
static uint16_t     temperature;
static int16_t      accel_xyz[3];
static int16_t      compass_xyz[3];

void deploy_parameters(const struct menuitem *menu)
{
    uint8_t level;

    level = 0;
    if (menu_uint8("Parameter level", &level, 0, 0, true) != EOF) {
        worker_schedule(false);
        param_menu(level, PARAMETERS, NULL);
        while (ts_time(NULL).ms > 10)
            ; /* try to set RTC around second boundary */
        isl12020m_save(&isl12020m_phy);
        flash_save();
        worker_schedule(true);
        if (!TSTIN(HALL_SIGNAL))
            printf("Error: hall sensor signal inversion or magnet detected!\n");
    }
}

static void deploy_write_item(int fd, uint16_t item)
{
    uint8_t b;

    b = item >> 8;
```

```
    if (deploy.file_logging)
        file_write(fd, &b, 1);
    if (deploy.rs232_logging == 2)
        putchar(b);
    b = item & 255;
    if (deploy.file_logging)
        file_write(fd, &b, 1);
    if (deploy.rs232_logging == 2)
        putchar(b);
}

static void deploy_write(int fd, const uint16_t *buf, int nelem)
{
    static struct time ts;
    uint16_t sample;
    int i, order;
    char buf2[TSIC506F_SZ];

    if (deploy.rs232_logging == 1) {
        for (i = 0; i < SENSORS; ++i) {
            sample = buf[i];
            order = 0;
            while (sample) {
                ++order;
                sample >>= 1;
            }
            putchar("_-^123456789X???"[order]);
        }
        tsic506f_sample2str(temperature, buf2, sizeof (buf2));
        printf("T = %s°C\r\n", buf2);
    }
    for (i = 0; i < nelem - (1 + 3 + 3 + 3); ++i)
        deploy_write_item(fd, buf[i]);
    deploy_write_item(fd, temperature);
    deploy_write_item(fd, accel_xyz[0]);
    deploy_write_item(fd, accel_xyz[1]);
    deploy_write_item(fd, accel_xyz[2]);
    deploy_write_item(fd, compass_xyz[0]);
    deploy_write_item(fd, compass_xyz[1]);
    deploy_write_item(fd, compass_xyz[2]);
    deploy_write_item(fd, timestamp.s >> 16);
    deploy_write_item(fd, timestamp.s & 0xffff);
    deploy_write_item(fd, timestamp.ms);
    if (ts_time(&ts).s > 3600) {
        if (deploy.file_logging)
            file_flush(fd);
        ts = ts_time(NULL);
    }
}

static void deploy_start(void)
{
    char fname[8 + 1 + 3 + 1];
    struct termio_mode tmode;

    if (deploy.file_logging) {
        sprintf(fname, "%03u_????.sed", deploy.id);
        deploy.fd = file_open(fname, FILE_WRITE | FILE_CREATE | FILE_NUMBERED);
        if (deploy.fd == -1)
            printf("Cannot open logfile, continuing anyway.\n");
    }
    Setup_Hardware();
    if (deploy.rs232_logging == 2) {
        /*
         * hack for labview: put uart in raw mode. The worker module will
         * restore termio mode for us anyway at the end so no need to do
         * this in deploy stop().
         */
        termio_get(&tmode);
        tmode.crlf = false;
        termio_set(&tmode);
    }
}

static void deploy_wakeup(struct time *next)
{
    Do_Measurement();
```

```
    timestamp = ts_time(NULL);
    temperature = tsic506f_sample();
    lsm303_resume(&lsm303_phy);
    lsm303_read_accelerometer(&lsm303_phy, accel_xyz);
    lsm303_read_magnetometer(&lsm303_phy, compass_xyz);
    lsm303_suspend(&lsm303_phy);
    if (deploy.measure_100)
        deploy_write(deploy.fd, measure_store_100_Ohm,
ARRAYSIZE(measure_store_100_Ohm));
    if (deploy.measure_10k)
        deploy_write(deploy.fd, measure_store_10k_Ohm,
ARRAYSIZE(measure_store_10k_Ohm));
    if (deploy.measure_1M)
        deploy_write(deploy.fd, measure_store_1M_Ohm,
ARRAYSIZE(measure_store_1M_Ohm));
    ts_add(next, &deploy.interval);
}

static void deploy_stop(void)
{
    file_close(deploy.fd);
    deploy.fd = -1;
}

static void deploy_heartbeat_led(bool_t on)
{
    if (on)
        WHITE_LED_ON;
    else
        WHITE_LED_OFF;
}

static const struct worker_callback deploy_worker_cb = {
    .start_event    = deploy_start,
    .wakeup_event   = deploy_wakeup,
    .stop_event = deploy_stop,
    .heartbeat_led  = deploy_heartbeat_led
};

void deploy_init(void)
{
    flash_register(&flash_id);
    if (deploy.id > 999)
        deploy.id = 999;
    deploy.fd = -1;
    deploy.interval.s = 10;
    deploy.measure_100 = true;
    deploy.measure_10k = true;
    deploy.measure_1M = true;
    deploy.file_logging = true;
    flash_save();
    worker_config(&deploy_worker_cb);
}

void deploy_labview(const struct menuitem *menu)
{
    struct termio_mode tmode, oldmode;
    int old_rs232_logging;
    int c;

    old_rs232_logging = deploy.rs232_logging;
    deploy.rs232_logging = 2;
    termio_get(&tmode);
    oldmode = tmode;
    tmode.crlf = false;
    termio_set(&tmode);
    Setup_Hardware();
    while (1) {
        do {
            c = getchar();
        } while (c != ESC && c != 'e' && c != 't' && c != 'l' && c != 'm' && c !=
'h');
        if (c == ESC || c == 'e')
            break;
        Do_Measurement();
        if (c == 't') {
            if (deploy.measure_100)
```

```
                deploy_write(-1, measure_store_100_Ohm,
ARRAYSIZE(measure_store_100_Ohm));
            if (deploy.measure_10k)
                deploy_write(-1, measure_store_10k_Ohm,
ARRAYSIZE(measure_store_10k_Ohm));
            if (deploy.measure_1M)
                deploy_write(-1, measure_store_1M_Ohm,
ARRAYSIZE(measure_store_1M_Ohm));
        }
        if (c == 'l')
            deploy_write(-1, measure_store_100_Ohm,
ARRAYSIZE(measure_store_100_Ohm));
        if (c == 'm')
            deploy_write(-1, measure_store_10k_Ohm,
ARRAYSIZE(measure_store_10k_Ohm));
        if (c == 'h')
            deploy_write(-1, measure_store_1M_Ohm, ARRAYSIZE(measure_store_1M_Ohm));

    }
    termio_set(&oldmode);
    deploy.rs232_logging = old_rs232_logging;
}
```

5.2.3 Measure.c

```
#include "common/bits.h"
#include "common/ts.h"
#include "hardware.h"
#include "common/ts timer a.h"
#include "deploy.h"

#include "measure.h"
measure_store_t measure_store_100_Ohm;
measure_store_t measure_store_10k_Ohm;
measure_store_t measure_store_1M_Ohm;

static conv_store_t ADC_store;
static divider_resistor_t resistors;

static void sensor_power(bool_t on)
{
    istate_t istate;

    saveicli(istate);
    if (on)
        SENSOR_POWER_ON;
    else
        SENSOR_POWER_OFF;
    restorei(istate);
}

void Setup_Hardware(void)
{
    istate_t istate;

    saveicli(istate);
    OUTPUT(R_EN_N);
    OUTPUT(R_100);
    OUTPUT(BIT_A);
    OUTPUT(BIT_B);
    OUTPUT(BIT_C);
    OUTPUT(BANK_1);
    OUTPUT(BANK_2);
    OUTPUT(BANK_3);
    OUTPUT(BANK_4);
    CLR(R_EN_N);    // analog divider tri-state
    CLR(R_100);    // select 10k_Ohm (not really nescessary as   divider is in tri-
state)
    restorei(istate);
}

static void Prepare_Measurement(void)
{
    sensor_power(true);
    ts_waitms(1);           // 5*50us +32*4 = 370us << 1 ms
    ADC12CTL0 = ADC12ON + SHT0_8 + MSC + REF2_5V + REFON;       //  Turn on ADC12,
extend sampling time
    ADC12CTL1 = SHP + CONSEQ_1;          // Use sampling timer, repeated sequence
    ADC12MCTL0 = INCH_0 + SREF_1;        // ref+=2.5v, channel = A0
    ADC12MCTL1 = INCH_1 + SREF_1;        // ref+=2.5v, channel = A1
    ADC12MCTL2 = INCH_2 + SREF_1;        // ref+=2.5v, channel = A2
    ADC12MCTL3 = INCH_3 + SREF_1;        // ref+=2.5v, channel = A3
    ADC12MCTL4 = INCH_4 + SREF_1;        // ref+=2.5v, channel = A4
    ADC12MCTL5 = INCH_5 + SREF_1;        // ref+=2.5v, channel = A5
    ADC12MCTL6 = INCH_6 + SREF_1;        // ref+=2.5v, channel = A6
    ADC12MCTL7 = INCH_7 + SREF_1 + EOS;  // ref+=2.5v, channel = A7, end seq.
}

static void Select_Divider(divider_resistor_t divres)
{
    istate_t istate;

    saveicli(istate);
    switch (divres) {
    case _1M_Ohm:
        SET(R_EN_N);
        break;
    case _10k_Ohm:
        CLR(R_EN_N);
```

```
        CLR(R_100);
        break;
    case _100_Ohm:
        CLR(R_EN_N);
        SET(R_100);
        break;
    }
    restorei(istate);
}

static void Finish_Measurement(void)
{
    istate_t istate;

    sensor_power(false);
    Select_Divider(_1M_Ohm);
    saveicli(istate);
    CLR(BIT_A);
    CLR(BIT_B);
    CLR(BIT_C);
    SET(BANK_1);
    SET(BANK_2);
    SET(BANK_3);
    SET(BANK_4);
    restorei(istate);
    ADC12CTL0 = 0;
}

static void Do_ADC(void)
{
    ADC12CTL0 |= ENC;            // Enable conversion
    ADC12CTL0 |= ADC12SC;        // Start conversion
    while ((ADC12IFG & BIT7) == 0)
        ;                        // wait for the 8 channels to be sampled
    ADC12CTL0 &= ~ENC;           // stop conversion
    ADC_store[0] = ADC12MEM0;
    ADC_store[1] = ADC12MEM1;
    ADC_store[2] = ADC12MEM2;
    ADC_store[3] = ADC12MEM3;
    ADC_store[4] = ADC12MEM4;
    ADC_store[5] = ADC12MEM5;
    ADC_store[6] = ADC12MEM6;
    ADC_store[7] = ADC12MEM7;
}

static void Set_Bank(uint8_t B)
{
    uint8_t p_low,p_high;
    istate_t istate;

    saveicli(istate);
    p_low = P4OUT & 0x0F;        // lines P4_0 -- P4_4 must be invariable
    p_high = ~(1<<(4+B));        // inverted enable, lines P4_4 -- P4_7
    p_high = p_high & 0xF0;      // zero the lines P4_0 -- P4_3
    Select_Divider(_100_Ohm);
    P4OUT = p_low | p_high;
    Select_Divider(resistors);   // testing purpose
    restorei(istate);
}

static void Set_Shift(uint8_t S)
{
    uint8_t p_low,p_high;
    istate_t istate;

    p_low = S;                   // lines P4_0 -- P4_3 0..7
    saveicli(istate);
    p_high = P4OUT & 0xF8;       // lines P4_3 -- P4_7 must be invariable
    Select_Divider(_100_Ohm);
    P4OUT = p_low | p_high;
    restorei(istate);
    Select_Divider(resistors);
}

void Do_Measurement(void)
{
    uint8_t bank, shift, line, base_now;
```

```
    Prepare_Measurement();

    if (deploy.measure_100) {
        resistors = _100_Ohm;
        Select_Divider(resistors);
        for (bank = 0; bank <= 3; bank++) {
            Set_Bank(bank);
            base_now = 64 * bank;
            for (shift = 0; shift <= 7; shift++) {
                Set_Shift(shift);
                udelay(_100_Ohm_delay);
                Do_ADC();
                for (line = 0; line <= 7; line++) {
                    measure_store_100_Ohm[base_now + shift + 8 * line] =
ADC_store[line];
                }
            }
            measure_store_100_Ohm[208] = resistors;
        }
    }

    if (deploy.measure_10k) {
        resistors = _10k_Ohm;
        Select_Divider(resistors);
        for (bank = 0; bank <= 3; bank++) {
            Set_Bank(bank);
            base_now = 64 * bank;
            for (shift = 0; shift <= 7; shift++) {
                Set_Shift(shift);
                udelay(_10k_Ohm_delay);
                Do_ADC();
                for (line = 0; line <= 7; line++) {
                    measure_store_10k_Ohm[base_now + shift + 8 * line] =
ADC_store[line];
                }
            }
            measure_store_10k_Ohm[208] = resistors;
        }
    }

    if (deploy.measure_1M) {
        resistors = _1M_Ohm;
        Select_Divider(resistors);
        for (bank = 0; bank <= 3; bank++) {
            Set_Bank(bank);
            base_now = 64 * bank;
            for (shift = 0; shift <= 7; shift++) {
                Set_Shift(shift);
                ts_waitms(_1M_Ohm_delay);
                Do_ADC();
                for (line = 0; line <= 7; line++) {
                    measure_store_1M_Ohm[base_now + shift + 8 * line] =
ADC_store[line];
                }
            }
            measure_store_1M_Ohm[208] = resistors;
        }
    }

    Finish_Measurement();
}
```