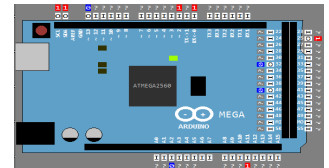


UnoArduSimV2.8.2 Ajuda Completa



Taula de continguts

[Visió General](#)

[Panell de Codi, Preferències i Editar/Examinar](#)

[Panell de Codi](#)

[Preferències](#)

[Editat/Examinar](#)

[Panell de Variables i Editar/Monitoritzar Variable finestra](#)

[Panell de Proves](#)

[El 'Uno' o 'Mega'](#)

['I/O' Dispositius](#)

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

[Sèrie Alternatiu \('ALTSER'\)](#)

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

[Pantalla TFT \('TFT'\)](#)

[SPI configurable Esclau \('SPISLV'\)](#)

[I2C de dos fils Esclau \('I2CSLV'\)](#)

[LCD de Text I2C \('LCDI2C'\)](#)

[LCD de Text SPI \('LCDSPI'\)](#)

[LCD de Text D4 \('LCD_D4'\)](#)

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

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

[Port d'Expansió SPI \('EXPSPi'\)](#)

[Port d'Expansió I2C \('EXPI2C'\)](#)

[Registre de Desplaçament Esclau \('SRSLV'\)](#)

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

[Programable 'I/O' Dispositiu \('PROGIO'\)](#)

[Un-Tret \('1SHOT'\)](#)

[Generador de Pols Digital \('PULSER'\)](#)

[Analògic Generador de Funcions \('FUNCGEN'\)](#)

[Motor Pas a Pas \('STEPR'\)](#)

[Impulsada Motor Pas a Pas \('PSTEPR'\)](#)

[Motor CC \('MOTOR'\)](#)

[Servo-Motor \('SERVO'\)](#)

[Altaveu Piezo-elèctric \('PIEZO'\)](#)

[Resistència de diapositives \('R=1K'\)](#)

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

[LED de color \('LED'\)](#)

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

[De 7 segments LED Dígit \('7SEG'\)](#)

[Control Analògic Lliscant](#)

[Connexió de Pont \('JUMP'\)](#)

[Menús](#)

[Arxiu:](#)

[Carregar INO o PDE Prog \(ctrl-L\)](#)

[Editat/Examinar \(ctrl-E\)](#)

[Desar](#)

[Desar com](#)

[Pròxim \('#include'\)](#)

[Anterior](#)

[Sortida](#)

Trobar:

Pila de trucades ascendents
Descendeu la pila de trucades
Definiu el text del Cercar (ctrl-F)
Trobar Text següent
Trobar Text anterior

Executar:

Pas a (F4)
Pas sobre (F5)
Pas cap a fora (F6)
Executar cap a (F7)
Executar fins a (F8)
Executar (F9)
Aturar (F10)
Reiniciar
Animació
Càmera Lenta

Opcions:

Pas sobre Estructors/ Operadors
Registre-Assignació
Error al no inicialitzar
Afegit 'loop()' Retard
Permeten les interrupcions imbricades

Configurar:

'I/O' Dispositius
Preferències

VarActualitzar:

Permet automàticament (-) Encongir
Mínim
Ressaltar Canvis

Finestres:

Monitor 'Serial'
Restaura tot
Formes d'Ona Digitals
Formes d'Ona Analògiques

Ajuda:

Ajuda ràpida Arxiu
Ajuda complet Arxiu
Correccions Errada
Canvis / Millores
Sobre

'Uno' o 'Mega' Placa i 'I/O' Dispositius

Temporització

'I/O' Dispositiu Temporització

Sons

Limitacions i elements no compatibles

Inclòs Arxius

Assignació de memòria dinàmica i RAM

'Flash' Assignacions de memòria

'String' Variables

Biblioteques Arduino

Punters

'class' i 'struct' Objectes

Àmbit

Qualificadors 'unsigned', 'const', 'volatile', 'static'

Directives Compilador

[Elements de llenguatge arduino](#)

[C / C ++ - elements del llenguatge](#)

[Plantilles Funció](#)

[Emulació en temps real](#)

[Notes de llançament - a partir d'V2.5 en endavant](#)

[Correccions Errada](#)

[V2.8.2- Setembre 2020](#)

[V2.8.1- juny de 2020](#)

[V2.8.0- juny de 2020](#)

[V2.7- març 2020](#)

[V2.6.0- | gener 2020](#)

[V2.5.0- oct 2019](#)

[Canvis / Milllores](#)

[V2.8.2- Setembre 2020](#)

[V2.8.0- juny de 2020](#)

[V2.7 març 2020](#)

[V2.6.0 gener de 2019](#)

[V2.5.0 oct. 2019](#)

Visió General

UnoArduSim és un programari gratuït **temps real** (vegeu per a Temporització **restriccions**) eina de simulador que he desenvolupat per als estudiants i entusiastes d'Arduino. Està dissenyat per permetre experimentar amb Arduino programes i experimentar fàcilment la depuració **sense necessitat de cap maquinari real**. Està dirigit a la web **Arduino 'Uno' o 'Mega'** placa i us permet triar entre un conjunt de 'I/O' virtual dispositius i configurar i connectar aquests dispositius al vostre 'Uno' virtual o 'Mega' a la **Panell de proves**. - no us haureu de preocupar dels errors de cablejat, de les connexions trencades / desconnectades o del dispositius defectuosos que tingui en compte el vostre desenvolupament i proves del programa.

UnoArduSim proporciona missatges d'error simples per a qualsevol error analitzar o execució que tingui, i permet depurar amb **Reiniciar**, **Executar**, **Executar cap a**, **Executar fins a**, **Aturar**, i flexible **Pas** operacions a la **Panell de Codi**, amb una visió simultània de tots els locals variables, matrius i objectes locals actuals i actualment actius a **Panell de Variables**. Es proporciona una verificació de límits a Executar i es detectarà un desbordament de la memòria RAM ATmega (es destaca la línia programa culpable). Qualsevol problema amb elèctric unit 'I/O' dispositius està indicat i es produeix a mesura que es produeixi.

Quan s'obre un KO1K INO o PDE programa, es carrega al programa **Panell de Codi**. Al programa se li dona a Analitzar, per transformar-lo en un executable tokenitzat, que ja està a punt per **execució simulat** (a diferència d'Arduino.exe, un executable autònom de binari és *no* creat) Qualsevol error de analitzar és detectat i marcat destacant la línia que ha fallat a analitzar i informant de l'error en el **Barra d'estat** a la part inferior de l'aplicació UnoArduSim finestra. Un **Editar/Examinar** finestra es pot obrir per permetre-vos veure i editar una versió ressaltada per la sintaxi del vostre usuari programa. Els informes que es produeixen durant el execució simulat (com ara un velocitat de transmissió coincident) es mostren a la barra d'estat i mitjançant una caixa de missatges emergent.

UnoArduSim V2.7 és una implementació substancialment completa de **Llenguatge de programació d'Arduino V1.8.8 tal com es documenta a la secció arduino.cc**. Pàgina web de referència d'idioma i amb addicions com s'observa a la versió de la versió Baixar Notes de llançament. Tot i que UnoArduSim no admet la implementació completa de C++ que fa l'Arduino.exe subjacent a la GNU compilador, és probable que només els programadors més avançats trobin que falta algun element C / C++ que desitgen fer servir (i, per descomptat, sempre n'hi ha de simples codificació entorns de treball per a aquestes funcions que falten). En general, només he donat suport a allò que considero que són les funcions de C / C++ més útils per als aficionats i estudiants de l'Arduino (per exemple, **'enum'** i **'#define'** s'admet, però els punters funció no ho són. Tot i que el objectes està definit per l'usuari (**'class'** i **'struct'**) i (la majoria) sobrecàrregues d'operadors són compatibles, *l'herència múltiple no ho és*.

Com que UnoArduSim és un simulador d'alt nivell d'idiomes, **només són admeses les instruccions C / C++**, *les declaracions del llenguatge de muntatge no ho són*. Igualment, perquè no es tracta d'una màquina de simulació de baix nivell, **Els registres ATmega328 no són accessibles al vostre programa** tant per llegir com per escriure, tot i que l'assignació, el retorn i el retorn del registre s'emularan (el trieu en el menú **Opcions**).

A partir de V2.6, UnoArduSim compta amb el suport automàtic integrat per a un subconjunt limitat de les biblioteques que Arduino proporciona: **'Stepper.h'**, **'Servo.h'**, **'SoftwareSerial.h'**, **'SPI.h'**, **'Wire.h'**, **'OneWire.h'**, **'SD.h'**, **'TFT.h'**, i **'EEPROM.h'** (versió 2). V2.6 introdueix un mecanisme per a 3rd suport de la biblioteca de festes mitjançant arxius inclòs al document **'include_3rdParty'** carpeta que es pot trobar al directori d'instal·lació de UnoArduSim. Per ningú **'#include'** d'altres biblioteques (és a dir, creades per l'usuari), UnoArduSim ho farà **no** busqueu l'estructura habitual del directori d'instal·lació d'Arduino per localitzar la biblioteca; en lloc vostre **necessitat** per copiar l'encapçalament corresponent (".h") i la font (".cpp") arxiu al mateix directori que el programa arxiu en què trebal·leu (subjecte, per descomptat, a la limitació que el contingut de qualsevol **'#include'** El arxiu ha de ser totalment comprensible el model UnoArduSim analitzador).

He desenvolupat UnoArduSimV2.x a QtCreator amb suport multilingüe, i actualment només està disponible per a Finestres™. Portar a Linux o MacOS, és un projecte de futur! UnoArduSim va sorgir dels simuladors que havia desenvolupat al llarg dels anys per als cursos que vaig impartir a la Universitat de la Queen i que s'ha provat raonablement de forma extensiva, però hi ha alguns errades que encara s'hi amaguen. Si voleu informar d'un errada,

descriuiu-lo (breument) en un correu electrònic a unoArduSim@gmail.com i **assegureu-vos d'adjuntar el codi font complet programa que indueix programa Arduino** de manera que puc reproduir el errada i solucionar-ho. No respondré als informes individuals de errada i no tinc terminis garantits per a les solucions en un llançament posterior (recordeu que gairebé sempre hi ha solucions solucionades!).

Salutacions,

Stan Simmons, doctor, P.Eng.
Professor associat (jubilat)
Departament d'Enginyeria Elèctrica i Informàtica
Queen's University, Kingston, Ontario, Canadà



Panell de Codi, Preferències i Editar/Examinar


(A part: la mostra finestres que es mostra a continuació es troba sota un Finestres-OS escollit per l'usuari tema de color que té un color de fons blau fosc finestra).

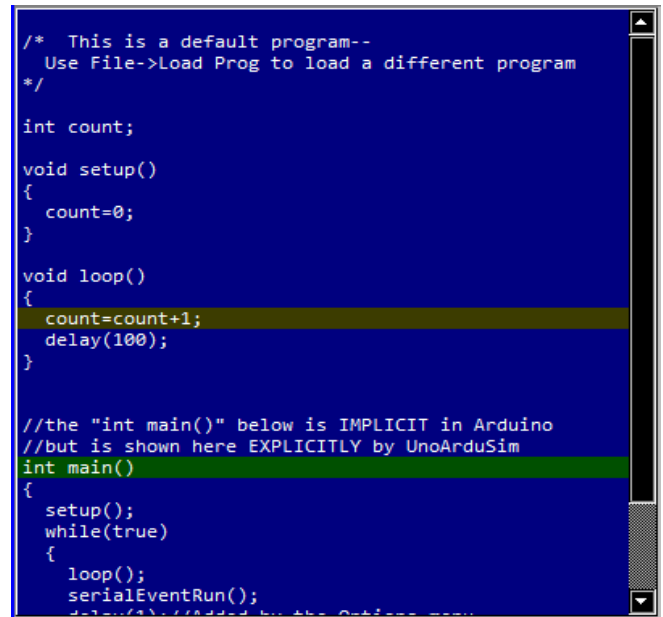
Panell de Codi

El **Panell de Codi** mostra el vostre usuari programa i **verd** destacant les pistes del seu executió. (o destacavermell per un error)

Després d'un programa carregat té un Analitzar reeixit, la primera línia entre '**main()**' es destaca i el programa està preparat per a executió. Tingues en compte que '**main()**' està afegit implícitament per Arduino (i per UnoArduSim) i ho feu **no** incloure-la com a part del vostre usuari programa arxiu. Execució està sota el control de menú **Executar**, i els seus associats **Barra d'eines** botons i dreceres de tecles funció.

Després de trepitjar executió seguint una (o més) instruccions (podeu utilitzar-la **Barra d'eines** botons **,, , o**), la línia programa que serà executat després es posa en relleu en verd - la línia destacada en verd sempre és la següent **llest per ser executat**.

Si actualment el programa executió està aturat, feu clic a **Panell de Codi** finestra, la línia que acabeu de fer clic es posa en relleu en olivera fosca (tal com es mostra a la imatge): la següent línia executat es manté ressaltada en verd (a partir de la V2.7). Però podeu provocar executió *per avançar fins* la línia en què acabeu de fer clic fent clic a **Executar cap a**  **Barra d'eines** botó. Aquesta característica us permet arribar ràpidament i fàcilment a línies específiques en un programa de manera que podríeu posteriorment trepitjar línia per línia per una porció d'interès de programa.



```
/* 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);
}

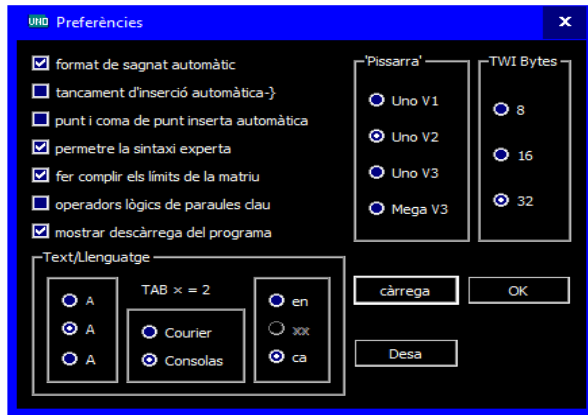
//the "int main()" below is IMPLICIT in Arduino
//but is shown here EXPLICITLY by UnoArduSim
int main()
{
  setup();
  while(true)
  {
    loop();
    serialEventRun();
  }
}
```

Si el vostre programa carregat té algun '**#include**' arxius,

podeu desplaçar-los entre ells mitjançant **Arxiu | Anterior** i **Arxiu | Pròxim** (amb **Barra d'eines** botons  i ). La darrera línia amb un clic de l'usuari de cadascun d'aquests mòduls continua sent ressaltada i defineix una possible línia punt d'aturada a la qual s'executarà, però només el punt d'aturada al *actualment es mostra el mòdul* està actiu al següent **Executar cap a**.

El **Trobar** les accions del menú us permeten fer-ho **TANT** Cerqueu el text a la secció **Panell de Codi** o **Panell de Variables** (**Barra d'eines** botons  i  o dreceres de teclat **Fletxa cap amunt** i **fletxa cap avall**) **després d'utilitzar primer** **Trobar | Definiu el text del Cercar** o **Barra d'eines** ), **O ALTRAMENT** a **navegar per magatzem de crides** a la **Panell de Codi** (**Barra d'eines** botons  i  o dreceres de teclat **Fletxa cap amunt** i **fletxa cap avall**). Claus **PgDn** i **PgUp** salta la selecció al funció següent / anterior.

Preferències



Configurar | Preferències permet als usuaris per definir el programa i les preferències de visualització (que un usuari normalment vol adoptar a que la propera sessió). Per tant, es poden desar i carregar des de a **'myArduPrefs.txt'** arxiu que resideix al mateix directori que el 'Uno' carregat o 'Mega' programa (**'myArduPrefs.txt'** es carrega automàticament si existeix).

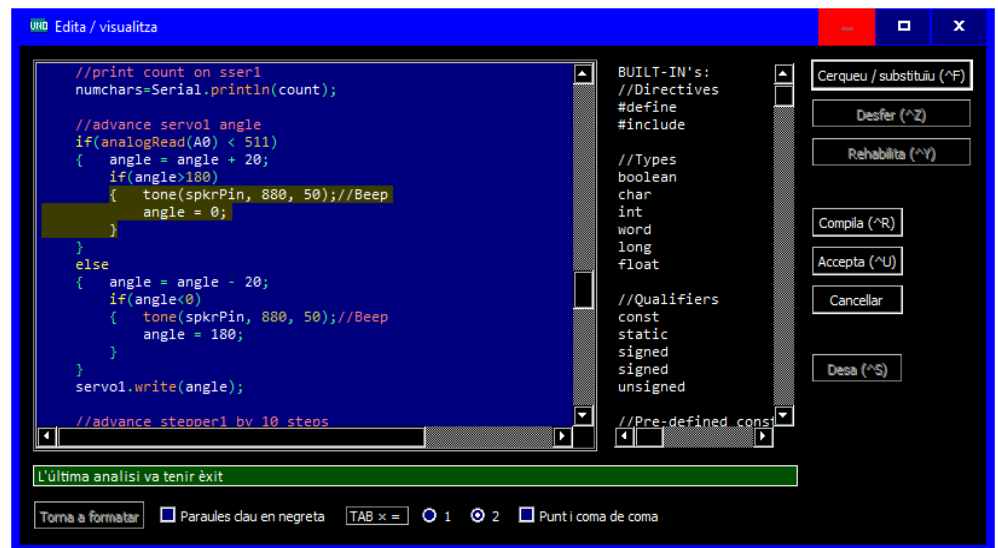
Aquest quadre de diàleg permet triar entre dos tipus de lletra monoespaiats i tres mides de tipus, i altres preferències diverses. A partir de V2.0, ara s'inclou l'elecció de l'idioma. - sempre inclou anglès (**ca**), Més un o dos altres idiomes regional de l'usuari (quan existeixen), i una substitució basada en el codi de dues lletres ISO-639 de dues lletres a

la primera línia del **'myArduPrefs.txt'** arxiu (si s'hi proporciona un). Les opcions només apareixen si Una traducció ".qm" arxiu existeix a la carpeta de traduccions (dins) el directori inicial UnoArduSim.exe).

Editar/Examinar

Feu doble clic sobre qualsevol línia del quadre de control **Panell de Codi** (o mitjançant el menú **Arxiu**), an **Editar/Examinar** finestra s'obre per permetre canvis al programa arxiu: s'obrirà amb el **línia seleccionada** actualment a la **Panell de Codi** destacat.

Aquest finestra té una capacitat d'edició completa amb un ressalt de sintaxi dinàmic (s'utilitzen diferents colors ressaltar per a paraules clau C ++, comentaris, etc.). Hi ha un ressaltat de sintaxi opcional negreta i un format automàtic de nivell de sagnia (suposant que heu seleccionat aquesta opció) **Configurar | Preferències**). També podeu seleccionar convenientment les trucades integrat funció (o integrat **'#define'** constants) a afegir al vostre programa des del quadre de llista que es proporciona - només cal fer un doble clic sobre l'element del quadre de llista desitjat per afegir-lo al programa a la posició actual de cura (truqueu a funció variable **tipus** són només per obtenir informació i es retiren per deixar els marcadors de posició maniquí quan s'afegeix al vostre programa).



El finestra té **Trobar** (ús **ctrl-F**) i **Trobar / Reemplazar** capacitat (ús **ctrl-H**) . El **Editar/Examinar** finestra té **Desfer** (**ctrl-Z**), i **Refer** (**ctrl-Y**) botons (que apareixen automàticament).

Feu servir la fletxa dreta ALT per sol·licitar opcions d'acabament automàtic per a integrat **variables mundial**, i per **membre variables i funcions**.

Descartar **tots els canvis** que heu creat des que heu obert el programa per editar-lo, feu clic a **Cancel·lar** botó. Per acceptar el estat actual, feu clic a **Acceptar** i el programa rep automàticament un altre Analitzar (i es descarrega als 'Uno' o 'Mega' si no es troben errors) i el nou estat apareix a la UnoArduSim finestra principal. **Barra d'estat** .

A **Compilar** (**ctrl-R**) botó (més un associat **Estat Analitzar** La caixa de missatges que es pot veure a la imatge superior) s'ha afegit per permetre les proves de les edicions sense necessitat de tancar primer el finestra. A **Desar** (

ctrl-S) El botó també s'ha afegit com a drecera (equivalent a an **Acceptar** més una separació posterior **Desar** del principal finestra).

En qualsevol dels dos **Cancel·lar** o **Acceptar** sense edicions realitzades, la **Panell de Codi** la línia actual canvia per convertir-se en **última posició de Editar/Examinar** i podeu utilitzar aquesta funció per saltar la opció **Panell de Codi** a una línia específica (possiblement per preparar-se per a realitzar una **Executar cap a**), També podeu utilitzar **ctrl-PgDn** i **ctrl-PgUp** per saltar al següent (o anterior) salt de línia buit del programa, això és útil per navegar ràpidament cap amunt o cap avall cap a ubicacions significatives (com les línies buides entre funcionses). També podeu utilitzar **ctrl-Home** i **ctrl-End** per saltar a l'inici i al final del programa, respectivament.

Formatació automàtica a nivell de 'Tab' es fa quan s'obre el finestra, si aquesta opció s'ha definit a **Configurar | Preferències**. Podeu tornar a fer aquesta format en qualsevol moment fent clic al botó Re-format (només està habilitat si prèviament heu seleccionat **Preferència de sagnat automàtic**). També podeu afegir o esborrar fitxes a un grup de línies consecutives preseleccionades mitjançant el teclat **fletxa dreta** o **fletxa esquerra** claus - però **sagnat automàtic La preferència ha d'estar desactivada** per no perdre els vostres nivells de pestanya personalitzada.




Quan **Punt i coma de coma** es comprova, premeu **Entra** per acabar una línia s'insereix automàticament el punt i coma de finalització de la línia.

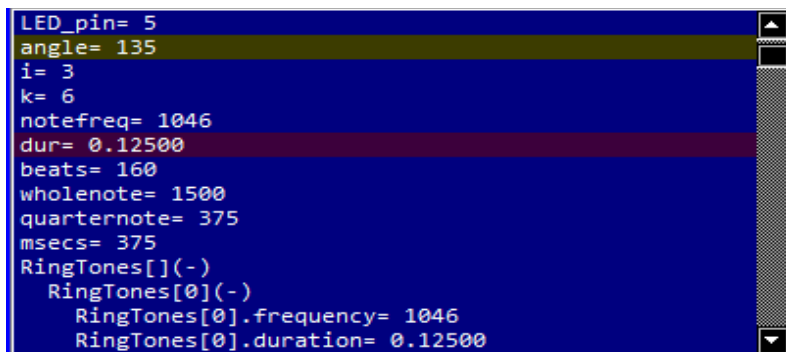
I per ajudar-vos a fer un millor seguiment dels vostres contextos i claus, feu clic a a ' { ' o ' } ' clau **ressalta tot el text entre clau i el seu soci de concordança** .

Panell de Variables i Editar/Monitoritzar Variable finestra

El **Panell de Variables** està situat a sota de la **Panell de Codi**. Mostra els valors actuals per a cada usuari local variable / matriu / objecte actiu (en àmbit) local i programa carregat. A mesura que el programa execució es mou entre funcionses, **el contingut canvia només per reflectir aquells variables locals accessibles al funció / àmbit actual, a més de tots els globals declarats per l'usuari**. Qualsevol variables declarat com 'const' o com 'PROGMEM'

(assignat a 'Flash' memòria) tenen valors que no es poden canviar i, per tant, per estalviar espai *no es mostra*. 'Servo' i 'SoftwareSerial' Les instàncies de objecte no contenen valors útils, així com també, no es visualitza

Tu pots **trobar** especificat **text** amb el **Trobar** ordres de cerca de text del menú (amb **Barra d'eines** botons  i  o dreceres de teclat **Fletxa cap amunt** i **fletxa cap avall**), després d'utilitzar primer **Trobar | Conjunt Cercar text** o  .



```
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
```

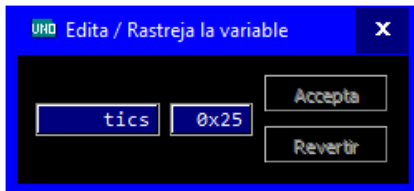
Matrius i objectes es mostren a qualsevol dels dos **un-expandit** o **expandit** format, amb un avantatge final ' (+) ' o menys ' (-) ' signe, respectivament. El símbol d'un matriu **x** mostra com '**x[]**'. Al expandir, per mostrar tots els elements del matriu, només cal que feu clic amb un sol clic '**x[] (+)**' a la **Panell de Variables**. Per tornar a encongir a la visualització d'un expandit, feu clic a '**x[] (-)**'. El valor predeterminat no expandit per a un objecte '**p1**' mostra com '**p1 (+)**'. A expandir s'hi mostraran tots els membres '**class**' o '**struct**' instància, amb un sol clic a '**p1 (+)**' a la **Panell de Variables**. Per tornar a encongir a la visualització que no és expandit, feu un clic '**p1 (-)**' .

Si tu **feu un clic sobre qualsevol línia de ressaltar per oliva fosca** (pot ser simple variable o el conjunt ' (+) ' o ' (-) ' una línia d'un matriu o objecte, o d'un únic element matriu o membre de objecte), després fent un **Executar fins a** farà que el següent execució es reprengui i es congeli **accés a escriptura** en qualsevol part de l'agregat seleccionat, o a la ubicació única del variable seleccionada.

Quan s'utilitza **Pas** o **Executar**, les actualitzacions dels valors mostrats per variable es realitzen segons la configuració de l'usuari que es realitza al menú **VarActualitzar** - això permet una gamma completa de comportaments des de les actualitzacions periòdiques mínimes fins a les actualitzacions immediates immediates. Les actualitzacions reduïdes o mínimes són útils per reduir la càrrega de la CPU i pot ser necessària per evitar que execució caigui enrere en temps real, en cas que fos excessiu **Panell de Variables** Carregues d'actualització finestra. Quan **Animació** està

en vigor o si el **Canvis de Ressaltar** l'opció de menú està seleccionada, canvia el valor d'un variable durant **Executar** donarà lloc al seu valor actualitzat **immediatament** , i es posa en relleu de color porpra: això farà que **Panell de Variables** per desplaçar-se (si cal) fins a la línia que conté variable i execució deixarà de ser en temps real.

Quan el execució es congela després **Pas** , **Executar cap a** , **Executar fins a** , o **Executar -augment- Aturar** , la **Panell de Variables** ressaltat *en morat* el variable corresponent al **ubicació (adreces) que es van modificar** (si n'hi ha) per part de **instrucció molt darrera** durant aquest execució (incloses les inicialitzacions de la declaració variable). Si aquesta instrucció **completament** omplint un **objecte o matriu** , la **línia parental (+) o (-)** per a aquest agregat es destaca Si, en canvi, la instrucció modificava a ubicació que actualment és visible i es posa de manifest. Però si actualment s'ubiquen les ubicacions modificades s'amaguen un expandit un matriu o objecte, que agregat **línia parental** obté una **Relleu de lletra cursiva** com a indicador visual en què es va escriure allò que hi havia al seu interior - fent clic a expandir llavors es produirà **últim** element modificat o membre per ressaltar.



El **Editar/Monitoritzar** finestra us ofereix **la possibilitat de seguir qualsevol valor de variable durant el execució** , o a **canvieu el seu valor al mig de (aturat) programa execució** (de manera que podeu provar quin seria l'efecte de continuar endavant amb aquest nou valor). **Aturar** execució primer, després **feu doble clic esquerre** al variable el valor del qual voleu fer el seguiment o el canvi. Per controlar simplement el valor durant el programa execució, **deixeu el quadre de diàleg obert** i després un dels **Executar** o **Pas** ordres: el seu valor

s'actualitzarà a **Editar/Monitoritzar** segons les mateixes regles que regulen les actualitzacions en el document de dades **Panell de Variables** . **Per canviar el valor variable** , empleneu el valor de la casella d'edició i **Acceptar** . Continuar execució (utilitzant qualsevol dels paràmetres **Pas** o **Executar** ordres) per utilitzar aquest nou valor des d'aquest punt (o podeu fer-ho) **Revertir** al valor anterior).

El programa Carregar o Reiniciar tingueu en compte que tot *el valor no inicialitzat-variables es restableix al valor 0 i tots els punters no inicialitzats-variables es restableixen a 0x0000*.

Panell de Proves

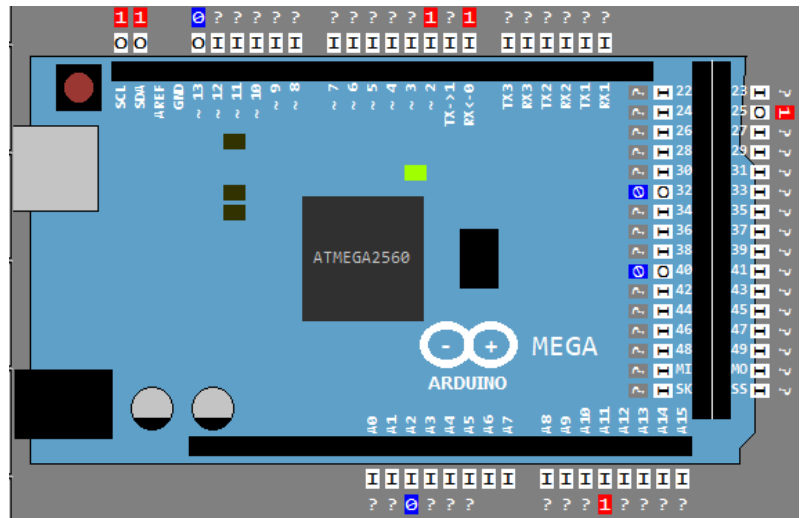
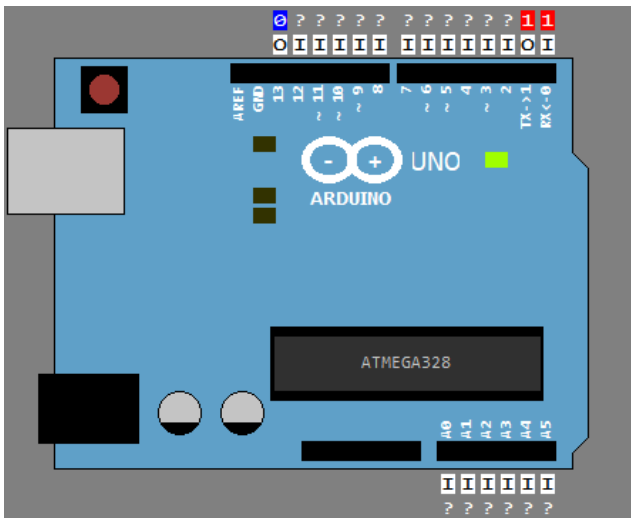
El Panell de proves mostra un 'Uno' de 5 volts o un K1077 'Mega' envoltat d'un conjunt de 'I/O' dispositius que podeu seleccionar / personalitzar i connectar-vos al vostre desitjat 'Uno' o 'Mega' contactes.

El 'Uno' o 'Mega'

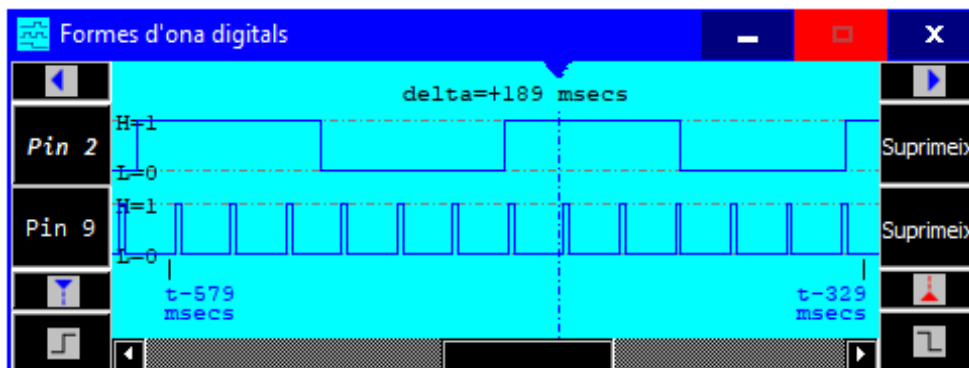
Aquesta és una representació del 'Uno' o 'Mega' placa i els seus LED incorporats. Quan carregueu un nou programa a UnoArduSim, si l'analitza amb èxit, se sotmet a un "baixar simulat" al placa que imita la manera com és real. placa es comporta: veureu que la sèrie RX i TX LED parpellegen (juntament amb l'activitat a contactes 1 i 0 que són *cablejat per a la comunicació en sèrie amb un ordinador amfitrió*). Tot seguit, es produeix un flaix contacte de 13 LED que significa un restabliment de placa i (i s'aturen automàticament UnoArduSim) l'inici del programa executió carregat. Podeu evitar aquesta visualització i el retard de càrrega associats deseleccionant-lo **Mostra el Baixar** des de **Configurar | Preferències**.

El finestra permet visualitzar els nivells de lògica digital en tots els 20 'Uno' contactes o els 70 'Mega' contactes ('1' en vermell per a 'HIGH' , '0' sobre blau per 'LOW' , i '?' en gris per a una tensió indeterminada no definida) i indicacions programat ('I' per 'INPUT' , o 'O' per 'OUTPUT'). Per a contactes que s'impulsa mitjançant PWM via 'analogWrite()' o bé 'tone()' o bé 'Servo.write()' , el color canvia a violeta i el símbol que es mostra esdevé '^' .









Tingues en compte que **Els Digital contactes 0 i 1 són de cable dur mitjançant resistències 1-KOhm al xip USB de comunicació en sèrie amb un equip host**.





Feu clic amb el botó esquerre en qualsevol 'Uno' o 'Mega' contacte s'obrirà un **Formes d'Ona Digitals** finestra que mostra el passat **un segon de valor de Activitat a nivell digital** en aquest contacte. Podeu fer clic en altres contactes per afegir-los a la pantalla Formes d'Ona Digitals (fins a un màxim de 4 formes d'ona alhora).



Feu clic per visualitzar la pàgina a l'esquerra o a la dreta o utilitzeu les tecles Inici, PgUp, PgDn, Fin

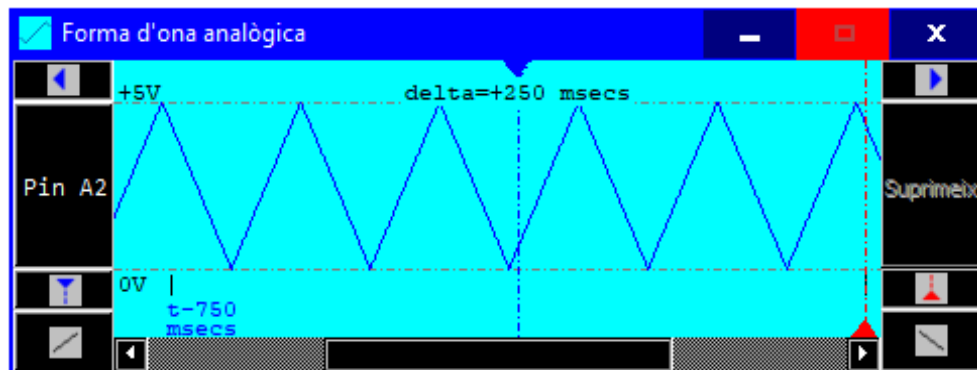
Una de les formes d'ona mostrades serà la **contacte actiu** forma d'ona, indicat amb el botó "Contacte" que es mostra com deprimat (com en la captura de pantalla Formes d'Ona Digitals anterior). Podeu seleccionar un forma d'ona fent clic al botó número Contacte i, a continuació, seleccioneu la polaritat de vora d'interès fent clic al botó de selecció de la polaritat de la vora o de la caiguda adequat, , o  o bé mitjançant les tecles de direcció **Fletxa cap amunt** i **fletxa cap avall**. Podeu llavors **saltar** el cursor actiu (bé les línies del cursor blau o vermell amb el temps delta mostrat) enrere o endavant fins a la vora de digital de polaritat escollida *d'aquest contacte actiu* forma d'ona mitjançant els botons del cursor, ,  o ,  (depenent del qual es va activar abans amb el cursor  o ) , o simplement utilitzeu les tecles del teclat \leftarrow i \rightarrow .

Per activar un cursor, feu clic al botó d'activació de colors ( o  mostrat anteriorment) - *aquest salt també desplaça la vista a la ubicació actual de aquell cursor*. Alternativament, podeu alternar ràpidament l'activació entre cursors (amb les vistes respectivament centrades) mitjançant la direcció **'Tab'** clau.







Tu pots **saltar** el cursor actualment activat per **feu clic esquerre a qualsevol lloc** a la regió de visualització forma d'ona que es mostra a la pantalla. També podeu seleccionar la línia de cursor vermella o blava fent clic dret a sobre (per activar-la) i, a continuació, *arrossegueu-lo a una ubicació nova*, i alliberar. Quan un cursor desitjat es troba en algun lloc fora de la pantalla, podeu fer-ho **feu clic dret en qualsevol part** per desplaçar-lo a la ubicació nova de la pantalla. Si els dos cursors ja són a la pantalla, feu clic amb el botó dret simplement alternarà entre el cursor activat.

Per seleccionar ZOOM IN i ZOOM OUT (el zoom sempre està centrat en el cursor ACTIVA), utilitzeu la roda del ratolí o les dreceres de teclat CTRL-up-arrow i CTRL-down-arrow.

En lloc d'una **clic dret en qualsevol 'Uno' o 'Mega' contacte** obre una **Formes d'Ona Analògiques** finestra que mostra el **passat un segon de valor** de **Activitat de nivell analògic** en aquest contacte. A diferència del Formes d'Ona Digitals finestra, podeu mostra l'activitat de analògic en un sol contacte alhora.



Feu clic per visualitzar la pàgina a l'esquerra o a la dreta o utilitzeu les tecles Inici, PgUp, PgDn, Fin

Tu pots **saltar** el/les línies de cursor blau o vermell al següent "punt de pendent" ascendent o baixant mitjançant els botons de fletxa endavant o endarrere (,  o , , de nou en funció del cursor activat, o bé utilitzeu el botó \leftarrow i \rightarrow tecles) en concert amb els botons de selecció del pendent ascendent / baixant ,  (el "punt de pendent" es produeix quan la tensió analògica passa pel llindar d'alt nivell digital ATmega contacte). De forma alternativa, podeu tornar a fer clic per saltar o arrossegar aquestes línies de cursors semblants al seu comportament al Formes d'Ona Digitals finestra

Prement **'Ctrl-S'** a dins *o finestra us permet desar el forma d'ona (X, Y) dades* al text arxiu que trieu, on **X** es troba a microsegons del costat esquerre i **Y** està en caràcters.

'I/O' Dispositius

Diversos dispositius envolten la 'Uno' o 'Mega' placa al perímetre de la **Panell de proves** . El dispositiu 'I/O' "petit" (del qual teniu permès fins a 16 en total) resideixen al costat esquerre i dret del Panell. 'I/O' dispositius "gran" (se li permet tants que encaixin) tenen elements "actius" i resideixen al llarg i a la part superior i inferior del **Panell de proves** . Es pot configurar el número desitjat de cada tipus de 'I/O' dispositiu disponible amb el menú **Configurar | 'I/O' Dispositius** .

Cada 'I/O' dispositiu té un o més fitxers adjunts contacte que es mostren com a **dos-dígit** Número contacte (00, 01, 02, ... 10,11,12, 13 i A0-A5 o 14-19, després d'això) en un quadre de edició corresponent. Per als números contacte del 2 al 9, simplement podeu introduir el dígit senzill; el 0 principal s'oferirà automàticament, però per a contactes 0 i 1 primer heu d'introduir els primers 0. Les entrades són *normalment* al costat esquerre d'un K1411 'I/O' i les sortides són *normalment* a la dreta (*espai permís*). Tot el 'I/O' dispositius respondrà directament als canvis de nivells de contacte i nivells de contacte, de manera que respondrà a qualsevol biblioteca funcionses dirigida al contactes adjunt o a programat '`digitalWrite()`' (per a l'operació "bit-banged").

Podeu connectar diversos dispositius al mateix ATmega contacte sempre que no es creï **elèctric problema**. Tal problema pot ser creat ja sigui per un 'Uno' o 'Mega' contacte com '**OUTPUT**' conduint contra una conducció forta (de baixa impedància) connectada dispositiu (per exemple, conduint contra una sortida 'PUSH' o una 'MOTOR' **Enc** de sortida), o per dos dispositius connectats que competeixen entre ells (per exemple, un botó 'PULSER' i un botó 'PUSH' connectat al mateix contacte). Qualsevol d'aquest problema seria desastrós en una implementació de maquinari real, per la qual cosa no se'ls permetrà l'ús de l'usuari a través d'un quadre de missatges emergent).

El quadre de diàleg es pot utilitzar per permetre a l'usuari triar els tipus i números del 'I/O' dispositius desitjat. Des d'aquest quadre de diàleg també podeu accedir-hi **Desar 'I/O' dispositius** a un text arxiu, i / o **Carregar 'I/O' dispositius** d'un text desat prèviament (o editat) arxiu (**incloses totes les connexions de contacte i la configuració de clic i tots els valors de la caixa d'edició tipificats**).

Tingues en compte que els valors de les caselles d'edició del període, del retard i de l'amplada de pols a l'I/O dispositius rellevant es poden donar el sufix 'S' (o 's'). Això indica que haurien de ser-ho **escalada** segons la posició d'un global '**I/O ____S**' control lliscant que apareix al Main finestra **Barra d'eines**. Amb aquest control lliscant completament cap a la dreta, el factor escala és 1.0 (unitat) i amb el control lliscant completament a l'esquerra el factor d'escala és 0,0 (subjecte a valors mínims aplicats per cada 'I/O' dispositiu particular). Podeu escalar més d'un valor de caixa de edició **simultàniament** fent servir aquest control lliscant. Aquesta característica us permet arrossegat el control lliscant durant l'execució per emular fàcilment els canvis d'amplada, els períodes i els retards dels pols per als que s'adjunten 'I/O' dispositius.

La resta d'aquesta secció proporciona descripcions per a cada tipus de dispositiu.

Alguns d'aquests dispositius **escalament suport del seu escrit a màquina en els valors** utilitzant el control lliscant del finestra principal **Barra d'eines**. Si el valor dispositiu té com a sufix la lletra 'S', el seu valor es multiplicarà per un factor d'escala (entre 0,0 i 1,0) que es determina per la posició del botó lliscant, subjecte a la restricció de valor mínim dispositiu. (1.0 es troba totalment a la dreta, 0.0 és a l'esquerra) --see '**I/O ____S**' sota cadascuna de mànegas dispositius detalla a continuació.



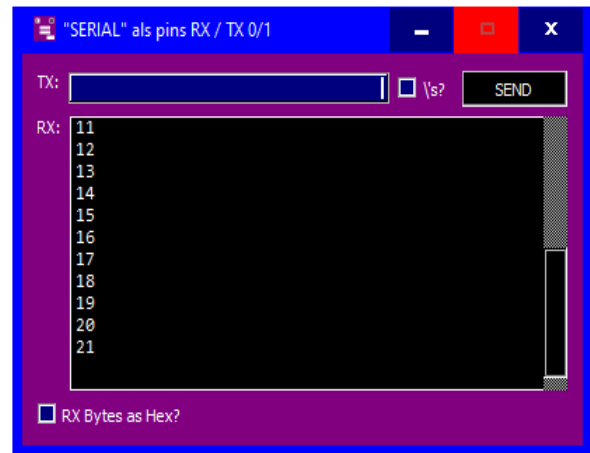
Monitor 'Serial' ('SERIAL')



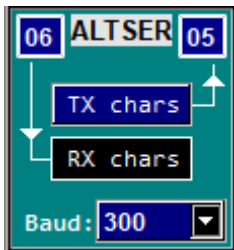
Aquest dispositiu 'I/O' permet l'entrada i sortida en sèrie de la ATmega mediada per maquinari (a través del xip USB 'Uno' o 'Mega') a contactes 0 i 1. El velocitat de transmissió es configura mitjançant la llista desplegable de la seva part inferior: el velocitat de transmissió seleccionat **ha de coincidir amb** el valor que el vostre programa passa a '`Serial.begin()`' funció per a una transmissió / recepció adequada. La comunicació en sèrie es fixa en 8 bits de dades, 1 bit d'aturada i cap bit de paritat. Teniu permís **desconnectar** (en blanc) **però no substituir** TX contacte 00, però no RX contacte 01.

Per enviar entrada del teclat al vostre programa, escriviu un o més caràcters a la part superior (caràcters TX), editeu finestra i, a continuació, colpejar el '**Enter**' tecla del teclat. (els caràcters es converteixen en cursiva per indicar que han començat les transmissions) o, si ja s'està avançant, els caràcters escrits seran en cursiva. A continuació, podeu fer servir el botó '`Serial.available()`' i '`Serial.read()`' funcions per llegir els caràcters en l'ordre en què van ser rebuts al buffer contacte 0 (el personatge mecanografiat més a l'esquerra s'enviarà primer). Es poden enviar impressions textuais i numèriques amb format o valors de bytes no formatats a la sortida de la consola inferior (caràcters RX) finestra trucant a l'Arduino '`print()`', '`println()`', o '`write()`' funcions.

A més, **es pot obrir un finestra més gran per configurar / visualitzar caràcters TX i RX fent doble clic (o fent clic dret) sobre aquest 'SERIAL' dispositiu**. Aquest nou finestra té un quadre d'edició de caràcters TX més gran i un botó 'Send' separat al qual es pot fer clic per enviar els caràcters TX als 'Uno' o 'Mega' (a contacte 0). També hi ha una opció de casella per reinterpretar seqüències de caràcters que es poden escapar de manera inversa com '`\n`' o '`\t`' per a la visualització no prima.



Sèrie Alternatiu ('ALTSER')

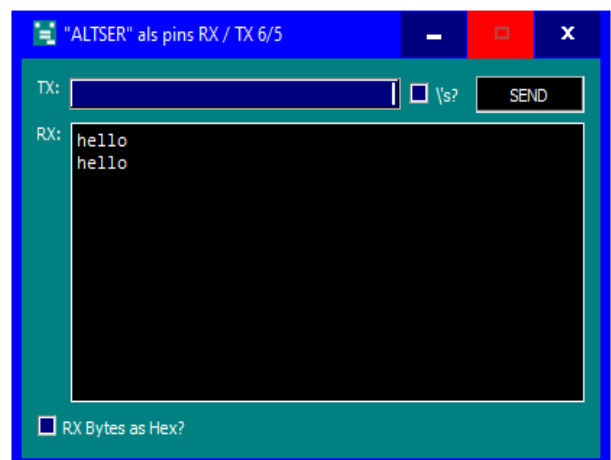


Aquest 'I/O' dispositiu permet la biblioteca o, alternativament, l'entrada i la sortida de l'usuari amb "bit-banged", en qualsevol parell de 'Uno' o 'Mega' contactes que trieu a omplir (**excepte per** contactes 0 i 1 dedicats al maquinari '`Serial`' comunicació). El vostre programa ha de tenir un '`#include <SoftwareSerial.h>`' línia que es troba a la part superior de la pàgina si voleu utilitzar la funcionalitat d'aquesta biblioteca. Igual que amb el 'SERIAL' dispositiu, el velocitat de transmissió per a 'ALTSER' es configura mitjançant la llista desplegable que hi ha a la part inferior: el velocitat de transmissió seleccionat ha de coincidir amb el valor que passa el programa al '`begin()`' funció per a una transmissió / recepció adequada. La comunicació en sèrie es fixa en 8 bits de dades, 1 bit d'aturada i cap bit de

paritat.

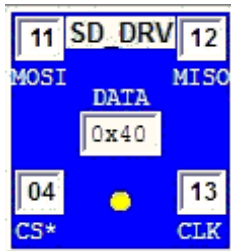
També, com passa 'SERIAL', **es pot obrir una configuració / visualització finestra més gran de TX i RX fent doble clic (o feu clic amb el botó dret) a l'ALTSER dispositiu**.

Tingueu en compte que, a diferència de la implementació de maquinari de '`Serial`', a '`SoftwareSerial`' no es proporciona Buffer TX suportat per operacions d'interrupció ATmega internes (només un buffer RX), així això '`write()`' (o '`print()`') les trucades es bloquegen (és a dir, el programa no es farà fins que no estiguin completades).



Disc SD ('SD_DRV')

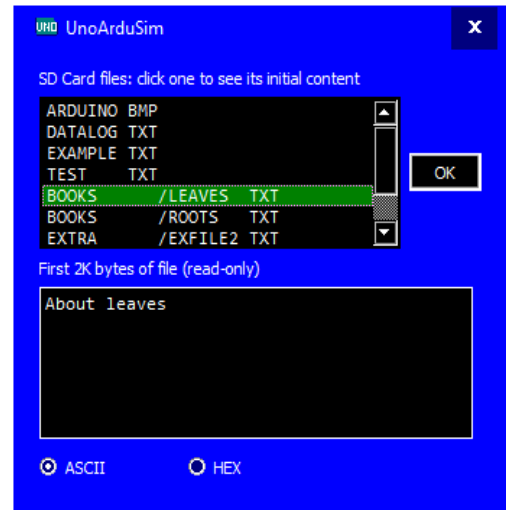
Aquest dispositiu 'I/O' permet la mediació de programari de biblioteques (però **no** "bit-banged") operacions d'entrada i sortida arxius al 'Uno' o 'Mega' **SPI** contactes (podeu triar-ne) **CS *** contacte utilitzaràs). El vostre programa pot simplement `#include <SD.h>` línia propera a la part superior, i podeu utilitzar `<SD.h>` Truca directament funcionses O directament `SdFile` funcionses tu mateix.



Es pot obrir un finestra més gran que mostri directoris i arxius (i contingut) fent doble clic (o fent clic dret) al 'SD_DRV' dispositiu. Tot el contingut del disc és carregat de un **SD** subdirector al directori programa carregat (si existeix) a `'SdVolume::init()'`, *i es reflecteix igual* **SD** subdirector al arxíu `'close()'`,

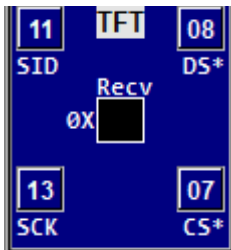
`'remove()'`, i en `'makeDir()'` i `'rmDir()'`.

Un LED groc parpadeja durant les transferències d'SPI i el 'DATA' mostra l'últim 'SD_DRV' **resposta** byte. Tots els senyals SPI són precisos i es poden visualitzar en un **Forma d'ona** finestra.



Pantalla TFT ('TFT')

Aquest 'I/O' dispositiu emula un Adafruit™ Pantalla TFT de 1280by-160 píxels (en la seva rotació nativa = 0, però quan s'utilitza el format TFT) **'TFT.h'** biblioteca, `'TFT.begin()'` conjunts d'inicialització per a la rotació = 1, que proporciona una vista "paisatgística" de 160 per 128 píxels). Podeu controlar aquest dispositiu trucant al funcionses de `'TFT.h'` biblioteca (que primer requereix `#include <TFT.h>`), o podeu utilitzar el sistema SPI per enviar-vos la pròpia seqüència de bytes a controlar.

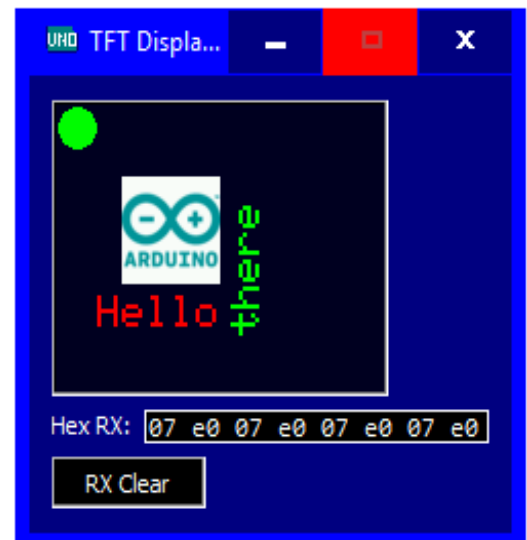


El TFT sempre està connectat a 'SPI' contactes 'MOSI' (per a 'SID') i 'SCK' (a 'SCK'): no es poden canviar. El 'DS*' contacte és per selecció de dades / comandes (el 'LOW' selecciona el mode de dades) i el 'CS*' contacte és el selecció de xip actiu-baix

No hi ha cap Reiniciar contacte proporcionat, de manera que no podeu fer un restabliment del maquinari aquest dispositiu conduint un contacte baix (com a `'TFT::begin()'` funció intenta fer quan s'ha aprovat una sèrie 'reset' contacte vàlida com a tercer paràmetre a `'TFT(int cs, int ds, int rst)'` constructor). Tanmateix, el dispositiu té una connexió

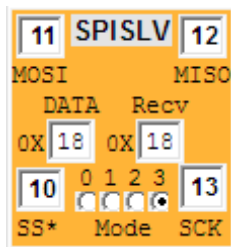
oculta a la línia Reiniciar del sistema, de manera que es torna a restablir; cada vegada que feu clic a la UnoArduSim principal La icona de la barra d'eines Reiniciar o el botó de restabliment 'Uno' o 'Mega' placa.

Per **feu doble clic** (o **clic dret**) en aquest dispositiu, s'obre un finestra més gran per mostrar aquesta pantalla LCD completa de 160 píxels a 128 píxels, juntament amb els 8 bytes més recents (com es mostra a continuació)



SPI configurable Esclau ('SPISLV')

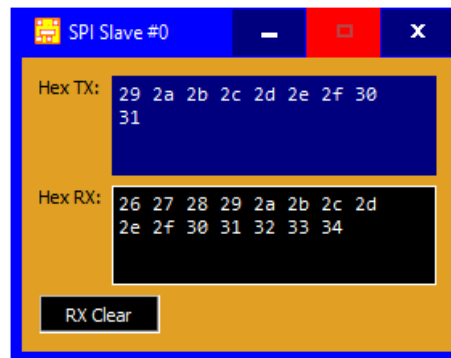
Aquest 'I/O' dispositiu emula un esclau SPI de mode escollit amb un baix actiu **SS *** ("selecció esclau") contacte controlant el **MISO** sortida contacte (quan **SS *** és alt, **MISO** no és controlat). El vostre programa ha de tenir un `'#include <SPI.h>'` línia si voleu utilitzar la funcionalitat de la biblioteca integrat SPI Arduino objecte i la biblioteca. També podeu triar la creació del vostre "bit-banged" **MOSI** i **SCK** senyala a controlar aquest dispositiu.



El dispositiu intueix les transicions de vora **CLK** entrada segons el mode seleccionat (`'MODE0'`, `'MODE1'`, `'MODE2'`, o `'MODE3'`), que s'ha d'escollir per combinar el mode SPI programat del programa.

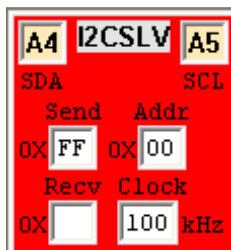
Fent doble clic (o fent clic amb el botó dret) al dispositiu podeu obrir un finestra acompanyant més gran que en canvi permet

y cal omplir un buffer de 32 bytes com a màxim (per emular SPI dispositius que retorna automàticament les seves dades) i veure els últims 32 bytes rebuts (tots com parells hex). **Tingues en compte que** el següent byte del buffer TX és s'envia automàticament només a `'DATA'` **després** un ple `'SPI.transfer()'` s'ha completat



I2C de dos fils Esclau ('I2CSLV')

Aquest 'I/O' dispositiu només emula a *mode esclau* dispositiu. Al dispositiu es pot assignar una adreça de bus I2C mitjançant una entrada de dos hex-dígit a la seva caixa de canvis 'Addr' (només respondrà aI2C transaccions d'autobús que impliquen la seva adreça assignada). El dispositiu envia i rep dades sobre el seu desguàs obert (només desplegable) **SDA** contacte i respon al senyal del rellotge del bus del seu desguàs obert (només desplegable) **SCL** contacte. Tot i que el 'Uno' o el 'Mega' serà el mestre del bus responsable de generar el **SCL** senyal, aquest esclau dispositiu també tira **SCL** baixa durant la seva fase baixa per estendre (si cal) el temps de bus baix a un adequat a la seva velocitat interna (que es pot configurar a la caixa de edició 'Clock').

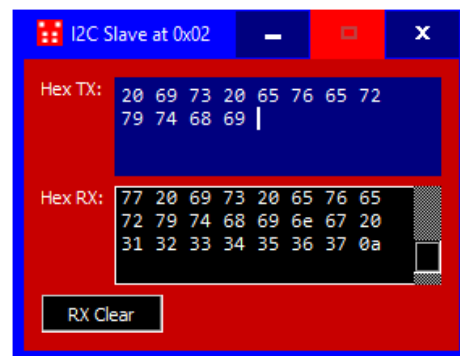


El vostre programa ha de tenir un `'#include <Wire.h>'` línia si voleu utilitzar la funcionalitat del directori `'TwoWire'` biblioteca per interactuar amb aquest dispositiu. Si us plau, podeu optar per crear les vostres dades i senyals de rellotge de pica-pica a controlar amb aquest esclau dispositiu.

Es pot configurar un únic byte per a la seva transmissió de tornada al mestre 'Uno' o 'Mega' a la caixa d'edició 'Send', i es pot veure un byte únic (el que s'ha rebut més recentment) a la caixa de edició 'Recv' (de només lectura). **Tingues en compte que** el valor de la caixa d'edició 'Send' sempre reflecteix el valor **Pròxim** byte per a la transmissió d'aquest buffer de dades

intern de dispositiu.

Fent doble clic (o fent clic amb el botó dret) al dispositiu podeu obrir un finestra acompanyant més gran que en canvi permet omplir un buffer FIFO de 32 bytes com a màxim (per emular TWI dispositius amb aquesta funcionalitat) i visualitzar (fins a un màxim de 32) bytes de les dades rebudes més recentment (com a dos) pantalla hex-dígit de 8 bytes per línia). El nombre de línies d'aquestes dues caselles d'edició correspon a la mida del buffer TWI escollida (que es pot seleccionar mitjançant **Configurar | Preferències**). Això s'ha afegit com a opció des de l'Arduino `'Wire.h'` usos de la biblioteca **cinc** aquests buffers RAM en el seu codi d'implementació, que és car per a la memòria RAM. Editant la instal·lació d'Arduino `'Wire.h'` arxiu per canviar la constant definida `'BUFFER_LENGTH'` (i també editant el company `'utility/twi.h'` arxiu per canviar la longitud del buffer TWI) per ser un usuari, en canvi, 16 o 8 *podria* reduir significativament la sobrecàrrega de la memòria RAM del 'Uno' o 'Mega' en una segmentació **implementació de maquinari** Per tant, UnoArduSim reflecteix aquesta possibilitat del món real a través **Configurar | Preferències** .



LCD de Text I2C ('LCDI2C')

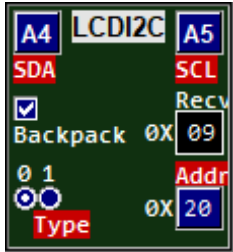
Aquest 'I/O' dispositiu emula una línia de 1,2, o4 de 4 LCD de caràcters, en un dels tres modes:

a) motxilla tipus 0 (expansor de port d'estil Adafruit amb maquinari amb adreça d'autobús I2C 0x20-0x27)

b) motxilla tipus 1 (expansor de port d'estil DFRobot amb dispositiu de maquinari) Adreça de bus I2C 0x20-0x27)

c) sense motxilla (interfície I2C integrada en mode nadiu amb adreça I2C bus 0x3C-0x3F)

El codi de biblioteca compatible per a cada mode dispositiu s'ha proporcionat al 'include_3rdParty' carpeta del vostre directori d'instal·lació UnoArduSIm: 'Adafruit_LiquidCrystal.h', 'DFRobot_LiquidCrystal.h', i 'Native_LiquidCrystal.h', respectivament.



El dispositiu pot assignar-li qualsevol adreça d'autobús I2C mitjançant una entrada de dos hex-dígit a la seva casella d'edició 'Addr' (només respondrà a I2C transaccions d'autobús que impliquen la seva adreça assignada). El dispositiu rep direcció de bus i de dades (i respon amb ACK = 0 o NAK = 1) en el seu desguàs obert (només desplegable) **SDA** contacte. Només podeu escriure ordres de LCD i dades de DDRAM: vosaltres **no pot** llegir dades de les ubicacions de DDRAM escrites ..



Feu doble clic o clic dret per obrir el monitor de pantalla LCD finestra des del qual també podeu definir la mida de la pantalla i el joc de caràcters.

LCD de Text SPI ('LCDSPI')

Aquest 'I/O' dispositiu emula una 1,2, o4 4 línies LCD de caràcters, en un dels dos modes:

a) motxilla (Expansor de port SPI estil Adafruit)

b) sense motxilla (mode integrat SPI integrat interfície: com a shown below)

El codi de biblioteca compatible per a cada mode dispositiu s'ha proporcionat a la 'include_3rdParty' carpeta del vostre directori d'instal·lació UnoArduSIm: 'Adafruit_LiquidCrystal.h', i 'Native_LiquidCrystal.h', respectivament.



El Contacte 'SID' té dades en sèrie, 'SS*' és la selecció dispositiu de baix nivell actiu, 'SCK' és el rellotge contacte, i 'RS' el contacte. Només podeu escriure ordres de LCD i dades de DDRAM (totes les transaccions SPI són escrites): vosaltres **no pot** llegir dades de les ubicacions de DDRAM escrites.

Feu doble clic o clic dret per obrir el monitor de pantalla LCD finestra des del qual també podeu definir la mida de la pantalla i el joc de caràcters.



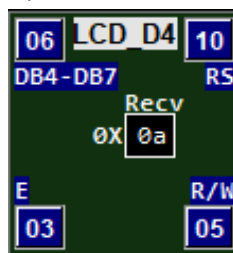
LCD de Text D4 ('LCD_D4')

Aquest 'I/O' dispositiu emula una 1,2, o4 4 línies LCD de caràcters amb una interfície de bus paral·lela de 4 bits. Els bytes de dades s'escriuen / es llegeixen **dues meitats** en les seves 4 dades contactes 'DB4-DB7' (on el quadre d'edició conté la secció *nombre mínim dels seus 4 números consecutius contacte*), - les dades es rellotgen en les cares caient del 'E' (habilitar) contacte, amb la direcció de les dades controlada pel 'R/W' contacte, i les dades LCD / mode de comandament del 'RS' contacte.

S'ha proporcionat codi de biblioteca de suport al directori 'include_3rdParty' carpeta del vostre directori d'instal·lació UnoArduSIm:

'Adafruit_LiquidCrystal.h', i

'Native_LiquidCrystal.h' tots dos treballen.

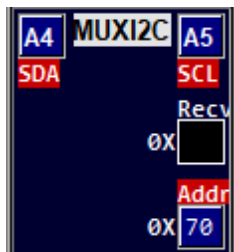


Feu doble clic o clic dret per obrir el monitor de pantalla LCD finestra des del qual també podeu definir la mida de la pantalla i el joc de caràcters.



Multiplexor LED I2C ('MUXI2C')

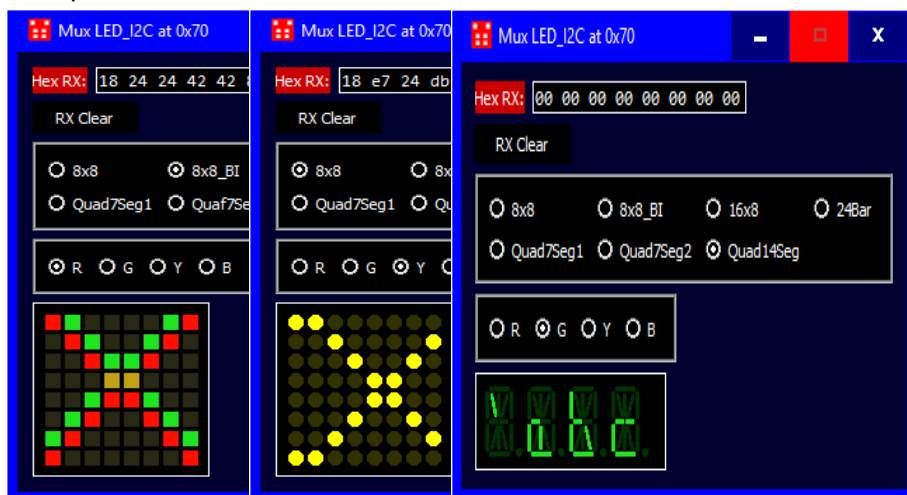
Aquest 'I/O' dispositiu emula un controlador HT16K33 interfacat amb I2C (h adreça d'autobús aviat I2C 0x70-0x77) a la qual Es pot adjuntar un dels diversos tipus de pantalles LED multiplexades



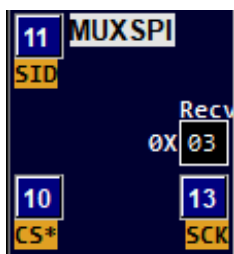
- a) 8x8, o 16x8, LED matriu
- b) 8x8 LED LED matriu
- c) 24-bi-color-bar LED
- d) dos estils de pantalles de 4 segments de 4 segments
- e) una pantalla alfanumèrica de 14 segments de 4 segments

Totes són compatibles amb el programa 'Adafruit_LEDBackpack.h' codi indicat a la 'include_3rdParty' carpeta:

Feu doble clic (o feu clic amb el botó dret) per obrir un finestra més gran per triar i veure un dels diversos LED de colors pantalles

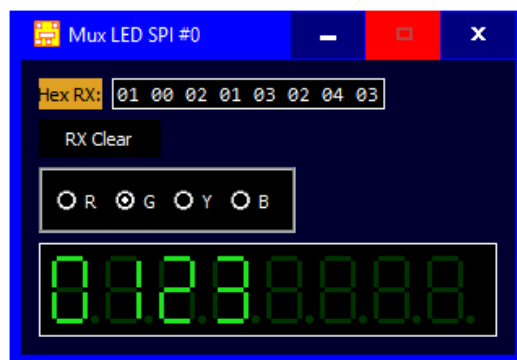


Multiplexer LED SPI ('MUXSPI')

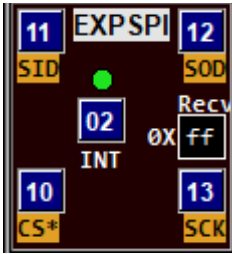


Un controlador multiplexat de LED basat en MAX6219 , amb suport 'MAX7219.h' codi indicat a la 'include_3rdParty' carpeta a controlar fins a vuit dígit de 7 segments.

Feu doble clic (o feu clic amb el botó dret) per obrir un finestra més gran t vista el color 8-dígit Pantalla de 7 segments.



Port d'Expansió SPI ('EXPSPi')

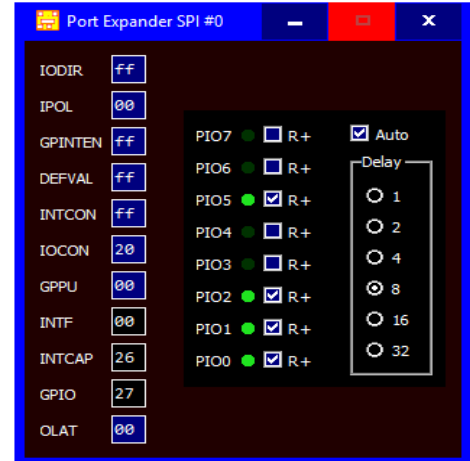


Un 8 bits expansor del port basat en el MCP23008, amb suport 'MCP23008.h' codi inclòs dins del 'include_3rdParty' carpeta. Podeu escriure als registres MCP23008 i llegir la GPIO contacte nivells. Es poden activar les interrupcions a cada canvi de GPIO contacte: una interrupció activada controlar el 'INT' contacte.

Feu doble clic (o feu clic amb el botó dret) obrir un finestra més gran veure el 8 línies de port GPIO i les resistències de traça adjuntes.

Podeu canviar els desplegaments manualment fent clic o bé adjuntar un comptador que els

canviarà periòdicament de manera contable. La velocitat a la qual s'incrementa el recompte és determinada pel retard de reducció factor escollit per l'usuari (un factor 1x correspon a un increment aproximadament cada 30 mil·lisegons; els factors de retard més elevats donen una taxa de recompte més lenta)

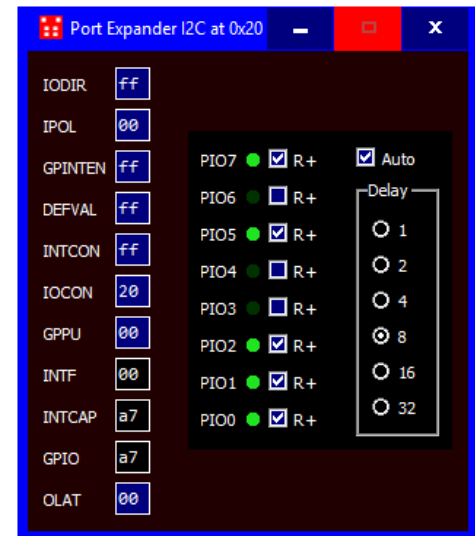


Port d'Expansió I2C ('EXPI2C')



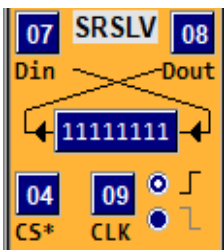
Un 8 bits expansor del port basat en el MCP23008, amb suport 'MCP23008.h' codi indicat a la 'include_3rdParty' carpeta. Les capacitats coincideixen amb el 'EXPSPi' dispositiu.

Feu doble clic (o feu clic amb el botó dret) per obrir un finestra més gran des del 'EXPSPi' dispositiu.



Registre de Desplaçament Esclau ('SRSLV')

Aquest dispositiu 'I/O' emula un registre de canvis simple dispositiu amb un baix actiu **SS*** ("selecció esclau") contacte controlant el '**Dout**' sortida contacte (quan **SS*** és alt, '**Dout**' no és controlat). El vostre programa podria utilitzar la funcionalitat de la biblioteca integrat SPI Arduino objecte i la biblioteca. També podeu triar la creació del vostre "bit-banged" '**Din**' i **CLK** senyala a controlar aquest dispositiu.



El dispositiu intueix les transicions de vora **CLK** entrada que desencadena el canvi del seu registre: la polaritat del sentit **CLK** es pot triar la vora mitjançant un control de botó de ràdio. en cada **CLK** edge (de la polaritat percebuda), el registre en capta el seu **Din** nivell a la posició de bit menys significativa (LSB) del registre de desplaçament, ja que els bits restants es desplacen simultàniament cap a l'esquerra cap a la posició MSB. Sempre que **SS*** és baix, el valor actual de la posició MSB del registre de canvis és controlat a '**Dout**'.

'1-Wire' Esclau ('OWISLV')

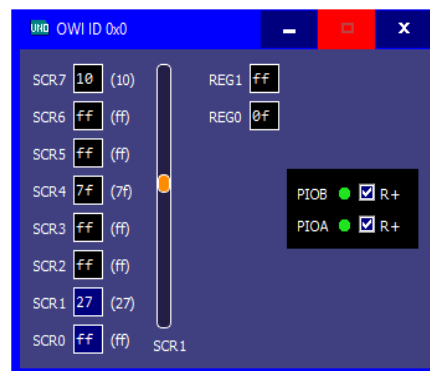
Aquest 'I/O' dispositiu emula un conjunt reduït de bus '1-Wire' dispositius connectat a contacte OWIO. Podeu crear un autobús '1-Wire' (amb un o més d'aquests esclaus '1-Wire' dispositius) al 'Uno' o 'Mega' contacte que trieu. Podeu utilitzar aquesta cabina dispositiu trucant al '**OneWire.h**' biblioteca funcionses després de col·locar un '**#include <OneWire.h>**' línia a la part superior del programa. Alternativament, també podeu fer servir senyals bit-bang a OWIO per a aquest dispositiu (tot i que és molt complicat de fer-ho correctament sense causar un problema elèctric - un problema encara és possible fins i tot quan es fa servir el '**OneWire.h**' funcionses, però el problema es mostra a UnoArduSim).



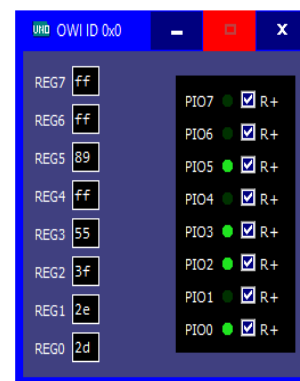
Cada OWISLV dispositiu del món real ha de tenir un número de sèrie intern de 8 bytes i (64-bit!) Únic: a UnoArduSim això és simplificat per l'usuari proporcionant un hexadecimal curt d'1 byte. '**ID**' el valor (que s'assigna seqüencialment per defecte a la càrrega / addició de dispositiu), més el '**Fam**' Codi de família per al dispositiu. UnoArduSim reconeix un petit conjunt de codis Family a partir de V2.3 (0x28, 0x29, 0x3A, 0x42) que cobreix un sensor de temperatura i IO (PIO) dispositius paral·lel (un codi de família no reconegut estableix el dispositiu per convertir-se en un raspador genèric de 8 bytes. dispositiu amb sensor genèric.

Si la família dispositiu no té registres PIO, registres **D0** i **D1** representen els primers dos bytes de scratchpad, en cas contrari representen el PIO Registre "estat" (nivells reals de contacte) i registre de dades de pestanyes PIO contacte, respectivament.

Per **feu doble clic** (o **clic dret**) al dispositiu, un nombre més gran **PROPIETARI** S'obre finestra. A partir d'aquesta finestra més gran, podeu inspeccionar tots els registres dispositiu, canviar les ubicacions del padró de scratch SCR0 i SCR1 mitjançant edicions i un control lliscant (de nou, SCR0 i SCR1 només corresponen a **D0** i **D1** si no hi ha PIO) o configureu els extrems PIO contacte externs. Quan s'editen SCR0 i SCR1, UnoArduSim recorda aquests valors editats com a "preferència" de l'usuari que representa un valor inicial (a partir de Reiniciar) que representa la sortida de valor signada del sensor dispositiu; s - el control lliscant es restableix al 100% (un factor d'escala d'1,0) en el moment de l'edició. Quan es desplaça el control lliscant posteriorment, el '**signed**' el valor en SCR1 es redueix en funció de la posició del control lliscant (factor d'escala d'1,0 a 0,0): la seva funció permet provar fàcilment la resposta del vostre programa per canviar suaument els valors del sensor. .



Per a un dispositiu amb PIO contactes, quan configureu les caselles de control del nivell de contacte, UnoArduSim recorda aquests valors marcats com els pull-ups actuals s'apliquen externament al contacte. Aquests valors externs s'utilitzen juntament amb les dades del pestell de contacte (registre **D1**) per determinar els nivells reals de contacte i il·luminar o apagar el LED verd connectat al PIO contacte (el contacte només va '**HIGH**' si s'aplica una externa de pull-up, i el corresponent **D1** el bit de pestanya és un '**1**').



Programable 'I/O' Dispositiu ('PROGIO')



Aquest 'I/O' dispositiu és en realitat un 'Uno' placa nu que podeu programar (amb un programa separat) per emular un 'I/O' dispositiu el comportament del qual es pot definir completament. Podeu triar fins a quatre contactes (IO1, IO2, IO3 i IO4) que aquest esclau 'Uno' compartirà en comú amb el mestre (main 'Uno' o 'Mega') que apareix al mig de la vostra **Panell de proves**. Com en qualsevol altre dispositiu, qualsevol problema elèctric entre aquest esclau 'Uno' i el mestre placa serà detectat i marcat. Tingueu en compte que totes les connexions ho són **directament** per cable, **excepte per contacte 13** (on s'assumeix una resistència R-1K de sèrie entre els dos contactes per tal d'evitar un problema elèctric a Reiniciar). A partir d'v2.8, la connexió entre el mestre i l'esclau contactes són **mapejats**: si el mestre també és un 'Uno', el mapatge predeterminat serà una identitat (tret que contacte 1 estigui mapejat a contacte 0 i viceversa per permetre comunicacions 'Serial'); si

el mestre és un 'Mega', els contactes 1 i 0 es tornen a posar i, **totals els SPI i TWI contactes són mapped** per a la connexió directa entre els subsistemes mestre i esclaus corresponents. Aquest mapa per defecte es pot substituir especificant en el vostre **IODevs.txt** arxiu és una llista explícita de 4 números de contacte mestre que segueixen el nom PROGIO programa arxiu: si algun d'aquests valors és -1, és el mateix que el mapatge predeterminat. Per a aquest dispositiu escriuiu els números de contacte **són els de l'esclau 'Uno'**. La imatge de l'esquerra mostra el 4 esclaus contactes especificat al seu sistema SPI contactes (SS *, MISO, MOSI, SCK) - independentment de si era amo un 'Uno' o un 'Mega', això li permetria programa aquest esclau com a esclau SPI genèric (o mestre) el comportament que podeu definir de manera programàtica.

Per **feu doble clic** (o **clic dret**) en aquest dispositiu, s'obre una finestra més gran per demostrar que aquest esclau 'Uno' té el seu propi **Panell de Codi** i associada **Panell de Variables**, tal com té el mestre. També té el seu **Barra d'eines**. Que podeu utilitzar **càrrega** i **controlar execució** d'un esclau programa: les accions de la icona tenen les mateixes dreceres del teclat que les de la finestra principal. (**Carregar** és **Ctrl-L**, **Desar** és **Ctrl-S** etc.) Un cop carregat, podeu executar des de **tampoc** el principal UnoArduSim finestra o des de dins d'aquest monitor Esclau finestra: en qualsevol cas, el programa i Esclau 'Uno' programa principals es mantenen bloquejats en la sincronització al pas del temps real a mesura que avancen les seves execucions. **Per triar el Panell de Codi que tindrà el càrrega, cerca i focus execució**, **feu clic** endavant **la seva barra de títol finestra principal: el Panell de Codi no centrat aleshores té la seva barra d'eines accions de color gris**.

A continuació es detallen algunes idees possibles per a l'esclau dispositius que podria ser programat en aquest 'PROGIO' dispositiu. Per a l'emulació en sèrie, I2C o SPI dispositiu, podeu utilitzar la codificació programa adequada amb matrius per a buffers d'enviament i recepció, a fi per emular comportaments complexos del dispositiu, **com ara GPS dispositius, EEPROM serial dispositius, etc.**

a) Un mestre o esclau SPI dispositiu. UnoArduSimV2.4.has estès el '**SPI.h**' biblioteca per permetre el funcionament del mode esclau S {PI mitjançant una opció '**mode**' paràmetre in '**SPI.begin(int mode = SPI_MASTR)**'. Passar explícitament '**SPI_SLAVE**' seleccionar el mode esclau (en lloc de confiar en el mode mestre predeterminat). Ara també podeu definir una interrupció de l'usuari funció (anomenem-ho '**onSPI**') en programa mestre o esclau per transferir bytes trucant una altra extensió afegida '**SPI.attachInterrupt(user_onSPI)**'. Ara calling '**rxbyte=SPI.transfer(tx_byte)**' des de dins seu '**user_onSPI**' funció esborrarà la bandera interrompuda i la farà **tornar immediatament** amb el byte acabat de rebre al vostre variable '**rxbyte**'. Alternativament, podeu evitar enganxar 1 SPI d'interrupció, i en lloc de trucar simplement '**rxbyte=SPI.transfer(tx_byte)**' des de dins del vostre programa principal, la trucada serà **bloc execució** fins que s'hagi transferit un byte SPI i llavors **tornar** amb el byte recent rebut al seu interior '**rxbyte**'.

b) Un genèric **serial 'I/O'** dispositiu. Podeu comunicar-vos amb el Master placa mitjançant qualsevol dels dos elements '**Serial**', o a '**SoftwareSerial**' definit al vostre esclau programa— per '**SoftwareSerial**' heu de definir '**txpin**' i '**rxpin**' oposadament als de Matser perquè l'esclau rebi el contacte sobre el qual transmet l'amo (i viceversa), però per **Només 'Serial'**, el 1 i el 0 contactes ja estan activats.

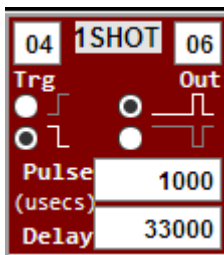
c) Un mestre o esclau genèric de 'I2C' dispositiu. Ara s'ha afegit l'operació Esclau per completar la implementació de UnoArduSim; s '**Wire.h**' biblioteca (funcionses '**begin(address)**', '**onReceive()**' i '**onRequest**' ara s'han implementat per tal de suportar les operacions en mode esclau).

d) Un digital genèric **Impulsor**. Utilitzant '**delayMicroseconds()**' i '**digitalWrite()**' trucades a l'interior '**loop()**' al vostre 'PROGIO' programa, podeu definir el '**HIGH**' i '**LOW**' intervals d'un tren de pols. Afegint un separat '**delay()**' truca al teu interior '**setup()**' funció, podeu retardar l'inici d'aquest tren de pols. Fins i tot podeu variar les amplades de pols a mesura que avança el temps mitjançant l'ús d'un comptador variable. També podeu utilitzar una persona per separat '**IOx**' contacte com a disparador per iniciar el temporització d'un K3611 emulat (o de doble tir, triple tret, etc.) dispositiu, i podeu controlar l'amplada de pols produïda per canviar de la manera que desitgeu a mesura que avança el temps.

e) Un senyalitzador aleatori. Això és una variació sobre a digital **Impulsor** que també utilitza trucades a '**random()**' i '**delayMicroseconds()**' generar temps aleatoris a què '**digitalWrite()**' un senyal en qualsevol contacte triat compartida amb el mestre. Utilitzant els quatre '**IOx**' contactes permetria quatre senyals simultànies (i úniques).

Un-Tret ('1SHOT')

Aquest 'I/O' dispositiu emula un digital d'una sola presa que pot generar un pols de polaritat escollida i amplada de pols a la seva 'Out' contacte, que es produeix després d'un retard especificat, d'un extrem desencadenant rebut a la seva **Trg** (activador) entrada contacte. Un cop rebut el límit d'actuació especificat, s'inicia temporització i després no es reconeixera un nou pols d'atenció fins que no s'hagi produït el pols 'Out' (i s'hagi acabat completament).



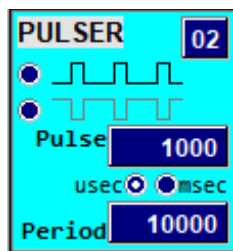
Un possible ús d'aquest dispositiu és simular Sensors d'ultrasons que generen un pols d'abast en resposta a un pols desencadenant. També es pot utilitzar allà on vulgueu generar un senyal d'entrada contacte sincronitzat (després del retard escollit) amb un senyal de sortida contacte creat pel vostre programa.

'Pulse' i els valors de 'Delay' es poden escalar a partir del finestra principal **Barra d'eines** Afegint el sufix un control lliscant de factor d'escala 'I/O ____ S' 'S' (o 's') a un (o als dos).

Un altre ús per a aquest dispositiu és provar un programa que utilitza interrupcions i voldríeu veure què passa si un **instrucció específica programa** s'interromp. Desconnecteu temporalment el 'I/O' Dispositiu que heu connectat a contacte 2 (o contacte 3) i substituïu-lo per un '1SHOT' dispositiu el 'Out' contacte del qual estigui connectat a contacte 2 (o contacte3, respectivament), i després podeu activar la seva entrada 'Trg'. (suposant que hi ha una sensibilitat d'avantguarda) inserint el parell d'instruccions { '**digitalWrite(LOW)**' , '**digitalWrite(HIGH)**' } **just anterior** a la instrucció a la qual es vol produir la interrupció. Establiu el 1SHOT; s 'Delay' amb el temps, el pols produït a 'Out' es produeix dins de la instrucció programa que segueix aquest parell d'instruccions de tret. Tingueu en compte que algunes instruccions de la màscara s'interrompen (com '**SoftwareSerial.write(byte)**' , aEn primer lloc no es pot interrompre.

Generador de Pols Digital ('PULSER')

Aquest 'I/O' dispositiu emula un generador simple digital pols forma d'ona que produeix un senyal periòdic que es pot aplicar a qualsevol 'Uno' o 'Mega' contacte escollit.



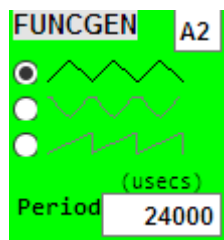
Les amplades de període i de pols (en microsegons) es poden configurar mitjançant quadres d'edició: el període mínim permès és de 50 microsegons i l'amplada mínima de pols de 10 microsegons. Podeu triar entre els valors de temporització en microsegons ('usec') i mil·lisegons ('msec'), i aquesta opció es guardarà juntament amb els altres valors quan feu 'Save' de **Configurar | I / O Dispositius**.

La polaritat també es pot triar: polsos d'avantguarda positius (0 a 5V) o polsos negatius d'avantguarda (5V a 0V).

Els valors de 'Pulse' i 'Period' es poden escalar de la manera principal **Barra d'eines** Control lliscant del factor d'escala 'I/O ____ S' afegint com a sufix 'S' (o 's') a un (o als dos). Això li permet a continuació, canvia el valor de 'Pulse' o 'Period' **dinàmicament** durant execució.

Analògic Generador de Funcions ('FUNCGEN')

Aquest 'I/O' dispositiu emula un generador simple analògic forma d'ona que produeix un senyal periòdic que es pot aplicar a qualsevol 'Uno' o 'Mega' contacte escollit.



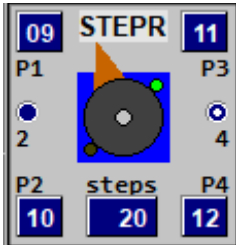
El període (en microsegons) es pot configurar mitjançant la casella d'edició: el període mínim permès és de 100 microsegons. El forma d'ona que crea es pot triar com a sinusoidal, triangular o dent de serra (per crear una ona quadrada, utilitzeu en lloc seu un 'PULSER'). En períodes menors, s'utilitzen menys mostres per cicle per modelar el forma d'ona produït (només 4 mostres per cicle en període = 100 microsegons).

La 'Period' el valor es pot escalar des del finestra principal **Barra d'eines** Control lliscant de factor d'escala 'I/O ____ S' afegint com a sufix la lletra 'S' (o 's'). Amb això, podreu canviar el

botó Valor de 'Period' **dinàmicament** durant execució.

Motor Pas a Pas ('STEPR')

Aquest 'I/O' dispositiu emula un Motor Pas a Pas bipolar o unipolar de 6V amb un controlador controlador integrat controlat de **dos** (endavant **P1** , **P2**) o **quatre** (endavant **P1** , **P2** , **P3** , **P4**) senyals de control. També es pot definir el nombre de passos per revolució. Podeu utilitzar el botó '**Stepper.h**' funcionses '**setSpeed()**' i '**step()**' a controlar el 'STEPR'. Com a alternativa, 'STEPR' ho farà *també respon* al vostre compte '**digitalWrite()**' " senyals de controlar amb interacció de bits



El motor es modelitza amb precisió tant mecànicament com elèctricament. Les caigudes de tensió del motor controlador i diverses reticències i inductàncies es modelen juntament amb un moment real d'inèrcia respecte al parell de retenció. L'enrotllament del rotor del motor té una resistència modelada de $R = 6$ ohms i una inductància de $L = 6$ milli-Henries que crea una constant de temps elèctrica d'1,0 mil·lisegons. A causa del modelisme realista, notareu que els polsos de control contacte són molt estrets *no reben* el motor a pas, tant a causa del temps de pujada del corrent finit com de l'efecte de la inèrcia del rotor. Això coincideix amb el que s'observa quan es condueix un motor pas a pas real d'un 'Uno' o 'Mega' amb, per descomptat, un adequat (*i requerit*) xip del motor controlador entre els cables del motor i el 'Uno' o 'Mega'!

Un desgraciat errada a l'Arduino '**Stepper.h**' El codi de biblioteca significa que al restablir el motor Stepper no estarà a la posició 1 de Pas (de quatre passos). Per superar-ho, l'usuari hauria d'utilitzar '**digitalWrite()**' en el seu / ella '**setup()**' rutina per inicialitzar els nivells de control de contacte a '**step(1)**' nivells adequats al control de 2-contacte (0,1) o 4-contacte (1,0,1,0), i permetre el motor ~~40~~ 100 mil·lisegons per desplaçar-se a la posició inicial del motor de referència a les dotze del migdia.

Tingues en compte que **La reducció d'engranatges no és compatible directament** per manca d'espai, però podeu emular-lo al vostre programa implementant un comptador modulo-N variable i només trucant '**step()**' quan el comptador colpeja 0 (per a la reducció del canvi per factor N).

AS de V2.6, aquest dispositiu ara inclou un 'sync' LED (verd per sincronitzada, o vermell quan fora a una o més etapes). A més, a més del nombre de passos per revolució, de forma opcional hi ha dos valors addicionals (ocults) especifiqueu-se a la IODEvs.txt arxiu per especificar la càrrega mecànica; per exemple, els valors 20, 50, 30 especifiquen 20 passos per revolució, un moment de inèrcia de càrrega 50 vegades més que del rotor del motor i un parell de càrrega del 30 per cent. de parell motor complet de retenció.

Impulsada Motor Pas a Pas ('PSTEPR')

Aquest 'I/O' dispositiu emula un 6V **micromecenatge** bipolar Motor Pas a Pas amb un controlador controlador integrat controlat per a **polsada 'Step'** contacte, un baix actiu '**EN***' (Activa) contacte, i una '**DIR**' (direcció) contacte . El nombre de passos complets per revolució també es pot configurar directament, juntament amb el nombre de micro-passos per pas complet (1,2,4,8 o 16). A més d'aquesta configuració, de forma opcional, hi ha dos valors addicionals (ocults) especifiqueu-se a l'IODev's, txt arxiu per especificar la càrrega mecànica; per exemple, els valors 20, 4, 50, 30 especifica 20 passos per revolució, 4 micro-passos per pas complet, un moment de càrrega d'inèrcia 50 vegades més que el motor. el rotor en si, i un parell de càrrega del 30% del parell de retenció del motor complet.

Heu d'escriure codi a controlar per al control contactes adequadament.

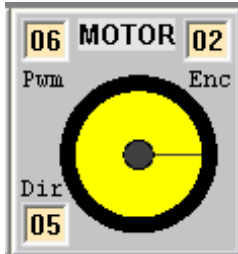


El motor es modelitza amb precisió tant mecànicament com elèctricament. Les caigudes de tensió del motor controlador i diverses reticències i inductàncies es modelen juntament amb un moment real d'inèrcia respecte al parell de retenció. L'enrotllament del rotor del motor té una resistència modelada de $R = 6$ ohms i una inductància de $L = 6$ milli-Henries que crea una constant de temps elèctrica d'1,0 mil·lisegons.

Aquest dispositiu inclou una LLK38 de color groc LED i una 'sync' LED (VERD per a sincronitzar o VERMELL quan està desactivat per un o diversos passos).

Motor CC ('MOTOR')

Aquest 'I/O' dispositiu emula un subministrament de 6 volts 100: 1 motor de corrent continu amb un controlador controlador controlat integrat per un senyal de modulació d'amplada de pols **Pwm** entrada), i un senyal de control de direcció (a la seva **Dir** entrada). El motor també té una sortida de codificador de roda que controla seva **Enc** sortida contacte. Pots fer servir '**analogWrite()**' a controlar el **Pwm** contacte amb un 490 Hz (en contactes 3,9,10,11) o 980 Hz (en contactes 5,6) PWM forma d'ona de cicle de treball entre 0,0 i 1,0 ('**analogWrite()**' valors 0 a 255). Com a alternativa, 'MOTOR' serà *també respon* al vostre compte '**digitalWrite()**' " senyals de controlar amb interacció de bits



El motor està modelat amb precisió tant mecànicament com elèctricament. Tenint en compte les caigudes de tensió (baixes) del transistor del conductor del motor MOS i el parell realista de l'engrenatge sense càrrega, proporciona una velocitat completa de gairebé 3 revolucions per segon i un parell de parada de poc menys de 10 kg-cm (que es produeix a un cicle de treball PWM constant de 1.0), amb una constant de temps mecànica d'aproximadament 40 mil·lisegons (que s'incrementa per la inèrcia de qualsevol càrrega especificada). Es poden especificar tres valors de paràmetres (opcionals) en un fitxer "IODevs.txt" de l'usuari: "F" o "B" (per al mode freewheel-coast o Brake quan Pwm és BAIX), a continuació, un parell de càrrega constant com a percentatge de parada de parell, després carregueu el moment d'inèrcia com

a múltiple enter de la inèrcia intrínseca del rotor del motor (aquests valors són per defecte "F", 0 i 1 si no n'hi havia cap especificat). A més, hi ha un parell de caixa de canvis oposat constant del 10% del parell de parada (que s'afegeix al parell de càrrega).

L'enrotllament del rotor del motor té una resistència modelada de $R = 2$ ohms i una inductància de $L = 300$ micro-Henries que crea una constant de temps elèctrica de 150 microsegons. A causa del modelisme realista, notareu que els polsos PWM molt estrets *no reben* el motor a girar, tant a causa del temps de pujada del corrent finit, com del temps apagat important després de cada pols estret. Es combinen per provocar un impuls insuficient del rotor per vèncer el revulsiu semblant a la molla de la caixa de canvis sota fricció estàtica. La conseqüència és quan s'utilitza '**analogWrite()**', un cicle de treball inferior a 0,125 no farà que el motor es posi en marxa, això coincideix amb el que s'observa quan es condueix un motor de canvi real d'un 'Uno' o 'Mega' amb, per descomptat, un adequat (*i requerit*) controlador mòdul de l'motor entre el motor i el 'Uno' o 'Mega' !.

El codificador del motor emulat és un sensor d'interrupció òptica muntat en eix que produeix un cicle de treball del 50% forma d'ona que té 8 períodes complets d'alta-baixa per rotació de rodes (de manera que el programa pot intuir canvis de rotació de les rodes fins a una resolució de 22,5 graus).

Servo-Motor ('SERVO')

Aquest 'I/O' dispositiu emula un motor servo DC de 6 volts PWM-controlat controlat en posició. Els paràmetres de modelatge mecànic i elèctric per al funcionament del servo coincidiran estretament amb els d'un servo estàndard HS-422. El servo té una velocitat de rotació màxima d'aproximadament 60 graus en 180 mil·lisegons . Si la a la part inferior esquerra la casella de selecció està marcada, el servidor passa a ser **rotació contínua** servo amb la mateixa velocitat màxima, però ara l'amplada de pols PWM estableix la **velocitat** més que l'angle



El vostre programa ha de tenir un '**#include <Servo.h>**' línia abans de declarar la vostra '**Servo**' instància (es *si trieu utilitzar la funcionalitat de la biblioteca 'Servo.h'* , per exemple '**Servo.write()**' , '**Servo.writeMicroseconds()**' Com a alternativa, 'SERVO' també respon '**digitalWrite()**' Senyals de "bit-banged" Degut a la implementació interna de UnoArduSim, estàs limitat a 6 'SERVO' dispositius.

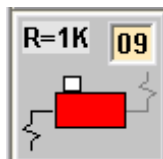
Altaveu Piezo-elèctric ('PIEZO')



Aquest dispositiu permet "escoltar" els senyals de qualsevol 'Uno' o 'Mega' contacte escollit i pot ser un complement útil per a LEDs per depurar el vostre funcionament programa. També podeu divertir-vos amb els tons de trucada adequadament `'tone()'` i `'delay()'` trucades (tot i que no hi ha cap filtrat del rectangle forma d'ona, de manera que no sentireu notes "pures").

També podeu escoltar un 'PULSER' o un 'FUNCEN' dispositiu connectats connectant un 'PIEZO' al contacte que el dispositiu controls.

Resistència de diapositives ('R=1K')



Aquest dispositiu permet a l'usuari connectar-se a una 'Uno' o 'Mega' contacte ja sigui a una resistència de tracció d'1 Ohm a + 5V o a una resistència desplegable d'1 Ohm a terra. Això permet simular les càrregues elèctriques afegides a un maquinari real dispositiu. Feu clic amb el botó esquerre sobre el commutador de diapositives **cos** podeu canviar la selecció desplegable o desplegable desitjada. Si utilitzeu un o diversos d'aquests dispositius, us podríeu establir un "codi" únic (o multi) -bit que us permeti llegir i respondre el vostre programa.

Polsador ('PUSH')

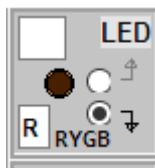


Aquest 'I/O' dispositiu emula un obert normalment **momentani o lliscador momentani** polsador d'un sol pol, d'un sol llançament (SPST) amb una resistència de desplegament (o desplegable) de 10 k-ohm. Si es tria una selecció de transició de punta a l'alça per al dispositiu, els contactes del botó es connectaran entre el dispositiu contacte i el + 5V, amb un desplegament de 10 k-Ohm a terra. Si es tria una transició d'avantguarda per al dispositiu, els contactes polsadors es connectaran entre el dispositiu contacte i el terra, amb un desplegament de 10 k-Ohm fins a + 5V.

Fent clic amb el botó esquerre al botó o prement qualsevol tecla, tanqueu el contacte del botó. Dins **momentània** mode, es manté tancat mentre manteniu premut el botó o el botó del ratolí i endavant **pestell** mode (habilitat fent clic al 'latch' botó) es manté tancat (i un color diferent) fins que torni a prémer el botó. Cada rebut es produirà el rebot en contacte (durant 1 milisegon) utilitzar el **barra**

d'espai per prémer el botó.

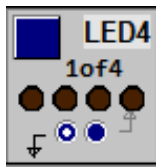
LED de color ('LED')



Podeu connectar un LED entre el 'Uno' o el 'Mega' contacte escollit (mitjançant una resistència de límit de corrent integrat de la sèrie oculta 1 k-Ohm) a terra o bé a + 5V, cosa que us permet triar el LED quan es connecti el 'Uno' o 'Mega' contacte és **'HIGH'** o, en canvi, quan ho sigui **'LOW'**.

El color LED es pot triar com a vermell ('R'), groc ('Y'), verd ('G') o blau ('B') mitjançant la seva caixa de canvis.

4-LED fila ('LED4')



Podeu connectar aquesta fila de 4 LED de colors entre el conjunt escollit de 'Uno' o 'Mega' contactes (cadascun té una resistència de límit de corrent de 1K-Ohm de sèrie integrat) a terra o a + 5V, cosa que us permet triar Els LED s'encenen quan hi ha el 'Uno' o 'Mega' contacte connectat 'HIGH' o, en canvi, quan ho sigui 'LOW'.

El '1of4' El quadre d'edició de contacte accepta un sol número contacte, que serà el que vol dir **el primer de quatre consecutius** 'Uno' o 'Mega' contactes que es connectaran als 4 LEDs.

El color LED ('R', 'Y', 'G' o 'B') és un **opció oculta** això pot ser **només serà escollit per edició de la IODevices.txt arxiu** (quin podeu crear utilitzant **Desar** de la **Configurar | 'I/O' Dispositius** Caixa de diàleg).

De 7 segments LED Dígit ('7SEG')



Podeu connectar aquesta pantalla de Dígit de 7 segments LED a una conjunt escollit de **quatre 'Uno' o 'Mega' contactes consecutius que donen el codi hexadecimal** per al dígit desitjat (de '0' a 'F') desitjat i activeu o desactiveu aquest dígit mitjançant el CS * contacte (actiu-LOW per a ON).

Aquest dispositiu inclou un descodificador que utilitza el integrat **actiu-ALT** nivells en els quatre consecutius '1of4' contactes per determinar el hexadecimal dígit sol·licitat que es mostrarà. Nivell te el número més baix de contacte (el que es mostra a la versió) '1of4' caixa d'edició) representa el bit menys significatiu del codi hexadecimal de 4 bits.

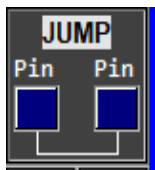
El color dels segments LED ('R', 'Y', 'G' o 'B') és un **opció oculta** això pot ser **només serà escollit per edició de la IODevices.txt arxiu** podeu crear utilitzant **Desar** de la **Configurar | 'I/O' Dispositius** Caixa de diàleg.

Control Analògic Lliscant

Un potenciòmetre de 0-5V controlat per control lliscant es pot connectar a qualsevol 'Uno' o 'Mega' contacte escollit per produir un nivell de tensió analògic estàtic (o canviant lentament) que es llegiria 'analogRead()' com un valor de 0 a 1023. L'ús d'el ratolí per arrossegar o fer clic per saltar, la barra lliscant analògic.



Connexió de Pont ('JUMP')



Pot connectar dos 'Uno' o 'Mega' contactes utilitzant aquest dispositiu (si es detecta qualsevol problema elèctric quan empleu el segon número de contacte, la connexió escollida no està permesa i que contacte està desconnectat).

Aquest pont dispositiu té una utilitat limitada i és més útil si es combina amb interrupcions, per a proves, experimentacions i programa finalitats d'aprenentatge. **A partir de UnoArduSim V2.4 és possible que utilitzeu un K511 'PROGIO' en lloc que ofereix més flexibilitat que els mètodes**

d'interrupció controlat a continuació.

Tres possibles usos per a aquest dispositiu són els següents:

1) Tu pots **crear una entrada digital per provar el vostre programa** que té un temporització més complex del que es pot produir mitjançant qualsevol conjunt de subministraments estàndard 'I/O' dispositius, de la següent manera:

Definiu una interrupció funció (anomenem-la 'myIntr') i fer 'attachInterrupt(0, myIntr, RISING)' dins del vostre 'setup()'. Connecta a **Impulsor** dispositiu a contacte2 - ara 'myIntr()' serà executar cada vegada a **Impulsor** es produeix la vora ascendent. La seva 'myIntr()' funció pot ser un algorisme que tingueu

programat (utilitzant el comptador global variables i potser fins i tot `random()`) per produir un forma d'ona del seu propi disseny en qualsevol disponible **OUTPUT** contacte (diguem que és contacte 9). Ara **SALT** contacte 9 al vostre 'Uno' o 'Mega' desitjat **INPUT** contacte per aplicar el digital forma d'ona a aquesta entrada contacte (per provar la resposta de la vostra programa; la resposta a aquell forma d'ona concret). . Podeu generar una seqüència de polsos o caràcters en sèrie o simplement transicions de vora, totes de complexitat arbitrària i intervals variats. Tingueu en compte que si el vostre programa principal truca `micros()` (o truca a qualsevol funció que depengui), seu `return` valor **s'incrementarà** pel temps dedicat al vostre interior `myIntr()` funció cada cop que s'interromp l'incendi. Podeu produir una ràpida ràfega de vores cronometrades amb precisió mitjançant trucades a `delayMicroseconds()` de dins `myIntr()` (potser per generar un tot **byte** d'un transferència elevada de velocitat de transmissió) o simplement genereu una transició per interrupció (potser per generar **una mica** d'una transferència baixa de velocitat de transmissió) amb la versió de velocitat de transmissió **Impulsor** dispositiu **'Period'** escollida adequada a les vostres necessitats temporització (recordeu-ho **Impulsor** limita el seu mínim **'Period'** a 50 microsegons).

2) Tu pots *experimenta amb respostes de bucles del subsistema:*

Per exemple, desconnecteu el vostre 'SERIAL' I/O dispositiu TX **'00'** contacte (editeu-lo en blanc) i, a continuació, **SALT** 'Uno' o 'Mega' contacte **'01'** tornar a 'Uno' o 'Mega' contacte **'00'** per emular un loop-back de maquinari de l'ATmega **'Serial'** subsistema. Ara a la prova programa, dins `setup()` fer un **solter** `Serial.print()` d'un paraula o personatge, i dins del teu `loop()` fer ressò dels personatges rebuts (quan `Serial.available()`) fent un `Serial.read()` seguit d'un `Serial.write()` i, a continuació, mireu què passa. Podríeu observar que és similar **'SoftwareSerial'** loop-back **fallarà** (com passaria a la vida real: el programari no pot fer dues coses alhora).

També podeu provar-ho **SPI** loop-back mitjançant un **SALT** a el tornar de connexió contacte 11 (MOSI) a contacte 12 (MISO).





3) Tu pots *comptar el nombre i / o mesurar l'espai entre transicions de nivells específics en qualsevol 'Uno' o 'Mega' sortida contacte X* que es produeixen com a resultat d'un complex Instrucció Arduino o biblioteca funció (com a exemples: `analogWrite()`, o `OneWire::reset()`, o `Servo::write()`), com segueix:

SALT contacte **X** interrompre contacte **2** i dins del teu `myIntr()` utilitzar a `digitalRead()` i a `micros()` anomenada, i compareu-los als nivells i els temps desats (d'interrupcions anteriors). Podeu canviar la sensibilitat de vora per a la següent interrupció, si cal, utilitzant `detachInterrupt()` i `attachInterrupt()` des de **a dins** la seva `myIntr()`. Tingueu en compte que no podreu fer el seguiment de contacte transicions que es produeixen massa a prop (més a prop del temps total de execució la seva `myIntr()` funció), com ara els que succeeixen amb transferències I2C o SPI, o amb velocitat de transmissió alt **'Serial'** transferències (tot i que la vostra interrupció funció no molestaria el temporització inter-edge d'aquestes transferències produïdes per maquinari). També tingueu en compte que les transferències mediades per programari (com `OneWire::write()` i `SoftwareSerial::write()`) són protegit deliberadament de la interrupció (el codi de la seva biblioteca desactiva temporalment totes les interrupcions, per tal d'evitar les interrupcions del temporització), de manera que no es poden mesurar a l'interior d'aquests que utilitzen aquest mètode.






Encara que podeu fer aquestes mateixes mesures d'espaiament de vora **visualment** en una **Les formes d'ona Digital** finestra, si esteu interessats en l'espai entre el mínim o el màxim d'un gran nombre de transicions, o en comptant transicions, fent-ho fent servir aquesta operació `myIntr()` -més- **SALT** la tècnica és més convenient. I pots mesura, per exemple, variacions en l'espai entre transicions produïdes per programa principal (a causa de l'efecte que el vostre programari pren diferents rutes de execució de diferents vegades execució), fer una mena de "perfilat" de programa.

Menús





Arxiu:

<u>Carregar INO o PDE Prog (ctrl-L)</u> 	Permet a l'usuari triar un programa arxiu amb l'extensió seleccionada. Al programa se li dóna immediatament un Analitzar
<u>Editar/Examinar (ctrl-E)</u>	Obre el programa carregat per a la seva visualització / edició.
<u>Desar</u> 	Desar ha editat el contingut de programa al programa original de arxiu.
<u>Desar com</u>	Desar ha editat el contingut de programa amb un nom diferent de arxiu.
<u>Pròxim ('#include')</u> 	Avança Panell de Codi per mostrar el següent '#include' arxiu
<u>Anterior</u> 	Retorna el Panell de Codi mostrar el arxiu anterior
<u>Sortida</u>	Surt UnoArduSim després de recordar a l'usuari per desar qualsevol arxiu (s) modificat (s).

Trobar:

<u>Pila de trucades ascendents</u> 	Vés a la trucada anterior funció del magatzem de crides: el fitxer Panell de Variables s'ajustarà per mostrar el variables local per a funció
<u>Descendeu la pila de trucades</u> 	Vés al següent funció anomenat al magatzem de crides: el fitxer Panell de Variables s'ajustarà per mostrar el variables local per a funció
<u>Definiu el text del Cercar (ctrl-F)</u> 	Activa el Barra d'eines Quadre d'edició del Trobar per definir el text que cal cercar al següent (i afegeix la primera paraula de la línia ressaltada actualment a la Panell de Codi o Panell de Variables si un d'aquests té la concentració).
<u>Trobar Text següent</u> 	Vés a la següent ocurrència de text a la secció Panell de Codi (si té el focus actiu) o a la següent aparició de text a la secció Panell de Variables (si en canvi, té el focus actiu).
<u>Trobar Text anterior</u> 	Saltar a l'ocurrència de text anterior al document Panell de Codi (si té l'enfocament actiu), o bé a l'ocurrència de text anterior al fitxer Panell de Variables (si en canvi, té el focus actiu).

Executar:

<u>Pas a (F4)</u>	Passos execució endavant amb una instrucció o <i>en un anomenat funció</i> .
<u>Pas sobre (F5)</u>	Passos execució endavant amb una instrucció o <i>mitjançant una trucada completa de funció</i> .
<u>Pas cap a fora (F6)</u>	Avança execució per <i>prou per deixar l'actual funció</i> .
<u>Executar cap a (F7)</u> 	Executa el programa, <i>aturant-se a la línia desitjada de programa</i> - primer heu de fer clic per ressaltar a la línia programa desitjada abans d'utilitzar Executar cap a.
<u>Executar fins a (F8)</u> 	Executa el programa fins que es produeixi una escriptura al variable que tenia el ressaltar actual al Panell de Variables (feu clic sobre un per establir el ressaltar inicial).
<u>Executar (F9)</u>	Executa el programa.
<u>Aturar (F10)</u> 	Altes programa execució (<i>i congela el temps</i>).
<u>Reiniciar</u> 	Restableix el programa (tot el valor variables es restableix al valor 0 i tots els punters-variables es restableixen a 0x0000).
<u>Animació</u>	Passa automàticament les línies consecutives de programa <i>amb més retard artificial</i> i ressaltar la línia de codi actual. Es perd i es produeixen sons de temps real.
<u>Càmera Lenta</u>	Redueix el temps en un factor de 10.

Opcions:

<u>Pas sobre Estructors/ Operadors</u>	Voleu cap a la dreta a través dels constructors, destructors i la sobrecàrrega dels operadors funcionses durant qualsevol pas (és a dir, no s'aturarà dins d'aquest funcionses).
<u>Registre-Assignació</u>	Assigna els locals de funció als registres ATmega de lliure en lloc de a la pila (genera un ús RAM bastant reduït).
<u>Error al no inicialitzar</u>	Marca com a error de Analitzar a qualsevol punt que el programa intenti utilitzar un variable sense haver inicialitzat el seu valor (o almenys un valor dins d'un matriu).
<u>Afegit 'loop()' 'Retard</u>	Afegeix 1000 microsegons de retard cada vegada 'loop()' es diu (en cas que no hi hagi altres trucades al programa 'delay()' en qualsevol lloc): útil per intentar evitar caure enrere en temps real.
<u>Permeten les interrupcions imbricades</u>	Permet tornar a habilitar amb 'interrupts()' des de dins una rutina del servei d'interrupció de l'usuari.

Configurar:

<u>I/O' Dispositius</u>	Obre un quadre de diàleg per permetre a l'usuari triar els tipus i els números del 'I/O' dispositius desitjat. Des d'aquest quadre de diàleg també podeu Desar 'I/O' dispositius a un text arxiu i / o Carregar 'I/O' dispositius a partir d'un text arxiu desat prèviament (o editat) (inclòs totes les connexions contacte i les configuracions de clic i els valors d'introducció)
<u>Preferències</u>	Obre un quadre de diàleg per permetre a l'usuari establir preferències, incloent el sagnat automàtic de les línies font programa, permetent la sintaxi Expert, l'elecció del tipus de lletra font de la lletra, optar per una mida de tipus de lletra més gran, fer complir els límits de matriu, permetre les paraules clau de l'operador lògic, mostrant programa baixar , elecció de la versió placa i longitud del buffer TWI (per a I2C dispositius).

VarActualitzar:

<u>Permet automàticament (-) Encongir</u>	Permetre UnoArduSim a encongir mostra expandit matrius / objectes a l'caure darrere en temps real.
<u>Mínim</u>	Només actualitzar la Panell de Variables mostrar 4 vegades per segon.
<u>Ressaltar Canvis</u>	Ressaltar va canviar els valors de variable quan s'executa (pot causar alentiment).

Finestres:

<u>Monitor 'Serial'</u>	Connecteu una E / S serial dispositiu a contactes 0 i 1 (si no n'hi ha) i obteniu una mida més gran ' Serial ' monitor de text TX / RX finestra.
<u>Restaura tot</u>	Restaureu tot el mínim de finestres infantil.
<u>Formes d'Ona Digitals</u>	Restaureu un Formes d'Ona Digitals finestra minimitzat.
<u>Formes d'Ona Analògiques</u>	Restaurar un minimitzat Formes d'Ona Analògiques finestra.

Ajuda:

<u>Ajuda ràpida Arxiu</u>	Obre el document UnoArduSim_QuickHelp PDF arxiu.
<u>Ajuda complet Arxiu</u>	Obre el UnoArduSim_FullHelp PDF arxiu.
<u>Correccions Errada</u>	Veure solucions importants del errada des del llançament anterior.
<u>Canvis / Millores</u>	Consulteu canvis i millores importants des del llançament anterior.
<u>Sobre</u>	Mostra la versió, drets d'autor.

'Uno' o 'Mega' Placa i 'I/O' Dispositius

Els 'Uno' o 'Mega' i els 'I/O' dispositius adjunts es modelen amb precisió elèctrica i podreu fer-vos una bona idea a casa vostra de com es comportarà el vostre programari amb el maquinari real, i tot el contacte problemes elèctric serà marcat.

Temporització

UnoArduSim executa prou ràpidament en un ordinador o tauleta que pugui (*en la majoria dels casos*) modelex les accions de programa en temps real, **però només si el vostre programa incorpora** almenys alguns petits 'delay()' trucades o altres trucades (com ara 'print()' o 'SPI.transfer()' etc.) que serà mantenir-lo sincronitzat de manera natural a temps real (vegeu més avall).

Per aconseguir-ho, UnoArduSim utilitza un temporitzador de devolució de trucades Finestres funció, que li permet fer un seguiment exacte en temps real. El executió d'un nombre d'instruccions de programa es simula durant una porció de temporitzador i instruccions que requereixen un executió més llarg (com les trucades a 'delay()') Pot ser que necessiteu utilitzar diverses llesques de temporitzador. Cada iteració del temporitzador de devolució de trucada funció corregeix el temps del sistema mitjançant el rellotge de maquinari del sistema de manera que programa executió s'ajusta constantment per mantenir el temps de bloqueig en temps real. *La única tarifa de vegades executió haver de caure enrere en temps real* és quan l'usuari ha creat llaços estrets **sense retard afegit** , o 'I/O' dispositius estan configurats per funcionar amb freqüències molt altes de 'I/O' dispositiu (i / o velocitat de transmissió) que generaran un nombre excessiu d'esdeveniments de canvi de nivell contacte i una sobrecàrrega de processament associada. Copes UnoArduSim amb aquesta sobrecàrrega per l'omissió d'alguns intervals de temporització per compensar, i això llavors s'alenteix la progressió a programa **per sota del temps real** .

A més, es mostra el programari amb un matriu gran o tornarà a tenir bucles estretes **sense retard afegit** pot provocar una freqüència de trucada funció alta i generar una alta **Panell de Variables** càrrega d'actualització de pantalla que fa que es quedi enrere en temps real: UnoArduSim redueix automàticament la freqüència de refresc de variable per intentar mantenir-se, però quan es necessiti encara més reducció, trieu **Mínim**, de la **VarActualitzar** menú per especificar només quatre actualitzacions per segon.

Modelar amb precisió el temps executió del sub-milisegon per a cada instrucció o operació de programa **no es fa** - només s'han adoptat estimacions molt aproximades per a la majoria amb finalitats de simulació. Tanmateix, el temporització de 'delay()' , i 'delayMicroseconds()' funcionses i funcionses 'millis()' i 'micros()' són perfectament exactes i **sempre que utilitzeu almenys un dels retards funcionses** en un llaç en algun lloc del vostre programa, **o** utilitzeu un funció que, naturalment, es vincula a operacions en temps real (com 'print()' doncs està lligat al velocitat de transmissió escollit) el rendiment simulat del vostre programa serà molt proper al temps real (de nou, exceptuant els esdeveniments de canvis de nivell contacte d'alta freqüència descaradament excessius o les actualitzacions excessives de Variables permeses per l'usuari que podrien alentir-lo).

Per veure l'efecte de les instruccions del programa individual w *gallina que corre* , pot ser desitjable poder alentir les coses. El menú es pot configurar un factor de desaceleració de 10 segons el menú **Executar** .

'I/O' Dispositiu Temporització

Aquests dispositius virtual rebre senyalització dels canvis que es produeixen en la seva entrada contactes en temps real, i produeixen corresponents sortides en la seva sortida contactes que després pot ser detectada pel 'Uno' o 'Mega' - que estan per tant inherentment sincronitzen amb programa executió. El 'I/O' intern dispositiu temporització el defineix l'usuari (per exemple mitjançant la selecció velocitat de transmissió o la freqüència de rellotge) i es configuren esdeveniments del simulador per fer un seguiment del funcionament intern en temps real.

Sons

Cada dispositiu 'PIEZO' produeix un so que correspon als canvis de nivell elèctrics que es produeixen en la contacte adjunt, independentment de la font d'aquests canvis. Per mantenir els sons sincronitzats amb programa executió, UnoArduSim s'inicia i s'atura la reproducció d'un buffer de so associat quan s'inicia / s'atura el executió.

A partir de V2.0, el so ara s'ha modificat per utilitzar l'API d'àudio Qt, malauradament la seva QAudioOutput 'class' no és compatible amb l'enllaç del buffer de so per evitar que es quedin sense mostres de so (com pot succeir durant

més temps dels retards operatius de la finestra del sistema operatiu). Per tant, per tal d'evitar la gran majoria dels clics so molest i interrupcions del so durant els retards de sistema operatiu, el so es silencia ara d'acord amb la següent regla:

El so està silenciats mentre UnoArduSim no sigui el finestra "actiu" (excepte quan s'ha creat i activat un nou fill finestra), **i fins i tot** quan UnoArduSim és el finestra principal "actiu", però el punter del ratolí és **fora** de seva principal àrea de client finestra.

Tingueu en compte que això implica que el so serà temporalment Va callar com a rei mentre el punter del ratolí passa **finestra sobre un nen**, i quedarà silenciats **si es fa clic a aquest nen finestra per activar-lo** (fins que es faci clic de nou a UnoArduSim finestra per tornar a activar-lo) .

El so sempre es pot silenciar fent clic a qualsevol lloc de l'àrea del client del finestra principal d'UnoArduSim.

A causa de el tampó, s Ond té un retard en temps real de fins a 250 mil·lisegons a partir del temps d'esdeveniment corresponent al contacte del 'PIEZO' adjunt.

Limitacions i elements no compatibles

Inclòs Arxius

A '< >': entre claudàtors '#include' de '<Servo.h>', '<Wire.h>', '<OneWire.h>', '<SoftwareSerial.h>', '<SPI.h>', '<EEPROM.h>' i '<SD.h>' és és compatible, però només s'emula; el arxius real no es busca; en canvi, la seva funcionalitat està directament "integrada en" UnoArduSim, i són vàlides per a la versió Arduino compatible.

Qualsevol citat '#include' (per exemple de "supp.ino", "myutil.cpp", o "mylib.h") és compatible, però tot arxius és obligatori **resideix a la mateix directori com a pare programa arxiu** això conté els seus '#include' (no hi ha cap cerca feta en altres directoris). El '#include' pot ser útil per minimitzar la quantitat de codi programa que es mostra a la versió **Panell de Codi** a la vegada Capçalera arxius amb '#include' (és a dir, els que tinguin una ".h" extensió) farà que el simulador intenti incloure el mateix nom arxiu que tingui un ".cpp" d'extensió (si també existeix al directori del pare programa).

Assignació de memòria dinàmica i RAM

operadors 'new' i 'delete' són compatibles, com són els originaris d'Arduino 'String' objectes, **però les trucades no directes a 'malloc()', 'realloc()' i 'free()' que confien en aquestes**

L'ús excessiu de RAM per a les declaracions de variable es marca en el temps Analitzar i el desbordament de memòria RAM es marca durant el programa execució. Un element del menú **Opcions** permet emular la assignació de registre normal ATmega com ho faria l'AVR compilador, o modelar un esquema de recopilació alternatiu que només utilitza la pila (com a opció de seguretat en cas que aparegui un errada al model de l'assignació de registres). Si es va a utilitzar un punter per veure contingut de la pila, ha de reflectir amb precisió el que apareixeria en una implementació de maquinari real.

'Flash' Assignacions de memòria

memòria 'Flash' 'byte', 'int' i 'float' El variables / matrius i el seu corresponent funcionses d'accés de lectura són compatibles. Cap 'F()' anomenada funció ('Flash'-macro) de qualsevol cadena literal és és compatible, però els únics funcionses d'accés directe amb cadena de memòria 'Flash' són compatibles 'strcpy_P()' i 'memcpy_P()', de manera que per utilitzar un altre funcionses, primer haureu de copiar la cadena 'Flash' a una RAM normal 'String' variable, i després treballar amb la memòria RAM 'String'. Quan utilitzeu el codi 'PROGMEM' paraula clau variable-modificador, ha d'aparèixer **Davant de** el nom de variable i el de variable **També ha de ser declarada** com 'const'.

'String' Variables

El nadiu 'String' biblioteca està gairebé completament compatible amb alguns molt (i menor) excepcions.

El 'String' operadors compatibles són +, + =, <, <=, >, > =, ==, !=, i []. Tingues en compte que: 'concat()' pren un **solter** argument que és el 'String', o 'char', o 'int' per afegir-se a l'original 'String' objecte, **no** dos arguments que s'exposen erròniament a les pàgines web de la referència d'Arduino).

Biblioteques Arduino

Només `'SoftwareSerial.h'`, `'SPI.h'`, `'Wire.h'`, `'OneWire.h'`, `'Servo.h'`, `'Stepper.h'`, `'SD.h'`, `'TFT.h'` i `'EEPROM.h'` per als **Arduino V1.8.8** l'alliberament s'admet actualment en UnoArduSimV2.6. Tractant de `'#include' el ".cpp" i ".h"` arxius d'altres que encara no són compatibles les biblioteques ***no funcionar*** ja que contindran instruccions de muntatge de baix nivell i directives no admeses i arxius no reconegut.

Punters

Tots els punters a tipus simples, matrius o objectes són compatibles. Un punter es pot equiparar a un matriu del mateix tipus (ex `'iptr = intarray'`), però hi hauria *no hi ha cap límit de matrius posterior* en una expressió com `'iptr[index]'`.

Funcionses pot tornar punters, o `'const'` apunts, però qualsevol nivell posterior de `'const'` al punter retornat s'ignora.

N'hi ha ***sense suport*** per a trucades de funció mitjançant ***punters declarats per l'usuari funció***.

'class' i 'struct' Objectes

Encara que es suporta el polimorfisme i l'herència (fins a qualsevol profunditat), a `'class'` o `'struct'` només pot ser definit a tenir com a màxim ***un*** base `'class'` (és a dir ***múltiples***-l'herència no és compatible). S'admeten les trucades d'inicialització del constructor Base- `'class'` (mitjançant notació de dos punts) a les línies de declaració del constructor, però ***no*** inicialitzacions de membres que utilitzen aquesta mateixa notació de dos punts. Això significa que el objectes que conté `'const'` no són compatibles amb el `'static'` variables o el tipus de referència variables (només són possibles amb les inicialitzacions de membres en temps de construcció especificats)

Les sobrecàrregues de l'operador d'assignació de còpia són compatibles amb els constructors de moviments i les assignacions de moviment, però no s'admeten les objecte de conversió definides per l'usuari ("tipus-repartició") funcionses.

Àmbit

No hi ha suport per a aquest programa `'using'` paraula clau, o per a `'namespace'`, o per `'file'` àmbit. Totes les declaracions no locals són presumptes per implementació globals.

Cap `'typedef'`, `'struct'`, o `'class'` definició (és a dir, es pot fer servir per a futures declaracions) ***global*** àmbit (***local*** Les definicions d'aquests ítems dins d'un funció no són compatibles).

Qualificadors 'unsigned', 'const', 'volatile', 'static'

El `'unsigned'` el prefix funciona en tots els contextos normals legals. El `'const'` quan s'utilitza, cal que la paraula clau ***precedit*** el nom variable o el nom de funció o `'typedef'` nom que s'està declarant: la seva col·locació després del nom provocarà un error de Analitzar. Per Declaracions de funció, només pot tenir el funcionses de retorn de punter `'const'` apareixen en la seva definició.

Tots UnoArduSim variables són `'volatile'` per implementació, així que `'volatile'` la paraula clau es ignora simplement a totes les declaracions de variable. Funcionses no poden ser declarades `'volatile'` ni tampoc són arguments de trucades de funció.

El `'static'` paraula clau està permès per variables normal, i per objecte membres i membre-funcionses, però no està permesa de manera explícita per a les instàncies objecte si mateixos (`'class'` / `'struct'`), Per no membre funcionses, i per a tots els arguments funció.

Directives Compilador

`'#include'` i regular `'#define'` estan tots dos suportats, però ***no macro*** `'#define'`. El `'#pragma'` directives i

directives d'inclusió condicional (`'#ifndef'` , `'#if'` , `'#endif'` , `'#else'` i `'#elif'`) també són ***no compatible*** . El `'#line'` , `'#error'` i macros predefinides (com `'_LINE_'` , `'_FILE_'` , `'_DATE_'` , i `'_TIME_'`) també són ***no compatible*** .

Elements de llenguatge arduino

Tots els elements originaris de la llengua Arduino són compatibles amb l'excepció del dubtós `'goto'` instrucció (l'únic ús raonable per a la qual puc pensar seria com un salt (a un bucle sense sortida) i en cas contrari en cas de error que el programa no pugui fer front.

C / C ++ - elements del llenguatge

Bit d'estalvi "qualificadors camp de bits" per als membres en les definicions d'estructura es ***no compatible*** .

`'union'` és ***no compatible*** .

L'oddball "operador de comes" és ***no compatible*** (així que no podeu realitzar diverses expressions separades per comes quan normalment només s'espera una sola expressió, per exemple a `'while()'` i `'for(; ;)'` constructes).

Plantilles Funció

El funcioness definit per l'usuari que utilitza la paraula clau "plantilla" per permetre-li acceptar arguments de tipus "genèric" ho són ***no compatible*** .

Emulació en temps real

Com s'ha apuntat anteriorment, hi ha diverses vegades execució de les instruccions de Arduino programa diferents ***no*** modelada amb precisió, de manera que per tal de funcionar a una velocitat en temps real de la seva programa necessitarà algun tipus de dominar `'delay()'` instrucció (almenys una vegada per dia `'loop()'`) o una instrucció que es sincronitza de manera natural amb canvis de nivell en temps real contacte (com, per exemple, `'pulseIn()'` , `'shiftIn()'` , `'Serial.read()'` , `'Serial.print()'` , `'Serial.flush()'` etc.)

Veure **Temporització** i **Sons** anterior per obtenir més detall sobre les limitacions.

Notes de llançament - a partir d'V2.5 en endavant

Correccions Errada

V2.8.2- Setembre 2020

- 1) '`analogRead()`' no estava acceptant '`A5`' com a número analògic contacte vàlid.
- 2) Des de la V2.6, els botons d'opció '`PULSER`' no sempre mostraven l'estat seleccionat (però funcionaven correctament).
- 3) Des de la V2.4, canviant de forma constant '`HIGH`' Nivell contacte a '`analogWrite(pin, 0)`' ha provocat la pèrdua del canvi de nivell 0.
- 4) Snce V2.8, execució (sense error), les finestres emergents de wrnning no han pogut ressaltar el problema code-lne.
- 5) Interrompre una sortida contacte periòdicament invertint amb '`digitalWrite()`' o bé '`digitalRead()`' ha causat la pèrdua de les tirades PWM anteriors en aquell contacte al widowow de Wavafeforms.
- 6) Problemes fixats amb la connexió de dos '`INPUT`' contactes amb un '`JUMP`' dispositiu: els canvis de 'I/O' dispositiu contacte ja no apareixen simplement al pont a contacte (ara apareixen als dos contactes) i adjunts '`PULSER`' o bé '`FUNCGEN`' ara els senyals periòdics s'estenen també a jumperd-to pn.
- 7) Tancament momentani '`PUSH`' Els dispositius no tornaven al seu estat obert quan el punter del ratolí va deixar el dispositiu.

V2.8.1- juny de 2020

- 8) Les línies en blanc addicionals eliminades per la preferència de format automàtic van fer que la línia inicialment ressaltada en Editar/Examinar es compensés a la baixa pel nombre de línies en blanc que es van eliminar per sobre.
- 9) '`analoRead()`' només acceptava el número complet de digital contacte.
- 10) Un 'class' que contenia sobrecàrregues de funció que difereixin únicament de l'atribut 'unsigned' d'un argument funció podria tenir una crida de sobrecàrrega equivocada funció. Això va afectar el '`print(char/int/long)`' LCD on es cridaria funció a la sobrecàrrega d'impressió de la base de base 'unsigned' i això va fer que el '`printFloat()`' LCD imprimís '46' en lloc del punt decimal '.'.
- 11) La inserció automàtica (després de 'Enter') d'un integrat ja emparellat no ha funcionat després de l'enllaçar enrere per corregir un error d'escriptura a la línia.
- 12) El dispositiu 'TFT' amb un 'RS' en blanc podria causar un bloqueig en l'execució.

V2.8.0- juny de 2020

- 1) Quan es va produir un avís de Analitzar (però no un error de Analitzar), V2.7 podria intentar ressaltar la línia problemàtica del buffer incorrecte (en el més recent '`#include`' al seu lloc), i això provocaria un bloqueig silencios (fins i tot abans que aparegui l'avís emergent) si el número de línia estava més enllà del final '`#include`' buffer
- 2) Els bytes rebuts al monitor finestres més gran de 'I2CSLV', 'SPISLV', 'TFT', 'LCDI2C' i 'LCDSPI' dispositius encara no es van informar correctament (això no va afectar el seu funcionament).
- 3) Variables de tipus '`unsigned char`' es van promoure al tipus '`int`' en expressions aritmètiques, però mal cura de la seva '`unsigned`' estat, que pot provocar un error '`unsigned`' resultant expressió.
- 4) Canviar (o embrutar) el contacte d'un 'LED4' o un '7SEG' dispositiu d'un paràmetre inicial de contacte vàlid no va poder desconnectar el seu contactes superior 3 i podria produir xocs inexplicables, a més, canvis realitzats en

V2.7 per harmonitzar el maneig de tot el 'LED' els tipus van trencar completament el 'LED4' dispositiu.

5) Un error en els canvis realitzats per a la Versió 2.6 per donar suport 'LOW' interrupcions causada 'FALLING' interrupcions units a contacte 3 per corrompre interrupció de nivell de canvi de detecció en contacte 2.

6) 'SoftwareSerial' es incapacitant incorrectament interrupcions d'usuari durant cada personatge que estava rebent.

7) Quan es va carregar un programes 'PROGIO' que contenia un error Analitzar, es va marcar l'error, però que el programa ja s'havia substituït per un 'PROGIO' programa per defecte dins del monitor 'PROGIO' finestra.

8) Des de la versió V2.4, contacte canvia en 1 Contacte i Contacte 0 van ser no van ser vists durant la descàrrega.

9) L'enllaç de les operacions de lectura o escriptura en un SD arxiu obert va fer que la seqüenciació de execució fallés, cosa que va provocar un eventual error UnoArduSim intern a causa de la profunditat de nivell àmbit.

10) L'intent de canviar el 'MISO' contacte en una 'SD_DRV' dispositiu corromput el MOSI contacte.

11) No s'han avisat totes les versions anteriors '`analogWrite()`' el contactes 9 o 10 es corrompia 'Servo' temporització per a tot el 'Servo' dispositius actiu.

12) Escriviu un número diferent a sobre d'un 'EN' contacte ja vàlid en un 'LCD_D4' dispositiu podria provocar un accident (un nombre parcialment mecanografiat tornaria -1 per al valor contacte).

V2.7- març 2020

1) Quan es va adoptar el tema del sistema operatiu lleuger (Finestres per defecte), el **Panell de Codi** no mostrava el color ressaltat introduït a V2.6 (en canvi, només un ressaltar gris resultant d'una anul·lació del sistema).

2) La versió 2.6 va trencar involuntàriament la format de pestanya de sagnat automàtic a la primera '`switch()`' construir.

3) Es mostrava la nova funció de navegació magatzem de crides introduïda a la versió 2.6 **valors incorrectes** per a locals variables quan no es troba dins de l'execució actual de funció i no s'ha pogut fer les trucades de funció al membre imbricat.

4) '`TFT::text()`' estava treballant, però '`TFT::print()`' funcionses no ho eren (simplement s'han bloquejat per sempre). A més, '`TFT::loadImage()`' va fallar si '`Serial.begin()`' que s'havia fet anteriorment (que és el cas normal, i ara és necessari).

5) La versió 2.6 va introduir un errada que mostrava el valor incorrecte dels bytes 'RX' actuals i passats per a 'I2CSLV', 'SPISLV', 'TFT', 'LCDI2C' and 'LDCSPI' dispositius (i el seu monitor finestres).

6) Es va produir un canvi en V2.4 **Executar | Animació** destacant per saltar-se de moltes línies de codi executat.

7) Des del V2.4, la desafiament de 'SS*' o 'CS*' en un 'I/O' dispositiu a la instrucció immediatament següent a '`SPI.transfer()`' faria que dispositiu a deixar de rebre el byte de dades transferides. A més, el byte recepció a '`SPI_MODE1`' i '`SPI_MODE3`' no es va marcar fins a l'inici del següent byte enviat pel mestre (i es va perdre completament l'octet si es deseleccionava el dispositiu 'CS*' abans d'aquell moment).

8) En el nou '`SPI_SLV`' mode permès des de V2.4, '`bval = SPI.transfer()`' només ha retornat el valor correcte per a '`bval`' si la transferència d'octets ja estava completa i espera quan '`transfer()`' es deia.

9) El quadre de 'DATA' edició el 'SPISLV' dispositius ara rep el valor per defecte 0xFF quan no hi ha més bytes per a respondre amb.

10) L'estat de sincronització LED era incorrecte per a 'PSTEPR' dispositius amb més d'un micro-pas per pas complet.

11) El problemes elèctric provocat per la reacció de 'I/O' dispositius a les transicions sobre el rellotge 'SPI', un senyal 'PWM' o un '`tone`' senyal, no es va informar i pot provocar una explicació (corrompuda) recepció de dades

12) Quan l'interval entre interrupcions era massa petit (menys de 250 microsegons), un canvi (defectuós) a V2.4 va alterar el temporització de integrat funcionses que utilitzen temporitzadors del sistema o bucles d'instrucció, per generar retards (alguns exemples són '`delay()`' i '`delayMicroseconds()`'). Un canvi posterior de V2.5 va provocar una alineació errònia de '`shiftOut()`' senyals de dades i rellotge quan es va produir una interrupció entre bits.

- 13) L'acceptació d'un text d'acabament automàtic de integrat funció mitjançant la tecla Enter no ha pogut eliminar els tipus de paràmetres del text de la trucada funció inserida.
- 14) La càrrega d'un nou (fàcil d'interrupció-controlat) programa quan un programa prèviament funcionant encara tenia una interrupció pendent podria causar un accident durant la descàrrega (a causa de la defectuosa intentat execució de la nova rutina d'interrupció).
- 15) Completacions automàtiques per a membres del Objecte (accessible mitjançant fletxa dreta 'ALT') per a objectes al seu interior '**#include**'. El arxius ja és accessible tan aviat com el seu '**#include**' arxiu és analitzat amb èxit.
- 16) Una declaració funció que tenia un espai inadvertit dins d'un nom de paràmetre funció va causar un missatge d'error Analitzar poc clar.
- 17) Quan el execució es va aturar en un mòdul diferent del programa principal, el Arxiu | L'acció anterior no s'ha pogut activar.
- 18) claus de cotització única ('{' i '}') encara es comptaven (incorrectament) com a claudàtors àmbit en els fitxers Analitzar, i també confonem el format amb guia automàtica de pestanyes.
- 19) '**OneWire::readBytes(byte* buf, int count)**' s'havia fallat en actualitzar immediatament la visualitzada '**buf**' contingut del Panell de Variables.
- 20) Octal-latch 'OWISLV' dispositius va mostrar els nivells de contacte de sortida que es van mantenir en un registre de pestanyes.

V2.6.0- | gener 2020

- 1) Un errada introduït a V2.3 va provocar un bloqueig quan es va utilitzar el botó Tancar afegit al fitxer **Trobar / Reemplazar** diàleg (més que no pas el seu **Sortida** botó de barra de títol).
- 2) Si un usuari programa ho va fer '**#include**' d'un altre usuari arxius, la **Desar** botó a l'interior **Editar/Examinar** Hauria fallat en guardar un arxiu modificat si hi hagués un error Analitzar o Execució existent marcat dins d'un arxiu diferent.
- 3) A **Cancel·lar** després d'una **Desar** també pot ser confús, per aquestes raons **Desar** i **Cancel·lar** les funcionalitats del botó s'han canviat (vegeu **Canvis i millores**).
- 4) Brackets desequilibrats dins de a '**class**' definició podria provocar un penjat des de V2.5.
- 5) les proves de lògica directe a '**long**' valors retornats '**false**' si no s'ha establert cap dels 16 bits més baixos.
- 6) UnoArduSim havia marcat un error quan es va declarar un punter variable com a llaç variable dins dels claudàtors de '**for()**' declaració.
- 7) UnoArduSim havia estat desestimant proves lògiques amb què es comparaven els apunts '**NULL**' o '**0**'.
- 8) UnoArduSim havia estat anul·lant l'aritmètica de punter que implicava un nombre enter variable (només es van permetre les constants enteres).
- 9) Interrupcions establertes amb '**attachInterrupt(pin, name_func, LOW)**' només s'havia intuït el dia a **transició** a '**LOW**'.
- 10) Quan s'utilitza més d'un 'I2CSLV' dispositiu, un esclau no adreçat podria interpretar les dades del bus posterior com a coincidències amb la seva adreça de bus (o de trucada global 0x00) i, per tant, senyalitzar falsament ACK, corrompre el nivell ACK del bus i penjant un '**requestFrom()**'.
- 11) Passant valor numèric '**0**' (o '**NULL**') Com un argument funció a un punter a una trucada funció ara es permet.
- 12) El nivell de sagnat del tabbing després de nidificar '**switch()**' construccions era massa superficial quan va triar el 'auto-indent formatting' **Configurar | Preferències** fou utilitzat.
- 13) La resta de dos punters compatibles ara produeix un tipus de '**int**'.
- 14) UnoArduSim esperava un constructor predeterminat definit per l'usuari per a un membre objecte, encara que no s'hagués declarat com a '**const**'.
- 15) En una pausa execució, la posició dibuixada d'un 'STEPR', 'SERVO' o 'MOTOR' El motor podria restar fins a

30 mil·lisegons de moviment darrere de la seva posició real calculada per última vegada.

16) '**Stepper::setSpeed(0)**' estava causant un accident a causa d'una divisió per zero.

17) Una sola línia '**if()**', '**for()**', i '**else**' les construccions ja no provoquen massa pestanyes d'indentació automàtica.

V2.5.0– oct 2019

1) Un errada introduït a V2.4 es va trencar la inicialització de la targeta 'SD' (va provocar un accident).

2) Utilitzant el subsistema 'SPI' a la nova '**SPI_SLV**' mode funcionat incorrectament '**SPI_MODE1**' i '**SPI_MODE3**'.

3) Les finestres emergents d'acabament automàtic (tal i com ho sol·licita 'ALT-right=arrow') es van solucionar per a funcionses amb paràmetres objecte; a la llista emergent ara també s'inclouen membres de la classe heretats (base) i, ara, també apareixen els complements automàtics '**Serial**'.

4) Des de V2.4 el valor de devolució de '**SPI.transfer()**' i '**SPI.transfer16()**' era incorrecta si una rutina d'interrupció de l'usuari acomiadat durant la transferència.

5) A V2.4, les formes d'ona periòdiques ràpides es van mostrar que tenien una durada més llarga que la seva durada real, evident quan es visualitzava a zoom superior.

6) El constructor '**File::File(SdFile &sdf, char *fname)**' no estava funcionant, per la qual '**File::openNextFile()**' (que confia en aquest constructor) tampoc no funcionava.

7) UnoArduSim va declarar erròniament un error de Analitzar a objecte-variables i objecte de retorn de funcionses, declarat com a '**static**'.

8) Declaracions d'assignació amb un '**Servo**', '**Stepper**', o '**OneWire**' objecte variable a la LHS, i un objecte-tornar funció o constructor en el RHS, causat un miscount interna de el nombre de objectes associada, que condueix a una eventual xocar.

9) Proves booleanes de tipus objectes '**File**' sempre tornaven '**true**' fins i tot si el arxiu no estava obert.

Canvis / Milliores

V2.8.2– Setembre 2020

- 1) Per evitar confusions, UnoArduSim ara ho permet '`< >`' claudàtors i cometes dobles per utilitzar indistintament al voltant del nom arxiu a '`#include`' declaracions tant per a l'usuari local com per a '3rdParty' arxius.
- 2) El '**MOTOR**' dispositiu ara pot acceptar tres valors ocults d'un fitxer '**IODevs.txt**' arxiu: 'F' o 'B' (Freewheel- (o mode de frenada), llavors el parell de càrrega constant com a percentatge del parell de parada, i després el moment de càrrega a inèrcia com a múltiple de la '**MOTOR**' inèrcia del rotor.
- 3) Ara OneArduSim accepta '`long int`' com a sinònim de '`long`'.
- 4) Ara, UnoArduSim emet un avís únic per utilitzar-lo '`volatile`' quan un variable global és modificat per una rutina d'interrupció de l'usuari.

V2.8.0- juny de 2020

- 5) S'ha afegit un nou 'Mega2560' placa **Preferències** opció (numerador Placa == 10). Aquesta característica molts més 'I/O' contactes, 4 interrupcions externes més (contactes 18, 19, 20, 21), tres més '**HardwareSerial**' ports (el contactes 22-27) i la memòria RAM va augmentar de 2 KBytes a 8 Kbytes ..
- 6) El 'SFTSER' dispositiu es va canviar de nom a 'ALTSER' (ja que ara també es pot utilitzar amb 'Serial1', 'Serial2' i 'Serial3').
- 7) Al fer clic a un quadre d'edició de dispositiu contacte ara se'ns selecciona el número sencer per facilitar-ne l'entrada, i fer clic dins d'un quadre d'edició de números de dispositius a Configurar | 'I/O' Dispositius passa igual.
- 8) S'ha afegit ressaltat taronja de la línia d'origen rellevant per a finestres emergents d'advertència Analitzar i Execució.
- 9) L'autoreformat automàtic ara elimina les línies en blanc que es produeixen després de les línies de paraules clau o que segueixen les línies en blanc anteriors.
- 10) S'ha afegit suport per a '`digitalPinToInterrupt()`' trucades
- 11) Ara les classes sempre reben un constructor predeterminat si no tenen un constructor especificat per l'usuari (fins i tot si no tenen una classe base, ni membres de objecte).

V2.7 març 2020

- 1) A més de la línia de codi actual (verda si està llesta per executar-se, vermell si error), UnoArduSim ara manté, per a cada mòdul, l'última línia de codi de navegació feta amb l'usuari o una pila (ressaltada amb un fons d'olivera fosc). és més fàcil establir i trobar línies de punt d'aturada temporals (ara es permet una per mòdul, però només hi ha una versió del 'Run-To').
- 2) S'ha afegit el nou 'I/O' dispositius (i és compatible amb el codi de la biblioteca de tercers), inclosos els 'SPI' i 'I2C' **Ports d'expansió**, i 'SPI' i 'I2C' **Multiplexor LED** Controladors i pantalles (LED matrius, 4-alfanumèrics, i 4-dígit o 8 dígit-displays de 7 segments).
- 3) '**Wire**' les operacions ja no es permeten les rutines d'interrupció de l'usuari (això suporta les interrupcions externes d'un Port d'Expansió 'I2C').
- 4) Les formes d'ona Digital ara mostren un nivell intermedi (entre '**HIGH**' i '**LOW**') Quan el contacte no està sent controlat.
- 5) Per evitar confusions en trepitjar un sol '`SPI.transfer()`' instruccions, UnoArduSim ara s'assegura que el 'I/O' dispositius adjunt ara rebrà el seu avantatge del rellotge 'SCK' (amb retard lògic) abans que torni el funció.
- 6) Quan l'acte-tab-format **Preferència** és activada, a l'escriure un tancament clau '`}`' a **Editar/Examinar** ara provoca un salt a la posició del signat de la pestanya de la seva obertura clau '`{`' soci.

- 7) A **Torna a formatar** el botó s'ha afegit a **Editar/Examinar** (per provocar un formatat immediat del sagnat automàtic) - aquest botó només està habilitat quan està activada la preferència de la sagnada automàtica.
- 8) Un missatge d'error més clar ara es produeix quan una paraula clau de prefix (com '**const**', '**unsigned**', o '**PROGMEM**') segueix un identificador en una declaració (ha de precedir l'identificador).
- 9) El variables inicial inicialitzat, fins i tot si no es va utilitzar mai més tard, ara se li assigna sempre una adreça de memòria, i així apareixerà visible.

V2.6.0 gener de 2019

- 1) Afegit **LCD de caràcters** mostrar dispositius tenir 'SPI', 'I2C', i la interfície de 4-bi-paral·lel. Donant suport codi font de la biblioteca s'ha afegit a la carpeta 'include_3rdParty' nova instal·lació (i es pot accedir mitjançant l'ús d'una normal de '**#include**' directiva): els usuaris poden triar alternativament escriure el seu propi funcionses al controlar al LCD dispositiu.
- 2) **Panell de Codi** la millora ha estat millorada, amb colors ressaltar separats per a una línia de codi preparada, per a una línia de codi d'error i per a qualsevol altra línia de codi.
- 3) El **Trobar** les accions 'func' del menú i la barra d'eines (anterior i posteriors de baixada) ja no salten a la línia inicial funció anterior / següent i, en canvi, ara ascendeixen (o descendir) el magatzem de crides, destacant la línia de codi pertinent a la persona que truca (o es diu) funció, respectivament, on el **Panell de Variables** el contingut s'ajusta per mostrar el variables per al funció que conté la línia de codis actualitzada.
- 4) Per evitar confusions, a **Desar** fet per dins **Editar/Examinar** provoca una immediata re-**Compilari** si la Desar va tenir èxit, utilitzant una subseqüent Cancel·lar o Sortida ara només revertir el text al text desat per última vegada.
- 5) Afegit a-entrada polsada Motor Pas a Pas ('PSTEPR') amb 'STEP' (pols), 'EN*' (activar), i entrades 'DIR' (direcció), i un ajust de micro-passos-per-etapa de (1,2,4,8, o 16) .
- 6) Tant el 'STEPR' com el 'PSTEPR' dispositius tenen ara un LED 'sync' (VERD per sincronitzar o VERMELL quan es desprèn un altre pas més.)
- 7) El 'PULSER' dispositius té ara una elecció entre microsegons i mil·lisegons per a 'Period' i 'Pulse'.
- 8) Les completacions automàtiques Integrat-funció ja no conserven el tipus de paràmetre davant del nom del paràmetre.
- 9) A el tornar a una anterior **Panell de Codi**, la seva línia anteriorment ressaltada ara es torna a ressaltar.
- 10) Com una ajuda per a l'establiment d'un punt de ruptura temporal, utilitzant Cancel·lar o de Sortida **Editar/Examinar** deixa el ressaltar a la web **Panell de Codi** a la línia visitada per última vegada pel cursor a **Editar/Examinar**.
- 11) Un usuari definit (o tercer) '**class**' ara es pot utilitzar '**Print**' o '**Stream**' com la seva classe base. Com a suport d'això, s'ha afegit una nova carpeta 'include_Sys' (a la carpeta d'instal·lació UnoArduSim) que proporciona el codi font de cada base '**class**'. En aquest cas, truca a aquesta base- '**class**'. El funcionses es tractarà de manera idèntica amb un codi d'usuari (que es pugui incorporar), en lloc d'un integrat funció que no es pot afegir (com ara '**Serial.print()**').
- 12) Les completacions automàtiques del membre-funció ara inclouen el nom del paràmetre **en lloc de** el seu tipus.
- 13) L'UnoArduSim Analitzar ara permet que un nom objecte d'una declaració variable es prefereixi amb la seva opció (i que coincideixi) '**struct**' o paraula clau 'class', seguida de la '**struct**' o '**class**' nom.

V2.5.0 oct. 2019

- 1) S'ha afegit suport per a '**TFT.h**' biblioteca (a excepció de '**drawBitmap()**'), i va afegir un 'TFT' associat 'I/O' Dispositiu (128 per 160 píxels). Tingueu en compte que per evitar retards excessius en temps real durant les grans

'fillXXX()'
'SPI' transferència, porció sa de les transferències 'SPI' a la meitat del farcit estarà absent de l'autobús 'SPI'.

2) Durant grans transferències de arxiu mitjançant 'SD', a una part de les transferències de 'SPI' al centre de la seqüència d'octets no seran igualment absents del bus 'SPI'.

3) Disminució 'Stream' -utilitzar els bytes de sobrecàrrega perquè 'RAM free' el valor coincideix més amb la recopilació Arduino.

4) UnoArduSim ara avisa l'usuari quan una 'class' té diversos membres van declarar en una línia de declaració.

5) Utilitzant 'File | Save As' ara defineix el directori actual que el directori desat.

6) Els dos desapareguts 'remove()' el membre funcionses s'ha afegit al directori 'String' classe.

7) UnoArduSim ara desactiva les trucades bàsiques del constructor en un constructor-funció prototip tret que la definició completa del cos funció siga immediatament (de manera que coincideixi amb l'Arduino compilador).

8) Edtemps de transició ge Les formes d'ona digital es van reduir per donar suport a la visualització de senyals 'SPI' ràpids al màxim zoom.

9) UnoArduSim permet ara declarar alguns constructors 'private' o 'protected' (per a ús de classe interna)