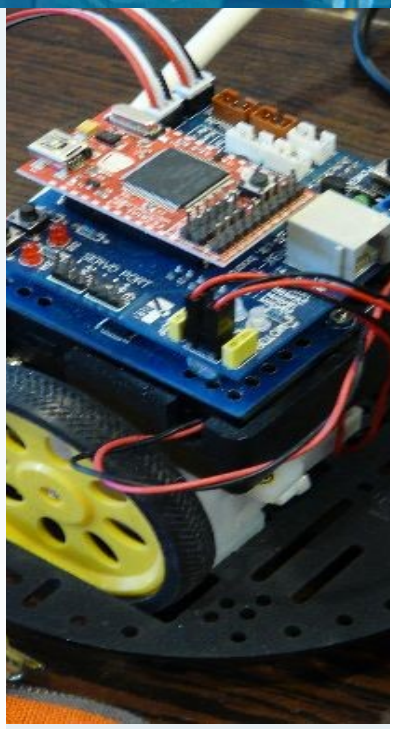
HCC ROBOTICA

September 2020
Jaargang 23
Nummer 3

ROBOBITS-89



VAN DE BESTUURSTAFEL

Beste clubgenoten,

Zaterdag 3 oktober, 11:00 uur is de 13e virtuele bijeenkomst van de HCC Robotica IG / RobotMC.

Aarzel niet om deel te nemen, vooraf aanmelden is niet nodig MAAR MAAK JE BEKEND IN DE ROOM = naam invoeren.

Mogelijke agendapunten

- Robotica bijeenkomst op zaterdag 3 oktober?
- Wvttk

Aandacht voor:

- microfoon uit wanneer niet aan het woord.
- scherm delen NA overleg met de overige deelnemers gelijktijdig delen van het scherm door meerdere deelnemers werkt verwarrend.
- het woord vragen door fysiek hand op te steken, dat werkt beter dan het hand icoontje.

Deelnemen = klik op de link

Het Robotica team

IN DIT NUMMER

Van de bestuurstafel 1
Lippo beveiliging 2
JTAG debug 4
HCC Agenda 8

(robot)kapper

Hoewel de beperkingen voor kapperszaken en kapsalons zijn versoepeld, is nog niet iedereen klaar om in het openbaar naar de kapper te gaan. Vandaar deze 'hair cut ' robot.

Lipo beveiliging

Mijn ervaring met het werken met Lipo accu's is nog niet zo groot. Ik heb inmiddels voldoende redenen om er meer aandacht aan te besteden en me er meer in te verdiepen.
Vandaar dit korte artikeltje.

Mijn eerste kennismaking met Lipo's was tijdens het bouwen van een drone. Na een eerste testvlucht bleek ik te enthousiast de Lipo te ver te hebben ontladen en de (dure) Lipo kon meteen naar het speciale afval.

Een andere reden om met Lipo's te gaan werken was de behoefte om een 'draadloze stroombron' te hebben voor mijn Raspberry Pi projecten. De Pi's zijn zeer populair. Helaas is het energiegebruik van deze kleine microcontroller/computer aanzienlijk wat vraagt om een stevige stroombron.

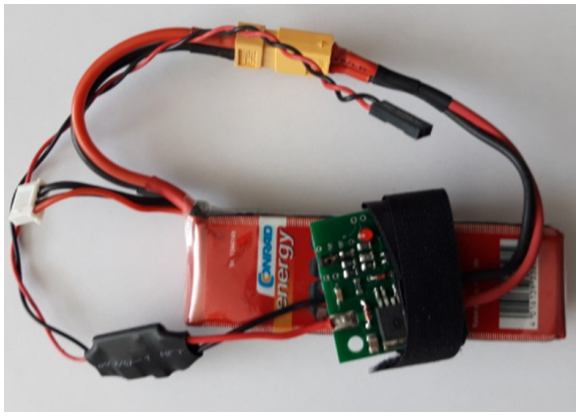
…. Met een drone hoog in de lucht is dat ook geen oplossing….

Om de kostbare Lipo's veilig te kunnen gebruiken is er dus behoefte aan een 'lipo discharge protection circuit' (dit is tevens een goede google zoekterm wanneer je wat meer informatie zoekt op internet).

Lipo zoemer
De eerste beveiliging die ik vond bleek een goedkoop zoemertje te zijn. Nadat een minimale celspanning is bereikt klinkt er een irritant snerpend piepgeluid. Met een drone hoog in de lucht is dat ook geen oplossing.

BatWat
Gelukkig heeft Coen de Robotica leden de mogelijkheid gegeven om zijn ontworpen 'Batwat' te gaan gebruiken. Er waren op het google forum veel discussies over het onderwerp en ontwikkeling van het product. Op de website van Coen is ook nog informatie te vinden.

Ik heb de Batwat uitgeprobeerd en kan alleen maar tevreden zijn met het resultaat. Hieronder zie je de Batwat aangesloten op een Lipo accu met drie cellen. Een spanningsregelaar is aangesloten om een Raspberry Pi van 5 volt voedingsspanning te voorzien. Wanneer de Lipo spanning een grenswaarde bereikt schakelt de BatWat de voedingsspanning naar de Pi uit. Vervolgens moet je dan niet vergeten de accu alsnog los te koppelen van de BatWat omdat de accu altijd nog een klein beetje belast blijft.



Lipo-accu (zie ook wiki)

Een lithium-ion-polymeer-accu, beter bekend als een lithium-polymeer-accu of kortweg Lipo-accu is een oplaadbare batterij. De batterij is een variant van de lithium-ion-accu. De Lipo-accu wordt met name in de modelbouw gebruikt vanwege het grote vermogen in verhouding tot het gewicht. Tevens hebben deze accu's een lage interne weerstand, waardoor ze een hoge stroom kunnen afgeven. De Lipo-cel heeft geen last van het geheugeneffect, zoals bij de NiCd-cel.

Het laden van een Lipo-accu gaat vrij snel: in zo'n 1 à 1,5 uur is hij opgeladen. Wel is het belangrijk dat het opladen met de juiste lader gebeurt. Lipo-accu's kunnen bij verkeerd laden, waarbij de temperatuur aardig kan oplopen, in brand vliegen of explo-deren. Er zijn speciale Lipo-celladers in de handel.

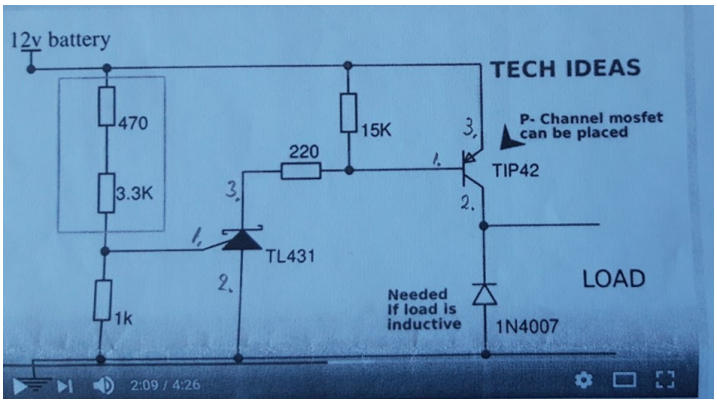
Een Lipo-accu kan uit meerdere cellen bestaan. De celspanning van een LiPo-cel is gemiddeld (nominaal) 3,7 V en in volgeladen toestand 4,2 V. Bij het ontladen mag de celspanning niet lager worden dan 3,0 V, anders gaat de cel stuk. Afhankelijk van de stroom die de aandrijving trekt, moet je de accu niet meer belasten als de celspanning is afgenomen tot een waarde tussen 3,6 V (lage stroom) en 3,2 V (hoge stroom).

Door meerdere cellen in serie te schakelen, krijg je accupakketten met een nominale spanning van 7,4 V, 11,1 V, 14,8 V, 18,5 V, 22,2 V etc. Dit wordt aangegeven met "2S", "3S", "4S", "5S", "6S" etc. ("S" staat voor serieschakeling).

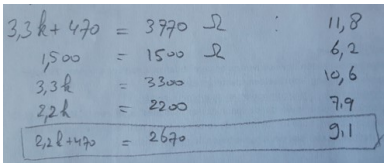
Door de interne weerstand wordt een LiPo-accu tijdens het ontladen warm. Dit leidt tot gasvorming waardoor de accu enigszins opbolt. Zolang de bolling na afkoelen weer verdwijnt, is er niets aan de hand. Blijvende bolling duidt echter op schade. Daarmee is de accu niet meteen onbruikbaar, maar het is dan wel zaak om de afzonderlijke celspanningen en de capaciteit in de gaten te houden.

Youtube alternatief

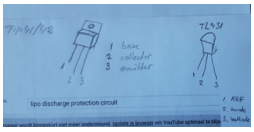
Op Youtube vond ik een alternatief schema voor een Lipo protection unit. Interessant was dit ontwerp omdat ik bijna alle onderdelen nog wel ergens had liggen. Alleen de TL431 heb ik nog moeten aanschaffen.
Het ontwerp is te gebruiken voor meerdere Lipo varianten (1,2,3,4 cellen). De spanning waarbij de belasting wordt uitgeschakeld is instelbaar met de weerstanden weergegeven binnen het kader in het schema:



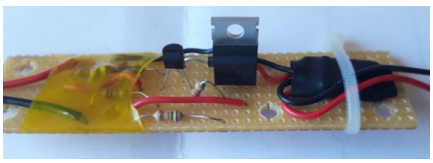
Vervolgens heb ik enkele metingen uitgevoerd. Dit leverde het volgende tabelletje op. Ik heb de waarden van de twee weerstanden gevarieerd en de spanning gemeten waarbij de belasting uitvalt. Voor de protectie van mijn Lipo met drie cellen heb ik de weerstanden gekozen van respectievelijk 2.2k + 470 .

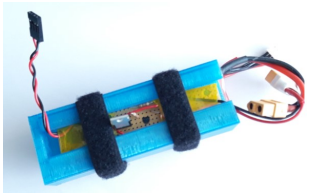


Het aansluiten van de TIP42 en de TL431 gaat als volgt:



Vervolgens heb ik alles in elkaar gezet, een 5 volt regulator aangesloten en de schakeling en de Lipo in een 3D geprinte bescherming/behuizing geplaatst:





Ook bij deze variant van de Lipo protectie geldt dat indien de beveiliging is ingeschakeld de Lipo nog wel moet worden losgekoppeld omdat er ook in deze schakeling nog een beperkte stroom blijft lopen die de Lipo ontlaadt.

Zeno Otten

BOEKEN

========

Getting Started with Java on Raspberry Pi:

Advanced control ROBOTICS

Als je van doe-het-zelf-elektronica, projecten, software en robots houdt, zal je dit boek stimulerend en onmiddellijk nuttig vinden. Met de juiste onderdelen en een beetje begeleiding kun je robotsystemen bouwen die beter aan de behoeften voldoen dan te dure commerciële systemen.

JTAG: een handige robot debug mogelijkheid.

Bij het maken van die fantastische robot die je in gedachten hebt kom je onvermijdelijk leuke uitdagingen tegen. Zo is daar welke mechanische onderdelen ga ik gebruiken, welke motoren en uiteindelijk ook, hoe gaat de software werken ?

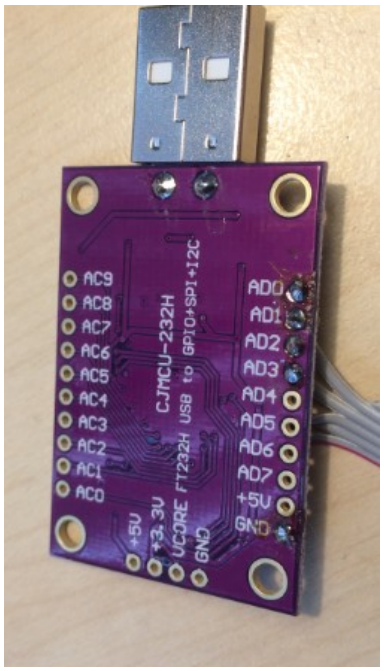
Voor een beetje robot wordt die software al snel ingewikkeld met bijvoorbeeld meerdere servo's die je wilt aansturen. Je wilt kunnen reageren op sensoren en dan een actie van de robot laten uitvoeren die dan weer in verschillende stappen verloopt. Vaak lopen ook meerdere taken tegelijk op je robot.
Alles bij elkaar wordt de software dan een uitgebreid geheel. Als dan iets niet helemaal loopt zoals verwacht dan is de vraag, waarom niet ?

JTAGInterface1.JPG

Je hebt die processor waar je programma op draait, een paar snoeren ernaar toe voor de sensoren en een paar snoeren eruit naar de motoren en de servo's.
Om inzicht te krijgen in wat de processor doet wordt dan vaak een seriële kabel toegevoegd die informatie regels geeft over wat de sensoren en dergelijke doen.
Aan je robot software worden dan regels toegevoegd die op diverse plekken informatie naar de seriële poort schrijven. Dit zorgt onvermijdelijk voor extra belasting van je processor en kan ook je software wat vertragen. Het kan zelfs zo zijn dat door timing verschillen je robot zich anders gaat gedragen met en zonder debug regels.

Zou het niet handig zijn als je onder de motorkap van je robot kon kijken op het moment dat die iets niet helemaal doet zoals je zou willen en dat zonder dat je processor extra belast wordt of dat je zelf programma code hoeft toe te voegen.

Nu daar is de industrie in het verleden ook al eens mee bezig geweest en die kwam met een standaard met de afkorting JTAG. Dat staat voor Joint Test Action Group en die vind je in veel hardware terug. Het is een standaard debug poort op je processor board die je kunt aankoppelen met een klein stukje goedkope hardware. Wel moet je processor board de JTAG signalen ondersteunen anders gaat het feest niet door.



JTAGInterface2.JPG

Een JTAG poort bestaat uit vier digitale signalen en een optioneel reset signaal:

TDI : Test Data In
TDO: Test Data Out
TMS : Test Mode Select
TCK: Test Clock
(TRST: Test Reset, optioneel)

Die 4 signalen koppel je aan een PC via bijvoorbeeld het hier links afgebeelde goedkope USB naar JTAG interface printje met de FTDI 232H chip.
Deze chip kent een JTAG communicatie modus. [JTAGInterface1.JPG en JTAGInterface2.JPG]

Een belangrijk punt is verder dat je deze 4 signalen van je processor niet meer voor andere doeleinden kunt gebruiken. Ze mogen dus geen verbinding met je sensoren of motoren hebben.

De volgende signalen zijn terug te vinden op het JTAG interface printje:

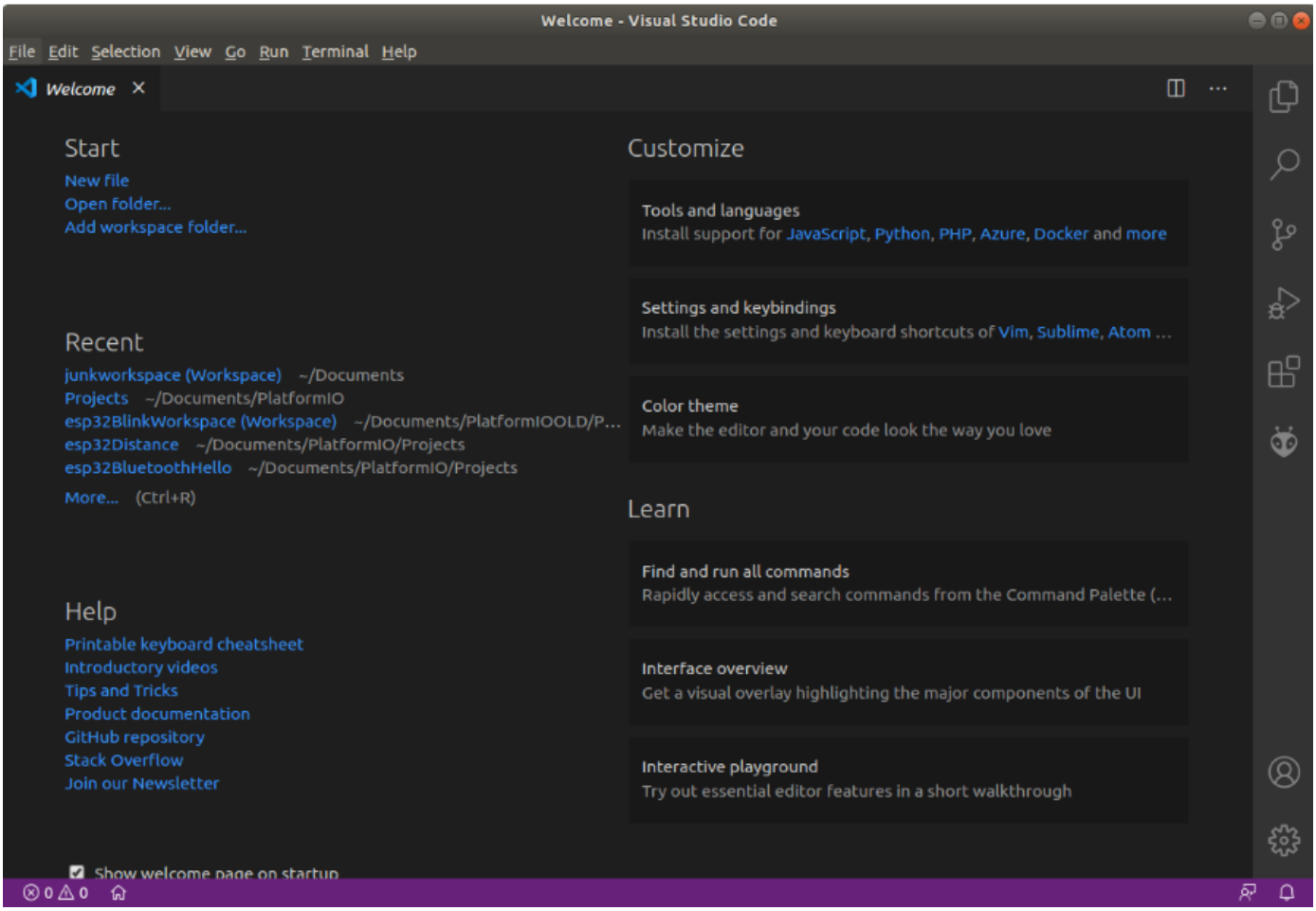
GND –> de min aansluiting
AD0 –> TCK
AD1 –> TDI
AD2 –> TDO
AD3 –> TMS

Naast deze hardware koppeling heb je ontwikkelomgeving software nodig op je ontwikkel systeem zoals een PC of een laptop. Deze software draait zowel onder Linux als onder Windows.

Hiervoor is de PlatformIO software beschikbaar op https://platformio.org/ . Dit is een uitgebreide ontwikkelomgeving die meer dan 800 processor boards ondersteunt. De benodigde compilers en dergelijke worden automatisch voor je gedownload en geconfigureerd.

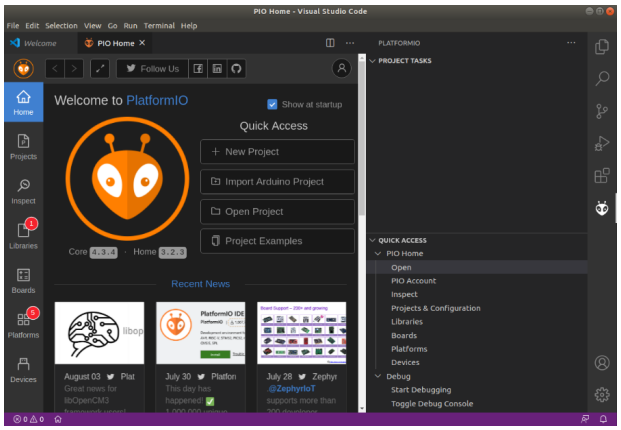
Voor het aan de praat krijgen van je eerste blink projectje doorloop je de volgende stappen:

Download en installeer Visual Studio Code (zie de urls aan het einde van dit artikel). Vervolgens voeg je de PlatformIO plugin aan Visual Studio Code toe. Dit gaat via de VSCode Extension Manager.
Als dit goed gelukt is dan ziet je Visual Studio Code ontwikkelomgeving er uit zoals in PlatformIO1.png. Check dat je het alien icoontje met de twee antennes terug kunt vinden rechts van het scherm. Dat is een teken dat ook het installeren van de PlatformIO plugin gelukt is.

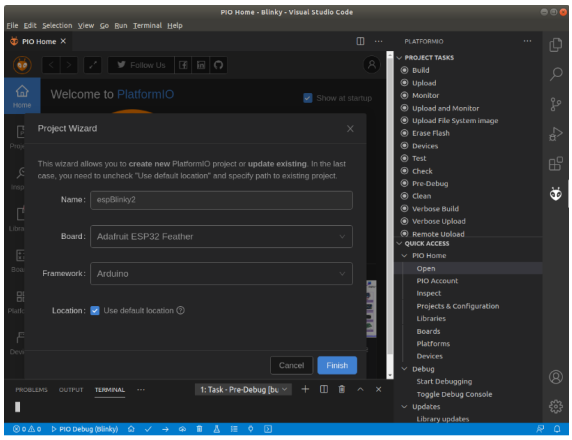


PlatformIO1.png

Om een nieuw project aan te maken klik je op het kleine alien icoon met de twee antennes rechts onderaan. Vervolgens kies je bij 'Quick Access'->'PIO Home'->'Open'. Dit ziet er dan zo uit : [PlatformIO2.png]



[PlatformIO2.png]



[PlatformIO3.png]

Kies hier dan 'New Project'. Dit ziet er dan uit zoals [PlatformIO3.png].

Hier geef je de naam van je project op (bijvoorbeeld espBlinky2) en je kiest jouw processor board type (bijvoorbeeld 'Adafruit ESP32 Feather'). Als het je gaat om een specifieke microcontroller dan is het voldoende om een board te kiezen dat deze processor heeft, het board hoeft niet helemaal overeen te komen met het board dat je hebt.
Vervolgens klik je op 'Finish'. Als het de eerste keer is dat je met dit processorboard gaat werken dan zal PlatformIO eerst de compilers en dergelijke voor je downloaden. Dit kan een paar minuten duren. Daarna zie je rechts de files van je nieuwe project staan.

Een belangrijk punt is dat je aan moet geven aan PlatformIO dat je de debug informatie wilt mee compileren in je broncode naar de robot processor. Als je dit niet doet dan krijg je tijdens het debuggen alleen de assembler te zien van je robotcode en dat debugged een stuk minder makkelijk. Wel neemt de gebruikte ruimte op je robot processor toe als je debug informatie toe laat voegen.

Open de file platformio.ini die te vinden is in de project door erop te dubbel-klikken.
Voeg de volgende twee regels toe:

```
build_flags = -g -O0
debug_init_break = tbreak setup
```

Sla deze nieuwe versie van de platformio.ini file op. (Ctrl-s)
Dit vertelt de compiler om debug informatie mee te compileren en zet ook de code optimalisatie uit. Dit laatste is om te voorkomen dat je in de debugger soms wat vreemde sprongen ziet of variabelen niet kunt bekijken omdat die door de compiler uit geoptimaliseerd zijn.
Eventueel kun je die optie '-O0' achterwege laten, de '-g' optie is het belangrijkste.
Verder zorgt de debug_init_break regel ervoor dat je bij het opstarten van je robot programma direct op de eerste regel van je setup() procedure van je Arduino code terecht komt.

De broncode is te vinden onder src, file main.cpp. Hierin zijn de twee bekende procedures setup() en loop() van het Arduino framework terug te vinden. Voeg hier wat code toe om de LED van je specifieke processor board te laten knipperen, net zoals je gewent was bij de Arduino IDE.

Belangrijk is om als eerste regel in je main.cpp bestand de volgende include regel te hebben:

```
#include <Arduino.h>
```

Dit zorgt ervoor dat de functies van Arduino binnen je programma beschikbaar zijn.

Om je programma via de JTAG kabel naar je robot processor te krijgen klik je op het debug icoon (driehoekje met kevertje) rechts van het scherm.

Vervolgens klik je op het groene driehoekje naast de tekst RUN. Als het goed is zie je onderaan je scherm nu een aantal compileer boodschappen voorbij komen en de tekst 'SUCCESS'. Kijk onder de tab 'DEBUG CONSOLE' of daar nog foutmeldingen staan als na een paar minuten niets lijkt te gebeuren. Controleer of je USB kabel ook op je robot processor aangesloten zit en die dus stroom krijgt. Daarnaast moet de FT232H interface ook op je robot processor aangesloten zitten via de eerder beschreven JTAG signalen.

Je programma code stopt automatisch op de eerste regel van de setup() functie.
Dit ziet er als volgt uit [PlatformIO4.png].

Urls:

CJMCU FT232H :
https://nl.aliexpress.com/item/32814913865.html

Visual Studio Code :
https://code.visualstudio.com/

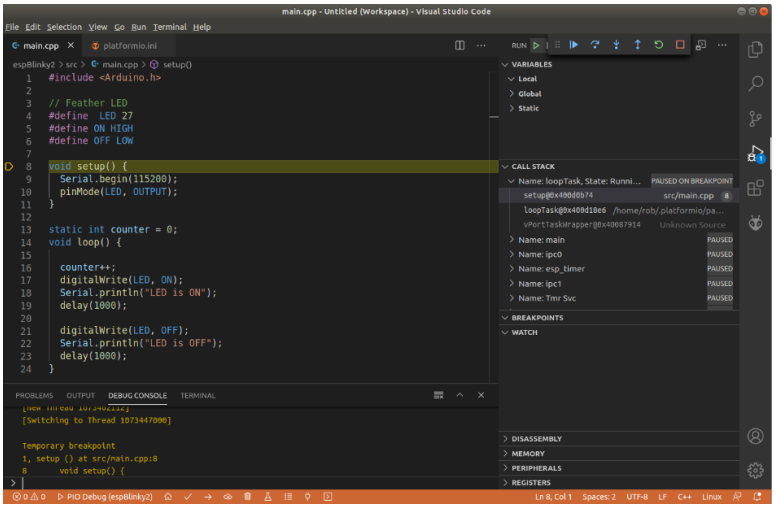
Visual Studio Code extension manager uitleg:
https://platformio.org/install/ide?install=vscode

PlatformIO software :
https://platformio.org/

Start de verdere uitvoering van je robot programma door in de startbalk rechtsboven het blauwe play driehoekje aan te klikken.

Je kunt je robot programma even pauzeren door op de pauzeer button te klikken, rechtsboven in het scherm. Als je programma meerdere taken bevat die tegelijk lopen dan kun je bij 'CALL STACK' de taak kiezen die jouw code bevat en daarop dubbel

Als je wilt dat je programma vanzelf pauzeert op een bepaalde regel dan kun je een breakpoint plaatsen. Open hiervoor het betreffende programma bestand en klik links in de kantlijn van dit bestand. Als het goed is verschijnt een kleine rode stip voor die regel.
Bij het uitvoeren van je programma zal nu automatisch op die regel gepauzeerd worden. Je kunt dan kijken wat de variabelen zijn etc. en daarna weer doorstarten door F5 of het kleine blauwe play icoontje te kiezen.



[PlatformIO4.png]

Het is mogelijk om regel voor regel door je programma code te lopen maar ook om je programma tot op een bepaald punt te laten lopen. Bekijk hiervoor de menu opties die je onder menu 'Run' terug vindt zoals 'Step Over', 'Step Into' en 'Continue'. (je programma moet eerst gepauzeerd zijn voordat je de step functies kunt gebruiken)

Als je je robot programma gepauzeerd hebt dan is het mogelijk om de waarden van je variabelen te bekijken door met je muis boven de betreffende variabele naam te gaan hangen. Om je files terug te vinden tijdens debuggen klik je op het blader icoontje rechtsboven. Je kunt dan onder de src directory bijvoorbeeld je main.cpp file terug vinden.

Als je via de Arduino Serial class iets naar buiten hebt geschreven vanuit je robot dan kun je dit binnen PlatformIO zien door bij het Terminal het menu 'New Terminal' te kiezen. Je krijgt dan een shell prompt waarop je je eigen communicatie programma kunt starten onder de tab 'TERMINAL' onderaan je scherm. Onder Linux is dat bijvoorbeeld het 'cu' programma.

Dit artikel is een vogelvlucht door het inrichten en gebruiken van JTAG voor het debuggen van je robot processor code met de PlatformIO tool.
Ik hoop dat je hier veel plezier van gaat hebben en je debuggen op een hoger plan kan krijgen dan het wat omslachtiger gebruik van print statements in je code.
Met de beschreven aanpak kun je ook lastige situaties de baas waarin je robot niets meer lijkt te doen en je je afvraagt wat die aan het doen is. Als je dan met een actieve JTAG verbinding op de pauzeer button klikt dan zie in je debugger de actieve code en de Call Stack laat je zien welke procedures geleid hebben tot dit punt in je programma code.

Veel plezier met het debuggen van je robot !

Rob van der Ouderaa (rouderaa@hccnet.nl)

HCC!Robotica ig

HCC-Robotica is een interessegroep die zich bezig houdt met het ontwikkelen, ontwerpen, programmeren en bouwen van elektronica en mechatronica, toegepast op robots. Deze meer of minder intelligente en autonome robots en machines met verschillende sensoren, actuatoren, processoren en bewegende onderdelen worden onder andere ingezet bij de jaarlijkse georganiseerde Roborama wedstrijden. Wij komen elke eerste zaterdag van de maand bijeen in dorpshuis de Dissel te Hooglanderveen. Kennis delen, kennis vergaren, presentaties en workshops bijwonen zijn terugkerende activiteiten tijdens deze bijeenkomsten.

U bent van harte welkom!

Agenda

zaterdag 3 oktober, 11:00 uur

13e virtuele bijeenkomst van de
HCC Robotica IG / RobotMC

Discussiegroepen
HCCROBOTICA:

http://groups.google.nl/group/hcc_robotmc

Blogs
http://zotten.wordpress.com/
Website van Coen
https://avretro.wordpress.com/
http://www.robotblog.nl/

Facebook

HCC!Robotica ook op Facebook.

Gewoon om te laten weten, dat wij ook op Facebook actief zijn.

Voor de aankomende kerst?

De WLED03 is een met Wifi-verbonden PWM-driver voor 12-V-LED-verlichting voorzien van de populaire ESP32. Hij biedt drie kanalen die elk tot 3 ampère kunnen worden belast.

HCC!Robotica ig

Dagelijks bestuur:
Voorzitter : Wim de Boer
Secretaris : Edith van Putten
Penningmeester : Ed Buzzi

Het Kernledenbestand ziet er als volgt uit en zal het dagelijks bestuur ondersteunen:
Redactie : Zeno Otten
Website : Bert Berrevoets
Techniek : Tim Woldring
Roborama : Bert Ruben
Public Relations : Rien van Harmelen
Externe Contacten : Ed Buzzi

Website: http://www.hccrobotica.nl