

# UnoArduSimV2.7 Quick Help

The screenshot displays the UnoArduSim V2.7 software interface. The top menu bar includes File, Find, Execute, Options, Configure, VarRefresh, Windows, and Help. The title bar shows the file path [C:/Users/Stan/Documents/Qt/UNOTests/IOTest/DemoProg1.ino] and a status bar on the right indicates 'I/O' Value Multiply by 0.0<S<=1.0.

The main workspace is divided into several panes:

- Code Pane:** Contains the C++ code for the simulation, including comments and function definitions like `servo1.write(angle);`, `stepper1.step(10);`, and `void wheelTic()`.
- Lab Bench Pane:** Features a central diagram of an Arduino Uno board with an ATMEGA328 microcontroller. Surrounding the board are various peripheral components: two push buttons (01, 05), three 1K resistors (09), two piezo buzzers (08, 03), a motor (06), a stepper motor (03), a pulser (06), a serial monitor (01), a servo (09), and seven LEDs (02-07). Each component has a corresponding control panel with parameters like pulse width, period, and mode.
- Variables Pane:** Located at the bottom left, it displays the current values of variables defined in the code, such as `backval`, `count`, `tics`, `digital_level`, `analog_level`, `numchars`, and `angle`.

The bottom status bar shows a 'Fly-over Hint' and a message: 'REACHED A Run Temporary Breakpoint'.

Toolbar fly-over Hints

Status Bar

## Code Pane:







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



int count;




void setup()
{
  count=0;
}



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

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

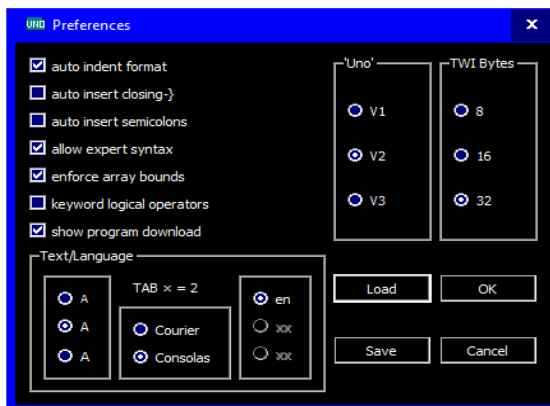
Step or Run using , , , or . To Halt at a specific program line , first click to highlight that line, and then click **Run-To** . To **Halt when a specific variable is written to**, first click on it to highlight it, and then click **Run-Till** .

Navigate the call-stack using  and , or jump between functions by clicking anywhere, then use **PgDn** and **PgUp**.

Set search text with , and then jump to that text using  and .

Move between '#include' files using  .

## Preferences:



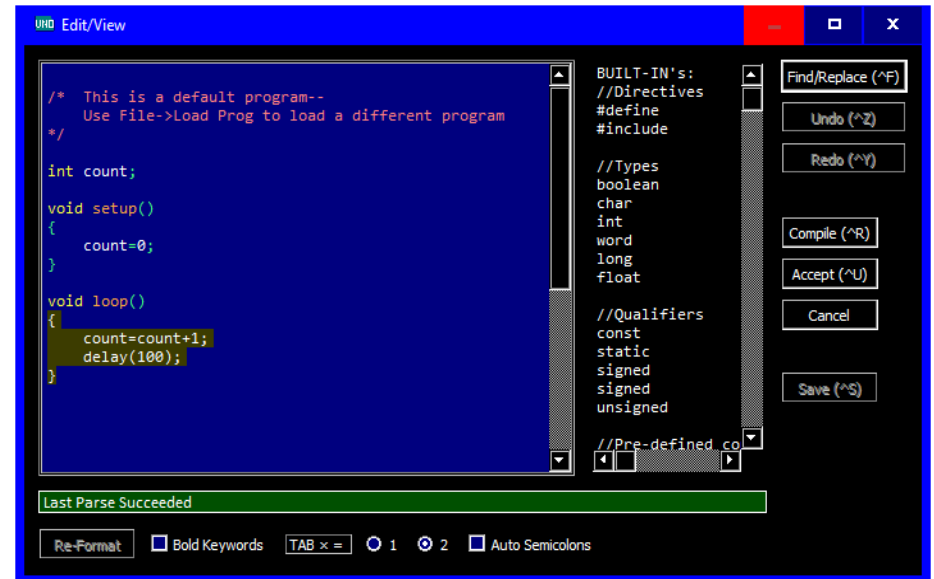
**Configure | Prereferences** to set, save ,and load user choices.

Alternate language(s) set by the user locale, and by a two-letter code on the very first line of the **myArduPrefs.txt** Preferences file

## Edit/View:

To open at a specific line, **double-click** on that line n the **Code Pane** or use **File | Edit/View** (and it opens at the last highlighted line)

Tab-indentation will be automatically done if that preference is chosen from **Configure | Prereferences** – you can also single or double-size the Tab width.



Add or delete tabs to a group of lines using **right-arrow** or **TAB**, and **left-arrow** (after first selecting a group of 2 or more consecutive lines).

To add an item (after the caret) from the right-hand list of **Built-ins**, double click on it .

**Find** (use ctrl-F), **Find/Replace** (use ctrl-H), **Undo** (ctrl-Z), **Redo** (ctrl-Y)

Use **ALT-right-arrow** to request auto-completion choices for built-in **global variables**, and for **member variables** and **functions**.

**Compile** and leave open (ctrl-R), or **Accept** (ctrl-U) or **Save** (ctrl-S) to close.

Find a **matching brace-pair** partner by double-clicking on it – both braces, plus all text between, become highlighted (as in the image above).

Use **ctrl-PgDn** and **ctrl-PgUp** to jump to next (or previous) empty-line break.

## Variables Pane:

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

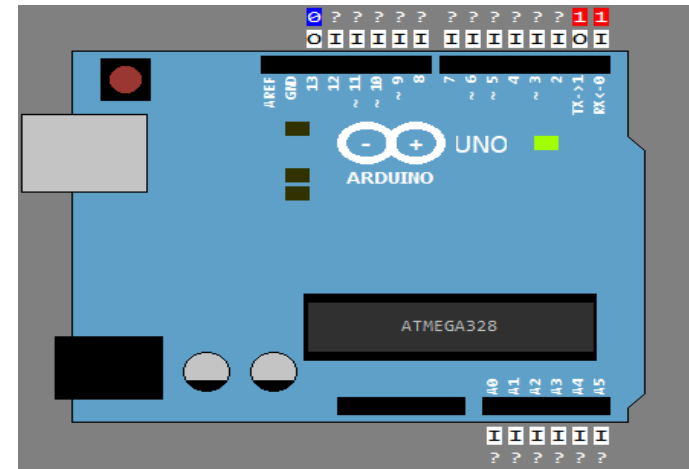
Click on (+) to expand, or on (-) to contract arrays and objects.

Use the **VarRefresh** menu to control update frequency when executing.

**Double-click** on any variable to track its value during execution, or to change it to a new value in the middle of (halted) program execution:

Or **single-click** to highlight any variable (or object-member, or array-element), then use **Run-Till** to advance execution up to the next **write-access** to that variable or location.

## Lab Bench Pane and the 'Uno':



**Left-click** on any pin to create (or add to) Pin Digital Waveforms:

**Right-click** on any pin to create a Pin Analog Waveform window:

To **ZOOM IN** and **ZOOM OUT** use the mouse wheel, or shortcuts **CTRL-up-arrow** and **CTRL-down-arrow**.

Type '**Ctrl-S**' to save the waveform (**X,Y**) points to a text file ('**X**' is microseconds from the left, '**Y**' is volts)

## Lab Bench Pane 'I/O' Devices

Set numbers and types of each using Configure | 'I/O' Devices .

Set pins using a 2-digit value from 00 to 19 (or A0-A5).

Several of these devices support scaling of their typed-in values using the slider on the main window Tool-Bar (see 'I/O \_\_\_\_S' under each of those devices below):



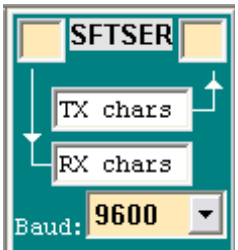
### 'Serial' Monitor ( 'SERIAL' )



Type one or more characters in the upper ('TX chars') edit box and **hit Return**.

Double-click (or right-click) to open **a larger window for TX and RX characters**.

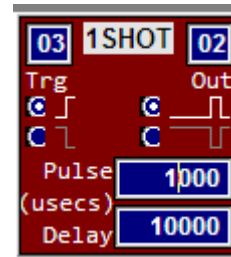
### Software Serial ( 'SFTSER' )



Type one or more characters in the upper ('TX chars') edit box and **hit Return**.

Double-click (or right-click) to open **a larger window for TX and RX characters**.

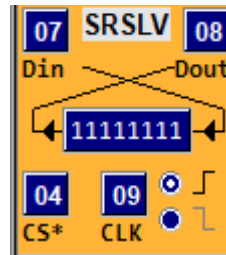
### One-Shot ( '1SHOT' )



A digital one-shot. Produces a pulse of chosen polarity on '**Out**' after a specified delay from either a rising or a falling triggering edge seen on its '**Trg**' input. Once triggered, it will ignore subsequent trigger edges until the pulse on '**Out**' has been fully completed.

'Pulse' and '**Delay**' values (if suffixed with an 'S'). will be scaled from the toolbar 'I/O \_\_\_\_S' slider

### Shift Register Slave ( 'SRSLV' )

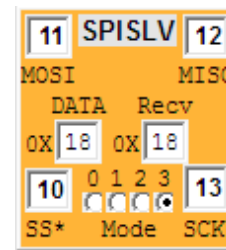


A simple shift-register device.

Edge transitions on CLK will trigger shifting.

SS\* low, drives MSB onto Dout.

### SPI Slave ( 'SPISLV' )

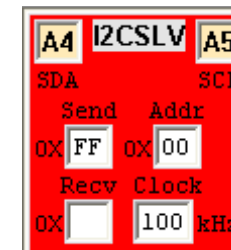


A mode-Configurable SPI slave device ('MODE0', 'MODE1', 'MODE2', or 'MODE3')

Double-click (or right-click) to open **a larger window** to set/view hex '**DATA**' and '**Recv**' bytes.

SS\* low, drives MSB onto MISO.

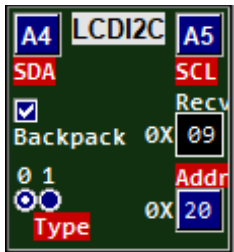
### Two-Wire I2C Slave ( 'I2CSLV' )



A slave-mode-only I2C device.

Double-click (or right-click) to open **a larger window** to set/view hex '**Send**' and '**Recv**' bytes

## Text LCD I2C ('LCDI2C')



A 1,2, or 4-line character-LCD, in one of three modes (2 backpack styles, plus a native mode), with supporting library code for each device mode provided inside the 'include\_3rdParty' folder .

Double-click (or right-click) to open a larger window to see the LCD screen (and sets it size)

## Text LCD SPI ('LCDSPI')

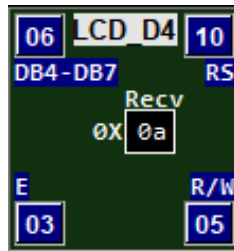


A 1,2, or 4-line character-LCD, in one of two modes (a backpack style, plus a native mode), with supporting library code for each device mode provided inside the 'include\_3rdParty' folder .

Double-click (or right-click) to open a larger window to see the LCD screen (and sets it size)

## Text LCD D4 ('LCD\_D4')

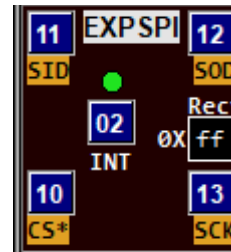
A 1,2, or 4-line character-LCD, in one of two modes (a backpack style, plus a native mode), with supporting library code for each device mode provided inside the 'include\_3rdParty' folder .



Double-click (or right-click) to open a larger window to see the LCD screen (and sets it size)



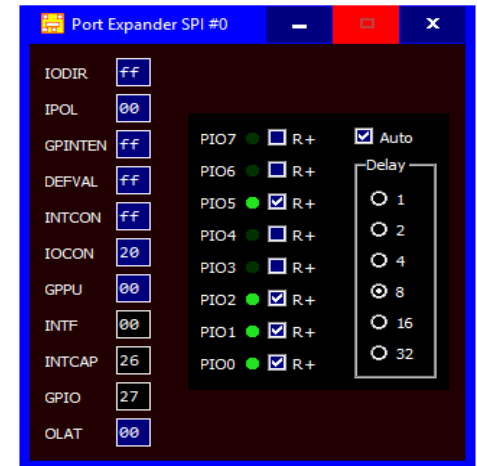
## Expansion Port SPI ('EXPSPi')



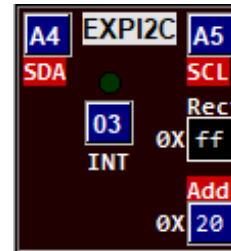
An 8-bit port expander based on the MCP23008, with supporting 'MCP23008.h' code provided inside the 'include\_3rdParty' folder. You can write to MCP23008 registers, and read back the GPIO pin levels. Interrupts can be enabled on each GPIO pin change – a triggered interrupt will drive the 'INT' pin.

Double-click (or right-click)

to open a larger window to see the 8 GPIO port lines, and the attached pull-up resistors. You can change pull-ups manually by clicking, or attach a counter that will periodically change them in a up-count manner.

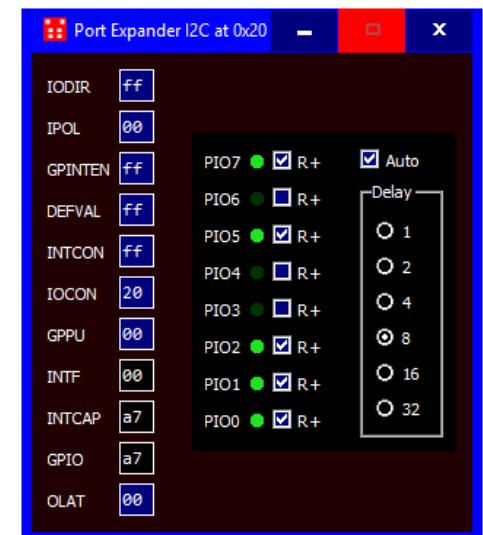


## Expansion Port I2C ('EXPI2C')

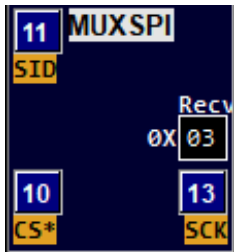


An 8-bit port expander based on the MCP23008, with supporting 'MCP23008.h' code provided inside the 'include\_3rdParty' folder. Capabilities match the 'EXPSPi' device.

Double-click (or right-click) to open a larger window as from the 'EXPSPi' device.

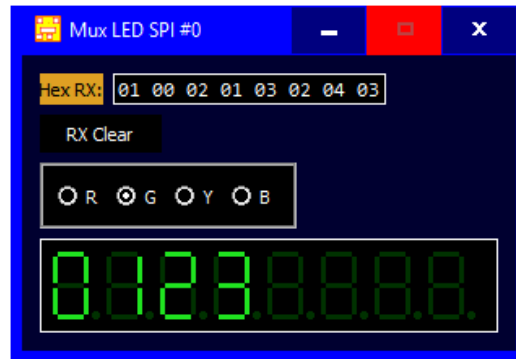


## Mux LED SPI ('MUXSPI')

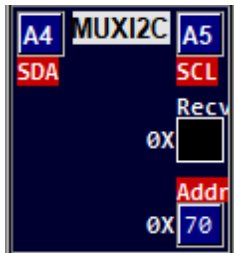


A multiplexed-LED controller based on the MAX6219, with supporting 'MAX7219.h' code provided inside the 'include\_3rdParty' folder to drive up to eight 7-segment digits.

Double-click (or right-click) to open a larger window to view the colored 7-segment-digit display.

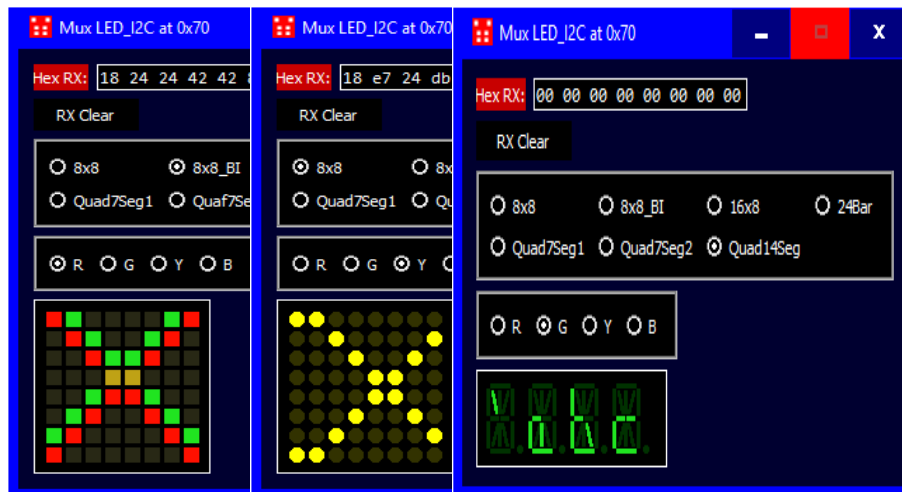


## Mux LED I2C ('MUXI2C')

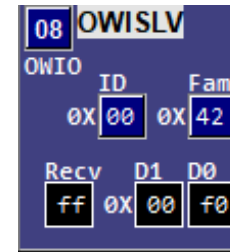


A multiplexed-LED controller based on the HT16K33, with supporting Adafruit\_LEDBackpack.h code provided inside the 'include\_3rdParty' folder.

Double-click (or right-click) to open a larger window to choose and view one of several colored LED displays.

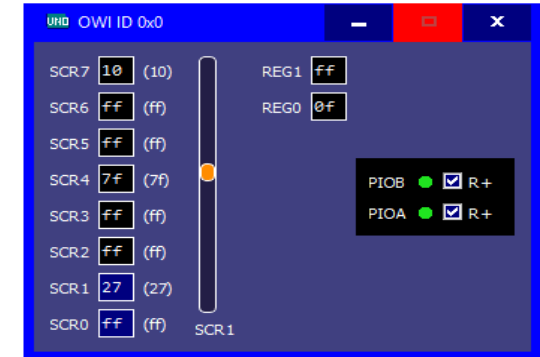


## '1-Wire' Slave ('OWISLV')



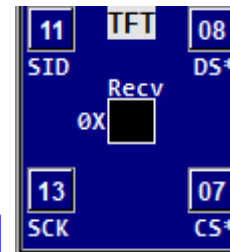
A slave-mode-only I2C device.

Double-click (or right-click) to open a larger window to set/view internal registers and parallel IO pins. You can change IO pull-up resistors manually by clicking, or attach a counter that will periodically change them in a up-count manner.



## TFT Display ('TFT')

An Adafruit™ thin-film-transistor LCD display of 128-by-160 pixels driven from the 'SPI' bus.

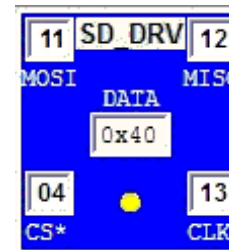


The 'DS\*' pin is data/command select, and the 'CS\*' pin is the active-low chip-select. There is no Reset pin provided but system Reset resets it..

Double-click (or right-click) to open a larger window to see the actual TFT display screen.

## SD Disk Drive ('SD DRV')

A small 8-Mbyte SD disk driven from SPI signals, and mirrored in an 'SD' subdirectory in the directory of the loaded program (an 'SD' sub-directory will be created if absent).



Double-click (or right-click) to open a larger window to see Directories, Files, and content.

CS\* low to activate



### Stepper Motor ( 'STEPR' )



Accepts coil control signals *on either 2 or 4 pins. 'Steps' must be a multiple of 4.*

Use `'#include <Stepper.h>'`.

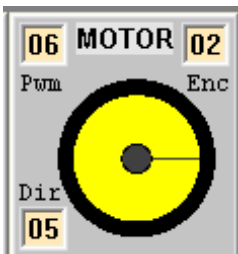
To emulate gear reduction by N in your program , use a modulo-N counter to determine when to actually call `'Stepper.step ()'`

### Pulsed Stepper Motor ( 'PSTEPR' )



Each rising edge on **'STEP'** causes one (micro)step in the direction controlled by **'DIR'** when enabled by a low on **'EN'**. **'Steps' must be a multiple of 4**, and **'micro'** must be **1,2,4,8, or 16** micro-steps *per full step*.

### DC Motor ( 'MOTOR' )



Accepts PWM signals on **Pwm** pin, level signal on **Dir**, and outputs 8 highs and 8 lows per wheel revolution on **Enc**.

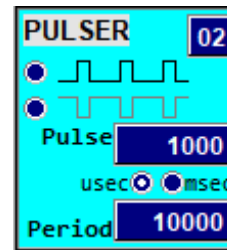
Full speed is approximately 2 revs per second.

### Servo Motor ( 'SERVO' )



Accepts pulsed control signals on specified pin. Can be modified to become continuous-rotation by checking the lower left check-box

### Digital Pulser ( 'PULSER' )



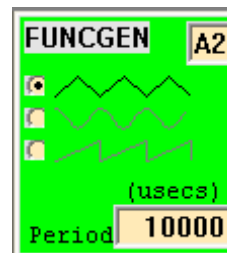
Generates digital waveform signals on specified pin.

Choose time base in milliseconds ('msec') or microseconds ('usec')

Minimum period is 50 microseconds, minimum pulse width 10 microseconds. Both values (if suffixed with an 'S'). will be scaled from the toolbar 'I/O \_\_\_\_S' slider

Choose positive-going pulses (0 to 5V) or negative-going pulses (5V to 0V).

### Analog Function Generator ( 'FUNCGEN' )

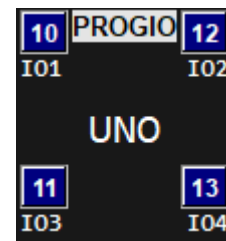


Generates analog waveform signals on specified pin.

Minimum 'Period' is 100 microseconds, scaled from the toolbar 'I/O \_\_\_\_S' slider (if suffixed with an 'S').

Sinusoidal, triangular, or sawtooth waveforms.

### Programmable 'I/O' Device ( 'PROGIO' )



A bare 'Uno' board that you can program (with a separate program) in order to emulate an 'I/O' device whose behaviour you completely define.

This slave 'Uno' can have no **'I/O'** devices of its own – it can only share up to 4 pins (IO1, IO2, IO3, and IO4) in common with the master 'Uno' that sits in the main window **Lab Bench Pane**.

**Right-click** (or **double-click**) on it to open a larger window showing its **Code Pane** and **Variables Pane**. Use **File|Load** to load a new program into this 'Uno' slave – its execution always remains synchronized to that of the master 'Uno'.

**After clicking inside its Code Pane** , you can even use **File|Execute** to **Step** or **Run-To** or **Run-Till** inside its slave program (the master 'Uno' will execute just enough to stay in sync).

### Piezo Speaker ( 'PIEZO')

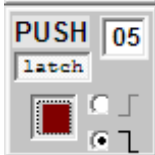


"Listen" to signals on any chosen 'Uno' pin.

### Push Button ( 'PUSH')



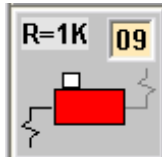
A normally-open **momentary** push-button to +5V or ground



A normally-open **latching** push-button to +5V or ground (depress "latch" button too get this mode).

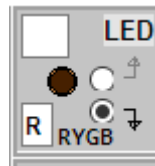
You can close the push-button by clicking it. or by pressing any keyboard key – contact bouncing will only be produced if you use the **space-bar** key.

### Slide Resistor ( 'R=1K')



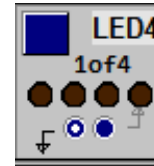
A 1 k-Ohm pull-up to +5V OR a 1 k-Ohm pull-down to ground.

### Coloured LED ( 'LED')



R,Y,G, or B LED connected between any chosen 'Uno' pin and either ground or +5V.

### 4-LED Row ( 'LED4')



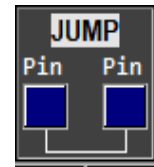
R,Y,G, or B row of 4 LEDs connected between **four consecutive** 'Uno' pins and either ground or +5V. The supplied **1of4** pin number corresponds to the left-most LED.

### 7-Segment LED Digit ( '7SEG')



A 7-LED\_segment coloured digit. The supplied **1of4** pin number represents the first of **four consecutive** 'Uno' pins. The active-HIGH levels on these 4 pins define the hexadecimal code for the desired display digit ('0' through 'F'), where the lowest pin number corresponds to the least-significant bit of the hexadecimal code.

### Pin Jumper ( 'JUMP')



Allows you to connect two 'Uno' pins together as long as that does not create an electrical conflict.

See the Full Help file for possible uses for this device (most of those involve interrupts)

### Analog Slider





A slider-controlled potentiometer. 0-5V to drive any chosen 'Uno' pin.





## Menus






### File:

<b><u>Load INO or PDE Prog</u></b> 	Allows the user to choose a program file having the selected extension. The program is immediately parsed
<b><u>Edit/View</u></b>	Opens the loaded program for viewing/editing.
<b><u>Save</u></b> 	Save the edited program contents back to the original program file.
<b><u>Save As</u></b>	Save the edited program contents under a different file name.
<b><u>Next ('#include')</u></b> 	Advances the Code Pane to display the next '#include' file
<b><u>Previous</u></b> 	Returns the Code Pane display to the previous file
<b><u>Exit</u></b>	Exits UnoArduSim.









### Configure:

<b><u>'I/O' Devices</u></b>	Choose desired number of each type of device (8 large, and 16 small, 'I/O' devices are allowed)
<b><u>Preferences</u></b>	Choose automatic indentation, font typeface, optional larger type size, expert syntax, keyword logical operators, enforcing array bounds, showing download, 'Uno' board version, and TWI buffer length

### Find :

<b><u>Ascend Call Stack</u></b> 	Jump to the previous caller function in the call-stack – the <b>Variables Pane</b> will adjust to show that functions locals
<b><u>Descend Call Stack</u></b> 	Jump to the next called function in the call-stack – the <b>Variables Pane</b> will adjust to show that functions locals
<b><u>Set Search Text (ctrl-F)</u></b> 	Activate toolbar Find edit box to define your next-to-be-searched-for text..
<b><u>Find Next Text</u></b> 	Jump to the next Text occurrence in the Code Pane (if it has the active focus), or to the next Text occurrence in the Variables Pane (if instead it has the active focus).
<b><u>Find Previous Text</u></b> 	Jump to the previous Text occurrence in the Code Pane (if it has the active focus), or to the previous Text occurrence in the Variables Pane (if instead it has the active focus).

## Execute:

<b><u>Step-Into (F4)</u></b>		Steps execution forward by one instruction, or <i>into a called function</i> .
<b><u>Step-Over (F5)</u></b>		Steps execution forward by one instruction, or <i>by one complete function call</i> .
<b><u>Step-Out-Of (F6)</u></b>		Advances execution by <i>just enough to leave the current function</i> .
<b><u>Run-To (F7)</u></b>		Runs the program, <i>halting at the desired program line</i> -- you must first click to highlight a desired program line before using Run-To.
<b><u>Run-Till (F8)</u></b>		Runs the program, <i>halting when the highlighted Variables Pane variable location is next written to</i> (click to highlight a desired item before using Run-Till).
<b><u>Run (F9)</u></b>		Runs the program.
<b><u>Halt (F10)</u></b>		Halts program execution ( <i>and freezes time</i> ).
<b><u>Reset</u></b>		Resets the program (all value-variables are reset to value 0, and all pointer variables are reset to 0x0000).
<b><u>Animate</u></b>		Automatically steps consecutive program lines <i>with added artificial delay</i> and highlighting of the current code line.
<b><u>Slow Motion</u></b>		Slows time by a factor of 10.

## Options:

<b><u>Step-Over Structors/Operators</u></b>	Fly right through constructors, destructors, and operator overload function during any stepping (i.e. it will not stop inside these functions).
<b><u>Register-Allocation Modelling</u></b>	Assign function locals to free ATmega registers instead of to the stack..
<b><u>Added loop() Delay</u></b>	Add 1 millisecond. (by default) to each call to <code>loop()</code> (in case user has not added any delays anywhere)
<b><u>Error on Uninitialized</u></b>	Flag as a Parse error anywhere your program attempts to use a variable without having first initialized its value.
<b><u>Show Program Download</u></b>	Show program download to the 'Uno' board (with attendant delay).
<b><u>Allow Nested Interrupts</u></b>	Allow re-enabling using ' <code>interrupts.()</code> ' from inside a user interrupt service routine.

## Configure menu commands:

<b><u>'I/O' Devices</u></b>	Choose desired number of each type of device (8 large, and 16 small, 'I/O' devices are allowed)
<b><u>Preferences</u></b>	Choose automatic indentation, font typeface, optional larger type size, expert syntax, keyword logical operators, enforcing array bounds, showing download, tab size multiplier, 'Uno' board version, TWI buffer length

## VarRefresh:

<b><u>Allow Auto (-) Contract</u></b>	Allow UnoArduSim to contract displayed expanded arrays/structs/objects when falling behind real-time.
<b><u>Minimal</u></b>	Only refresh the Variables Pane display 4 times per second.
<b><u>HighLight Updates</u></b>	Highlight the last-changed variable value (can cause slowdown).

## Help menu commands:

<b><u>Quick Help File</u></b>	Opens the UnoArduSim_QuickHelp PDF file.
<b><u>Full Help File</u></b>	Opens the UnoArduSim_FullHelp PDF file.
<b><u>Bug Fixes</u></b>	View significant bug fixes since the previous release..
<b><u>Changes/Improvements</u></b>	View significant changes and improvements since the previous release.
<b><u>About</u></b>	Displays version, copyright

## Windows:

<b><u>'Serial' Monitor</u></b>	Add a serial IO device (if none) and pull up a larger 'Serial' monitor TX/RX text window.
<b><u>Restore All</u></b>	Restore all minimized child windows.
<b><u>Pin Digital Waveforms</u></b>	Restore a minimized Pin Digital Waveforms window.
<b><u>Pin Analog Waveform</u></b>	Restore a minimized Pin Analog Waveform window.