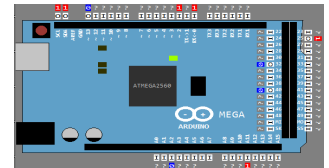


UnoArduSimV2.9.2 Volledige Hulp



Inhoudsopgave

[Overzicht](#)

[CodePaneel, Voorkeuren en Editeren/Bekijken](#)

[CodePaneel](#)

[Voorkeuren](#)

[Editeren/Bekijken](#)

[VariabelenPaneel en Editeren/Opvolgen Variabele window](#)

[LabtafelPaneel](#)

[De 'Uno' of 'Mega'](#)

['I/O' Apparaten](#)

['Serial' Monitor \('SERIAL'\)](#)

[Afwisselend Serieel \('ALTSER'\)](#)

[SD Schijf \('SD_DRV'\)](#)

[TFT-Scherm \('TFT'\)](#)

[Configureerbare SPI Slaaf \('SPISLV'\)](#)

[Tweedraads I2C Slaaf \('I2CSLV'\)](#)

[Tekst LCD I2C \('LCDI2C'\)](#)

[Tekst LCD SPI \('LCDSPI'\)](#)

[Tekst LCD D4 \('LCD_D4'\)](#)

[Multiplexer LED I2C \('MUXI2C'\)](#)

[Multiplexer LED SPI \('MUXSPI'\)](#)

[Uitbreidingspoort SPI \('EXPSPI'\)](#)

[Uitbreidingspoort I2C \('EXPI2C'\)](#)

['1-Wire' Slaaf \('OWIISLV'\)](#)

[Digitale Enkele Puls \('1SHOT'\)](#)

[Schuifregister Slaaf \('SRSLV'\)](#)

[Programmeerbare 'I/O' Apparaat \('PROGIO'\)](#)

[Pulsgenerator \('PULSER'\)](#)

[Analoge Functie Generator \('FUNCGEN'\)](#)

[Stappenmotor \('STEPR'\)](#)

[Gepulseerde Stappenmotor \('PSTEPR'\)](#)

[Gelijkstroommotor \('MOTOR'\)](#)

[Servomotor \('SERVO'\)](#)

[Piëzo Luidspreker \('PIEZO'\)](#)

[Schuifweerstand \('R=1K'\)](#)

[Drukknop \('PUSH'\)](#)

[Gekleurde LED \('LED'\)](#)

[Rij 4-LED \('LED4'\)](#)

[7-segment LED Cijfer \('7SEG'\)](#)

[Analoge Schuifweerstand](#)

[Pins Verbindingsdraad \('JUMP'\)](#)

[Menu's](#)

[Bestand:](#)

[Laden INO of PDE Prog \(ctrl-L\)](#)

[Editeren/Bekijken \(Ctrl-E\)](#)

[Opslaan](#)

[Opslaan Als](#)

[Volgende \('#include'\)](#)

[Voorgaand](#)

[Exit](#)

[Vinden:](#)

[Klimmen Stapel Oproepen](#)

[Daal Stapel Oproepen](#)

[Stel Zoeken-tekst in \(Ctrl-F\)](#)

[Vinden Volgende tekst](#)

[Vinden Vorige tekst](#)

[**Uitvoeren:**](#)

[Stap-In \(F4\)](#)

[Stap-Over \(F5\)](#)

[Stap-Uit \(F6\)](#)

[Uitvoeren-Tot \(F7\)](#)

[Uitvoeren-Totdat \(F8\)](#)

[Uitvoeren \(F9\)](#)

[Halt \(F10\)](#)

[Reset](#)

[Animeren](#)

[Trage Weergave](#)

[**Opties:**](#)

[Stap-Over-constructors/ Operators](#)

[Register-Allocation](#)

[Fout bij niet-geïnitieerd](#)

[Toegevoegd 'loop\(\)' Vertraging](#)

[Nested Interrupts toestaan](#)

[**Configureren:**](#)

['I/O' Apparaten](#)

[Voorkeuren](#)

[**VarRefresh:**](#)

[Laat Auto \(-\) Samentrekken toe](#)

[Minimaal](#)

[Markeer Veranderingen](#)

[**Windows:**](#)

['Serial' Monitor](#)

[Herstel alles](#)

[Pins Golfvormen Digitale](#)

[Pin Golfvorm Analoge](#)

[**Hulp:**](#)

["Wat is dit" -modus \(Ctrl >\)](#)

[Quick Hulp Bestand \(Ctrl ?\)](#)

[Volledige Hulp Bestand](#)

[Bug-oplossingen](#)

[Wijzigen / Verbeteringen](#)

[Wat betreft](#)

['Uno of 'Mega' Printplaat en 'I/O' Apparaten](#)

[Timing](#)

['I/O' Apparaat Timing](#)

[Sounds](#)

[**Beperkingen en niet-ondersteunde Elementen**](#)

[Inclusief Bestanden](#)

[Dynamische geheugentoewijzingen en RAM](#)

['Flash' Geheugentoewijzingen](#)

['String' Variabelen](#)

[Arduino-bibliotheken](#)

[Pointers](#)

['class' en 'struct' Objecten](#)

[Bereik](#)

[qualifiers 'unsigned', 'const', 'volatile', 'static'](#)

[Compiler-richtlijnen](#)

[Arduino-taalelementen](#)

[C / C ++ - Taalelementen](#)

[Functie-sjablonen](#)

[Realtime Emulatie](#)

[**Releaseopmerkingen**](#)

[Bug-Oplossingen](#)

[V2.9.2 - april 2021](#)

[V2.9.1 - februari 2021](#)

[V2.9 - januari 2021](#)

[V2.8.2-September 2020](#)

[V2.8.1-Juni 2020](#)

[V2.8.0-Juni 2020](#)

[V2.7.0- maart 2020](#)
[V2.6.0- jan 2020](#)
[V2.5.0- oktober 2019](#)
[V2.4 - mei 2019](#)
[V2.3 - december 2018](#)
[V2.2 - Jun. 2018](#)
[V2.1.1- mrt. 2018](#)
[V2.1- mrt. 2018](#)
[V2.0.2 feb. 2018](#)
[V2.0.1- Jan. 2018](#)
[V2.0- december 2017](#)

[Veranderingen / Verbeteringen](#)

[V2.9.2 - aopril 2021](#)
[V2.9 - januari 2021](#)
[V2.8.2- September 2020](#)
[V2.8.0- Juni 2020](#)
[V2.7.0- maart 2020](#)
[V2.6.0 januari 2020](#)
[V2.5.0 oktober 2019](#)
[V2.4 mei 2019](#)
[V2.3 december 2018](#)
[V2.2 jun. 2018](#)
[V2.1 mrt. 2018](#)
[V2.0.1 januari 2018](#)
[V2.0 september 2017](#)

Overzicht

UnoArduSim is een freeware **echte tijd** (zien voor Timing **beperkingen**) simulatortool die ik heb ontwikkeld voor de student en Arduino-liefhebber. Het is ontworpen om u te laten experimenteren met, en eenvoudig te debuggen, Arduino programmas **zonder de noodzaak voor enige echte hardware**. Het is gericht op de **Arduino 'Uno' of 'Mega'** printplaat, en kunt u kiezen uit een set virtuele 'I/O' apparaten, en deze apparaten configureren en verbinden met uw virtuele 'Uno' of 'Mega' in de **LabtafelPaneel**. - u hoeft zich geen zorgen te maken over bedradingsfouten, kapotte / losse verbindingen of een defecte apparaten die de ontwikkeling en het testen van uw programma in de war brengt.

UnoArduSim biedt eenvoudige foutmeldingen voor alle analyseren- of uitvoering-fouten die het tegenkomt en maakt foutopsporing met **Reset**, **Uitvoeren**, **Uitvoeren-Tot**, **Uitvoeren-Totdat**, **Halt** en flexibel **Stap** operaties in de **CodePaneel**, met een gelijktijdige weergave van alle globale en momenteel actieve lokale variabelen, reeksen en objecten in de **VariabelenPaneel**. Uitvoeren-tijd reeks-grenscontroles worden verstrekt en ATmega RAM-overloop wordt gedetecteerd (en de boosaardige programma-regel gemarkeerd!). Alle elektrische conflicten met-bevestigde 'I/O' apparaten worden gemarkeerd en gerapporteerd wanneer ze zich voordoen.

Wanneer een INO of PDE programma bestand wordt geopend, wordt deze in de programma geladen **CodePaneel**. De programma krijgt dan een Analyseren, om het in een tokenized uitvoerbaar bestand te veranderen dat dan klaar is voor **gesimuleerde uitvoering** (In tegenstelling tot Arduino.exe, is een zelfstandig binaire-uitvoerbaar bestand *niet* gemaakt). Elke analyseren-fout wordt gedetecteerd en gemarkeerd door de regel te markeren die niet naar analyseren is gestuurd en de fout op de analyseren te rapporteren. **Statusbalk** helemaal onderaan de UnoArduSim-applicatie window. Een **Editeren/Bekijken** window kan worden geopend zodat u een op syntaxis gemarkeerde versie van uw programma-gebruiker kunt zien en bewerken. Fouten tijdens gesimuleerde uitvoering (zoals een verkeerd aangepaste overdrachtsnelheid) worden gerapporteerd op de statusbalk en via een pop-upberichtvak.

UnoArduSim V2.8 is een vrijwel volledige implementatie van de **Arduino programmeertaal V1.8.8 zoals gedocumenteerd op de arduino.cc**. Taalreferentie-webpagina, en met toevoegingen zoals vermeld in de versie Downloaden-pagina Uitgaveopmerkingen. Hoewel UnoArduSim de volledige C ++-implementatie die de Arduino.exe onder GNU compiler ondersteunt niet ondersteunt, is het waarschijnlijk dat alleen de meest geavanceerde programmeurs zullen ontdekken dat sommige C / C ++-elementen die ze willen gebruiken ontbreken (en natuurlijk zijn er altijd eenvoudige codeerwerkronde voor dergelijke ontbrekende functies). Over het algemeen heb ik alleen de meest bruikbare C / C ++-functies voor Arduino-hobbyisten en -studenten ondersteund, bijvoorbeeld **'enum'** en **'#define'** worden ondersteund, maar functie-pointers niet. Hoewel door de gebruiker gedefinieerd objecten (**'class'** en **'struct'**) en (de meeste) overbelasting van de bediener wordt ondersteund, *meerdere overerving is dat niet*.

Omdat UnoArduSim een simulator met een hoog niveau is, **alleen C / C ++-instructies worden ondersteund**, *assemblage taal verklaringen zijn dat niet*. Evenzo, omdat het geen computersimulatie op laag niveau is, **ATmega328-registers zijn niet toegankelijk voor uw programma** voor lezen of schrijven, hoewel registertoewijzing, doorgeven en retourneren worden nagebootst (u kiest dat onder het menu **Opties**).

Vanaf V2.6, UnoArduSim heeft ingebouwd automatische steun voor een beperkte subset van de Arduino voorzien bibliotheken, deze zijn: **'Stepper.h'**, **'Servo.h'**, **'SoftwareSerial.h'**, **'SPI.h'**, **'Wire.h'**, **'OneWire.h'**, **'SD.h'**, **'TFT.h'** en **'EEPROM.h'** (versie 2). V2.6 introduceert een mechanisme voor 3rd partij bibliotheek ondersteuning via bestanden in de **'include_3rdParty'** map die kan worden gevonden in het UnoArduSim install directory. Voor enige **'#include'** van door de gebruiker gemaakte bibliotheken, zal UnoArduSim dat doen **niet** doorzoek de gebruikelijke Arduino-installatiemapstructuur om de bibliotheek te lokaliseren; in plaats daarvan jij **nodig hebben** om de corresponderende header (".h") en source (".cpp") bestand te kopiëren naar dezelfde map als de programma bestand waar je aan werkt (uiteraard met de beperking dat de inhoud van een **'#include'** bestand moet volledig begrijpelijk zijn voor de UnoArduSim analyzer).

Ik heb UnoArduSimV2.0 ontwikkeld in QtCreator met ondersteuning voor meerdere talen, en het is momenteel alleen beschikbaar voor Windows TM . Porten naar Linux of MacOS, is een project voor de toekomst! UnoArduSim is ontstaan uit simulatoren die ik in de loop der jaren heb ontwikkeld voor cursussen die ik heb gegeven aan de Queen's University, en die redelijk uitgebreid is getest, maar er zijn vast wel een paar bugs die zich daar nog steeds schuilhouden. Als u een bug wilt melden, beschrijf dit dan (kort) in een e-mail aan unoArduSim@gmail.com en **zorg ervoor dat je je volledige bug-inducerende programma Arduino-broncode bijvoegt** dus ik kan de bug repliceren en repareren. Ik zal niet reageren op individuele bug-rapporten en ik heb geen gegarandeerde tijdlijnen voor fixes in een volgende release (denk eraan dat er bijna altijd oplossingen zijn!).

cheers,

Stan Simmons, PhD, P. Eng.
Universitair hoofddocent (gepensioneerd)
Afdeling Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada







CodePaneel, Voorkeuren en Editeren/Bekijken

(Terzijde: de hieronder getoonde voorbeelden van windows vallen allemaal onder een door de gebruiker gekozen Windows-OS kleurenthema met een donkerblauwe window-achtergrondkleur).

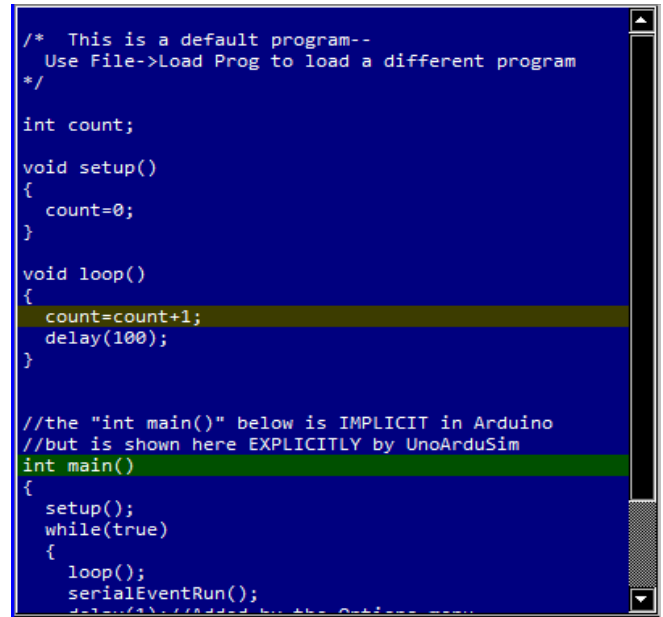
CodePaneel

De **CodePaneel** geeft uw programma-gebruiker weer en markeert nummers van zijn uitvoering. Nadat een geladen programma een succesvolle Analyseren heeft, komt de eerste regel binnen '**main()**' is gemarkeerd en de programma is klaar voor uitvoering. Let daar op '**main()**' wordt impliciet toegevoegd door Arduino (en door UnoArduSim) en dat is het geval **niet** neem het op als onderdeel van uw gebruiker programma bestand. Uitvoering is onder controle van het menu **Uitvoeren** en de bijbehorende **Tool-Bar** toetsen en functie-toets snelkoppelingen.

Na het stappen uitvoering door één (of meer) instructies (die u kunt gebruiken **Tool-Bar** toetsen , ,  of ), De lijn die programma uitvoerde volgende wordt vervolgens groen gemarkeerd wordt - de groene-gemarkeerde lijn is altijd de volgende regel **klaar om te uitvoerde zijn** .

Op dezelfde manier, wanneer een lopende programma een (tijdelijk) raakt **Uitvoeren-Tot** breekpunt, uitvoering is gestopt en de breekpunt-regel is gemarkeerd (en is dan klaar voor uitvoering).

Als programma uitvoering momenteel wordt gestopt, en klikt u in de **CodePaneel** window, de lijn die je net hebt geklikt, wordt gemarkeerd in een donkere olijf (zoals afgebeeld op de foto) - de next-to-be-uitvoerde lijn blijft altijd in het groen (vanaf V2.7). . Maar u kunt uitvoering veroorzaken *om vooruitgang te boeken* de regel die u zojuist hebt gemarkeerd door op de te klikken **Uitvoeren-Tot**  **Tool-Bar** knop. Met deze functie kunt u snel en eenvoudig bepaalde lijnen in een programma bereiken, zodat u vervolgens regel voor regel over een programma-interessant gedeelte kunt stappen.





```
/* This is a default program--
   Use File->Load Prog to load a different program
*/






int count;

void setup()
{
    count=0;
}

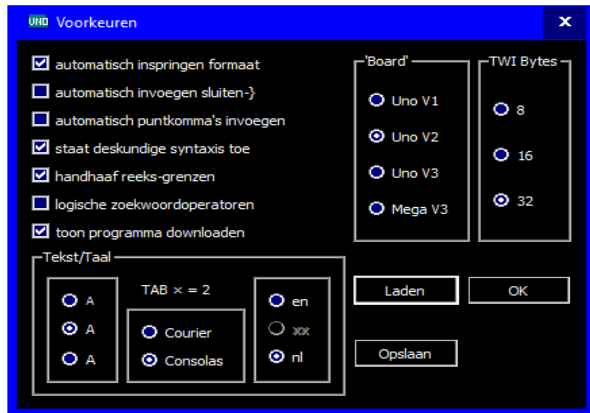
void loop()
{
    count=count+1;
    delay(100);
}

//the "int main()" below is IMPLICIT in Arduino
//but is shown here EXPLICITLY by UnoArduSim
int main()
{
    setup();
    while(true)
    {
        loop();
        serialEventRun();
        delay(1); //added by the Outpass menu
```

Als uw geladen programma een heeft '**#include**' bestanden, u kunt ertussen schakelen met behulp van **Bestand | voorgaand** en **Bestand | volgende** (met **Tool-Bar** toetsen  en ).

De acties van de **Vinden** menu kunt u **EEN VAN BEIDE** zoeken naar tekst in de **CodePaneel** of **VariabelenPaneel** (**Tool-Bar** toetsen  en ). Of sneltoetsen **pijlje omhoog** en **pijlje)** *na het eerste gebruik* **Vinden | Set Zoeken tekst** of **Tool-Bar** , **OF ANDERS** naar *navigeren door de call-stack* in de **CodePaneel** (**Tool-Bar** toetsen  en ). Of sneltoetsen **pijlje omhoog** en **pijlje)**. Keys **PgDn** en **PgUp** springen selectie naar het volgende / vorige functie .

Voorkeuren



Configureren | Voorkeuren staat gebruikers toe om programma en kijkvoorkeuren in te stellen (die een gebruiker normaal gesproken wil adopteren op hij de volgende sessie). Deze kunnen daarom worden opgeslagen en geladen vanuit een '**myArduPrefs.txt**' bestand dat zich in dezelfde directory bevindt als de geladen 'Uno' of 'Mega' programma ('**myArduPrefs.txt**' wordt automatisch geladen als deze bestaat).

In dit dialoogvenster kunt u kiezen tussen twee lettertypen met één spatie en drie lettertypen en andere overige voorkeuren. Vanaf V2.0 is de taalkeuze nu inbegrepen. - dit omvat altijd Engels (**nl**), plus een of twee andere locale talen van de gebruiker (waar deze beschikbaar zijn), en één

opheffing op basis van de tweeletterige ISO-639-taalcode op de allereerste regel van de '**myArduPrefs.txt**' bestand (als daar een wordt verstrekt). Keuzes verschijnen alleen als een ".qm" -vertaling bestand bestaat in de map met vertalingen (binnen de UnoArduSim.exe-thuismap).

Editeren/Bekijken

Door te dubbelklikken op een regel in de **CodePaneel** (of gebruik het menu **Bestand**), an **Editeren/Bekijken** window wordt geopend om wijzigingen in uw programma bestand mogelijk te maken, met de **momenteel geselecteerde lijn** in de **CodePaneel** is gemarkeerd.

Deze window heeft volledige bewerkingsmogelijkheden met dynamische syntax-highlighting (verschillende markeer-kleuren worden gebruikt voor C ++-trefwoorden, opmerkingen, enz.). Er zijn optionele vetgedrukte-syntaxisaccentuering en automatische inspringingniveau-opmaak (ervan uitgaande dat u dat hebt geselecteerd

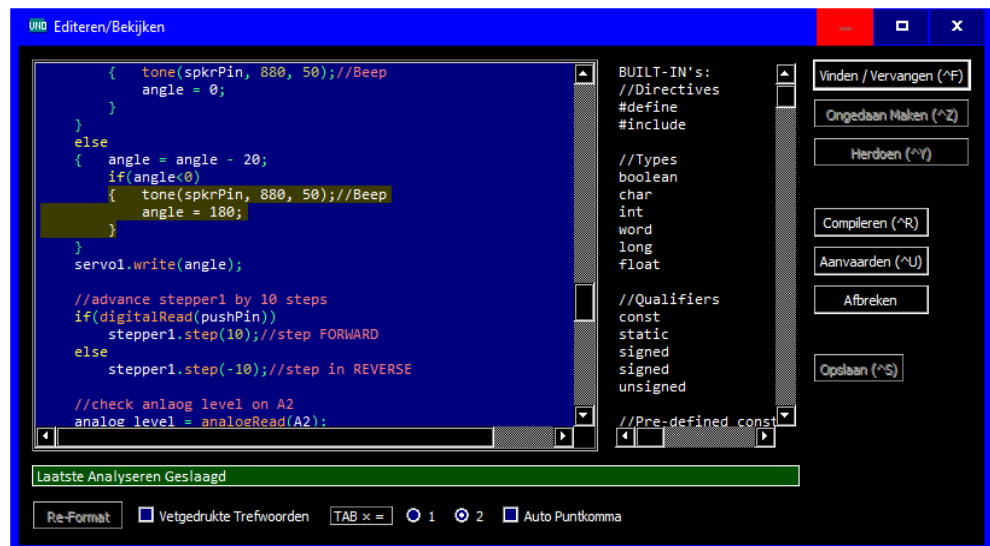
Configureren | Voorkeuren). U kunt ook gemakkelijk ingebouwd

functie-oproepen selecteren (of ingebouwd '**#define**' constanten) die moet worden toegevoegd aan uw programma vanuit de opgegeven keuzelijst - dubbelklik eenvoudig op het gewenste item in de lijstbox om het toe te voegen aan uw programma op de huidige caret positie (functie-call variabele **types** zijn alleen voor informatie en zijn uitgekleeft om dummy plaatshouders te laten wanneer deze aan uw programma worden toegevoegd).

De window heeft **Vinden** (gebruik **ctrl-F**) en **Vinden / Vervangen** mogelijkheid (gebruik **ctrl-H**). De **Editeren/Bekijken** window heeft **Ongedaan Maken** (**ctrl-Z**), en **Herdoen** (**ctrl-Y**) knoppen (die automatisch verschijnen).

Gebruik ALT-pijl naar rechts op het verzoek van auto-completion keuzes voor ingebouwd **global variabelen**, en voor **lid variabelen en functies**.

Wegwerpen **alle veranderingen** Nadat je de programma voor het eerst hebt geopend om te bewerken, klik je op de



Afbreken knop. Om het te accepteren huidige staat, klik op de **Aanvaarden** knop en de programma ontvangt automatisch een andere Analyseren (en wordt gedownload naar de 'Uno' of 'Mega' als er geen fouten worden gevonden) en de nieuwe status verschijnt in de hoofdunit van de UnoArduSim window **Statusbalk** .

EEN **Compileren** (**CTRL-R**) knop (plus een bijbehorende **Analyseren-status** bericht-box zoals te zien in de afbeelding hierboven) is toegevoegd om het testen van bewerkingen mogelijk te maken zonder eerst de window te hoeven sluiten. EEN **Opslaan** (**ctrl-S**) knop is ook toegevoegd als een snelkoppeling (gelijk aan een **Aanvaarden** plus een later gescheiden **Opslaan** van de hoofd window).

Op beide **Afbreken** of **Aanvaarden** zonder gemaakte bewerkingen, de **CodePaneel** huidige regel verandert in de **laatste Editeren/Bekijken caret positie** en u kunt die functie gebruiken om de **CodePaneel** naar een specifieke regel (mogelijk om je voor te bereiden om een te doen **Uitvoeren-Tot**), Je kan ook gebruiken **ctrl-PgDn** en **ctrl-PgUp** om naar de volgende (of vorige) lege regelonderbreking in uw programma te springen - dit is handig om snel omhoog of omlaag te navigeren naar belangrijke locaties (zoals lege lijnen tussen functies). Je kan ook gebruiken **ctrl-huis** en **ctrl-End** om naar respectievelijk de programma-start en -einde te springen.




'**Tab**'-level automatisch inspringen opmaak wordt gedaan wanneer de window opent, als die optie werd vastgelegd in een **Configureren | Voorkeuren**. U kunt opnieuw dat het formatteren op elk gewenst moment door te klikken op de **Re-Format** knop (deze wordt alleen ingeschakeld als u eerder hebt geselecteerd **automatische inspringen Voorkeur**). U kunt uzelf ook toevoegen of verwijderen tabs om een groep van vooraf geselecteerde opeenvolgende lijnen met het toetsenbord **rechter pijl** of **linker pijl** sleutels - maar **automatische inspringen voorkeur moet uitgeschakeld** om te voorkomen dat uw eigen aangepaste tabblad levels te verliezen.

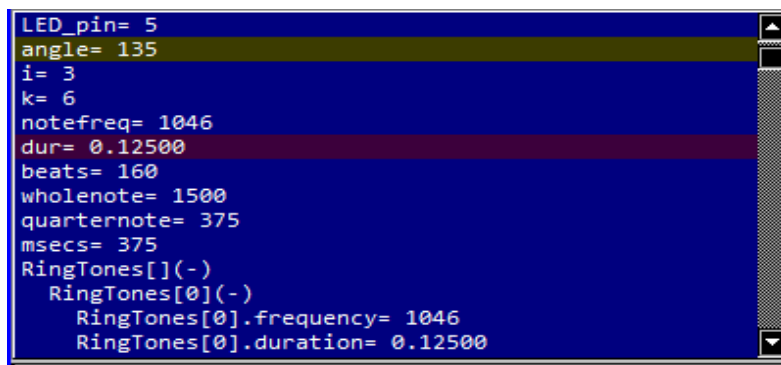
Wanneer **Auto puntkomma's** is aangevinkt, druk op **invoeren** om een regel te beëindigen, wordt automatisch de regelkomma's met regelafsluiting ingevoegd.

En om u te helpen uw contexten en accoladen beter bij te houden, klikt u op a ' { ' of ' } ' accolade **markeert alle tekst tussen die accolade en de bijbehorende partner** .

VariabelenPaneel en Editeren/Opvolgen Variabele window

De **VariabelenPaneel** bevindt zich net onder de **CodePaneel**. Het toont de huidige waarden voor elke gebruiker globale en actieve (in-bereik) lokale variabele / reeks / object in de geladen programma. Aangezien uw programma uitvoering zich tussen functies beweegt, **de inhoud verandert alleen om die lokale variabelen weer te geven die toegankelijk is voor de huidige functie / bereik, plus alle door de gebruiker gedeclareerde globals**. Elke variabelen aangegeven als 'const' of zo 'PROGMEM' (toegewezen aan 'Flash' geheugen) hebben waarden die niet kunnen veranderen, en om ruimte te besparen zijn deze daarom *niet getoond*. 'Servo' en 'SoftwareSerial' object-instanties bevatten geen bruikbare waarden, dus worden ook niet weergegeven.

Jij kan **vind** gespecificeerd **tekst** met zijn text-search commando's (met **Tool-Bar** toetsen  en  Of sneltoetsen **pijlje omhoog** en **pijlje**) Na het eerste gebruik **Vinden | set Zoeken** tekst of  .



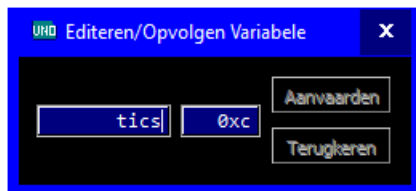
```
LED_pin= 5
angle= 135
i= 3
k= 6
notefreq= 1046
dur= 0.12500
beats= 160
wholenote= 1500
quarternote= 375
msec= 375
RingTones[0](-)
RingTones[0](-)
RingTones[0].frequency= 1046
RingTones[0].duration= 0.12500
```

Reeksen en **objecten** worden weergegeven in een van beide **un-uitgebreid** of **uitgebreid** formaat, met ofwel een trailing plus ' (+) ' of minus ' (-) ' teken, respectievelijk. Het symbool voor een reeks **x** shows als '**x** [] ' . Om uitbreiden te tonen alle elementen van de reeks, klik je gewoon met één muisklik '**x** [] (+) ' in de **VariabelenPaneel** . Om naar samentrekken terug te keren naar een niet-uitgebreid-weergave, klikt u op de '**x** [] (-) ' . De standaardinstelling voor de uitgebreid voor een object '**p1** ' shows als '**p1** (+) ' Om uitbreiden het om alle leden van dat te tonen '**class**' of '**struct**' bijvoorbeeld, één klik op '**p1** (+) ' in de **VariabelenPaneel** . Om samentrekken terug te zetten naar een niet-uitgebreid-weergave, klikt u één keer op '**p1** (-) ' . Als jij een enkele muisklik op een lijn om markeer in donkerolijf (Het kan eenvoudig variabele of het aggregaat ' (+) ' of ' (-) ' lijn een reeks of object, of een enkel element of reeks object-lid), dan het doen van een **Uitvoeren-Totdat** zal leiden uitvoering te

hervatten en bevroren op het volgende *write-toegang* ergens in dat geselecteerde aggregaat, of die enkel geselecteerd variabele locatie.

Tijdens gebruik **Stap** of **Uitvoeren**, updates van weergegeven variabele-waarden worden gemaakt volgens de gebruikersinstellingen die in het menu zijn gemaakt **VarRefresh** - dit biedt een volledig scala aan gedrag van minimale periodieke updates tot volledige onmiddellijke updates. Verminderde of minimale updates zijn nuttig om de CPU-belasting te verminderen en kunnen nodig zijn om te voorkomen dat uitvoering achterloopt in real-time onder wat anders buitensporig zou zijn **VariabelenPaneel** window-update wordt geladen. Wanneer **Animeren** is van kracht, of als het **Markeer-wijzigingen** menuoptie is geselecteerd, verandert de waarde van een variabele tijdens **Uitvoeren** zal resulteren in de weergave van de weergegeven waarde *per direct*, en het wordt gemarkeerd - dit veroorzaakt de **VariabelenPaneel** om te scrollen (indien nodig) naar de regel die die variabele bevat en uitvoering is niet langer realtime !.

Wanneer uitvoering befrist na **Stap**, **Uitvoeren-Tot**, **Uitvoeren-Totdat** of **Uitvoeren -dan- Halt**, de **VariabelenPaneel** markeert de variabele die overeenkomt met de **adreslocatie (s) die zijn gewijzigd** (indien aanwezig) door de **allerlaatste instructie** tijdens die uitvoering (inclusief door variabele-verklaring initialisaties). Als die instructie **helemaal vol object of reeks**, de **bovenliggende (+) of (-) lijn** voor dat aggregaat wordt gemarkeerd. Als in plaats daarvan de instructie is gewijzigd plaats dat is op dit moment zichtbaar, dan wordt het gemarkeerd. Maar als de gemodificeerde locatie (s) zich momenteel binnenin bevindt (zijn) een VN-uitgebreid reeks of object, die aggregaten **ouderlijn** krijgt een **cursief lettertype markeren** als een visuele cue waar iets erin geschreven is - klikken naar uitbreiden zal het dan veroorzaken **laatste** gewijzigd element of lid om gemarkeerd te worden.



De **Editeren/Opvolgen** window geeft je **de mogelijkheid om elke variabele-waarde te volgen tijdens uitvoering**, of te **verander de waarde ervan in het midden van (stopgezet) programma uitvoering** (dus je kunt testen wat het effect zou zijn om verder te gaan met die nieuwe waarde). **Halt** uitvoering eerst dan **dubbelklik links** op de variabele waarvan u de waarde wilt bijhouden of wijzigen. Om eenvoudig de waarde te bewaken tijdens programma uitvoering, **laat het dialoogvenster open** en dan een van de **Uitvoeren** of **Stap**

opdrachten - de waarde wordt bijgewerkt in **Editeren/Opvolgen** volgens dezelfde regels die gelden voor updates in de **VariabelenPaneel**. **De variabele-waarde wijzigen**, vul de waarde van de edit-box in, en **Aanvaarden**. Vervolg uitvoering (met een van de **Stap** of **Uitvoeren** commando's) om die nieuwe waarde vanaf dat punt voorwaarts te gebruiken (of u kunt het **Terugkeren** naar de vorige waarde).

Op programma Laden of Reset merk op dat alles *niet-geïnitieerde waarde-variabelen worden teruggezet naar waarde 0 en alle niet-geïnitieerde aanwijzer-variabelen worden teruggezet naar 0x0000*.

LabtafelPaneel

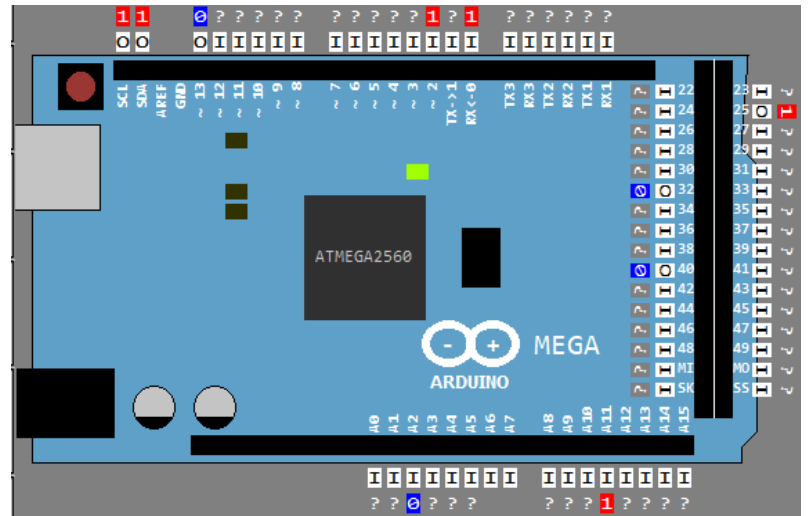
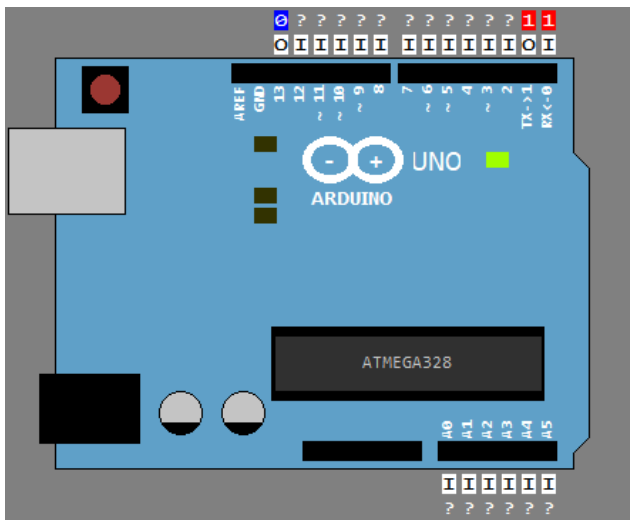
De LabtafelPaneel toont een 5-volt 'Uno' of 'Mega' printplaat die wordt omringd door een set 'I/O' apparaten die u kunt selecteren / aanpassen en aansluiten op uw gewenste 'Uno' of 'Mega' pins.

De 'Uno' of 'Mega'

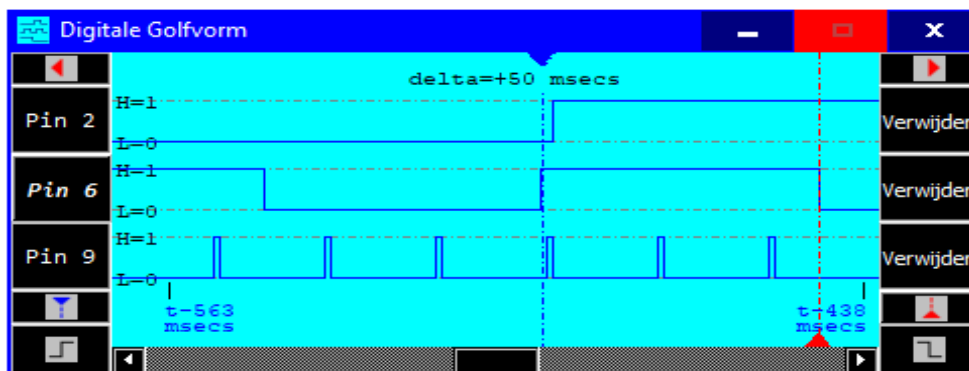
Deze is een afbeelding van de 'Uno' of 'Mega' printplaat en de ingebouwde LED's. Wanneer u een nieuwe programma in UnoArduSim laadt, ondergaat deze, als deze met succes is geparseerd, een "gesimuleerde downloaden" naar de printplaat die de manier nabootst van een daadwerkelijke printplaat gedraagt zich - u zult de seriële RX en TX LED zien knipperen (samen met activiteit op pins 1 en 0 die *bedraad voor seriële communicatie met een hostcomputer*). Dit wordt onmiddellijk gevolgd door een pin 13 LED-flitser die printplaat-reset en (en UnoArduSim automatische stop aan) het begin van uw geladen programma uitvoering betekent. U kunt deze weergave en bijbehorende laadvertraging vermijden door de selectie ongedaan te maken **Toon Downloaden** van **Configureren | Voorkeuren**.

Met de window kunt u de logische niveaus van de digitaal visualiseren op alle 20 'Uno' pins of alle 70 'Mega' pins ('1' op rood voor 'HIGH' , '0' op blauw voor 'LOW' , en '?' op grijs voor een onbepaalde onbepaalde spanning), en geprogrammeerd richtingen ('I' voor 'INPUT' , of 'O' voor 'OUTPUT'). Voor pins die gepulseerd worden met PWM via 'analogWrite()' , of door 'tone()' , of door 'Servo.write()' , de kleur verandert in paars en het weergegeven symbool wordt '^' .

Let daar op ***Digitaal pins 0 en 1 zijn hard-wired via 1-kOhm-weerstanden naar de USB-chip voor seriële communicatie met een hostcomputer.***





Links te klikken op elke 'Uno' of 'Mega' zal pin a openen **Pins Golfvormen Digitale** window die het verleden weergeeft **een seconde waard** van **digitaal-niveau activiteit** op die pin. U kunt op andere pins klikken om deze aan de Pins Golfvormen Digitale-display toe te voegen (tot een maximum van 4 golfvormen op elk willekeurig moment).



Klik om pagina links of rechts te bekijken, of gebruik de toetsen Home, PgUp, PgDn, End

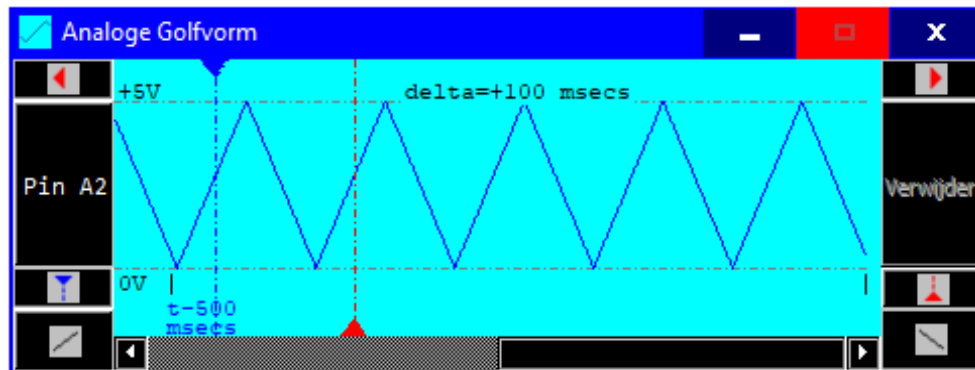
Een van de weergegeven golfvormen is de **actieve pin** golfvorm , aangegeven door de "Pin" -knop die wordt weergegeven als ingedrukt (zoals in de bovenstaande Pins Golfvormen Digitale-schermafbeelding). U kunt een golfvorm selecteren door op de Pin-nummerknop te klikken en vervolgens de betreffende randpolariteit te selecteren door op de juiste stijgende / dalende randpolariteitsselectieknop te klikken.  of  of met behulp van de sneltoetsen **pijlje omhoog** en **pijlje** . Dat kan je dan **springen** de actieve cursor (de blauwe of rode cursorlijnen met de weergegeven delta-tijd) achteruit of vooruit naar de digitaal-rand met de gekozen polariteit **van deze actieve pin** golfvorm met behulp van de cursorknoppen (,  of ,  (afhankelijk van welke cursor was eerder geactiveerd met  of ) of gebruik eenvoudig de klaviertoetsen ← en → .

Klik op de gekleurde activeringsknop om een cursor te activeren ( of  hierboven weergegeven) - **hiermee wordt ook de weergave doorlopen naar de huidige locatie van de cursor** . Als alternatief kunt u snel de activering tussen cursors (met hun respectievelijk gecentreerde weergaven) gebruiken met behulp van de snelkoppeling 'Tab' sleutel.







Jij kan **springen** de momenteel geactiveerde cursor voorbij **met de linkermuisknop ergens klikken** in het scherm golfvorm op het scherm. Als alternatief kunt u de rode of blauwe cursorlijn selecteren door er rechts bovenop te klikken (om deze te activeren) en vervolgens **sleep het naar een nieuwe locatie** en release. Wanneer een gewenste cursor zich momenteel ergens buiten het scherm bevindt, kunt u dit doen **klik met de rechtermuisknop ergens** in de weergave om naar die nieuwe locatie op het scherm te springen. Als beide cursors al op het scherm staan, kunt u met de rechtermuisknop klikken tussen de geactiveerde cursor.

Om IN en ZOOM UIT te ZOOMEN (zoom is altijd gecentreerd op de ACTIVE-cursor), gebruik je het muiswiel of de sneltoetsen CTRL-pijl-omhoog en CTRL-pijl-omlaag.

In plaats daarvan een **klik met de rechtermuisknop op elke 'Uno' of 'Mega' pin** opent een **Pin Golfvorm Analoge** window die de **afgelopen een seconde waard** van **analoge-niveau activiteit** op die pin. In tegenstelling tot de Pins Golfvormen Digitale window, kunt u analoge-activiteit op slechts één pin tegelijk weergeven.



Klik om pagina links of rechts te bekijken, of gebruik de toetsen Home, PgUp, PgDn, End

Jij kan **springen** de blauwe of rode cursorlijnen naar het volgende stijgende of dalende "hellingspunt" met behulp van de voorwaartse of achterwaartse pijlknoppen (,  of ,  , opnieuw afhankelijk van de geactiveerde cursor, of gebruik de ← en → toetsen) in combinatie met de stijgende / dalende hellingskeuzeknoppen  ,  (het "hellingspunt" vindt plaats waar de analoge-spanning de ATmega pin hoge digitaal-logisch niveau-drempelwaarde passeert). Als alternatief kunt u opnieuw klikken om te springen of deze cursorlijnen slepen die vergelijkbaar zijn met hun gedrag in de Pins Golfvormen Digitale window

persing 'Ctrl-S' binnen **Met window kunt u de golfvorm opslaan (X, Y) gegevens** naar een tekst bestand van uw keuze, waar **X** bevindt zich in microseconden vanaf de linkerkant, en **Y** zit in bouten.

'I/O' Apparaten

Een aantal verschillende apparaten omringen de 'Uno' of 'Mega' op de omtrek van de **LabtafelPaneel**. "Kleine" 'I/O' apparaten (waarvan u in totaal maximaal 16 kunt hebben) bevinden zich langs de linker- en rechterkant van de Paneel. "Grote" 'I/O' apparaten (waarvan u in totaal maximaal 8 heeft) hebben "actieve" elementen en bevinden zich langs de boven- en onderkant van de **LabtafelPaneel**. Het gewenste aantal van elk type beschikbare 'I/O' apparaat kan worden ingesteld met behulp van het menu **Configureren | 'I/O' Apparaten**.

Kleinere 'I/O' Apparaten	Big 'I/O' Apparaten
Druknop 2	Servomotor 1
Geschakelde Weerstand 4	Gelijktroommotor 1
Piezo Luidspreker 2	Stappenmotor 1
gekleurde LED 6	Pulsed Stappenmotor
4-LED Row	Pulsgenerator 1
7-Segment LED	Functie Generator 1
Pins Verbindingsdraad	SFT Serial
Analoge Schuifweerstand 2	SR Slaaf
Totaal (max 16) 16	1-Wire Slaaf
Laden	Digitale Enkele Puls Generator
Opslaan Als	SPI Slaaf 1
OK	I2C Slaaf
Afbreken	Tekst LCD (SPI)
	Tekst LCD (I2C)
	Tekst LCD (D4)
	Uitbreidingspoort (SPI)
	Uitbreidingspoort (I2C)
	Multiplexer LED (SPI)
	Multiplexer LED (I2C)
	ProgIO UNO
	Totaal (max 8) 7

Elke 'I/O' apparaat heeft een of meer pin-bijlagen die worden weergegeven als een **twee-cijfer** pin-nummer (00, 01, 02, ... 10, 11, 12, 13 en ofwel A0-A5, of 14-19, daarna) in een corresponderende edit-box. Voor pin-nummers 2 tot en met 9 kunt u eenvoudig de enkele cijfer invoeren - de eerste 0 wordt automatisch verstrekt, maar voor pins 0 en 1 moet u eerst de eerste 0 invoeren. Invoer is *normaal gesproken* aan de linkerkant van een 'I/O' apparaat en uitgangen zijn *normaal gesproken* aan de rechterkant (ruimte toelaat). Alle 'I/O' apparaten zullen direct reageren op pin-niveaus en pin-niveau-veranderingen, dus reageren op elke bibliotheek functies gericht op hun aangesloten pins, of op geprogrammeerd '`digitalWrite()`' (voor "bit-banged" werking).

U kunt meerdere apparaten's verbinden met dezelfde ATmega pin zolang dit geen **elektrische conflict**. Zo'n conflict kan worden gemaakt door een 'Uno' of 'Mega' pin als '**OUTPUT**' rijden tegen een apparaat met een sterke geleiding (lage impedantie) (bijvoorbeeld rijden tegen een 'FUNCGEN'-uitgang of een 'MOTOR' **Enc** uitvoer), of door twee verbonden apparaten die met elkaar concurreren (bijvoorbeeld zowel een 'PULSER' als een 'PUSH' - knop bevestigd aan dezelfde pin). Een dergelijke conflict zou rampzalig zijn in een echte hardware-implementatie en dus niet worden toegestaan en door een pop-upberichtbox aan de gebruiker worden gemarkeerd).

Het dialoogvenster kan worden gebruikt om de gebruiker de typen en nummers van de gewenste 'I/O' apparaten te laten kiezen. Vanuit dit dialoogvenster kunt u ook **Opslaan** 'I/O' apparaten naar een tekst bestand en / of **Laden** 'I/O' apparaten van een eerder opgeslagen (of bewerkte) tekst bestand (**inclusief alle pin-verbindingen en klikbare instellingen en alle ingetypte bewerk-boxwaarden**).

Merk op dat vanaf versie 1.6 de waarden in de period, delay en pulse-width edit-boxes in de relevante IO apparaten het achtervoegsel 'S' (of 's') kunnen krijgen. Dat geeft aan dat ze zouden moeten zijn **geschubd** volgens de positie van een globaal '**I/O ____S**' schuifregelaar die wordt weergegeven in de window **Tool-Bar**. Met die schuif helemaal naar rechts is de schaafactor 1,0 (eenheid), en met de schuif helemaal naar links is de schaafactor 0,0 (afhankelijk van minimumwaarden afgedwongen door elke specifieke 'I/O' apparaat). U kunt meer dan één edit-box-waarde schalen **gelijktijdig** met behulp van deze schuifregelaar. Met deze functie kunt u de schuifregelaar slepen tijdens het uitvoeren eenvoudig emuleren van veranderende pulsbreedten, -perioden en -vertragingen voor de aangesloten 'I/O' apparaten.

De rest van deze sectie bevat beschrijvingen voor elk type apparaat.

Verschillende van deze apparaten **ondersteuning van schaling van hun ingetypte waarden** met behulp van de schuifregelaar op de window **Tool-Bar**. Als de apparaat-waarde de letter 'S' als achtervoegsel heeft, wordt de waarde

ervan vermenigvuldigd met een schaafactor (tussen 0,0 en 1,0) die wordt bepaald door de schuif-duimpositie, afhankelijk van de apparaat-minimumwaardebepierking (1.0 is helemaal rechts, 0.0 is volledig links) --zie '**I/O ____S**' onder elk van de hieronder



beschreven slang apparaten.

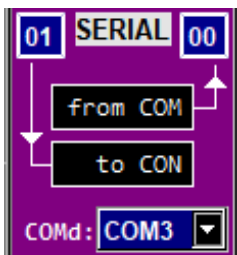
'Serial' Monitor ('SERIAL')



Deze 'I/O' apparaat maakt ATmega hardware-gemedieerde seriële invoer en uitvoer (via de 'Uno' of 'Mega' USB-chip) op 'Uno' of 'Mega' pins 0 en 1 mogelijk. De overdrachtsnelheid wordt ingesteld met behulp van de vervolgkeuzelijst onderaan - de geselecteerde overdrachtsnelheid **moet overeenkomen** de waarde die uw programma doorgeeft aan de '`Serial.begin()`' functie voor juiste verzending / ontvangst. *De seriële communicatie is vastgelegd op 8 databits, 1 stopbit en geen pariteitsbit.* Je mag **Loskoppelen** (blanco) **maar niet vervangen** TX pin 00, maar niet RX pin 01.

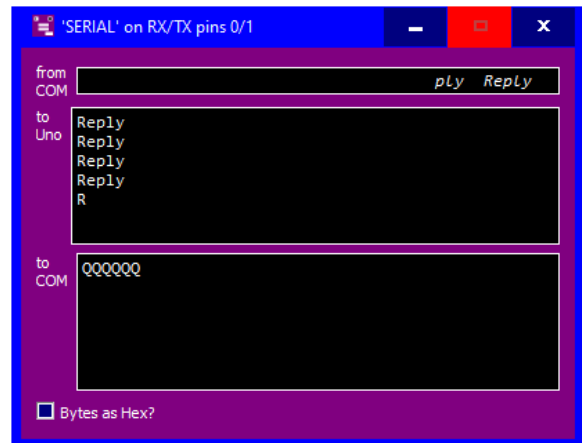
Als u toetsenbordinput naar uw programma wilt verzenden, typt u een of meer tekens in de bovenste (TX-tekens), bewerkt u window en vervolgens druk op de '**Enter**' toets op het toetsenbord. (tekens worden cursief gedrukt om aan te geven dat de verzendingen zijn begonnen) - of als de getypte tekens al in uitvoering zijn, worden ze cursief weergegeven. U kunt dan de '`Serial.available()`' en '`Serial.read()`' functies om de tekens te lezen in de volgorde waarin ze zijn ontvangen in de pin 0-buffer (het meest linkse getypte teken wordt als eerste verzonden). Geformatteerde tekstuele en numerieke afdrukken, of niet-geformatteerde byte-waarden, kunnen naar de onderste console-uitgang (RX-tekens) window worden gestuurd door de Arduino aan te roepen '`print()`', '`println()`' of '`write()`' functies.

Bovendien, **een grotere window voor het instellen / bekijken van TX- en RX-tekens kan worden geopend door te dubbelklikken (of rechts klikken) op deze 'SERIAL' apparaat**. Deze nieuwe window heeft een groter bewerkingsvak voor TX-tekens en een afzonderlijke 'Send'-knop waarop kan worden geklikt om de TX-tekens naar de 'Uno' of 'Mega' te verzenden (op pin 0). Er is ook een selectievakje optie om backslash-escaped karaktersvolgorden zoals te herinterpreteren '`\n`' of '`\t`' voor niet-onbewerkte weergave.

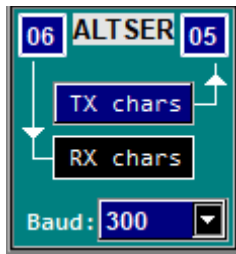


U kunt in plaats daarvan aan een 'COM'-poort koppelen door het meervoud 'values' op te geven en vervolgens een komma plus de bijbehorende poortnaam met dubbele aanhalingstekens toe te voegen na de baudrate in een 'myIODEvs.txt' bestand. In dat geval worden bytes die zijn ontvangen door de 'COM'-poort doorgestuurd naar de 'Uno'

of 'Mega' printplaat, en alle bytes die naar 'Serial' zijn geschreven, worden naar de 'COM'-poort gestuurd. De uitgebreid Monitor window toont in dit geval de volledige geschiedenis van bytes die in de 'COM'-poort zijn binnengekomen, evenals de bytes die door uw programma naar de aangesloten 'COM'-poort zijn gestuurd.



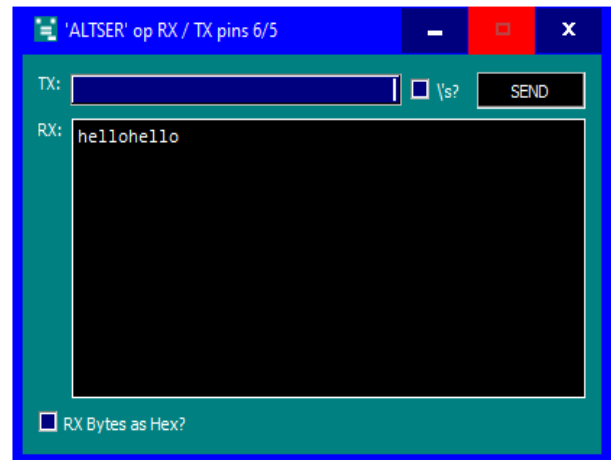
Afwisselend Serieel ('ALTSER')



Deze 'I/O' apparaat maakt bibliotheek-gemedieerde of, als alternatief, gebruiker "bit-banged", seriële invoer en uitvoer mogelijk op elk paar 'Uno' of 'Mega' pins dat u wilt invullen (**behalve voor** pins 0 en 1 voor hardware 'Serial' communicatie). Uw programma moet een `#include <SoftwareSerial.h>` regel in de buurt van de top als u de functionaliteit van die bibliotheek wilt gebruiken. Net als bij de op hardware gebaseerde 'SERIAL' apparaat, wordt de overdrachtsnelheid voor 'ALTSER' ingesteld met behulp van de vervolgkeuzelijst onderaan - de geselecteerde overdrachtsnelheid moet overeenkomen met de waarde die uw programma doorgeeft aan de `begin()` functie voor juiste verzending / ontvangst. *De seriële communicatie is vastgelegd op 8 databits, 1 stopbit en geen pariteitsbit.*

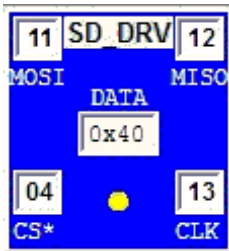
Ook, net als bij de hardware gebaseerd 'SERIAL', **een grotere window voor TX en RX instelling / weergave kan worden geopend door te dubbelklikken (of rechts klikken) op de ALTSER apparaat**.

Merk op dat in tegenstelling tot de hardware-implementatie van 'Serial', er is geen voorzien TX-buffer ondersteund door interne ATmega-interruptbewerkingen (alleen een RX-buffer), dus dat `write()` (of `print()`) oproepen blokkeren (dat wil zeggen, uw programma zal niet doorgaan totdat deze voltooid zijn).



SD Schijf ('SD_DRV')

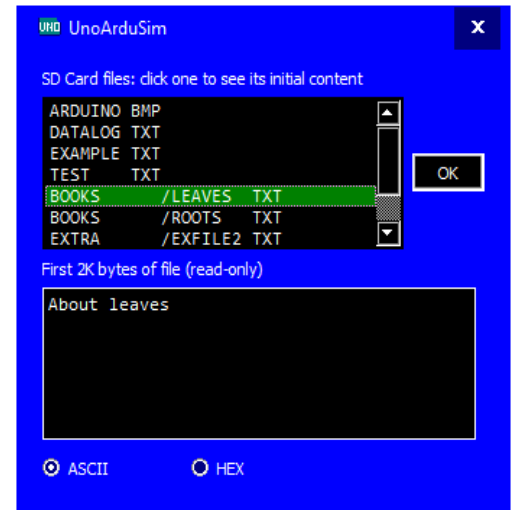
Deze 'I/O' apparaat maakt library-gemedieerde (maar **niet** "bit-banged") bestand invoer- en uitvoerbewerkingen op de 'Uno' of 'Mega' **SPI** pins (u kunt kiezen welke **CS *** pin die u zult gebruiken). Je programma kan eenvoudigweg `'#include <SD.h>'` lijn in de buurt van de top, en je kunt gebruiken `'<SD.h>'` functies OF rechtstreeks bellen `'SdFile'` functies zelf.



Een grotere window met mappen en bestanden (en inhoud) kan worden geopend door te dubbelklikken (of rechts klikken) op de 'SD_DRV' apparaat . Alle schijfinhoud is geladen van een SD subdirectory in de geladen programma-directory (als deze bestaat) bij `'SdVolume::init()'` , en wordt weerspiegeld datzelfde SD subdirectory op bestand `'close()'` , `'remove()'` , en verder

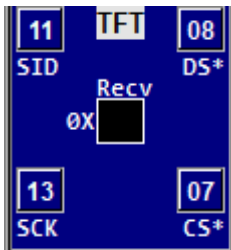
`'makeDir()'` en `'rmDir()'` .

Een gele LED knippert tijdens SPI-overdrachten en 'DATA' geeft de laatste 'SD_DRV' weer **antwoord** byte. Alle SPI-signalen zijn correct en kunnen worden bekeken in a **Golfvorm window**.



TFT-Scherm ('TFT')

Dit 'I/O' apparaat emuleert een Adafruit™ TFT omvang 1280by-160 pixels (in de oorspronkelijke rotatie = 0, maar bij gebruik van 'TFT.h' bibliotheek `'TFT.begin()'` initialisatie sets roteerbaar = 1, dat een "landschap" Gezien 160 per 128 pixels) geeft. U kunt deze apparaat besturen door te bellen naar de functies van de `'TFT.h'` library (die voor het eerst vereist `'#include <TFT.h>'`), of u kunt de SPI-systeem te gebruiken om je eigen volgorde van de bytes te sturen naar besturen het.

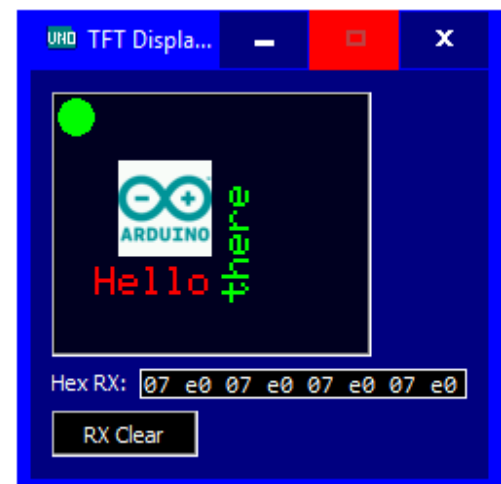


De TFT wordt altijd verbonden om 'SPI' pins 'MOSI' (voor 'SID') en 'SCK' (voor 'SCK') - deze kan niet worden gewijzigd. De 'DS*' pin is voor data / opdracht te selecteren ('LOW' selecteert data mode), en 'CS*' pin is de actieve laag chipselectie

Er is geen Reset pin voorzien, zodat u een hardware reset van het niet kunnen doen Dit apparaat door het aandrijven van een pin laag (als `'TFT::begin()'` functie probeert te doen als je

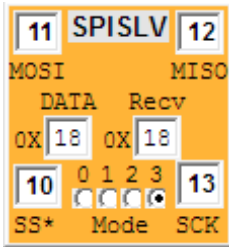
een geldig 'reset' pin aantal verstrekken als derde parameter aan de `'TFT(int cs, int ds, int rst)'` constructeur). De apparaat heeft echter een verborgen verbinding met het systeem Reset lijn dus het reset zichzelf elke keer dat u op de belangrijkste UnoArduSim Reset werkbalkpictogram, of de 'Uno' of 'Mega' printplaat resetknop ..

Door **dubbelklikken** (of **rechtermuisknop te klikken**) Op deze apparaat, window groter wordt geopend dat volledige 160 per 128 pixels LCD display, samen met de meest recent ontvangen 8 bits (zie hieronder)



Configureerbare SPI Slaaf ('SPISLV')

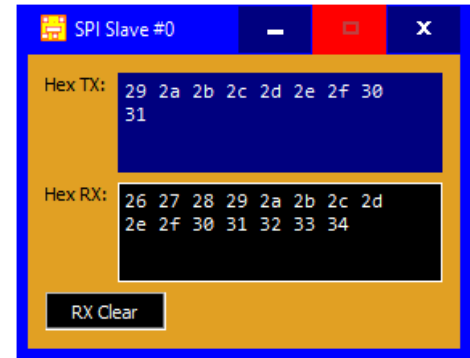
Deze 'I/O' apparaat emuleert een SPI-slave met geselecteerde modus met een actief-laag **SS** * ("slave-select") pin bestuurt de **MISO** output pin (wanneer **SS** * is hoog, **MISO** is niet gedreven). Uw programma moet een `'#include <SPI.h>'` regel als u de functionaliteit van de ingebouwd SPI Arduino object en bibliotheek wilt gebruiken. Je kunt er ook voor kiezen om je eigen "bit-banged" te maken **MOSI** en **SCK** signalen naar besturen deze apparaat.



De apparaat detecteert randovergangen op zijn **CLK** invoer volgens de geselecteerde modus (`'MODE0'`, `'MODE1'`, `'MODE2'` of `'MODE3'`), die moet worden gekozen om overeen te stemmen met de geprogrammeerd SPI-modus van uw programma.

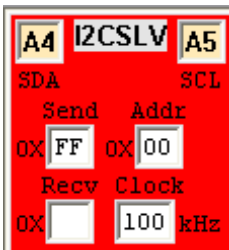
Door te dubbelklikken (of rechts klikken) op de apparaat, kunt u een grotere metgezel window openen dat in plaats daarvan staat y toe ou om

een 32-byte-maximum buffer in te vullen (om SPI apparaten te emuleren die automatisch hun gegevens retourneert) en om de laatste 32 ontvangen bytes te zien (allemaal als hexadecimale paren). Let daar op de volgende TX-bufferbyte is automatisch alleen naar 'DATA' verzonden na een volle `'SPI.transfer()'` heeft afgerond!



Tweedraads I2C Slaaf ('I2CSLV')

Deze 'I/O' apparaat emuleert alleen a *slave-modus* apparaat. Aan de apparaat kan een I2C-busadres worden toegewezen met behulp van een ingang met twee hex-cijfer in zijn 'Addr'-edit-box (deze reageert alleen op I2C bustransacties waarbij het toegewezen adres is betrokken). De apparaat verzendt en ontvangt gegevens over de open-afvoer (pull-down-only) **SDA** pin en reageert op het buskloksignaal op de open-afvoer (alleen-pull-down) **SCL** pin. Hoewel de 'Uno' of 'Mega' de busmaster is die verantwoordelijk is voor het genereren van de **SCL** signaal, deze slaaf apparaat zal ook trekken **SCL** laag tijdens zijn lage fase om (indien nodig) de bus-low-time uit te breiden tot een die geschikt is voor zijn interne snelheid (die kan worden ingesteld in zijn 'Clock'-edit-box).

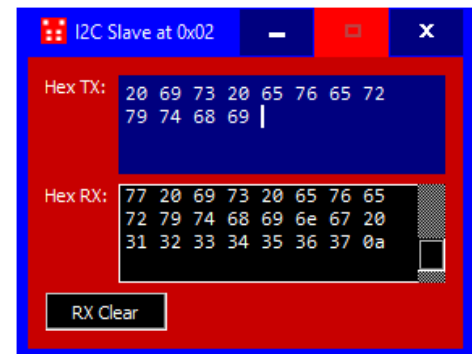


Uw programma moet een `'#include <Wire.h>'` regel als u de functionaliteit van de `'TwoWire'` bibliotheek om te communiceren met deze apparaat. Als alternatief kunt u ervoor kiezen om uw eigen bit-banged-gegevens en kloksignalen naar besturen te maken met deze slave apparaat.

Een enkele byte voor verzending terug naar de 'Uno' of 'Mega'-master kan worden ingesteld in het 'Send'-bewerkingsvak en een enkele (meest recent ontvangen) byte kan worden bekeken

in zijn (alleen-lezen) 'Recv'-bewerkingsvak. Let daar op de 'Send'-edit-boxwaarde weerspiegelt altijd de volgende byte voor verzending vanuit deze apparaat interne gegevensbuffer.

Door te dubbelklikken (of rechts klikken) op de apparaat, kunt u een grotere metgezel window openen waarmee u in plaats daarvan een maximaal 32-byte-maximum FIFO-buffer kunt invullen (om TWI apparaten met een dergelijke functionaliteit te emuleren) en om (tot een maximum van 32) bytes van de meest recent ontvangen gegevens (als een twee- hex-cijfer weergave van 8 bytes per regel). Het aantal regels in deze twee invoervelden komt overeen met de gekozen TWI-buffergrootte (die kan worden geselecteerd met **Configureren | Voorkeuren**). Dit is als optie toegevoegd sinds de Arduino `'Wire.h'` bibliotheek gebruikt **vijf** dergelijke RAM-buffers in zijn implementatiecode, die RAM-geheugen duur is. Door de Arduino-installatie te bewerken `'Wire.h'` bestand om gedefinieerde constante te wijzigen `'BUFFER_LENGTH'` (en ook de begeleidende bewerking bewerken `'utility/twi.h'` bestand om te veranderen TWI-buffellengte) om in plaats daarvan ofwel 16 of 8, een gebruiker te zijn *kon* aanzienlijk verminderen de RAM-geheugen overhead van de 'Uno' of 'Mega' als doelwit **hardware-implementatie** - UnoArduSim weerspiegelt daarom deze reële mogelijkheid door **Configureren | Voorkeuren** .

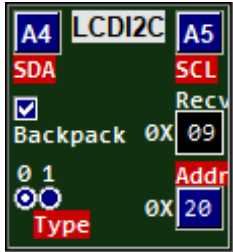


Tekst LCD I2C ('LCDI2C')

Dit 'I/O' apparaat emuleert een 1,2, of 4-line karakter-LCD, in een van de drie modi:

- a) rugzak Typ 0 (Adafruit stijl poort expander met hardware met I2C bus adres 0x20-0x27)
- b) het type rugzak 1 (DFRobot stijl poort expander met hardware die I2C-bus adres 0x20-0x27)
- c) geen backpack (modus Native geïntegreerde I2C interface beschikt over I2C busadres 0x3C-0x3F)

Ondersteunende bibliotheek code voor elke apparaat functie is voorzien in de map 'include_3rdParty' van uw UnoArduSIm installatiemap: 'Adafruit_LiquidCrystal.h', 'DFRobot_LiquidCrystal.h' en 'Native_LiquidCrystal.h' Respectievelijk.



De apparaat kan elk I2C-bus adres behulp van een twee-hex-cijfer invoering in het 'Addr' bewerken-box (het zal alleen reageren op worden toegewezen I2C bus transacties met betrekking tot de toegewezen adres). De apparaat ontvangt bus adres en data (en reageert met ACK = 0 of NAK = 1) op de open-drain (pull-down only) SDA pin. U kunt alleen schrijven LCD-commando's en DDRAM data - u **kan niet** teruglezen van gegevens uit de schriftelijke DDRAM locaties.



Dubbeltklik of **klik met de rechtermuisknop** naar het LCD-scherm monitor window van waaruit u ook de grootte van het scherm en het karakter set kunt instellen.

Tekst LCD SPI ('LCDSPI')

Dit 'I/O' apparaat emuleert een 1,2, of 4-line karakter-LCD, in een van de twee modi:

- a) rugzak (Adafruit stijl SPI-poort expander)
- b) geen backpack (modus Native geïntegreerde SPI -interface - zoals hieronder)

Ondersteunende bibliotheek code voor elke apparaat functie is voorzien in de map 'include_3rdParty' van uw UnoArduSIm installatiemap: 'Adafruit_LiquidCrystal.h' ,, en 'Native_LiquidCrystal.h' Respectievelijk.



Pin 'SID' is seriële data in, 'SS' is de actieve-low apparaat-selecteren, 'SCK' is de klok pin en 'RS' is de data / opdracht pin. U kunt alleen schrijven LCD-commando's en DDRAM data (alle SPI transacties schrijft) - u **kan niet** teruglezen van gegevens uit de schriftelijke DDRAM locaties.

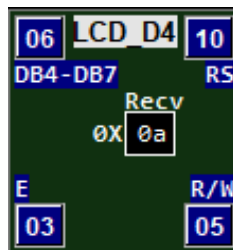
Dubbeltklik of **klik met de rechtermuisknop** naar het LCD-scherm monitor window van waaruit u ook de grootte van het scherm en het karakter set kunt instellen.



Tekst LCD D4 ('LCD_D4')

Dit 'I/O' apparaat emuleert een 1,2, of 4-line karakter-LCD met een 4-bit-parallele bus interface. De gegevens bytes worden geschreven / gelezen in **twee helften** op de 4 gegevens pins 'DB4-DB7' (indien het invoervak bevat *het laagste nummer van zijn 4 opeenvolgende pin nummers*) - data wordt geklokt op dalende flank bij de 'E' (enable) pin, met datarichting bestuurd door 'R/W' pin en LCD data / command mode door 'RS' pin.

Ondersteuning archiefcode is aangebracht binnen de 'include_3rdParty' map van uw UnoArduSIm installatiemap: 'Adafruit_LiquidCrystal.h', En 'Native_LiquidCrystal.h' beiden werken.



Dubbeltklik of **klik met de rechtermuisknop** naar het LCD-scherm monitor window van waaruit u ook de grootte van het scherm en het karakter set kunt instellen.



Multiplexer LED I2C ('MUXI2C')

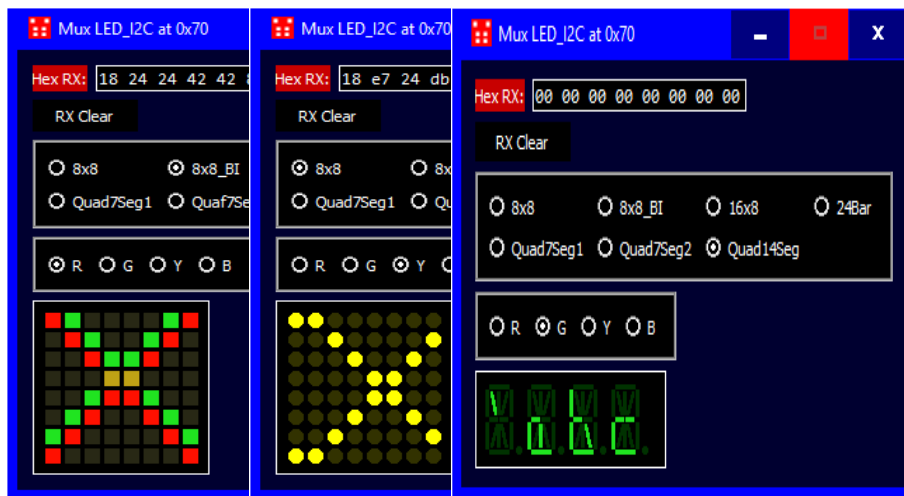
Dit 'I/O' apparaat emuleert een I2C-interface HT16K33 controller (having I2C-bus-adres 0x70-0x77) waartoe één van verscheidene verschillende soorten gemultiplext LED schermen worden bevestigd:



- a) 8x8 of 16x8 LED reeks
- b) 8x8 tweekleurige LED reeks
- c) 24-bi-color-LED bar
- d) twee stijlen 4-cijfer 7-segment displays
- e) een 4-cijfer 14-segment alfanumeriek display

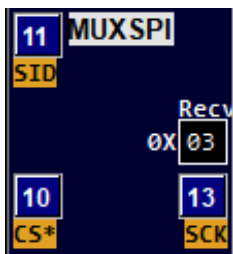
Alle worden ondersteund door de 'Adafruit_LEDBackpack.h' code aangebracht binnen de 'include_3rdParty' map:

Dubbelklik (Of klik met de rechtermuisknop) naar een grotere window openen te kiezen en te bekijken een van de vele gekleurde LED displays.



Multiplexer LED SPI ('MUXSPI')

Een multiplex-LED controller op basis van de MAX6219 een met 'MAX7219.h' code die in de map 'include_3rdParty' om besturen maximaal acht 7-delige cijfers.

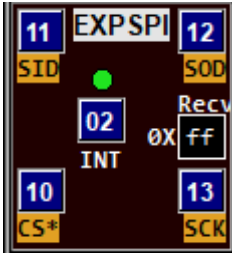


Dubbelklik (Of klik met de rechtermuisknop) naar een grotere window openen bekijken de gekleurde 8-cijfer 7-segment- display.



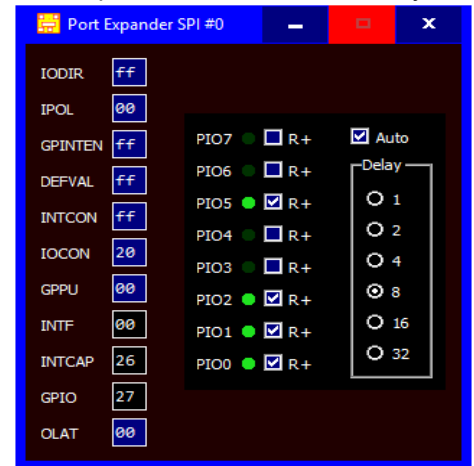
Uitbreidingspoort SPI ('EXPSPi')

Een 8-bits port expander betrokken op het MCP23008 een met 'MCP23008.h' code aangebracht binnen het 'include_3rdParty' map. U kunt schrijven naar MCP23008 registers en lees de rug van de GPIO pin levels. Interrupts kunnen worden ingeschakeld op elke GPIO pin verandering - een triggered interrupt zal de 'INT' pin besturen.



Dubbelklik (Of klik met de rechtermuisknop) openen een **grotere window** zien de 8 GPIO poortlijnen en de bijgevoegde afstopweerstand. U kunt pull-ups handmatig wijzigen door te klikken op, of als bijlage een teller die ze op gezette tijden zal veranderen in een up-count manier. De snelheid waarmee het aantal incrementen wordt bepaald door de afbouw delay factor gekozen door de gebruiker (a 1x factor correspondeert met één increment ongeveer elke 30

milliseconden hogere vertragingfactoren te brengen langzamere-telsnelheid)

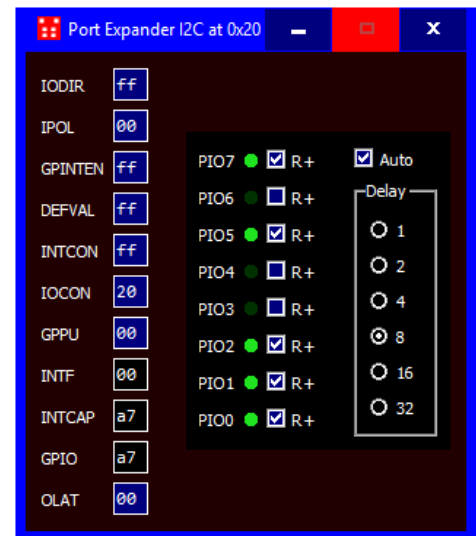


Uitbreidingspoort I2C ('EXPI2C')

Een 8-bits port expander betrokken op het MCP23008 een met 'MCP23008.h' code aangebracht binnen het 'include_3rdParty' map. Mogelijkheden overeenkomen met de 'EXPSPi' apparaat.



Dubbelklik (Of klik met de rechtermuisknop) naar een grotere window openen zoals va de apparaat 'EXPSPi'.



'1-Wire' Slaaf ('OWIISLV')

Deze 'I/O' apparaat emuleert een van een kleine set '1-Wire'-bus apparaten aangesloten op pin OWIO. U kunt een '1-Wire'-bus (met een of meer van deze slave '1-Wire' apparaten) maken op de 'Uno' of 'Mega' pin van uw keuze. Deze apparaat-cabine kan worden gebruikt door de 'OneWire.h' bibliotheek functies na het plaatsen van een '#include <OneWire.h>' lijn aan de bovenkant van je programma. Als alternatief kunt u ook bit-banged signalen op OWIO gebruiken op deze apparaat (hoewel dat erg lastig is om goed te doen zonder een elektrische conflict te veroorzaken - zo'n conflict is nog steeds mogelijk, zelfs wanneer u de 'OneWire.h' functies, maar dergelijke conflicten worden gerapporteerd in UnoArduSim).

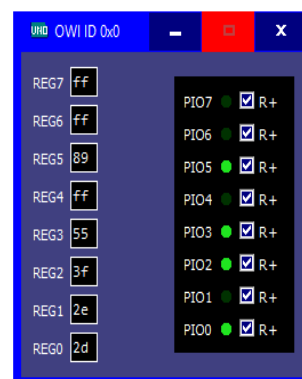
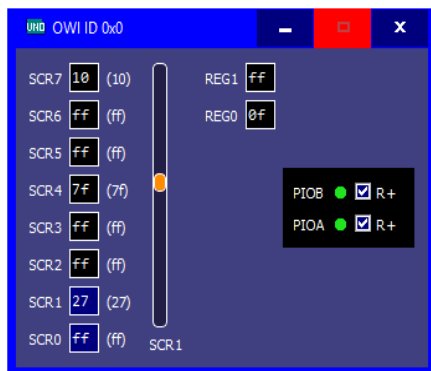


Elke real-world OWISLV apparaat moet een unieke 8-byte i hebben (64-bit!) intern serienummer - in UnoArduSim is dit vereenvoudigd door de gebruiker die een korte 1-bytes hexadecimaal levert 'ID' waarde (die standaard wordt toegewezen bij apparaat laden / optellen), plus de 'Fam' Gezinscode voor die apparaat. UnoArduSim herkent een kleine set familiecodes vanaf V2.3 (0x28, 0x29, 0x3A, 0x42) met betrekking tot temperatuursensor en parallelle IO (PIO) apparaten (een niet-herkende familiecode stelt de apparaat in als een generieke 8-bytes scratchpad apparaat met generieke sensor).

Als de apparaat-familie geen PIO-registers heeft, registreert deze **D0** en **D1** vertegenwoordigen de eerste twee bypass-krasblokken, anders vertegenwoordigen ze de PIO "Status" (actuele pin-niveaus) register en PIO pin-latchgegevensregister, respectievelijk.

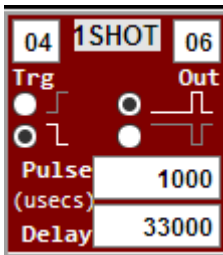
Door **dubbelklikken** (of **rechtermuisknop te klikken**) op de apparaat, een grotere **OWIMonitor** window is geopend. Vanuit die grotere window kun je alle apparaat-registers inspecteren, de scratchpadlocaties SCR0 en SCR1 wijzigen met bewerkingen en een schuifregelaar (opnieuw komen SCR0 en SCR1 alleen overeen met **D0** en **D1** als er geen PIO aanwezig is), of stel externe pin PIO-pull-ups in. Wanneer SCR0 en SCR1 worden bewerkt, onthoudt UnoArduSim deze bewerkte waarden als de gebruiker "voorkeur" die een initiële (beginnend vanaf Reset) waarde vertegenwoordigt die de ondertekende waarde uitvoer van de apparaat vertegenwoordigt; s sensor - de schuifregelaar wordt gereset naar 100% (een schaaftactor van 1,0) op het moment van de bewerking. Wanneer de schuifregelaar vervolgens wordt verplaatst, wordt de schuifregelaar verplaatst 'signed' de waarde in SCR1 wordt verkleind volgens de schuifregelaarpositie (schaaftactor van 1,0 naar beneden naar 0,0) - met zijn functie kunt u gemakkelijk de respons van uw programma testen op soepel veranderende sensorwaarden. .

Voor een apparaat met PIO pins, wanneer u het keuzevak pin-niveau instelt, onthoudt UnoArduSim deze gecontroleerde waarden als de huidige pull-ups die extern op de pins worden toegepast. Deze externe pull-upwaarden worden vervolgens gebruikt samen met de pin-vergrendelingsgegevens (register **D1**) om de uiteindelijke werkelijke pin-niveaus te bepalen en de groene LED op de PIO pin aan te steken of te doven (de pin gaat alleen 'HIGH' als een externe pull-up wordt toegepast, en het corresponderende **D1** grendel bit is een '1').



Digitale Enkele Puls ('1SHOT')

Deze 'I/O' apparaat emuleert een digitaal-one-shot die een puls met de gekozen polariteit en pulsbreedte op de 'Out' kan genereren pin, optredend na een gespecificeerde vertraging van een triggering edge ontvangen op zijn **trg** (trigger) input pin. Nadat de gespecificeerde triggerrand is ontvangen, begint timing en wordt een nieuwe triggerpuls pas herkend als de 'Out' puls is geproduceerd (en is volledig voltooid).



Een mogelijk gebruik van deze apparaat is simuleren ultrasone sensoren die een bereikpuls genereren in reactie op een triggerpuls. Het kan ook overall worden gebruikt waar u een pin-ingangssignaal wilt genereren dat is gesynchroniseerd (na de door u gekozen vertraging) naar een pin-uitgangssignaal dat door uw programma is gemaakt.

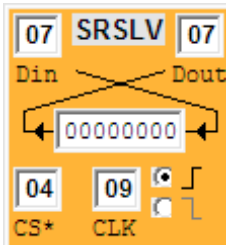
'Pulse' en 'Delay'-waarden kunnen worden geschaald van de hoofd-window **Tool-Bar**

'I/O ____S' schaalfactor schuifregelaar door toevoeging van het achtervoegsel 'S' (of 's') naar een van beide (of beide).

Een ander gebruik van deze apparaat is het testen van een programma die interrupts gebruikt, en je zou graag zien wat er gebeurt als een **specifieke programma-instructie** wordt onderbroken. Verbreek tijdelijk de 'I/O' Apparaat die u hebt aangesloten met pin 2 (of pin 3) en vervang deze door een '1SHOT' apparaat waarvan de 'Out' pin is verbonden met 'Uno' of 'Mega' pin 2 (of pin3, respectievelijk). U kunt dan de 'Trg'-invoer activeren (ervan uitgaande dat de gevoeligheid van de oplopende hoek is ingesteld) door het invoegen het instructiepaar { '**digitalWrite(LOW)** ', '**digitalWrite(HIGH)** ' } **lets ervoor** naar de instructie waarin u wilt dat de interrupt optreedt. Stel de 1SHOT; s 'Delay' in om de in de 'Out' geproduceerde puls te laten plaatsvinden binnen de programma-instructie die volgt op dit triggerende instructiepaar. Merk op dat sommige instructies interrupts maskeren (zoals '**SoftwareSerial.write(byte)** ', aen zo kan niet worden onderbroken.

Schuifregister Slaaf ('SRSLV')

Deze 'I/O' apparaat emuleert een eenvoudig schuifregister apparaat met een actief-laag **SS *** ("slave-select") pin bestuurt de '**Dout**' output pin (wanneer **SS *** is hoog, '**Dout**' is niet gedreven). Uw programma zou de functionaliteit van de ingebouwd SPI Arduino object en bibliotheek kunnen gebruiken. Je kunt er ook voor kiezen om je eigen "bit-banged" te maken '**Din**' en **CLK** signalen naar besturen deze apparaat.



De apparaat detecteert randovergangen op zijn **CLK** input die de verschuiving van het register activeert - de polariteit van de waargenomen **CLK** rand kan worden gekozen met behulp van een radioknopbediening. Op elke **CLK** rand (van de waargenomen polariteit), vangt het register het op **geraas** niveau in de minst significante bit (LSB) positie van het schuifregister, terwijl de resterende bits tegelijkertijd één positie naar de MSB-positie worden verschoven. telkens als **SS *** laag is, de huidige waarde in de MSB-positie van het schuifregister is gedreven '**Dout**'.

Programmeerbare 'I/O' Apparaat ('PROGIO')



Deze 'I/O' apparaat is eigenlijk een kale 'Uno' printplaat die je kunt programma (met een aparte programma) om een 'I/O' apparaat te emuleren wiens gedrag u volledig kunt definiëren. U kunt maximaal vier pins (IO1, IO2, IO3 en IO4) kiezen die deze slaaf 'Uno' gemeen heeft met de master (main' Uno 'or 'Mega') die in het midden van uw **LabtafelPaneel** . Net als bij andere apparaten wordt elke elektrische conflict tussen deze 'Uno'-slave en de master printplaat gedetecteerd en gemarkeerd. Merk op dat alle verbindingen zijn **direct** bedrade, **behalve voor pin 13** (waarbij wordt uitgegaan van een serie R-1K-weerstand tussen de twee pins om een elektrische conflict bij Reset te voorkomen). Vanaf V2.8 zijn de verbindingen tussen master en

slave pins **in kaart gebracht** : als de master ook een 'Uno' is, dit **standaard** mapping is een identiteit (behalve dat pin 1 is toegewezen aan pin 0, en vice versa om 'Serial'-communicatie mogelijk te maken); anders als de master een 'Mega' is, worden pins 1 en 0 opnieuw omgedraaid, **en alles** SPI en TWI pins zijn toegewezen voor directe verbinding tussen de corresponderende master- en slave-subsystemen. Deze **standaard** mapping kan zijn **overschreven** door in uw **IODevs.txt** bestand een expliciete lijst van 4 master pin-nummers die volgt op de PROGIO programma bestand-naam - als deze waarden -1 zijn, is dat hetzelfde als de standaardtoewijzing. Voor deze apparaat typten de pin-

nummers in ***zijn die van de slaaf 'Uno'***. De afbeelding aan de linkerkant toont de 4 slaaf pins gespecificeerd voor het SPI-systeem pins (SS *, MISO, MOSI, SCK) - ongeacht of hij een 'Uno' of een 'Mega' was, dit zou je in staat stellen om deze slaaf als een programma te programma generieke SPI-slave (of master) waarvan u het gedrag programmatisch kunt definiëren.

Door ***dubbelklikken*** (of ***rechtermuisknop te klikken***) op deze apparaat wordt een grotere window geopend om aan te geven dat deze 'Uno'-slave een eigen heeft **CodePaneel** en geassocieerd **VariabelenPaneel**, net zoals de master dat heeft. Het heeft ook zijn eigen **Tool-Bar**, . Waarop je kunt gebruiken **laden** en **controle uitvoering** van een slaaf programma - de pictogramacties hebben dezelfde sneltoetsen als die in de hoofd window. (**Laden** is **Ctrl-L**, **Opslaan** is **Ctrl-S** enz.). Eenmaal geladen, kunt u uitvoeren uit **een van beide** de Main UnoArduSim window, of van binnenuit deze Slaaf Monitor window - in beide gevallen blijven de Main programma en Slaaf programma vergrendeld in synchronisatie met de passage van realtime naarmate hun uitvoeringen vorderen. ***Om de CodePaneel te kiezen die de laden, zoeken en uitvoering-focus, Klik op de bovenliggende window-titelbalk - de niet-focus CodePaneel heeft dan zijn werkbalk acties worden grijs weergegeven.***

Enkele mogelijke ideeën voor slaaf apparaten die geprogrammeerd in deze 'PROGIO' apparaat zouden kunnen zijn, worden hieronder vermeld. Voor seriële, I2C- of SPI apparaat-emulatie kunt u geschikte programma-codering met reeksen gebruiken voor verzend- en ontvangstbuffers, om om complexe apparaat-gedrag te emuleren:

a) Een SPI-master of slave apparaat. UnoArduSimV2.4 heeft het '**SPI.h**' bibliotheek om slaafmodus S {PI-bewerking mogelijk te maken via een optioneel '**mode**' parameter in '**SPI.begin(int mode = SPI_MASTR)**'. Expliciet doorgeven '**SPI_SLV**' om de slavemodus te selecteren (in plaats van te vertrouwen op de standaardmaster-modus). U kunt nu ook een gebruikersinterrupt functie definiëren (laten we het noemen '**onSPI**') in om bytes over te zetten door te bellen een andere toegevoegde extensie '**SPI.attachInterrupt(user_onSPI)**'. Nu c Alling '**rxbyte=SPI.transfer(tx_byte)**' van binnenuit '**user_onSPI**' functie zal de interrupt-vlag wissen en zal **terugkeer per direct** met de zojuist ontvangen byte in uw variabele '**rxbyte**'. Als alternatief kunt u voorkomen dat u een bijlage toevoegt een SPI-interrupt, en in plaats daarvan gewoon bellen '**rxbyte=SPI.transfer(tx_byte)**' van binnenuit je hoofd programma - die oproep zal **blokkeer uitvoering** totdat een SPI-byte is overgedragen, en zal dan **terugkeer** met de nieuw ontvangen byte binnen '**rxbyte**'.

b) Een generiek **seriële 'I/O'** apparaat. U kunt met beide communiceren met de Master printplaat '**Serial**', of een '**SoftwareSerial**' gedefinieerd in uw slaaf programma- voor '**SoftwareSerial**' je moet definiëren '**txpin**' en '**rxpin**' tegengesteld aan die van de Matser zodat de slaaf ontvangt op de pin waarop de meester zendt (en vice versa), maar voor **Alleen** '**Serial**', de 1 en 0 pins zijn al voor je omgedraaid.

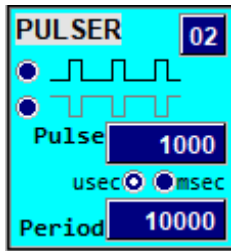
c) Een generieke 'I2C'-master of slave apparaat. De Slaaf-bewerking is nu toegevoegd om de implementatie van de UnoArduSim; te voltooien '**Wire.h**' bibliotheek (functies '**begin(address)**', '**onReceive()**' en '**onRequest**' zijn nu geïmplementeerd om de bewerkingen van de slavemodus te ondersteunen).

d) Een generieke digitaal **Pulser**. Gebruik makend van '**delayMicroseconds()**' en '**digitalWrite()**' roept naar binnen '**loop()**' in uw 'PROGIO' programma kunt u de '**HIGH**' en '**LOW**' intervallen van een pulstrein. Door een afzonderlijk toe te voegen '**delay()**' bel binnen in je '**setup()**' functie, je kunt het begin van deze pulstrein uitstellen. U kunt zelfs de pulsbreedtes variëren naarmate de tijd voortschrijdt met behulp van een teller variabele. U kunt ook een afzonderlijke gebruiken '**IOx**' pin als trigger voor het starten van de timing van een geëmuleerde '1Shot' (of double-shot, triple-shot etc.) apparaat, en u kunt de geproduceerde pulsbreedte regelen om op elke gewenste manier te veranderen naarmate de tijd vordert.

e) Een willekeurige signalering. Dit is een variatie op a digitaal **Pulser** die ook gebruik maakt van oproepen naar '**random()**' en '**delayMicroseconds()**' om willekeurige tijden te genereren waarop te '**digitalWrite()**' een signaal op elke gekozen pin gedeeld met de master. Alle vier gebruiken '**IOx**' pins zou vier gelijktijdige (en unieke) signalen mogelijk maken.

Pulsgenerator ('PULSER')

Dit 'I/O' apparaat emuleert een eenvoudige digitaal golfvorm pulse generator die een periodiek signaal dat kan worden toegepast op elke gewenste 'Uno of 'Mega" pin produceert.



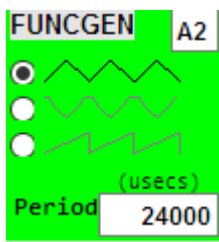
De periode en pulsbreedten (in microseconden) kan worden ingesteld met behulp geven-boxes-de minimaal toegestane periode 50 microseconden, en de minimale pulsbreedte 10 microseconden. U kunt kiezen tussen timing waarden in microseconden ('usec') en milliseconden ('msec'), en deze keuze wordt opgeslagen, samen met de andere waarden wanneer u 'Save' uit de Configureren | I / O Apparaten dialoog.

De polariteit kan ook worden gekozen: positief geavanceerde pulsen (0 tot 5V) of negatieve geavanceerde pulsen (5V tot 0V).

'Pulse' en 'Period' waarden kan worden geschaald van de belangrijkste **Tool-Bar** 'I/O_____S' schaaftactordecoder schuifbalk door het toevoegen van een suffix 'S' (of 's') aan één (of beide). Hiermee kunt u dan veranderen de 'Pulse' of 'Period' waarde **dynamisch** tijdens uitvoering.

Analoge Functie Generator ('FUNCGEN')

Deze 'I/O' apparaat emuleert een eenvoudige analoge golfvorm-generator die een periodiek signaal produceert dat op elke gekozen 'Uno of 'Mega" pin kan worden toegepast.



De periode (in microseconden) kan worden ingesteld met behulp van het invoervak - de minimaal toegestane periode is 100 microseconden. De golfvorm die wordt gemaakt, kan worden gekozen als sinusoïdale, driehoekige of zaagtand (om een blokgolf te maken, gebruikt u in plaats hiervan een 'PULSER'). In kleinere perioden worden minder monsters per cyclus gebruikt om de geproduceerde golfvorm te modelleren (slechts 4 monsters per cyclus in een periode van 100 microseconden).

De 'Period' waarde kan worden geschaald van de Main window **Tool-Bar** 'I/O_____S' schaaftactor schuifregelaar door als achtervoegsel de letter 'S' (of 's') toe te voegen.

Stappenmotor ('STEPR')

Deze 'I/O' apparaat emuleert een 6V bipolaire of unipolaire stappenmotor met een geïntegreerde bestuurder-controller gedreven door **of twee** (op P1 , P2) **of vier** (op P1 , P2 , P3 , P4) besturingssignalen. Het aantal stappen per omwenteling kan ook worden ingesteld. U kunt de 'Stepper.h' functies 'setSpeed()' en 'step()' naar besturen de 'STEPR'. Als alternatief kan 'STEPR' *reageer ook* naar je eigen land 'digitalWrite()' "bit-geneukt" besturen-signalen.



De motor is nauwkeurig gemodelleerd, zowel mechanisch als elektrisch. Motor-bestuurder spanningsdalingen en variërende reluctance en inductantie worden gemodelleerd samen met een realistisch traagheidsmoment met betrekking tot houdkoppel. De motorrotorwikkeling heeft een gemodelleerde weerstand van $R = 6$ ohm, en een inductantie van $L = 6$ milli-Henries die een elektrische tijdconstante van 1,0 milliseconde creëert. Vanwege de realistische modellering zult u merken dat zeer smalle pin-besturingselementen pulsen *niet krijgen* de motor naar stap - zowel vanwege de eindige stroomstijgtijd als het effect van rotorinertie. Dit komt overeen met wat wordt waargenomen bij het besturen van een echte stappenmotor van een 'Uno of 'Mega" met, uiteraard, een geschikte (**en vereist**) motor bestuurder chip tussen de motordraden in en de 'Uno of 'Mega"!

Een ongelukkige bug in de Arduino 'Stepper.h' bibliotheekcode betekent dat bij het resetten de stappenmotor zich niet in Stap-positie 1 (van vier stappen) bevindt. Om dit te overwinnen, zou de gebruiker moeten gebruiken 'digitalWrite()' in zijn / haar 'setup()' routine om de besturingsk K163-niveaus te initialiseren naar de 'step(1)' niveaus geschikt voor 2-pin (0,1) of 4-pin (1,0,1,0), en laat de motor 10 milliseconden bewegen naar de 12-uur referentie-initiële gewenste motorpositie.

AS van V2.6, dit apparaat bevat nu een 'sync' LED (groen voor gesynchroniseerde of rood wanneer uitgeschakeld door één of meer stappen) .Ook, Naast het aantal stappen per omwenteling, kunnen twee extra (verborgen) waarden eventueel worden gespecificeerd in de IODEvs.txt bestand de mechanische laad- bijvoorbeeld specificeren waarden 20, 50, 30 specificeert 20 stappen per omwenteling, een belasting traagheidsmoment 50 maal die van de rotor zelf, en lastkoppel van 30 procent volledige hold motorkoppel.

Let daar op **tandwielreductie wordt niet rechtstreeks ondersteund** vanwege ruimtegebrek, maar je kunt het in je programma emuleren door een modulo-N-teller variabele te implementeren en alleen te bellen 'step()' wanneer die teller op 0 komt (voor versnellingreductie met factor N).

Gepulseerde Stappenmotor ('PSTEPR')

Dit 'I/O' apparaat emuleert een 6V **stappenmotoren** bipolaire Stappenmotor met geïntegreerde controller bestuurder gedreven door **een gepulseerd 'Step'** pin, een actieve laag 'EN*' (Enable) pin, en 'DIR' (richting) pin . Het aantal volledige stappen per omwenteling kan ook direct ingesteld, samen met het aantal microstappen per volle stap (1,2,4,8 of 16). In aanvulling op deze instellingen, twee extra (verborgen) waarden kan eventueel worden gespecificeerd in de IODEvs.txt bestand de mechanische laad- bijvoorbeeld specificeren waarden 20, 4, 50, 30 specificeert 20 stappen per omwenteling, 4 microstappen per volle stap een belasting traagheidsmoment 50 maal die van de motor rotor zelf, en lastkoppel van 30 procent van de volledige hold motorkoppel.

U moet code schrijven om besturen de besturing pins adequaat.

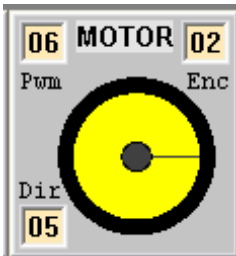


De motor wordt nauwkeurig gemodelleerd zowel mechanisch als elektrisch. Motor-bestuurder spanningsvallen en variërende reluctance en inductie worden gemodelleerd met een realistische traagheidsmoment ten opzichte houdkoppel. De motor rotorwikkeling een gemodelleerde weerstand van $R = 6$ Ohm en een zelfinductie $L = 6$ milli-Henries waarbij elektrische tijdconstante van 1,0 msec ontstaat.

dit apparaat is voorzien van een gele 'STEP' activiteit LED en een 'sync' LED (groen voor gesynchroniseerde of rood wanneer uitgeschakeld door één of meer stappen).

Gelijkstroommotor ('MOTOR')

Deze 'I/O' apparaat emuleert een 6-volt voeding 100: 1 geschakelde gelijkstroommotor met een geïntegreerde bestuurder-controller gedreven door een pulsbreedtemodulatiesignaal (op zijn **pwm** ingang) en een richtingsbesturingssignaal (op zijn **dir** invoer). De motor heeft ook een wiel-encoder-uitgang die bestuurt is **Enc** output pin. Je kunt gebruiken `'analogWrite()'` naar besturen de **pwm** pin met een 490 Hz (op pins 3,9,10,11) of 980 Hz (op pins 5,6) PWM golfvorm van duty cycle tussen 0,0 en 1,0 (`'analogWrite()'` waarden 0 tot 255). Als alternatief, 'MOTOR' zal *reageer* ook naar je eigen land `'digitalWrite()'` "bit-geneukt" besturen-signalen.



De motor is zowel mechanisch als elektrisch nauwkeurig gemodelleerd. Rekening houdend met (lage) MOS-motor-driver transistor spanningsvallen en realistisch onbelast tandwielkoppel geeft een volle snelheid van bijna 3 omwentelingen per seconde en een stilstandkoppel van iets minder dan 10 kg-cm (optredend bij een constante PWM-duty-cycle van 1.0), met een mechanische tijdconstante van ongeveer 40 milliseconden (die wordt verhoogd door de traagheid van een gespecificeerde belasting). Drie (optionele) parameterwaarden kunnen worden gespecificeerd in een gebruikersbestand 'IODEvs.txt' - 'F' of 'B' (voor vrijloop-uitloop- of remmodus wanneer Pwm LAAG is), en vervolgens een constant belastingskoppel als percentage van afslaan koppel, dan belast het traagheidsmoment als een geheel veelvoud van de intrinsieke traagheid van de motorrotor (deze waarden zijn standaard ingesteld op 'F', 0 en 1 als er geen zijn gespecificeerd). Bovendien is er een ingebouwd constant tegengesteld versnellingsbakkoppel van 10% van het afslagkoppel (wat bijdraagt aan het belastingskoppel).

De motorrotorwikkeling heeft een gemodelleerde weerstand van $R = 2 \text{ ohm}$, en een inductie van $L = 300 \text{ micro-Henries}$ die een elektrische tijdconstante van 150 microseconden creëert. Vanwege de realistische modellering zult u die zeer smalle PWM-pulsen opmerken niet krijgen de motor moet draaien - zowel vanwege de eindige stroomstijgtijd als de significante uitschakeltijd na elke smalle puls. Deze combineren om onvoldoende rotormomentum te veroorzaken om de veerachtige achterspoeling van de versnellingsbak onder statische wrijving te overwinnen. Het gevolg is bij gebruik `'analogWrite()'` een duty cycle van minder dan ongeveer 0,125 zal de motor niet doen bewegen - dit komt overeen met wat wordt waargenomen bij het besturen van een echte tandwielmotor van een 'Uno of 'Mega" met, uiteraard, een geschikte (en vereist) motor bestuurder-module tussen de motor en de 'Uno of 'Mega"!

De geëmuleerde motor-encoder is een op een as gemonteerde optische onderbrekingssensor die een duty cycle golfvorm van 50% produceert met 8 volledige hoog-lage perioden per wielomwenteling (dus uw programma kan veranderingen in wielrotatie detecteren met een resolutie van 22,5 graden).

Servomotor ('SERVO')

Deze 'I/O' apparaat emuleert een positiegestuurde PWM-gedreven 6 volt DC-servomotor. Mechanische en elektrische modelleringsparameters voor servobediening komen sterk overeen met die van een standaard HS-422-servo. De servo heeft een maximale rotatiesnelheid van ongeveer 60 graden in 180 milliseconden . Als het linksonder checkbox is aangevinkt, de servo wordt een **continue rotatie** servo met dezelfde maximumsnelheid, maar nu stelt de PWM-pulsbreedte de **snelheid** in plaats van de hoek



Uw programma moet een `'#include <Servo.h>'` regel voordat u uw `'Servo'` Bijvoorbeeld (s) *als u ervoor kiest om de bibliotheekfunctionaliteit van de 'Servo.h' te gebruiken* , bijv `'Servo.write()'` , `'Servo.writeMicroseconds()'` Als alternatief reageert 'SERVO' ook op `'digitalWrite()'` "Bit-banged" signalen. Vanwege de interne implementatie van UnoArduSim, bent u beperkt tot 6 'SERVO' apparaten.

Piëzo Luidspreker ('PIEZO')



Met deze apparaat kunt u "luisteren" naar signalen op elke gekozen 'Uno of 'Mega" pin, en het kan een nuttige aanvulling zijn op LED's voor het debuggen van uw programma-bediening. Je kunt ook een beetje plezier hebben bij het spelen van beltonen ' `tone()` ' en ' `delay()` ' oproepen (hoewel er geen filtering van de rechthoekige golfvorm is, dus u zult geen "zuivere" noten horen).

Je kunt ook luisteren naar een aangesloten 'PULSER' of 'FUNCGEN' apparaat door een 'PIEZO' aan de pin te koppelen die apparaat bestuurt is.

Schuifweerstand ('R=1K')



Met deze apparaat kan de gebruiker een pull-up-weerstand van 1 k-Ohm aansluiten op een 'Uno of 'Mega" pin of + 5V, of een pull-down-weerstand van 1 k-Ohm op aarde. Hiermee kunt u elektrische belastingen simuleren die zijn toegevoegd aan een echte apparaat-hardware. Door met de linkermuisknop op de schuifschakelaar te klikken **lichaam** u kunt uw gewenste pull-up of pull-down selectie schakelen. Als je een of meer van deze apparaten gebruikt, kun je een enkele (of meerdere) bit-code instellen om je programma te lezen en erop te reageren.

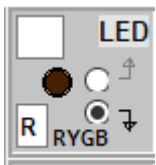
Drukknop ('PUSH')



Deze 'I/O' apparaat emuleert een normaal open **kortstondig OF vergrendelend** eenpolige SPST-drukknop met een 10 k-ohm pull-up (of pull-down) weerstand. Als voor de apparaat een overgangselectie met stijgende flanken wordt gekozen, worden de contacten van de drukknop bedraad tussen de apparaat pin en + 5V, met een 10 k-Ohm pull-down naar aarde. Als er voor de apparaat een dalende flankovergang wordt gekozen, worden de contacten van de drukknop bedraad tussen de apparaat pin en aarde, met een 10 k-Ohm pull-up naar + 5V.

Door met de linkermuisknop op de knop te klikken of door op een willekeurige toets te drukken, sluit u het contact met de drukknop. In **kortstondig** modus, deze blijft gesloten zolang u de muisknop of toets ingedrukt houdt en ingedrukt houdt **klink** modus (ingeschakeld door op de 'latch' te klikken knop) blijft het gesloten (en een andere kleur) totdat u nogmaals op de knop drukt. Contact stuiten (voor 1 milliseconde) wordt geproduceerd elke keer dat u gebruik de **intervaltoets** om op de drukknop te drukken.

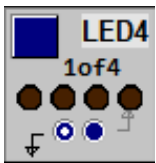
Gekleurde LED ('LED')



U kunt een LED aansluiten tussen de gekozen 'Uno of 'Mega" pin (via een ingebouwd verborgen reeks 1 k-ohm stroombeperkende weerstand) op aarde of op + 5V - dit geeft u de keuze om de LED op te laten lichten wanneer de aangesloten pin is 'HIGH' of in plaats daarvan wanneer is het 'LOW'.

De LED-kleur kan worden gekozen als rood ('R'), geel ('Y'), groen ('G') of blauw ('B') met behulp van de invoervak.

Rij 4-LED ('LED4')



U kunt deze rij van 4 gekleurde LED's aansluiten op de gekozen set 'Uno of 'Mega" pins (elk heeft een ingebouwd verborgen reeks 1 k-Ohm stroombeperkende weerstand) op aarde of op + 5V - dit geeft u de keuze om de LED's te laten branden omhoog wanneer de aangesloten pin is 'HIGH' of in plaats daarvan wanneer is het 'LOW'.

De '1of4' pin invoervak accepteert een enkel pin-nummer dat wordt verstaan als **de eerste van de vier opeenvolgende** 'Uno of 'Mega" pins die verbinding maakt met de 4 LED's.

De LED-kleur ('R', 'Y', 'G' of 'B') is een **verborgen optie** dat kan zo zijn **alleen worden gekozen door bewerken van de IODevices.txt bestand** (welke je kunt maken met **Opslaan** van de **Configureren | I/O' Apparaten** dialoog venster).

7-segment LED Cijfer ('7SEG')



U kunt dit 7-segment Cijfer LED-display verbinden met een gekozen set van ***vier opeenvolgende 'Uno of 'Mega' pins die de hexadecimaal-code geven*** voor de gewenste weergegeven cijfer, ('0' tot en met 'F') en zet deze cijfer aan of uit met behulp van de CS * pin (actief-LOW voor AAN).

Deze apparaat bevat een ingebouwd-decoder die de ***active-HIGH*** niveaus op de vier opeenvolgende '1of4' pins om te bepalen welke gevraagde hexadecimaal cijfer moet worden weergegeven. Het niveau op het laagste pin-nummer (het getal dat wordt weergegeven in de '1of4' invoervak) vertegenwoordigt het minst significante bit van de 4-bit hexadecimaal-code.

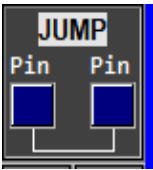
De kleur van de LED-segmenten ('R', 'Y', 'G' of 'B') is een ***verborgen optie*** dat kan zo zijn ***alleen worden gekozen door bewerken van de IODevices.txt bestand*** je kunt maken met ***Opslaan*** van de ***Configureren | 'I/O' Apparaten*** dialoog venster.

Analoge Schuifweerstand

Een schuifgestuurde 0-5V potentiometer kan op elke gekozen 'Uno of 'Mega' pin worden aangesloten om een statisch (of langzaam veranderend) analoge-spanningsniveau te produceren dat zou worden gelezen door 'analogRead()' als een waarde van 0 tot 1023. Gebruik de muis om de analoge-schuif te slepen of klik om te springen.



Pins Verbindingsdraad ('JUMP')



Je kunt er twee verbinden 'Uno of 'Mega' pins samen met deze apparaat (als er een elektrische conflict wordt gedetecteerd wanneer u het tweede pin-nummer invult, wordt de gekozen verbinding niet toegestaan en wordt de pin losgekoppeld).

Deze jumper apparaat heeft een beperkte bruikbaarheid en is vooral handig in combinatie met interrupts voor programma testen, experimenteren en leerdoelen. ***Vanaf UnoArduSim V2.4 kan het zijn dat het gebruik van een 'PROGIO' apparaat in plaats daarvan meer flexibiliteit biedt dan de onderstaande interrupt-gedreven-methoden.***

Drie mogelijke toepassingen voor deze apparaat zijn als volgt:

- 1) Jij kan ***maak een digitaal-invoer voor het testen van uw programma*** dat heeft meer complexe timing dan kan worden geproduceerd met behulp van een van de set geleverde set standaard 'I/O' apparaten, als volgt:

Definieer een interrupt functie (laten we het noemen 'myIntr') en do 'attachInterrupt(0, myIntr, RISING)' in je 'setup()' . Verbind a ***Pulser*** apparaat tot pin2 - nu 'myIntr()' zal uitvoeren zijn elke keer een ***Pulser*** stijgende flank treedt op. Jouw 'myIntr()' functie kan een algoritme zijn dat u geprogrammeerd hebt (met behulp van de globale teller variabelen, en misschien zelfs 'random()') om een golfvorm van uw eigen ontwerp te produceren op elke beschikbare 'OUTPUT' pin (laten we zeggen dat dit pin 9 is). Nu ***SPRINGEN*** pin 9 naar uw gewenste 'Uno of 'Mega' 'INPUT'pin om die gegenereerde digitaal golfvorm op die ingang pin toe te passen (om de reactie van uw programma op die bepaalde golfvorm te testen). . U kunt een reeks pulsen of seriële tekens genereren, of gewoon randovergangen, alle willekeurige complexiteit en variërende intervallen. Houd er rekening mee dat als je hoofd programma belt 'micros()' (of noemt elke functie die ervan afhankelijk is), zijn 'return' waarde ***zal worden verhoogd*** tegen de tijd die je in je hebt doorgebracht 'myIntr()' functie elke keer dat de interrupts vuurt. U kunt een snelle burst van nauwkeurig getimed randen produceren met behulp van oproepen naar 'delayMicroseconds()' van binnen 'myIntr()' (misschien om een geheel te genereren ***byte*** van een overdracht met hoge overdrachtsnelheid), of genereer eenvoudigweg één overgang per interrupt (misschien om te genereren ***een beetje*** van een lage overdrachtsnelheid-overdracht) met de ***Pulser*** apparaat ***'Period'*** gekozen volgens uw timing-behoefte (herinner dat ***Pulser*** beperkt het minimum ***'Period'*** tot 50 microseconden).

- 2) Jij kan ***experimenteer met lusbacks van het subsysteem:***

Koppel bijvoorbeeld uw 'SERIAL' 'I/O' apparaat TX los '00' pin (bewerk het in een blanco) en vervolgens

SPRINGEN 'Uno of 'Mega' pin '01' terug naar pin '00' om een hardware loop-back van de ATmega te emuleren 'Serial' subsysteem. Nu in je test programma, binnen 'setup()' doe a *single* 'Serial.print()' van een woord of karakter, en in je 'loop()' echo terug ontvangen tekens (wanneer 'Serial.available()') door een te doen 'Serial.read()' gevolgd door een 'Serial.write()' en kijk dan wat er gebeurt. Je zou kunnen waarnemen dat een soortgelijk 'SoftwareSerial' loopback *zal mislukken* (zoals het in het echte leven zou zijn - de software kan niet tegelijkertijd twee dingen doen).

Je kunt het ook uitproberen **SPI** terugloop door a te gebruiken **SPRINGEN** om pin 11 (MOSI) weer aan te sluiten op pin 12 (MISO).

3) Jij kan *tel het aantal en / of meet de spatiëring van specifieke niveauovergangen op elke 'Uno of 'Mega' output pin X* die optreden als gevolg van een complex Arduino-instructie of bibliotheek functie (als voorbeelden: 'analogWrite()' of 'OneWire::reset()', of 'Servo::write()'), als volgt:

SPRINGEN pin **X** onderbreken pin **2** en in je 'myIntr()' gebruik een 'digitalRead()' en een 'micros()' noemen, en vergelijk met opgeslagen niveaus en tijden (van eerdere interrupts). U kunt de randgevoeligheid voor de volgende onderbreking zo nodig wijzigen, gebruik makend van 'detachInterrupt()' en 'attachInterrupt()' van *binnen* jouw 'myIntr()'. Merk op dat u pin niet kunt volgen overgangen die te dicht bij elkaar plaatsvinden (dichterbij dan de totale uitvoering-tijd van jouw 'myIntr()' functie), zoals degene die gebeuren met I2C- of SPI-overdrachten, of met hoge overdrachtsnelheid 'Serial' transfers (ook al zou uw interrupt functie de inter-edge timing van deze hardware geproduceerde transfers niet storen). Merk ook op dat software-gemedieerde overdrachten (zoals 'OneWire::write()' en 'SoftwareSerial::write()') zijn opzettelijk beschermd tegen onderbreking (door hun bibliotheekcode tijdelijk alle interrupts uit te schakelen, om timing-verstoringen te voorkomen), zodat u niet kunt meten binnen degenen die deze methode gebruiken.

Hoewel u in plaats daarvan dezelfde afstandsmaten op de randen kunt maken *visueel* in een **Digitaal golfvormen** window, als je geïnteresseerd bent in de minimale of maximale spatiëring over een groot aantal overgangen, of bij het tellen van overgangen, doe dit dan met deze 'myIntr()' -plus- **SPRINGEN** techniek is handiger. En jij kan meet bijvoorbeeld variaties in de tussenruimte van door programma geproduceerde overgangen (vanwege het effect van uw software met verschillende uitvoering-paden van verschillende uitvoering keer), om een soort te doen van programma "profilering".

Menu's

Bestand:

<u>Laden INO of PDE Prog (ctrl-L)</u> 	Hiermee kan de gebruiker een programma bestand kiezen met de geselecteerde extensie. De programma krijgt onmiddellijk een Analyseren
<u>Editeren/Bekijken (Ctrl-E)</u>	Opent de geladen programma voor weergave / bewerking.
<u>Opslaan</u> 	Opslaan de bewerkte programma-inhoud terug naar de originele programma bestand.
<u>Opslaan Als</u>	Opslaan de bewerkte programma-inhoud onder een andere bestand-naam.
<u>Volgende ('#include')</u> 	Voert het CodePaneel om de volgende weer te geven '#include' bestand
<u>Voorgaand</u> 	Retourneert de CodePaneel weergegeven naar de vorige bestand
<u>Exit</u>	Sluit UnoArduSim af nadat de gebruiker eraan is herinnerd gewijzigde bestand (s) op te slaan.

Vinden:

<u>Klimmen Stapel Oproepen</u> 	Spring naar de vorige bellerfunctie in de Stapel Oproepen - het VariabelenPaneel past zich aan die functie aan
<u>Daal Stapel Oproepen</u> 	Spring naar de volgende opgeroepen functie in de Stapel Oproepen - het VariabelenPaneel past zich aan die functie aan
<u>Stel Zoeken-tekst in (Ctrl-F)</u> 	Activeer de Tool-Bar Vinden-invoervak om de tekst te definiëren waarnaar moet worden gezocht (en voegt het eerste woord uit de momenteel gemarkeerde regel in de CodePaneel of VariabelenPaneel als een van die heeft de focus).
<u>Vinden Volgende tekst</u> 	Spring naar de volgende tekst die voorkomt in de CodePaneel (als het de actieve focus heeft), of naar de volgende tekstvorm in de VariabelenPaneel (in plaats daarvan heeft het de actieve focus).
<u>Vinden Vorige tekst</u> 	Spring naar de vorige tekstvorm in de CodePaneel (als deze de actieve focus heeft), of naar de vorige tekstoccurrence in de VariabelenPaneel (in plaats daarvan heeft het de actieve focus).

Uitvoeren:

<u>Stap-In (F4)</u>		Stappen uitvoering doorsturen met één instructie, of <i>in een geroepen functie</i> .
<u>Stap-Over (F5)</u>		Stappen uitvoering doorsturen met één instructie, of <i>door een complete functie-oproep</i> .
<u>Stap-Uit (F6)</u>		Voert uitvoering vooruit door <i>net genoeg om de huidige functie te verlaten</i> .
<u>Uitvoeren-Tot (F7)</u>		Voert de programma uit, <i>stoppen bij de gewenste programma-lijn</i> - u moet eerst naar markeer een gewenste programma-lijn klikken voordat u Uitvoeren-Tot gebruikt.
<u>Uitvoeren-Totdat (F8)</u>		Voert de programma uit totdat er een schrijfactie plaatsvindt naar de variabele met de huidige markeer in de VariabelenPaneel (klik op een om de eerste markeer vast te stellen).
<u>Uitvoeren (F9)</u>		Voert de programma uit.
<u>Halt (F10)</u>		Halts programma uitvoering (<i>en bevriest de tijd</i>).
<u>Reset</u>		Reset de programma (alle waarden-variabelen worden gereset naar waarde 0 en alle pointer-variabelen worden teruggesteld naar 0x0000).
<u>Animeren</u>		Staps automatisch opeenvolgende programma-lijnen <i>met toegevoegde kunstmatige vertraging</i> en markeren van de huidige coderegel. Realtime bediening en geluiden gaan verloren.
<u>Trage Weergave</u>		Vertraagt de tijd met een factor 10.

Opties:

<u>Stap-Over-constructors/Operators</u>	Vlieg dwars door constructeurs, destructors en operator functies overbelasting tijdens een stap (dwz het zal niet stoppen in deze functies).
<u>Register-Allocation</u>	Wijs functie-inwoners toe aan vrij ATmega-registers in plaats van aan de stapel (genereert enigszins verminderd RAM-gebruik).
<u>Fout bij niet-geïnitieerd</u>	Markeer als een Analyseren-fout overal waar uw programma een variabele probeert te gebruiken zonder eerst de waarde (of ten minste één waarde in een reeks) te hebben geïnitieerd.
<u>Toegevoegd 'loop()' -Vertraging</u>	Voegt elke keer 1000 microseconden vertraging toe 'loop()' wordt gebeld (voor het geval er geen andere programma-aanroepen zijn 'delay()' overal) - handig om te proberen te vermijden dat je te ver achter raakt in real-time.
<u>Nested Interrupts toestaan</u>	Opnieuw inschakelen toestaan met 'interrupts()' van binnenuit een gebruikersinterrupt-service-routine.

Configureren:

<u>'I/O' Apparaten</u>	Hiermee wordt een dialoogvenster geopend waarin de gebruiker het type of de typen en nummers van de gewenste 'I/O' apparaten kan kiezen. Vanuit dit dialoogvenster kunt u ook Opslaan 'I/O' apparaten naar een tekst bestand en / of Laden 'I/O' apparaten van een eerder opgeslagen (of bewerkte) tekst bestand (inclusief alle pin-verbindingen en aanklikbare instellingen en ingetypte waarden
<u>Voorkeuren</u>	Opent een dialoogvenster om de gebruiker toe te staan voorkeuren in te stellen, inclusief automatische inspringing van bron programma-regels, waardoor Expert-syntaxis, lettertype lettertype, een grotere tekengrootte, reeks-grenzen worden toegestaan, logische operatorzoekwoorden worden toegestaan, met programma downloaden , keuze uit 'Uno of 'Mega" printplaat-versie en TWI-buffellengte (voor I2C apparaten).



VarRefresh:

<u>Laat Auto (-) Samentrekken toe</u>	Toestaan dat UnoArduSim to samentrekken uitgebreid reeksen / objecten toont als deze achterloopt in real-time.
<u>Minimaal</u>	Vernieuw alleen de VariabelenPaneel 4 keer per seconde weergeven.
<u>Markeer Veranderingen</u>	Markeer veranderde variabele-waarden tijdens het hardlopen (kan vertraging veroorzaken).

Windows:

<u>'Serial' Monitor</u>	Sluit een seriële I / O apparaat aan op pins 0 en 1 (indien geen) en trek een grotere omhoog 'Serial' controleer TX / RX-tekst window.
<u>Herstel alles</u>	Herstel alle geminimaliseerde onderliggende windows.
<u>Pins Golfvormen Digitale</u>	Herstel een geminimaliseerde Pins Golfvormen Digitale window.
<u>Pin Golfvorm Analoge</u>	Herstel een geminimaliseerde Pin Golfvorm Analoge window.

Hulp:

<u>"Wat is dit" -modus (Ctrl >)</u> 	Ga naar "Wat is dit" Help-modus - klik vervolgens op een Menu- of Werkbalkitem of op de Statusbalktekst en er verschijnt een pop-up met een uitleg in het Engels die kan worden gekopieerd / geplakt in uw vertaalapp naar keuze.
<u>Quick Hulp Bestand (Ctrl ?)</u> 	Opent de UnoArduSim_QuickHelp PDF bestand.
<u>Volledige Hulp Bestand</u>	Opent de UnoArduSim_FullHelp PDF bestand.
<u>Bug-oplossingen</u>	Bekijk belangrijke bug-fixes sinds de vorige release.
<u>Wijzigen / Verbeteringen</u>	Bekijk belangrijke wijzigingen en verbeteringen sinds de vorige release.
<u>Wat betreft</u>	Geeft de versie weer, copyright.

'Uno of 'Mega" Printplaat en 'I/O' Apparaten

De 'Uno of 'Mega" en de bijgevoegde 'I/O' apparaten zijn allemaal nauwkeurig elektrisch gemodelleerd en u kunt thuis een goed idee krijgen van hoe uw programmas zich zal gedragen met de feitelijke hardware en alle elektrische pin conflicten zal worden gemarkeerd.

Timing

UnoArduSim uitvoeren snel genoeg op een pc of tablet die het kan (*in de meeste gevallen*) model programma-acties in realtime, **maar alleen als uw programma deze bevat** op zijn minst wat klein `'delay()'` oproepen of andere oproepen die het natuurlijk synchroniseren tot in realtime (zie hieronder).

Om dit te bereiken, maakt UnoArduSim gebruik van een Windows callback-timer functie, waarmee het een nauwkeurige track van real-time kan bijhouden. De uitvoering van een aantal programma-instructies wordt gesimuleerd gedurende één timerplak en instructies die langer duren uitvoering (zoals oproepen naar `'delay()'`) kan het nodig zijn om meerdere timerplakken te gebruiken. Elke iteratie van de terugteltimer functie corrigeert de systeemtijd met behulp van de systeemhardwareklok zodat programma uitvoering constant wordt aangepast om in lock-step te blijven met real-time. *De enige keren uitvoering-snelheid moet achterblijven in real-time* is wanneer de gebruiker strakke lussen heeft gemaakt **zonder extra vertraging** of 'I/O' apparaten zijn geconfigureerd voor gebruik met zeer hoge 'I/O' apparaat-frequenties (en / of overdrachtsnelheid) die een buitensporig aantal pin-niveau-wijzigingsgebeurtenissen en bijbehorende verwerkingsoverbelasting zouden genereren. UnoArduSim omzeilt deze overbelasting door een aantal timer-intervallen over te slaan om te compenseren, en dit vertraagt dan de voortgang van programma naar **onder realtime** .

Bovendien wordt programmas met grote reeksen weergegeven of opnieuw met krappe lussen **zonder extra vertraging** kan een hoge functie-oproepfrequentie veroorzaken en een hoge genereren **VariabelenPaneel** laat de update laden waardoor deze in real-time achterloopt - UnoArduSim verlaagt automatisch de variabele-verseringsfrequentie om bij te blijven, maar wanneer nog meer reductie nodig is, kiest u **Minimaal** , van de **VarRefresh** menu om slechts vier vernieuwingen per seconde op te geven.

Nauwkeurig modelleren van de sub-milliseconde uitvoering-tijd voor elke programma-instructie of -bewerking **is niet gedaan** - slechts zeer ruwe schattingen voor de meeste zijn aangenomen voor simulatiedoeleinden. Echter, de timing van `'delay()'` , en `'delayMicroseconds()'` functies en functies `'millis()'` en `'micros()'` zijn allemaal perfect nauwkeurig, en **zolang u tenminste één van de vertraging functies gebruikt** in een lus ergens in je programma, **of** je gebruikt een functie die zich op natuurlijke wijze verbindt met real-time bediening (zoals `'print()'` die is gekoppeld aan de gekozen overdrachtsnelheid), dan de gesimuleerde uitvoering van uw programma zal zeer dicht bij real-time zijn (opnieuw, met blanco overdreven hoogfrequente pin-niveau veranderingsgebeurtenissen of overdreven door de gebruiker toegestane Variabelen-updates die het zouden kunnen vertragen).

Om het effect van individuele programma-instructies te zien w *ze rennen* , kan het wenselijk zijn om dingen te kunnen vertragen. Een tijdvertragsfactor van 10 kan door de gebruiker onder het menu worden ingesteld **Uitvoeren** .

'I/O' Apparaat Timing

Deze virtuele apparaten ontvangen real-time signalering van veranderingen die optreden op hun ingang pins en produceren overeenkomstige outputs op hun output pins die dan kunnen worden gedetecteerd door de 'Uno of 'Mega" - ze zijn daarom inherent gesynchroniseerd met programma uitvoering. Interne 'I/O' apparaat timing wordt ingesteld door de gebruiker (bijvoorbeeld via overdrachtsnelheid-selectie of klokfrequentie) en simulatorgebeurtenissen worden ingesteld om de realtime interne werking bij te houden.

Sounds

Elke 'PIEZO' apparaat produceert geluid dat overeenkomt met de veranderingen in het elektrisch niveau die optreden op de aangesloten pin, ongeacht de bron van dergelijke wijzigingen. Om de geluiden gesynchroniseerd te houden met programma uitvoering, start en stopt UnoArduSim het afspelen van een bijbehorende geluidsbuffer terwijl uitvoering wordt gestart / gestopt.

Vanaf V2.0 is het geluid nu aangepast om de Qt-audio-API te gebruiken - helaas de QAudioOutput `'class'` ondersteunt geen lusvorming van de geluidsbuffer om te voorkomen dat er onvoldoende geluidsfragmenten zijn (zoals

kan gebeuren tijdens operationele vertragingen bij OS OS). Daarom wordt het geluid nu gedempt volgens de volgende regel om het grootste deel van irritante klikgeluiden en het verbreken van geluid tijdens OS-vertragingen te voorkomen:

Geluid wordt gedempt zolang UnoArduSim niet de "actieve" window is (behalve wanneer een nieuw kind window zojuist is gemaakt en geactiveerd), **en zelfs** wanneer UnoArduSim de "actieve" hoofd window is maar de muisaanwijzer is **buiten** van het belangrijkste klantengebied van window.

Merk op dat dit impliceert dat het geluid tijdelijk zal zijn gedempt als koning terwijl de muisaanwijzer zweeft **over een kind window**, en wordt gedempt **als op dat kind window is geklikt om het te activeren** (totdat de hoofd UnoArduSim window opnieuw wordt aangeklikt om deze opnieuw te activeren) .

Geluid kan altijd worden gedempt door ergens in het clientgebied van de UnoArduSim main window te klikken.

Vanwege buffering, sound heeft een realtime vertraging van maximaal 250 milliseconden ten opzichte van de overeenkomstige event-tijd op de pin van de aangesloten 'PIEZO'.

Beperkingen en niet-ondersteunde Elementen

Inclusief Bestanden

A '<>' - tussen haakjes '#include' van '<Servo.h>', '<Wire.h>', '<OneWire.h>', '<SoftwareSerial.h>', '<SPI.h>', '<EEPROM.h>' en '<SD.h>' is ondersteund, maar deze worden alleen nagebootst - er wordt niet naar de daadwerkelijke bestanden gezocht; in plaats daarvan is hun functionaliteit direct "ingebouwd" in UnoArduSim en gelden ze voor de vaste, ondersteunde Arduino-versie.

Geciteerd '#include' (bijvoorbeeld van "supp.ino", "Myutil.cpp" of "Mylib.h") wordt ondersteund, maar al dergelijke bestanden moet **wonen in de dezelfde map als de ouder programma bestand** dat bevat hun '#include' (er wordt niet gezocht in andere mappen). De '#include' functie kan nuttig zijn voor het minimaliseren van de hoeveelheid programma-code die wordt weergegeven in de **CodePaneel** op elk willekeurig moment. Header bestanden met '#include' (dwz die met een ".h" extensie) zal er bovendien voor zorgen dat de simulator probeert dezelfde bestand met dezelfde naam te gebruiken als een "C++" extensie (als deze ook voorkomt in de directory van de bovenliggende programma).

Dynamische geheugentoe wijzingen en RAM

operators 'new' en 'delete' worden ondersteund, net als native Arduino 'String' objecten, **maar geen rechtstreekse oproepen naar** 'malloc()', 'realloc()' en 'free()' waar deze op vertrouwen.

Overmatig RAM-gebruik voor variabele-aangiften wordt gemarkeerd op Analyseren-tijd en RAM-geheugenoverloop wordt gemarkeerd tijdens programma uitvoering. Een item op menu **Opties** stelt u in staat om de normale ATmega-registrallocatie te emuleren zoals door de AVR compiler zou worden gedaan, of om een alternatief compilatieschema te modelleren dat alleen de stapel gebruikt (als een veiligheidsoptie voor het geval een bug opduikt in mijn modellering van de registratieverrekening). Als u een aanwijzer zou gebruiken om de inhoud van de stapel te bekijken, zou dit nauwkeurig moeten weergeven wat er zou verschijnen in een daadwerkelijke hardware-implementatie.

'Flash' Geheugentoe wijzingen

'Flash' geheugen 'byte', 'int' en 'float' variabelen / reeksen en hun bijbehorende leestoegang functies worden ondersteund. Ieder 'F()' functie-oproep ('Flash'-macro) van elke letterlijke reeks is ondersteund, maar de enige ondersteunde 'Flash'-geheugenreeks directe toegang functies zijn 'strcpy_P()' en 'memcpy_P()', dus om andere functies te gebruiken, moet je eerst de 'Flash'-string naar een normaal RAM-geheugen kopiëren 'String' variabele, en werk dan met dat RAM 'String'. Wanneer u de 'PROGMEM' variabele-wijzigingszoekwoord, dit moet verschijnen **voor** de variabele-naam en die variabele **moet ook worden aangegeven** zoals 'const'.

'String' Variabelen

De inheemse 'String' bibliotheek wordt bijna volledig ondersteund met een paar zeer (en kleine) uitzonderingen.

De 'String' ondersteunde operators zijn +, + =, <, <=, >, > =, ==, !=, en []. Let daar op: 'concat()' kost een **single** argument dat is de 'String', of 'char' of 'int' worden toegevoegd aan het origineel 'String' object, **niet** twee argumenten zoals ten onrechte vermeld op de Arduino Reference-webpagina's).

Arduino-bibliotheken

Enkel en alleen 'Servo.h', 'SoftwareSerial.h', 'SPI.h', 'Wire.h', 'OneWire.h', 'Stepper.h', 'SD.h', 'TFT.h' en 'EEPROM.h' voor de **Arduino V1.8.8** laat dit moment worden ondersteund in UnoArduSim. V2.6 introduceert een mechanisme voor 3rd partij bibliotheek ondersteuning via bestanden in de 'include_3rdParty' map die kan worden gevonden in het UnoArduSim install directory. Proberen '#include' de ".cpp" en ".h" bestanden andere nog-niet-ondersteunde bibliotheken **werkt niet** want zij zullen low-level montage-instructies en niet-ondersteunde richtlijnen en niet-herkende bestanden bevatten!

Pointers

Verwijzingen naar eenvoudige typen, reeksen of objecten worden allemaal ondersteund. Een aanwijzer kan worden gelijkgesteld aan een reeks van hetzelfde type (bijv 'iptr = intarray'), maar dan zou er zijn *geen volgende reeksen-grenzencontrole* op een expressie zoals 'iptr[index]'.

Functies kan pointers retourneren, of 'const' wijzers, maar elk volgend niveau van 'const' op de geretourneerde aanwijzer wordt genegeerd.

Er bestaat **geen steun** voor functie-oproepen die worden gemaakt via **door de gebruiker gedeclareerde functie-pointers**.

'class' en 'struct' Objecten

Hoewel poly-morphism en inheritance (tot elke diepte) wordt ondersteund, 'class' of 'struct' kan alleen worden gedefinieerd om maximaal te hebben **een** baseren 'class' (d.w.z **meerdere**-overerving wordt niet ondersteund). Base-'class' constructor initialisatie calls (via colon notatie) in constructor declaratieregels worden ondersteund, maar **niet** lidinitialisaties met dezelfde dubbele notatie. Dit betekent dat objecten die bevatten 'const' niet-'static' variabelen of referentiekatype variabelen worden niet ondersteund (dit zijn alleen mogelijk met opgegeven constructielidinitialisaties)

Kopieertaakoperatoroverbelastingen worden ondersteund, samen met verplaats-constructors en verplaatsingstoewijzingen, maar de door de gebruiker gedefinieerde object-conversie ("overzetting") functies worden niet ondersteund.

Bereik

Er is geen ondersteuning voor de 'using' zoekwoord of voor 'namespace', of voor 'file' bereik. Alle niet-lokale aangiften worden per implementatie verondersteld globaal te zijn.

Ieder 'typedef', 'struct' of 'class' definitie (dat wil zeggen dat dit kan worden gebruikt voor toekomstige aangiften), moet worden gemaakt **globaal** bereik (**lokaal** definities van dergelijke items in een functie worden niet ondersteund).

qualifiers 'unsigned', 'const', 'volatile', 'static'

De 'unsigned' voorvoegsel werkt in alle normale wettelijke contexten. De 'const' sleutelwoord, indien gebruikt, moet **voorafgaan** de variabele-naam of functie-naam of 'typedef' naam die wordt gedeclareerd. Als u deze na de naam plaatst, wordt er een Analyseren-fout veroorzaakt. Voor functie-aangiften, alleen aanwijssapparaat functies kan hebben 'const' verschijnen in hun verklaring.

Allesmaal UnoArduSim variabelen zijn 'volatile' door implementatie, dus de 'volatile' sleutelwoord wordt

eenvoudig genegeerd in alle variabele-verklaringen. Functies mogen niet worden gedeclareerd `'volatile'` , noch zijn functie-call-argumenten.

De `'static'` sleutelwoord is toegestaan voor normale variabelen en voor object leden en lid-functies, maar is expliciet niet toegestaan voor object-instanties zelf (`'class'` / `'struct'`), voor niet-leden functies en voor alle functie-argumenten.

Compiler-richtlijnen

`'#include'` en regelmatig `'#define'` worden beide ondersteund, maar **niet functie-macro** `'#define'` . Vanaf V2.9 wordt voorwaardelijke compilatie ondersteund via het `'#ifdef'` , `'#ifndef'` , `'#else'` , en `'#endif'` richtlijnen (**maar niet** `'#if'` of `'#elif'`) - **nesten van deze richtlijnen is ook niet ondersteund** . De richtlijnen `'#pragma'` , `'template'` , `'#line'` , `'#error'` en voorgedefinieerde macro's (zoals `'_LINE_'` , `'_FILE_'` , `'_DATE_'` , en `'_TIME_'`) zijn ook **niet ondersteund** .

Arduino-taalelementen

Alle native Arduino-taalelementen worden ondersteund, met uitzondering van dubieus `'goto'` instructie (het enige redelijke gebruik dat ik kan bedenken zou zijn als een sprong (naar een bail-out en veilige shutdown eindeloze lus) in het geval van een foutconditie waar je programma anders niet mee om kan gaan)

C / C ++ - Taalelementen

Bitbesparende "bit-field qualifiers" voor leden in structuurdefinities zijn **niet ondersteund** .

`'union'` is **niet ondersteund** .

De oddball "comma-operator" is **niet ondersteund** (u kunt dus niet meerdere expressies gescheiden door komma's uitvoeren als u slechts één uitdrukking normaal verwacht, bijvoorbeeld in `'while()'` en `'for(; ;)'` constructen).

Functie-sjablonen

Door de gebruiker gedefinieerde functies die het sleutelwoord "sjabloon" gebruikt om het toe te staan argumenten van "algemeen" type te accepteren **niet ondersteund** .

Realtime Emulatie

Zoals hierboven vermeld, zijn uitvoering keer zoveel van de vele individuele mogelijke Arduino programma-instructies **niet** nauwkeurig gemodelleerd, zodat uw programma een soort van dominantie nodig heeft om met een realtime-snelheid te werken `'delay()'` instructie (minstens één keer per `'loop()'`), of een instructie die van nature is gesynchroniseerd met real-time pin-niveau wijzigingen (zoals `'pulseIn()'` , `'shiftIn()'` , `'Serial.read()'` , `'Serial.print()'` , `'Serial.flush()'` enz.).

Zien **Timing** en **Sounds** hierboven voor meer informatie over beperkingen.

Releaseopmerkingen

Bug-Oplossingen

V2.9.2 - april 2021

- 1) Het wijzigen van de '**String**'-variabele van een 'PROGIO'-programma in Editeren/Opvolgen kan mislukken.

V2.9.1 - februari 2021

- 1) Als er geen pin is ingevoerd in een 'PULSER' of 'FUNCGEN', kan een crash optreden als er wijzigingen worden aangebracht aan een instelling van keuzerondjes, of naar een invoerveld.

V2.9 - januari 2021

- 1) Pre- en post-increment / decrement-expressies erfden geen '**unsigned**' van hun operand.
- 2) Wijzigingen in de keuzerondjes 'LED' en 'LED4' werden pas van kracht bij de volgende wijziging van het pinniveau.
- 3) Na het resetten schrijft het 'LCD'-apparaat nu standaard naar het begin van het display-RAM (in plaats van 'CGRAM').
- 4) Wanneer de gebruiker het 's?'-selectievakje selecteerde om het invoeren van escapetekens in een uitgebreid 'Serial' Monitor window toe te staan, werd het 'Send'-tekstvak niet bijgewerkt tijdens de daadwerkelijke verzending van de 'Send'-tekens.

V2.8.2-September 2020

- 1) '**analogRead()**' accepteerde niet '**A5**' als een geldig analoge pin-nummer.
- 2) Sinds V2.6 gaven 'PULSER'-keuzerondjes niet altijd de geselecteerde status weer (maar werkten ze verder correct).
- 3) Sinds V2.4, flippen vanuit een stabiel '**HIGH**' pin niveau naar '**analogWrite(pin, 0)**' zorgde ervoor dat de verandering op 0-niveau werd gemist.
- 4) Sinds V2.8, uitvoering (niet-fout) watrning pop-ups slaagden er niet in om markeer de probleemcode-lne.
- 5) Onderbreken van een lopende periodiek omkerende uitgang pin met '**digitalWrite()**' of '**digitalRead()**' veroorzaakte verlies van de voorgaande PWM-salto's op die pin in de Wavafeforms-widnow.
- 6) Problemen met het verbinden van twee '**INPUT**' pins met een '**JUMP**' apparaat: 'I/O' apparaat pin-wijzigingen worden niet langer eenvoudigweg weergegeven op de doorverbonden pin (ze verschijnen nu op beide pins) en bijgevoegd '**PULSER**' of '**FUNCGEN**' periodieke signalen worden nu ook uitgebreid naar de jumperd-to pn.
- 7) Kortstondig dichtbij '**PUSH**' apparaten keerden niet terug naar hun open toestand toen de muisaanwijzer de apparaat verliet.

V2.8.1-Juni 2020

- 8) Extra lege regels verwijderd door de automatische opmaakvoorkeur zorgden ervoor dat de aanvankelijk gemarkeerde regel in Editeren/Bekijken naar beneden werd verschoven door het aantal lege regels dat erboven werd verwijderd.
- 9) '**analogRead()**' accepteerde alleen het volledige digitaal pin-nummer.
- 10) Een 'class' die functie-overbelastingen bevatte die alleen verschilden in het 'unsigned'-kenmerk van een

functie-argument, zou de verkeerde overbelasting functie kunnen hebben. Dit beïnvloedde LCD 'print(char/int/long)' waar de 'unsigned' base-10 printoverload functie in plaats daarvan zou worden aangeroepen, en dit veroorzaakte dat LCD 'printFloat()' '46' printte in plaats van de '.' decimale punt.

11) Het automatisch aanvullen van invoegen (op 'Enter') van een reeds gematchte ingebouwd werkte niet na backtracking om een typefout op de regel te corrigeren.\

12) Het 'TFT'-apparaat met een lege RS 'kan een crash bij de uitvoering veroorzaken.

V2.8.0-Juni 2020

1) Wanneer een Analyseren-waarschuwing (maar geen Analyseren-fout) optrad, kon V2.7 proberen de problematische regel in de verkeerde buffer te markeren (in plaats daarvan in de meest recente '#include'-buffer), en dat zou een stille crash veroorzaken (zelfs vóór de waarschuwingspop-up verschijnt) als het regelnummer het einde van die '#include'-buffer overschreed.

2) De ontvangen bytes in de grotere monitorvensters van 'I2CSLV', 'SPISLV', 'TFT', 'LCDI2C' en 'LCDSPI'-apparaten werden nog steeds niet allemaal correct gerapporteerd (dit had geen invloed op hun werking).

3) Variabelen van het type '**unsigned char**' werden gepromoveerd tot type '**int**' in rekenkundige uitdrukkingen, maar hun onjuist gehandhaafd '**unsigned**' status, leidend tot een fout '**unsigned**' resulterende uitdrukking.

4) Het wijzigen (of onderdrukken) van de pin van een 'LED4' of '7SEG' apparaat van een aanvankelijke geldige pin-instelling kon de bovenste 3 pins niet losmaken en zou kunnen leiden tot onverklaarbare crashes - bovendien zijn er wijzigingen aangebracht in V2.7 om de afhandeling van alle 'LED's te harmoniseren brak de 'LED4' apparaat volledig.

5) Een fout in de wijzigingen die zijn aangebracht voor versie 2.6 om 'LOW'-onderbrekingen te ondersteunen, heeft ertoe geleid dat 'FALLING'-onderbrekingen die aan pin 3 zijn gekoppeld, de detectie van onderbrekingsniveauveranderingen op pin 2 beschadigen.

6) 'SoftwareSerial' was het op een onjuiste manier uitschakelen van gebruikersinterrupts tijdens elk karakter dat het ontving.

7) Wanneer een 'PROGIO'-programma met een Parse-fout werd geladen, werd de fout gemarkeerd, maar dat programma was al vervangen door een standaard 'PROGIO'-programma in het 'PROGIO'-monitorvenster.

8) Sinds versie V2.4 waren pin-wijzigingen op Pin 1 en Pin 0 niet zichtbaar tijdens het downloaden.

9) Het herhalen van lees- of schrijfbewerkingen op een open SD bestand zorgde ervoor dat uitvoering-sequencing mislukte, wat resulteerde in een eventuele interne UnoArduSim-fout als gevolg van bereik-niveaudiepte.

10) Een poging om de 'MISO' pin op een 'SD_DRV' apparaat te wijzigen, heeft de MOSI pin beschadigd.

11) Alle eerdere versies konden dat niet waarschuwen '**analogWrite()**' op pins 9 of 10 zou 'Servo' timing voor alle actieve 'Servo' apparaten beschadigen.

V2.7.0- maart 2020

1) Wanneer de (Windows-standaard) licht OS thema werd goedgekeurd, de **CodePaneel** niet met de kleur markeren geïntroduceerd in V2.6 (in plaats daarvan alleen een grijze markeer gevolg van een systeem te onderdrukken).

2) Versie 2.6 per ongeluk brak auto-streepje-tabblad Opmaak bij de eerste '**switch()**' construeren.

3) De nieuwe stapel oproepen navigatie-functie geïntroduceerd in versie 2.6 is gebleken **onjuiste waarden** voor de lokale variabelen wanneer deze niet binnen het moment wordt uitgevoerd functie, en mislukte met geneste lid functie gesprekken.

4) '**TFT :: text()**' werkte, maar '**TFT :: print()**' functies waren dat niet (ze werden gewoon voor altijd geblokkeerd). Bovendien is '**TFT :: loadImage()**' mislukt als '**Serial.begin()**' eerder was gedaan (wat normaal is en nu vereist is).

5) Versie 2.6 introduceert een bug dat de onjuiste waarde voor heden en verleden 'RX' bytes voor 'I2CSLV',

'SPISLV', 'TFT', 'LCDI2C' and 'LDCSPI' apparaten (en hun Monitor windows) weergegeven.

6) Een wijziging in V2.4 veroorzaakt Uitvoeren | Animeren markeren veel uitvoerde code-lijnen over te slaan.

7) Sinds V2.4, de-beweren 'SS*' of 'CS*' op een 'I/O' apparaat in de instructie onmiddellijk na een '`SPI.transfer()`' zou leiden dat apparaat niet in slagen om de overgedragen gegevens byte te ontvangen. Bovendien, byte ontvangst in '`SPI_MODE1`' en '`SPI_MODE3`' was niet onder de vlag tot aan het begin van de volgende byte verzonden door de master (en de byte was volledig verloren gaan als de apparaat 'CS*' werd gedeselecteerd voor die tijd).

8) In de nieuwe '`SPI_SLV`' mode toegestaan, daar V2.4, '`bval = SPI.transfer()`' alleen terug de juiste waarde voor '`bval`' indien de byte overdracht was al klaar en wachten toen '`transfer()`' heette.

9) De 'DATA' bewerken vak op 'SPISLV' apparaten krijgt nu de standaard 0xFF als er geen bytes meer om te antwoorden met.

10) De synchronisatie LED staat onjuist voor 'PSTEPR' apparaten met meer dan 1 microstap per volledige stap.

11) Elektrische conflicten veroorzaakt door 'I/O' apparaten reactie op overgangen aan het 'SPI' klok, een 'PWM' signaal, of '`tone`' signaal, werden niet gemeld, en kan leiden tot onverklaarbare (beschadigde) data ontvangst.

12) Wanneer het interval tussen interrupts te klein (minder dan 250 microseconden), a (defect) wijziging V2.4 veranderde de timing van ingebouwd functies die oftewel systeemtimers of instructielussen, vertragingen (voorbeelden van elk te genereren '`delay()`' en '`delayMicroseconds()`'). Een verdere verandering in V2.5 veroorzaakte foutieve uitlijning van '`shiftOut()`' data en kloksignalen wanneer een interrupt er tussen bits.

13) Het accepteren van een ingebouwd functie auto-completion tekst via de Enter-toets niet in geslaagd om te strippen uit types parameter uit de tekst van het ingevoegde functie gesprek.

14) Het laden van een nieuwe (gebruiker-interrupt-gedreven) programma wanneer een eerder running programma had nog een onderbreking in afwachting van een crash tijdens het downloaden kunnen veroorzaken (door een verkeerde poging uitvoering van de nieuwe interrupt routine).

15) Object-lid automatische aanvullingen (via 'ALT'-pijl naar rechts) voor objecten binnen '`#include`' bestanden zijn nu toegankelijk zodra hun '`#include`' bestand is met succes geanalyseerd.

16) Een verklaring functie met een onbedoelde ruimte binnen een functie parameternaam veroorzaakt een onduidelijke Analyseren foutmelding.

17) Wanneer uitvoering gestopt in een module verschilt van de belangrijkste programma, de Bestand | Vorige actie niet raken ingeschakeld.

18) Single-geciteerd accoladen ('{' en '}') Nog steeds geteld (incorrect) als bereik haakjes in de Analyseren, en ook verward auto-tab-streepje opmaak ..

19) '`OneWire::readBytes(byte* buf, int count)`' moest zijn wanneer u niet direct de weergegeven '`buf`' inhoud van de VariabelenPaneel.

20) Octal-klink 'OWISLV' apparaten toonde uitgang pin niveaus die achterbleef met een klink-register schrijven.

V2.6.0- jan 2020

1) Een bug geïntroduceerd in V2.3 leidde tot een crash bij de toegevoegde **Dichtbij** knop werd gebruikt in de **Vinden / Vervangen** dialoogvenster (in plaats van de Exit titelbalk knop).

2) Als een gebruiker programma deed '`#include`' andere gebruikers bestanden de **Opslaan** knop in **Editeren/Bekijken** zou er niet in geslaagd om daadwerkelijk save een gemodificeerde bestand als er een bestaande Analyseren of Uitvoering fout gemarkeerd in een andere bestand.

3) EEN **Afbreken** na een **Opslaan** kan ook verwarrend zijn - om deze redenen de **Opslaan** en **Afbreken** knop functionaliteiten zijn veranderd (zie **Wijzigingen en verbeteringen**).

4) Onevenwichtige beugels in een '`class`' definitie kan een hang veroorzaken sinds V2.5.

5) Direct logische testen op '`long`' waarden terug '`false`' als geen van de laagste 16-bits zijn ingesteld.

6) UnoArduSim was verslapt een foutmelding wanneer een pointer variabele werd aangegeven als de lus variabele binnen de haken van een '`for()`' uitspraak.

- 7) UnoArduSim was niet toe te staan logische tests die ten opzichte van verwijzingen naar '**NULL**' of '0'.
- 8) UnoArduSim was niet toe te staan pointers met een integer variabele (alleen integer constanten was toegestaan).
- 9) Interrupts ingesteld met behulp van '**attachInterrupt(pin, name_func, LOW)**' was alleen gevoeld op een **overgang** naar '**LOW**'.
- 10) Bij gebruik van meer dan één 'I2CSLV' apparaat, kan een niet-geadresseerde slave latere busgegevens interpreteren als overeenkomend met zijn bus (of global-call 0x00) adres en zo vals ACK signaleren, het bus ACK-niveau beschadigen en een 'requestFrom()' hangen.
- 11) Passing numerieke waarde '0' (of '**NULL**') Als een functie argument om een pointer in een functie oproep wordt nu toegestaan.
- 12) De Tab-toets streepje niveau na een nesten van '**switch()**' constructies was te ondiep toen de 'auto-indent formatting' keuze van **Configureren | Voorkeuren** was gebruikt.
- 13) Aftrekken van twee compatibele pointers resulteert nu in een soort '**int**'.
- 14) UnoArduSim hadden verwacht van een door de gebruiker gedefinieerde standaard constructor voor een lid object zelfs als het niet was verklaard '**const**'.
- 15) Bij een uitvoering break, de getekende positie van een 'STEPR', 'SERVO' of 'MOTOR' motor achterblijven tot 30 milliseconden bewegingsrichting achter de werkelijke laatst berekende positie.
- 16) '**Stepper::setSpeed(0)**' was het veroorzaken van een crash als gevolg van een deling door nul.
- 17) Enkele lijn '**if()**', '**for()**' en '**else**' construeert niet langer oorzaak een te veel auto-inspringen tabs.

V2.5.0- oktober 2019

- 1) Een bug geïntroduceerd in V2.4 blut 'SD' card initialisatie (veroorzaakt een crash).
- 2) Met behulp van het subsysteem 'SPI' in de nieuwe '**SPI_SLV**' modus werkte niet goed in '**SPI_MODE1**' en '**SPI_MODE3**'.
- 3) Automatisch aanvullen pop-ups (zoals door 'ALT-right=arrow') werden gedurende functies met object parameters; de lijst met pop-ups nu ook onder meer erfelijke (basis-) klas, en auto-aanvullingen nu ook weergegeven voor '**Serial**'.
- 4) Aangezien V2.4 de retourwaarde '**SPI.transfer()**' en '**SPI.transfer16()**' onjuist was als een gebruiker interrupt routine tijdens die overdracht ontslagen.
- 5) In V2.4 werden snel periodieke golfvormen getoond met een langere duur dan de daadwerkelijke duur zichtbaar wanneer bekeken bij hogere zoom.
- 6) de constructeur '**File::File(SdFile &sdf, char *fname)**' werkte niet, dus '**File::openNextFile()**' (Die zich baseert op die constructeur) werd ook niet.
- 7) UnoArduSim werd ten onrechte waarbij een Analyseren fout op object-variabelen en object terugkerende functies, verklaard '**static**'.
- 8) Toewijzingsinstructies met een 'Servo', '**Stepper**' of '**OneWire**' object variabele op de LHS, en een object terugkerende functie of de bouwer op de RHS, veroorzaakte een interne mistellen het aantal geassocieerde objecten, wat leidt tot een uiteindelijke Botsing.
- 9) Boolean proeven op objecten van het type '**File**' werden altijd terug '**true**' zelfs als de bestand was niet open.

V2.4 - mei 2019

- 1) Uitvoeren-Totdat-watchpoints kunnen ten onrechte worden geactiveerd door te schrijven naar een aangrenzende variabele (1 byte lager in adres).
- 2) Baudsnelheidselecties kunnen worden gedetecteerd wanneer de muis zich in een 'SERIAL' of 'SFTSER' bevond apparaat baud vervolkeuzelijst (zelfs wanneer er niet op een baudrate is geklikt).
- 3) Sinds V2.2 zijn seriële ontvangstfouten opgetreden bij een baudsnelheid van 38400.

- 4) Een wijziging in V2.3 veroorzaakt '**SoftwareSerial**' om soms een onderbrekingsonderbreking ten onrechte te melden (en dus mislukken bij de eerste RX-ontvangst).
- 5) Een wijziging in V2.3 heeft ertoe geleid dat SPISLV apparaten hexwaarden verkeerd heeft geïnterpreteerd die rechtstreeks zijn ingevoerd in hun 'DATA'-bewerkingsvak.
- 6) Een wijziging in V2.3 veroorzaakte SRSLV apparaten om soms valselijk, en stil, een elektrische conflict op hun 'Dout' pin te detecteren, en zo een pin-toewijzing daar af te wijzen. Herhaalde pogingen om een pin aan 'Dout' te bevestigen, kunnen vervolgens leiden tot een eventuele crash nadat de apparaat is verwijderd.
- 7) Posing om een 'LED4' apparaat voorbij pin 16 te bevestigen, zou een crash veroorzaken.
- 8) Een bug geactiveerd door snel herhaalde oproepen naar '**analogWrite(255)**' voor '**MOTOR**' controle in de gebruiker programma veroorzaakte het resultaat '**MOTOR**' snelheden zijn onjuist (veel te traag).
- 9) Sinds V2.3 kon SRSLV apparaten niet worden toegewezen aan een 'Dout' pin vanwege een defecte elektrische conflict-detectie (en dus een toewijzing pin niet toestaan).
- 10) SPI-slaven resetten hun zender- en ontvangerlogica nu opnieuw wanneer zij '**SS***' pin gaat '**HIGH**'.
- 11) Roeping '**Wire.h**' functies '**endTransmission()**' of '**requestFrom()**' wanneer interrupts momenteel \ disabled zijn, genereert dit nu een uitvoering-fout ('**Wire.h**' moet onderbrekingen ingeschakeld zijn om te kunnen werken).
- 12) 'Ctrl-Home' en 'Ctrl-End' werken nu zoals verwacht in Editeren/Bekijken.
- 13) De '**OneWire**' bus commando **0x33** ('**ROM_READ**') werkte niet en hing de bus.

V2.3 - december 2018

- 1) Een bug geïntroduceerd in V2.2 maakte het onmogelijk om de waarde van een reeks-element erin te bewerken **Editeren/Opvolgen**.
- 2) Sinds versie 2.0 is de tekst in de '**RAM free**' Controle van de werkbalk was alleen zichtbaar als een donker Windows-OS-thema werd gebruikt.
- 3) Op **Bestand | Laden** op I / O apparaat bestand **Laden**, de bestand-filters (zoals '***.ino**' en '***.txt**') werkte niet - alle soorten bestanden werden in plaats daarvan getoond.
- 4) De "gewijzigde" status van een bestand ging verloren na het doen **Aanvaarden** of **Compileren** in een volgende **Bestand | Editeren/Bekijken als er geen verdere bewerkingen zijn gedaan** (**Opslaan** werd uitgeschakeld en er was geen automatische prompt voor **Opslaan** op programma **Exit**).
- 5) operators '**/=**' en '**%=**' gaf alleen correcte resultaten voor '**unsigned**' linker variabelen
- 6) Het ternair conditioneel '**(bval) ? S1:S2**' gaf een onjuist resultaat wanneer '**S1**' of '**S2**' was een lokale uitdrukking in plaats van een variabele.
- 7) Functie '**noTone()**' is gecorrigeerd om te worden '**noTone(uint8_t pin)**'.
- 8) Een al lang bestaande bug veroorzaakte een crash na de procedure vanaf a **Reset** wanneer die Reset midden in een functie werd gedaan, die werd opgeroepen met een van zijn parameters die ontbrak (en dus een standaard initialisatorwaarde ontving).
- 9) Uitdrukkingen van leden (bijv. '**myobj.var**' of '**myobj.func()**') erfden het '**unsigned**' eigendom van hun rechterzijde ('**var**' of '**func()**') en kon dus niet direct worden vergeleken of gecombineerd met andere '**unsigned**' typen - een tussentijdse toewijzing aan een '**unsigned**' variabele was eerst vereist.
- 10) UnoArduSim stond erop dat als een functie; s-definitie een parameter had met een standaard-initialisator, die op de functie een eerdere prototype-declaratie heeft.
- 11) Oproepen naar '**print(byte bvar, base)**' ten onrechte gepromoot '**bvar**' aan een '**unsigned long**', en dus teveel cijfers geprint.
- 12) '**String(unsigned var)**' en '**concat(unsigned var)**' en operatoren '**+=(unsigned)**' en '**+(unsigned)**' onjuist gemaakt '**signed**' strings in plaats daarvan.
- 13) Een 'R=1K' apparaat geladen van een **IODevices.txt** bestand met positie '**U**' werd ten onrechte getekend met zijn schuifregelaar (altijd) in de **tegenovergestelde positie** van zijn echte elektrische positie.

- 14) Poging om op de standaard te vertrouwen `'inverted=false'` argument bij het declareren van een `'SoftwareSerial()'` object veroorzaakte een val en passeerde `'inverted=true'` werkte alleen als de gebruiker programma een volgende deed `'digitalWrite(txpin, LOW)'` om eerst de vereiste inactiviteit vast te leggen `'LOW'` niveau op de `TX` pin.
- 15) 'I2CSLV' apparaten heeft niet gereageerd op wijzigingen in hun pin-invoervakken (de standaardwaarden voor A4 en A5 bleven van kracht).
- 16) 'I2CSLV' en 'SPISLV' apparaten hebben gedeeltelijke bewerkingen niet gedetecteerd en gecorrigeerd toen de muis de randen verliet
- 17) Pin-waarden voor apparaten die een SPISLV of SRSLV volgden, werden onjuist opgeslagen in de **IODevs.txt** bestand als hexadecimals.
- 18) Als u probeerde meer dan één SPISLV apparaat MISO op pin 12 aan te sluiten, genereerde u altijd een elektrische Conflict-fout.
- 19) Schakelen van een pin van `'OUTPUT'` terug naar `'INPUT'-modus` kan het gegevensgrendelniveau van de pin niet resetten naar `'LOW'`.
- 20) Gebruik makend van `'sendStop=false'` in oproepen naar `'Wire.endTransmission()'` of `'Wire.requestFrom()'` heeft een fout veroorzaakt.
- 21) UnoArduSim heeft een fout toegestaan `'SoftwareSerial'` ontvangst gebeurt gelijktijdig met een `'SoftwareSerial'` transmissie.
- 22) Variabelen verklaard met `'enum'` type kon na hun aangiftelijik geen nieuwe waarde toegewezen krijgen, en UnoArduSim herkende geen 'enum'-leden wanneer er naar verwezen werd met een (wettelijk) `'enumname::'` voorvoegsel.

V2.2 - Jun. 2018

- 1) Het aanroepen van een functie met minder argumenten dan nodig was voor de definitie (wanneer die functie "vooraf gedefinieerd" was, dat wil zeggen, wanneer het geen eerdere prototype-declaratieregels had) veroorzaakte een geheugenfout en crash.
- 2) Sinds V2.1, **waveforms** was niet bijgewerkt tijdens **Uitvoeren** (alleen bij **Halt** na een **Stap**) - in aanvulling op, **VariabelenPaneel** waarden werden niet lang bijgewerkt **Stap** activiteiten.
- 3) Een beetje minderjarig **Golfvorm** scrollen en zoomen problemen die hebben bestaan sinds V2.0 zijn nu opgelost.
- 4) Zelfs in eerdere versies, de Reset op $t = 0$ met een PULSER of FUNCGEN, waarvan de periode zijn allerlaatste cyclus zou zijn voorafgaand aan $t = 0$ wees alleen een gedeeltelijk cyclus, resulteerde in zijn **Golfvorm** na $t = 0$ wordt verschoven van zijn werkelijke positie door deze (of de resterende) fractionele cyclushoeveelheid, ofwel naar rechts ofwel naar links (respectievelijk).
- 5) Enkele problemen opgelost met accentuering van syntax-kleuren in **Editeren/Bekijken**.
- 5) Sinds V2.0 werkte één objecten op uitbreiden in een reeks van objecten niet goed.
- 6) `'delay(long)'` is gecorrigeerd om te zijn `'delay(unsigned long)'` en `'delayMicroseconds(long)'` is gecorrigeerd om te zijn `'delayMicroseconds(unsigned int)'`.
- 7) Vanaf V2.0, functies bevestigd met behulp van `'attachInterrupt()'` werden niet gecontroleerd als geldig functies voor dat doel (d.w.z. `'void'` terugkeer en zonder oproepparameters).
- 8) Het effect van `'noInterrupts()'` op functies `'micros()'`, `'mills()'`, `'delay()'`, `'pulseInLong()'`; de impact op `'Stepper::step()'` en verder `'read()'` en `'peek()'` time-outs; bij alle RX seriële ontvangst en op `'Serial'` overdracht, wordt nu nauwkeurig gereproduceerd.
- 9) De tijd doorgebracht in de interruptroutines van de gebruiker wordt nu verantwoord in de geretourneerde waarde `'pulseIn()'`, de vertraging geproduceerd door `'delayMicroseconds()'`, en in de positie van randen in weergegeven **Pins Golfvormen Digitale**.
- 10) Oproepen naar **object-lid** functies die deel uitmaakten van grotere complexe expressies, of zelf functie-calls waren met meerdere complexe argumenten, bijv. `'myobj.memberfunc1() + count/2'` of `'myfunc(myobj.func1(), count/3)'`, zou onjuiste waarden berekend / gepasseerd op uitvoering tijd als

gevolg van defecte stack-ruimte toewijzingen.

- 11) Reeksen van wijzer variabelen werkte naar behoren, maar had defecte weergegeven waarden die worden weergegeven in de **VariabelenPaneel**.
- 12) Wanneer dynamische reeksen van eenvoudige soort werden gemaakt met '**new**', alleen het eerste element had een (standaard) initialisatie naar waarde 0 gekregen - nu alle elementen.
- 13) '**noTone()**', of het einde van een eindige toon, stelt de pin niet langer opnieuw in (het blijft '**OUTPUT**' en gaat '**LOW**').
- 14) 'SERVO' apparaten met continue rotatie is nu perfect stationair met een pulsbreedte van 1500 microseconden.
- 15) Het aanroepen van `SdFile::ls()` (directoryvermelding SD-kaart) werkte naar behoren, maar vertoonde ten onrechte enkele gedupliceerde blok-SPI-overdrachten in de Waveforms window.

V2.1.1- mrt. 2018

- 1) Inconsistenties in niet-Engelse landinstellingen opgelost met de taal waarin is opgeslagen '**myArduPrefs.txt**', met weergegeven radioknoppen in de **Voorkeuren** dialog-box, en met bijpassende naar vertaalde regels in '**myArduPrefs.txt**'.
- 2) Toewijzingen met '**new**' accepteer nu een reeks-dimensie van een geheel getal die geen constante is.
- 3) Klik in de **VariabelenPaneel** naar uitbreiden zou een multi-dimensionale reeks een overbodige leegte laten zien '**[]**' beugelvormige paar.
- 4) Reeks-elementreferenties met achterliggende overvloedige tekens (bijv. '**y[2]12**') werden niet gepakt als fouten op Analyseren-tijd (de extra karakters werden gewoon genegeerd).

V2.1- mrt. 2018

- 1) Een bug in de nieuwe versie V2.0.x zorgde ervoor dat de Windows-hoop groeide bij elke update in de **VariabelenPaneel** -- na miljoenen updates (vele minuten uitvoering waard), een crash kan het gevolg zijn.
- 2) Oproepen naar 'static' lid functies met dubbele dubbele punt '::' notatie mislukte Analyseren binnenin '**if()**', '**while()**', '**for()**', en '**switch()**' haakjes, en wanneer inside-expressies worden gebruikt als functie-call-argumenten of reeks-indexen.

V2.0.2 feb. 2018

- 1) Een bug geïntroduceerd in V2.0 veroorzaakte een **Bestand | Laden** crash als een '**#include**' verwezen naar een ontbrekende of lege bestand
- 2) Binnen een **IOdevs.txt** bestand, hij '**I/O**' naam 'One-Shot' werd verwacht in plaats van de oudere 'Oneshot'; beide zijn nu geaccepteerd.

V2.0.1- Jan. 2018

- 3) In niet-Engelstalige talen, '**en**' werd onjuist weergegeven zoals geselecteerd in **Voorkeuren**, waardoor terugkeren naar het Engels onhandig is (vereist de-selectie en opnieuw selecteren).
- 4) De gebruiker had de mogelijkheid om een Apparaat pin-bewerkingsvakwaarde in een onvolledige staat (zoals '**A_**') achter te laten en de '**DATA**'-bits van een '**SRS:V**' onvolledig te laten.
- 5) Het maximale aantal Analoge-schuifregelaars was beperkt tot 4 (nu gecorrigeerd tot 6).
- 6) UnoArduSim dringt niet langer door '**=**' verschijnen in een reeks-aggregatie-initialisatie.
- 7) UnoArduSim had aangedrongen op het argument "**inverted_logic**" '**SoftwareSerial()**'.
- 8) Bit-shift-bewerkingen maken nu verschuivingen langer dan de grootte van de verschoven variabele mogelijk.

V2.0- december 2017

- 1) Alle functies die zijn gedeclareerd als '**unsigned**' waren niettemin waarden terugkerend alsof ze dat waren '**signed**'. Dit had geen effect als het '**return**' waarde was toegewezen aan een '**unsigned**' variabele, maar zou een ongepast hebben veroorzaakt negatieve interpretatie als het MSB == 1 had, en het werd vervolgens toegewezen aan een '**signed**' variabele, of getest in een ongelijkheid.
- 2) Analoge Sliders bereikten alleen een maximum '**analogRead()**' waarde van 1022, niet de juiste 1023.
- 3) Een bug per ongeluk geïntroduceerd in V1.7.0 in logica gebruikt om de afhandeling van het SPI-systeem te versnellen SCK pin veroorzaakte SPI-overdrachten voor '**SPI_MODE1**' en '**SPI_MODE3**' om te falen na de overdracht van de eerste byte (een onechte extra SCK-overgang volgde elke byte). Ook updates voor een 'SPISLV'-bewerkingsbox 'DATA' voor bytes die werden overgedragen, waren vertraagd,
- 4) De gekleurde LED apparaat maakte geen lijst van 'B' (voor blauw) als kleuroptie (hoewel het werd geaccepteerd).
- 5) De instellingen voor 'SPISLV' en 'I2CSLV' apparaten werden niet opgeslagen voor de gebruiker '**I/O Apparaten**' bestand.
- 6) Kopiëren '**Servo**' instanties zijn mislukt vanwege een defect '**Servo::Servo(Servo &toCopy)**' copy-constructor implementatie.
- 7) Buiten bereik '**Servo.writeMicroseconds()**' waarden werden correct als een fout gedetecteerd, maar de opgegeven grenswaarden bij de tekst van de foutmelding waren verkeerd.
- 8) Een wettelijke overdrachtsnelheid van 115200 werd niet geaccepteerd wanneer geladen van een '**I/O Apparaten**' tekst bestand.
- 9) Elektrische pin conflicten veroorzaakt door een bevestigde Analoge Schuifweerstand apparaat werd niet altijd gedetecteerd.
- 10) In zeldzame gevallen wordt een foute string pointer (met de string 0-terminator ontbreekt) doorgegeven aan a '**String**' functie kan ervoor zorgen dat UnoArduSim crasht.
- 11) De **CodePaneel** kon markeer de huidige Analyseren-foutlijn in de **fout** programma-module (wanneer '**#include**' was gebruikt).
- 12) Het laden van een 'I/O Apparaten' bestand die een apparaat had die (niet correct) besturen tegen 'Uno' pin 13 veroorzaakte, liet een programma hangen bij de pop-up met foutmeldingen.
- 13) UnoArduSim had per abuis toegestaan de gebruiker die niet-hex-tekens in de uitgebreid TX-buffer windows voor SPISLV en I2CSLV plakt.
- 14) Initialisaties van de aangiftelijin mislukt toen de waarde aan de rechterkant de waarde was '**return**' waarde van een object lid-functie (zoals in '**int angle = myservo1.read();**').
- 15) '**static**' lid variabelen expliciet zijn '**ClassName::**' voorvoegsels werden niet herkend als ze helemaal aan het begin van een regel stonden (bijvoorbeeld in een toewijzing aan een basis- 'class' variabele),
- 16) Roeping '**delete**' op een aanwijzer gemaakt door '**new**' werd alleen herkend als functie haakje-notatie werd gebruikt, zoals in '**delete(pptr)**'.
- 17) UnoArduSim implementatie van '**noTone()**' was ten onrechte volhardend dat een pin-argument werd verstrekt.
- 18) Wijzigingen waarbij globale 'RAM'-bytes zijn toegenomen in een programma die is gebruikt '**String**' variabelen (via **Editeren/Bekijken** of **Bestand | Laden**), zou kunnen leiden tot corruptie in die 'Uno' globale ruimte als gevolg van heap deletie van de '**String**' objecten behorend tot de oude programma tijdens het (verkeerd) gebruiken van de heap behorende bij de nieuwe programma. In sommige omstandigheden kan dit leiden tot een crash met programma. Hoewel een tweede Laden of Analyseren het probleem heeft opgelost, is deze bug eindelijk opgelost.
- 19) De retourwaarden voor '**Wire.endTransmission()**' en '**Wire.requestFrom()**' waren allebei vastgelopen op 0 - deze zijn nu opgelost.

Veranderingen / Verbeteringen

V2.9.2 - april 2021

- 1) Een "Wat is dit" Help-moduscommando (Ctrl +>) toegevoegd aan het hoofdvenster, en aan alle pop-upberichtvensters (titelbalk '?') - nadat u deze Help-modus hebt geopend, kunt u op het hoofdvenster klikken Menu-items, Tool-Bar-items, of Status-Bar-tekst, of waarschuwings- / foutmeldingen in het pop-upvenster, respectievelijk om hun Engelse uitleg te bekijken, en (optioneel) om die tekst naar het systeemklembord te kopiëren, van waaruit u deze kunt plakken uw favoriete vertaal-app om de uitleg in uw taal naar keuze te krijgen.
- 2) Regelnummers toegevoegd in **Editeren/Bekijken**.
- 3) Het invoervak voor de variabelenaam alleen-lezen gemaakt in **Editeren/Opvolgen**.

V2.9 - januari 2021

- 4) Er is een verborgen waarde toegevoegd aan de USB 'SERIAL' apparaat om een directe verbinding met een 'COM'-poort mogelijk te maken - specificeer in uw 'IODevices.txt' bestand het meervoudige sleutelwoord 'waarden' en voeg vervolgens na de baudrate-waarde een komma toe gevolgd door de dubbele aanhalingstekens naam voor de gewenste 'COM'-poort om toe te voegen. Dit zou bijvoorbeeld uw UnoArduSim programma in staat stellen om te praten met een echte 'Uno' printplaat die is aangesloten op 'COM3', of om 'COM'-poortcommunicatie van een aparte toepassing te verbinden met de 'Serial' van uw UnoArduSim programma (met behulp van een nulmodem-emulator programma).
- 5) Ondersteuning toegevoegd voor voorwaardelijke compilatie met behulp van compiler-richtlijnen '**#ifdef**', '**#ifndef**', '**#else**' en '**#endif**' (enkel en alleen). Ondersteuning voor '**#undef**' is nu ook toegevoegd (om een vorig '**#define**').
- 6) Wanneer ze snel worden omgeschakeld, zijn de helderheidsniveaus van lichtemitterende diodes nu afhankelijk van hun percentage aangestoken tijd

V2.8.2- September 2020

- 1) Om verwarring te voorkomen, staat UnoArduSim nu toe ' < > ' driehoekige haakjes en dubbele aanhalingstekens die moeten worden gebruikt verwisselbaar rond de naam bestand in '**#include**' statements voor zowel user-local als '3rdParty' bestanden.
- 2) De '**MOTOR**' apparaat kan nu drie verborgen waarden accepteren van een '**IODevs.txt**' bestand: 'F' of 'B' (vrijloop- uitloop- of remmodus), vervolgens constant-load-koppel als percentage van het stall-koppel en vervolgens het laadmoment van inertie als een veelvoud van de '**MOTOR**' traagheid van de rotor.
- 3) UnoArduSim accepteert nu '**long int**' als synoniem voor '**long**'.
- 4) UnoArduSim geeft nu een eenmalige waarschuwing om te gebruiken '**volatile**' wanneer een globale variabele wordt gewijzigd door een gebruikersonderbrekingsroutine.

V2.8.0- Juni 2020

- 5) Nieuwe 'Mega2560' printplaat toegevoegd **Voorkeuren** optie (Printplaat nummer == 10). Deze functies veel meer 'I/O' pins, nog 4 externe interrupts (pins 18, 19, 20, 21), nog drie '**HardwareSerial**' poorten (op pins 22-27) en het RAM-geheugen is toegenomen van 2 KBytes tot 8 Kbytes.
- 6) Het 'SFTSER'-apparaat is hernoemd naar 'ALTSER' (aangezien het nu ook kan worden gebruikt met 'Serial1', 'Serial2', en 'Serial3').
- 7) De 'PROGIO' apparaat biedt nu een optionele volgorde van 4 master pin-nummers om een te definiëren expliciete toewijzing van de 4 'Uno'-slave printplaat pins naar de 4 master ('Uno' of 'Mega') printplaat pins.

- 8) Als u in een apparaat pin-invoervak klikt, wordt nu het hele nummer geselecteerd om de invoer gemakkelijker te maken, en klikken in een apparaten-nummer-invoervak in Configureren | 'I/O' Apparaten doet hetzelfde.
- 9) Oranje markering toegevoegd van de relevante bronregel voor waarschuwingspop-ups voor parsen en uitvoeren.
- 10) Ondersteuning toegevoegd voor '`digitalPinToInterrupt()`' oproepen.
- 11) Klassen ontvangen nu altijd een standaardconstructor als ze geen door de gebruiker opgegeven constructor hebben (zelfs als ze geen basisklasse of object-leden hebben).

V2.7.0- maart 2020

- 1) In aanvulling op de huidige code-lijn (groen als klaar om te draaien, rood als fout), UnoArduSim onderhoudt nu, voor elke module, de laatste-gebruiker heeft geklikt of stack-navigatie code-lijn (gemarkeerd met een donkere olijf achtergrond), het maken van het makkelijker in te stellen en te vinden tijdelijke breekpunt lijnen (één per module is nu toegestaan, maar alleen de ene in de weergegeven module is in feite een 'Run-To').
- 2) Toegevoegd nieuwe 'I/O' apparaten (en het ondersteunen van 3rd-party bibliotheek code), met inbegrip van 'SPI' en 'I2C' **poortuitbreiders** En 'SPI' en 'I2C' **Multiplexer LED** Controllers en displays (LED reeksen, 4-alfanumerieke en 4-cijfer of 8-cijfer 7-segment displays).
- 3) '**Wire**' operaties niet langer geweigerd gebruiker vanuit interruptroutines (deze dragers externe interrupts uit een 'I2C' poortexpander).
- 4) Digitaal golfvormen tonen nu een tussenliggend niveau (tussen '**HIGH**' en '**LOW**') Wanneer de pin niet wordt gedreven.
- 5) Om verwarring te voorkomen bij het doorlopen van meer dan één '`SPI.transfer()`' instructies, UnoArduSim maakt het nu ervoor dat bevestigd 'I/O' apparaten nu hun (-logic vertraagd) definitief 'SCK' klok rand te ontvangen voor de functie terugkeert.
- 6) Wanneer de auto-tab-opmaak **Voorkeur** is ingeschakeld, het typen van een sluiting accolade '}' in **Editeren/Bekijken** veroorzaakt nu een sprong naar het tabblad streepje positie van de bijpassende opening accolade '{' partner.
- 7) EEN **Re-Format** knop is toegevoegd aan **Editeren/Bekijken** (Veroorzaken onmiddellijke auto-tab-streepje opnieuw formatteren) - deze knop is alleen ingeschakeld als de auto-tab-streepje voorkeur is ingeschakeld.
- 8) Een duidelijker foutbericht nu wanneer een prefix zoekwoord (zoals '`const`', '`unsigned`', of 'PROGMEM') voert een identificatiecode in een verklaring (het moet de identifier voorafgaan).
- 9) Geïnitieerd wereldwijde variabelen, zelfs als nooit later gebruikt, worden nu steeds een geheugen-adres toegewezen, en zo zal zichtbaar verschijnen.

V2.6.0 januari 2020

- 1) Toegevoegd karakter LCD display apparaten met 'SPI', 'I2C', en 4-bi-parallele interface. Ondersteunende bibliotheek broncode is toegevoegd aan de map nieuwe installatie 'include_3rdParty' (en kan worden benaderd door gebruik te maken van een normale '`#include`' richtlijn) - gebruikers kunnen daarnaast voor kiezen om in plaats daarvan schrijven hun eigen functies om besturen de LCD apparaat.
- 2) **CodePaneel** markering is verbeterd, met aparte markeer kleuren voor een kant code-lijn, een foutcode-lijn, en een andere code-lijn.
- 3) De **Vinden** menu en het gereedschap-bar 'func' acties (vorig-up en next-down) niet langer naar het vorige / volgende functie startlijn, en in plaats daarvan nu stijgen (of afdalen) de call-stack, aandacht voor de betreffende code-lijn in de beller (of opgeroepen) functie, respectievelijk, waarbij de **VariabelenPaneel** Inhoud wordt op variabelen tonen de functie met het gemarkeerde code-lijn.
- 4) Om verwarring te voorkomen, een **Opslaan** gedaan binnen **Editeren/Bekijken** veroorzaakt een onmiddellijke re-**Compileren**, En als de Opslaan was succesvol, met behulp van een daaropvolgende Afbreken of Exit zal nu alleen tekst terug dat laatst opgeslagen tekst.

- 5) Voegde een gepulseerde ingang Stappenmotor ('PSTEPR') met 'STEP' (puls), 'EN*' (inschakelen) en 'DIR' (richting) ingangen en een microstappen per stap-instelling (1,2,4,8 of 16) .
- 6) Zowel 'STEPR' en 'PSTEPR' apparaten hebben nu een 'sync' LED (GROEN voor gesynchroniseerde of rood wanneer uitgeschakeld door één of meer stappen.)
- 7) 'PULSER' apparaten hebben nu de keuze tussen microseconden en milliseconden voor 'Period' en 'Pulse'.
- 8) Ingebouwd-functie automatische aanvullingen niet langer behouden het type parameter voor de naam van de parameter.
- 9) Bij het schakelen naar een eerdere **CodePaneel**, De eerder geselecteerde regel wordt nu opnieuw gemarkeerd.
- 10) Als hulpmiddel voor het instellen van een tijdelijke knikpunt gebruik of Afbreken Exit van **Editeren/Bekijken** laat de markeer in de **CodePaneel** op de lijn voor het laatst bezocht door de cursor in **Editeren/Bekijken**.
- 11) Een door de gebruiker gedefinieerde (of 3rd party) '**class**' nu mag gebruiken '**Print**' of '**Stream**' als basisklasse. Ter ondersteuning hiervan heeft een nieuwe map 'include_Sys' toegevoegd (in het UnoArduSim installatiemap) dat de broncode voorziet elke basis '**class**'. In dit geval, vraagt om een dergelijke base '**class**' functies zal dus hetzelfde worden behandeld user code (dat) kan worden opgevoerd in), in plaats van als een ingebouwd functie die niet kan worden opgevoerd in (bijvoorbeeld '**Serial.print()**').
- 12) Lid-functie auto-aanvullingen nu ook de naam van de parameter **in plaats van** het type.
- 13) De UnoArduSim Analyseren maakt nu een object naam in een variabele verklaring voorafgegaan door zijn optioneel (en matching) '**struct**' of 'class' sleutelwoord, gevolgd door '**struct**' of '**class**' naam.

V2.5.0 oktober 2019

- 1) Ondersteuning toegevoegd voor '**TFT.h**' library (met uitzondering van de '**drawBitmap()**'), En ook een bijbehorende 'TFT' 'I/O' Apparaat (128 bij 160 pixels). Merk op dat om buitensporige real-time vertragingen tijdens grote vermijden '**fillXXX()**' overdracht sa gedeelte van de 'SPI' overdrachten **in het midden van de vulling** afwezig zal zijn van de 'SPI' bus.
- 2) Tijdens grote bestand transfers door middel van 'SD', een gedeelte van de 'SPI' overdrachten in het midden van de reeks bytes soortgelijke afwezig 'SPI' de bus.
- 3) Verminderde '**Stream**'-Gebruik overhead bytes, zodat '**RAM free**' waarde beter overeenkomt met Arduino compilatie.
- 4) UnoArduSim waarschuwt de gebruiker nu wanneer een '**class**' heeft meerdere leden verklaard op een verklaring lijn.
- 5) Met behulp van 'File | Save As'zet nu de huidige directory dat opgeslagen-in directory.
- 6) De twee ontbrekende '**remove()**' lid functies zijn toegevoegd aan de '**String**' klasse.
- 7) UnoArduSim nu verbiedt aannemer base-gesprekken in een constructor-functie prototype, tenzij de volledige functie lichaamsdefinitie onmiddellijk volgt (om zo te stemmen met de Arduino compiler).
- 8) edge overgangstijd van digitaal golfvormen werd teruggebracht tot visualisatie van snelle 'SPI' signalen te ondersteunen op hoogste zoom.
- 9) Un oArduSim laat nu een aantal constructeurs worden verklaard '**private**' of '**protected**' (Voor intern klasse gebruik).

V2.4 mei 2019

- 1) Alle 'I/O'-apparaatbestanden worden nu opgeslagen in een vertaalde taal en worden samen met het

Voorkeuren bestand nu opgeslagen in UTF-8-tekstcodering om overeenkomende fouten bij het volgende lezen te voorkomen.

2) Nieuwe toegevoegd '**PROGIO**' Apparaat, een blote programmeerbare slave 'Uno' printplaat die maximaal 4 pins gemeen heeft met de **LabtafelPaneel** meester 'Uno', - een slaaf 'Uno' kan hebben Nee 'I/O' apparaten van zijn.

3) U kunt nu elke 'I/O' apparaat verwijderen door erop te klikken terwijl u op de 'Ctrl'-toets drukt.

4) In **Editeren/Bekijken** tekst automatisch aanvullen is toegevoegd voor globale, ingebouwde ins en lid variabelen en functies (gebruik ALT-rechterpijl om een voltooiing aan te vragen, of **invoeren** als de lijst Ingebouwde apparaten momenteel een overeenkomende selectie markeert).

5) In **Voorkeuren** , een nieuwe keuze maakt automatische invoeging van een regelafsluitende puntkomma bij n mogelijk **invoeren** toetsaanslag (als de huidige regel een uitvoerbare instructie is die lijkt op zichzelf te staan en volledig is).

6) persing '**Ctrl-S**' van een **Golfvorm** window stelt u in staat om alles op bestand op te slaan (**X, Y**) wijst langs de weergegeven sectie van elke golfvorm (waarbij X microseconden is van de meest linkse golfvorm punt en Y is volt).

7) Een 'SFTSER' 'apparaat heeft nu een verborgen (optioneel) '**inverted**' waarde (**geldt voor zowel TX als RX**) die kan worden toegevoegd na de baudrate waarde aan het einde van zijn regel in een **IODevs.txt** bestand.

8) Toegevoegd '**SPISettings**' klasse, functies '**SPI.transfer16()**' , '**SPI.transfer(byte* buf, int count)**' , '**SPI.beginTransaction()**' , en '**SPI.endTransaction()**' , net zoals '**SPI.usingInterrupt()**' en '**SPI.notUsingInterrupt()**' .

9) SPI-bibliotheek functies toegevoegd '**SPI.detachInterrupt()**' samen met een SPI-bibliotheekuitbreiding '**SPI.attachInterrupt(void myISRfunc)**' (in plaats van de eigenlijke bibliotheek functie '**SPI.attachInterrupt(void)**' (om te voorkomen dat generieke informatie moet worden herkend '**ISR(int vector)**' low-level vectored interrupt functie declaraties).

10) Het SPI-systeem kan nu in de slavemodus worden gebruikt, door het maken van de '**SS**' pin (pin 10) an '**INPUT**' pin en rijden '**LOW**' na '**SPI.begin()**' of door op te geven '**SPI_SLV**' als de optionele '**mode**' parameter in '**SPI.begin(int mode=SPI_MSTR)**' (een andere UnoArduSim-extensie voor '**SPI.h**'). Ontvangen bytes kunnen vervolgens worden verzameld met behulp van '**rxbyte = SPI.transfer(tx_byte)**' ofwel in een niet-SPI-onderbrekings functie of in een gebruikeronderbrekingsservice functie die eerder is aangesloten door '**SPI.attachInterrupt(myISRfunc)**'. In de slavemodus, '**transfer()**' wacht tot een gegevensbyte klaar is in SPDR (dus blokkeer normaal gesproken wachten op een complete byte-ontvangst, maar in een interruptroutine zal dit gebeuren **terugkeer** onmiddellijk omdat de ontvangen SPI-byte er al is). In elk geval, '**tx_byte**' wordt in de SPDR geplaatst, dus bij de volgende ontvangt de bijgevoegde hoofd-SPI deze '**transfer()**' .

11) Ondersteuning voor de Slaaf-modus is toegevoegd aan de UnoArduSim-implementatie van 'Wire.h'. Functie '**begin(uint8_t slave_address)**' is nu beschikbaar, zoals het is '**onReceive(void*)**' en '**onRequest(void*)**' .

12) '**Wire.end()**' en '**Wire.setClock(freq)**' kan nu worden gebeld; de laatste om de SCL-frequentie in te stellen met a '**freq**' waarde van ofwel 100.000 (de standaard standaardmodus SCL-frequentie) of 400.000 (snelmodus).

13) 'I2CSLV' apparaten reageren nu allemaal op de **0x00** algemene bus-adres, en zo **0x00** mag niet langer worden gekozen als een uniek I2C-busadres voor een van die slaven.

14) De gemodelleerde uitvoering-vertragingen van elementaire integer- en toewijzingsbewerkingen en reeks en aanwijzerbewerkingen zijn verminderd en 4 microseconden zijn nu toegevoegd voor elke bewerking met drijvende komma.

V2.3 december 2018

- 1) Volgen is nu ingeschakeld op de **Tool-Bar** 'I/O ____ S' **schuif** voor continu en soepel schalen van 'I/O' apparaat-waarden dat de gebruiker het achtervoegsel 'S' heeft toegevoegd.
- 2) Een nieuw '**LED4**' 'I/O' apparaat (rij van 4 LED's aan **4 opeenvolgende pin-nummers**) is toegevoegd.
- 3) Een nieuw '**7SEG**' 'I/O' apparaat (7-segment LED cijfer met hexadecimaal-code aan **4 opeenvolgende pin-nummers** en met actief-laag **CS** * selecteer invoer), is toegevoegd.
- 4) Een nieuw '**JUMP**' 'I/O' apparaat die fungeert als een draadbrug tussen twee 'Uno' pins is toegevoegd. Dit staat een '**OUTPUT**' pin moet worden aangesloten op een '**INPUT**' pin (zie de apparaat hierboven voor mogelijk gebruik van deze nieuwe functie).
- 5) Een nieuw '**OWISLV**' 'I/O' apparaat is toegevoegd en de derde partij '**<OneWire.h>**' bibliotheek kan nu worden gebruikt met '**#include**' zodat gebruiker programmas interfacing kan testen op een kleine subset van '1-Wire'-bus apparaten.
- 6) De **Uitvoeren** menu **Reset** opdracht is nu verbonden met de **Reset** knop.
- 7) Voor meer duidelijkheid, wanneer **Kunstmatige 'loop()' vertraging** is geselecteerd onder de **Opties** menu, een expliciete '**delay(1)**' gesprek wordt toegevoegd aan de binnenkant van de lus '**main()**' - dit is nu een echte vertraging die onderbroken kan worden door onderbrekingen van de gebruiker 'Uno' pins 2 en 3.
- 8) Elektrische pin conflicten met open-afvoer of CS-geselecteerd, 'I/O' apparaten (bijv. I2CLV of SPISLV) zijn nu gedeclareerd **alleen wanneer een echte conflict optreedt op uitvoering-tijd** in plaats van een directe fout te veroorzaken wanneer de apparaat voor het eerst wordt aangesloten.
- 9) Functie '**pulseInLong()**' is nu nauwkeurig tot 4-8 microseconden om het eens te zijn met Arduino (de vorige nauwkeurigheid was 250 microseconden).
- 10) Fouten gemarkeerd tijdens initialisatie van een globale variabele nu markeer die variabele in de **CodePaneel**.

V2.2 jun. 2018

- 1) Op **Opslaan** van de **Voorkeuren** dialoogvenster, of uit **Configureren | 'I/O' Apparaten**, de '**myArduPrefs.txt**' bestand wordt nu opgeslagen in de directory van de momenteel geladen programma - elke volgende **Bestand | Laden** laadt vervolgens automatisch de bestand, samen met de opgegeven IODev's bestand, uit dezelfde programma-directory.
- 2) Functie '**pulseInLong()**' **was geweest** ontbreekt, maar is nu toegevoegd (het is afhankelijk van '**micros()**' voor zijn metingen).
- 3) Wanneer een gebruiker programma doet een '**#include**' van een '***.h**' bestand, UnoArduSim probeert nu ook automatisch het overeenkomstige te laden '***.c**' bestand **als** een overeenkomstige '***.cpp**' bestand is niet gevonden.
- 4) Automatisch invoegen van een close-accolade '**}**' (na elke open-accolade '**{**') is toegevoegd aan **Voorkeuren**.
- 5) Een nieuw **Opties** menu keuze maakt nu mogelijk '**interrupts()**' te worden aangeroepen vanuit een gebruikersinterruptieroutine - dit is alleen voor educatieve doeleinden, aangezien het nesten van interrupts in de praktijk moet worden vermeden.
- 6) Overzetting van verwijzingen naar een '**int**' waarde wordt nu ondersteund (maar er verschijnt een waarschuwingpop-upbericht).
- 7) UnoArduSim ondersteunt nu gelabelde programma-lijnen (bijv '**LabelName: count++;**' voor gebruiksgemak (maar '**goto**' is nog steeds **afgekeurd**)
- 8) Uitvoering-waarschuwingen komen nu voor wanneer een oproep naar '**tone()**' kan interfereren met actieve PWM op pins 3 of 11, wanneer '**analogWrite()**' zou interfereren met een Servo die al actief is op dezelfde pin, wanneer een serie karakter aankomst gemist wordt omdat interrupts momenteel uitgeschakeld zijn, en wanneer interrupts zo snel zouden komen dat UnoArduSim enkele van hen zal missen.

V2.1 mrt. 2018

- 1) weergegeven **VariabelenPaneel** waarden worden nu slechts elke 30 milliseconden vernieuwd (en de Minimal-optie kan die verversingsfrequentie nog verder verlagen), maar de **VarRefresh** menuoptie om update-reductie te verwijderen is verwijderd.
- 2) Bewerkingen die slechts een gedeelte van de bytes van een variabele-waarde targeten (zoals die die via pointers zijn gemaakt), nu de verandering veroorzaken die de variabele-waarde moet weerspiegelen in de **VariabelenPaneel** display.

V2.0.1 januari 2018

- 1) Onbeschreven Arduino functies '**exp()**' en '**log()**' zijn nu toegevoegd.
- 2) 'SERVO' apparaten kan nu continu worden gedraaid (zodat de pulsbreedte de snelheid regelt in plaats van de hoek).
- 3) In **Editeren/Bekijken**, een afsluitende accolade '**}**' wordt nu automatisch toegevoegd wanneer u een openings accolade typt '**{**' als je dat hebt geselecteerd **Voorkeur**.
- 4) Als u op klikt **Editeren/Bekijken** window-titelbalk '**x**' om af te sluiten, krijgt u nu de kans om af te breken als u de weergegeven programma bestand had gewijzigd, maar niet opgeslagen.

V2.0 september 2017

- 1) De implementatie is geporteerd naar QtCreator, dus de GUI heeft wat kleine visuele verschillen, maar geen functionele verschillen, behalve enkele verbeteringen:
 - a) Statusberichten aan de onderkant van de hoofd-window en in de **Editeren/Bekijken** dialoogvenster is verbeterd en er is een markeer-kleurcodering toegevoegd.
 - b) De verticale ruimte toegewezen tussen de **CodePaneel** en **VariabelenPaneel** is nu instelbaar via een versleepbare (maar niet zichtbare) splitsbalk aan hun gedeelde rand.
 - c) 'I/O' apparaat-bewerkingsvakwaarden worden nu pas gevalideerd als de gebruiker de muisaanwijzer buiten de apparaat heeft verplaatst - dit voorkomt onhandige automatische wijzigingen om wettelijke waarden af te dwingen terwijl de gebruiker typt.
- 2) UnoArduSim ondersteunt nu meerdere talen via **Configureren | Voorkeuren**. Engels kan altijd worden geselecteerd, naast de taal voor de landinstelling van de gebruiker (zolang een aangepaste *.qm-vertaling bestand voor die taal aanwezig is in de map UnoArduSim 'translations').
- 3) Geluid is nu aangepast om de Qt-audio-API te gebruiken - hiervoor is onder bepaalde omstandigheden dempen vereist vermijd vervelende geluiden en kliks tijdens de langere OS-venster operationele vertragingen veroorzaakt door normale muisklikken van de gebruiker - zie de sectie -sounds voor meer informatie op dit.
- 4) Als gebruiksgemak worden spaties nu gebruikt om een 0-waarde voor te stellen in de apparaat-tel-invoervakken in **Configureren | 'I/O' Apparaten** (u kunt nu de spatiebalk gebruiken om apparaten te verwijderen).
- 5) De niet-geschaalde (U) kwalificatie is nu optioneel op 'PULSER', 'FUNCGEN' en '1SHOT' apparaten (dit is de veronderstelde standaardwaarde).
- 6) UnoArduSim staat nu toe (naast letterlijke numerieke waarden) '**const**' variabelen met een gehele waarde, en '**enum**'

